

Илья Григорьев

The logo consists of a stylized blue 'A' character followed by the word 'anylogic' in a lowercase sans-serif font, and the Russian phrase 'за три дня' (in three days) below it.

Anylogic  
за три дня

Практическое пособие  
по имитационному моделированию

2022

## **2 AnyLogic 8 за три дня**

### ***О книге***

Это первое практическое пособие по программе AnyLogic от ее разработчиков. AnyLogic является уникальным программным продуктом, поддерживающим все три методологии имитационного моделирования: системную динамику, дискретно-событийное и агентное моделирование, а также позволяющим создавать многоподходные модели.

Книга содержит подробные пошаговые инструкции по построению трех практических примеров: агентной модели потребительского рынка, системно-динамической модели распространения эпидемии и дискретно-событийной модели небольшого заводского цеха.

Книга может использоваться как практическое пособие по работе с AnyLogic, знакомящее пользователя с базовым функционалом продукта.

### ***Об авторе***

Илья Григорьев занимает пост руководителя отдела обучения в Компании AnyLogic. Илья Григорьев является автором справочной документации по AnyLogic, а также автором программ обучения AnyLogic, проводимых по всему миру. Он имеет опыт работы консультантом по созданию имитационных моделей для нескольких коммерческих организаций. Илья Григорьев работает в Компании AnyLogic уже почти двадцать лет и знает практически все об имитационном моделировании и об AnyLogic.

# Оглавление

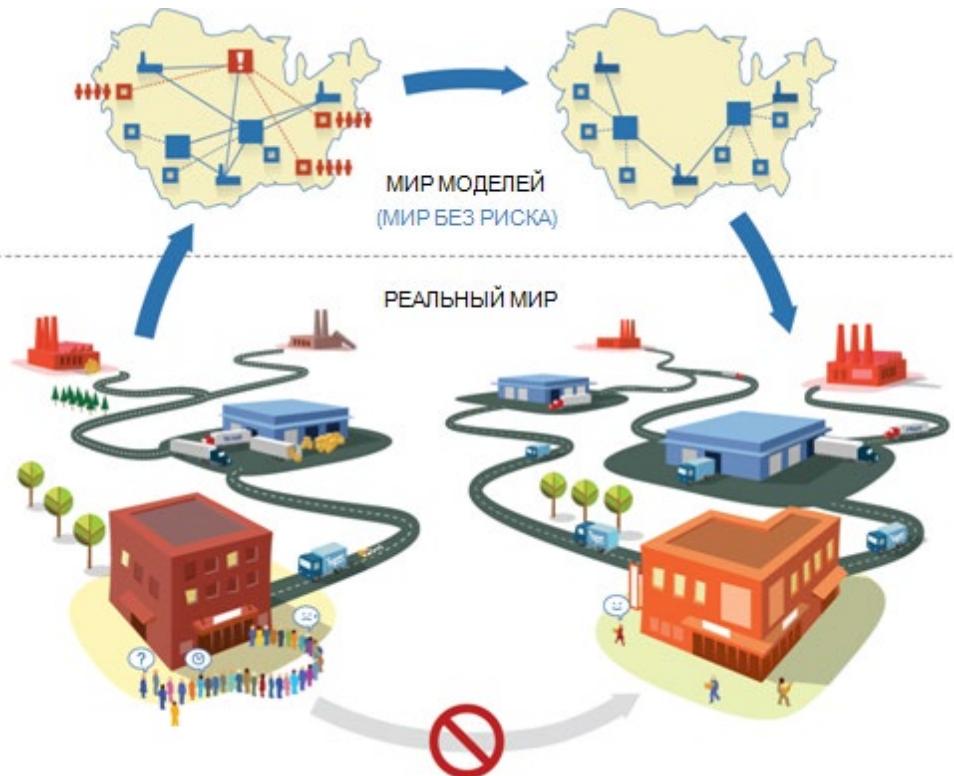
Имитационное моделирование.....	5
Установка и активация AnyLogic.....	14
Агентное моделирование .....	18
Модель потребительского рынка .....	21
Фаза 1. Создание популяции агентов .....	21
Фаза 2. Задание поведения потребителей .....	40
Фаза 3. Добавление графика для визуализации результатов моделирования	53
Фаза 4. Добавление эффекта рекомендаций .....	64
Фаза 5. Учет повторных продаж продукта .....	71
Фаза 6. Учет времени доставки продукта .....	75
Фаза 7. Моделирование отказов от покупки товара .....	82
Фаза 8. Сравнение прогонов модели.....	93
Системная динамика.....	101
Модель распространения эпидемии .....	103
Фаза 1. Создание диаграммы потоков и накопителей.....	103
Фаза 2. Добавление графика для визуализации динамики процесса .....	116
Фаза 3. Эксперимент варьирования параметров .....	122
Фаза 4. Калибровка параметров модели.....	129
Дискретно-событийное моделирование в AnyLogic .....	139
Модель заводского цеха .....	141
Фаза 1. Создание простой модели.....	141
Фаза 2. Добавление ресурсов .....	160
Фаза 3. Создание трехмерной анимации .....	168
Фаза 4. Моделирование доставки поддонов фурами.....	182
Пешеходное моделирование. Модель аэропорта.....	202

#### **4 AnyLogic 8 за три дня**

Фаза 1. Задание потока пешеходов .....	203
Фаза 2. Создание 3D анимации.....	213
Фаза 3. Моделирование предполетного досмотра пассажиров .....	221
Фаза 4. Добавление стоек регистрации .....	230
Фаза 5. Моделирование посадки на самолет.....	242
Фаза 6. Считывание данных о рейсах из файла MS Excel .....	251
Заключение .....	271
Список литературы .....	272

# Имитационное моделирование

Моделирование является одним из способов решения практических задач. Зачастую решение проблемы нельзя найти путем проведения натурных экспериментов: строить новые объекты, разрушать или вносить изменения в уже имеющуюся инфраструктуру может быть слишком дорого, опасно или просто невозможно. В таких случаях мы строим модель реальной системы, то есть описываем ее на языке моделирования. Данный процесс подразумевает переход на определенный уровень абстракции: опуская несущественные детали, мы учтываем только то, что считаем важным. Система в реальном мире всегда сложнее своей модели.



Моделирование

## 6 AnyLogic 8 за три дня

- ◆ Все этапы разработки модели – проекция реального мира в мир моделей, выбор уровня абстракции и выбор языка моделирования – менее стандартизированы, чем процесс использования моделей для решения задач. Моделирование до сих пор больше искусство, чем наука.

После создания модели – а иногда и в процессе разработки – мы начинаем исследовать структуру и понимать поведение системы, проверять, как она ведет себя при определенных условиях, сравнивать различные сценарии и оптимизировать ее. Когда оптимальное решение будет найдено, мы сможем применить его в реальном мире.

- ◆ В сущности, моделирование является поиском решения задачи в защищенном от риска мире моделей, в котором мы можем ошибаться, отменять операции, возвращаться в прошлое и начинать все сначала.

### *Типы моделей*

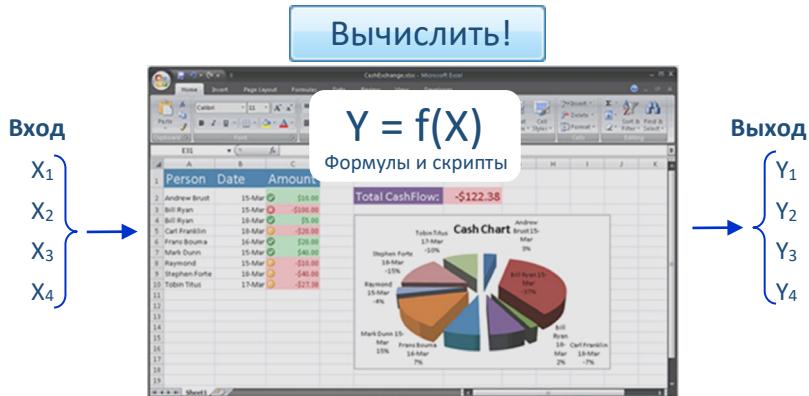
Есть много типов моделей, включая ментальные модели, которыми все мы пользуемся, чтобы понять, как устроен мир вокруг нас: друзья, семья, коллеги, город, в котором мы живем. Все наши решения - что следует сказать своему ребенку, что съесть на завтрак, за кого голосовать и в какой ресторан сходить на выходных, – основаны на ментальных моделях.

Мощным инструментом моделирования являются компьютеры, ведь они предоставляют легко управляемый виртуальный мир, в котором мы можем создать практически все, что способны представить. Конечно, существует множество различных типов компьютерных моделей: от электронных таблиц, позволяющих моделировать расходы, до сложных инструментов имитационного моделирования, которые помогают исследовать динамические системы, например, потребительский рынок или зону боевых действий.

### *Сравнение аналитического и имитационного моделирования*

Если вы спросите у сотрудников отдела стратегического планирования глобальной корпорации (равно как и у сотрудников отделов прогнозирования продаж, логистики, маркетинга, управления проектами и т.д.), какой инструмент моделирования они предпочитают использовать в своей работе, то самым

популярным ответом будет: «Excel». У программы Microsoft Excel есть неоспоримые преимущества: она широко распространена и крайне проста в использовании.



### Аналитическая модель (таблица MS Excel)

Суть технологии моделирования с использованием электронных таблиц крайне проста: вы вводите данные модели в одни ячейки и получаете выходные данные в других. Входные и выходные данные связаны формулами. Для задания дополнительной логики вы можете добавить в таблицу макросы. Различные надстройки позволяют вам выполнять эксперименты варьирования параметров, оптимизации или Монте-Карло.

Однако существует множество задач, для которых аналитическое (основанное на формулах) решение крайне сложно найти, а иногда оно и вовсе отсутствует. К таким задачам относятся в том числе и *динамические системы*, которым свойственно:

- Нелинейное поведение
- "Память"
- Неочевидные зависимости между переменными
- Причинно-следственные связи
- Неопределенность и большое количество параметров.

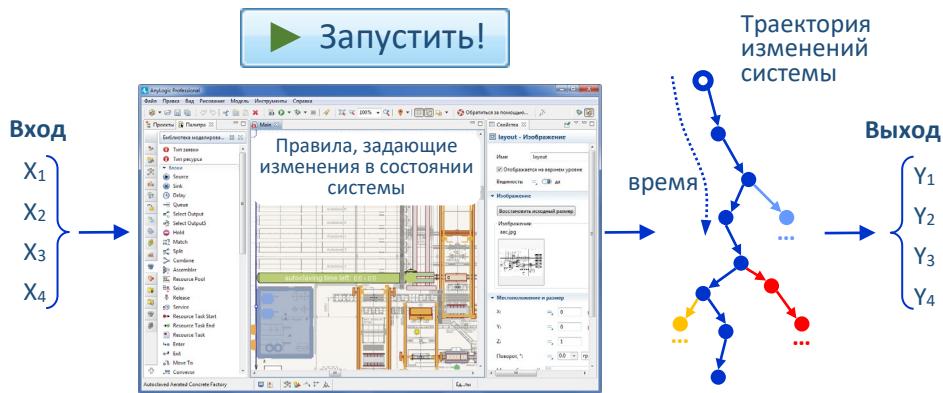
В большинстве случаев практически невозможно найти точные формулы и тем более построить ментальную модель такой системы.

## 8 AnyLogic 8 за три дня

Давайте рассмотрим задачу оптимизации автопарка логистической компании. Для ее решения необходимо учесть такие факторы, как расписание поездок, время погрузки и разгрузки товара, ограничения на время доставки и вместимость терминалов. При этом доступность транспортного средства в определенном месте в определенное время зависит от цепочки предшествующих событий, а выбор нового пункта назначения для свободной машины требует анализа связанных между собой будущих событий. Решить такую задачу средствами MS Excel практически невозможно.

- Формулы, хорошо описывающие статические зависимости между переменными, как правило, плохо подходят для систем с динамическим поведением. Поэтому для анализа динамических систем мы используем другую технологию – имитационное моделирование.

Имитационная модель всегда является выполняемой моделью: вы запускаете ее, и она строит для вас траекторию изменений состояния системы. Можно сказать, что имитационная модель – это набор правил, согласно которым система переходит из одного состояния в другое. Правила могут задаваться самыми различными способами, например, дифференциальными уравнениями, диаграммами состояний, диаграммами процессов, расписаниями. Выходные данные модели всегда можно проанализировать прямо по ходу моделирования.



Имитационная модель

Имитационные модели разрабатываются с помощью специализированного программного обеспечения, в котором используются различные языки моделирования. Для овладения навыком моделирования вам потребуется

обучение, однако затраченное время и усилия окупятся, когда ваша модель предоставит высококачественный анализ сложной динамической системы.

Многие опытные пользователи MS Excel, владеющие навыками программирования, пытаются моделировать динамические системы с помощью электронных таблиц. В попытках учесть все больше и больше деталей, они неизбежно начинают воспроизводить функционал имитационных инструментов средствами в MS Excel. Получившиеся в результате модели работают очень медленно, в них невозможно разобраться, и вскоре выкидываются за ненадобностью.

Практически все перечисленные выше особенности динамических систем невозможно отразить в аналитической модели. Даже если общую конфигурацию системы удалось описать формулами, ее малейшее изменение может сделать эти формулы неверными, и для их исправления вам понадобится помочь профессионального математика.

### **Преимущества имитационного моделирования**

Можно выделить шесть основных преимуществ имитационного моделирования:

1. Имитационные модели позволяют анализировать системы и находить решения в тех случаях, когда такие методы, как аналитические вычисления и линейное программирование не справляются с задачей.
2. После того, как вы определитесь с уровнем абстракции, разрабатывать имитационную модель будет гораздо проще, чем аналитическую, поскольку процесс создания модели будет инкрементальным и модульным.
3. Структура имитационной модели естественным образом отображает структуру моделируемой системы.
4. Имитационная модель позволяет вам отслеживать все объекты системы, учтенные в выбранном уровне абстракции, добавлять метрики и проводить статистический анализ.
5. Одним из главных преимуществ имитационного моделирования является возможность проигрывать модель во времени и анимировать ее поведение. Анимация будет неоспоримым преимуществом при демонстрации модели и может оказаться полезной для верификации модели и нахождения ошибок.

6. Имитационные модели намного убедительнее электронных таблиц. Если вы используете имитационное моделирование, то при презентации проекта у вас будет явное преимущество перед теми, у кого на руках только цифры и решение, полученное из «черного ящика».

## Области применения имитационного моделирования

Имитационное моделирование доказало свою успешность во многих областях применения. Появление новых методов моделирования и рост вычислительной мощности компьютеров позволяет утверждать, что количество этих областей будет только расти.



## Области применения имитационного моделирования

На рисунке выше вы можете видеть распределение областей применения имитационного моделирования соответственно используемым в моделях уровням абстракции.

В нижней части рисунка располагаются модели физического уровня, в которых объекты реального мира моделируются максимально подробно. На этом уровне мы учитываем физическое взаимодействие, размеры, скорости, расстояния. Антиблокировочная система тормозов автомобиля, эвакуация болельщиков со

стадиона, движение на регулируемом перекрестке, взаимодействие солдат на поле боя – все эти примеры требуют низкого уровня абстракции при их моделировании.

Модели, расположенные в верхней части схемы, более абстрактны и чаще всего оперируют обобщенными понятиями, такими как совокупность потребителей или статистика уровня занятости, а не отдельными объектами. Так как взаимодействие между объектами происходит на высоком уровне, такие модели помогают понять взаимосвязи в системе без необходимости моделировать промежуточные шаги, например, изучить влияние вложений в рекламу на продажи продукта компании.

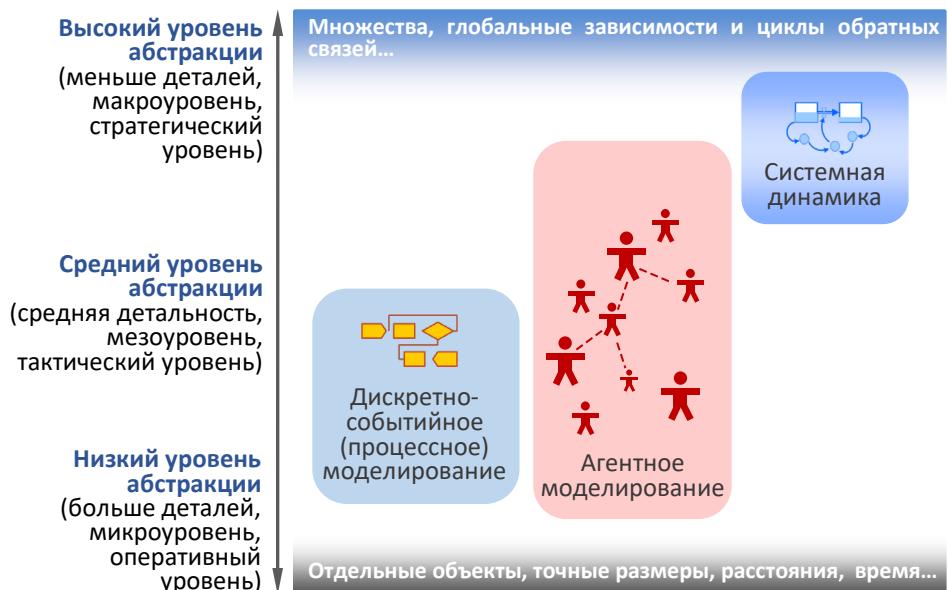
Другие модели имеют средний уровень абстракции. Например, при моделировании отделения скорой помощи необходимо учитывать реальные размеры помещения, чтобы узнать, как долго пациент будет идти от приемной до рентгеновского кабинета. При этом, предположив, что помещение не переполнено, мы можем исключить из рассмотрения физическое взаимодействие между людьми.

Разрабатывая модель бизнес-процесса или работы центра обработки звонков, мы моделируем последовательность и длительность операций, а не место, в котором они происходят. В модели грузоперевозок мы учитываем скорость грузовика или поезда, но в модели цепочки поставок на более высоком уровне мы просто считаем, что доставка заказа занимает от семи до десяти дней.

- ◆ От правильности выбора уровня абстракции зависит успешность проекта моделирования. После того, как вы решите, что включать в модель, а что оставить за пределами уровня абстракции, выбрать метод моделирования будет уже не так сложно.
- ◆ Нормально и даже ожидаемо, что в процессе разработки модели вам порой придется пересматривать выбранный уровень абстракции. В большинстве случаев вы начинаете с высокого уровня, а позже добавляете необходимые детали.

## Три метода имитационного моделирования

В современном имитационном моделировании используются три подхода (методологии): дискретно-событийное моделирование, агентное моделирование и системная динамика.



### Методы имитационного моделирования

В имитационном моделировании под *методом* понимается некая основа, которую мы используем, чтобы «перевести» систему из реального мира в мир моделей. Метод предполагает определенный язык, «*положения и условия*» для разработки модели. На данный момент существует три метода:

- *Системная динамика*
- *Дискретно-событийное моделирование*
- *Агентное моделирование*

Каждый метод применяется в некотором диапазоне уровней абстракции. Системная динамика предполагает очень высокий уровень абстракции и, как правило, используется для стратегического моделирования. Дискретно-событийное моделирование поддерживает средний и низкий уровни абстракции. Между ними находятся агентные модели, которые могут быть как очень детализированными, когда агенты представляют физические объекты, так и предельно абстрактными, когда с помощью агентов моделируются конкурирующие компании или правительства государств.

Прежде чем выбрать метод моделирования, следует тщательно исследовать моделируемую систему и цели моделирования. На схеме ниже показано, что

конкретная задача, стоящая перед разработчиком, во многом определяет подход к моделированию супермаркета. Разработчик может построить диаграмму процессов, в которой участвуют покупатели-заявки и кассиры-ресурсы, или агентную модель, в которой на покупателей-агентов влияет реклама и общение между собой и с сотрудниками-агентами компании, или диаграмму потоков и накопителей, в которой продажи связаны с рекламой, качеством сервиса, ценами и лояльностью клиентов.

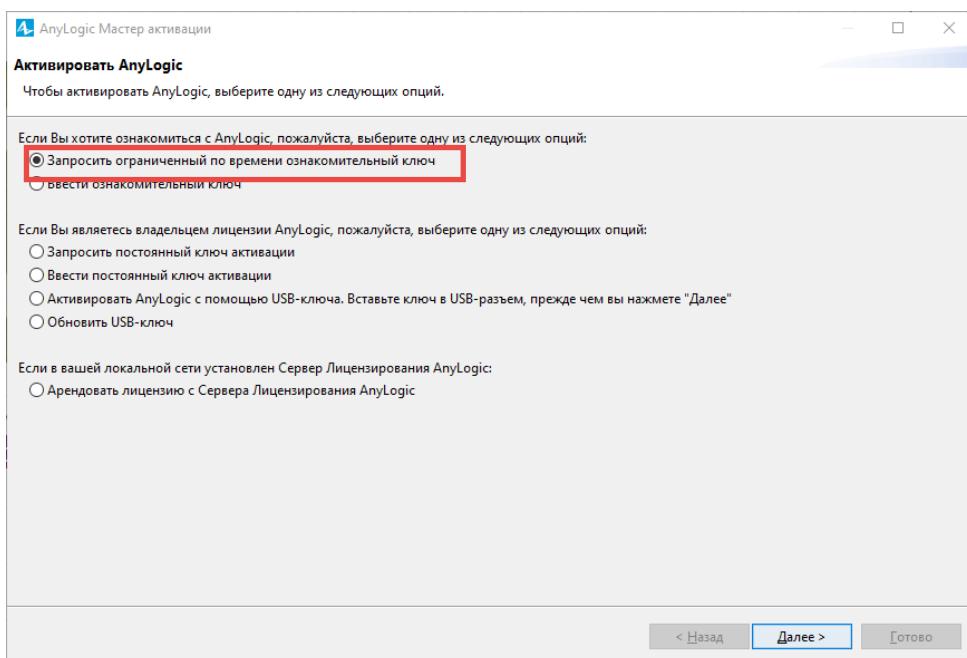


Иногда отдельные части системы проще моделировать с помощью методов, отличных от основного. В таких ситуациях лучше всего строить многоподходные модели.

## Установка и активация AnyLogic

Версия для самостоятельного обучения AnyLogic Personal Learning Edition (PLE) устанавливается без активации. Наиболее же функционально полная версия AnyLogic Professional требует пройти несложную операцию активации, которую мы и опишем ниже. Загрузите AnyLogic с сайта [www.anylogic.com](http://www.anylogic.com), затем установите AnyLogic и следуйте этой инструкции:

1. Запустите AnyLogic. Если программа еще не была активирована, появится окно **Мастера активации AnyLogic**.
2. На первой странице мастера выберите опцию **Запросить ограниченный по времени ознакомительный ключ** и затем щелкните по кнопке **Далее**.



3. На второй странице мастера, **Запрос лицензии AnyLogic**, укажите свою личную информацию и щелкните по кнопке **Далее**.

AnyLogic Мастер активации

**Запрос ознакомительного ключа для активации AnyLogic**

Введите данные о себе и укажите действительный адрес электронной почты. Ознакомительный ключ активации будет выслан на этот адрес.

\*Имя: Владимир

\*Фамилия: Татарский

\*Компания: Рекламное агентство "Тайный советчик"

\*Адрес рабочей эл. почты: vtatarsky@tainiy.ru

Укажите действительный адрес электронной почты. Мы используем его, чтобы отправить вам ознакомительный ключ активации.

Я хочу получать ежемесячную рассылку AnyLogic

\*Телефон: +7 (123) 456 78 90

\*Страна: Russian Federation

\*Сфера деятельности: Студент

\*В какой области вы используете AnyLogic?

Другое

\*Как вы узнали об AnyLogic?

Книга "AnyLogic за три дня"

Ваш запрос будет отправлен на сервер активации AnyLogic

Настройки прокси...

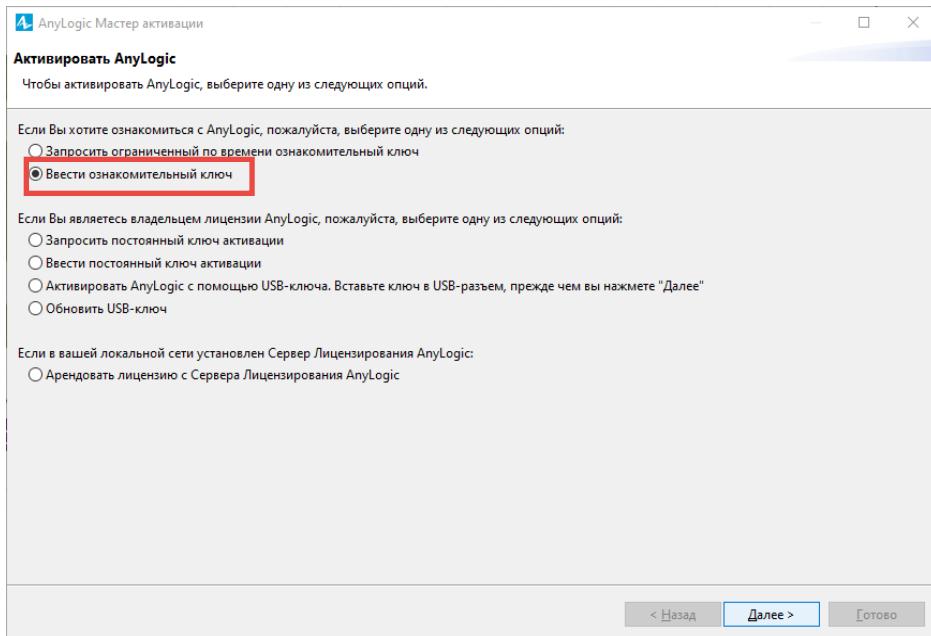
< Назад Далее > Готово

Вскоре после отправки запроса вы получите письмо с ключом и рекомендациями по активации. Также вы получите письмо с дополнительной информацией и полезными ссылками.

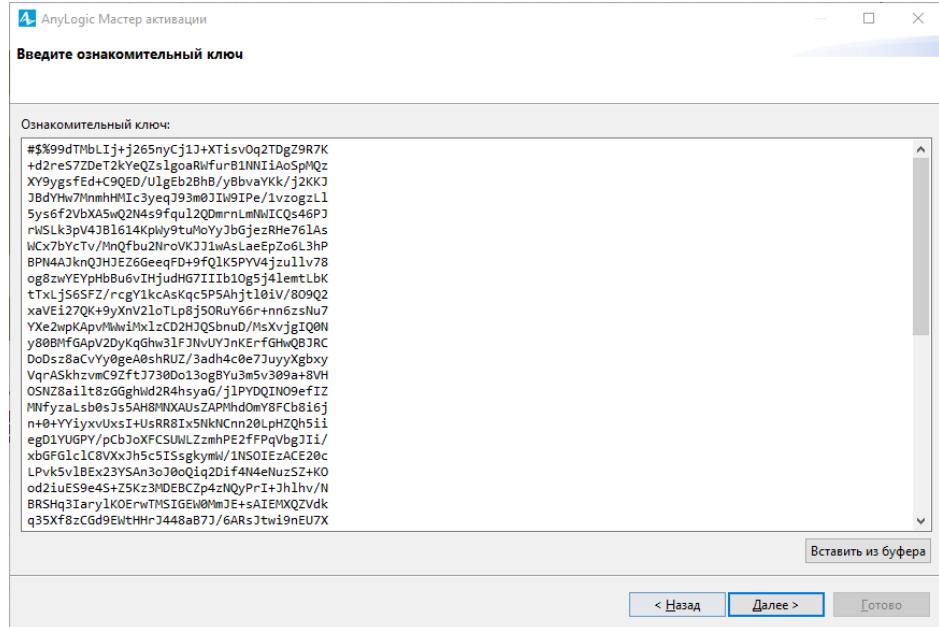
## 16 AnyLogic 8 за три дня

4. Если вы закрыли окно мастера активации, откройте его снова, запустив AnyLogic. Затем выберите опцию **Ввести ознакомительный ключ** на первой странице мастера и щелкните **Далее**.

Если вы не закрывали окно мастера активации после того, как запросили ключ, то просто щелкните по кнопке **Далее**.



5. Скопируйте полученный ключ активации из сообщения, вставьте ключ в поле **Ознакомительный ключ** с помощью кнопки **Вставить из буфера** и щелкните по кнопке **Далее**.



6. Мастер активации подтвердит, что продукт был успешно активирован. Щелкните по кнопке **Готово**.

Вы завершили процесс активации AnyLogic и теперь можете приступить к созданию первой модели.

## Агентное моделирование

*Агентное моделирование* - относительно новый метод моделирования. Поначалу оно являлось преимущественно предметом теоретических дискуссий в академических кругах, а начиная с 2000-х годов разработчики имитационных моделей стали использовать его на практике.

Переход к агентному моделированию был вызван:

- Желанием глубже изучить системы, которые сложно описать традиционными методами моделирования.
- Развитием технологии агентного моделирования (объектно-ориентированное моделирование, диаграммы состояний).
- Быстрому росту мощности процессоров и объема оперативной памяти компьютеров. Агентные модели более требовательны к ресурсам, чем модели системной динамики или дискретно-событийные модели.

Агентное моделирование предлагает разработчику моделей альтернативный взгляд на поведение системы.

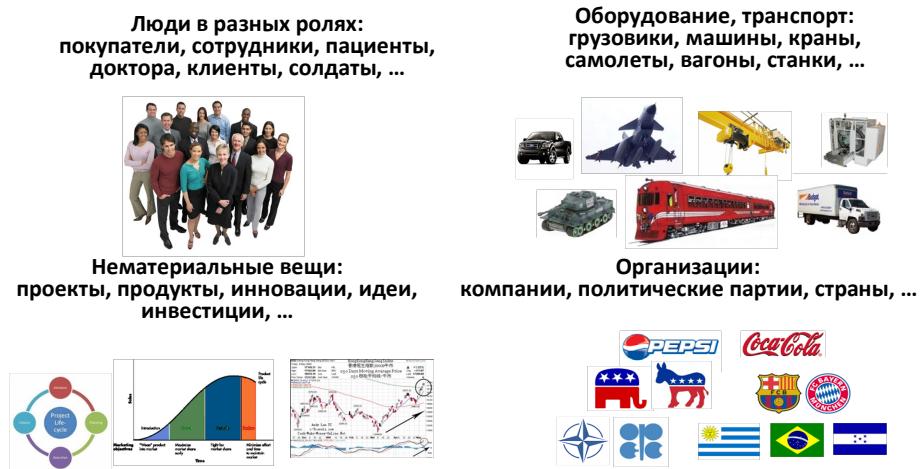
◆ **Вы можете не знать ни поведения системы в целом, ни ее главных переменных и зависимостей между ними, или не видеть четкой схемы процессов, но при этом понимать, как ведут себя отдельные элементы системы. В таком случае вы можете начать создание модели с идентификации моделируемых объектов (агентов) и задания их поведения. Иногда вам может понадобиться объединить агентов в сеть и позволить им взаимодействовать друг с другом, либо же поместить агентов в среду, которая имеет свою собственную динамику. Таким образом, глобальное поведение системы формируется из многих десятков (тысяч, миллионов) параллельно протекающих процессов.**

На данный момент не существует стандартного языка агентного моделирования. Структура агентной модели может быть задана как графически, так и с помощью сценариев. Поведение агента может быть задано различными способами. Если у агента есть состояние, от которого зависят его действия и реакции, то его поведение лучше всего задавать с помощью диаграммы состояний. Иногда

поведение агента задается действиями, выполняемыми при наступлении определенных событий.

Иногда внутренняя динамика агента лучше всего задается с помощью дискретных событий или системной динамики. Так же и динамика среды, в которой живут агенты, может моделироваться с помощью традиционных методологий. По этой причине многие агентные модели совмещают в себе несколько подходов к моделированию.

Агентами могут быть самые разные объекты: транспортные средства, оборудование, проекты, организации, земельные участки, люди и так далее.



Товарищи ученые (доценты с кандидатами) до сих пор спорят, какими именно свойствами должен обладать объект, чтобы называться агентом: способностью действовать и реагировать на действия других, ориентироваться в пространстве, обучаться, взаимодействовать и общаться, обладать «интеллектом» и т.п. На практике в агентных моделях вам могут встретиться агенты любых типов: одни общаются друг с другом, а другие находятся в полной изоляции; одни живут в пространстве, а другие – нет; одни обучаются и приспосабливаются, а другие никогда не меняют своего поведения.

Приведем несколько полезных фактов об агентах, чтобы многообразие теорий не вводило вас в заблуждение:

- **Агенты не являются клеточными автоматами** и не обязательно обитают в дискретном пространстве (как в игре «Жизнь»). Во многих агентных моделях пространство вообще отсутствует. Когда пространство

все же необходимо, оно чаще всего является непрерывным (это может быть карта мира или план здания).

- **Агенты – не обязательно люди.** Агентом может быть все, что угодно: транспортное средство, оборудование, проект, организация или даже идея.
- **Агентом может быть объект, кажущийся абсолютно пассивным.** Например, в модели нефтепровода вы можете представить сегмент трубы как агента, задав для него графики техобслуживания, вероятности происхождения аварий, логику проведения ремонтных работ, затраты и т.д.
- **Агентов в модели может быть как много, так и мало.** При этом агенты могут быть как одного типа, так и разных.
- **Существуют агентные модели, в которых агенты вообще не взаимодействуют друг с другом.** Например, в моделях потребления алкоголя, развития ожирения или хронических заболеваний индивидуальная динамика агента зависит только от его личных параметров и, в некоторых случаях, от среды.

## Модель потребительского рынка

Давайте построим агентную модель, которая поможет нам изучить процесс вывода нового продукта на рынок.

- Мы рассмотрим относительно небольшой потребительский рынок численностью в 5000 человек. С точки зрения реализации модели каждый потребитель будет являться агентом.
- Поскольку мы рассматриваем процесс вывода на рынок нового продукта, то изначально никто этим продуктом не пользуется.
- Люди начнут покупать продукт под влиянием рекламы.
- После этого начального этапа куда более сильное влияние на продажи будет оказывать общение людей друг с другом, рекомендации и положительные отзывы потребителей продукта, побуждающие других на его приобретение.

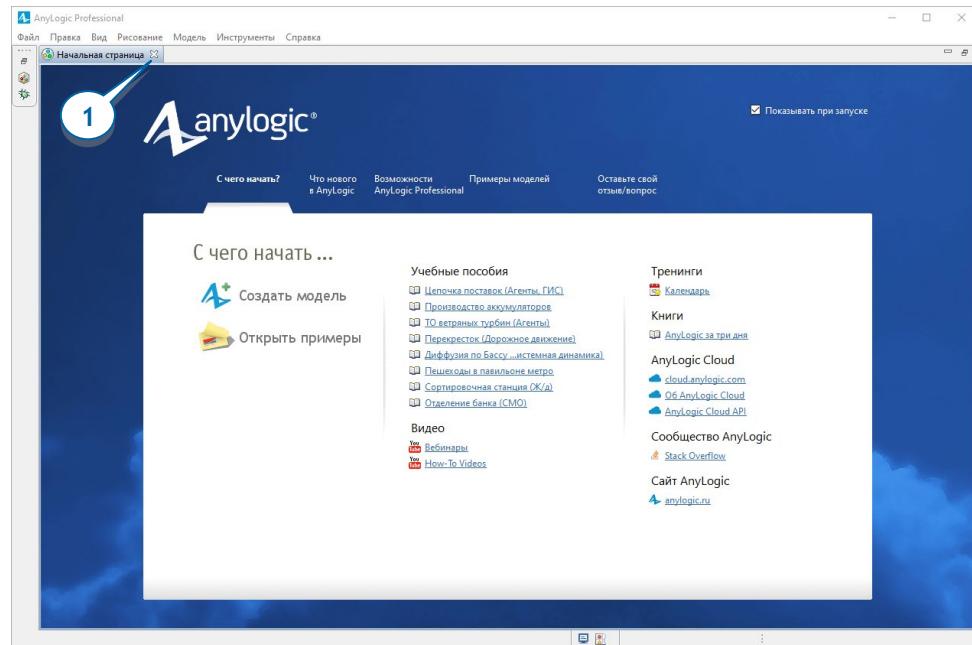
### Фаза 1. Создание популяции агентов

Начнем с создания простой модели, которая продемонстрирует влияние рекламы на начальные продажи продукта.

Люди в нашей модели поначалу не будут пользоваться продуктом, но потенциально могут быть в нем заинтересованы. Для начала мы создадим популяцию агентов, а потом зададим то, как люди приобретают товар под влиянием рекламы.

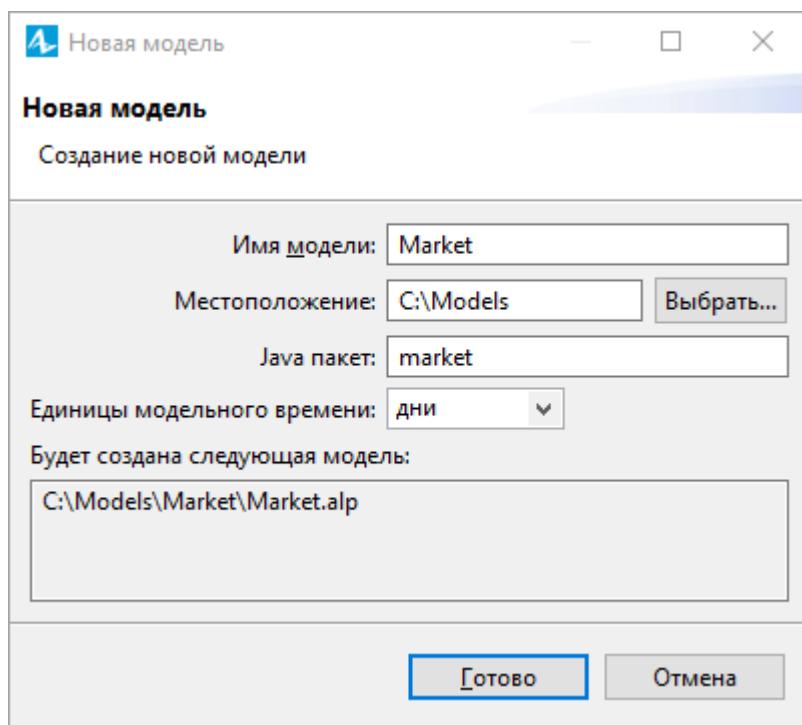
Запустите AnyLogic. Откроется *Начальная страница*.

*Начальная страница* предлагает обзор программы AnyLogic и ее функционала, а также позволяет открывать различные примеры моделей.



### Начальная страница

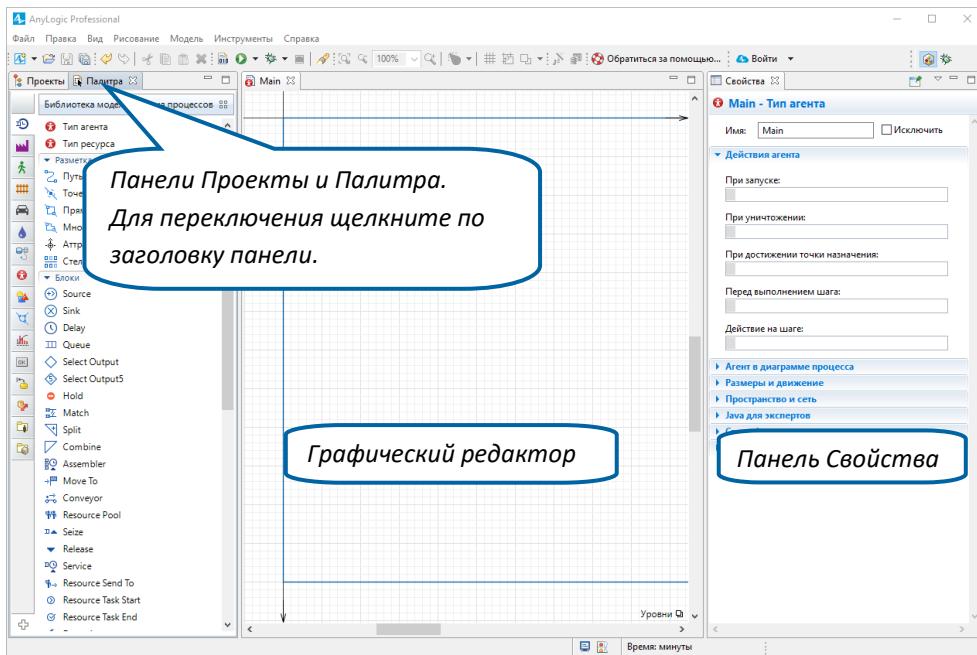
1. Закройте Начальную страницу и создайте новую модель. Для этого выберите **Файл > Создать > Модель** из главного меню AnyLogic. Откроется диалоговое окно **Новая модель**.



2. В поле **Имя модели** введите имя новой модели: *Market*.
3. В поле **Местоположение** выберите каталог, в котором вы хотите сохранить файлы модели.
4. Щелкните по кнопке **Готово**.

Будет создана новая модель.

Самое время бегло изучить пользовательский интерфейс AnyLogic.



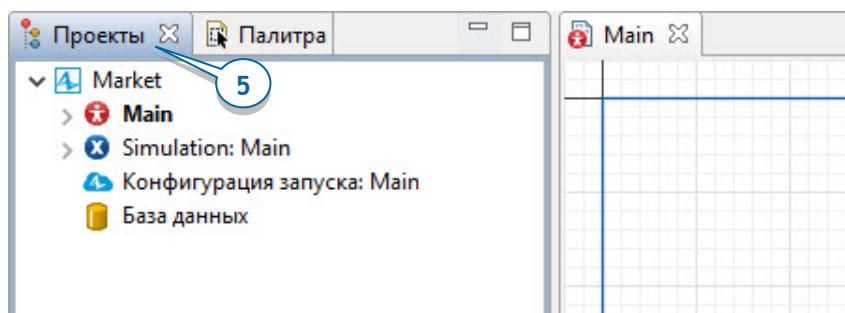
## Рабочее пространство AnyLogic

- Графический редактор позволяет редактировать диаграмму агента. Вы можете добавлять элементы на диаграмму, перетаскивая их из Палитры на холст редактора. Синяя прямоугольная рамка ограничивает ту область холста, которая будет отображаться в окне модели при ее запуске.
- Панель Проекты отображает содержимое моделей AnyLogic, открытых в рабочем пространстве в текущий момент. Элементы каждой модели отображаются в виде иерархического дерева, для облегчения навигации.
- Панель Палитра содержит все графические элементы AnyLogic, сгруппированные в отдельные палитры. Чтобы добавить тот или иной элемент в модель, перетащите соответствующий элемент из палитры в графический редактор.
- Панель Свойства позволяет вам просматривать и изменять свойства выделенных в текущий момент элементов модели.

- Чтобы открыть или закрыть панель, выберите в меню **Вид** соответствующий пункт с именем панели.
- Чтобы изменить размер панели, перетащите мышью ее границу.
- Вы всегда можете воспользоваться опцией **Восстановить расположение панелей** в меню **Инструменты**, чтобы вернуть расположение панелей по умолчанию.

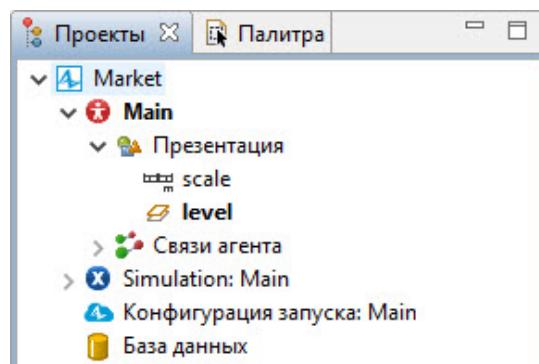
5. Давайте откроем панель **Проекты**, чтобы изучить структуру новой модели.

Панели **Палитра** и **Проекты** находятся в левой части рабочего пространства, и вы можете переключаться с панели **Палитра** на панель **Проекты**, щелкнув по заголовку панели.



### Навигация по элементам модели в панели Проекты

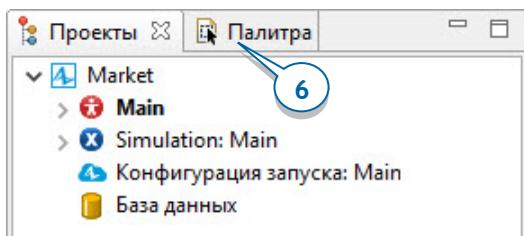
- Панель **Проекты** предоставляет простой и быстрый доступ к содержимому моделей, открытых в рабочем пространстве AnyLogic. AnyLogic отображает структуру каждой модели в виде иерархического дерева.



- По умолчанию в каждой модели создается один тип агента – **Main** – и один эксперимент **Simulation**, хранящий настройки запуска этой модели. **Конфигурация запуска** позволяет настраивать входные и выходные параметры модели перед ее загрузкой в облако AnyLogic Cloud.
- Двойной щелчок по типу агента или эксперименту открывает его диаграмму в графическом редакторе.
- Также у каждой модели есть своя встроенная **База данных**. База данных изначально пуста, но при необходимости вы можете импортировать в нее данные из внешней базы данных (например, MS Excel), а также собрать информацию о выполнении модели в специальные журналы (для этого нужно выбрать в свойствах базы данных опцию **Записывать в журнал информацию о выполнении модели**).
- Щелчок по элементу модели в дереве выделяет этот элемент и показывает его в центре графического редактора. Если вы не можете найти какой-либо элемент на графической диаграмме, воспользуйтесь таким способом нахождения элемента.

Чтобы добавить в нашу модель потребителей, нам нужно создать новый тип агента (потребитель) и затем создать популяцию агентов, которая будет состоять из заданного количества агентов этого типа. Вы можете выполнить оба этих действия с помощью удобного мастера создания агентов.

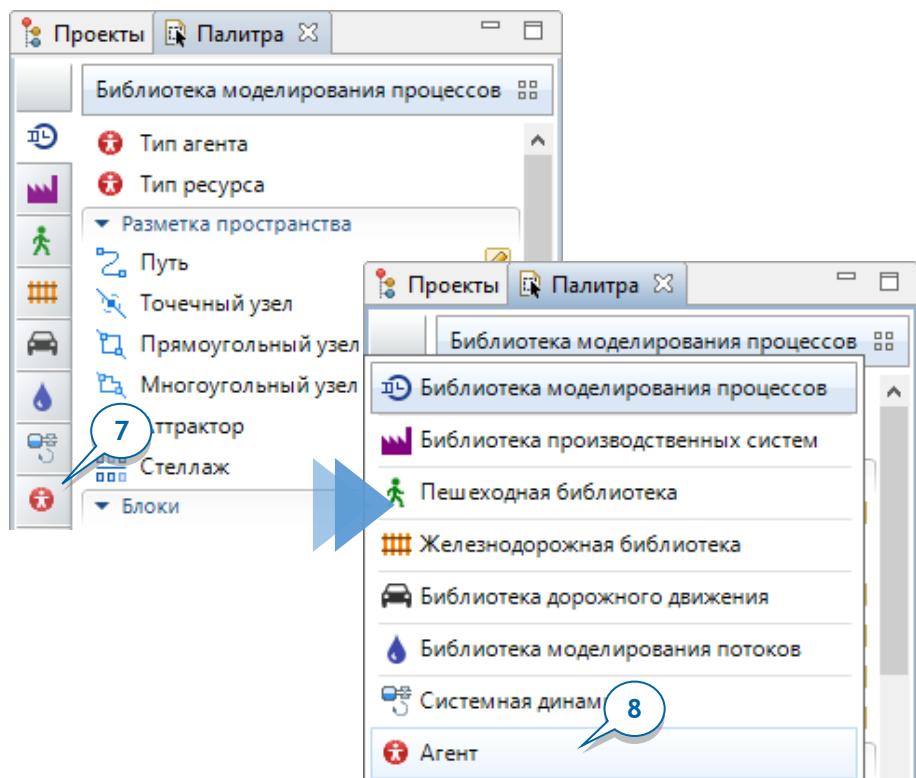
- Мы собираемся добавить новый элемент в нашу модель, поэтому давайте перейдем в панель **Палитра** , щелкнув по заголовку этой панели.



- Откройте палитру **Агент**. Чтобы открыть другую палитру, перейдите в панель **Палитра** и наведите курсор мыши на вертикальную панель навигации.

8. Откроется список всех палитр, и вы сможете выбрать нужную вам палитру.

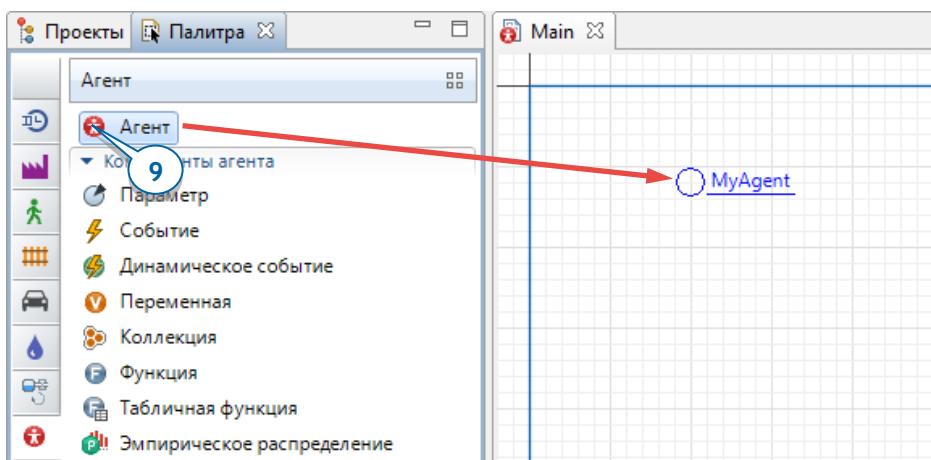
Щелкните в списке по палитре **Агент** .



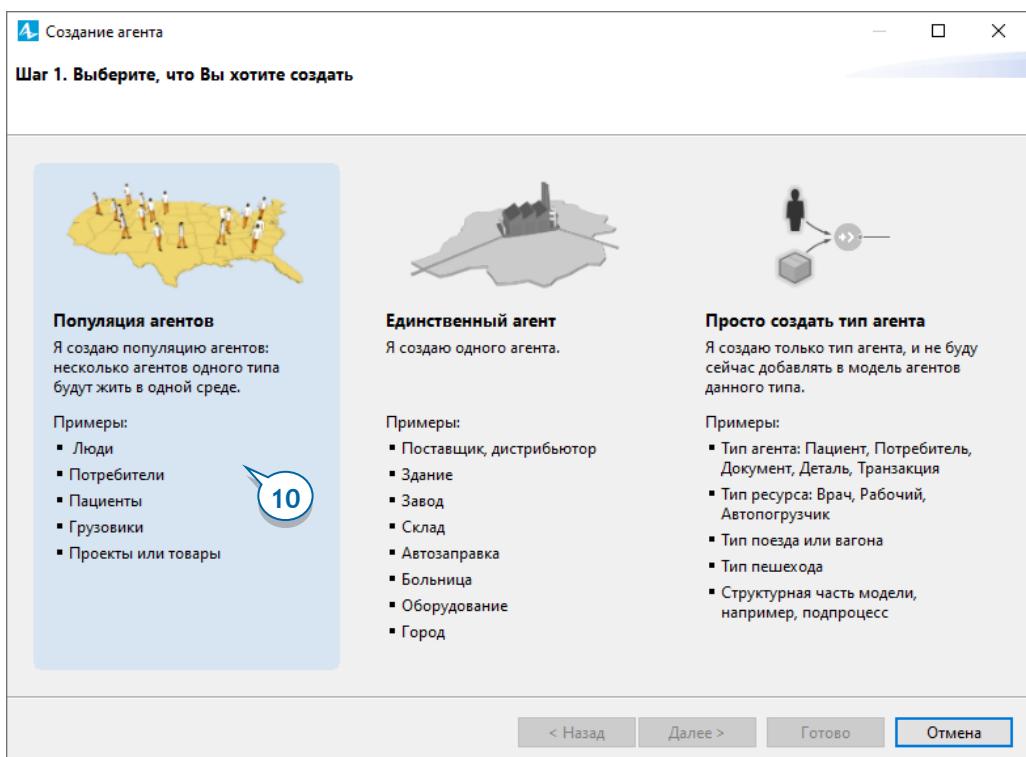
Запомнив значки палитр, вы сможете открывать палитры простым щелчком мыши по нужному значку.

В графическом редакторе нашей модели сейчас отображается пустая диаграмма агента *Main*.

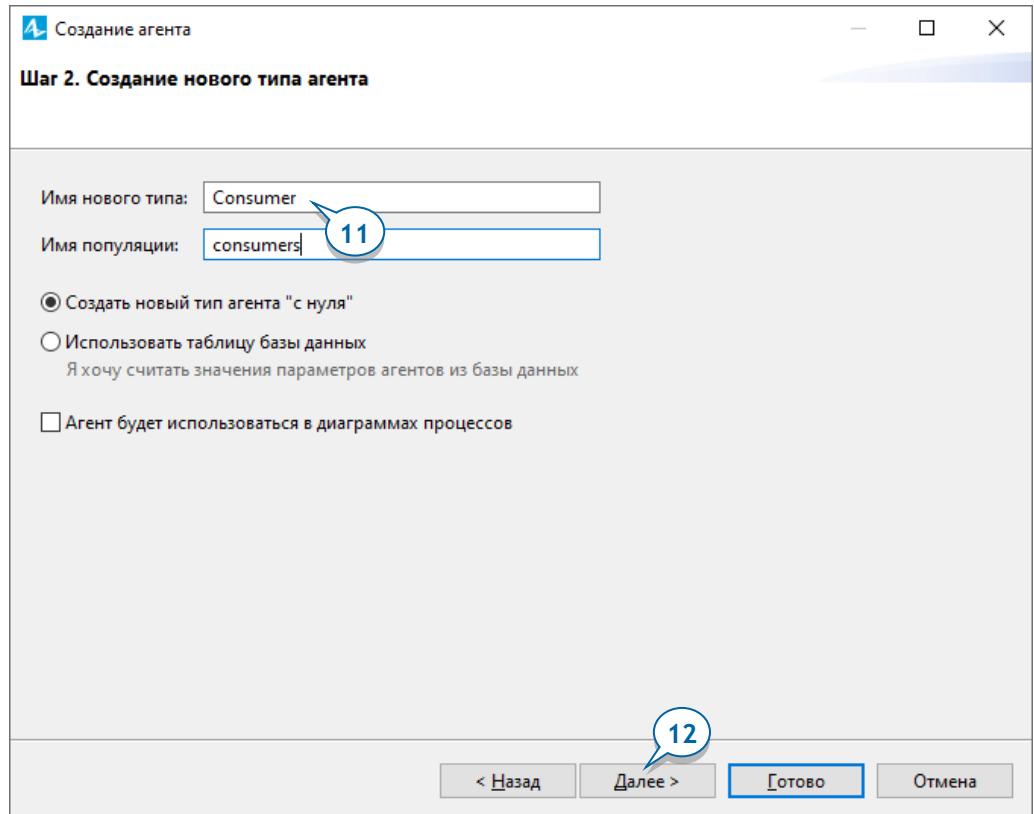
9. Перетащите элемент **Агент**  из палитры **Агент** на диаграмму *Main*.



10. Откроется мастер создания агентов **Новый агент**. Мы хотим создать большое количество агентов одного типа, поэтому на первой странице мастера выберите опцию **Популяция агентов**.

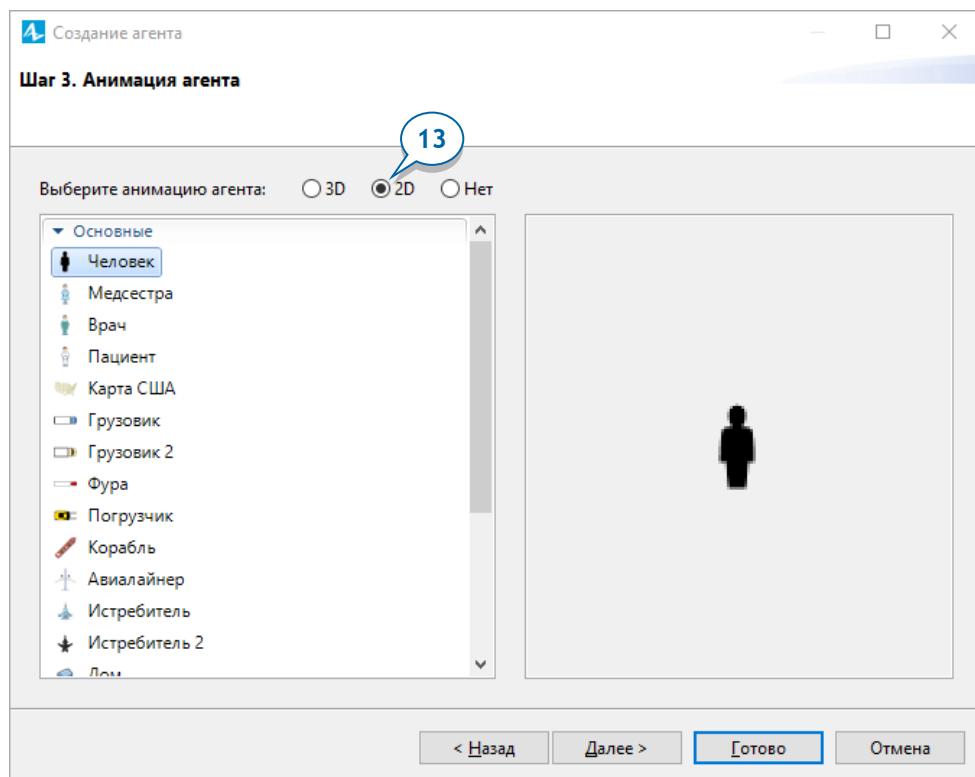


11. На странице мастера **Шаг 2. Создание нового типа агента**, в поле **Имя нового типа**, введите *Consumer*, то есть потребитель. Содержание поля **Имя популяции** автоматически изменится на подходящее *consumers*.



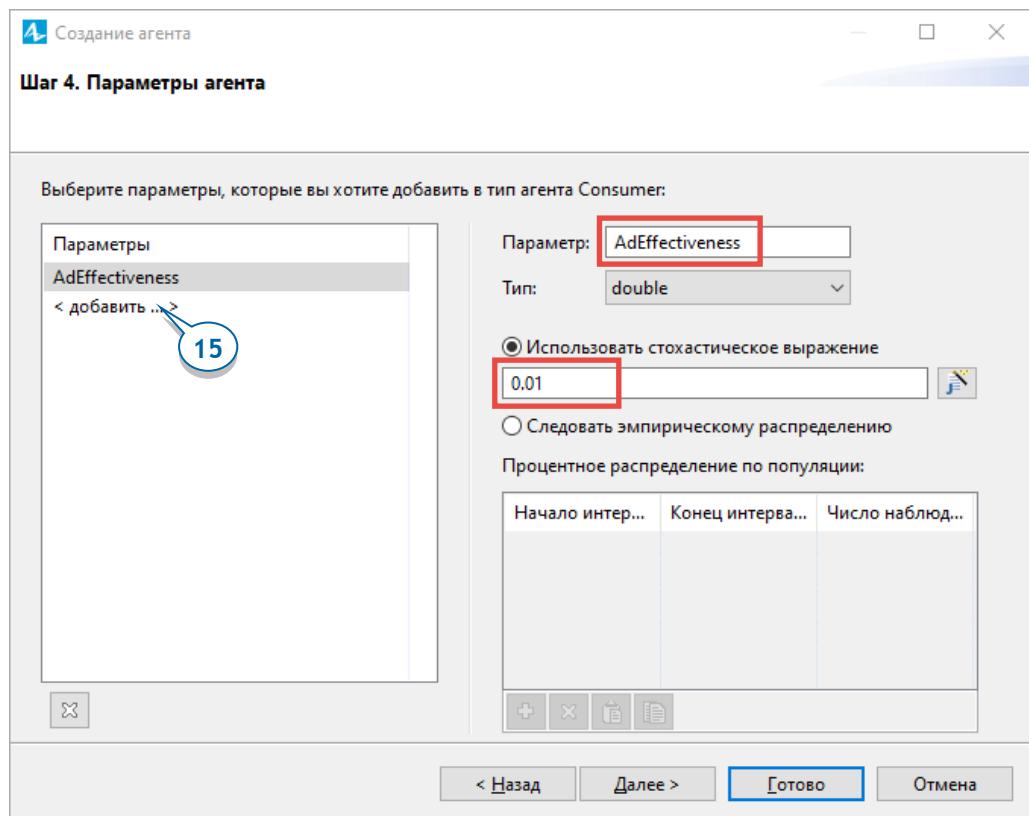
12. Щелкните по кнопке **Далее**.

13. На странице мастера **Шаг 3. Анимация агента** выберите фигуру анимации агента. Поскольку мы создаем простую модель с двумерной анимацией, выберите опцию **2D** и затем выберите первую фигуру (**Человек**) из расположенного ниже списка. Щелкните по кнопке **Далее**.



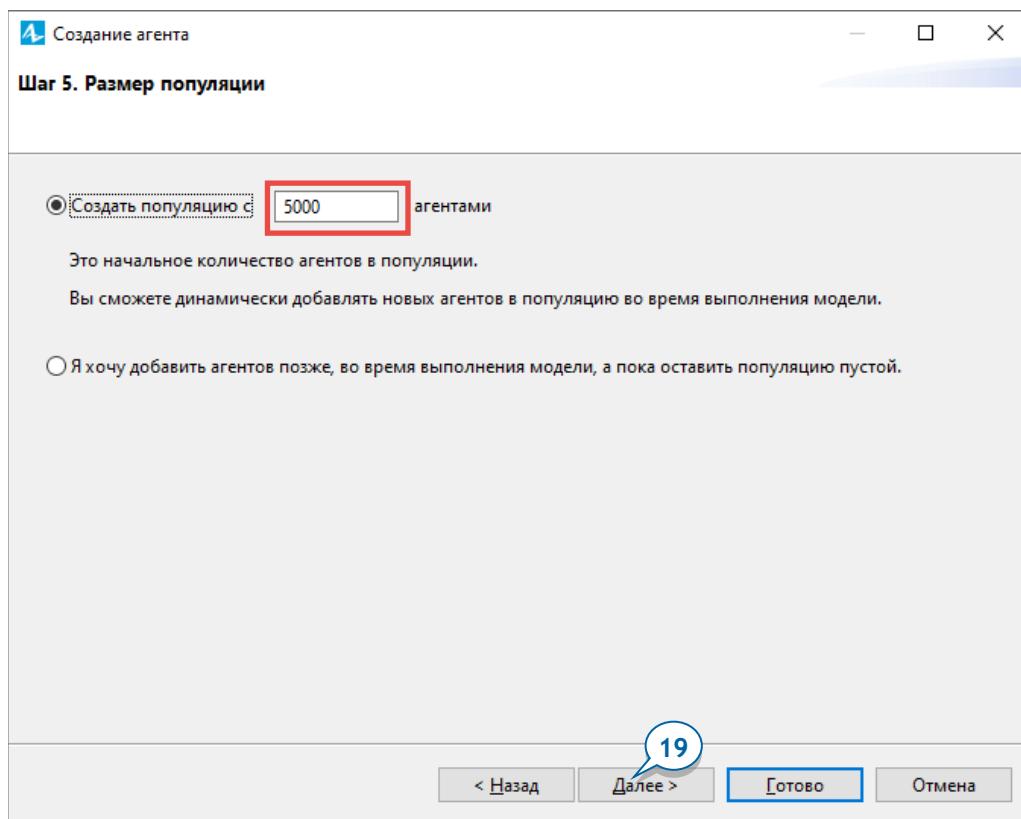
14. На следующей странице мастера можно задать параметры агента (обычно представляющие собой его статические характеристики).

Мы добавим параметр *AdEffectiveness* (эффективность рекламы), чтобы задать процентную долю потенциальных потребителей, которые захотят купить продукт в течение дня вследствие воздействия рекламы.



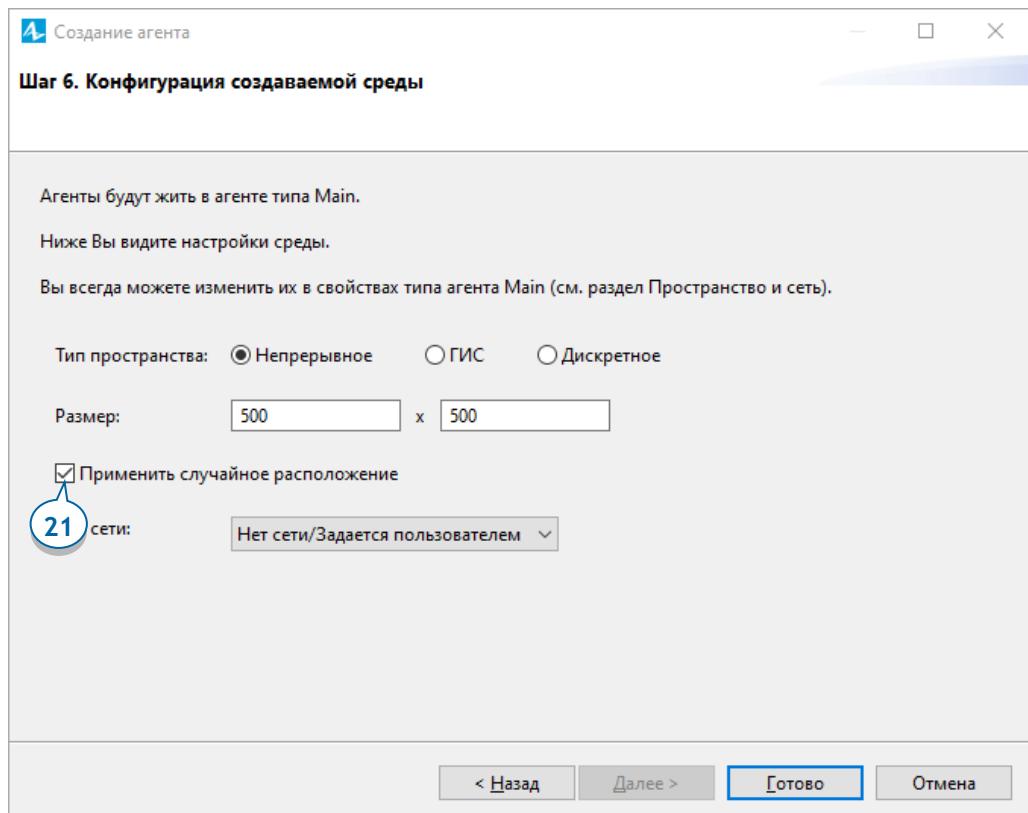
15. В секции слева вы увидите таблицу **Параметры**. Щелкните по строке **<добавить...>**, чтобы создать новый параметр.
16. Справа, в поле **Параметр**, измените заданное по умолчанию имя параметра на *AdEffectiveness*. Выберите в поле **Тип** опцию **double** (параметр будет принимать вещественные значения). Этот параметр задает эффективность рекламы. Мы предполагаем, что за день к решению о приобретении продукта приходит в среднем 1% потенциальных потребителей, поэтому мы задаем *0.01* в качестве значения данного параметра.
17. Щелкните по кнопке **Далее**.

18. На следующей странице мастера, **Размер популяции**, в поле **Создать популяцию с ... агентами** введите значение 5000, чтобы создать 5000 агентов типа *Consumer*. Каждый агент, живущий в создаваемой нами популяции, будет моделировать отдельного агента-потребителя.

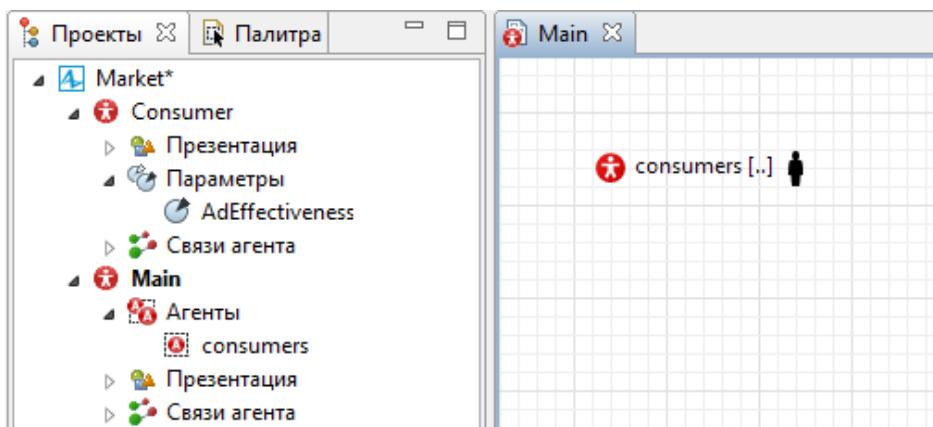


19. Щелкните по кнопке **Далее**.

- 20.** На странице мастера **Конфигурация создаваемой среды** оставьте выбранный по умолчанию тип пространства среды (**Непрерывное**) и значения его размерностей **Ширина** и **Высота** (500). Тогда при запуске модели AnyLogic отобразит агентов внутри прямоугольного пространства размером 500x500 пикселей.
- 21.** Выберите опцию **Применить случайное расположение**, чтобы расположить агентов в заданном нами выше пространстве случайным образом.



- 22.** Щелкните по кнопке **Готово**.
- 23.** Давайте откроем панель **Проекты**  и посмотрим, какие именно новые элементы были созданы мастером. Разверните ветви дерева нашей модели, чтобы посмотреть на их содержимое.



В нашей модели теперь два типа агентов: *Main* и *Consumer*.

- Тип агента *Consumer* содержит фигуру анимации агента (*person* в ветке *Презентация*) и параметр *AdEffectiveness*.
- Тип агента *Main* содержит популяцию агентов, которая называется *consumers* (набор из 5000 агентов типа *Consumer*).

### Среда обитания агентов

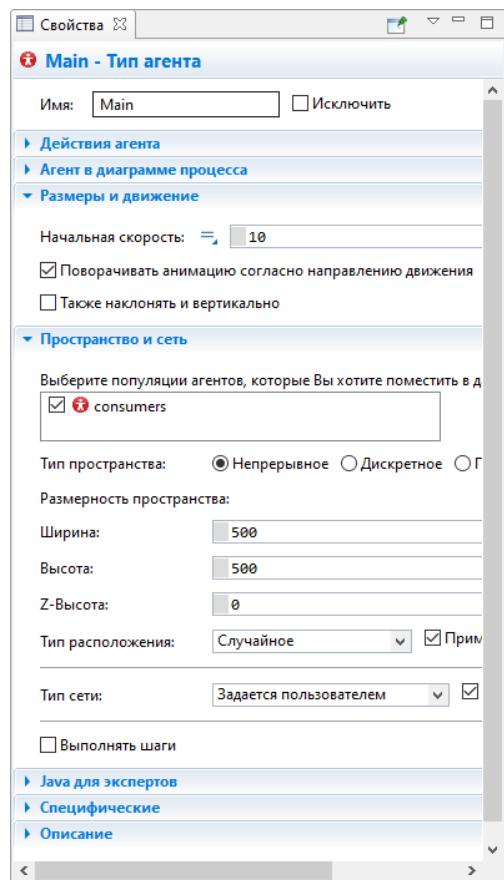
Типичная архитектура агентной модели в AnyLogic - агент *Main*, на диаграмме которого заданы популяции агентов других типов. В этом случае агент *Main* играет роль *среды обитания* для других агентов. Среда задает тип пространства, в котором живут агенты, расположение агентов в пространстве и сеть контактов агентов, которая может использоваться при их общении друг с другом.

- 24.** В панели **Проекты**, щелкните по элементу *Main*. Тем самым, вы откроете свойства этого элемента в панели **Свойства** (эта панель находится в правой части окна AnyLogic).

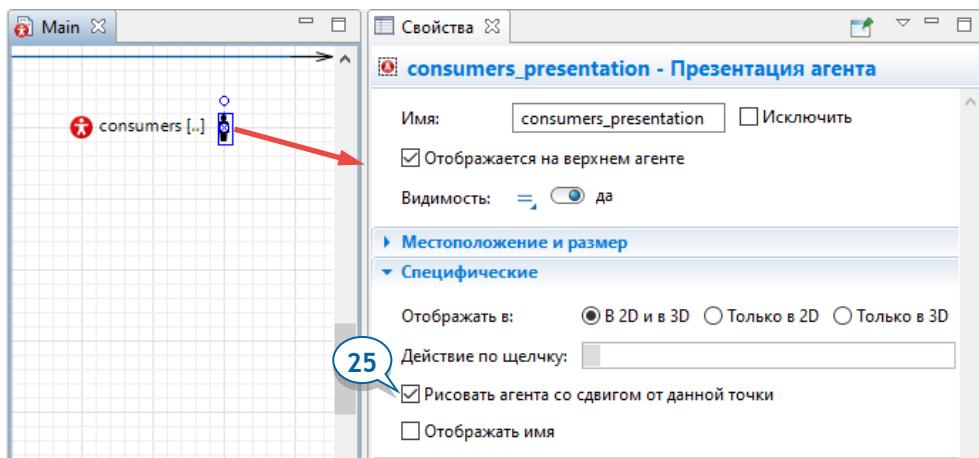
Вы можете изменить настройки среды обитания популяции агентов *consumers* в секции свойств агента *Main* **Пространство и сеть**.

## Панель Свойства

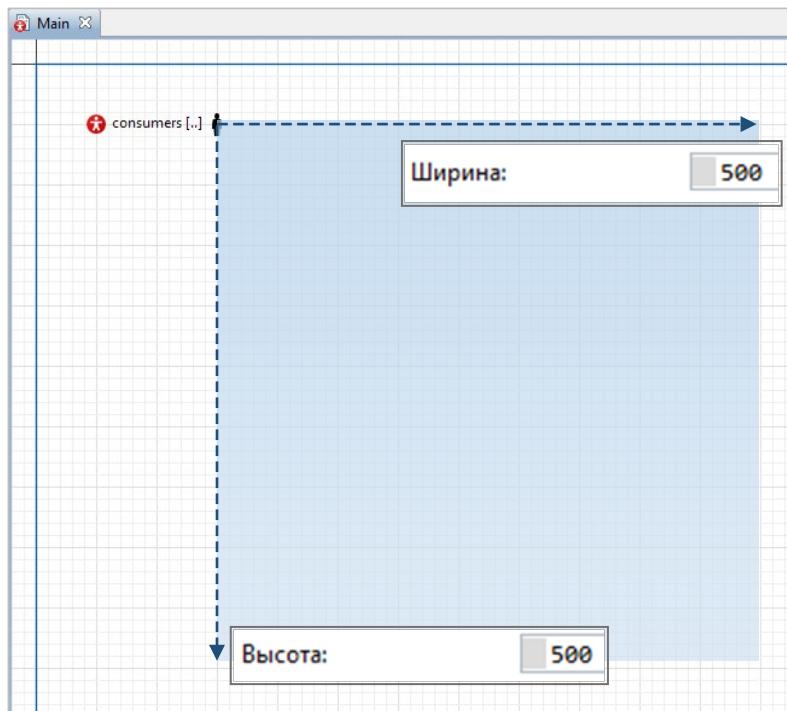
- Панель **Свойства** является контекстно-зависимой и отображает свойства выделенного в данный момент элемента.
- Чтобы изменить какое-либо свойство элемента, сначала выделите элемент щелчком в графическом редакторе или в панели **Проекты** и затем перейдите в панель **Свойства**.
- Панель **Свойства** может содержать секции. Чтобы раскрыть или свернуть секцию, щелкните по ее заголовку.
- Имя и тип выделенного элемента отображаются в шапке панели.



- 25.** Выделите фигуру анимации популяции агентов , расположенную на диаграмме *Main*, откройте секцию ее свойств **Специфические** и выберите опцию **Рисовать агента со сдвигом от данной точки**.

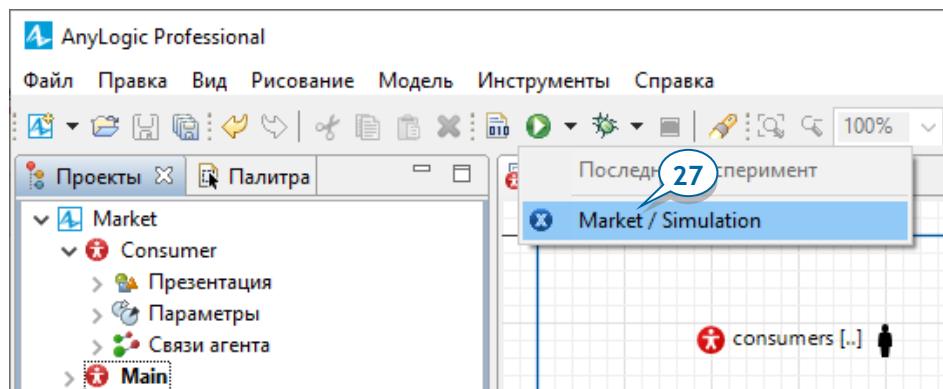


Следующий рисунок показывает, как фигура анимации популяции задает верхний левый угол прямоугольника (в нашем случае - размером 500x500 пикселей), внутри которого будут располагаться фигуры анимации агентов данной популяции во время выполнения модели.



