

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
БАШКИРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. М.АКМУЛЛЫ

А.Р. Исхаков

Разработка программного тренажера в среде
многоагентного моделирования NetLogo

Учебное пособие

Уфа 2021

ББК 32.972я73

УДК 004.45

И 91

Исхаков А.Р. Разработка программного тренажера в среде многоагентного моделирования NetLogo. пособие [Текст] / А.Р. Исхаков. – Изд-во БГПУ, 2021. – 42 с.

В данном учебном пособии описывается программный продукт NetLogo разработки Center of Connected Learning MIT. Программный комплекс NetLogo является узкоспециализированным программным обеспечением для разработки многоагентных систем. Учебное пособие состоит из двух глав. Первая глава знакомит с программным комплексом NetLogo, с историей возникновения его языка программирования и пользовательским интерфейсом приложения NetLogo в составе данного комплекса. Вторая глава направлена на формирование профессиональной компетенции по разработке программных тренажеров на базе многоагентной системы. Особенностью второй главы является полное описание разработки программного тренажера по оптимальному решению одной из актуальных задач военного дела. Приложение учебного пособия содержит полный код программного тренажера.

Предназначено для бакалавров и магистров, обучающихся по направлениям подготовки 09.03.02 «Информационные системы и технологии» и 09.03.03 «Прикладная информатика», а также аспирантов, инженеров, научных работников, специализирующихся в области проектирования и разработки программных и программно-аппаратных тренажеров.

Рецензенты:

Р.Ф. Маликов, д-р физ.-мат. наук, профессор БГПУ им. М.Акмиллы

О.В. Даринцев, д-р техн. наук, профессор, г.н.с. лаб. «Робототехника и управление в технических системах» Института механики им. Р.Р. Мавлютова УФИЦ РАН

© Исхаков А.Р., 2021

Список сокращений

АС – авиационная система

БПЛА – беспилотный летательный аппарат

ЗРК – зенитно-ракетный комплекс

ИИ – искусственный интеллект

МАС – многоагентная система

ПВО – противовоздушная оборона

CCL (Center for Connected Learning and Computer-Based Modeling) – Центр сетевого обучения и компьютерного моделирования

GNU GPL (General Public License) – бесплатная лицензия программного продукта

MIT (Massachusetts Institute of Technology) – Массачусетский технологический институт

СОДЕРЖАНИЕ

Введение	4
Глава 1. Среда разработки многоагентных систем NetLogo	6
1.1 История создания языка программирования NetLogo	6
1.2 Структура и интерфейс программного приложения NetLogo	7
1.3 Разработка первой программы в системе NetLogo	9
1.4 Агенты системы программирования NetLogo	13
Глава 2. Разработка программного тренажера на языке NetLogo	18
2.1 Программный тренажер по оптимальному уничтожению системы противоздушной обороны противника и наземной инфраструктуры	18
2.2 Дизайн пользовательского интерфейса программного тренажера	20
2.3 Описание используемых переменных и вывод скрина игры	22
2.4 Начальная инициализация игрового уровня в тренажере	24
2.5 Управление полетом БПЛА	26
2.6 Управление бортовым вооружением БПЛА	28
2.7 Главный цикл программного тренажера	30
Литература	34
Приложение	35

Введение

Искусственный интеллект (ИИ) в XXI веке стал одним из передовых направлений исследований. Прикладные решения, полученные с использованием технологий ИИ, нашли свое применение в различных сферах человеческой деятельности: в образовании, в медицине, в промышленности, на производстве, в военном деле и т.п. Разработка программных и программно-аппаратных тренажеров является той областью, где технологии ИИ стали незаменимым инструментом в создании эффективных прикладных решений.

Современный искусственный интеллект условно можно разделить на классическое и бионическое направление исследований. Методологию классического направления искусственного интеллекта образуют инженерия знаний и модели представления знаний, представление и решение задач посредством неинформированного и информированного поиска в пространстве состояний, языки программирования искусственного интеллекта, теория экспертных систем и т.п. Методология бионического искусственного интеллекта образована теориями нечетких и нейросетевых систем, методами эволюционного моделирования. Последними достижениями научно-технического прогресса в области ИИ стала теория машинного обучения и теория нейронных сетей глубинного обучения. С появлением этих технологий были разработаны высокоэффективные методы обучения систем искусственного интеллекта в различных областях, в том числе и в военном деле. Нечеткие системы, нейронные сети глубинного обучения и системы машинного обучения позволяют разрабатывать программные системы и тренажеры, работающие в реальном времени с высокими показателями обработки, анализа и распознавания данных.

Беспилотные летательные аппараты (БПЛА) прочно заняли свою нишу на поле современного боя. БПЛА в небе над театром военных действий (ТВД) используются в целях разведки, корректировки артиллерийского огня и для самостоятельного подавления огневых позиций противника. Для войн XXI века характерным элементом боя станет использование роя БПЛА. Исследования и разработки передовых боевых систем, использующих тактику роя для воздушных и наземных робототехнических комплексов, ведутся во многих странах мира.

Минобороны РФ на форуме «Армия-2019» впервые представила систему управления роем БПЛА для массированного удара «Стая-93». Данная система разработана Военно-учебным научным центром (ВУНЦ) имени Жуковского (г. Воронеж) ВВС России. Как отмечают разработчики, проект «Стая-93» представляет комплекс монопланов, количество которых может динамично увеличиваться в ходе решения боевой задачи. Каждый моноплан способен нести боевую нагрузку до 2,5 кг. Комплекс может иметь различную конфигурацию (топологию) сети монопланов, в которой есть ведущий и ведомые БПЛА. Связь между монопланами осуществляется через

ИК-канал. Данный комплекс обладает возможностью не только изменять свою топологию, но и, в случае уничтожения ведущего БПЛА, заменить его одним из доступных монопланов. Российские военные уже сталкивались с атаками «роя беспилотников», которые периодически атакуют российскую авиационную базу Хмеймим в Сирии. Известно, что ведущими разработчиками роя БПЛА являются США и Китай. Но пока ни одна из ведущих стран мира официально не представила реальный проект роя БПЛА, который был бы способен принимать участие в общевойсковом бою. Одной из проблем является сложность реализации роевых алгоритмов для комплекса БПЛА, которая разработана на базе определенной физической платформы. Практика применения эшелонированной системы ПВО, хоть и является тактикой прошлого века, она доказала свою эффективность во время Великой отечественной войны и локальных конфликтах на территории других стран. В России же, эшелонированную оборону военно-технических объектов еще никто не отменял. Однако, как показывает опыт, использование роя ударных БПЛА может поставить под угрозу любую подобную тактику обороны.

В связи с этим, профессиональная компетенция по разработке программных и программно-аппаратных тренажеров является ключевой в подготовке бакалавров и магистров по направлениям подготовки 09.03.02 «Информационные системы и технологии» и 09.03.03 «Прикладная информатика». Учебное пособие содержит две главы, которые закладывают основы знаний и формируют базовые умения, необходимые в разработке программных тренажеров.

В первой главе описывается история возникновения языка программирования и среды разработки NetLogo, пользовательский интерфейс приложения NetLogo и основные принципы работы с данным программным продуктом. Подробно рассмотрен пример многоагентной системы с точки зрения ее состава, применяемых команд и порядка запуска программы в среде NetLogo.

Во второй главе сформулирована задача атаки БПЛА на ЗРК противника, которая является актуальной современной задачей в военном деле. Описан пользовательский интерфейс программного тренажера, предназначенного для решения данной задачи на интуитивном уровне в ходе игрового процесса. Решение задачи представлено в виде последовательности подзадач и приведен ход их решения. Решение каждой подзадачи изучается детально с подробным описанием фрагментов кода программного тренажера.

Приложение содержит полную программу программного тренажера на языке программирования NetLogo.

Глава 1. Среда разработки многоагентных систем NetLogo

1.1. История создания языка программирования NetLogo

Язык программирования NetLogo [1] разработан на базе языка программирования Logo и является преемником языка StarLogo [2]. Язык Logo был создан в середине 60-х годов Сеймуром Папертом. Особенностью языка Logo является его ориентированность на широкий круг разработчиков. Язык среда программирования обладают интуитивно понятным и простым синтаксисом и интерфейсом соответственно. Виртуальным исполнителем языка Logo является вычислительная процедура, имеющая форму черепахи. Программирование сводится к написанию программ для черепахи, с достаточно серьезными возможностями визуализации. Исполнитель может менять свою форму и иметь неограниченное количество своих дублеров. Причем дублеры могут управляться как отдельными командами, так и программным путем независимо друг от друга.

Концепция программирования Паперта стали основой для новой системы многоагентной системы (МАС) имитационного моделирования StarLogo, предшественника современной системы многоагентного моделирования NetLogo. StarLogo разрабатывался в Массачусетском технологическом институте (Massachusetts Institute of Technology, MIT) [3]. Большим недостатком языка StarLogo была его ориентированность только на платформу Macintosh.

В 1999 г. подчиненным MIT учреждением CCL (Center of Connected Learning) была официально представлена концепция нового языка программирования и программной системы агентного моделирования NetLogo [2]. Система изначально создавалась с использованием кроссплатформенной технологии Java.

В данном учебном пособии используется версия NetLogo 6.1.1 (рис. 1).

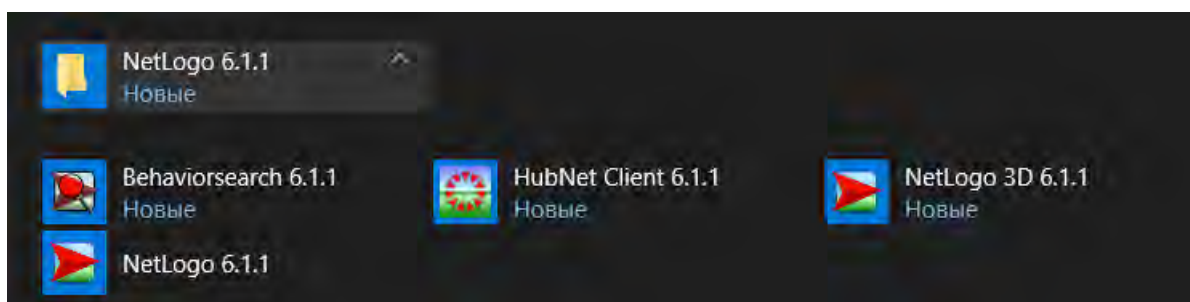


Рис. 1. Состав программного комплекса NetLogo.

Программный комплекс имеет в своем составе такие дополнительные программные продукты, как приложение Behaviorsearch 6.1.1 для работы с генетическими алгоритмами, клиентское приложение HubNetClient 6.1.1 для организации сетевой разработки и приложение Net Logo 3D 6.1.1, являющееся трехмерным аналогом NetLogo 6.1.1. Приложение NetLogo 6.1.1 является

интерпретатором, поэтому в данной среде программирования можно редактировать листинги прикладных программы и запускать их.

Программный комплекс NetLogo является бесплатным программным продуктом и распространяется по лицензии GNU GPL (General Public License) [2]. Программа многоагентной системы компилируется в Java-байткод и также распространяется бесплатно в соответствии с лицензией GNU GPL.

1.2. Структура и интерфейс программного приложения NetLogo

Интерфейс программы NetLogo содержит все необходимые элементы для разработки многоагентных систем [4-6] и их тестирования в одном приложении. Для запуска среды разработки нужно дважды кликнуть мышкой по ярлыку NetLogo, после которого начинается запуск системы с предварительной загрузкой логотипа (рис. 2).



Рис. 2. Логотип системы NetLogo при запуске.

Если же запуск среды разработки NetLogo успешно завершен, то пользователь увидит рабочую область программного приложения (рис. 3).

