Агентная модель обслуживания ветрогенераторов

Шаг 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.

Вы можете создавать внутри агента переменные, диаграммы состояний, задавать события, потоковые диаграммы системной динамики, а также добавлять внутрь агента объекты библиотек AnyLogic. Вы можете создать в одной модели столько типов агентов, сколько разных типов агентов вам нужно промоделировать.

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 путем вызова методов друг у друга.

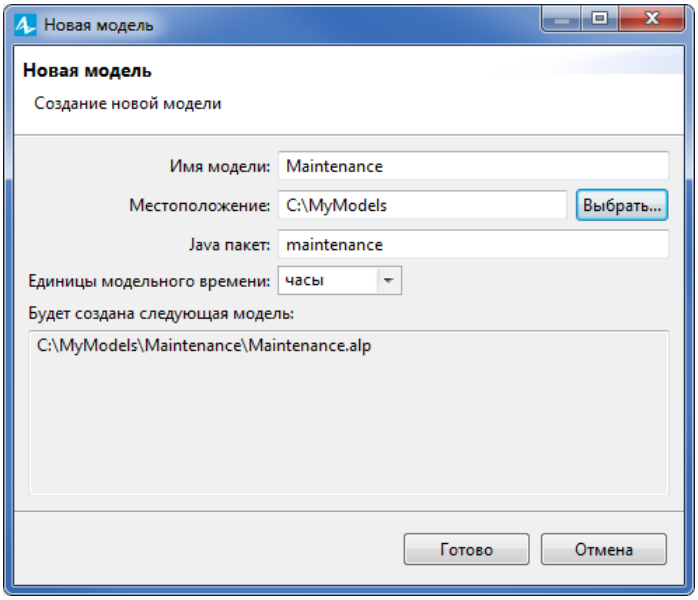
Начальное состояние и поведение агента могут быть реализованы различными способами. Состояние (накопленная история) агента может быть представлено с помощью переменных, либо состояния диаграммы состояний. Поведение может быть либо пассивным (агенты реагируют только на прибытие сообщений или на вызов 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/или диаграммы состояний.

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

Создание модели

Создайте новую модель

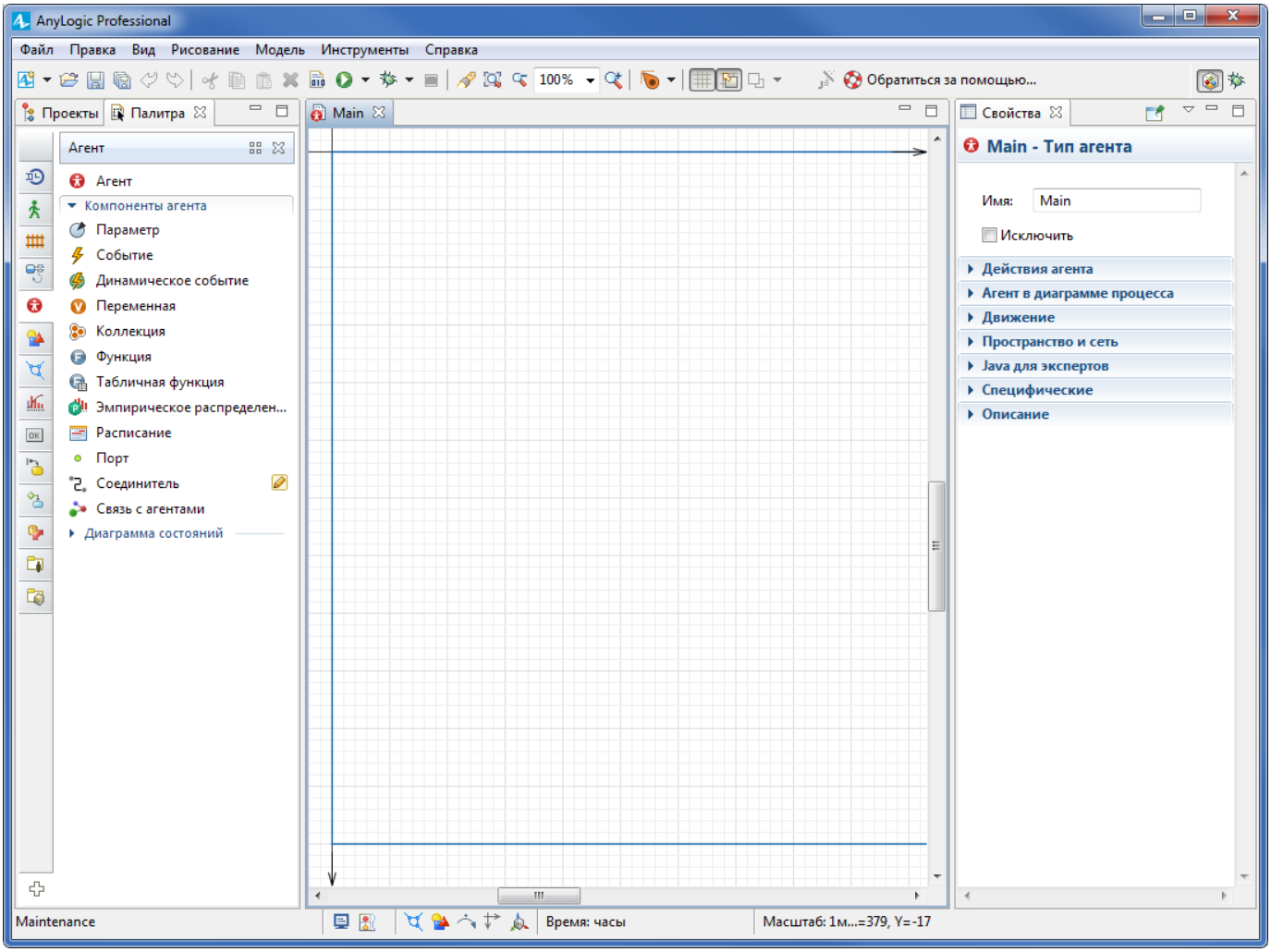
1. Щелкните по кнопке **Создать** на панели управления. Откроется диалоговое окно **Новая модель**.
2. Задайте имя модели. Введите `Maintenance` в поле **Имя модели**.



3. Укажите местоположение, где вы хотите хранить файлы модели. Используйте кнопку **Выбрать...**, чтобы выбрать нужную папку, или введите путь к папке в поле **Местоположение**.
4. В качестве **Единиц модельного времени** выберите *часы*.
5. 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 задает 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**, объединенную 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.

Вы можете открывать и закрывать конкретные панели с помощью меню **Вид**.

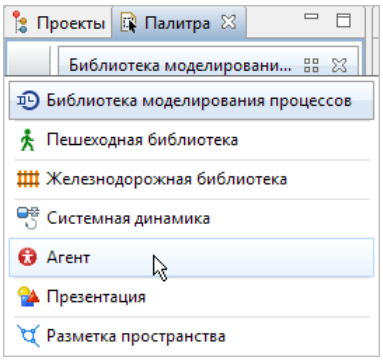
Теперь начнем разрабатывать нашу модель.

Создание агентов

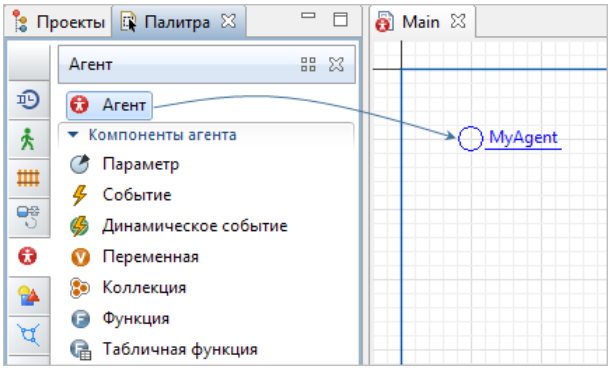
Вначале, создадим одного агента - сервисный центр, затем пустую популяцию для обслуживающего транспорта, три популяции для моделирования турбин, грузовиков и вертолетов, а также тип агента, представляющего запросы на обслуживание. Далее мы сможем задать процессы внутри типов агентов - на их диаграммах.

Добавьте сервисный центр

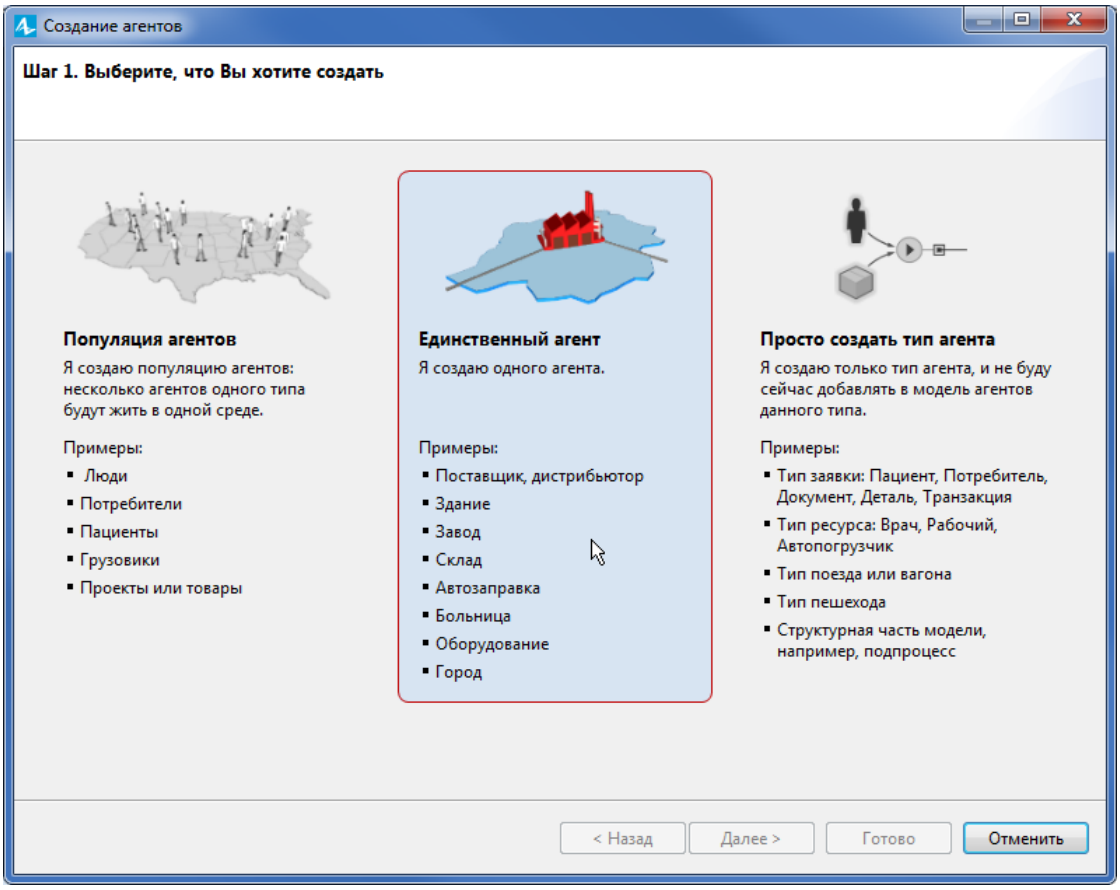
1. В панели **Палитра**, наведите мышь на вертикальную полосу навигации и выберите палитру **Агент**.



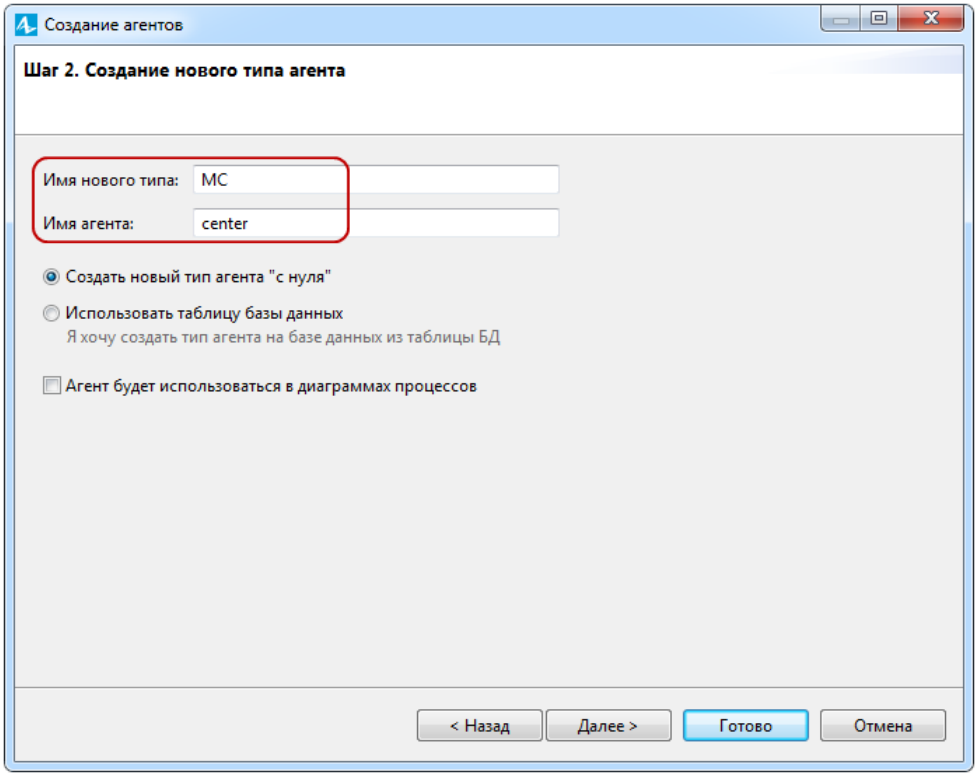
2. Перетащите элемент **Агент** из палитры на диаграмму типа агента `Main`. Окно мастера **Создание агентов** откроется автоматически.



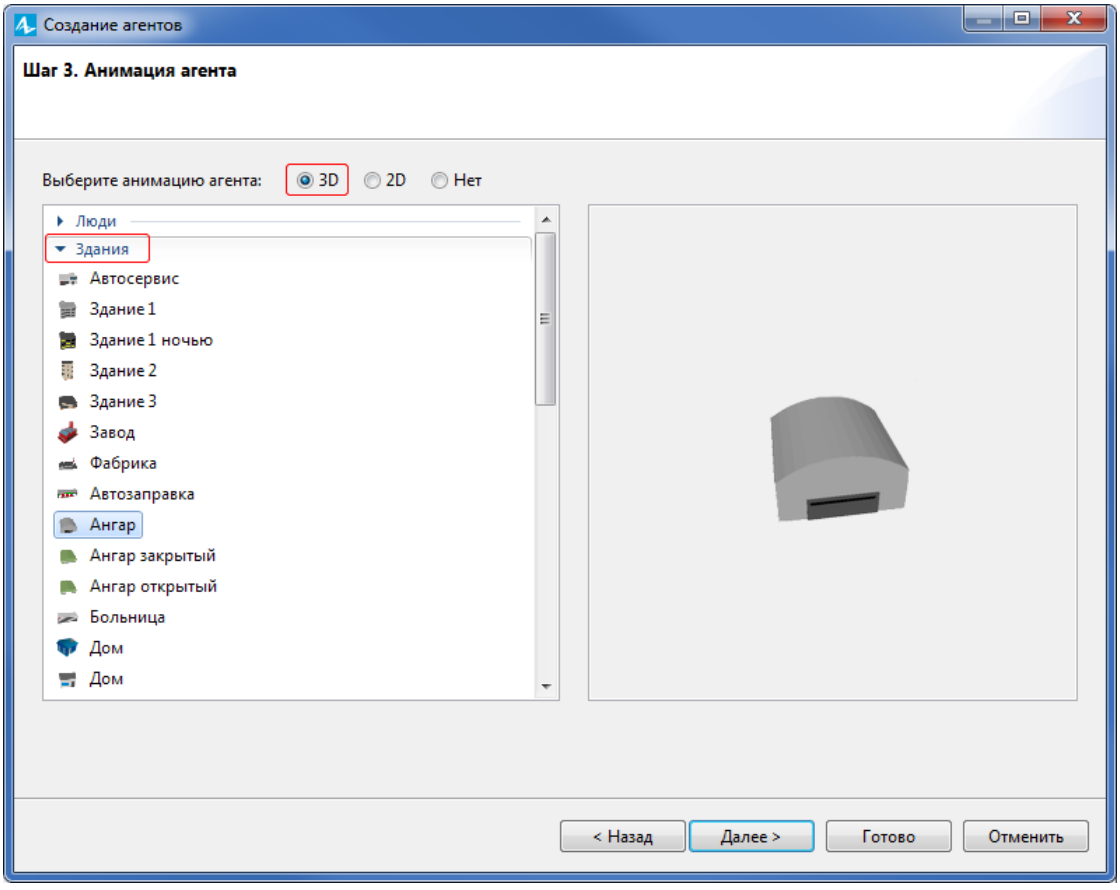
3. Выберите опцию **Единственный агент** на первом шаге мастера. Нам нужно создать только один сервисный центр, который будет отправлять транспорт к турбинам по запросам на плановое обслуживание или при авариях.



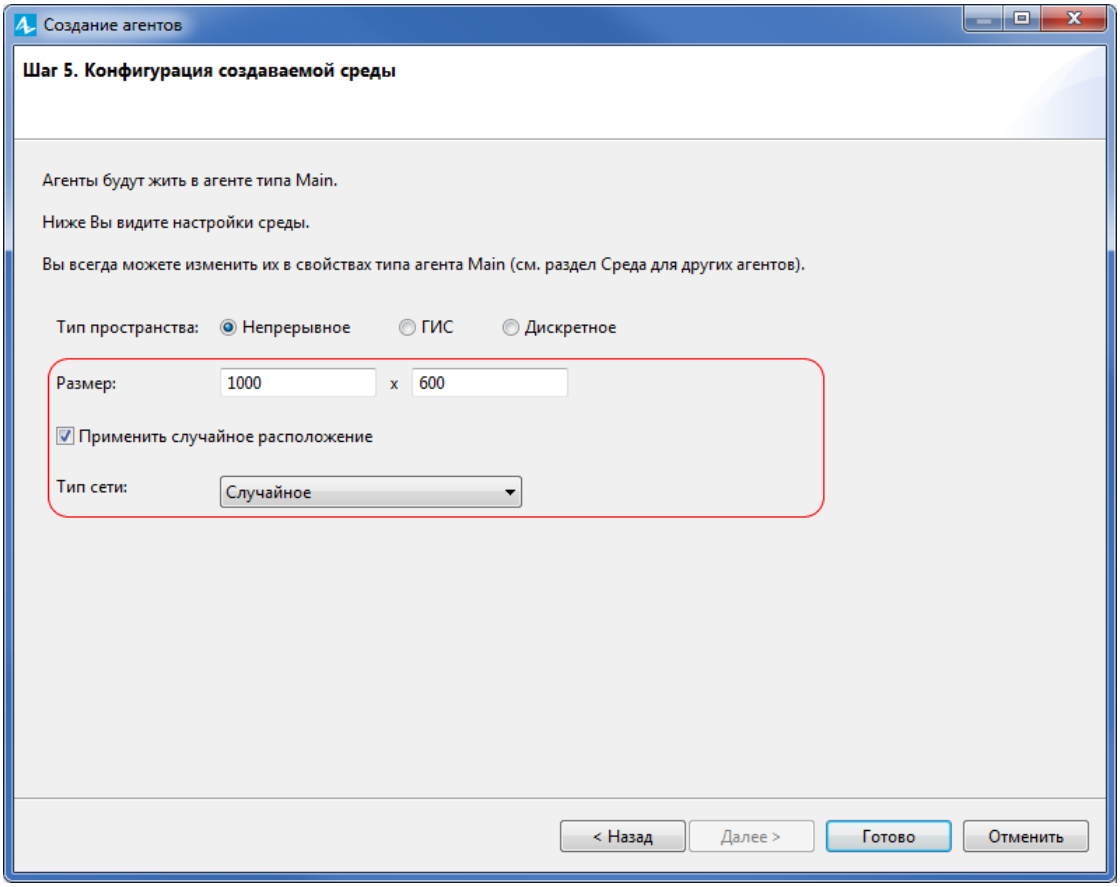
4. Мы не будем использовать данные из базы данных, поэтому оставьте выбранным опцию **Создать новый тип агента "с нуля"**. Задайте **Имя нового типа**: *MC* и имя самого агента *center* в поле ниже. Щелкните **Далее**, чтобы продолжить.



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

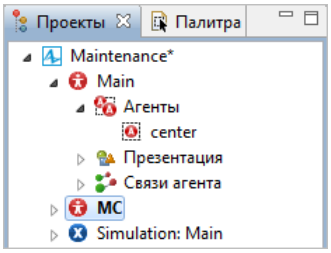


6. Пропустите четвертый шаг по добавлению параметров, просто щелкните **Далее**.
7. Мы хотим, чтобы все наши агенты "жили" в непрерывном пространстве, размерами 1000x600, в сети со случайным расположением агентов. Популяции агентов, которые мы позже так же добавим на диаграмму `Main`, будут жить в той же среде.



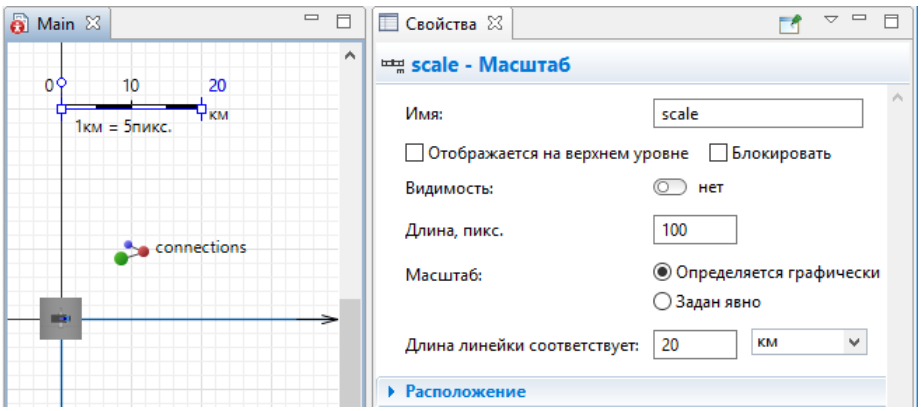
8. Щелкните **Готово**.

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 модеи, а также масштаб отображения этой фигуры на анимации.

1. Чтобы изменить масшта 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 *Определяется графически*, не изменяя длину линейки шкалы. В свойствах элемента, задайте соответствие линейки 20 километрам, тогда на диаграме вы увидите подпись, информирующую, что масштаб данной диаграммы: 1 км = 5 пикселей.

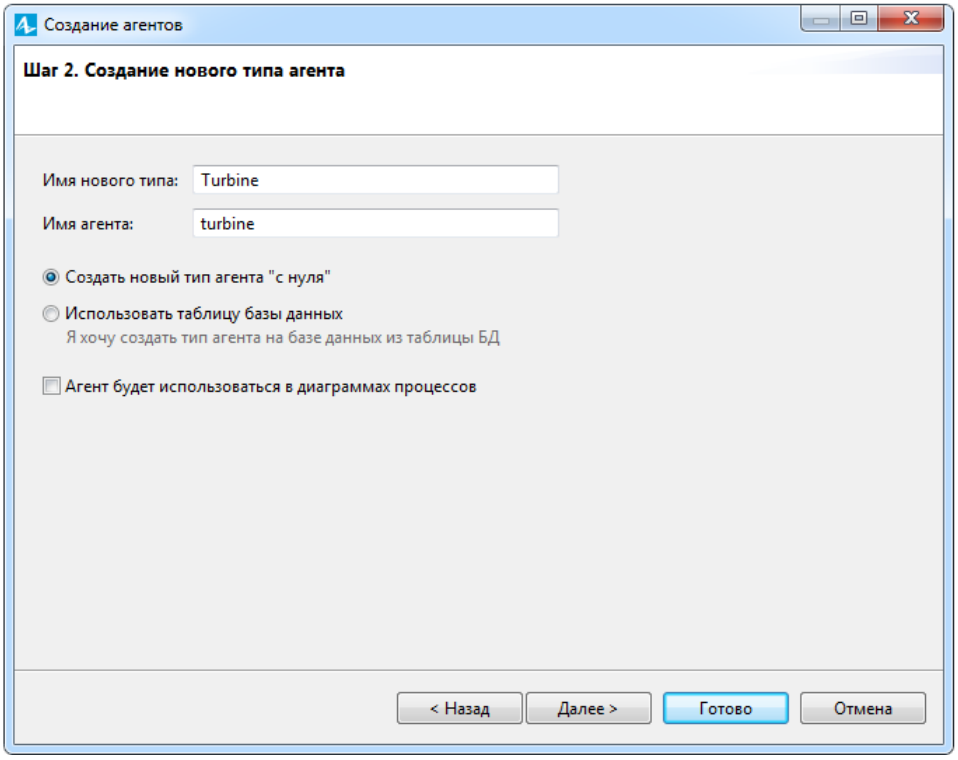


Теперь давайте изменим масштаб фигуры анимации сервисного центра (ангара).

