ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ “ВЫСШАЯ ШКОЛА ЭКОНОМИКИ” (НИУ ВШЭ)

На правах рукописи

Бекларян Армен Левонович

АГЕНТНАЯ МОДЕЛЬ ПОВЕДЕНИЯ ТОЛПЫ В УСЛОВИЯХ ЧРЕЗВЫЧАЙНОЙ СИТУАЦИИ ДЛЯ ОЦЕНКИ ИНТЕНСИВНОСТИ ФРОНТА ВЫХОДНОГО ПОТОКА

Специальность 05.13.18 —
Математическое моделирование, численные методы и комплексы программ

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
доктор технических наук, доцент
Акопов Андраник Сумбатович

Москва — 2016

Оглавление

Введение 5

Раздел 1. Агентная модель поведения толпы 12
1.1 Особенности моделирования поведения толпы 13
1.1.1 Анализ существующих моделей 13
1.1.2 Требования к предлагаемой агентной модели 18
1.2 Модель поведения толпы при отсутствии чрезвычайной ситуации . 23
1.2.1 Описание пространства, состояния агентов и правил принятия решений в случае отсутствия их взаимодействий . 23
1.2.2 Правила взаимодействия агентов 35
1.3 Модель поведения толпы при возникновении чрезвычайной ситуации 37
1.4 Разработка алгоритма эволюционной кластеризации динамики толпы для снижения размерности задачи 40
1.4.1 Модификация алгоритма нечеткой кластеризации 42
1.4.2 Основные результаты кластерного анализа 51
1.5 Выводы 52

Раздел 2. Разработка программного комплекса для реализации агентной имитационной модели поведения толпы и эволюционного алгоритма нечеткой кластеризации 54
2.1 Подготовка программной реализации 54
2.1.1 Выбор среды агентного моделирования 54
2.1.2 Разработка базы данных 56
2.2 Разработка агентной модели толпы в среде AnyLogic 64
2.3 Архитектура программного комплекса 69
2.4 Выводы 74

Раздел 3. “Фронт выхода” толпы и оценка интенсивности потока . . . 75
3.1 Понятие “фронта выхода”. Уравнение максимальной интенсивности потока и порождаемая краевая задача 75
3.2 Теорема существования для краевой задачи 76

3.2.1 Основные понятия и предварительные конструкции 76
3.2.2 Постановка задачи 79
3.2.3 Основные результаты 79
3.2.4 Уточненные оценки емкости 100
3.3 Выводы 110

Заключение 111

Список сокращений и условных обозначений 112

Словарь терминов 113

Список литературы 114

Список рисунков 124

Список таблиц 126

Приложение А. Внедрение результатов 127
А.1 Справка о внедрении результатов 127

Приложение Б. Пользовательский интерфейс модели 128
Б.1 Скрин пользовательского интерфейса 128
Б.2 Листинг кода пользовательского интерфейса 129

Приложение В. Основной класс модели 161
В.1 Скрин основного класса 161
В.2 Листинг кода основного класса 161
В.3 Скрин окна статистики 251

Приложение Г. Класс агента 252
Г.1 Скрин класса агента 252
Г.2 Листинг кода класса агента 252

Приложение Д. Класс взрыва 276
Д.1 Скрин класса взрыва 276
Д.2 Листинг кода класса взрыва 276

Приложение Е. Класс столкновений 284

Е.1 Листинг кода класса столкновений 284

Приложение Ж. Класс принятия решения 286
Ж.1 Листинг кода класса принятия решения 286

Приложение З. Класс точек пространства 294
З.1 Листинг кода класса точек пространства 294

Приложение И. Класс векторов пространства 296
И.1 Листинг кода класса векторов пространства 296

Приложение К. Класс результатов 301
К.1 Листинг кода класса результатов 301

Введение

Актуальность темы. Коллективное поведение людей в замкнутом пространстве таит в себе формы поведения, опасные для жизни человека. Особая роль отводится ситуациям, при которых возникает массовая паника, например, вследствие возникновения чрезвычайной ситуации (далее, ЧС). Так, в результате пожара в ночном клубе «Хромая лошадь» (5 декабря 2009 года) погибло 156 человек и 64 человека получили тяжкий вред здоровью [46]. Давка, произошедшая 22 ноября 2010 года в столице Камбоджи во время традиционного камбоджийского праздника – фестиваля воды, – повлекла за собой гибель 456 человек, ещё более пятисот получили ранения различной степени тяжести [51]. При этом стоит заметить, что во многих ситуациях основные людские потери возникают не столько в сам момент возникновения ЧС, а являются следствиями дальнейших событий (задымление, эффект толпы, давка и т.д.), а также зависят от характеристик внешней системы (геометрия помещения, расположение выходов и т.д.), существенно влияющих на возможность эффективной эвакуации. Таким образом, паника и дальнейшая давка многократно увеличивают число жертв среди людей даже в ситуациях, напрямую не угрожающих жизни.

К сожалению, последствия ЧС являются трудно прогнозируемыми, так как зависят от множества факторов. Кроме того, большая часть наблюдений за местами скопления людей, и тем более за процессом поведения толпы в той или иной ЧС, относятся либо к закрытой информации, либо, как минимум, к трудно доступной. Не говоря уже о том, что само множество однотипных ЧС статистически мало и не дает возможности построения точной аналитической модели. Тем не менее, при наличии максимально “гибкого” инструмента моделирования поведения, где изменение параметров модели позволяет имитировать поведение разных типов агентов, появляется возможность как подготовки лиц, ответственных за снижение числа пострадавших при возникновении ЧС, так и повышения степени идентичности результатов моделирования поведения толпы и реальных процессов.

Все вышесказанное обуславливает актуальность темы исследования.

Диссертация выполнена на кафедре бизнес-аналитики школы бизнес-информатики факультета бизнеса и менеджмента НИУ ВШЭ.

Степень разработанности темы. Несмотря на высокий интерес к проблематике, долгое время основные работы по данной теме были посвящены психологическим и социальным аспектам вопроса. Так, например в [88], детально описаны условия и причины возникновения паники, которые сводятся к доминированию коллективного бессознательного как основного фактора. То есть солидная часть исследователей рассматривает толпу с фрейдистской точки зрения, основанной на гипотезе, что люди как часть толпы действуют иначе, чем люди как индивиды [29; 49; 53]. Совокупность разумов членов группы синергируются в некий коллективный разум. Соответственно, и предлагаемые решения проблемы возникновения паники также основаны на данном подходе [7], который мы назовем *наивным*.

На фоне описанных исследований, изучение толпы с привлечением математических моделей сложных систем началось сравнительно недавно. Здесь стоит отметить работы пионера этой области — Дирка Хелбинга. В его работе 2000 года в журнале *Nature* [79] впервые удалось воспроизвести ряд характерных для толпы явлений, таких, как образование пробок, вовлечение новых людей в панику и другие, с помощью математического моделирования. В основе этой работы лежала идея применения к толпе людей методов молекулярной динамики, где психологические и социальные факторы рассматриваются как потенциалы взаимодействия между молекулами-людьми [81]. Такой подход будем называть *молекулярным*. На основе модели Хелбинга были построены ряд других моделей [8; 97], рассматривающие различные аспекты возможных усложнений системы взаимодействий. Правда, основная часть моделей основывается на двухчастичном взаимодействии и игнорирует тот факт, что в определенной точке пространства сталкиваются трое и более людей. Тем не менее, в работе [89] была рассмотрена модель многочастичного взаимодействия, которая привела к появлению модельного эффекта турбулентности толпы, который не раз был зарегистрирован в реальных ситуациях. Здесь стоит упомянуть работы группы российских ученых (Д.А. Брацун и др.) [9; 64], ставящие своей целью создание модели поведения толпы на основе моделей Хелбинга, отличительной особенностью которых является сложность геометрии пространства и формирование индивидумом плана выхода из многоуровнего разветвленного помещения. К сожалению, дальнейшее усложнение моделей Хелбинга, как в части взаимодействия людей, так и в части анализа окружающей обстановки, ведет к громозд-

кой процедуре совместного интегрирования уравнений движений, что требует либо распараллеливания вычислительных процессов, либо сверхпроизводительных процессоров.

Наряду с двумя описанными подходами, в работе [56] был предложен *феноменологический* подход, в рамках которого формализована агентная модель поведения толпы. В такой агентной модели априори определяются состояния агентов с их характеристиками, правила взаимодействия агентов и правила принятия решений. Это позволяет смоделировать динамику состояния системы как результат взаимодействия автономных агентов, чья система принятия решений задается в явном виде, а не является результатом решения системы уравнений Ньютона. При этом удается заложить такие эффекты как турбулентность толпы, волны сжатия толпы и другие, которые в рамках моделей Хелбинга требуют задания соответствующих потенциалов, что, в свою очередь, ведет к поиску уникального динамического решения для весьма сложной системы уравнений и порождает самостоятельную неординарную задачу. Также стоит отметить, что при феноменологическом подходе удается добавить ряд стохастических процессов в систему принятия решений агента с целью приближения моделируемой динамики к реально наблюдаемым случайным флуктуациям в поведении толпы.

Среди других работ, нацеленных на создание программно-графического пакета реализации человеческого поведения, особое место занимает коммерческий продукт DI–Guy [38]. Как следует из официального релиза компании, основным заказчиком программного продукта является министерство обороны США и крупнейшие военно-промышленные корпорации (Boeing, BAE Systems, Raytheon). То есть создание систем прогнозирования человеческого поведения также является предметом коммерциализации и представляет большой интерес как для государственных структур, так и для частных компаний.

Целью данной работы является разработка агентной модели поведения толпы для оценки интенсивности потока на фронте выхода и прогнозирования динамики развития ЧС с реализацией в виде программного комплекса.

Для достижения поставленной цели сформулированы следующие **задачи** исследования:

1. Создание имитационной агентной модели поведения толпы, основанной на индивидуальной системе принятия решений агентов.

2. Разработка модифицированного алгоритма нечеткой кластеризации, агрегированного с имитационной моделью поведения толпы.
3. Проведение кластерного анализа с целью идентификации динамики возникающих кластеров.
4. Выявление зависимости динамики людских таксонов (кластеров) от параметров модели.
5. Синтез уравнения “фронта выхода” для модели поведения толпы.
6. Проектирование комплекса программ для реализации агентной модели поведения толпы, агрегированной с предложенным модифицированным алгоритмом нечеткой кластеризации.

В диссертационной работе получены следующие результаты, характеризующиеся **научной новизной:**

– Разработана имитационная агентная модель поведения толпы, новизной которой является учет как многофакторности системы принятия решений агентами, так и стохастичности ряда процессов, в частности, учет влияния факторов внешней среды (стены, другие агенты, препятствия, взрыв и т.д.) на систему принятия решений агентом, учет радиуса личного пространства агентов и эффекта турбулентности толпы, детальная параметризация начального распределения агентов.
– Определена максимальная интенсивность потока при эвакуации на основе решения соответствующей первой краевой задачи на “фронте выхода”.
– Разработан эволюционный алгоритм нечеткой кластеризации, учитывающий факт наличия препятствий на пути следования агента, с учетом текущего направления движения агента, позволяющий существенно улучшить точность идентификации таксонов (кластеров толпы) с целью повышения временной эффективности модели.
– Впервые получена аналитическая зависимость между параметрами модели и динамикой таксонов (кластеров) при эвакуации.
– Разработан комплекс программ, отличающийся интеграцией имитационной модели поведения толпы с эволюционным алгоритмом нечеткой кластеризации, модулем кластерного анализа и базой данных системы, а также обеспечивающий возможность дальнейшего развития модели за счет объектно-ориентированного подхода.

Теоретическая и практическая значимость. Разработанная агентная модель может служить основой для дальнейших усложнений, направленных на повышение степени идентичности результатов имитационного моделирования поведения толпы и реальных процессов. Результаты работы могут быть рекомендованы для формулировки требований к строительным нормам помещений с целью снижения числа пострадавших при возникновении ЧС. На основе полученного решения первой краевой задачи на “фронте выхода” разработаны рекомендации по повышению эффективности процесса эвакуации с целью максимизации числа спасенных агентов. Разработанное программное средство эволюционной динамической кластеризации следует рекомендовать проектным и научно-исследовательским организациям в качестве инструмента повышения временной эффективности модели.

Разработанный программный комплекс внедрен в компании ООО “ГЕНКЕЙ-ТЕЛЕКОМ” и используется при проектировании систем пожаробезопасности, в частности, для определения наилучших мест для установки систем видеонаблюдения за динамикой толпы, что позволило существенно повысить качество оценки интенсивности потока на выходе из помещения (приложение А.1).

Методология и методы исследования. При выполнении диссертационной работы применялись методы: дискретной математики, теории вероятностей и математической статистики, численных методов, аналитической геометрии, дифференциальных уравнений, современных технологий программирования (ООП, Java, R) и имитационного моделирования (AnyLogic).

Основные положения, выносимые на защиту:

1. Имитационная агентная модель поведения толпы, основанная на индивидуальной системе принятия решений агентами.
2. Теорема существования решения краевой задачи на “фронте выхода” как основа для создания инструмента оценки характеристик процедуры эвакуации при ЧС.
3. Результаты анализа динамики системы, полученные с помощью имитационного моделирования, в зависимости от параметров модели как основа принятия эффективных решений при управлении поведением толпы.
4. Эволюционный алгоритм нечеткой кластеризации, позволяющий снизить размерность задачи и повысить временную эффективность модели.

5. Установленные зависимости между динамикой людских таксонов (кластеров) и параметрами модели для выработки управленческих решений, повышающих эффективность управления рисками.
6. Оригинальный программный комплекс для проведения симуляций динамики толпы.

Достоверность изложенных в работе результатов подтверждается теоретическими выкладками, а также результатами численного моделирования и экспериментальных исследований на модели в системе имитационного моделирования AnyLogic.

Апробация работы. Основные результаты работы докладывались на:

1. Семинаре лаборатории социального моделирования (ЦЭМИ РАН, 25 февраля 2015). Тема доклада: “Агентная модель поведения толпы при чрезвычайных ситуациях”.
2. Семинаре отдела математического моделирования экономических систем (ВЦ РАН, 18 марта 2015). Тема доклада: “Феноменологическая модель поведения толпы в чрезвычайной ситуации”.
3. XVI Апрельской международной научной конференции «Модернизация экономики и общества» (НИУ ВШЭ, 9 апреля 2015). Тема доклада: “Агентная модель поведения толпы при чрезвычайных ситуациях”.
4. Научно-исследовательском семинаре “Многомерный статистический анализ и вероятностное моделирование реальных процессов” (ЦЭМИ РАН, 15 апреля 2015). Тема доклада: “Феноменологическая модель поведения толпы в чрезвычайной ситуации”.
5. Международной конференции «Устойчивость и процессы управления» (СПбГУ, 5 октября 2015). Тема доклада: “Агентно-ориентированная модель поведения толпы в чрезвычайной ситуации”.
6. Международной конференции «КОЛМОГОРОВСКИЕ ЧТЕНИЯ – VII. Общие проблемы управления и их приложения» (ТГУ им. Г.Р. Державина, 20 октября 2015). Тема доклада: “Фронт выхода в модели поведения толпы при чрезвычайных ситуациях”.
7. Научно-методическом семинаре “Математическое моделирование, численные методы и комплексы программ” (НИУ ВШЭ, 17 ноября 2015). Тема доклада: “Агентная модель поведения толпы в условиях чрезвычайной ситуации”.

8. Общемосковском семинаре “Экспертные оценки и анализ данных” (ИПУ РАН, 10 февраля 2016). Тема доклада: “Агентная модель поведения толпы в условиях чрезвычайной ситуации для оценки интенсивности фронта выходного потока”.

Личный вклад. Все представленные в диссертации результаты получены лично автором, а именно: создана имитационная модель поведения толпы в условиях чрезвычайной ситуации; разработан новый эволюционный алгоритм нечеткой кластеризации, агрегированный с данной имитационной моделью; спроектирован программный комплекс для проведения симуляций динамики толпы с использованием разработанной имитационной модели, эволюционного алгоритма и других компонентов, который затем был внедрен и успешно используется в действующей компании. Подготовка к публикации полученных результатов проводилась совместно с научным руководителем, причем вклад диссертанта был определяющим. Апробация результатов исследования на конференциях проводилась как автором самостоятельно, так и с участием научного руководителя.

Публикации. Основные результаты по теме диссертации изложены в 11 печатных изданиях, 6 из которых изданы в журналах, рекомендованных ВАК, 3 — в журналах, индексируемых в SCOPUS, 2 — в журналах, индексируемых в Web of Science, 5 — в тезисах докладов.

Объем и структура работы. Диссертационная работа изложена на 126 страницах (без приложений), включает 9 таблиц, 33 рисунка, 4 определения и 5 утверждений с доказательствами. Состоит из введения, трех глав, заключения, списка литературы из 100 наименований и 10 приложений.

Краткое содержание работы. Первая глава посвящена анализу существующих моделей, а также формальному описанию предлагаемой модели как в случае обычного поведения толпы, так и при ЧС. Дается описание применяемых алгоритмов кластеризации, дается их сравнительный анализ, описывается разработанный эволюционный алгоритм кластеризации, формулируются ряд утверждений, связанных с оценкой характеристик динамики кластеров. Во второй главе обоснован выбор среды моделирования, описана программная реализация и проводится калибровка модели. В третьей главе дано определение “фронта выхода” и его характеристик, изложены основные результаты в форме лемм и теорем с доказательствами.

Раздел 1. Агентная модель поведения толпы

Одна из ключевых составляющих жизнедеятельности человека, особенно в крупных городах, заключена в безопасности движения в условиях ограниченного пространства и большого скопления других движущихся людей. Подобная проблема становится особенно актуальной при пользовании общественным транспортом, при проведении культурно-массовых мероприятий, на митингах и при других неотъемлемых эпизодах повседневной жизни человека. Отдельно стоит отметить проблему эвакуации людей из зданий при чрезвычайных ситуациях (ЧС).

Очевидно, что проведение реальных экспериментов в рамках данной задачи требует огромного финансирования и сопряжено с большими организационными сложностями. Отсюда возникает необходимость математического моделирования описанных процессов с целью определения, как наилучшей геометрии помещений, так и возникновения устойчивых групп скоплений, что позволит максимально эффективно, с точки зрения количества спасенных людей, выстроить процесс ликвидации ЧС.

Толпа — бесструктурное скопление людей, лишенных ясно осознаваемой общности целей, но связанных между собой сходством эмоционального состояния и общим объектом внимания [23]. Основным фактором возникновения толпы и развития ее качеств является циркулирующая реакция (нарастающее обоюдонаправленное эмоциональное заражение) и вследствие этого эффект *притяжения* человека (агента) к группе людей.

Ключевыми особенностями толпы являются эффекты типа давка, паника, турбулентность.

Нарастающая плотность агентов по достижению критического значения приводит к появлению *давки*, которая в состоянии спровоцировать возникновение *паники*. Сама паника характеризуется стремлением агента расширить свое личное пространство, что приводит к возникновению эффекта *турбулентности толпы* [72; 78]. При этом в толпе образуются *волны сжатия*, выталкивающие агентов на опасные участки помещения (в частности, на стены).

В результате учета имеющихся моделей и подходов, а также исследований по психологии толпы, были сформулированы основные априорные предположе-

ния, которые легли в основу данной агентной модели поведения при ЧС, среди которых стоит отметить:

- частичная или полная потеря ориентации в пространстве и во времени;
- высокая степень турбулентности толпы, т.е. наличие хаотичного движения во всех направлениях в условиях высокой плотности агентов;
- существенное замедление скорости передвижения при определенных условиях (ранение, уплотнение и т.д.);
- стремление к ближайшему выходу в случае нахождения выхода в пределах видимости;
- стремление к присоединению к ближайшей группе агентов (эффект *притяжения толпы*).

1.1 Особенности моделирования поведения толпы

1.1.1 Анализ существующих моделей

В процессе моделирования поведения толпы, и тем более при попытке моделирования ЧС, сопряженной с паникой и другими сложными внутренними процессами, возникает проблема невозможности полного математического описания поведения индивида, поскольку действия и сама система принятия решений человека определяется большим количеством факторов, как физического, так и психо-социального характера. Также каждому человеку в разной степени свойственна иррациональность принимаемых решений. Все это обуславливает некоторую степень “усреднения” моделируемых процессов. Однако стоит отметить, что в основе подобного подхода лежит разумная гипотеза о применимости к толпе закона больших чисел: нетипичное поведение индивида не влияет на качественную картину поведения всей толпы.

Ввиду неуклонного роста актуальности данной тематики, математическому моделированию поведения толпы уделено большое внимание в научной литературе и с каждым годом (видимо, вследствие роста вероятности возникновения ЧС по всему миру) подобный интерес только возрастает. Анализ существующих

работ показал, что большая часть исследований следует отнести к трем ключевым подходам:

1. *наивный*, основанный на психологических и социальных аспектах вопроса [29; 49; 88], рассматривающий толпу с фрейдистской точки зрения;
2. *молекулярный*, представленный в основном, так называемыми, моделями Хелбинга [9; 79], в основе которых лежит идея применения к толпе людей методов молекулярной динамики, где дается интерпретация психологическим и социальным факторам в форме потенциалов взаимодействия между молекулами-людьми;
3. *феноменологический*, предложенный, в частности, в работах Бекларяна-Акопова [6; 56] (которые положены в основу предлагаемой модели), в рамках которого априори определяются состояния агентов с их характеристиками, правила взаимодействия агентов и правила принятия решений.

Молекулярный подход предполагает, что поведение человека описывается дифференциальным уравнением, соответствующему второму закону Ньютона со сложной правой частью, в которой аккумулированы силы взаимодействия индивида с другими людьми и препятствиями. В качестве достоинства таких моделей необходимо отметить учет всех заложенных характеристик индивида в динамике, а также возможность неограниченного усложнения процессов взаимодействия путем добавления новой вектор-силы. Однако численное интегрирование подобной модели требует в среднем $O(N^2)$ [82] вычислительных операций (N – количество людей) для каждого временного шага, что даже при современных вычислительных возможностях является весьма трудоемкой задачей. Тем более, что в погоне за точностью и устойчивостью решений, авторы прибегают к методам Рунге-Кутты высокого порядка и их модификациям (например, метод Рунге-Кутты-Фельберга) [11], что, в свою очередь, еще более усложняет вычислительный процесс.

В рамках феноменологического подхода существует ряд подклассов моделей таких, как непрерывные агентные, дискретные и вероятностные. Дискретные модели (например, [24; 48]) предполагают дискретизацию пространства, а само помещение рассматривается как поле клеточного автомата. При таком подходе клетка, занимаемая индивидуумом, суть конечный автомат, состояние кото-

рого зависит от состояния его соседей. Подобная интерпретация задачи, с одной стороны, отличается быстротой вычислительных процессов (сложность $O(N)$), но с другой характеризуется чрезмерным “усреднением” сил взаимодействия.

Вероятностный класс моделей (например, [19; 20; 27]) сравнивает личное априорное (до общения с коллективом) отношение к некому состоянию и соответствующая вероятность того, что индивид готов перейти в это состояние, и финальное апостериорное отношение, сформировавшееся после общения с коллективом. Подобные модели достаточно точно описывают коллективное поведение в рамках условных “игровых” процессов таких, как митинг, переговоры, выборы, парламент и др., а также дают оценки степени конформизма и, наоборот, индивидуализма агентов. Тем не менее, применение таких моделей к процессу перемещения в пространстве (а не к осуществлению того или иного выбора) видится затруднительным.

Класс агентных моделей представлен широким списком работ (например, [70], [72], [74]) и является самой естественной реализацией феноменологического подхода. Тем не менее, доля российских исследований в данной тематике непростительно мала. Анализ существующих работ показал, что значительная часть исследований в рамках агентного моделирования поведения толпы вылились в полноценные коммерческие проекты, среди которых стоит отметить такие продукты как DI–Guy [38], Myriad II [39], Pedestrian Dynamics [40]. Как следует из официальных релизов указанных компаний, основными заказчиками программных продуктов являются министерство обороны США и крупнейшие военно-промышленные корпорации. То есть создание систем прогнозирования человеческого поведения также является предметом коммерциализации и представляет большой интерес как для государственных структур, так и для частных компаний. При этом, очевидно, что исходные коды продуктов, как и ключевые принципы, заложенные в основу моделей, являются объектами коммерческой тайны и интеллектуальной собственности.

Несмотря на острую необходимость большого объема финансирования для построения и инициализации полноценных агентных моделей (вычислительные мейнфреймы, 3D моделирование, доступ к видео записям и распознавание образов и пр.), работы академического характера по данной тематике также имеют место. Например, в работе [94] рассматривается агентная модель поведения толпы при пожаре в концертном зале. В рамках такой модели каждый агент исходит

из близости выхода, его достижимости (выход не перекрыт пожаром), а также из стремления сохранить прямолинейную траекторию движения. При этом факторы взаимодействия агентов (в частности, столкновения) в модели не рассматриваются. В работах [57;71], также посвященных эвакуации при пожаре, дополнительно вводятся такие параметры как состояния агента (степень удушья), оценка агентом ситуации (паника или нормальное состояние), а также появляется гендерно-возрастная зависимость ряда характеристик.

В статье [70] система принятия решений агента основана на “плотности” траектории движения и времени необходимом на ее преодоление. Указанная модель примечательна таким дополнительным фактором, как степень информированности агента об окружающей ситуации. Данный фактор меняется со временем как за счет анализа ситуации самим агентом, так и за счет других источников информации (звонок от другого агента, данные из интернета и пр.).

Также нельзя не упомянуть про проект Gamma на базе Университета Северной Каролины [41], поддерживаемый рядом крупных спонсоров, среди которых министерство обороны США, а также компании Intel и Google. В рамках данного проекта группа ученых и программистов разработали модульный фреймворк Menge [42], в котором реализованы наработки коллектива для таких моделей поведения толпы, как модель сверхплотной толпы [90], модель энергозатрат агента вдоль траектории движения [75], модель психотипа агента [74], модель превентивного уклонения от столкновения [73], модель общего адаптационного синдрома [86] и др. Тем не менее, при том, что сам фреймворк является бесплатным, но исходные коды классов и интерфейсов разработчиками не предоставляются, а статьи, на основе которых был создан программный продукт, при всей ценности описания базовых принципов не предоставляют читателю явные вычислительные формулы. Данный факт, по-видимому, также напрямую коррелирует с коммерческой составляющей проекта. При этом, в части самой динамики движения агентов, в качестве дальнейшего развития, относительно предлагаемой в данной работе модели, автор видит именно уточнение расчетов траектории движения, согласно указанным выше работам.

Наиболее близкими в части системы принятия решений, к предлагаемой в данной диссертации модели, являются агентные модели, представленные в работах [65] и [58]. В обеих указанных моделях каждому агенту сопоставляется некий сектор обзора, в рамках которого агент выбирает дальнейший вектор сво-

его движения на основе функции полезности, которая, в свою очередь, отражает такие стремления, как избегать столкновения с другими агентами, не менять направление своего движения, прямолинейность движения к выходу, максимизация скорости движения в случае паники и пр. В предлагаемой модели функция полезности расширена дополнительными факторами и учитывает такие характеристики, как состояние агента, состояние ситуации и др.

Также стоит упомянуть модели в рамках четвертого, так называемого, потокового подхода (например, [43]). Данные модели рассматривают движение людских потоков по аналогии с движением жидкости или газа, считая людскую массу несжимаемой средой. Потоковые модели не оперируют человеком как агентом модели, а рассматривают толпу в целом, где основной характеристикой является плотность потока. Данный подход, скорее, противопоставляется описанным выше, которые как раз ставят своей задачей получение макрохарактеристик толпы, исходя из законов взаимодействия агентов, без эмпирических заключений о макроуровне изначально. Вдобавок, толпа не может быть корректно сведена ни к жидкости, ни к газу, так как относится к умеренно сжимаемым средам [83], не говоря уже о том, что возникающие важные эффекты паникующей толпы в таких моделях не регистрируются.

В данной работе рассматривается непрерывная стохастическая агентная модель в ограниченном пространстве с заданной геометрией, основанная на феноменологической модели Бекларяна-Акопова [55;56] с использованием уточнений характеристик агента и системы принятия решений агентом, приведенных в модели Антонини [58] и моделях Хелбинга [79;80;89]. Такая интеграция видится автору наиболее перспективным развитием данного типа задач, ввиду того, что феноменологический подход (модели Бекларяна-Акопова) позволяет привнести естественную дискретизацию задачи с последующим вычислением приращения всех характеристик агентов в каждый момент времени. Это снимает вопрос численного интегрирования уравнений Ньютона (молекулярный подход), и предлагает явные вычисления всех характеристик системы. С другой стороны, уточнение характеристик агента и его системы принятия решений, заимствованное из моделей Хелбинга и Антонини, позволит получить максимально реалистичную динамику толпы. В предлагаемом подходе автором даются явные вычислительные формулы для всех значимых характеристик модели.

Отметим, что в рассматриваемой модели совокупность агентов является совокупностью индивидуумов, лишенных каких-либо общих изначальных целеполаганий.

1.1.2 Требования к предлагаемой агентной модели

Предлагаемая модель не ставит своей целью решение задачи оптимального управления динамикой ЧС, а предназначена для проведения вычислительных экспериментов, нацеленных на прогнозирование интенсивности потока на “фронте выхода”, с учетом возникающих эффектов турбулентности, притяжения толпы и ряда других.

В предлагаемой модели реализуется концепция перехода от фиксированных значений ряда показателей, отражающих как геометрию помещения, так и физику процесса перемещения агентов, к представлению их в качестве управляющих параметров модели. В результате удается построить гибкую, универсальную модель, позволяющую варьировать управляющими параметрами и, как следствие, калибровать модель с целью максимизации правдоподобия с реальными процессами. Также построенная модель допускает дальнейшее усложнение во всех аспектах (геометрия помещения, механика взаимодействия агентов, характеристики самих агентов и т.д.) и введение новых уравнений связи и условий.

В результате анализа имеющихся моделей и подходов, законодательных актов, а также исследований по психологии толпы были сформулированы основные требования, которые легли в основу предлагаемой агентной модели поведения при ЧС. Перечень требований представлен в Таблице 1, где также отражено выполнение указанных требований рядом существующих моделей. Для сравнения были выбраны наиболее часто используемые модели.

Таблица 1 — Требования к разрабатываемой модели и программному комплексу

№	Требование	Модели Хелбинга	Дискретные модели	Имитационно-стохастическая модель движения людских потоков МЧС России	Источник
1	*среднее время расчета одного сценария ЧС не более 10 мин.*	–	+	+	[44] (свод правил МЧС)
2	*моделирование фронта выхода*	–	–	–	требования ЛПР*
3	*расчет интенсивности выходного потока на фронте выхода*	–	–	–	требования ЛПР*
4	учет пола/возраста агентов	+	+	+	[37] (методика МЧС)
5	наличие случайных препятствий	+	–	–	требования ЛПР*
6	учет влияния взрыва на поведение агентов	+	–	–	[88] (исследования социальных-психологов)
7	возможность изменения геометрии помещения	+	+	–	требования ЛПР*

8	учет эффекта “притяжения” толпы	–	–	+	[88]
9	учет эффекта “турбулентности” толпы	+	–	–	Видеозаписи Швейцарской высшей технической школы Цюриха
10	расчет плотности потока	+	–	+	[44]
11	наличие личного пространства агента	+	–	+	[49] (исследования психологов)
12	расчет для большого количества агентов	–	+	+	требования ЛПР*
13	прогнозирование количества эвакуированных агентов с точностью 95%	–	–	–	[44]
14	разная степень ранения агентов	–	–	–	[54] (федеральный закон РФ)

15	индивидуальное восприятие ситуации агентом	–	–	–	[7] (исследования социологов)
16	учет площади проекции агента	+	–	+	[37]
17	различное начальное распределение агентов	+	+	–	[44]
18	учет угла обзора агента	+	+	–	[67] (исследования физиков)
19	учет близости агента к стенам	+	+	–	[88]
20	учет скорости потока	+	+	+	[37]
21	учет направления потока	–	+	–	Видеозаписи Швейцарской высшей технической школы Цюриха
22	учет комфортной скорости агента	+	–	–	[88]

23	учет многочастичного взаимодействия	–	–	–	Видеозаписи Швейцарской высшей технической школы Цюриха
24	возможность конфигурирования и масштабирования модели	–	–	–	ЛПР*
25	параметризация модели для имитации различных условий реализации ЧС	–	–	–	ЛПР*

*ЛПР — лицо принимающее решения при возникновении ЧС. В частности, для постановки задачи в качестве ЛПР выступило ООО “ГЕНКЕЙ-ТЕЛЕКОМ”. Тем не менее, иные ЛПР имеют схожие требования [38; 45].

Курсивом в Таблице 1 выделены центральные требования к модели.

Как видно из Таблицы 1, модели Хелбинга хотя и отличаются высокой степенью адаптации к реальности, тем не менее имеют ряд ключевых недостатков, таких как низкое быстродействие, отсутствие ряда важных индивидуальных характеристик агентов, а также высоким уровнем сложности внесения изменений в модель, что делает невозможным ее дальнейшее масштабирование. Дискретные модели и утвержденная на законодательном уровне модель МЧС, в свою очередь, отличаются высокой скоростью работы, но при этом значительно уступают в части соответствия реальным процессам.

Предлагаемая модель имеет следующую структуру. Задано активное пространство прямоугольной формы с диаметральными выходами. Все простран-

ство поделено на одинаковые области также прямоугольной формы, в каждой из которой задано свое распределение агентов по площади. Каждый агент характеризуется своим состоянием и правилами взаимодействия с другими агентами. При этом как состояние, так и правила перемещения каждого из агентов являются функциями от статуса ситуации, которая характеризует степень экстремальности обстановки в восприятии агента.

Параметры модели, характеризующие геометрию пространства, состояние агента и правила взаимодействия агентов, делятся на две группы: параметры, принимающие абсолютные значения; параметры, требующие калибровки. Первая группа параметров (координаты углов помещения, координаты выходов, количество агентов и др.) состоит из показателей, которые подаются на вход модели и могут принимать любые значения. Вторая группа параметров отвечает за правила поведения агентов и требует проведения экспериментов с целью калибровки их значений.

1.2 Модель поведения толпы при отсутствии чрезвычайной ситуации

1.2.1 Описание пространства, состояния агентов и правил принятия решений в случае отсутствия их взаимодействий

В условии отсутствия ЧС основное стремление агента – покинуть помещение, с наименьшими потерями, в которые входят отклонение от прямолинейной траектории движения к выходу, пересечение с другими агентами, замедление скорости движения и ряд других параметров. Среди допущений модели стоит выделить тот факт, что рассматривается одноэтажное помещение прямоугольной формы с диаметральными выходами. Приведем формальное описание модели. Введем следующие обозначения:

N – общее число агентов (рассматривается как *параметр* модели);
(a_0,b_0) – координата левого верхнего угла помещения (*параметр*),
$(a_{11},b_{11}),(a_{12},b_{12})$ – координаты вершин первого выхода (*параметр*),
$(a_{21},b_{21}),(a_{22},b_{22})$ – координаты вершин второго выхода (*параметр*),

len_1, len_2 – длина и ширина помещения (*параметр*), соответственно (геометрия помещения представлена на Рисунке 1.1).

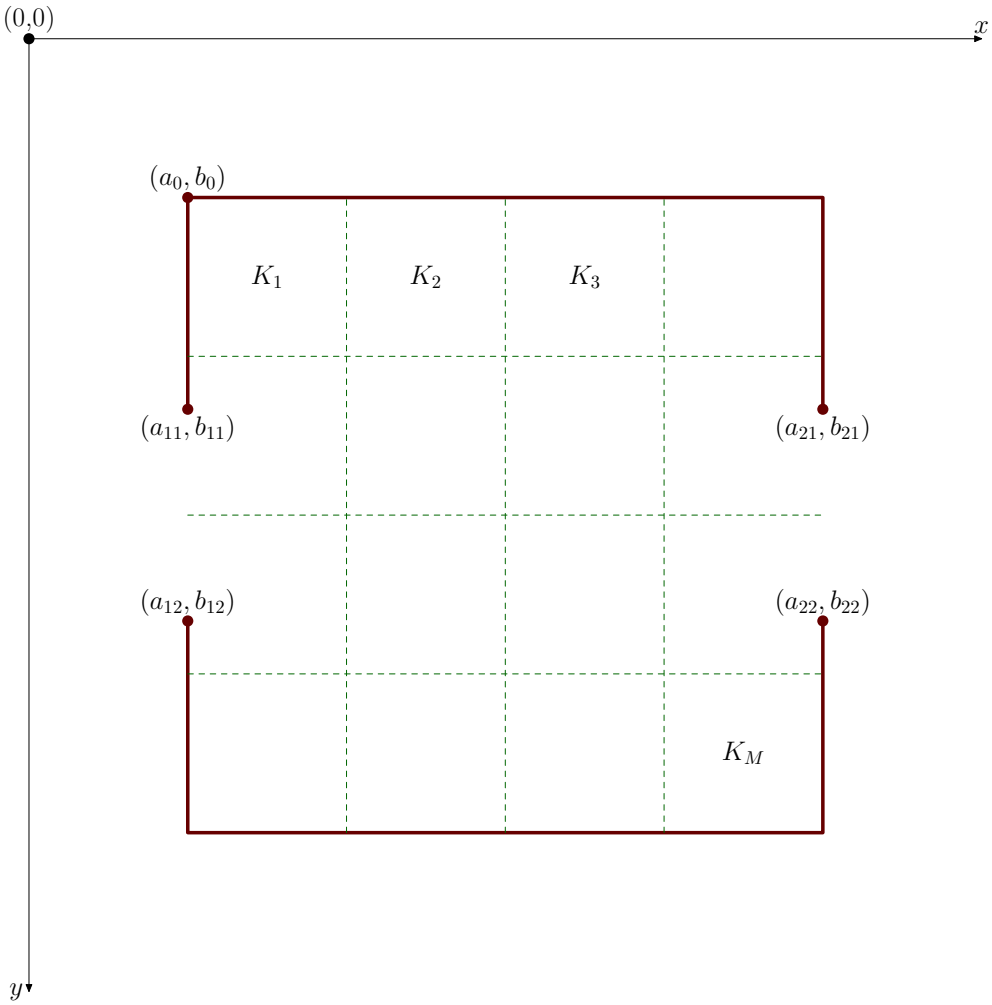


Рисунок 1.1 — Описание моделируемого помещения.

Имеют место ограничения

$$\begin{cases} a_0 > 0 \\ a_{lq} \geq a_0 \\ b_0 > 0 \\ b_{lq} \geq b_0 \\ a_{11} = a_{12} \qquad l,q \in \{1,2\} \\ a_{21} = a_{22} \\ a_{2q} - a_{1q} = len_1 \\ b_{22} - b_{21} \leq len_2 \\ b_{12} - b_{11} \leq len_2 \end{cases}$$

Заметим, что положительное направление оси Oy направлено вниз, а левый верхний угол помещения отстоит от начала координат. Данная конструкция

обусловлена необходимостью наличия зазора между выходом и осями, чтобы агенты, покидающие помещение, не создавали искусственный затор по причине близости к краю области проектирования.

Само помещение разбито на M прямоугольных областей за счет равномерно распределенных горизонтальных и вертикальных прямых. Количество клеток по вертикали (m_{vert}) и горизонтали (m_{hor}) также являются *параметрами* моделирования. Очевидно, что $M = m_{vert} \times m_{hor}$.

Количество агентов в клетке K_l в начальный момент обозначается n_{K_l}. Имеет место равенство

$$\sum_{l=1}^{M} n_{K_l} = N.$$

В каждой клетке K_l, $l = 1, \ldots, M$, задается собственное распределение положений агентов в начальный момент времени, обозначаемое $\mathbb{F}_{K_l}$. Распределение $\mathbb{F}_{K_l}$ вместе с n_{K_l} также являются *параметрами*.

$t = 1, 2, \ldots, T$, где $T \in [1, +\infty]$ – модельное время, допускающее дробление вплоть до миллисекунд (квантование времени необходимо для аппроксимации непрерывной модели дискретной). Подобное квантование времени обусловлено тем, что оно намного меньше, чем величина минимального времени для принятия решения агентом;

$i = 1, 2, \ldots, N$ – индекс агентов;

$v_{i,comf}$ – значение комфортной скорости (скалярная величина) ходьбы агента,

$v_{i,max}$ – значение максимальной скорости (скалярная величина) ходьбы агента,

(комфортное и максимальное значения скорости вычисляются по Таблице 2);

$v_i(t)$ – абсолютное значение скорости (скалярная величина) перемещения агента в момент времени t;

o_i – возраст агента. Значения нормально распределены в отрезке [6, 79];

g_i – пол агента. Случайная величина, равновероятно принимающая значения 1 (мужчина) или 0 (женщина);

$x_i(t)$ – абсцисса положения агента в момент времени t;

$y_i(t)$ – ордината положения агента в момент времени t;

$\vec{r}_i(t) = \{x_i(t), y_i(t)\}$ – радиус-вектор положения агента в момент времени t;

Таблица 2 — Комфортная и максимальная скорости ходьбы для разных гендерно-возрастных групп, приведенные в работах [63; 96]

Возраст, лет	Комфортная скорость, м/с	Максимальная скорость, м/с
Мужчины		
6 – 12	1,183	2,151
13 – 19	1,217	2,213
20 – 29	1,393	2,533
30 – 39	1,458	2,456
40 – 49	1,462	2,462
50 – 59	1,393	2,069
60 – 69	1,359	1,933
70 – 79	1,330	2,079
Женщины		
6 – 12	1,133	1,987
13 – 19	1,217	2,134
20 – 29	1,407	2,467
30 – 39	1,415	2,342
40 – 49	1,391	2,123
50 – 59	1,395	2,010
60 – 69	1,296	1,774
70 – 79	1,272	1,749

$\vec{d_i}(t)$ – направляющий единичный вектор перемещения агента в момент времени t.

$\delta_i(t)$ – радиус “личного пространства” агента;

$dist_{i,j}(t)$ – расстояние между i-м и j-м агентами

$$dist_{i,j}(t) = \sqrt{[x_i(t) - x_j(t)]^2 + [y_i(t) - y_j(t)]^2};$$

$st_i(t) \in \{0,1,2,3\}$ – статус агента в момент времени t (0 – убит, 1 – ранен, 2 – дезориентация, 3 – жив). Статус 2 является временным и спустя некоторый период меняется на 3. Присвоение и смена статуса агента более подробно разобраны в параграфе 1.3. В условиях отсутствия ЧС и давки, $st_i(t) = 3$ для всех i;

$sit_i(t) \in \{0,1,2,3\}$ – статус восприятия агентом окружающей ситуации в момент времени t (0 – отсутствие ЧС, 3 – время сразу после ЧС, 2 – активная стадия ЧС, 1 – угасание ЧС). Присвоение и смена статуса ситуации более по-

дробно разобраны в параграфе 1.3. В условиях отсутствия ЧС, $sit_i(t) = 0$ для всех i;

$s_i(t)$ – площадь горизонтальной проекции агента,

$$s_i(t) = \begin{cases} 0, & \text{если } st_i(t) = 0 \\ \sigma_1 \overline{s}_i, & \text{если } st_i(t) = 1 \\ \sigma_2 \overline{s}_i, & \text{если } st_i(t) = 2 \\ \overline{s}_i, & \text{если } st_i(t) = 3, \end{cases}$$

где $\overline{s}_i$ – базовое значение, σ_1, σ_2 – поправочные коэффициенты (*параметры*), причем $\sigma_1 > \sigma_2 \geq 1$. Значения коэффициентов обусловлены тем, что раненный человек имеет большую площадь проекции, ввиду появившейся, например, хромоты, контузии или просто ухудшения координации движения. Дезориентированный человек, хоть и в меньшей степени, но также склонен к ухудшению своего позиционирования, что влечет увеличение площади проекции.

Таблица 3 — Площади горизонтальной проекции взрослых людей

Тип одежды	Ширина, м	Толщина, м	Площадь горизонтальной проекции, м2/чел.
летняя	0,46	0,28	0,100
весенне-осенняя	0,48	0,30	0,113
зимняя	0,50	0,32	0,125

Таблица 4 — Площади горизонтальной проекции детей и подростков

Тип одежды	Возрастные группы		
	Младшая до 9 лет	Средняя 10 – 13 лет	Старшая 14 – 16 лет
домашняя одежда	0,04	0,06	0,08
домашняя одежда со школьной сумкой	0,07	0,10	0,14
уличная одежда	0,09	0,13	0,16

Базовое ($\overline{s}_i$) значение площади горизонтальной проекции агента рассчитывается на основании данных из методики МЧС России [37], представленных в Таблицах 3 и 4.

В результате усреднения данных по типу одежду, получаем итоговые базовые значения площади горизонтальной проекции в зависимости от возрастной группы, отраженные в Таблице 5.

Таблица 5 — Усредненные площади горизонтальной проекции

Возрастная группа	Площадь горизонтальной проекции, м2/чел.
до 9 лет	0,070
10 – 13 лет	0,100
14 – 16 лет	0,130
старше 16 лет	0,113

Геометрическая интерпретация площади горизонтальной проекции представляет из себя круг с центром в текущем местоположении агента и радиусом $\sqrt{s_i/\pi}$.

$\vec{D}_i(t)$ – направляющий единичный вектор агента к точке выхода в момент времени t.

Точка выхода определяется следующим образом. При начальном распределении агентов каждый из них выбирает один из двух альтернативных выходов с заданным вероятностным распределением $\mathbb{E}$. Далее, в случае отсутствия ЧС, агент выбирает точку выхода $(x_{ex}(t),y_{ex}(t))$ по следующему закону

$$(x_{ex}(t),y_{ex}(t)) = \begin{cases} (a_{l1},b_{l1}+\sqrt{s_i/\pi}), & \text{если } y_i(t) < b_{l1}+\sqrt{s_i/\pi}; \\ (a_{l1},y_i(t)), & \text{если } y_i(t) \in [b_{l1}+\sqrt{s_i/\pi},b_{l2}-\sqrt{s_i/\pi}]; \\ (a_{l1},b_{l2}-\sqrt{s_i/\pi}), & \text{если } y_i(t) > b_{l2}-\sqrt{s_i/\pi}, \end{cases} \quad (1.1)$$

где $l \in \{1,2\}$ – номер выбранного выхода. Таким образом, агент движется прямолинейно к той точке интервала выбранного выхода, которая ближе к его текущей координате (см. рис. 1.2).

Наличие отступа в виде $\sqrt{s_i/\pi}$ от краев выхода аргументировано необходимостью наличия зазора для прохода агентом через проем выхода. Выбор $\vec{D}_i(t)$ в случае ЧС будет описан в параграфе 1.3.

$\rho_i(t)$ – плотность людей в толпе относительно агента в момент времени t,

$$\rho_i(t) = \frac{\sum\limits_{j=1}^{N} \chi_i^j(t) s_j}{\eta^2 s_i},$$

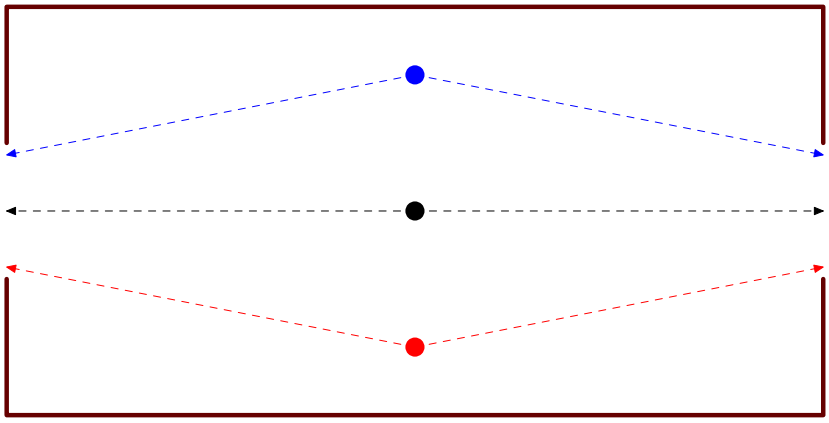


Рисунок 1.2 — Описание точки выхода.

где η – коэффициент пропорции между окружающим пространством вокруг агента, где вычисляется плотность, и его площади горизонтальной проекции (*параметр*), $\chi_i^j(t)$ – характеристическая функция присутствия j-ого агента в окружении i-ого агента, т.е.

$$\chi_i^j(t) = \begin{cases} 1, & \text{если } (x_j(t),y_j(t)) \in B(\eta\sqrt{\frac{s_i}{\pi}},(x_i(t),y_i(t))) \text{ и } st_j(t) \neq 0, \\ 0, & \text{если } (x_j(t),y_j(t)) \notin B(\eta\sqrt{\frac{s_i}{\pi}},(x_i(t),y_i(t))) \text{ или } st_j(t) = 0, \end{cases}$$

где $B(r,(x,y))$ – круг радиуса r с центром в точке (x,y).

Изначально каждый агент имеет ограниченное собственное личное пространство [56]. Однако в результате увеличения плотности толпы вокруг агента его личное пространство в начале сжимается вплоть до критического уровня, а потом резко разжимается, и возникает волна сжатия, выраженная в выталкивании агентов от центра толпы к краям. При этом часть агентов погибает как вследствие волн сжатия, то есть отбрасывания на стены, так и по причине давки, то есть превышения плотности толпы критического уровня. Сказанное задается в виде функциональной связи между радиусом личного пространства $\delta_i(t)$ i-ого агента и плотностью $\rho_i(t)$ вокруг i-ого агента

$$\delta_i(t) = \begin{cases} \varsigma_i(t)\overline{\delta}_{1,i}, & \rho_i(t) \leq \overline{\rho}_{1,i}(t), \\ \varsigma_i(t)\overline{\delta}_{2,i}, & \overline{\rho}_{1,i}(t) < \rho_i(t) \leq \overline{\rho}_{2,i}(t), \\ \varsigma_i(t)\overline{\delta}_{3,i}, & \overline{\rho}_{2,i}(t) < \rho_i(t) \leq \overline{\rho}_{3,i}(t), \\ \varsigma_i(t)\overline{\delta}_{4,i}, & \overline{\rho}_{3,i}(t) < \rho_i(t) \leq \overline{\rho}_{4,i}(t), \end{cases}$$

$$\varsigma_i(t) = \begin{cases} \overline{\varsigma}_0, & \text{если } sit_i(t) = 0, \\ \overline{\varsigma}_1, & \text{если } sit_i(t) = 1, \\ \overline{\varsigma}_2, & \text{если } sit_i(t) = 2, \\ \overline{\varsigma}_3, & \text{если } sit_i(t) = 3, \end{cases}$$

$$\begin{gathered} \overline{\varsigma}_3 > \overline{\varsigma}_2 > \overline{\varsigma}_1 > \overline{\varsigma}_0, \\ \overline{\rho}_{1,i}(t) = \frac{\overline{\rho}_{3,i}(t)}{3}, \quad \overline{\rho}_{2,i}(t) = \frac{2\overline{\rho}_{3,i}(t)}{3}, \\ \overline{\rho}_{3,i}(t) = \varrho_3(sit_i(t)) \frac{s_i \sum\limits_{j=1}^{N} \chi_i^j(t)}{\sum\limits_{j=1}^{N} \chi_i^j(t) s_j}, \quad \overline{\rho}_{4,i}(t) = \varrho_4(sit_i(t)) \frac{s_i \sum\limits_{j=1}^{N} \chi_i^j(t)}{\sum\limits_{j=1}^{N} \chi_i^j(t) s_j} \\ \varrho_4(sit_i(t)) > \varrho_3(sit_i(t)), \quad \varrho_3(0) > \varrho_3(1) > \varrho_3(2) > \varrho_3(3), \\ \varrho_4(3) > \varrho_4(2) > \varrho_4(1) > \varrho_4(0), \\ \overline{\delta}_{1,i} = \vartheta_1 \sqrt{\frac{s_i}{\pi}}, \quad \overline{\delta}_{2,i} = \vartheta_2 \sqrt{\frac{s_i}{\pi}}, \quad \overline{\delta}_{3,i} = \vartheta_3 \sqrt{\frac{s_i}{\pi}}, \quad \overline{\delta}_{4,i} = \vartheta_4 \sqrt{\frac{s_i}{\pi}}, \\ \vartheta_4 > \vartheta_1 > \vartheta_2 > \vartheta_3. \end{gathered} \tag{1.2}$$

Радиус личного пространства $\delta_i(t)$ является кусочно постоянной функцией (см. рис. 1.3) и, в отличие от площади горизонтальной проекции $s_i(t)$, характеризует не физическое, а психологическое состояние агента. Коэффициент $\varsigma_i(t)$, выступающий также в качестве *параметра* модели, отражает поправки в радиусе личного пространства в зависимости от статуса ситуации: чем более чрезвычайна ситуация в восприятии агента, тем больше его стремление расширить свое личное пространство. Сами же базовые значения радиуса личного пространства $\overline{\delta}_{l,i}, l = 1,\ldots,4$, пропорциональны площади горизонтальной проекции с некоторым множителем ϑ_l (*параметр*). Пороговые значения плотности толпы $\overline{\rho}_{l,i}(t), l = 1,\ldots,4$, индуцирующие скачкообразное изменение радиуса личного пространства, отражают готовность агента уменьшить радиус личного пространства при уплотнении окружающих агентов. В качестве фактора анализа окружающей плотности выступает отношение площади горизонтальной проекции агента и средней площади горизонтальной проекции окружающих агентов (см. (1.2)). Таким образом, чем меньше площадь проекции самого агента, тем меньше пороговые значения плотности, и наоборот, чем меньше средняя площадь проекции окружающих агентов, тем больше пороговые значения. Тем не

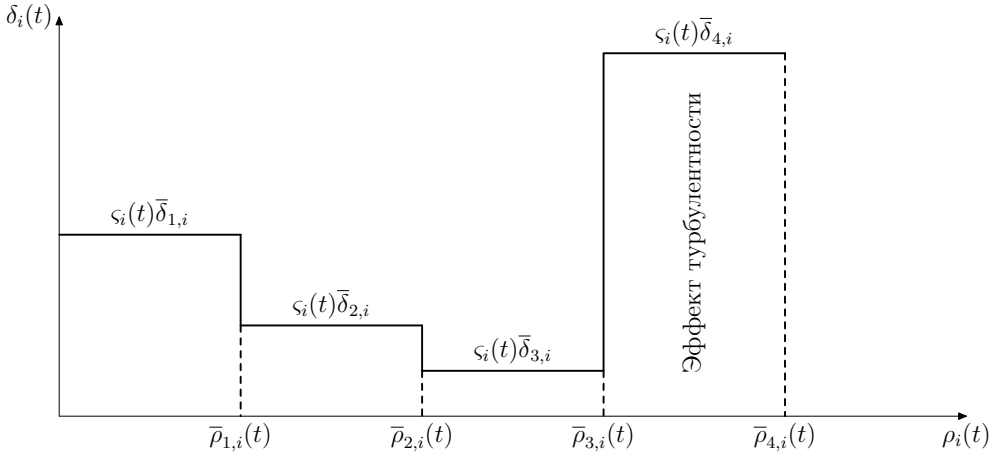
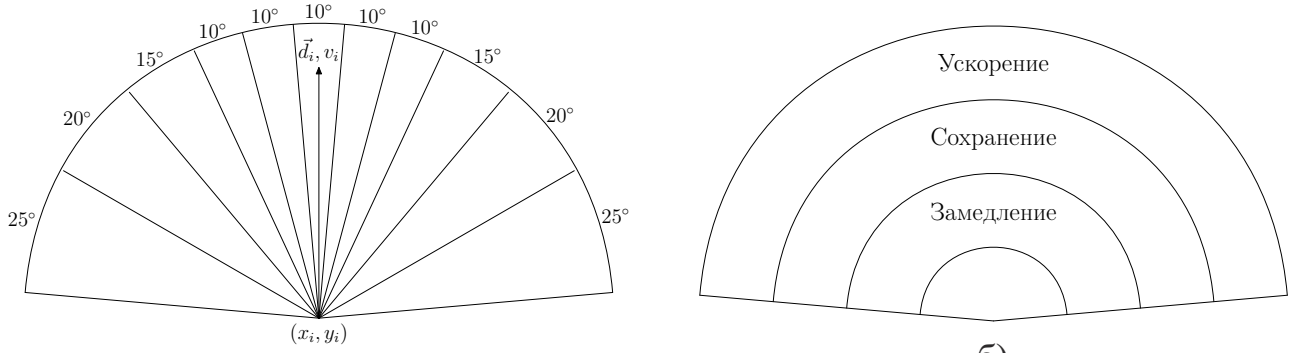


Рисунок 1.3 — Зависимость радиуса личного пространства i-ого агента от плотности толпы $\rho_i(t)$.

менее, при сверхвысокой плотности ($\rho_i(t) > \overline{\rho}_{3,i}(t)$) агент начинает паниковать и резко увеличивает радиус личного пространства, стремясь высвободиться из окружения, расталкивая всех вокруг, тем самым и создавая эффект турбулентности. Более детальное описание самого эффекта турбулентности будет дано позже. Коэффициенты ϱ_3, ϱ_4, являющиеся *параметрами*, отражают зависимость пороговых значений плотности от соотношения площадей горизонтальных проекций и статуса ситуации. Чем экстремальней ситуация в восприятии агента, тем раньше наступает паника. Также стоит отметить, что при превышении плотностью критического значения ($\rho_i(t) > \overline{\rho}_{4,i}(t)$) агент погибает в результате давки и его статус меняется на $st_i(t) = 0$.

Система принятия решения каждого агента базируется на анализе ситуации в рамках его сектора обзора с углом развертки $\theta(sit_i(t))$, который также является *параметром* [58]. Следуя [67], максимальный угол обзора принимается равным 170° для всех агентов, при этом весь сектор обзора разбивается на одиннадцать подсекторов, центральный из которых отражает текущее направление движения, а десять остальных, симметричных относительно центрального, отражают отклонение агента от текущего направления движения (см. рис. 1.4а). Заметим, что меры углов подсекторов тем больше, чем дальше подсектор от центрального, что объясняется более детальным просчетом движения в случае незначительного отклонения, т.к. периферийное зрение уступает в части детализации. Радиус сектора обзора равен $\eta\sqrt{\frac{s_i}{\pi}}$. Изменение угла обзора в зависимости от статуса ситуации происходит посредством симметричного увеличения или уменьшения числа подсекторов.

В каждый момент времени агент определяет новый единичный вектор перемещения $\vec{d}_{i,new}$ по центру выбранного подсектора, а сам подсектор далее отождествляется со своим единичным вектором.



а) б)

Рисунок 1.4 — Пространство выбора агента.

При этом в каждый момент времени агент имеет три альтернативы по отношению к выбору абсолютного значения своей новой скорости $v_{i,new}$: сохранить текущую скорость v_i, увеличить скорость до значения $v_{acc} = v_i + \gamma_{acc}(v_{i,max} - v_i)$ или уменьшить скорость до значения $v_{dec} = (1 - \gamma_{dec})v_i$, где γ_{acc},γ_{dec} являются *параметрами* модели и принимают значения в интервале $(0,1)$ (см. рис. 1.4б).

Таким образом пара значений $(v_{i,new}, \vec{d}_{i,new})$, определяющих перемещение агента, выбираются агентом из 33 альтернативных возможностей (11 направлений по центрам подсекторов и 3 варианта выбора скорости). Итоговое положение агента вычисляется по формуле

$$\vec{r}_{i,new} = \vec{r}_i + v_{i,new}\vec{d}_{i,new}.$$

При этом те альтернативы, которые попадают вне или на стены помещения, агентом не рассматриваются.

Выбор из 33 возможных итоговых положений осуществляется на основе следующего функционала

$$\mathfrak{L}_i = \beta_{occ}(sit_i(t))\text{occupation}_i + \beta_{dir}(sit_i(t))\text{direction}_i + \beta_{dest}(sit_i(t))\text{destination}_i + \beta_{angle}(sit_i(t))\text{angle}_i + \beta_{wall}(sit_i(t))\text{wall}_i + \beta_{vel}(sit_i(t))(|v_{i,new} - v_{i,comf}|)^{\lambda_{vel}} + \beta_{max}(sit_i(t))(|v_{i,new} - \hat{v}_{i,max}|)^{\lambda_{max}} + \beta_{avg}(sit_i(t))(|v_{i,new} - v_{i,avg}|)^{\lambda_{avg}} + \beta_{expl}(sit_i(t))\text{explosion}_i.$$

β_{occ},β_{dir},β_{dest},β_{angle},β_{wall},β_{vel},β_{max},β_{avg},$\beta_{expl} > 0$ – весовые коэффициенты, выступающие в качестве *параметров* модели и меняющие свои значения в зависимости от статуса ситуации.

occupation$_i$ отражает значимость присутствия других агентов в выбранном секторе $\vec{d}_{i,new}$, другими словами,

$$\text{occupation}_i = \sum_{j=1}^{N} I_{ij} e^{-\gamma_1 \|\vec{r}_{i,new} - \vec{r}_j\|},$$

где I_{ij} равно единице, если j-ый агент находится в секторе $\vec{d}_{i,new}$, и нулю иначе; $\|\vec{r}_{i,new} - \vec{r}_j\|$ – расстояние между j-ым агентом и новым положением i-ого; γ_1 – нормирующий коэффициент. Очевидно, что чем меньше количество агентов в выбранном секторе, тем меньше значение occupation$_i$. Значение occupation$_i$ также монотонно убывает в зависимости от степени удаленности других агентов от i-ого агента в секторе.

direction$_i$ отражает угол между единичными векторами $\vec{d}_{i,new}$ и $\vec{d}_i$, другими словами,

$$\text{direction}_i = \arccos(\vec{d}_{i,new}, \vec{d}_i),$$

где $(\cdot,\cdot)$ – скалярное произведение векторов. Данный показатель отражает нежелание агента менять направление своего движения.

destination$_i$ отражает угол между единичными векторами $\vec{d}_{i,new}$ и $\vec{D}_i$, другими словами

$$\text{destination}_i = \arccos(\vec{d}_{i,new}, \vec{D}_i).$$

Данный показатель отражает желание агента двигаться напрямую к точке выхода.

angle$_i$ пропорционален сумме углов между единичными векторами $\vec{d}_i$ и $\vec{d}_j$, другими словами,

$$\text{angle}_i = \sum_{j=1}^{N} I_{ij} \alpha_{ij} e^{-\gamma_2 \|\vec{r}_{i,new} - \vec{r}_j\|},$$

где $\alpha_{ij} = \arccos(\vec{d}_i, \vec{d}_j)$ – угол между направлениями движения агентов; γ_2 – нормирующий коэффициент. Очевидно, что чем более сонаправленно двигаются агенты, тем меньше значение angle$_i$. В то же время значение монотонно убывает

в зависимости от степени удаленности других агентов от i-ого агента в выбранном секторе.

wall_i зависит от минимального расстояния от агента до стен, другими словами,

$$\text{wall}_i = e^{-\gamma_3 \triangle_{i,new}},$$

где $\triangle_{i,new}$ – минимальное из расстояний от нового положения агента до стен; γ_3 – нормирующий коэффициент. Очевидно, что чем дальше агент от стен, тем меньше wall_i. Значение $\triangle_{i,new}$ вычисляется по формуле

$$\triangle_{i,new} = \begin{cases} \min\{x_{i,new} - a_{11}, a_{21} - x_{i,new}, y_{i,new} - b_0, b_0 + len_2 - y_{i,new}\}, \\ \text{если } y_{i,new} \leq \min\{b_{11}, b_{21}\} \text{ или } y_{i,new} \geq \max\{b_{12}, b_{22}\}; \\ \min\{y_{i,new} - b_0, b_0 + len_2 - y_{i,new}\}, \\ \text{если } \min\{b_{11}, b_{21}\} < y_{i,new} < \max\{b_{12}, b_{22}\}. \end{cases}$$

$(|v_{i,new} - v_{i,comf}|)^{\lambda_{vel}}$ отражает отклонение скорости агента от его комфортабельной скорости, λ_{vel} – нормирующий коэффициент. Данный фактор дает свой вклад только при $sit_i = 0$.

$(|v_{i,new} - \hat{v}_{i,max}|)^{\lambda_{max}}$ отражает разницу между новой скоростью и максимальной скоростью бега, λ_{max} – нормирующий коэффициент. Данный фактор дает свой вклад только при $sit_i \neq 0$. Определение $\hat{v}_{i,max}$ будет дано позже.

$(|v_{i,new} - v_{i,avg}|)^{\lambda_{avg}}$ отражает разницу между новой скоростью и средней скоростью агентов в секторе, λ_{avg} – нормирующий коэффициент,

$$v_{i,avg} = \frac{\sum\limits_{j=1}^{N} I_{ij} v_j e^{-\gamma_4 \|\vec{r}_{i,new} - \vec{r}_j\|}}{\sum\limits_{j=1}^{N} I_{ij}},$$

где γ_4 – нормирующий коэффициент.

explosion_i зависит от расстояния от агента до взрыва, другими словами,

$$\text{explosion}_i = e^{-\gamma_5 dist_{i,expl}},$$

где $dist_{i,expl}$ – расстояние от нового положения агента до взрыва; γ_5 – нормирующий коэффициент. Очевидно, что чем дальше агент от взрыва, тем меньше

explosion_i. Вопрос взрыва более подробно будет рассмотрен позже. Данный фактор дает свой вклад только при $sit_i \neq 0$.

Система принятия решений агента основана на выборе той альтернативы, при которой достигается минимальное значение описанного функционала, т.е.

$$(v_{i,new}, \vec{d}_{i,new}) = \arg\min \mathfrak{L}_i.$$

1.2.2 Правила взаимодействия агентов

Взаимодействие агентов в рамках описываемой модели рассматривается как абсолютно упругий нецентральный удар. В качестве критерия наступления взаимодействия выступает пересечение площадей горизонтальных проекций агентов. Другими словами,

$$dist_{i,j}(t) < \sqrt{s_i/\pi} + \sqrt{s_j/\pi}.$$

Итог взаимодействия, то есть измененные значение скорости $\tilde{v}_i$ и направление движения $\vec{\tilde{d}}_i$, является решением следующей системы соотношений

$$\begin{cases} s_i v_i \vec{d}_i + s_j v_j \vec{d}_j = s_i \tilde{v}_i \vec{\tilde{d}}_i + s_j \tilde{v}_j \vec{\tilde{d}}_j, & (1.3) \\ s_i v_i^2 + s_j v_j^2 = s_i \tilde{v}_i^2 + s_j \tilde{v}_j^2. & (1.4) \end{cases}$$

Соотношение (1.3) отражает закон сохранения импульса, а соотношение (1.4) – закон сохранения кинетической энергии [30], где в качестве массы выступает площадь горизонтальной проекции. Решение данной системы соотношений имеет вид

$$\begin{gathered} \tilde{v}_i = \sqrt{\tilde{v}_{i,n}^2 + \tilde{v}_{i,\tau}^2}, \quad \tilde{v}_j = \sqrt{\tilde{v}_{j,n}^2 + \tilde{v}_{j,\tau}^2}, \\ \tilde{v}_{i,\tau} = v_{i,\tau}, \quad \tilde{v}_{j,\tau} = v_{j,\tau}, \\ \tilde{v}_{i,n} = \frac{(s_i - s_j)v_{i,n} + 2s_j v_{j,n}}{s_i + s_j}, \quad \tilde{v}_{j,n} = \frac{(s_j - s_i)v_{j,n} + 2s_i v_{i,n}}{s_i + s_j}, \\ \tan\tilde{\alpha}_i = \frac{\tilde{v}_{i,\tau}}{\tilde{v}_{i,n}}, \quad \tan\tilde{\alpha}_j = \frac{\tilde{v}_{j,\tau}}{\tilde{v}_{j,n}}, \end{gathered}$$

где $\tilde{v}_{i,n}, \tilde{v}_{i,\tau}$ – нормальная и тангенсальная проекции относительно оси центров, соответственно; углы $\tilde{\alpha}_i$ и $\tilde{\alpha}_j$ – углы образованные векторами $\vec{\tilde{d}}_i$ и $\vec{\tilde{d}}_j$ относительно оси центров, соответственно.

В случае возникновения эффекта турбулентности описанный выше закон взаимодействия претерпевает изменение в части критерия наступления взаимодействия. Тот агент, который находится в зоне эффекта турбулентности, руководствуется не площадью горизонтальной проекции, а гибридной областью, представленной на рисунке 1.5, а само взаимодействие проходит с ближайшим агентом в центральном удлиненном секторе, причем паникующему агенту присваивается максимальная скорость.

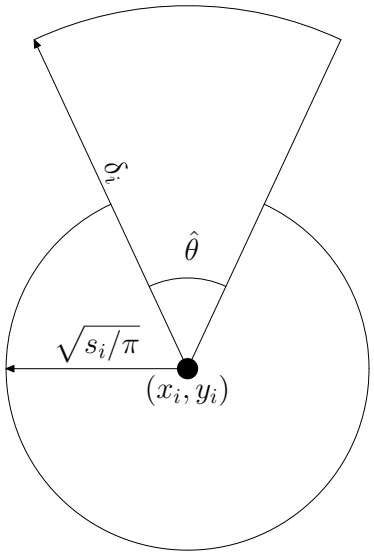


Рисунок 1.5 — Гибридная область анализа в случае эффекта турбулентности.

Заметим, что центральное направление гибридной области есть случайная величина с вероятностным распределением $\mathbb{T}$, а угол развертки центрального сектора $\hat{\theta}(sit_i(t))$ является *параметром* модели. Подобный подход обусловлен тем, что в случае панического состояния агента, вследствие сверхплотности окружающих агентов, тот начинает хаотичные расталкивания, стремясь высвободиться из окружения. Таким образом, в случае паники i-ого агента система

соотношений для взаимодействия с j-ым агентом принимает вид

$$\begin{cases} \pi\delta_i^2 v_i \vec{d_i} + s_j v_j \vec{d_j} = \pi\delta_i^2 \tilde{v}_i \vec{\tilde{d}_i} + s_j \tilde{v}_j \vec{\tilde{d}_j}, & (1.5) \\ \pi\delta_i^2 v_i^2 + s_j v_j^2 = \pi\delta_i^2 \tilde{v}_i^2 + s_j \tilde{v}_j^2, & (1.6) \\ \text{где} \quad j = \arg\min\limits_{j \in J} dist_{i,j}, \quad v_i = v_{i,max}, & (1.7) \\ J = \left\{ j \mid dist_{i,j} \le \delta_i, \arccos \dfrac{(\vec{r_j} - \vec{r_i}, \vec{d_i})}{\|\vec{r_j} - \vec{r_i}\|} \le \dfrac{\hat{\theta}}{2} \right\}. & (1.8) \end{cases}$$

Необходимо отметить, что если в результате взаимодействия новое положение агента выходит за пределы помещения, то агент считается погибшим в результате отбрасывания на стены и его статус меняется на $st_i(t) = 0$.

1.3 Модель поведения толпы при возникновении чрезвычайной ситуации

В качестве чрезвычайной ситуации рассматривается одиночный взрыв, центр которого $Expl(x_{expl}, y_{expl})$ является случайной величиной с вероятностным распределением $\mathbb{P}$. С центром взрыва связаны три концентрические окружности радиуса ε_0,ε_1 и ε_2, образующие зоны различного поражения агентов (см. рис. 1.6).

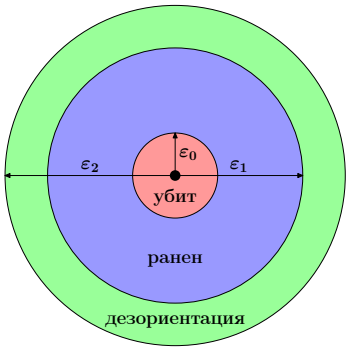


Рисунок 1.6 — Геометрия взрыва.

Попадание в каждую из зон поражения в момент взрыва меняет статус агента st_i на соответствующее значение. В случае смерти агента он перестает влиять на дальнейшее развитие модели, в том числе, не является преградой для перемещения. Если в результате взрыва агент дезориентирован, то он на протяжении ζ секунд остается неподвижным, а по истечении меняет свой статус на

$st_i = 3$. Расстояние от агента до взрыва $dist_{i,expl}$ рассчитывается по формуле

$$dist_{i,expl} = \sqrt{[x_i(t^*) - x_{expl}]^2 + [y_i(t^*) - y_{expl}]^2},$$

где t^* – момент взрыва.

При ЧС каждый из агентов, испытывая стресс и страх, перестает ориентироваться на комфортную скоростью ходьбы и готов даже на бег. На основе работ [69; 98; 99] была составлена Таблица 6, отражающая максимальные скорости $\hat{v}^{tab}_{i,max}$ бега агентов.

Таблица 6 — Максимальная скорость бега для разных гендерно-возрастных групп

Возраст, лет	Максимальная скорость, м/с
Мужчины	
6 – 12	4,176
13 – 19	4,328
20 – 29	6,600
30 – 39	6,900
40 – 49	6,900
50 – 59	5,700
60 – 69	3,500
70 – 79	2,400
Женщины	
6 – 12	4,176
13 – 19	4,328
20 – 29	5,200
30 – 39	4,500
40 – 49	4,200
50 – 59	4,300
60 – 69	2,640
70 – 79	1,810

При этом, в случае попадания агента в зону ранения при взрыве, его максимальная скорость бега меняется пропорционально расстоянию от взрыва. Другими словами,

$$\hat{v}_{i,max} = \begin{cases} \frac{dist_{i,expl} - \varepsilon_0}{\varepsilon_1 - \varepsilon_0} \hat{v}^{tab}_{i,max}, & \text{если } \varepsilon_0 < dist_{i,expl} \leq \varepsilon_1; \\ \hat{v}^{tab}_{i,max}, & \text{если } dist_{i,expl} > \varepsilon_1. \end{cases}$$

Также в случае ЧС, т.е. $sit_i \neq 0$, определение $v_{i,new}$ базируется не на максимальной скорости ходьбы $v_{i,max}$, а на максимальной скорости бега $\hat{v}_{i,max}$.

Функция $sit_i(t)$ определяется следующим образом (см. рис. 1.7)

$$sit_i(t) = \begin{cases} 0, & \text{если } 0 \leq t < t^* \text{ или } t \geq t^*_{3,i}; \\ 3, & \text{если } t^* \leq t < t^*_{1,i}; \\ 2, & \text{если } t^*_{1,i} \leq t < t^*_{2,i}; \\ 1, & \text{если } t^*_{2,i} \leq t < t^*_{3,i}, \end{cases}$$

где $t^*_{2,i} = t^* + \kappa_2(t^*_{1,i} - t^*), t^*_{3,i} = t^* + \kappa_3(t^*_{1,i} - t^*)$, а $t^*_{1,i}$ – функция от $dist_{i,expl}$ и также носит скачкообразный характер, имеющий следующий вид

$$t^*_{1,i} = \begin{cases} 0, & \text{если } 0 \leq dist_{i,expl} \leq \varepsilon_0; \\ \infty, & \text{если } \varepsilon_0 < dist_{i,expl} \leq \varepsilon_1; \\ t^* + \kappa_1 e^{-(dist_{i,expl} - \varepsilon_1)}, & \text{если } dist_{i,expl} > \varepsilon_1. \end{cases}$$

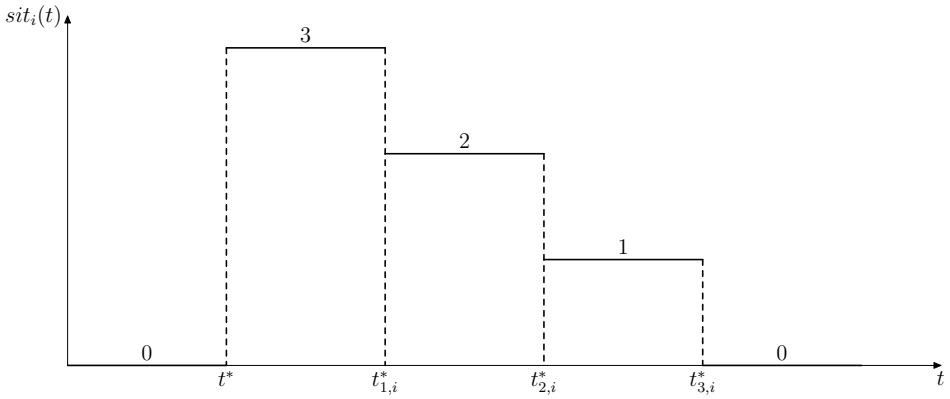


Рисунок 1.7 — График зависимость $sit_i(t)$ от времени.

Стресс, вызванный экстремальной ситуацией, провоцирует в агенте потерю ориентации на выход и переключение внимания на перемещения окружающих агентов. Другими словами, в случае ЧС агент в меньшей степени ориентируется на собственное мнение о местоположении выхода, а в большей на динамику толпы. Поэтому точка выхода агента продолжается определяться согласно системе (1.1) только, если выход находится в непосредственной зоне видимости агента. Для всех остальных агентов имеет место функция распределения $\mathbb{D}$ вероятности ориентации на положение выхода. Сам же вектор $\vec{D}_i$ в случае неориентированности агента на выход вычисляется как взвешаное среднее векторов

$\vec{d_j}$ окружающих агентов в соответствующем подсекторе, т.е.

$$\vec{\tilde{D}}_i = \sum_{j=1}^{N} I_{ij}\vec{d_j}e^{-\gamma_{exit}\|\vec{r}_{i,new}-\vec{r}_j\|},$$

$$\vec{D}_i = \frac{\vec{\tilde{D}}_i}{\|\vec{\tilde{D}}_i\|}.$$

1.4 Разработка алгоритма эволюционной кластеризации динамики толпы для снижения размерности задачи

С целью выявления устойчивых групп агентов в основную модель был импортирован программный пакет динамической кластеризации. В каждый момент модельного времени на множестве всех агентов проводится кластеризация одновременно по трем алгоритмам [2; 62; 77]:

1. Метод k-средних (k-means++).
2. Иерархический метод.
3. Метод нечеткой кластеризации C-средних (Fuzzy C-means).

Ввиду того, что для первого и третьего алгоритмов в качестве входного параметра выступает количество кластеров, чье значение, с учетом частоты проведения анализа и вариативности количества агентов и геометрии активного пространства, не поддается никакой оценке, то для вычисления оптимального количества кластеров использовался известный критерий компактности [66]:

$$S(U,c) = \frac{\frac{1}{n}\sum_{i=1}^{c}\sum_{k=1}^{n}\mu_{ik}^2\|x_k - v_i\|^2}{\min\limits_{i,j}\|v_i - v_j\|^2}, \tag{1.9}$$

$$c = \arg\min_{c}\{\min_{\Omega_c} S(U,c)\}, \tag{1.10}$$

где U – множество всех агентов, i,j – индекс кластеров, k – индекс агентов в кластере, μ_{ik} – вероятность принадлежности k-ого агента i-ому кластеру, x_k – координата k-ого агента, v_i – координата центра i-ого кластера, Ω_c – всевозможные разбиения U на c кластеров.

С учетом высокой степени перестроения кластеров (распад имеющихся кластеров и соединения в новые), а также максимального “правдоподобия” в качестве основного алгоритма был выбран алгоритм нечеткой кластеризации (см. рис. 1.8). Среди исследуемых характеристик кластерного анализа отдельно стоит

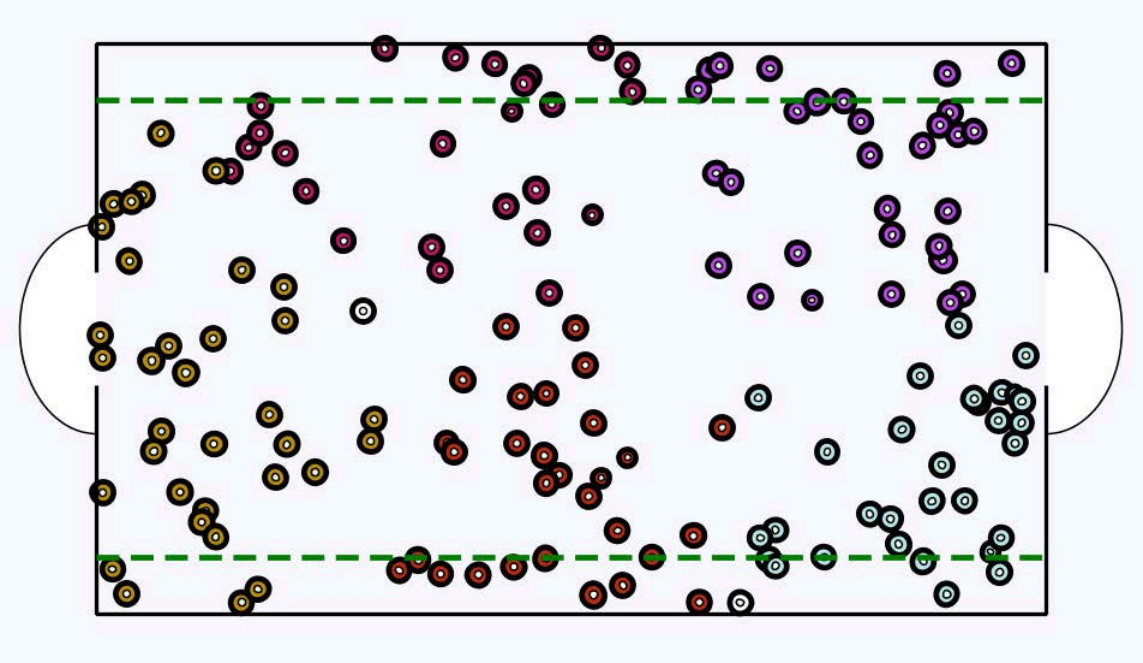


Рисунок 1.8 — Пример кластеризации.

отметить:

– Динамика количества кластеров.
– Динамика центров кластеров.
– Динамика диаметров кластеров.
– Количество агентов кластера.

Также важно отразить предпочтения агента относительно различных кластеров:

– Мера принадлежности агента к кластеру.
– Частота выбытия и примыкания к кластеру.