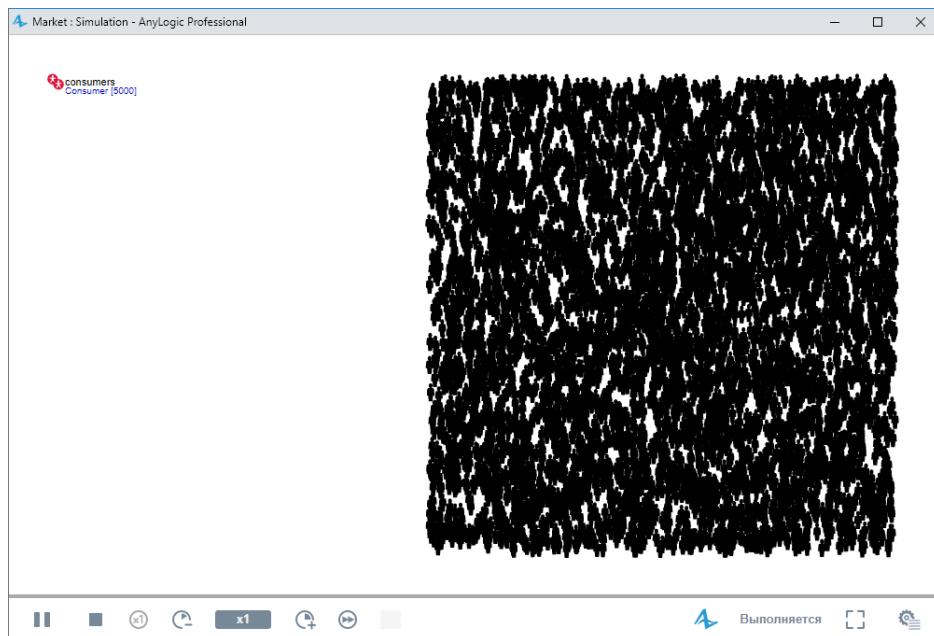
Мы закончили создание простейшей агентной модели, и теперь мы можем запустить ее и понаблюдать за поведением агентов.

26. Щелкните по кнопке панели инструментов **Построить модель** , чтобы скомпилировать нашу модель и проверить ее на наличие ошибок компиляции.
27. Щелкните по маленькому треугольнику справа от кнопки панели инструментов **Запустить**  . Выберите из раскрывшегося списка тот эксперимент, который вы хотите запустить. В нашем случае это **Market / Simulation**.



Так как в рабочем пространстве может быть открыто сразу несколько моделей, и у каждой модели может быть несколько экспериментов, а в этом списке находятся все эксперименты всех открытых моделей, то вам следует выбирать из списка именно нужный вам эксперимент.

После того, как вы запустите модель, появится окно модели. Вы увидите презентацию модели - 5000 фигур анимации агентов популяции *consumers*. Так как мы пока не задавали правила поведения агентов, на анимации больше ничего не происходит.



Управлять выполнением модели можно с помощью панели управления, расположенной в нижней части окна запущенной модели.

## Управление выполнением модели



### Запустить

[Кнопка видна, если в данный момент модель не выполняется] Начинает выполнение. Если же моделирование было приостановлено, то продолжает его с текущего состояния.



### Пауза

[Кнопка видна, если в данный момент модель выполняется] Приостанавливает выполнение модели. Вы можете продолжить ее выполнение в любой момент времени.



### Прекратить выполнение эксперимента

Прекращает выполнение модели.

Чтобы убедиться, что модель запущена, проверьте статус текущего прогона модели (**Выполняется**, **Пауза**, **Готов** или **Завершен**), отображаемый в панели разработчика окна модели.

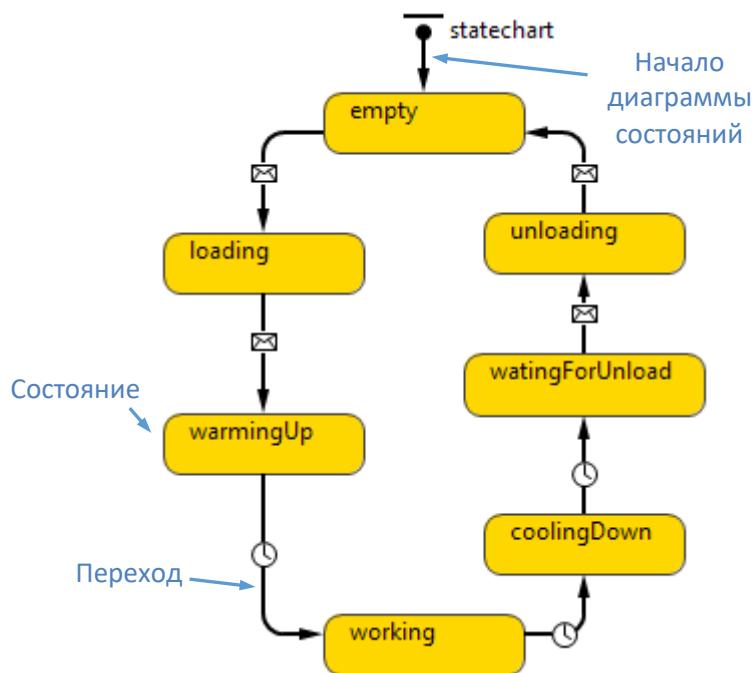
**28.** Мы готовы продолжить разработку модели. Закройте окно модели, щелкнув по кнопке, расположенной в правом верхнем углу окна.

## Фаза 2. Задание поведения потребителей

Теперь давайте зададим поведение потребителей. Лучше всего задавать поведение агента с помощью диаграммы состояний.

### Диаграммы состояний

- Диаграммы состояний (карты состояний, стейтчарты) являются самым удобным средством задания поведения агента. Диаграммы состояний содержат **состояния** и **переходы**. Состояния диаграммы являются взаимоисключающими, то есть в каждый момент времени агент может находиться только в одном состоянии. Срабатывание перехода приводит к смене состояния и активации новых переходов. Допускается создание иерархических состояний, которые содержат внутри себя другие состояния и переходы.
- У одного агента может быть сразу несколько диаграмм состояний, каждая из которых описывает независимые аспекты поведения агента.



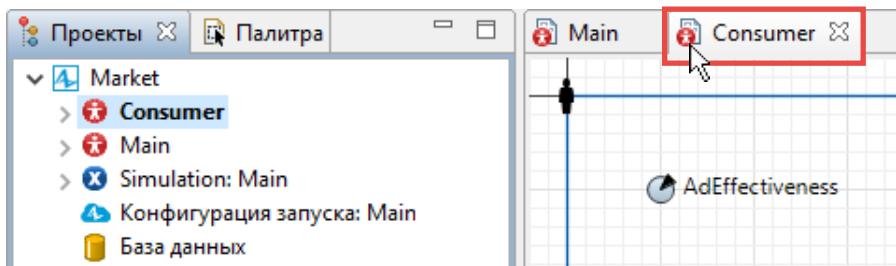
Мы зададим поведение агента-потребителя как два последовательных состояния:

- *PotentialUser* - находящийся в данном состоянии агент является потенциальным покупателем и может быть заинтересован в покупке.
  - *User* - потребитель, находящийся в этом состоянии, уже купил продукт.
1. Откройте диаграмму агента-потребителя *Consumer*, дважды щелкнув по этому типу агента в панели **Проекты**. Вы увидите графическую диаграмму этого агента с фигурой анимации, расположенной на пересечении осей координат, и одним параметром.

### Как узнать, какой тип агента вы редактируете?

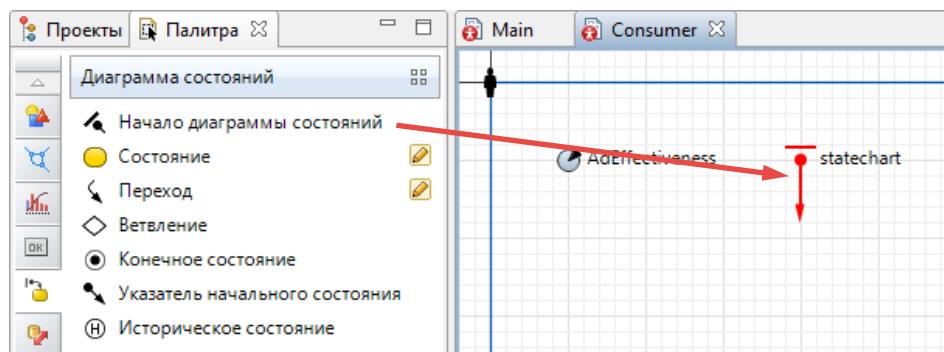
Так как в нашей модели теперь два типа агента, то и наши инструкции будут относиться теперь то к диаграмме одного, то к диаграмме другого агента. Поэтому давайте объясним, как понять, диаграмма какого именно агента открыта в вашем графическом редакторе в текущий момент,

- AnyLogic выделяет активную вкладку графического редактора синим цветом, а также выделяет в дереве модели в панели **Проекты** тот тип агента, диаграмма которого активна в текущий момент.
- Вы можете переключаться между диаграммами разных типов агентов щелчком по заголовку их вкладки в графическом редакторе (например, по заголовку вкладки *Consumer*, как на рисунке ниже):



2. Начнем рисовать диаграмму состояний потребителя с добавления двух состояний. Откройте палитру **Диаграмма состояний** .
3. Перетащите элемент **Начало диаграммы состояний**  из палитры **Диаграмма состояний** на диаграмму *Consumer*. Рисование диаграммы состояний всегда начинается с добавления *начала диаграммы состояний*.

Этот элемент определяет точку, инициирующую управление диаграммой, а также задает имя этой диаграммы состояний.

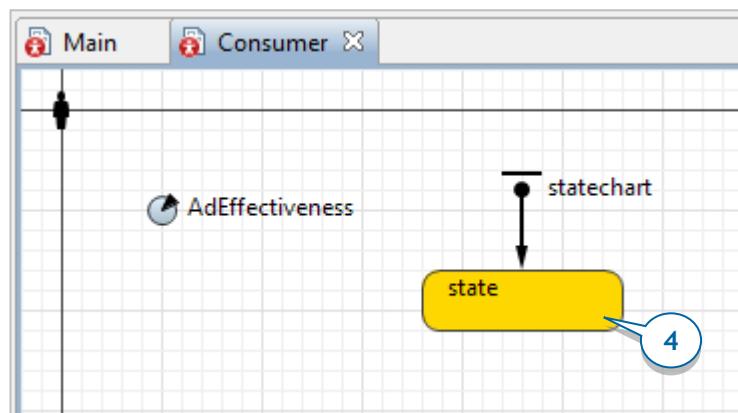


- ◆ Пожалуйста, будьте внимательны – Начало диаграммы состояний легко спутать с Указателем начального состояния ⚡ или Переходом ↗, поскольку их значки похожи.

Вы увидите, что AnyLogic отображает начало диаграммы состояний красным цветом. Это индикация того, что данный элемент не соединен ни с одним состоянием, и поэтому текущая диаграмма состояний считается незаконченной и неверно заданной.

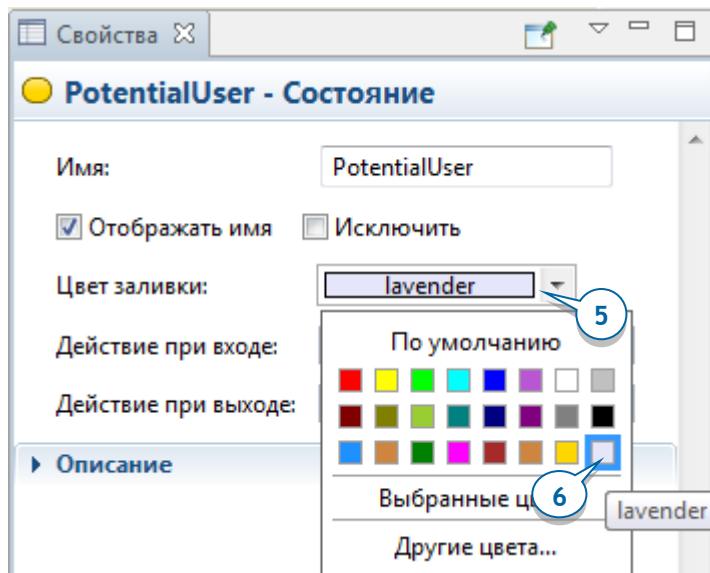
Давайте добавим первое состояние в диаграмму состояний потребителя.

4. Перетащите элемент **Состояние** 💫 из палитры **Диаграмма состояний** в графический редактор и соедините его с началом диаграммы.

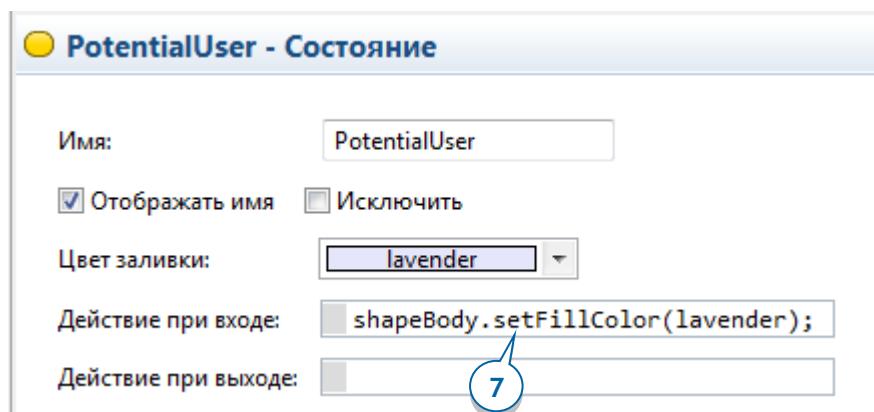


- Убедитесь, что вы рисуете диаграмму состояний на диаграмме агента **Consumer**, а не на диаграмме **Main**.

- Выделите состояние в графическом редакторе и измените его свойства. Назовите состояние *PotentialUser*.
- Щелкните в поле элемента управления **Цвет заливки** и измените цвет заливки состояния на *lavender*.

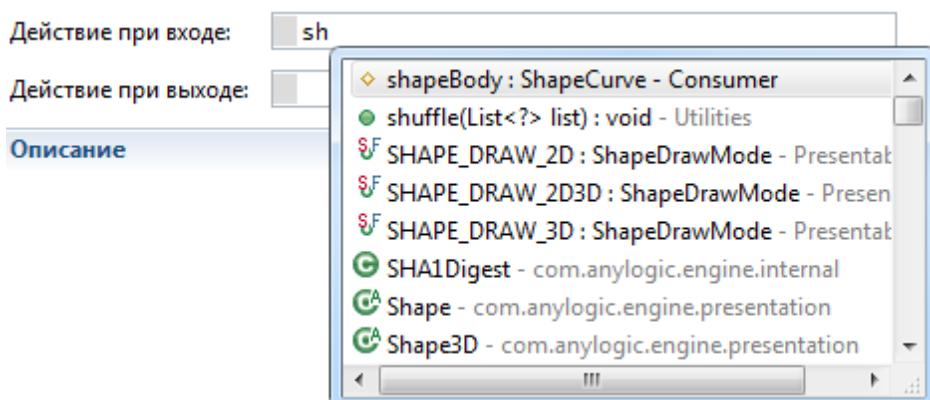


- Ведите следующий Java код в поле состояния **Действие при входе**:  
`shapeBody.setFillColor(lavender);`

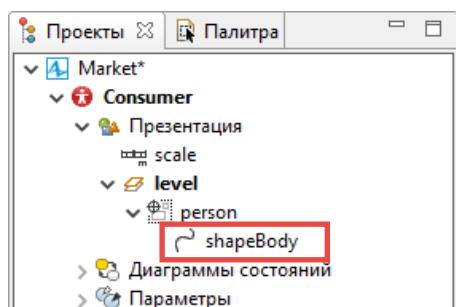


## Мастер подстановки кода

- Чтобы не печатать полностью имена элементов и функций, используйте помощник подстановки кода. Чтобы открыть помощник, щелкните в поле редактирования и нажмите на клавиатуре Ctrl+пробел (Alt+пробел на Mac OS). Во всплывающем окне будут перечислены все элементы модели, доступные в текущем контексте, такие как переменные, параметры и функции.
- Прокрутите до имени нужного элемента или введите первые буквы его имени, пока он не будет выделен в списке, затем нажмите Enter, чтобы вставить имя элемента в поле редактирования.



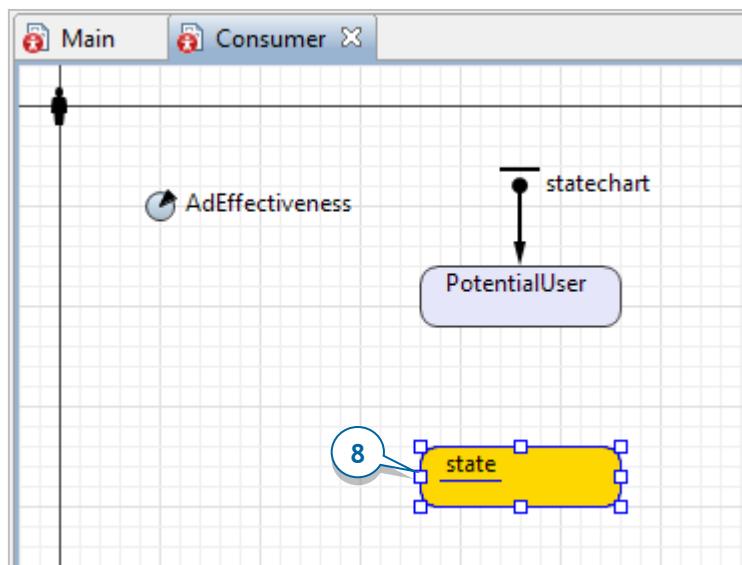
Код, заданный в поле **Действие при входе**, будет выполнен в момент перехода потребителя в это состояние стейтчарта. Вызываемая здесь функция поможет нам понять, что произошла смена состояния потребителя, изменив цвет его фигуры анимации.



Используемое здесь *shapeBody* является именем фигуры анимации потребителя, которую мы выбрали в мастере создания агентов. (Если вы раскроете ветку **Презентация** типа агента *Consumer* в панели **Проекты**, то вы увидите фигуру *shapeBody* внутри группы *person*).

Далее мы вызываем функцию фигуры *shapeBody*. Чтобы получить доступ к функции элемента, введите имя элемента (*shapeBody*), поставьте точку, затем вызовите мастер подстановки кода, чтобы просмотреть все функции элемента и выбрать функцию из списка. Функция *setFillColor()* является стандартной функцией фигуры, которая позволяет динамически изменять цвет заливки фигуры. У этой функции только один аргумент – новый цвет. Мы передаем в качестве значения имя специальной константы AnyLogic, задающей лавандовый цвет (*lavender*). Примеры других цветовых констант - *red, yellow, green, purple, silver, black* и т.д.

**8.** Добавьте еще одно состояние в диаграмму состояний потребителя:

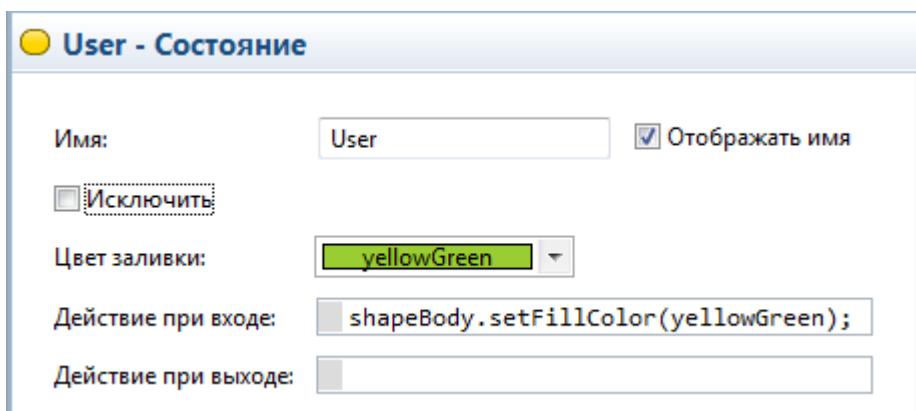


**9.** Измените свойства этого состояния:

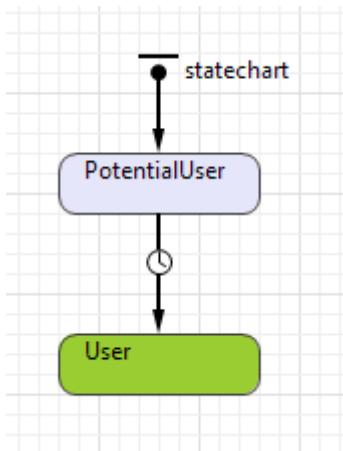
Имя: *User*

Цвет заливки: *yellowGreen*

Действие при входе: *shapeBody.setFillColor(yellowGreen);*

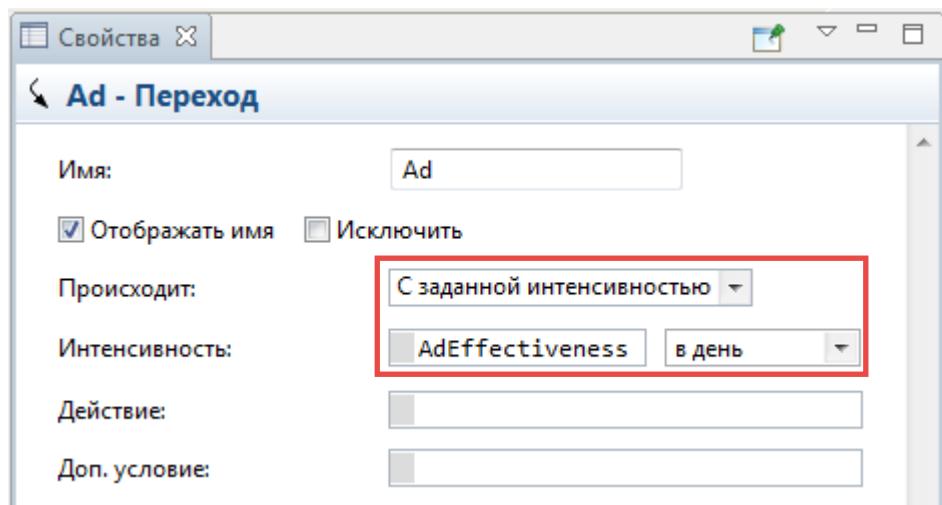


10. Нарисуйте переход из состояния *PotentialUser* в состояние *User*, чтобы промоделировать, как человек приобретает продукт (и становится его потребителем). Для этого сделайте двойной щелчок мышью по элементу **Переход** в палитре **Диаграмма состояний** (значок элемента при этом должен поменяться на ) , затем щелкните сначала по состоянию *PotentialUser*, а потом по состоянию *User*.



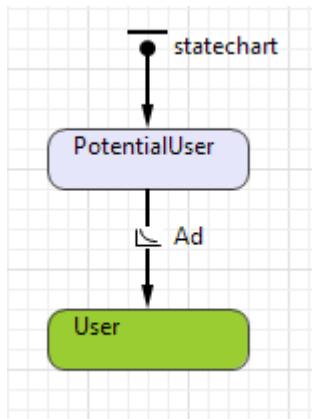
- ◆ Убедитесь, что переход соединяет состояния. Если переход некорректно соединен, AnyLogic выделит его красным цветом.
11. Назовите переход *Ad*, потому что он представляет действие рекламы, от английского слова «advertisig».

12. Поставьте галочку в свойстве **Отображать имя**, чтобы имя перехода показывалось в графическом редакторе.
13. Переход, ведущий из состояния *PotentialUser* в состояние *User*, будет моделировать процесс покупки продукта под воздействием рекламы. В свойстве перехода **Происходит** выберите опцию **С заданной интенсивностью**. В появившемся поле **Интенсивность** введите имя переменной *AdEffectiveness*, а справа выберите единицы интенсивности срабатывания перехода - **в день**.



Вы увидите, как значок перехода изменится с на . Этот значок отображает тип срабатывания перехода.

Если вы захотите переместить имя перехода или его значок, то вначале выделите этот переход в редакторе, а затем перетащите имя/значок мышью в новое местоположение.



### Тип срабатывания перехода

Переход из одного состояния в другое может быть вызван событиями различных типов. Приведенная ниже таблица перечисляет все возможные типы таких событий. Для каждого типа приведен специальный значок, по которому вы легко можете распознать тип перехода в диаграмме состояний.

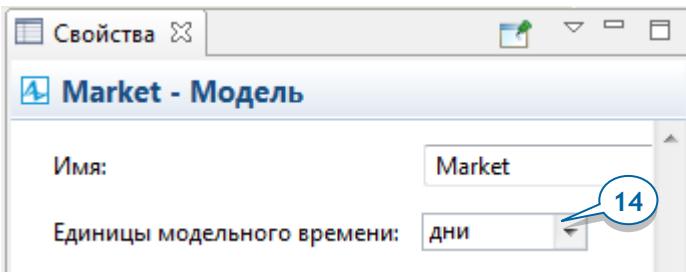
Переход происходит:	Описание
По таймауту 	Срабатывает после того, как состояние диаграммы пребудет активным в течение заданного промежутка времени (таймаута) с момента перехода управления в это состояние. Таймаут может быть задан как определенным числом, так и стохастическим (случайным) выражением.
С заданной интенсивностью 	Моделирует смену состояния через случайный промежуток времени, когда известно среднее время пребывания в состоянии. Аналогичен переходу по таймауту, у которого таймаут считается согласно экспоненциальному распределению, параметром которого является значение интенсивности. Если интенсивность = 0.2, то время пребывания в состоянии будет в среднем равно $1/0.2 = 5$ единиц модельного времени.

<b>При выполнении условия</b> 	<p>Переход отслеживает выполнение заданного логического (булевского) условия и срабатывает, когда это условие будет выполнено. Условие может зависеть как от текущего состояния данного агента, так и от состояний других объектов моделируемой системы.</p> <p>Условие проверяется только в моменты происхождения событий в модели. Чтобы быть уверенными в том, что модель не пропустит момент срабатывания перехода, рекомендуется добавить на диаграмму этого агента циклически срабатывающее событие, и задать для этого события достаточно малый таймаут срабатывания.</p>
<b>При получении сообщения</b> 	<p>Реагирует на получение сообщений от других агентов. Сообщения чаще всего моделируют общение/взаимодействие агентов друг с другом. Вы можете задать в свойствах перехода шаблон сообщения, в этом случае переход будет срабатывать только при получении сообщений, удовлетворяющих заданному шаблону.</p>
<b>По прибытию агента</b> 	<p>Срабатывает по прибытии агента в заданное место назначения.</p> <p><i>Обратите внимание, что переход данного типа будет срабатывать только в том случае, если движение было начато вследствие вызова функции агента <code>moveTo()</code>.</i></p>

Наш переход происходит с заданной интенсивностью. В нашем случае, когда управление диаграммы состояний переходит в состояние *PotentialUser*, происходит вычисление времени срабатывания перехода согласно экспоненциальному распределению. Время до покупки продукта для каждого отдельного потребителя будет отличаться, но в среднем продукт будет приобретать 1% потенциальных потребителей в день.

14. Давайте теперь зададим единицы модельного времени. Чтобы изменить настройки модели, переключитесь из **Палитры** в панель **Проекты**, и затем щелкните по элементу модели в дереве (самый верхний уровень дерева,

элемент Market). Перейдите в панель Свойства и выберите дни в качестве Единиц модельного времени.

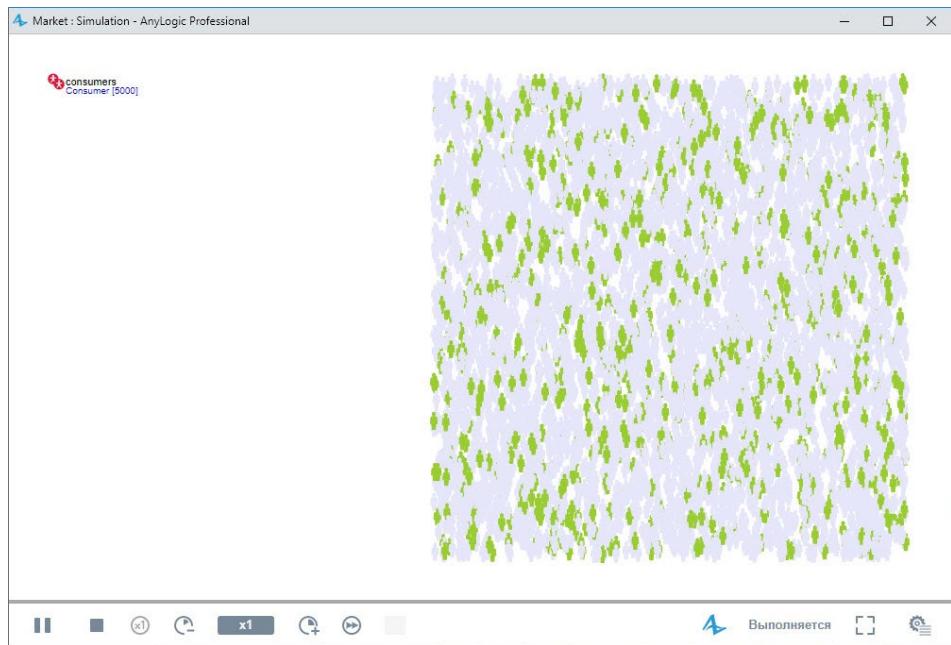


### Модельное время. Единицы модельного времени

- Модельное время – это виртуальное (моделируемое) время ("внутреннее" время "движка" AnyLogic). Модельное время никак не соотносится с реальным временем и часами на компьютере, хотя вы и можете выполнять модель с привязкой модельного времени к реальному.
- Чтобы определить соотношение модельного времени ко времени реального мира, в котором существует моделируемая система, вам нужно задать единицы модельного времени. Выберите наиболее подходящие единицы модельного времени в соответствии с длительностью типичных операций в вашей модели.
- К примеру, в пешеходных моделях, в качестве единиц модельного времени, как правило, используются секунды, в моделях производства и системах обслуживания – минуты, а в глобальных экономических, экологических или социальных моделях – месяцы или даже годы.

- 15.** Запустите модель. Популяция агентов постепенно окрашивается в зеленый цвет (изменение, к которому приводит эффект рекламы), пока каждый потенциальный потребитель не купит продукт.

Когда в результате просмотра рекламы агент принимает решение о покупке продукта, его состояние *User* становится активным и выполняется **Действие при входе** этого состояния, меняющее цвет фигуры анимации этого агента на светло-зеленый (*yellowGreen*). Чем больше людей покупают продукт, тем больше появляется зеленых фигур анимации агентов.



## Режимы выполнения модели

Вы можете выполнять модель AnyLogic в режиме *реального* или *виртуального* времени.

- В режиме *реального времени* вы устанавливаете соотношение между модельным и реальным временем, выбирая, сколько единиц модельного времени проходит за одну секунду астрономического времени. Как правило, режим реального времени используется тогда, когда вы хотите следить за анимацией моделируемых процессов.
- В режиме *виртуального времени* модель выполняется на максимальной скорости. Режим виртуального времени пригодится вам для того, чтобы моделировать длительный период жизни модели. При этом соотношение между единицами модельного времени и секундами реального (астрономического) времени не задается, и анимация модели может отображаться "рывками".

Панель управления **Модельное время** позволяет вам настраивать скорость выполнения модели в режиме реального времени:

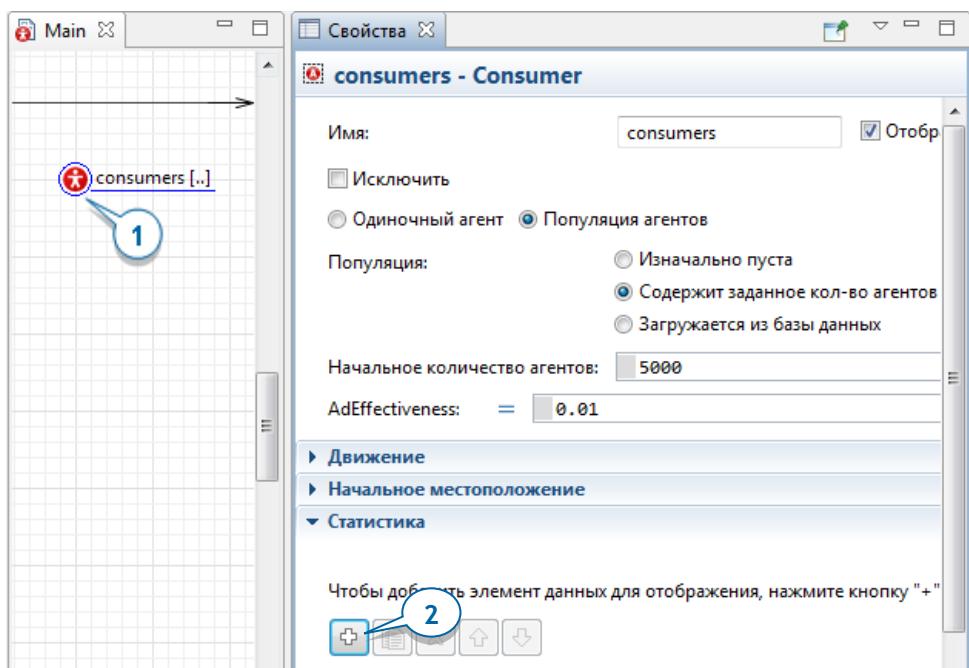


- 16.** Чтобы изменить скорость выполнения модели, щелкните по кнопке **Замедлить** или **Ускорить** на панели управления окна запущенной модели. Увеличьте скорость выполнения модели в десять раз (выбрав значение  $x10$ ), и вы увидите, что популяция будет быстрее окрашиваться в зеленый цвет.

## Фаза 3. Добавление графика для визуализации результатов моделирования

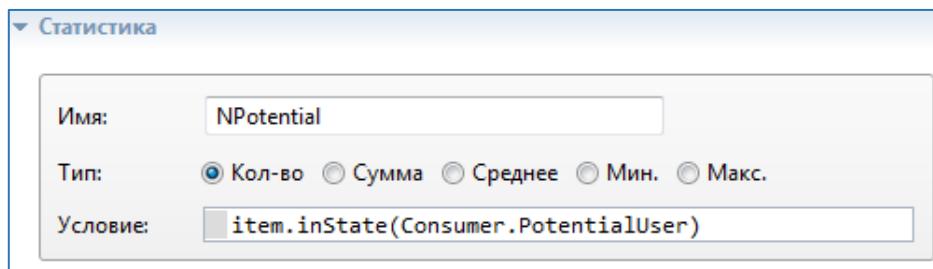
Мы хотим знать, сколько людей приобрело наш продукт в определенный момент времени. Для этого мы зададим функции, которые будут считать количество потребителей и потенциальных потребителей продукта соответственно, а затем добавим график, чтобы наблюдать за динамикой изменения рынка.

1. Сначала зададим функцию, которая будет считать количество потенциальных потребителей. Чтобы добавить новую функцию подсчета статистики по популяции агентов, откройте диаграмму агента *Main*, выделите популяцию агентов *consumers* и перейдите в раздел свойств **Статистика**.
2. Щелкните по кнопке  **Добавить**.



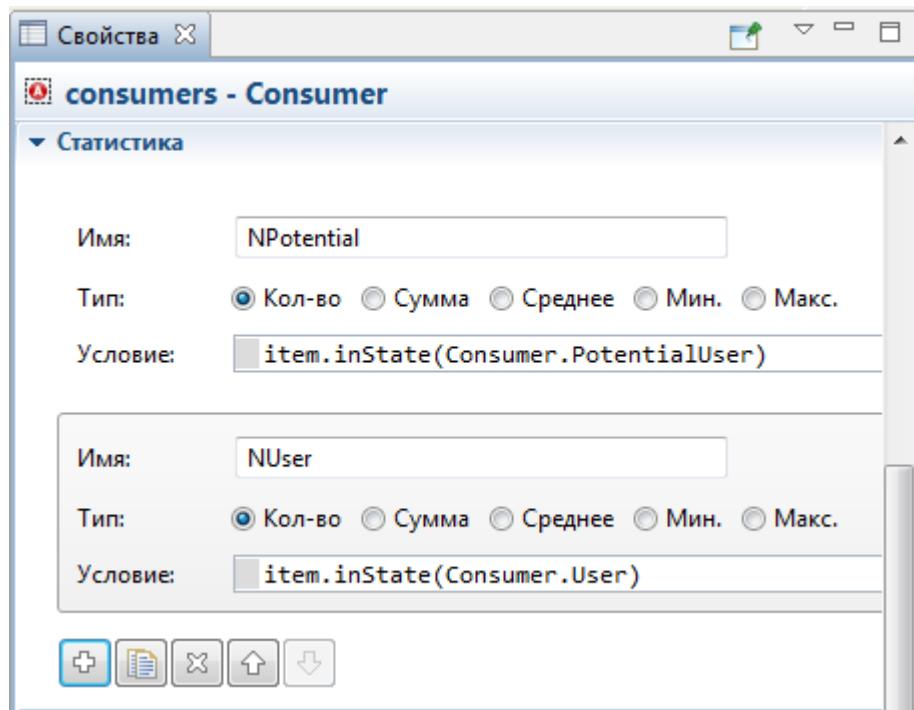
Мы хотим узнать, сколько агентов находятся в состоянии *PotentialUser*.

3. Задайте функцию типа **Кол-во**, в поле **Имя** введите *NPotential*. Функция статистики типа **количество** проходит по всем агентам популяции и подсчитывает тех агентов, для которых выполняется заданное условие.

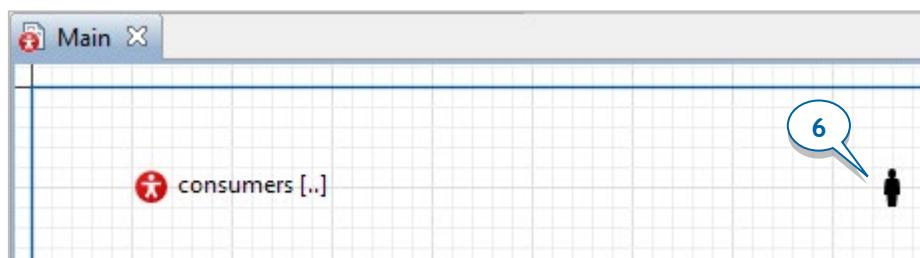


4. Введите *item.inState(Consumer.PotentialUser)* в качестве **Условия** функции.

- *item* - локальная переменная, предоставляющая доступ к агенту, проверяемому в данный момент в процессе итерирования по популяции.
- Функция *inState()* проверяет, является ли для этого агента активным указанное состояние диаграммы состояний.
- *PotentialUser* – имя состояния. Поскольку оно имеет смысл для агента определенного типа, мы добавляем к имени префикс соответствующего типа агента - *Consumer*.

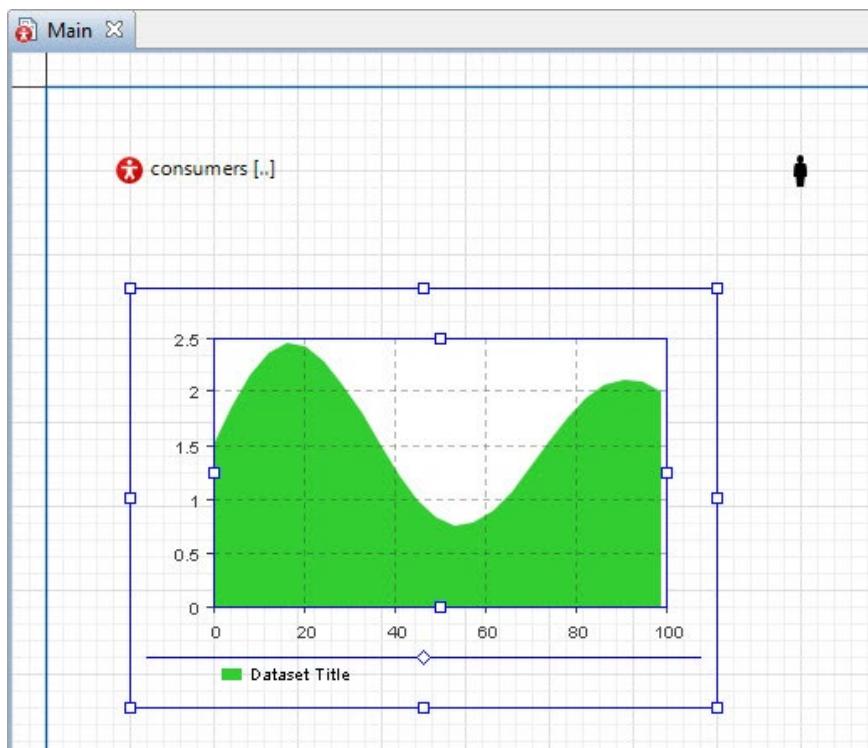


5. Задайте вторую функцию статистики для подсчета потребителей продукта. Назовите ее *NUser*. Пусть она считает количество агентов, для которых выполняется Условие *item.inState(Consumer.User)*. Вы можете создать копию ранее созданной функции сбора статистики, щелкнув по кнопке Дублировать и изменив Имя и Условие созданной функции.
6. На диаграмме *Main*, переместите вправо фигуру презентации популяции агентов *consumers*.



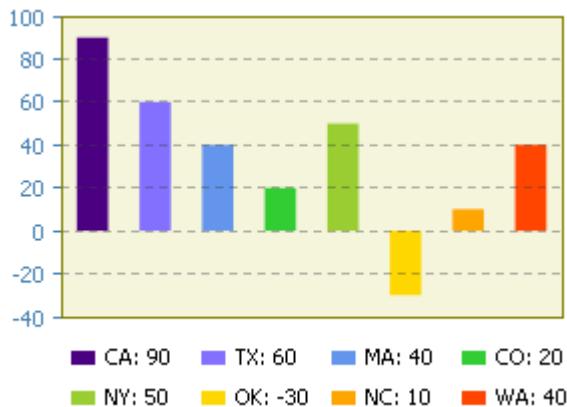
Теперь давайте добавим график для визуального отображения статистики, собираемой заданными только что функциями, и понаблюдаем за динамикой внедрения нового продукта на рынок.

7. Откройте палитру **Статистика**  и перетащите элемент **Временная диаграмма с накоплением**  из палитры на диаграмму *Main*, чтобы создать график, который будет отображать динамику изменений числа потенциальных потребителей и владельцев продукта. Увеличьте размер временной диаграммы с накоплением, как показано на рисунке ниже:



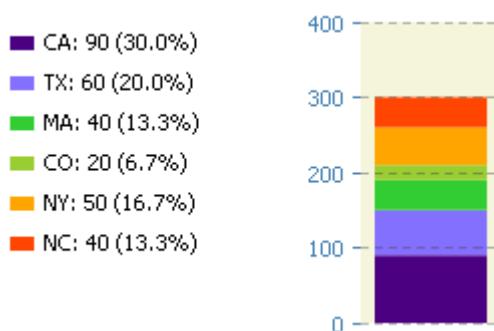
## Диаграммы

С помощью диаграмм вы можете визуализировать данные, полученные в результате прогона модели. Все диаграммы находятся в палитре **Статистика**, раздел **Диаграммы**. Ниже мы кратко рассмотрим каждую из них.



### Столбиковая диаграмма

Отображает несколько элементов данных в виде столбцов, «растущих» в заданном направлении. Размеры столбцов пропорциональны значениям соответствующих элементов данных.



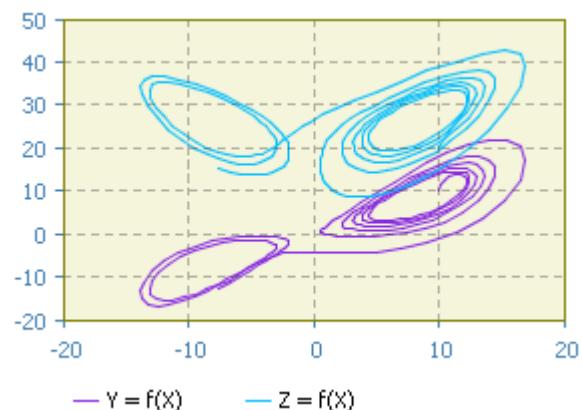
### Диаграмма с накоплением

Показывает вклад нескольких элементов данных в суммирующий результат в виде столбцов, расположенных друг над другом. Высота каждого столбца пропорциональна значению соответствующего элемента данных.

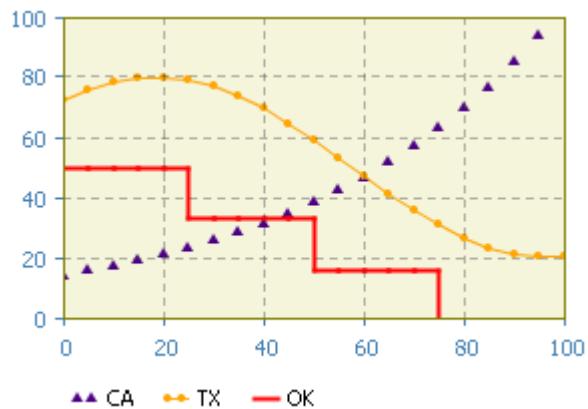


### Круговая диаграмма

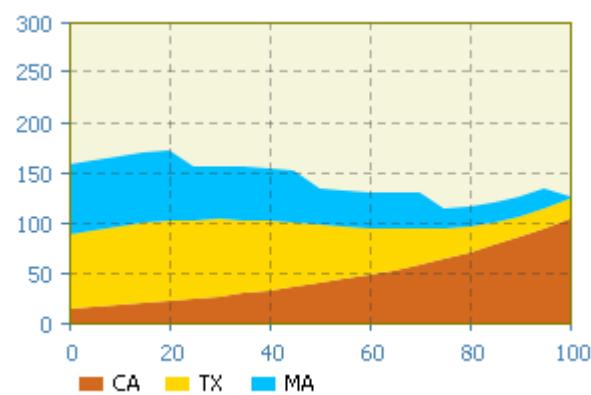
Отображает вклад нескольких элементов данных в общую составляющую в виде секторов круга. Дуги секторов пропорциональны значениям соответствующих элементов данных.

**График**

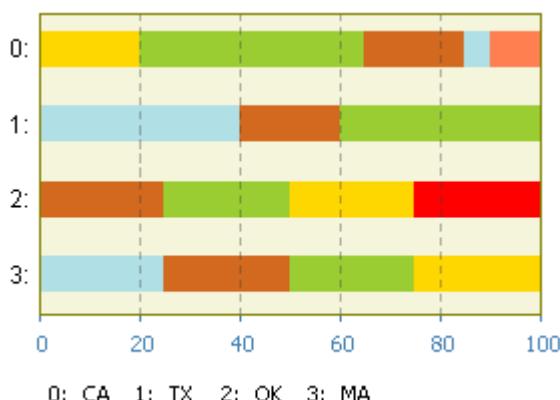
Обычно выполняет роль фазовой диаграммы. Отображает зависимость Y-компонент значений набора данных от соответствующих им X-компонент. Каждому измерению набора данных на графике соответствует точка с координатами  $\langle x, y \rangle$ .

**Временной график**

Отображает динамику изменения (временной тренд) одного или нескольких наборов данных в течение последних N единиц модельного времени (заданного временного диапазона).

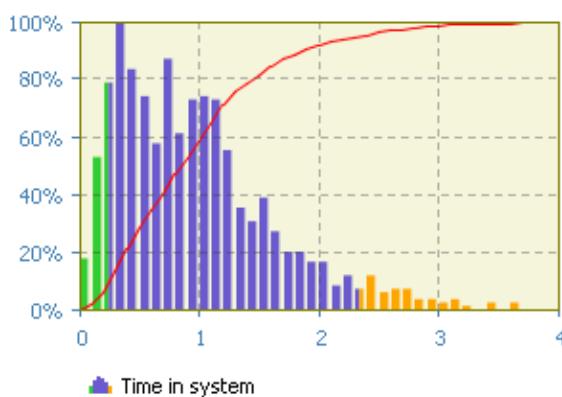
**Временная диаграмма с накоплением**

Отображает в виде располагающихся друг над другом областей историю вклада нескольких наборов данных в общую составляющую в течение заданного временного диапазона.



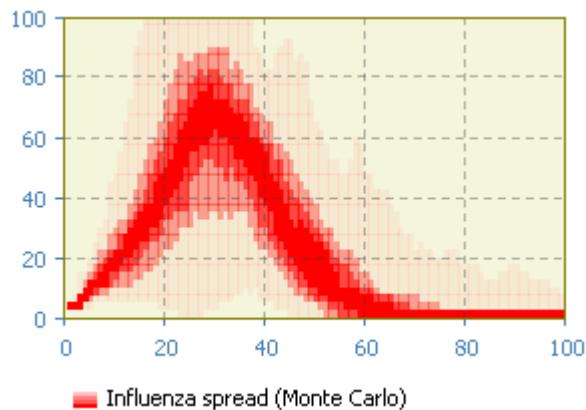
### Временная цветовая диаграмма

Отображает динамику изменения данных в виде столбцов, каждый из которых составляется из полосок различного цвета (цвет зависит от того, какое из заданных условий для отображаемого данным столбцом набора данных верно в текущий момент).



### Гистограмма

Отображает статистику, собранную элементом **Данные гистограммы**. Высота каждого прямоугольника пропорциональна числу элементов выборки, попавших в соответствующий интервал по оси X.



### Гистограмма 2D

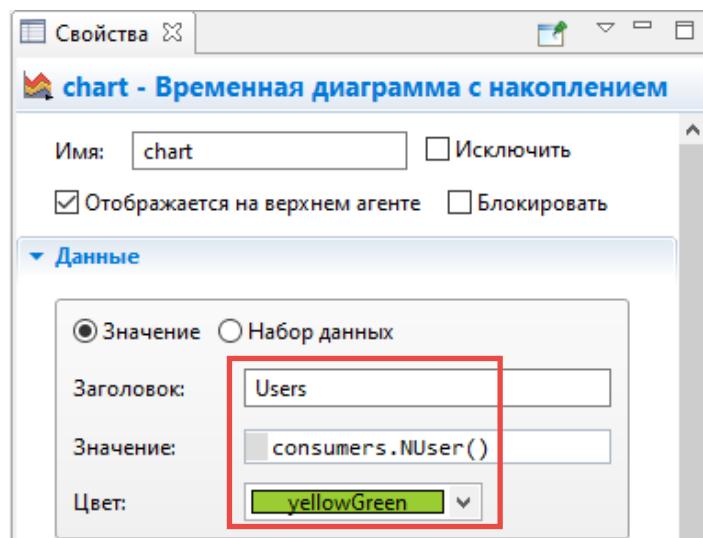
Отображает статистику, собранную элементом **Данные двумерной гистограммы**. Гистограмма отображается в виде закрашенных разным цветом прямоугольников (ячеек), отражающих значение плотности вероятности в соответствующей точке (X,Y), или вложений ("конвертов").

Укажите, какие данные будет отображать график. Мы воспользуемся теми самыми функциями статистики *NUser* и *NPotential*, которые мы создали ранее для популяции *consumers*.

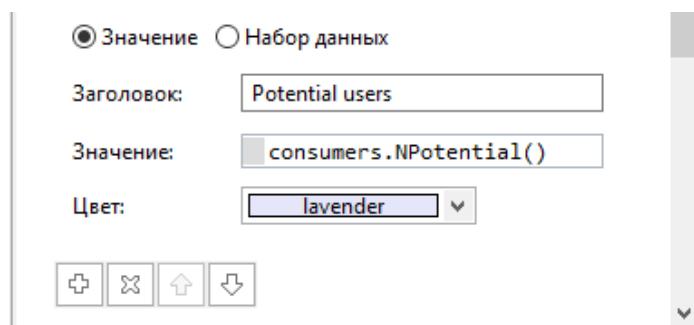
- В секции **Данные** свойств временной диаграммы с накоплением, измените свойства элемента данных следующим образом:

- Заголовок:** *Users* – заголовок элемента данных.
- Цвет:** *yellowGreen*
- Значение:** *consumers.NUser()*

Здесь *consumers* - это имя нашей популяции агентов, а *NUser()* - это функция сбора статистики, которую мы задали ранее.

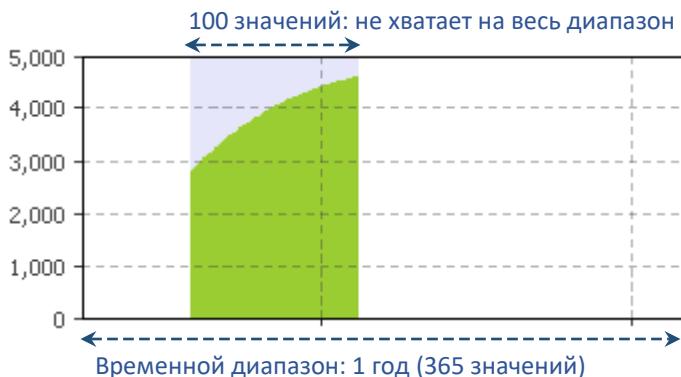


- Добавьте еще один элемент данных, щелкнув по кнопке **Добавить**. Задайте для этого элемента следующие свойства:
- Заголовок:** *Potential users*
  - Цвет:** *lavender*
  - Значение:** *consumers.NPotential()*



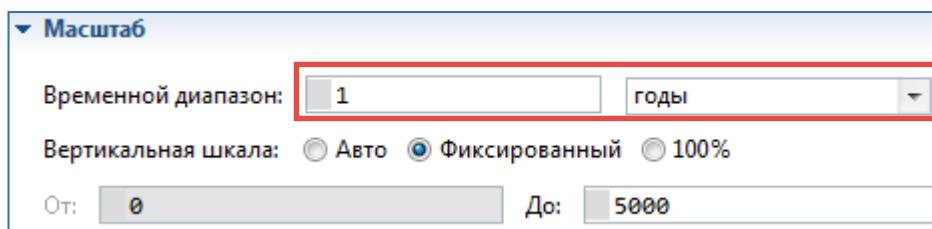
## Подбор временного масштаба графика

- Диаграммы, используемые для отображения динамики изменения значений во времени (временной график, временная диаграмма с накоплением, временная цветовая диаграмма), позволяют вам настраивать временной диапазон - интервал времени, для которого диаграмма отображает значения.
- Если в процессе выполнения модели вы увидите, что ваш график напоминает приведенный на рисунке ниже, то вам будет нужно увеличить количество значений, отображаемых на графике или же уменьшить временной интервал графика.

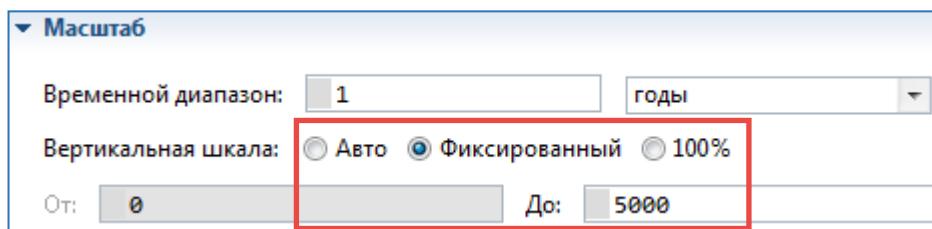


Так как мы хотим видеть данные сразу за целый год, нам нужно изменить свойства графика.

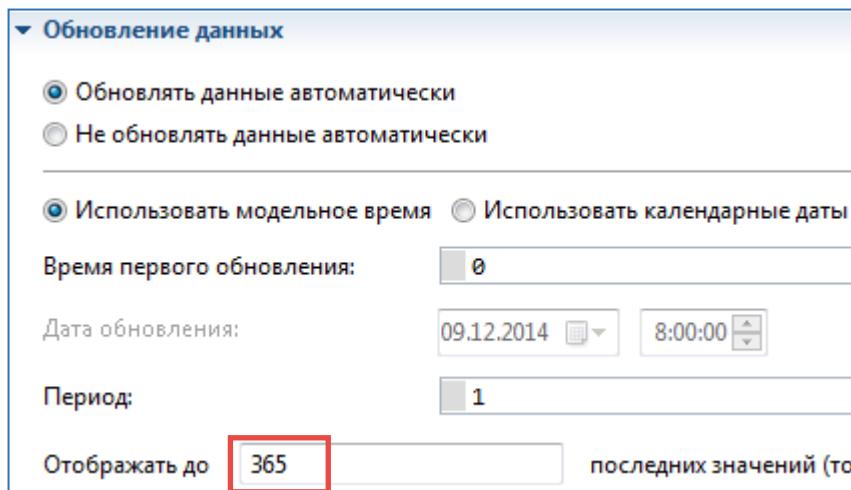
10. Перейдите в раздел свойств **Масштаб** и задайте **Временной диапазон**, равный **1 году**.



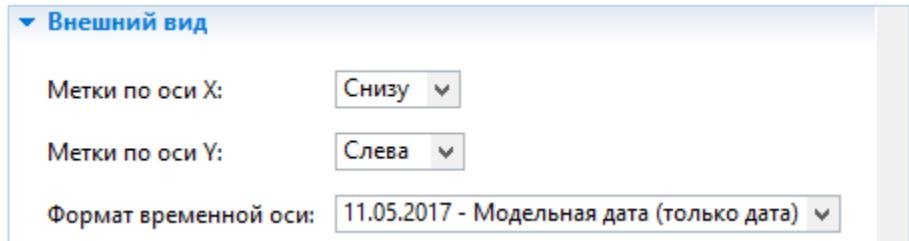
11. Теперь задайте диапазон значений для оси Y графика. Так как наш график будет показывать статистику для популяции *consumers*, а потребителей в нашей модели 5000, установите **Фиксированный** тип **Вертикальной шкалы** и введите 5000 в поле **До**.



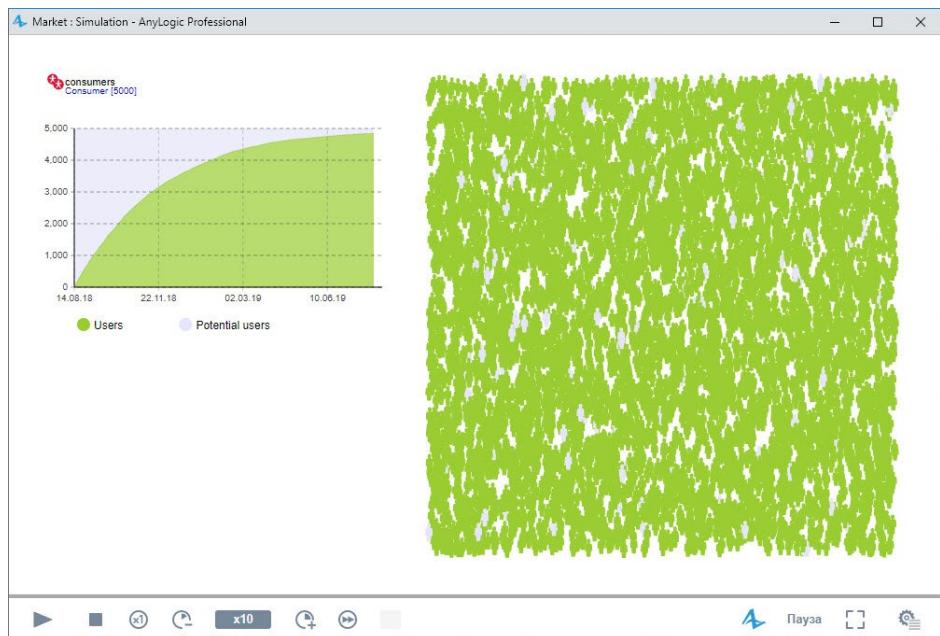
12. Теперь, когда мы задали временной диапазон, давайте изменим максимальное количество значений данных, отображаемых на графике. Перейдите в секцию **Обновление данных** и введите 365 в поле **Отображать до... последних значений**. Так как мы добавляем по одному значению каждый день, то это идеально подходит для временного диапазона длительностью в один год.



13. Перейдите в секцию свойств графика **Внешний вид** и выберите опцию **Модельная data (только data)** из списка **Формат временной оси**. Мы меняем формат меток по временной оси графика - теперь они будут отображать только дату, но без времени, тем самым метки станут более компактными, и внешний вид графика улучшится.



14. Запустите модель и понаблюдайте за моделируемым процессом с помощью добавленной нами диаграммы.



## Фаза 4. Добавление эффекта рекомендаций

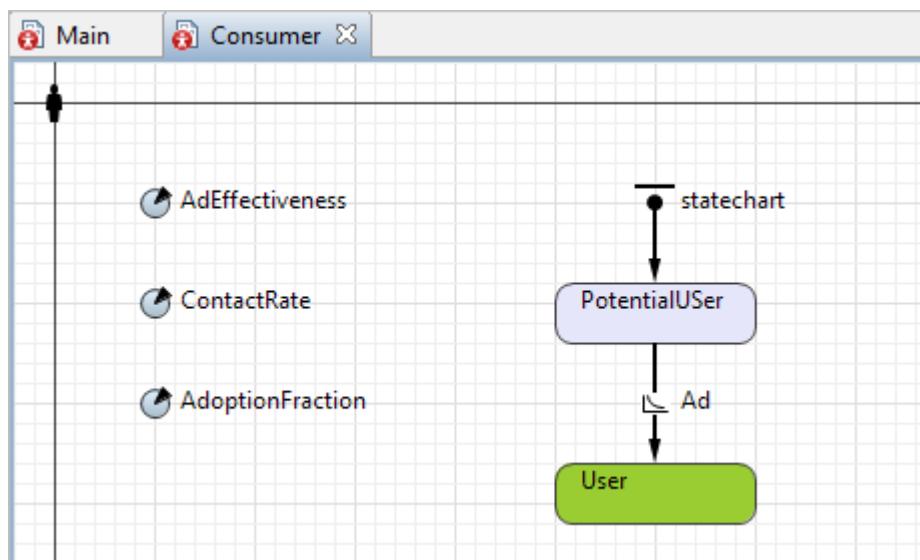
Теперь мы промоделируем эффект, который оказывают на потенциальных потребителей положительные отзывы о продукте его владельцев.

- В нашей модели каждый человек в течение дня будет в среднем общаться с одним своим знакомым.
- Во время общения друг с другом владельцы продукта могут повлиять на потенциальных потребителей. Мы зададим вероятность приобретения продукта потенциальным потребителем под воздействием общения с помощью параметра  $AdoptionFraction = 0.01$ .

Для начала добавим два новых параметра: *ContactRate* (определяет интенсивность контактов) и *AdoptionFraction* (вероятность приобретения продукта в результате общения с пользователем этого продукта).

1. Откройте диаграмму типа агента *Consumer*, сделав двойной щелчок по элементу *Consumer* в панели **Проекты**.
2. Добавьте параметр, который будет задавать среднее количество контактов потребителя с другими людьми в течение дня. Перетащите элемент **Параметр**  из палитры **Агент**  на диаграмму агента *Consumer*.
3. Назовите параметр *ContactRate*.
4. В данной модели средняя интенсивность контактов равна одному контакту в день. Перейдите в свойства этого параметра и введите 1 в поле **Значение по умолчанию**.
5. Добавьте еще один параметр, *AdoptionFraction*, который задает вероятность приобретения продукта в результате общения с пользователем этого продукта. В свойствах данного параметра задайте **Значение по умолчанию: 0.01**.

Диаграмма типа агента *Consumer* теперь выглядит так:



Теперь промоделируем общение агентов между собой – те самые разговоры о продукте, побуждающие людей к его приобретению.

## Взаимодействие агентов

Взаимодействие агентов в AnyLogic чаще всего реализуется с помощью *передачи сообщений*. Агент может посылать сообщения какому-то определенному агенту или группе агентов. Сообщение может представлять собой строку текста, число, и вообще любой объект Java, со своей структурой данных и множеством параметров.

Для отправки сообщения вызываются специальные функции агента (самые популярные из них приведены ниже):

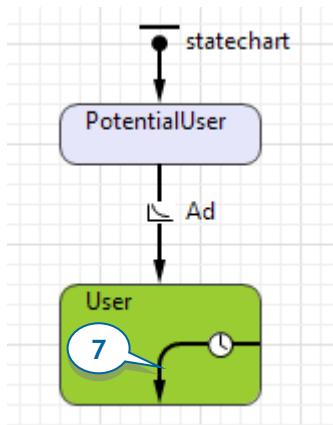
*sendToAll( msg )* – агент отправляет сообщение *msg* всем другим агентам своей популяции.

*sendToRandom( msg )* – агент отправляет сообщение *msg* одному случайно выбранному агенту из своей популяции.

*send( msg, agent )* – агент отправляет сообщение *msg* указанному агенту *agent* (вы передаете ссылку на агента-получателя с помощью второго аргумента функции)

В нашей модели сообщения будут посыпать только те агенты, которые находятся в состоянии *User*. Лучшим способом задать действие, которое агент выполняет, не выходя из текущего состояния, является добавление *внутреннего перехода*.

6. Откройте диаграмму агента *Consumer* и измените размер состояния *User* (см. рисунок ниже), чтобы в него поместился внутренний переход, который мы сейчас добавим.
7. Нарисуйте переход внутри состояния *User*, как показано на рисунке ниже. Для этого перетащите элемент **Переход**  из палитры **Диаграмма состояния**  внутрь состояния, чтобы начальная точка перехода расположилась на границе состояния. Затем поместите на границу состояния и конечную точку этого перехода. Чтобы добавить изгиб фигуры перехода, сделайте по переходу двойной щелчок мышью.

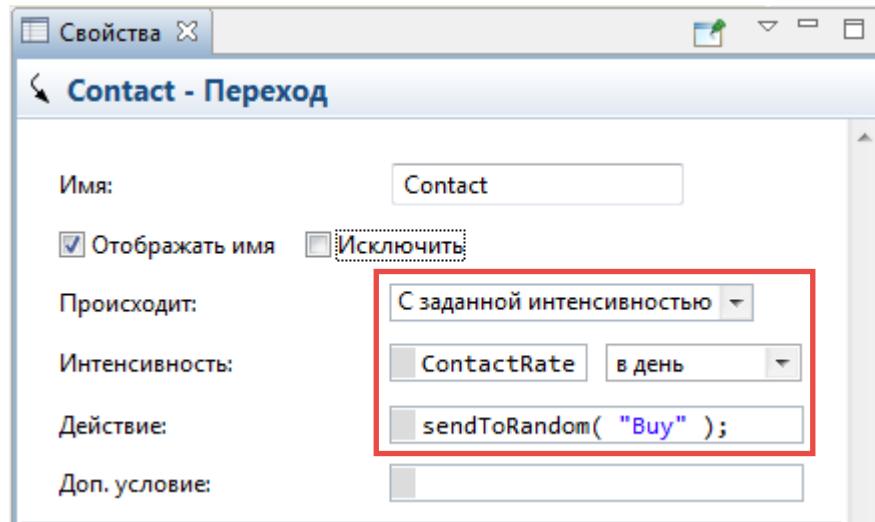


- ◆ Внутренние и внешние переходы отличаются по своей логике, поэтому убедитесь, что этот переход лежит полностью *внутри состояния*.

## Внутренние переходы

- Внутренний переход располагается внутри состояния. Обе крайние точки такого перехода находятся на границе состояния.
- Так как внутренний переход не выходит за границы состояния, то он не выводит диаграмму из этого состояния. При срабатывании такого перехода не выполняются ни действие **При входе**, ни действие **При выходе** этого состояния.

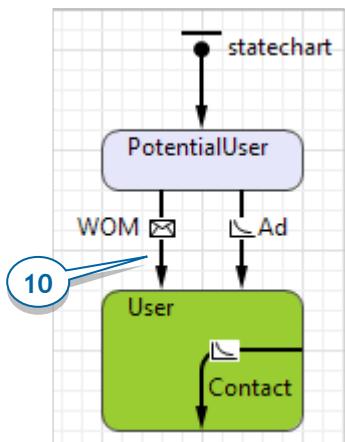
8. Измените свойства перехода. Этот переход будет происходить **С заданной интенсивностью**, равной значению параметра *ContactRate* (чтобы не печатать имя параметра полностью, используйте мастер подстановки кода). Назовите переход *Contact* и включите отображение имени этого перехода в графическом редакторе.



9. Укажите **Действие**, которое должно выполняться при срабатывании перехода: `sendToRandom("Buy");`

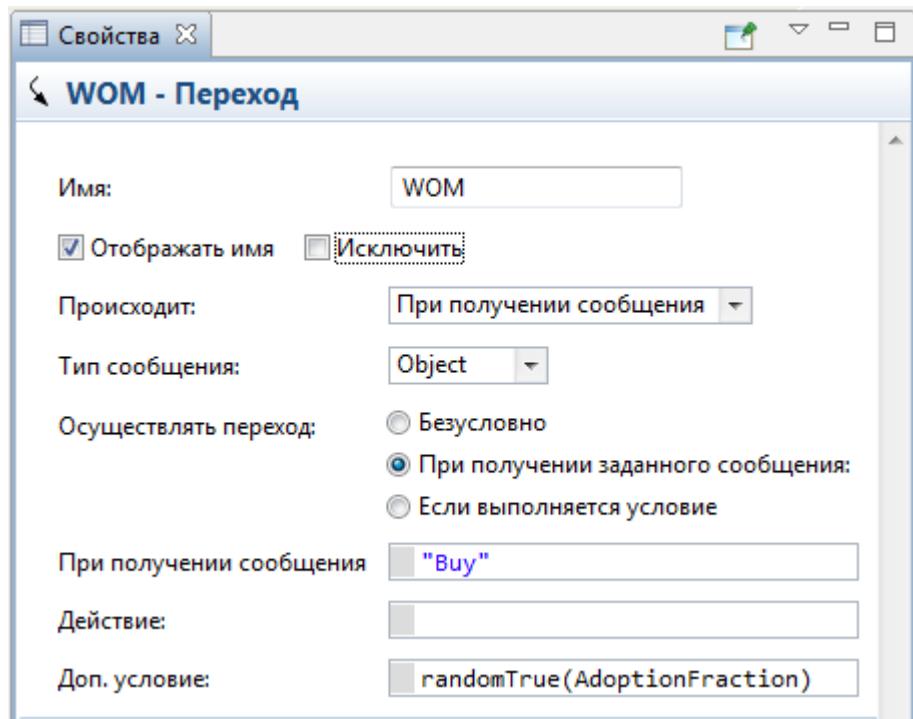
Мы создали циклический переход в состоянии *User*. Каждый раз, когда срабатывает этот переход, вызываемая функция `sendToRandom("Buy")`; моделирует отправку этим потребителем сообщения "Buy" случайно выбранному агенту. Если агент, который получает сообщение, является потенциальным потребителем (то есть находится в состоянии *PotentialUser*), то текущим состоянием агента-получателя станет состояние *User* (согласно еще одному переходу, который мы нарисуем сейчас).

10. Нарисуйте еще один переход из состояния *PotentialUser* в состояние *User* и назовите его *WOM*. Этот переход будет моделировать покупку продукта в результате рекомендаций других людей.



11. Измените свойства перехода:

- В списке **Происходит** выберите **При получении сообщения**.
- В свойстве **Осуществлять переход** выберите **При получении заданного сообщения**.
- В поле **Сообщение** ниже введите сообщение "**Buy**"
- Так как мы понимаем, что не каждый контакт приводит к новым продажам, то мы ограничим долю успешных контактов с помощью параметра *AdoptionFraction*. Задайте следующее **Доп. условие** перехода: `randomTrue(AdoptionFraction)`



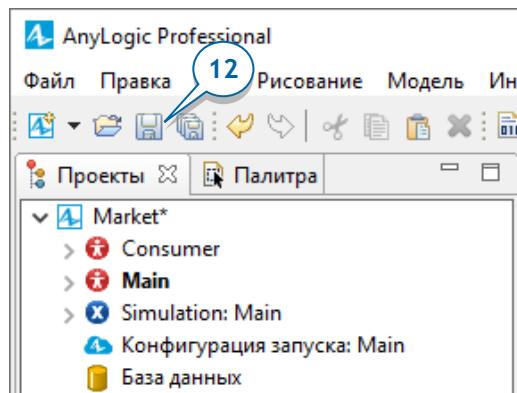
### Дополнительные условия переходов

- При переходе в простое состояние инициируются все исходящие переходы этого состояния, и диаграмма состояний начинает ждать, когда один из них произойдет.
- При происхождении события, ведущего к срабатыванию перехода, также оценивается дополнительное условие этого перехода. Если это условие выполняется, то тогда переход срабатывает.

Итак, сейчас в нашей модели агент-потребитель отсылает сообщения другим агентам. Если агент-получатель является потенциальным потребителем (то есть, находится в состоянии *PotentialUser*), то это сообщение вызовет переход управления его диаграммы в состояние *User*. Если диаграмма состояний агента-получателя уже находится в состоянии *User*, то сообщение будет просто проигнорировано.

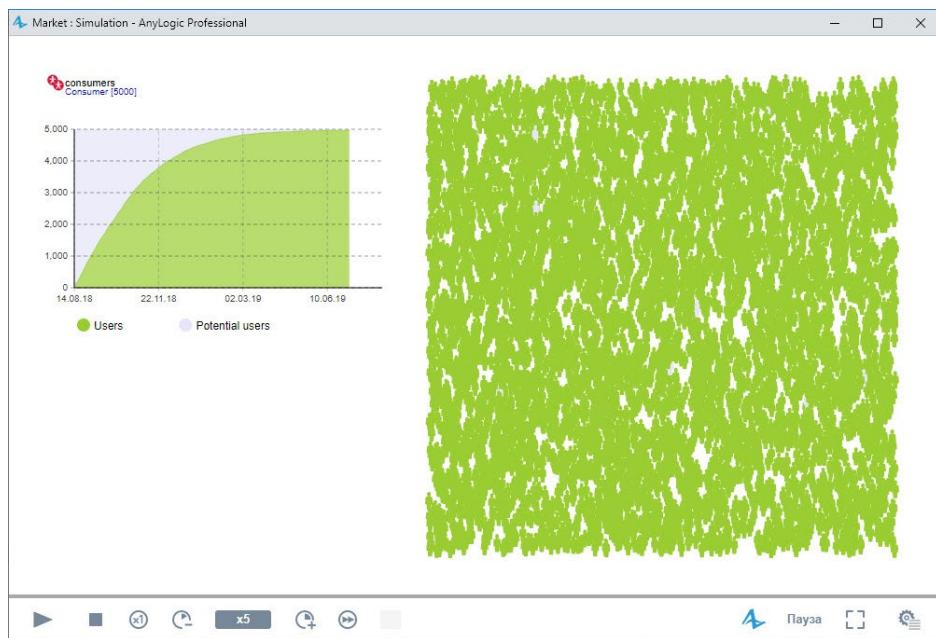
12. Перейдите в панель **Проекты**. Если вы видите звездочку у имени модели, значит, в вашей модели есть несохраненные изменения. Щелкните по

кнопке панели управления Сохранить модель, чтобы сохранить изменения.



13. Запустите модель.

Насыщение рынка теперь будет происходить быстрее, а график покажет известную S-образную кривую выхода нового продукта на рынок.

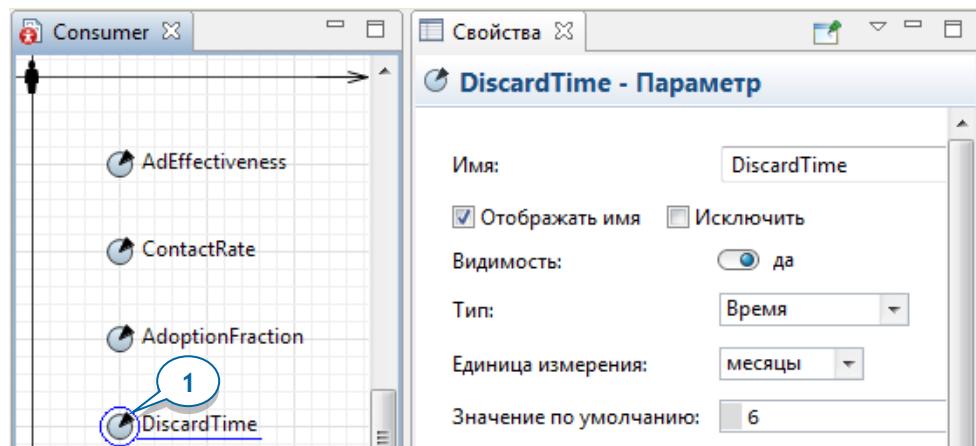


## Фаза 5. Учет повторных продаж продукта

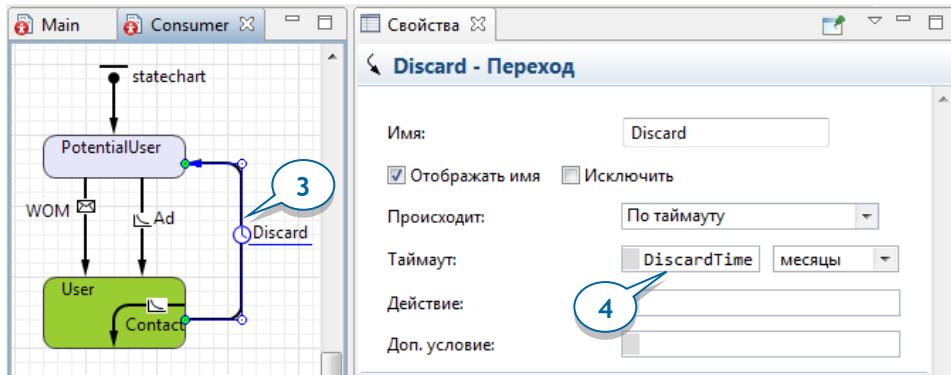
Допустим, что у рассматриваемого нами продукта ограниченный срок годности (или срок эксплуатации), равный шести месяцам.

Когда потребитель больше не сможет пользоваться продуктом, ему понадобится замена продукта. Мы смоделируем повторные покупки, предположив, что по истечении срока годности товара потребители вновь становятся потенциальными потребителями (то есть, агенты переходят из состояния *User* обратно в состояние *PotentialUser*).

1. Откройте диаграмму агента *Consumer* и добавьте на нее параметр *DiscardTime*.

