ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ ИМИТАЦИОННОГО ИССЛЕДОВАНИЯ ВАРИАНТОВ РАЗМЕЩЕНИЯ ЛОГИСТИЧЕСКИХ ЦЕНТРОВ С ИСПОЛЬЗОВАНИЕМ КАРТОГРАФИЧЕСКИХ ТЕХНОЛОГИЙ

А.С. Арсеньев (Казань)

Введение

Экономически целесообразное размещение транспортно-логистических центров (ТЛЦ), оптимизация их функций, а также взаимодействие центров в рамках единой и согласованной сети является важнейшей задачей при оптимизации транспортно-логистической системы региона (ТЛС). Еще более актуальной эта задача становится в свете реализации целей, поставленных в Транспортной стратегии России [1] и Государственной Программе развития транспортной системы Республики Татарстан [2].

Действительно ТЛС любого региона, в том числе и РТ, непрерывно развивается. Одним из последних и важнейших структурных изменений этой системы стало появление ТЛЦ. В настоящее время крупные торговые сети закупают товары по всему миру и везут их к местам продажи. Производители товаров, например, нефтехимический комплекс, огромные объемы полуфабрикатов и готовой продукции, большая часть которых транспортируется на переработку на предприятия РТ, России и мира. Также производители нуждаются в закупке большого количества сырья и материалов. Все вместе означает постоянное движение миллионов тонн по автомобильным и железным дорогам, водным или авиационным транспортом [3]. Все это движение грузов представляется в виде многоуровневых логистических цепочек и управляется логистическими операторами. В этой сложнейшей системе очень важными звеньями являются ТЛЦ, так как у операторов неизбежно возникают вопросы: временного хранения товара, перегрузки с одного вида транспорта на другой, необходимость укрупнения/разукрупнения партий товара, таможенного или страхового оформления и т.д.

При оптимизации сети ТЛЦ необходимо:

1. решить задачу размещение ТЛЦ с учетом существующей схемы и перспективных планов размещения производительных сил на данной территории, сложившихся объемов и направлений грузопотоков;

2. для каждого ТЛЦ определить логистический функционал и перерабатывающие мощности;

3. осуществить подбор характеристик ТЛЦ таким образом, чтобы все центры дополняли друг друга, обеспечивали их взаимодействие и при этом решали в полном объеме задачи и государства, и бизнеса.

Одним из основных и результативных научных методов для решения подобных задач является представление ТЛС в виде дискретно-событийной модели и исследование этой модели с помощью средств имитационного моделирования. Сначала создается детальнейшая модель работы с использованием языка ИМ, затем с ней проводится направленная серия экспериментов. В АН РТ сформулирована усовершенствованная методология и собственная программная технология такого имитационного исследования [4], разработаны программные инструментальные средства для создания систем автоматизации имитационных исследований (САИИ) в любых предметных областях. Одной из областей практического применения САИИ стало создание практической модели сети логистических центров в РТ [5].

Одним из важнейших элементов имитационного исследования является удобный и доступный язык ввода исходных данных и анализа результатов. В задаче размещения и оптимизации работы ТЛЦ наиболее удобным и очевидным способом реализации ввода данных в модель и анализа результатов является работа с картой территории, на которой располагаются ТЛЦ. Нами была поставлена задачи доработки языка взаимодействия исследователя с моделью технологий работы с картографической информацией в рамках созданной модели.

Картографические возможности для совершенствования языка взаимодействия исследователя с моделью

Имитационная модель сети ТЛС предназначена для исследования основных показателей функционирования системы с целью повышения эффективности ее работы. Во-первых, при проектировании системы. Анализ различных проектных решений по инфраструктуре, планируемой пропускной способности транспортных коммуникаций, потенциальным грузопотокам. Во-вторых, в процессе эксплуатации системы. Для обеспечения оптимальных показателей доставки грузов по времени, стоимости, качеству и надежности, а также для поиска «узких мест». В-третьих, при модернизации системы. Для оценки различных вариантов устранения «узких мест».