Также необходимо отметить, что оценки плотности и диаметров кластеров играют важную роль в процессе регистрации появления эффекта турбулентности и дальнейшего развития волн сжатий. Именно кластерный анализ рассматривается автором как основной инструмент дальнейшего анализа и выявления условий как возникновения, так и разрушения волн сжатия, которые постоянно наблюдаются в процессе экспериментов. Помимо этого проводимый кластерный анализ дает оценки локализации крупных скоплений людей, что позволит в дальнейшем оценить момент максимальной интенсивности на “фронте выхода”.

1.4.1 Модификация алгоритма нечеткой кластеризации

Алгоритм нечеткой кластеризации Fuzzy C-means является широко используемым подходом к проблеме категоризации наблюдаемых объектам по признакам. В отличие от четких алгоритмов, алгоритм Fuzzy C-means не относит объект однозначно к какому-нибудь кластеру, а сопоставляет каждому кластеру вероятность отнесения к нему наблюдаемых объектов, формируя, так называемую, матрицу принадлежности. Таким образом, на каждом шаге алгоритм каждый объект одновременно относится ко всем кластерам.

Впервые данный алгоритм был предложен в работе [92], далее развит в работе [68]. Окончательная форма была представлена в работе [61], а доказательство сходимости – в монографии [62]. Ниже идет описание шагов алгоритма.

Исходной информацией для кластеризации является матрица наблюдений $l \times n$

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ & & \dots & \\ x_{l1} & x_{l2} & \dots & x_{ln} \end{pmatrix},$$

где l – число объектов, n — число признаков (наблюдений) для каждого объекта (в нашем случае – координаты).

Задача кластеризации состоит в разбиении множества объектов на группы (кластеры) “похожих” между собой объектов. В n-мерном метрическом пространстве признаков мерой “сходства” двух объектов будем считать расстояние между ними.

Кластерная структура задаётся матрицей принадлежности ($c \times l$ матрица):

$$M = \begin{pmatrix} m_{11} & m_{12} & \dots & m_{1l} \\ m_{21} & m_{22} & \dots & m_{2l} \\ & & \dots & \\ m_{c1} & m_{c2} & \dots & m_{cl} \end{pmatrix},$$

где m_{ij} – степень принадлежности j-го элемента i-му кластеру.

Отметим, что матрица принадлежности должна удовлетворять следующим условиям:

1. $m_{ij} \in [0,1], i = \overline{1,c}, j = \overline{1,l}$,
2. $\sum\limits_{i=1}^{c} m_{ij} = 1, j = \overline{1,l}$, то есть каждый объект должен быть распределён между всеми кластерами,
3. $0 < \sum\limits_{j=1}^{l} m_{ij} < l$, то есть ни один кластер не должен быть пустым или содержать все элементы.

Для оценки качества разбиения используется критерий разброса, показывающий сумму расстояний от объектов до центров кластеров с соответствующими степенями принадлежности:

$$J = \sum_{i=1}^{c} \sum_{j=1}^{l} (m_{ij})^w d(v_i,x_j), \tag{1.11}$$

где $d(v_i,x_j)$ – евклидово расстояние между j-м объектом $x_j = (x_{j1}, x_{j2}, \ldots, x_{jn})$ и i-м центром кластера $v_i = (v_{i1}, v_{i2}, \ldots, v_{in})$, $w \in (1,\infty)$ – экспоненциальный вес, определяющий нечёткость, размытость кластеров.

$$V = \begin{pmatrix} v_{11} & v_{12} & \ldots & v_{1n} \\ v_{21} & v_{22} & \ldots & v_{2n} \\ & & \ldots & \\ v_{c1} & v_{c2} & \ldots & v_{cn} \end{pmatrix}$$

– $c \times n$ матрица координат центров кластеров, элементы которой вычисляются по формуле

$$v_{ik} = \frac{\sum\limits_{j=1}^{l} (m_{ij})^w x_{jk}}{\sum\limits_{j=1}^{l} (m_{ij})^w}, k = \overline{1,n}. \tag{1.12}$$

Задачей является нахождение матрицы M, минимизирующей критерий J. Для этого используется алгоритм нечётких C-средних, в основе которого лежит метод множителей Лагранжа. Он позволяет найти локальный оптимум, поэтому для различных запусков могут получиться разные результаты.

На первом шаге матрица принадлежностей M, удовлетворяющая условиям 1-3, генерируется случайным образом. Далее запускается итерационный процесс вычисления центров кластеров и пересчёта элементов матрицы степеней

принадлежности:

$$m_{ij} = \frac{1}{(d_{ij})^{\frac{2}{w-1}} \sum\limits_{k=1}^{c} \frac{1}{(d_{kj})^{\frac{2}{w-1}}}} \text{ при } d_{ij} > 0 \text{ и } m_{kj} = \begin{cases} 1, & k = i \\ 0, & k \neq i \end{cases} \text{ при } d_{ij} = 0, \quad (1.13)$$

где $d_{ij} = d(v_i,x_j)$ для $i = \overline{1,c}, j = \overline{1,l}$.

Вычисления продолжаются до тех пор, пока изменение матрицы M, характеризующееся величиной $\|M - M^*\|^2$, где M^* – матрица на предыдущей итерации, не станет меньше заранее заданного параметра остановки ε.

Остановимся на выборе значения w — экспоненциального веса. Чем больше это значение, тем матрица принадлежности более размазанная и при $w \to \infty$ элементы примут вид $m_{ij} = \frac{1}{c}$, что является плохим решением, так как все объекты с одинаковой степенью распределены по всем кластерам. При $w = 1$ алгоритм Fuzzy C-means вырождается в обычный k-means. Теоретически обоснованного правила выбора веса пока не существует, и обычно устанавливают $w = 2$.

Схема работы стандартного алгоритма Fuzzy C-means представлена в левой части на Рисунке 1.9.

Необходимо отметить ключевые недостатки данного алгоритма:

– Высокая степень зависимости результирующего разбиения от начальной матрицы принадлежности.
– Необходимость априорного задания числа кластеров.
– Отсутствие учета специфики предметной области (динамика толпы).

С целью уменьшения влияния перечисленных недостатков был разработан и имплементирован в модель модифицированный (эволюционный) алгоритм нечеткой кластеризации. Схема работы эволюционного алгоритма Fuzzy C-means представлена в правой части на Рисунке 1.9.

Также как и в стандартном алгоритме кластеризации, в начале экспертно задается экспоненциальный вес w. Далее осуществляется цикл по количеству кластеров от 2 до $N - 1$. В рамках каждого шага цикла осуществляется работа иерархического агломеративного алгоритма кластеризации с расстоянием Уорда [95]

$$R(P,S) = \frac{|S||P|}{|S| + |P|} \rho^2 \left(\sum_{p \in P} \frac{p}{|P|}, \sum_{s \in S} \frac{s}{|S|} \right),$$

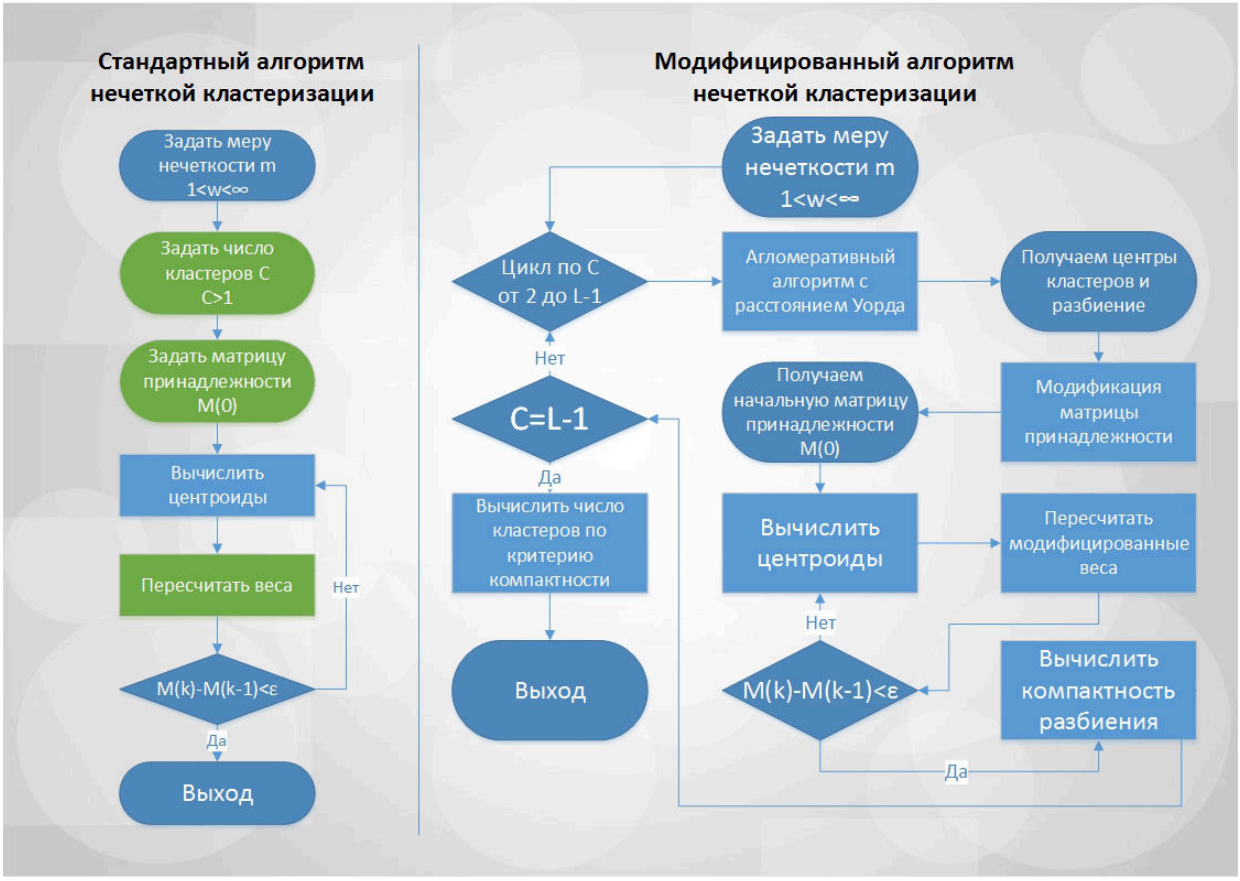


Рисунок 1.9 — Схема алгоритмов нечеткой кластеризации.

где $p \in P$ – элементы кластера P, $s \in S$ – элементы кластера S, $|\cdot|$ – число элементов в кластере, $\rho(\cdot,\cdot)$ – длина наименьшего пути с учетом наличия препятствий. Выбор именно расстояния Уорда в качестве оценки меры принадлежности к кластеру обусловлен тем фактом, что данное расстояние обладает свойствами монотонности, растяжения и редуктивности.

В результате работы агломеративного алгоритма получена матрица принадлежности $M_0^{'}$

$$M_0^{'} = \begin{pmatrix} m_{11}^{'} & m_{12}^{'} & \dots & m_{1l}^{'} \\ m_{21}^{'} & m_{22}^{'} & \dots & m_{2l}^{'} \\ & & \dots & \\ m_{c1}^{'} & m_{c2}^{'} & \dots & m_{cl}^{'} \end{pmatrix},$$

где $m_{ij}^{'} \in \{0,1\}$ и в каждом столбце встречается встречается ровно одна единица. Далее, чтобы получить начальную матрицу принадлежности для работы

алгоритма нечеткой кластеризации проводится следующее преобразование

$$m_{ij} = \frac{m_{ij}^{'} + \frac{\frac{A+1}{A+\frac{\alpha_{ij}}{\pi}} - 1 + \frac{\rho_j}{\rho_{ij}}}{1+\frac{1}{A}}}{\sum\limits_{i=1}^{c} m_{ij}^{'} + \frac{\frac{A+1}{A+\frac{\alpha_{ij}}{\pi}} - 1 + \frac{\rho_j}{\rho_{ij}}}{1+\frac{1}{A}}}, \tag{1.14}$$

где ρ_{ij} – расстояние от j-ого агента до i-ого кластера с учетом наличия препятствий, ρ_j – расстояние от j-ого агента до того кластера, которому он принадлежал по итогам работы агломеративного алгоритма, α_{ij} – угол между направлением движения j-ого агента ($\vec{d}_j$) и направлением от агента к центру i-ого кластера, $A \in (0,1]$ – коэффициент компенсации (на сколько совпадение направления движения агента и направления от агента к центру кластера компенсирует дальность от кластера). Чем меньше значение A, тем более значимо совпадение направлений.

Таким образом, в результате преобразования 1.14 начальная матрица принадлежности формируется не случайно, а с учетом как метрической близости, так и динамики агентов.

Новый, модифицированный критерий разброса также учитывает динамику агентов:

$$J^{'} = \sum_{i=1}^{c}\sum_{j=1}^{l}(m_{ij})^{w} e^{B\frac{\alpha_{ij}}{\pi}} d(v_i,x_j), \tag{1.15}$$

где $B > 0$ – весовой коэффициент значимости направления движения агента.

Как следствие, соответствующая модификация распространяется и на формулы расчета центров кластеров и степеней принадлежности:

$$v_{ik} = \frac{\sum\limits_{j=1}^{l}(m_{ij})^{w} e^{B\frac{\alpha_{ij}}{\pi}} x_{jk}}{\sum\limits_{j=1}^{l}(m_{ij})^{w} e^{B\frac{\alpha_{ij}}{\pi}}}, k = \overline{1,n}$$

$$m_{ij} = \frac{1}{(d_{ij}^2 e^{B\frac{\alpha_{ij}}{\pi}})^{\frac{1}{w-1}} \sum\limits_{k=1}^{c} \frac{1}{(d_{kj}^2 e^{B\frac{\alpha_{ij}}{\pi}})^{\frac{1}{w-1}}}} \text{ при } d_{ij} > 0,$$

$$m_{kj} = \begin{cases} 1, & k = i \\ 0, & k \neq i \end{cases} \text{ при } d_{ij} = 0.$$

По итогам работы модифицированного алгоритма нечеткой кластеризации вычисляется компактность разбиения по формуле 1.9. После завершения цикла по количеству кластеров выбирается то, разбиение, которое дало минимальную компактность.

Сравнение качества кластеризации представлено в Таблице 7.

Таблица 7 — Сравнение алгоритмов кластеризации

	Иерархический	k-means++	Fuzzy C-means	Modified Fuzzy C-means
Частота смены кластера	6,3	4,2	1,8	1,4
Компактность	2,8	2,3	1,3	1,1
Период двух кластеров	2,7	3,1	1,7	1,5

Частота смены кластера – общее число смены кластера каждым агентом за весь период симуляции. Для нечетких алгоритмов считается смена основного кластера с порогом 0,6.

Компактность -– среднее значение критерия компактности за весь период симуляции.

Период двух кластеров -– отношение времени симуляции, в течение которого алгоритм определял два кластера, к общему времени симуляции.

Каждая оценка выражена относительным значением: значения каждого критерия нормировались относительно наименьшего, а затем усреднялись по числу прогонов.

В результате использования эволюционного алгоритма кластеризации для помещения длиной 110 метров, шириной 65 метров и количества агентов 150 человек удалось уменьшить время симуляции с одного часа до 8 минут при 20 прогонах модели (рис. 1.10,1.11).

Таким образом, возвращаясь к требованиям, предъявляемым к модели, необходимо указать степень соответствия, а также механизм их реализации (Таблица 8).

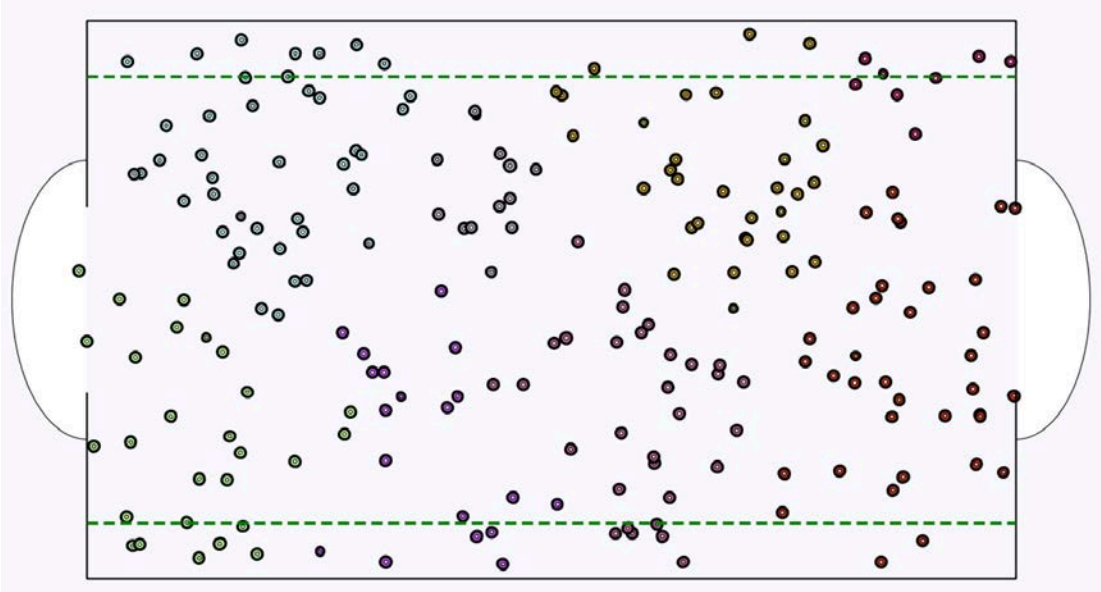


Рисунок 1.10 — Динамика на уровне агентов.

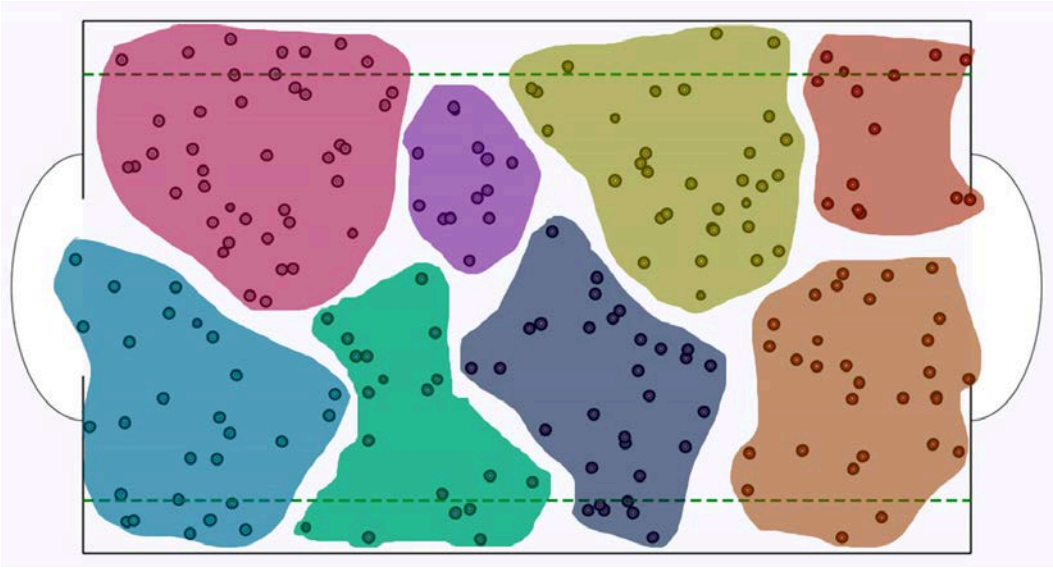


Рисунок 1.11 — Динамика на уровне кластеров.

Таблица 8 — Соответствие требованиям, предъявляемым к разрабатываемой модели и программному комплексу

№	Требование	Механизм реализации
1	*среднее время расчета одного сценария ЧС не более 10 мин.*	Эволюционный алгоритм нечеткой кластеризации
2	*моделирование фронта выхода*	Фронт выхода задан явным образом (см. главу 3)
3	*расчет интенсивности выходного потока на фронте выхода*	Интенсивность потока на фронте выхода (см. главу 3)
4	учет пола/возраста агентов	Индивидуальные характеристики агента, например, комфортная скорость и площадь проекции, зависят от пола и возраста агента
5	наличие случайных препятствий	Взрыв возникает случайно
6	учет влияния взрыва на поведение агентов	В зависимости от динамики ЧС и положения агента у последнего меняется как статус восприятия ЧС, так и его личный статус
7	возможность изменения геометрии помещения	Решение на классах допускает дальнейшее усложнение геометрии помещения
8	учет эффекта “притяжения” толпы	Расчет вектора направления движения, особенно в ситуации отсутствия выхода в зоне видимости, основан именно на направлении движения потока
9	учет эффекта “турбулентности” толпы	Турбулентность, как результат возникновения сверхплотных областей, заложена в явном виде
10	расчет плотности потока	Плотность потока ρ_i

11	наличие личного пространства агента	Радиус личного пространства δ_i
12	расчет для большого количества агентов	Модель поддерживает более 300 агентов
13	прогнозирование количества эвакуированных агентов с точностью 95%	Итоги моделирования с применением эволюционного кластерного анализа дали отклонение меньше 5%
14	разная степень ранения агентов	Статус агента st_i
15	индивидуальное восприятие ситуации агентом	Восприятие ситуации sit_i
16	учет площади проекции агента	Площадь проекции s_i
17	различное начальное распределение агентов	Начальное распределение $\mathbb{F}_{K_l}$
18	учет угла обзора агента	Угол обзора θ_i
19	учет близости агента к стенам	Компонента $wall_i$
20	учет скорости потока	Компонента $v_{i,avg}$
21	учет направления потока	Компонента $angle_i$
22	учет комфортной скорости агента	Компонента $v_{i,comf}$
23	учет многочастичного взаимодействия	Дискритизация времени
24	возможность конфигурирования и масштабирования модели	Реализация модели на классах
25	параметризация модели для имитации различных условий реализации ЧС	Все модельные константы вынесены в качестве параметров

1.4.2 Основные результаты кластерного анализа

Приведенные ниже результаты были получены при значениях $m_{vert} = m_{hor} = 1$. При этом необходимо отметить, что программная реализация допускает задание начального распределения $\mathbb{F}_{K_l}$ из четырех альтернатив: равномерное распределение, треугольное распределение (Симпсона), нормальное распределение (Гаусса) и распределение Лапласа (двойное экспоненциальное), каждое из которых ограничено размерами помещения. Все распределения, кроме равномерного, требуют задание параметров, первый из которых соответствует математическому ожиданию, а второй определяет дисперсию (для треугольного распределения дисперсия не требуется). Каждое распределение задается независимо относительно своих параметров вдоль обеих осей. Задание математического ожидания в центре помещения видится естественным, но сложность проведения экспериментов при непрерывном изменении дисперсии и неясность пределов варьирования влекут необходимость параметризации данных параметров. Математическое ожидание было параметризовано таким образом, чтобы значение варьировалось от минус до плюс единицы при изменении вдоль линейного размера помещения, а значение равное нулю соответствовало центру помещения. Параметризация дисперсии была проведена таким образом, чтобы среднее квадратическое отклонение равное единице соответствовало шестой части линейного размера помещения по соответствующей оси (правило трех сигм) для нормального распределения и четверти размера помещения для распределения Лапласа. Эксперименты проводились при математическом ожидании равном нулю и среднем квадратическом отклонении равном единице.

Благодаря проведенному тесту Грэйнджера на причинность и проверке гипотезы F-тестом при количестве экспериментов $n = 68$ и уровне значимости $\alpha = 0{,}05$, было получено следующее утверждение.

Утверждение 1. *Начальное количество кластеров является функцией количества агентов и начального распределения агентов, но не зависит от конфигурации активного пространства.*

Результатом проведенного анализа t-критерием Стьюдента проверки гипотез при уровне значимости $\alpha = 0{,}05$ стало следующее утверждение.

Утверждение 2. *Динамика количества кластеров есть марковский процесс по отношению к параметрам взаимодействия агентов. Существуют моменты времени t',t'', зависящие от количества агентов, конфигурации активного пространства и начальных распределений такие, что для $t' < t < t''$ количество кластеров стационарно. С момента t'' количество кластеров уменьшается до двух.*

Рисунок 1.12 — Количество кластеров по времени.

При этом стоит отметить справедливость оценки суммы диаметров кластеров.

Утверждение 3. *С момента времени t' сумма диаметров кластеров является убывающей функцией по времени.*

1.5 Выводы

В первой главе был представлен анализ существующих моделей поведения толпы и обозначено место описываемой феноменологической модели. Также было дано формальное описание модели, включающее в себя описание параметров

геометрии помещения, состояний агентов, правил принятия решений и законов взаимодействия агентов. Для случая возникновения чрезвычайной ситуации описаны модификации состояний и уравнений взаимодействия. В построенной модели были учтены такие эффекты, как давка, паника, эффект турбулентности толпы и волны сжатия.

Описан процесс выявления устойчивых групп агентов за счет импортирования в основную модель программного пакета динамической кластеризации, позволяющего в каждый момент модельного времени на множестве всех агентов проводить кластеризацию одновременно по нескольким алгоритмам. С целью повышения качества проводимого кластерного анализа и для уменьшения размерности задачи был разработан модифицированный эволюционный алгоритм нечеткой кластеризации, учитывающий факт наличия препятствия на пути следования агента, а также текущее направление движения агента. Получены результаты, описывающие динамику кластеров, а также определяющие зависимость характеристик кластеров от параметров модели.

Раздел 2. Разработка программного комплекса для реализации агентной имитационной модели поведения толпы и эволюционного алгоритма нечеткой кластеризации

2.1 Подготовка программной реализации

2.1.1 Выбор среды агентного моделирования

Ввиду того, что желаемая модель должна отражать логику и закономерности поведения моделируемого объекта во времени (временная динамика) и пространстве (пространственная динамика), то реализация такой модели должна быть осуществлена в системе имитационного моделирования. Сравнение характеристик современных систем имитационного моделирования представлено на Рисунке 2.1 [36]. Необходимость использования дискретно-событийного моделирования для реализации процесса поочередного принятия решений, необходимость наличия анимации движения агентов, а также использование методов агентного моделирования обуславливают выбор пакета AnyLogic в качестве среды разработки модели.

Необходимо отметить, что несмотря на встроенную в AnyLogic библиотеку пешеходов со своей логикой движения агентов, был разработан новый тип агента, чья система принятия решений основана на минимизации функционала $\mathfrak{L}_i$ и на подходе, описанном в главе 1. Разработка авторского моделирования в среде AnyLogic, как и само программное обеспечение, реализованы на языке Java [52]. Особенностью данного объектно-ориентированного языка программирования является тот факт, что результат разработки в виде программного архива (jar-файл) может работать на любой виртуальной Java-машине вне зависимости от компьютерной архитектуры. Таким образом, достигается универсальность написанного кода, который может быть портирован и запущен на любом компьютере. В частности, можно преобразовать разработанный программный комплекс в Java-апплет, расположенный на серверной стороне, который будет интерактивно выполняться в веб-обозревателе клиентского компьютера, что позволит

Система моделирования	Производитель ПО	Приложения	Моделирующая среда и поддержка			
			Графическая конструкция ИМ	Авторское моделирование, программирование моделей	Анимация (в реальном времени)	Поддержка анализа результатов
ARENA	System Modeling Corporation	Производство, анализ бизнес-процессов, дискретное моделирование	Блок-схемы	+	+	+
EXTEND	Imagine That, Inc.	Стратегическое планирование, бизнес-моделирование	Компоновочные блоки, непрерывные и дискретные модели	+ язык Modl	+	Анализ чувствительности
GPSS/H-PROOF	Wolverine Software Corporation	Общего назначения, производство, транспорт и др.	Блок-схемы	+	+	ANOVA
ITHINK ANALYST	High Performance System, Inc.	Управление финансовыми потоками, реинжиниринг предприятий, банков, инвестиционных компаний и др.	CASE-средства, имитационное моделирование, потоковые диаграммы	+	+	Анализ чувствительности
PROCESS MODEL	PROMODEL Corporation	Общее производство, реинжиниринг	Блок-схемы, дискретное моделирование	-	-	+
SIMUL8	Visual Thinking International	Универсальное средство имитации дискретных процессов	-	Объектно-ориентированное программирование	+	+
TAYLOR SIMULATION SOFTWARE	F&H SimulationInc.	Производство, стоимостный анализ	Блок-схемы, дискретное моделирование	-	+	+
WITNESS	Lanner Group Inc.	Бизнес-планирование, производство, финансы	+	+	+	+ Блок оптимизации
VENSIM	Ventana Systems	Модели системной динамики	Потоковые диаграммы	-	-	-
POWERSIM	Powersin Co.	Непрерывное моделирование	Потоковые диаграммы	-	+	-
DYNAMO	Expectation Software	Модели системной динамики вычислительного типа	Блок-схемы	-	-	-
AnyLogic	XJ Technologies	Системная динамика, Дискретно-событийное, агентное моделирование	Потоковые и UML диаграммы, дискретное моделирование	Java	2D	Оптимизация OptQuest

Рисунок 2.1 — Сравнение характеристик современных систем имитационного моделирования.

обеспечить удаленный доступ к модели без необходимость разворачивания и настройки решения на конкретной машине.

Также важной особенностью среды AnyLogic является наличие широкого спектра инструментов для разработки пользовательского интерфейса, что позволяет значительно упростить и сделать более понятным для конечного пользователя как процесс задания начальных параметров, так и сам процесс моделирования. При этом, благодаря интеграции решения с базой данных, удается преобразовать плоскую анимацию движения агентов в режим 3D. При этом внутренние процессы сериализации и синхронизации, встроенные в AnyLogic, позволяют добиться высокой скорости расчетов и моделирования движения агентов

на стандартном персональном компьютере даже в условиях высокой сложности самой модели.

2.1.2 Разработка базы данных

С целью хранения входных параметров агентной модели, задаваемых пользователем, а также для хранения динамики характеристик агентов в процессе моделирования, была разработана база данных (далее, БД) под управлением СУБД (система управления базы данных) MS SQL Server. Выбор СУБД был обоснован, с одной стороны, необходимостью развертывания решения на компьютере под управлением ОС Windows, а с другой, такой критически важной характеристикой, как скорость обработки массовых транзакций, которые сопровождают процесс моделирования.

Сохранение в БД динамики состояния агента позволяет в последствии воссоздать движение агента по координатам и другим характеристикам и провести кластерный анализ. Также подобный подход упрощает дальнейший анализ результатов моделирования, например, в процессе анализа интенсивности потока на “фронте выхода”, о котором пойдет речь в главе 3.

Помимо этого, немаловажным фактором при выборе СУБД, являлась возможность осуществления бесшовного соединения системы AnyLogic с MS SQL Server посредством JDBC драйвера. В итоге, построенная архитектура решения позволяет сохранять пользовательские настройки, реплицировать и восстанавливать состояния агентов и проводить различный анализ итогов моделирования для каждого прогона в среде моделирования.

Структура БД представляет из себя набор из двенадцати таблиц, представленных на Рисунке 2.2.

В таблице *square* содержатся исходные данные о базовом значении площади горизонтальной проекции в зависимости от возраста (рис. 2.3). Таблица имеет следующие атрибуты:

– Идентификатор записи;
– Возраст;
– Базовое значение площади горизонтальной проекции.

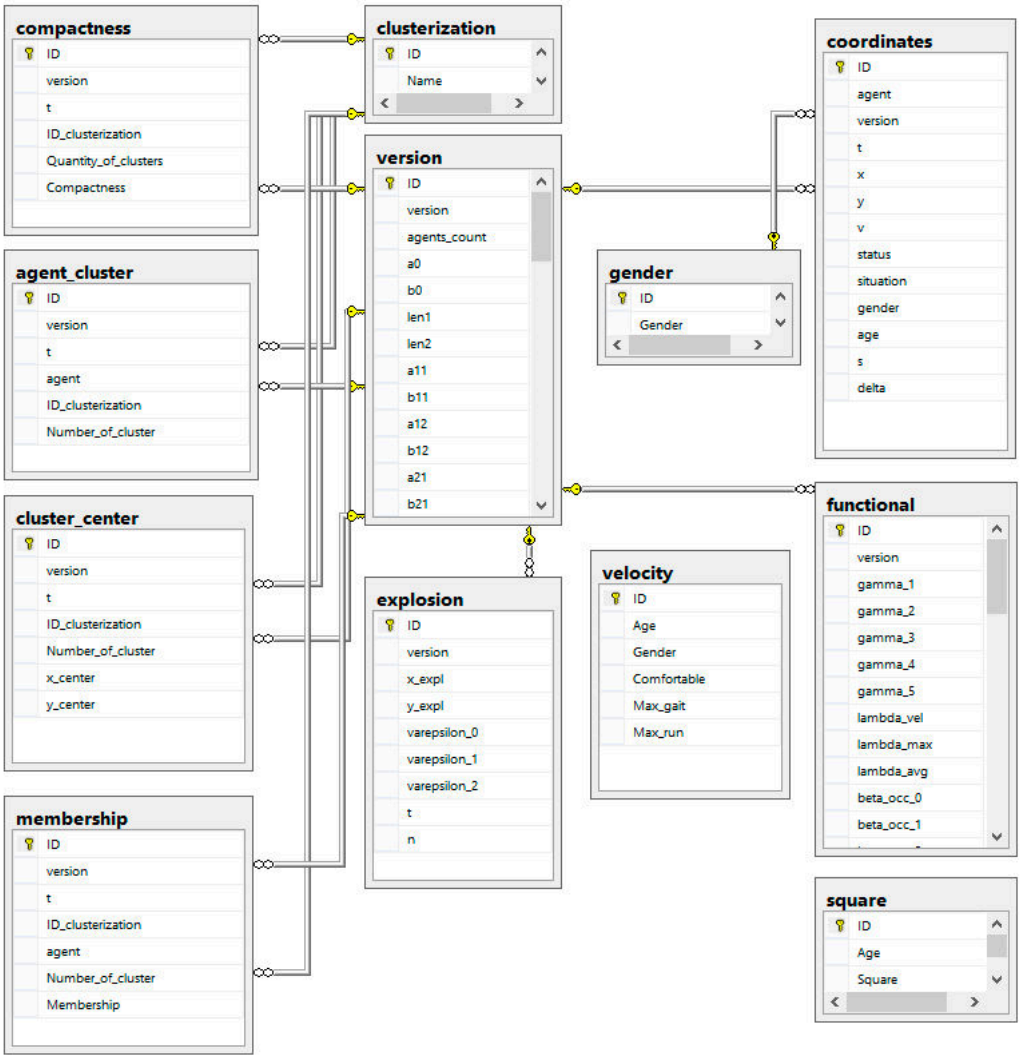


Рисунок 2.2 — Схема БД.

Таблица *gender* представляет из себя справочник, в котором содержится информации о кодах полов (мужском и женском) (рис. 2.4). Таблица имеет следующие атрибуты:

– Идентификатор записи;
– Пол.

В таблице *velocity* содержатся исходные данные о комфортной и максимальной скорости ходьбы, а также о максимальной скорости бега в зависимости от возраста и пола агента (рис. 2.5). Данные в таблице имеют единицы измерения “метры в секунду”, а преобразование в пиксели происходит уже на стороне модели. Таблица имеет следующие атрибуты:

– Идентификатор записи;
– Возраст;
– Пол;

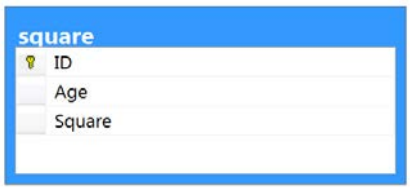


Рисунок 2.3 — Таблица *square*.

Рисунок 2.4 — Таблица *gender*.

– Комфортная скорость ходьбы;
– Максимальная скорость ходьбы;
– Максимальная скорость бега.

Рисунок 2.5 — Таблица *velocity*.

В таблице *version* содержатся исходные данные о параметрах геометрии помещения, а также о параметрах плотности и радиуса личного пространства в зависимости от версии прогона модели (рис. 2.6). Таблица имеет следующие атрибуты:

– Идентификатор записи;
– Версия прогона модели;
– Общее количество агентов;
– Длина помещения;
  ...
– Параметр κ_3;
– Дата прогона;
– Количество единиц модельного времени в одной секунде.

В таблице *functional* содержатся исходные данные о весах и нормирующих коэффициентах функционала $\mathfrak{L}$ (рис. 2.7). Таблица имеет следующие атрибуты:

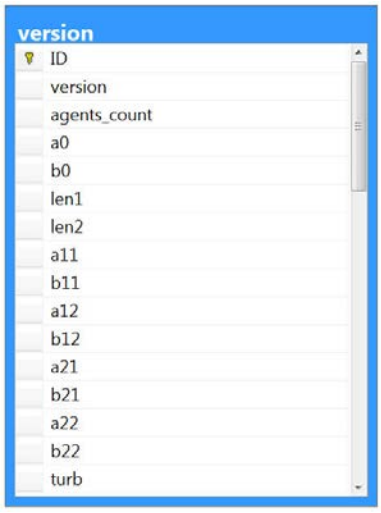


Рисунок 2.6 — Таблица *version*.

– Идентификатор записи;
– Версия прогона модели;
– Параметр γ_1;
– Параметр γ_2;

  ...

– Параметр $\beta_{expl}(3)$;
– Дата прогона.

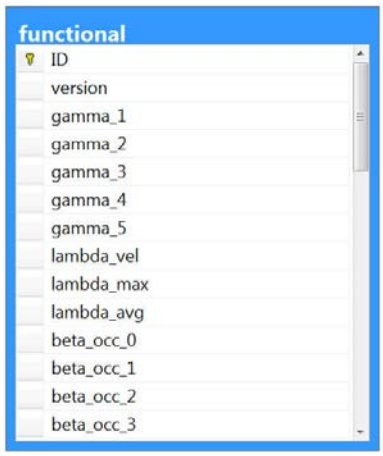


Рисунок 2.7 — Таблица *functional*.

В таблице *coordinates* содержатся данные об изменении координат и других характеристик агента в процессе моделирования (рис. 2.8). Здесь стоит отметить тот факт, что координаты хранятся в своем абсолютном выражении, а не относительно координат стен помещения. Скорость агента, также как и в таблице *version*, хранится в метрах в секунду, а обратное преобразование из пикселей происходит на стороне модели. Таблица имеет следующие атрибуты:

– Идентификатор записи;
– Номер агента;
– Версия прогона модели;
– Модельное время;
– x координата;
– y координата;
– Скорость агента;
– Статус агента;
– Статус ситуации;
– Пол;
– Возраст.

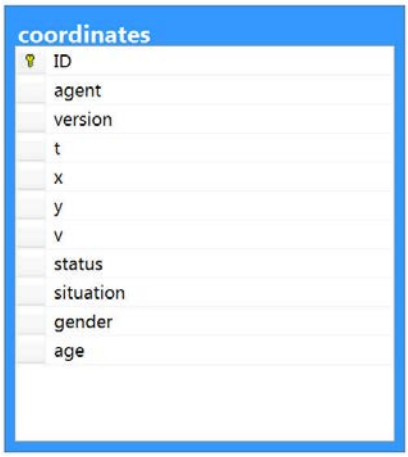


Рисунок 2.8 — Таблица *coordinates*.

В таблице *explosion* содержатся данные о взрыве, произошедшем в процессе моделирования (рис. 2.9). Радиусы концентрических окружностей взрыва хранятся в единицах метров, а преобразование в пиксели происходит уже на стороне модели. Таблица имеет следующие атрибуты:

– Идентификатор записи;
– Версия прогона модели;
– x координата взрыва;
– y координата взрыва;
– радиус ε_0;
– радиус ε_1;
– радиус ε_2;
– Модельное время;
– Порядковый номер взрыва.

Рисунок 2.9 — Таблица *explosion*.

Таблица *clusterization* представляет из себя справочник, в котором содержится информации о методах кластеризации (рис. 2.10). Более подробно о применяемых алгоритмах кластеризации было рассказано в главе 1. Таблица имеет следующие атрибуты:

– Идентификатор записи;
– Метод кластеризации.

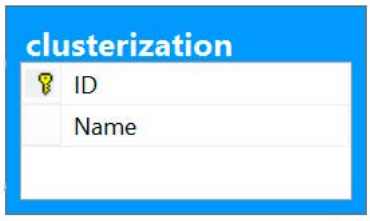


Рисунок 2.10 — Таблица *clusterization*.

В таблице *compactness* содержатся данные об изменении плотностей кластеров в процессе моделирования для каждого алгоритма кластеризации (рис. 2.11). При этом данные сохраняются для каждой версии прогона независимо. Таблица имеет следующие атрибуты:

– Идентификатор записи;
– Версия прогона модели;
– Модельное время;
– Идентификатор алгоритма кластеризации;
– Количество кластеров;
– Компактность кластеров.

В таблице *agent_cluster* содержатся данные об изменении принадлежности агента к кластеру в процессе моделирования (рис. 2.12). Для каждого алгоритма

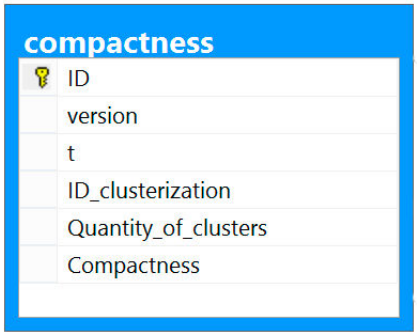


Рисунок 2.11 — Таблица *compactness*.

кластеризации принадлежность к кластеру рассчитывается независимо. Таблица имеет следующие атрибуты:

– Идентификатор записи;
– Версия прогона модели;
– Модельное время;
– Номер агента;
– Идентификатор алгоритма кластеризации;
– Порядковый номер кластера.

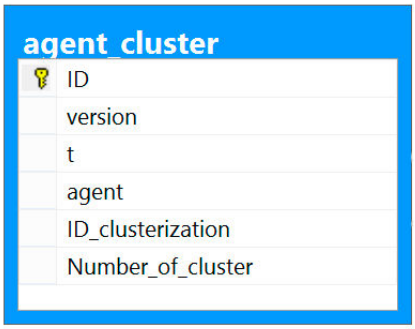


Рисунок 2.12 — Таблица *agent_cluster*.

В таблице *cluster_center* содержатся данные об изменении координат центров кластеров в процессе моделирования (рис. 2.13). Здесь стоит отметить тот факт, что координаты хранятся в своем абсолютном выражении, а не относительно координат стен помещения. Для каждого алгоритма кластеризации центры кластеров рассчитываются независимо. Таблица имеет следующие атрибуты:

– Идентификатор записи;
– Версия прогона модели;
– Модельное время;
– Идентификатор алгоритма кластеризации;
– Порядковый номер кластера;

– x координата центра кластера;
– y координата центра кластера.

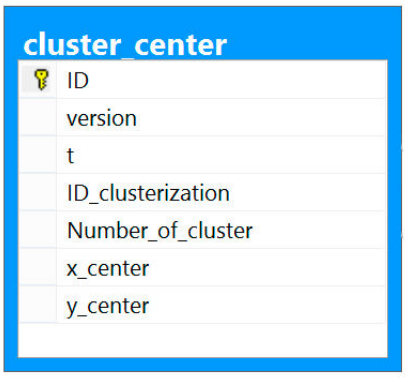


Рисунок 2.13 — Таблица *cluster_center*.

В таблице *membership* содержатся данные о мере принадлежности агентов к кластеру для алгоритма нечеткой кластеризации (рис. 2.14). Мера принадлежности выражена числом из интервала $[0,1]$. Таблица имеет следующие атрибуты:

– Идентификатор записи;
– Версия прогона модели;
– Модельное время;
– Идентификатор алгоритма кластеризации;
– Номер агента;
– Порядковый номер кластера;
– Мера принадлежности к кластеру.

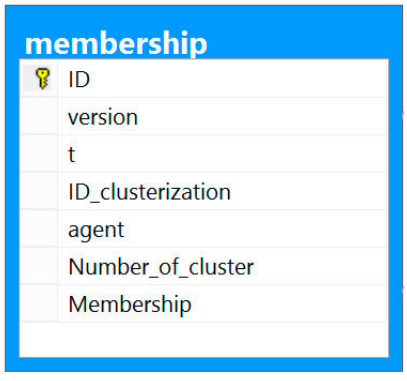


Рисунок 2.14 — Таблица *membership*.

2.2 Разработка агентной модели толпы в среде AnyLogic

Структурно разработанная агентная модель состоит из следующих классов:

- Активные классы модели
    - Пользовательский интерфейс (Simulation).
    - Основной класс (Main);
    - Агент (Person);
    - Взрыв (Explosion);
- Вспомогательные классы модели
    - Столкновения (Collision);
    - Принятие решения (Decision);
    - Точки пространства (MyPoint);
    - Векторы пространства (MyVector);
    - Возвращаемый результат (Result).

Пользовательский интерфейс (Simulation)

Данный класс предназначен для задания входных параметров пользователем. При старте модели можно наблюдать интерфейс, представленный в приложении Б.1. Все входные параметры моделирования разбиты на шесть групп (рис. 2.15). Код данного класса, включающий верификацию введенных данных, представлен в приложении Б.2.

Меняя значения параметров в текстовых полях, пользователь имеет возможность менять как характеристики помещения и количества агентов, так и параметры, отвечающие за систему принятия решений агента. Таким образом можно, с одной стороны, подобрать значения параметров для наиболее достоверной динамики агентов и тем самым провести калибровку модели, а с другой, появляется возможность варьировать психо-физические аспекты поведения агентов для подбора тех значений параметров, которые обеспечат максимизацию количество спасенных агентов. Другими словами, появляется возможность разработки системы руководств к действию для агентов попавших в ЧС с целью повышения вероятности выживания. Более подробно вопрос подбора параметров и коэффициентов для калибровки модели рассмотрен в параграфе 2.4.

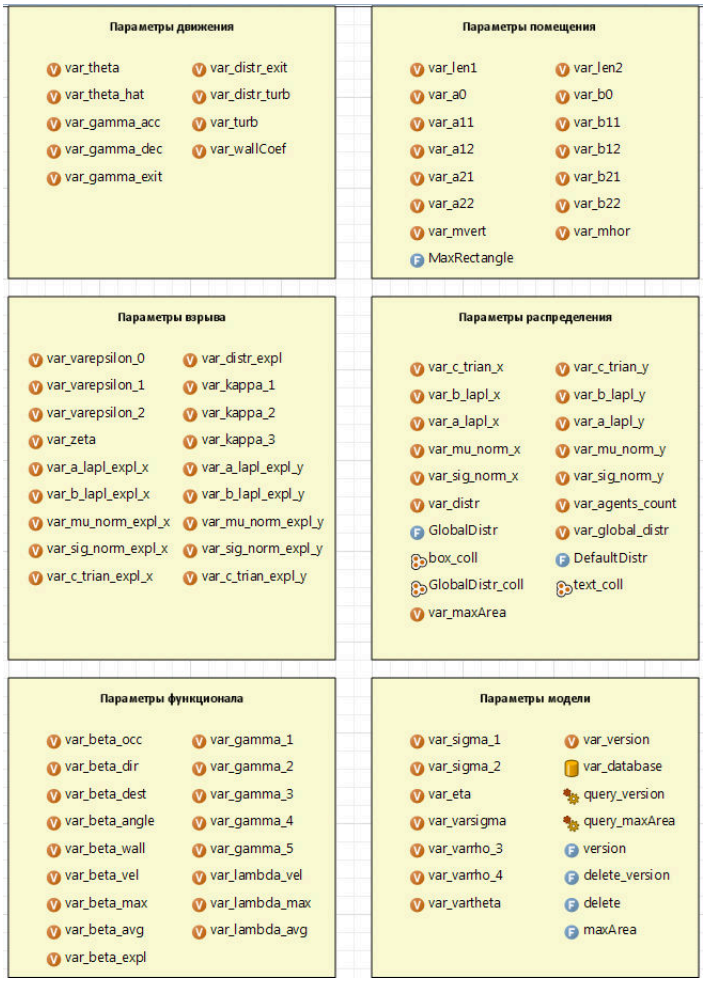


Рисунок 2.15 — Группы параметров класса *Simulation*.

В рамках каждого текстового поля, управляемого пользователем, встроена проверка как допустимости вводимых данных, так и каскадное обновление зависимых полей. Ниже представлен фрагмент кода подобной проверки.

```
// каскадное обновление значений других текстовых полей для сохранения логики
    соотношений между ними
if (box_vartheta_4.getDoubleValue()<0.7) {
  box_vartheta_4.setText(0.7, true);
}
if (box_vartheta_4.getDoubleValue()<box_vartheta_1.getDoubleValue()) {
  box_vartheta_4.setText(box_vartheta_1.getDoubleValue(), true);
}
if (box_vartheta_4.getDoubleValue()<box_vartheta_2.getDoubleValue()) {
  box_vartheta_4.setText(box_vartheta_2.getDoubleValue(), true);
}
if (box_vartheta_4.getDoubleValue()<box_vartheta_3.getDoubleValue()) {
  box_vartheta_4.setText(box_vartheta_3.getDoubleValue(), true);
}
var_vartheta[3]=box_vartheta_4.getDoubleValue();
```

В случае выбора пользователем ручного режима распределения количества агентов по клеткам открывается соответствующая область заполнения (рис. 2.16).

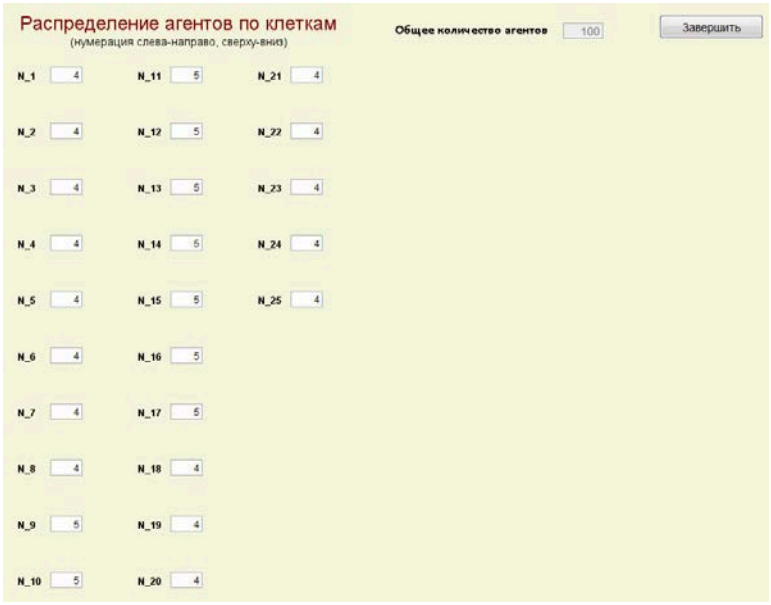


Рисунок 2.16 — Область заполнения количества агентов в клетках.

Также при смене типа распределения того или иного стохастического параметра происходит динамическое замена характеристик распределения (рис. 2.17).

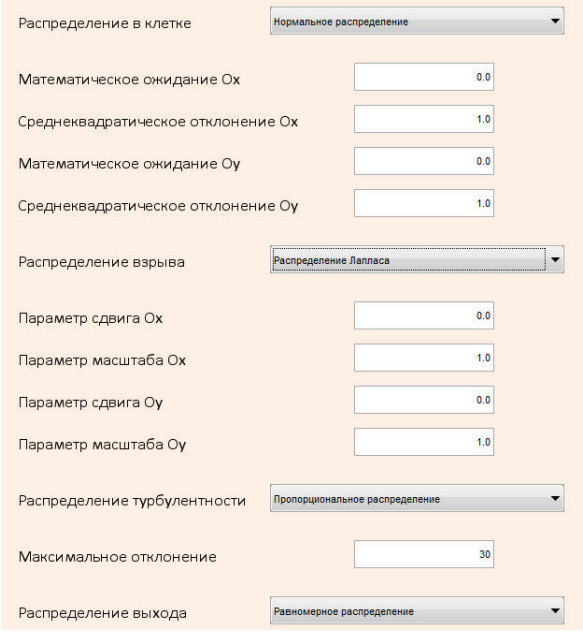


Рисунок 2.17 — Динамическое отображение характеристик для разных распределений.

При завершении работы класса *Simulation* происходит передача входных параметров моделирования, введенных пользователем, в класс *Main*, а также

проверка непротиворечивость БД на предмет уникальности текущей версии прогона модели.

Основной класс (Main)

Данный класс является средой исполнения всех остальных классов и содержит в себе как процесс отрисовки помещения, так и процесс движения и взаимодействия агентов. Интерфейс класса *Main* представлен в приложении В.1. Все входные и расчетные параметры моделирования данного класса разбиты на восемь групп (рис. 2.18). Код класса *Main* представлен в приложении В.2.

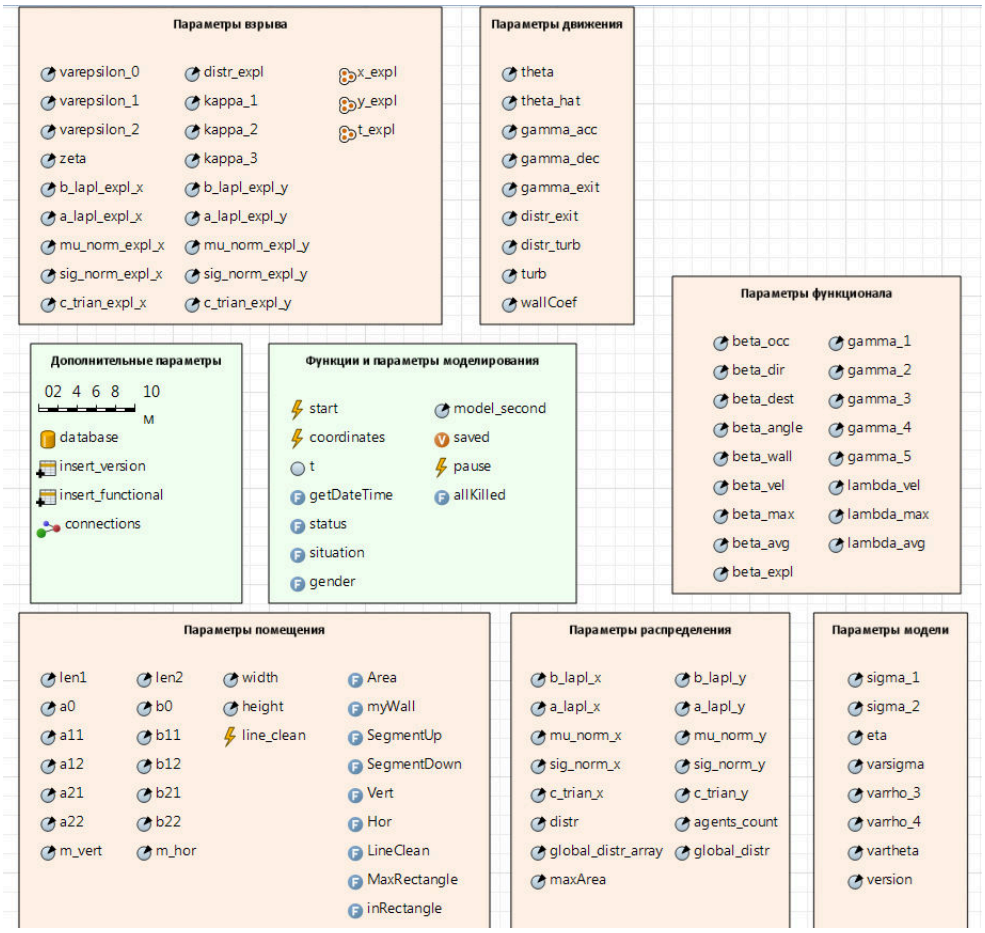


Рисунок 2.18 — Группы параметров класса *Main*.

Также в классе *Main* осуществляется управление появлением взрыва и переходом к окну статистики. Интерфейс окна статистики представлен в приложении В.3.

Класс агента (Person)

Данный класс реализует популяцию агентов-пешеходов. Интерфейс класса *Person* представлен в приложении Г.1. В качестве логики работы данного класса была выбрана схема синхронизации движения агентов, в рамках которой про-

цесс моделирования движения разбивается на шаги, а действия выполняются в следующем порядке:

1. Выполняется Действие перед выполнением шага класса *Main*.
2. У всех агентов выполняются Действия перед выполнением шага (в каком-то определенном порядке).
3. В том же самом порядке у всех агентов выполняются Действия на шаге.
4. Выполняется Действие после выполнения шага класса *Main*.

Код класса *Main* представлен в приложении Г.2.

Класс взрыва (Explosion)

Класс *Explosion* задает визуализацию взрыва, а также его геометрические характеристики – радиусы концентрических окружностей поражения агентов, которые в момент запуска класса *Main* получают свои фактические значения. Интерфейс класса *Main* представлен в приложении Д.1. Код класса *Explosion* представлен в приложении Д.2.

Класс столкновений (Collision)

Данный класс предназначен для обработки события столкновения агентов в процессе моделирования. Класс представлен двумя методами: регистрация факта взаимодействия и расчет результата взаимодействия агентов, согласно законам сохранения импульса и кинетической энергии. Код класса *Collision* представлен в приложении Е.1.

Класс принятия решения (Decision)

Данный класс реализует процесс принятия решения агентом. Результатом работы является единичный вектором нового направления движения агента, а также его новая скорость. Методы данного класса рассчитывают как возможность перехода в тот или иной сектор, так и компоненты функционала $\mathfrak{L}$. Завершающий метод класса высчитывает аргументы, минимизирующие функционал, и возвращает соответствующие значения. Код класса *Decision* представлен в приложении Ж.1.

Класс точек пространства (MyPoint)

Данный класс предназначен для описания точек пространства как пары значений координат точки. Также в рамках данного класса реализован метод, проверяющий отрезки на факт пересечения. Данный метод используется в процессе проверки выхода агента за пределы помещения. Код класса *MyPoint* представлен в приложении З.1.

Класс векторов (MyVector)

В этом классе реализовано векторное исчисление пространства. Помимо непосредственного задания векторов различными конструкторами, данный класс реализует такие методы, как скалярное произведение векторов, косое произведение, нормирование вектора, растяжение, поворот вектора, вычисление угла между векторами и т.д. Код класса *MyVector* представлен в приложении И.1.

Класс результатов (Result)

Данный класс предназначен для описания результатов, возвращаемых вспомогательными классами. Поля данного класса представлены единичным вектором нового направления движения агента, а также его новой скоростью. Возвращать данный класс может как вектор целиком, так его отдельные компоненты. Код класса *Result* представлен в приложении К.1.

2.3 Архитектура программного комплекса

Разработанный комплекс программ, обеспечивающий реализацию предложенной агентной модели поведения толпы в условиях чрезвычайных ситуаций с использованием эволюционного алгоритма нечеткой кластеризации основан на использовании следующих важнейших технологий:

1. *Имитационная модель толпы в среде AnyLogic* — обеспечивающая численную реализацию на объектно-ориентированном языке программирования высокого уровня Java разработанную агентную модель поведения толпы в условиях ЧС, интегрированную с предложенным эволюционным алгоритмом нечеткой кластеризации.
2. *Набор классов Java*, разработанных для агентной модели поведения толпы в условиях чрезвычайных ситуаций, в частности, следующих классов:
   - Пользовательский интерфейс модели;
   - Основной класс модели;
   - Класс агента;
   - Класс взрыва;
   - Класс столкновений;

- Класс принятия решения;
- Класс точек пространства;
- Класс векторов пространства;
- Класс результатов.

3. *СУБД MS SQL Server* — реляционная база данных, обеспечивающая хранение и обработку данных имитационной модели, в том числе, исходных значений и результатов имитационного моделирования необходимых для последующего кластерного анализа.
4. *JDBC* — стандарт взаимодействия Java-приложений с различными СУБД, в том числе, используемой СУБД MS SQL Server.
5. *Эволюционный алгоритм нечеткой кластеризации* (реализованный в виде подключаемой программной библиотеки – JAR-файла), в частности, обеспечивающая реализацию и применение данного алгоритма для предложенной агентной модели поведения толпы.
6. *Подсистема статистической обработки* результатов кластерного анализа в среде RStudio, поддерживающей язык программирования R и интегрированная с базой данных системы (MS SQL Server).

Отметим, что комплексы программ (интегрированные подсистемы) 1, 2, 3 и 5 предложены впервые. Подобная комбинация позволяет принципиально повысить эффективность оценки потока толпы на выходе фронта за счет перехода от динамики толпы к динамике кластеров, идентифицируемых с использованием предложенного эволюционного алгоритма нечеткой кластеризации.

Как было отмечено ранее, платформа AnyLogic является самостоятельным коммерческим продуктом, обеспечивающим возможность программной реализации сложных имитационных моделей агентного типа. Несмотря на наличие в данной системе готовых библиотек, реализующих стандартную логику движения людей в окружающей среде (например, пешеходных потоков “pedestrian library”), в предложенной имитационной модели подобные библиотеки не используются, так как они не учитывают влияние динамики ЧС, в том числе, особенности взаимодействия агентов при развитии ЧС, а также сложные коллизии, влияющие на динамику агентов.

Для предложенной имитационной модели поведения толпы разработана реляционная база данных, схема которой представлена на Рисунке 2.2, позволяющая, в частности, сохранять сценарную динамику состояния агентов для це-

лей дальнейшего кластерного анализа. Данная СУБД бесшовно интегрирована с разработанной имитационной модели поведения толпы с помощью стандартного интерфейса JDBC, позволяющего, в частности, взаимодействовать с базой данных посредством SQL-запросов (на запись, чтение и обновление данных).

Важнейшей компонентой спроектированного программного комплекса является набор разработанных с использованием языка программирования Java классов, таких как, класс агента, класс столкновений, класс принятия решения и др. Подобные классы, реализованные с использованием объектно-ориентированного подхода на языке программирования Java, могут быть имплементированы в различные компьютерные модели поведения толпы агентного типа (также разработанные на Java), и поэтому, являются в определенным смысле реферетными (универсальными) классами, применяемыми для различных вариантов построения имитационной модели (например, при различных конфигурациях окружающего пространства, для различного числа агентов, и т.д.).

Ключевым элементом предложенного программного комплекса является *эволюционный алгоритм нечеткой кластеризации*, реализованный в виде подключаемой программной библиотеки, интегрированной с разработанной имитационной моделью поведения толпы. Особенностью данной программной библиотеки является программная реализация эволюционного алгоритма нечеткой кластеризации с использованием языка программирования Java, и поддержка параметрических функций, обеспечивающих динамическую идентификацию кластеров толпы, на основе данных о ее динамике, с учетом пространственного расположения агентов относительно друг друга. В результате, определение кластеров является более точным по сравнению с другими известными методами кластеризации применительно к задаче кластеризации толпы. Кроме, того использование данной программной библиотеки позволяет перейти от динамики агентов к динамике (центров) кластеров, и, таким образом, принципиально уменьшить размерность данной задачи (зависящую от числа агентов), обеспечивая численное моделирование развития ЧС за полиномиальное время. Для этого, библиотека содержит специальную функцию, которая на основе данных о динамике агентов, сохраняемых в БД, формирует динамику центров кластеров непосредственно в имитационной модели AnyLogic, существенно сокращая, таким образом, время на проведение экспериментов с моделью.

Как видно из Таблицы 9, за счет применения эволюционного алгоритма нечеткой кластеризации удается повысить временную эффективность модели, при этом рост времени симуляции при кластерной динамике в зависимости от числа агентов и начального распределения значительно ниже, чем при динамике модели на уровне агентов (приведена усредненная статистика по 100 прогонам для каждого случая).

Таблица 9 — Сравнение эффективности работы модели

Ширина помещения, м	Длина помещения, м	Число агентов, чел	Начальное распределение	Тип динамики	Время симуляции, мин
110	65	150	Равномерное	Агент	40
110	65	150	Равномерное	Кластер	8
110	65	150	Нормальное	Агент	55
110	65	150	Нормальное	Кластер	10
110	65	300	Равномерное	Агент	100
110	65	300	Равномерное	Кластер	15
110	65	300	Нормальное	Агент	112
110	65	300	Нормальное	Кластер	17
200	100	300	Равномерное	Агент	150
200	100	300	Равномерное	Кластер	21
200	100	300	Нормальное	Агент	170
200	100	300	Нормальное	Кластер	23

Последней подсистемой является модуль статистической обработки результатов кластерного анализа в среде RStudio, поддерживающей язык программирования R и интегрированная с базой данных системы (MS SQL Server). Использование данной подсистемы позволяет применить к результатам имитационного моделирования различные методы кластерного анализа, в том числе, следующие:

1. Метод k-средних (k-means++).
2. Иерархический метод.
3. Метод нечеткой кластеризации C-средних (Fuzzy C-means).

В результате можно дополнить предложенный метод нечеткой кластеризации другими известными методами, с целью уточнения полученных результатов и оценки эффективности разработанного алгоритма.

Архитектура спроектированного программного комплекса представлена на рис. 2.19.

Рисунок 2.19 — Архитектура программного комплекса.

2.4 Выводы

Во второй главе было дано обоснование выбора среды разработки агентной модели поведения толпы, приведено описание структуры базы данных и описано назначение отдельных сущностей. В рамках конечной реализации агентной модели дано описание разработанных классов, их предназначения, а также описан пользовательский интерфейс. Приведена архитектура всего программного комплекса, включающая в себя отдельные модули системы, точки интеграции и связи между ними.

Раздел 3. “Фронт выхода” толпы и оценка интенсивности потока

3.1 Понятие “фронта выхода”. Уравнение максимальной интенсивности потока и порождаемая краевая задача

Ввиду того, что, пройдя через раствор выхода, агент продолжает влиять на модель (он продолжает оставаться препятствием для находящихся позади него агентов, что влечет изменение поведения всей группы агентов), то в качестве окончательной точки выбытия агента из модели на основе проведенных экспериментов была выбрана дуга эллипса – фронт выхода. Также, очевидно, что с течением времени и в случае большого количества агентов будут образовываться скопления у каждого из выходов (см. рис. 3.1). Ввиду ограничен-

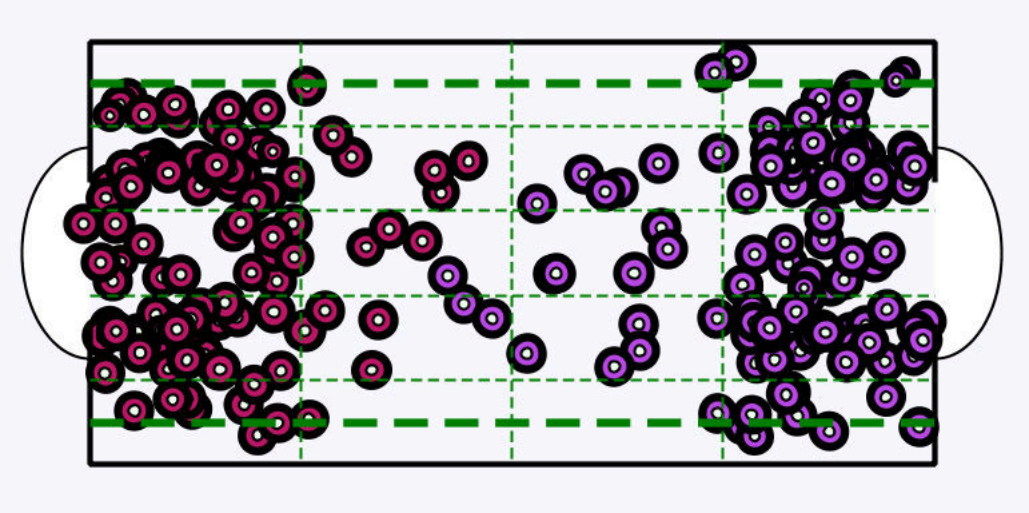


Рисунок 3.1 — Скопление агентов у выходов.

ной пропускной способности раствора выхода, настанет момент максимальной интенсивности прохождения агентов через фронт выхода. При этом саму интенсивность прохождения агентов, то есть изменение количества агентов, прошедших через дугу эллипса за единицу времени, мы назовем интенсивностью фронта выходного потока. Период указанной максимальной интенсивности отличается невозможностью улучшений показателей эвакуации, что, в свою очередь, говорит об отсутствии необходимости вмешательства со стороны служб спасения, так как попытка проникнуть в помещения против максимального потока чревато еще большими жертвами. С другой стороны, падение интенсивно-

сти (а точнее, скорость падения) может свидетельствовать как о естественном уменьшении числа агентов в помещении (что не требует дополнительного вмешательства со стороны спасательных служб), так и о чрезвычайных процессах (давка, турбулентность и пр.), требующих экстренного вмешательства. Таким образом, регистрация интенсивности фронта выходного потока и сравнение этой интенсивности с аналитически полученной интенсивностью для максимального потока дает возможность спасателям оценить и стадию процесса эвакуации, и степень необходимости своего вмешательства. Функция плотности, соответствующая максимальной интенсивности фронта выходного потока, может быть описана, как решение следующей краевой задачи:

$$\begin{cases} \triangle u & = 0 \ \text{ в } \Omega \\ u|_{\partial\Omega} & = \varphi, \end{cases} \tag{3.1}$$

где $u(x,y)$ – усредненная плотность агентов в области Ω, а $\varphi \in L^1_{2,loc}(\Omega)$ – усредненная плотность на границе неограниченной области Ω. Сама область Ω представляет из себя неограниченную область, заданную фронтом выхода (граница эллипса), стенами раствора выхода и двумя горизонтальными прямыми, параллельными стенам, но отстоящие от них на некотором расстоянии (см. рис. 3.2). При этом область считается бесконечной в сторону противоположную выходу. Такой выбор области объясняется тем, что прибывающие с “бесконечности” агенты никак не влияют на максимальную интенсивность потока, а только лишь поддерживают его состояние.

3.2 Теорема существования для краевой задачи

3.2.1 Основные понятия и предварительные конструкции

В дальнейшем рассматривается первая краевая задача для эллиптических систем, заданных в неограниченных областях $\Omega \subset \mathbb{R}^n$, решения которых удовлетворяют условию конечности интеграла Дирихле, называемого также интегра-

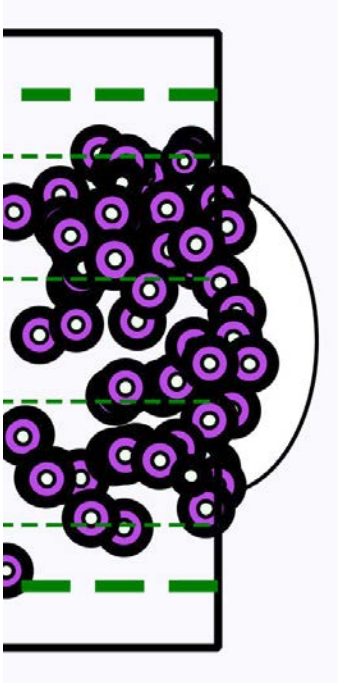


Рисунок 3.2 — Фронт выхода.

лом энергии

$$\int\limits_{\Omega} |\nabla u|^2 dx < \infty.$$

Пусть Ω – произвольное открытое множество в $\mathbb{R}^n$. Как это принято, через $W^1_{2,\,loc}(\Omega)$ обозначим пространство функций локально являющихся соболевскими, т.е.

$$W^1_{2,\,loc}(\Omega) = \{f : f \in W^1_2(\Omega \cap B^x_\rho)\, \forall\, \rho > 0\,,\, \forall\, x \in \mathbb{R}^n\}\,,$$

где B^x_ρ – открытый шар с центром в точке x и радиусом ρ. В случае $x = 0$ будем писать B_ρ. Будем обозначать через $\overset{\circ}{W}{}^1_{2,\,loc}(\Omega)$ множество функций из $W^1_{2,\,loc}(\,\mathbb{R}^n)$, которое получается замыканием $C^\infty_0(\Omega)$ в системе полунорм $\|u\|_{W^1_2(\mathcal{K})}$, где $\mathcal{K} \subset \mathbb{R}^n$ – всевозможные компакты. Через $L^1_2(\Omega)$ обозначим пространство обобщенных функций в Ω, первые производные которых принадлежат пространству $L_2(\Omega)$ ([31] с.12), другими словами

$$L^1_2(\Omega) = \{f \in \mathcal{D}'(\Omega) : \int\limits_{\Omega} |\nabla f|^2 dx < \infty\}.$$

Пусть $\omega \subseteq \mathbb{R}^n$ – открытое множество, $\mathcal{K} \subset \omega$ – компакт. Обозначим за $\mathfrak{M}_\varphi(\mathcal{K},\omega)$ множество функций $\psi \in C^\infty_0(\omega)$ таких, что $\psi = \varphi$ в окрестности $\mathcal{K}$,

или другими словами $\psi - \varphi \in \overset{\circ}{W}{}^{1}_{2,\,loc}(\mathbb{R}^n \setminus \mathcal{K})$. Обозначим за $\mathfrak{R}(\mathcal{K},\omega)$ множество $\mathfrak{R}(\mathcal{K},\omega) = \{\psi \in C_0^\infty(\omega) : \psi = 1$ в окрестности $\mathcal{K}\}$.

Емкостью компакта $\mathcal{K}$ относительно множества ω назовем величину ([31] с. 293):

$$\operatorname{cap}_\varphi(\mathcal{K},\omega) = \inf_{\psi \in \mathfrak{M}_\varphi(\mathcal{K},\,\omega)} \int\limits_\omega |\nabla\psi|^2 dx\,.$$

Емкость произвольного замкнутого в $\mathbb{R}^n$ множества $E \subset \omega$ определяется формулой

$\operatorname{cap}_\varphi(E,\omega) = \sup\limits_{\mathcal{K}\subset E} \operatorname{cap}_\varphi(\mathcal{K},\omega)$. Если $\omega = \mathbb{R}^n$, то вместо $\operatorname{cap}_\varphi(E,\mathbb{R}^n)$ будем писать $\operatorname{cap}_\varphi(E)$.

Нам также потребуется следующая емкость ([31] с. 326):

$$\operatorname{Cap}(\mathcal{K},W_2^1(\omega)) = \inf_{\psi \in \mathfrak{R}(\mathcal{K},\,\omega)} \left(\int\limits_\omega |\psi|^2\,dx + \int\limits_\omega |\nabla\psi|^2 dx \right)\,.$$

Емкость произвольного замкнутого в $\mathbb{R}^n$ множества $E \subset \omega$ определяется формулой $\operatorname{Cap}(E,W_2^1(\omega)) = \sup\limits_{\mathcal{K}\subset E} \operatorname{Cap}(\mathcal{K},W_2^1(\omega))$.

Из определения емкости $\operatorname{cap}_\varphi(E,\omega)$ непосредственно вытекают следующие два свойства.

Монотонность. Если $e_1 \subset e_2$ и $\Omega_1 \supset \Omega_2$, то

$$\operatorname{cap}_\varphi(e_1,\Omega_1) \le \operatorname{cap}_\varphi(e_2,\Omega_2).$$

Непрерывность справа. Для каждого $\varepsilon > 0$ существует такая окрестность ω, $\overline{\omega} \subset \Omega$ компакта e, что для любого компакта e', $e \subset e' \subset \omega$

$$\operatorname{cap}_\varphi(e',\Omega) \le \operatorname{cap}_\varphi(e,\Omega) + \varepsilon.$$

Обозначим через W_2^{-1} пространство линейных непрерывных функционалов на W_2^1. Множество $E \subset \mathbb{R}^n$ назовем $(2,1)$-полярным, если единственным элементом из W_2^1 с носителем в E является нуль ([31] с. 331).

3.2.2 Постановка задачи

Здесь и далее L – дивергентный эллиптический оператор, то есть

$$L = \sum_{i,j=1}^{n} \frac{\partial}{\partial x_i} \left(a_{ij}(x) \frac{\partial}{\partial x_j} \right) ,$$

где a_{ij} – измеримые ограниченные функции, удовлетворяющие условию

$$c_1|\xi|^2 \leq \sum_{i,j=1}^{n} a_{ij}(x)\, \xi_i\, \xi_j \leq c_2|\xi|^2 \quad \xi \in \mathbb{R}^n, \quad c_1, c_2 > 0.$$

Решением задачи Дирихле

$$\begin{cases} Lu & = 0 \ \text{ в } \Omega \\ u|_{\partial\Omega} & = \varphi, \end{cases} \tag{3.2}$$

где $\varphi \in W^1_{2,\,loc}(\mathbb{R}^n)$, называется функция $u \in W^1_{2,\,loc}(\Omega)$ такая что:

1) $u - \varphi \in \overset{\circ}{W}{}^1_{2,\,loc}(\Omega)$, т.е. $(u-\varphi)\mu \in \overset{\circ}{W}{}^1_2(\Omega)$ для любой функции $\mu \in C_0^\infty(\,\mathbb{R}^n)$;

2) функция u имеет ограниченный интеграл Дирихле

$$\int\limits_{\Omega} |\nabla u|^2 dx < \infty\,;$$

3)

$$\int\limits_{\Omega} \sum_{i,j=1}^{n} a_{ij}(x) \frac{\partial u}{\partial x_j} \frac{\partial \psi}{\partial x_i}\, dx = 0$$

для любой функции $\psi \in C_0^\infty(\Omega)$.

3.2.3 Основные результаты

Сформулируем ряд утверждений относительно краевой задачи (3.2).

Теорема 1. *Пусть* $cap_{\varphi - c}(\mathbb{R}^n \setminus \Omega) < \infty$ *для некоторой константы* $c \in \mathbb{R}^n$. *Тогда краевая задача* (3.2) *имеет решение.* ■

Теорема 2. *Пусть краевая задача* (3.2) *имеет решение и верно, что*

$$\int\limits_{\mathbb{R}^n \setminus \Omega} |\nabla \varphi|^2 dx < \infty \,.$$

Тогда существует такая константа $c \in \mathbb{R}^n$, *что* $cap_{\varphi - c}(\mathbb{R}^n \setminus \Omega) < \infty$. ■

Теорема 3. *Пусть* $n \geq 2$. *Тогда* $cap_{\varphi - c}(\mathbb{R}^n \setminus \Omega) < \infty$ *тогда и только тогда, когда*

$$\sum_{k=1}^{\infty} cap_{\varphi - c}((\overline{B}_{r_{k+1}} \setminus B_{r_{k-1}}) \cap (\mathbb{R}^n \setminus \Omega), B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}) < \infty \,,$$

где

$$r_k = \begin{cases} 2^k \,, & \text{если } n \geq 3 \\ 2^{2^k} \,, & \text{если } n = 2 \end{cases}$$

■

Пусть $\omega \subset \mathbb{R}^n$ — ограниченная липшицева область и μ – мера в ω такая, что

$$\sup_{x \in \mathbb{R}^n,\, \rho > 0} \rho^{1-n} \mu(B_\rho^x \cap \omega) < \infty,$$

Тогда для любой функции $v \in W_2^1(\omega)$ существует $c \in \mathbb{R}$ такое, что

$$\sigma(\omega,\mu) \|v - c\|_{L_2(\omega,\mu)} \leq \|\nabla v\|_{L_2(\omega)}, \tag{3.3}$$

где постоянная $\sigma(\omega,\mu) > 0$ не зависит от v [31, §1.4.5].

Теорема 4. *Пусть краевая задача* (3.2) *имеет решение и* μ_k *— семейство мер в* ω_k, $k = 1,2,\ldots$, *попарно не пересекающихся ограниченных липшицевых областей в* $\mathbb{R}^n$, *удовлетворяющие условиям*

$$\sup_{x \in \mathbb{R}^n,\, \rho > 0} \rho^{1-n} \mu_k(B_\rho^x \cap \omega_k) < \infty,$$

$$\sum_{k=1}^{\infty} \int\limits_{\omega_k \setminus \Omega} |\nabla \varphi|^2 dx < \infty. \tag{3.4}$$

Тогда

$$\sum_{k=1}^{\infty} \sigma^2(\omega_k,\mu_k) m_k^2(\varphi) < \infty, \tag{3.5}$$

где $\sigma(\omega_k,\mu_k)$ *— коэффициент в неравенстве* (3.3) *и* $m_k(\varphi) = \inf_{c\in\mathbb{R}} \|\varphi - c\|_{L_2(\omega_k\setminus\Omega,\mu_k)}$. ■

Для доказательства теорем 1 – 4 нам понадобится ряд вспомогательных утверждений.

Неравенство из следующей леммы является достаточно известным (например, [50] с. 288, с. 398) и встречается в различных модификациях. Тем не менее, в целях полноты изложения приведем подробное доказательство неравенства.

Лемма 1 (Частное неравенство Харди)**.** *Пусть* $\psi \in C_0^\infty(\mathbb{R}^n)$ *и* $n \geq 3$. *Тогда*

$$\int\limits_{\mathbb{R}^n} |\nabla\psi|^2 \, dx \geq k \int\limits_{\mathbb{R}^n} \frac{|\psi|^2}{|x|^2} \, dx \,,$$

где константа k не зависит от и.