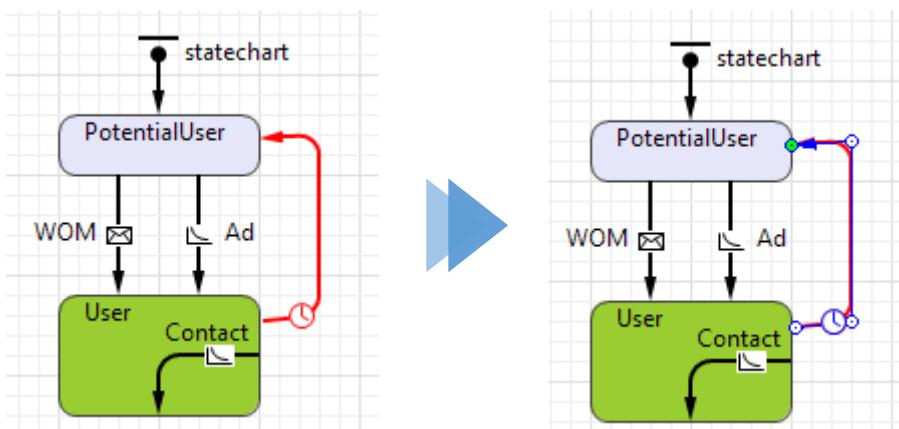


2. Этот параметр задает срок службы нашего продукта. Выберите Время в качестве Типа параметра, затем выберите месяцы в списке Единица измерения и введите 6 в поле Значение по умолчанию.
3. Нарисуйте переход из состояния *User* в состояние *PotentialUser*, чтобы промоделировать истечение срока службы товара. Мы хотим нарисовать переход сложной формы, как на рисунке ниже. Для этого сделайте двойной щелчок мыши по элементу Переход в палитре Диаграмма состояний (значок элемента при этом сменится на ). Чтобы начать рисование, щелкните по состоянию *User*, после чего щелкните по холсту диаграммы в тех точках, где вы хотите поместить точки изгиба перехода. Завершите рисование, сделав двойной щелчок мыши по конечному состоянию перехода *PotentialUser*.



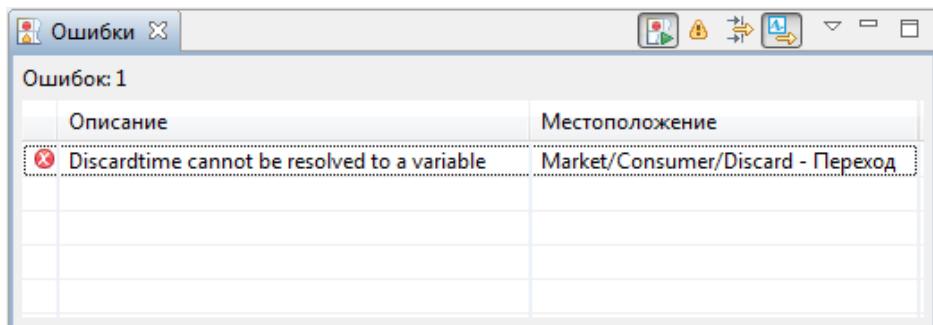
4. Назовите переход *Discard*, пусть он срабатывает по таймауту, равному значению параметра *DiscardTime*. В списке справа от значения, выберите **месяцы**.

◆ AnyLogic выделяет красным цветом переходы, не соединенные с состоянием (как на рисунке слева). Поскольку визуально обе точки у перехода могут казаться соединенными с соответствующими состояниями, воспользуемся дополнительным инструментом диагностики. Чтобы понять, где именно допущена ошибка, выделите такой переход. Точки корректного соединения будут выделены зеленым цветом. Если AnyLogic не подсвечивает зеленым точку соединения перехода и состояния, вам нужно вручную подвинуть эту точку на границу состояния, и тем самым исправить ошибку.



## Исправление опечаток

Иногда при компиляции модели показывается сообщение о том, что была обнаружена ошибка - не удается распознать то или иное имя:



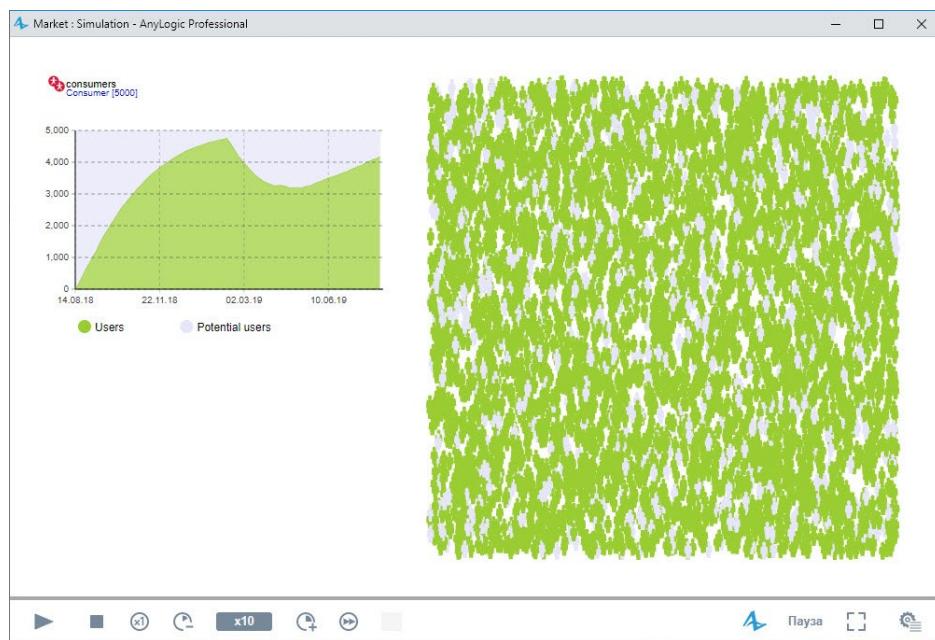
Чаще всего такая ошибка вызвана тем фактом, что AnyLogic учитывает регистр клавиатуры в именах элементов. В данном случае ошибка заключается в том, что буква *t*, с которой начинается слово в имени параметра *DiscardTime*, изначально в имени параметра - заглавная, а пользователь ссылается на это имя, используя прописную букву *t*, и поэтому имена не совпадают, и происходит ошибка компиляции модели.

Чтобы исправить ошибку, дважды щелкните по строке этой ошибки в панели **Ошибки**. Если ошибка графическая, то AnyLogic подсветит вызвавший ошибку элемент в графическом редакторе. Если ошибка допущена при задании значения какого-то свойства элемента, то AnyLogic откроет свойства этого элемента и выделит предполагаемое место ошибки.

Итак, мы учли в модели ограничение срока службы товара, приводящее к повторным покупкам товара потребителями.

5. Запустите модель. Вы можете заметить изменение динамики продаж продукта: по прошествии определенного времени рост насыщения рынка сменится локальным спадом.

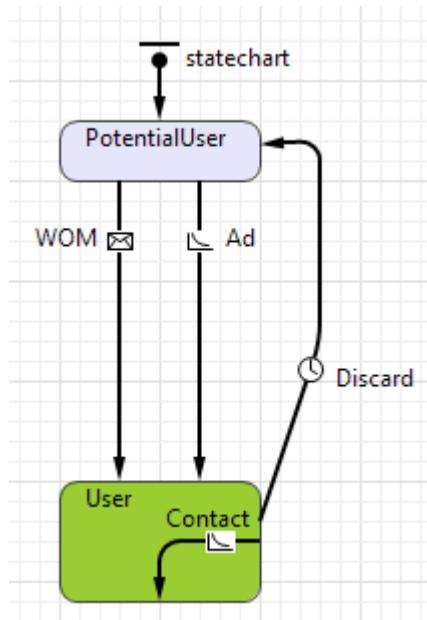
## 74 AnyLogic 8 за три дня



## Фаза 6. Учет времени доставки продукта

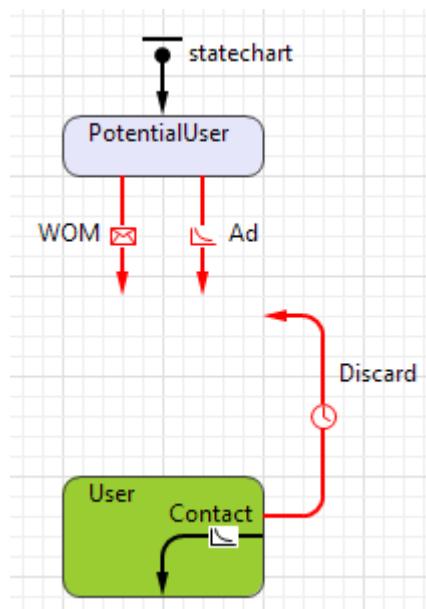
В нашей текущей модели предполагается, что продукт всегда есть в наличии, и поэтому переход из состояния *PotentialUser* в состояние *User* происходит моментально. Теперь мы усовершенствуем модель, добавив у потребителя еще одно состояние, которое будет соответствовать времени, проходящему с момента принятия решения о покупке продукта до момента появления товара в продаже и доставки его покупателю.

1. Подготовьте место для нового состояния между состояниями *PotentialUser* и *User*, перетащив состояние *User* вниз.

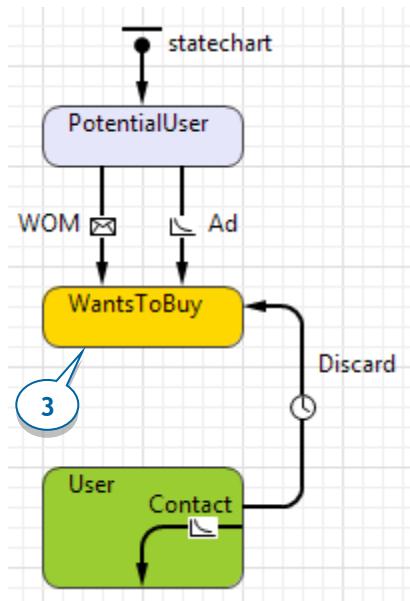


2. Отсоедините состояние *User* от переходов.

Выделите переходы *WOM* и *Ad* и переместите их конечные точки выше, затем отсоедините переход *Discard* от состояния *PotentialUser*. Вы заметите, что теперь эти переходы отображаются красным цветом.



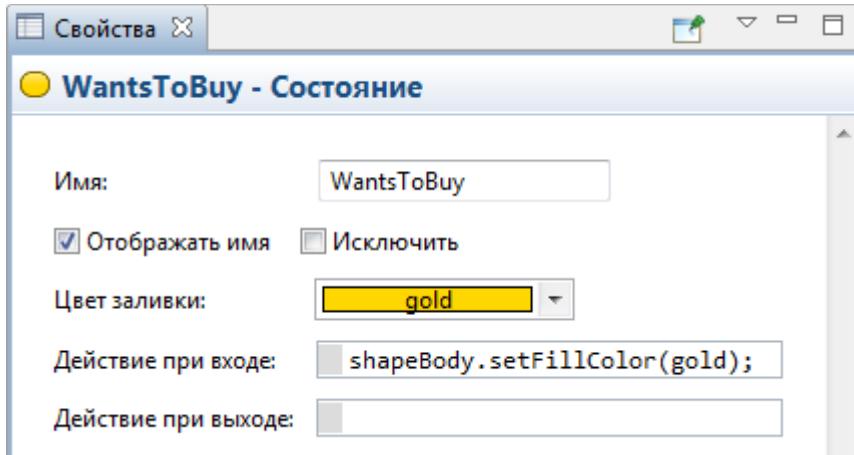
3. Добавьте новое Состояние из палитры Диаграмма состояний в середину диаграммы состояний потребителя и назовите его *WantsToBuy* («хочет купить»). Потребители в этом состоянии решили купить продукт, но продукт пока еще не приобрели.



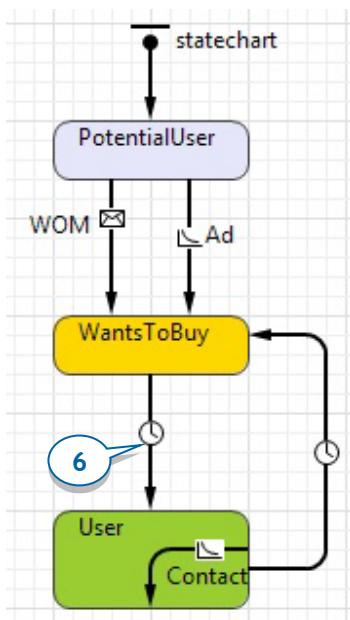
4. Подсоедините переходы *WOM*, *Ad*, и *Discard* к среднему состоянию *WantsToBuy*.
5. Измените свойства состояния *WantsToBuy*:

**Цвет заливки:** *gold*

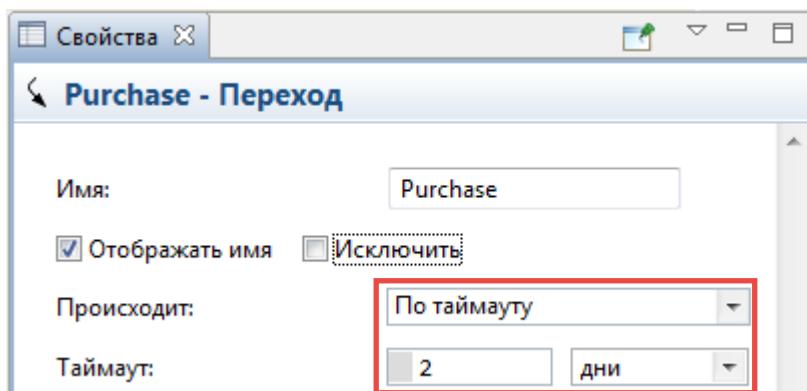
**Действие при входе:** *shapeBody.setFillColor(gold);*



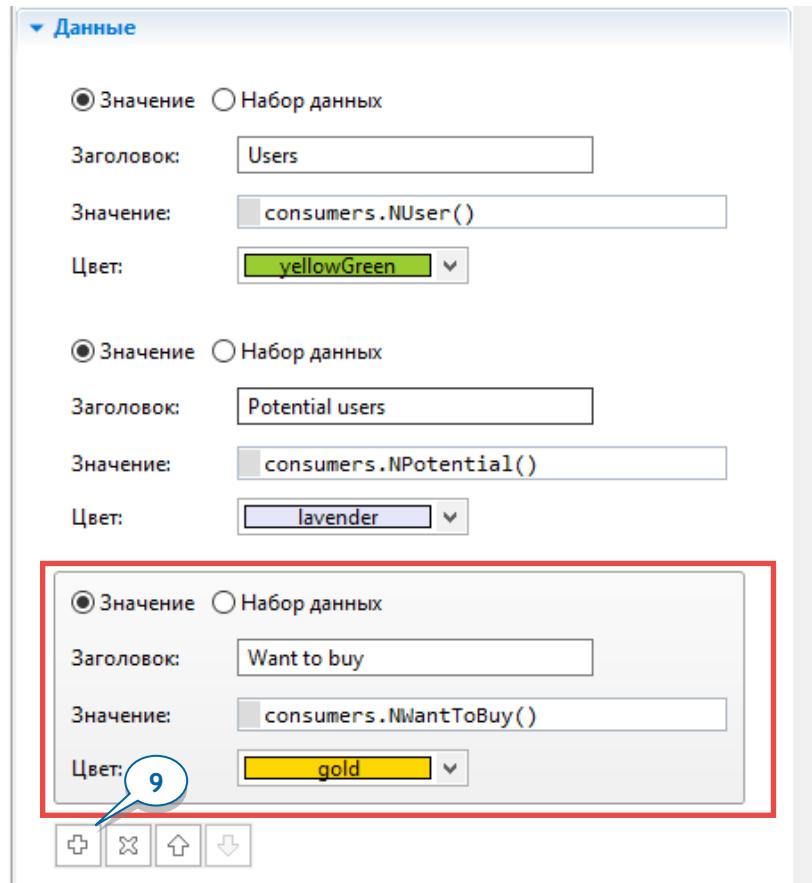
6. Добавьте переход из состояния *WantsToBuy* в состояние *User*, чтобы смоделировать доставку и, соответственно, покупку товара. Назовите этот переход *Purchase*.



7. Давайте предположим, что в среднем доставка продукта занимает два дня. Это означает, что наш агент перейдет в состояние *User* через два дня после момента перехода в состояние *WantsToBuy*. Исходя из этого, задайте для перехода *Purchase* таймаут длительностью в 2 дня:

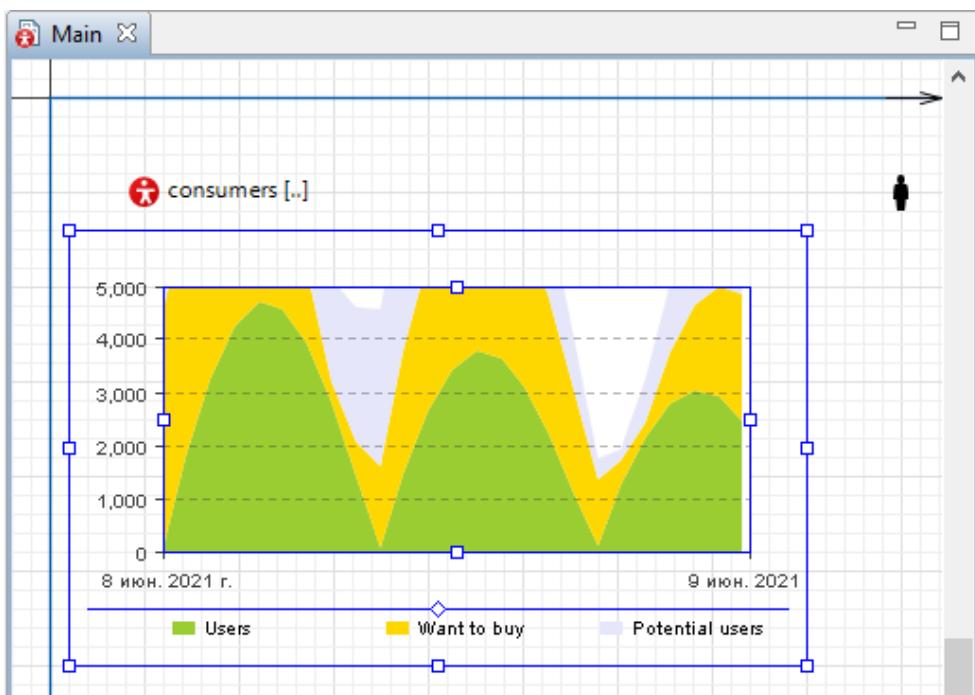
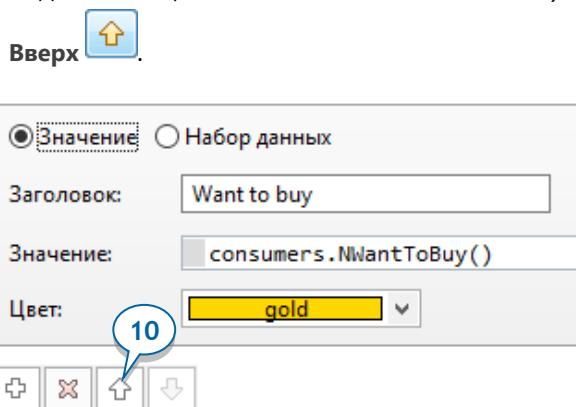


8. Создайте еще одну функцию сбора статистики, чтобы вести учет заявок на приобретение товара. Выделите популяцию *consumers* на диаграмме *Main*, перейдите в секцию свойств **Статистика** и добавьте новую функцию статистики с именем *NWantToBuy* и условием *item.inState(Consumer.WantsToBuy)*
9. Далее, на диаграмме *Main*, выделите наш график и добавьте еще один элемент данных для отображения, со значением *consumers.NWantToBuy()*, заголовком *Want to buy* и цветом *gold*.

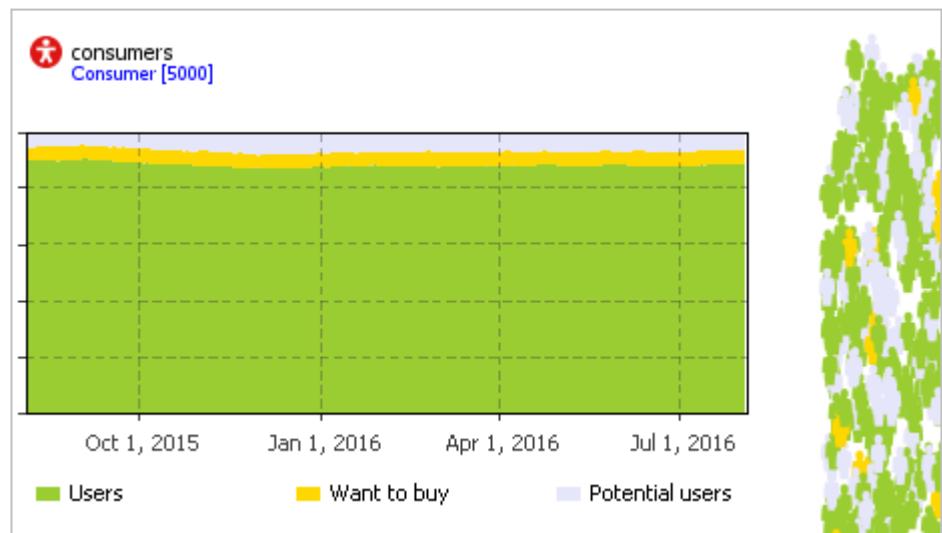


10. Переместите новый элемент данных в середину списка, чтобы во время моделирования график отображал категории на графике в следующем порядке: внизу - пользователи, затем - те, кто хочет прибрести продукт, и верхняя категория - потенциальные потребители продукта. Для этого

выделите секцию свойств элемента *Want to buy* и затем щелкните по кнопке



11. Запустите модель. Люди, ожидающие доставки товара, будут отображаться на графике и на анимации желтым цветом.

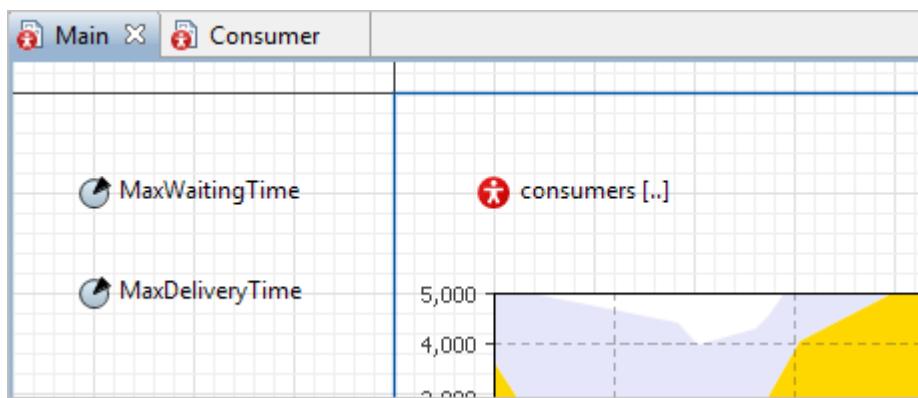


## Фаза 7. Моделирование отказов от покупки товара

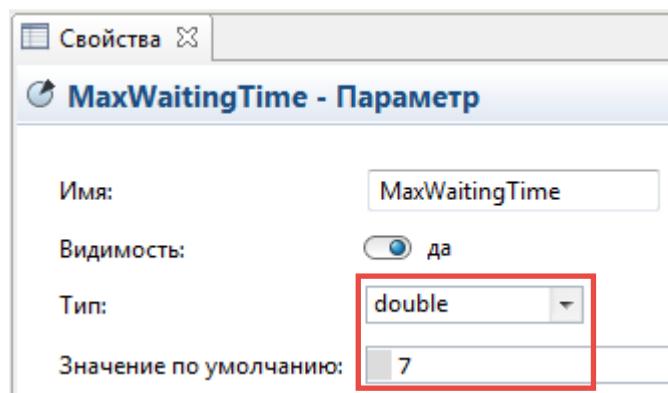
Теперь давайте учтем тот факт, что время, которое потребители согласны потратить на ожидание доставки товара, конечно. Если время доставки превысит предельно допустимое время ожидания, потребитель откажется от покупки.

Давайте начнем с того, что добавим на диаграмму *Main* два параметра, задающих максимальное время доставки товара (25 дней) и максимальное время ожидания доставки (7 дней) соответственно.

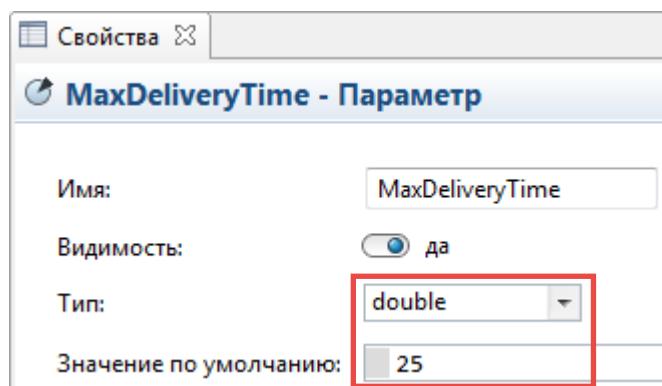
1. Откройте диаграмму *Main*.
2. Передвиньте холст графического редактора вправо, чтобы мы могли расположить элементы за пределами видимой области окна модели.
  - ◆ Чтобы передвинуть холст графического редактора, нажмите правую кнопку мыши в редакторе и перемещайте мышь, не отпуская кнопку.
  - ◆ Синяя прямоугольная рамка на диаграмме *Main* очерчивает границы окна модели. При запуске модели вы увидите те элементы, которые расположены внутри этой рамки.



3. Создайте два параметра. Параметр *MaxWaitingTime* задает максимальное время, в течение которого потребитель готов ждать доставки продукта (в нашем случае - семь дней).



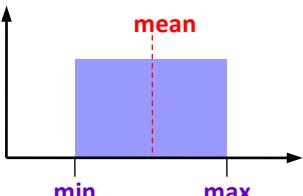
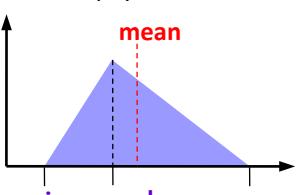
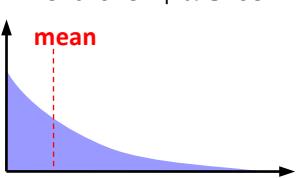
- Другой параметр, *MaxDeliveryTime*, задает максимально возможное время доставки товара. Поскольку мы должны учесть специфику работы определенных отечественных служб доставки, зададим значение этого параметра равным 25 дням.

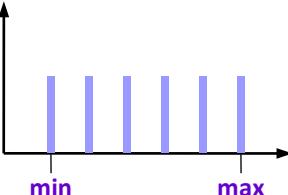


Таким образом, доставка товара может длиться от одного до 25 дней, в среднем же доставка занимает два дня. Давайте изменим значение времени доставки с фиксированного периода, равного двум дням, на стохастическое выражение, которое использует вышеуказанный диапазон значений.

## Функции распределения вероятностей

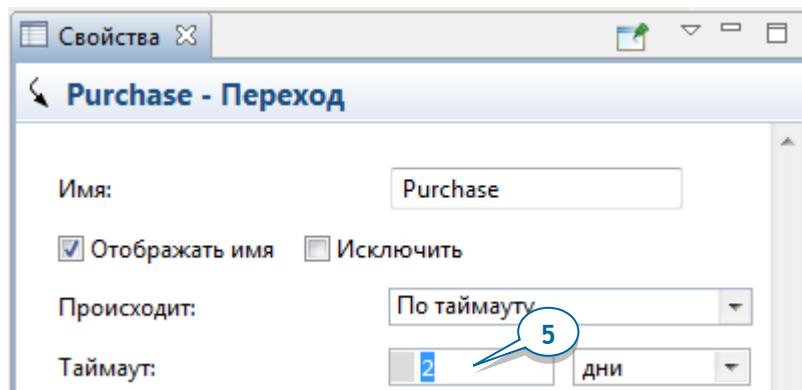
В следующей таблице мы рассмотрим самые часто используемые функции распределения вероятностей. Полный список функций, поддерживаемых AnyLogic, вы можете найти в документации, в разделе **Приложение. Java в AnyLogic > Функции AnyLogic**.

Функция распределения вероятностей	Описание
 <b>uniform( min, max )</b>	<p>Используется, когда вы знаете минимальное и максимальное значения, но не знаете, являются ли одни значения в этом интервале значений более частотными, чем другие. Поэтому, выбирая данное распределение вероятности, вы просто полагаете, что величина принимает любое значение из заданного интервала с равной вероятностью.</p>
 <b>triangular( min, mode, max )</b>	<p>Используется в том случае, когда у вас недостаточно измерений для построения эмпирического распределения, но вы знаете минимальное, максимальное и наиболее часто встречаемое (модальное) значение.</p> <p>Треугольное распределение обычно используется для задания времени обслуживания или длительности операции.</p>
 <b>exponential( lambda, min )</b>	<p>Описывает время между происхождением событий как Пуассоновский процесс, т.е. цепочку событий, происходящих независимо друг от друга с постоянной (в среднем) интенсивностью.</p> <p>Используется для определения времени между прибытиями посетителей, звонков, заказов, деталей в процессно-ориентированных моделях.</p>

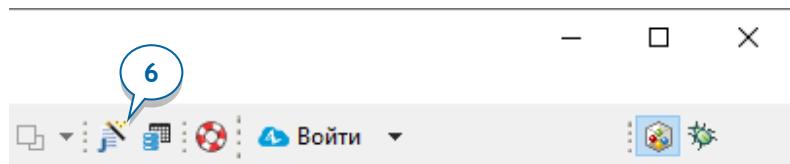
 <p><b>Дискретное равномерное</b></p> <p><b>uniform_discr( min, max )</b></p>	<p>Используется для моделирования конечного числа результатов, происходящих с равной вероятностью.</p> <p>В интервал возможных значений включаются и минимальное, и максимальное значения, поэтому вызов функции <code>uniform_discr(3, 7)</code> может вернуть 3, 4, 5, 6 или 7. (Borshchev, 2013)</p>
--	---

Ознакомившись с информацией в приведенной выше таблице, можно прийти к выводу, что самым подходящим распределением вероятностей для задания времени ожидания является треугольное.

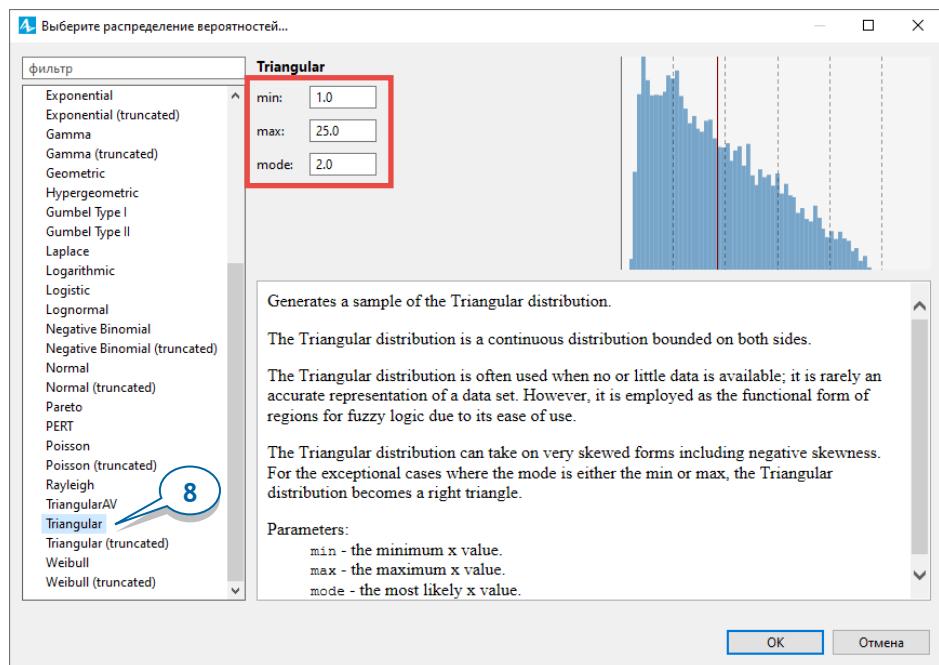
- Откройте диаграмму агента *Consumer* и выделите переход *Purchase*. Мы хотим изменить значение таймаута, по которому срабатывает переход. Для этого мы воспользуемся мастером выбора функций распределения, который вставит вызов функции в поле свойства перехода. Чтобы заменить текущее значение поля **Таймаут**, выделите его мышью.



6. Щелкните по кнопке панели управления Выберите распределение вероятностей...



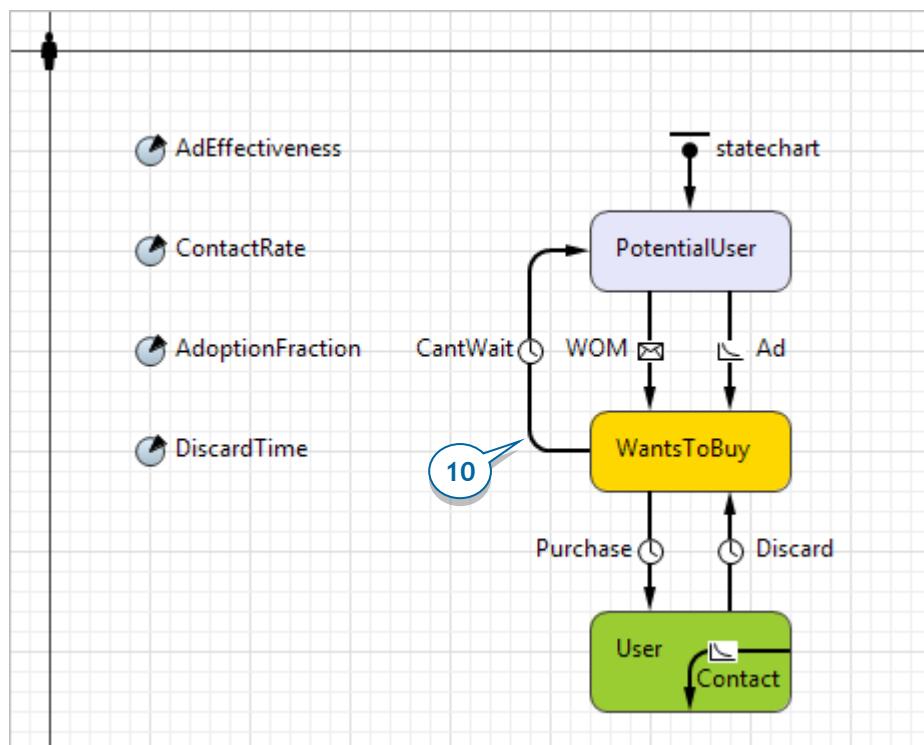
7. Откроется окно Мастера выбора распределения вероятностей.



8. Диалоговое окно Выберите распределение вероятностей позволяет выбрать одну из поддерживаемых AnyLogic функций распределения вероятностей. Выберите функцию *Triangular* в расположеннном слева списке. Введите в поля параметров **min**, **max** и **mode** значения 1, 25 и 2 соответственно. В правом верхнем углу вы увидите автоматически построенную гистограмму значений, сгенерированных функцией с заданными параметрами. Щелкните по кнопке **OK**, чтобы вставить вызов функции в кодовое поле.
9. В поле задания значения таймаута будет автоматически вставлено выражение *triangular(1, 25, 2)*. Давайте изменим эту строку на выражение *triangular(1, main.MaxDeliveryTime, 2)*

Здесь мы используем префикс *main*, чтобы получить доступ к агенту *Main* из агента *Consumer*.

10. Нарисуйте переход под названием *CantWait*, который выходит из состояния *WantsToBuy* и ведет в состояние *PotentialUser*. Этот переход моделирует то, как потребитель отказывается от покупки товара ввиду его долгого отсутствия. В результате диаграмма состояний агента *Consumer* будет выглядеть так:



11. Измените свойства перехода, задав его Таймаут равным *triangularAV(main.MaxWaitingTime, 0.15)* дней.

CantWait - Переход	
Имя:	<input type="text" value="CantWait"/> <input checked="" type="checkbox"/> Отображать имя <input type="checkbox"/> Исключить
Происходит:	<input type="text" value="По таймауту"/>
Таймаут:	<input type="text" value="triangularAV(main.MaxWaitingTime, 0.15)"/> <input type="text" value="дни"/>

Мы задаем максимальное время ожидания с помощью треугольного распределения со средним значением, равным параметру *MaxWaitingTime* (т.е., одной неделе), и отклонением от этого значения, равным 15 процентам.

Мы используем параметр, а не просто указываем соответствующее значение времени для того, чтобы впоследствии иметь возможность варьировать это значение динамически и наблюдать эффект от производимых изменений прямо по ходу моделирования. Одним из способов создания интерактивной модели является добавление элементов управления и связывание их с варьируемыми параметрами.

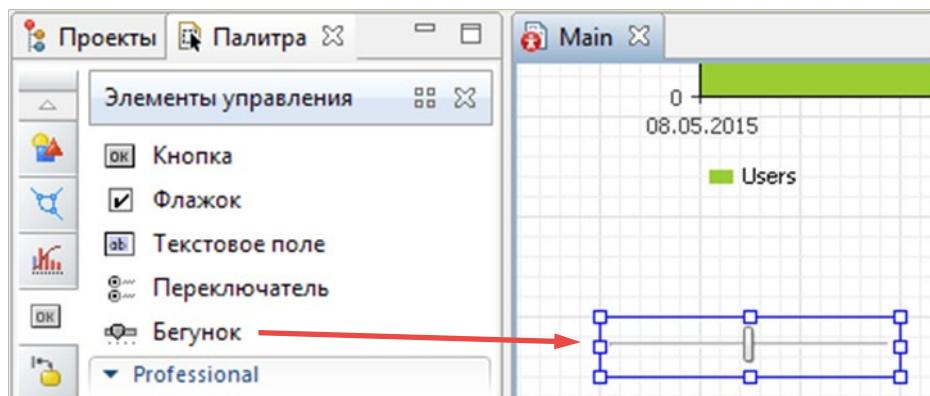
## Элементы управления

Модели можно сделать интерактивными, добавив в интерфейс модели различные элементы управления (кнопки, бегунки, текстовые поля и т.д.). Элементы управления могут использоваться как для задания значений параметров перед началом выполнения модели, так и для изменения модели прямо по ходу ее выполнения.

У элементов управления, имеющих состояние или содержимое (таких, как бегунок, переключатель, текстовое поле и т.д.), есть текущее значение, и они могут быть связаны с переменными и параметрами, так что, когда пользователь изменяет состояние такого элемента управления, изменяется и значение связанного с ним элемента (но не наоборот). Кроме того, вы можете задать для элемента управления определенное действие, например: вызов функции, планирование события, передачу сообщения, остановку модели и т.д. Действие будет выполняться каждый раз, когда пользователь меняет состояние элемента управления. Значение элемента управления обычно доступно в коде его поля **Действие как value**, а также возвращается функцией элемента *getValue()*.

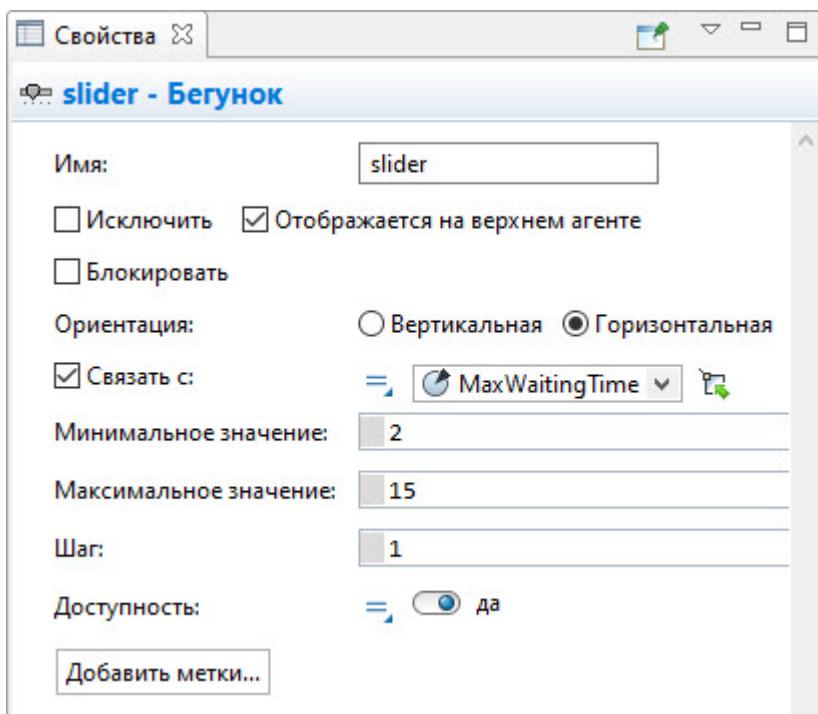
Мы добавим бегунок - элемент управления, который позволяет выбирать числовое значение из определенного интервала. Бегунок часто используется для того, чтобы изменять значения численных переменных и параметров.

12. Вернитесь на диаграмму *Main*. Откройте палитру  **Элементы управления**, перетащите элемент **Бегунок**  на диаграмму и расположите его под графиком. Сейчас мы свяжем этот бегунок с одним из наших параметров.

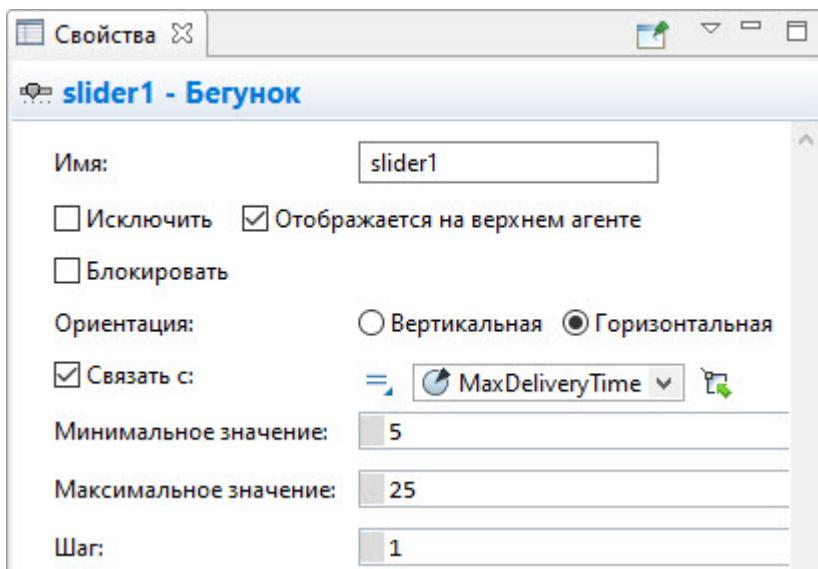


**13.** Измените свойства бегунка:

- Установите флажок **Связать с** и выберите параметр *MaxWaitingTime* из расположенного справа списка.
- Задайте минимальное и максимальное значения бегунка. Вы сможете варьировать значение параметра в заданном интервале значений. Введите 2 в поле **Минимальное значение** и 15 в поле **Максимальное значение**.
- Чтобы при работе с бегунком выбирайались только целые числа, введите 1 в поле **Шаг**.
- Затем щелкните по кнопке **Добавить метки...**, чтобы отображать эти значения бегунка во время моделирования (при этом под бегунком появятся текстовые метки *min*, *value* и *max*).

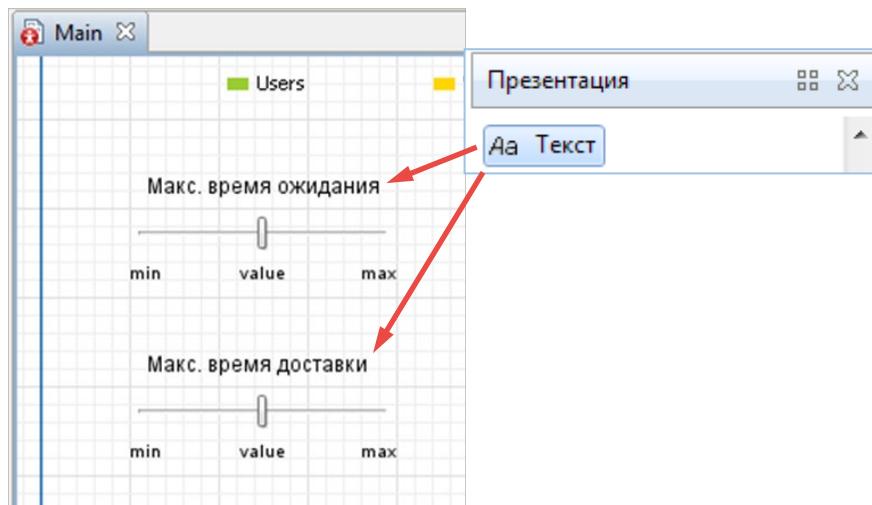


14. Добавьте еще один бегунок под предыдущим и настройте его следующим образом:

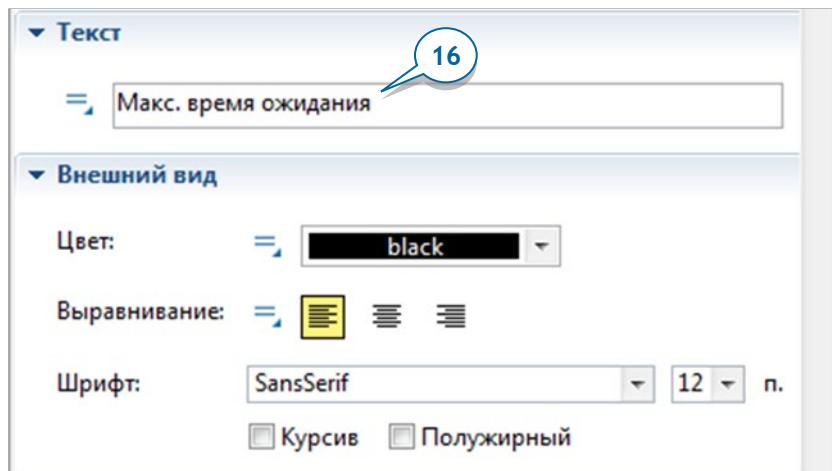


Давайте добавим подписи для каждого созданного бегунка. Для этого воспользуемся фигурой презентации **Текст Аа**.

- 15.** Откройте палитру **Презентация**, перетащите две фигуры **Текст Аа** на диаграмму и расположите их над бегунками.



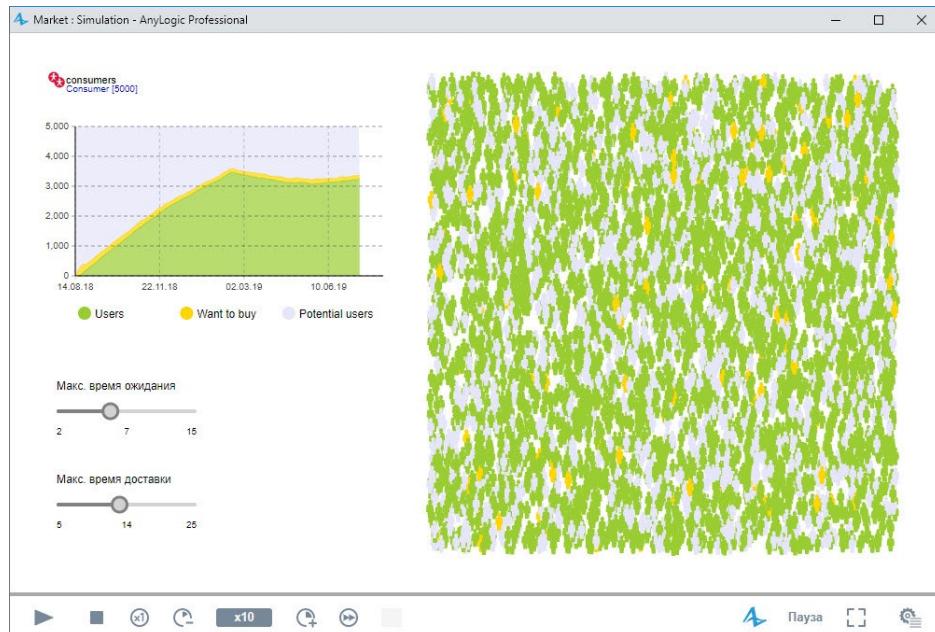
- 16.** В свойствах текстовых меток, в секции **Текст**, задайте текст, который вы хотите отображать с помощью этих меток. Для верхней метки введите *Макс. время ожидания*, для нижней - *Макс. время доставки*.



В секции свойств **Внешний вид** вы можете отформатировать текст, изменив его цвет, выравнивание, шрифт и размер.

Подписи под бегунками также являются фигурами Текст Аа. Вы можете редактировать их, как любую другую текстовую фигуру AnyLogic.

**17.** Запустите модель и понаблюдайте за ее поведением. Изменяя максимальное время ожидания и максимальное время доставки, вы можете оценить влияние этих изменений на поведение потребителей и состояние рынка.



## Фаза 8. Сравнение прогонов модели

Теперь давайте сравним поведение модели при различных начальных условиях. Мы могли бы вручную изменять значения параметров, запускать модель и сохранять результаты моделирования в специализированном инструменте для их сравнения, но намного проще будет воспользоваться встроенным экспериментом сравнения "прогонов" AnyLogic.

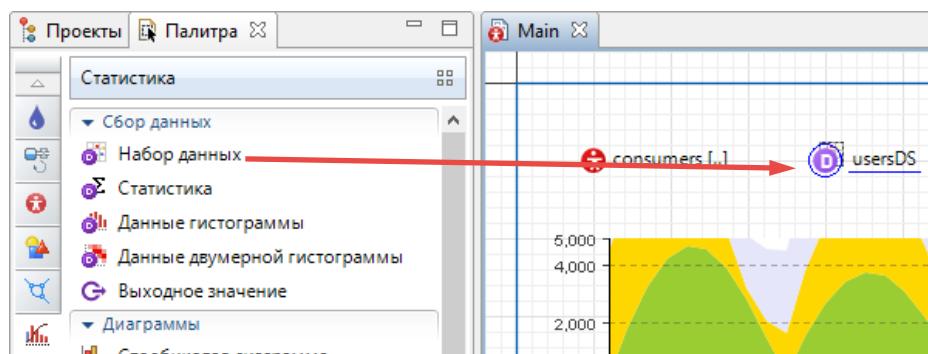
### Эксперимент сравнения «прогонов»

Этот интерактивный эксперимент позволяет менять значения параметров модели, заново запускать модель и сравнивать результаты нескольких прогонов, полученных при различных значениях параметров, на одном графике.

Стандартный пользовательский интерфейс эксперимента включает в себя элементы управления для ввода значений параметров и график(и) для отображения полученных результатов. Тип каждого элемента управления зависит от настроек в секции свойств **Редактор значения** управляемого этим элементом параметра.

Мы создадим эксперимент, который позволит вручную изменять значение параметра *ContactRate* и сравнивать поведение модели, наблюдаемое при различных значениях этого параметра.

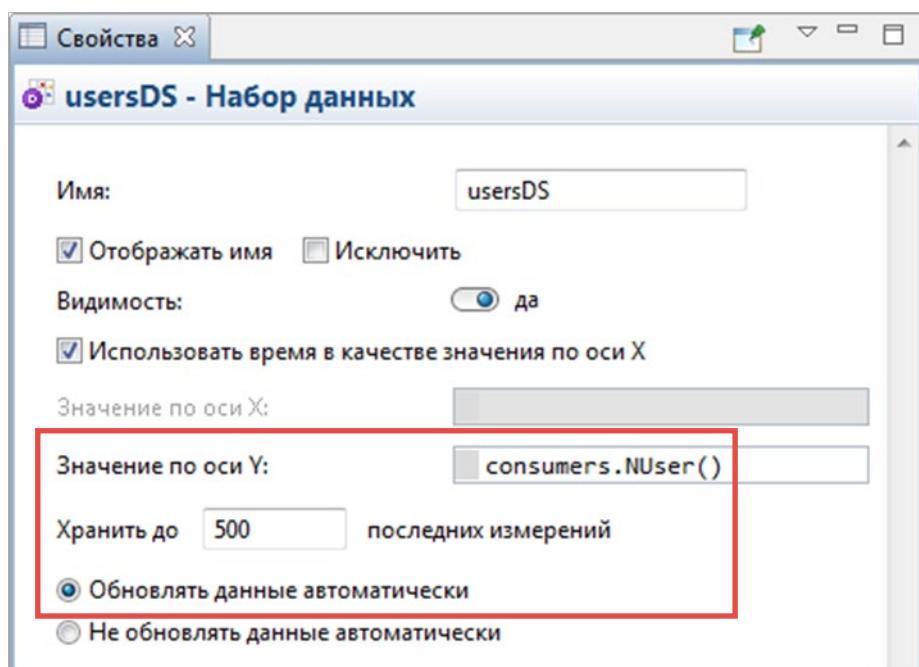
1. Откройте диаграмму *Main*. Добавьте на диаграмму **Набор данных**  из палитры **Статистика** . Назовите этот набор данных *usersDS*.



**Набор данных** может хранить пары значений (X,Y). Мы хотим, чтобы этот набор данных хранил историю динамики продаж продукта. С определенной периодичностью в набор данных будут записываться текущее значение

модельного времени и соответствующее ему количество потребителей продукта.

2. Чтобы записывать в набор данных временные метки, оставьте выбранной опцию **Использовать время в качестве значения по оси X** в свойствах этого набора данных.
3. Теперь укажите, какое значение будет запоминаться помимо временной метки. Введите в поле **Значение по оси Y**: `consumers.NUser()`.
4. Набор данных может хранить только ограниченное количество значений. Мы ограничим нашу выборку последними 500 значениями. Укажите, что данный набор данных будет **Хранить до 500 последних измерений**. Выберите опцию **Обновлять данные автоматически с Периодом обновления: 1 день**. Мы добавляем по одному значению на каждый моделируемый день.



Теперь у нас есть набор данных, который будет хранить динамику изменений количества потребителей продукта. Каждое новое значение получается путем вызова функции статистики `NUser()`, созданной нами ранее в популяции агентов `consumers`.

5. Теперь давайте изменим свойства в секции **Редактор значения** для обоих параметров на диаграмме *Main* (параметры *MaxWaitingTime* и *MaxDeliveryTime*). Выберите *Бегунок* как **Тип управления**, укажите те же **мин** и **макс** значения, что мы задавали для бегунков на диаграмме *Main*. При желании можете также изменить текстовые метки (например, на *Макс. время ожидания* и *Макс. время доставки*).

**▼ Редактор значения**

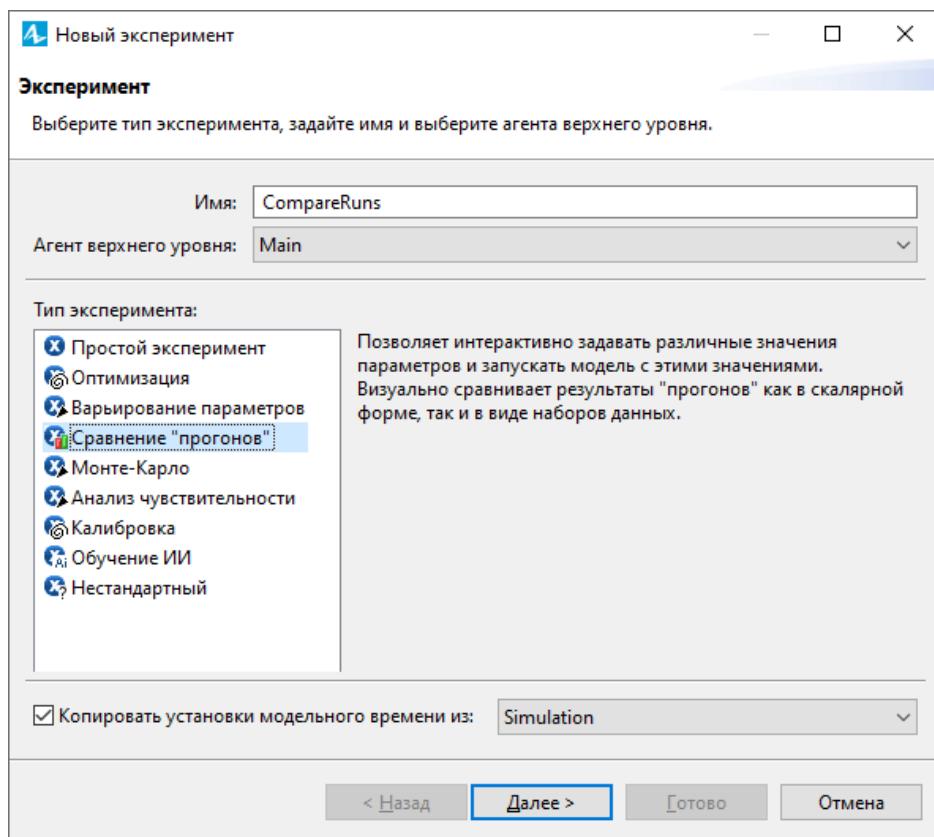
Метка:	Макс. время ожидания
Тип управления:	Бегунок
мин.:	2
макс.:	15

**▼ Редактор значения**

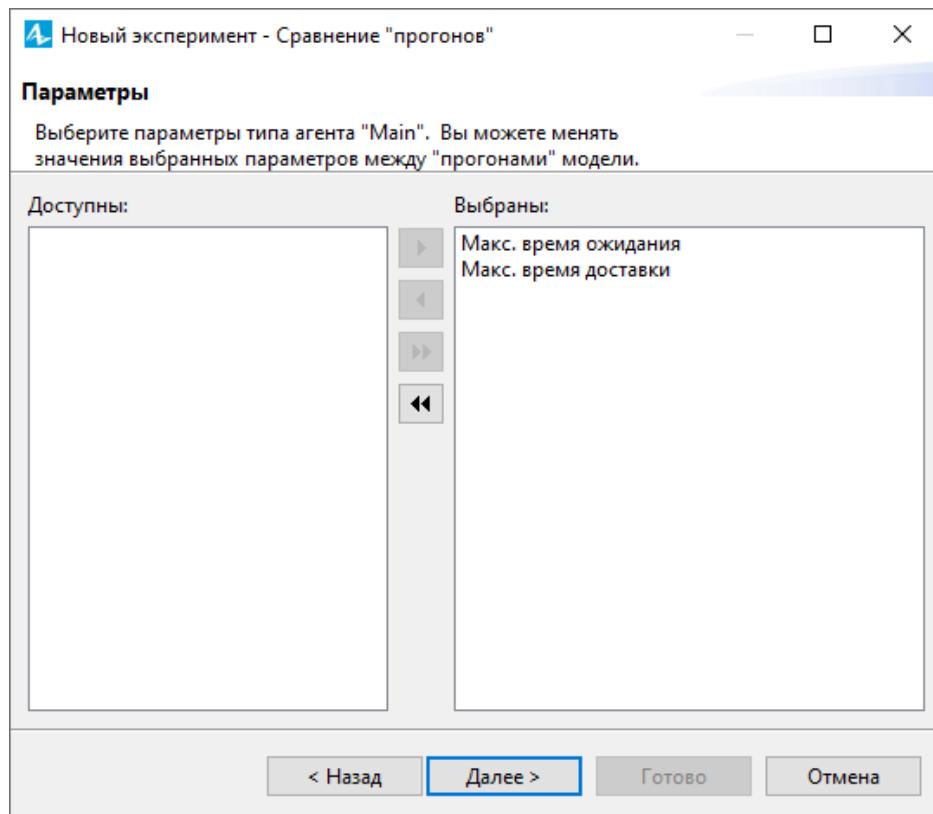
Метка:	Макс. время доставки
Тип управления:	Бегунок
мин.:	5
макс.:	25

Теперь мы готовы создать эксперимент сравнения прогонов.

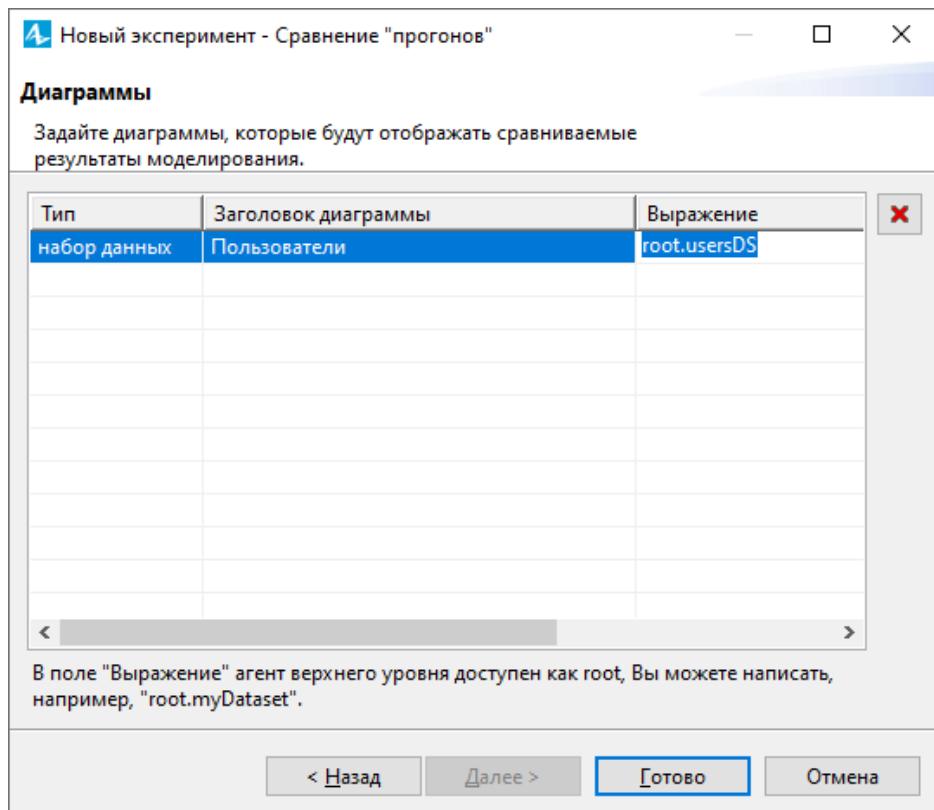
6. Откройте панель **Проекты**, щелкните в дереве правой кнопкой мыши по элементу модели **Market** и выберите **Создать > Эксперимент** из контекстного меню. При этом откроется диалоговое окно **Новый эксперимент**.
7. В списке **Тип эксперимента**, выберите **Сравнение «прогонов»**  . Щелкните по кнопке **Далее**.



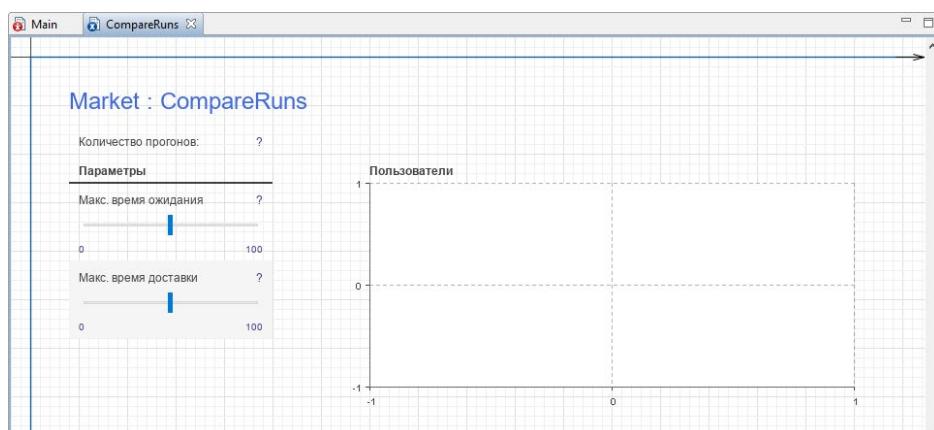
- На странице **Параметры**, добавьте оба параметра в колонку **Выбраны**. Чтобы добавить параметр, выделите его в левой колонке **Доступны** и щелкните по кнопке . Или же вы можете просто щелкнуть по кнопке , чтобы сразу выбрать все параметры. Щелкните по кнопке **Далее**, когда оба параметра будут добавлены в колонку **Выбраны**.



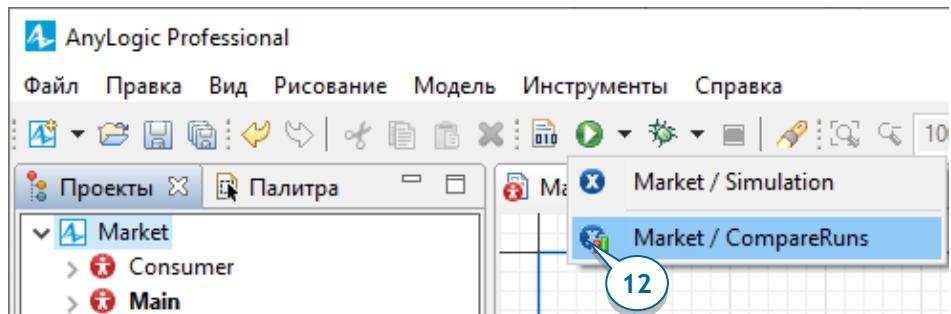
9. На следующей странице мастера вы можете задать, какие графики вы хотите добавить для отображения результатов эксперимента. Нам будет достаточно одного графика. Задайте его свойства, введя следующие данные в таблицу **Диаграммы**:
  - a. В колонке **Тип**, выберите опцию **набор данных**.
  - b. В колонке **Заголовок диаграммы**, введите *Пользователи*.
  - c. В колонке **Выражение**, укажите тот набор данных, который мы задали на диаграмме *Main*. Для этого введите *root.usersDS*. Имя *root* используется здесь для доступа к агенту верхнего уровня модели (в нашем случае это агент *Main*). Этот график будет отображать данные, собранные набором данных *usersDS*.
10. Щелкните по кнопке **Готово**.



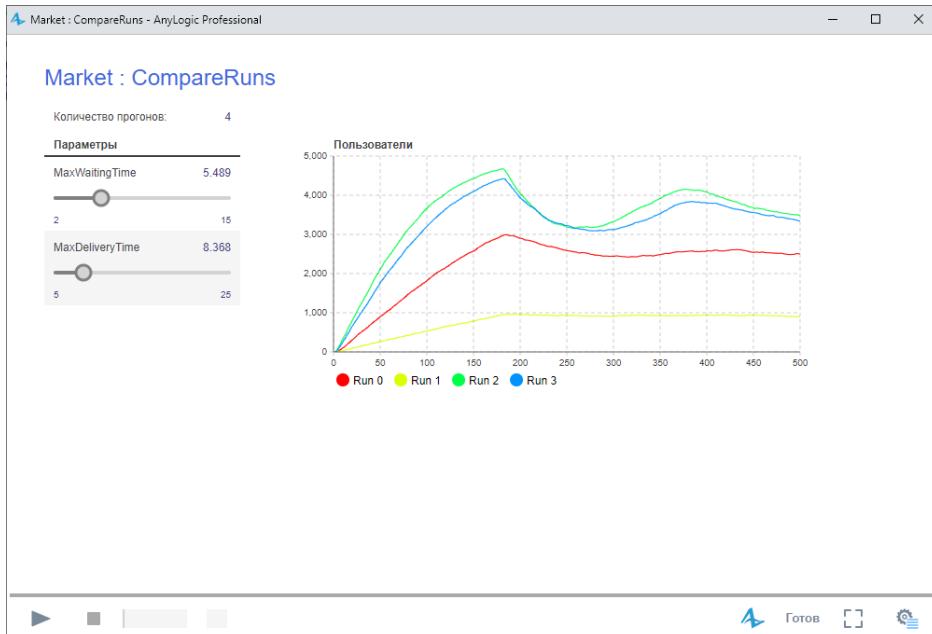
Автоматически откроется диаграмма эксперимента сравнения прогонов *CompareRuns*. Вы увидите на ней элементы стандартного интерфейса данного типа эксперимента.



- 11.** Мы хотим, чтобы наш эксперимент исследовал данные, собранные за период, превышающий один год. Пусть наш эксперимент моделирует 500 дней. Чтобы ограничить время выполнения эксперимента, выделите эксперимент *CompareRuns* в панели **Проекты**, затем раскройте секцию свойств эксперимента **Модельное время** и введите *500* в поле **Конечное время**.
- 12.** Запустите эксперимент. Выберите наш новый эксперимент из списка под кнопкой Запустить: Market / CompareRuns, или щелкните правой кнопкой мыши по эксперименту *CompareRuns* в дереве панели **Проекты** и выберите Запустить из контекстного меню.



- 13.** В окне модели, щелкните по кнопке **Запустить**. Сначала вы увидите результаты, полученные при заданных по умолчанию значениях параметров. Теперь измените значения параметров и снова щелкните по кнопке **Запустить**, чтобы изучить поведение системы при других значениях параметров.



- 14.** Каждая кривая на графике соответствует отдельному прогону модели. Вы можете выбрать одну из кривых, щелкнув по подписи с ее названием в легенде графика. Расположенные слева элементы управления покажут значения, которые использовались в этом прогоне и привели к данному результату.

Мы закончили создание агентной модели рынка. Вы можете усовершенствовать эту модель, усложнив логику принятия решений потребителями (например, добавив конкурирующие продукты). Похожую модель, которая называется *Statechart for Choice of Competing Products*, можно найти в секции примеров AnyLogic Модели из книги "Big Book of Simulation Modeling". Чтобы посмотреть список моделей, выберите пункт **Примеры моделей** из меню **Справка**.

## Системная динамика

*“Системная динамика – это подход к имитационному моделированию, своими методами и инструментами позволяющий понять структуру и динамику сложных систем. Также системная динамика – это метод моделирования, использующийся для создания точных компьютерных моделей сложных систем для дальнейшего их использования с целью проектирования более эффективной организации и политики взаимоотношений с данными системами. Вместе эти инструменты позволяют нам создавать микромирсымуляторы, где пространство и время могут быть сжаты и замедлены так, чтобы мы могли изучить последствия наших решений, быстро освоить методы и понять структуру сложных систем, спроектировать тактики и стратегии для большего успеха.”*

*Джон Штерман, “Бизнес-процессы: Системное мышление и моделирование сложного мира”*

Метод *системной динамики* был изобретен в 1950-х Джоем Форрестером из Массачусетского Технологического Института (MIT). Используя свой научный и инженерный опыт, Форрестер искал способ применения законов физики, в частности, законов электрических цепей, к исследованиям и описанию динамики процессов социальных и экономических систем.

Системная динамика чаще всего используется для разработки долгосрочных стратегических моделей и предполагает высокий уровень агрегации объектов: модели системной динамики рассматривают людей, товары, ресурсы и другие отдельные элементы в количественных терминах.

Системная динамика предоставляет методы изучения динамических систем. Предполагается, что вы:

- Моделируете систему как закрытую структуру, которая сама определяет собственное поведение.
- Обнаруживаете циклы обратной связи, уравновешивающего или усиливающего типа. Циклы обратной связи занимают центральное место в системной динамике.

- Задаете накопители и потоки, которые на них влияют.

Накопители характеризуют состояние системы. Они содержат память системы. Модель работает только с совокупностью объектов: отдельные элементы, содержащиеся в накопителе, не различимы. Потоки представляют интенсивность, с которой меняются эти состояния системы.

Если вам сложно разделить понятия потока и накопителя, представьте, что мы ими измеряем. Накопители обычно используются, чтобы обозначить совокупность людей, уровни запасов, денежные средства или знания, тогда как потоки измеряют количество чего-либо за период времени, например, количество клиентов в месяц или долларов в год.

Цель данной главы – научить вас разрабатывать модели системной динамики в AnyLogic. Если вы хотите получить больше информации о самом подходе моделирования, мы рекомендуем книгу Джона Штермана “Бизнес-процессы: Системное мышление и моделирование сложного мира”.

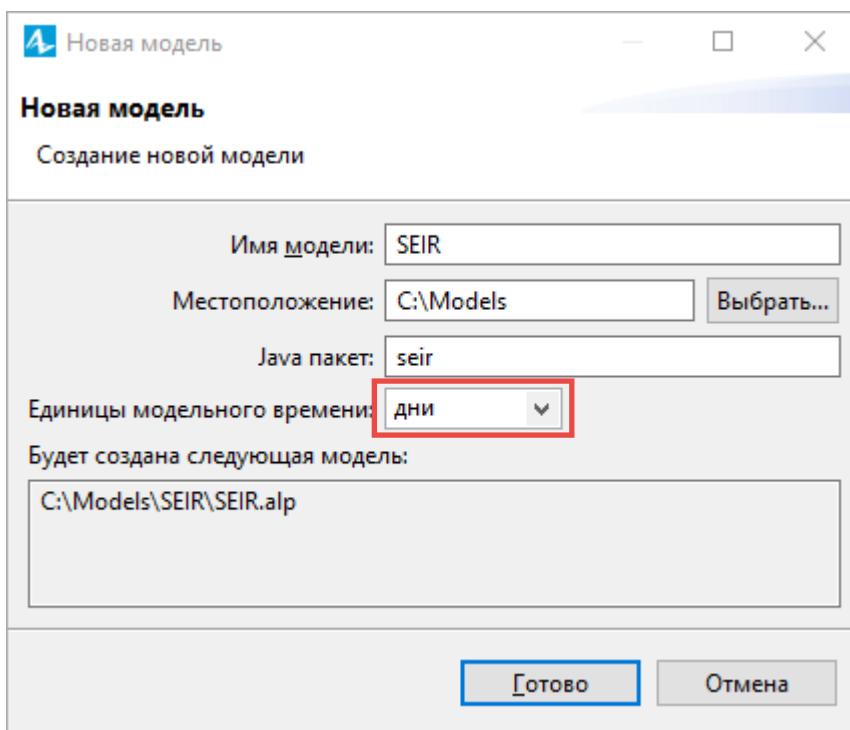
## Модель распространения эпидемии

Мы построим модель, изучающую распространение инфекционного заболевания среди населения. Давайте рассмотрим численность населения, равную 10 000 человек, которую обозначим как *TotalPopulation*. Вначале заражен только один человек, а все остальные лишь восприимчивы к болезни.

- Во время болезни один человек в среднем контактирует с другими с интенсивностью *ContactRateInfectious*, равной 1.25 человека в день. Если заразившийся человек контактирует с восприимчивым к болезни, то вероятность передачи инфекции *Infectivity* равняется 0.6.
- После того, как человек заражается, инкубационный период *AverageIncubationTime* длится 10 дней.
- Средняя длительность болезни после инкубационного периода *AverageIllnessDuration* (другими словами, длительность периода, когда этот человек может заражать других) составляет 15 дней.
- Выздоровевшие люди получают иммунитет к болезни и не могут снова заболеть.

## Фаза 1. Создание диаграммы потоков и накопителей

1. Создайте новую модель, выбрав пункт меню **Файл > Создать > Модель**. Назовите модель *SEIR* и выберите **дни** в качестве единиц модельного времени.



Давайте начнем с того, что нарисуем диаграмму накопителей и потоков.

В данной модели мы не будем учитывать все разнообразие населения, а лишь выделим четыре категории людей, имеющие значение для изучаемого нами процесса:

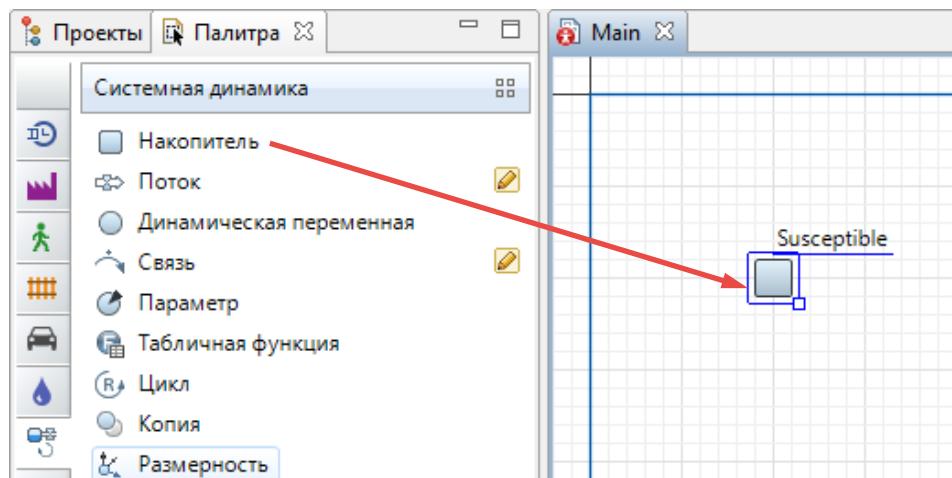
- *Susceptible* – Восприимчивые к заражению люди, которые еще не были заражены вирусом.
- *Exposed* – Люди, находящиеся в латентной стадии заражения (они уже заражены, но еще не могут заражать других).
- *Infectious* – Люди в активной стадии заражения (они могут заражать других людей).
- *Recovered* – Выздоровевшие люди (они приобрели иммунитет к данному заболеванию).

Название модели SEIR – это аббревиатура, образованная сокращением названий основных стадий распространения инфекции: Susceptible - Exposed - Infectious - Recovered.

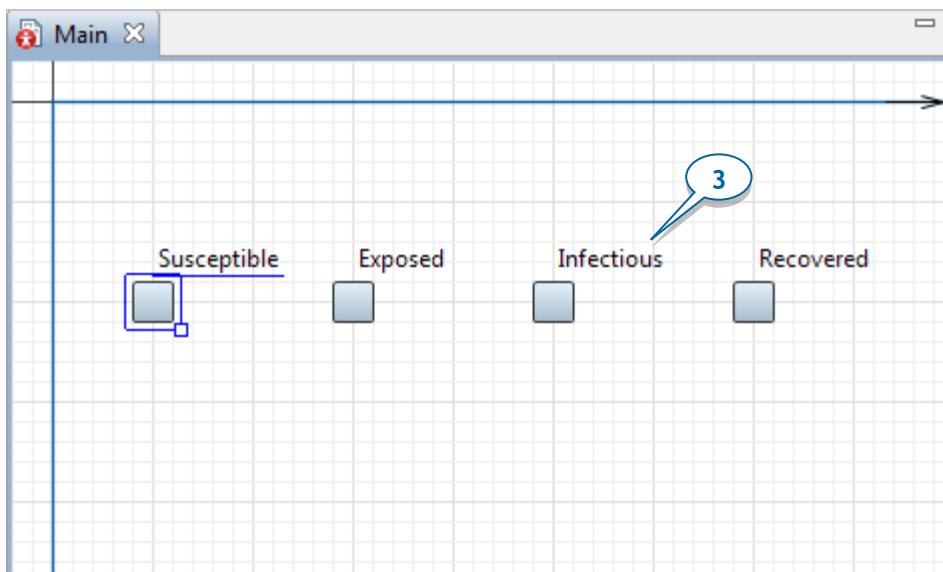
Терминология и общая структура модели взяты из книги ("Compartmental models in epidemiology").

