

Министерство науки и высшего образования
Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Кузбасский государственный технический университет
имени Т. Ф. Горбачева»

Институт профессионального образования
Кафедра информатики и информационных систем

Составитель К. А. Кулиничев

ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ

Методические материалы

Рекомендовано цикловой методической комиссией
Информационных систем и программирования
в качестве электронного издания
для использования в образовательном процессе

Кемерово 2026

Рецензенты:

К. В. Ерошевич, преподаватель кафедры информатики и информационных систем.

Кулиничев Константин Андреевич

Имитационное моделирование : методические материалы для обучающихся специальности СПО 09.02.11 Разработка и управление программным обеспечением / Кузбасский государственный технический университет имени Т. Ф. Горбачева, Кафедра информатики и информационных систем ; составитель К. А. Кулиничев. – Кемерово : КузГТУ, 2026. – 1 файл (2993 КБ). – Текст: электронный.

Методические материалы для дисциплины «Имитационное моделирование» содержат указания для выполнения лабораторных занятий и самостоятельной работы, перечень вопросов на защиту выполненных работ.

© Кузбасский государственный
технический университет
имени Т. Ф. Горбачева, 2026
© Кулиничев К. А.,
составление, 2026

Лабораторная работа №1.

Построение дискретно-событийных моделей.

Основные блоки потоковой диаграммы. Модели прибытия

Моделирование прибытия агентов в систему

Для построения простейшей системы массового обслуживания применяются блоки библиотеки дискретно-событийного моделирования: Source, Delay, Queue, Sink.

Блок Source  out – создает агентов. Обычно используется в качестве начальной точки потока агентов.

Блок Sink  in – уничтожает поступивших агентов. Обычно используется в качестве конечной точки потока агентов.

Блок Delay  in out – задерживает агентов на заданный период времени.

Блок Queue  in out – моделирует очередь агентов, ожидающих обслуживания объектами, следующими за ним в потоковой диаграмме.

outPreempted outTimeout

in out

Основные функции блока Source

- void inject() – создает одного агента.
- void inject(int n) – создает заданное количество агентов (равное n).
- long count() – возвращает количество агентов, созданных этим объектом.

Например, `text4.setText(source.count());` выводит в текстовый блок количество созданных агентов.

Параметры блока Source

- Прибывают согласно: интенсивности, времени между прибытиями, расписанию прибытий из базы данных (БД), расписанию интенсивностей, расписанию прибытий, вызовам метода `inject()` – агенты создаются не автоматически, а только при вызове метода `inject()`.

Имя параметра: arrivalType.

Значение по умолчанию: Интенсивности.

Допустимые значения:

Source.RATE,

Source.INTERARRIVAL_TIME,

Source.DATABASE_ARRIVAL_TABLE,

Source.RATE_TABLE,

Source.ARRIVAL_TABLE,

Source.MANUAL.

- Интенсивность прибытия – интенсивность прибытия агентов.

Синтаксис: double rate

Значение по умолчанию: 1 в секунду

Например: увеличить интенсивность прибытия в 2 раза

`source.rate = source.rate*2.`

- Расписание интенсивностей – имя расписания, в котором задано, как интенсивность прибытия агентов изменяется с течением времени.
- Расписание прибытий – имя расписания, в котором заданы моменты появления агентов и количество агентов, прибывающее в каждый указанный в расписании момент времени.
- Агент – тип агентов, создаваемых блоком Source.
- Изменить размеры – если опция выбрана (true), можно будет изменить размеры агента, создаваемого этим блоком, задав соответствующие значения в параметрах Длина, Ширина и Высота.

Рассмотрим различные варианты прибытия агентов в систему.

Задание

Вариант 1. Прибытие агентов, согласно заданной интенсивности (рисунок 1).

Для построения модели необходимо использовать следующие элементы:

1. Текстовый блок (Палитра->Презентация)
2. Параметр int КоличествоЗаявок (Палитра->Агент)
3. Бегунок (sliderRate), связанный с параметром КоличествоЗаявок. Минимальное значение – 0, максимальное – 5 заявок/ч.

(Палитра->Элементы управления). Бегунок необходим для изменения интенсивности прибытия во время имитации.

4. Блок Source (sourceRate). Настройки параметров блока: прибывает согласно – интенсивности; Интенсивность прибытия – КоличествоЗаявок (ссылка на динамический параметр).

5. Блок Delay (delayRate). Время задержки – 10 ч. Место агентов – polylineRate. Вместимость – Максимальная вместимость.

6. Блок Sink (sink4).

7. Путь (polylineRate). Предназначен для отображения обслуживаемых агентов.

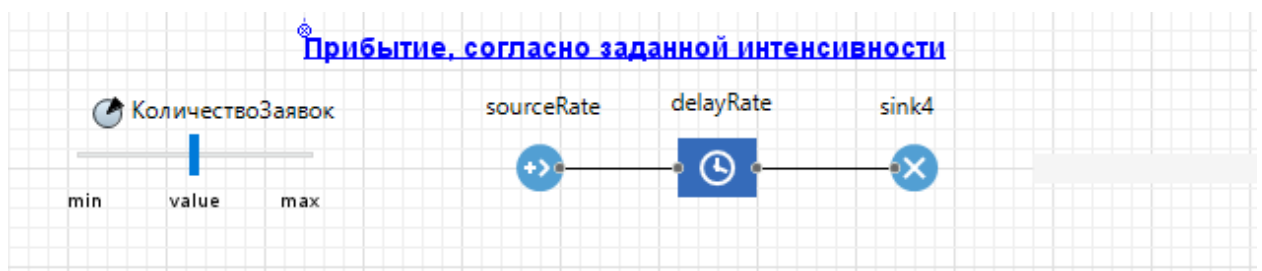


Рисунок 1 – Модель прибытия агентов (вариант 1)

Вариант 2. Прибытие агентов, согласно времени прибытия (рисунок 1.1).

Для построения модели необходимо использовать следующие элементы:

1. Текстовый блок (Палитра->Презентация).

2. Параметр int interarrivalTime (Палитра->Агент).

3. Бегунок (slider1), связанный с параметром interarrivalTime. Минимальное значение – 1, максимальное – 5 ч. (Палитра->Элементы управления). Бегунок необходим для изменения времени между прибытиями во время имитации.

4. Блок Source (sourceIAT). Настройки параметров блока: Прибывает согласно – Времени между прибытиями; Время между прибытиями – interarrivalTime (ссылка на динамический параметр).

5. Блок Delay (delayIAT). Время задержки – 10 ч. Место агентов – polylineIAT.

6. Блок Sink (sink).

7. Путь (polylineIAT). Предназначен для отображения обслуживаемых агентов.

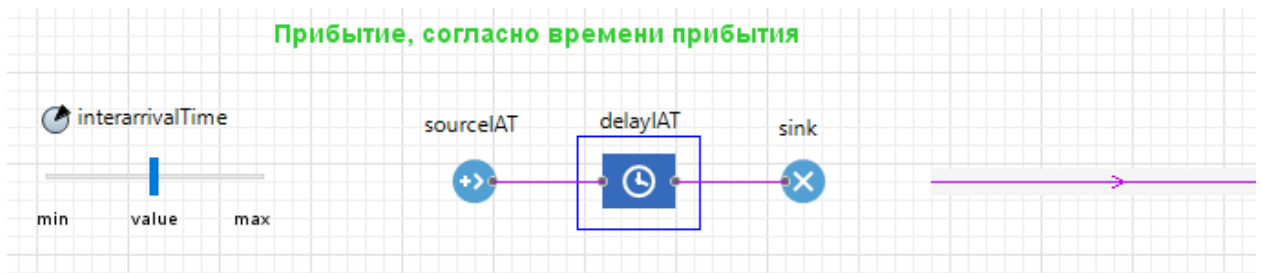


Рисунок 1.1 – Модель прибытия агентов (вариант 2)

Вариант 3. Прибытие агентов, согласно вызовам функции inject().

Для построения модели необходимо использовать следующие элементы (Рисунок 1.2):

1. Текстовый блок (Палитра->Презентация).
2. Параметр int numberToInject (Палитра->Агент). Отвечает за количество поступающих в систему агентов.
3. Бегунок (slider2), связанный с параметром numberToInject. Минимальное значение – 1, максимальное – 10. (Палитра->Элементы управления). Бегунок необходим для изменения количества поступающих в систему агентов.
4. Блок Source (sourceInject). Настройки параметров блока: Прибывает согласно – вызовам функции inject.
5. Блок Delay (delayInject). Время задержки – 10 ч. Место агентов – rectangleInject.
6. Блок Sink (sink5).
7. Прямоугольный узел (rectangleInject). Предназначен для отображения обслуживаемых агентов.
8. Кнопка (button).

Действие: sourceInject.inject(numberToInject).

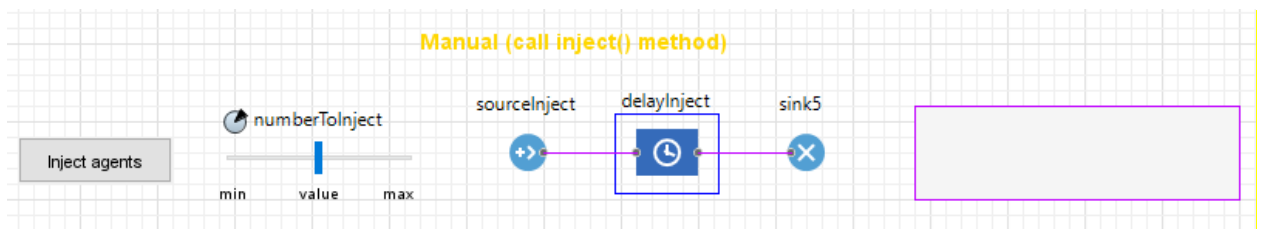


Рисунок 1.2 – Модель прибытия агентов (вариант 3)

Вариант 4. Прибытие агентов, согласно расписанию интенсивности (рисунок 1.3).

Для построения модели необходимо использовать следующие элементы:

1. Текстовый блок (Палитра->Презентация).
2. Расписание интенсивностей (rateSchedule) (Палитра->Агент).
3. Блок Source (sourceRateShedule). Настройки параметров блока: прибывает согласно – расписанию интенсивностей, Расписание интенсивностей – rateSchedule.
4. Блок Sink (sink1).
5. Данные гистограммы (dataRS). Предназначен для сбора данных. Количество интервалов – 24 ч. (Палитра->Статистика).
6. Гистограмма (chart). Данные: dataRS. (Палитра->Статистика).

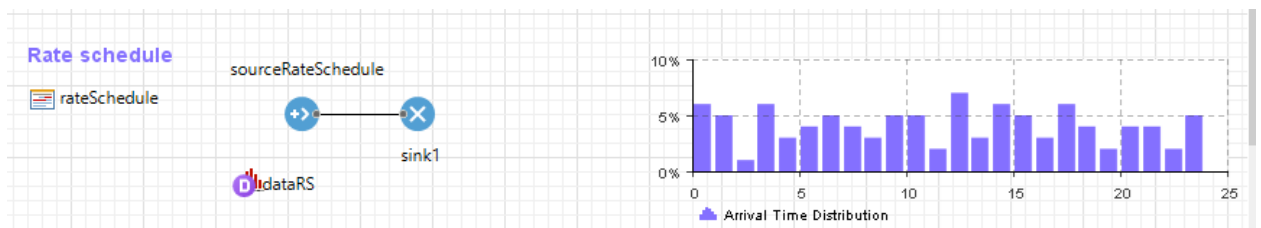


Рисунок 1.3 – Модель прибытия агентов (вариант 4)

Вариант 5. Прибытие агентов, согласно расписанию прибытий (рисунок 1.4).

Для построения модели необходимо использовать следующие элементы:

1. Текстовые блоки (Палитра->Презентация) – «Arrival schedule», «Timetable», «Last arrival», «Next arrival».
2. Расписание прибытий (arrivalSchedule) (Палитра->Агент).
3. Блок Source (sourceArrivalShedule). Настройки параметров блока: Прибывает согласно – расписанию прибытий, Расписание прибытий – arrivalSchedule.
4. Блок Sink (sink2).
5. Прямоугольник (rectangle). Используется как обрамление выводимой текстовой информации. (Палитра->Презентация).

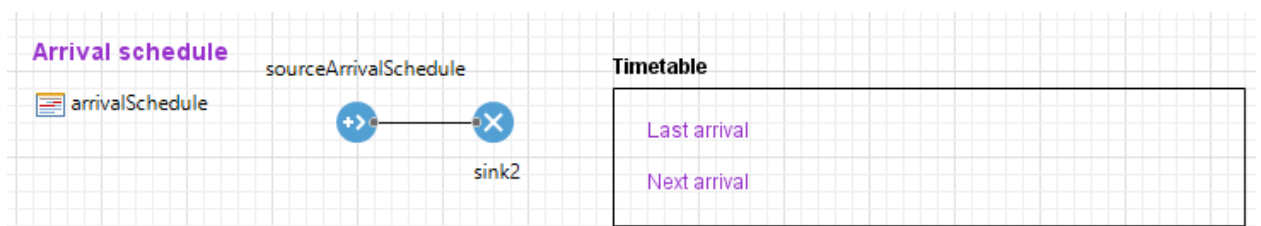


Рисунок 1.4 – Модель прибытия агентов (вариант 5)

Контрольные вопросы

1. Какие блоки используются для построения простейшей системы массового обслуживания?
2. Какие основные функции блока Source?
3. Какие основные функции блока Sink?
4. Какие основные функции блока Delay?
5. Какие основные функции блока Queue?

Лабораторная работа №2.

Построение дискретно-событийных моделей.

Модели прибытия с очередью. Блок Queue

Моделирование очереди

Блок Queue моделирует очередь агентов, ожидающих обработки блоками, следующими за ним. Поступающие агенты помещаются в очередь в определенном порядке:

- согласно правилу FIFO (в порядке поступления в очередь, «First In – First Out» – «Первым пришел – первым ушел»);
- согласно правилу LIFO (обратный порядок обслуживания, «Last In – First Out» – «последним пришел – первым ушел»);
- согласно приоритету агентов. Приоритет может либо явно задаваться для агента, либо вычисляться согласно свойствам агента и каким-то внешним условиям. Очередь с приоритетами всегда примет нового входящего агента, вычислит его приоритет и поместит его в очередь в позицию, соответствующую его приоритету.

Если очередь не имеет максимальную вместимость и полностью заполнена, то блок Queue можно настроить так, что приход нового агента вынудит последнего хранящегося в очереди агента покинуть объект через порт outPreempted. Если для блока Queue настроена возможность выхода по таймауту, то агент может покинуть очередь через специальный порт outTimeout, если проведет в очереди заданное количество времени.

Для блока Queue можно настроить действия, которые будут выполняться при входе, при подходе к выходу, при выходе (через стандартный порт), при уходе по таймауту (через специальный порт outTimeout), при вытеснении (через порт outPreempted), при извлечении (с помощью функции remove()). Во всех действиях актуальный агент доступен как локальная переменная agent.

Параметры блока Queue

- *Вместимость* – количество агентов, которые одновременно могут находиться в очереди.

Синтаксис: int capacity.

Максимальная вместимость – если опция выбрана (true), то вместимость очереди будет максимально возможной (ограничена константой Integer.MAX_VALUE).

Синтаксис: boolean maximumCapacity.

- *Место агентов* – фигура разметки (узел или путь), где располагаются агенты, находящиеся в очереди.

Синтаксис: AnimationStaticLocationProvider entityLocation

- *Очередь* – порядок поступления в очередь (FIFO, LIFO, По приоритету, Сравнение агентов).

Имя: queuing.

Допустимые значения: Queue.QUEUEING_FIFO – FIFO

Queue.QUEUEING_PRIORITY – по приоритету

Queue.QUEUEING_COMPARISON – сравнение агентов

Queue.QUEUEING_LIFO – LIFO

- *Разрешить уход по таймауту* – если опция выбрана (true), то агенты могут покидать очередь по таймауту.

Синтаксис: boolean enableTimeout.

Основные функции блока Queue

- int size() – возвращает количество агентов, находящихся в данный момент в очереди.

- T get(int index) – возвращает агента, находящегося в позиции с номером index (ближайшая к выходу из очереди позиция имеет номер 0).

- T remove(Agent agent) – извлекает агента agent из очереди и возвращает его. Если такого агента в очереди обнаружено не будет, метод вернет null.

- boolean release(T agent) – сообщает очереди отпустить данного агента и направить его к выходному порту.

- void sortAgents() – перераспределяет агентов в очереди, сортируя их соответственно правилам очереди.

- void resetStats() – удаляет статистику, собранную объектом к текущему моменту времени.

Агенты в очереди могут отображаться на презентации двумя способами: последовательно – один за другим (тип анимации path) или в произвольном порядке – в отдельных узлах (node).

Рассмотрим различные варианты нахождения агентов в очереди.

Задание

Вариант 1. Модель очереди с различными правилами обслуживания (рисунок 2).

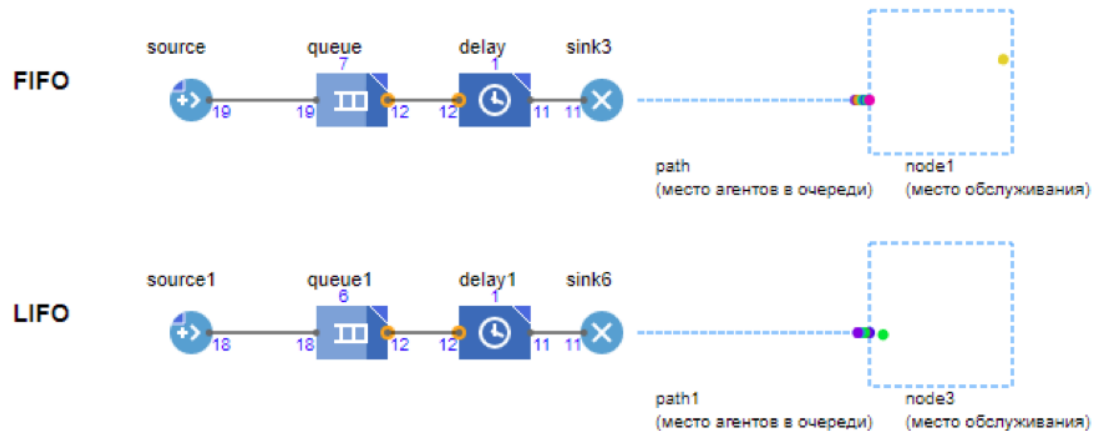


Рисунок 2 – Имитация очереди с различным порядком обслуживания

Моделируется две очереди с различными правилами обслуживания – FIFO и LIFO (параметр *Очередь* блока Queue). В очередь с порядком FIFO заявка помещается в конец и обслуживается после тех заявок, которые вошли в систему ранее. В очередь с порядком LIFO заявка помещается в начало и обслуживается раньше тех заявок, которые вошли в систему ранее. Как видно из прогона модели, есть вероятность того, что заявка, вошедшая первой, никогда не будет обслужена, если порядок обслуживания очереди установлен как LIFO.

Вариант 2. Модель очереди с возможностью выхода по таймауту (Рисунок 2.1).

Для добавления в модель с очередью LIFO разрешения ухода из системы по таймауту необходимо для блока Queue установить возможность выхода из блока через специальный порт outTimeout (параметр – *Разрешить уход по таймауту*). Также необходимо соединить данный порт outTimeout блока Queue с блоком, в который будут попадать вышедшие агенты.

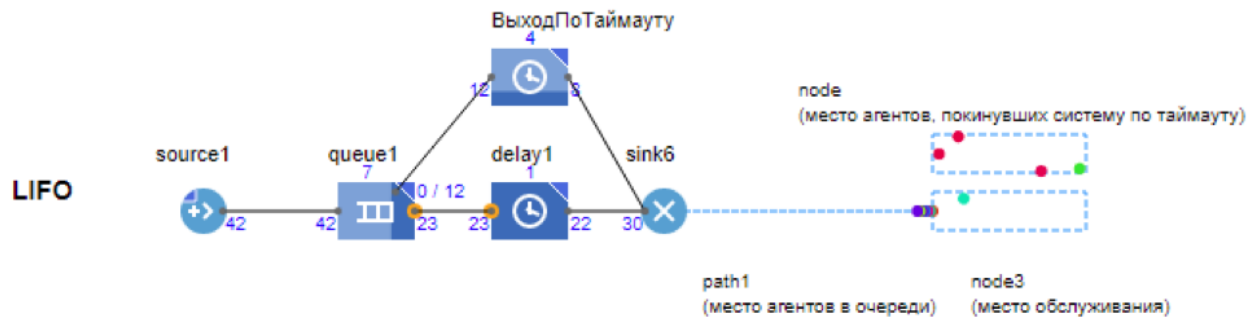


Рисунок 2.1 – Имитация очереди с разрешением ухода из системы по таймауту

Вариант 3. Сбор статистики блока Queue (рисунок 2.2).

Для построения модели необходимо использовать следующие элементы:

1. Блок Source (source). Настройки параметров блока: Прибывает согласно – Интенсивности, 1 в час.
2. Блок Queue (queue). Вместимость – 100. Место агентов – path. Очередь – FIFO.
3. Блок Delay (delay). Время задержки – 2 ч. Вместимость – 1. Место агентов – node1.
4. Блок Sink (sink3).
5. Данные гистограммы (data). Предназначен для формирования набора данных. Значение – queue.size().
6. Гистограмма (chart1). Данные – data.

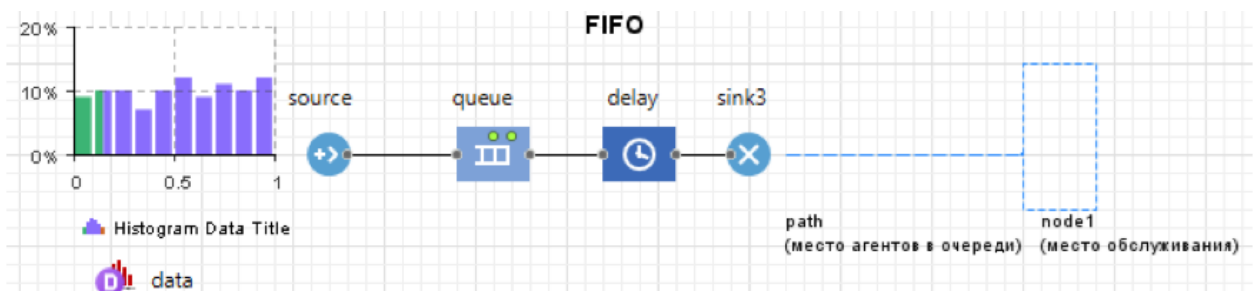


Рисунок 2.2 – Сбор статистики блока Queue

Контрольные вопросы

1. Опишите порядок поступления в очередь по принципу «First In – First Out».
2. Опишите обратный порядок обслуживания по принципу «Last In – First Out».
3. Параметры блока Queue.
4. Основные функции блока Queue.

Лабораторная работа №3.

Построение дискретно-событийных моделей. Блок Delay

Моделирование обслуживания агентов в системе

Простейшим блоком, осуществляющим обслуживание агентов в системе, является блок Delay, который задерживает агентов на заданный период времени. Время задержки может быть постоянной величиной, вычисляться динамически, изменяться по закону распределения случайной величины, зависеть от текущего агента или от каких-то других условий.

Объектом Delay могут быть одновременно задержаны несколько агентов.

Объект Delay может отображать анимации находящихся в нем агентов как движущимися вдоль заданного пути, так и ожидающими в заданных точках. В том случае, если агенты отображаются движущимися, за время их задержки они должны будут пройти весь путь выбранной фигуры анимации объекта Delay.

Параметры блока Delay

- Тип задержки – определяет способ задания времени задержки: задержка может быть задана либо как *Определенное время*, либо *До вызова функции stopDelay()*.

Имя: type.

Возможные значения: Определенное время – Delay.TIMEOUT

До вызова функции stopDelay() – Delay.MANUAL

- Время задержки – выражение, вычисляющее время задержки для агента.

Тип значения: double.

Локальная переменная: T entity – текущий агент.

- Вместимость – вместимость объекта Delay. Задаёт максимальное количество агентов, которые могут одновременно находиться в объекте.

Синтаксис: int capacity.

Значение по умолчанию: 1.

- Максимальная вместимость – если опция выбрана (true), то вместимость объекта Delay будет максимально возможной (ограничена константой Integer.MAX_VALUE).

Синтаксис: boolean maximumCapacity.

Значение по умолчанию: false.

Основные функции блока Delay

- void suspend() – приостанавливает работу блока Delay;
- delay.suspend() – приостанавливает работу блока до вызова функции resume();
- void resume() – возобновляет приостановленный ранее объект Delay;
- boolean isSuspended() – возвращает статус приостановленного блока Delay: true, если блок Delay приостановлен, false – если нет.
- double getRemainingTime(T agent) – возвращает оставшееся время обслуживания для данного агента;
- void extendDelay(Agent agent, double dt) – увеличивает время обслуживания для данного агента;
- double getElapsedTime(T agent) – возвращает время, проведенное данным агентом в блоке Delay.

Рассмотрим различные варианты обслуживания агентов.

Задание

Вариант 1. Моделирование обслуживания заявок в системе с динамически изменяющимся количеством одновременно обслуживаемых агентов (рисунок 3).

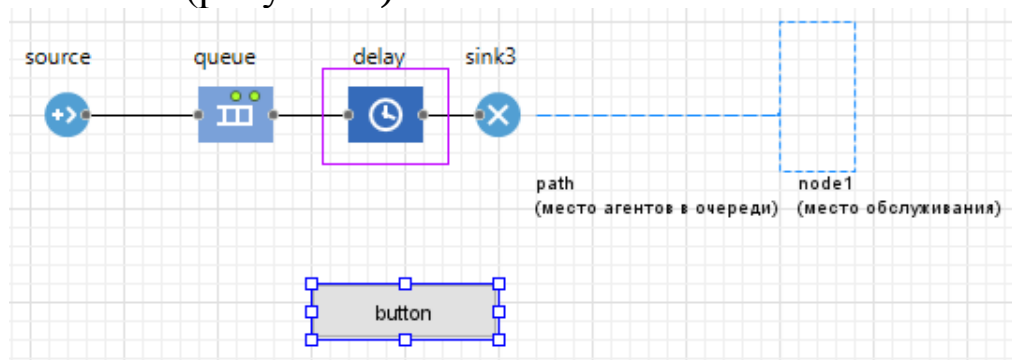


Рисунок 3 – Моделирование обслуживания агентов (вариант 1)

Для построения модели необходимо использовать следующие элементы:

1. Блок Source (source).
2. Блок Queue (queue). Вместимость – максимальная вместимость.