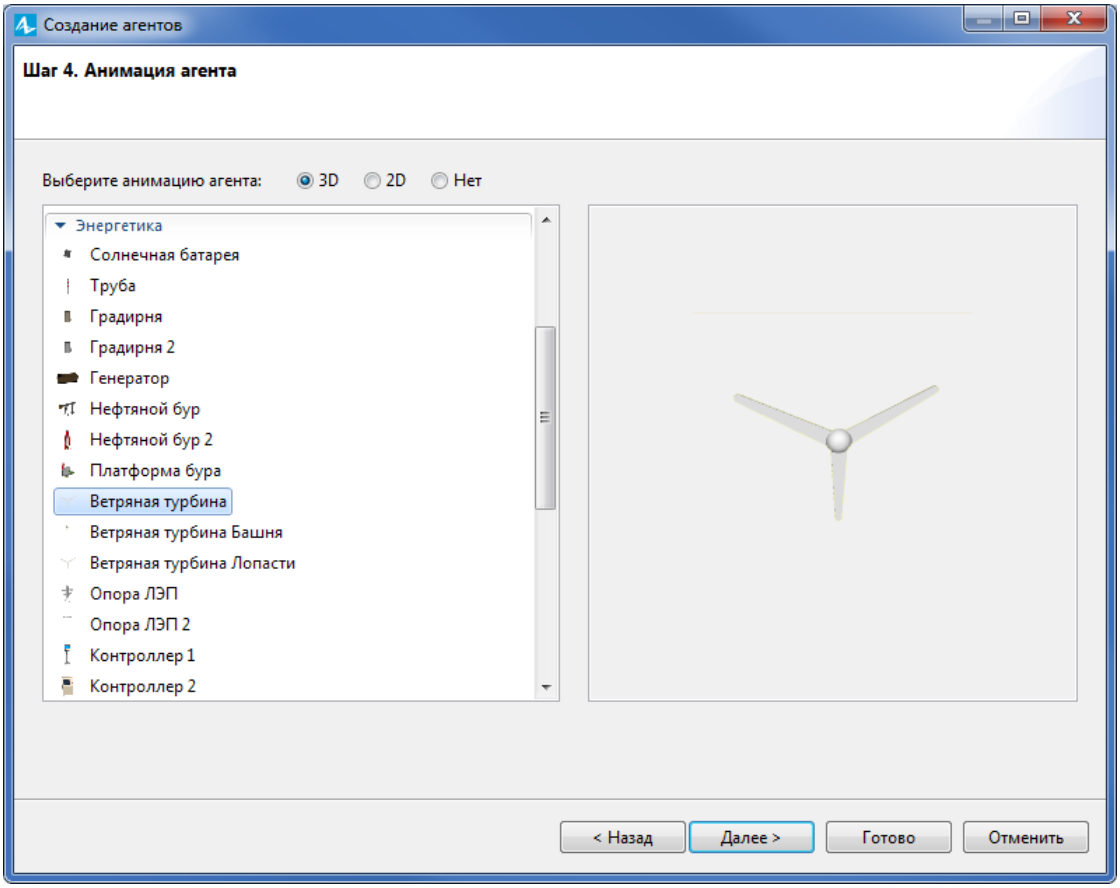
1. Сделайте двойной щелчок мышью по типу агента MC в дереве элементов модели, чтобы открыть его диаграмму. Найдите и выделите линейку масштаба этого типа агента.
2. Очевидно, что при выбранном масштабе фигура ангара будет отображаться на анимации модели такой крошечной, что мы не сможеи ее разглядеть. Поэтому мы зададим для этго типа агента более крупный масштаб, чтобы мы могли легко увидеть его фигуру на сцене анимации нашей модели.
3. Переключите опцию **Масштаб** в режим *Задан явно*. Ниже, задайте масштаб: *10 пикселей в км*.

Добавьте ветряные турбины

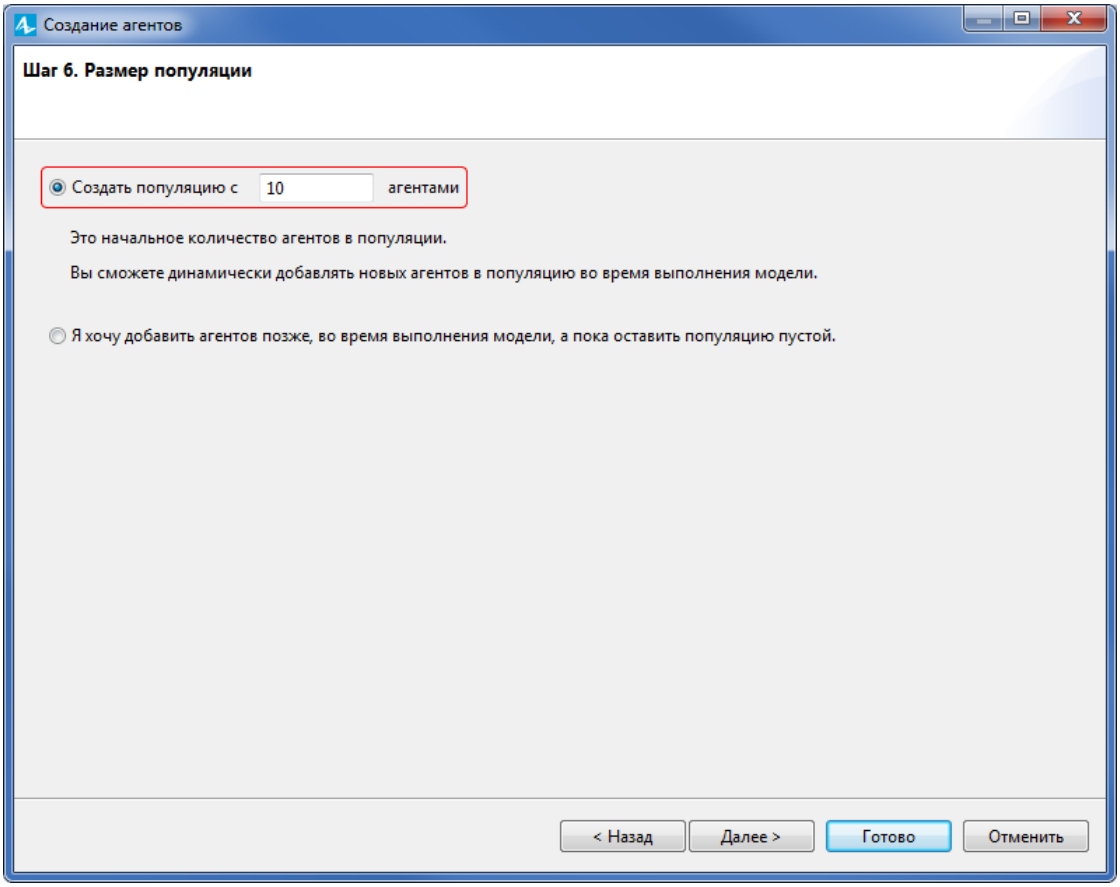
1. Перетащите элемент **Агент** из палитры **Агент** на диаграмму типа агента `Main`.
2. В этот раз выберите опцию **Популяция агентов**. Щелкните **Далее** на шаге 2 (по умолчанию выбрана нужная нам опция **Я хочу создать новый тип агента**). На следующем шаге мастера создания агентов введите имя нового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



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



4. 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.



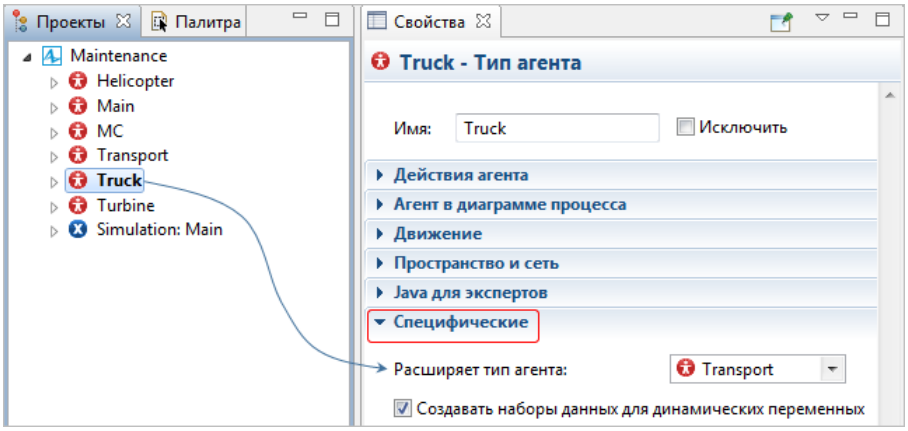
5. Вы можете щелкнуть **Готово** уже на шаге 6. Настойки шага 7, **Конфигурация создаваемой среды**, будут автоматически заполнены, так как мы добавляем популяцию агентов в тип агента `Main`, где мы уже задавали среду при создании агента сервисного центра.
6. Откройте диаграмму типа агента **Turbine**,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

Добавьте транспорт

1. Перетащите элемент **Агент** из палитры **Агент** на диаграмму типа агента `Main`.
2. 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*, то же введите в поле **Имя популяции**: *transport*. Щелкните **Далее**.
3. 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**.

Добавьте в модель грузовики и вертолеты

1. Перетащите элемент **Агент** из палитры **Агент** на диаграмму типа агента `Main`.
2. 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**.
3. 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.
4. После того, как создадите тип агента `Truck`, выделите `Truck` в дереве элементов модели и перейдите в панель **Свойства**. Сначала откройте секцию свойств **Движение**. Мы считаем, что грузовики в среднем движутся со скоростью *70 км в час*.
5. Откройте секцию свойств **Специфические** и задайте типу агента `Truck` наследовать конфигурацию типа агента `Transport`, выбрав его в параметре **Расширяет тип агента:** *Transport*.

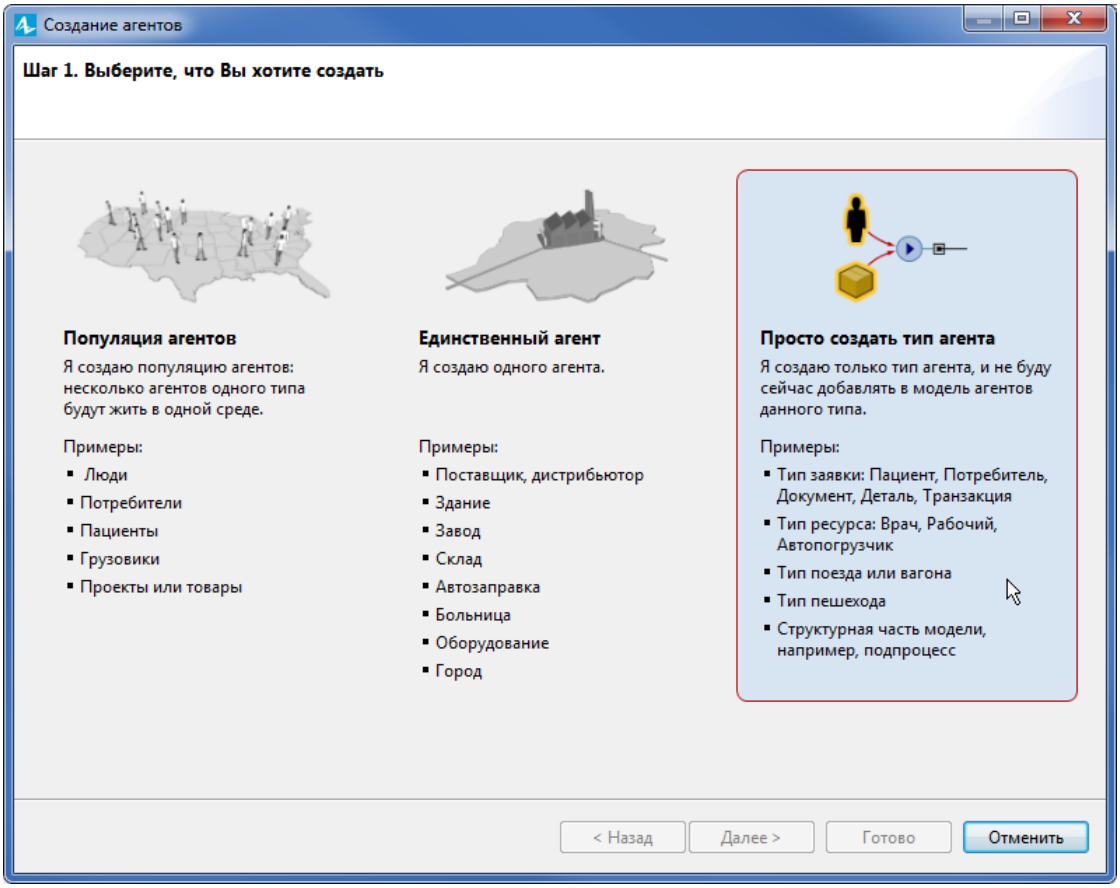


6. Зададим для грузовика чуть увеличенный масштаб. Для этого в свойствах элемента масштаб на диаграмме `Truck`, снимите флажок **Унаследовано от родительского класса**. Пусть длина линейки соответствует 5 километрам, и масштаб будет равен 1 км = 20 пикс.
7. Теперь тип агента `Truck` полностью задан. Нам необходимо добавить еще одну популяцию для вертолетов. Вернитесь на `Main` и снова перетащите элемент **Агент** из палитры **Агент** и выберите **Популяция агентов** на первом шаге.

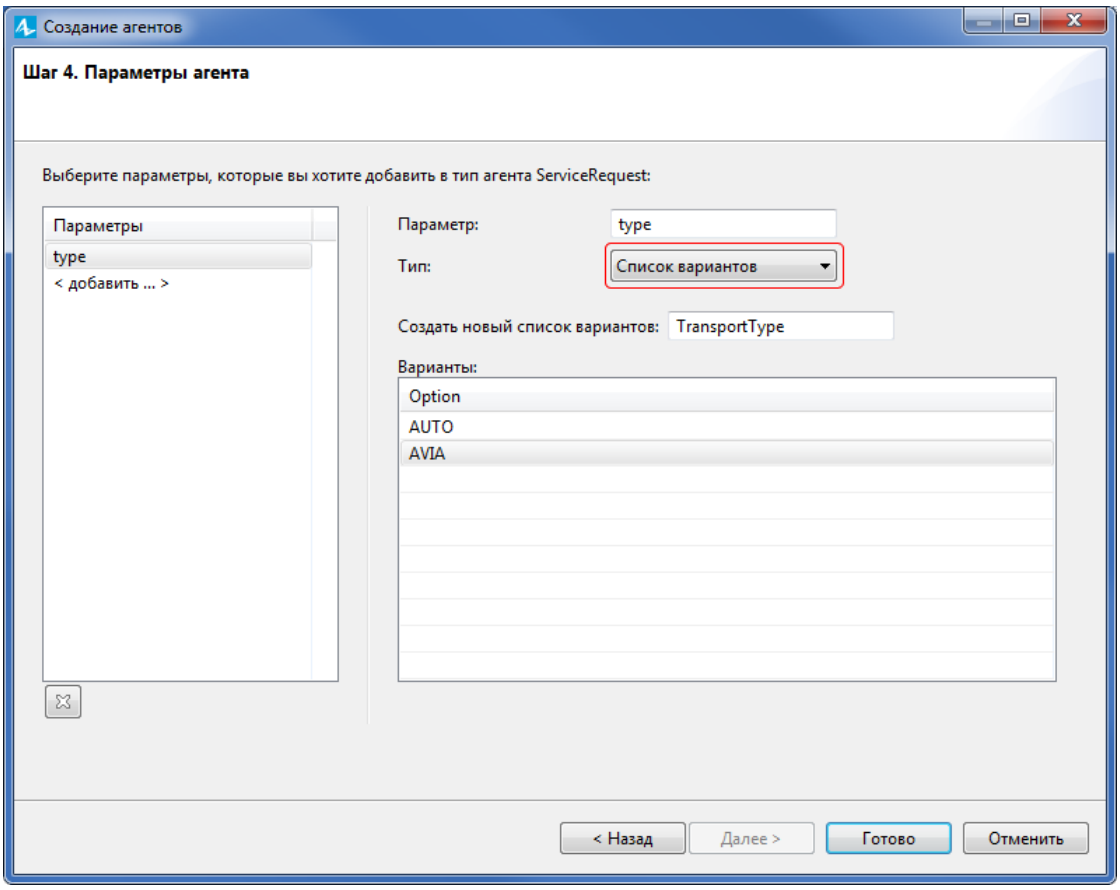
8. Введите имя нового типа *Helicopter*, оставьте имя популяции *helicopters*. Вы можете найти фигуру **3D** анимации *Вертолет* в секции **Военного назначения**. Мы хотим использовать *2* вертолета для обслуживания турбин. Эта популяция живет в той же среде, что и все остальные.
9. `Helicopter` также **Расширяет тип агента**: `Transport`. Мы предполагаем, что вертолеты движутся со скоростью, равной *200 км в час*.
10. Зададим для вертолета масштаб 1 км = 10 пикселей (так же, как и чуть ранее, запретив наследование масштаба от родительского класса).

Добавьте запросы на обслуживание

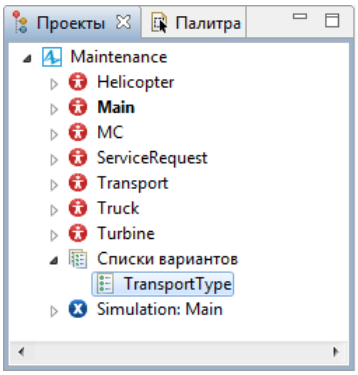
1. Перетащите элемент **Агент** из палитры **Агент** на диаграмму типа агента `Main`.
2. Выберите опцию **Просто создать тип агента**. Нам нужен тип агента, чтобы задать логику модели специальными параметрами, которые мы создадим на его диаграмме.



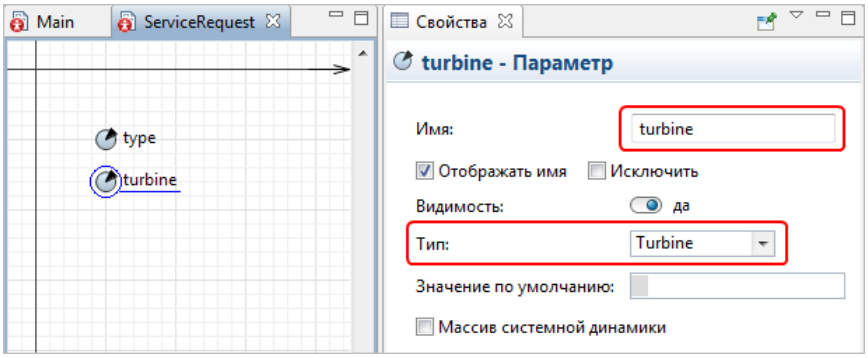
3. Введите имя типа *ServiceRequest* в соответствующем поле и щелкните **Далее**. Выберите **Нет** для типа анимации и перейдите на следующий шаг.
4. На шаге 4 мы создадим в мастере параметр, а также список вариантов. **Список вариантов** - это элемент, который позволяет задавать параметры агента, которые имеют ограниченный выбор вариантов. В нашем случае, нам необходимо различать грузовики и вертолеты 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*.



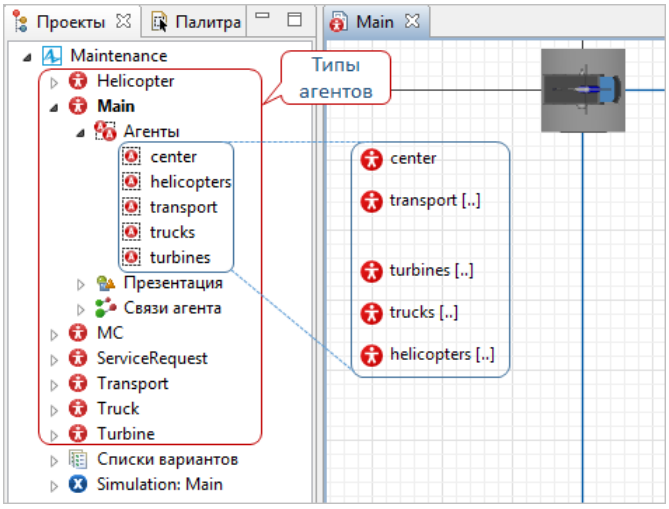
5. Щелкните **Готово**. Список вариантов не имеет отдельного значка, который можно выбрать щелчком в графическом редакторе. Вы можете найти секцию **Список вариантов** в дереве модели:



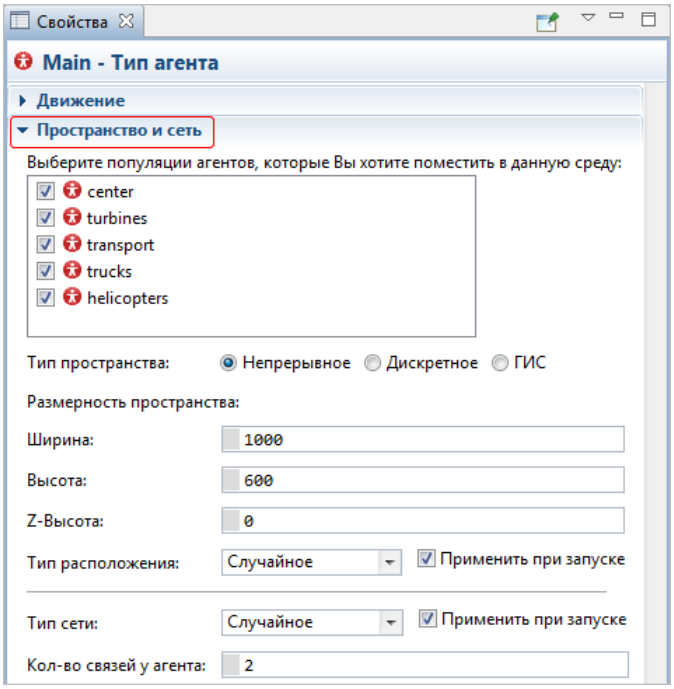
6. После того, как вы щелкните **Готово**, AnyLogic откроет диаграмму типа агента `ServiceRequest`. Здесь вы можете увидеть параметр *type*, который мы создали в мастере. Добавьте еще один **Параметр** из палитры **Агент**. Назовите его *turbine* и выберите его **Тип** из выпадающего списка: **Turbine**. Теперь тип агента `ServiceRequest` полностью задан.



Все агенты появятся на диаграмме `Main`. Вы можете заметить, что популяции отмечены **[..]**. Переместите агентов за рамку окна презентации, а центры всех фигур анимации поместите в начало координат.



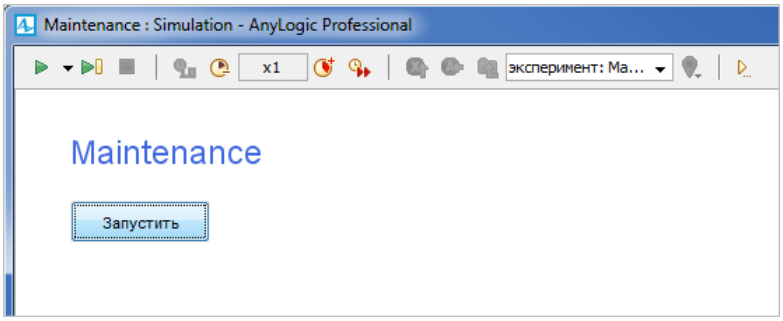
Вы можете найти настройки среды в секции **Пространство и сеть** свойств типа агента `Main`:



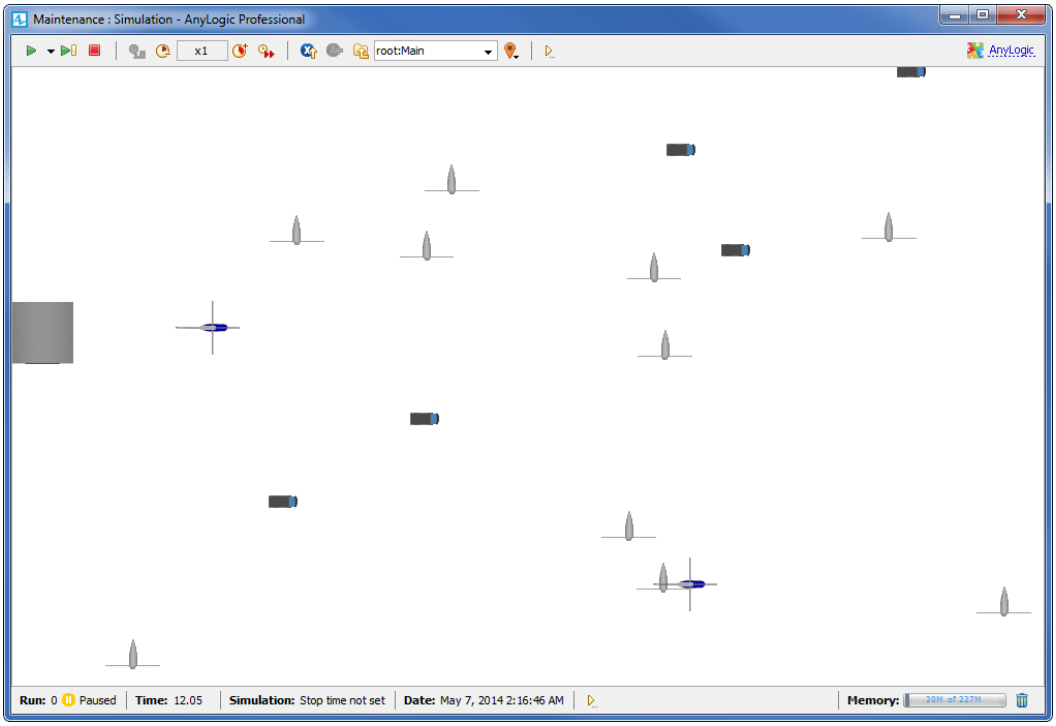
Последнее, что нам необходимо сделать на этом шаге разработки модели: задать масштаб типа агента **Main**:

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. Ваш 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 увидите окно презентации. На ней отображается презентация, созданная для этого эксперимента. AnyLogic автоматически создает заголовок и кнопку запуска модели.
4. Щелкните кнопку **Запустить**.



5. Мы только начали разрабатывать модель. В данный момент мы можем видеть сервисный центр, турбины и разные типы транспорта, случайно расположенные в непрерывном пространстве.



Далее давайте зададим местоположение транспорта, грузовиков и вертолетов, в сервисном центре.

Шаг 2. Задание транспортной базы

Разрабатывая дальше модель, мы будем задавать логику и процессы модели на диаграммах каждого агента в отдельности. Начнем с конфигурации типа агента `Main`, которая задает начальную позицию грузовиков и вертолетов в пространстве.