В нашей модели можно естественным образом выделить четыре накопителя, по одному на каждую стадию заболевания. Давайте и начнем с их создания.

2. Откройте палитру **Системная Динамика**. Перетащите элемент **Накопитель**  из палитры **Системная динамика** на диаграмму *Main*. Назовите его *Susceptible*.



3. Добавьте еще три накопителя. Расположите их, как показано на рисунке ниже, и назовите *Exposed*, *Infectious* и *Recovered*.



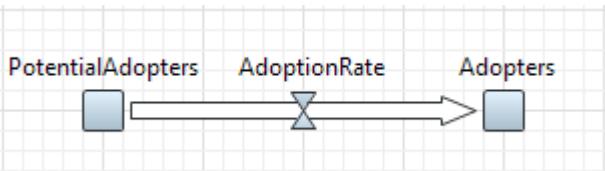
## Накопители и потоки

В системной динамике *накопители* (иногда они также называются *уровнями* или *фондами*) представляют собой переменные, которые эквивалентны объему определенного «вещества» (это могут быть деньги, знания, люди, жидкости и т.п.).

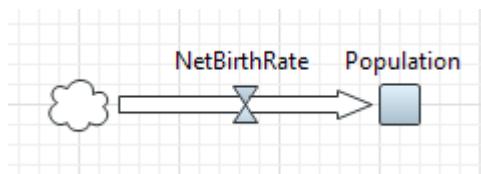
*Потоки* задают динамику системы. Значения накопителей изменяются с течением времени именно согласно существующим в системе потокам. Входящий в накопитель поток увеличивает значение данного накопителя, исходящий из накопителя поток уменьшает его значение. Ниже приведены примеры накопителей и потоков:

Накопитель	Входящие потоки	Исходящие потоки
<b>Население</b>	Рождаемость	Смертность
	Иммиграция	Эмиграция
<b>Бак с горючим</b>	Заправка	Потребление горючего

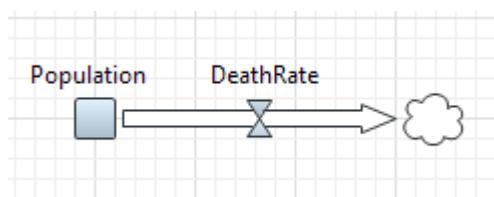
Один и тот же поток может служить исходящим потоком для одного накопителя и входящим - для другого; в этом случае говорится, что это поток из первого накопителя во второй:



Если поток начинает течение "из ниоткуда", то в его начальной точке рисуется общепринятый для таких случаев символ облака.



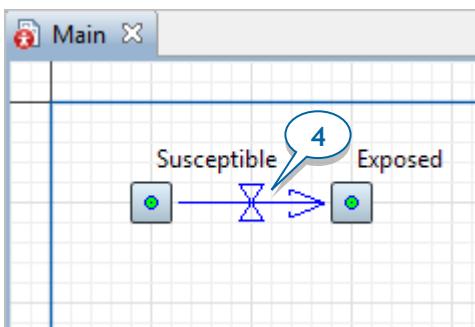
Аналогично, облако рисуется у конечной точки потока, если поток течет не в какой-то другой накопитель, а "в никуда".



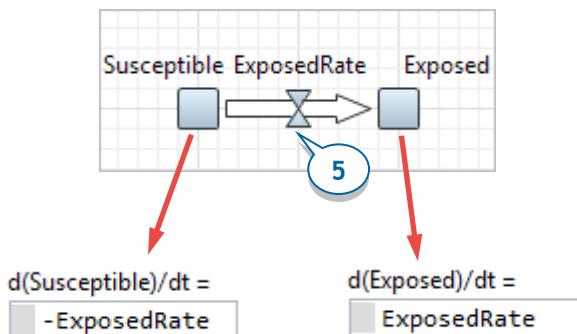
Направление стрелки потока обозначает влияние этого потока на накопители.

Основная логика нашей модели такова: восприимчивые к заболеванию люди подвергаются заражению вирусом, болеют и заражают других, а затем выздоравливают. Чтобы промоделировать перемещение людей между нашими четырьмя накопителями, нам нужно добавить три потока.

4. Добавьте первый поток, который ведет из накопителя *Susceptible* в накопитель *Exposed*. Сделайте двойной щелчок мышью по накопителю, из которого поток выходит (*Susceptible*) и затем щелкните по накопителю, в который поток входит (*Exposed*).



5. Назовите этот поток *ExposedRate*.



6. Обратите внимание на формулы накопителей *Susceptible* и *Exposed*. Из них следует, что поток *ExposedRate* уменьшает значение накопителя *Susceptible* и увеличивает значение накопителя *Exposed*.

### Формулы накопителей

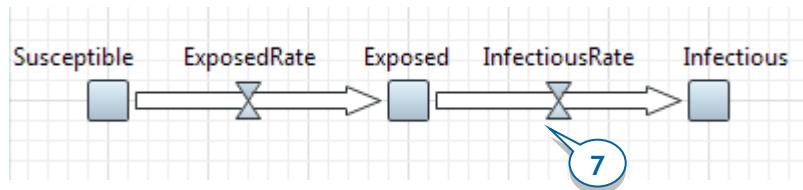
AnyLogic автоматически формирует формулу накопителя в соответствии с создаваемой пользователем диаграммой потоков и накопителей.

Значение накопителя вычисляется согласно потокам, входящим и исходящим из него. Значения входящих потоков, то есть тех, которые увеличивают значение накопителя, прибавляются к текущему значению накопителя, а значения исходящих потоков, соответственно, вычитаются из него:

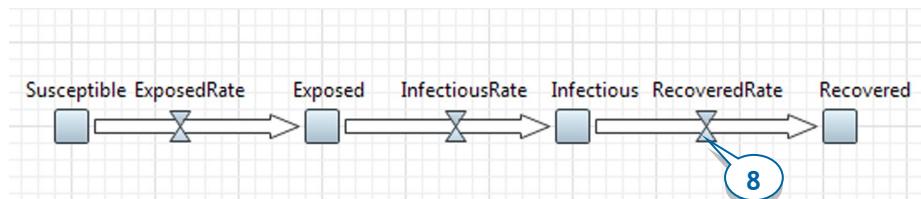
*входящийПоток1 + входящийПоток2 + ... - исходящийПоток1 - исходящийПоток2 ...*

В классическом режиме задания формулы накопителя формула является нередактируемой, и в ней могут фигурировать только потоки.

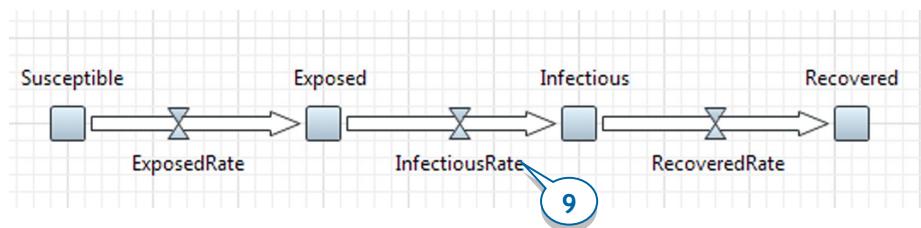
7. Добавьте поток, ведущий из накопителя *Exposed* в накопитель *Infectious*, и назовите его *InfectiousRate*.



8. Добавьте поток из накопителя *Infectious* в накопитель *Recovered* и назовите его *RecoveredRate*.



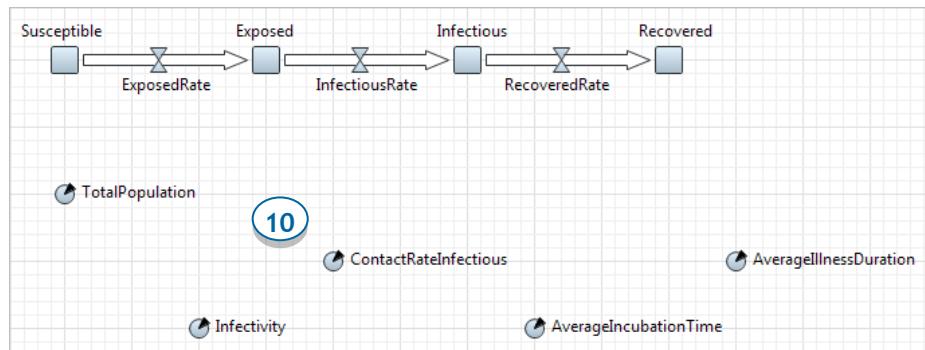
9. Расположите метки с именами потоков, как показано на рисунке ниже. Чтобы переместить метку, выделите поток в графическом редакторе и затем переместите его имя.



10. Теперь давайте зададим параметры и зависимости. Добавьте пять элементов **Параметр**, задайте их имена и значения по умолчанию, как указано ниже:

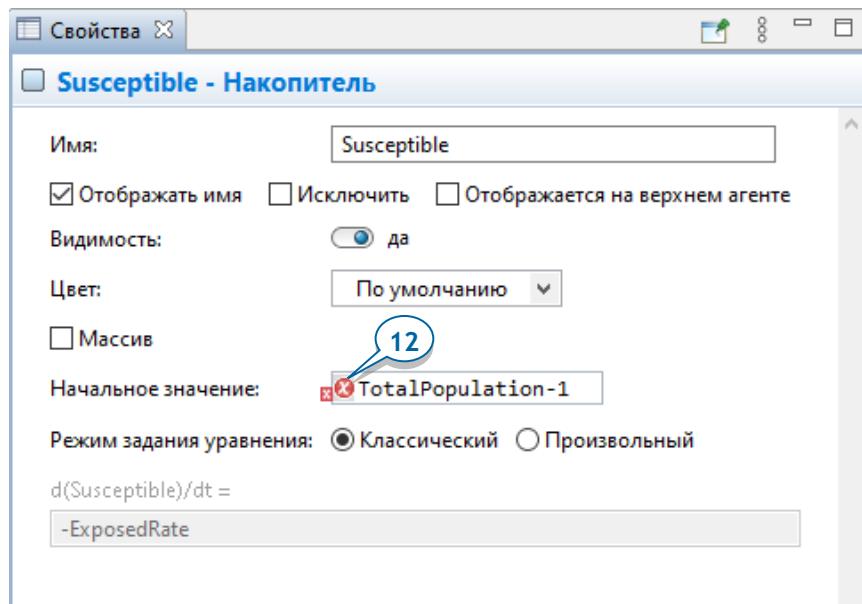
- *TotalPopulation* = 10 000
- *Infectivity* = 0.6
- *ContactRateInfectious* = 1.25
- *AverageIncubationTime* = 10
- *AverageIllnessDuration* = 15

## 110 AnyLogic 8 за три дня



11. Задайте первоначальное количество инфицированных людей, указав значение *1* в качестве **Начального значения** накопителя *Infectious*.
12. Задайте **Начальное значение** накопителя *Susceptible*: *TotalPopulation-1*.

Вы можете нажать Ctrl+пробел (Mac OS: Alt+пробел) и затем выбрать имя параметра из мастера подстановки кода.



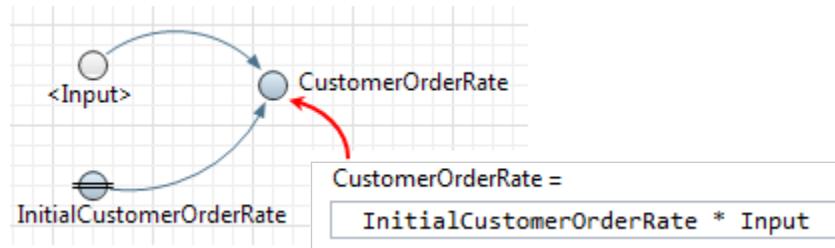
Слева от введенного выражения вы увидите красный значок — индикатор ошибки. Причина ошибки в том, что вы задали логическую зависимость между элементами диаграммы накопителей и потоков (начальное значение накопителя *Susceptible* зависит от параметра *TotalPopulation*), но эта зависимость не задана графически на диаграмме.

## Связи зависимостей

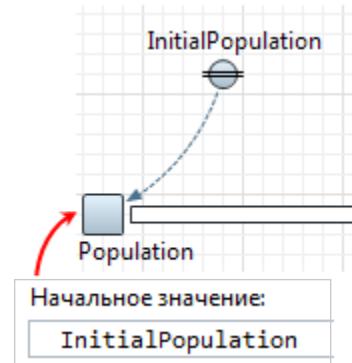
Связь используется для задания зависимости между элементами диаграммы потоков и накопителей.

Зависимости в диаграммах потоков и наполнителей могут быть двух типов:

- Переменная (это может быть накопитель, поток, вспомогательная переменная или параметр) упоминается в формуле потока или вспомогательной переменной. Такой тип связи отображается сплошной линией:

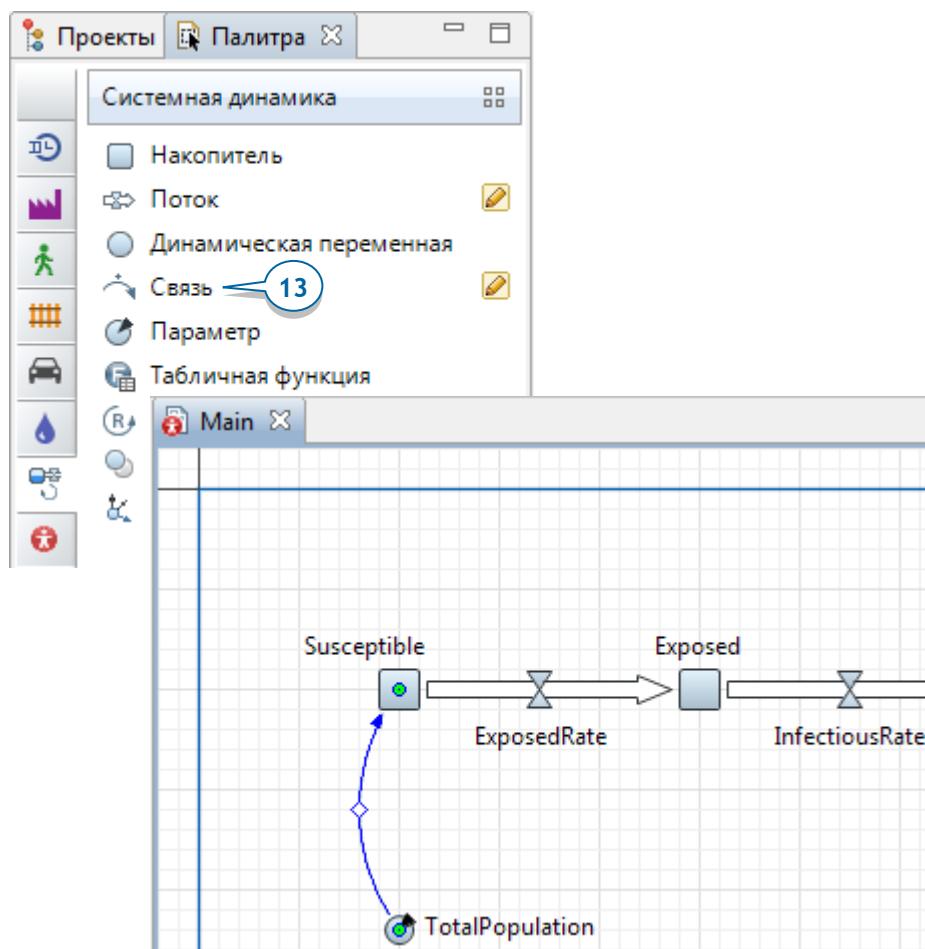


- Переменная фигурирует в формуле начального значения накопителя. Этот тип связи отображается пунктирной линией:

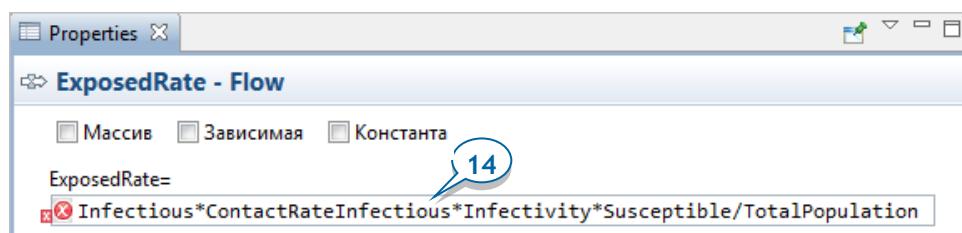


13. Нарисуйте связь, ведущую из параметра *TotalPopulation* в накопитель *Susceptible*:

Сделайте двойной щелчок мышью по элементу **Связь** палитры **Системная динамика**, щелкните по параметру *TotalPopulation* и затем щелкните по накопителю *Susceptible*. Вы увидите связь с точками соединения на ее концах:

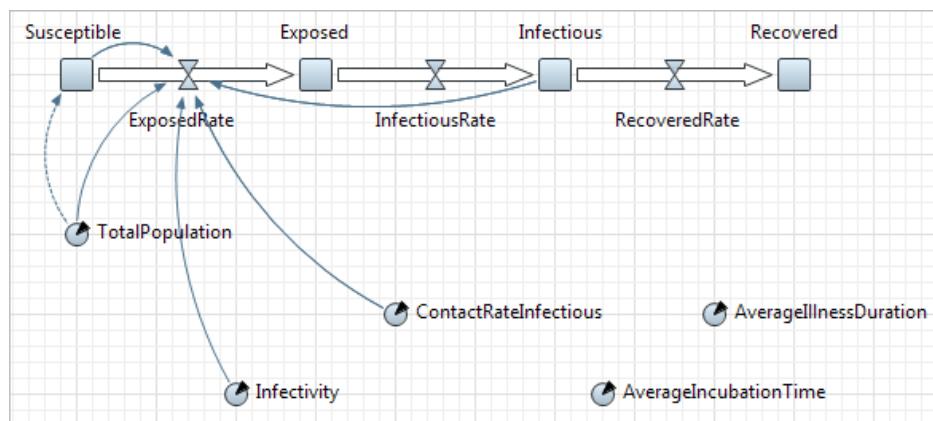


14. Давайте зададим формулу потока *ExposedRate*. Выделите поток щелчком мыши и введите следующую формулу с помощью мастера подстановки кода:
- $$\text{Infectious} * \text{ContactRate} * \text{Infectivity} * \text{Susceptible} / \text{TotalPopulation}$$

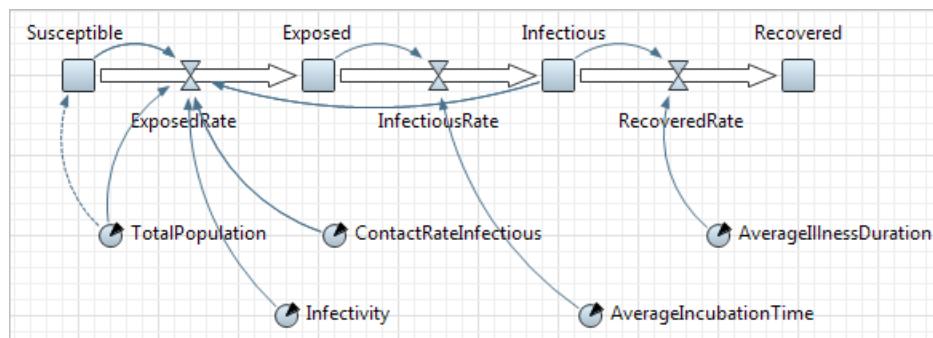


Нам необходимо нарисовать связи зависимостей, ведущие от указанных в формуле переменных и параметров к этому потоку. Может показаться забавным, но в некоторых других инструментах системной динамики все связи придется рисовать вручную. Мы же предпочитаем использовать механизм автоматического создания связей.

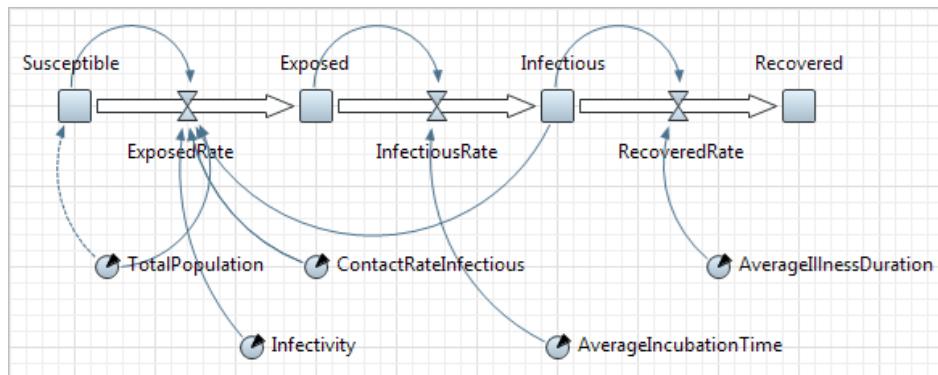
- 15.** Щелкните правой кнопкой мыши по потоку *ExposedRate* в графическом редакторе и выберите опцию **Исправить ошибки в связях > Создать недостающие связи** из контекстного меню. При этом появятся недостающие для этого потока связи зависимости:



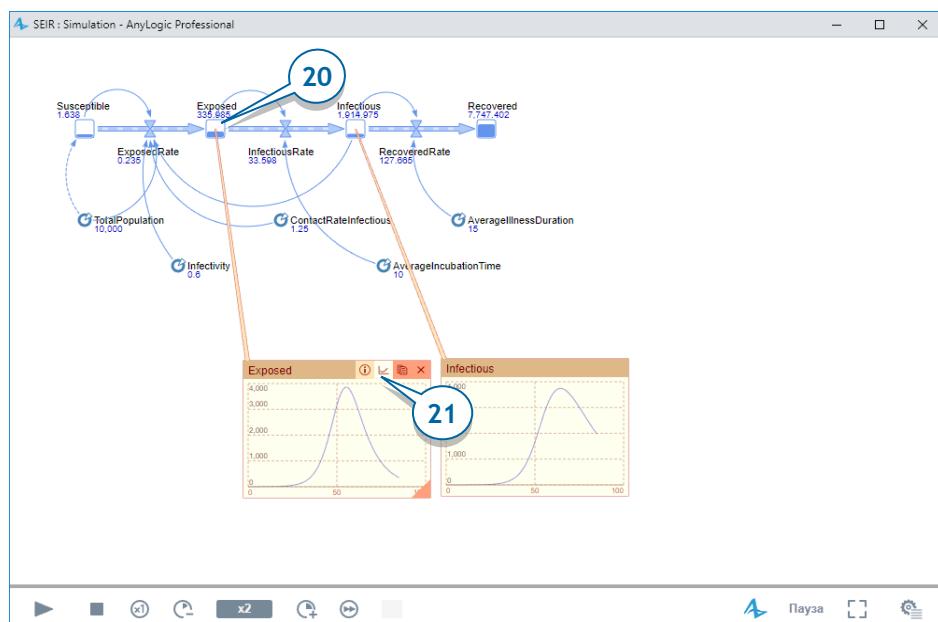
- 16.** Задайте следующую формулу для потока *InfectiousRate*:  
 $\text{Exposed}/\text{AverageIncubationTime}$
- 17.** Задайте следующую формулу для потока *RecoveredRate*:  
 $\text{Infectious}/\text{AverageIllnessDuration}$
- 18.** Добавьте все недостающие связи зависимостей. В результате диаграмма потоков и накопителей должна будет выглядеть следующим образом:



- 19.** Подкорректируйте вид связей. Измените радиусы дуг связей, чтобы сделать диаграмму более красивой и читаемой, например, как на рисунке ниже. Чтобы изменить изгиб связи, выделите связь в редакторе и перетащите метку, расположенную посередине связи.



- 20.** Запустите модель и исследуйте динамику процесса с помощью похожих на виджеты информационных окон этих переменных. Открыть информационное окно переменной можно, щелкнув мышью по этой переменной. Чтобы изменить размер окна, потяните за правый нижний угол этого окна.



- 21.** Чтобы переключить информационное окно в режим графика, проведите курсором мыши по заголовку окна и щелкните появившийся справа значок графика.
- 22.** Увеличьте скорость выполнения модели, чтобы моделирование проходило быстрее.

## Фаза 2. Добавление графика для визуализации динамики процесса

### Циклы обратной связи: уравновешивающие и усиливающие

Системная динамика изучает системы с обратными связями, то есть системы, образованные (возможно, зависящими друг от друга) циклами обратной связи.

Есть два типа циклов обратной связи: *усиливающие* и *уравновешивающие*. Определить тип цикла можно с помощью следующих правил.

Начните с предположения, что значение переменной увеличивается, и проследите за изменением значений входящих в цикл переменных.

Цикл является:

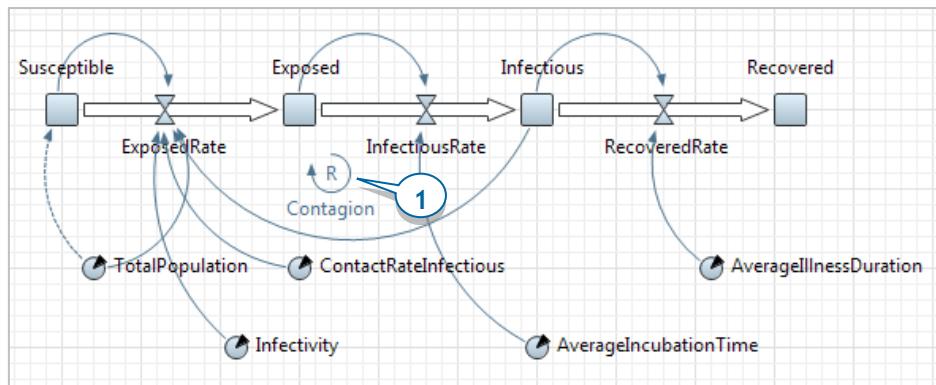
- *усиливающим*, если после прохождения по циклу вы видите тот же результат, что был допущен при начальном предположении;
- *уравновешивающим*, если результат противоречит начальному предположению.

Есть и другой способ определения типа цикла:

- Усиливающие циклы содержат четное (или нулевое) количество отрицательных связей (то есть, связей, уменьшающих значение зависимой переменной).
- Уравновешивающие циклы содержат нечетное количество отрицательных связей.

Добавим на диаграмму метку для образовавшегося в нашей системе цикла зависимостей.

1. Перетащите элемент **Цикл**  из палитры **Системная динамика** на диаграмму и расположите его так, как показано на следующем рисунке.



2. Перейдите в панель **Свойства** и измените **Тип** цикла на **R** (что означает *Reinforcing*, то есть «усиливающий»). Оставьте заданное по умолчанию **Направление:** **по часовой стрелке** и укажите текст, который AnyLogic будет отображать возле значка цикла: *Contagion* (то есть, «заражение»).

## Элемент «Цикл»

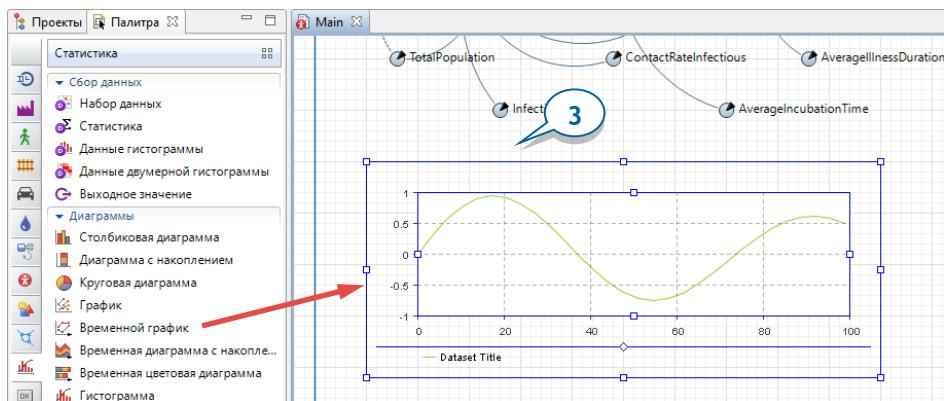
Элемент AnyLogic **Цикл** представляет собой графический значок, состоящий из метки с описанием смысла цикла и стрелки, показывающей направление этого цикла. Элемент не задает саму логику зависимостей в моделируемой системе, а только показывает информацию об образовавшемся цикле влияний переменных друг на друга. Добавляя на диаграмму значки циклов, вы можете облегчить понимание существующих в этой диаграмме циклов обратной связи будущим пользователям этой модели.

Давайте определим тип нашего цикла *Contagion*. Увеличение значения накопителя *Infectious* ведет к увеличению значения потока *ExposedRate*, что в свою очередь увеличивает значение накопителя *Exposed*. Следовательно, цикл *Contagion* является усиливающим. Все связи в этом цикле положительные.

Определите, какие еще циклы присутствуют в моделируемой системе? Каких они типов?

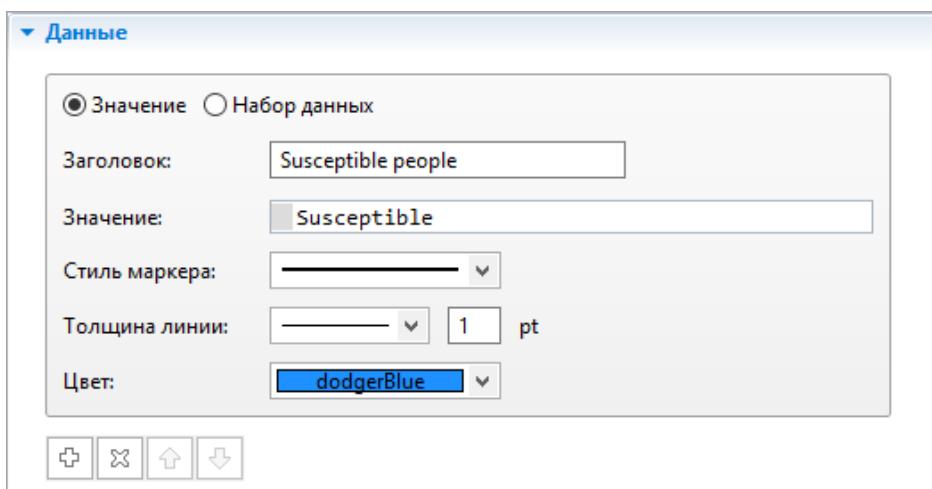
Теперь давайте добавим временной график для просмотра динамики изменения численности каждой категории людей в нашей модели.

3. Перетащите элемент **Временной график** из палитры **Статистика** на диаграмму и увеличьте размер графика, как показано на рисунке ниже.

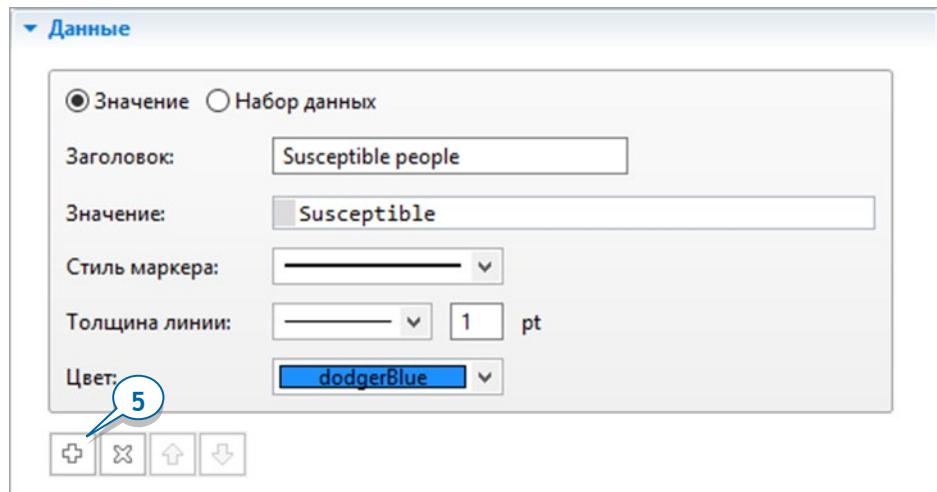


4. В панели **Свойства** перейдите в раздел **Данные**. Измените свойства элемента данных, созданного для графика по умолчанию:

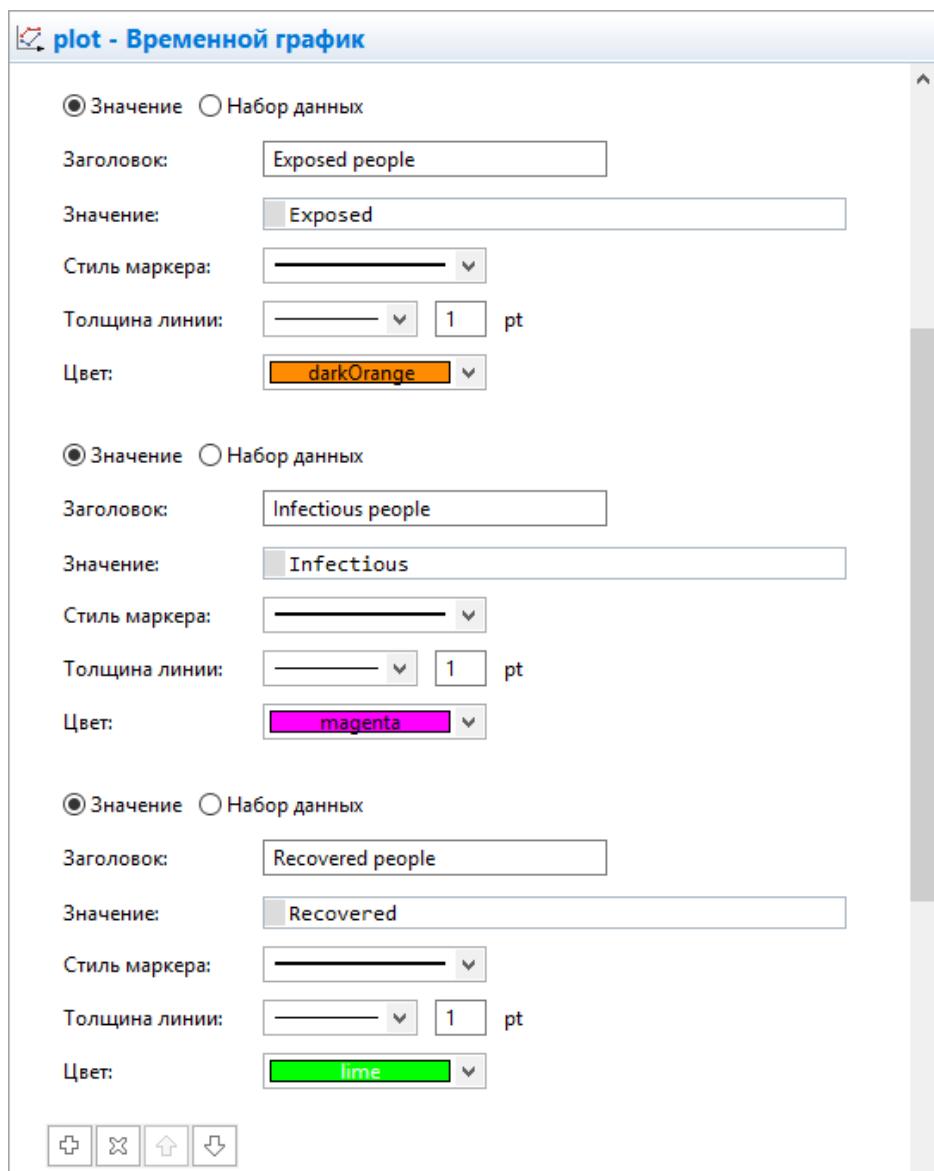
- Заголовок:** *Susceptible people* (то есть, восприимчивые к заболеванию люди).
- Значение:** *Susceptible* (при введении имени переменной используйте мастер подстановки кода).



5. Добавьте еще три элемента данных, которые будут отображать значения накопителей *Exposed*, *Infectious*, и *Recovered* соответственно. Чтобы добавить элемент, нажмите кнопку **Добавить**.



Не забудьте прописать каждому элементу соответствующий **Заголовок**.



- Чтобы график собирал данные на протяжении всего времени выполнения модели, в разделе свойств **Обновление данных** измените значение опции **Отображать до: 300 последних значений**.

7. В разделе свойств **Масштаб** убедитесь, что график отражает данные для 300 единиц модельного времени. Для этого установите **Временной диапазон** на **300 единиц мод. времени**.
8. Мы закончили создание модели. Запустите ее и понаблюдайте за динамикой распространения болезни с помощью добавленного нами графика.

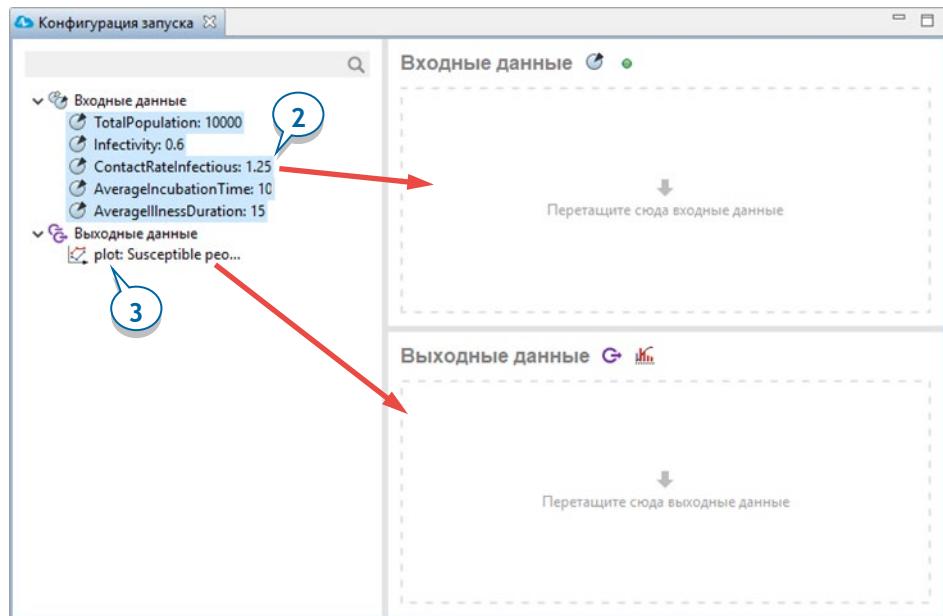


## Фаза 3. Эксперимент варьирования параметров

Теперь давайте изучим, как меняется динамика распространения эпидемии при различных значениях интенсивности контактов между людьми, воспользовавшись экспериментом варьирования параметров. Мы запустим этот эксперимент в облачном сервисе («облаке») AnyLogic Cloud.

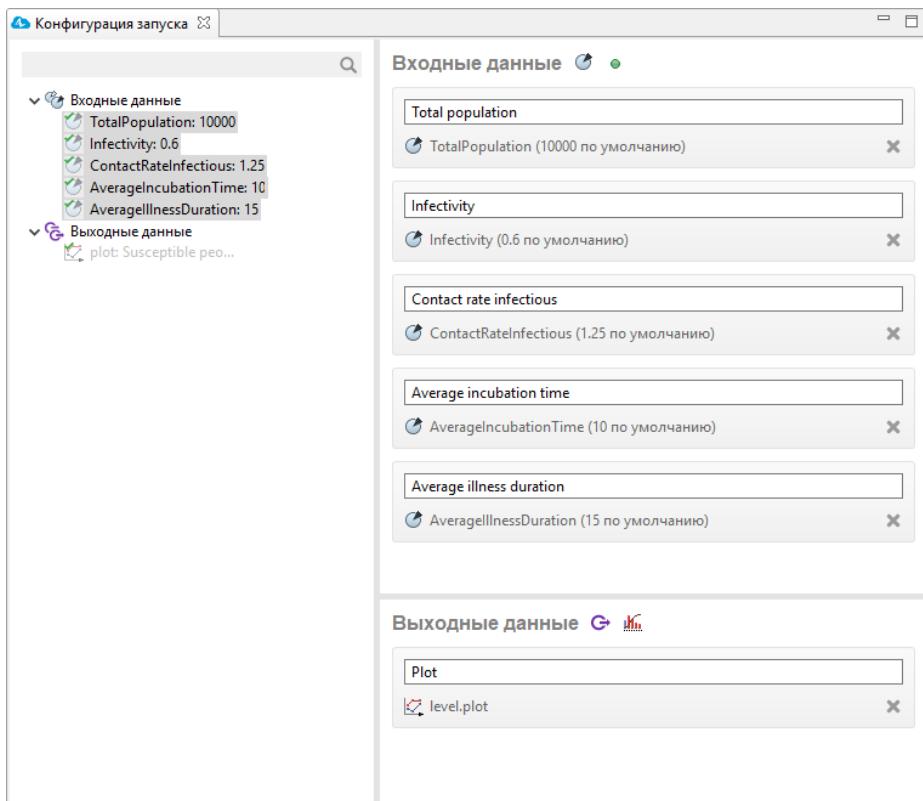
### *AnyLogic Cloud*

- AnyLogic Cloud — это облачный сервис, позволяющий запускать модели онлайн с любого устройства, в том числе с телефонов и планшетов, и делиться моделями с другими пользователями.
  - AnyLogic Cloud — это мощный инструмент для анализа моделей, предлагающий широкий набор экспериментов и средств анализа данных.
  - Сервис AnyLogic Cloud располагается на платформе Amazon Web Services и доступен каждому. Даже если вы не используете AnyLogic, вы все равно можете воспользоваться Cloud для получения представления о моделировании.
1. Дважды щелкните по элементу **Конфигурация запуска: Main** в панели **Проекты**.
  2. Откроется редактор **Конфигурация запуска**. Здесь можно указать входные и выходные данные модели перед тем, как загрузить ее в облако AnyLogic. Выберите все параметры из списка **Входные данные** и перетащите их вправо в пустое пространство **Входные данные**.



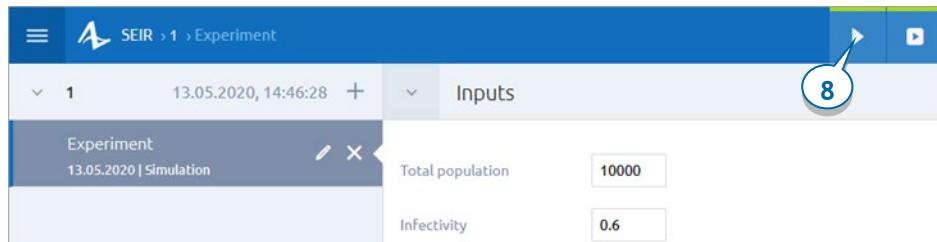
3. Перетащите единственный элемент из списка **Выходные данные (plot)** на пустую правую панель **Выходные данные**.

Панели **Входные данные** и **Выходные данные** заполняются выбранными нами элементами. Когда модель будет экспортирована в AnyLogic Cloud, значения выбранных параметров можно будет регулировать. График будет использоваться для вывода данных облачной модели.

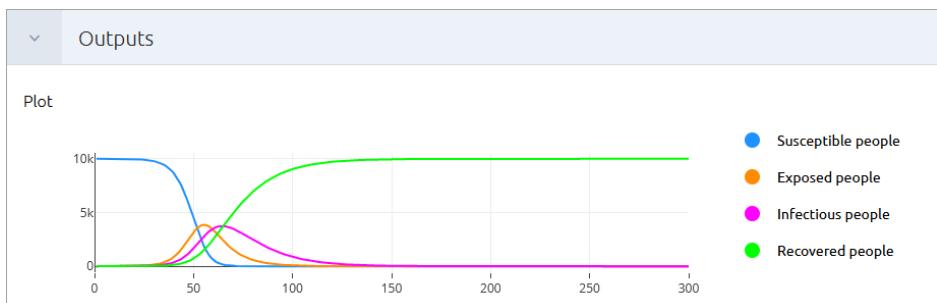


4. Убедимся, что эксперимент моделирует ровно 300 дней: для этого следует ограничить время выполнения эксперимента. На панели свойств **Конфигурации запуска** раскройте раздел **Модельное время**. Выберите **В заданное время** в выпадающем списке **Остановить** и укажите **300** в качестве **Конечного времени**.
5. Щелкните по ссылке **Экспорт модели в облако AnyLogic** в основной части панели свойств.
6. Откроется диалоговое окно **Экспорт модели в облако AnyLogic**, с помощью которого нужно зарегистрироваться или авторизоваться в AnyLogic Cloud, а также настроить параметры загрузки модели.
7. После завершения настройки и загрузки, в новой вкладке установленного по умолчанию браузера откроется страница AnyLogic Cloud, автоматически созданная для только что загруженной модели. Обратите внимание – чтобы увидеть эту страницу, необходимо предварительно авторизоваться в Cloud.

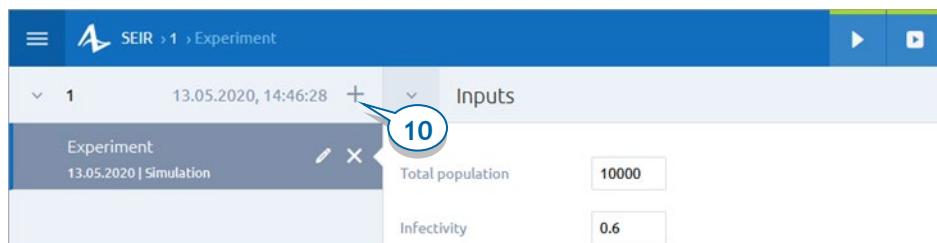
8. Для начала давайте запустим в облаке простой эксперимент. Откройте эксперимент, щелкнув по его имени (*Experiment*) в списке слева, а затем нажмите кнопку **Run** в верхней части веб-страницы.



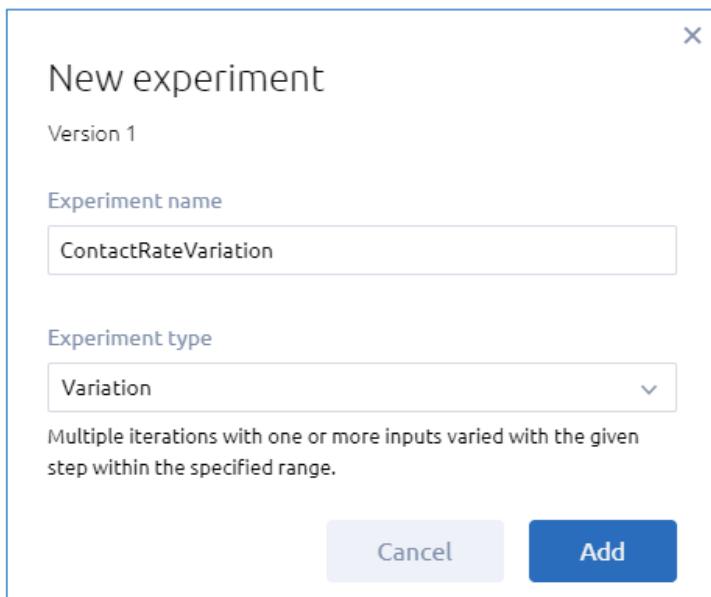
9. В разделе **Outputs** появится график, похожий на тот, который вы видели после запуска модели в AnyLogic.



10. Давайте создадим другой эксперимент в Cloud. Нажмите кнопку с плюсом сверху на левой панели.



11. Откроется всплывающее окно **New experiment**. В поле **Experiment name** введите *ContactRateVariation*. Выберите **Variation** в выпадающем списке **Experiment Type** и нажмите **Add**.

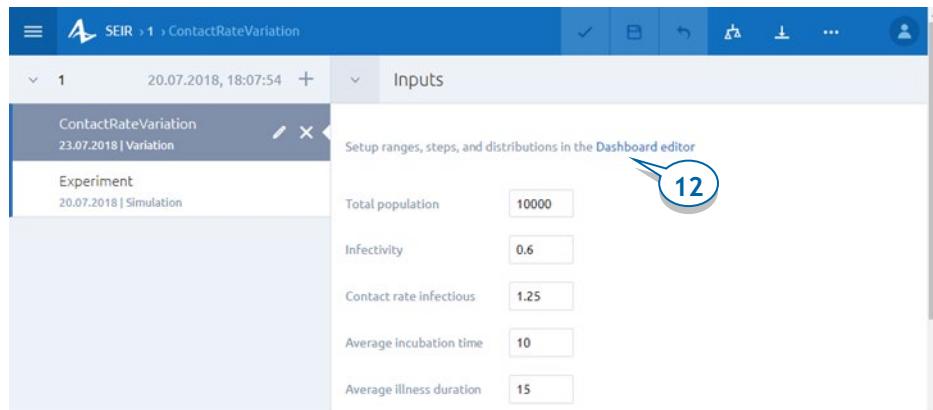


### Эксперимент варьирования параметров

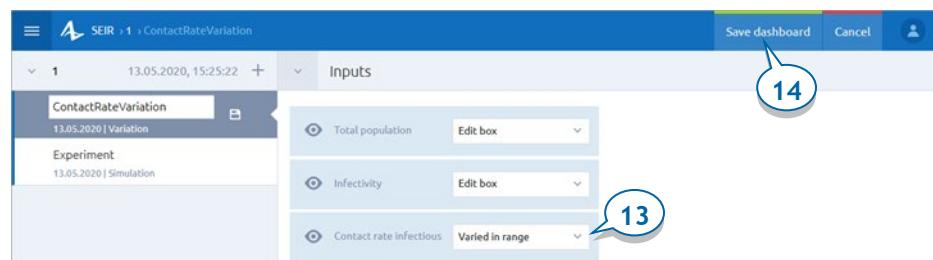
С помощью Эксперимента варьирования параметров мы можем осуществлять сложное моделирование, в рамках которого производится серия запусков модели с отличающимися значениями одного или нескольких параметров. После завершения эксперимента результаты всех запусков отображаются на одной диаграмме, что позволяет понять, как изменение значений параметров влияет на результат прогона модели.

Если мы запустим эксперимент с фиксированными значениями параметров, мы также сможем оценить влияние случайных факторов на стохастические модели.

**12.** На левой панели появится еще один эксперимент. В разделе **Inputs** отображаются параметры агента верхнего уровня этого эксперимента: в нашем случае это *Main*. По умолчанию у всех этих параметров фиксированные значения, которые не будут изменяться в ходе моделирования. Чтобы наш эксперимент варьировал интенсивность контактов зараженных людей, это поведение нужно настроить в **Dashboard editor**. Щелкните по ссылке, показанной на рисунке ниже.

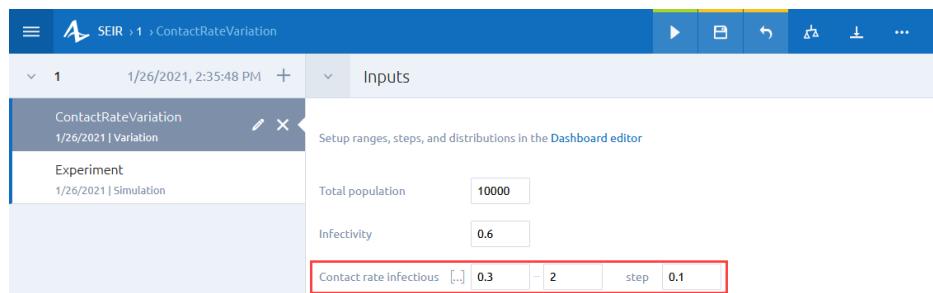


13. Найдите параметр *Contact rate infectious* и измените его тип на **Varied in range** в выпадающем списке.



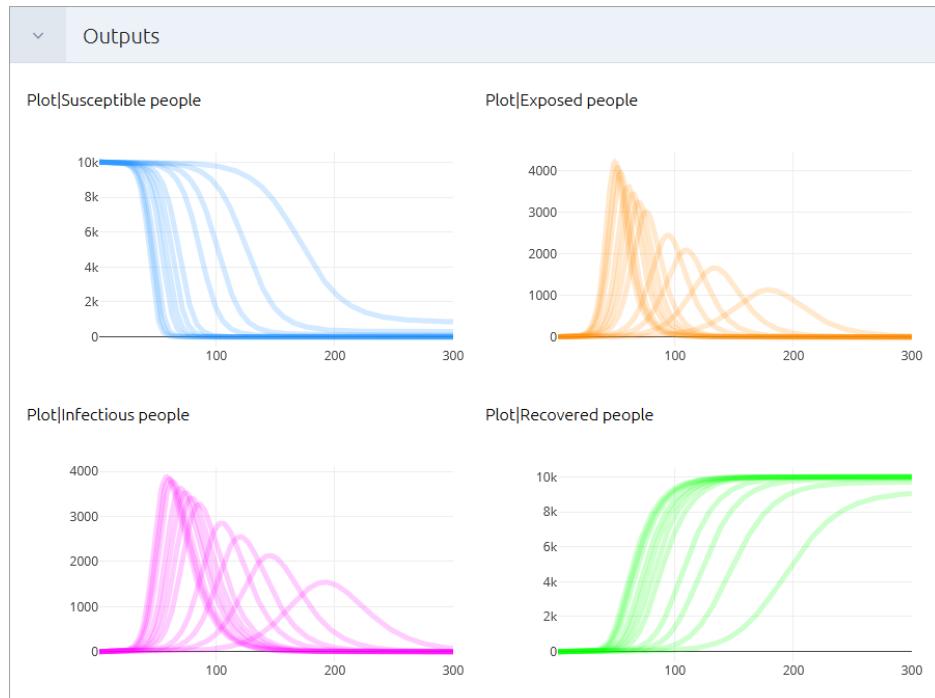
14. Щелкните по кнопке **Save Dashboard**, чтобы сохранить изменения.

15. Укажите 0.3 в качестве минимального значения параметра, а 2 — в качестве максимального. В поле **step** введите шаг: 0.1.



16. Мы готовы запустить эксперимент и понаблюдать за изменением динамики распространения заболевания на нескольких прогонах с помощью графиков. Нажмите кнопку **Run**, чтобы начать эксперимент варьирования параметров.

Эксперимент варьирования параметров произведет несколько прогонов модели с отличающимися значениями параметра *Contact rate infectious* и выведет результаты моделирования на графиках.



Каждый график включает результаты нескольких прогонов (по одной кривой на запуск) — всего 18. Другими словами, мы видим 18 сценариев заболеваемости для разных показателей интенсивности контактов, варьирующихся от 0.3 до 2. Эти сценарии отражают 18 шагов в рамках диапазона значений для параметра, который мы задали ранее.

Наведите курсор мыши на кривую, чтобы увидеть значение параметра, использованное для получения этой кривой. Вы увидите, как увеличение интенсивности контактов позволяет инфекции распространяться быстрее.

## Фаза 4. Калибровка параметров модели

Мы знаем, что созданная нами диаграмма потоков и накопителей абсолютно точно отражает моделируемый процесс распространения эпидемии. Но значения определенных параметров (а именно - *Infectivity* и *ContactRateInfectious*) нельзя измерить напрямую, поэтому, чтобы быть уверенными в достоверности результатов работы модели, нам нужно провести валидацию значений этих параметров до того, как мы начнем использовать модель. Если у вас есть наблюдения за поведением моделируемой системы в реальной жизни, то лучше всего с этой задачей справится *калибровка* - эксперимент, который подбирает значения параметров таким образом, чтобы поведение модели наиболее точно совпадало с поведением моделируемого объекта реального мира.

### Калибровка

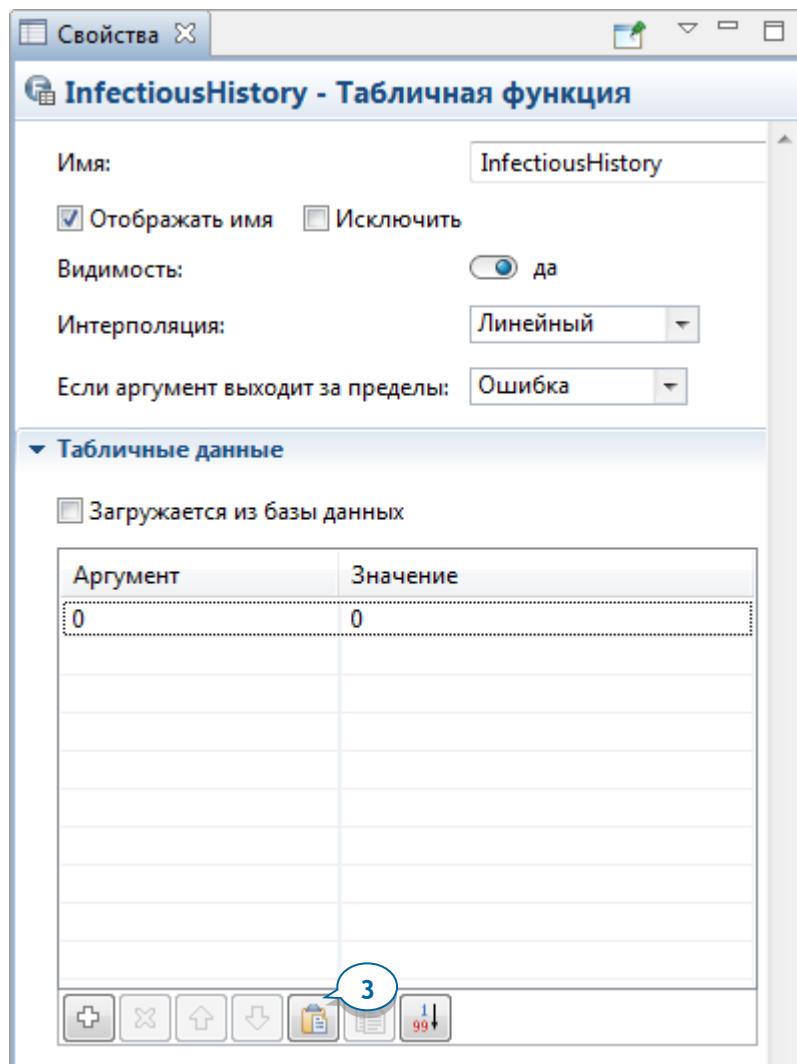
- Эксперимент калибровки многократно запускает модель и сравнивает результаты каждого прогона с реальными (историческими) данными. Используя эвристический алгоритм, оптимизатор предлагает новую комбинацию значений параметров для каждого последующего прогона модели. После выполнения указанного количества прогонов эксперимент выберет значения параметров, при которых результаты моделирования были наиболее близки к реальной ситуации.
- Эксперимент калибровки AnyLogic использует встроенный оптимизатор OptQuest, разработанный компанией OptTek Systems, Inc.

Вначале добавим в модель исторические данные: проведенные в реальной жизни ежедневные измерения количества заболевших людей во время вспышки эпидемии. Эти данные хранятся в текстовом файле в виде таблицы, и мы сможем построить по этим данным график динамики распространения заболевания с помощью табличной функции AnyLogic.

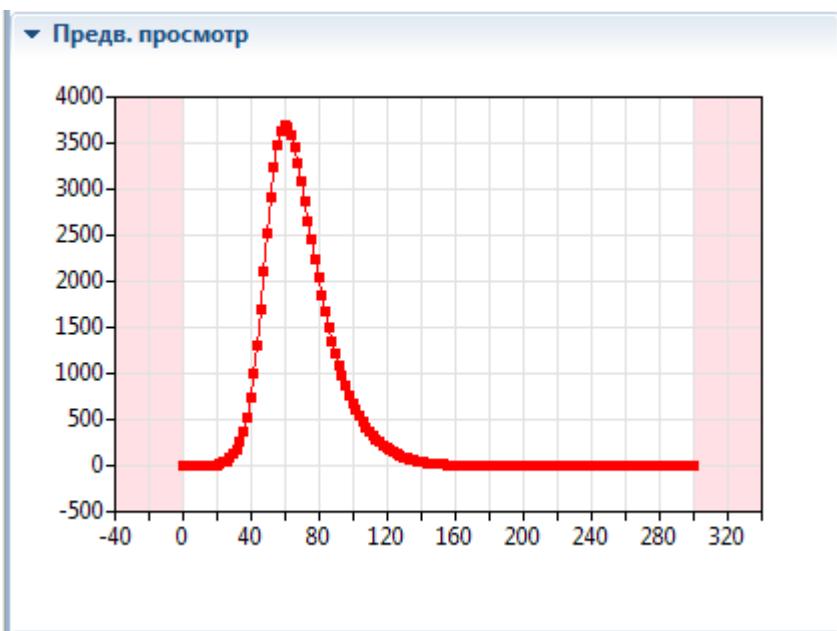
## Табличная функция

- Табличная функция – это функция, данные которой заданы в табличном виде. Пользователь указывает несколько пар (аргумент, значение), и AnyLogic строит функцию, используя эти данные и заданный пользователем тип интерполяции.
- При вызове функции пользователь указывает значение аргумента функции, и она возвращает соответствующее этому аргументу значение. Если данный аргумент отсутствует в табличных данных, то для получения соответствующего ему значения функции используется интерполяция.
- С помощью табличных функций можно задавать сложные нелинейные зависимости, которые не могут быть описаны как сочетание стандартных функций. Также табличные функции могут использоваться для приведения заданных в табличном виде экспериментальных данных в непрерывный вид.

1. Откройте диаграмму *Main* и добавьте на нее Табличную функцию  из палитры Системная динамика. Назовите эту функцию *InfectiousHistory*.
2. Откройте текстовый файл *HistoricData.txt* из папки Каталог *AnyLogic/resources/AnyLogic in 3 days/SEIR*. Каталог AnyLogic здесь – это тот каталог на вашем компьютере, куда установлен AnyLogic, например, *Program Files/AnyLogic 8 Professional*.
3. Скопируйте содержимое текстового файла в буфер обмена, затем перейдите в секцию свойств табличной функции Табличные данные и щелкните по кнопке Вставить из буфера  . Столбцы таблицы Аргумент и Значение заполняются данными.



4. Откройте секцию свойств табличной функции **Предв. просмотр**, и вы увидите кривую динамики распространения болезни, которая наблюдалась в реальной жизни.

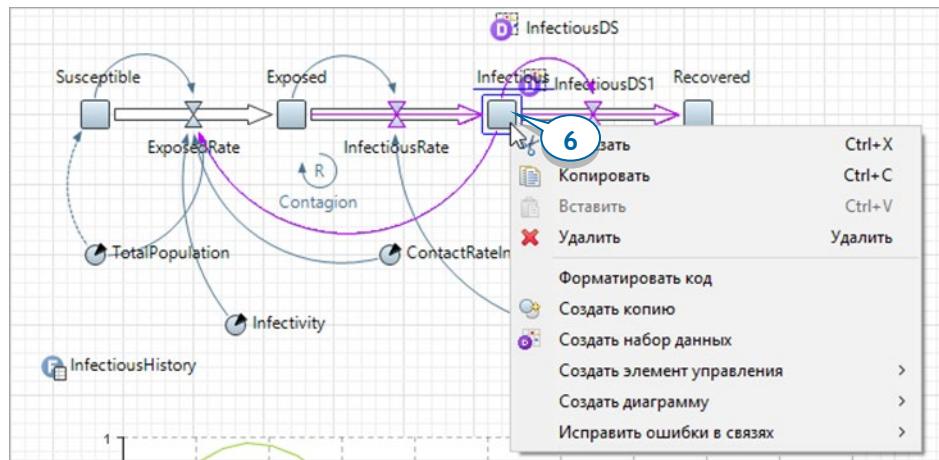


5. Выше в свойствах, выберите у параметра **Если аргумент выходит за пределы** опцию **Ближайший**. В этом случае функция будет корректно обрабатывать случаи, когда функции передается аргумент, лежащий за пределами интервала аргументов, заданного нами в секции **Табличные данные**.

Поскольку мы выбрали опцию **Ближайший**, функция будет экстраполироваться, используя значения, заданные для крайних (минимального и максимального) допустимых аргументов. Это значит, что для всех аргументов слева (справа) от допустимой области аргументов, значение функции будет равно значению, которое она принимает в самой левой (правой) точке интервала своих аргументов. График в секции **Предв. просмотр** отображает текущие настройки интерполяции и экстраполяции табличной функции.

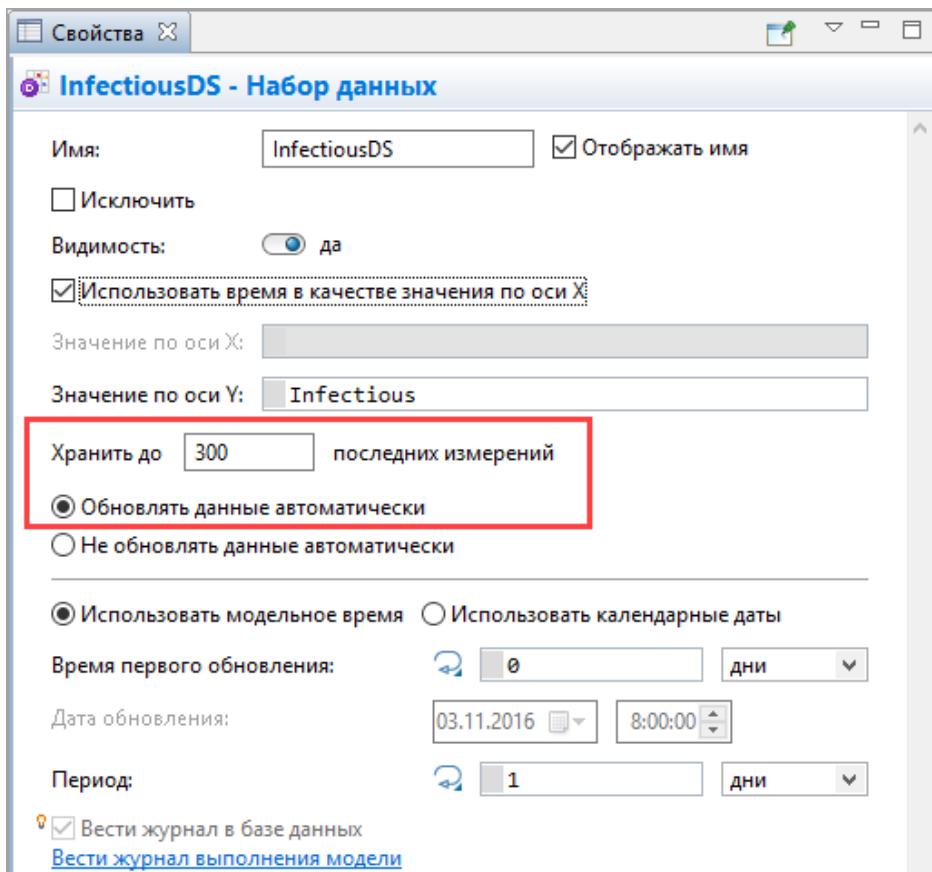
Теперь мы добавим набор данных, чтобы собирать информацию о количестве заразных людей во время выполнения модели.

6. Щелкните правой кнопкой мыши по накопителю *Infectious* и выберите пункт **Создать набор данных**.



7. Когда набор данных InfectiousDS появится на диаграмме, откройте его свойства.

Поскольку мы хотим понаблюдать за динамикой распространения заболевания, оставьте флажок на опции **Использовать время в качестве значения по оси X**.



8. Отметьте флажком опцию **Обновлять данные автоматически** и установите **Период обновления** в 1, чтобы добавлять к набору данных по одному значению на каждый моделируемый день.
9. Чтобы сохранить данные обо всем прогоне модели, укажите, что данный набор данных будет **Хранить до 300 последних измерений**.

Мы готовы создать наш эксперимент.

10. Щелкните правой кнопкой мыши по модели *SEIR* в панели **Проекты** и выберите из контекстного меню пункт **Создать > Эксперимент**. В окне мастера **Новый эксперимент** выберите **Калибровка** в секции **Тип эксперимента** и затем щелкните **Далее**.
11. В случае эксперимента калибровки настройка его параметров проводится прямо в окне Мастера создания эксперимента. Измените тип параметров,

которые мы хотим калибровать (*Infectivity* и *ContactRateInfectious*), с фиксированного на **непрерывный**. Задайте минимальное (Мин) и максимальное (Макс) значения диапазона калибровки.

Параметры:						
Параметр	Тип	значение	Мин	Макс	Шаг	
TotalPopulation	фиксированный					
Infectivity	непрерывный		0.005	1		
ContactRateInfectious	непрерывный		0.01	3		
AverageIncubationTime	фиксированный					
AverageIllnessDuration	фиксированный					

12. Введите следующую информацию в расположенную ниже таблицу критериев калибровки:

- **Заголовок:** *Infectious curve match*
- **Тип:** выберите из списка **набор данных**
- **Результат моделирования:** *root.InfectiousDS*
- **Реальные данные:** *root.InfectiousHistory*
- **Коэффициент:** *1.0*

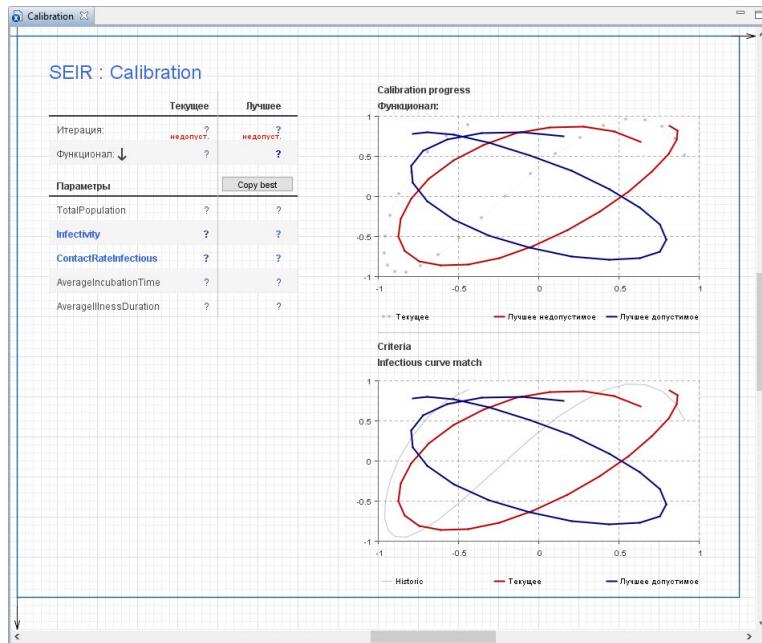
Критерии:				
Заголовок	Тип	Результат моделирования	Реальные данные	К...
<i>Infectious curve match</i>	набор данных	<i>root.InfectiousDS</i>	<i>root.InfectiousHistory</i>	1.0

Агент верхнего уровня *Main* доступен здесь как *root*. Результаты работы нашей модели сохраняются в набор данных *InfectiousDS*. После окончания прогона модели хранящиеся в этом наборе данные будут сравниваться с данными из табличной функции *InfectiousHistory*.

В нашей модели только один критерий калибровки, но их может быть и несколько, в таком случае вы можете присвоить критериям различные весовые коэффициенты.

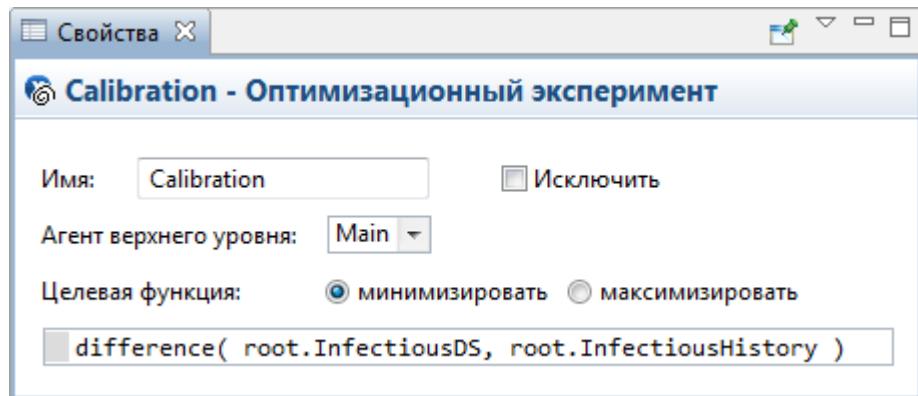
13. Щелкните по кнопке **Готово**. При этом откроется диаграмма эксперимента *Calibration* с созданным по умолчанию интерфейсом этого эксперимента.

- 14.** Щелкните по синей линии, располагающейся над осями координат, в графическом редакторе *Calibration*. Это линия рамки: рамка задает область, которая будет отображаться в окне модели. В свойствах рамки установите **Ширину** на 900, а **Высоту** — на 700, чтобы сгенерированные графики поместились в окно модели.

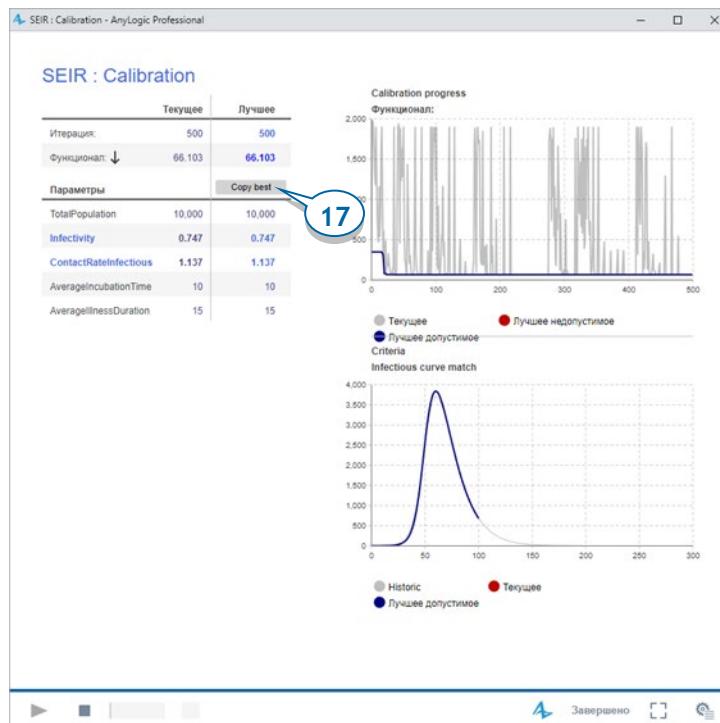


На рисунке ниже показаны свойства эксперимента.

Цель эксперимента — минимизировать различия между результатами моделирования и реальными (историческими) данными. Для определения оптимального результата используется метод наименьших квадратов.



- 15.** Откройте секцию свойств **Специфические** и снимите флагок **Разрешить параллельное выполнение итераций**.
- 16.** Запустите эксперимент: щелкните правой кнопкой мыши в панели **Проекты** по эксперименту *Calibration* и выберите **Запустить** из контекстного меню.



**17.** Когда эксперимент калибровки будет завершен, вы сможете скопировать полученные значения параметров, щелкнув по кнопке **Copy best**. Чтобы использовать скопированные значения параметров в нашем эксперименте *Simulation*, щелкните по кнопке **Вставить из буфера** на странице свойств этого эксперимента.

Найденные экспериментом калибровки значения параметров *Infectivity* и *ContactRateInfectious* будут подставлены в соответствующие поля.

Теперь мы можем провести эксперимент *Simulation* с откалиброванными параметрами и на этом завершить работу с данной моделью.

# Дискретно-событийное моделирование в AnyLogic

*Дискретно-событийное моделирование* зародилось примерно тогда же, когда появилась системная динамика. В 1961 году инженер компании IBM Джонни Гордон представил программу GPSS, которая считается первой реализацией метода моделирования на основе дискретных событий. В наши дни существует множество различных программных инструментов для дискретно-событийного моделирования, в том числе и современная версия GPSS.

- ◆ Дискретно-событийное моделирование предполагает представление моделируемой системы в виде процесса, то есть последовательности операций, выполняемых с агентами.

Модель задается графически в виде диаграммы процесса, блоки которой представляют собой отдельные операции. Как правило, диаграмма процесса начинается с блока «источник», генерирующего агентов. Этот блок передает агентов в последующие блоки диаграммы, задающие операции моделируемого процесса. Завершается диаграмма процесса обычно блоком, уничтожающим этих агентов.

Под *агентами*, называющимися в GPSS *транзакциями*, а в некоторых других моделирующих программах - *заявками*, могут пониматься клиенты, пациенты, телефонные звонки, документы, компоненты изделий или сами изделия, поддоны, автомобили, проекты, идеи и т.д.

Под *ресурсами* может пониматься персонал, врачи, рабочие, оборудование и транспорт.

Типовыми операциями дискретно-событийной модели являются задержка (моделирующая выполнение определенной операции, например, обработку звонка или детали), обслуживание агента ресурсом, ветвление процесса и т.д. Поскольку агенты конкурируют за обладание ресурсами, необходимыми для выполнения операций, то это может приводить к задержкам, и практически во всех дискретно-событийных моделях присутствуют очереди.

Как времена прибытия агентов, так и времена их обслуживания обычно являются случайными величинами, и их значения генерируются функциями распределения вероятностей. Поэтому и сами дискретно-событийные модели являются стохастическими, и для получения репрезентативного результата модель должна проработать определенное время, или же нужно выполнить определенное количество прогонов модели.

Типовыми результатами дискретно-событийной модели являются:

- занятость ресурсов;
- время, проведенное агентом в системе или определенной ее части;
- длины очередей;
- время ожидания;
- пропускная способность и узкие места системы.