3. Блок Delay (delay). Время задержки – 2 ч. Вместимость –1. Место агентов – node1.
4. Блок Sink (sink3).
5. Кнопка (button). Действие – delay.capacity=3.

Вариант 2. Моделирование обслуживания заявок в системе с динамически изменяющимся количеством одновременно обслуживаемых агентов. Количество меняется с помощью переключателя (рисунок 3.1).

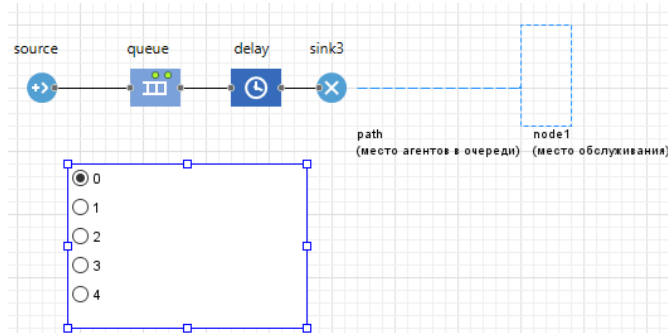


Рисунок 3.1 – Моделирование обслуживания агентов (вариант 2)

У объекта *Переключатель* есть возможность связывания с другим объектом. При этом выбирается параметр, в который будут подставляться значения элементов объекта *Переключатель* (Рисунок 22).

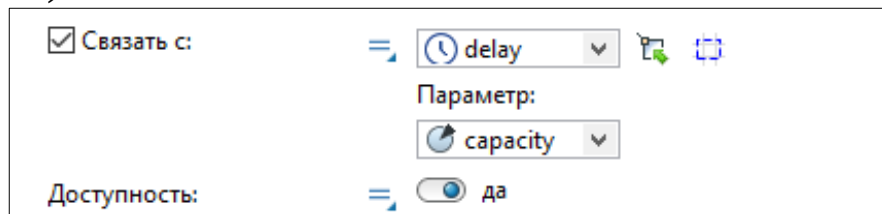


Рисунок 3.2 – Связывание элемента управления с параметром объекта

Контрольные вопросы

1. Опишите назначение и основные свойства блока Source.
2. Опишите назначение и основные свойства блока Queue.
3. Опишите назначение и основные свойства блока Delay.
4. Каким образом можно собрать статистику работы блоков?

Лабораторная работа №4.

Параметры агентов. Сортировка агентов в зависимости от значения параметра

Агентное моделирование

В агентном моделировании процесс функционирования модели представляется как результат взаимодействия агентов. *Агент* – это класс с заданным поведением. Поведение класса задается конечным автоматом, где состояния агента являются состояниями автомата, а переход между состояниями задается по какому-либо условию. Также поведение агента может быть задано обычными для объектно-ориентированного подхода средствами: событиями, функциями и пр.

Агенты помещаются в среду, в которой устанавливаются связи между агентами, что позволяет агентам пересылать сообщения между собой. При получении сообщения агент может переходить в новое состояние, выполняться какое-либо событие агента и т. д. Среда жизни агента может быть дискретной и непрерывной. Дискретная среда подобна шахматной доске – агент может находиться строго в заданных точках среды. Непрерывная среда аналогична реальному миру – агент может быть в любой ее точке.

При создании новой модели AnyLogic автоматически создаст тип агента Main и простой эксперимент Simulation. При этом в центре рабочего пространства будет расположен графический редактор, который отображает диаграмму типа агента Main.

Для создания новых агентов необходимо в палитре «*Агент*» выбрать соответствующий элемент и перенести его в графический редактор. При этом последовательно откроется ряд окон (рисунки 4–4.3).

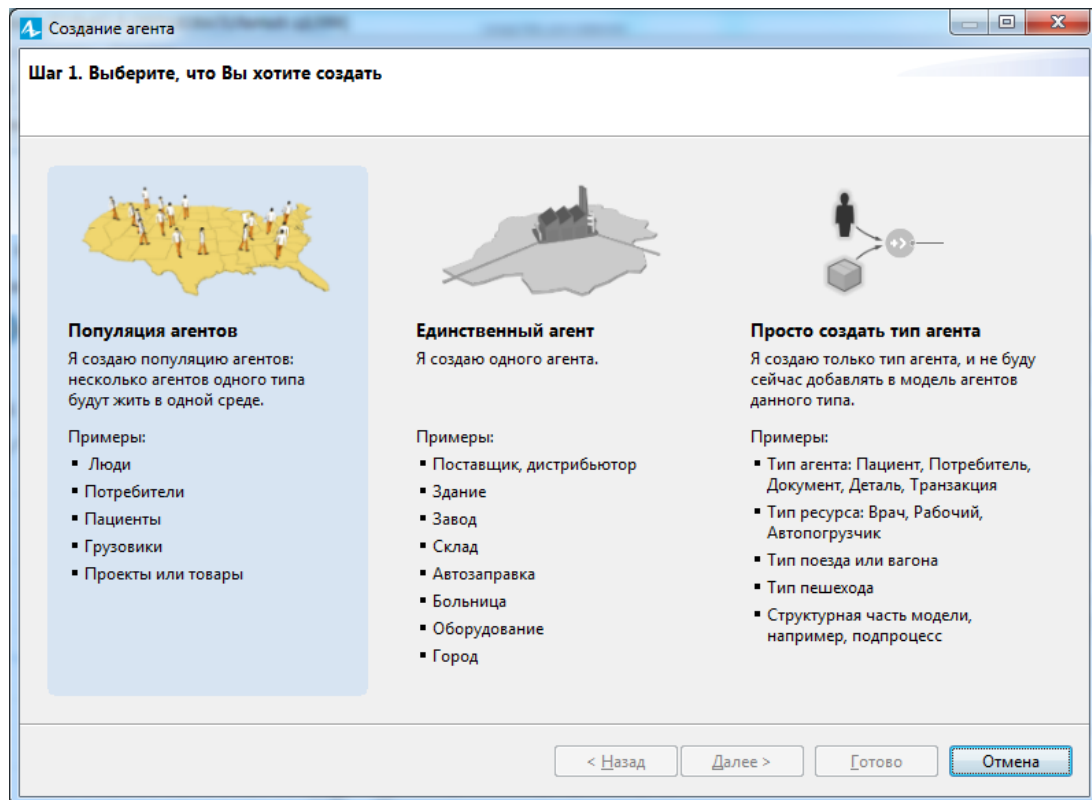


Рисунок 4 – Создание агента. Шаг 1

Шаг 2. Создание нового типа агента

Имя нового типа:

Имя популяции:

☒ Создать новый тип агента "с нуля"

☐ Использовать таблицу базы данных
Я хочу считать значения параметров агентов из базы данных

☐ Агент будет использоваться в диаграммах процессов

< Назад Далее > Готово Отмена

Рисунок 4.1 – Создание агента. Шаг 2

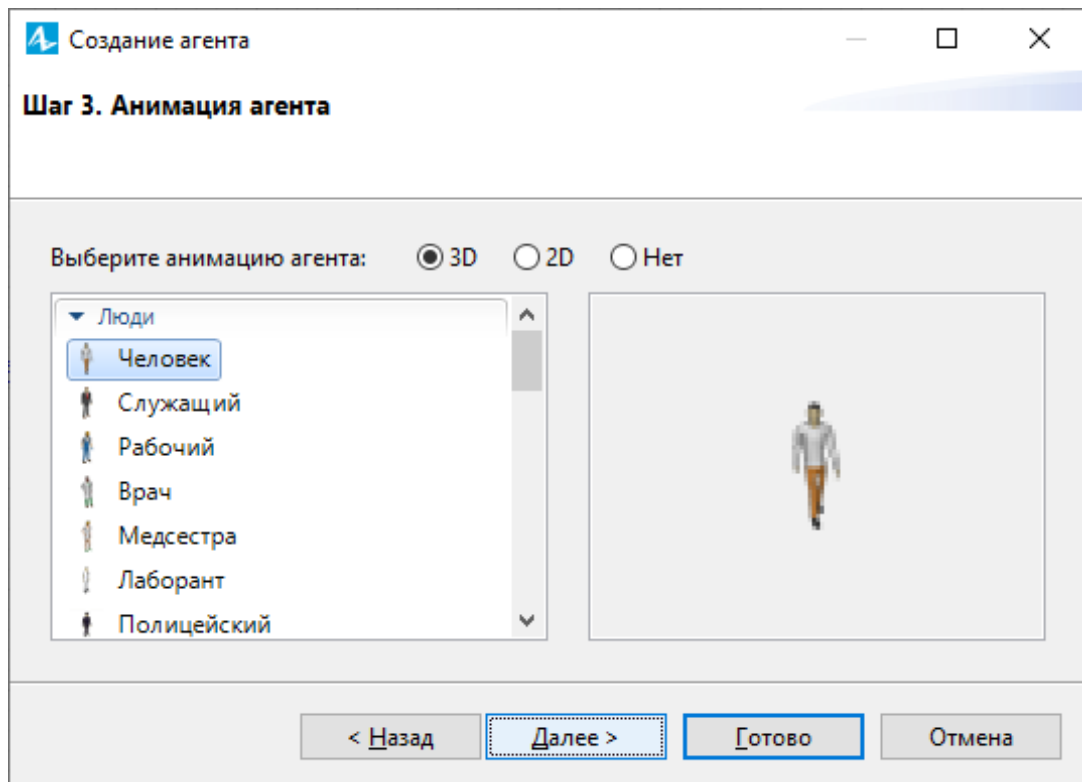


Рисунок 4.2 – Создание агента. Шаг 3

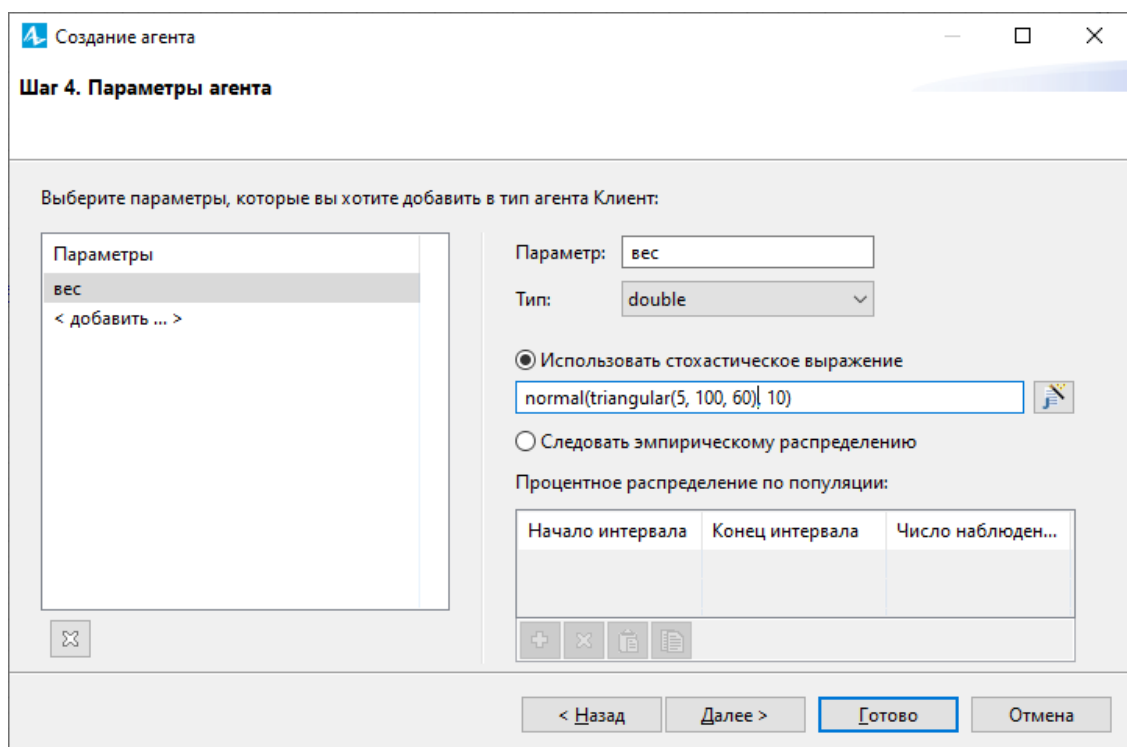


Рисунок 4.3 – Создание агента. Шаг 4

Доступны три варианта создания популяции:

1. *Изначально пуста* – опция используется, если необходимо создавать агентов популяции динамически при запуске или во время выполнения модели.

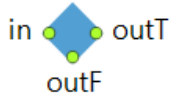
2. *Содержит заданное количество агентов* – опция используется, если необходимо создавать популяцию, содержащую известное начальное количество агентов.
3. *Загружается из базы данных* – опция используется, если данные об агентах хранятся в базе данных и необходимо создать популяцию агентов на основе данных из таблицы базы данных.

Получить доступ к определенному агенту популяции можно с помощью метода `get()`: `имя_популяции.get(номер_агента)`, где `номер_агента` – индекс агента в популяции (нумерация начинается с 0). Например: `клиенты.get(0)`. Номер последнего агента можно получить как `клиенты.get(size() – 1)`.

Получить доступ к параметру агента популяции можно с помощью выражения вида: `agent.имя_параметра = значение`;
Например: `agent.вес = 67`.

Можно напрямую указать тип агента, путем приведения типов Java:

`(Имя_типа_агента)agent.имя_параметра = значение`;
Например: `((Клиент)agent).вес=67`;

Для перенаправления агентов по различным путям используется блок `SelectOutput` . Он направляет входящих агентов в один из двух выходных портов в зависимости от выполнения заданного (детерминистического или заданного с помощью вероятностей) условия. Условие может зависеть как от агента, так и от каких-то внешних факторов.

Может использоваться для сортировки агентов согласно заданному критерию, для случайного разделения потока агентов на части и т. д. Иногда требуется иметь более двух выходов, тогда необходимо использовать комбинацию блоков `SelectOutput5`, `SelectOutputIn` и `SelectOutputOut`.

Параметры блока `SelectOutput`

- *Выход true выбирается*. Назначение: определяет, как будет производиться маршрутизация агентов (случайно, с заданной вероятностью, при выполнении условия).

Синтаксис: `boolean conditionIsProbabilistic`.

Значение по умолчанию: С заданной вероятностью (`true`).

- *Вероятность*. Назначение: выражение, вычисляющее вероятность того, что текущий агент покинет объект через порт outT.

Значение по умолчанию: 0.5 – не зависит от поступающих агентов и просто разделяет поток агентов на две равные части.

- *Условие*. Назначение: условие, вычисляемое для входящего агента. Если оно выполняется (равно true), то агент покидает объект через порт outT, если нет – через порт outF.

Для перемещения агентов по пути можно использовать блок

Conveyor  , который перемещает агентов по пути заданной длины с заданной скоростью, сохраняя их порядок и оставляя заданные промежутки между ними.

Рассмотрим различные варианты обслуживания агентов в зависимости от типа агента.

Задание

Вариант 1. Сортировка агентов в зависимости от типа агента. Агенты относятся к одной популяции (рисунок 4.4).

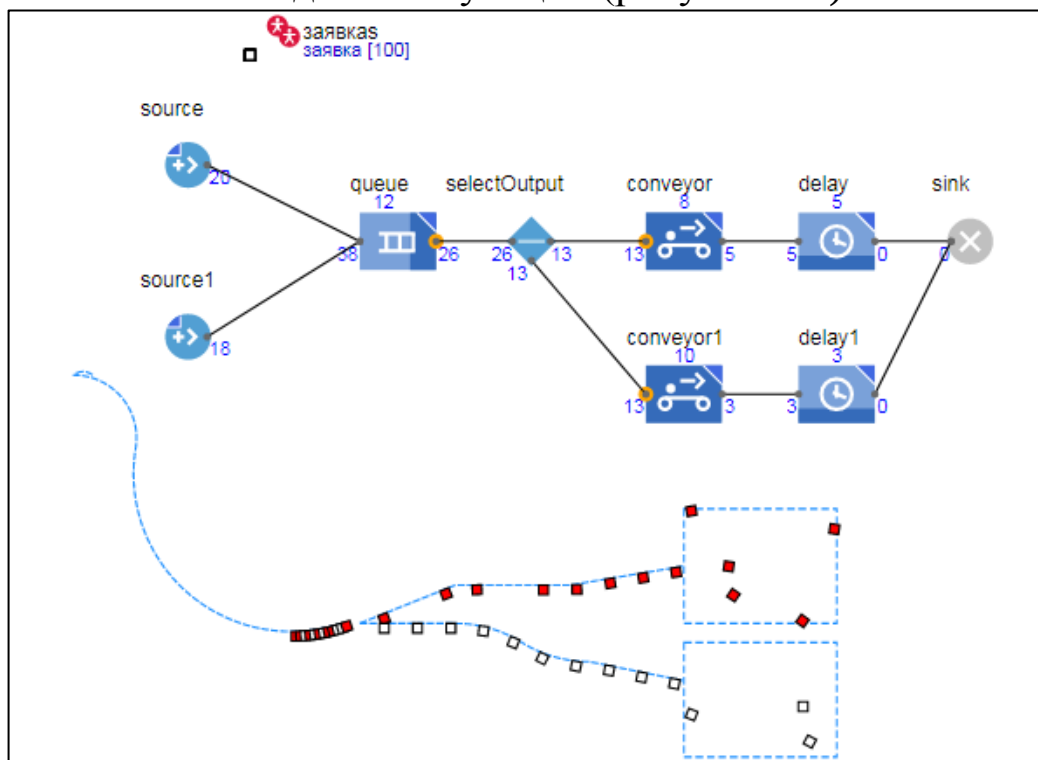


Рисунок 4.4 – Сортировка агентов одной популяции

Этапы создания модели:

1. Создается популяция агентов заявка с фигурой анимации *rectangle* и параметром *int min*.

2. Для блока source устанавливаем значение параметра Новый агент – заявка, действие при выходе – `agent.тип = 0`.

3. Для блока source1 устанавливаем значение параметра Новый агент – заявка, действие при выходе – `agent.тип = 1`; `agent.rectangle.setFillColor(red)`.

4. Для блока selectOutput прописываем условие выхода через порт outT: `agent.тип == 0`.

Вариант 2. Сортировка агентов в зависимости от типа агента. Агенты относятся к разным популяциям (рисунок 4.5).

Для проверки принадлежности объекта к определенному подклассу используется оператор `instanceof` с синтаксисом: `<объект> instanceof <класс>`. Далее:

1. Создаются две популяции агентов: популяция заявка с фигурой анимации *rectangle* и популяция *новаяЗаявка* с фигурой анимации *person*.

2. Для блока selectOutput прописываем условие выхода через порт outT: `agent instanceof заявка`.

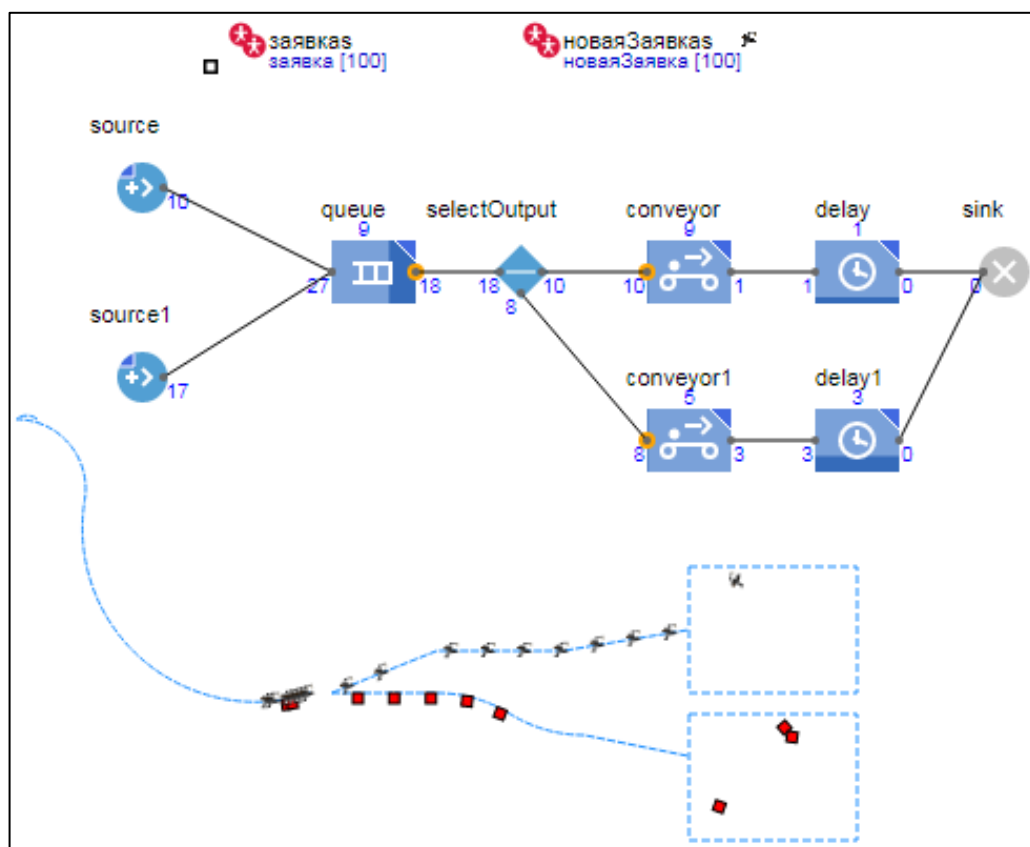


Рисунок 4.5 – Сортировка агентов разных популяций

Контрольные вопросы

1. Опишите этапы создания агентов пользовательского типа.
2. Опишите сортировку агентов, принадлежащих одной популяции.
3. Опишите сортировку агентов, принадлежащих разным популяциям.
4. Опишите назначение блока selectOutput.

Лабораторная работа №5. Визуализация дискретно-событийной модели

Создание анимации модели. Добавление фигур разметки пространства

1. Рисуем точечный узел, обозначающий устройство обслуживания (например, банкомат). Вначале открываем палитру **Разметка пространства** панели **Палитра**.

Перетаскиваем элемент **Точечный узел** из палитры **Разметка пространства** в графический редактор и помещаем его под блок-схемой процесса (рисунок 5).

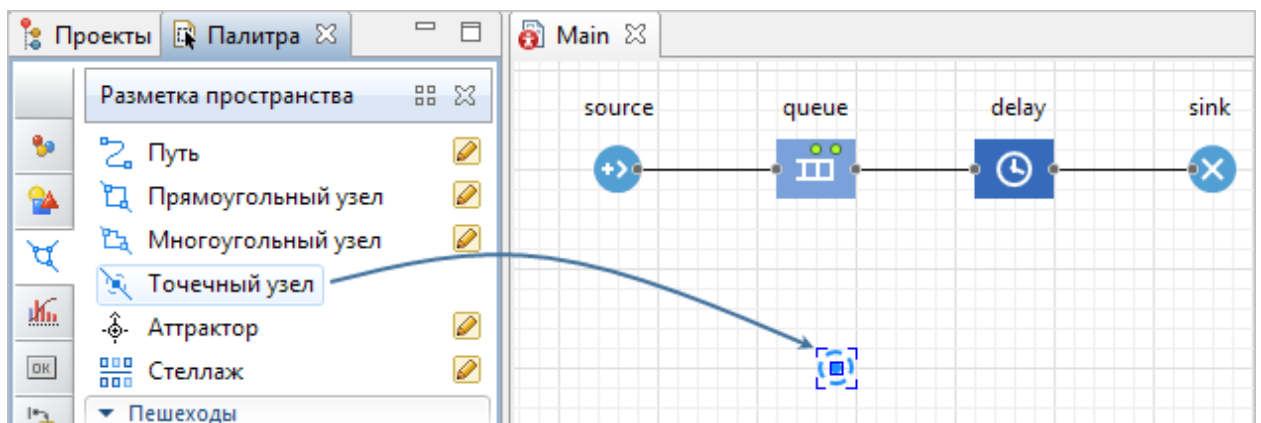


Рисунок 5 – Разметка пространства

2. Выделяем щелчком блок *Delay*, переименованный в *ATM*, чтобы открыть его свойства. Далее выберите точечный узел *point*, который нарисовали в параметре **Место агентов** (рисунок 5.1).

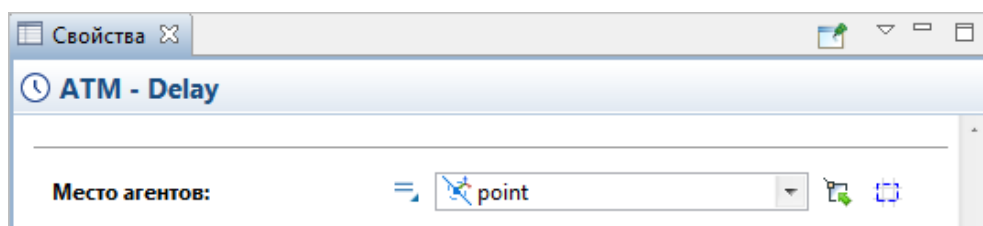


Рисунок 5.1 – Свойства блока *Delay*

Задание фигуры анимации очереди

1. Рисуем путь, обозначающий очередь к банкомату. Открываем палитру **Разметка пространства** панели **Палитра**.

2. Двойным щелчком выделяем элемент **Путь** палитры **Разметка пространства**, чтобы перейти в *режим рисования*.

