Приднестровский государственный университет им. Т. Г. Шевченко

Физико-математический факультет

Кафедра прикладной математики и информатики

Квалификационная работа

«Допущено к защите»

зав. каф. пр. математики и информатики,

доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Коровай

ПРИМЕНЕНИЕ ANYLOGIC ДЛЯ РЕШЕНИЯ ЭКОНОМИЧЕСКИХ ЗАДАЧ

Выполнил:

студент 403 гр. д/о

физ.-мат. факультета

Епифанов Андрей Сергеевич

Руководитель:

ст. преподаватель кафедры ПМ и И

Белая Елена Ивановна

Тирасполь, 2017

Оглавление

[Аннотация 3](#_Toc485739761)

[Введение 4](#_Toc485739762)

[1. Основы моделирования систем 5](#_Toc485739763)

[1.1. Понятия модели и моделирования 5](#_Toc485739764)

[1.2. Классификация моделей 6](#_Toc485739765)

[1.3. Особенности имитационного моделирования 8](#_Toc485739766)

[1.4. Среда имитацмонного моделирования Anylogic 11](#_Toc485739767)

[1.4.1. Описание продукта 11](#_Toc485739768)

[1.4.2. Пользовательский интерфейс 12](#_Toc485739769)

[1.4.3. Средства для моделирования 14](#_Toc485739770)

[1.4.4. Элементы построения модели 16](#_Toc485739771)

[1.5. Вывод по главе 17](#_Toc485739772)

[2. Имитационная модель общественного транспорта в AnyLogic 19](#_Toc485739773)

[2.1. Постановка задачи 19](#_Toc485739774)

[2.2. Построение модели общественного транспорта 19](#_Toc485739775)

[2.2.1. Создание модели и разметка рабочего пространства 19](#_Toc485739776)

[2.2.2. Модель поведения автобусов 23](#_Toc485739777)

[2.3. Полученные результаты 31](#_Toc485739778)

[3. Глава 3 32](#_Toc485739779)

[3.1. Пункт 1 32](#_Toc485739780)

[3.2. Пункт 2 32](#_Toc485739781)

[3.3. Пункт 3 32](#_Toc485739782)

[3.4. Пункт 4 32](#_Toc485739783)

[Заключение 33](#_Toc485739784)

[Литература 34](#_Toc485739785)

Аннотация

В данной квалификационной работе рассматривается одна из наиболее распространенных задач, встречающихся в математическом программировании –транспортная задача. Проведено исследование особенностей транспортных задач.

Для решения применяется метод потенциалов, дано подробное изложение этапов метода с иллюстрацией их на конкретных примерах. Приводятся доказательства двух теорем: об условии разрешимости транспортной задачи и об условиях оптимальности плана перевозок. Для нахождения исходного опорного плана применяются методы северо-западного угла и минимального элемента (минимальной стоимости).

Рассмотрено решение конкретных экономических задач: задачи о транспортировке производственных и продуктовых товаров потребителям, задача перегона вагонов к пунктам погрузки, транспортировка материалов на строительные объекты.

Решение транспортных задач не может быть выполнена вручную, в приемлемые сроки, при количестве поставщиков и потребителей более пяти. В связи с этим возникаем вопрос о создании и разработке программного продукта для оптимизации решения транспортных задач.

На основе теории и разработанного алгоритма написана прикладная программа на языке С#, позволяющая рационально решать транспортные задачи.

Целью дипломной работы была разработка информационной системы и программно-аппаратного комплекса для работы с информационной системой.

Информационная система должна удовлетворять ряду технических требований, таких как процесс хранения, заполнения и изменения данных.

Реализовать возможность бесконтактной проверки документа, а также облегчить обработку информации, реализовать возможность хранения данных на сервере.

Введение

Моделирование является самым распространенным средством изучения объектов, явлений и процессов реального мира. Моделирование необходимо в тех случаях, когда изучать реальные объекты и процессы затруднительно или практически невозможно и существенно упрощает и удешевляет разработку и оптимизацию сложных и дорогих систем.

Актуальность выбранной темы находится на достаточно высоком уровне, обусловлено это тем, что решение тех или иных экономических задач в большинстве случаев невозможно в реальном мире из-за дороговизны исследования, поэтому моделирование является неотъемлемой частью решения таких задач.

Предмет исследования: основные принципы и методы моделирования систем. Объект исследования: оптимизация затрат на перевозку пассажиров в общественном транспорте.

Цельюработы является приобретение навыка работы со средой имитационного моделирования систем AnyLogic и применение ее к конкретным задачам.

Для достижения поставленной цели следует решить следующие *задачи*:

- изучить литературу по теме моделирования,

- научиться работать со средой AnyLogic,

- разработать модель движения общественного транспорта,

- сравнить затраты и прибыль перевозок на различных маршрутах.

Первая глава квалификационной работы раскрывает теоретические аспекты моделирования.

Во второй главе наглядно показан процесс построения модели в AnyLogic и получение данных о затратах.

# Основы моделирования систем

## Понятия модели и моделирования

В своей деятельности человек очень часто использует модели, то есть создает образ того объекта, явления или процесса, с которым ему предстоит работать.

**Модель -** это некий новый упрощенный объект, который отражает существенные особенности реального объекта, процесса или явления.  
Анализ модели и наблюдение за ней позволяют познать суть реально существующего, более сложного объекта, процесса, явления, называемого прототипом или оригиналом. Для моделей существует ряд общих требований:

* Адекватность – достаточно точное отображение свойств объекта;
* Полнота – предоставление получателю всей необходимой информации об объекте;
* Гибкость – возможность воспроизведения различных ситуаций во всем диапазоне изменения условий и параметров;
* Трудоемкость разработки должна быть приемлемой для имеющегося времени и программных средств.