## Модель заводского цеха

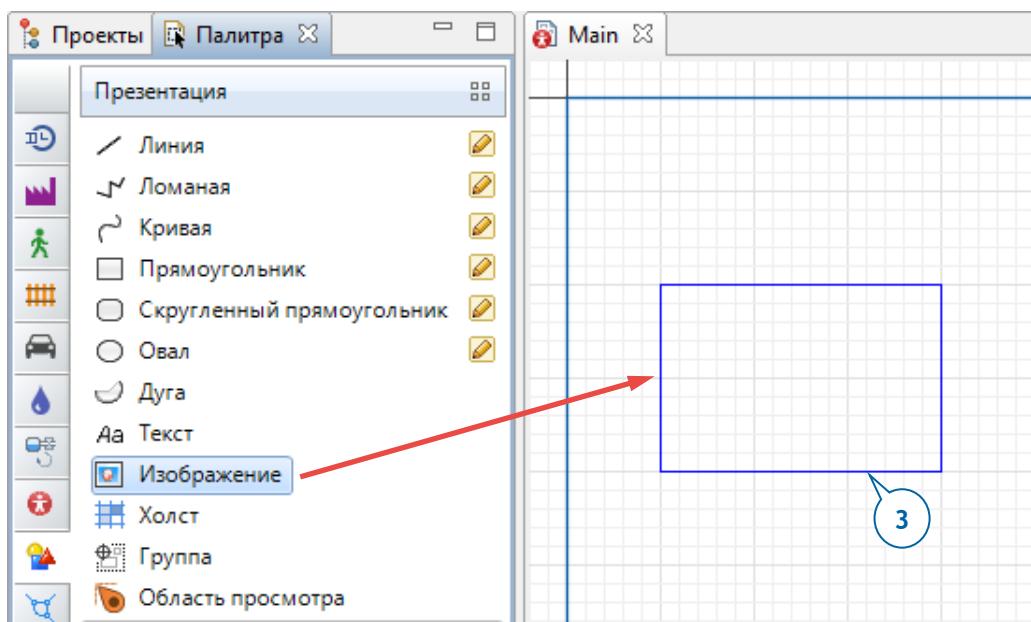
Мы промоделируем производственные процессы в небольшом заводском цеху:

- Каждый час на завод приезжает грузовик с поддонами. На каждом поддоне находится заготовка, готовая к обработке в данном цеху.
- Все находящиеся на грузовике поддоны разгружаются в приемной зоне цеха.
- Далее эти поддоны с помощью автопогрузчиков помещаются в подготовительную зону хранения.
- По прошествии определенного времени поддоны с заготовками доставляются автопогрузчиками к станку с ЧПУ. Здесь происходит обработка заготовок – производство конечных изделий.

## Фаза 1. Создание простой модели

Мы начнем с создания простой модели, имитирующей появление поддонов в приемной зоне заводского цеха и их последующее пребывание в зоне хранения.

1. Создайте новую модель. В мастере **Новая модель** задайте **Имя модели: Job Shop**, а также выберите **Единицы модельного времени: минуты**.
2. Откройте палитру **Презентация** . В этой палитре представлены графические элементы, с помощью которых вы можете нарисовать анимацию модели: прямоугольник, линия, овал, ломаная линия, кривая и т.д.
3. Перетащите элемент **Изображение** из палитры **Презентация** на диаграмму *Main*. Элемент **Изображение** можно использовать для добавления в модель изображений в различных графических форматах: PNG, JPEG, GIF, и BMP.

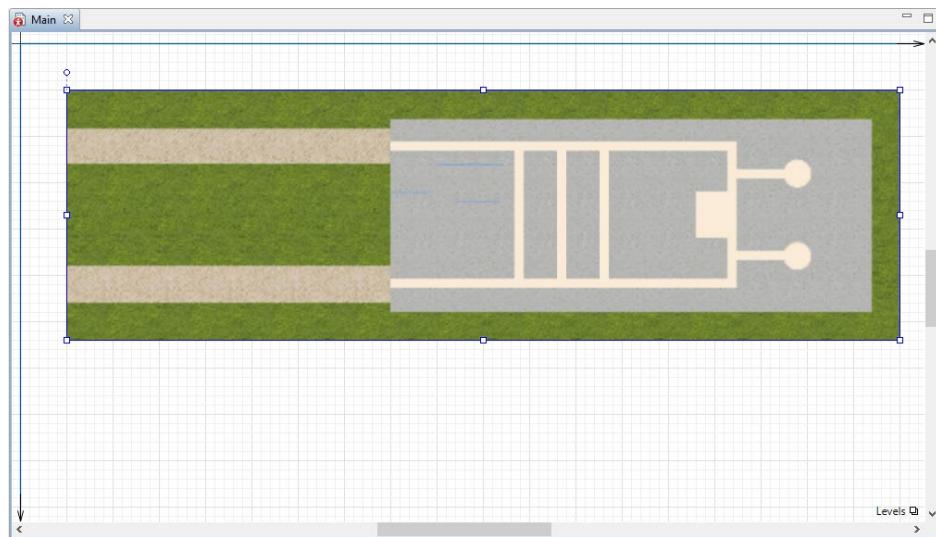


4. Откроется диалоговое окно, в котором вам будет предложено выбрать нужный файл изображения.
5. Перейдите в следующий каталог и выберите изображение *layout.png* :

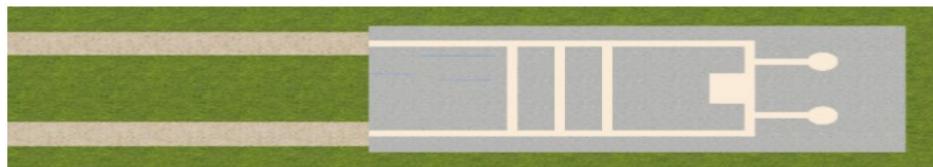
каталог *AnyLogic /resources/AnyLogic in 3 days/Job Shop*

Еще раз напомним, что каталог *AnyLogic* - это каталог, в который была установлена программа AnyLogic, например: *C:/Program Files/AnyLogic 8 Professional*.

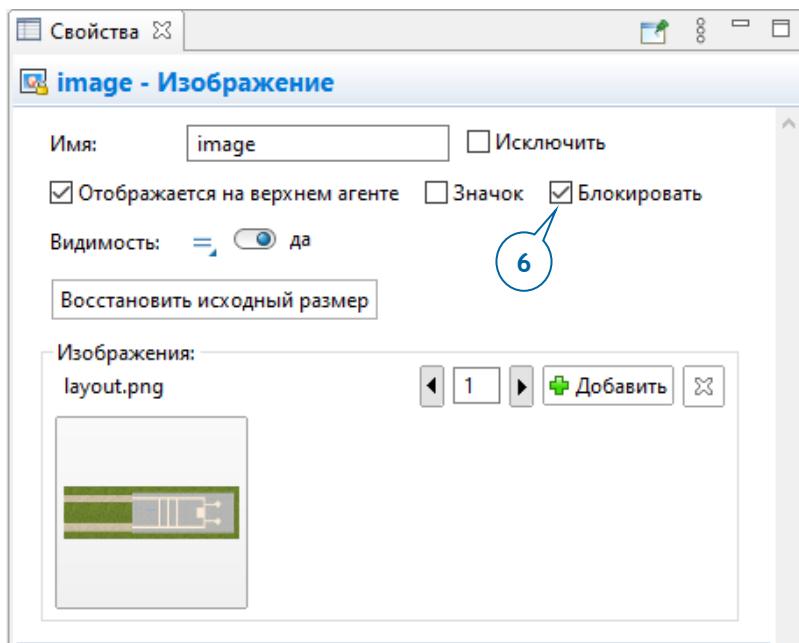
Когда вы выберете изображение, диаграмма агента *Main* станет выглядеть следующим образом:



AnyLogic добавляет изображение на диаграмму *Main* в его исходном размере, но вы можете изменить длину или ширину изображения. Если вы искажили пропорции изображения (как это показано на следующем рисунке), то вы можете восстановить исходные размеры изображения, щелкнув по кнопке **Восстановить исходный размер** на странице свойств этого изображения.



6. Выберите изображение в графическом редакторе. Если требуется зафиксировать местоположение и размер изображения, то в панели **Свойства** установите флажок **Блокировать**.



### Блокировка графического элемента

- Если вы заблокируете графический элемент, то он не будет реагировать на нажатия кнопок мыши и его будет невозможно выбрать в графическом редакторе. Блокировка очень полезна в тех случаях, когда вы рисуете элементы (обычно - элементы сети - пути, узлы, и т.д.) поверх изображений, представляющих собой планы заводов, больниц, складов.
- Если вам нужно снять блокировку с элемента, щелкните правой кнопкой мыши в графическом редакторе и выберите из контекстного меню пункт **Блокировка > Снять блокировку со всех фигур**.

Нашим следующим шагом будет рисование элементов из палитры **Разметка пространства** поверх плана фабричного цеха. Мы создадим транспортную сеть с помощью путей и узлов.

## Создание сети

Пути и узлы являются элементами разметки пространства, задающими местоположение агентов:

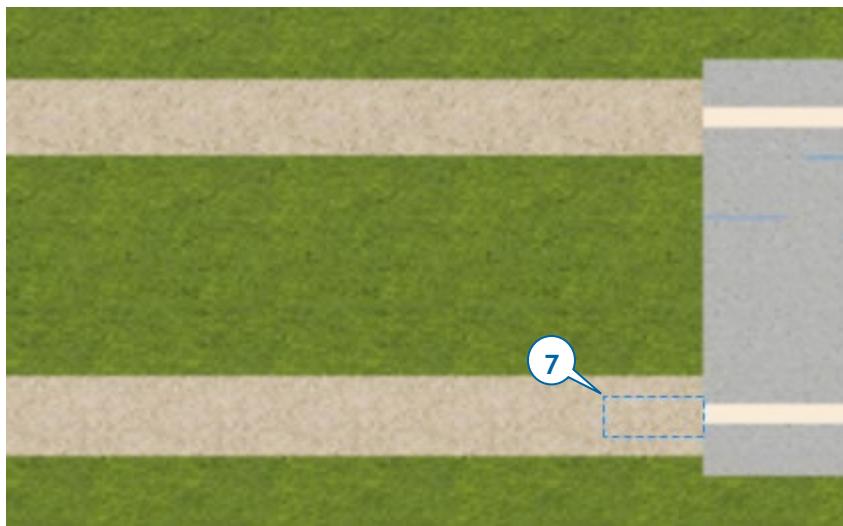
- Узел задает место, где агенты могут находиться в течение определенного времени для выполнения определенных действий;
- Путь задает маршрут, по которому агенты могут перемещаться между узлами.

Из узлов и путей формируется *сеть*, которую агенты могут использовать для перемещения по кратчайшему маршруту между исходным и конечным узлами. Сеть обычно создается, если происходящие в модели процессы протекают в определенном физическом пространстве, и в моделируемой вами системе есть подвижные агенты и ресурсы. Предполагается, что вместимость сегментов сети неограничена, и несколько агентов могут одновременно перемещаться по одному и тому же пути.

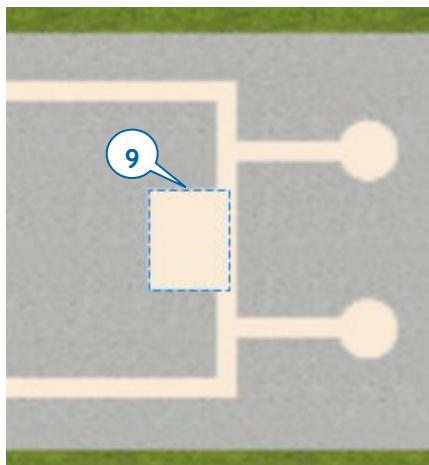
Давайте нарисуем сеть, задающую пути перемещения поддонов в нашей модели.

Вначале нарисуйте прямоугольный узел у входа в цех. Этот узел будет представлять в нашей модели приемную зону для поддонов.

7. Откройте палитру **Разметка пространства** и перетащите элемент **Прямоугольный узел**  на диаграмму *Main*. Измените размер узла, чтобы он выглядел так, как показано на следующем рисунке.

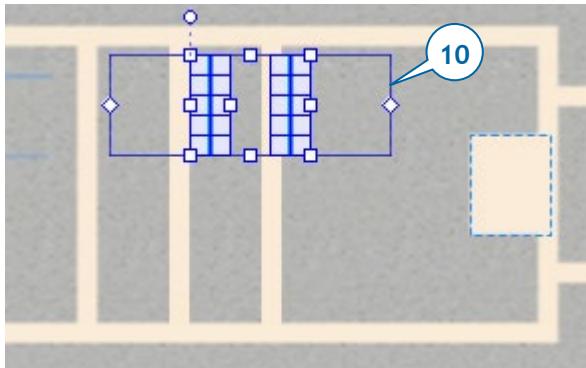


8. Присвойте созданному узлу имя *receivingDock* (приемная зона).
9. Нарисуйте узел, определяющий место парковки автопогрузчиков, когда они находятся в ожидании и не выполняют никаких заданий. С помощью еще одного **Прямоугольного узла**  нарисуйте зону стоянки согласно следующему рисунку и назовите этот узел *forkliftParking*.



Теперь давайте нарисуем путь перемещения автопогрузчиков в нашей модели.

- 10.** Нарисуйте склад, перетащив элемент **Склад**  из раздела **Производственные системы** палитры **Разметка пространства** на план цеха и расположив его, как показано на рисунке ниже.

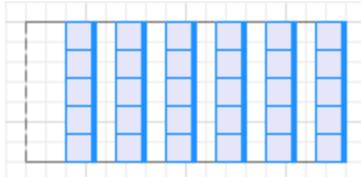


### Склад

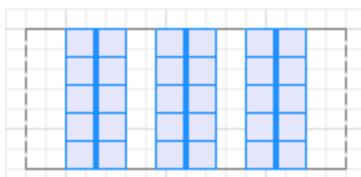
Элемент разметки пространства **Склад**  позволяет графически задать склад, где агенты (материальные объекты) размещаются в ячейках стеллажей.

Один элемент **Склад** может содержать несколько стеллажей. На данный момент мы поддерживаем два способа размещения стеллажей на складе:

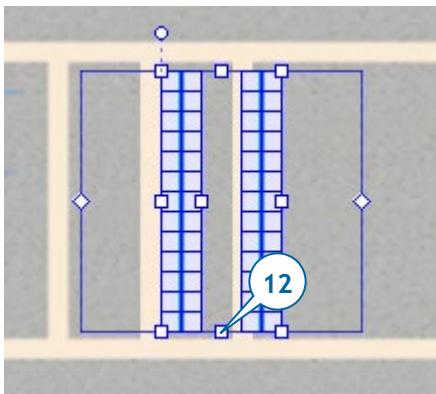
**По отдельности** - все стеллажи развернуты в одну сторону, и с каждого проезда есть доступ только к одному стеллажу.



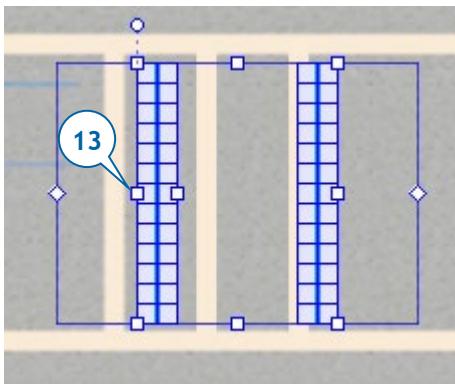
**Спина к спине** - стеллажи расположены попарно спиной друг к другу. Таким образом, с каждого проезда есть доступ к двум стеллажам.



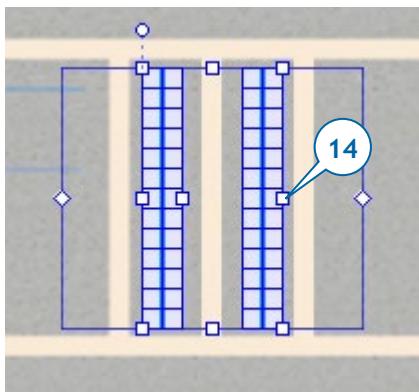
- 11.** В свойствах склада сделайте следующее:
- Выберите опцию **Кол-во секций**: вычисляется на основании размеров склада.
  - В секции **Стеллаж** задайте **Количество полок**: 2
- 12.** Измените размер фигуры склада, как показано на рисунке ниже. С текущими настройками, у склада должно стать 13 секций в каждом стеллаже.



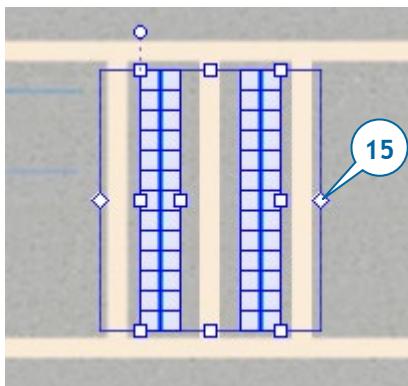
- 13.** Расположите левый стеллаж так, как показано на рисунке ниже, перетащив влево указанную метку-манипулятор.



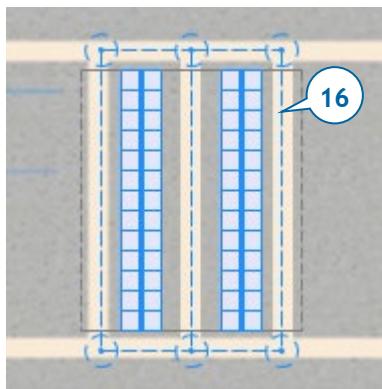
- 14.** Аналогично, перетащив вправо отмеченную на рисунке ниже метку-манипулятор, поместите правый стеллаж между проходами.



15. Уменьшите размер зоны доступа как на рисунке ниже с помощью самой правой метки-манипулятора.



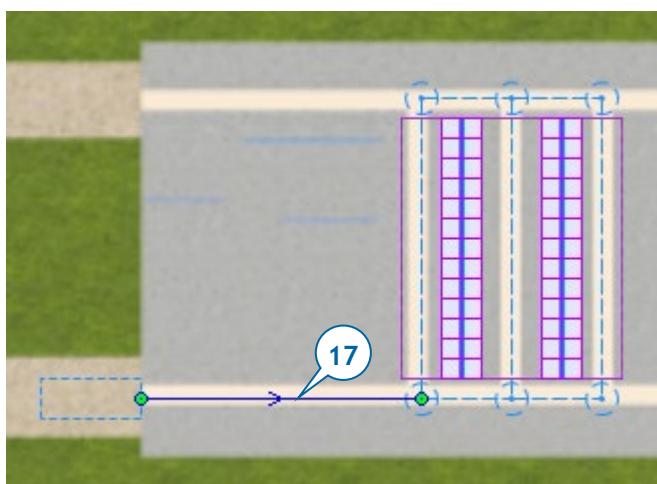
16. Теперь, когда мы завершили настройку склада и его расположения, пришло время создать транспортную сеть. Щелкните по кнопке **Создать складскую сеть** в панели свойств склада. Подтвердите действие в диалоговом окне, и вы увидите созданную сеть из путей, расположенных в проходах склада, и соединяющих эти пути узлов. Эта сеть будет использоваться автопогрузчиками, доставляющими и забирающими поддоны со склада.



Продолжим рисование сети, добавив пути, которые соединят оба нарисованных ранее узла с сетью склада. Вначале нарисуем путь, соединяющий сеть склада с узлом у входа в цех (*receivingDock*).

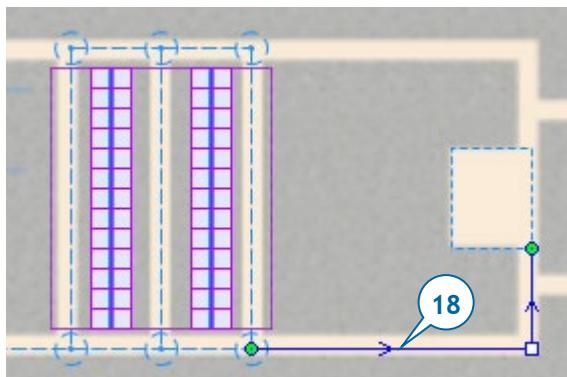
17. Чтобы нарисовать путь, необходимо выполнить следующее:

- На палитре **Разметка пространства** дважды щелкните по элементу **Путь**, при этом активируется режим его рисования.
- Нарисуйте путь согласно следующему рисунку. Начните рисование, щелкнув по границе узла *receivingDock*, а затем щелкните по центру нижнего левого точечного узла складской сети.

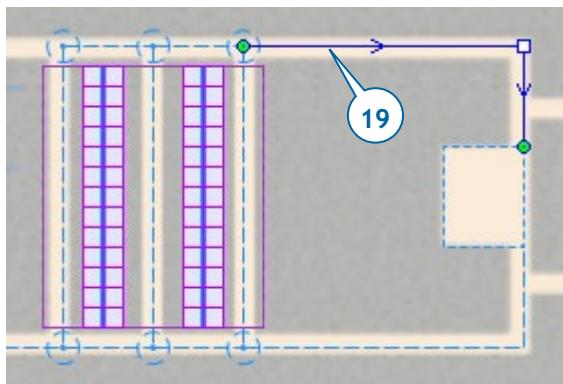


18. Нарисуйте еще один путь, как на рисунке ниже. Для этого щелкните по центру нижнего правого точечного узла складской сети, затем щелкните в

точке поворота дорожки на плане, чтобы добавить точку изгиба пути, и наконец щелкните по границе узла *forkliftParking*.



19. Аналогично нарисуйте еще один путь, как показано на рисунке ниже.



Если узлы сети правильно соединены путями, то при щелчке мышью по пути обе его точки соединения с узлами должны будут выделиться зеленым цветом.

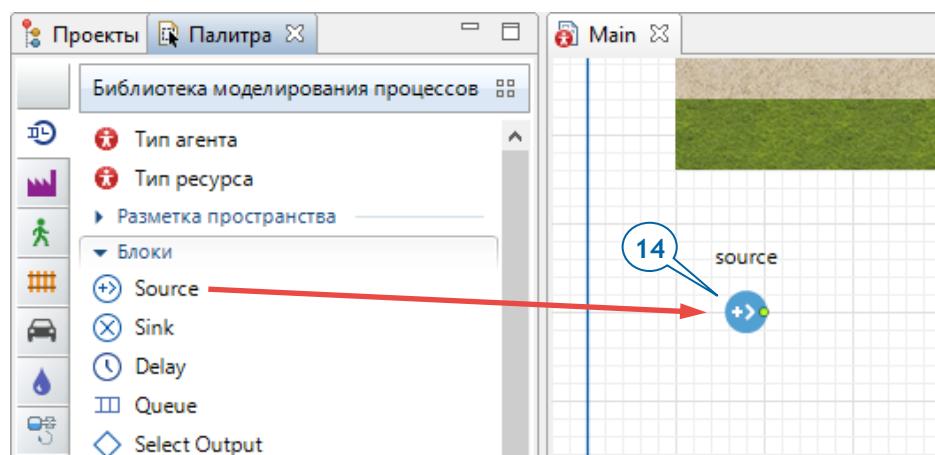
По умолчанию пути создаются как двунаправленные. Однако в случае необходимости вы можете разрешить движение по пути только в одном определенном направлении. Для этого нужно сбросить флагок **Двунаправленный** и затем указать направление движения. Направление движения по пути можно увидеть, выбрав путь; при этом в графическом редакторе будет показана стрелка направления.

Мы разметили пространство нашей модели, отметив поверх плана важные узлы и пути. Теперь мы займемся моделированием процессов с помощью Библиотеки моделирования процессов AnyLogic.

## Библиотека моделирования процессов

Если постановка вашей задачи формулирует систему как процесс - последовательность операций с заданными длительностями, то такую модель проще задавать с помощью дискретно-событийного моделирования. Для создания дискретно-событийных моделей AnyLogic предоставляет **Библиотеку моделирования процессов**. С ее помощью вы можете описывать процессы с помощью графических **диаграмм процесса**, собираемых из блоков библиотеки.

Давайте создадим диаграмму моделируемого нами процесса из блоков библиотеки.



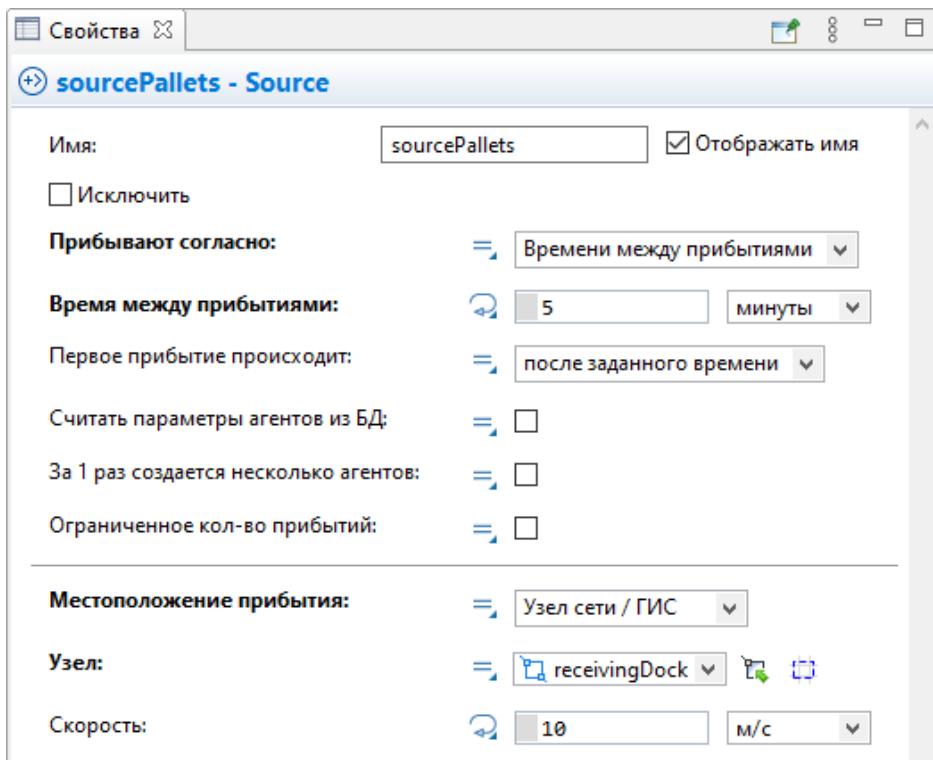
- 20.** Перетащите элемент **Source** из палитры **Библиотека моделирования процессов** на графическую диаграмму. Назовите созданный блок *sourcePallets*.

Обычно блок **Source** выступает в качестве стартовой точки процесса. Так и в нашей модели он будет использоваться для создания поддонов.

- 21.** Для того, чтобы поддоны появлялись в модели каждые пять минут и попадали в узел *receivingDock*, в панели **Свойства** блока *sourcePallets* необходимо выполнить следующее.
- Из списка **Прибывают согласно** выберите **Времени между прибытиями**.
  - В поле **Время между прибытиями** введите *5*, а из списка справа выберите **минуты**. Поддоны будут поступать раз в пять минут.

c. В области **Местоположение прибытия** выберите из списка Узел сети / ГИС.

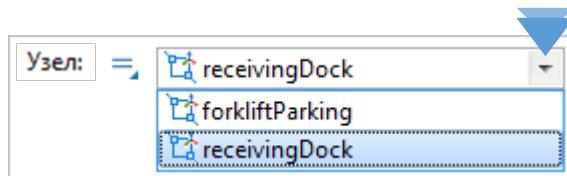
d. В расположеннном ниже списке Узел выберите *receivingDock*.



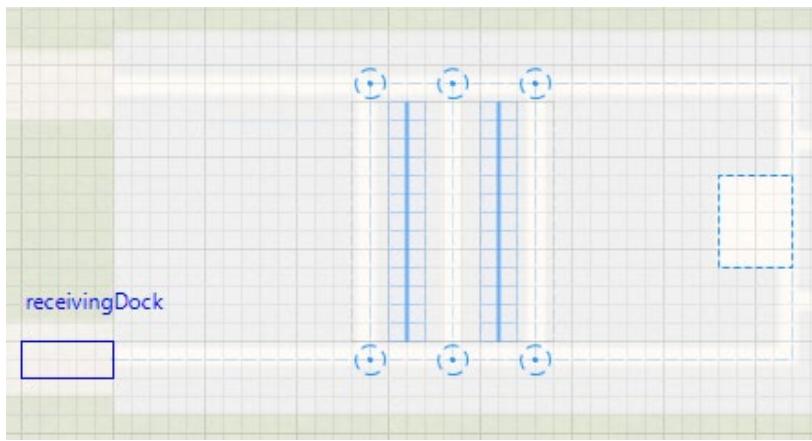
### Как сослаться на элемент модели из параметра блока

Существует два способа подстановки имени графического элемента в поле параметра блока библиотеки.

- Вы можете выбрать графический элемент прямо в панели Свойства, из выпадающего списка:

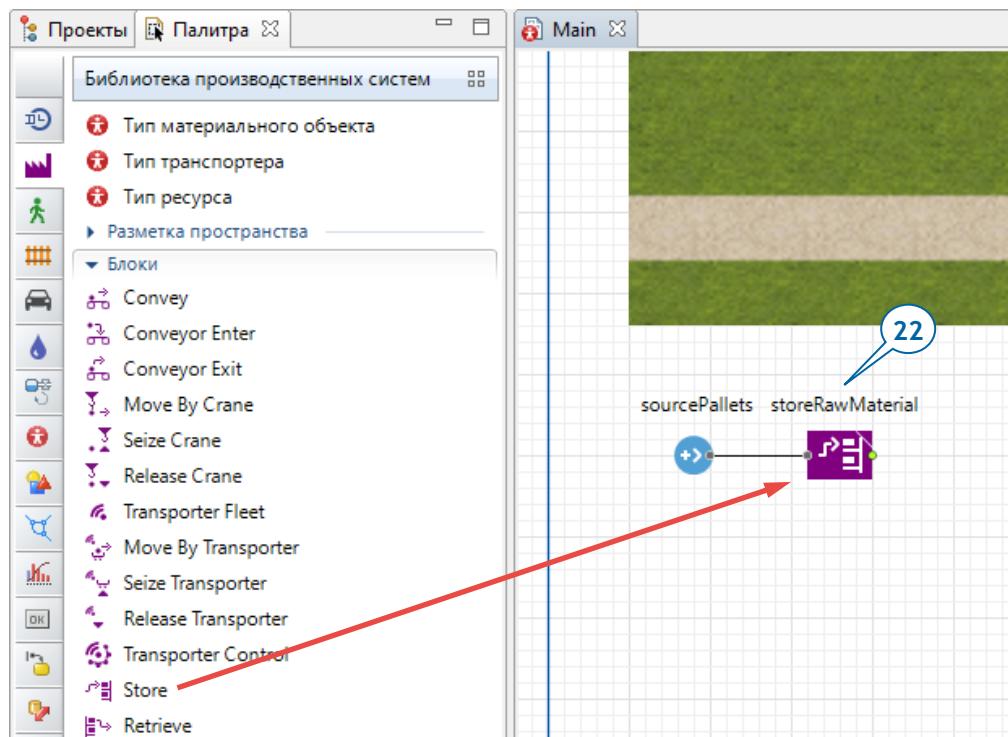


- Вы можете щелкнуть по кнопке  справа от показанного выше списка, а затем выбрать элемент в графическом редакторе, щелкнув по нему мышью. Обратите внимание, что в режиме выбора элемента на холсте, большинство элементов на диаграмме будет затенено, ограничивая ваш выбор только элементами допустимых типов:



Продолжайте создание диаграммы процесса. Теперь мы хотим добавить логику управления складом, и для того мы воспользуемся блоками **Библиотеки производственных систем**.

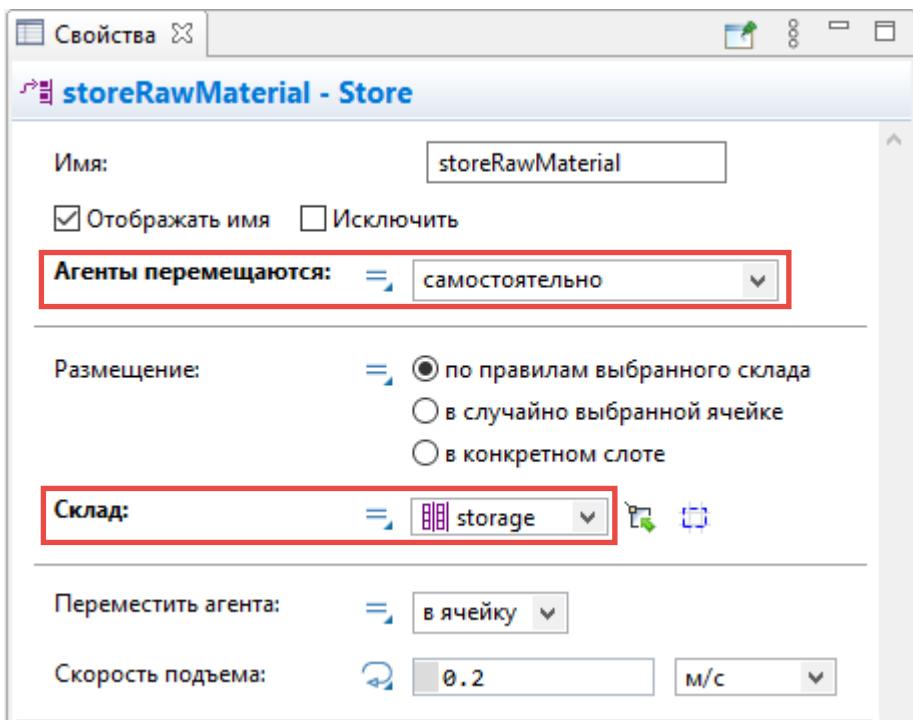
22. Переключитесь на палитру **Библиотека производственных систем**  и перетащите блок **Store**  на диаграмму, поместив его сразу за блоком *sourcePallets*, чтобы они автоматически соединились друг с другом, как показано на следующем рисунке.



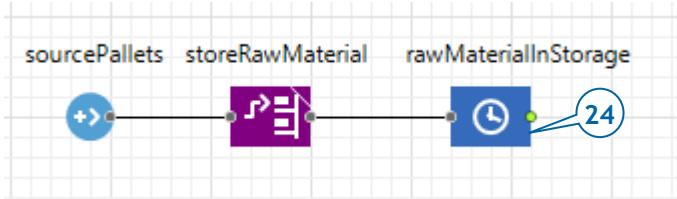
Блок **Store** моделирует помещение поддонов в заданные ячейки склада.

**23.** На странице свойств блока необходимо сделать следующее.

- В поле **Имя** введите *storeRawMaterial*.
- Чтобы мы смогли успешно запустить модель уже в конце первой фазы, давайте пока что выберем, что **Агенты перемещаются: самостоятельно**. В следующей фазе мы добавим в модель ресурсы (автопогрузчики), и изменим эту настройку на доставку с помощью ресурсов.
- В списке **Склад** выберите *storage*.



24. Чтобы промоделировать пребывание поддонов в стеллаже, добавьте блок Delay . Назовите этот блок *rawMaterialInStorage*.

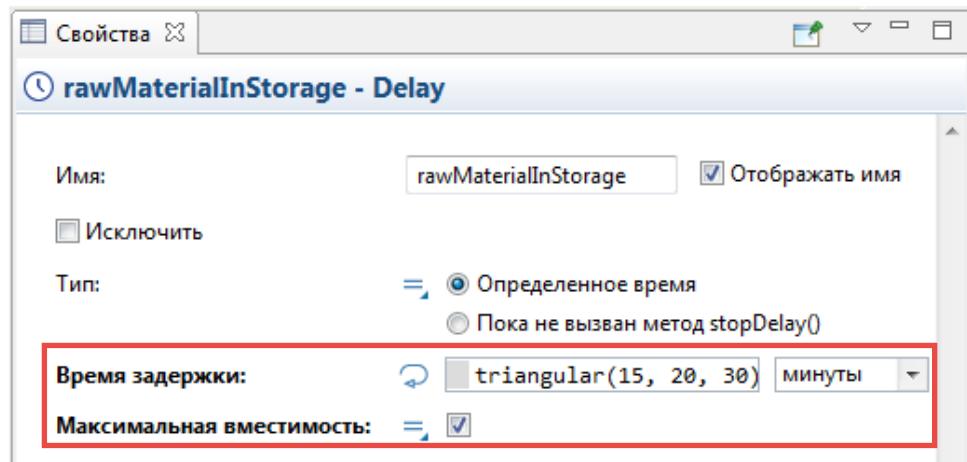


Как вы заметили, мы соединяем правый порт одного блока с левым портом следующего блока. У каждого блока имеется левый *входной порт* и правый *выходной порт*, и вы можете соединять входные порты только с выходными.

25. На странице свойств блока *rawMaterialInStorage* необходимо выполнить следующее.

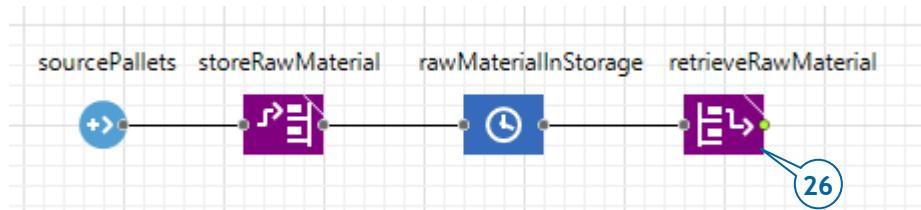
- В поле **Время задержки** введите *triangular(15, 20, 30)*. В расположеннном справа списке выберите **минуты**.

- b. Установите флажок **Максимальная вместимость**, означающий, что в этом блоке (моделирующем пребывание поддонов на складе) может одновременно находиться сразу несколько (условно неограниченное количество) агентов-поддонов.



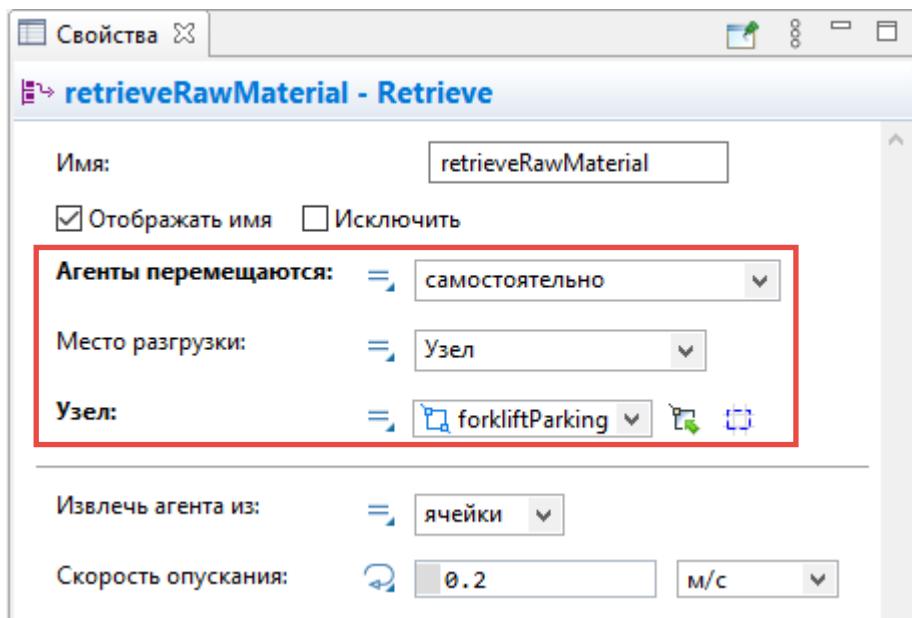
- 26.** Добавьте блок **Retrieve** , подсоедините его к диаграмме процесса и назовите этот блок *retrieveRawMaterial*.

В нашей модели блок **Retrieve** извлекает поддон из ячейки стеллажа и перемещает в заданное место.

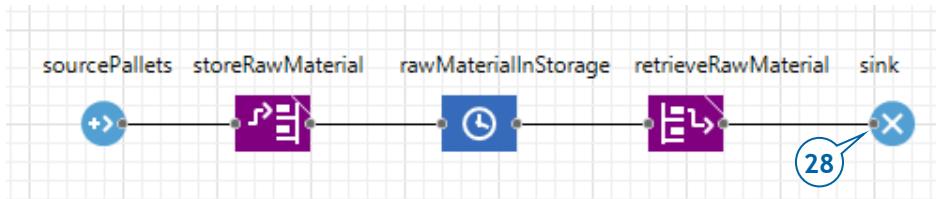


- 27.** На странице свойств блока *retrieveRawMaterial* необходимо выполнить следующее:

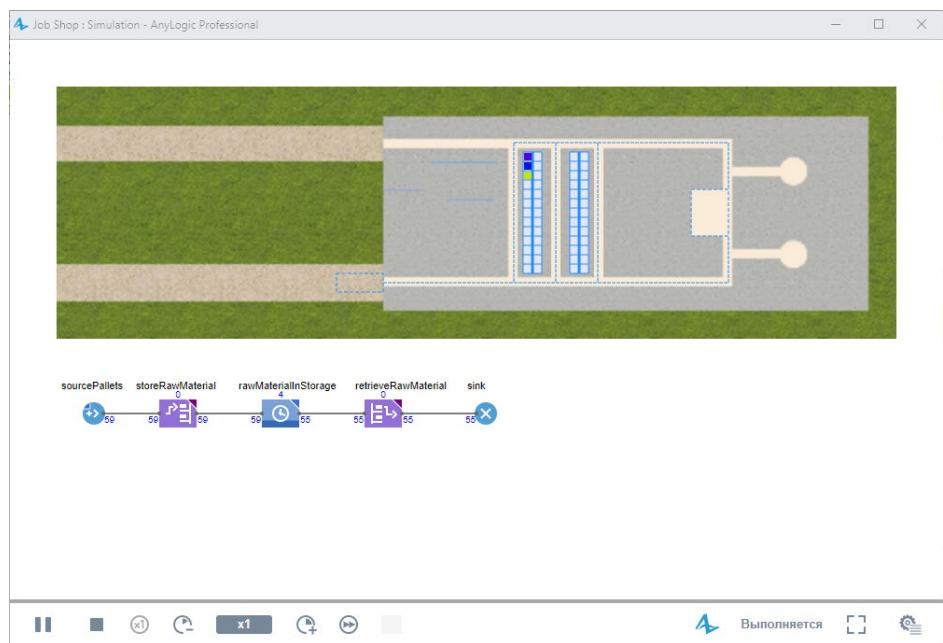
- Как и ранее в блоке **Store**, выберите опцию **Агенты перемещаются: самостоятельно**.
- В списке **Узел** выберите *forkliftParking* - тот узел сети, куда автопогрузчики будут доставлять поддоны.



28. Добавьте блок **Sink** Блок **Sink** уничтожает поступающих агентов и обычно выступает в качестве конечной точки диаграммы процесса.



29. Мы завершили создание этой простой модели. Теперь вы можете запустить модель (эксперимент *Job Shop / Simulation*) и понаблюдать за ее поведением. Вы увидите, как поддоны доставляются в цех и затем помещаются для хранения на склад.



## Фаза 2. Добавление ресурсов

Продолжим создание нашей модели. Теперь мы добавим ресурсы (автопогрузчики), с помощью которых будет производиться как помещение поддонов в стеллаж, так и их последующее перемещение в производственную зону.

### Ресурсы

Ресурсами называются объекты, используемые агентами для выполнения определенной операции. В случае необходимости агент должен получить ресурс, выполнить операцию, а затем освободить ресурс.

Примеры ресурсов:

- В модели больницы: врачи, медицинские сестры, оборудование, кресла-каталки;
- В модели цепочки поставок: автотранспортные средства и контейнеры;
- В модели склада: автопогрузчики и рабочие.

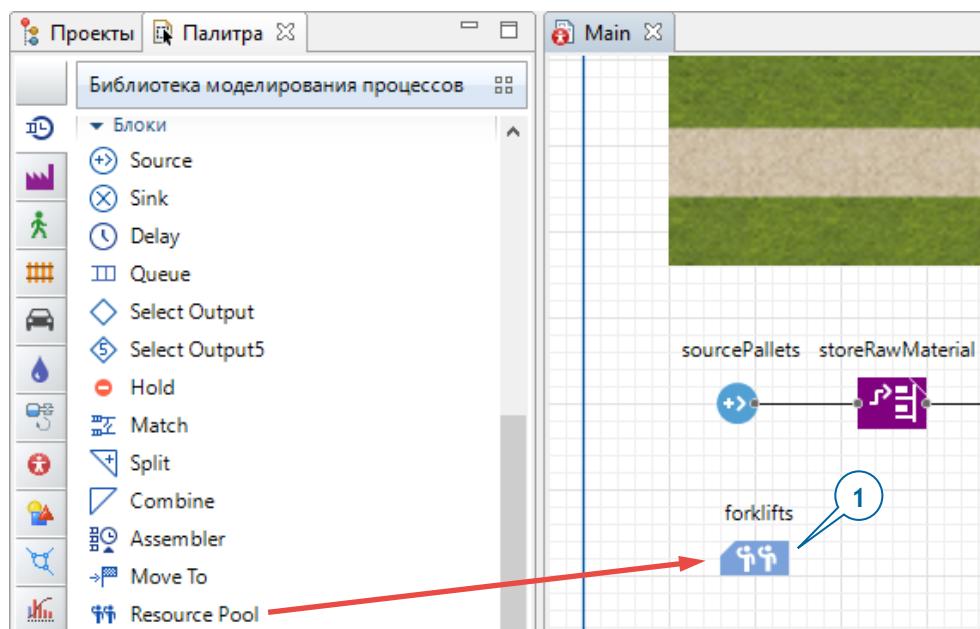
Ресурсы в AnyLogic бывают трех типов: *статические*, *перемещаемые* и *движущиеся*.

- Статические ресурсы привязаны к определенному местоположению и не могут перемещаться ни самостоятельно, ни принудительно.
- Движущиеся ресурсы могут перемещаться самостоятельно.
- Перемещаемые ресурсы могут перемещаться агентами или движущимися ресурсами.

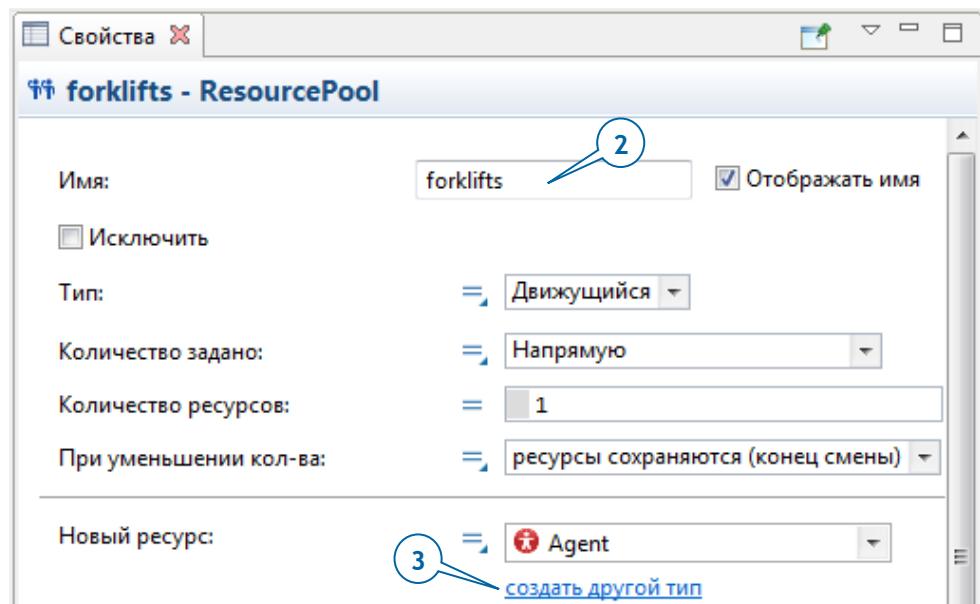
Ресурсами нашей модели являются автопогрузчики, перемещающие поддоны из зоны разгрузки в стеллаж, а затем доставляющие поддоны из стеллажа в производственную зону.

Каждый набор ресурсов задается в AnyLogic блоком библиотеки **ResourcePool** .

1. Перетащите блок **ResourcePool**  из палитры  **Библиотека моделирования процессов** на диаграмму *Main*. Подсоединять этот блок к диаграмме процесса не требуется.



2. Присвойте блоку имя *forklifts* (автопогрузчики).

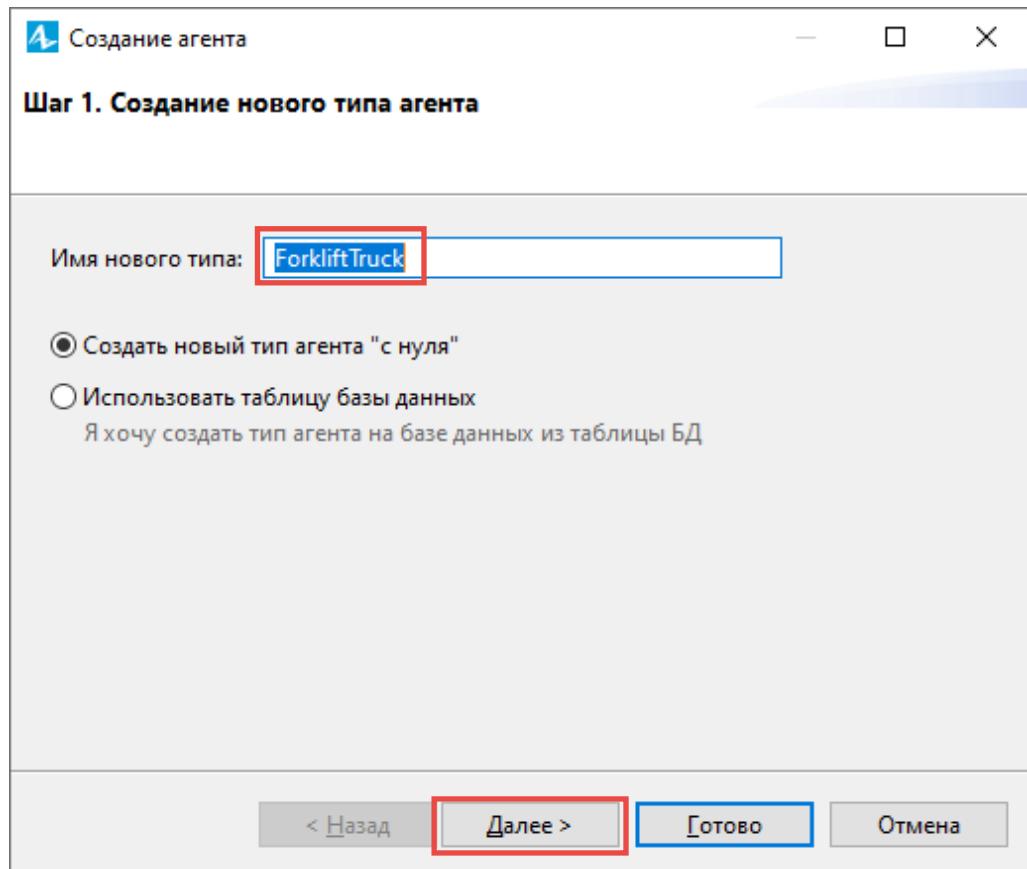


3. На странице свойств блока *forklifts* щелкните по метке *создать другой тип*. Таким способом мы создадим в модели новый тип ресурса. У каждого типа

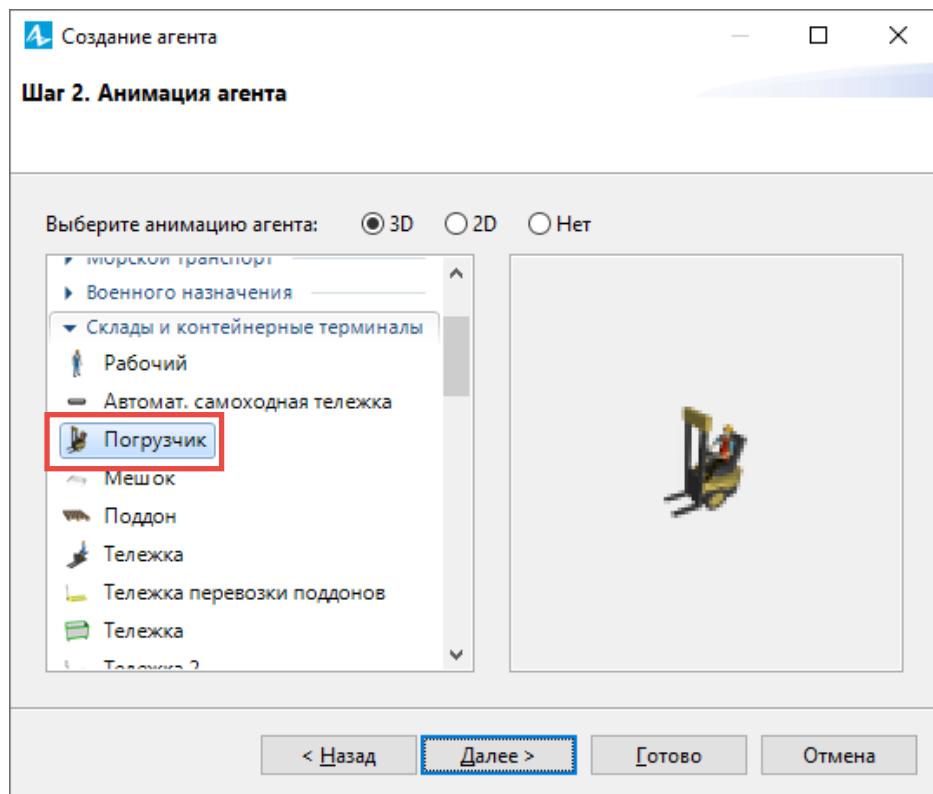
ресурса есть своя графическая диаграмма, на которой вы можете нарисовать анимацию этого ресурса, а также задать специфические характеристики этого ресурса с помощью параметров, переменных и функций.

4. В мастере **Создание агента**:

- a. В поле **Имя нового типа** введите *ForkliftTruck*.
- b. Перейдите к следующей странице мастера, щелкнув по кнопке **Далее**.

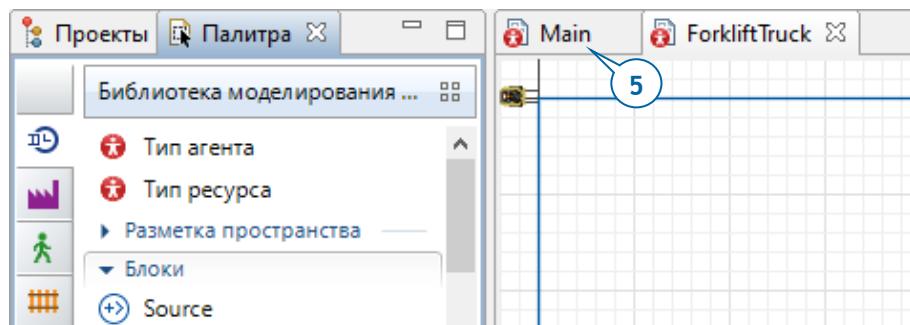


- c. В списке в левой части мастера раскройте раздел **Склады и контейнерные терминалы** и выберите картинку **Погрузчик**.
- d. Щелкните по кнопке **Готово**.



Откроется диаграмма типа ресурса *ForkliftTruck*, на которой будет присутствовать фигура анимации, выбранная вами в мастере.

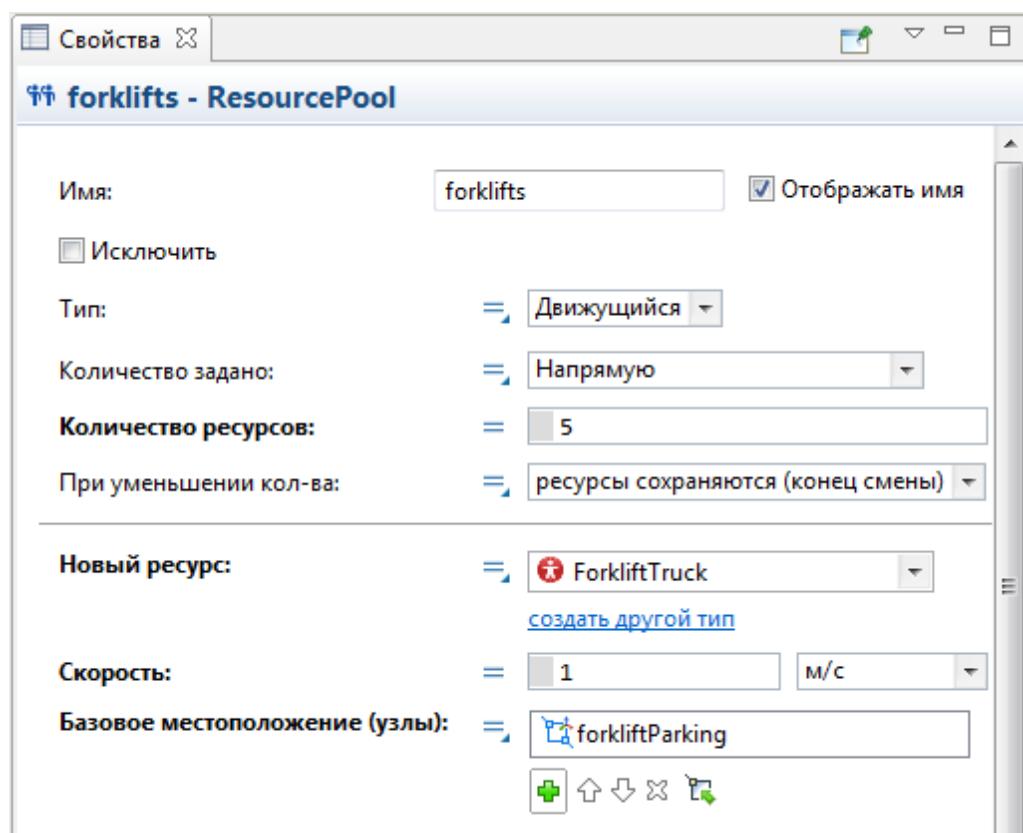
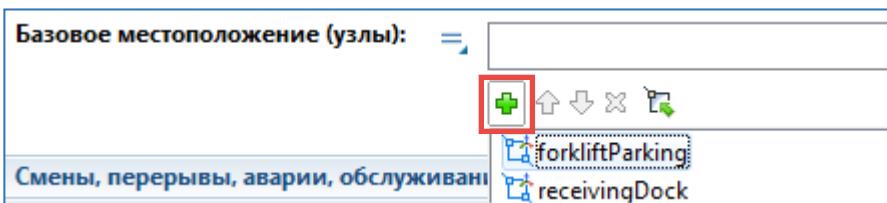
##### 5. Откройте диаграмму *Main*, щелкнув по вкладке *Main*.



Вы увидите, что в поле **Новый ресурс** блока **ResourcePool** будет выбран тип ресурса *ForkliftTruck*.

6. Измените остальные параметры блока *forklifts*, задающего набор ресурсов:

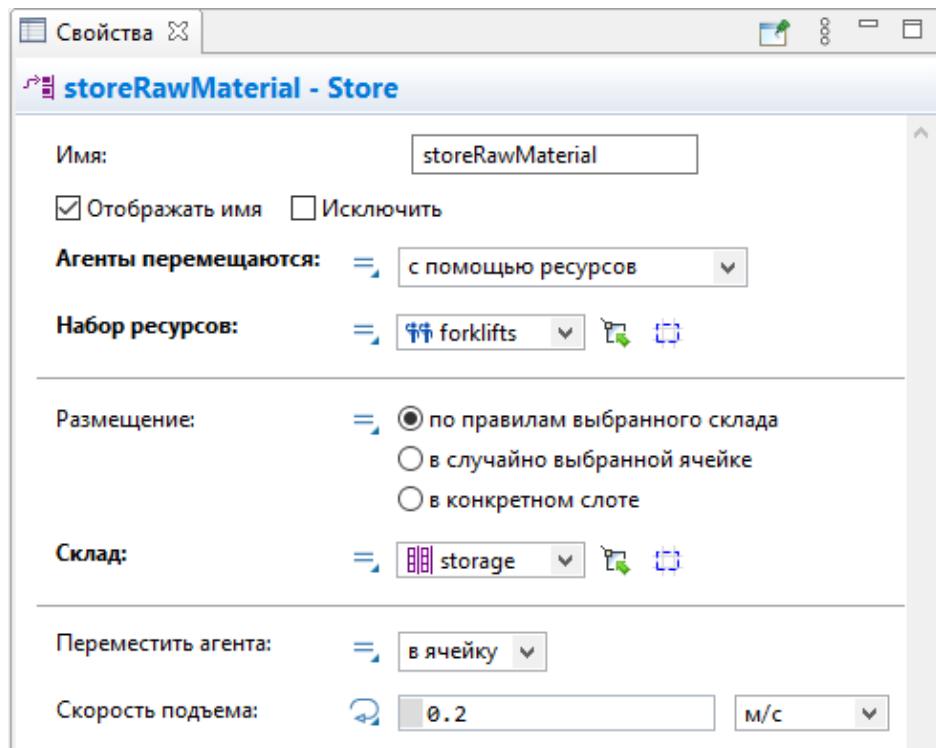
- В поле **Количество ресурсов** введите **5** - количество автопогрузчиков в моделируемом нами цеху.
- В поле **Скорость** введите **1** и выберите из списка справа **м/с**.
- В области **Базовое местоположение (узлы)** выберите узел *forkliftParking*. Щелкните по кнопке  , а затем выберите из списка узлов *forkliftParking*.



Мы задали ресурсы, и теперь нам нужно сделать так, чтобы блоки диаграммы процесса нашей модели использовали эти ресурсы.

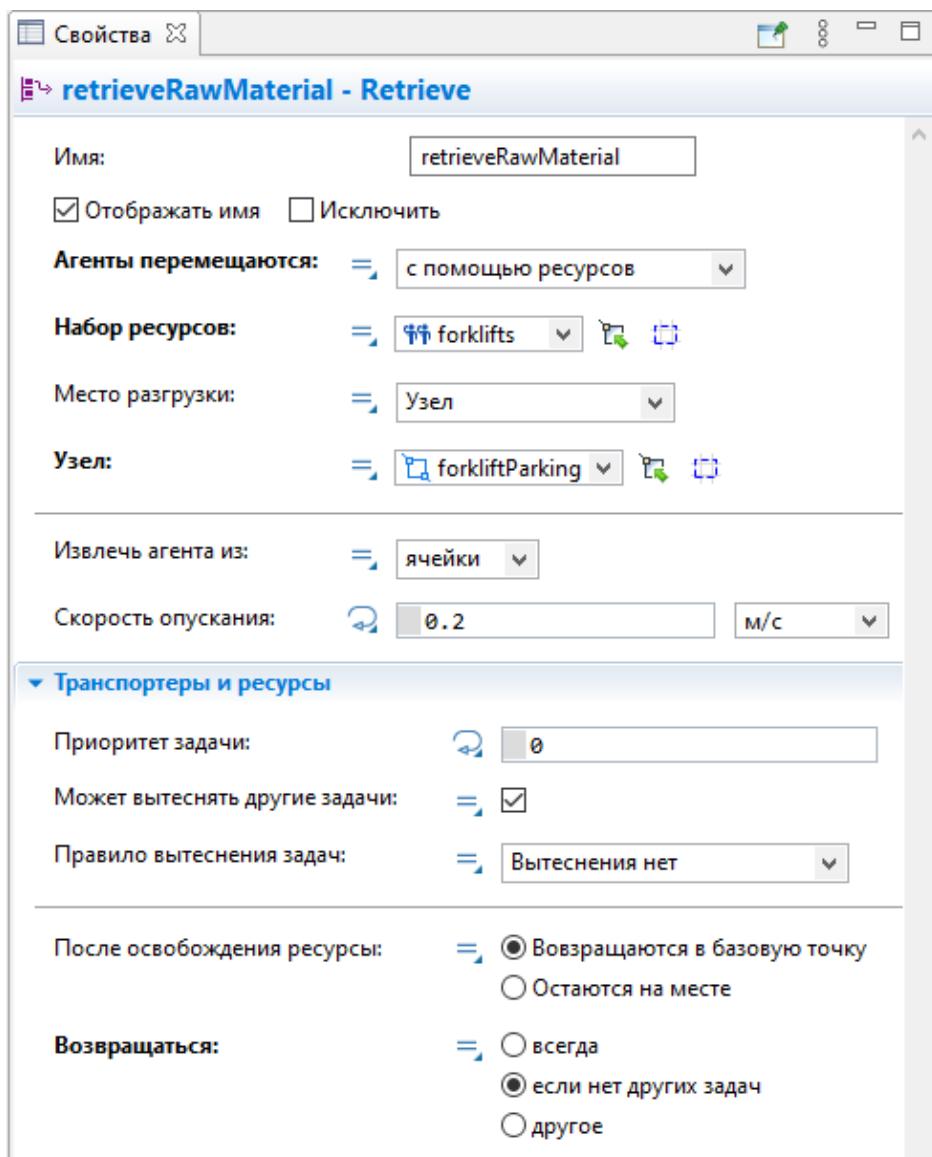
7. На странице свойств блока *storeRawMaterial*/ нужно выполнить следующее:

- В списке **Агенты перемещаются** выберите опцию **с помощью ресурсов**.
- В списке **Набор ресурсов** выберите **forklifts**, при этом блок диаграммы процесса будет использовать ресурс выбранного типа (в нашем случае - автопогрузчик) для перемещения поддона.



8. На странице свойств блока *retrieveRawMaterial*/ выполните следующее:

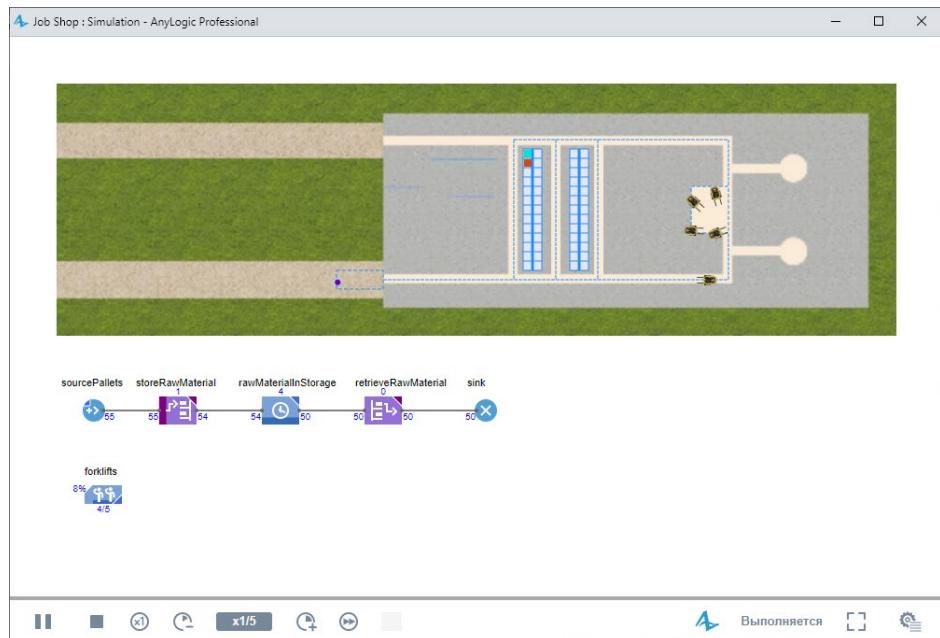
- В списке **Агенты перемещаются** выберите опцию **с помощью ресурсов**.
- В списке **Набор ресурсов** выберите **forklifts**.
- В секции свойств **Транспортеры и ресурсы** в списке **Возвращаться** выберите опцию **если нет других задач**.



При перемещении агента блок **Store** захватывает свободный ресурс (автопогрузчик), перемещает его в место расположения агента (поддона), прикрепляет ресурс к агенту, перемещает агента с помощью ресурса в ячейку склада, а затем освобождает ресурс. Схожим образом ведет себя и блок **Retrieve** (разница в том, что он извлекает поддоны со склада).

9. Запустите модель.

Вы увидите, как автопогрузчики забирают поддоны из зоны разгрузки и помещают их на склад. По истечении небольшой задержки они перемещают поддоны в зону парковки автопогрузчиков, при попадании в которую поддоны исчезают.



## Фаза 3. Создание трехмерной анимации

Давайте теперь добавим трехмерную анимацию моделируемого нами процесса.

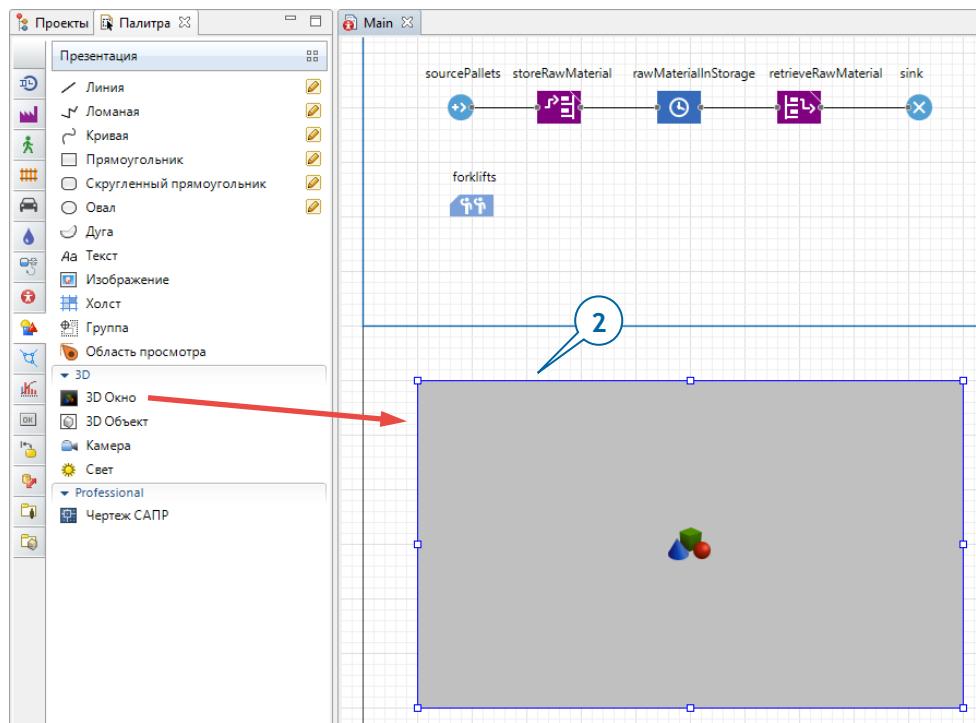
Начнем с добавления камеры.

### Камера

Камера "снимает" ту сцену, которую вы видите в окне трехмерной анимации. Добавив в вашу модель камеру, вы можете выбрать, что именно вы хотите видеть в 3D окне.

Вы можете добавить несколько камер, это позволит вам наблюдать за различными зонами моделируемой системы (например, разными цехами фабрики) или за одним и тем же объектом, но с разных точек обзора. Если вы используете несколько камер, то по ходу выполнения модели вы можете легко переключаться с одного изображения на другое.

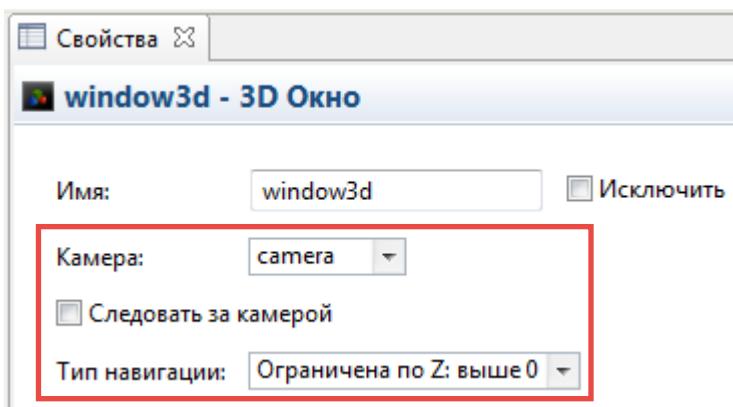
1. Перетащите элемент Камера  из палитры Презентация  на диаграмму *Main*. Поместите камеру таким образом, чтобы она "смотрела" на анимацию фабричного цеха.
2. Из этой же палитры, перетащите на диаграмму *Main* 3D окно  . Поместите его под диаграммой процесса.



## 3D окно

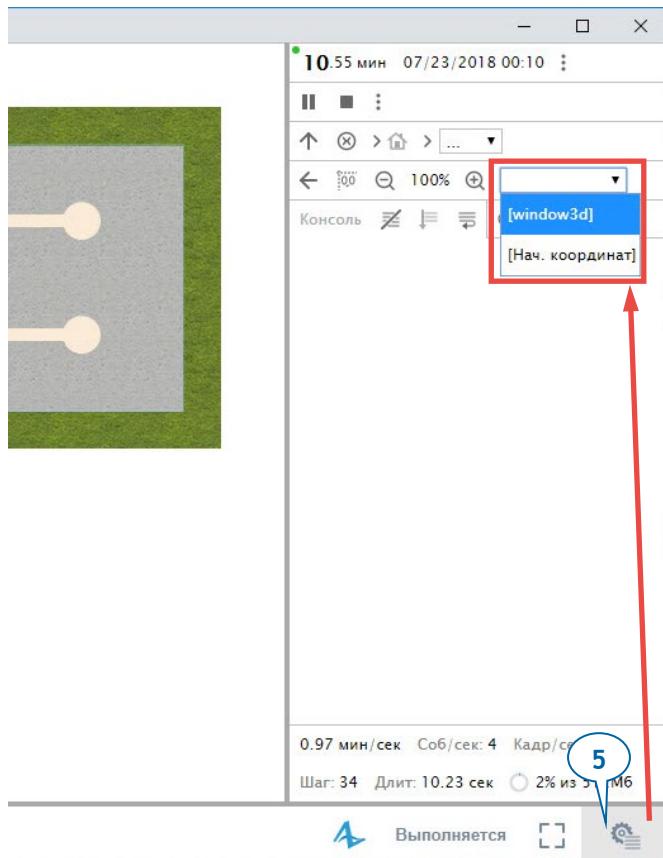
При необходимости вы можете добавить несколько 3D окон для обзора анимации с разных точек обзора.

3. В панели **Свойства** 3D окна выберите в списке **Камера** элемент **camera**. Теперь наша камера будет отвечать за формирование картинки для 3D окна.
4. Чтобы предотвратить съемку "из-под пола", выберите из списка **Тип навигации** опцию **Ограничена по Z: выше 0**.



5. Запустите модель. При создании 3D окна AnyLogic автоматически добавляет область просмотра 3D сцены, благодаря чему в процессе выполнения модели можно легко перейти к просмотру 3D анимации. Чтобы переключиться на 3D анимацию, щелкните по кнопке панели разработчика **Выбрать область и показать** и выберите из раскрывающегося списка область просмотра трехмерной анимации [window3d].

При этом окно трехмерной анимации будет развернуто до размера окна модели.

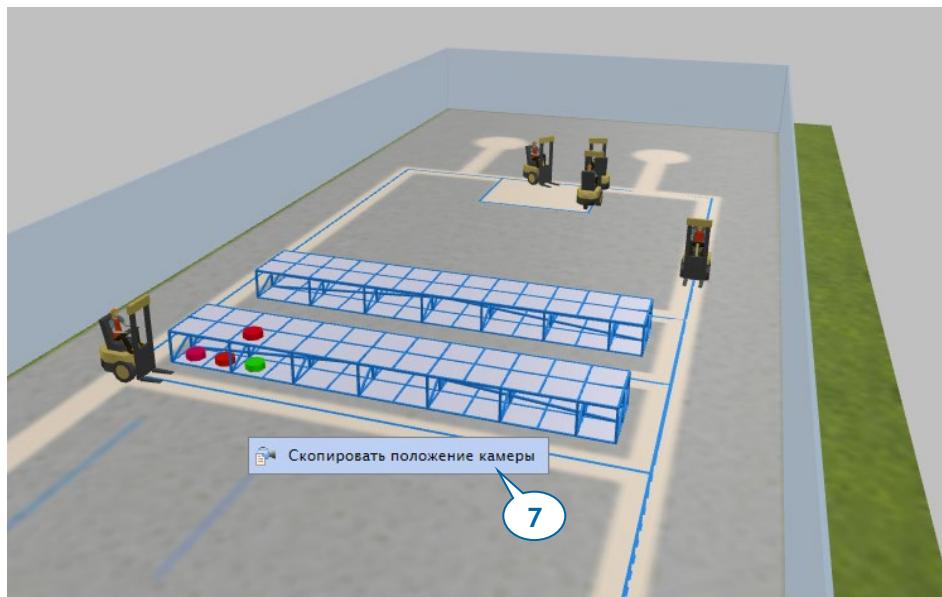


**6.** Вы можете перемещаться по сцене трехмерной анимации в процессе выполнения модели с помощью следующих действий:

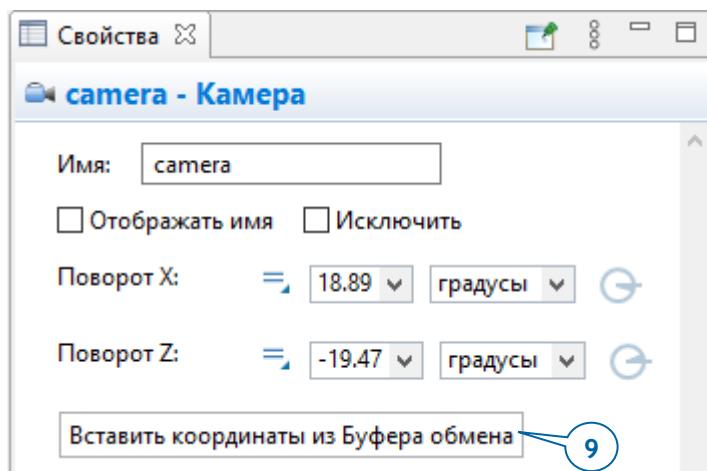
- Переместить камеру влево, вправо, вперед или назад можно с помощью перемещения мыши в нужном направлении (с нажатой левой кнопкой мыши).
- Приблизить камеру к центру сцены или удалить ее от центра можно вращением колеса мыши.
- Повернуть изображение относительно камеры можно перемещением мыши, предварительно нажав и держа нажатой клавишу ALT и левую кнопку мыши.