Теперь можете рисовать путь точка за точкой, последовательно щелкая мышью в тех точках диаграммы, куда хотите поместить вершины линии. Чтобы завершить рисование, добавьте последнюю точку пути двойным щелчком мыши (рисунок 5.2).

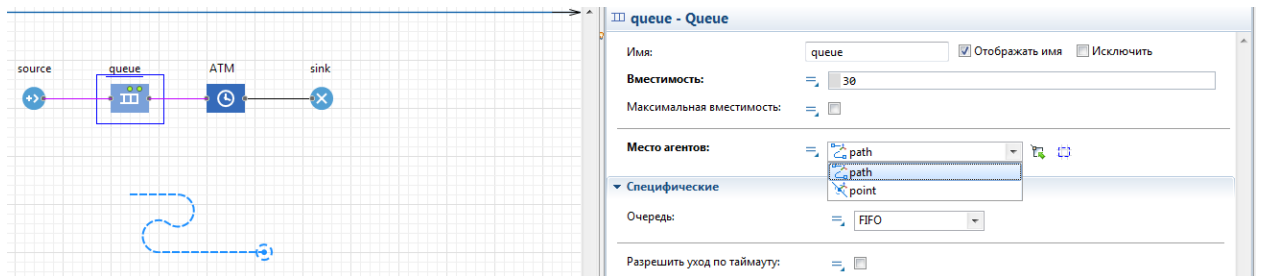


Рисунок 5.2 – Разметка пространства

3. Теперь можно запустить модель и изучить ее поведение. Для ускорения работы модели, переключитесь в режим виртуального времени, щелкнув мышью по кнопке панели инструментов **Реальное/виртуальное время**. В режиме виртуального времени модель будет выполняться максимально быстро, без привязки модельного времени к реальному.

Запуск модели с простейшей анимацией

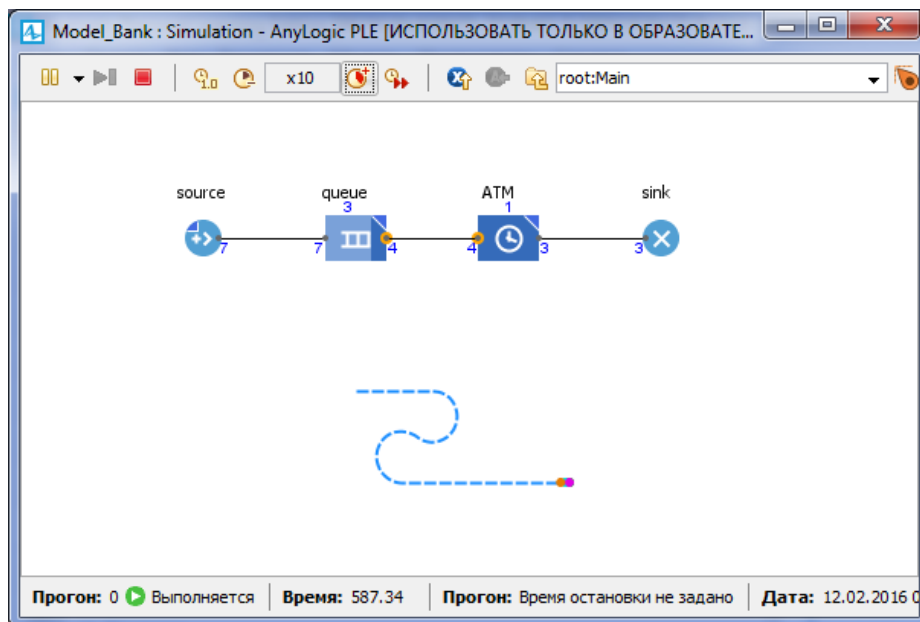


Рисунок 5.3 – Запуск модели

Примечание: Чтобы при имитации агенты не отображались слишком близко друг к другу необходимо растянуть элемент **Путь**.

Добавление 3D-анимации

Добавляем на диаграмму элемент 3D-Окно. **3D-Окно** используется для задания на диаграмме области агента, в которой во время запуска модели будет отображаться трехмерная анимация этой модели.

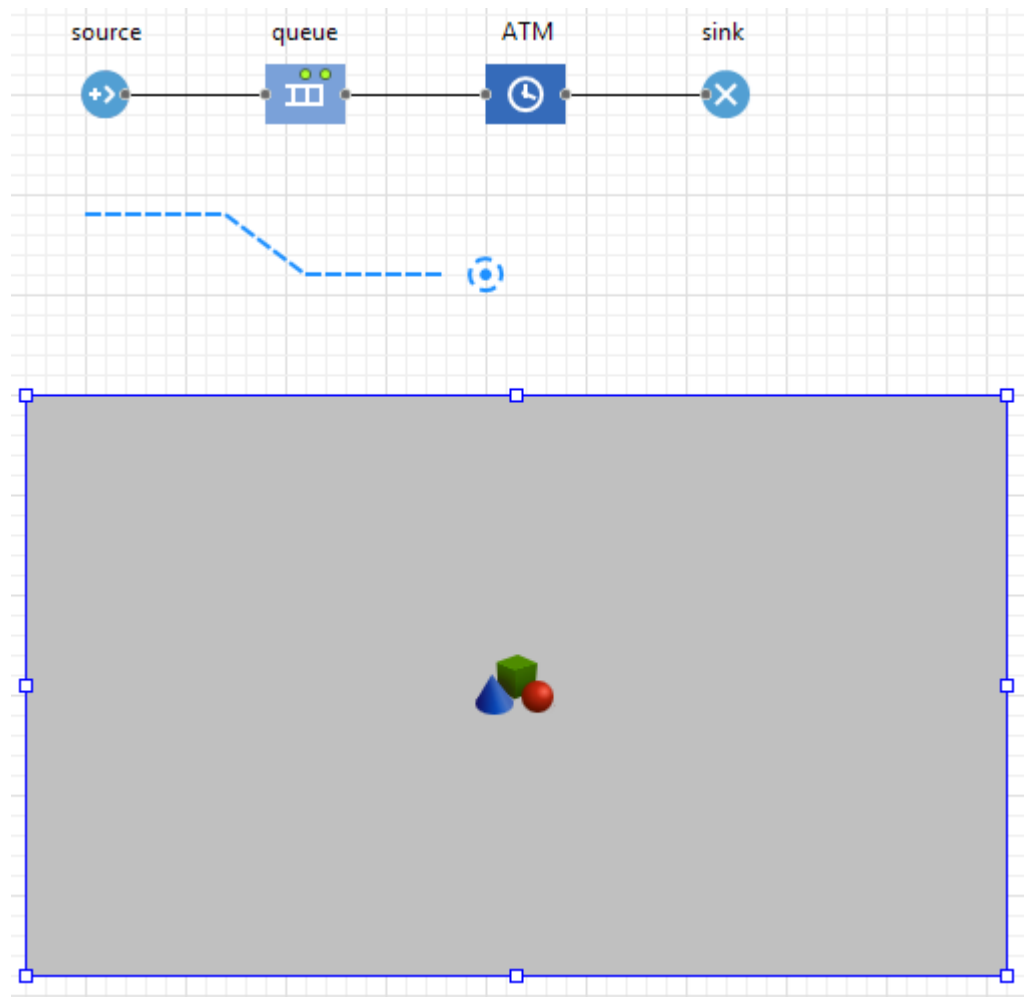


Рисунок 5.4 – 3D Окно

Запуск 3D-анимации

Запустите модель и опробуйте навигацию по сцене трехмерной анимации.

Щелкните кнопку панели инструментов **Показать область...** и выберите [window3D].

Таблица 5 – Действия трехмерной модели

Чтобы	Выполните следующие действия
Переместить сцену	1. Нажмите левую кнопку мыши в области 3D окна и держите ее нажатой. 2. Передвиньте мышь в направлении перемещения
Повернуть сцену	1. Нажмите клавишу Alt и держите ее нажатой. 2. Нажмите левую кнопку мыши в области 3D окна и держите ее нажатой. 3. Передвиньте мышь в направлении вращения
Приблизить/отдалить	1. Покрутите колесо мыши от/на себя в области 3D-окна

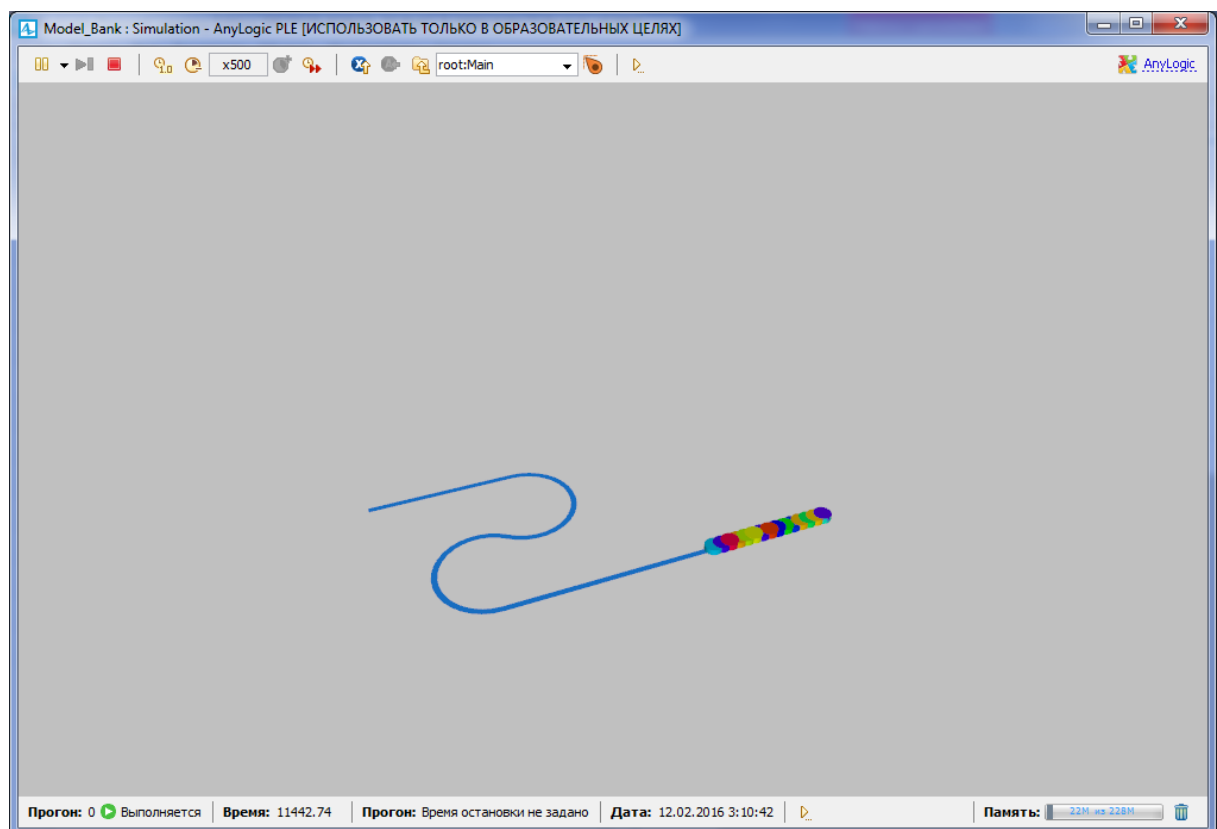


Рисунок 5.5 – 3D-анимация

Добавление 3D-объектов

По умолчанию клиенты в модели обозначаются цветными точками и отображаются в виде цветных цилиндров в 3D-анимации. Если необходимо задать нестандартный тип клиента и выбрать для него фигуру анимации, нужно создать новый тип агента.

Для этого откройте **Библиотеку моделирования процессов** в панели **Палитра**. Перетащите элемент **Тип агента** в графический редактор.

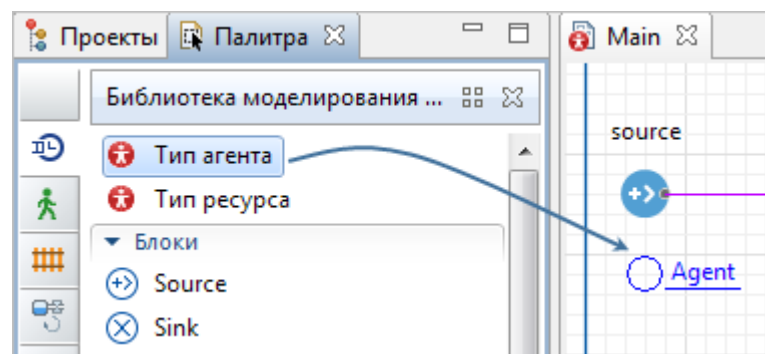


Рисунок 5.6 – Добавление агента

Откроется диалоговое окно Мастера создания агентов на шаге **Анимация агента**. Введите *Customer* в поле **Имя нового типа**, выберите опцию **3D** для типа анимации и фигуру анимации *Человек* из списка 3D-фигур.

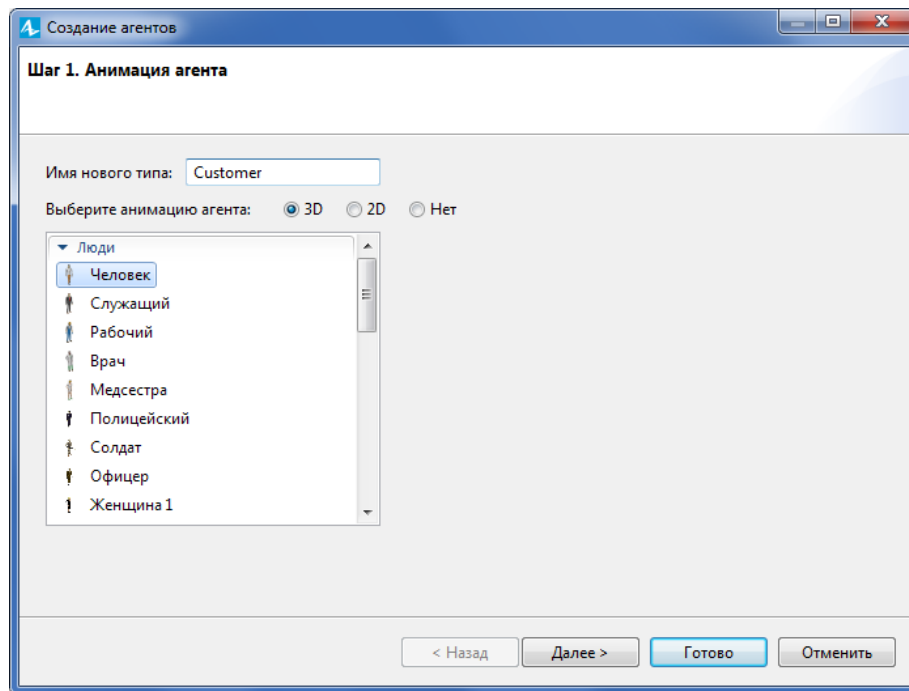


Рисунок 5.7 – Создание агентов
**Настройка использования нового типа агентов
 в блок-схеме**

На диаграмме Main, выделите блок *Source* в графическом редакторе.

Выберите тип агента *Customer* в выпадающем списке параметра **Новый агент** (рисунок 5.8).

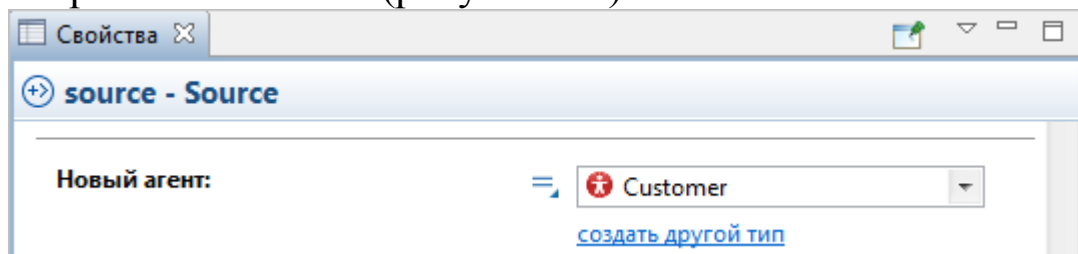


Рисунок 5.8 – Свойства *Source*

Запустите модель, чтобы увидеть анимацию клиентов в очереди (рисунок 5.9).

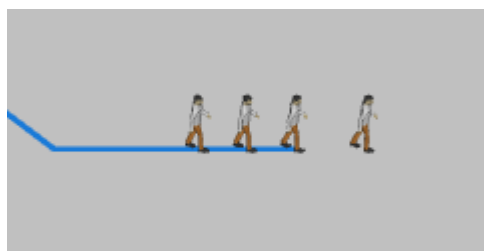


Рисунок 5.9 – Анимация

Добавьте объект «Банкомат».

Откройте палитру **3D-Объекты** в панели **Палитра**.

Перетащите 3D-фигуру **Банкомат** из секции палитры **Супермаркет** в графический редактор и поместите ее на точечный узел (рисунок 5.10).

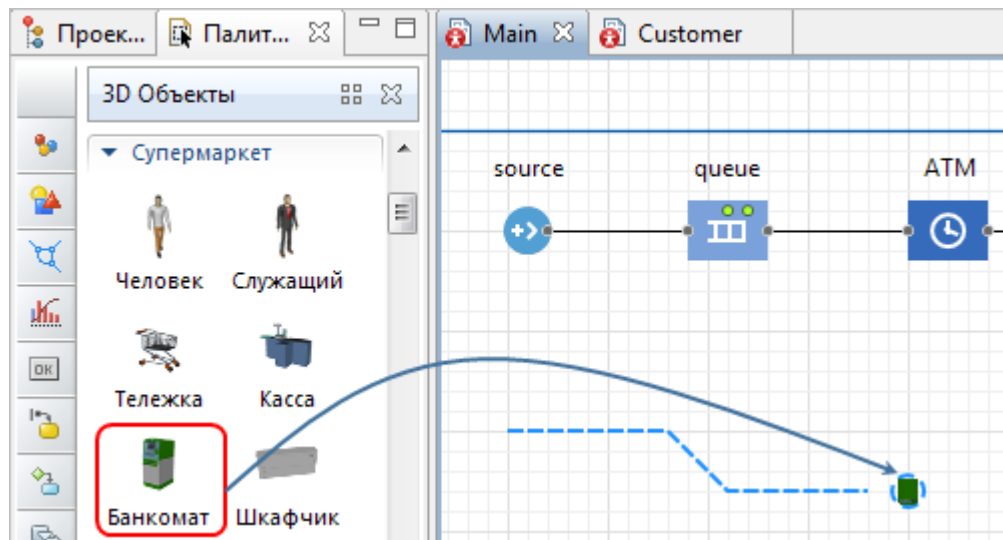


Рисунок 5.10 – Добавление «Банкомат»

Разворот 3D-объектов

Если устройство обслуживания стоит не той стороной по направлению к очереди клиентов, то необходимо развернуть его в правильную сторону.

Выделите 3D-объект банкомата АТМ в графическом редакторе и откройте секцию свойств **Расположение**.

Выберите из выпадающего списка параметра **Поворот Z** 0 градусов.

Запустите модель, чтобы убедиться, что фигура банкомата стоит «лицом» к клиентам (рисунок 5.11).

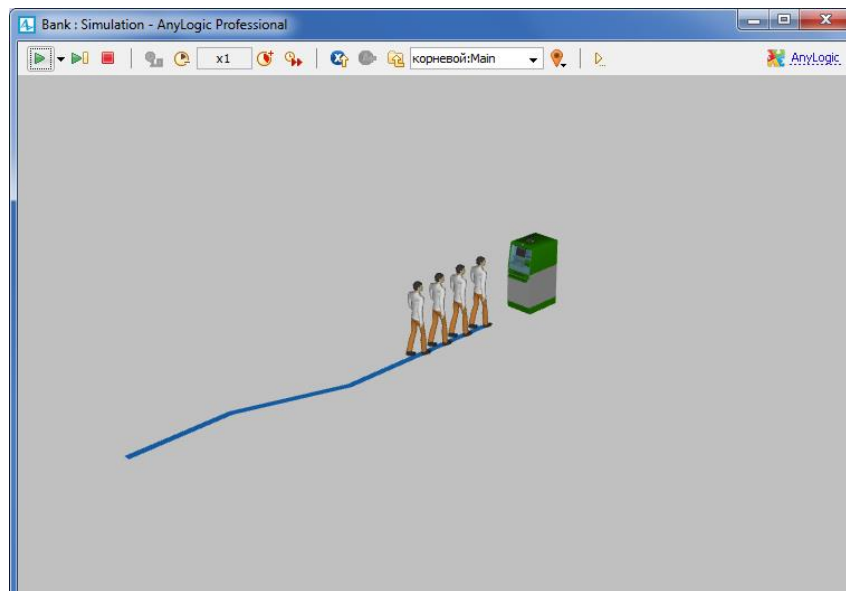


Рисунок 5.11 – 3D-объекты

Добавление камеры

Для отображения наиболее удачного ракурса моделируемого процесса используется 3D-объект **Камера** (рисунок 5.12).

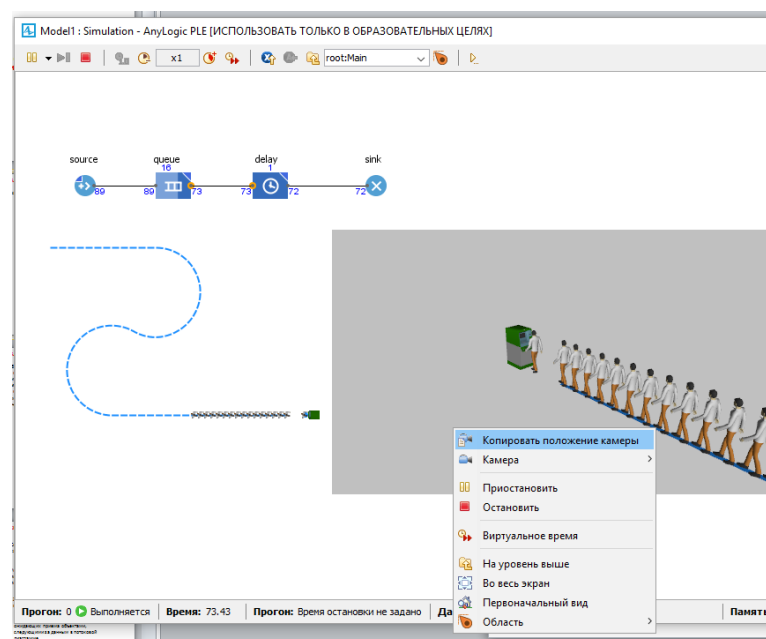


Рисунок 5.12 – 3D-объект Камера

1. При имитации копируется наиболее удачный ракурс с помощью опции контекстного меню **«Копировать положение камеры»**.

2. В графическом редакторе добавляется **Камера** (рисунок 5.13).

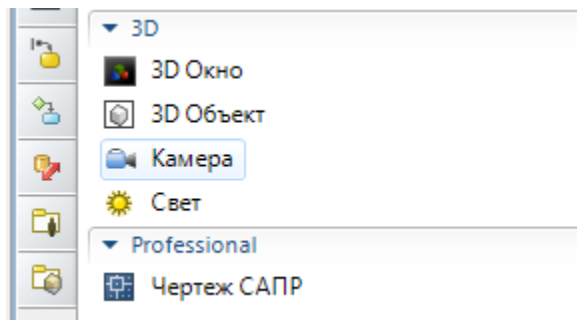


Рисунок 5.13 – Добавление объекта Камера

3. В свойствах камеры выбирается «Вставить координаты из Буфера обмена» (рисунок 5.14).

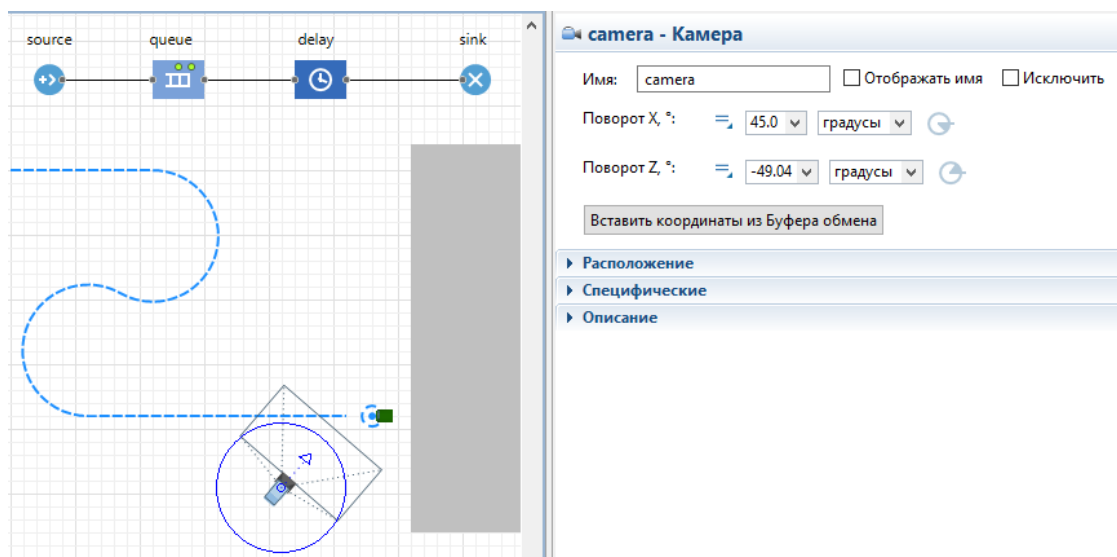


Рисунок 5.14 – Свойства объекта Камера

Добавление объекта Conveyor

Объект **Conveyor** моделирует конвейер или очередь. Перемещает заявки по пути заданной длины с заданной скоростью (одинаковой для всех заявок), сохраняя их порядок и оставляя заданные промежутки между ними (рисунок 5.15).

Конвейер может быть накапливающим и не накапливающим. Если заявка достигла конца конвейера, но не может его покинуть, то она там и останется. Не накапливающий конвейер в этом случае вообще прекратит продвижение заявок, в то время как накапливающий конвейер продолжит двигать заявки, которые имеют достаточно свободного места перед собой до тех пор, пока конвейер не будет заполнен заявками.