**Моделирование** представляет из себя метод решения задач, при котором исследуемую систему заменяем на более простой объект, который описывает нашу систему и называется моделью. Моделирование применяется в тех случаях, когда проведение экспериментов над системой невозможно, например, из-за высокой стоимости или длительности проведения эксперимента в реальном времени.

**Имитационной моделью** называется компьютерная программа, которая описывает структуру и демонстрирует поведение системы во времени. Имитационная модель позволяет получать подробную статистику о разных аспектах функционирования системы в зависимости от входных данных.

**Имитационное моделирование** – это разработка и постановка экспериментов на моделях. Целью моделирования является принятие целесообразных управленческих решений. Компьютерное моделирование становится обязательным этапом в принятии ответственных решений во всех областях деятельности человека в связи с усложнением систем, в которых человек должен действовать и которыми он должен управлять. Знание принципов и возможностей имитационного моделирования, умение создавать и применять модели являются необходимыми требованиями к инженеру, менеджеру, бизнес-аналитику.

Современные системы моделирования поддерживают все из новейших информационных технологий, включая развитые графические оболочки для целей конструирования моделей и интерпретации выходных результатов моделирования, мультимедийные средства, анимацию в реальном масштабе времени, объектно-ориентированное программирование, Internet - решения и др.

В настоящее время предметную область имитационного моделирования связывают в первую очередь с системным анализом, занимающимся исследованиями сложных систем в макроэкономике, геополитике, экологии, при создании автоматизированных систем управления и пр.

## Классификация моделей

Объектов моделирования существует огромное количество. И для того, чтобы ориентироваться в их многообразии необходимо все это классифицировать, то есть каким-либо образом упорядочить, систематизировать.

При классификации объектов по «родственным» группам необходимо правильно выделить некий единый признак (параметр, а затем объединить те объекты, у которых он совпадает). Рассмотрим наиболее распространенные признаки, по которым можно классифицировать модели.

РАСПИСАТЬ КАЖДУЮ

1. С учетом фактора времени:

динамические;

статические.

1. По области использования:

учебные;

опытные;

игровые;

научно-технические;

имитационные.

1. По области знаний:

математические;

химические;

физические;

географические;

ит.д.

1. По способу реализации:

компьютерные;

некомпьютерные.

1. По способу представления:

материальные;

информационные

вербальные;

графические;

математические;

табличные;

специальные.

Процесс построения информационных моделей с помощью формальных языков (математических, логических и т.д.) называется формализацией   
Более полное определение формализации – это приведение (сведение) существенных свойств и признаков объекта моделирования к выбранной форме.

Формами представления информационной модели могут быть: словесное описание, таблица, схема, чертеж, формула, алгоритм, компьютерная программа и т.д.

## Особенности имитационного моделирования

При имитационном моделировании структура системы отображается в модели, а процессы ее функционирования проигрываются (имитируются) на построенной модели. Выделяют статическое описание структуры системы, для чего нужно выполнять структурный анализ процессов, и описание динамики взаимодействий элементов системы, для чего нужно построить функциональную модель динамических процессов. При программной реализации имитационного моделирования элементам системы ставятся в соответствие некоторые программные компоненты, а состояние этих элементов описывается с помощью переменных. Моделирующий алгоритм имитирует функционирование отдельных элементов, которые взаимодействуют или обмениваются информацией. Есть алгоритм изменения переменных, описывающий состояния системы. Динамика реализуется с помощью механизма течения модельного времени. Чтобы создать имитационную модель надо представить реальную систему (процесс), как совокупность взаимодействующих элементов, и алгоритмически описать функционирование отдельных элементов. После этого надо описать процесс взаимодействия разных элементов между собой и с внешней средой.

Ключевым моментом в имитационном моделировании является выделение и описание состояний системы набором переменных состояний, каждая комбинация которых описывает конкретное состояние. Изменяя значения этих переменных, можно имитировать переход системы из одного состояния в другой. Таким образом, имитационное моделирование - это представление динамического поведения системы с помощью ее перехода от одного состояния к другому в соответствии с определенными операционными правилами. Эти изменения состояний могут происходить или непрерывно, или в дискретные моменты времени.

Отдельные элементы, процессы могут описываться в имитационной модели интегральными, дифференциальными и другими уравнениями и реализовываться с помощью традиционных вычислительных процедур.

Имитационное моделирование включает в себя идеи и приемы статистического моделирования на компьютере, исследования стохастических систем и случайных процессов. На входе используются переменные, которые задаются известными законами распределения. Можно реализовать вероятностное развитие ситуаций, описывать случайные процессы, проводить вероятностное оценивание характеристик модели на выходе. Для имитации параллельных событий, вводят глобальную переменную названную модельным (или системным) временем. Она обеспечивает синхронизацию всех событий в системе. Существуют два основных способа изменения модельного времени: пошаговый - с фиксированными интервалами его изменения, и событийный, при котором величина шага измеряется переменным интервалом до последующего события.

Пошаговое продвижение времени применяется, если закон изменения переменных во времени описывается интегро-дифференциальными уравнениями, которые решаются численными методами. При этом динамика модели является дискретным приближением реальных непрерывных процессов. Событийный метод применяется, когда события распределены неравномерно на часовой оси и появляются через значительные интервалы времени, когда изменяется состояние системы. Модельное время изменяется от текущего до ближайшего момента наступления последующего события. На практике этот метод получил наибольшее распространение.

Различают непрерывные, дискретные и непрерывно-дискретные виды имитационных моделей. В непрерывных имитационных моделях состояние системы меняется как непрерывная функция времени и, как правило, это изменение описывается системами дифференциальных уравнений. Соответственно продвижение модельного времени зависит от численных методов решения дифференциальных уравнений.

