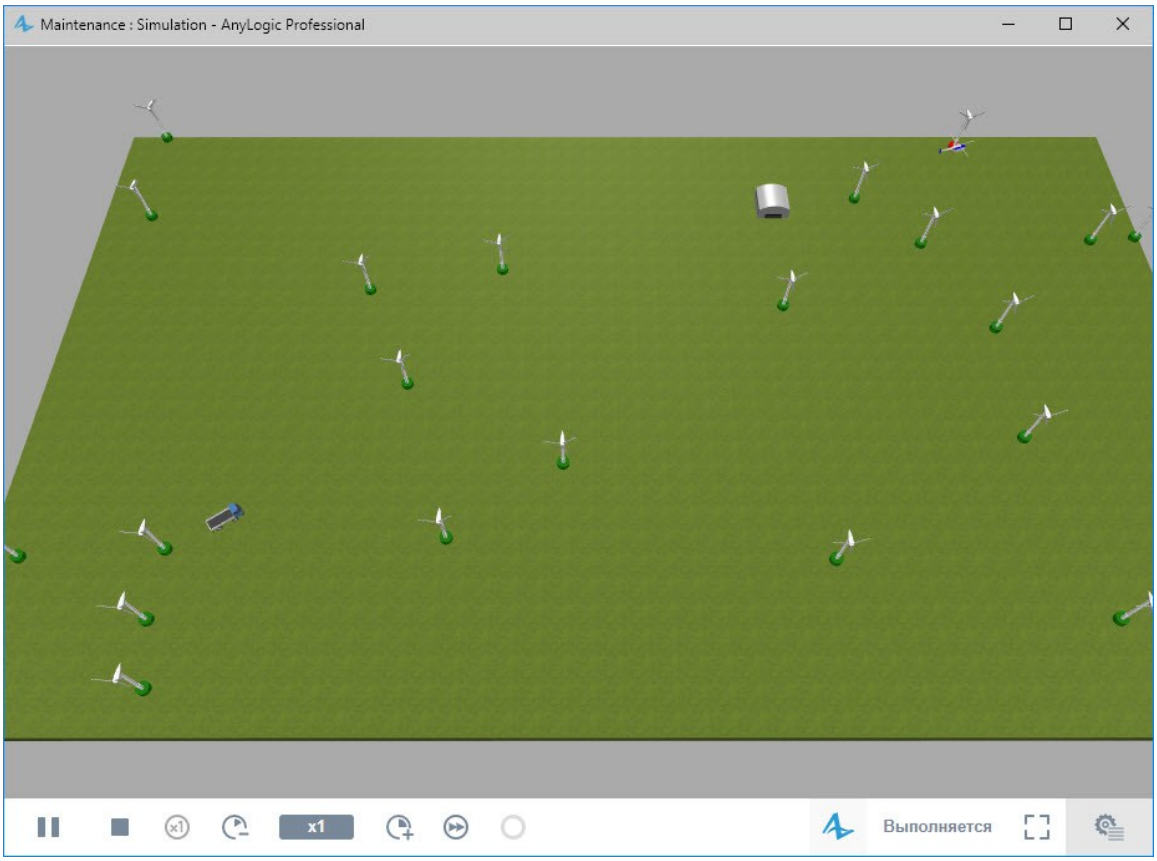
Трудоемкость – 2 ч 20 мин.

Агентная модель обслуживания ветряных турбин

Рассмотрим ветропарк, где имеется десять ветряных турбин, расположенных случайным образом в непрерывном пространстве, с одним центром обслуживания. Смоделируем, как сервисный центр производит обслуживание этих ветряных турбин.



В модели будем различать два вида сервисных работ:

Периодически проводимое техническое обслуживание

- ТО должно проводиться не реже одного раза в две недели.
- Сервисная бригада выезжает к турбине на грузовике.
- Время проведения работ равно 10 часам.

Срочное устранение поломок

- Среднее время между поломками – 50 дней.
- Сервисная бригада вылетает к турбине на вертолете.
- Время устранения поломки равномерно распределено от 10 до 20 часов.

Ветряные турбины обслуживаются одним сервисным центром. В Центре есть парк транспортных средств, состоящий из двух вертолетов и пяти грузовиков.

Промоделируем один год. При желании в модели можно учесть затраты по видам деятельности, добавить стоимость бригад, логику замены деталей деталями различной стоимости и т.д.

Фаза 1. Создание различных типов агентов

Под агентом в агентном моделировании понимается элемент модели, который может иметь поведение, память (историю), контакты и т.д. Агенты могут моделировать людей, компании, проекты, автомобили, города, животных, корабли, товары и т.д.

Можно создавать внутри агента переменные, диаграммы состояний, задавать события, потоковые диаграммы системной динамики, а также добавлять внутрь агента объекты библиотек AnyLogic.
В одной модели может быть любое количество типов агентов.

Создание агента обычно начинается с определения его интерфейса для связи с внешним миром.
В случае систем с большим количеством агентов с динамическими связями (например, в моделях социальных сетей) агенты могут взаимодействовать друг с другом путём вызова методов друг друга.

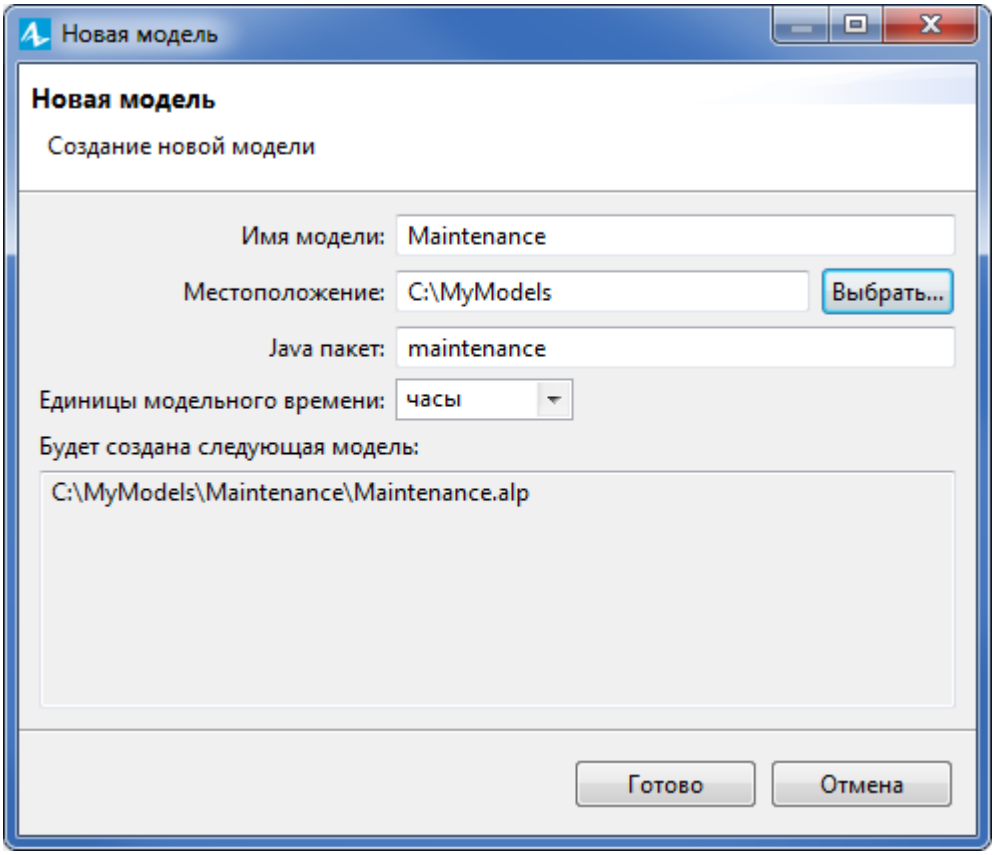
Начальное состояние и поведение агента могут быть реализованы различными способами. Состояние (накопленная история) агента может быть представлено с помощью переменных, либо состояния диаграммы состояний. Поведение может быть либо пассивным (агенты реагируют только на прибытие сообщений или на вызов методов и не имеют собственных событий, запланированных на будущее) или активным, когда внутренняя динамика агента (события, запланированные через заданные таймауты или процессы системной динамики) является причиной действий, совершаемых агентом. В последнем случае внутри агентов, скорее всего, должны быть заданы события и диаграммы состояний.

В нашей модели будем использовать параметры, переменные, коллекции, функции, события, списки вариантов, а также диаграммы состояний.

Создадим новую модель.

Для этого щёлкните по кнопке **Создать** на панели управления. Откроется диалоговое окно **Новая модель**.

Задайте имя модели. Введите Maintenance в поле **Имя модели**.

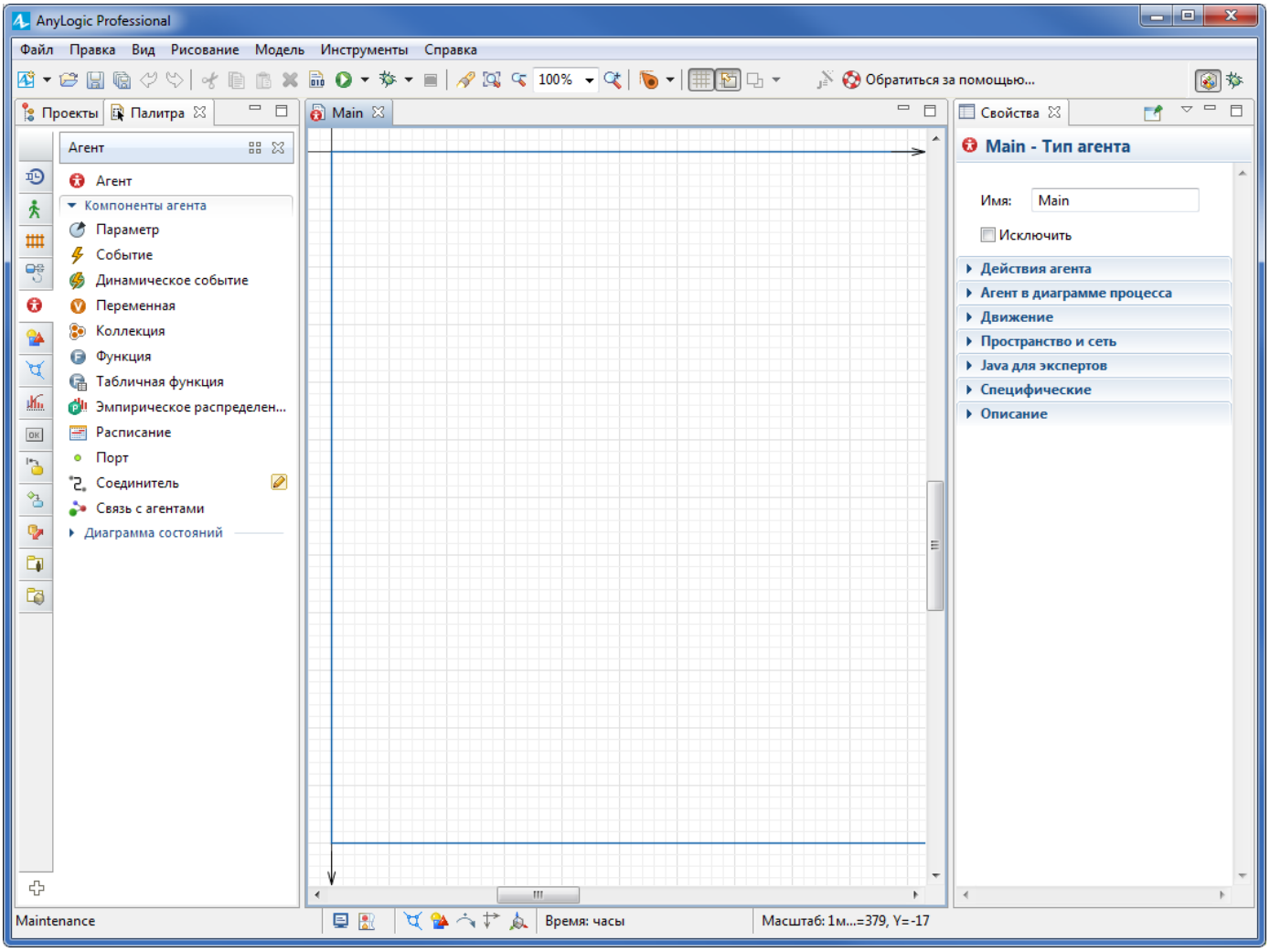


Укажите местоположение, где вы хотите хранить файлы модели. Используйте кнопку **Выбрать...**, чтобы выбрать нужную папку, или введите путь к папке в поле **Местоположение**.

В качестве **Единиц модельного времени** выберите **часы**. Щелкните **Готово**.

Будет создана новая модель. AnyLogic автоматически создаст тип агента Main и простой эксперимент Simulation.

В центре рабочего пространства вы увидите графический редактор. Он отображает диаграмму типа агента Main. Рамка синего цвета задаёт область диаграммы, которая будет отображена в окне модели при ее запуске (а также размеры этого окна).



Слева от графического редактора вы можете видеть панель **Проекты**, объединённую с панелью **Палитра**. Панель **Проекты** обеспечивает доступ к моделям AnyLogic, открытым в данный момент в рабочем пространстве. Дерево элементов модели позволяет легко ориентироваться в ее структуре. Панель **Палитра** содержит все графические элементы, которые вы можете добавлять на диаграмму типа агента, просто перетаскивая их в графический редактор. Элементы сгруппированы в отдельные палитры.

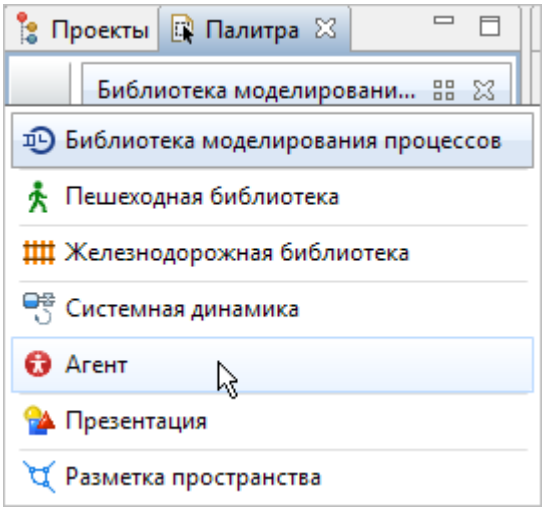
Справа вы можете найти панель **Свойства**. Панель **Свойства** отображает и позволяет изменять свойства выбранного в данный момент элемента (группы элементов) модели. Когда вы выделяете какой-либо элемент, например, в панели **Проекты** или в графическом редакторе, панель **Свойства** отображает свойства выделенного элемента.

Создание агентов

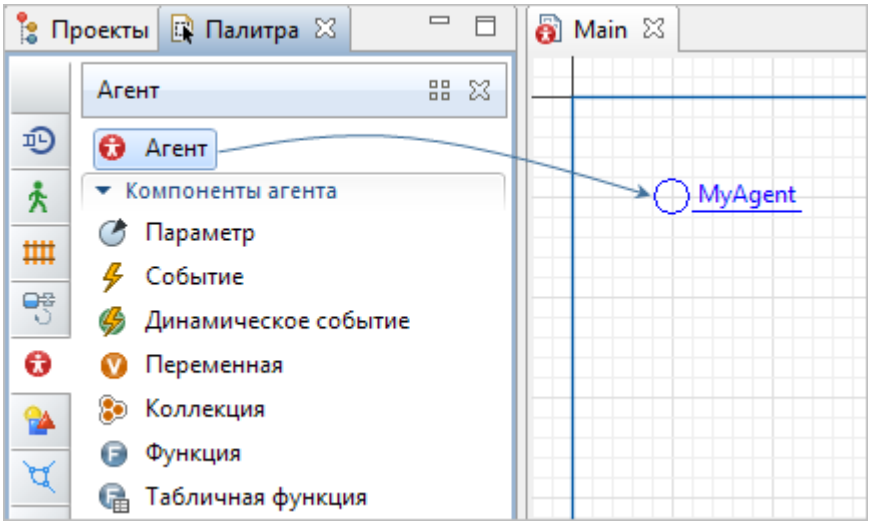
Вначале создадим одного агента - сервисный центр, затем пустую популяцию для обслуживающего транспорта, три популяции для моделирования турбин, грузовиков и вертолетов, а также тип агента,

представляющего запросы на обслуживание. После этого мы сможем задать процессы внутри типов агентов - на их диаграммах.

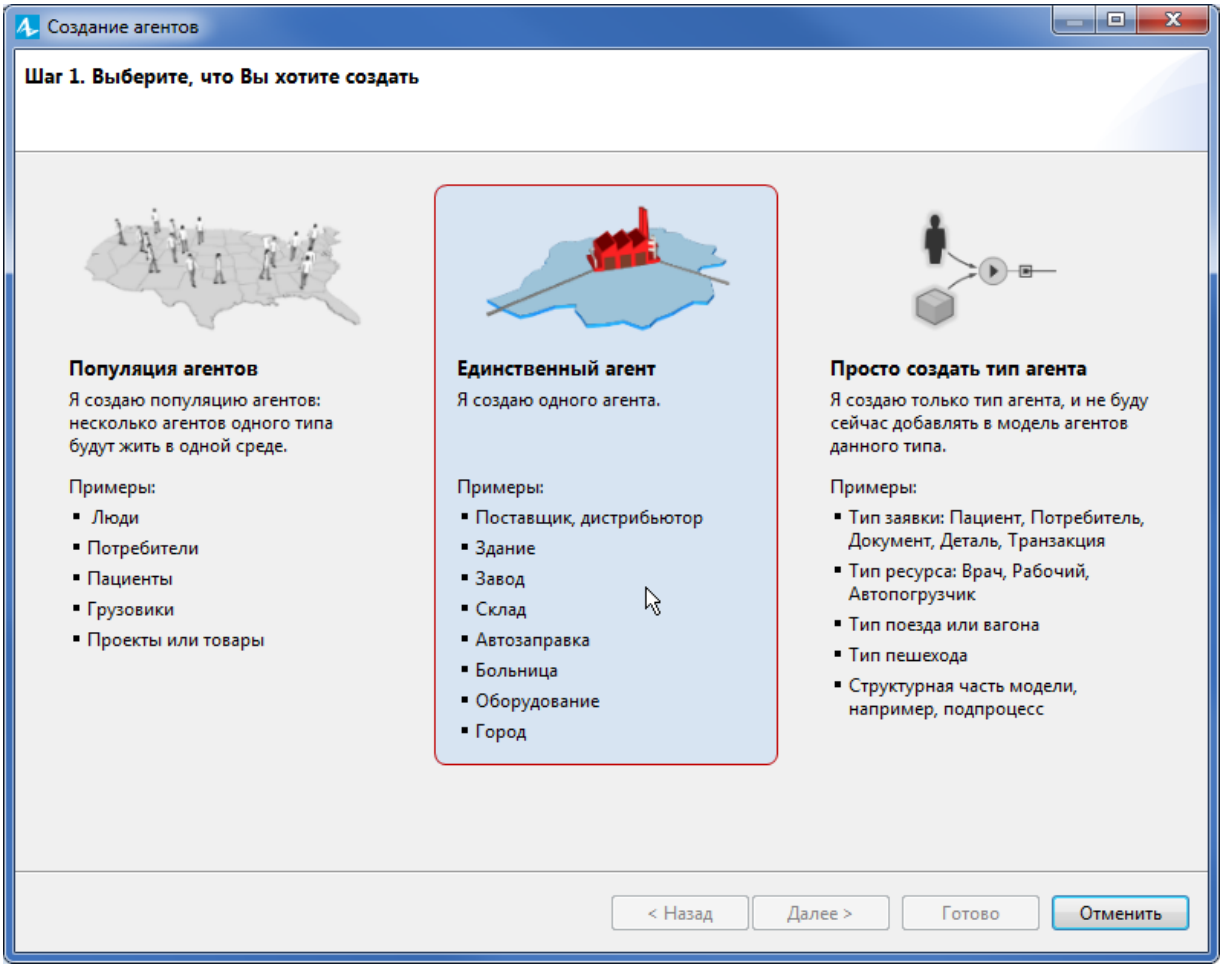
Добавим сервисный центр. Для этого в панели **Палитра** наведите мышь на вертикальную полосу навигации (она располагается по левому краю панели), и выберите палитру **Агент**.



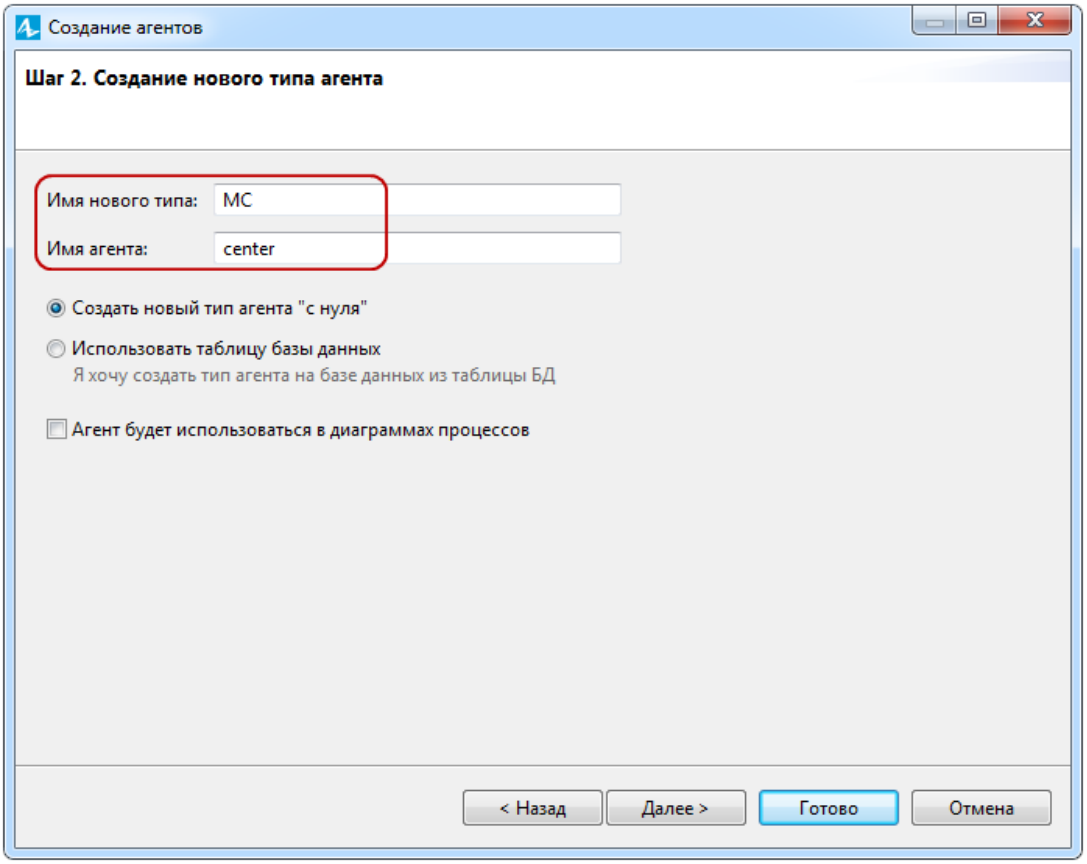
Перетащите элемент **Агент** из палитры на диаграмму типа агента Main. Окно мастера **Создание агентов** откроется автоматически.



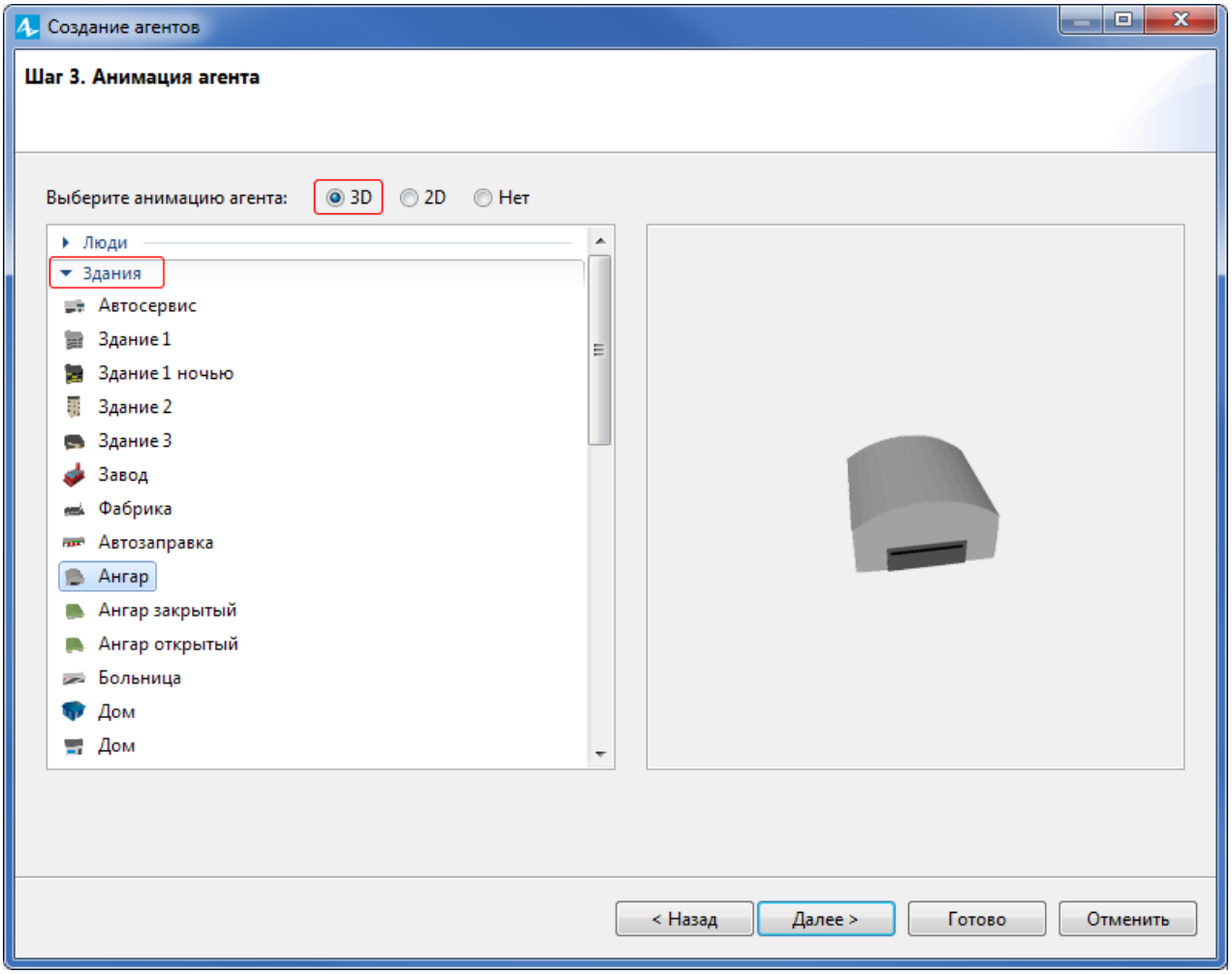
Выберите опцию **Единственный агент** на первом шаге мастера. Нам нужно создать только один сервисный центр, который будет отправлять транспорт к турбинам по запросам на плановое обслуживание или при авариях.



Мы не будем использовать данные из базы данных, поэтому оставьте выбранным опцию **Создать новый тип агента "с нуля"**. Задайте **Имя нового типа**: MC и имя самого агента **center** в поле ниже. Щёлкните **Далее**, чтобы продолжить.

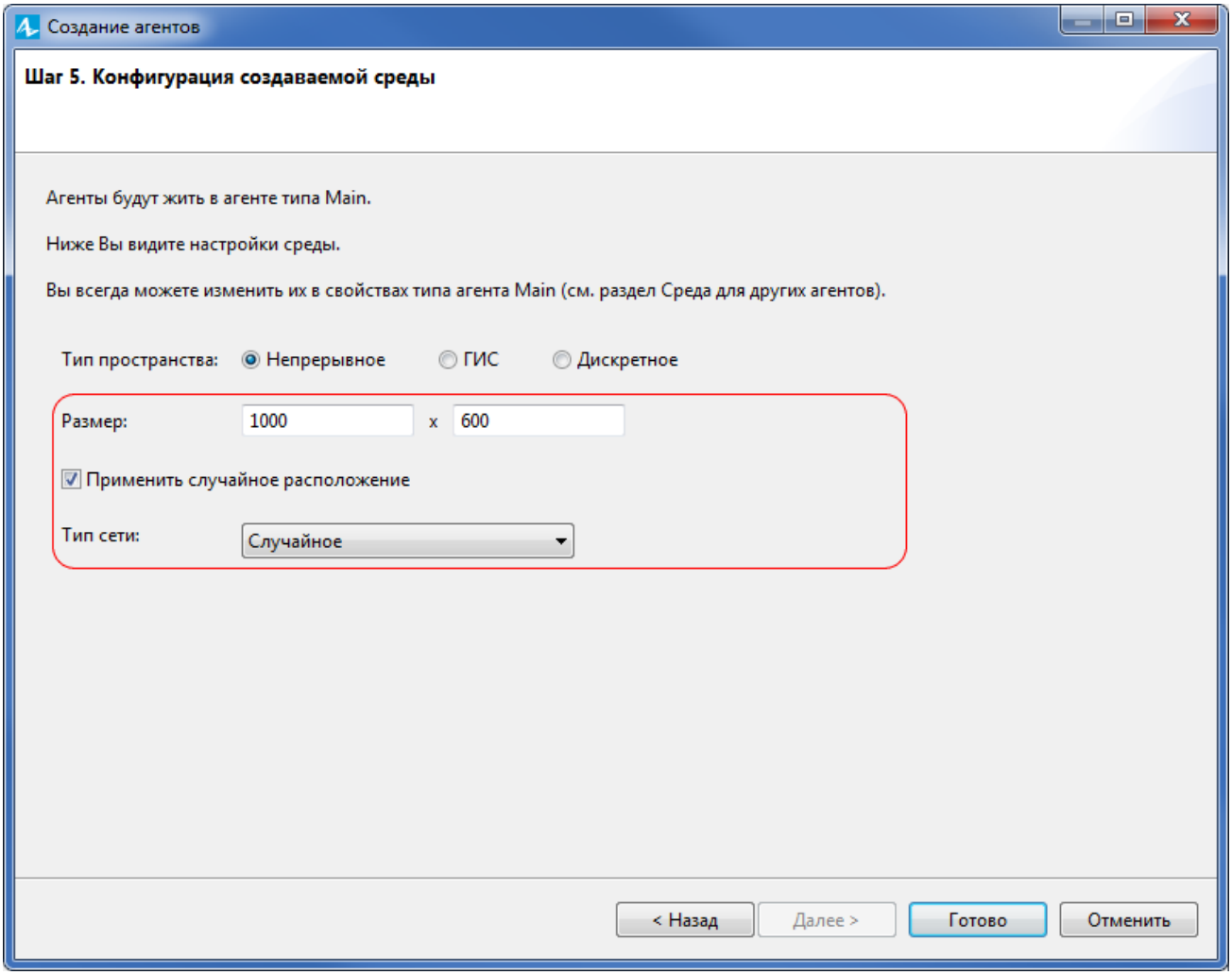


Выберите тип анимации **3D**, затем фигуру анимации **Ангар** из секции **Здания** и щёлкните **Далее**.