Объект **Conveyor** может использоваться не только для моделирования конвейеров в производственных моделях, но и для моделирования очереди пассажиров, автомобилей.

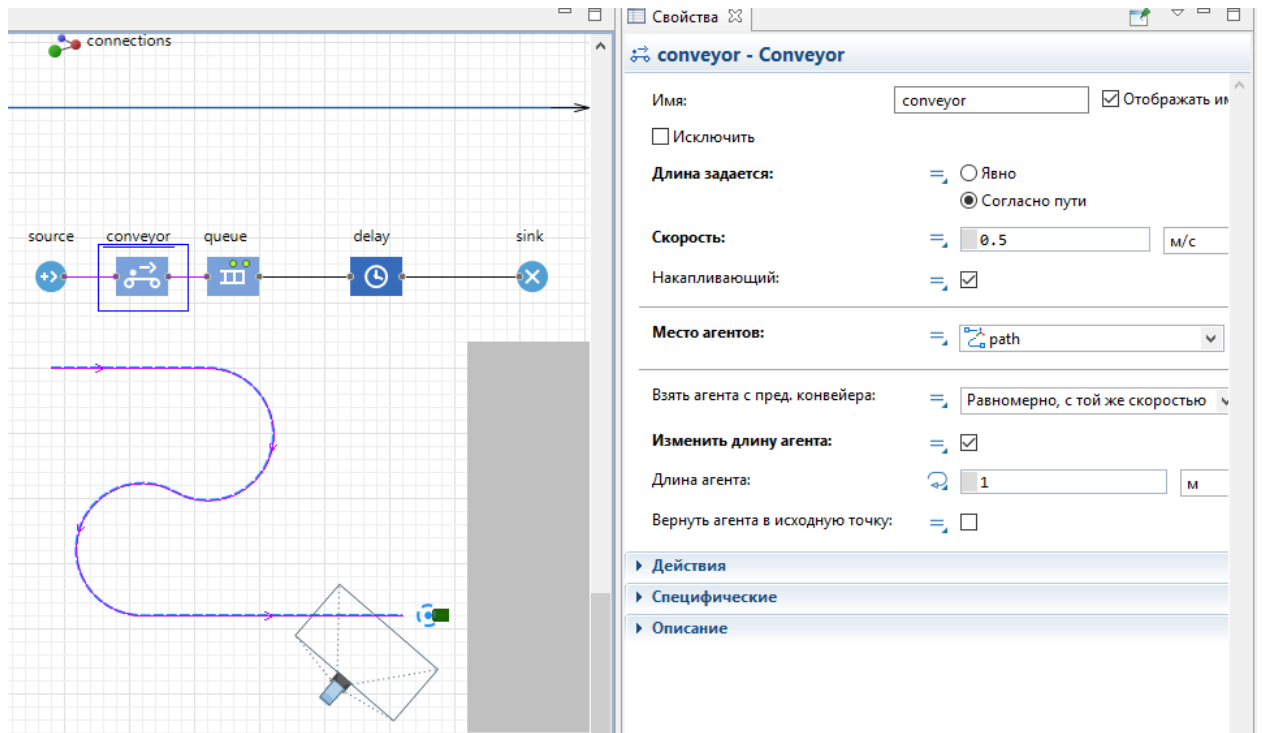


Рисунок 5.15 – Добавление объекта Conveyor

Примечание: Для моделирования (и анимации) независимого движения заявок, которые могут обгонять одна другую, лучше использовать объект **Delay**.

Анимация очереди

Если существует вероятность того, что две заявки могут поступить на вход конвейера с интервалом времени, меньшим, чем space/speed , то необходимо добавить перед объектом **Conveyor** специальный объект буферизации (или Буфер) – например, **Queue** (рисунок 5.16).

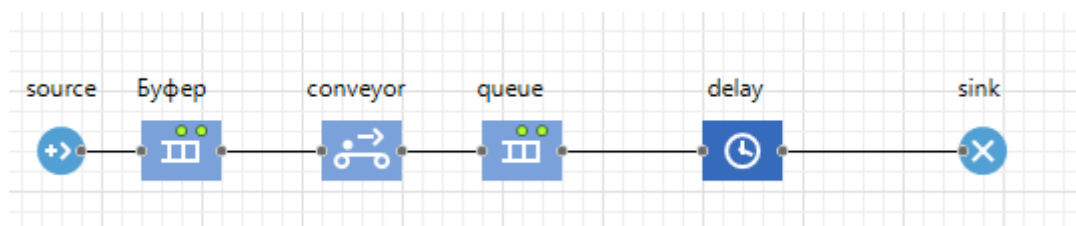


Рисунок 5.16 – Добавление объекта Буфер

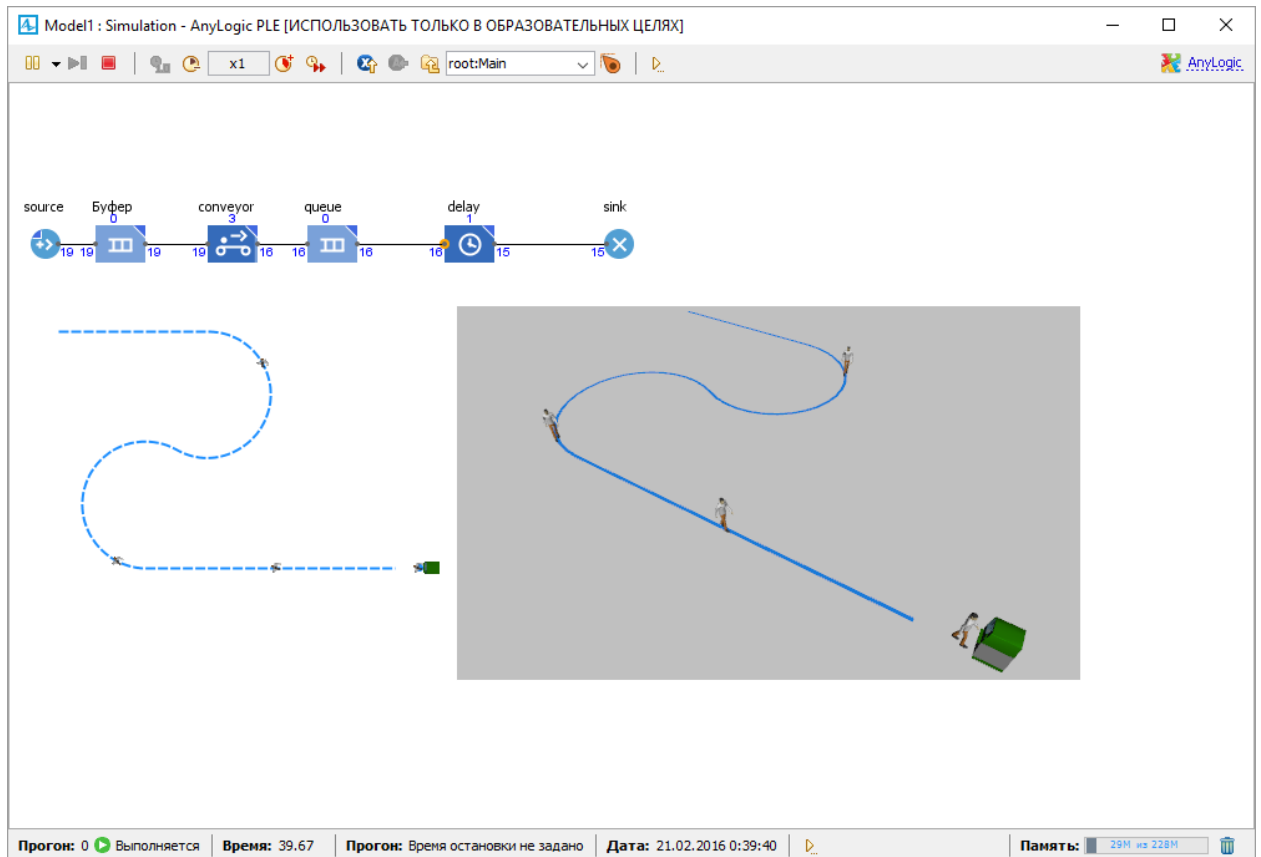


Рисунок 5.17 – Модель Conveyor

Задание

1. Смоделировать зону погрузки/разгрузки транспортных средств (ТС) вручную. Количество погрузочно-разгрузочных механизмов – 1 шт., количество платформ для погрузки/разгрузки – 1 шт.

2. Добавить 3 платформы для разгрузки ТС механизированным способом. Создать 3D-анимацию погрузки ТС. Добавить камеры.

Контрольные вопросы

1. Как задать фигуры для анимации очереди?
2. Как добавить 3D-анимацию?
3. Как добавить 3D-объекты?
4. Как произвести поворот 3D-объектов?
5. Как добавить объект Камера?
6. Как добавить объект Conveyor?

Лабораторная работа №6.

Сбор статистики. Оптимизация дискретно-событийной модели

Добавление статистики модели

AnyLogic предоставляет пользователю удобные средства для сбора статистики по работе блоков диаграммы процесса. Объекты Библиотеки моделирования процессов самостоятельно производят сбор основной статистики.

Сбор статистики использования ресурсов

Добавляем диаграмму для отображения средней занятости банкомата (см. палитру Статистика) (рисунок 6).

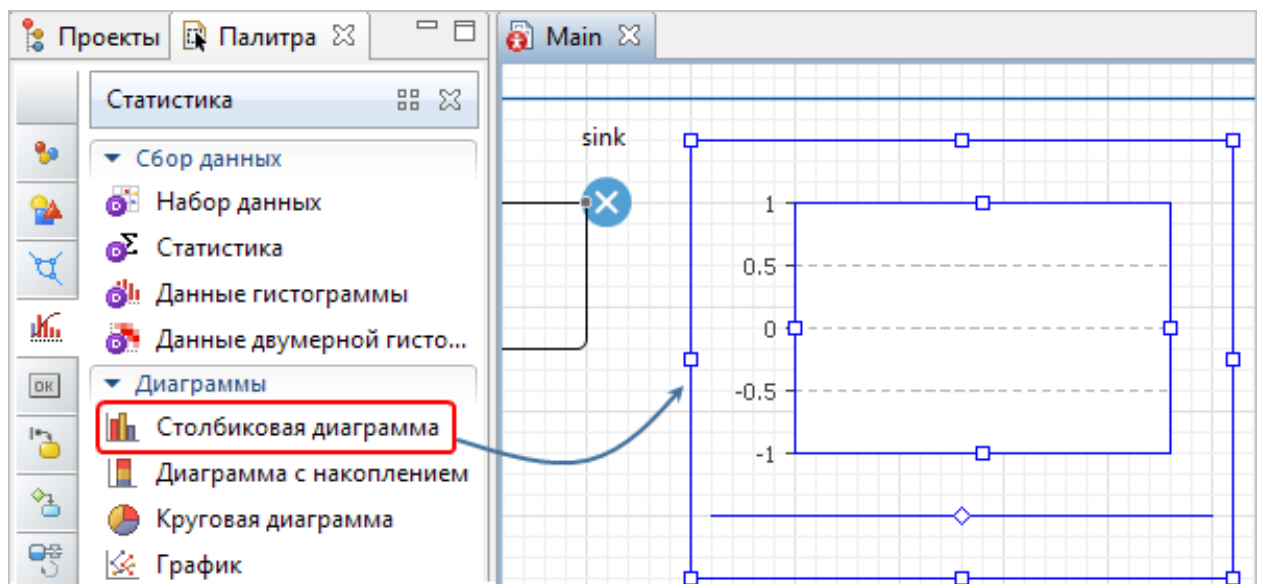


Рисунок 6 – Добавление диаграммы

Задание свойств диаграммы

1. Перейдите в секцию **Данные** свойств столбиковой диаграммы. Щелкните кнопку **Добавить элемент данных**, чтобы задать данные для отображения в диаграмме. Измените **Заголовок** на *Банкомат*.

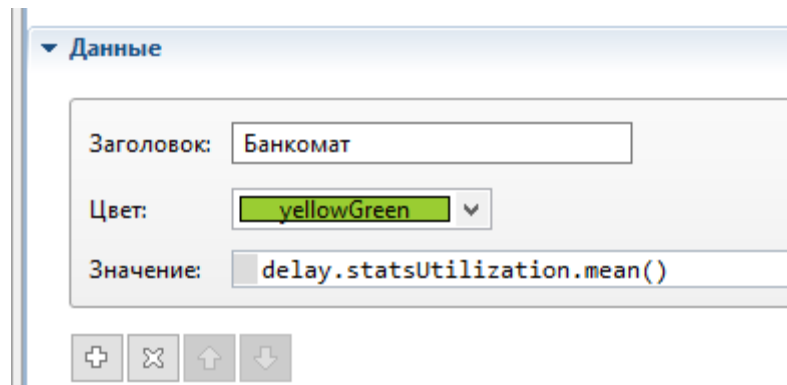


Рисунок 6.1 – Данные диаграммы

2. Введите `Delay.statsUtilization.mean()` в поле **Значение**. Здесь `Delay` – это имя объекта **Delay**. У каждого объекта **Delay** есть встроенный набор данных `statsUtilization`, занимающийся сбором статистики использования этого объекта (рисунок 6.1). Функция `mean()` возвращает среднее из всех измеренных этим набором данных значений. Можно использовать и другие методы сбора статистики, такие, как `min()` или `max()`.

3. Перейдите в секцию **Легенда** панели **Свойства**. Измените расположение легенды относительно диаграммы (рисунок 6.2).

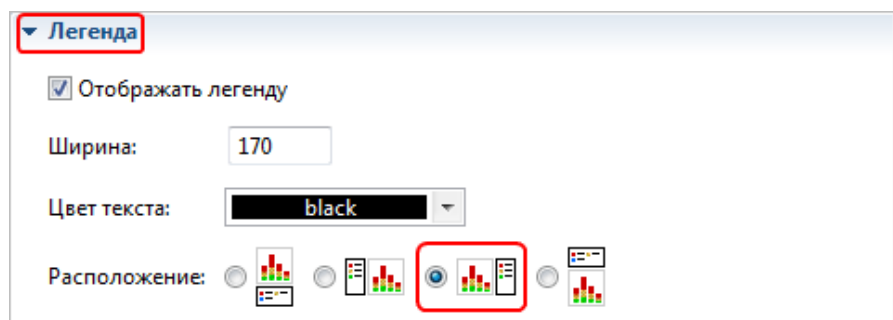


Рисунок 6.2 – Легенда панели Свойства

4. Измените размер столбиковой диаграммы (рисунок 6.3).



Рисунок 6.3 – Столбиковая диаграмма

Добавление диаграммы для отображения средней длины очереди

1. Аналогичным образом добавим еще одну столбиковую диаграмму. Изменим ее размер.

2. Перейдите в секцию **Данные** настройки диаграммы. Добавим новый элемент данных. Задаем **Заголовок**: *Длина очереди* и задаем **Значение**: `queue.statsSize.mean()`. Здесь `statsSize` – это имя объекта типа «статистика» **StatisticsContinuous**, производящего сбор статистики размера очереди объекта **Queue** (рисунок 6.4).

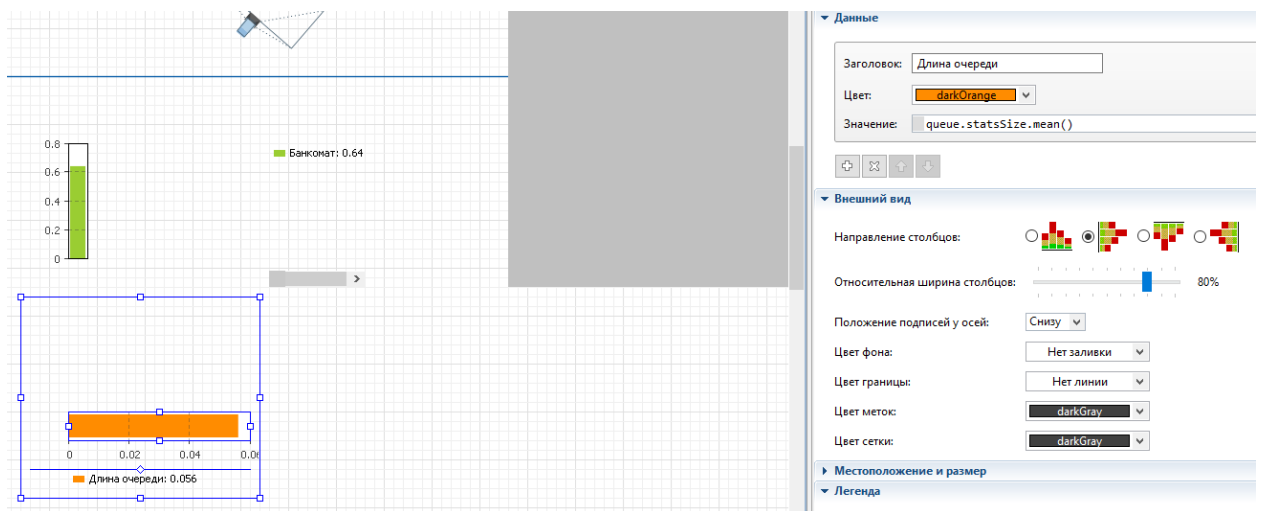


Рисунок 6.4 – Элементы данных

Сбор статистики по времени обслуживания

Определим, сколько времени клиент проводит в банковском отделении и сколько времени он теряет, ожидая своей очереди.

Добавление параметров

1. Переключитесь в панель **Проекты**. Дважды щелкните по типу агента **Customer**, чтобы открыть его диаграмму. Необходимо создать параметры на диаграмме агента **Customer**, так как мы хотим собирать статистику клиентов по времени их обслуживания.

2. Откройте палитру **Агент** в панели **Палитра**. Перетащите два элемента **Параметр** на диаграмму **Customer**. Назовите параметры *startWaiting* и *enteredSystem*, оставьте тип `double`, заданный по умолчанию (рисунок 6.5).

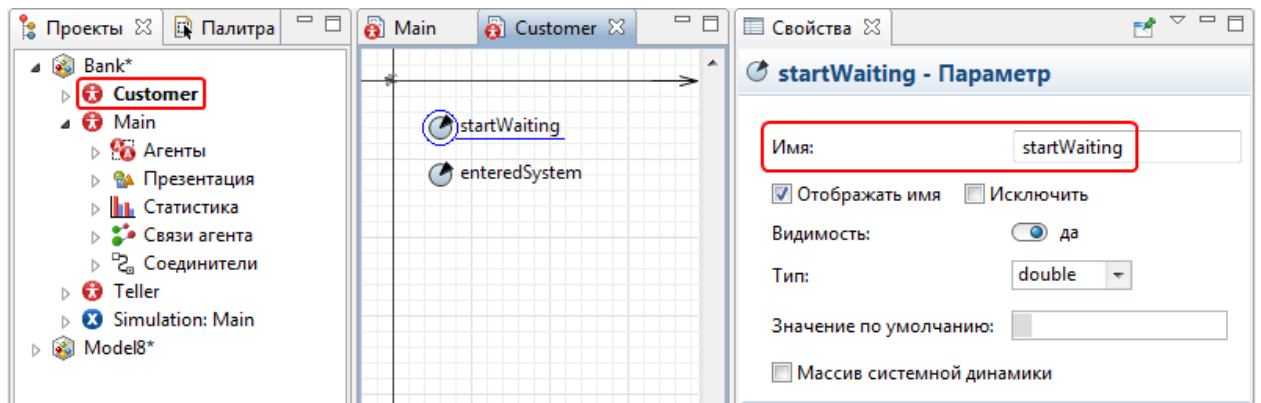


Рисунок 6.5 – диаграмма Customer

3. Перейдите на диаграмму Main. Добавьте элементы сбора статистики по времени ожидания клиентов и времени пребывания клиентов в системе. Эти элементы будут запоминать соответствующие значения времен для каждого клиента и предоставят пользователю стандартную статистическую информацию: среднее, минимальное, максимальное из измеренных значений, среднеквадратичное отклонение, доверительный интервал для среднего и т. д.

Добавление элементов сбора данных

1. Чтобы добавить объект сбора данных гистограммы на диаграмму, перетащите элемент **Данные гистограммы** с палитры **Статистика** на диаграмму агента Main (рисунок 6.6).

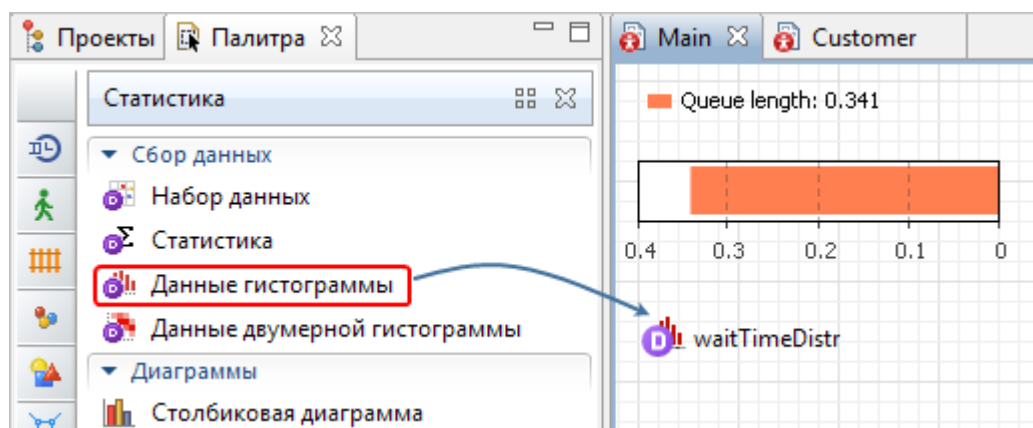


Рисунок 6.6 – Данные гистограммы

2. Задайте свойства элемента (рисунок 6.7).
Измените **Имя** на *waitTimeDistr*.
Сделайте **Количество интервалов** равным 50.
Задайте **Начальный размер интервала**: 0,01.

Свойства

waitTimeDistr - Данные гистограммы

Имя: waitTimeDistr

☒ Отображать имя ☐ Исключить

Видимость: да

Значение:

Кол-во интервалов: 50

☒ Считать CDF

☐ Вычислять процентили: Нижний: 10 Верхний: 10

Диапазон значений

☒ Выбирается автоматически

☐ Фиксированный

Нач. размер интервала: 0.01

Рисунок 6.7 – Свойства waitTimeDistr

3. Создайте еще один элемент сбора данных гистограммы. Измените **Имя** этого элемента на *timeInSystemDistr* (рисунок 6.8).

Свойства

timeInSystemDistr - Данные гистограммы

Имя: timeInSystemDistr

☒ Отображать имя ☐ Исключить

Видимость: да

Значение:

Кол-во интервалов: 50

☒ Считать CDF

☐ Вычислять процентили: Нижний: 10 Верхний: 10

Диапазон значений

☒ Выбирается автоматически

☐ Фиксированный

Нач. размер интервала: 0.01

Рисунок 6.8 – Свойства timeInSystemDistr

Добавление гистограмм для отображения распределений времен ожидания клиента и пребывания клиента в системе

1. Чтобы добавить гистограмму на диаграмму агента, перетащите элемент **Гистограмма** из палитры **Статистика** в графический редактор.

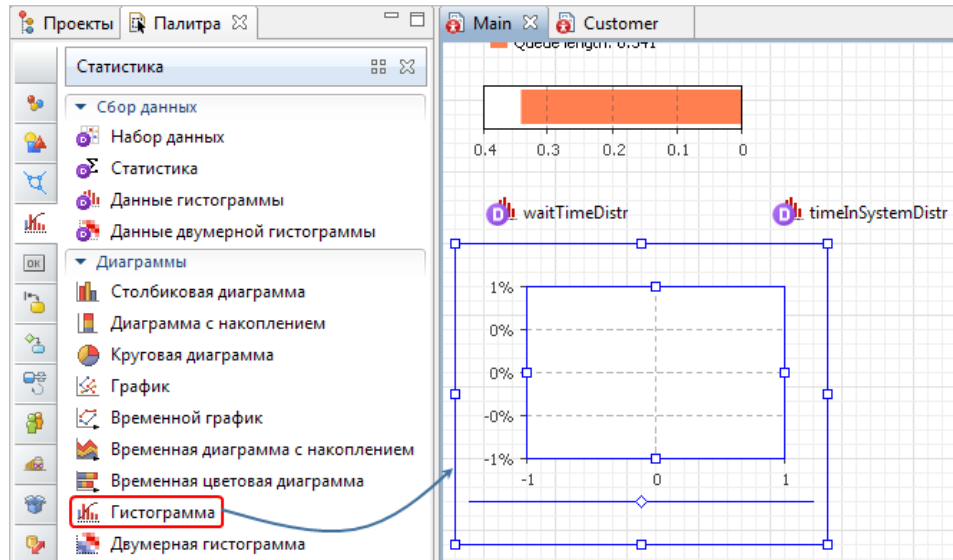


Рисунок 6.9 – Добавление Гистограммы

2. Укажите, какой элемент сбора данных хранит данные, которые Вы хотите отображать на гистограмме: в секции **Данные** свойств гистограммы щелкните мышью по кнопке **Добавить данные** и измените **Заголовок** отображаемых данных на *Waiting Time Distribution*. Введите в поле **Данные** имя соответствующего элемента: waitTimeDistr.