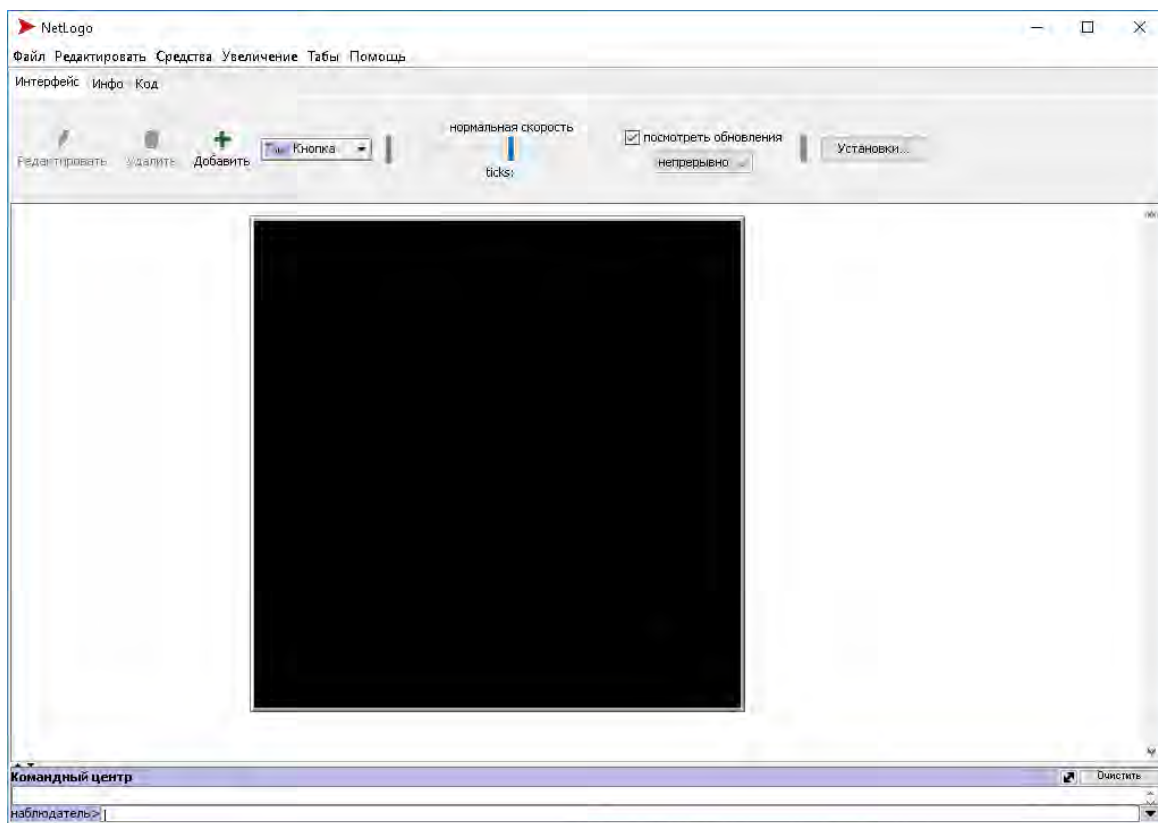


Рис. 3. Главное окно приложения NetLogo.

Рабочая область NetLogo состоит из пунктов главного меню: файл, редактировать, средства, увеличение, табы и помощь. По умолчанию открывается редактор с новым листингом разработки прикладных программ. Листинг состоит из вкладок «Интерфейс», «Инфо» и «Код».

Вкладка «Интерфейс» предназначен для дизайна интерфейса прикладной программы. Вопросы, связанные с дизайном интерфейса пользовательских программ, будут обсуждаться далее. Для общего понимания кратко рассмотрим предназначение отдельных элементов вкладки «Интерфейс». Вкладка «Интерфейс» условно делится на три части: область горячих кнопок, область дизайна интерфейса прикладной программы с окном визуализации, а также «Командный центр». Используя горячие кнопки можно добавлять новые элементы управления в создаваемое приложение, редактировать существующие или удалять их полностью. Кроме элементов управления в области горячих кнопок присутствует управляющий элемент в форме слайдера (ползунок) для регулирования скорости моделирования Ticks. Этот управляющий элемент позволяет ускорять или замедлять процесс имитационного моделирования. Используя выпадающий список, можно выбирать один из двух режимов работы – тиковый или непрерывный. Кнопка «Установки...» отвечает за вызов дополнительного окна «Model Settings». Оно предназначено для изменения: размеров области визуализации, размеров пятен, которые являются пассивными аналогами агентов, шрифта ярлыка агента, скорости смены кадров и визуализации текущих тиков на форме разрабатываемой программы.

Вкладка «Инфо» предназначена для описания разрабатываемой пользовательской программы. Кнопка «Найти» этой вкладки позволяет найти текст в данном описании, а кнопка «Редактировать» – открывает описание в редакторе типа «Блокнот». При нажатии на кнопку «Редактировать» появляется дополнительная кнопка «Помощь», предназначенная для оказания оперативной помощи по созданию описаний для разрабатываемых программ.

Вкладка «Код» также является многострочным редактором (рис. 4), но в отличие от вкладки «Инфо», он тесно связан с элементами управления вкладки «Интерфейс». На панели горячих кнопок присутствует функция оперативного поиска нужного фрагмента программы. Кнопка «Проверка» позволяет разработчику выявить в коде синтаксические ошибки или непривязанные процедуры. Всплывающий перечень «Процедуры» содержит информацию о созданных пользователем процедурах в коде программы.

Принципы разработки программ в системе NetLogo можно описать следующей последовательностью шагов:

1. на форме вкладки «Интерфейс» расположить элементы управления пользовательской программой, для каждого из которых также необходимо указать привязанную к ней процедуру;

2. во вкладке «Код» описать, привязываемые к элементам управления на форме, пользовательские процедуры;
3. на вкладке «Интерфейс» нажать кнопку запуска программы и получить требуемый выходной результат в окне визуализации.

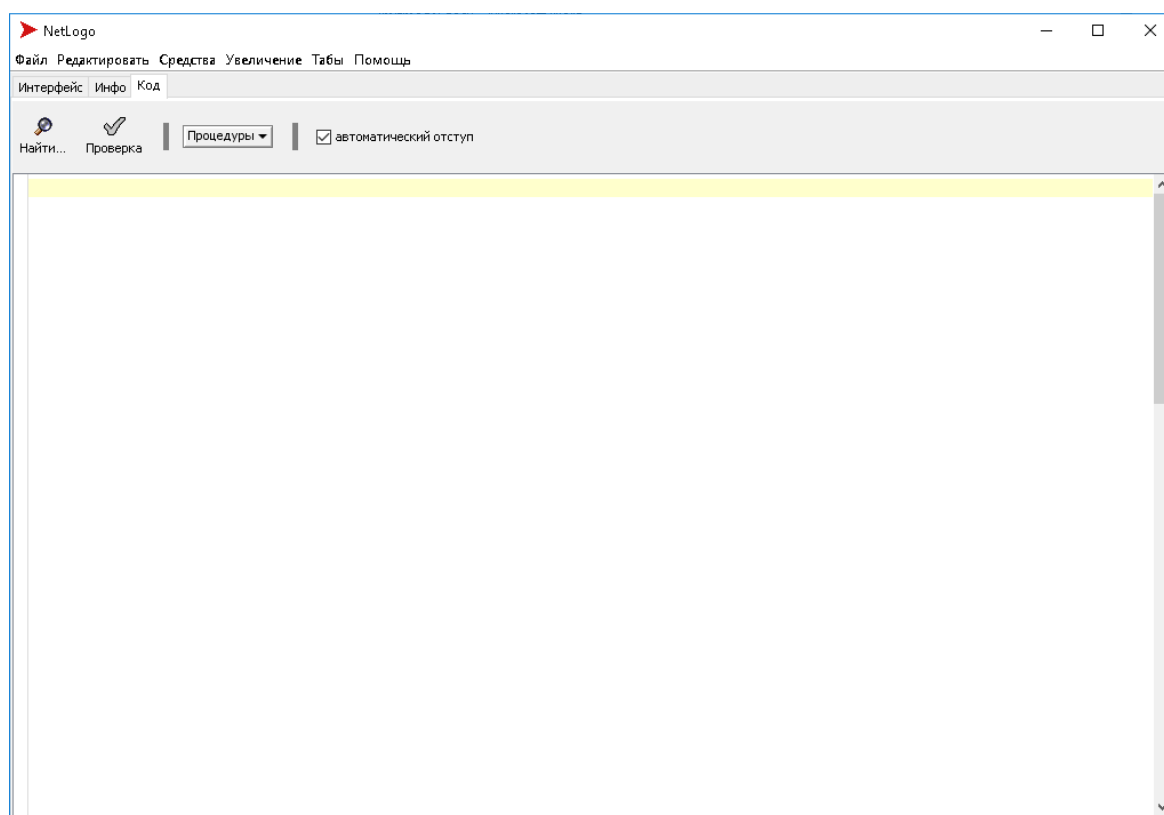


Рис. 4. Вкладка «Код системы» NetLogo.

В следующем параграфе рассмотрим процесс создания нашей первой программы согласно перечисленным выше трем шагам. Забегая вперед, хотелось бы отметить, что в стандартной программе NetLogo обычно присутствуют две кнопки. Первая из них необходимо для генерации агентов в области визуализации. Вторая кнопка содержит код по запуску и непрерывному имитационному моделированию процесса [4-6].

1.3. Разработка первой программы в системе NetLogo

Любая программа в системе NetLogo состоит из двух частей – формы с управляющими элементами и кодов обработчиков событий этих управляющих элементов.

Сделаем постановку задачи следующим образом: разработать имитационную модель в виде многоагентной системы, состоящей из 5 агентов, каждый из которых двигается в случайном направлении (на целое значение угла от 0 до 360 градусов) на случайное число шагов (на целое значение шага от 0 до 5 единиц).

Для решения данной задачи понадобятся две кнопки. Первая кнопка будет использована в качестве элемента управления для запуска процедуры инициализации. В окне редактирования параметров кнопки нужно указать название процедуры обработчика нажатия на кнопку (процедура `setup`), наименование кнопки управления (слово «Инициализация») и необязательный параметр «Клавиша действия» (кнопку можно будет активизировать не только нажатием кнопки мыши, но нажатием на кнопку клавиатуры S) (рис. 5).

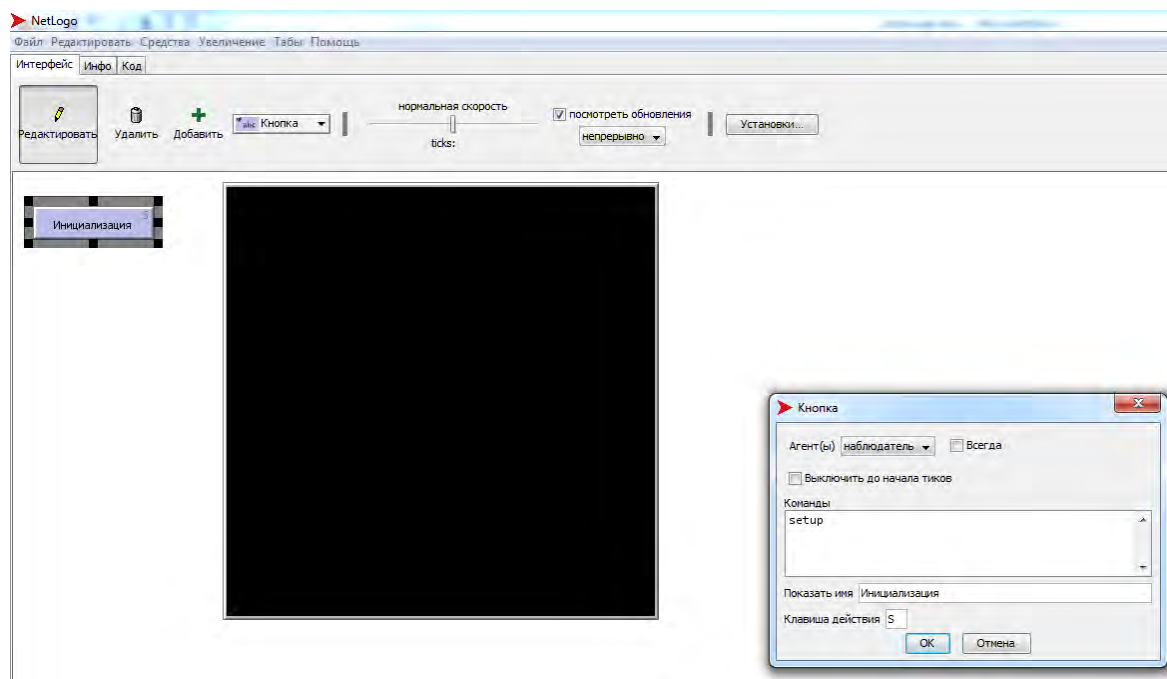


Рис. 5. Создание управляющего элемента «Кнопка» для начальной инициализации агентов.

Есть еще три важных параметра: обработчик создается для агента-наблюдателя (это пользователь), флажок «Всегда» не установлен, в противном случае обработчик будет работать бесконечно.