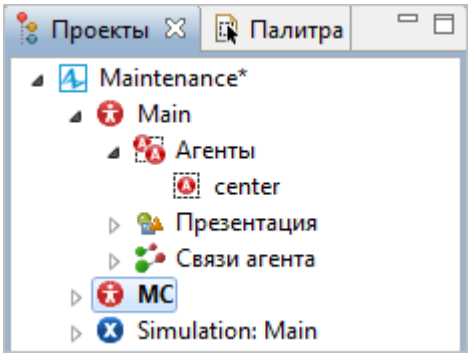
Пропустите четвёртый шаг по добавлению параметров, просто щёлкните **Далее**.

Мы хотим, чтобы все наши агенты "жили" в непрерывном пространстве, размерами 1000x600, в сети со случайным расположением агентов. Популяции агентов, которые мы позже так же добавим на диаграмму Main, будут жить в той же среде.



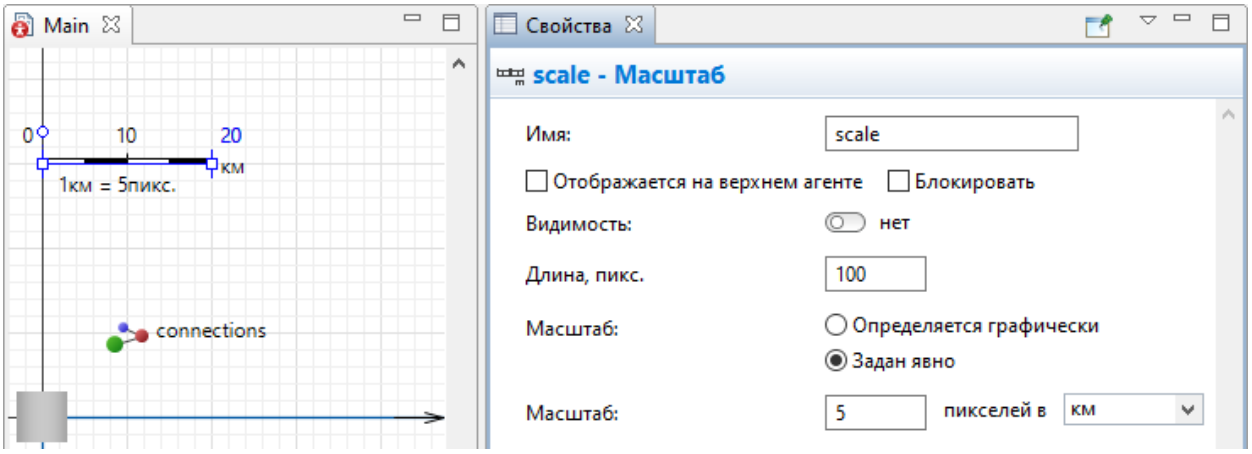
Щелкните **Готово**.

После создания нового типа агента MC, вы можете найти его в дереве модели на том же уровне, что и тип агента Main, а сам агент **center** находится в ветке **Main > Агенты**.



Фигура анимации агента отображается как на диаграмме его типа, так и на Main. Нам необходимо изменить масштаб всей модели, а также масштаб отображения этой фигуры на анимации.

1. Чтобы изменить масштаб модели, сделайте двойной щелчок мышью по типу агента Main в дереве элементов модели, чтобы открыть его диаграмму. Затем передвиньте диаграмму вниз, чтобы видеть элементы, находящиеся над осью X.
2. Здесь вы найдете объект **Масштаб**. Выделите его, чтобы открыть его свойства.
3. Установите **Масштаб** в режиме **Задан явно**, не изменяя длину линейки шкалы. В свойствах элемента, задайте **Масштаб**: 5 пикселей на 1 км.



Теперь давайте изменим масштаб фигуры анимации сервисного центра (ангара).

Сделайте двойной щелчок мышью по типу агента **MC** в дереве элементов модели, чтобы открыть его диаграмму. Найдите и выделите линейку масштаба этого типа агента.

Очевидно, что при выбранном масштабе фигура ангара будет отображаться на анимации модели такой крошечной, что мы не сможем ее разглядеть. Поэтому мы зададим для этого типа агента более крупный масштаб, чтобы мы могли легко увидеть его фигуру на сцене анимации нашей модели. В свойствах фигуры анимации ангара снимите флажок **Автоматически изменять размер для соответствия масштабу агента**.

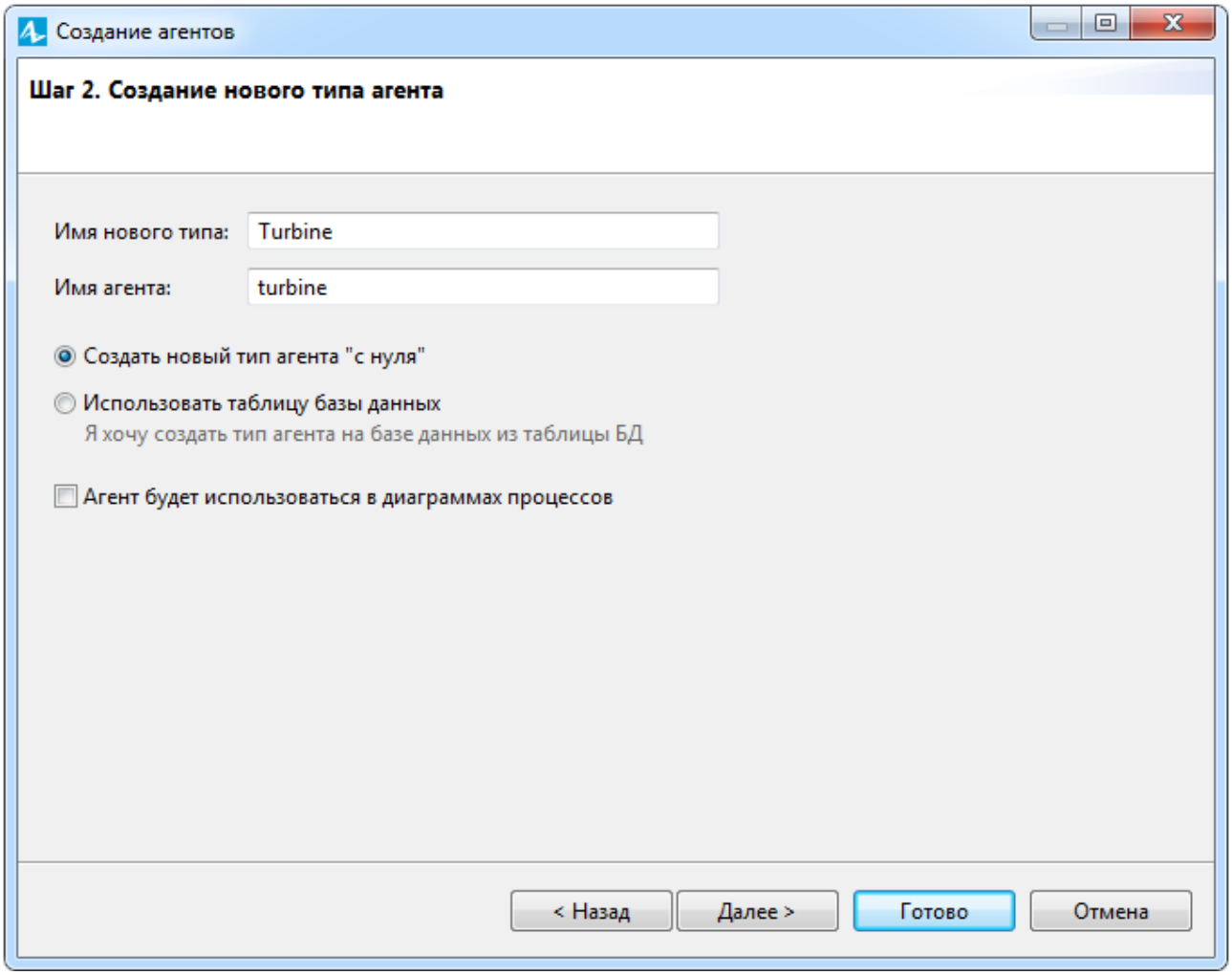
Переключите опцию **Масштаб** в режим **Задан явно**. Ниже, задайте масштаб: **10 пикселей в км**.

Выберите фигуру анимации сервисного центра и в свойствах задайте **Доп. масштабирование: 75%**.

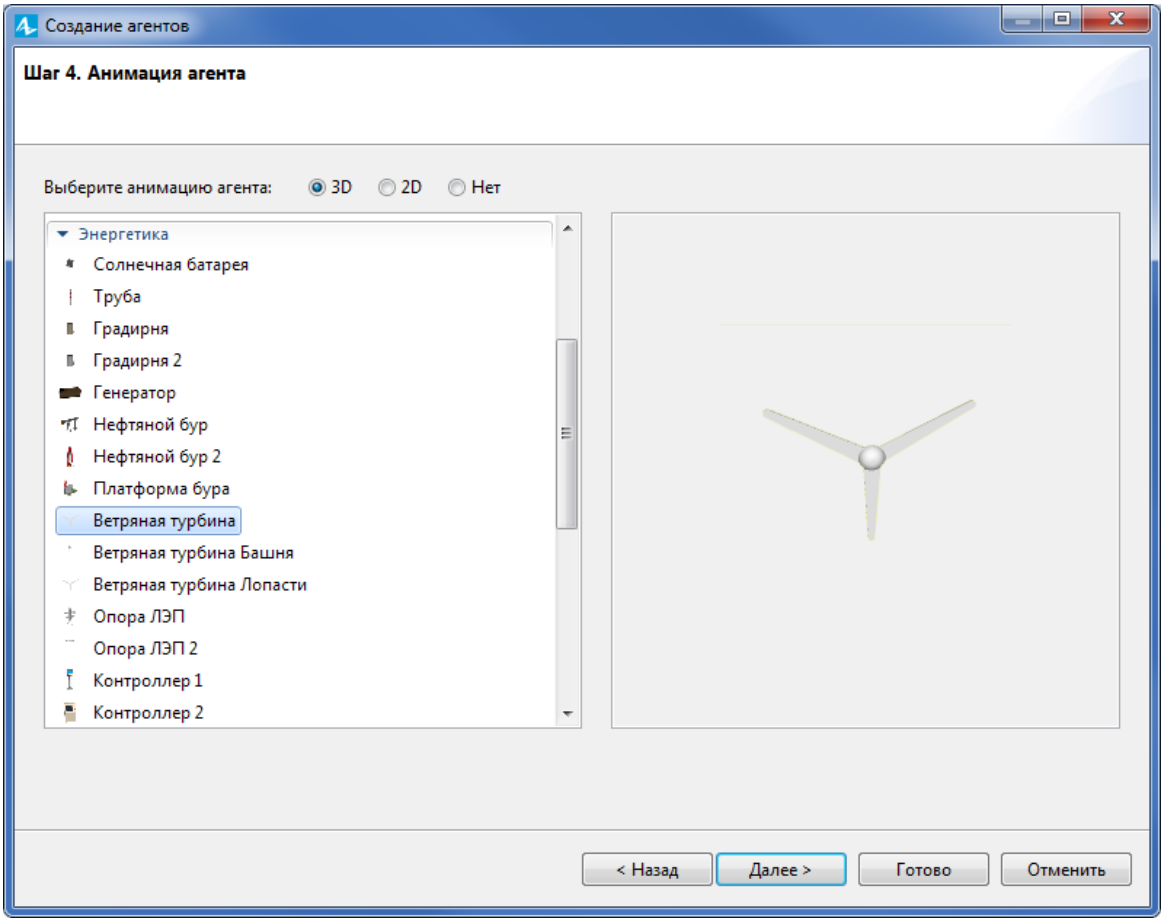
Добавим ветряные турбины.

Для этого перетащите элемент **Агент** из палитры **Агент** на диаграмму типа агента Main.

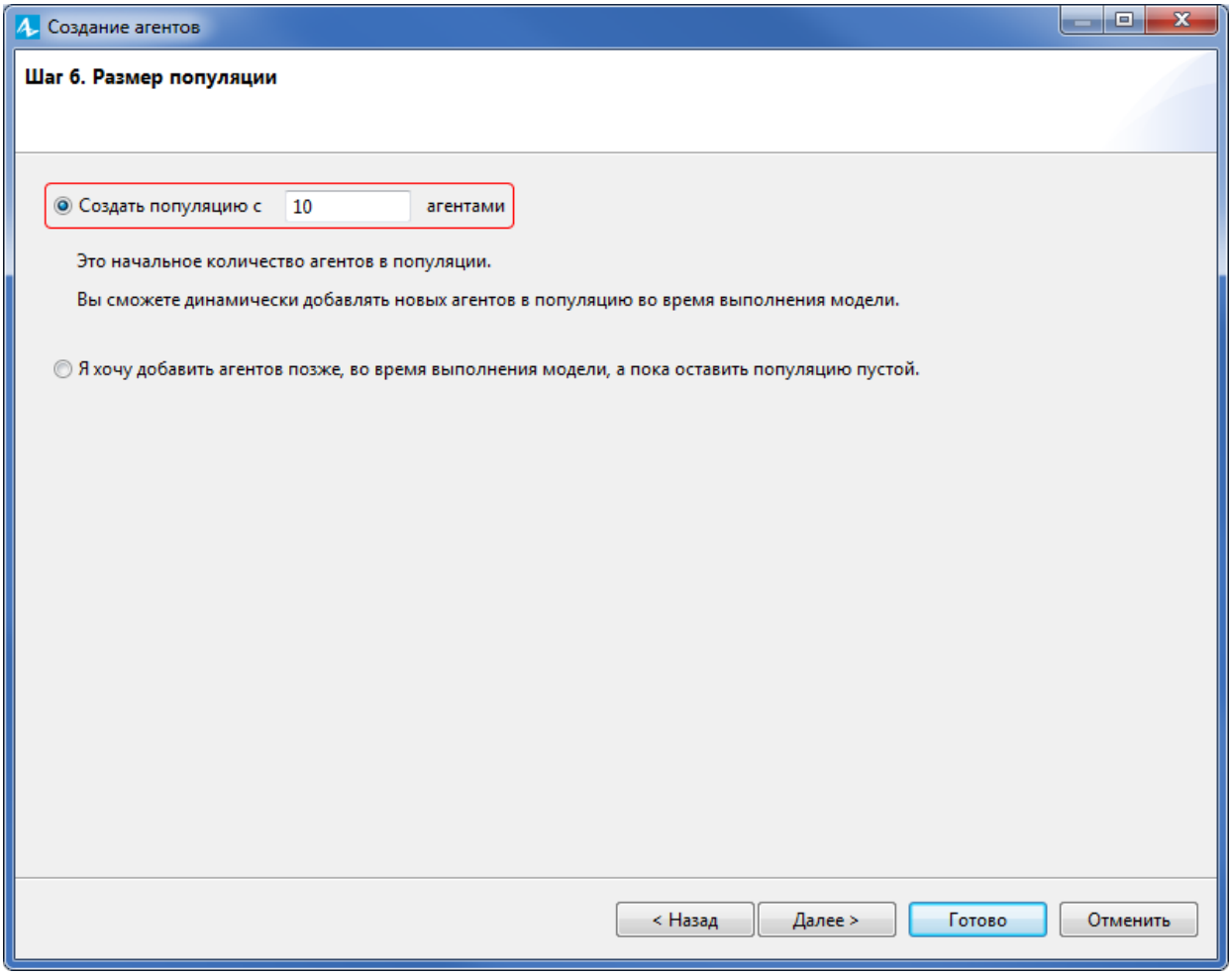
В этот раз выберите опцию **Популяция агентов**. Щелкните **Далее** на шаге 2 (по умолчанию выбрана нужная нам опция **Я хочу создать новый тип агента**). На следующем шаге мастера создания агентов введите имя нового типа: Turbine и тогда автоматически появится имя популяции: turbines. Щелкните **Далее**.



Выберите **3D** фигуру анимации **Ветряная турбина** из секции списка объектов **Энергетика**.



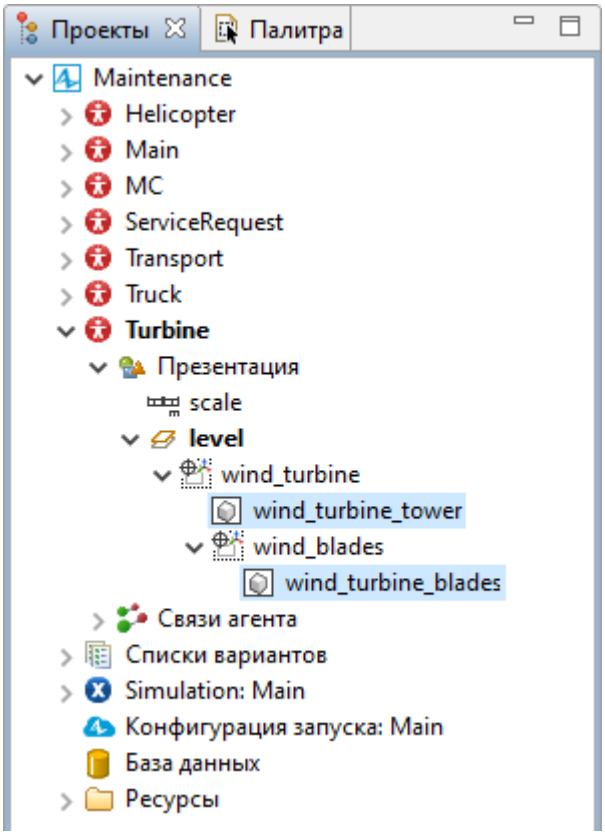
Нам не нужно сейчас создавать какие-либо параметры, так что пропустите шаг 5. На шаге 6, задайте количество агентов, которые мы хотим видеть в этой популяции: 10 агентов.



Вы можете щелкнуть **Готово** уже на шаге 6. Настройки шага 7, **Конфигурация создаваемой среды**, будут автоматически заполнены, так как мы добавляем популяцию агентов в тип агента Main, где мы уже задавали среду при создании агента сервисного центра.

Откройте диаграмму типа агента Turbine и измените масштаб. Поскольку мы хотим, чтобы элементы нашей модели изображались на анимации чуть крупнее, и их можно было разглядеть, зададим чуть увеличенный масштаб. Пусть длина линейки соответствует 10 километрам, и масштаб будет равен **1 км = 10 пикс**.

После того как вы увеличили масштаб, фигура анимации перестала отображаться на графическом редакторе. Это произошло потому, что в 3D объектах, входящих в группу, которой является эта фигура, по умолчанию выбрана опция **Автоматически изменять размер для соответствия масштабу агента**. Чтобы получить доступ к свойствам этих объектов, перейдите в панель **Проекты** и раскройте папку **Презентация** агента **Turbine**.



В группе **wind_turbine** найдите два 3D объекта: **wind_turbine_tower** и **wind_turbine_blades**.
В свойствах обоих объектов снимите флажок **Автоматически изменять размер для соответствия масштабу агента** и выберите в поле **Доп. масштабирование: 50%**.

Поскольку мы отключили автоматическое масштабирование, нам придется вручную настроить расположение элементов, составляющих группу **wind_turbine** относительно друг друга. Для этого выберите в дереве модели группу **wind_blades** и в поле свойств **Z** поместите следующее выражение:
*22.25 * Scale.DEFAULT_SCALE.pixelsPerUnit(METER)*

При вводе Java-кода пользуйтесь Мастером подстановки кода, доступным по нажатию в поле ввода клавиш Ctrl + пробел (macOS: Alt + пробел). С его помощью вы узнаете список доступных функций и элементов модели и сможете просто выбрать нужное вам имя из списка, тем самым избежав возможных ошибок при написании имени.

Добавим транспорт.

Для этого перетащите элемент **Агент** из палитры **Агент** на диаграмму типа агента Main.

Снова выберите опцию **Популяция агентов**. Щелкните **Далее** на шаге 2 (**Я хочу создать новый тип агента**). На следующей странице мастера (шаг 3) введите в поле **Имя нового типа**: Transport, и введите в поле **Имя популяции:** transport. Щелкните **Далее**.

Это пустая популяция, которой не нужна анимация. Мы будем использовать этот тип агента, чтобы задать логику модели. Выберите **Нет** для типа анимации, пропустите шаг создания параметров, выберите опцию **Я хочу добавить агентов позже, во время выполнения модели, а пока оставить популяцию пустой** и щелкните **Готово**.

Теперь добавьте в модель грузовики и вертолеты.

Перетащите элемент **Агент** из палитры **Агент** на диаграмму типа агента Main.

1. Снова выберите опцию **Популяция агентов**. Щелкните **Далее** на шаге 2 (**Я хочу создать новый тип агента**). На следующей странице мастера (шаг 3) введите в поле **Имя нового типа**: Truck, пусть имя популяции автоматически заполнится как trucks. Щелкните **Далее**.
2. Выберите фигуру **3D** анимации **Грузовик** из секции **Автодорожный транспорт** на следующем шаге. Нам не нужно добавлять сейчас параметры в мастере. Всего в популяции будет 5 агентов, а настройки среды уже будут заполнены.
3. После того, как создадите тип агента Truck, выделите Truck в дереве элементов модели и перейдите в панель **Свойства**. Сначала откройте секцию свойств **Размеры и движение**. Мы считаем, что грузовики в среднем движутся со скоростью **70 км в час**.
4. Откройте секцию свойств **Специфические** и укажите, что тип агента Truck наследуется от типа агента Transport. Для этого выберите в параметре **Расширяет тип агента:** Transport.
5. Зададим для грузовика чуть увеличенный масштаб. Для этого в свойствах элемента масштаб на диаграмме Truck, снимите флажок **Унаследовано от родительского класса**. Пусть длина линейки соответствует 10 километрам, и масштаб будет равен **1 км = 10 пикс**.
6. Теперь тип агента Truck полностью задан. Нам необходимо добавить еще одну популяцию для вертолётов. Вернитесь на Main и снова перетащите элемент **Агент** из палитры **Агент** и выберите **Популяция агентов** на первом шаге.
7. Введите имя нового типа Helicopter, оставьте имя популяции helicopters. Вы можете найти фигуру **3D** анимации **Вертолет** в секции **Военного назначения**. Мы хотим использовать 2 вертолета для обслуживания турбин. Эта популяция живёт в той же среде, что и все остальные.
8. Helicopter также **Расширяет тип агента:** Transport. Мы предполагаем, что вертолёты движутся со скоростью, равной **200 км в час**.
9. Зададим для вертолёта масштаб **1 км = 5 пикселей** (так же, как и чуть ранее, запретив наследование масштаба от родительского класса).

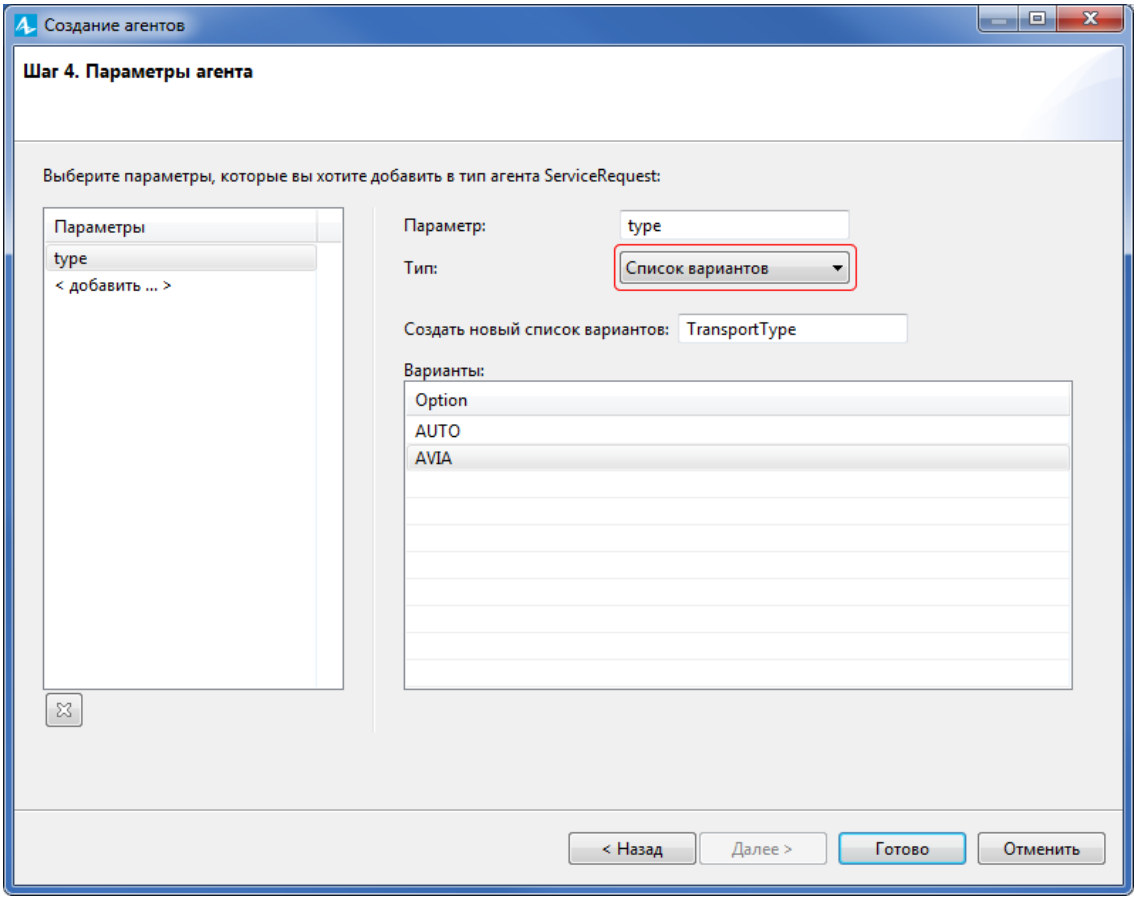
Добавим объекты - запросы на обслуживание.

1. Перетащите элемент **Агент** из палитры **Агент** на диаграмму типа агента Main.
2. Выберите опцию **Просто создать тип агента**. Нам нужен тип агента, чтобы задать логику модели специальными параметрами, которые мы создадим на его диаграмме.

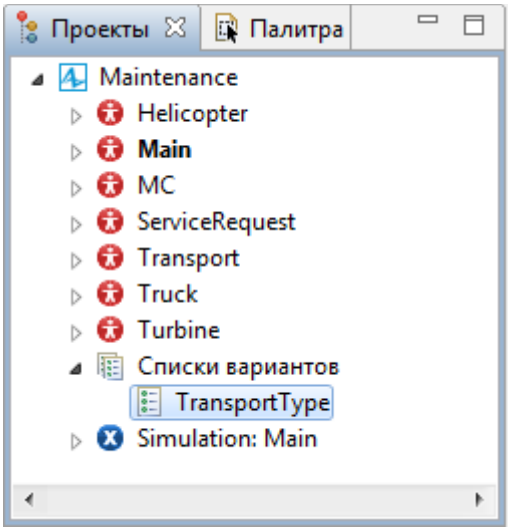
Введите имя типа ServiceRequest в соответствующем поле и щелкните **Далее**. Выберите **Нет** для типа анимации и перейдите на следующий шаг.