Важней составляющей успешного внедрения САИИ на практике является то, каким образом пользователь модели может с ней работать. Будет ли язык взаимодействия при вводе исходных данных простым, удобным и наглядным? Насколько доступно и понятно управление процессом экспериментирования при исследовании? Соответствует ли способ представления полученных результатов форме, принятой в данной предметной области?

Одним из важнейших элементов языка взаимодействия в последнее время становится картографический способ ввода данных и анализа результатов. Интерактивная работа с картой позволит вводить данные для моделирования на карту с привязкой к реальным географическим координатам, а также просматривать результаты моделирования, выведенные на карту. Программное обеспечение, реализующее данный функционал, в случае моделирования ТЛС должно иметь следующие основные возможности:

1. интерактивная работа с картой, скачанной с одной из геоинформационных систем (навигация по карте, масштабирование, детальный просмотр объектов в привязке к результатам моделирования, копирование и распечатка) в процессе имитационного исследования;

2. актуализация состояния транспортных коммуникаций для проведения имитационного эксперимента – формирование графа дорог, нанесение новых дорог, ввод и просмотр характеристик дорог и их участков (название, категория, количество полос, направление, покрытие, пропускная способность и т.д.);

3. возможность размещения на карте новых логистических объектов (например, ТЛЦ) с привязкой к географическим координатам и указанием их основных характеристик, требуемых при моделировании;

4. настройка и изменение характеристик ТЛЦ при формировании различных сценариев исследования, интерактивный детальный просмотр характеристик ТЛЦ по результатам моделирования;

5. использование картографической информации при формировании отчета по результатам моделирования.

Выбор основных инструментов разработки

В качестве геоинформационной основы для картографической подсистемы САИИ ТЛС (далее приложение) можно выбрать любой некоммерческий сервис – OpenStreetMap, Яндекс-Карта, Google и др. Все они достаточно функциональны, широко используются на практике. Учитывая ранее имеющийся опыт, нами в качестве базисного средства был выбран бесплатный сервис OpenStreetMap (OSM). В основном элементе управления нашего приложения расположим саму карту, необходимой для использования в модели области, которую можно загрузить в «OSM-формате» с сайта openstreetmap.org. «OSM формат» по своей сути представляет собой некий xml-файл, в котором содержатся все объекты карты заданной области, в том числе, и их характеристики [6]. Также там хранится граф дорог, который необходим для просчета маршрутов движения транспортных средств.

Далее необходимо решить вопрос по выбору платформы для разработки. Основным критерием выбора технологии будет обеспечение необходимой производительности программного обеспечения, то есть просмотр карты должен быть плавным, даже при отображении большого количества объектов модели. В первой версии приложения использовалась технология WPF компании Microsoft.

Технология WPF. Оптимизация производительности

Так как наше приложение будет картографического типа, то есть будет содержать в себе огромное множество объектов для отрисовки, то при выборе платформы Windows Presentation Foundation (WPF) встает проблема производительности. WPF предоставляет несколько уровней доступа к изображениям и службам отрисовки. На самом верхнем уровне находятся объекты Shape. Они просты в использовании, доступны в разметке и участвуют в системе событий WPF. Все объекты фигур наследуются от класса Shape. Доступные объекты фигур включают в себя: Ellipse, LinePath, Polygon, Polyline, Rectangle.

Однако такое проектное решение не подходит для нашего приложения, так как придется отображать громадное число графических элементов, каждый из которых представляет собой самостоятельный объект с возможностью подписки на всевозможные события. В данной ситуации основные проблемы производительности будут связаны не со сложностью графики, а с огромным количеством индивидуальных графических элементов.

На следующем уровне находятся объекты Drawing. Они являются более простыми конструкциями, чем объекты Shape и, соответственно, меньше влияют на производительность. Объекты Drawing не являются производными класса FrameworkElement и предоставляют упрощенную реализацию отрисовки фигур, изображений и текста.