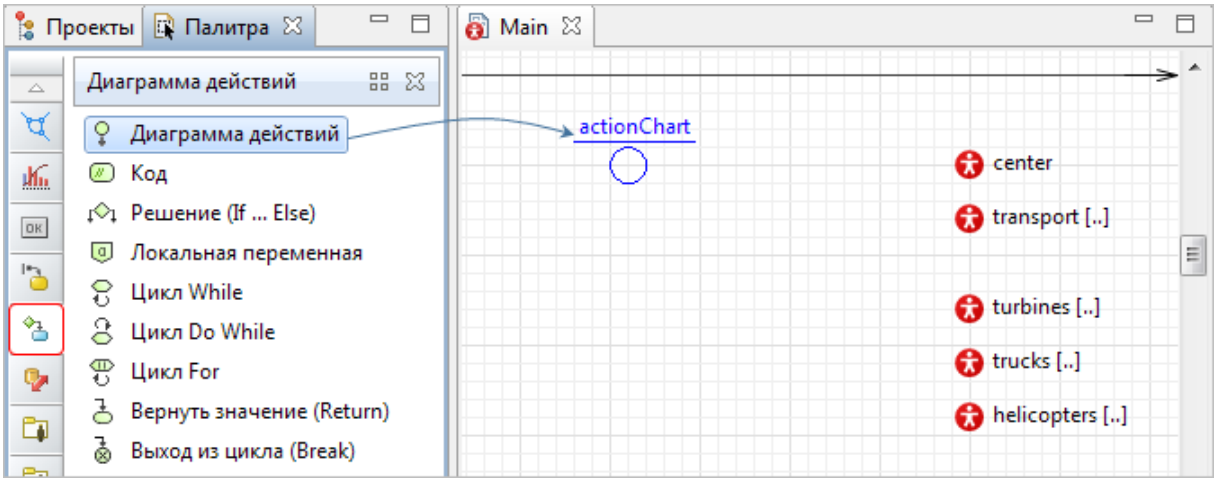
Сложное имитационное моделирование не может существовать без возможности задания алгоритмов, обычно выполняющих определенную обработку данных или вычисления. AnyLogic поддерживает *диаграммы действий* - структурированные блок-схемы, позволяющие задавать алгоритмы графически в стиле структурированного программирования. Мы используем широко известное расширение подхода, предложенного в свое время Дейкстра. Суть подхода состоит в том, что алгоритмы разбиваются в подразделы с одной точкой входа. Утверждается, что трех способов объединения программ — упорядочения, повторения и выбора — достаточно для задания алгоритма любой сложности. Такой стиль сводит понимание целого алгоритма к пониманию составляющих его частностей.

Диаграммы действий облегчают задание алгоритмов, делая необязательным знание синтаксиса Java операторов.

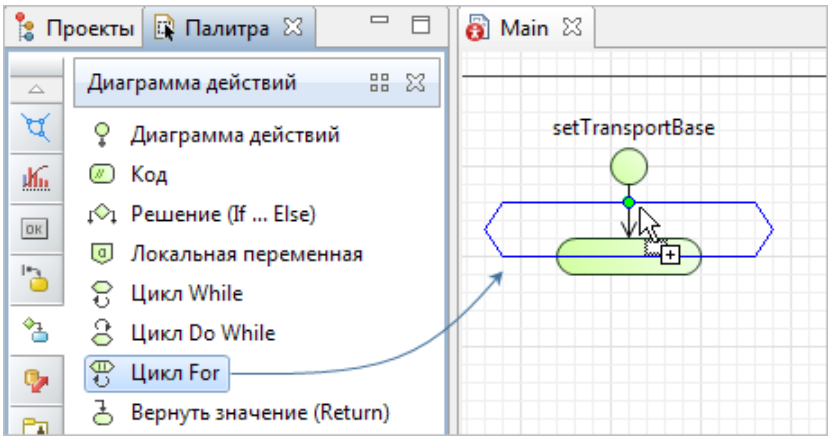
По сути диаграммы действий используются для визуального задания функций, и вы можете использовать для этого обычные функции. Но использование диаграмм действий дает еще одно преимущество: с их помощью вы можете визуализировать алгоритмы, делая их более понятными для других пользователей модели.

Поместите транспорт в сервисный центр

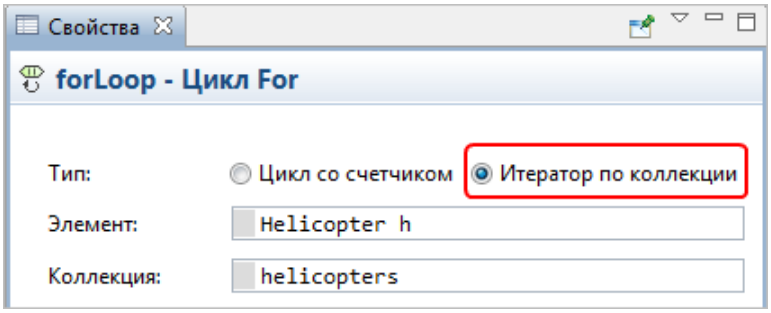
1. Перейдите на диаграмму `Main` и откройте палитру **Диаграмма действий** в панели **Палитра**.
2. Перетащите элемент **Диаграмма действий** в графический редактор `Main`.



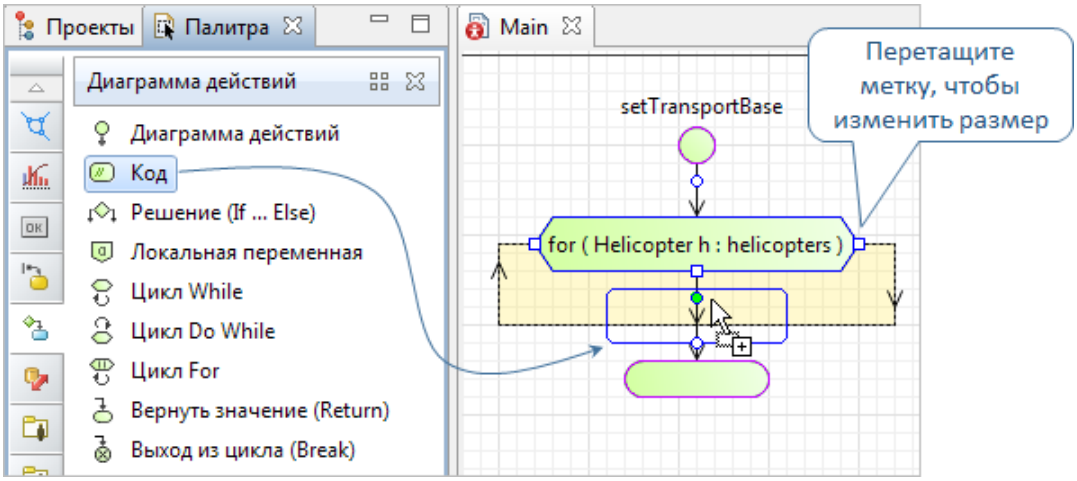
3. Назовите диаграмму *setTransportBase*. Эта диаграмма действий ничего не возвращает, то есть, ее тип - **Действие**. Далее, добавьте в диаграмму элемент **Цикл For**. Работая с элементами диаграммы действий, проверяйте, что вставляете элементы в нужное место:



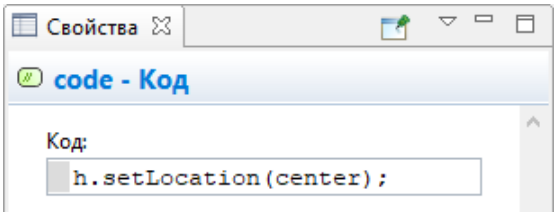
4. Этот цикл *for* является *Итератором по коллекции*, где **Коллекция** - *helicopters*. Определите **Элемент** как *Helicopter h*.



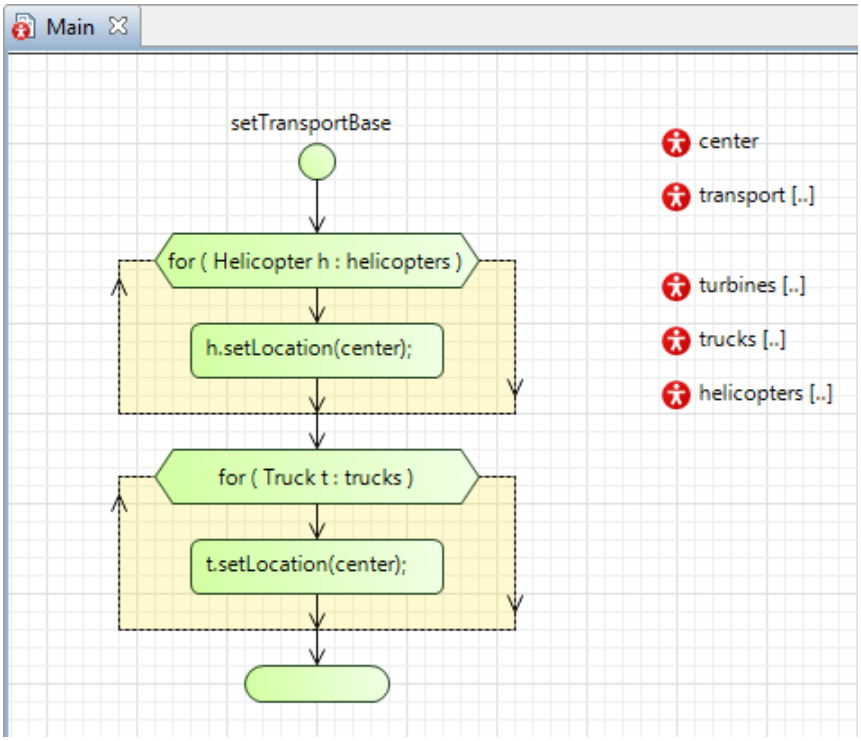
5. Далее, вставьте элемент **Код** в этот цикл *for*.



6. Мы хотим, чтобы вертолеты изначально находились в том же месте, где находится сервисный центр:

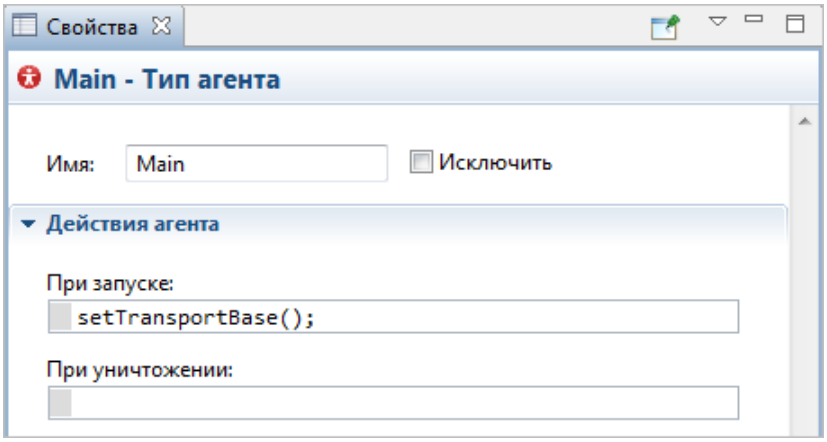


7. Добавьте еще один блок, состоящий из **Цикла For** с элементом **Код** внутри в эту диаграмму действий и сконфигурируйте его размещать грузовики, так же как и вертолеты, в сервисном центре.
8. Вы можете изменить размер блоков диаграммы действий, чтобы содержащийся в них код был полностью виден:

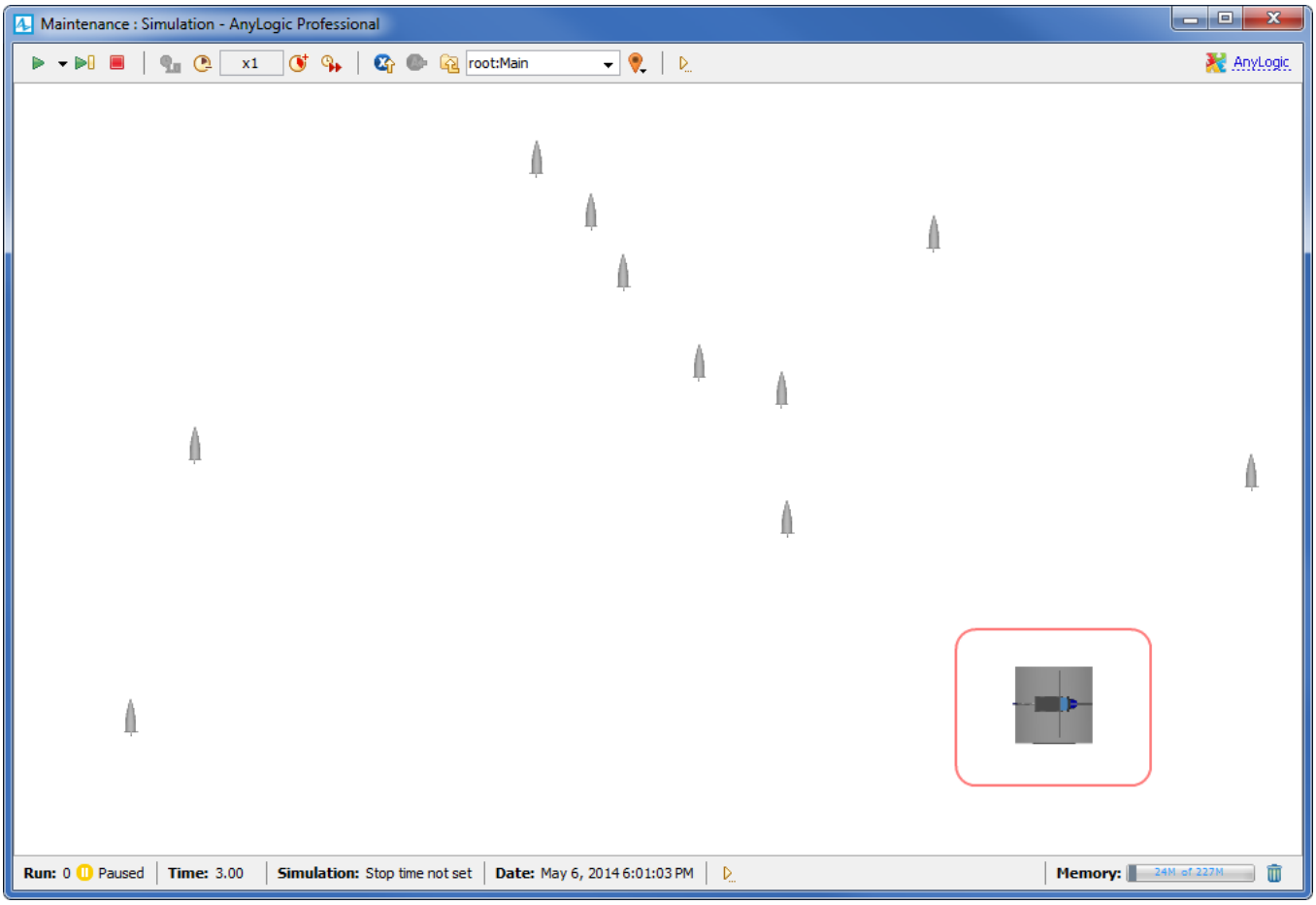


Итак, мы создали диаграмму действий под названием *setTransportBase*, которая представляет функцию, размещающую транспортные средства в сервисном центре. Там мы хотим разместить транспорт при старте модели.

Выделите `Main` в дереве модели (или сделайте щелчок мышью в пустое место в графическом редакторе этого типа агента) и перейдите в панель **Свойства**. Разверните секцию свойств **Действия агента** и задайте запуск нашей функции **При запуске** модели:



Снова запустите модель. Вы увидите, что теперь турбины распределены случайно, а транспортные средства находятся "в ангаре". Мы зададим движение транспорта и поведение других агентов в следующих шагах нашего учебного пособия.



Шаг 3. Настройка логических процессов транспорта

На этом этапе мы зададим начальную конфигурацию типа агента `Transport`.

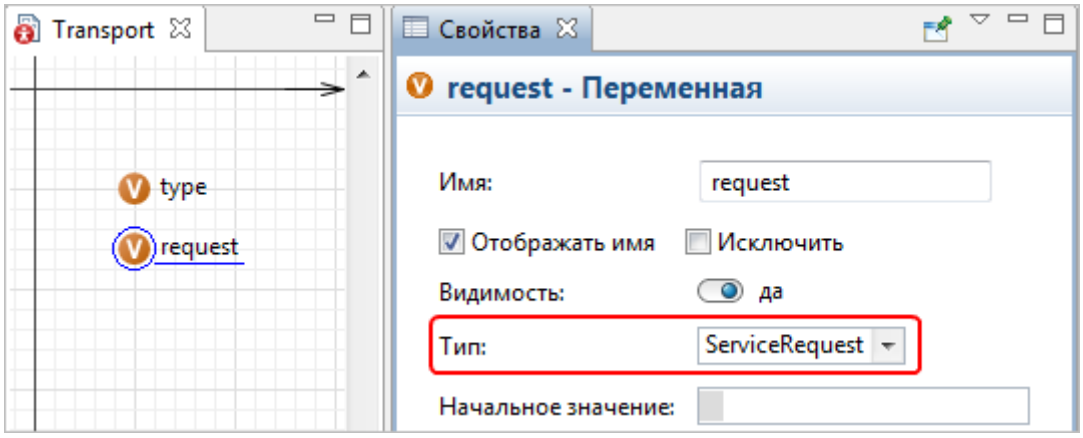
Если у агента можно выделить несколько состояний, выполняющих различные действия при происхождении каких-то событий, или если у агента есть несколько качественно различных поведений, последовательно сменяющих друг друга при происхождении определенных событий, то поведение такого объекта может быть описано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

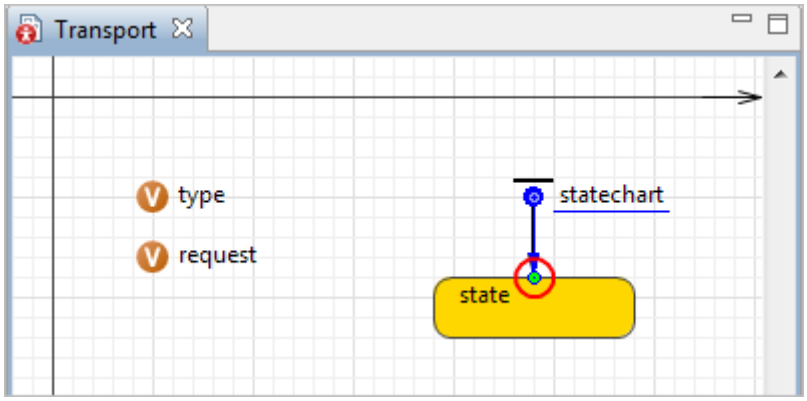
Добавьте необходимые переменные

1. Перейдите в панель **Проекты** и откройте двойным щелчком тип агента `Transport`. Добавьте два элемента **Переменная** из палитры **Агент**.
2. Как вы помните, мы хотим использовать два вида транспорта: грузовики и вертолеты, для обслуживания турбин. Мы используем переменную *type*, чтобы различать эти два вида транспорта. Переменная *type* должна быть типа **TransportType**, чтобы мы могли использовать варианты *AUTO* и *AVIA*.
3. Переменная *request* является переменной типа **ServiceRequest**:

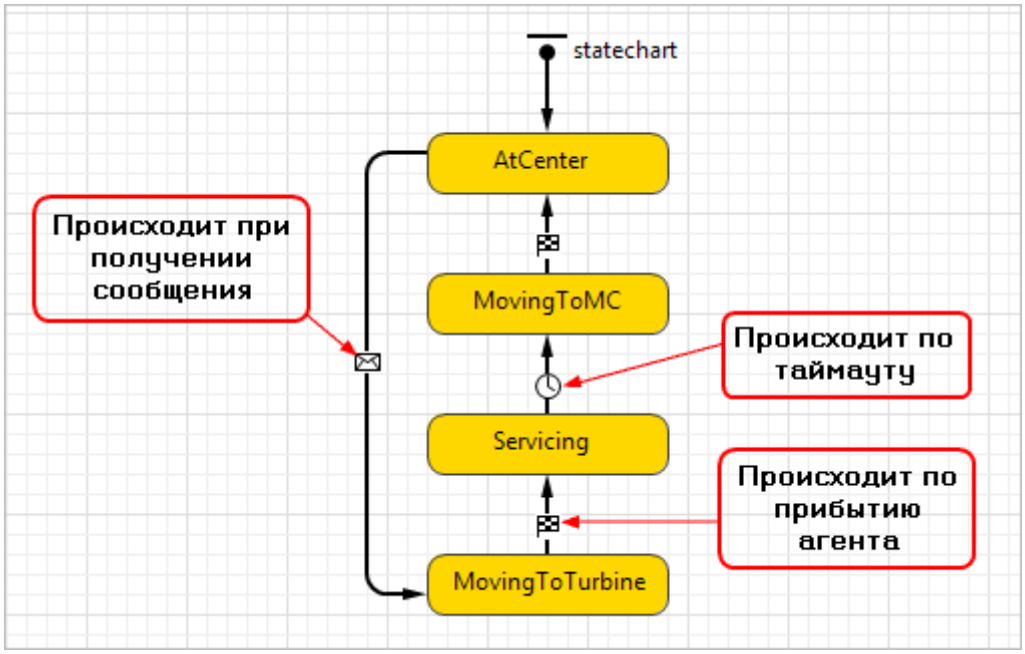


Нарисуйте диаграмму состояний транспорта

1. Мы используем элементы палитры **Диаграмма состояний**, чтобы задать поведение этого типа агента. Перетащите элемент **Начало диаграммы состояний** в графический редактор типа агента `Transport`. Этот элемент задает начало всей диаграммы. Добавьте **Состояние**, перетащив этот элемент из палитры **Диаграмма состояний**, и поместите его в конец стрелки начала диаграммы состояний. 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 зеленую точку в месте соединения.



2. На этом шаге мы создадим "пустую" диаграмму состояний, которая состоит из четырех состояний и трех переходов разных типов.
3. 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.



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

Шаг 4. Настройка поведения сервисного центра

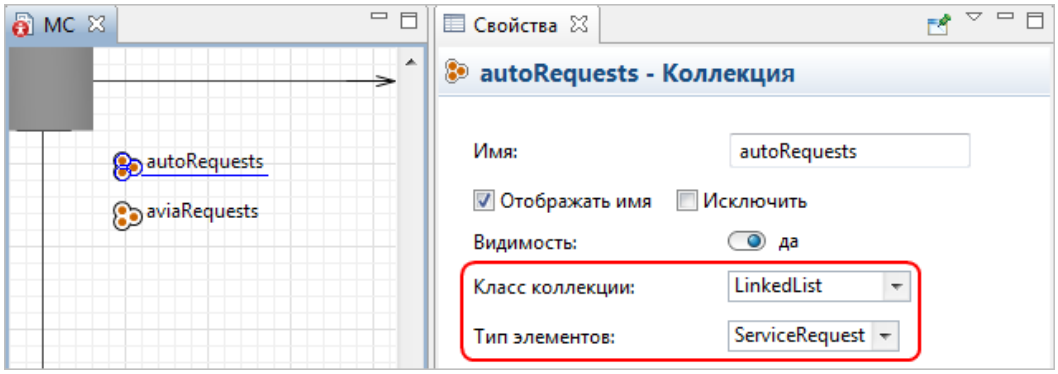
Мы создадим диаграмму действий, с помощью которой сервисный центр определяет, какой тип транспорта необходимо отправить на обслуживание, грузовик или вертолет, и проверяет, есть ли свободное транспортное средство в сервисном центре.

Задайте запросы на транспорт для сервисного центра

Агент может содержать *переменные*. Переменные обычно используются для моделирования изменяющихся характеристик объекта или для хранения результатов работы модели. AnyLogic поддерживает два типа переменных – простые переменные и коллекции.

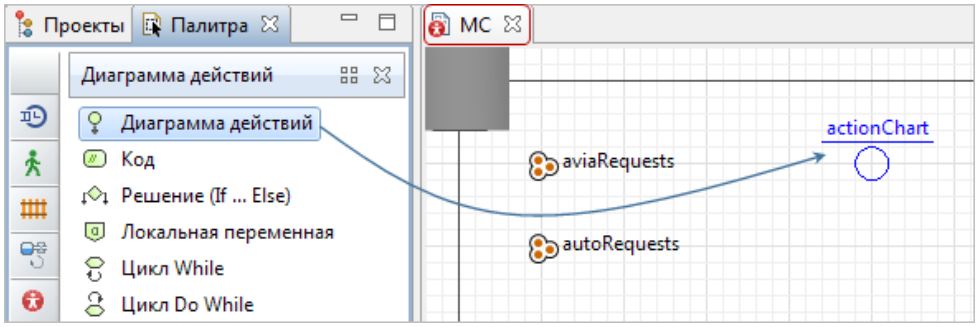
Коллекция представляет собой группу объектов, известных как ее элементы. Некоторые коллекции позволяют хранение нескольких одинаковых элементов, некоторые - нет. Некоторые коллекции упорядочены, некоторые - нет. *Коллекция* используется для задания объекта данных, объединяющего в себе сразу несколько 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 имен и телефонных номеров).

1. Откройте диаграмму `MC`. Добавьте два элемента **Коллекция** из палитры **Агент**.
2. Коллекции называются *aviaRequests* и *autoRequests* и обе имеют **Класс коллекции** *LinkedList* и **Тип элементов** *ServiceRequest*.