На шаге 4 мы создадим в мастере параметр, а также список вариантов. Список вариантов - это элемент, который позволяет задавать параметры агента, которые имеют ограниченный выбор вариантов. В нашем случае, нам необходимо различать грузовики и вертолёты в общем транспортном парке. Назовите параметр type и выберите его **Тип**: **Список вариантов**. Так как в этой модели мы еще не создавали списков вариантов, по умолчанию мастер предложит вам **Создать новый список вариантов:** введите имя TransportType

Добавьте варианты в таблицу, по одному в каждой строке: AUTO, AVIA

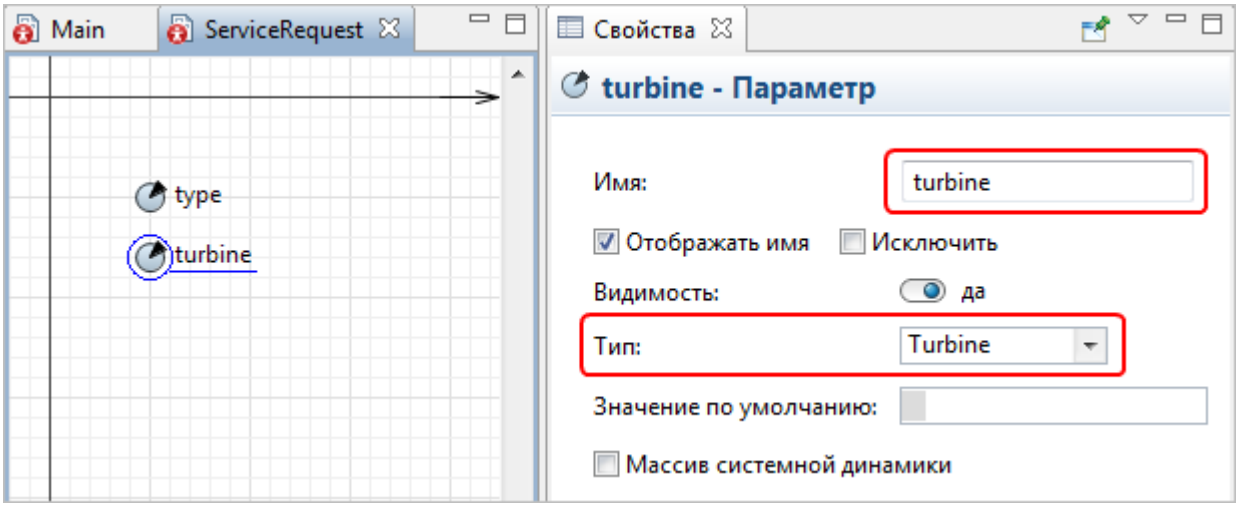


Щелкните **Готово**. Список вариантов не имеет отдельного значка, который можно выбрать щелчком в графическом редакторе. Вы можете найти секцию **Список вариантов** в дереве модели:

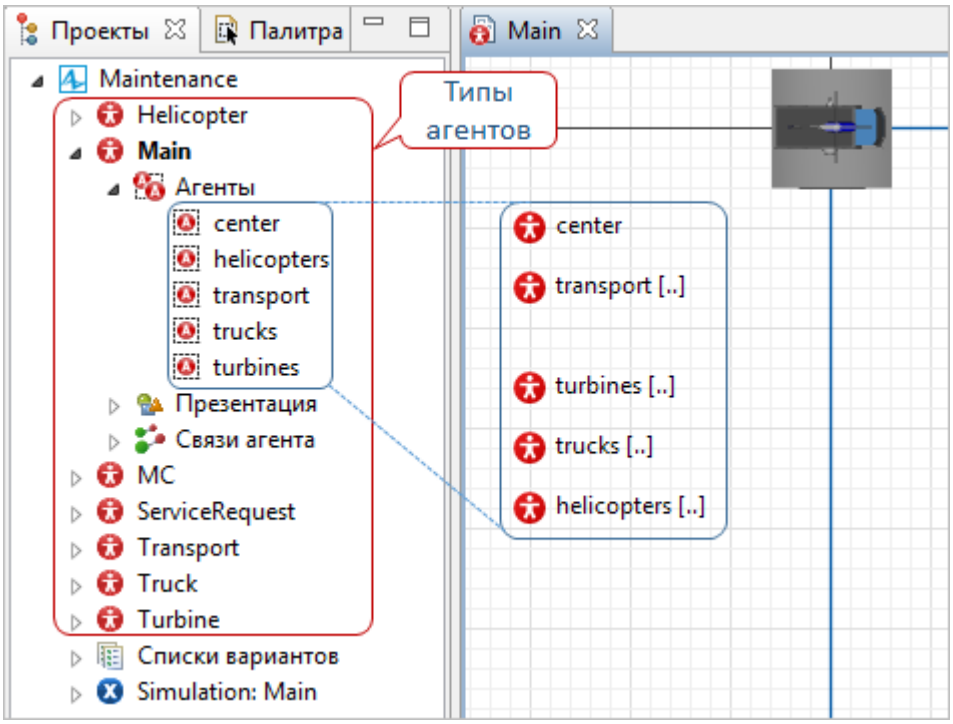


После того, как вы щелкните **Готово**, AnyLogic откроет диаграмму типа агента ServiceRequest. Здесь вы можете увидеть параметр type, который мы создали в мастере.

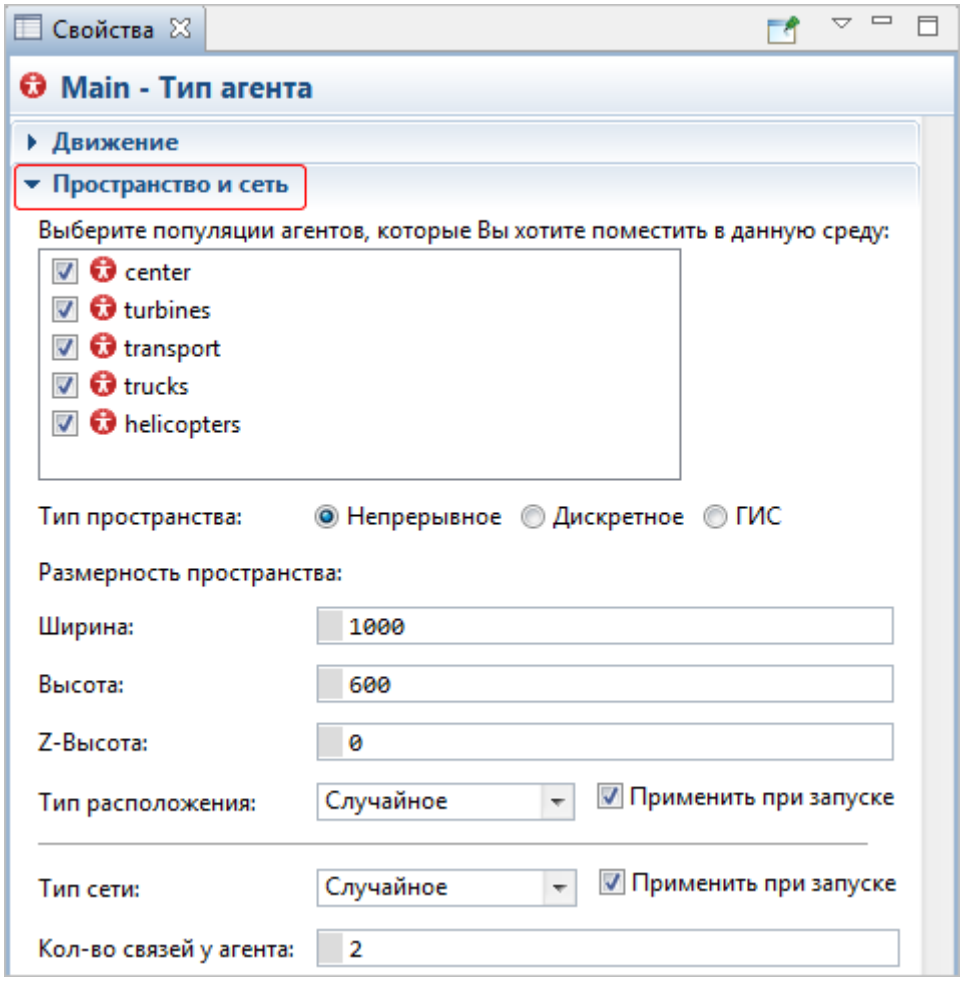
Добавьте еще один **Параметр** из палитры **Агент**. Назовите его turbine и выберите его **Тип** из выпадающего списка: **Turbine**. Теперь тип агента ServiceRequest полностью задан.



Все агенты появятся на диаграмме Main. Можно заметить, что параметры типа популяции отмечены символами **[..]**. Переместите агентов за рамку окна презентации для удобства наблюдения, а центры всех фигур анимации поместите в начало координат.

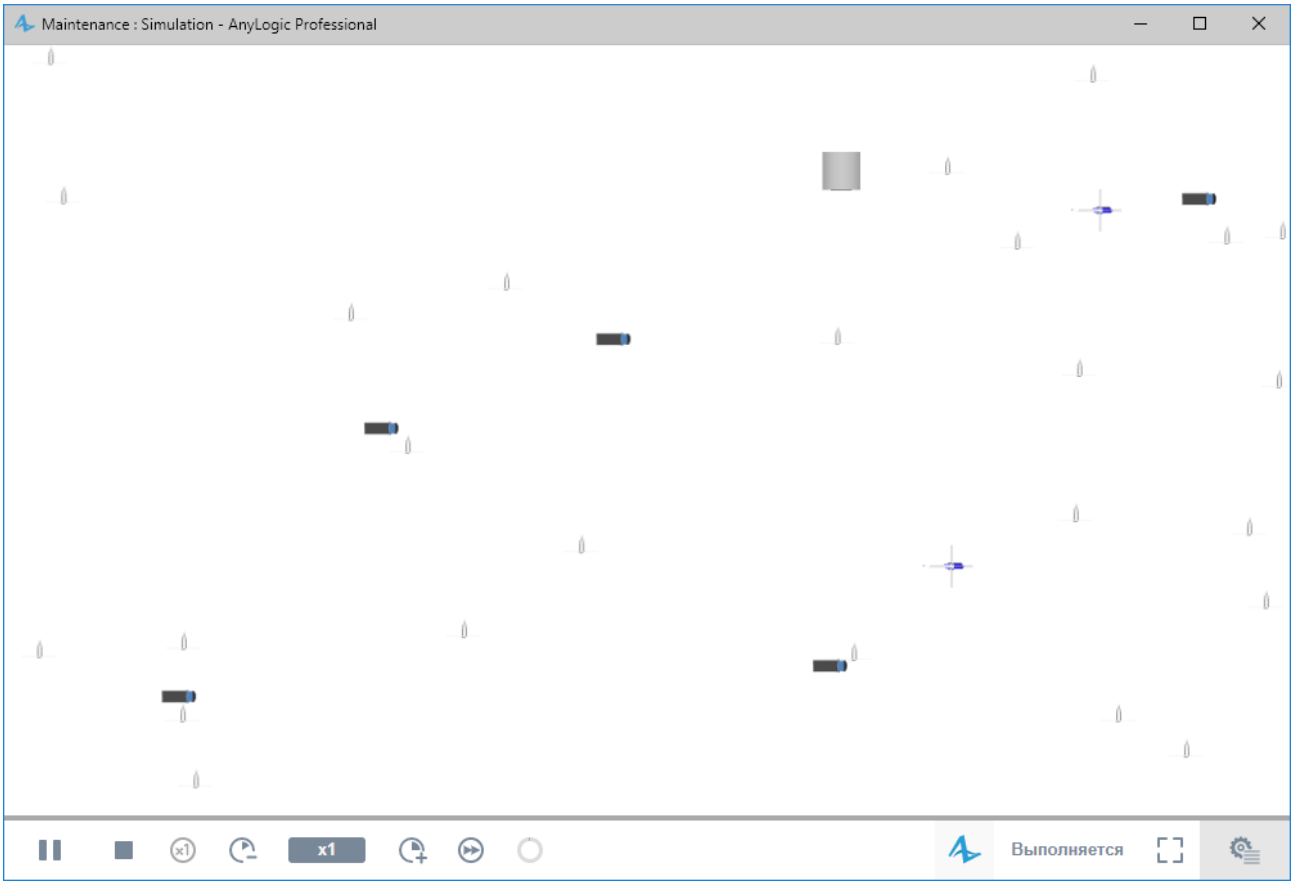


Настройки среды доступны в секции **Пространство и сеть** свойств типа агента Main:



Запустите модель

1. Постройте свой проект, щелкнув кнопку **Построить модель** на панели управления. Если в модели есть какие-либо ошибки, то построение будет неудачным, и вы увидите панель **Ошибки**, в которой будут перечислены все ошибки в модели. Вы можете открыть место ошибки двойным щелчком по ее описанию в списке, чтобы исправить ее. После того, как модель будет успешно построена, вы можете запустить ее. Запуская эксперимент, вы автоматически обновляете модель.
2. Выберите эксперимент, который хотите запустить, открыв выпадающий список рядом с кнопкой запуска модели **Запуск** на панели управления. Наш простой эксперимент называется Maintenance/Simulation. Потом вы сможете запускать этот эксперимент, просто щелкая кнопку **Запуск** , поскольку она будет запускать последний запущенный эксперимент.
3. Запустив модель, вы автоматически начнёте ее выполнение и увидите окно презентации. На ней отображается презентация агента верхнего уровня модели. Мы только начали разрабатывать модель. В данный момент мы можем видеть сервисный центр, турбины и разные типы транспорта, случайно расположенные в непрерывном пространстве.

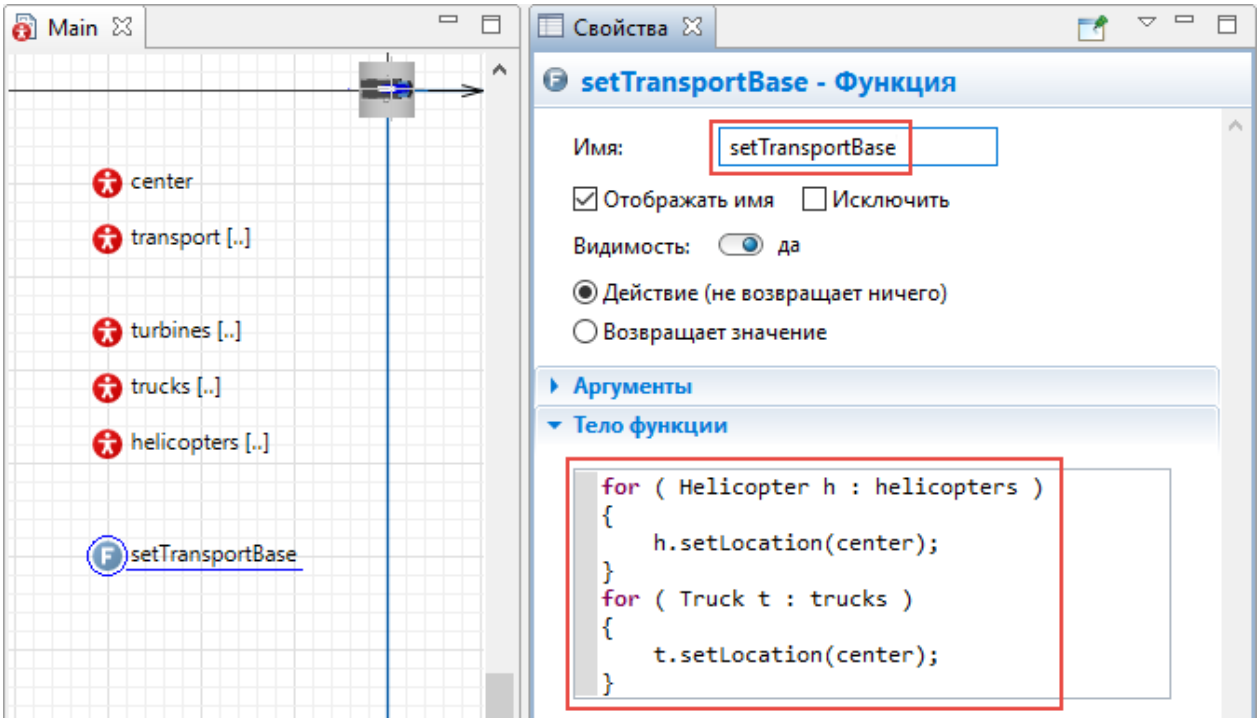


Фаза 2. Задание транспортной базы

Разрабатывая дальше модель, мы будем задавать логику и процессы модели на диаграммах каждого агента в отдельности. Для начала мы добавим алгоритм, с помощью которого при запуске модели все наши грузовики и вертолеты будут размещены в сервисном центре.

Поместим исходный транспорт в сервисный центр.

1. Откройте диаграмму Main в графическом редакторе.
2. Перетащите элемент **Функция** из палитры **Агент** на диаграмму Main.
3. Назовите функцию setTransportBase. Эта функция ничего не возвращает, поэтому оставьте выбранной опцию **Действие (не возвращает ничего)**.
4. Разверните секцию свойств **Тело функции** и введите код Java как указано на изображении:



Мы создали функцию, которая разместит транспорт в сервисном центре. Алгоритм этой функции содержит два цикла ***for***.

Первый цикл выполняет итерирование по всем агентам, входящим в популяцию вертолетов. В строке инициализации цикла for используется следующий синтаксис: в круглых скобках вы сначала указываете имя типа агентов, находящихся в популяции (Helicopter). Затем h – имя локальной переменной, которую мы задаем на этом участке кода. Вы можете использовать любое другое допустимое имя (hel, a, item, и так далее). После этого необходимо указать helicopters – имя популяции агентов, итерирование по которой мы будем выполнять.

Поскольку иногда мы хотим совершить не одно, а несколько действий с каждым агентом популяции, необходимо сказать компилятору Java, какие именно выражения Java должны выполняться во время каждой итерации цикла. Для этого мы помещаем необходимые действия в фигурные скобки. В нашем случае это всего лишь одна строка кода: { h.setLocation(center) }

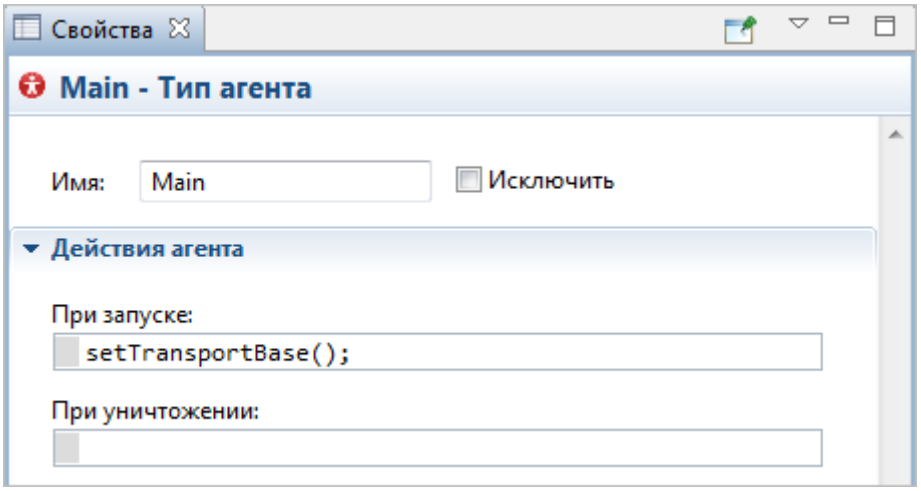
С помощью этого кода мы выполняем следующее: мы задаём местоположение для агента популяции, по которому выполняется итерирование в данный момент, используя ранее заданную локальную переменную h. Местоположение (в нашем случае это сервисный центр center) передаётся в качестве аргумента функции setLocation().

Второй цикл for выполняет то же самое для популяции trucks.

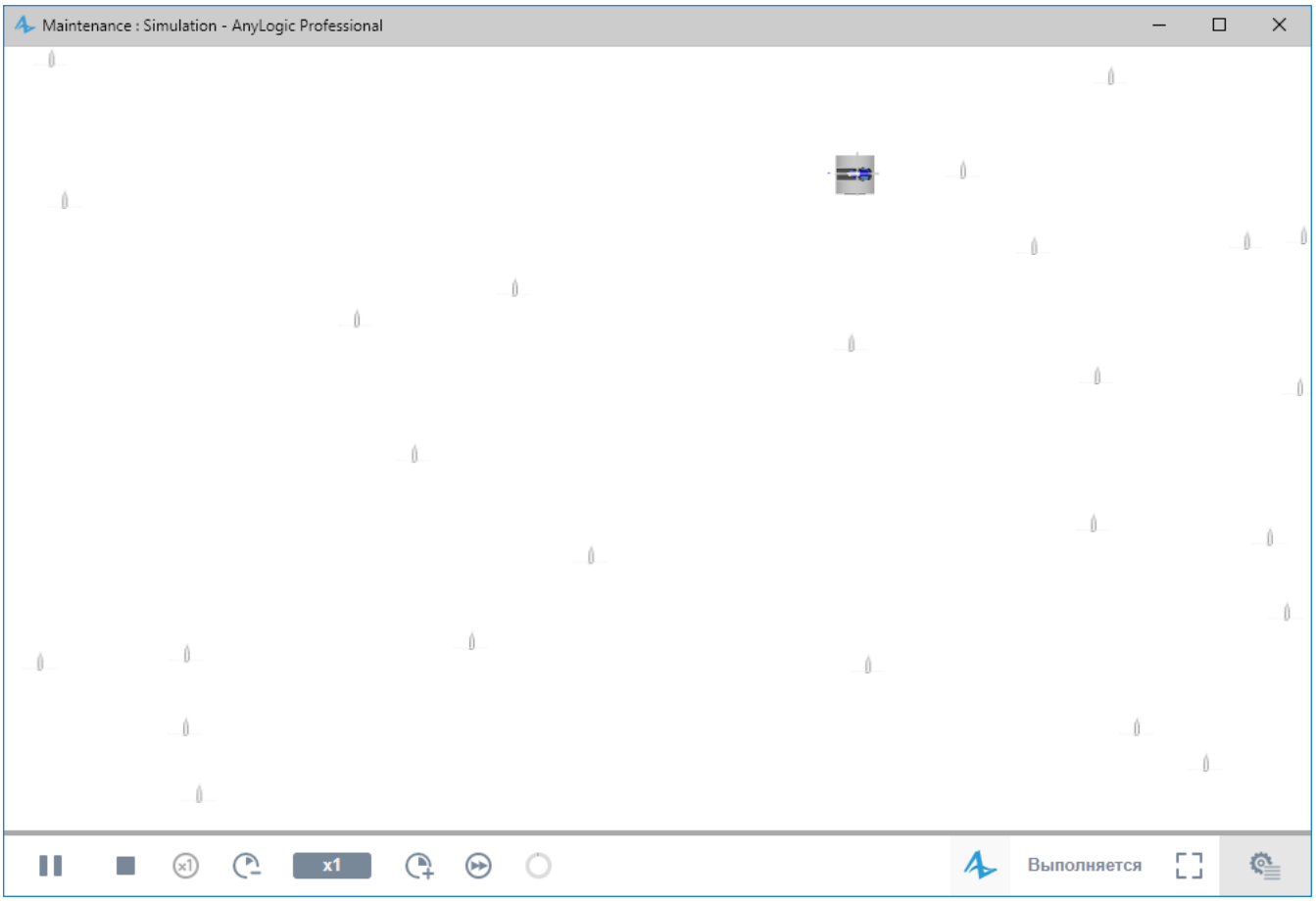
Теперь, чтобы функция заработала, ее нужно вызвать из кода модели.

Выделите Main в дереве модели (или сделайте щелчок мышью в пустом месте графического редактора этого типа агента) и перейдите в панель **Свойства**. Разверните секцию свойств **Действия агента** и поместите вызов нашей функции в поле **При запуске**:

```
setTransportBase();
```



Снова запустите модель. Вы увидите, что все грузовики и вертолеты находятся в сервисном центре. Мы зададим движение транспорта и поведение других агентов на следующих этапах работы.



Фаза 3. Настройка логических процессов транспорта

На этом этапе мы зададим начальную конфигурацию типа агента Transport.

Если у агента можно выделить несколько состояний, выполняющих различные действия при происхождении каких-то событий, или если у агента есть несколько качественно различных поведений, последовательно сменяющих друг друга при происхождении определенных событий, то поведение такого объекта может быть описано в терминах диаграммы состояний. Диаграмма

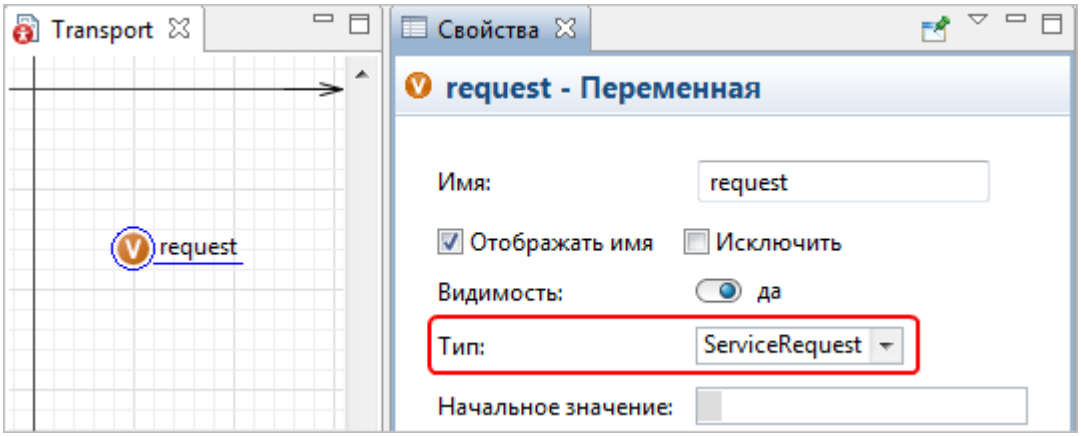
состояний позволяет графически задать пространство состояний алгоритма поведения объекта, а также события, которые являются причинами срабатывания переходов из одних состояний в другие, и действия, происходящие при смене состояний.

С помощью диаграмм состояний можно графически задать дискретные поведения объектов любой сложности, куда более разнообразные, чем элементарные состояния свободен/занят (idle/busy), открыт/закрыт (open/closed), исправен/неисправен (up/down) и т.п., предлагаемые большинством блочных инструментов моделирования.

Диаграмма состояний представляет собой состояния, соединенные переходами. Переходы могут сработать в результате заданного в качестве условия перехода события - это может быть истечение заданного таймаута, получение диаграммой состояний сообщения, выполнение заданного логического условия и т.д. Срабатывание перехода приводит к переходу управления диаграммы состояний в то состояние, в которое ведет этот переход. Состояния могут быть иерархическими, т.е. содержать другие состояния и переходы.

Добавьте необходимые переменные

1. Перейдите в панель **Проекты** и откройте двойным щелчком тип агента Transport.
2. Добавьте элемент **Переменная** из палитры **Агент**.
3. Назовите переменную request и выберите в качестве ее типа ServiceRequest:

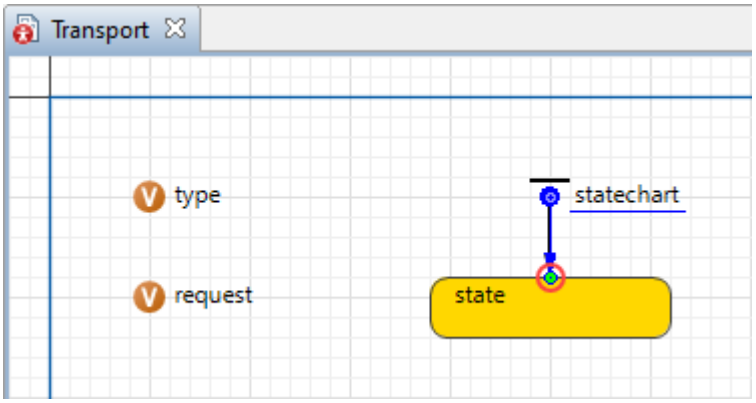


Таким же образом добавьте еще одну переменную. Назовите эту переменную type и в качестве ее типа выберите TransportType.

Нарисуем диаграмму состояний транспорта.

Для этого используем элементы палитры **Диаграмма состояний**, задающие поведение этого типа агентов.

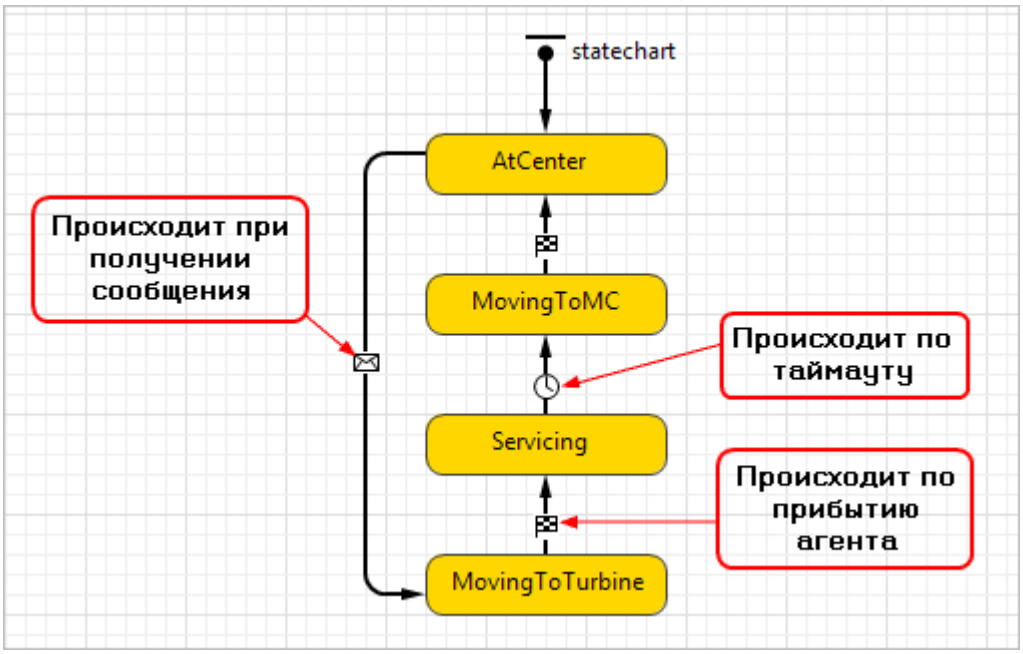
1. Перетащите элемент **Начало диаграммы состояний** в графический редактор типа агента Transport. Этот элемент задает начало всей диаграммы.
2. Добавьте **Состояние**, перетащив этот элемент из палитры **Диаграмма состояний**, и поместите его в конец стрелки начала диаграммы состояний.
3. Чтобы добавить переход, сделайте двойной щелчок по элементу **Переход** в палитре. Его иконка поменяется на, это значит, что элемент теперь в режиме рисования. Вы можете рисовать переходы, делая щелчки по состояниям. Удостоверьтесь, что элементы соединяются друг с другом. Когда они соединены, AnyLogic отображает зелёную точку в месте соединения.



На этом шаге мы создадим "пустую" диаграмму состояний, которая состоит из четырёх состояний, шести переходов разных типов и одного ветвления.

> Состояния называются AtCenter (в сервисном центре), MovingToMC (движение к сервисному центру), Servicing (обслуживание турбины), MovingToTurbine (движение к турбине).

Это, собственно, все действия и передвижения грузовиков и вертолетов в нашей модели. Пожалуйста, следуйте структуре диаграммы, представленной на рисунке ниже.



Внутренние настройки и условия некоторых состояний и переходов должны будут ссылаться на элементы модели, которые мы еще не создали. С другой стороны, нам уже сейчас необходима эта диаграмма, поскольку на нее ссылаются некоторые из тех элементов, которые мы зададим позднее. Поэтому в следующих фазах мы продолжим разрабатывать другие типы агентов, а логику типа Transport закончим в самом конце разработки модели.

Давайте теперь зададим выбор типа транспортного средства в зависимости от запроса и доступность этих средств в сервисном центре.