Доказательство. Перейдем к полярным координатам. Тогда интеграл в правой части неравенства приобретает вид

$$\int dS \int\limits_0^\infty \frac{|\psi|^2}{r^2} \, r^{n-1} \, dr \,,$$

где первый интеграл берется по всем угловым координатам. Фиксируем угловые координаты и получаем цепочку преобразований

$$\int\limits_0^\infty \frac{|\psi|^2}{r^2} \, r^{n-1} \, dr = \int\limits_0^\infty |\psi^2| \, r^{n-3} \, dr = \frac{1}{n-2} \int\limits_0^\infty |\psi|^2 \left(r^{n-2}\right)' dr =$$

$$= \frac{1}{n-2} (r^{n-2} \, |\psi|^2 \Big|_{r=0}^{\infty} - \int\limits_0^\infty 2 \, |\psi| \, |\psi|' \, r^{n-2} \, dr).$$

Первое слагаемое в финальной скобке, очевидно, равняется нулю, т.к. ψ – пробная функция. Оценим модуль второго слагаемого, используя неравенство $ab \leq$

$\varepsilon\, a^2 + \frac{1}{\varepsilon}\, b^2$, считая $a = |\psi|\, r^{\frac{n-3}{2}}, b = r^{\frac{n-1}{2}} |\psi\,'|\,.$

$$\left| \int\limits_0^\infty 2\,|\psi|\,|\psi|'\, r^{n-2}\, dr \right| \le 2 \int\limits_0^\infty |\psi|\,|\psi'|\, r^{n-2}\, dr \le$$

$$\le 2 \left(\varepsilon \int\limits_0^\infty |\psi|^2\, r^{n-3}\, dr + \frac{1}{\varepsilon} \int\limits_0^\infty |\psi'|^2\, r^{n-1}\, dr \right)\,.$$

Таким образом, получаем цепочку неравенств

$$\int\limits_0^\infty |\psi|^2\, r^{n-3}\, dr \le \frac{2}{n-2} \int\limits_0^\infty |\psi|\,|\psi'|\, r^{n-2}\, dr \le$$

$$\le \frac{2\,\varepsilon}{n-2} \int\limits_0^\infty |\psi|^2\, r^{n-3}\, dr + \frac{2}{\varepsilon(n-2)} \int\limits_0^\infty |\psi'|^2\, r^{n-1}\, dr\,.$$

Следовательно, переносом первого слагаемого в левую часть получаем

$$(1 - \frac{2\,\varepsilon}{n-2}) \int\limits_0^\infty |\psi|^2\, r^{n-3}\, dr \le \frac{2}{\varepsilon(n-2)} \int\limits_0^\infty |\psi'|^2\, r^{n-1}\, dr\,.$$

С учетом того, что $|\psi'|^2 \le |\nabla\psi|^2$, а r^{n-1} символизирует собой якобиан перехода к полярным координатам, после возвращения к исходным координатам получаем

$$\int\limits_{\mathbb{R}^n} |\nabla\psi|^2\, dx \ge k \int\limits_{\mathbb{R}^n} \frac{|\psi|^2}{|x|^2}\, dx,.$$

□

Замечание. *Взяв последовательность* $\{\psi_k\} \in C_0^\infty(\mathbb{R}^n)$ *фундаментальную в* L_2^1, *то есть в полунорме*

$$\|\cdot\|_{L_2^1(\mathbb{R}^n)} = \left(\int\limits_{\mathbb{R}^n} |\nabla\psi|^2\, dx \right)^{\frac{1}{2}}\,,$$

мы очевидным образом из частного неравенства Харди получим фундаментальность этой последовательности в метрике

$$\|\cdot\| = \left(\int\limits_{\mathbb{R}^n} |\nabla\psi|^2 \, dx + \int\limits_{\mathbb{R}^n} \frac{|\psi|^2}{|x|^2} \, dx \right)^{\frac{1}{2}}.$$

Следовательно, частное неравенство Харди справедливо и для $\psi \in \overset{\circ}{L}{}_2^1(\mathbb{R}^n)$.

На основе полученного частного неравенства Харди будет получена некоторая модификация общего неравенства Харди. Такая модификация неравенства Харди, за счет конструктивной модификации правой части неравенства, позволит нам расширить класс функций, для которых оно оказывается справедливым.

Лемма 2 (Общее неравенство Харди)**.** *Пусть* $u \in L_2^1(\mathbb{R}^n)$ *и* $n \geq 3$. *Тогда существует такая константа* c, *что справедливо неравенство*

$$\int\limits_{\mathbb{R}^n} |\nabla u|^2 \, dx \geq k \int\limits_{\mathbb{R}^n} \frac{|u-c|^2}{|x|^2} \, dx \,,$$

где константа k *не зависит от* u.

Доказательство. То, что u принадлежит пространству $L_2^1(\mathbb{R}^n)$, эквивалентно тому, что

$$\int\limits_{\mathbb{R}^n} |\nabla u|^2 \, dx < \infty \,.$$

Разложим пространство $L_2^1(\mathbb{R}^n)$ в прямое произведение $\overset{\circ}{L}{}_2^1(\mathbb{R}^n)$ и его ортогонального дополнения. Пусть u_0 – проекция u на $\overset{\circ}{L}{}_2^1(\mathbb{R}^n)$, а h – составляющая из ортогонального дополнения. Ввиду гильбертовости и сепарабельности $\overset{\circ}{L}{}_2^1(\mathbb{R}^n)$ получаем, что для любого $v \in \overset{\circ}{L}{}_2^1(\mathbb{R}^n)$ будем иметь

$$\int\limits_{\mathbb{R}^n} \nabla v \nabla h \, dx = 0 \,.$$

Значит, $\triangle h = 0$ в $\mathbb{R}^n$. Из равенства Парсеваля получаем, что

$$\int\limits_{\mathbb{R}^n} |\nabla h|^2 \, dx + \int\limits_{\mathbb{R}^n} |\nabla u_0|^2 \, dx = \int\limits_{\mathbb{R}^n} |\nabla u|^2 \, dx$$

Ввиду ограниченности правой части, получаем ограниченность каждого слагаемого в левой части. В частности, получаем, что

$$\int\limits_{\mathbb{R}^n} |\nabla h|^2 \, dx < \infty \, .$$

Вспоминая об эллиптичности h, получаем, что h константа. Тогда, используя частное неравенство Харди применительно к $u_0 = u - h = u - c$, получаем общее неравенство Харди. □

Случай $n = 2$ требует отдельного доказательства.

Лемма 3. *В случае $n = 2$ существует константа k, не зависящая от u, что*

$$\int\limits_{\mathbb{R}^2} |\nabla u|^2 \, dx \geq k \int\limits_{|x| \geq 2\delta} \frac{|u|^2}{|x|^2 \ln^2 \frac{|x|}{\delta}} \, dx \, ,$$

для любой функции $u \in L_2^1(\mathbb{R}^2)$ и любой константы $\delta > 0$. Более того, для любой функции $u \in L_2^1(\mathbb{R}^2)$ такой, что $u = 0$ почти всюду в окрестности нуля, полученное неравенство эквивалентно неравенству

$$\int\limits_{\mathbb{R}^2} |\nabla u|^2 \, dx \geq k \int\limits_{\mathbb{R}^2} \frac{|u|^2}{|x|^2 \ln^2 |x|} \, dx \, .$$

Доказательство. Сначала проведем доказательство для $u \in C^\infty(\mathbb{R}^2)$.

Перейдем к полярным координатам. Тогда интеграл в правой части неравенства приобретает вид

$$\int\limits_0^{2\pi} d\phi \int\limits_0^\infty \frac{r|u|^2}{r^2 \ln^2 r} \, dr \, .$$

Фиксируем угловую координату и получаем цепочку преобразований

$$\int\limits_0^\infty \frac{r|u|^2}{r^2 \ln^2 r} \, dr = \int\limits_0^\infty \frac{|u|^2}{r \ln^2 r} \, dr = -\int\limits_0^\infty \left(\frac{1}{\ln r} \right)' |u|^2 \, dr =$$

$$= -\frac{1}{\ln r} |u|^2 \Big|_{r=0}^\infty + \int\limits_0^\infty \frac{1}{\ln r} \, 2 \, |u| |u|' dr \, .$$

Первое слагаемое в финальной скобке, очевидно, равняется нулю, т.к. u равна нулю в окрестности нуля. Оценим модуль второго слагаемого, используя неравенство $ab \le \varepsilon\, a^2 + \frac{1}{\varepsilon}\, b^2$, считая $a = \frac{|u|}{r^{\frac{1}{2}} \ln r}, b = r^{\frac{1}{2}}|u'|$.

$$\left| \int\limits_0^\infty \frac{1}{\ln r} 2|u||u'|dr \right| = \left| \int\limits_0^\infty \frac{2r^{\frac{1}{2}}|u||u|'}{r^{\frac{1}{2}} \ln r}\, dr \right| \le 2 \int\limits_0^\infty \frac{r^{\frac{1}{2}}|u||u'|}{r^{\frac{1}{2}} \ln r} dr \le$$
$$\le 2 \left(\varepsilon \int\limits_0^\infty \frac{|u|^2}{r \ln^2 r}\, dr + \frac{1}{\varepsilon} \int\limits_0^\infty r|u'|^2\, dr \right).$$

Таким образом, получаем цепочку неравенств

$$\int\limits_0^\infty \frac{|u|^2}{r \ln^2 r}\, dr \le 2\,\varepsilon \int\limits_0^\infty \frac{|u|^2}{r \ln^2 r}\, dr + \frac{2}{\varepsilon} \int\limits_0^\infty r|u'|^2\, dr\,.$$

Следовательно, переносом первого слагаемого в левую часть получаем

$$(1 - 2\,\varepsilon) \int\limits_0^\infty \frac{|u|^2}{r \ln^2 r}\, dr \le \frac{2}{\varepsilon} \int\limits_0^\infty r|u'|^2\, dr\,.$$

С учетом того, что $|u'|^2 \le |\nabla u|^2$, а r символизирует собой якобиан перехода к полярным координатам, после возвращения к исходным координатам получаем

$$\int\limits_{\mathbb{R}^2} |\nabla u|^2\, dx \ge k \int\limits_{\mathbb{R}^2} \frac{|u|^2}{|x|^2 \ln^2 |x|}\, dx\,.$$

Далее, пользуясь предельным переходом и тем фактом, что C^∞ плотно в L_2^1 ([31] с. 18), получаем доказательство леммы. □

Не сложно получить утверждение о поведении функции выделенного класса на $(2,1)$-полярном множестве.

Лемма 4. *Пусть E – $(2,1)$-полярное множество. Тогда $u|_E = 0$ для любой $u \in W^1_{2,\,loc}(\mathbb{R}^n)$, то есть $\mu u \in \overset{\circ}{W}{}^1_2(\mathbb{R}^n \setminus E)$ для любой $\mu \in C_0^\infty(\mathbb{R}^n)$.*

Доказательство. Известно ([31] с. 331, теорема 1), что пространство $\mathcal{D}(\Omega)$ плотно в W_2^1 в том и только в том случае, если $\mathbb{R}^n \setminus \Omega$ – $(2,1)$-полярное множество. Откуда и следует утверждение леммы. □

Сформулированные утверждения позволяют нам получить аналог неравенства Фридрихса, дающего оценку норму функции через норму градиента.

Лемма 5. *Пусть* $Cap((\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{r_0}, W_2^1(\mathbb{R}^n)) > 0$ *для некоторого* r_0. *Тогда существует константа* A, *не зависящая от* φ, *что*

$$\|\varphi\|_{L_2(B_r)} \leq A \|\nabla\varphi\|_{L_2(B_r)}$$

при любых $r > 2\,r_0$ *и любых* $\varphi \in W^1_{2,\,loc}(\mathbb{R}^n)$, *удовлетворяющих условию*

$$\varphi\,|_{(\mathbb{R}^n\setminus\Omega)\cap\overline{B}_{r_0}} = 0\,.$$

Доказательство. Предположим противное. Тогда для любой константы A существует $r > 2\,r_0$ и функция $\varphi \in W^1_{2,\,loc}(\mathbb{R}^n)$ такая, что

$$\varphi\,|_{(\mathbb{R}^n\setminus\Omega)\cap\overline{B}_{r_0}} = 0\,,$$

и при этом

$$\|\varphi\|_{L_2(B_r)} > A \|\nabla\varphi\|_{L_2(B_r)}\,.$$

Возьмем последовательность $A_s = s$, $s = 1,2,\ldots$ Найдется последовательность φ_s такая, что $\|\varphi_s\|_{L_2(B_r)} > s\,\|\nabla\varphi_s\|_{L_2(B_r)}$. Положим

$$\psi_s = \frac{\varphi_s}{\|\varphi_s\|_{L_2(B_r)}}\,.$$

Очевидно, что $\|\psi_s\|_{L_2(B_r)} = 1$, и при этом

$$\|\nabla\psi_s\|_{L_2(B_r)} \to 0 \ \text{ при } s \to \infty\,.$$

Следовательно, $\|k - \psi_s\|_{W_2^1(B_r)}$ стремится к нулю при $s \to \infty$ и при некоторой константе k, а значит, взяв функцию $(k - \psi_s)\eta$, где $\eta \in C_0^\infty(B_{2r_0})$, $\eta \equiv 1$ в окрестности B_{r_0}, будем иметь

$$\text{Cap}((\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{r_0}, W_2^1(\mathbb{R}^n)) \leq \int\limits_{\mathbb{R}^n} |\nabla((k-\psi_s)\eta)|^2 dx \leq \text{const}\,\|k - \psi_s\|_{W_2^1(B_r)}.$$

Перейдя к пределу при $s \to \infty$, будем иметь $\text{Cap}((\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{r_0}, W_2^1(\mathbb{R}^n)) = 0$. Полученное противоречие доказывает лемму. □

Доказательство Теоремы 1. $\{r_i\}_{i=1}^n$ и $\{\rho_i\}_{i=1}^n$ – бесконечно возрастающие последовательности вещественных чисел. Пусть $r_i < \rho_i$ для всех i, и

$$\operatorname{cap}_{\varphi - c}((\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{r_i}, B_{\rho_i}) < \operatorname{cap}_{\varphi - c}(\mathbb{R}^n \setminus \Omega) + \frac{1}{2^i},\ i = 1,2,\ldots$$

Очевидно [35], что емкость $\operatorname{cap}_{\varphi - c}((\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{r_i}, B_{\rho_i})$ достигается на функции $v_i \in \overset{\circ}{W}{}_2^1(B_{\rho_i})$ такой, что

$$\begin{cases} \operatorname{tg} v_i = 0 \ \text{ в } B_{\rho_i} \setminus ((\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{r_i}) \\ v_i|_{(\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{r_i}} = \varphi - c\,, \end{cases} \tag{3.6}$$

где последнее равенство означает, что $(v_i - (\varphi - c))\mu \in \overset{\circ}{W}{}_2^1(B_{\rho_i} \setminus ((\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{r_i}))$ для любого $\mu \in C_0^\infty(B_{\rho_i})$. Наряду с задачей (3.6) рассмотрим другую задачу:

$$\begin{cases} Lu_i = 0 \ \text{ в } B_{\rho_i} \setminus ((\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{r_i}) \\ u_i|_{(\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{r_i}} = \varphi - c\,, \end{cases} \tag{3.7}$$

где $u_i \in \overset{\circ}{W}{}_2^1(B_{\rho_i})$.

Имеет место следующее утверждение: пусть функция u_i – решение задачи (3.7), а функция v_i – решение задачи (3.6). Тогда

$$\int\limits_{B_{\rho_i}} |\nabla v_i|^2 dx \le \int\limits_{B_{\rho_i}} |\nabla u_i|^2 dx \le c \int\limits_{B_{\rho_i}} |\nabla v_1|^2 dx\,, \tag{3.8}$$

где c – неотрицательная константа, не зависящая от u_i и v_i. Проведем доказательство этого факта. Левое неравенство, очевидно, следует из определения емкости. Докажем правое неравенство. Ввиду того, что функция u_i – решение задачи (3.7), будем иметь

$$\int\limits_{B_{\rho_i}} \sum_{l,m=1}^{n} a_{lm}(x) \frac{\partial u_i}{\partial x_m} \frac{\partial \psi}{\partial x_l}\, dx = 0$$

для любой функции $\psi \in \overset{\circ}{W}{}_2^1(\Omega)$. В частности, взяв $\psi = u - v$, получим

$$\int\limits_{B_{\rho_i}} \sum_{l,m=1}^{n} a_{lm}(x) \frac{\partial u_i}{\partial x_m} \frac{\partial u_i}{\partial x_l}\, dx - \int\limits_{B_{\rho_i}} \sum_{l,m=1}^{n} a_{lm}(x) \frac{\partial u_i}{\partial x_m} \frac{\partial v_i}{\partial x_l}\, dx \leq = 0$$

Откуда следуют оценки

$$\gamma \int\limits_{B_{\rho_i}} |\nabla u_i|^2 dx \leq \int\limits_{B_{\rho_i}} \sum_{l,m=1}^{n} a_{lm}(x) \frac{\partial u_i}{\partial x_m} \frac{\partial u_i}{\partial x_l}\, dx = \int\limits_{B_{\rho_i}} \sum_{l,m=1}^{n} a_{lm}(x) \frac{\partial u_i}{\partial x_m} \frac{\partial v_i}{\partial x_l}\, dx \leq$$

$$const(\int\limits_{B_{\rho_i}} |\nabla u_i|^2 dx)^{\frac{1}{2}} (\int\limits_{\Omega} |\nabla v_i|^2 dx)^{\frac{1}{2}}\,,$$

что доказывает правое неравенство в (3.8).

Очевидно,

$$\text{cap}_{\,\varphi - c}((\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{r_i}) \leq \int\limits_{B_{\rho_i}} |\nabla v_i|^2 dx < \text{cap}_{\,\varphi - c}(\mathbb{R}^n \setminus \Omega) + \frac{1}{2^i}\,.$$

В то же время, из неравенства (3.8) следует, что

$$\int\limits_{B_{\rho_i}} |\nabla u_i|^2 dx < c_2(\text{cap}_{\,\varphi - c}(\mathbb{R}^n \setminus \Omega) + \frac{1}{2^i})\,.$$

Если $\text{Cap}(\mathbb{R}^n \setminus \Omega, W_2^1(\mathbb{R}^n)) = 0$, то множество $\mathbb{R}^n \setminus \Omega$ является $(2,1)$-полярным ([31] с. 331), а значит, согласно лемме 4, функция $1 - \varphi$ равна нулю на $\mathbb{R}^n \setminus \Omega$. Тем самым, взяв единичную функцию, мы получим искомое решение задачи (3.2).

Пусть теперь $\text{Cap}((\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{r_i}, W_2^1(\mathbb{R}^n)) > 0$ для некоторого r_i. Тогда из леммы 5 получим, что последовательность $\{u_i\}_{i=1}^n$ ограничена как в $L_2(B_r)$ так и в $W_2^1(B_r)$ для любого r. В самом деле, для достаточно больших i,j имеем

$$u_i - u_j \in \overset{\circ}{W}{}_{2,\,loc}^1(\mathbb{R}^n \setminus ((\mathbb{R}^n \setminus \Omega) \cap \overline{B}_r))\,, \quad i,j > i_0\,.$$

Таким образом, фиксируя j, будем иметь

$$\|u_i\|_{L_2^1}^2 + \|u_i\|_{W_2^1(K)}^2 \leq \alpha(\|u_j\|_{W_2^1(K)}^2 + cap_{\,\varphi - c}(\mathbb{R}^n \setminus \Omega) + \frac{1}{2^i} + \frac{1}{2^j})\,,$$

для всякого компакта $K \subset \mathbb{R}^n$, где константа $\alpha > 0$ не зависит от u_i.

В силу компактности вложения $W_2^1(B_r)$ в $L_2(B_r)$ мы можем выбрать подпоследовательность последовательности $\{u_i\}_{i=1}^n$ фундаментальную в $L_2(B_r)$. Чтобы не загромождать индексов, обозначим эту подпоследовательность также $\{u_i\}_{i=1}^n$. Возьмем функцию $\eta \in C_0^\infty(B_r)$ такую, что $\eta \equiv 1$ в открытой окрестности множества $\overline{B}_{r/2}$. Ввиду того, что u_i удовлетворяет (3.7), то для разности $u_i - u_j$ получим

$$\int\limits_{B_r} \sum_{l,m=1}^{n} a_{lm}(x) \frac{\partial (u_i - u_j)}{\partial x_m} \frac{\partial \psi}{\partial x_l}\, dx = 0\,,$$

где $\psi = \eta^2 (u_i - u_j)$. Другими словами,

$$\int\limits_{B_r} \sum_{l,m=1}^{n} a_{lm}(x) \frac{\partial (u_i - u_j)}{\partial x_m} \frac{\partial \eta^2}{\partial x_l} (u_i - u_j)\, dx +$$

$$+ \int\limits_{B_r} \sum_{l,m=1}^{n} a_{lm}(x) \eta^2 \frac{\partial (u_i - u_j)}{\partial x_m} \frac{\partial (u_i - u_j)}{\partial x_l}\, dx = 0\,.$$

Перепишем последнее соотношение в виде

$$\gamma \int\limits_{B_r} \eta^2 |\nabla (u_i - u_j)|^2 dx \le -2 \int\limits_{B_r} \sum_{k,l=1}^{n} a_{kl}(x) \frac{\partial (u_i - u_j)}{\partial x_l} \frac{\partial \eta}{\partial x_k}\, \eta\, (u_i - u_j)\, dx\,,$$

откуда ввиду неравенства $ab \le \frac{1}{2}\, a^2 + \frac{1}{2}\, b^2$, находим

$$\int\limits_{B_r} \eta^2 |\nabla (u_i - u_j)|^2 dx \le c_1 \int\limits_{B_r} |\nabla (u_i - u_j)|^2 \eta^2 dx + c_2 \int\limits_{B_r} |\nabla \eta|^2 (u_i - u_j)^2\, dx\,,$$

где c_1,c_2 – неотрицательные константы, не зависящие от u_i. Таким образом, будем иметь

$$\int\limits_{B_{r/2}} |\nabla (u_i - u_j)|^2 dx \le \beta \int\limits_{B_r} (u_i - u_j)^2\, dx\,,$$

где константа $\beta > 0$ не зависит от u_i, что доказывает фундаментальность последовательности $\{u_i\}_{i=1}^n$ в $W_2^1(B_{r/2})$ для любого $r > 0$. Тем самым, существует функция $u \in W_{2,\,loc}^1(\mathbb{R}^n)$ такая, что для любого $r > 0$ последовательность $\{u_i\}_{i=1}^n$

стремится к u в $W_2^1(B_r)$. Очевидно, что функция u является искомым решением задачи (3.2). $\square$

Доказательство Теоремы 2. Предположим, что функция u – решение задачи (3.2). Продолжим u на $\mathbb{R}^n \setminus \Omega$ значением φ. Пусть $n \geq 3$, тогда существует константа $c \in \mathbb{R}^n$ такая, что для функции u имеет место общее неравенство Харди. Обозначая

$$\nu_R = \eta\left(\frac{|x|}{R}\right)(u-c)\,,$$

где $\eta \in C_0^\infty(B_2)$ и $\eta \equiv 1$ в открытой окрестности $\overline{B}_1$, получим

$$\nu_R\big|_{(\mathbb{R}^n\setminus\Omega)\cap\overline{B}_R} = \varphi - c\,.$$

Интеграл Дирихле для функции ν_R допускает оценку

$$\int\limits_{B_{2R}} \left|\nabla\left(\eta\left(\frac{|x|}{R}\right)(u-c)\right)\right|^2 dx \leq 2\left(\int\limits_{B_{2R}} \left|\nabla\eta\left(\frac{|x|}{R}\right)(u-c)\right|^2 dx + \right.$$
$$\left. + \int\limits_{B_{2R}} \left|\eta\left(\frac{|x|}{R}\right)\nabla(u-c)\right|^2 dx\right).$$

Заметим, что

$$\left|\nabla\eta\left(\frac{|x|}{R}\right)\right| \leq \frac{p}{R} \text{ и } \frac{1}{R^2} \leq \frac{4}{|x|^2} \text{ при } x \in B_{2R}\,,$$

где $p > 0$ – некоторая константа. Тогда с учетом неравенства Харди будем иметь

$$\int\limits_{B_{2R}} \left|\nabla\eta\left(\frac{|x|}{R}\right)(u-c)\right|^2 dx \leq \frac{p^2}{R^2}\int\limits_{B_{2R}\setminus B_R} |u-c|^2\,dx \leq$$
$$\leq 4p^2 \int\limits_{B_{2R}\setminus B_R} \frac{|u-c|^2}{|x|^2}dx \leq \frac{4p^2}{k}\int\limits_{B_{2R}} |\nabla u|^2 dx\,.$$

Таким образом

$$\operatorname{cap}_{\varphi-c}((\mathbb{R}^n\setminus\Omega)\cap\overline{B}_R) \leq \int\limits_{B_{2R}} |\nabla\nu_R|^2 dx \leq \gamma\int\limits_{B_{2R}} |\nabla u|^2 dx < \infty\,,$$

где $\gamma > 0$ – некая константа, не зависящая от ν_R. Переходя к пределу при $R \to \infty$, получим

$$\operatorname{cap}_{\varphi - c}(\mathbb{R}^n \setminus \Omega) \le \gamma \int\limits_{\mathbb{R}^n} |\nabla u|^2 dx < \infty \,,$$

что доказывает теорему 2 для $n \ge 3$.

В случае $n = 2$, обозначая

$$\nu_R = \eta \left(\frac{\ln \frac{|x|}{R}}{\ln R} \right) u \,,$$

где $\eta \in C_0^\infty(\mathbb{R}^2)$, η равна нулю в окрестности нуля и $\eta \equiv 1$ в открытой окрестности дополнения B_1, получим

$$\nu_R\big|_{(\mathbb{R}^n \setminus \Omega) \cap \overline{B}_{R^2}} = \varphi \,.$$

Интеграл Дирихле для функции ν_R оценивается следующим образом

$$\int\limits_{B_{2R^2}} \left| \nabla (\eta \left(\frac{\ln \frac{|x|}{R}}{\ln R} \right) u) \right|^2 dx \le$$

$$\le 2 \left(\int\limits_{B_{2R^2}} \left| \nabla \eta \left(\frac{\ln \frac{|x|}{R}}{\ln R} \right) u \right|^2 dx + \int\limits_{B_{2R^2}} \left| \eta \left(\frac{\ln \frac{|x|}{R}}{\ln R} \right) \nabla u \right|^2 dx \right) .$$

Имеем

$$\left| \nabla \eta \left(\frac{\ln \frac{|x|}{R}}{\ln R} \right) \right| \le \frac{2q}{|x| \ln R^2} \;\text{ и }\; \frac{1}{\ln^2 R^2} \le \frac{m}{\ln^2 |x|} \;\text{ при }\; x \in B_{2R^2} \,,$$

где q,$m > 0$ – некоторые константы. Тогда с учетом неравенства Харди находим

$$\int\limits_{B_{2R^2}} \left| \nabla \eta \left(\frac{\ln \frac{|x|}{R}}{\ln R} \right) u \right|^2 dx \le \int\limits_{B_{2R^2} \setminus B_{R^2}} \frac{4\, q^2}{|x|^2 \ln^2 R^2} |u|^2 \, dx \le$$

$$4\, q^2 m \int\limits_{B_{2R^2} \setminus B_{R^2}} \frac{|u|^2}{|x|^2 \ln^2 |x|} \, dx \le \frac{4 q^2 m}{k} \int\limits_{B_{2R^2}} |\nabla u|^2 dx \,.$$

Таким образом,

$$\operatorname{cap}_{\varphi}((\mathbb{R}^2 \setminus \Omega) \cap \overline{B}_{R^2}) \leq \int\limits_{B_{2R^2}} |\nabla \nu_R|^2 dx \leq \gamma \int\limits_{B_{2R^2}} |\nabla u|^2 dx < \infty\,,$$

где $\gamma > 0$ – некая константа, не зависящая от ν_R. Переходя к пределу при $R \to \infty$, получим

$$\operatorname{cap}_{\varphi}(\mathbb{R}^2 \setminus \Omega) \leq \gamma \int\limits_{\mathbb{R}^2} |\nabla u|^2 dx < \infty\,.$$

Тем самым, теорема 2 полностью доказана. □

Доказательство Теоремы 3. Пусть $\operatorname{cap}_{\varphi - c}(\mathbb{R}^n \setminus \Omega) < \infty$. Тогда по Теореме 1 существует функция u, являющаяся решением задачи (3.2). Пусть $n \geq 3$. Возьмем срезывающую функцию $\eta_k \in C_0^\infty(\mathbb{R}^n)$ такую, что $\eta_k(x) = 1$, на $B_{2^{k+1}} \setminus B_{2^{k-1}}$ и $\operatorname{supp} \eta_k(x) \subset B_{2^{k+2}} \setminus B_{2^{k-2}}$, $k = 1,2,\ldots$, построенную следующим образом. Пусть $\eta(x)$ – монотонная неубывающая функция из $C^\infty(\mathbb{R}^n)$, равная нулю на отрезке $[-\infty, \frac{1}{4}]$ и равная единице на отрезке $[\frac{3}{4}, +\infty]$. Далее, принимаем за $\eta_k(x)$ следующую функцию

$$\eta_k(x) = \begin{cases} \eta\left(\dfrac{|x| - r_{k-2}}{r_{k-1} - r_{k-2}}\right), & \text{если } x \in \overline{B}_{r_{k-1}} \setminus B_{r_{k-2}} \\ 1\,, & \text{если } x \in \overline{B}_{r_{k+1}} \setminus B_{r_{k-1}} \\ \eta\left(\dfrac{r_{k+2} - |x|}{r_{k+2} - r_{k+1}}\right), & \text{если } x \in \overline{B}_{r_{k+2}} \setminus B_{r_{k+1}} \end{cases}$$

Имеем оценку

$$|\nabla \eta_k(x)|^2 \leq \frac{c}{|x|^2}\,,$$

где c не зависит от k. Тогда, согласно неравенству Харди, получим цепочку неравенств

$$\operatorname{cap}_{\varphi- c}((\overline{B}_{r_{k+1}} \setminus B_{r_{k-1}}) \cap (\mathbb{R}^n \setminus \Omega), B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}) \leq$$
$$\leq \int\limits_{B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}} |\nabla (\eta_k(x) u(x))|^2 \, dx \leq$$
$$2 \int\limits_{B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}} |\nabla \eta_k(x) u(x)|^2 \, dx + 2 \int\limits_{B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}} |\eta_k(x) \nabla u(x)|^2 \, dx \leq$$
$$2c \int\limits_{B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}} \frac{|u(x)|^2}{|x|^2} \, dx + b_1 \int\limits_{B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}} |\nabla u(x)|^2 \, dx$$

где b_1 – положительная константа, не зависящая от u. Таким образом,

$$\sum_{k=1}^{\infty} \operatorname{cap}_{\varphi- c}((\overline{B}_{r_{k+1}} \setminus B_{r_{k-1}}) \cap (\mathbb{R}^n \setminus \Omega), B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}) \leq$$
$$\sum_{k=1}^{\infty} 2 \int\limits_{B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}} \frac{|u(x)|^2}{|x|^2} \, dx + \sum_{k=1}^{\infty} b_1 \int\limits_{B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}} |\nabla u(x)|^2 \, dx$$

Ввиду того, что каждая точка $x \in \mathbb{R}^n$ принадлежит не более, чем трем областям $B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}$, то получаем

$$\sum_{k=1}^{\infty} 2 \int\limits_{B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}} \frac{|u(x)|^2}{|x|^2} \, dx + \sum_{k=1}^{\infty} b_1 \int\limits_{B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}} |\nabla u(x)|^2 \, dx \leq b_2 \int\limits_{\mathbb{R}^n} \frac{|u(x)|^2}{|x|^2} \, dx +$$
$$b_3 \int\limits_{\mathbb{R}^n} |\nabla u(x)|^2 \, dx \leq b_4 \int\limits_{\mathbb{R}^n} |\nabla u(x)|^2 \, dx < \infty \,,$$

где b_2,b_3,b_4 – положительные константы, не зависящие от u.

Пусть теперь $n = 2$. Введем обозначение

$$\eta_k(x) = \begin{cases} \eta\left(\dfrac{\ln\frac{|x|}{r_{k-2}}}{\ln\frac{r_{k-1}}{r_{k-2}}}\right), & \text{если } x \in \overline{B}_{r_{k-1}} \setminus B_{r_{k-2}} \\ 1\,, & \text{если } x \in \overline{B}_{r_{k+1}} \setminus B_{r_{k-1}} \\ \eta\left(\dfrac{\ln\frac{r_{k+2}}{|x|}}{\ln\frac{r_{k+2}}{r_{k+1}}}\right), & \text{если } x \in \overline{B}_{r_{k+2}} \setminus B_{r_{k+1}} \end{cases}$$

где $\eta(x)$ – монотонная неубывающая функция из $C^\infty(\mathbb{R}^n)$, равная нулю на отрезке $[-\infty, \frac{1}{4}]$ и равная единице на отрезке $[\frac{3}{4}, +\infty]$. Имеем

$$|\nabla\eta_k(x)|^2 \le \frac{c}{|x|^2 \ln^2|x|}\,,$$

где c не зависит от k. Тогда, согласно неравенству Харди, получим цепочку неравенств

$$\begin{aligned} &\operatorname{cap}_{\varphi-c}((\overline{B}_{r_{k+1}} \setminus B_{r_{k-1}}) \cap (\mathbb{R}^n \setminus \Omega), B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}) \le \\ \le \int\limits_{B_{r_{k+2}}\setminus\overline{B}_{r_{k-2}}} &|\nabla(\eta_k(x)u(x))|^2\,dx \le 2\int\limits_{B_{r_{k+2}}\setminus\overline{B}_{r_{k-2}}} |\nabla\eta_k(x)u(x)|^2\,dx + \\ +2\int\limits_{B_{r_{k+2}}\setminus\overline{B}_{r_{k-2}}} &|\eta_k(x)\nabla u(x)|^2\,dx \le 2c\int\limits_{B_{r_{k+2}}\setminus\overline{B}_{r_{k-2}}} \frac{|u(x)|^2}{|x|^2\ln^2|x|}\,dx + \\ &+b_1\int\limits_{B_{r_{k+2}}\setminus\overline{B}_{r_{k-2}}} |\nabla u(x)|^2\,dx, \end{aligned}$$

где b_1 – положительная константа, не зависящая от u. Таким образом,

$$\begin{aligned} &\sum_{k=1}^{\infty} \operatorname{cap}_{\varphi-c}((\overline{B}_{r_{k+1}} \setminus B_{r_{k-1}}) \cap (\mathbb{R}^n \setminus \Omega), B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}) \le \\ \le \sum_{k=1}^{\infty} 2 &\int\limits_{B_{r_{k+2}}\setminus\overline{B}_{r_{k-2}}} \frac{|u(x)|^2}{|x|^2\ln^2|x|}\,dx + \sum_{k=1}^{\infty} b_1 \int\limits_{B_{r_{k+2}}\setminus\overline{B}_{r_{k-2}}} |\nabla u(x)|^2\,dx. \end{aligned}$$

Ввиду того, что каждая точка $x \in \mathbb{R}^n$ принадлежит не более, чем трем областям $B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}$, то получаем

$$\sum_{k=1}^{\infty} 2 \int\limits_{B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}} \frac{|u(x)|^2}{|x|^2 \ln^2 |x|}\, dx + \sum_{k=1}^{\infty} b_1 \int\limits_{B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}} |\nabla u(x)|^2\, dx \le$$

$$\le b_2 \int\limits_{\mathbb{R}^n} \frac{|u(x)|^2}{|x|^2 \ln^2 |x|}\, dx + b_3 \int\limits_{\mathbb{R}^n} |\nabla u(x)|^2\, dx \le b_4 \int\limits_{\mathbb{R}^n} |\nabla u(x)|^2\, dx < \infty\,,$$

где b_2,b_3,b_4 – положительные константы, не зависящие от u.

В обратную сторону. Пусть

$$\sum_{k=1}^{\infty} \operatorname{cap}_{\varphi - c}((\overline{B}_{r_{k+1}} \setminus B_{r_{k-1}}) \cap (\mathbb{R}^n \setminus \Omega), B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}) < \infty$$

и $n \ge 3$. Введем срезывающую функцию

$$\widetilde{\psi}_k(x) = \begin{cases} \eta\left(\dfrac{|x| - r_{k-1}}{r_k - r_{k-1}}\right), & \text{если } |x| \le r_k \\ \eta\left(\dfrac{r_{k+1} - |x|}{r_{k+1} - r_k}\right), & \text{если } |x| \ge r_k\,. \end{cases}$$

Обозначим

$$\psi_k(x) = \frac{\widetilde{\psi}_k(x)}{\sum\limits_{i=0}^{\infty} \widetilde{\psi}_i(x)}\,.$$

Очевидно,

$$\sum_{k=1}^{\infty} \psi_k(x) = 1\,.$$

Из условия на емкость мы имеем функции $u_k(x)$, реализующие емкость и которые равны $\varphi - c$ в $B_{r_{k+1}} \setminus B_{r_{k-1}}$ и с носителем из $B_{r_{k+2}} \setminus B_{r_{k-2}}$. Заметим, что

$$\sum_{k=N_1}^{N_2} u_k(x)\psi_k(x) = \varphi - c$$

при x из окрестности множества $(\overline{B}_{r_{N_2-1}} \setminus B_{r_{N_1+1}}) \cap (\mathbb{R}^n \setminus \Omega)$. Имеем

$$\left|\nabla \sum_{k=N_1}^{N_2} u_k(x)\psi_k(x)\right|^2 = \left|\sum_{k=N_1}^{N_2} \nabla u_k(x)\psi_k(x) + \sum_{k=N_1}^{N_2} u_k(x)\nabla\psi_k(x)\right|^2 \leq$$
$$\leq 2\left|\sum_{k=N_1}^{N_2} \nabla u_k(x)\psi_k(x)\right|^2 + 2\left|\sum_{k=N_1}^{N_2} u_k(x)\nabla\psi_k(x)\right|^2 .$$

Так как для любого $x \in \mathbb{R}^n$ существует не более трех натуральных чисел $k \in \{N_1, \ldots, N_2\}$, таких что $\psi_k(x) \neq 0$, получим

$$\left|\sum_{k=N_1}^{N_2} \nabla u_k(x)\psi_k(x)\right|^2 \leq 9 \sum_{k=N_1}^{N_2} |\nabla u_k(x)|^2 |\psi_k(x)|^2 .$$

Аналогично

$$\left|\sum_{k=N_1}^{N_2} u_k(x)\nabla\psi_k(x)\right|^2 \leq 9 \sum_{k=N_1}^{N_2} |u_k(x)|^2 |\nabla\psi_k(x)|^2 .$$

В итоге, получаем

$$\left|\nabla \sum_{k=N_1}^{N_2} u_k(x)\psi_k(x)\right|^2 \leq 18 \sum_{k=N_1}^{N_2} |\nabla u_k(x)|^2|\psi_k(x)|^2 + 18 \sum_{k=N_1}^{N_2} |\nabla u_k(x)|^2|\psi_k(x)|^2 .$$

Следовательно,

$$\int\limits_{\mathbb{R}^n} \left|\nabla\left(\sum_{k=N_1}^{N_2} u_k(x)\psi_k(x)\right)\right|^2 dx \leq$$
$$\leq 18\left(\sum_{k=N_1}^{N_2} \int\limits_{B_{r_{k+2}}\setminus B_{r_{k-2}}} |\nabla u_k(x)\psi_k(x)|^2 dx + \sum_{k=N_1}^{N_2} \int\limits_{B_{r_{k+2}}\setminus B_{r_{k-2}}} |u_k(x)\nabla\psi_k(x)|^2 dx\right) .$$

Первое слагаемое в последнем выражении допускает оценку

$$\sum_{k=N_1}^{N_2} \int\limits_{B_{r_{k+2}}\setminus B_{r_{k-2}}} |\nabla u_k(x)\psi_k(x)|^2 dx \leq \sum_{k=N_1}^{N_2} \int\limits_{B_{r_{k+2}}\setminus B_{r_{k-2}}} |\nabla u_k(x)|^2 dx ,$$

так как $|\psi_k(x)| \leq 1$. Используя тот факт, что

$$|\nabla\psi_k(x)| \leq \frac{r_{k+1} - r_k}{2^k}$$

и неравенство Фридрихса, оценим второе слагаемое

$$\sum_{k=N_1}^{N_2} \int\limits_{B_{r_{k+2}} \setminus B_{r_{k-2}}} |u_k(x)\nabla\psi_k(x)|^2 dx \leq \sum_{k=N_1}^{N_2} \frac{(r_{k+1} - r_k)^2}{4^k} \int\limits_{B_{r_{k+2}} \setminus B_{r_{k-2}}} |u_k(x)|^2 dx \leq$$

$$\leq c_1 \sum_{k=N_1}^{N_2} \int\limits_{B_{r_{k+2}} \setminus B_{r_{k-2}}} |\nabla u_k(x)|^2 dx\,,$$

где c_1 – положительная константа, не зависящая от u_k и ψ_k. Получаем цепочку неравенств

$$\operatorname{cap}_{\varphi-c}((\overline{B}_{r_{N_2}} \setminus B_{r_{N_1}}) \cap (\mathbb{R}^n \setminus \Omega)) \leq \int\limits_{\mathbb{R}^n} \left| \nabla \left(\sum_{k=N_1}^{N_2} u_k(x)\psi_k(x) \right) \right|^2 dx \leq$$

$$\leq c_2 \sum_{k=N_1}^{N_2} \int\limits_{B_{r_{k+2}} \setminus B_{r_{k-2}}} |\nabla u_k(x)|^2 dx =$$

$$= \sum_{k=N_1}^{N_2} \operatorname{cap}_{\varphi-c}((\overline{B}_{r_{k+1}} \setminus B_{r_{k-1}}) \cap (\mathbb{R}^n \setminus \Omega), B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}})\,,$$

где c_2 – положительная константа, не зависящая от u_k и ψ_k. Устремляя N_2 к бесконечности, получаем

$$\operatorname{cap}_{\varphi-c}((\mathbb{R}^n \setminus \Omega) \setminus B_{r_{N_1}}) \leq$$

$$\leq \sum_{k=N_1}^{\infty} \operatorname{cap}_{\varphi-c}((\overline{B}_{r_{k+1}} \setminus B_{r_{k-1}}) \cap (\mathbb{R}^n \setminus \Omega), B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}) < \infty\,,$$

откуда и следует, что

$$\operatorname{cap}_{\varphi-c}(\mathbb{R}^n \setminus \Omega) < \infty\,.$$

Пусть теперь $n = 2$. Тогда введем срезывающую функцию

$$\widetilde{\psi}_k(x) = \begin{cases} \eta\left(\dfrac{\ln\frac{|x|}{r_{k-1}}}{\ln\frac{r_k}{r_{k-1}}}\right), & \text{если } |x| \le r_k \\ \eta\left(\dfrac{\ln\frac{r_{k+1}}{|x|}}{\ln\frac{r_{k+1}}{r_k}}\right), & \text{если } |x| \ge r_k \end{cases}$$

и пусть

$$\psi_k(x) = \frac{\widetilde{\psi}_k(x)}{\sum\limits_{i=0}^{\infty}\widetilde{\psi}_i(x)}\,.$$

Очевидно,

$$\sum_{k=1}^{\infty}\psi_k(x) = 1\,.$$

Из условия на емкость мы имеем функции $u_k(x)$, реализующие емкость и которые равны $\varphi - c$ в $B_{r_{k+1}} \setminus B_{r_{k-1}}$ и с носителем из $B_{r_{k+2}} \setminus B_{r_{k-2}}$. Заметим, что

$$\sum_{k=N_1}^{N_2} u_k(x)\psi_k(x) = \varphi - c$$

при x из окрестности множества $(\overline{B}_{r_{N_2-1}} \setminus B_{r_{N_1+1}}) \cap (\mathbb{R}^n \setminus \Omega)$. Несложно увидеть, что функции ψ_k будут снова удовлетворять соотношениям

$$\int\limits_{\mathbb{R}^n}\left|\nabla\left(\sum_{k=N_1}^{N_2}u_k(x)\psi_k(x)\right)\right|^2 dx \le$$

$$\le 18\left(\sum_{k=N_1}^{N_2}\int\limits_{B_{r_{k+2}}\setminus B_{r_{k-2}}}|\nabla u_k(x)\psi_k(x)|^2dx + \sum_{k=N_1}^{N_2}\int\limits_{B_{r_{k+2}}\setminus B_{r_{k-2}}}|u_k(x)\nabla\psi_k(x)|^2dx\right).$$

Первое слагаемое из последнего выражения допускает очевидную оценку

$$\sum_{k=N_1}^{N_2}\int\limits_{B_{r_{k+2}}\setminus B_{r_{k-2}}}|\nabla u_k(x)\psi_k(x)|^2dx \le \sum_{k=N_1}^{N_2}\int\limits_{B_{r_{k+2}}\setminus B_{r_{k-2}}}|\nabla u_k(x)|^2dx\,,$$

так как $|\psi_k(x)| \le 1$. Теперь, используя тот факт, что $|\nabla\psi_k(x)| \le \dfrac{\text{const}}{|x|\ln|x|}$ и неравенство Харди, оценим второе слагаемое

$$\sum_{k=N_1}^{N_2} \int\limits_{B_{r_{k+2}}\setminus B_{r_{k-2}}} |u_k(x)\nabla\psi_k(x)|^2 dx \le \sum_{k=N_1}^{N_2} \int\limits_{B_{r_{k+2}}\setminus B_{r_{k-2}}} \frac{|u_k(x)|^2}{|x|^2\ln^2|x|} dx$$
$$\le c_1 \sum_{k=N_1}^{N_2} \int\limits_{B_{r_{k+2}}\setminus B_{r_{k-2}}} |\nabla u_k(x)|^2 dx\,,$$

где c_1 – положительная константа, не зависящая от u_k и ψ_k. Получаем цепочку неравенств

$$\operatorname{cap}_{\varphi-c}((\overline{B}_{r_{N_2}} \setminus B_{r_{N_1}}) \cap (\mathbb{R}^n \setminus \Omega)) \le \int\limits_{\mathbb{R}^n} \left|\nabla\left(\sum_{k=N_1}^{N_2} u_k(x)\psi_k(x)\right)\right|^2 dx \le$$
$$\le c_2 \sum_{k=N_1}^{N_2} \int\limits_{B_{r_{k+2}}\setminus B_{r_{k-2}}} |\nabla u_k(x)|^2 dx =$$
$$= \sum_{k=N_1}^{N_2} \operatorname{cap}_{\varphi-c}((\overline{B}_{r_{k+1}} \setminus B_{r_{k-1}}) \cap (\mathbb{R}^n \setminus \Omega), B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}})\,,$$

где c_2 – положительная константа, не зависящая от u_k и ψ_k. Устремляя N_2 к бесконечности, получаем

$$\operatorname{cap}_{\varphi-c}((\mathbb{R}^n \setminus \Omega) \setminus B_{r_{N_1}}) \le$$
$$\le \sum_{k=N_1}^{\infty} \operatorname{cap}_{\varphi-c}((\overline{B}_{r_{k+1}} \setminus B_{r_{k-1}}) \cap (\mathbb{R}^n \setminus \Omega), B_{r_{k+2}} \setminus \overline{B}_{r_{k-2}}) < \infty\,,$$

откуда и следует, что

$$\operatorname{cap}_{\varphi-c}(\mathbb{R}^n \setminus \Omega) < \infty\,.$$

□

Доказательство Теоремы 4. Пусть u – решение задачи (3.2). Продолжим u на $\mathbb{R}^n \setminus \Omega$ значением φ. Тогда согласно неравенству (3.3) будем иметь

$$\sigma(\omega_k,\mu_k)\|u - c_k\|_{L_2(\omega_k,\mu_k)} \le \|\nabla u\|_{L_2(\omega_k)}.$$

Откуда получим

$$\sigma^2(\omega_k,\mu_k)\|\varphi - c_k\|^2_{L_2(\omega_k\setminus\Omega,\mu_k)} = \sigma^2(\omega_k,\mu_k)\|u - c_k\|^2_{L_2(\omega_k\setminus\Omega,\mu_k)} \le \|\nabla u\|^2_{L_2(\omega_k)}.$$

Суммируя последнее соотношение, получим

$$\sum_{k=1}^{\infty}\sigma^2(\omega_k,\mu_k)\|\varphi - c_k\|^2_{L_2(\omega_k\setminus\Omega,\mu_k)} \le \sum_{k=1}^{\infty}\|\nabla u\|^2_{L_2(\omega_k)} =$$
$$= \sum_{k=1}^{\infty}\int\limits_{\omega_k\cap\Omega}|\nabla u|^2dx + \sum_{k=1}^{\infty}\int\limits_{\omega_k\setminus\Omega}|\nabla u|^2dx.$$

Заметим, что

$$\sum_{k=1}^{\infty}\int\limits_{\omega_k\cap\Omega}|\nabla u|^2dx \le \int\limits_{\Omega}|\nabla u|^2dx < \infty$$

и

$$\sum_{k=1}^{\infty}\int\limits_{\omega_k\setminus\Omega}|\nabla u|^2dx = \sum_{k=1}^{\infty}\int\limits_{\omega_k\setminus\Omega}|\nabla\varphi|^2dx < \infty.$$

Таким образом, будем иметь

$$\sum_{k=1}^{\infty}\sigma^2(\omega_k,\mu_k)\|\varphi - c_k\|^2_{L_2(\omega_k\setminus\Omega,\mu_k)} < \infty,$$

откуда немедленно следует (3.5). Теорема полностью доказана. □

3.2.4 Уточненные оценки емкости

В случае верхней полуплоскости и задания степенной граничной функции на действительной оси удается конкретизировать условия теоремы существования в терминах показателя роста граничной функции.

Следствие 1. *Пусть* $\Omega = \{(x', x_n) \in \mathbb{R}^n | x_n \ge 0\}$ *и* $\varphi(x) = (1 + |x|)^\alpha$. *Тогда краевая задача* (3.2) *имеет решение тогда и только тогда, когда* $\alpha < -\dfrac{n-2}{2}$ *или* $\alpha = 0$ *в случае* $n \ge 3$ *и* $\alpha \le 0$ *в случае* $n = 2$.

Доказательство следствия 1. Пусть $\alpha = 0$. Тогда, взяв в теореме 1 $c = 1$, получим, что задача (3.2) имеет решение.

Пусть теперь $\alpha \neq 0$ и $n \geq 3$. Согласно лемме 2,

$$\int\limits_{\mathbb{R}^n} |\nabla\varphi|^2\, dx \geq k \int\limits_{\mathbb{R}^n} \frac{|\varphi - c|^2}{|x|^2}\, dx\,. \tag{3.9}$$

Возьмем срезывающую функцию $\eta \in C_0^\infty(\mathbb{R}^n)$ такую, что $\eta(x) = 1$, на B_1 и $\operatorname{supp}\eta(x) \subset B_2$ и пусть

$$\varphi_N(x) = \eta\left(\frac{|x|}{N}\right)(\varphi(x) - c)\,.$$

Тогда получаем цепочку неравенств

$$\operatorname{cap}_{\varphi - c}(B_N \cap (\mathbb{R}^n \setminus \Omega)) \leq \int\limits_{B_{2N}} |\nabla\varphi_N|^2\, dx \leq \text{const} \int\limits_{B_{2N}} |\nabla\varphi|^2\, dx\,. \tag{3.10}$$

Действительно,

$$\int\limits_{B_{2N}} |\nabla\varphi_N|^2\, dx \leq 2\left(\int\limits_{B_{2N}} \left|\nabla\eta\left(\frac{|x|}{N}\right)(\varphi - c)\right|^2 dx + \int\limits_{B_{2N}} \left|\eta\left(\frac{|x|}{N}\right)\nabla(\varphi - c)\right|^2 dx\right).$$

Заметим, что

$$\left|\nabla\eta\left(\frac{|x|}{N}\right)\right| \leq \frac{p}{N} \text{ и } \frac{1}{N^2} \leq \frac{4}{|x|^2} \text{ при } x \in B_{2N},$$

где $p > 0$ – некоторая константа. Тогда с учетом неравенства (3.9) будем иметь

$$\int\limits_{B_{2N}} \left|\nabla\eta\left(\frac{|x|}{N}\right)(\varphi - c)\right|^2 dx \leq \frac{p^2}{N^2} \int\limits_{B_{2N}\setminus B_N} |\varphi - c|^2\, dx \leq$$

$$\leq 4p^2 \int\limits_{B_{2N}\setminus B_N} \frac{|\varphi - c|^2}{|x|^2}\, dx \leq \frac{4p^2}{k} \int\limits_{B_{2N}} |\nabla\varphi|^2\, dx,$$

что и доказывает неравенство (3.10).

Пусть теперь $n = 2$. Согласно лемме 3,

$$\int\limits_{\mathbb{R}^2} |\nabla\varphi|^2\, dx \geq k \int\limits_{\mathbb{R}^2} \frac{|\varphi|^2}{|x|^2 \ln^2 |x|}\, dx. \tag{3.11}$$

Возьмем срезывающую функцию $\eta \in C_0^\infty(\mathbb{R}^2)$ такую, что $\eta(x) = 0$ в окрестности нуля и $\eta \equiv 1$ в открытой окрестности дополнения B_1, и пусть

$$\varphi_N = \eta\left(\frac{\ln\frac{|x|}{N}}{\ln N}\right)\varphi\,.$$

Тогда получаем цепочку неравенств

$$\operatorname{cap}_{\,\varphi}(B_{N^2} \cap (\mathbb{R}^2 \setminus \Omega)) \leq \int\limits_{B_{2N^2}} |\nabla\varphi_N|^2\, dx \leq \text{const} \int\limits_{B_{2N^2}} |\nabla\varphi|^2\, dx\,. \tag{3.12}$$

Действительно,

$$\int\limits_{B_{2N^2}} |\nabla\varphi_N|^2\, dx \leq 2\left(\int\limits_{B_{2N^2}} \left|\nabla\eta\left(\frac{\ln\frac{|x|}{N}}{\ln N}\right)\varphi\right|^2 dx + \int\limits_{B_{2N^2}} \left|\eta\left(\frac{\ln\frac{|x|}{N}}{\ln N}\right)\nabla\varphi\right|^2 dx\right).$$

Заметим, что

$$\left|\nabla\eta\left(\frac{\ln\frac{|x|}{N}}{\ln N}\right)\right| \leq \frac{2q}{|x|\ln N^2} \text{ и } \frac{1}{\ln^2 N^2} \leq \frac{m}{\ln^2 |x|} \text{ при } x \in B_{2N^2},$$

где q,$m > 0$ – некоторые константы. Тогда с учетом неравенства (3.11) будем иметь

$$\int\limits_{B_{2N^2}} \left|\nabla\eta\left(\frac{\ln\frac{|x|}{N}}{\ln N}\right)\varphi\right|^2 dx \leq \int\limits_{B_{2N^2}\setminus B_{N^2}} \frac{4q^2}{|x|^2\ln^2 N^2}|\varphi|^2\, dx \leq$$

$$4q^2 m \int\limits_{B_{2N^2}\setminus B_{N^2}} \frac{|\varphi|^2}{|x|^2\ln^2|x|}\, dx \leq \frac{4q^2 m}{k}\int\limits_{B_{2N^2}} |\nabla\varphi|^2\, dx,$$

что и доказывает неравенство (3.12).

Теперь рассмотрим при каких условиях последний интеграл будет конечен, а значит согласно теореме 1 задача (3.2) имеет решение. Очевидно, что конечность последнего интеграла эквивалентна условию

$$\int_0^\infty r^{n-1} r^{2(\alpha-1)} dr < \infty \,,$$

что приводит к неравенству $n-1+2(\alpha-1)+1 < 0$, откуда получается требуемое неравенство.

В обратную сторону. Пусть задача (3.2) имеет решение. Рассмотрим $\omega_k = \{x \in \mathbb{R}^n | 2^{k-1} \le |x| \le 2^k, x_n \ge -\varepsilon_k\}$, где ε_k такое, что справедливо условие (3.4). Вычислим коэффициенты $\sigma(\omega_k,\mu_k)$, где μ_k является $(n-1)$-мерной мерой Лебега на $\{x \in \mathbb{R}^n | 2^{k-1} \le |x| \le 2^k, x_n = 0\}$. Имеем

$$\int_{\partial I_k} |u - A|^2 \, dS \le 2^k B \int_{I_k} |\nabla u|^2 \, dx \,, \tag{3.13}$$

где A и B – некоторые константы, причем A зависит от u,k,n, а B зависит только от n, $I_k = \{x \in \mathbb{R}^n | 2^{k-1} \le |x| \le 2^k, x_n > 0\}$. В самом деле, заменой переменной $y = 2^{-k}x$ неравенство (3.13) приводится к виду

$$\int_{\partial I_0} |u - A|^2 \, dS \le B \int_{I_0} |\nabla u|^2 \, dy. \tag{3.14}$$

Установим справедливость последнего соотношения. Согласно неравенству Пуанкаре,

$$\|u - A\|^2_{W_2^1(I_0)} \le C \int_{I_0} |\nabla u|^2 \, dy \,,$$

где постоянная A зависит от u,n, а постоянная C – только от n. В то же время, согласно теореме вложения С.Л.Соболева,

$$\|u - A\|_{L_2(\partial I_0)} \le C_0 \|\nabla u\|_{W_2^1(I_0)} \,,$$

где постоянная C_0 зависит только от n. Таким образом, для доказательства (3.14) достаточно объединить две последние оценки. Из соотношения (3.13) немедлен-

но получим

$$\sigma(\omega_k,\mu_k)\|v-c\|_{L_2(\omega_k,\mu_k)} \le \|\nabla v\|_{L_2(\omega_k)}\,,$$

где

$$\sigma(\omega_k,\mu_k) = \frac{1}{B^{1/2}2^{k/2}}.$$

Тем самым, неравенство (3.14) теоремы 4 принимает вид

$$\sum_{k=1}^{\infty} 2^{-k} m_k^2(\varphi) < \infty\,. \tag{3.15}$$

Не сложно увидеть, что $m_k^2(\varphi) \ge \lambda 2^{k(2\alpha+n-1)}$ для некоторой постоянной $\lambda > 0$, не зависящей от k, поэтому (3.15) позволяет утверждать, что

$$\sum_{k=1}^{\infty} 2^{k(2\alpha+n-2)} < \infty\,,$$

откуда получим $2\alpha + n - 2 < 0$. □

Напомним, что для компакта e через $\mathfrak{M}_\varphi(e,\Omega)$ и $\mathfrak{R}(e,\Omega)$ обозначаются следующие классы функций

$$\mathfrak{M}_\varphi(e,\Omega) = \{u \in C_0^\infty(\Omega) : u = \varphi \text{ в окрестности } e\},$$
$$\mathfrak{R}(e,\Omega) = \{u \in C_0^\infty(\Omega) : u = 1 \text{ в окрестности } e\}.$$

Как обычно, в случае $\Omega = \mathbb{R}^n$ указание на Ω будем опускать.

В следующих предложениях устанавливаются основные свойства емкости $\mathrm{cap}_\varphi(e,\omega)$, аналогичные приведенным в монографии [31].

Предложение 1. *Пусть ω и Ω – открытые множества в $\mathbb{R}^n$, $D = diam\ \Omega < \infty$, $\overline{\omega} \subset \Omega$. Тогда для любого компакта $e \subset \omega$, удаленного на расстояние d_e от $\partial\omega$, имеет место неравенство*

$$cap_\varphi(e,\omega) \le C\left(\frac{D}{d_e}\right)^2 cap_\varphi(e,\Omega),$$

где C – положительная константа.

Доказательство. Пусть $u \in \mathfrak{M}_\varphi(e,\Omega)$ и $\alpha \in \mathfrak{R}(e,\omega)$, а также $|\nabla\alpha(x)| \le c\, d_e^{-1}$ [47, гл.6, §2], где c – положительная константа. Тогда

$$\text{cap}_{\,\varphi}(e,\omega) \le \|\nabla(\alpha u)\|^2_{L_2(\omega)} = \int\limits_\omega |u\nabla\alpha + \alpha\nabla u|^2 dx \le$$

$$\le 2\left(\int\limits_\omega |u\nabla\alpha|^2 dx + \int\limits_\omega |\alpha\nabla u|^2 dx\right) \le \frac{2c^2}{d_e^2}\int\limits_\omega |u|^2 dx + 2k\int\limits_\omega |\nabla u|^2 dx \le$$

$$\le c_1\left(\frac{\|u\|^2_{L_2(\Omega)}}{d_e^2} + \|\nabla u\|^2_{L_2(\Omega)}\right),$$

где $c_1 = \max\{2c^2, 2k\}$. Остается воспользоваться неравенством Фридрихса

$$\|u\|_{L_2(\Omega)} \le c_2 D\|\nabla u\|_{L_2(\Omega)},$$

где c_2 – положительная константа. Следовательно

$$c_1\left(\frac{\|u\|^2_{L_2(\Omega)}}{d_e^2} + \|\nabla u\|^2_{L_2(\Omega)}\right) \le c_1\left(\frac{c_2^2 D^2\|\nabla u\|^2_{L_2(\Omega)}}{d_e^2} + \|\nabla u\|^2_{L_2(\Omega)}\right) =$$

$$c_1\|\nabla u\|^2_{L_2(\Omega)}\left(\frac{c_2^2 D^2}{d_e^2} + 1\right) \le c_1\|\nabla u\|^2_{L_2(\Omega)}\left(\frac{c_2^2 D^2}{d_e^2} + \frac{D^2}{d_e^2}\right) \le C\left(\frac{D}{d_e}\right)^2\|\nabla u\|^2_{L_2(\Omega)},$$

где $C = c_1(c_2^2 + 1)$. Переходя к инфинимуму по всем функциям $u \in \mathfrak{M}_\varphi(e,\Omega)$, получаем необходимое неравенство. □

Предложение 2. *Пусть* $e \subset \Omega$, $D = diam\ \Omega$ *и* $d_e = dist\ \{e,\partial\Omega\}$. *Тогда*

$$c_{1,\varphi}\min\{1,d_e^2\}cap_{\,\varphi}(e,\Omega) \le Cap(e).$$

При этом существует константа c, *что*

$$Cap(e) \le c_2\max\{1, D^2\}cap_{\,\varphi-c}(e,\Omega).$$

Доказательство. Пусть $u \in \mathfrak{R}(e)$ и $\alpha \in \mathfrak{M}_\varphi(e,\Omega)$, а также $|\nabla\alpha(x)| \le c_\varphi\, d_e^{-1}$. Тогда

$$\operatorname{cap}_\varphi(e,\Omega) \le \|\nabla(\alpha u)\|^2_{L_2(\Omega)} = \int\limits_\Omega |u\nabla\alpha + \alpha\nabla u|^2 dx \le$$

$$\le 2\left(\int\limits_\Omega |u\nabla\alpha|^2 dx + \int\limits_\Omega |\alpha\nabla u|^2 dx\right) \le \frac{2c_\varphi^2}{d_e^2}\int\limits_{\mathbb{R}^n} |u|^2 dx + 2k_\varphi \int\limits_{\mathbb{R}^n} |\nabla u|^2 dx \le$$

$$\le c_{1,\varphi}^{-1}\left(\frac{\|u\|^2_{L_2(\mathbb{R}^n)}}{d_e^2} + \|\nabla u\|^2_{L_2(\mathbb{R}^n)}\right),$$

где $c_{1,\varphi} = (\max\{2c_\varphi^2, 2k_\varphi\})^{-1}$. Откуда и следует оценка для $\operatorname{Cap}(e)$ снизу.

Пусть теперь $u \in \mathfrak{R}(e,\Omega)$. Тогда

$$\operatorname{Cap}(e) \le \operatorname{Cap}(\Omega) \le \|u\|^2_{L_2(\Omega)} + \|\nabla u\|^2_{L_2(\Omega)} = \int\limits_\Omega |u|^2 dx + \int\limits_\Omega |\nabla u|^2 dx \le$$

$$k^2 D^2 \int\limits_\Omega |\nabla u|^2 dx + \int\limits_\Omega |\nabla u|^2 dx \le c'(D^2+1)\int\limits_\Omega |\nabla u|^2 dx,$$

где $c' = \max\{k^2,1\}$. Очевидно, что для любой функции u существуют такие константа c и функция $\alpha \in \mathfrak{M}_{\varphi-c}(e,\Omega)$, что

$$\|\nabla u\|_{L_2(\Omega)} \le \|\nabla\alpha\|_{L_2(\Omega)}.$$

Тогда, обозначая $c_2 = 2c'$, получаем необходимое неравенство. □

Предложение 3. *Пусть $e \subset B_r$. Тогда*

$$cap_\varphi(e,B_{2r}) \le k\, cap_\varphi(e), \tag{3.16}$$

где константа k зависит только от n.

Доказательство. Пусть $u \in \mathfrak{M}_\varphi(e)$ и $\alpha \in \mathfrak{R}(\overline{B}_r,B_{2r})$. Имеем

$$\|\nabla(\alpha u)\|_{L_2(B_{2r})} \le c(\|u\|_{L_2(B_{2r})} + \|\nabla u\|_{L_2(B_{2r})}) \le$$
$$\le c_1(\|u\|_{L_{\frac{2n}{n-2}}} + \|\nabla u\|_{L_2}) \le c_2\|\nabla u\|_{L_2}.$$

□

Предложение 4. *Пусть $e \subset \Omega \cap \mathbb{R}^{n-1}$. Тогда существует такая константа c и такая константа k, зависящая только от n, что*

$$cap_{\varphi-c}(e,\Omega) \geq k(m_{n-1}(e))^{\frac{n-2}{n-1}},$$

где m_s – s-мерная мера Лебега.

Доказательство. Для доказательства достаточно применить неравенство

$$\|u\|_{L_{\frac{2(n-1)}{n-2}}(\mathbb{R}^{n-1})} \leq c\|\nabla u\|_{L_2(\mathbb{R}^n)}$$

к произвольной функции $u \in \mathfrak{M}_{\varphi-c}(e,\Omega)$. □

С учетом [87], полезно иметь в виду следующие соотношения:
Если $n > 2$, то

$$\operatorname{cap}_{\varphi}(B_r) = c_{\varphi}\, r^{n-2}.$$

Если $n = 2$, то

$$\operatorname{cap}_{\varphi}(B_r) = c_{\varphi}\,(\log(2/r))^{-1}.$$

Лемма 6. *Пусть $n > 2$ и e – континуум с диаметром d. Тогда*

$$cap_{\varphi}(e) \sim d^{n-2}.$$

Доказательство. Заключим e в шар $\overline{B}_d^x$ с радиусом d (далее, просто B_d), где $x \in e$, и обозначим через B_{2d} концентрический шар с радиусом $2d$. Используя монотонность емкости, получаем

$$\operatorname{cap}_{\varphi}(e) \leq \operatorname{cap}_{\varphi}(\overline{B}_d) = c_{1,\varphi} d^{n-2}.$$

Пусть O и P – точки компакта e, $|O-P| = d$. Направим ось Ox_n от точки O к точке P. Введем обозначения: $x = (x',x_n), x' = (x_1,\ldots,x_{n-1}), e(t) = e \cap \{x : x_n = t\}, B_{2d}^{(n-1)}(t) = B_{2d} \cap \{x : x_n = t\}, \nabla' = (\partial/\partial x_1,\ldots,\partial/\partial x_{n-1})$. Для любой функции $u \in \mathfrak{M}_{\varphi}(e,B_{2d})$

$$\int\limits_{B_{2d}} |\nabla u|^2 dx \geq \int\limits_0^d dt \int\limits_{B_{2d}^{(n-1)}(t)} |\nabla' u|^2 dx' \geq \int\limits_0^d \operatorname{cap}_{\varphi}(e(t),B_{2d}^{(n-1)}(t))dt.$$

Так как $e(t) \neq \emptyset, e(t) \subset \overline{B}_d$, то

$$\text{cap}_{\,\varphi}(e(t), B_{2d}^{(n-1)}(t)) \geq c_{2,\varphi} d^{n-3}.$$

Минимизируя $\|\nabla u\|^2_{L_2(B_{2d})}$ на множестве $\mathfrak{M}_\varphi(e, B_{2d})$, получаем

$$\text{cap}_{\,\varphi}(e, B_{2d}) \geq \frac{c_{2,\varphi}}{n-2} d^{n-2}.$$

Остается воспользоваться оценкой (3.16). □

Лемма 7. *Пусть $n = 2$ и e – континуум с диаметром d, $2d < D$. Тогда*

$$\textit{cap}_{\,\varphi}(e, B_D) \sim (\log(D/d))^{-1},$$

где B_D – открытый шар с радиусом D и центром $O \in e$.

Доказательство. Сначала оценим емкость сверху. Пусть $B_d = \{x : |x| < d\}$. Определим на множестве $B_D \setminus B_d$ функцию $v : v(x) = (\log(D/d))^{-1} \log(D/|x|)$ и обозначим через α функцию из $C^\infty[0,1]$, равную нулю вблизи точки $t = 0$, φ вблизи $t = 1$. Пусть еще $u(x) = \alpha(v(x))$ при $x \in B_D \setminus B_d$, $u(x) = \varphi$ в B_d и $u(x) = 0$ вне B_D. Очевидно, что $u \in \mathfrak{M}_\varphi(\overline{B}_d, B_D)$. Кроме того, легко видеть, что $|\nabla u(x)| \leq c_\varphi (\log(D/d))^{-1} |x|^{-1}$ на $B_D \setminus B_d$. Отсюда получаем

$$\text{cap}_{\,\varphi}(\overline{B}_d, B_D) \leq \int\limits_{B_D} |\nabla u(x)|^2 dx \leq c_\varphi (\log(D/d))^{-2} \int\limits_{B_D \setminus B_d} |x|^{-2} dx = c_\varphi (\log(D/d))^{-1}.$$

Перейдем к доказательству нижней оценки емкости. Пусть P и Q – точки множества e, удаленные на расстояние d. Обозначим через (r, ω) сферические координаты точки в системе с центром Q, $r > 0, \omega \in S^{n-1}$ и через u – функцию из $\mathfrak{M}_\varphi(e, B_{2D}^Q)$, такую, что

$$\int\limits_{B_{2D}^Q} |\nabla u(x)|^2 dx \leq \gamma - \varepsilon,$$

где $\gamma = \text{cap}_{\,\varphi}(e, B_{2D}^Q) \leq \text{cap}_{\,\varphi}(e, B_D)$, а ε – малое положительное число. Введем еще функцию $U(r) = \|u(r, \cdot)\|_{L_2(S^{n-1})}$. Так как, по крайней мере, в одной точке сферы $\{x : |x| = r\}$, где $r < d$, функция u равна □

Лемма 8. *Пусть* $C_{\delta,d}$ *– цилиндр* $\{x=(x',x_n):|x'|\le\delta,|x_n|\le d/2\}$, *где* $0<2\delta<d$ *и* $Q_{2d}=\{x:|x_i|<d\}$. *Справедливы о-отношения*

$$cap_{\,\varphi}(C_{\delta,d},Q_{2d})\sim\begin{cases}d\delta^{n-3}, & \text{если } n>3\\ d(\log(d/\delta))^{-1}, & \text{если } n=3.\end{cases}$$

Доказательство. Пусть $u\in\mathfrak{M}_{\varphi}(C_{\delta,d},Q_{2d})$. Очевидно, что

$$\int\limits_{Q_{2d}}|\nabla u|^2dx\ge\int\limits_{-d/2}^{d/2}dx_n\int\limits_{Q_{2d}^{(n-1)}}|\nabla' u|^2dx',\tag{3.17}$$

где $\nabla'=(\partial/\partial x_1,\ldots,\partial/\partial x_{n-1}),Q_{2d}^{(n-1)}=\{x':|x_i|<d,i=1,\ldots,n-1\}$. Внутренний интеграл в правой части неравенства (3.17) не меньше, чем $\mathrm{cap}_{\,\varphi}(B_{\delta}^{(n-1)},B_{\rho}^{(n-1)})$, где $B_{\rho}^{(n-1)}$ – $(n-1)$-мерный шар $\{x':|x'|<\rho\}$ и $\rho=2(n-1)^{1/2}d$. Отсюда и из лемм 6 и 7 получаем, что этот интеграл мажорируется $c\delta^{n-3}$ при $n>3$ и $c(\log(d/\delta))^{-1}$ при $n=3$. Минимизируя левую часть неравенства (3.17) на множестве $\mathfrak{M}_{\varphi}(C_{\delta,d},Q_{2d})$, получаем требуемую оценку емкости снизу.

Перейдем к выводу верхней оценки. Пусть $u(x')$ – гладкая функция с носителем в шаре $B_d^{(n-1)}$, равная φ в окрестности шара $B_{\rho}^{(n-1)}$. Введем еще функцию η_d равенством $\eta_d(x)=\eta(x/d)$, где $\eta\in\mathfrak{R}(Q_1,Q_2)$. Так как функция $\eta_d(x)u(x')$ принадлежит классу $\mathfrak{M}_{\varphi}(C_{\delta,d},Q_{2d})$, то

$$\mathrm{cap}_{\,\varphi}(C_{\delta,d},Q_{2d})\le\int\limits_{Q_{2d}}|\nabla(\eta_d u)|^2dx\le cd\int\limits_{B_d^{(n-1)}}|\nabla' u|^2dx'.$$

Минимизируя последний интеграл и используя леммы 6 и 7, получаем требуемую верхнюю оценку емкости. □

3.3 Выводы

В третьей главе дано определение понятия “фронта выхода” и интенсивности фронта выходного потока. Описана зависимость интенсивности потока и эффективности процесса эвакуации при возникновении ЧС. Сформулирована краевая задача для эллиптических систем, заданных в неограниченных областях, решение которых соответствуют максимальной интенсивности потока. Получены прямая и обратная теоремы существования решения для таких систем.

Заключение

В ходе работы были получены следующие основные результаты:

1. Разработана имитационная агентная модель поведения толпы в условиях ЧС, основанная на индивидуальной системе принятия решений агентов, новизной которой является учет как многофакторности системы принятия решений агентами, так и стохастичности ряда процессов, в частности, учет влияния факторов внешней среды (стены, другие агенты, препятствия, взрыв и т.д.) на систему принятия решений агентом, учет радиуса личного пространства агентов и эффекта турбулентности толпы, детальная параметризация начального распределения агентов. Указанная модель была реализована в системе имитационного моделирования AnyLogic.
2. С целью повышения временной эффективности модели разработан эволюционный алгоритм нечеткой кластеризации, учитывающий факт наличия препятствий на пути следования агента, с учетом текущего направления движения агента, позволяющий существенно улучшить точность идентификации таксонов (кластеров толпы).
3. Доказана прямая и обратная теоремы существования решения для первой краевой задачи, определяющие максимальную интенсивностью фронта выходного потока при эвакуации.
4. Впервые получена аналитическая зависимость между параметрами модели и динамикой таксонов (кластеров) при эвакуации.
5. Разработан комплекс программ, отличающийся интеграцией имитационной модели поведения толпы с эволюционным алгоритмом нечеткой кластеризации, модулем кластерного анализа и базой данных системы, а также обеспечивающий возможность дальнейшего развития модели за счет объектно-ориентированного подхода.

Построенный комплекс программ отвечает требованиям прогнозирования, учета специфики агентов, а также требованиям быстродействия, основанным на законодательных актах. Разработанный программный комплекс внедрен в деятельность компании специализирующейся на проектировании систем пожарной безопасности и видеонаблюдений (ООО “ГЕНКЕЙ-ТЕЛЕКОМ”, приложение А.1).

Список сокращений и условных обозначений

БД - База данных

ОС - Операционная система

РАН - Российская академия наук

СУБД - Система управления базами данных

ЧС - Чрезвычайная ситуация

IDE - Integrated Development Environment

JDBC - Java DataBase Connectivity

ODBC - Open Database Connectivity

Словарь терминов

Толпа - Бесструктурное скопление людей, лишенных ясно осознаваемой общности целей, но связанных между собой сходством эмоционального состояния и общим объектом внимания

AnyLogic - Программное обеспечение для имитационного моделирования

Microsoft SQL Server - Система управления реляционными базами данных

RStudio - Среда разработки программного обеспечения для языка программирования R

Windows - Семейство операционных систем корпорации Microsoft

Список литературы

1. *Айвазян С. А., Енюков И. С., Мешалкин Л. Д.* Прикладная статистика: Исследование зависимостей. — М.: Финансы и статистика, 1985. — 487 с.

2. Прикладная статистика: Классификация и снижение размерности / С. А. Айвазян, В. М. Бухштабер, И. С. Енюков, Л. Д. Мешалкин. — М.: Финансы и статистика, 1989. — 607 с.

3. *Айвазян С. А., Енюков И. С., Мешалкин Л. Д.* Прикладная статистика: Основы моделирования и первичная обработка данных. — М.: Финансы и статистика, 1983. — 471 с.

4. *Айвазян С. А., Мхитарян В. С.* Прикладная статистика. Основы эконометрики: учебник для вузов в 2 т. — 2-е изд., испр. изд. — М.: ЮНИТИ-ДАНА, 2001. — Т. Т. 1. Теория вероятностей и прикладная статистика. — 656 с.

5. *Айвазян С. А.* Прикладная статистика. Основы эконометрики: учебник для вузов в 2 т. — 2-е изд., испр. изд. — М.: ЮНИТИ-ДАНА, 2001. — Т. Т. 2. Основы эконометрики. — 432 с.

6. *Акопов А. С.* Имитационное моделирование: учебник и практикум для академического бакалавриата. — М.: Юрайт, 2014. — 389 с.

7. *Андреева Г. М.* Социальная психология. Учебник для высших учебных заведении. — М.: Аспект-Пресс, 2001. — 290 с.

8. *Аптуков А. М., Брацун Д. А.* Моделирование групповой динамики толпы, паникующей в ограниченном пространстве // *Вестник Пермского университета. Механика*. — 2009. — № 3. — С. 18–23.

9. *Аптуков А. М., Брацун Д. А., Люшнин А. В.* Моделирование поведения паникующей толпы в многоуровневом разветвленном помещении // *Компьютерные исследования и моделирование*. — 2013. — Т. 5, № 3. — С. 491–508.

10. Микро- и макромодели социальных сетей: идентификация и имитационные эксперименты / А. В. Батов, В. В. Бреер, Д. А. Новиков, А. Д. Рогаткин // *Проблемы управления*. — 2014. — № 6. — С. 45–51.

11. *Бахвалов Н. С., Жидков Н. П., Кобельков Г.М.* Численные методы. — М.: Бином. Лаборатория знаний, 2011. — 640 с.

12. *Бекларян А. Л.* Имитационная модель поведения толпы в среде разработки AnyLogic // *Вестник Бурятского государственного университета*. — 2015. — № 9. — С. 40–53.

13. *Бекларян А. Л., Акопов А. С.* Моделирование поведения толпы на основе интеллектуальной динамики взаимодействующих агентов // *Бизнес-информатика*. — 2015. — Т. 31, № 1. — С. 69–77.

14. *Бекларян А. Л.* О существовании решений первой краевой задачи для эллиптических систем высокого порядка в неограниченных областях // *Математические заметки*. — 2014. — Т. 96, № 2. — С. 310–313.

15. *Бекларян А. Л.* Фронт выхода в модели поведения толпы при чрезвычайных ситуациях // *Вестник Тамбовского государственного университета. Серия: естественные и технические науки*. — 2015. — Т. 20, № 4. — С. 851–856.

16. *Бернс Л., Джон Ф., Шехтер М.* Уравнения с частными производными. — М.: Мир, 1966. — 352 с.

17. *Боев В. Д.* Компьютерное моделирование: Пособие для практических занятий, курсового и дипломного проектирования в AnyLogic 7. — СПб.: ВАС, 2014. — 432 с.

18. *Бреер В. В., Новиков Д. А., Рогаткин А. Д.* Микро- и макромодели социальных сетей: теория // *Проблемы управления*. — 2014. — № 5. — С. 28–33.

19. *Бреер В. В., Новиков Д. А.* Модели управления толпой // *Проблемы управления*. — 2012. — № 2. — С. 38–44.

20. *Бреер В. В., Новиков Д. А., Рогаткин А. Д.* Стохастические модели управления толпой // *Управление большими системами*. — 2014. — № 52. — С. 85–117.

21. *Владимиров В. С.* Уравнения математической физики. — 4-е изд., испр. и доп. изд. — М.: Наука, 1981. — 512 с.

22. *Гельфанд И. М., Шилов Г. Е.* Обобщенные функции и действия над ними (Обобщенные функции, выпуск 1). — 2-е изд. изд. — М.: Физматлит, 1959. — 470 с.

23. *Головин С. Ю.* Словарь практического психолога. — Минск: Харвест, 1998. — 976 с.

24. *Кирик Е. С., Круглов Д. В., Юргельян Т. Б.* О дискретной модели движения людей с элементом анализа окружающей обстановки // *Журнал СФУ, Серия «Математика и физика»*. — 2008. — Т. 1, № 3. — С. 266–276.

25. *Колмогоров А. Н., Фомин С. В.* Элементы теории функций и функционального анализа. — 7-е изд. изд. — М.: Физматлит, 2004. — 572 с.

26. *Коньков А. А.* О размерности пространства решений эллиптических систем в неограниченных областях // *Математический сборник*. — 1993. — Т. 184, № 12. — С. 23–52.

27. *Краснощеков П. С.* Простейшая математическая модель поведения. Психология конформизма // *Математическое моделирование*. — 1998. — Т. 10, № 7. — С. 76–92.

28. *Ладыженская О. А., Уральцева Н. Н.* Линейные и квазилинейные уравнения эллиптического типа. — 2-е изд. изд. — М.: Наука, 1973. — 576 с.

29. *Лебон Г.* Психология народов и масс. — М.: Академический проект, 2011. — 238 с.

30. *Ландау Л. Д., Лифшиц Е. М.* Теоретическая физика: учебное пособие в 10 т. — 5-е изд., стереотип. изд. — М.: ФИЗМАТЛИТ, 2012. — Т. I. Механика. — 224 с.

31. *Мазья В. Г.* Пространства С.Л. Соболева. — Л.: Изд-во ЛГУ, 1985. — 416 с.

32. *Максимова М. З., Барбин Н. М.* Моделирование эвакуации людей различных возрастных групп // *Компьютерные исследования и моделирование*. — 2013. — Т. 5, № 3. — С. 483–490.

33. *Мезенцев К. Н.* Практикум «Моделирование систем в среде AnyLogic 6.4.1». Часть 1. — М.: МАДИ, 2011. — 109 с.

34. *Мезенцев К. Н.* Практикум «Моделирование систем в среде AnyLogic 6.4.1». Часть 2 / Под ред. Заслуженного деятеля науки РФ д.т.н. профессора А.Б.Николаева. — М.: МАДИ, 2011. — 103 с.

35. *Михлин С. Г.* Линейные уравнения в частных производных. — М.: Высшая школа, 1977. — 423 с.

36. Моделирование и прогнозирование многопараметрических экономических и технологических процессов. — URL: http://www.mallenom.ru/article021.php (дата обращения: 12.08.2015).

37. Об утверждении методики определения расчетных величин пожарного риска в зданиях, сооружениях и строениях различных классов функциональной пожарной опасности: Приказ МЧС РФ от 30.06.2009 № 382 (ред. от 02.12.2015). — СПС КонсультантПлюс, 2015. — 73 с.

38. Официальный сайт коммерческого симулятора Di-Guy, имитирующего поведение больших масс людей. — URL: http://www.diguy.com/diguy (дата обращения: 12.08.2015).

39. Официальный сайт коммерческого симулятора Myriad II, имитирующего процесс эвакуации. — URL: http://www.crowddynamics.com/evacuation-demo.php (дата обращения: 12.08.2015).