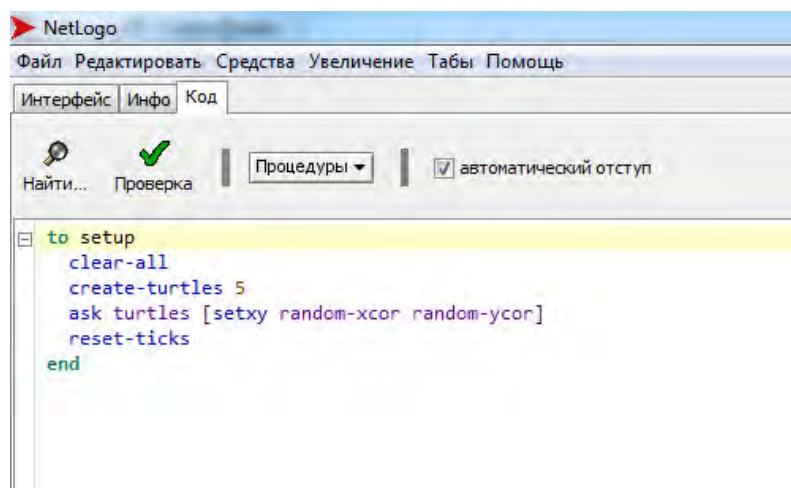


Рис. 6. Код обработчика кнопки «Инициализация».

Флажок «Выключить до начала тиков» предназначен для деактивации создаваемого элемента управления в начале запуска программы. Кнопка активизируется только после запуска тиков программным путем. Код обработчика должен выглядеть следующим образом (рис. 6). Детальный разбор кода обработчика кнопки «Инициализация» представлен в таблице 1 [4-6].

Таблица 1
Описание строк кода обработчика кнопки «Инициализация»

Команда	Описание
to setup	Заголовок процедуры обработчика с его названием
clear-all	Сброс всех параметров объектов имитационного мира
create-turtles 5	Создание 5 агентов типа turtles
ask turtles []	Отправка набора команд, расположенных внутри скобок [], на выполнение всем агентам типа turtles
setxy random-хcor random-ycor	Установка случайных координат у агентов, которым отправляется данная команда
reset-ticks	Сброс счетчика дискретных шагов на начальное значение
end	Конец обработчика элемента управления

Аналогично создается кнопка go, которая предназначена для запуска имитационной модели многоагентной системы и бесконечного выполнения кода (рис. 7).

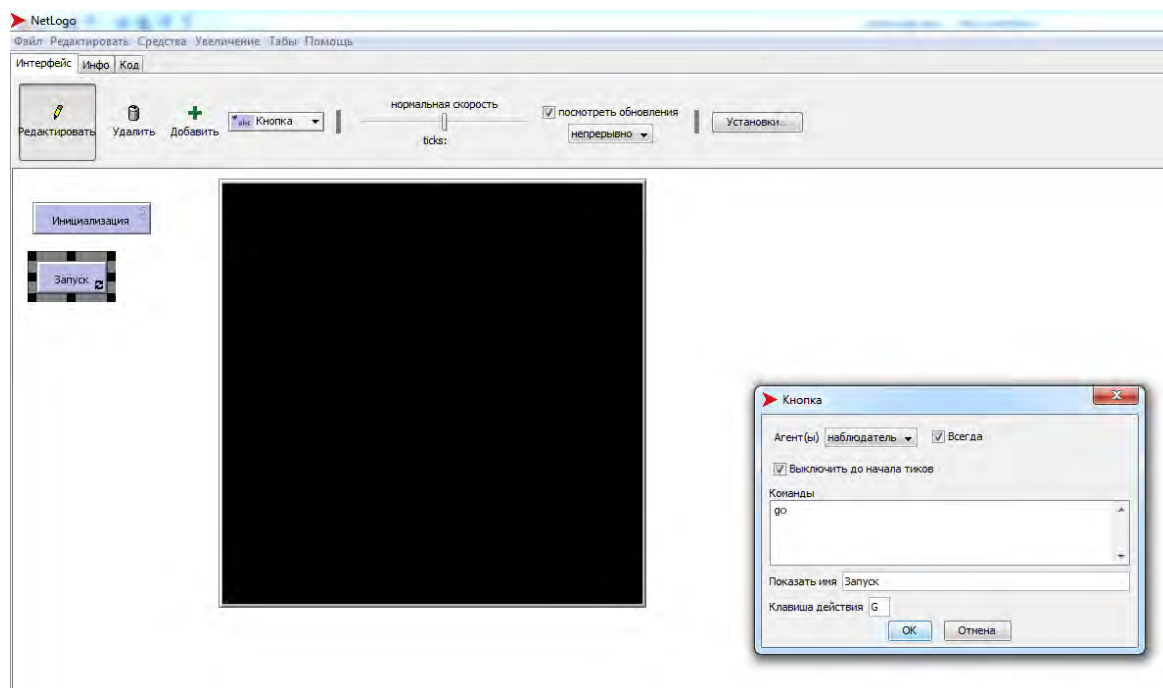


Рис. 7. Создание управляющего элемента «Запуск» для бесконечного выполнения команд

Код обработчика элемента управления «Запуск» приведен на рисунке 8. Таблица 2 содержит разъяснения кода данного обработчика [4-6].

```

to go
  move-agents
  tick
end

to move-agents
  ask turtles [
    right random 360
    forward random 5
  ]
end

```

Рис. 8. Код обработчика кнопки «Запуск» и дополнительной пользовательской процедуры движения агентов

Таблица 2
Описание строк кода обработчика кнопки «Запуск»

Команда	Описание
to go	Заголовок процедуры обработчика с его названием
move-agents	Вызов пользовательской процедуры move-agents
tick	Запуск процесса дискретного моделирования
ask turtles []	Отправка набора команд, расположенных внутри скобок [], на выполнение всем агентам типа turtles
right random 360	Поворот вправо на случайный целый угол $0 \leq a < 360$
forward random 5	Движение вперед на случайную единицу $0 \leq p < 5$
end	Конец обработчика элемента управления

Нажатие на кнопку «Инициализация» приведет к созданию 5 случайных агентов на поле визуализации, а нажатие на кнопку «Запуск» - к их движению в случайных направлениях на случайную длину шага (рис. 9).

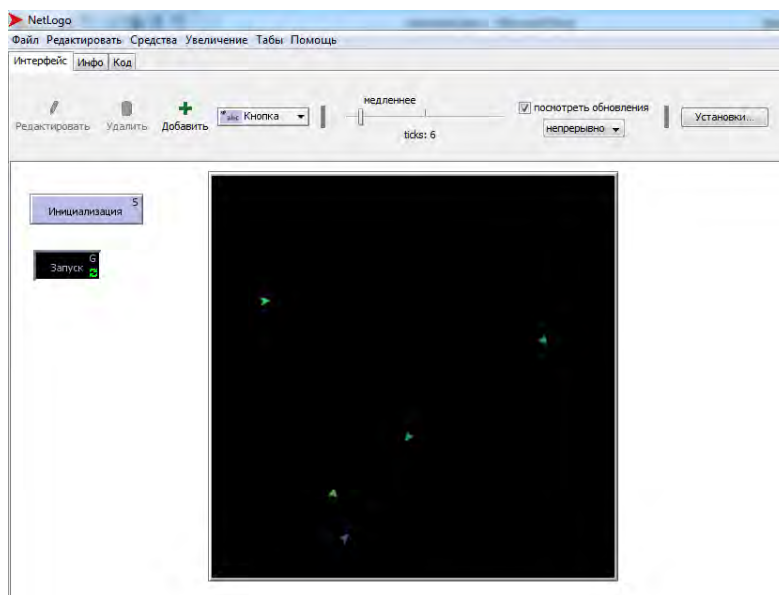


Рис. 9. Работа имитационной модели многоагентной системы.

1.4. Агенты системы программирования NetLogo

Система программирования NetLogo позволяет разрабатывать многоагентные системы и проводить их имитационное моделирование [4-6]. Основными элементами агентной модели NetLogo являются:

- агенты patches – пятна;
- агенты turtles – черепахи;
- элемент observer – наблюдатель;
- элементы links – связи.

Пятна являются статичными агентами и имеют прямоугольную форму. В ходе имитационного моделирования они свое местоположение не меняют и используются для построения области визуализации (виртуальный мир). Черепахи являются, в отличие от пятен, активными агентами и могут перемещаться в виртуальном мире. Элемент наблюдатель существует в единственном экземпляре и фиксирует любые изменения в виртуальном мире и может вмешиваться в ход имитационного процесса. Элементы связи используются для установления связи между черепахами.

Виртуальный мир агентов является двумерным и состоит из набора пятен (рис. 9). Область визуализации является интерактивным элементом или может быть настроена через окно «Model Settings» [4-6] путем вызова ее через контекстное меню (рис. 10).

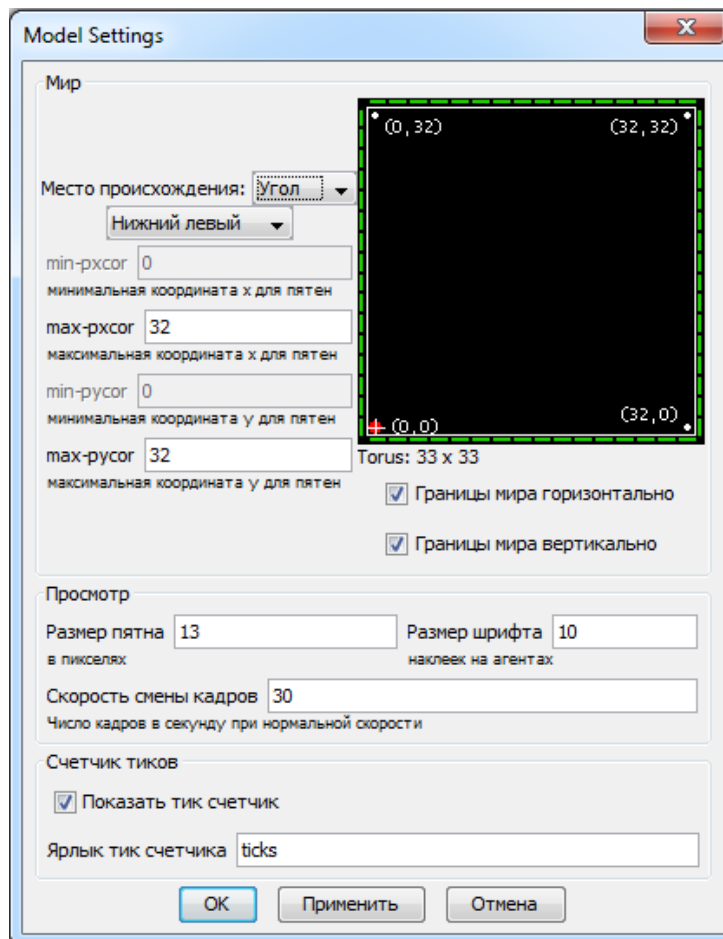


Рис. 10. Окно настройки области визуализации «Model Settings».

Начальный элемент виртуального мира имеет координаты (0,0) по осям X и Y. Начало отсчета выбирается из выпадающего списка «Место происхождения». Если выбрана точка отсчета «Угол», то возможен один из 4 вариантов выбора. Аналогично для точки отсчета «Ребро» - возможно 4 варианта отсчета. Точка отсчета «Центр» имеет только один вариант расположения, а пользовательская точка «Custom» указывается пользователем самостоятельно. Параметры min-pxcor, max-pxcor, min-pycor и max-pycor используются при указании координат двух граничных точек. Эти же координаты можно изменить путем интерактивного перетаскивания мышкой границы виртуального мира. Флажки «Границы мира горизонтально» и «Границы мира вертикально» позволяют настраивать форму взаимодействия черепах с границами виртуального мира. Возможны 4 варианта конфигураций (таблица 3) [4-6].

Таблица 3
Конфигурации границ виртуального мира

№	Горизонтальная граница	Вертикальная граница	Тип виртуального мира
1	+	+	Тип «Torus». Агент при движении пересекает любую из границ и

			появляется с противоположной стороны
2	-	+	Тип «Cylinder H»: Агент может пересекать верхнюю и нижнюю границы.
3	+	-	Тип «Cylinder V»: Агент может левую и правую границы.
4	-	-	Тип «Box»: Агенты движутся в замкнутом мире

Размеры пятен также являются изменяемыми. При установленных параметрах min-pxcor, max-pxcor, min-pxcor и max-pxcor изменение размера пятна также приводит к визуальному изменению виртуального мира.

Центральными элементами многоагентной системы являются черепахи и пятна. С точки зрения программиста, черепаха является программно управляемым объектом. Любой программируемый объект имеет свойства, которые используются в ходе решения задач. Свойства агентов черепах приведены на таблице 4 [4-6].

Таблица 4
Свойства агентов-черепах

№	Свойство	Описание
1	who	Номер агента (начальное значение - 0)
2	color	Цвет агента
3	heading	Угол поворота агента (от 0 до 360 градусов)
4	xcor, ycor	Координаты положения агента по осям
5	shape	Номер фигуры агента (форма агента)
6	label	Текст – метки агента
7	label-color	Цвет текста метки
8	hidden?	Видимость агента
9	size	Размер агента в относительных единицах

Агенты-пятна тоже имеют свойства. В отличие от черепах, программно доступных свойств у них меньше. Свойства пятен позволяют работать с их координатами, цветом и меткой (таблица 5) [4-6].