Фаза 4. Настройка поведения сервисного центра

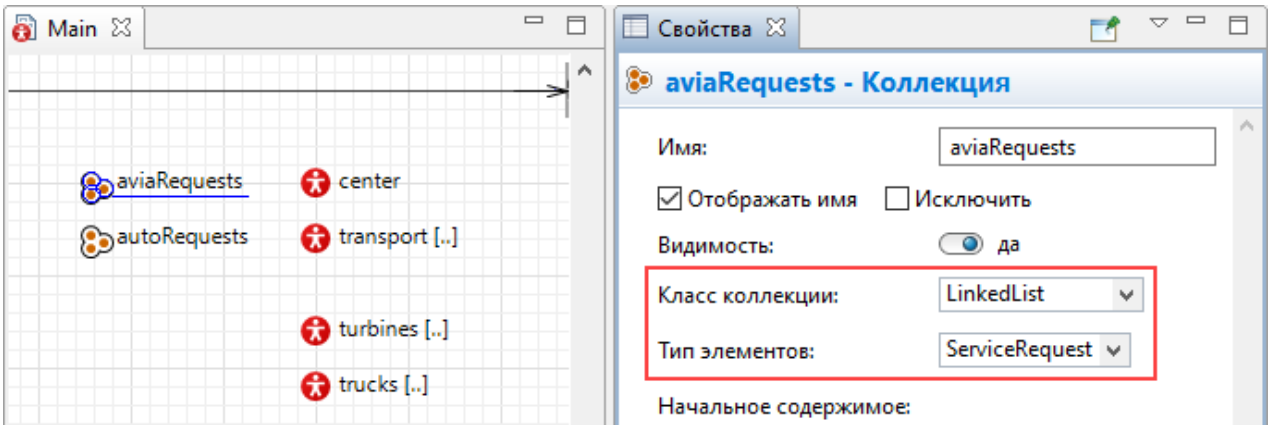
Зададим запросы к транспорту для сервисного центра.

Агент может содержать переменные как в любой программе. Переменные обычно используются для моделирования изменяющихся характеристик объекта или для хранения результатов работы модели.

Коллекция представляет собой группу объектов, известных как ее элементы. Некоторые коллекции позволяют хранение нескольких одинаковых элементов, некоторые - нет. Некоторые коллекции упорядочены, некоторые - нет. Коллекция используется для задания объекта данных, объединяющего в себе сразу несколько однотипных элементов. С помощью коллекций вы можете хранить, извлекать и управлять агрегированными данными. Обычно коллекции представляют элементы данных, которые образуют группу, например, очередь (в этом случае элементы представляют собой людей, ожидающих в очереди) или автопарк (элементы задают автомобили), или телефонный справочник (коллекция хранит соответствие имён и телефонных номеров).

Откройте диаграмму Main. Добавьте два элемента **Коллекция** из палитры **Агент**.

Назовите коллекции aviaRequests и autoRequests. Обе коллекции имеют **Класс коллекции** LinkedList и **Тип элементов** ServiceRequest.

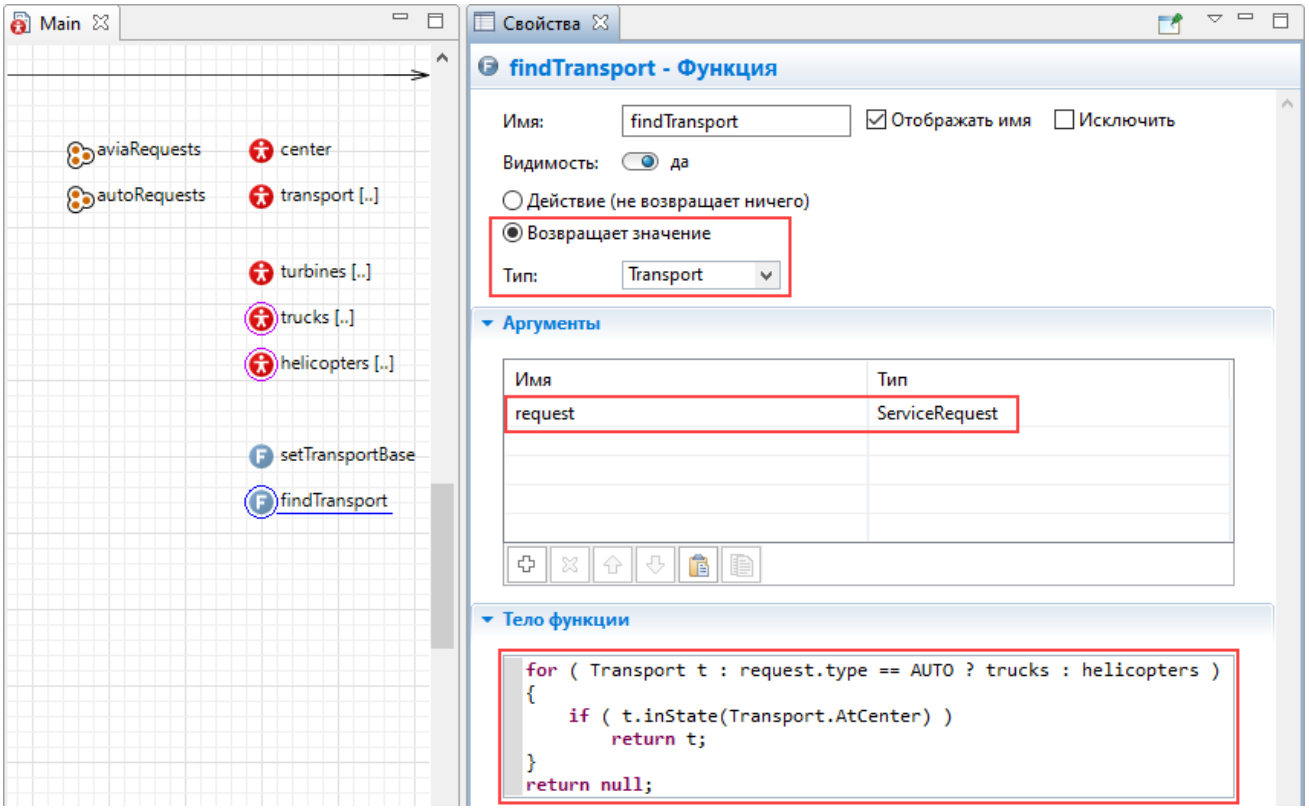


Теперь мы создадим функцию, которая будет искать свободный транспорт, чтобы выполнить сервисные работы.

Зададим логику управления транспортным парком.

1. Перетащите элемент **Функция** из палитры **Агент** на диаграмму Main.
2. Назовите функцию findTransport
3. Эта функция будет выполнять поиск свободного транспорта. Обнаружив свободный транспорт, функция должна его вернуть, поэтому выберите опцию **Возвращает значение** и задайте **Тип** значения Transport.
4. Функция должна проанализировать тип входящих запросов на сервисные работы. Для этого нам необходимо добавить один аргумент функции - с его помощью мы сможем передать функции запрос на сервисные работы. Разверните секцию **Аргументы** в свойствах функции и задайте в таблице аргумент **Типа** ServiceRequest.

Теперь осталось задать алгоритм функции. Разверните секцию **Тело функции** в панели **Свойства** и введите код Java, как указано на изображении:



В теле функции мы задаём цикл ***for***. Этот цикл не всегда выполняет итерирование агентов одной популяции. Здесь он анализирует тип запроса на сервисные работы и проверяет, выполняется ли условие request.type == AUTO. Если условие выполняется, значит, поступил запрос на автомобильный транспорт, и далее цикл производит итерирование популяции trucks. Если же проверка условия возвращает значение false, это значит, что поступил запрос на вертолет. В таком случае цикл выполняет итерацию другой популяции: helicopters.

Внутри цикла мы проверяем, свободен ли в данный момент итерируемый агент. Для этого мы используем функцию агента inState(). Если управление диаграммой состояний агента на данный момент находится в состоянии **AtCenter**, это значит, что мы нашли свободный транспорт. В таком случае выполнение функции завершается на строке return t; и функция возвращает найденное значение (свободный транспорт). Если свободного транспорта нет, эта строка не будет выполняться и после цикла ***for*** выполнится следующая строка кода: return null; Таким образом, мы сообщаем модели, что грузовик не найден (функция вернула значение null - свободный транспорт не существует).

В AnyLogic иногда нужно будет вводить код в параметрах различных элементов модели. Важно понимать, где именно находится объект, вписывая код (какому типу агента принадлежит этот элемент), и как получить доступ к другим элементам из этого поля.

Элементы модели, принадлежащие тому же типу агента, доступны просто по именам.

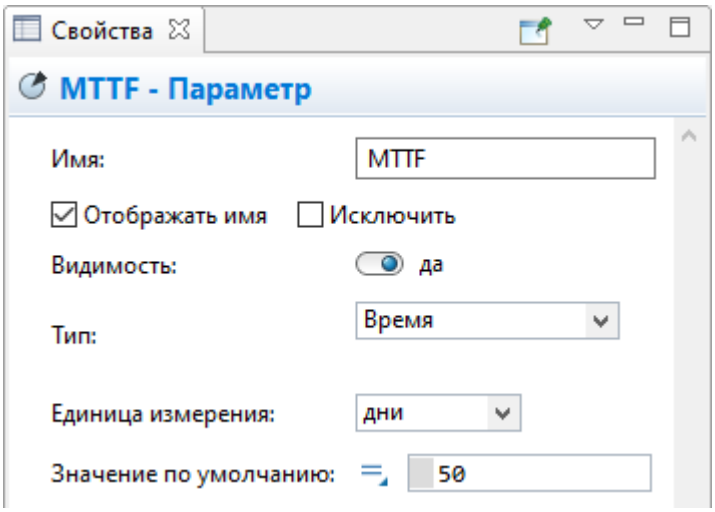
Чтобы получить доступ к полю вложенного объекта, в соответствии с объектно-ориентированной нотацией нужно поставить точку "." после имени объекта и затем написать имя этого поля. Например, чтобы получить доступ к состоянию **AtCenter** диаграммы состояний, которая находится в агенте Transport, из агента Main, мы вызываем Transport.AtCenter.

Фаза 5. Настройка поведения турбин

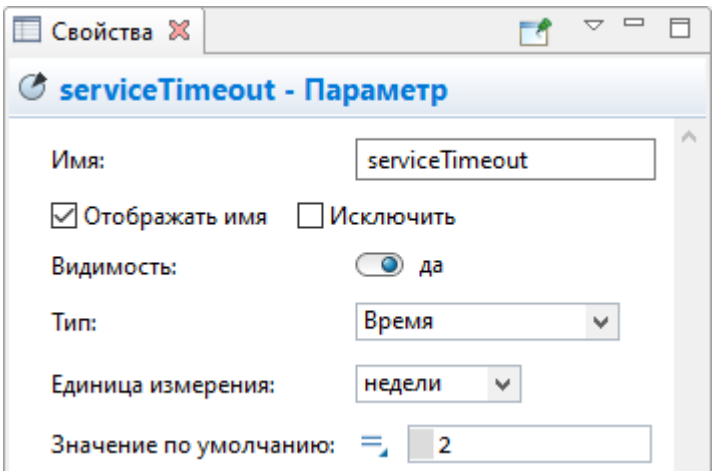
На этом этапе мы зададим поведение турбин с помощью диаграммы состояний: когда и как турбины выходят из строя или требуют планового техобслуживания. Также мы настроим анимацию турбин: пусть лопасти турбины перестают крутиться при экстренной поломке. К тому же, будет удобно увидеть цветовую индикацию состояния турбины.

Зададим временные интервалы работы турбин.

1. Двойным щелчком откройте тип агента **Turbine** из дерева элементов модели. Начните с добавления двух элементов **Параметр** из палитры **Агент** в графический редактор.
2. Параметр с именем MTTF (среднее время до аварии, Mean Time To Failure) имеет тип **Время** и его **Значение по умолчанию** равняется **50 дням**. **Дни** вы можете выбрать в свойстве **Единица измерения**.

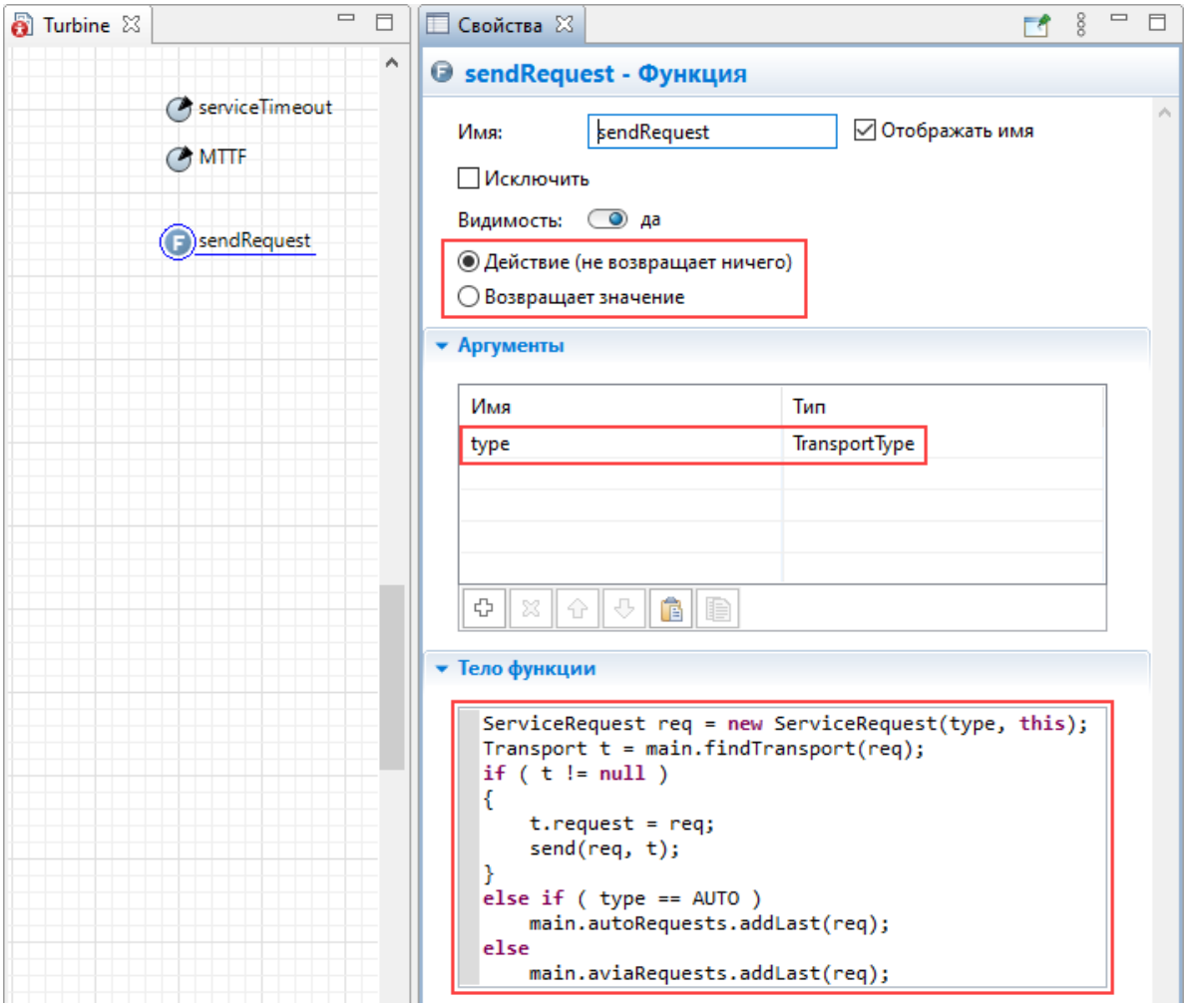


Второй параметр, serviceTimeout, имеет тип **Время**; его Значение по умолчанию равняется **2** неделям.



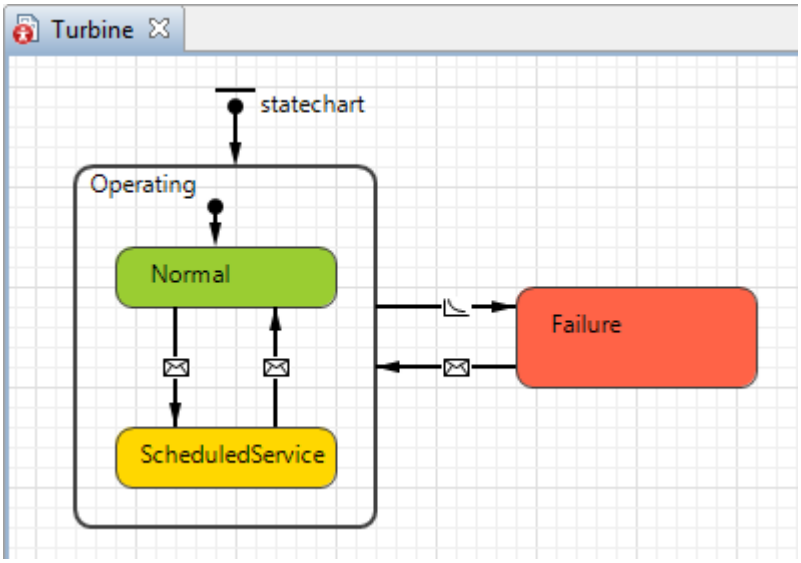
Добавим функции, управляющие запросами на вызов транспорта.

Добавьте **Функцию** из палитры **Агент** и назовите ее sendRequest. Задайте ее свойства, как указано на рисунке ниже. В секции свойств **Аргументы** добавьте аргумент type типа TransportType. Тело функции ссылается на функцию findTransport(), которую мы ранее создали на диаграмме Main. Когда турбина отправляет запрос на обслуживание, сервисный центр должен отправлять соответствующий тип транспорта: AUTO или AVIA.



Зададим состояния турбины:

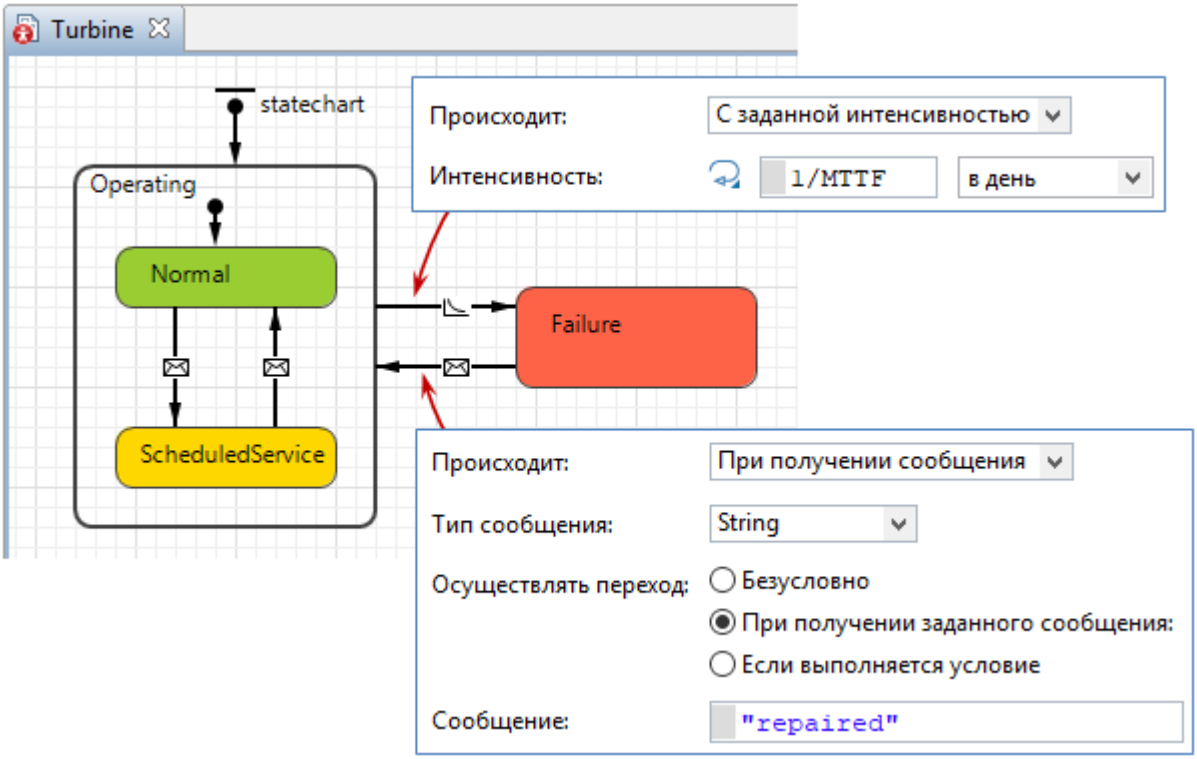
1. Теперь мы готовы приступить к созданию диаграммы состояний турбины. Откройте палитры **Диаграмма состояний** и перетащите на диаграмму Turbine элемент **Начало диаграммы состояний**.
2. Добавьте состояния и переходы , как показано на рисунке. Внутри сложного состояния Operating используйте **Указатель начального состояния**, ведущий в состояние Normal.



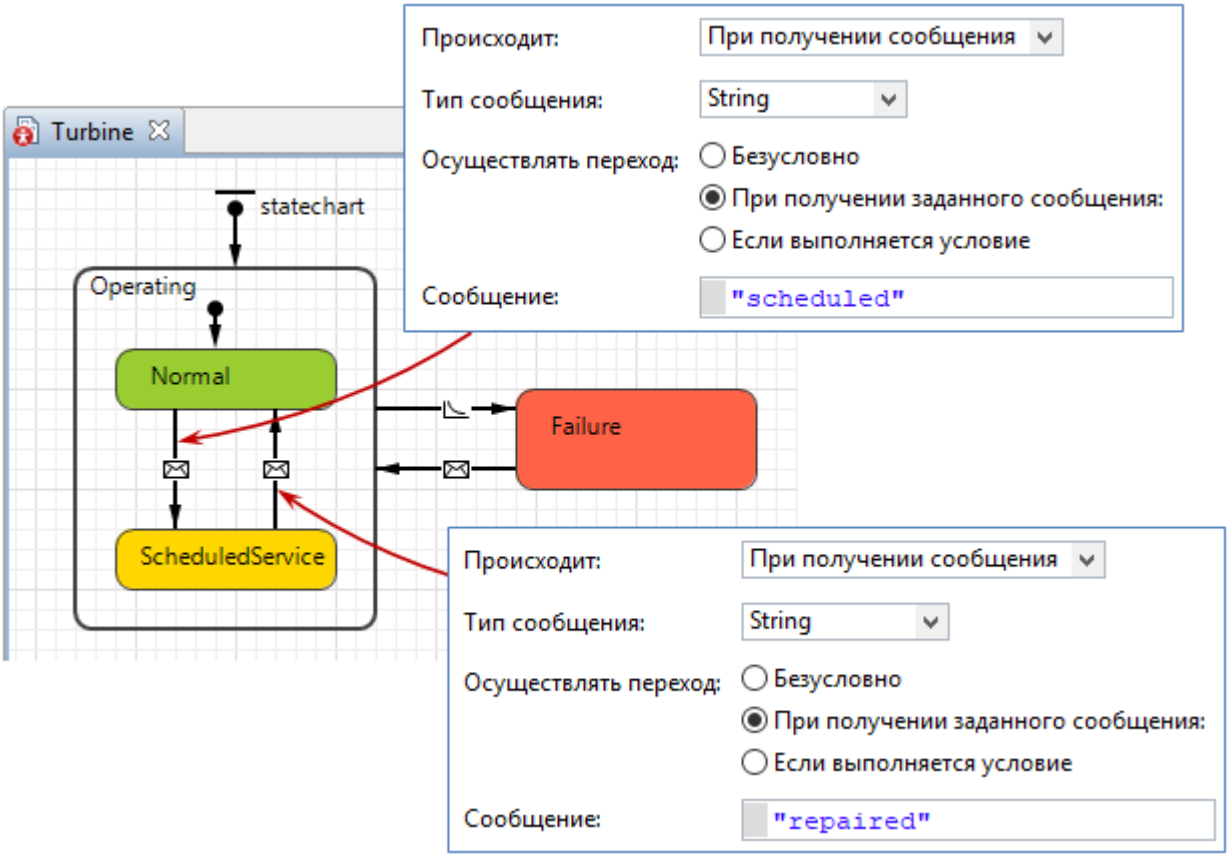
Для каждого состояния турбины задаётся своё **Действие при входе**. Когда происходит авария, турбина находится в состоянии Failure (авария) и отправляет запрос на обслуживание AVIA транспортом (вертолетом). Когда подходит время планового обслуживания, турбина отправляет запрос на AUTO транспорт - грузовик.

Состояние	**Действие при входе**
Failure	sendRequest(AVIA);
ScheduledService	sendRequest(AUTO);

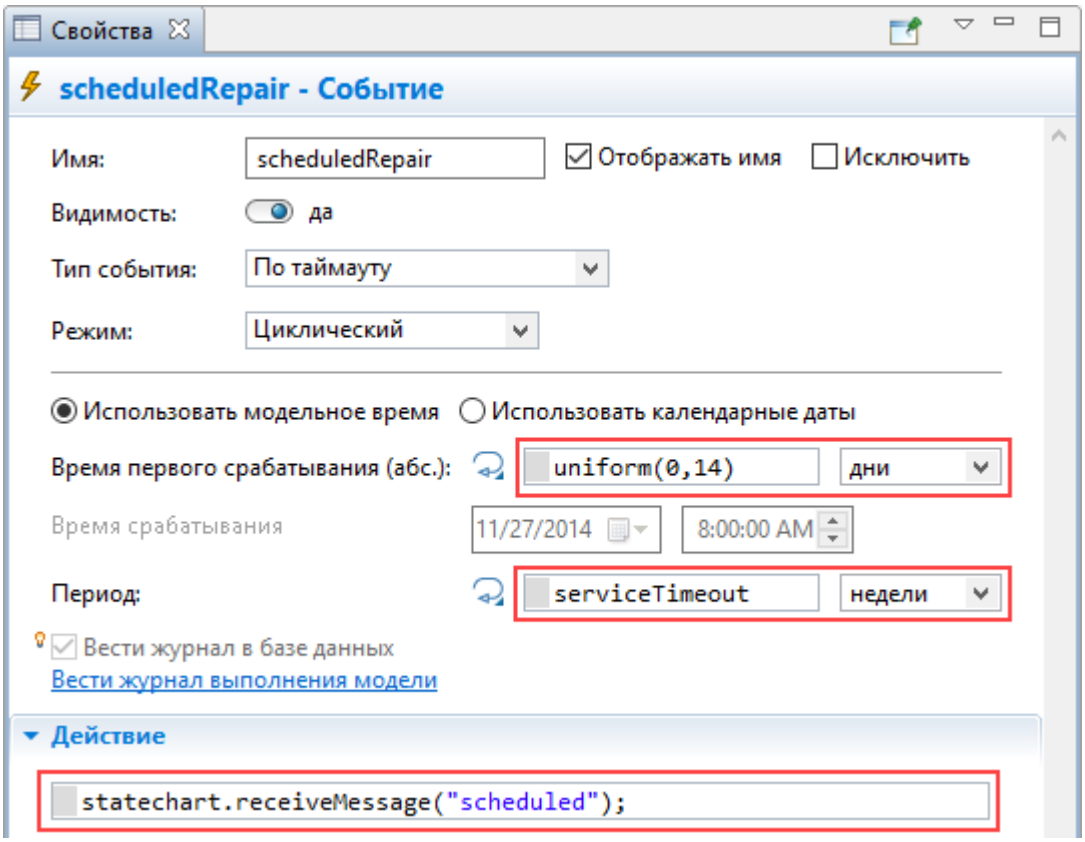
Переход от состояния Operating (работает) к состоянию Failure происходит **С заданной интенсивностью**, которая равняется 1/MTTF в день - один раз за среднее время до аварии. Переход обратно из состояния Failure в состояние Operating происходит **При получении заданного сообщения** ✉ - "repaired" (исправлено).



Иногда работающая турбина получает сообщение "scheduled" (запланировано), и это означает, что ей требуется плановое обслуживание - при этом происходит переход в соответствующее состояние ScheduledService. Когда турбина получает сообщение "repaired", обслуживание завершено, и она снова может вернуться в рабочее состояние Operating.

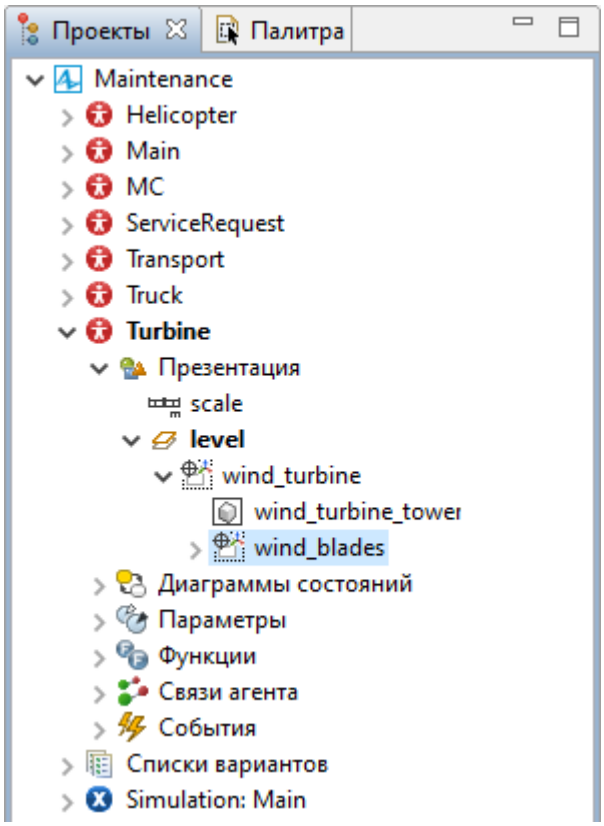


Добавьте циклическое **Событие по таймауту** и назовите его scheduledRepair Это событие будет запускаться, когда турбине требуется плановое обслуживание (согласно переменной serviceTimeout), и оно будет запускать переход от состояния Operating в состояние ScheduledService.