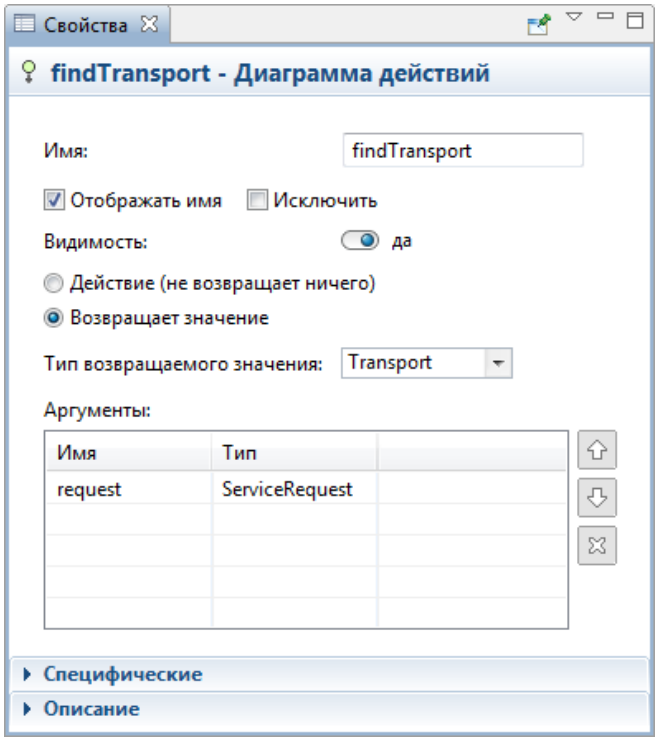


Задайте логику управления транспортным парком

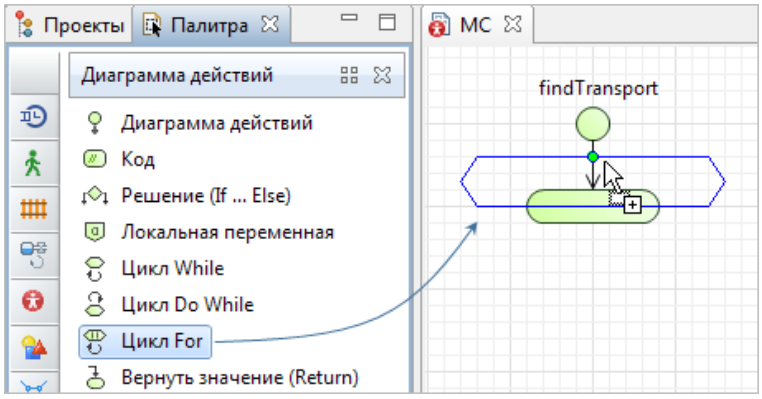
1. Откройте диаграмму `MC`. Мы зададим нужные нам процессы с помощью элементов палитры **Диаграммы действий**.
2. Вначале перетащите элемент **Диаграмма действий** из палитры в графический редактор.



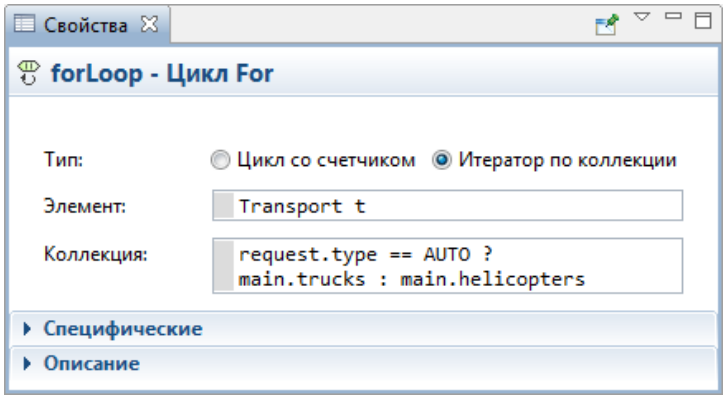
3. Назовите диаграмму *findTransport*. Эта диаграмма действий **Возвращает значение**, и **Типом возвращаемого значения** является **Transport**. Аргументы диаграммы действий задаются ниже в таблице **Аргументы**: аргумент `request` **Типа** `ServiceRequest`:



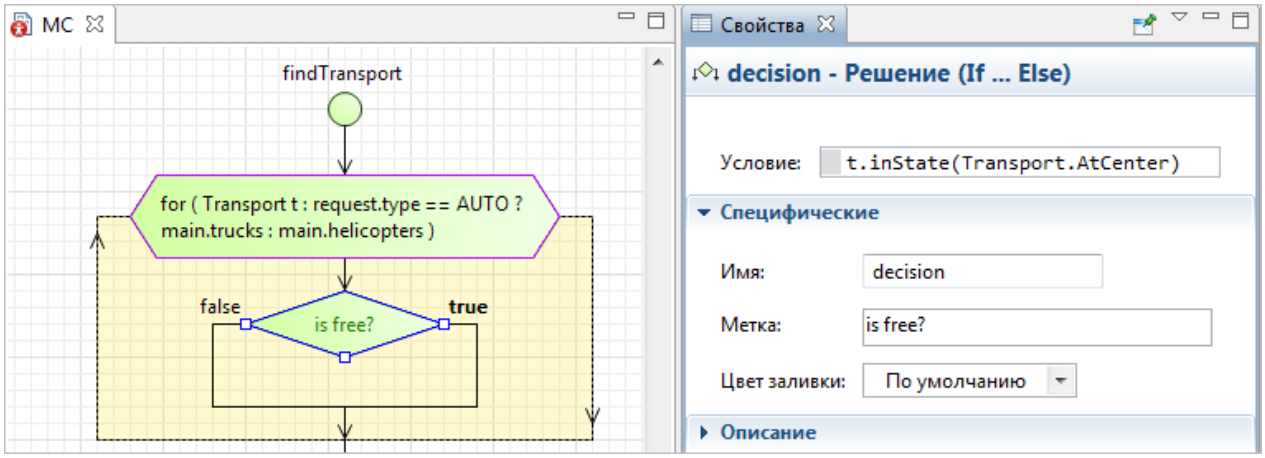
4. Далее, вставьте элемент **Цикл For** в нашу диаграмму действий *findTransport*. Работая с элементами диаграммы действий, убедитесь, что вставляете элементы в правильное место:



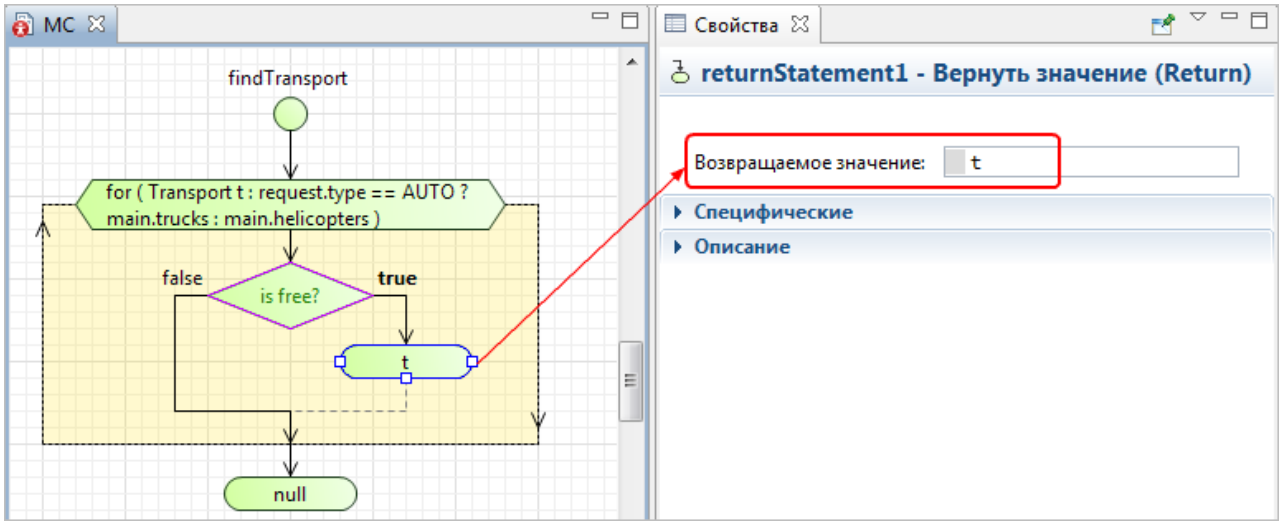
5. **Тип** этого цикла *for*: **Итератор по коллекции**. Укажите поля **Элемент** и **Коллекция** как показано на рисунке ниже: необходимость в грузовике или вертолете определяется типом запроса на обслуживание. Вы можете изменить размер элемента для лучшего отображения содержимого.



6. Нам необходимо добавить **Решение (If ... Else)**, которое будет проверять, имеется ли требуемый транспорт в наличии. Условие заявляет, что транспортное средство должно быть в состоянии *AtCenter* (в сервисном центре) той самой диаграммы состояний, которую мы создали на типе агента `Transport`. Обратите внимание, что в графическом редакторе отображается содержимое параметра **Метка**:



7. Вставьте элемент **Вернуть значение** в ветку `true`, выходящую из решения. Пусть этот элемент возвращает `t`. В качестве последнего шага, выделите элемент диаграммы **Вернуть значение**, который возвращает значение и для всей диаграммы, и для ветки `false` и укажите **Возвращаемое значение**: `null`.



8. Эта диаграмма действий будет проверять, есть ли в наличии в сервисном центре запрашиваемый вид транспорта: грузовик или вертолет.

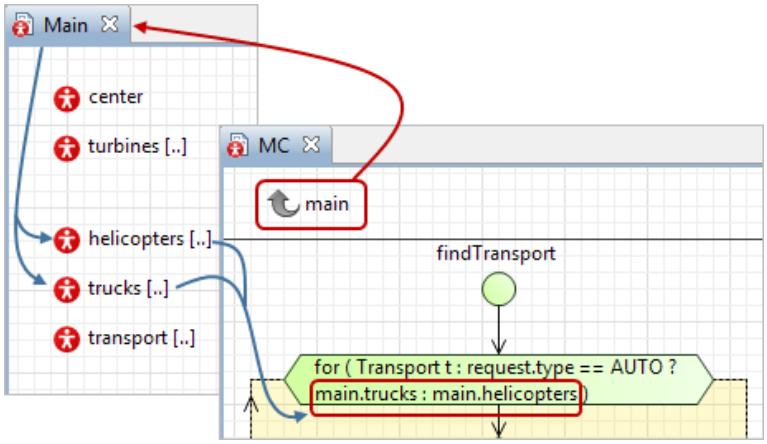
В AnyLogic иногда вам необходимо вводить код в параметрах различных элементов модели. Важно понимать, где именно вы "находитесь", вписывая код (какому типу агента принадлежит этот элемент), и как получить доступ к другим элементам из этого поля.

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

Чтобы получить доступ к полю вложенного объекта, вы должны поставить точку "." после имени объекта и затем написать имя этого поля. Например, мы ссылались на функцию `getX()` сервисного центра *center* при создании диаграммы действий на `Main`: `center.getX()`. Если объект является реплицированным, то его имя является именем коллекции объектов, и вам нужно указать, какой именно объект из этой коллекции вам нужен.

Чтобы получить доступ к "равному по иерархии" агенту (который вложен в тот же контейнер, что и текущий агент), нужно вначале перейти на уровень выше к агенту верхнего уровня, а затем спуститься по иерархии модели вниз, к нужному вложенному агенту.

Чтобы получить доступ к контейнеру текущего объекта (того агента, куда вложен текущий), укажите его имя. Обратите внимание на специальный элемент, **Ссылка на агента верхнего уровня**, который присутствует на диаграмме каждого типа, чьи агенты живут в среде этого агента верхнего уровня (в нашей модели это тип агента `Main`). Ссылка называется по имени этого агента верхнего уровня и позволяет обращаться к нему по этому имени, чтобы получить доступ к элементам, находящимся на его диаграмме. Например, чтобы получить доступ к элементу **trucks[..]**, который находится на диаграмме `Main`, из агента `MC`, мы пишем `main.trucks`, находясь на диаграмме `MC`.

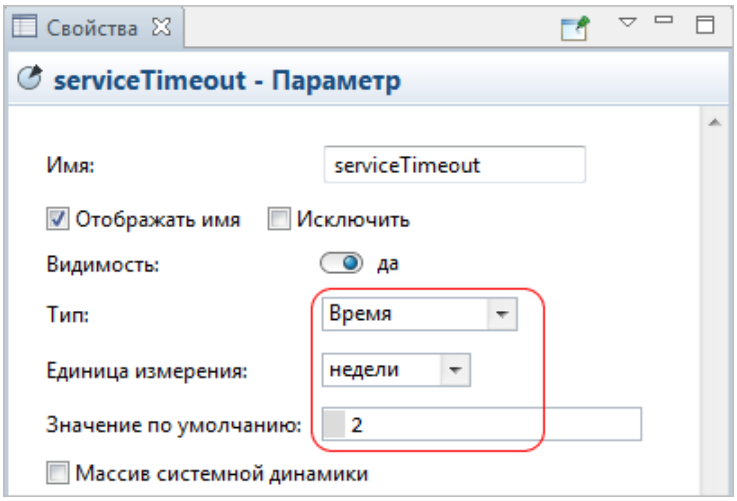

Main
center
turbines [..]
helicopters [..]
trucks [..]
transport [..]
MC
main
findTransport
for (Transport t : request.type == AUTO ?
main.trucks : main.helicopters)

Шаг 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, мы хотели бы видеть цветовую индикацию состояния турбины.

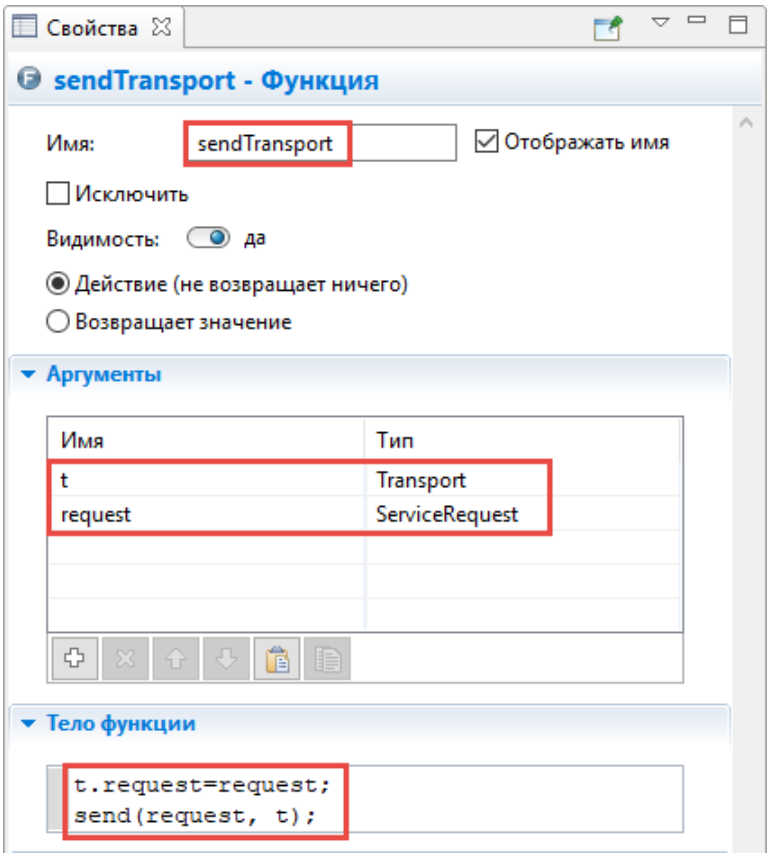
Задайте временные интервалы работы турбин

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) имеет тип **Время** и его **Значение по умолчанию** равняется *50 дням*. *Дни* Вы можете выбрать в свойстве **Единица измерения**. Второй параметр, *serviceTimeout*, также имеет тип **Время**; его **Значение по умолчанию** равняется *2 неделям*.

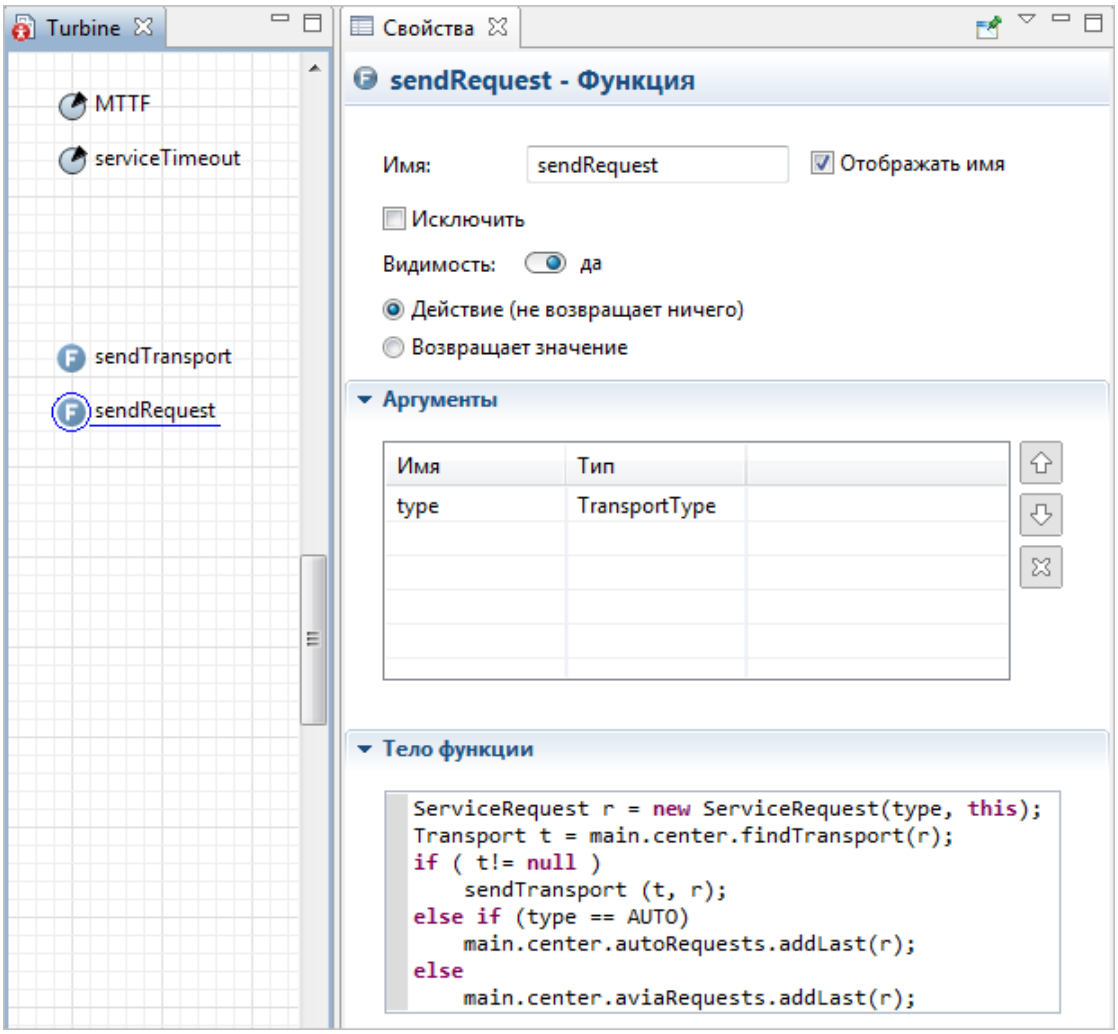


Добавьте функции, управляющие запросами на транпорт

1. Добавьте **Функцию** из палитры **Агент** и назовите ее *sendTransport*. Эта функция является просто **Действием (не возвращает ничего)** и имеет два аргумента, которые необходимо добавить в секции **Аргументы**: `Transport t` и `ServiceRequest request`. Эта функция отправляет транспорт к турбине, когда это необходимо, передава собщение диаграмме состояний, заданной на диаграмме типа агента `Transport`.

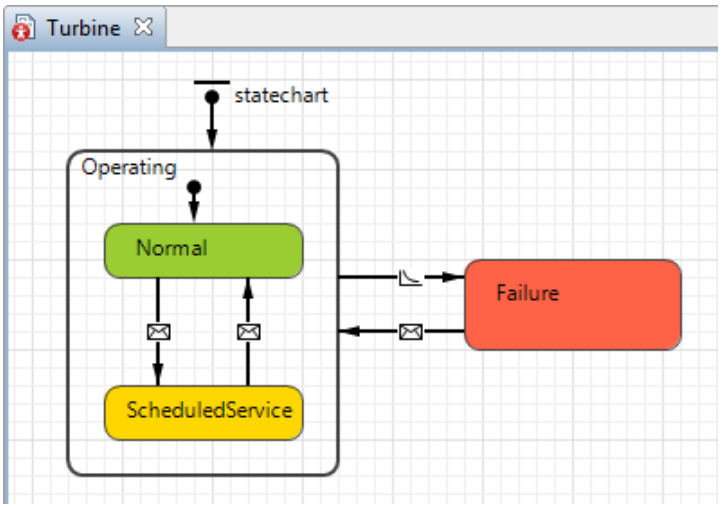


2. Добавьте еще одну **Функцию** . Назовите ее *sendRequest* и задайте ее свойства, как указано на рисунке ниже. В секции свойств **Аргументы** добавьте аргумент `TransportType type`. Тело функции ссылается на диаграмму действий *findTransport*, которую мы ранее создали на диаграмме `MC`. Когда турбина отправляет запрос на обслуживание, сервисный центр должен отправлять соответствующий тип транспорта: *AUTO* или *AVIA*.



Задайте состояния турбины

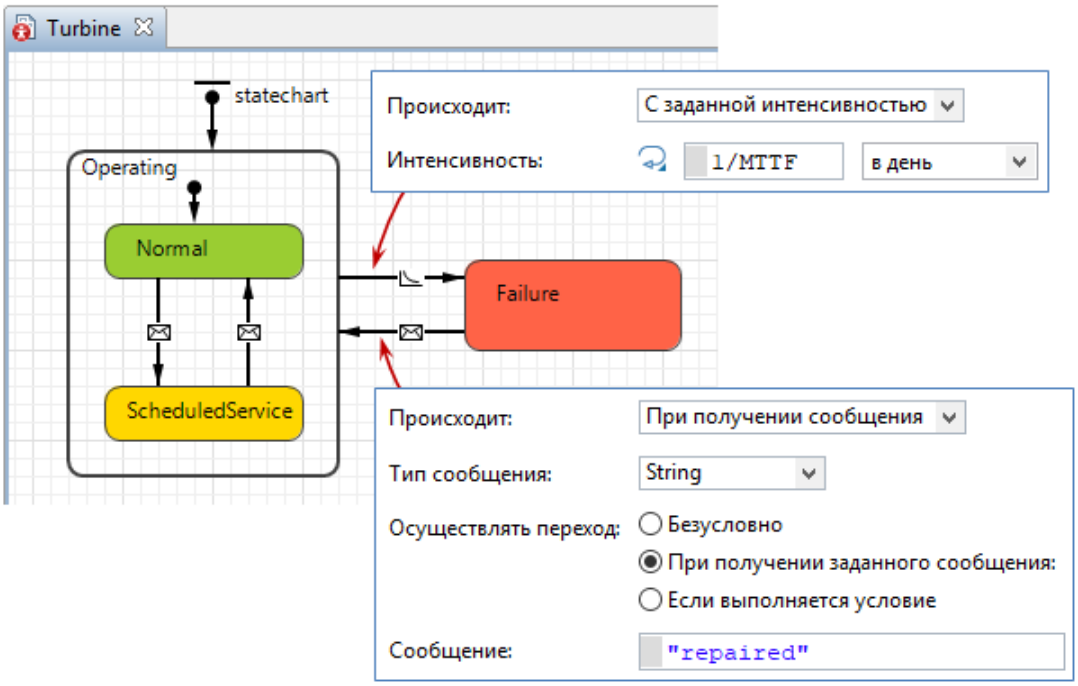
1. Теперь мы готовы приступить к созданию диаграммы состояний турбины. Откройте палитры **Диаграмма состояний** и перетащите на диаграмму `Turbine` элемент **Начало диаграммы состояний** .
2. Добавьте состояния и переходы , как показано на рисунке. Внутри сложного состояния *Operating* используйте **Указатель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



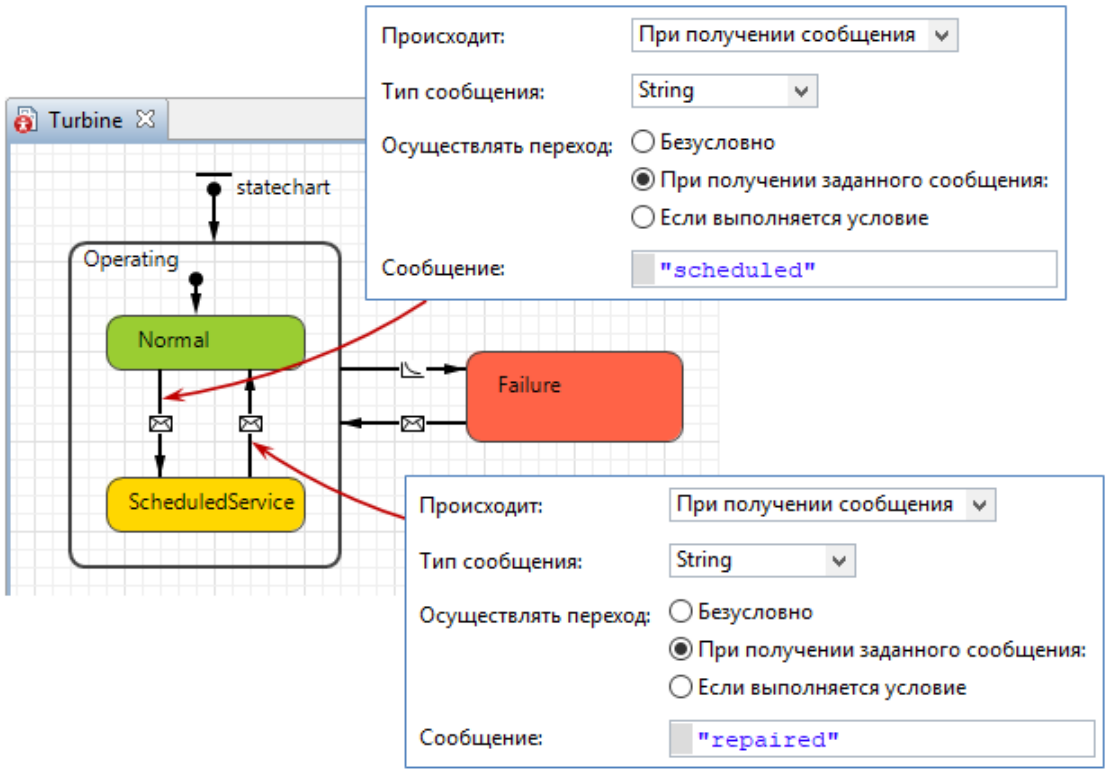
3. Для каждого состояния турбины задается свое **Действие при входе**. Когда происходит авария, турбина находится в состоянии *Failure* (авария) и отправляет запрос на обслуживание *AVIA* транспортом (вертолетом). Когда подходит время планового обслуживания, турбина отправляет запрос на *AUTO* транспорт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

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

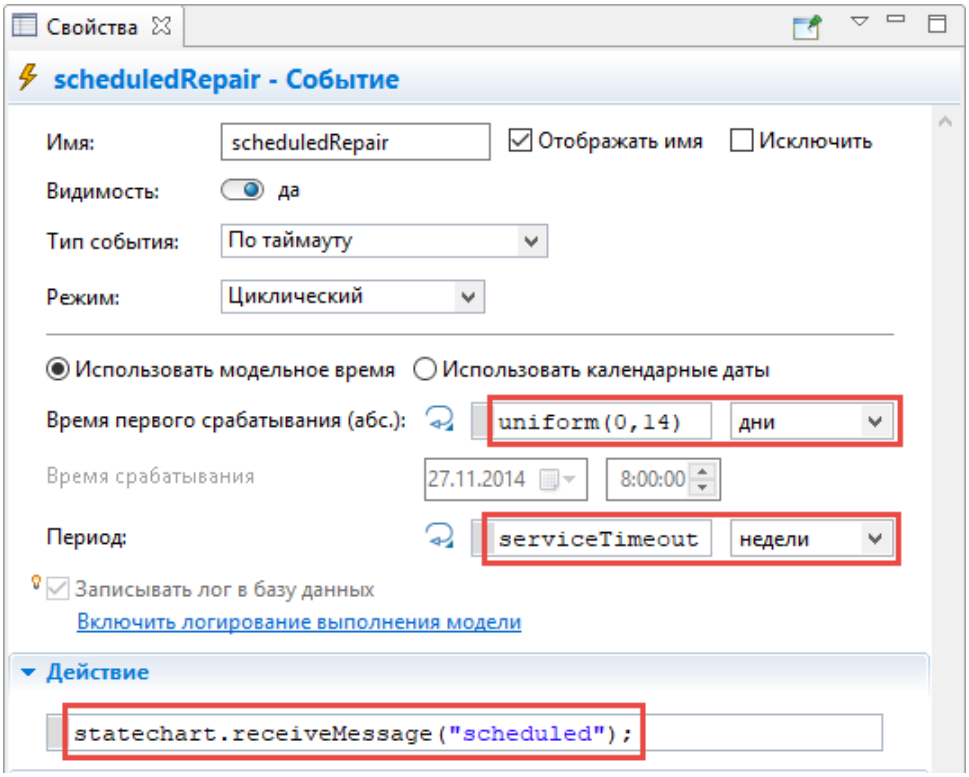
4. Переход от состояния *Operating* (работает) к состоянию *Failure* происходит **С заданной интенсивностью**, которая равняется *1/MTTF* в день - один раз за среднее время до аварии. Переход обратно из состояния *Failure* в состояние *Operating* происходит **При получении заданного сообщения** *"repaired"*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



5. 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*.