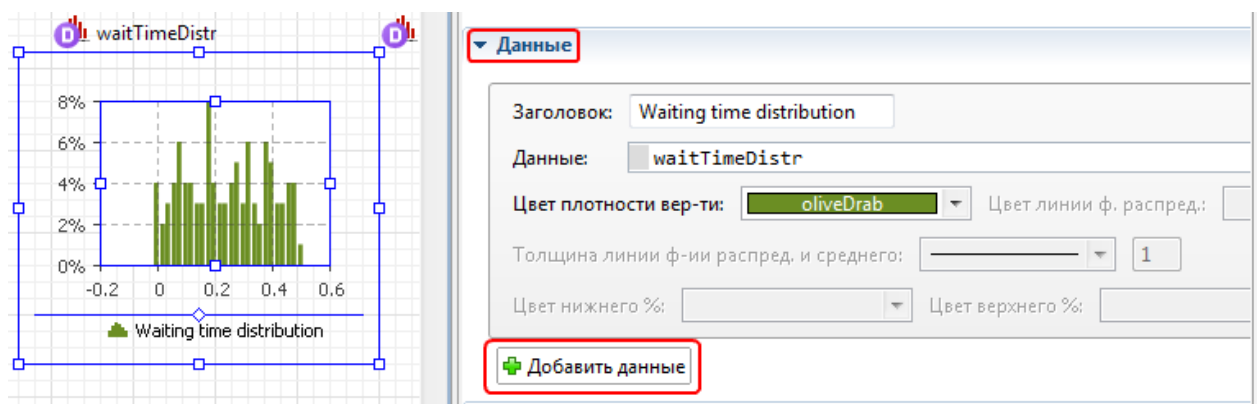


Рисунок 6.10 – Данные гистограммы waitTimeDistr

3. Добавьте еще одну гистограмму и расположите ее под ранее созданной. Измените **Заголовок** отображаемых данных на *Time in system distribution*.

В поле **Данные** введите имя элемента, хранящего данные, которые будут отображаться на гистограмме: `timeInSystemDistr` (рисунок 6.11).

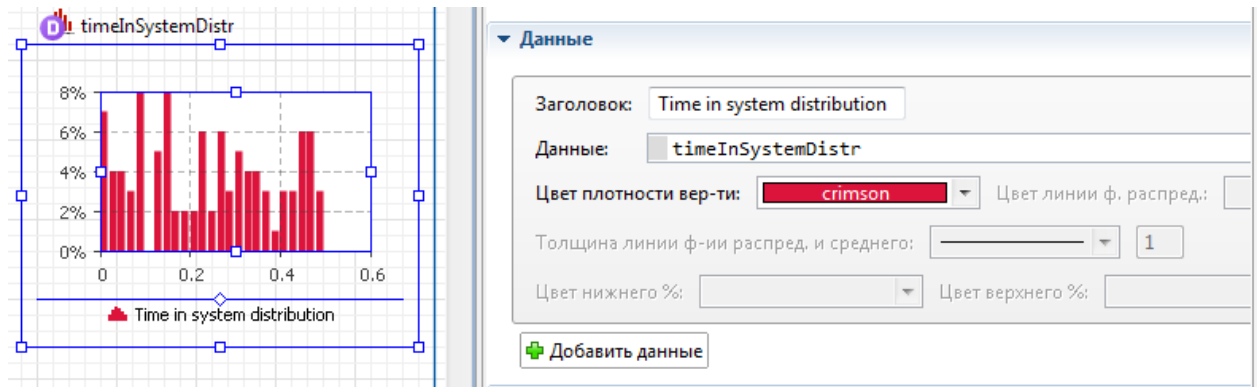


Рисунок 6.11 – Данные гистограммы `timeInSystemDistr`

4. Запустите модель. Включите режим виртуального времени и наблюдайте за тем, какой вид примет распределение времен ожидания и пребывания клиента в системе (рисунок 6.12).

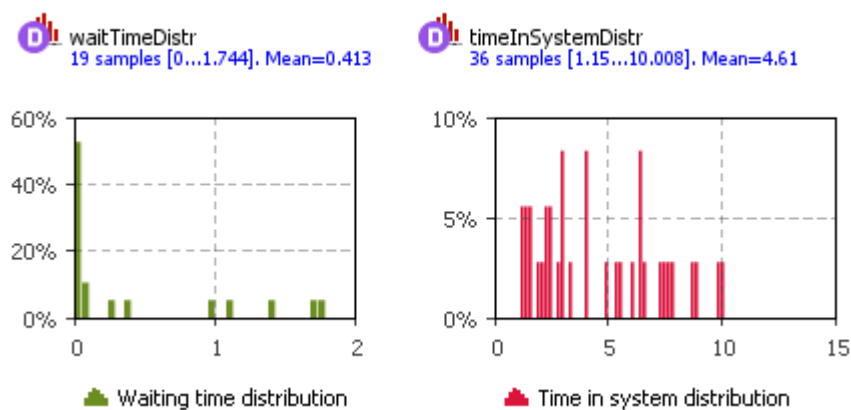


Рисунок 6.12 – Распределение времени ожидания и пребывания клиента в системе

Оптимизация дискретно-событийной модели. Создание интерфейса

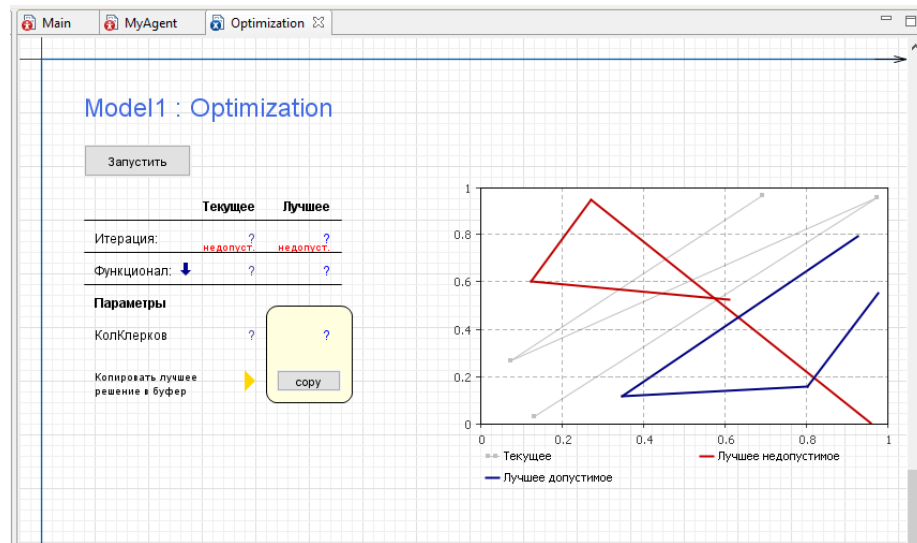


Рисунок 6.13 – Интерфейс Оптимизации

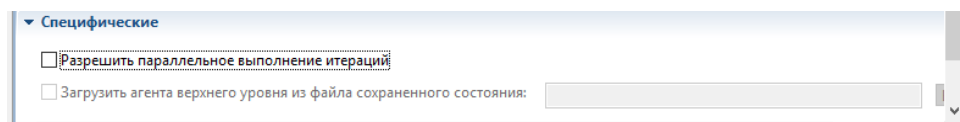


Рисунок 6.14 – Свойства Оптимизации

Оптимизация

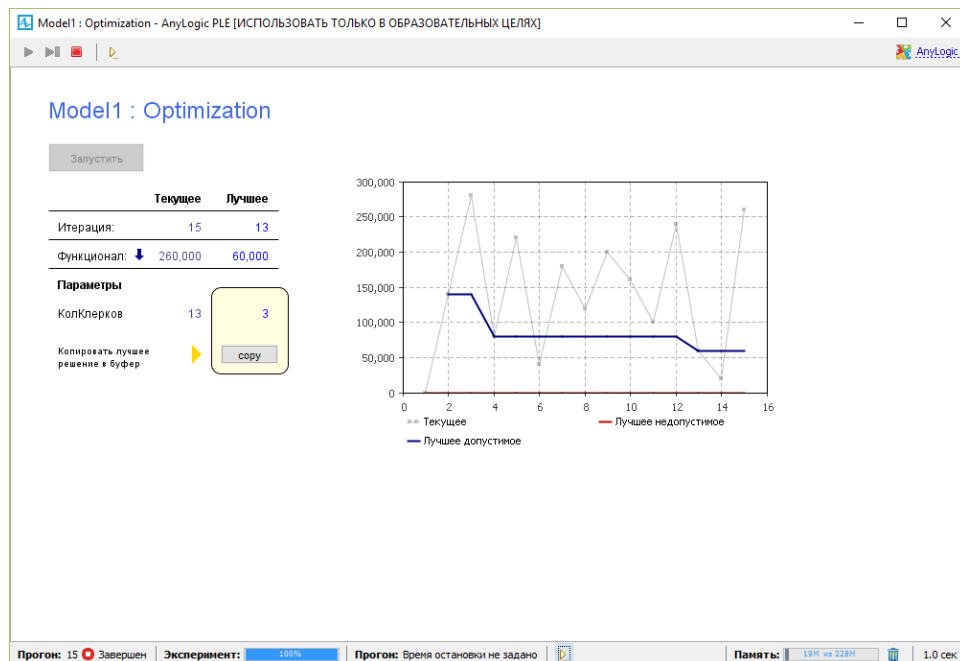


Рисунок 6.15 – Модель Оптимизации

Задание

1. Собрать статистику по использованию объектов в системе.
2. Провести оптимизацию зоны погрузки/разгрузки транспортных средств. Определить оптимальное количество разгрузочных платформ при различной интенсивности прибытия автомобилей.

Контрольные вопросы

1. Как провести сбор статистики использования ресурсов?
2. Как провести сбор статистики по времени обслуживания?
3. Какие элементы необходимы для сбора данных?
4. Что такое оптимизация?
5. Для чего используется оптимизация?

Лабораторная работа №7. Построение пешеходной модели

Пешеходное моделирование

Пешеходная библиотека AnyLogic является высокоуровневой библиотекой моделирования движения пешеходов в физическом пространстве. С ее помощью можно моделировать здания с движущимися пешеходами (станции метро, стадионы, музеи), а также улицы и другие места с большим скоплением людей.

С помощью *Пешеходной библиотеки* можно:

- визуализировать моделируемый процесс;
- собирать статистику: плотности пешеходов, времени пребывания пешеходов в определенных участках модели;
- выявить возможные проблемы, которые могут возникнуть при перепланировке интерьера здания, изменении интенсивности пешеходного движения и т. д.

В моделях, созданных с помощью объектов *Пешеходной библиотеки*, пешеходы перемещаются в непрерывном пространстве, реагируя на различные виды препятствий: стены, особые области и другие пешеходы.

Основными составляющими модели движения пешеходов являются:

- **Среда** – объекты физической среды (стены, различные области, сервисы, очереди и т. д.). Объекты среды создаются с помощью специальных графических элементов разметки пространства. Ресурсы (сервисы) также являются объектами среды.

- **Поведение пешеходов** – задается блок-схемой.

Основным объектом библиотеки является пешеход. Пешеход задается с помощью объекта типа *Ped*. Пешеход «обитает» в заданном физическом пространстве (моделируемой среде) и передвигается согласно заданным правилам. Тип пешехода унаследован от типа агента *Agent*, поэтому пешеходы перемещаются по блок-схеме так же, как агенты.

Класс *Ped* предоставляет следующие функции для работы с пешеходами:

- `double getX()` – возвращает x-координату пешехода в модели (в метрах).

- `double getY()` – возвращает у-координату пешехода в модели (в метрах).
- `double getZ()` – возвращает z-координату (в метрах) пешехода в модели (в метрах).
- `double getTargetX()` – возвращает x-координату места назначения, к которому движется данный пешеход (в метрах).
- `double getTargetY()` – возвращает у-координату места назначения, к которому движется данный пешеход (в метрах).
- `double getTargetZ()` – возвращает z-координату места назначения, к которому движется данный пешеход (в метрах).
- `double getSpeed()` – возвращает текущую скорость пешехода, в метрах в секунду.
- `double getComfortableSpeed()` – возвращает скорость, с которой данному пешеходу комфортно двигаться.
- `double getDiameter()` – возвращает диаметр пешехода в метрах.
- `void setDiameter(double diameter)` – задает новый диаметр пешехода в метрах.
- `Level getLevel()` – возвращает уровень, на котором находится пешеход.
- `Group getGroup()` – возвращает группу, к которой принадлежит этот пешеход, или `null`, если он не входит в состав никакой группы.
- `int getId()` – возвращает уникальный идентификатор пешехода. Возвращает `-1`, если пешеход еще не был добавлен в моделируемую пешеходную среду.

В AnyLogic возможны 2 вида визуализации пешеходов:

- **Стандартная 2D-анимация.** По умолчанию пешеход отображается на анимации кружком, радиус которого соответствует заданному радиусу пешехода. При этом изменение цвета фигуры анимации пешеходов на красный во время имитации, можно осуществить с помощью кода `ped.setColor(red)`;
- **3D-анимация.** Если необходимо отображать пешеходов на сцене трехмерной анимации с помощью 3D-моделей людей, то для этого нужно создать новый тип пешехода, например, `Pedestrian` (рисунок 7). Затем для созданного типа пешехода необходимо выбрать фигуру анимации и добавить параметры.

Созданный тип пешехода необходимо выбрать в качестве значения параметра *Новый пешеход* объекта *PedSource*. Тогда пешеходы в моделируемом процессе будут иметь тип *Pedestrian*, и объекты на блок-схеме позволят явно обращаться к дополнительной функциональности типа *Pedestrian* через локальную переменную *ped*.

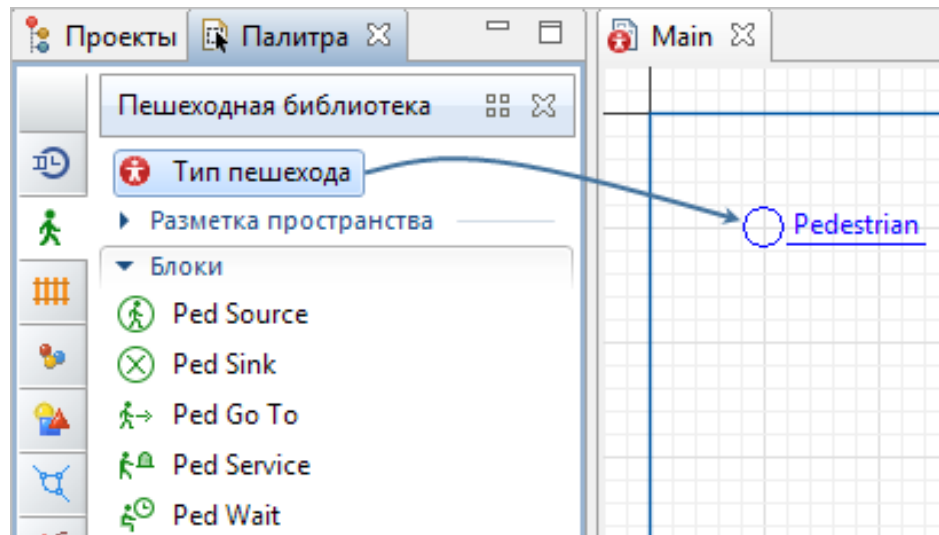


Рисунок 7 – Создание агента типа Pedestrian

Блок-схемы пешеходных моделей строятся с помощью объектов, содержащихся в *Пешеходной библиотеке*. В библиотеке есть объекты не только для создания пешеходов, но и для управления потоком пешеходов. Правила задания потока пешеходов аналогичны правилам задания потока агентов в *Библиотеке моделирования процессов*. Разница заключается в том, что пешеходы двигаются согласно правилам движения в физическом пространстве и выбирают свой путь, анализируя текущее положение в пространстве.

Для построения простейшей модели перемещения пешеходов в моделируемой среде используются следующие блоки Пешеходной библиотеки: *PedSource*, *PedEnter*, *PedExit*, *PedSink*, *PedGoTo*, *PedWait*, *PedSelectOutput*.

Блок *PedSource*  создает пешеходов. Обычно используется в качестве начальной точки блок-схемы, формализующей поток пешеходов. Созданные пешеходы добавляются в моделируемую среду и направляются далее согласно созданной диаграмме процесса. Хотя пешеходы движутся в блок-

схеме, их движение между блоками блок-схемы определяется моделируемой средой.

Параметры блока PedSource

- **Место появления** – указывается, будут ли появляться пешеходы в заданной линии, точке или области. Имя: locationType.

Возможные значения: PedSource.LOCATION_LINE – линия;

PedSource.LOCATION_POINT – точка;

PedSource.LOCATION_AREA – область.

- **Прибывают согласно** – определяет, будут ли пешеходы прибывать согласно: интенсивности, времени между прибытиями, расписанию интенсивностей, расписанию прибытий, вызовам функции inject().
- **Новый пешеход** – тип создаваемого пешехода.
- **Комфортная скорость** – скорость, с которой будет двигаться пешеход при отсутствии внешних факторов.

Функции блока PedSource

- void inject() – добавляет одного пешехода или группу пешеходов в моделируемую среду.
- void inject(int count) – метод добавляет заданное количество пешеходов (count) в моделируемую среду.
- long countPeds() – возвращает количество всех созданных блоком пешеходов.
- long countGroups() – возвращает количество прибывших в этот блок групп пешеходов.
- int size() – возвращает количество пешеходов, находящихся внутри блока.

Блок PedEnter  Объект принимает созданных пешеходов во входном порту, задает их физические характеристики и добавляет их в заданное место моделируемой среды. Объект может формировать группы из поступающих пешеходов, позволяя задавать время прибытия пешеходов, размеры создаваемых групп, интенсивность создания групп и т. д.

Блок PedExit  принимает поступающих во входной порт пешеходов, удаляет их из моделируемой среды и посылает на

выходной порт. С помощью блоков PedEnter и PedExit можно создать сложную схему перенаправления пешеходов в другие подпроцессы, не устанавливая графические соединения между блоками.

Блок PedSink  удаляет поступивших в объект пешеходов из моделируемой среды. Обычно объект используется в качестве конечной точки диаграммы пешеходного процесса.

Блок PedGoTo  Заставляет пешеходов перейти в заданное место моделируемого пространства, которое может быть задано линией, точкой или областью. Переход будет считаться выполненным, когда пешеход пересечет заданную линию, либо достигнет заданной точки или области. Пешеходы будут искать путь к заданной цели в пределах текущего уровня. Существуют два режима перемещения пешеходов: можно направить пешеходов к заданной цели или направить их по заданному пути.

Основные параметры блока PedGoTo

- **Режим движения пешеходов** – достичь цели или следовать по заданному пути.

Имя: mode.

Возможные значения:

pedGoTo.MODE_REACH_TARGET – Достичь цели.

pedGoTo.MODE_FOLLOW_ROUTE – Следовать по заданному пути.

- **Цель** – задает цель движения для пешехода. Целью может быть: линия, точка или область.

Имя: locationType.

Возможные значения:

pedGoTo.LOCATION_LINE – линия.

pedGoTo.LOCATION_POINT – точка (x, y).

pedGoTo.LOCATION_AREA – область.

- **Целевая линия** – Имя целевой линии, к которой будут перемещаться пешеходы.

Тип значения: TargetLine.

- **Область** – имя области, к которой будут перемещаться пешеходы.

Тип значения: AreaNode

Основные функции блока PedGoTo

- `long countPeds()` – возвращает количество всех прошедших через блок пешеходов.
- `int size()` – возвращает количество пешеходов, находящихся внутри объекта.
- `void cancel(Agent ped)` – заставляет заданного пешехода немедленно покинуть блок через порт `ccl` (экстренным/аварийным способом).
- `void cancelAll()` – заставляет всех пешеходов немедленно покинуть блок через порт `ccl`.
- `boolean contains(Agent ped)` – возвращает `true`, если заданный пешеход находится в данный момент внутри объекта, иначе возвращает `false`.
- `Set<Agent> getPeds()` – возвращает неизменяемую коллекцию пешеходов, находящихся в этом блоке.



Блок PedService направляет поток пешеходов через группу сервисов и очередей. Сервис может быть указан как двунаправленный, тогда пешеходы смогут проходить сервис в любом направлении. Объект позволяет задавать очереди и сервисы в любой комбинации и задавать правила выбора сервисов.

Основные параметры блока PedService

- Сервисы – указывается фигура, задающая сервис.
- Выбирается очередь – задает правило выбора очереди.
- Время задержки – задает время обслуживания в сервисе.
- Задержка на восстановление – задает время восстановления сервиса после обслуживания пешехода, в течение которого сервис остается недоступным.
- Проходить в обратном направлении – если опция выбрана (`true`), то пешеход будет проходить через этот сервис в обратном направлении.

Основные функции блока PedService

- `long countPeds()` – возвращает количество всех прошедших через блок пешеходов.

- `int size()` – возвращает количество пешеходов, находящихся в этом блоке.
- `void cancel(Agent ped)` – заставляет заданного пешехода немедленно покинуть блок через порт `ccl`. Для пешехода выполняется код параметра «Действие при отмене».
- `void cancelAll()` – заставляет всех пешеходов немедленно покинуть блок через порт `ccl`. Для каждого пешехода выполняется код параметра «Действие при отмене».
- `int queueSize(QueueUnit queueUnit)` – возвращает количество пешеходов в очередях данного блока.
- `boolean contains(Agent ped)` – возвращает `true`, если заданный пешеход находится в данный момент внутри объекта, иначе возвращает `false`.
- `Set<Agent> getPeds()` – возвращает неизменяемую коллекцию пешеходов, находящихся в этом блоке.



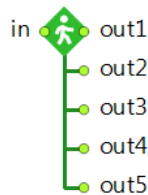
Блок PedWait заставляет пешеходов перейти в заданное место и ожидать там в течение определенного периода времени. Для определения места ожидания необходимо указать соответствующую фигуру разметки пространства: линию, точку или область. Пешеходы могут ожидать в течение заданного промежутка времени, отсчитанного относительно одного из событий (достижения точки ожидания, входа в область, либо же время может быть задано вручную), или вечно, в таком случае прекращение ожидания осуществляется вручную путем вызова метода `free()`.

Основные функции блока PedWait

- `void activate(Agent ped)` – начинает ожидание для заданного пешехода.
- `long countPeds()` – возвращает количество всех прошедших через блок пешеходов.
- `int size()` – возвращает количество пешеходов, находящихся внутри блока.
- `void free(Agent ped)` – прерывает выполнение команды для заданного пешехода и заставляет его покинуть объект через

порт out. Для каждого пешехода, покидающего блок, вызывается код параметра Действие при выходе.

- void freeAll() – прерывает выполнение команды для всех пешеходов в блоке и заставляет их покинуть объект через порт out.
- void freeAllWaitingPeds() – прерывает ожидание пешехода и заставляет его покинуть объект через порт out.
- void cancel(Agent ped) – прерывает команду для заданного пешехода и заставляет его покинуть объект через порт csl.



Блок PedSelectOutput направляет входящих в блок пешеходов на один из пяти выходных портов в соответствии с заданными условиями или коэффициентами. Дальнейшие действия пешеходов определяются отдельными диаграммами процесса, присоединенными к выходным портам блока.

Можно выделить следующие этапы создания пешеходной модели:

1. Добавление рисунка-плана моделируемого пространства (помещения, здания) (рисунок 7.1).

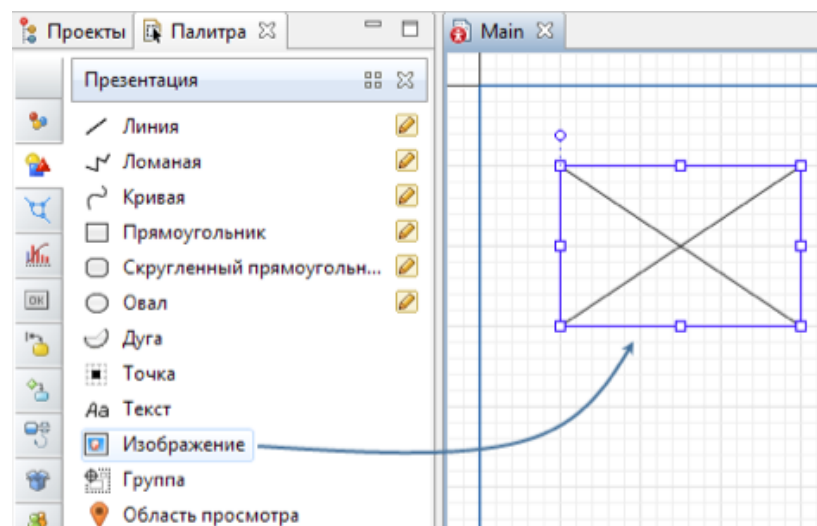


Рисунок 7.1 – Добавление рисунка-плана моделируемого пространства

2. Поверх стен на этом плане рисуются стены с помощью специальных элементов разметки пространства AnyLogic (рисунок 7.2).

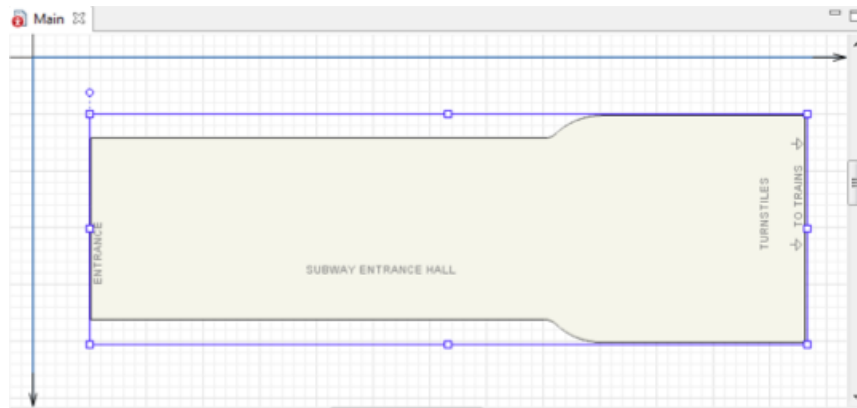


Рисунок 7.2 – Использование фигур разметки пространства для рисования стен

3. Создание диаграммы процесса, предназначенной для задания логики перемещения пешеходов внутри здания (рисунок 7.3).