Таблица 5
Свойства агентов-пятен

№	Свойство	Описание
1	pxcor, pycor	Координаты положения пятна в пикселях по осям координат
2	pcolor	Цвет пятна
3	plabel	Текстовая метка пятна
4	plabel-color	Цвет текстовой метки

Черепашки и пятна могут изменять цвета с использованием базовой или расширенной палитры цветов. Базовая палитра задается цветовыми константами или числовой константой, кратной 5 (рис. 11) [4-6].

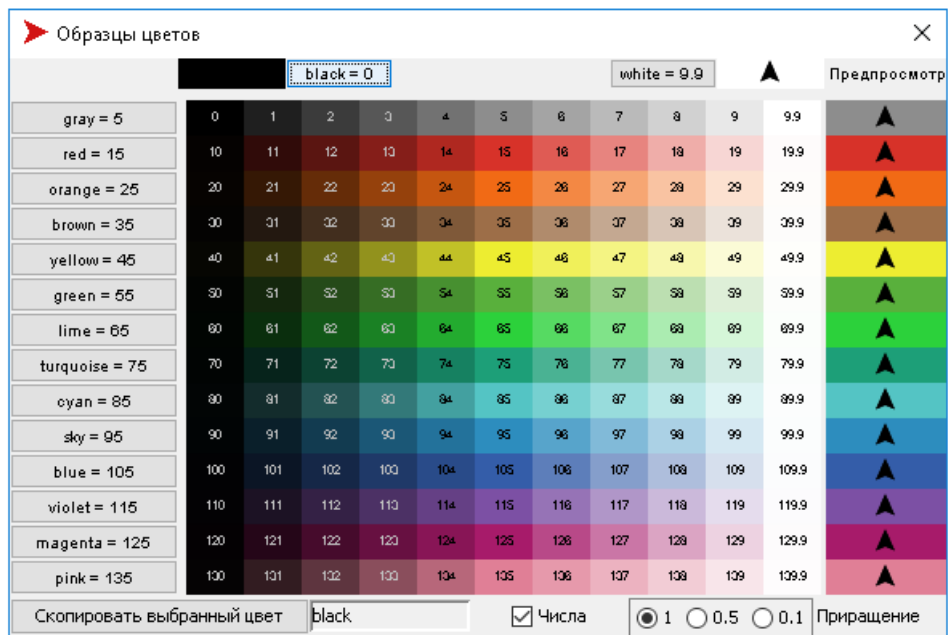


Рис. 11. Базовая палитра цветов.

Каждый агент-черепашка имеет свое графическое представление в виртуальном мире, называемое пиктограммой. Тип пиктограммы черепахи записывается в свойство `shape` в виде строчной константы. По умолчанию используется константа `default`, имеющий вид стрелки (рис. 12) [4-6].

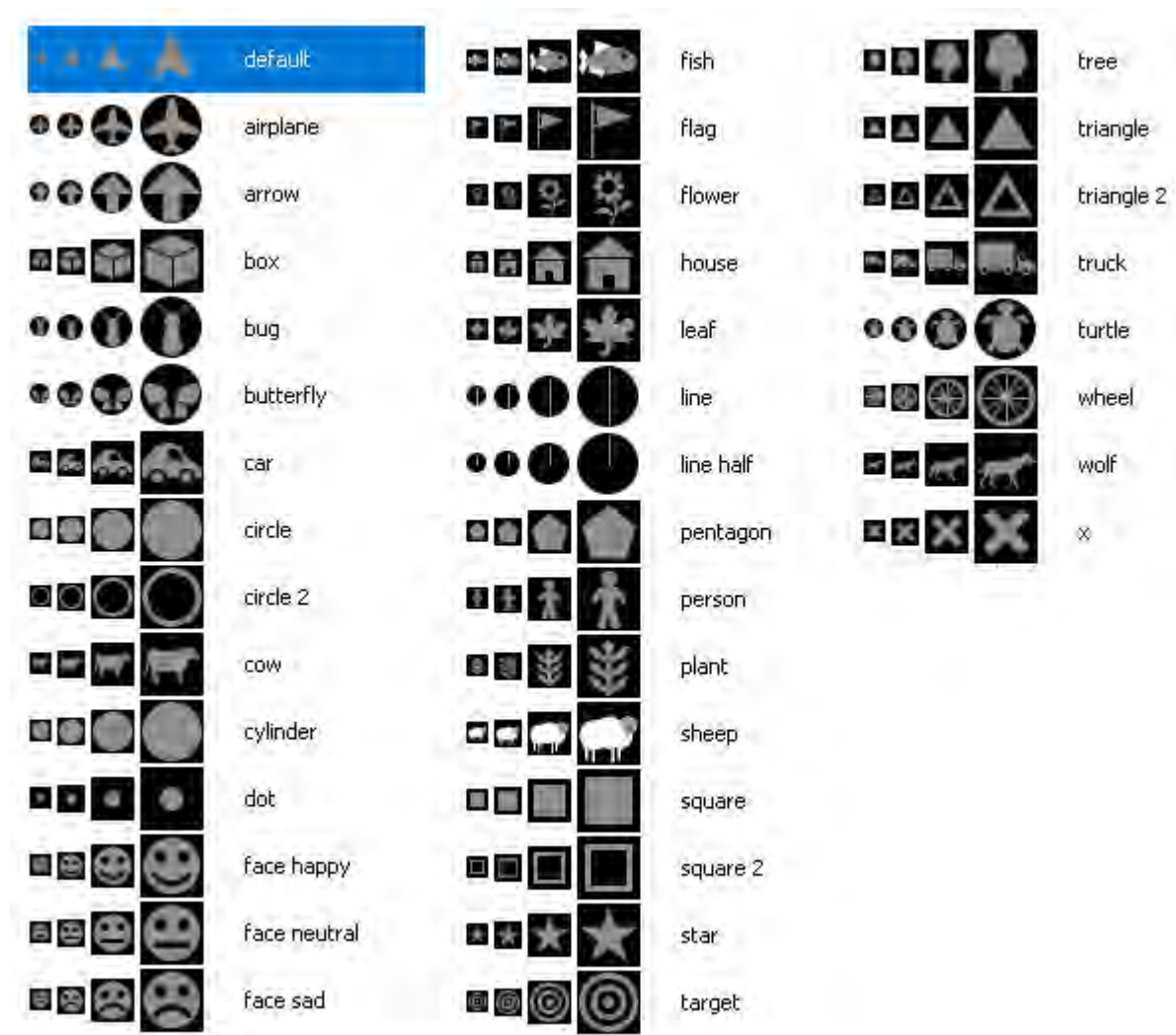


Рис. 12. Пиктограммы агентов-черепах.

Агенты индексируются с 0. Направление вращения агентов с применением свойства `heading` осуществляется согласно левой системе координат, направление вращения которых приведено на рисунке 13 [4-6]. Острие агента указывает на его направление движения.

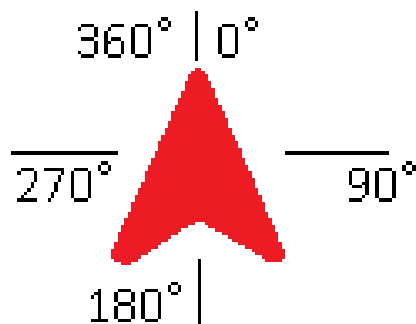


Рис. 13. Направление вращения агентов.

Глава 2. Разработка программного тренажера на языке NetLogo

2.1 Программный тренажер по оптимальному уничтожению системы противовоздушной обороны противника и наземной инфраструктуры

Программные тренажеры используются в различных областях и сферах человеческой жизнедеятельности, от образования и медицины до производства и промышленности. Создание программных тренажеров является достаточно сложным технологическим процессом. Это связано, в первую очередь, сложностью реалистичного представления окружающей среды в программном комплексе или сложностью алгоритма работы, что зависит от высокой детализированности процедуры проверки компетенций. Диагностика компетенций или их формирование является основной задачей программных тренажеров. В большинстве случаев современных задач недостаточно создание тренажера в виде отдельного программного приложения, а требуется разработка целого программно-аппаратного комплекса [7]. Учебное пособие не ставит целью создание таких тренажеров. В то же время во 2 главе пособия будет полностью раскрыт технологический цикл по созданию программного тренажера в военном деле.

Военная дело одна из тех немногочисленных сфер, где программные и программно-аппаратные тренажеры интенсивно используются, а их разработка остается актуальной задачей. Рассмотрим одну из задач в этой области в упрощенной форме. Допустим комплекс противовоздушной обороны (ПВО) из нескольких зенитных орудий обороняет ограниченную территорию, на которой расположены здания оборонного значения (жилые помещения, производственные помещения или комплексы, ремонтные мастерские, ангары военной техники и т.п.) [8-10]. Любой комплекс ПВО можно условно представить в виде двух взаимодействующих частей – из подсистемы обнаружения и слежения за целями (командный пункт), и комплекса зенитных орудий для уничтожения воздушных целей, управляемого командным пунктом [10]. Любое зенитное вооружение имеет эффективный радиус поражения. В нашем случае, орудия будут отслеживать перемещение воздушной цели и автоматически открывать огонь на поражение при пересечении ею границ территории обороны. Пусть охраняемые здания территориально удалены друг от друга на достаточно большие расстояния. В реальном случае, эта ситуация очень часто встречается, что связано с необходимостью иметь большие прилегающие территории рядом с охраняемыми зданиями (аэродромы, парковки, зоны выгрузки военной техники и т.п.) [8-10]. Таким образом, является реальностью случай, когда огневые мощности системы ПВО недостаточно охватывают обороняемую территорию. Поэтому комплекс ПВО дополним воздушными дирижаблями, которые на борту имеют контактные боевые части. Под термином контактная боевая часть подразумевается бомба с

контактным взрывателем. Если воздушная цель войдет в непосредственный контакт, то она будет уничтожена из-за столкновения с ней. Если же уровень вибрации дирижабля будет выше некоторого порога, то сработает взрыватель и цель будет уничтожена воздушной волной осколков. Эти два случая в программном тренажере объединены в один случай, когда дирижабль входит в контакт с авиационной системой. Кроме того, будут введены дополнительные функции расположения объектов противника: зенитное вооружение располагается случайно на территории обороны, охраняемые здания располагаются случайно на территории, воздушные дирижабли располагаются случайно на обороны и периодически исчезают из поля зрения пользователя. Исчезновение дирижаблей связано с дополнительным усложнением тренировок. Это позволяет привлекать и тренировать визуальную память пользователя, в силу необходимости иметь полную информация о расположении дирижаблей для исключения столкновений в ходе игры.

Объектом управления в игровом тренажере является ударная авиационная система (АС) (может быть как пилотируемым самолетом, так и беспилотным летательным аппаратом, т.е. БПЛА). АС на борту имеет всегда ограниченное количество бортового вооружения и топливо для решения боевой задачи. Вооружение АС будет состоять из ракет двух классов – кинетических без боевой части с взрывателем и стандартных ракет с боевой частью. Кинетические ракеты позволяют уничтожать цели при непосредственном контакте с целью путем передачи своего импульса ей, т.е. уничтожение цели идет из-за прямого контакта с ракетой. Если же цель не является подземным бункером или является постройкой плохого качества, то кинетические ракеты могут пролетать сквозь этих объектов, и лететь дальше для уничтожения следующих целей, расположенных на ее траектории. Таким образом, кинетические ракеты обладают большим потенциалом по контактному уничтожению нескольких целей на траектории ее движения. Стандартные же ракеты уничтожают цели путем срабатывания взрывателя бомбы при непосредственном контакте или по таймеру. На территории воздушной атаки будут периодически появляться бонусы в виде пиктограмм «сердце», которые являются временными точками дозаправки АС.

Боевая задача в разрабатываемом тренажере будет заключаться в уничтожении зенитного вооружения и охраняемых территорий с применением доступного бортового вооружения за ограниченное время и суммарную дальность полета. Дальность полета может быть увеличена за счет бонусов.

Программный тренажер будет вести учет игрового времени (время затраченное на решение боевой задачи), длины траектории полета, количества использованного бортового вооружения (отдельно по каждому типу ракет), количества уничтоженных зенитных орудий противника, количества уничтоженных зданий обороны, числа использованных бонусов, причину завершения работы тренажера (игра завершена по причине решения

боевой задачи или уничтожения АС противником). Целью тренажера является оценка качества решения боевой задачи пользователем на интуитивном уровне. В формальном виде эту задачу можно сформулировать, как задачу оптимизации функции (1).

$$F(X(t), Y(t)) = \frac{|N_X - P(X(t))| + |N_Y - P(Y(t))|}{N_X + N_Y} \rightarrow \max, \quad (1)$$

$$X(0) = N_X, \quad Y(0) = N_Y$$

где $X(t)$ и $Y(t)$ - множества зенитных орудий противника и обороняемых зданий соответственно в момент времени t ; N_X, N_Y - первоначальное количество зенитных орудий противника и обороняемых зданий соответственно; $P(X(t))$ и $P(Y(t))$ - мощности множеств $X(t)$ и $Y(t)$ в момент времени t в смысле количества их элементов.