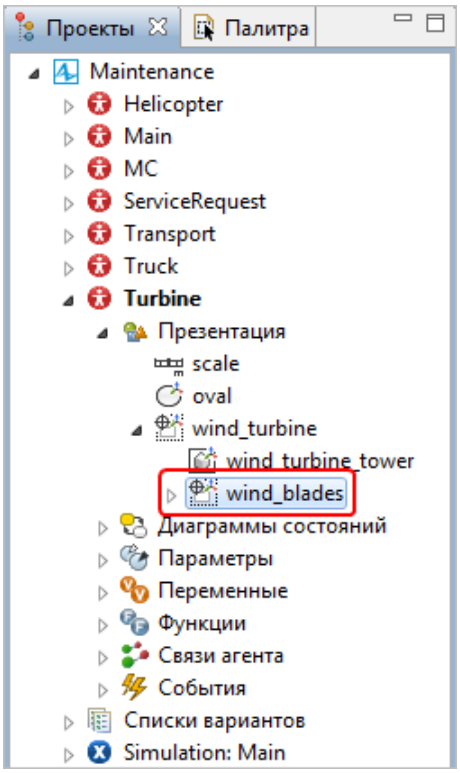


6. Добавьте циклическое **Событие** по таймауту и назовите его *scheduledRepair*. Это событие будет запускаться, когда турбине требуется плановое обслуживание (согласно переменной *serviceTimeout*), и оно будет запускать переход от состояния *Operating* в состояние *ScheduledService*.

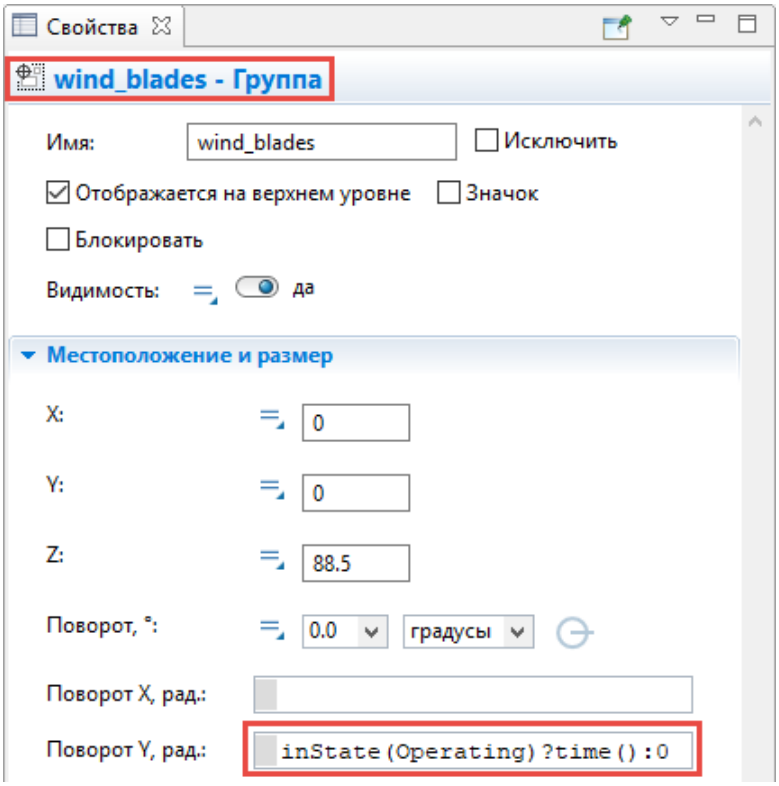


Настройте анимацию лопастей турбины

1. Нам необходимо изменить свойства лопастей турбины. Фигура анимации *Ветряная турбина* является группой, состоящей из двух 3D фигур: башни турбины и лопастей. Нам нужно открыть свойства лопастей.
2. Вы можете выделить лопасти щелчком мыши в графическом редакторе, или открыть панель **Проекты** и раскрывать уровни дерева модели, пока не найдете группу **wind_blades**:

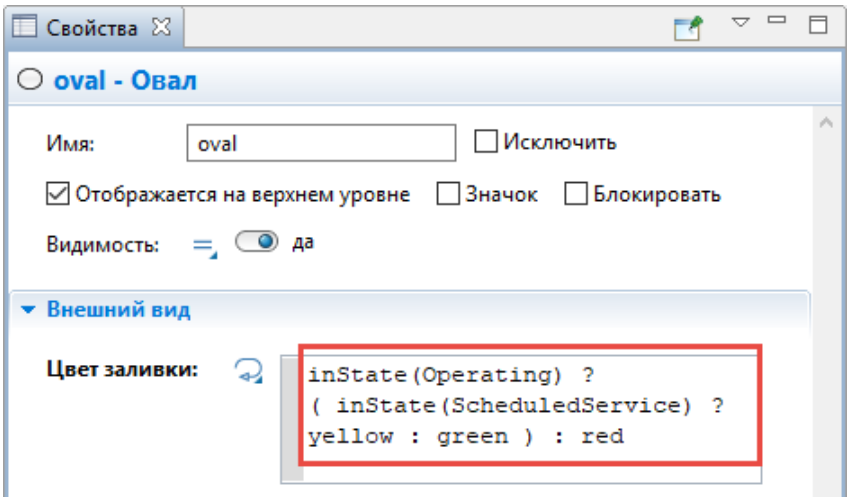


3. Перейдите в свойства этой группы. Разверните секцию **Местоположение и размер** и введите `inState(Operating)?time():0` в поле свойства **Поворот Y**. Теперь лопасти будут вращаться, когда турбина находится в рабочем состоянии, а при поломке лопасти будут останавливаться.



Добавьте индикацию состояния турбины

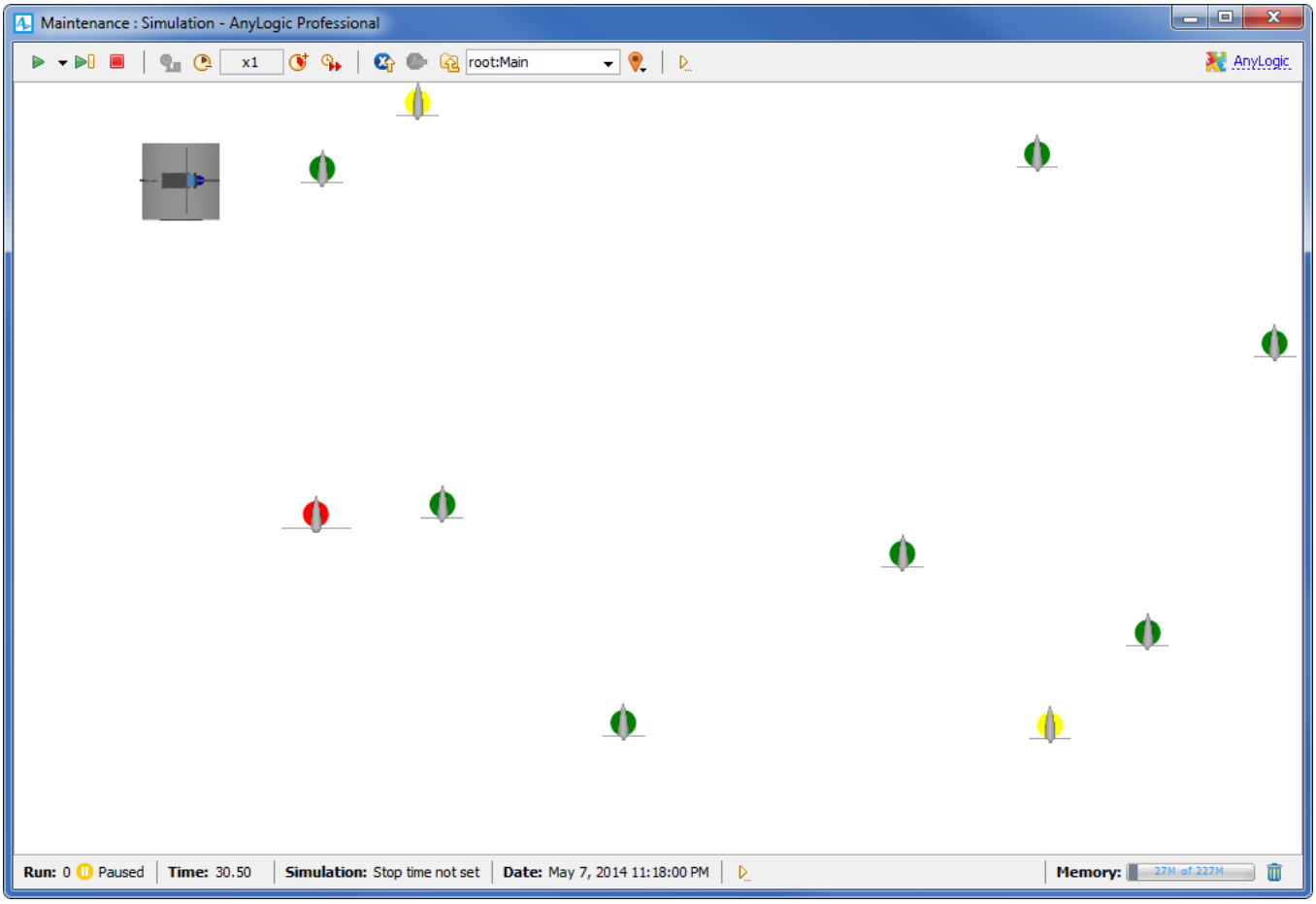
1. Откройте палитру **Презентация** и сделайте двойной щелчок по элементу **Овал**, чтобы перейти в режим рисования для этого элемента.
2. Нарисуйте круг вокруг фигуры турбины радиусом 10.
3. Щелкните по фигуре круга правой кнопкой мыши и выберите **Порядок > На задний план** в контекстном меню.
4. Перейдите 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, щелкните его значок .



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

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

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


Maintenance : Simulation - AnyLogic Professional
x1
root:Main
AnyLogic
Run: 0 Paused
Time: 30.50
Simulation: Stop time not set
Date: May 7, 2014 11:18:00 PM
Memory:
27M of 227M

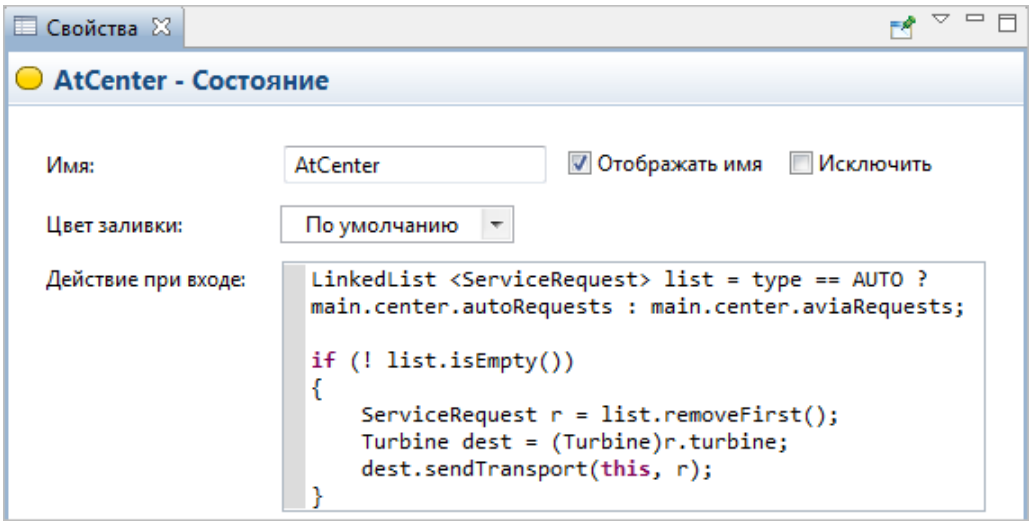
-

Шаг 6. Завершение настройки логики транспорта

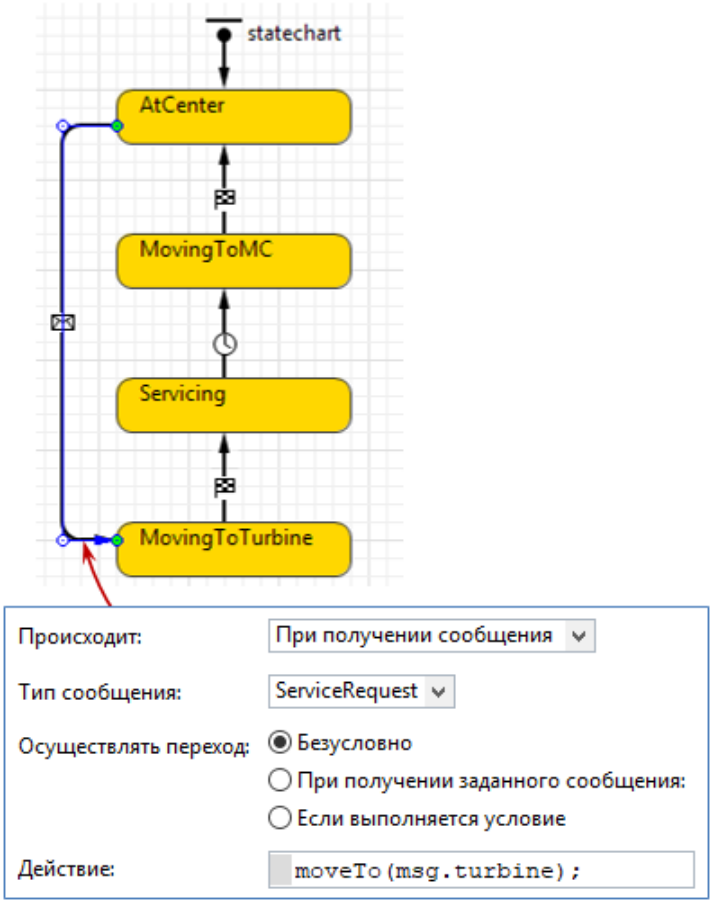
На этом шаге мы добавим необходимые настройки в диаграмму состояний типа агента `Transport`. Мы зададим движение транспорта между сервисным центром и турбинами.

Настройте диаграмму состояний

1. Откройте диаграмму агента `Transport` двойным щелчком из дерева модели. На ней вы найдете диаграмму состояний, которую мы создали ранее. Теперь давайте зададим логику ее работы.
2. Выделите состояние *AtCenter*. Нам необходимо задать его **Действие при входе**:

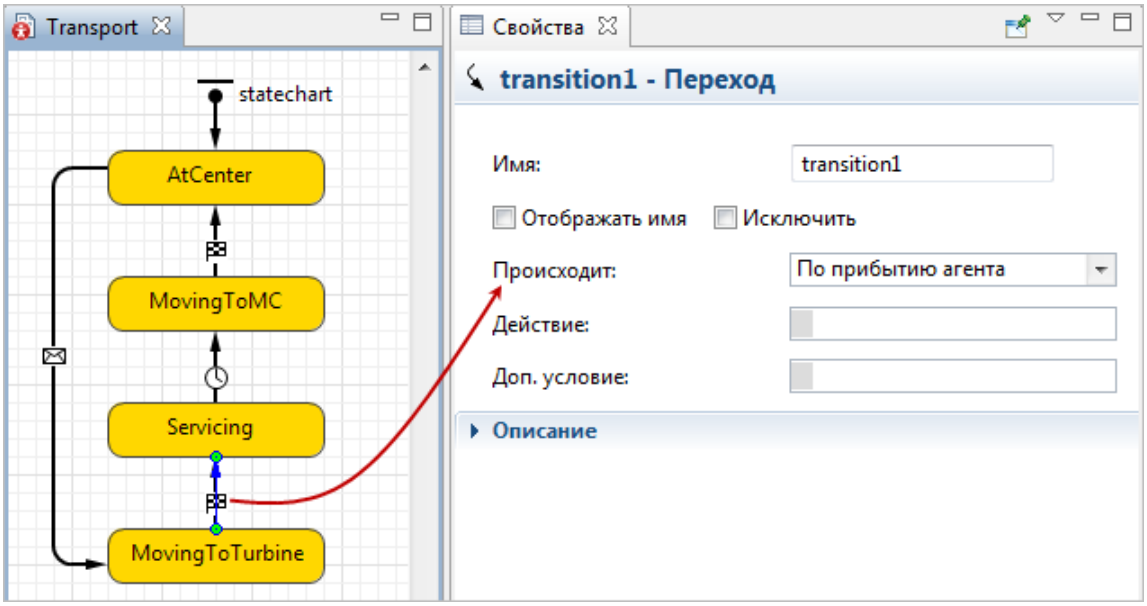


3. Остальные три состояния (*MovingToMC*, *Servicing*, *MovingToTurbine*) не имеют дополнительных настроек. Мы настроим логику движения транспорта с помощью переходов различных типов.
4. Переход из состояния *AtCenter* в состояние *MovingToTurbine* происходит **При получении сообщения** типа **ServiceRequest**. Так как переход осуществляется **Безусловно**, вам необходимо только указать **Действие**, которое следует выполнить транспорту: двигаться к турбине.

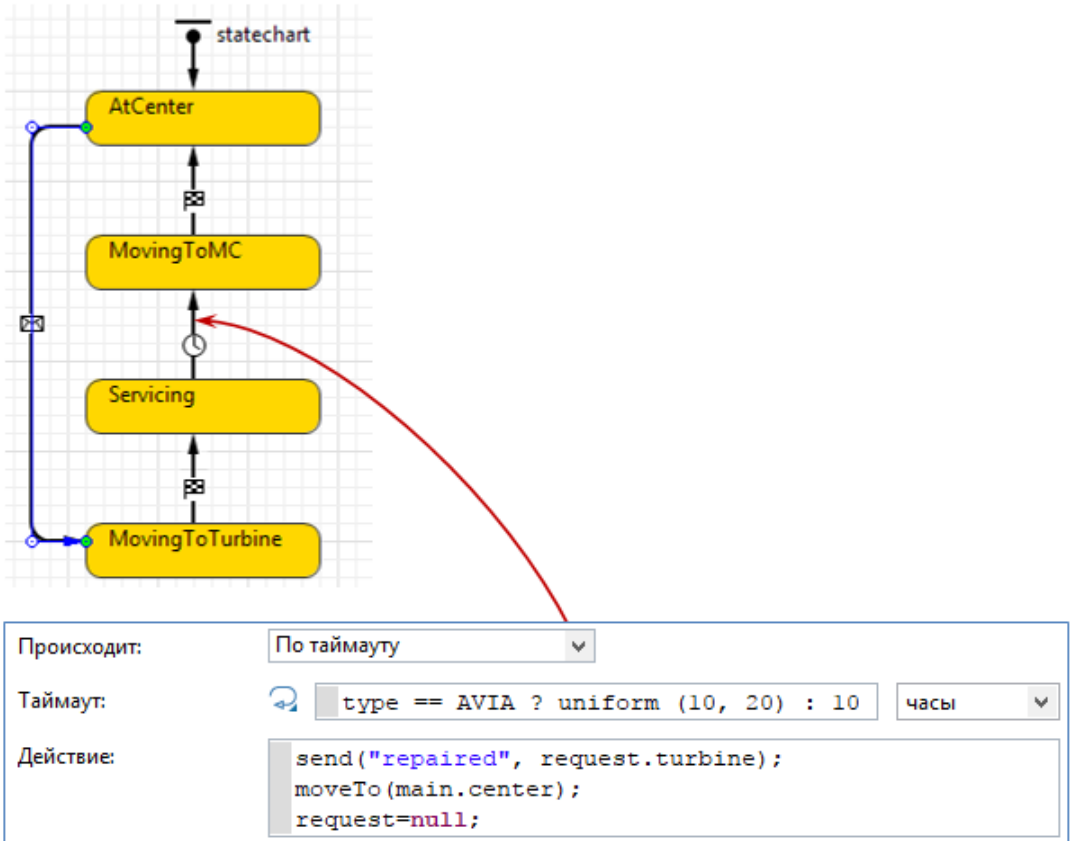


5. Переход из состояния *MovingToTurbine* в состояние *Servicing* запускается **По прибытию агента**. Грузовик или вертолет, который достиг турбины, начинает выполнять свою задачу сразу после прибытия на место. По

завершению обслуживания, о чем нас "оповещает" турбина, транспортное средство может отправляться обратно на базу.

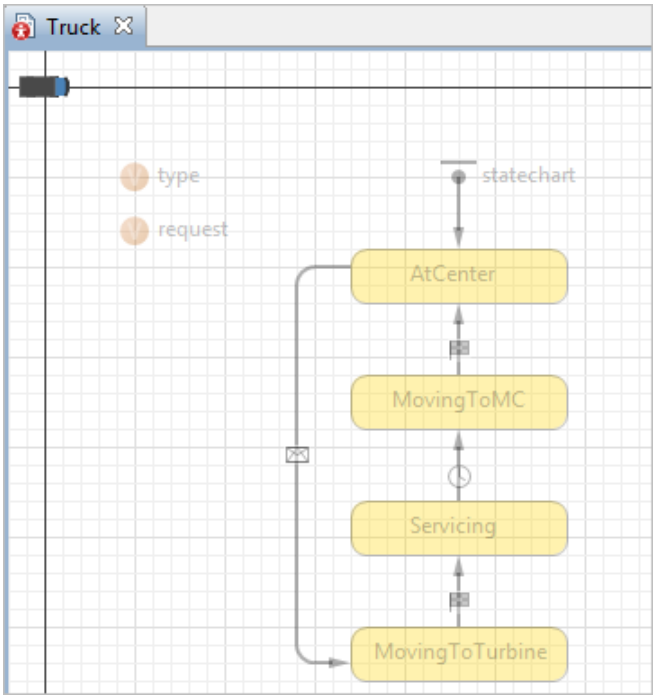


6. Следующий переход идет из состояния *Servicing* в состояние *MovingToMC*. Действие, задающее возвращение в сервисный центр, происходит **По таймауту**.