**7.** С помощью приведенных выше команд задайте подходящий вам ракурс съемки изображения. Щелкните правой кнопкой мыши (для Mac OS: CTRL +

щелчок левой кнопкой) в пределах трехмерного изображения и выберите из контекстного меню команду **Скопировать положение камеры**.



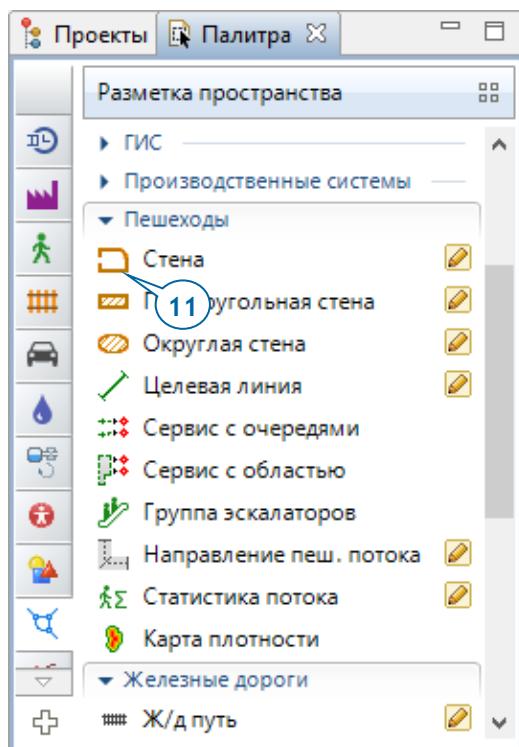
8. Закройте окно модели.
9. На странице свойств камеры установите выбранное на предыдущем шаге положение камеры, щелкнув по кнопке **Вставить координаты из Буфера обмена**.



Если вам не удается найти камеру на холсте, ее можно легко найти в дереве **Проекты**. В нем камера *camera* будет показана в ветви **Презентация** агента *Main*.

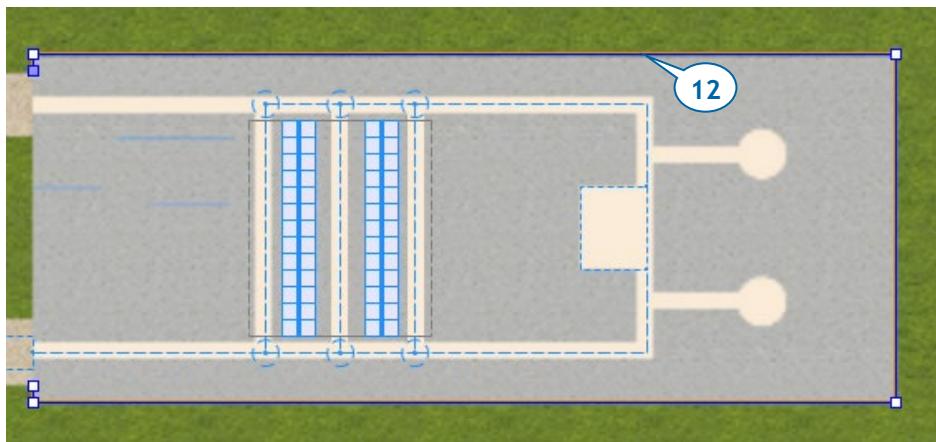
**10.** Запустите модель. Убедитесь, что теперь 3D анимация показывается с заданного вами ракурса, после чего закройте окно модели.

**11.** Раскройте секцию **Пешеходы** палитры **Разметка пространства**, а затем сделайте двойной щелчок по значку элемента **Стена** . При этом активируется режим рисования стены.



**12.** Чтобы нарисовать в графическом редакторе стену вокруг рабочей зоны производства, нужно сделать следующее.

- Щелкните в той точке, с которой вы хотите начать рисование стены.
- Чтобы добавить угол стены, щелкните в соответствующей точке.
- В той точке, где вы хотите завершить рисование стены, сделайте двойной щелчок.



**13.** Чтобы изменить цвет стены (или задать для нее определенную текстуру), необходимо выполнить следующее.

- Откройте раздел свойств стены **Внешний вид**.
- Щелкните по выпадающему списку **Цвет** и выберите **Другие цвета...** из появившегося списка.
- С помощью диалогового окна **Цвета** выберите на палитре или в спектре нужный цвет стены.

Вы также можете установить уровень прозрачности (с помощью ползунка **Прозрачность** в диалоговом окне **Цвета**) или применить к стене любую предлагаемую текстуру (выбрав в меню цвета элемент **Текстуры...**).

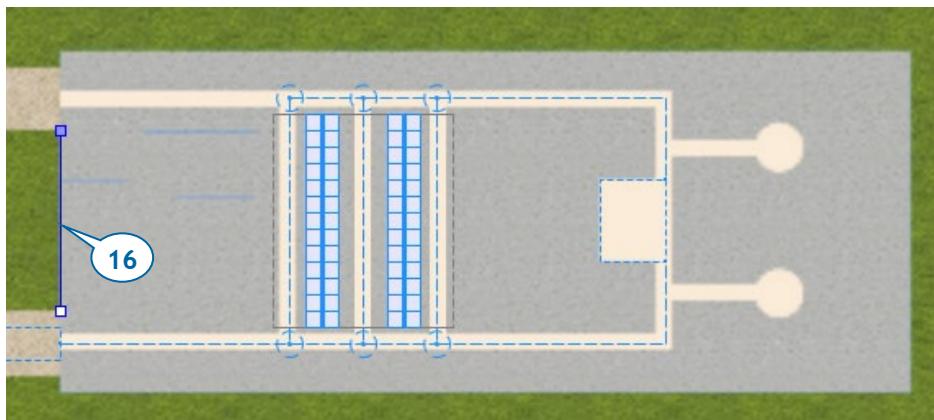
В этом разделе мы использовали стены исключительно для анимационных целей. На самом деле стены чаще используются в пешеходном моделировании для задания стен и других препятствий для движения людей.

**14.** Сделайте стену тоньше, выбрав в поле **Толщина линии** значение **1 pt**.

**15.** Перейдите в раздел свойств стены **Местоположение и размер** и измените значение параметра **Z-Высота** на **40**.

AnyLogic автоматически устанавливает высоту стены равной 20 пикселям, мы же увеличиваем эту высоту до 40 пикселей.

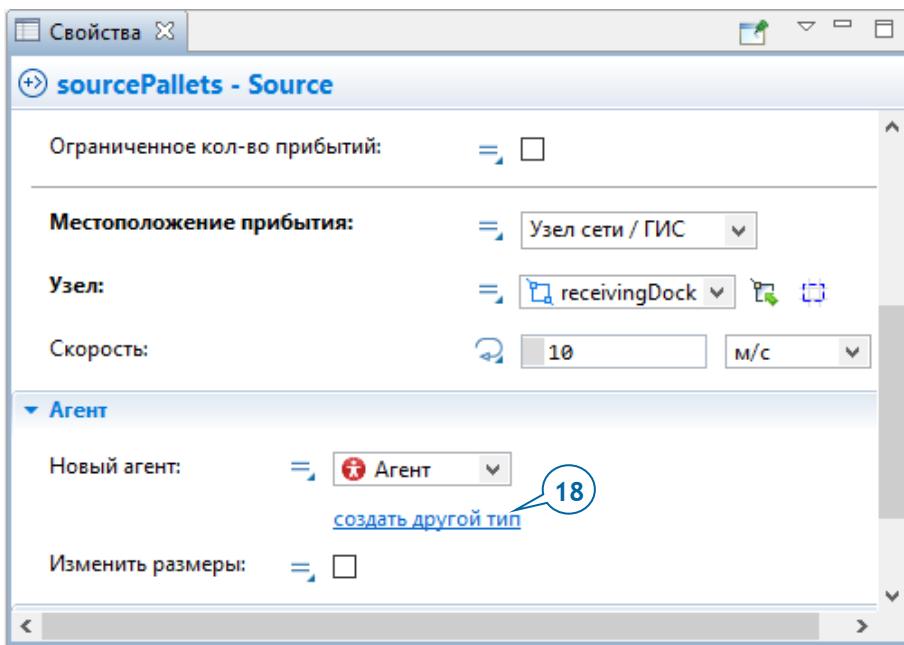
16. Нарисуйте еще одну стену между выходами, а затем задайте у нее свойства, аналогичные свойствам первой стены.



17. Запустите модель и понаблюдайте за трехмерной анимацией процесса.

Вы увидите, что поддоны отображаются на анимации фигурами случайно выбранных цветов. Чтобы выбрать в качестве фигуры анимации поддона его трехмерную модель, нужно будет создать для поддона специальный тип агента, на чьей диаграмме мы и сможем задать нужную нам анимацию.

18. В разделе свойств **Агент** блока *sourcePallets* под списком **Новый агент** щелкните по ссылке **создать другой тип**.



19. В мастере **Создание агентов** сделайте следующее.

- В поле **Имя нового типа** введите *Pallet*.
- Перейдите к следующей странице мастера, щелкнув по кнопке **Далее**.
- В списке в левой части мастера раскройте раздел **Склады и контейнерные терминалы** и выберите в списке картинку **Поддон**.
- Щелкните по кнопке **Готово**.

AnyLogic создаст тип агента *Pallet* и откроет в редакторе графическую диаграмму этого типа. В левом верхнем углу холста (а именно, в точке начала координат) вы увидите фигуру анимации, выбранную нами только что в мастере создания типа агента.

20. В свойствах типа агента *Pallet*, раскройте секцию **Агент в диаграмме процесса** и выберите опцию **Материальный объект** из списка **Использовать в диаграмме процесса как**. Теперь у агентов типа *Pallet* будет дополнительная функциональность, которая может быть полезна при обработке агентов этого типа в блоках Библиотеки производственных систем. В частности, у агентов–поддонов появятся явно заданные

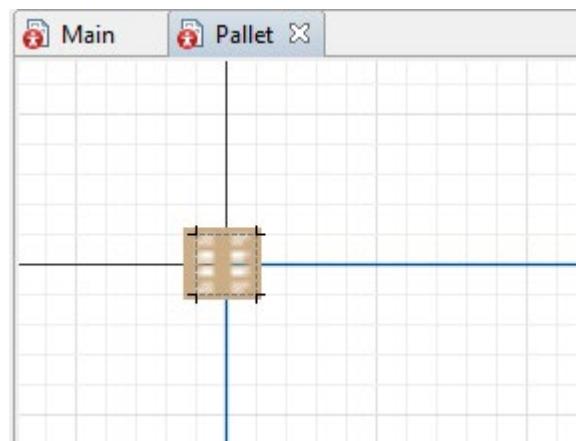
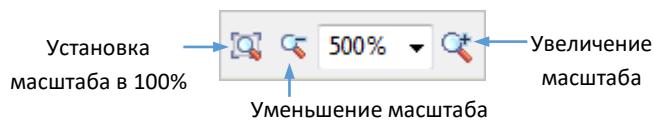
размерности (они задаются в свойствах **Длина**, **Ширина** и **Высота** в разделе свойств **Размеры и движение**).

Нашим следующим шагом будет добавление фигуры коробки поверх фигуры анимации поддона. Поскольку мы будем работать с объектами небольшого размера, сначала давайте увеличим масштаб диаграммы.

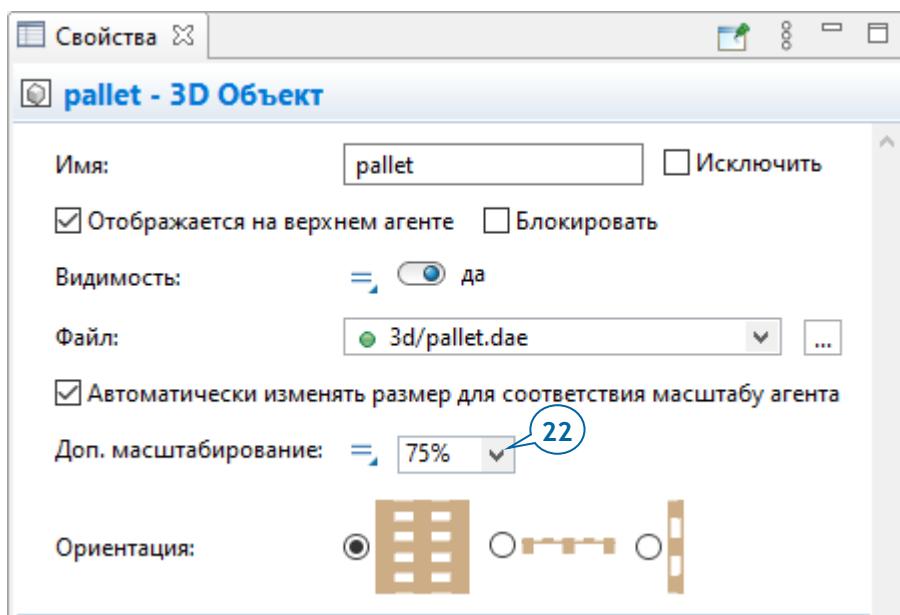
21. С помощью панели инструментов **Масштаб** увеличьте масштаб графической диаграммы типа агента *Pallet* до 300%, а затем переместите холст диаграммы вправо и вниз, чтобы увидеть начало координат, в котором расположена фигура анимации поддона.

### *Увеличение или уменьшение масштаба изображения*

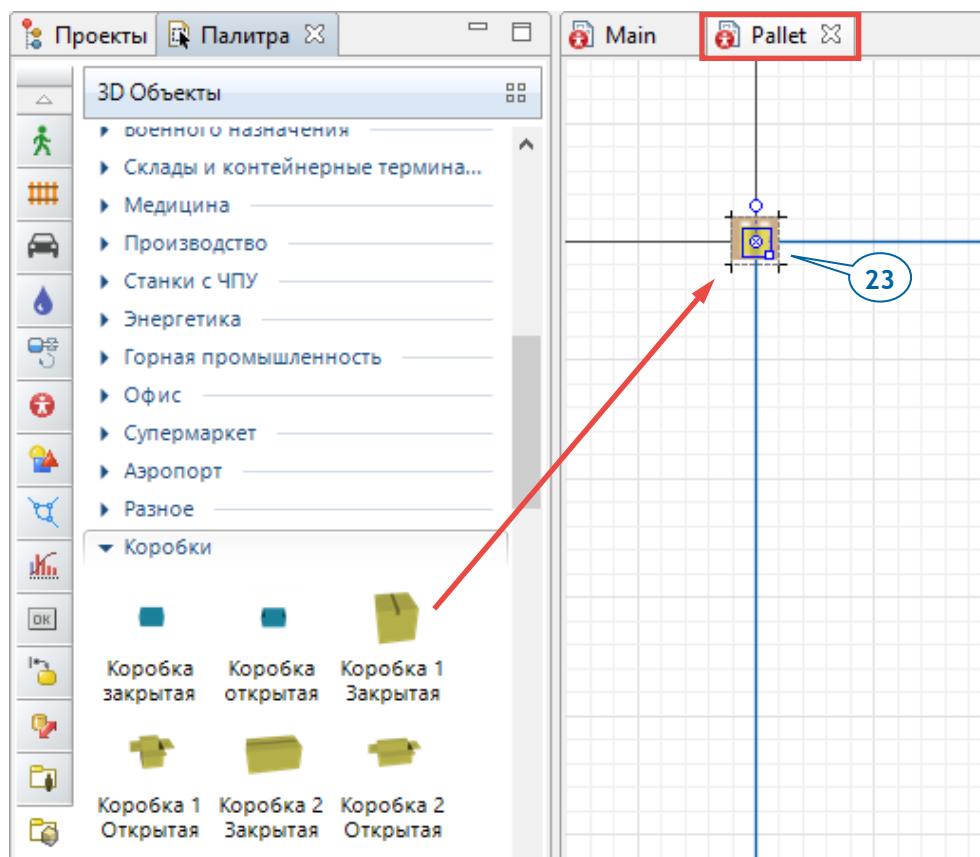
Панель инструментов **Масштаб** в AnyLogic позволяет изменять масштаб графической диаграммы:



22. Поскольку фигура поддона кажется слишком большой и не помещается в соответствующий заданным размерностям материального объекта размер (пунктирный прямоугольник с выделенными углами), уменьшим ее, задав в свойствах фигуры **Доп. масштабирование: 75%**.



23. Теперь добавьте фигуру анимации коробки поверх фигуры анимации поддона:
- Откройте палитру 3D объекты (это последняя палитра в списке), найдите и раскройте секцию палитры Коробки.
  - Перетащите объект Коробка 1 Закрытая на фигуру поддона.



**24.** Выделите фигуру коробки и задайте для нее **Доп. масштабирование: 125%**.

**25.** Раскройте раздел свойств **Расположение** и измените координату Z коробки на 2.

Это изменение необходимо потому, что нам нужно поместить коробки на поддоны, а высота каждого поддона - около двух пикселей.

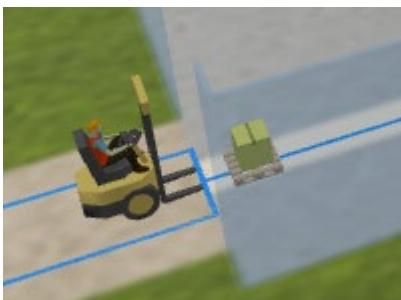
**26.** Теперь мы можем вернуться к исходному масштабу диаграммы, щелкнув по кнопке панели инструментов **100%**.

**27.** Вернитесь на диаграмму *Main*.

Если вы откроете свойства блока *sourcePallets*, то вы увидите, что в свойстве **Новый агент** выбран тип *Pallet*. Это значит, что данный блок будет создавать агентов типа *Pallet*.

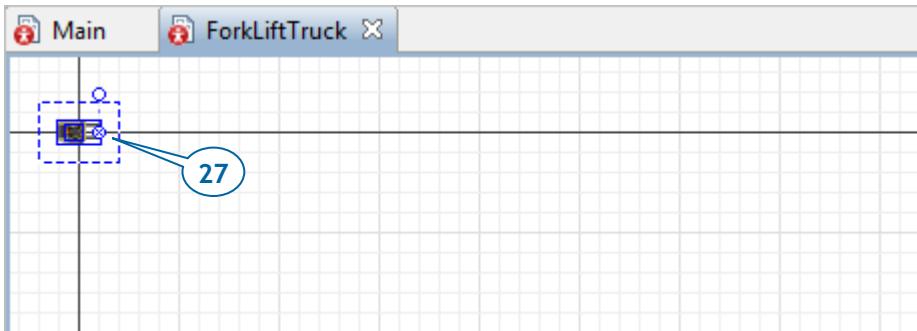
**28.** Запустите модель.

Вы увидите, что поддоны отображаются натуралистичными трехмерными моделями. Однако если вы увеличите масштаб трехмерного изображения, то вы заметите, что поддоны расположены чуть в стороне от вил погрузчиков.

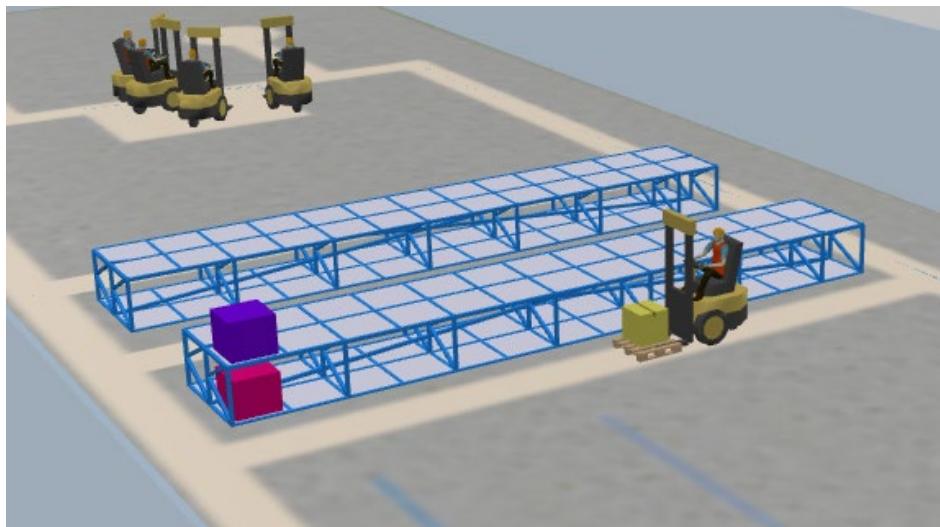


Давайте устраним эту неточность.

**29.** В панели **Проекты**, сделайте двойной щелчок по элементу *ForkliftTruck*. При этом откроется диаграмма этого типа агента. Переместите рисунок *forkliftWithWorker* на одну ячейку вправо.

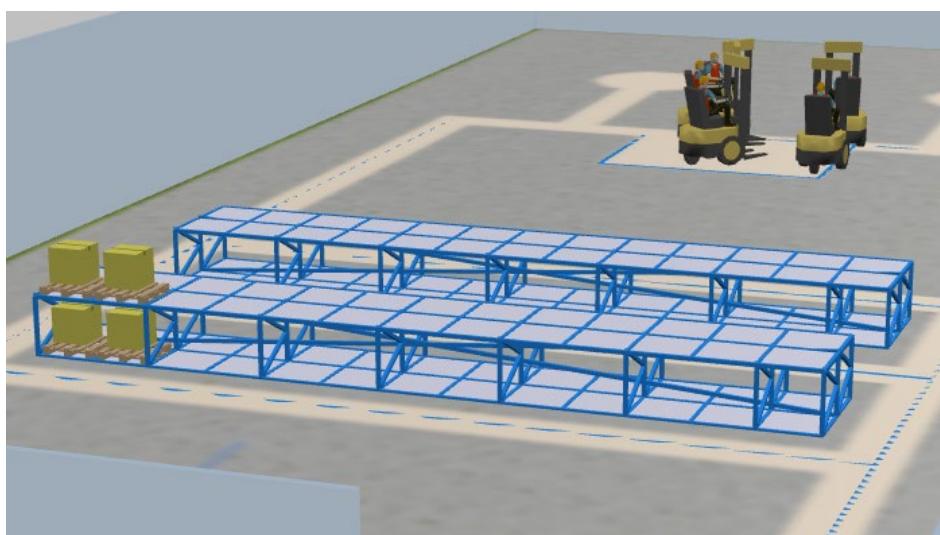


**30.** Запустите модель. Теперь на анимации модели поддоны будут помещаться точно поверх вил автопогрузчика.



В то же время, во время нахождения на стеллажах поддоны с коробками отображаются не трехмерными моделями, а кубиками разных цветов. Это сделано специально для повышения производительности модели, но при желании эту настройку можно отключить.

31. Для этого выберите фигуру склада, и в секции свойств **Внешний вид** выберите в списке **Анимация заполненных ячеек** опцию **анимация агента**.
32. Запустите модель и вы увидите, что теперь объекты визуализируются трехмерными моделями и находясь на стеллаже склада.



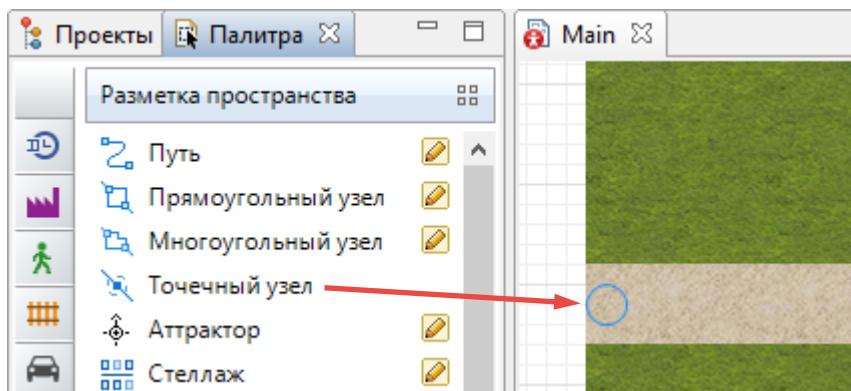
## Фаза 4. Моделирование доставки поддонов фурами

На этом этапе обучения мы добавим фуры, доставляющие поддоны на завод. Начнем с создания еще одного типа агента, задающего фуру.

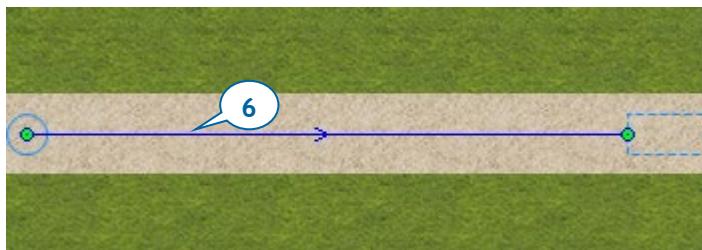
1. Перетащите элемент **Тип агента**  из палитры **Библиотека моделирования процессов** на диаграмму *Main*.
2. В мастере **Создание агентов** выполните следующее.
  - a. В поле **Имя нового типа** введите *Truck*.
  - b. Перейдите к следующей странице мастера, щелкнув по кнопке **Далее**.
  - c. В списке фигур анимации раскройте раздел **Автодорожный транспорт** и выберите из списка фигуру **Фура**.
  - d. Щелкните по кнопке **Готово**.

Добавим в нашу сеть два новых элемента: узел, в котором будут появляться фуры, и путь, по которому они будут следовать до приемной зоны.

3. Откройте диаграмму *Main*.
4. Перетащите элемент **Точечный узел**  из палитры **Разметка пространства** на графическую диаграмму. Поместите его у начала подъездной дороги.

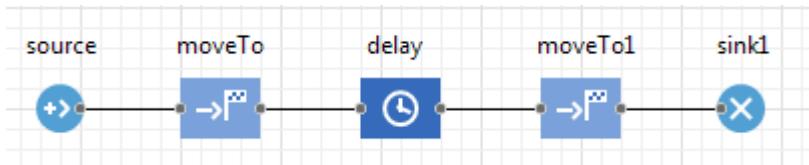


5. Назовите этот точечный узел *exitNode*.
6. Нарисуйте Путь  соединяющий *exitNode* с *receivingDock*.



7. Создайте еще одну диаграмму процесса, которая будет описывать логику движения фуры. Для этого добавьте на диаграмму *Main* новые блоки **Библиотеки моделирования процессов** и соедините в следующем порядке:

**Source** – **MoveTo** – **Delay** – **MoveTo** – **Sink**.



- В блоке **Source** создается фура.
- Первый блок **MoveTo** перемещает фуру ко въезду в цех.

Блок **MoveTo** перемещает агентов в заданный узел сети. Если к агенту в данный момент прикреплены ресурсы, то они будут перемещаться вместе с агентом.

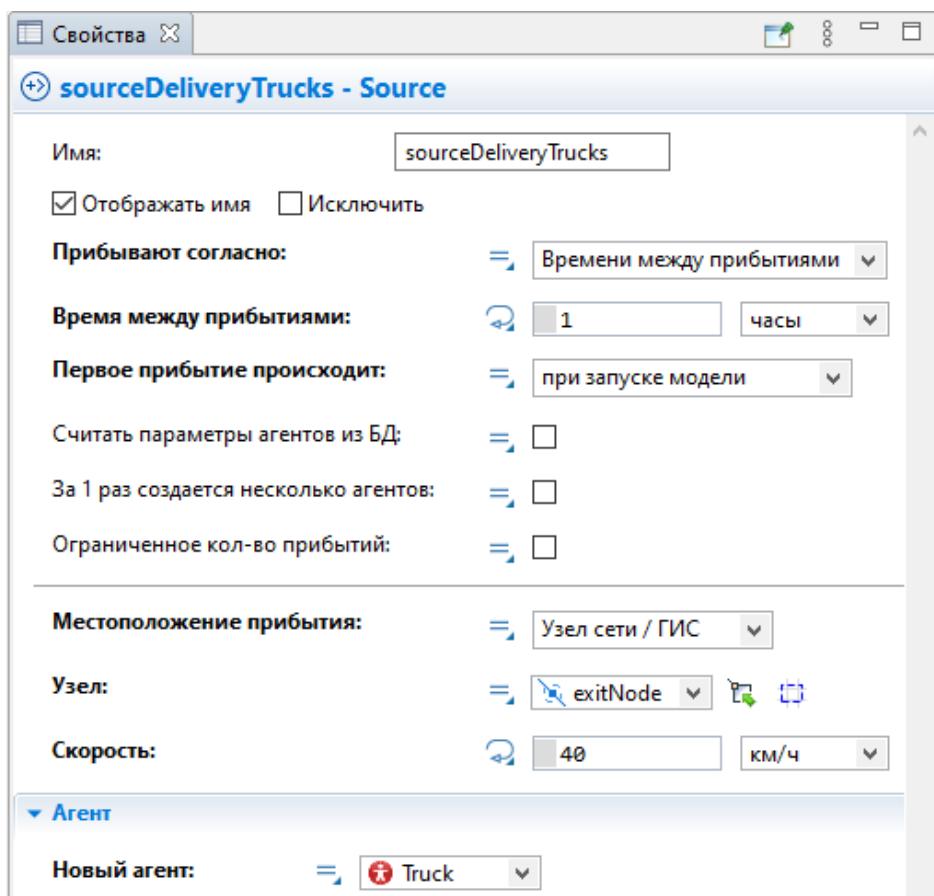
- Блок **Delay** моделирует разгрузку поддонов.
- Второй блок **MoveTo** моделирует отъезд фуры.
- Блок **Sink** удаляет фуры из модели.

8. Присвойте блоку **Source** имя *sourceDeliveryTrucks*.

9. Для того, чтобы агент типа *Truck* прибывал ко въезду на подъездную дорогу раз в час и с заданной скоростью, в свойствах блока *sourceDeliveryTrucks* выполните следующее.

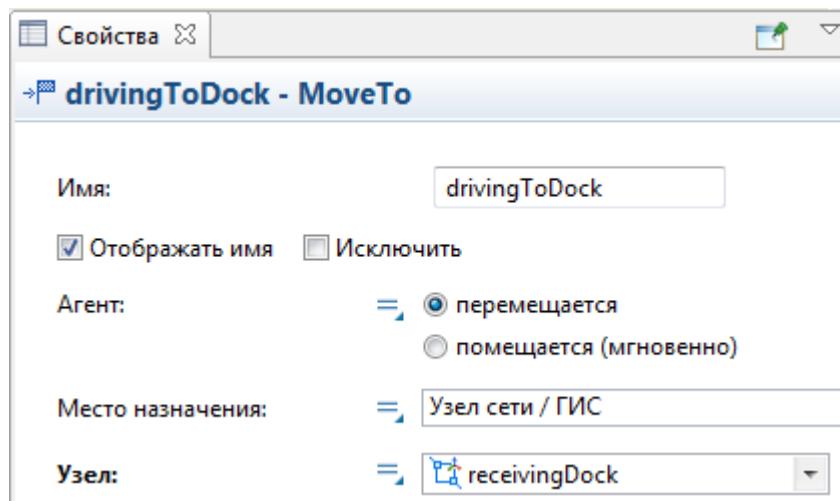
- В списке **Призывают согласно** выберите **Время между прибытиями**.
- В поле **Время между прибытиями** введите 1 и выберите из списка справа **часы**.

- c. В списке **Первое прибытие** выберите опцию **при запуске модели**. Тем самым, нам не надо будет ждать появления фуры в течение часа модельного времени – первая фура появится сразу в момент запуска модели.
- d. В списке **Местоположение прибытия** выберите **Узел сети / ГИС**.
- e. В списке **Узел** выберите **exitNode**.
- f. В поле **Скорость** введите **40** и выберите в списке справа **км/ч**.
- g. В разделе свойств **Агент** выберите **Truck** в списке **Новый агент**.

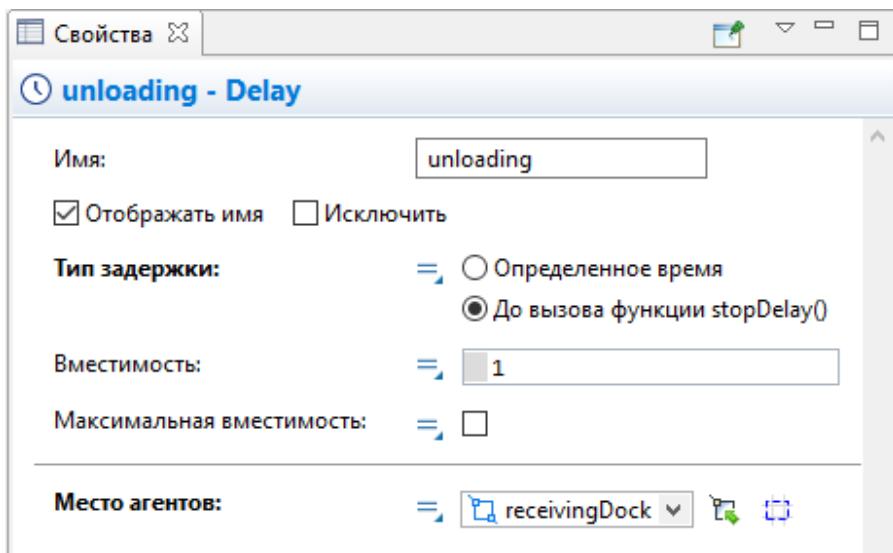


10. Присвойте первому блоку **MoveTo** имя *drivingToDock*.

11. Чтобы задать цель движения агента, в панели Свойства блока *drivingToDock* выберите в списке Узел имя узла *receivingDock*.



12. Переименуйте блок *Delay* в *unloading*.
13. В свойствах блока *unloading* необходимо выполнить следующее.
- У параметра Тип задержки нужно выбрать опцию До вызова функции *stopDelay()*.
  - В списке Место агентов выберите *receivingDock*.



Продолжительность операции определяется скоростью разгрузки и отвоза поддонов автопогрузчиками. Мы будем считать эту операцию выполненной, когда блок **Store** завершит установку поддонов на хранение, и смоделируем это изменением режима работы блока **Delay**.

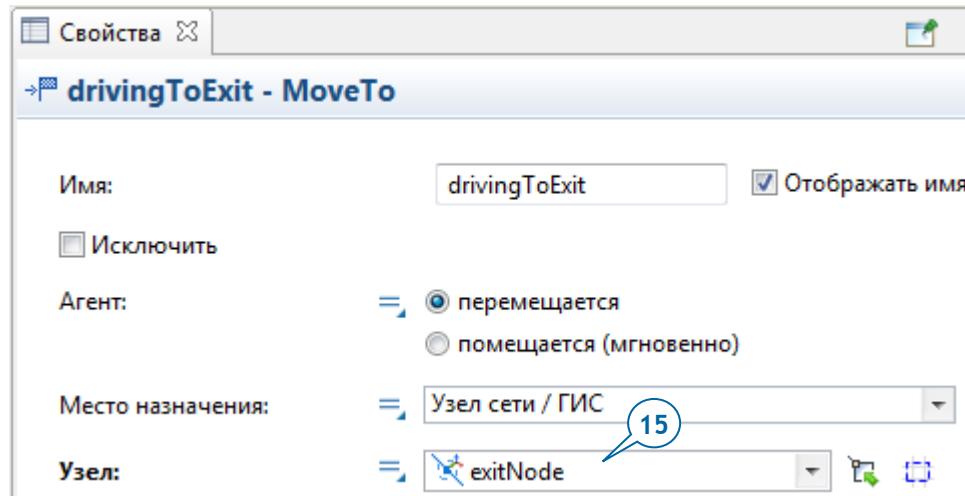
### Программное управление временем задержки

Как правило вы будете задавать **Время задержки** для работы блока **Delay**. Время может быть фиксированным, например, равным пяти минутам, или быть стохастическим (случайным), т.е. определяться функцией распределения вероятности, например: *triangular(1, 2, 6)*.

Вы также можете программно управлять длительностью операции и при необходимости прервать задержку, вызвав соответствующую функцию блока. Если вам необходимо прекратить ожидание всех агентов, находящихся в состоянии **Delay**, вызовите функцию блока *stopDelayForAll()*. Другая функция - *stopDelay(agent)* - завершает операцию и освобождает указанного агента.

**14.** Назовите второй блок **MoveTo** *drivingToExit*.

**15.** Чтобы задать конечный узел, в свойствах блока *drivingToExit* выберите в списке Узел вариант *exitNode*.

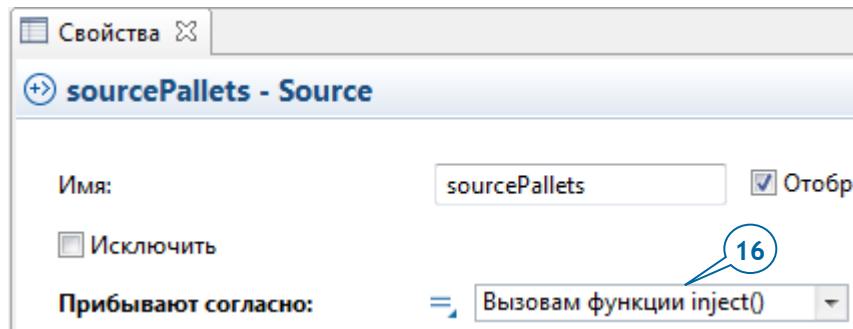


Два блока **Source** нашей модели создают агентов двух разных типов: фуры, появляющиеся каждый час, и поддоны, появляющиеся каждые пять минут. Поскольку нам нужно, чтобы поддоны появлялись при разгрузке фуры, мы изменим настройки того блока **Source**, который генерирует поддоны.

## Управление созданием агентов

Вы можете управлять созданием агентов блоком **Source** во время выполнения модели, генерируя требуемое количество агентов в определенные моменты времени жизни моделируемой системы. Для этого нужно выбрать в параметре блока **Прибывают согласно** опцию **Вызовом функции inject()** и вызывать функцию блока *inject(int n)*. Эта функция при ее вызове создаст заданное количество агентов. Вы указываете это количество с помощью аргумента функции, например: *sourcePallets.inject(12);*

- 16.** В свойствах блока *sourcePallets* выберите в списке **Прибывают согласно** опцию **Вызовом функции inject()**.

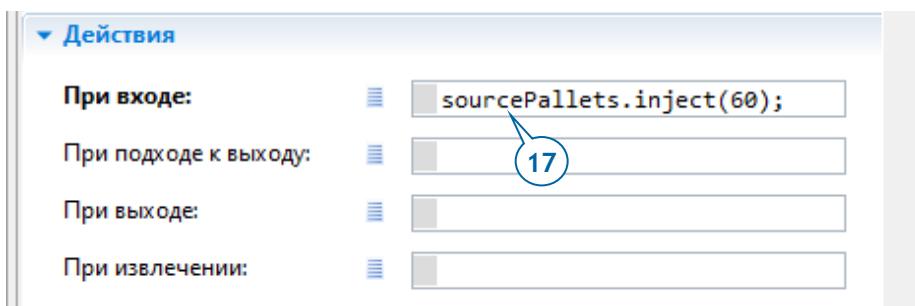


- 17.** Чтобы блок *sourcePallets* создавал поддоны при прибытии фуры в блок *unloading*, необходимо выполнить следующее.

- a. В свойствах блока *unloading* раскройте раздел **Действия**.
- b. В поле **При входе** введите следующее:

*sourcePallets.inject(60);*

Эта функция создаст 60 поддонов в момент начала разгрузки фуры.



Давайте сделаем так, чтобы первая фура появлялась при запуске модели, и нам не нужно было ждать ее появления целый час модельного времени.

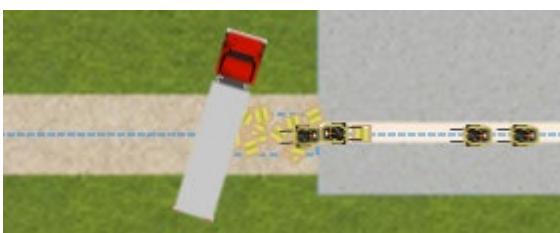
- 18.** В свойствах блока *storeRawMaterial* раскройте раздел **Действия**. В поле **При выходе** введите следующий код:

```
if( self.nWaitingForResource() == 0 )
unloading.stopDelayForAll();
```

В нашем примере *self* - это ссылка на блок *storeRawMaterial* из кода его собственного действия

Когда требующие разгрузки поддоны заканчиваются, операция блока *unloading* завершается (путем вызова его функции *stopDelayForAll()*). При этом фура покидает блок *unloading* и поступает в следующий блок диаграммы процесса: *drivingToExit*.

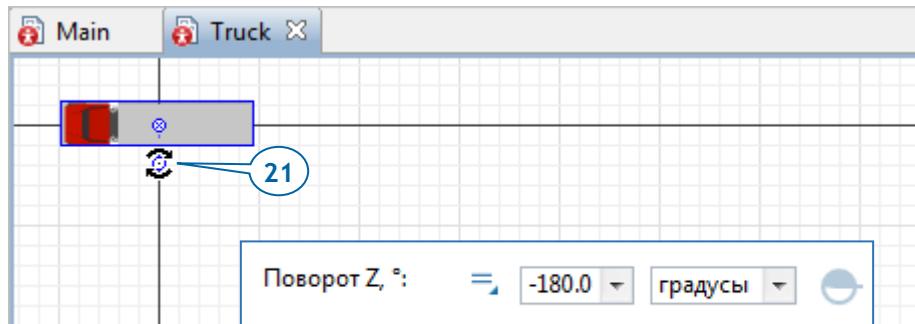
- 19.** Запустите модель.



- 20.** Если фура не так повернута в пространстве (как на рисунке выше), исправьте это, выполнив следующее.

- В дереве панели **Проекты** дважды щелкните по типу агента **Truck**, при этом откроется его диаграмма, и можно будет посмотреть на то, как задана фигура анимации фуры.

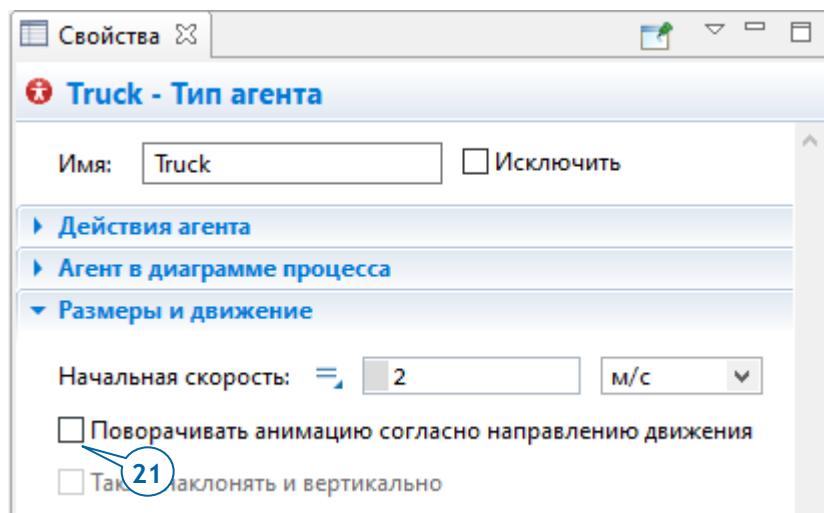
- b. Выберите фигуру фуры, и с помощью круглого маркера (или параметра Поворот Z,° в области свойств Расположение) поверните фигуру фуры на -180 градусов.



Мы изменили изначальный угол поворота фигуры.

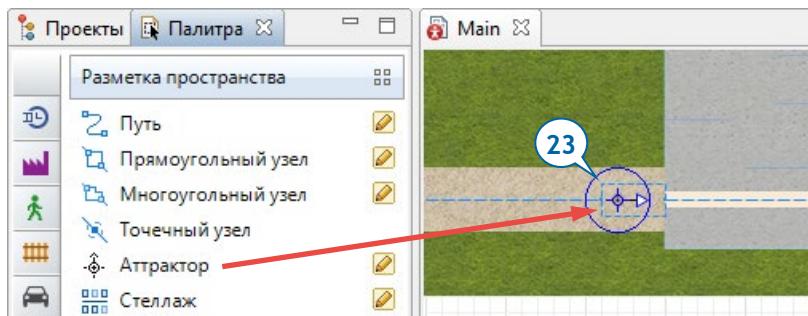
- 21.** Теперь мы должны сделать так, чтобы заданный нами угол поворота не менялся при изменении направления движения фуры. Для этого:

- В панели Проекты, щелкните по элементу **Truck**.
- Раскройте раздел свойств **Размеры и движение**.
- Снимите флажок **Поворачивать анимацию согласно направлению движения**.



- 22.** Откройте диаграмму *Main*.

**23.** Чтобы обеспечить правильное расположение поддонов и фуры внутри узла сети *receivingDock*, откройте палитру **Разметка пространства** и перетащите **АтTRACTор** внутрь узла *receivingDock*. Поместите его так, как показано на рисунке ниже:



### АтTRACTоры в узлах

АтTRACTор позволяет задать точное местоположение агента внутри узла.

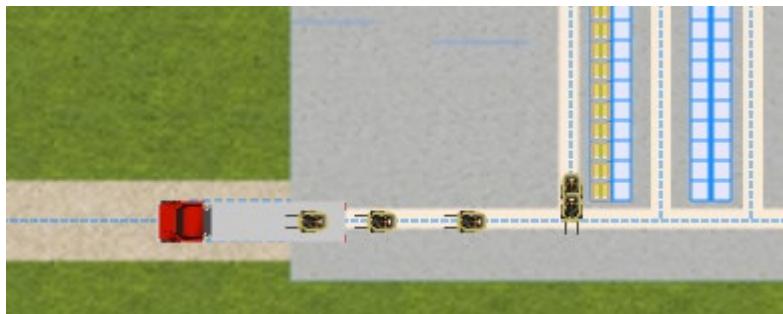
- Если узел задает пункт назначения, к которому движутся агенты, то атTRACTор задает точку - цель движения внутри узла.
- Если узел задает место ожидания, то атTRACTоры задают точки внутри узла, в которых агенты будут находиться во время ожидания.

АтTRACTоры также задают угол поворота анимации агента, когда агент находится внутри узла. В нашем случае мы будем использовать именно это свойство атTRACTора.

АтTRACTоры можно добавить, перетаскивая их по отдельности из палитры, но если они образуют регулярную структуру, то проще будет добавить их всех разом с помощью специального мастера. У этого мастера имеется несколько режимов создания, он также способен удалить все атTRACTоры узла. Открыть мастер можно, щелкнув по кнопке **АтTRACTоры...** в панели свойств узла.

- 24.** Чтобы понять, что изменилось в модели с добавлением атTRACTора, запустите модель.

Теперь анимация процесса разгрузки фуры должна будет выглядеть следующим образом.



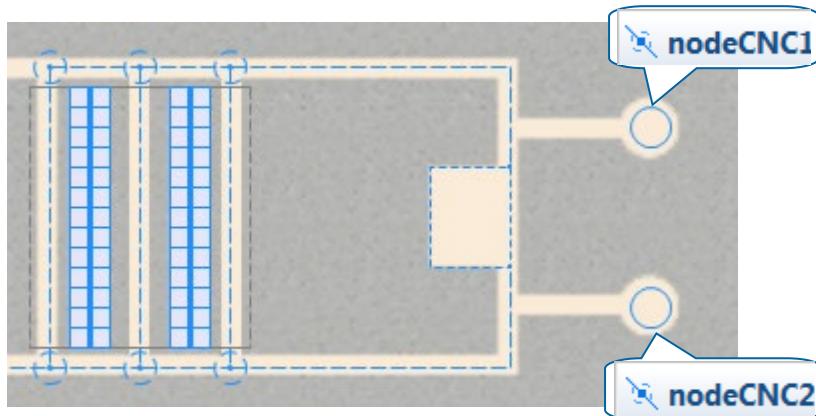
## Фаза 5. Моделирование станков с ЧПУ

На этом этапе обучения мы добавим в модель станки с ЧПУ, на которых будет производиться изготовление готовой продукции.

Давайте начнем с задания мест расположения станков с помощью точечных узлов.

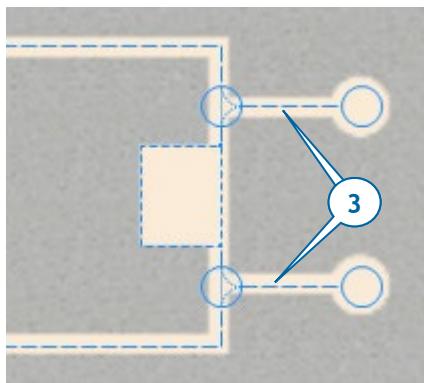
1. Перетащите элемент **Точечный узел** из палитры **Разметка пространства** на план цеха. Назовите этот узел *nodeCNC1*.
2. Скопируйте этот узел, чтобы отметить местонахождение еще одного станка.

AnyLogic присвоит второму узлу имя *nodeCNC2*.



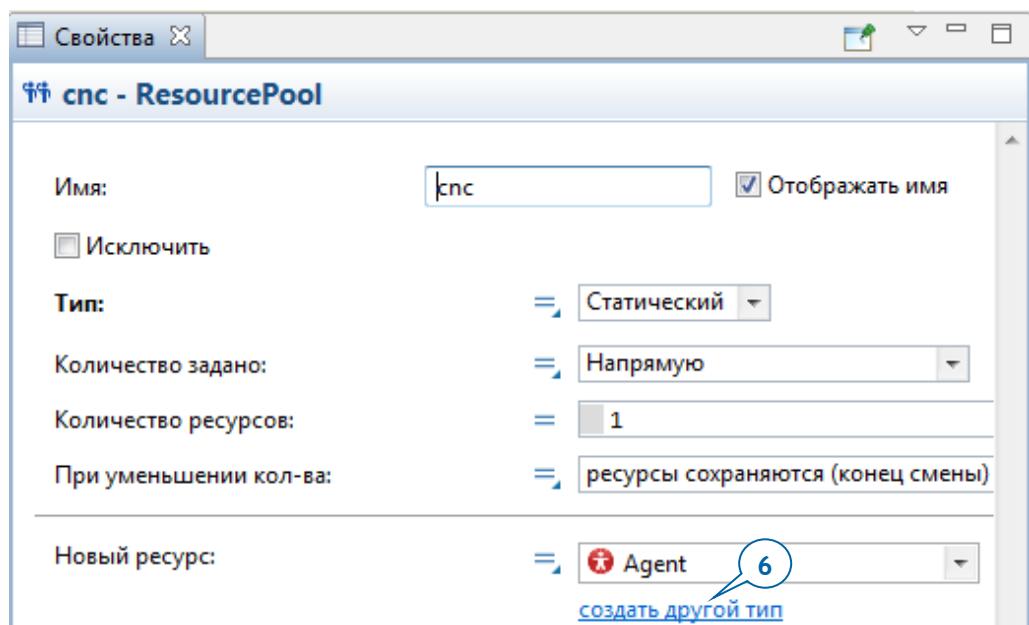
Нам потребуется нарисовать пути, чтобы подключить оба эти узла к нашей сети. Эти пути потребуются автопогрузчикам для подъезда к станкам.

3. В палитре **Разметка пространства** сделайте двойной щелчок по элементу **Путь** и нарисуйте два пути в соответствии со следующим рисунком. Убедитесь, что нарисованные вами пути действительно подсоединяют *nodeCNC1* и *nodeCNC2* к сети. Проверить соединения путей можно, выделив их на диаграмме. Если путь подсоединен к сети, то его конечные точки будут выделены зеленым цветом.



В нашем случае станок с ЧПУ - это ресурс, поэтому мы добавим его в нашу модель, создав новый тип ресурса с помощью блока **ResourcePool**.

4. Перетащите блоки **ResourcePool** из палитры **Библиотека моделирования процессов** на диаграмму *Main*.
5. В области **Свойства** блока **ResourcePool** необходимо выполнить следующее.
  - a. В поле **Имя** введите *cnc*.
  - b. В списке **Тип** выберите **Статический**.



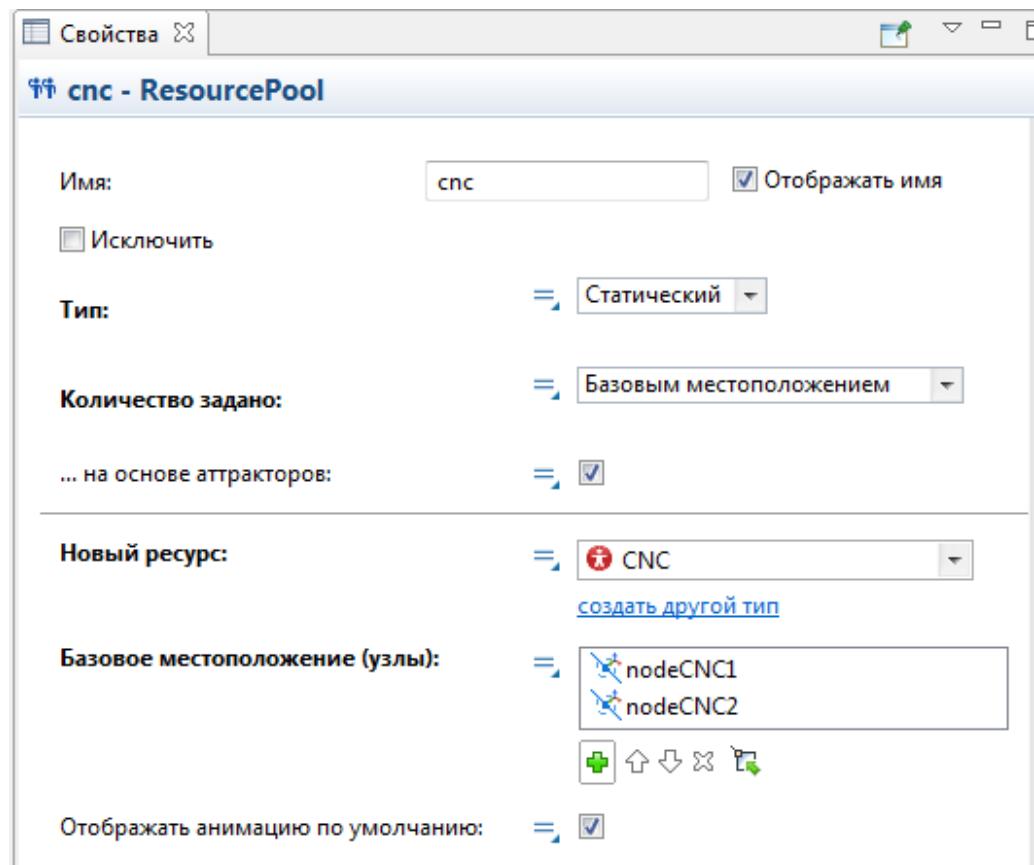
## 194 AnyLogic 8 за три дня

Завершив задание набора ресурсов, мы можем создать новый тип ресурса.

6. Щелкните по ссылке **создать другой тип**, расположенной под списком **Новый ресурс**.
7. В мастере **Создание агентов** сделайте следующее.
  - a. В поле **Имя нового типа** введите *CNC*.
  - b. Перейдите к следующей странице мастера, щелкнув по кнопке **Далее**.
  - c. В списке фигур анимации раскройте раздел **Станки с ЧПУ** и выберите фигуру **Вертикальный станок 2 Сост 1**.
  - d. Щелкните по кнопке **Готово**.
8. Закройте диаграмму типа агента *CNC* и вернитесь на диаграмму *Main*.
9. Чтобы поместить два станка с ЧПУ в точки, заданные точечными узлами *nodeCNC1* и *nodeCNC2*, сделайте следующее.
  - a. Откройте свойства блока *clsc*.
  - b. В списке **Количество задано** выберите опцию **Базовым местоположением**.

Тем самым, мы задаем количество ресурсов равным количеству узлов базового местоположения, указанных для этого пула ресурсов (это мы сделаем следующим делом).
  - c. Щелкните по кнопке  и добавьте *nodeCNC1* и *nodeCNC2* в список **Базовое местоположение (узлы)**.

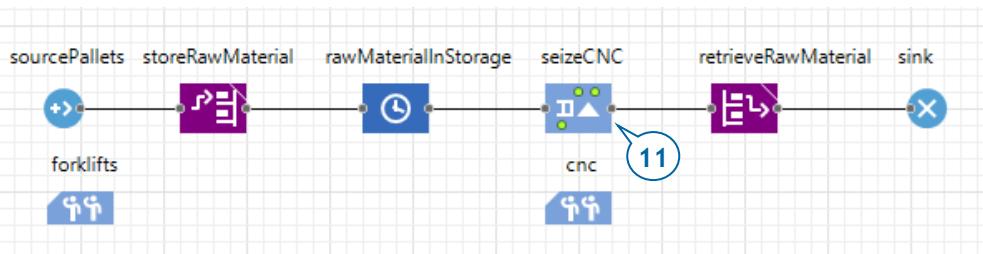
После добавления узлов список должен быть похож на показанный ниже.



Теперь мы готовы изменить диаграмму процесса, описывающую поведение поддонов: мы добавим блок **Seize**, который будет занимать ресурс-станок. Следующий за ним блок **Delay** будет моделировать обработку заготовок на станке, а блок **Release** будет освобождать станок с ЧПУ, делая его готовым для обработки заготовок со следующего поддона.

Как вы помните, в диаграмме процесса уже есть блок *retrieveRawMaterial*, моделирующий доставку поддонов в зону парковки автопогрузчиков.

10. Перетащите блоки диаграммы процесса *retrieveRawMaterial* и *sink* вправо, чтобы освободить место для нового блока.
11. Перетащите блок **Seize** из палитры **Библиотека моделирования процессов** на диаграмму таким образом, чтобы вставить его в диаграмму процесса поддона после блока *rawMaterialInStorage*.



**12.** В свойствах блока **Seize** необходимо выполнить следующее:

- В поле **Имя** введите *seizeCNC*.
- Нажмите кнопку под параметром **Набор ресурсов**, а затем выберите из раскрывающегося списка **cnc**.

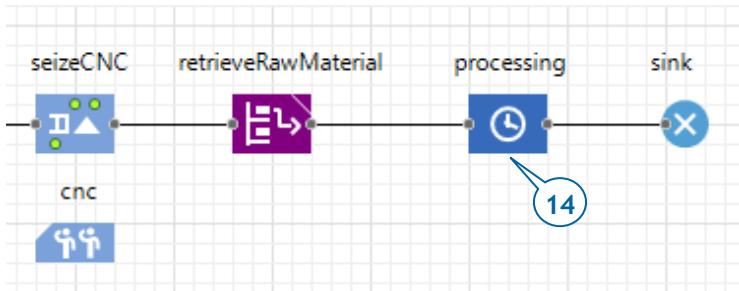
Тем самым, мы задаем, что блок **Seize** будет захватывать один ресурс из набора ресурсов *cnc*.

**13.** В свойствах блока *retrieveRawMaterial* необходимо сделать следующее:

- В списке **Место назначения** выберите опцию **Захваченный ресурс**.
- В расположеннном ниже списке **Ресурс** выберите **cnc**.

Этот блок теперь будет моделировать перемещение поддонов не в зону стоянки автопогрузчиков, а к зарезервированному для выполнения операции станку.

**14.** Чтобы промоделировать обработку заготовок на станке с ЧПУ, добавьте блок **Delay** , поместите его непосредственно после блока *retrieveRawMaterial* и присвойте ему имя *processing*.



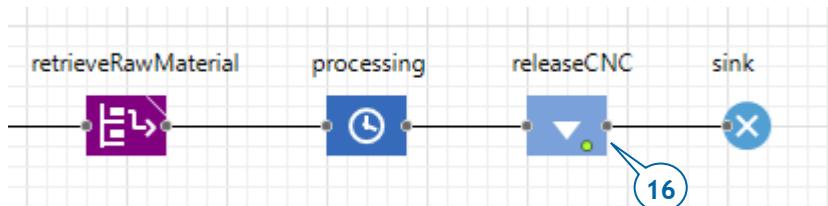
**15.** В свойствах блока **Delay** необходимо выполнить следующее.

- В поле **Время задержки** введите *1*, и из списка справа выберите **минуты**.
- Чтобы несколько станков могли работать одновременно, установите флашок **Максимальная вместимость**.

У каждого агента, прибывающего в блок **Delay**, должен быть зарезервирован один из двух имеющихся в нашей модели станков с ЧПУ.

**16.** Перетащите блок **Release** ▾ из палитры **Библиотека моделирования процессов** в диаграмму процесса поддонов. Поместите его после блока *processing*.

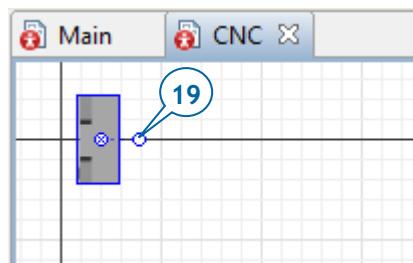
**17.** Присвойте этому блоку **Release** имя *releaseCNC*.



Когда вы запустите модель, то увидите, что хотя процессы и смоделированы верно, но на трехмерной анимации поддоны отображаются прямо по центру фигуры анимации станка с ЧПУ. Это происходит потому, что и станок, и обрабатываемый им поддон используют один и тот же точечный узел для отображения анимации. Чтобы решить эту проблему, нам необходимо переместить станок с ЧПУ вправо и повернуть его так, чтобы он был обращен к поддону.

**18.** В панели **Проекты**, сделайте двойной щелчок по элементу *CNC*. При этом откроется диаграмма этого типа агента.

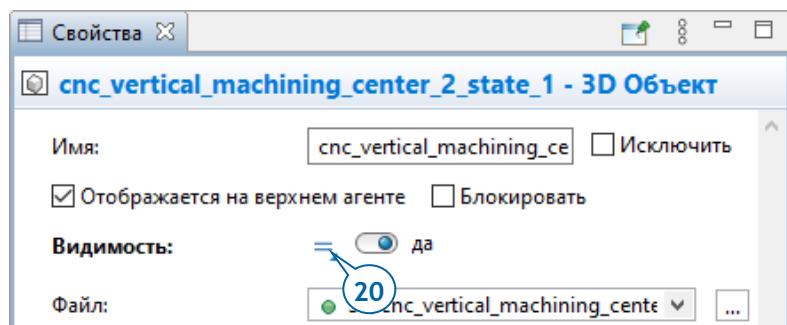
**19.** Переместите анимацию вправо и поверните фигуру анимации станка с помощью круглого маркера, как показано на рисунке ниже.



Давайте используем для анимации станка два схожих объекта трехмерной анимации, один из которых будет представлять станок в режиме ожидания, а второй - в процессе обработки заготовок. У этих объектов мы настроим значения свойства **Видимость**, благодаря чему наша модель будет отображать ту или иную фигуру в зависимости от текущего состояния станка.

**20.** Для этого необходимо выполнить следующее.

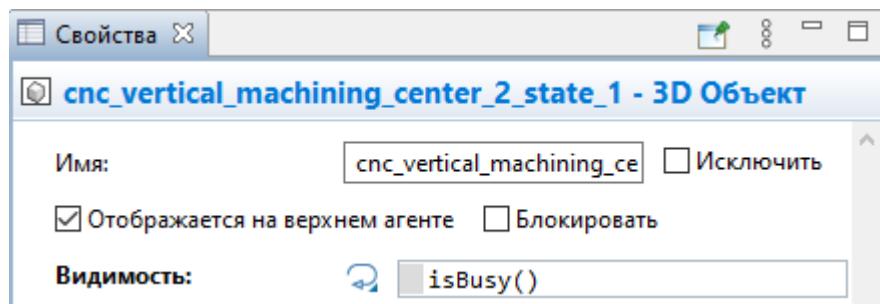
- Выделите фигуру анимации станка.
- В свойствах, наведите курсор мыши на значок  , расположенный рядом с меткой свойства **Видимость**, и выберите опцию **Динамическое значение**.



Значок статического свойства  поменяется на значок динамического свойства  , при этом появится поле для задания динамического выражения. В это поле можно ввести выражение Java, возвращающее значение *true* или *false*.

- Введите в поле выражение *isBusy()*

Эта функция ресурса возвращает *true* в случае занятости ресурса. В нашем случае этот трехмерный объект будет отображаться, если в данный момент станок обрабатывает заготовки.



## Динамические свойства

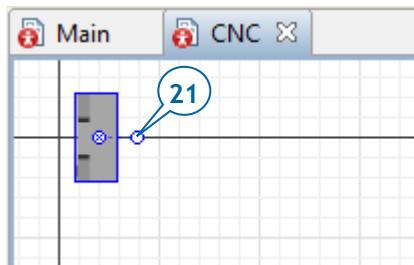
Когда вы задаете выражение для динамического значения свойства, по ходу выполнения модель будет вычислять это выражение на каждом кадре анимации и применять вычисленное значение в качестве текущего значения свойства. С помощью этой возможности пользователи AnyLogic могут анимировать свои модели, задавая динамические значения для координат, размерностей или цвета графических объектов.

Если вы не зададите динамическое значение свойства, то в процессе выполнения модели свойство сохранит свое статическое значение.

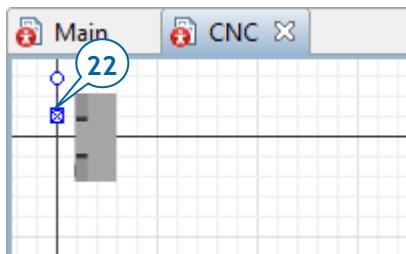
- У блоков, входящих в диаграмму процесса, могут быть:
  - = **Статические свойства**, имеющие постоянное значение на протяжении всего процесса моделирования, которое, однако, может быть изменено функцией *set\_имяСвойства (новое значение)*;
  - ⌚ **Динамические свойства**, значения которых вычисляются заново для каждого прибывающего в блок агента;
  - ☰ **Кодовые свойства**, с помощью которых вы можете задавать действия, которые необходимо выполнить в особые моменты жизни агентов в данном блоке диаграммы процесса, например, действия **При входе** или **При выходе** из блока. Как правило, кодовые свойства размещены в разделе свойств **Действия**.
- Небольшой треугольник у значка свойства говорит о том, что при щелчке по значку вы можете переключаться между редактором статических значений и полем, в котором вы можете ввести выражение для регулярного вычисления динамического значения.

**21.** Чтобы добавить трехмерный объект анимации, который будут отображаться, только если станок не обрабатывает заготовки, необходимо выполнить следующее.

- a. Откройте палитру **3D Объекты**, на которой находятся готовые к использованию трехмерные объекты AnyLogic.
- b. Раскройте раздел фигур **Станки с ЧПУ** и перетащите графический объект **Вертикальный станок 2 Сост 2** на диаграмму типа агента *CNC*.
- c. Поверните графический объект и расположите его непосредственно поверх добавленной нами ранее фигуры.
- d. Перейдите в редактор динамических значений поля **Видимость**, и в качестве выражения введите *isIdle()*.



**22.** Раскройте раздел **Люди** палитры **3D Объекты** и перетащите графический объект **Рабочий** на диаграмму *CNC*.



**23.** Запустите модель и понаблюдайте за процессом.

Вы увидите, как автопогрузчики подвозят поддоны к станкам с ЧПУ для обработки заготовок. Вы также должны увидеть анимацию процесса обработки - трехмерное изображение станков будет меняться в зависимости от их состояния.



Мы завершили работу с нашей простой моделью, моделирующей процесс производства в небольшом заводском цеху. Теперь вы обладаете базовыми знаниями о ресурсах AnyLogic и приемах работы с ними. Вы также научились описывать процессы, собирая диаграмму процесса из блоков **Библиотеки моделирования процессов**.

Вы можете самостоятельно промоделировать перемещение поддонов с готовой продукцией в другой стеллаж, находящийся перед отгрузочной зоной, где поддоны будут находиться вплоть до момента отправки.

## Пешеходное моделирование. Модель аэропорта

Давайте создадим модель небольшого аэропорта с двумя гейтами. Пассажиры будут прибывать в аэропорт и регистрироваться на рейс (если они не сделали этого заблаговременно через интернет). Затем все пассажиры должны будут пройти процедуру предполетного досмотра, после чего они смогут направиться в зону ожидания перед гейтами, дожидаясь начала посадки на свой рейс. При объявлении начала посадки, пассажиры направляются к соответствующему гейту. У гейта служащие аэропорта проводят проверку посадочных талонов, после чего пассажиры проходят на посадку на самолет.

Процесс создания модели будет разбит на шесть фаз. В последней из них вы научитесь считывать данные из базы данных (мы считаем информацию о совершаемых рейсах из файла Microsoft Excel).

## Фаза 1. Задание потока пешеходов

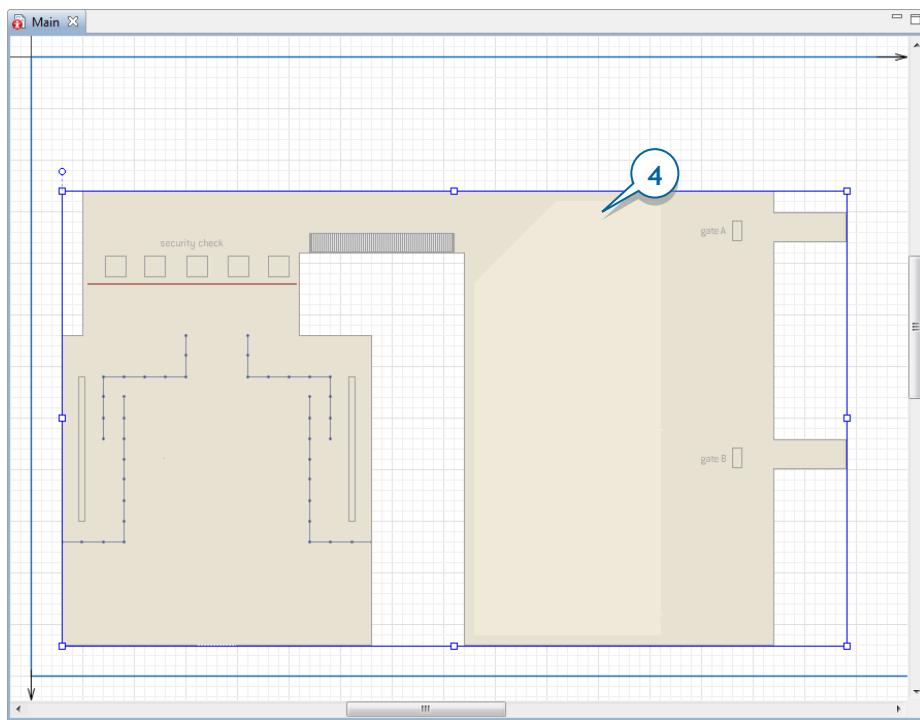
Мы начнем с создания простой модели, в которой пассажиры прибывают в аэропорт и движутся к выходу на посадку. Логику модели мы зададим с помощью блоков *Пешеходной библиотеки AnyLogic*.

### Пешеходная библиотека

- Пешеходная библиотека AnyLogic является высокоуровневой библиотекой моделирования движения пешеходов в физическом пространстве. Она позволяет моделировать здания, в которых скапливаются большие количества людей (станции метро, железнодорожные вокзалы, аэропорты, торговые центры, стадионы, музеи и т.д.).
- В моделях, созданных с помощью блоков Пешеходной библиотеки, пешеходы движутся в непрерывном пространстве, реагируя на различные виды препятствий в виде стен и других пешеходов.
- Вы можете создать впечатляющую визуализацию для презентации и валидации вашего проекта, собрать статистику плотности пешеходов в различных областях модели для того, чтобы убедиться, что сервисы смогут справиться с потенциальным ростом нагрузки, вычислить время пребывания пешеходов в каких-то определенных участках модели, выявить возможные проблемы, которые можно решить путем перепланировки здания или добавления дополнительных сервисов, и т.д.

Обычно создание пешеходных моделей начинается с добавления плана моделируемого пространства и рисования стен, обозначенных на плане.

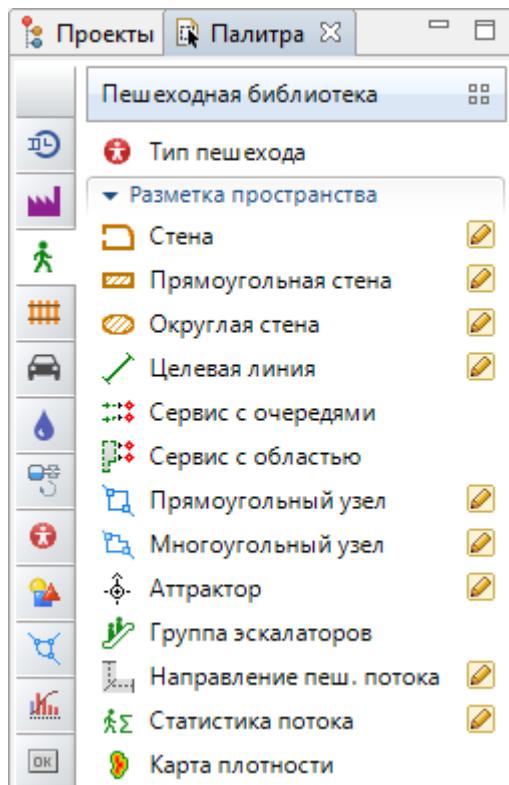
1. Создайте новую модель и назовите ее *Airport*.
2. Добавьте **Изображение**  из палитры **Презентация**  на диаграмму *Main*.
3. Выберите изображение плана аэропорта из файла *terminal.png*, находящегося в каталоге AnyLogic */resources/AnyLogic in 3 days/Airport*.



- Поместите изображение ближе к левому нижнему углу синей рамки на диаграмме *Main*. Эта рамка задает область, которая будет отображаться в окне модели при ее запуске. Если пропорции изображения нарушены, то в свойствах изображения щелкните по кнопке **Восстановить исходный размер**, а затем выберите опцию **Блокировать**, чтобы заблокировать эту фигуру.

Теперь давайте зададим стены, ограничивающие моделируемое пространство. Для этого мы воспользуемся специальными элементами разметки пространства, созданными для пешеходного моделирования в AnyLogic. Все эти фигуры можно найти в палитре **Пешеходная библиотека**, в разделе **Разметка пространства**.

## Элементы разметки пространства для пешеходных моделей



### Секция Разметка пространства палитры Пешеодная библиотека

Обычно создание модели начинается с рисования стен поверх имеющегося плана моделируемого помещения. Стены в пешеходном моделировании представляют собой объекты, через которые пешеходы не могут пройти.

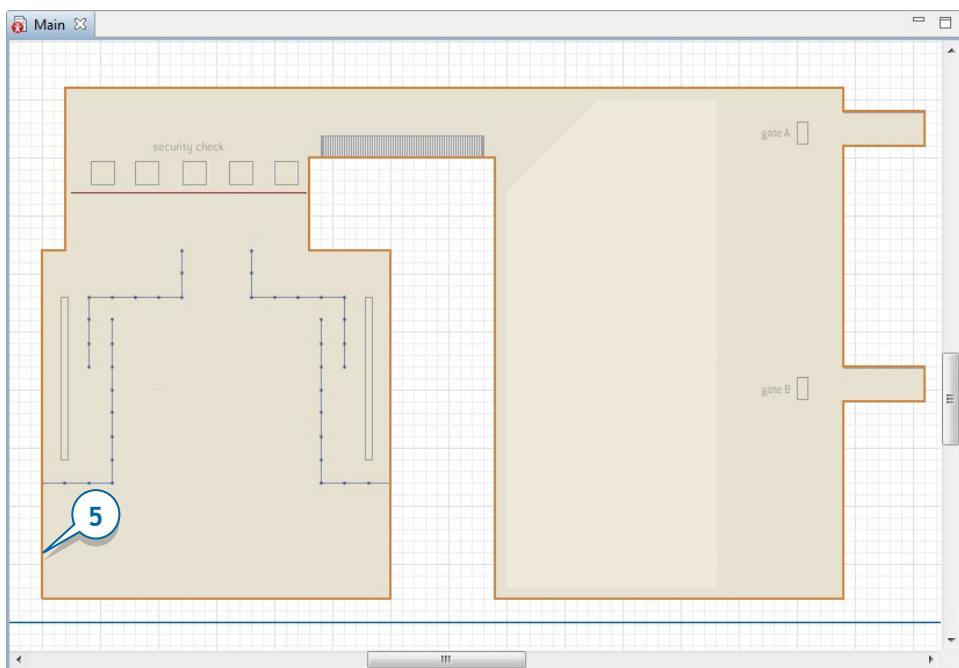
#### Стены

- **Стена** – Используется для рисования всех внешних и внутренних стен моделируемого помещения.
- **Прямоугольная стена** – Обычно используется для задания прямоугольных помещений внутри заданного стеной пространства, закрытых для пешеходов (служебные помещения и т.д.).



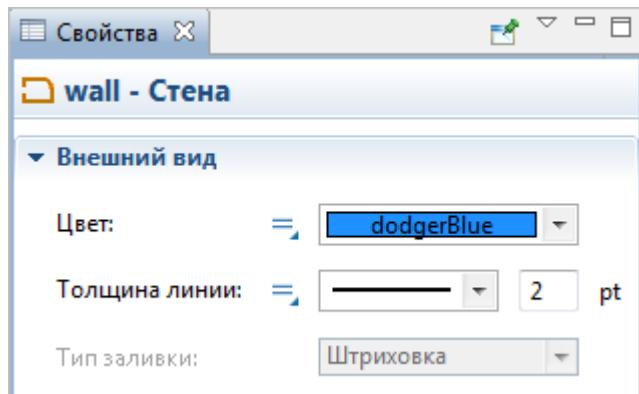
**Округлая стена** - Используется для задания округлых пространств, недоступных для прохода пешеходов (колонны, фонтаны и т.д.).

5. Нарисуйте стены терминала. Используйте режим рисования: вначале сделайте двойной щелчок по элементу **Стена** в секции **Разметка пространства** палитры **Пешеходная библиотека** , а затем нарисуйте стену на диаграмме, последовательно щелкая мышью в точках изгиба стены и добавив конечную точку двойным щелчком.

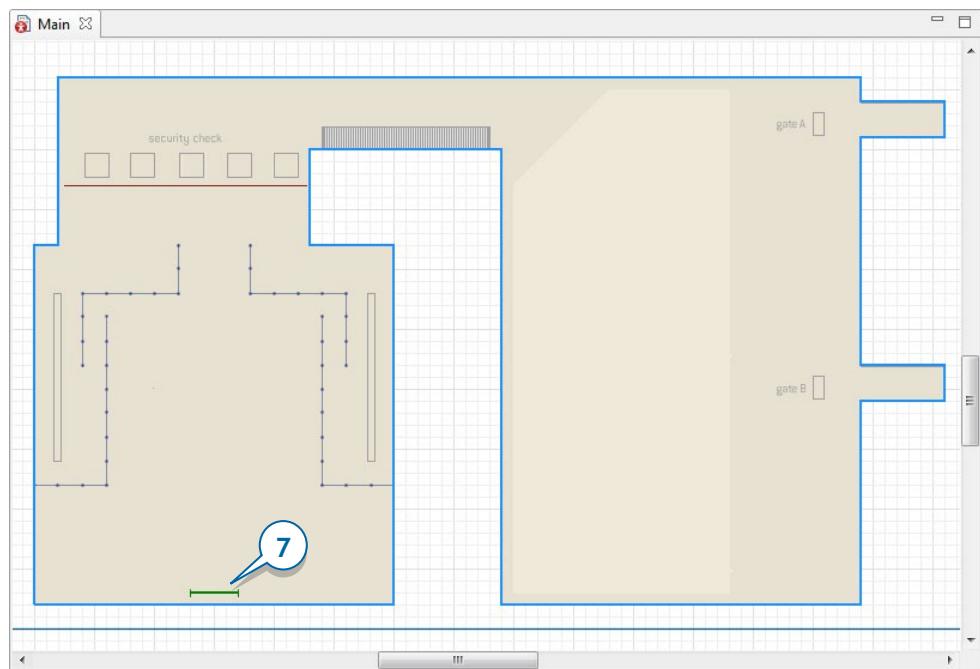


Измените внешний вид стены.

6. Перейдите в секцию свойств стены **Внешний вид** и выберите другой **Цвет**: *dodgerBlue*.

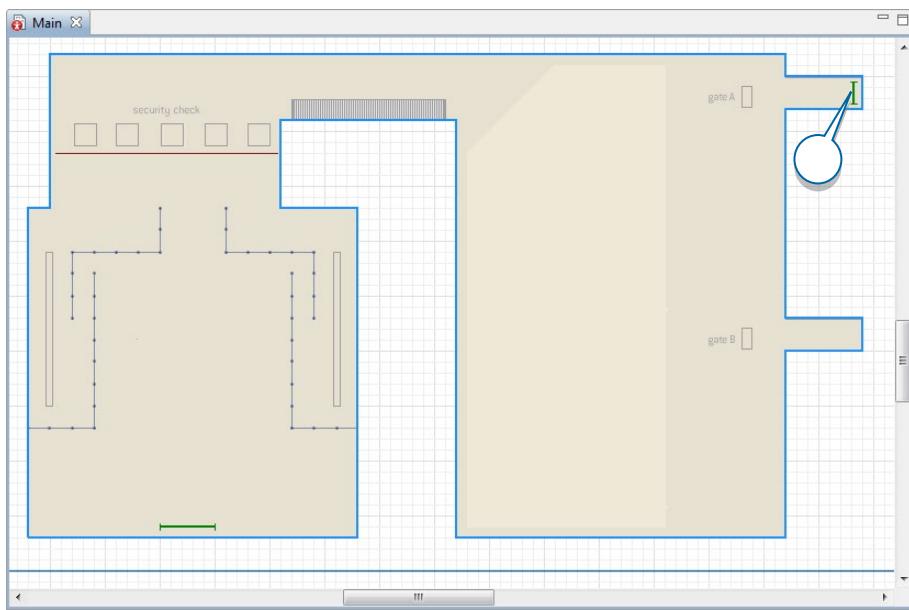


7. Задайте место, в котором будут появляться пешеходы в моделируемом пространстве. Для этого перетащите элемент **Целевая линия** из секции **Разметка пространства** палитры **Пешеходная библиотека** на графическую диаграмму, как это показано на рисунке ниже.



8. Назовите эту целевую линию *arrivalLine*.
9. Добавьте еще одну целевую линию. Поместите ее в область выхода на посадку, как показано на рисунке ниже, и назовите ее *gateLine1*. Прибывающие

в терминал пассажиры будут двигаться к этой линии (мы начнем с простейшей модели, где они просто идут на посадку в гейт).



- Как линия, на которой будут появляться пешеходы, так и линии, задающие сервисы, очереди к ним и т.д. – все они должны быть помещены **внутрь** стен, ограничивающих моделируемое пространство, чтобы они были достижимы пешеходами. Иначе во время выполнения модели возникнет ошибка “Target is not reachable” – “Цель недостижима”.

### Целевая линия

Элемент разметки пространства *Целевая линия* используется в пешеходных моделях для задания следующих элементов:

- Место появления пешеходов в моделируемой среде (используется в объектах **PedSource** и **PedEnter**).
- Цель движения пешеходов (используется в объекте **PedGoTo**).
- Место ожидания пешехода (используется в объекте **PedWait**).
- Место выхода с текущего этажа и перехода на новый этаж (используется в объекте **PedChangeLevel**).

Мы нарисовали ключевые объекты моделируемого пространства с помощью фигур разметки пространства. Теперь можно приступить к заданию логики модели с помощью блоков Пешеходной библиотеки.

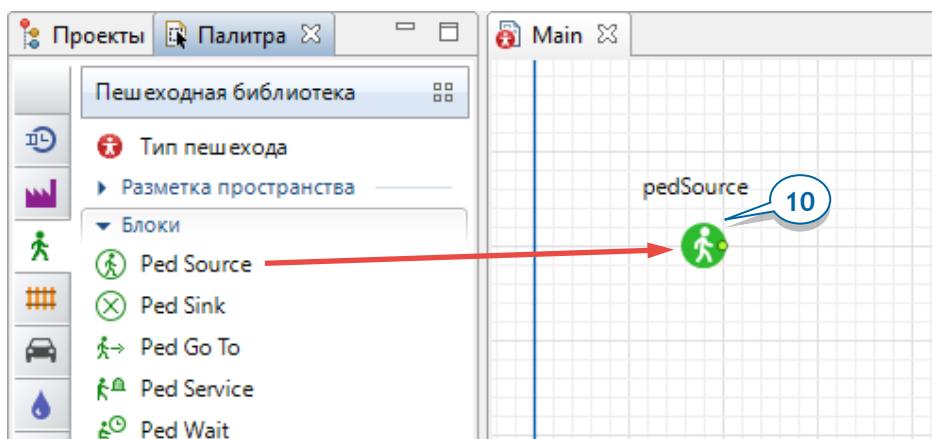
## Задание логики движения потока пешеходов

Все процессы, происходящие в моделируемом пространстве, вы задаете с помощью диаграммы процесса, собираемые из блоков **Пешеходной библиотеки**.

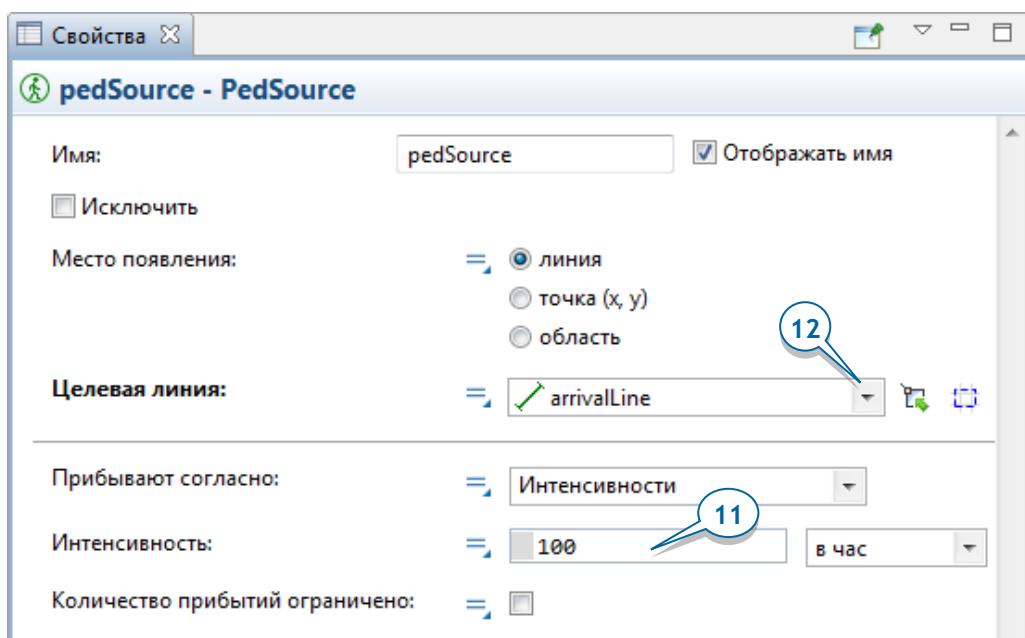
Перечислим самые важные и часто используемые блоки библиотеки:

-  **PedSource** – Этот блок создает пешеходов, аналогично тому, как блок **Source** создает агентов в диаграмме, собранной из блоков Библиотеки Моделирования Процессов. Обычно с этого блока начинается рисование диаграммы логики движения пешеодного потока.
-  **PedGoTo** – Этот блок моделирует движение пешеходов в заданную точку.
-  **PedService** – Этот блок моделирует то, как пешеходы обслуживаются в заданной точке обслуживания (сервисе).
-  **PedWait** – Этот блок моделирует то, как пешеходы ждут в течение заданного времени в указанной точке или области.
-  **PedSelectOutput** – Блок перенаправляет потоки пешеходов в разные подпроцессы в заданных пропорциях, либо же направляет пешеходов с разными характеристиками на выполнение различных действий.
-  **PedSink** – Блок удаляет из модели пешеходов, выполнивших все заданные операции. Обычно этот блок завершает диаграмму процесса.

10. Начнем создание диаграммы процесса с добавления на диаграмму *Main* блока **PedSource**  из палитры **Пешеходная библиотека**.



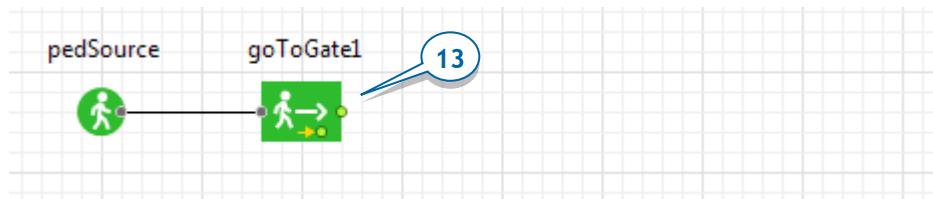
11. Пусть в среднем в терминал прибывает сто пассажиров в час. Откройте свойства блока *pedSource* и введите *100* в поле **Интенсивность**. Укажите блоку *pedSource*, чтобы он добавлял 100 пешеходов в час.



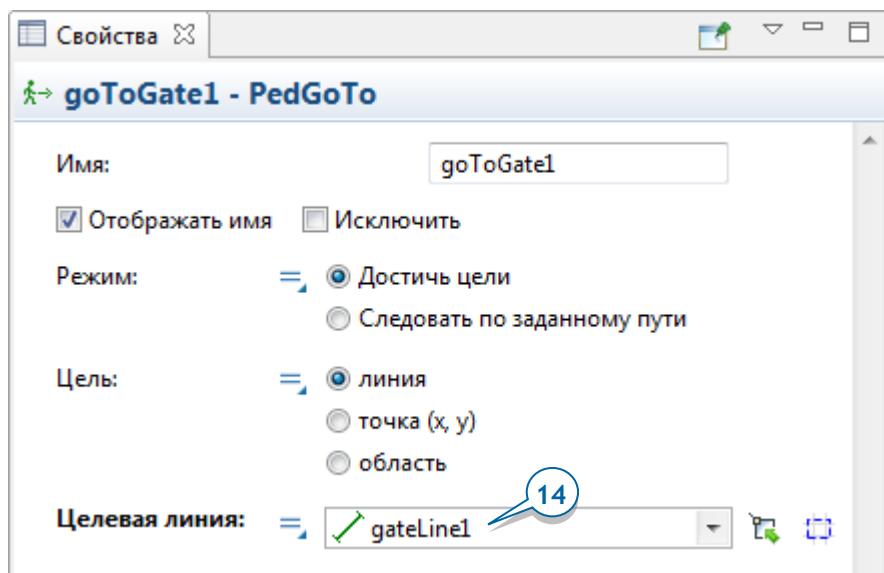
12. Задайте место появления пассажиров в моделируемой среде, выбрав *arrivalLine* в поле **Целевая линия**.

13. Затем добавьте объект **Ped Go To**, моделирующий движение пешеходов к заданному месту, и соедините его с блоком *pedSource*. Поскольку мы хотим,

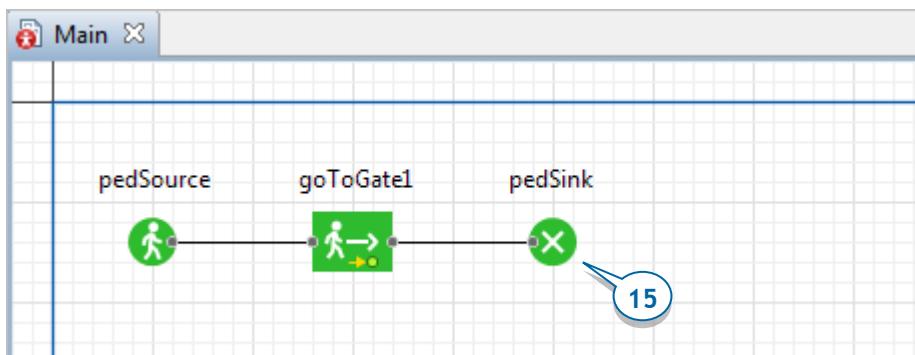
чтобы пассажиры направлялись к первому (верхнему) гейту, назовите объект `goToGate1`.



- 14.** В свойствах этого блока, задайте цель движения пассажиров, выбрав `gateLine1` из списка Целевая линия.



- 15.** Завершите диаграмму блоком **PedSink** . Он будет удалять из модели поступающих в него пешеходов. Диаграмма пешеходного процесса практически всегда начинается с объекта **PedSource**, а заканчивается объектом **PedSink**.



Ваша диаграмма процесса должна выглядеть так, как на приведенном выше рисунке.

- 16.** Запустите модель. Вы увидите, как пассажиры движутся по аэропорту от входа к заданному выходу на посадку.

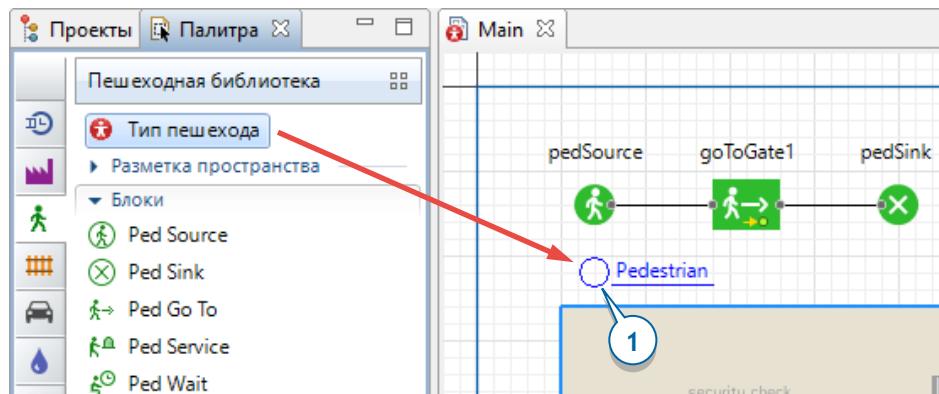


## Фаза 2. Создание 3D анимации

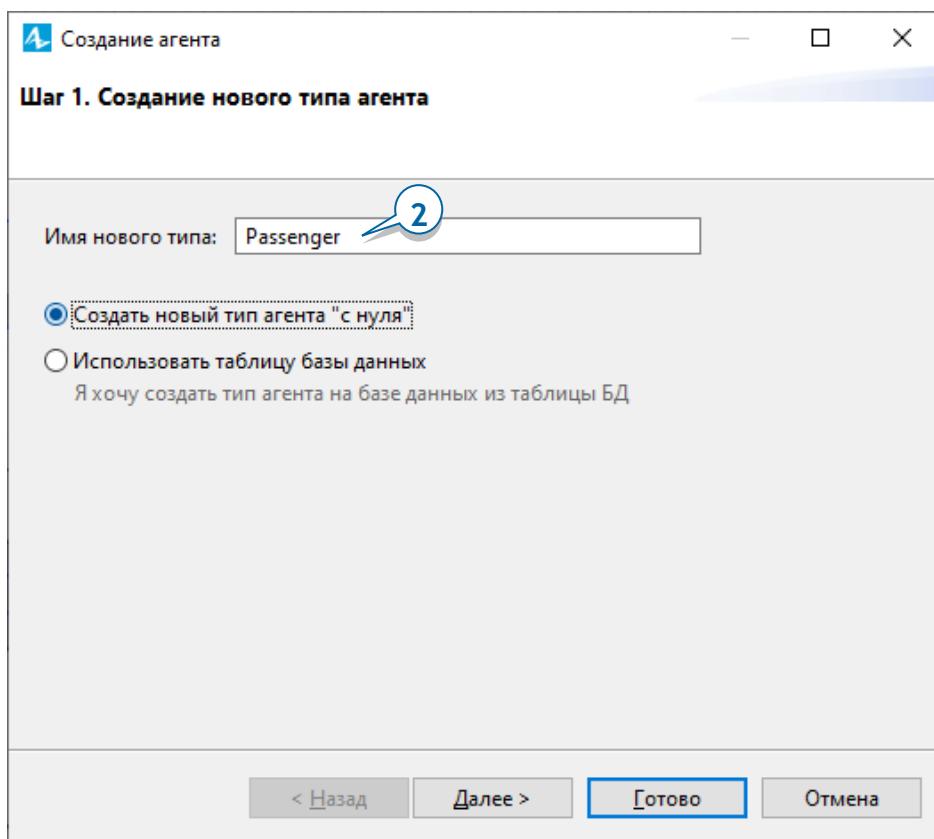
Теперь давайте добавим в нашу модель 3D анимацию. Для этого, как мы уже знаем, нам понадобится добавить 3D окно, камеру и 3D изображение пассажира. Начнем именно с задания трехмерной фигуры анимации, что потребует создания нового **Типа пешехода**.

- Если вы хотите создать трехмерную анимацию или задать у пешехода специфические характеристики, то для этого потребуется создать тип пешехода. На диаграмме созданного типа вы сможете как нарисовать трехмерную фигуру пешехода, так и задать его характеристики с помощью параметров.

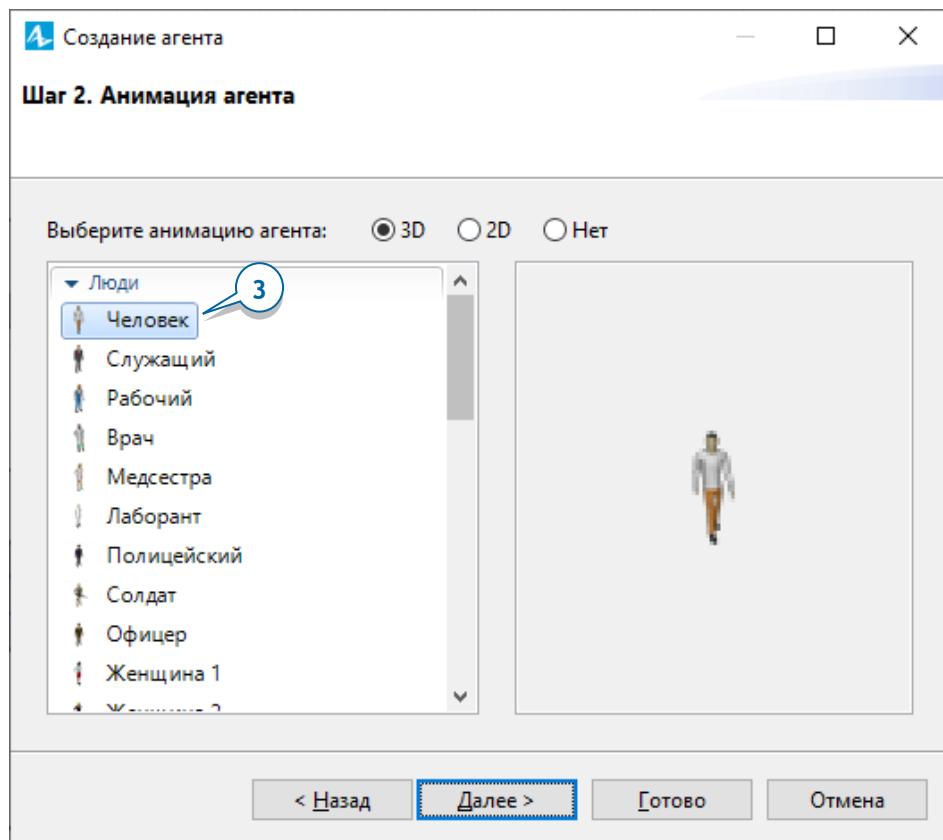
- Перетащите элемент **Тип пешехода**  из палитры **Пешеходная библиотека** на диаграмму *Main*.



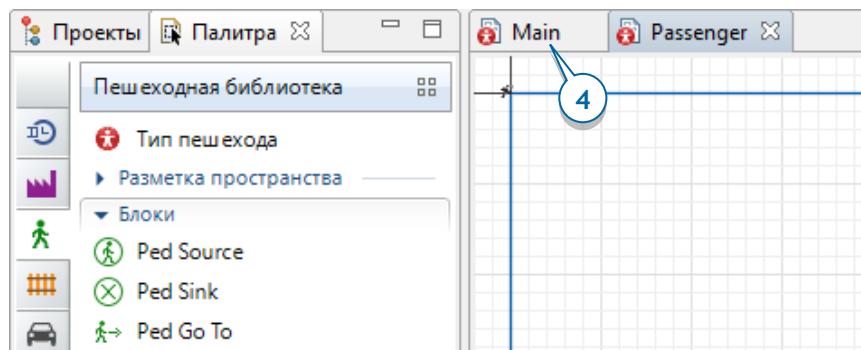
- На первой странице Мастера создания нового агента, введите имя нового типа: *Passenger*. Щелкните по кнопке **Далее**.



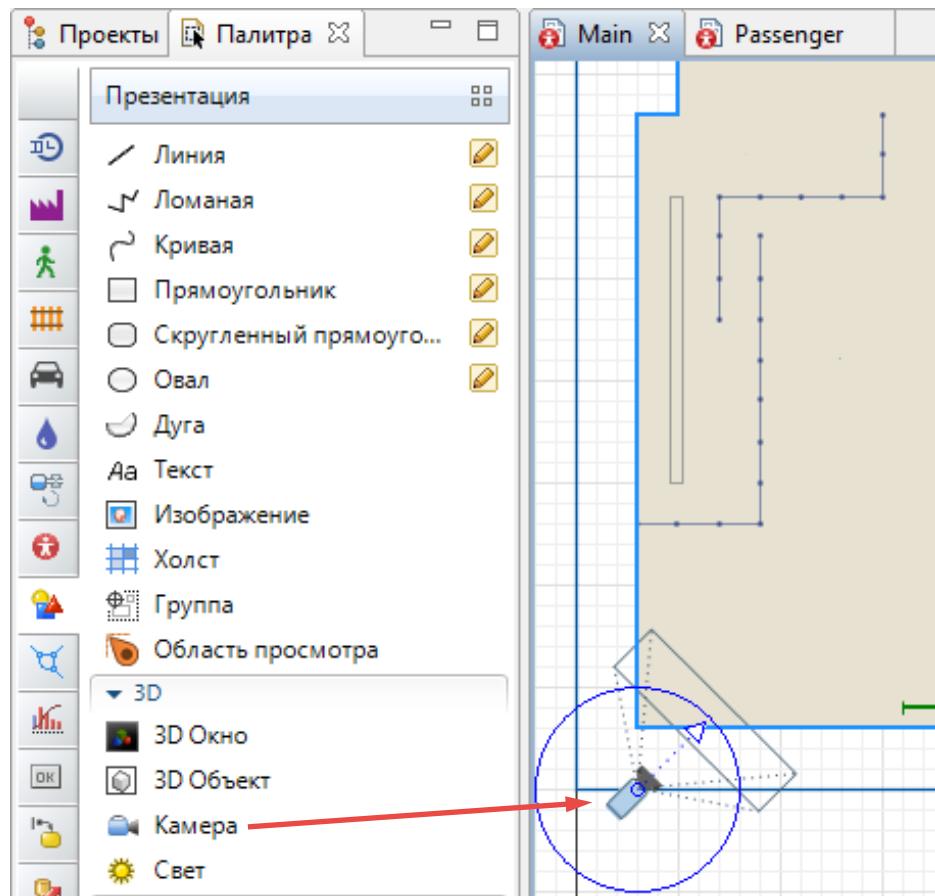
3. На второй странице Мастера оставьте выбранной опцию 3D и фигуру Человек. Щелкните по кнопке Готово.



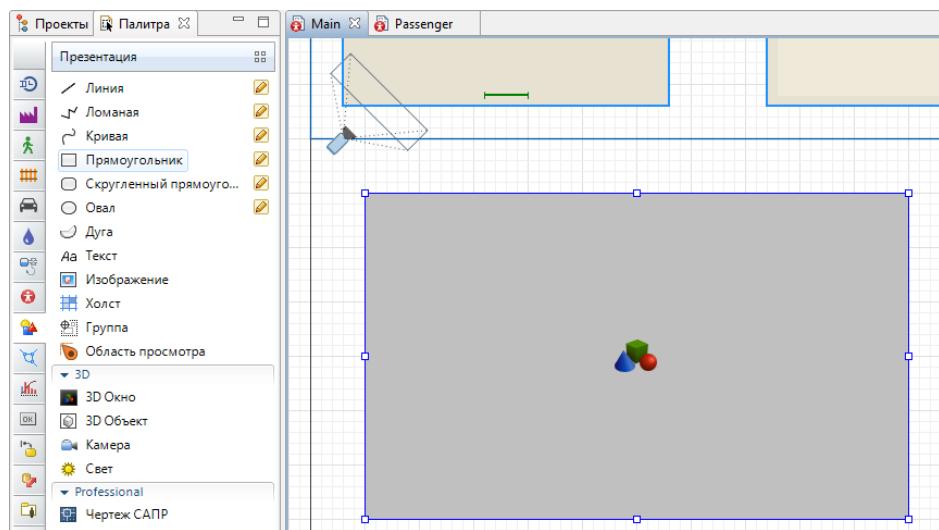
4. AnyLogic откроет диаграмму типа агента *Passenger*. Перейдите обратно на диаграмму *Main*, чтобы продолжить создание модели.



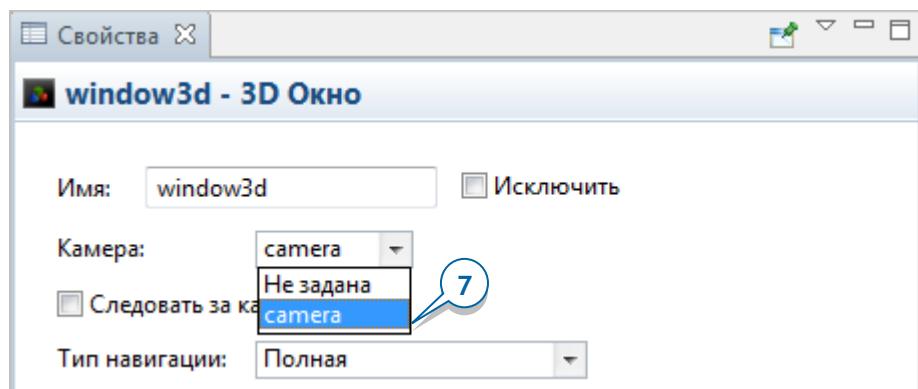
5. Перетащите элемент Камера из палитры Презентация на диаграмму Main. Поместите камеру так, чтобы она была направлена на план терминала (то есть, как бы "снимала" происходящее в терминале).



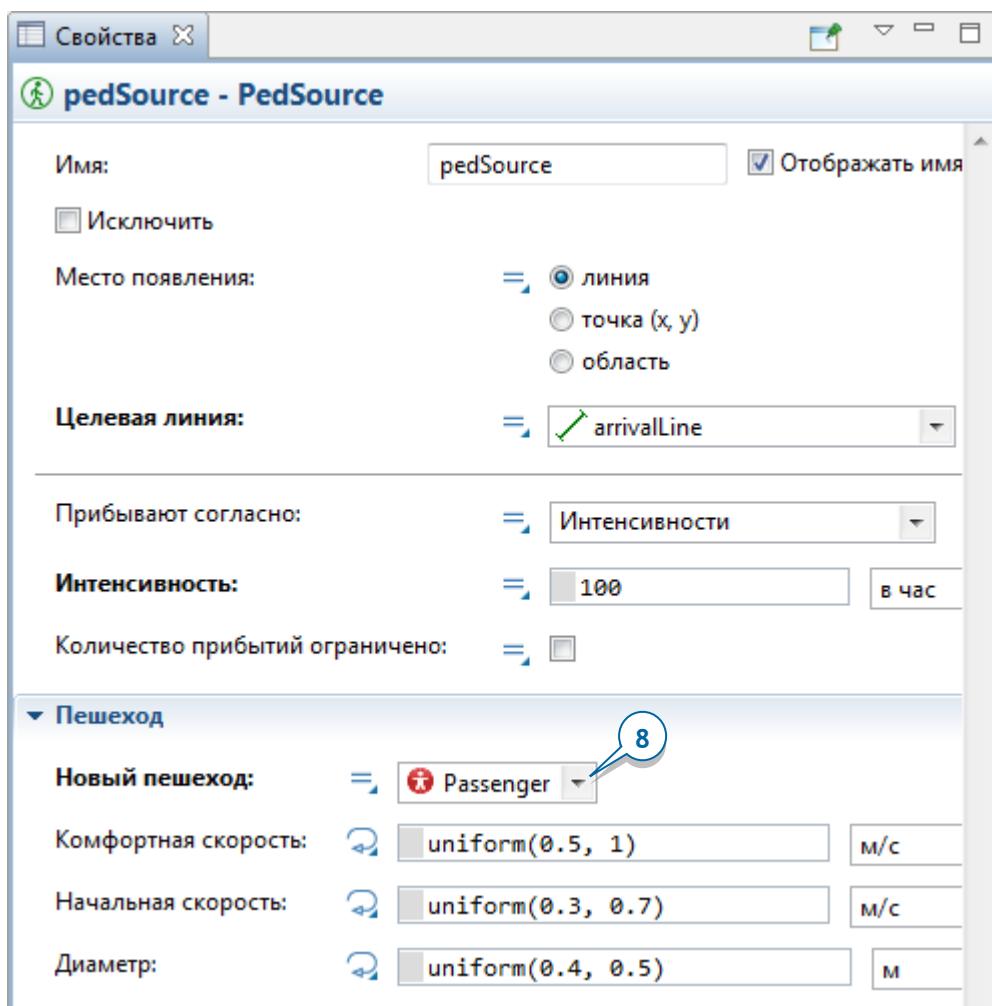
6. Перетащите элемент 3D Окно из палитры Презентация на диаграмму Main. Расположите 3D окно под планом терминала, как показано ниже:



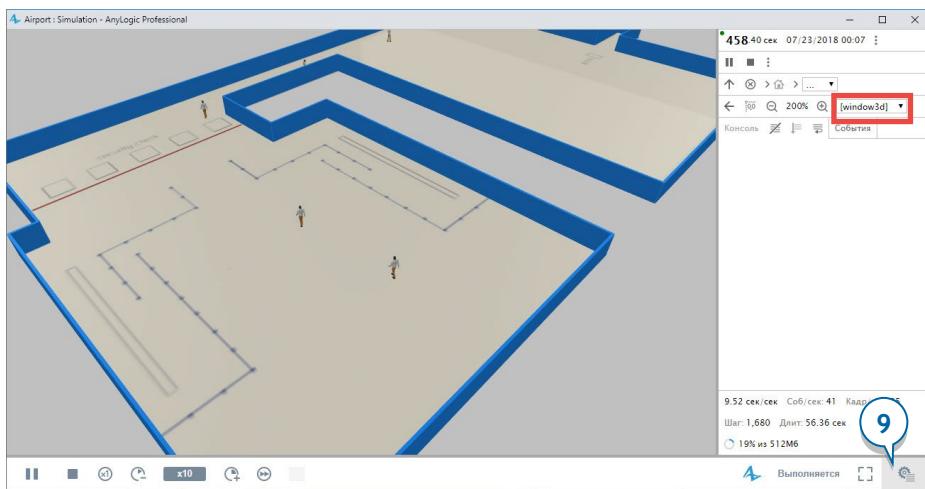
7. Откройте свойства 3D окна и выберите *camera* в свойстве Камера.



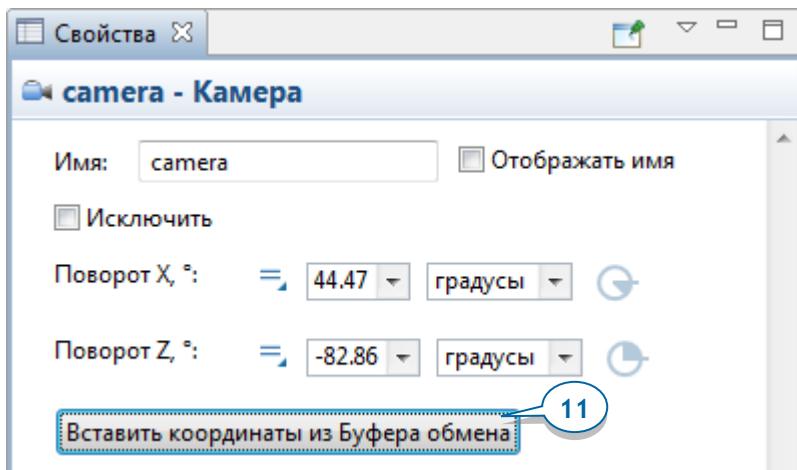
8. Мы хотим, чтобы блок диаграммы процесса *pedSource* создавал пешеходов созданного нами типа *Passenger*. Откройте свойства блока *pedSource* и выберите *Passenger* из списка **Новый пешеход**, расположенного в разделе свойств **Пешеход**.



- Запустите модель. Вы увидите, как пассажиры идут внутри здания, следуя от входа к выходу на посадку. Вы можете переключиться на 3D анимацию, щелкнув по кнопке панели разработчика **Выбрать область и показать** и выбрав опцию **[window3d]**.



10. Перемещайтесь по сцене трехмерной анимации до тех пор, пока не найдете наилучшее расположение камеры. Тогда щелкните правой кнопкой мыши по 3D сцене (Mac OS: Ctrl+щелчок) и выберите из контекстного меню команду **Копировать положение камеры**.
11. Закройте окно презентации и откройте свойства камеры. Примените оптимальное расположение камеры, щелкнув по кнопке **Вставить координаты из Буфера обмена**.



Если вы не знаете, где расположена камера на холсте графической диаграммы, то ее можно легко найти в панели **Проекты** (в ветви **Презентация** агента *Main*).

**12.** Снова запустите модель и удостоверьтесь, что теперь 3D анимация отображается с учетом заданного расположения камеры.

## Фаза 3. Моделирование предполетного досмотра пассажиров

Теперь мы можем начать моделирование процессов, происходящих в аэропорту.

Давайте начнем с моделирования процедуры предполетного досмотра пассажиров. Для этого мы добавим в нашу модель пункты досмотра пассажиров. С точки зрения терминологии пешеходного моделирования, пункт досмотра является *сервисом* – здесь пассажиры должны быть обслужены, а если пункт досмотра в данный момент занят, то пассажирам придется ждать в очереди, пока он не освободится.

### *Сервисы в пешеходных моделях*

В пешеходном моделировании объекты, в которых пешеходы проводят определенное время для выполнения какой-либо операции/процедуры, называются *сервисами*. Примеры сервисов: турникеты, кассы, автоматы по продаже билетов/напитков и т.д.

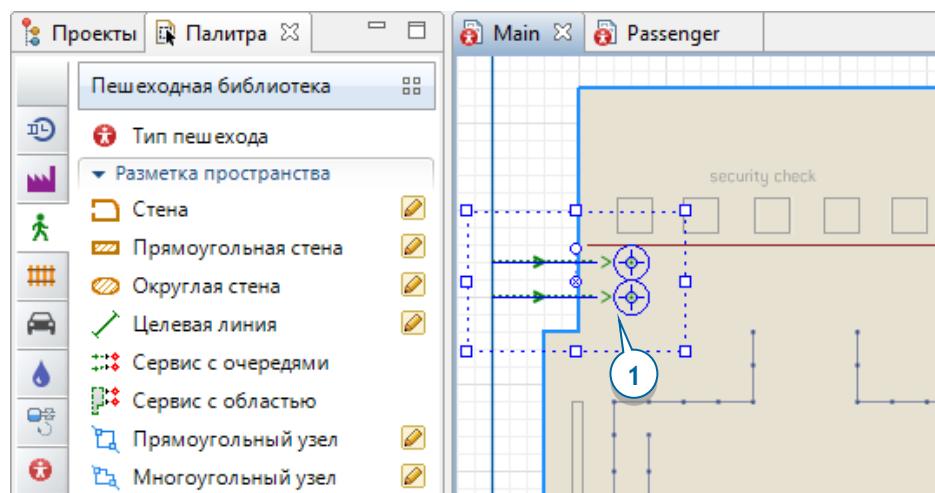
Задание процесса обслуживания состоит из двух шагов. Вначале с помощью фигур разметки пространства **Сервис с очередями** или **Сервис с областью** вы задаете, где в моделируемом пространстве располагаются точки сервиса и очереди к ним.

-  **Сервис с очередями** – С помощью этой фигуры разметки пространства вы можете задать те сервисы, при ожидании доступа к которым люди стоят в очередях (турникеты, билетные кассы и т.д.).
-  **Сервис с областью** – Эта фигура используется для задания сервисов с электронной очередью, когда люди получают номерки и ждут своей очереди, располагаясь в прилегающей области ожидания. Электронная очередь применяется все чаще в банковских отделениях, информационных пунктах на вокзалах и т.д.

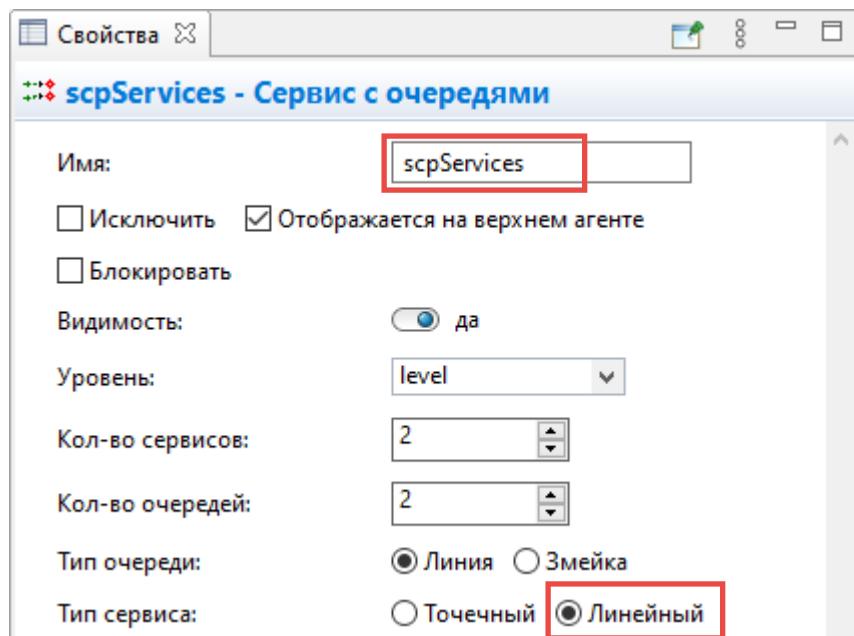
Задав сервисы графически, нужно добавить в диаграмму процесса блок **PedService**, в свойствах которого следует указать соответствующую фигуру сервиса и задать время прохождения процедуры в данном пункте обслуживания.

В моделируемом нами терминале находятся пять пунктов досмотра, поэтому нам нужно будет добавить пять пунктов сервиса, к каждому из которых должна вести очередь.

- Перетащите элемент **Сервис с очередями**  из палитры **Пешеходная библиотека** на диаграмму и поместите его поверх плана терминала аэропорта. По умолчанию этот элемент имеет две точки сервиса и две линии очереди, ведущие к этим сервисам.



- Откройте свойства элемента **Сервис с очередями**, назовите эту фигуру `scpServices` (`scp` – сокращение от *security check points*) и смените **Тип сервиса** на **Линейный**.



После того, как вы смените тип сервиса с точечного на линейный, точки обслуживания сменят свою форму на линии.

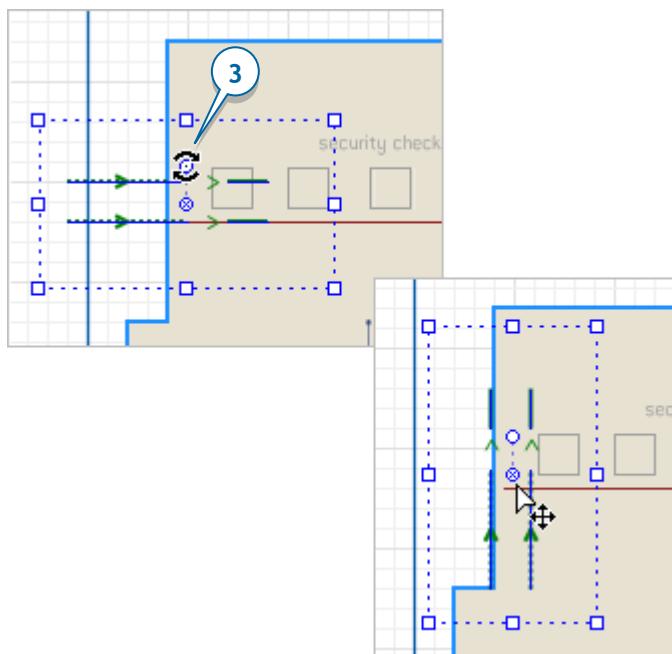
### *Линейные и точечные сервисы в пешеходных моделях*

Существует два типа сервисов в пешеходных моделях: *линейные* и *точечные*.

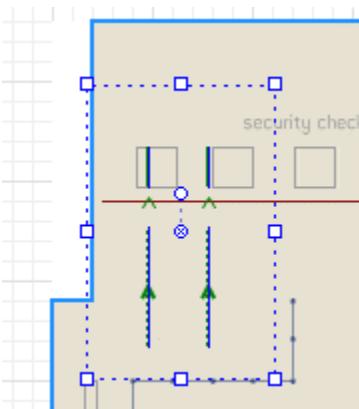
- *Линейный* сервис задается линией. Пешеход начинает процедуру обслуживания в начальной точке линии и затем продвигается к ее конечной точке, откуда он может покинуть сервис. Примеры линейного сервиса: турникет, рамка металлодетектора.
- *Точечные* сервисы задаются точкой. Во время обслуживания пешеход будет находиться у заданной точки сервиса. Примеры точечного сервиса: билетная касса, банкомат.

Мы используем линейные сервисы, потому что хотим, чтобы пассажир следовал вдоль линии сервиса и таким образом проходил через рамку металлодетектора, как это и происходит при предполетном досмотре в аэропорту. Нам будет нужно развернуть и переместить линии сервисов так, чтобы они пересекали предполагаемые места расположения рамок металлодетекторов на плане аэропорта.

3. Поверните сервисы с помощью круглого маркера, расположенного чуть выше центра фигуры.



4. Переместите фигуру так, чтобы линия первого сервиса пересекала квадрат, обозначающий рамку металлодетектора.

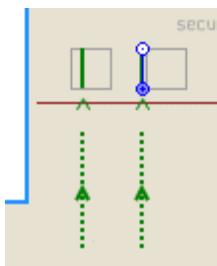


## Как перемещать элементы, игнорируя сетку

Если вам нужно переместить элемент на графической диаграмме без его привязки к сетке, то вы можете нажать клавишу Alt на клавиатуре и переместить объект, не отпуская эту клавишу. Либо же отключите привязку к сетке с помощью кнопки панели управления Включить/Отключить сетку.



5. Выделите вторую линию сервиса.



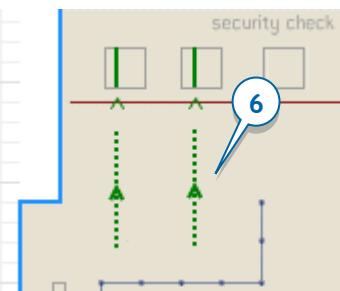
## Сложные фигуры разметки пространства

Некоторые фигуры разметки пространства состоят из нескольких отдельных фигур. Так, например, фигура **Сервис с очередями** состоит из фигур **Сервис** и **Очередь**, а фигура **Сервис с областью** состоит из фигур **Сервис** и **Прямоугольная область**.

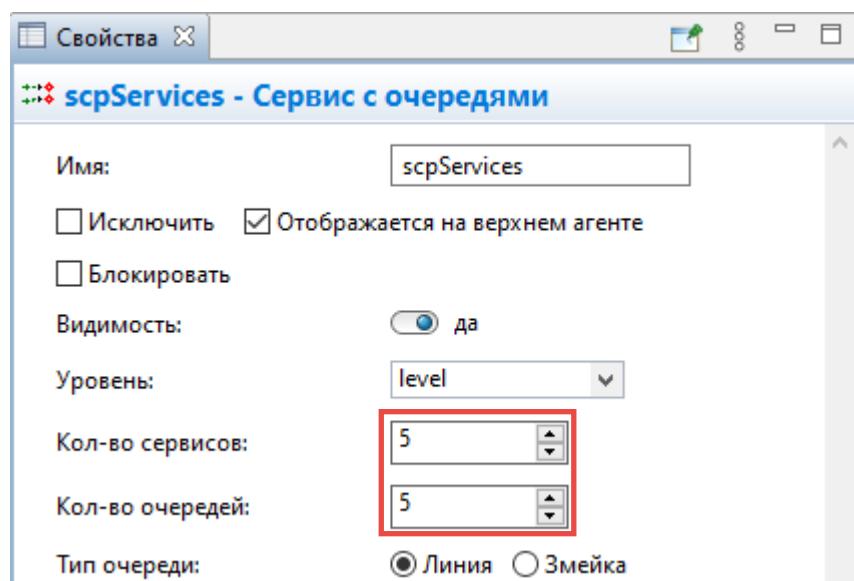
Обратите внимание на правила выделения фигур, входящих в состав сложных фигур разметки пространства:

- Первый щелчок мыши по фигуре выделит саму сложную фигуру разметки пространства (**Сервис с очередями**).
- После того, как вы выделите сложную фигуру разметки пространства, вы можете выделить любую простую фигуру, входящую в ее состав (в данном случае - **Сервис** или **Очередь**), щелкнув по ней мышью.

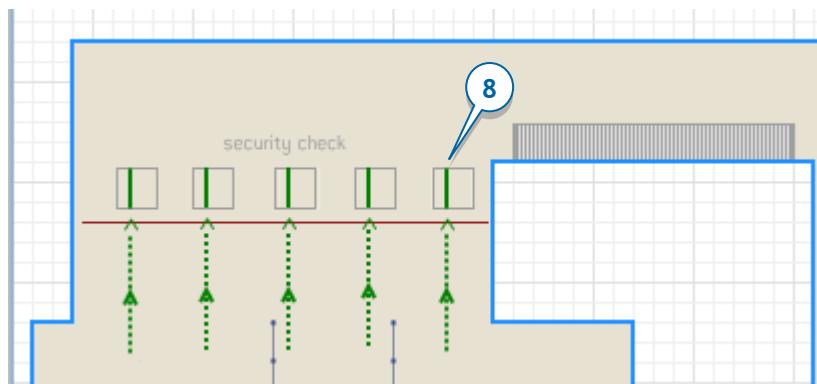
6. Переместите линию второго сервиса в то место плана, где находится второй пункт досмотра. Не забудьте соответственно переместить и линии очереди, ведущие к этим сервисам.



7. Перейдите в свойства фигуры Сервис с очередями и увеличьте Количество сервисов и Количество очередей до 5.



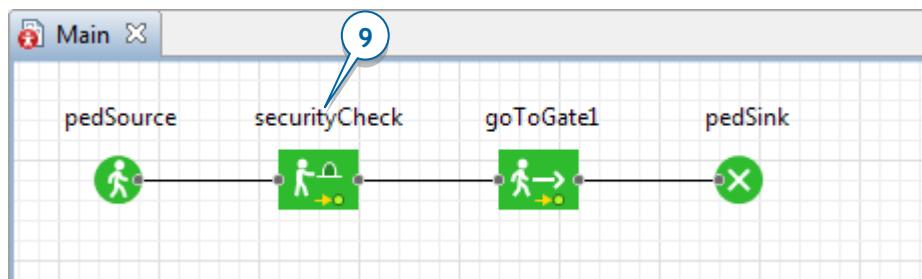
8. Если нужно, поправьте расположение новых линий сервисов и линий очередей. Сервисы и линии должны в итоге быть расположены так, как показано на рисунке ниже.



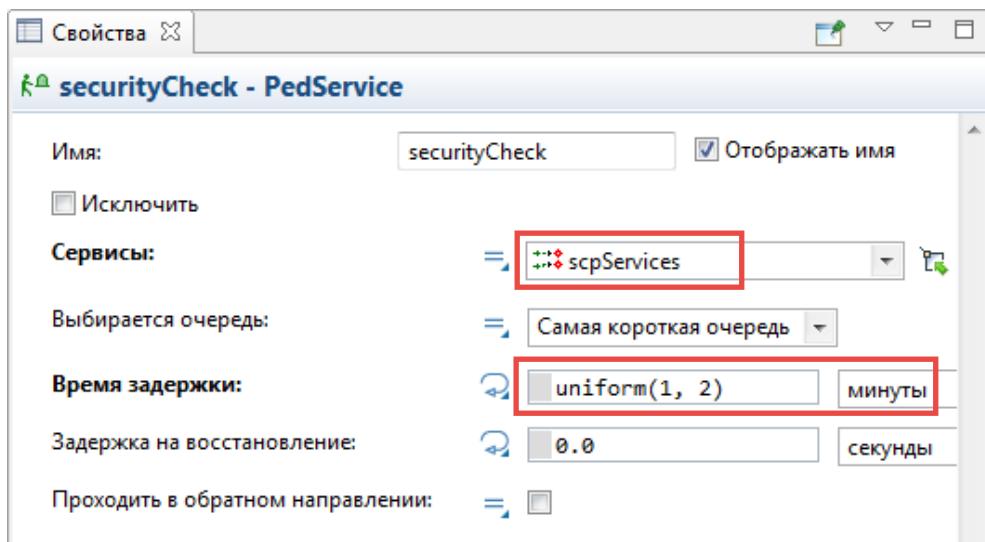
Теперь, когда мы закончили рисование пунктов досмотра с помощью фигур разметки пространства, мы можем добавить и сам процесс прохождения предполетного досмотра в текущую логику модели. Для этого мы воспользуемся специальным блоком **Пешеходной библиотеки PedService**, в свойствах которого мы укажем наш элемент разметки пространства **Сервис с очередями scpServices**.

Давайте добавим еще один блок диаграммы процесса, который будет моделировать предполетный досмотр пассажиров.

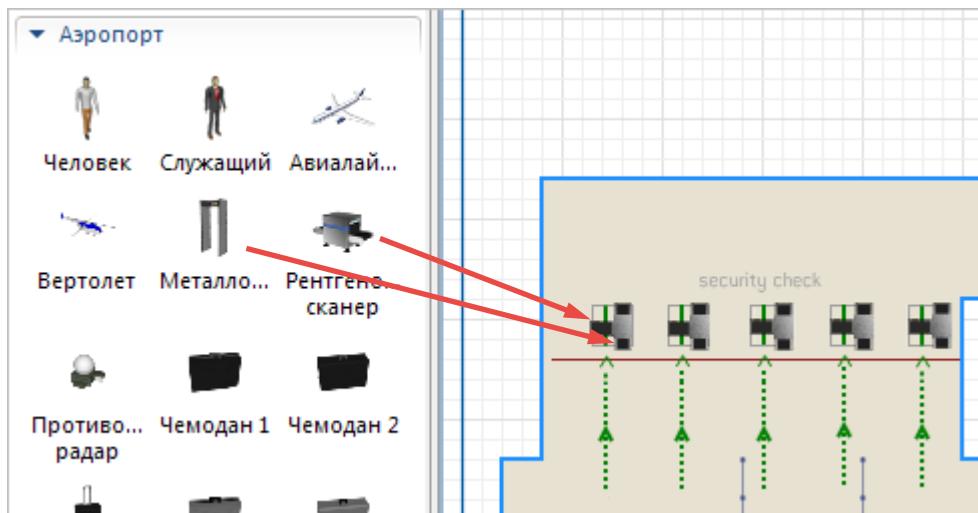
9. Добавьте блок **PedService**. Поместите его в диаграмму процесса между блоками **PedSource** и **PedGoTo**. Теперь пешеходы будут проходить через заданные пункты сервиса (в нашем случае это пункты досмотра). Назовите этот блок *securityCheck*.



10. Откройте свойства блока *securityCheck*. В поле **Сервисы** выберите *scpServices* (имя добавленной ранее фигуры разметки пространства).



11. Мы считаем, что процедура предполетного досмотра в среднем занимает от одной до двух минут, поэтому введите *uniform(1, 2)* в поле Время задержки и выберите минуты в поле справа.
12. Теперь давайте нарисуем пять пунктов досмотра с помощью объектов Металлодетектор и Рентгеновский сканер из раздела Аэропорт палитры 3D Объекты. После того, как вы добавите объекты Рентгеновский сканер, измените их Масштаб на 75%.



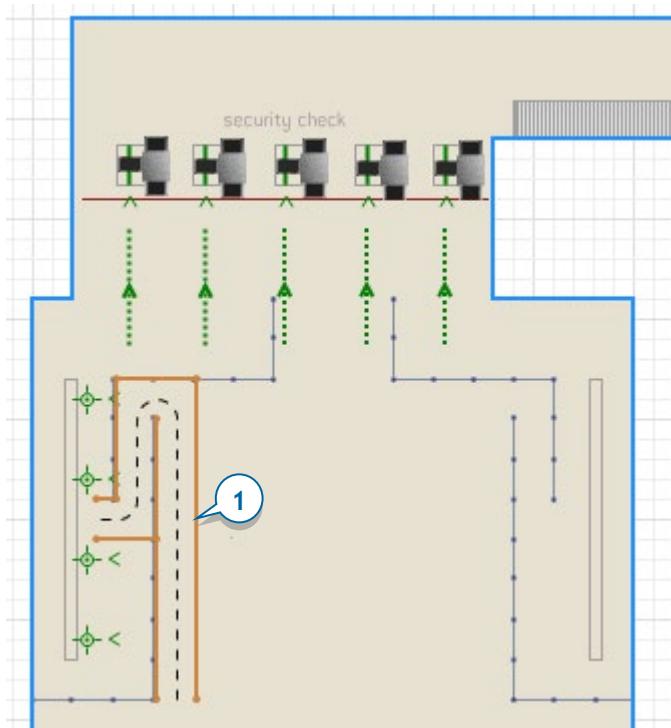
**13.** Запустите модель. Вы увидите, что теперь пассажиры проходят процедуру предполетного досмотра.



## Фаза 4. Добавление стоек регистрации

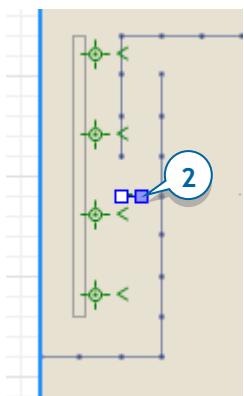
Пассажиры могут регистрироваться на рейсы различными способами – заранее (онлайн-регистрация) или привычным образом - в аэропорту у стоек регистрации. Давайте добавим в нашу модель стойки регистрации на рейсы, и направим туда тех пассажиров, кто не зарегистрировался на свой рейс заблаговременно.

- Добавьте еще один объект **Сервис с очередями** на план аэропорта, чтобы промоделировать стойки регистрации. В этот раз нам необходимо задать точечный тип сервиса, состоящий из четырех пунктов сервиса и одной общей очереди. Назовите этот элемент *checkInServices*.

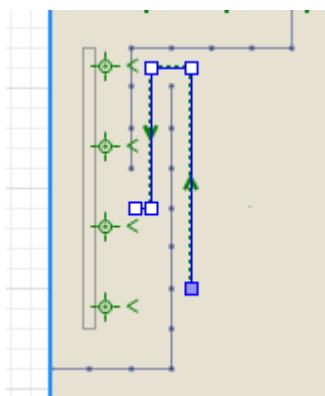


Переместите диаграмму процесса, собранную из блоков Пешеходной библиотеки, выше оси X, вынеся ее тем самым за пределы области, видимой во время работы модели. Если во время работы модели вам понадобится изучить ход моделируемых процессов с помощью этой диаграммы, вы всегда можете посмотреть на нее, просто переместив полотно окна модели с помощью мыши.

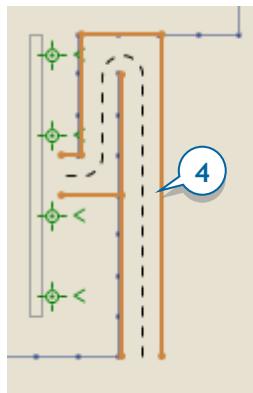
2. Разместите фигуру сервисов так, как показано на рисунке ниже. Обратите внимание, что мы начинаем рисование очереди, ограниченной ленточными барьерами с того, что помещаем начальную точку линии очереди у подхода к стойке регистрации. Вторую точку линии нужно поместить точно посередине ограниченного барьера прохода, в месте первого изгиба очереди.



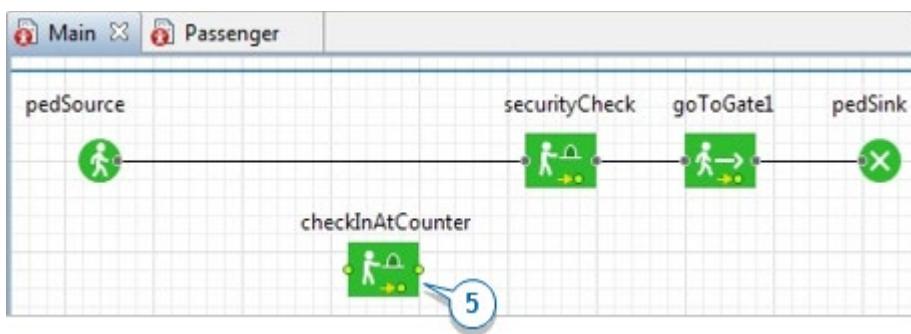
3. Теперь давайте продолжим рисование линии очереди, чтобы она следовала обозначенным на плане терминала ленточным барьерам. Сделайте щелчок правой кнопкой мыши по очереди и выберите пункт контекстного меню **Добавить точки**. Добавьте новые точки, щелкнув мышью в точках, выделенных на рисунке ниже. Завершите рисование, сделав двойной щелчок мышью в точке окончания очереди- «змейки». В итоге должна будет получиться «змейка» следующей формы:



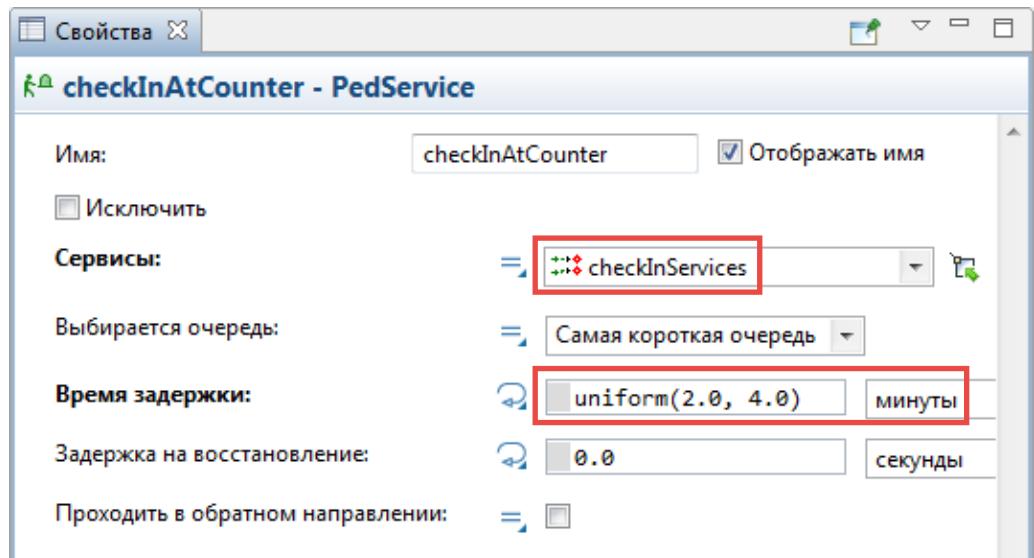
4. В свойствах этой фигуры **Сервис с очередями**, измените **Тип очереди** на **Змейка**. Вы увидите, что теперь у очереди есть границы. На сцене 3D анимации они будут отображаться в виде ленточных барьеров.



5. Добавьте еще один объект **PedService** и назовите его *checkInAtCounter*.



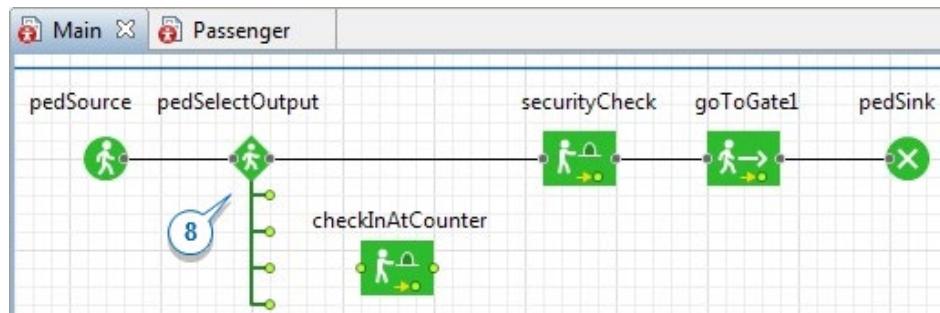
6. В свойствах объекта выберите сервис *checkInServices* в поле **Сервисы**.



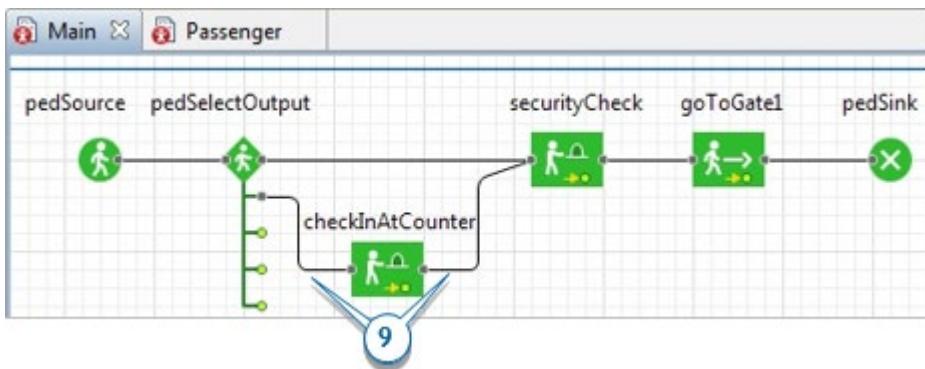
- Поскольку мы полагаем, что в среднем процесс регистрации на рейс занимает от двух до четырех минут, то давайте введем в поле **Время задержки** вызов функции равномерного распределения *uniform(2, 4)* и выберем **минуты** из расположенного справа списка.

Поскольку пассажиры могут регистрироваться на рейсы различными способами, нам нужно направить разные группы пассажиров в разные подпроцессы.

- Чтобы направлять пешеходов в различные ветви диаграммы процесса, мы используем объект **PedSelectOutput**.



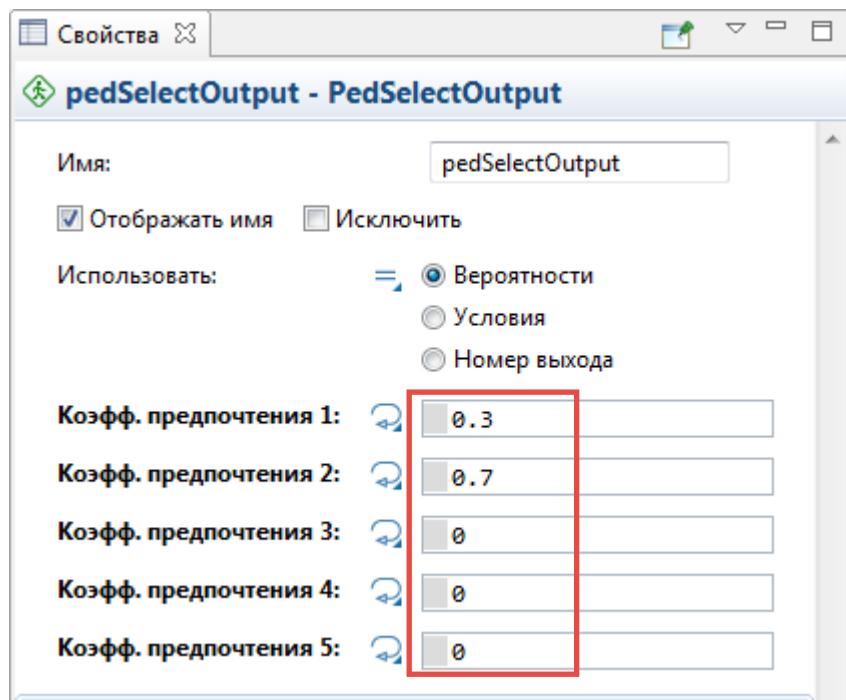
- Соедините блок *checkInAtCounter* с текущей диаграммой процесса так, как показано на рисунке.



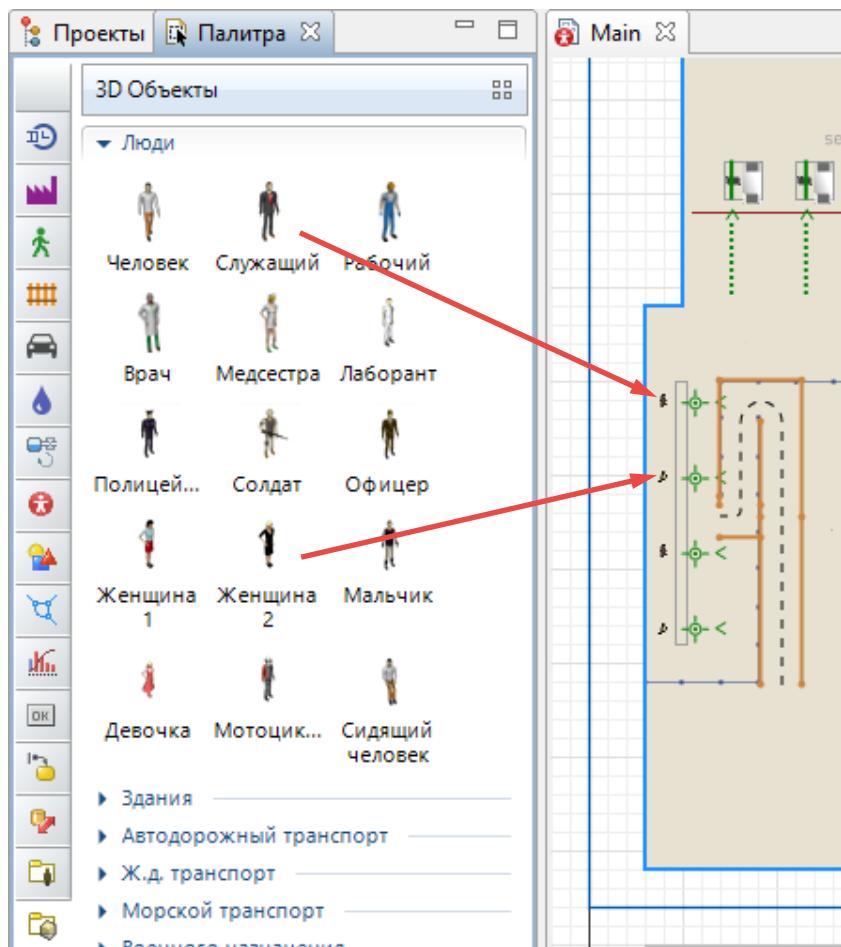
### Как нарисовать соединители сложной формы

Когда вы добавляете блоки из палитры на графическую диаграмму, ближайшие друг к другу порты расположенных поблизости блоков будут соединяться автоматически. Если же вам понадобится нарисовать соединитель более сложной формы (например, как на рисунке выше), то это можно сделать, нарисовав такой соединитель вручную. Для этого нужно сделать двойной щелчок по первому соединяемому поту, затем щелкнуть по диаграмме в точках изгиба соединителя, и закончить рисование, сделав щелчок по второму соединяемому порту. Чтобы удалить точку изгиба соединителя, сделайте двойной щелчок по этой точке.

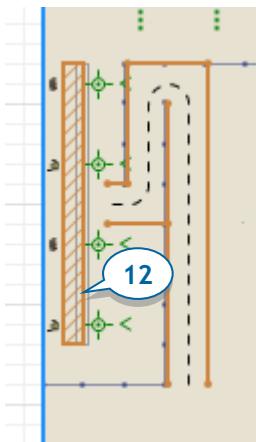
10. Мы полагаем, что 30 процентов пассажиров регистрируются онлайн, а 70 процентов – у стоек регистрации. Чтобы задать такое деление потока пассажиров, задайте в свойствах блока *pedSelectOutput* Коэффи. предпочтения 1 равным 0.3, а Коэффи. предпочтения 2 равным 0.7. При этих значениях блок будет перенаправлять 30 процентов пешеходов в верхнюю ветвь диаграммы процесса, а 70 – в нижнюю. Нужно будет также задать значение 0 в полях Коэффи. предпочтения 3, Коэффи. предпочтения 4 и Коэффи. предпочтения 5, чтобы предотвратить перенаправление пешеходов в три нижних порта блока, которые не соединены ни с какими другими блоками.



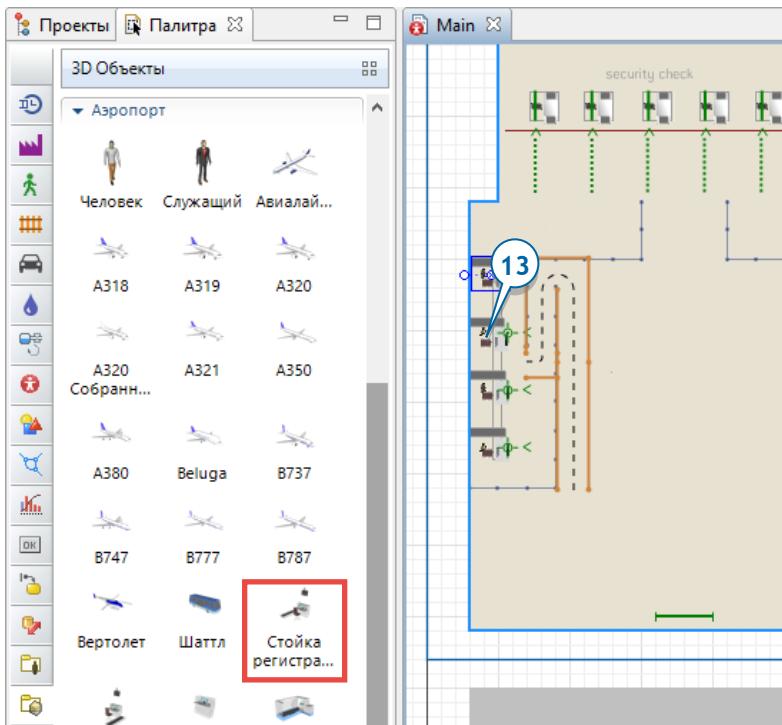
11. Теперь давайте добавим 3D модели объектов служащих авиакомпании у стоек регистрации. Добавьте на диаграмму четыре фигуры, используя объекты из раздела **Люди** палитры **3D Объекты: Служащий и Женщина 2**.



- 12.** Сделайте область стоек регистрации недоступной для прохождения пешеходов. Для этого добавьте элемент **Прямоугольная стена** из раздела **Разметка пространства** палитры **Пешеходная библиотека** и поместите ее так, как показано на рисунке ниже. В свойствах стены, смените значение свойства **Видимость** на **нет**, чтобы сделать эту фигуру невидимой во время работы модели (иначе эта стена будет отображаться на трехмерной анимации в виде параллелепипеда).



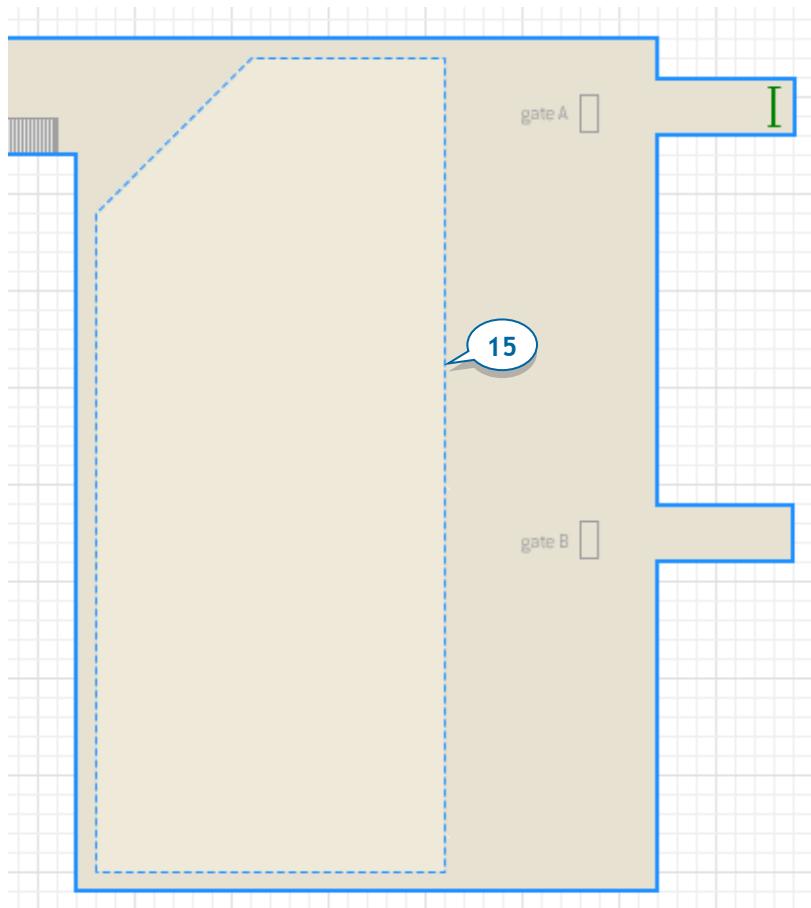
13. Добавьте в область стоек регистрации четыре 3D объекта **Стойка регистрации** из секции **Аэропорт** палитры **3D Объекты**. Поверните эти фигуры, выбрав в поле **Поворот, Z: -90 градусов** (в секции свойств **Расположение**).



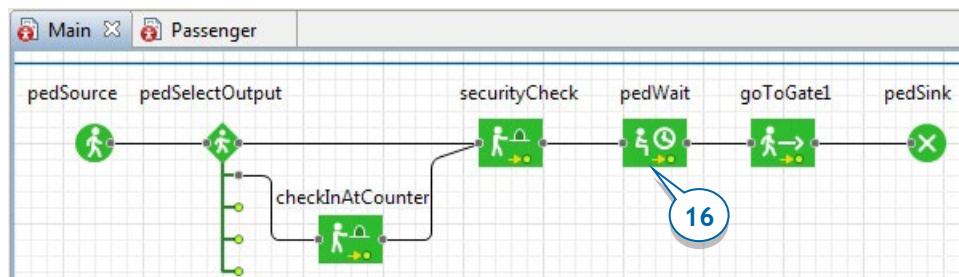
- 14.** Запустите модель. Вы увидите, что теперь многие пассажиры вначале регистрируются на рейс у стойки регистрации, и только затем проходят к выходу на посадку.

Мы хотим, чтобы прежде чем отправиться на посадку, пассажиры проводили определенное время в зоне перед гейтами. Для этого нам нужно будет нарисовать область ожидания, в которой пассажиры будут ожидать начала посадки на рейс, а затем добавить в диаграмму процесса блок (**PedWait**), который будет моделировать само ожидание.

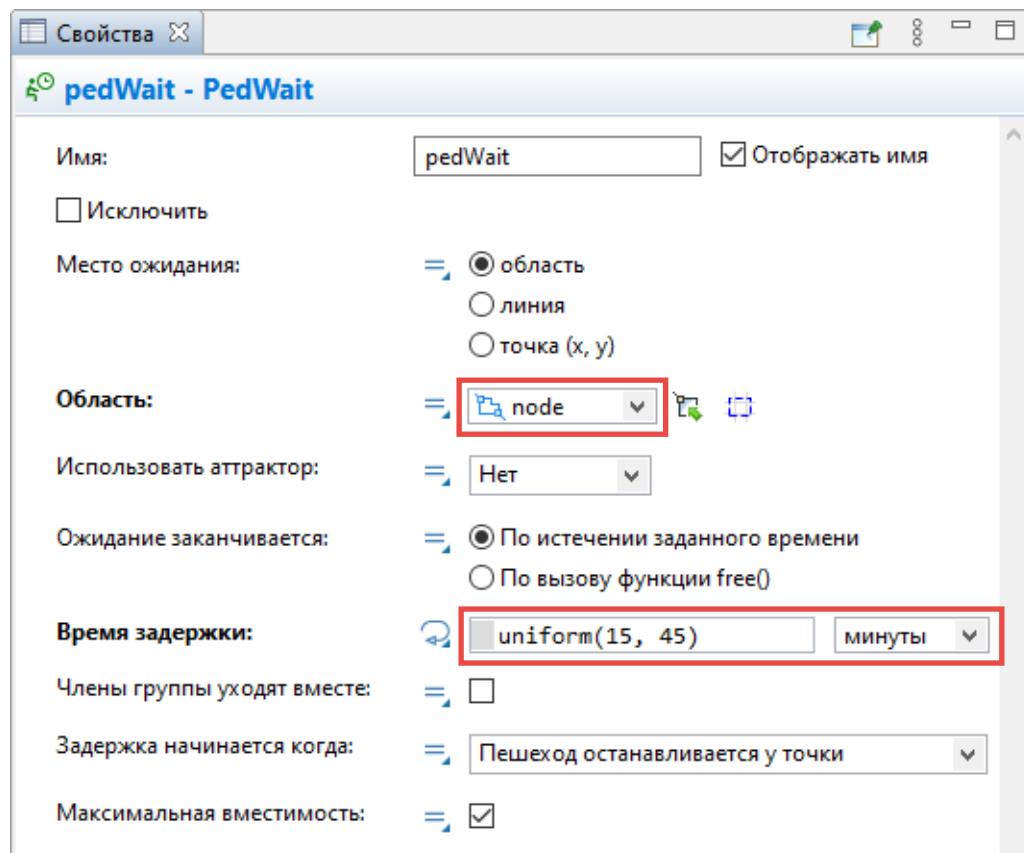
- 15.** Нарисуйте область ожидания перед выходами на посадку с помощью элемента разметки пространства **Многоугольный узел** (совет - используйте режим рисования). Последовательно щелкайте мышью в углах области на плане терминала аэропорта, последовательно добавляя их по направлению часовой стрелки. Если потребуется, вы можете отредактировать получившуюся фигуру уже после того, как вы закончите ее рисование.



16. Добавьте блок PedWait  в диаграмму процесса и поместите его между блоками PedService и PedGoTo.



- 17.** Измените свойства блока `pedWait`. Выберите в поле **Область** нарисованный нами ранее многоугольный узел `node`, а затем укажите время ожидания в свойстве **Время задержки:** `uniform(15, 45)` минут.



- 18.** Снова запустите модель. Вы сможете наблюдать, как пассажиры на какое-то время задерживаются в области ожидания и только затем отправляются на посадку.



Вы можете добавить дополнительные стойки в заготовленной для этого части на плане терминала (это потребует также соответствующего изменения свойств блока **PedSelectOutput**, который будет разделять поток пешеходов на большее количество подпотоков).



Вопрос: как промоделировать автоматы для самостоятельной регистрации пассажиров?

## Фаза 5. Моделирование посадки на самолет

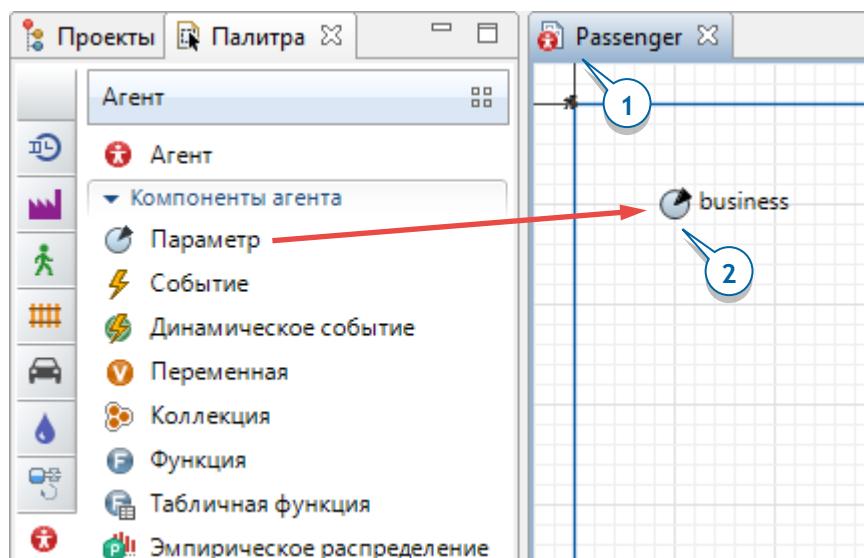
Пассажиры будут ожидать начала посадки на рейс в области ожидания (общей для обоих выходов на посадку). Когда начнется посадка, они должны будут пройти процедуру проверки посадочных талонов, после чего они смогут пройти на борт самолета.

К стойке проверки посадочных талонов и документов ведут две очереди – одна для пассажиров бизнес-класса, другая – для пассажиров эконом-класса. Проверка документов каждого пассажира занимает в среднем от 1 до 3 секунд.

1. Откройте диаграмму типа агента *Passenger*, сделав двойной щелчок по элементу *Passenger* в панели **Проекты**.

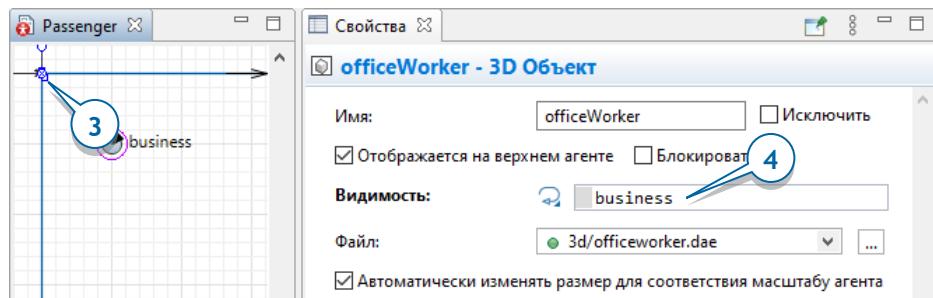
Нам нужно как-то отличать в нашей модели пассажиров бизнес-класса от пассажиров эконом-класса. Для этого потребуется хранить дополнительную информацию о пассажире в заданном нами ранее типе пешехода.

2. Добавьте **Параметр**  из палитры **Агент**  . Назовите этот параметр *business* и выберите Тип: *boolean*. Этот параметр будет определять, летит ли данный пассажир бизнес-классом. Если значение параметра равно *true*, то это пассажир бизнес-класса, в противном случае (если значение параметра равно *false*), это пассажир экономического класса.



Мы хотим, чтобы пассажиров, летящих разными классами, можно было легко различить визуально на анимации модели. Для этого мы будем использовать две разные 3D модели людей для отображения пассажиров бизнес-класса и эконом-класса.

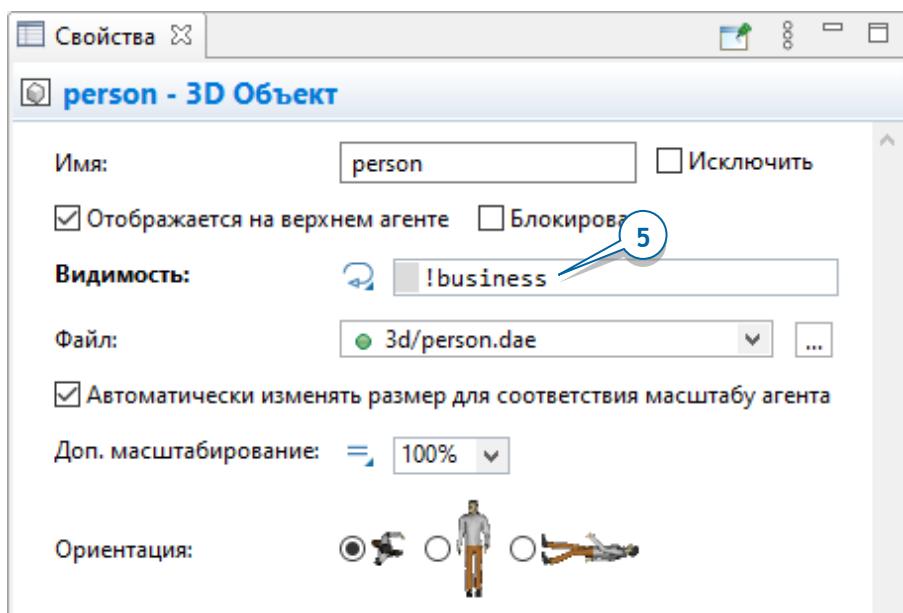
- Добавьте 3D объект **Служащий**, чтобы задать фигуру анимации пассажира бизнес-класса. Поместите эту фигуру в точку начала координат (0,0) диаграммы типа пешехода *Passenger*.



- Измените свойство **Видимость** для обоих 3D объектов на диаграмме типа пешехода *Passenger*. Вначале щелкните по фигуре анимации *Служащий*. Мы хотим, чтобы эта фигура была видна только тогда, когда это пассажир бизнес-класса, то есть, значение его параметра *business* равно *true*. Для этого переключите поле задания значения свойства **Видимость** в режим задания динамического выражения, щелкнув по расположенному слева от поля значку , а затем введите *business* в этом поле.

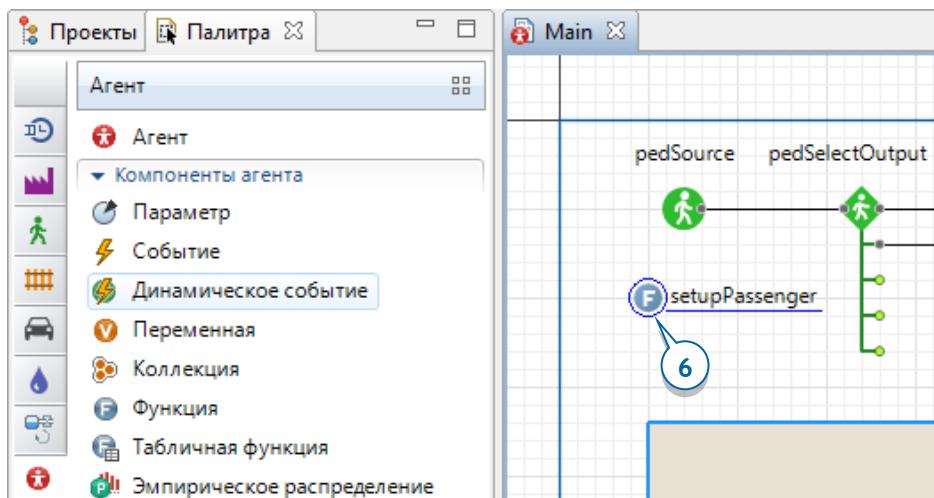
- Теперь выберите фигуру *person* (это можно сделать из дерева в панели **Проекты**, фигура находится в ветке **Презентация** типа агента *Passenger*), и аналогично введите следующее динамическое выражение в поле **Видимость:** `! business`. Теперь эта фигура будет отображаться только в том случае, если данный пассажир летит эконом-классом.

Символ `!` является булевским оператором НЕ в языке Java. Выражение `! business` возвращает *true*, когда значение параметра *business* НЕ РАВНО *true* (не истинно), то есть это пассажир не бизнес-класса, а эконом-класса.



Мы будем задавать класс пассажира в момент его создания блоком PedSource.

- Перейдите обратно на диаграмму *Main* и добавьте Функцию из палитры Агент. Назовите эту функцию *setupPassenger*.



- Задайте следующие параметры функции:

- Добавьте аргумент, чтобы с его помощью передать данной функции ссылку на только что созданного пешехода.

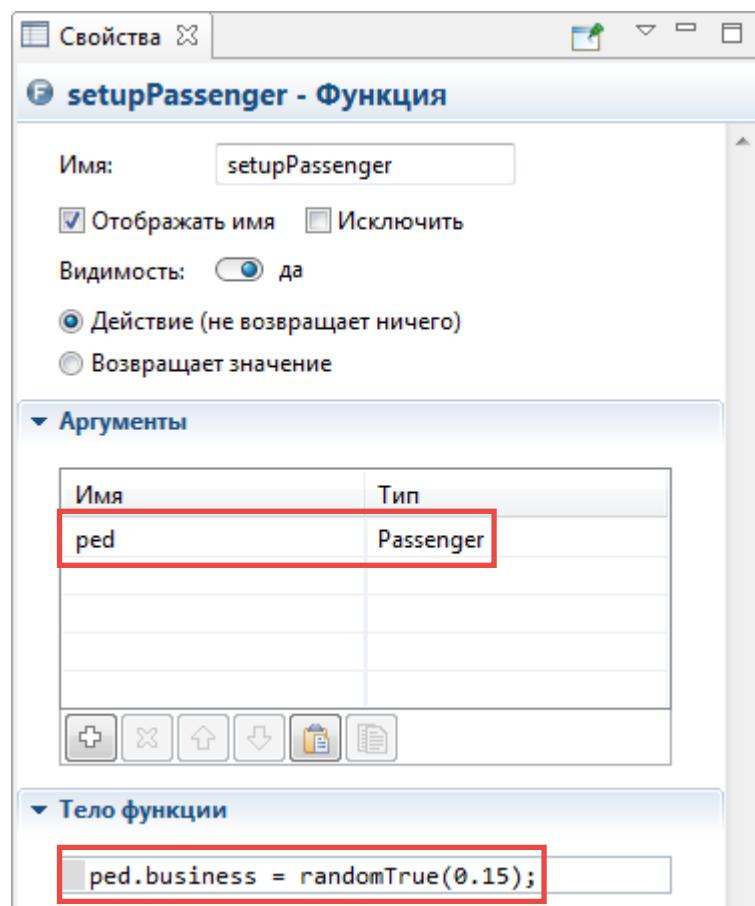
Задайте для этого аргумента:

Имя: *ped*

Тип: *Passenger*

- Ведите следующий код функции:  
*ped.business = randomTrue(0.15);*

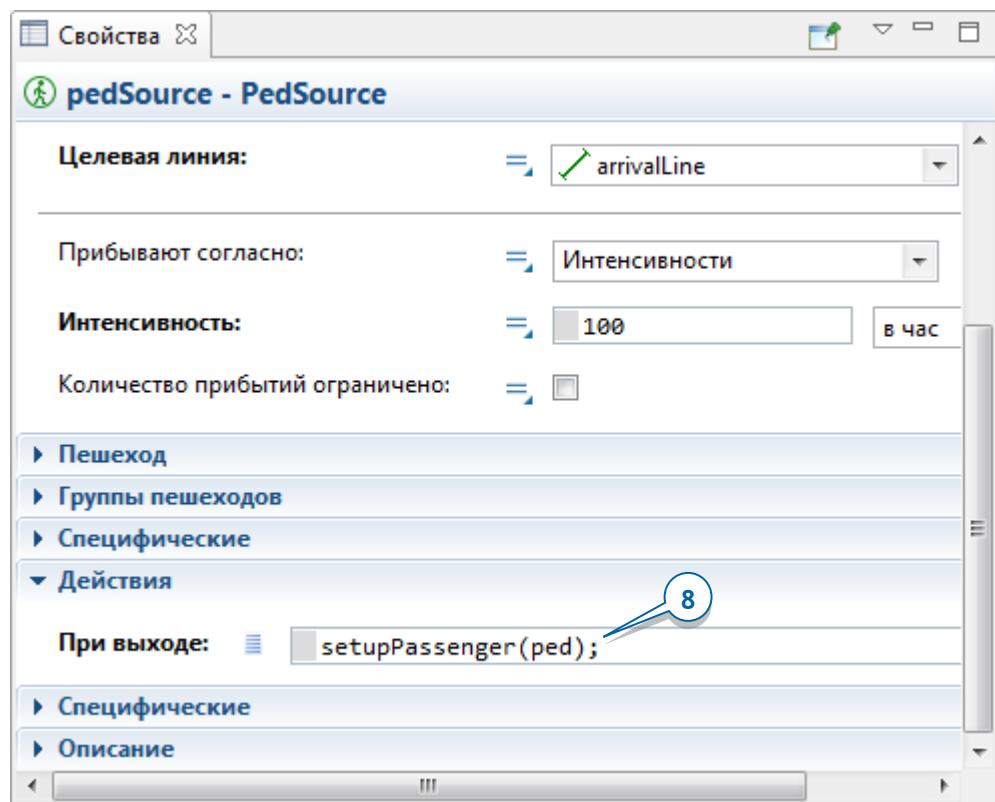
Эта строка будет присваивать полю пешехода *business* значение *true* с вероятностью 15 процентов:



Здесь *ped* – это заданный нами аргумент функции, пешеход типа *Passenger*. Выбрав *Passenger* в качестве типа аргумента, мы можем напрямую обращаться к параметру этого пешехода *business* просто как *ped.business*. Функция *randomTrue(0.15)* возвращает *true* в среднем в 15 процентах случаев, что означает, что в среднем бизнес-классом в нашей модели будет лететь 15 процентов пассажиров.

- Будем вызывать эту функцию в объекте *pedSource* при создании нового пешехода. Разверните секцию свойств **Действия** и введите код в поле параметра **При выходе**:

*setupPassenger(ped);*

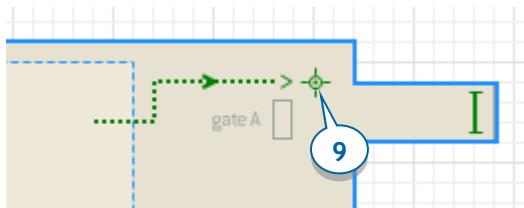


В данном случае мы вызываем нашу функцию *setupPassenger* для каждого созданного пешехода. Аргумент нужен нам для того, чтобы передать функции ссылку на текущего пешехода. Тогда мы сможем выполнить в коде функции необходимые действия с этим пешеходом (в нашем случае – изменить значение его параметра).

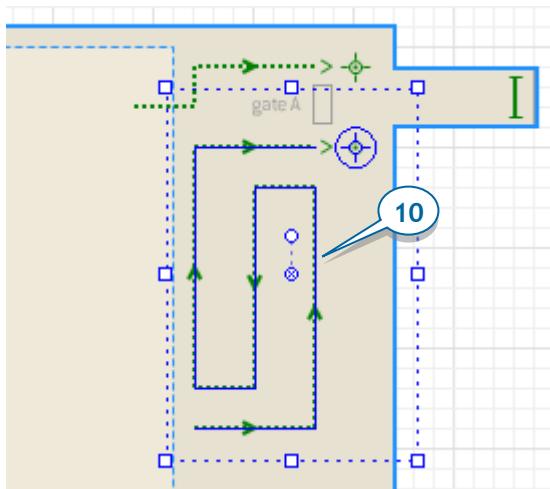
Нарисуйте два сервиса с очередями у верхнего выхода на посадку: один для пассажиров бизнес-класса, другой – для эконом-класса.

Обратите внимание, что это два разных сервиса, а не просто две очереди у одной фигуры.

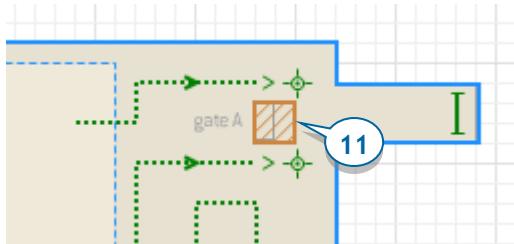
- Добавьте на диаграмму **Сервис с очередями**, который будет отвечать за регистрацию пассажиров бизнес-класса (это должен быть точечный сервис с одной точкой сервиса и одной очередью). Назовите эту фигуру **Сервис с очередями business1**.



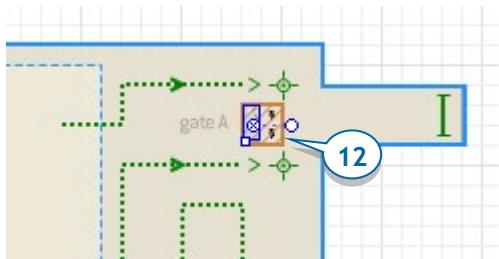
- Добавьте еще один **Сервис с очередями** (как показано на рисунке ниже). Назовите этот сервис *economy1*.



- С помощью элемента **Прямоугольная стена** нарисуйте область стойки проверки документов около выхода на посадку.

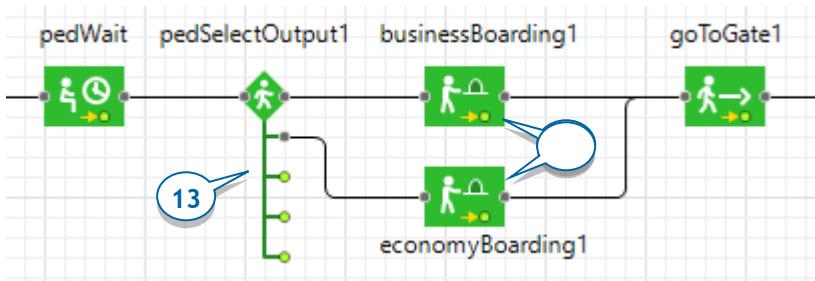


12. Добавьте 3D фигуру Стойка у гейта из секции Аэропорт палитры 3D Объекты. Поверните ее (Поворот Z в секции свойств Расположение: 90 градусов) и добавьте две фигуры служащих за стойку. Чтобы стена не была видна на анимации во время работы модели, смените значение свойства Видимость этой стены на нет.



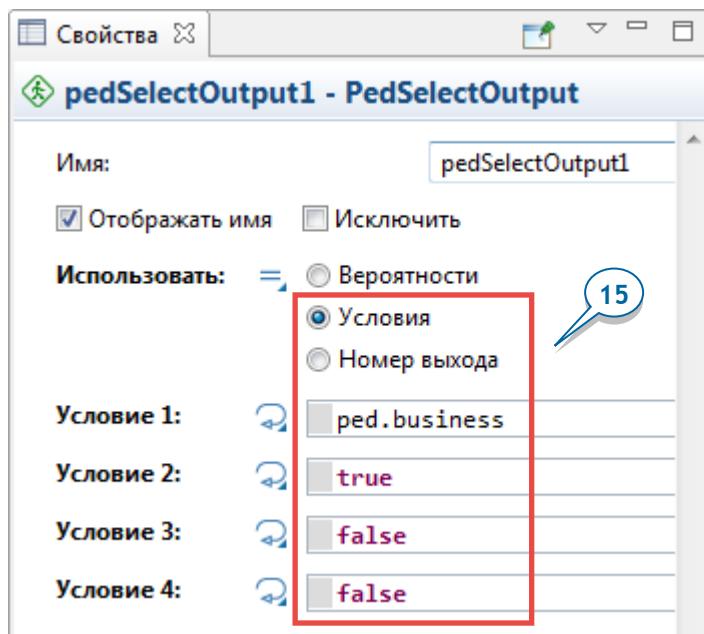
Добавьте новые объекты Пешеходной библиотеки между объектами *pedWait* и *goToGate1*.

13. Добавьте блок **PedSelectOutput** , который будет перенаправлять пассажиров бизнес-класса и пассажиров эконом-класса в разные очереди.



14. Добавьте два блока **PedService** : *businessBoarding1* и *economyBoarding1*. Они будут моделировать процесс проверки посадочных талонов у пассажиров бизнес-класса и эконом-класса соответственно.

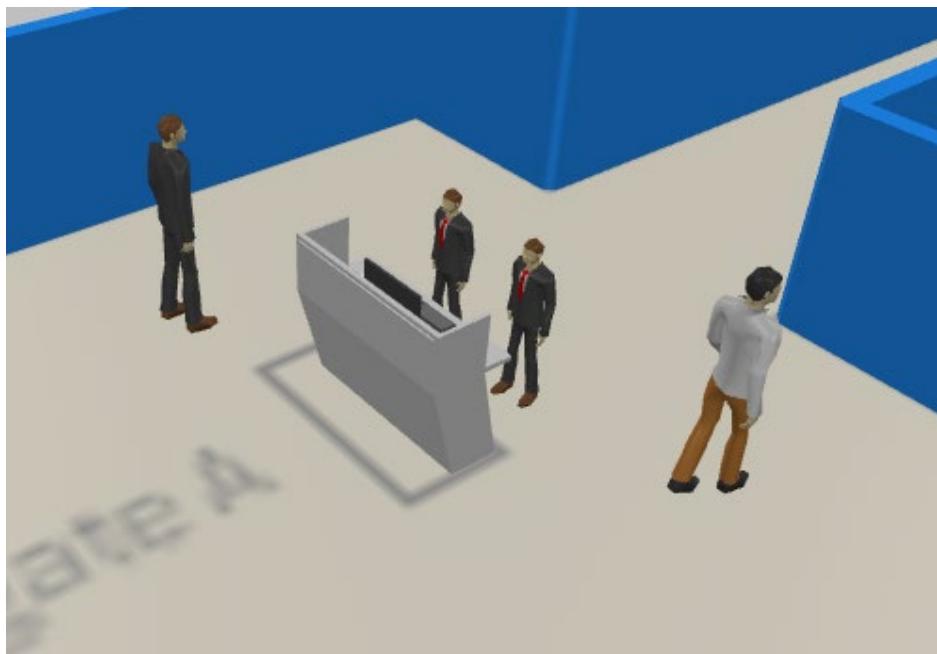
15. Чтобы блок **PedSelectOutput** направлял пассажиров бизнес-класса и эконом-класса в разные очереди, выберите опцию **Использовать: Условия**, и введите *ped.business* в поле **Условие 1**. Это выражение вернет значение *true* («истина») для всех пассажиров бизнес-класса, и вследствие этого эти пешеходы проследуют по той части диаграммы процесса, которая соединена с верхним выходным портом блока, и пройдут на проверку посадочных талонов, встав в очередь приоритетного обслуживания. После того, как вы зададите условия для оставшихся выходных портов блока (*true*, *false*, *false*), все оставшиеся пассажиры (летящие эконом-классом) проследуют во второй выходной порт.



16. В свойствах блока **PedService businessBoarding1**, выберите *business1* в поле **Сервисы**. Поскольку время проверки документов в среднем занимает от двух до пяти секунд, задайте соответствующее значение в поле **Время задержки**.
17. В свойствах блока **economyBoarding1**, выберите *economy1* в поле **Сервисы** и аналогично измените **Время задержки**.
18. Запустите модель. Вы увидите, как служащие аэропорта проверяют посадочные талоны пассажиров. Пассажиры встают в одну из двух очередей

## 250 AnyLogic 8 за три дня

в зависимости от того, имеют ли они право на приоритетное обслуживание или нет.

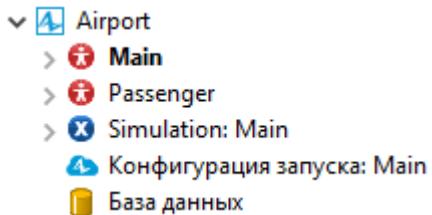


## Фаза 6. Считывание данных о рейсах из файла MS Excel

Теперь мы добавим в нашу модель рейсы, считав расписание вылетов из таблицы Excel.

### База данных AnyLogic

В каждой модели AnyLogic есть своя встроенная база данных, в которой могут храниться значения входных параметров модели, а также результаты работы модели.

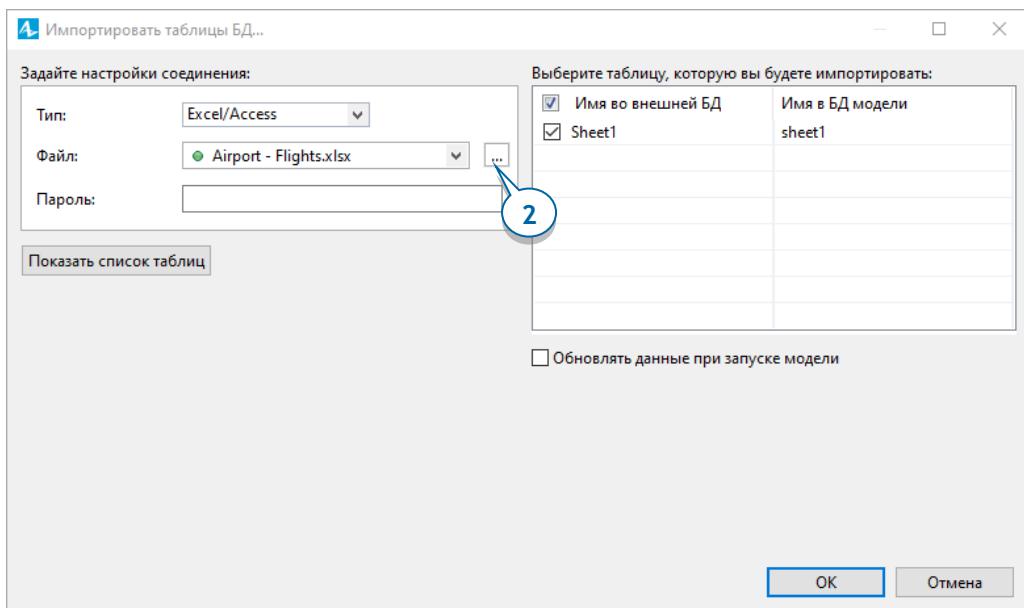


База данных AnyLogic позволяет:

- Конфигурировать модель согласно заданным в БД параметрам.
- Параметризовать популяции агентов.
- Задавать частоту прибытия агентов-заявок в процессных моделях.
- Импортировать данные из других БД или таблиц Excel и хранить их в доступной форме.
- Записывать логи (журналы выполнения) модели, в которые добавляется информация обо всех произошедших событиях в диаграммах процессов, диаграммах состояний, статистика по взаимодействию и перемещению агентов, и т.д.
- Сохранять и экспортить статистику, хранящуюся в наборах данных.
- Экспортить данные в электронные таблицы MS Excel.
- Создавать резервные копии БД.

Мы покажем, как импортировать данные из внешней базы данных во встроенную БД модели.

1. В дереве **Проекты**, щелкните правой кнопкой мыши по элементу **База данных**  и выберите **Импортировать таблицы БД...** из контекстного меню.
2. Вы увидите диалоговое окно **Импортировать таблицы БД**. Выберите файл базы данных, хранящий необходимые для нашего проекта данные. Щелкните по кнопке  и выберите файл *Flights.xlsx* (его можно найти в папке *каталог установки AnyLogic/resources/AnyLogic in 3 days/Airport*).



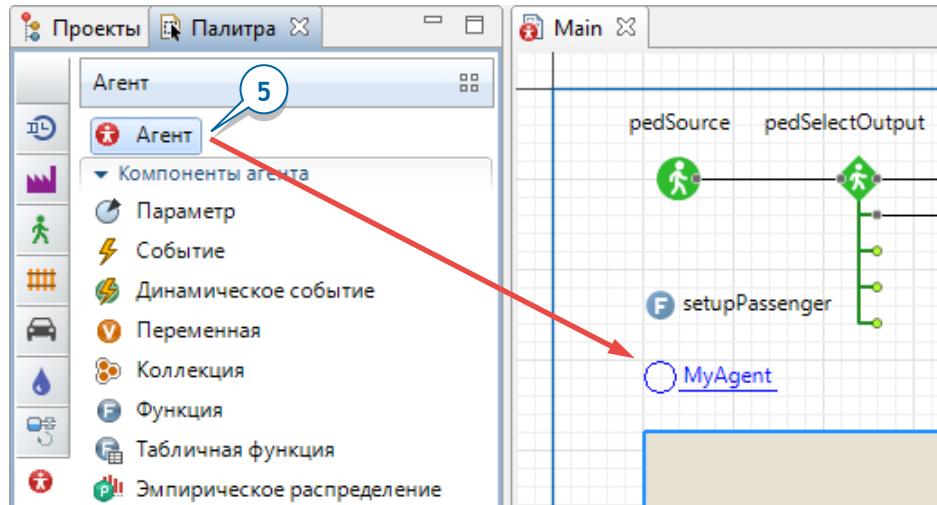
3. В расположенный в правой части окна список **Выберите таблицу, которую вы будете импортировать** будет добавлена таблица *Sheet1*. Щелкните по кнопке **OK**, чтобы импортировать данные из этой таблицы БД в вашу модель.
4. Вы увидите, что в ветке **База данных**  в панели **Проекты** появится элемент *sheet1*. Сделайте двойной щелчок по этому элементу. При этом в центре рабочего пространства AnyLogic откроется редактор таблицы. Здесь вы сможете увидеть данные, предварительно считанные из таблицы Excel в эту таблицу встроенной базы данных AnyLogic. Таблица содержит три столбца, хранящих следующую информацию: пункт назначения рейса, время отправления и номер выхода на посадку (гейта).

The screenshot shows the AnyLogic Main window. On the left, the Project Explorer displays a project named 'Airport\*' containing 'Main', 'Passenger', 'Simulation: Main', and 'База данных sheet1'. A red arrow points from the 'sheet1' entry to the right pane. On the right, the 'Main' sheet contains a table with 12 rows of flight information:

	destination	departure_time	gate
1	Chicago	21-12-2014 13:20:00	2
2	Twin Peaks	21-12-2014 13:55:00	1
3	New York	21-12-2014 15:10:00	2
4	Dogville	21-12-2014 16:10:00	1
5	Las Vegas	21-12-2014 16:30:00	2
6	Zabriskie Point	21-12-2014 17:15:00	1
7	San Francisco	21-12-2014 17:45:00	2
8	Mesa Verde	21-12-2014 18:40:00	1
9	Rimini	21-12-2014 19:25:00	2
10	Mandalay	21-12-2014 20:30:00	1
11	Washington	21-12-2014 21:30:00	2
12	Springfield	21-12-2014 22:20:00	1
*			

Теперь мы создадим новый тип агента: *Flight*.

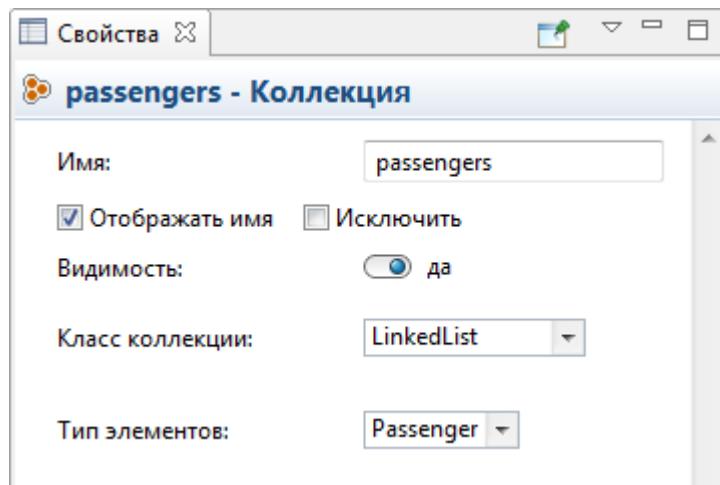
5. Добавьте пустую популяцию агентов типа *Flight*, перетащив элемент **Агент** из палитры **Агент** на диаграмму *Main*.



6. Вы увидите окно Мастера создания агентов. Выполните следующие шаги:

- На первой странице Мастера, выберите пункт **Популяция агентов**.
- Выберите пункт **Я хочу создать новый тип агента**. Щелкните по кнопке **Далее**.

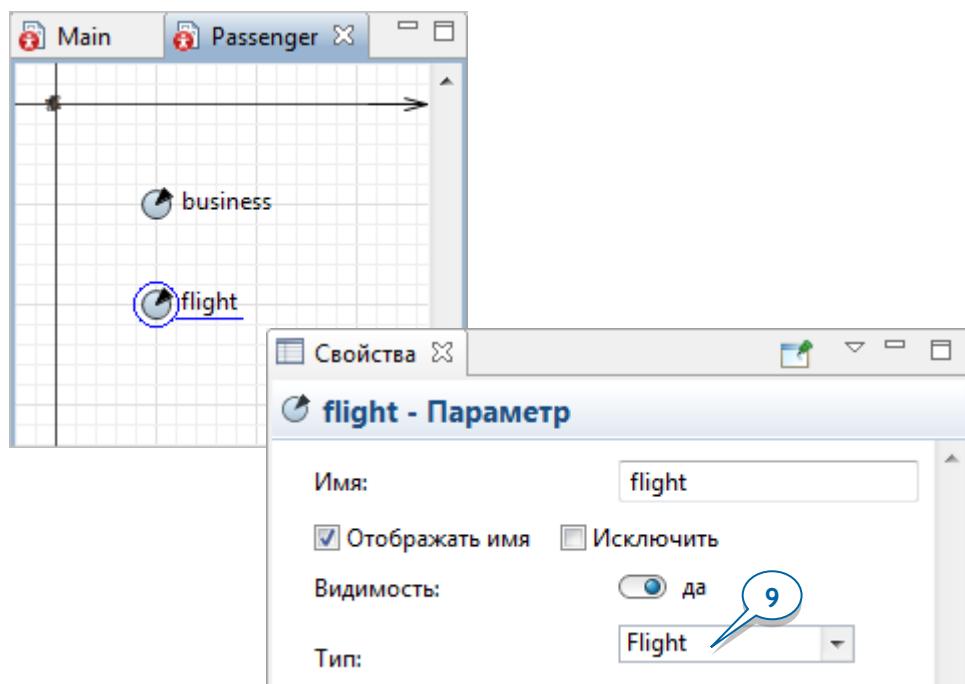
- c. Задайте **Имя типа агента:** *Flight*. Имя популяции автоматически изменится на *flights*. Выберите опцию **Использовать таблицу базы данных** и щелкните по кнопке **Далее**.
  - d. На следующей странице Мастера, оставьте заданные по умолчанию настройки (мы будем считывать данные из таблицы *sheet1* базы данных нашей модели). Щелкните по кнопке **Далее**.
  - e. На следующей странице Мастера вам будет предложено создать параметры *destination*, *departureTime* и *gate* для каждого агента типа *Flight*. Это как раз то, что нам нужно. Щелкните по кнопке **Далее**.
  - f. Поскольку мы не планируем отображать рейсы на анимации, на странице выбора анимации агента выберите опцию **Нет**.
  - g. Щелкните по кнопке **Готово**.
7. В панели **Проекты**, сделайте двойной щелчок мышью по элементу *Flight*. На открывшейся диаграмме типа агента *Flight* вы должны будете увидеть три параметра:
- *destination*. Тип: *String*.
  - *departureTime*. Тип: *Date*.
  - *gate*. Тип: *int*.
- Эти параметры будут использоваться для хранения времени вылета рейса, пункта назначения, а также номера выхода на посадку, используемого этим рейсом.
8. Добавьте на диаграмму *Flight* элемент **Коллекция**  из палитры **Агент**. Назовите эту коллекцию *passengers* и смените **Класс коллекции** на *LinkedList*, а **Тип элементов** на *Passenger*. В этой коллекции будет храниться список пассажиров, приобретших билеты на рейс.



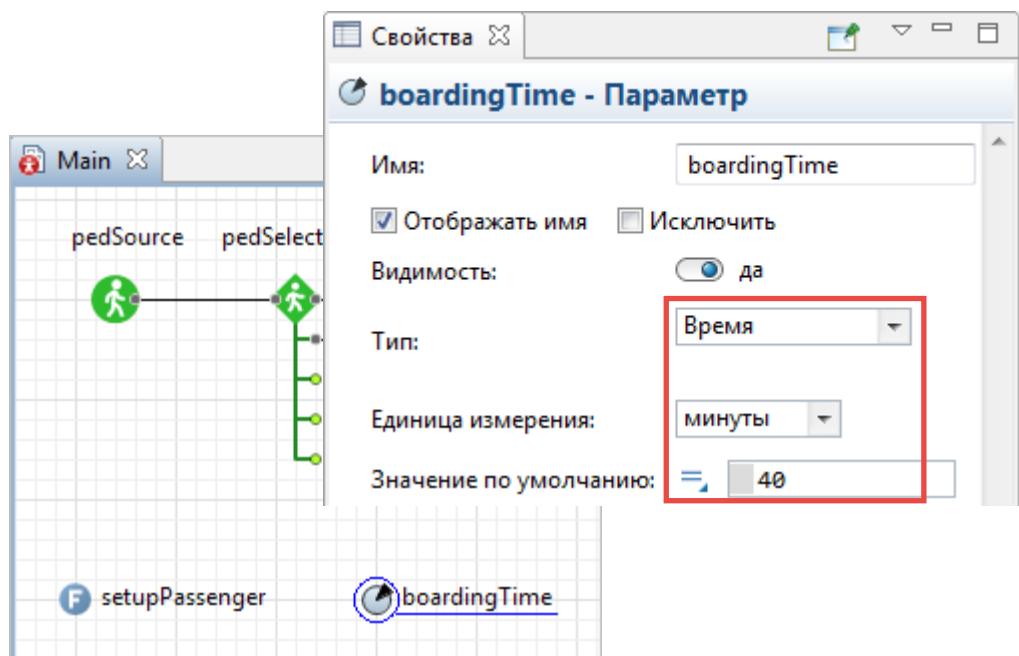
## Коллекции

Коллекция представляет собой структуру данных, объединяющую вместе несколько однотипных элементов, и предоставляющую удобные механизмы для доступа к ним, управления этими элементами и сбору агрегированной статистики по элементам. Обычно элементы коллекции образуют логическую группу – это может быть список контактов человека, список невыполненных заказов и т.д.

9. Теперь, когда мы создали тип агента *Flight*, мы добавим параметр *flight* на диаграмму пешехода *Passenger* и сменим Тип параметра на *Flight*. Этот параметр будет хранить информацию о рейсе пассажира (с технической точки зрения, это ссылка на объект *Flight*, задающий этот рейс).

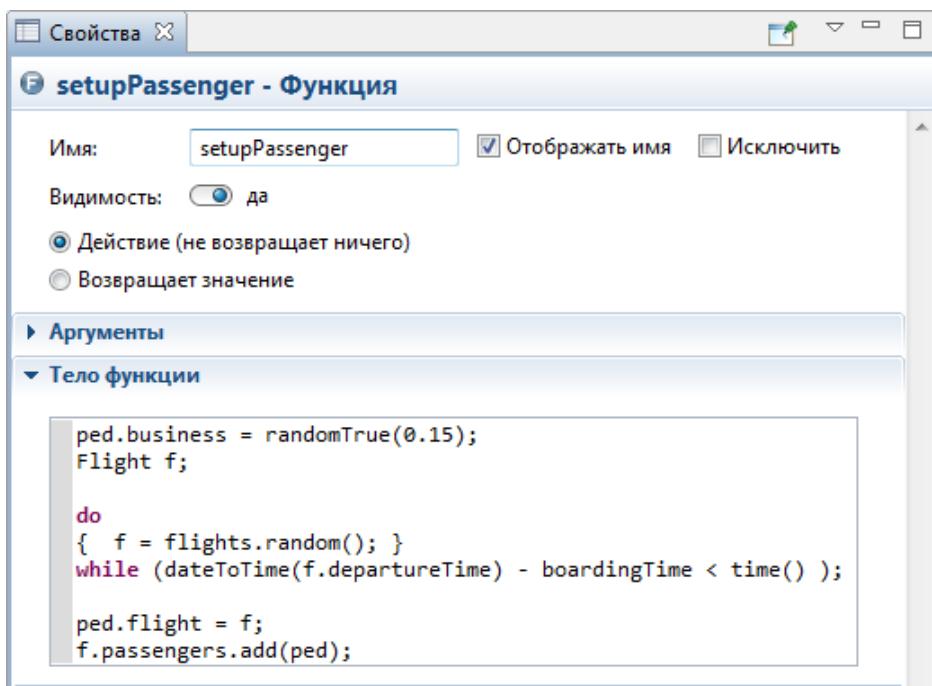


10. Вернитесь на диаграмму *Main* и добавьте параметр, который будет задавать длительность времени посадки в данном аэропорту. Назовите новый параметр *boardingTime*, выберите для него Тип: Время, Единица измерения: минуты и Значение по умолчанию: 40.



11. Выберите ранее созданную функцию *setupPassenger* и завершите процесс инициализации пешеходов. Теперь функция использует метод *random()* для случайного выбора рейса. Выбранный рейс сохраняется в параметре пешехода *flight*. Сам пешеход добавляется в коллекцию пассажиров данного рейса.

Измените Java код в поле Тело функции:



Функция `dateToTime()` преобразовывает заданную дату в модельное время с учетом начальной даты эксперимента и выбранных единиц модельного времени.

Функция `add()` добавляет элемент в коллекцию.

## Работа с содержимым коллекции

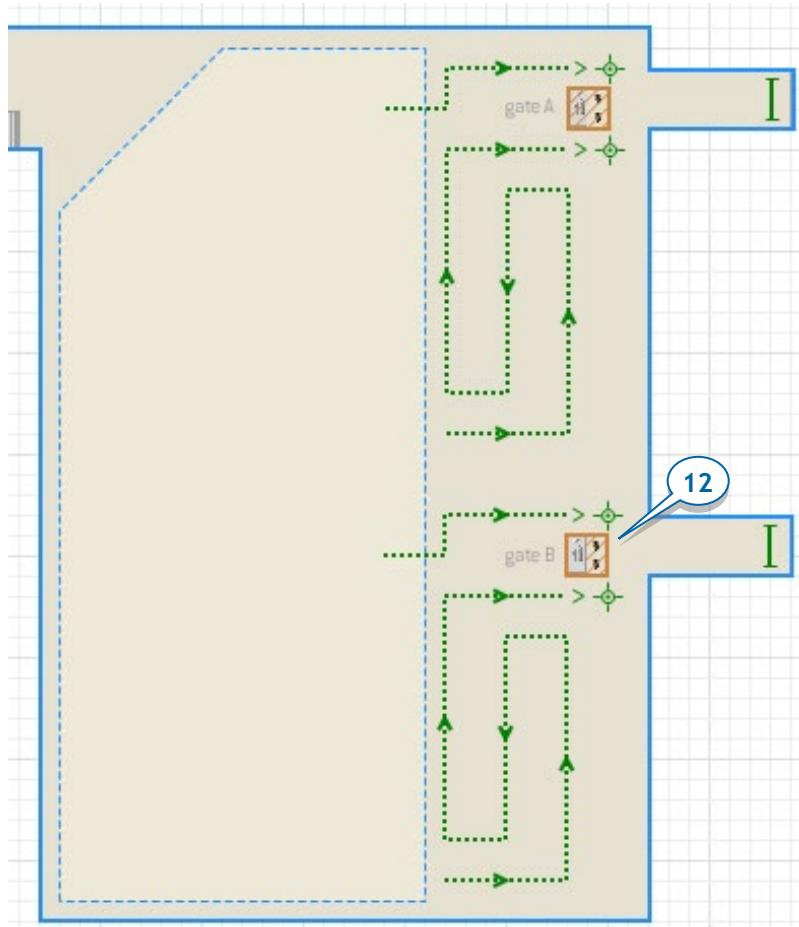
Используйте следующие функции для работы с содержимым коллекции:

- `int size()` – Возвращает количество элементов в коллекции.
- `boolean isEmpty()` – Возвращает `true`, если в коллекции нет элементов, иначе возвращает `false`.
- `add(element)` – Добавляет заданный элемент в коллекцию (на последнюю позицию).
- `clear()` – Удаляет все элементы из коллекции.
- `get(int index)` – Возвращает элемент коллекции с указанным порядковым номером (нумерация начинается с нуля).

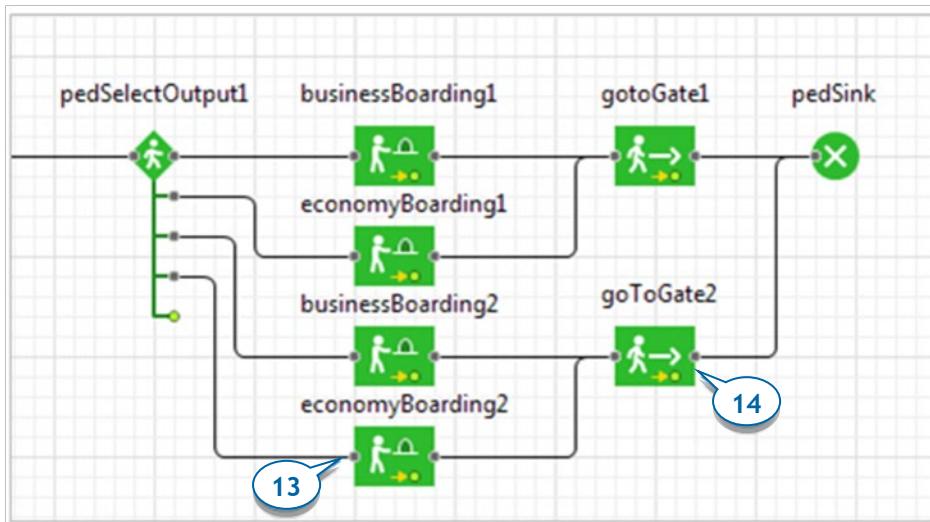
- `boolean remove(element)` – Удаляет заданный элемент из коллекции (если он в ней присутствовал). Возвращает `true`, если коллекция содержала этот элемент.
- `boolean contains(element)` – Возвращает `true`, если коллекция содержит заданный элемент.

12. Добавьте второй выход на посадку (гейт):

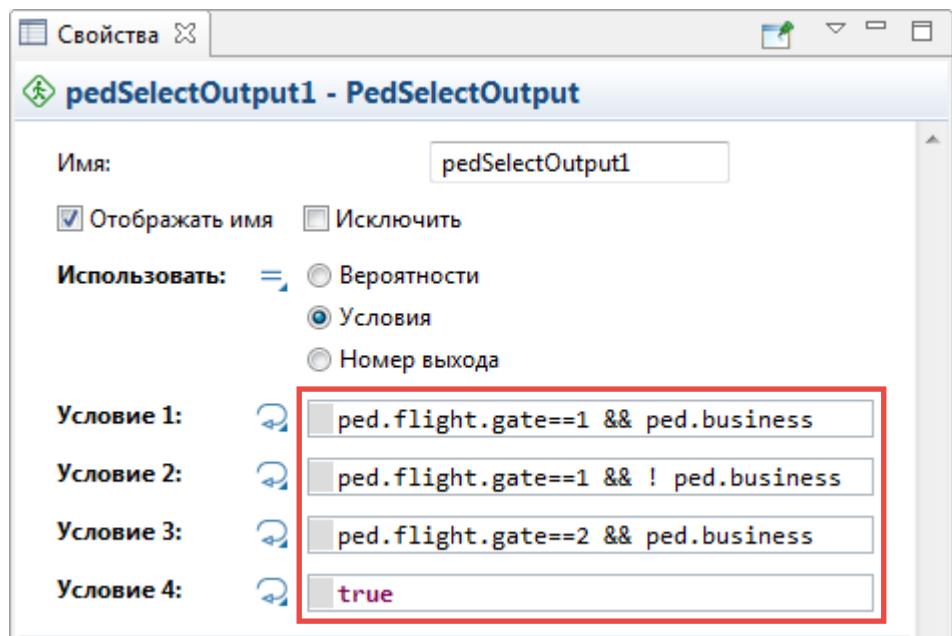
- Добавьте два элемента **Сервис с линиями**: `business2` и `economy2`.
- Добавьте **Прямоугольную стену**, 3D модели стола и служащих.
- Нарисуйте **Целевую линию** `gateLine2`.



- 13.** Добавьте еще два блока **PedService** : *businessBoarding2* и *economyBoarding2*. Соедините эти блоки с указанными ниже на рисунке выходными портами блока *pedSelectOutput1*. Теперь блок *pedSelectOutput1* будет перенаправлять пассажиров в один из четырех альтернативных выходных портов.



- 14.** Добавьте еще один блок **PedGoTo** .
- Этот блок будет моделировать то, как пассажиры направляются на посадку ко второму гейту. Выберите *gateLine2* в свойстве **Целевая линия** этого блока. Соедините этот порт с ранее созданными блоками *businessBoarding2* и *economyBoarding2*.
- 15.** В свойствах блока *businessBoarding2*, задайте **Сервисы: business2**. У блока *economyBoarding2*, укажите **Сервисы: economy2**. Для обоих этих блоков, задайте **Время задержки: uniform(2, 5) секунд**.
- 16.** Теперь, когда мы добавили всю логику, связанную с выбором рейса, давайте изменим условия блока *pedSelectOutput1*, чтобы пассажиры направлялись к нужному им гейту.



Теперь давайте перейдем к заданию логики вылетающих рейсов. Для этого мы воспользуемся динамическими событиями, с помощью которых мы зададим действия, связанные с объявлением посадки на рейсы и вылетом рейсов.

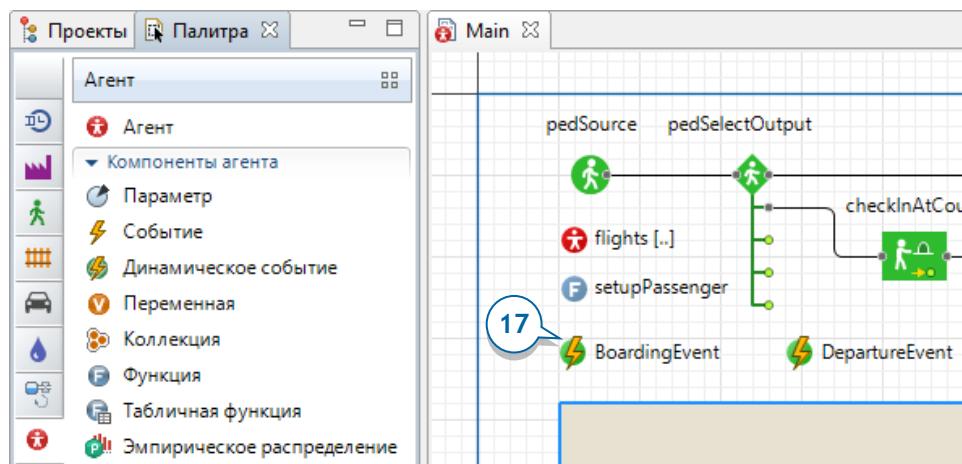
## Динамические события

*Динамические события* позволяют планировать выполнение заданных действий в будущем, в определенные моменты времени.

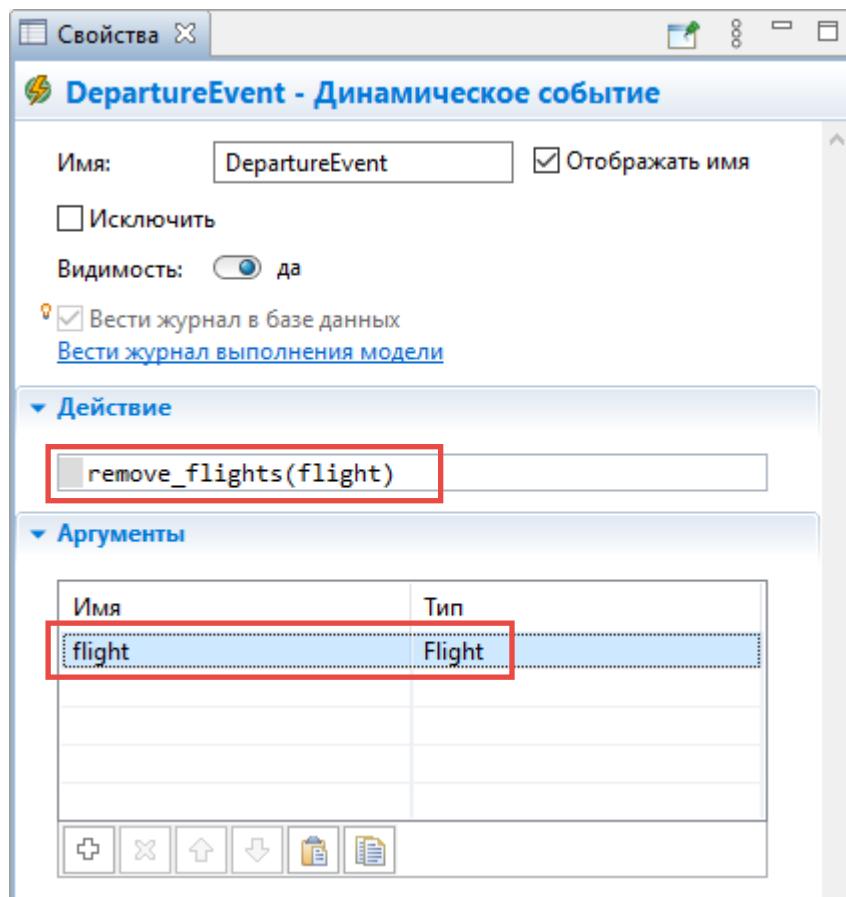
Используйте динамические события:

- Если в модели предполагается планирование большого количества однотипных событий (возможно, даже запланированных одновременно).
- Если действие события зависит от специфической информации (в этом случае можно передать событию необходимую информацию с помощью параметров этого события).
- Поскольку AnyLogic задает шаблон динамического события в виде Java класса, то имена динамических событий должны начинаться с заглавной буквы (в соответствии с соглашением о наименовании объектов в Java).

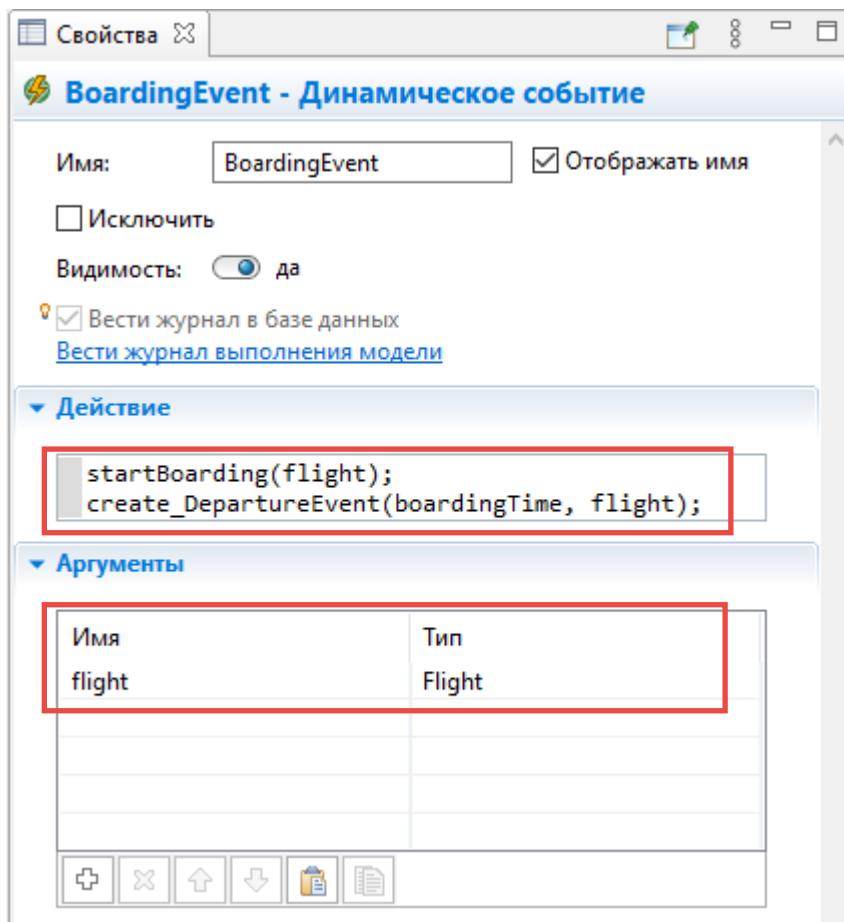
17. Добавьте на диаграмму *Main* два Динамических события  из палитры Агент .



18. Динамическое событие *DepartureEvent* планирует вылет самолета путем удаления рейса из популяции предстоящих рейсов. Настройте это динамическое событие так, как показано на рисунке ниже.



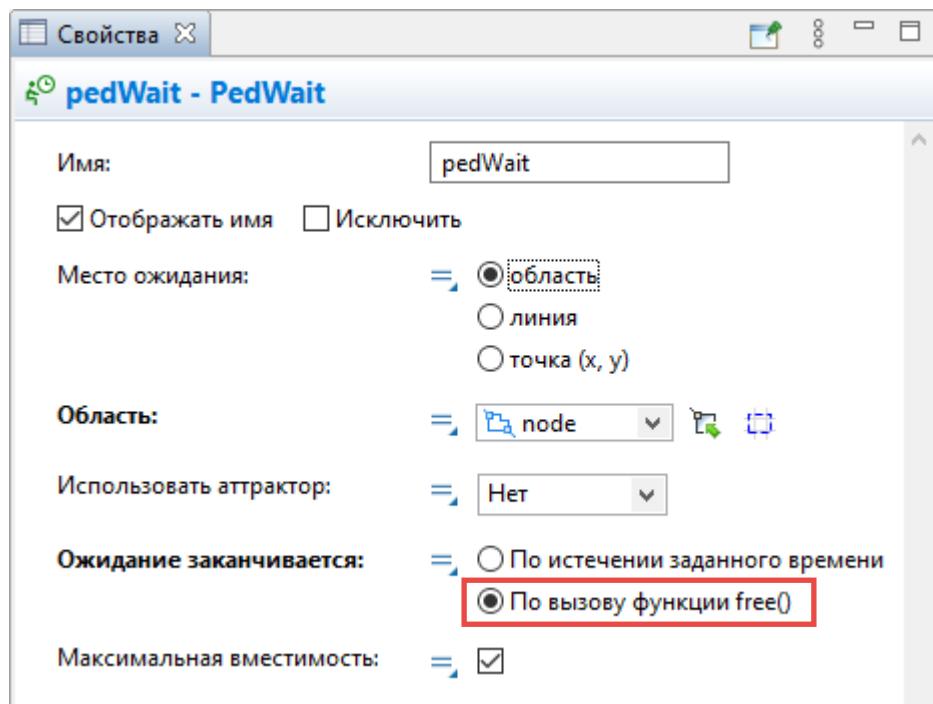
19. Другое динамическое событие, *BoardingEvent*, планирует начало посадки на рейс и затем создает экземпляр динамического события *DepartureEvent*, которое планирует вылет самолета через 40 минут после начала посадки.



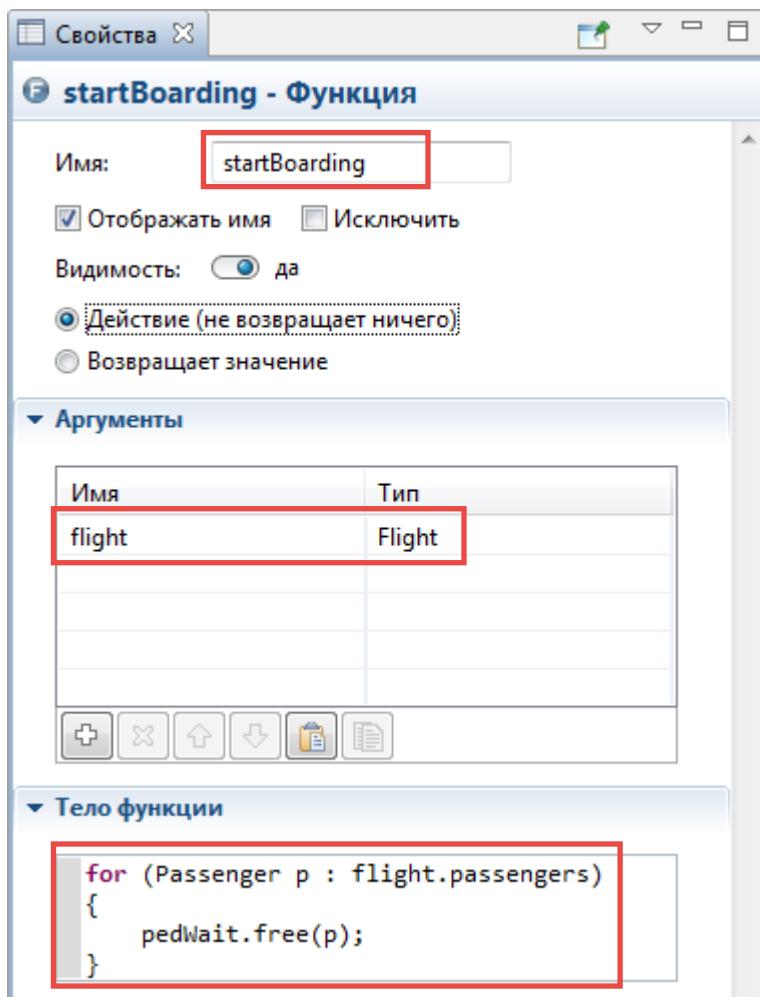
Чтобы запланировать новое динамическое событие (начать отсчет времени до его происхождения), нужно вызвать функцию `create_<ИмяДинамическогоСобытия>()`. В нашем случае имя функции будет `create_DepartureEvent()`. В качестве аргумента нужно передать длительность таймаута – количество единиц модельного времени, через которое это событие произойдет (отсчет начнется с момента вызова этой функции). После этого, обязательного, аргумента могут следовать значения заданных у динамического события аргументов. В нашем случае у события один аргумент – ссылка на рейс `flight`.

20. У блока `pedWait` измените значение параметра **Ожидание заканчивается со значениями По истечении заданного времени на По вызову функции `free()`**.

Теперь пассажиры будут находиться в области ожидания, пока не услышат объявление о начале посадки на рейс.

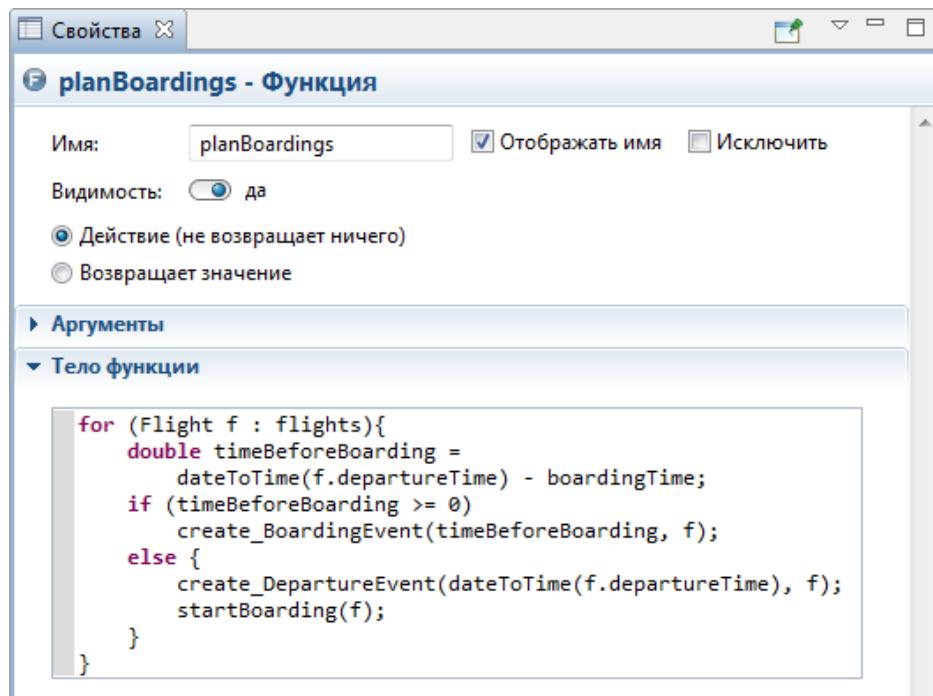


21. Добавьте Функцию *startBoarding*, которая будет моделировать начало посадки на рейс. Эта функция проходит в цикле по всем пассажирам указанного рейса и завершает для них процедуру ожидания путем вызова функции *free()* блока *pedWait*.



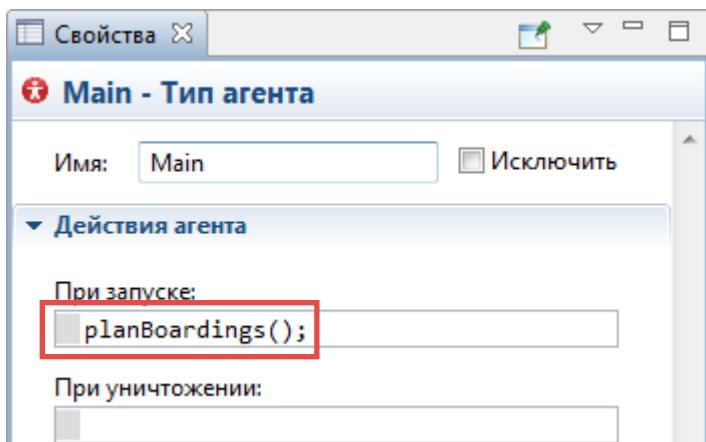
В нашем случае мы используем цикл *for* для того, чтобы пройти по всем элементам коллекции *passengers* (пассажирам заданного рейса *flight*). Все пассажиры данного рейса завершат ожидание в блоке *pedWait* и покинут область ожидания у гейта.

- 22.** Создайте функцию *planBoardings*, которая запланирует посадку на все рейсы. Эта функция итеративно проходит в цикле по всем вылетающим рейсам - агентам популяции *flights*.



Оператор принятия решения *if* проверяет заданное условие. Если посадка на данный рейс еще не начата, то происходит планирование динамического события *BoardingEvent* на определенный момент времени в будущем. В противном случае происходит планирование вылета, и вызывается функция *startBoarding*, разрешающая посадку (при этом ссылка на рейс передается с помощью аргумента функции).

- 23.** В секции свойств *Действия агента Main*, в поле *При запуске*, поместите вызов функции *planBoardings()*.

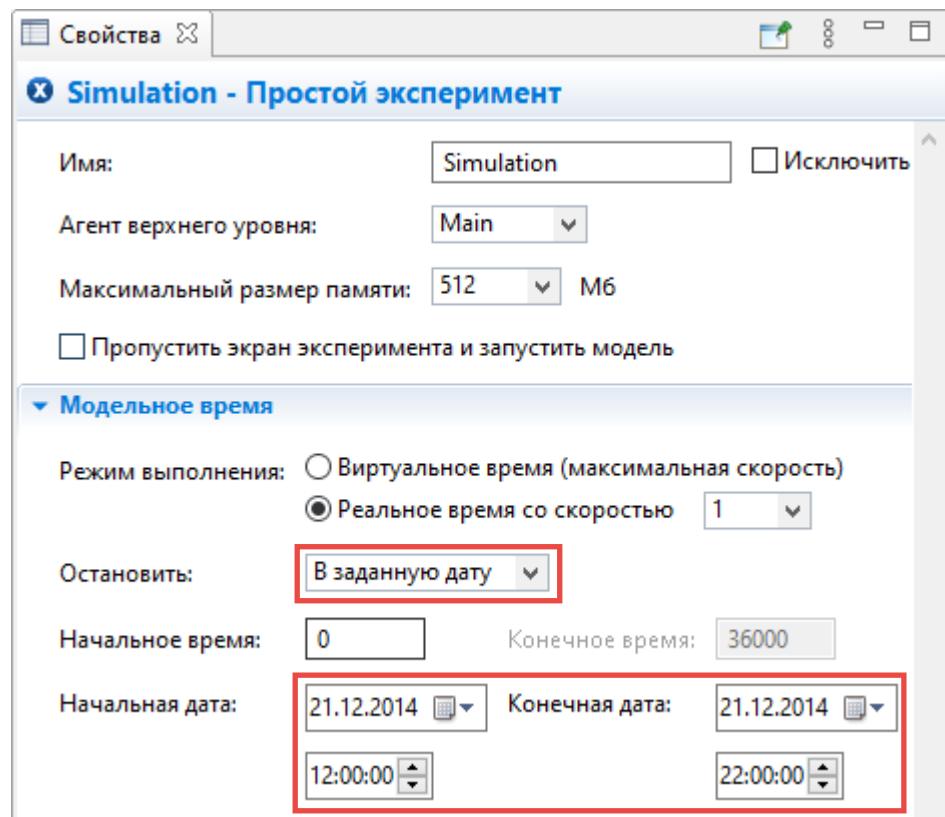


### Порядок инициализации модели

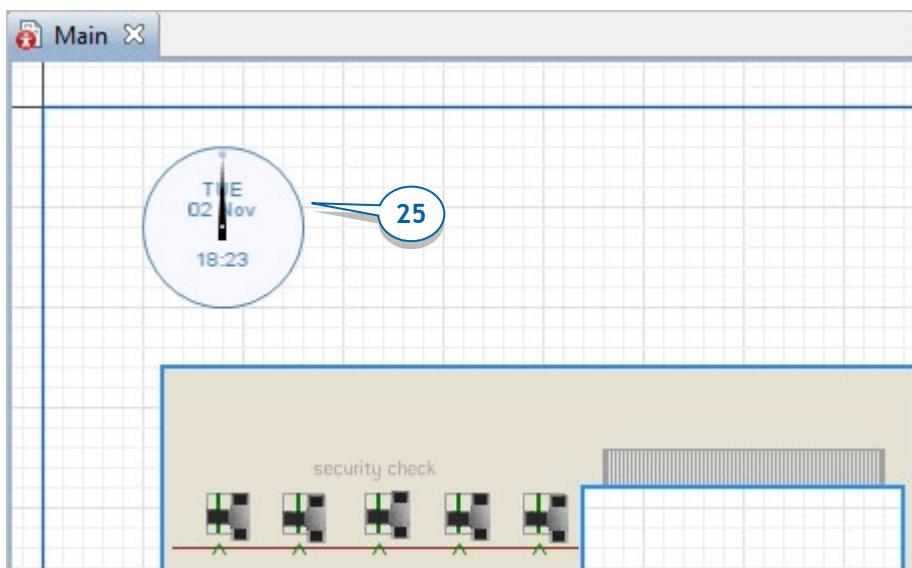
Код, заданный в поле **При запуске** агента верхнего уровня модели, исполняется на заключительном этапе инициализации модели после создания, соединения и инициализации всех объектов модели. В это время проходит дополнительная инициализация и запуск действий агентов, таких как события.

Теперь нам осталось лишь задать у эксперимента ту же начальную дату, что используется в таблице БД.

- 24.** В панели **Проекты**, выберите эксперимент *Simulation*. Откройте раздел свойств эксперимента **Модельное время**. Задайте **Начальную дату**: *21/12/2014, 12:00:00*. В списке **Остановить**, выберите опцию **В заданную дату**, а затем задайте **Конечную дату**: *21/12/2014, 22:00:00*.



25. Для того, чтобы иметь представление о модельном времени по ходу выполнения модели, добавьте на графическую диаграмму Часы из палитры Картинки.



26. Запустите модель. Вы увидите, как пассажиры ожидают объявления о начале посадки на рейс, после чего направляются к гейту для прохождения процедуры проверки посадочных талонов.



## Заключение

Вот мы и закончили наш стремительный практический курс по ознакомлению с AnyLogic. Как вы уже поняли, диапазон применения продукта неограничен, и я мог бы продолжить, описывая создание новых и новых моделей, например, бизнес-процессов в госпитале, оптимизации дорожного движения или цепочки поставок нефтегазовой корпорации. Но, думаю, необходимые навыки для начала самостоятельной работы с продуктом у вас уже есть, и теперь ответный ход за вами! Правильнее будет поставить точку и дать пару советов, как можно продолжить изучение AnyLogic, чтобы успешно выполнить поставленную перед вами задачу.

Следующим шагом может стать изучение учебных пособий, входящих в стандартную документацию AnyLogic.

Затем вы можете изучить набор стандартных примеров, поставляемых вместе с продуктом, и попытаться детально разобрать те из них, которые наиболее близки по постановке задачи вашему текущему проекту.

Если вам понадобится экспертная помощь, вы можете обратиться с вопросом к членам группы AnyLogic Users в социальной сети LinkedIn (на английском языке) или же написать письмо в Службу Поддержки AnyLogic с помощью кнопки панели инструментов **Обратиться за помощью** (она недоступна в версии AnyLogic PLE). Сотрудники службы поддержки с радостью ответят на любые ваши вопросы.

Удачи вам в ваших проектах!

## Список литературы

Borshchev, A. (2013). *The Big Book of Simulation Modeling. Multimethod modeling with AnyLogic 6*. AnyLogic North America.

Compartmental models in epidemiology. (б.д.).

[http://en.wikipedia.org/wiki/Compartmental\\_models\\_in\\_epidemiology](http://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology)

Conway's Game of Life. (б.д.). [http://en.wikipedia.org/wiki/Conway%27s\\_Game\\_of\\_Life](http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life)

Geer Mountain Software Corporation (2002) *Stat::Fit* (Version 2) [Software]. Geer Mountain Software Corporation. <http://www.geerms.com>

Oracle. (2011). *Java™ Platform, Standard Edition 6. API Specification*. [Online].

<http://docs.oracle.com/javase/6/docs/api/>

Random number generator. (б.д.).

[http://en.wikipedia.org/wiki/Random\\_number\\_generator](http://en.wikipedia.org/wiki/Random_number_generator)

Sterman, J. (2000). *Business dynamics: Systems thinking and modeling for a complex world*. New York: McGraw.

Sun Microsystems, Inc. (1999). *Code Conventions for the Java™ Programming Language*. [Online]. <http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html>

System Dynamics Society, Inc. (2014). System Dynamics Society [Online].

[www.systemdynamics.org](http://www.systemdynamics.org)

The AnyLogic Company. (2014). *AnyLogic Help*. [Online].

<http://www.anylogic.com/anylogic/help/>

The Game of Life. (б.д.). [http://en.wikipedia.org/wiki/The\\_Game\\_of\\_Life](http://en.wikipedia.org/wiki/The_Game_of_Life)

UML state machine. (б.д.). [http://en.wikipedia.org/wiki/UML\\_state\\_machine](http://en.wikipedia.org/wiki/UML_state_machine)