Существуют четыре типа объектов Drawing:

– объект GeometryDrawing рисует фигуру;

– объект ImageDrawing рисует изображение;

– объект GlyphRunDrawing выводит текст;

– объект DrawingGroup выводит другие объекты Drawing, с его помощью можно объединять разные объекты Drawing в один составной объект Drawing.

Объект GeometryDrawing используется для отрисовки содержимого геометрического объекта. Класс Geometry и классы, производные от него (например, CombinedGeometry, Ellips eGeometry и PathGeometry), предоставляют средства для отрисовки двумерных изображений, а также обеспечивают поддержку проверки на попадание мышкой и отсечения. Например, геометрические объекты можно использовать для определения области элемента управления или для определения отсекаемой области, применяемой к изображению. Но в нашем случае, даже если произвести замену всех элементов Path облегченными объектами Geometry, расходы на отрисовку все равно сведут на нет производительность приложения.

Итак, самым низким уровнем визуализации в WPF является использование класса Visual. Все сводится к тому, что все графические элементы в конечном счете будут представлять один элемент Visual. Нас интересует класс-наследник *DrawingVisual*, унаследованный от ContainerVisual и добавляющий поддержку, необходимую для «рисования» графического содержимого, которое требуется поместить в визуальный объект [7].

Чтобы использовать объекты DrawingVisual, первоначально необходимо создать контейнер для объектов. Унаследовать его нужно от класса FrameworkElement, что позволит нам использовать его в разметке, а также подключать к нему события. При создании контейнера необходимо объявить объект-коллекцию типа VisualCollection, где мы будем хранить сами визуальные элементы. Добавлять эти элементы в коллекцию можно при помощи

метода Add(). В следующем примере рассмотрим создание контейнера для объектов, а так же добавление в его коллекцию прямоугольника, эллипса и обычной надписи [8].

```
public class MyVisualHost : FrameworkElement
{
   // Create a collection of child visual objects.
   private VisualCollection _children;

   public MyVisualHost()
   {
      _children = new VisualCollection(this);
      _children.Add(CreateDrawingVisualRectangle());
      _children.Add(CreateDrawingVisualText());
      _children.Add(CreateDrawingVisualEllipses());

      this.MouseLeftButtonUp += new System.Windows.Input.MouseButtonEventHandler(MyVis
ualHost_MouseLeftButtonUp);
   }
}
```

Чтобы нарисовать содержимое в DrawingVisual, вызывается метод DrawingVisual. RenderOpen(). Этот метод возвращает DrawingContext, который можно использовать для определения содержимого визуального объекта. Далее необходимо произвести рисование. После того как рисование завершено, следует вызвать DrawingContext.Close().

Так как мы создаем картографическое приложение, нам необходимо, чтобы к нашим графическим объектам подключались события мышки и другие. Так как объектам DrawingVisual по умолчанию подключить события нельзя, мы будем использовать события объекта-контейнера, которые будут действительны для всех его дочерних элементов [8].

```
void MyVisualHost_MouseLeftButtonUp(object sender, System.Windows.Input.
MouseButtonEventArgs e)
{
   System.Windows.Point pt = e.GetPosition((UIElement)sender);
   VisualTreeHelper.HitTest(this, null, new HitTestResultCallback(myCallback), new
PointHitTestParameters(pt));
}
public HitTestResultBehavior myCallback(HitTestResult result)
{
   if (result.VisualHit.GetType() == typeof(DrawingVisual))
   {
      if (((DrawingVisual)result.VisualHit).Opacity == 1.0)
      {
         ((DrawingVisual)result.VisualHit).Opacity = 0.4;
      }
      else { ((DrawingVisual)result.VisualHit).Opacity = 1.0;  }
   }
   return HitTestResultBehavior.Stop;
}
```

Класс DrawingVisual доступен с версии .NET Framework 3.0, поэтому можно смело его использовать в нашем приложении, не задумываясь о проблеме совместимости.