В дискретных моделях переменные изменяются в моменты наступления событий. В непрерывно-дискретных моделях объединяются механизмы продвижения времени, характерные для обоих видов моделей.

Для того чтобы логико-математические модели, используемые в имитационном моделировании сложной системы, могли быть реализованы на компьютере, строится моделирующий алгоритм, который описывает структуру и логику взаимодействия элементов в системе. Программная реализация моделирующего алгоритма и является имитационной моделью. Она составляется с применением средств автоматизации моделирования. Эксперимент на имитационной модели по исследованию бизнес-системы проводится для того, чтобы получить информацию о ее функционировании, которая необходима для принятия решения.  
Имитационные модели - это модели прогонного типа, у которых есть вход и выход. Если подать на ее вход определенные значения параметров, то можно получить соответствующий им результат. Для новых значений параметров или взаимосвязей имитационная программа должна быть запущена снова, т.е. имитационные модели не решаются, а прогоняются. Они не формируют свое собственное решение, как это имеет место в аналитических моделях, а служат средством для анализа поведения системы в условиях, определяемых системным аналитиком.

Особенностью моделирования стохастических систем, динамика которых зависит от случайных факторов, а входные и исходные переменные описываются как случайные величины, функции, процессы, последовательности, является то, что искомые величины при исследовании процессов определяют как средние значения относительно большого количества данных реализации процесса. Поэтому эксперимент на модели содержит несколько реализаций, прогонок и допускает оценивание по данным совокупности (выборки). При этом по закону больших чисел, чем большее число реализаций, тем получаемые оценки все более приобретают статистическую стабильность. Одной прогонки по определенным операционным правилам и конкретному набору параметров достаточно только для детерминированного моделирования. Если целями моделирования являются исследование системы при разных условиях, оценка альтернатив, поиск зависимости выхода модели от ряда параметров и, в конечном итоге, поиск некоторого оптимального варианта, то аналитик, изменяя значения параметров на входе модели, должен выполнить многочисленные машинные прогонки имитационной модели. Это необходимо для сбора, накопления и последующей обработки данных о функционировании системы. При этом возникают проблемы: как собирать эти данные, как проводить серию прогонов и как организовать целенаправленный эксперимент. Так как выходных данных, полученных в результате эксперимента, может оказаться много, то возникает проблема их обработки, более сложная, чем задача статистической оценки.

При организации и планировании имитационного эксперимента аналитик должен выбрать метод сбора информации для достижения поставленной цели исследования, и определить ее объем, стремясь при этом уменьшить расходы времени на эксплуатацию модели за счет минимизации количества имитационных прогонов. При стратегическом планировании выясняется взаимосвязь между управляемыми переменными или ищется комбинация значений управляемых переменных, дающая наилучший результат. Тактическое планирование связано с определением способов проведения намеченных имитационных прогонов. Здесь решаются задачи определения длительности прогонов, оценки точности результатов моделирования и др.

## Среда имитацмонного моделирования Anylogic

### Описание продукта

Пакет AnyLogic является отечественным профессиональным инструментом нового поколения, предназначенный для разработки и исследования имитационных моделей. Продукт был разработан компанией (XJ Technologies), в г. Санкт-Петербург. Разработка осуществлялась на основе новейших идей в области информационных технологий, теории параллельных взаимодействующих процессов и теории гибридных систем. Благодаря этим идеям сильно упрощается построение сложных имитационных моделей, есть возможность использования одного инструмента при изучении различных стилей моделирования.

Программный инструмент AnyLogic является объектно-ориентированным. Другой базовой концепцией является представление модели как набора взаимодействующих, параллельно функционирующих активностей. Активный объект в AnyLogic является объектом со своим собственным функционированием, взаимодействующий с окружением. Он может включать в себя любое количество экземпляров других активных объектов. Графическая среда поддерживает проектирование, разработку, документирование модели, выполнение компьютерных экспериментов, оптимизацию параметров относительно некоторого критерия.

При создании модели возможно применение элементов визуальной графики (диаграммы состояний, сигналы, события (таймеры), порты и т.д.), планирование событий, библиотеки активных объектов.

Благодаря простому интерфейсу AnyLogic есть возможность создавать модели даже для неопытных пользователей.

При разработке модели на AnyLogic возможно использование концепций и средств из нескольких классических областей имитационного моделирования: динамических систем, дискретно-событийного моделирования, системной динамики, агентного моделирования. Так же в AnyLogic есть возможность интегрировать различные подходы для получения наиболее полной картины взаимодействия различных процессов.

### Пользовательский интерфейс

После запуска AnyLogic открывается рабочее окно, в котором для продолжения работы необходимо создать новый проект либо открыть уже существующий.

Окно проекта обеспечивает легкую навигацию по элементам проекта, таким как пакеты, классы и т.д. Поскольку проект организован иерархически, то он отображается в виде дерева: сам проект образует верхний уровень дерева рабочего проекта, пакеты - следующий уровень, классы активных объектов и сообщений - следующий и т.д. возможно копирование, перемещение и удаление любых элементов дерева объектов, легко управляя рабочим проектом.

Главное окно проекта представлено на рисунке 1.1.



Рисунок 1.1 – Главное окно

В *AnyLogic* для всех элементов есть собственное окно свойств, в котором указаны свойства (параметры) этого элемента. Для того чтобы открыть свойства элемента, достаточно выделить этот элемент. В окне свойств содержится несколько вкладок, каждая из которых имеет элементы управления, такие как поля ввода, флажки, переключатели, кнопки и т.д., с помощью которых можно просматривать и изменять свойства элементов. Число вкладок и их внешний вид зависит от типа выбранного элемента. На рисунке 1.2 изображено окно свойств.



Рисунок 1.2 – Окно свойств

В **Окне палитры**имеются элементы, которые могут быть добавлены на диаграмму. Элементы разбиты по группам, отображающиеся на разных вкладках. Чтобы добавить объект палитры на диаграмму, нужно щелкнуть по элементу в палитре, а затем щелкнуть по диаграмме.