2.2 Дизайн пользовательского интерфейса программного тренажера

На рисунке 14 изображена главная форма тренажера. Полный листинг программного тренажера приведен в Приложении данного учебного пособия. Управляющими кнопками тренажера являются: кнопки запуска тренажера, джойстик (из четырех основных кнопок), кнопки управления огнем (дополнительные кнопки) и слайдеры (ползунки) для изменения скорости угла поворота и поступательного приращения модели ударного БПЛА.

Кнопки «Start» и «Time» предназначены для запуска тренажера и применяются в указанном порядке. Первым действием нажимается кнопка «Start», что приводит к случайной генерации зенитных установок комплекса ПВО (4 зенитных орудия), объектов обороны (5 зданий), военных дирижаблей (10 единиц) и одного ударного БПЛА в пределах виртуального мира. Движение и направление БПЛА осуществляются джойстиком. Слайдеры «Angle» и «Speed» позволяют изменять приращения поворота и перемещения.

На форме тренажера имеются информационные панели. Можно выделить 2 группы информационных панелей – панели бортового вооружения и игровых параметров. В состав бортового вооружения БПЛА входят 3 кинетические ракеты без боевой части и 5 стандартных ракет. Отличие кинетических ракет от стандартных состоит в возможности поражения одной ракетой несколько подряд расположенных неподалеку друг от друга объектов противника. Комплекс ПВО имеет ограниченный радиус поражения БПЛА, но неограниченный радиус автоматического сопровождения цели в пределах виртуального мира. Если БПЛА попадает в зону поражения, ближайшее зенитное орудие открывает огонь на поражение. Так как территория противника не полностью охватывается комплексом ПВО, используются военные дирижабли в количестве 10 единиц. В течение игры дирижабли периодически исчезают из поля зрения игрока. Данный

процесс моделирует ограниченность бортовой аппаратуры пеленгации целей у БПЛА.

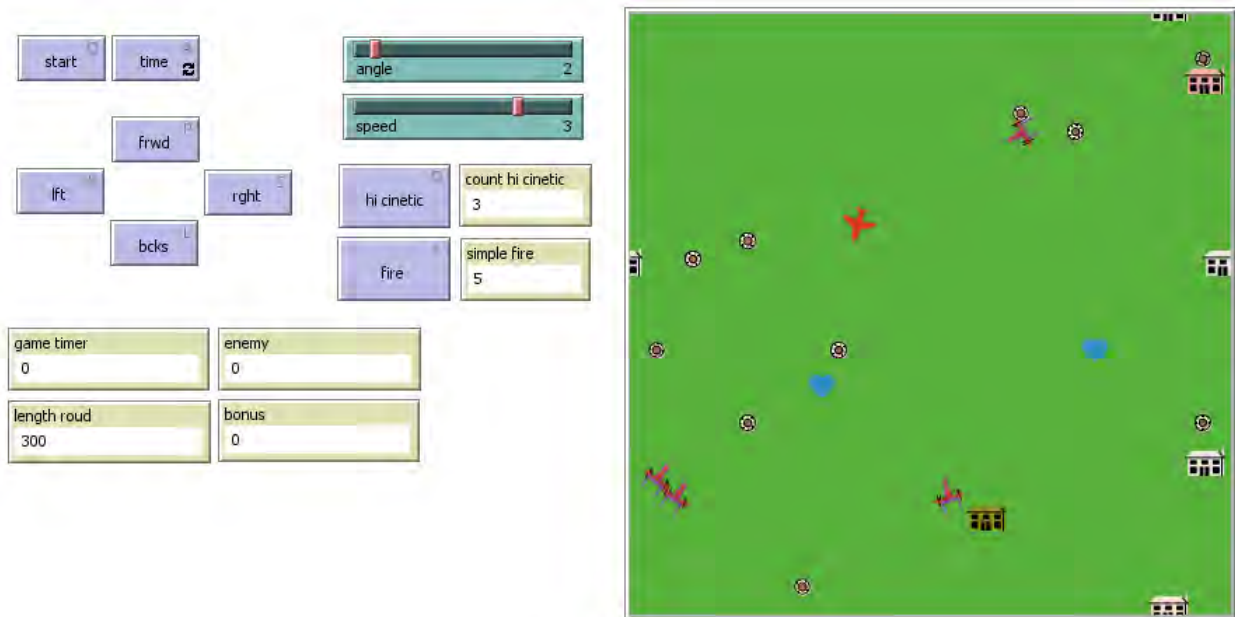


Рис. 14. Пользовательский интерфейс программного тренажера

Подобный подход направлен на тренировку визуальной памяти игроком в процессе игры с целью запоминания местоположения дирижаблей. Контакт с дирижаблем приводит к уничтожению БПЛА.

К панелям игрового процесса относятся «Game Timer», «Enemy», «Length Roud» и «Bonus». Боевая задача БПЛА заключается в уничтожении комплекса ПВО и обороняемых зданий за минимальное количество времени в условиях ограниченности вооружения и пролетаемого пути.

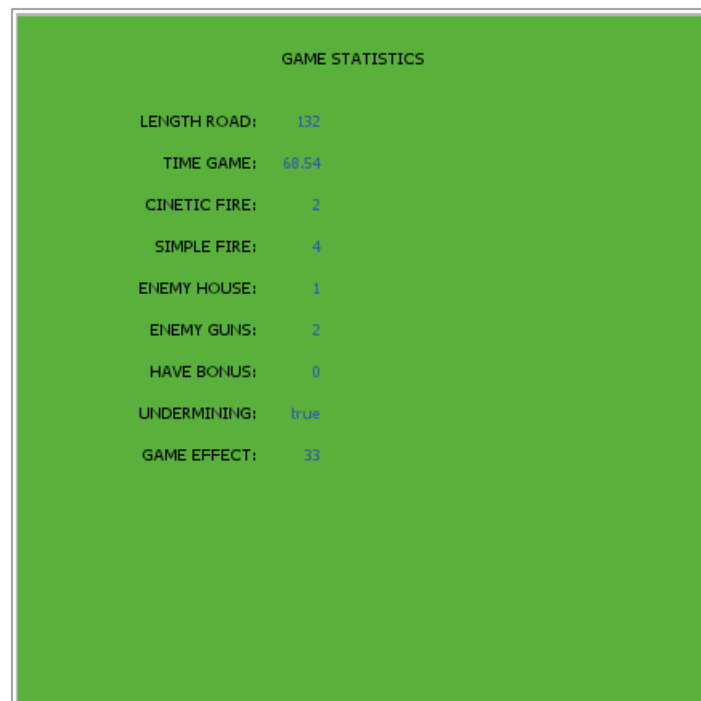


Рис. 15. Комплексная оценка эффективности решения задачи.

Вооружение в процессе игры не подлежит восполнению, а длина пролетаемого пути может быть увеличена за счет ограниченного количества бонусов, которые моделируют дозаправку в воздухе. Результатом игры являются показатели (рис. 15), которые комплексно характеризуют эффективность решения поставленной боевой задачи.

Согласно полученной оценке эффективности после решения боевой задачи остались 2 кинетические ракеты и 4 стандартные ракеты. Задача была решена за 68,54 сек, и АС пролетела суммарно 132 км по территории противника. В ходе решения боевой задачи было уничтожено 1 здание и 2 зенитных орудия. АС не совершала дозаправку в воздухе. Тренировка закончилась по причине столкновения с военным дирижаблем. Эффективность решения боевой задачи составило 33%.

2.3 Описание используемых переменных и вывод скрина игры

Язык программирования NetLogo допускает разделение множества агентов на классы согласно обстановке в предметной области [4-6]. Для создания класса агентов применяется команда `breed` в формате

`breed [имя_класса комментарий]`

В решаемой задаче выделим следующие классы агентов:

- 1 агент-БПЛА `bpla` является персонажем игрока
- 2 агент-снаряд `balls` выстреливается агентом-БПЛА
- 3 агент-противник `prot` уничтожается агентом-БПЛА
- 4 агент-бонус `lifes` продлевает выделенный путь
- 5 информационный агент `infos` используется для информирования
- 6 агент-мина `minas` является военным дирижаблем
- 7 агент-зенитка `guns` является зенитным орудием противника
- 8 агент бомба `bombs` выстреливается зенитным орудием противника

Все классы определяются во вкладке `Code` в указанном порядке из списка агентов.

`breed [bpla robot]`

`breed [balls ball]`

`breed [prot prot]`

`breed [lifes life]`

`breed [infos info]`

`breed [minas mina]`

`breed [guns gun]`

`breed [bombs bomb]`

Создаем параметры времени `vrlife` и `lftime` для классов `lifes` и `minas` соответственно, с использованием команды `own`.

`lifes-own [vrlife]`

minas-own [lftime]

Создаем массив (вектор) именованных переменных с глобальным доступом. Это означает, что перечисленные переменные будут доступны внутри обработчиков событий.

```
globals [tx ty cnt nmb tm gsm nmres nmval enemy bonus bns mns boom? gns  
gn bmflag? bmxcor bmycor]
```

Поясним предназначение некоторых переменных из вектора-параметра:

tx, ty - координаты танка

cnt, nmb - счетчики снарядов танка

tm - время игры

nmres, nmval - массив наименований статистики

enemy, bonus - количество врагов и бонусов

Все переменные и классы должны быть описаны вне обработчиков событий управляющих элементов. Создаем обработчик кнопки «Start». Некоторые фрагменты кода этого обработчика потребуют объяснения. Следующий фрагмент кода выводит название игры 96 цветом, где в массиве **nx** содержатся X координаты пятен, а в массиве **ny** – Y координаты.

```
let nx [6 6 6 6 6 6 7 7 7 8 8 8 9 9 9 9 9 11 11 11 11 11 11 12 13 14 15 15 15 15 15  
15 17 18 18 18 18 19 20 21 21 21 21 21 21 23 24 24 24 24 25 26 27 27 27 27 27  
27 24 25 26]  
let ny [15 16 17 18 19 20 15 18 20 15 18 20 15 16 17 18 20 15 16 17 18 19 20 20  
20 20 15 16 17 18 19 20 15 16 17 18 19 20 20 15 16 17 18 19 20 15 16 17 18 19  
20 20 15 16 17 18 19 20 17 17 17]  
ask patches [set pcolor 96]  
let k 0  
repeat length nx [  
  ask patch ((item k nx) - 3) item k ny [set pcolor 63]  
  wait 0.01  
  set k k + 1  
]
```

Следующий фрагмент кода выводит дополнительные элементы на экране – пиктограмму самолета и прицел. На рисунке 15 изображен результат работы выше описанных фрагментов кода, которые создают скрин лист игры.

```
create-bpla 1 [set shape "airplane" set size 8 set color 67 setxy 14 12 set  
heading 35 ]
```



```
create-trgt 1 [set shape "target" set size 6 set color red setxy 25 22 set heading 35]
```

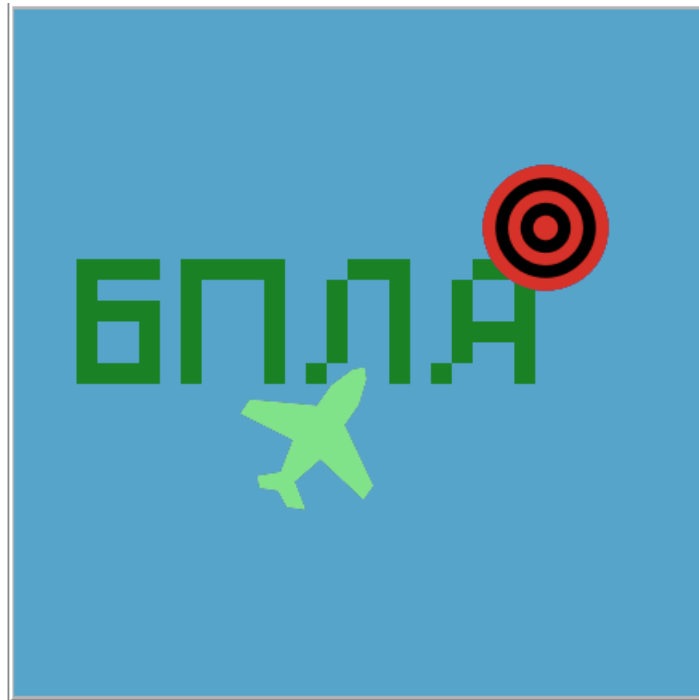


Рис. 15. Скрин лист игры.