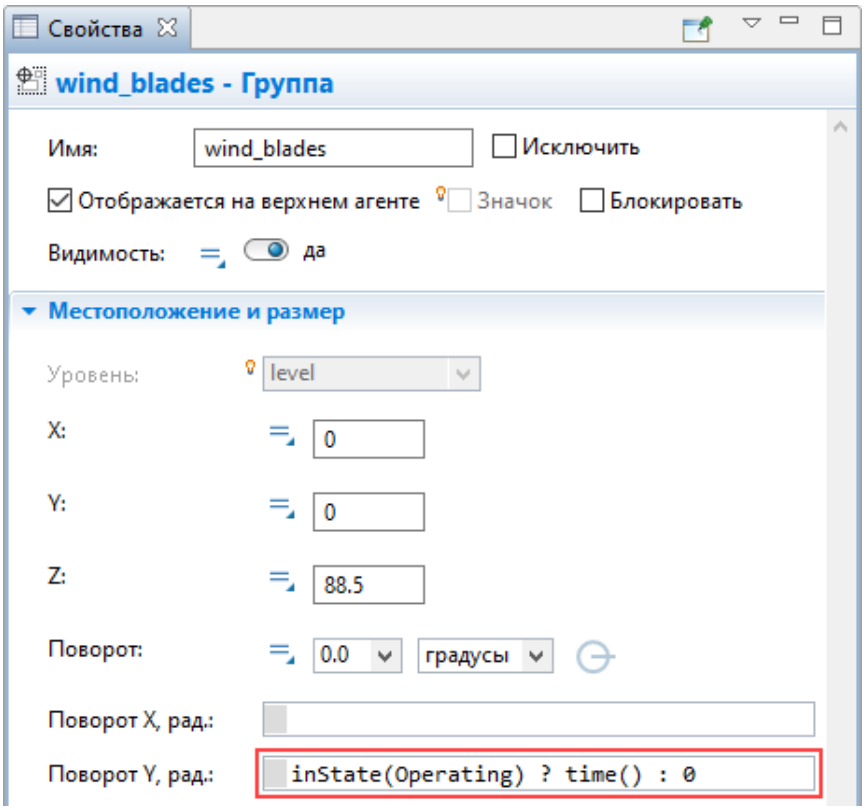


Настройте анимацию лопастей турбины

Нам необходимо изменить свойства лопастей турбины. Откройте панель **Проекты** и выберите группу **wind_blades**:

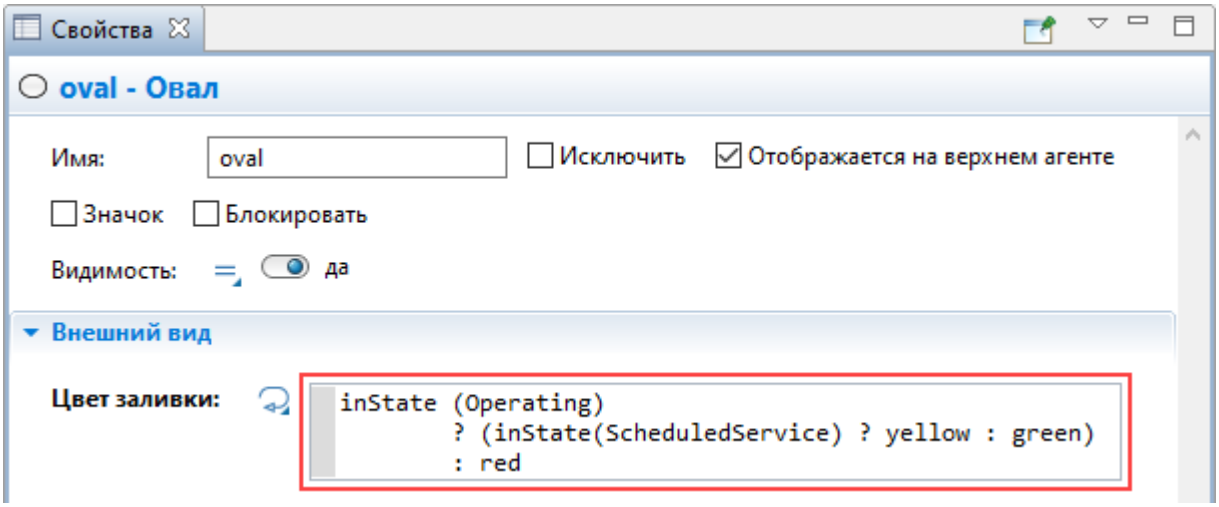


Перейдите в свойства этой группы. Разверните секцию **Местоположение и размер** и введите inState(Operating) ? time() : 0 в поле свойства **Поворот Y**. Теперь лопасти будут вращаться, когда турбина находится в рабочем состоянии, а при поломке турбины будут останавливаться.



Добавим индикацию состояния турбины:

1. Откройте палитру **Презентация** и сделайте двойной щелчок по элементу **Овал**, чтобы перейти в режим рисования для этого элемента.
2. Нарисуйте круг вокруг фигуры турбины радиусом 10.
3. Щёлкните по фигуре круга правой кнопкой мыши и выберите **Порядок > На задний план** в контекстном меню.
4. Перейдите в секцию **Внешний вид** свойств круга. Введите выражение, которое будет вычисляться во время прогона модели в поле свойства **Цвет заливки**, чтобы цвет менялся в зависимости от состояния турбины.
5. Чтобы иметь возможность задать динамическое значение в поле свойства, щёлкните его значок .



Если турбина ожидает запланированного обслуживания, круг станет отображать жёлтый цвет, иначе - зелёный, когда турбина в рабочем состоянии; в случае, когда турбина выходит из строя, мы получим красный цвет фигуры.

Дополнительно, можно изменить размер элементов диаграммы состояний и изменить цвет состояний по умолчанию на более подходящий.

Запустите модель, и теперь увидите, что некоторые турбины работают, некоторые ожидают планового обслуживания, а некоторые - вышли из строя.

Фаза 6. Завершение настройки логики транспорта

На этом шаге мы добавим необходимые настройки в диаграмму состояний типа агента Transport, зададим движение транспорта между сервисным центром и турбинами с помощью переходов различных типов.

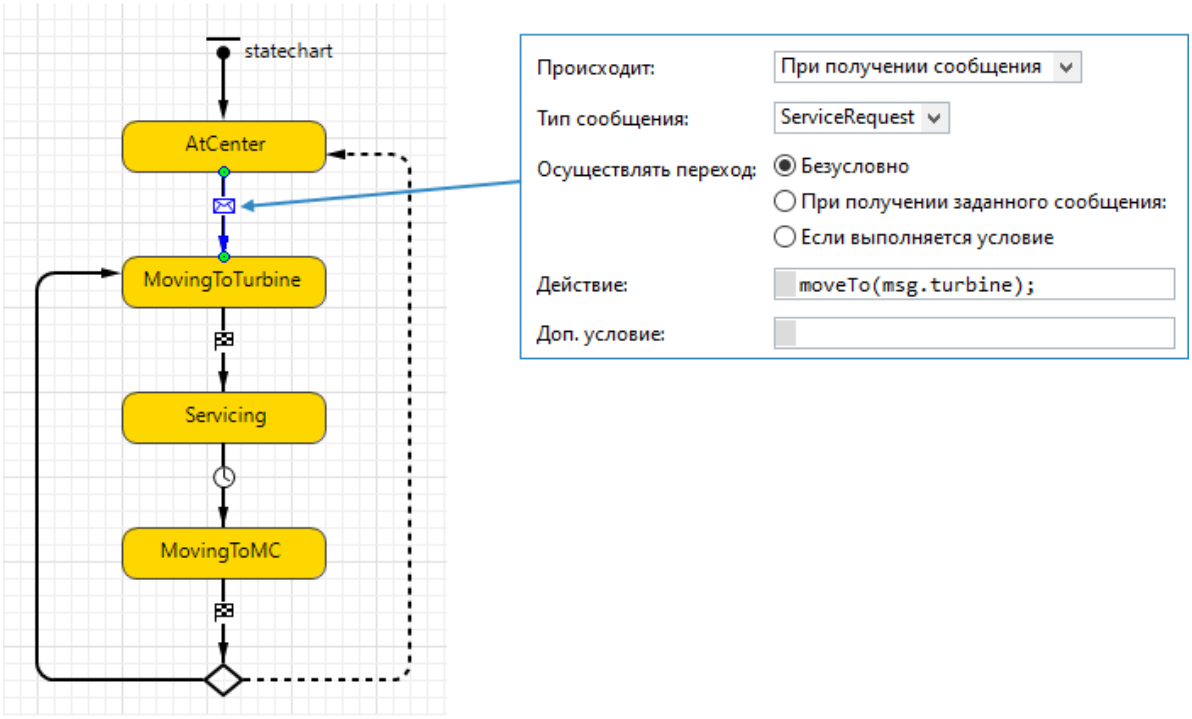
Укажем тип транспорта.

Откройте графическую диаграмму агента Truck, дважды щёлкнув по соответствующему элементу в дереве модели. В секции свойств **Действия агента** укажите тип агента в поле **При запуске** с помощью следующего выражения: type = AUTO;

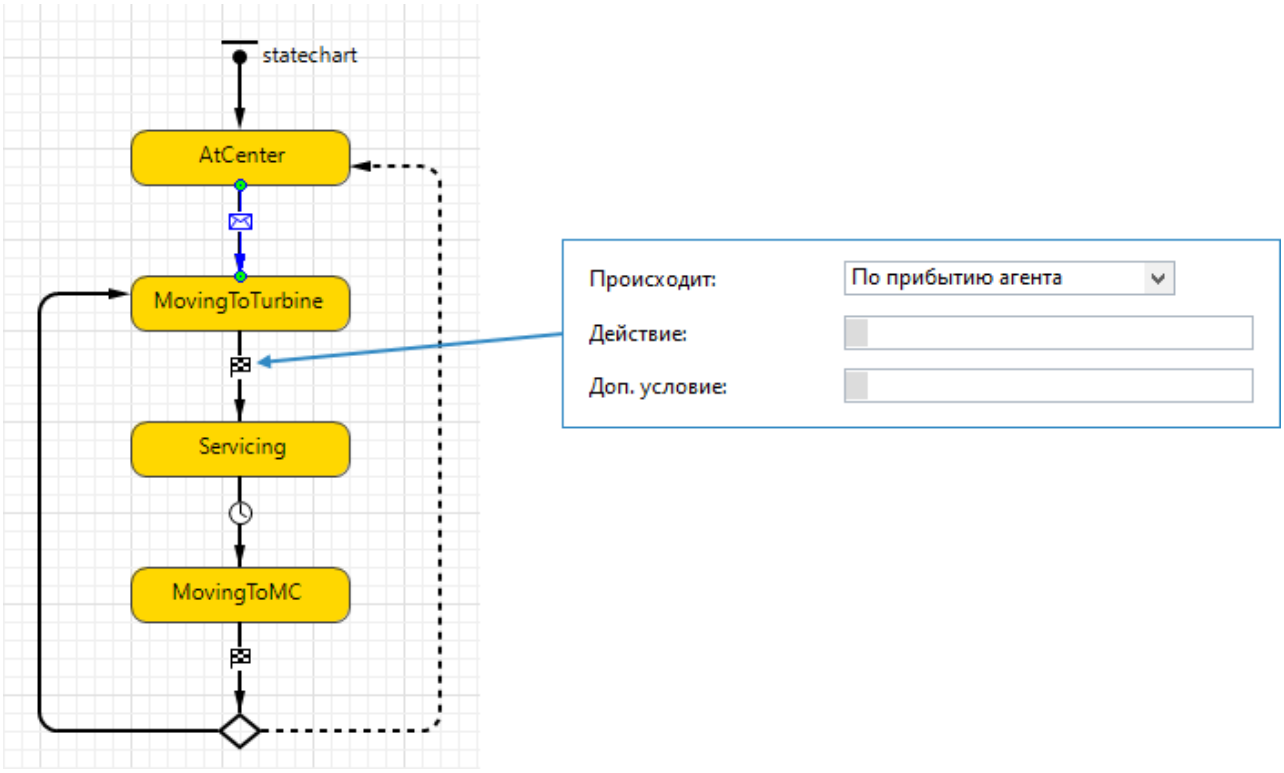
Тем же способом в свойствах агента Helicopter укажите соответствующий тип: type = AVIA;

Настроим диаграмму состояний:

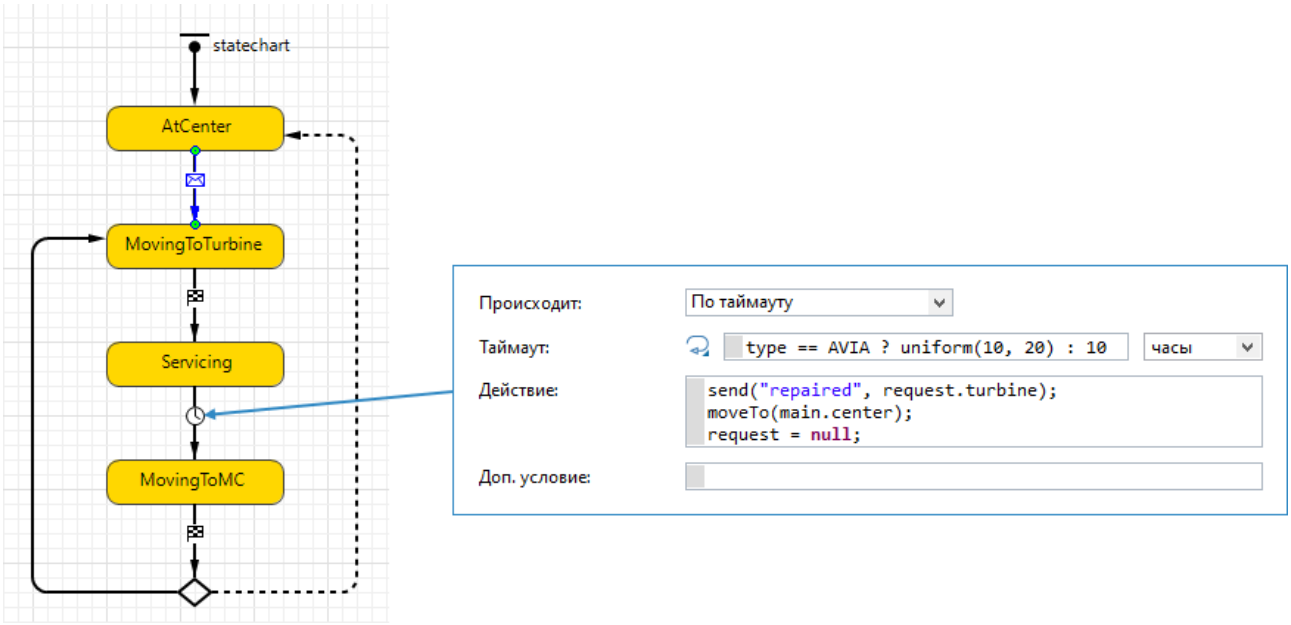
1. Дважды щелкните по агенту Transport, чтобы открыть его графическую диаграмму. На ней вы обнаружите ранее созданную диаграмму состояний. Теперь мы можем задать ее настройки.
2. У состояний в диаграмме (**AtCenter**, **MovingToTurbine**, **Servicing**, **MovingToMC**) нет каких-то особенных свойств. Мы будем задавать логику движения транспорта с помощью различных переходов между состояниями.
3. Переход из состояния **AtCenter** в состояние **MovingToTurbine** происходит **При получении сообщения** ✉ типа ServiceRequest. Здесь необходимо указать **Действие**, которое следует выполнить транспорту: двигаться к турбине, от которой пришло сообщение.
moveTo(msg.turbine);



Переход из состояния **MovingToTurbine** в состояние **Servicing** запускается **По прибытию агента** 🏁 Грузовик или вертолёт, который достиг турбины, начинает выполнять свою задачу сразу после прибытия на место. По завершению обслуживания, о чем нас "оповещает" турбина, транспортное средство может отправляться обратно на базу.

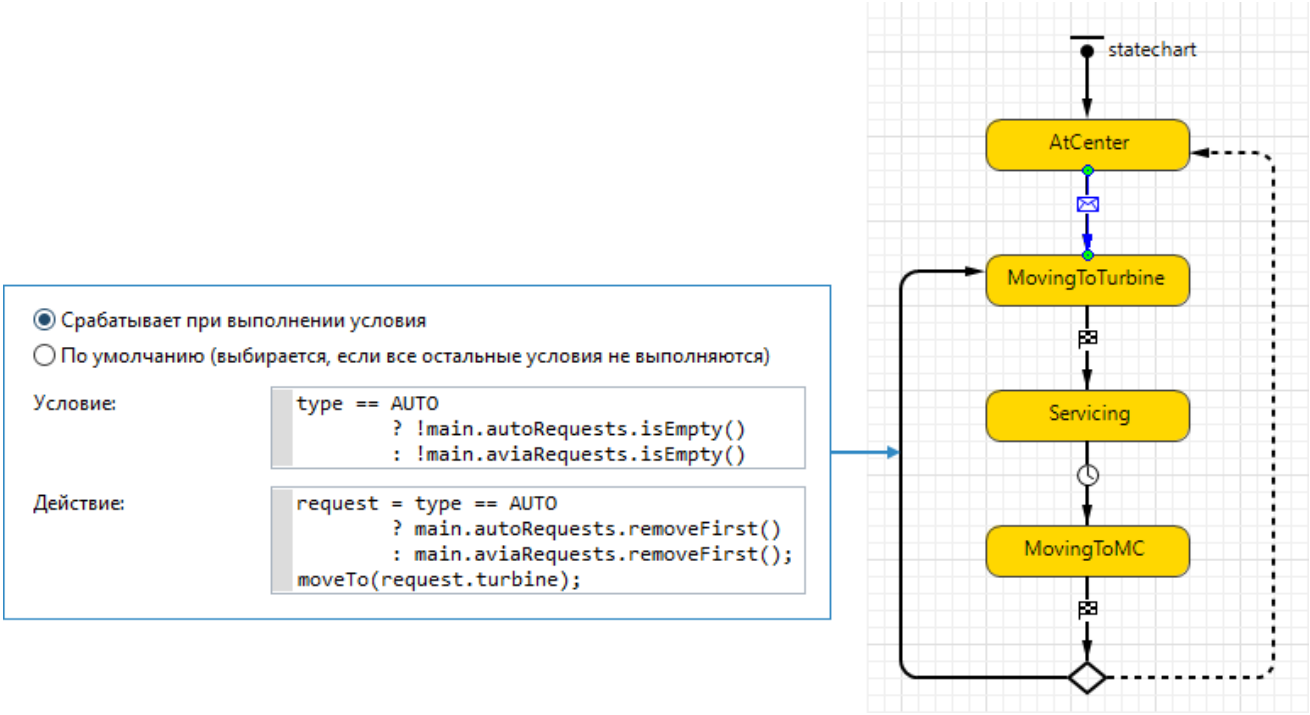


Следующий переход идёт из состояния **Servicing** в состояние **MovingToMC**. Действие, задающее возвращение в сервисный центр, происходит **По таймауту** .

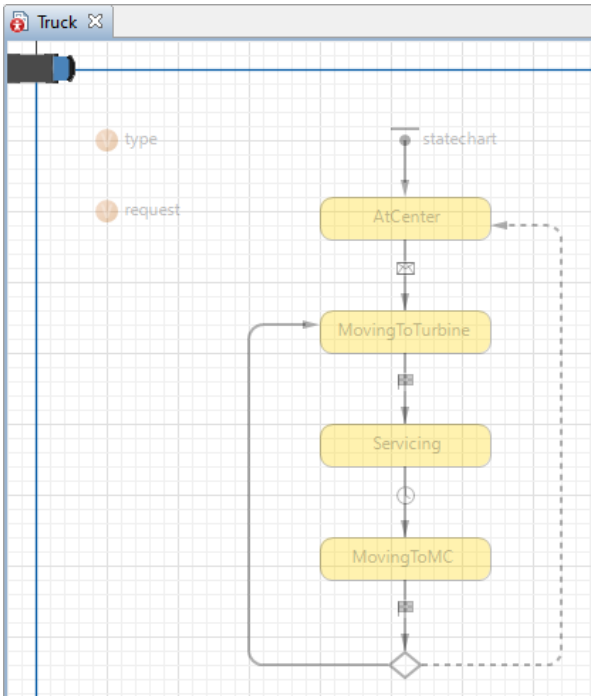


На рисунке видно, что последний переход, из **MovingToMC** в ветвление, откуда транспорт может направиться либо в состояние **MovingToTurbine**, либо вернуться в **AtCenter**, происходит **По прибытию агента** . Когда он происходит, выполняется проверка, при прохождении которой транспортное средство либо выезжает на обслуживание очередной турбины, либо отправляется обратно в сервисный центр и остается там, пока снова не понадобится для обслуживания.

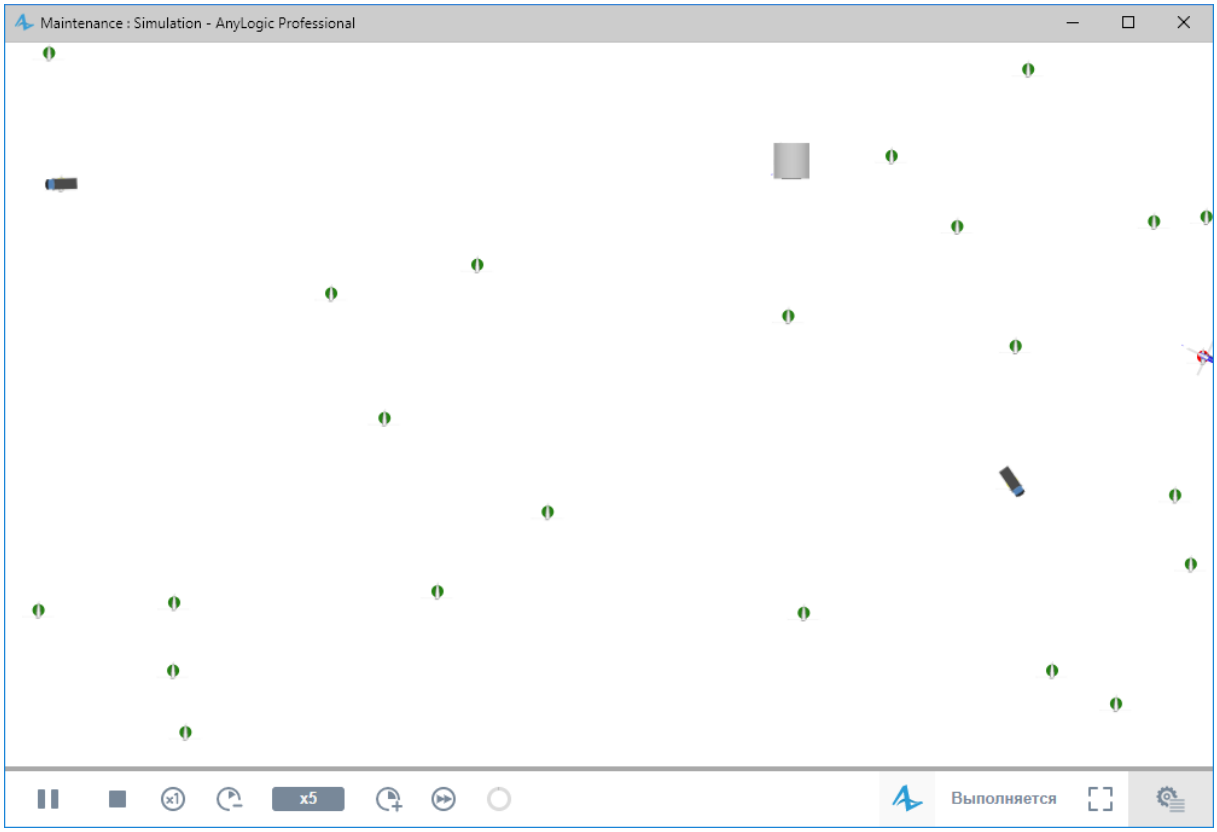
Выберите переход из ветвления в состояние **MovingToTurbine**. Здесь нам необходимо задать **Условие** и **Действие**: код в поле **Условие** проверит, есть ли в данный момент запросы от турбины на запланированные ремонтные работы или срочный ремонт из-за поломки. Если такие запросы есть, код в поле **Действие** проверит, какой именно запрос поступил, и отправит соответствующий тип транспорта к турбине:



Теперь, если откроете тип агента Truck или Helicopter, вы увидите там проекцию элементов типа агента Transport, который они расширяют. Эти элементы отображаются на их диаграммах для удобства, но чтобы изменять их свойства, вернитесь обратно на диаграмму самого агента:



Запустите модель. Изначально все турбины находятся в рабочем состоянии (обозначаются зеленым цветом). Затем вы увидите, как грузовик направляется к той турбине, которой требуется плановое обслуживание (желтые турбины), а вертолет направится к той турбине, которая вышла из строя (красные турбины), чтобы выполнить срочные ремонтные работы.



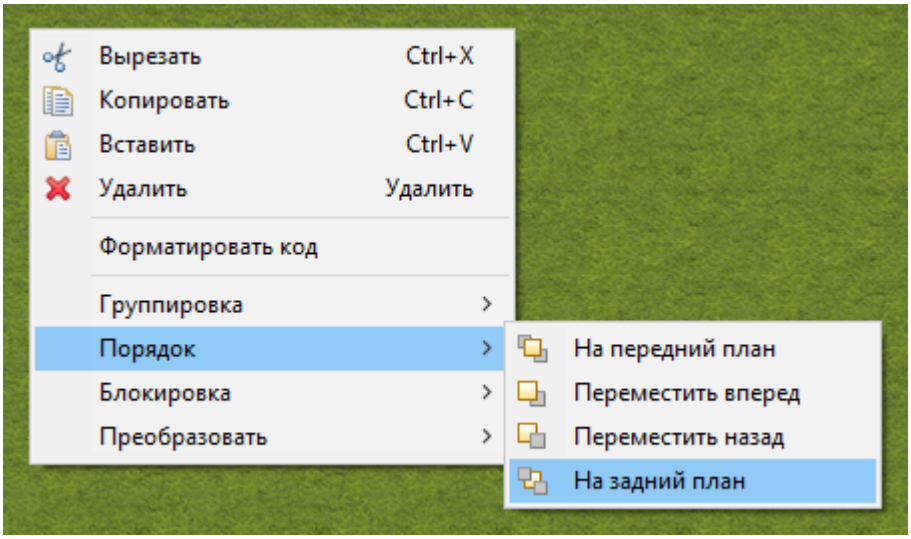
Фаза 7. Запуск модели и исследование ее элементов

Это последний шаг в разработке агентной модели обслуживания турбин. Добавим элементы презентации.

Например, можете добавить фон для анимации агентов - турбин и транспортных средств - на диаграмме типа агента **Прямоугольник**. Выделите элемент **Прямоугольник** в палитре **Презентация**, чтобы перейти в режим рисования. Затем щёлкните в графический редактор и тащите прямоугольник, не отпуская кнопки мыши, пока не получите нужную форму прямоугольника.

Перейдите в секцию свойств фигуры **Внешний вид** и выберите текстуру **Grass** в качестве **Цвета заливки** и опцию **Нет цвета** для **Цвета линии**. Обратите внимание на свойства **Z** и **Z-Высота** в секции **Местоположение и размер**. Например, если **Z-Высота** равняется **10**, вы можете установить **Z** на **-10**, иначе другие фигуры анимации "утонут" в этом фоне, ведь они по умолчанию перемещаются на нулевом уровне.

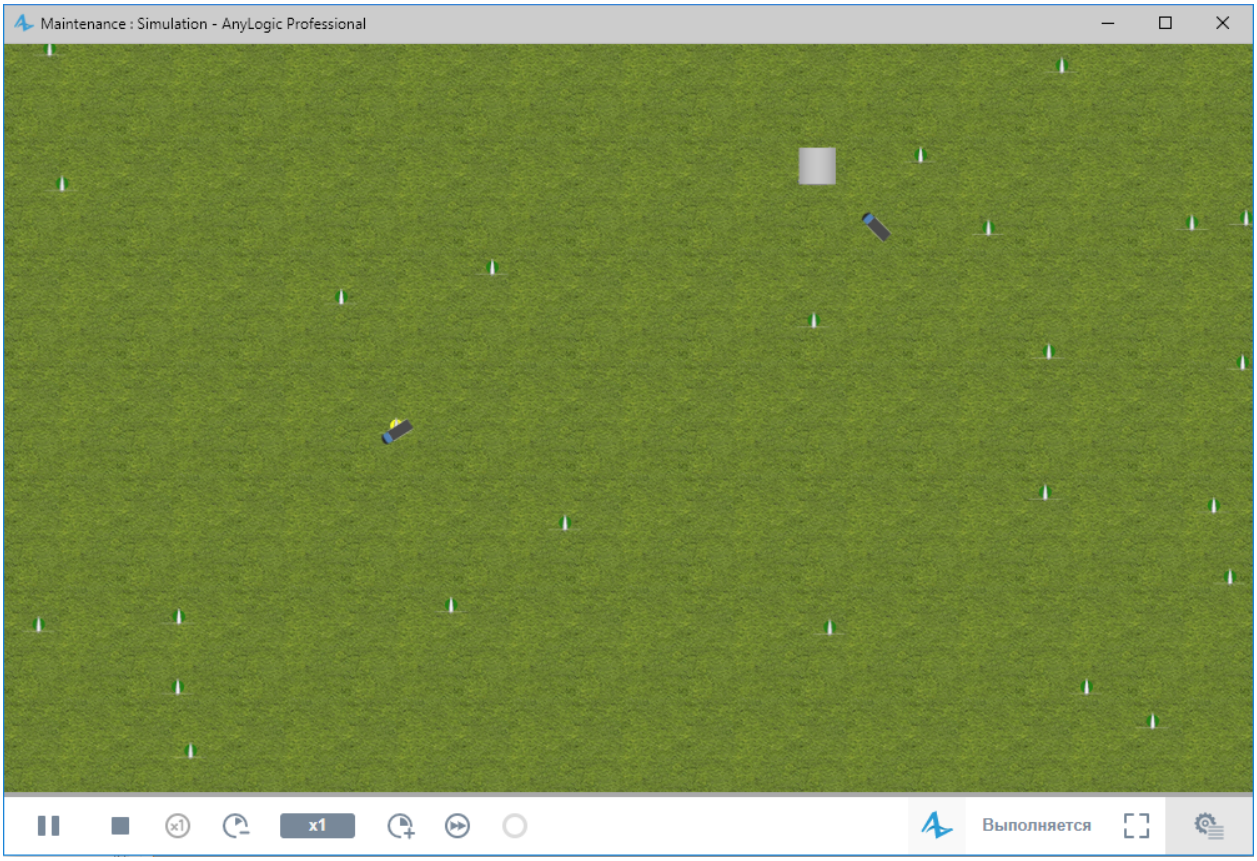
Так как прямоугольник – это последняя добавленная нами фигура, он отображается поверх всех остальных. Щелкните прямоугольник правой кнопкой мыши и выберите **Порядок > На задний план**.