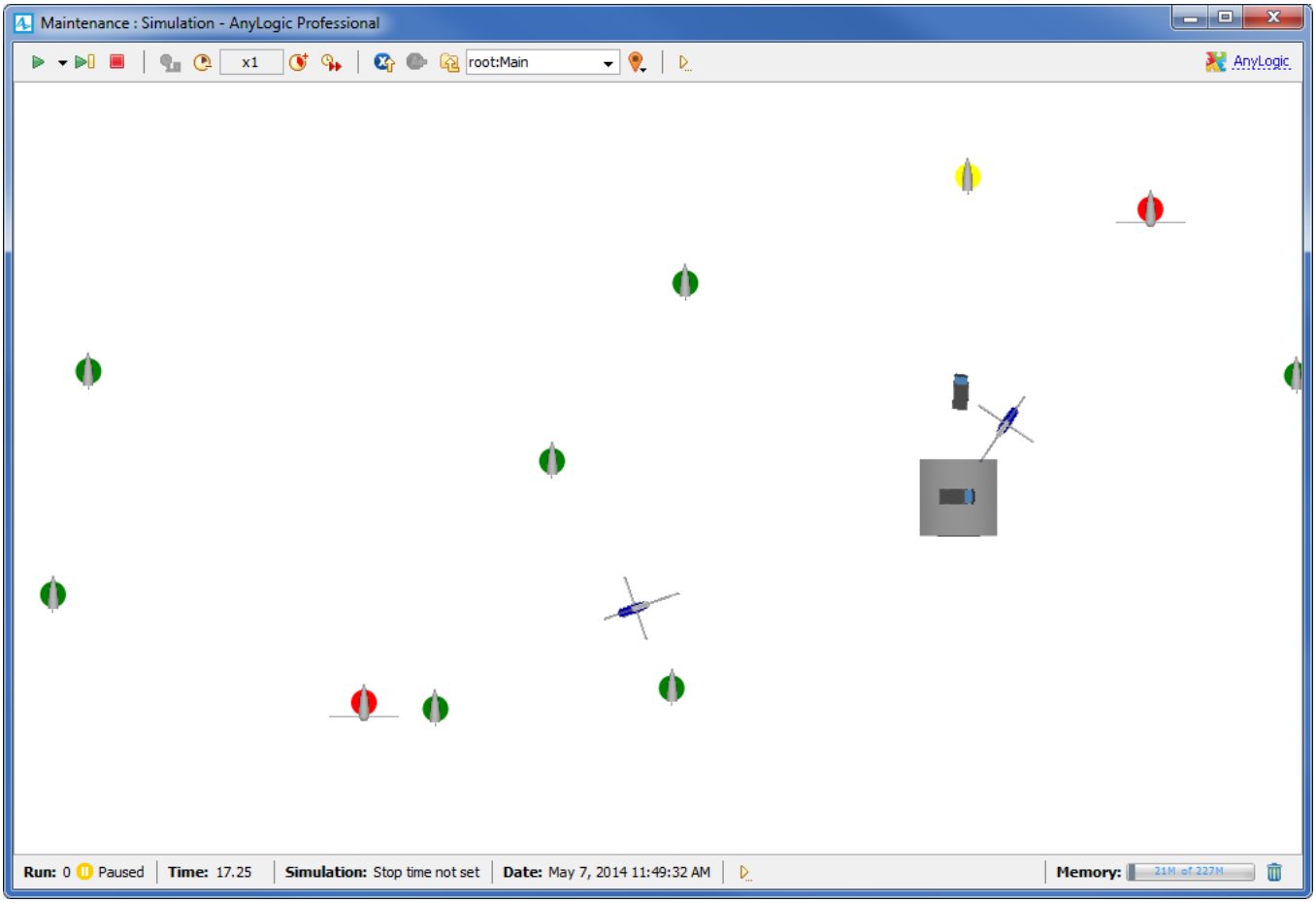


7. На рисунке выше вы видите, что последний переход, из *MovingToMC* в *AtCenter*, происходит **По прибытию агента**. Когда он происходит, транспортное средство отправляется обратно в сервисный центр и остается там, пока снова не понадобится для обслуживания очередной турбины.

Теперь, если вы 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.

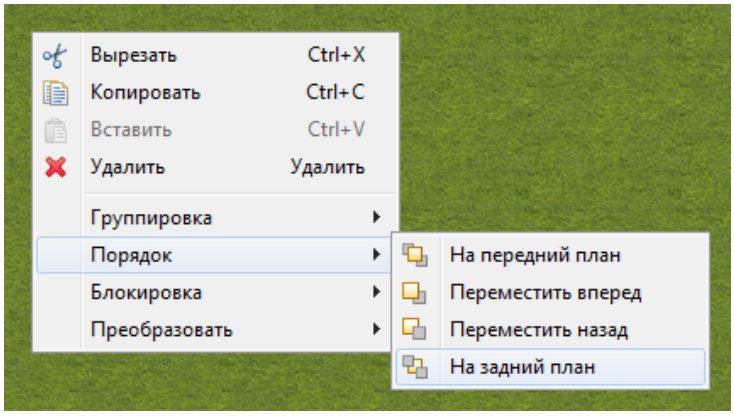


Шаг 7. Запуск модели и исследование ее элементов

Это последний шаг нашего учебного пособия по разработке агентной модели обслуживания турбин.

Добавьте элементы презентации

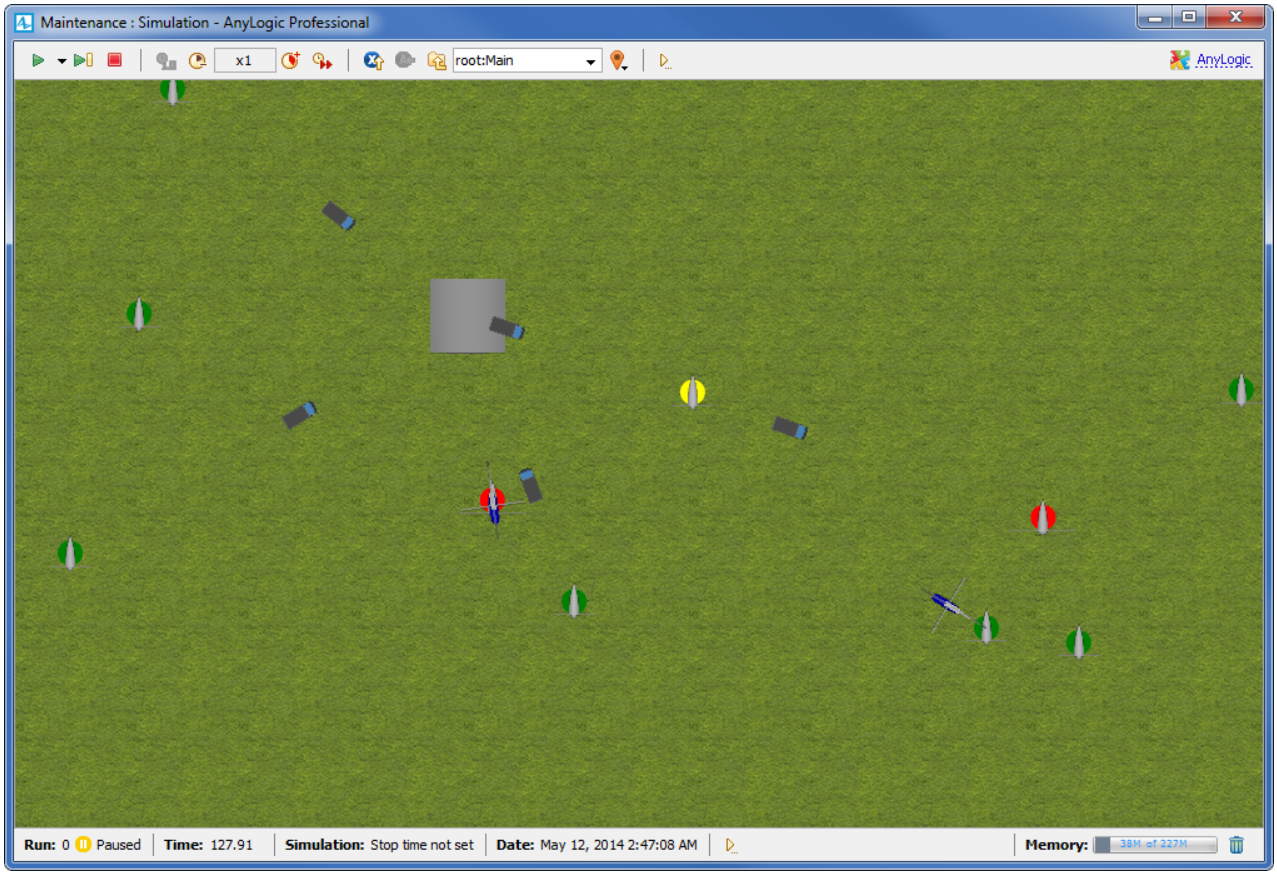
1. Вы можете добавить фон для анимации агентов - турбин и транспортных средств - на диаграмме типа агента `Main` с помощью элемента **Прямоугольник**. Выделите элемент **Прямоугольник** в палитре **Презентация**, чтобы перейти в режим рисования. Затем щелкните в графический редактор и тащите прямоугольник, не отпуская кнопки мыши, пока не получите нужную форму прямоугольника.
2. 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.
3. Так как прямоугольник - это последняя добавленная нами фигура, он отображается поверх всех остальных. Щелкните прямоугольник правой кнопкой мыши и выберите **Порядок > На задний план**.



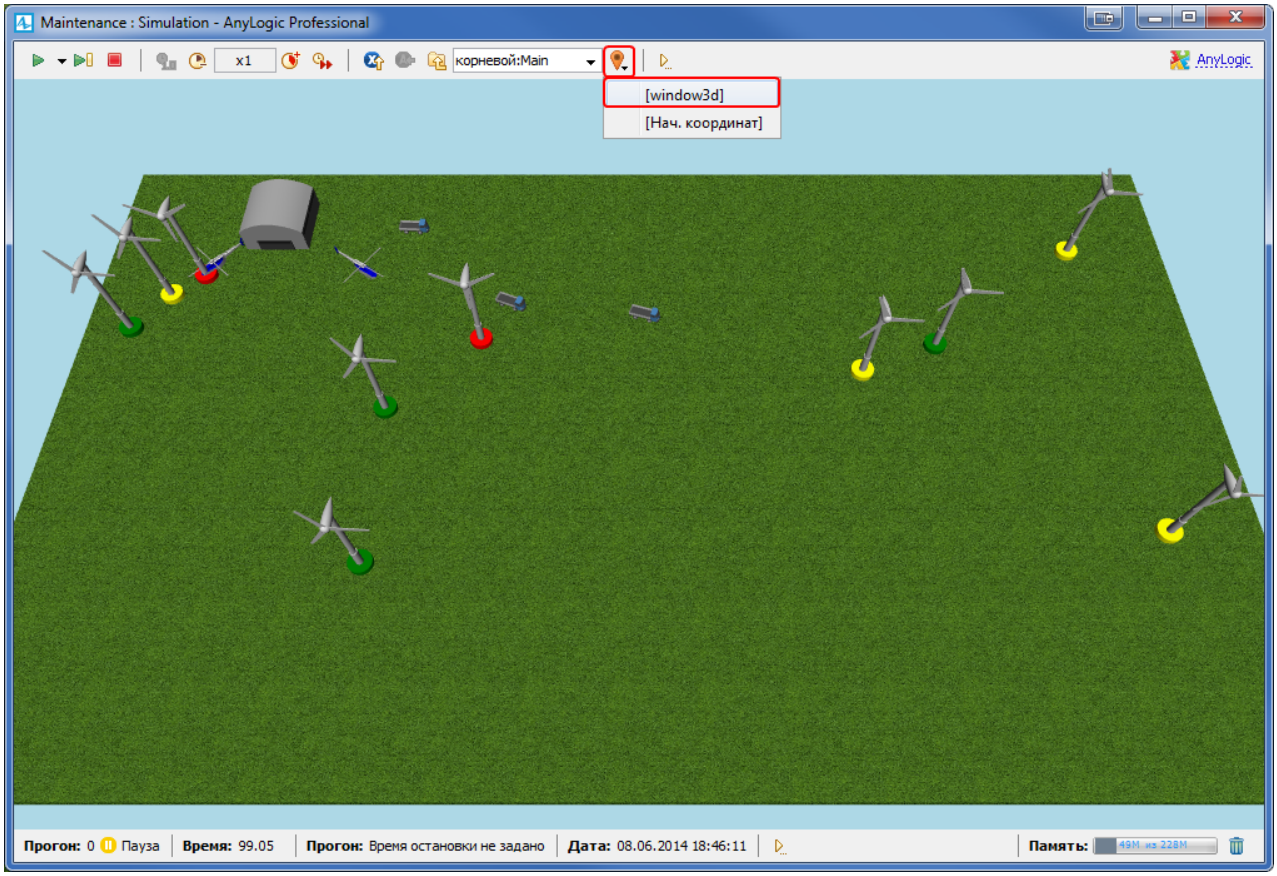
4. Вы также можете щелкнуть фигуру анимации агента `MC` *hangar* и выбрать **Порядок > На передний план**, чтобы "спрятать" транспорт за ней.
5. Чтобы задействовать 3D анимацию, откройте палитру **Презентация** и перетащите элемент **3D Окно** в графический редактор. Затем вы можете изменить его **Свойства**: размер или цвет.

Запустите модель

1. Щелкните кнопку панели управления **Запуск** и запустите модель.



2. Затем щелкните кнопку **Показать область...** и выберите **[window3d]**.

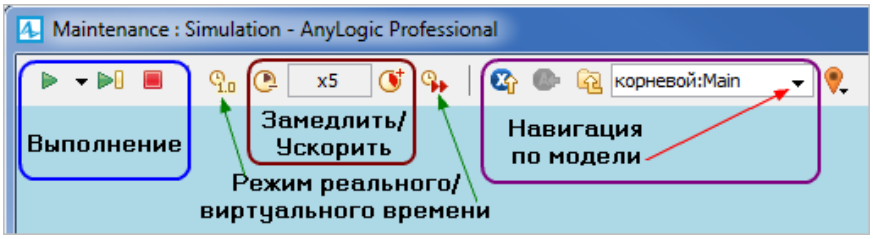


3. Вы можете перемещаться по 3D сцене с помощью мыши и следующих клавиш:

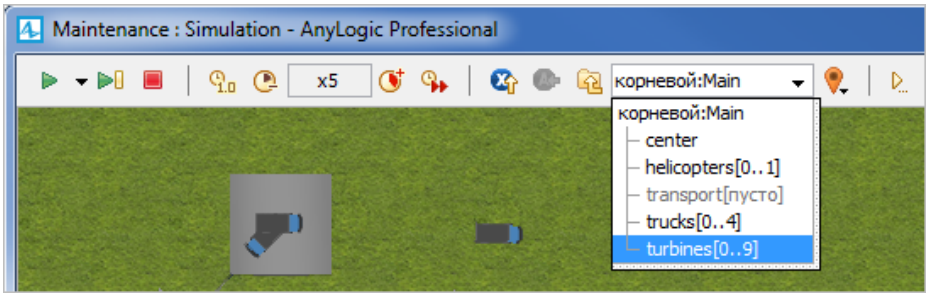
Чтобы	**Выполните следующие действия**
Переместить сцену	1. Нажмите левую кнопку мыши в области 3D окна и держите ее нажатой. 2. Передвиньте мышь в направлении перемещения.
Повернуть сцену	1. Нажмите клавишу `Alt` и держите ее нажатой.

	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.

4. Вы можете управлять запуском модели с помощью кнопок панели управления: **Приостановить** и **Прекратить выполнение** запуска, **Ускорить** или **Замедлить** модель. Кроме того, вы можете следить за разными типами агентов с помощью кнопок навигации по структуре модели.



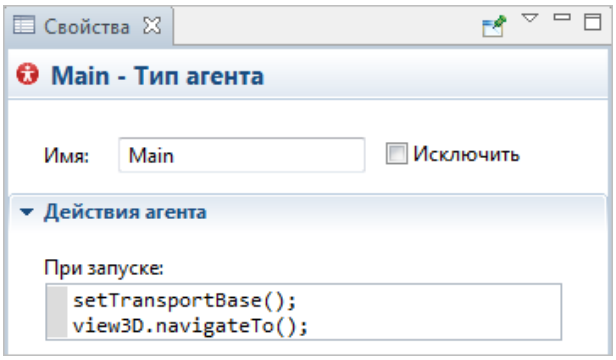
5. Щелкните стрелку вниз в навигации по модели, чтобы открыть список агентов. Рядом с именами популяций агентов вы увидите их количество в квадратных скобках **[..]**. Выберите, например, **turbines [0..9]**. Нумерация начинается с 0 - у нас десять турбин в модели.



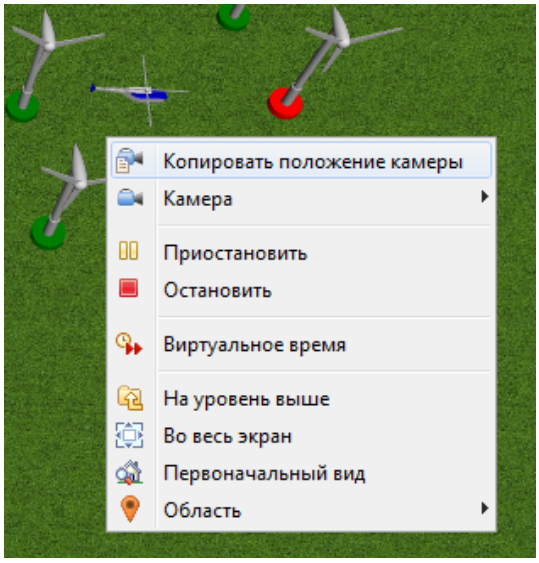
6. Вы увидите тип агента `Turbine` и процессы, происходящие на нем. Вы можете проверять состояние разных турбин соответственно их порядковому номеру. Тогда турбина появится в пространстве, будет выделено ее текущее состояние, а все элементы типа агента будут отображать релевантные для нее показатели. Таким образом вы можете исследовать поведение любого типа агента или конкретного агента в модели.

Установите отображение 3D анимации по умолчанию

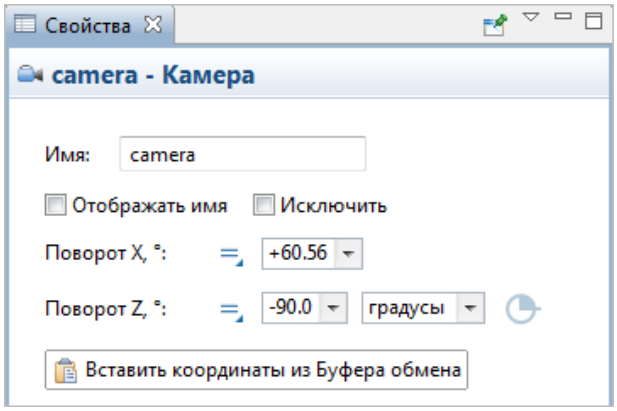
1. Создайте область просмотра в 3D окне. Для этого перетащите элемент **Область просмотра** из палитры **Презентация** и поместите его в верхний левый угол 3D окна.
2. Назовите область просмотра *view3D*. Нам не нужно изменять еще какие-либо ее свойства.
3. Для того, чтобы 3D анимация автоматически загружалась при запуске модели, измените свойство типа агента `Main` **При запуске**. Добавьте функцию перехода в нужную нам область просмотра:



4. Если вы теперь запустите модель, то сразу же увидите 3D анимацию.
5. Также, вы можете добавить элемент **Камера** к 3D окну, чтобы зафиксировать определенное положение вида.
6. Перетащите объект **Камера** из палитры **Презентация**, затем выберите эту камеру в свойстве 3D окна **Камера** и запустите модель. Перемещайтесь и вращайте сцену, пока не получите желаемый угол обзора. Тогда щелкните полотно анимации правой кнопкой мыши и выберите **Копировать положение камеры** в контекстном меню.



7. Вернитесь обратно к разработке модели и щелкните кнопку **Вставить координаты из Буфера обмена** в свойствах камеры.



8. Снова запустите модель. Вы увидите, что теперь автоматически загружается выбранный вами вид 3D.

И, пожалуй, последний штрих. Как вы можете видеть, сейчас и грузовики и вертолеты паркуются прямо в точке расположения турбины, что, конечно, придает анимации некую незавершенность.

Мы можем поправить и этот момент, слегка сдвинув анимацию турбины от ее логической точки местоположения. Для этого откройте диаграму турбины `Turbine` и переместите анимацию турбины чуть вверх (например, чтобы лопасти стали располагаться прямо по оси X).

Запустите модель еще раз. Создание нашей модели завершено, и теперь вы можете добавить в эту модель необходимую аналитику и провести эксперимент оптимизации.

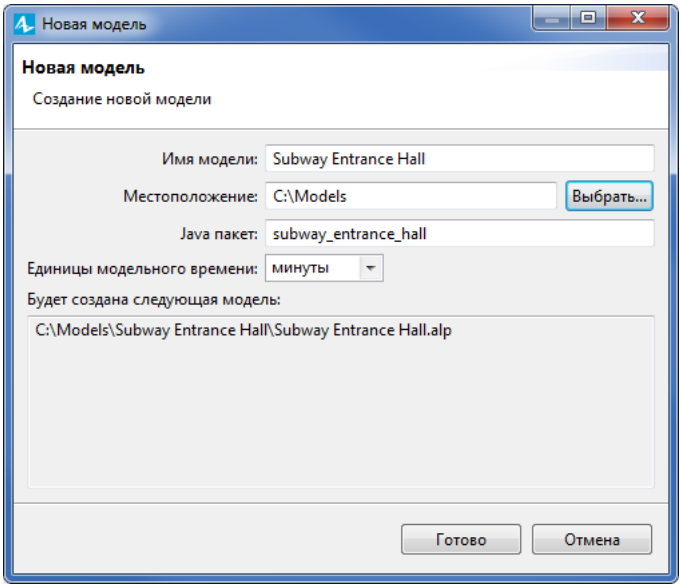
Шаг 1. Моделирование простого потока пассажиров

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

Создание новой модели

Создайте новую модель

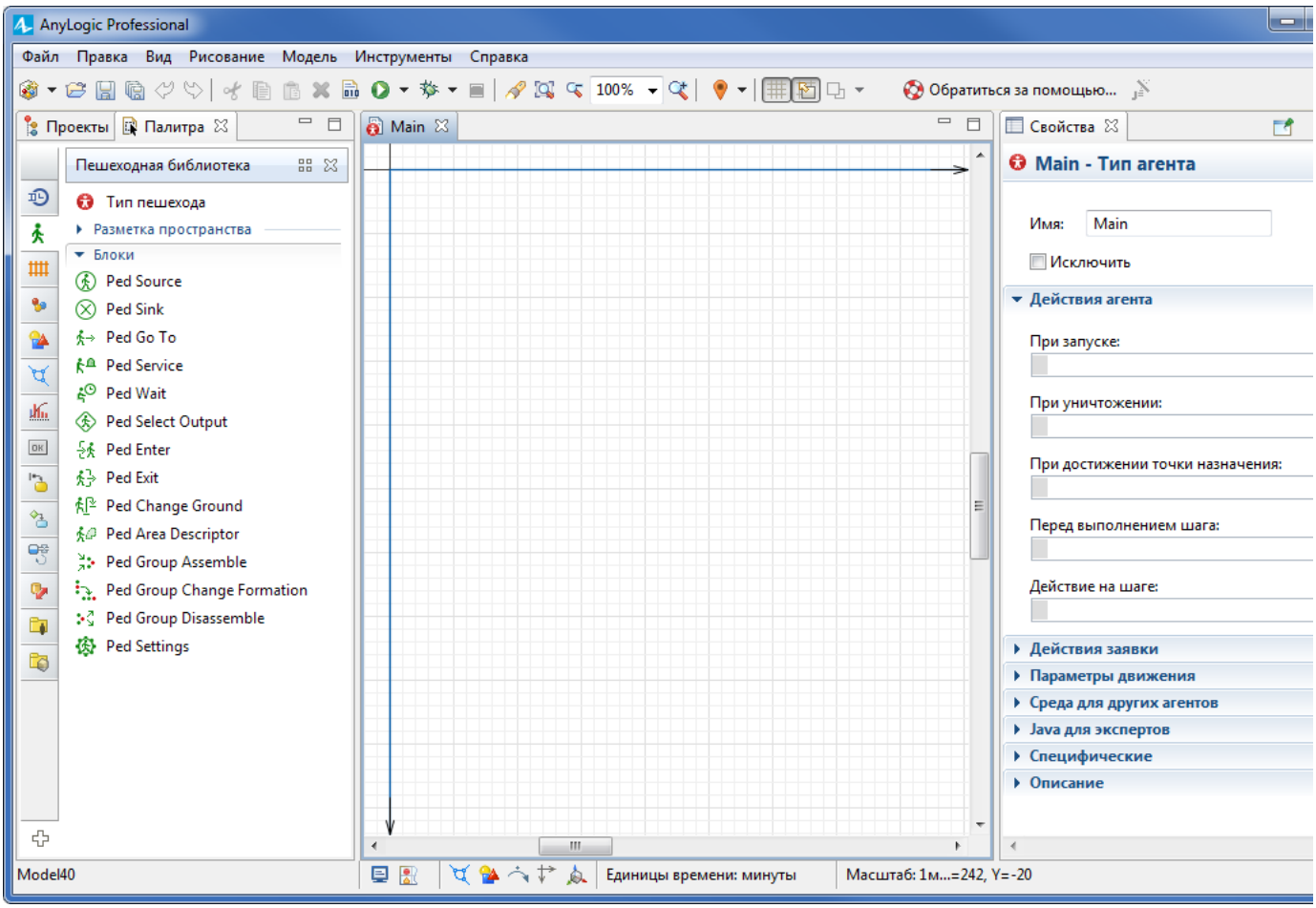
1. Щелкните мышью по кнопке панели инструментов **Создать** . Появится окно **Мастера создания модели**.



2. Задайте имя новой модели. В поле **Имя модели** введите `Subway Entrance Hall`.
3. 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**.
4. Выберите **минуты** в качестве **Единиц модельного времени**.
5. Щелкните **Готово**, чтобы закончить создание модели.