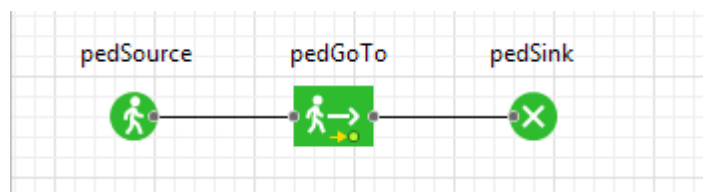


Рисунок 7.3 – Потокковая диаграмма

Элементами разметки пространства являются:

- **Стены** – объекты, которые пешеходы и транспортеры, передвигающиеся в режиме произвольной навигации, не могут пересекать. Они используются для задания стен и препятствий в моделях. В зависимости от необходимости можно использовать объект *Стена* (для рисования стен сложной формы), Прямоугольная стена (для задания прямоугольных помещений), Округлая стена (для задания округлых пространств). Стены являются частью уровня, следовательно, одна стена не может быть использована для нескольких уровней сразу.
- **Целевая линия** используется в моделях пешеходной динамики для задания места появления пешеходов в моделируемой среде, цели перемещения пешеходов, места ожидания пешеходов, места выхода с текущего уровня и перехода на новый уровень.
- **Сервис** задает группу одинаковых физических объектов обслуживания (например, несколько турникетов или автома-

тов по продаже билетов). Различают сервисы с очередями (используются для того, чтобы задавать сервисы, в которых пешеходы ждут в очереди, пока сервис не будет доступен) и сервисы с областью (используется для того, чтобы задавать сервисы с электронной очередью).

Для выявления участков пространства, где плотность посетителей достигает критических значений, применяется *Карта плотности пешеходов*. «Перегруженные» области отображаются на карте плотности красным цветом. По умолчанию критическая плотность задана на уровне 1,5 пешехода на квадратный метр. Синий цвет используется для участков низкой плотности. При нулевой плотности закрашивание соответствующего участка не производится вообще. Можно отключить отображение карты плотности, но при этом продолжать собирать соответствующую статистику, без отображения карты при анимации модели. Для этого необходимо сбросить флажок *Показывать карту плотности на анимации*. В таком случае можно увеличить скорость выполнения модели. Собранный статистика может храниться в наборах данных, откуда, например, по окончании моделирования, записываться в базу данных для последующего статистического анализа.

Рассмотрим различные варианты перемещения и обслуживания пешеходов в ограниченном пространстве.

Задание

Вариант 1. Построение модели перемещения и обслуживания пассажиров на автовокзале заданной планировки.

Для построения модели необходимо выполнить несколько этапов:

1 этап. Сбор информации о транспортном объекте: планировка, дислокация мест обслуживания пассажиров, входов/выходов в здание и на перрон, зон отдыха (рисунок 7.4).

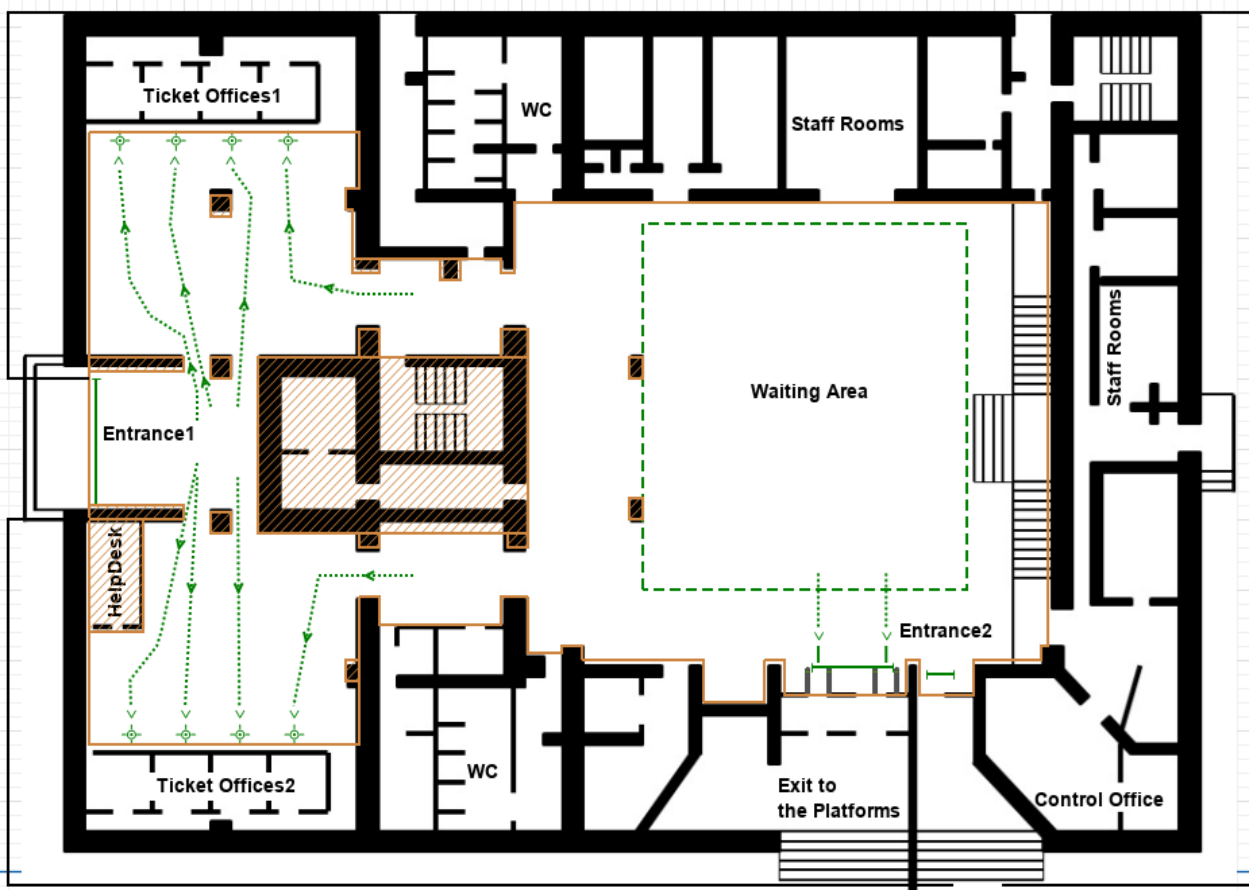


Рисунок 7.4 – 2D модель объекта транспортной инфраструктуры

2 этап. Сбор информации о пассажирах: состав пассажиропотока (возраст, пол, категория), поведение внутри транспортного объекта, интенсивность прибытия в здание автовокзала.

3 этап. Задание фигур разметки пространства в среде моделирования.

4 этап. Построение потоковой диаграммы, отражающей логику перемещения пассажиров внутри ограниченного пространства (рисунок 7.5).

Всех пассажиров в зависимости от их поведения условно можно разделить на несколько типов:

- Passenger1 – посетители покупают билет заранее, после покупки покидают здание автовокзала;
- Passenger2 – посетители имеют билет, купленный заранее (онлайн/офлайн), в здании автовокзала перемещаются либо в зал ожидания (время ожидания – случайная величина, распределенная по нормальному закону распределения со средним 20 минут);
- Passenger3 – пассажиры покупают билет в кассах ticketOffices1 или ticketOffices2, в здании автовокзала перемеща-

ются либо в зал ожидания, либо на перрон в зависимости от времени, оставшегося до отправления рейса.

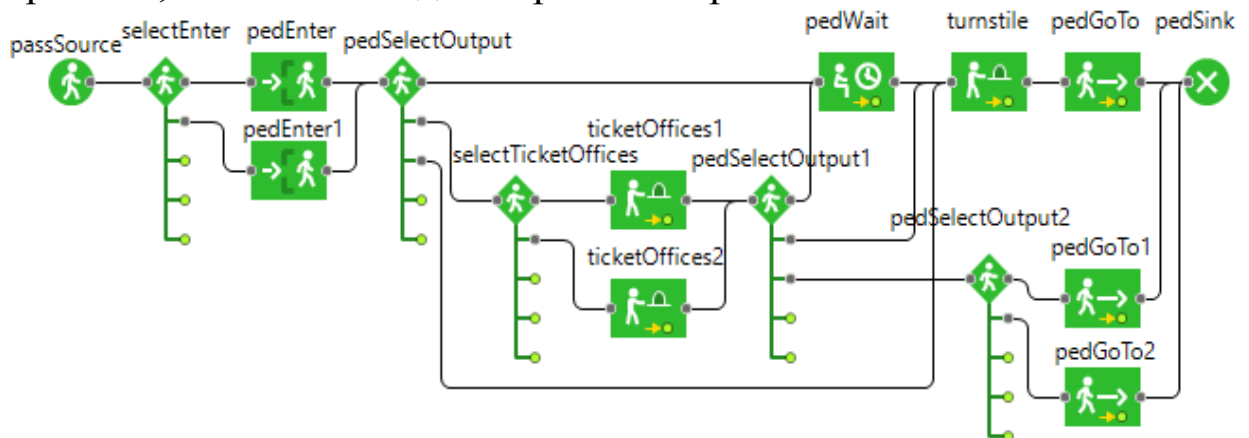


Рисунок 7.5 – Потокная диаграмма, отражающая логику перемещения посетителей внутри объекта транспортной инфраструктуры

Вариант 2. Построение модели эвакуации пассажиров.

Для добавления в модель возможности эвакуации необходимо создать 2 кнопки «Начать эвакуацию» и «Закончить эвакуацию». Также необходимо создать область, в которой будут находиться эвакуированные посетители. У всех блоков порт csl (при наличии) необходимо соединить с блоком pedEnter (Рисунок 7.6).

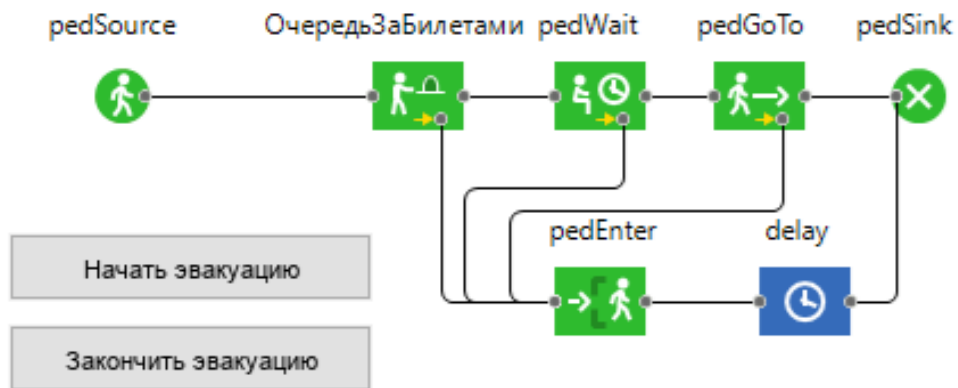


Рисунок 7.6 – Потокная диаграмма, отражающая логику эвакуации посетителей

Для кнопки «Начать эвакуацию» необходимо прописать действия по «освобождению» агентов в виде `имяБлока.cancelAll()`; и переключить блок (блоки) `pedSource` в ручной режим: `имяБлока.set_arrivalType(pedSource.MANUAL)`.

Для кнопки «Закончить эвакуацию» необходимо прописать действия по переключению режима работы блока (блоков) PedSource в режим создания агентов-пешеходов согласно заданной интенсивности: `имяБлока.set_arrivalType(pedSource.RATE)`.

Следует заметить, что при рисовании помещения сложной конфигурации необходимо, чтобы все области (node, area и т. д.), и сервисы находились на одном уровне, например, *ground*. Иначе пешеход не сможет достигнуть нужного сервиса, области. Чтобы пешеход «видел» стены, необходимо, чтобы стены также находились на одном уровне *ground*.

Контрольные вопросы

1. Опишите этапы создания агентов-пешеходов.
2. Опишите процесс создания пешеходной модели.
3. Опишите программную реализацию процесса эвакуации пешеходов.
4. Опишите основные блоки пешеходной модели.
5. Какие фигуры разметки пространства используются при построении модели?

Лабораторная работа №8. Оптимизация пешеходной модели

Порядок выполнения

Для того, чтобы оптимизировать пешеходную модель из предыдущей практической работы нужно сделать следующие шаги:

1. Добавление оптимизируемого параметра для агента верхнего уровня (рисунок 8).

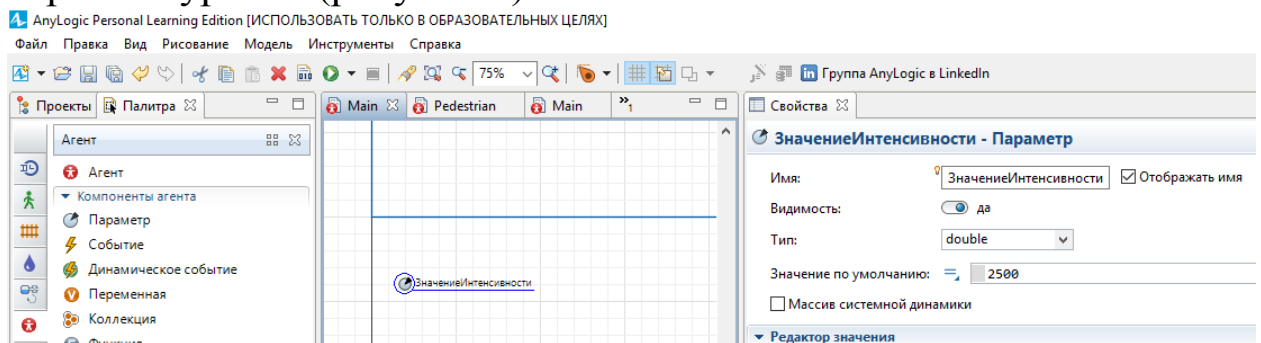


Рисунок 8 – Добавление оптимизируемого параметра

2. Изменение свойств объекта PedSource (рисунок 8.1).

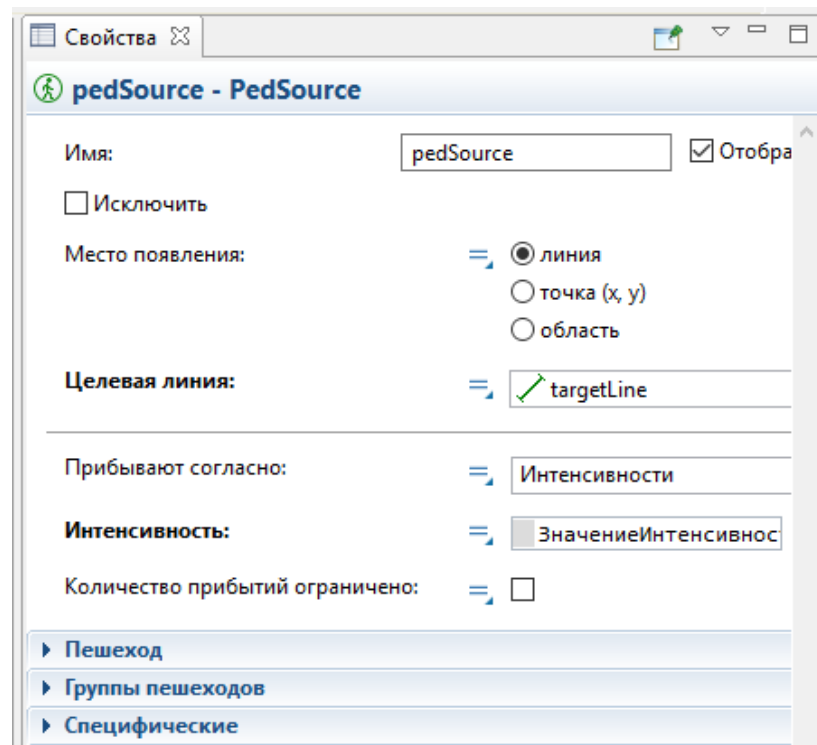


Рисунок 8.1 – Свойства объекта PedSource

3. Задание целевой функции (рисунок 8.2).

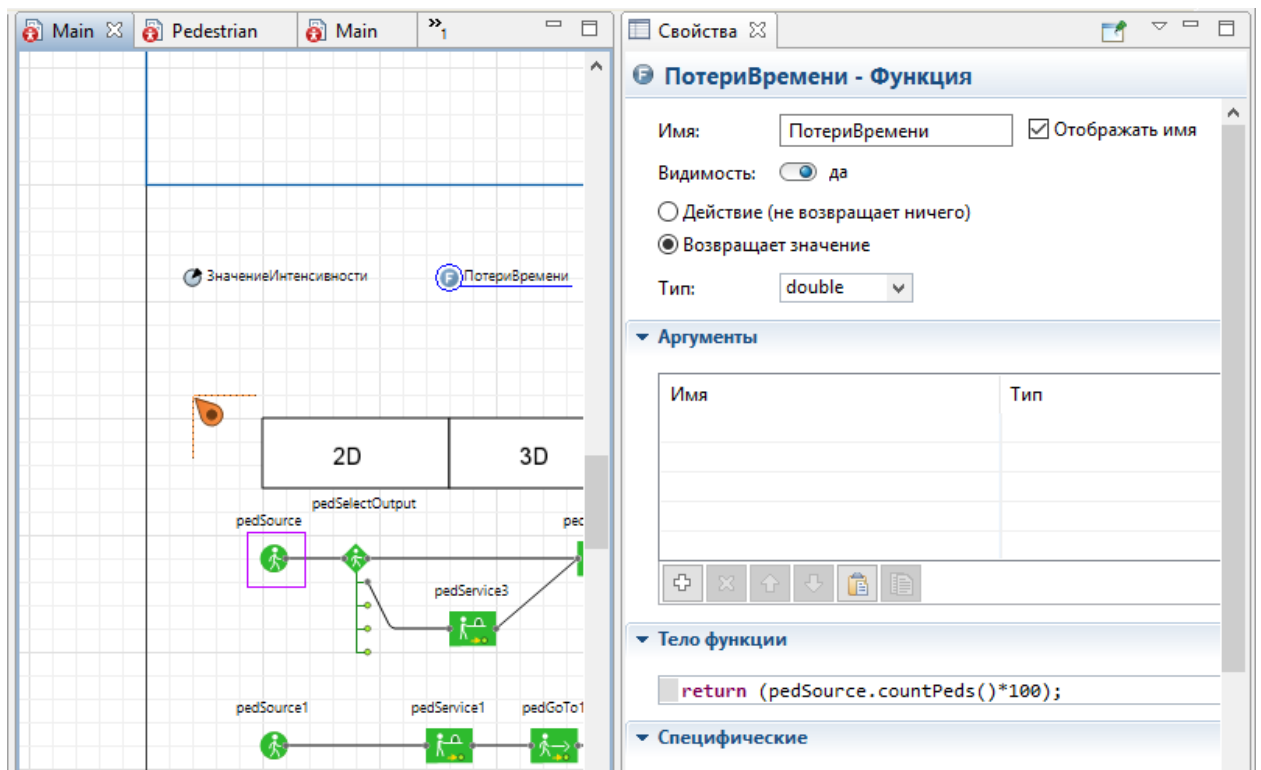


Рисунок 8.2 – Свойства целевой функции

4. Создание эксперимента (рисунок 8.3).

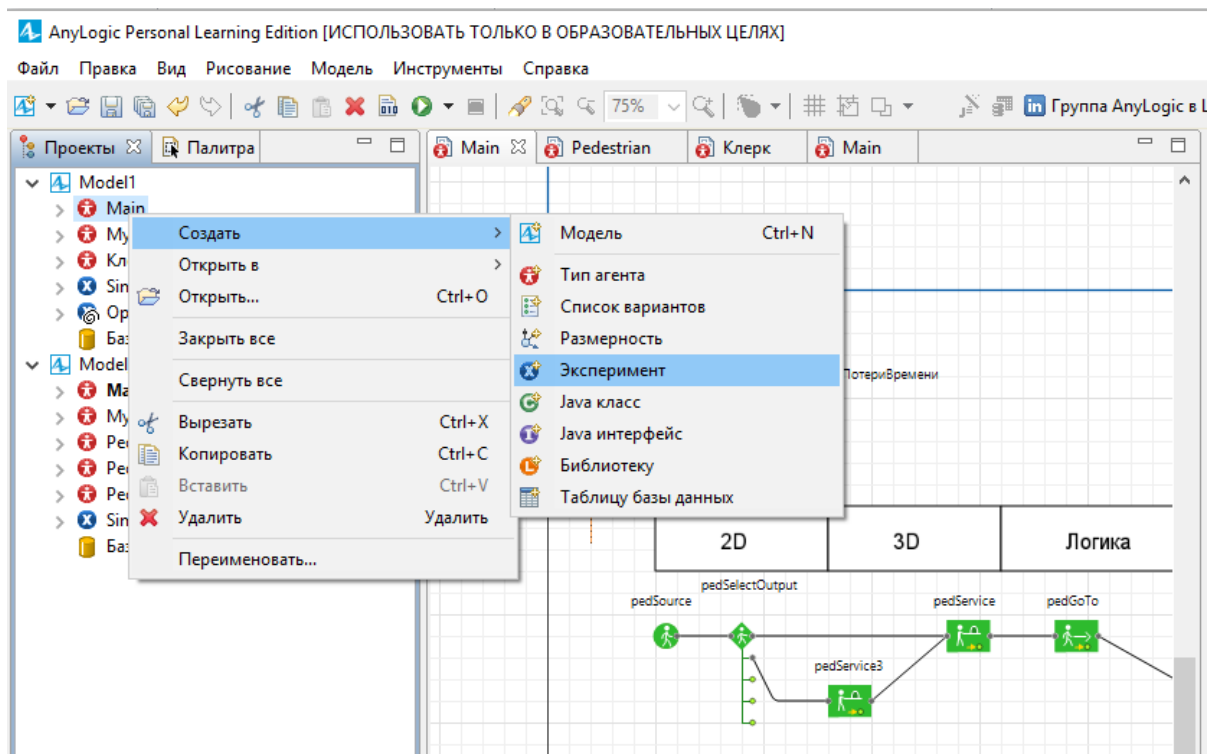


Рисунок 8.3 – Создание эксперимента

5. Добавление переменной variable для хранения времени ожидания у кассы (рисунок 8.4).

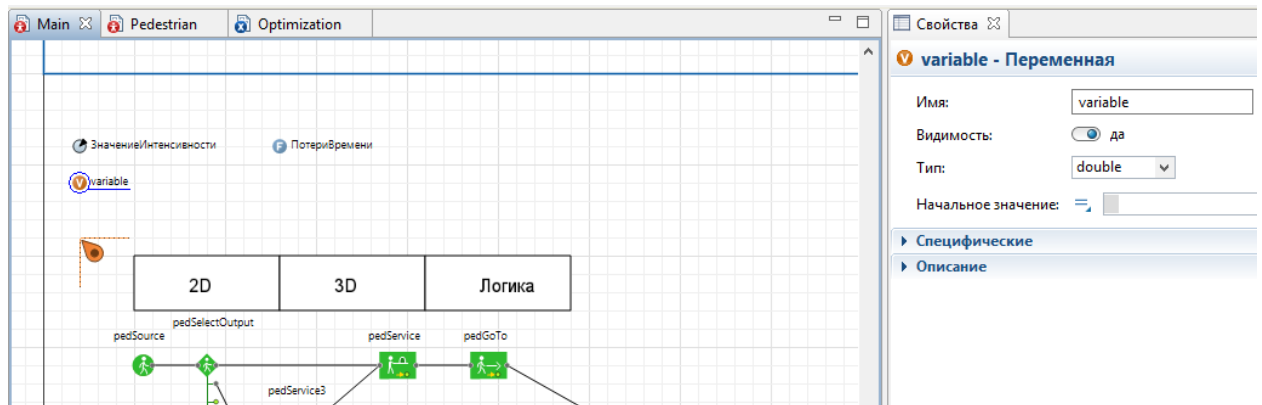


Рисунок 8.4 – Добавление переменной variable

6. Изменение исполняемого кода для объекта pedService3 (рисунок 8.5).

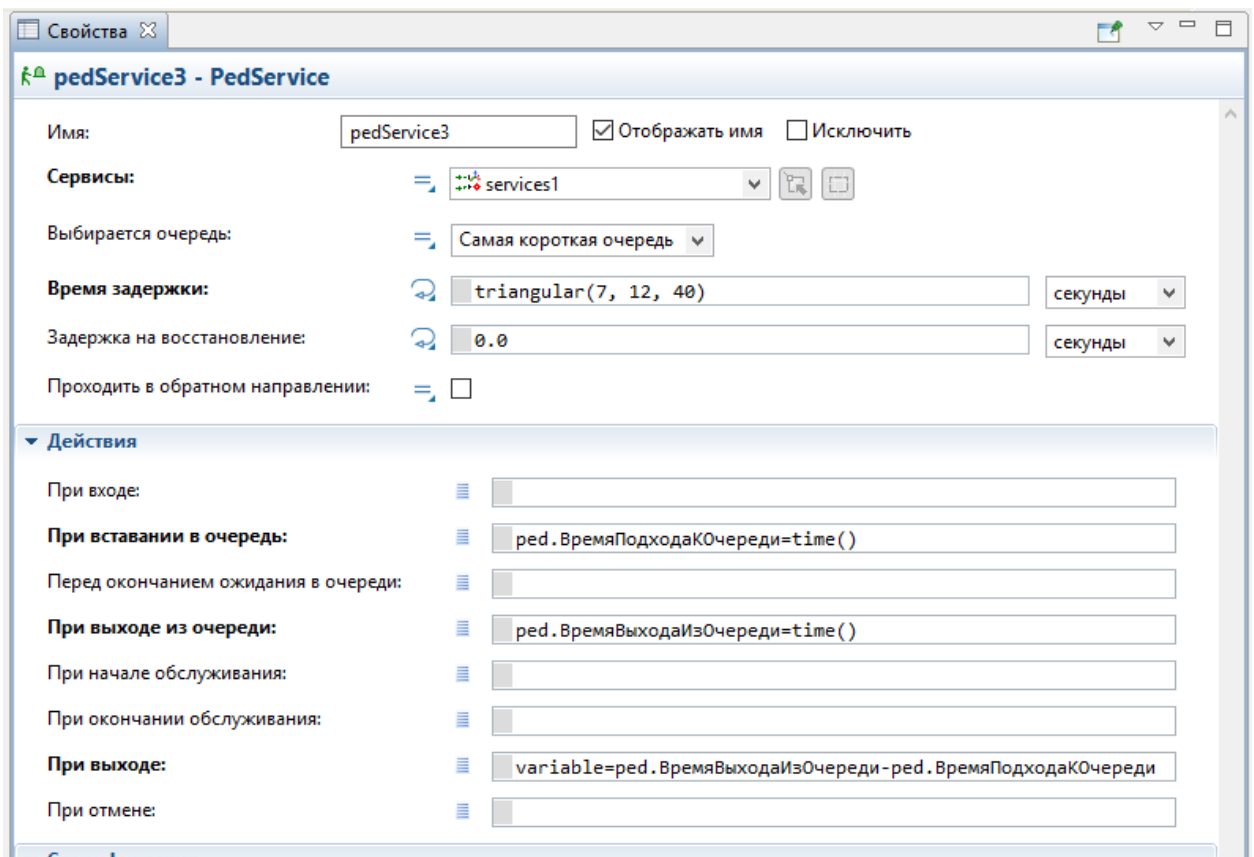


Рисунок 8.5 – Свойства объекта pedService3

7. Настройка эксперимента (рисунок 8.6).

Свойства

Optimization - Оптимизационный эксперимент

Имя: Optimization ☐ Исключить

Агент верхнего уровня: Main

Целевая функция: ☐ минимизировать ☒ максимизировать

root.ПотериВремени()

☒ Количество итераций: 50

☐ Автоматическая остановка

Максимальный размер памяти: 256 M6

Создать интерфейс

Параметры

Параметры:

Параметр	Тип	Значение			
		Мин.	Макс.	Шаг	Начальное
Значен...ности	дискретный	0	3000	10	

► Модельное время

► Ограничения

▼ Требования

Требования (проверяются после "прогона" для определения того, допустимо ли найденное решение):

Вкл.	Выражение	Тип	Гран...
☒	root.variable	<=	10.0

Рисунок 8.6 – Настройка эксперимента

8. Задание модельного времени

Свойства

Optimization - Оптимизационный эксперимент

Имя: Optimization ☐ Исключить

Агент верхнего уровня: Main

Целевая функция: ☐ минимизировать ☒ максимизировать

root.ПотериВремени()

☒ Количество итераций: 50

☐ Автоматическая остановка

Максимальный размер памяти: 256 M6

Создать интерфейс

Параметры

Параметры:

Параметр	Тип	Значение			
		Мин.	Макс.	Шаг	Начальное
Значен...ности	дискретный	0	3000	10	

▼ Модельное время

Остановить: В заданную дату

Начальное время: 0 Конечное время: 60

Начальная дата: 20.03.2016 Конечная дата: 20.03.2016

9:00:00 10:00:00

Дополнительные условия остановки оптимизации:

Вкл.	Выражение

Рисунок 8.7 – Задание модельного времени

9. Создание интерфейса (рисунок 8.8).

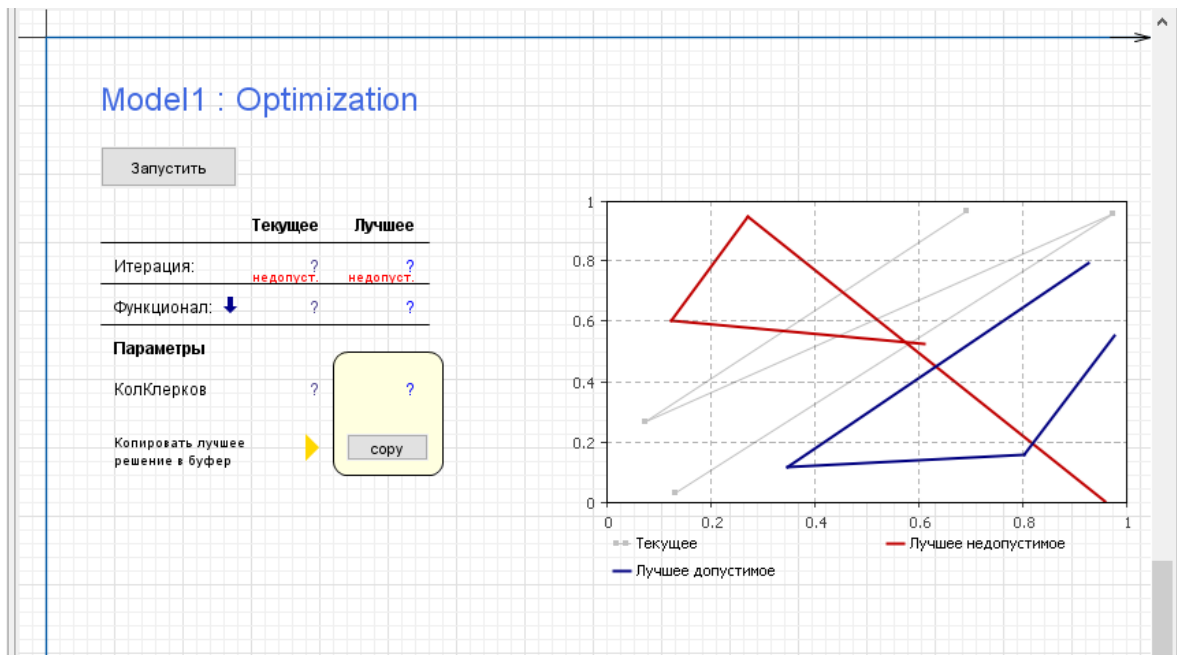


Рисунок 8.8 – Создание интерфейса оптимизации

10. Оптимизация (рисунок 8.9).

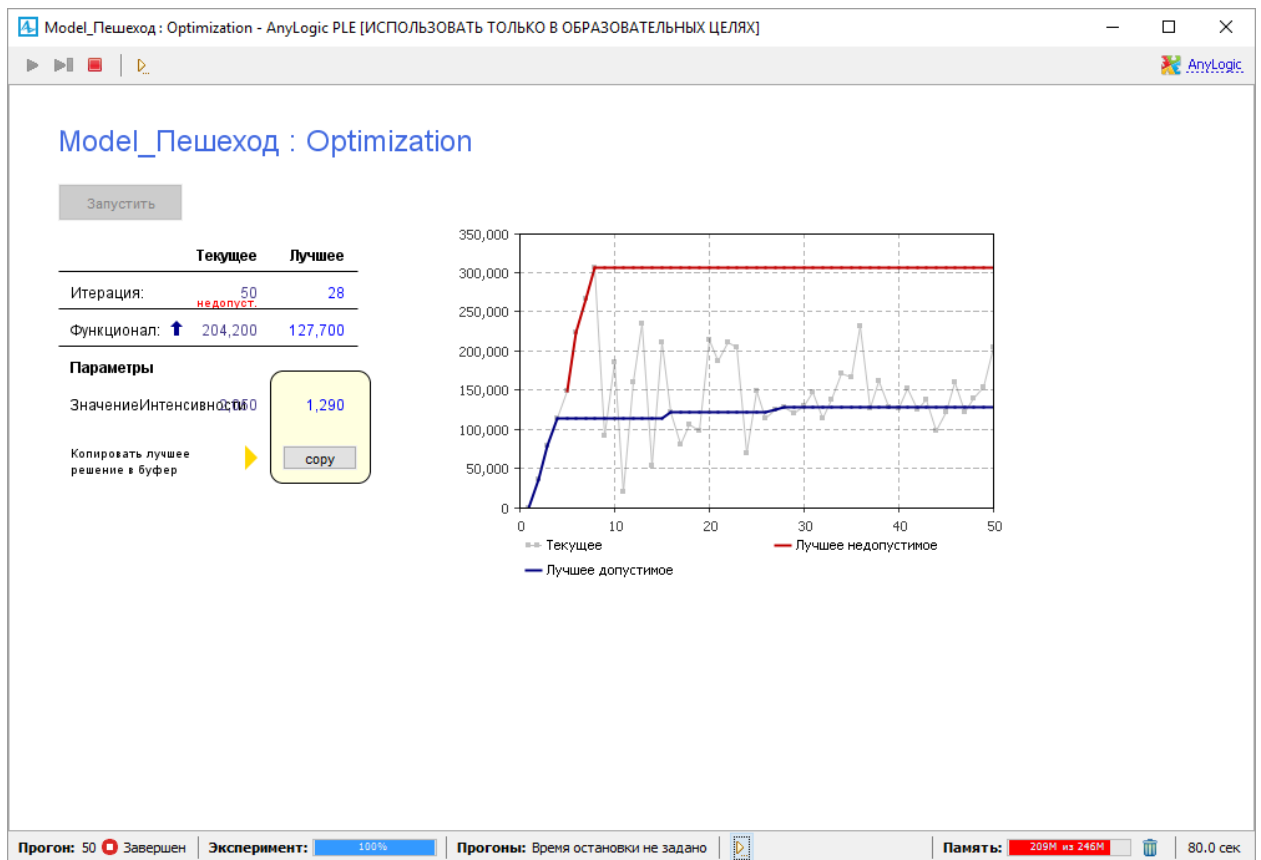


Рисунок 8.9 – Результаты оптимизации

Задание

Проведите оптимизацию модели. Определите интенсивность, при которой потери времени пешеходов в системе будут *минимальны*. В качестве ограничения установить *размер очереди у касс* равным 5 пешеходам.

Контрольные вопросы

1. Как провести оптимизацию пешеходной модели?
2. Какие объекты нужны для оптимизации?
3. Как задать целевую функцию?
4. Как создать эксперимент?
5. Что такое модельное время? Для чего оно применяется?

Лабораторная работа №9. Построение моделей в MS Excel

Теоретические положения

Эффективным средством построения моделей является MS Excel. Входящий в состав данного программного продукта пакет Поиск решения (Solver) позволяет проводить решения задач подобного рода с большим числом переменных и ограничений.

Процесс построения математической модели для решения задачи начинается, как правило, с ответов на следующие вопросы:

- Для определения каких величин должна быть построена модель, т. е. как идентифицировать переменные задачи?
- Какие ограничения должны быть наложены на переменные, чтобы выполнялись условия, характерные для моделируемой системы?
- В чем состоит цель задачи, для достижения которой из всех допустимых значений переменных нужно выбрать те, которые будут соответствовать оптимальному (наилучшему) решению задачи?

После ответа на данные вопросы для построения модели остается только идентифицировать переменные и представить цель и ограничения в виде математических функций этих переменных.

Надлежащий анализ вопросов подобного рода и корректная формулировка математической модели являются центральным звеном решения задач линейной (и не только линейной) оптимизации.

При моделировании в Excel необходимо определить переменные входа-выхода; разработать внутреннюю логику модели путем построения диаграммы влияния; определить переменные решения и показатели эффективности.

Модель должна быть:

1. Логически корректной.
2. Представлять основные альтернативы для сравнения.
3. Удобна для проведения манипуляций, необходимых для анализа.
4. Понятной. Чтобы люди, не участвовавшие в создании модели, могли ее легко понять.
5. Внешнее оформление модели должно быть привлекательным.

Рассмотрим процесс построения и исследования модели на примере движения тела, брошенного под углом к горизонту.

Пример. В процессе тренировок теннисистов используются автоматы по бросанию мячика в определенное место площадки. Необходимо задать автомату необходимую скорость и угол бросания мячика для попадания в мишень определенного размера, находящуюся на известном расстоянии.

Из условия задачи можно сформулировать следующие предположения:

- Мячик мал по сравнению с землей, поэтому его можно считать материальной точкой;
- Изменение высоты мячика мало, поэтому ускорение свободного падения можно считать постоянной величиной $g = 9,8 \text{ м/с}^2$ и движение по оси ОУ можно считать равноускоренным;
- Скорость бросания тела мала, поэтому сопротивлением воздуха можно пренебречь и движение по оси ОХ можно считать равномерным.

Используем известные из физики формулы равномерного и равноускоренного движения. При заданной начальной скорости V_0 и угле бросания A значения координат дальности полета X и высоты Y от времени можно описать следующими формулами:

$$X = V_0 \cos(A) \cdot T,$$

$$Y = V_0 \sin(A) \cdot T - \frac{GT^2}{2}.$$

Пусть мишень высотой H размещается на расстоянии S от автомата. Из первой формулы выражаем время, которое потребуется мячику для преодоления расстояния S . Подставляем полученное значение в формулу для Y и получаем высоту мячика над землей на расстоянии S :

$$L = S \cdot \operatorname{tg}(A) - \frac{GS^2}{2V_0^2 \cos^2(A)}.$$

Формализуем условие попадания мячика в мишень. Попадание произойдет, если $0 \leq L \leq H$ (если $L < 0$ – недолет, если $L > H$ – перелет).

Для построения модели в Excel требуется:

1. Объединить ячейки от A1 до C1.
2. Поместить в объединенную ячейку текст «Движение тела, брошенного под углом к горизонту».
3. Расширить колонки B и C так, чтобы заголовок поместился в ячейках с A1 по C1.
4. Ввести в ячейки A2, A3 и A4 соответственно $V_0=$, $A=$, $G=$.
5. В ячейки C2, C3 и C4 ввести м/сек, град, м/сек² соответственно.
6. Для ячеек B2, B3 и B4 установить формат числовой, установив число десятичных знаков – 1.
7. Ввести в ячейки B2, B3 и B4 соответственно значения 18,0; 35,0; 9,8.
8. Ввести в ячейки A5 – T; B5 – $X = V_0 \cdot \cos(A) \cdot T$; C5 – $Y = V_0 \cdot \sin(A) \cdot T - (G \cdot T^2) / 2$.
9. Выделить ячейки с A6 по C19 и установить числовой формат с числом десятичных знаков – 1.
10. В ячейку A6 ввести число 0,0.
11. Выделить ячейки с A6 по A19 и заполнить их значением времени с интервалом 0,2.
12. В ячейку B6 ввести формулу $=B\$2 \cdot \cos(\text{радианы}(B\$3)) \cdot A6$.
13. В ячейку C6 ввести формулу $=B\$2 \cdot \sin(\text{радианы}(B\$3)) \cdot A6 - (B\$4 \cdot A6^2) / 2$.
14. Скопировать формулы в ячейки B7:B19 и C7:C19 соответственно.
15. Выделить ячейки с A5 по C19 и установить границы таблицы.
16. Визуализировать модель, построив график зависимости координаты Y от координаты X (траекторию движения тела). Поместить график рядом с таблицей.

Исследуем модель и определим с заданной точностью 0,1 – диапазон изменений угла, который обеспечивает попадание в мишень, находящуюся на расстоянии 30 м. И имеющую высоту 1 м, при заданной начальной скорости 18 м/сек.

Воспользуемся надстройкой **Подбор параметра**. Для этого требуется:

1. Установить для ячеек B21:B25 точность один знак после запятой;
2. Ввести в ячейки B21, B22, и B23 значения расстояния до мишени $S = 30$ м, начальной скорости $V_0 = 18$ м/сек и угла $A = 35^\circ$;
3. В ячейку B25 ввести формулу для вычисления высоты мячика над землей на расстоянии для заданных начальных условий: $L = S * \tan(A) - G * S^2 / (2 * V_0^2 * \cos^2(A))$;
4. Выделить ячейку B25 и запустить надстройку **Подбор параметра**. На появившейся диалоговой панели ввести в поле Значения: наименьшую высоту попадания в мишень (то есть 0). В поле *Изменяя значение ячейки*: ввести адрес ячейки, содержащей значение угла (в данном случае \$B\$23). В ячейке B23 появится значение 32,6. Повторить процедуру подбора параметра для максимальной высоты попадания в мишень – в ячейке B23 получим значение 36,1.

Таким образом, исследование компьютерной модели показало, что существует диапазон значений угла бросания от $32,6^\circ$ до $36,1^\circ$, который обеспечивает попадание в мишень высотой 1 м, находящуюся на расстоянии 30 м, мячиком, брошенным со скоростью 18 м/сек.

Задание

Компания получила предложение собрать примерно 15 тыс. калькуляторов по цене \$26,5. Компания оценила, что при использовании существующих производственных мощностей переменные затраты на сборку 1 калькулятора составят \$21. В качестве альтернативного решения компания может заключить субконтракт и поручить некоторые операции сторонней фирме Wizard, сократив тем самым свои удельные затраты до \$18. По контракту компания должна выплатить фирме Wizard фиксированную сумму \$42000. Еще один вариант заключается в том, чтобы взять в аренду работа-сборщика, что позволит компании снизить удельные затраты на сборку до \$11. Создайте модель прогнозирования валовой прибыли для каждого из описанных вариантов. Определите точки безразличия.

Контрольные вопросы

1. Опишите процесс построения математической модели.
2. Какие условия должны быть соблюдены в модели?
3. Что нужно определить при моделировании в Excel?

Лабораторная работа №10.
Построение линейных моделей и их оптимизация
с помощью надстройки «Поиск решения»

Теоретические положения

Экономическая суть методов оптимизации заключается в том, что, исходя из наличия определенных ресурсов, выбирается такой способ их использования (распределения), при котором обеспечивается максимум (или минимум) интересующего показателя.

Задачи нахождения значений параметров, обеспечивающих экстремум функции при наличии ограничений, наложенных на аргументы (независимые переменные), носят общее название задач математического программирования. Среди задач математического программирования самыми простыми и наиболее хорошо изученными являются так называемые задачи линейного программирования (линейной оптимизации). Для них характерно то, что целевая функция линейно зависит от переменных, а также то, что ограничения, накладываемые на независимые переменные, имеют вид линейных равенств или неравенств относительно этих переменных.

Общая задача линейной оптимизации заключается в нахождении максимума (минимума) линейной целевой функции $f(x_1, x_2, \dots, x_n)$ при ограничениях (условиях) вида $g_1(x_1, x_2, \dots, x_n) = b_1$; $h_1(x_1, x_2, \dots, x_n) \leq r_2$; $d_1(x_1, x_2, \dots, x_n) \geq u_1$. Функция называется целевой функцией или критерием оптимальности.

Вектор значений неизвестных, удовлетворяющих условию задачи, называется допустимым решением или допустимым планом задачи линейной оптимизации. Совокупность всех допустимых планов называется множеством допустимых планов. Допустимое решение называется оптимальным, если оно обеспечивает максимальное (или, в зависимости от условий задачи, минимальное) значение целевой функции.

Для того чтобы создать модель в Excel необходимо:

1. Описать словами цель и целевую функцию.
2. Дать словесное описание каждого ограничения. Определить вид ограничения (равенство или неравенство)
3. Дать словесное описание переменных, основываясь на шагах 1–2.

4. Выразить все ограничения через обозначенные переменные решения.
5. Выразить с помощью обозначенных переменных целевую функцию.
6. На основе символической модели линейного программирования (ЛП) создать ее представление в Excel.
7. Проверить полученную табличную модель путем задания различных значений переменных решения с целью выявить возможные очевидные ошибки.
8. Запустить надстройку *Поиск решения*. Задать целевую ячейку, изменяемые ячейки, ограничения.
9. При наличии ошибок перейти к первому этапу.

Задание

Имеется 4 предмета, каждый из которых характеризуется весом и ценой. Нужно выбрать из них такие и столько, чтобы их общий вес не превышал 83, а суммарная цена была максимальной.

Для решения задачи необходимо рассчитать:

- Вес предметов каждого вида (с учетом их количества);
- Стоимость предметов каждого вида (с учетом их количества);
- Суммарный вес;
- Суммарную стоимость (целевая ячейка).

Также необходимо задать ограничения:

- Количество предметов ≥ 0 (в ограничениях можно указывать диапазон, например $\$E\$3:\$E\$6 \geq 0$);
- Предметы не разделяются; общий вес $\leq$ допустимому.

Контрольные вопросы

1. Понятие модели. Примеры.
2. Типы моделей.
3. Этапы принятия управленческих решений. Процесс моделирования. Роль менеджера в моделировании.
4. Построение символических моделей. Характеристики символических моделей.
5. Детерминированные и вероятностные модели.
6. Формализация модели. Использование «черного ящика» на первом этапе формализации модели.

7. Разработка внутренней логики модели с помощью диаграммы влияния. Рекомендации по построению модели.
8. Точки безубыточности и точки безразличия. Отличия и область применения.
9. Анализ чувствительности табличной модели. Использование таблиц подстановок для анализа чувствительности.
10. Свойства количественных (символических) моделей. Правила построения количественных моделей.
11. Модели ЛП. Свойства моделей ЛП.
12. Формализация модели. Задание ограничений.
13. Формализация модели. Задание целевой функции.
14. Целочисленные решения моделей ЛП.
15. Создание моделей ЛП.
16. Область допустимых значений моделей ЛП. Лимитирующие ограничения в моделях ЛП.

Лабораторная работа №11.

Транспортная модель

Теоретические положения

Транспортная задача относится к задачам линейного программирования и, следовательно, может быть решена симплексным методом. В общей постановке транспортная задача состоит в отыскании оптимального плана перевозок некоторого однородного груза от поставщиков к потребителям.

В зависимости от *наличия баланса* транспортные модели делят на:

1. Сбалансированные модели (спрос = предложение).
2. Несбалансированные модели (спрос > предложение или спрос < предложение):

а) Предложение превышает спрос. При этом невостребованный товар в каждом отправном пункте представляется в виде резерва ограничения по объему предложения для данного пункта.

б) Спрос превышает предложение. Если спрос превышает предложение, то модель не имеет допустимых решений. Подходы к моделированию данной ситуации:

- Переписать ограничения для запасов в виде равенств, а все ограничения для спроса записать в виде неравенств $\leq$.
- Ввести в модель фиктивный отправной пункт, запас в котором в точности соответствует разности между общим спросом и предложением. Этот фиктивный отправной пункт делает модель сбалансированной, когда предложение равно спросу. Стоимость доставки из этого пункта в любой пункт назначения равна нулю. Каждая поставка из фиктивного пункта в любой пункт назначения интерпретируется как неудовлетворенный спрос.

В зависимости от *характеристики связей между исходным пунктом и пунктом назначения* транспортные модели делят на *модели с допустимыми путями и модели с недопустимыми путями*.

В случае *недопустимых путей при моделировании* необходимо присвоить удельным затратам по такому маршруту произвольно большое значение F удельным затратам для данного маршрута. Это заставит инструмент *Поиск решения* отказаться от использования данного пути, т. к. при его использовании затраты

будут гораздо больше, чем для всех других допустимых альтернатив.

Пример. Найти самый дешевый способ удовлетворить спрос в четырех пунктах назначения (магазинах), осуществляя поставки из трех исходных (отправных) пунктов (рисунок 11). Затраты на транспортировку даны в таблице 11.

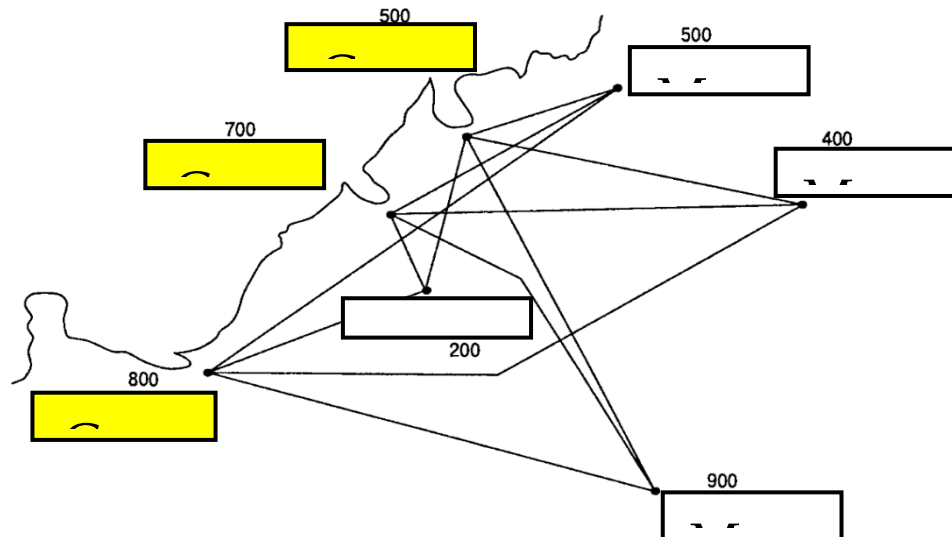


Рисунок 11 – Дислокация пунктов отправления и пунктов назначения

Таблица 11

Удельные затраты на транспортировку				
Пункт отправки	Магазин 1	Магазин 2	Магазин 3	Магазин 4
А	120	130	41	59,50
В	61	40	100	110
С	102,50	90	122	42

Решение. Построим символическую модель задачи. Данная модель является сбалансированной, так как суммарный спрос равен суммарному предложению. Целевая функция – минимизация суммарных затрат на транспортировку со складов в магазины, то есть $120X_{A1} + 130X_{A2} + \dots + 42X_{C4} \rightarrow \min$, где X_{ij} – количество товара, отправленного со склада i в магазин j . В данной модели на переменные решения накладываются следующие ограничения:

а) количество товара, отправленных с любого склада, не может превышать количество товара, имеющегося на складе, то есть, например, для склада А: $X_{A1} + X_{A2} + X_{A3} + X_{A4} \leq 500$;

б) необходимо удовлетворить спрос во всех магазинах, то есть, например, для магазина 1: $X_{A1} + X_{B1} + X_{C1} \geq 400$.
Создадим табличную модель (рисунок 11.1).

	A	B	C	D	E	F	G	H	I
1									
2			Транспортная модель						
3		Склад	Магазин 1	Магазин 2	Магазин 3	Магазин 4			
4	A		120	130	41	59,5			
5	B		61	40	100	110			
6	C		102,5	90	122	42			
7									
8		Склад	Магазин 1	Магазин 2	Магазин 3	Магазин 4	Всего	Доступно	
9	A		0	0	0	0	=СУММ(C9:F9)	500	
10	B		0	0	0	0	=СУММ(C10:F10)	700	
11	C		0	0	0	0	=СУММ(C11:F11)	800	
12		Всего	=СУММ(C9:C11)	=СУММ(D9:D11)	=СУММ(E9:E11)	=СУММ(F9:F11)			
13		Требуется	400	900	200	500			
14									
15		Склад	Магазин 1	Магазин 2	Магазин 3	Магазин 4	Всего		
16	A		=C4*C9	=D4*D9	=E4*E9	=F4*F9	=СУММ(C16:F16)		
17	B		=C5*C10	=D5*D10	=E5*E10	=F5*F10	=СУММ(C17:F17)		
18	C		=C6*C11	=D6*D11	=E6*E11	=F6*F11	=СУММ(C18:F18)		
19		Всего	=СУММ(C16:C18)	=СУММ(D16:D18)	=СУММ(E16:E18)	=СУММ(F16:F18)	=СУММ(C19:F19)		
20									

Рисунок 11.1 – Табличная модель

Оптимизируем табличную модель с помощью надстройки *Поиск решения* (рисунок 11.2).