40. Официальный сайт коммерческого симулятора Pedestrian Dynamics. — URL: http://www.pedestrian-dynamics.com/pedestrian-dynamics (дата обращения: 12.08.2015).

41. Официальный сайт проекта Gamma. — URL: http://gamma.cs.unc.edu (дата обращения: 12.08.2015).

42. Официальный сайт фреймфорка Menge. — URL: http://gamma.cs.unc.edu/Menge/ (дата обращения: 12.08.2015).

43. *Предтеченский В. М., Милинский А. И.* Проектирование зданий с учетом организации движения людских потоков: учебник для вузов. — М.: Стройиздат, 1979. — 375 с.

44. “СП 1.13130.2009. Свод правил. Системы противопожарной защиты. Эвакуационные пути и выходы” (утв. Приказом МЧС России от 25.03.2009 N 171) (ред. от 09.12.2010). — СПС КонсультантПлюс, 2010. — 50 с.

45. Сибирские ученые научились моделировать эвакуацию при пожаре. — URL: http://www.sbras.info/articles/science/sygraem-v-pozhar (дата обращения: 25.11.2015).

46. Список погибших в клубе «Хромая лошадь». — URL: http://permloshad.narod.ru (дата обращения: 12.08.2015).

47. *Стейн И.* Сингулярные интегралы и дифференциальные свойства функций. — М.: Мир, 1973. — 344 с.

48. *Степанцов М.Е.* Математическая модель направленного движения группы людей // *Математическое моделирование*. — 2004. — Т. 16, № 3. — С. 43–49.

49. *Фрейд З.* Психология масс и анализ человеческого “Я”. — М.: Азбука-Аттикус, 2013. — 192 с.

50. *Харди Г. Г., Литлвуд Дж. Е., Пойа Г.* Неравенства. — М.: Изд-во иностр. лит., 1948. — 456 с.

51. Число погибших в давке на фестивале в Камбодже превысило 450 человек. — URL: http://www.rian.ru/world/20101124/300360894.html (дата обращения: 12.08.2015).

52. *Шилдт Г.* Java. Полное руководство. — М.: Вильямс, 2012. — 1104 с.

53. *Юнг К. Г.* Психология бессознательного. — М.: Канон+РООИ “Реабилитация”, 2012. — 320 с.

54. Федеральный закон от 22.07.2008 N 123-ФЗ (ред. от 13.07.2015) “Технический регламент о требованиях пожарной безопасности”. — СПС КонсультантПлюс, 2015. — 102 с.

55. *Akopov A. S., Beklaryan L. A.* An Agent Model of Crowd Behavior in Emergencies // *Automation and Remote Control*. — 2015. — Vol. 76, no. 10. — Pp. 1817–1827.

56. *Akopov A. S., Beklaryan L. A.* Simulation of human crowd behavior in extreme situations // *International Journal of Pure and Applied Mathematics*. — 2012. — Vol. 79, no. 1. — Pp. 121–138.

57. *Almeida J. E., Rosseti R., Coelho A. L.* Crowd Simulation Modeling Applied to Emergency and Evacuation Simulations using Multi-Agent Systems // DSIE'11 – 6th Doctoral Symposium on Informatics Engineering. — Engineering Faculty of Porto University, Porto, 2011. — Pp. 93–104.

58. *Antonini G., Bierlaire M., Weber M.* Discrete choice models of pedestrian walking behavior // *Transportation Research Part B*. — 2006. — Vol. 40, no. 8. — Pp. 667–687.

59. *Beklaryan A. L.* On the existence of solutions of the first boundary value problem for elliptic equations on unbounded domains // *International Journal of Pure and Applied Mathematics*. — 2013. — Vol. 88, no. 4. — Pp. 499–522.

60. *Beklaryan A. L.* On the existence of solutions of the first boundary value problem for elliptic equations on unbounded domains // *Russian Journal of Mathematical Physics*. — 2012. — Vol. 19, no. 4. — Pp. 509–510.

61. *Bezdek C. J.* Cluster Validity with Fuzzy Sets // *Journal of Cybernetics*. — 1974. — Vol. 3, no. 3. — Pp. 58–73.

62. *Bezdek C. J.* Pattern Recognition with Fuzzy Objective Function Algorithms. — Norwell, Mass.: Kluwer Academic Publishers, 1981. — 256 pp.

63. *Bohannon R. W.* Comfortable and maximum walking speed of adults aged 20-79 years: reference values and determinants // *Age and Ageing*. — 1997. — Vol. 26, no. 1. — Pp. 15–19.

64. Computational Modeling of Collective Behavior of Panicked Crowd Escaping Multi-floor Branched Building / D. Bratsun, I. Dubova, M. Krylova, A. Lyushnin // Proceedings of the European Conference on Complex Systems 2012. — Springer International Publishing, 2013. — Pp. 659–663.

65. *Cherif F., Chighoub R.* Crowd simulation influenced by agent's sociopsychological state // *Journal of computing*. — 2010. — Vol. 2, no. 4. — Pp. 48–54.

66. *Chi Z., Yan H., Pham T.* Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition. — Singapore: World Scientific, 2006. — 225 pp.

67. *Costella J. P.* Galilean Antialiasing for Virtual Reality Displays. — Parkville Vic. 3052, Australia: School of Physics, The University of Melbourne, 1992. — 79 pp.

68. *Dunn J. C.* A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters // *Journal of Cybernetics*. — 1973. — Vol. 3, no. 3. — Pp. 32–57.

69. *Eubanks J., Hill P. F.* Pedestrian accident reconstruction and litigation. — Tucson: Lawyers & Judges Pub., 1998. — 992 pp.

70. Agent Perception Modeling for Movement in Crowds / K. Farrahi, K. Zia, A. Sharpanskykh et al. // 11th International Conference Advances on Practical Applications of Agents and Multi-Agent Systems. — Springer Berlin Heidelberg, 2013. — Pp. 73–84.

71. *Fangqin T., Aizhu R.* Agent-Based Evacuation Model Incorporating Fire Scene and Building Geometry // *Tsinghua Science And Technology*. — 2008. — Vol. 13, no. 5. — Pp. 708–714.

72. *Golas A., Narain R., Lin M. C.* Continuum modeling of crowd turbulence // *Physical review E*. — 2014. — Vol. 90, no. 4. — P. 042816.

73. *Golas A., Narain R., Lin M. C.* Hybrid long-range collision avoidance for crowd simulation // I3D'13 Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. — ACM New York, 2013. — Pp. 29–36.

74. Simulating heterogeneous crowd behaviors using personality trait theory / S. J. Guy, S. Kim, M. C. Lin, D. Manocha // SCA'11 Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. — ACM New York, 2011. — Pp. 43–52.

75. PLEdestrians: A Least-Effort Approach to Crowd Simulation / S. J. Guy, J. Chhugani, S. Curtis et al. // SCA'10 Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. — Eurographics Association Aire-la-Ville, Switzerland, 2010. — Pp. 119–128.

76. ClearPath: highly parallel collision avoidance for multi-agent simulation / S. J. Guy, J. Chhugani, S. Kim et al. // SCA'09 Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. — ACM New York, 2009. — Pp. 177–187.

77. *Hartigan J. A., Wong M. A.* A K-Means Clustering Algorithm // *Journal of the Royal Statistical Society. Series C (Applied Statistics)*. — 1979. — Vol. 28, no. 1. — Pp. 100–108.

78. *Helbing D., Johansson A., Al-Abideen H. Z.* Dynamics of crowd disasters: An empirical study // *Physical review E*. — 2007. — Vol. 75, no. 4. — Pp. 0461091–0461097.

79. *Helbing D., Farkas I., Vicsek T.* Simulating dynamical features of escape panic // *Nature*. — 2000. — no. 407. — Pp. 487–490.

80. Simulation of Pedestrian Crowds in Normal and Evacuation Situations / D. Helbing, I. Farkas, P. Molnar, T. Vicsek // *Pedestrian and evacuation dynamics*. — 2002. — Vol. 21, no. 2. — Pp. 21–58.

81. *Helbing D., Molnar P.* Social force model for pedestrian dynamics // *Physical review E*. — 1995. — Vol. 51, no. 5. — Pp. 4282–4286.

82. *Helbing D.* Traffic and related self-driven many-particle systems // *Reviews of Modern Physics*. — 2001. — Vol. 73, no. 4. — Pp. 1067–1141.

83. *Henderson L. F.* The statistics of crowd fluids // *Nature*. — 1971. — no. 229. — Pp. 381–383.

84. *Huang H.-J., Guo R.-Y.* Static floor field and exit choice for pedestrian evacuation in rooms with internal obstacles and multiple exits // *Physical review E*. — 2008. — Vol. 78, no. 2. — P. 021131.

85. Collective Behavior in a Simulated Panic Situation / H. Kelley, J. Condry, A. Dahlke, A. Hill // *Journal of experimental social psychology*. — 1965. — Vol. 1, no. 1. — Pp. 20–54.

86. Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory / S. Kim, S. J. Guy, D. Manocha, M. C. Lin // I3D'12 Proceedings

of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. — ACM New York, 2012. — Pp. 55–62.

87. *Meyers N. G.* A theory of capacities for potentials of functions in Lebesgue classes // *Math. Scand.* — 1970. — Vol. 6, no. 2. — Pp. 255–292.

88. *Mintz A.* Non-adaptive group behavior // *Journal of Abnormal Psychology*. — 1951. — Vol. 46, no. 2. — Pp. 150–159.

89. *Moussaida M., Helbing D., Theraulaza G.* How simple rules determine pedestrian behavior and crowd disasters // *PNAS*. — 2011. — Vol. 108, no. 17. — P. 6884–6892.

90. Aggregate Dynamics for Dense Crowd Simulation / R. Narain, A. Golas, S. Curtis, M. C. Lin // SIGGRAPH Asia'09 ACM SIGGRAPH Asia 2009 papers. — ACM New York, 2009. — Pp. 122:1–122:8.

91. Tech. Rep.: / N. Pelechano, K. O'Brien, B. Silverman, N. Badler: Pennsylvania univ philadelphia center for human modeling and simulation, 2005.

92. *Ruspini E. H.* Numerical methods for fuzzy clustering // *Information Sciences*. — 1970. — Vol. 2, no. 3. — Pp. 319–350.

93. *Teknomo K., Takeyama Y., Inamura H.* Data collection method for pedestrian movement variables // *Civil Engineering Dimension*. — 2000. — Vol. 2, no. 1. — Pp. 43–48.

94. *Wagner N., Agrawal V.* An agent-based simulation system for concert venue crowd evacuation modeling in the presence of a fire disaster // *Expert Systems with Applications*. — 2014. — Vol. 41, no. 6. — Pp. 2807–2815.

95. *Ward J. H.* Hierarchical grouping to optimize an objective function // *Journal of the American Statistical Association*. — 1963. — Vol. 58, no. 301. — Pp. 236–244.

96. Energy Cost of Walking in Normal Children and Teenagers / R. L. Waters, H. J. Hislop, L. Thomas, J. Campbell // *Developmental Medicine & Child Neurology*. — 1983. — Vol. 25, no. 2. — Pp. 184–188.

97. *Yu W., Johansson A.* Modeling crowd turbulence by many-particle simulations // *Physical review E*. — 2007. — Vol. 76, no. 4. — P. 046105.

98. *Zebala J., Ciepka P., Reza A.* Pedestrian acceleration and speeds // *Problems of Forensic Sciences*. — 2012. — Vol. 91. — Pp. 227–234.

99. Pedestrian motion speed while crossing the road / J. Zebala, P. Ciepka, A. Reza et al. // Proceedings of the 6th International Scientific Conference TRANSBALTICA 2009. — Vilnius Gediminas Technical University, 2012. — Pp. 251–255.

100. Automatic rule identification for agent-based crowd models through gene expression programming / J. Zhong, L. Luo, W. Cai, M. Lees // Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems. — International Foundation for Autonomous Agents and Multiagent Systems, Richland, 2014. — Pp. 1125–1132.

Список рисунков

1.1 Описание моделируемого помещения. 24
1.2 Описание точки выхода. 29
1.3 Зависимость радиуса личного пространства i-ого агента от плотности толпы $\rho_i(t)$. 31
1.4 Пространство выбора агента. 32
1.5 Гибридная область анализа в случае эффекта турбулентности. . . . 36
1.6 Геометрия взрыва. 37
1.7 График зависимость $sit_i(t)$ от времени. 39
1.8 Пример кластеризации. 41
1.9 Схема алгоритмов нечеткой кластеризации. 45
1.10 Динамика на уровне агентов. 48
1.11 Динамика на уровне кластеров. 48
1.12 Количество кластеров по времени. 52

2.1 Сравнение характеристик современных систем имитационного моделирования. 55
2.2 Схема БД. 57
2.3 Таблица *square*. 58
2.4 Таблица *gender*. 58
2.5 Таблица *velocity*. 58
2.6 Таблица *version*. 59
2.7 Таблица *functional*. 59
2.8 Таблица *coordinates*. 60
2.9 Таблица *explosion*. 61
2.10 Таблица *clusterization*. 61
2.11 Таблица *compactness*. 62
2.12 Таблица *agent_cluster*. 62
2.13 Таблица *cluster_center*. 63
2.14 Таблица *membership*. 63
2.15 Группы параметров класса *Simulation*. 65
2.16 Область заполнения количества агентов в клетках. 66

2.17 Динамическое отображение характеристик для разных распределений. 66
2.18 Группы параметров класса *Main*. 67
2.19 Архитектура программного комплекса. 73

3.1 Скопление агентов у выходов. 75
3.2 Фронт выхода. 77

Список таблиц

1 Требования к разрабатываемой модели и программному комплексу 19
2 Комфортная и максимальная скорости ходьбы для разных гендерно-возрастных групп, приведенные в работах [63; 96] 26
3 Площади горизонтальной проекции взрослых людей 27
4 Площади горизонтальной проекции детей и подростков 27
5 Усредненные площади горизонтальной проекции 28
6 Максимальная скорость бега для разных гендерно-возрастных групп 38
7 Сравнение алгоритмов кластеризации 47
8 Соответствие требованиям, предъявляемым к разрабатываемой модели и программному комплексу 49
9 Сравнение эффективности работы модели 72

Приложение А

Внедрение результатов

А.1 Справка о внедрении результатов

GEN·KEY

ООО "ГЕНКЕЙ-ТЕЛЕКОМ" 107113, Москва, ул. Сокольнический вал, 38.
Тел: (499)268-5915, факс: (499)737-8901
ИНН 7718153493 рег. № 1851
КПП 771801001

Исх. № 04/04 от 25.01.2016 г.

СПРАВКА О ВНЕДРЕНИИ

Настоящим подтверждаю, что результаты диссертационной работы Бекларяна Армена Левоновича на соискание ученой степени кандидата технических наук на тему «Агентная модель поведения толпы в условиях чрезвычайной ситуации для оценки интенсивности фронта выходного потока», внедрены и используются в деятельности компании ООО «ГЕНКЕЙ-ТЕЛЕКОМ».

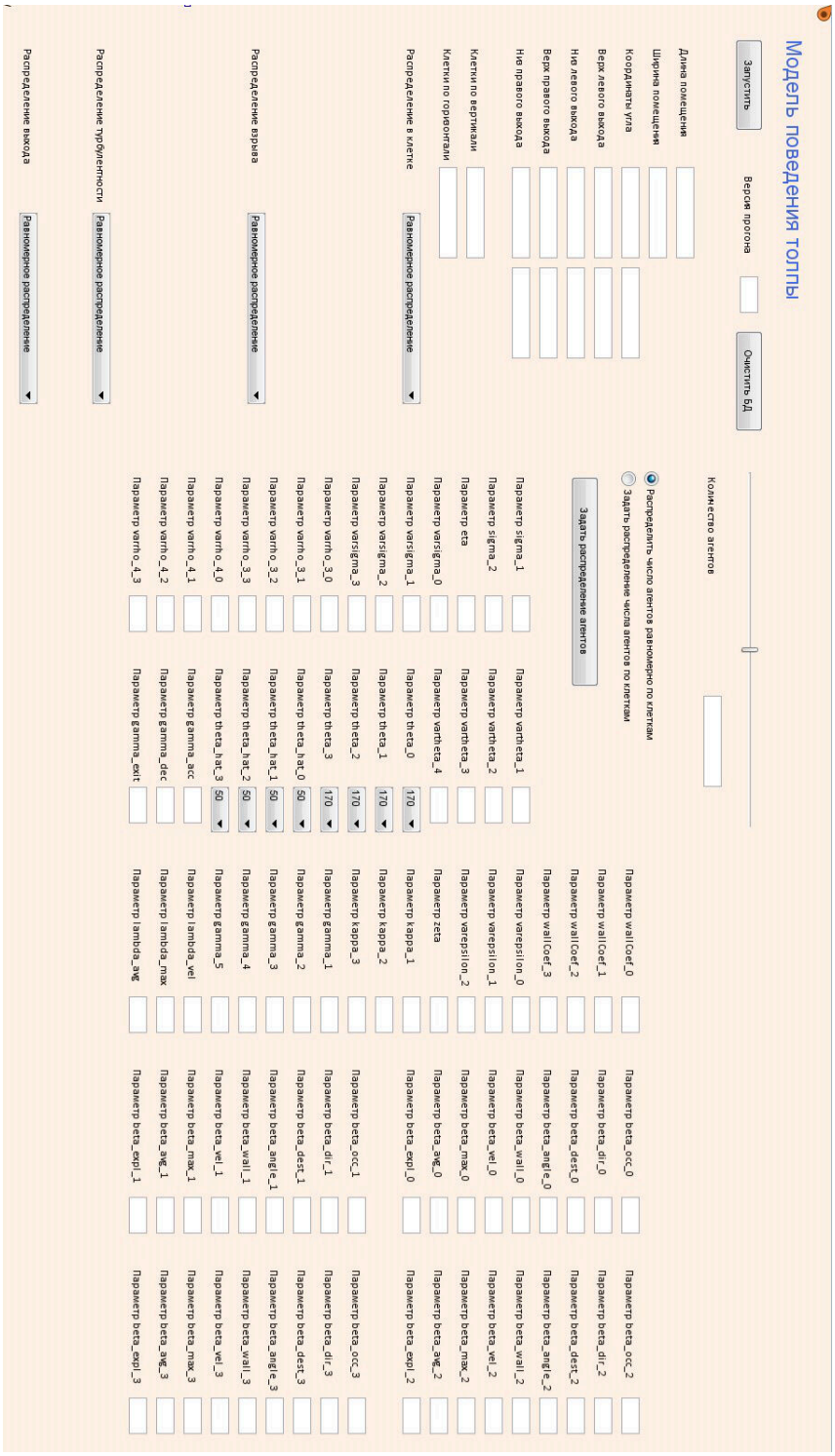
Бекларяном А.Л. разработана имитационная модель поведения толпы в условиях чрезвычайной ситуации (ЧС), реализованная в виде программного комплекса, и позволяющая в частности, оценивать интенсивность выходного потока на выходе из различных помещений. Данный программный комплекс используется ООО «ГЕНКЕЙ-ТЕЛЕКОМ» при проектировании и монтаже систем охранно-пожарной сигнализации и пожаротушения, в основном для определения наилучших мест установки видеорегистраторов и датчиков движения, для локализации зон возможной «турбулентности толпы» и давки при возникновении ЧС, для прогнозирования динамики развития ЧС с целью формирования оптимальных планов противопожарной эвакуации.

Генеральный директор
ООО «ГЕНКЕЙ-ТЕЛЕКОМ» _____/Сотский С.В./

Приложение Б

Пользовательский интерфейс модели

Б.1 Скрин пользовательского интерфейса



Б.2 Листинг кода пользовательского интерфейса

```
package crowd;

import java.io.Serializable;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.LinkedHashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.PriorityQueue;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.markup.Network;
import com.xj.anylogic.engine.Position;
import com.xj.anylogic.engine.markup.PedFlowStatistics;
import com.xj.anylogic.engine.markup.DensityMap;

import static java.lang.Math.*;
import static com.xj.anylogic.engine.UtilitiesArray.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;

import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.markup.*;
import com.xj.anylogic.engine.presentation.*;

import static crowd.Gender.*;
import static crowd.Status.*;
import static crowd.Situation.*;
import static crowd.Exit.*;

import javax.swing.JApplet;


import com.sun.media.sound.Toolkit;
import java.awt.GraphicsEnvironment;
import java.awt.GraphicsDevice;
import java.awt.Rectangle;
import java.awt.Insets;
import java.awt.GraphicsConfiguration;


public class Simulation extends ExperimentSimulation<Main> {
```

```
  @AnyLogicInternalCodegenAPI
  public static String[] COMMAND_LINE_ARGUMENTS_xjal = new String[0];
  {
    setCommandLineArguments_xjal( COMMAND_LINE_ARGUMENTS_xjal );
  }
  // Базы данных
  public Database var_database = new Database( this, "var_database", Database.DRIVER_JDBC_MICROSOFT_SQLSERVER, "ARMEN", "CROWD", "sa", new
      char[] { 't', 'a', 't', 'a', 'm', 'o', 'l', 'o', 'd', 'e', 'c' } );

  // Запросы
  public Query query_version = new Query( "query_version",
var_database
, false, "SELECT ISNULL((SELECT MAX(version) from version),0) as Version" );
  public Query query_maxArea = new Query( "query_maxArea",
var_database
, false, "SELECT max(Square) as MAX from square" );

  // Простые переменные
  /**
   * Параметр len1
   */
  public
double
 var_len1 =
100 ;
  /**
   * Параметр len2
   */
  public
double
 var_len2 =
60 ;
  /**
   * Параметр a0
   */
  public
double
 var_a0 =
15 ;
  /**
   * Параметр b0
   */
  public
double
 var_b0 =
15 ;
  /**
   * Параметр a11
   */
  public
double
 var_a11 =
15 ;
  /**
   * Параметр b11
   */
  public
double
 var_b11 =
35 ;
  /**
   * Параметр a12
   */
  public
double
 var_a12 =
15 ;
  /**
   * Параметр b12
   */
  public
double
 var_b12 =
55 ;
  /**
   * Параметр a21
   */
  public
double
 var_a21 =
115 ;
  /**
```

```
    * Параметр b21
160  */
   public
double
 var_b21 =
35 ;
165 /**
    * Параметр a22
    */
   public
double
170 var_a22 =
115 ;
   /**
    * Параметр b22
    */
175 public
double
 var_b22 =
55 ;
   /**
180  * Параметр mvert
    */
   public
int
 var_mvert =
185 1 ;
   /**
    * Параметр mhor
    */
   public
190 int
 var_mhor =
1 ;
   /**
    * Распределение агентов в клетке
195  */
   public
String
 var_distr =
"Равномерное распределение" ;
200 /**
    * Параметр сдвига распределения Лапласа по оси X для агентов
    */
   public
double
205 var_b_lapl_x =
0 ;
   /**
    * Параметр растяжения распределения Лапласа по оси X для агентов
    */
210 public
double
 var_a_lapl_x =
1 ;
   /**
215  * Математическое ожидание нормального распределения по оси X для агентов
    */
   public
double
 var_mu_norm_x =
220 0 ;
   /**
    * Дисперсия нормального распределения по оси X для агентов
    */
   public
225 double
 var_sig_norm_x =
1 ;
   /**
    * Мода треуголного распределения по оси X для агентов
230  */
   public
double
 var_c_trian_x =
0 ;
235 /**
    * Общее количество агентов
    */
   public
int
240 var_agents_count =
```

```
10 ;
  /**
   * Равномерное распределение или ручное распределение агентов по клеткам<br>
   * false — равномерное<br>
   * true — ручное
   */
  public
boolean
 var_global_distr =
false ;
  /**
   * Параметр sigma_2
   */
  public
double
 var_sigma_2 =
1.1 ;
  /**
   * Параметр sigma_1
   */
  public
double
 var_sigma_1 =
var_sigma_2 ;
  /**
   * Параметр eta
   */
  public
double
 var_eta =
5 ;
  /**
   * Массив параметра varsigma
   */
  public
double[]
 var_varsigma = new double[]
{1,1.05,1.1,1.15} ;
  /**
   * Массив параметра varrho_3
   */
  public
double[]
 var_varrho_3 = new double[]
{0.83,0.82,0.81,0.8} ;
  /**
   * Массив параметра varrho_4
   */
  public
double[]
 var_varrho_4 = new double[]
{0.9,0.91,0.92,0.93} ;
  /**
   * Массив параметра vartheta
   */
  public
double[]
 var_vartheta = new double[]
{3,2,1,4} ;
  /**
   * Массив параметра theta
   */
  public
int[]
 var_theta = new int[]
{170,170,120,80} ;
  /**
   * Массив параметра theta_hat
   */
  public
int[]
 var_theta_hat = new int[]
{50,50,30,30} ;
  /**
   * Параметр gamma_acc
   */
  public
double
 var_gamma_acc =
0.5 ;
  /**
   * Параметр gamma_dec
```

```
    */
   public
double
 var_gamma_dec =
0.5 ;
   /**
    * Параметр gamma_exit
    */
   public
double
 var_gamma_exit =
0.5 ;
   /**
    * Параметр varepsilon_0
    */
   public
double
 var_varepsilon_0 =
5 ;
   /**
    * Параметр varepsilon_1
    */
   public
double
 var_varepsilon_1 =
10 ;
   /**
    * Параметр varepsilon_2
    */
   public
double
 var_varepsilon_2 =
13 ;
   /**
    * Павраметр zeta
    */
   public
int
 var_zeta =
3 ;
   /**
    * Параметр kappa_1
    */
   public
double
 var_kappa_1 =
0.5 ;
   /**
    * Параметр kappa_2
    */
   public
double
 var_kappa_2 =
2 ;
   /**
    * Параметр kappa_3
    */
   public
double
 var_kappa_3 =
3 ;
   /**
    * Массив весового коэффициента beta_occ
    */
   public
double[]
 var_beta_occ = new double[]
{0.5,0.7,0.8,0.9} ;
   /**
    * Массив весового коэффициента beta_dir
    */
   public
double[]
 var_beta_dir = new double[]
{0.3,0.5,0.5,0.1} ;
   /**
    * Массив весового коэффициента beta_wall
    */
   public
double[]
 var_beta_wall = new double[]
{1,1,1,1} ;
```

```
405   /**
       * Массив весового коэффициента beta_vel
       */
      public
double[]
410  var_beta_vel = new double[]
{0.2,0,0,0} ;
      /**
       * Массив весового коэффициента beta_dest
       */
415   public
double[]
 var_beta_dest = new double[]
{0.7,0.7,0.8,0.9} ;
      /**
420    * Массив весового коэффициента beta_angle
       */
      public
double[]
 var_beta_angle = new double[]
425 {0.3,0.5,0.6,0.7} ;
      /**
       * Массив весового коэффициента beta_max
       */
      public
430 double[]
 var_beta_max = new double[]
{0,0.6,0.7,0.8} ;
      /**
       * Массив весового коэффициента beta_expl
435    */
      public
double[]
 var_beta_expl = new double[]
{0,0.5,0.6,1} ;
440   /**
       * Нормирующий коэффициент gamma_1
       */
      public
double
445  var_gamma_1 =
0.5 ;
      /**
       * Нормирующий коэффициент gamma_2
       */
450   public
double
 var_gamma_2 =
0.5 ;
      /**
455    * Нормирующий коэффициент gamma_3
       */
      public
double
 var_gamma_3 =
460 0.5 ;
      /**
       * Нормирующий коэффициент gamma_4
       */
      public
465 double
 var_gamma_4 =
0.5 ;
      /**
       * Нормирующий коэффициент lambda_vel
470    */
      public
double
 var_lambda_vel =
0.5 ;
475   /**
       * Нормирующий коэффициент lambda_max
       */
      public
double
480  var_lambda_max =
0.5 ;
      /**
       * Распределение центра взрыва
       */
485   public
String
```

```
 var_distr_expl =
"Равномерное распределение" ;
  /**
   * Параметр сдвига распределения Лапласа по оси X для взрыва
   */
  public
double
 var_b_lapl_expl_x =
0 ;
  /**
   * Параметр растяжения распределения Лапласа по оси X для взрыва
   */
  public
double
 var_a_lapl_expl_x =
1 ;
  /**
   * Математическое ожидание нормального распределения по оси X для взрыва
   */
  public
double
 var_mu_norm_expl_x =
0 ;
  /**
   * Дисперсия нормального распределения по оси X для взрыва
   */
  public
double
 var_sig_norm_expl_x =
1 ;
  /**
   * Мода треуголного распределения по оси X для взрыва
   */
  public
double
 var_c_trian_expl_x =
0 ;
  /**
   * Распределение distr_exit
   */
  public
String
 var_distr_exit =
"Равномерное распределение" ;
  /**
   * Распределение distr_turb
   */
  public
String
 var_distr_turb =
"Равномерное распределение" ;
  /**
   * Максимальный угол отклонения при турбулентности
   */
  public
int
 var_turb =
30 ;
  /**
   * Нормирующий коэффициент gamma_5
   */
  public
double
 var_gamma_5 =
0.5 ;
  /**
   * Нормирующий коэффициент lambda_avg
   */
  public
double
 var_lambda_avg =
0.5 ;
  /**
   * Массив весового коэффициента beta_avg
   */
  public
double[]
 var_beta_avg = new double[]
{0.5,0.5,0.6,0.7} ;
  /**
   * Параметр сдвига распределения Лапласа по оси Y для агентов
   */
```

```
  public
570 double
 var_b_lapl_y =
0 ;
  /**
   * Параметр растяжения распределения Лапласа по оси Y для агентов
575  */
  public
double
 var_a_lapl_y =
1 ;
580 /**
   * Математическое ожидание нормального распределения по оси Y для агентов
   */
  public
double
585 var_mu_norm_y =
0 ;
  /**
   * Дисперсия нормального распределения по оси Y для агентов
   */
590 public
double
 var_sig_norm_y =
1 ;
  /**
595  * Мода треугольного распределения по оси Y для агентов
   */
  public
double
 var_c_trian_y =
600 0 ;
  /**
   * Параметр сдвига распределения Лапласа по оси Y для взрыва
   */
  public
605 double
 var_b_lapl_expl_y =
0 ;
  /**
   * Параметр растяжения распределения Лапласа по оси Y для взрыва
610  */
  public
double
 var_a_lapl_expl_y =
1 ;
615 /**
   * Математическое ожидание нормального распределения по оси Y для взрыва
   */
  public
double
620 var_mu_norm_expl_y =
0 ;
  /**
   * Дисперсия нормального распределения по оси Y для взрыва
   */
625 public
double
 var_sig_norm_expl_y =
1 ;
  /**
630  * Мода треугольного распределения по оси Y для взрыва
   */
  public
double
 var_c_trian_expl_y =
635 0 ;
  /**
   * Текущая версия прогона модели
   */
  public
640 int
 var_version =
1 ;
  /**
   * Массив коэффициента близости к стене
645  */
  public
double[]
 var_wallCoef = new double[]
{1.2,1,0.8,0.7} ;
650 /**
```

```
    * Максимальная площадь горизонтальной проекции
    */
  public
double
 var_maxArea =
0.13 ;

  // Коллекции
  /**
    * Коллекция распределения агентов по клеткам<br>
    * индекс — номер клетки<br>
    * значение — количество агентов в клетке
    */
  public
java.util.ArrayList <
Integer > GlobalDistr_coll = new java.util.ArrayList<Integer>();
  /**
    * Коллекция текстовых полей для ручного распределения агентов по клеткам
    */
  public
java.util.ArrayList <
ShapeTextField > box_coll = new java.util.ArrayList<ShapeTextField>();
  /**
    * Коллекция подписей текстовых полей для ручного распределения агентов по клеткам
    */
  public
java.util.ArrayList <
ShapeText > text_coll = new java.util.ArrayList<ShapeText>();

  // Функции

  /**
    * Генерация интерфейса ручного распределения агентов по клеткам
    */
  void GlobalDistr( int count ) {

ShapeText text_n; // подпись к текстовому полю
ShapeTextField box_n; // текстовое поле
box_coll.clear();
text_coll.clear();
int i;
int default_value;
for (i=0; i<count;i++) {
  if (iter>1) {
    try {
      default_value=GlobalDistr_coll.get(i);
    }
    catch (IndexOutOfBoundsException ex ) {
      default_value=0;
    }
    if (default_value>var_agents_count) {
      default_value=0;
    }
  } else {
    GlobalDistr_coll.clear();
    DefaultDistr();
    default_value=GlobalDistr_coll.get(i);
  }
  box_n = new ShapeTextField( Simulation.this, true, 80.0+150*Math.floor(i/10), -1550.0+70*(i % 10), 40.0, 20.0, controlDefault,
      controlDefault, new Font("Dialog", Font.PLAIN, 12), ShapeControl.ValueType.TYPE_INT, 0, var_agents_count);
  box_n.setText(default_value);
  box_coll.add(i, box_n);
  text_n = new ShapeText(SHAPE_DRAW_2D, true, 40.0+150*Math.floor(i/10), -1545.0+70*(i % 10), 0.0, 0.0, black, "N_"+Integer.toString(i+1),
       new Font("Arial", Font.BOLD, 12) , ALIGNMENT_LEFT);
  text_coll.add(i, text_n);
  box_coll.get(i).setVisible(true);
  text_coll.get(i).setVisible(true);
  presentation.add(box_coll.get(i));
  presentation.add(text_coll.get(i));
}
  }

  /**
    * Функция определение максимальной области видимости монитора
    */

Rectangle
 MaxRectangle(  ) {

GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
GraphicsDevice gd = ge.getDefaultScreenDevice();
GraphicsConfiguration gc = gd.getDefaultConfiguration();
```

```
Insets ins = java.awt.Toolkit.getDefaultToolkit().getScreenInsets(gc);

Rectangle maxRect = gc.getBounds();
maxRect.setBounds(maxRect.x - ins.left, maxRect.y - ins.top, maxRect.width - ins.right, maxRect.height - ins.bottom);

return (maxRect);
  }

  /**
   * Равномерное распределение агентов по клеткам
   */
  void DefaultDistr( ) {

int i, j, h, k;
k = var_agents_count - (int) ((var_mvert*var_mhor)*Math.round(Math.floor(var_agents_count/(var_mvert*var_mhor)))); // количество агентов,
      оставшихся после равного распределения по клеткам
if (k>0) { // если остались не распределенные агенты
  j = (int) Math.round(Math.floor((k-1)/2));
  h = (int) Math.round((var_mvert*var_mhor)/2);
  if ((h-j)!=var_mvert*var_mhor-(h-j+k)+2) { // соотношение четности количества агентов и количества клеток
    for (i=0;i<h-j;i++) {
      GlobalDistr_coll.add(i, (int) Math.round(Math.floor(var_agents_count/(var_mvert*var_mhor))));
    }
    for (i=h-j;i<h-j+k;i++) {
      GlobalDistr_coll.add(i, 1+(int) Math.round(Math.floor(var_agents_count/(var_mvert*var_mhor))));
    }
    if (h-j+k<var_mvert*var_mhor) {
      for (i=h-j+k;i<var_mvert*var_mhor;i++) {
        GlobalDistr_coll.add(i, (int) Math.round(Math.floor(var_agents_count/(var_mvert*var_mhor))));
      }
    }
  } else {
    for (i=0;i<h-j-1;i++) {
      GlobalDistr_coll.add(i, (int) Math.round(Math.floor(var_agents_count/(var_mvert*var_mhor))));
    }
    for (i=h-j-1;i<h-j-1+k;i++) {
      GlobalDistr_coll.add(i, 1+(int) Math.round(Math.floor(var_agents_count/(var_mvert*var_mhor))));
    }
    if (h-j-1+k<var_mvert*var_mhor) {
      for (i=h-j-1+k;i<var_mvert*var_mhor;i++) {
        GlobalDistr_coll.add(i, (int) Math.round(Math.floor(var_agents_count/(var_mvert*var_mhor))));
      }
    }
  }
} else { // равное распределение по всем клеткам
  for (i=0;i<var_mvert*var_mhor;i++) {
    GlobalDistr_coll.add(i, (int) Math.round(Math.floor(var_agents_count/(var_mvert*var_mhor))));
  }
}
  }

  /**
   * Функция присвоения значения текущей версии прогона модели
   */
  void version( ) {

ResultSet rs = query_version.execute(); // исполнение запроса
rs.next(); // сдвиг на первую запись
int v = rs.getInt("Version")+1;
var_version=v;
box_version.setText(v, true);
  }

  /**
   * Функция удаления всех записей из базы данных для данной версии прогона модели
   */
  void delete_version( ) {

var_database.modify("DELETE FROM version WHERE version="+Integer.toString(var_version));
var_database.modify("DELETE FROM coordinates WHERE version="+Integer.toString(var_version));
var_database.modify("DELETE FROM functional WHERE version="+Integer.toString(var_version));
var_database.modify("DELETE FROM explosion WHERE version="+Integer.toString(var_version));
  }

  /**
   * Функция полной очистки базы данных
   */
  void delete( ) {

var_database.modify("DELETE FROM version");
var_database.modify("DELETE FROM coordinates");
var_database.modify("DELETE FROM functional");
```

```
var_database.modify("DELETE FROM explosion");
  }

  /**
   * Функция присвоения значения максимальной площади горизонтальной проекции
   */
  void maxArea(  ) {

ResultSet rs = query_maxArea.execute(); // исполнение запроса
rs.next(); // сдвиг на первую запись
double s = rs.getDouble("MAX");
var_maxArea=s;
  }
  // Области
  public ViewArea viewArea_base = new ViewArea( this, null, 0, 0, ViewArea.TOP_LEFT, ViewArea.NONE, 1.0, 1200, 800 );
  public ViewArea viewArea_global_distr = new ViewArea( this, null, 0, -1650, ViewArea.TOP_LEFT, ViewArea.NONE, 1.0, 850, 650 );
  @Override
  @AnyLogicInternalCodegenAPI
  public int getViewAreas(Map<String, ViewArea> _output) {
    if ( _output != null ) {
      _output.put( "viewArea_base", this.viewArea_base );
      _output.put( "viewArea_global_distr", this.viewArea_global_distr );
    }
    return 2 + super.getViewAreas( _output );
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void executeShapeControlAction( int _shape, int index ) {
    switch( _shape ) {
      case _button: {
          ShapeButton self = this.button;
delete_version(); // очистка базы данных для данной версии прогона
if ( getState() == IDLE )
  run();
  //pause();
getPresentation().setPresentable( getEngine().getRoot() ); // запуск прогона
;}
        break;
      case _button_global_distr: {
          ShapeButton self = this.button_global_distr;
iter++;
var_agents_count=box_agents_count.getIntValue();
box_agents_count_new.setText(var_agents_count);
GlobalDistr(var_mvert*var_mhor);
GlobalDistr_coll.clear();
rectangle_global_distr.setHeight(MaxRectangle().height);
rectangle_global_distr.setWidth(MaxRectangle().width);
viewArea_global_distr.navigateTo(); // переход к области задания распределения агентов в клетках
;}
        break;
      case _button_end: {
          ShapeButton self = this.button_end;
int i;
int sum = 0;
for (i=0;i<var_mvert*var_mhor;i++) { // расчет общего количества агентов
  sum = sum + box_coll.get(i).getIntValue();
}
if (sum!=var_agents_count) { // проверка общего количества агентов
  MessageBox.WarningBox("Сумма количества агентов в клетках не совпадает с общим числом агентов!", "Ошибка ввода");
} else {
  GlobalDistr_coll.clear();
  for (i=0;i<var_mvert*var_mhor;i++) { // присвоение количества агентов в клетке значениям коллекции
    GlobalDistr_coll.add(i, box_coll.get(i).getIntValue());
    presentation.remove(box_coll.get(i)); // удаление тектового поля
    presentation.remove(text_coll.get(i)); // удаление подписи тектового поля
  }
  box_coll.clear();
  text_coll.clear();
  viewArea_base.navigateTo(); // возвращение к начальной области презентации
}
;}
        break;
      case _button1: {
          ShapeButton self = this.button1;
delete(); // полная очистка базы данных
version(); // переприсвоение версии прогона
;}
        break;
      default:
        super.executeShapeControlAction( _shape, index );
        break;
```

```
    }
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void executeShapeControlAction( int _shape, int index, int value ) {
    switch( _shape ) {
      case _radio_global_distr: {
          ShapeRadioButtonGroup self = this.radio_global_distr;
if(radio_global_distr.getValue()==1) {
  var_global_distr=true;
  button_global_distr.setVisible(true);
} else {
  var_global_distr=false;
  button_global_distr.setVisible(false);
}
;}
        break;
      default:
        super.executeShapeControlAction( _shape, index, value );
        break;
    }
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void executeShapeControlAction( int _shape, int index, double value ) {
    switch( _shape ) {
      case _slider_agents_count:
        var_agents_count = (int) value; {
          ShapeSlider self = this.slider_agents_count;
var_agents_count=slider_agents_count.getIntValue();
box_agents_count.setText(slider_agents_count.getIntValue());
iter=0;
;}
        break;
      default:
        super.executeShapeControlAction( _shape, index, value );
        break;
    }
  }


  @Override
  @AnyLogicInternalCodegenAPI
  public void executeShapeControlAction( int _shape, int index, String value ) {
    switch( _shape ) {
      case _box_mvert:
        var_mvert = Integer.parseInt(value); {
          ShapeTextField self = this.box_mvert;
var_mvert=box_mvert.getIntValue();
iter=0;
;}
        break;
      case _box_mhor:
        var_mhor = Integer.parseInt(value); {
          ShapeTextField self = this.box_mhor;
var_mhor=box_mhor.getIntValue();
iter=0;
;}
        break;
      case _combobox:
        var_distr = value; {
          ShapeComboBox self = this.combobox;
if (combobox.getValue()=="Равномерное распределение") {
  text15.setVisible(false);
  box_c_trian_x.setVisible(false);
  text11.setVisible(false);
  box_b_lapl_x.setVisible(false);
  text12.setVisible(false);
  box_a_lapl_x.setVisible(false);
  text13.setVisible(false);
  box_mu_norm_x.setVisible(false);
  text14.setVisible(false);
  box_sig_norm_x.setVisible(false);
  text117.setVisible(false);
  box_c_trian_y.setVisible(false);
  text120.setVisible(false);
  box_b_lapl_y.setVisible(false);
  text121.setVisible(false);
  box_a_lapl_y.setVisible(false);
  text118.setVisible(false);
```

```
    box_mu_norm_y.setVisible(false);
    text119.setVisible(false);
    box_sig_norm_y.setVisible(false);
} else if (combobox.getValue()=="Нормальное распределение") {
    text15.setVisible(false);
    box_c_trian_x.setVisible(false);
    text11.setVisible(false);
    box_b_lapl_x.setVisible(false);
    text12.setVisible(false);
    box_a_lapl_x.setVisible(false);
    text13.setVisible(false);
    box_mu_norm_x.setVisible(false);
    text14.setVisible(false);
    box_sig_norm_x.setVisible(false);
    text117.setVisible(false);
    box_c_trian_y.setVisible(false);
    text120.setVisible(false);
    box_b_lapl_y.setVisible(false);
    text121.setVisible(false);
    box_a_lapl_y.setVisible(false);
    text118.setVisible(false);
    box_mu_norm_y.setVisible(false);
    text119.setVisible(false);
    box_sig_norm_y.setVisible(false);
    text13.setX(50);
    text13.setY(500);
    text13.setZ(0);
    box_mu_norm_x.setX(290);
    box_mu_norm_x.setY(495);
    text14.setX(50);
    text14.setY(530);
    text14.setZ(0);
    box_sig_norm_x.setX(290);
    box_sig_norm_x.setY(525);
    text118.setX(50);
    text118.setY(560);
    text118.setZ(0);
    box_mu_norm_y.setX(290);
    box_mu_norm_y.setY(555);
    text119.setX(50);
    text119.setY(590);
    text119.setZ(0);
    box_sig_norm_y.setX(290);
    box_sig_norm_y.setY(585);
    text13.setVisible(true);
    box_mu_norm_x.setVisible(true);
    text14.setVisible(true);
    box_sig_norm_x.setVisible(true);
    text118.setVisible(true);
    box_mu_norm_y.setVisible(true);
    text119.setVisible(true);
    box_sig_norm_y.setVisible(true);
} else if (combobox.getValue()=="Треугольное распределение") {
    text11.setVisible(false);
    box_b_lapl_x.setVisible(false);
    text12.setVisible(false);
    box_a_lapl_x.setVisible(false);
    text13.setVisible(false);
    box_mu_norm_x.setVisible(false);
    text14.setVisible(false);
    box_sig_norm_x.setVisible(false);
    text15.setVisible(false);
    box_c_trian_x.setVisible(false);
    text117.setVisible(false);
    box_c_trian_y.setVisible(false);
    text120.setVisible(false);
    box_b_lapl_y.setVisible(false);
    text121.setVisible(false);
    box_a_lapl_y.setVisible(false);
    text118.setVisible(false);
    box_mu_norm_y.setVisible(false);
    text119.setVisible(false);
    box_sig_norm_y.setVisible(false);
    text15.setX(50);
    text15.setY(500);
    text15.setZ(0);
    box_c_trian_x.setX(290);
    box_c_trian_x.setY(495);
    text117.setX(50);
    text117.setY(530);
    text117.setZ(0);
    box_c_trian_y.setX(290);
```

```
    box_c_trian_y.setY(525);
    text15.setVisible(true);
    box_c_trian_x.setVisible(true);
    text117.setVisible(true);
    box_c_trian_y.setVisible(true);
} else {
    text15.setVisible(false);
    box_c_trian_x.setVisible(false);
    text11.setVisible(false);
    box_b_lapl_x.setVisible(false);
    text12.setVisible(false);
    box_a_lapl_x.setVisible(false);
    text13.setVisible(false);
    box_mu_norm_x.setVisible(false);
    text14.setVisible(false);
    box_sig_norm_x.setVisible(false);
    text117.setVisible(false);
    box_c_trian_y.setVisible(false);
    text120.setVisible(false);
    box_b_lapl_y.setVisible(false);
    text121.setVisible(false);
    box_a_lapl_y.setVisible(false);
    text118.setVisible(false);
    box_mu_norm_y.setVisible(false);
    text119.setVisible(false);
    box_sig_norm_y.setVisible(false);
    text11.setX(50);
    text11.setY(500);
    text11.setZ(0);
    box_b_lapl_x.setX(290);
    box_b_lapl_x.setY(495);
    text12.setX(50);
    text12.setY(530);
    text12.setZ(0);
    box_a_lapl_x.setX(290);
    box_a_lapl_x.setY(525);
    text120.setX(50);
    text120.setY(560);
    text120.setZ(0);
    box_b_lapl_y.setX(290);
    box_b_lapl_y.setY(555);
    text121.setX(50);
    text121.setY(590);
    text121.setZ(0);
    box_a_lapl_y.setX(290);
    box_a_lapl_y.setY(585);
    text11.setVisible(true);
    box_b_lapl_x.setVisible(true);
    text12.setVisible(true);
    box_a_lapl_x.setVisible(true);
    text120.setVisible(true);
    box_b_lapl_y.setVisible(true);
    text121.setVisible(true);
    box_a_lapl_y.setVisible(true);
}
var_distr=combobox.getValue();
;}
        break;
      case _box_b_lapl_x:
        var_b_lapl_x = Double.parseDouble(value); {
          ShapeTextField self = this.box_b_lapl_x;
var_b_lapl_x=box_b_lapl_x.getDoubleValue();
;}
        break;
      case _box_a_lapl_x:
        var_a_lapl_x = Double.parseDouble(value); {
          ShapeTextField self = this.box_a_lapl_x;
var_a_lapl_x=box_a_lapl_x.getDoubleValue();
;}
        break;
      case _box_mu_norm_x:
        var_mu_norm_x = Double.parseDouble(value); {
          ShapeTextField self = this.box_mu_norm_x;
var_mu_norm_x=box_mu_norm_x.getDoubleValue();
;}
        break;
      case _box_sig_norm_x:
        var_sig_norm_x = Double.parseDouble(value); {
          ShapeTextField self = this.box_sig_norm_x;
var_sig_norm_x=box_sig_norm_x.getDoubleValue();
;}
        break;
```

```
      case _box_c_trian_x:
        var_c_trian_x = Double.parseDouble(value); {
          ShapeTextField self = this.box_c_trian_x;
var_c_trian_x=box_c_trian_x.getDoubleValue();
;}
        break;
      case _box_agents_count:
        var_agents_count = Integer.parseInt(value); {
          ShapeTextField self = this.box_agents_count;
var_agents_count=box_agents_count.getIntValue();
slider_agents_count.setValue(box_agents_count.getIntValue());
iter=0;
;}
        break;
      case _box_agents_count_new:
        var_agents_count = Integer.parseInt(value);
        break;
      case _box_sigma_1:
        var_sigma_1 = Double.parseDouble(value); {
          ShapeTextField self = this.box_sigma_1;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_sigma_1.getDoubleValue()<box_sigma_2.getDoubleValue()) {
  box_sigma_1.setText(box_sigma_2.getDoubleValue(), true);
}
var_sigma_1=box_sigma_1.getDoubleValue();
;}
        break;
      case _box_sigma_2:
        var_sigma_2 = Double.parseDouble(value); {
          ShapeTextField self = this.box_sigma_2;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_sigma_1.getDoubleValue()<box_sigma_2.getDoubleValue()) {
  box_sigma_1.setText(box_sigma_2.getDoubleValue(), true);
}
var_sigma_2=box_sigma_2.getDoubleValue();
;}
        break;
      case _box_eta:
        var_eta = Double.parseDouble(value); {
          ShapeTextField self = this.box_eta;
var_eta=box_eta.getDoubleValue();
;}
        break;
      case _box_varsigma_0:
        var_varsigma[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_varsigma_0;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varsigma_0.getDoubleValue()<1) {
  box_varsigma_0.setText(1, true);
}
if (box_varsigma_1.getDoubleValue()<box_varsigma_0.getDoubleValue()) {
  box_varsigma_1.setText(box_varsigma_0.getDoubleValue(), true);
}
if (box_varsigma_2.getDoubleValue()<box_varsigma_0.getDoubleValue()) {
  box_varsigma_2.setText(box_varsigma_0.getDoubleValue(), true);
}
if (box_varsigma_3.getDoubleValue()<box_varsigma_0.getDoubleValue()) {
  box_varsigma_3.setText(box_varsigma_0.getDoubleValue(), true);
}
var_varsigma[0]=box_varsigma_0.getDoubleValue();
;}
        break;
      case _box_varsigma_1:
        var_varsigma[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_varsigma_1;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varsigma_1.getDoubleValue()<1) {
  box_varsigma_1.setText(1, true);
}
if (box_varsigma_2.getDoubleValue()<box_varsigma_1.getDoubleValue()) {
  box_varsigma_2.setText(box_varsigma_1.getDoubleValue(), true);
}
if (box_varsigma_3.getDoubleValue()<box_varsigma_1.getDoubleValue()) {
  box_varsigma_3.setText(box_varsigma_1.getDoubleValue(), true);
}
if (box_varsigma_1.getDoubleValue()<box_varsigma_0.getDoubleValue()) {
  box_varsigma_1.setText(box_varsigma_0.getDoubleValue(), true);
}
var_varsigma[1]=box_varsigma_1.getDoubleValue();
;}
        break;
      case _box_varsigma_2:
```

```
        var_varsigma[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_varsigma_2;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varsigma_2.getDoubleValue()<1) {
  box_varsigma_2.setText(1, true);
}
if (box_varsigma_3.getDoubleValue()<box_varsigma_2.getDoubleValue()) {
  box_varsigma_3.setText(box_varsigma_2.getDoubleValue(), true);
}
if (box_varsigma_2.getDoubleValue()<box_varsigma_0.getDoubleValue()) {
  box_varsigma_2.setText(box_varsigma_0.getDoubleValue(), true);
}
if (box_varsigma_2.getDoubleValue()<box_varsigma_1.getDoubleValue()) {
  box_varsigma_2.setText(box_varsigma_1.getDoubleValue(), true);
}
var_varsigma[2]=box_varsigma_2.getDoubleValue();
;}
        break;
      case _box_varsigma_3:
        var_varsigma[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_varsigma_3;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varsigma_3.getDoubleValue()<1) {
  box_varsigma_3.setText(1, true);
}
if (box_varsigma_3.getDoubleValue()<box_varsigma_2.getDoubleValue()) {
  box_varsigma_3.setText(box_varsigma_2.getDoubleValue(), true);
}
if (box_varsigma_3.getDoubleValue()<box_varsigma_0.getDoubleValue()) {
  box_varsigma_3.setText(box_varsigma_0.getDoubleValue(), true);
}
if (box_varsigma_3.getDoubleValue()<box_varsigma_1.getDoubleValue()) {
  box_varsigma_3.setText(box_varsigma_1.getDoubleValue(), true);
}
var_varsigma[3]=box_varsigma_3.getDoubleValue();
;}
        break;
      case _box_varrho_3_0:
        var_varrho_3[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_varrho_3_0;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varrho_3_0.getDoubleValue()<0.00001) {
  box_varrho_3_0.setText(0.00001, true);
}
if (box_varrho_3_0.getDoubleValue()<box_varrho_3_1.getDoubleValue()) {
  box_varrho_3_0.setText(box_varrho_3_1.getDoubleValue(), true);
}
if (box_varrho_3_0.getDoubleValue()<box_varrho_3_2.getDoubleValue()) {
  box_varrho_3_0.setText(box_varrho_3_2.getDoubleValue(), true);
}
if (box_varrho_3_0.getDoubleValue()<box_varrho_3_3.getDoubleValue()) {
  box_varrho_3_0.setText(box_varrho_3_3.getDoubleValue(), true);
}
if (box_varrho_4_0.getDoubleValue()<box_varrho_3_0.getDoubleValue()) {
  box_varrho_4_0.setText(box_varrho_3_0.getDoubleValue(), true);
}
var_varrho_3[0]=box_varrho_3_0.getDoubleValue();
;}
        break;
      case _box_varrho_3_1:
        var_varrho_3[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_varrho_3_1;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varrho_3_1.getDoubleValue()<0.00001) {
  box_varrho_3_1.setText(0.00001, true);
}
if (box_varrho_3_0.getDoubleValue()<box_varrho_3_1.getDoubleValue()) {
  box_varrho_3_0.setText(box_varrho_3_1.getDoubleValue(), true);
}
if (box_varrho_3_1.getDoubleValue()<box_varrho_3_2.getDoubleValue()) {
  box_varrho_3_1.setText(box_varrho_3_2.getDoubleValue(), true);
}
if (box_varrho_3_1.getDoubleValue()<box_varrho_3_3.getDoubleValue()) {
  box_varrho_3_1.setText(box_varrho_3_3.getDoubleValue(), true);
}
if (box_varrho_4_1.getDoubleValue()<box_varrho_3_1.getDoubleValue()) {
  box_varrho_4_1.setText(box_varrho_3_1.getDoubleValue(), true);
}
var_varrho_3[1]=box_varrho_3_1.getDoubleValue();
;}
        break;
      case _box_varrho_3_2:
```

```
        var_varrho_3[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_varrho_3_2;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varrho_3_2.getDoubleValue()<0.00001) {
  box_varrho_3_2.setText(0.00001, true);
}
if (box_varrho_3_1.getDoubleValue()<box_varrho_3_2.getDoubleValue()) {
  box_varrho_3_1.setText(box_varrho_3_2.getDoubleValue(), true);
}
if (box_varrho_3_0.getDoubleValue()<box_varrho_3_2.getDoubleValue()) {
  box_varrho_3_0.setText(box_varrho_3_2.getDoubleValue(), true);
}
if (box_varrho_3_2.getDoubleValue()<box_varrho_3_3.getDoubleValue()) {
  box_varrho_3_2.setText(box_varrho_3_3.getDoubleValue(), true);
}
if (box_varrho_4_2.getDoubleValue()<box_varrho_3_2.getDoubleValue()) {
  box_varrho_4_2.setText(box_varrho_3_2.getDoubleValue(), true);
}
var_varrho_3[2]=box_varrho_3_2.getDoubleValue();
;}
        break;
      case _box_varrho_3_3:
        var_varrho_3[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_varrho_3_3;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varrho_3_3.getDoubleValue()<0.00001) {
  box_varrho_3_3.setText(0.00001, true);
}
if (box_varrho_3_2.getDoubleValue()<box_varrho_3_3.getDoubleValue()) {
  box_varrho_3_2.setText(box_varrho_3_3.getDoubleValue(), true);
}
if (box_varrho_3_1.getDoubleValue()<box_varrho_3_3.getDoubleValue()) {
  box_varrho_3_1.setText(box_varrho_3_3.getDoubleValue(), true);
}
if (box_varrho_3_0.getDoubleValue()<box_varrho_3_3.getDoubleValue()) {
  box_varrho_3_0.setText(box_varrho_3_3.getDoubleValue(), true);
}
if (box_varrho_4_3.getDoubleValue()<box_varrho_3_3.getDoubleValue()) {
  box_varrho_4_3.setText(box_varrho_3_3.getDoubleValue(), true);
}
var_varrho_3[3]=box_varrho_3_3.getDoubleValue();
;}
        break;
      case _box_varrho_4_0:
        var_varrho_4[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_varrho_4_0;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varrho_4_0.getDoubleValue()<0.00001) {
  box_varrho_4_0.setText(0.00001, true);
}
if (box_varrho_4_1.getDoubleValue()<box_varrho_4_0.getDoubleValue()) {
  box_varrho_4_1.setText(box_varrho_4_0.getDoubleValue(), true);
}
if (box_varrho_4_2.getDoubleValue()<box_varrho_4_0.getDoubleValue()) {
  box_varrho_4_2.setText(box_varrho_4_0.getDoubleValue(), true);
}
if (box_varrho_4_3.getDoubleValue()<box_varrho_4_0.getDoubleValue()) {
  box_varrho_4_3.setText(box_varrho_4_0.getDoubleValue(), true);
}
if (box_varrho_4_0.getDoubleValue()<box_varrho_3_0.getDoubleValue()) {
  box_varrho_4_0.setText(box_varrho_3_0.getDoubleValue(), true);
}
var_varrho_4[0]=box_varrho_4_0.getDoubleValue();
;}
        break;
      case _box_varrho_4_1:
        var_varrho_4[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_varrho_4_1;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varrho_4_1.getDoubleValue()<0.00001) {
  box_varrho_4_1.setText(0.00001, true);
}
if (box_varrho_4_2.getDoubleValue()<box_varrho_4_1.getDoubleValue()) {
  box_varrho_4_2.setText(box_varrho_4_1.getDoubleValue(), true);
}
if (box_varrho_4_3.getDoubleValue()<box_varrho_4_1.getDoubleValue()) {
  box_varrho_4_3.setText(box_varrho_4_1.getDoubleValue(), true);
}
if (box_varrho_4_1.getDoubleValue()<box_varrho_4_0.getDoubleValue()) {
  box_varrho_4_1.setText(box_varrho_4_0.getDoubleValue(), true);
}
if (box_varrho_4_1.getDoubleValue()<box_varrho_3_1.getDoubleValue()) {
```

```
  box_varrho_4_1.setText(box_varrho_3_1.getDoubleValue(), true);
}
var_varrho_4[1]=box_varrho_4_1.getDoubleValue();
;}
        break;
      case _box_varrho_4_2:
        var_varrho_4[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_varrho_4_2;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varrho_4_2.getDoubleValue()<0.00001) {
  box_varrho_4_2.setText(0.00001, true);
}
if (box_varrho_4_3.getDoubleValue()<box_varrho_4_2.getDoubleValue()) {
  box_varrho_4_3.setText(box_varrho_4_2.getDoubleValue(), true);
}
if (box_varrho_4_2.getDoubleValue()<box_varrho_4_1.getDoubleValue()) {
  box_varrho_4_2.setText(box_varrho_4_1.getDoubleValue(), true);
}
if (box_varrho_4_2.getDoubleValue()<box_varrho_4_0.getDoubleValue()) {
  box_varrho_4_2.setText(box_varrho_4_0.getDoubleValue(), true);
}
if (box_varrho_4_2.getDoubleValue()<box_varrho_3_2.getDoubleValue()) {
  box_varrho_4_2.setText(box_varrho_3_2.getDoubleValue(), true);
}
var_varrho_4[2]=box_varrho_4_2.getDoubleValue();
;}
        break;
      case _box_varrho_4_3:
        var_varrho_4[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_varrho_4_3;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varrho_4_3.getDoubleValue()<0.00001) {
  box_varrho_4_3.setText(0.00001, true);
}
if (box_varrho_4_3.getDoubleValue()<box_varrho_4_2.getDoubleValue()) {
  box_varrho_4_3.setText(box_varrho_4_2.getDoubleValue(), true);
}
if (box_varrho_4_3.getDoubleValue()<box_varrho_4_1.getDoubleValue()) {
  box_varrho_4_3.setText(box_varrho_4_1.getDoubleValue(), true);
}
if (box_varrho_4_3.getDoubleValue()<box_varrho_4_0.getDoubleValue()) {
  box_varrho_4_3.setText(box_varrho_4_0.getDoubleValue(), true);
}
if (box_varrho_4_3.getDoubleValue()<box_varrho_3_3.getDoubleValue()) {
  box_varrho_4_3.setText(box_varrho_3_3.getDoubleValue(), true);
}
var_varrho_4[3]=box_varrho_4_3.getDoubleValue();
;}
        break;
      case _box_vartheta_1:
        var_vartheta[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_vartheta_1;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_vartheta_1.getDoubleValue()<0.7) {
  box_vartheta_1.setText(0.7, true);
}
if (box_vartheta_4.getDoubleValue()<box_vartheta_1.getDoubleValue()) {
  box_vartheta_4.setText(box_vartheta_1.getDoubleValue(), true);
}
if (box_vartheta_1.getDoubleValue()<box_vartheta_2.getDoubleValue()) {
  box_vartheta_1.setText(box_vartheta_2.getDoubleValue(), true);
}
if (box_vartheta_1.getDoubleValue()<box_vartheta_3.getDoubleValue()) {
  box_vartheta_1.setText(box_vartheta_3.getDoubleValue(), true);
}
var_vartheta[0]=box_vartheta_1.getDoubleValue();
;}
        break;
      case _box_vartheta_2:
        var_vartheta[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_vartheta_2;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_vartheta_2.getDoubleValue()<0.7) {
  box_vartheta_2.setText(0.7, true);
}
if (box_vartheta_1.getDoubleValue()<box_vartheta_2.getDoubleValue()) {
  box_vartheta_1.setText(box_vartheta_2.getDoubleValue(), true);
}
if (box_vartheta_4.getDoubleValue()<box_vartheta_2.getDoubleValue()) {
  box_vartheta_4.setText(box_vartheta_2.getDoubleValue(), true);
}
if (box_vartheta_2.getDoubleValue()<box_vartheta_3.getDoubleValue()) {
```

```
  box_vartheta_2.setText(box_vartheta_3.getDoubleValue(), true);
}
var_vartheta[1]=box_vartheta_2.getDoubleValue();
;}
        break;
      case _box_vartheta_3:
        var_vartheta[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_vartheta_3;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_vartheta_3.getDoubleValue()<0.7) {
  box_vartheta_3.setText(0.7, true);
}
if (box_vartheta_2.getDoubleValue()<box_vartheta_3.getDoubleValue()) {
  box_vartheta_2.setText(box_vartheta_3.getDoubleValue(), true);
}
if (box_vartheta_1.getDoubleValue()<box_vartheta_3.getDoubleValue()) {
  box_vartheta_1.setText(box_vartheta_3.getDoubleValue(), true);
}
if (box_vartheta_4.getDoubleValue()<box_vartheta_3.getDoubleValue()) {
  box_vartheta_4.setText(box_vartheta_3.getDoubleValue(), true);
}
var_vartheta[2]=box_vartheta_3.getDoubleValue();
;}
        break;
      case _box_vartheta_4:
        var_vartheta[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_vartheta_4;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_vartheta_4.getDoubleValue()<0.7) {
  box_vartheta_4.setText(0.7, true);
}
if (box_vartheta_4.getDoubleValue()<box_vartheta_1.getDoubleValue()) {
  box_vartheta_4.setText(box_vartheta_1.getDoubleValue(), true);
}
if (box_vartheta_4.getDoubleValue()<box_vartheta_2.getDoubleValue()) {
  box_vartheta_4.setText(box_vartheta_2.getDoubleValue(), true);
}
if (box_vartheta_4.getDoubleValue()<box_vartheta_3.getDoubleValue()) {
  box_vartheta_4.setText(box_vartheta_3.getDoubleValue(), true);
}
var_vartheta[3]=box_vartheta_4.getDoubleValue();
;}
        break;
      case _combo_theta_0:
        var_theta[0] = Integer.parseInt(value);
        break;
      case _combo_theta_1:
        var_theta[1] = Integer.parseInt(value); {
          ShapeComboBox self = this.combo_theta_1;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (combo_theta_1.getIntValue()<combo_theta_2.getIntValue()) {
  combo_theta_1.setValue(combo_theta_2.getIntValue(), true);
}
if (combo_theta_1.getIntValue()<combo_theta_3.getIntValue()) {
  combo_theta_1.setValue(combo_theta_3.getIntValue(), true);
}
var_theta[1]=combo_theta_1.getIntValue();
;}
        break;
      case _combo_theta_2:
        var_theta[2] = Integer.parseInt(value); {
          ShapeComboBox self = this.combo_theta_2;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (combo_theta_1.getIntValue()<combo_theta_2.getIntValue()) {
  combo_theta_1.setValue(combo_theta_2.getIntValue(), true);
}
if (combo_theta_2.getIntValue()<combo_theta_3.getIntValue()) {
  combo_theta_2.setValue(combo_theta_3.getIntValue(), true);
}
var_theta[2]=combo_theta_2.getIntValue();
;}
        break;
      case _combo_theta_3:
        var_theta[3] = Integer.parseInt(value); {
          ShapeComboBox self = this.combo_theta_3;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (combo_theta_1.getIntValue()<combo_theta_3.getIntValue()) {
  combo_theta_1.setValue(combo_theta_3.getIntValue(), true);
}
if (combo_theta_2.getIntValue()<combo_theta_3.getIntValue()) {
  combo_theta_2.setValue(combo_theta_3.getIntValue(), true);
}
```

```
var_theta[3]=combo_theta_3.getIntValue();
;}
        break;
      case _combo_theta_hat_0:
        var_theta_hat[0] = Integer.parseInt(value);
        break;
      case _combo_theta_hat_1:
        var_theta_hat[1] = Integer.parseInt(value); {
          ShapeComboBox self = this.combo_theta_hat_1;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (combo_theta_hat_1.getIntValue()<combo_theta_hat_2.getIntValue()) {
  combo_theta_hat_1.setValue(combo_theta_hat_2.getIntValue(), true);
}
if (combo_theta_hat_1.getIntValue()<combo_theta_hat_3.getIntValue()) {
  combo_theta_hat_1.setValue(combo_theta_hat_3.getIntValue(), true);
}
var_theta_hat[1]=combo_theta_hat_1.getIntValue();
;}
        break;
      case _combo_theta_hat_2:
        var_theta_hat[2] = Integer.parseInt(value); {
          ShapeComboBox self = this.combo_theta_hat_2;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (combo_theta_hat_1.getIntValue()<combo_theta_hat_2.getIntValue()) {
  combo_theta_hat_1.setValue(combo_theta_hat_2.getIntValue(), true);
}
if (combo_theta_hat_2.getIntValue()<combo_theta_hat_3.getIntValue()) {
  combo_theta_hat_2.setValue(combo_theta_hat_3.getIntValue(), true);
}
var_theta_hat[2]=combo_theta_hat_2.getIntValue();
;}
        break;
      case _combo_theta_hat_3:
        var_theta_hat[3] = Integer.parseInt(value); {
          ShapeComboBox self = this.combo_theta_hat_3;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (combo_theta_hat_1.getIntValue()<combo_theta_hat_3.getIntValue()) {
  combo_theta_hat_1.setValue(combo_theta_hat_3.getIntValue(), true);
}
if (combo_theta_hat_2.getIntValue()<combo_theta_hat_3.getIntValue()) {
  combo_theta_hat_2.setValue(combo_theta_hat_3.getIntValue(), true);
}
var_theta_hat[3]=combo_theta_hat_3.getIntValue();
;}
        break;
      case _box_gamma_acc:
        var_gamma_acc = Double.parseDouble(value); {
          ShapeTextField self = this.box_gamma_acc;
var_gamma_acc=box_gamma_acc.getDoubleValue();
;}
        break;
      case _box_gamma_dec:
        var_gamma_dec = Double.parseDouble(value); {
          ShapeTextField self = this.box_gamma_dec;
var_gamma_dec=box_gamma_dec.getDoubleValue();
;}
        break;
      case _box_gamma_exit:
        var_gamma_exit = Double.parseDouble(value); {
          ShapeTextField self = this.box_gamma_exit;
var_gamma_exit=box_gamma_exit.getDoubleValue();
;}
        break;
      case _box_varepsilon_0:
        var_varepsilon_0 = Double.parseDouble(value); {
          ShapeTextField self = this.box_varepsilon_0;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varepsilon_1.getDoubleValue()<box_varepsilon_0.getDoubleValue()) {
  box_varepsilon_1.setText(box_varepsilon_0.getDoubleValue(), true);
}
if (box_varepsilon_2.getDoubleValue()<box_varepsilon_0.getDoubleValue()) {
  box_varepsilon_2.setText(box_varepsilon_0.getDoubleValue(), true);
}
var_varepsilon_0=box_varepsilon_0.getDoubleValue();
;}
        break;
      case _box_varepsilon_1:
        var_varepsilon_1 = Double.parseDouble(value); {
          ShapeTextField self = this.box_varepsilon_1;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varepsilon_1.getDoubleValue()<box_varepsilon_0.getDoubleValue()) {
  box_varepsilon_1.setText(box_varepsilon_0.getDoubleValue(), true);
```

```
}
if (box_varepsilon_2.getDoubleValue()<box_varepsilon_1.getDoubleValue()) {
  box_varepsilon_2.setText(box_varepsilon_1.getDoubleValue(), true);
}
var_varepsilon_1=box_varepsilon_1.getDoubleValue();
;}
        break;
      case _box_varepsilon_2:
        var_varepsilon_2 = Double.parseDouble(value); {
          ShapeTextField self = this.box_varepsilon_2;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_varepsilon_2.getDoubleValue()<box_varepsilon_0.getDoubleValue()) {
  box_varepsilon_2.setText(box_varepsilon_0.getDoubleValue(), true);
}
if (box_varepsilon_2.getDoubleValue()<box_varepsilon_1.getDoubleValue()) {
  box_varepsilon_2.setText(box_varepsilon_1.getDoubleValue(), true);
}
var_varepsilon_2=box_varepsilon_2.getDoubleValue();
;}
        break;
      case _box_zeta:
        var_zeta = Integer.parseInt(value); {
          ShapeTextField self = this.box_zeta;
var_zeta=box_zeta.getIntValue();
;}
        break;
      case _box_kappa_1:
        var_kappa_1 = Double.parseDouble(value); {
          ShapeTextField self = this.box_kappa_1;
var_kappa_1=box_kappa_1.getDoubleValue();
;}
        break;
      case _box_kappa_2:
        var_kappa_2 = Double.parseDouble(value); {
          ShapeTextField self = this.box_kappa_2;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_kappa_3.getDoubleValue()<box_kappa_2.getDoubleValue()) {
  box_kappa_3.setText(box_kappa_2.getDoubleValue(), true);
}
var_kappa_2=box_kappa_2.getDoubleValue();
;}
        break;
      case _box_kappa_3:
        var_kappa_3 = Double.parseDouble(value); {
          ShapeTextField self = this.box_kappa_3;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_kappa_3.getDoubleValue()<box_kappa_2.getDoubleValue()) {
  box_kappa_3.setText(box_kappa_2.getDoubleValue(), true);
}
var_kappa_3=box_kappa_3.getDoubleValue();
;}
        break;
      case _box_beta_occ_0:
        var_beta_occ[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_occ_0;
if (box_beta_occ_0.getDoubleValue()<0) {
  box_beta_occ_0.setText(0, true);
}
var_beta_occ[0]=box_beta_occ_0.getDoubleValue();
;}
        break;
      case _box_beta_dir_0:
        var_beta_dir[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_dir_0;
if (box_beta_dir_0.getDoubleValue()<0) {
  box_beta_dir_0.setText(0, true);
}
var_beta_dir[0]=box_beta_dir_0.getDoubleValue();
;}
        break;
      case _box_beta_dest_0:
        var_beta_dest[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_dest_0;
if (box_beta_dest_0.getDoubleValue()<0) {
  box_beta_dest_0.setText(0, true);
}
var_beta_dest[0]=box_beta_dest_0.getDoubleValue();
;}
        break;
      case _box_beta_angle_0:
        var_beta_angle[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_angle_0;
```

```
if (box_beta_angle_0.getDoubleValue()<0) {
  box_beta_angle_0.setText(0, true);
}
var_beta_angle[0]=box_beta_angle_0.getDoubleValue();
;}
        break;
      case _box_beta_wall_0:
        var_beta_wall[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_wall_0;
if (box_beta_wall_0.getDoubleValue()<0) {
  box_beta_wall_0.setText(0, true);
}
var_beta_wall[0]=box_beta_wall_0.getDoubleValue();
;}
        break;
      case _box_beta_vel_0:
        var_beta_vel[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_vel_0;
if (box_beta_vel_0.getDoubleValue()<0) {
  box_beta_vel_0.setText(0, true);
}
var_beta_vel[0]=box_beta_vel_0.getDoubleValue();
;}
        break;
      case _box_beta_max_0:
        var_beta_max[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_max_0;
if (box_beta_max_0.getDoubleValue()<0) {
  box_beta_max_0.setText(0, true);
}
var_beta_max[0]=box_beta_max_0.getDoubleValue();
;}
        break;
      case _box_beta_expl_0:
        var_beta_expl[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_expl_0;
if (box_beta_expl_0.getDoubleValue()<0) {
  box_beta_expl_0.setText(0, true);
}
var_beta_expl[0]=box_beta_expl_0.getDoubleValue();
;}
        break;
      case _box_gamma_1:
        var_gamma_1 = Double.parseDouble(value); {
          ShapeTextField self = this.box_gamma_1;
var_gamma_1=box_gamma_1.getDoubleValue();
;}
        break;
      case _box_gamma_2:
        var_gamma_2 = Double.parseDouble(value); {
          ShapeTextField self = this.box_gamma_2;
var_gamma_2=box_gamma_2.getDoubleValue();
;}
        break;
      case _box_gamma_3:
        var_gamma_3 = Double.parseDouble(value); {
          ShapeTextField self = this.box_gamma_3;
var_gamma_3=box_gamma_3.getDoubleValue();
;}
        break;
      case _box_gamma_4:
        var_gamma_4 = Double.parseDouble(value); {
          ShapeTextField self = this.box_gamma_4;
var_gamma_4=box_gamma_4.getDoubleValue();
;}
        break;
      case _box_lambda_vel:
        var_lambda_vel = Double.parseDouble(value); {
          ShapeTextField self = this.box_lambda_vel;
var_lambda_vel=box_lambda_vel.getDoubleValue();
;}
        break;
      case _box_lambda_max:
        var_lambda_max = Double.parseDouble(value); {
          ShapeTextField self = this.box_lambda_max;
var_lambda_max=box_lambda_max.getDoubleValue();
;}
        break;
      case _combo_expl:
        var_distr_expl = value; {
          ShapeComboBox self = this.combo_expl;
if (combo_expl.getValue()=="Равномерное распределение") {
```

```
    text82.setVisible(false);
    box_c_trian_expl_x.setVisible(false);
    text78.setVisible(false);
    box_b_lapl_expl_x.setVisible(false);
    text79.setVisible(false);
    box_a_lapl_expl_x.setVisible(false);
    text80.setVisible(false);
    box_mu_norm_expl_x.setVisible(false);
    text81.setVisible(false);
    box_sig_norm_expl_x.setVisible(false);
    text122.setVisible(false);
    box_c_trian_expl_y.setVisible(false);
    text125.setVisible(false);
    box_b_lapl_expl_y.setVisible(false);
    text126.setVisible(false);
    box_a_lapl_expl_y.setVisible(false);
    text123.setVisible(false);
    box_mu_norm_expl_y.setVisible(false);
    text124.setVisible(false);
    box_sig_norm_expl_y.setVisible(false);
  } else if (combo_expl.getValue()=="Нормальное распределение") {
    text82.setVisible(false);
    box_c_trian_expl_x.setVisible(false);
    text78.setVisible(false);
    box_b_lapl_expl_x.setVisible(false);
    text79.setVisible(false);
    box_a_lapl_expl_x.setVisible(false);
    text80.setVisible(false);
    box_mu_norm_expl_x.setVisible(false);
    text81.setVisible(false);
    box_sig_norm_expl_x.setVisible(false);
    text122.setVisible(false);
    box_c_trian_expl_y.setVisible(false);
    text125.setVisible(false);
    box_b_lapl_expl_y.setVisible(false);
    text126.setVisible(false);
    box_a_lapl_expl_y.setVisible(false);
    text123.setVisible(false);
    box_mu_norm_expl_y.setVisible(false);
    text124.setVisible(false);
    box_sig_norm_expl_y.setVisible(false);
    text80.setX(50);
    text80.setY(670);
    text80.setZ(0);
    box_mu_norm_expl_x.setX(290);
    box_mu_norm_expl_x.setY(665);
    text81.setX(50);
    text81.setY(700);
    text81.setZ(0);
    box_sig_norm_expl_x.setX(290);
    box_sig_norm_expl_x.setY(695);
    text123.setX(50);
    text123.setY(730);
    text123.setZ(0);
    box_mu_norm_expl_y.setX(290);
    box_mu_norm_expl_y.setY(725);
    text124.setX(50);
    text124.setY(760);
    text124.setZ(0);
    box_sig_norm_expl_y.setX(290);
    box_sig_norm_expl_y.setY(755);
    text80.setVisible(true);
    box_mu_norm_expl_x.setVisible(true);
    text81.setVisible(true);
    box_sig_norm_expl_x.setVisible(true);
    text123.setVisible(true);
    box_mu_norm_expl_y.setVisible(true);
    text124.setVisible(true);
    box_sig_norm_expl_y.setVisible(true);
  } else if (combo_expl.getValue()=="Треугольное распределение") {
    text78.setVisible(false);
    box_b_lapl_expl_x.setVisible(false);
    text79.setVisible(false);
    box_a_lapl_expl_x.setVisible(false);
    text80.setVisible(false);
    box_mu_norm_expl_x.setVisible(false);
    text81.setVisible(false);
    box_sig_norm_expl_x.setVisible(false);
    text82.setVisible(false);
    box_c_trian_expl_x.setVisible(false);
    text122.setVisible(false);
    box_c_trian_expl_y.setVisible(false);
```

```
    text125.setVisible(false);
    box_b_lapl_expl_y.setVisible(false);
    text126.setVisible(false);
    box_a_lapl_expl_y.setVisible(false);
    text123.setVisible(false);
    box_mu_norm_expl_y.setVisible(false);
    text124.setVisible(false);
    box_sig_norm_expl_y.setVisible(false);
    text82.setX(50);
    text82.setY(670);
    text82.setZ(0);
    box_c_trian_expl_x.setX(290);
    box_c_trian_expl_x.setY(665);
    text122.setX(50);
    text122.setY(700);
    text122.setZ(0);
    box_c_trian_expl_y.setX(290);
    box_c_trian_expl_y.setY(695);
    text82.setVisible(true);
    box_c_trian_expl_x.setVisible(true);
    text122.setVisible(true);
    box_c_trian_expl_y.setVisible(true);
} else {
    text82.setVisible(false);
    box_c_trian_expl_x.setVisible(false);
    text78.setVisible(false);
    box_b_lapl_expl_x.setVisible(false);
    text79.setVisible(false);
    box_a_lapl_expl_x.setVisible(false);
    text80.setVisible(false);
    box_mu_norm_expl_x.setVisible(false);
    text81.setVisible(false);
    box_sig_norm_expl_x.setVisible(false);
    text122.setVisible(false);
    box_c_trian_expl_y.setVisible(false);
    text125.setVisible(false);
    box_b_lapl_expl_y.setVisible(false);
    text126.setVisible(false);
    box_a_lapl_expl_y.setVisible(false);
    text123.setVisible(false);
    box_mu_norm_expl_y.setVisible(false);
    text124.setVisible(false);
    box_sig_norm_expl_y.setVisible(false);
    text78.setX(50);
    text78.setY(670);
    text78.setZ(0);
    box_b_lapl_expl_x.setX(290);
    box_b_lapl_expl_x.setY(665);
    text79.setX(50);
    text79.setY(700);
    text79.setZ(0);
    box_a_lapl_expl_x.setX(290);
    box_a_lapl_expl_x.setY(695);
    text125.setX(50);
    text125.setY(730);
    text125.setZ(0);
    box_b_lapl_expl_y.setX(290);
    box_b_lapl_expl_y.setY(725);
    text126.setX(50);
    text126.setY(760);
    text126.setZ(0);
    box_a_lapl_expl_y.setX(290);
    box_a_lapl_expl_y.setY(755);
    text78.setVisible(true);
    box_b_lapl_expl_x.setVisible(true);
    text79.setVisible(true);
    box_a_lapl_expl_x.setVisible(true);
    text125.setVisible(true);
    box_b_lapl_expl_y.setVisible(true);
    text126.setVisible(true);
    box_a_lapl_expl_y.setVisible(true);
}
var_distr_expl=combo_expl.getValue();
;}
        break;
      case _box_b_lapl_expl_x:
        var_b_lapl_expl_x = Double.parseDouble(value); {
          ShapeTextField self = this.box_b_lapl_expl_x;
var_b_lapl_expl_x=box_b_lapl_expl_x.getDoubleValue();
;}
        break;
      case _box_a_lapl_expl_x:
```

```
        var_a_lapl_expl_x = Double.parseDouble(value); {
          ShapeTextField self = this.box_a_lapl_expl_x;
var_a_lapl_expl_x=box_a_lapl_expl_x.getDoubleValue();
;}
        break;
      case _box_mu_norm_expl_x:
        var_mu_norm_expl_x = Double.parseDouble(value); {
          ShapeTextField self = this.box_mu_norm_expl_x;
var_mu_norm_expl_x=box_mu_norm_expl_x.getDoubleValue();
;}
        break;
      case _box_sig_norm_expl_x:
        var_sig_norm_expl_x = Double.parseDouble(value); {
          ShapeTextField self = this.box_sig_norm_expl_x;
var_sig_norm_expl_x=box_sig_norm_expl_x.getDoubleValue();
;}
        break;
      case _box_c_trian_expl_x:
        var_c_trian_expl_x = Double.parseDouble(value); {
          ShapeTextField self = this.box_c_trian_expl_x;
var_c_trian_expl_x=box_c_trian_expl_x.getDoubleValue();
;}
        break;
      case _combo_turb:
        var_distr_turb = value; {
          ShapeComboBox self = this.combo_turb;
if (combo_turb.getValue()=="Равномерное распределение") {
  text85.setVisible(false);
  box_turb.setVisible(false);
} else {
  text85.setVisible(false);
  box_turb.setVisible(false);
  text85.setX(50);
  text85.setY(840);
  text85.setZ(0);
  box_turb.setX(290);
  box_turb.setY(835);
  text85.setVisible(true);
  box_turb.setVisible(true);
}
var_distr_turb=combo_turb.getValue();
;}
        break;
      case _combo_exit:
        var_distr_exit = value; {
          ShapeComboBox self = this.combo_exit;
var_distr_exit=combo_exit.getValue();
;}
        break;
      case _box_turb:
        var_turb = Integer.parseInt(value); {
          ShapeTextField self = this.box_turb;
var_turb=box_turb.getIntValue();
;}
        break;
      case _box_gamma_5:
        var_gamma_5 = Double.parseDouble(value); {
          ShapeTextField self = this.box_gamma_5;
var_gamma_5=box_gamma_5.getDoubleValue();
;}
        break;
      case _box_lambda_avg:
        var_lambda_avg = Double.parseDouble(value); {
          ShapeTextField self = this.box_lambda_avg;
var_lambda_avg=box_lambda_avg.getDoubleValue();
;}
        break;
      case _box_beta_avg_0:
        var_beta_avg[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_avg_0;
if (box_beta_avg_0.getDoubleValue()<0) {
  box_beta_avg_0.setText(0, true);
}
var_beta_avg[0]=box_beta_avg_0.getDoubleValue();
;}
        break;
      case _box_beta_occ_1:
        var_beta_occ[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_occ_1;
if (box_beta_occ_1.getDoubleValue()<0) {
  box_beta_occ_1.setText(0, true);
}
```

```
var_beta_occ[1]=box_beta_occ_1.getDoubleValue();
;}
        break;
      case _box_beta_dir_1:
        var_beta_dir[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_dir_1;
if (box_beta_dir_1.getDoubleValue()<0) {
  box_beta_dir_1.setText(0, true);
}
var_beta_dir[1]=box_beta_dir_1.getDoubleValue();
;}
        break;
      case _box_beta_dest_1:
        var_beta_dest[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_dest_1;
if (box_beta_dest_1.getDoubleValue()<0) {
  box_beta_dest_1.setText(0, true);
}
var_beta_dest[1]=box_beta_dest_1.getDoubleValue();
;}
        break;
      case _box_beta_angle_1:
        var_beta_angle[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_angle_1;
if (box_beta_angle_1.getDoubleValue()<0) {
  box_beta_angle_1.setText(0, true);
}
var_beta_angle[1]=box_beta_angle_1.getDoubleValue();
;}
        break;
      case _box_beta_wall_1:
        var_beta_wall[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_wall_1;
if (box_beta_wall_1.getDoubleValue()<0) {
  box_beta_wall_1.setText(0, true);
}
var_beta_wall[1]=box_beta_wall_1.getDoubleValue();
;}
        break;
      case _box_beta_vel_1:
        var_beta_vel[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_vel_1;
if (box_beta_vel_1.getDoubleValue()<0) {
  box_beta_vel_1.setText(0, true);
}
var_beta_vel[1]=box_beta_vel_1.getDoubleValue();
;}
        break;
      case _box_beta_max_1:
        var_beta_max[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_max_1;
if (box_beta_max_1.getDoubleValue()<0) {
  box_beta_max_1.setText(0, true);
}
var_beta_max[1]=box_beta_max_1.getDoubleValue();
;}
        break;
      case _box_beta_expl_1:
        var_beta_expl[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_expl_1;
if (box_beta_expl_1.getDoubleValue()<0) {
  box_beta_expl_1.setText(0, true);
}
var_beta_expl[1]=box_beta_expl_1.getDoubleValue();
;}
        break;
      case _box_beta_avg_1:
        var_beta_avg[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_avg_1;
if (box_beta_avg_1.getDoubleValue()<0) {
  box_beta_avg_1.setText(0, true);
}
var_beta_avg[1]=box_beta_avg_1.getDoubleValue();
;}
        break;
      case _box_beta_occ_2:
        var_beta_occ[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_occ_2;
if (box_beta_occ_2.getDoubleValue()<0) {
  box_beta_occ_2.setText(0, true);
}
var_beta_occ[2]=box_beta_occ_2.getDoubleValue();
```

```
;}
        break;
      case _box_beta_dir_2:
        var_beta_dir[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_dir_2;
if (box_beta_dir_2.getDoubleValue()<0) {
  box_beta_dir_2.setText(0, true);
}
var_beta_dir[2]=box_beta_dir_2.getDoubleValue();
;}
        break;
      case _box_beta_dest_2:
        var_beta_dest[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_dest_2;
if (box_beta_dest_2.getDoubleValue()<0) {
  box_beta_dest_2.setText(0, true);
}
var_beta_dest[2]=box_beta_dest_2.getDoubleValue();
;}
        break;
      case _box_beta_angle_2:
        var_beta_angle[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_angle_2;
if (box_beta_angle_2.getDoubleValue()<0) {
  box_beta_angle_2.setText(0, true);
}
var_beta_angle[2]=box_beta_angle_2.getDoubleValue();
;}
        break;
      case _box_beta_wall_2:
        var_beta_wall[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_wall_2;
if (box_beta_wall_2.getDoubleValue()<0) {
  box_beta_wall_2.setText(0, true);
}
var_beta_wall[2]=box_beta_wall_2.getDoubleValue();
;}
        break;
      case _box_beta_vel_2:
        var_beta_vel[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_vel_2;
if (box_beta_vel_2.getDoubleValue()<0) {
  box_beta_vel_2.setText(0, true);
}
var_beta_vel[2]=box_beta_vel_2.getDoubleValue();
;}
        break;
      case _box_beta_max_2:
        var_beta_max[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_max_2;
if (box_beta_max_2.getDoubleValue()<0) {
  box_beta_max_2.setText(0, true);
}
var_beta_max[2]=box_beta_max_2.getDoubleValue();
;}
        break;
      case _box_beta_expl_2:
        var_beta_expl[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_expl_2;
if (box_beta_expl_2.getDoubleValue()<0) {
  box_beta_expl_2.setText(0, true);
}
var_beta_expl[2]=box_beta_expl_2.getDoubleValue();
;}
        break;
      case _box_beta_avg_2:
        var_beta_avg[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_avg_2;
if (box_beta_avg_2.getDoubleValue()<0) {
  box_beta_avg_2.setText(0, true);
}
var_beta_avg[2]=box_beta_avg_2.getDoubleValue();
;}
        break;
      case _box_beta_occ_3:
        var_beta_occ[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_occ_3;
if (box_beta_occ_3.getDoubleValue()<0) {
  box_beta_occ_3.setText(0, true);
}
var_beta_occ[3]=box_beta_occ_3.getDoubleValue();
;}
```

```
        break;
      case _box_beta_dir_3:
        var_beta_dir[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_dir_3;
if (box_beta_dir_3.getDoubleValue()<0) {
  box_beta_dir_3.setText(0, true);
}
var_beta_dir[3]=box_beta_dir_3.getDoubleValue();
;}
        break;
      case _box_beta_dest_3:
        var_beta_dest[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_dest_3;
if (box_beta_dest_3.getDoubleValue()<0) {
  box_beta_dest_3.setText(0, true);
}
var_beta_dest[3]=box_beta_dest_3.getDoubleValue();
;}
        break;
      case _box_beta_angle_3:
        var_beta_angle[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_angle_3;
if (box_beta_angle_3.getDoubleValue()<0) {
  box_beta_angle_3.setText(0, true);
}
var_beta_angle[3]=box_beta_angle_3.getDoubleValue();
;}
        break;
      case _box_beta_wall_3:
        var_beta_wall[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_wall_3;
if (box_beta_wall_3.getDoubleValue()<0) {
  box_beta_wall_3.setText(0, true);
}
var_beta_wall[3]=box_beta_wall_3.getDoubleValue();
;}
        break;
      case _box_beta_vel_3:
        var_beta_vel[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_vel_3;
if (box_beta_vel_3.getDoubleValue()<0) {
  box_beta_vel_3.setText(0, true);
}
var_beta_vel[3]=box_beta_vel_3.getDoubleValue();
;}
        break;
      case _box_beta_max_3:
        var_beta_max[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_max_3;
if (box_beta_max_3.getDoubleValue()<0) {
  box_beta_max_3.setText(0, true);
}
var_beta_max[3]=box_beta_max_3.getDoubleValue();
;}
        break;
      case _box_beta_expl_3:
        var_beta_expl[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_expl_3;
if (box_beta_expl_3.getDoubleValue()<0) {
  box_beta_expl_3.setText(0, true);
}
var_beta_expl[3]=box_beta_expl_3.getDoubleValue();
;}
        break;
      case _box_beta_avg_3:
        var_beta_avg[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_beta_avg_3;
if (box_beta_avg_3.getDoubleValue()<0) {
  box_beta_avg_3.setText(0, true);
}
var_beta_avg[3]=box_beta_avg_3.getDoubleValue();
;}
        break;
      case _box_c_trian_y:
        var_c_trian_y = Double.parseDouble(value); {
          ShapeTextField self = this.box_c_trian_y;
var_c_trian_y=box_c_trian_y.getDoubleValue();
;}
        break;
      case _box_mu_norm_y:
        var_mu_norm_y = Double.parseDouble(value); {
          ShapeTextField self = this.box_mu_norm_y;
```

```
var_mu_norm_y=box_mu_norm_y.getDoubleValue();
;}
        break;
      case _box_sig_norm_y:
        var_sig_norm_y = Double.parseDouble(value); {
          ShapeTextField self = this.box_sig_norm_y;
var_sig_norm_y=box_sig_norm_y.getDoubleValue();
;}
        break;
      case _box_b_lapl_y:
        var_b_lapl_y = Double.parseDouble(value); {
          ShapeTextField self = this.box_b_lapl_y;
var_b_lapl_y=box_b_lapl_y.getDoubleValue();
;}
        break;
      case _box_a_lapl_y:
        var_a_lapl_y = Double.parseDouble(value); {
          ShapeTextField self = this.box_a_lapl_y;
var_a_lapl_y=box_a_lapl_y.getDoubleValue();
;}
        break;
      case _box_c_trian_expl_y:
        var_c_trian_expl_y = Double.parseDouble(value); {
          ShapeTextField self = this.box_c_trian_expl_y;
var_c_trian_expl_y=box_c_trian_expl_y.getDoubleValue();
;}
        break;
      case _box_mu_norm_expl_y:
        var_mu_norm_expl_y = Double.parseDouble(value); {
          ShapeTextField self = this.box_mu_norm_expl_y;
var_mu_norm_expl_y=box_mu_norm_expl_y.getDoubleValue();
;}
        break;
      case _box_sig_norm_expl_y:
        var_sig_norm_expl_y = Double.parseDouble(value); {
          ShapeTextField self = this.box_sig_norm_expl_y;
var_sig_norm_expl_y=box_sig_norm_expl_y.getDoubleValue();
;}
        break;
      case _box_b_lapl_expl_y:
        var_b_lapl_expl_y = Double.parseDouble(value); {
          ShapeTextField self = this.box_b_lapl_expl_y;
var_b_lapl_expl_y=box_b_lapl_expl_y.getDoubleValue();
;}
        break;
      case _box_a_lapl_expl_y:
        var_a_lapl_expl_y = Double.parseDouble(value); {
          ShapeTextField self = this.box_a_lapl_expl_y;
var_a_lapl_expl_y=box_a_lapl_expl_y.getDoubleValue();
;}
        break;
      case _box_version:
        var_version = Integer.parseInt(value); {
          ShapeTextField self = this.box_version;
var_version=box_version.getIntValue();
;}
        break;
      case _box_wallCoef_1:
        var_wallCoef[1] = Double.parseDouble(value); {
          ShapeTextField self = this.box_wallCoef_1;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_wallCoef_0.getDoubleValue()<box_wallCoef_1.getDoubleValue()) {
  box_wallCoef_0.setText(box_wallCoef_1.getDoubleValue(), true);
}
if (box_wallCoef_1.getDoubleValue()<box_wallCoef_2.getDoubleValue()) {
  box_wallCoef_1.setText(box_wallCoef_2.getDoubleValue(), true);
}
if (box_wallCoef_1.getDoubleValue()<box_wallCoef_3.getDoubleValue()) {
  box_wallCoef_1.setText(box_wallCoef_3.getDoubleValue(), true);
}
var_wallCoef[1]=box_wallCoef_1.getDoubleValue();
;}
        break;
      case _box_wallCoef_2:
        var_wallCoef[2] = Double.parseDouble(value); {
          ShapeTextField self = this.box_wallCoef_2;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_wallCoef_0.getDoubleValue()<box_wallCoef_2.getDoubleValue()) {
  box_wallCoef_0.setText(box_wallCoef_2.getDoubleValue(), true);
}
if (box_wallCoef_1.getDoubleValue()<box_wallCoef_2.getDoubleValue()) {
  box_wallCoef_1.setText(box_wallCoef_2.getDoubleValue(), true);
```

```
}
if (box_wallCoef_2.getDoubleValue()<box_wallCoef_3.getDoubleValue()) {
  box_wallCoef_2.setText(box_wallCoef_3.getDoubleValue(), true);
}
var_wallCoef[2]=box_wallCoef_2.getDoubleValue();
;}
        break;
      case _box_wallCoef_3:
        var_wallCoef[3] = Double.parseDouble(value); {
          ShapeTextField self = this.box_wallCoef_3;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_wallCoef_0.getDoubleValue()<box_wallCoef_3.getDoubleValue()) {
  box_wallCoef_0.setText(box_wallCoef_3.getDoubleValue(), true);
}
if (box_wallCoef_1.getDoubleValue()<box_wallCoef_3.getDoubleValue()) {
  box_wallCoef_1.setText(box_wallCoef_3.getDoubleValue(), true);
}
if (box_wallCoef_2.getDoubleValue()<box_wallCoef_3.getDoubleValue()) {
  box_wallCoef_2.setText(box_wallCoef_3.getDoubleValue(), true);
}
var_wallCoef[3]=box_wallCoef_3.getDoubleValue();
;}
        break;
      case _box_wallCoef_0:
        var_wallCoef[0] = Double.parseDouble(value); {
          ShapeTextField self = this.box_wallCoef_0;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_wallCoef_0.getDoubleValue()<box_wallCoef_1.getDoubleValue()) {
  box_wallCoef_0.setText(box_wallCoef_1.getDoubleValue(), true);
}
if (box_wallCoef_0.getDoubleValue()<box_wallCoef_2.getDoubleValue()) {
  box_wallCoef_0.setText(box_wallCoef_2.getDoubleValue(), true);
}
if (box_wallCoef_0.getDoubleValue()<box_wallCoef_3.getDoubleValue()) {
  box_wallCoef_0.setText(box_wallCoef_3.getDoubleValue(), true);
}
var_wallCoef[0]=box_wallCoef_0.getDoubleValue();
;}
        break;
      case _box_len1:
        var_len1 = Double.parseDouble(value); {
          ShapeTextField self = this.box_len1;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
box_a21.setText(box_a0.getDoubleValue()+box_len1.getDoubleValue(), true);
box_a22.setText(box_a0.getDoubleValue()+box_len1.getDoubleValue(), true);
var_len1=box_len1.getDoubleValue();
;}
        break;
      case _box_len2:
        var_len2 = Double.parseDouble(value); {
          ShapeTextField self = this.box_len2;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_b11.getDoubleValue()-box_b0.getDoubleValue()>=box_len2.getDoubleValue()) {
box_b11.setText(box_b0.getDoubleValue(), true);
}
if (box_b12.getDoubleValue()-box_b0.getDoubleValue()>=box_len2.getDoubleValue()) {
box_b12.setText(box_b0.getDoubleValue()+box_len2.getDoubleValue(),true);
}
if (box_b21.getDoubleValue()-box_b0.getDoubleValue()>=box_len2.getDoubleValue()) {
box_b21.setText(box_b0.getDoubleValue(), true);
}
if (box_b22.getDoubleValue()-box_b0.getDoubleValue()>=box_len2.getDoubleValue()) {
box_b22.setText(box_b0.getDoubleValue()+box_len2.getDoubleValue(), true);
}
var_len2=box_len2.getDoubleValue();
;}
        break;
      case _box_a0:
        var_a0 = Double.parseDouble(value); {
          ShapeTextField self = this.box_a0;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
box_a11.setText(box_a0.getDoubleValue(), true);
box_a12.setText(box_a0.getDoubleValue(), true);
box_a21.setText(box_a0.getDoubleValue()+box_len1.getDoubleValue(), true);
box_a22.setText(box_a0.getDoubleValue()+box_len1.getDoubleValue(), true);
var_a0=box_a0.getDoubleValue();
;}
        break;
      case _box_b0:
        var_b0 = Double.parseDouble(value); {
          ShapeTextField self = this.box_b0;
var_b0=box_b0.getDoubleValue();
```

