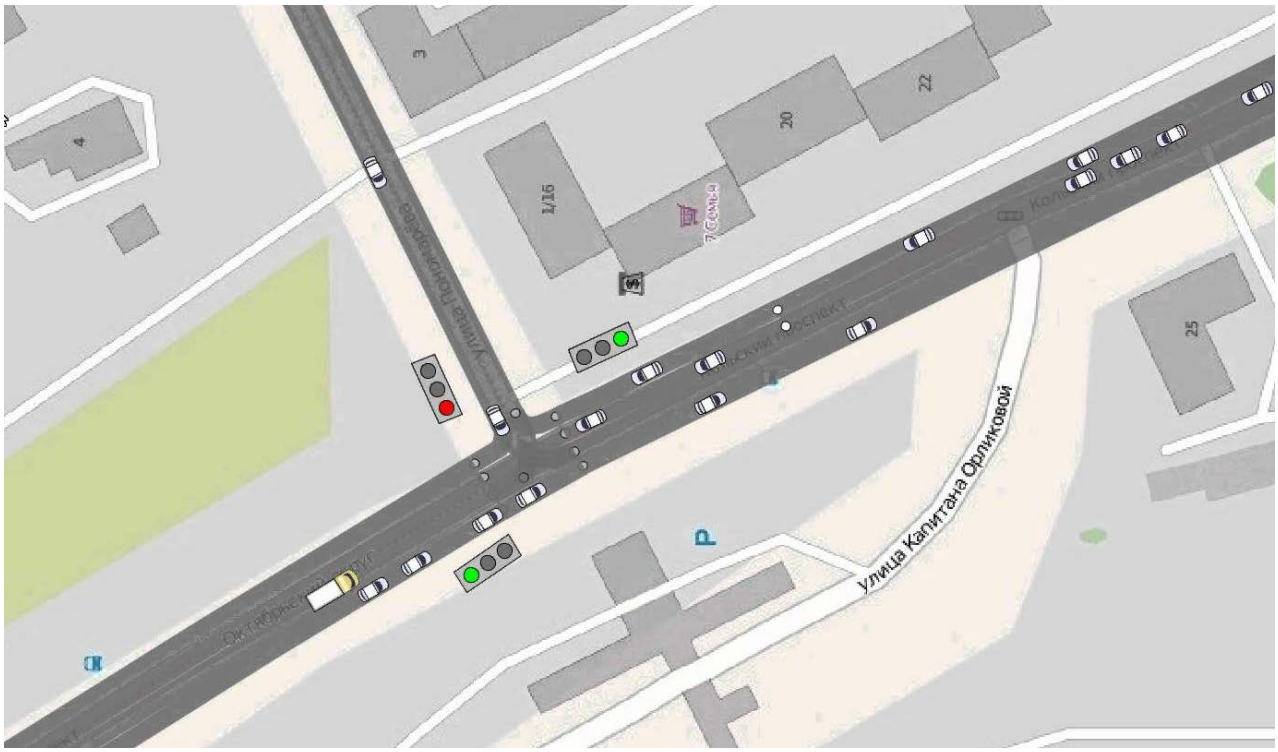
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Курганский государственный университет»

Кафедра «Организация и безопасность движения»

МОДЕЛИРОВАНИЕ ДОРОЖНОГО ДВИЖЕНИЯ

Методические указания
к выполнению лабораторных работ
для студентов всех форм обучения
направления подготовки 190700.62



Курган 2015

Кафедра: «Организация и безопасность движения»
Дисциплина: «Информационные технологии на транспорте»
(направление 190700.62).

Составил: ст. преподаватель Н.С. Безотеческих.

Утверждены на заседании кафедры «29» ноября 2013 г.
Рекомендованы методическим советом университета «31» декабря 2013 г.

Содержание

Введение

Дисциплина «Моделирование дорожного движения» дат знания, позволяющие студенту решать задачи, связанные с организацией и управлением производственной деятельностью организаций, занимающихся организацией дорожного движения.

Изучение дисциплины «Моделирование дорожного движения» играет важную роль в подготовке специалиста. В рамках изучения дисциплины студенты знакомятся с основами моделирования сложных систем, в том числе стохастической системы ВАДС.

Приобретаемые студентами знания применяются при изучении дисциплин «Информационные технологии», Организация транспортных услуг и безопасность транспортного процесса», «организация и безопасность движения».

Целью изучения курса «Моделирования дорожного движения» является формирование системного мышления в отношении моделирования процессов дорожного движения, подготовка студентов к решению сложных задач, требующих использования методологии системного анализа транспортных процессов и систем.

Задачами изучения дисциплины являются:

- освоение теоретических принципов моделирования дорожного движения;
- умение анализировать основные модели дорожного движения;
- овладение методами программного моделирования процессов дорожного движения.

Сборник описаний лабораторных работ содержит наименование лабораторных работ согласно плану учебной программы. Для каждой лабораторной работы изложены цель и задачи работы, порядок выполнения и форма отчетности. В конце лабораторного занятия имеются контрольные вопросы для закрепления полученных знаний и навыков, а также приведен библиографический список рекомендуемой литературы.

После успешного выполнения заданий студент предоставляет результат работы, оформленный в виде отчета, в электронном виде.

После проверки результатов преподаватель допускает студента к защите, в ходе которой студент отвечает на контрольные вопросы.

1 ЛАБОРАТОРНАЯ РАБОТА № 1
НЕПРЕРЫВНЫЕ МОДЕЛИ АВТОМОБИЛЬНОГО И ПЕШЕХОДНОГО ДВИЖЕНИЯ. СТЕЙТЧАРТЫ. МОДЕЛЬ ПЕШЕХОДНОГО ПЕРЕХОДА

Цели:

- построение стейтчартов;
- действия при входе и выходе из состояния, иерархические состояния;
- переход по исчерпании таймаута;
- переход при получении сообщения;

Обеспеченность:

- компьютер с установленной программой AnyLogic версии 6.

1.1 ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Построим модель регулируемого пешеходного перехода со светофором, разрешающим или запрещающим движение транспорта.

Светофор, регулирующий движение автомобилей на пешеходном переходе, может находиться в следующих состояниях: движение транспорта разрешено (зеленый), приготовиться к запрещающему сигналу (мигающий зеленый), приготовиться к остановке (желтый), движение запрещено (красный) и приготовиться к движению (красный и желтый).

Светофор работает в автоматическом режиме. В каждом состоянии светофор находится определенный постоянный период времени.

1.2 ПОСТРОЕНИЕ МОДЕЛИ

Создайте новый проект под названием *Svetofor* и назовите класс корневого активного объекта *Model*.

Наша модель будет иметь только один активный объект, представляющий светофор, поэтому корневой объект *Model* будет единственным активным объектом нашей модели. На диаграмму класса активного объекта *Model* поместите Начало диаграммы состояний из панели Диаграмма состояний и назовите ее Для автомобилей, заметьте, что AnyLogic может работать с элементами, набранными кириллицей. Перетащите мышью элемент Состояние под стрелочку начала диаграммы.

Имя состояния, как и все другие его параметры, можно редактировать в окне его свойств. Для того чтобы построить стейтчарт, следует использовать элементы из палитры Диаграмма состояний.

Для любого выделенного объекта внизу появляется панель его свойств, в котором можно изменить параметры и, в частности, имя объекта, если это необходимо.

Структурные ошибки при рисовании стейтчарта – повисшие переходы, дублированные указатели начального состояния и т.п. – выделяются в панели Проекты значком *X* красного цвета и записью в панели *Ошибки*. Выделенные переходы должны иметь на концах зеленые точки, если точки белые, это значит, что переход не соединен с состоянием – висит.

В соответствии с алгоритмом работы светофора помимо начального состояния в модель нужно ввести дополнительные состояния (рисунок 1.1). Начальное состояние назовите «*движение*» – движение автомобилям разрешено (зеленый свет), затем светофор переходит в состояние *«внимание»* – внимание (мигающий зеленый), *«медленно»* – приготовиться к остановке (желтый свет), остановка транспорта «*stop*» – запрет движения (красный свет) и *«приготовиться»* – приготовиться к движению (красный и желтый свет горят одновременно).

Состояние *«внимание»* представим гиперсостоянием с парой переключающихся элементарных состояний: в одном из них зеленый горит (*состояние А*), в другом – нет (*состояние В*). Постройте эти состояния и соедините их переходами, как показано на рисунке 1.1.

Рисунок 1.1 – Схема работы светофора

Зададим условия срабатывания переходов. Переходы в нашем автоматическом светофоре выполняются по таймауту, т. е. по истечении интервала времени, который прошел с момента прихода системы в данное состояние.

1) В состоянии *«движение»* светофор находится 10 секунд,

2) затем 7 секунд зеленый сигнал мигает,

3) в состоянии *«медленно»* 4 секунды горит желтый,

4) в течение 10 секунд движение запрещено,

5) 4 секунды светофор находится в состоянии *«приготовиться»*.

В нашей модели единица модельного времени соответствует 1 секунде реального времени.

Для задания условий срабатывания переходов, выделите переход «t1», и в поле *Происходит* оставьте без изменения вариант *По таймауту*, а в поле *По таймауту* введите 25 (рисунок 1.2).

Аналогично задайте условия срабатывания других переходов.

Между состояниями *«A»* и *«B»* переходы должны срабатывать через 1 секунду.

Запустите модель. Активное в данный момент состояние подсвечивается красным.

Переход, ожидающий истечения таймаута подсвечивается синим.

Проведите эксперименты с моделью при различных масштабах времени.

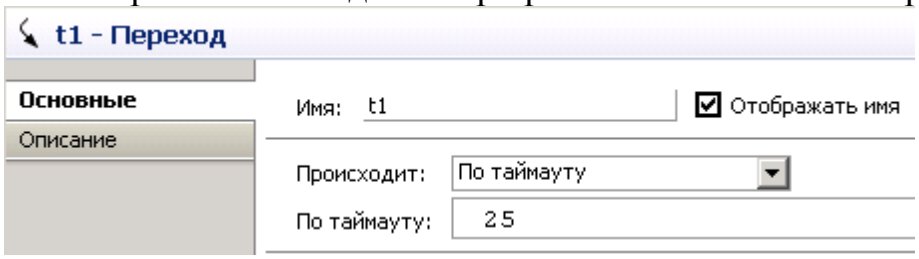


Рисунок 1.2 – Параметры перехода t1

В каждом состоянии светофора должен гореть определенный сигнал: в состоянии «движение» должен гореть зеленый, в состоянии «приготовиться» должны гореть красный и желтый одновременно и т.п. Перейдите на диаграмму класса активного объекта *Model*.

Создайте три параметра логического типа: *красный*, *желтый* и *зеленый*, которые будут принимать истинное значение тогда, когда у светофора горит соответствующий сигнал: красный, желтый или зеленый (рисунок 1.3). Начальные значения этих булевых параметров можно не задавать: по умолчанию они будут равны *«false»*.

Мы создали стейтчарт именно для управления значениями этих параметров, каждое состояние отвечает за зажигание своего света или их комбинации. Например, в состоянии *«медленно»* должен гореть желтый, в состоянии «*stop*» должен загореться красный свет, а в состоянии *«приготовиться»* должны гореть красный и желтый одновременно.

Запрограммируем эти действия.

1 В свойствах состояния *«движение»* в поле *Действие при входе* запишите *зеленый=true*, а в поле *Действие при выходе* запишите *зеленый=false* (рисунок 1.4).