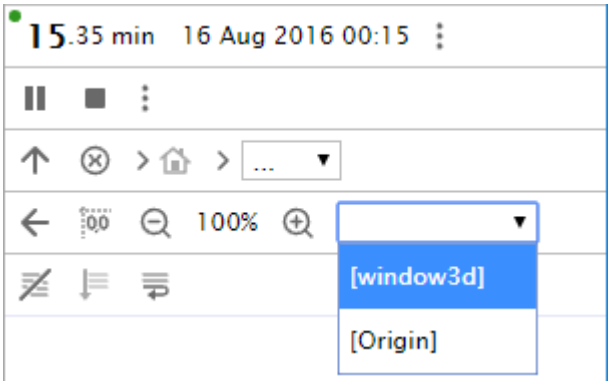
Также можно щелкнуть фигуру анимации агента MC и выбрать **Порядок > На передний план**, чтобы "спрятать" транспорт за ней.

Чтобы задействовать 3D анимацию, откройте палитру **Презентация** и перетащите элемент **3D Окно** в графический редактор. Затем можно изменить его **Свойства**: размер или цвет.

Щёлкните по кнопке панели управления **Запуск** и запустите модель.

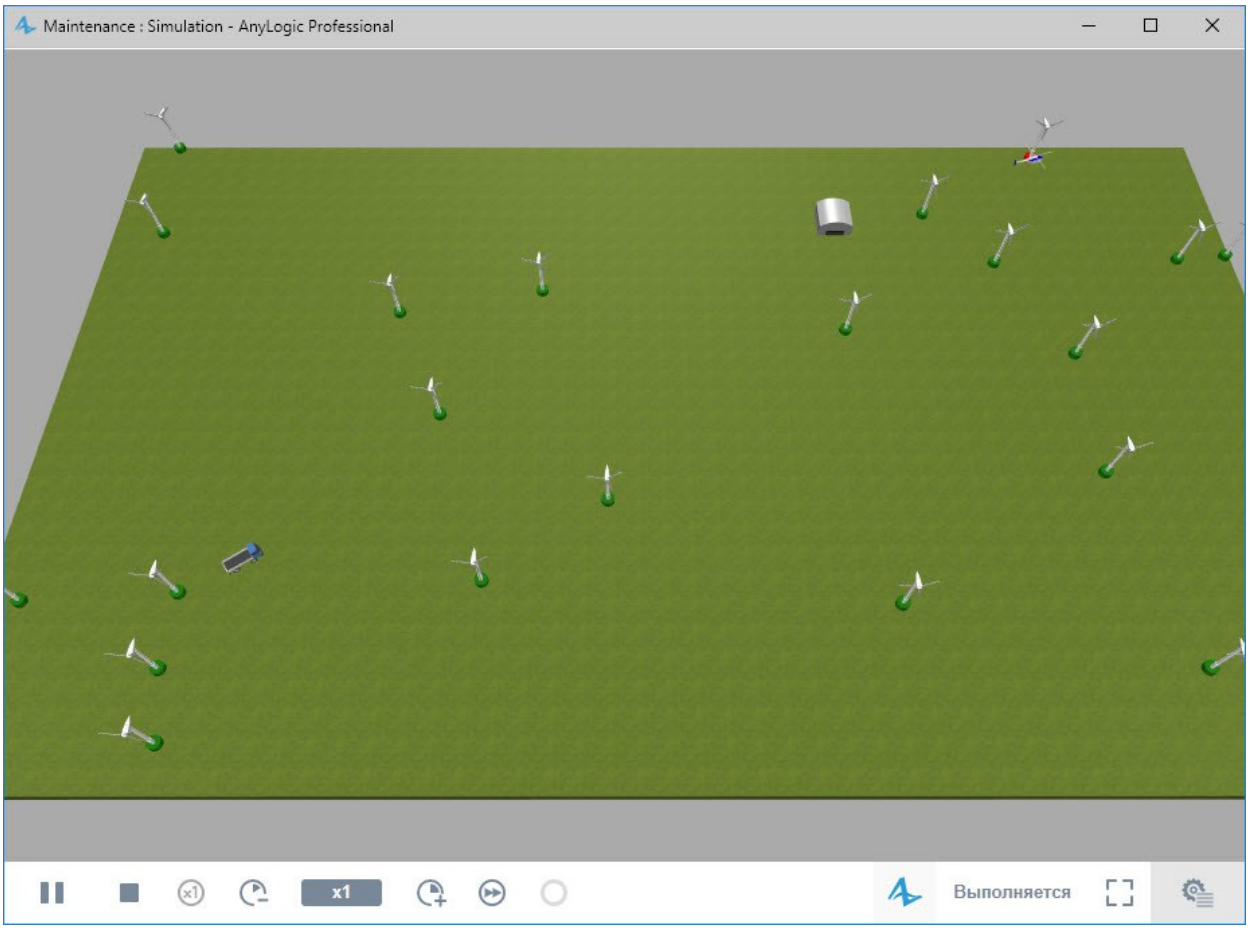


При создании 3D окна, AnyLogic добавляет область просмотра, которая позволяет легко переключаться к сцене 3D анимации во время выполнения модели. Чтобы переключиться к 3D анимации в запущенной модели, откройте панель разработчика, щелкнув **Показать /скрыть панель разработчика** в правом нижнем углу панели управления. В открывшейся панели разработчика, раскройте список **Выбрать область и показать** и выберите опцию **[window3d]**.



Можно перемещаться по 3D сцене с помощью мыши и следующих клавиш:

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



Вы можете управлять запуском модели с помощью кнопок панели управления: **Пауза** и **Прекратить выполнение эксперимента** , **Ускорить** или **Замедлить** модель.

Кроме того, можно следить за разными типами агентов. В панели разработчика раскройте список **Выбрать агента уровнем ниже и перейти** и выберите **turbines[10]**. В квадратных скобках указано количество агентов в этой популяции. Выбрав этот элемент в списке, вы увидите диаграмму первого агента популяции турбин. Чтобы переместиться к другому агенту этой популяции, введите индекс требуемого агента справа. Таким способом можно наблюдать за любым агентом в модели.

Как видно из модели, сейчас и грузовики и вертолёты паркуются прямо в точке расположения турбины, что, конечно, придаёт анимации некую нереалистичность. Мы можем поправить и этот момент, слегка сдвинув анимацию турбины от ее логической точки местоположения. Для этого откройте диаграмму турбины Turbine и переместите анимацию турбины чуть вверх (например, чтобы лопасти стали располагаться прямо по оси X).

Создание нашей модели завершено, и теперь можно добавить в эту модель необходимую аналитику.

Случай из практики эксплуатации ВЭС

В Германии обрушилась одна из крупнейших и самых новых в стране наземных ветряных турбин, находившаяся в лесу в Рурской области земли Северная Рейн-Вестфалии. Ветроэлектростанция запущена в работу в марте 2021 года, и 29 сентября полностью разрушилась. Ветряк был большой: высота башни 149 метров, вес башни 1200 тонн, фундамент - 2100 тонн, диаметр ветроколеса - 240 м.. Согласно информации, в строительство вложено почти 11 миллионов евро. Каждая система имеет номинальную мощность 4,5 мегаватт. По данным полиции доказательств совершения преступления нет. Никто не пострадал в результате обрушения огромного ветряка в лесу недалеко от Хальтерн-ам-Зее. Об обрушении ветротурбины "лесник уведомил полицию и пожарную службу в среду, 29 сентября около 18:30." Ветряк был запущен в эксплуатацию в середине марта 2021 года вместе с соседним такой же конструкции. Как сообщил представитель RAG MI (девелопер компании-эксплуатанта), ветряная турбина была полностью разрушена в результате обрушения. Остался только "пень" из бетонных частей высотой 40 метров. Место обрушения было оцеплено на большой территории. RAG предполагает, что мусор не представляет угрозы для окружающей среды. Далее представитель эксплуатанта оптимистично заявил, что ветряная турбина "работала отлично, пока не рухнула". Сразу после обрушения аналогичный ветряк, который находится на расстоянии около 450 метров, был выведен из эксплуатации по соображениям безопасности. Он не будет работать до тех пор, пока не выяснится причина аварии. По данным муниципальных коммунальных служб, во время обрушения ветряная турбина была подключена к электросети. По данным пожарной команды Хальтерна в момент аварии сильного ветра не было.

Производитель - гамбургская компания Nordex - прекратила эксплуатацию в общей сложности ещё 17 систем и ввод в эксплуатацию двух систем той же конфигурации, что и в Хальтерне, по соображениям безопасности. По информации официального представителя Nordex, эти ВЭС расположены в Германии. Еще три ВЭС находятся в стадии строительства. Но уже решено, что в этой конфигурации больше не будет построено никаких систем. Он также сообщил, что данный "коллапс" был первым случаем, когда такая модель башни сломалась. Всего по миру было продано 1222 такие системы.

Трудоемкость – 25 мин.

Модель павильона метро

Пешеходная библиотека является высокоуровневой библиотекой для моделирования движения пешеходов в физическом пространстве. Она позволяет моделировать здания, в которых движутся пешеходы (станции метро, стадионы, музеи), улицы, парки отдыха и т.д. В моделях, созданных с помощью Пешеходной библиотеки, пешеходы движутся в непрерывном пространстве, реагируя на различные виды препятствий в виде стен и других пешеходов.

Пешеходная библиотека AnyLogic позволяет собирать статистику работы моделируемой системы, и наглядно визуализировать моделируемый процесс с помощью анимации. Вы можете отслеживать плотность пешеходов в различных областях модели для того, чтобы убедиться в том, что система сможет справиться с потенциальным ростом нагрузки, вычислить время пребывания пешеходов в каких-то определённых участках модели, выявить возможные проблемы, которые могут возникнуть при перепланировке интерьера здания, и т.д.

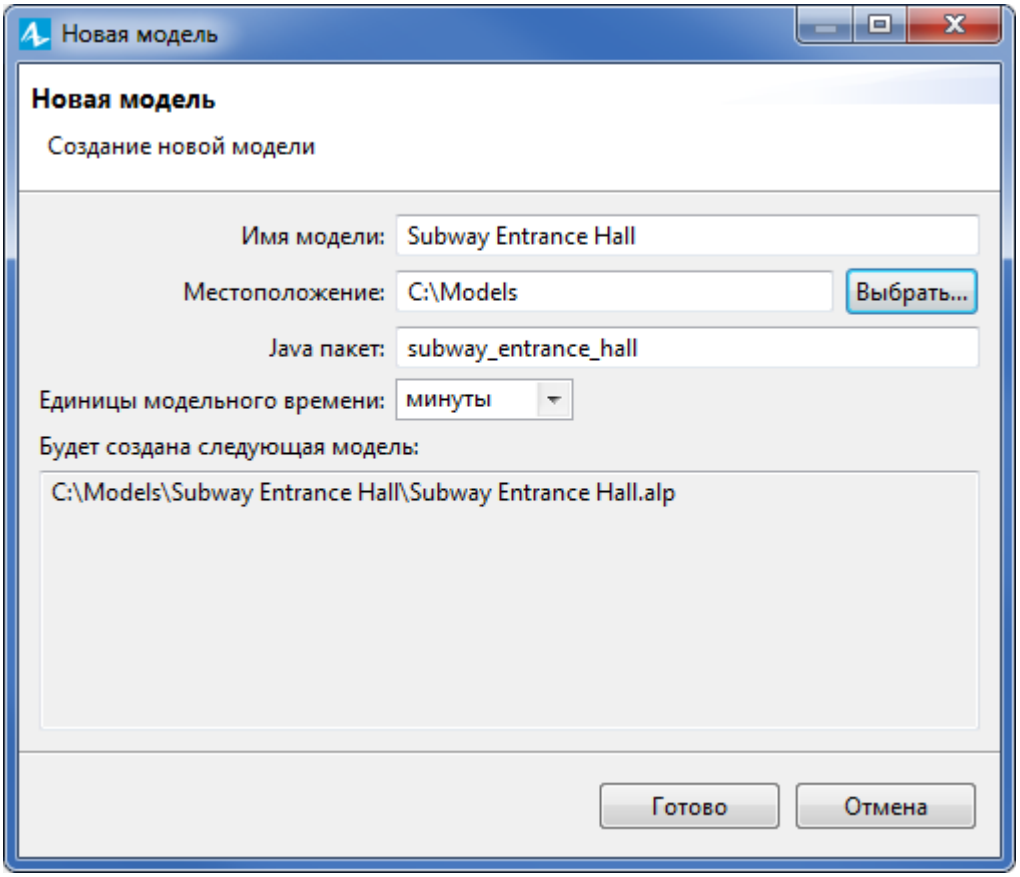
Этот пример научит вас создавать простые модели с помощью Пешеходной библиотеки. Мы создадим простейшую модель наземного павильона метро, с помощью которой наглядно покажем, как моделировать простейшие пассажиропотоки и сервисы.

В этом примере мы создадим простейшую модель, моделирующую движение пассажиров в наземном павильоне метро. Перед тем, как пройти к поездам метро, пассажиры проходят через турникеты, проверяющие наличие билетов. Те пассажиры, которые не купили билеты заранее, должны будут вначале приобрести их в находящейся в павильоне билетной кассе, и только потом они смогут пройти к поездам. Эта модель продемонстрирует, как промоделировать поток пешеходов и простейшие сервисы из Пешеходной библиотеке AnyLogic.

Фаза 1. Моделирование простого потока пассажиров

Вначале создадим простую модель потока людей, двигающихся внутри нашего здания.

Создадим новую модель. Для этого щелкните мышью по кнопке панели инструментов **Создать**  . Появится окно **Мастера создания модели**.



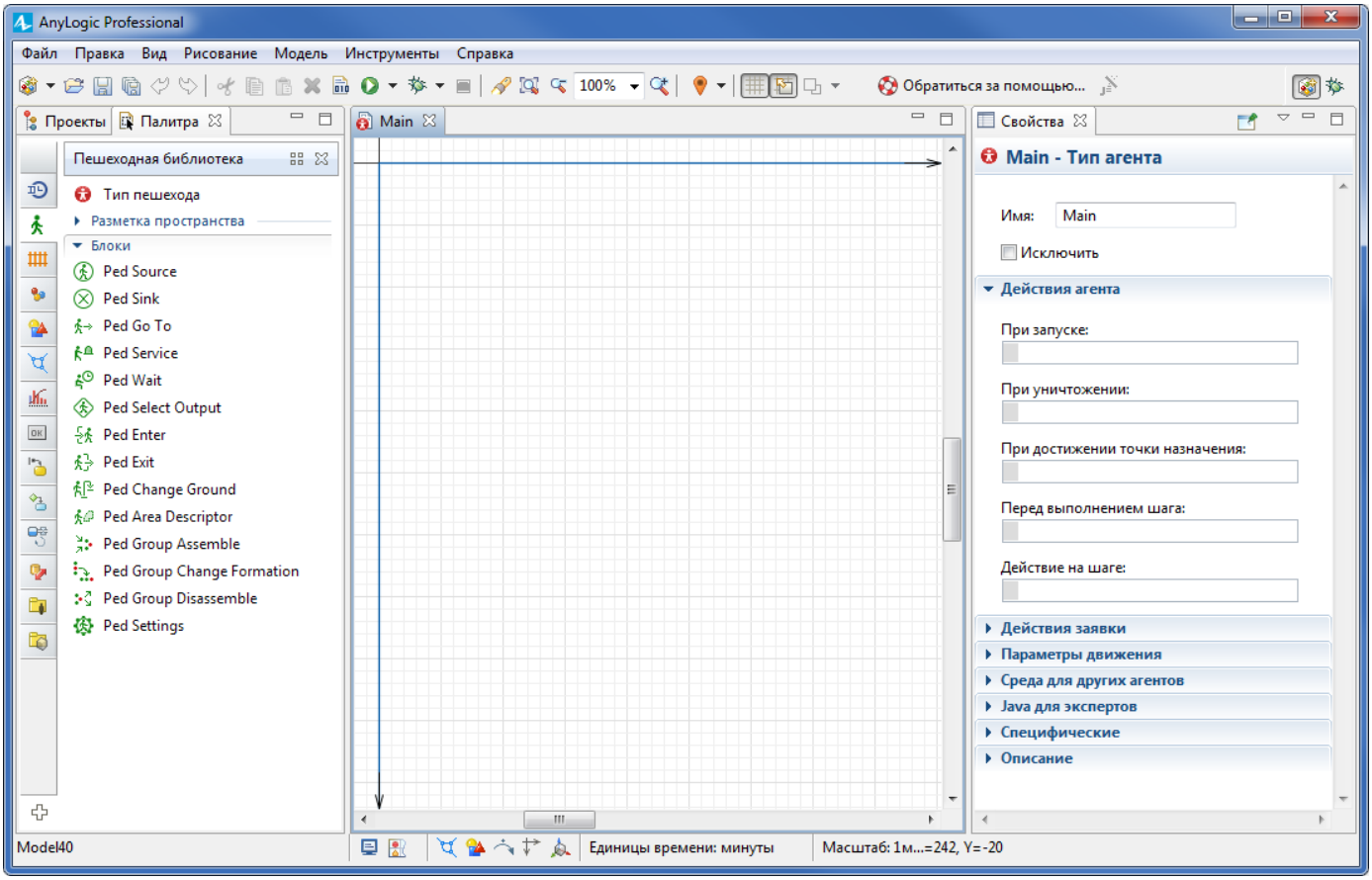
Задайте имя новой модели. В поле **Имя модели** введите Subway Entrance Hall. Укажите каталог, в котором будут сохранены файлы модели. Выберите каталог с помощью диалога навигации по файловой системе, открывающегося по нажатию на кнопку **Выбрать**, или введите путь к каталогу в поле **Местоположение**.

Выберите **минуты** в качестве **Единиц модельного времени**.

Щёлкните **Готово**, чтобы закончить создание модели.

Мы создали новую модель. В ней уже имеется один тип агента Main и эксперимент Simulation. Агенты - это главные строительные блоки модели AnyLogic. В нашем случае агент Main послужит местом, где мы зададим всю логику модели: здесь мы расположим чертеж павильона и зададим диаграмму процесса пассажиропотока.

В центре рабочей области находится графический редактор диаграммы типа агента Main.



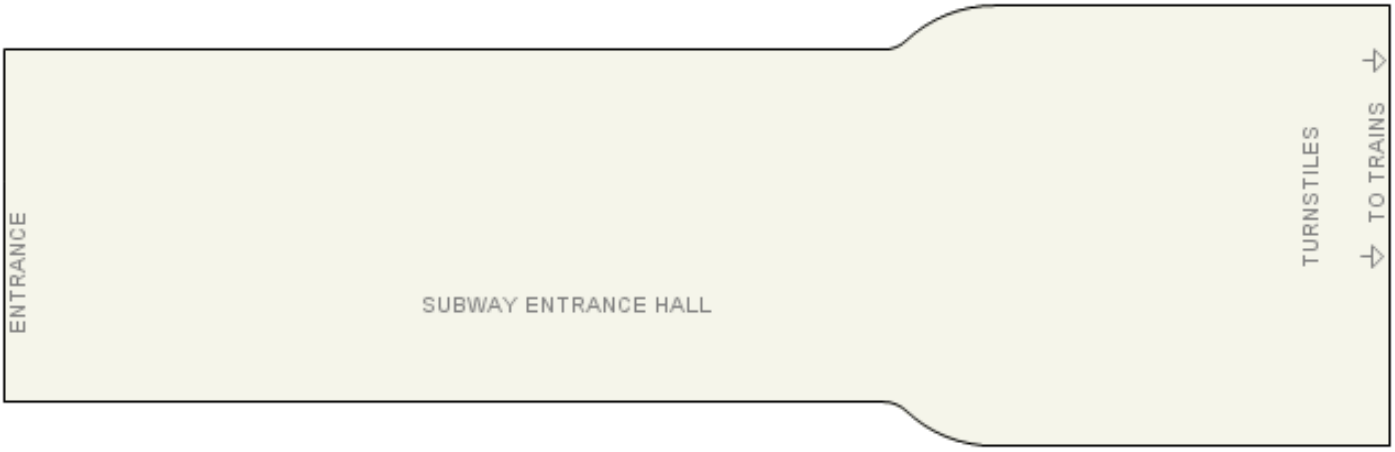
В левой части рабочей области находятся панель **Проекты** и панель **Палитра**. Панель **Проекты** обеспечивает лёгкую навигацию по элементам моделей, открытых в текущий момент времени. Поскольку модель организована иерархически, то она отображается в виде дерева. Панель **Палитра** содержит разделенные по палитрам элементы, которые могут быть добавлены на диаграмму типа агента или эксперимента.

В правой рабочей области будет отображаться панель **Свойства**. Панель **Свойства** используется для просмотра и изменения свойств выбранного в данный момент элемента (или элементов) модели. Когда вы выделяете какой-либо элемент, например, в панели **Проекты** или графическом редакторе, панель **Свойства** показывает свойства выбранного элемента.

Добавление чертежа моделируемого здания

При создании пешеходной модели вначале обычно добавляется рисунок - план моделируемого пространства (помещения, здания). Затем поверх стен на этом плане рисуются стены (с помощью специальных элементов разметки пространства AnyLogic), и затем создается диаграмма процесса: как пешеходы перемещаются внутри здания.

В этой модели мы будем использовать следующий рисунок - план павильона метро:



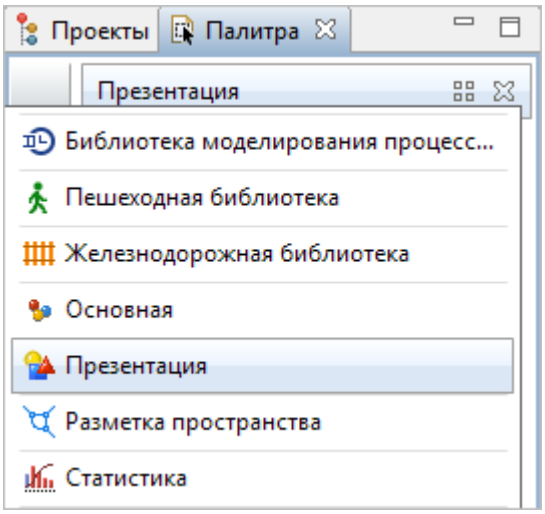
Сохраните рисунок - план на ваш компьютер.

Щелкните правой кнопкой мыши по рисунку выше и выберите **Сохранить изображение как...** из контекстного меню. В открывшемся диалоговом окне выберите, куда вы хотите сохранить файл изображения.

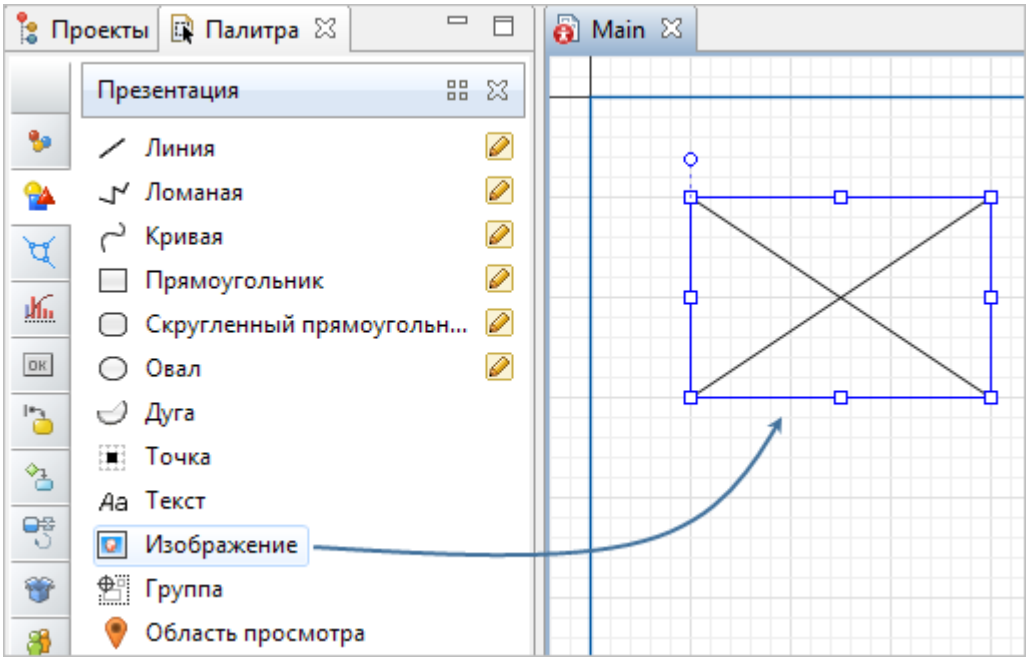
Добавьте рисунок с изображением плана павильона в модель

Вначале откройте палитру **Презентация**. Эта палитра содержит элементы, которые вы можете использовать для рисования анимации модели.

Чтобы открыть какую-либо закладку панели **Палитра**, нужно щелкнуть мышью по соответствующей иконке на вертикальной панели слева от палитры. Пока вы привыкаете к иконкам палитры, вы можете навести указатель мыши на панель и подождать, пока появится всплывающее окно, в котором вы увидите названия палитр.

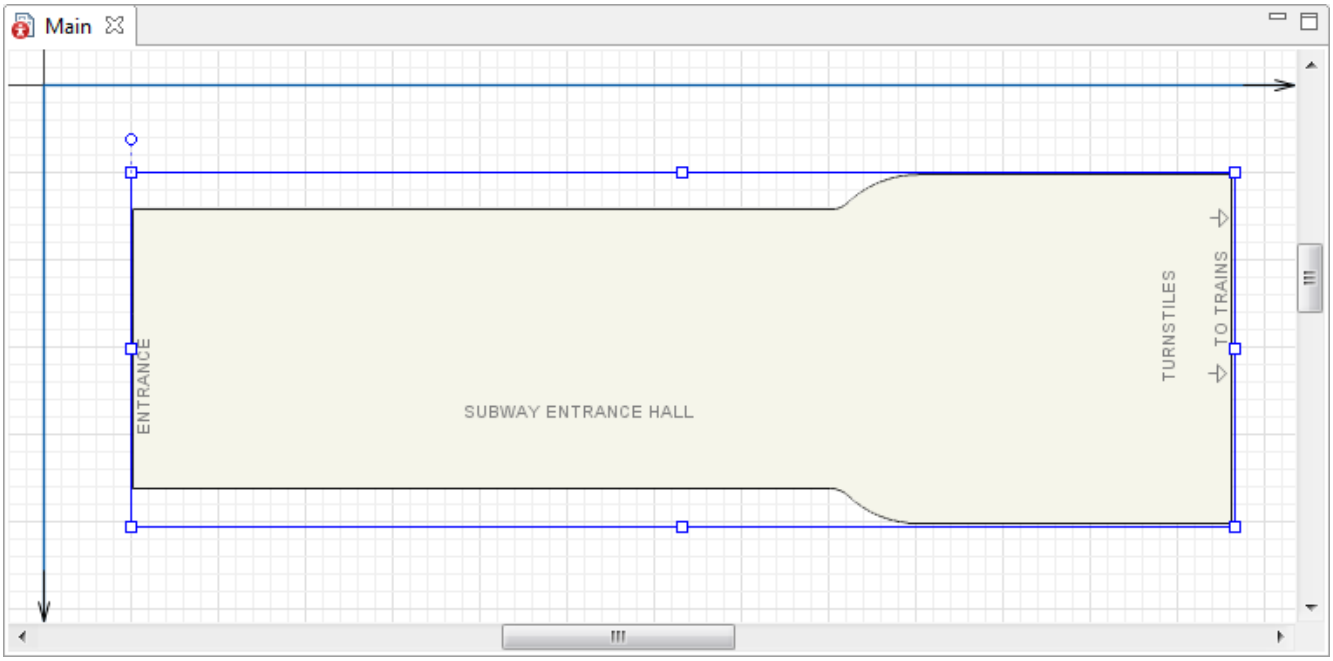


Итак, вы открыли палитру **Презентация**. Перетащите элемент **Изображение** из палитры **Презентация** на графическую диаграмму Main:

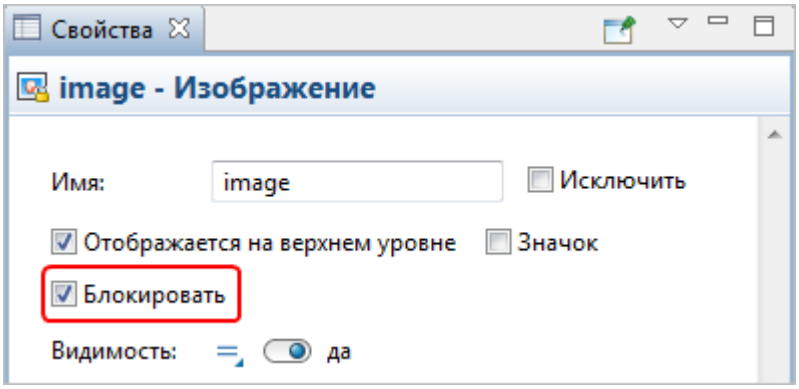


Теперь нам необходимо выбрать рисунок для отображения. Диалог для выбора файла появится автоматически. Откройте папку, в которую вы только что сохранили файл изображения, и выберите его.

Изображение будет выглядеть в графическом редакторе следующим образом:



Заблокируйте изображение, установив флажок **Блокировать** в панели **Свойства**. Вы не сможете выбрать заблокированную фигуру в графическом редакторе до тех пор, пока вы не снимете с неё блокировку. Мы делаем так потому, что мы будем рисовать другие фигуры поверх этого изображения, и поэтому хотим исключить возможность случайного редактирования изображения при рисовании этих фигур.

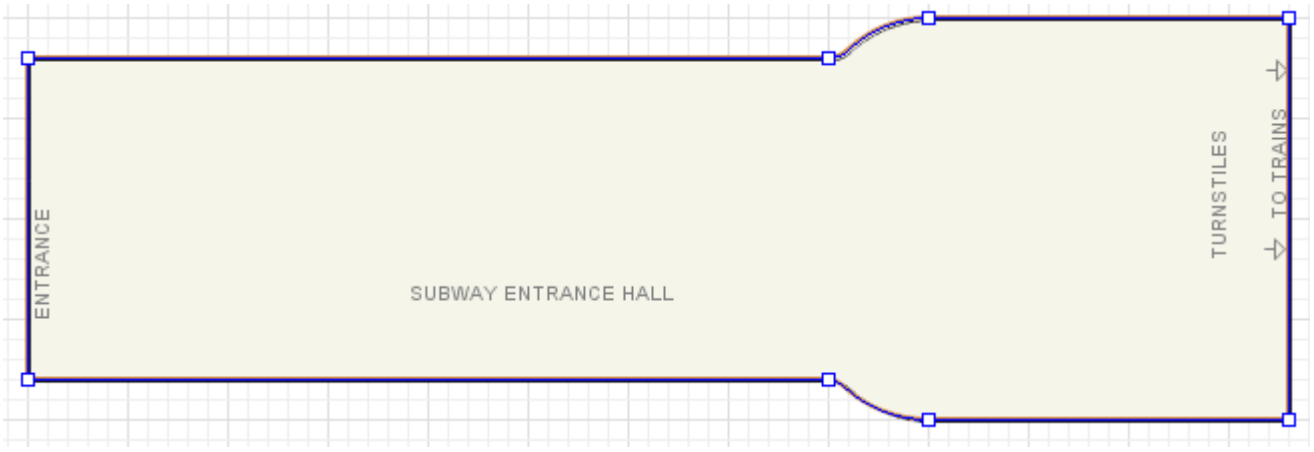


Только некоторые зоны отмечены на этом рисунке. Нам хотелось бы поэкспериментировать с разными диаграммами, и на данный момент нам не известно, где именно располагаются кассы и автоматы по продаже билетов. Поэтому на рисунке отмечены не все области.

Рисование границ здания

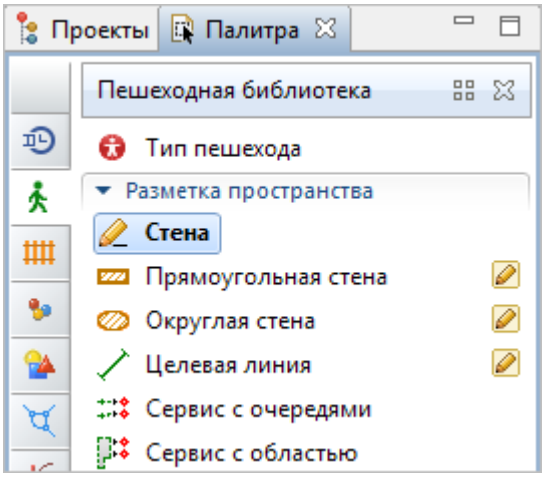
Теперь мы нарисуем на анимации объекты моделируемой среды. Вначале мы нарисуем границу моделируемого нами пространства, играющую роль стен здания.

Откройте палитру **Пешеходная библиотека**. Нарисуйте стены, как показано на рисунке:



Стены рисуются так. Сначала выберите двойным щелчком мыши элемент **Стена** в разделе **Разметка** палитры **Пешеходная библиотека**. При этом его значок должен поменяться на этот: . Это значит, что режим рисования активен и вы можете рисовать стену в графическом редакторе точка за точкой.

Последовательно щелкайте мышью в тех точках диаграммы, куда вы хотите поместить углы стены. Каждый щелчок добавляет новую часть линии стены.



Чтобы добавить кривую линию, щелкните левой кнопкой мыши в точку конца кривой линии и двигайте указатель мыши, удерживая кнопку. Пока вы ведете указатель мыши, видно, как изменяется радиус кривизны. Чтобы нарисовать окружность, двигайте указателем ровно вдоль сетки координат. Отпустите левую кнопку мыши, когда рисунок готов.

Чтобы завершить рисование, добавьте последнюю точку стены двойным щелчком мыши.

Рисование входа и цели движения для потока пешеходов

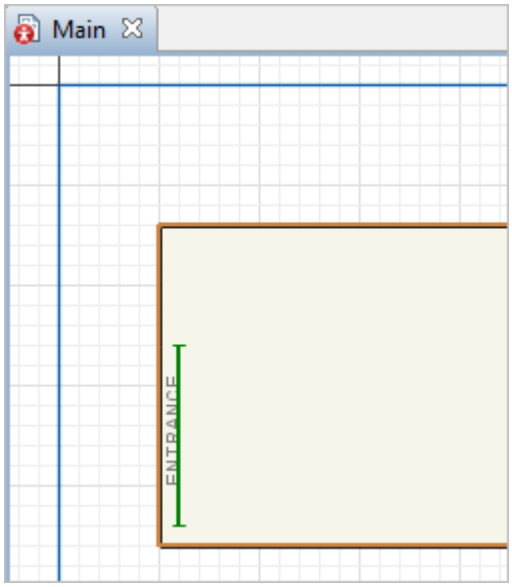
Теперь нужно задать области входа и выхода пешеходов.

Вначале нарисуем область входа - линию, в которой пешеходы будут появляться. Линия входа для пешеходного потока может быть задана специальным элементом разметки **Целевая линия**.

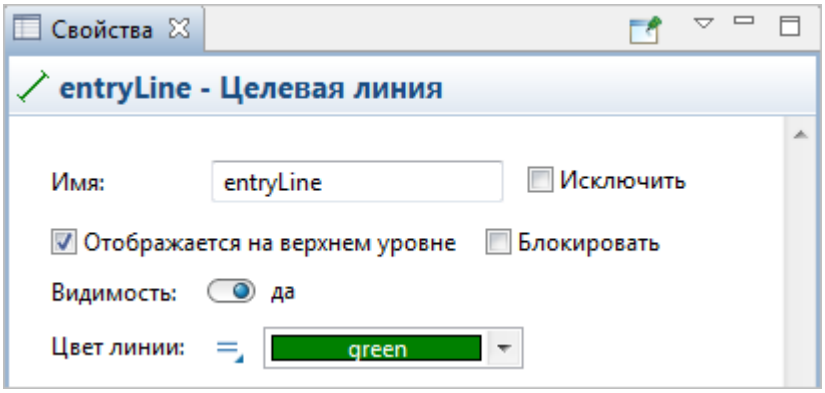
Мы хотим нарисовать линию входа точно там, где на рисунке вход обозначен текстом - ENTRANCE:

Нарисуем линию, где появляются пассажиры. Перетащите элемент **Целевая линия** из секции **Разметка** палитры **Пешеходная библиотека** в графический редактор.

Измените ее размер и расположите точно так, как показано на рисунке:



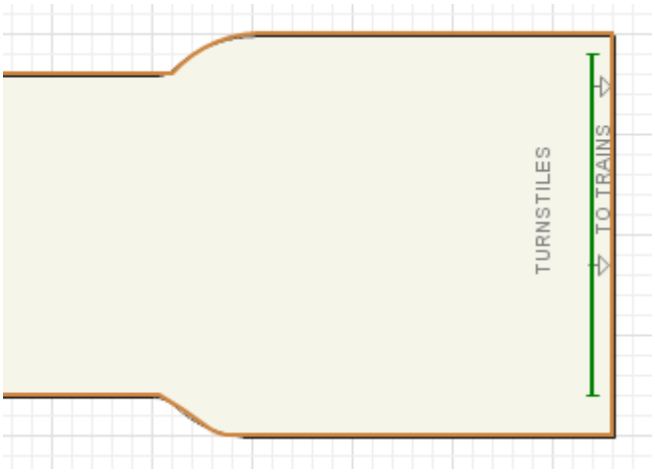
Назовите линию entryLine.



Теперь давайте добавим ещё одну целевую линию, которая определяет место, куда движутся пассажиры после того, как они зашли в павильон метро.

Мы хотим, чтобы они шли к поездам метро, поэтому давайте расположим эту целевую линию у прохода к поездам - над текстом TO TRAINS.

Нарисуйте еще одну целевую линию и расположите ее так, как показано на рисунке. Зайдя в павильон метро, пассажиры будут двигаться к ней, чтобы попасть к поездам метро.



Внимание! Все элементы разметки (целевые линии и пр.) должны находиться внутри стены.

Создание диаграммы, задающей поток пешеходов

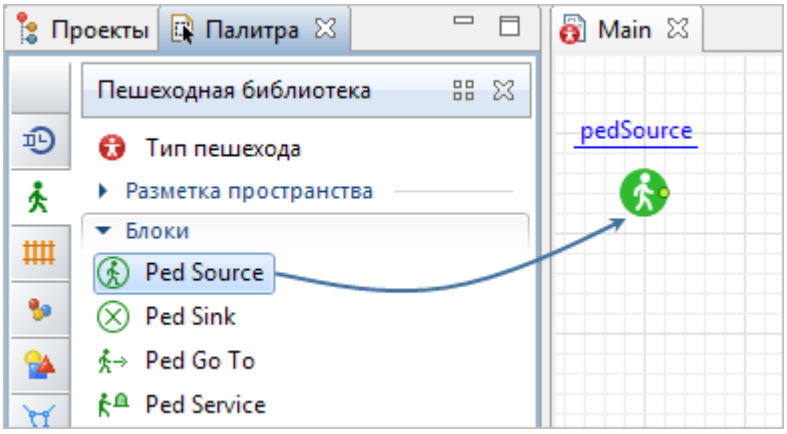
Теперь закончим создание модели, моделирующей простейший пассажиропоток, создав диаграмму моделируемого нами процесса из блоков **Пешеходной библиотеки**.

Мы начнем с самого простого процесса: пассажиры входят на станцию метро (там, где мы нарисовали **entryLine**) и затем двигаются в направлении поездов (к нашей **targetLine**).

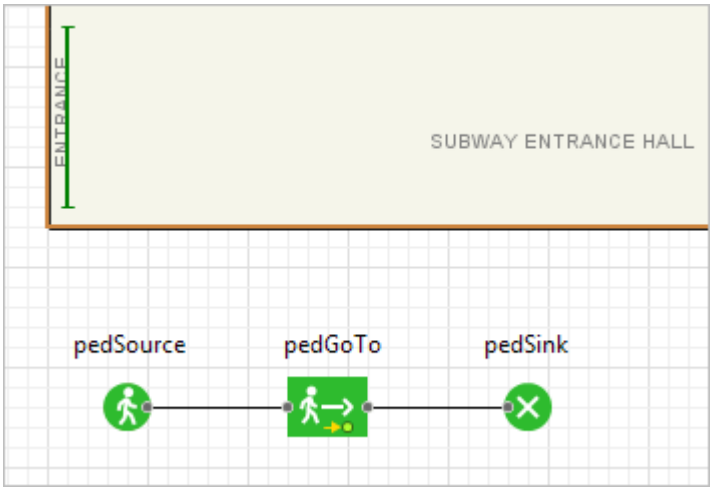
Диаграмма процесса в AnyLogic создается путем добавления объектов библиотеки из палитры на диаграмму типа агентов, соединения их портов и изменения значений свойств объектов в соответствии с требованиями вашей модели.

Создадим диаграмму процесса.

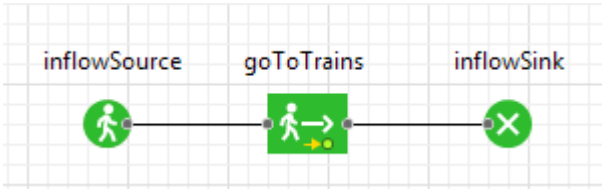
Для этого добавьте объекты **Пешеходной библиотеки** на диаграмму и соедините их так, как показано на рисунке. Чтобы добавить на диаграмму объект Пешеходной библиотеки, нужно открыть в панели **Палитра** палитру этой библиотеки, щелкнув мышью по панели с ее заголовком, а затем перетащить нужный вам объект из палитры на диаграмму типа агента.



Пока вы перетаскиваете блоки и располагаете их друг рядом с другом, можно увидеть, как появляются соединительные линии между блоками. Будьте внимательны, эти линии должны соединять только «порты», находящиеся с правой или левой стороны иконок. Это очень важно, потому что, например, присоединение порта блока **pedSink** к нижнему порту блока **pedGoTo** вызовет ошибку.



Переименуйте блоки. Назовите их inflowSource, goToTrains, inflowSink. Это можно сделать в свойствах блока.



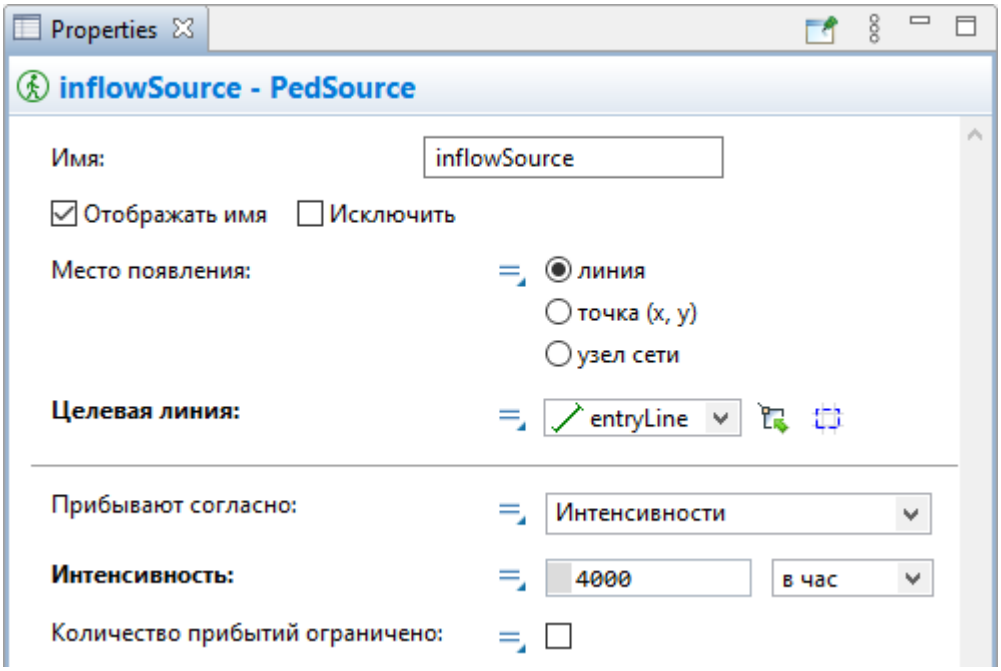
Скажем несколько слов об этих блоках диаграммы.

- Блок **PedSource** создает пешеходов. Обычно он используется в качестве начальной точки диаграммы процесса, формализующей поток пешеходов. В нашем примере он моделирует приход пассажиров в павильон.
- Блок **PedGoTo** моделирует перемещение пешеходов из текущего местоположения в другое (заданное параметром этого объекта). С помощью этого блока мы будем моделировать то, как пассажиры перемещаются от входа в павильон к поездам метро.
- Блок **PedSink** удаляет поступивших в блок пешеходов из моделируемой среды. Обычно блок используется в качестве конечной точки диаграммы процесса.

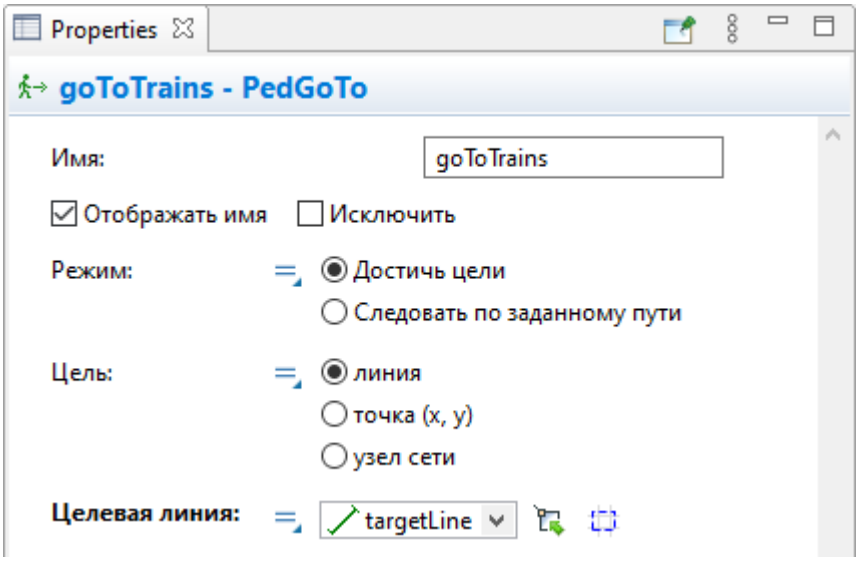
Измените свойства блоков диаграммы

Выделите блок **inflowSource**. В панели **Свойства** задайте место, где появляются пассажиры. Выберите entryLine (название нашей целевой линии, нарисованной ранее у входа) из выпадающего списка **Целевая линия**.

Задайте 4000 в час в параметре **Интенсивность**.



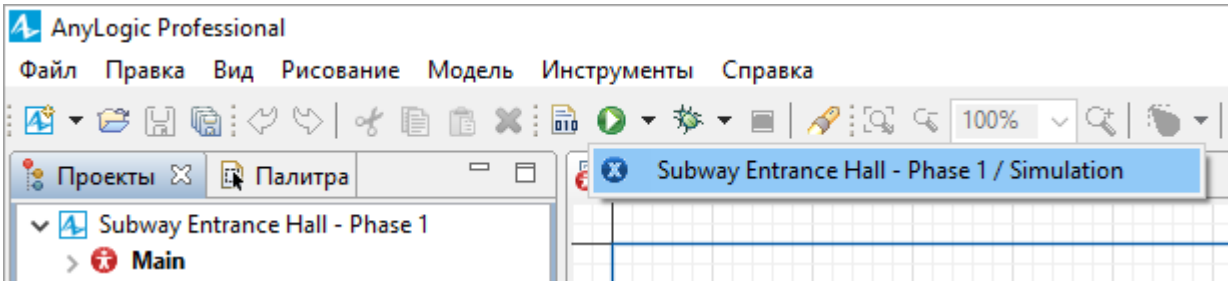
Теперь измените свойства объекта **goToTrains**. Укажите пункт назначения для пассажиров. После того, как пассажиры войдут в здание, они будут двигаться к той цели, которую вы здесь укажете. На данный момент мы хотим, чтобы пассажиры, вошедшие в павильон, сразу двигались к поездам метро. Укажите targetLine (название целевой линии, которую мы нарисовали второй) в списке **Целевая линия**.



Оставьте все свойства объекта **PedSink** установленными по умолчанию.

Давайте запустим нашу модель.

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



Запустив модель, вы увидите окно модели. В нем будет отображена презентация агента верхнего уровня модели. По умолчанию это тип агента Main. Можно увидеть, что пассажиры входят в павильон и движутся по коридору, ведущему к поездам метро.

Фаза 2. Моделирование турникетов

На начальном этапе мы промоделировали простой поток пешеходов: пассажиры входят в здание станции метро и движутся через павильон к поездам.

Теперь же мы хотим, чтобы пассажиры проходили через турникеты для проверки билетов до того, как они проходят на платформу отправления поездов, поэтому давайте добавим турникеты в конце павильона.

Моделирование сервисов

Турникеты являются типичным примером использования сервисов в моделях с пешеходами.

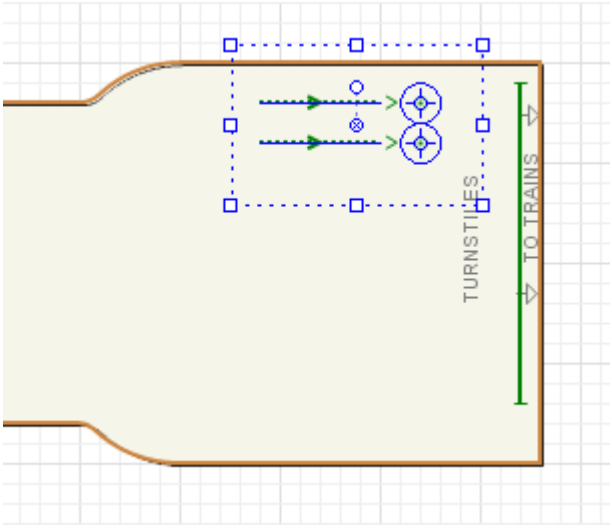
Имеется два типа элементов разметки пространства, которые можете использовать, чтобы добавить сервисы в вашу модель:

Сервис с очередями - используется для того, чтобы задавать сервисы, в которых пешеходы ждут в очереди, пока сервис не будет доступен.
Сервис с областью - используется для того, чтобы задавать сервисы с электронной очередью. В таком случае пешеходы не стоят в очереди, а ждут в расположенной рядом области.

Турникеты обычно моделируются элементом **Сервис с очередями**.

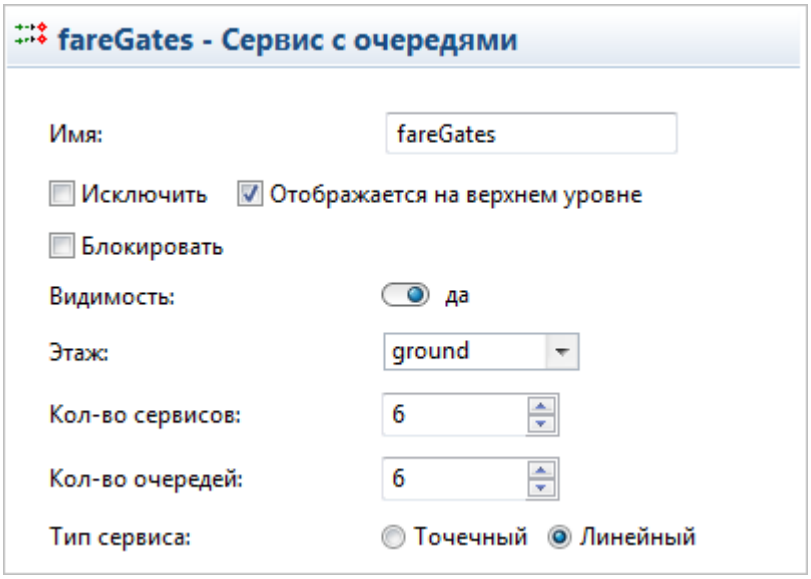
Нарисуем турникеты.

Перетащите элемент **Сервис с очередями** из раздела **Разметка** палитры **Пешеходная Библиотека** в графический редактор. Вы увидите две точки сервиса и две очереди, ведущие к этим точкам. Разместите эти элементы, как показано на рисунке:



Настройте сервисы. Назовите их fareGates. Очевидно, что двух турникетов недостаточно. Увеличьте значение параметра **Количество сервисов** до 6. Соответственно, увеличьте значение параметра **Количество очередей** также до 6.

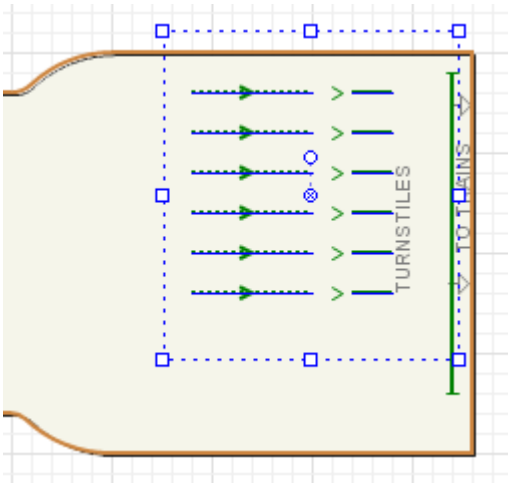
Измените значение свойства **Тип сервиса** с **Точечный** на **Линейный**.



Есть два типа сервисов: **Линейные** и **Точечные**.

- Линейные сервисы используются тогда, когда пешеходы должны пройти вдоль заданной линии сервиса, от начальной точки до конечной. Турникеты обычно моделируются линейным сервисом.
- Точечные сервисы используются тогда, когда для того, чтобы быть обслуженным, пешеход должен просто подойти к любой точке фигуры, задающей соответствующий сервис, и подождать.

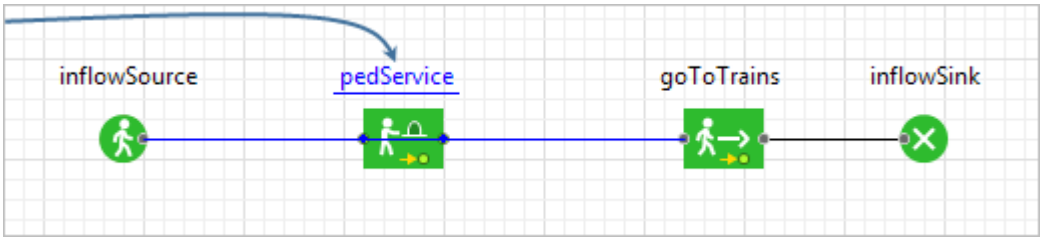
Вы увидите, что сервисные точки стали линиями:



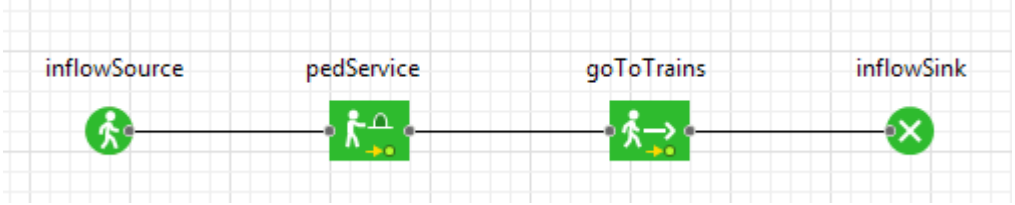
Теперь внесём небольшие изменения в диаграмму процесса.

Освободите место для нового блока на нашей диаграмме. Сдвиньте блоки **goToTrains** и **inflowSink** вправо, как показано на рисунке:

Теперь можно вставить блок **PedService** в нашу диаграмму. Блок **PedService** моделирует то, как пешеходы движутся к сервисам, заданным графически элементом разметки и проходят через сервис. Перетащите блок **PedService** из палитры **Пешеходной библиотеки** в графическую диаграмму, поместите сразу за блоком создания пешеходов. Размещая блок посередине соединителя, мы соединяем порты автоматически (но помните, что нужно присоединить правый порт блока **pedService**, а не нижний!).



Блок появится в диаграмме.

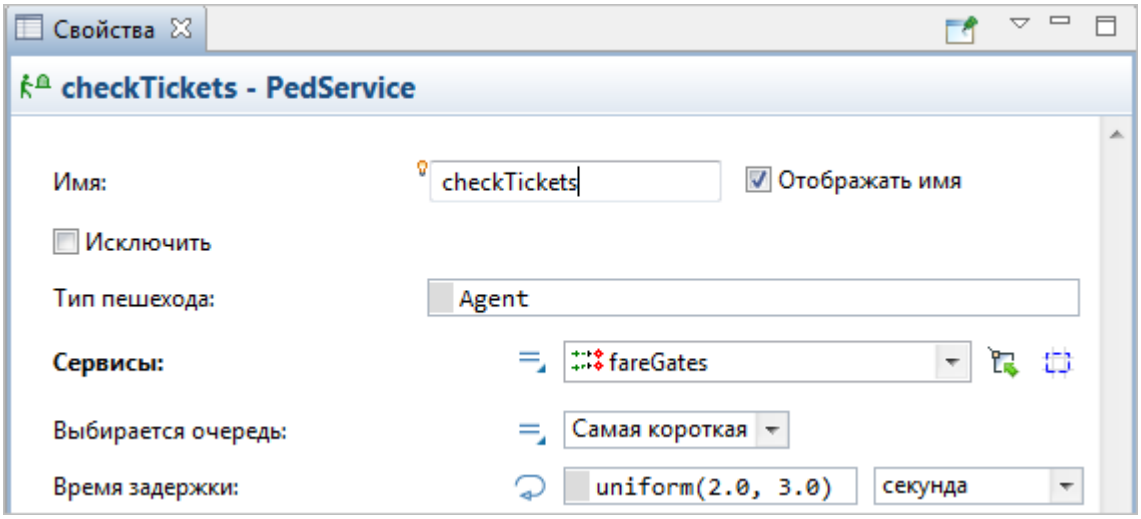


Если вы выделите соединитель щелчком мыши, и его конечные точки в портах будут подсвечиваться светло-зеленым цветом, то это будет означать, что вы успешно соединили порты. Иначе вам придется проверить, обе ли конечные точки соединителя были помещены точно в порты, и если нет, то передвиньте их туда.

Измените блоки диаграммы

Откройте панель **Свойства** блока **pedService**. Измените название блока на checkTickets. Выберите fareGates (название нашего элемента разметки **Сервис с очередями**) в поле **Сервисы**.

В панели видно, что заданное **Время задержки** распределено по равномерному закону с минимальным значением, равным 2 секундам и максимальным, равным 3 секундам. Оставьте это значение, поскольку оно является типичным временем задержки при прохождении турникетов.

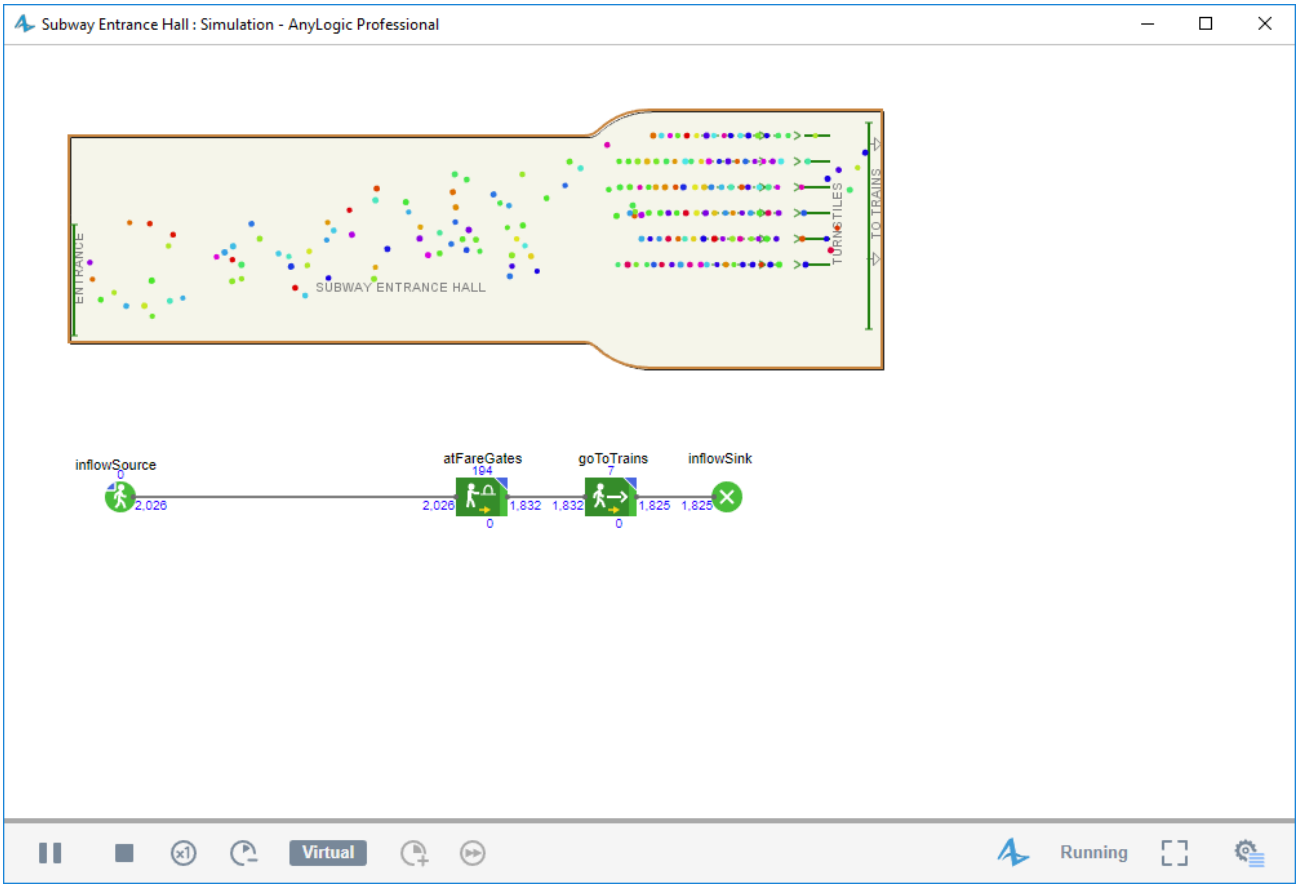


Мы задали новую логику и теперь можем запустить модель и наблюдать за динамикой моделируемого процесса.

Постройте вашу модель с помощью кнопки панели инструментов **Построить модель** . Если в модели есть какие-нибудь ошибки, то построение не будет завершено, и в панель **Ошибки** будет выведена информация об ошибках, обнаруженных в модели. Двойным щелчком мыши по ошибке в этом списке вы можете перейти к предполагаемому месту ошибки, чтобы исправить ее.

После того, как вы исправите все ошибки и успешно построите вашу модель, вы можете ее запустить. Перед запуском модели автоматически обновляется существующая версия файла.

Запустите модель. Вы можете увидеть, что теперь пассажиры проходят через турникеты и перед турникетами быстро образуются длинные очереди. Значит, необходимо увеличить количество турникетов.



Остановите модель и снова откройте свойства элемента сервиса. Увеличьте количество сервисов и очередей до 7.

Снова запустите модель и проследите за положением вещей сейчас. Вам может показаться, что теперь все проходит благополучно. Но, на самом деле, моделируемый процесс исполняется медленно, и, чтобы увидеть всю динамику, мы рекомендуем вам увеличить скорость исполнения модели, чтобы понять, будут ли расти очереди.

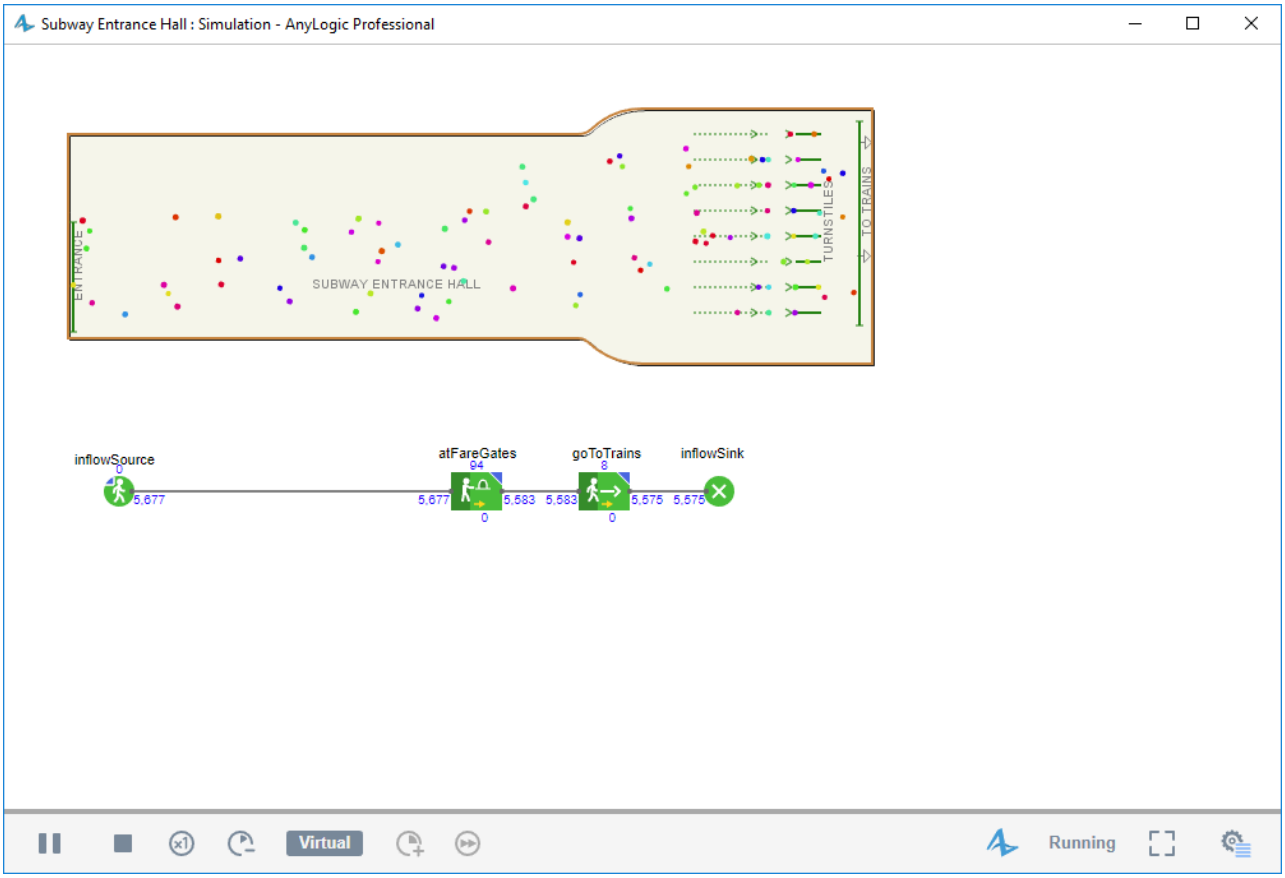
Вы можете изменить скорость выполнения модели с помощью кнопок панели инструментов **Замедлить** и **Ускорить** .

На данный момент, лучше всего переключиться в режим виртуального времени, чтобы модель выполнялась на максимально возможной скорости и вы смогли быстро промоделировать работу системы за долгий период времени.

Чтобы переключиться в режим виртуального времени, нужно щелкнуть мышью по кнопке панели инструментов .

Очереди все еще растут, не так быстро как раньше, но иногда они могут стать довольно значительными. Семи турникетов до сих пор недостаточно, чтобы обслуживать поток из четырех тысяч пешеходов в час. (Не забывайте, что наши турникеты имеют время задержки, равное двум-трем секундам).

Итак, давайте еще раз увеличим количество турникетов. Измените количество сервисов и количество очередей до 8 и снова запустите модель. Очереди в пределах разумного. Наконец наша конфигурация приемлема: станция успешно справляется с таким потоком пешеходов.



Мы извлекли практическую пользу из нашей модели. Данная модель помогла найти требуемое количество точек сервиса. Вы можете изменять интенсивность пассажиропотока в настройках блока **PedSource** и наилучшим образом смоделировать средства обслуживания согласно нагрузке.

Сохраните изменения в модели, щелкнув кнопку панели инструментов **Сохранить** (продолжайте в дальнейшем сохранять изменения время от времени). Теперь мы можем продолжить разрабатывать нашу модель дальше.

Фаза 3. Отображение карты плотности пешеходов

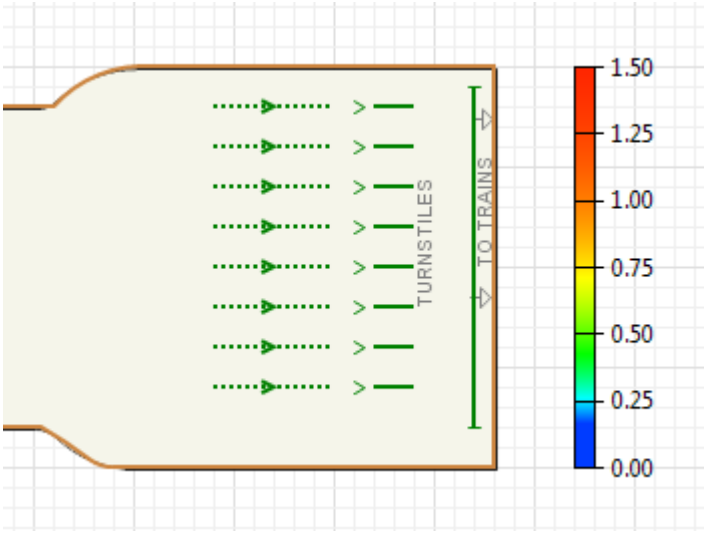
Мы создали модель динамики пешеходного потока в здании павильона метро. Теперь нам хотелось бы получить статистические данные этого потока. Самым значительным инструментом в моделировании пешеходов является **Карта плотности пешеходов**.

Отобразите карту плотности пешеходов.

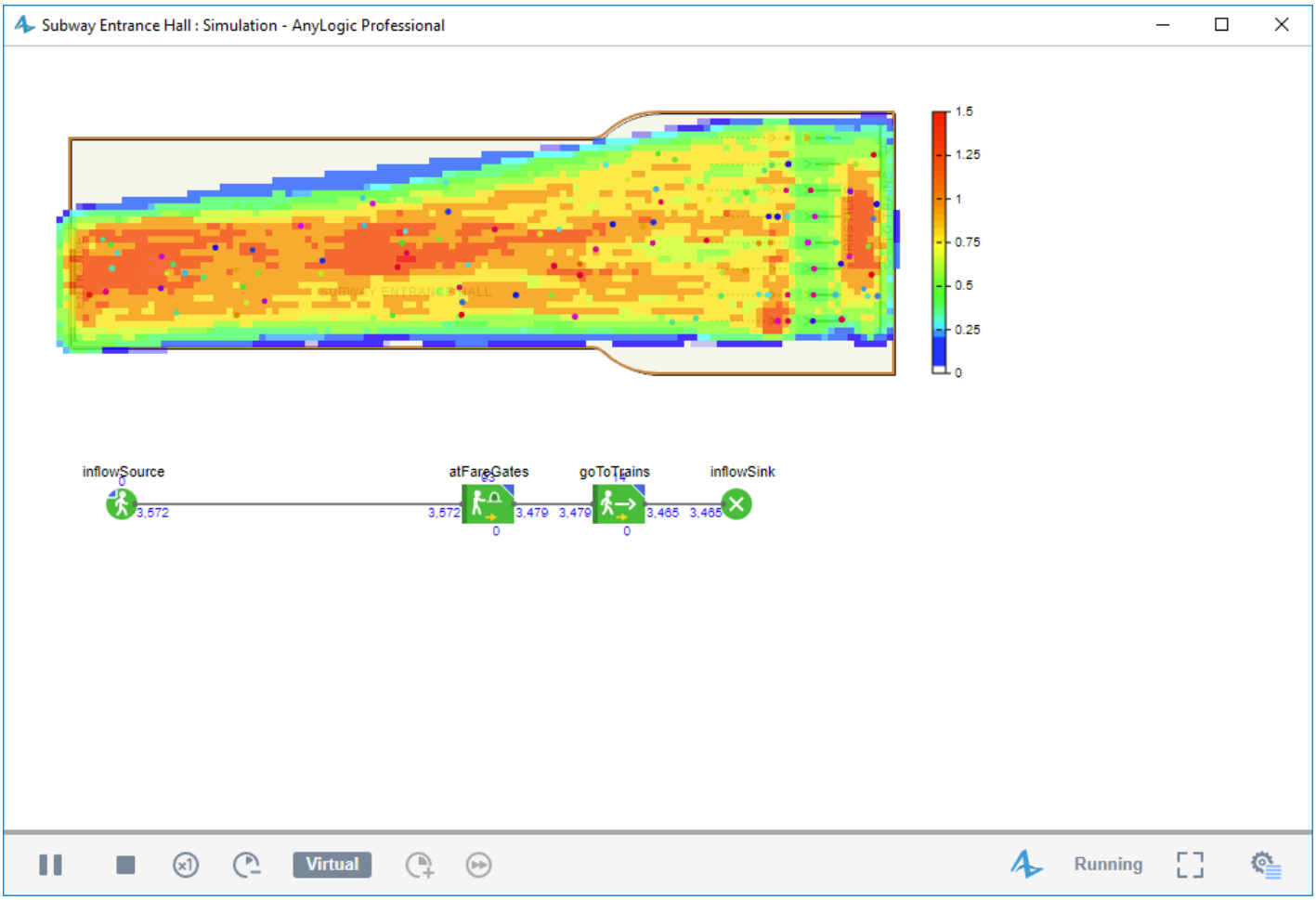
Для этого перетащите элемент **Карта плотности** из секции **Разметка пространства** палитры **Пешеходная библиотека** в графический редактор. В отличие от других объектов библиотек, у этого объекта вместо привычного небольшого значка отображается шкала. На анимации отображается легенда карты. Легенда карты плотности помогает понять, какие цвета соответствуют каким значениям плотности.

Откройте панель **Свойства** этого блока и сбросьте флажок **Отображать имя**.

Выберите значение level из списка **Уровень**.



Теперь вы можете запустить модель и наблюдать за динамикой моделируемого процесса.



Вы увидите, что по мере того, как пешеходы двигаются в моделируемом пространстве, план помещений будет постепенно закрашиваться различными цветами. В каждой точке пространства цвет будет соответствовать измеренной в этой точке плотности пешеходов. Карта плотности постоянно перерисовывается в соответствии с актуальными значениями, при изменении плотности на определенном участке цвет динамически перерисовывается другим цветом.

Карта плотности чаще всего используется для обнаружения участков пространства, на которых значение плотности достигает критических значений. Такие области отображаются на карте плотности красным цветом. По умолчанию значение критической плотности задано равным 1,5 пешехода на квадратный метр. Вы можете изменить это значение в свойстве **Критическая**

плотность (отображается красным), пешеходов/м^2 элемента **Карта плотности**. Синий цвет используется для участков низкой плотности. При нулевой плотности закрашивание соответствующего участка не производится вообще. Приведённая на рисунке шкала информирует нас о том, что, например, жёлтый цвет на карте плотности будет соответствовать плотности 0,75 пешеходов/м ^{2}.

По умолчанию AnyLogic использует логарифмическую цветовую схему. При логарифмической схеме цвет стремительно приближается к "критическому" (красному) только при приближении к зоне критических значений плотности, а при малых значениях остается нейтральным. Вы можете сменить логарифмическую схему на линейную, выбрав **Линейная** в свойстве **Цветовая схема**. В этом случае цвета будут меняться от синего к красному линейно согласно градиенту спектра цветов. При желании вы можете задать и свою собственную цветовую схему любой сложности, выбрав **Другая** в свойстве **Цветовая схема**. В расположенном ниже свойстве **Другой цвет** вы можете написать выражение, которое, в зависимости от значения плотности, будет возвращать тот или иной цвет.

Можно заметить, что даже когда в области совсем нет пешеходов, карта плотности все равно отображается. Это обусловлено там, что карта плотности может либо запоминать и отображать цвета, соответствующие максимальным историческим значениям (когда затухание выключено), либо карта будет соответствовать картине, полученной только за недавнее время (когда затухание включено). Затухание включено по умолчанию. Если опция выбрана, и карта плотности будет затухать, то цвет будет постепенно стремиться к текущему значению плотности (не моментально, а с заданным опозданием). Чтобы включить затухание, установите флажок **Включить затухание** в свойствах элемента **Карта плотности**.

Если вас не устраивает плотность карты тем, что за ней практически не видно плана/чертежа здания, то можете увеличить степень прозрачности карты плотности с помощью ее свойства **Прозрачность**.

Фаза 4. Добавление автоматов продажи билетов

На данный момент все пассажиры в нашей модели входят в павильон метро, затем проходят через турникеты и направляются к поездам. Таким образом, мы предполагаем, что все пассажиры заранее купили себе билеты. Но вряд ли это верно на самом деле. Некоторые люди входят в павильон метро уже с билетами в кармане, но большинство людей покупают билеты, лишь когда заходят в станцию метро.

На станции могут находиться различные виды услуг продажи билетов. Небольшие павильоны метро могут быть оборудованы только автоматами по продаже билетов, а большие и просторные станции могут также иметь билетные кассы.

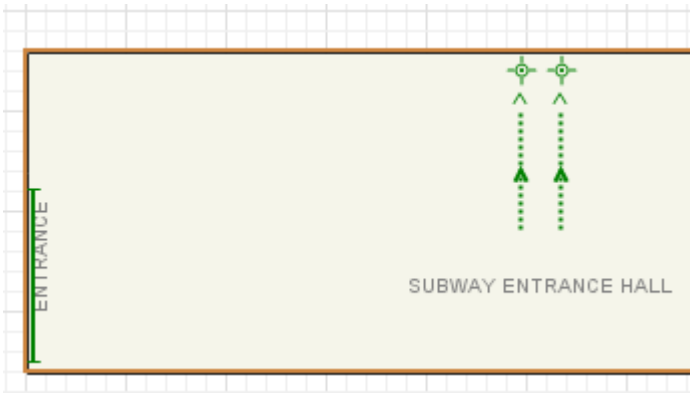
Давайте сначала добавим в нашу модель автоматы продажи билетов. Создавая такую модель, нам необходимо знать количество автоматов, требуемое для того, чтобы успешно обслужить такое количество пассажиров; также, мы сможем найти самое подходящее место расположения автоматов, чтобы минимизировать пересечения потоков пассажиров и образование толп.

Как и турникеты, автоматы продажи билетов логично моделировать элементом **Сервис с очередями**.

Нарисуйте автоматы продажи билетов

Перетащите элемент **Сервис с очередями** из секции **Разметка** палитры **Пешеходная Библиотека** в графический редактор.

Повращайте элементы сервиса и разместите их так, как показано на рисунке:



Откройте страницу свойств сервисов и настройте эту группу сервисов.

В этот раз наши сервисы не линейные, а точечные. Точечные сервисы используются тогда, когда для того, чтобы быть обслуженным, пешеход должен просто подойти к любой точке фигуры, задающей соответствующий сервис и провести там время, заданное как **Время задержки** для этого сервиса. Так что оставьте выбор **Точечный** в свойстве **Тип сервиса**.

Назовите сервисы ticketMachines.

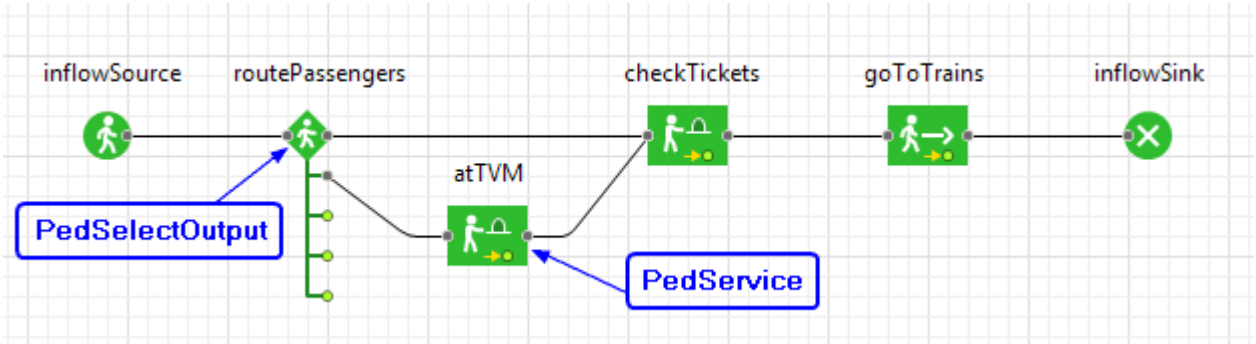
Увеличьте параметр **Количество сервисов** до 7. Соответственно, увеличьте параметр **Количество очередей** также до 7.

Теперь мы хотим направить некоторых из пассажиров сразу к турникетам, а некоторых - на обслуживание у автоматов продажи билетов. Для этого измените диаграмму модели.

Добавьте блок **PedSelectOutput**, чтобы разделить поток пассажиров. Нам нужен этот блок, чтобы перенаправлять пассажиров без билетов к автоматам продажи билетов, а пассажиров с билетами – к турникетам. Блок **PedSelectOutput** является блоком принятия решения Пешеходной библиотеки. Пешеход, вошедший в блок **PedSelectOutput**, будет перенаправляться в один из пяти выходных портов в зависимости от заданных для этих портов коэффициентов предпочтения.

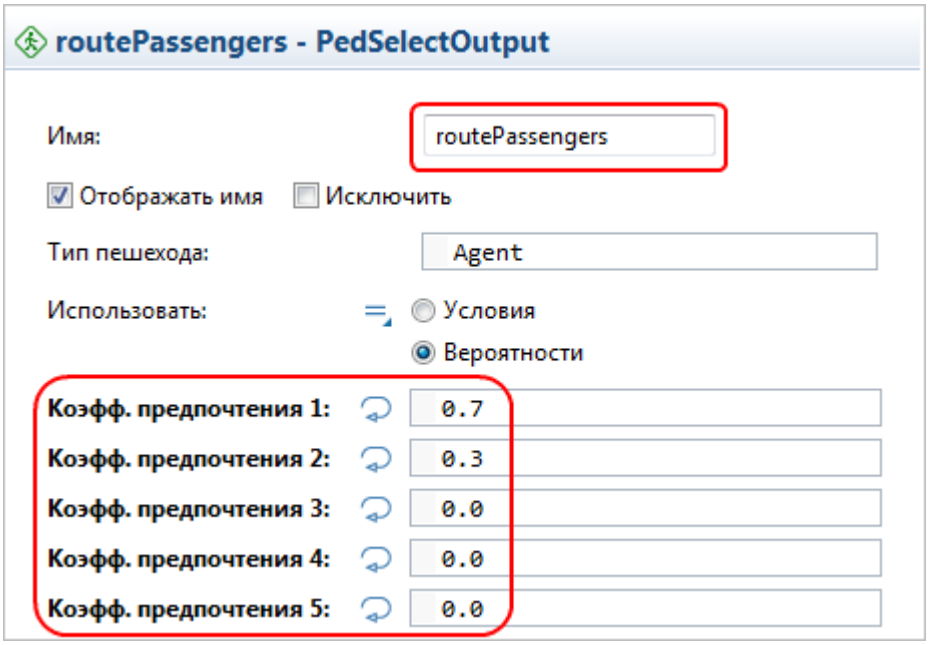
Добавьте еще один блок **PedService**. Этот блок будет моделировать обслуживание пассажиров у автоматов продажи билетов. Поместите его между блоком **PedSelectOutput** и ранее созданным блоком **PedService** (**checkTickets**).

Соедините блоки, как показано на рисунке.



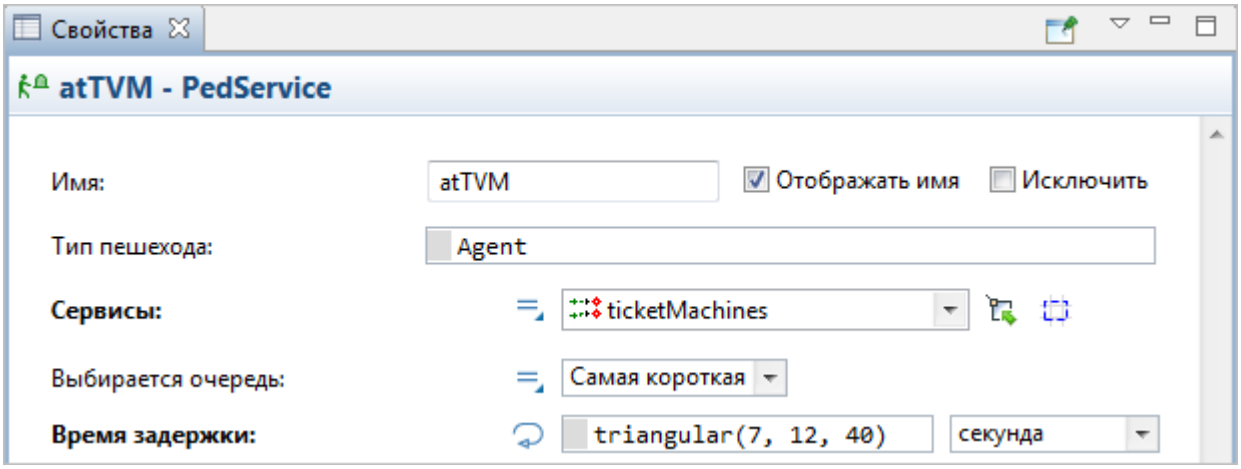
Измените свойства блока **PedSelectOutput**. Назовите его routePassengers. Укажите значение 0.7 в поле **Коэфф. предпочтения 1** (коэффициент для потока, направляющегося напрямую к турникетам) и значение 0.3 в поле **Коэфф. предпочтения 2** (коэффициент для потока пассажиров, направляющихся к автоматам продажи билетов соответственно). На этой диаграмме мы допускаем,

что количество пассажиров, которые уже купили билеты, значительно выше. Укажите в полях **Коэфф. предпочтения 3, 4, 5** значение, равное 0.

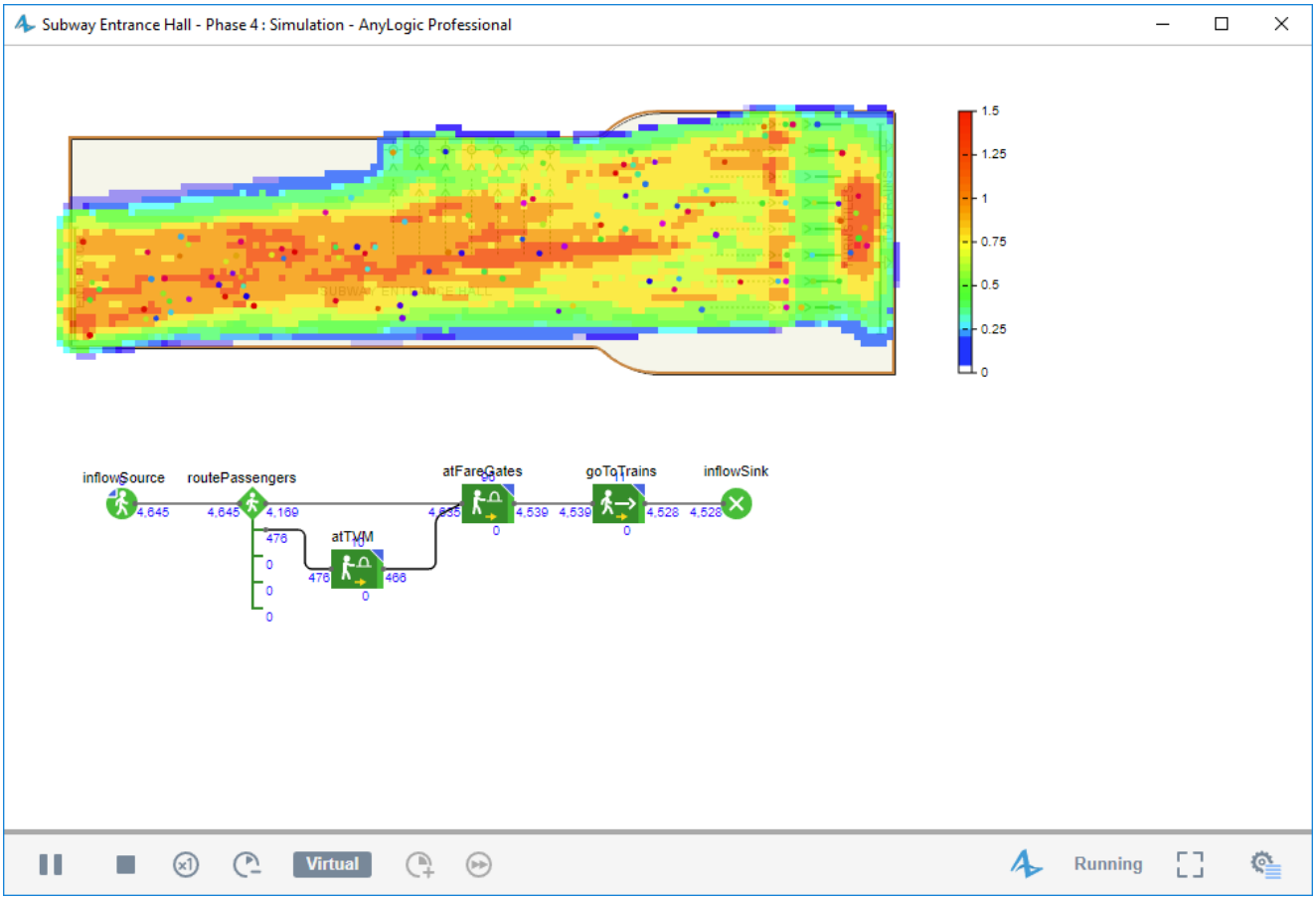


Настройте только что добавленный блок **PedService**. Переименуйте его как atTVM. Выберите ticketMachines (название нашего элемента разметки **Сервис с очередями**) в свойстве **Сервисы**.

Измените параметр **Время задержки**. Введите в поле: **triangular**(7, 12, 40) и выберите секунды в качестве единиц времени. Мы допускаем, что время обслуживания неравнозначно распределено с минимальным значением 7 секунд, средним 12 и максимальным 40 секунд.



Давайте запустим модель и понаблюдаем за ее динамикой. Теперь видно, что некоторые пассажиры перед тем, как пройти к турникетам, сначала подходят к кассовым автоматам, чтобы приобрести билет, а затем следуют к турникетам.


Subway Entrance Hall - Phase 4 : Simulation - AnyLogic Professional
1.5
1.25
1
0.75
0.5
0.25
0
inflowSource
routePassengers
atFareGates
goToTrains
inflowSink
atTVM
4,645
4,645
4,169
476
0
0
0
476
466
0
4,539
4,539
0
4,528
4,528
Virtual
Running