Спустя 2 секунды происходит удаление скрина – уничтожаются все агенты класса `bpla` и пятна `patches`. Фрагмент кода, отвечающий за удаление, приведен ниже:

```
ask bpla [die]
ask patches [set pcolor green]
```

2.4 Начальная инициализация игрового уровня в тренажере

Генерация уровня игры происходит внутри обработчика `Start` сразу же после уничтожения скрин-листа. По умолчанию персонаж получает форму самолета. Создается 1 экземпляр агента класса `bpla` размера 2 и в пределах виртуального мира. Персонажу назначается красный цвет. Его координаты сохраняются в переменных `tx` и `ty`. Ниже приведен фрагмент, совершающий все перечисленные действия.

```
set-default-shape bpla "airplane"
create-bpla 1 [
  set size 2
  set shape "airplane"
  setxy random max-pxcor random max-pycor
  set color red
  set tx [xcor] of bpla
```

```
set ty [ycor] of bpla
```

```
]
```

Далее с использованием команды `create-protos` и `create-guns` создаются здания и зенитные орудия противника в количестве `enemy=5` и `gns=4` единиц. Положение объектов противника генерируется в пределах `[0; world-width)` для `x` координаты и в пределах `[0; world-height)` для `y` координаты агентов в виртуальном мире.

Особенностью фрагмента генерации зданий противника является размер пиктограммы, форма пиктограммы и случайно выбираемый цвет здания.

```
create-protos enemy [
```

```
  setxy random world-width random world-height
```

```
  set size 2.5
```

```
  set shape "house colonial"
```

```
  set color 15 + random 35
```

```
]
```

В отличие от зданий зенитные орудия имеют меньший размер, имеют только красный цвет. Кроме всего сказанного, при создании зенитного орудия каждый экземпляр устанавливает флаг `bmflag=true` и в начальный момент времени повернут в сторону агента класса `bpla`.

```
create-guns gns [
```

```
  setxy random world-width random world-height
```

```
  set size 2
```

```
  set color red
```

```
  set shape "gun"
```

```
  set bmflag? false
```

```
  face aircraft 2
```

```
]
```

Аналогичным фрагментом кода создается множество военных дирижаблей. Особенностью этой группы агентов является наличие постоянного цвета `brown` и параметра времени их видимости `lftime` в пределах `[2; 12)` секунд.

```
create-minas mns [
```

```
  setxy random world-width random world-height
```

```
  set shape "mina"
```

```
  set color brown
```

```
  set lftime (2 + random 10)
```

```
]
```

Еще одним элементом виртуального мира являются бонусы, которые продлевают расстояние, пролетаемое БПЛА. Особенностью агентов класса bonus является их размер и случайное время видимости в пределах [0; 39] секунд.

```
create-lifes bonus [  
  setxy random world-width random world-height  
  set vrlife random 40  
  set size 2  
  set shape "heart"  
]
```

Все перечисленные фрагменты коды содержатся в теле обработчика событий кнопки «Start». После нажатия на кнопку система, выполнив описанные действия, генерирует очередной уровень игры (рис. 16).

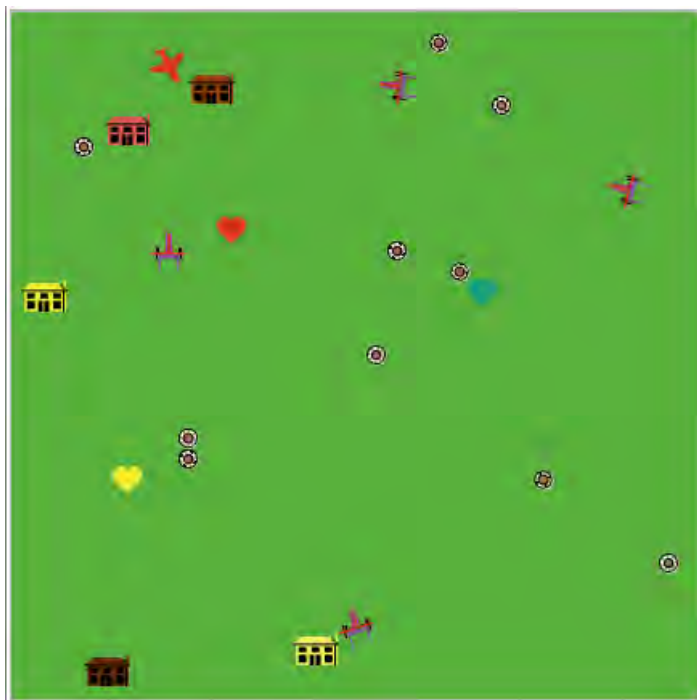


Рис. 16. Начальное состояние виртуального мира тренажера

2.5 Управление полетом БПЛА

Управление игровой моделью БПЛА осуществляется через клавиатуру или мышкой. На пользовательской панели расположены 4 кнопки, образующие джойстик (рис. 17). С каждой кнопкой связан один обработчик, название которого написано на кнопке. Кроме того, к каждой кнопке имеет привязанную «горячую клавишу». Модель БПЛА управляется, тем самым, как мышкой, так и через клавиатуру при нажатии на «горячие клавиши». На

изображении приведены кнопки джойстика (рис. 17, слева) и слайдеры (рис. 17, справа) параметров скорости и перемещения.

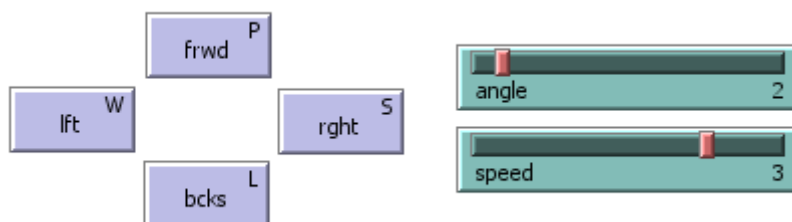


Рис. 17. Кнопки джойстика для поворотов и перемещений

Движение влево или вправо осуществляется в ходе вызова обработчиков `lft` и `rght` соответственно.

```
to lft
  ask bpla [lt angle]
end
```

```
to rght
  ask bpla [rt angle]
end
```

Движение прямо или назад осуществляется вызовами обработчиков `frwd` и `bcks`. Обработчик `frwd` совершает движение БПЛА вперед на 10% от значения параметра `speed` и уменьшает на 1 параметр `gsm`. Дополнительно сохраняются координаты БПЛА в глобальных переменных `tx` и `ty`.

Внутри обработчика `frwd` содержатся блоки, которые отвечают за контакт с бонусом и военным дирижаблем.

```
to frwd
  ask bpla [fd 0.1 * speed set gsm gsm - speed]
  set tx [xcor] of bpla
  set ty [ycor] of bpla
```

Контакт с бонусом возникает на расстоянии менее 2 единиц. При этом увеличивается на 50 единиц параметр `gsm`, который отвечает за продолжительность полета. Захват бонуса приводит к уменьшению их общей численности и увеличению числа приобретенных бонусов.

```
ask lifes [
  if (distancexy item 0 tx item 0 ty) < 2 [
    set gsm gsm + 50
    set bns bns - 1
```

```
die  
]  
]
```

В случае контакта с дирижаблем на расстоянии менее 1 единицы возникает пиктограмма взрыва оранжевого цвета. Устанавливается флаг boom, который фиксирует уничтожение персонажа для результирующей статистики. БПЛА уничтожается как агент.

```
ask minas [  
  if (distancexy item 0 tx item 0 ty) < 1 [  
    set boom? true  
    set color orange  
    set shape "fire"  
    wait 0.1  
    set size 3  
    wait 0.2  
    die  
  ]  
]  
end
```

Обработчик кнопки для движения назад аналогичен предыдущему обработчику. Он содержит те же самые действия: совершает движение модели назад, сохраняет ее координаты, обрабатывает контакт с бонусом и обрабатывает контакт с дирижаблем.

2.6 Управление бортовым вооружением БПЛА

Панель управления тренажера содержит блок дополнительных кнопок по управлению бортовым вооружением (рис. 18).

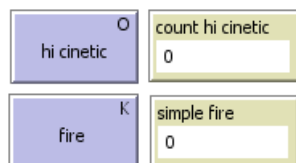


Рис. 18. Кнопки управления бортовым вооружением

Этот блок содержит кнопки для пуска кинетических (кнопка «Hi Cinetic») стандартных (кнопка «Fire») ракет. Рядом с каждой кнопкой располагается информационная панель, на которой отображается численность имеющегося вооружения.

Ниже приведен полный код открытия огня простыми ракетами. Проведем его детальный анализ. Если счетчик простых ракет ненулевой, то создается на позиции БПЛА 1 ракета в направлении противника.

```
to fire
  ifelse (nmb > 0) [
    let fprot count prots
    let tb timer
    create-balls 1 [
      setxy item 0 tx item 0 ty
      set shape "rocket"
      set color orange
      set heading [heading] of aircraft 2
    ]
  ]
```

Пока количество противников не изменилось, ракета летит в указанном направлении. На расстоянии 2 единиц от зданий или пушек противника происходит их уничтожение. В качестве спецэффекта, возникающего при взрыве противника, используется пиктограмма «Огонь» различных цветов.

```
while [count prots = fprot] [
  ask balls [fd 1 wait 0.1]
  ask balls [
    ask prots in-radius 2 [
      set color orange
      set shape "fire"
      wait 0.1
      set size 3
      wait 0.2
      die
    ]
  ]
  ask guns in-radius 2 [
    set color violet
    set shape "fire"
    wait 0.1
    set size 3
    wait 0.2
    die
  ]
]
```

Ракеты обладают временем жизни, при истечении которого происходит их самоуничтожение. В противном случае, их полет продолжался бы бесконечно в виртуальном мире. Выпущенная ракета уничтожается вслед за

противником и одновременно уменьшается количество доступного бортового вооружения. Если же ракет не осталось, то выводится сообщение «Простые ракеты закончились»

```
if (timer - tb) > 1 [ask balls [die] set nmb nmb - 1 stop]
ask balls [die]
set nmb nmb - 1
]
[user-message("Простые снаряды закончены !!!")]
end
```

Использование кинетических ракет отличается от стандартных тем, что кроме самого обработчика sprfire используется дополнительная пользовательская функция skydot1, которая содержит тот же самый код, что и в предыдущем случае. С точки зрения результата применения кинетических ракет, их время жизни увеличено и широта обхвата целей шире, чем у простых ракет. Кинетические ракеты нашли свое широкое применение в локальных конфликтах XXI века. Особенностью данного вида оружия является создание коридора с сильной взрывной волной, которая разрушает и уничтожает за счет мощнейшего импульса. Если стандартная ракета уничтожает первый контактный объект и за счет вторичной волны осколков или огня близлежащие другие объекты, то кинетическая ракета способна уничтожать все объекты в пределах кинетического коридора до потери разрушительного импульса [8-10].

2.7 Главный цикл программного тренажера

Под главным циклом программного тренажера подразумевается обработчик события нажатия на кнопку «Time» (рис. 19). При создании кнопки и подключении обработчика устанавливается флажок «Всегда».

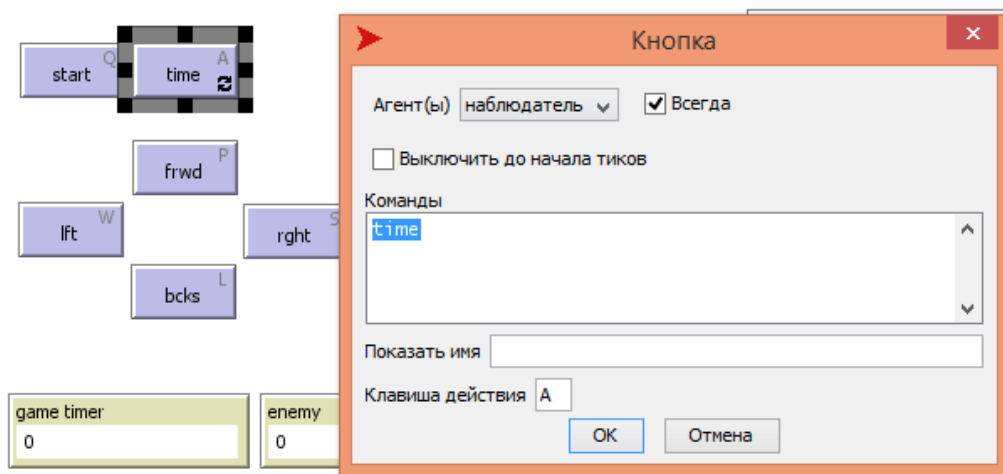


Рис. 19. Настройки обработчика главного цикла тренажера

Это заставляет обработчик функционировать бесконечно, пока не выполнится условие выхода или пользователь не нажмет повторно эту кнопку.

В теле кода обработчика расположено условие программного завершения главного цикла:

- параметр timer достиг значения 90 сек.;
- пользователь исчерпал доступный объем горючего, т.е. пролетел путь больший, чем позволяет топливо на борту;
- уничтожены все противники;
- игра завершается по причине подрыва БПЛА при столкновении с дирижаблем.

Кратко совокупность всех этих условий можно записать следующим образом:

```
ifelse ((timer > 90) or (gsm <= 0) or (not any? prots) or boom?)
```

При выполнении условия ветвления на основе начального количества зданий противника enemy и текущего их остатка count prots, начального количества орудий ЗРК gns и текущего их остатка count guns рассчитывается эффективность решения боевой задачи bal.

```
let encnt (enemy - count prots)  
let guncnt (gns - count guns)  
let bal round(((encnt + guncnt) * 100)/ (gns + enemy))  
set nmval (list gsm tm cnt nmb encnt guncnt bns boom? bal)  
set cnt 0  
set nmb 0
```

Далее формируется комплексная оценка на основе пройденного пути, времени выполнения задания, количества стандартных и кинетических ракет, количества уничтоженных зданий и орудий противника, причины завершения работы тренажера и эффективности выполнения боевой задачи. Данная статистика выводится в качестве отчета после завершения игры. Работа программного тренажера завершается командой stop.

```
ask patches [set pcolor green]  
let z 0  
let nmx 10  
let nmy 28  
create-infos 1 [  
setxy 15 30  
set color green  
set label-color black  
set size 2  
set heading 0  
set label "GAME STATISTICS"
```

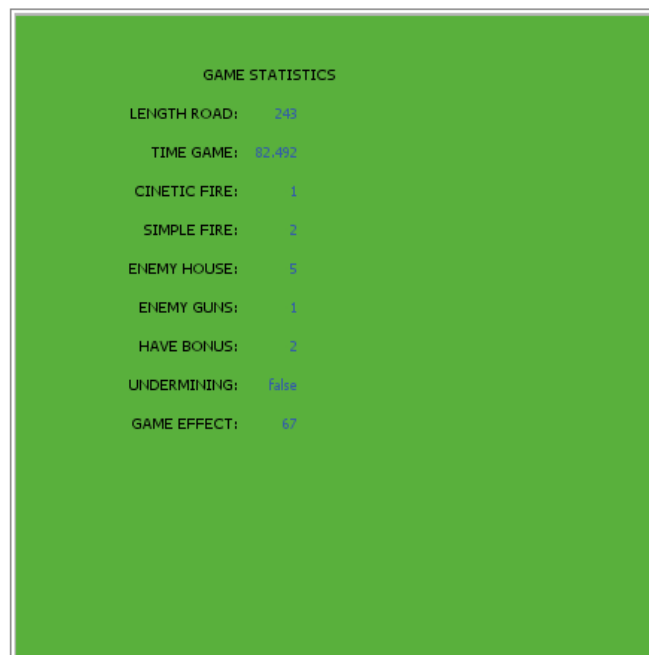


```

]
repeat 9 [
  create-infos 1 [
    setxy nmx nmy
    set color green
    set label-color black
    set size 2
    set heading 0
    set label (item z nmres)
  ]
  create-infos 1 [
    setxy (nmx + 3) nmy
    set color green
    set label-color blue
    set size 2
    set heading 0
    set label (item z nmval)
  ]
  set nmy nmy - 2
  set z z + 1
]
stop
set tm timer

```

Результирующая статистика (рис. 20) позволяет ответить на большое количество вопросов об эффективности решения боевой задачи пользователем.



The image shows a green rectangular area with white text centered on it. The text is organized into a list of statistics. At the top, it says 'GAME STATISTICS'. Below that, there are several lines, each with a label and a value. The values are: LENGTH ROAD: 243, TIME GAME: 82.492, CINETIC FIRE: 1, SIMPLE FIRE: 2, ENEMY HOUSE: 5, ENEMY GUNS: 1, HAVE BONUS: 2, UNDERMINING: false, and GAME EFFECT: 67.

GAME STATISTICS	
LENGTH ROAD:	243
TIME GAME:	82.492
CINETIC FIRE:	1
SIMPLE FIRE:	2
ENEMY HOUSE:	5
ENEMY GUNS:	1
HAVE BONUS:	2
UNDERMINING:	false
GAME EFFECT:	67

Рис. 20. Результирующая статистика после завершения игры

Например, на рисунке 20 приведена статистика, согласно которой пользователь в ходе решения боевой задачи уничтожил больше зданий, чем орудий ЗРК. При этом он использовал 1 кинетическую ракету и 2 стандартных ракет. Средняя огневая мощь равно $(1+5)/(1+2)=6.3=2$, что означает, что каждой ракетой, в среднем, пользователь уничтожал 2 объектов противника. Ни один из дирижаблей не стал причиной завершения тренировок. Средняя эффективность решения боевой задачи равна 67%.

Литература

1. Официальный сайт Центра сетевого обучения и компьютерного моделирования (CCL) Northwestern MIT [Электронный ресурс]: – Режим доступа - <https://ccl.northwestern.edu/>
2. Официальный сайт проекта Scheller Teacher Educational Program [Электронный ресурс]. – Режим доступа: <https://education.mit.edu/starlogo>
3. Официальный сайт Massachusetts Institute of Technology [Электронный ресурс]. – Режим доступа: <https://web.mit.edu>
4. Мезенцев К.Н. Мультиагентное моделирование в среде NetLogo: Учебное пособие. – СПб: Издательство «Лань», 2015. – 176 с.
5. Радченко, И. А. Интеллектуальные мультиагентные системы : учеб. пособие. — СПб. : Балтийский гос. техн. ун-т, 2006. – 88 с.
6. NetLogo. User manual [Электронный ресурс]. – Режим доступа: <http://ccl.northwestern.edu/netlogo>
7. Новиков Ф.А., Опалева Э.А. и др. Управление проектами и разработкой программного ПО. - СПб: СПбГУ ИТМО, 2008. - 256 с.
8. Официальный сайт Центра анализа мировой торговли оружием [Электронный ресурс]: – Режим доступа: <https://armstrade.org/includes/periodics/news/2019/0628/162553142/detail.htm>
9. Официальный сайт сетевого издания «Военное обозрение». [Электронный ресурс]: – Режим доступа: <https://topwar.ru/164570-roj-bespilotnikov-buduschee-boevyh-dejstvij.html>.
10. Чуев Ю.В., Мельников П.М., Петухов С.И., Степанов Г.Ф., Шор Я.Б. Основы исследования операций в военной технике. – М.: Издательство «Советское радио», 1965. – 592 с.

ПРИЛОЖЕНИЕ. Программа тренажера

;ГРУППИРОВКА АГЕНТОВ, ДОБАВЛЕНИЕ НОВОГО ПОЛЯ,
ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ

breed [bpla aircraft] ;тип агентов танк
breed [trgt mishen] ;мишень на скрине игры
breed [balls ball] ;тип агентов снаряд
breed [prots prot] ;тип агентов противник
breed [lifes life] ;тип агентов жизнь
breed [infos info] ;тип агента информация
breed [minas mina] ;тип агента противотанковая мина
breed [guns gun] ;тип агента пушка противника
breed [bombs bomb] ;тип агента бомба пушки

lifes-own [vrlife] ;время жизни бонуса
minas-own [lftime] ;время идентификации позиции мины

globals [tx ty cnt nmb tm gsm nmres nmval enemy bonus bns mns boom? gns gn
bmflag? bmxcor bmycor]

;tx, ty - координаты танка
;cnt, nmb - счетчики снарядов танка
;tm - время игры
;nmres, nmval - массив наименований статистики
;enemy, bonus - количество врагов и бонусов

.*****
,

;НАЧАЛЬНАЯ ПРОЦЕДУРА

to start

clear-all ;очистка фона

;СКРИНШОТ ИГРЫ

let nx [6 6 6 6 6 7 7 7 8 8 8 9 9 9 9 11 11 11 11 11 11 12 13 14 15 15 15 15 15
15 17 18 18 18 18 19 20 21 21 21 21 21 21 23 24 24 24 24 25 26 27 27 27 27 27
27 24 25 26]

let ny [15 16 17 18 19 20 15 18 20 15 18 20 15 16 17 18 20 15 16 17 18 19 20 20
20 20 15 16 17 18 19 20 15 16 17 18 19 20 20 15 16 17 18 19 20 15 16 17 18 19
20 20 15 16 17 18 19 20 17 17 17]

;НАЧАЛЬНЫЕ ЗНАЧЕНИЯ ВАЖНЫХ ПАРАМЕТРОВ

set nmres ["LENGTH ROAD:" "TIME GAME:" "CINETIC FIRE:" "SIMPLE
FIRE:" "ENEMY HOUSE:" "ENEMY GUNS:" "HAVE BONUS:"
"UNDERMINING:" "GAME EFFECT:"]

set tm 0 ;начальное значение времени

```

set gsm 300 ;отведенное танку расстояние
set enemy 5 ;число баз противника
set bonus 3 ;число бонусов
set bns 0 ;дополнительная переменная для учета бонусов
set mns 10 ;число мин
set boom? false ;признак попадания на мину
set cnt 3 ;количество гиперснарядов
set nmb 5 ;количество простых снарядов
set gns 4 ;количество пушек противника
set bmflag? false
set bmxcor 0
set bmycor 0

```

;ВЫВОД ФОНА СКРИНШОТА И НАДПИСИ С СИМВОЛИКОЙ ИГРЫ

```

ask patches [ ;выводим фон
set pcolor 96 ;оранжевым цветом
]
let k 0 ;выводим надпись скрина
repeat length nx [
  ask patch ((item k nx) - 3) item k ny [set pcolor 63]
  wait 0.01 ;с задержкой 0.7 сек.
  set k k + 1
]
;выводим на скрин
create-bpla 1 [set shape "airplane" set size 8 set color 67 setxy 14 12 set heading
35 ] ;БПЛА
create-bpla 1 [set shape "target" set size 6 set color red setxy 25 22 set heading
35] ;прицел
wait 2 ;задержка скрина на 4 сек.

```

;ПЕРЕХОД К УРОВНЮ ИГРЫ - УНИЧТОЖЕНИЕ ВСЕХ ЭЛЕМЕНТОВ
СКРИНШОТА

```

ask bpla [die] ;уничтожаем все танки
ask patches [ ;выводим фон уровня игры
set pcolor green ;зеленым цветом
]

```

;ВЫВОД ПЕРСОНАЖА ИГРЫ

```

set-default-shape bpla "airplane"
create-bpla 1 [ ;создание 1 танка
set size 2 ;размера 1.7
set shape "airplane" ;форма акт.агента в виде танка
setxy random max-rxcor random max-rycor ;со случайными координатами
set color red ;синего цвета

```

```
set tx [xcor] of bpla ;сохранение координаты танка
set ty [ycor] of bpla
]
```

;ВЫВОД ПРОТИВНИКОВ

```
create-prots enemy [;создание 5 противников
  setxy random world-width random world-height ;со случайными
координатами
  set size 2.5 ;размер - 1.5
  set shape "house colonial" ;противник в форме дома
  set color 15 + random 35
] ;случайный цвет в диапазоне [15;50]
```

;ВЫВОД МИН ПРОТИВНИКОВ

```
create-minas mns [
  setxy random world-width random world-height ;со случайными
координатами
  set shape "mina" ;противник в форме дома
  set color brown
  set lftime (2 + random 10) ;частота раскрытия мин перед игроком
]
```

;ВЫВОД БОНУСОВ ДЛЯ ПЕРСОНАЖА

```
create-lives bonus [
  setxy random world-width random world-height
  set vrlife random 40 ;живут в течении 40 сек. игры
  set size 2
  set shape "heart"
]
```

;ВЫВОД ПУШЕК ПРОТИВНИКИ

```
create-guns gns [
  setxy random world-width random world-height
  set size 2
  set color red
  set shape "gun"
  set bmflag? false
  face aircraft 2 ;поворот пушек в сторону танка игрока
]
reset-timer ;запускаем таймер игры
end
```

```
.;*****
;ПРОЦЕДУРА ДВИЖЕНИЯ КИНЕТИЧЕСКОГО СНАРЯДА
```

```

to sprfire ;открыть огонь высококинетическими снарядами
  ifelse (cnt > 0) [ ;контроль числа снарядов
    create-balls 1 [ ;создание 1 снаряда
      setxy item 0 tx item 0 ty ;на позиции танка
      set shape "dart" ;в виде окружности
      set color yellow ;желтого цвета
      set heading [heading] of aircraft 2 ;в направлении танка
      skydot1 ;вызов процедуры движения снаряда
    ]
    set cnt cnt - 1 ;уменьшаем счетчик снарядов после выстрела
  ]
  [user-message("Гиперскоростные снаряды закончены !!!")] ;вывод
  сообщения об отсутствие снарядов
end