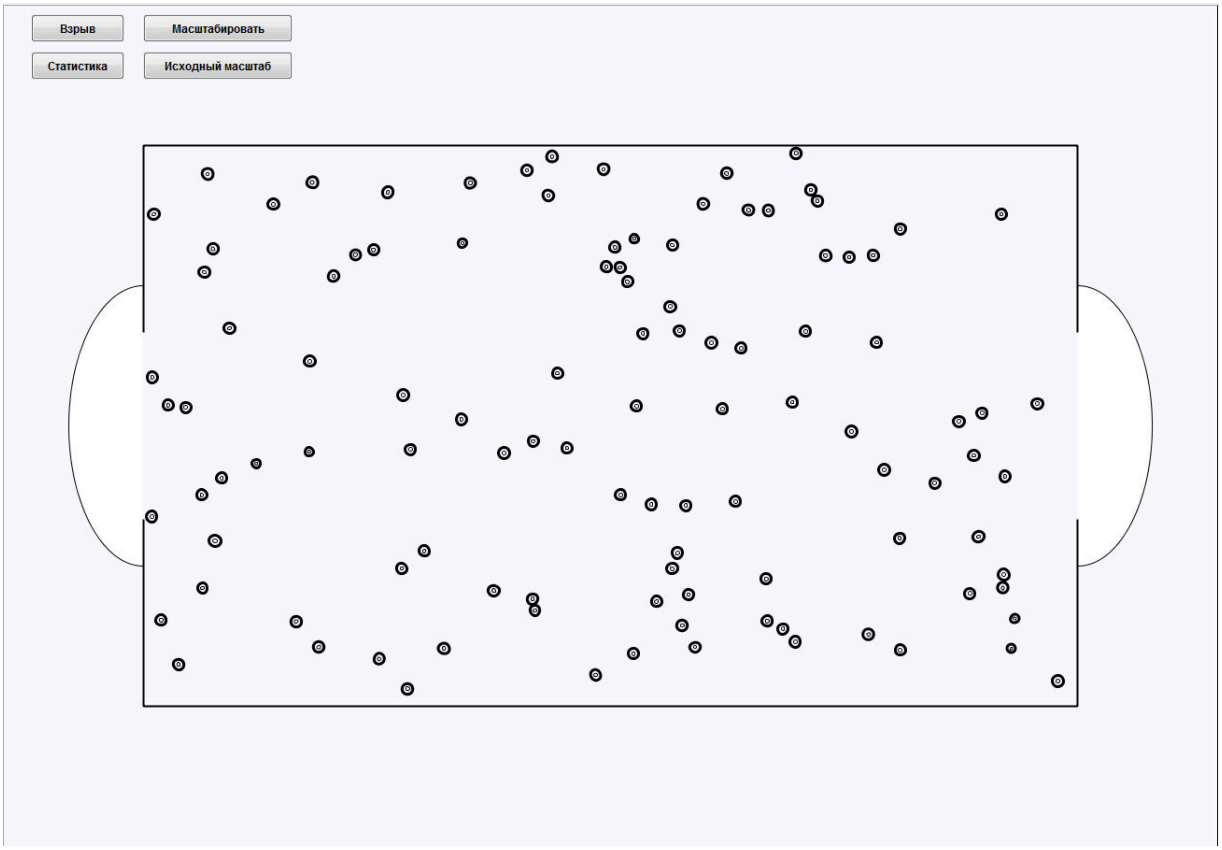
```
;}
        break;
      case _box_a11:
        var_a11 = Double.parseDouble(value); {
          ShapeTextField self = this.box_a11;
var_a11=box_a11.getDoubleValue();
;}
        break;
      case _box_b11:
        var_b11 = Double.parseDouble(value); {
          ShapeTextField self = this.box_b11;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_b11.getDoubleValue()<box_b0.getDoubleValue()) {
box_b11.setText(box_b0.getDoubleValue());
}
if (box_b11.getDoubleValue()>box_b12.getDoubleValue()) {
box_b11.setText(box_b0.getDoubleValue());
}
var_b11=box_b11.getDoubleValue();
;}
        break;
      case _box_a12:
        var_a12 = Double.parseDouble(value); {
          ShapeTextField self = this.box_a12;
var_a12=box_a12.getDoubleValue();
;}
        break;
      case _box_b12:
        var_b12 = Double.parseDouble(value); {
          ShapeTextField self = this.box_b12;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_b12.getDoubleValue()<box_b11.getDoubleValue()) {
box_b12.setText((box_b11.getDoubleValue()+box_b0.getDoubleValue()+box_len2.getDoubleValue())/2);
}
if (box_b12.getDoubleValue()>box_b0.getDoubleValue()+box_len2.getDoubleValue()) {
box_b12.setText(box_b0.getDoubleValue()+box_len2.getDoubleValue());
}
var_b12=box_b12.getDoubleValue();
;}
        break;
      case _box_a21:
        var_a21 = Double.parseDouble(value); {
          ShapeTextField self = this.box_a21;
var_a21=box_a21.getDoubleValue();
;}
        break;
      case _box_b21:
        var_b21 = Double.parseDouble(value); {
          ShapeTextField self = this.box_b21;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_b21.getDoubleValue()<box_b0.getDoubleValue()) {
box_b21.setText(box_b0.getDoubleValue());
}
if (box_b21.getDoubleValue()>box_b12.getDoubleValue()) {
box_b21.setText(box_b0.getDoubleValue());
}
var_b21=box_b21.getDoubleValue();
;}
        break;
      case _box_a22:
        var_a22 = Double.parseDouble(value); {
          ShapeTextField self = this.box_a22;
var_a22=box_a22.getDoubleValue();
;}
        break;
      case _box_b22:
        var_b22 = Double.parseDouble(value); {
          ShapeTextField self = this.box_b22;
// каскадное обновление значений других текстовых полей для сохранения логики соотношений между ними
if (box_b22.getDoubleValue()<box_b21.getDoubleValue()) {
box_b22.setText((box_b21.getDoubleValue()+box_b0.getDoubleValue()+box_len2.getDoubleValue())/2);
}
if (box_b22.getDoubleValue()>box_b0.getDoubleValue()+box_len2.getDoubleValue()) {
box_b22.setText(box_b0.getDoubleValue()+box_len2.getDoubleValue());
}
var_b22=box_b22.getDoubleValue();
;}
        break;
      default:
        super.executeShapeControlAction( _shape, index, value );
        break;
    }
```

```
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public double getShapeControlMinimum( int _shape, int index ) {

  }

  @Override
  @AnyLogicInternalCodegenAPI
  public double getShapeControlMaximum( int _shape, int index ) {

  }

  @Override
  @AnyLogicInternalCodegenAPI
  public int getShapeControlDefaultValueInt( int _shape, int index ) {
    switch(_shape) {
      case _radio_global_distr: return
0
;
      default: return super.getShapeControlDefaultValueInt( _shape, index );
    }
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public double getShapeControlDefaultValueDouble( int _shape, int index ) {
    switch(_shape) {
      case _slider_agents_count: return
var_agents_count
;
      default: return super.getShapeControlDefaultValueDouble( _shape, index );
    }
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public String getShapeControlDefaultValueString( int _shape, int index ) {

  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void onDestroy_xjal() {
    onDestroy();
    // Уничтожение объектов баз данных
    var_database.destroy();
    super.onDestroy_xjal();
  }
version(); // назначение версии прогона
maxArea(); // назначение максимальной площади горизонтальной проекции
viewArea_base.navigateTo(); // переход к области презентации
  }

}
```