На рис. 1 представлен пример изображения при работе с приложением.

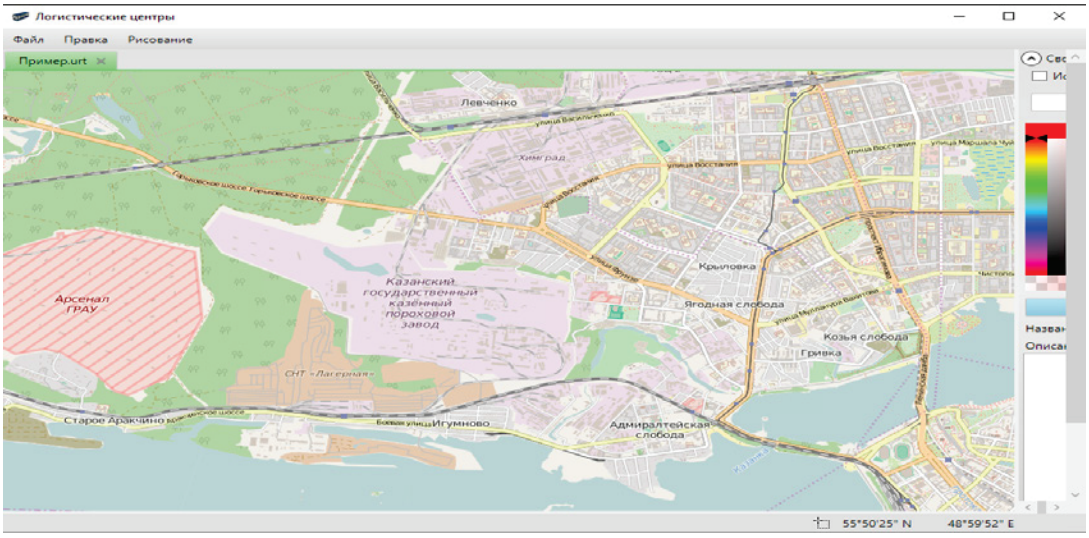

Рис. 1. Прототип приложения

Выводы

По результатам разработок и исследования, представленных в данной публикации, можно сделать основные выводы:

1. современные IT технологии и методология ИМ позволяют органично вписать картографическое представление для ввода исходных данных и для представления результатов моделирования;

2. практическое использование такого способа ввода и вывода информации позволяет существенно упростить процесс работы с моделью и приблизить ее к реальным потребителям;

3. подходящим инструментом создания картографического приложения для взаимодействия исследователя с моделью является технология WPF, позволяющая обеспечить высокую производительность при обработке больших объемов графических данных.

Литература

1. Транспортная стратегия Российской Федерации на период до 2030 г. М.: 2008-2014 г. 365 с. Утверждена распоряжением Правительства Российской Федерации от 22 ноября 2008 г. № 1734-р.
2. Государственная программа «Развитие транспортной системы Республики Татарстан на 2014-2022 годы». Казань: 2013-2016г. 45 с. Утверждена Постановлением Кабинета Министров Республики Татарстан № 1012 от 20.12.2013 г.
3. СОЮЗМОРНИИПРОЕКТ – http://www.smniip.ru/index.php/stati1?id=39.
4. Имитационные исследования в среде моделирования GPSS STUDIO : учеб. пособие / В.В. Девятков, Т.В. Девятков, М.В. Федотов; под общ. ред. В.В. Девяткова. М. : Вузовский учебник : ИНФРА-М, 2018. 283 с.
5. Инновации в управлении транспортными логистическими системами: монография / О.Н. Рожко, В.В. Хоменко, Е.В. Макарова; под общ. ред. О.Н. Рожко. Казань : Изд-во «Бриг». 2015. 187 с.
6. OpenStreetMap – http://www.openstreetmap.org.
7. https://professorweb.ru/my/WPF/graphics_and_animation/level14/14_1.php.
8. MSDN - https://msdn.microsoft.com.