В активном объекте могут содержаться переменные. Они могут быть либо внутренними, либо интерфейсными. Так же могут иметься переменные, моделирующие, меняющиеся во времени величины. Переменные могут быть связаны с переменными других активных объектов. Тогда при изменении значения одной переменной будет немедленно меняться и значение переменной другого объекта. Это обеспечивает непрерывное объектов.

Запуск и отладка модели производится с помощью меню Модель и панели инструментов:



Рисунок 1.3 – Модель и панели инструментов

### Средства для моделирования

Создание модели осуществляется в графическом редакторе AnyLogic с использованием многочисленных средств поддержки, упрощающих работу. После чего модель компилируется встроенным компилятором AnyLogic и запускается на выполнение. В процессе выполнения модели можно видеть ее поведение, менять параметры, выводить результаты в различных формах и выполнять эксперименты с моделью. Для вычислений и описания логики поведения объектов в AnyLogic используется язык Java. Основными строительными блоками модели AnyLogic являются активные объекты, которые позволяют моделировать любые объекты реального мира. Классы являются хорошим средством, который позволяет структурировать сложную систему. Класс определяет шаблон, в соответствии с которым строятся отдельные экземпляры класса. Эти экземпляры определяются как объекты других активных объектов. Активный объект является экземпляром класса активного объекта. Чтобы создать модель AnyLogic, необходимо создать классы активных объектов (или использовать объекты библиотек AnyLogic) и задать их взаимосвязи. AnyLogic превращает создаваемые графически классы активных объектов в классы Java, благодаря этому мы имеем возможность воспользоваться всеми преимуществами объектно-ориентированного моделирования.

Активные объекты могут содержать вложенные объекты, причем уровень вложенности не ограничен. Это позволяет производить декомпозицию модели на любое количество уровней детализации.

Активные объекты имеют четко определенные интерфейсы взаимодействия. Они взаимодействуют со своим окружением только посредством своих интерфейсных элементов. Это помогает нам легче создавать системы со сложной структурой, а также активные объекты можно использовать повторно. После создания класса активного объекта, появляется возможность создавать любое количество объектов - экземпляров данного класса. Каждый активный объект имеет структуру (совокупность включенных в него активных объектов и их связи), а также поведение, определяемое совокупностью переменных, параметров и т.п.

Каждый экземпляр активного объекта в работающей модели имеет свое собственное поведение, имеет свои значения параметров, работает независимо от других объектов, взаимодействуя с ними и с внешней средой. При построении модели могут быть использованы средства визуальной разработки, задания численных значений параметров, аналитических записей соотношений переменных и аналитических записей условий наступления событий. Основой программирования в AnyLogic является визуализация - построение с помощью графических объектов и пиктограмм иерархий структуры и поведения активных объектов является надстройкой над языком Java - одним из самых мощных и в то же время самых простых современных объектно-ориентированных языков. Все объекты, которые определены пользователем, компилируются в конструкции языка Java, а затем происходит компиляция всей собранной программы на Java, в исполняемый код. Хотя программирование сведено к минимуму, разработчику модели необходимо иметь некоторое представление об этом языке (например, знать синтаксически правильные конструкции).

Средствами задания поведения объектов являются переменные, события и диаграммы состояний. Переменные отражают изменяющиеся характеристики объекта. События могут наступать с заданным интервалом времени и выполнять заданное действие. Диаграммы состояний позволяют увидеть поведение объекта во времени под воздействием событий или условий, они состоят из графического изображения состояний и переходов между ними.

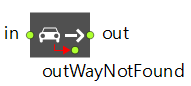
Для того что бы показать поведение сложной системы, в Anylogic есть средства, позволяющие представить моделируемую систему в живой форме динамической анимации. Визуализация процесса функционирования системы позволяет проверить адекватность модели, выявить ошибки при задании логики. Благодаря анимации можно легко создавать виртуальный мир, который управляется динамическими параметрами по законам, определенным пользователем с помощью уравнений и логики моделируемых объектов. Графические элементы, добавленные на анимацию, называются динамическими, так как все их параметры: видимость, цвет и т.п. - можно сделать зависимыми от переменных и параметров модели, которые в свою очередь меняются со временем при выполнении модели.

### Элементы построения модели

Для выполнения данной курсовой работы нам понадобятся такие элементы библиотек, как:

 Car Source. Создает автомобили и пытается поместить их в указанное место дорожной сети. Автомобиль можно поместить на указанную дорогу или парковку (это задается параметром *Появляется*).

Если в качестве места появления автомобиля указана дорога, необходимо дополнительно указать направление дороги. Автомобили будут появляться в начале указанной дороги, после чего будут начинать движение в указанном направлении. По умолчанию, выбран параметр *Случайная полоса*, поэтому автомобили будут появляться на случайных полосах дороги (если на указанной дороге больше одной полосы). Можно явно указать номер полосы, на которой будут появляться автомобили. Для этого необходимо снять флажок с опции *Случайная**полоса* и указать номер в параметре *Номер**полосы*;

CarMoveTo. Блок, который управляет движением автомобиля. Автомобиль может ехать, только когда он находится в блоке CarMoveTo. Автомобиль пытается рассчитать путь от своего текущего места до указанного места назначения, когда поступает в блок CarMoveTo.

В качестве цели движения могут выступать: *дорога*, *парковка*, *автобусная остановка* или *стоп-линия*. Указанное место назначения должно находиться в той же дорожной сети, что и автомобиль. Если от текущего местоположения автомобиля к указанному месту нет пути, автомобиль покидает блок через порт *outWayNotFound*;

http://127.0.0.1:59329/help/topic/com.anylogic.help/html/_RoadTraffic/reference/images/carDispose.png Car Dispose. Удаляет машины из модели;

 SelectOutput. Объект направляет входящие заявки в один из двух выходных портов в зависимости от выполнения заданного условия;

Delay. Задерживает заявки на заданный период времени.