Приложение В

Основной класс модели

В.1 Скрин основного класса



В.2 Листинг кода основного класса

```
package crowd;

import java.io.Serializable;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
```

```
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.LinkedHashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.PriorityQueue;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.markup.Network;
import com.xj.anylogic.engine.Position;
import com.xj.anylogic.engine.markup.PedFlowStatistics;
import com.xj.anylogic.engine.markup.DensityMap;

import static java.lang.Math.*;
import static com.xj.anylogic.engine.UtilitiesArray.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;

import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.markup.*;
import com.xj.anylogic.engine.presentation.*;

import static crowd.Gender.*;
import static crowd.Status.*;
import static crowd.Situation.*;
import static crowd.Exit.*;

import java.awt.geom.Arc2D;

import java.awt.Frame;
import java.awt.geom.Area;
import com.sun.media.sound.Toolkit;
import java.awt.GraphicsEnvironment;
import java.awt.GraphicsDevice;
import java.awt.Rectangle;
import java.awt.Insets;
import java.awt.GraphicsConfiguration;
import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import com.sun.org.apache.xpath.internal.operations.Variable;

public class Main extends Agent
{
  // Базы данных
  public Database database = new Database( this, "database", Database.DRIVER_JDBC_MICROSOFT_SQLSERVER, "ARMEN", "CROWD", "sa", new char[]
      { 't', 'a', 't', 'a', 'm', 'o', 'l', 'o', 'd', 'e', 'c' } );

  // Объекты Insert
  public Insert insert_version = new Insert( "insert_version",
database
, "version", new String[] {
    "version",
    "agents_count",
    "a0",
    "b0",
    "len1",
    "len2",
    "a11",
    "b11",
```

```
    "a12",
    "b12",
    "a21",
    "b21",
    "a22",
    "b22",
    "turb",
    "sigma_1",
    "sigma_2",
    "eta",
    "gamma_acc",
    "gamma_dec",
    "gamma_exit",
    "varsigma_0",
    "varsigma_1",
    "varsigma_2",
    "varsigma_3",
    "varrho_3_0",
    "varrho_3_1",
    "varrho_3_2",
    "varrho_3_3",
    "varrho_4_0",
    "varrho_4_1",
    "varrho_4_2",
    "varrho_4_3",
    "vartheta_1",
    "vartheta_2",
    "vartheta_3",
    "vartheta_4",
    "theta_0",
    "theta_1",
    "theta_2",
    "theta_3",
    "theta_hat_0",
    "theta_hat_1",
    "theta_hat_2",
    "theta_hat_3",
    "zeta",
    "kappa_1",
    "kappa_2",
    "kappa_3",
    "date",
}, new String[] {
    "version",
    "agents_count",
    "a0",
    "b0",
    "len1",
    "len2",
    "a11",
    "b11",
    "a12",
    "b12",
    "a21",
    "b21",
    "a22",
    "b22",
    "turb",
    "sigma_1",
    "sigma_2",
    "eta",
    "gamma_acc",
    "gamma_dec",
    "gamma_exit",
    "varsigma[0]",
    "varsigma[1]",
    "varsigma[2]",
    "varsigma[3]",
    "varrho_3[0]",
    "varrho_3[1]",
    "varrho_3[2]",
    "varrho_3[3]",
    "varrho_4[0]",
    "varrho_4[1]",
    "varrho_4[2]",
    "varrho_4[3]",
    "vartheta[0]",
    "vartheta[1]",
    "vartheta[2]",
    "vartheta[3]",
    "theta[0]",
    "theta[1]",
```

```
    "theta[2]",
    "theta[3]",
    "theta_hat[0]",
    "theta_hat[1]",
    "theta_hat[2]",
    "theta_hat[3]",
    "zeta",
    "kappa_1",
    "kappa_2",
    "kappa_3",
    "getDateTime()",
  } ) {
    @Override
    public void evaluateValues_xjal(Object[] values) {
      _insert_version_evaluateValues(values);
    }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = -9043181999061322049L;
  };
  @AnyLogicInternalCodegenAPI
  private void _insert_version_evaluateValues(Object[] values) {
    values[ 0 ] = version
;
    values[ 1 ] = agents_count
;
    values[ 2 ] = a0
;
    values[ 3 ] = b0
;
    values[ 4 ] = len1
;
    values[ 5 ] = len2
;
    values[ 6 ] = a11
;
    values[ 7 ] = b11
;
    values[ 8 ] = a12
;
    values[ 9 ] = b12
;
    values[ 10 ] = a21
;
    values[ 11 ] = b21
;
    values[ 12 ] = a22
;
    values[ 13 ] = b22
;
    values[ 14 ] = turb
;
    values[ 15 ] = sigma_1
;
    values[ 16 ] = sigma_2
;
    values[ 17 ] = eta
;
    values[ 18 ] = gamma_acc
;
    values[ 19 ] = gamma_dec
;
    values[ 20 ] = gamma_exit
;
    values[ 21 ] = varsigma[0]
;
    values[ 22 ] = varsigma[1]
;
    values[ 23 ] = varsigma[2]
;
    values[ 24 ] = varsigma[3]
;
    values[ 25 ] = varrho_3[0]
;
    values[ 26 ] = varrho_3[1]
;
    values[ 27 ] = varrho_3[2]
;
    values[ 28 ] = varrho_3[3]
;
```

```
    values[ 29 ] = varrho_4[0]
;
    values[ 30 ] = varrho_4[1]
;
    values[ 31 ] = varrho_4[2]
;
    values[ 32 ] = varrho_4[3]
;
    values[ 33 ] = vartheta[0]
;
    values[ 34 ] = vartheta[1]
;
    values[ 35 ] = vartheta[2]
;
    values[ 36 ] = vartheta[3]
;
    values[ 37 ] = theta[0]
;
    values[ 38 ] = theta[1]
;
    values[ 39 ] = theta[2]
;
    values[ 40 ] = theta[3]
;
    values[ 41 ] = theta_hat[0]
;
    values[ 42 ] = theta_hat[1]
;
    values[ 43 ] = theta_hat[2]
;
    values[ 44 ] = theta_hat[3]
;
    values[ 45 ] = zeta
;
    values[ 46 ] = kappa_1
;
    values[ 47 ] = kappa_2
;
    values[ 48 ] = kappa_3
;
    values[ 49 ] = getDateTime()
;
  }
  public Insert insert_functional = new Insert( "insert_functional",
database
, "functional", new String[] {
    "version",
    "gamma_1",
    "gamma_2",
    "gamma_3",
    "gamma_4",
    "gamma_5",
    "lambda_vel",
    "lambda_max",
    "lambda_avg",
    "beta_occ_0",
    "beta_occ_1",
    "beta_occ_2",
    "beta_occ_3",
    "beta_dir_0",
    "beta_dir_1",
    "beta_dir_2",
    "beta_dir_3",
    "beta_dest_0",
    "beta_dest_1",
    "beta_dest_2",
    "beta_dest_3",
    "beta_angle_0",
    "beta_angle_1",
    "beta_angle_2",
    "beta_angle_3",
    "beta_wall_0",
    "beta_wall_1",
    "beta_wall_2",
    "beta_wall_3",
    "beta_vel_0",
    "beta_vel_1",
    "beta_vel_2",
    "beta_vel_3",
    "beta_max_0",
    "beta_max_1",
    "beta_max_2",
```

```
    "beta_max_3",
    "beta_avg_0",
    "beta_avg_1",
    "beta_avg_2",
    "beta_avg_3",
    "beta_expl_0",
    "beta_expl_1",
    "beta_expl_2",
    "beta_expl_3",
  }, new String[] {
    "version",
    "gamma_1",
    "gamma_2",
    "gamma_3",
    "gamma_4",
    "gamma_5",
    "lambda_vel",
    "lambda_max",
    "lambda_avg",
    "beta_occ[0]",
    "beta_occ[1]",
    "beta_occ[2]",
    "beta_occ[3]",
    "beta_dir[0]",
    "beta_dir[1]",
    "beta_dir[2]",
    "beta_dir[3]",
    "beta_dest[0]",
    "beta_dest[1]",
    "beta_dest[2]",
    "beta_dest[3]",
    "beta_angle[0]",
    "beta_angle[1]",
    "beta_angle[2]",
    "beta_angle[3]",
    "beta_wall[0]",
    "beta_wall[1]",
    "beta_wall[2]",
    "beta_wall[3]",
    "beta_vel[0]",
    "beta_vel[1]",
    "beta_vel[2]",
    "beta_vel[3]",
    "beta_max[0]",
    "beta_max[1]",
    "beta_max[2]",
    "beta_max[3]",
    "beta_avg[0]",
    "beta_avg[0]",
    "beta_avg[2]",
    "beta_avg[3]",
    "beta_expl[0]",
    "beta_expl[1]",
    "beta_expl[2]",
    "beta_expl[3]",
  } ) {
    @Override
    public void evaluateValues_xjal(Object[] values) {
      _insert_functional_evaluateValues(values);
    }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = -9043181999040355670L;
  };
  @AnyLogicInternalCodegenAPI
  private void _insert_functional_evaluateValues(Object[] values) {
    values[ 0 ] = version
;
    values[ 1 ] = gamma_1
;
    values[ 2 ] = gamma_2
;
    values[ 3 ] = gamma_3
;
    values[ 4 ] = gamma_4
;
    values[ 5 ] = gamma_5
;
    values[ 6 ] = lambda_vel
;
```

```
        values[ 7 ] = lambda_max
    ;
        values[ 8 ] = lambda_avg
    ;
        values[ 9 ] = beta_occ[0]
    ;
        values[ 10 ] = beta_occ[1]
    ;
        values[ 11 ] = beta_occ[2]
    ;
        values[ 12 ] = beta_occ[3]
    ;
        values[ 13 ] = beta_dir[0]
    ;
        values[ 14 ] = beta_dir[1]
    ;
        values[ 15 ] = beta_dir[2]
    ;
        values[ 16 ] = beta_dir[3]
    ;
        values[ 17 ] = beta_dest[0]
    ;
        values[ 18 ] = beta_dest[1]
    ;
        values[ 19 ] = beta_dest[2]
    ;
        values[ 20 ] = beta_dest[3]
    ;
        values[ 21 ] = beta_angle[0]
    ;
        values[ 22 ] = beta_angle[1]
    ;
        values[ 23 ] = beta_angle[2]
    ;
        values[ 24 ] = beta_angle[3]
    ;
        values[ 25 ] = beta_wall[0]
    ;
        values[ 26 ] = beta_wall[1]
    ;
        values[ 27 ] = beta_wall[2]
    ;
        values[ 28 ] = beta_wall[3]
    ;
        values[ 29 ] = beta_vel[0]
    ;
        values[ 30 ] = beta_vel[1]
    ;
        values[ 31 ] = beta_vel[2]
    ;
        values[ 32 ] = beta_vel[3]
    ;
        values[ 33 ] = beta_max[0]
    ;
        values[ 34 ] = beta_max[1]
    ;
        values[ 35 ] = beta_max[2]
    ;
        values[ 36 ] = beta_max[3]
    ;
        values[ 37 ] = beta_avg[0]
    ;
        values[ 38 ] = beta_avg[0]
    ;
        values[ 39 ] = beta_avg[2]
    ;
        values[ 40 ] = beta_avg[3]
    ;
        values[ 41 ] = beta_expl[0]
    ;
        values[ 42 ] = beta_expl[1]
    ;
        values[ 43 ] = beta_expl[2]
    ;
        values[ 44 ] = beta_expl[3]
    ;
    }

    // Параметры

    /**
     * Длина помещения
```

```
     */
    public
double  len1;

    /**
     * Возвращает значение по умолчанию параметра <code>len1</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    @AnyLogicInternalCodegenAPI
    public double _len1_DefaultValue_xjal() {
      final Main self = this;
      return 0.0;
    }

    public void set_len1( double len1 ) {
      if (len1 == this.len1) {
        return;
      }
      double _oldValue_xjal = this.len1;
      this.len1 = len1;
      onChange_len1_xjal( _oldValue_xjal );
      onChange();
    }

    /**
     * Calls "On change" action for parameter len1.<br>
     * Note that 'oldValue' in that action will be unavailable if this method is called by user
     * (current parameter value will be passed as 'oldValue').<br>
     * Please call <code>set_len1()</code> method instead.
     */
    void onChange_len1() {
      onChange_len1_xjal( len1 );
    }

    @AnyLogicInternalCodegenAPI
    void onChange_len1_xjal( double oldValue ) {
    }


    /**
     * Ширина помещения
     */
    public
double  len2;

    /**
     * Возвращает значение по умолчанию параметра <code>len2</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    @AnyLogicInternalCodegenAPI
    public double _len2_DefaultValue_xjal() {
      final Main self = this;
      return 0.0;
    }

    public void set_len2( double len2 ) {
      if (len2 == this.len2) {
        return;
      }
      double _oldValue_xjal = this.len2;
      this.len2 = len2;
      onChange_len2_xjal( _oldValue_xjal );
      onChange();
    }

    /**
     * Calls "On change" action for parameter len2.<br>
     * Note that 'oldValue' in that action will be unavailable if this method is called by user
     * (current parameter value will be passed as 'oldValue').<br>
     * Please call <code>set_len2()</code> method instead.
     */
    void onChange_len2() {
      onChange_len2_xjal( len2 );
    }

    @AnyLogicInternalCodegenAPI
    void onChange_len2_xjal( double oldValue ) {
    }


    /**
     * x—координата верхнево левого угла помещения
```

```
590     */
  public
double  a0;

  /**
595   * Возвращает значение по умолчанию параметра <code>a0</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _a0_DefaultValue_xjal() {
600   final Main self = this;
    return 0.0;
  }

  public void set_a0( double a0 ) {
605   if (a0 == this.a0) {
      return;
    }
    double _oldValue_xjal = this.a0;
    this.a0 = a0;
610   onChange_a0_xjal( _oldValue_xjal );
    onChange();
  }

  /**
615   * Calls "On change" action for parameter a0.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_a0()</code> method instead.
   */
620 void onChange_a0() {
    onChange_a0_xjal( a0 );
  }

  @AnyLogicInternalCodegenAPI
625 void onChange_a0_xjal( double oldValue ) {
  }


  /**
630   * y—координата верхнево левого угла помещения
   */
  public
double  b0;

635 /**
   * Возвращает значение по умолчанию параметра <code>b0</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
640 public double _b0_DefaultValue_xjal() {
    final Main self = this;
    return 0.0;
  }

645 public void set_b0( double b0 ) {
    if (b0 == this.b0) {
      return;
    }
    double _oldValue_xjal = this.b0;
650 this.b0 = b0;
    onChange_b0_xjal( _oldValue_xjal );
    onChange();
  }

655 /**
   * Calls "On change" action for parameter b0.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_b0()</code> method instead.
660  */
  void onChange_b0() {
    onChange_b0_xjal( b0 );
  }

665 @AnyLogicInternalCodegenAPI
  void onChange_b0_xjal( double oldValue ) {
  }


670 /**
   * x—координата верхнево угла левого выхода
```

```
     */
    public
  double  a11;

    /**
     * Возвращает значение по умолчанию параметра <code>a11</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    @AnyLogicInternalCodegenAPI
    public double _a11_DefaultValue_xjal() {
      final Main self = this;
      return 0.0;
    }

    public void set_a11( double a11 ) {
      if (a11 == this.a11) {
        return;
      }
      double _oldValue_xjal = this.a11;
      this.a11 = a11;
      onChange_a11_xjal( _oldValue_xjal );
      onChange();
    }

    /**
     * Calls "On change" action for parameter a11.<br>
     * Note that 'oldValue' in that action will be unavailable if this method is called by user
     * (current parameter value will be passed as 'oldValue').<br>
     * Please call <code>set_a11()</code> method instead.
     */
    void onChange_a11() {
      onChange_a11_xjal( a11 );
    }

    @AnyLogicInternalCodegenAPI
    void onChange_a11_xjal( double oldValue ) {
    }


    /**
     * y—координата верхнево угла левого выхода
     */
    public
  double  b11;

    /**
     * Возвращает значение по умолчанию параметра <code>b11</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    @AnyLogicInternalCodegenAPI
    public double _b11_DefaultValue_xjal() {
      final Main self = this;
      return 0.0;
    }

    public void set_b11( double b11 ) {
      if (b11 == this.b11) {
        return;
      }
      double _oldValue_xjal = this.b11;
      this.b11 = b11;
      onChange_b11_xjal( _oldValue_xjal );
      onChange();
    }

    /**
     * Calls "On change" action for parameter b11.<br>
     * Note that 'oldValue' in that action will be unavailable if this method is called by user
     * (current parameter value will be passed as 'oldValue').<br>
     * Please call <code>set_b11()</code> method instead.
     */
    void onChange_b11() {
      onChange_b11_xjal( b11 );
    }

    @AnyLogicInternalCodegenAPI
    void onChange_b11_xjal( double oldValue ) {
    }


    /**
     * x—координата нижнего угла левого выхода
```

```
     */
755   public
    double  a12;

    /**
     * Возвращает значение по умолчанию параметра <code>a12</code>.
760  * <i>Пользователь не должен вызывать этот метод</i>
     */
    @AnyLogicInternalCodegenAPI
    public double _a12_DefaultValue_xjal() {
      final Main self = this;
765   return 0.0;
    }

    public void set_a12( double a12 ) {
      if (a12 == this.a12) {
770     return;
      }
      double _oldValue_xjal = this.a12;
      this.a12 = a12;
      onChange_a12_xjal( _oldValue_xjal );
775   onChange();
    }

    /**
     * Calls "On change" action for parameter a12.<br>
780  * Note that 'oldValue' in that action will be unavailable if this method is called by user
     * (current parameter value will be passed as 'oldValue').<br>
     * Please call <code>set_a12()</code> method instead.
     */
    void onChange_a12() {
785   onChange_a12_xjal( a12 );
    }

    @AnyLogicInternalCodegenAPI
    void onChange_a12_xjal( double oldValue ) {
790 }


    /**
     * y—координата нижнего угла левого выхода
795  */
    public
    double  b12;

    /**
800  * Возвращает значение по умолчанию параметра <code>b12</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    @AnyLogicInternalCodegenAPI
    public double _b12_DefaultValue_xjal() {
805   final Main self = this;
      return 0.0;
    }

    public void set_b12( double b12 ) {
810   if (b12 == this.b12) {
        return;
      }
      double _oldValue_xjal = this.b12;
      this.b12 = b12;
815   onChange_b12_xjal( _oldValue_xjal );
      onChange();
    }

    /**
820  * Calls "On change" action for parameter b12.<br>
     * Note that 'oldValue' in that action will be unavailable if this method is called by user
     * (current parameter value will be passed as 'oldValue').<br>
     * Please call <code>set_b12()</code> method instead.
     */
825 void onChange_b12() {
      onChange_b12_xjal( b12 );
    }

    @AnyLogicInternalCodegenAPI
830 void onChange_b12_xjal( double oldValue ) {
    }


    /**
835  * x—координата верхнево угла правого выхода
```

```
   */
  public
double  a21;

  /**
   * Возвращает значение по умолчанию параметра <code>a21</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _a21_DefaultValue_xjal() {
    final Main self = this;
    return 0.0;
  }

  public void set_a21( double a21 ) {
    if (a21 == this.a21) {
      return;
    }
    double _oldValue_xjal = this.a21;
    this.a21 = a21;
    onChange_a21_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter a21.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_a21()</code> method instead.
   */
  void onChange_a21() {
    onChange_a21_xjal( a21 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_a21_xjal( double oldValue ) {
  }


  /**
   * y—координата верхнево угла правого выхода
   */
  public
double  b21;

  /**
   * Возвращает значение по умолчанию параметра <code>b21</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _b21_DefaultValue_xjal() {
    final Main self = this;
    return 0.0;
  }

  public void set_b21( double b21 ) {
    if (b21 == this.b21) {
      return;
    }
    double _oldValue_xjal = this.b21;
    this.b21 = b21;
    onChange_b21_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter b21.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_b21()</code> method instead.
   */
  void onChange_b21() {
    onChange_b21_xjal( b21 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_b21_xjal( double oldValue ) {
  }


  /**
   * x—координата нижнего угла правого выхода
```

```
     */
    public
double  a22;

    /**
     * Возвращает значение по умолчанию параметра <code>a22</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    @AnyLogicInternalCodegenAPI
    public double _a22_DefaultValue_xjal() {
      final Main self = this;
      return 0.0;
    }

    public void set_a22( double a22 ) {
      if (a22 == this.a22) {
        return;
      }
      double _oldValue_xjal = this.a22;
      this.a22 = a22;
      onChange_a22_xjal( _oldValue_xjal );
      onChange();
    }

    /**
     * Calls "On change" action for parameter a22.<br>
     * Note that 'oldValue' in that action will be unavailable if this method is called by user
     * (current parameter value will be passed as 'oldValue').<br>
     * Please call <code>set_a22()</code> method instead.
     */
    void onChange_a22() {
      onChange_a22_xjal( a22 );
    }

    @AnyLogicInternalCodegenAPI
    void onChange_a22_xjal( double oldValue ) {
    }


    /**
     * y—координата нижнего угла правого выхода
     */
    public
double  b22;

    /**
     * Возвращает значение по умолчанию параметра <code>b22</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    @AnyLogicInternalCodegenAPI
    public double _b22_DefaultValue_xjal() {
      final Main self = this;
      return 0.0;
    }

    public void set_b22( double b22 ) {
      if (b22 == this.b22) {
        return;
      }
      double _oldValue_xjal = this.b22;
      this.b22 = b22;
      onChange_b22_xjal( _oldValue_xjal );
      onChange();
    }

    /**
     * Calls "On change" action for parameter b22.<br>
     * Note that 'oldValue' in that action will be unavailable if this method is called by user
     * (current parameter value will be passed as 'oldValue').<br>
     * Please call <code>set_b22()</code> method instead.
     */
    void onChange_b22() {
      onChange_b22_xjal( b22 );
    }

    @AnyLogicInternalCodegenAPI
    void onChange_b22_xjal( double oldValue ) {
    }


    /**
     * Количество клеток по вертикали
```

```
1000    */
  public
int   m_vert;

  /**
1005   * Возвращает значение по умолчанию параметра <code>m_vert</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public int _m_vert_DefaultValue_xjal() {
1010    final Main self = this;
    return
1
;
  }
1015
  public void set_m_vert( int m_vert ) {
    if (m_vert == this.m_vert) {
      return;
    }
1020    int _oldValue_xjal = this.m_vert;
    this.m_vert = m_vert;
    onChange_m_vert_xjal( _oldValue_xjal );
    onChange();
  }
1025
  /**
   * Calls "On change" action for parameter m_vert.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
1030   * Please call <code>set_m_vert()</code> method instead.
   */
  void onChange_m_vert() {
    onChange_m_vert_xjal( m_vert );
  }
1035
  @AnyLogicInternalCodegenAPI
  void onChange_m_vert_xjal( int oldValue ) {
  }


1040
  /**
   * Количество клеток по горизонтали
   */
  public
1045int   m_hor;

  /**
   * Возвращает значение по умолчанию параметра <code>m_hor</code>.
1050 * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public int _m_hor_DefaultValue_xjal() {
    final Main self = this;
    return
1055
;
  }

  public void set_m_hor( int m_hor ) {
1060  if (m_hor == this.m_hor) {
      return;
    }
    int _oldValue_xjal = this.m_hor;
    this.m_hor = m_hor;
1065  onChange_m_hor_xjal( _oldValue_xjal );
    onChange();
  }

  /**
1070 * Calls "On change" action for parameter m_hor.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_m_hor()</code> method instead.
   */
1075 void onChange_m_hor() {
    onChange_m_hor_xjal( m_hor );
  }

  @AnyLogicInternalCodegenAPI
1080 void onChange_m_hor_xjal( int oldValue ) {
  }
```

```
  /**
1085   * Функция распределения агентов в клетке
   */
  public
String  distr;

1090  /**
   * Возвращает значение по умолчанию параметра <code>distr</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
1095  public String _distr_DefaultValue_xjal() {
    final Main self = this;
    return
"Равномерное распределение"
;
1100  }

  public void set_distr( String distr ) {
    if (distr == this.distr) {
      return;
1105  }
    String _oldValue_xjal = this.distr;
    this.distr = distr;
    onChange_distr_xjal( _oldValue_xjal );
    onChange();
1110  }

  /**
   * Calls "On change" action for parameter distr.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
1115  * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_distr()</code> method instead.
   */
  void onChange_distr() {
    onChange_distr_xjal( distr );
1120  }

  @AnyLogicInternalCodegenAPI
  void onChange_distr_xjal( String oldValue ) {
  }
1125


  /**
   * Параметр сдвига распределения Лапласа по оси X для агентов
   */
1130  public
double  b_lapl_x;

  /**
   * Возвращает значение по умолчанию параметра <code>b_lapl_x</code>.
1135  * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _b_lapl_x_DefaultValue_xjal() {
    final Main self = this;
1140  return
0
;
  }

1145  public void set_b_lapl_x( double b_lapl_x ) {
    if (b_lapl_x == this.b_lapl_x) {
      return;
    }
    double _oldValue_xjal = this.b_lapl_x;
1150  this.b_lapl_x = b_lapl_x;
    onChange_b_lapl_x_xjal( _oldValue_xjal );
    onChange();
  }

1155  /**
   * Calls "On change" action for parameter b_lapl_x.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_b_lapl_x()</code> method instead.
1160  */
  void onChange_b_lapl_x() {
    onChange_b_lapl_x_xjal( b_lapl_x );
  }
```

```
  @AnyLogicInternalCodegenAPI
  void onChange_b_lapl_x_xjal( double oldValue ) {
  }


  /**
   * Параметр растяжения распределения Лапласа по оси X для агентов
   */
  public
double  a_lapl_x;

  /**
   * Возвращает значение по умолчанию параметра <code>a_lapl_x</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _a_lapl_x_DefaultValue_xjal() {
    final Main self = this;
    return
1
;
  }

  public void set_a_lapl_x( double a_lapl_x ) {
    if (a_lapl_x == this.a_lapl_x) {
      return;
    }
    double _oldValue_xjal = this.a_lapl_x;
    this.a_lapl_x = a_lapl_x;
    onChange_a_lapl_x_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter a_lapl_x.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_a_lapl_x()</code> method instead.
   */
  void onChange_a_lapl_x() {
    onChange_a_lapl_x_xjal( a_lapl_x );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_a_lapl_x_xjal( double oldValue ) {
  }


  /**
   * Математическое ожидание нормального распределения по оси X для агентов
   */
  public
double  mu_norm_x;

  /**
   * Возвращает значение по умолчанию параметра <code>mu_norm_x</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _mu_norm_x_DefaultValue_xjal() {
    final Main self = this;
    return
0
;
  }

  public void set_mu_norm_x( double mu_norm_x ) {
    if (mu_norm_x == this.mu_norm_x) {
      return;
    }
    double _oldValue_xjal = this.mu_norm_x;
    this.mu_norm_x = mu_norm_x;
    onChange_mu_norm_x_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter mu_norm_x.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_mu_norm_x()</code> method instead.
```

```
     */
    void onChange_mu_norm_x() {
      onChange_mu_norm_x_xjal( mu_norm_x );
    }

    @AnyLogicInternalCodegenAPI
    void onChange_mu_norm_x_xjal( double oldValue ) {
    }


    /**
     * Дисперсия нормального распределения по оси X для агентов
     */
    public
double  sig_norm_x;

    /**
     * Возвращает значение по умолчанию параметра <code>sig_norm_x</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    @AnyLogicInternalCodegenAPI
    public double _sig_norm_x_DefaultValue_xjal() {
      final Main self = this;
      return
1
;
    }

    public void set_sig_norm_x( double sig_norm_x ) {
      if (sig_norm_x == this.sig_norm_x) {
        return;
      }
      double _oldValue_xjal = this.sig_norm_x;
      this.sig_norm_x = sig_norm_x;
      onChange_sig_norm_x_xjal( _oldValue_xjal );
      onChange();
    }

    /**
     * Calls "On change" action for parameter sig_norm_x.<br>
     * Note that 'oldValue' in that action will be unavailable if this method is called by user
     * (current parameter value will be passed as 'oldValue').<br>
     * Please call <code>set_sig_norm_x()</code> method instead.
     */
    void onChange_sig_norm_x() {
      onChange_sig_norm_x_xjal( sig_norm_x );
    }

    @AnyLogicInternalCodegenAPI
    void onChange_sig_norm_x_xjal( double oldValue ) {
    }


    /**
     * Мода треугольного распределения по оси X для агентов
     */
    public
double  c_trian_x;

    /**
     * Возвращает значение по умолчанию параметра <code>c_trian_x</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    @AnyLogicInternalCodegenAPI
    public double _c_trian_x_DefaultValue_xjal() {
      final Main self = this;
      return
0
;
    }

    public void set_c_trian_x( double c_trian_x ) {
      if (c_trian_x == this.c_trian_x) {
        return;
      }
      double _oldValue_xjal = this.c_trian_x;
      this.c_trian_x = c_trian_x;
      onChange_c_trian_x_xjal( _oldValue_xjal );
      onChange();
    }

    /**
```

```
   * Calls "On change" action for parameter c_trian_x.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_c_trian_x()</code> method instead.
   */
  void onChange_c_trian_x() {
    onChange_c_trian_x_xjal( c_trian_x );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_c_trian_x_xjal( double oldValue ) {
  }


  /**
   * Общее количество агентов
   */
  public
int  agents_count;

  /**
   * Возвращает значение по умолчанию параметра <code>agents_count</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public int _agents_count_DefaultValue_xjal() {
    final Main self = this;
    return
10
;
  }

  public void set_agents_count( int agents_count ) {
    if (agents_count == this.agents_count) {
      return;
    }
    int _oldValue_xjal = this.agents_count;
    this.agents_count = agents_count;
    onChange_agents_count_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter agents_count.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_agents_count()</code> method instead.
   */
  void onChange_agents_count() {
    onChange_agents_count_xjal( agents_count );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_agents_count_xjal( int oldValue ) {
  }


  /**
   * Равномерное распределение или ручное распределение агентов по клеткам<br>
   * false — равномерное<br>
   * true — ручное
   */
  public
boolean  global_distr;

  /**
   * Возвращает значение по умолчанию параметра <code>global_distr</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public boolean _global_distr_DefaultValue_xjal() {
    final Main self = this;
    return
false
;
  }

  public void set_global_distr( boolean global_distr ) {
    if (global_distr == this.global_distr) {
      return;
    }
    boolean _oldValue_xjal = this.global_distr;
```

```
    this.global_distr = global_distr;
    onChange_global_distr_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter global_distr.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_global_distr()</code> method instead.
   */
  void onChange_global_distr() {
    onChange_global_distr_xjal( global_distr );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_global_distr_xjal( boolean oldValue ) {
  }


  /**
   * Параметр sigma_1
   */
  public
double  sigma_1;

  /**
   * Возвращает значение по умолчанию параметра <code>sigma_1</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _sigma_1_DefaultValue_xjal() {
    final Main self = this;
    return
sigma_2
;
  }

  public void set_sigma_1( double sigma_1 ) {
    if (sigma_1 == this.sigma_1) {
      return;
    }
    double _oldValue_xjal = this.sigma_1;
    this.sigma_1 = sigma_1;
    onChange_sigma_1_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter sigma_1.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_sigma_1()</code> method instead.
   */
  void onChange_sigma_1() {
    onChange_sigma_1_xjal( sigma_1 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_sigma_1_xjal( double oldValue ) {
  }


  /**
   * Параметр sigma_2
   */
  public
double  sigma_2;

  /**
   * Возвращает значение по умолчанию параметра <code>sigma_2</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _sigma_2_DefaultValue_xjal() {
    final Main self = this;
    return
1
;
  }

  public void set_sigma_2( double sigma_2 ) {
```

```
    if (sigma_2 == this.sigma_2) {
      return;
    }
    double _oldValue_xjal = this.sigma_2;
    this.sigma_2 = sigma_2;
    onChange_sigma_2_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter sigma_2.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_sigma_2()</code> method instead.
   */
  void onChange_sigma_2() {
    onChange_sigma_2_xjal( sigma_2 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_sigma_2_xjal( double oldValue ) {
  }


  /**
   * Параметр eta
   */
  public
double  eta;

  /**
   * Возвращает значение по умолчанию параметра <code>eta</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _eta_DefaultValue_xjal() {
    final Main self = this;
    return
5
;
  }

  public void set_eta( double eta ) {
    if (eta == this.eta) {
      return;
    }
    double _oldValue_xjal = this.eta;
    this.eta = eta;
    onChange_eta_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter eta.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_eta()</code> method instead.
   */
  void onChange_eta() {
    onChange_eta_xjal( eta );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_eta_xjal( double oldValue ) {
  }


  /**
   * Параметр gamma_acc
   */
  public
double  gamma_acc;

  /**
   * Возвращает значение по умолчанию параметра <code>gamma_acc</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _gamma_acc_DefaultValue_xjal() {
    final Main self = this;
    return
0.5
```

```
;
  }

  public void set_gamma_acc( double gamma_acc ) {
    if (gamma_acc == this.gamma_acc) {
      return;
    }
    double _oldValue_xjal = this.gamma_acc;
    this.gamma_acc = gamma_acc;
    onChange_gamma_acc_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter gamma_acc.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_gamma_acc()</code> method instead.
   */
  void onChange_gamma_acc() {
    onChange_gamma_acc_xjal( gamma_acc );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_gamma_acc_xjal( double oldValue ) {
  }


  /**
   * Параметр gamma_dec
   */
  public
double  gamma_dec;

  /**
   * Возвращает значение по умолчанию параметра <code>gamma_dec</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _gamma_dec_DefaultValue_xjal() {
    final Main self = this;
    return
0.5
;
  }

  public void set_gamma_dec( double gamma_dec ) {
    if (gamma_dec == this.gamma_dec) {
      return;
    }
    double _oldValue_xjal = this.gamma_dec;
    this.gamma_dec = gamma_dec;
    onChange_gamma_dec_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter gamma_dec.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_gamma_dec()</code> method instead.
   */
  void onChange_gamma_dec() {
    onChange_gamma_dec_xjal( gamma_dec );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_gamma_dec_xjal( double oldValue ) {
  }


  /**
   * Параметр gamma_exit
   */
  public
double  gamma_exit;

  /**
   * Возвращает значение по умолчанию параметра <code>gamma_exit</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
```

```
  public double _gamma_exit_DefaultValue_xjal() {
    final Main self = this;
    return
0.5
;
  }

  public void set_gamma_exit( double gamma_exit ) {
    if (gamma_exit == this.gamma_exit) {
      return;
    }
    double _oldValue_xjal = this.gamma_exit;
    this.gamma_exit = gamma_exit;
    onChange_gamma_exit_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter gamma_exit.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_gamma_exit()</code> method instead.
   */
  void onChange_gamma_exit() {
    onChange_gamma_exit_xjal( gamma_exit );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_gamma_exit_xjal( double oldValue ) {
  }


  /**
   * Параметр varepsilon_0
   */
  public
double  varepsilon_0;

  /**
   * Возвращает значение по умолчанию параметра <code>varepsilon_0</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _varepsilon_0_DefaultValue_xjal() {
    final Main self = this;
    return
5
;
  }

  public void set_varepsilon_0( double varepsilon_0 ) {
    if (varepsilon_0 == this.varepsilon_0) {
      return;
    }
    double _oldValue_xjal = this.varepsilon_0;
    this.varepsilon_0 = varepsilon_0;
    onChange_varepsilon_0_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter varepsilon_0.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_varepsilon_0()</code> method instead.
   */
  void onChange_varepsilon_0() {
    onChange_varepsilon_0_xjal( varepsilon_0 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_varepsilon_0_xjal( double oldValue ) {
  int index;
    index = 0;
    for ( Explosion self : explosion ) {
      self.set_epsilon0( scale.toPixels(varepsilon_0, LENGTH_UNIT_METER)
        );
      index++;
    }

  }
```

```
  /**
   * Параметр varepsilon_1
   */
  public
double  varepsilon_1;

  /**
   * Возвращает значение по умолчанию параметра <code>varepsilon_1</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _varepsilon_1_DefaultValue_xjal() {
    final Main self = this;
    return
10
;
  }

  public void set_varepsilon_1( double varepsilon_1 ) {
    if (varepsilon_1 == this.varepsilon_1) {
      return;
    }
    double _oldValue_xjal = this.varepsilon_1;
    this.varepsilon_1 = varepsilon_1;
    onChange_varepsilon_1_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter varepsilon_1.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_varepsilon_1()</code> method instead.
   */
  void onChange_varepsilon_1() {
    onChange_varepsilon_1_xjal( varepsilon_1 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_varepsilon_1_xjal( double oldValue ) {
  int index;
    index = 0;
    for ( Explosion self : explosion ) {
      self.set_epsilon1( scale.toPixels(varepsilon_1, LENGTH_UNIT_METER)
        );
      index++;
    }

  }


  /**
   * Параметр varepsilon_2
   */
  public
double  varepsilon_2;

  /**
   * Возвращает значение по умолчанию параметра <code>varepsilon_2</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _varepsilon_2_DefaultValue_xjal() {
    final Main self = this;
    return
13
;
  }

  public void set_varepsilon_2( double varepsilon_2 ) {
    if (varepsilon_2 == this.varepsilon_2) {
      return;
    }
    double _oldValue_xjal = this.varepsilon_2;
    this.varepsilon_2 = varepsilon_2;
    onChange_varepsilon_2_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter varepsilon_2.<br>
```

```
1820     * Note that 'oldValue' in that action will be unavailable if this method is called by user
     * (current parameter value will be passed as 'oldValue').<br>
     * Please call <code>set_varepsilon_2()</code> method instead.
     */
    void onChange_varepsilon_2() {
1825      onChange_varepsilon_2_xjal( varepsilon_2 );
    }

    @AnyLogicInternalCodegenAPI
    void onChange_varepsilon_2_xjal( double oldValue ) {
1830    int index;
      index = 0;
      for ( Explosion self : explosion ) {
        self.set_epsilon2( scale.toPixels(varepsilon_2 , LENGTH_UNIT_METER)
          );
1835      index++;
      }

    }


1840

    /**
     * Параметр zeta
     */
    public
1845 int  zeta;

    /**
     * Возвращает значение по умолчанию параметра <code>zeta</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
1850  */
    @AnyLogicInternalCodegenAPI
    public int _zeta_DefaultValue_xjal() {
      final Main self = this;
      return
1855 10
;
    }

    public void set_zeta( int zeta ) {
1860    if (zeta == this.zeta) {
        return;
      }
      int _oldValue_xjal = this.zeta;
      this.zeta = zeta;
1865  onChange_zeta_xjal( _oldValue_xjal );
      onChange();
    }

    /**
1870  * Calls "On change" action for parameter zeta.<br>
     * Note that 'oldValue' in that action will be unavailable if this method is called by user
     * (current parameter value will be passed as 'oldValue').<br>
     * Please call <code>set_zeta()</code> method instead.
     */
1875  void onChange_zeta() {
      onChange_zeta_xjal( zeta );
    }

    @AnyLogicInternalCodegenAPI
1880  void onChange_zeta_xjal( int oldValue ) {
    }


    public
1885 double  kappa_1;

    /**
     * Возвращает значение по умолчанию параметра <code>kappa_1</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
1890  */
    @AnyLogicInternalCodegenAPI
    public double _kappa_1_DefaultValue_xjal() {
      final Main self = this;
      return
1895 0.5
;
    }

    public void set_kappa_1( double kappa_1 ) {
1900    if (kappa_1 == this.kappa_1) {
        return;
```

```
    }
    double _oldValue_xjal = this.kappa_1;
    this.kappa_1 = kappa_1;
    onChange_kappa_1_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter kappa_1.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_kappa_1()</code> method instead.
   */
  void onChange_kappa_1() {
    onChange_kappa_1_xjal( kappa_1 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_kappa_1_xjal( double oldValue ) {
  }


  public
double  kappa_2;

  /**
   * Возвращает значение по умолчанию параметра <code>kappa_2</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _kappa_2_DefaultValue_xjal() {
    final Main self = this;
    return
2
;
  }

  public void set_kappa_2( double kappa_2 ) {
    if (kappa_2 == this.kappa_2) {
      return;
    }
    double _oldValue_xjal = this.kappa_2;
    this.kappa_2 = kappa_2;
    onChange_kappa_2_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter kappa_2.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_kappa_2()</code> method instead.
   */
  void onChange_kappa_2() {
    onChange_kappa_2_xjal( kappa_2 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_kappa_2_xjal( double oldValue ) {
  }


  public
double  kappa_3;

  /**
   * Возвращает значение по умолчанию параметра <code>kappa_3</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _kappa_3_DefaultValue_xjal() {
    final Main self = this;
    return
3
;
  }

  public void set_kappa_3( double kappa_3 ) {
    if (kappa_3 == this.kappa_3) {
      return;
    }
    double _oldValue_xjal = this.kappa_3;
```

```
    this.kappa_3 = kappa_3;
    onChange_kappa_3_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter kappa_3.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_kappa_3()</code> method instead.
   */
  void onChange_kappa_3() {
    onChange_kappa_3_xjal( kappa_3 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_kappa_3_xjal( double oldValue ) {
  }


  public
double  gamma_1;

  /**
   * Возвращает значение по умолчанию параметра <code>gamma_1</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _gamma_1_DefaultValue_xjal() {
    final Main self = this;
    return
0.5
;
  }

  public void set_gamma_1( double gamma_1 ) {
    if (gamma_1 == this.gamma_1) {
      return;
    }
    double _oldValue_xjal = this.gamma_1;
    this.gamma_1 = gamma_1;
    onChange_gamma_1_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter gamma_1.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_gamma_1()</code> method instead.
   */
  void onChange_gamma_1() {
    onChange_gamma_1_xjal( gamma_1 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_gamma_1_xjal( double oldValue ) {
  }


  public
double  gamma_2;

  /**
   * Возвращает значение по умолчанию параметра <code>gamma_2</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _gamma_2_DefaultValue_xjal() {
    final Main self = this;
    return
0.5
;
  }

  public void set_gamma_2( double gamma_2 ) {
    if (gamma_2 == this.gamma_2) {
      return;
    }
    double _oldValue_xjal = this.gamma_2;
    this.gamma_2 = gamma_2;
    onChange_gamma_2_xjal( _oldValue_xjal );
```

```
    onChange();
  }

  /**
   * Calls "On change" action for parameter gamma_2.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_gamma_2()</code> method instead.
   */
  void onChange_gamma_2() {
    onChange_gamma_2_xjal( gamma_2 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_gamma_2_xjal( double oldValue ) {
  }


  public
double  gamma_3;

  /**
   * Возвращает значение по умолчанию параметра <code>gamma_3</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _gamma_3_DefaultValue_xjal() {
    final Main self = this;
    return
0.5
;
  }

  public void set_gamma_3( double gamma_3 ) {
    if (gamma_3 == this.gamma_3) {
      return;
    }
    double _oldValue_xjal = this.gamma_3;
    this.gamma_3 = gamma_3;
    onChange_gamma_3_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter gamma_3.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_gamma_3()</code> method instead.
   */
  void onChange_gamma_3() {
    onChange_gamma_3_xjal( gamma_3 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_gamma_3_xjal( double oldValue ) {
  }


  public
double  gamma_4;

  /**
   * Возвращает значение по умолчанию параметра <code>gamma_4</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _gamma_4_DefaultValue_xjal() {
    final Main self = this;
    return
0.5
;
  }

  public void set_gamma_4( double gamma_4 ) {
    if (gamma_4 == this.gamma_4) {
      return;
    }
    double _oldValue_xjal = this.gamma_4;
    this.gamma_4 = gamma_4;
    onChange_gamma_4_xjal( _oldValue_xjal );
    onChange();
  }
```

```
  /**
   * Calls "On change" action for parameter gamma_4.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_gamma_4()</code> method instead.
   */
  void onChange_gamma_4() {
    onChange_gamma_4_xjal( gamma_4 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_gamma_4_xjal( double oldValue ) {
  }


  public
double  lambda_vel;

  /**
   * Возвращает значение по умолчанию параметра <code>lambda_vel</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _lambda_vel_DefaultValue_xjal() {
    final Main self = this;
    return
0.5
;
  }

  public void set_lambda_vel( double lambda_vel ) {
    if (lambda_vel == this.lambda_vel) {
      return;
    }
    double _oldValue_xjal = this.lambda_vel;
    this.lambda_vel = lambda_vel;
    onChange_lambda_vel_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter lambda_vel.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_lambda_vel()</code> method instead.
   */
  void onChange_lambda_vel() {
    onChange_lambda_vel_xjal( lambda_vel );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_lambda_vel_xjal( double oldValue ) {
  }


  public
double  lambda_max;

  /**
   * Возвращает значение по умолчанию параметра <code>lambda_max</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _lambda_max_DefaultValue_xjal() {
    final Main self = this;
    return
0.5
;
  }

  public void set_lambda_max( double lambda_max ) {
    if (lambda_max == this.lambda_max) {
      return;
    }
    double _oldValue_xjal = this.lambda_max;
    this.lambda_max = lambda_max;
    onChange_lambda_max_xjal( _oldValue_xjal );
    onChange();
  }

  /**
```

```
2230    * Calls "On change" action for parameter lambda_max.<br>
        * Note that 'oldValue' in that action will be unavailable if this method is called by user
        * (current parameter value will be passed as 'oldValue').<br>
        * Please call <code>set_lambda_max()</code> method instead.
        */
2235    void onChange_lambda_max() {
          onChange_lambda_max_xjal( lambda_max );
        }

        @AnyLogicInternalCodegenAPI
2240    void onChange_lambda_max_xjal( double oldValue ) {
        }


        public
2245  String  distr_expl;

        /**
         * Возвращает значение по умолчанию параметра <code>distr_expl</code>.
         * <i>Пользователь не должен вызывать этот метод</i>
2250     */
        @AnyLogicInternalCodegenAPI
        public String _distr_expl_DefaultValue_xjal() {
          final Main self = this;
          return
2255  "Равномерное распределение"
  ;
        }

        public void set_distr_expl( String distr_expl ) {
2260      if (distr_expl == this.distr_expl) {
            return;
          }
          String _oldValue_xjal = this.distr_expl;
          this.distr_expl = distr_expl;
2265      onChange_distr_expl_xjal( _oldValue_xjal );
          onChange();
        }

        /**
2270     * Calls "On change" action for parameter distr_expl.<br>
         * Note that 'oldValue' in that action will be unavailable if this method is called by user
         * (current parameter value will be passed as 'oldValue').<br>
         * Please call <code>set_distr_expl()</code> method instead.
         */
2275    void onChange_distr_expl() {
          onChange_distr_expl_xjal( distr_expl );
        }

        @AnyLogicInternalCodegenAPI
2280    void onChange_distr_expl_xjal( String oldValue ) {
        }


        public
2285  double  b_lapl_expl_x;

        /**
         * Возвращает значение по умолчанию параметра <code>b_lapl_expl_x</code>.
         * <i>Пользователь не должен вызывать этот метод</i>
2290     */
        @AnyLogicInternalCodegenAPI
        public double _b_lapl_expl_x_DefaultValue_xjal() {
          final Main self = this;
          return
2295  0
  ;
        }

        public void set_b_lapl_expl_x( double b_lapl_expl_x ) {
2300      if (b_lapl_expl_x == this.b_lapl_expl_x) {
            return;
          }
          double _oldValue_xjal = this.b_lapl_expl_x;
          this.b_lapl_expl_x = b_lapl_expl_x;
2305      onChange_b_lapl_expl_x_xjal( _oldValue_xjal );
          onChange();
        }

        /**
2310     * Calls "On change" action for parameter b_lapl_expl_x.<br>
         * Note that 'oldValue' in that action will be unavailable if this method is called by user
```

```
     * (current parameter value will be passed as 'oldValue').<br>
     * Please call <code>set_b_lapl_expl_x()</code> method instead.
     */
2315   void onChange_b_lapl_expl_x() {
         onChange_b_lapl_expl_x_xjal( b_lapl_expl_x );
       }

       @AnyLogicInternalCodegenAPI
2320   void onChange_b_lapl_expl_x_xjal( double oldValue ) {
       }


       public
2325 double  a_lapl_expl_x;

       /**
        * Возвращает значение по умолчанию параметра <code>a_lapl_expl_x</code>.
        * <i>Пользователь не должен вызывать этот метод</i>
2330    */
       @AnyLogicInternalCodegenAPI
       public double _a_lapl_expl_x_DefaultValue_xjal() {
         final Main self = this;
         return
2335 1
       ;
       }

       public void set_a_lapl_expl_x( double a_lapl_expl_x ) {
2340     if (a_lapl_expl_x == this.a_lapl_expl_x) {
           return;
         }
         double _oldValue_xjal = this.a_lapl_expl_x;
         this.a_lapl_expl_x = a_lapl_expl_x;
2345     onChange_a_lapl_expl_x_xjal( _oldValue_xjal );
         onChange();
       }

       /**
2350    * Calls "On change" action for parameter a_lapl_expl_x.<br>
        * Note that 'oldValue' in that action will be unavailable if this method is called by user
        * (current parameter value will be passed as 'oldValue').<br>
        * Please call <code>set_a_lapl_expl_x()</code> method instead.
        */
2355   void onChange_a_lapl_expl_x() {
         onChange_a_lapl_expl_x_xjal( a_lapl_expl_x );
       }

       @AnyLogicInternalCodegenAPI
2360   void onChange_a_lapl_expl_x_xjal( double oldValue ) {
       }


       public
2365 double  mu_norm_expl_x;

       /**
        * Возвращает значение по умолчанию параметра <code>mu_norm_expl_x</code>.
        * <i>Пользователь не должен вызывать этот метод</i>
2370    */
       @AnyLogicInternalCodegenAPI
       public double _mu_norm_expl_x_DefaultValue_xjal() {
         final Main self = this;
         return
2375 0
       ;
       }

       public void set_mu_norm_expl_x( double mu_norm_expl_x ) {
2380     if (mu_norm_expl_x == this.mu_norm_expl_x) {
           return;
         }
         double _oldValue_xjal = this.mu_norm_expl_x;
         this.mu_norm_expl_x = mu_norm_expl_x;
2385     onChange_mu_norm_expl_x_xjal( _oldValue_xjal );
         onChange();
       }

       /**
2390    * Calls "On change" action for parameter mu_norm_expl_x.<br>
        * Note that 'oldValue' in that action will be unavailable if this method is called by user
        * (current parameter value will be passed as 'oldValue').<br>
        * Please call <code>set_mu_norm_expl_x()</code> method instead.
```

```
     */
2395   void onChange_mu_norm_expl_x() {
         onChange_mu_norm_expl_x_xjal( mu_norm_expl_x );
       }

       @AnyLogicInternalCodegenAPI
2400   void onChange_mu_norm_expl_x_xjal( double oldValue ) {
       }


       public
2405 double  sig_norm_expl_x;

       /**
        * Возвращает значение по умолчанию параметра <code>sig_norm_expl_x</code>.
        * <i>Пользователь не должен вызывать этот метод</i>
2410    */
       @AnyLogicInternalCodegenAPI
       public double _sig_norm_expl_x_DefaultValue_xjal() {
         final Main self = this;
         return
2415 1
       ;
       }

       public void set_sig_norm_expl_x( double sig_norm_expl_x ) {
2420     if (sig_norm_expl_x == this.sig_norm_expl_x) {
           return;
         }
         double _oldValue_xjal = this.sig_norm_expl_x;
         this.sig_norm_expl_x = sig_norm_expl_x;
2425     onChange_sig_norm_expl_x_xjal( _oldValue_xjal );
         onChange();
       }

       /**
2430    * Calls "On change" action for parameter sig_norm_expl_x.<br>
        * Note that 'oldValue' in that action will be unavailable if this method is called by user
        * (current parameter value will be passed as 'oldValue').<br>
        * Please call <code>set_sig_norm_expl_x()</code> method instead.
        */
2435   void onChange_sig_norm_expl_x() {
         onChange_sig_norm_expl_x_xjal( sig_norm_expl_x );
       }

       @AnyLogicInternalCodegenAPI
2440   void onChange_sig_norm_expl_x_xjal( double oldValue ) {
       }


       public
2445 double  c_trian_expl_x;

       /**
        * Возвращает значение по умолчанию параметра <code>c_trian_expl_x</code>.
        * <i>Пользователь не должен вызывать этот метод</i>
2450    */
       @AnyLogicInternalCodegenAPI
       public double _c_trian_expl_x_DefaultValue_xjal() {
         final Main self = this;
         return
2455 0
       ;
       }

       public void set_c_trian_expl_x( double c_trian_expl_x ) {
2460     if (c_trian_expl_x == this.c_trian_expl_x) {
           return;
         }
         double _oldValue_xjal = this.c_trian_expl_x;
         this.c_trian_expl_x = c_trian_expl_x;
2465     onChange_c_trian_expl_x_xjal( _oldValue_xjal );
         onChange();
       }

       /**
2470    * Calls "On change" action for parameter c_trian_expl_x.<br>
        * Note that 'oldValue' in that action will be unavailable if this method is called by user
        * (current parameter value will be passed as 'oldValue').<br>
        * Please call <code>set_c_trian_expl_x()</code> method instead.
        */
2475   void onChange_c_trian_expl_x() {
```

```
    onChange_c_trian_expl_x_xjal( c_trian_expl_x );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_c_trian_expl_x_xjal( double oldValue ) {
  }


  /**
   * Функция распределения стороны выхода
   */
  public
String  distr_exit;

  /**
   * Возвращает значение по умолчанию параметра <code>distr_exit</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public String _distr_exit_DefaultValue_xjal() {
    final Main self = this;
    return
"Равномерное распределение"
;
  }

  public void set_distr_exit( String distr_exit ) {
    if (distr_exit == this.distr_exit) {
      return;
    }
    String _oldValue_xjal = this.distr_exit;
    this.distr_exit = distr_exit;
    onChange_distr_exit_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter distr_exit.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_distr_exit()</code> method instead.
   */
  void onChange_distr_exit() {
    onChange_distr_exit_xjal( distr_exit );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_distr_exit_xjal( String oldValue ) {
  }


  /**
   * Функция распределения направления турбулентности
   */
  public
String  distr_turb;

  /**
   * Возвращает значение по умолчанию параметра <code>distr_turb</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public String _distr_turb_DefaultValue_xjal() {
    final Main self = this;
    return
"Равномерное распределение"
;
  }

  public void set_distr_turb( String distr_turb ) {
    if (distr_turb == this.distr_turb) {
      return;
    }
    String _oldValue_xjal = this.distr_turb;
    this.distr_turb = distr_turb;
    onChange_distr_turb_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter distr_turb.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
```

```
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_distr_turb()</code> method instead.
   */
  void onChange_distr_turb() {
    onChange_distr_turb_xjal( distr_turb );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_distr_turb_xjal( String oldValue ) {
  }


  /**
   * Максимальный угол отклонения при турбулентности
   */
  public
int  turb;

  /**
   * Возвращает значение по умолчанию параметра <code>turb</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public int _turb_DefaultValue_xjal() {
    final Main self = this;
    return
30
;
  }

  public void set_turb( int turb ) {
    if (turb == this.turb) {
      return;
    }
    int _oldValue_xjal = this.turb;
    this.turb = turb;
    onChange_turb_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter turb.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_turb()</code> method instead.
   */
  void onChange_turb() {
    onChange_turb_xjal( turb );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_turb_xjal( int oldValue ) {
  }


  /**
   * Параметр сдвига распределения Лапласа по оси Y для агентов
   */
  public
double  b_lapl_y;

  /**
   * Возвращает значение по умолчанию параметра <code>b_lapl_y</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _b_lapl_y_DefaultValue_xjal() {
    final Main self = this;
    return
0
;
  }

  public void set_b_lapl_y( double b_lapl_y ) {
    if (b_lapl_y == this.b_lapl_y) {
      return;
    }
    double _oldValue_xjal = this.b_lapl_y;
    this.b_lapl_y = b_lapl_y;
    onChange_b_lapl_y_xjal( _oldValue_xjal );
    onChange();
  }
```

```
2640
  /**
   * Calls "On change" action for parameter b_lapl_y.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
2645  * Please call <code>set_b_lapl_y()</code> method instead.
   */
  void onChange_b_lapl_y() {
    onChange_b_lapl_y_xjal( b_lapl_y );
  }
2650
  @AnyLogicInternalCodegenAPI
  void onChange_b_lapl_y_xjal( double oldValue ) {
  }


2655
  /**
   * Параметр растяжения распределения Лапласа по оси Y для агентов
   */
  public
2660 double  a_lapl_y;

  /**
   * Возвращает значение по умолчанию параметра <code>a_lapl_y</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
2665  */
  @AnyLogicInternalCodegenAPI
  public double _a_lapl_y_DefaultValue_xjal() {
    final Main self = this;
    return
2670 1
;
  }

  public void set_a_lapl_y( double a_lapl_y ) {
2675  if (a_lapl_y == this.a_lapl_y) {
      return;
    }
    double _oldValue_xjal = this.a_lapl_y;
    this.a_lapl_y = a_lapl_y;
2680  onChange_a_lapl_y_xjal( _oldValue_xjal );
    onChange();
  }

  /**
2685  * Calls "On change" action for parameter a_lapl_y.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_a_lapl_y()</code> method instead.
   */
2690 void onChange_a_lapl_y() {
    onChange_a_lapl_y_xjal( a_lapl_y );
  }

  @AnyLogicInternalCodegenAPI
2695 void onChange_a_lapl_y_xjal( double oldValue ) {
  }


  /**
2700  * Математическое ожидание нормального распределения по оси Y для агентов
   */
  public
double  mu_norm_y;

2705 /**
   * Возвращает значение по умолчанию параметра <code>mu_norm_y</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
2710 public double _mu_norm_y_DefaultValue_xjal() {
    final Main self = this;
    return
0
;
2715 }

  public void set_mu_norm_y( double mu_norm_y ) {
    if (mu_norm_y == this.mu_norm_y) {
      return;
2720  }
    double _oldValue_xjal = this.mu_norm_y;
```

```
    this.mu_norm_y = mu_norm_y;
    onChange_mu_norm_y_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter mu_norm_y.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_mu_norm_y()</code> method instead.
   */
  void onChange_mu_norm_y() {
    onChange_mu_norm_y_xjal( mu_norm_y );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_mu_norm_y_xjal( double oldValue ) {
  }


  /**
   * Дисперсия нормального распределения по оси Y для агентов
   */
  public
double  sig_norm_y;

  /**
   * Возвращает значение по умолчанию параметра <code>sig_norm_y</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _sig_norm_y_DefaultValue_xjal() {
    final Main self = this;
    return
1
;
  }

  public void set_sig_norm_y( double sig_norm_y ) {
    if (sig_norm_y == this.sig_norm_y) {
      return;
    }
    double _oldValue_xjal = this.sig_norm_y;
    this.sig_norm_y = sig_norm_y;
    onChange_sig_norm_y_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter sig_norm_y.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_sig_norm_y()</code> method instead.
   */
  void onChange_sig_norm_y() {
    onChange_sig_norm_y_xjal( sig_norm_y );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_sig_norm_y_xjal( double oldValue ) {
  }


  /**
   * Мода треугольного распределения по оси Y для агентов
   */
  public
double  c_trian_y;

  /**
   * Возвращает значение по умолчанию параметра <code>c_trian_y</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _c_trian_y_DefaultValue_xjal() {
    final Main self = this;
    return
0
;
  }

  public void set_c_trian_y( double c_trian_y ) {
```

```
    if (c_trian_y == this.c_trian_y) {
      return;
    }
    double _oldValue_xjal = this.c_trian_y;
    this.c_trian_y = c_trian_y;
    onChange_c_trian_y_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter c_trian_y.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_c_trian_y()</code> method instead.
   */
  void onChange_c_trian_y() {
    onChange_c_trian_y_xjal( c_trian_y );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_c_trian_y_xjal( double oldValue ) {
  }


  public
double  b_lapl_expl_y;

  /**
   * Возвращает значение по умолчанию параметра <code>b_lapl_expl_y</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _b_lapl_expl_y_DefaultValue_xjal() {
    final Main self = this;
    return
0
;
  }

  public void set_b_lapl_expl_y( double b_lapl_expl_y ) {
    if (b_lapl_expl_y == this.b_lapl_expl_y) {
      return;
    }
    double _oldValue_xjal = this.b_lapl_expl_y;
    this.b_lapl_expl_y = b_lapl_expl_y;
    onChange_b_lapl_expl_y_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter b_lapl_expl_y.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_b_lapl_expl_y()</code> method instead.
   */
  void onChange_b_lapl_expl_y() {
    onChange_b_lapl_expl_y_xjal( b_lapl_expl_y );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_b_lapl_expl_y_xjal( double oldValue ) {
  }


  public
double  a_lapl_expl_y;

  /**
   * Возвращает значение по умолчанию параметра <code>a_lapl_expl_y</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _a_lapl_expl_y_DefaultValue_xjal() {
    final Main self = this;
    return
1
;
  }

  public void set_a_lapl_expl_y( double a_lapl_expl_y ) {
    if (a_lapl_expl_y == this.a_lapl_expl_y) {
      return;
```

```
    }
    double _oldValue_xjal = this.a_lapl_expl_y;
    this.a_lapl_expl_y = a_lapl_expl_y;
    onChange_a_lapl_expl_y_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter a_lapl_expl_y.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_a_lapl_expl_y()</code> method instead.
   */
  void onChange_a_lapl_expl_y() {
    onChange_a_lapl_expl_y_xjal( a_lapl_expl_y );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_a_lapl_expl_y_xjal( double oldValue ) {
  }


  public
double  mu_norm_expl_y;

  /**
   * Возвращает значение по умолчанию параметра <code>mu_norm_expl_y</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _mu_norm_expl_y_DefaultValue_xjal() {
    final Main self = this;
    return
0
;
  }

  public void set_mu_norm_expl_y( double mu_norm_expl_y ) {
    if (mu_norm_expl_y == this.mu_norm_expl_y) {
      return;
    }
    double _oldValue_xjal = this.mu_norm_expl_y;
    this.mu_norm_expl_y = mu_norm_expl_y;
    onChange_mu_norm_expl_y_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter mu_norm_expl_y.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_mu_norm_expl_y()</code> method instead.
   */
  void onChange_mu_norm_expl_y() {
    onChange_mu_norm_expl_y_xjal( mu_norm_expl_y );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_mu_norm_expl_y_xjal( double oldValue ) {
  }


  public
double  sig_norm_expl_y;

  /**
   * Возвращает значение по умолчанию параметра <code>sig_norm_expl_y</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _sig_norm_expl_y_DefaultValue_xjal() {
    final Main self = this;
    return
1
;
  }

  public void set_sig_norm_expl_y( double sig_norm_expl_y ) {
    if (sig_norm_expl_y == this.sig_norm_expl_y) {
      return;
    }
    double _oldValue_xjal = this.sig_norm_expl_y;
```

```
    this.sig_norm_expl_y = sig_norm_expl_y;
    onChange_sig_norm_expl_y_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter sig_norm_expl_y.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_sig_norm_expl_y()</code> method instead.
   */
  void onChange_sig_norm_expl_y() {
    onChange_sig_norm_expl_y_xjal( sig_norm_expl_y );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_sig_norm_expl_y_xjal( double oldValue ) {
  }


  public
double  c_trian_expl_y;

  /**
   * Возвращает значение по умолчанию параметра <code>c_trian_expl_y</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _c_trian_expl_y_DefaultValue_xjal() {
    final Main self = this;
    return
0
;
  }

  public void set_c_trian_expl_y( double c_trian_expl_y ) {
    if (c_trian_expl_y == this.c_trian_expl_y) {
      return;
    }
    double _oldValue_xjal = this.c_trian_expl_y;
    this.c_trian_expl_y = c_trian_expl_y;
    onChange_c_trian_expl_y_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter c_trian_expl_y.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_c_trian_expl_y()</code> method instead.
   */
  void onChange_c_trian_expl_y() {
    onChange_c_trian_expl_y_xjal( c_trian_expl_y );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_c_trian_expl_y_xjal( double oldValue ) {
  }


  /**
   * Текущая версия прогона модели
   */
  public
int  version;

  /**
   * Возвращает значение по умолчанию параметра <code>version</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public int _version_DefaultValue_xjal() {
    final Main self = this;
    return
1
;
  }

  public void set_version( int version ) {
    if (version == this.version) {
      return;
    }
```

```
    int _oldValue_xjal = this.version;
    this.version = version;
    onChange_version_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter version.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_version()</code> method instead.
   */
  void onChange_version() {
    onChange_version_xjal( version );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_version_xjal( int oldValue ) {
  }


  public
double  lambda_avg;

  /**
   * Возвращает значение по умолчанию параметра <code>lambda_avg</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _lambda_avg_DefaultValue_xjal() {
    final Main self = this;
    return
0.5
;
  }

  public void set_lambda_avg( double lambda_avg ) {
    if (lambda_avg == this.lambda_avg) {
      return;
    }
    double _oldValue_xjal = this.lambda_avg;
    this.lambda_avg = lambda_avg;
    onChange_lambda_avg_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter lambda_avg.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_lambda_avg()</code> method instead.
   */
  void onChange_lambda_avg() {
    onChange_lambda_avg_xjal( lambda_avg );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_lambda_avg_xjal( double oldValue ) {
  }


  public
double  gamma_5;

  /**
   * Возвращает значение по умолчанию параметра <code>gamma_5</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _gamma_5_DefaultValue_xjal() {
    final Main self = this;
    return
0.5
;
  }

  public void set_gamma_5( double gamma_5 ) {
    if (gamma_5 == this.gamma_5) {
      return;
    }
    double _oldValue_xjal = this.gamma_5;
    this.gamma_5 = gamma_5;
```

```
    onChange_gamma_5_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter gamma_5.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_gamma_5()</code> method instead.
   */
  void onChange_gamma_5() {
    onChange_gamma_5_xjal( gamma_5 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_gamma_5_xjal( double oldValue ) {
  }


  /**
   * Количество секунд в единице модельного времени
   */
  public
double  model_second;

  /**
   * Возвращает значение по умолчанию параметра <code>model_second</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _model_second_DefaultValue_xjal() {
    final Main self = this;
    return
second()
;
  }

  public void set_model_second( double model_second ) {
    if (model_second == this.model_second) {
      return;
    }
    double _oldValue_xjal = this.model_second;
    this.model_second = model_second;
    onChange_model_second_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter model_second.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_model_second()</code> method instead.
   */
  void onChange_model_second() {
    onChange_model_second_xjal( model_second );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_model_second_xjal( double oldValue ) {
  }


  /**
   * Массив параметра varsigma
   */
  public
double[]  varsigma;

  /**
   * Возвращает значение по умолчанию параметра <code>varsigma</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double[] _varsigma_DefaultValue_xjal() {
    final Main self = this;
    return new double[]
{1,1,1,1}
;
  }

  public void set_varsigma( double[] varsigma ) {
    if (varsigma == this.varsigma) {
```

```
      return;
    }
    double[] _oldValue_xjal = this.varsigma;
    this.varsigma = varsigma;
    onChange_varsigma_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter varsigma.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_varsigma()</code> method instead.
   */
  void onChange_varsigma() {
    onChange_varsigma_xjal( varsigma );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_varsigma_xjal( double[] oldValue ) {
  }


  /**
   * Массив параметра varrho_3
   */
  public
double[]  varrho_3;

  /**
   * Возвращает значение по умолчанию параметра <code>varrho_3</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double[] _varrho_3_DefaultValue_xjal() {
    final Main self = this;
    return new double[]
{0.8,0.8,0.8,0.8}
;
  }

  public void set_varrho_3( double[] varrho_3 ) {
    if (varrho_3 == this.varrho_3) {
      return;
    }
    double[] _oldValue_xjal = this.varrho_3;
    this.varrho_3 = varrho_3;
    onChange_varrho_3_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter varrho_3.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_varrho_3()</code> method instead.
   */
  void onChange_varrho_3() {
    onChange_varrho_3_xjal( varrho_3 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_varrho_3_xjal( double[] oldValue ) {
  }


  /**
   * Массив параметра varrho_4
   */
  public
double[]  varrho_4;

  /**
   * Возвращает значение по умолчанию параметра <code>varrho_4</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double[] _varrho_4_DefaultValue_xjal() {
    final Main self = this;
    return new double[]
{0.9,0.9,0.9,0.9}
;
```

```
  }

  public void set_varrho_4( double[] varrho_4 ) {
    if (varrho_4 == this.varrho_4) {
      return;
    }
    double[] _oldValue_xjal = this.varrho_4;
    this.varrho_4 = varrho_4;
    onChange_varrho_4_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter varrho_4.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_varrho_4()</code> method instead.
   */
  void onChange_varrho_4() {
    onChange_varrho_4_xjal( varrho_4 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_varrho_4_xjal( double[] oldValue ) {
  }


  /**
   * Массив параметра vartheta
   */
  public
double[]  vartheta;

  /**
   * Возвращает значение по умолчанию параметра <code>vartheta</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double[] _vartheta_DefaultValue_xjal() {
    final Main self = this;
    return new double[]
{3,2,1,4}
;
  }

  public void set_vartheta( double[] vartheta ) {
    if (vartheta == this.vartheta) {
      return;
    }
    double[] _oldValue_xjal = this.vartheta;
    this.vartheta = vartheta;
    onChange_vartheta_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter vartheta.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_vartheta()</code> method instead.
   */
  void onChange_vartheta() {
    onChange_vartheta_xjal( vartheta );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_vartheta_xjal( double[] oldValue ) {
  }


  /**
   * Массив параметра theta
   */
  public
int[]  theta;

  /**
   * Возвращает значение по умолчанию параметра <code>theta</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public int[] _theta_DefaultValue_xjal() {
```

```
    final Main self = this;
    return new int[]
{170,170,120,80}
;
  }

  public void set_theta( int[] theta ) {
    if (theta == this.theta) {
      return;
    }
    int[] _oldValue_xjal = this.theta;
    this.theta = theta;
    onChange_theta_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter theta.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_theta()</code> method instead.
   */
  void onChange_theta() {
    onChange_theta_xjal( theta );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_theta_xjal( int[] oldValue ) {
  }


  /**
   * Массив параметра theta_hat
   */
  public
int[]  theta_hat;

  /**
   * Возвращает значение по умолчанию параметра <code>theta_hat</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public int[] _theta_hat_DefaultValue_xjal() {
    final Main self = this;
    return new int[]
{50,50,30,30}
;
  }

  public void set_theta_hat( int[] theta_hat ) {
    if (theta_hat == this.theta_hat) {
      return;
    }
    int[] _oldValue_xjal = this.theta_hat;
    this.theta_hat = theta_hat;
    onChange_theta_hat_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter theta_hat.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_theta_hat()</code> method instead.
   */
  void onChange_theta_hat() {
    onChange_theta_hat_xjal( theta_hat );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_theta_hat_xjal( int[] oldValue ) {
  }


  /**
   * Коллекция распределения агентов по клеткам<br>
   * индекс — номер клетки<br>
   * значение — количество агентов в клетке
   */
  public
int[]  global_distr_array;
```

```
3460  /**
       * Возвращает значение по умолчанию параметра <code>global_distr_array</code>.
       * <i>Пользователь не должен вызывать этот метод</i>
       */
      @AnyLogicInternalCodegenAPI
3465  public int[] _global_distr_array_DefaultValue_xjal() {
        final Main self = this;
        return null;
      }

3470  public void set_global_distr_array( int[] global_distr_array ) {
        if (global_distr_array == this.global_distr_array) {
          return;
        }
        int[] _oldValue_xjal = this.global_distr_array;
3475    this.global_distr_array = global_distr_array;
        onChange_global_distr_array_xjal( _oldValue_xjal );
        onChange();
      }

3480  /**
       * Calls "On change" action for parameter global_distr_array.<br>
       * Note that 'oldValue' in that action will be unavailable if this method is called by user
       * (current parameter value will be passed as 'oldValue').<br>
       * Please call <code>set_global_distr_array()</code> method instead.
3485   */
      void onChange_global_distr_array() {
        onChange_global_distr_array_xjal( global_distr_array );
      }

3490  @AnyLogicInternalCodegenAPI
      void onChange_global_distr_array_xjal( int[] oldValue ) {
      }


3495  /**
       * Ширина области моделирования
       */
      public
    double  width;
3500
      /**
       * Возвращает значение по умолчанию параметра <code>width</code>.
       * <i>Пользователь не должен вызывать этот метод</i>
       */
3505  @AnyLogicInternalCodegenAPI
      public double _width_DefaultValue_xjal() {
        final Main self = this;
        return 0.0;
      }
3510
      public void set_width( double width ) {
        if (width == this.width) {
          return;
        }
3515    double _oldValue_xjal = this.width;
        this.width = width;
        onChange_width_xjal( _oldValue_xjal );
        onChange();
      }
3520
      /**
       * Calls "On change" action for parameter width.<br>
       * Note that 'oldValue' in that action will be unavailable if this method is called by user
       * (current parameter value will be passed as 'oldValue').<br>
3525   * Please call <code>set_width()</code> method instead.
       */
      void onChange_width() {
        onChange_width_xjal( width );
      }
3530
      @AnyLogicInternalCodegenAPI
      void onChange_width_xjal( double oldValue ) {
      }


3535
      /**
       * Высота области моделирования
       */
      public
3540  double  height;
```

```
  /**
   * Возвращает значение по умолчанию параметра <code>height</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _height_DefaultValue_xjal() {
    final Main self = this;
    return 0.0;
  }

  public void set_height( double height ) {
    if (height == this.height) {
      return;
    }
    double _oldValue_xjal = this.height;
    this.height = height;
    onChange_height_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter height.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_height()</code> method instead.
   */
  void onChange_height() {
    onChange_height_xjal( height );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_height_xjal( double oldValue ) {
  }


  public
double[]  beta_occ;

  /**
   * Возвращает значение по умолчанию параметра <code>beta_occ</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double[] _beta_occ_DefaultValue_xjal() {
    final Main self = this;
    return new double[]
{1,1,1,1}
;
  }

  public void set_beta_occ( double[] beta_occ ) {
    if (beta_occ == this.beta_occ) {
      return;
    }
    double[] _oldValue_xjal = this.beta_occ;
    this.beta_occ = beta_occ;
    onChange_beta_occ_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter beta_occ.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_beta_occ()</code> method instead.
   */
  void onChange_beta_occ() {
    onChange_beta_occ_xjal( beta_occ );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_beta_occ_xjal( double[] oldValue ) {
  }


  public
double[]  beta_dir;

  /**
   * Возвращает значение по умолчанию параметра <code>beta_dir</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
```

```
  @AnyLogicInternalCodegenAPI
  public double[] _beta_dir_DefaultValue_xjal() {
    final Main self = this;
    return new double[]
{1,1,1,1}
;
  }

  public void set_beta_dir( double[] beta_dir ) {
    if (beta_dir == this.beta_dir) {
      return;
    }
    double[] _oldValue_xjal = this.beta_dir;
    this.beta_dir = beta_dir;
    onChange_beta_dir_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter beta_dir.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_beta_dir()</code> method instead.
   */
  void onChange_beta_dir() {
    onChange_beta_dir_xjal( beta_dir );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_beta_dir_xjal( double[] oldValue ) {
  }


  public
double[]  beta_dest;

  /**
   * Возвращает значение по умолчанию параметра <code>beta_dest</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double[] _beta_dest_DefaultValue_xjal() {
    final Main self = this;
    return new double[]
{1,1,1,1}
;
  }

  public void set_beta_dest( double[] beta_dest ) {
    if (beta_dest == this.beta_dest) {
      return;
    }
    double[] _oldValue_xjal = this.beta_dest;
    this.beta_dest = beta_dest;
    onChange_beta_dest_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter beta_dest.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_beta_dest()</code> method instead.
   */
  void onChange_beta_dest() {
    onChange_beta_dest_xjal( beta_dest );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_beta_dest_xjal( double[] oldValue ) {
  }


  public
double[]  beta_angle;

  /**
   * Возвращает значение по умолчанию параметра <code>beta_angle</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double[] _beta_angle_DefaultValue_xjal() {
```

```
    final Main self = this;
    return new double[]
{1,1,1,1}
;
  }

  public void set_beta_angle( double[] beta_angle ) {
    if (beta_angle == this.beta_angle) {
      return;
    }
    double[] _oldValue_xjal = this.beta_angle;
    this.beta_angle = beta_angle;
    onChange_beta_angle_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter beta_angle.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_beta_angle()</code> method instead.
   */
  void onChange_beta_angle() {
    onChange_beta_angle_xjal( beta_angle );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_beta_angle_xjal( double[] oldValue ) {
  }


  public
double[]  beta_wall;

  /**
   * Возвращает значение по умолчанию параметра <code>beta_wall</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double[] _beta_wall_DefaultValue_xjal() {
    final Main self = this;
    return new double[]
{1,1,1,1}
;
  }

  public void set_beta_wall( double[] beta_wall ) {
    if (beta_wall == this.beta_wall) {
      return;
    }
    double[] _oldValue_xjal = this.beta_wall;
    this.beta_wall = beta_wall;
    onChange_beta_wall_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter beta_wall.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_beta_wall()</code> method instead.
   */
  void onChange_beta_wall() {
    onChange_beta_wall_xjal( beta_wall );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_beta_wall_xjal( double[] oldValue ) {
  }


  public
double[]  beta_vel;

  /**
   * Возвращает значение по умолчанию параметра <code>beta_vel</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double[] _beta_vel_DefaultValue_xjal() {
    final Main self = this;
    return new double[]
```

```
{1,1,1,1}
;
  }

  public void set_beta_vel( double[] beta_vel ) {
    if (beta_vel == this.beta_vel) {
      return;
    }
    double[] _oldValue_xjal = this.beta_vel;
    this.beta_vel = beta_vel;
    onChange_beta_vel_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter beta_vel.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_beta_vel()</code> method instead.
   */
  void onChange_beta_vel() {
    onChange_beta_vel_xjal( beta_vel );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_beta_vel_xjal( double[] oldValue ) {
  }


  public
double[]  beta_max;

  /**
   * Возвращает значение по умолчанию параметра <code>beta_max</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double[] _beta_max_DefaultValue_xjal() {
    final Main self = this;
    return new double[]
{1,1,1,1}
;
  }

  public void set_beta_max( double[] beta_max ) {
    if (beta_max == this.beta_max) {
      return;
    }
    double[] _oldValue_xjal = this.beta_max;
    this.beta_max = beta_max;
    onChange_beta_max_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter beta_max.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_beta_max()</code> method instead.
   */
  void onChange_beta_max() {
    onChange_beta_max_xjal( beta_max );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_beta_max_xjal( double[] oldValue ) {
  }


  public
double[]  beta_avg;

  /**
   * Возвращает значение по умолчанию параметра <code>beta_avg</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double[] _beta_avg_DefaultValue_xjal() {
    final Main self = this;
    return new double[]
{1,1,1,1}
;
```

```
3870    }

        public void set_beta_avg( double[] beta_avg ) {
          if (beta_avg == this.beta_avg) {
            return;
          }
          double[] _oldValue_xjal = this.beta_avg;
          this.beta_avg = beta_avg;
          onChange_beta_avg_xjal( _oldValue_xjal );
          onChange();
3880    }

        /**
         * Calls "On change" action for parameter beta_avg.<br>
         * Note that 'oldValue' in that action will be unavailable if this method is called by user
3885     * (current parameter value will be passed as 'oldValue').<br>
         * Please call <code>set_beta_avg()</code> method instead.
         */
        void onChange_beta_avg() {
          onChange_beta_avg_xjal( beta_avg );
3890    }

        @AnyLogicInternalCodegenAPI
        void onChange_beta_avg_xjal( double[] oldValue ) {
        }
3895

        public
    double[]  beta_expl;

3900    /**
         * Возвращает значение по умолчанию параметра <code>beta_expl</code>.
         * <i>Пользователь не должен вызывать этот метод</i>
         */
        @AnyLogicInternalCodegenAPI
3905    public double[] _beta_expl_DefaultValue_xjal() {
          final Main self = this;
          return new double[]
    {1,1,1,1}
    ;
3910    }

        public void set_beta_expl( double[] beta_expl ) {
          if (beta_expl == this.beta_expl) {
            return;
3915      }
          double[] _oldValue_xjal = this.beta_expl;
          this.beta_expl = beta_expl;
          onChange_beta_expl_xjal( _oldValue_xjal );
          onChange();
3920    }

        /**
         * Calls "On change" action for parameter beta_expl.<br>
         * Note that 'oldValue' in that action will be unavailable if this method is called by user
3925     * (current parameter value will be passed as 'oldValue').<br>
         * Please call <code>set_beta_expl()</code> method instead.
         */
        void onChange_beta_expl() {
          onChange_beta_expl_xjal( beta_expl );
3930    }

        @AnyLogicInternalCodegenAPI
        void onChange_beta_expl_xjal( double[] oldValue ) {
        }
3935

        /**
         * Массив коэффициента близости к стене
         */
3940    public
    double[]  wallCoef;

        /**
         * Возвращает значение по умолчанию параметра <code>wallCoef</code>.
3945     * <i>Пользователь не должен вызывать этот метод</i>
         */
        @AnyLogicInternalCodegenAPI
        public double[] _wallCoef_DefaultValue_xjal() {
          final Main self = this;
3950      return new double[]
    {1.2,1,0.8,0.7}
```

```
;
  }

  public void set_wallCoef( double[] wallCoef ) {
    if (wallCoef == this.wallCoef) {
      return;
    }
    double[] _oldValue_xjal = this.wallCoef;
    this.wallCoef = wallCoef;
    onChange_wallCoef_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter wallCoef.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_wallCoef()</code> method instead.
   */
  void onChange_wallCoef() {
    onChange_wallCoef_xjal( wallCoef );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_wallCoef_xjal( double[] oldValue ) {
  }


  /**
   * Максимальная площадь горизонтальной проекции
   */
  public
double  maxArea;

  /**
   * Возвращает значение по умолчанию параметра <code>maxArea</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _maxArea_DefaultValue_xjal() {
    final Main self = this;
    return
0.13
;
  }

  public void set_maxArea( double maxArea ) {
    if (maxArea == this.maxArea) {
      return;
    }
    double _oldValue_xjal = this.maxArea;
    this.maxArea = maxArea;
    onChange_maxArea_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter maxArea.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_maxArea()</code> method instead.
   */
  void onChange_maxArea() {
    onChange_maxArea_xjal( maxArea );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_maxArea_xjal( double oldValue ) {
  }


  @Override
  public void setParametersToDefaultValues() {
    super.setParametersToDefaultValues();
    len1 = _len1_DefaultValue_xjal();
    len2 = _len2_DefaultValue_xjal();
    a0 = _a0_DefaultValue_xjal();
    b0 = _b0_DefaultValue_xjal();
    a11 = _a11_DefaultValue_xjal();
    b11 = _b11_DefaultValue_xjal();
    a12 = _a12_DefaultValue_xjal();
    b12 = _b12_DefaultValue_xjal();
```

```
    a21 = _a21_DefaultValue_xjal();
    b21 = _b21_DefaultValue_xjal();
    a22 = _a22_DefaultValue_xjal();
    b22 = _b22_DefaultValue_xjal();
    m_vert = _m_vert_DefaultValue_xjal();
    m_hor = _m_hor_DefaultValue_xjal();
    distr = _distr_DefaultValue_xjal();
    b_lapl_x = _b_lapl_x_DefaultValue_xjal();
    a_lapl_x = _a_lapl_x_DefaultValue_xjal();
    mu_norm_x = _mu_norm_x_DefaultValue_xjal();
    sig_norm_x = _sig_norm_x_DefaultValue_xjal();
    c_trian_x = _c_trian_x_DefaultValue_xjal();
    agents_count = _agents_count_DefaultValue_xjal();
    global_distr = _global_distr_DefaultValue_xjal();
    sigma_1 = _sigma_1_DefaultValue_xjal();
    sigma_2 = _sigma_2_DefaultValue_xjal();
    eta = _eta_DefaultValue_xjal();
    gamma_acc = _gamma_acc_DefaultValue_xjal();
    gamma_dec = _gamma_dec_DefaultValue_xjal();
    gamma_exit = _gamma_exit_DefaultValue_xjal();
    varepsilon_0 = _varepsilon_0_DefaultValue_xjal();
    varepsilon_1 = _varepsilon_1_DefaultValue_xjal();
    varepsilon_2 = _varepsilon_2_DefaultValue_xjal();
    zeta = _zeta_DefaultValue_xjal();
    kappa_1 = _kappa_1_DefaultValue_xjal();
    kappa_2 = _kappa_2_DefaultValue_xjal();
    kappa_3 = _kappa_3_DefaultValue_xjal();
    gamma_1 = _gamma_1_DefaultValue_xjal();
    gamma_2 = _gamma_2_DefaultValue_xjal();
    gamma_3 = _gamma_3_DefaultValue_xjal();
    gamma_4 = _gamma_4_DefaultValue_xjal();
    lambda_vel = _lambda_vel_DefaultValue_xjal();
    lambda_max = _lambda_max_DefaultValue_xjal();
    distr_expl = _distr_expl_DefaultValue_xjal();
    b_lapl_expl_x = _b_lapl_expl_x_DefaultValue_xjal();
    a_lapl_expl_x = _a_lapl_expl_x_DefaultValue_xjal();
    mu_norm_expl_x = _mu_norm_expl_x_DefaultValue_xjal();
    sig_norm_expl_x = _sig_norm_expl_x_DefaultValue_xjal();
    c_trian_expl_x = _c_trian_expl_x_DefaultValue_xjal();
    distr_exit = _distr_exit_DefaultValue_xjal();
    distr_turb = _distr_turb_DefaultValue_xjal();
    turb = _turb_DefaultValue_xjal();
    b_lapl_y = _b_lapl_y_DefaultValue_xjal();
    a_lapl_y = _a_lapl_y_DefaultValue_xjal();
    mu_norm_y = _mu_norm_y_DefaultValue_xjal();
    sig_norm_y = _sig_norm_y_DefaultValue_xjal();
    c_trian_y = _c_trian_y_DefaultValue_xjal();
    b_lapl_expl_y = _b_lapl_expl_y_DefaultValue_xjal();
    a_lapl_expl_y = _a_lapl_expl_y_DefaultValue_xjal();
    mu_norm_expl_y = _mu_norm_expl_y_DefaultValue_xjal();
    sig_norm_expl_y = _sig_norm_expl_y_DefaultValue_xjal();
    c_trian_expl_y = _c_trian_expl_y_DefaultValue_xjal();
    version = _version_DefaultValue_xjal();
    lambda_avg = _lambda_avg_DefaultValue_xjal();
    gamma_5 = _gamma_5_DefaultValue_xjal();
    model_second = _model_second_DefaultValue_xjal();
    varsigma = _varsigma_DefaultValue_xjal();
    varrho_3 = _varrho_3_DefaultValue_xjal();
    varrho_4 = _varrho_4_DefaultValue_xjal();
    vartheta = _vartheta_DefaultValue_xjal();
    theta = _theta_DefaultValue_xjal();
    theta_hat = _theta_hat_DefaultValue_xjal();
    global_distr_array = _global_distr_array_DefaultValue_xjal();
    width = _width_DefaultValue_xjal();
    height = _height_DefaultValue_xjal();
    beta_occ = _beta_occ_DefaultValue_xjal();
    beta_dir = _beta_dir_DefaultValue_xjal();
    beta_dest = _beta_dest_DefaultValue_xjal();
    beta_angle = _beta_angle_DefaultValue_xjal();
    beta_wall = _beta_wall_DefaultValue_xjal();
    beta_vel = _beta_vel_DefaultValue_xjal();
    beta_max = _beta_max_DefaultValue_xjal();
    beta_avg = _beta_avg_DefaultValue_xjal();
    beta_expl = _beta_expl_DefaultValue_xjal();
    wallCoef = _wallCoef_DefaultValue_xjal();
    maxArea = _maxArea_DefaultValue_xjal();
  }

  @Override
  public boolean setParameter(String _name_xjal, Object _value_xjal, boolean _callOnChange_xjal) {
    if ("len1".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
```

```
        set_len1( ((Number) _value_xjal).doubleValue() );
      } else {
        len1 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("len2".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_len2( ((Number) _value_xjal).doubleValue() );
      } else {
        len2 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("a0".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_a0( ((Number) _value_xjal).doubleValue() );
      } else {
        a0 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("b0".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_b0( ((Number) _value_xjal).doubleValue() );
      } else {
        b0 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("a11".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_a11( ((Number) _value_xjal).doubleValue() );
      } else {
        a11 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("b11".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_b11( ((Number) _value_xjal).doubleValue() );
      } else {
        b11 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("a12".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_a12( ((Number) _value_xjal).doubleValue() );
      } else {
        a12 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("b12".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_b12( ((Number) _value_xjal).doubleValue() );
      } else {
        b12 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("a21".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_a21( ((Number) _value_xjal).doubleValue() );
      } else {
        a21 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("b21".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_b21( ((Number) _value_xjal).doubleValue() );
      } else {
        b21 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("a22".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_a22( ((Number) _value_xjal).doubleValue() );
      } else {
```

```
        a22 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("b22".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_b22( ((Number) _value_xjal).doubleValue() );
      } else {
        b22 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("m_vert".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_m_vert( ((Number) _value_xjal).intValue() );
      } else {
        m_vert = ((Number) _value_xjal).intValue();
      }
      return true;
    }
    if ("m_hor".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_m_hor( ((Number) _value_xjal).intValue() );
      } else {
        m_hor = ((Number) _value_xjal).intValue();
      }
      return true;
    }
    if ("distr".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_distr( (String) _value_xjal );
      } else {
        distr = (String) _value_xjal;
      }
      return true;
    }
    if ("b_lapl_x".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_b_lapl_x( ((Number) _value_xjal).doubleValue() );
      } else {
        b_lapl_x = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("a_lapl_x".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_a_lapl_x( ((Number) _value_xjal).doubleValue() );
      } else {
        a_lapl_x = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("mu_norm_x".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_mu_norm_x( ((Number) _value_xjal).doubleValue() );
      } else {
        mu_norm_x = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("sig_norm_x".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_sig_norm_x( ((Number) _value_xjal).doubleValue() );
      } else {
        sig_norm_x = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("c_trian_x".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_c_trian_x( ((Number) _value_xjal).doubleValue() );
      } else {
        c_trian_x = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("agents_count".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_agents_count( ((Number) _value_xjal).intValue() );
      } else {
        agents_count = ((Number) _value_xjal).intValue();
      }
```

```
4280        return true;
          }
          if ("global_distr".equals( _name_xjal )) {
            if ( _callOnChange_xjal ) {
              set_global_distr( (Boolean) _value_xjal );
4285        } else {
              global_distr = (Boolean) _value_xjal;
            }
            return true;
          }
4290      if ("sigma_1".equals( _name_xjal )) {
            if ( _callOnChange_xjal ) {
              set_sigma_1( ((Number) _value_xjal).doubleValue() );
            } else {
              sigma_1 = ((Number) _value_xjal).doubleValue();
4295        }
            return true;
          }
          if ("sigma_2".equals( _name_xjal )) {
            if ( _callOnChange_xjal ) {
4300          set_sigma_2( ((Number) _value_xjal).doubleValue() );
            } else {
              sigma_2 = ((Number) _value_xjal).doubleValue();
            }
            return true;
4305      }
          if ("eta".equals( _name_xjal )) {
            if ( _callOnChange_xjal ) {
              set_eta( ((Number) _value_xjal).doubleValue() );
            } else {
4310          eta = ((Number) _value_xjal).doubleValue();
            }
            return true;
          }
          if ("gamma_acc".equals( _name_xjal )) {
4315        if ( _callOnChange_xjal ) {
              set_gamma_acc( ((Number) _value_xjal).doubleValue() );
            } else {
              gamma_acc = ((Number) _value_xjal).doubleValue();
            }
4320        return true;
          }
          if ("gamma_dec".equals( _name_xjal )) {
            if ( _callOnChange_xjal ) {
              set_gamma_dec( ((Number) _value_xjal).doubleValue() );
4325        } else {
              gamma_dec = ((Number) _value_xjal).doubleValue();
            }
            return true;
          }
4330      if ("gamma_exit".equals( _name_xjal )) {
            if ( _callOnChange_xjal ) {
              set_gamma_exit( ((Number) _value_xjal).doubleValue() );
            } else {
              gamma_exit = ((Number) _value_xjal).doubleValue();
4335        }
            return true;
          }
          if ("varepsilon_0".equals( _name_xjal )) {
            if ( _callOnChange_xjal ) {
4340          set_varepsilon_0( ((Number) _value_xjal).doubleValue() );
            } else {
              varepsilon_0 = ((Number) _value_xjal).doubleValue();
            }
            return true;
4345      }
          if ("varepsilon_1".equals( _name_xjal )) {
            if ( _callOnChange_xjal ) {
              set_varepsilon_1( ((Number) _value_xjal).doubleValue() );
            } else {
4350          varepsilon_1 = ((Number) _value_xjal).doubleValue();
            }
            return true;
          }
          if ("varepsilon_2".equals( _name_xjal )) {
4355        if ( _callOnChange_xjal ) {
              set_varepsilon_2( ((Number) _value_xjal).doubleValue() );
            } else {
              varepsilon_2 = ((Number) _value_xjal).doubleValue();
            }
4360        return true;
          }
```

```
    if ("zeta".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_zeta( ((Number) _value_xjal).intValue() );
      } else {
        zeta = ((Number) _value_xjal).intValue();
      }
      return true;
    }
    if ("kappa_1".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_kappa_1( ((Number) _value_xjal).doubleValue() );
      } else {
        kappa_1 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("kappa_2".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_kappa_2( ((Number) _value_xjal).doubleValue() );
      } else {
        kappa_2 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("kappa_3".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_kappa_3( ((Number) _value_xjal).doubleValue() );
      } else {
        kappa_3 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("gamma_1".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_gamma_1( ((Number) _value_xjal).doubleValue() );
      } else {
        gamma_1 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("gamma_2".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_gamma_2( ((Number) _value_xjal).doubleValue() );
      } else {
        gamma_2 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("gamma_3".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_gamma_3( ((Number) _value_xjal).doubleValue() );
      } else {
        gamma_3 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("gamma_4".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_gamma_4( ((Number) _value_xjal).doubleValue() );
      } else {
        gamma_4 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("lambda_vel".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_lambda_vel( ((Number) _value_xjal).doubleValue() );
      } else {
        lambda_vel = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("lambda_max".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_lambda_max( ((Number) _value_xjal).doubleValue() );
      } else {
        lambda_max = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("distr_expl".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
```

```
        set_distr_expl( (String) _value_xjal );
      } else {
        distr_expl = (String) _value_xjal;
      }
      return true;
    }
    if ("b_lapl_expl_x".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_b_lapl_expl_x( ((Number) _value_xjal).doubleValue() );
      } else {
        b_lapl_expl_x = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("a_lapl_expl_x".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_a_lapl_expl_x( ((Number) _value_xjal).doubleValue() );
      } else {
        a_lapl_expl_x = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("mu_norm_expl_x".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_mu_norm_expl_x( ((Number) _value_xjal).doubleValue() );
      } else {
        mu_norm_expl_x = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("sig_norm_expl_x".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_sig_norm_expl_x( ((Number) _value_xjal).doubleValue() );
      } else {
        sig_norm_expl_x = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("c_trian_expl_x".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_c_trian_expl_x( ((Number) _value_xjal).doubleValue() );
      } else {
        c_trian_expl_x = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("distr_exit".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_distr_exit( (String) _value_xjal );
      } else {
        distr_exit = (String) _value_xjal;
      }
      return true;
    }
    if ("distr_turb".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_distr_turb( (String) _value_xjal );
      } else {
        distr_turb = (String) _value_xjal;
      }
      return true;
    }
    if ("turb".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_turb( ((Number) _value_xjal).intValue() );
      } else {
        turb = ((Number) _value_xjal).intValue();
      }
      return true;
    }
    if ("b_lapl_y".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_b_lapl_y( ((Number) _value_xjal).doubleValue() );
      } else {
        b_lapl_y = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("a_lapl_y".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_a_lapl_y( ((Number) _value_xjal).doubleValue() );
      } else {
```

```
        a_lapl_y = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("mu_norm_y".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_mu_norm_y( ((Number) _value_xjal).doubleValue() );
      } else {
        mu_norm_y = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("sig_norm_y".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_sig_norm_y( ((Number) _value_xjal).doubleValue() );
      } else {
        sig_norm_y = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("c_trian_y".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_c_trian_y( ((Number) _value_xjal).doubleValue() );
      } else {
        c_trian_y = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("b_lapl_expl_y".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_b_lapl_expl_y( ((Number) _value_xjal).doubleValue() );
      } else {
        b_lapl_expl_y = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("a_lapl_expl_y".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_a_lapl_expl_y( ((Number) _value_xjal).doubleValue() );
      } else {
        a_lapl_expl_y = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("mu_norm_expl_y".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_mu_norm_expl_y( ((Number) _value_xjal).doubleValue() );
      } else {
        mu_norm_expl_y = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("sig_norm_expl_y".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_sig_norm_expl_y( ((Number) _value_xjal).doubleValue() );
      } else {
        sig_norm_expl_y = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("c_trian_expl_y".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_c_trian_expl_y( ((Number) _value_xjal).doubleValue() );
      } else {
        c_trian_expl_y = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("version".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_version( ((Number) _value_xjal).intValue() );
      } else {
        version = ((Number) _value_xjal).intValue();
      }
      return true;
    }
    if ("lambda_avg".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_lambda_avg( ((Number) _value_xjal).doubleValue() );
      } else {
        lambda_avg = ((Number) _value_xjal).doubleValue();
      }
```

```
      return true;
    }
    if ("gamma_5".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_gamma_5( ((Number) _value_xjal).doubleValue() );
      } else {
        gamma_5 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("model_second".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_model_second( ((Number) _value_xjal).doubleValue() );
      } else {
        model_second = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("varsigma".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_varsigma( (double[]) _value_xjal );
      } else {
        varsigma = (double[]) _value_xjal;
      }
      return true;
    }
    if ("varrho_3".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_varrho_3( (double[]) _value_xjal );
      } else {
        varrho_3 = (double[]) _value_xjal;
      }
      return true;
    }
    if ("varrho_4".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_varrho_4( (double[]) _value_xjal );
      } else {
        varrho_4 = (double[]) _value_xjal;
      }
      return true;
    }
    if ("vartheta".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_vartheta( (double[]) _value_xjal );
      } else {
        vartheta = (double[]) _value_xjal;
      }
      return true;
    }
    if ("theta".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_theta( (int[]) _value_xjal );
      } else {
        theta = (int[]) _value_xjal;
      }
      return true;
    }
    if ("theta_hat".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_theta_hat( (int[]) _value_xjal );
      } else {
        theta_hat = (int[]) _value_xjal;
      }
      return true;
    }
    if ("global_distr_array".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_global_distr_array( (int[]) _value_xjal );
      } else {
        global_distr_array = (int[]) _value_xjal;
      }
      return true;
    }
    if ("width".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_width( ((Number) _value_xjal).doubleValue() );
      } else {
        width = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
```

```
4690    if ("height".equals( _name_xjal )) {
          if ( _callOnChange_xjal ) {
            set_height( ((Number) _value_xjal).doubleValue() );
          } else {
            height = ((Number) _value_xjal).doubleValue();
4695      }
          return true;
        }
        if ("beta_occ".equals( _name_xjal )) {
          if ( _callOnChange_xjal ) {
4700        set_beta_occ( (double[]) _value_xjal );
          } else {
            beta_occ = (double[]) _value_xjal;
          }
          return true;
4705    }
        if ("beta_dir".equals( _name_xjal )) {
          if ( _callOnChange_xjal ) {
            set_beta_dir( (double[]) _value_xjal );
          } else {
4710        beta_dir = (double[]) _value_xjal;
          }
          return true;
        }
        if ("beta_dest".equals( _name_xjal )) {
4715      if ( _callOnChange_xjal ) {
            set_beta_dest( (double[]) _value_xjal );
          } else {
            beta_dest = (double[]) _value_xjal;
          }
4720      return true;
        }
        if ("beta_angle".equals( _name_xjal )) {
          if ( _callOnChange_xjal ) {
            set_beta_angle( (double[]) _value_xjal );
4725      } else {
            beta_angle = (double[]) _value_xjal;
          }
          return true;
        }
4730    if ("beta_wall".equals( _name_xjal )) {
          if ( _callOnChange_xjal ) {
            set_beta_wall( (double[]) _value_xjal );
          } else {
            beta_wall = (double[]) _value_xjal;
4735      }
          return true;
        }
        if ("beta_vel".equals( _name_xjal )) {
          if ( _callOnChange_xjal ) {
4740        set_beta_vel( (double[]) _value_xjal );
          } else {
            beta_vel = (double[]) _value_xjal;
          }
          return true;
4745    }
        if ("beta_max".equals( _name_xjal )) {
          if ( _callOnChange_xjal ) {
            set_beta_max( (double[]) _value_xjal );
          } else {
4750        beta_max = (double[]) _value_xjal;
          }
          return true;
        }
        if ("beta_avg".equals( _name_xjal )) {
4755      if ( _callOnChange_xjal ) {
            set_beta_avg( (double[]) _value_xjal );
          } else {
            beta_avg = (double[]) _value_xjal;
          }
4760      return true;
        }
        if ("beta_expl".equals( _name_xjal )) {
          if ( _callOnChange_xjal ) {
            set_beta_expl( (double[]) _value_xjal );
4765      } else {
            beta_expl = (double[]) _value_xjal;
          }
          return true;
        }
4770    if ("wallCoef".equals( _name_xjal )) {
          if ( _callOnChange_xjal ) {
```

```
        set_wallCoef( (double[]) _value_xjal );
      } else {
        wallCoef = (double[]) _value_xjal;
      }
      return true;
    }
    if ("maxArea".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_maxArea( ((Number) _value_xjal).doubleValue() );
      } else {
        maxArea = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    return super.setParameter( _name_xjal, _value_xjal, _callOnChange_xjal );
  }
  // Простые переменные

  /**
   * Количество спасенных агентов
   */
  public
int
 saved;

  // Коллекции
  public
java.util.ArrayList <
Double > x_expl = new java.util.ArrayList<Double>();
  public
java.util.ArrayList <
Double > y_expl = new java.util.ArrayList<Double>();
  public
java.util.ArrayList <
Double > t_expl = new java.util.ArrayList<Double>();

  // Динамические переменные

  /**
   * Модельное время
   */
  public double t;
  @AnyLogicInternalCodegenAPI
  public void assignInitialConditions_xjal() {
  super.assignInitialConditions_xjal();
    _assign_t_Formula_xjal();
  }

  @AnyLogicInternalCodegenAPI
  public void setupInitialConditions_xjal(Class<?> callerClass) {
    if (callerClass != Main.class) {
      return;
    }
    if (getInitialAlgebraicFlatEquationsCount_xjal() > 0) {
      SDIntegrationManager integrationManagerForInitialConditions = new SDIntegrationManager( 0,
      getInitialAlgebraicFlatEquationsCount_xjal(), getInitialFormulaFlatEquationsCount_xjal() );
      integrationManagerForInitialConditions.doStep( this, 0, 0.1, true );
    } else {
      assignInitialConditions_xjal();
    }
  }

  @AnyLogicInternalCodegenAPI
  public void _assign_t_Formula_xjal() {
    t =
time()
;
  }
  @AnyLogicInternalCodegenAPI
  public void formulasExecute_xjal() {
    super.formulasExecute_xjal();
    _assign_t_Formula_xjal();
  }

  @AnyLogicInternalCodegenAPI
  protected SDIntegrationManager integrationManager_xjal = null;

  @AnyLogicInternalCodegenAPI
  public SDIntegrationManager getIntegrationManager_xjal() {
    if (integrationManager_xjal == null) {
      integrationManager_xjal = new SDIntegrationManager( getDifferentialFlatEquationsCount_xjal(),
      getRuntimeAlgebraicFlatEquationsCount_xjal(), getRuntimeFormulaFlatEquationsCount_xjal() );
```

```
    }
    return integrationManager_xjal;
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public int getRuntimeFormulaFlatEquationsCount_xjal() {
    return super.getRuntimeFormulaFlatEquationsCount_xjal() + 1;
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public int getInitialFormulaFlatEquationsCount_xjal() {
    return super.getInitialFormulaFlatEquationsCount_xjal() + 1;
  }

  @AnyLogicCustomProposalPriority(type = AnyLogicCustomProposalPriority.Type.STATIC_ELEMENT)
  public static final Scale scale = new Scale( 10.0 );

  @Override
  public Scale getScale() {
    return scale;
  }
  // События

  public EventTimeout start = new EventTimeout(this);
  public EventTimeout line_clean = new EventTimeout(this);
  public EventTimeout coordinates = new EventTimeout(this);
  public EventCondition pause = new EventCondition(this);
  @AnyLogicInternalCodegenAPI
  public EventTimeout _plot_st_autoUpdateEvent_xjal = new EventTimeout(this);
  @AnyLogicInternalCodegenAPI
  public EventTimeout _plot_saved_autoUpdateEvent_xjal = new EventTimeout(this);
  @AnyLogicInternalCodegenAPI
  public EventTimeout _autoCreatedDS_xjal = new EventTimeout(this);


  @Override
  @AnyLogicInternalCodegenAPI
  public String getNameOf( EventTimeout _e ) {
     if( _e == start ) return "start";
     if( _e == line_clean ) return "line_clean";
     if( _e == coordinates ) return "coordinates";
    if ( _e == _plot_st_autoUpdateEvent_xjal ) return "plot_st auto update event";
    if ( _e == _plot_saved_autoUpdateEvent_xjal ) return "plot_saved auto update event";
    if ( _e == _autoCreatedDS_xjal ) return "Auto-created DataSets auto update event";
    return super.getNameOf( _e );
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public EventTimeout.Mode getModeOf( EventTimeout _e ) {
    if ( _e == start ) return EVENT_TIMEOUT_MODE_ONCE;
    if ( _e == line_clean ) return EVENT_TIMEOUT_MODE_ONCE;
    if ( _e == coordinates ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _plot_st_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _plot_saved_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _autoCreatedDS_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    return super.getModeOf( _e );
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public double getFirstOccurrenceTime( EventTimeout _e ) {
    if ( _e == start ) return
0
;
    if ( _e == line_clean ) return
2*model_second
;
    if ( _e == coordinates ) return
0
;
    if (
        _e == _plot_st_autoUpdateEvent_xjal
        || _e == _plot_saved_autoUpdateEvent_xjal
        || _e == _autoCreatedDS_xjal
      ) return getEngine().getStartTime();
    return super.getFirstOccurrenceTime( _e );
  }

  @Override
```

```
  @AnyLogicInternalCodegenAPI
  public double evaluateTimeoutOf( EventTimeout _e ) {
    if( _e == coordinates) return
1
;
    if ( _e == _plot_st_autoUpdateEvent_xjal ) return
0.1
;
    if ( _e == _plot_saved_autoUpdateEvent_xjal ) return
1
;
    if ( _e == _autoCreatedDS_xjal )  return
1
;
    return super.evaluateTimeoutOf( _e );
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void executeActionOf( EventTimeout _e ) {
    if ( _e == start ) {
      EventTimeout self = _e;

insert_version.execute();
insert_functional.execute();
;
      return;
    }
    if ( _e == line_clean ) {
      EventTimeout self = _e;

LineClean();
;
      return;
    }
    if ( _e == coordinates ) {
      EventTimeout self = _e;
      return;
    }
    if ( _e == _plot_st_autoUpdateEvent_xjal ) {
      plot_st.updateData();
      return;
    }
    if ( _e == _plot_saved_autoUpdateEvent_xjal ) {
      plot_saved.updateData();
      return;
    }
    if ( _e == _autoCreatedDS_xjal ) {
      _ds_t.update();
      return;
    }
    super.executeActionOf( _e );
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public String getNameOf( EventCondition _e ) {
    if ( _e == pause ) return "pause";
    return super.getNameOf( _e );
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public boolean testConditionOf( EventCondition _e ) {

    if ( _e == pause) return
((time()>0)&&(person.size()==0))||(allKilled()==person.size())
;
    return super.testConditionOf( _e );
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void executeActionOf( EventCondition _e ) {
    if ( _e == pause) {
      EventCondition self = _e;

pauseSimulation();
;
      return ;
    }
    super.executeActionOf( _e );
```

```
}


/** Internal constant, shouldn't be accessed by user */
@AnyLogicInternalCodegenAPI
protected static final short _STATECHART_ELEMENT_NEXT_ID_xjal = 0;
// Вложенные объекты


public String getNameOf( Agent ao ) {
  return super.getNameOf( ao );
}

public class _explosion_Class extends AgentArrayList<Explosion> {
  _explosion_Class( Agent owner ) {
    super( owner );
  }

  @AnyLogicInternalCodegenAPI
  public void callSetupParameters( Explosion agent, int index ) {
    setupParameters_explosion_xjal( agent, index );
  }

  @AnyLogicInternalCodegenAPI
  public void callCreate( Explosion agent, int index ) {
    create_explosion_xjal( agent, index );
  }

  /**
   * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
   */
  @AnyLogicInternalCodegenAPI
  private static final long serialVersionUID = -9043181999309756418L;
}

public _explosion_Class explosion = new _explosion_Class( this );
public class _person_Class extends AgentArrayList<Person> {
  _person_Class( Agent owner ) {
    super( owner );
  }

  @AnyLogicInternalCodegenAPI
  public void callSetupParameters( Person agent, int index ) {
    setupParameters_person_xjal( agent, index );
  }

  @AnyLogicInternalCodegenAPI
  public void callCreate( Person agent, int index ) {
    create_person_xjal( agent, index );
  }

  /**
   * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
   */
  @AnyLogicInternalCodegenAPI
  private static final long serialVersionUID = -9043181998977681477L;
}

public _person_Class person = new _person_Class( this );

public String getNameOf( AgentList<?> aolist ) {
  if( aolist == explosion ) return "explosion";
  if( aolist == person ) return "person";
  return super.getNameOf( aolist );
}

/**
 * Этот метод создает и добавляет новый вложенный объект в список экземпляров реплицированного объектов explosion<br>
 * @return только что созданный вложенный объект
 */
public Explosion add_explosion() {
  int index = explosion.size();
  Explosion _result_xjal = instantiate_explosion_xjal( index );
  explosion.callSetupParameters( _result_xjal, index );
  explosion.callCreate( _result_xjal, index );
  _result_xjal.start();
  return _result_xjal;
}

/**
 * Этот метод создает и добавляет новый вложенный объект в список экземпляров реплицированного объектов explosion<br>
 * Этот метод использует заданные значения параметров для инициализации созданного вложенного объекта<br>
```

```
     * Индекс нового экземпляра реплицированного объекта может быть получен с помощью метода <code>explosion.size()</code>  <strong>перед</
        strong> вызовом этого метода
     * @param epsilon0
     * @param epsilon1
     * @param epsilon2
     * @return только что созданный вложенный объект
     */
    public Explosion add_explosion( double epsilon0, double epsilon1, double epsilon2 ) {
      int index = explosion.size();
      Explosion _result_xjal = instantiate_explosion_xjal( index );
      // Задание значений параметров
      _result_xjal.markParametersAreSet();
      _result_xjal.epsilon0 = epsilon0;
      _result_xjal.epsilon1 = epsilon1;
      _result_xjal.epsilon2 = epsilon2;
      // Завершение создания вложенного объекта
      explosion.callCreate( _result_xjal, index );
      _result_xjal.start();
      return _result_xjal;
    }

    /**
     * Этот метод удаляет заданный вложенный объект из коллекции экземпляров реплицированного объекта explosion<br>
     * Данный объект будет уничтожен, но в общем случае это произойдет позже.
     * @param object активный объект — элемент реплицированного вложенного объекта explosion, который должен быть удален
     * @return <code>true</code>, если объект успешно удален, <code>false</code>, если он не принадлежит explosion
     */
    public boolean remove_explosion( Explosion object ) {
      if( ! explosion._remove( object ) ) {
        return false;
      }
      object.setDestroyed();
      return true;
    }
    /**
     * Этот метод создает и добавляет новый вложенный объект в список экземпляров реплицированного объектов person<br>
     * @return только что созданный вложенный объект
     */
    public Person add_person() {
      int index = person.size();
      Person _result_xjal = instantiate_person_xjal( index );
      person.callSetupParameters( _result_xjal, index );
      person.callCreate( _result_xjal, index );
      _result_xjal.start();
      return _result_xjal;
    }

    /**
     * Этот метод создает и добавляет новый вложенный объект в список экземпляров реплицированного объектов person<br>
     * Этот метод использует заданные значения параметров для инициализации созданного вложенного объекта<br>
     * Индекс нового экземпляра реплицированного объекта может быть получен с помощью метода <code>person.size()</code>  <strong>перед</
        strong> вызовом этого метода
     * @param age
     * @param gender
     * @param v_comf
     * @param v_max
     * @param v_hat_max
     * @param status
     * @param situation
     * @param exit
     * @param expl
     * @param status_change
     * @param s
     * @param cross_exit
     * @return только что созданный вложенный объект
     */
    public Person add_person( int age, Gender gender, double v_comf, double v_max, double v_hat_max, Status status, Situation situation,
        Exit exit, boolean expl, boolean status_change, double s, boolean cross_exit ) {
      int index = person.size();
      Person _result_xjal = instantiate_person_xjal( index );
      // Задание значений параметров
      _result_xjal.markParametersAreSet();
      _result_xjal.age = age;
      _result_xjal.gender = gender;
      _result_xjal.v_comf = v_comf;
      _result_xjal.v_max = v_max;
      _result_xjal.v_hat_max = v_hat_max;
      _result_xjal.status = status;
      _result_xjal.situation = situation;
      _result_xjal.exit = exit;
      _result_xjal.expl = expl;
      _result_xjal.status_change = status_change;
```

```
    _result_xjal.s = s;
    _result_xjal.cross_exit = cross_exit;
    // Завершение создания вложенного объекта
    person.callCreate( _result_xjal, index );
    _result_xjal.start();
    return _result_xjal;
  }

  /**
   * Этот метод удаляет заданный вложенный объект из коллекции экземпляров реплицированного объекта person<br>
   * Данный объект будет уничтожен, но в общем случае это произойдет позже.
   * @param object активный объект — элемент реплицированного вложенного объекта person, который должен быть удален
   * @return <code>true</code>, если объект успешно удален, <code>false</code>, если он не принадлежит person
   */
  public boolean remove_person( Person object ) {
    if( ! person._remove( object ) ) {
      return false;
    }
    object.setDestroyed();
    return true;
  }

  /**
   * Создает экземпляр вложенного объекта и добавляет его в конец списка экземпляров реплицированных объектов<br>
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  protected Explosion instantiate_explosion_xjal( final int index ) {
    Explosion _result_xjal = new Explosion( getEngine(), this, explosion );

    explosion._add( _result_xjal );

    return _result_xjal;
  }

  /**
   * Инициализация параметров экземпляра вложенного объекта<br>
   * Пользователь не должен вызывать этот метод
   */
  private void setupParameters_explosion_xjal( final Explosion self, final int index ) {
    self.epsilon0 =
scale.toPixels(varepsilon_0, LENGTH_UNIT_METER)
;
    self.epsilon1 =
scale.toPixels(varepsilon_1, LENGTH_UNIT_METER)
;
    self.epsilon2 =
scale.toPixels(varepsilon_2, LENGTH_UNIT_METER)
;
  }

  /**
   * Инициализация экземпляра вложенного объекта<br>
   * Пользователь не должен вызывать этот метод
   */
  private void create_explosion_xjal(Explosion self, final int index ) {
    self.create();

    // Port connections
  }
  /**
   * Создает экземпляр вложенного объекта и добавляет его в конец списка экземпляров реплицированных объектов<br>
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  protected Person instantiate_person_xjal( final int index ) {
    Person _result_xjal = new Person( getEngine(), this, person );

    person._add( _result_xjal );

    return _result_xjal;
  }

  /**
   * Инициализация параметров экземпляра вложенного объекта<br>
   * Пользователь не должен вызывать этот метод
   */
  private void setupParameters_person_xjal( final Person self, final int index ) {
    self.age =
uniform_discr(6, 79)
;
    self.gender =
Gender.values()[bernoulli(0.5)]
;
```

```
    self.v_comf = self._v_comf_DefaultValue_xjal();
    self.v_max = self._v_max_DefaultValue_xjal();
    self.v_hat_max = self._v_hat_max_DefaultValue_xjal();
    self.status = self._status_DefaultValue_xjal();
    self.situation = self._situation_DefaultValue_xjal();
    self.exit = self._exit_DefaultValue_xjal();
    self.expl = self._expl_DefaultValue_xjal();
    self.status_change = self._status_change_DefaultValue_xjal();
    self.s = self._s_DefaultValue_xjal();
    self.cross_exit = self._cross_exit_DefaultValue_xjal();
  }

  /**
   * Инициализация экземпляра вложенного объекта<br>
   * Пользователь не должен вызывать этот метод
   */
  private void create_person_xjal(Person self, final int index ) {
    self.setEnvironment( this );
    self.create();

    // Port connections
  }

  // Функции

  /**
   * Функция расчета параметров области моделирования
   */
  void Area(  ) {

viewArea_base.setWidth(width);
viewArea_base.setHeight(height);
rectangle_base.setWidth(width);
rectangle_base.setHeight(height);
viewArea_base.navigateTo();
  }

  /**
   * Функция отрисовки стен
   */
  void myWall(  ) {

Wall myWallUp;
Wall myWallDown;
myWallUp = new Wall(this, SHAPE_DRAW_2D3D, true, WALL_FILL_NONE, black, 2.0, 50.0, SegmentUp());
myWallDown = new Wall(this, SHAPE_DRAW_2D3D, true, WALL_FILL_NONE, black, 2.0, 50.0, SegmentDown());
myWallUp.initialize();
myWallDown.initialize();
myWallUp.setVisible(true);
myWallDown.setVisible(true);
presentation.add(myWallUp);
presentation.add(myWallDown);
  }

  /**
   * Функция расчетов верхних сегментов стены
   */

MarkupSegment[]
 SegmentUp(  ) {

MarkupSegment[] MarkupSegmentUp;
if (b11==b0 && b21==b0){
MarkupSegmentUp = new MarkupSegment[] {
       new MarkupSegmentLine( a0, b0, 0.0, a21, b0, 0.0 )};
} else if (b11!=b0 && b21==b0){
MarkupSegmentUp = new MarkupSegment[] {
       new MarkupSegmentLine( a11, b11, 0.0, a0, b0, 0.0 ),
       new MarkupSegmentLine( a0, b0, 0.0, a21, b0, 0.0 )};
} else if (b11==b0 && b21!=b0){
MarkupSegmentUp = new MarkupSegment[] {
       new MarkupSegmentLine( a0, b0, 0.0, a21, b0, 0.0 ),
       new MarkupSegmentLine( a21, b0, 0.0, a21, b21, 0.0 )};
} else {
MarkupSegmentUp = new MarkupSegment[] {
       new MarkupSegmentLine( a11, b11, 0.0, a0, b0, 0.0 ),
       new MarkupSegmentLine( a0, b0, 0.0, a21, b0, 0.0 ),
       new MarkupSegmentLine( a21, b0, 0.0, a21, b21, 0.0 )};
}
return MarkupSegmentUp;
  }