Рисунок 11.2 – Параметры *Поиска решения*

В результате оптимизации получаем план перевозок с минимальными затратами равными \$121 450 (рисунок 11.3).

	A	B	C	D	E	F	G	H	I
1									
2		Транспортная модель							
3		Склад	Магазин 1	Магазин 2	Магазин 3	Магазин 4			
4	A		\$ 120,0	\$ 130,0	\$ 41,0	\$ 59,5			
5	B		\$ 61,0	\$ 40,0	\$ 100,0	\$ 110,0			
6	C		\$ 102,5	\$ 90,0	\$ 122,0	\$ 42,0			
7									
8		Склад	Магазин 1	Магазин 2	Магазин 3	Магазин 4	Всего	Доступно	
9	A		0	0	200	300	500	<=500	
10	B		0	700	0	0	700	<=700	
11	C		400	200	0	200	800	<=800	
12		Всего	400	900	200	500			
13		Требуется	>=400	>=900	>=200	>=500			
14									
15		Склад	Магазин 1	Магазин 2	Магазин 3	Магазин 4	Всего		
16	A		\$ -	\$ -	\$ 8 200	\$ 17 850	\$26 050		
17	B		\$ -	\$ 28 000	\$ -	\$ -	\$28 000		
18	C		\$ 41 000	\$ 18 000	\$ -	\$ 8 400	\$67 400		
19		Всего	\$ 41 000	\$ 46 000	\$ 8 200	\$ 26 250	\$121 450		

Рисунок 11.3 – Решение

Проанализируем устойчивость модели (рисунок 11.4).

Для этого создадим отчет по устойчивости, который показывает возможности для улучшения значения целевой функции. Особое внимание следует уделить ограничениям, имеющим ненулевое значение теневой цены. Теневая цена показывает, насколько улучшится оптимальное значение целевой функции, если правую часть ограничения увеличить на 1 (для ограничений $\leq$) или уменьшить на 1 (для ограничений $\geq$).

Из анализа ограничения на поставку товаров в магазин 2 следует, что уменьшение правой части ограничения на 1 приведет к снижению затрат на \$107,5. Таким образом, если в данный магазин будет требоваться не 900 единиц товара, а 899, то затраты на транспортировку уменьшатся на \$107,5.

Увеличение правой части ограничения, отражающего наличие товара на складе B, на 1 приведет к снижению затрат на \$67,5. Допустимое увеличение запаса на данном складе – не более чем на 200 единиц.

	A	B	C	D	E	F	G	H
1	Microsoft Excel 14.0 Отчет об устойчивости							
2	Ячейки переменных							
3			Окончательное Приведенн. Целевая функция Допустимое Допустимое					
4	Ячейка	Имя	Значение	Стоимость	Коэффициент	Увеличение	Уменьшение	
5	\$C\$9	A Магазин 1	0	0	120	1E+30	0	
6	\$D\$9	A Магазин 2	0	22,5	130	1E+30	22,5	
7	\$E\$9	A Магазин 3	200	0	41	98,5	41	
8	\$F\$9	A Магазин 4	300	0	59,5	0	17,5	
9	\$C\$10	B Магазин 1	0	8,5	61	1E+30	8,5	
10	\$D\$10	B Магазин 2	700	0	40	8,5	1E+30	
11	\$E\$10	B Магазин 3	0	126,5	100	1E+30	126,5	
12	\$F\$10	B Магазин 4	0	118	110	1E+30	118	
13	\$C\$11	C Магазин 1	400	0	102,5	0	120	
14	\$D\$11	C Магазин 2	200	0	90	22,5	8,5	
15	\$E\$11	C Магазин 3	0	98,5	122	1E+30	98,5	
16	\$F\$11	C Магазин 4	200	0	42	17,5	0	
17								
18	Ограничения							
19			Окончательное	Тень	Ограничение	Допустимое	Допустимое	
20	Ячейка	Имя	Значение	Цена	Правая сторона	Увеличение	Уменьшение	
21	\$C\$12	Всего Магазин 1	400	120	400	0	300	
22	\$D\$12	Всего Магазин 2	900	107,5	900	0	200	
23	\$E\$12	Всего Магазин 3	200	41	200	0	200	
24	\$F\$12	Всего Магазин 4	500	59,5	500	0	300	
25	\$G\$9	A Всего	500	0	500	1E+30	0	
26	\$G\$10	B Всего	700	-67,5	700	200	0	
27	\$G\$11	C Всего	800	-17,5	800	300	0	

Рисунок 11.4 – Отчет по устойчивости

Задание

У компании есть 3 склада, на которых она отгружает товар в 3 торговые точки. Спрос на продукцию составляет 100 банок в торговой точке 1, 250 банок – в торговой точке 2 и 150 банок – в торговой точке 3. Запас продукции на складе 1 составляет 50 банок, на складе 2 – 275, на складе 3 – 175. Стоимость транспортировки одной банки продукции со складов в торговые точки приводится в таблице 11.1.

Таблица 11.1

Удельные затраты на транспортировку 1 банки

Пункт отправки	Точка 1	Точка 2	Точка 3
Склад 1	5	7	6
Склад 2	8	9	10
Склад 3	4	3	11

Постройте модель ЛП, позволяющую определить, сколько продукции необходимо отправить с каждого склада в каждую торговую точку, чтобы удовлетворить существующий спрос с минимальными затратами.

Контрольные вопросы

1. На какие составляющие в зависимости от наличия баланса делят транспортные модели?
2. Подходы к моделированию транспортной модели.
3. Как оптимизировать табличную модель с помощью надстройки «Поиск решения»?
4. Что отображает отчет по устойчивости?

Лабораторная работа №12. Модель назначений

Теоретические положения

В моделях назначений решается задача нахождения оптимального распределения n неделимых агентов по n заданиям. Это может быть распределение продавцов по отделам, компьютеров по сетям и т. д. Распределяемые агенты являются неделимыми, т. е. один агент не может заниматься несколькими задачами. Модель назначений является разновидностью транспортной модели. Она отличается только тем, что в ней единица предложения не может распределяться по нескольким местам назначения.

Пример. Минимизировать затраты на командировки, связанные с проверкой работы четырех заводов. Затраты на командировку приведены в таблице 12.

Таблица 12

Удельные затраты на командировку

Специализация инспектора	Завод 1	Завод 2	Завод 3	Завод 4
Финансы (Ф)	24	10	21	11
Маркетинг (М)	14	22	10	15
Производство (П)	15	17	20	19
Персонал (Пл)	11	19	14	13

Решение. Построим символическую модель задачи. Данная модель является сбалансированной, так как суммарный спрос равен суммарному предложению.

Целевая функция – минимизация суммарных затрат на командировки $24X_{Ф1} + 14X_{М1} + \dots + 13X_{Пл4} \rightarrow \min$, где X_{ij} – число инспекторов специализации i (Ф, М, П, Пл), направленных на завод j (1, 2, 3, 4).

В данной модели на переменные решения накладываются следующие ограничения:

а) число инспекторов на каждом заводе не должно превышать 1;

б) один инспектор отправляется только на один завод.

Создадим табличную модель (рисунок 12).

	A	B	C	D	E	F	G	H
1								
2		Модель назначений						
3		Удельные затраты/ вице-президент	Завод1	Завод2	Завод3	Завод4		
4		по финансам	24	10	21	11		
5		по маркетингу	14	22	10	15		
6		по производству	15	17	20	19		
7		по персоналу	11	19	14	13		
8								
9		Назначение вице-президента	Завод1	Завод2	Завод3	Завод4	Всего	Имеется
10		по финансам	0	0	0	0	=СУММ(C10:F10)	1
11		по маркетингу	0	0	0	0	=СУММ(C11:F11)	1
12		по производству	0	0	0	0	=СУММ(C12:F12)	1
13		по персоналу	0	0	0	0	=СУММ(C13:F13)	1
14		Всего	=СУММ(C10:C13)	=СУММ(D10:D13)	=СУММ(E10:E13)	=СУММ(F10:F13)		
15		Необходимо	1	1	1	1		
16								
17		Суммарные затраты/ вице-президент	Завод1	Завод2	Завод3	Завод4	Всего	
18		по финансам	=C4*C10	=D4*D10	=E4*E10	=F4*F10	=СУММ(C18:F18)	
19		по маркетингу	=C5*C11	=D5*D11	=E5*E11	=F5*F11	=СУММ(C19:F19)	
20		по производству	=C6*C12	=D6*D12	=E6*E12	=F6*F12	=СУММ(C20:F20)	
21		по персоналу	=C7*C13	=D7*D13	=E7*E13	=F7*F13	=СУММ(C21:F21)	
22		Всего	=СУММ(C18:C21)	=СУММ(D18:D21)	=СУММ(E18:E21)	=СУММ(F18:F21)	=СУММ(C22:F22)	

Рисунок 12 – Табличная модель

Оптимизируем табличную модель с помощью надстройки *Поиск решения* (рисунок 12.1).

Рисунок 12.1 – Параметры *Поиска решения*

В результате оптимизации получаем план перевозок с минимальными затратами равными \$48 (рисунок 12.2).

Модель назначений						
Удельные затраты/ вице-президент	Завод1	Завод2	Завод3	Завод4		
по финансам	\$ 24	\$ 10	\$ 21	\$ 11		
по маркетингу	\$ 14	\$ 22	\$ 10	\$ 15		
по производству	\$ 15	\$ 17	\$ 20	\$ 19		
по персоналу	\$ 11	\$ 19	\$ 14	\$ 13		
Назначение вице-президента	Завод1	Завод2	Завод3	Завод4	Всего	Имеется
по финансам	0	1	0	0	1	<=1
по маркетингу	0	0	1	0	1	<=1
по производству	1	0	0	0	1	<=1
по персоналу	0	0	0	1	1	<=1
Всего	1	1	1	1		
Необходимо	>=1	>=1	>=1	>=1		
Суммарные затраты/ вице-президент	Завод1	Завод2	Завод3	Завод4	Всего	
по финансам	-	\$ 10	-	-	\$ 10	
по маркетингу	-	-	\$ 10	-	\$ 10	
по производству	\$ 15	-	-	-	\$ 15	
по персоналу	-	-	-	\$ 13	\$ 13	
Всего	\$ 15	\$ 10	\$ 10	\$ 13	\$ 48	

Рисунок 12.2 – План перевозок с минимальными затратами

Задание

Развезти сборный груз 6 тонн в каждый населенный пункт с помощью парка транспортных средств. В каждый населенный пункт отправляется 1 транспортное средство (ТС). Затраты на транспортировку приведены в таблице 12.1.

Таблица 12.1

Удельные затраты

Гос. номер ТС	Город 1	Город 2	Город3
A654PP	1500	10000	3400
P657TM	1400	10500	3400
B543PP	1600	10700	3300
K121BC	1540	9800	3200

Контрольные вопросы

1. Графический метод решения задач ЛП. Задание целевой функции и ограничений. Оптимизация модели. Зависимость оптимального решения от угла наклона целевой функции.

2. Неограниченные и недопустимые модели.
3. Отчет по устойчивости. Анализ изменяемых ячеек. Анализ ограничений.
4. Применение моделей ЛП. Транспортная модель. Свойства транспортной модели ЛП.
5. Варианты транспортной модели.
6. Оптимизация несбалансированной транспортной модели. Оптимизация транспортной модели с недопустимыми путями.
7. Применение моделей ЛП. Модель назначений. Отличия модели назначений от транспортной модели.
8. Свойства модели назначений. Возможности поиска решения методом перебора.

Лабораторная работа №13.
Поиск кратчайшего пути.
Модель планирования транспортной сети

Теоретические положения

Задачу *поиска кратчайшего пути* можно решить, построив сетевую модель (рисунок 13). Эта задача сводится к поиску самого короткого пути (цепи) между двумя точками (вершинами) на графе, в которой минимизируется сумма весов ребер, составляющих путь.

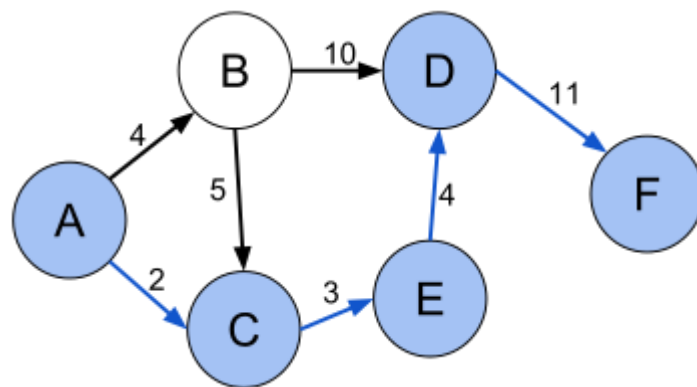


Рисунок 13 – Кратчайший путь (A, C, E, D, F) между вершинами A и F во взвешенном ориентированном графе

Модель планирования транспортной сети позволяет оценить пропускную способность сети, то есть найти максимальное количество (нефти, денежных средств, пакетов Internet, транспортных средств), которые можно пропустить через данную физическую сеть (от источника до стока) за единицу времени. Отличительные особенности данной модели состоят в следующем:

1. существует один узел-источник (вход в систему) и один узел стока (выход из системы);
2. поток в единицу времени по каждой дуге сети ограничен ее пропускной способностью;
3. для узлов пропускные способности не задаются. Однако, для каждого узла (кроме исходного и конечного) должен выполняться баланс потоков: поток, выходящий из узла, должен быть равен потоку, входящему в данный узел.

Пример. Определить максимально возможное количество транспортных средств, которое может перемещаться по объездной дороге из пункта 1 (начало объезда) до пункта 6 (конец объезда) в

единицу времени. Пропускные способности зависят от направления движения (рисунок 13.1).

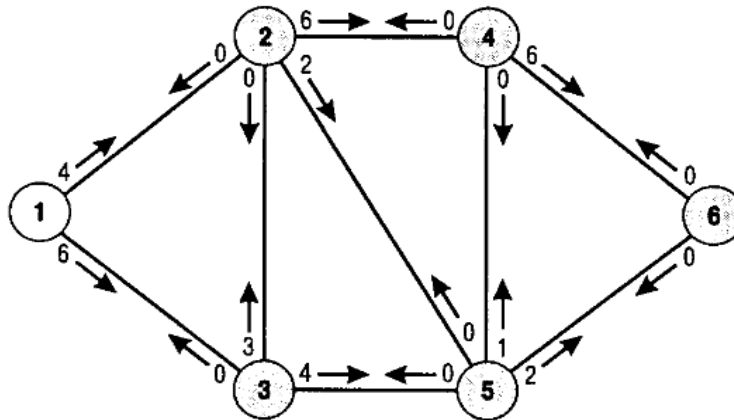


Рисунок 13.1 – Сетевая диаграмма

Решение. Создадим табличную модель (рисунок 13.2).

	A	B	C	D	E	F	G	H	I
1		Модель максимизации потока							
2		Пропускная способность	Узел 1	Узел 2	Узел 3	Узел 4	Узел 5	Узел 6	
3		Узел 1	0	4	6	0	0	0	
4		Узел 2	0	0	0	6	2	0	
5		Узел 3	0	3	0	0	4	0	
6		Узел 4	0	0	0	0	0	6	
7		Узел 5	0	0	0	0	0	2	
8									
9		Поток	Узел 1	Узел 2	Узел 3	Узел 4	Узел 5	Узел 6	Итого
10		Узел 1	0	0	0	0	0	0	=СУММ(C10:H10)
11		Узел 2	0	0	0	0	0	0	=СУММ(C11:H11)
12		Узел 3	0	0	0	0	0	0	=СУММ(C12:H12)
13		Узел 4	0	0	0	0	0	0	=СУММ(C13:H13)
14		Узел 5	0	0	0	0	0	0	=СУММ(C14:H14)
15		Итого	=СУММ(D10:D14)		=СУММ(E10:E14)	=СУММ(F10:F14)	=СУММ(G10:G14)	=СУММ(H10:H14)	
16		Всего	=СУММ(D10:D14)		=СУММ(E10:E14)	=СУММ(F10:F14)	=СУММ(G10:G14)	=СУММ(H10:H14)	
17		Необходимо	0		0	0	0	0	

Альтернативная целевая ячейка

Рисунок 13.2 – Табличная модель

В данной модели диапазон ячеек C10:H14 – это переменные решения, потоки из одного узла в другой. Они не должны превышать соответствующие пропускные способности, заданные в ячейках C3:H7; C17:G17 – результирующий поток в каждом промежуточном узле должен быть равен нулю.

Целевая функция – максимизация количества транспортных средств, выходящих из узла 1. Альтернативная целевая функция –

максимизация количества транспортных средств, входящих в узел 6.

Оптимизируем табличную модель с помощью надстройки *Поиск решения* (рисунок 13.3).

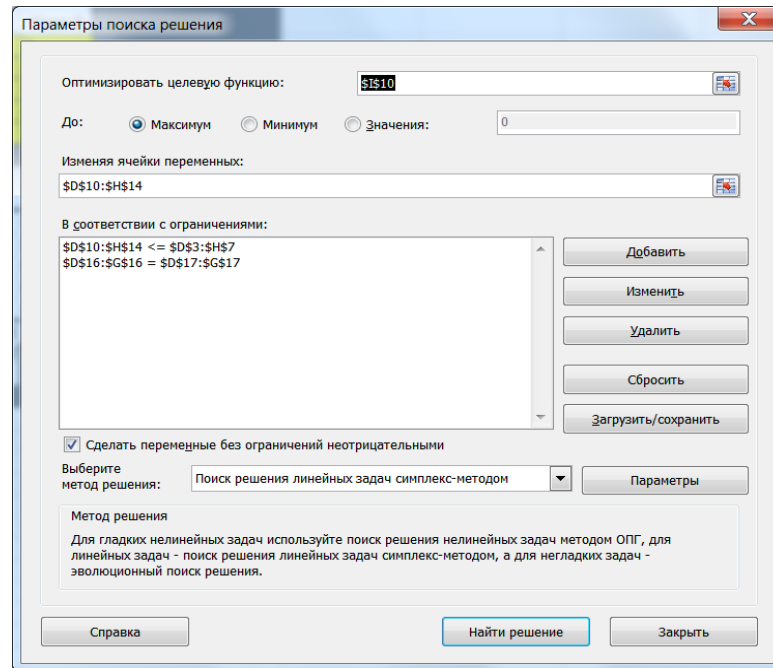


Рисунок 13.3 – Параметры *Поиска решения*

В результате оптимизации получаем план перевозок генераторов в пункты строительства с минимальными затратами, равными \$1615 (рисунок 13.4).

	A	B	C	D	E	F	G	H	I	J
1		Модель максимизации потока								
2		Пропускная способность	Узел 1	Узел 2	Узел 3	Узел 4	Узел 5	Узел 6		
3		Узел 1		4	6					
4		Узел 2				6	2			
5		Узел 3		3			4			
6		Узел 4						6		
7		Узел 5						2		
8										
9		Поток	Узел 1	Узел 2	Узел 3	Узел 4	Узел 5	Узел 6	Итого	
10		Узел 1		4	4				8	
11		Узел 2				6	1		7	
12		Узел 3		3			1		4	
13		Узел 4						6	6	
14		Узел 5						2	2	
15		Итого		7	4	6	2	8		
16		Всего		0	0	0	0			
17		Необходимо		=0	=0	=0	=0			

Рисунок 13.4 – план перевозок генераторов в пункты строительства с минимальными затратами

Задание

Найти кратчайший путь из узла 1 в узел 7 (рисунок 13.5).
Найти максимальный поток из узла 2 в узел 6. Насколько нужно уменьшить «цену» дуги от узла 1 до узла 3, чтобы эта дуга стала частью кратчайшего пути из узла 1 в узел 7?

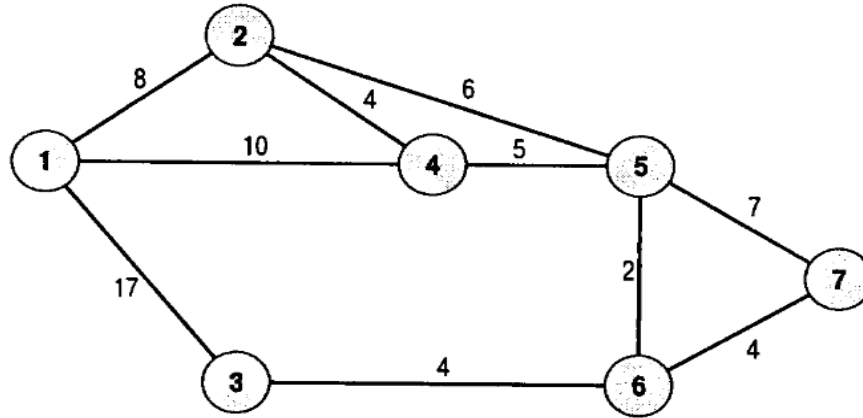


Рисунок 13.5 – Сетевая модель

Контрольные вопросы

1. Применение моделей ЛП. Модель управления запасами.
2. Причины создания материальных запасов.
3. Виды запасов.
4. Факторы, влияющие на модель управления запасами.
5. Обобщенная модель управления запасами с учетом невыполненных заявок.
6. Обобщенная модель управления запасами с потерей невыполненных заявок.
7. Варианты моделей управления запасами.
8. Упрощения и предположения, используемые при моделировании системы управления запасами.
9. Применение моделей ЛП. Модель перевозок.
10. Свойства модели перевозок. Отличия от транспортной модели.
11. Построение символической модели перевозок.
12. Поиск кратчайшего пути. Существующие алгоритмы поиска кратчайшего пути.
13. Постановка задачи о кратчайшем пути. Практическое применение. Свойства модели.
14. Модель планирования транспортной сети.

СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Цель самостоятельной работы обучающихся – получить новые знания по дисциплине «Имитационное моделирование».

Самостоятельная работа необходима для формирования у обучающихся способности самостоятельно решать задачи профессиональной деятельности, формирования умения и навыков планирования времени, формирования стремления развиваться и совершенствоваться.

Виды самостоятельной работы обучающихся указаны в таблице 1.

Таблица 1.

Виды самостоятельной работы

№ п/п	Вид СРС
1	Самостоятельное изучение тем и подготовка к контрольным вопросам
2	Подготовка к лабораторным занятиям

УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Основная литература

1. Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул ; под ред. Л. Г. Гагариной. – Москва : ФОРУМ : ИНФРА-М, 2025. – 400 с. – (Среднее профессиональное образование). – ISBN 978-5-8199-0812-9. – URL: <https://znanium.ru/catalog/product/2183867> (дата обращения: 29.04.2026). – Режим доступа: для авториз. пользователей.

Дополнительная литература

1. Спирина, М. С. Теория вероятностей и математическая статистика: учебник для студентов учреждений среднего профессионального образования / М. С. Спирина, П. Спирин. – 6-е изд., стер. – Москва: Академия, 2023. – 352 с. – <https://academia-moscow.ru/reader/?id=715105> (дата обращения: 29.04.2026).
2. Губарь, Ю. В. Введение в математическое моделирование : учебное пособие для СПО / Ю. В. Губарь. – 2-е изд. – Саратов : Профобразование, 2024. – 178 с. – ISBN 978-5-4488-0991-0 // Электронный ресурс цифровой образовательной среды СПО PROОбразование: [сайт]. – URL: <https://profspo.ru/books/139744> (дата обращения: 29.04.2026). – Режим доступа: для авторизир. пользователей
3. Боев, В. Д. Компьютерное моделирование систем: учебник для среднего профессионального образования / В. Д. Боев. – Москва: Юрайт, 2026. – 253 с. – (Профессиональное образование). – ISBN 978-5-534-10710-4 // Образовательная платформа Юрайт [сайт]. – URL: <https://urait.ru/bcode/598783> (дата обращения: 29.04.2026)

Программное обеспечение и интернет-ресурсы

- Университетская библиотека онлайн. Режим доступа: www.biblioclub.ru
- Лань. Режим доступа: <http://e.lanbook.com>
- Электронно-библиотечная система Znanium.com. Режим доступа: <https://znanium.ru/>

- Электронная библиотека издательства Юрайт <https://biblio-online.ru/catalog/spo>
- Электронный ресурс цифровой образовательной среды СПО PROОбразование. Режим доступа: <https://profspo.ru>

.

Содержание

Лабораторная работа №1. Построение дискретно-событийных моделей. Основные блоки потоковой диаграммы. Модели прибытия.	3
Лабораторная работа №2. Построение дискретно-событийных моделей. Модели прибытия с очередью. Блок Queue.....	9
Лабораторная работа №3. Построение дискретно-событийных моделей. Блок Delay.....	13
Лабораторная работа №4. Параметры агентов. Сортировка агентов в зависимости от значения параметра.	17
Лабораторная работа №5. Визуализация дискретно-событийной модели.	24
Лабораторная работа №6. Сбор статистики. Оптимизация дискретно-событийной модели.	35
Лабораторная работа №7. Построение пешеходной модели.	44
Лабораторная работа №8. Оптимизация пешеходной модели.	57
Лабораторная работа №9. Построение моделей в MS Excel.....	63
Лабораторная работа №10. Построение линейных моделей и их оптимизация с помощью надстройки «Поиск решения».	68
Лабораторная работа №11. Транспортная модель.	71
Лабораторная работа №12. Модель назначений.	77
Лабораторная работа №13. Поиск кратчайшего пути. Модель планирования транспортной сети.	81
СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ	87
УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ	87