```

;РЕКУРСИВНАЯ ПРОЦЕДУРА ДВИЖЕНИЯ КИНЕТИЧЕСКОГО СНАРЯДА

```

to skydot1
  ask balls [ ;для каждого снаряда
    ;УНИЧТОЖЕНИЕ ДОМОВ ВРАГА
    ask prots in-radius 3 [ ;в радиусе 3 ед.
      set color yellow ;цвет снаряда желтый
      set shape "fire" ;меняем текущую форму снаряда на форму взрыва
      wait 0.1 ;ждем 0.1 сек.
      set size 3 ;меняем размер взрыва на 3
      wait 0.2 ;ждем 0.2 сек.
      die ;уничтожаем снаряд
    ]
    ;УНИЧТОЖЕНИЕ ПУШЕК ВРАГА
    ask guns in-radius 3 [ ;в радиусе 3 ед.
      set color violet ;цвет снаряда желтый
      set shape "fire" ;меняем текущую форму снаряда на форму взрыва
      wait 0.1 ;ждем 0.1 сек.
      set size 3 ;меняем размер взрыва на 3
      wait 0.2 ;ждем 0.2 сек.
      die ;уничтожаем снаряд
    ]
  ]
  fd 3 ;вперед на 3 ед.
  wait 0.1 ;ждем 0.1 сек.
  let sbal (distancexy item 0 tx item 0 ty) ;расстояние от танка до снаряда
  ifelse (sbal > 15) [die] [skydot1] ;если расстояние больше 15 ед., то
  уничтожаем снаряд, иначе продолжаем движение
end

```

```

;*****
;ПРОЦЕДУРА ДВИЖЕНИЯ ПРОСТЫХ СНАРЯДОВ
to fire ;открыть огонь простыми снарядами
  ifelse (nmb > 0) [ ;счетчик простых снарядов
    let fprot count prots ;счетчик уничтоженных противников
    let tb timer ;время полета простого снаряда
    create-balls 1 [ ;создание 1 снаряда
      setxy item 0 tx item 0 ty ;на позиции танка
      set shape "rocket" ;в виде ракеты
      set color orange ;оранжевого цвета
      set heading [heading] of aircraft 2 ;в направлении танка
    ]
  while [count prots = fprot] [ ;пока число противников не изменилось
    ask balls [fd 1 wait 0.1] ;снаряд летит
    ask balls [ ;уничтожаем противника
      ;УНИЧТОЖЕНИЕ ДОМОВ ВРАГА
      ask prots in-radius 2 [
        set color orange ;цвет снаряда оранжевый
        set shape "fire" ;меняем текущую форму снаряда на форму взрыва
        wait 0.1 ;ждем 0.1 сек.
        set size 3 ;меняем размер взрыва
        wait 0.2 ;ждем 0.2 сек.
        die ;уничтожение агента-противника
      ]
      ;УНИЧТОЖЕНИЕ ПУШЕК ВРАГА
      ask guns in-radius 2 [
        set color violet ;цвет снаряда оранжевый
        set shape "fire" ;меняем текущую форму снаряда на форму взрыва
        wait 0.1 ;ждем 0.1 сек.
        set size 3 ;меняем размер взрыва
        wait 0.2 ;ждем 0.2 сек.
        die ;уничтожение агента-противника
      ]
    ]
  ]
; время полета больше 1 сек. - уничтожить снаряд и уменьшить счетчик
снарядов
  if (timer - tb) > 1 [ask balls [die] set nmb nmb - 1 stop]
]
ask balls [die] ;уничтожить снаряд после уничтожения противника
set nmb nmb - 1 ;уменьшить число снарядов
]
[user-message("Простые ракеты закончились !!!")] ;вывод инфо об
отсутствии снарядов

```


end

;

;ПРОЦЕДУРЫ-ОБРАБОТЧИКИ КЛАВИШ УПРАВЛЕНИЯ ТАНКОМ

;ВЛЕВО

to lft ;поворот танка влево
ask bpla [lt angle]
end

;ВПРАВО

to right ;поворот танка вправо
ask bpla [rt angle]
end

;ВПЕРЕД

to frwd ;движение танка вперед
;с каждым нажатием на кнопку назад, уменьшается ГСМ на speed ед.
ask bpla [fd 0.1 * speed set gsm gsm - 1]
set tx [xcor] of bpla ;сохранение координат танка
set ty [ycor] of bpla
ask lifes [;захват бонуса при движении вперед
if (distancexy item 0 tx item 0 ty) < 2 [;на расстоянии менее 2 ед.
set gsm gsm + 50 ;величина бонуса
set bns bns + 1 ;уменьшаем число бонусов
die
]
]
ask minas [;уничтожение танка при попадании на мину
if (distancexy item 0 tx item 0 ty) < 1 [;на расстоянии менее 2 ед.
set boom? true
set color orange ;цвет снаряда оранжевый
set shape "fire" ;меняем текущую форму снаряда на форму взрыва
wait 0.1 ;ждем 0.1 сек.
set size 3 ;меняем размер взрыва
wait 0.2 ;ждем 0.2 сек.
die ;уничтожение агента-противника
]
]
end

;НАЗАД

```

to bcks ;движение танка влево
;с каждым нажатием на кнопку назад, уменьшается ГСМ на 1 ед.
ask bpla [bk 0.1 * speed set gsm gsm - 1]
set tx [xcor] of bpla ;сохранение координат танка
set ty [ycor] of bpla
ask lifes [ ;захват бонуса при движении назад
if (distancexy item 0 tx item 0 ty) < 1 [ ;на расстоянии менее 2 ед.
set gsm gsm + 50 ;величина бонуса
set bns bns + 1 ;уменьшаем число бонусов
die
]
]
ask minas [ ;уничтожение танка при попадании на мину
if (distancexy item 0 tx item 0 ty) < 2 [ ;на расстоянии менее 2 ед.
set boom? true
set color orange ;цвет снаряда оранжевый
set shape "fire" ;меняем текущую форму снаряда на форму взрыва
wait 0.1 ;ждем 0.1 сек.
set size 3 ;меняем размер взрыва
wait 0.2 ;ждем 0.2 сек.
die ;уничтожение агента-противника
]
]
end

```

,*****

;ПРОЦЕДУРА УЧЕТА ВРЕММЕНИ

```

to time ;процедура учета характеристик прохождения уровня игры
;игра заканчивается, если прошло время прохождения timer уровня игры
;игра заканчивается, если закончилось топливо gsm
;игра заканчивается, если игрок уничтожил всех противников - prots пустое
ifelse ((timer > 90) or (gsm <= 0) or (not any? prots) or boom?) [

```

;БЛОК "ТО" СРАБОТАЕТ ПРИ СРАБАТЫВАНИИ УСЛОВИЙ ОКОНЧАНИЯ ИГРЫ

```

let encnt (enemy - count prots)
let guncnt (gns - count guns)
let bal round(((encnt + guncnt) * 100)/ (gns + enemy)) ;расчет эффективности
игры
set nmval (list gsm tm cnt nmb encnt guncnt bns boom? bal) ;список названий
полей
set cnt 0 ;сброс количества гиперснарядов
set nmb 0 ;сброс количества простых снарядов

```

;УНИЧТОЖЕНИЕ ТАНКА

```
ask bpla [  
    set color yellow ;  
    set shape "fire" ;  
    wait 0.1 ;  
    set size 3 ;  
    wait 0.2 ;  
    die ;  
]  
wait 0.3
```

;УНИЧТОЖЕНИЕ ПРОТИВНИКА

```
ask prots [  
    set color yellow ;  
    set shape "fire" ;  
    wait 0.1 ;  
    set size 3 ;  
    wait 0.2 ;  
    die ;  
]  
wait 0.3
```

;УНИЧТОЖЕНИЕ БОНУСОВ

```
ask lifes [  
    set color yellow ;  
    set shape "fire" ;  
    wait 0.1 ;  
    set size 3 ;  
    wait 0.2 ;  
    die ;  
]  
wait 0.3
```

;УНИЧТОЖЕНИЕ МИН ПРОТИВНИКА

;бонусы также уничтожаются, если до окончания уровня игрок их не использовал

```
ask minas [  
    set color yellow ;  
    set shape "fire" ;  
    wait 0.1 ;  
    set size 3 ;  
    wait 0.2 ;  
    die ;  
]  
]
```

```
wait 0.3
```

```
;УНИЧТОЖЕНИЕ ПУШЕК ПРОТИВНИКА
```

```
ask guns [  
    set color yellow ;  
    set shape "fire" ;  
    wait 0.1 ;  
    set size 3 ;  
    wait 0.2 ;  
    die ;  
]  
wait 0.3
```

```
;УНИЧТОЖЕНИЕ БОМБ ПУШЕК ПРОТИВНИКА
```

```
ask bombs [  
    set color yellow ;  
    set shape "fire" ;  
    wait 0.1 ;  
    set size 3 ;  
    wait 0.2 ;  
    die ;  
]
```

```
;ВЫВОД СТАТИСТИКИ ПОСЛЕ ИГРЫ
```

```
ask patches [set pcolor green] ;заполняем фон статистики  
;вывод информации о результатах игры  
let z 0 ;индекс надписи в статистике  
let nmx 10 ;координаты расположения полей информации  
let nmy 28  
create-infos 1 [ ;вывод заголовка статистики  
    setxy 15 30  
    set color green  
    set label-color black  
    set size 2  
    set heading 0  
    set label "GAME STATISTICS"  
]  
repeat 9 [ ;вывод первого столбы статистики  
    create-infos 1 [  
        setxy nmx nmy  
        set color green  
        set label-color black  
        set size 2  
        set heading 0
```

```

    set label (item z nmres)
  ]
create-infos 1 [ ;вывод второго столбца статистики
  setxy (nm $x$  + 3) nmy
  set color green
  set label-color blue
  set size 2
  set heading 0
  set label (item z nmval)
  ]
set nmy nmy - 2
set z z + 1
]
stop ;стоп игры
set tm timer
]

```

```

;БЛОК "ИНАЧЕ" РАБОТАЕТ ПОСТОЯННО;
;ИДЕТ УЧЕТ ВРЕМЕНИ ЖИЗНИ БОНУСОВ И МИН ПРОТИВНИКА
[
  set tm timer

```

```

;УНИЧТОЖЕНИЕ БОНУСОВ
  ask lifes [ ;бонус уничтожается, если прошло время его жизни
    if (vrlife < tm) [die]
  ]

```

```

;МИГАНИЕ МИН ПРОТИВНИКА
  ask minas [ ;периодически показываем игроку расположение мин
    ifelse (round(tm) mod lftime = 0)
      [set hidden? true ] ;показываем мины
      [set hidden? false ] ;скрываем мины
  ]

```

```

;ПОВОРОТ ПУШЕК К ТАНКУ И ВЫЧИСЛЕНИЕ РАССТОЯНИЯ ДО НЕГО
  ask guns [
    face aircraft 2 ;в ходе движения танка пушки поворачиваются к нему
    let sgun (distancexy item 0 tx item 0 ty) ;расстояние от танка до снаряда
    if (sgun <= 6) [set bmflag? true set bmxcor xcor set bmycor ycor]
  ]

```

```

;СТРЕЛЬБА ПУШЕК ПО ТАНКУ ПО СИГНАЛУ ОТ РАССТОЯНИЯ
  ifelse bmflag? and (count bombs = 0) [
    create-bombs 1 [

```

```
set shape "dot"  
set color red  
setxy bmxcor bmycor  
face aircraft 2  
]  
]  
[ask bombs [fd 0.0005]]
```

;ФИКСАЦИЯ ПОПАДАНИЯ БОМБЫ ОТ ПУШКИ В ТАНК И ИХ
УНИЧТОЖЕНИЕ

```
ask bombs [  
  if (distancexy item 0 tx item 0 ty) < 2 [ ;на расстоянии менее 2 ед.  
    set color magenta ;цвет снаряда оранжевый  
    set shape "fire" ;меняем текущую форму снаряда на форму взрыва  
    wait 0.1 ;ждем 0.1 сек.  
    set size 3 ;меняем размер взрыва  
    wait 0.2 ;ждем 0.2 сек.  
    set boom? true  
    die ;уничтожение агента-противника  
  ]  
]  
]  
end
```

Для замечаний

Учебное издание

Алмаз Раилевич Исхаков

Разработка программного тренажера в среде
многоагентного моделирования NetLogo

Подписано в печать 03.02.2021

Формат 60X84/16. Компьютерный набор. Гарнитура Times New Roman.

Отпечатано в ризографе. Усл. печ. л. – 3,0 Уч.-изд.л. – 2,8

Тираж 100 экз. Заказ № 21