```

```
  /**
   * Функция расчетов нижних сегментов стены
   */

MarkupSegment[]
 SegmentDown(  ) {

MarkupSegment[] MarkupSegmentDown;
if (b12==b0+len2 && b22==b0+len2) {
MarkupSegmentDown = new MarkupSegment[] {
       new MarkupSegmentLine( a22, b0+len2, 0.0, a12, b0+len2, 0.0 )};
} else if (b12!=b0+len2 && b22==b0+len2) {
MarkupSegmentDown = new MarkupSegment[] {
       new MarkupSegmentLine( a22, b0+len2, 0.0, a12, b0+len2, 0.0 ),
       new MarkupSegmentLine( a12, b0+len2, 0.0, a12, b12, 0.0 )};
} else if (b12==b0+len2 && b22!=b0+len2) {
MarkupSegmentDown = new MarkupSegment[] {
       new MarkupSegmentLine( a22, b22, 0.0, a22, b0+len2, 0.0 ),
       new MarkupSegmentLine( a22, b0+len2, 0.0, a12, b0+len2, 0.0 )};
} else {
MarkupSegmentDown = new MarkupSegment[] {
       new MarkupSegmentLine( a22, b22, 0.0, a22, b0+len2, 0.0 ),
       new MarkupSegmentLine( a22, b0+len2, 0.0, a12, b0+len2, 0.0 ),
       new MarkupSegmentLine( a12, b0+len2, 0.0, a12, b12, 0.0 )};
}

return MarkupSegmentDown;
  }

  /**
   * Функция отрисовки вертикальных границ клеток
   */
  void Vert(  ) {

int i;
if (m_hor>1) {
  Line_vert = new ShapeLine[m_hor-1];
  for (i=0; i<=m_hor-2;i++) {
    Line_vert[i] = new ShapeLine();
    Line_vert[i].setDrawMode(ShapeDrawMode.SHAPE_DRAW_2D3D);
    Line_vert[i].setLineWidth(1.0);
    Line_vert[i].setColor(green);
    Line_vert[i].setZHeight(50.0);
    Line_vert[i].setX(len1*(i+1)/m_hor+a0);
    Line_vert[i].setY(b0);
    Line_vert[i].setZ(0);
    Line_vert[i].setDx(0);
    Line_vert[i].setDy(len2);
    Line_vert[i].setDz(0);
    Line_vert[i].setLineStyle(LINE_STYLE_DASHED);
    Line_vert[i].setVisible(true);
    presentation.add(Line_vert[i]);
  }
}
  }

  /**
   * Функция отрисовки горизонтальных границ клеток
   */
  void Hor(  ) {

int i;
if (m_vert>1) {
  Line_hor = new ShapeLine[m_vert-1];
  for (i=0; i<=m_vert-2;i++) {
    Line_hor[i] = new ShapeLine();
    Line_hor[i].setDrawMode(ShapeDrawMode.SHAPE_DRAW_2D3D);
    Line_hor[i].setLineWidth(1.0);
    Line_hor[i].setColor(green);
    Line_hor[i].setZHeight(50.0);
    Line_hor[i].setX(a11);
    Line_hor[i].setY(len2*(i+1)/m_vert+b0);
    Line_hor[i].setZ(0);
    Line_hor[i].setDx(len1);
    Line_hor[i].setDy(0);
    Line_hor[i].setDz(0);
    Line_hor[i].setLineStyle(LINE_STYLE_DASHED);
    Line_hor[i].setVisible(true);
    presentation.add(Line_hor[i]);
  }
}
  }
```

```
  /**
5425   * Функция сокрытия границ клеток
   */
  void LineClean(  ) {

int i;
5430 if (m_vert>1) {
  for (i=0; i<m_vert-1;i++) {
    Line_hor[i].setVisible(false);
  }
}
5435 if (m_hor>1) {
  for (i=0; i<m_hor-1;i++) {
    Line_vert[i].setVisible(false);
  }
}
5440   }

  /**
   * Функция определение максимальной области видимости монитора
   */
5445
Rectangle
 MaxRectangle(  ) {

GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
5450 GraphicsDevice gd = ge.getDefaultScreenDevice();
GraphicsConfiguration gc = gd.getDefaultConfiguration();
Insets ins = java.awt.Toolkit.getDefaultToolkit().getScreenInsets(gc);

Rectangle maxRect = gc.getBounds();
5455 maxRect.setBounds(maxRect.x - ins.left, maxRect.y - ins.top, maxRect.width - ins.right, maxRect.height - ins.bottom);

return (maxRect);
  }

5460   /**
   * Функция определения принадлежности точки помещению
   */

boolean
5465 inRectangle( double x_lu, double y_lu, double x_rd, double y_rd, double x, double y ) {

if ((x>=x_lu)&&(x<=x_rd)&&(y>=y_lu)&&(y<=y_rd)) {
  return true;
} else {
5470   return false;
}
  }

  /**
5475   * Функция возвращающее текущее системное время
   */

String
 getDateTime(  ) {
5480
DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
Date date = new Date();
return dateFormat.format(date);
  }
5485
  /**
   * Функция соотвествия статуса агента и номера в списке
   */

5490 int
 status( Status status ) {

if (status.name()=="alive") {
  return 3;
5495 } else if (status.name()=="disoriented") {
  return 2;
} else if (status.name()=="injured") {
  return 1;
} else {
5500   return 0;
}
  }

  /**
```

```
    * Функция соотвествия статуса ситуации и номера в списке
    */

int
 situation( Situation situation ) {

if (situation.name()=="lack") {
  return 0;
} else if (situation.name()=="immediately") {
  return 3;
} else if (situation.name()=="active") {
  return 2;
} else {
  return 1;
}
  }

  /**
   * Функция соотвествия пола агента и номера в списке
   */

int
 gender( Gender gender ) {

if (gender.name()=="Female") {
  return 0;
} else {
  return 1;
}
  }

  /**
   * Расчет количества агентов по статусам
   */

int[]
 stat_status(  ) {

int[] status = {0, 0, 0, 0};
for (int i=0;i<person.size();i++) {
  if (person.get(i).status==Status.alive) {
    status[3]++;
  } else if (person.get(i).status==Status.disoriented) {
    status[2]++;
  } else if (person.get(i).status==Status.injured) {
    status[1]++;
  } else {
    status[0]++;
  }
}
return status;
  }

  /**
   * Функция скроллинга области моделирования
   */
  void scrolling( int stepX, int stepY ) {

arrow.setPos(arrow.getX()+stepX,arrow.getY()+stepY);
getPresentation().getPanel().setOffsets(-(int)arrow.getX(),-(int)arrow.getY());

  }

  /**
   * Количество погибших агентов
   */

int
 allKilled(  ) {

int i = 0;
for (Person p: person) {
  if (p.status==Status.killed){
    i++;
  }
}
return i;
  }
  /**
   * Автоматически создаваемый(е) набор(ы) данных для динамической переменной t
   */
  @AnyLogicInternalCodegenAPI
```

```
  public DataSet _ds_t = new DataSet( 100, new DataUpdater_xjal() {
    double _lastUpdateTime = Double.NaN;
    @Override
    public void update( DataSet _d ) {
      if ( time() == _lastUpdateTime ) { return; }
      _d.add( time(), Main.this.t );
      _lastUpdateTime = time();
    }
    /**
     * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
     */
    @AnyLogicInternalCodegenAPI
    private static final long serialVersionUID = -9043181999061309441L;
  } );
  @AnyLogicInternalCodegenAPI
  public DataSet _plot_st_expression0_dataSet_xjal = new DataSet( 100, new DataUpdater_xjal() {
    double _lastUpdateX = Double.NaN;
    @Override
    public void update( DataSet _d ) {
      if ( time() == _lastUpdateX ) { return; }
      _d.add( time(), __plot_st_expression0_dataSet_xjal_YValue() );
      _lastUpdateX = time();
    }
    /**
     * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
     */
    @AnyLogicInternalCodegenAPI
    private static final long serialVersionUID = -9043181998978419990L;
  } );
  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private double __plot_st_expression0_dataSet_xjal_YValue() {
    return
stat_status()[0]
;
  }

  @AnyLogicInternalCodegenAPI
  public DataSet _plot_st_expression1_dataSet_xjal = new DataSet( 100, new DataUpdater_xjal() {
    double _lastUpdateX = Double.NaN;
    @Override
    public void update( DataSet _d ) {
      if ( time() == _lastUpdateX ) { return; }
      _d.add( time(), __plot_st_expression1_dataSet_xjal_YValue() );
      _lastUpdateX = time();
    }
    /**
     * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
     */
    @AnyLogicInternalCodegenAPI
    private static final long serialVersionUID = -9043181998978419990L;
  } );
  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private double __plot_st_expression1_dataSet_xjal_YValue() {
    return
stat_status()[1]
;
  }

  @AnyLogicInternalCodegenAPI
  public DataSet _plot_st_expression2_dataSet_xjal = new DataSet( 100, new DataUpdater_xjal() {
    double _lastUpdateX = Double.NaN;
    @Override
    public void update( DataSet _d ) {
      if ( time() == _lastUpdateX ) { return; }
      _d.add( time(), __plot_st_expression2_dataSet_xjal_YValue() );
      _lastUpdateX = time();
    }
    /**
     * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
     */
    @AnyLogicInternalCodegenAPI
    private static final long serialVersionUID = -9043181998978419990L;
  } );
  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
```

```
  @AnyLogicInternalCodegenAPI
  private double __plot_st_expression2_dataSet_xjal_YValue() {
    return
stat_status()[2]
;
  }

  @AnyLogicInternalCodegenAPI
  public DataSet _plot_st_expression3_dataSet_xjal = new DataSet( 100, new DataUpdater_xjal() {
    double _lastUpdateX = Double.NaN;
    @Override
    public void update( DataSet _d ) {
      if ( time() == _lastUpdateX ) { return; }
      _d.add( time(), __plot_st_expression3_dataSet_xjal_YValue() );
      _lastUpdateX = time();
    }
    /**
     * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
     */
    @AnyLogicInternalCodegenAPI
    private static final long serialVersionUID = -9043181998978419990L;
  } );
  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private double __plot_st_expression3_dataSet_xjal_YValue() {
    return
stat_status()[3]
;
  }

  @AnyLogicInternalCodegenAPI
  public DataSet _plot_saved_expression0_dataSet_xjal = new DataSet( 100, new DataUpdater_xjal() {
    double _lastUpdateX = Double.NaN;
    @Override
    public void update( DataSet _d ) {
      if ( time() == _lastUpdateX ) { return; }
      _d.add( time(), __plot_saved_expression0_dataSet_xjal_YValue() );
      _lastUpdateX = time();
    }
    /**
     * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
     */
    @AnyLogicInternalCodegenAPI
    private static final long serialVersionUID = -9043181998978419990L;
  } );
  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private double __plot_saved_expression0_dataSet_xjal_YValue() {
    return
saved
;
  }

  // Области
  public ViewArea viewArea_base = new ViewArea( this, null, 0, 0, ViewArea.TOP_LEFT, ViewArea.NONE, 1.0, 100, 100 );
  public ViewArea viewArea_stat = new ViewArea( this, null, 0, -2600, ViewArea.TOP_LEFT, ViewArea.NONE, 1.0, 100, 100 );
  public ViewArea _origin_VA = new ViewArea( this, "[Нач. координат]", 0, 0, ViewArea.TOP_LEFT, ViewArea.SPECIFIED_ZOOM, 1, 100, 100 );
  @Override
  @AnyLogicInternalCodegenAPI
  public int getViewAreas(Map<String, ViewArea> _output) {
    if ( _output != null ) {
      _output.put( "viewArea_base", this.viewArea_base );
      _output.put( "viewArea_stat", this.viewArea_stat );
      _output.put( "_origin_VA", this._origin_VA );
    }
    return 3 + super.getViewAreas( _output );
  }
  @AnyLogicInternalCodegenAPI
  protected static final Font _button_expl_Font = new Font("Arial", 1, 12 );
  @AnyLogicInternalCodegenAPI
  protected static final Font _button_stat_Font = _button_expl_Font;
  @AnyLogicInternalCodegenAPI
  protected static final Font _button_base_Font = _button_expl_Font;
  @AnyLogicInternalCodegenAPI
  protected static final Font _button_zoom_Font = _button_expl_Font;
  @AnyLogicInternalCodegenAPI
  protected static final Font _button_scale_Font = _button_expl_Font;
  @AnyLogicInternalCodegenAPI
```

```
protected static final Font _text_Font = new Font("Calibri", 1, 14 );
@AnyLogicInternalCodegenAPI
protected static final Font _text1_Font = _text_Font;
@AnyLogicInternalCodegenAPI
protected static final Font _text2_Font = _text_Font;
@AnyLogicInternalCodegenAPI
protected static final Font _text3_Font = _text_Font;
@AnyLogicInternalCodegenAPI
protected static final Font _text4_Font = _text_Font;
@AnyLogicInternalCodegenAPI
protected static final Font _text5_Font = _text_Font;
@AnyLogicInternalCodegenAPI
protected static final Font _text6_Font = _text_Font;
@AnyLogicInternalCodegenAPI
protected static final Font _text7_Font = _text_Font;
@AnyLogicInternalCodegenAPI
protected static final Font _text8_Font = _text_Font;
protected static final Color _geometry_Fill_Color = new Color( 0xFFFEF1E5, true );
protected static final Color _geometry1_Fill_Color = new Color( 0xFFFEF1E5, true );
protected static final Color _geometry2_Fill_Color = new Color( 0xFFFEF1E5, true );
protected static final Color _geometry3_Fill_Color = new Color( 0xFFFEF1E5, true );
protected static final Color _geometry4_Fill_Color = new Color( 0xFFFEF1E5, true );
protected static final Color _geometry5_Fill_Color = new Color( 0xFFFEF1E5, true );
@AnyLogicInternalCodegenAPI
protected static final int _scale_Shape = 1;
@AnyLogicInternalCodegenAPI
protected static final int _button_expl = 2;
@AnyLogicInternalCodegenAPI
protected static final int _button_stat = 3;
@AnyLogicInternalCodegenAPI
protected static final int _button_base = 4;
@AnyLogicInternalCodegenAPI
protected static final int _button_zoom = 5;
@AnyLogicInternalCodegenAPI
protected static final int _button_scale = 6;
@AnyLogicInternalCodegenAPI
protected static final int _rectangle_base = 7;
@AnyLogicInternalCodegenAPI
protected static final int _arc_right = 8;
@AnyLogicInternalCodegenAPI
protected static final int _arc_left = 9;
@AnyLogicInternalCodegenAPI
protected static final int _rectangle_stat = 10;
@AnyLogicInternalCodegenAPI
protected static final int _geometry = 11;
@AnyLogicInternalCodegenAPI
protected static final int _text = 12;
@AnyLogicInternalCodegenAPI
protected static final int _geometry1 = 13;
@AnyLogicInternalCodegenAPI
protected static final int _text1 = 14;
@AnyLogicInternalCodegenAPI
protected static final int _geometry2 = 15;
@AnyLogicInternalCodegenAPI
protected static final int _text2 = 16;
@AnyLogicInternalCodegenAPI
protected static final int _geometry3 = 17;
@AnyLogicInternalCodegenAPI
protected static final int _text3 = 18;
@AnyLogicInternalCodegenAPI
protected static final int _geometry4 = 19;
@AnyLogicInternalCodegenAPI
protected static final int _text4 = 20;
@AnyLogicInternalCodegenAPI
protected static final int _geometry5 = 21;
@AnyLogicInternalCodegenAPI
protected static final int _text5 = 22;
@AnyLogicInternalCodegenAPI
protected static final int _explosion_presentation = 23;
@AnyLogicInternalCodegenAPI
protected static final int _rectangle1 = 24;
@AnyLogicInternalCodegenAPI
protected static final int _text6 = 25;
@AnyLogicInternalCodegenAPI
protected static final int _rectangle2 = 26;
@AnyLogicInternalCodegenAPI
protected static final int _text7 = 27;
@AnyLogicInternalCodegenAPI
protected static final int _person_presentation = 28;
@AnyLogicInternalCodegenAPI
protected static final int _rectangle3 = 29;
@AnyLogicInternalCodegenAPI
```

```
protected static final int _text8 = 30;
@AnyLogicInternalCodegenAPI
protected static final int _arrow3 = 31;
@AnyLogicInternalCodegenAPI
protected static final int _arrow2 = 32;
@AnyLogicInternalCodegenAPI
protected static final int _arrow4 = 33;
@AnyLogicInternalCodegenAPI
protected static final int _arrow1 = 34;
@AnyLogicInternalCodegenAPI
protected static final int _arrow = 35;
@AnyLogicInternalCodegenAPI
protected static final int _plot_st = 36;
@AnyLogicInternalCodegenAPI
protected static final int _plot_saved = 37;

/** Internal constant, shouldn't be accessed by user */
@AnyLogicInternalCodegenAPI
protected static final int _SHAPE_NEXT_ID_xjal = 38;


/**
 * Идентификатор группы presentation верхнего уровня
 */
@AnyLogicInternalCodegenAPI
protected static final int _presentation = 0;

@AnyLogicInternalCodegenAPI
public boolean isPublicPresentationDefined() {
  return true;
}
/**
 * Идентификатор группы icon верхнего уровня
 */
@AnyLogicInternalCodegenAPI
protected static final int _icon = -1;

@AnyLogicInternalCodegenAPI
protected static final double[] _arrow3_pointsDX_xjal() {
  return new double[] { 0.0, 0.0, -1.0, 2.0, 7.0, 5.0, 5.0,  };
}
@AnyLogicInternalCodegenAPI
protected static final double[] _arrow3_pointsDY_xjal() {
  return new double[] { 0.0, -9.0, -9.0, -15.0, -9.0, -9.0, 0.0,  };
}
@AnyLogicInternalCodegenAPI
protected static final double[] _arrow3_pointsDZ_xjal() {
  return new double[] { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  };
}
@AnyLogicInternalCodegenAPI
protected static final double[] _arrow2_pointsDX_xjal() {
  return new double[] { 0.0, 9.0, 9.0, 15.0, 9.0, 9.0, 0.0,  };
}
@AnyLogicInternalCodegenAPI
protected static final double[] _arrow2_pointsDY_xjal() {
  return new double[] { 0.0, 0.0, -1.0, 2.0, 7.0, 5.0, 5.0,  };
}
@AnyLogicInternalCodegenAPI
protected static final double[] _arrow2_pointsDZ_xjal() {
  return new double[] { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  };
}
@AnyLogicInternalCodegenAPI
protected static final double[] _arrow4_pointsDX_xjal() {
  return new double[] { 0.0, -9.0, -9.0, -15.0, -9.0, -9.0, 0.0,  };
}
@AnyLogicInternalCodegenAPI
protected static final double[] _arrow4_pointsDY_xjal() {
  return new double[] { 0.0, 0.0, 1.0, -2.0, -7.0, -5.0, -5.0,  };
}
@AnyLogicInternalCodegenAPI
protected static final double[] _arrow4_pointsDZ_xjal() {
  return new double[] { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  };
}
@AnyLogicInternalCodegenAPI
protected static final double[] _arrow1_pointsDX_xjal() {
  return new double[] { 0.0, 0.0, 1.0, -2.0, -7.0, -5.0, -5.0,  };
}
@AnyLogicInternalCodegenAPI
protected static final double[] _arrow1_pointsDY_xjal() {
  return new double[] { 0.0, 9.0, 9.0, 15.0, 9.0, 9.0, 0.0,  };
}
@AnyLogicInternalCodegenAPI
```

```
  protected static final double[] _arrow1_pointsDZ_xjal() {
    return new double[] { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,  };
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public boolean onShapeClick( int _shape, int index, double clickx, double clicky ){
    switch( _shape ){
      case _arrow3:
        if (true) {
          ShapePolyLine self = this.arrow3;

scrolling(0,-100);
        }
        break;
      case _arrow2:
        if (true) {
          ShapePolyLine self = this.arrow2;

scrolling(100,0);
        }
        break;
      case _arrow4:
        if (true) {
          ShapePolyLine self = this.arrow4;

scrolling(-100,0);
        }
        break;
      case _arrow1:
        if (true) {
          ShapePolyLine self = this.arrow1;

scrolling(0,100);
        }
        break;
      default: return super.onShapeClick( _shape, index, clickx, clicky );
  }
  return false;
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void executeShapeControlAction( int _shape, int index ) {
    switch( _shape ) {
      case _button_expl: {
          ShapeButton self = this.button_expl;
double x_try, y_try;
do {
  if (distr_expl.equals("Равномерное распределение")) {
    x_try = a0 + len1*Math.random();
    y_try = b0 + len2*Math.random();
  } else if (distr_expl.equals("Нормальное распределение")) {
    x_try = a0 + normal(0, len1, len1/2 + mu_norm_expl_x*len1/2, scale.toPixels(sig_norm_expl_x, LENGTH_UNIT_METER));
    y_try = b0 + normal(0, len2, len2/2 + mu_norm_expl_y*len2/2, scale.toPixels(sig_norm_expl_y, LENGTH_UNIT_METER));
  } else if (distr_expl.equals("Треугольное распределение")) {
    x_try = a0 + triangular(0, len1, len1/2 + c_trian_expl_x*len1/2);
    y_try = b0 + triangular(0, len2, len2/2 + c_trian_expl_y*len2/2);
  } else {
    x_try = a0 + laplace(scale.toPixels(a_lapl_expl_x, LENGTH_UNIT_METER), len1/2 + b_lapl_expl_x*len1/2);
    y_try = b0 + laplace(scale.toPixels(a_lapl_expl_y, LENGTH_UNIT_METER), len2/2 + b_lapl_expl_y*len2/2);
  }
} while (!inRectangle(a0,b0,a0+len1,b0+len2,x_try,y_try));
add_explosion().setXYZ(x_try, y_try, 0);
t_expl.add(expl, t);
x_expl.add(expl, x_try);
y_expl.add(expl, y_try);
database.modify("INSERT INTO explosion values("+Integer.toString(version)+","
+Double.toString(x_try)+","+Double.toString(y_try)+","+Double.toString(varepsilon_0)+","
+Double.toString(varepsilon_1)+","+Double.toString(varepsilon_2)+","
+Double.toString(t_expl.get(expl))+","+Integer.toString(expl+1)+")");
for (int i=0;i<person.size();i++) {
  person.get(i).t_expl.add(expl, t_expl.get(expl));
  person.get(i).n_expl++;
  if (person.get(i).distanceTo(x_try, y_try)<=explosion.get(expl).epsilon0) {
    person.get(i).t_1 = 0;
    person.get(i).set_status(Status.killed);
    person.get(i).set_status_change(true);
    person.get(i).set_situation(Situation.immediately);
  } else if (person.get(i).distanceTo(x_try, y_try)<=explosion.get(expl).epsilon1) {
    person.get(i).t_1 = Double.POSITIVE_INFINITY;
    person.get(i).set_status(Status.injured);
```

```
    person.get(i).set_status_change(true);
    person.get(i).set_v_hat_max((person.get(i).distanceTo(x_try, y_try)-explosion.get(expl).epsilon0)/(explosion.get(expl).epsilon1-
      explosion.get(expl).epsilon0)*person.get(i).v_hat_max);
    if (person.get(i).getVelocity()>person.get(i).v_hat_max) {
      person.get(i).setVelocity(person.get(i).v_hat_max);
    }
    person.get(i).set_situation(Situation.immediately);
  } else if (person.get(i).distanceTo(x_try, y_try)<=explosion.get(expl).epsilon2) {
    person.get(i).t_1 = t_expl.get(expl) + kappa_1*Math.exp(-(person.get(i).distanceTo(x_try, y_try)-explosion.get(expl).epsilon1));
    person.get(i).set_status(Status.disoriented);
    person.get(i).set_status_change(true);
    person.get(i).set_situation(Situation.immediately);
  } else {
    person.get(i).t_1 = t_expl.get(expl) + kappa_1*Math.exp(-(person.get(i).distanceTo(x_try, y_try)-explosion.get(expl).epsilon1));
    person.get(i).set_situation(Situation.immediately);
  }
  person.get(i).set_expl(true);
}
expl++;
button_expl.setVisible(false);
;}
        break;
      case _button_stat: {
          ShapeButton self = this.button_stat;
viewArea_stat.navigateTo();
;}
        break;
      case _button_base: {
          ShapeButton self = this.button_base;
viewArea_base.navigateTo();
;}
        break;
      case _button_zoom: {
          ShapeButton self = this.button_zoom;
viewArea_base.setScaling(viewArea_base.ZOOM_TO_FIT);
viewArea_base.navigateTo();
;}
        break;
      case _button_scale: {
          ShapeButton self = this.button_scale;
viewArea_base.setScaling(viewArea_base.SPECIFIED_ZOOM);
viewArea_base.navigateTo();
;}
        break;
      default:
        super.executeShapeControlAction( _shape, index );
        break;
    }
  }

  ShapeScale scale_Shape;
  ShapeButton button_expl;
  ShapeButton button_stat;
  ShapeButton button_base;
  ShapeButton button_zoom;
  ShapeButton button_scale;
  TimePlot plot_st;
  TimePlot plot_saved;
  ShapeRectangle rectangle_base;

  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private void _arc_right_SetDynamicParams_xjal( ShapeArc shape ) {
    shape.setX(
a0+len1
);
    shape.setY(
b21+(b22-b21)/2
);
    shape.setRadiusX(
scale.toPixels(edge_x, LENGTH_UNIT_METER)
);
    shape.setRadiusY(
(b22-b21)/2+scale.toPixels(edge_y, LENGTH_UNIT_METER)
);
  }

  ShapeArc arc_right;

  /**
```

```
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private void _arc_left_SetDynamicParams_xjal( ShapeArc shape ) {
    shape.setX(
a0
);
    shape.setY(
b11+(b12-b11)/2
);
    shape.setRadiusX(
scale.toPixels(edge_x, LENGTH_UNIT_METER)
);
    shape.setRadiusY(
(b12-b11)/2+scale.toPixels(edge_y, LENGTH_UNIT_METER)
);
  }

  ShapeArc arc_left;
  ShapeRectangle rectangle_stat;
  ShapeRectangle geometry;
  ShapeText text;
  ShapeRectangle geometry1;
  ShapeText text1;
  ShapeRectangle geometry2;
  ShapeText text2;
  ShapeRectangle geometry3;
  ShapeText text3;
  ShapeRectangle geometry4;
  ShapeText text4;
  ShapeRectangle geometry5;
  ShapeText text5;

  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private void _explosion_presentation_SetDynamicParams_xjal( ShapeEmbeddedObjectPresentation shape, int index ) {
    shape.setEmbeddedObject_xjal(
explosion.get( index )
);
  }


  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public ShapeEmbeddedObjectPresentation _explosion_presentation_createShapeWithStaticProperties_xjal( final int _index ) {
    ShapeEmbeddedObjectPresentation shape = new ShapeEmbeddedObjectPresentation( Main.this, SHAPE_DRAW_2D3D, true, -160.0, -170.0, 0.0,
      0.0,
    false, explosion.get( _index ) );
    return shape;
  }

  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private int _explosion_presentation_Replication() {
    return
explosion.size()
;
  }

  ReplicatedShape<ShapeEmbeddedObjectPresentation> explosion_presentation;
  ShapeRectangle rectangle1;
  ShapeText text6;
  ShapeRectangle rectangle2;
  ShapeText text7;

  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private void _person_presentation_SetDynamicParams_xjal( ShapeEmbeddedObjectPresentation shape, int index ) {
    shape.setEmbeddedObject_xjal(
person.get( index )
);
  }
```

```
  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public ShapeEmbeddedObjectPresentation _person_presentation_createShapeWithStaticProperties_xjal( final int _index ) {
    ShapeEmbeddedObjectPresentation shape = new ShapeEmbeddedObjectPresentation( Main.this, SHAPE_DRAW_2D3D, true, -160.0, -400.0, 0.0,
      0.0,
    false, person.get( _index ) );
    return shape;
  }

  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private int _person_presentation_Replication() {
    return
person.size()
;
  }

  ReplicatedShape<ShapeEmbeddedObjectPresentation> person_presentation;
  ShapeRectangle rectangle3;
  ShapeText text8;
  ShapePolyLine arrow3;
  ShapePolyLine arrow2;
  ShapePolyLine arrow4;
  ShapePolyLine arrow1;
  ShapeGroup arrow;
  @AnyLogicInternalCodegenAPI
  private void _createPersistentElementsBP0_xjal() {
    scale_Shape = new ShapeScale( this, false, 60.0, -580.0, 0.0, 100.0, LENGTH_UNIT_METER );
    button_expl = new ShapeButton(
      Main.this, true, 30.0, 10.0,
      100.0, 30.0,
            controlDefault, controlDefault,
            _button_expl_Font,
      "Взрыв" ) {

      @Override
      @AnyLogicInternalCodegenAPI
      public void action(){
        executeShapeControlAction( _button_expl, 0 );
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = -9043181999309035862L;
    };
    button_stat = new ShapeButton(
      Main.this, true, 30.0, 50.0,
      100.0, 30.0,
            controlDefault, controlDefault,
            _button_stat_Font,
      "Статистика" ) {

      @Override
      @AnyLogicInternalCodegenAPI
      public void action(){
        executeShapeControlAction( _button_stat, 0 );
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = -9043181995820177413L;
    };
    button_base = new ShapeButton(
      Main.this, true, 60.0, -2580.0,
      100.0, 30.0,
            controlDefault, controlDefault,
            _button_base_Font,
      "Назад" ) {

      @Override
      @AnyLogicInternalCodegenAPI
      public void action(){
        executeShapeControlAction( _button_base, 0 );
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
```

```
6240          */
              @AnyLogicInternalCodegenAPI
              private static final long serialVersionUID = -9043181995819370582L;
            };
            button_zoom = new ShapeButton(
              Main.this, true, 150.0, 10.0,
              160.0, 30.0,
                    controlDefault, controlDefault,
                    _button_zoom_Font,
              "Масштабировать") {
6250
              @Override
              @AnyLogicInternalCodegenAPI
              public void action(){
                executeShapeControlAction(_button_zoom, 0);
6255          }
              /**
               * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
               */
              @AnyLogicInternalCodegenAPI
6260          private static final long serialVersionUID = -9043181995819064406L;
            };
            button_scale = new ShapeButton(
              Main.this, true, 150.0, 50.0,
              160.0, 30.0,
6265                  controlDefault, controlDefault,
                      _button_scale_Font,
              "Исходный масштаб") {

              @Override
6270          @AnyLogicInternalCodegenAPI
              public void action(){
                executeShapeControlAction(_button_scale, 0);
              }
              /**
6275           * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
               */
              @AnyLogicInternalCodegenAPI
              private static final long serialVersionUID = -9043181995819309122L;
            };
6280        rectangle_base = new ShapeRectangle(
               SHAPE_DRAW_2D3D, true,0.0, 0.0, 0.0, 0.0,
                    black, ghostWhite,
              1100.0, 700.0, 10.0, 1.0, LINE_STYLE_SOLID);
            arc_right = new ShapeArc(
6285         SHAPE_DRAW_2D3D, true, -100.0, 100.0, 10.0, 0.0,
                    black, white,
              50.0, 70.0, 0.0, 0.0, -1.5707963267948966, 3.141592653589793,
              1.0, LINE_STYLE_SOLID) {
              @Override
6290          public void updateDynamicProperties(boolean publicOnly) {
                _arc_right_SetDynamicParams_xjal(this);
                super.updateDynamicProperties(publicOnly);
              }
              /**
6295           * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
               */
              @AnyLogicInternalCodegenAPI
              private static final long serialVersionUID = -9043181995752265730L;
            };
6300        arc_left = new ShapeArc(
             SHAPE_DRAW_2D3D, true, -160.0, 100.0, 10.0, 0.0,
                    black, white,
              50.0, 70.0, 0.0, 0.0, 1.5707963267948966, 3.141592653589793,
              1.0, LINE_STYLE_SOLID) {
6305          @Override
              public void updateDynamicProperties(boolean publicOnly) {
                _arc_left_SetDynamicParams_xjal(this);
                super.updateDynamicProperties(publicOnly);
              }
6310          /**
               * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
               */
              @AnyLogicInternalCodegenAPI
              private static final long serialVersionUID = -9043181995819326805L;
6315        };
            rectangle_stat = new ShapeRectangle(
               SHAPE_DRAW_2D3D, true,0.0, -2600.0, 0.0, 0.0,
                    black, floralWhite,
              1320.0, 860.0, 10.0, 1.0, LINE_STYLE_SOLID);
6320        geometry = new ShapeRectangle(
               SHAPE_DRAW_2D3D, true,40.0, -370.0, 0.0, 0.0,
```

```
            black, _geometry_Fill_Color,
        490.0, 330.0, 10.0, 1.0, LINE_STYLE_SOLID );
    geometry.setVisible( false );
    text = new ShapeText(
        SHAPE_DRAW_2D, true,285.0, -360.0, 0.0, 0.0,
        black,"Параметры помещения",
        _text_Font, ALIGNMENT_CENTER );
    text.setVisible( false );
    geometry1 = new ShapeRectangle(
       SHAPE_DRAW_2D3D, true,552.0, -370.0, 0.0, 0.0,
            black, _geometry1_Fill_Color,
      290.0, 330.0, 10.0, 1.0, LINE_STYLE_SOLID );
    geometry1.setVisible( false );
    text1 = new ShapeText(
        SHAPE_DRAW_2D, true,697.0, -360.0, 0.0, 0.0,
        black,"Параметры распределения",
        _text1_Font, ALIGNMENT_CENTER );
    text1.setVisible( false );
    geometry2 = new ShapeRectangle(
       SHAPE_DRAW_2D3D, true,867.0, -370.0, 0.0, 0.0,
            black, _geometry2_Fill_Color,
      160.0, 330.0, 10.0, 1.0, LINE_STYLE_SOLID );
    geometry2.setVisible( false );
    text2 = new ShapeText(
        SHAPE_DRAW_2D, true,947.0, -360.0, 0.0, 0.0,
        black,"Параметры модели",
        _text2_Font, ALIGNMENT_CENTER );
    text2.setVisible( false );
    geometry3 = new ShapeRectangle(
       SHAPE_DRAW_2D3D, true,1050.0, -370.0, 0.0, 0.0,
            black, _geometry3_Fill_Color,
      160.0, 330.0, 10.0, 1.0, LINE_STYLE_SOLID );
    geometry3.setVisible( false );
    text3 = new ShapeText(
        SHAPE_DRAW_2D, true,1130.0, -360.0, 0.0, 0.0,
        black,"Параметры движения",
        _text3_Font, ALIGNMENT_CENTER );
    text3.setVisible( false );
    geometry4 = new ShapeRectangle(
       SHAPE_DRAW_2D3D, true,1230.0, -370.0, 0.0, 0.0,
            black, _geometry4_Fill_Color,
      440.0, 330.0, 10.0, 1.0, LINE_STYLE_SOLID );
    geometry4.setVisible( false );
    text4 = new ShapeText(
        SHAPE_DRAW_2D, true,1450.0, -360.0, 0.0, 0.0,
        black,"Параметры взрыва",
        _text4_Font, ALIGNMENT_CENTER );
    text4.setVisible( false );
    geometry5 = new ShapeRectangle(
       SHAPE_DRAW_2D3D, true,1690.0, -370.0, 0.0, 0.0,
            black, _geometry5_Fill_Color,
      300.0, 330.0, 10.0, 1.0, LINE_STYLE_SOLID );
    geometry5.setVisible( false );
    text5 = new ShapeText(
        SHAPE_DRAW_2D, true,1840.0, -360.0, 0.0, 0.0,
        black,"Параметры функционала",
        _text5_Font, ALIGNMENT_CENTER );
    text5.setVisible( false );
    rectangle1 = new ShapeRectangle(
       SHAPE_DRAW_2D3D, true,40.0, -650.0, 0.0, 0.0,
            black, honeyDew,
      490.0, 270.0, 10.0, 1.0, LINE_STYLE_SOLID );
    rectangle1.setVisible( false );
    text6 = new ShapeText(
        SHAPE_DRAW_2D, true,285.0, -640.0, 0.0, 0.0,
        black,"Дополнительные параметры",
        _text6_Font, ALIGNMENT_CENTER );
    text6.setVisible( false );
    rectangle2 = new ShapeRectangle(
       SHAPE_DRAW_2D3D, true,550.0, -650.0, 0.0, 0.0,
            black, honeyDew,
      660.0, 270.0, 10.0, 1.0, LINE_STYLE_SOLID );
    rectangle2.setVisible( false );
    text7 = new ShapeText(
        SHAPE_DRAW_2D, true,880.0, -640.0, 0.0, 0.0,
        black,"Функции и параметры моделирования",
        _text7_Font, ALIGNMENT_CENTER );
    text7.setVisible( false );
    rectangle3 = new ShapeRectangle(
       SHAPE_DRAW_2D3D, true,1230.0, -650.0, 0.0, 0.0,
            black, honeyDew,
      440.0, 270.0, 10.0, 1.0, LINE_STYLE_SOLID );
```

```
    rectangle3.setVisible( false );
    text8 = new ShapeText(
        SHAPE_DRAW_2D, true,1450.0, -640.0, 0.0, 0.0,
        black,"Статистика",
        _text8_Font, ALIGNMENT_CENTER );
    text8.setVisible( false );
    arrow3 = new ShapePolyLine(
    SHAPE_DRAW_2D, true, 36.0, 26.0, 0.0, black, silver,
            7, _arrow3_pointsDX_xjal(), _arrow3_pointsDY_xjal(), _arrow3_pointsDZ_xjal(), true, 0.0, 1.0, LINE_STYLE_SOLID ) {

      @Override
      @AnyLogicInternalCodegenAPI
      public boolean onClick( double clickx, double clicky ) {
        return onShapeClick( _arrow3, 0, clickx, clicky );
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = -9043181999309031426L;
    };
    arrow2 = new ShapePolyLine(
    SHAPE_DRAW_2D, true, 43.0, 28.0, 0.0, black, silver,
            7, _arrow2_pointsDX_xjal(), _arrow2_pointsDY_xjal(), _arrow2_pointsDZ_xjal(), true, 0.0, 1.0, LINE_STYLE_SOLID ) {

      @Override
      @AnyLogicInternalCodegenAPI
      public boolean onClick( double clickx, double clicky ) {
        return onShapeClick( _arrow2, 0, clickx, clicky );
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = -9043181999309031430L;
    };
    arrow4 = new ShapePolyLine(
    SHAPE_DRAW_2D, true, 34.0, 32.0, 0.0, black, silver,
            7, _arrow4_pointsDX_xjal(), _arrow4_pointsDY_xjal(), _arrow4_pointsDZ_xjal(), true, 0.0, 1.0, LINE_STYLE_SOLID ) {

      @Override
      @AnyLogicInternalCodegenAPI
      public boolean onClick( double clickx, double clicky ) {
        return onShapeClick( _arrow4, 0, clickx, clicky );
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = -9043181999309031430L;
    };
    arrow1 = new ShapePolyLine(
    SHAPE_DRAW_2D, true, 40.0, 33.0, 0.0, black, silver,
            7, _arrow1_pointsDX_xjal(), _arrow1_pointsDY_xjal(), _arrow1_pointsDZ_xjal(), true, 0.0, 1.0, LINE_STYLE_SOLID ) {

      @Override
      @AnyLogicInternalCodegenAPI
      public boolean onClick( double clickx, double clicky ) {
        return onShapeClick( _arrow1, 0, clickx, clicky );
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = -9043181999309031426L;
    };
  }

  @AnyLogicInternalCodegenAPI
  private void _createPersistentElementsAP0_xjal() {
    {
    DataSet _item;
    List<DataSet> _items = new ArrayList<DataSet>( 4 );
    _items.add( _plot_st_expression0_dataSet_xjal );
    _items.add( _plot_st_expression1_dataSet_xjal );
    _items.add( _plot_st_expression2_dataSet_xjal );
    _items.add( _plot_st_expression3_dataSet_xjal );
    List<String> _titles = new ArrayList<String>( 4 );
    _titles.add( "Убитые" );
    _titles.add( "Раненые" );
    _titles.add( "Дезориентированные" );
    _titles.add( "Вменяемые" );
```

```
    List<Chart2DPlot.Appearance> _appearances = new ArrayList<Chart2DPlot.Appearance>( 4 );
    _appearances.add( new Chart2DPlot.Appearance( red, true, Chart.INTERPOLATION_STEP, 2, Chart.POINT_CIRCLE ) );
    _appearances.add( new Chart2DPlot.Appearance( blue, true, Chart.INTERPOLATION_STEP, 2, Chart.POINT_CIRCLE ) );
    _appearances.add( new Chart2DPlot.Appearance( green, true, Chart.INTERPOLATION_STEP, 2, Chart.POINT_CIRCLE ) );
    _appearances.add( new Chart2DPlot.Appearance( darkMagenta, true, Chart.INTERPOLATION_STEP, 2, Chart.POINT_CIRCLE ) );
    plot_st = new TimePlot(
        Main.this, true, 40.0, -2520.0,
      700.0, 440.0,
            null, null,
            50.0, 10.0,
      620.0, 350.0, white, black, black,
            30.0, Chart.NORTH,

4
            , Chart.WINDOW_MOVES_WITH_TIME, null, Chart.SCALE_FIXED,

0
            ,
agents_count+5
            , Chart.GRID_DEFAULT, Chart.GRID_DEFAULT,
            darkGray, darkGray, _items, _titles, _appearances );
    }
    {
    DataSet _item;
    List<DataSet> _items = new ArrayList<DataSet>( 1 );
    _items.add( _plot_saved_expression0_dataSet_xjal );
    List<String> _titles = new ArrayList<String>( 1 );
    _titles.add( "Спасенные" );
    List<Chart2DPlot.Appearance> _appearances = new ArrayList<Chart2DPlot.Appearance>( 1 );
    _appearances.add( new Chart2DPlot.Appearance( navy, true, Chart.INTERPOLATION_STEP, 3, Chart.POINT_CIRCLE ) );
    plot_saved = new TimePlot(
        Main.this, true, 40.0, -2080.0,
      700.0, 320.0,
            null, null,
            50.0, 10.0,
      620.0, 230.0, white, black, black,
            30.0, Chart.NORTH,

4
            , Chart.WINDOW_MOVES_WITH_TIME, null, Chart.SCALE_FIXED,

0
            ,
agents_count+5
            , Chart.GRID_DEFAULT, Chart.GRID_DEFAULT,
            darkGray, darkGray, _items, _titles, _appearances );
    }
    explosion_presentation = new ReplicatedShape<ShapeEmbeddedObjectPresentation>() {
      @Override
      public Class<ShapeEmbeddedObjectPresentation> getShapeClass() {
        return ShapeEmbeddedObjectPresentation.class;
      }

      @Override
      public int getReplication() {
        return _explosion_presentation_Replication();
      }

      @Override
      public ShapeEmbeddedObjectPresentation createShapeWithStaticProperties_xjal( int index ) {
        ShapeEmbeddedObjectPresentation _e = _explosion_presentation_createShapeWithStaticProperties_xjal( index );
        return _e;
      }

      @Override
      public void setShapeDynamicProperties_xjal( ShapeEmbeddedObjectPresentation shape, int index ) {
        _explosion_presentation_SetDynamicParams_xjal( shape, index );
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = -9043181999309756501L;
    };
    person_presentation = new ReplicatedShape<ShapeEmbeddedObjectPresentation>() {
      @Override
      public Class<ShapeEmbeddedObjectPresentation> getShapeClass() {
        return ShapeEmbeddedObjectPresentation.class;
      }

      @Override
      public int getReplication() {
```

```
        return _person_presentation_Replication();
      }

      @Override
      public ShapeEmbeddedObjectPresentation createShapeWithStaticProperties_xjal( int index ) {
        ShapeEmbeddedObjectPresentation _e = _person_presentation_createShapeWithStaticProperties_xjal( index );
        return _e;
      }

      @Override
      public void setShapeDynamicProperties_xjal( ShapeEmbeddedObjectPresentation shape, int index ) {
        _person_presentation_SetDynamicParams_xjal( shape, index );
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = -9043181998977703237L;
    };
    {
    arrow = new ShapeGroup( Main.this, SHAPE_DRAW_2D, true, 350.0, 0.0, 0.0, 0.0

        ,arrow3
        ,arrow2
        ,arrow4
        ,arrow1 );
    }
    arrow.setVisible( false );
  }


  // Статическая инициализация элементов, у которых разрешено программное управление
  {
    _createPersistentElementsBP0_xjal();
  }
  ShapeTopLevelPresentationGroup presentation;
  ShapeGroup icon;

  @Override
  @AnyLogicInternalCodegenAPI
  public Object getPersistentShape( int _shape ) {
    switch (_shape) {
      case _presentation: return presentation;
      case _icon: return icon;
      case _scale_Shape: return scale_Shape;
      case _button_expl: return button_expl;
      case _button_stat: return button_stat;
      case _button_base: return button_base;
      case _button_zoom: return button_zoom;
      case _button_scale: return button_scale;
      case _plot_st: return plot_st;
      case _plot_saved: return plot_saved;
      case _rectangle_base: return rectangle_base;
      case _arc_right: return arc_right;
      case _arc_left: return arc_left;
      case _rectangle_stat: return rectangle_stat;
      case _geometry: return geometry;
      case _text: return text;
      case _geometry1: return geometry1;
      case _text1: return text1;
      case _geometry2: return geometry2;
      case _text2: return text2;
      case _geometry3: return geometry3;
      case _text3: return text3;
      case _geometry4: return geometry4;
      case _text4: return text4;
      case _geometry5: return geometry5;
      case _text5: return text5;
      case _explosion_presentation: return explosion_presentation;
      case _rectangle1: return rectangle1;
      case _text6: return text6;
      case _rectangle2: return rectangle2;
      case _text7: return text7;
      case _person_presentation: return person_presentation;
      case _rectangle3: return rectangle3;
      case _text8: return text8;
      case _arrow3: return arrow3;
      case _arrow2: return arrow2;
      case _arrow4: return arrow4;
      case _arrow1: return arrow1;
      case _arrow: return arrow;
      default: return super.getPersistentShape( _shape );
```

```
    }
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public String getNameOfShape_xjal( Object _shape ) {
    try {
      if ( _shape == null ) return null;
      String _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, presentation, "presentation" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, icon, "icon" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, scale_Shape, "scale_Shape" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, button_expl, "button_expl" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, button_stat, "button_stat" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, button_base, "button_base" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, button_zoom, "button_zoom" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, button_scale, "button_scale" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, plot_st, "plot_st" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, plot_saved, "plot_saved" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, rectangle_base, "rectangle_base" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, arc_right, "arc_right" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, arc_left, "arc_left" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, rectangle_stat, "rectangle_stat" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, geometry, "geometry" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text, "text" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, geometry1, "geometry1" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text1, "text1" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, geometry2, "geometry2" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text2, "text2" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, geometry3, "geometry3" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text3, "text3" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, geometry4, "geometry4" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text4, "text4" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, geometry5, "geometry5" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text5, "text5" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, explosion_presentation, "explosion_presentation" ); if (_name_xjal != null) return
     _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, rectangle1, "rectangle1" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text6, "text6" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, rectangle2, "rectangle2" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text7, "text7" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, person_presentation, "person_presentation" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, rectangle3, "rectangle3" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text8, "text8" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, arrow3, "arrow3" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, arrow2, "arrow2" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, arrow4, "arrow4" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, arrow1, "arrow1" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, arrow, "arrow" ); if (_name_xjal != null) return _name_xjal;
    } catch (Exception e) {
      return null;
    }
    return super.getNameOfShape_xjal( _shape );
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void drawModelElements( Panel _panel, Graphics2D _g, boolean _publicOnly, boolean _isSuperClass ) {
    super.drawModelElements( _panel, _g, _publicOnly, true );
  }


  @Override
  @AnyLogicInternalCodegenAPI
  public boolean onClickModelAt( Panel _panel, double _x, double _y, int _clickCount, boolean _publicOnly, boolean _isSuperClass ) {
    return super.onClickModelAt( _panel, _x, _y, _clickCount, _publicOnly, true );
  }



  /**
   * Конструктор
   */
  public Main( Engine engine, Agent owner, AgentList<? extends Main> collection ) {
    super( engine, owner, collection );
    if (isTopLevelClass_xjal( Main.class )) {
      instantiateBaseStructure_xjal();
    }
  }

  @AnyLogicInternalCodegenAPI
  public void onOwnerChanged_xjal() {
```

```
    super.onOwnerChanged_xjal();
    setupReferences_xjal();
  }

  @AnyLogicInternalCodegenAPI
  public void instantiateBaseStructure_xjal() {
    super.instantiateBaseStructure_xjal();
    setupReferences_xjal();
    // Connect to "database" database:
    database.getConnection();
    // Регистрация в непрерывной части исполняющего модуля Engine
    getEngine().registerAgentWithEquations( this );
  }

  @AnyLogicInternalCodegenAPI
  private void setupReferences_xjal() {
  }

  /**
   * Simple constructor. Please add created agent to some population by calling goToPopulation() function
   */
  public Main() {
  }

  /**
   * Simple constructor. Please add created agent to some population by calling goToPopulation() function
   */
  public Main( double len1, double len2, double a0, double b0, double a11, double b11, double a12, double b12, double a21, double b21,
      double a22, double b22, int m_vert, int m_hor, String distr, double b_lapl_x, double a_lapl_x, double mu_norm_x, double sig_norm_x,
      double c_trian_x, int agents_count, boolean global_distr, double sigma_1, double sigma_2, double eta, double gamma_acc, double
      gamma_dec, double gamma_exit, double varepsilon_0, double varepsilon_1, double varepsilon_2, int zeta, double kappa_1, double
      kappa_2, double kappa_3, double gamma_1, double gamma_2, double gamma_3, double gamma_4, double lambda_vel, double lambda_max,
      String distr_expl, double b_lapl_expl_x, double a_lapl_expl_x, double mu_norm_expl_x, double sig_norm_expl_x, double c_trian_expl_x,
       String distr_exit, String distr_turb, int turb, double b_lapl_y, double a_lapl_y, double mu_norm_y, double sig_norm_y, double
      c_trian_y, double b_lapl_expl_y, double a_lapl_expl_y, double mu_norm_expl_y, double sig_norm_expl_y, double c_trian_expl_y, int
      version, double lambda_avg, double gamma_5, double model_second, double[] varsigma, double[] varrho_3, double[] varrho_4, double[]
      vartheta, int[] theta, int[] theta_hat, int[] global_distr_array, double width, double height, double[] beta_occ, double[] beta_dir,
       double[] beta_dest, double[] beta_angle, double[] beta_wall, double[] beta_vel, double[] beta_max, double[] beta_avg, double[]
      beta_expl, double[] wallCoef, double maxArea ) {
    markParametersAreSet();
    this.len1 = len1;
    this.len2 = len2;
    this.a0 = a0;
    this.b0 = b0;
    this.a11 = a11;
    this.b11 = b11;
    this.a12 = a12;
    this.b12 = b12;
    this.a21 = a21;
    this.b21 = b21;
    this.a22 = a22;
    this.b22 = b22;
    this.m_vert = m_vert;
    this.m_hor = m_hor;
    this.distr = distr;
    this.b_lapl_x = b_lapl_x;
    this.a_lapl_x = a_lapl_x;
    this.mu_norm_x = mu_norm_x;
    this.sig_norm_x = sig_norm_x;
    this.c_trian_x = c_trian_x;
    this.agents_count = agents_count;
    this.global_distr = global_distr;
    this.sigma_1 = sigma_1;
    this.sigma_2 = sigma_2;
    this.eta = eta;
    this.gamma_acc = gamma_acc;
    this.gamma_dec = gamma_dec;
    this.gamma_exit = gamma_exit;
    this.varepsilon_0 = varepsilon_0;
    this.varepsilon_1 = varepsilon_1;
    this.varepsilon_2 = varepsilon_2;
    this.zeta = zeta;
    this.kappa_1 = kappa_1;
    this.kappa_2 = kappa_2;
    this.kappa_3 = kappa_3;
    this.gamma_1 = gamma_1;
    this.gamma_2 = gamma_2;
    this.gamma_3 = gamma_3;
    this.gamma_4 = gamma_4;
    this.lambda_vel = lambda_vel;
    this.lambda_max = lambda_max;
    this.distr_expl = distr_expl;
```

```
    this.b_lapl_expl_x = b_lapl_expl_x;
    this.a_lapl_expl_x = a_lapl_expl_x;
    this.mu_norm_expl_x = mu_norm_expl_x;
    this.sig_norm_expl_x = sig_norm_expl_x;
    this.c_trian_expl_x = c_trian_expl_x;
    this.distr_exit = distr_exit;
    this.distr_turb = distr_turb;
    this.turb = turb;
    this.b_lapl_y = b_lapl_y;
    this.a_lapl_y = a_lapl_y;
    this.mu_norm_y = mu_norm_y;
    this.sig_norm_y = sig_norm_y;
    this.c_trian_y = c_trian_y;
    this.b_lapl_expl_y = b_lapl_expl_y;
    this.a_lapl_expl_y = a_lapl_expl_y;
    this.mu_norm_expl_y = mu_norm_expl_y;
    this.sig_norm_expl_y = sig_norm_expl_y;
    this.c_trian_expl_y = c_trian_expl_y;
    this.version = version;
    this.lambda_avg = lambda_avg;
    this.gamma_5 = gamma_5;
    this.model_second = model_second;
    this.varsigma = varsigma;
    this.varrho_3 = varrho_3;
    this.varrho_4 = varrho_4;
    this.vartheta = vartheta;
    this.theta = theta;
    this.theta_hat = theta_hat;
    this.global_distr_array = global_distr_array;
    this.width = width;
    this.height = height;
    this.beta_occ = beta_occ;
    this.beta_dir = beta_dir;
    this.beta_dest = beta_dest;
    this.beta_angle = beta_angle;
    this.beta_wall = beta_wall;
    this.beta_vel = beta_vel;
    this.beta_max = beta_max;
    this.beta_avg = beta_avg;
    this.beta_expl = beta_expl;
    this.wallCoef = wallCoef;
    this.maxArea = maxArea;
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void create() {
    super.create();
    // Создание экземпляров вложенных объектов
    {
      int _cnt =
0
;
      for ( int i = 0; i < _cnt; i++ ) {
        instantiate_explosion_xjal( i );
      }
    }
    {
      int _cnt =
agents_count
;
      for ( int i = 0; i < _cnt; i++ ) {
        instantiate_person_xjal( i );
      }
    }
    // Присвоение начальных значений простым переменным
    setupPlainVariables_Main_xjal();
    // Динамическая инициализация элементов, у которых разрешено программное управление
    _createPersistentElementsAP0_xjal();
    presentation = new ShapeTopLevelPresentationGroup( Main.this, true, 0, 0, 0, 0 , rectangle_base, arc_right, arc_left, rectangle_stat,
      geometry, text, geometry1, text1, geometry2, text2, geometry3, text3, geometry4, text4, geometry5, text5, explosion_presentation,
      rectangle1, text6, rectangle2, text7, person_presentation, rectangle3, text8, arrow, scale_Shape, button_expl, button_stat,
      button_base, button_zoom, button_scale, plot_st, plot_saved );
    icon = new ShapeGroup( Main.this, true, 0, 0, 0  );
    // Создание реплицированных фигур
    person_presentation.createShapes();
    explosion_presentation.createShapes();
    // Инициализация агентных сред
    {
      double _x_xjal =
500
;
```

```
      double _y_xjal =
500
;
      double _z_xjal =
0
;
      setupSpace( _x_xjal, _y_xjal, _z_xjal );
    }
    {
      double _v1_xjal =
period
;
      enableSteps( _v1_xjal );
    }
    setNetworkUserDefined();
    setLayoutType( LAYOUT_USER_DEFINED );
   // Соединители с нереплицированными объектами
    // Создание реплицированных вложенных объектов
    for ( int i = 0; i < explosion.size(); i++ ) {
       setupParameters_explosion_xjal( explosion.get(i), i );
       create_explosion_xjal( explosion.get(i), i );
    }
    person.setEnvironment( this );
    for ( int i = 0; i < person.size(); i++ ) {
       setupParameters_person_xjal( person.get(i), i );
       create_person_xjal( person.get(i), i );
    }
    setupInitialConditions_xjal( Main.class );
    if (isTopLevelClass_xjal( Main.class )) {
      onCreate();
    }
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void start() {
    super.start();
    start.start();
    line_clean.start();
    coordinates.start();
    pause.start();
    _plot_st_autoUpdateEvent_xjal.start();
    _plot_saved_autoUpdateEvent_xjal.start();
    _autoCreatedDS_xjal.start();
    for (Agent embeddedObject : explosion){
      embeddedObject.start();
    }
    for (Agent embeddedObject : person){
      embeddedObject.start();
    }
    if (isTopLevelClass_xjal( Main.class )) {
      onStartup();
    }
  }

  @AnyLogicInternalCodegenAPI
  public void onStartup() {
    super.onStartup();

int l;
int k = 0;
double x_try, y_try;
double x_exit;
double y_exit;
final double step = Math.sqrt(Math.pow(scale.toPixels(1, LENGTH_UNIT_METER),2)*maxArea/Math.PI);
t_expl.clear();
x_expl.clear();
y_expl.clear();
Area();
myWall();
Vert();
Hor();
rectangle_stat.setHeight(MaxRectangle().height);
rectangle_stat.setWidth(MaxRectangle().width);
if (distr.equals("Равномерное распределение")) {
  for (int i=0;i<m_vert*m_hor;i++) {
    for (int j=0;j<global_distr_array[i];j++) {
      x_try = a0 + (i % m_hor)*len1/m_hor + step + (len1/m_hor-2*step)*Math.random();
      y_try = b0 + Math.floor(i/m_hor)*len2/m_vert + step + (len2/m_vert-2*step)*Math.random();
      if (distr_exit.equals("Равномерное распределение")) {
        l=bernoulli(0.5);
      } else {
```

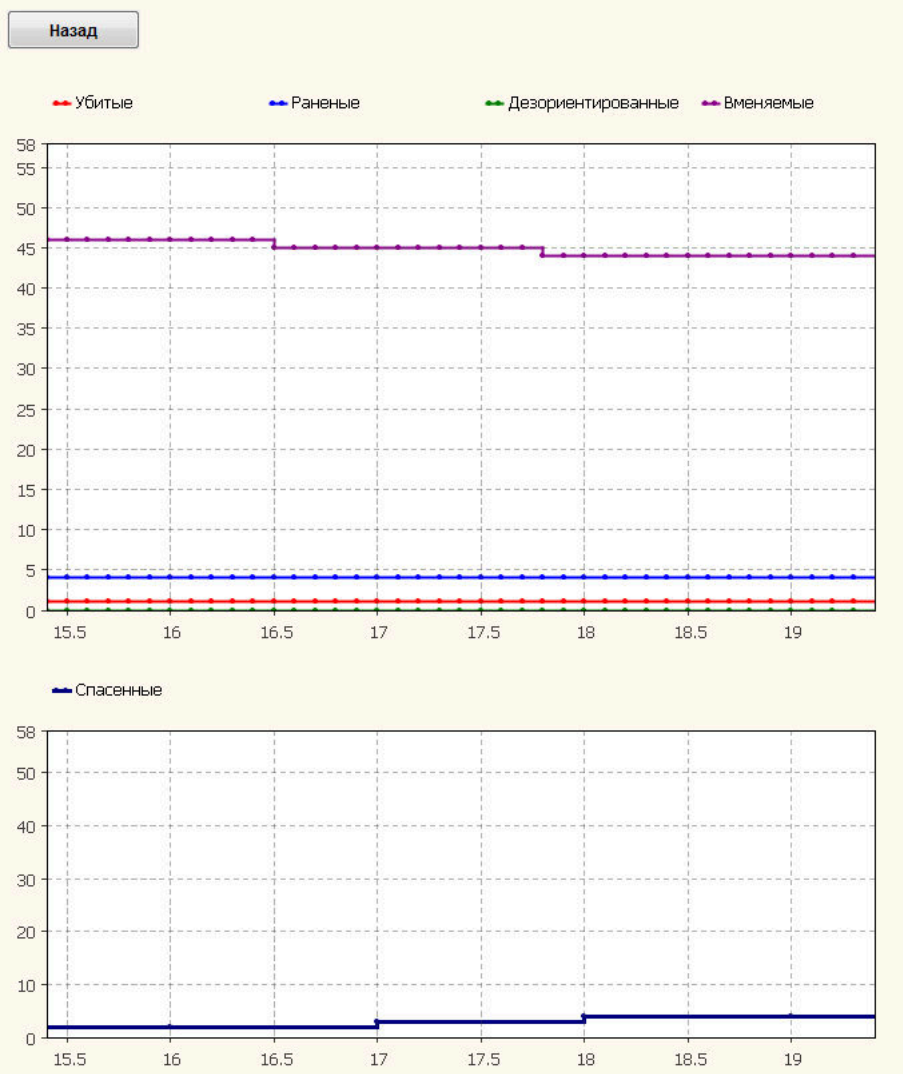
```
        l=bernoulli((x_try-a11)/len1);
      }
      if (l==0) {
        if (y_try<b11+person.get(k).area.getRadiusX()) {
          x_exit = a11;
          y_exit = b11+person.get(k).area.getRadiusX();
        } else if ((y_try>=b11+person.get(k).area.getRadiusX())&&(y_try<=b12-person.get(k).area.getRadiusX())) {
          x_exit = a11;
          y_exit = y_try;
        } else {
          x_exit = a11;
          y_exit = b12-person.get(k).area.getRadiusX();
        }
      } else {
        if (y_try<b21+person.get(k).area.getRadiusX()) {
          x_exit = a21;
          y_exit = b21+person.get(k).area.getRadiusX();
        } else if ((y_try>=b21+person.get(k).area.getRadiusX())&&(y_try<=b22-person.get(k).area.getRadiusX())) {
          x_exit = a21;
          y_exit = y_try;
        } else {
          x_exit = a21;
          y_exit = b22-person.get(k).area.getRadiusX();
        }
      }
      person.get(k).setXYZ(x_try, y_try,0);
      person.get(k).x = x_try;
      person.get(k).y = y_try;
      MyVector vec = new MyVector(x_exit, y_exit, x_try, y_try);
      person.get(k).d_x = vec.unitVector().getX();
      person.get(k).D_x = vec.unitVector().getX();
      person.get(k).d_y = vec.unitVector().getY();
      person.get(k).D_y = vec.unitVector().getY();
      person.get(k).set_exit(Exit.values()[l]);
      person.get(k).x_exit = x_exit;
      person.get(k).y_exit = y_exit;
      person.get(k).moveTo(x_exit, y_exit);
      k++;
    }
  }
} else if (distr.equals("Нормальное распределение")) {
  for (int i=0;i<m_vert*m_hor;i++) {
    for (int j=0;j<global_distr_array[i];j++) {
      x_try = a0 + (i % m_hor)*len1/m_hor + step + normal(0, len1/m_hor-2*step, len1/m_hor/2-step + mu_norm_x*(len1/m_hor/2-step), scale.
      toPixels(sig_norm_x, LENGTH_UNIT_METER));
      y_try = b0 + Math.floor(i/m_hor)*len2/m_vert + step + normal(0, len2/m_vert-2*step, len2/m_vert/2-step + mu_norm_y*(len2/m_vert/2-
      step), scale.toPixels(sig_norm_y, LENGTH_UNIT_METER));
      if (distr_exit.equals("Равномерное распределение")) {
        l=bernoulli(0.5);
      } else {
        l=bernoulli((x_try-a11)/len1);
      }
      if (l==0) {
        if (y_try<b11+person.get(k).area.getRadiusX()) {
          x_exit = a11;
          y_exit = b11+person.get(k).area.getRadiusX();
        } else if ((y_try>=b11+person.get(k).area.getRadiusX())&&(y_try<=b12-person.get(k).area.getRadiusX())) {
          x_exit = a11;
          y_exit = y_try;
        } else {
          x_exit = a11;
          y_exit = b12-person.get(k).area.getRadiusX();
        }
      } else {
        if (y_try<b21+person.get(k).area.getRadiusX()) {
          x_exit = a21;
          y_exit = b21+person.get(k).area.getRadiusX();
        } else if ((y_try>=b21+person.get(k).area.getRadiusX())&&(y_try<=b22-person.get(k).area.getRadiusX())) {
          x_exit = a21;
          y_exit = y_try;
        } else {
          x_exit = a21;
          y_exit = b22-person.get(k).area.getRadiusX();
        }
      }
      person.get(k).setXYZ(x_try, y_try,0);
      person.get(k).x = x_try;
      person.get(k).y = y_try;
      MyVector vec = new MyVector(x_exit, y_exit, x_try, y_try);
      person.get(k).d_x = vec.unitVector().getX();
      person.get(k).D_x = vec.unitVector().getX();
      person.get(k).d_y = vec.unitVector().getY();
```

```
        person.get(k).D_y = vec.unitVector().getY();
        person.get(k).set_exit(Exit.values()[l]);
        person.get(k).x_exit = x_exit;
        person.get(k).y_exit = y_exit;
        person.get(k).moveTo(x_exit, y_exit);
        k++;
      }
    }
  } else if (distr.equals("Треугольное распределение")) {
    for (int i=0;i<m_vert*m_hor;i++) {
      for (int j=0;j<global_distr_array[i];j++) {
        x_try = a0 + (i % m_hor)*len1/m_hor + step + triangular(0, len1/m_hor-2*step, len1/m_hor/2-step + c_trian_x*(len1/m_hor/2-step));
        y_try = b0 + Math.floor(i/m_hor)*len2/m_vert + step + triangular(0, len2/m_vert-2*step, len2/m_vert/2-step + c_trian_y*(len2/m_vert
       /2-step));
        if (distr_exit.equals("Равномерное распределение")) {
          l=bernoulli(0.5);
        } else {
          l=bernoulli((x_try-a11)/len1);
        }
        if (l==0) {
          if (y_try<b11+person.get(k).area.getRadiusX()) {
            x_exit = a11;
            y_exit = b11+person.get(k).area.getRadiusX();
          } else if ((y_try>=b11+person.get(k).area.getRadiusX())&&(y_try<=b12-person.get(k).area.getRadiusX())) {
            x_exit = a11;
            y_exit = y_try;
          } else {
            x_exit = a11;
            y_exit = b12-person.get(k).area.getRadiusX();
          }
        } else {
          if (y_try<b21+person.get(k).area.getRadiusX()) {
            x_exit = a21;
            y_exit = b21+person.get(k).area.getRadiusX();
          } else if ((y_try>=b21+person.get(k).area.getRadiusX())&&(y_try<=b22-person.get(k).area.getRadiusX())) {
            x_exit = a21;
            y_exit = y_try;
          } else {
            x_exit = a21;
            y_exit = b22-person.get(k).area.getRadiusX();
          }
        }
        person.get(k).setXYZ(x_try, y_try,0);
        person.get(k).x = x_try;
        person.get(k).y = y_try;
        MyVector vec = new MyVector(x_exit, y_exit, x_try, y_try);
        person.get(k).d_x = vec.unitVector().getX();
        person.get(k).D_x = vec.unitVector().getX();
        person.get(k).d_y = vec.unitVector().getY();
        person.get(k).D_y = vec.unitVector().getY();
        person.get(k).set_exit(Exit.values()[l]);
        person.get(k).x_exit = x_exit;
        person.get(k).y_exit = y_exit;
        person.get(k).moveTo(x_exit, y_exit);
        k++;
      }
    }
  } else {
    for (int i=0;i<m_vert*m_hor;i++) {
      for (int j=0;j<global_distr_array[i];j++) {
        x_try = a0 + (i % m_hor)*len1/m_hor + step + laplace(scale.toPixels(a_lapl_x, LENGTH_UNIT_METER), len1/m_hor/2-step + b_lapl_x*(len1
       /m_hor/2-step));
        y_try = b0 + Math.floor(i/m_hor)*len2/m_vert + step + laplace(scale.toPixels(a_lapl_y, LENGTH_UNIT_METER), len2/m_vert/2-step +
       b_lapl_y*(len2/m_vert/2-step));
        if (distr_exit.equals("Равномерное распределение")) {
          l=bernoulli(0.5);
        } else {
          l=bernoulli((x_try-a11)/len1);
        }
        if (l==0) {
          if (y_try<b11+person.get(k).area.getRadiusX()) {
            x_exit = a11;
            y_exit = b11+person.get(k).area.getRadiusX();
          } else if ((y_try>=b11+person.get(k).area.getRadiusX())&&(y_try<=b12-person.get(k).area.getRadiusX())) {
            x_exit = a11;
            y_exit = y_try;
          } else {
            x_exit = a11;
            y_exit = b12-person.get(k).area.getRadiusX();
          }
        } else {
          if (y_try<b21+person.get(k).area.getRadiusX()) {
```

```
          x_exit = a21;
          y_exit = b21+person.get(k).area.getRadiusX();
        } else if ((y_try>=b21+person.get(k).area.getRadiusX())&&(y_try<=b22-person.get(k).area.getRadiusX())) {
          x_exit = a21;
          y_exit = y_try;
        } else {
          x_exit = a21;
          y_exit = b22-person.get(k).area.getRadiusX();
        }
      }
      person.get(k).setXYZ(x_try, y_try,0);
      person.get(k).x = x_try;
      person.get(k).y = y_try;
      MyVector vec = new MyVector(x_exit, y_exit, x_try, y_try);
      person.get(k).d_x = vec.unitVector().getX();
      person.get(k).D_x = vec.unitVector().getX();
      person.get(k).d_y = vec.unitVector().getY();
      person.get(k).D_y = vec.unitVector().getY();
      person.get(k).set_exit(Exit.values()[1]);
      person.get(k).x_exit = x_exit;
      person.get(k).y_exit = y_exit;
      person.get(k).moveTo(x_exit, y_exit);
      k++;
    }
  }
}
  }

  /**
   * Присвоение начальных значений простым переменным<br>
   * <em>This method isn't designed to be called by user and may be removed in future releases.</em>
   */
  @AnyLogicInternalCodegenAPI
  public void setupPlainVariables_xjal() {
    setupPlainVariables_Main_xjal();
  }

  /**
   * Присвоение начальных значений простым переменным<br>
   * <em>This method isn't designed to be called by user and may be removed in future releases.</em>
   */
  @AnyLogicInternalCodegenAPI
  private void setupPlainVariables_Main_xjal() {
    saved =
0
;
  }

  // API пользователя ------------------------------------------------------
  @AnyLogicInternalCodegenAPI
  static LinkToAgentAnimationSettings _connections_commonAnimationSettings_xjal = new LinkToAgentAnimationSettingsImpl( false, black, 1.0,
      LINE_STYLE_SOLID, ARROW_NONE, 0.0 );

  public LinkToAgentCollection<Agent, Agent> connections = new LinkToAgentStandardImpl<Agent, Agent>(this,
      _connections_commonAnimationSettings_xjal);
  @Override
  public LinkToAgentCollection<? extends Agent, ? extends Agent> getLinkToAgentStandard_xjal() {
    return connections;
  }


  @AnyLogicInternalCodegenAPI
  public void drawLinksToAgents(boolean _underAgents_xjal, LinkToAgentAnimator _animator_xjal) {
    super.drawLinksToAgents(_underAgents_xjal, _animator_xjal);
    if ( _underAgents_xjal ) {
      _animator_xjal.drawLink( this, connections, true, true );
    }
  }

  public List<Object> getEmbeddedObjects() {
    List<Object> list = super.getEmbeddedObjects();
    if (list == null) {
      list = new LinkedList<Object>();
    }
    list.add( explosion );
    list.add( person );
    return list;
  }

  public AgentList<? extends Main> getPopulation() {
    return (AgentList<? extends Main>) super.getPopulation();
  }
```

```
  public List<? extends Main> agentsInRange( double distance ) {
    return (List<? extends Main>) super.agentsInRange( distance );
  }

  // Действие при изменении ----------------------------------
  public void onChange() {
    super.onChange();
    pause.onChange();
  }

  @AnyLogicInternalCodegenAPI
  public void onDestroy() {
    start.onDestroy();
    line_clean.onDestroy();
    coordinates.onDestroy();
    pause.onDestroy();
    _plot_st_autoUpdateEvent_xjal.onDestroy();
    _plot_saved_autoUpdateEvent_xjal.onDestroy();
    _autoCreatedDS_xjal.onDestroy();
    for (Agent _item : explosion) {
      _item.onDestroy();
    }
    for (Agent _item : person) {
      _item.onDestroy();
    }
    // Отмена регистрации в непрерывной части исполняющего модуля Engine
    getEngine().unregisterAgentWithEquations( this );
    database.destroy();
    _ds_t.destroyUpdater_xjal();
    _plot_st_expression0_dataSet_xjal.destroyUpdater_xjal();
    _plot_st_expression1_dataSet_xjal.destroyUpdater_xjal();
    _plot_st_expression2_dataSet_xjal.destroyUpdater_xjal();
    _plot_st_expression3_dataSet_xjal.destroyUpdater_xjal();
    _plot_saved_expression0_dataSet_xjal.destroyUpdater_xjal();
    super.onDestroy();
  }

  // Дополнительный класс кода

public ShapeLine[] Line_hor;
public ShapeLine[] Line_vert;
public int expl = 0;
final double edge_x = 8;
final double edge_y = 5;
final double period = 0.1;
  // Конец дополнительного класса кода
  /**
   * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
   */
  @AnyLogicInternalCodegenAPI
  private static final long serialVersionUID = -9043181996024581462L;

}
```

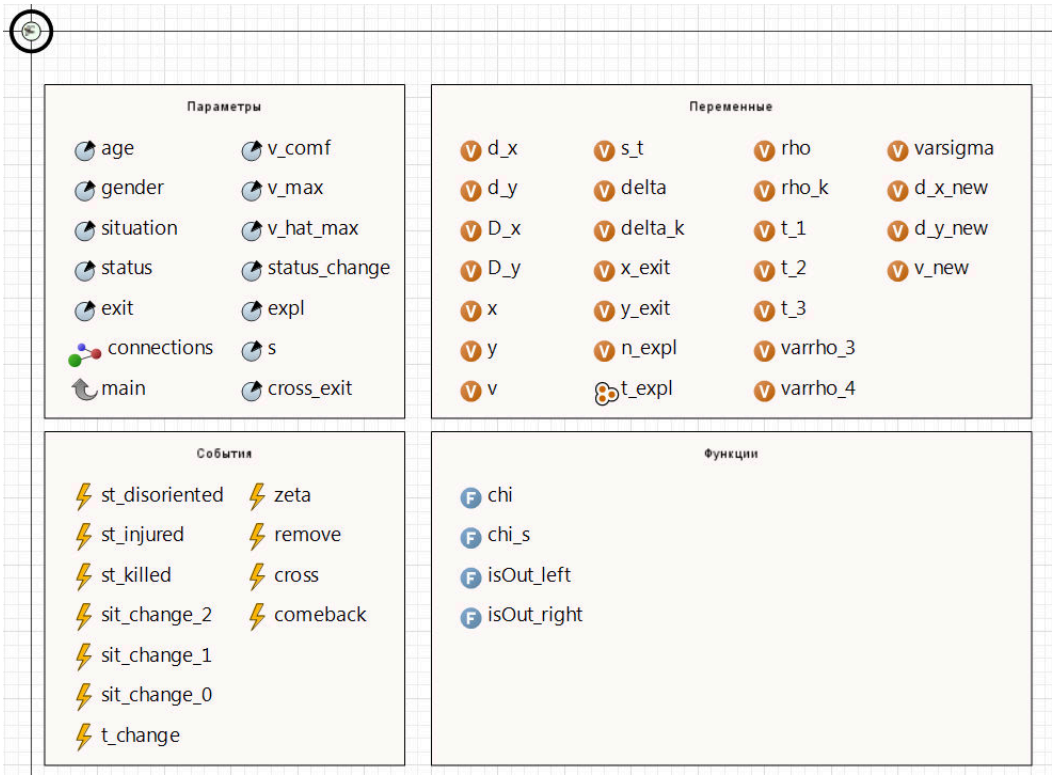
В.3 Скрин окна статистики



Приложение Г

Класс агента

Г.1 Скрин класса агента



Г.2 Листинг кода класса агента

```
package crowd;

import java.io.Serializable;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
```

```
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.LinkedHashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.PriorityQueue;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.markup.Network;
import com.xj.anylogic.engine.Position;
import com.xj.anylogic.engine.markup.PedFlowStatistics;
import com.xj.anylogic.engine.markup.DensityMap;

import static java.lang.Math.*;
import static com.xj.anylogic.engine.UtilitiesArray.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;

import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.markup.*;
import com.xj.anylogic.engine.presentation.*;

import static crowd.Gender.*;
import static crowd.Status.*;
import static crowd.Situation.*;
import static crowd.Exit.*;

import java.awt.geom.Arc2D;

public class Person extends Agent
{
  // Параметры

  /**
   * Возраст агента
   */
  public
int  age;

  /**
   * Возвращает значение по умолчанию параметра <code>age</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public int _age_DefaultValue_xjal() {
    final Person self = this;
    return
0
;
  }

  public void set_age( int age ) {
    if (age == this.age) {
      return;
    }
    int _oldValue_xjal = this.age;
    this.age = age;
    onChange_age_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter age.<br>
```

```
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_age()</code> method instead.
   */
  void onChange_age() {
    onChange_age_xjal( age );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_age_xjal( int oldValue ) {
  }


  /**
   * Пол агента
   */
  public
Gender  gender;

  /**
   * Возвращает значение по умолчанию параметра <code>gender</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public Gender _gender_DefaultValue_xjal() {
    final Person self = this;
    return
Gender.Female
;
  }

  public void set_gender( Gender gender ) {
    if (gender == this.gender) {
      return;
    }
    Gender _oldValue_xjal = this.gender;
    this.gender = gender;
    onChange_gender_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter gender.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_gender()</code> method instead.
   */
  void onChange_gender() {
    onChange_gender_xjal( gender );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_gender_xjal( Gender oldValue ) {
  }


  /**
   * Комфортная скорость ходьбы
   */
  public
double  v_comf;

  /**
   * Возвращает значение по умолчанию параметра <code>v_comf</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _v_comf_DefaultValue_xjal() {
    final Person self = this;
    return
0
;
  }

  public void set_v_comf( double v_comf ) {
    if (v_comf == this.v_comf) {
      return;
    }
    double _oldValue_xjal = this.v_comf;
    this.v_comf = v_comf;
    onChange_v_comf_xjal( _oldValue_xjal );
    onChange();
```

```
  }

  /**
   * Calls "On change" action for parameter v_comf.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_v_comf()</code> method instead.
   */
  void onChange_v_comf() {
    onChange_v_comf_xjal( v_comf );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_v_comf_xjal( double oldValue ) {
  }


  /**
   * Максимальная скорость ходьбы
   */
  public
double  v_max;

  /**
   * Возвращает значение по умолчанию параметра <code>v_max</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _v_max_DefaultValue_xjal() {
    final Person self = this;
    return
0
;
  }

  public void set_v_max( double v_max ) {
    if (v_max == this.v_max) {
      return;
    }
    double _oldValue_xjal = this.v_max;
    this.v_max = v_max;
    onChange_v_max_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter v_max.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_v_max()</code> method instead.
   */
  void onChange_v_max() {
    onChange_v_max_xjal( v_max );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_v_max_xjal( double oldValue ) {
  }


  /**
   * Максимальная скорость бега
   */
  public
double  v_hat_max;

  /**
   * Возвращает значение по умолчанию параметра <code>v_hat_max</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _v_hat_max_DefaultValue_xjal() {
    final Person self = this;
    return
0
;
  }

  public void set_v_hat_max( double v_hat_max ) {
    if (v_hat_max == this.v_hat_max) {
      return;
    }
```

```
    double _oldValue_xjal = this.v_hat_max;
    this.v_hat_max = v_hat_max;
    onChange_v_hat_max_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter v_hat_max.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_v_hat_max()</code> method instead.
   */
  void onChange_v_hat_max() {
    onChange_v_hat_max_xjal( v_hat_max );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_v_hat_max_xjal( double oldValue ) {
  }


  /**
   * Статус агента
   */
  public
Status  status;

  /**
   * Возвращает значение по умолчанию параметра <code>status</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public Status _status_DefaultValue_xjal() {
    final Person self = this;
    return
Status.alive
;
  }

  public void set_status( Status status ) {
    if (status == this.status) {
      return;
    }
    Status _oldValue_xjal = this.status;
    this.status = status;
    onChange_status_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter status.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_status()</code> method instead.
   */
  void onChange_status() {
    onChange_status_xjal( status );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_status_xjal( Status oldValue ) {
  }


  /**
   * Статус ситуации
   */
  public
Situation  situation;

  /**
   * Возвращает значение по умолчанию параметра <code>situation</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public Situation _situation_DefaultValue_xjal() {
    final Person self = this;
    return
Situation.lack
;
  }
```

```
  public void set_situation( Situation situation ) {
    if (situation == this.situation) {
      return;
    }
    Situation _oldValue_xjal = this.situation;
    this.situation = situation;
    onChange_situation_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter situation.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_situation()</code> method instead.
   */
  void onChange_situation() {
    onChange_situation_xjal( situation );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_situation_xjal( Situation oldValue ) {
  }


  /**
   * Направление выхода
   */
  public
Exit  exit;

  /**
   * Возвращает значение по умолчанию параметра <code>exit</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public Exit _exit_DefaultValue_xjal() {
    final Person self = this;
    return
Exit.left
;
  }

  public void set_exit( Exit exit ) {
    if (exit == this.exit) {
      return;
    }
    Exit _oldValue_xjal = this.exit;
    this.exit = exit;
    onChange_exit_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter exit.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_exit()</code> method instead.
   */
  void onChange_exit() {
    onChange_exit_xjal( exit );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_exit_xjal( Exit oldValue ) {
  }


  /**
   * Индикатор взрыва
   */
  public
boolean  expl;

  /**
   * Возвращает значение по умолчанию параметра <code>expl</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public boolean _expl_DefaultValue_xjal() {
    final Person self = this;
    return
```

```
false
;
  }

  public void set_expl( boolean expl ) {
    if (expl == this.expl) {
      return;
    }
    boolean _oldValue_xjal = this.expl;
    this.expl = expl;
    onChange_expl_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter expl.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_expl()</code> method instead.
   */
  void onChange_expl() {
    onChange_expl_xjal( expl );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_expl_xjal( boolean oldValue ) {
  }


  /**
   * Индикатор изменения статуса
   */
  public
boolean  status_change;

  /**
   * Возвращает значение по умолчанию параметра <code>status_change</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public boolean _status_change_DefaultValue_xjal() {
    final Person self = this;
    return
false
;
  }

  public void set_status_change( boolean status_change ) {
    if (status_change == this.status_change) {
      return;
    }
    boolean _oldValue_xjal = this.status_change;
    this.status_change = status_change;
    onChange_status_change_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter status_change.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_status_change()</code> method instead.
   */
  void onChange_status_change() {
    onChange_status_change_xjal( status_change );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_status_change_xjal( boolean oldValue ) {
  }


  /**
   * Базовая площадь горизонтальной проекции
   */
  public
double  s;

  /**
   * Возвращает значение по умолчанию параметра <code>s</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
```

```
  @AnyLogicInternalCodegenAPI
  public double _s_DefaultValue_xjal() {
    final Person self = this;
    return
0
;
  }

  public void set_s( double s ) {
    if (s == this.s) {
      return;
    }
    double _oldValue_xjal = this.s;
    this.s = s;
    onChange_s_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter s.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_s()</code> method instead.
   */
  void onChange_s() {
    onChange_s_xjal( s );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_s_xjal( double oldValue ) {
  }


  /**
   * Индикатор выхода из помещения
   */
  public
boolean  cross_exit;

  /**
   * Возвращает значение по умолчанию параметра <code>cross_exit</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public boolean _cross_exit_DefaultValue_xjal() {
    final Person self = this;
    return
false
;
  }

  public void set_cross_exit( boolean cross_exit ) {
    if (cross_exit == this.cross_exit) {
      return;
    }
    boolean _oldValue_xjal = this.cross_exit;
    this.cross_exit = cross_exit;
    onChange_cross_exit_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter cross_exit.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_cross_exit()</code> method instead.
   */
  void onChange_cross_exit() {
    onChange_cross_exit_xjal( cross_exit );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_cross_exit_xjal( boolean oldValue ) {
  }


  @Override
  public void setParametersToDefaultValues() {
    super.setParametersToDefaultValues();
    age = _age_DefaultValue_xjal();
    gender = _gender_DefaultValue_xjal();
    v_comf = _v_comf_DefaultValue_xjal();
```

```
    v_max = _v_max_DefaultValue_xjal();
    v_hat_max = _v_hat_max_DefaultValue_xjal();
    status = _status_DefaultValue_xjal();
    situation = _situation_DefaultValue_xjal();
    exit = _exit_DefaultValue_xjal();
    expl = _expl_DefaultValue_xjal();
    status_change = _status_change_DefaultValue_xjal();
    s = _s_DefaultValue_xjal();
    cross_exit = _cross_exit_DefaultValue_xjal();
  }

  @Override
  public boolean setParameter(String _name_xjal, Object _value_xjal, boolean _callOnChange_xjal) {
    if ("age".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_age( ((Number) _value_xjal).intValue() );
      } else {
        age = ((Number) _value_xjal).intValue();
      }
      return true;
    }
    if ("gender".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_gender( (Gender) _value_xjal );
      } else {
        gender = (Gender) _value_xjal;
      }
      return true;
    }
    if ("v_comf".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_v_comf( ((Number) _value_xjal).doubleValue() );
      } else {
        v_comf = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("v_max".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_v_max( ((Number) _value_xjal).doubleValue() );
      } else {
        v_max = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("v_hat_max".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_v_hat_max( ((Number) _value_xjal).doubleValue() );
      } else {
        v_hat_max = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("status".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_status( (Status) _value_xjal );
      } else {
        status = (Status) _value_xjal;
      }
      return true;
    }
    if ("situation".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_situation( (Situation) _value_xjal );
      } else {
        situation = (Situation) _value_xjal;
      }
      return true;
    }
    if ("exit".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_exit( (Exit) _value_xjal );
      } else {
        exit = (Exit) _value_xjal;
      }
      return true;
    }
    if ("expl".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_expl( (Boolean) _value_xjal );
      } else {
        expl = (Boolean) _value_xjal;
```

```
      }
      return true;
    }
    if ("status_change".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_status_change( (Boolean) _value_xjal );
      } else {
        status_change = (Boolean) _value_xjal;
      }
      return true;
    }
    if ("s".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_s( ((Number) _value_xjal).doubleValue() );
      } else {
        s = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("cross_exit".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_cross_exit( (Boolean) _value_xjal );
      } else {
        cross_exit = (Boolean) _value_xjal;
      }
      return true;
    }
    return super.setParameter( _name_xjal, _value_xjal, _callOnChange_xjal );
  }
  // Простые переменные

  /**
   * x-координата вектора движения
   */
  public
double
 d_x;
  /**
   * y-координата вектора движения
   */
  public
double
 d_y;
  /**
   * x-координата вектора на выход
   */
  public
double
 D_x;
  /**
   * y-координата вектора на выход
   */
  public
double
 D_y;
  /**
   * x-координата агента
   */
  public
double
 x;
  /**
   * y-координата агента
   */
  public
double
 y;
  /**
   * Параметр rho
   */
  public
double
 rho;
  /**
   * Масив парметра rho_k
   */
  public
double[]
 rho_k;
  /**
   * Параметр t_1
   */
```

```
  public
double
 t_1;
  /**
   * Параметр t_2
   */
  public
double
 t_2;
  /**
   * Параметр t_3
   */
  public
double
 t_3;
  /**
   * Параметр delta
   */
  public
double
 delta;
  /**
   * Скорость агента
   */
  public
double
 v;
  /**
   * x–координата точки выхода
   */
  public
double
 x_exit;
  /**
   * y–координата точки выхода
   */
  public
double
 y_exit;
  /**
   * Масив парметра delta_k
   */
  public
double[]
 delta_k;
  /**
   * Параметр varsigma
   */
  public
double
 varsigma;
  /**
   * Параметр varrho_3
   */
  public
double
 varrho_3;
  /**
   * Параметр varrho_4
   */
  public
double
 varrho_4;
  /**
   * Номер последнего взрыва
   */
  public
int
 n_expl;
  /**
   * Площадь горизонтальной проекции агента
   */
  public
double
 s_t;
  /**
   * x–координата нового вектора движения
   */
  public
double
 d_x_new;
  /**
```

```
    * y-координата нового вектора движения
    */
  public
double
 d_y_new;
  /**
    * Новая скорость агента
    */
  public
double
 v_new;

  // Коллекции
  /**
    * Коллекция времени моментов взрывов
    */
  public
java.util.ArrayList <
Double > t_expl = new java.util.ArrayList<Double>();
  // События

  public EventCondition st_disoriented = new EventCondition(this);
  public EventCondition st_injured = new EventCondition(this);
  public EventCondition st_killed = new EventCondition(this);
  public EventCondition t_change = new EventCondition(this);
  public EventCondition sit_change_2 = new EventCondition(this);
  public EventCondition sit_change_1 = new EventCondition(this);
  public EventCondition sit_change_0 = new EventCondition(this);
  public EventCondition zeta = new EventCondition(this);
  public EventCondition remove = new EventCondition(this);
  public EventCondition cross = new EventCondition(this);
  public EventCondition comeback = new EventCondition(this);


  @Override
  @AnyLogicInternalCodegenAPI
  public String getNameOf( EventCondition _e ) {
    if ( _e == st_disoriented ) return "st_disoriented";
    if ( _e == st_injured ) return "st_injured";
    if ( _e == st_killed ) return "st_killed";
    if ( _e == t_change ) return "t_change";
    if ( _e == sit_change_2 ) return "sit_change_2";
    if ( _e == sit_change_1 ) return "sit_change_1";
    if ( _e == sit_change_0 ) return "sit_change_0";
    if ( _e == zeta ) return "zeta";
    if ( _e == remove ) return "remove";
    if ( _e == cross ) return "cross";
    if ( _e == comeback ) return "comeback";
    return super.getNameOf( _e );
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public boolean testConditionOf( EventCondition _e ) {

    if ( _e == st_disoriented) return
(status_change==true)&&(status==Status.disoriented)
;
    if ( _e == st_injured) return
(status_change==true)&&(status==Status.injured)
;
    if ( _e == st_killed) return
(status_change==true)&&(status==Status.killed)
;
    if ( _e == t_change) return
(expl==true)&&(t_1>0)
;
    if ( _e == sit_change_2) return
(situation!=Situation.active)&&(time()>=t_1)&&(time()<t_2)
;
    if ( _e == sit_change_1) return
(situation!=Situation.extinction)&&(time()>=t_2)&&(time()<t_3)
;
    if ( _e == sit_change_0) return
(situation!=Situation.lack)&&(time()>=t_3)
;
    if ( _e == zeta) return
(status==Status.disoriented)&&(time()>=t_expl.get(n_expl-1)+main.zeta*main.model_second)
;
    if ( _e == remove) return
(cross_exit==true)&&(((getX()<main.a0-area.getRadiusX()*main.wallCoef[situation.ordinal()])&&(isOut_left()==true))||((getX()>main.a0+main.
      len1+area.getRadiusX()*main.wallCoef[situation.ordinal()])&&(isOut_right()==true)))
```

```
;
    if ( _e == cross) return
((getX()<=main.a0+area.getRadiusX()*main.wallCoef[situation.ordinal()])&&(getY()>main.b11+area.getRadiusX()*main.wallCoef[situation.
      ordinal()])&&(getY()<main.b12-area.getRadiusX()*main.wallCoef[situation.ordinal()])&&(cross_exit==false))||((getX()>=main.a0+main.
      len1-area.getRadiusX()*main.wallCoef[situation.ordinal()])
      &&(getY()<main.b22-area.getRadiusX()*main.wallCoef[situation.ordinal()])&&(cross_exit==false))||((getX()<main.a0)&&(cross_exit==
      false))&&((getX()>main.a0+main.len1)&&(cross_exit==false))
;
    if ( _e == comeback) return
((cross_exit==true)&&(getX()>main.a0+area.getRadiusX()*main.wallCoef[situation.ordinal()])&&(getX()<main.a0+main.len1/2))||((cross_exit==
      true)&&(getX()<main.a0+main.len1-area.getRadiusX()*main.wallCoef[situation.ordinal()])&&(getX()>main.a0+main.len1/2))
;
    return super.testConditionOf( _e );
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void executeActionOf( EventCondition _e ) {
    if ( _e == st_disoriented) {
      EventCondition self = _e;

stop();
personal.setLineColor(green);
personal.setLineStyle(LINE_STYLE_SOLID);
set_status_change(false);
;
      return ;
    }
    if ( _e == st_injured) {
      EventCondition self = _e;

personal.setLineColor(blue);
personal.setLineStyle(LINE_STYLE_DASHED);
set_status_change(false);
;
      return ;
    }
    if ( _e == st_killed) {
      EventCondition self = _e;

stop();
area.setFillColor(red);
personal.setRadius(0);
personal.setVisible(false);
set_status_change(false);
;
      return ;
    }
    if ( _e == t_change) {
      EventCondition self = _e;

t_2 = t_expl.get(n_expl-1)+main.kappa_2*(t_1-t_expl.get(n_expl-1));
t_3 = t_expl.get(n_expl-1)+main.kappa_3*(t_1-t_expl.get(n_expl-1));
set_expl(false);
;
      return ;
    }
    if ( _e == sit_change_2) {
      EventCondition self = _e;

set_situation(Situation.active);
;
      return ;
    }
    if ( _e == sit_change_1) {
      EventCondition self = _e;

set_situation(Situation.extinction);
;
      return ;
    }
    if ( _e == sit_change_0) {
      EventCondition self = _e;

set_situation(Situation.lack);
;
      return ;
    }
    if ( _e == zeta) {
      EventCondition self = _e;

set_status(Status.alive);
```

```
personal.setLineColor(black);
personal.setLineStyle(LINE_STYLE_SOLID);
setVelocity(v);
moveTo(x_exit, y_exit);
;
      return ;
    }
    if ( _e == remove) {
      EventCondition self = _e;

main.saved++;
main.remove_person(this);
;
      return ;
    }
    if ( _e == cross) {
      EventCondition self = _e;

set_cross_exit(true);
;
      return ;
    }
    if ( _e == comeback) {
      EventCondition self = _e;

set_cross_exit(false);
;
      return ;
    }
    super.executeActionOf( _e );
  }


  /** Internal constant, shouldn't be accessed by user */
  @AnyLogicInternalCodegenAPI
  protected static final short _STATECHART_ELEMENT_NEXT_ID_xjal = 0;
  // Функции

  /**
   * Функция расчета количества живых агентов в окружении
   */

int
 chi(  ) {

int i = 0;
for (Person p: main.person) {
  if ((distanceTo(p)<=main.eta*Math.sqrt(s_t/Math.PI))&&(p.status!=Status.killed)) {
    i++;
  }
}
return i;
  }

  /**
   * Функция расчета суммы площадей горизонтальных проекций живых агентов в окружении
   */

double
 chi_s(  ) {

double is = 0;
for (Person p: main.person) {
  if ((distanceTo(p)<=main.eta*Math.sqrt(s_t/Math.PI))&&(p.status!=Status.killed)) {
    is = is + p.s_t;
  }
}
return is;
  }

  /**
   * Индикация выхода из помещения слева
   */

boolean
 isOut_left(  ) {

double r;
double a,b;
double phi;
MyVector v1;
MyVector v2;
```

```
a = main.arc_left.getRadiusY();
b = main.arc_left.getRadiusX();
v1 = new MyVector(getX(), getY(), main.arc_left.getX(), main.arc_left.getY());
v2 = new MyVector(main.a0, main.b11, main.arc_left.getX(), main.arc_left.getY());
phi = MyVector.angleRadian(v1, v2);
r = a*b/Math.sqrt(Math.pow(b*Math.cos(phi), 2)+Math.pow(a*Math.sin(phi), 2));
if (distanceTo(main.arc_left.getX(), main.arc_left.getY())>=r) {
  return true;
} else {
  return false;
}
  }

  /**
   * Индикация выхода из помещения справа
   */

boolean
 isOut_right(  ) {

double r;
double a,b;
double phi;
MyVector v1;
MyVector v2;
a = main.arc_right.getRadiusY();
b = main.arc_right.getRadiusX();
v1 = new MyVector(getX(), getY(), main.arc_right.getX(), main.arc_right.getY());
v2 = new MyVector(main.a0+main.len1, main.b22, main.arc_right.getX(), main.arc_right.getY());
phi = MyVector.angleRadian(v1, v2);
r = a*b/Math.sqrt(Math.pow(b*Math.cos(phi), 2)+Math.pow(a*Math.sin(phi), 2));
if (distanceTo(main.arc_right.getX(), main.arc_right.getY())>=r) {
  return true;
} else {
  return false;
}
  }
  // Области
  public ViewArea _origin_VA = new ViewArea( this, "[Нач. координат]", 0, 0, ViewArea.TOP_LEFT, ViewArea.SPECIFIED_ZOOM, 1, 100, 100 );
  @Override
  @AnyLogicInternalCodegenAPI
  public int getViewAreas(Map<String, ViewArea> _output) {
    if ( _output != null ) {
      _output.put( "_origin_VA", this._origin_VA );
    }
    return 1 + super.getViewAreas( _output );
  }
  @AnyLogicInternalCodegenAPI
  protected static final Font _text_Font = new Font("SansSerif", 0, 10 );
  @AnyLogicInternalCodegenAPI
  protected static final Font _text1_Font = _text_Font;
  @AnyLogicInternalCodegenAPI
  protected static final Font _text2_Font = _text_Font;
  @AnyLogicInternalCodegenAPI
  protected static final Font _text3_Font = _text_Font;
  @AnyLogicInternalCodegenAPI
  protected static final int _personal = 1;
  @AnyLogicInternalCodegenAPI
  protected static final int _area = 2;
  @AnyLogicInternalCodegenAPI
  protected static final int _person = 3;
  @AnyLogicInternalCodegenAPI
  protected static final int _pedestrian = 4;
  @AnyLogicInternalCodegenAPI
  protected static final int _rectangle_par = 5;
  @AnyLogicInternalCodegenAPI
  protected static final int _text = 6;
  @AnyLogicInternalCodegenAPI
  protected static final int _rectangle_var = 7;
  @AnyLogicInternalCodegenAPI
  protected static final int _text1 = 8;
  @AnyLogicInternalCodegenAPI
  protected static final int _rectangle_event = 9;
  @AnyLogicInternalCodegenAPI
  protected static final int _text2 = 10;
  @AnyLogicInternalCodegenAPI
  protected static final int _rectangle_func = 11;
  @AnyLogicInternalCodegenAPI
  protected static final int _text3 = 12;

  /** Internal constant, shouldn't be accessed by user */
  @AnyLogicInternalCodegenAPI
```

```
  protected static final int _SHAPE_NEXT_ID_xjal = 13;


  /**
   * Идентификатор группы presentation верхнего уровня
   */
  @AnyLogicInternalCodegenAPI
  protected static final int _presentation = 0;

  @AnyLogicInternalCodegenAPI
  public boolean isPublicPresentationDefined() {
    return true;
  }
  /**
   * Идентификатор группы icon верхнего уровня
   */
  @AnyLogicInternalCodegenAPI
  protected static final int _icon = -1;



  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private void _personal_SetDynamicParams_xjal( ShapeOval shape ) {
    shape.setRadius(
delta
);
  }

  ShapeOval personal;

  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private void _area_SetDynamicParams_xjal( ShapeOval shape ) {
    shape.setRadius(
Math.sqrt(s_t/Math.PI)
);
  }

  ShapeOval area;
  Shape3DObject person;
  ShapeGroup pedestrian;
  ShapeRectangle rectangle_par;
  ShapeText text;
  ShapeRectangle rectangle_var;
  ShapeText text1;
  ShapeRectangle rectangle_event;
  ShapeText text2;
  ShapeRectangle rectangle_func;
  ShapeText text3;
  @AnyLogicInternalCodegenAPI
  private void _createPersistentElementsBP0_xjal() {
    personal = new ShapeOval(
       SHAPE_DRAW_2D3D, true,0.0, 0.0, 0.0, 0.0,
       black, null,
     15.0, 15.0, 10.0, 3.0, LINE_STYLE_SOLID ) {
      @Override
      public void updateDynamicProperties(boolean publicOnly) {
        _personal_SetDynamicParams_xjal( this );
        super.updateDynamicProperties(publicOnly);
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = -8500531889078614994L;
    };
    area = new ShapeOval(
       SHAPE_DRAW_2D3D, true,0.0, 0.0, 0.0, 0.0,
       black, honeyDew,
     7.0, 7.0, 10.0, 1.0, LINE_STYLE_SOLID ) {
      @Override
      public void updateDynamicProperties(boolean publicOnly) {
        _area_SetDynamicParams_xjal( this );
        super.updateDynamicProperties(publicOnly);
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
```

```
     */
    @AnyLogicInternalCodegenAPI
    private static final long serialVersionUID = -8500531889078550422L;
  };
  person = new Shape3DObject(
  Person.this, SHAPE_DRAW_2D3D, false, 0.0, 0.0, 0.0, 0.0,
    1.0, "/crowd/",
    "x3d/person.x3d", OBJECT_3D_YZX_AXIS_ORDER, true, -5.0, -4.0,
    8.0, 8.0, null );
  rectangle_par = new ShapeRectangle(
     SHAPE_DRAW_2D3D, false,10.0, 40.0, 0.0, 0.0,
          black, snow,
    270.0, 250.0, 10.0, 1.0, LINE_STYLE_SOLID );
  rectangle_par.setVisible( false );
  text = new ShapeText(
      SHAPE_DRAW_2D, false,145.0, 50.0, 0.0, 0.0,
      black,"Параметры",
      _text_Font, ALIGNMENT_CENTER );
  text.setVisible( false );
  rectangle_var = new ShapeRectangle(
     SHAPE_DRAW_2D3D, false,300.0, 40.0, 0.0, 0.0,
          black, snow,
    450.0, 250.0, 10.0, 1.0, LINE_STYLE_SOLID );
  rectangle_var.setVisible( false );
  text1 = new ShapeText(
      SHAPE_DRAW_2D, false,525.0, 50.0, 0.0, 0.0,
      black,"Переменные",
      _text1_Font, ALIGNMENT_CENTER );
  text1.setVisible( false );
  rectangle_event = new ShapeRectangle(
     SHAPE_DRAW_2D3D, false,10.0, 300.0, 0.0, 0.0,
          black, snow,
    270.0, 250.0, 10.0, 1.0, LINE_STYLE_SOLID );
  rectangle_event.setVisible( false );
  text2 = new ShapeText(
      SHAPE_DRAW_2D, false,145.0, 310.0, 0.0, 0.0,
      black,"События",
      _text2_Font, ALIGNMENT_CENTER );
  text2.setVisible( false );
  rectangle_func = new ShapeRectangle(
     SHAPE_DRAW_2D3D, false,300.0, 300.0, 0.0, 0.0,
          black, snow,
    450.0, 250.0, 10.0, 1.0, LINE_STYLE_SOLID );
  rectangle_func.setVisible( false );
  text3 = new ShapeText(
      SHAPE_DRAW_2D, false,525.0, 310.0, 0.0, 0.0,
      black,"Функции",
      _text3_Font, ALIGNMENT_CENTER );
  text3.setVisible( false );
}

@AnyLogicInternalCodegenAPI
private void _createPersistentElementsAP0_xjal() {
  {
  pedestrian = new ShapeGroup( Person.this, SHAPE_DRAW_2D3D, true, 0.0, 0.0, 0.0, 0.0

     ,personal
     ,area
     ,person );
  }
}


// Статическая инициализация элементов, у которых разрешено программное управление
{
  _createPersistentElementsBP0_xjal();
}
ShapeTopLevelPresentationGroup presentation;
ShapeGroup icon;

@Override
@AnyLogicInternalCodegenAPI
public Object getPersistentShape( int _shape ) {
  switch (_shape) {
    case _presentation: return presentation;
    case _icon: return icon;
    case _personal: return personal;
    case _area: return area;
    case _person: return person;
    case _pedestrian: return pedestrian;
    case _rectangle_par: return rectangle_par;
    case _text: return text;
```

```
      case _rectangle_var: return rectangle_var;
      case _text1: return text1;
      case _rectangle_event: return rectangle_event;
      case _text2: return text2;
      case _rectangle_func: return rectangle_func;
      case _text3: return text3;
      default: return super.getPersistentShape( _shape );
    }
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public String getNameOfShape_xjal( Object _shape ) {
    try {
      if ( _shape == null ) return null;
      String _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, presentation, "presentation" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, icon, "icon" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, personal, "personal" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, area, "area" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, person, "person" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, pedestrian, "pedestrian" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, rectangle_par, "rectangle_par" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text, "text" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, rectangle_var, "rectangle_var" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text1, "text1" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, rectangle_event, "rectangle_event" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text2, "text2" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, rectangle_func, "rectangle_func" ); if (_name_xjal != null) return _name_xjal;
      _name_xjal = checkNameOfShape_xjal( _shape, text3, "text3" ); if (_name_xjal != null) return _name_xjal;
    } catch (Exception e) {
      return null;
    }
    return super.getNameOfShape_xjal( _shape );
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void drawModelElements( Panel _panel, Graphics2D _g, boolean _publicOnly, boolean _isSuperClass ) {
    super.drawModelElements( _panel, _g, _publicOnly, true );
  }


  @Override
  @AnyLogicInternalCodegenAPI
  public boolean onClickModelAt( Panel _panel, double _x, double _y, int _clickCount, boolean _publicOnly, boolean _isSuperClass ) {
    return super.onClickModelAt( _panel, _x, _y, _clickCount, _publicOnly, true );
  }



  @Override
  @AnyLogicInternalCodegenAPI
  public void onStep() {
    super.onStep();

if (status!=Status.killed) {
  d_x = d_x_new;
  d_y = d_y_new;
  v = v_new;
  setVelocity(v);
  moveTo(x+moveCoef*d_x, y+moveCoef*d_y);
}
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void onBeforeStep() {
    super.onBeforeStep();

if (status!=Status.killed) {
  int i;
  if (status==Status.alive) { // установка площади горизонтальной проекции
    s_t = s;
  } else if (status==Status.disoriented) {
    s_t = main.sigma_2*s;
  } else if (status==Status.injured) {
    s_t = main.sigma_1*s;
  }
  if (situation==Situation.lack) { // установка параметров в зависимости от статуса ситуации
    varsigma = main.varsigma[0];
    varrho_3 = main.varrho_3[0];
```

```
    varrho_4 = main.varrho_4[0];
  } else if (situation==Situation.extinction) {
    varsigma = main.varsigma[1];
    varrho_3 = main.varrho_3[1];
    varrho_4 = main.varrho_4[1];
  } else if (situation==Situation.active) {
    varsigma = main.varsigma[2];
    varrho_3 = main.varrho_3[2];
    varrho_4 = main.varrho_4[2];
  } else {
    varsigma = main.varsigma[3];
    varrho_3 = main.varrho_3[3];
    varrho_4 = main.varrho_4[3];
  }
  for (i=0;i<4;i++) {
    delta_k[i] = main.vartheta[i]*Math.sqrt(s_t/Math.PI);
  }
  rho = chi_s()/(Math.pow(main.eta, 2)*s_t); // расчет плотности
  if (chi_s()!=0) {
    rho_k[2] = varrho_3*s_t*chi()/chi_s();
    rho_k[3] = varrho_4*s_t*chi()/chi_s();
  } else {
    rho_k[2] = Double.POSITIVE_INFINITY;
    rho_k[3] = Double.POSITIVE_INFINITY;
  }
  rho_k[0] = rho_k[2]/3;
  rho_k[1] = rho_k[2]*2/3;
  if (rho<=rho_k[0]) { // установка радиуса личного пространства
    delta = varsigma*delta_k[0];
  } else if (rho<=rho_k[1]) {
    delta = varsigma*delta_k[1];
  } else if (rho<=rho_k[2]) {
    delta = varsigma*delta_k[2];
  } else if (rho<=rho_k[3]) {
    delta = varsigma*delta_k[3];
  } else {
    set_status(Status.killed);
  }

  if (status!=Status.killed) {
    if (exit==Exit.left) { // установка точки выхода
      if (getY()<main.b11+Math.sqrt(s_t/Math.PI)) {
        x_exit = main.a11;
        y_exit = main.b11+Math.sqrt(s_t/Math.PI);
      } else if ((getY()>=main.b11+Math.sqrt(s_t/Math.PI))&&(getY()<=main.b12-Math.sqrt(s_t/Math.PI))) {
        x_exit = main.a11;
        y_exit = getY();
      } else {
        x_exit = main.a11;
        y_exit = main.b12-Math.sqrt(s_t/Math.PI);
      }
    } else {
      if (getY()<main.b21+Math.sqrt(s_t/Math.PI)) {
        x_exit = main.a21;
        y_exit = main.b21+Math.sqrt(s_t/Math.PI);
      } else if ((getY()>=main.b21+Math.sqrt(s_t/Math.PI))&&(getY()<=main.b22-Math.sqrt(s_t/Math.PI))) {
        x_exit = main.a21;
        y_exit = getY();
      } else {
        x_exit = main.a21;
        y_exit = main.b22-Math.sqrt(s_t/Math.PI);
      }
    }
    MyVector vec = new MyVector(x_exit, y_exit, getX(), getY());
    D_x = vec.unitVector().getX();
    D_y = vec.unitVector().getY();
    x = getX();
    y = getY();
    int index = getIndex();
    double r_s = Math.sqrt(s_t/Math.PI);
    double x_expl, y_expl;
    if (main.x_expl.isEmpty()) { // установка координат последнего взрыва
      x_expl = 0;
      y_expl = 0;
    }
    else {
      x_expl = main.x_expl.get(main.x_expl.size()-1);
      y_expl = main.y_expl.get(main.y_expl.size()-1);
    }
    crowd.Collision collision = new crowd.Collision(main.person, index, x, y, v, r_s, s_t, d_x, d_y);
    int nomer = collision.nomerCollision();
    //MessageBox.WarningBox(Integer.toString(nomer), "Ошибка ввода");
```

```
1490      if (nomer>-1) { // проверка наличия коллизии
            Result result_col = collision.calc(nomer);
            v_new = result_col.getV();
            d_x_new = result_col.getX();
            d_y_new = result_col.getY();
1495      } else { // расчет обычного передвижения
            crowd.Decision decision = new crowd.Decision(main.person, main.a0, main.b0, main.len1, main.len2, main.b11,
            main.b12, main.b21, main.b22, index, main.beta_occ, main.beta_dir, main.beta_dest, main.beta_angle,
            main.beta_wall, main.beta_vel, main.beta_max, main.beta_avg, main.beta_expl, main.gamma_1, main.gamma_2,
            main.gamma_3, main.gamma_4, main.gamma_5, main.lambda_vel, main.lambda_max, main.lambda_avg, main.theta,
1500        x, y, x_expl, y_expl, main.gamma_acc, main.gamma_dec, v, v_comf, v_max, v_hat_max, r_s, main.eta, d_x, d_y,
            D_x, D_y, status, situation, exit, main.wallCoef, main.period);
            Result result_dec = decision.newDirection();
            v_new = result_dec.getV();
            d_x_new = result_dec.getX();
1505        d_y_new = result_dec.getY();
          }
          v = getVelocity();
        }
      }
1510  }

      /**
       * Конструктор
       */
1515  public Person( Engine engine, Agent owner, AgentList<? extends Person> collection ) {
        super( engine, owner, collection );
        if (isTopLevelClass_xjal( Person.class )) {
          instantiateBaseStructure_xjal();
        }
1520  }

      @AnyLogicInternalCodegenAPI
      public void onOwnerChanged_xjal() {
        super.onOwnerChanged_xjal();
1525    setupReferences_xjal();
      }

      @AnyLogicInternalCodegenAPI
      public void instantiateBaseStructure_xjal() {
1530    super.instantiateBaseStructure_xjal();
        setupReferences_xjal();
      }

      @AnyLogicInternalCodegenAPI
1535  private void setupReferences_xjal() {
        main = get_Main();
      }

      /**
1540   * Simple constructor. Please add created agent to some population by calling goToPopulation() function
       */
      public Person() {
      }

1545  /**
       * Simple constructor. Please add created agent to some population by calling goToPopulation() function
       */
      public Person( int age, Gender gender, double v_comf, double v_max, double v_hat_max, Status status, Situation situation, Exit exit,
          boolean expl, boolean status_change, double s, boolean cross_exit ) {
        markParametersAreSet();
1550    this.age = age;
        this.gender = gender;
        this.v_comf = v_comf;
        this.v_max = v_max;
        this.v_hat_max = v_hat_max;
1555    this.status = status;
        this.situation = situation;
        this.exit = exit;
        this.expl = expl;
        this.status_change = status_change;
1560    this.s = s;
        this.cross_exit = cross_exit;
      }

      @Override
1565  @AnyLogicInternalCodegenAPI
      public void create() {
        super.create();
        // Присвоение начальных значений простым переменным
        setupPlainVariables_Person_xjal();
1570    // Динамическая инициализация элементов, у которых разрешено программное управление
```

```
    _createPersistentElementsAP0_xjal();
    presentation = new ShapeTopLevelPresentationGroup( Person.this, true, 0, 0, 0, 0 , pedestrian, rectangle_par, text, rectangle_var,
      text1, rectangle_event, text2, rectangle_func, text3 );
    icon = new ShapeGroup( Person.this, true, 0, 0, 0  );
   // Соединители с нереплицированными объектами
    // Создание реплицированных вложенных объектов
    setupInitialConditions_xjal( Person.class );
    if (isTopLevelClass_xjal( Person.class )) {
      onCreate();
    }
  }

  @AnyLogicInternalCodegenAPI
  public void setupExt_xjal(AgentExtension _ext) {
    // Инициализация свойств агентов
    if ( _ext instanceof ExtAgentWithSpatialMetrics ) {
      double _value =
0
;
      ((ExtAgentWithSpatialMetrics) _ext).setVelocity( _value );
    }
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void start() {
    super.start();
    st_disoriented.start();
    st_injured.start();
    st_killed.start();
    t_change.start();
    sit_change_2.start();
    sit_change_1.start();
    sit_change_0.start();
    zeta.start();
    remove.start();
    cross.start();
    comeback.start();
    if (isTopLevelClass_xjal( Person.class )) {
      onStartup();
    }
  }

  @AnyLogicInternalCodegenAPI
  public void onStartup() {
    super.onStartup();

int i;
t_expl.clear();
rho_k = new double[4];
delta_k = new double[4];

ResultSet rs_square = main.database.getResultSet("Select Square from square where Age="+Integer.toString(age)); // запрос площади
      коризонтальной проекции из базы данных
while (rs_square.next()) {
set_s(Math.pow(main.scale.toPixels(1, LENGTH_UNIT_METER),2)*rs_square.getDouble("Square"));
}
if ((status==Status.alive)||(status==Status.killed)) { // установка площади горизонтальной проекции
  s_t = s;
} else if (status==Status.disoriented) {
  s_t = main.sigma_2*s;
} else if (status==Status.injured) {
  s_t = main.sigma_1*s;
}
ResultSet rs_vel = main.database.getResultSet("Select Comfortable, Max_gait, Max_run from velocity where Age="+Integer.toString(age)+" and
      Gender="+Integer.toString(gender.ordinal())); // запрос скоростей из базы данных
while (rs_vel.next()) { // установка скоростей
set_v_comf(main.scale.toPixels(rs_vel.getDouble("Comfortable"), LENGTH_UNIT_METER)/main.model_second);
set_v_max(main.scale.toPixels(rs_vel.getDouble("Max_gait"), LENGTH_UNIT_METER)/main.model_second);
set_v_hat_max(main.scale.toPixels(rs_vel.getDouble("Max_run"), LENGTH_UNIT_METER)/main.model_second);
}
setVelocity(v_comf); // установка текущей скорости
v = getVelocity();
  }

  /**
   * Присвоение начальных значений простым переменным<br>
   * <em>This method isn't designed to be called by user and may be removed in future releases.</em>
   */
  @AnyLogicInternalCodegenAPI
  public void setupPlainVariables_xjal() {
    setupPlainVariables_Person_xjal();
```

```
1650  }

      /**
       * Присвоение начальных значений простым переменным<br>
       * <em>This method isn't designed to be called by user and may be removed in future releases.</em>
1655   */
      @AnyLogicInternalCodegenAPI
      private void setupPlainVariables_Person_xjal() {
        d_x =
0
1660  ;
        d_y =
0
;
        D_x =
1665  0
;
        D_y =
0
;
1670    x =
0
;
        y =
0
1675  ;
        rho =
0
;
        rho_k = new double[]
1680  {0,0,0,0}
;
        t_1 =
0
;
1685    t_2 =
0
;
        t_3 =
0
1690  ;
        delta =
0
;
        v =
1695  0
;
        x_exit =
0
;
1700    y_exit =
0
;
        delta_k = new double[]
{0,0,0,0}
1705  ;
        varsigma =
0
;
        varrho_3 =
1710  0
;
        varrho_4 =
0
;
1715    n_expl =
0
;
        s_t =
0
1720  ;
        d_x_new =
0
;
        d_y_new =
1725  0
;
        v_new =
0
;
1730  }
```