Вы 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 легкую 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

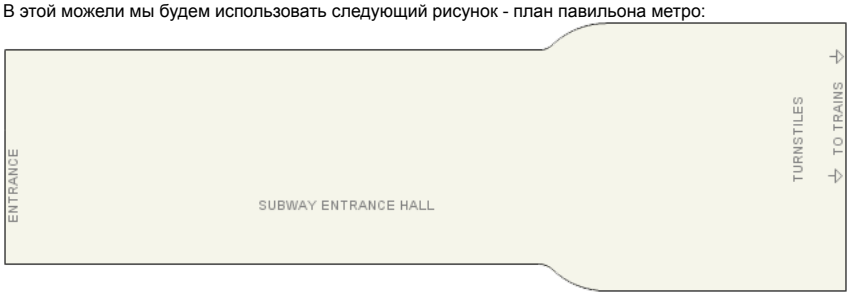
элементы, которые могут быть добавлены на диаграмму типа агента или эксперимента.

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 можели мы будем использовать следующий рисунок - план павильона метро:

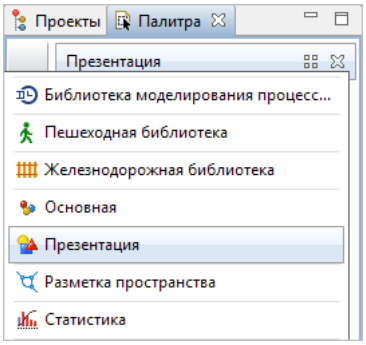


Сохраните рисунок - план на ваш компьютер

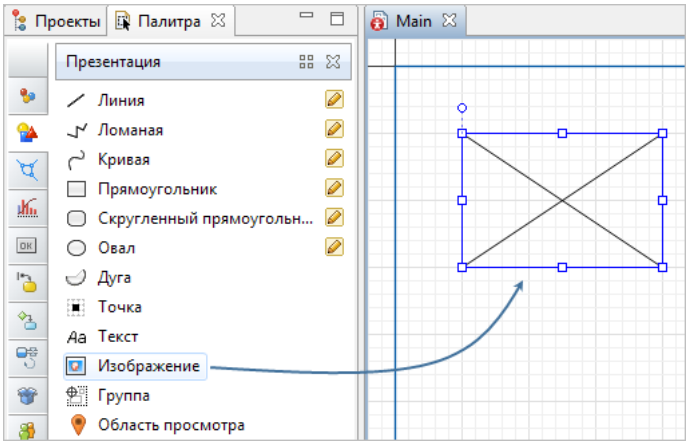
1. Щелкните правой кнопкой мыши по рисунку выше и выберите **Сохранить изображение как...** из контекстного меню. В открывшемся диалоговом окне выберите, куда вы хотите сохранить файл изображения.

Добавьте рисунок с изображением плана павильона в модель

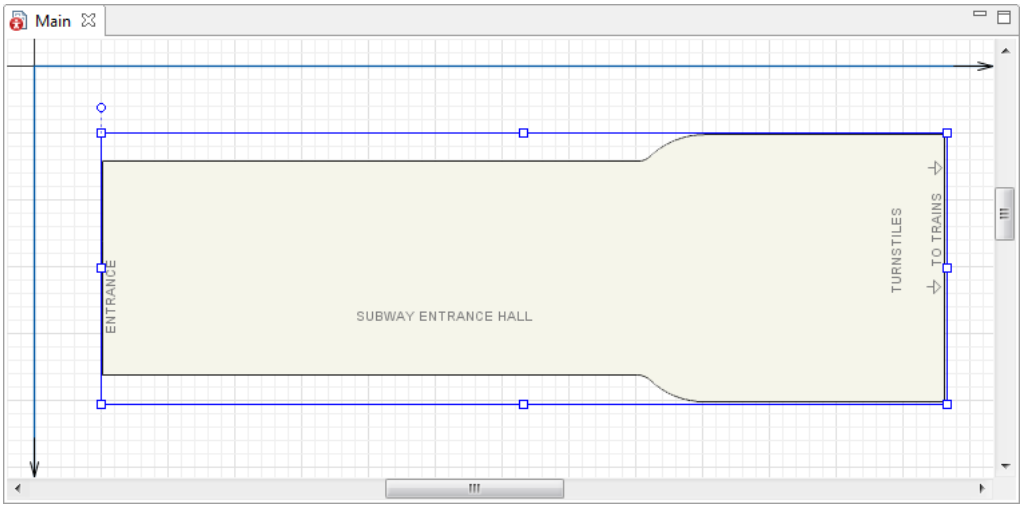
1. Вначале откройте палитру **Презентация**. Эта палитра содержит элементы, которые вы можете использовать для рисования анимации модели.
2. Чтобы открыть какую-либо закладку панели **Палитра** (именуемую в дальнейшем палитрой), нужно щелкнуть мышью по соответствующей иконке на вертикальной панели слева 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.



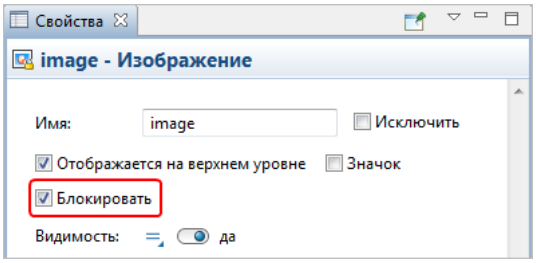
3. Итак, вы открыли палитру **Презентация.** Перетащите элемент **Изображение** из палитры **Презентация** на графическую диаграмму Main:



4. 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.
5. Изображение будет выглядеть в графическом редакторе следующим образом:



6. 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 нее блокировку. Мы делаем так потому, что мы будем рисовать другие фигуры поверх этого изображения, и поэтому мы хотим исключить возможность случайного редактирования изображения при рисовании этих фигур.



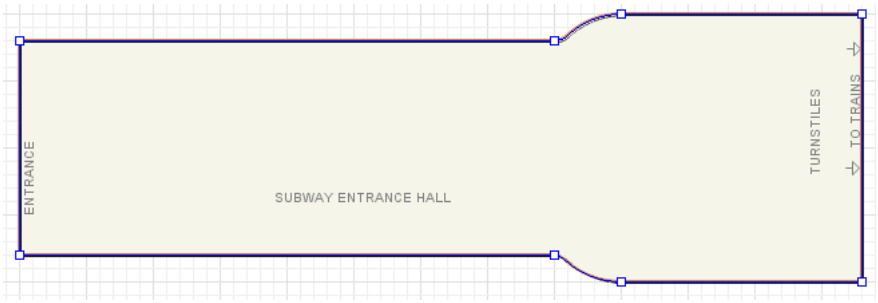
Только некоторые зоны отмечены на этом рисунке. Нам хотелось бы поэкспериментировать с разными диаграммами, и на данный момент нам не известно, где именно располагаются кассы и автоматы по продаже билетов. Поэтому на рисунке отмечены не все области.

Рисование границ здания

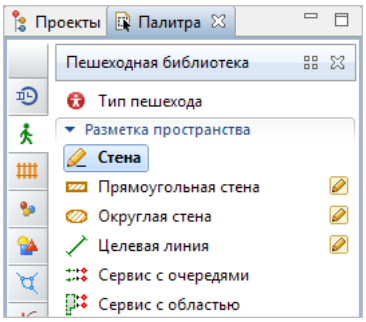
Теперь мы нарисуем на анимации объекты моделируемой среды. Вначале мы нарисуем границу моделируемого нами пространства, играющую роль стен здания.

Нарисуйте границы здания

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



3. 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.



4. Последовательно щелкайте мышью в тех точках диаграммы, куда вы хотите поместить углы стены. Каждый щелчок добавляет часть линии той стены, которую вы рисуете.
5. Чтобы добавить кривую линию, щелкните левой кнопкой мыши в точку конца кривой линии и двигайте указатель мыши, удерживая кнопку. Пока вы ведете указатель мыши, вы заметите, как изменяется радиус кривизны. Чтобы нарисовать окружность, двигайте указателем ровно вдоль сетки координат. Отпустите левую кнопку мыши, когда рисунок готов.
6. Чтобы завершить рисование, добавьте последнюю точку стены двойным щелчком мыши.

Рисование входа и цели движения для потока пешеходов

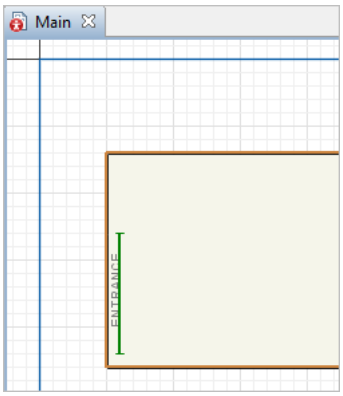
Теперь нужно задать области входа и выхода пешеходов.

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

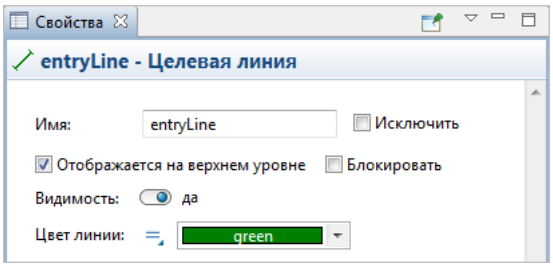
Мы хотим нарисовать линию входа точно там, где на рисунке вход обозначен текстом - ENTRANCE:

Нарисуйте линию, где появляются пассажиры

1. Перетащите элемент **Целевая линия** из секции **Разметка** палитры **Пешеходная библиотека** в графический редактор.
2. Измените ее размер и расположите точно так, как показано на рисунке:



3. Назовите линию `entryLine`.

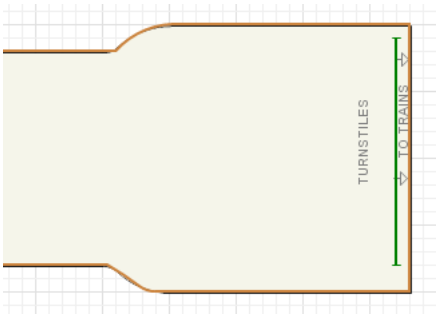


Теперь давайте добавим еще одну целевую линию, которая определяет место, куда движутся пассажиры после того, как они зашли в павильон метро.

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

Нарисуйте целевую линию

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



Обратите внимание на то, что все элементы разметки (целевые линии и др.) должны находиться внутри стены.

Создание диаграммы, задающей поток пешеходов

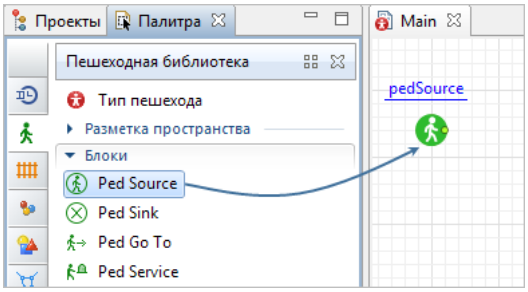
Теперь мы закончим создание модели, моделирующей простейший пассажиропоток, создав диаграмму моделируемого нами процесса из блоков **Пешеходной библиотеки**.

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`).

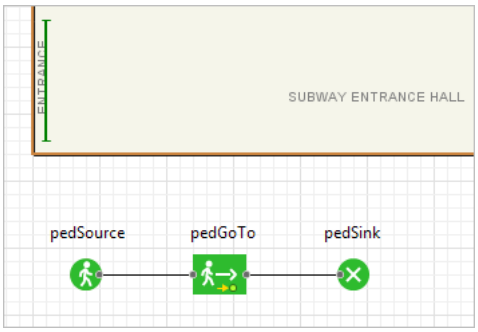
Диаграмма процесса в AnyLogic создается путем добавления объектов библиотеки из палитры на диаграмму типа агентов, соединения их портов и изменения значений свойств объектов в соответствии с требованиями вашей модели.

Создайте диаграмму процесса

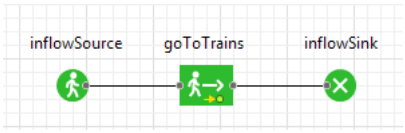
1. Добавьте объекты **Пешеходной библиотеки** на диаграмму и соедините их так, как показано на рисунке. Чтобы добавить на диаграмму объект Пешеходной библиотеки, нужно открыть в панели **Палитра** палитру этой библиотеки, щелкнув мышью по панели с ее заголовком, а затем перетащить нужный вам объект из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



2. Пока вы перетаскиваете блоки и располагаете их друг рядом с другом, вы можете видеть, 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.



3. Переименуйте блоки. Назовите их *inflowSource*, *goToTrains*, *inflowSink*. (вы можете это сделать в свойствах блока.)



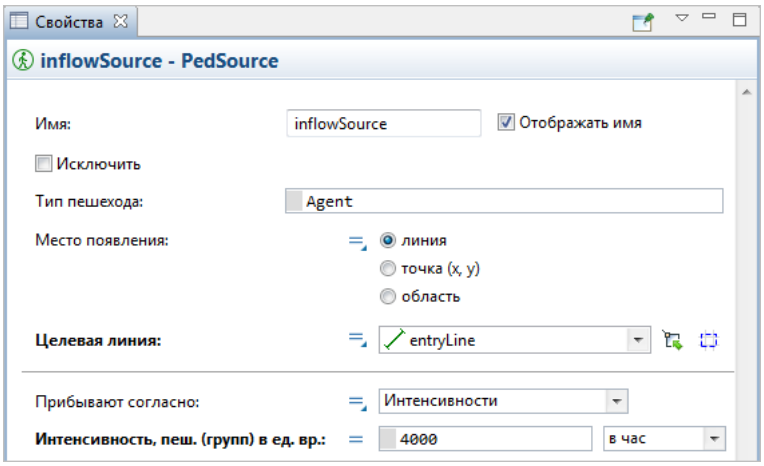
Скажем несколько слов об этих блоках диаграммы.

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

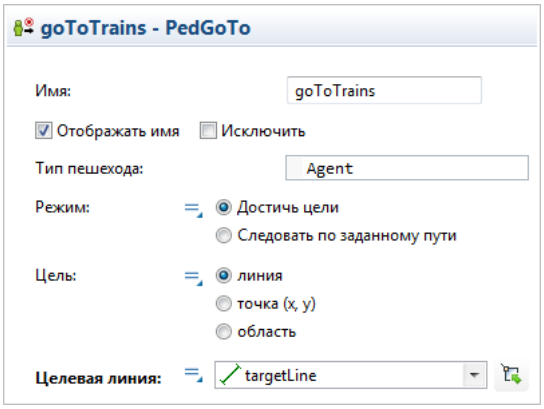
Детальное описание **объектов Пешеходной библиотеки**, их функций и параметров вы можете найти в *Справочном руководстве по Пешеходной библиотеке*.

Измените свойства блоков диаграммы

1. 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**.
2. Задайте `4000 в час` в параметре **Интенсивность**.



3. 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**.

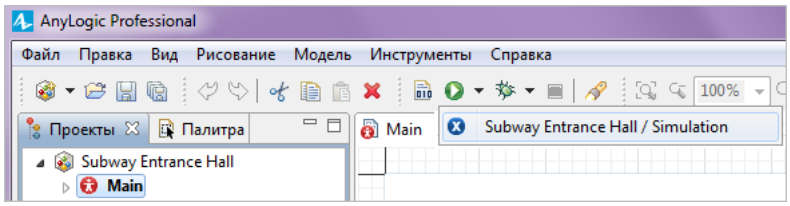


4. Оставьте все свойства объекта **PedSink** установленными по умолчанию.

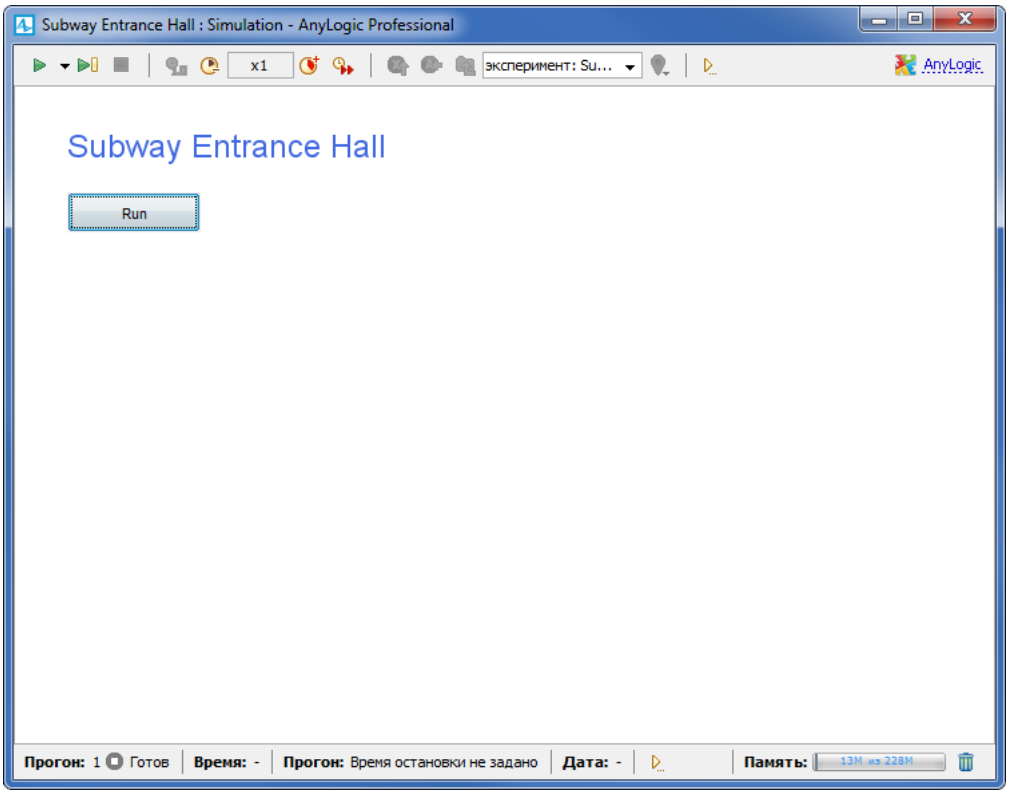
Итак, готово! Давайте запустим нашу модель.

Запустите модель

1. 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`.

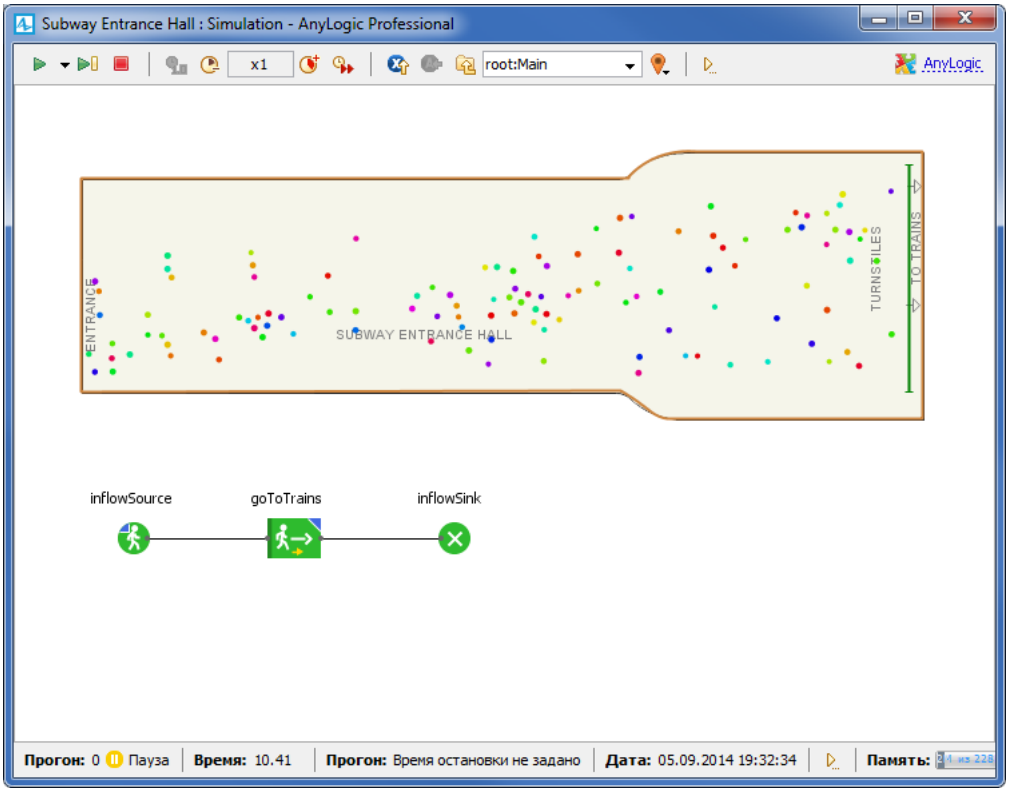


Запустив модель, вы увидите окно презентации этой модели. В нем будет отображена презентация запущенного эксперимента. AnyLogic автоматически помещает на презентацию каждого простого эксперимента заголовок и кнопку, позволяющую запустить модель и перейти на презентацию. Позднее вы можете добавить на эту страницу изображения, описание модели и т.д.



2. Щелкните по кнопке **Запустить**. Модель запустится и вы сможете понаблюдать за динамикой моделируемого процесса с помощью нарисованной вами презентации на диаграмме типа агента *Main*.

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


Subway Entrance Hall : Simulation - AnyLogic Professional
x1
root:Main
AnyLogic
ENTRANCE
SUBWAY ENTRANCE HALL
TURNSTILES
TO TRAINS
inflowSource
goToTrains
inflowSink
Прогон: 0 Пауза
Время: 10.41
Прогон: Время остановки не задано
Дата: 05.09.2014 19:32:34
Память:

Шаг 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.

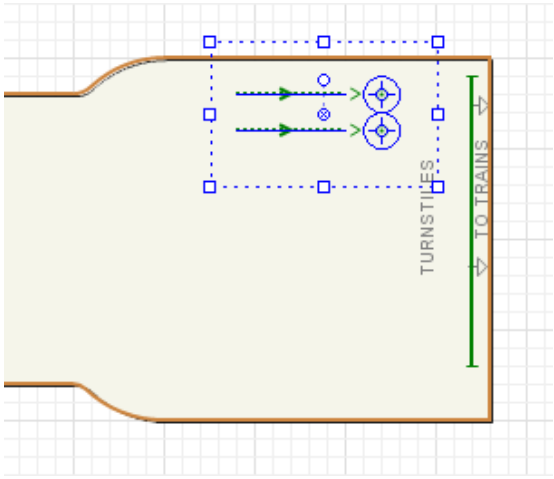
Имеется два типа элементов разметки пространства, которые вы можете использовать, чтобы добавить сервисы в вашу модель:

> **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.

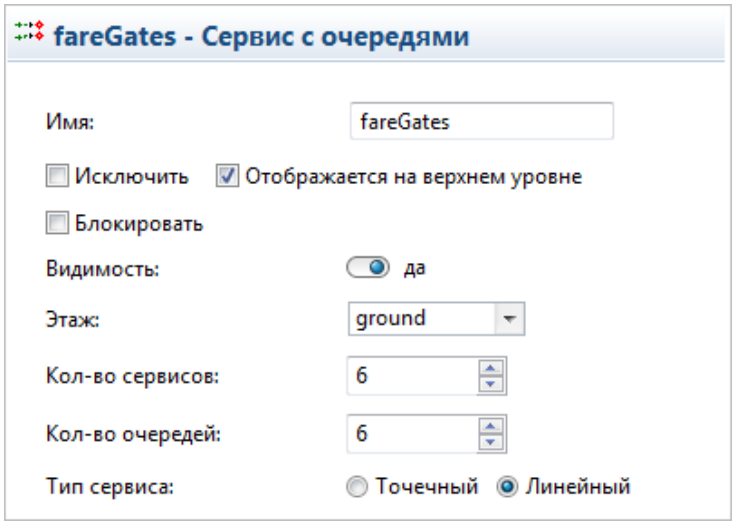
Турникеты обычно моделируются элементом **Сервис с очередями**.

Нарисуйте турникеты

1. 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:



2. Настройте сервисы. Назовите их *fareGates*.
3. Очевидно, что двух турникетов недостаточно. Увеличьте значение параметра **Количество сервисов** до 6. Соответственно, увеличьте значение параметра **Количество очередей** также до 6.
4. Измените значение свойства **Тип сервиса** с **Точечный** на **Линейный**.

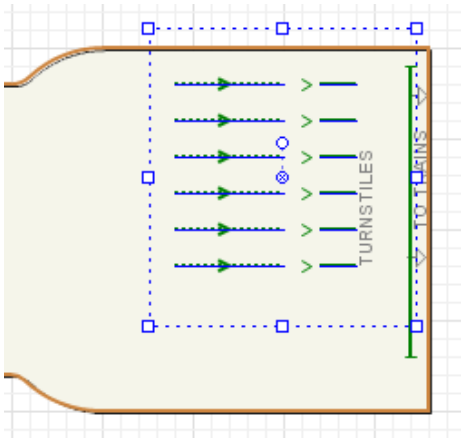


Точечные и линейные сервисы

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:

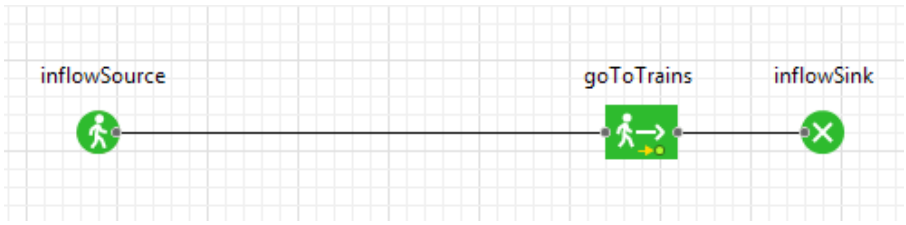


Изменение диаграммы процесса

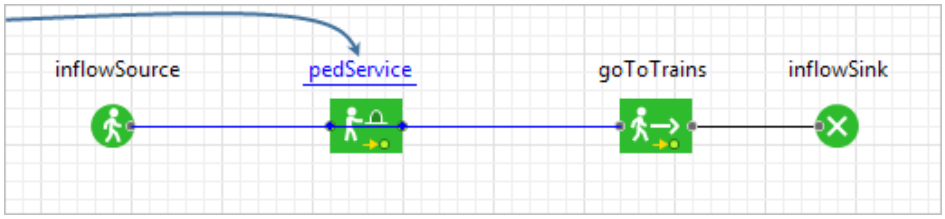
Теперь мы внесем небольшие изменения в диаграмму процесса.