2 То же самое запишите для состояния *«B»* гиперсостояния «*внимание*», а у состояния «*A*» эти поля нужно оставить пустыми – когда светофор находится в этом состоянии, он не горит.

3 Аналогично, в состоянии *«медленно»* нужно включить желтый сигнал, т.е. при входе в это состояние установить параметр желтый в *true*, а при выходе из этого состояния установить его в *false*.

4 Для состояния «*stop*» аналогично опишите состояния параметра красный, а для состояния приготовиться оба параметра красный и желтый нужно установить в *true* при входе, и в *false* при выходе.

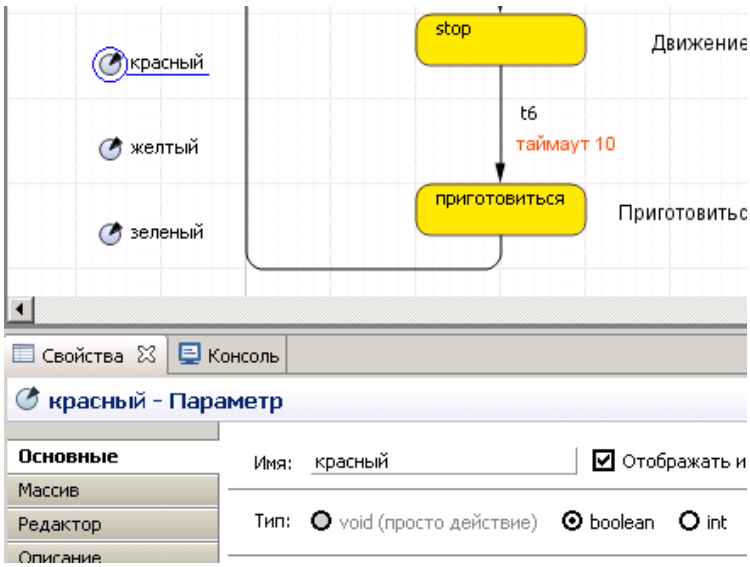


Рисунок 1.3 – Параметр красный

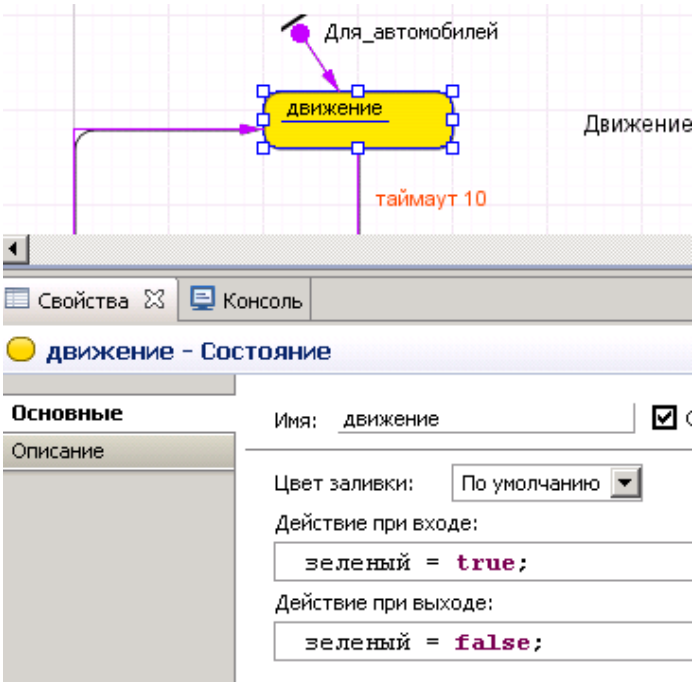


Рисунок 1.4 – Свойства состояния движение

Запустите модель на выполнение. Вы увидите, что параметры зеленый, желтый и красный будут переключаться между значениями истина и ложь в соответствии с алгоритмом переключения светофора.

1.3 ПРЕЗЕНТАЦИЯ МОДЕЛИ

Презентация модели рисуется в той же диаграмме (в графическом редакторе), в которой задается и диаграмма моделируемого процесса. Графические объекты цвета сигналов светофора в презентации имеют динамические параметры, все остальные – статические.

Светофор строится из трех овалов, повернутых на 45° (поле *Поворот* вкладки *Дополнительные* окна свойств овала).

Установим динамическое значение цвета верхнего сигнала светофора: если переменная *«красный»* истинна, то цвет должен быть *red* (красный), в противном случае его цвет нужно установить *gray* (серый). Это записывается следующим условным выражением на языке Java: *красный? red: gray*

Цвет среднего и нижнего овалов, следует установить в поле их динамических значений соответственно так: *желтый? yellow: gray*, *зеленый? green: gray*, *red, yellow, green* и *gray* – предопределенные константы, обозначающие стандартные цвета.

Запустите модель и проверьте ее работу.

1.4 СРАБАТЫВАНИЕ ПЕРЕХОДА ПО СИГНАЛУ

Добавим к нашей модели второй светофор, для пешеходов. Он будет иметь два сигнала, *зеленый* и *красный*, и три состояния: «*Идите*» (зеленый), «*Внимание*» (мигающий зеленый) и «*Стойте*» (красный). Добавим в модель два булевских параметра *Стойте* и *Идите*, их значениями будет управлять второй стейтчарт – для пешеходов. Создадим этот стейтчарт на той же диаграмме класса *Model*, назвав его *Для_пешеходов* (рисунок 1.5).

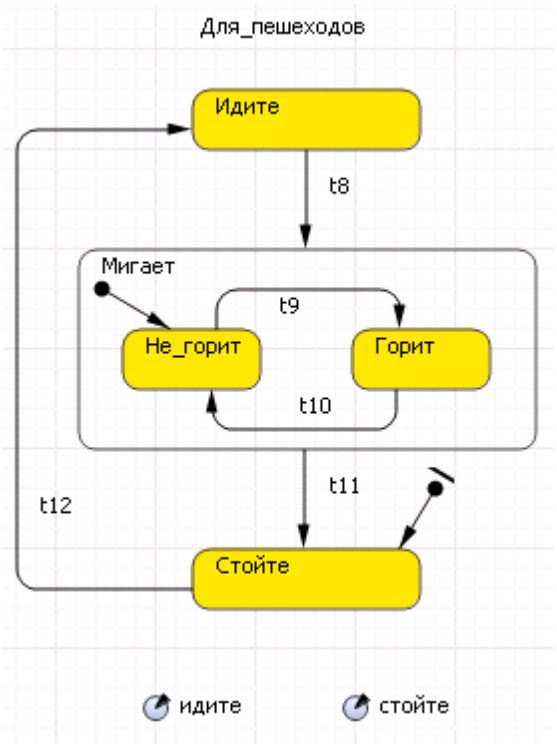


Рисунок 1.5 – Стейтчарт Для_пешеходов

Поскольку управление светофором пешеходов похоже на управление светофором автомобилей, новый стейтчарт можно построить копированием и изменением уже построенного стейтчарта *Для_автомобилей*. Выделите 3 состояния стейтчарта *Для_автомобилей* и вставьте их в другое место диаграммы. Эти элементы скопируются, и им автоматически будут даны уникальные имена, чтобы не было конфликта имен.

Переименуйте состояния стейтчарта *Для_пешеходов*, дорисуйте недостающий переход *t12* и перенесите начало диаграммы на состояние Стойте, как показано на рисунок 1.5.

Измените параметры в полях *«Действие при входе»* и *«Действие при выходе»* в свойствах состояний стейтчарта *Для_пешеходов*. Теперь наш стейтчарт должен управлять параметрами *«Идите»* и *«Стойте»*, которые, в свою очередь, будут управлять зажиганием света именно пешеходного светофора.

Настроим условия срабатывания переходов стейтчартов между состояниями. Для обеспечения безопасной работы пешеходного перехода необходимо синхронизировать срабатывания стейтчартов так, чтобы всегда, когда светофор пешеходов находится в состояниях «*Идите*» или «*Мигает*», светофор автомобилей обязательно находился бы в состоянии «*stop*». Для этого можно подобрать подходящие таймауты срабатывания переходов стейтчарта, но при каждом изменении модели, придется эти таймауты подбирать снова и снова. Более разумно синхронизировать стейтчарты, посылая специальные разрешающие сигналы из одного стейтчарта в другой. В нашей модели стейтчарты будут обмениваться следующими сигналами: *АВТОМОБИЛИ* и *ПЕШЕХОДЫ*. В стейтчарте *Для_пешеходов* переход *t12* будет срабатывать когда получен сигнал *ПЕШЕХОДЫ*, который будет генерироваться в стейтчарте *Для_автомобилей* при переходе *t5* в состояние *stop*. В свою очередь, в стейтчарте *Для_автомобилей* переход *t6* будет срабатывать когда получен сигнал *АВТОМОБИЛИ*, который генерируется в стейтчарте *Для_пешеходов* при переходе *t11* в состояние «*Стойте*» рисунок (1.6).

В AnyLogic есть несколько способов передачи сообщения в диаграмму состояний. В нашей модели мы будем использовать метод *fireEvent*( ), который должен вызываться в том стейтчарте, которому предназначено сообщение. То есть, если из некоего объекта мы хотим послать сообщение стейтчарту, то нужно в этом объекте написать команду *стейтчарт.fireEvent*(сообщение). Поэтому, в поле *Действие* перехода *t5* стейтчарта *Для_автомобилей* нужно вставить команду Для_пешеходов.fireEvent(ПЕШЕХОДЫ), в такое же поле перехода *t11* стейтчарта *Для_пешеходов* вставьте команду: Для_автомобилей.fireEvent(АВТОМОБИЛИ).

Таким образом, каждый из светофоров будет информировать другого о своем переходе в состояние запрещения движения, как пешеходов, так и автомобилей. Для срабатывания перехода стейтчарта при получении нужного сообщения, в стейтчарте *Для_пешеходов* в поле Происходит окна свойств перехода *t12* выберите вариант При получении сообщения, укажите тип сообщения String, а в поле Осуществлять переход выберите Если сообщение равно и введите «*ПЕШЕХОДЫ*» (рисунок 1.6).

Аналогично, для срабатывания перехода автомобильного стейтчарта по сигналу от пешеходного стейтчарта в стейтчарте *Для_автомобилей.* В поле *Происходит* окна свойств перехода *t6* выберите вариант П*ри получении сообщения*, укажите тип сообщения String, а в поле *Осуществлять переход* выберите Если сообщение равно и введите *АВТОМОБИЛИ*.

Остальные переходы этих стейтчартов будут срабатывать по таймаутам, как и прежде. Проверьте по рисунку 1.6 установленные параметры переходов стейтчартов. Запустите модель на выполнение.

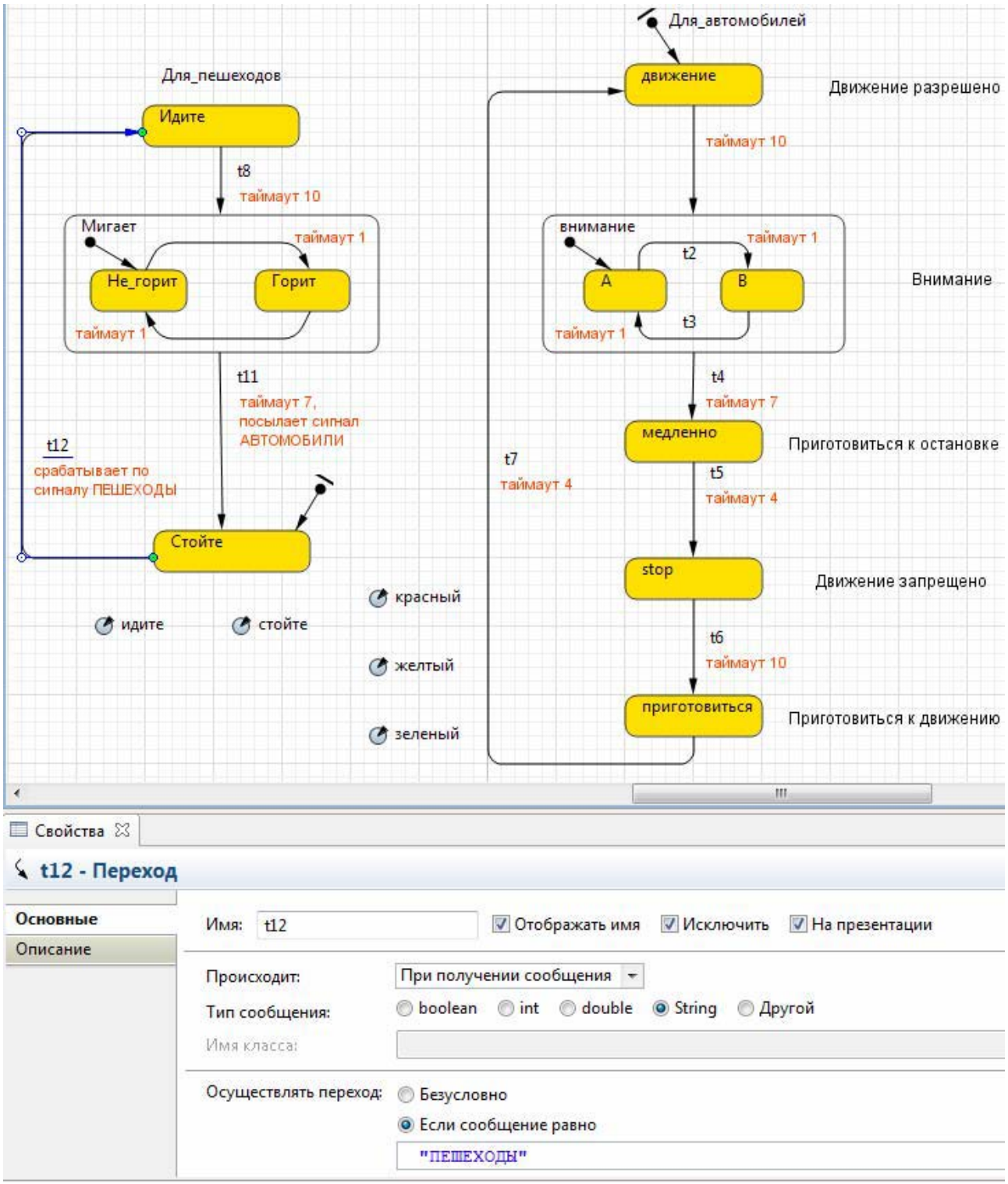


Рисунок 1.6 – Стейтчарт

На презентации модели, в дополнение к светофору для автомобилей, следует нарисовать светофор для пешеходов с двумя сигналами – красной надписью *СТОЙТЕ* и зеленой *ИДИТЕ*. Динамикой цвета этих надписей будут управ-

лять логические параметры *стоите* и *идите*, которые нужно создать на диаграмме по аналогии с параметрами *красный*, *желтый* и *зеленый*.

1.4 СРАБАТЫВАНИЕ ПЕРЕХОДА ПО УСЛОВИЮ

Если пешеходы переходят улицу достаточно редко, то автоматическое переключение светофора будет неоправданно часто прерывать движение автомобилей. В этом случае разумно установить специальную кнопку, при нажатии которой пешеходом светофор автомобилей остановит на некоторое время движение автомашин. Для этого перетащите мышью элемент *Кнопка* с палитры *Элементы управления* на диаграмму класса активного объекта *Model* рядом с пешеходным светофором. Назовите кнопку и ее метку – *переход*. Определите действие при нажатии на кнопку так, чтобы устанавливался в истину булевый параметр *ожидание*. Для этого нужно записать в поле *Действие* окна свойств этой кнопки команду *ожидание=true*. Логический параметр *ожидание* нужно ввести на диаграмме активного объекта *Model* с начальным значением *false*. Этот параметр будет определять, собирается ли пешеход перейти дорогу.

Значение этого параметра будем переводить в *false* каждый раз, как только пешеходный светофор перейдет в состояние «*Мигает*». Для этого нужно записать в поле *Действие при входе окна* свойств состояния «*Мигает*» стейтчарта *Для_пешеходов* команду: *ожидание=false*.

Таким образом, при нажатии кнопки переход, стейтчарт автомобилей «узнает» об ожидающих на переходе пешеходах и переключится из состояния «*Движение*» в состояние «*Внимание*» и затем в состояние «*Запрещение*» движения автомобилей. Если некий злоумышленник будет постоянно нажимать кнопку «переход», это может полностью парализовать движение автомобилей. Для того чтобы этого не произошло, в стейтчарте *Для_автомобилей* сделайте иерархическим состояние *Движение*, с двумя простыми состояниями «*непрерывное*» и «*обычное*», как показано на рисунке 1.7.

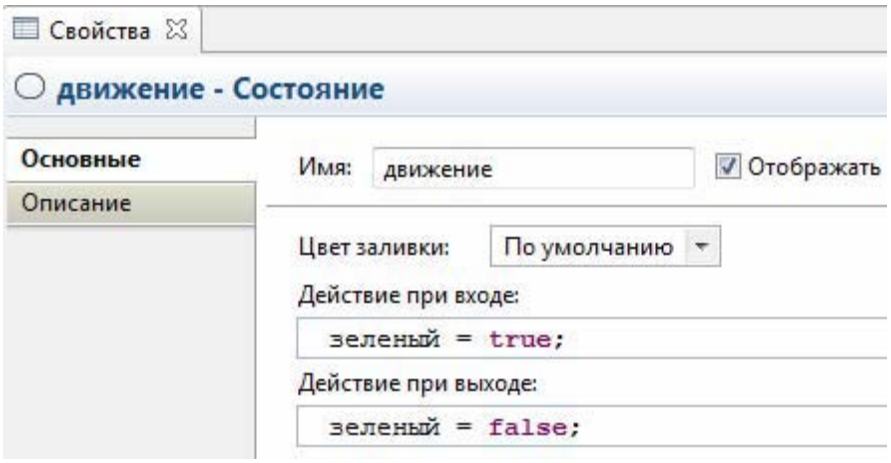


Рисунок 1.7 – Стейтчарт Для_автомобилей

В состоянии «*непрерывное*» автомобили будут двигаться до истечения таймаута 10 секунд, невзирая на состояние параметра *ожидание*, т.е. автомобилям будет гарантированно предоставлено некоторое время для движения, даже

если кнопка переход будет всегда нажата. После истечения таймаута 10 секунд светофор перейдет в «*обычное*» состояние движения, которое может прерываться. Переход *T* из состояния «*обычное*» сработает, когда будет нажата кнопка *переход*, т.е. параметр *ожидание* будет истинен.

Если кнопка *переход* не нажата (т.е. параметр ожидание имеет значение *false*), автомобили будут продолжать движение до нажатия этой кнопки. Если нажать кнопку один или более раз, то параметр ожидание станет истинным и автомобильный светофор из состояния «*обычное*» перейдет в состояние «*внимание*» и затем остановит движение автомобилей.

Проверьте работу модели и продемонстрируйте ее преподавателю.

1.5 КОНТРОЛЬНЫЕ ЗАДАНИЯ

1 Измените модель светофора с кнопкой таким образом, чтобы моделировалось автоматическое нажатие кнопки *ПЕРЕХОД* в случайные моменты времени в интервале от 5 секунд до 1 минуты.

2 Измените модель автоматического светофора таким образом, чтобы по кнопке *НОЧЬ* включался режим мигания желтого света, а по кнопке *ДЕНЬ* светофор переходил в нормальный режим работы.

3 Измените модель светофора с кнопкой таким образом, чтобы по кнопке *ВЫКЛ* светофор выключался (все серое), а по кнопке *ВКЛ* светофор переходил в нормальный режим работы.

4 Измените презентацию модели таким образом, чтобы рядом со светофором высвечивалось время в секундах, оставшееся до смены сигнала.

5 Измените презентацию модели таким образом, чтобы рядом со светофором высвечивалось время в секундах, прошедшее со смены сигнала.

6 Измените модель автоматического светофора таким образом, чтобы по кнопке *СТОП* включался красный свет у всех светофоров, а по кнопке *ДВИЖЕНИЕ* светофор переходил в нормальный режим работы.

7 Измените модель таким образом, чтобы регулировалось движение двух пересекающихся дорожных потоков в автоматическом режиме (без кнопки), т.е. модель двух трехцветных светофоров со следующей последовательностью сигналов: зеленый – зеленый мигающий – желтый – красный – (красный + желтый) – зеленый.

8 Измените модель светофора с кнопкой таким образом, чтобы пешеход видел, нажата уже кнопка или нет, например, с помощью надписи.

9 Измените модель таким образом, чтобы моделировать железнодорожный переезд: по кнопке *ПОЕЗД* включается сигнал и через 5 с. начинает мигать сдвоенный красный светофор (попеременно правый – левый), еще через 5 с. опускается шлагбаум. По истечении случайного промежутка времени в диапазоне от 30 до 60 с. (прохождение поезда), переезд переводится в исходное состояние.

10 Измените модель таким образом, чтобы трехцветный светофор переключался в полуавтоматическом режиме по нажатию кнопок регулировщиком: по кнопке *КРАСНЫЙ* начинает мигать зеленый свет, затем зажигается желтый

и через несколько секунд – красный; по кнопке *ЗЕЛЕНЫЙ* зажигается одновременно красный и желтый, затем горит зеленый.

11 Измените модель таким образом, чтобы частота мигания зеленого света регулировалась с помощью слайдера.

12 Измените модель автоматического светофора автомобилей и пешеходов (без кнопки), удалив переходы по сигналу и подберите подходящие таймауты для согласованной работы светофоров.

13 Измените модель светофора с кнопкой таким образом, чтобы с помощью слайдера можно было регулировать время, через которое повторное нажатие кнопки *ПЕРЕХОД* приведет к переключению светофора автомобилей в режим ожидания.

14 Как передать сигнал от *стейтчарта CT_1 стейтчарту CT_2* ?

15 Измените модель таким образом, чтобы время, в течение которого разрешено движение автомобилей, было 15 секунд и перед окончанием запрещающего сигнала автомобилям мигал красный свет.

16 Измените модель таким образом, чтобы время, в течение которого разрешено движение пешеходов, было 10 секунд и перед окончанием запрещающего сигнала пешеходам мигал красный свет.

17 Как в модели реализована защита от чрезмерно частого нажатия кнопки *ПЕРЕХОД*?

18 Измените модель таким образом, чтобы зеленый сигнал светофора не мигал.

2 ЛАБОРАТОРНАЯ РАБОТА № 2
СТОХАСТИЧЕСКИЕ МОДЕЛИ ТРАНСПОРТНЫХ И ПЕШЕХОДНЫХ ПОТОКОВ. ENTERPRISE LIBRARY. МОДЕЛИРОВАНИЕ ТРАНСПОРТНЫХ СИСТЕМ

Цели занятия:

• освоение методов дискретно-событийного моделирования;

• знакомство с библиотекой Enterprise Library;

• построение модели транспортной системы.

Обеспеченность:

• компьютер с установленной программой AnyLogic версии 6.

2.1 ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Библиотека Enterprise Library поддерживает дискретно-событийный подход моделирования. С помощью объектов библиотеки можно моделировать системы реального мира, динамика которых представляется последовательностью операций (прибытие, задержка, разделение и т.п.) над некими сущностями, представляющими клиентов, документы, звонки, пакеты данных, транспортные средства и т.п. Процессы задаются в форме потоковых диаграмм (блок-схем) в графическом представлении.

Объекты библиотеки Enterprise Library являются «кирпичиками» для построения блок-схемы, моделирующей процессы, происходящие с заявками.

2.2 ПОСТАНОВКА ЗАДАЧИ

Рассмотрим простую систему, предоставляющую собой транспортную развязку, например, перекресток. Через перекресток движутся потоки автомобилей. Для упорядочивания движения служит автоматический светофор или регулировщик. Наиболее типичной целью исследования в подобных задачах является оценка эффективности системы, т.е. нахождение числовых значений характеристик, описывающих качество управления системой. Такими характеристиками могут быть, например, время, требуемое для проезда перекрестка, длина очереди перед светофором и др.

Требуется построить имитационную модель транспортной развязки по индивидуальному заданию и отобразить гистограмму времени её проезда.

2.3 МОДЕЛЬ

Создайте новую модель *Дорога* на основе шаблона *Дискретно-событийное моделирование*. Поставьте галочки напротив выборов: Добавить гистограмму распределения времени пребывания в системе, Добавить график. Пункт – Использовать ресурсы выбирать не нужно.

В верхнем левом углу расположена потоковая диаграмма, состоящая из 4-элементов: *source*, *queue*, *delay* и *sink*. Рассмотрим их подробнее.

Source – создает заявки, в настраиваемые моменты времени. Объект *source* не разрешает заявкам храниться в буфере выходного порта, если они не могут покинуть объект. Поэтому в случае, когда за объектом source расположен объект, который по той или иной причине не может принять новую заявку,

нужно между ними поставить специальный объект буферизации, например, queue.

Queue – хранит заявки в определенном порядке. Моделирует очередь заявок, ожидающих приема объектами, следующими за ним в потоковой диаграмме.

Delay – задерживает заявки на заданный период времени. Время задержки вычисляется динамически, может быть случайным или зависеть от каких-то других условий. Это время может вычисляться как длина фигуры анимации этого объекта, поделенной на «скорость» заявки.

Sink – уничтожает поступившие заявки. Обычно используется в качестве конечной точки потока заявок.

Запустите модель и посмотрите, как она работает.

2.4 ПЕРЕКРЕСТОК

Автомобили подъезжают к перекрестку в произвольные моменты времени, поэтому объект *source* также должен создавать заявки в случайные моменты времени. Выделите объект *source*, и в строке свойств Заявки прибывают согласно выберите Времени между прибытиями. В появившемся поле ввода времени между прибытиями запишите exponential(0.1). Функция exponential генерирует реализацию случайной величины с экспоненциальным законом распределения.

Остальные параметры этого объекта пока оставим без изменения.

В свойствах объекта *queue* следует удалить связь с анимацией: в поле Фигура анимации панели свойств, удалите ссылку на *polyline*. Заявки, находящиеся в очереди, анимироваться не будут. Этот объект будет выполнять буферную функцию для предотвращения ошибки в момент, когда *source* создал заявку, а *delay* еще не готов ее принять.

В свойствах объекта *delay* следует сделать следующие изменения:

- задержка задается – укажите: *«Как длина пути/скорость»*,
- время задержки – исправьте на *triangular(5., 10., 15.)*,
- вместимость – исправьте на *10*,
- фигура анимации – исправьте на *polyline*,
- тип анимации – исправьте на *Путь*.

Изменим фигуру анимации, вместо человечка пусть будет автомобиль. Этот элемент получит имя по умолчанию *car*. В свойствах объекта *source* измените поле *Фигура анимации* заявки на *car* и поставьте галочку в поле *Разрешить вращение*. При запуске, модель должна выглядеть, как показано на рисунке 2.1.

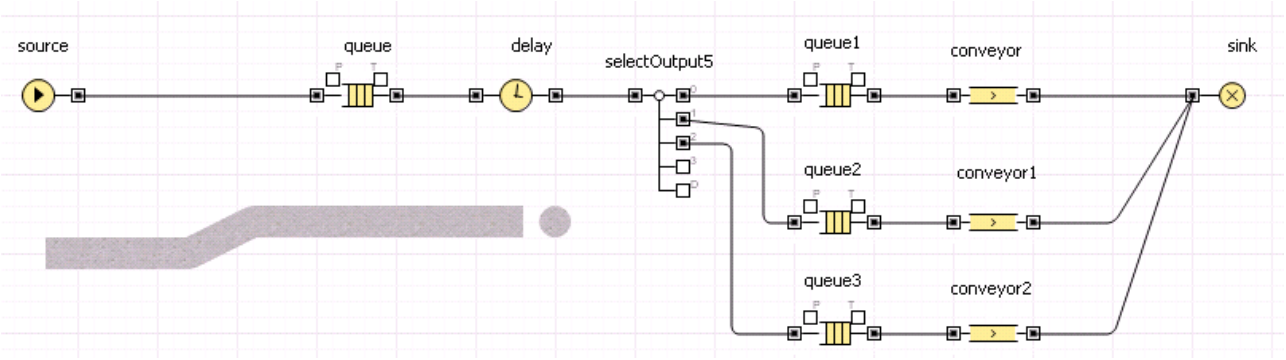


Рисунок 2.1 – Модель

Подъехав к перекрестку, автомобили должны перестроиться в 3 потока для проезда прямо, и поворотов направо и налево. Смоделируем это действие.

Для разделения потока заявок служат объекты *selectOutput* и *selectOutput5*.

Перенесите из палитры Enterprise Library на диаграмму класса Main объект *selectOutput5* и поместите его между *delay* и *sink*, предварительно удалив связь между ними. *SelectOutput5* направляет входящие заявки в один из пяти выходных портов в зависимости от выполнения заданных условий. В свойствах этого объекта укажите Вероятность 0, Вероятность 1 и Вероятность 2 – равными 1/3, а Вероятность 4 и 5 сделайте равными 0.

Двигаясь в 3 потока, автомобили должны останавливаться перед стоплинией на красный сигнал светофора. Для моделирования движения с остановкой подходит объект *Conveyor*, который моделирует конвейер, который перемещает заявки по пути заданной длины с заданной скоростью (одинаковой для всех заявок), сохраняя их порядок и оставляя заданные промежутки между ними. Когда заявка достигает конца конвейера, но не может его покинуть, то она там и остается. Если конвейер накапливающий, то он продолжит двигать заявки, которые имеют достаточно свободного места перед собой.

Перенесите из палитры *Enterprise Library* на диаграмму класса *Main* три объекта *conveyor* и поместите их между *selectOutput5* и *sink*. Так как в объекте *selectOutput5* заявки не могут оставаться, а конвейер может оказаться не готовым их принять, нужно между ними добавить буферный объект *queue*. Соедините объекты.

Для анимации движения заявок (автомобилей) по конвейеру (зоне остановки перед светофором) нужно нарисовать три ломаных линии и связать их с соответствующим объектом *conveyor*.

Чтобы не запутаться при построении модели, необходимо в названиях объектов, относящихся к одному потоку, придерживаться следующих правил: объекты потока № 1 до разделения по направлениям нумеровать соответствующей цифрой, например: *polyline1*, *delay1* и т.д.; объекты этого же потока после разделения по направлениям следует называть так: *polyline11* – для потока, движущегося через перекресток прямо, *polyline1R1 и polyline1L1* – для потоков, движущихся через перекресток соответственно направо и налево.

Нарисуйте ломаные и переименуйте все объекты в соответствии с рисун-

ком 2.2. В свойствах объектов *conveyor* следует сделать следующие изменения:

• длина задается – укажите: *Согласно пути*,

• расстояние между заявками – укажите: *55*,

• фигура анимации – укажите на соответствующую ломаную.

Не забудьте переименовать связь объекта *delay1* с объектом *polyline1*.

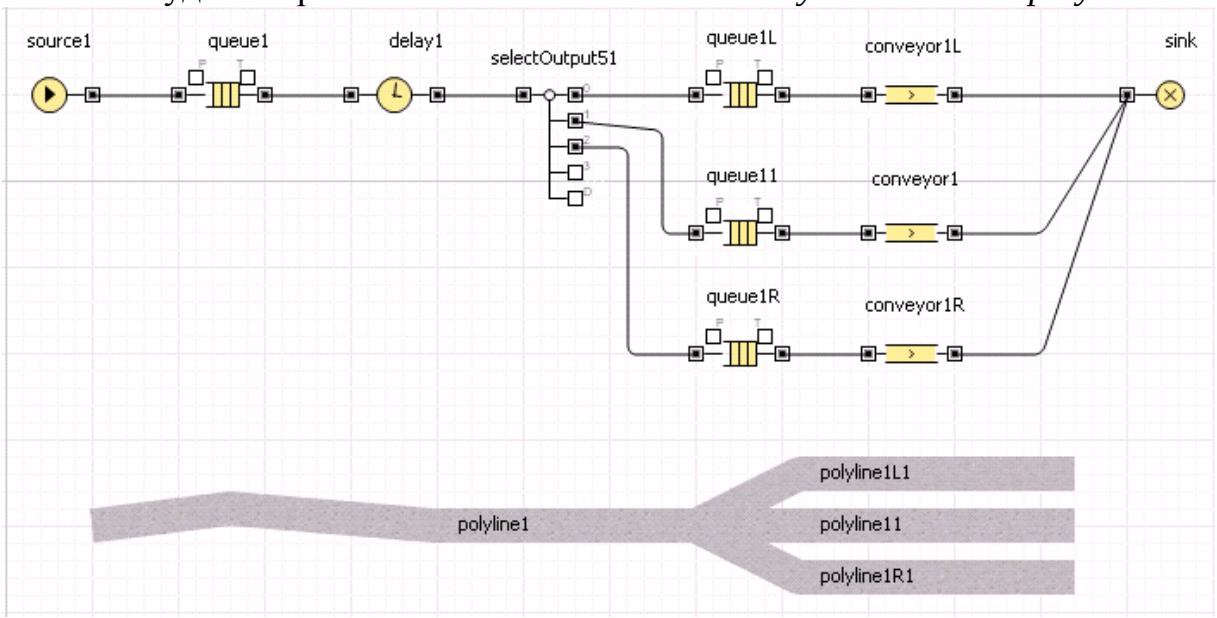


Рисунок 2.2 – Модель

Для моделирования остановки автомобилей на красный свет необходимо в диаграмму установить элемент, останавливающий движение заявок – *hold*. Этот объект блокирует/разблокирует поток заявок на определенном участке блок-схемы. Если объект находится в заблокированном состоянии, то заявки не будут поступать на его входной порт, и будут ждать, пока объект не будет разблокирован. Поставьте 3 объекта *hold* после *conveyor* (рисунок 2.3).

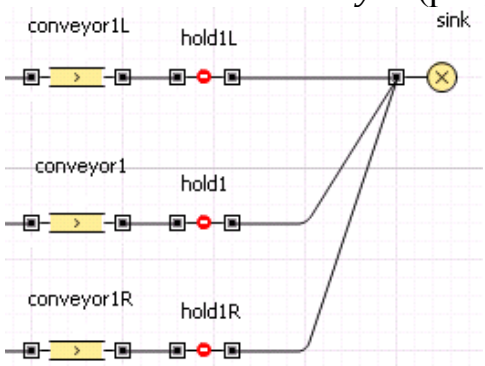


Рисунок 2.3 – Объекты hold

Добавьте на диаграмму две кнопки в свойствах, которых напишите команды для блокирования всех трех объектов *hold*:

```
hold1L.setBlocked( true );
hold1.setBlocked( true );
hold1R.setBlocked( true )
```

и команды для разблокирования объектов hold:

```
hold1L.setBlocked( false );
```

hold1.setBlocked(false);

hold1R.setBlocked(false).

Для отображения сигнала светофора нарисуйте овал рядом с местом остановки автомобилей и динамически изменяемым цветом заливки: красный и зеленый в зависимости от того – разрешено движение данному потоку или нет.

Теперь осталось только добавить элементы для моделирования движения автомобилей после светофора: *delay1L1, delay12, delay1R1 u delay13*, а также ломаные для анимации движения и визуализации границ перекрестка (рисунок 2.4).

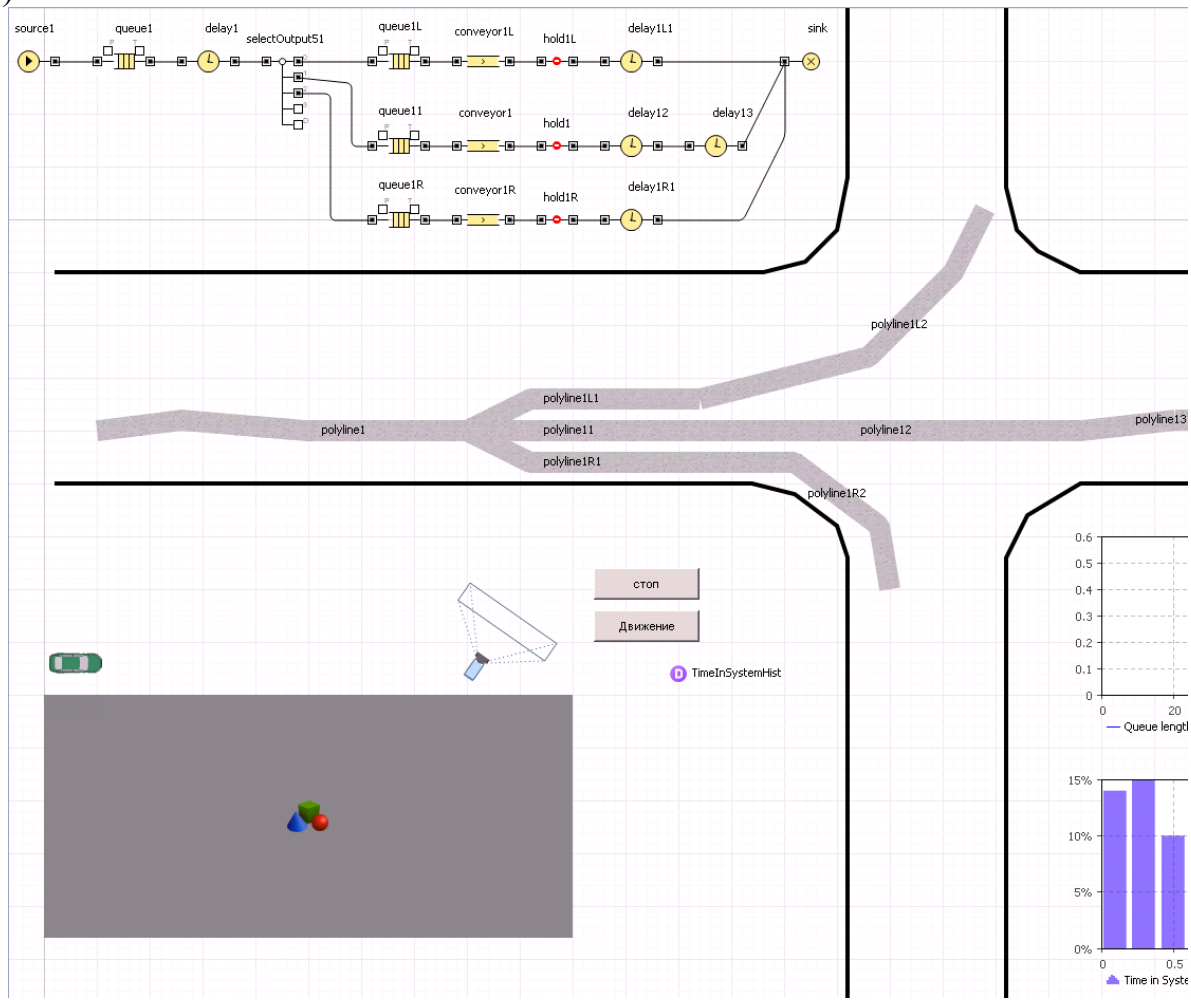


Рисунок 2.4 – Визуализация границ перекрестка

Запустите модель и выполните необходимые доработки в соответствии с заданием.

2.5 КОНТРОЛЬНЫЕ ЗАДАНИЯ

Для выполнения задания необходимо использовать справочную систему AnyLogic по библиотеке Enterprise Library.

1 Доработайте модель таким образом, чтобы отображалось 2 потока автомобилей, движущихся навстречу. Автомобили должны пропускать друг друга согласно правилу «помеха справа».

2 Доработайте модель в перекресток с круговым движением и «главным» кольцом.

3 Добавьте в модель пешеходный переход и анимацию пешеходов. Автомобили должны автоматически останавливаться, как только пешеход ступит на проезжую часть.

4 Доработайте модель в перекресток с круговым движением и «главным» въездом на кольцо.

5 Доработайте модель таким образом, чтобы отображалось 2 потока автомобилей, движущихся во встречном направлении. Поворот налево производится через поворот направо и движение по кольцу. Автомобили должны пропускать друг друга согласно правилу «помеха справа».

6 Доработайте модель в Т-образный перекресток.

7 Доработайте модель таким образом, чтобы отображалось 4 потока автомобилей, движущихся с разных сторон. Перекресток должен регулироваться вручную.

8 Доработайте модель в перекресток с круговым движением и автоматическим регулированием.

9 Доработайте модель таким образом, чтобы отображалось 2 потока автомобилей, движущихся перекрестно, и поток пешеходов, движущийся из «левого нижнего угла» в «правый верхний». Перекресток должен регулироваться автоматически. Пешеходы, подходя к перекрестку, должны идти на тот зеленый сигнал, который горит в данный момент.

10 Доработайте модель таким образом, чтобы отображалось 4 потока автомобилей, движущихся с разных сторон. Перекресток должен регулироваться автоматически.

11 Доработайте модель в железнодорожный переезд со шлагбаумом, поездом и семафором.

12 Доработайте модель таким образом, чтобы отображалось 2 потока автомобилей, движущихся во встречном направлении. Поворот налево производится через поворот направо и движение по кольцу.

3 ЛАБОРАТОРНАЯ РАБОТА № 3 МИКРОСКОПИЧЕСКИЕ МОДЕЛИ. PEDESTRIAN LIBRARY. МОДЕЛЬ ДВИЖЕНИЯ ПАССАЖИРОВ В НАЗЕМНОМ ПАВИЛЬОНЕ МЕТРО

Цели занятия:

• знакомство с библиотекой Pedestrian Library;

• построение модели движения пассажиров.

Обеспеченность:

• компьютер с установленной программой AnyLogic версии 6.

3.1 ПОСТАНОВКА ЗАДАЧИ

В ходе выполнения лабораторной работы необходимо создать простейшую модель, движение пассажиров в наземном павильоне метро. Перед тем, как пройти к поездам метро, пассажиры проходят через турникеты, проверяющие наличие билетов. Те пассажиры, которые не купили билеты заранее, должны будут вначале приобрести их в находящейся в павильоне билетной кассе, и

только потом они смогут пройти к поездам. Эта модель продемонстрирует, как промоделировать поток пешеходов и простейшие сервисы в AnyLogic Pedestrian Library.

3.2 ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Вначале мы создадим новую модель.

Начиная с версии 6.4, AnyLogic предоставляет пользователям возможность использования шаблонов моделей при создании новых моделей. Если раньше Вам приходилось всегда начинать создание модели «с чистого листа», зачастую выполняя одни и те же типовые действия для каждой новой создаваемой модели, то теперь Вы можете перепоручить выполнение первых, базовых, шагов Мастеру создания модели. Все, что Вам нужно – это указать, какой метод моделирования Вы будете использовать и выбрать те опции, которые Вам нужны в модели, – и Мастер автоматически создаст простейшую модель, а Вы сможете продолжить ее разработку, лишь изменив незначительные детали.

Порядок создания модели:

1 Щелкните мышью по кнопке панели инструментов *Создать*  . Появится окно Мастера создания модели.

2 Задайте имя новой модели. В поле *Имя модели* введите *Subway Entrance*.

3 Выберите каталог, в котором будут сохранены файлы модели. Если Вы хотите сменить предложенный по умолчанию каталог на какой-то другой, Вы можете ввести путь к нему в поле *Местоположени*» или выбрать этот каталог с помощью диалога навигации по файловой системе, открывающегося по нажатию на кнопку *Выбрать*.

4 Щелкните мышью по кнопке *Далее*. Откроется вторая страница Мастера создания модели.

5 Здесь Вам будет предложено выбрать шаблон модели, на базе которого Вы будете разрабатывать Вашу модель. Выбираем создания модели «с нуля», заканчиваем создание модели, щелкнув мышью по кнопке *Готово*.

Добавьте рисунок с изображением плана павильона:

1 Вначале откройте закладку Презентация панели Палитра. Чтобы открыть какую-либо закладку панели нужно щелкнуть мышью по заголовку этой палитры.

2 Палитра Презентация содержит элементы, используемые для рисования презентаций моделей: фигуры, с помощью которых Вы можете рисовать сложные презентации, а также элементы управления, с помощью которых Вы можете сделать Ваши презентации интерактивными.

3 Перетащите элемент *Изображение*  из палитры Презентация на диаграмму класса активного объекта. Поместите его так, как показано на рисунке 3.1

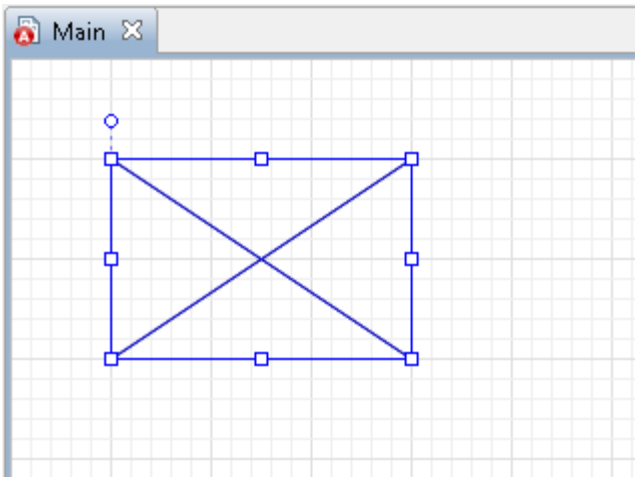


Рисунок 3.1 – Элемент изображения

4 Задайте свойства изображения в панели Свойства. Щелкните мышью по кнопке *Добавить* и выберите файл изображения плана павильона. Файл находится в каталоге <каталог AnyLogic>\resources\tutorials\Subway Entrance\entrance_layout.png .Вы увидите добавленное изображение в области предварительного просмотра на панели Свойства (рисунок 3.2).

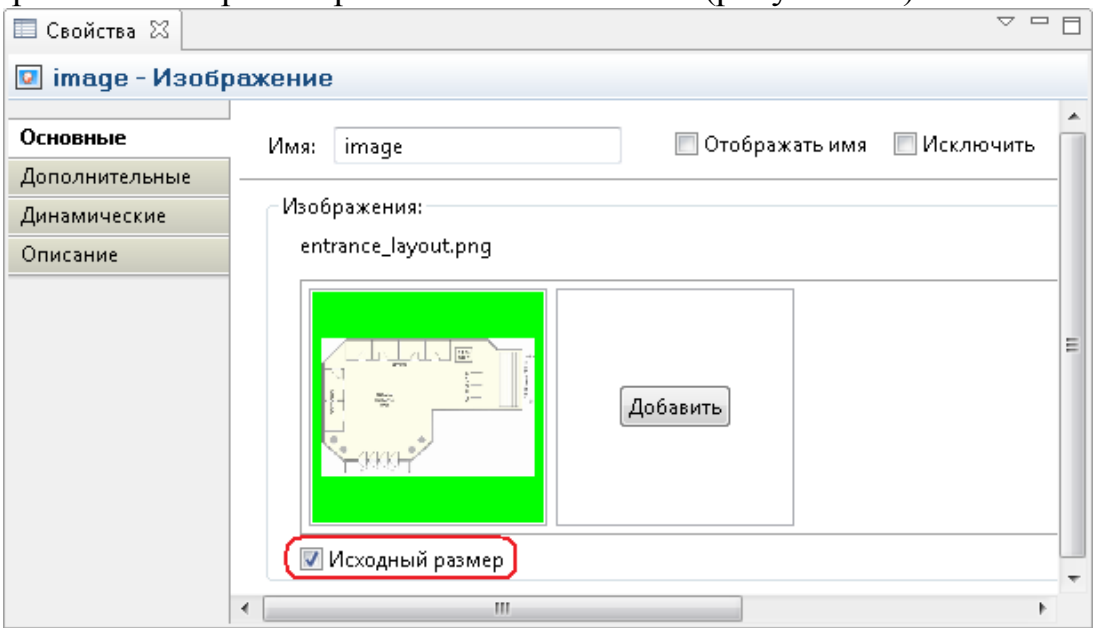


Рисунок 3.2 – Установка параметров

5 Чтобы сохранить исходный размер изображения, установите флажок Исходный размер.

Теперь нарисуем на анимации объекты моделируемой среды.

Сначала нарисуем границу моделируемого нами пространства, играющей роль стен здания.

Нарисуйте границы здания

1 Чтобы было легче рисовать, лучше отключить сетку и увеличить масштаб анимации с помощью соответствующих кнопок панели инструментов.

2 Нарисуйте ломаную, как показано на рисунке ниже. Чтобы нарисовать ломаную, сделайте двойной щелчок мышью по элементу *Ломаная* в па-

литре (при этом его значок должен поменяться на этот:). Теперь Вы можете рисовать ломаную точка за точкой, последовательно щелкая мышью в тех точках диаграммы, куда Вы хотите поместить вершины ломаной. Чтобы завершить рисование, добавьте последнюю точку ломаной двойным щелчком мыши. (рисунок 3.3)

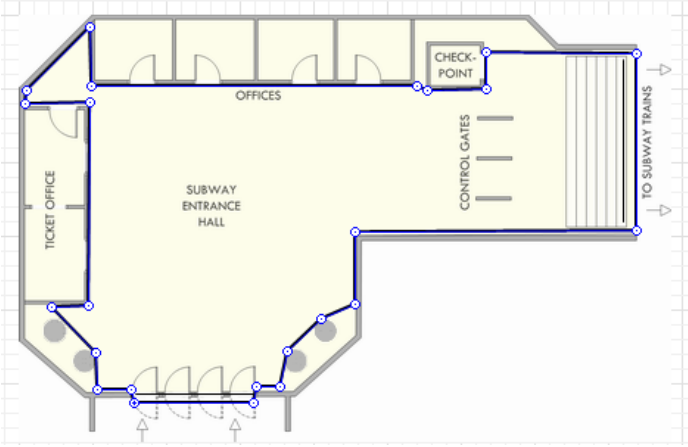


Рисунок 3.3 – Создание ломаной кривой

3 Измените свойства только что нарисованной ломаной:

a) назовите ломаную *walls*;

b) измените цвет и толщину линии, чтобы она была более заметна на презентации;

c) сделайте ломаную замкнутой. Установите флажок *Замкнутая*. Тем самым, Вы сделаете ломаную замкнутой, соединив ее первую и последнюю точки. (рисунок 3.4):

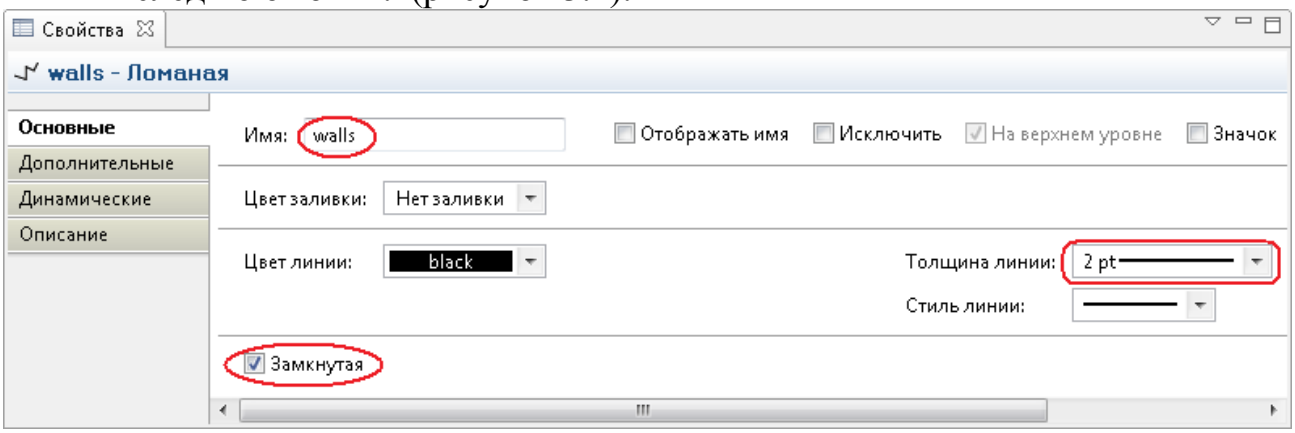


Рисунок 3.4 – Свойства ломаной walls

Зачастую границы моделируемого пространства рисуются с помощью нескольких фигур, поэтому у библиотеки *Pedestrian Library* есть требование, чтобы все такие фигуры были добавлены в одну группу, а эта группа была указана в параметре Стены соответствующего библиотечного объекта. Поэтому сейчас мы добавим нарисованную нами ломаную линию *walls* в группу.

Добавьте только что нарисованную ломаную в группу. Щелкните по ней правой кнопкой мыши (при этом она должна подсветиться синим цветом) и выберите Группировка|Создать группу из контекстного меню.

Теперь нужно задать области входа и выхода пешеходов.

Вначале мы нарисуем область входа – линию, в которой пешеходы будут

появляться. Область входа может быть задана линией или ломаной.

Нарисуйте вход:

1) перетащите элемент *Линия* ⟋ из палитры Презентация в графический редактор;

2) перетащите точки линии так, чтобы они были расположены примерно в тех же местах диаграммы, что и точки линии на рисунке;

3) назовите линию *entry*. Позднее мы будем ссылаться на эту линию в блоках диаграммы процесса именно по этому имени.

Нарисуйте выход:

1) аналогично нарисуйте линию выхода пешеходов из моделируемого пространства. При достижении линии выхода пешеходы будут удаляться из моделируемой среды.

2) назовите линию *exit*.

Обе линии должны находиться полностью внутри фигуры, задающей границу моделируемой среды, в нашем случае – внутри ломаной линии walls.

Теперь закончим создание модели, моделирующей простейший пассажиропоток, создав диаграмму моделируемого нами процесса.

С помощью диаграммы процесса в моделях пешеходной динамики задается поведение пешеходов (так же, как в библиотеке Enterprise Library с их помощью задается поведение заявок). Диаграмма процесса в AnyLogic создается путем добавления объектов библиотеки из палитры на диаграмму класса активного объекта, соединения их портов и изменения значений свойств объектов в соответствии с требованиями Вашей модели.

Попадающие в моделируемую систему пешеходы будут последовательно проходить по блокам созданной Вами диаграммы процесса.

Помимо объектов, составляющих диаграмму процесса, модели Pedestrian Library состоят из объектов, моделирующих объекты среды (стены, различные области, сервисы, очереди и т.д.). Чтобы задать объект среды, Вы должны вначале графически нарисовать его на анимации, а затем добавить соответствующий объект библиотеки на структурную диаграмму активного класса модели и задать необходимые свойства этого объекта.

Создайте диаграмму процесса

1 Добавьте на диаграмму класса *Main* объекты библиотеки *Pedestrian Library* (имена их типов показаны на рисунке 3.5).

2 Чтобы добавить на диаграмму объект библиотеки *Pedestrian Library*, нужно открыть в панели Палитра палитру этой библиотеки, щелкнув мышью по панели с ее заголовком, а затем перетащить нужный Вам объект из палитры на диаграмму класса.

Соедините порты добавленных на диаграмму объектов, как показано на рисунке 3.5.

1 Чтобы соединить порты объектов, сделайте двойной щелчок мышью по одному порту, затем при желании щелкните в тех местах диаграммы, где Вы хотите добавить точку изгиба соединителя, и завершите создание, сделав двойной щелчок мышью по второму порту. При этом между портами появится соединитель заданной Вами формы. (рисунок 3.5)

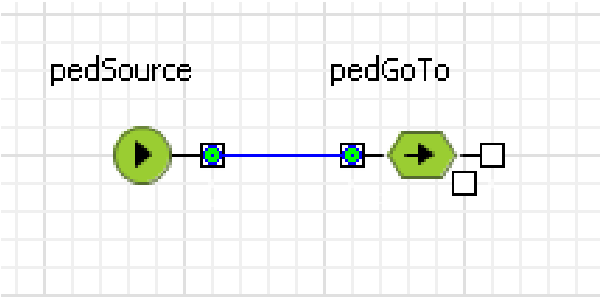


Рисунок 3.5 – Соединение портов

2 Если Вы выделите соединитель щелчком мыши, и его конечные точки в портах будут подсвечиваться светло-зеленым цветом, то это будет означать, что Вы успешно соединили порты. Иначе же Вам придется проверить, точно ли в порт была добавлена одна из двух конечных точек соединителя, и если нужно, то передвинуть ее туда.

Согласно принятым стандартам, блоки в диаграмме процесса обычно располагаются цепочкой слева направо, представляя собой последовательную очередность операций, которые будут производиться над пешеходом.

Теперь, когда мы создали диаграмму моделируемого нами процесса, нам осталось только немного подкорректировать значения объектов диаграммы в соответствии с нашими требованиями. Давайте последовательно сконфигурируем каждый добавленный нами библиотечный объект (на приведенном выше рисунке мы отметили тип каждого блока, чтобы Вам было легче привыкнуть к пока незнакомым пиктограммам этих объектов).

Объект *PedGround* позволяет задавать двумерное пространство в моделируемой среде, представляющее собой «этаж», т.е. поверхность, по которой будут перемещаться пешеходы. Этажи могут быть ограничены какой-то стеной или быть неограниченными. Стены - это объекты, которые пешеходы не могут пересекать. Стены являются частью этажа, то есть одна стена не может быть использована несколькими этажами.

Измените свойства блока *pedGround*

Свойства объекта (как и любого другого элемента AnyLogic) можно изменить в панели *Свойства*.

Обратите внимание, что панель *Свойства* является контекстно-зависимой – она отображает свойства выделенного в текущий момент элемента. Поэтому для изменения свойств элемента нужно будет предварительно щелчком мыши выделить его в графическом редакторе или в панели *Проект*.

Чтобы у Вас всегда была уверенность в том, что в текущий момент в рабочем пространстве выбран именно нужный Вам элемент, и именно его свойства Вы редактируете в панели *Свойства*, обращайте внимание на первую строку, показываемую в панели *Свойства* – в ней отображается имя выбранного в текущий момент времени элемента и его тип.

Задайте стены моделируемого этажа. Введите в поле *Стены* (группа, необязательный) имя нашей группы: *group*, созданной ранее именно для этой цели (рисунок 3.6).

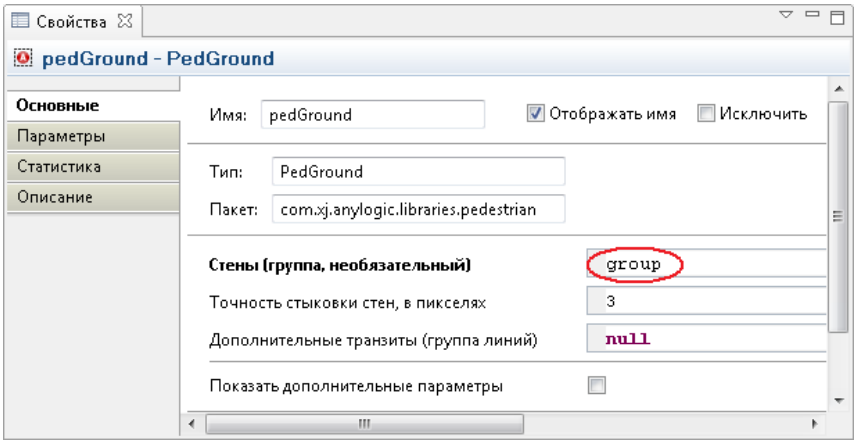


Рисунок 3.6 – Свойства блока pedGround

Объект *PedGround* создает пешеходов. Обычно он используется в качестве начальной точки диаграммы процесса, формализующей поток пешеходов. В нашем примере он моделирует приход пассажиров в павильон.

Измените свойства блока pedSource

Укажите имя объекта *PedGround*, задающего этаж, на который будут прибывать пассажиры. В поле *Этаж* (PedGround) введите имя добавленного Вами ранее объекта *PedGround*: *pedGround*.

Задайте место на этаже, где будут появляться пассажиры. Место появления пешеходов на этаже может быть задано линией или ломаной. Введите *entry* (имя ломаной, нарисованной нами ранее для этой цели) в поле *Место* появления (линия, ломаная). Теперь наши пешеходы будут появляться в случайно выбранной точке линии *entry* (рисунок 3.7):

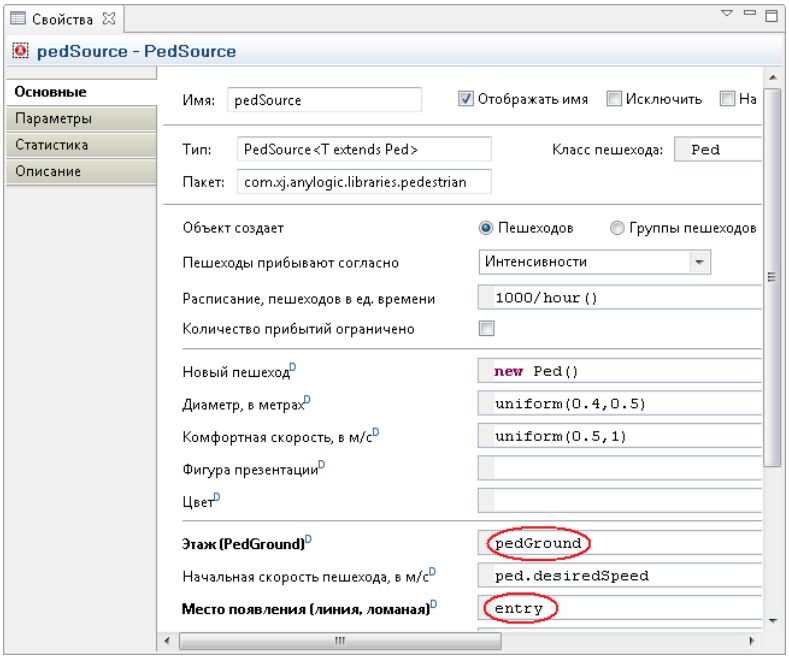


Рисунок 3.7 – Свойства блока pedSource

Следующий объект в созданной нами диаграмме процесса – *PedGoTo*. Этот объект моделирует перемещение пешеходов из текущего местоположения

в другое (заданное параметром этого объекта). С помощью этого объекта мы будем моделировать то, как пассажиры перемещаются от входа в павильон к поездам метро.

Измените свойства блока pedGoTo

Задайте то место, куда будут перемещаться пешеходы, достигшие этого блока в диаграмме процесса. Такое место может быть задано линией или точкой, нарисованной на презентации. На данный момент мы хотим, чтобы пассажиры, вошедшие в павильон, сразу двигались к поездам метро. Поэтому введите *exit* (имя линии, нарисованной нами на плане у выхода из коридора, ведущего к поездам метро) в поле *Цель* (точка, линия).

Теперь для того, чтобы покинуть моделируемую среду, пешеходы должны будут вначале дойти до заданной области выхода (рисунок 3.8).

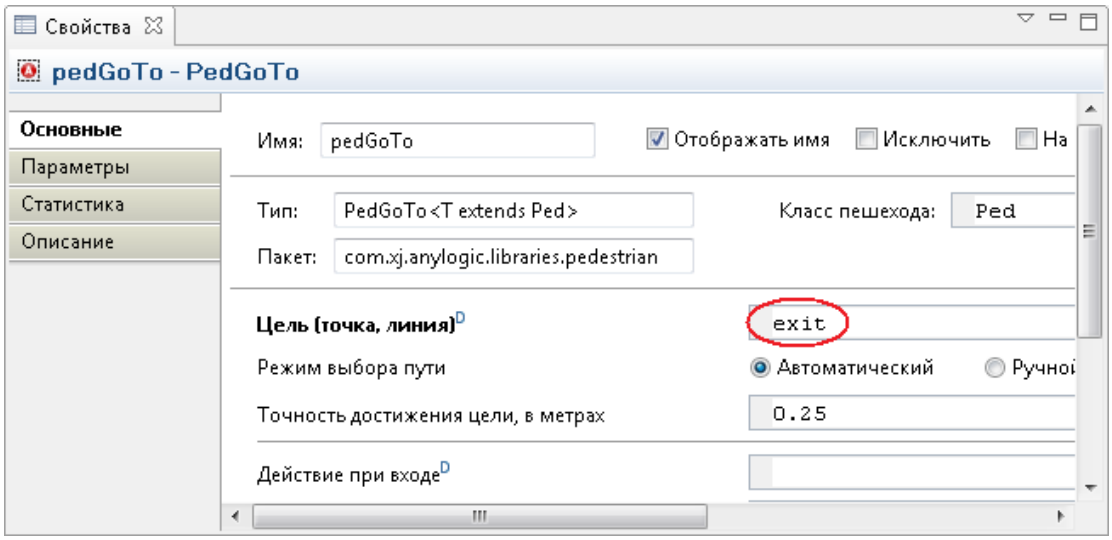


Рисунок 3.8 – Свойства блока pedGoTo

Оставьте заданные по умолчанию свойства объекта *PedSink* без изменений. Объект *PedSink* удаляет поступивших в объект пешеходов из моделируемой среды. Обычно объект используется в качестве конечной точки диаграммы процесса.

Объект *PedConfiguration* позволяет изменять общие настройки библиотеки, настроив ее под конкретную задачу таким образом, чтобы повысить производительность модели.

Измените свойства блока pedConfiguration

Сбросьте флажок *Скрыть* фигуры среды, чтобы не отображать на анимации фигуры, которые использовались для задания моделируемой среды (стен, различных областей, сервисов и т.д.) (рисунок 3.9).

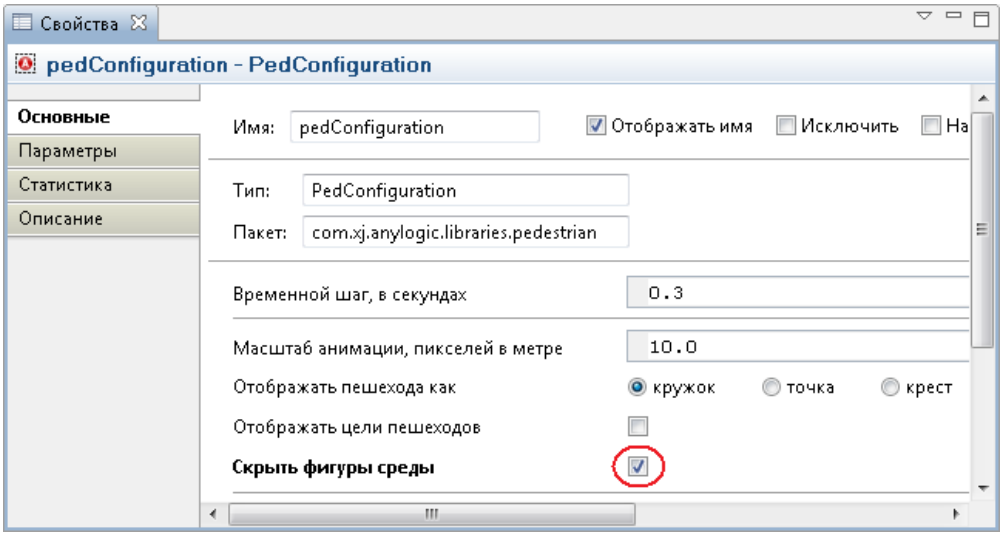


Рисунок 3.9 – Свойства блока pedConfiguration

Детальное описание объектов *Pedestrian Library*, их функций и параметров Вы можете найти в Справочном руководстве по *Pedestrian Library* (отдельная книга Справки AnyLogic).

Мы закончили создание простой модели павильона метро. Теперь давайте запустим ее и посмотрим, как она работает.

Вы можете сконфигурировать выполнение модели в соответствии с Вашими требованиями. Модель выполняется в соответствии с набором установок, задаваемым специальным элементом модели – экспериментом. Вы можете создать несколько экспериментов с различными установками и изменять рабочую конфигурацию модели, просто запуская тот или иной эксперимент модели.

В панели Проект эксперименты отображаются в нижней части дерева модели. Один эксперимент, названный Simulation, создается по умолчанию. Это простой эксперимент, позволяющий запускать модель с заданными значениями параметров, поддерживающий режимы виртуального и реального времени, анимацию и отладку модели.

Если мы сейчас запустим модель, то будут моделироваться 100 единиц модельного времени, после чего выполнение модели будет остановлено. Поскольку Вы можете захотеть наблюдать поведение модели в течение длительного периода (до того момента, пока Вы сами не остановите выполнение модели), нужно изменить соответствующие настройки эксперимента.

Уберите условие остановки модели по прошествии заданного времени:

1) в панели Проект, выделите эксперимент Simulation:Main щелчком мыши;

2) перейдите на страницу Модельное время панели Свойства. Здесь задаются установки модельного времени. Вы можете увидеть, что по умолчанию в качестве единиц модельного времени в моделях изучения движения пешеходов используются секунды;

3) чтобы убрать условие остановки модели, выберите *Нет* из выпадающего списка *Остановить* (рисунок 3.10).

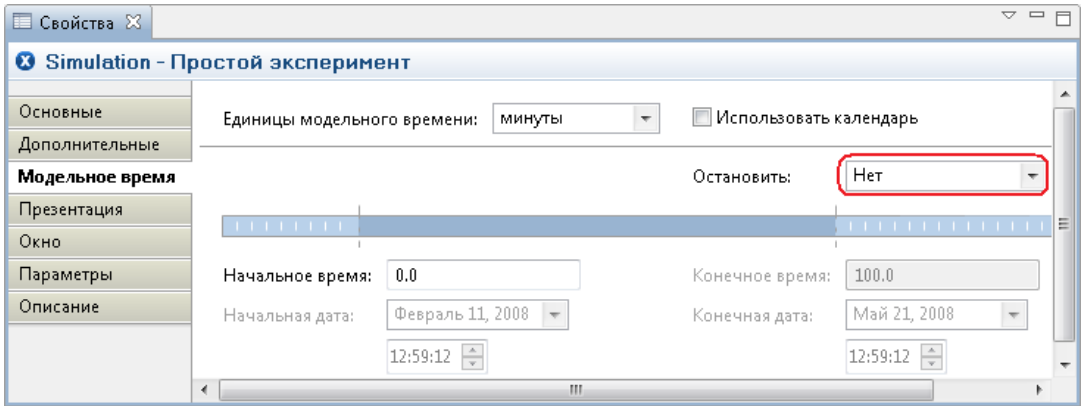


Рисунок 3.10 – Эксперимент Simulation

Постройте Вашу модель с помощью кнопки панели инструментов Построить модель (при этом в рабочей области AnyLogic должен быть выбран какой-то элемент именно этой модели). Если в модели есть какие-нибудь ошибки, то построение не будет завершено, и в панель *Ошибки* будет выведена информация об ошибках, обнаруженных в модели. Двойным щелчком мыши по ошибке в этом списке Вы можете перейти к предполагаемому месту ошибки, чтобы исправить ее.

Запустите модель

Щелкните мышью по кнопке панели инструментов *Запустить* и выберите из открывшегося списка эксперимент, который Вы хотите запустить. Эксперимент этой модели будет называться Subway Entrance/Simulation.

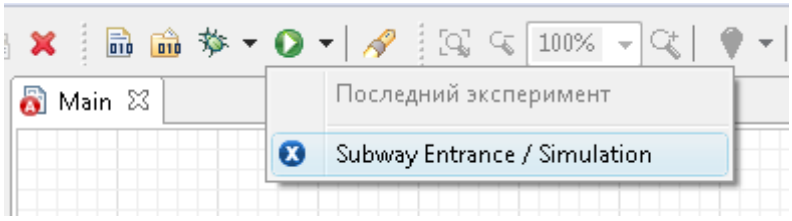


Рисунок 3.11 – Запуск эксперимента

В дальнейшем по нажатию на кнопку *Запустить* (или по нажатию F5) будет запускаться тот эксперимент, который запускался Вами в последний раз. Чтобы выбрать какой-то другой эксперимент, Вам будет нужно щелкнуть мышью по стрелке, находящейся в правой части кнопки *Запустить* и выбрать нужный Вам эксперимент из открывшегося списка (или щелкнуть правой кнопкой мыши по этому эксперименту в панели *Проект* и выбрать *Запустить из контекстного меню*).

AnyLogic автоматически помещает на презентацию каждого простого эксперимента заголовок и кнопку, позволяющую запустить модель и перейти на презентацию, нарисованную Вами для главного класса активного объекта этого эксперимента (Main).

Щелкните по кнопке *Запустить* модель и открыть презентацию класса *Main*. Модель запустится, и Вы сможете понаблюдать за динамикой моделируемого процесса, с помощью нарисованной Вами в классе *Main* презентации.

Запустив модель, Вы увидите окно презентации этой модели. В нем будет отображена презентация запущенного эксперимента (рисунок 3.12).

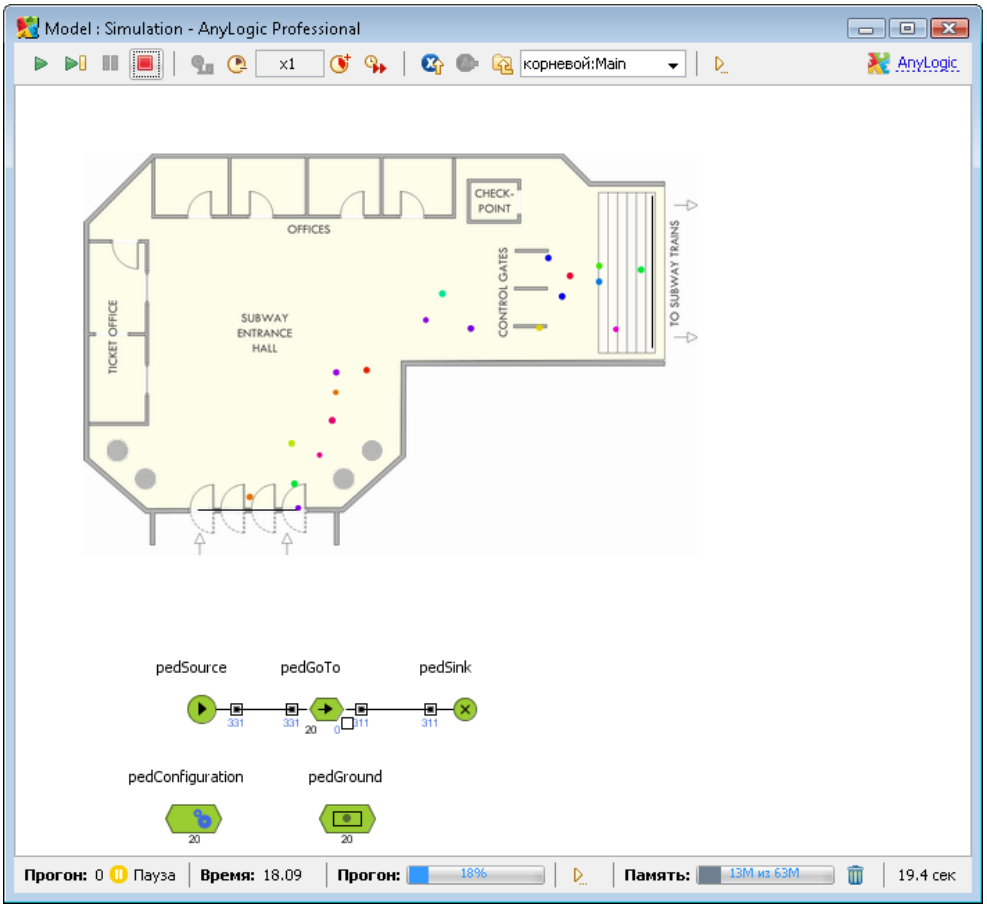


Рисунок 3.12 – Окно эксперимента

Можно увидеть, что пассажиры входят в павильон и движутся по коридору, ведущему к поездам метро.

Вы можете изменить скорость выполнения модели с помощью кнопок панели инструментов *Замедлить* и *Ускорить* 

.

Вы можете переключиться в режим виртуального времени, тогда модель будет выполняться на максимально возможной скорости и Вы сможете быстро промоделировать работу системы за долгий период времени. Чтобы переключиться в режим виртуального времени, нужно щелкнуть мышью по кнопке панели инструментов *Виртуальное время* Реальное/виртуальное время  .

Когда Вы захотите остановить выполнение модели, щелкните мышью по кнопке панели управления окна презентации Прекратить выполнение .

СПИСОК ЛИТЕРАТУРЫ

1 Карпов, Ю. Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5.0 [Текст] / Ю. Г. Карпов. – СПб. : БХВ-Петербург, 2005. – 400 с. : ил.

2 Карпов, Ю. Г. Изучение современных парадигм информационного моделирования в среде AnyLogic [Текст] / Ю. Г. Карпов // Компьютерные инструменты в образовании. – 2005. – № 12. – С. 3-14.

3 Копыльцов, А. В. Компьютерное моделирование: сферы и границы применения. Практикум [Текст] : учебное пособие для 10-11 классов общеобразовательных учреждений / А. В. Копыльцов. – СПб. : СМИО Пресс, 2005. – 144 с.

4 Осоргин, А. Е. Лабораторный практикум по дисциплине «компьютерное моделирование» с использованием программы AnyLogic [Текст] / А. Е. Осоргин. – Самара : Изд-во Самарского гос. пед. ун-та, 2008.

5 Официальный сайт разработчика системы AnyLogic. Дистрибутивы, примеры моделей, руководства, статьи и другая информация. 2013. URL: http://www.xjtek.ru (дата обращения: 1.12.2013).

6 Имитационное моделирование систем. 2013. URL: http://www.gpss.ru (дата обращения: 1.12.2013).

7 Примеры имитационных моделей, построенных в среде AnyLogic. 2013. URL: http://headwire.narod.ru/ (дата обращения: 1.12.2013).

8 Альтернативная система компьютерного моделирования VMS: дистрибутив, руководство, примеры моделей, примеры уроков и другие материалы. 2013. URL: http://www.exponenta.ru/soft/Others/mvs/mvs.asp (дата обращения: 1.12.2013).

Безотеческих Николай Сергеевич

МОДЕЛИРОВАНИЕ ДОРОЖНОГО ДВИЖЕНИЯ

Методические указания
к выполнению лабораторных работ
для студентов всех форм обучения
направления подготовки 190700.62

Редактор Е.А. Могутова

Подписано в печать 30.06.15	Формат	Бумага 65 г/м^2
Печать цифровая	Усл. печ.л. 2,0	Уч.-изд. л. 2,0
Заказ 149	Тираж 25	Не для продажи

РИЦ Курганского государственного университета.
640000, г. Курган, ул. Советская, 63/4.
Курганский государственный университет.