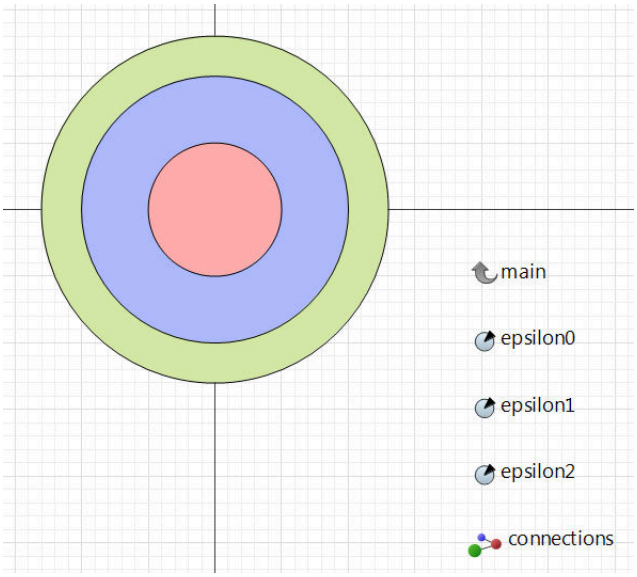
```
  // API пользователя ------------------------------------------------------
  public Main get_Main() {
    {
      Agent owner = getOwner();
      if ( owner instanceof Main ) return (Main) owner;
    }
    return null;
  }

  /**
   * Переменная только для чтения. <em>Значение переменной не должно изменяться пользователем.</em>
   */
  @AnyLogicCustomSerialization(AnyLogicCustomSerializationMode.REFERENCE)
  public transient crowd.Main main;

  @AnyLogicInternalCodegenAPI
  static LinkToAgentAnimationSettings _connections_commonAnimationSettings_xjal = new LinkToAgentAnimationSettingsImpl( false, black, 1.0,
      LINE_STYLE_SOLID, ARROW_NONE, 0.0 );

  public LinkToAgentCollection<Agent, Agent> connections = new LinkToAgentStandardImpl<Agent, Agent>(this,
      _connections_commonAnimationSettings_xjal);
  @Override
  public LinkToAgentCollection<? extends Agent, ? extends Agent> getLinkToAgentStandard_xjal() {
    return connections;
  }


  @AnyLogicInternalCodegenAPI
  public void drawLinksToAgents(boolean _underAgents_xjal, LinkToAgentAnimator _animator_xjal) {
    super.drawLinksToAgents(_underAgents_xjal, _animator_xjal);
    if ( _underAgents_xjal ) {
      _animator_xjal.drawLink( this, connections, true, true );
    }
  }

  public AgentList<? extends Person> getPopulation() {
    return (AgentList<? extends Person>) super.getPopulation();
  }

  public List<? extends Person> agentsInRange( double distance ) {
    return (List<? extends Person>) super.agentsInRange( distance );
  }

  // Действие при изменении ------------------------------------------------
  public void onChange() {
    super.onChange();
    st_disoriented.onChange();
    st_injured.onChange();
    st_killed.onChange();
    t_change.onChange();
    sit_change_2.onChange();
    sit_change_1.onChange();
    sit_change_0.onChange();
    zeta.onChange();
    remove.onChange();
    cross.onChange();
    comeback.onChange();
  }

  @AnyLogicInternalCodegenAPI
  public void onDestroy() {
    st_disoriented.onDestroy();
    st_injured.onDestroy();
    st_killed.onDestroy();
    t_change.onDestroy();
    sit_change_2.onDestroy();
    sit_change_1.onDestroy();
    sit_change_0.onDestroy();
    zeta.onDestroy();
    remove.onDestroy();
    cross.onDestroy();
    comeback.onDestroy();
    super.onDestroy();
  }

  // Дополнительный класс кода

final double moveCoef = 100;
  // Конец дополнительного класса кода
  /**
   * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
   */
```

```
    @AnyLogicInternalCodegenAPI
    private static final long serialVersionUID = -8500531886192933761L;

1815 }
```

Приложение Д

Класс взрыва

Д.1 Скрин класса взрыва



Д.2 Листинг кода класса взрыва

```
package crowd;

import java.io.Serializable;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.LinkedHashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.PriorityQueue;
```

```
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.markup.Network;
import com.xj.anylogic.engine.Position;
import com.xj.anylogic.engine.markup.PedFlowStatistics;
import com.xj.anylogic.engine.markup.DensityMap;

import static java.lang.Math.*;
import static com.xj.anylogic.engine.UtilitiesArray.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;

import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.markup.*;
import com.xj.anylogic.engine.presentation.*;

import static crowd.Gender.*;
import static crowd.Status.*;
import static crowd.Situation.*;
import static crowd.Exit.*;

import java.awt.geom.Arc2D;

public class Explosion extends Agent
{
  // Параметры

  /**
   * Параметр epsilon0
   */
  public
double  epsilon0;

  /**
   * Возвращает значение по умолчанию параметра <code>epsilon0</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _epsilon0_DefaultValue_xjal() {
    final Explosion self = this;
    return
5
;
  }

  public void set_epsilon0( double epsilon0 ) {
    if (epsilon0 == this.epsilon0) {
      return;
    }
    double _oldValue_xjal = this.epsilon0;
    this.epsilon0 = epsilon0;
    onChange_epsilon0_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter epsilon0.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_epsilon0()</code> method instead.
   */
  void onChange_epsilon0() {
    onChange_epsilon0_xjal( epsilon0 );
  }

  @AnyLogicInternalCodegenAPI
```

```
  void onChange_epsilon0_xjal( double oldValue ) {
  }


  /**
   * Параметр epsilon1
   */
  public
double  epsilon1;

  /**
   * Возвращает значение по умолчанию параметра <code>epsilon1</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _epsilon1_DefaultValue_xjal() {
    final Explosion self = this;
    return
10
;
  }

  public void set_epsilon1( double epsilon1 ) {
    if (epsilon1 == this.epsilon1) {
      return;
    }
    double _oldValue_xjal = this.epsilon1;
    this.epsilon1 = epsilon1;
    onChange_epsilon1_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter epsilon1.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_epsilon1()</code> method instead.
   */
  void onChange_epsilon1() {
    onChange_epsilon1_xjal( epsilon1 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_epsilon1_xjal( double oldValue ) {
  }


  /**
   * Параметр epsilon2
   */
  public
double  epsilon2;

  /**
   * Возвращает значение по умолчанию параметра <code>epsilon2</code>.
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  public double _epsilon2_DefaultValue_xjal() {
    final Explosion self = this;
    return
13
;
  }

  public void set_epsilon2( double epsilon2 ) {
    if (epsilon2 == this.epsilon2) {
      return;
    }
    double _oldValue_xjal = this.epsilon2;
    this.epsilon2 = epsilon2;
    onChange_epsilon2_xjal( _oldValue_xjal );
    onChange();
  }

  /**
   * Calls "On change" action for parameter epsilon2.<br>
   * Note that 'oldValue' in that action will be unavailable if this method is called by user
   * (current parameter value will be passed as 'oldValue').<br>
   * Please call <code>set_epsilon2()</code> method instead.
   */
  void onChange_epsilon2() {
```

```
    onChange_epsilon2_xjal( epsilon2 );
  }

  @AnyLogicInternalCodegenAPI
  void onChange_epsilon2_xjal( double oldValue ) {
  }


  @Override
  public void setParametersToDefaultValues() {
    super.setParametersToDefaultValues();
    epsilon0 = _epsilon0_DefaultValue_xjal();
    epsilon1 = _epsilon1_DefaultValue_xjal();
    epsilon2 = _epsilon2_DefaultValue_xjal();
  }

  @Override
  public boolean setParameter(String _name_xjal, Object _value_xjal, boolean _callOnChange_xjal) {
    if ("epsilon0".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_epsilon0( ((Number) _value_xjal).doubleValue() );
      } else {
        epsilon0 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("epsilon1".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_epsilon1( ((Number) _value_xjal).doubleValue() );
      } else {
        epsilon1 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    if ("epsilon2".equals( _name_xjal )) {
      if ( _callOnChange_xjal ) {
        set_epsilon2( ((Number) _value_xjal).doubleValue() );
      } else {
        epsilon2 = ((Number) _value_xjal).doubleValue();
      }
      return true;
    }
    return super.setParameter( _name_xjal, _value_xjal, _callOnChange_xjal );
  }

  /** Internal constant, shouldn't be accessed by user */
  @AnyLogicInternalCodegenAPI
  protected static final short _STATECHART_ELEMENT_NEXT_ID_xjal = 0;
  // Области
  public ViewArea _origin_VA = new ViewArea( this, "[Нач. координат]", 0, 0, ViewArea.TOP_LEFT, ViewArea.SPECIFIED_ZOOM, 1, 100, 100 );
  @Override
  @AnyLogicInternalCodegenAPI
  public int getViewAreas(Map<String, ViewArea> _output) {
    if ( _output != null ) {
      _output.put( "_origin_VA", this._origin_VA );
    }
    return 1 + super.getViewAreas( _output );
  }
  protected static final Color _circle2_Fill_Color = new Color( 0xFFD4E9A6, true );
  protected static final Color _circle1_Fill_Color = new Color( 0xFFAEBAFD, true );
  protected static final Color _circle0_Fill_Color = new Color( 0xFFFFADAD, true );
  @AnyLogicInternalCodegenAPI
  protected static final int _circle2 = 1;
  @AnyLogicInternalCodegenAPI
  protected static final int _circle1 = 2;
  @AnyLogicInternalCodegenAPI
  protected static final int _circle0 = 3;
  @AnyLogicInternalCodegenAPI
  protected static final int _circle = 4;

  /** Internal constant, shouldn't be accessed by user */
  @AnyLogicInternalCodegenAPI
  protected static final int _SHAPE_NEXT_ID_xjal = 5;


  /**
   * Идентификатор группы presentation верхнего уровня
   */
  @AnyLogicInternalCodegenAPI
  protected static final int _presentation = 0;

  @AnyLogicInternalCodegenAPI
```

```
  public boolean isPublicPresentationDefined() {
    return true;
  }
  /**
   * Идентификатор группы icon верхнего уровня
   */
  @AnyLogicInternalCodegenAPI
  protected static final int _icon = -1;



  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private void _circle2_SetDynamicParams_xjal( ShapeOval shape ) {
    shape.setRadius(
epsilon2
);
  }

  ShapeOval circle2;

  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private void _circle1_SetDynamicParams_xjal( ShapeOval shape ) {
    shape.setRadius(
epsilon1
);
  }

  ShapeOval circle1;

  /**
   * <i>Пользователь не должен вызывать этот метод</i>
   */
  @AnyLogicInternalCodegenAPI
  private void _circle0_SetDynamicParams_xjal( ShapeOval shape ) {
    shape.setRadius(
epsilon0
);
  }

  ShapeOval circle0;
  ShapeGroup circle;
  @AnyLogicInternalCodegenAPI
  private void _createPersistentElementsBP0_xjal() {
    circle2 = new ShapeOval(
       SHAPE_DRAW_2D3D, true,0.0, 0.0, 0.0, 0.0,
       black, _circle2_Fill_Color,
     130.0, 130.0, 10.0, 1.0, LINE_STYLE_SOLID ) {
      @Override
      public void updateDynamicProperties(boolean publicOnly) {
        _circle2_SetDynamicParams_xjal( this );
        super.updateDynamicProperties(publicOnly);
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = 3035612837820607528L;
    };
    circle1 = new ShapeOval(
       SHAPE_DRAW_2D3D, true,0.0, 0.0, 0.0, 0.0,
       black, _circle1_Fill_Color,
     100.0, 100.0, 10.0, 1.0, LINE_STYLE_SOLID ) {
      @Override
      public void updateDynamicProperties(boolean publicOnly) {
        _circle1_SetDynamicParams_xjal( this );
        super.updateDynamicProperties(publicOnly);
      }
      /**
       * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
       */
      @AnyLogicInternalCodegenAPI
      private static final long serialVersionUID = 3035612837820545085L;
    };
    circle0 = new ShapeOval(
       SHAPE_DRAW_2D3D, true,0.0, 0.0, 0.0, 0.0,
       black, _circle0_Fill_Color,
```

```
    50.0, 50.0, 10.0, 1.0, LINE_STYLE_SOLID ) {
     @Override
     public void updateDynamicProperties(boolean publicOnly) {
       _circle0_SetDynamicParams_xjal( this );
       super.updateDynamicProperties(publicOnly);
     }
     /**
      * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
      */
     @AnyLogicInternalCodegenAPI
     private static final long serialVersionUID = 303561283782185708 5L;
   };
 }

 @AnyLogicInternalCodegenAPI
 private void _createPersistentElementsAP0_xjal() {
   {
   circle = new ShapeGroup( Explosion.this, SHAPE_DRAW_2D3D, true, 0.0, 0.0, 0.0, 0.0

      ,circle2
      ,circle1
      ,circle0 );
   }
 }


 // Статическая инициализация элементов, у которых разрешено программное управление
 {
   _createPersistentElementsBP0_xjal();
 }
 ShapeTopLevelPresentationGroup presentation;
 ShapeGroup icon;

 @Override
 @AnyLogicInternalCodegenAPI
 public Object getPersistentShape( int _shape ) {
   switch (_shape) {
     case _presentation: return presentation;
     case _icon: return icon;
     case _circle2: return circle2;
     case _circle1: return circle1;
     case _circle0: return circle0;
     case _circle: return circle;
     default: return super.getPersistentShape( _shape );
   }
 }

 @Override
 @AnyLogicInternalCodegenAPI
 public String getNameOfShape_xjal( Object _shape ) {
   try {
     if ( _shape == null ) return null;
     String _name_xjal;
     _name_xjal = checkNameOfShape_xjal( _shape, presentation, "presentation" ); if (_name_xjal != null) return _name_xjal;
     _name_xjal = checkNameOfShape_xjal( _shape, icon, "icon" ); if (_name_xjal != null) return _name_xjal;
     _name_xjal = checkNameOfShape_xjal( _shape, circle2, "circle2" ); if (_name_xjal != null) return _name_xjal;
     _name_xjal = checkNameOfShape_xjal( _shape, circle1, "circle1" ); if (_name_xjal != null) return _name_xjal;
     _name_xjal = checkNameOfShape_xjal( _shape, circle0, "circle0" ); if (_name_xjal != null) return _name_xjal;
     _name_xjal = checkNameOfShape_xjal( _shape, circle, "circle" ); if (_name_xjal != null) return _name_xjal;
   } catch (Exception e) {
     return null;
   }
   return super.getNameOfShape_xjal( _shape );
 }

 @AnyLogicInternalCodegenAPI
 private void drawModelElements_AgentLinks_xjal(Panel _panel, Graphics2D _g, boolean _publicOnly, boolean _isSuperClass ) {
   if (_publicOnly) { return; }
     drawLinkToAgent( _panel, _g, 202, 249, 15, 0, "connections", true, connections );
 }

 @Override
 @AnyLogicInternalCodegenAPI
 public void drawModelElements( Panel _panel, Graphics2D _g, boolean _publicOnly, boolean _isSuperClass ) {
   super.drawModelElements( _panel, _g, _publicOnly, true );
   drawModelElements_AgentLinks_xjal( _panel, _g, _publicOnly, _isSuperClass );
 }

 @AnyLogicInternalCodegenAPI
 private boolean onClickModelAt_AgentLinks_xjal( Panel _panel, double _x, double _y, int _clickCount, boolean _publicOnly, boolean
     _isSuperClass ) {
   if ( modelElementContains(_x, _y, 202, 249) ) {
```

```
        _panel.addInspect_xjal( 202, 249, this, "connections", Panel.INSPECT_CONNECTIONS_xjal );
      return true;
    }
    return false;
  }


  @Override
  @AnyLogicInternalCodegenAPI
  public boolean onClickModelAt( Panel _panel, double _x, double _y, int _clickCount, boolean _publicOnly, boolean _isSuperClass ) {
    if ( onClickModelAt_AgentLinks_xjal( _panel, _x, _y, _clickCount, _publicOnly, _isSuperClass ) ) { return true; }
    return super.onClickModelAt( _panel, _x, _y, _clickCount, _publicOnly, true );
  }



  /**
   * Конструктор
   */
  public Explosion( Engine engine, Agent owner, AgentList<? extends Explosion> collection ) {
    super( engine, owner, collection );
    if (isTopLevelClass_xjal( Explosion.class )) {
      instantiateBaseStructure_xjal();
    }
  }

  @AnyLogicInternalCodegenAPI
  public void onOwnerChanged_xjal() {
    super.onOwnerChanged_xjal();
    setupReferences_xjal();
  }

  @AnyLogicInternalCodegenAPI
  public void instantiateBaseStructure_xjal() {
    super.instantiateBaseStructure_xjal();
    setupReferences_xjal();
  }

  @AnyLogicInternalCodegenAPI
  private void setupReferences_xjal() {
    main = get_Main();
  }

  /**
   * Simple constructor. Please add created agent to some population by calling goToPopulation() function
   */
  public Explosion() {
  }

  /**
   * Simple constructor. Please add created agent to some population by calling goToPopulation() function
   */
  public Explosion( double epsilon0, double epsilon1, double epsilon2 ) {
    markParametersAreSet();
    this.epsilon0 = epsilon0;
    this.epsilon1 = epsilon1;
    this.epsilon2 = epsilon2;
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void create() {
    super.create();
    // Присвоение начальных значений простым переменным
    setupPlainVariables_Explosion_xjal();
    // Динамическая инициализация элементов, у которых разрешено программное управление
    _createPersistentElementsAP0_xjal();
    presentation = new ShapeTopLevelPresentationGroup( Explosion.this, true, 0, 0, 0, 0 , circle );
    icon = new ShapeGroup( Explosion.this, true, 0, 0, 0  );
   // Соединители с нереплицированными объектами
    // Создание реплицированных вложенных объектов
    setupInitialConditions_xjal( Explosion.class );
    if (isTopLevelClass_xjal( Explosion.class )) {
      onCreate();
    }
  }

  @AnyLogicInternalCodegenAPI
  public void setupExt_xjal(AgentExtension _ext) {
    // Инициализация свойств агентов
    if ( _ext instanceof ExtAgentWithSpatialMetrics ) {
      double _value =
```

```
0
;
      ((ExtAgentWithSpatialMetrics) _ext).setVelocity( _value );
    }
  }

  @Override
  @AnyLogicInternalCodegenAPI
  public void start() {
    super.start();
    if (isTopLevelClass_xjal( Explosion.class )) {
      onStartup();
    }
  }


  /**
   * Присвоение начальных значений простым переменным<br>
   * <em>This method isn't designed to be called by user and may be removed in future releases.</em>
   */
  @AnyLogicInternalCodegenAPI
  public void setupPlainVariables_xjal() {
    setupPlainVariables_Explosion_xjal();
  }

  /**
   * Присвоение начальных значений простым переменным<br>
   * <em>This method isn't designed to be called by user and may be removed in future releases.</em>
   */
  @AnyLogicInternalCodegenAPI
  private void setupPlainVariables_Explosion_xjal() {
  }

  // API пользователя ------------------------------------------------------
  public Main get_Main() {
    {
      Agent owner = getOwner();
      if ( owner instanceof Main ) return (Main) owner;
    }
    return null;
  }
  @AnyLogicCustomSerialization(AnyLogicCustomSerializationMode.REFERENCE)
  public transient crowd.Main main;

  @AnyLogicInternalCodegenAPI
  static LinkToAgentAnimationSettings _connections_commonAnimationSettings_xjal = new LinkToAgentAnimationSettingsImpl( false, black, 1.0,
      LINE_STYLE_SOLID, ARROW_NONE, 0.0 );

  public LinkToAgentCollection<Agent, Agent> connections = new LinkToAgentStandardImpl<Agent, Agent>(this,
      _connections_commonAnimationSettings_xjal);
  @Override
  public LinkToAgentCollection<? extends Agent, ? extends Agent> getLinkToAgentStandard_xjal() {
    return connections;
  }


  @AnyLogicInternalCodegenAPI
  public void drawLinksToAgents(boolean _underAgents_xjal, LinkToAgentAnimator _animator_xjal) {
    super.drawLinksToAgents(_underAgents_xjal, _animator_xjal);
    if ( _underAgents_xjal ) {
      _animator_xjal.drawLink( this, connections, true, true );
    }
  }

  public AgentList<? extends Explosion> getPopulation() {
    return (AgentList<? extends Explosion>) super.getPopulation();
  }

  public List<? extends Explosion> agentsInRange( double distance ) {
    return (List<? extends Explosion>) super.agentsInRange( distance );
  }

  /**
   * Это число используется при сохранении состояния модели. Пользователь не должен изменять его значение.
   */
  @AnyLogicInternalCodegenAPI
  private static final long serialVersionUID = 3035612838139395128L;

}
```

Приложение Е

Класс столкновений

Е.1 Листинг кода класса столкновений

```
import crowd.Main._person_Class; // импорт класса агента
import crowd.Person;

public class Collision implements Serializable {

  _person_Class person; // класс агента для циклического перебора

  int index; // номер агента, вызвавшего класс
  double x; // x-координата агента, вызвавшего класс
  double y; // y-координата агента, вызвавшего класс
  double v; // скорость агента, вызвавшего класс
  double r_s; // радиус горизонтальной проекции агента, вызвавшего класс
  double s_t; // площадь горизонтальной проекции агента, вызвавшего класс
  double d_x; // x-координата вектора направления движения агента, вызвавшего класс
  double d_y; // y-координата вектора направления движения агента, вызвавшего класс

    /**
     * Конструктор по умолчанию
     */
    public Collision(_person_Class _person, int _index, double _x, double _y, double _v, double _r_s, double _s_t, double _d_x, double
     _d_y) { // конструктор

      this.person = _person;
    this.index = _index;
      this.x = _x;
    this.y = _y;
    this.v = _v;
    this.r_s = _r_s;
    this.s_t = _s_t;
    this.d_x = _d_x;
    this.d_y = _d_y;
    }

    public int nomerCollision() { // возвращает номер агента, с которым происходит коллизия, иначе возвращает -1

      int nomer = -1;

    for (Person p : person) {
      if (p.getIndex()!=index) {
        MyVector vec = new MyVector(p.getX()-x, p.getY()-y);
        if (vec.length()<=p.area.getRadiusX()+r_s) {
          nomer = p.getIndex();
          return nomer;
        }
      }
    }

    return nomer;
    }

    public Result calc(int index_j) { // расчет итоговых значений скорости и вектора направления движения после коллизии

      MyVector n = new MyVector(person.get(index_j).getX()-x, person.get(index_j).getY()-y).unitVector(); // вектор линии центров агентов
      MyVector tau = n.rotateDegree(-90); // тангенсальный вектор
      MyVector v_old = new MyVector(d_x, d_y).coefVector(v); // ветор старой скорости
      MyVector v_j_old = new MyVector(person.get(index_j).d_x, person.get(index_j).d_y).coefVector(person.get(index_j).v); // вектор
     старой скорости для другого агента
      double v_tau = MyVector.scalarProduct(v_old, tau); // тангенсальная составляющая старой скорости
      double v_n = MyVector.scalarProduct(v_old, n); // нормальная составляющая старой скорости
      double v_j_n = MyVector.scalarProduct(v_j_old, n); // нормальная составляющая старой скорости другого агента

      double v_hat_tau = v_tau; // // тангенсальная составляющая новой скорости
      double v_hat_n = ((s_t-person.get(index_j).s_t)*v_n+2*person.get(index_j).s_t*v_j_n)/(s_t+person.get(index_j).s_t); // нормальная
     составляющая новой скорости
```

```
        double v_hat = Math.sqrt(Math.pow(v_hat_tau,2)+Math.pow(v_hat_n,2)); // новая скорость
        double alpha = Math.atan(v_hat_tau/v_hat_n); // угол отклонения
        MyVector d_new = new MyVector(d_x, d_y).rotateRadian(alpha); // новый вектор направления движения

        return new Result(d_new, v_hat);
    }

    /**
     * Это число используется при сохранении состояния модели
     * Его рекомендуется изменить в случае изменения класса
     */
    private static final long serialVersionUID = 1L;

}
```

Приложение Ж

Класс принятия решения

Ж.1 Листинг кода класса принятия решения

```
import crowd.Main._person_Class; // импорт класса агента
import crowd.Person;

public class Decision implements Serializable {

  final private int phi_0 = 170;
  final private int phi_1 = 120;
  final private int phi_2 = 80;
  final private int phi_3 = 50;
  final private int phi_4 = 30;
  final private int phi_5 = 10;

  private MyVector d;
  private MyVector D;
  private MyVector[] d_new;
  private double[] v_new;
  private MyVector[][] coordinate;
  private double[][] L;
  private boolean[][] bad;
  private int[][] cross;

  double min;
  int imin,jmin;

  _person_Class person;

  double a0; // параметр a0
  double b0; // параметр b0
  double len1; // параметр len1
  double len2; // параметр len2
  double b11; // параметр b11
  double b12; // параметр b12
  double b21; // параметр b21
  double b22; // параметр b22
  int index; // номер агента, вызвавшего класс
  double[] beta_occ; // массив весового коэффициента beta_occ
  double[] beta_dir; // массив весового коэффициента beta_dir
  double[] beta_dest; // массив весового коэффициента beta_dest
  double[] beta_angle; // массив весового коэффициента beta_angle
  double[] beta_wall; // массив весового коэффициента beta_wall
  double[] beta_vel; // массив весового коэффициента beta_vel
  double[] beta_max; // массив весового коэффициента beta_max
  double[] beta_avg; // массив весового коэффициента beta_avg
  double[] beta_expl; // массив весового коэффициента beta_expl
  double gamma_1; // параметр gamma_1
  double gamma_2; // параметр gamma_2
  double gamma_3; // параметр gamma_3
  double gamma_4; // параметр gamma_4
  double gamma_5; // параметр gamma_5
  double lambda_vel; // параметр lambda_vel
  double lambda_max; // параметр lambda_max
  double lambda_avg; // параметр lambda_avg
  int[] theta; // массив параметра theta
  double x; // x-координата агента, вызвавшего класс
  double y; // y-координата агента, вызвавшего класс
  double x_expl; // x-координата агента последнего взрыва
  double y_expl; // y-координата агента последнего взрыва
  double gamma_acc; // параметр gamma_acc
  double gamma_dec; // параметр gamma_dec
  double v; // текущая скорость агента, вызвавшего класс
  double v_comf; // комфортная скорость ходьбы агента, вызвавшего класс
  double v_max; // максимальная скорость ходьбы агента, вызвавшего класс
  double v_hat_max; // максимальная скорость бега агента, вызвавшего класс
  double r_s; // радиус проекции агента, вызвавшего класс
```

```
    double eta; // параметр eta
    double d_x; // x-координата вектора направления движения агента, вызвавшего класс
    double d_y; // y-координата вектора направления движения агента, вызвавшего класс
    double D_x; // x-координата вектора направления выхода для агента, вызвавшего класс
    double D_y; // y-координата вектора направления выхода для агента, вызвавшего класс
    Status status; // текущий статус агента, вызвавшего класс
    Situation situation; // текущий статус ситуации для агента, вызвавшего класс
    Exit exit; // сторона выхода агента, вызвавшего класс
    double[] wallCoef; // массив коэффициетов близости к стене
    double measure; // период между шагами симуляции

    public Decision(_person_Class _person, double _a0, double _b0, double _len1, double _len2, double _b11, double _b12, double _b21, double
        _b22, int _index, double[] _beta_occ, double[] _beta_dir, double[] _beta_dest, double[] _beta_angle, double[] _beta_wall, double[]
       _beta_vel, double[] _beta_max, double[] _beta_avg, double[] _beta_expl, double _gamma_1, double _gamma_2, double _gamma_3, double
       _gamma_4, double _gamma_5, double _lambda_vel, double _lambda_max, double _lambda_avg, int[] _theta, double _x, double _y, double
       _x_expl, double _y_expl, double _gamma_acc, double _gamma_dec, double _v, double _v_comf, double _v_max, double _v_hat_max, double
       _r_s, double _eta, double _d_x, double _d_y, double _D_x, double _D_y, Status _status, Situation _situation, Exit _exit, double[]
       _wallCoef, double _measure) { // конструктор
      this.person = _person;
      this.a0 = _a0;
      this.b0 = _b0;
      this.len1 = _len1;
      this.len2 = _len2;
      this.b11 = _b11;
      this.b12 = _b12;
      this.b21 = _b21;
      this.b22 = _b22;
      this.index = _index;
      this.beta_occ = _beta_occ;
      this.beta_dir = _beta_dir;
      this.beta_dest = _beta_dest;
      this.beta_angle = _beta_angle;
      this.beta_wall = _beta_wall;
      this.beta_vel = _beta_vel;
      this.beta_max = _beta_max;
      this.beta_avg = _beta_avg;
      this.beta_expl = _beta_expl;
      this.gamma_1 = _gamma_1;
      this.gamma_2 = _gamma_2;
      this.gamma_3 = _gamma_3;
      this.gamma_4 = _gamma_4;
      this.gamma_5 = _gamma_5;
      this.lambda_vel = _lambda_vel;
      this.lambda_max = _lambda_max;
      this.lambda_avg = _lambda_avg;
      this.theta = _theta;
      this.x = _x;
      this.y = _y;
      this.x_expl = _x_expl;
      this.y_expl = _y_expl;
      this.gamma_acc = _gamma_acc;
      this.gamma_dec = _gamma_dec;
      this.v = _v;
      this.v_comf = _v_comf;
      this.v_max = _v_max;
      this.v_hat_max = _v_hat_max;
      this.r_s = _r_s;
      this.eta = _eta;
      this.d_x = _d_x;
      this.d_y = _d_y;
      this.D_x = _D_x;
      this.D_y = _D_y;
      this.status = _status;
      this.situation = _situation;
      this.exit = _exit;
      this.wallCoef = _wallCoef;
      this.measure = _measure;
      initialization();
    }

    private void initialization() {

      final int theta_max = theta[situation.ordinal()];

      int i,j;

      d = new MyVector(d_x, d_y); // вектор направления движения
      D = new MyVector(D_x, D_y); // вектор направления выхода
      v_new = new double[3]; // массив возможных скоростей

      v_new[0] = (1-gamma_dec)*v;
      v_new[1] = v;
```

```
    if (situation==Situation.lack) {
      v_new[2] = v + gamma_acc*(v_max-v);
    } else {
      v_new[2] = v + gamma_acc*(v_hat_max-v);
    }

    switch (theta_max) { // построение массива секторов
      case phi_0:
        d_new = new MyVector[11];
        d_new[0] = d;
        d_new[1] = d.rotateDegree(10);
        d_new[2] = d.rotateDegree(-10);
        d_new[3] = d.rotateDegree(20);
        d_new[4] = d.rotateDegree(-20);
        d_new[5] = d.rotateDegree(32.5);
        d_new[6] = d.rotateDegree(-32.5);
        d_new[7] = d.rotateDegree(50);
        d_new[8] = d.rotateDegree(-50);
        d_new[9] = d.rotateDegree(72.5);
        d_new[10] = d.rotateDegree(-72.5);
        break;
      case phi_1:
        d_new = new MyVector[9];
        d_new[0] = d;
        d_new[1] = d.rotateDegree(10);
        d_new[2] = d.rotateDegree(-10);
        d_new[3] = d.rotateDegree(20);
        d_new[4] = d.rotateDegree(-20);
        d_new[5] = d.rotateDegree(32.5);
        d_new[6] = d.rotateDegree(-32.5);
        d_new[7] = d.rotateDegree(50);
        d_new[8] = d.rotateDegree(-50);
        break;
      case phi_2:
        d_new = new MyVector[7];
        d_new[0] = d;
        d_new[1] = d.rotateDegree(10);
        d_new[2] = d.rotateDegree(-10);
        d_new[3] = d.rotateDegree(20);
        d_new[4] = d.rotateDegree(-20);
        d_new[5] = d.rotateDegree(32.5);
        d_new[6] = d.rotateDegree(-32.5);
        break;
      case phi_3:
        d_new = new MyVector[5];
        d_new[0] = d;
        d_new[1] = d.rotateDegree(10);
        d_new[2] = d.rotateDegree(-10);
        d_new[3] = d.rotateDegree(20);
        d_new[4] = d.rotateDegree(-20);
        break;
      case phi_4:
        d_new = new MyVector[3];
        d_new[0] = d;
        d_new[1] = d.rotateDegree(10);
        d_new[2] = d.rotateDegree(-10);
        break;
      case phi_5:
        d_new = new MyVector[1];
        d_new[0] = d;
        break;
    }
    coordinate = new MyVector[d_new.length][3]; // массив точек перехода
    bad = new boolean[d_new.length][3]; // массив проверок допустимости точек перехода
    cross = new int[d_new.length][3]; // массив проверок на выход из помещения (0 - вышел)
    for (i=0;i<d_new.length;i++) {
      for (j=0;j<3;j++) {
        coordinate[i][j] = d_new[i].coefVector(measure*v_new[j]); // вычисление точек перехода
        bad[i][j] = closeToWall(coordinate[i][j])||crossWall(coordinate[i][j]); // вычисление допустимости точек перехода
        if (crossExit(coordinate[i][j])) { // вычисление проверок на выход из помещения
          cross[i][j] = 0;
        } else {
          cross[i][j] = 1;
        }
      }
    }
  }

  private boolean closeToWall(MyVector q) { // проверка близости к стенам

    double x_new = q.getX()+x;
    double y_new = q.getY()+y;
```

```
    return ((x_new>a0-r_s*wallCoef[situation.ordinal()])&&(x_new<a0+len1+r_s*wallCoef[situation.ordinal()])&&(y_new>b0-r_s*wallCoef[
      situation.ordinal()])&&(y_new<b0+len2+r_s*wallCoef[situation.ordinal()])&&
        (((x_new<a0+r_s*wallCoef[situation.ordinal()])&&((y_new<b11+r_s*wallCoef[situation.ordinal()])||(y_new>b12-r_s*wallCoef[situation.
      ordinal()])))||((x_new>a0+len1-r_s*wallCoef[situation.ordinal()])&&
        ((y_new<b21+r_s*wallCoef[situation.ordinal()])||(y_new>b22-r_s*wallCoef[situation.ordinal()]))))||(y_new<b0+r_s*wallCoef[situation.
      ordinal()])||(y_new>b0+len2-r_s*wallCoef[situation.ordinal()]));
  }

  private boolean crossWall(MyVector q) { // проверка пересечения стен

    double x_new = q.getX()+x;
    double y_new = q.getY()+y;

    return ((MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0-r_s*wallCoef[situation.ordinal()], b0), new
      MyPoint(a0-r_s*wallCoef[situation.ordinal()], b11+r_s*wallCoef[situation.ordinal()])))||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0-r_s*wallCoef[situation.ordinal()], b11+r_s*
      wallCoef[situation.ordinal()]), new MyPoint(a0+r_s*wallCoef[situation.ordinal()], b11+r_s*wallCoef[situation.ordinal()])))||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0+r_s*wallCoef[situation.ordinal()], b11+r_s*
      wallCoef[situation.ordinal()]), new MyPoint(a0+r_s*wallCoef[situation.ordinal()], b0+r_s*wallCoef[situation.ordinal()])))||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0+r_s*wallCoef[situation.ordinal()], b0+r_s*wallCoef
      [situation.ordinal()]), new MyPoint(a0+len1-r_s*wallCoef[situation.ordinal()], b0+r_s*wallCoef[situation.ordinal()])))||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0+len1-r_s*wallCoef[situation.ordinal()], b0+r_s*
      wallCoef[situation.ordinal()]), new MyPoint(a0+len1-r_s*wallCoef[situation.ordinal()], b21+r_s*wallCoef[situation.ordinal()])))||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0+len1-r_s*wallCoef[situation.ordinal()], b21+r_s*
      wallCoef[situation.ordinal()]), new MyPoint(a0+len1+r_s*wallCoef[situation.ordinal()], b21+r_s*wallCoef[situation.ordinal()])))||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0+len1+r_s*wallCoef[situation.ordinal()], b21+r_s*
      wallCoef[situation.ordinal()]), new MyPoint(a0+len1+r_s*wallCoef[situation.ordinal()], b0)))||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0-r_s*wallCoef[situation.ordinal()], b0+len2), new
      MyPoint(a0-r_s*wallCoef[situation.ordinal()], b12-r_s*wallCoef[situation.ordinal()])))||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0-r_s*wallCoef[situation.ordinal()], b12-r_s*
      wallCoef[situation.ordinal()]), new MyPoint(a0+r_s*wallCoef[situation.ordinal()], b12-r_s*wallCoef[situation.ordinal()])))||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0+r_s*wallCoef[situation.ordinal()], b12-r_s*
      wallCoef[situation.ordinal()]), new MyPoint(a0+r_s*wallCoef[situation.ordinal()], b0+len2-r_s*wallCoef[situation.ordinal()])))||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0+r_s*wallCoef[situation.ordinal()], b0+len2-r_s*
      wallCoef[situation.ordinal()]), new MyPoint(a0+len1-r_s*wallCoef[situation.ordinal()], b0+len2-r_s*wallCoef[situation.ordinal()])))
      ||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0+len1-r_s*wallCoef[situation.ordinal()], b0+len2-
      r_s*wallCoef[situation.ordinal()]), new MyPoint(a0+len1-r_s*wallCoef[situation.ordinal()], b22-r_s*wallCoef[situation.ordinal()])))
      ||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0+len1-r_s*wallCoef[situation.ordinal()], b22-r_s*
      wallCoef[situation.ordinal()]), new MyPoint(a0+len1+r_s*wallCoef[situation.ordinal()], b22-r_s*wallCoef[situation.ordinal()])))||
        (MyPoint.intersect(new MyPoint(x, y), new MyPoint(x_new, y_new), new MyPoint(a0+len1+r_s*wallCoef[situation.ordinal()], b22-r_s*
      wallCoef[situation.ordinal()]), new MyPoint(a0+len1+r_s*wallCoef[situation.ordinal()], b0+len2))));
  }

  private boolean crossExit(MyVector q) { // проверка на выход из помещения

    double x_new = q.getX()+x;
    double y_new = q.getY()+y;

    return ((x_new<a0+r_s*wallCoef[situation.ordinal()])||(x_new>a0+len1-r_s*wallCoef[situation.ordinal()]));
  }

  private int indicator(int sector, int index_j) { // проверка присутствия другого агента взаданном секторе
    int i = 0;
    MyVector index_vec = new MyVector(person.get(index_j).getX(), person.get(index_j).getY(), x, y);

    switch (sector) {
      case 0:
        if ((index_vec.length()<=eta*r_s)&&(MyVector.angleDegree(d, index_vec)<=phi_5/2)) {
          i = 1;
        }
        break;
      case 1:
        if ((index_vec.length()<=eta*r_s)&&(MyVector.angleDegree(d, index_vec)>=phi_5/2)&&(MyVector.angleDegree(d, index_vec)<=phi_4/2)&&(
      MyVector.crossProduct(d, index_vec)>0)) {
          i = 1;
        }
        break;
      case 2:
        if ((index_vec.length()<=eta*r_s)&&(MyVector.angleDegree(d, index_vec)>=phi_5/2)&&(MyVector.angleDegree(d, index_vec)<=phi_4/2)&&(
      MyVector.crossProduct(d, index_vec)<0)) {
          i = 1;
        }
        break;
      case 3:
        if ((index_vec.length()<=eta*r_s)&&(MyVector.angleDegree(d, index_vec)>=phi_4/2)&&(MyVector.angleDegree(d, index_vec)<=phi_3/2)&&(
      MyVector.crossProduct(d, index_vec)>0)) {
          i = 1;
        }
        break;
```

```
      case 4:
        if ((index_vec.length()<=eta*r_s)&&(MyVector.angleDegree(d, index_vec)>=phi_4/2)&&(MyVector.angleDegree(d, index_vec)<=phi_3/2)&&(MyVector.crossProduct(d, index_vec)<0)) {
          i = 1;
        }
        break;
      case 5:
        if ((index_vec.length()<=eta*r_s)&&(MyVector.angleDegree(d, index_vec)>=phi_3/2)&&(MyVector.angleDegree(d, index_vec)<=phi_2/2)&&(MyVector.crossProduct(d, index_vec)>0)) {
          i = 1;
        }
        break;
      case 6:
        if ((index_vec.length()<=eta*r_s)&&(MyVector.angleDegree(d, index_vec)>=phi_3/2)&&(MyVector.angleDegree(d, index_vec)<=phi_2/2)&&(MyVector.crossProduct(d, index_vec)<0)) {
          i = 1;
        }
        break;
      case 7:
        if ((index_vec.length()<=eta*r_s)&&(MyVector.angleDegree(d, index_vec)>=phi_2/2)&&(MyVector.angleDegree(d, index_vec)<=phi_1/2)&&(MyVector.crossProduct(d, index_vec)>0)) {
          i = 1;
        }
        break;
      case 8:
        if ((index_vec.length()<=eta*r_s)&&(MyVector.angleDegree(d, index_vec)>=phi_2/2)&&(MyVector.angleDegree(d, index_vec)<=phi_1/2)&&(MyVector.crossProduct(d, index_vec)<0)) {
          i = 1;
        }
        break;
      case 9:
        if ((index_vec.length()<=eta*r_s)&&(MyVector.angleDegree(d, index_vec)>=phi_1/2)&&(MyVector.angleDegree(d, index_vec)<=phi_0/2)&&(MyVector.crossProduct(d, index_vec)>0)) {
          i = 1;
        }
        break;
      case 10:
        if ((index_vec.length()<=eta*r_s)&&(MyVector.angleDegree(d, index_vec)>=phi_1/2)&&(MyVector.angleDegree(d, index_vec)<=phi_0/2)&&(MyVector.crossProduct(d, index_vec)<0)) {
          i = 1;
        }
        break;
    }
    return i;
  }

  private double[][] occupation(MyVector[][] r) { // массив значений слагаемого функционала

    int i,j;
    double[][] occupation = new double[r.length][r[0].length];

    for (i=0;i<r.length;i++) {
      for (j=0;j<r[0].length;j++) {
        occupation[i][j] = 0;
        for (Person p : person) {
          if (p.getIndex()!=index) {
            occupation[i][j] = occupation[i][j] + indicator(i, p.getIndex())*Math.exp(-gamma_1*r[i][j].distance(new MyVector(p.getX(), p.getY(), x, y)));
          }
        }
      }
    }
    return occupation;
  }

  private double[][] angle(MyVector[][] r) { // массив значений слагаемого функционала

    int i,j;
    double[][] angle = new double[r.length][r[0].length];

    for (i=0;i<r.length;i++) {
      for (j=0;j<r[0].length;j++) {
        angle[i][j] = 0;
        for (Person p : person) {
          if (p.getIndex()!=index) {
            angle[i][j] = angle[i][j] + indicator(i, p.getIndex())*MyVector.angleRadian(d, new MyVector(p.d_x, p.d_y))*Math.exp(-gamma_2*r[i][j].distance(new MyVector(p.getX(), p.getY(), x, y)));
          }
        }
      }
    }
    return angle;
```

```
}

private double[][] direction(MyVector[][] r) { // массив значений слагаемого функционала

  int i,j;
  double[][] direction = new double[r.length][r[0].length];

  for (i=0;i<r.length;i++) {
    for (j=0;j<r[0].length;j++) {
      direction[i][j] = MyVector.angleRadian(r[i][j], d);
    }
  }
  return direction;
}

private double[][] destination(MyVector[][] r) { // массив значений слагаемого функционала

  int i,j;
  double[][] destination = new double[r.length][r[0].length];

  for (i=0;i<r.length;i++) {
    for (j=0;j<r[0].length;j++) {
      destination[i][j] = MyVector.angleRadian(r[i][j], D);
    }
  }
  return destination;
}

private double[][] vel(MyVector[][] r) { // массив значений слагаемого функционала

  int i,j;
  double[][] vel = new double[r.length][r[0].length];

  for (i=0;i<r.length;i++) {
    for (j=0;j<r[0].length;j++) {
      vel[i][j] = Math.pow(Math.abs(v_new[j]-v_comf), lambda_vel);
    }
  }
  return vel;
}

private double[][] max(MyVector[][] r) { // массив значений слагаемого функционала

  int i,j;
  double[][] max = new double[r.length][r[0].length];

  for (i=0;i<r.length;i++) {
    for (j=0;j<r[0].length;j++) {
      max[i][j] = Math.pow(Math.abs(v_new[j]-v_hat_max), lambda_max);
    }
  }
  return max;
}

private double[][] vAvg(MyVector[][] r) {

  int i,j;
  double[][] vSum = new double[r.length][r[0].length];
  double[][] vIndicator = new double[r.length][r[0].length];
  double[][] vAvg = new double[r.length][r[0].length];

  for (i=0;i<r.length;i++) {
    for (j=0;j<r[0].length;j++) {
      vSum[i][j] = 0;
      vIndicator[i][j] = 0;
      for (Person p : person) {
        if (p.getIndex()!=index) {
          vSum[i][j] = vSum[i][j] + indicator(i, p.getIndex())*p.getVelocity()*Math.exp(-gamma_4*r[i][j].distance(new MyVector(p.getX(),
    p.getY(), x, y)));
          vIndicator[i][j] = vIndicator[i][j] + indicator(i, p.getIndex());
        }
      }
      if ((vIndicator[i][j]==0)&&(vSum[i][j]==0)) {
        vAvg[i][j] = v;
      } else {
        vAvg[i][j] = vSum[i][j]/vIndicator[i][j];
      }
    }
  }
  return vAvg;
}
```

```
  private double[][] avg(MyVector[][] r) { // массив значений слагаемого функционала

440   int i,j;
    double[][] avg = new double[r.length][r[0].length];
    double[][] vAvg = new double[r.length][r[0].length];

    vAvg = vAvg(r);
445
    for (i=0;i<r.length;i++) {
      for (j=0;j<r[0].length;j++) {
        avg[i][j] = Math.pow(Math.abs(v_new[j]-vAvg[i][j]), lambda_avg);
      }
450   }
    return avg;
  }

  private double[][] explosion(MyVector[][] r) { // массив значений слагаемого функционала
455
    int i,j;
    double[][] explosion = new double[r.length][r[0].length];

    if (situation==Situation.lack) {
460   for (i=0;i<r.length;i++) {
        for (j=0;j<r[0].length;j++) {
          explosion[i][j] = 0;
        }
      }
465 } else {
      for (i=0;i<r.length;i++) {
        for (j=0;j<r[0].length;j++) {
          explosion[i][j] = Math.exp(-gamma_5*r[i][j].distance(new MyVector(x_expl, y_expl, x, y)));
        }
470   }
    }
    return explosion;
  }

475 private double[][] wall(MyVector[][] r) { // массив значений слагаемого функционала

    int i,j;
    double l; // минимальное расстояние до стен
    double l1,l2,l3,l4; // расстояния до стен
480 double[][] wall = new double[r.length][r[0].length];

    for (i=0;i<r.length;i++) {
      for (j=0;j<r[0].length;j++) {
        MyVector end = r[i][j].plus(new MyVector(x, y)); // переход в точку положения агента
485     l1 = MyVector.distance(end, new MyVector(a0, end.getY())); // расстояние до левой стены
        l2 = MyVector.distance(end, new MyVector(end.getX(), b0)); // расстояние до верхней стены
        l3 = MyVector.distance(end, new MyVector(a0+len1, end.getY())); // расстояние до правой стены
        l4 = MyVector.distance(end, new MyVector(end.getX(), b0+len2)); // расстояние до нижней стены
        if ((end.getY()>b11)&&(end.getY()<b12)) { // расстояние до левой стены при нахождении в растворе выхода
490       l1 = Math.min(MyVector.distance(end, new MyVector(a0, b11)),MyVector.distance(end, new MyVector(a0, b12)));
        }
        if ((end.getY()>b21)&&(end.getY()<b22)) {// расстояние до правой стены при нахождении в растворе выхода
          l3 = Math.min(MyVector.distance(end, new MyVector(a0+len1, b21)),MyVector.distance(end, new MyVector(a0+len1, b22)));
        }
495     l = Math.min(Math.min(l1,l2), Math.min(l3,l4)); // вычисление минимального расстояния до стен
        wall[i][j] = Math.exp(-gamma_3*l);
      }
    }
    return wall;
500 }

  private void minimize(MyVector[][] r) {

    int i,j,k;
505
    min = Double.POSITIVE_INFINITY;

    L = new double[r.length][r[0].length]; // массив значений функционала
    double[][] occupation = new double[r.length][r[0].length]; // массив значений слагаемого функционала
510 double[][] direction = new double[r.length][r[0].length]; // массив значений слагаемого функционала
    double[][] destination = new double[r.length][r[0].length]; // массив значений слагаемого функционала
    double[][] angle = new double[r.length][r[0].length]; // массив значений слагаемого функционала
    double[][] wall = new double[r.length][r[0].length]; // массив значений слагаемого функционала
    double[][] vel = new double[r.length][r[0].length]; // массив значений слагаемого функционала
515 double[][] max = new double[r.length][r[0].length]; // массив значений слагаемого функционала
    double[][] avg = new double[r.length][r[0].length]; // массив значений слагаемого функционала
    double[][] explosion = new double[r.length][r[0].length]; // массив значений слагаемого функционала

    occupation = occupation(r); // расчет массива значений слагаемого функционала
```

```
520     direction = direction(r); // расчет массива значений слагаемого функционала
        destination = destination(r); // расчет массива значений слагаемого функционала
        angle = angle(r); // расчет массива значений слагаемого функционала
        wall = wall(r); // расчет массива значений слагаемого функционала
        vel = vel(r); // расчет массива значений слагаемого функционала
525     max = max(r); // расчет массива значений слагаемого функционала
        avg = avg(r); // расчет массива значений слагаемого функционала
        explosion = explosion(r); // расчет массива значений слагаемого функционала

        for (i=0;i<r.length;i++) { // расчет всех значений функционала
530       for (j=0;j<r[0].length;j++) {
            L[i][j] = beta_occ[situation.ordinal()]*occupation[i][j]+beta_dir[situation.ordinal()]*direction[i][j]+cross[i][j]*beta_dest[
          situation.ordinal()]*destination[i][j]+
                beta_angle[situation.ordinal()]*angle[i][j]+beta_wall[situation.ordinal()]*wall[i][j]+beta_vel[situation.ordinal()]*vel[i][j]+
                beta_max[situation.ordinal()]*max[i][j]+beta_avg[situation.ordinal()]*avg[i][j]+beta_expl[situation.ordinal()]*explosion[i][j
          ];
            if ((L[i][j]<min)&&(!bad[i][j])) { // проверка убывания значения и его допустимости
535           min = L[i][j]; // значение промежуточного минимума функционала
              imin = i; // аргумент промежуточного минимума функционала
              jmin = j; // аргумент промежуточного минимума функционала
            }
          }
540     }
        if (imin>=0) { // проверка достижимости минимального значения
        } else {
          MessageBox.WarningBox(Integer.toString(imin), "L нет значений");
        }
545   }

      public Result newDirection() {

        minimize(coordinate); // запуск расчета минимизации функционала
550     return new Result(d_new[imin], v_new[jmin]);
      }

      /**
       * Это число используется при сохранении состояния модели<br>
555    * Его рекомендуется изменить в случае изменения класса
       */
      private static final long serialVersionUID = 1L;

    }
```

Приложение 3

Класс точек пространства

3.1 Листинг кода класса точек пространства

```
public class MyPoint implements Serializable {

  private double x; // x–координата точки
  private double y; // y–координата точки

    public MyPoint() { // конструктор по умолчанию
      this(0, 0);
    }

    public MyPoint(double x, double y) { // конструктор
      this.x = x;
      this.y = y;
    }

    public double getX() { // возвращение x–координаты точки
        return x;
    }

    public void setX(double x) { // установка x–координаты точки
        this.x = x;
    }

    public double getY() { // возвращение y–координаты точки
        return y;
    }

    public void setY(double y) { // установка y–координаты точки
        this.y = y;
    }

    @Override
    public boolean equals(Object o) { // переопределение метода equals()
        if (this == o) return true;
        if (!(o instanceof MyPoint)) return false;

        MyPoint point = (MyPoint) o;

        if (Double.compare(point.x, this.x) != 0) return false;
        if (Double.compare(point.y, this.y) != 0) return false;

        return true;
    }

    @Override
    public int hashCode() { // переопределение метода hashCode()
        int result;
        long temp;
        temp = x != +0.0d ? Double.doubleToLongBits(this.x) : 0L;
        result = (int) (temp ^ (temp >>> 32));
        temp = y != +0.0d ? Double.doubleToLongBits(this.y) : 0L;
        result = 31 * result + (int) (temp ^ (temp >>> 32));
        return result;
    }

  @Override
  public String toString() { // переопределение метода toString()
    return "( "+Double.toString(this.getX())+" , "+Double.toString(this.getY())+" )";
  }

  private static double area(MyPoint a, MyPoint b, MyPoint c) { // вспомогательный метод для intersect()
    return (b.getX()–a.getX())*(c.getY()–a.getY())–(b.getY()–a.getY())*(c.getX()–a.getX());
  }

  private static boolean boundingBox(double a, double b, double c, double d) { // вспомогательный метод для intersect()
```

```
    double k;
    if (a>b) {
      k = a;
      a = b;
      b = k;
    }
    if (c>d) {
      k = c;
      c = d;
      d = k;
    }
    return (max(a,c)<=min(b,d));
  }

  public static boolean intersect(MyPoint a, MyPoint b, MyPoint c, MyPoint d) { // проверка отрезков (a,b) и (c,d) на пересечение
    return (boundingBox(a.getX(),b.getX(),c.getX(),d.getX())&&
        boundingBox(a.getY(),b.getY(),c.getY(),d.getY())&&
        (area(a,b,c)*area(a,b,d)<=0)&&(area(c,d,a)*area(c,d,b)<=0));
  }

  /**
   * Это число используется при сохранении состояния модели
   * Его рекомендуется изменить в случае изменения класса
   */
  private static final long serialVersionUID = 1L;

}
```

Приложение И

Класс векторов пространства

И.1 Листинг кода класса векторов пространства

```
import java.math.*;

public class MyVector implements Serializable {

  private double x; // x–координата вектора
  private double y; // y–координата вектора

  public MyVector() { // конструктор по умолчанию

  }

  public MyVector(double x, double y) { // конструктор по координатам вектора
    setX(x);
    setY(y);
  }

  public MyVector(double x1, double y1, double x2, double y2) { // конструктор по координатам точек
    setX(x1–x2);
    setY(y1–y2);
  }

  public double getX() { // возвращение x–координаты вектора
    return x;
  }

  public double getY() { // возвращение y–координаты вектора
    return y;
  }

  public void setX(double x) { // установка x–координаты вектора
    this.x = x;
  }

  public void setY(double y) { // установка y–координаты точки
    this.y = y;
  }

  public static boolean isNull(MyVector v) { // проверка на нуль–вектор
    return (v.getX()==0)&&(v.getY()==0);
  }

  public boolean isNull() { // проверка на нуль–вектор
    return (this.getX()==0)&&(this.getY()==0);
  }

  public static MyVector unitVector(MyVector v) { // преобразование в единичный вектор
    if (isNull(v)) {
      return v;
    } else {
      return factorVector(v, 1/length(v));
    }
  }

  public MyVector unitVector() { // преобразование в единичный вектор
    if (isNull(this)) {
      return this;
    } else {
      return factorVector(this, 1/length(this));
    }
  }

  public static MyVector coefVector(MyVector v, double k) { // преобразование в вектор заданной длины
    if (isNull(v)) {
      return v;
```

```
    } else {
      return factorVector(unitVector(v), k);
    }
  }

  public MyVector coefVector(double k) { // преобразование в вектор заданной длины
    if (isNull(this)) {
      return this;
    } else {
      return factorVector(unitVector(this), k);
    }
  }

  public static MyVector factorVector(MyVector v, double k) { // умножение вектора на скаляр
    return new MyVector(k*v.getX(),k*v.getY());
  }

  public MyVector factorVector(double k) { // умножение вектора на скаляр
    return new MyVector(k*this.getX(),k*this.getY());
  }

  public static MyVector opposite(MyVector v) { // обратный вектор
    return factorVector(v, -1);
  }

  public MyVector opposite() { // обратный вектор
    return factorVector(this, -1);
  }

  public static MyVector createPolar(double rho, double theta) { // построение вектора из полярных координат
    return new MyVector(rho*Math.cos(theta),rho*Math.sin(theta));
  }

  public static MyVector plus(MyVector v1, MyVector v2) { // сложение веекторов
    return new MyVector(v1.getX()+v2.getX(),v1.getY()+v2.getY());
  }

  public MyVector plus(MyVector v2) { // сложение веекторов
    return new MyVector(this.getX()+v2.getX(),this.getY()+v2.getY());
  }

  public static MyVector minus(MyVector v1, MyVector v2) { // разность веекторов
    return new MyVector(v1.getX()-v2.getX(),v1.getY()-v2.getY());
  }

  public MyVector minus(MyVector v2) { // разность веекторов
    return new MyVector(this.getX()-v2.getX(),this.getY()-v2.getY());
  }

  public static MyVector rotateRadian(MyVector v, double theta) { // поворот вектора в радианах
    return new MyVector(v.getX()*Math.cos(theta)-v.getY()*Math.sin(theta),v.getX()*Math.sin(theta)+v.getY()*Math.cos(theta));
  }

  public MyVector rotateRadian(double theta) { // поворот вектора в радианах
    return new MyVector(this.getX()*Math.cos(theta)-this.getY()*Math.sin(theta),this.getX()*Math.sin(theta)+this.getY()*Math.cos(theta));
  }

  public static MyVector rotateDegree(MyVector v, double theta) { // поворот вектора в градусах
    return new MyVector(v.getX()*Math.cos(theta*Math.PI/180)-v.getY()*Math.sin(theta*Math.PI/180),v.getX()*Math.sin(theta*Math.PI/180)+v.
      getY()*Math.cos(theta*Math.PI/180));
  }

  public MyVector rotateDegree(double theta) { // поворот вектора в градусах
    return new MyVector(this.getX()*Math.cos(theta*Math.PI/180)-this.getY()*Math.sin(theta*Math.PI/180),this.getX()*Math.sin(theta*Math.PI
      /180)+this.getY()*Math.cos(theta*Math.PI/180));
  }

  public static double scalarProduct(MyVector v1, MyVector v2) { // скалярное произведение векторов
    return v1.getX()*v2.getX()+v1.getY()*v2.getY();
  }

  public double scalarProduct(MyVector v2) { // скалярное произведение векторов
    return this.getX()*v2.getX()+this.getY()*v2.getY();
  }

  public static double crossProduct(MyVector v1, MyVector v2) { // косое произведение векторов
    return v1.getX()*v2.getY()-v1.getY()*v2.getX();
  }

  public double crossProduct(MyVector v2) { // косое произведение векторов
    return this.getX()*v2.getY()-this.getY()*v2.getX();
  }
```

```
  public static double length(MyVector v) { // длина вектора
    return Math.sqrt(lengthSquared(v));
  }

  public double length() { // длина вектора
    return Math.sqrt(lengthSquared(this));
  }

  public static double lengthSquared(MyVector v) { // квадрат длины вектора
    return Math.pow(v.getX(),2)+Math.pow(v.getY(),2);
  }

  public double lengthSquared() { // квадрат длины вектора
    return Math.pow(this.getX(),2)+Math.pow(this.getY(),2);
  }

  public static double cosAngle(MyVector v1, MyVector v2) { // косинус угла между векторами (без модуля)
    return scalarProduct(v1, v2)/(length(v1)*length(v2));
  }

  public double cosAngle(MyVector v2) { // косинус угла между векторами (без модуля)
    return scalarProduct(this, v2)/(length(this)*length(v2));
  }

  public static double sinAngle(MyVector v1, MyVector v2) { // синус угла между векторами
    return Math.sqrt(1-Math.pow(cosAngle(v1, v2),2));
  }

  public double sinAngle(MyVector v2) { // синус угла между векторами
    return Math.sqrt(1-Math.pow(cosAngle(this, v2),2));
  }

  public static double angleRadian(MyVector v1, MyVector v2) { // угол (неориентированный) между векторами в радианах
    return Math.acos(cosAngle(v1, v2));
  }

  public double angleRadian(MyVector v2) { // угол (неориентированный) между векторами в радианах
    return Math.acos(cosAngle(this, v2));
  }

  public static double angleDegree(MyVector v1, MyVector v2) { // угол (неориентированный) между векторами в градусах
    return Math.acos(cosAngle(v1, v2))*180/Math.PI;
  }

  public double angleDegree(MyVector v2) { // угол (неориентированный) между векторами в градусах
    return Math.acos(cosAngle(this, v2))*180/Math.PI;
  }

  public static double distance(MyVector v1, MyVector v2) { // расстояние между концами векторов
    return length(new MyVector(v1.getX(), v1.getY(), v2.getX(), v2.getY()));
  }

  public double distance(MyVector v2) { // расстояние между концами векторов
    return length(new MyVector(this.getX(), this.getY(), v2.getX(), v2.getY()));
  }

  public static double cosAngleOx(MyVector v) { // косинус угла с положительным направлением оси Ox
    if (length(v)!=0) {
      return v.getX()/length(v);
    } else {
      return Double.POSITIVE_INFINITY;
    }
  }

  public double cosAngleOx() { // косинус угла с положительным направлением оси Ox
    if (length(this)!=0) {
      return this.getX()/length(this);
    } else {
      return Double.POSITIVE_INFINITY;
    }
  }

  public static double sinAngleOx(MyVector v) { // синус угла с положительным направлением оси Ox
    if (length(v)!=0) {
      return v.getY()/length(v);
    } else {
      return Double.POSITIVE_INFINITY;
    }
  }

  public double sinAngleOx() { // синус угла с положительным направлением оси Ox
```

```
    if (length(this)!=0) {
      return this.getY()/length(this);
    } else {
      return Double.POSITIVE_INFINITY;
    }
  }

  public static double angleRadianOx(MyVector v) { // угол (ориентированный) в радианах с положительным направлением оси Ox
    if (v.getY()>=0) {
      return Math.acos(cosAngleOx(v));
    } else {
      return -Math.acos(cosAngleOx(v));
    }
  }

  public double angleRadianOx() { // угол (ориентированный) в радианах с положительным направлением оси Ox
    if (this.getY()>=0) {
      return Math.acos(cosAngleOx(this));
    } else {
      return -Math.acos(cosAngleOx(this));
    }
  }

  public static double angleDegreeOx(MyVector v) { // угол (ориентированный) в градусах с положительным направлением оси Ox
    return angleRadianOx(v)*180/Math.PI;
  }

  public double angleDegreeOx() { // угол (ориентированный) в градусах с положительным направлением оси Ox
    return angleRadianOx(this)*180/Math.PI;
  }

  public static boolean isEquals(MyVector v1, MyVector v2) { // проверка на равенство векторов
    if ((v1.getX()==v2.getX())&&(v1.getY()==v2.getY())) {
      return true;
    } else {
      return false;
    }
  }

  public boolean isEquals(MyVector v2) { // проверка на равенство векторов
    if ((this.getX()==v2.getX())&&(this.getY()==v2.getY())) {
      return true;
    } else {
      return false;
    }
  }

  public static boolean isColinear(MyVector v1, MyVector v2) { // проверка на колинеарность
    if (v1.getX()/v2.getX()==v1.getY()/v2.getY()) {
      return true;
    } else {
      return false;
    }
  }

  public boolean isColinear(MyVector v2) { // проверка на колинеарность
    if (this.getX()/v2.getX()==this.getY()/v2.getY()) {
      return true;
    } else {
      return false;
    }
  }

  public static boolean isOrthogonal(MyVector v1, MyVector v2) { // проверка на ортогональность
    if (scalarProduct(v1, v2)==0) {
      return true;
    } else {
      return false;
    }
  }

  public boolean isOrthogonal(MyVector v2) { // проверка на ортогональность
    if (scalarProduct(this, v2)==0) {
      return true;
    } else {
      return false;
    }
  }

  public static String toString(MyVector v) { // переопределение метода toString()

    return "( "+Double.toString(v.getX())+" , "+Double.toString(v.getY())+" )";
```

```
310   }

      @Override
      public String toString() { // переопределение метода toString()

315     return "( "+Double.toString(this.getX())+" , "+Double.toString(this.getY())+" )";
      }

      @Override
        public boolean equals(Object o) { // переопределение метода equals()
320         if (this == o) return true;
            if (!(o instanceof MyVector)) return false;

            MyVector vector = (MyVector) o;

325         if (Double.compare(vector.x, this.x) != 0) return false;
            if (Double.compare(vector.y, this.y) != 0) return false;

            return true;
        }
330
        @Override
        public int hashCode() { // переопределение метода hashCode()
            int result;
            long temp;
335         temp = this.x != +0.0d ? Double.doubleToLongBits(this.x) : 0L;
            result = (int) (temp ^ (temp >>> 32));
            temp = this.y != +0.0d ? Double.doubleToLongBits(this.y) : 0L;
            result = 31 * result + (int) (temp ^ (temp >>> 32));
            return result;
340     }

      /**
       * Это число используется при сохранении состояния модели
       * Его рекомендуется изменить в случае изменения класса
345    */
      private static final long serialVersionUID = 1L;

}
```

Приложение К

Класс результатов

К.1 Листинг кода класса результатов

```
public class Result implements Serializable {

  MyVector d; // вектор направления движения
  double v; // скалярное значение скорости

    public Result(MyVector _d, double _v) { // конструктор

      d = _d;
      v = _v;
    }

  public MyVector getVector() { // возвращение вектора направления движения
    return d;
  }

  public double getX() { // возвращение x-координаты вектора направления движения
    return d.unitVector().getX();
  }

  public double getY() { // возвращение y-координаты вектора направления движения
    return d.unitVector().getY();
  }

  public double getV() { // возвращение скалярного значения скорости
    return v;
  }

  @Override
    public boolean equals(Object o) { // переопределение метода equals()
        if (this == o) return true;
        if (!(o instanceof Result)) return false;

        Result result = (Result) o;

        if (Double.compare(result.getX(), this.getX()) != 0) return false;
        if (Double.compare(result.getY(), this.getY()) != 0) return false;
        if (Double.compare(result.getV(), this.getV()) != 0) return false;

        return true;
    }

    @Override
    public int hashCode() { // переопределение метода hashCode()
        int result;
        long temp;
        temp = d.getX() != +0.0d ? Double.doubleToLongBits(this.d.getX()) : 0L;
        result = (int) (temp ^ (temp >>> 32));
        temp = d.getY() != +0.0d ? Double.doubleToLongBits(this.d.getY()) : 0L;
        result = 31 * result + (int) (temp ^ (temp >>> 32));
        temp = v != +0.0d ? Double.doubleToLongBits(this.v) : 0L;
        result = 31 * result + (int) (temp ^ (temp >>> 32));
        return result;
    }

  @Override
  public String toString() { // переопределение метода toString()
    return "(("+Double.toString(this.getX())+","+Double.toString(this.getY())+"),"+Double.toString(this.getV())+")";
  }
  private static final long serialVersionUID = 1L;

}
```