## Вывод по главе

В данной главе раскрыты теоретические аспекты транспортной задачи, которая является одной из самых распространенных задача линейного программирования специального вида. Рассматривается постановка транспортной задачи, построение математической модели, раскрывается сущность методов нахождения исходного опорного и оптимального планов, дается подробное изложение этапов метода потенциалов.

Для того чтобы определить исходный опорный план ТЗ должна быть разрешима. Для этого необходимо и достаточно, чтобы выполнялось условие: суммарные объемы поставок должны быть равны суммарным объемам потребления. В этом случае задача является закрытой. При невыполнении условия разрешимости задача будет открытого типа.

Исходный опорный план TЗ можно находить с помощью различных методов. В данном исследовании были рассмотрены два метода: «северо-западного» угла и минимального элемента.

Специфика ограничений TЗ позволяет применять для ее решения методы значительно менее громоздкие, чем симплексный метод. Одним из методов решения TЗ, который учитывает специфику ее ограничений, является метод потенциалов. По существу, его можно рассматривать, как результат реализации симплексного метода в условиях транспортной задачи.

# Имитационная модель общественного транспорта в AnyLogic

## Постановка задачи

В данной задаче мы смоделируем движение маршрутного транспортного средства № 10 по г. Тирасполь. В качестве маршрутов будут взяты, реальный маршрут № 10 и вымышленный. В результате моделирования мы должны получить набор данных о прибыли и затратах на перевозки пассажиров и сравнить их между собой для оценки оптимальности.

Для решения данной задачи будет использоваться инструмент имитационного моделирования *Anylogic*.

## Построение модели общественного транспорта

### Создание модели и разметка рабочего пространства

Сразу после запуска *Anylogic* перед нами появится начальная страница, где нам предлагают создать новую модель, открыть примеры, ознакомиться с учебными пособиями и т.д. Представлено на рисунке 2.1.

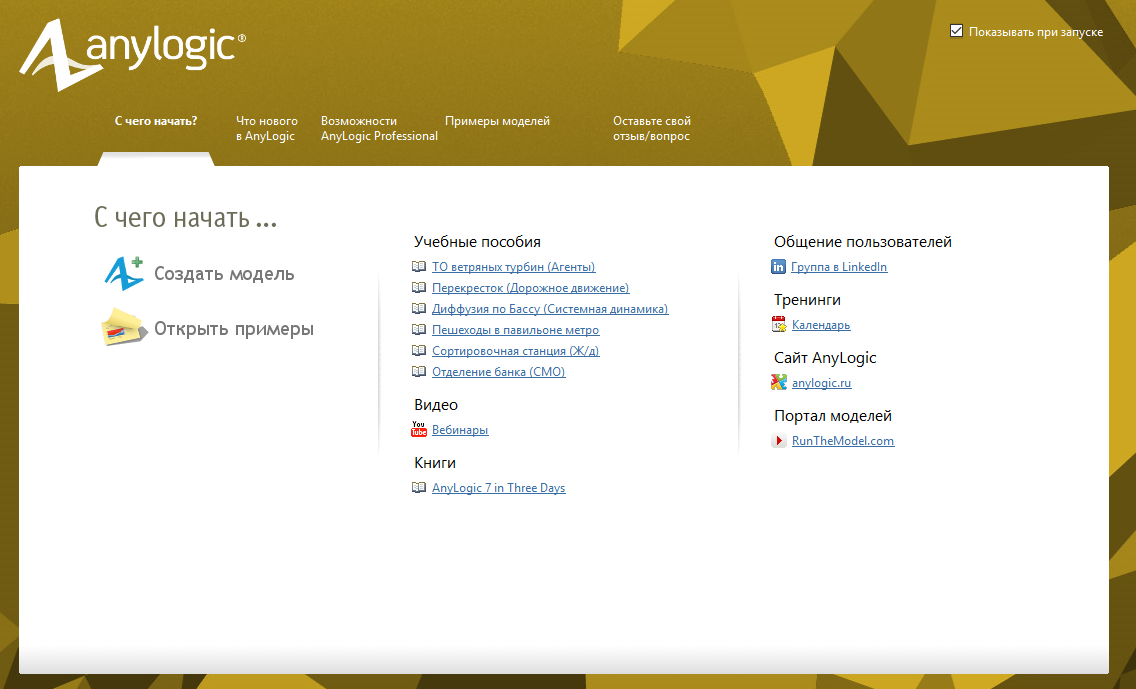


Рисунок 2.1 – Начальная страница *AnyLogic*

Выбираем пункт «создать модель», после чего откроется окно с настройками новой модели. Представлено на рисунке 2.2.

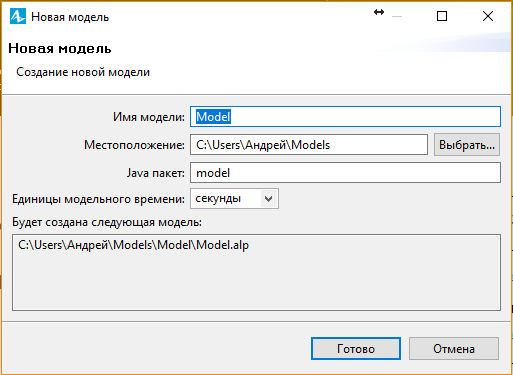


Рисунок 2.2 – Окно создания новой модели

Указываем имя модели, ее расположение на диске, *Java* пакет оставляем без изменений и единицы модального времени выбираем секунды, далее жмем кнопку «готово» и приступаем к работе.

В окне «палитра» заходим в раздел «презентация», выбираем объект «изображение» и перетаскиваем в рабочую область, откроется окно выбора изображения, выбираем заранее приготовленную карту города с размеченными маршрутами. Прежде чем приступить к дальнейшим действиям нам придется настроить масштаб, что бы он соответствовал нашей карте. Представлено на рисунке 2.3.

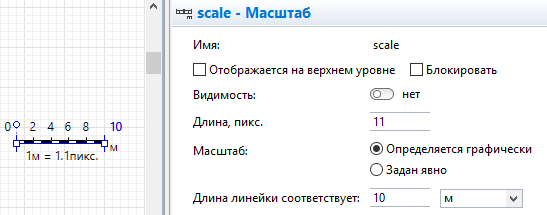


Рисунок 2.3 – Окно свойств объекта *scale*

Нарисуем дорожную сеть, для этого в окне «палитра» заходим в раздел «библиотека дорожного движения», представлено на рисунке 2.4, и элементами разметки пространства рисуем дорожную сеть с автобусными остановками.

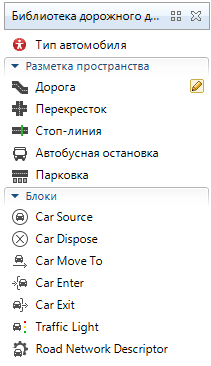


Рисунок 2.4 – Библиотека дорожного движения

После того как вся дорога по заданному маршрута была размечена и автобусные остановки расставлены, у нас должна получиться дорожная сеть как показано на рисунке 2.5.



Рисунок 2.5 – Дорожная сеть

### Модель поведения автобусов

Когда дорога с автобусными остановками уже построена, нам необходимо смоделировать логику движения по остановкам с высадкой и посадкой пассажиров. Добавим элемент *CarSource* из «библиотеки дорожного» движения и создадим новый тип агента для автобусов, агента назовем *Bus*. Для этого откроем свойства *CarSource* и во вкладке «автомобиль» нажмем «создать другой тип». Представлено на рисунке 2.6.

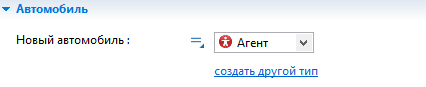


Рисунок 2.6 – Окно свойств *CarSource*

Откроется окно, где мы будем создавать нового агента «с нуля». Показано на ресунке 2.7.

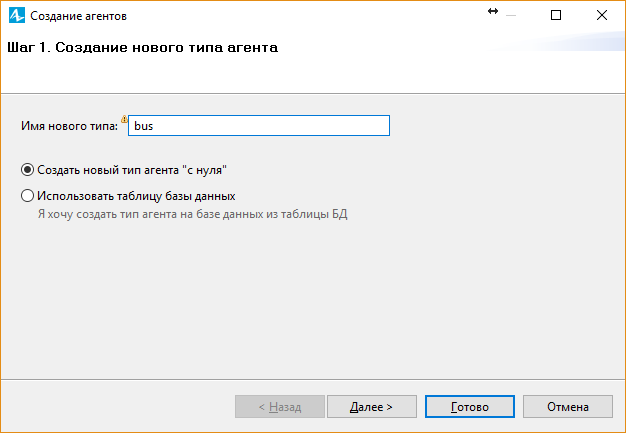


Рисунок 2.7 – Окно создания нового типа агента

Выберем внешний вид нашего автобуса. Как видно на рисунке 2.8.

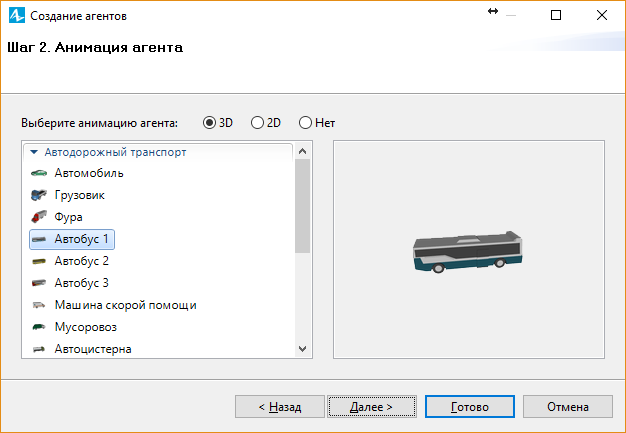


Рисунок 2.8 – Окно выбора анимации агента

На следующем шаге добавим некоторые параметры нашего автобуса, после чего нажимаем Готово. Представлено на рисунке 2.9.

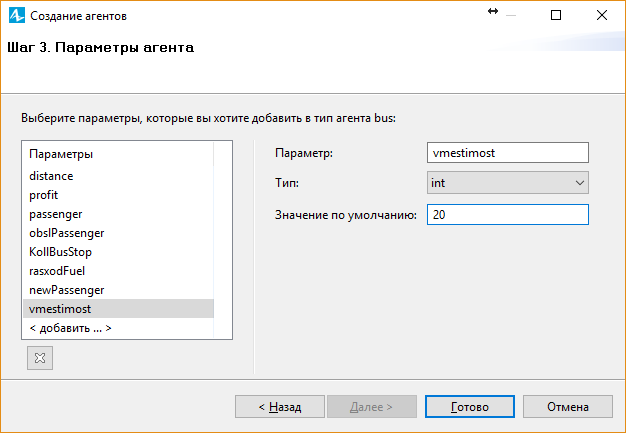


Рисунок 2.9 – Окно настройки параметров агента

Для того чтобы в дальнейшем было понятно какие параметры за что отвечают, распишем их значения:

**distance** – тип *double*, количество метров которые проехал автобус;

**profit** – тип *double*, прибыль за каждого пассажира;

**passenger** – тип *int*, количество пассажиров находящихся в автобусе;

**obslPassenger** – тип *int*, количество пассажиров которых обслужил автобус;

**Fuel** – тип double, количество потраченного топлива;

**KollBusStop** – тип *int*, количество пройдённых остановок;

**rasxodFuel** – тип *double*, расход топлива на 100 километров;

**newPassenger** – тип *int*, хранит в себе количество пассажиров которые зашли в автобус на очередной остановке;

**vmestimost** – тип *int*, вместимость автобуса;

Теперь в свойствах *CarSource* во вкладке «автомобиль» мы можем выбрать созданного нами агента *Bus*.

Для того чтобы наш автобус двигался по маршруту и останавливался на остановках, нам потребуется описать модель поведения автобуса, для этого мы создадим нового агента, назовем его «маршрут».

Из палитры «агент» перетащим в рабочую область элемент «агент» и откроется окно для создания нового типа агента. В этом окне выберем «единственный агент» как показано на рисунке 2.10.

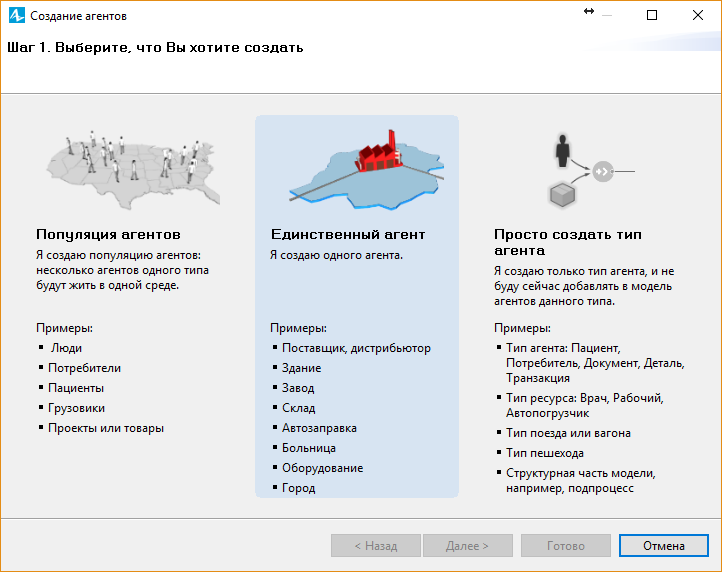


Рисунок 2.10 – Окно создания нового типа агента

Далее все тоже самое, как и при создании нового типа автомобиля, только анимации у этого агента не будет.

Модель поведения автобусов будет выглядеть следующим образом.Показано на рисунке 2.11.

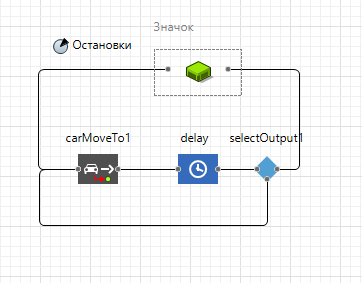


Рисунок 2.11 – Модель поведения автобусов

В свойствах зеленой картинки укажем, что это у нас будет значек агента и добавим ему входной и выходной «порт» из окна «палитра» библиотеки «агент».

У этого агента будет всего лишь один параметр **Остановки** который из себя представляет массив из всех остановок, тип данного параметра будет *BusStop[]* и тип управления будет «одноразмерный массив». Показано на рисунке 2.12.

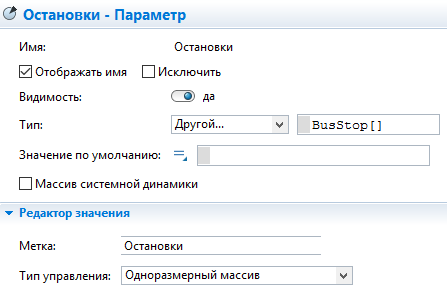


Рисунок 2.12 – Окно своств параметра

После чего мы сможем добавить все остановки в этот массив. Изображено на рисунке 2.13.

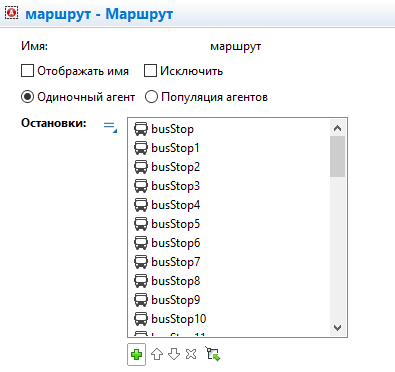


Рисунок 2.13 – Окно свойст агента

После того как во входной порт агента Маршрут поступает автобус, он попадает в блок *carMoveTo1* в котором и происходят основные действия этого агента. Рассмотрим по порядку что же там происходит. Откроем свойства этого блока как показано на рисунке 2.14.

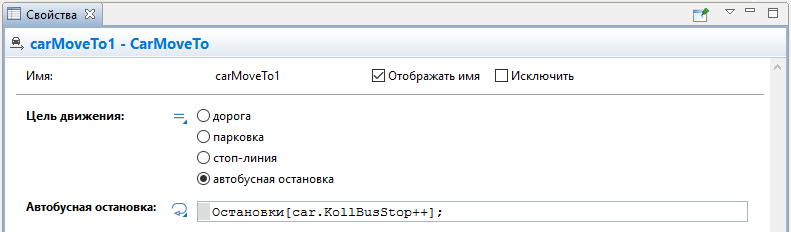


Рисунок 2.14 – Окно свойст элемента *carMoveTo1*

Как мы видим этот блок перемещает автобус на остановку вперед, т.е. он выбирает остановку из массива по индексу *KollBusStop*и увеличивает его на единицу. При выходе из данного блока выполняется код на языке программирования *Java* можно увидеть на рисунке 2.15.

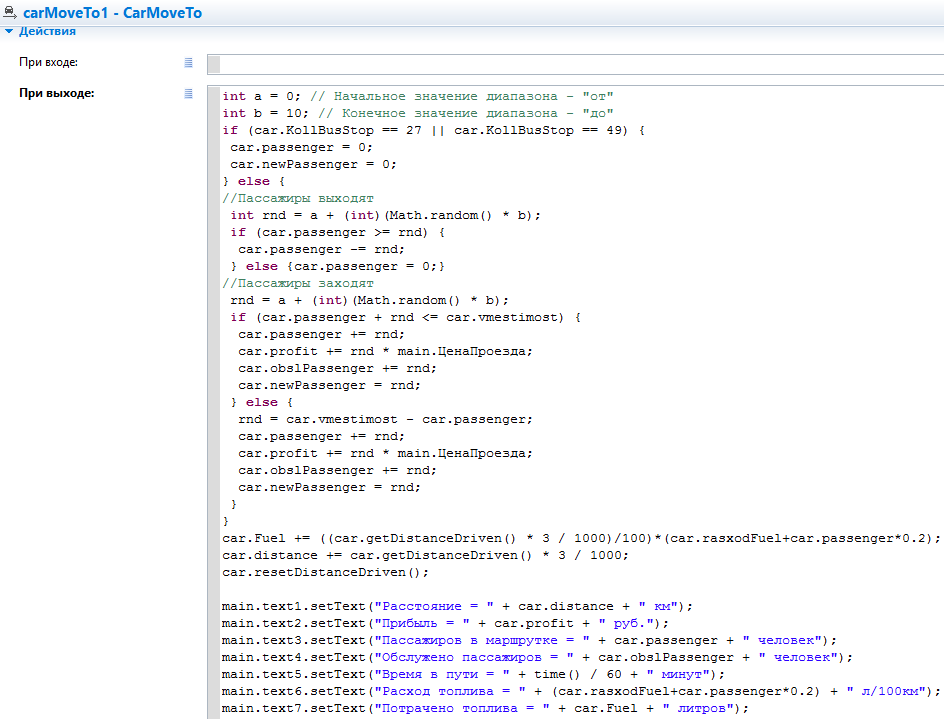


Рисунок 2.15 – Окно свойст элемента *carMoveTo1,* вкладка «действия»

Далее автобус попадает в блок *Deley*который задерживает его на время, в секундах, равное количеству новых пассажиров, умноженных на 10, т.е. в среднем по 10 секунд на каждого пассажира. Условие задержки указывается в свойствах элемента *deley*. Представлено на рисунке 2.16.

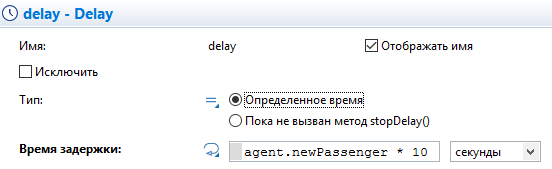


Рисунок 2.16 – Окно свойст элемента *deley*

После задержки автобус попадает в блок *selectOutput1* где проверяется условие, как показано на рисунке 2.17, если количество пройденных остановок сровнялось с общим количеством всех остановок, то автобус выходит из данного агента, доезжает до конца дороги и уничтожается, иначе все действия повторяются пока он не проедет все остановки.

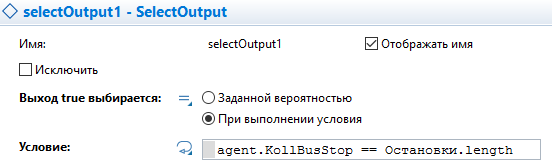


Рисунок 2.17 – Окно свойст элемента *selectOutput1*

Окончательная и основная модель поведения маршрутного такси показана на рисунке 2.18.

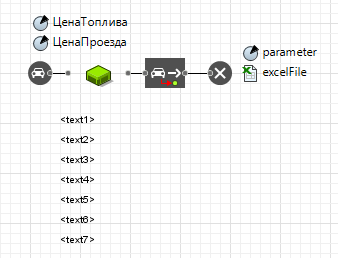


Рисунок 2.18 – Окно свойст элемента *selectOutput1*

## Полученные результаты

Заключение

Выводы по *результатам* работы. Проще всего заключение написать, обобщив выводы по каждой из глав.

Литература

1. Страуструп Б. Язык программирования C++: Пер. с англ. — М.: Бином, 2004. — 1054 с.
2. Гришкун В. В., Григорьев С. Г. Рекомендации по эффективному формированию информационных ресурсов образовательных порталов // Интернет-порталы: содержание и технологии: Сб. научных статей, вып. 3. — М.: Просвещение, 2006. — С. 8—13.
3. Информатика. Базовый курс / Симонович С. В., Евсеев Г. А., Мураховский В. И., Бобровский С. И.; под ред. С. В. Симоновича. — СПб.; Харьков; Минск: Питер, 2000. — 640 с.
4. Таров Д. А. Формирование адекватной самооценки учебной деятельности у подростков (на примере сельской школы): Автореф. дисс. … канд. пед. наук — Елец: ЕГУ им. И. А. Бунина, 2003. — 20 с.
5. О преподавании учебного предмета «Информатика и ИКТ» и информационных технологий в рамках других предметов в условиях введения федерального компонента государственного стандарта общего образования. // http://www.ed.gov.ru/d/ob-edu/noc/rub/standart/mp/06.doc
6. Могилёв А. В. О понятии «Информационное моделирование» // Информатика и образование. — 1997. — № 8. — С. 3—8.
7. Об электронной цифровой подписи: Федеральный закон РФ от 10.01.2002 г. № 1-ФЗ // Российская газета. — 2002. — 12 янв.