Измените диаграмму процесса

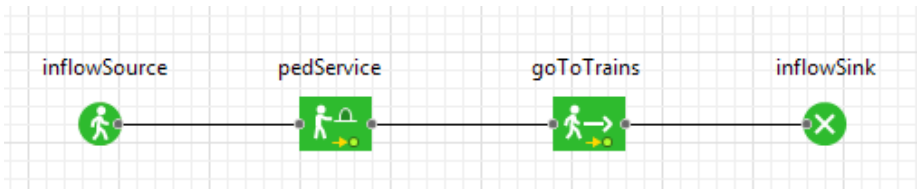
1. Освободите место для нового блока на нашей диаграмме. Сдвиньте блоки *goToTrains* и *inflowSink* вправо, как показано на рисунке:



2. Теперь мы можем 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).



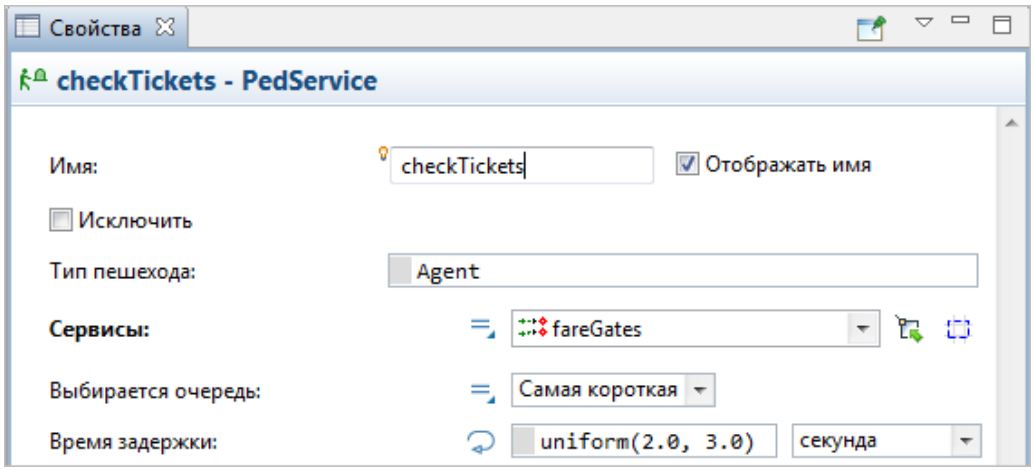
3. Объект появится в диаграмме.



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 же вам придется проверить, обе ли конечные точки соединителя были помещены точно в порты, и если нет, то передвиньте их туда.

Измените блоки диаграммы

1. Откройте панель **Свойства** блока *pedService*.
2. Измените название блока на *checkTickets*.
3. Выберите `fareGates` (название нашего элемента разметки **Сервис с очередями**) в поле **Сервисы**.
4. Вы увидите, 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.



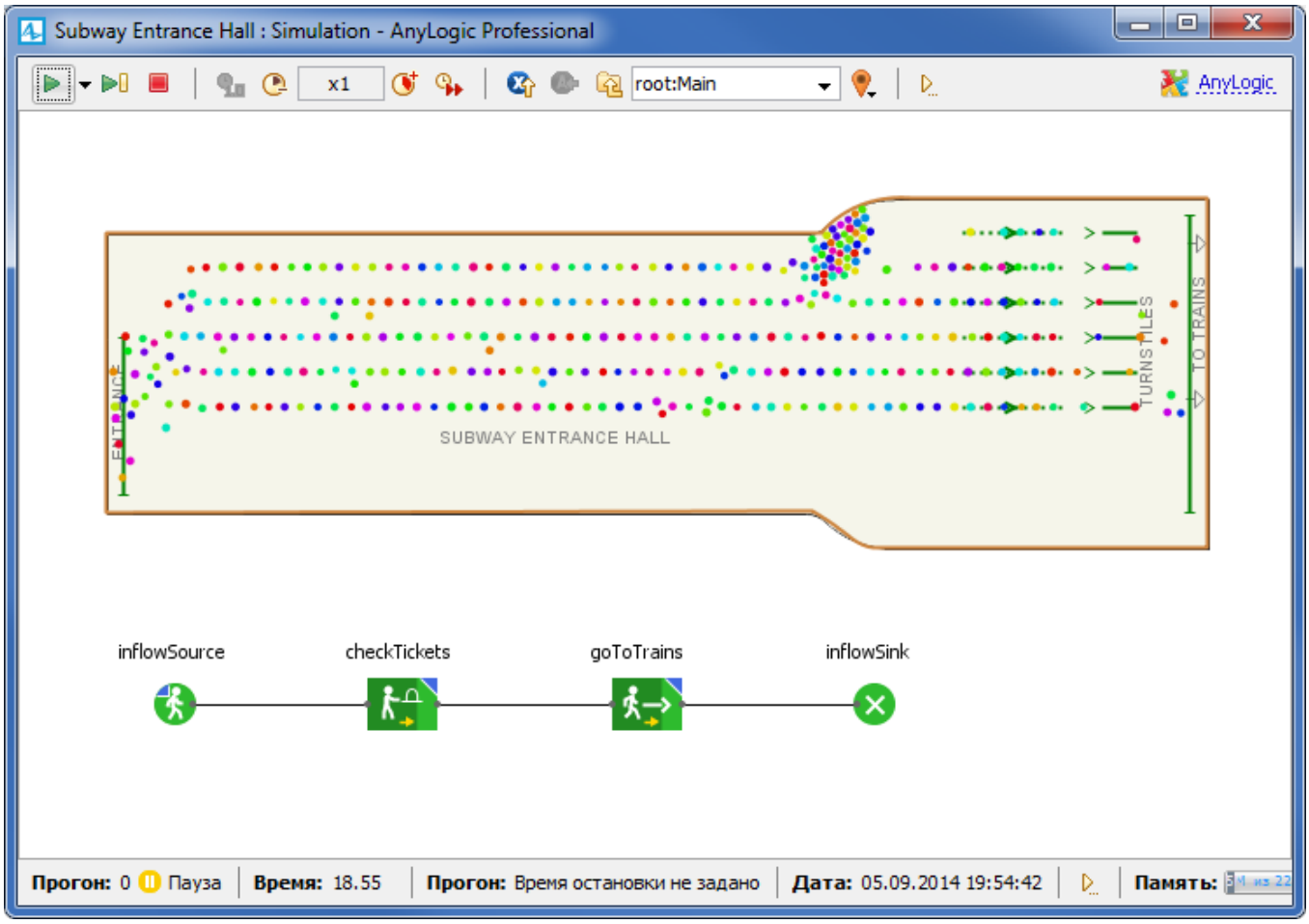
Мы задали новую логику и теперь можем запустить модель и наблюдать за динамикой моделируемого процесса.

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. Запустив модель, вы автоматически обновляете существующую версию.

Запустите модель

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



6 турникетов. моментально образуются огромные очереди.

2. 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, вам 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 **Реальное/ виртуальное время**

.

Итак, 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.

Наконец наша конфигурация приемлема: станция успешно справляется с таким потоком пешеходов.

8 турникетов. Очереди в пределах разумного.

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

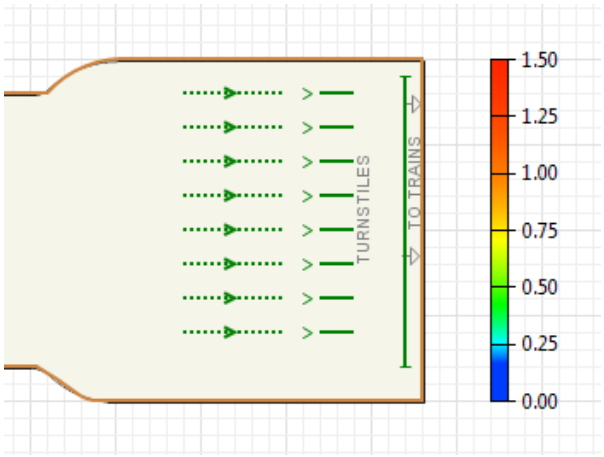
Сохраните изменения в модели, щелкнув кнопку панели инструментов **Сохранить** (продолжайте в дальнейшем сохранять изменения время от времени). Теперь мы можем продолжить разрабатывать нашу модель дальше.

Шаг 3. Отображение карты плотности пешеходов

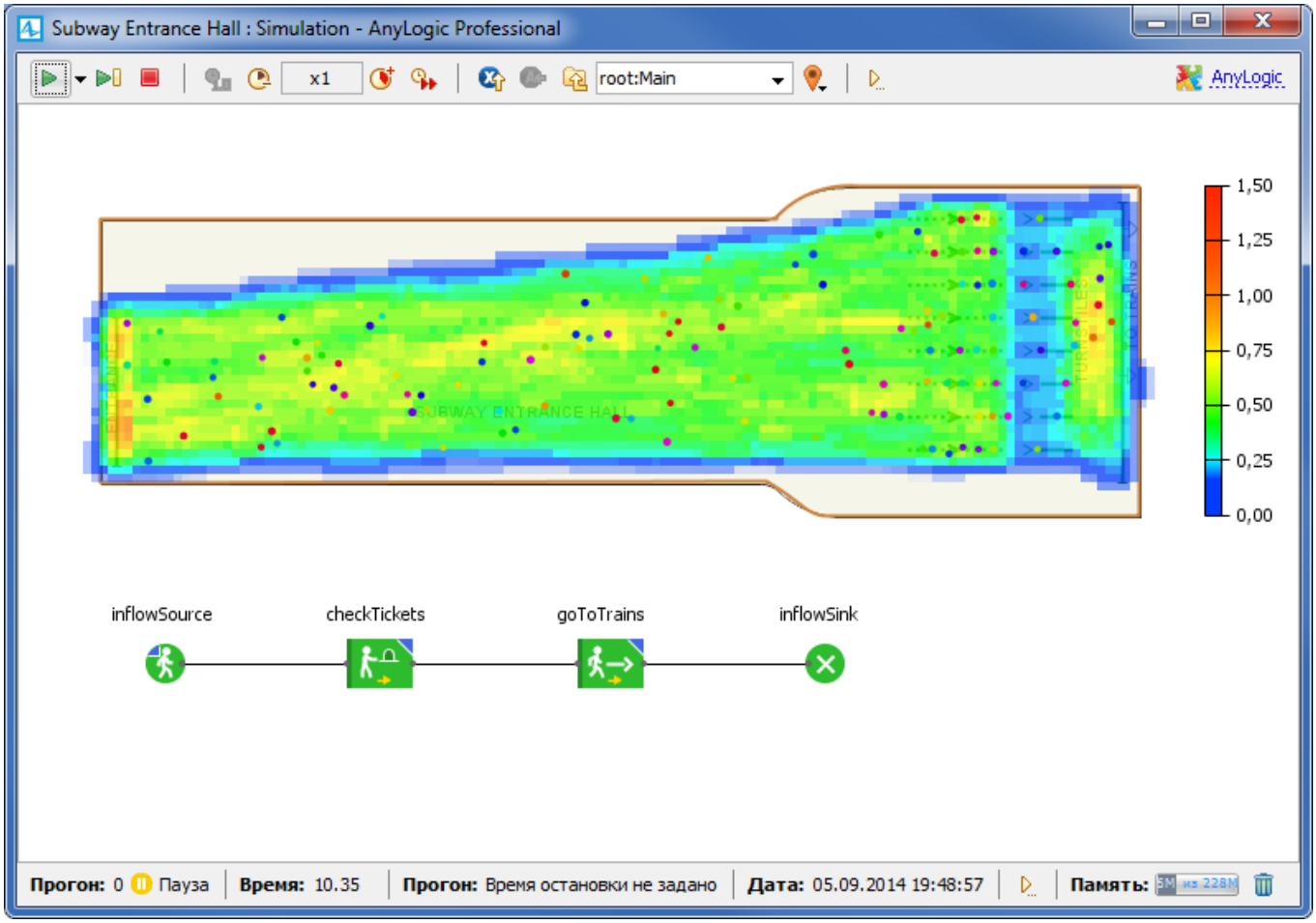
Мы создали модель динамики пешеходного потока в здании павильона метро. Теперь нам хотелось бы получить статистические данные этого потока. Самым значительным инструментом в моделировании пешеходов является **Карта плотности пешеходов**.

Отобразите карту плотности пешеходов

1. Перетащите элемент **Карта плотности пешеходов** 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.
2. Откройте панель **Свойства** этого блока и сбросьте флажок **Отображать имя**.
3. Выберите значение *ground* из списка **Этаж**.



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



Карта плотности

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 пешеходов**. Синий цвет используется для участков низкой плотности. При нулевой плотности закрашивание соответствующего участка не производится вообще. Приведенная на рисунке шкала информирует нас о том, что, например, желтый 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.

Вы можете 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 пешеходов**.

Если вас не устраивает плотность карты на чертеже тем, что его практически не видно, вы можете увеличить степень прозрачности отображаемой на анимации карты плотности с помощью бегунка **Прозрачность** в свойствах элемента **Карта плотности пешеходов**.

Вы можете отключить отображение карты плотности, но при этом продолжать собирать соответствующую статистику, без отображения карты при анимации модели. Для этого сбросьте флажок **Показывать карту плотности на анимации**. В таком случае вы можете увеличить скорость выполнения модели. Эта статистика может храниться в наборах данных, откуда, например, по окончании моделирования, записываться в базу данных для последующего статистического анализа.

Итак, мы узнали почти все о карте плотности пешеходов и можем добавлять новые элементы на диаграмму нашей модели.

Шаг 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.

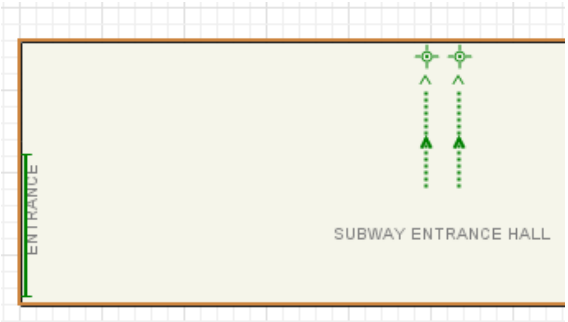
На станции могут находиться различные виды услуг продажи билетов. Небольшие павильоны метро могут быть оборудованы только автоматами по продаже билетов, а большие и просторные станции могут также иметь билетные кассы.

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.

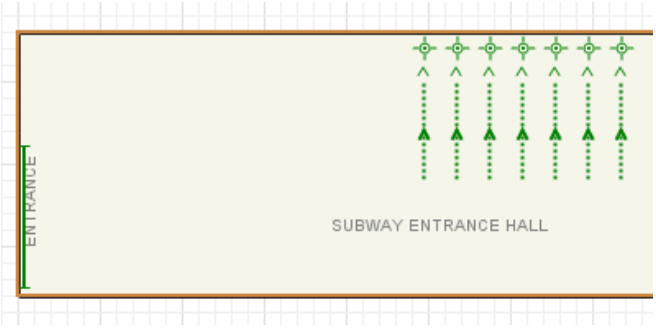
Как и турникеты, автоматы продажи билетов логично моделировать элементом **Сервис с очередями**.

Нарисуйте автоматы продажи билетов

1. Перетащите элемент **Сервис с очередями** из секции **Разметка** палитры **Пешеходная Библиотека** в графический редактор.
2. Повращайте элементы сервиса и разместите их так, как показано на рисунке:



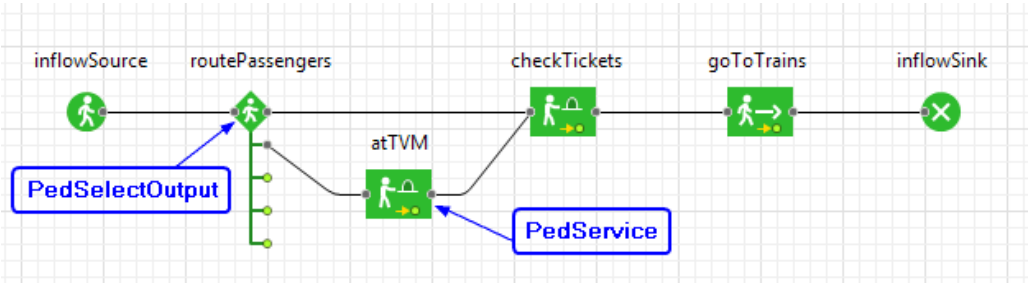
3. Откройте страницу свойств сервисов и настройте эту группу сервисов.
4. В этот раз наши сервисы не линейные, а точечные. Точечные сервисы используются тогда, когда для того, чтобы быть обслуженным, пешеход должен просто подойти к любой точке фигуры, задающей соответствующий сервис и провести там время, заданное как **Время задержки** для этого сервиса. Так что оставьте выбор **Точечный** в свойстве **Тип сервиса**.
5. Назовите сервисы *ticketMachines*.
6. Увеличьте параметр **Количество сервисов** до 7. Соответственно, увеличьте параметр **Количество очередей** так же до 7.



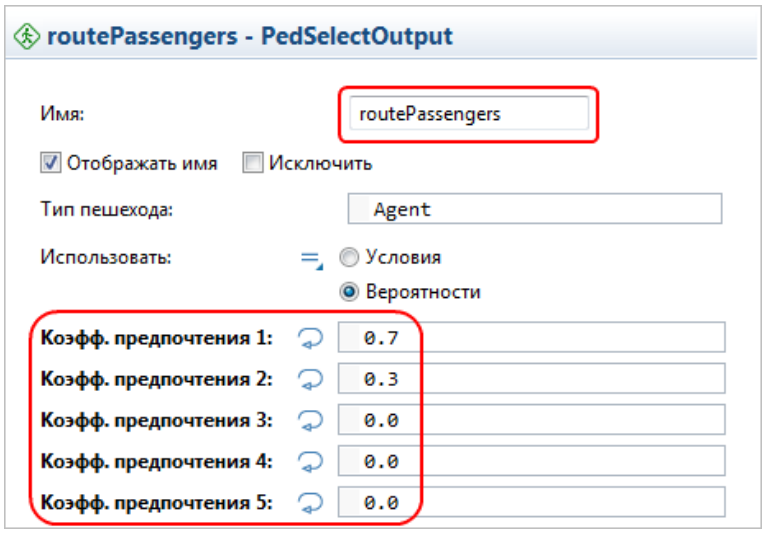
Теперь мы хотим направить некоторых из пассажиров сразу к турникетам, а некоторых - на обслуживание у автоматов продажи билетов.

Измените диаграмму модели

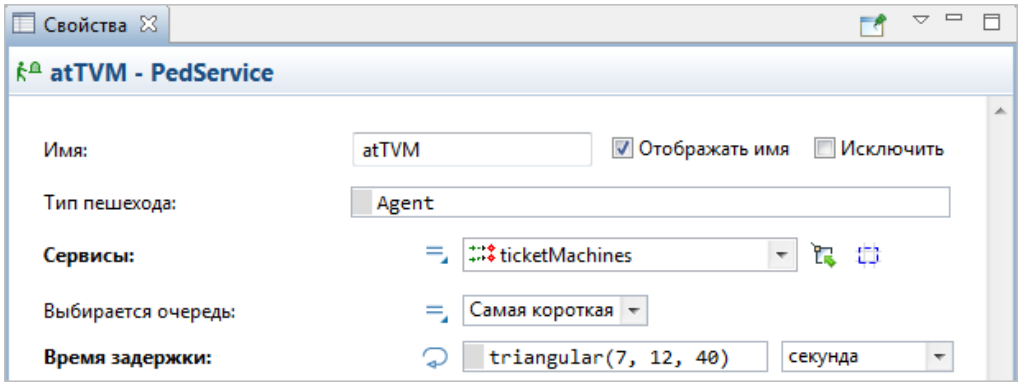
1. Добавьте объект **PedSelectOutput**, чтобы разделить поток пассажиров. Нам нужен этот объект, чтобы перенаправлять пассажиров без билетов к автоматам продажи билетов, а пассажиров с билетами – к турникетам.
   Объект **PedSelectOutput** является блоком принятия решения Пешеходной библиотеки. Пешеход, вошедший в блок **PedSelectOutput**, будет перенаправляться в один из пяти выходных портов в зависимости от заданных для этих портов коэффициентов предпочтения.
2. Добавьте еще один объект **PedService**. Этот блок будет моделировать обслуживание пассажиров у автоматов продажи билетов. Поместите его между объектом **PedSelectOutput** и ранее созданным объектом **PedService** (*checkTickets*).
3. Соедините блоки, как показано на рисунке.



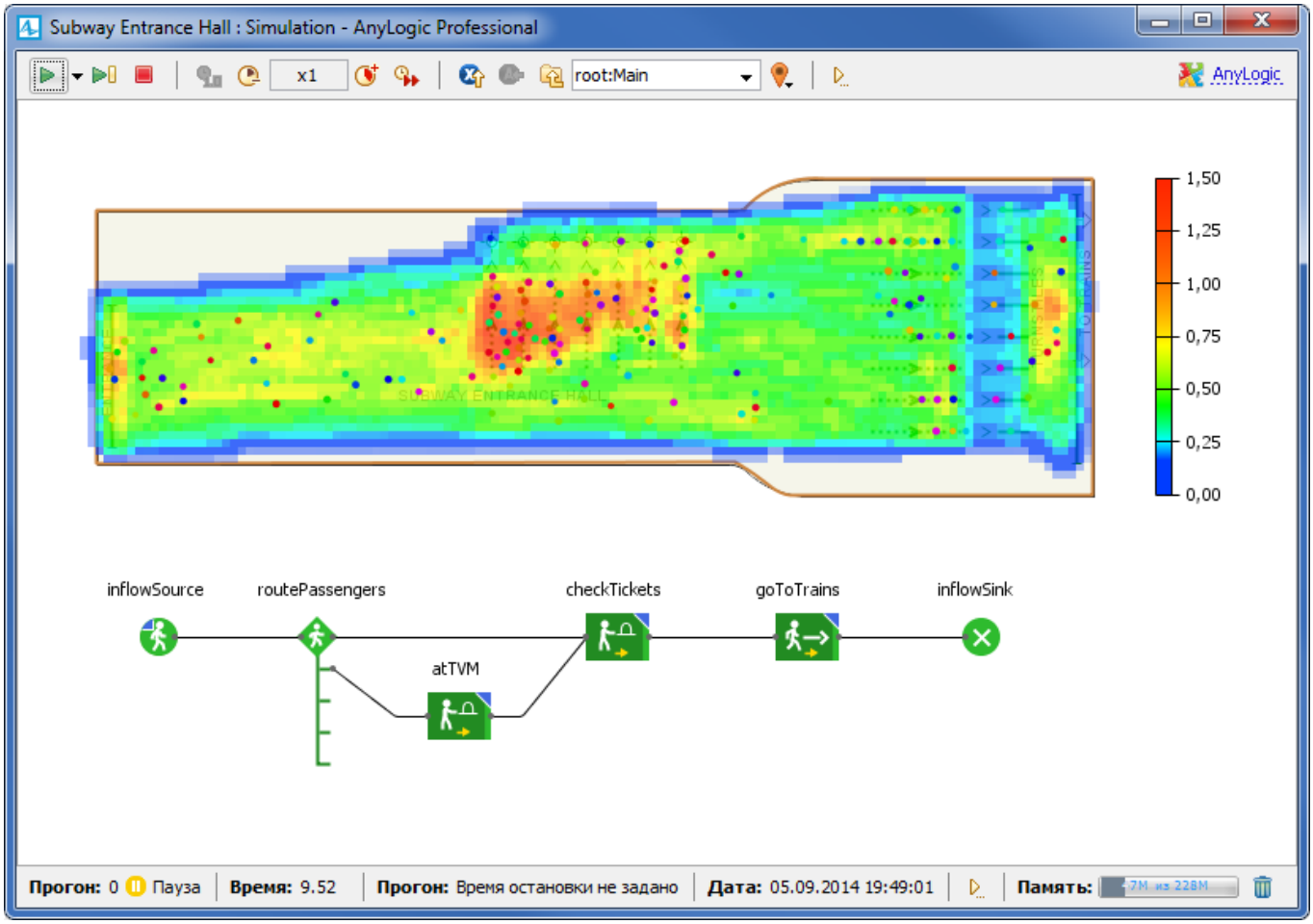
4. Измените свойства объекта **PedSelectOutput**. Назовите его *routePassengers*. Укажите значение *0.7* в поле **Коэфф. предпочтения 1** (коэффициент для потока, направляющегося напрямую к турникетам) и значение *0.3* в поле **Коэфф. предпочтения 2** (коэффициент для потока пассажиров, направляющихся 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*.



5. Настройте только что добавленный объект **PedService**. Переименуйте его как *atTVM*.
6. Выберите *ticketMachines* (название нашего элемента разметки **Сервис с очередями**) в свойстве **Сервисы**.
7. 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. Вы увидите, что теперь некоторые пассажиры перед тем, как пройти к турникетам, сначала подходят к кассам, чтобы приобрести билет.


Subway Entrance Hall : Simulation - AnyLogic Professional
x1
root:Main
AnyLogic
1,50
1,25
1,00
0,75
0,50
0,25
0,00
SUBWAY ENTRANCE HALL
inflowSource
routePassengers
checkTickets
goToTrains
inflowSink
atTVM
Прогон: 0 Пауза
Время: 9.52
Прогон: Время остановки не задано
Дата: 05.09.2014 19:49:01
Память: