Name of College : K.B.P College, Vashi

Name of Department : Computer Science

Student Name : Hemantkumar Shahaji Mohite

Roll No : 199354

Subject : Simulation & Modelling

INDEX

5	Design and develop a discrete-event model that will simulate process by: • Creating a simple model • Adding resources • Creating 3D animation • Modelling delivery [Use a case scenario like a company's manufacturing and shipping].	87		
6	Design and develop time-slice simulation for a scenario like airport model to design how passengers move within a small airport that hosts two airlines, each with their own gate. Passengers arrive at the airport, check in, pass the security checkpoint and then go to the waiting area. After boarding starts, each airline's representatives check their passengers' tickets before they allow them to board.	127		

PRACTICAL NO: 01

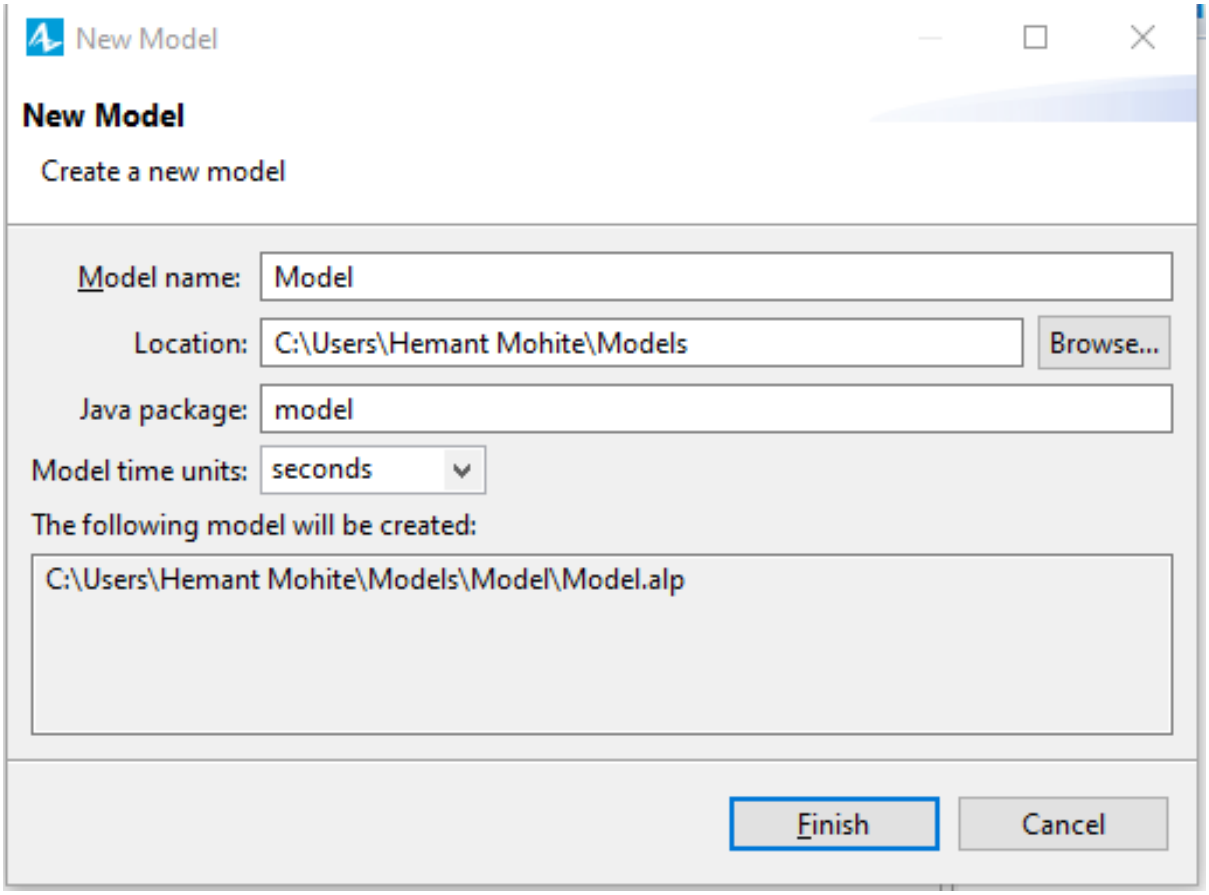
Aim: Design and develop agent based model by

- Creating the agent population
- Defining the agent behavior
- Add a chart to visualize the model output.

[Use a case scenario like grocery store, telephone call centre etc for the purpose].

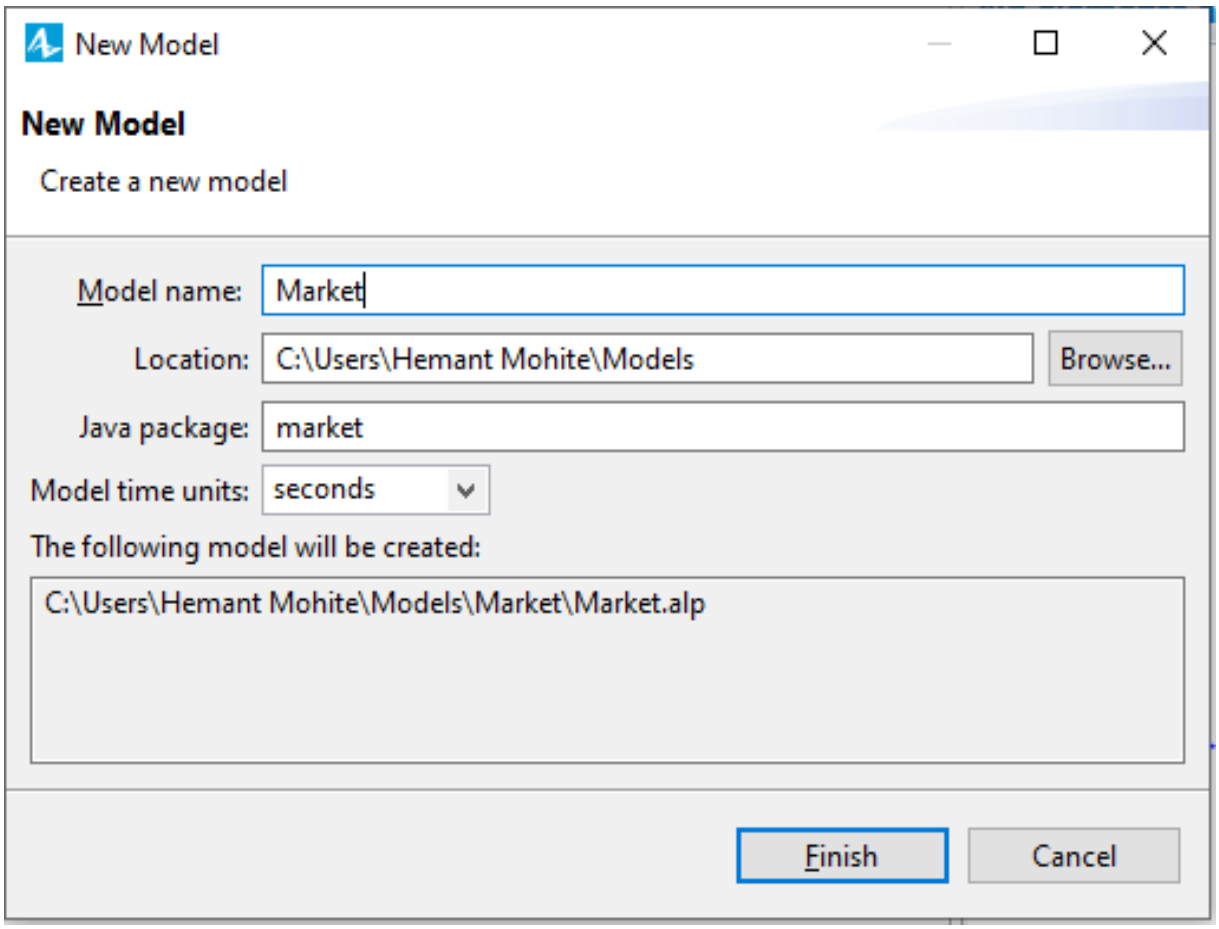
Code:

Close the Welcome page and create a new model by selecting File > New >Model from AnyLogic's main menu. The New Model wizard will open.

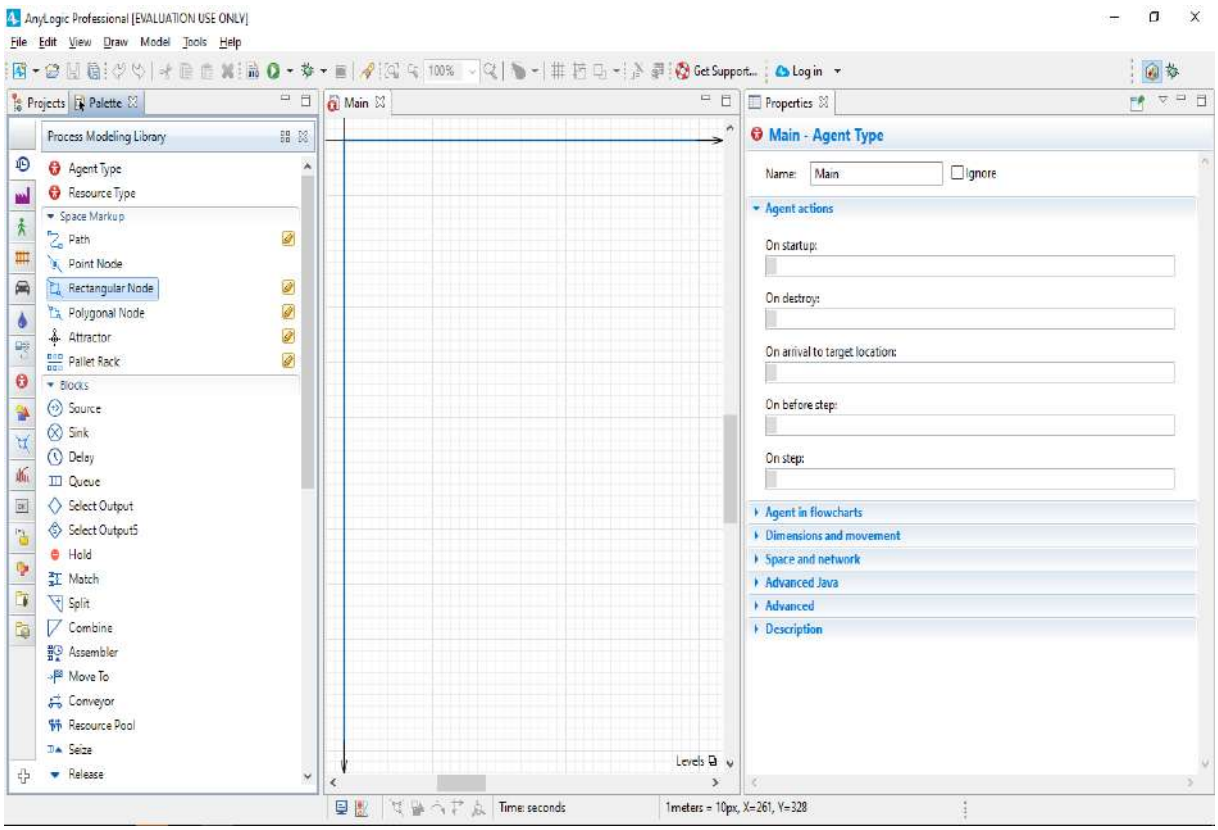


In the Model name box, enter the new model's name: Market.

In the Location box, select the folder where you want to create the model. You can browse for a folder by clicking Browse or type the name of the folder you want to create in the Location box.

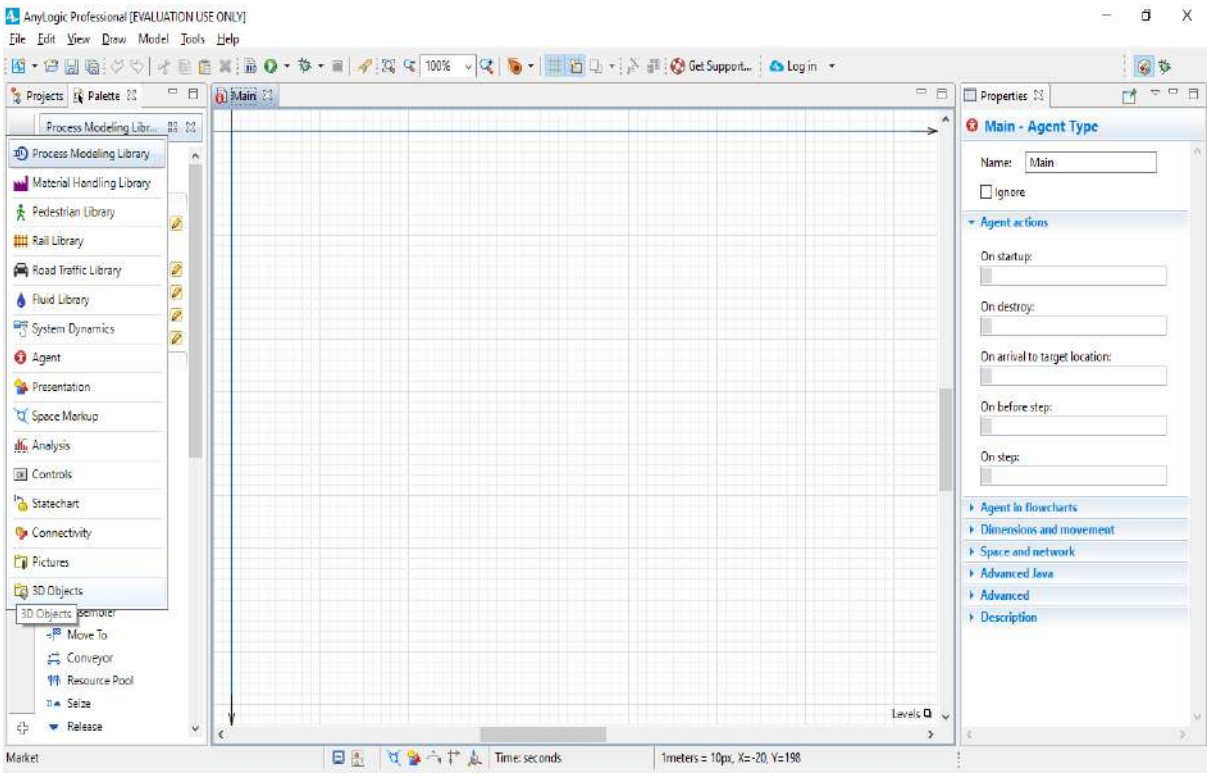


Click Finish.

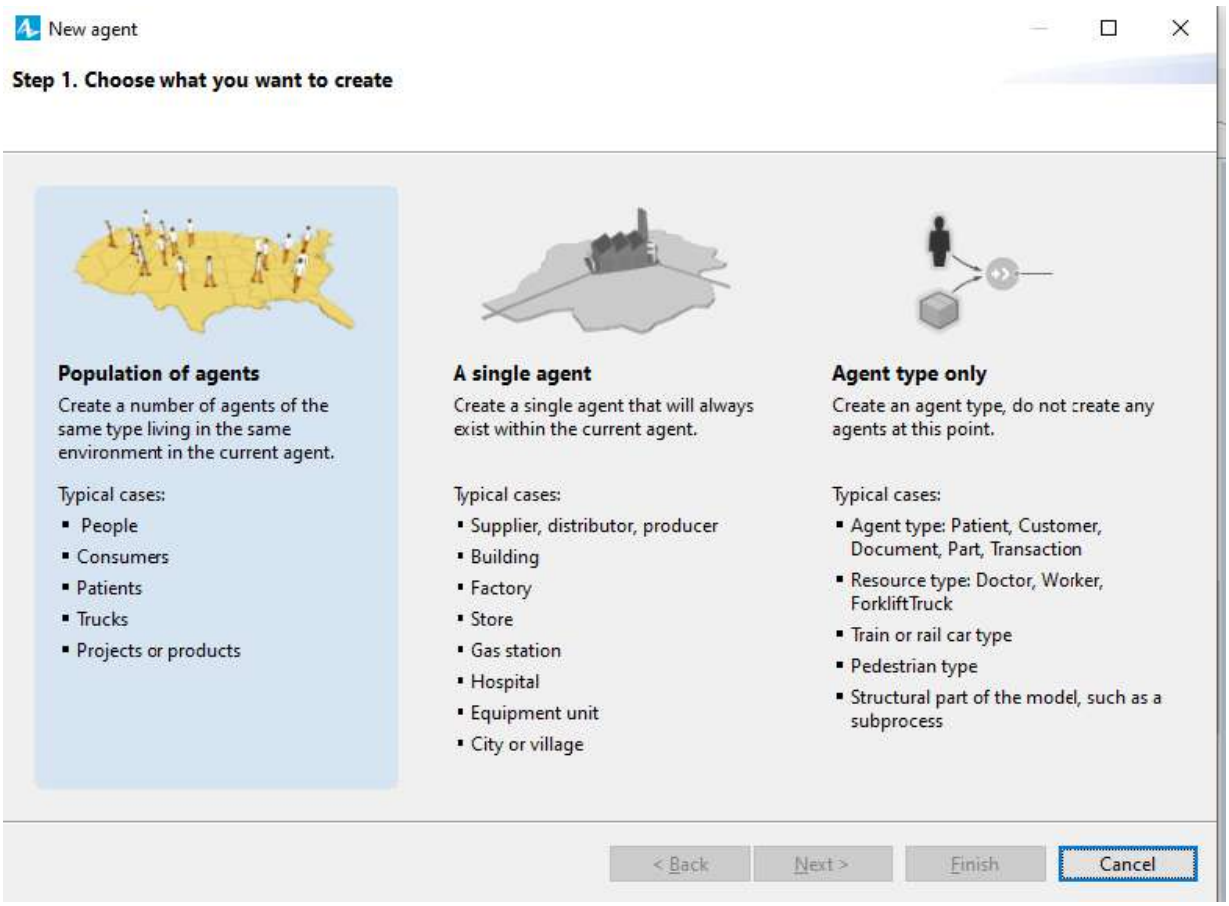


Our model has one agent type, Main. To add consumers, we'll need to create an agent type to represent consumers, and then create an agent population made up of instances of this consumer agent type. In AnyLogic 7, you can use the helpful New agent wizard to create agents.

We want to add a new model element, but we first need to switch to the Palette view by clicking the Palette tab.

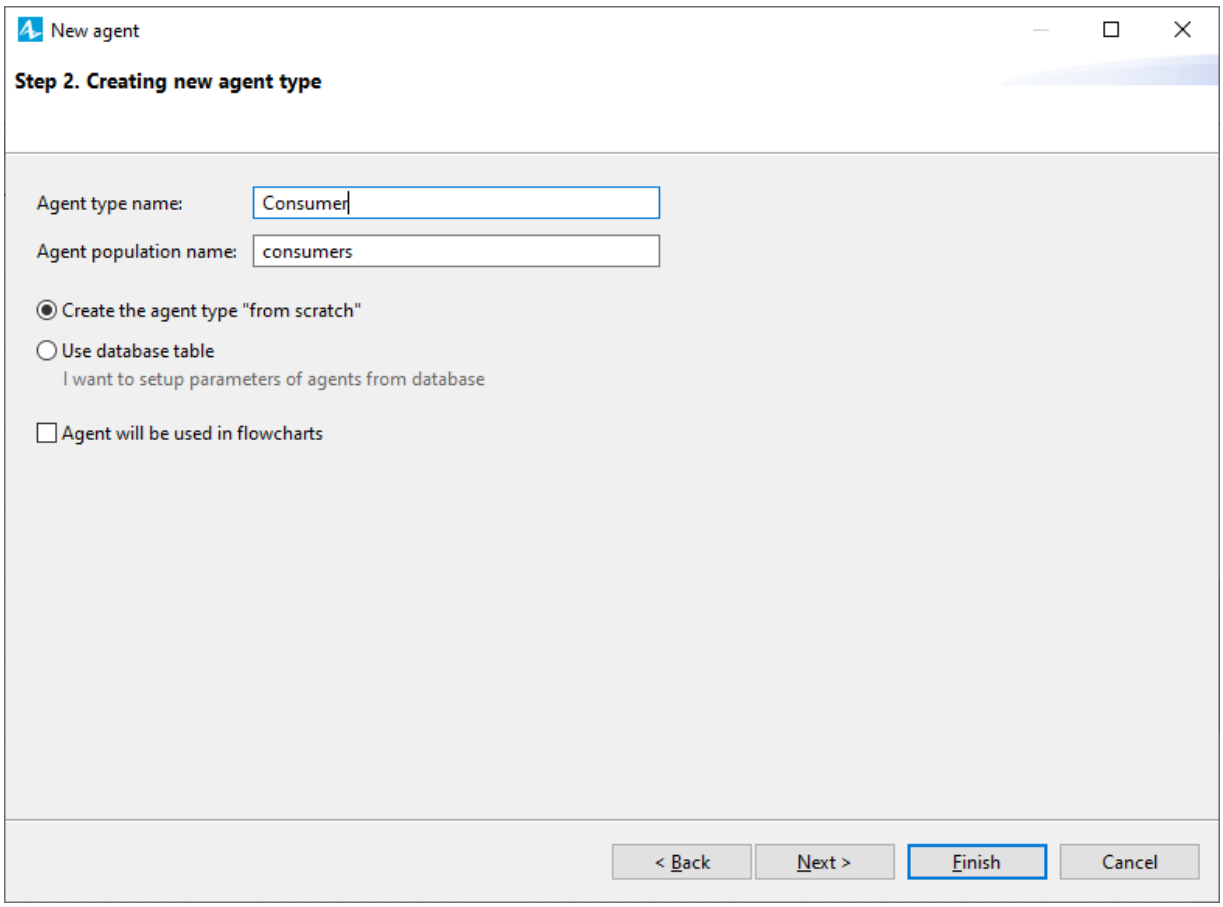


Drag the Agent from the Agent palette on to the Main diagram, and the New agent wizard will open.



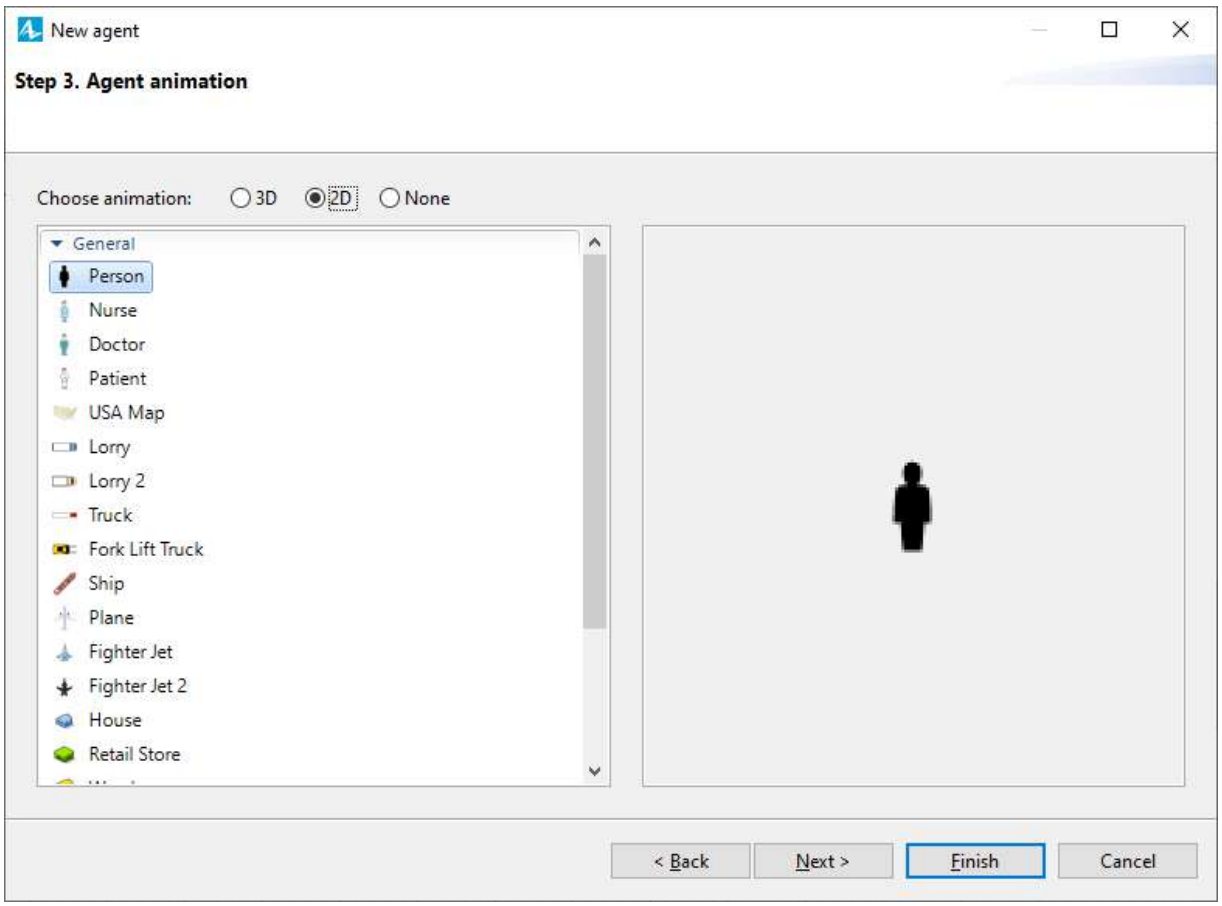
On the Step 1. Choose what you want to create page, select the option that best meets your needs. Since we want to create multiple agents of the same type, select Population of agents and click Next.10

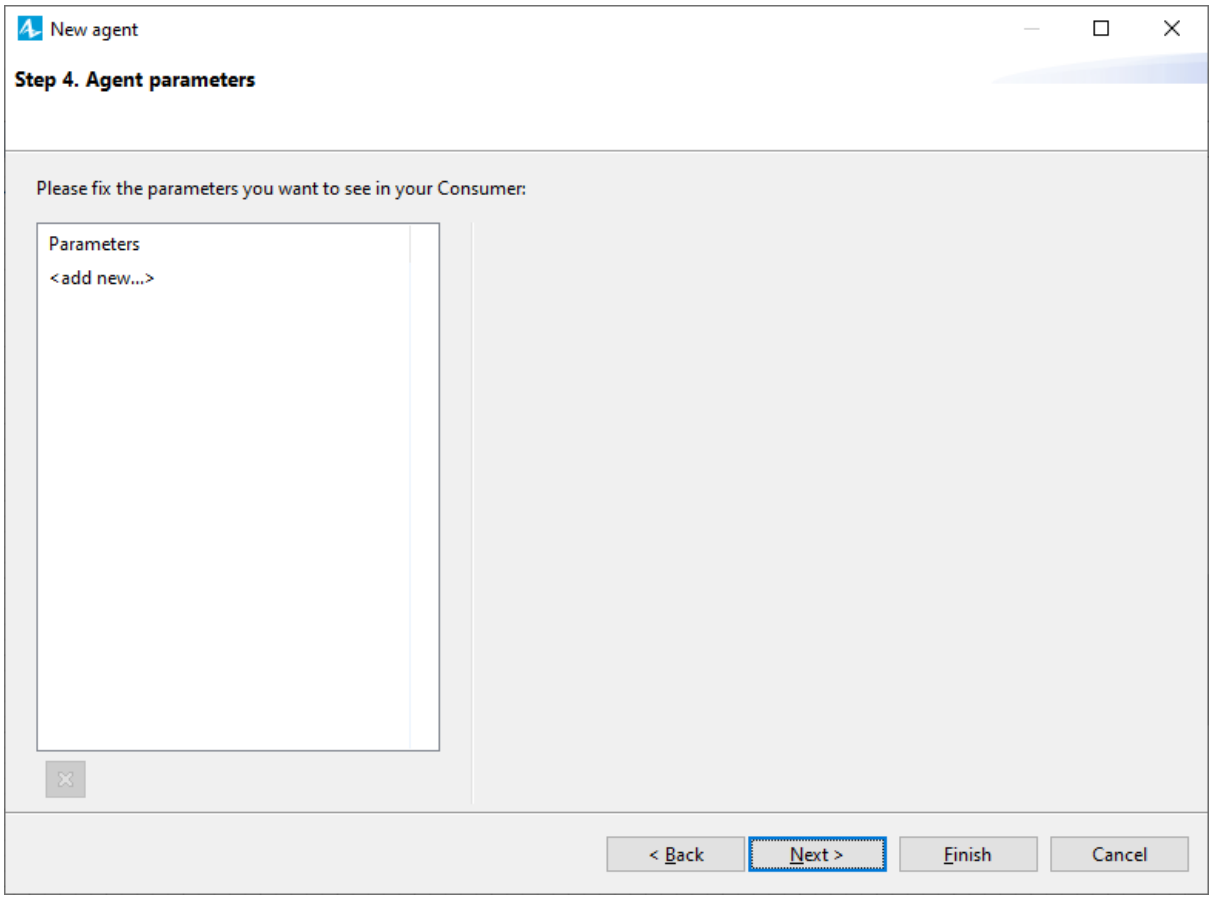
On the Step 2. Creating new agent type page, in Agent type name box, type Consumer. The information in the Agent population name box will automatically change to consumers



Click Next.

On the Agent animation page, choose the agent's animation shape. Since we're creating a simple model that uses 2D animation, choose 2D, select the General list's first item: Person, and click Next.

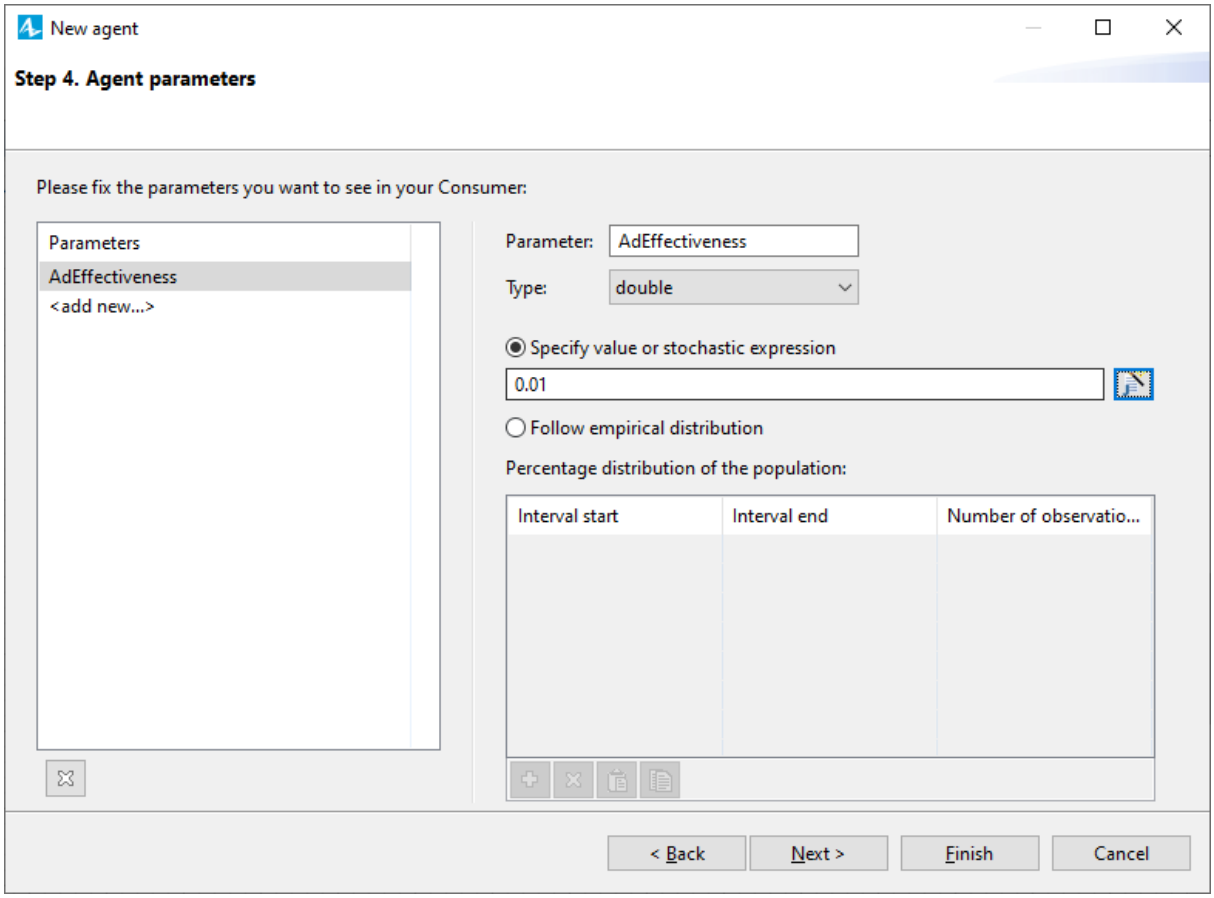

New agent
Step 3. Agent animation
Choose animation:
3D
2D
None
General
Person
Nurse
Doctor
Patient
USA Map
Lorry
Lorry 2
Truck
Fork Lift Truck
Ship
Plane
Fighter Jet
Fighter Jet 2
House
Retail Store
< Back
Next >
Finish
Cancel


New agent
Step 4. Agent parameters
Please fix the parameters you want to see in your Consumer:
Parameters
<add new...>
< Back
Next >
Finish
Cancel

On the Agent Parameters page, define the agent's parameters or characteristics.

Since our model only considers advertising-related product purchases, we'll add a parameter – AdEffectiveness – to define the percentage of potential users who become ready to buy the product during a given day.

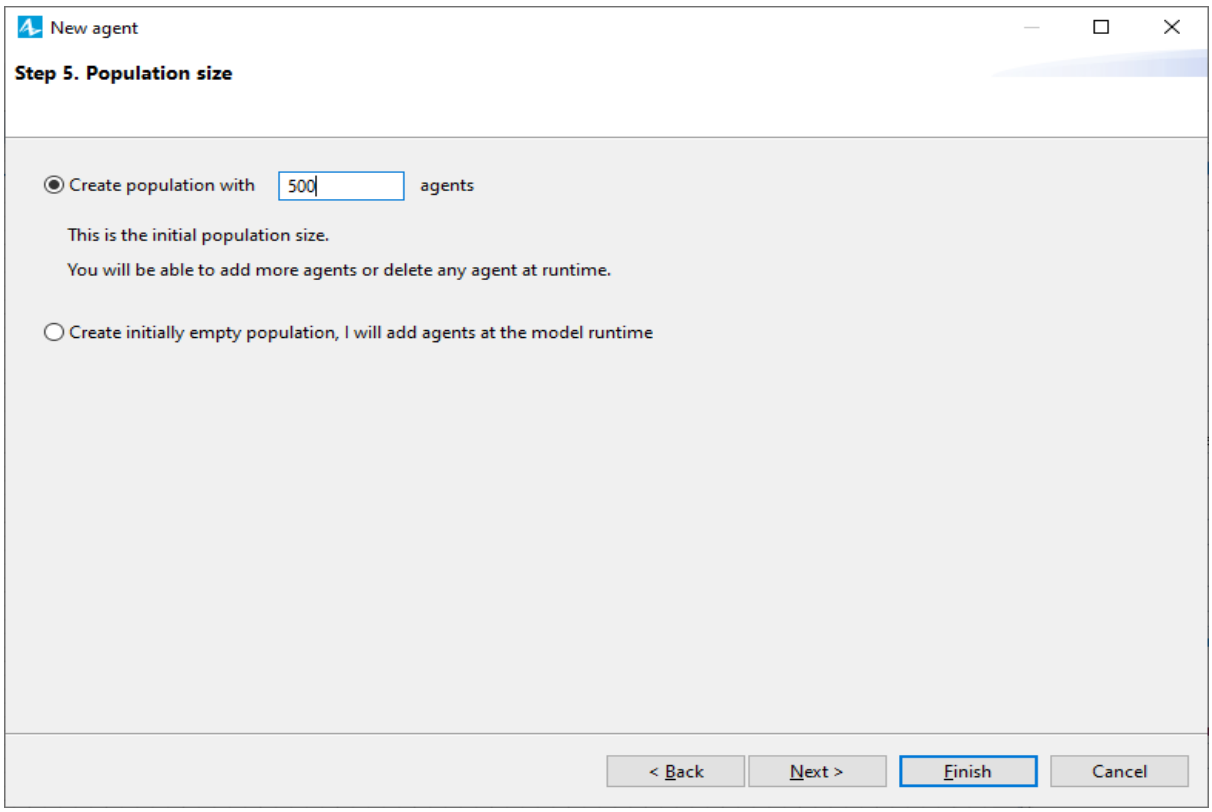
On the left section, in the Parameters table, click <add new…> to create a parameter.



Click next

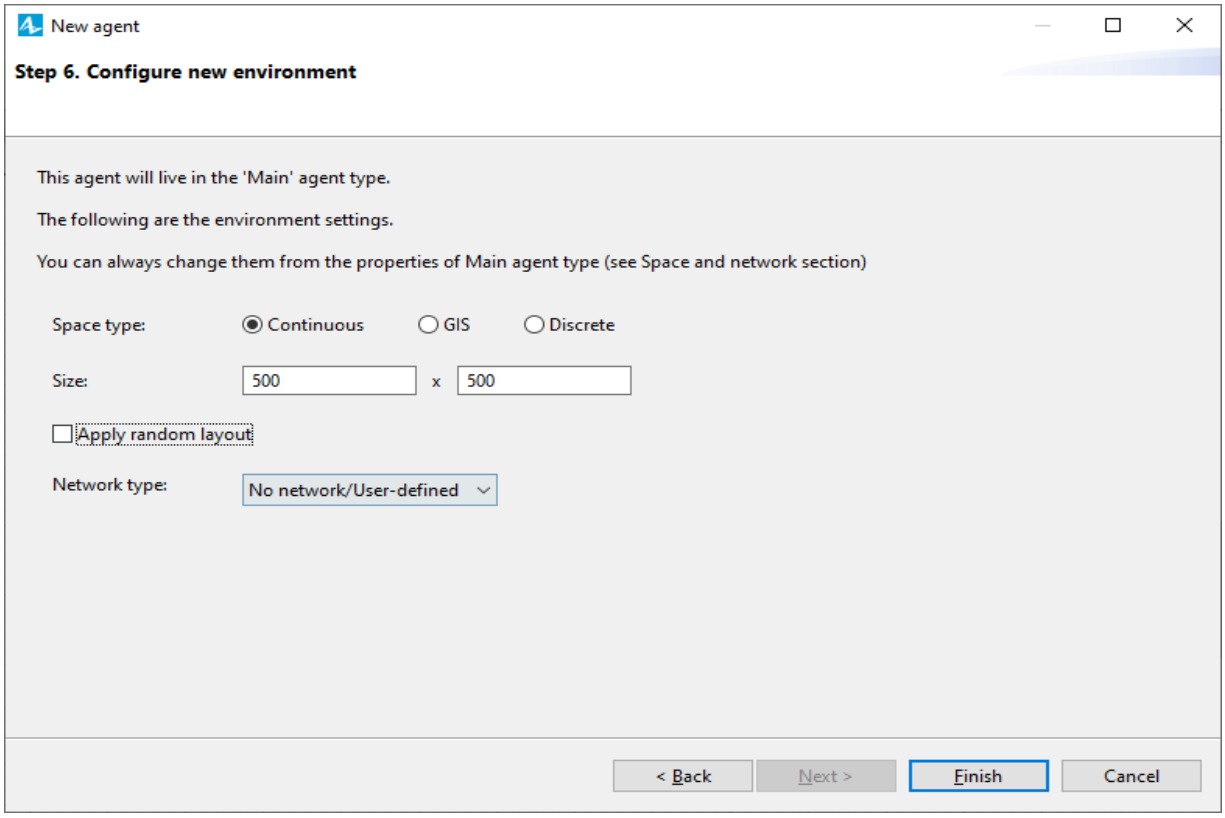
On the Population size page, type 500 in the Create population with …agents box to create 500 instances of the Consumer type. Each instance in the population will model a specific agent-consumer.

While we've created our agent population, we won't see 500 Person animation figures on Main diagram. Instead, AnyLogic will use the 500 agents in the population we've called consumers to simulate the market when we run our model.

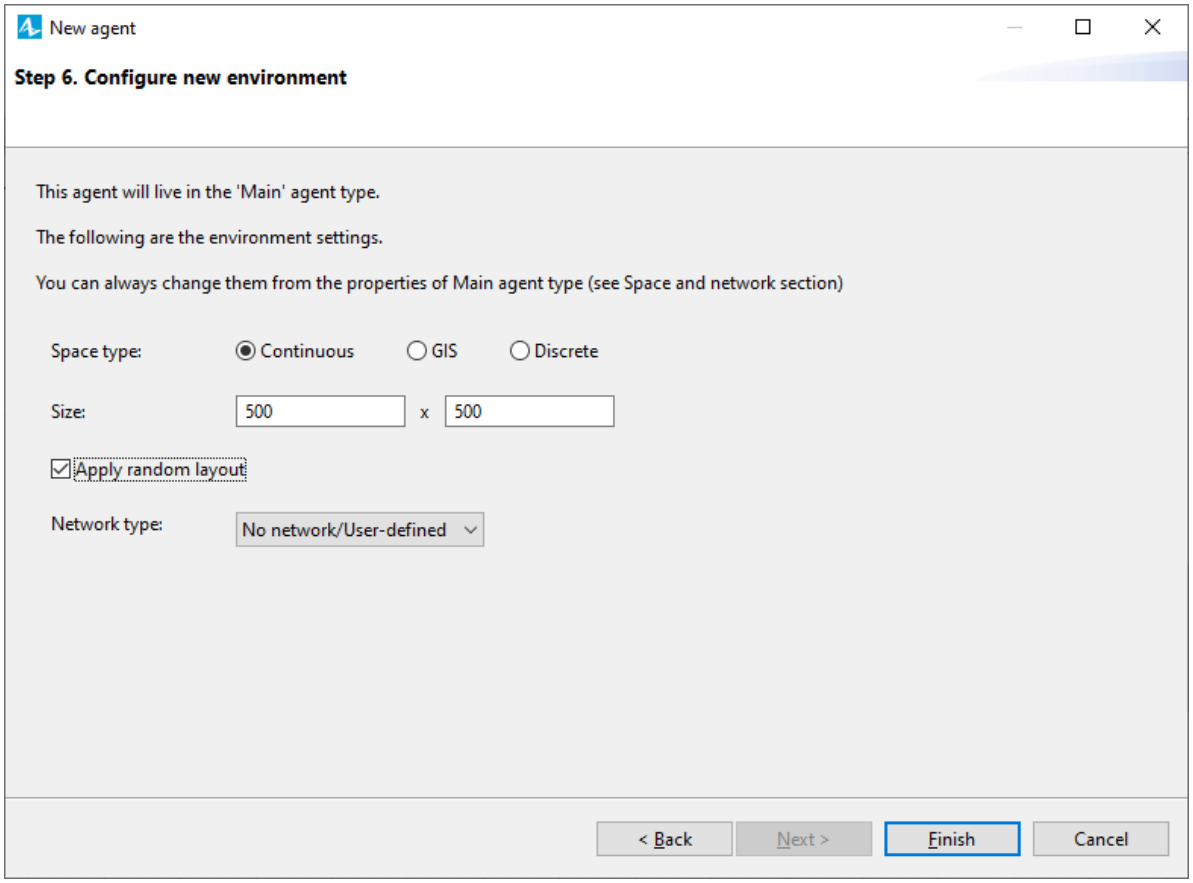


Click Next.

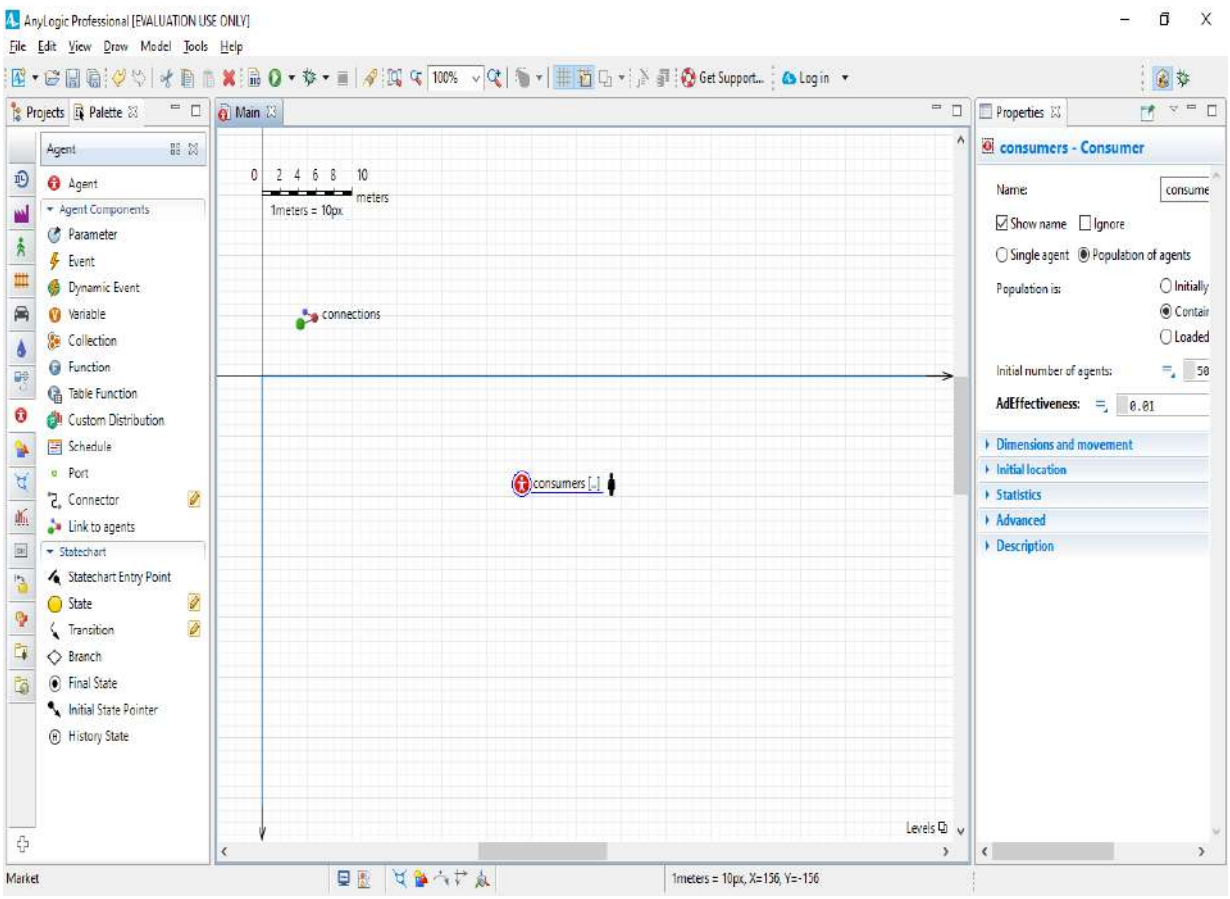
On the Configure new environment page, accept the default values for the environment's space type (Continuous) and both its Width and Height values (500). AnyLogic will display the agents in a 500x500 pixel rectangle.



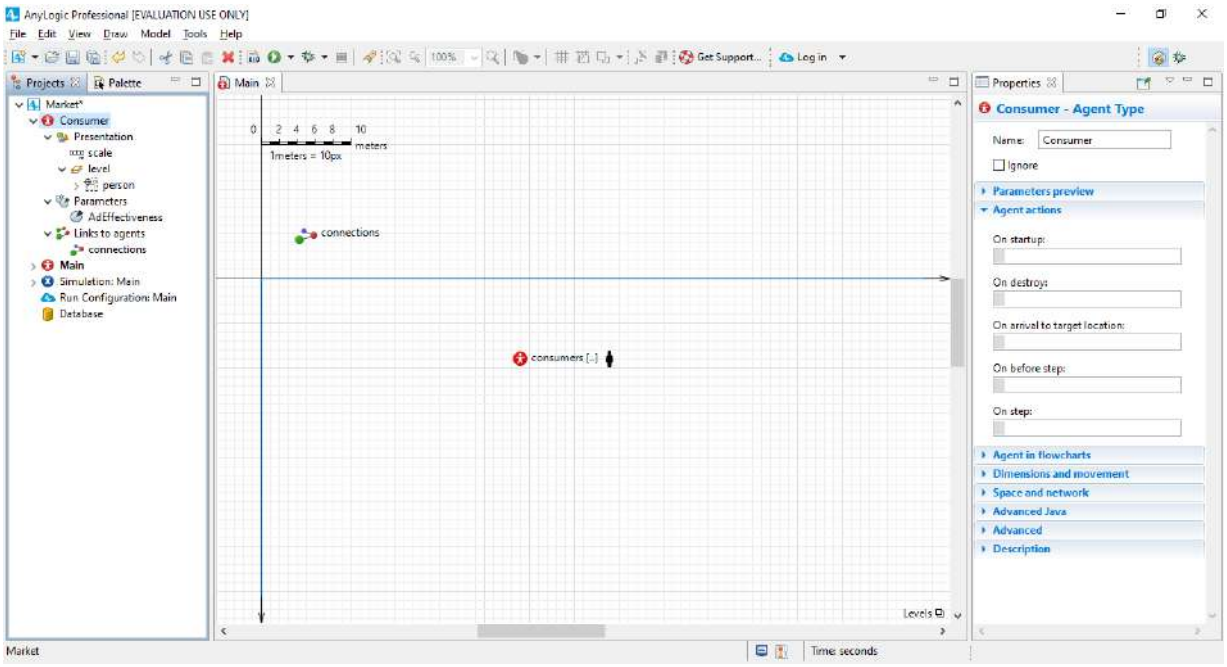
Select the Apply random layout box to randomly distribute the agents across the 500 pixel width and height we've defined. Since we don't want to create an agent network, we'll accept the default No network/User-defined network type.



Click Finish.



Let's use the Projects view to see the new elements that the wizard created. Expand the model tree branches to see the internals.

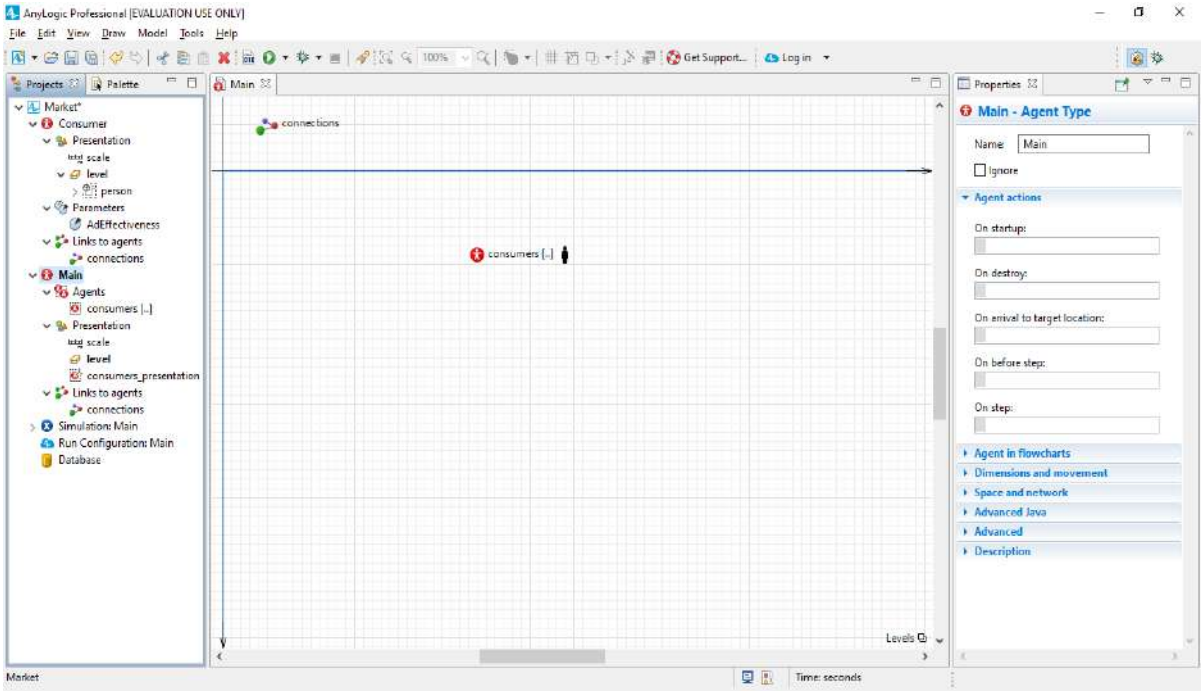


Our model now has two agent types: Main and Consumer.

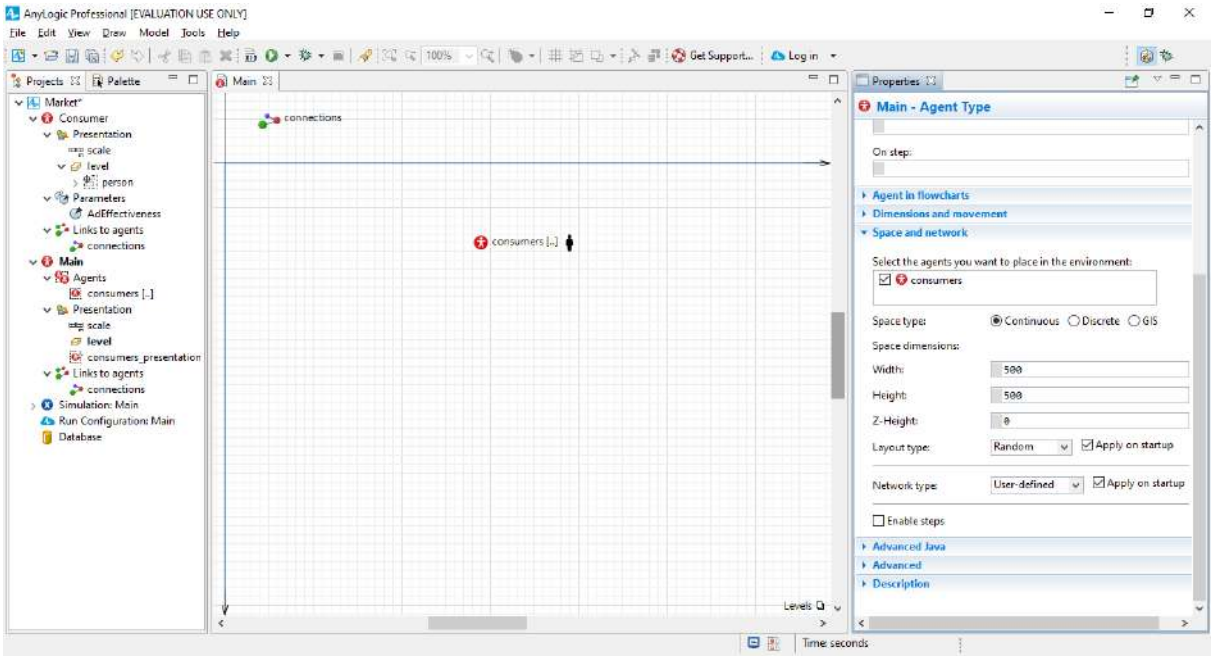
• The Consumer agent type has the agent's animation shape (person, in the Presentation branch) and the parameter AdEffectiveness.

• The Main agent type contains the agent population consumers (a set of 500 agents of type Consumer).

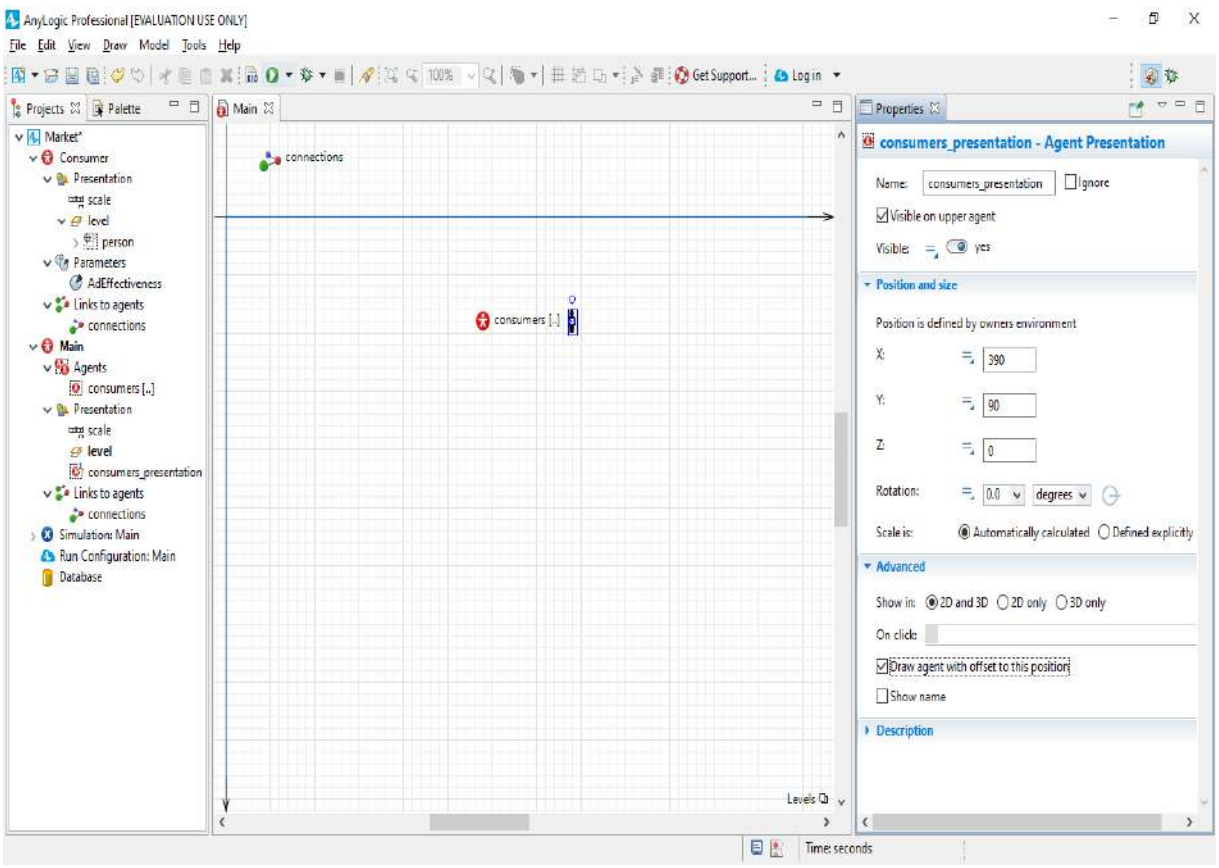
Click Main in the Projects to open its properties in the Properties view (you'll find Properties in the AnyLogic window's right half).



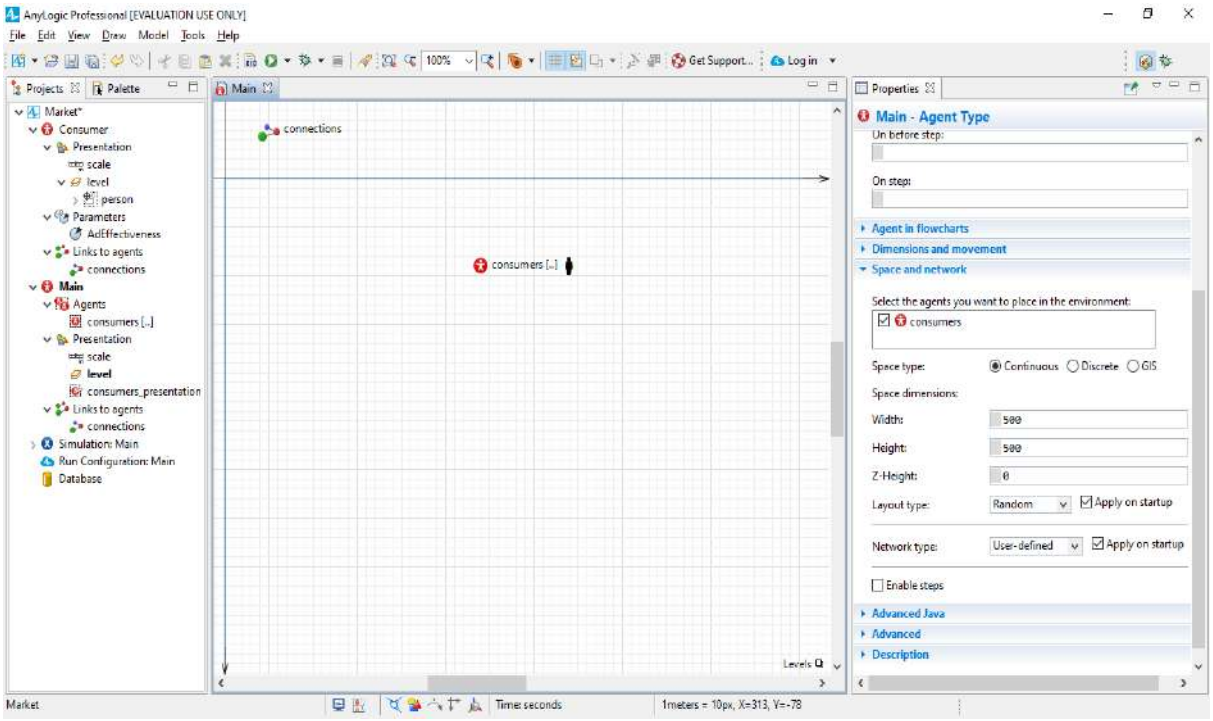
In the Space and network section of Main properties, you can adjust the environment settings for the consumers agent population.



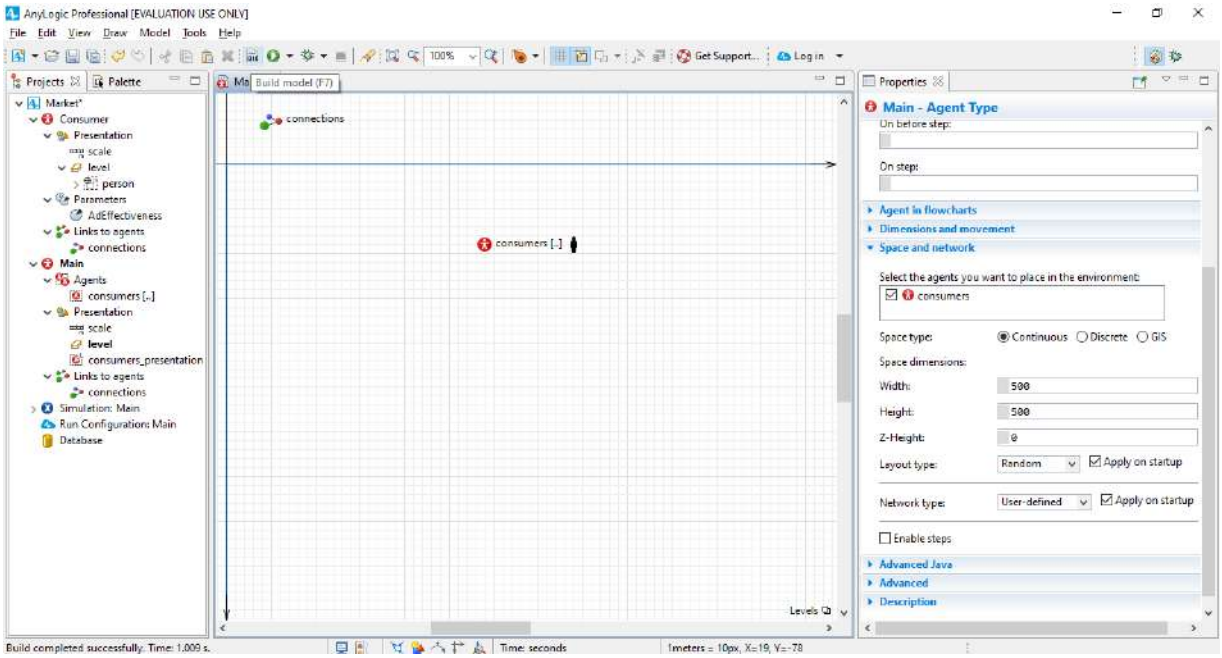
On Main diagram, select the agent population's non-editable embedded animation shape, open the Advanced properties section, and select the Draw agent with offset to this position option.



We've finished building this very simple model, and you can now run it and observe its behavior.

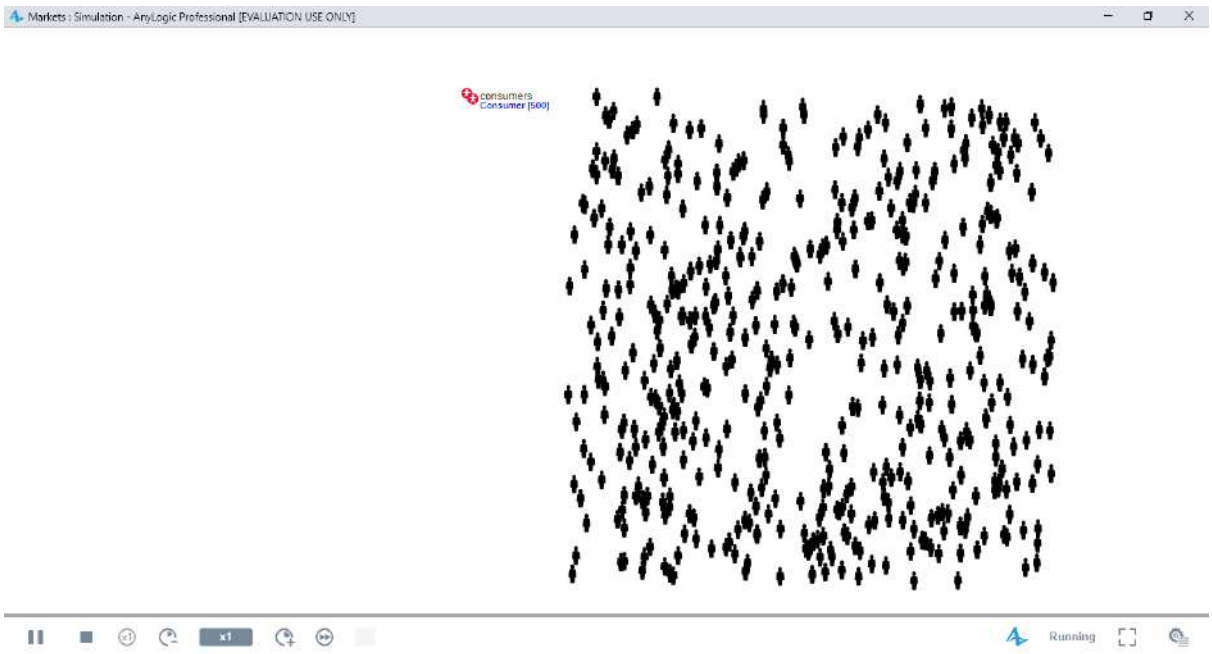


On the toolbar, click the Build button to build the model and check it for errors.



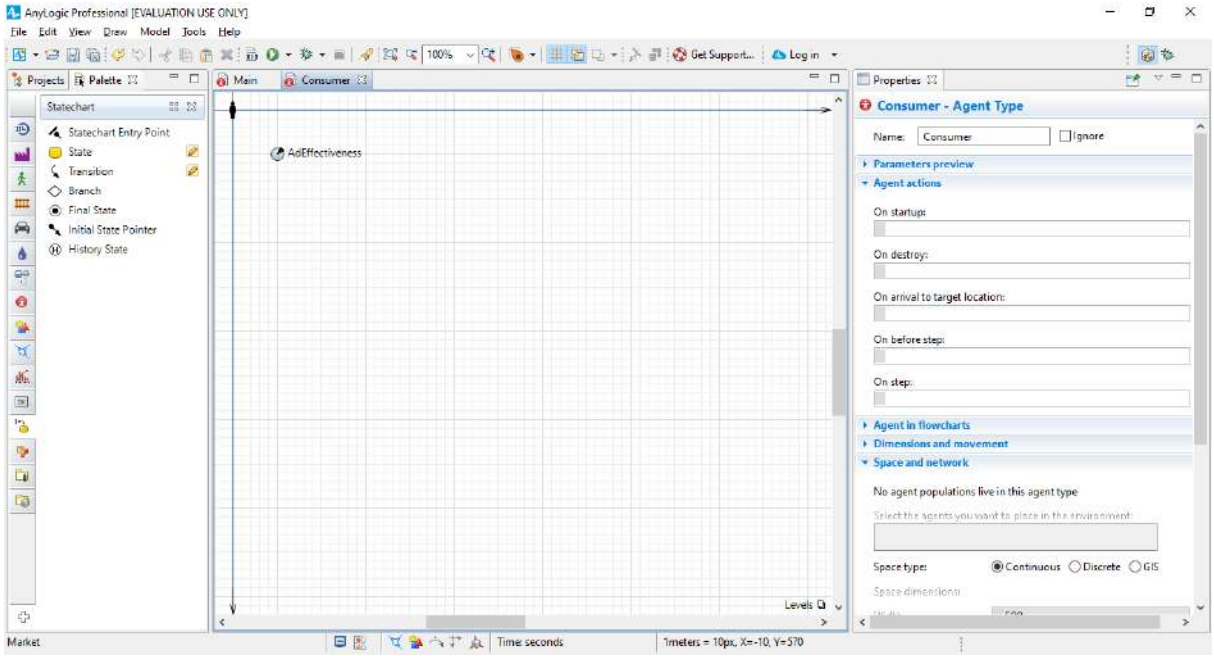
Locate the Run button and click the small triangle to the right. Select the experiment you want to run. Choose Markets / Simulation from the list.

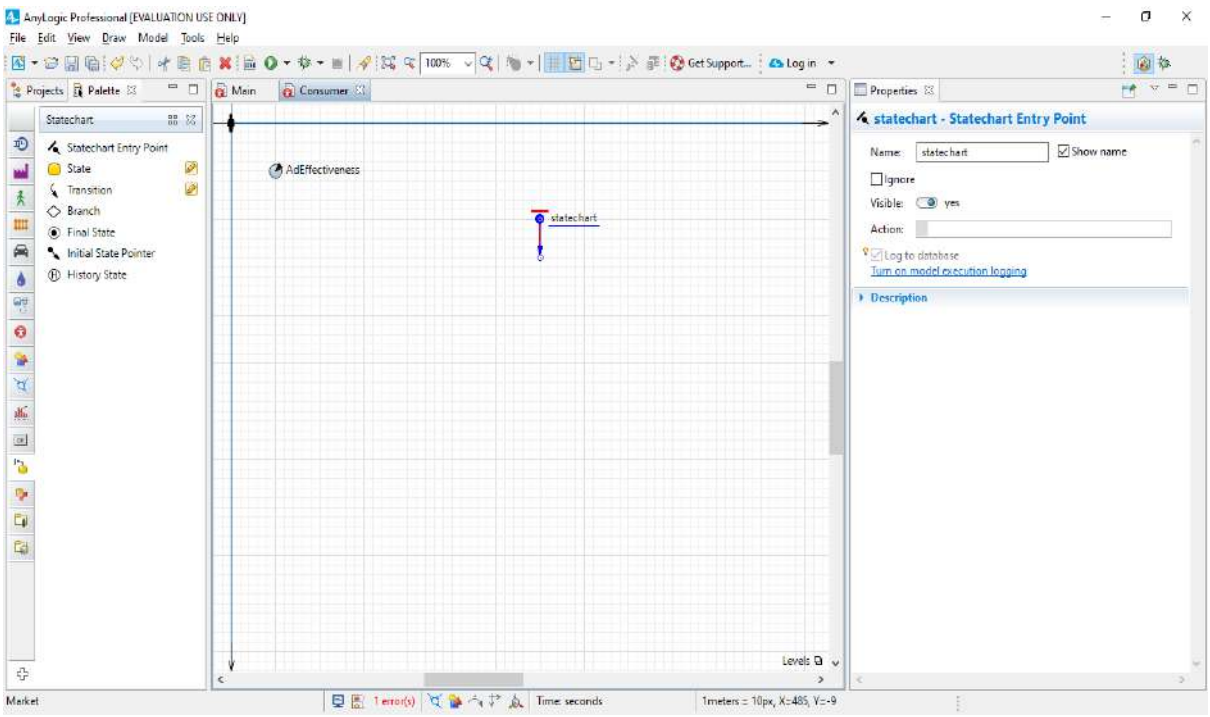
Click the Run button to run the model.



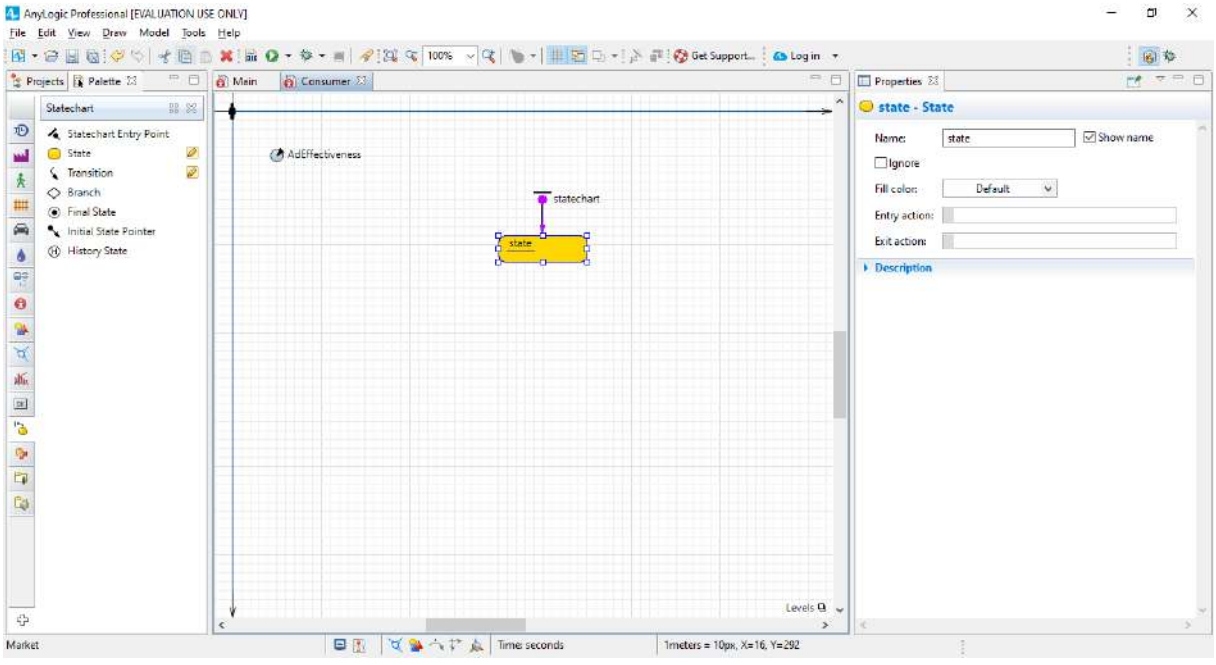
Defining Consumer Behavior

Double click on consumers a new tab of consumers open

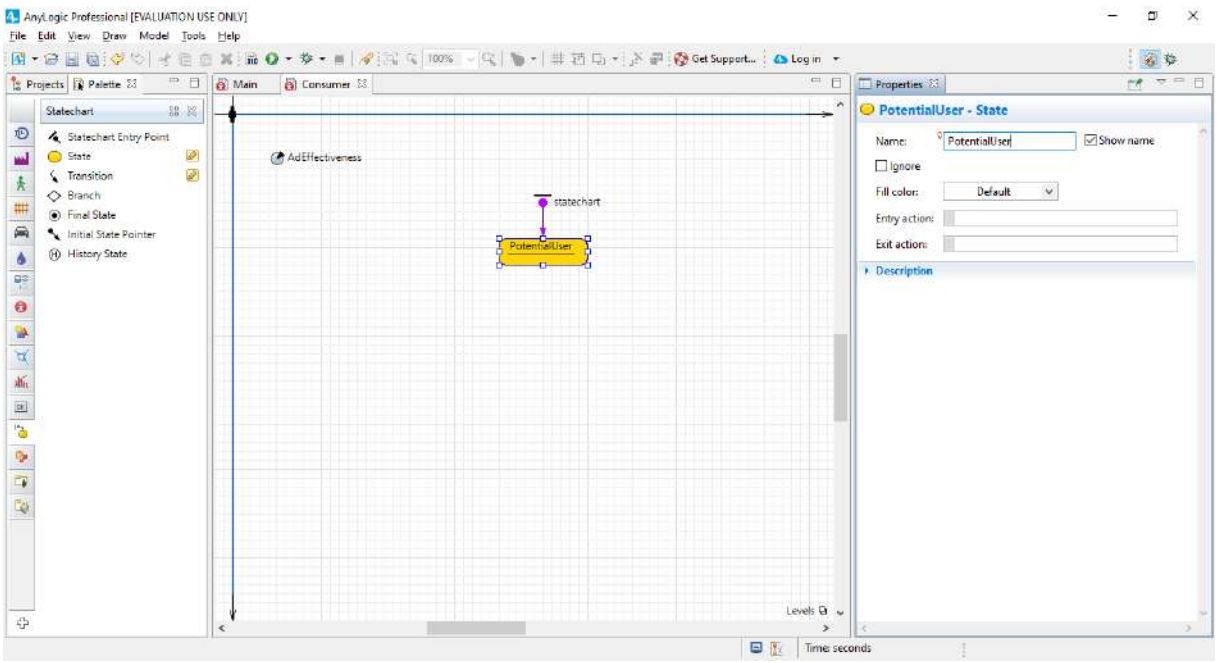




Drag the State and Statechart Entry Point from the Statechart palette on to the graphical diagram and connect it to the statechart entry point.



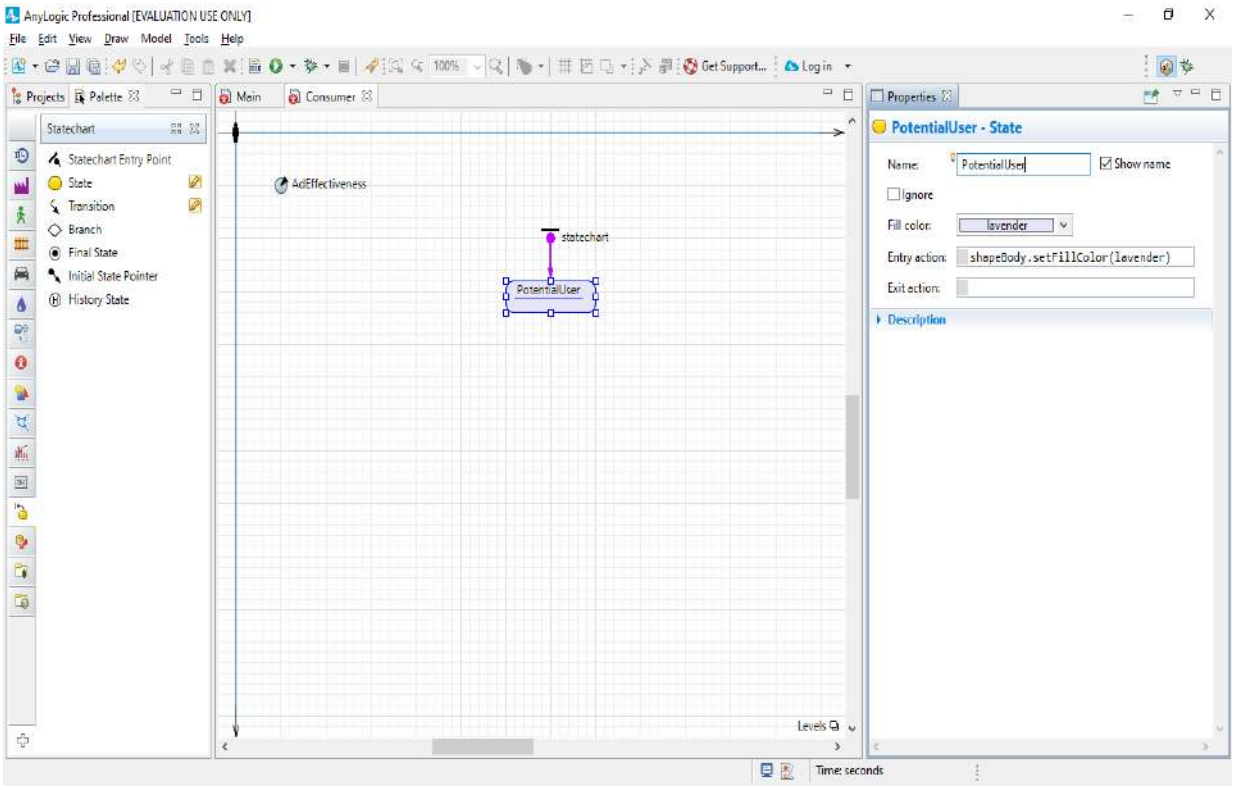
Select the state in the graphical editor and modify its properties. Name the state PotentialUser.



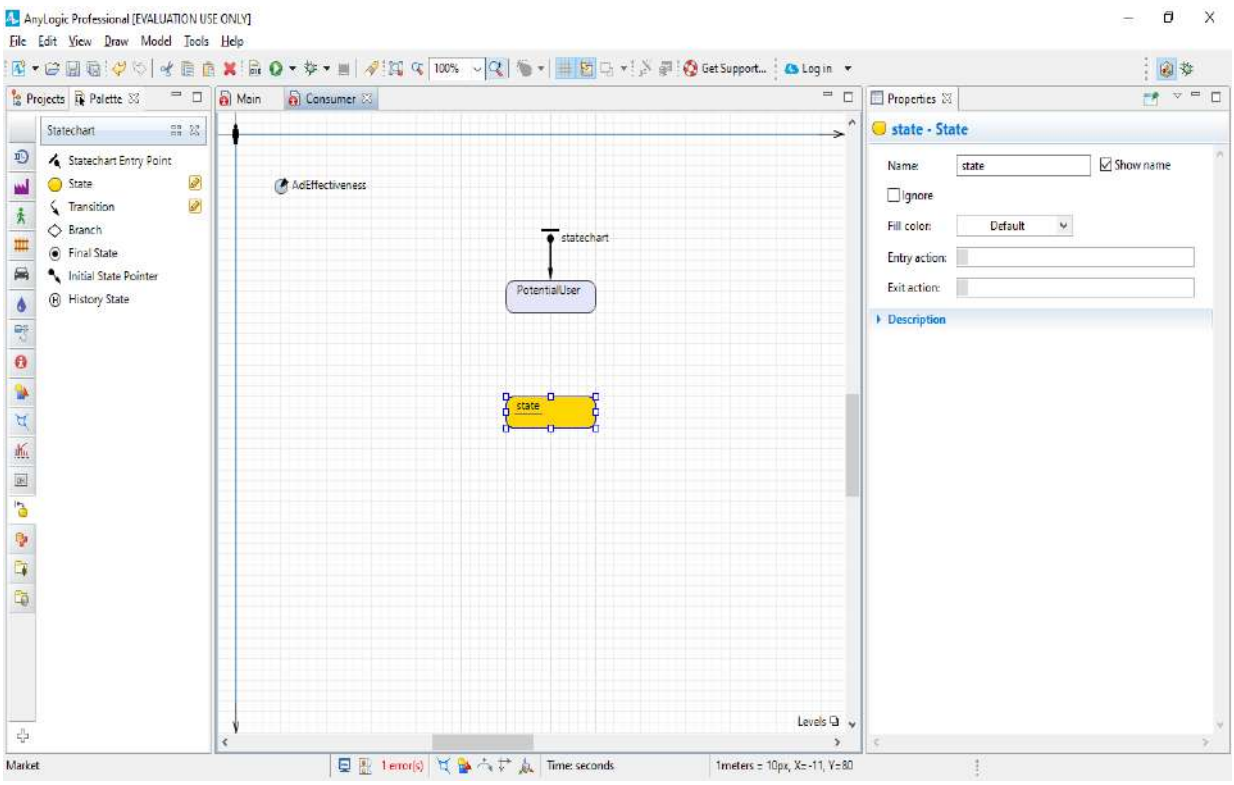
Use the Fill color control to change the state's color to lavender.

Type the following Java code in the state's Entry action field:

shapeBody.setFillColor(lavender)



Add another state in the consumer's statechart:

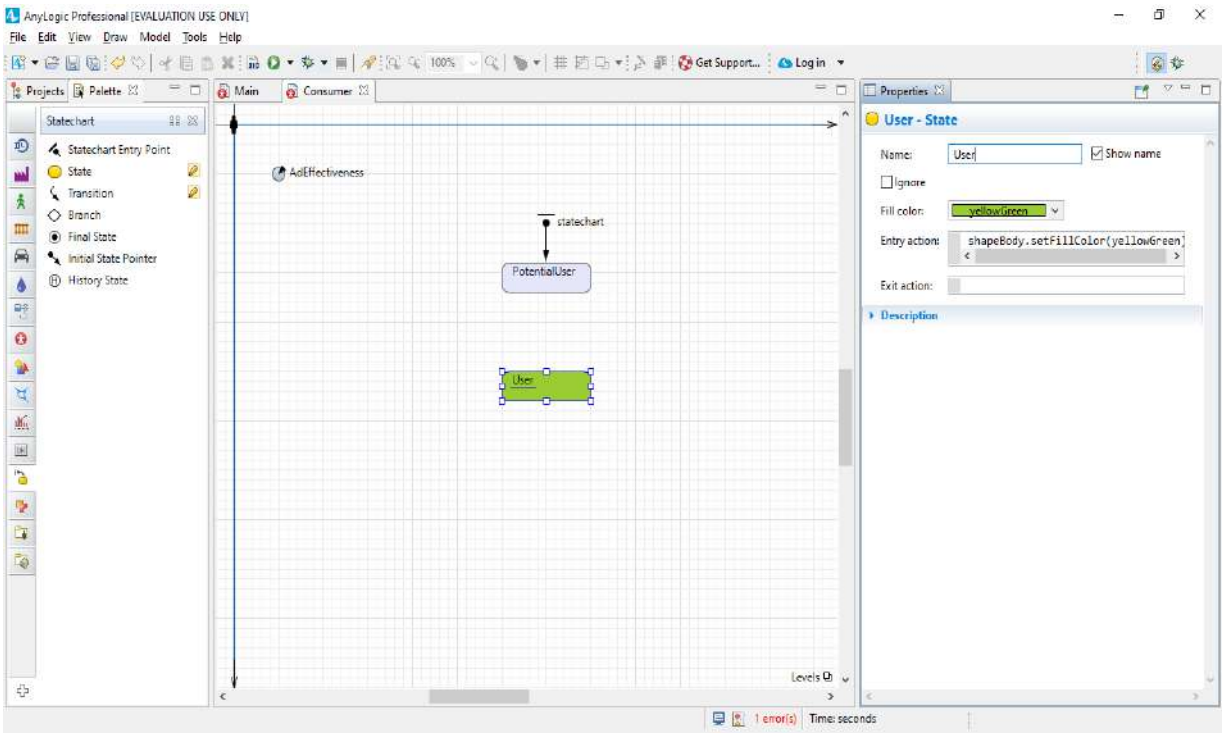


Modify the state's properties like you did earlier:

Name: User

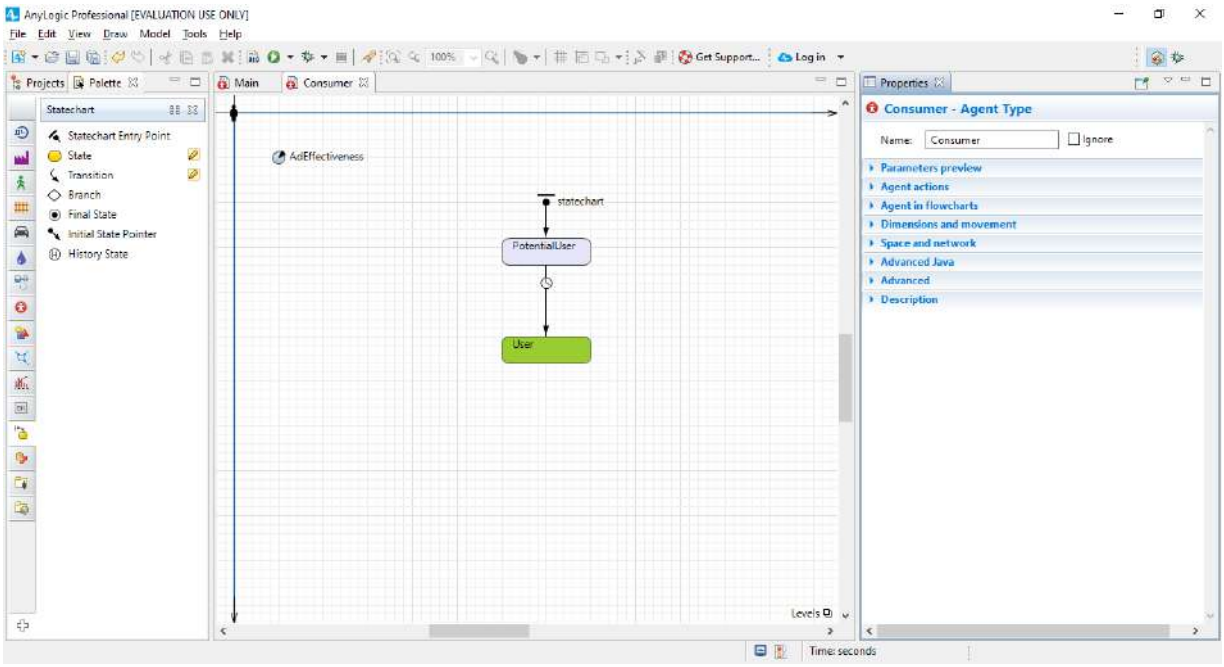
Fill color: yellowGreen

Entry action: shapeBody.setFillColor(yellowGreen);



Draw a transition from PotentialUser to User state to model how persons purchase the product and become product users.

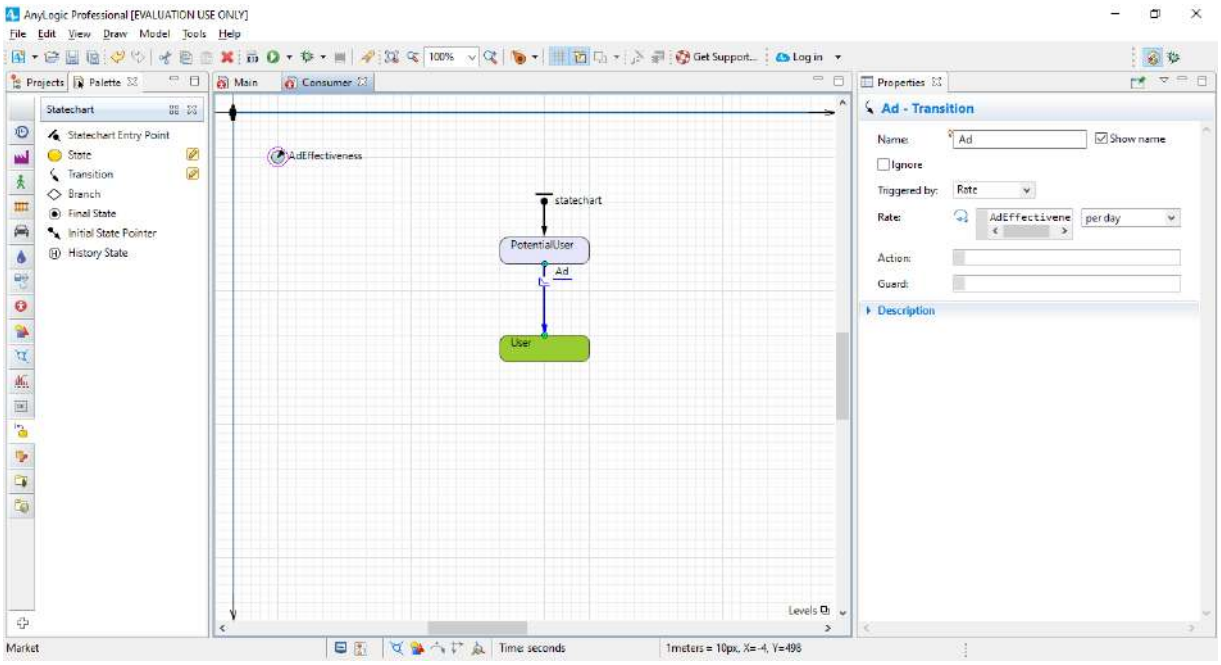
To do so, double-click the Statechart palette's Transition element (the element's palette icon should change to), click PotentialUser state, and click User.



Name the transition Ad to represent "advertising".

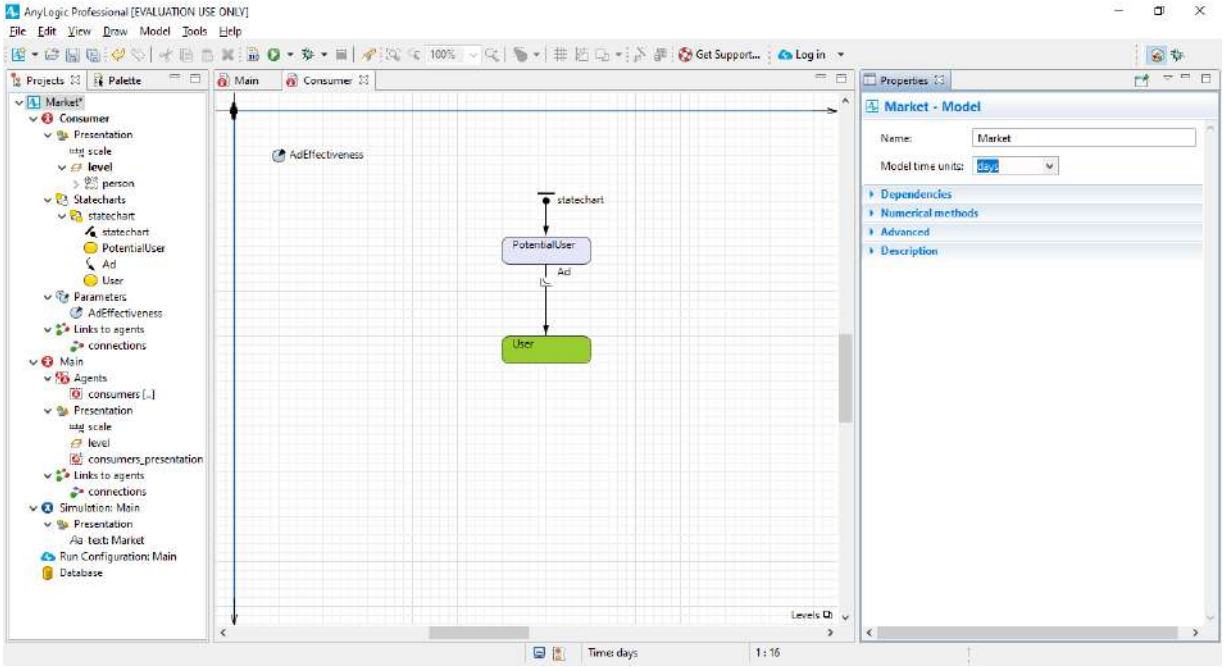
12. Select the Show name checkbox to display the transition's name on the graphical diagram.

13. The transition from PotentialUser to User state will model how advertising leads the person to buy the product. In the Triggered by list, click Rate. In the Rate field, type AdEffectiveness, and then click per day.

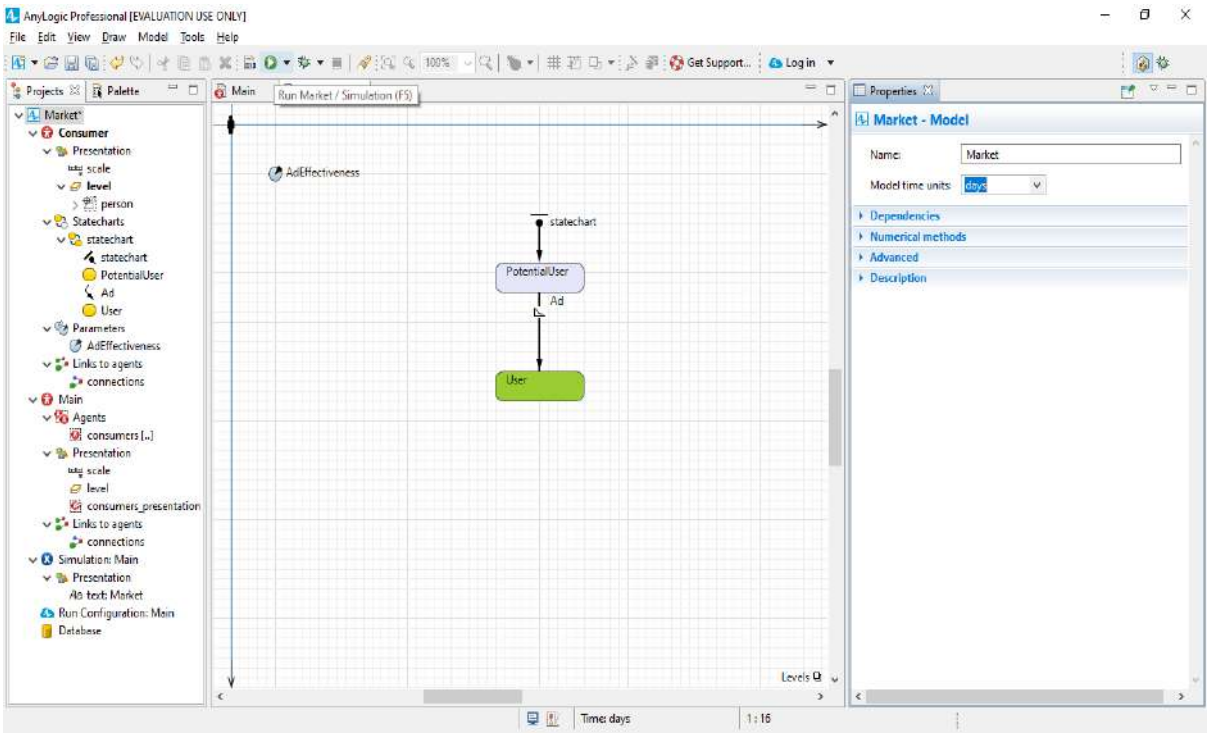


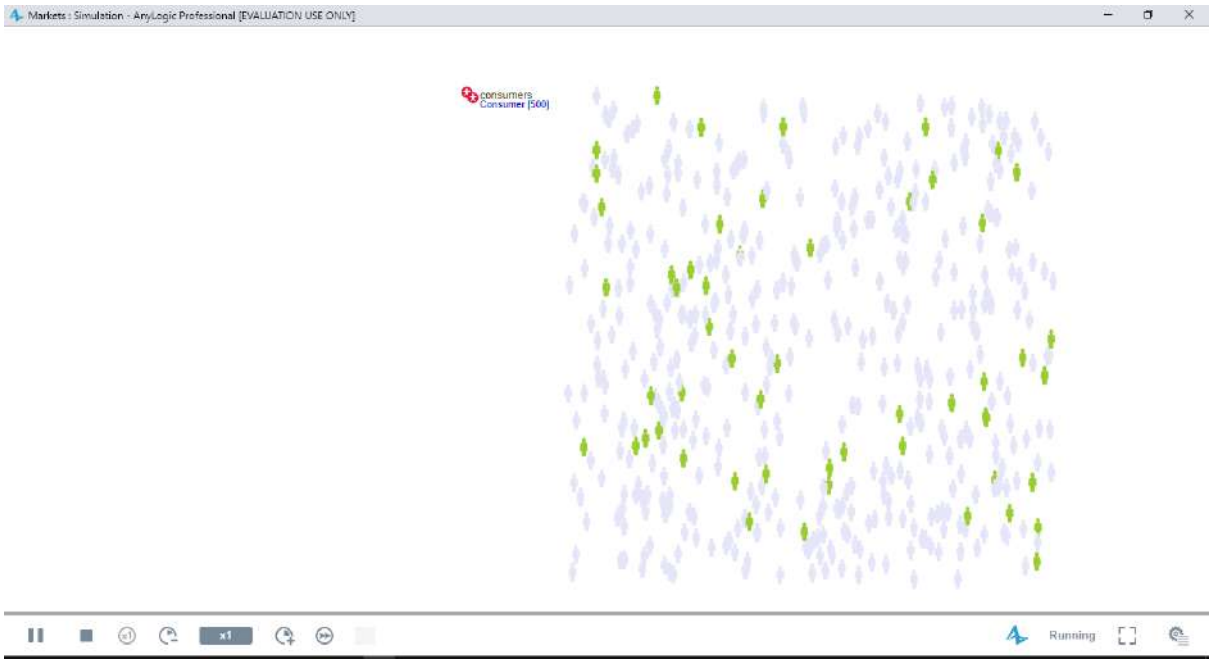
Now, let's set up the model's time units. To tune the model setting, switch from Palette to Projects, and then click the model item in the tree (the tree's top object, Market).

In the Properties view, choose days as the Model time units.

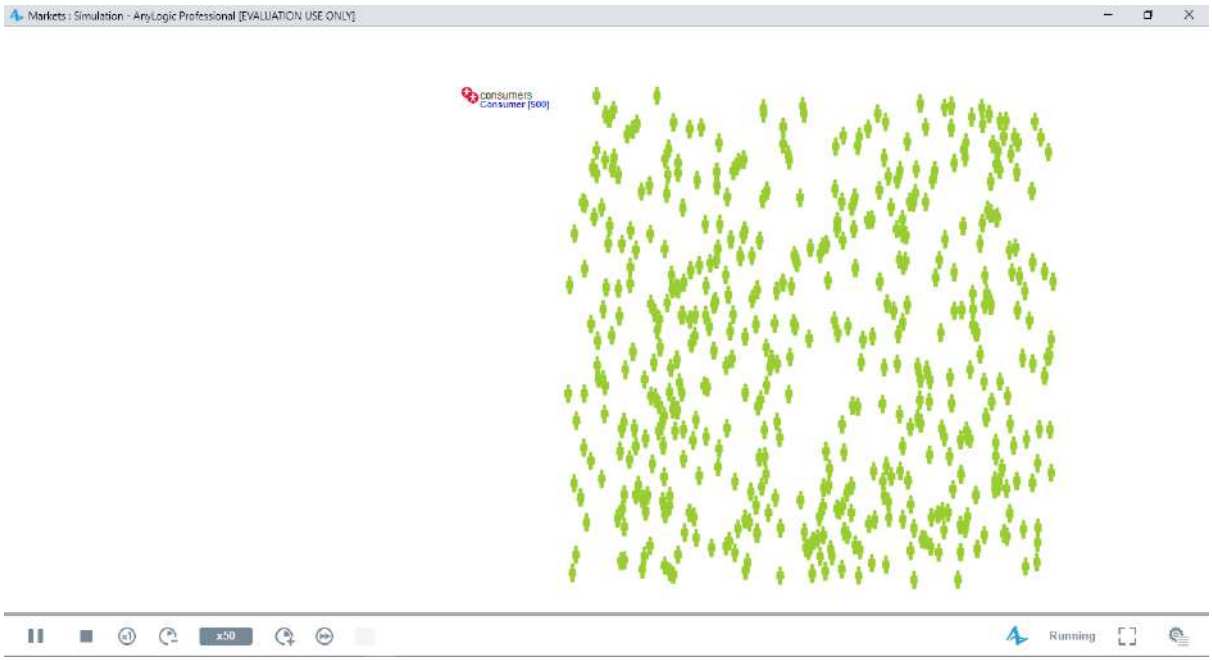


Run the model. The population should gradually turn green – a change that represents the effect of advertising - until every consumer buys the product.



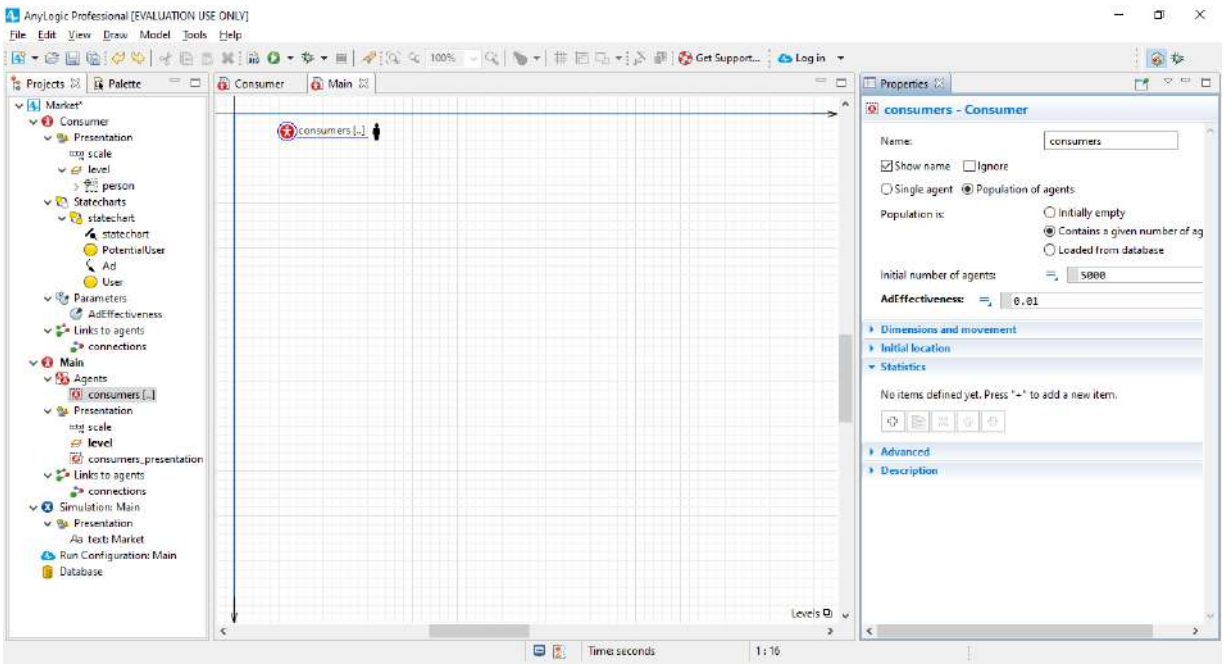


To adjust the model's execution speed, click the toolbar's Slow down or Speed up buttons. If you increase the speed to 10x – you'll see the speed at which the population turns green also increase

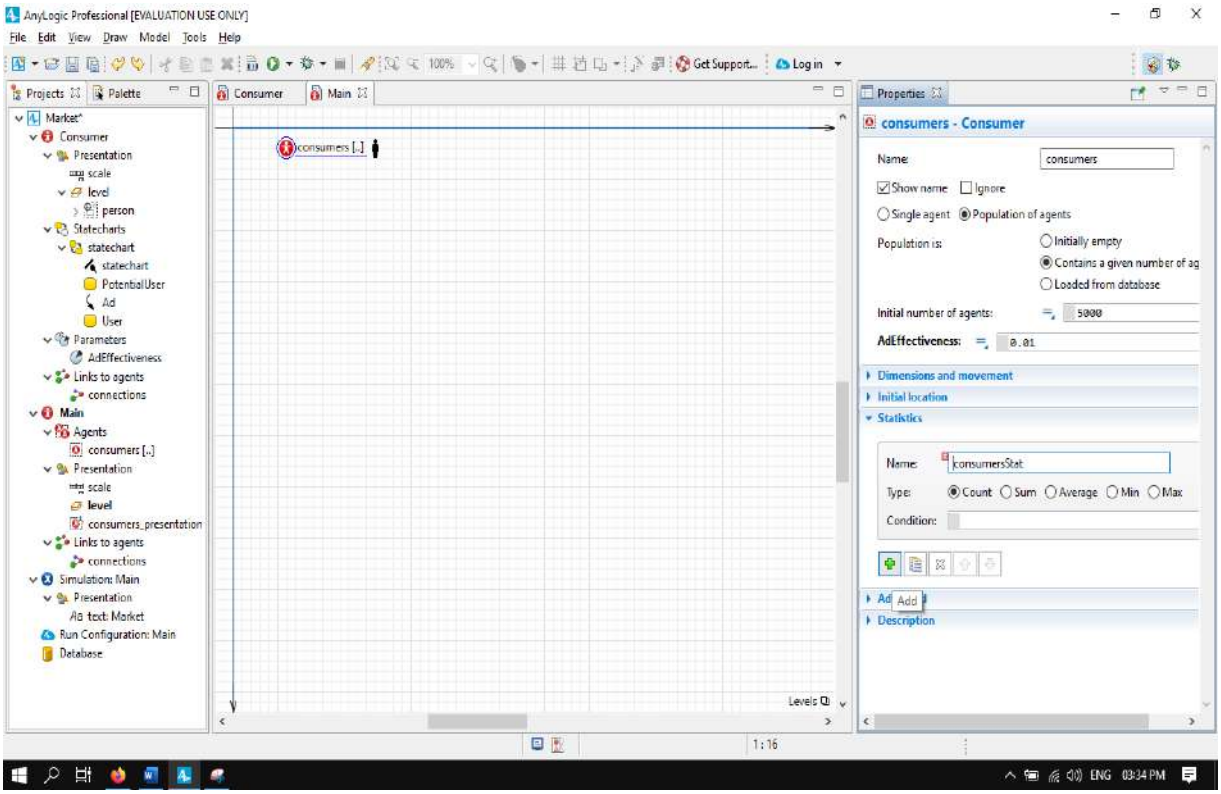


Adding a chart to visualize the model Output

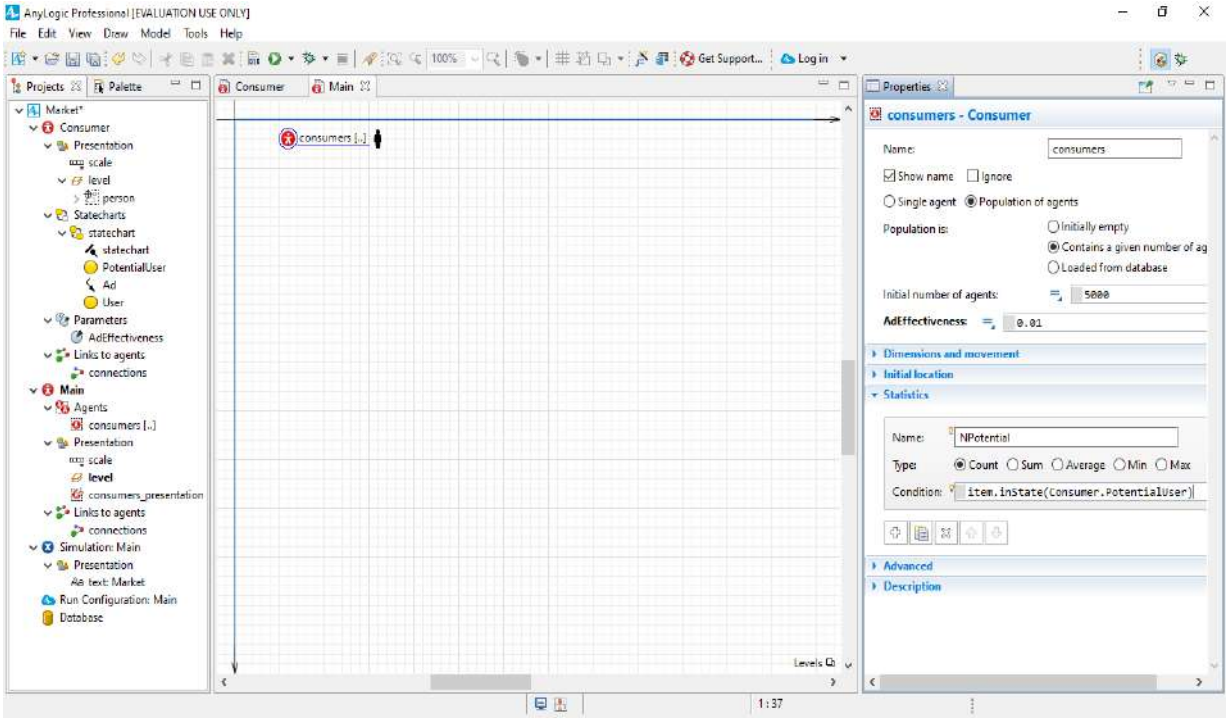
First, define a function to count potential users. To add a new function that collects statistics for agents, open the diagram of the agent type Main, select the agent population consumers, and go to the Statistics properties section.



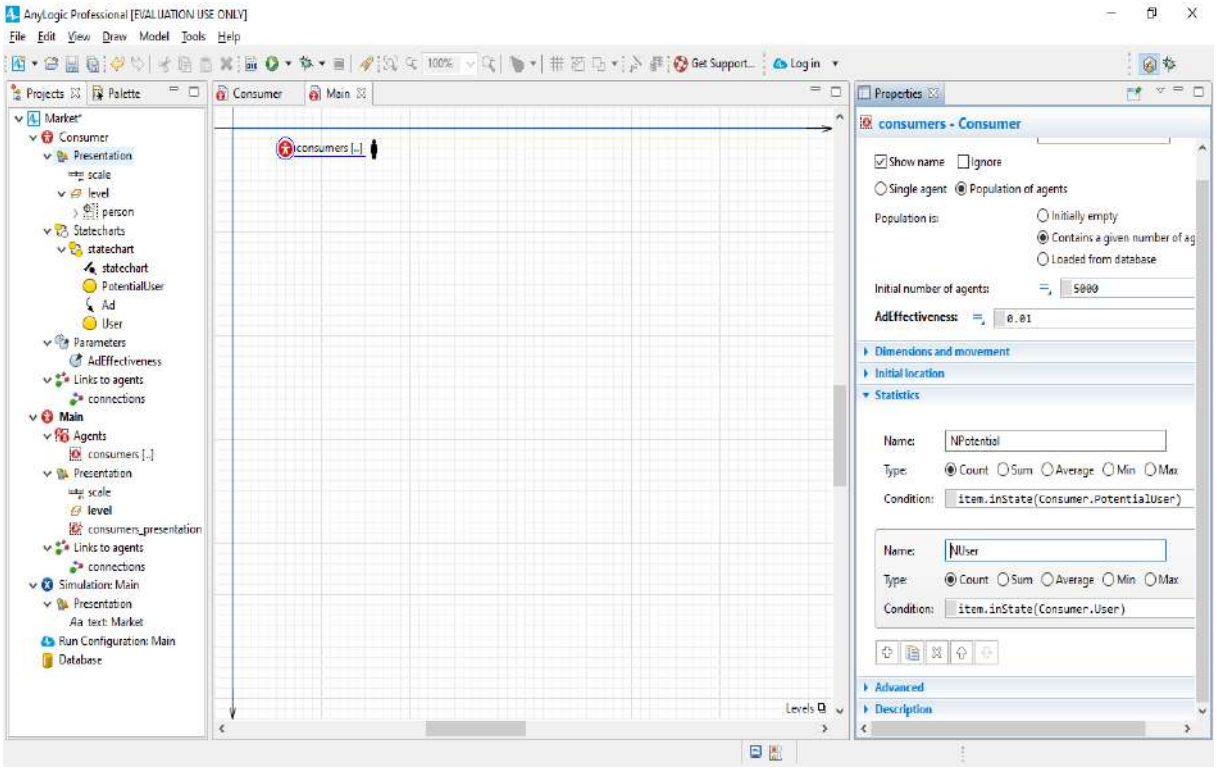
Define the function of type Count with the Name NPotential. The statistics of type count iterates through a given population – in our case, the number of agents – to count those that meet the selected condition.



Enter item.inState(Consumer.PotentialUser) as the function Condition.

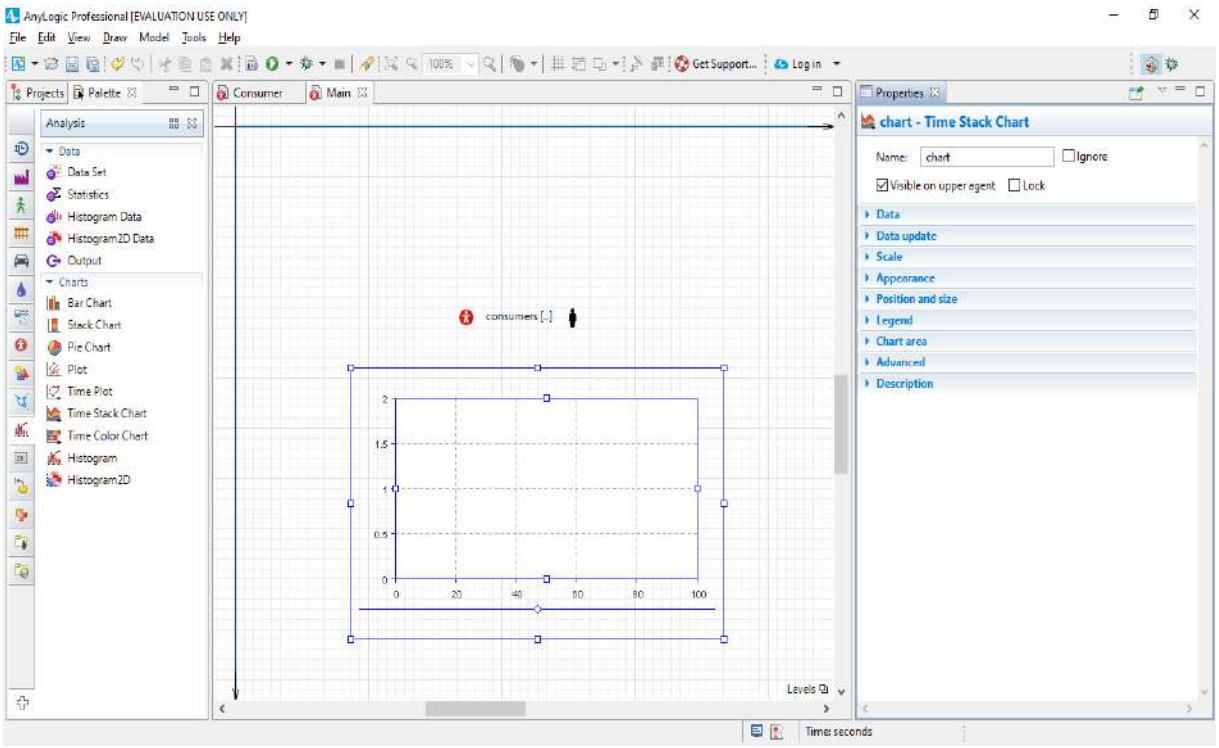


Define a second statistics function to calculate the number of product users. Name it NUser and let it count the number of agents, conforming the Condition item.inState(Consumer.User). You can duplicate the other statistics function by clicking the Duplicate button and changing its Name and the Condition.

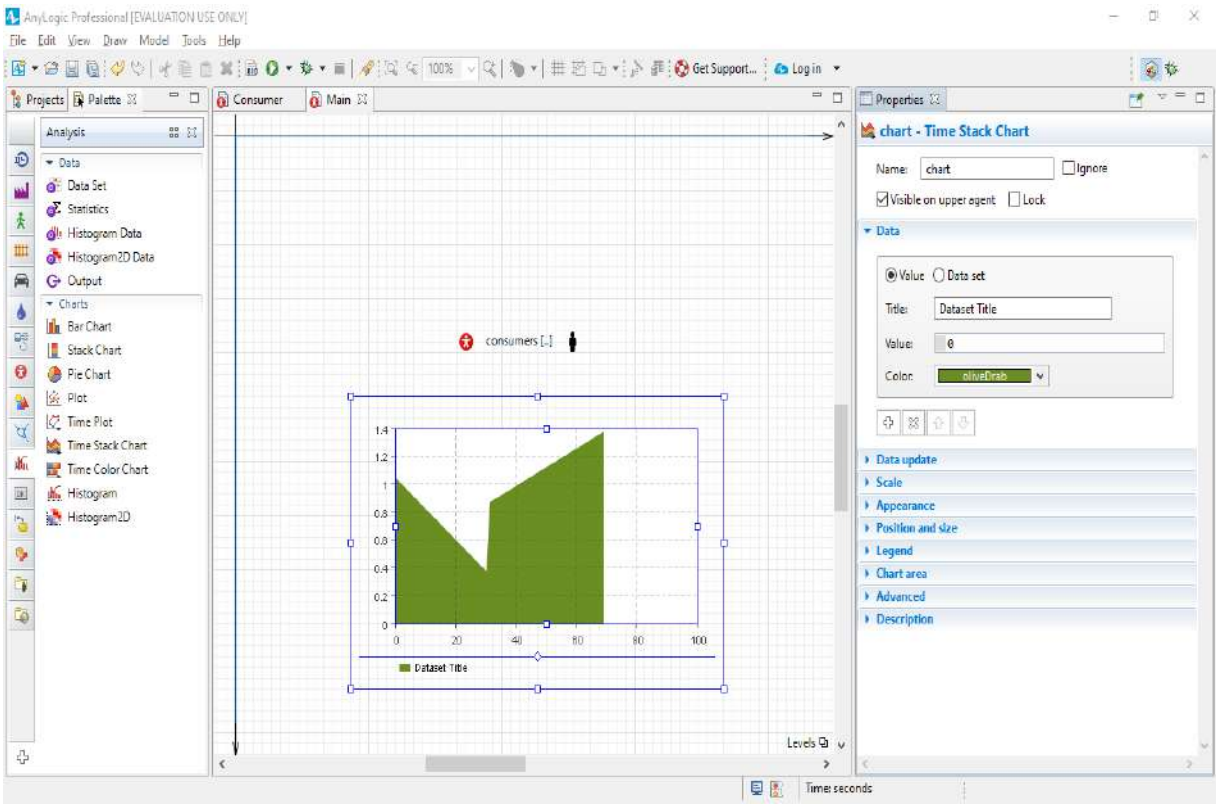


Now, let’s add a chart to show the statistics these functions collect and display the adoption process dynamics.

Open the Analysis palette and drag the Time Stack Chart from the Analysis palette on to the Main diagram to create a chart that will display the dynamics of users and potential users. Increase the time stack chart as shown in the figure below:



Add two data items for the chart to display. Here we'll call our statistics functions NUser and NPotential we have defined for consumers population on the previous step. Click Add data item to add the statistics you want to draw on the chart.



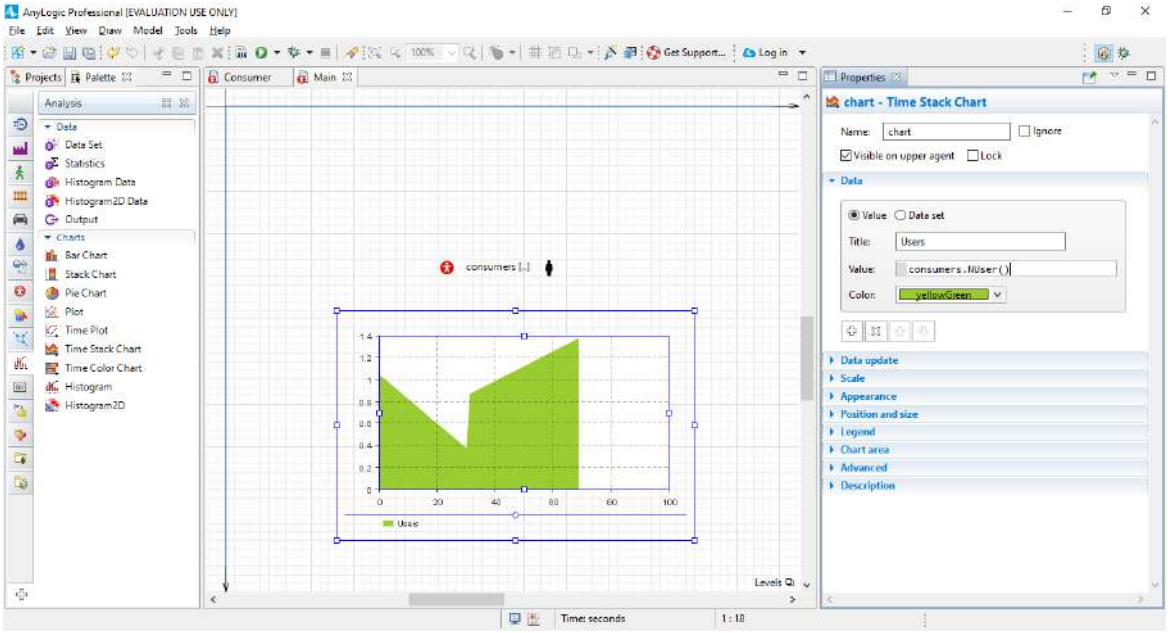
Modify the data item's properties:

• Title: Users – the data item's title.

• Color: yellowGreen

• Value: consumers.NUser()

Our agent population name is consumers, and NUser() is the statistics function that we defined for this population.

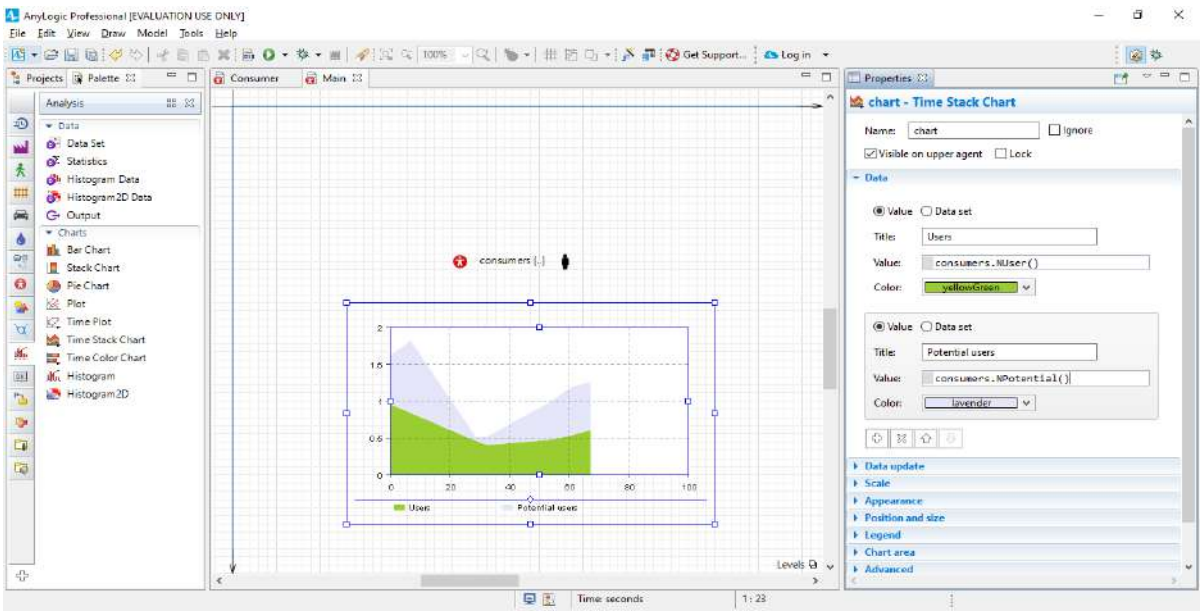


Add one more data item:

• Title: Potential users

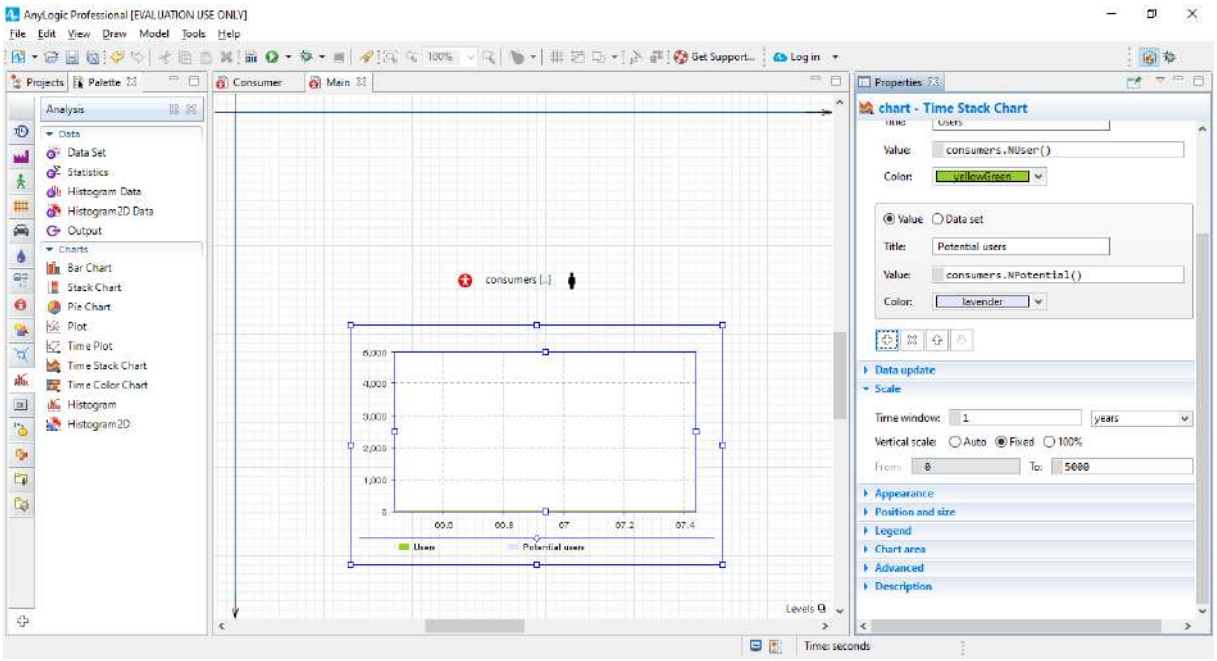
• Color: lavender

• Value: consumers.NPotential()

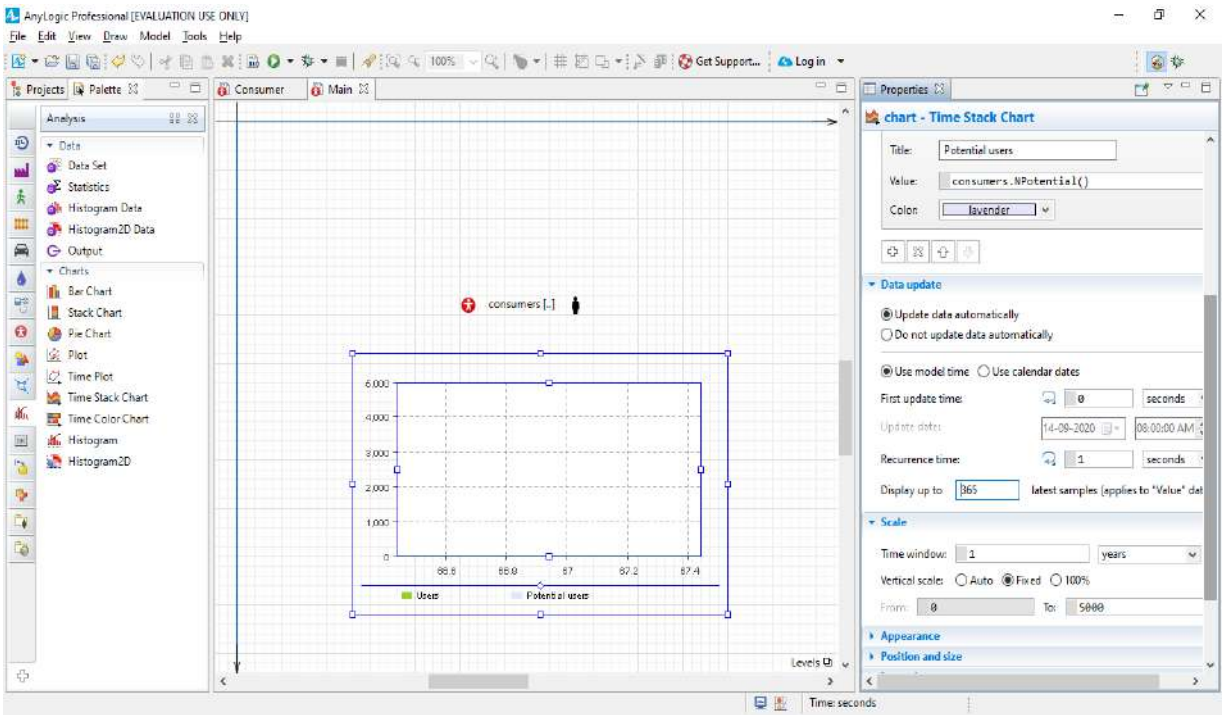


Go to the Scale section and set Time window equal to 1 year.

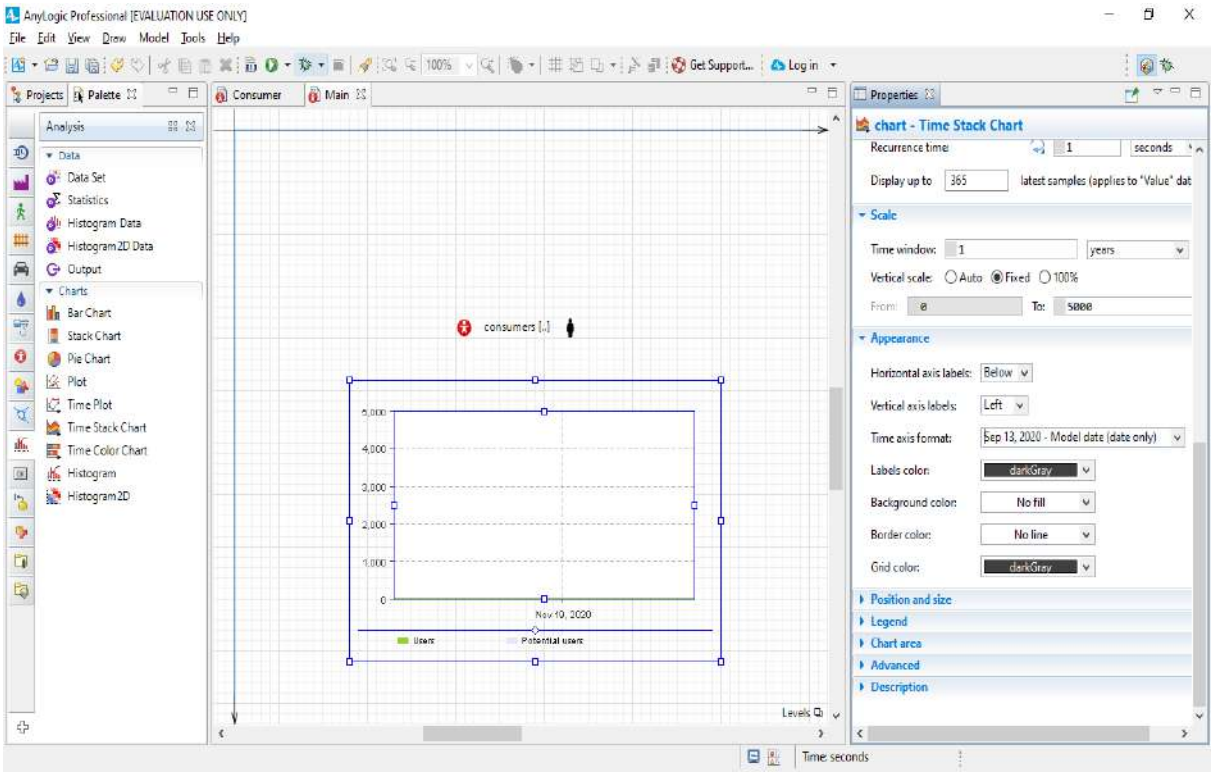
Since our chart will show statistics for consumers population and our model has 5,000 consumers, set the chart's Vertical scale to Fixed, and enter 5000 in the to: box.



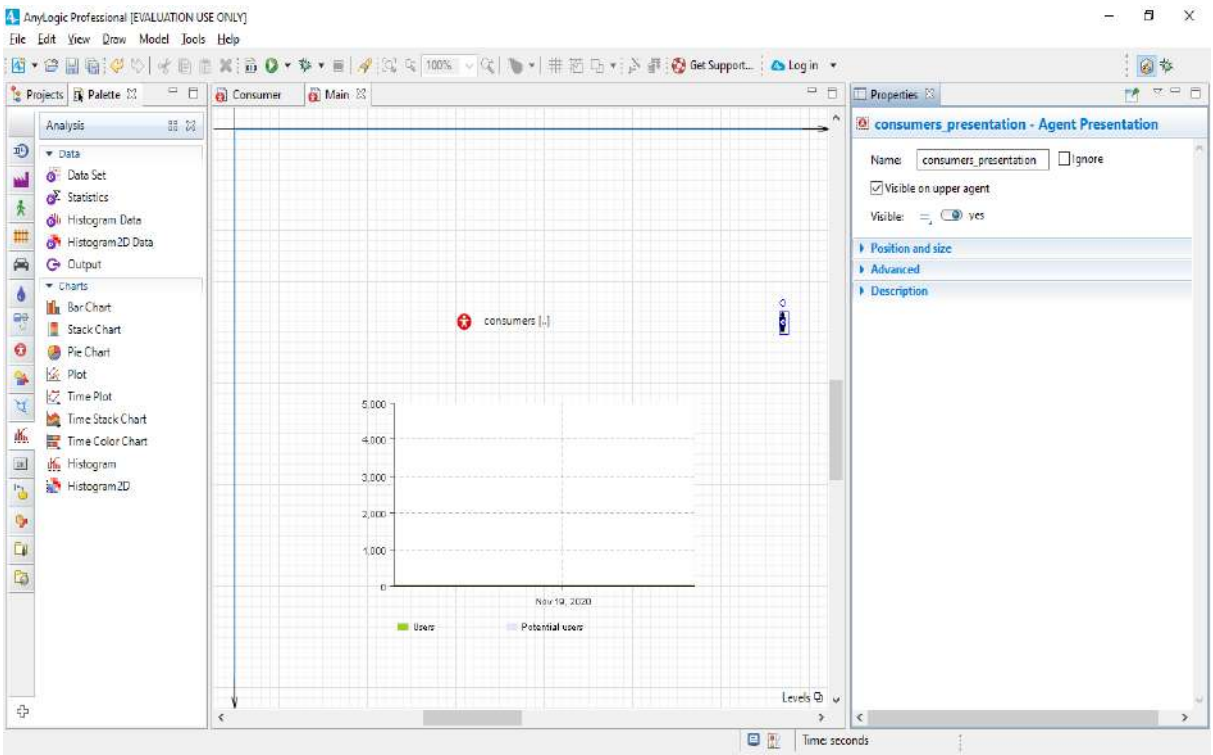
Now that we've set the time window, change the maximum number of data samples that the chart displays by navigating to the section Data update and setting Display up to 365 latest samples. Since we'll add one data sample each day, 365 data sample is an ideal amount for a one year range.



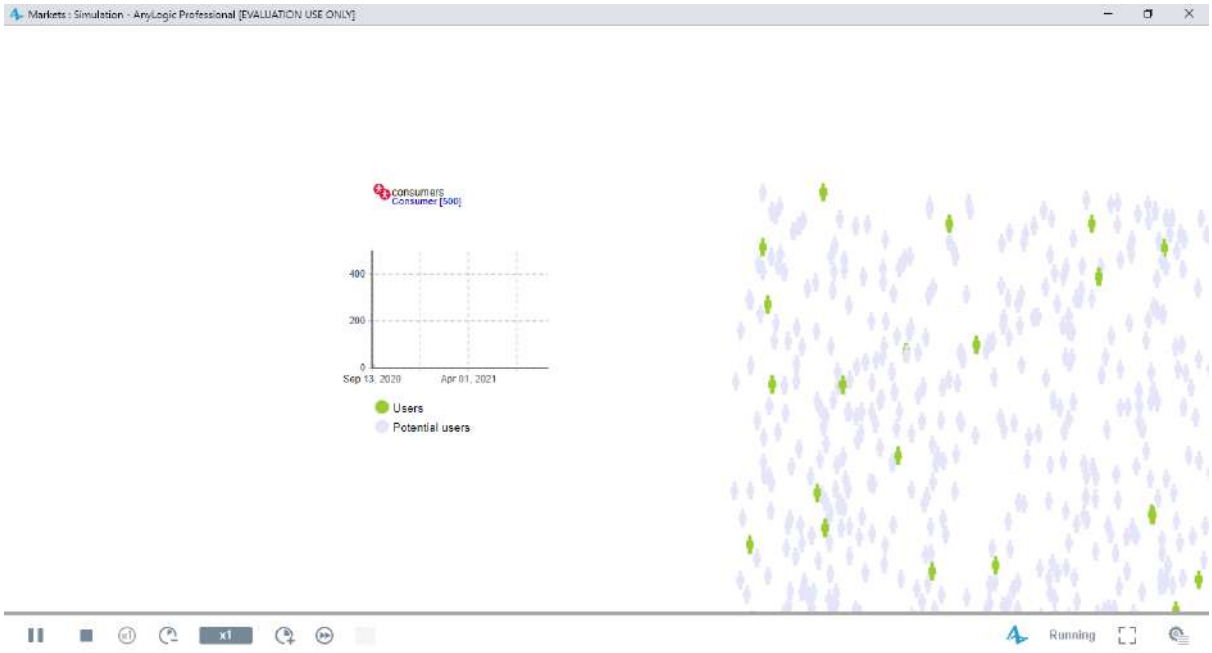
Go to the time stack chart's Appearance properties and set it to display Model date (date only) near the time axis.

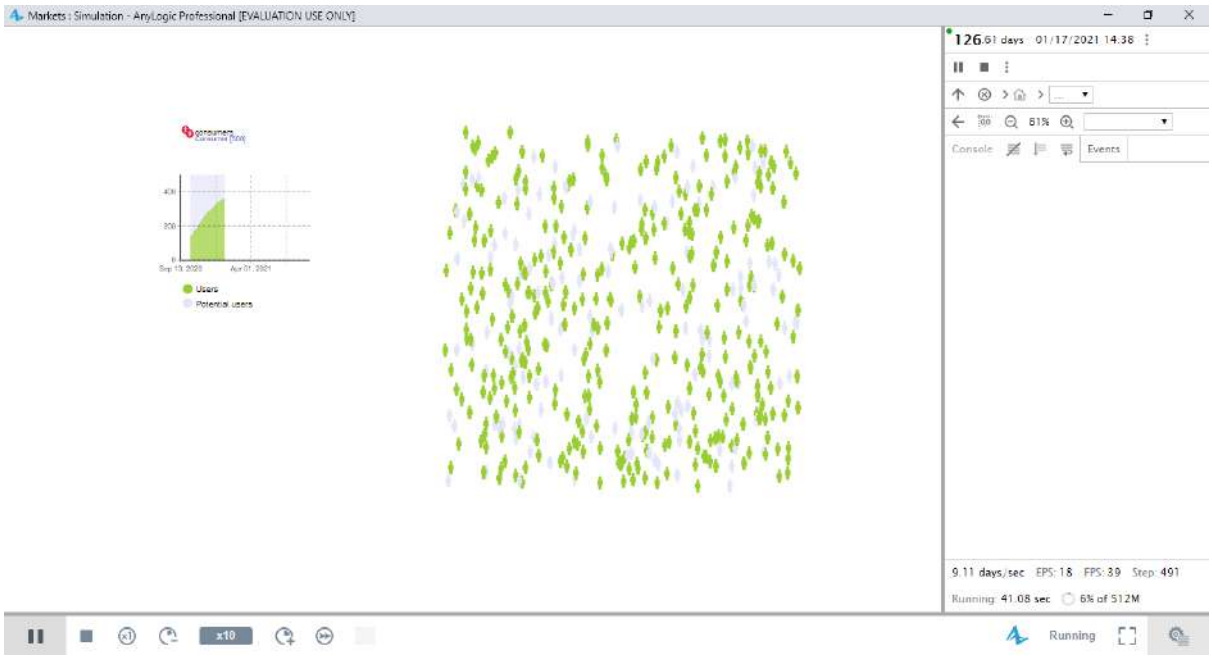


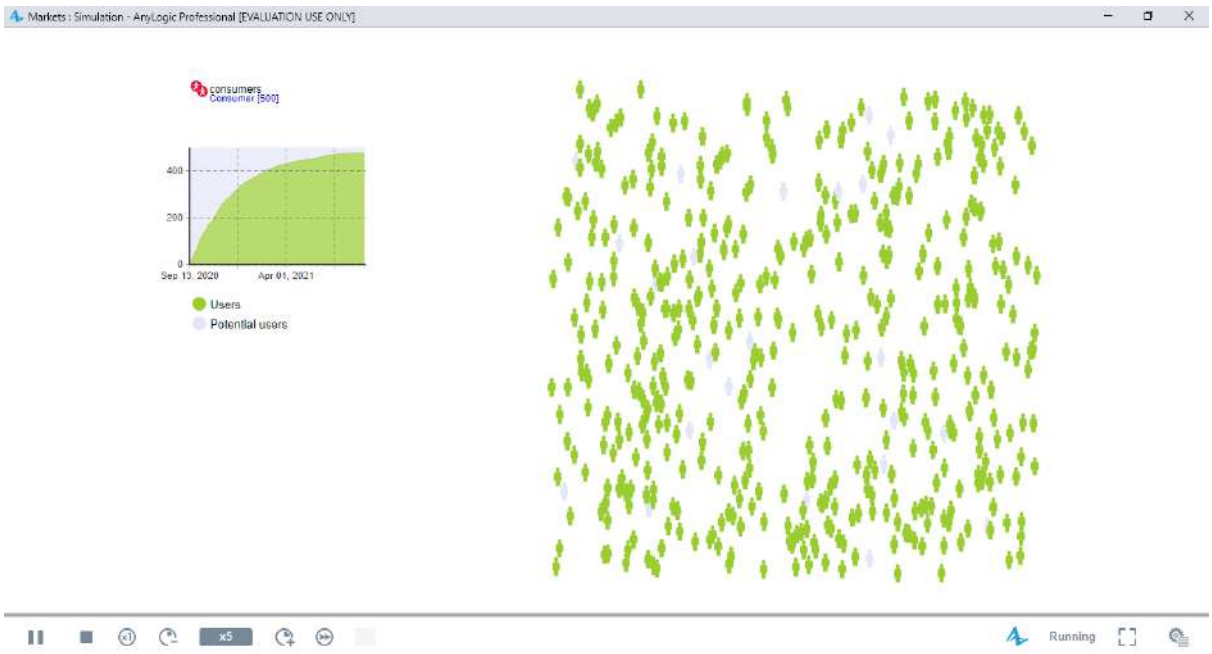
On the Main diagram, move the presentation of the consumers agent population to the right.

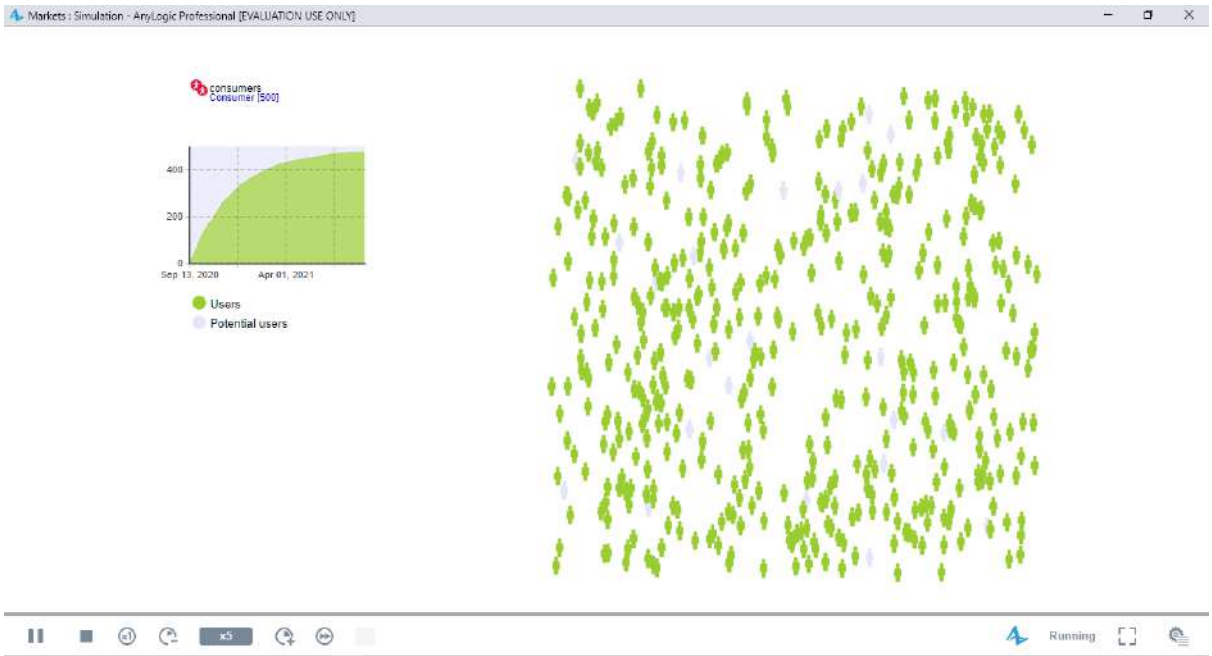


Run the model and use the time stack chart to review the process.






Markets : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
consumers
Consumer [500]
400
200
0
Sep 13, 2020
Apr 01, 2021
Users
Potential users
x5
Running


Markets : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
consumers
Consumer [500]
400
200
0
Sep 13, 2020
Apr 01, 2021
Users
Potential users
x5
Running

PRACTICAL NO: 02

Aim: Design and develop agent based model by

- Creating the agent population
- Defining the agent behavior
- Add a chart to visualize the model output.
- Adding word of mouth effect
- Considering product discards
- Considering delivery time

[Use a case scenario like restaurant].

Code:

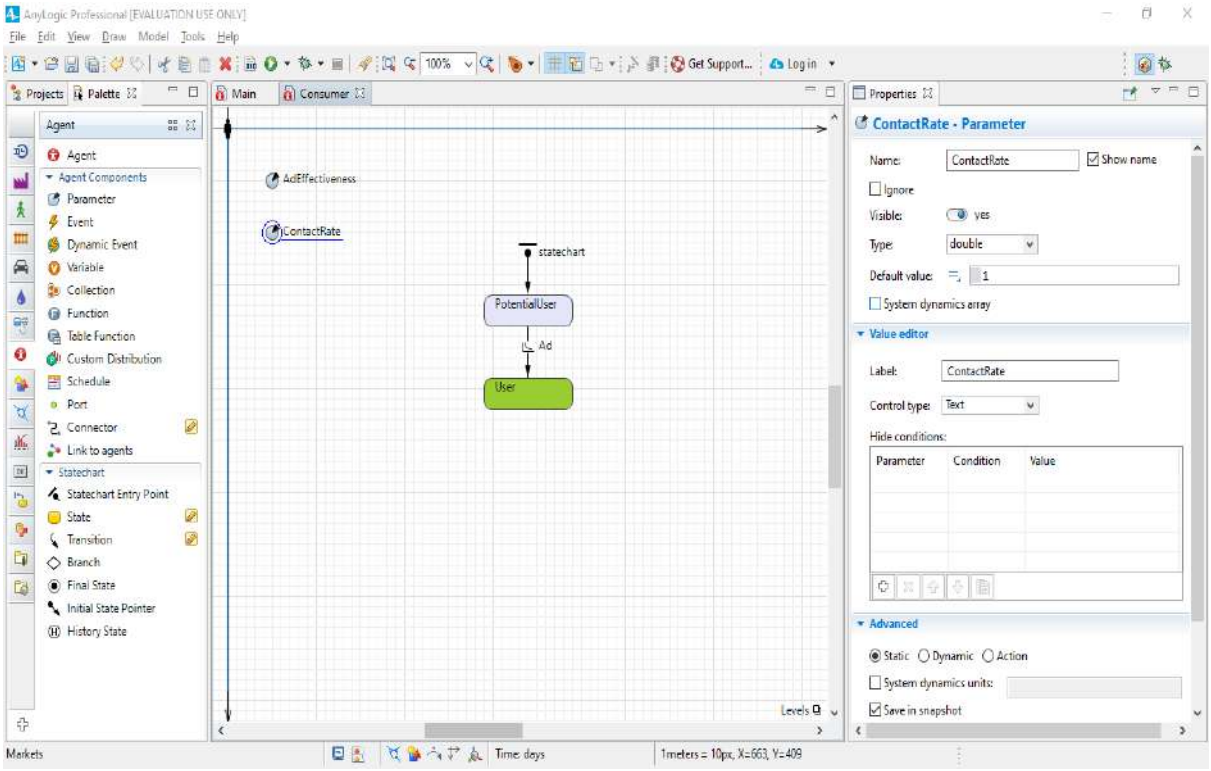
Adding word of mouth effect

In the Projects tree, open Consumer diagram by double-clicking on Consumer.

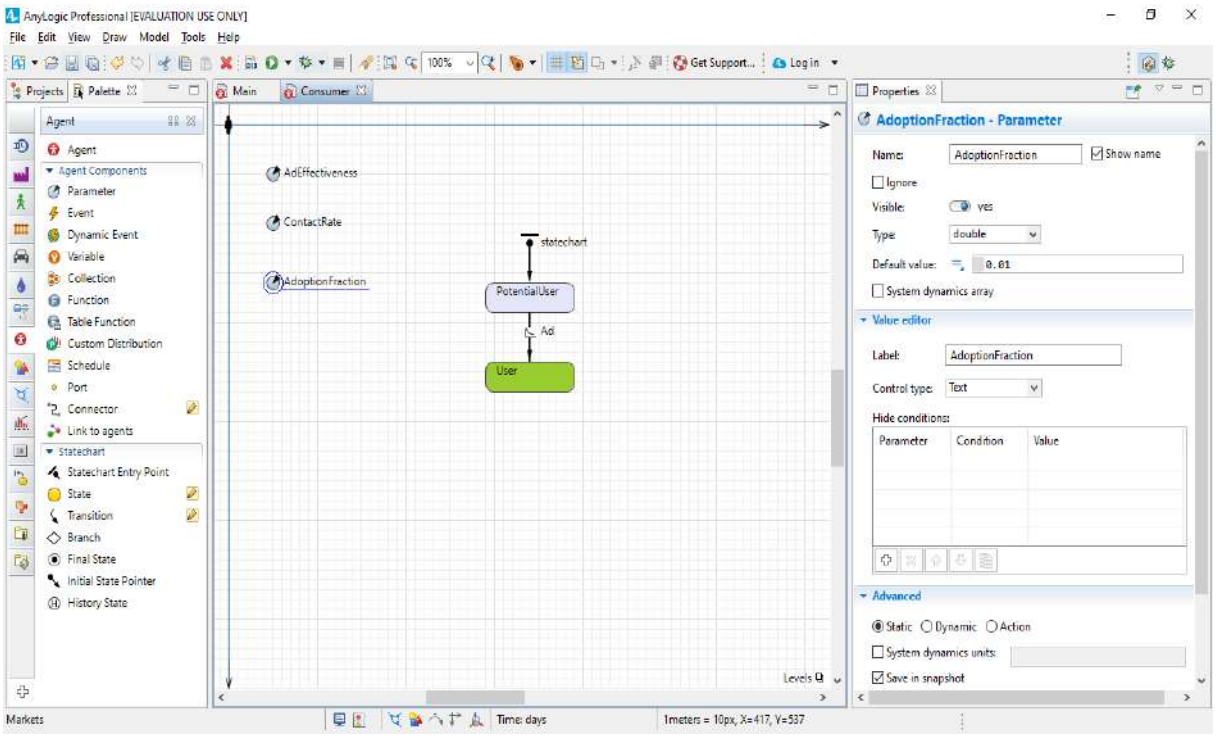
Add a parameter to define a consumer's average daily contacts. Drag the Parameter from the Agent palette on to the diagram.

The rate is 1 contact per day, so type 1 as the parameter's diagram.

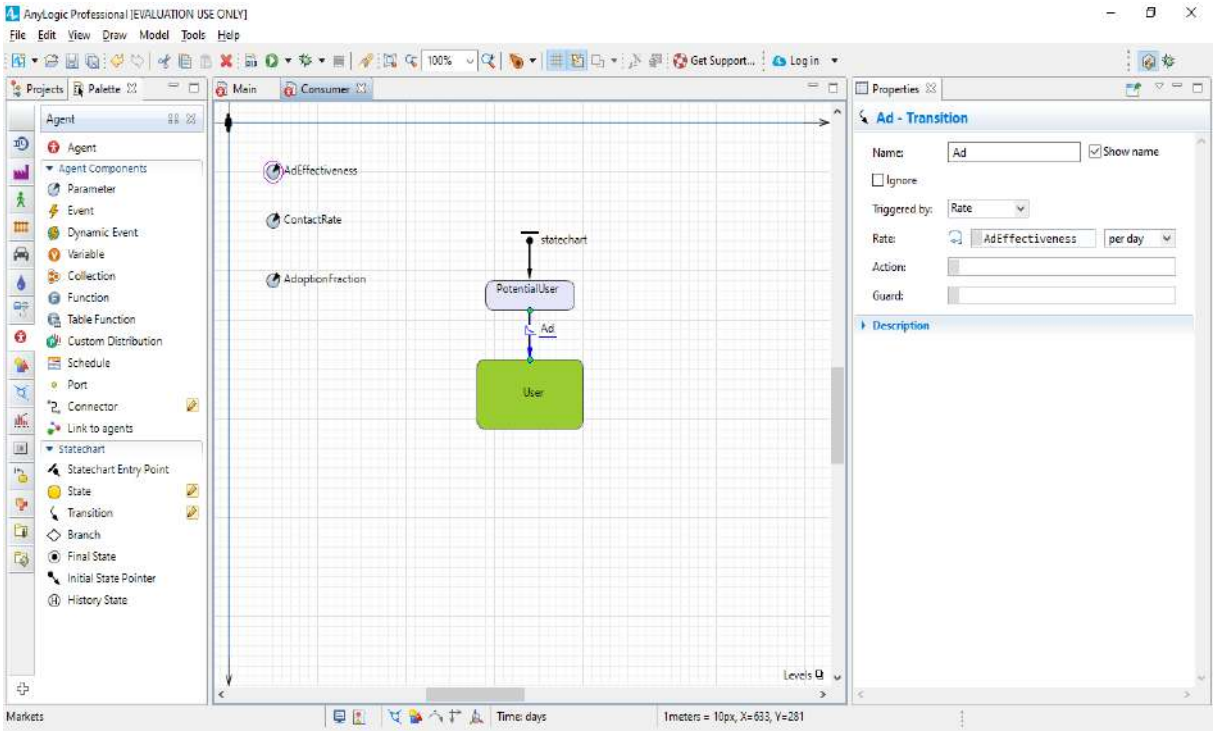
Name the parameter ContactRate default value.



Add another parameter - AdoptionFraction - to define a person's influence on others, a number that we'll express as the percentage of people who will use the product after they come into contact with the consumer. Leave the default parameter's Type: double and set the Default value: 0.01.The Consumer diagram should look like this:

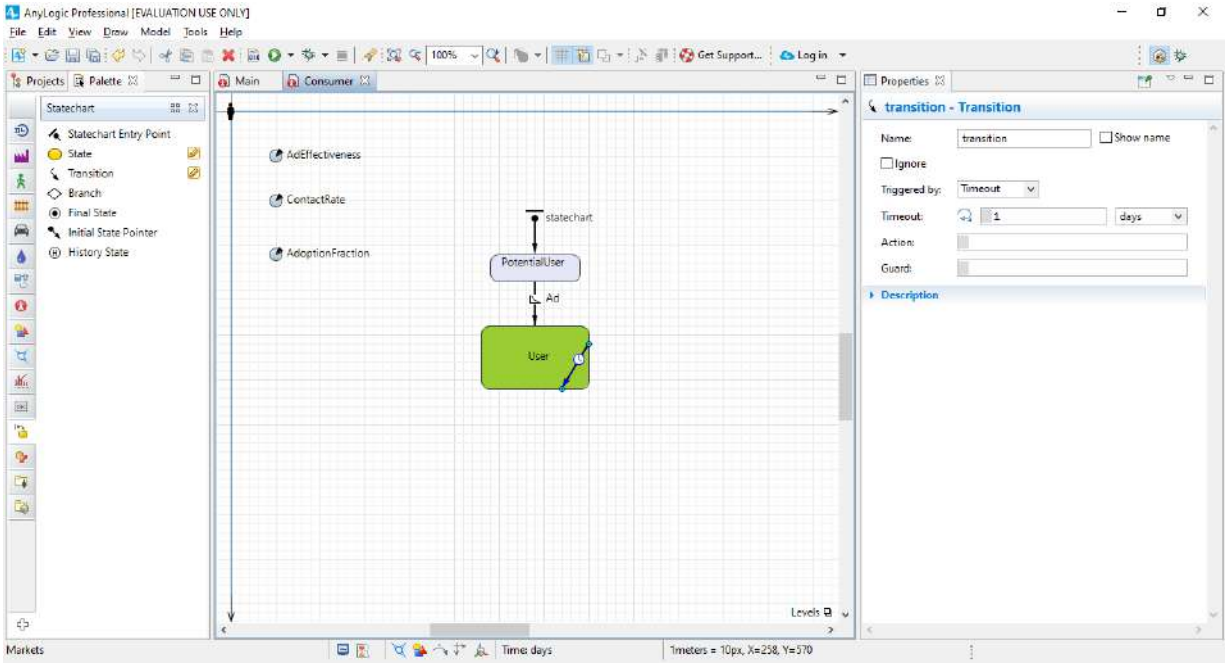


Open the Consumer diagram and increase the User state to fit the internal transition we'll draw inside the state on the next step.



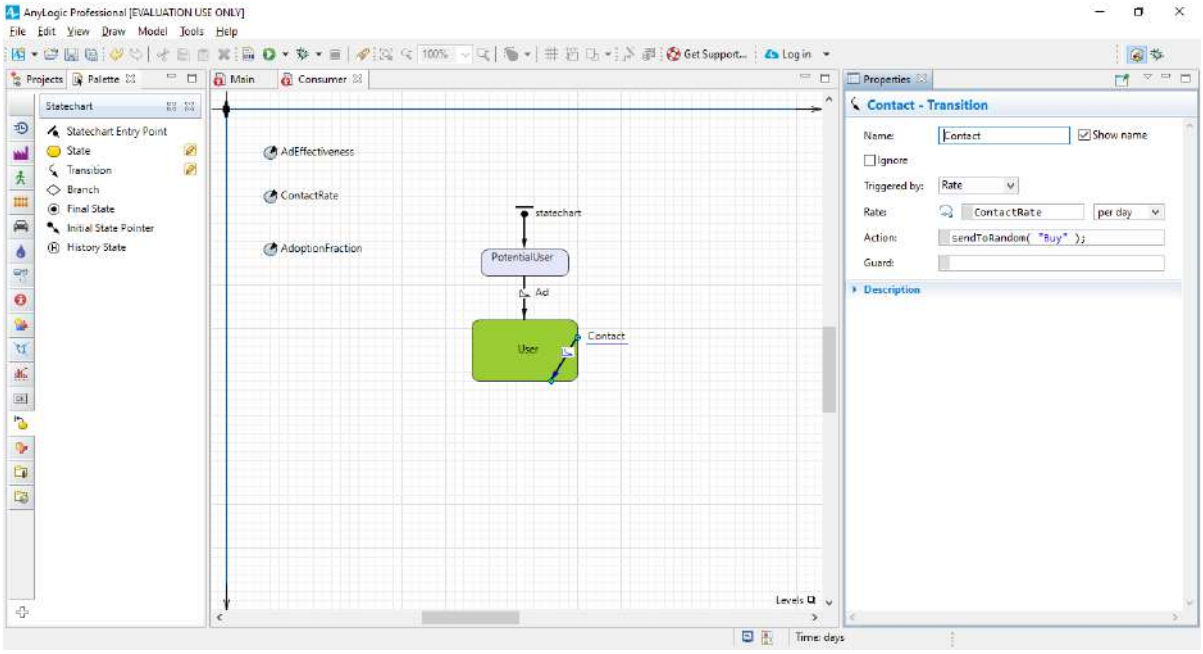
Draw an internal transition inside the User state. To draw a transition like the one shown below, drag the Transition from the Statechart palette inside the state so the transition's start point lies on the state border.

Afterward, you can move the transition end point to another point on the state border. To add a salient point, double-click the transition.



Modify the transition properties. This transition will occur with the specified Rate ContactRate (use code completion rather than typing the parameter's full name). Name the transition Contact and set it to show its name.

Specify the Action that will be executed on triggering this transition (use the code completion to write the code):sendToRandom("Buy");



Draw another transition from PotentialUser to User state, and name it WOM (Word of Mouth). This transition will model purchases caused by word of mouth.

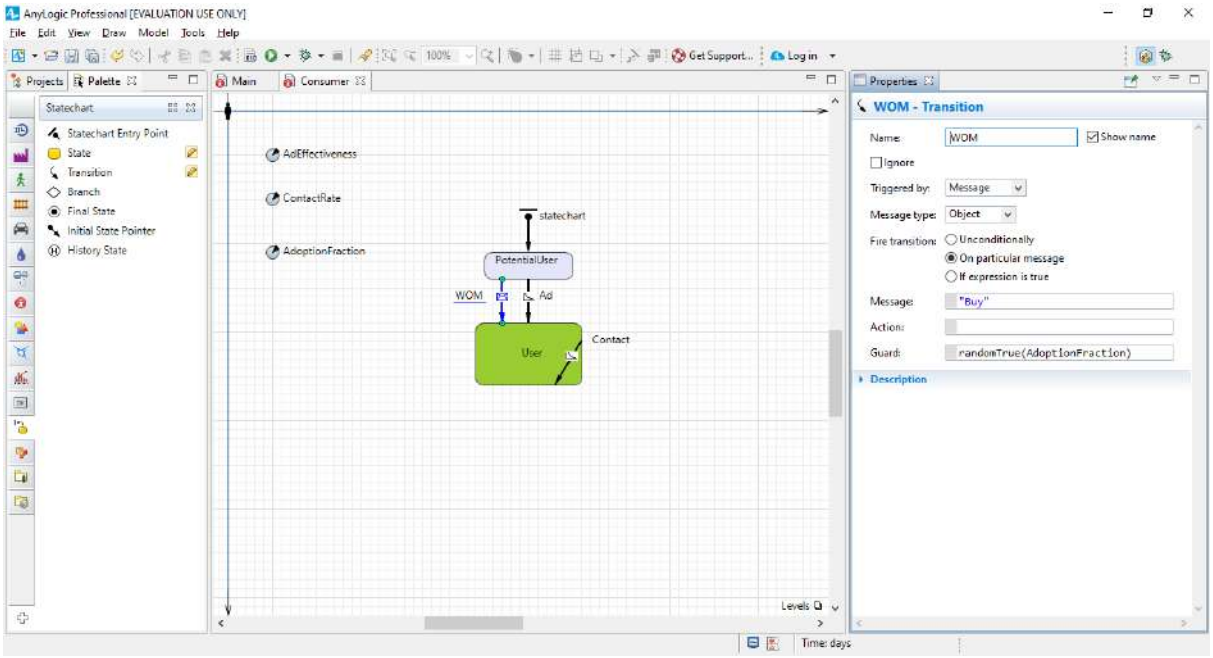
Modify the transition properties:

• In the Triggered by list, click Message.

• In the Fire transition area, select on particular message.

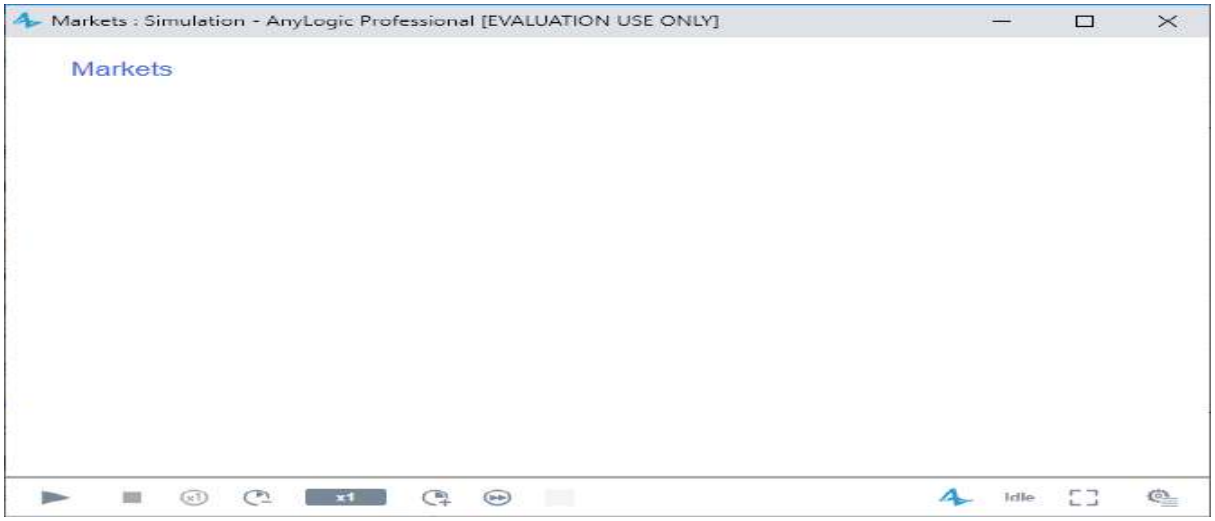
• In the Message field, type "Buy"

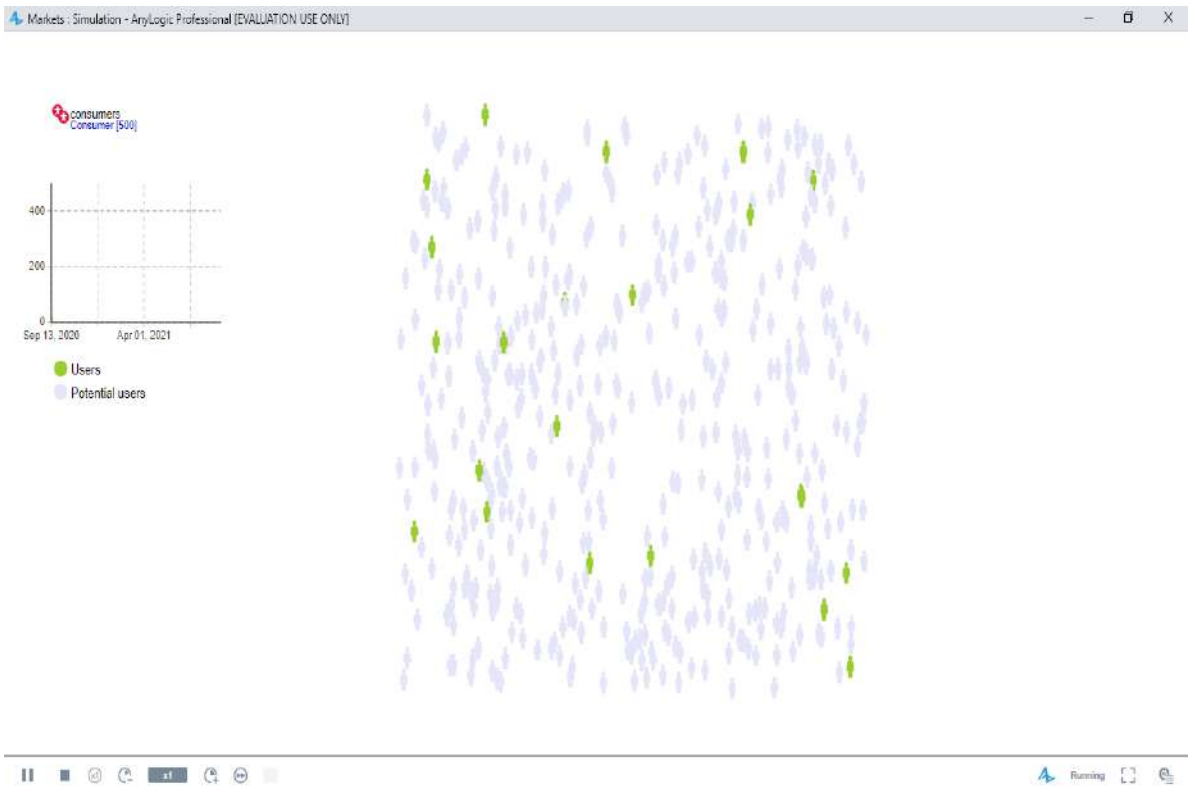
• Since we know not every contact is successful – in other words, a contact may not convince the potential user to buy our product – we'll use AdoptionFraction to make successful contacts less common. Specify the transition's Guard: randomTrue(AdoptionFraction)

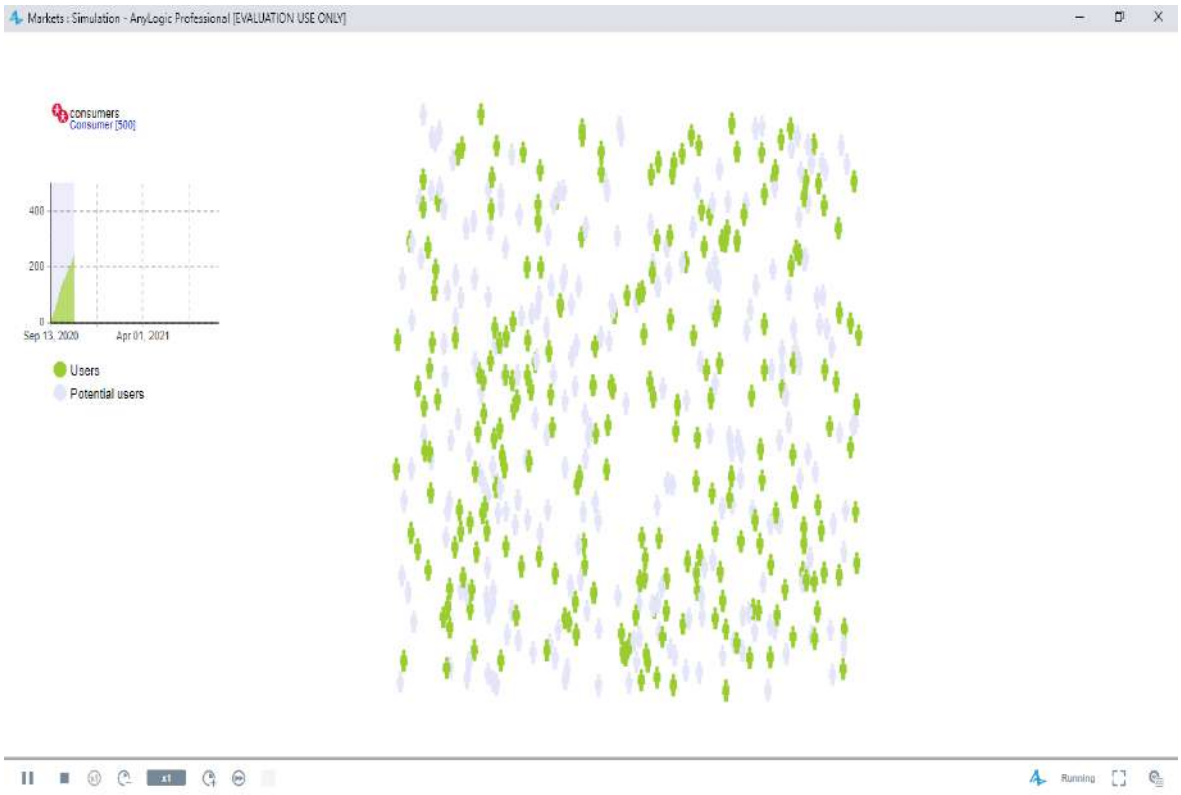


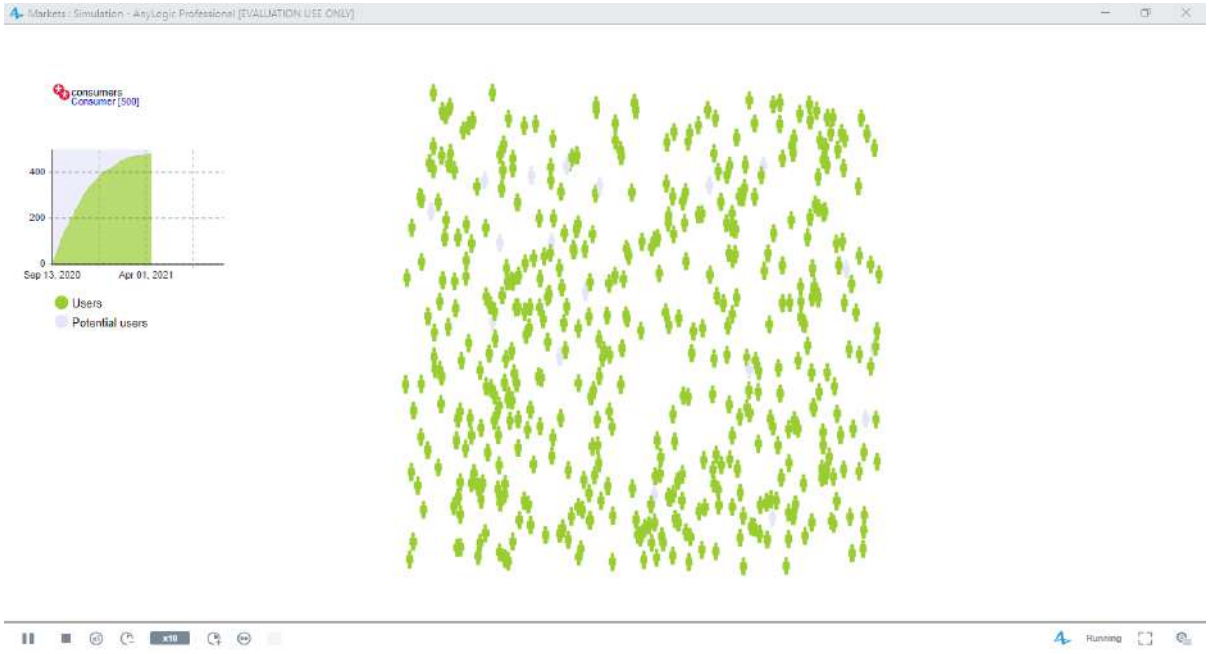
In the Projects view, you may see an asterisk near the model item that shows your model has unsaved changes. On the toolbar, click Save to save your model.

Run the model.



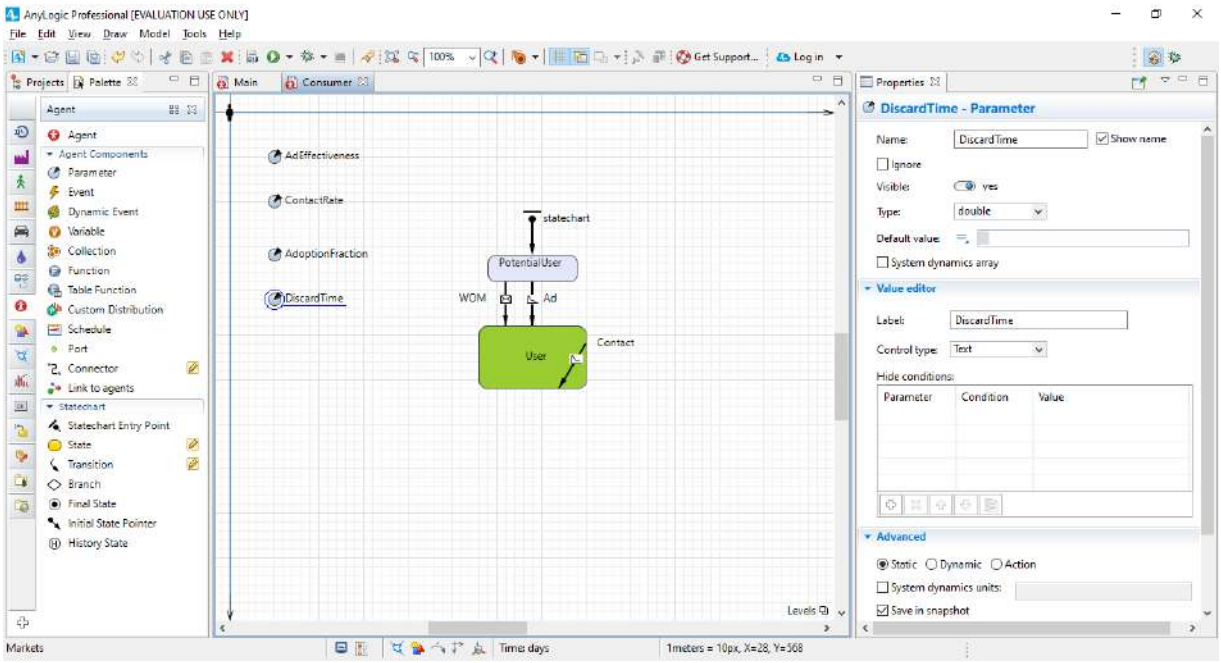

Markets : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
consumers
Consumer [500]
400
200
0
Sep 13, 2020
Apr 01, 2021
Users
Potential users
Running


Markets : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
consumers
Consumer [500]
400
200
0
Sep 13, 2020
Apr 01, 2021
Users
Potential users
Running



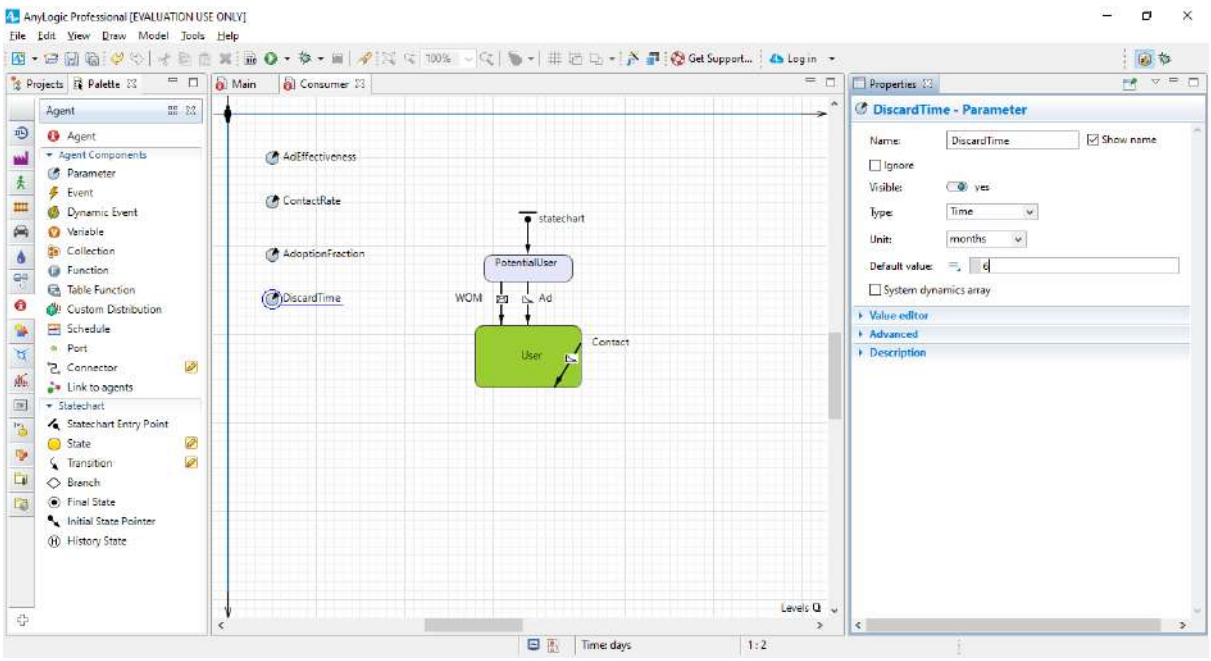
Considering Product Discards

Open the Consumer diagram and add a DiscardTime parameter



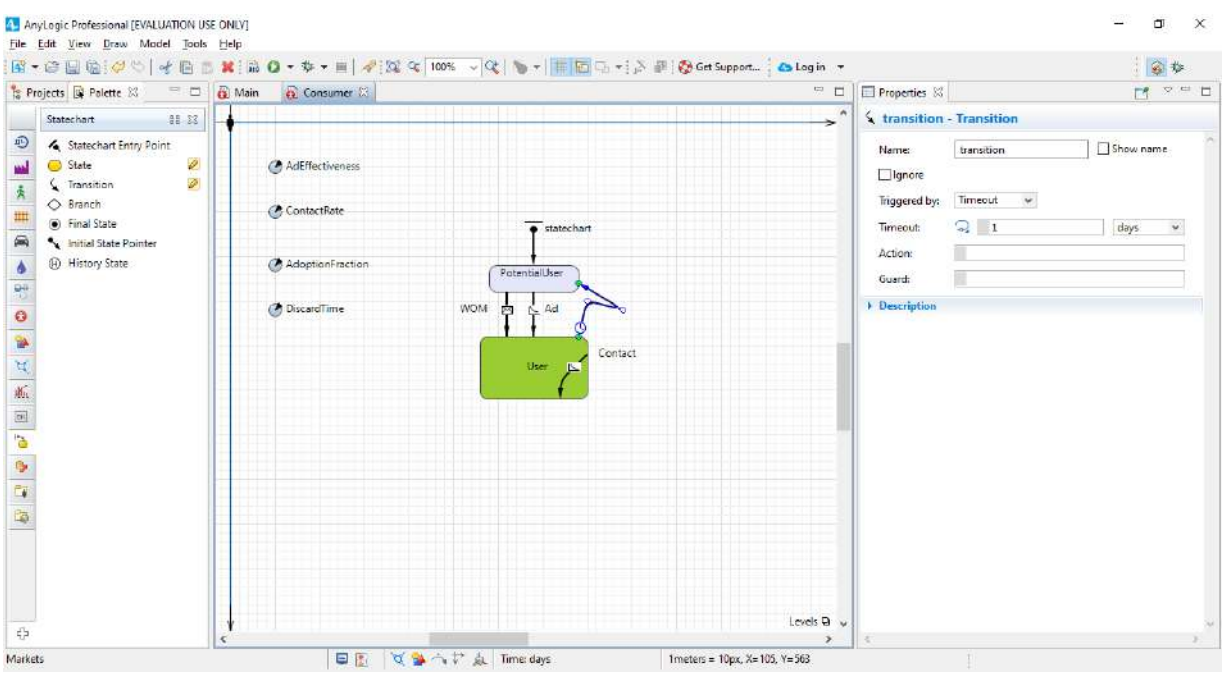
This parameter will define our product's lifespan.

Choose Time as the parameter's Type, click months in the Unit list, and type 6 as the Default value.

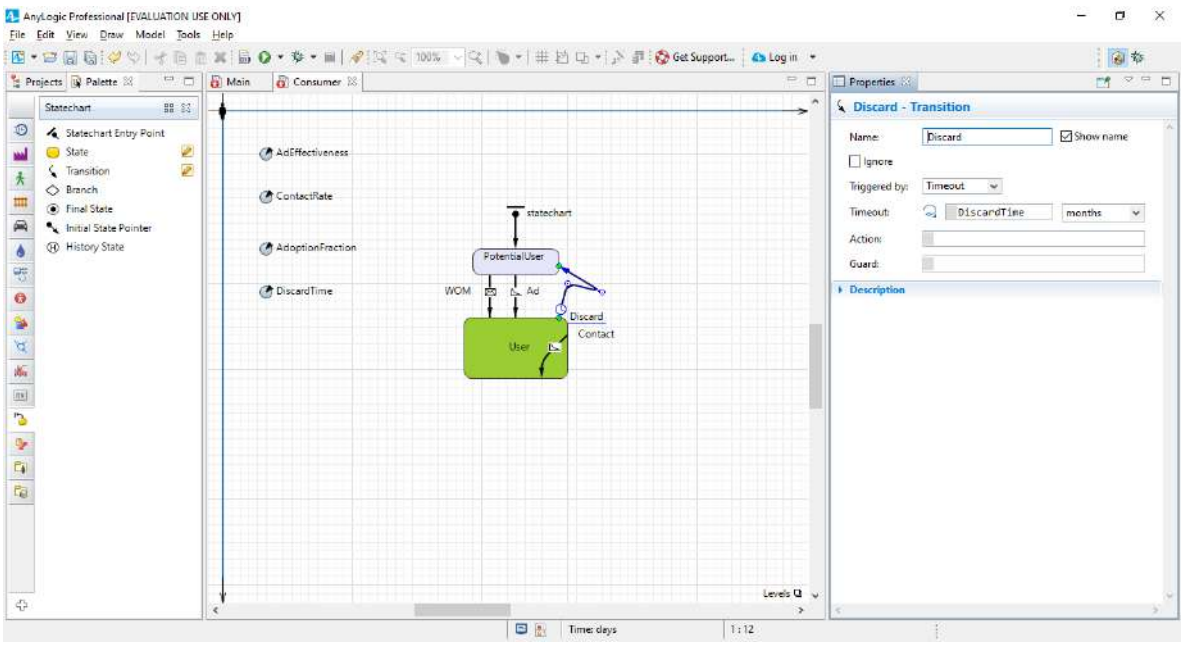


Draw a transition from User to PotentialUser state to model product discards.

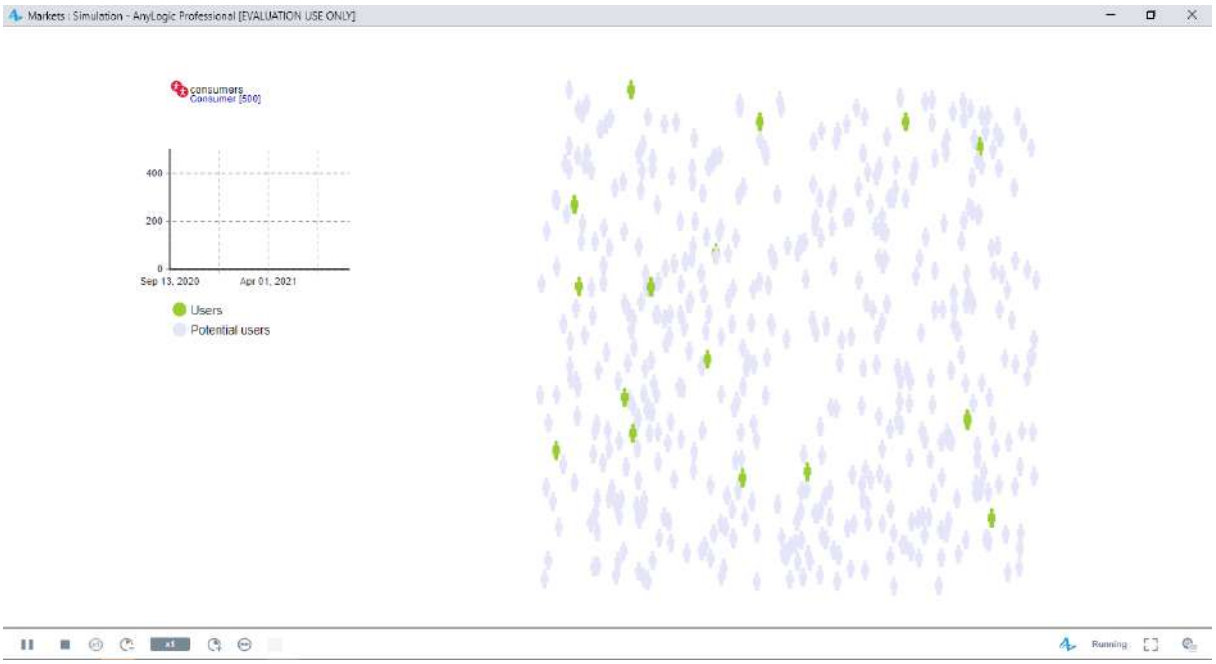
To draw a transition with salient points like those shown in the figure, double-click the Transition element in the Statechart palette (this should change the element's icon in the palette to), click the transition's source state User, click at the salient point places, and click the target state PotentialUser.

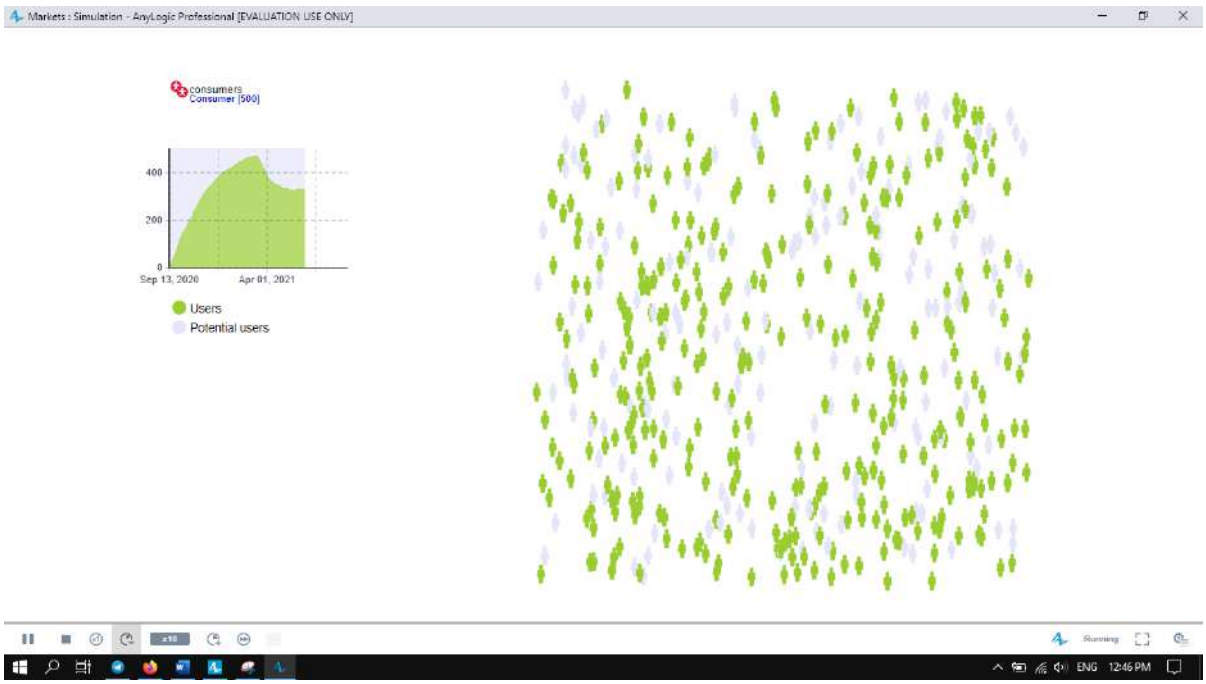


Name the transition Discard and set it to be triggered by a constant timeout DiscardTime. In the list to the right, click months.



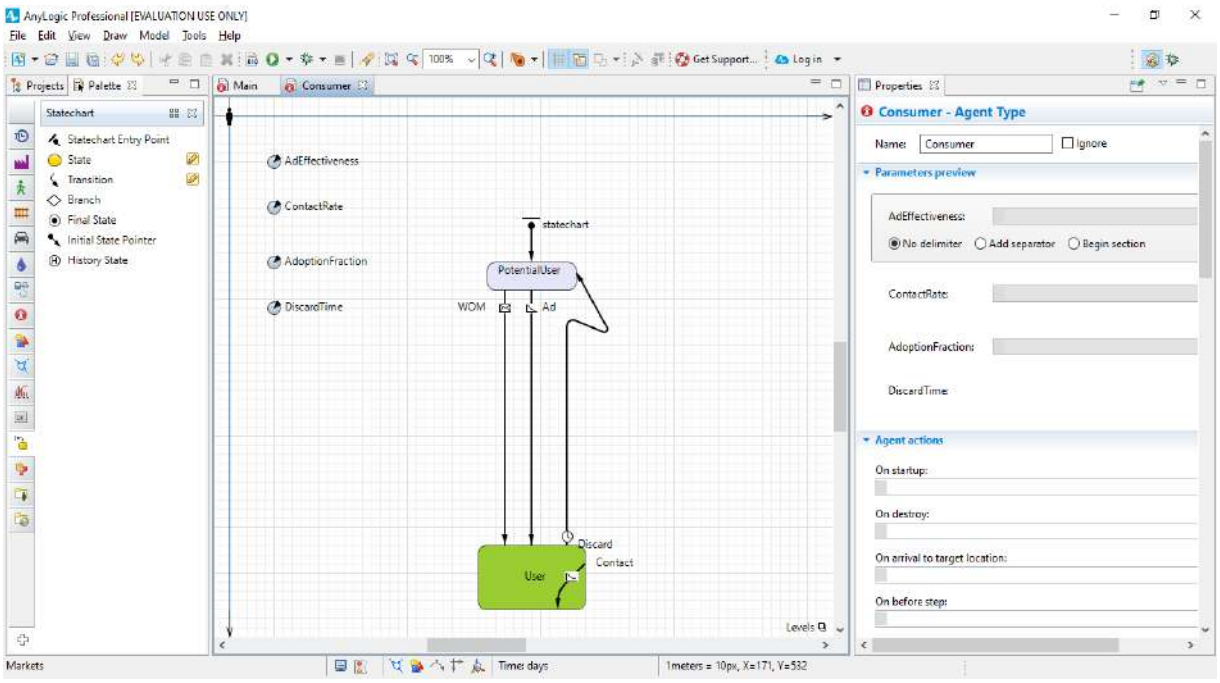
Run the model.



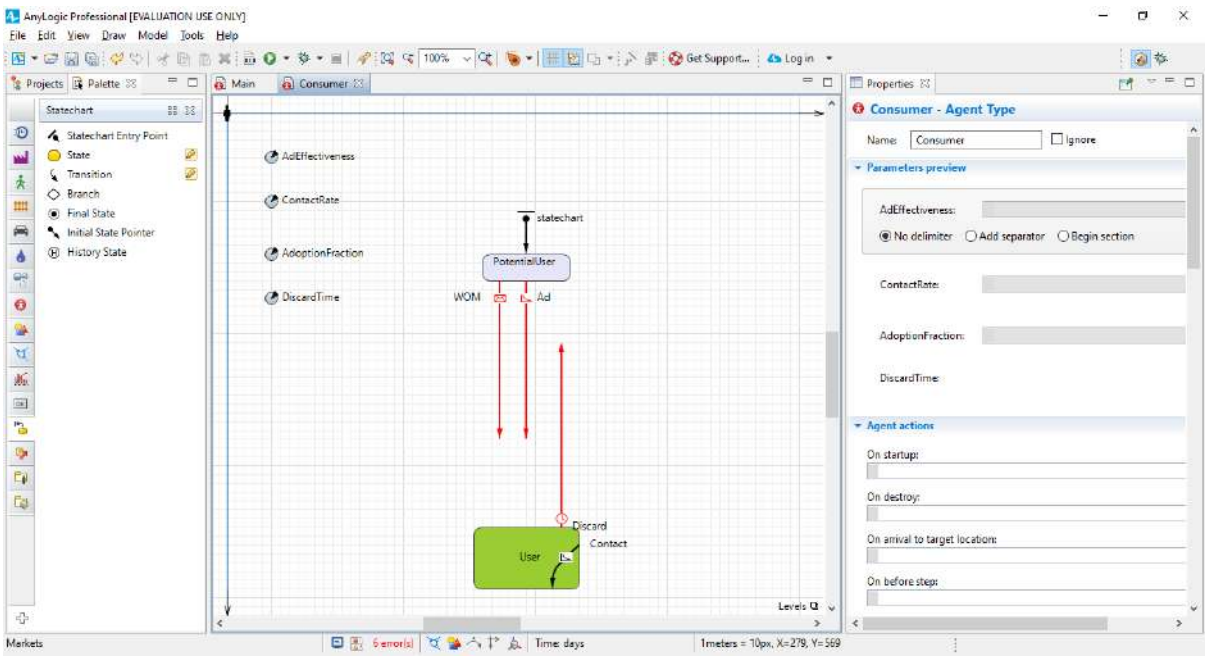


Considering Delivery Time

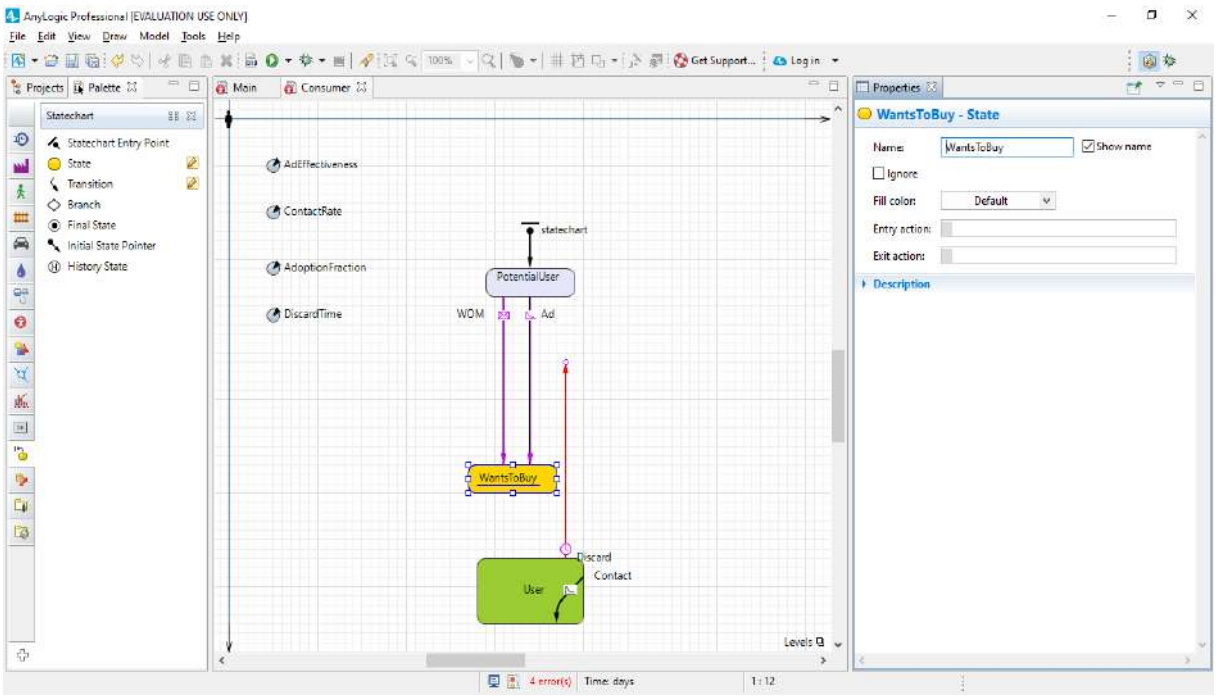
Prepare a place for another state between PotentialUser and User by moving the User state toward the bottom of the screen.



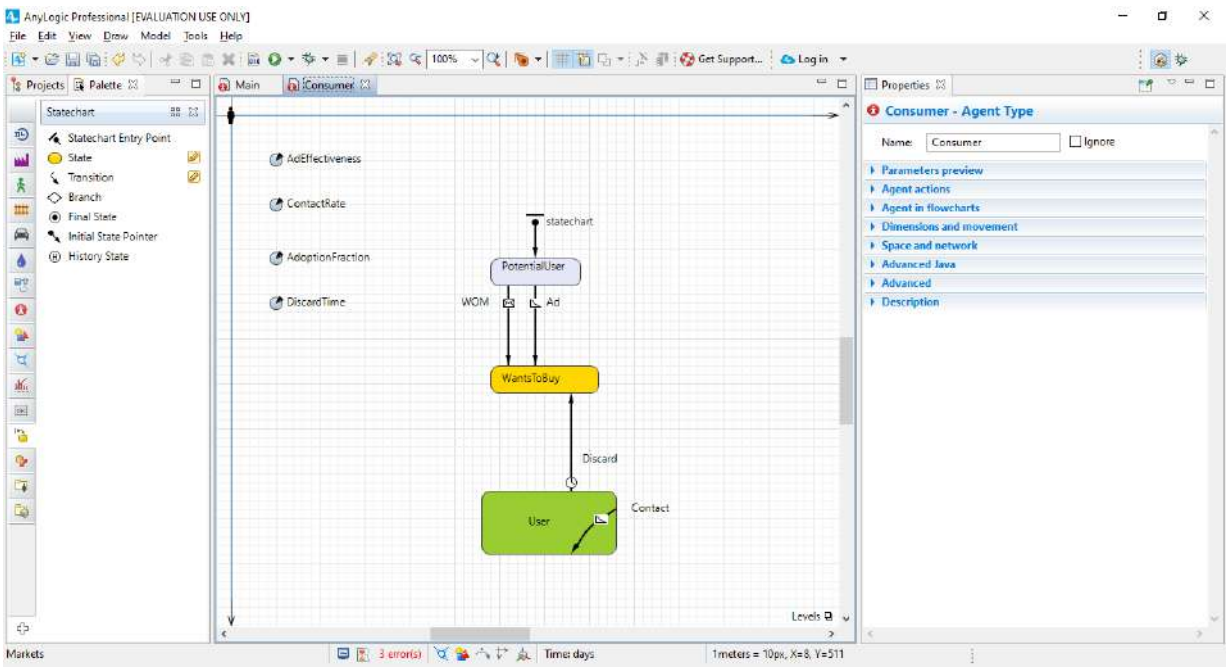
Disconnect the User state from the transitions. Select the WOM and Ad transitions, move their end points toward the top of the screen, and disconnect the discard transition from PotentialUser. Afterward, you'll notice the disconnected transitions are drawn in red.



Add another State from the Statechart palette to the middle of the consumer's statechart and name it WantsToBuy. Consumers in this state have decided to purchase the product, but they have not done so.



Reconnect transitions to the middle state: the WOM, Ad, and Discard transitions should now end in the WantsToBuy state.



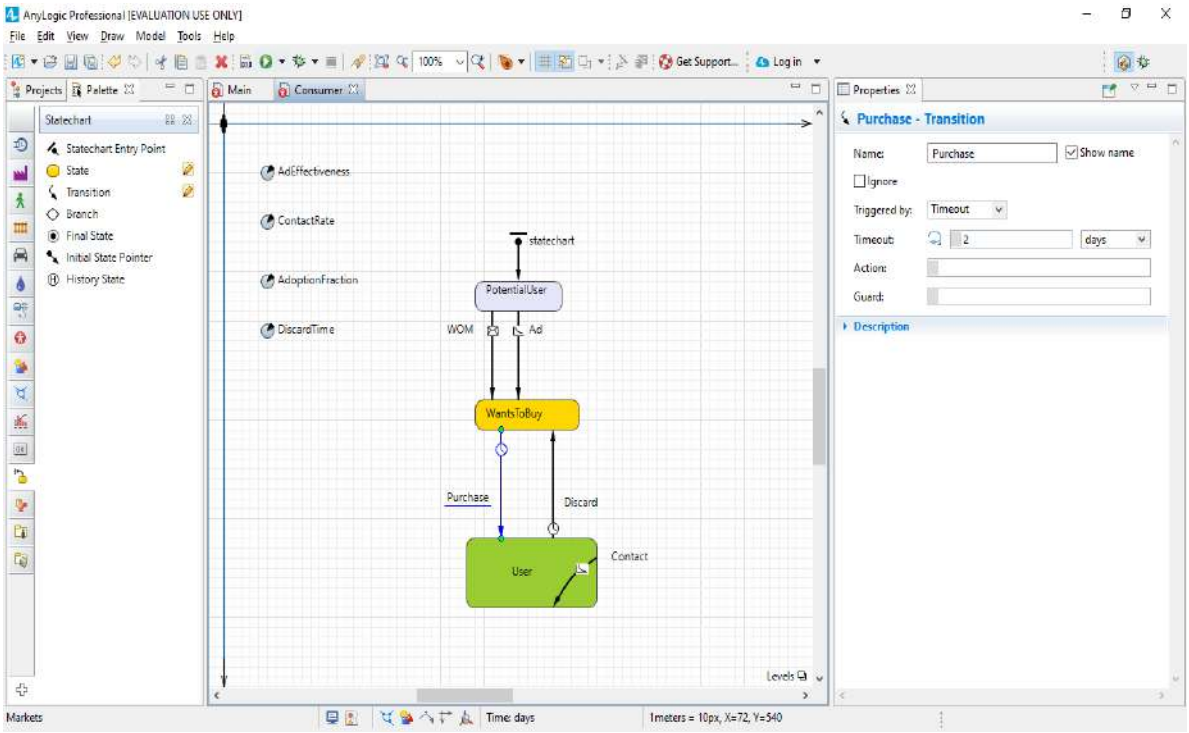
Modify WantsToBuy similar to other states:

Fill color: gold

Entry action: shapeBody.setFillColor(gold);

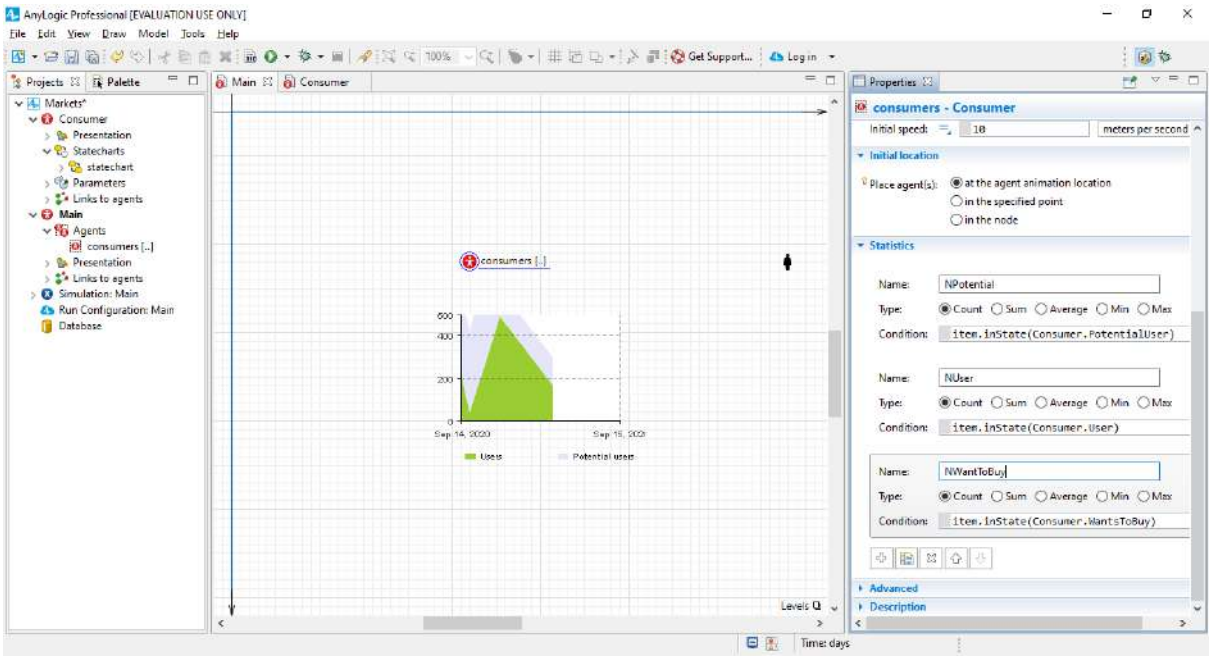
Add a transition from WantsToBuy to User state to model the product shipment and name it Purchase.

Let's assume it typically takes a user two days to get the product. This means once the consumer's statechart enters the state WantsToBuy, it will proceed to the state User with a two-day delay. With this in mind, set 2 days timeout for the Purchase transition:

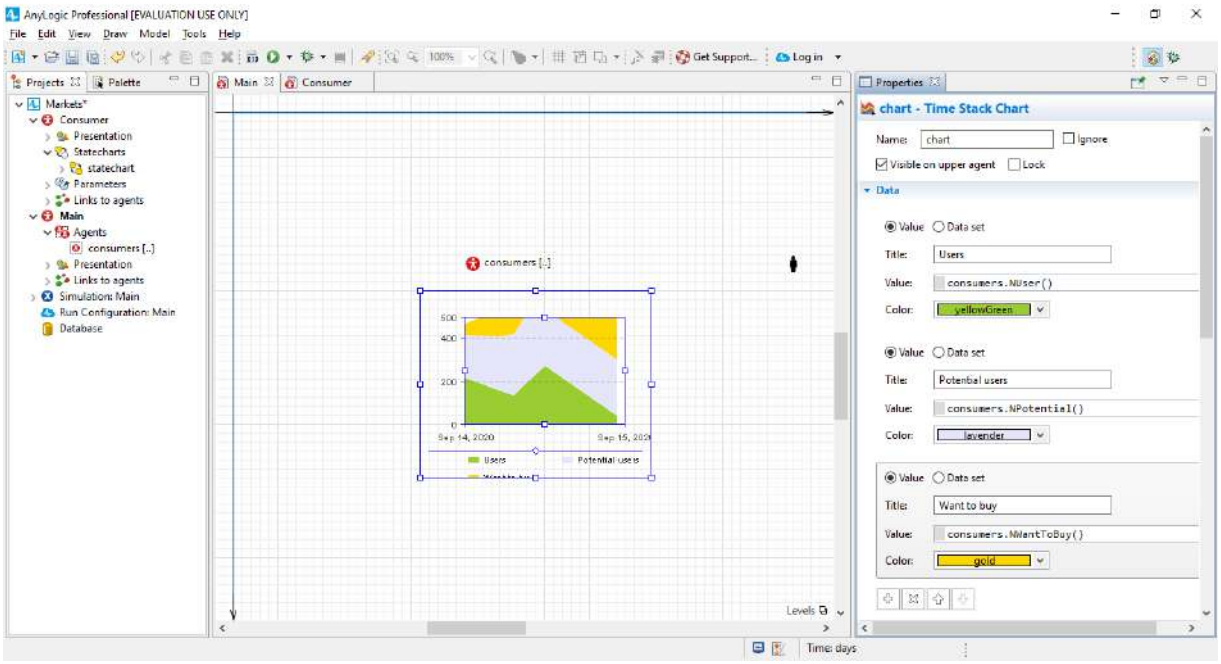


Define one more statistics function to count the product's market-driven demand.

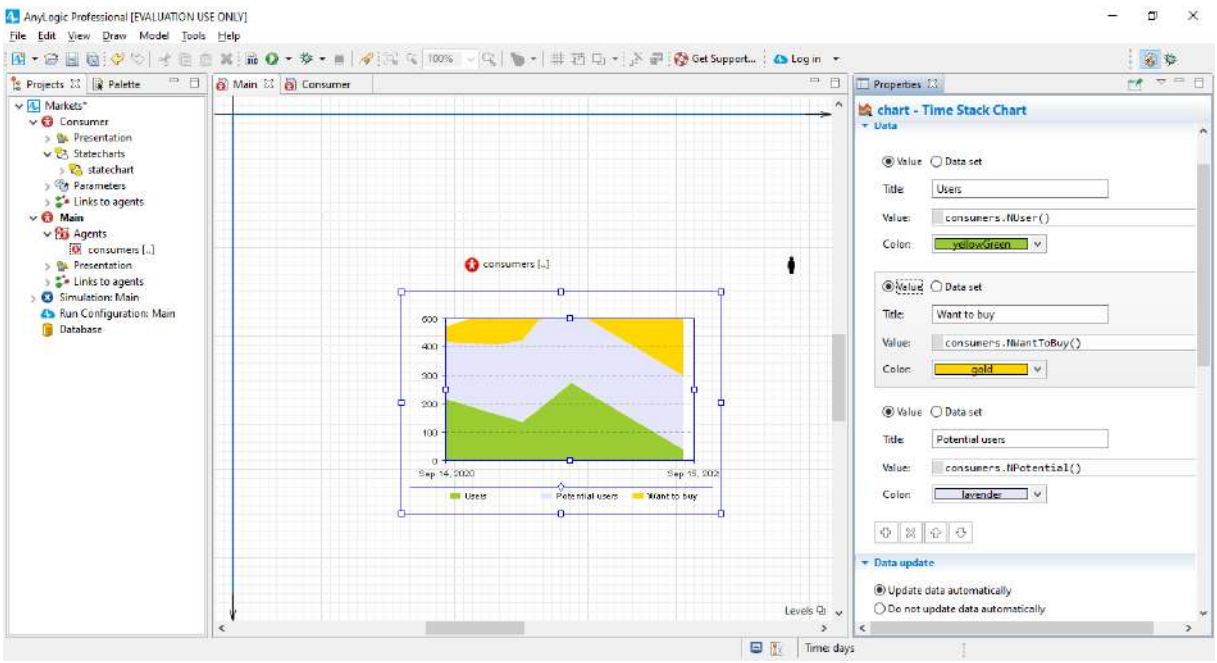
In the editor of Main, click the consumers, go to the Statistics properties section, and add a statistics item: NWantToBuy with condition item.inState(Consumer.WantsToBuy)



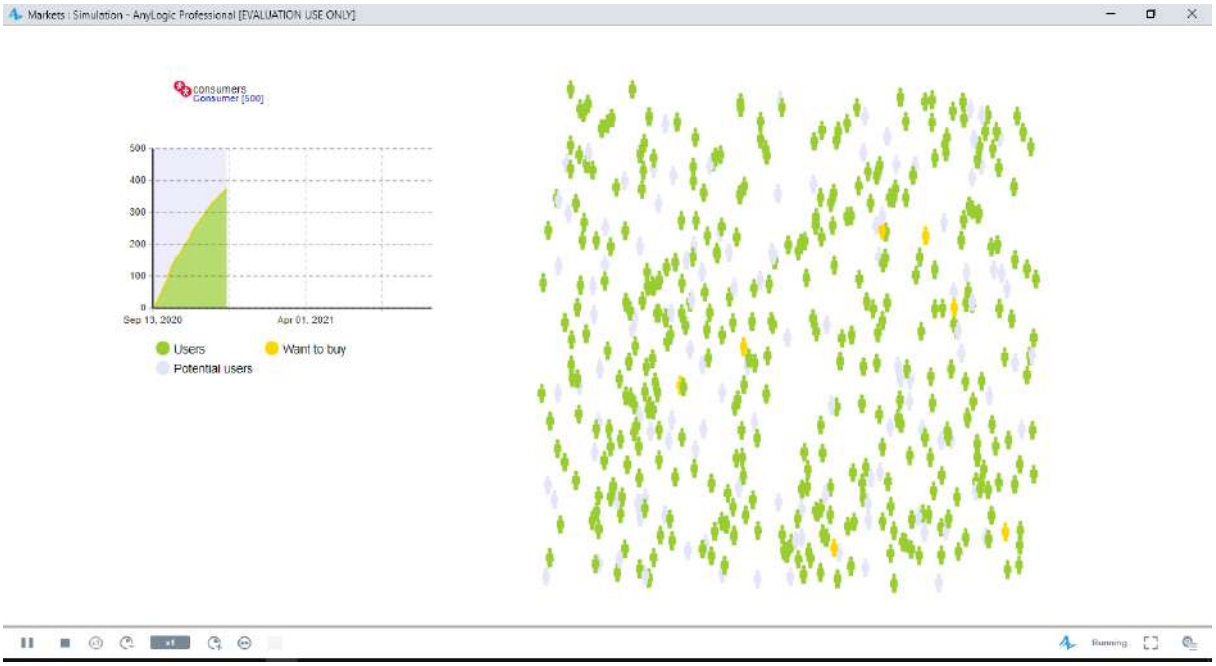
On Main, select the time stack chart, and add another data item to be displayed with the chart: consumers.NWantToBuy() in value with the title Want to buy and color gold.

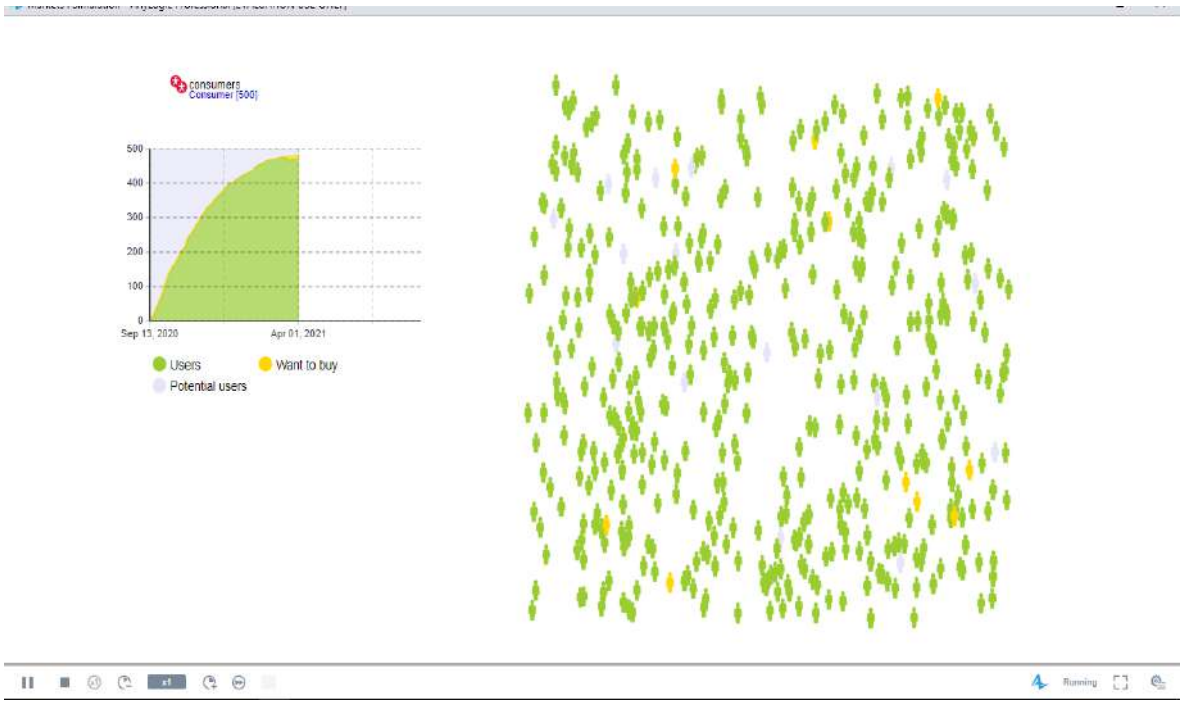


Make the newly-defined data item second in the list by selecting the item's section and clicking the "up" button.



Run the model, and you'll notice AnyLogic displays the number of consumers who are waiting for the product in yellow.




consumers
Consumer [500]
500
400
300
200
100
0
Sep 13, 2020
Apr 01, 2021
Users
Want to buy
Potential users
Running

PRACTICAL NO: 03

Aim: Design and develop agent based model by

- Creating the agent population
- Defining the agent behavior
- Adding a chart to visualize the model output.
- Adding word of mouth effect
- Considering product discards
- Considering delivery time
- Simulating agent impatience
- Comparing model runs with different parameter values

[Use a case scenario like market model].

Code:

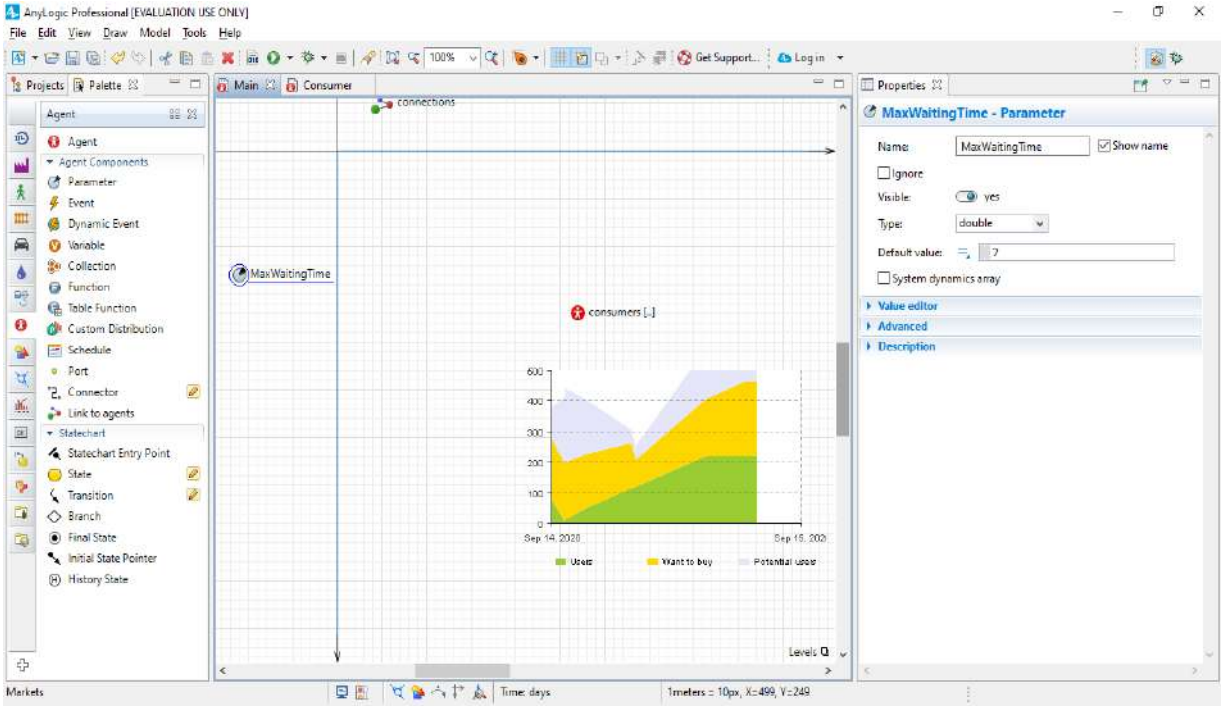
Open the Main agent type diagram.

Since we don't want the model window to display the model's parameters at runtime, we can place them outside the model window's default display area.

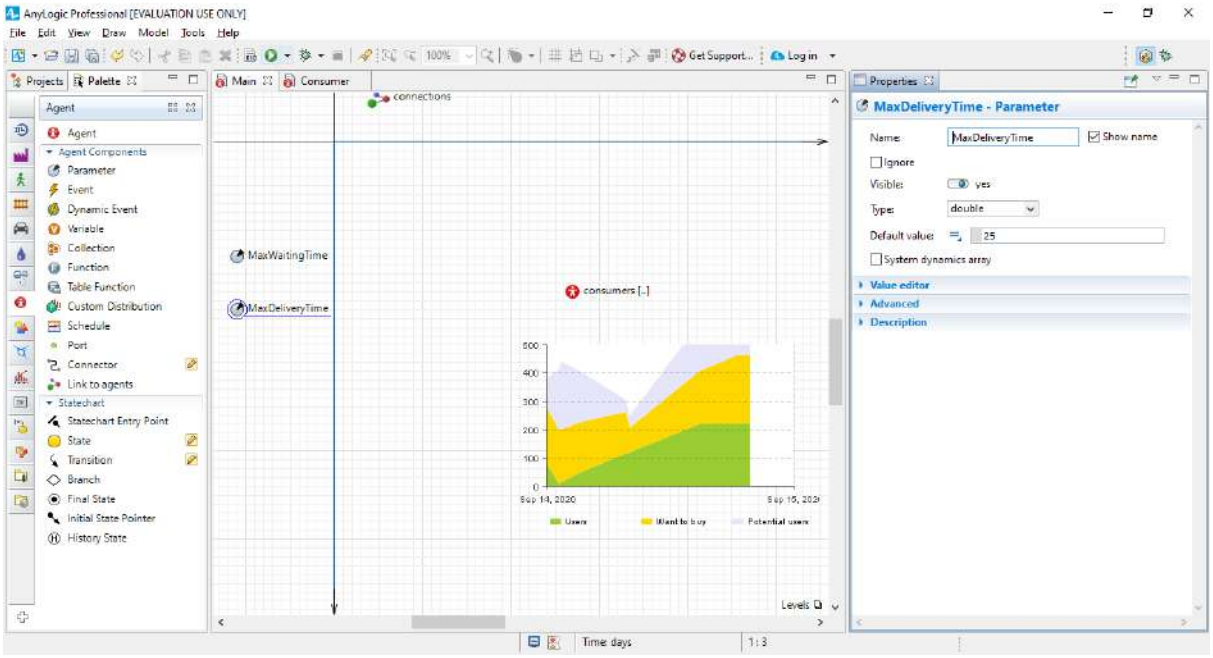
On Main, the model window is depicted with a blue rectangular frame. Elements inside the frame will be visible at the model runtime, but you can hide them by moving the graphical diagram's canvas slightly to the right and placing two parameters as shown in the figure below.

To move the graphical diagram's canvas, hold down the right mouse button as you move the mouse.

Configure the parameters. MaxWaitingTime defines the maximum time a consumer will wait for the product (in this case, seven days).



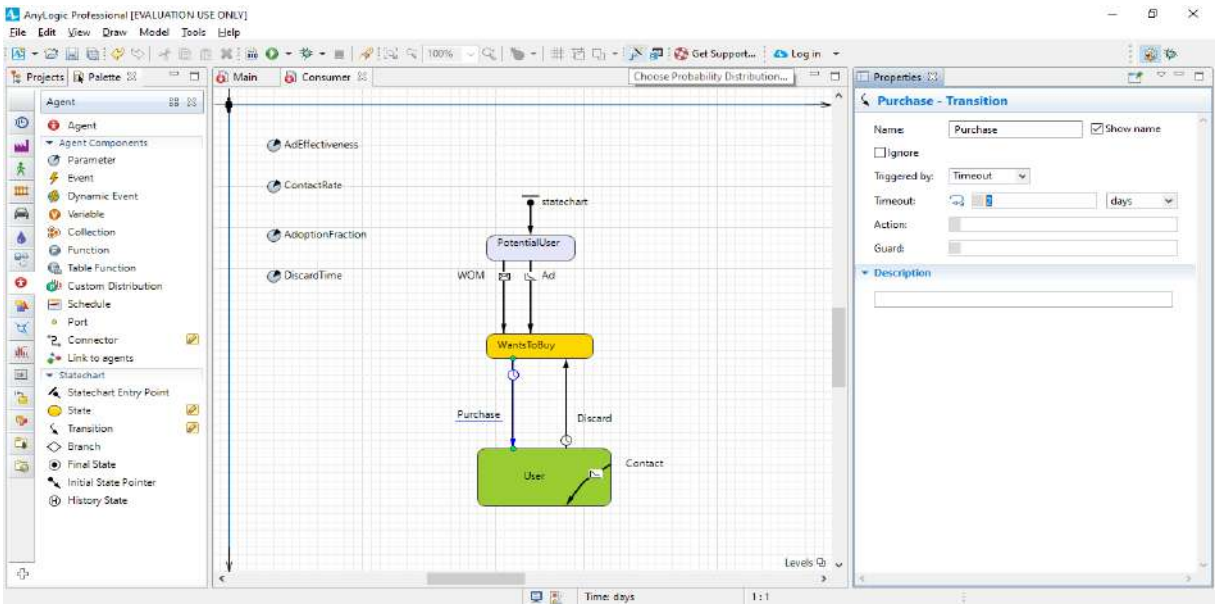
Set the other parameter, MaxDeliveryTime to 25 days to reflect our assumption it may take up to 25 days to deliver a product.



We assume it takes between one and 25 days – with an average of two days – to deliver the product. With that in mind, let's change the delivery time from a fixed two day delivery period to the stochastic expression that describes this pattern.

Open the Consumer diagram and select the Purchase transition. We want to change the transition's timeout expression, and we'll do that by using a wizard to choose the distribution function and insert the function's name in the property. To substitute the existing value, use your mouse to select the existing Timeout expression.

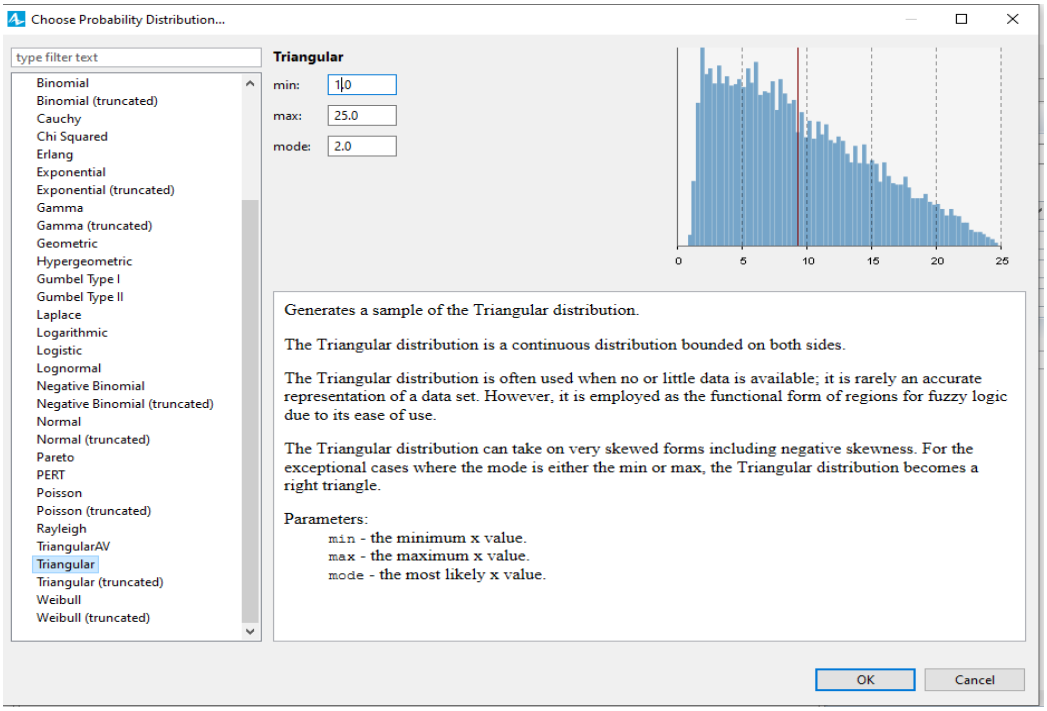
Click the Choose Probability Distribution... button.



You'll see the Choose Probability Distribution... dialog box.

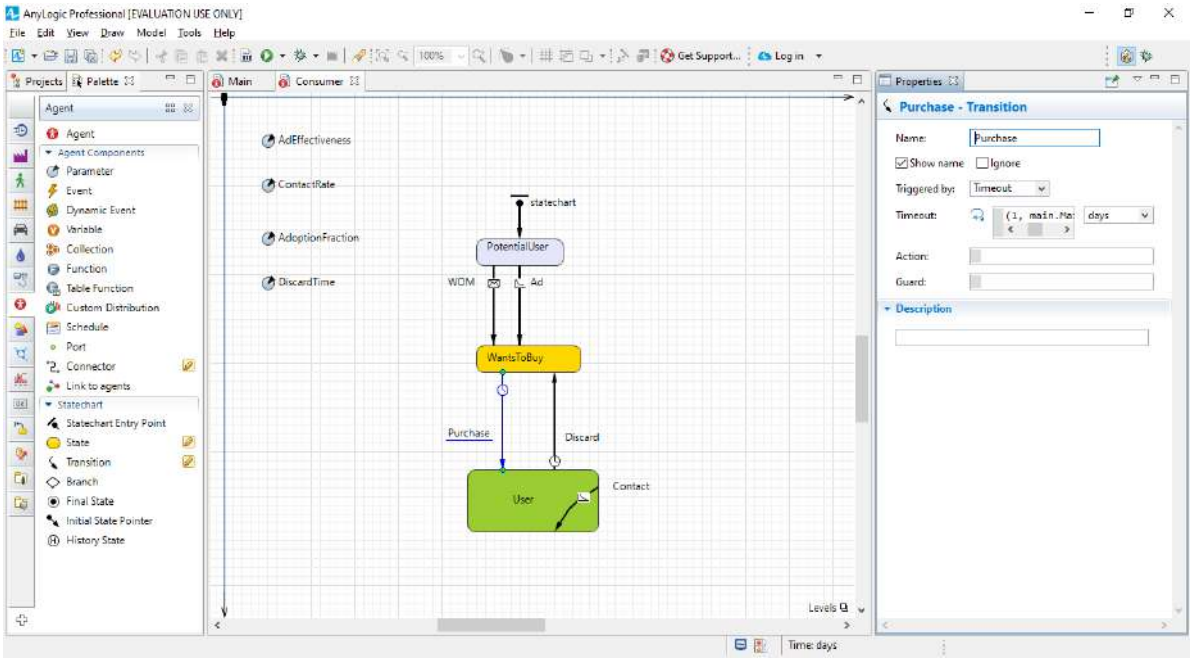
The Choose Probability Description screen allows you to view the list of supported distributions, and you can click any name in the list to view the distribution's description.

Choose triangular in the list. Set min, max and mode parameters equal to 1, 25, 2 respectively. In the upper right, you'll see PDF instantly built for the distribution with the specified parameters. Click OK when finished.

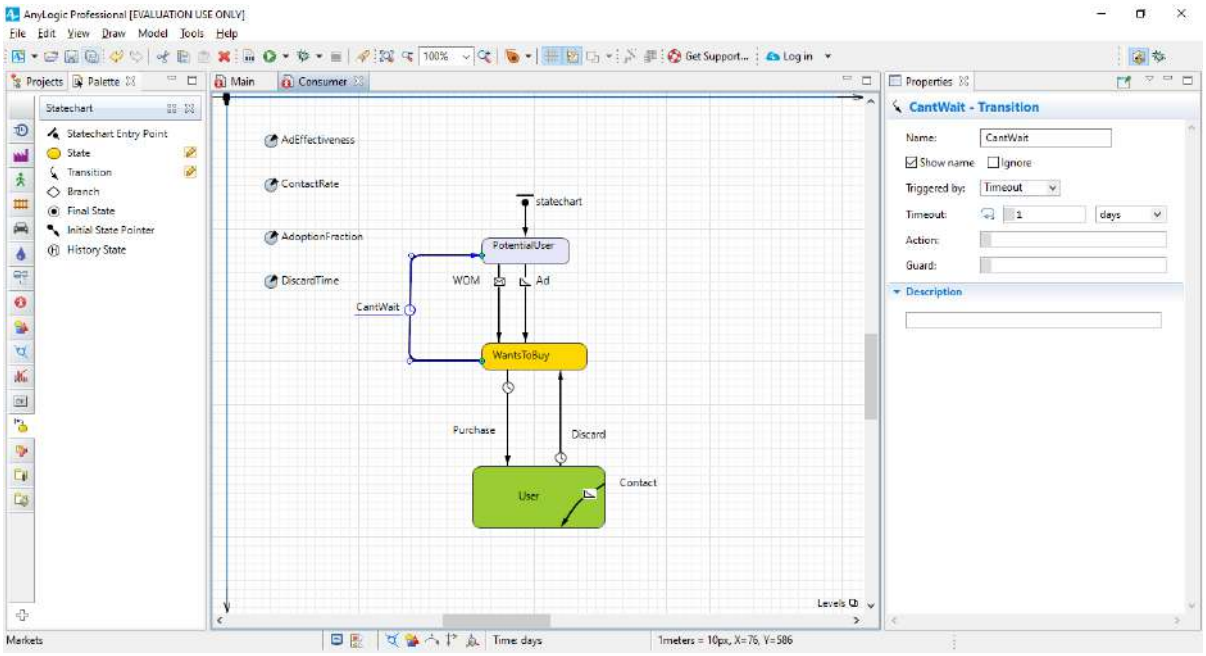


You'll see the expression triangular(1, 25, 2) automatically inserted as the timeout value. Let's modify the line to triangular(1, main.MaxDeliveryTime, 2)

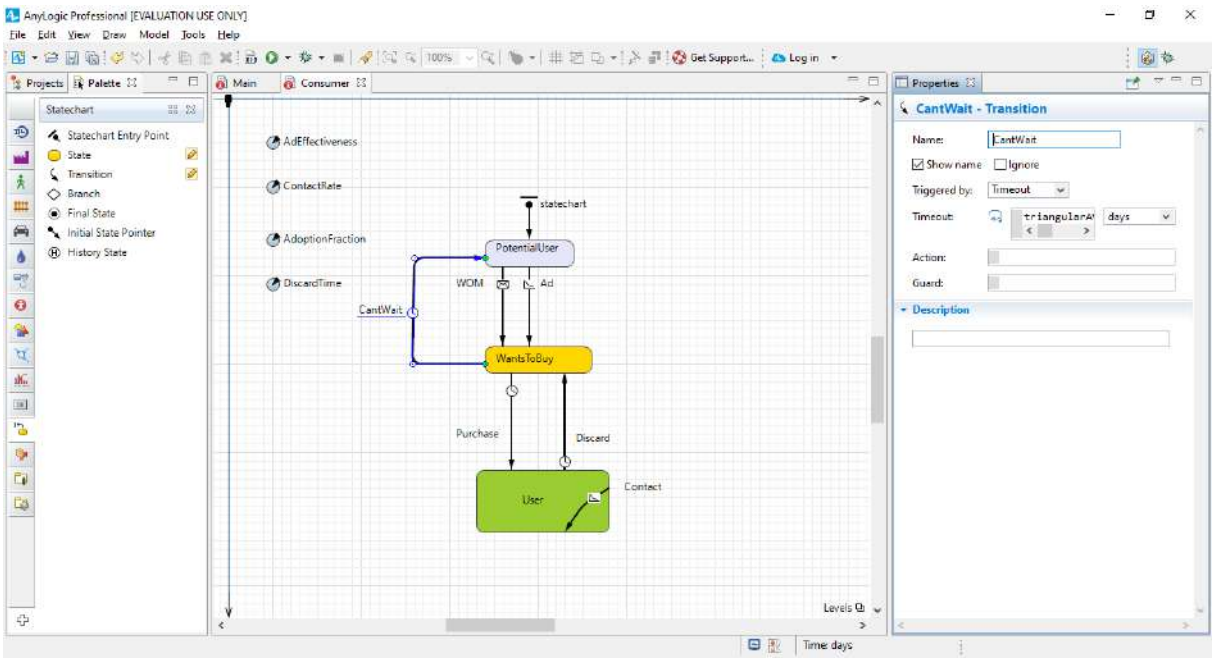
Here main is how we access the Main agent from the consumer agent.



Draw the last transition CantWait that goes from WantsToBuy to PotentialUser state. This transition will model how a consumer's impatience causes them to change their purchase decision, and the Consumer diagram will look like this:

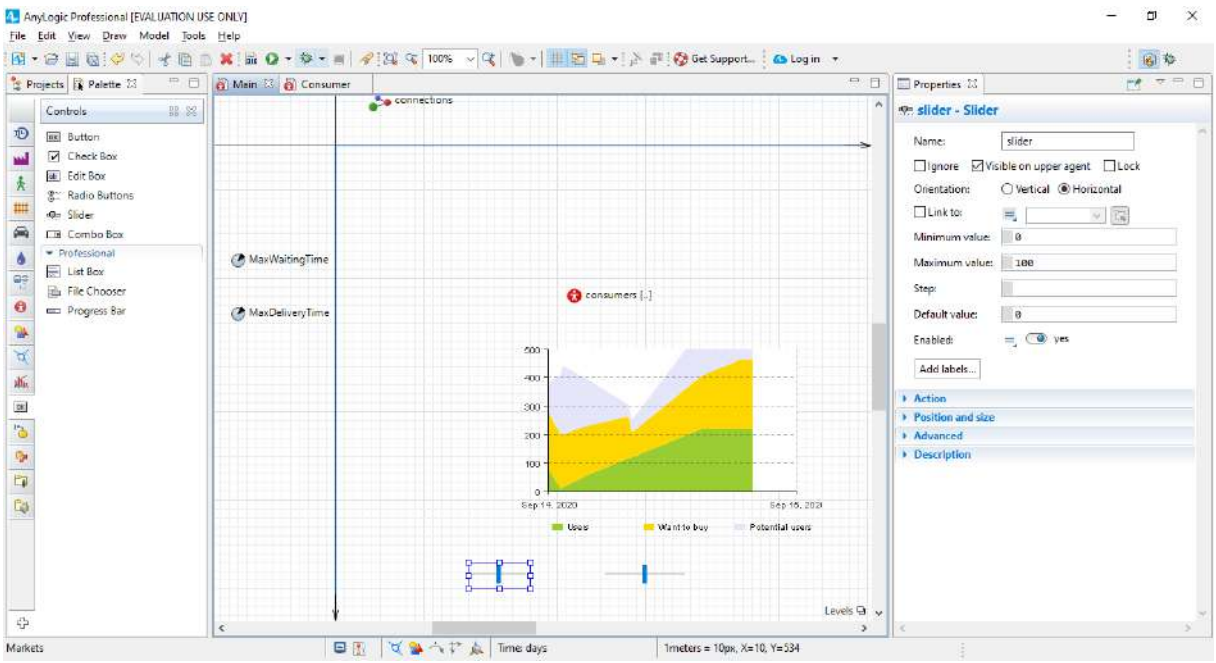


Modify the transition properties so it is triggered by Timeout which equals triangularAV(main.MaxWaitingTime, 0.15) days



Rather than setting the maximum waiting time equal to constant MaxWaitingTime, we assume it follows a triangular distribution with an average of one week and a possible variation to up to 15 percent.

Go back to Main diagram. Open the Controls palette and drag two Sliders on to the diagram below the chart. We'll eventually link the sliders to our two parameters.

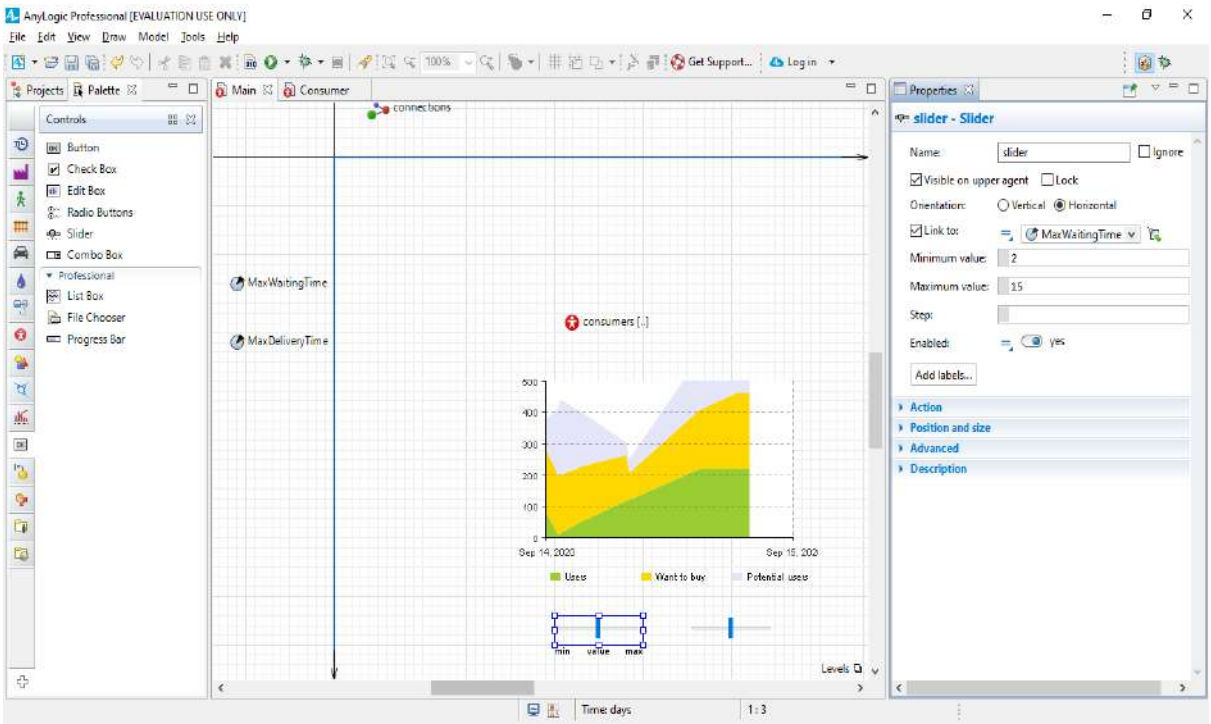


Modify the slider's properties:

Select the checkbox Link to and select the parameter MaxWaitingTime to the right.

Set the slider's Minimum and Maximum values. The parameter value can vary within the range you define here, and we'll set 2 as Minimum and 15 as Maximum value.

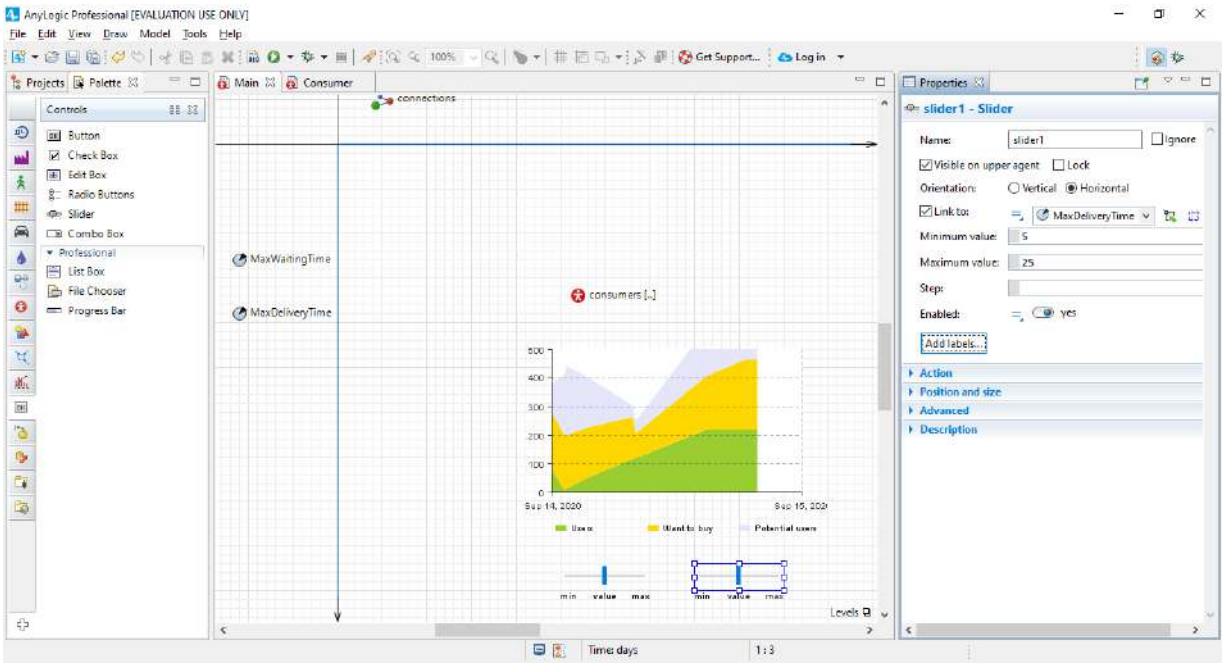
Finally, click the Add labels... button to display the slider's minimum, maximum, and current values at runtime (the min, value, and max text shapes will appear beneath the slider).



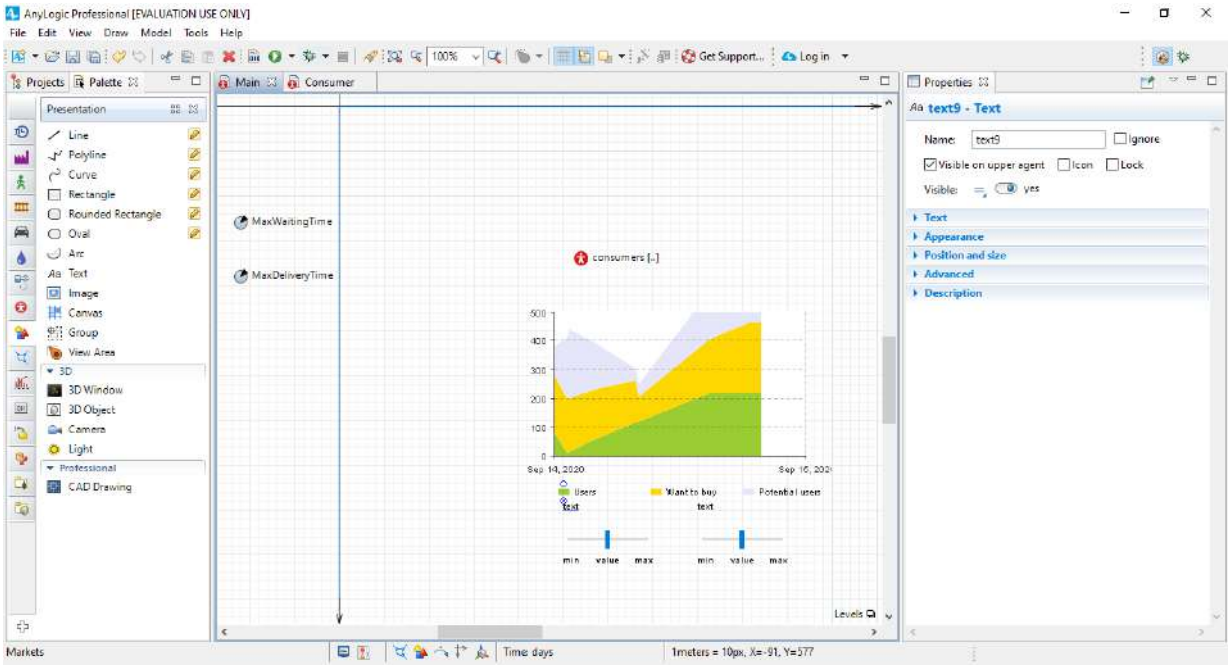
On other slider Select the checkbox Link to and select the parameter MaxDeliveryTime to the right.

Set Minimum value as 5 and Maximum value as 25.

Then click on Add label.

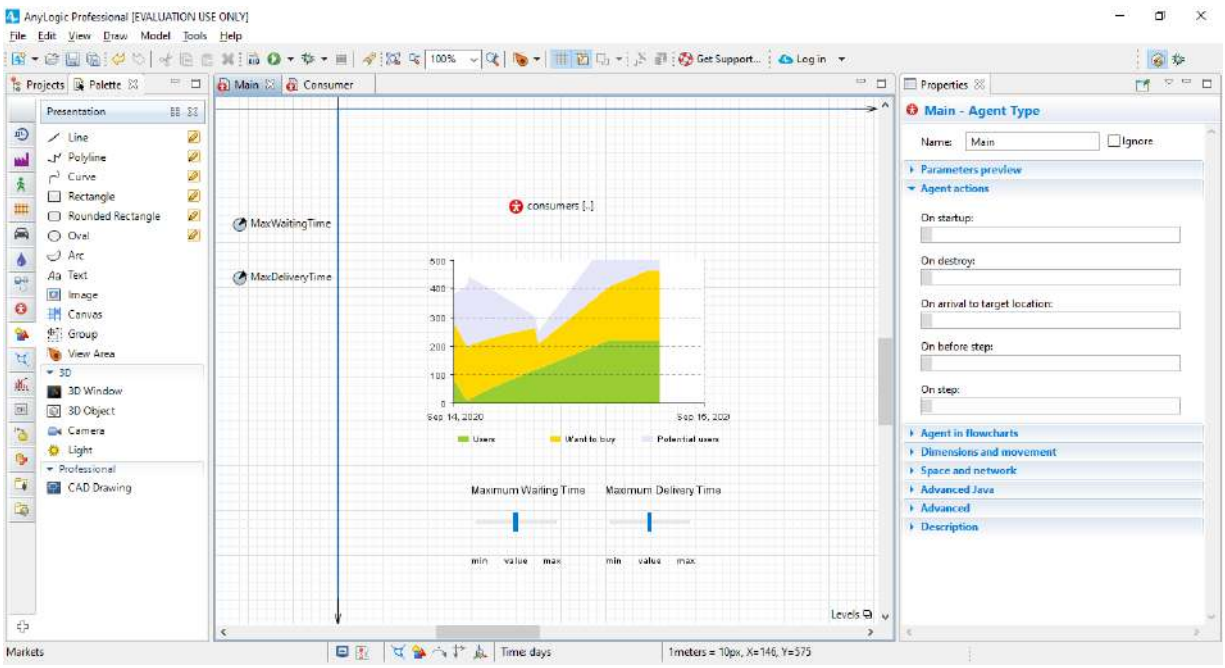


Open the Presentation palette, drag two Text shapes on to the diagram, and place them above the sliders. Let's configure the titles of these controls.

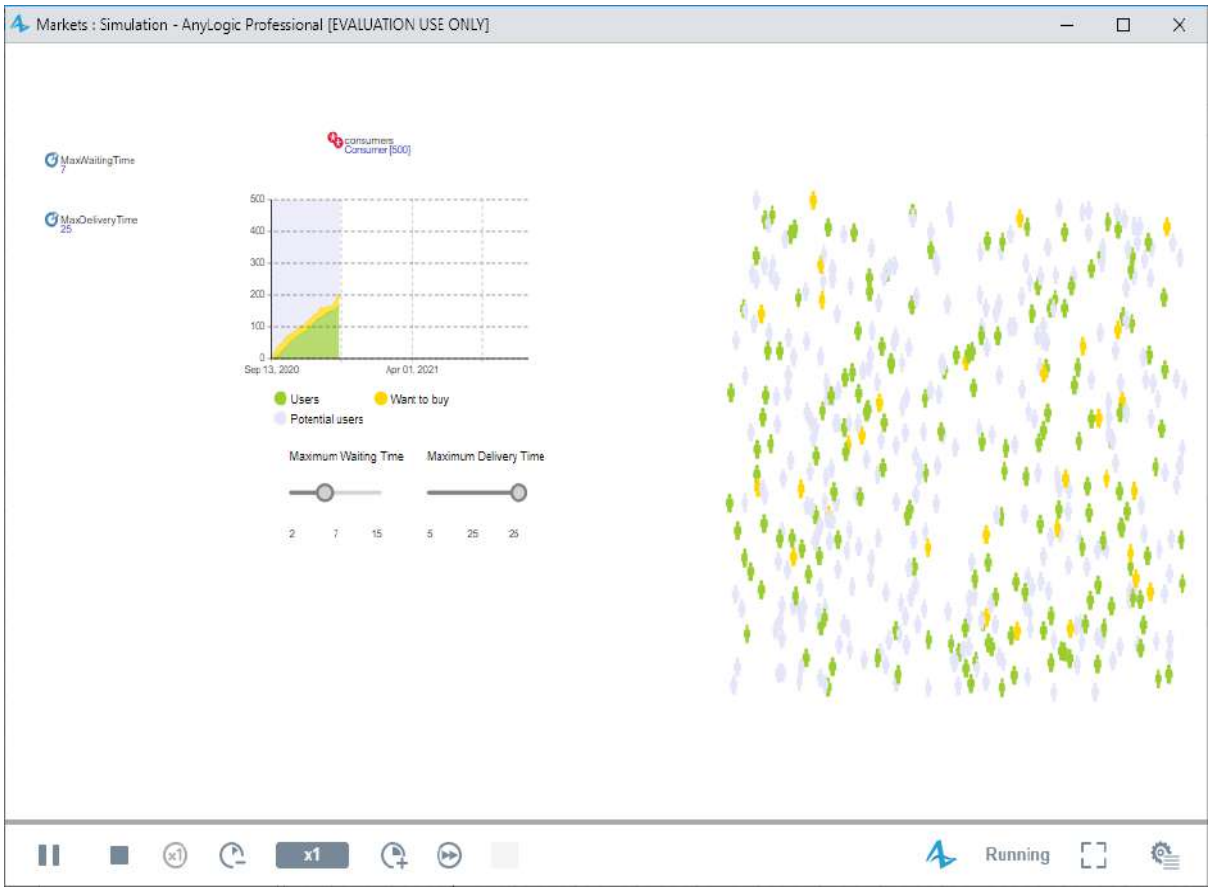


In the properties view, in the Text section, enter the text that the model will display. Using text shapes, name one slider Maximum waiting time and the other slider Maximum delivery time.

In the properties section, under Appearance, you can customize the text's color, alignment, font, and point size.

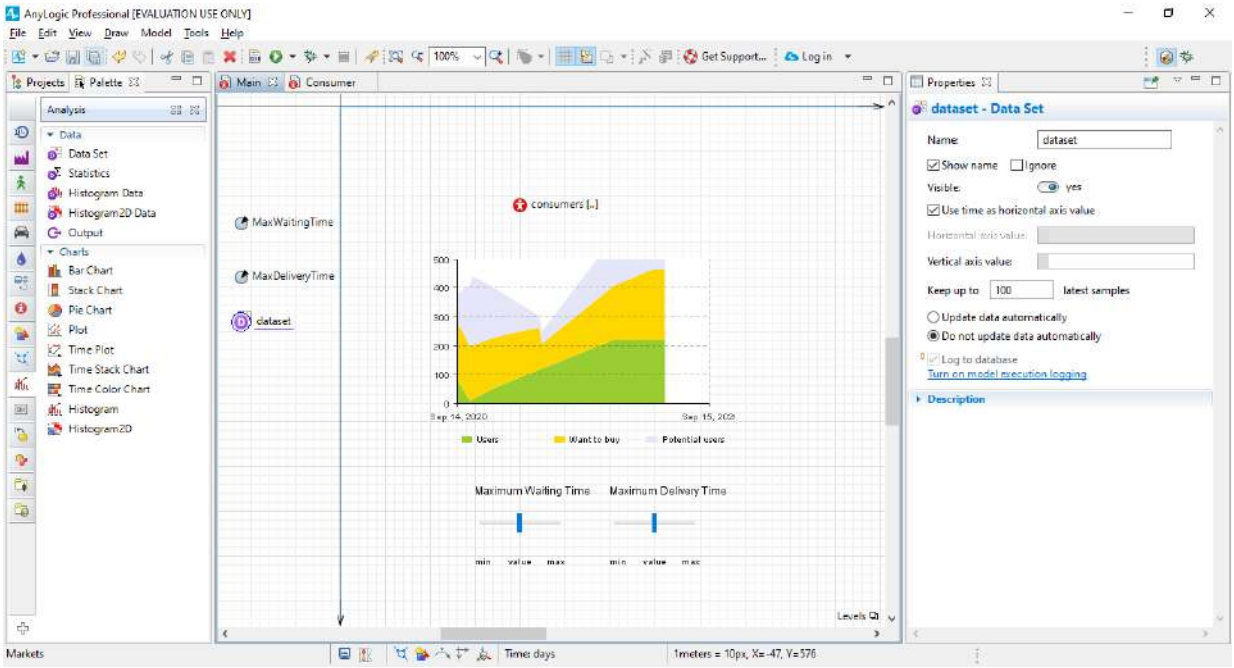


Run the model and observe the behavior. As you use the sliders to change the maximum waiting time or delivery time, you'll see your changes reflected in consumer behaviors and whole adoption dynamics.



Comparing model runs with different parameter values

Open the Main diagram and add a Data Set from the Analysis palette. Name it userDS.

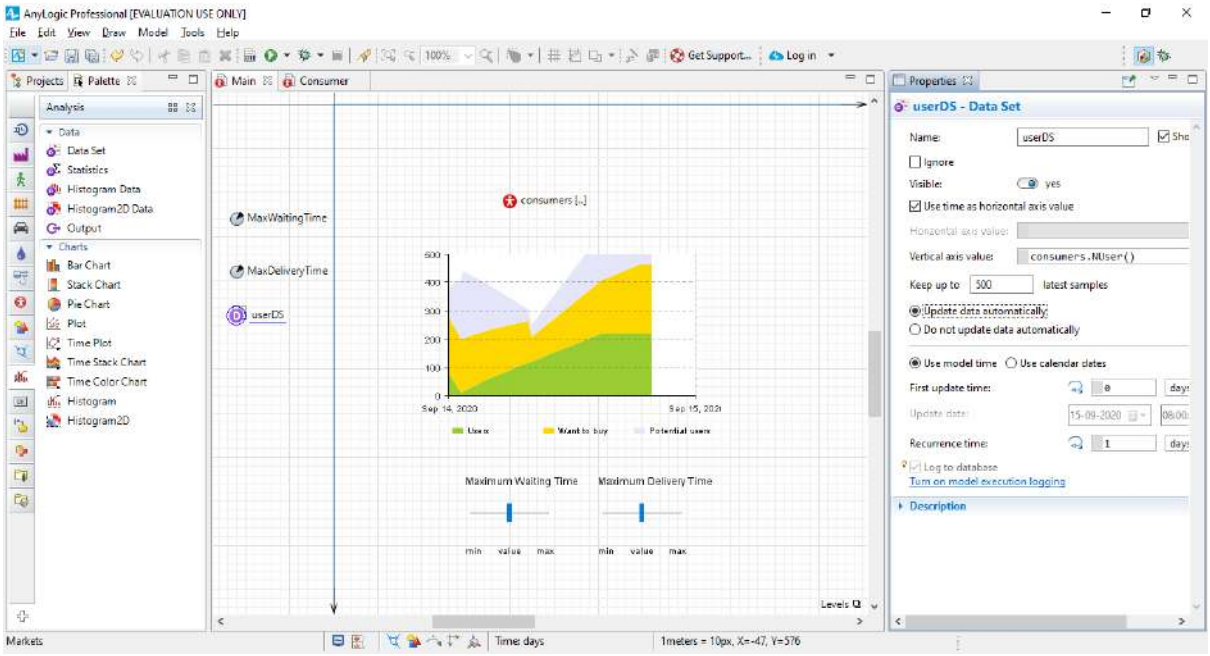


Data Set is capable of storing 2D (X,Y) data of type double. We want this data set to store the history of product sales dynamics. We'll store data samples, each with a timestamp and the current number of the product users.

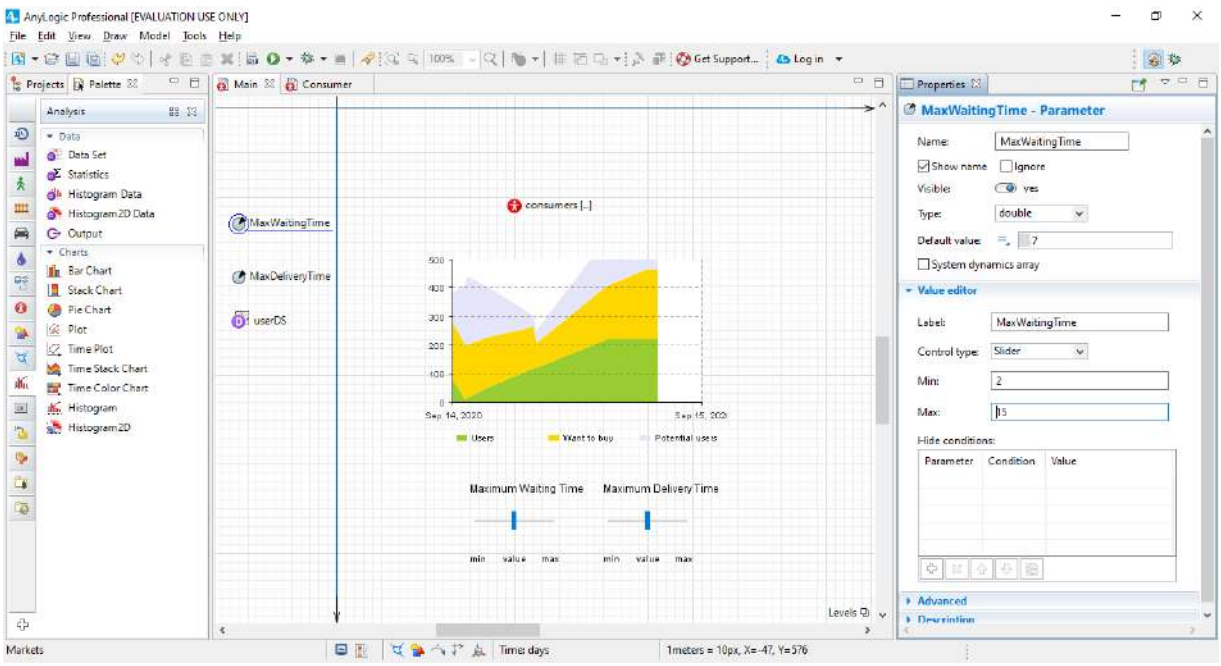
To store the timestamps, leave the dataset's option Use time as horizontal axis value selected.

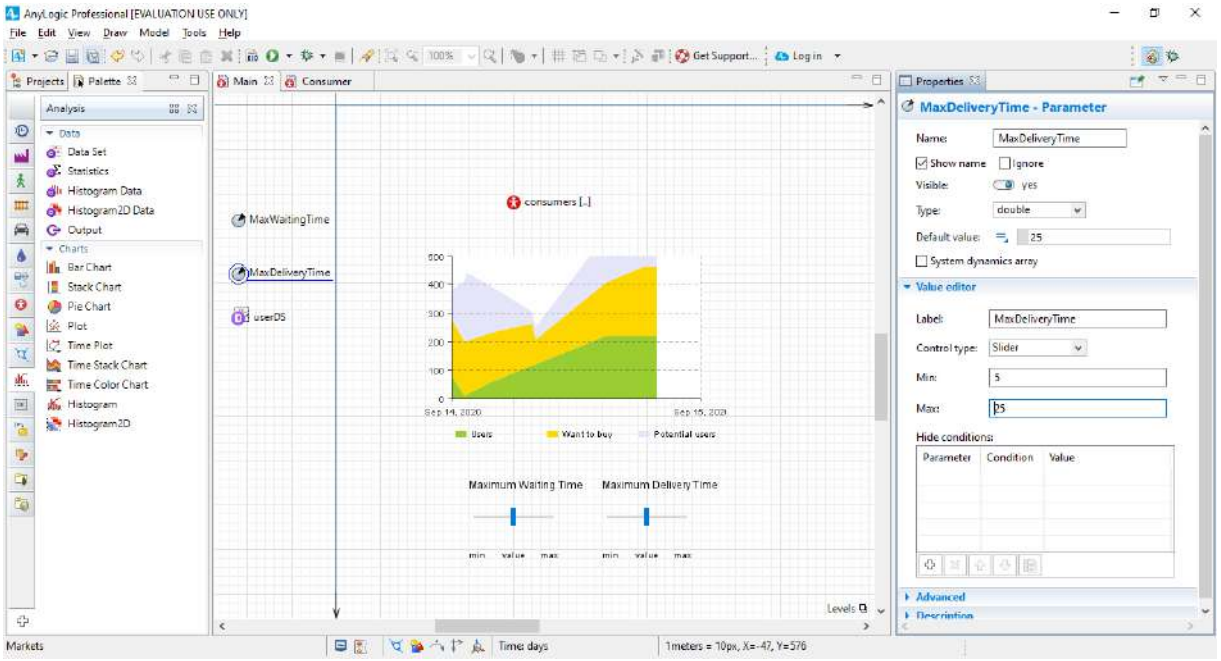
Set the value that the dataset will store. In the Vertical axis value property, type: consumers.NUser().

The dataset keeps a limited number of recent latest data items, and we'll limit our sample size to 500. Set the dataset to Keep up to 500 latest samples. Set it to Update data automatically with the default Recurrence time: 1. We'll add one data sample for one day of the model's lifetime.

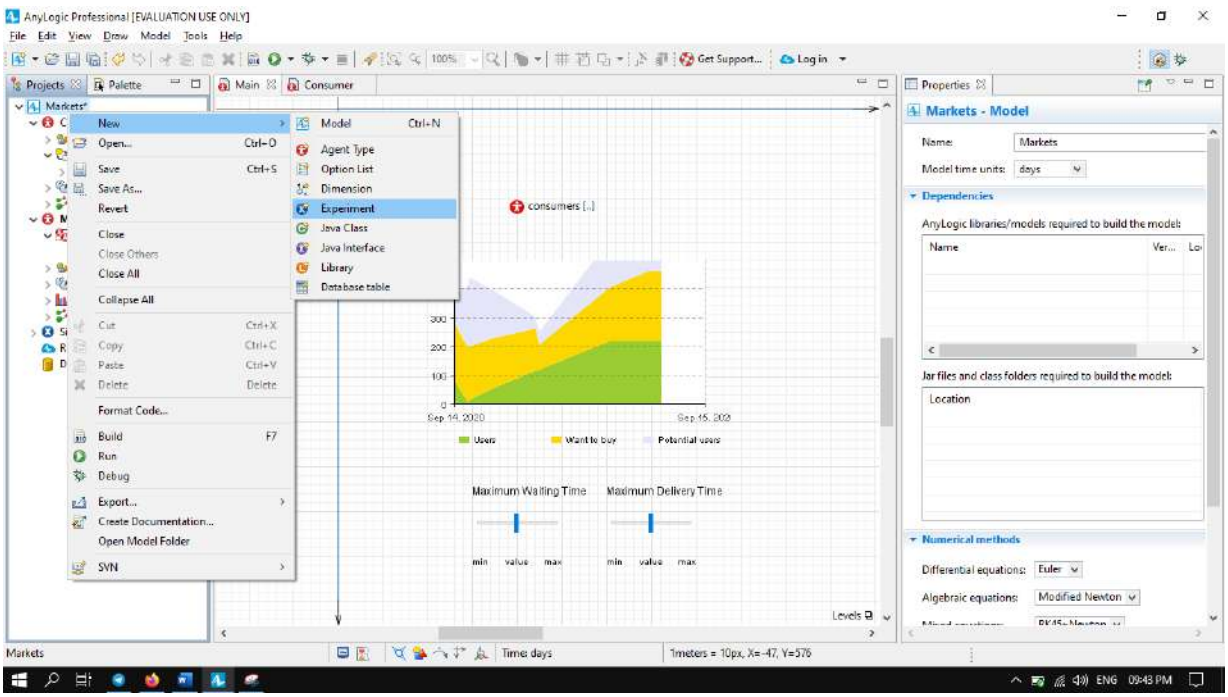


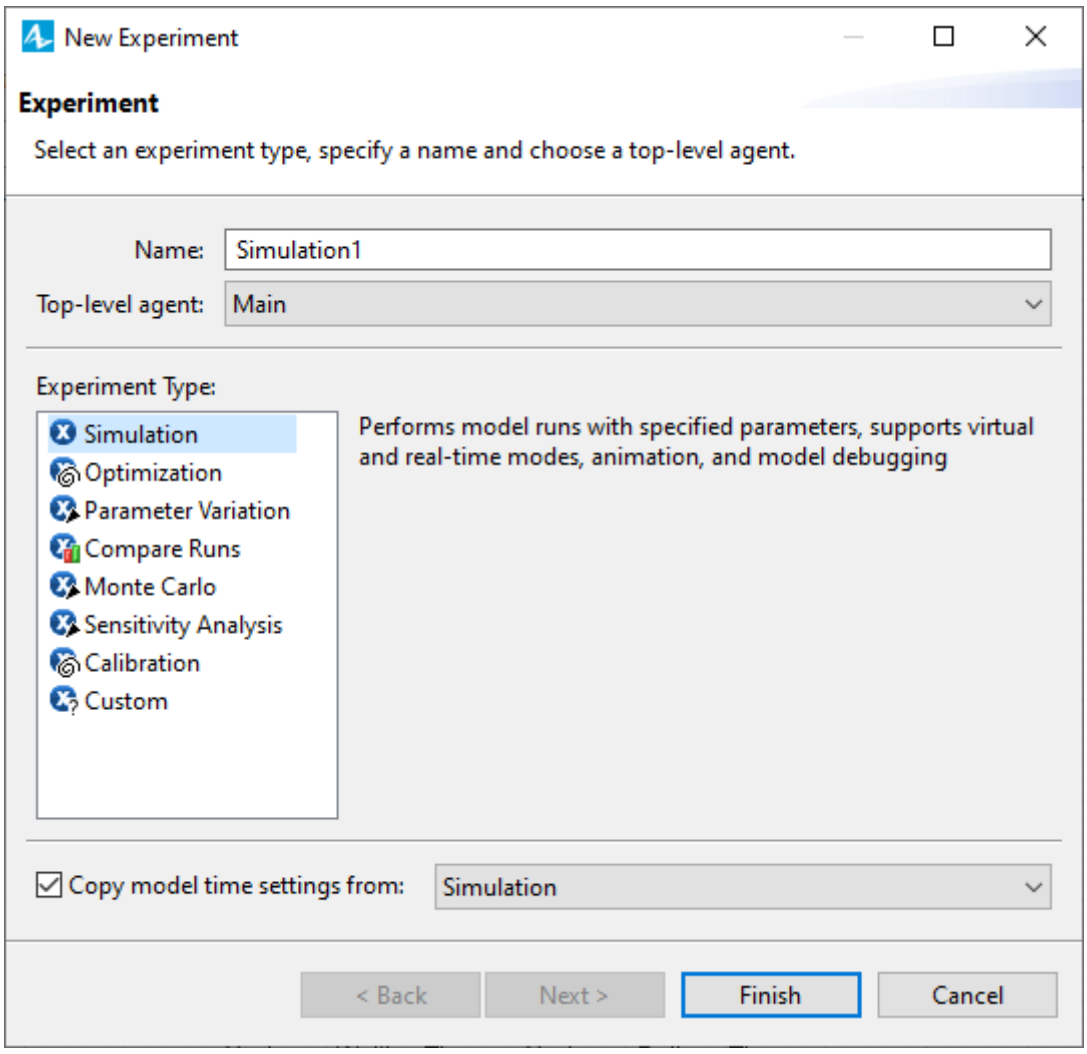
Next, make changes in the Value editor section for both parameters on Main diagram (MaxWaitingTime and MaxDeliveryTime). Choose Slider as Control type, set Min and Max values the same as we have in the sliders on Main, and if you want, change the default label (say, Maximum waiting time).



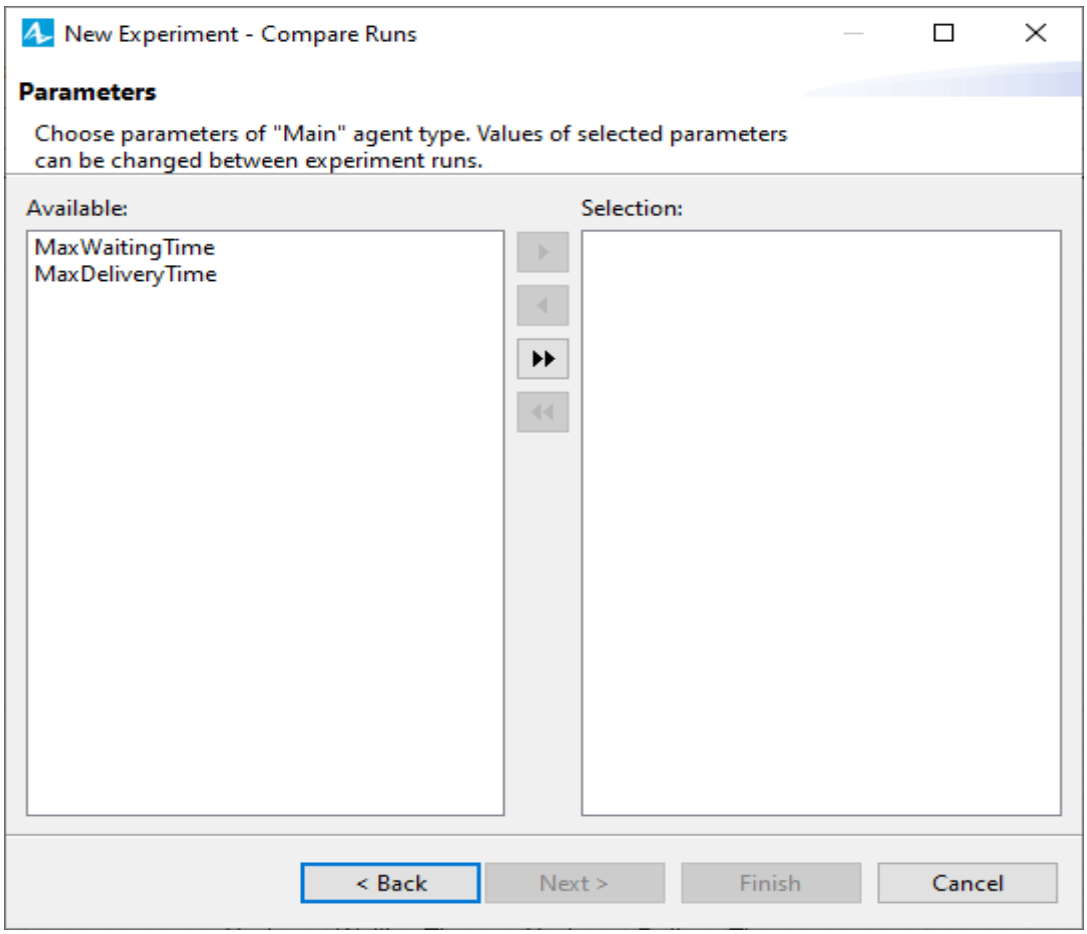


Open the Projects view, right-click the model item, and select New > Experiment from the context menu. The New experiment wizard will pop up.

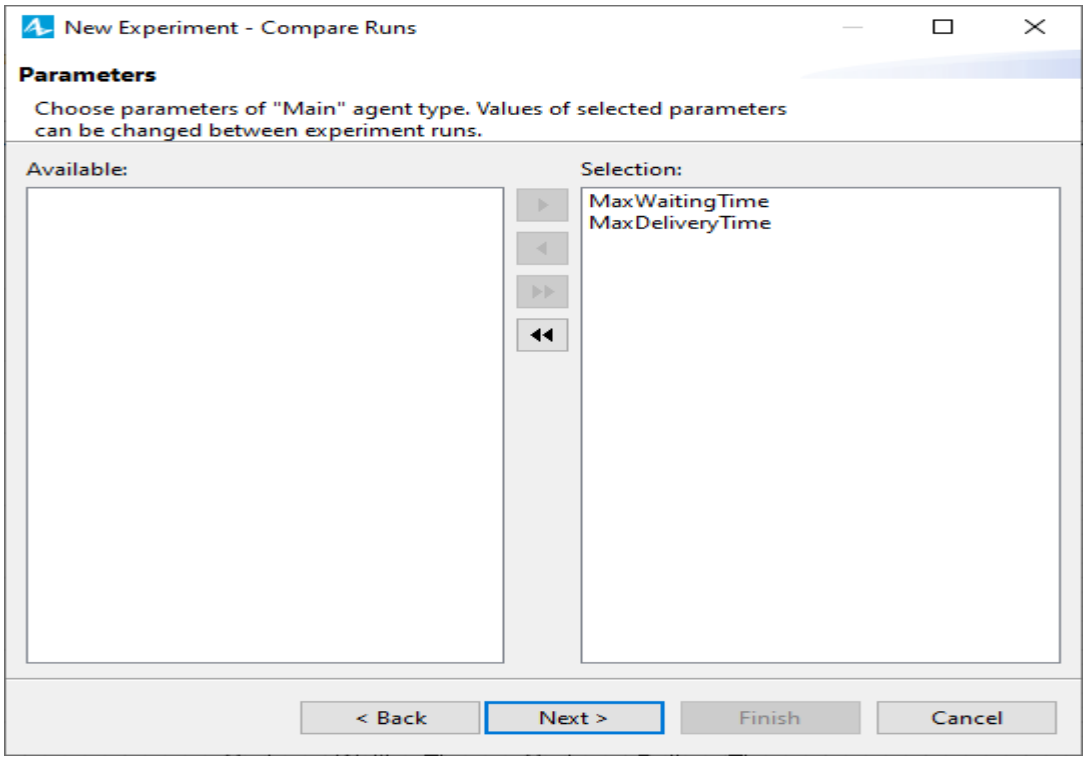




Select Compare Runs experiment from the list of experiment types and click Next.



On the Parameters page, add both parameters to the Selection column. To add a parameter, select it in the Available list on the left and click the arrow. You can also click the button to add all the parameters. Click Next after both parameters are in Selection.

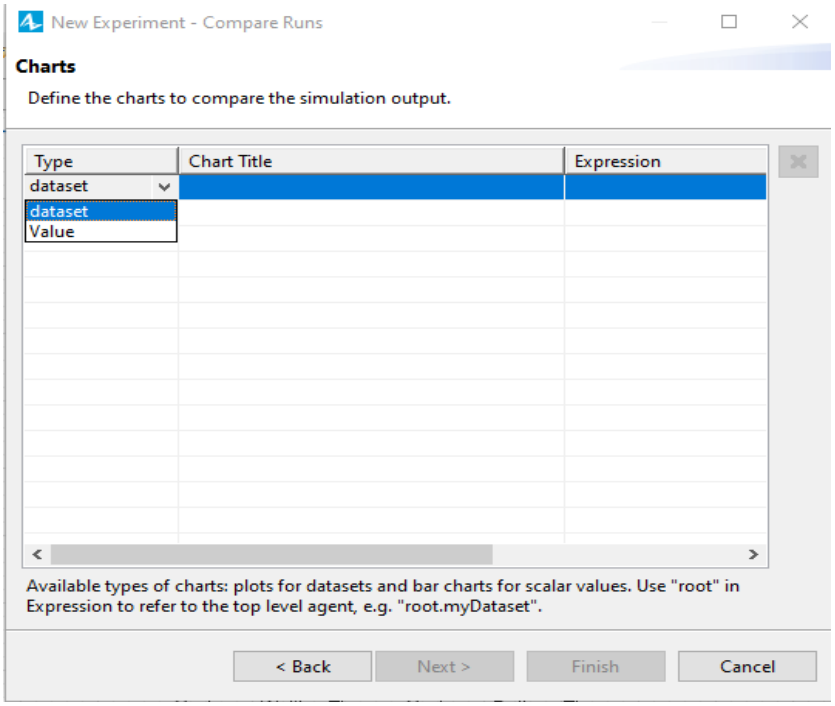


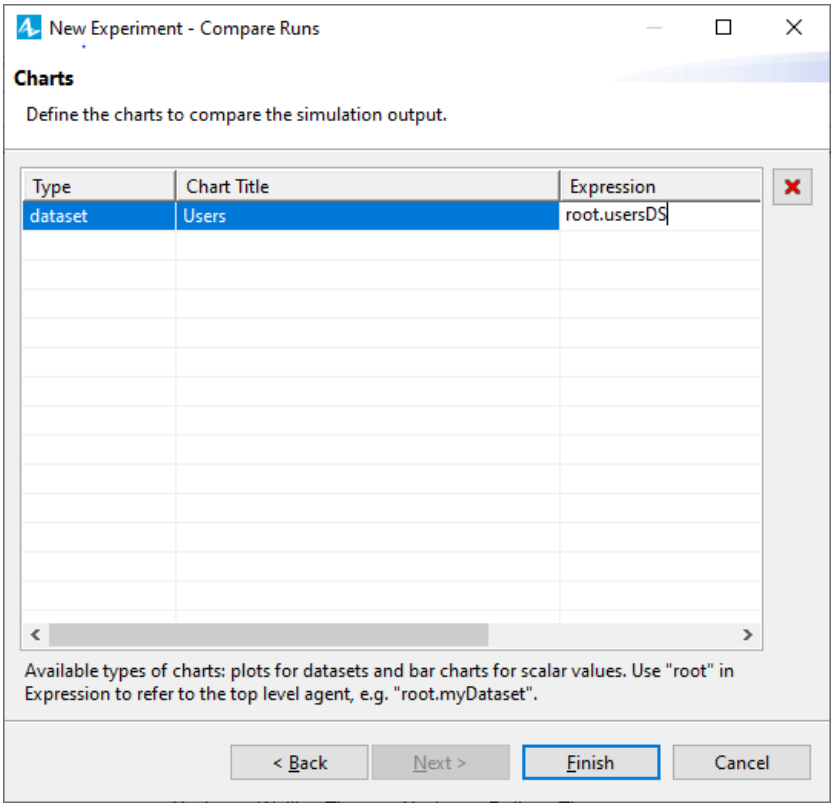
On the wizard's following page, configure the output charts for this experiment. The chart will display the data collected by the dataset usersDS. In the Charts table, do all of the following :

a. In the Type column, select dataset.

b. In the Chart Title column, type Users.

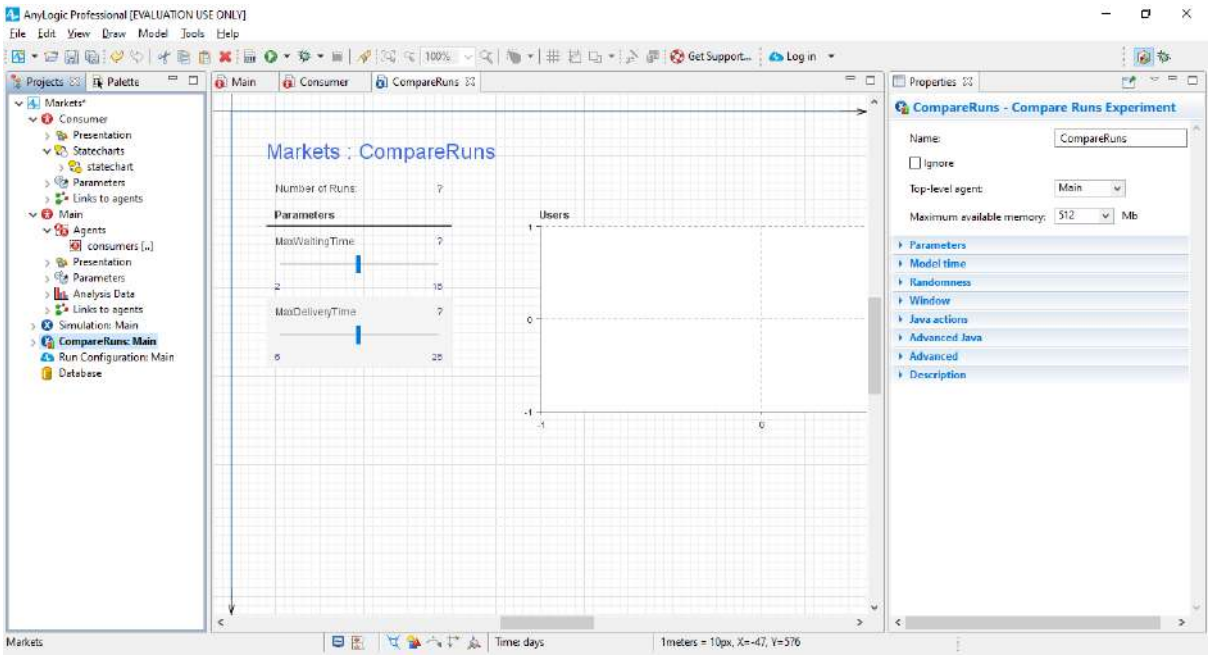
c. In the Expression column, refer to the dataset you defined in Main as root.usersDS where root is the model's top-level agent (Main)



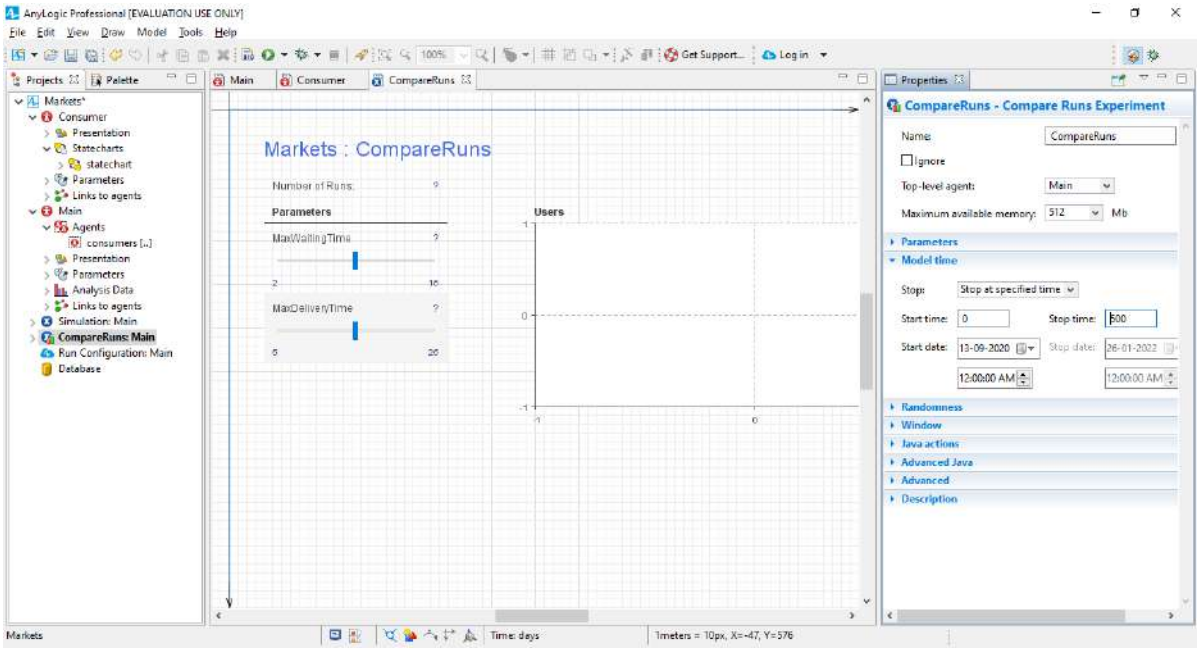


The chart will display the data collected by the dataset usersDS. Click Finish.

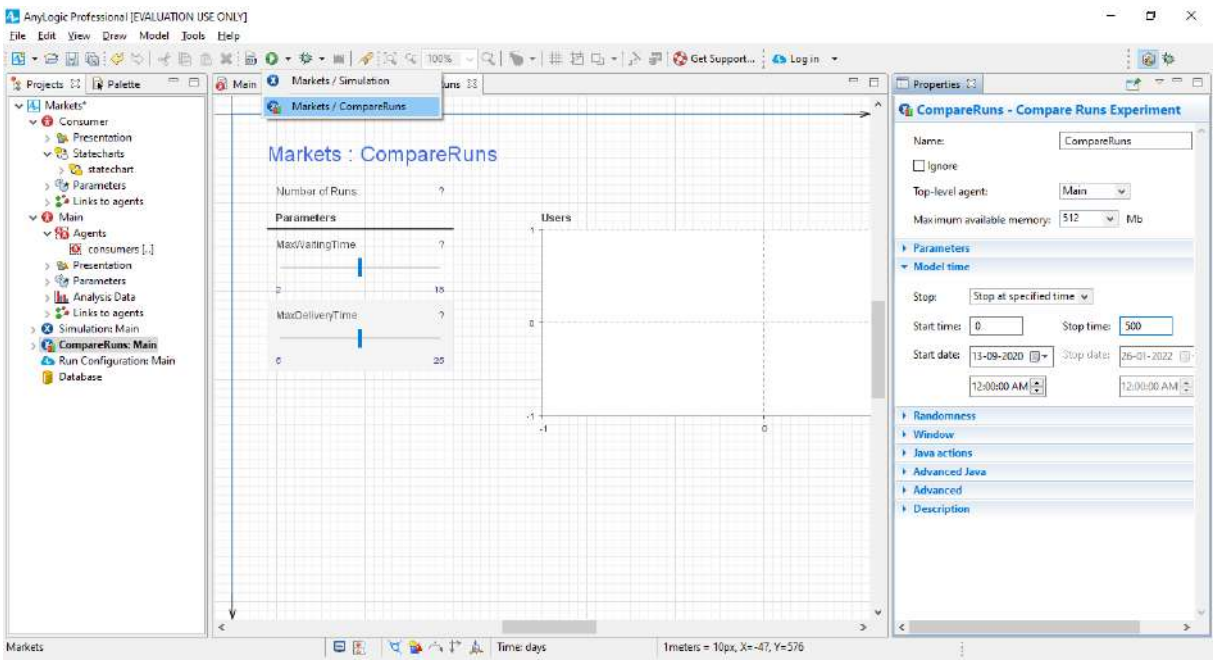
The CompareRuns experiment diagram should open automatically, and you'll see the default user interface we created with the wizard.



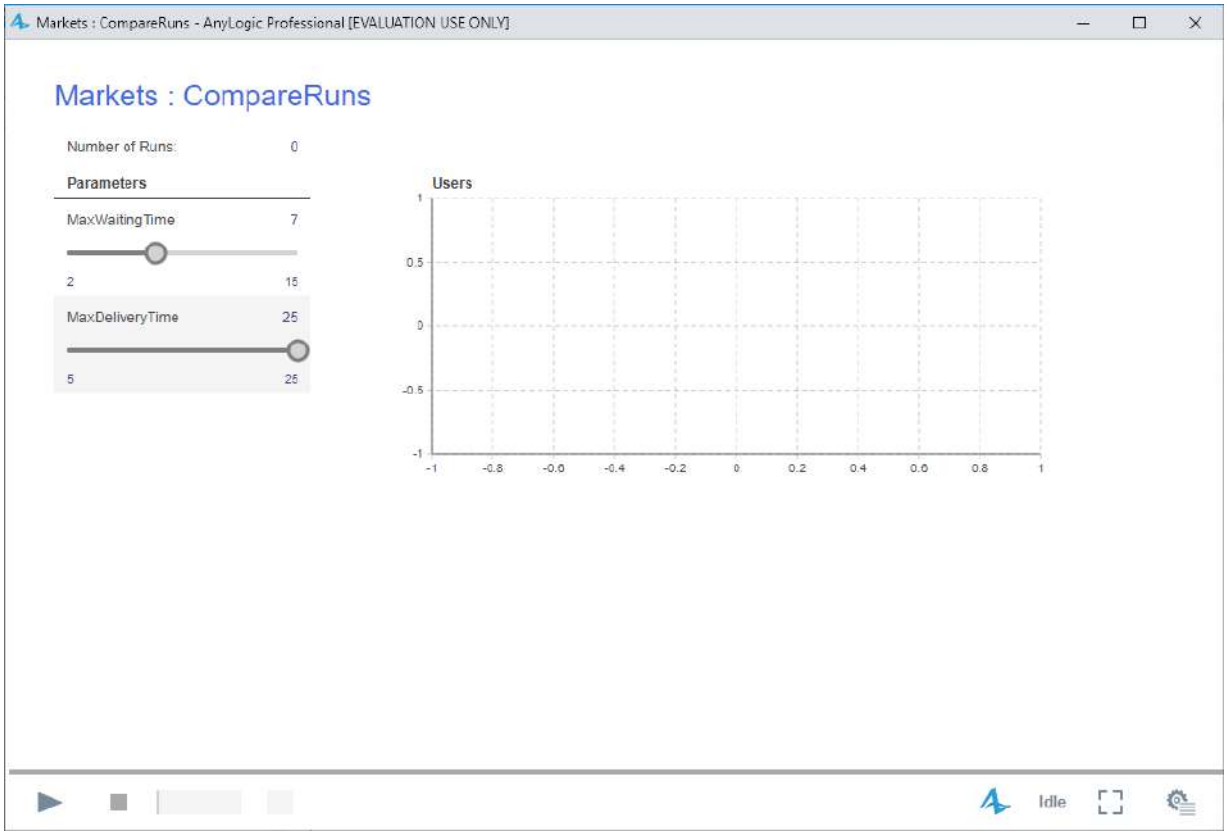
We want our experiment to simulate the model for just 500 days. To do this, select CompareRuns experiment in the Projects tree. In the experiment properties, open the Model time properties section, and type 500 in the Stop time field.

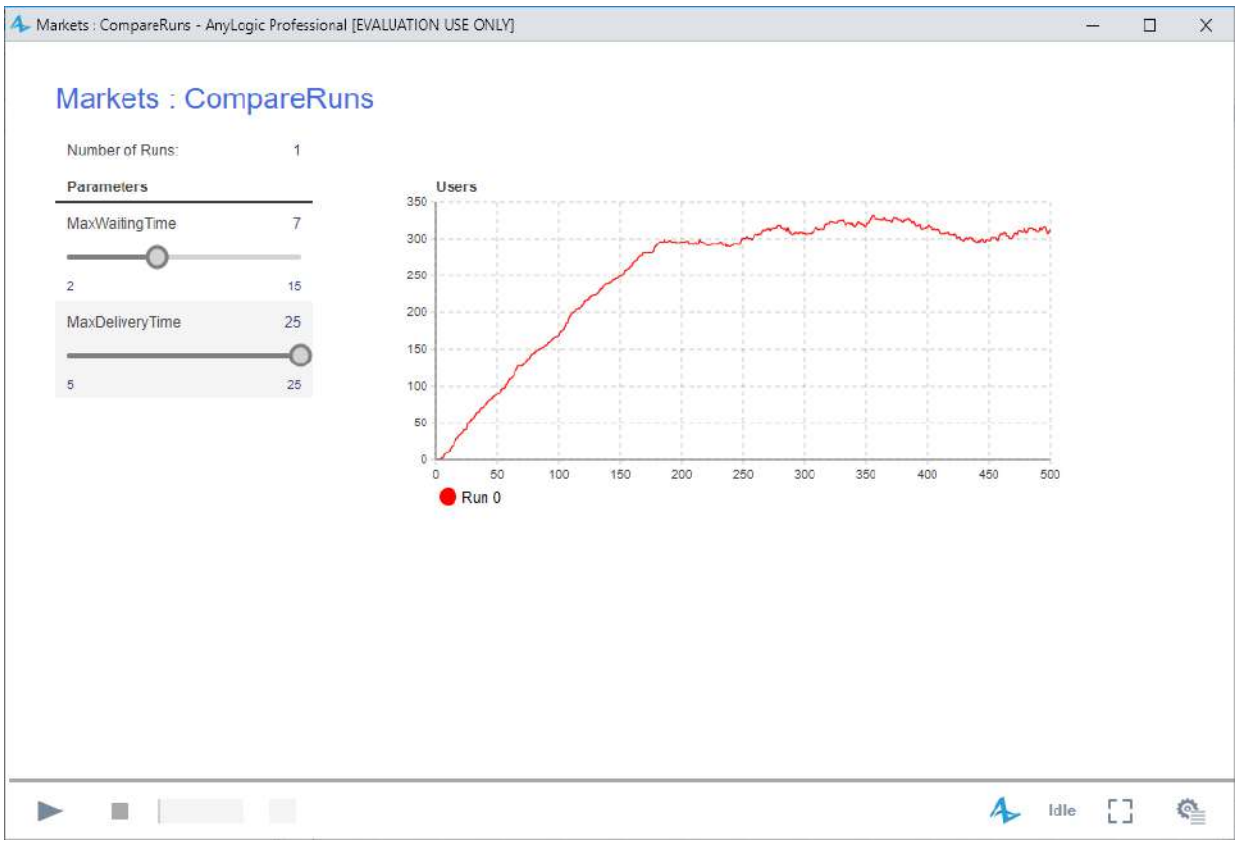


Run the experiment. Select the newly-created experiment from the Run list: Market / CompareRuns, or right-click the CompareRuns experiment in theProjects tree and select Run from the context menu.



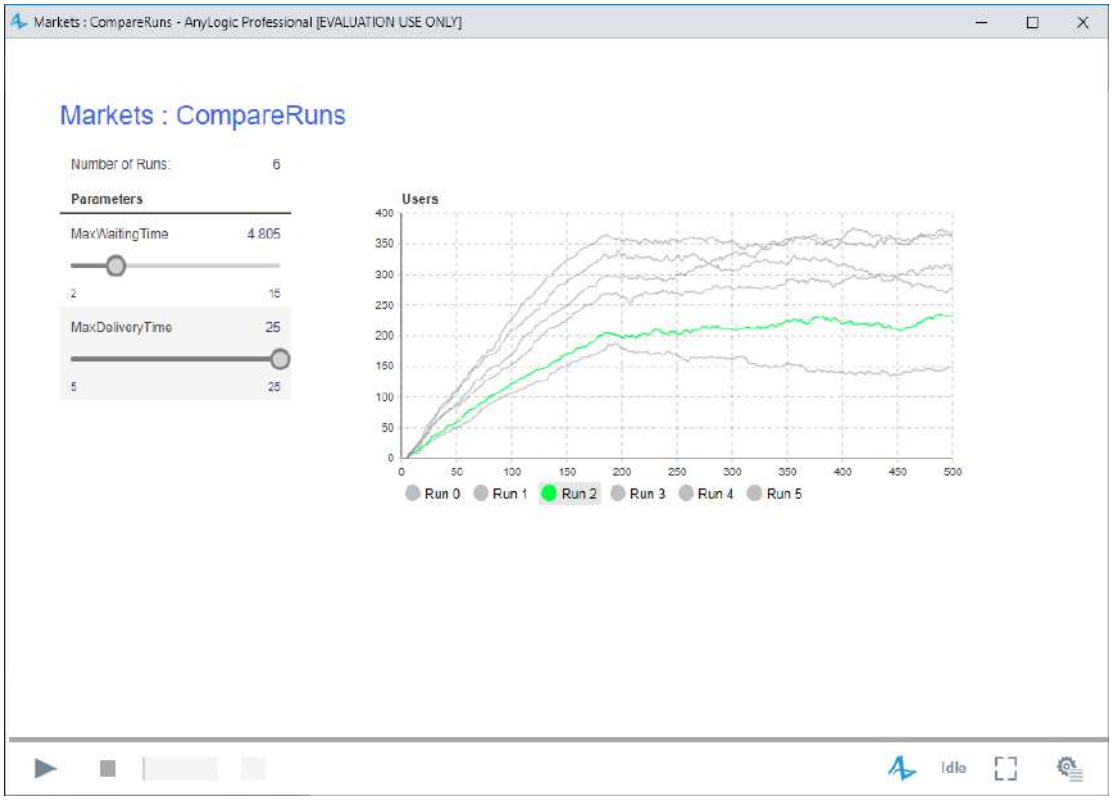
In the model window, click the Run button to see the result associated with the default parameter values. Afterward, change the parameter values and click Run again to observe the system behavior for the new settings. The chart displays all the results for your review.

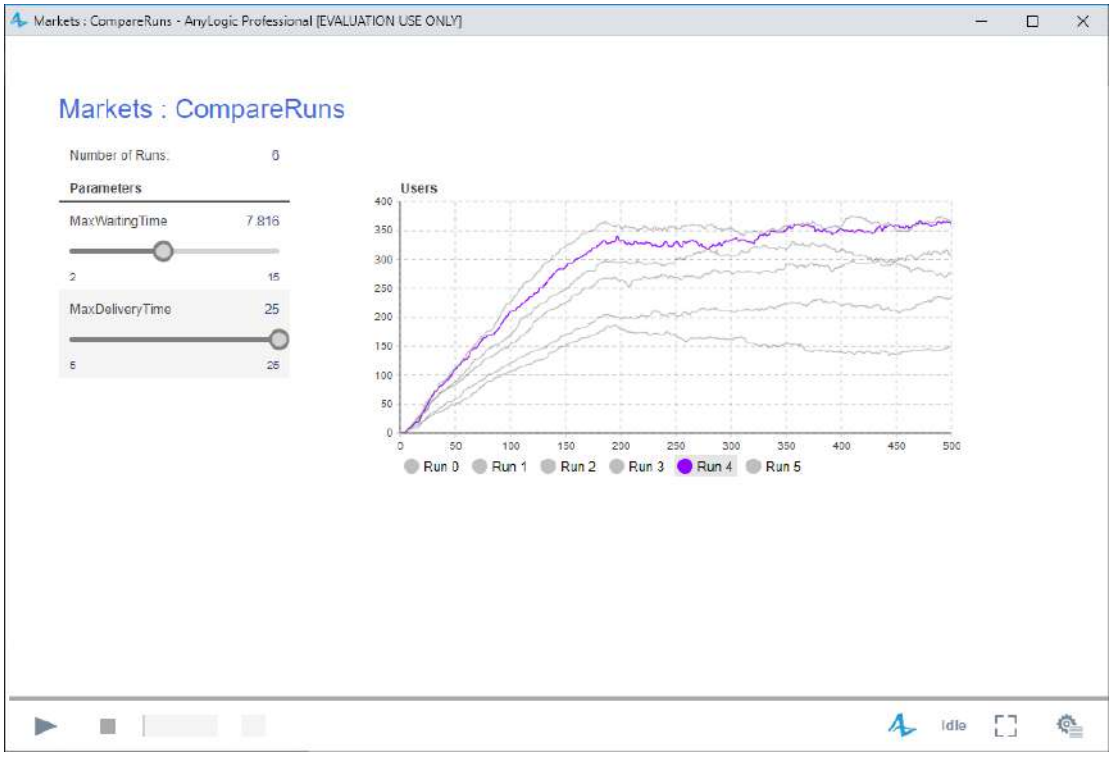



Markets : CompareRuns - AnyLogic Professional [EVALUATION USE ONLY]
Markets : CompareRuns
Number of Runs: 1
Parameters
MaxWaitingTime 7
2 15
MaxDeliveryTime 25
5 25
Users
350
300
250
200
150
100
50
0
0 50 100 150 200 250 300 350 400 450 500
Run 0
Idle


Markets : CompareRuns - AnyLogic Professional [EVALUATION USE ONLY]
Markets : CompareRuns
Number of Runs: 6
Parameters
MaxWaitingTime 8.842
2 15
MaxDeliveryTime 25
5 25
Users
400
350
300
250
200
150
100
50
0
0 50 100 150 200 250 300 350 400 450 500
Run 0 Run 1 Run 2 Run 3 Run 4 Run 5
Idle

Each curve in a chart corresponds to a specific simulation run, and you can click any item in the chart's legend to highlight the curves that correspond to the run. The controls on the left will show the values that led to this result. To deselect a curve, click on its legend a second time.





You can copy the datasets by clicking in the legend and selecting Copy all or Copy selected from the context menu.

PRACTICAL NO: 04

Aim: Design and develop System Dynamic model by

- Creating a stock and flow diagram
- Adding a plot to visualize dynamics
- Parameter variation
- Calibration

[Use a case situation like spread of contagious disease for the purpose].

Code:

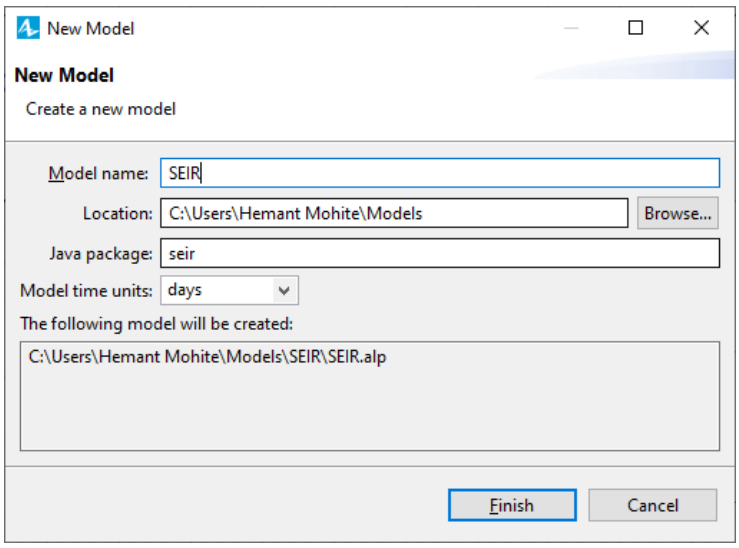
SIER Model

We're about to build a model that displays the spread of a contagious disease among a large population. Our sample model will have a population of 10,000 people – a value we call TotalPopulation – of which one person is infectious.

- During the infectious phase, a person comes into contact with an average of ContactRateInfectious = 1.25 people each day. If an infectious person comes into contact with a susceptible person, the susceptible person's probability of infection is Infectivity = 0.6.
- After a susceptible person is infected, the infection latent phase lasts for AverageIncubationTime= 10 days. We'll use the word exposed to describe people who are in the latent phase.
- After the latent phase, infectious phase starts. This phase lasts for AverageIllnessDuration = 15 days.
- Persons who have recovered from the disease are immune to a second infection.

Phase 1. Creating a stock and flow diagram

Create a new model by selecting File > New > Model from the menu, and then name it SEIR. Select days as Model time units.



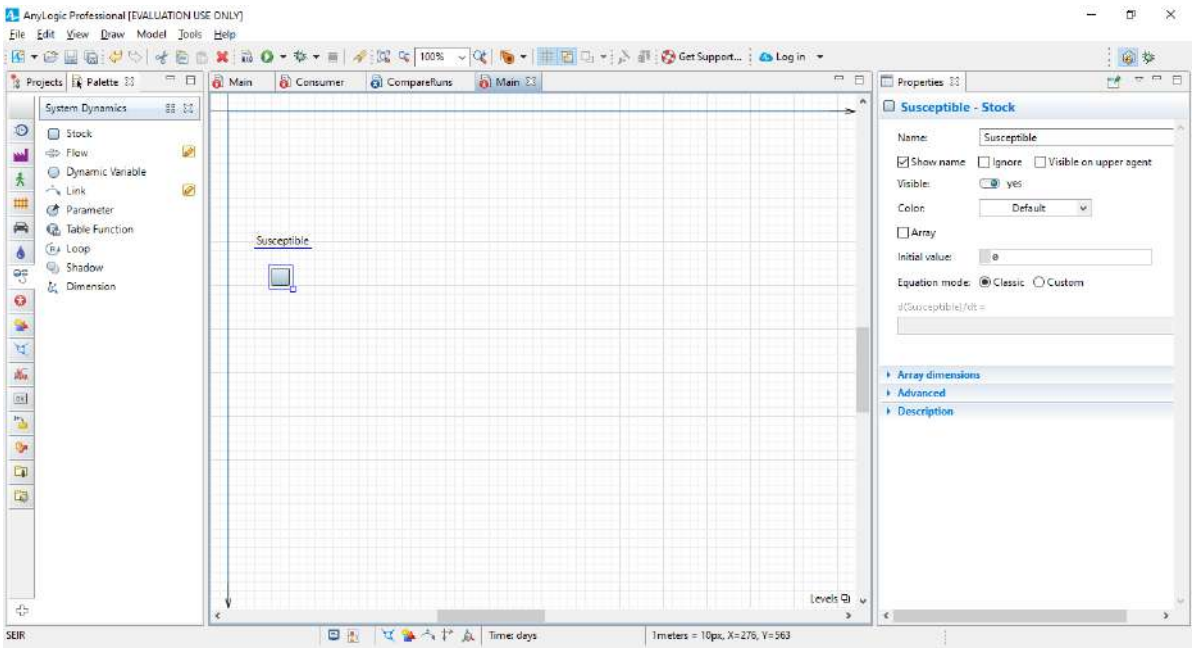
In this example, we'll consider four important characteristics:

- Susceptible - people who are not infected by the virus.
- Exposed - people who are infected but who can't infect others.
- Infectious - people who are infected and who can infect others.
- Recovered – people who have recovered from the virus

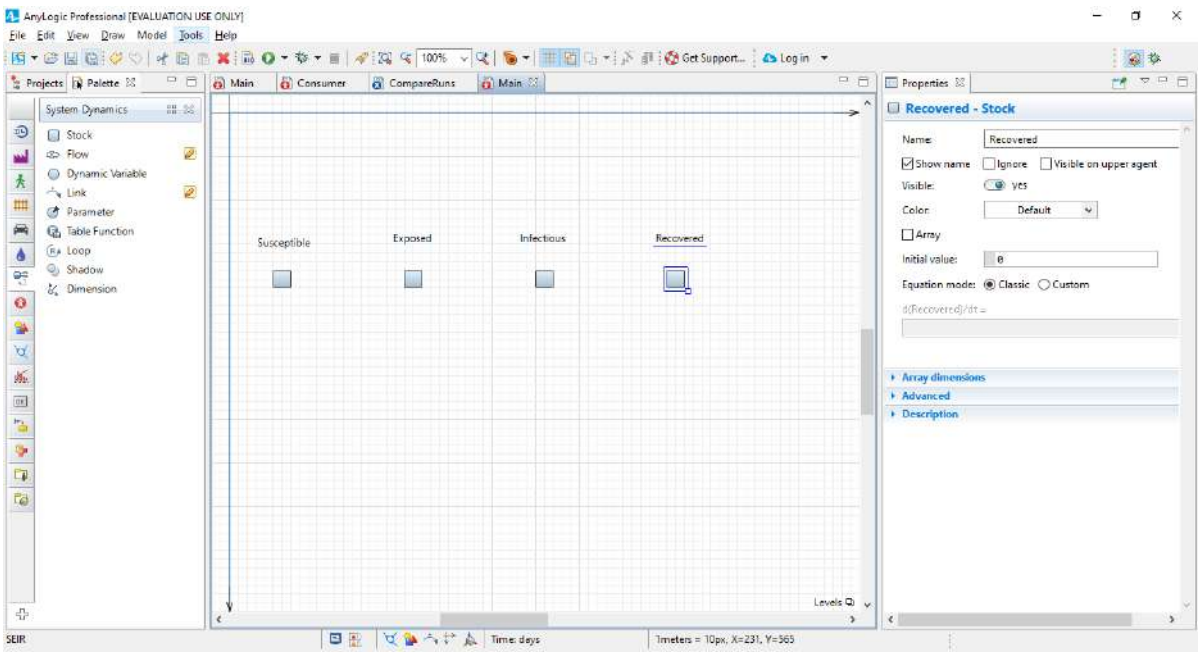
SEIR is an acronym that represents the four stages: Susceptible-Exposed-Infectious-Recovered. The terminology and the overall structure of the problem is taken from the ("Compartmental models in epidemiology". n.d.) --namely, from the SEIR (Susceptible Exposed Infectious Recovered) model.

There are four stocks in our model -one for each stage.

Open the System Dynamics palette. Drag the Stock from the System Dynamics palette on to the diagram. Name it Susceptible.



Add three more stocks. Place them as shown in the figure and name them Exposed, Infectious, and Recovered.

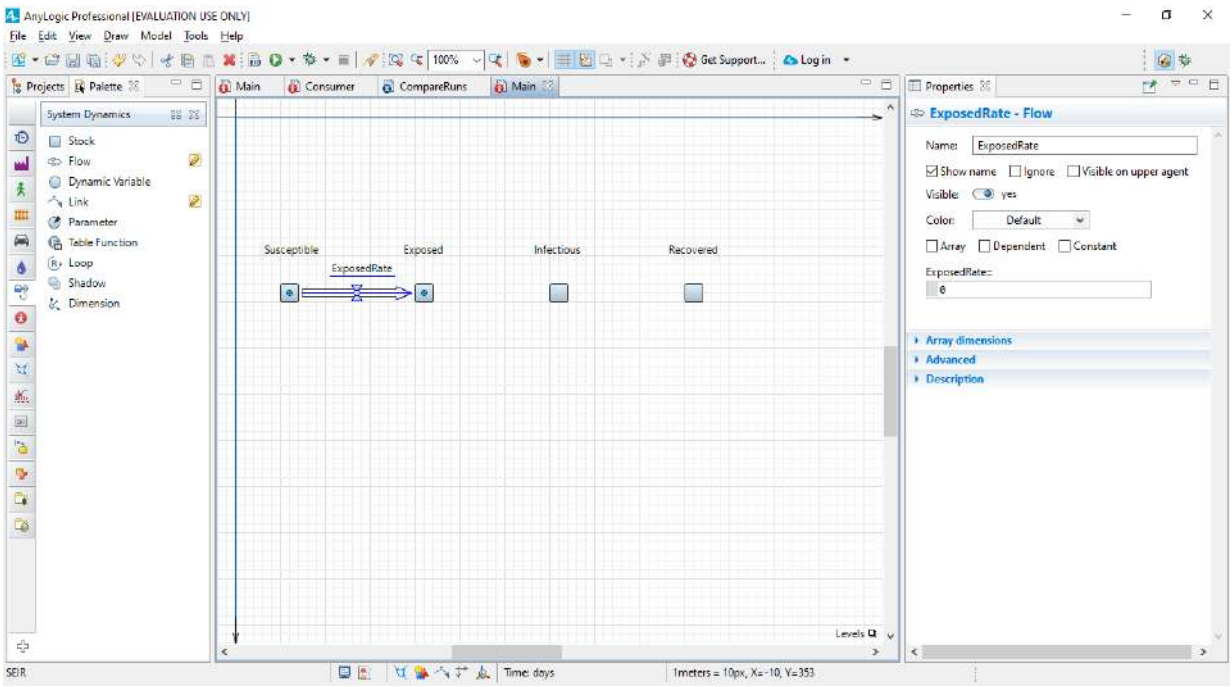


The flow's arrow shows its direction.

In our model, susceptible people are exposed to the virus, become infectious, and then recover. It's a progression that requires our model to use three flows to drive people from one stock to the next.

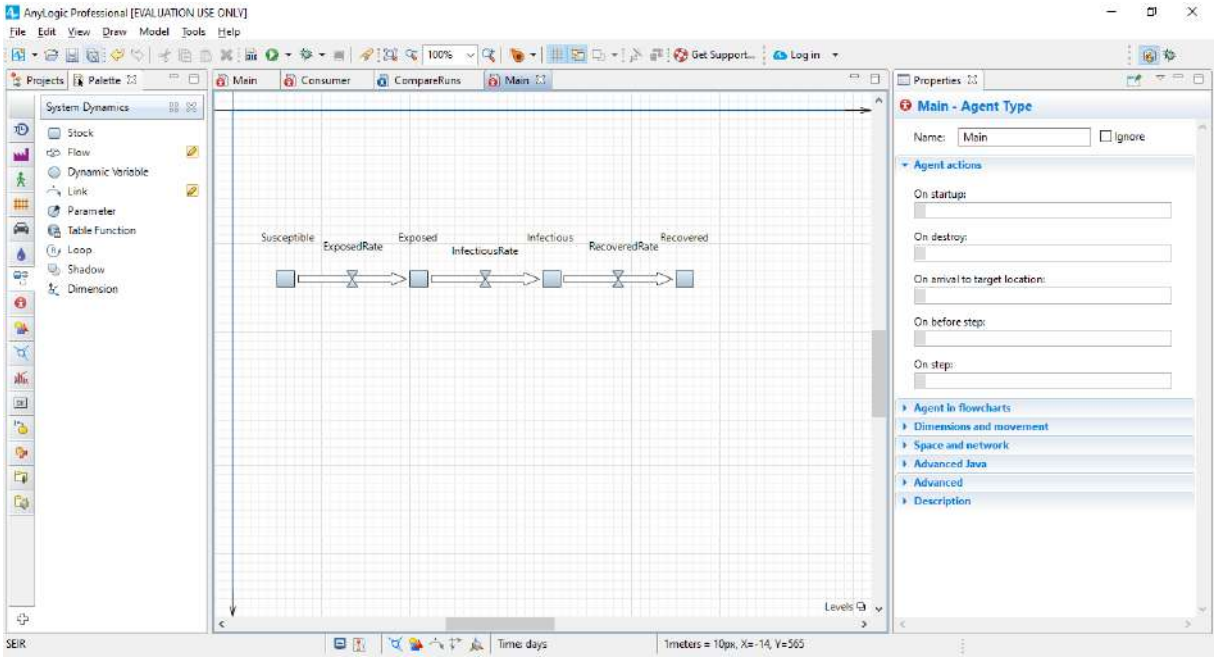
Add the first flow that flows from the stock Susceptible to Exposed. Double-click the stock where the flow flows out (Susceptible), and then click the stock where it flows in (Exposed).

Name the flow ExposedRate.

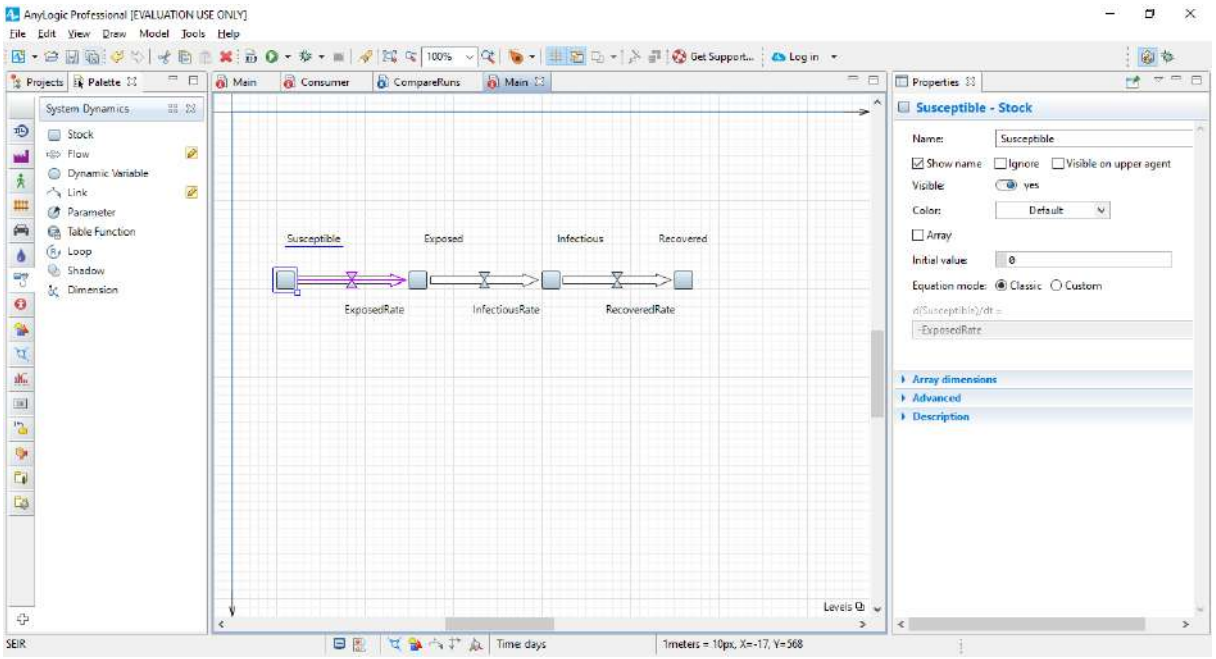


Add a flow from Exposed to Infectious, and then name it InfectiousRate.

Add a flow from Infectious to Recovered, and then name it RecoveredRate.



Rearrange the flow names as shown in the figure below. To do this, select a flow and then drag its name.



Add five Parameters, name them, and then define their default values according to the information below:

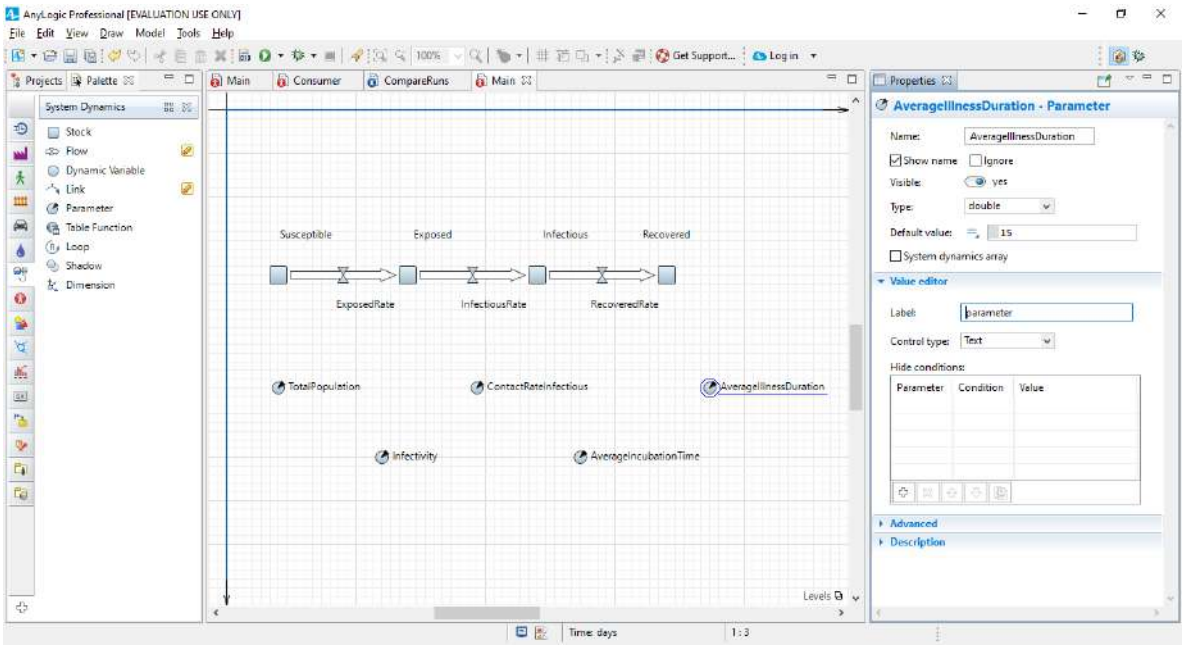
•TotalPopulation = 10000

•Infectivity = 0.6

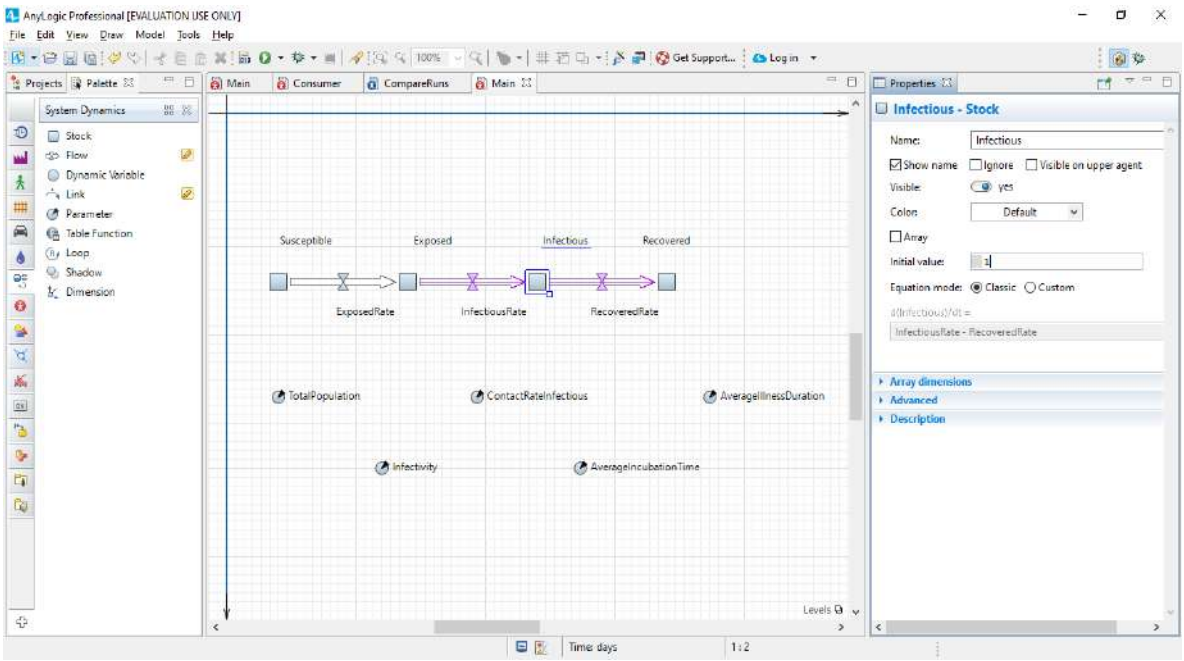
•ContactRateInfectious = 1.25

•AverageIncubationTime = 10

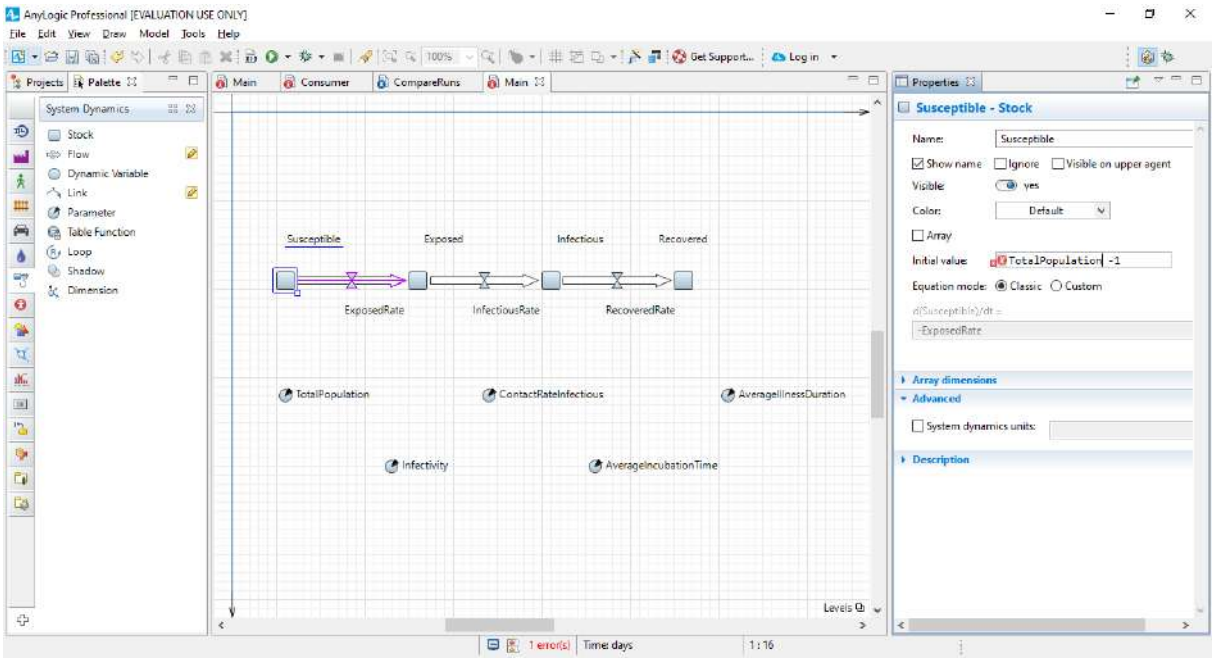
•AverageIllnessDuration = 15



Define the number of infected people by specifying 1 as the Initial Value of the stock Infectious.

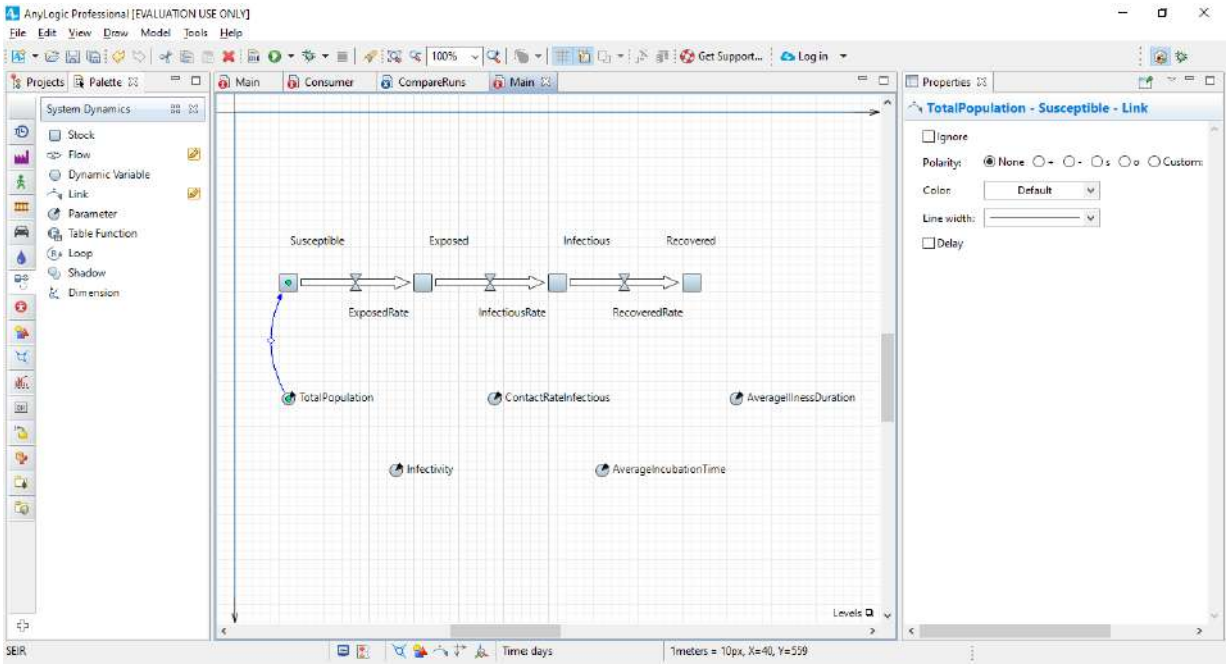


Define the Initial Value for the stock Susceptible: TotalPopulation-1.



Draw a dependency link from TotalPopulation to Susceptible.

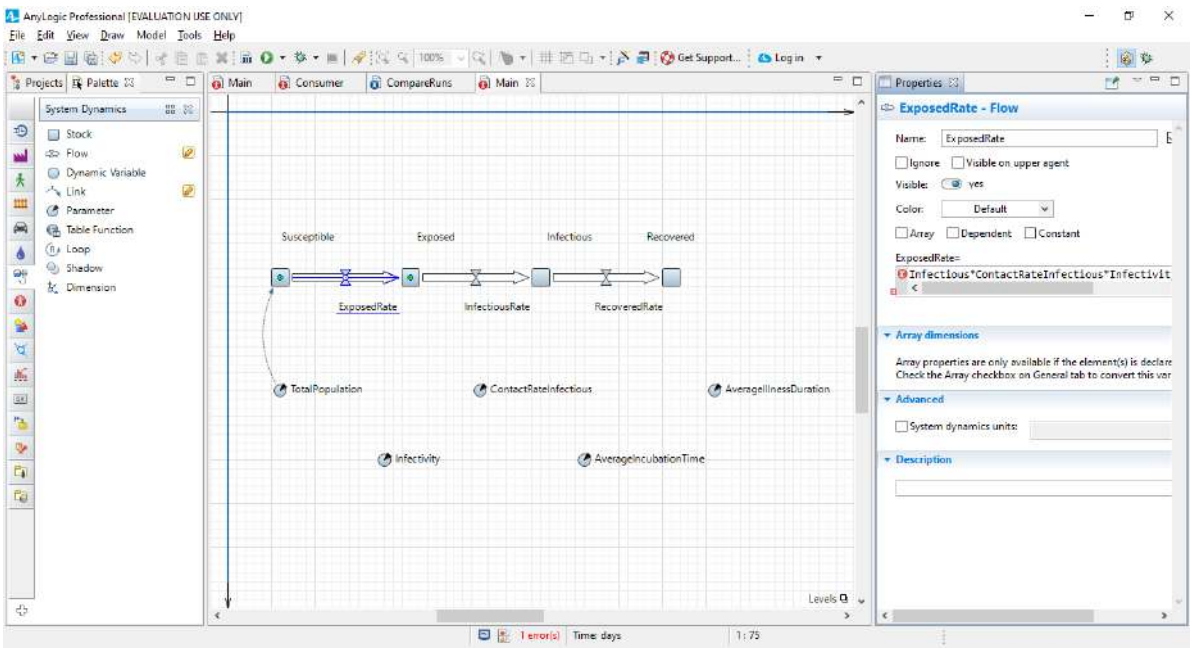
In the System Dynamics palette, double-click the Link element, click TotalPopulation, and then click the stock Susceptible. You should see the link with small circles drawn on its end points:



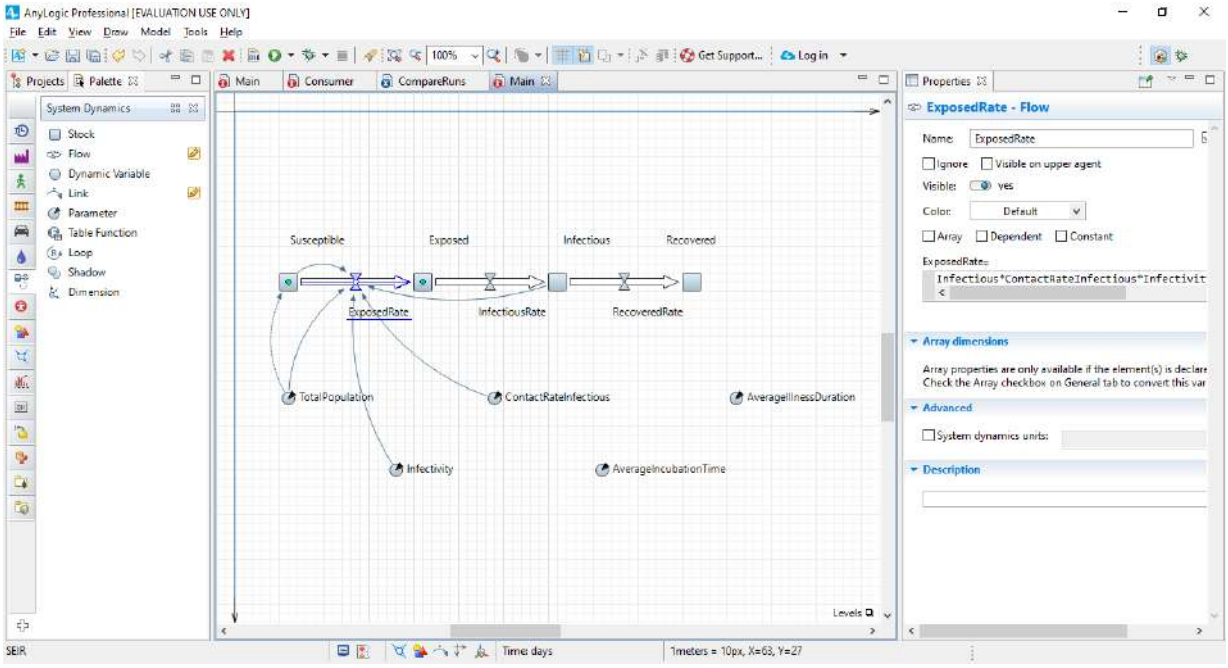
Let's define the formula for the flow ExposedRate.

Click the flow and define the following formula using the Code Completion assistant:

Infectious*ContactRateInfectious*Infectivity/TotalPopulation+Susceptible



Right-click ExposedRate flow in the graphical diagram and choose Fix Variable Links > Create Missing Links from the context menu. Afterward, you should see the links in the stock and flow diagram:



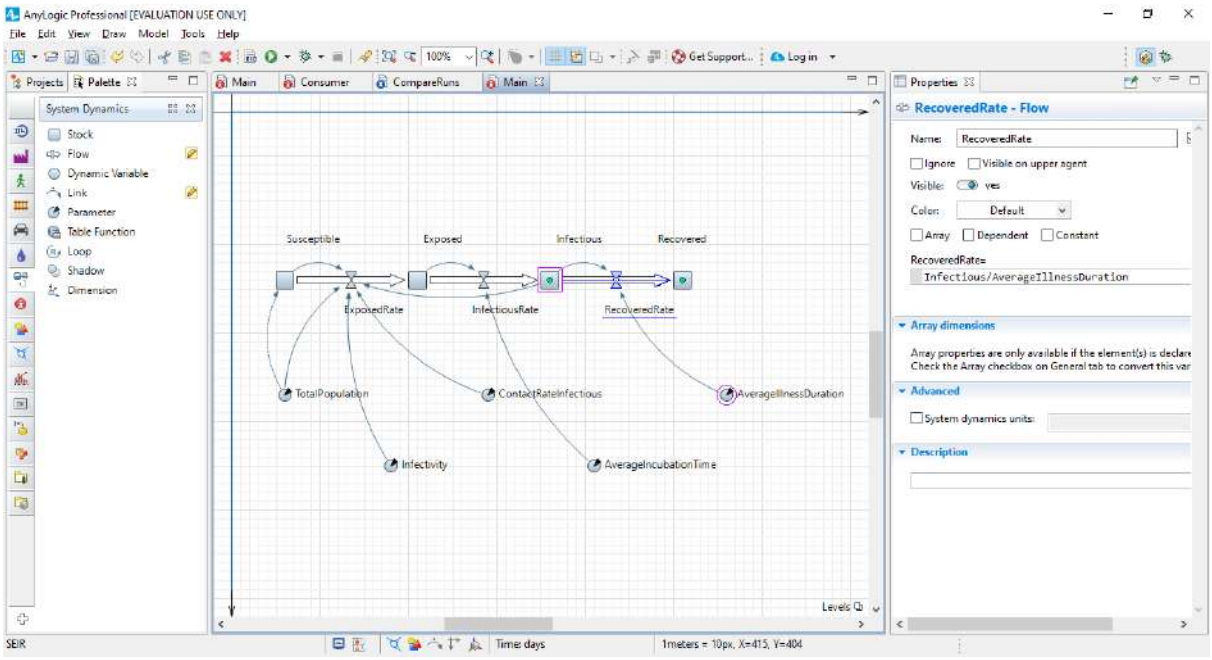
Define the following formula for InfectiousRate:

Exposed/AverageIncubationTime

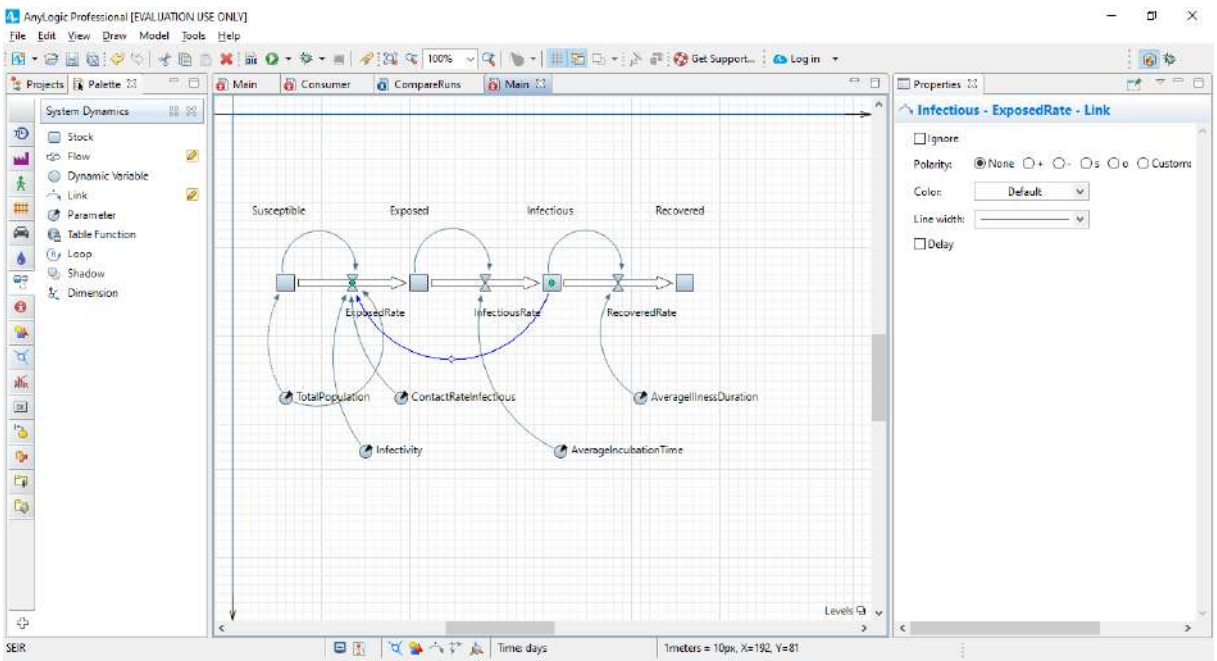
Define the following formula for RecoveredRate:

Infectious/AverageIllnessDuration

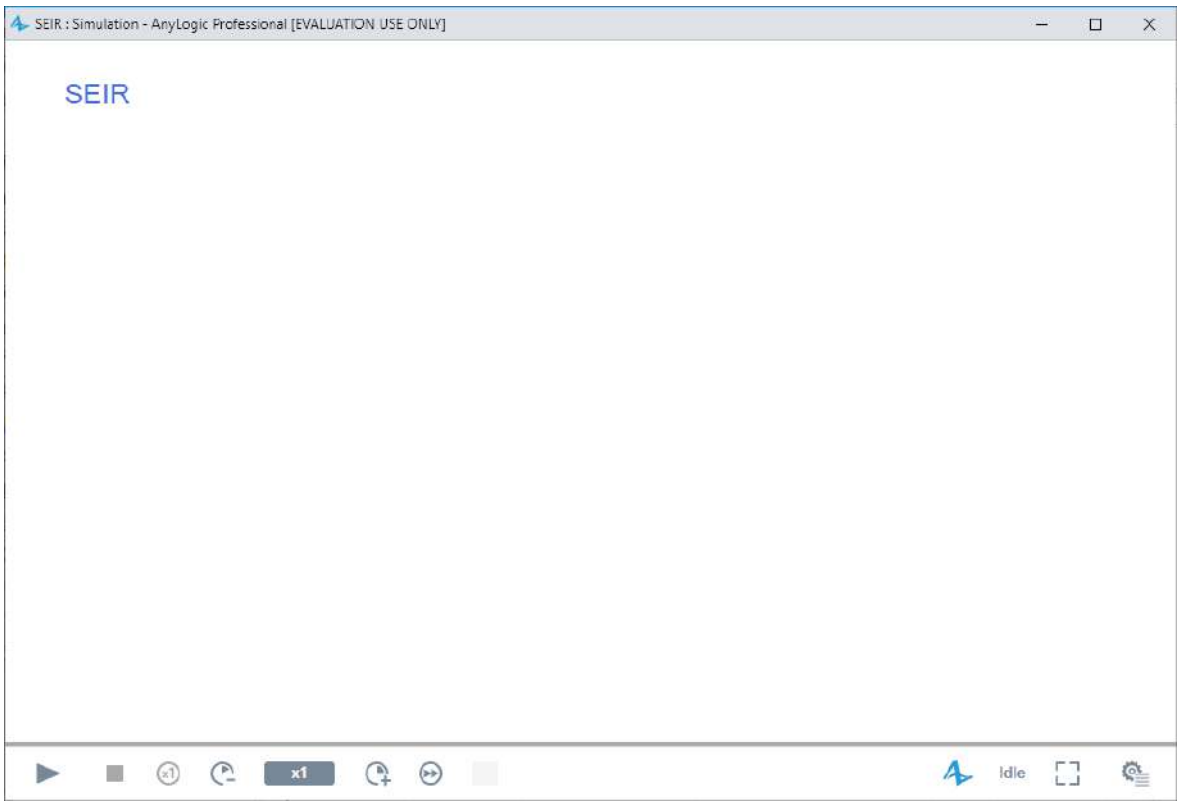
Draw the missing dependency links, and your stock and flow diagram should resemble the following image:

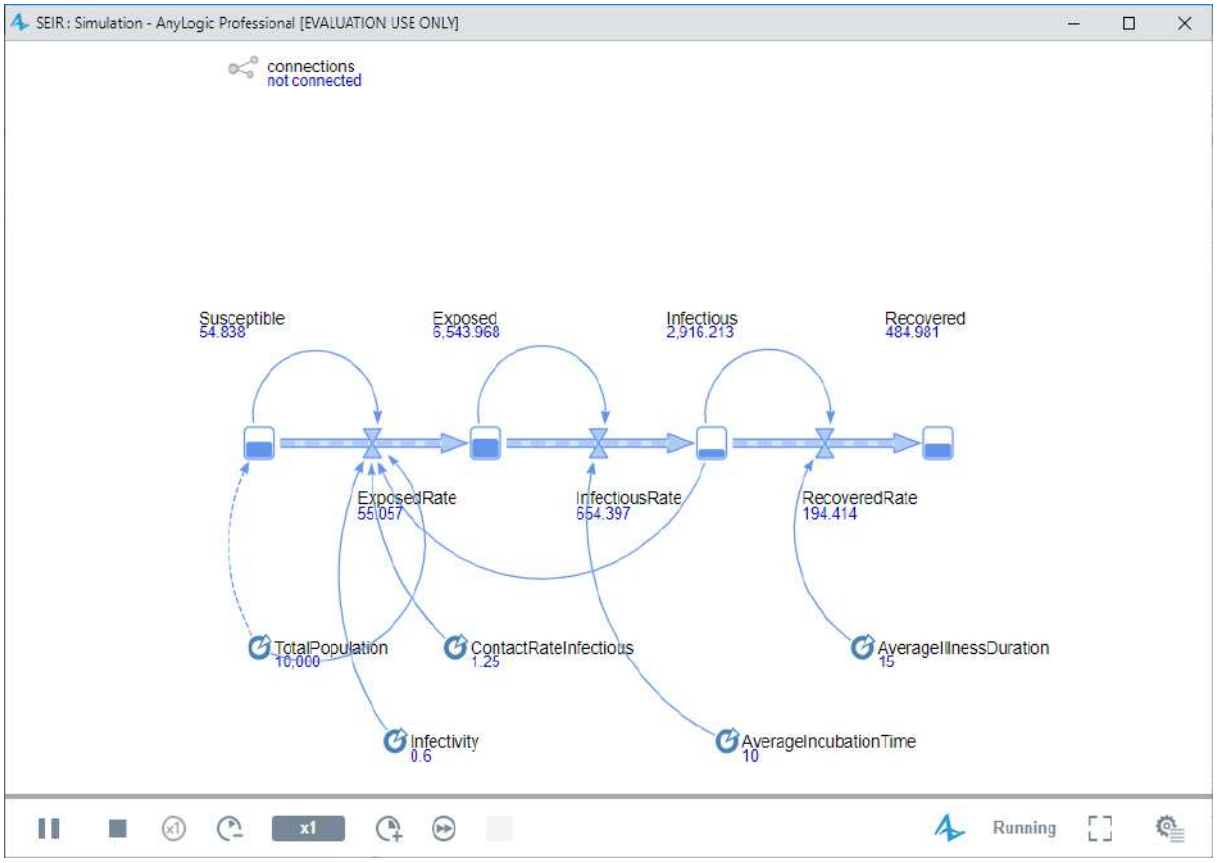


Adjust the appearance of dependency links. Modify the link's bend angles to make the diagram match the figure below. To adjust the link's bend angle, select it, and then drag the handle in the middle of the link.



Run the model and inspect the dynamics using the variable's inspect windows. To open a variable's, inspect window, click the variable to select it. To resize the window, drag its lower right corner.


SEIR : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
SEIR
x1
Idle

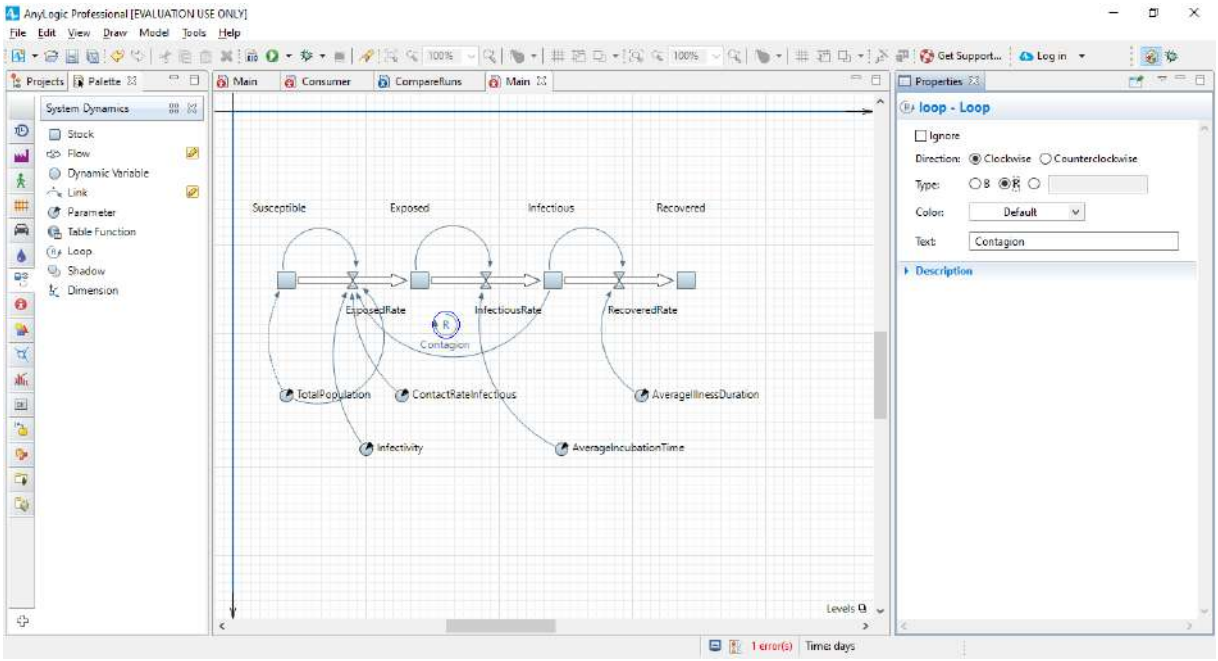

SEIR : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
connections
not connected
Susceptible
54,838
Exposed
6,543.968
Infectious
2,916.213
Recovered
484.981
ExposedRate
55.057
InfectiousRate
654.397
RecoveredRate
194.414
TotalPopulation
10,000
ContactRateInfectious
1.25
AverageIllnessDuration
15
Infectivity
0.6
AverageIncubationTime
10
x1
Running

Adding a plot to visualize dynamics

We'll add a loop identifier for one loop to show you.

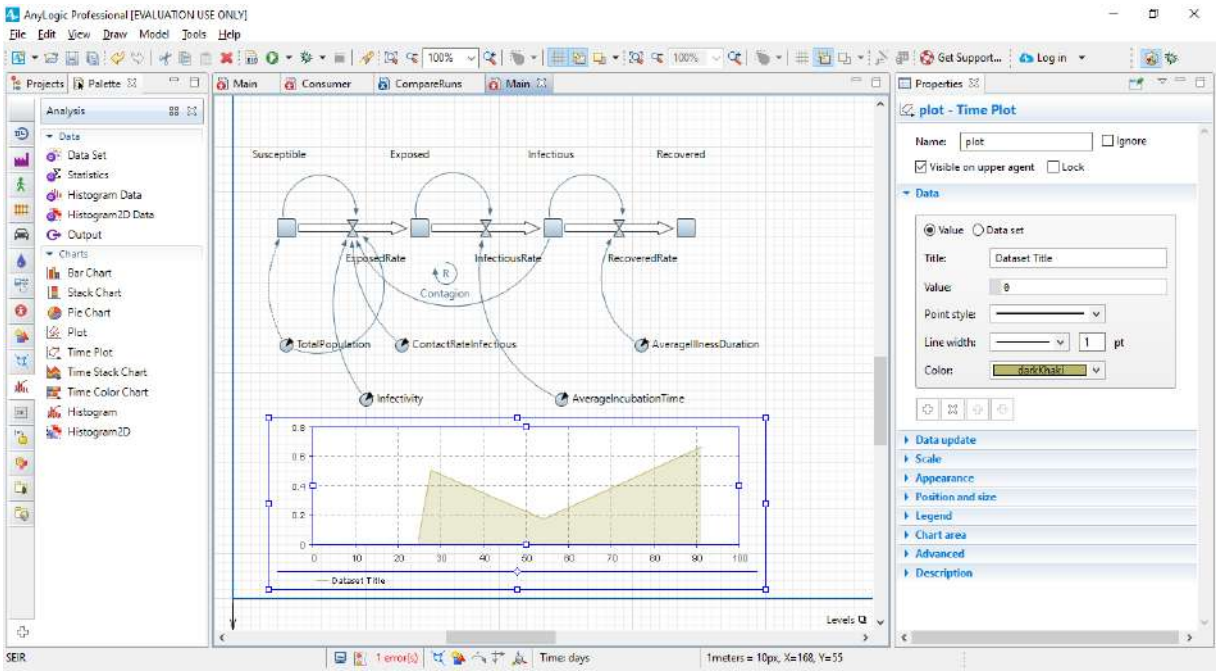
Drag the Loop element from the System Dynamics palette on to the diagram, and then place it as shown in the figure.

Go to the loop's Properties, change its Type to R(stands for Reinforcing), leave the default Clockwise Direction, and specify the text AnyLogic will display near the loop icon: Contagion.



Let’s add a time chart to plot Susceptible, Exposed, Infectious, and Recovered people.

Drag the Time Plot from the Analysis palette on to the diagram, and extend the time plot as shown in the figure below:

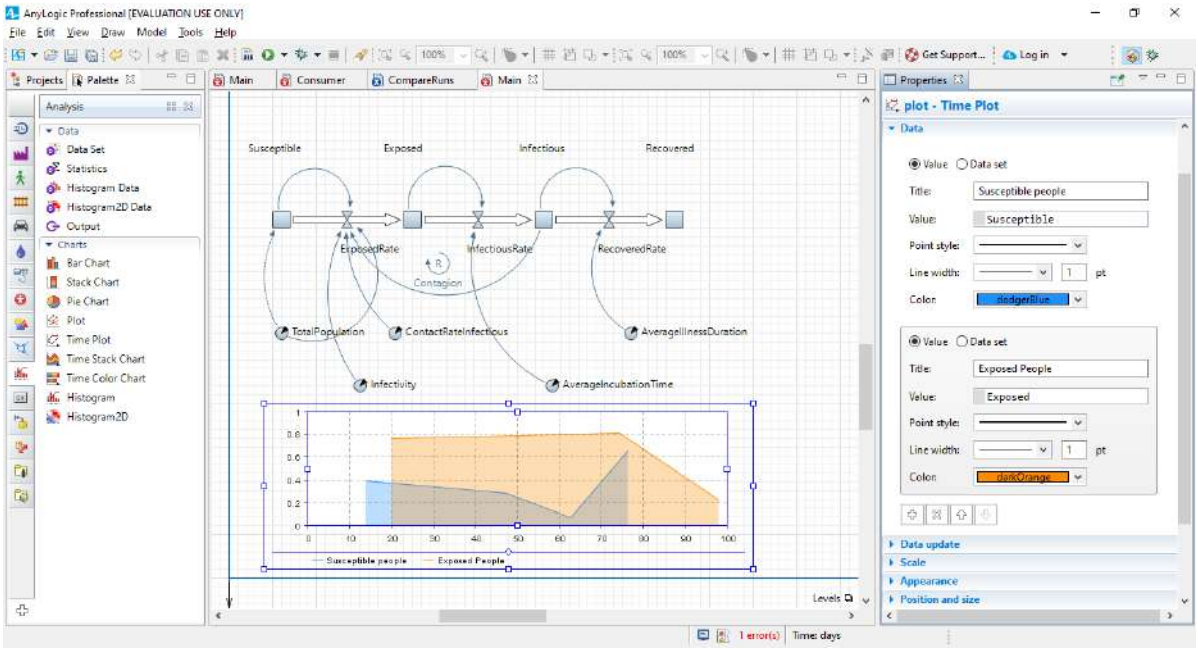


In the Properties view, go the section Data and click the Plus button to add a new data item.

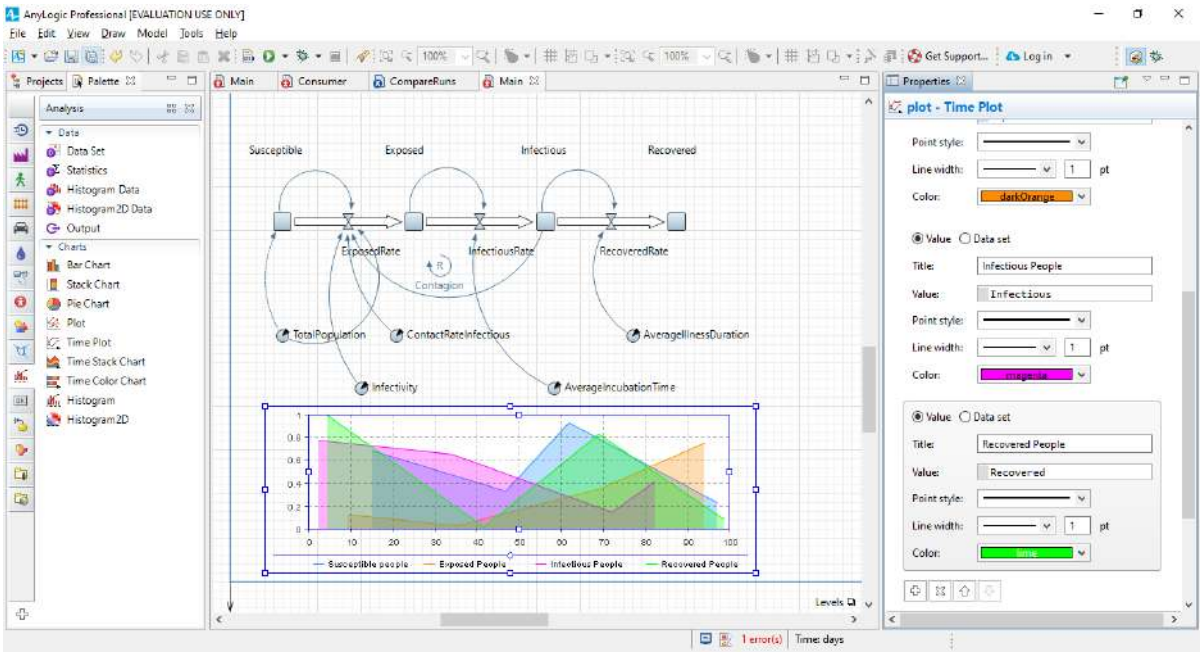
Modify the data item's properties:

Title : Susceptible people – title of the data item.

Value : Susceptible (use Code Completion Master).



Add three data items to display the values of stocks Exposed, Infectious, and Recovered in the same way -and don't forget to define the corresponding Titles.

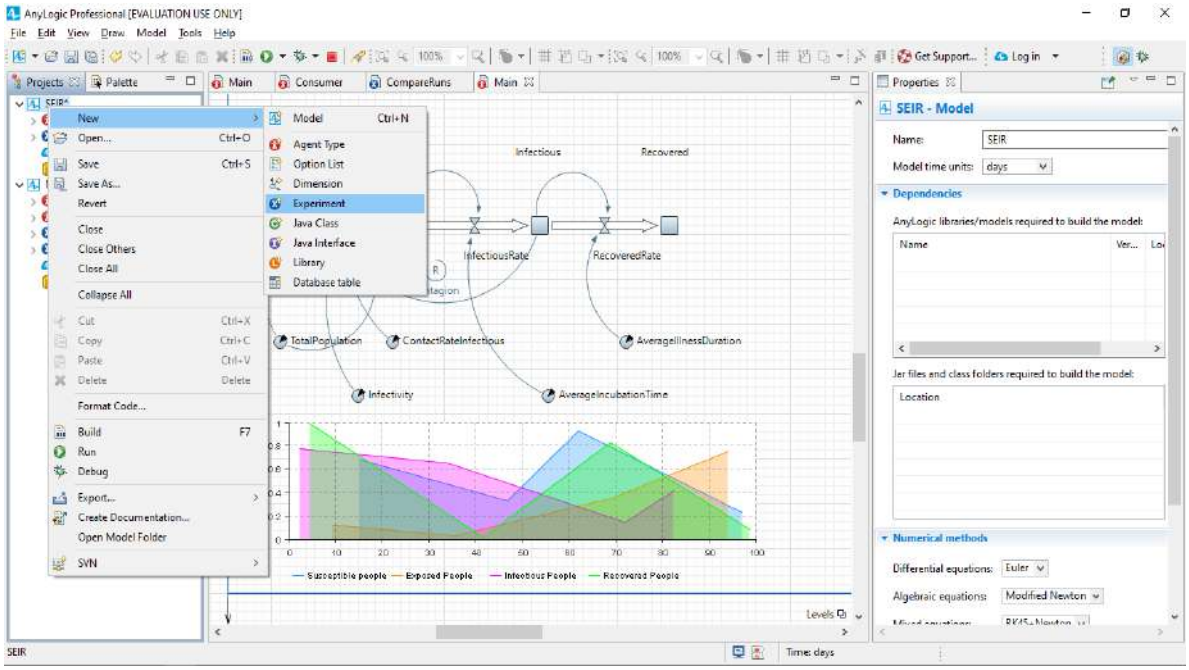


We've finished our last model. Now, run the model and use the chart you added to view its dynamics.

Parameter Variation

The parameter variation experiment allows us to create a complex model simulation that performs a series of single model runs that vary by one or many parameters. After the experiment is complete, AnyLogic can display each run's results on a single diagram to help us better understand how the varying parameters affected our model's results.

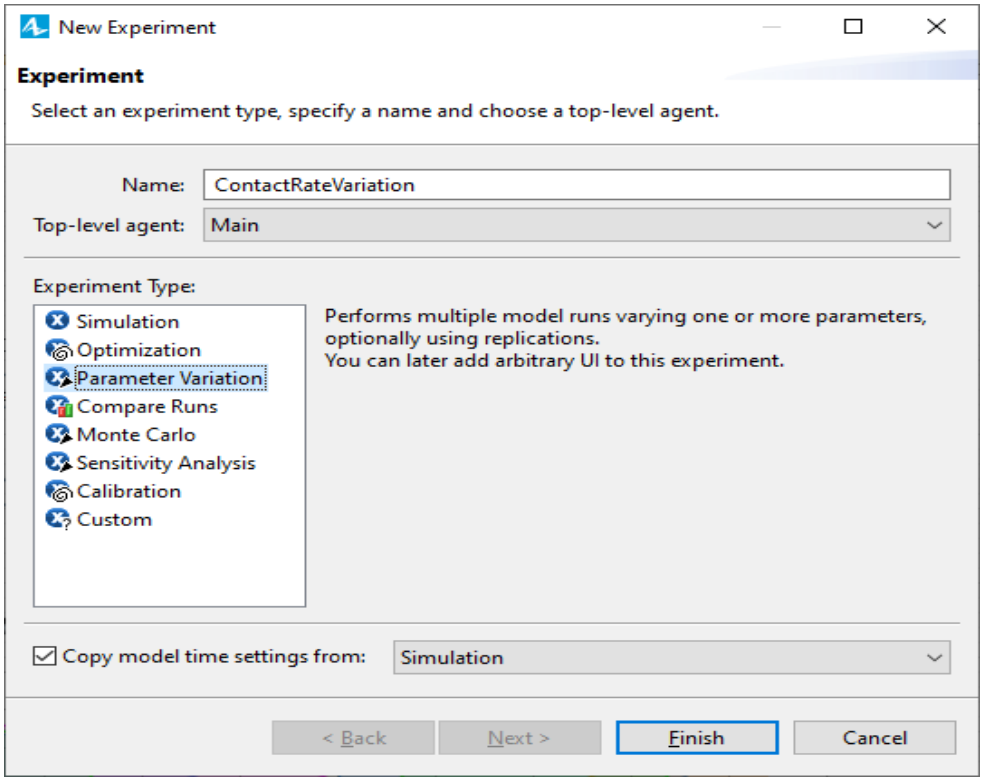
To add an experiment to the model, right-click the model item (SEIR) in the Projects tree, point to New, and then click Experiment.



In the New Experiment wizard's Namefield, type ContactRateVariation.

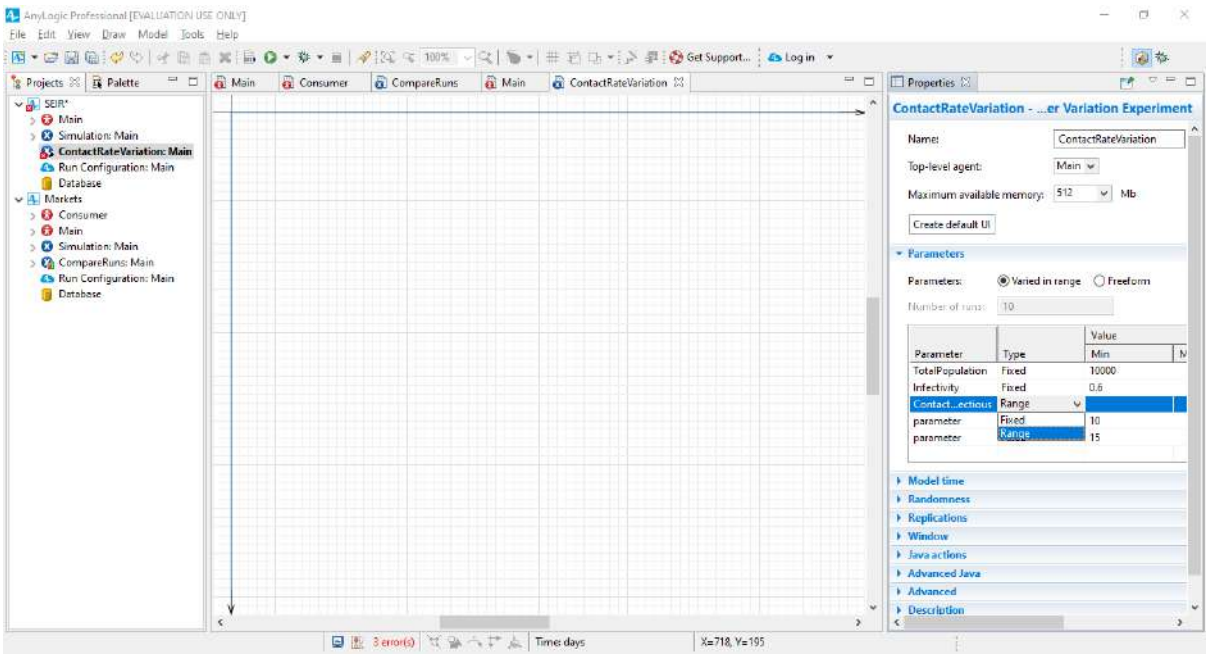
AnyLogic will automatically select the Main agent type as the Top-level agent.

In the Experiment Type area, click Parameter Variation, and click Finish.

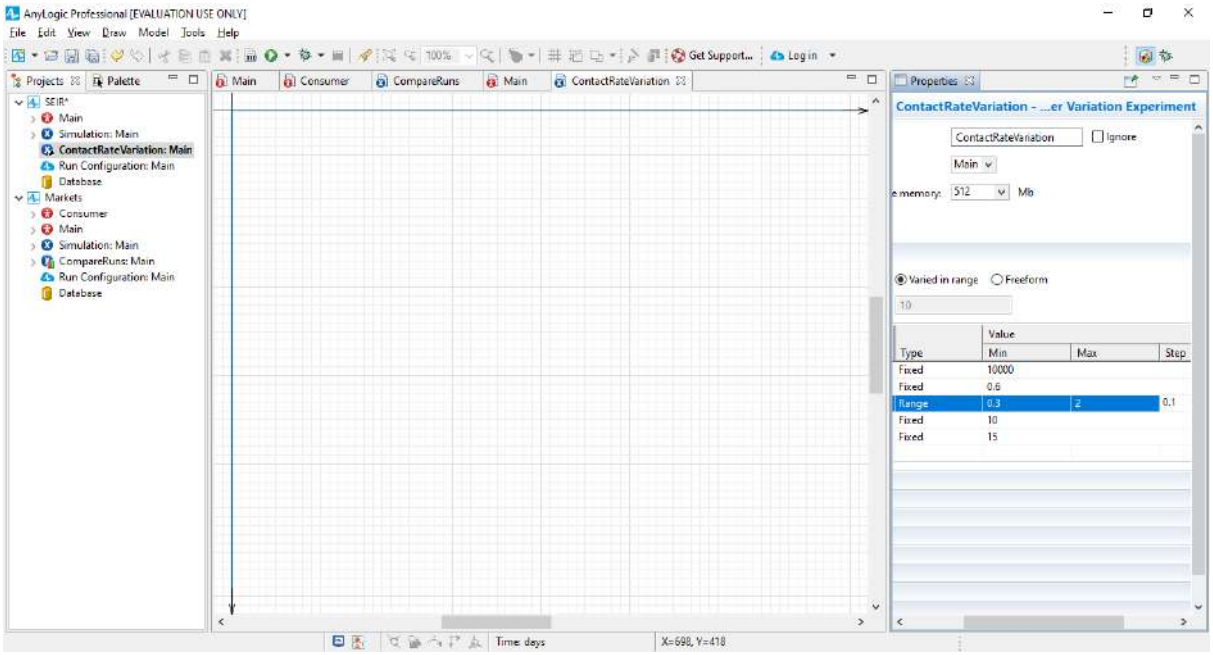


In the experiment's properties, open the Parameters section. The parameters of the experiment's top-level agent (in our case, Main) display.

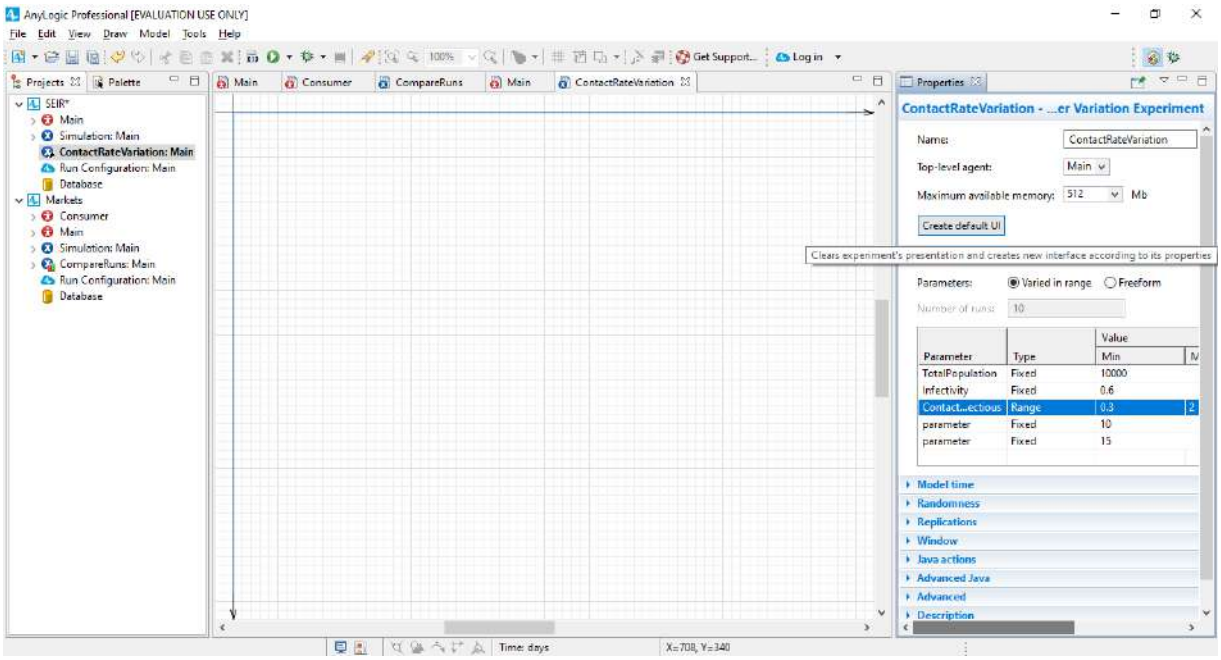
To ensure our experiment varies the contact rate, locate the table's ContactRateInfectious parameter and change its Type to Range



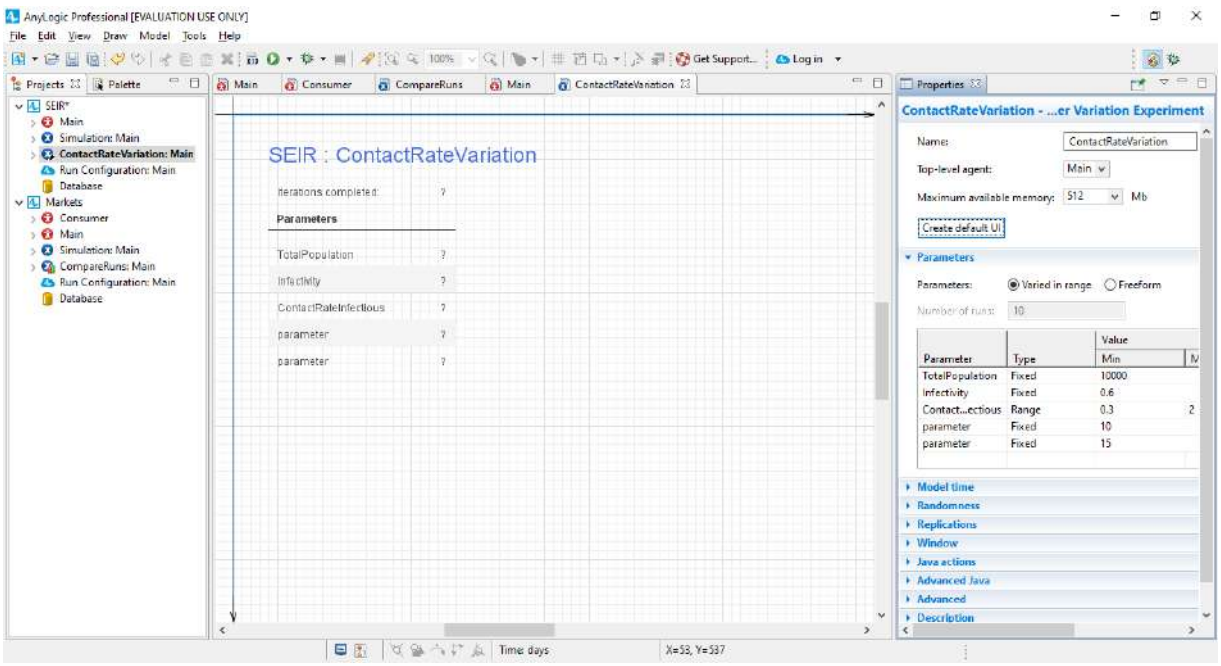
Set the parameter's minimum and maximum values by setting Min: 0.3 and Max: 2 with a Step of 0.1.



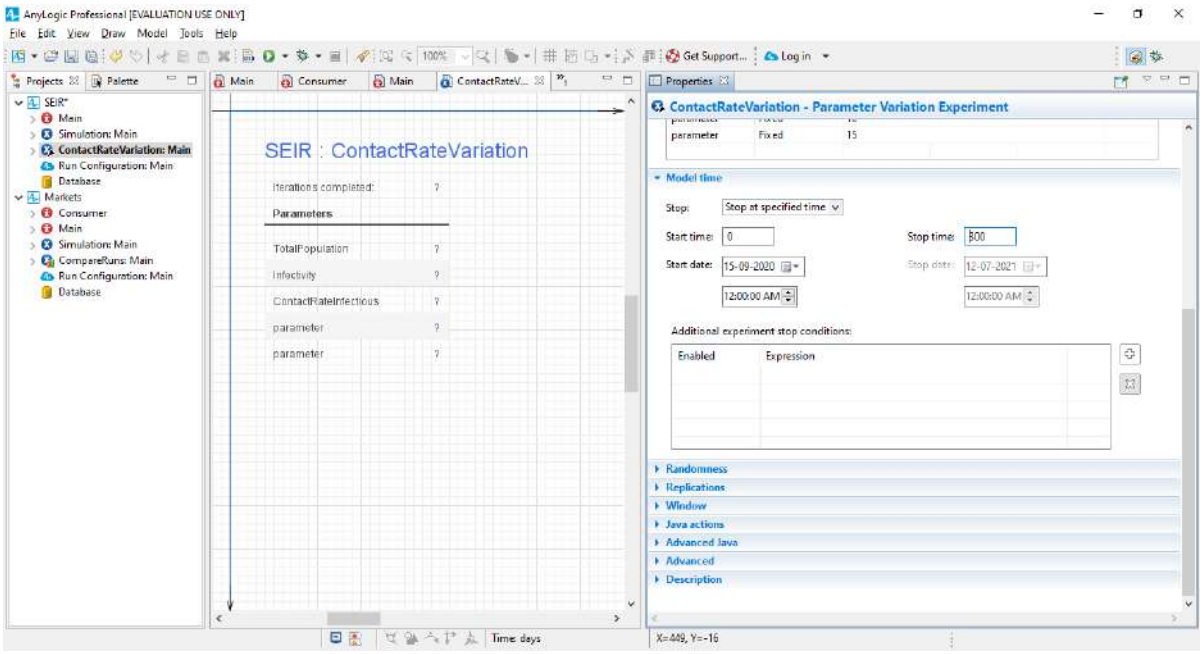
In the Properties area, click Create default UI.



The experiment diagram will display the simple user interface

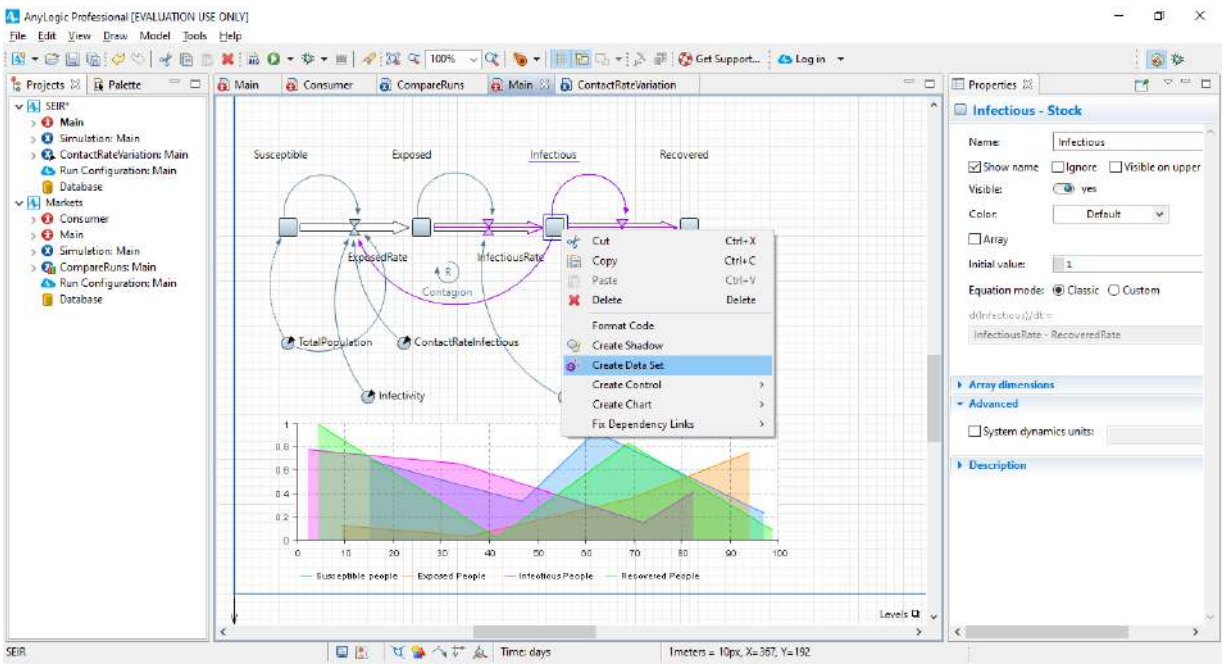


To ensure each run simulates exactly 300 days, we need to limit the model's lifetime to 300 days. Click ContactRateVariation in the Projects tree to open its properties. In the Properties view, open the Model time section, select Stop at specified time from the Stop list, and type 300in the Stop timebox



Now, we'll add a time plot to display our experiment's results. Our first step is to gather data about the number of infectious persons.

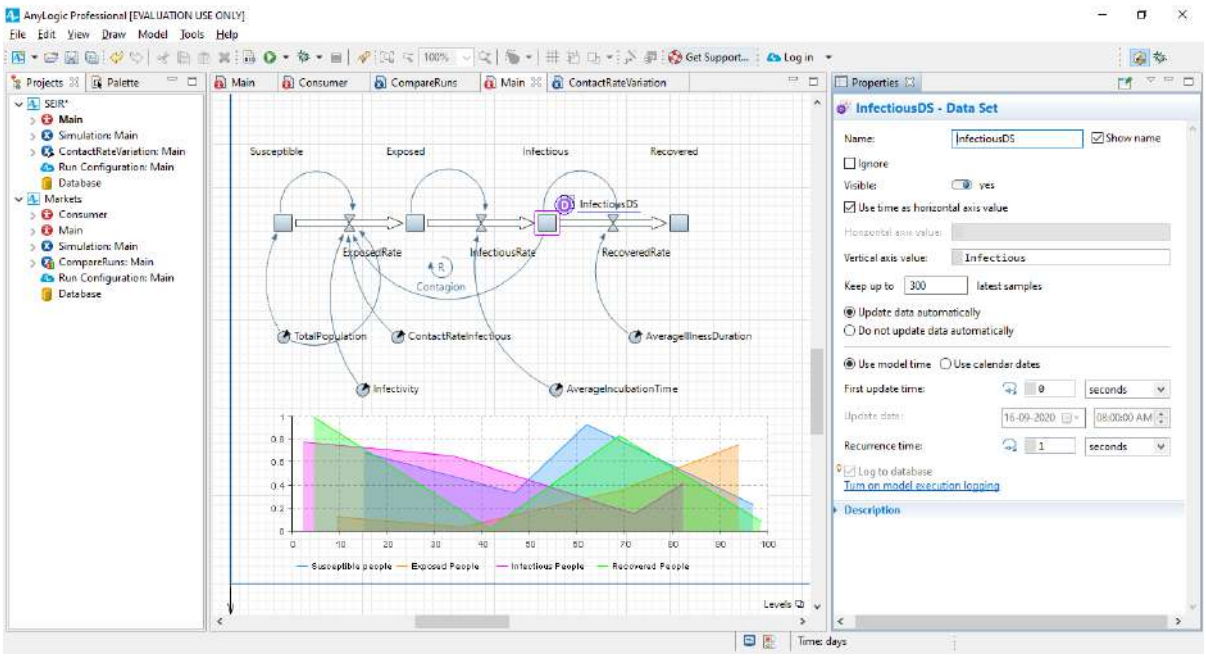
Open the Main diagram, right-click the Infectious stock and then click Create Data Set.



After the InfectiousDS data set displays, navigate to its properties. Since we want to view the infectious disease's dynamics, leave the Use time as horizontal axis value checkbox selected.

Select Update data automatically and leave Recurrence time: 1 to add one data sample to our dataset for each model life day.

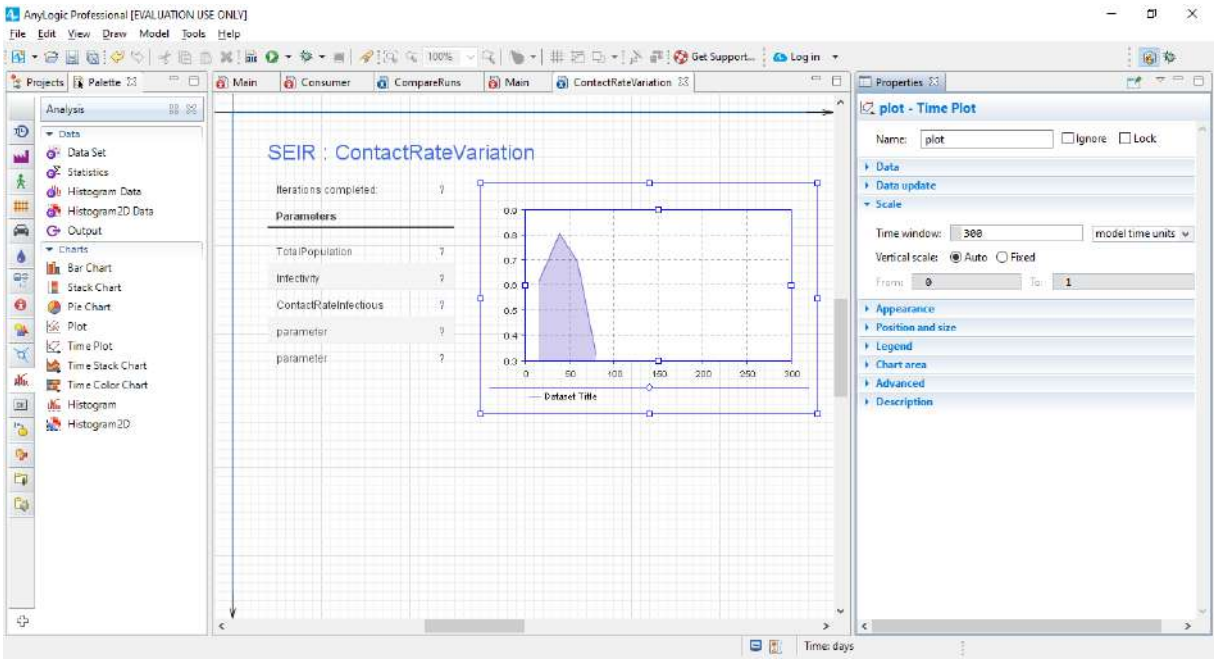
To obtain data samples for the whole model run, set the dataset to Keep up to 300 latest samples.



We're ready to add a chart to the ContactRateVariation experiment diagram that will display our results.

Open ContactRateVariation diagram, and drag the Time Plot from the Analysis palette.

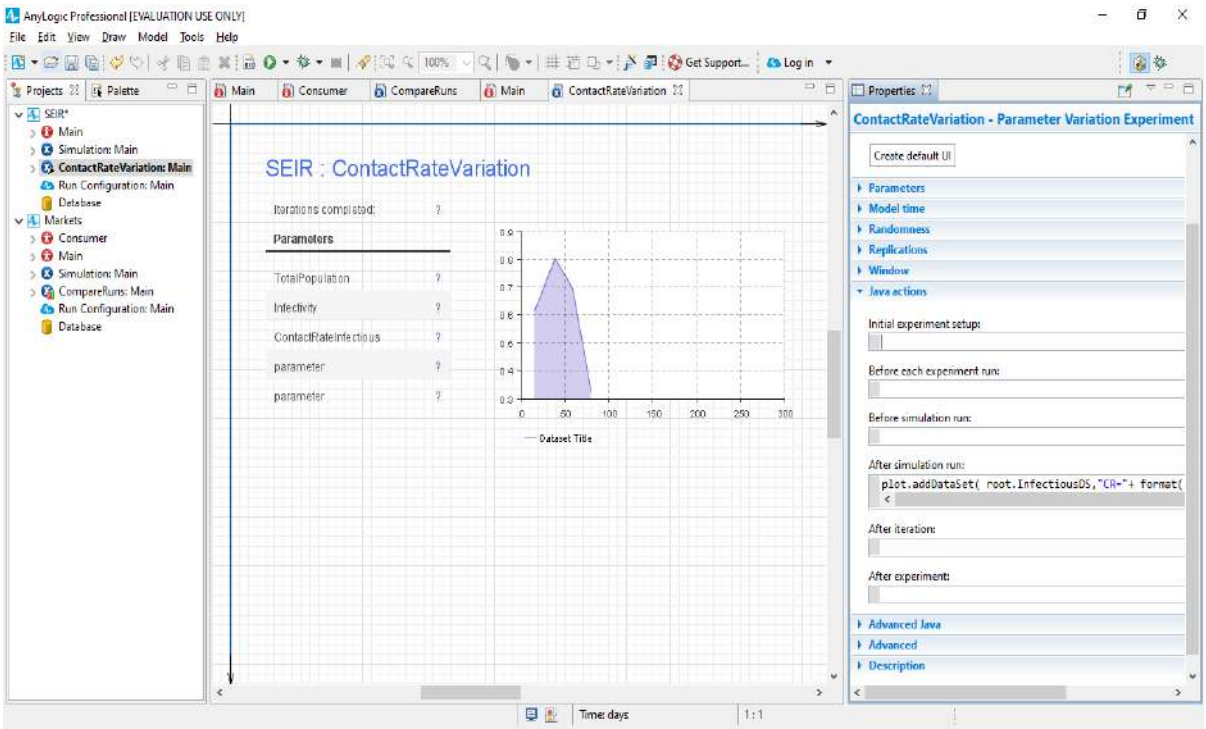
Open the time plot's properties. In the Scale section, ensure the time plot displays data for 300 model time units by setting the Time window to 300 model time units. Enlarge the area available for the plot's legend by dragging the diamond handle toward the top of the screen.



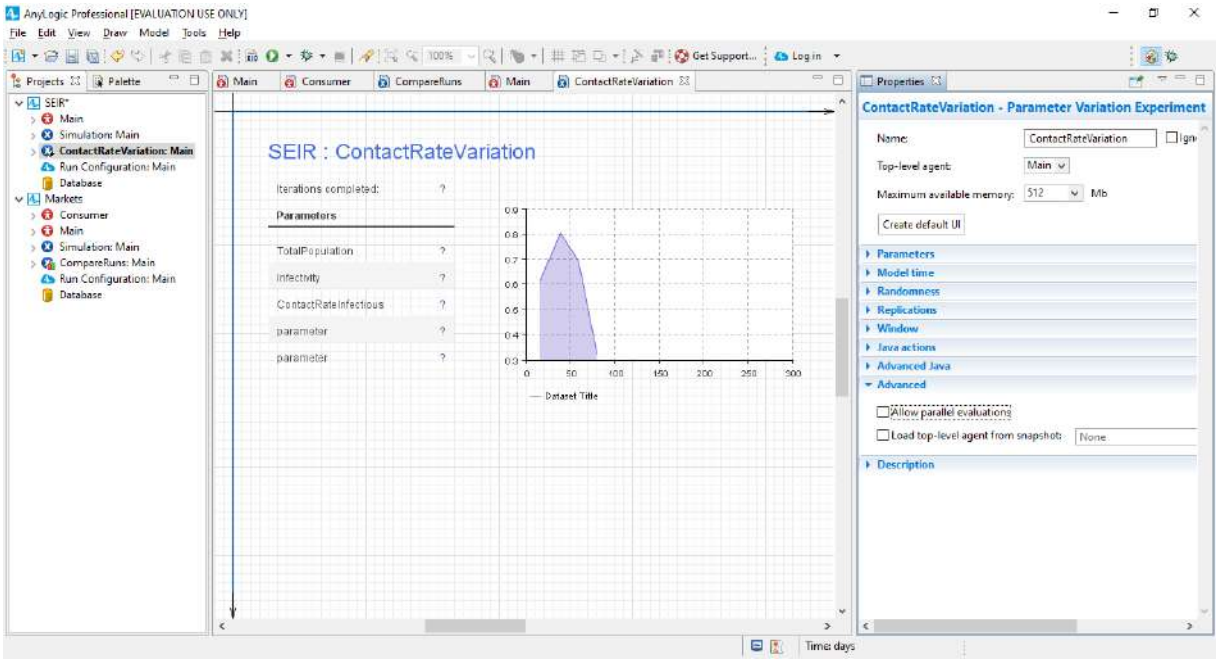
The plot's curves will each display the results from one model run: the history of disease spread for a contact rate collected by the InfectiousDS dataset.

Click ContactRateVariation in the Projects tree to open its properties, and then add data to the plot by navigating to the Java actions section and typing the following code in the After simulation run field:

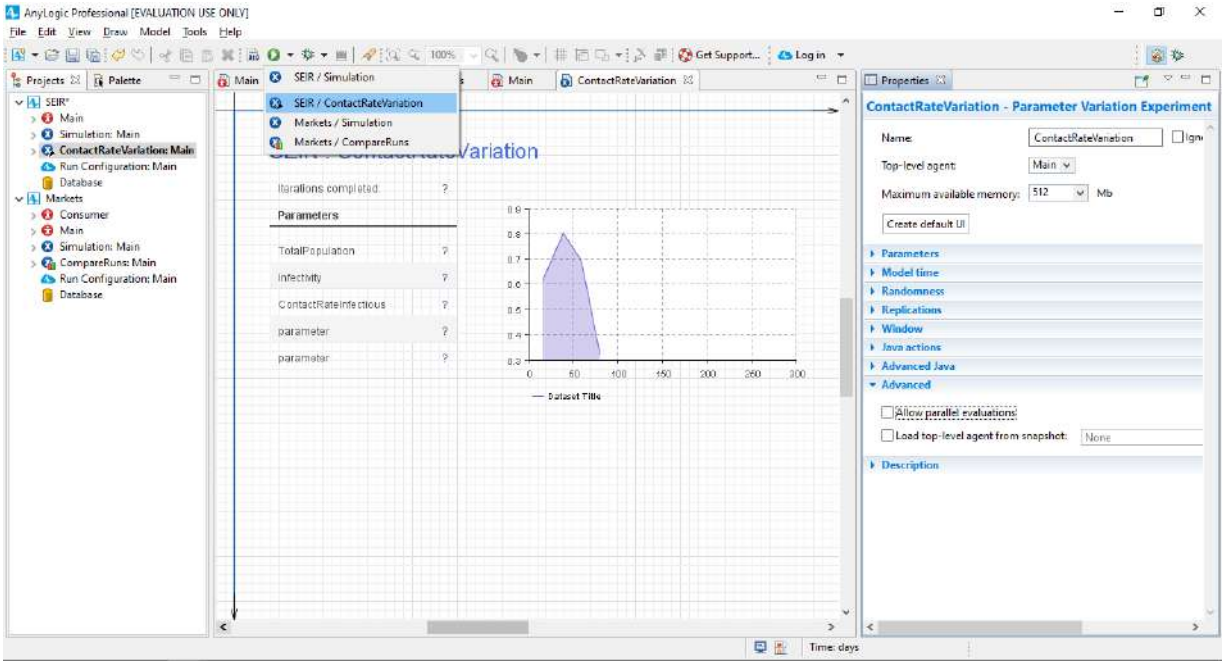
plot.addDataSet(root.InfectiousDS,"CR="+ format(root.ContactRateInfectious));



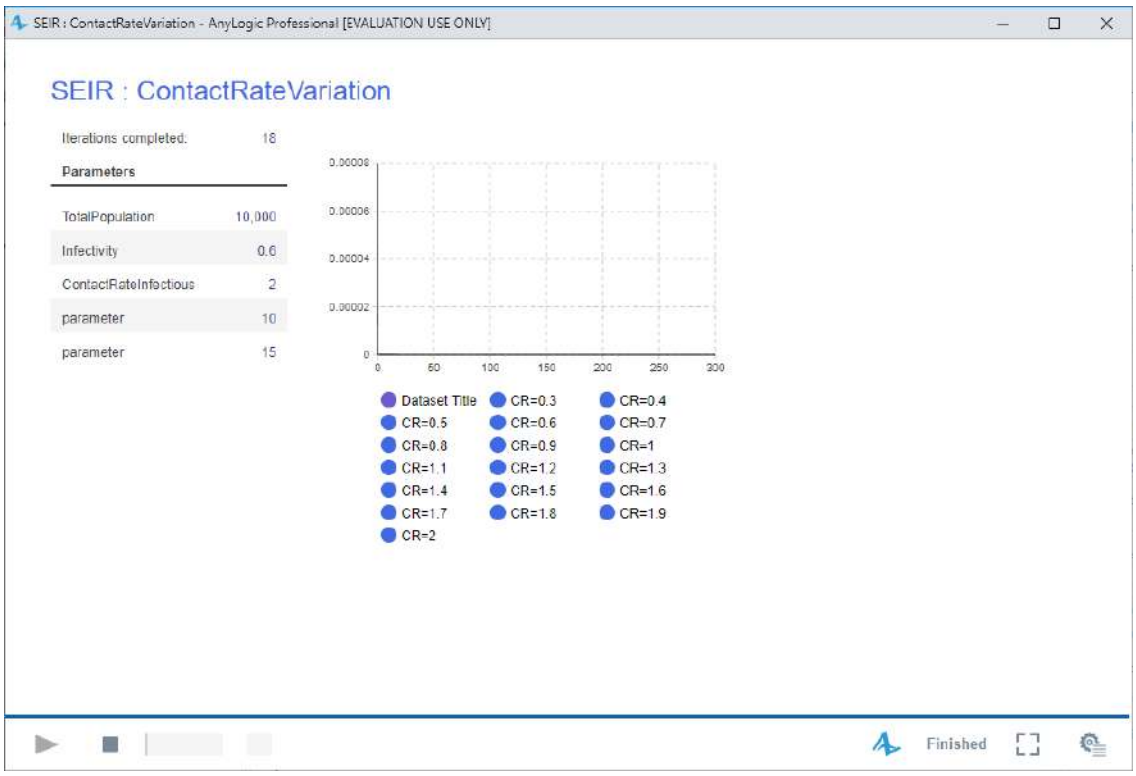
Open the ContactRateVariation experiment properties Advanced section and then clear the Allow parallel evaluations checkbox.



We're ready to run the experiment and use our chart to observe the data we've gathered from multiple simulation runs. In the toolbar, on the Runlist, select SEIR / ContactRateVariation.



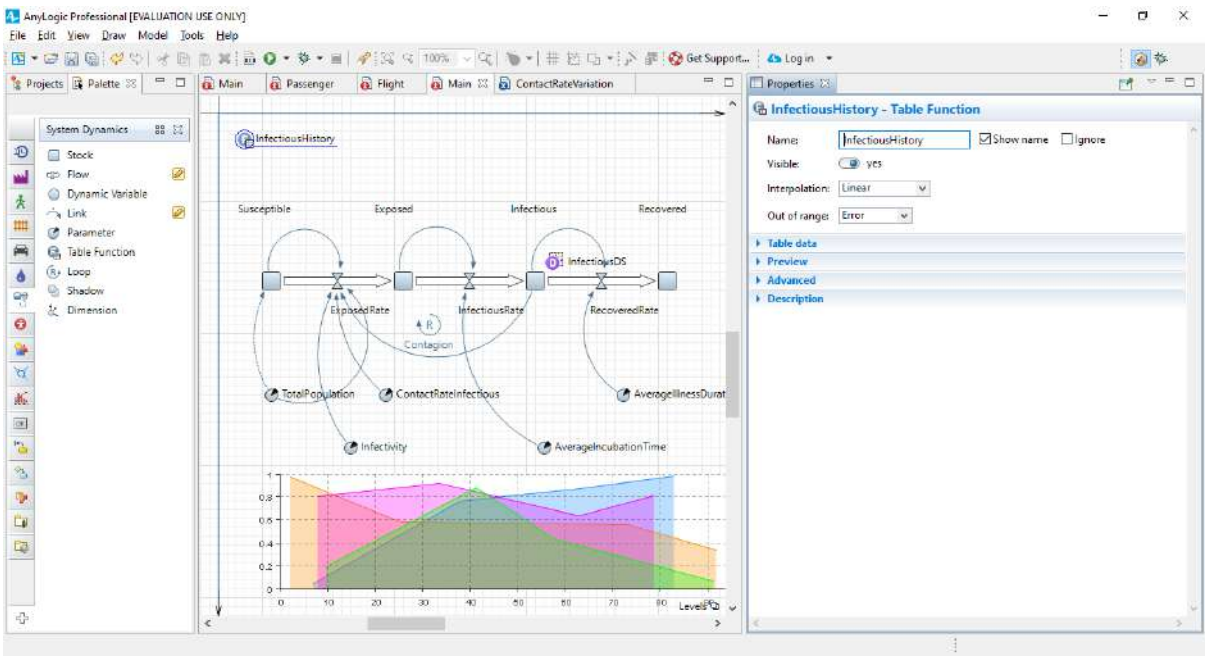
In the presentation window, click Run.



Calibration experiment

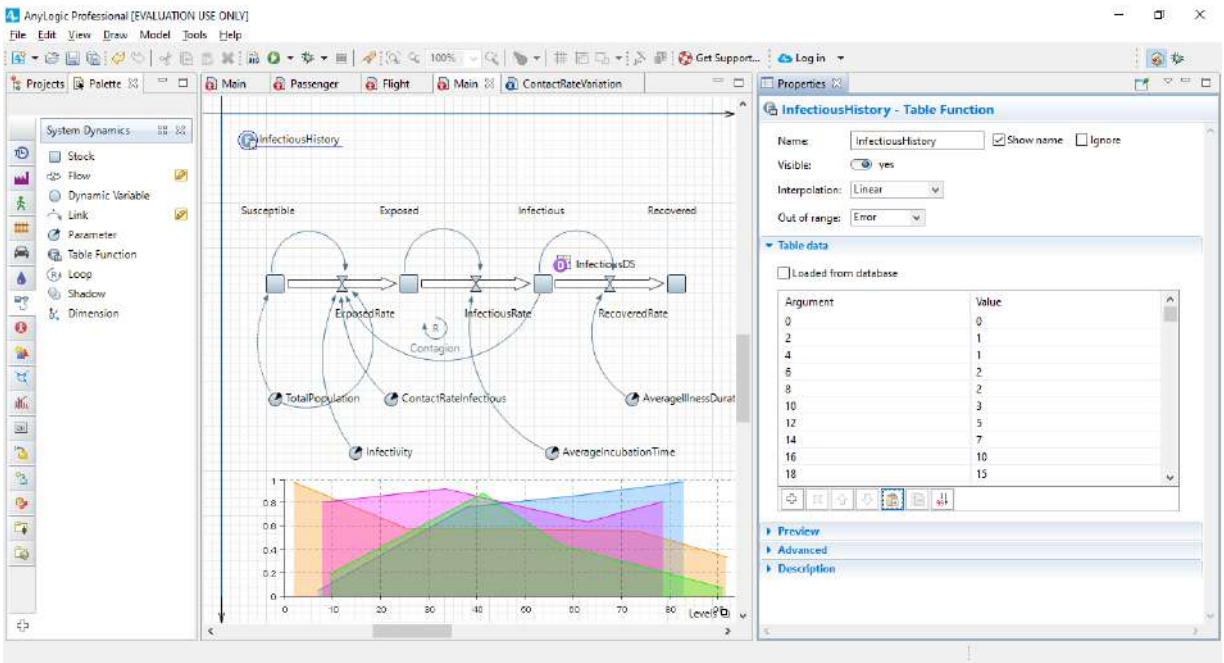
- Calibration experiment uses the built-in OptQuest optimizer to locate the model parameter values that correspond to the simulation output that best fits the given data.
- Calibration experiment iteratively runs the model, compares the model's output to the historical pattern, and then changes the parameter values. After a series of runs, the experiment will determine which parameter values produce the results that best match the historical pattern.

Open the Main diagram and add a Table function from the System Dynamics palette. Name it InfectiousHistory.

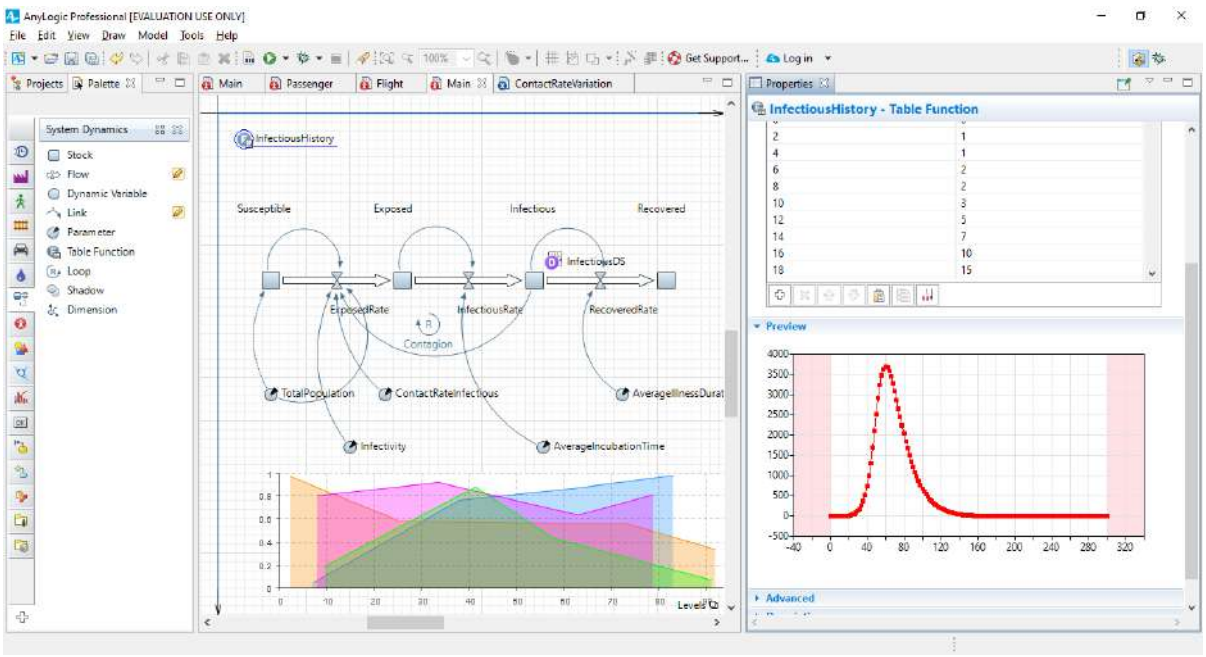


Open the HistoricData.txt file from AnyLogic folder/resources/AnyLogic in 3 days/SEIR. The AnyLogic folder is the location on your computer where you installed AnyLogic, such as Program Files/AnyLogic 7 Professional.

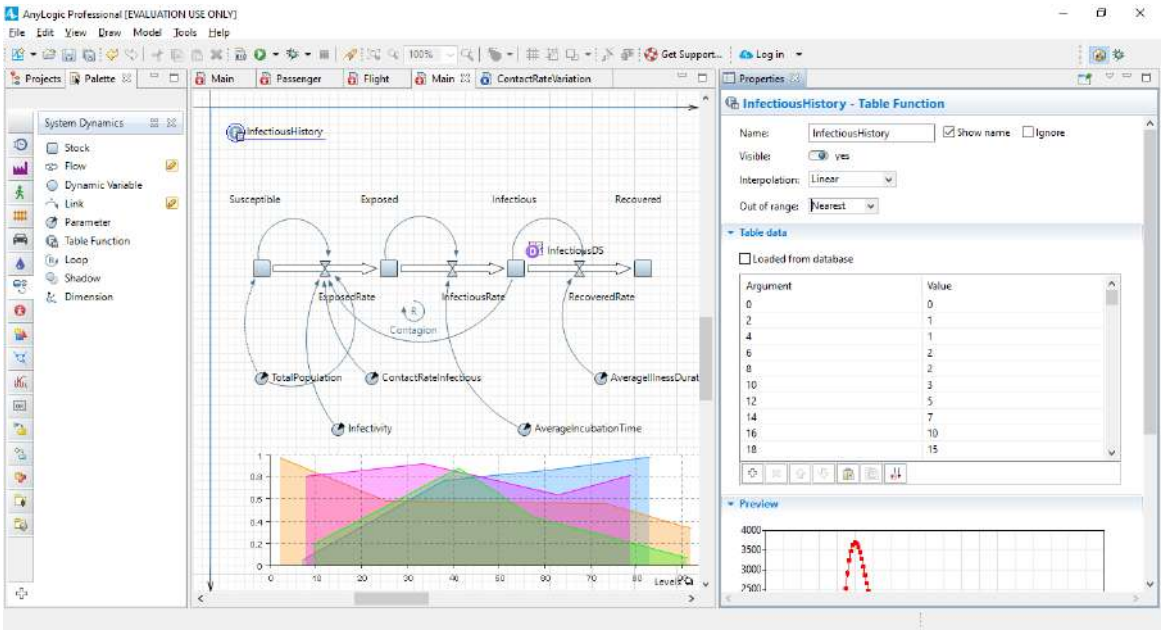
Copy the text file's contents to the Clipboard, go to the table function properties Table Data section, and then click the Paste from clipboard button. The Argument and Value columns will automatically update.



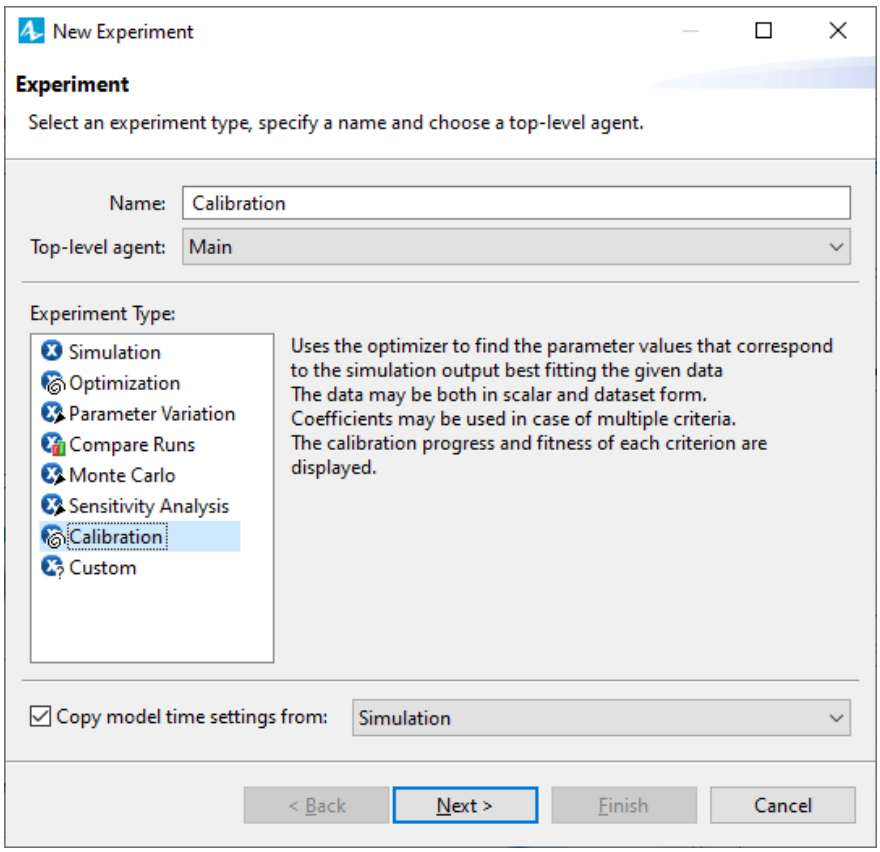
You can preview the curve built for the table function in the table function's properties' Preview section.



Set the Out of range option to Nearest to ensure the function correctly addresses cases where the function's argument exceeds the value of 300 that we defined in the Table data.



Right-click the model item (SEIR) in the Projects tree, point to New, and then click Experiment. In the New Experiment wizard, choose Calibration as the Experiment type and then click Next. This time, we'll use the wizard to set the parameters.



Change the parameter types we want to calibrate (Infectivity and ContactRateInfectious) from fixed to continuous, and then set the range's Min and Max values as shown in the figure below.

In the Criteria table shown below, enter the following information.

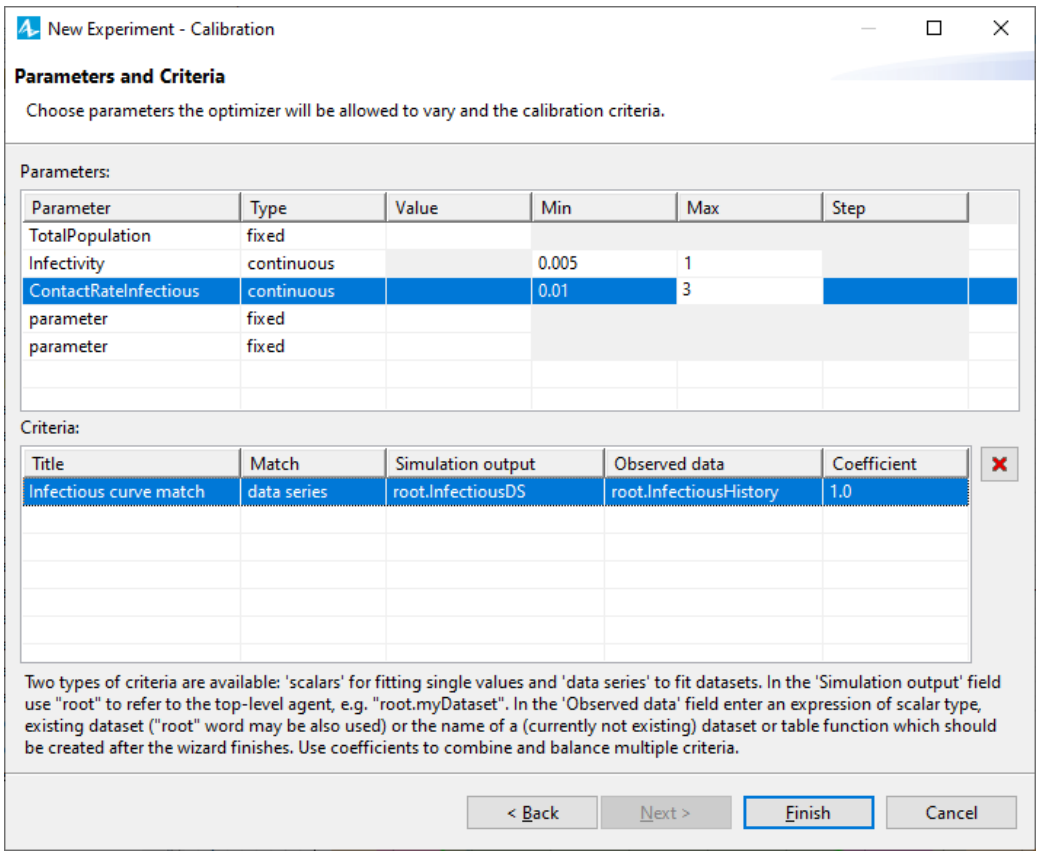
Title: Infectious curve match

Match: data series

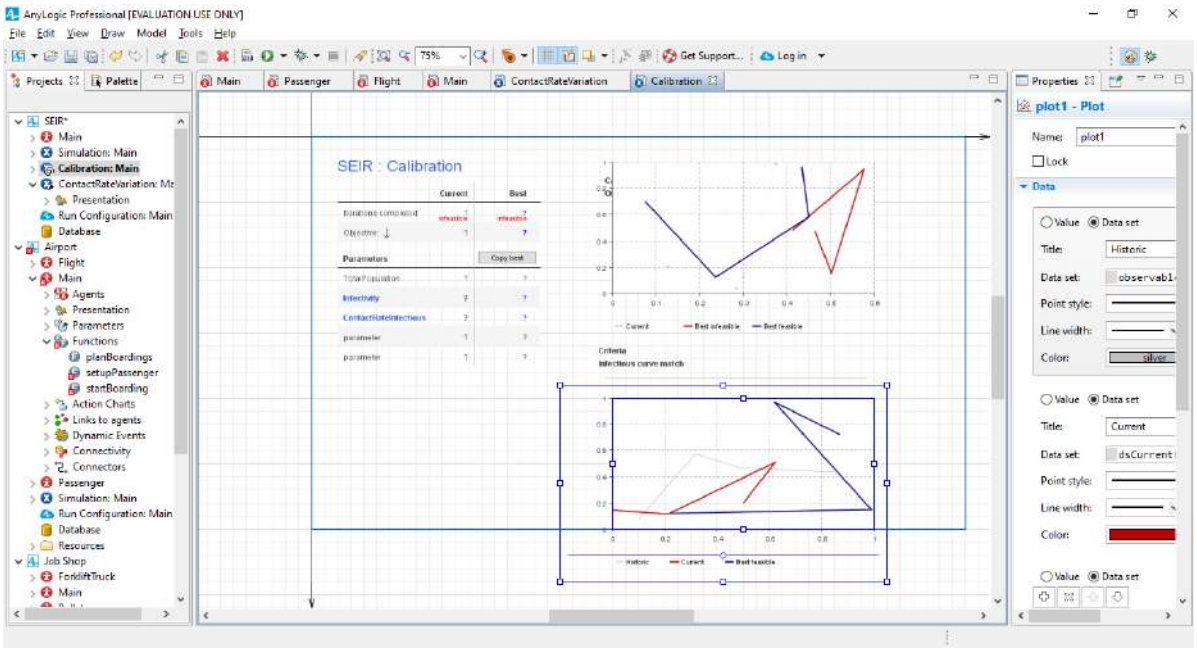
Simulation output: root.InfectiousDS

Observed data: root.InfectiousHistory

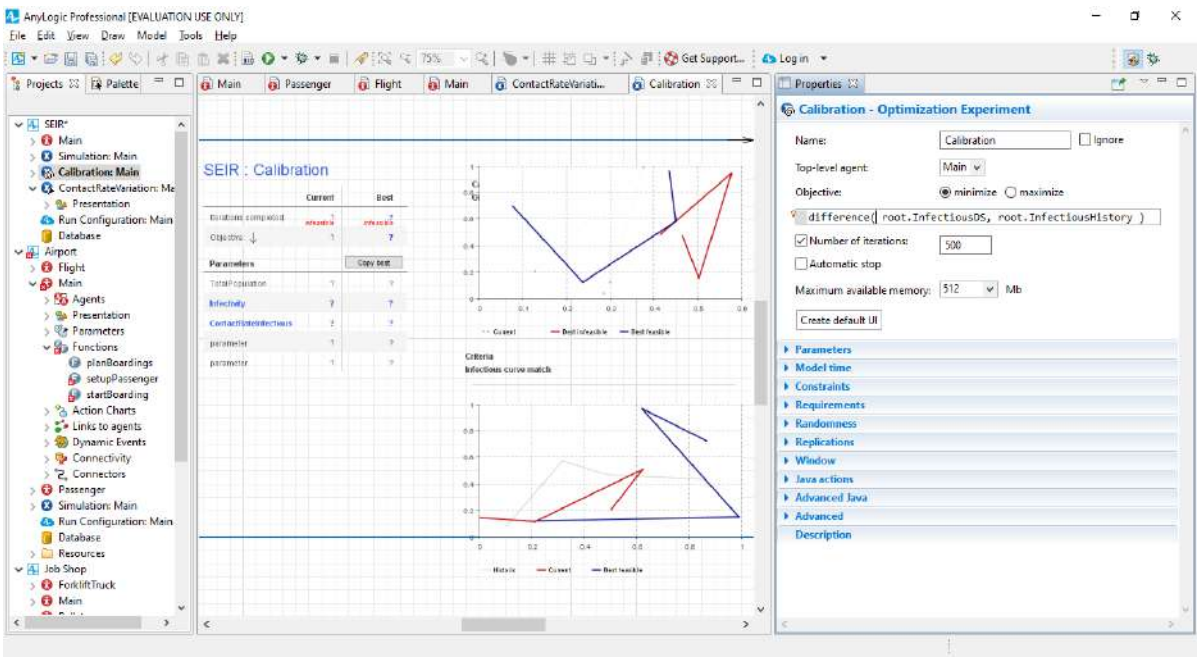
Coefficient: 1.0



Click Finish. The Calibration experiment diagram will display the configured user interface (UI).

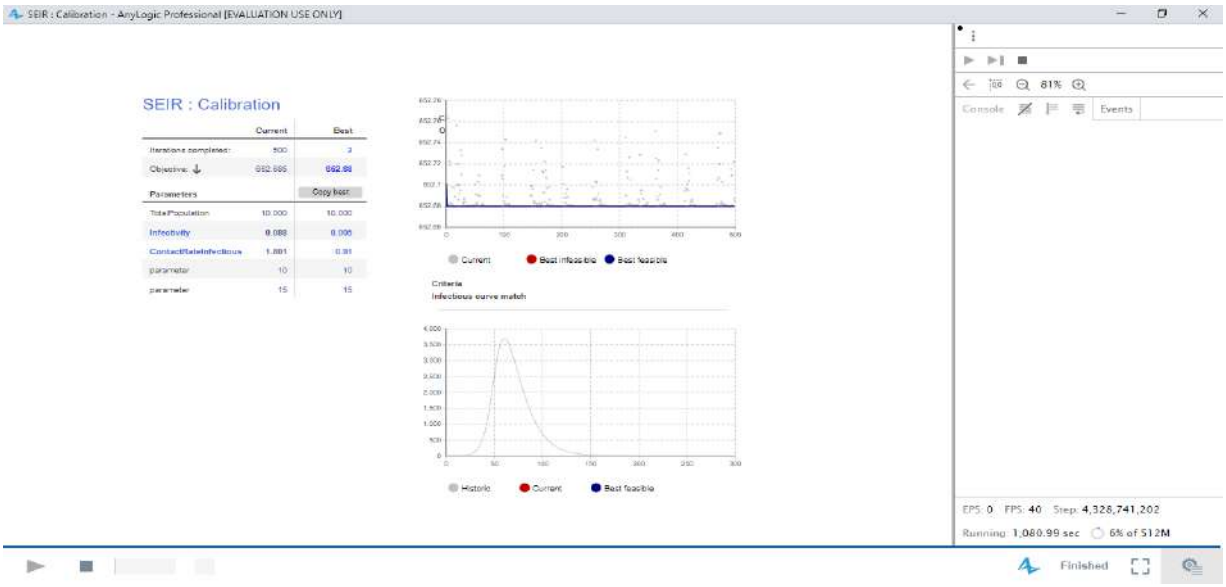


The image below shows the experiment's properties. Its objective is to minimize the difference between the model output and historical data.



Open the calibration experiment properties' Advanced section and then clear the Allow parallel evaluations checkbox.

Run the calibration experiment by either right-clicking Calibration in the Projects view and then clicking Run, or by selecting SEIR / Calibration from the list of experiments in the Run toolbar menu



After the calibration is complete, you can copy the best fitting parameter values by clicking the experiment window's copy button and then paste them into the simulation experiment by clicking the Paste from clipboard button that you'll find on the Simulation experiment's properties page.

After you've pasted the parameter values into the experiment, you can then run the Simulation with the newly - calibrated parameter values.

PRACTICAL NO: 05

Aim: Design and develop a discrete-event model that will simulate process by

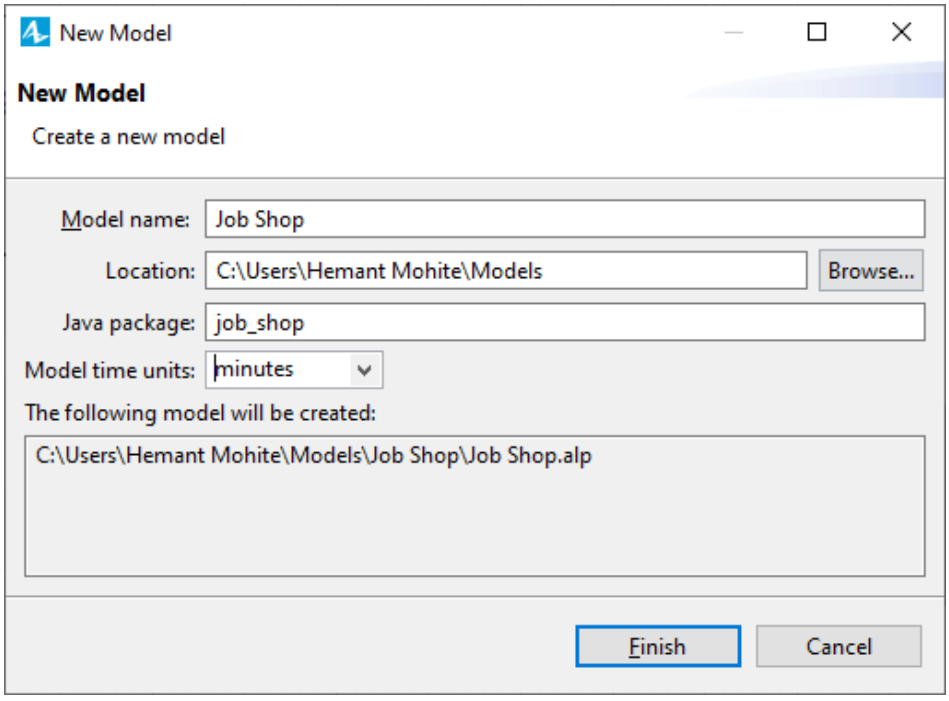
- Creating a simple model
- Adding resources
- Creating 3D animation
- Modelling delivery

[Use a case situation like a company's manufacturing and shipping].

Code:

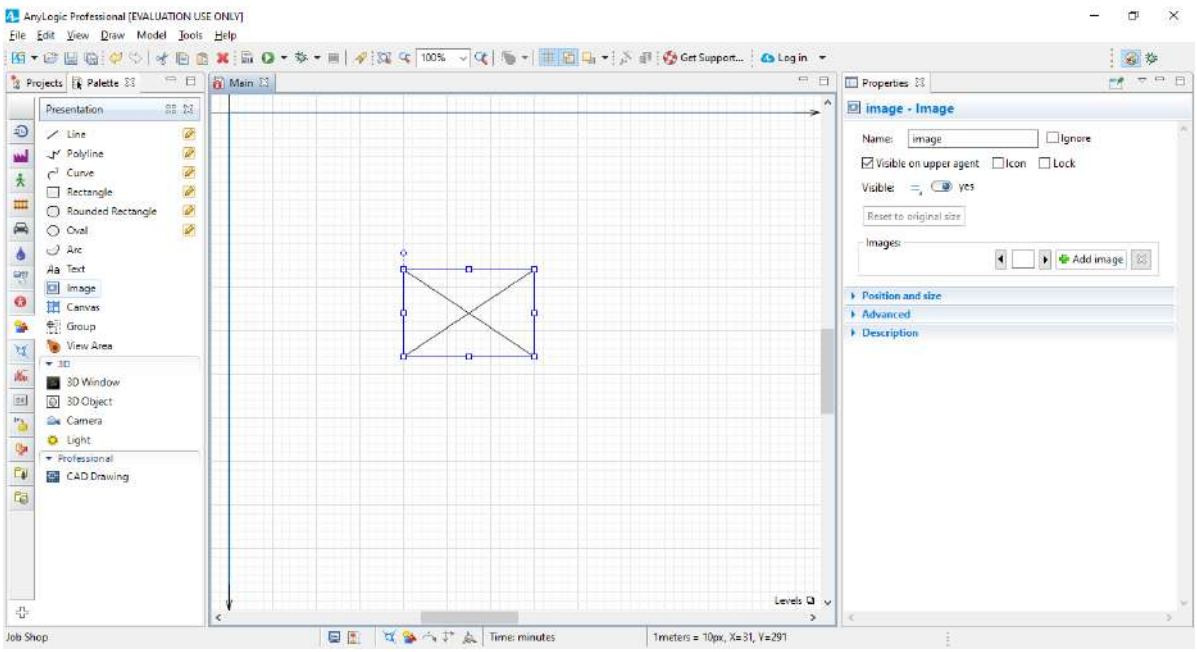
Creating Simple Model

Create a new model. In the New Model wizard, set the Model name: Job Shop, and Model time units: minutes.



Open the Presentation palette. The palette has several shapes that you can use to draw model animation, including a rectangle, a line, an oval, a polyline and a curve.

On the Presentation palette, select the Image shape and then drag it onto the Main diagram. You can use the Image shape to add images in several graphic formats -- including PNG, JPEG, GIF, and BMP – to your presentation.

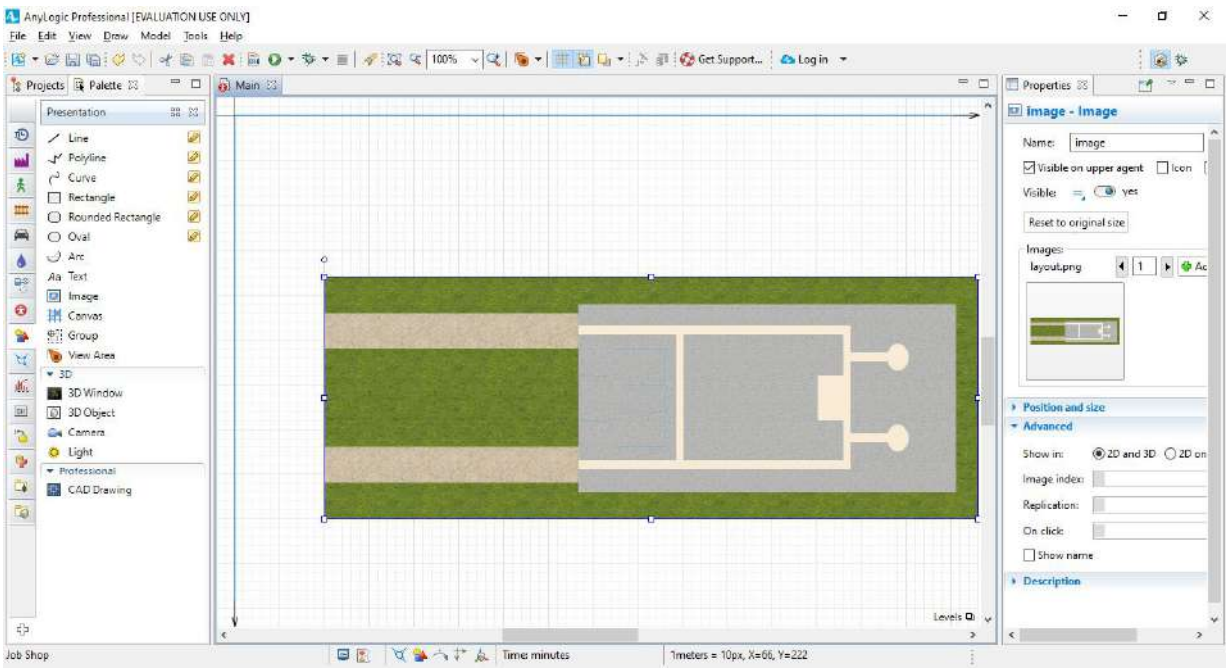


You'll see the dialog box that prompts you to choose the image file the shape will display.

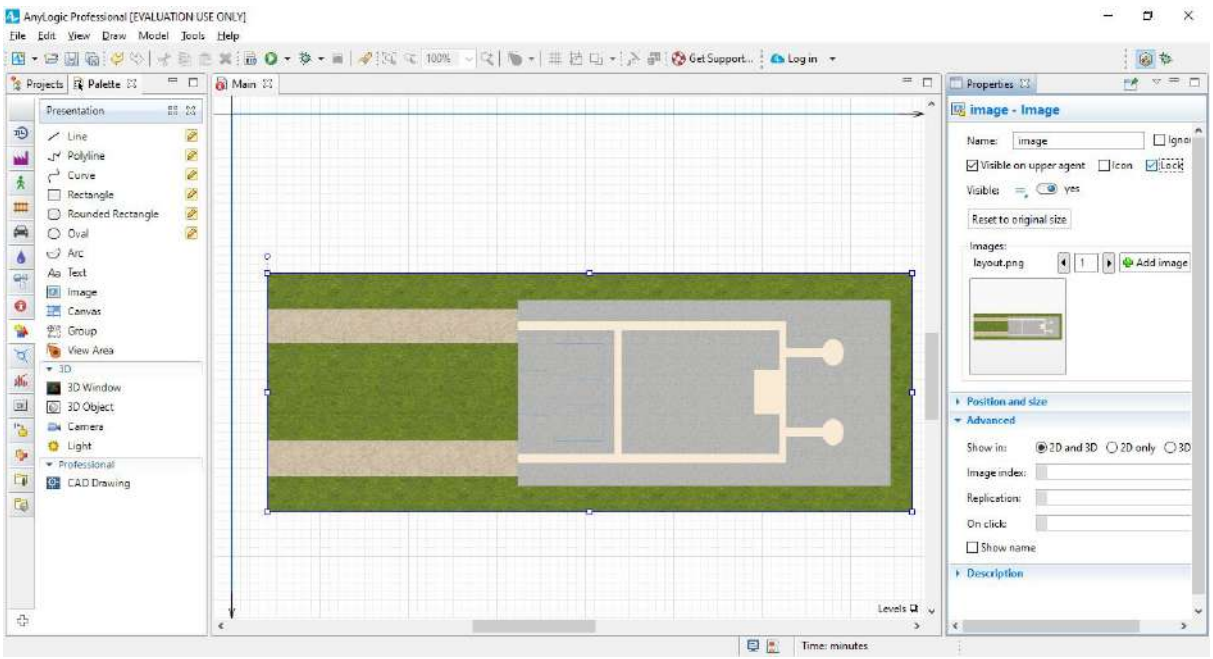
Browse to the following location and then select the layout.png image:

AnyLogic folder/resources/AnyLogic in 3 days/Job Shop

After you select the layout.png image, our diagram of the Main agent type should look like the following image:

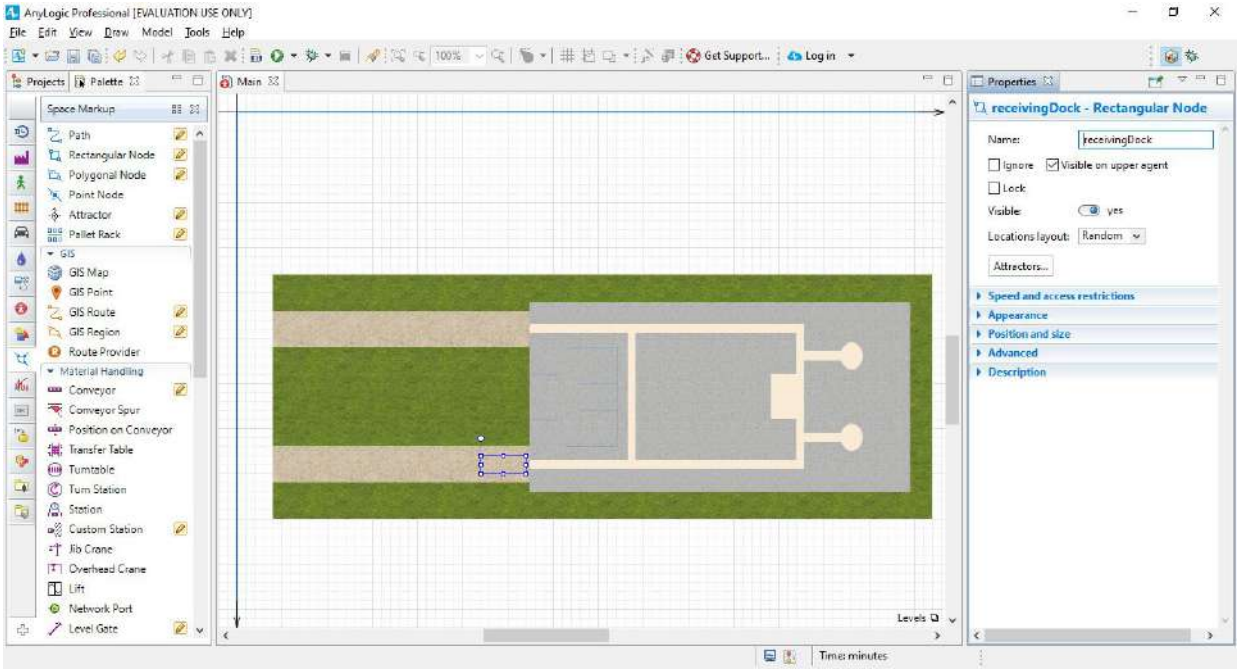


Select the image in the graphical editor. In the Properties view, select the Lock checkbox to lock the image.

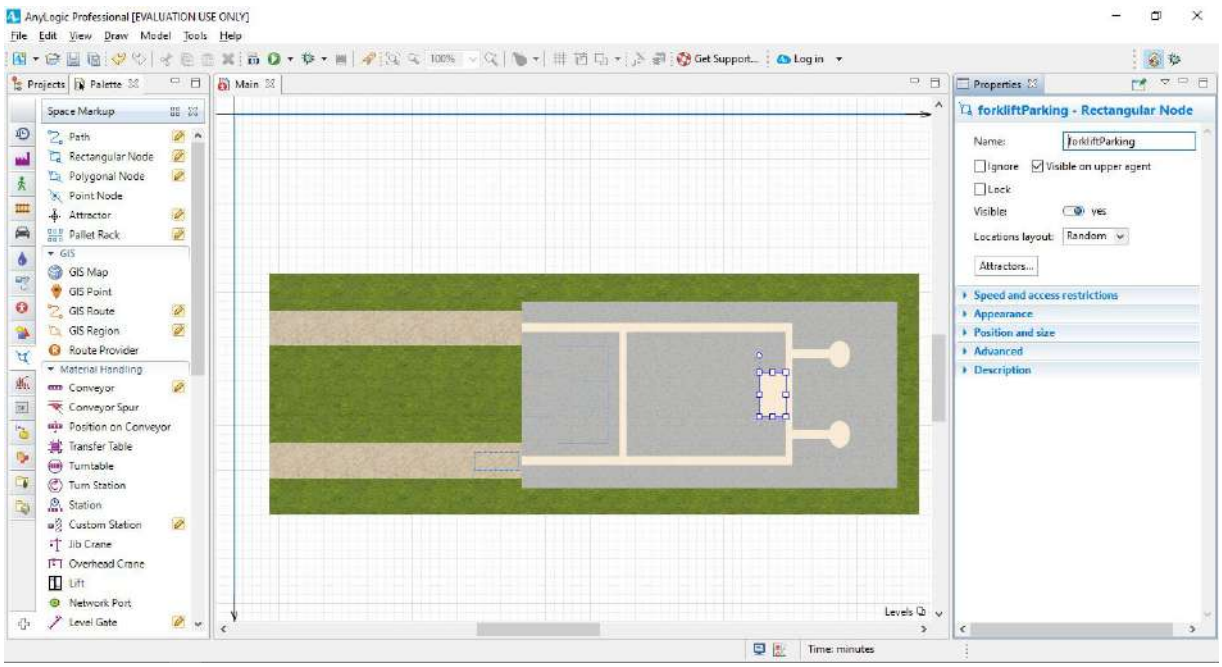


Open the Space Markup palette and drag the Rectangular Node element on to the Main diagram. Resize the node. The node should look as in the figure below.

Name the created node receivingDock.



Draw a node to define the location where the model's agents will park forklift trucks once the trucks are idle or the agents no longer need them to complete a task. Use another Rectangular node to draw the parking area as shown in the figure below and then name this node forkliftParking.



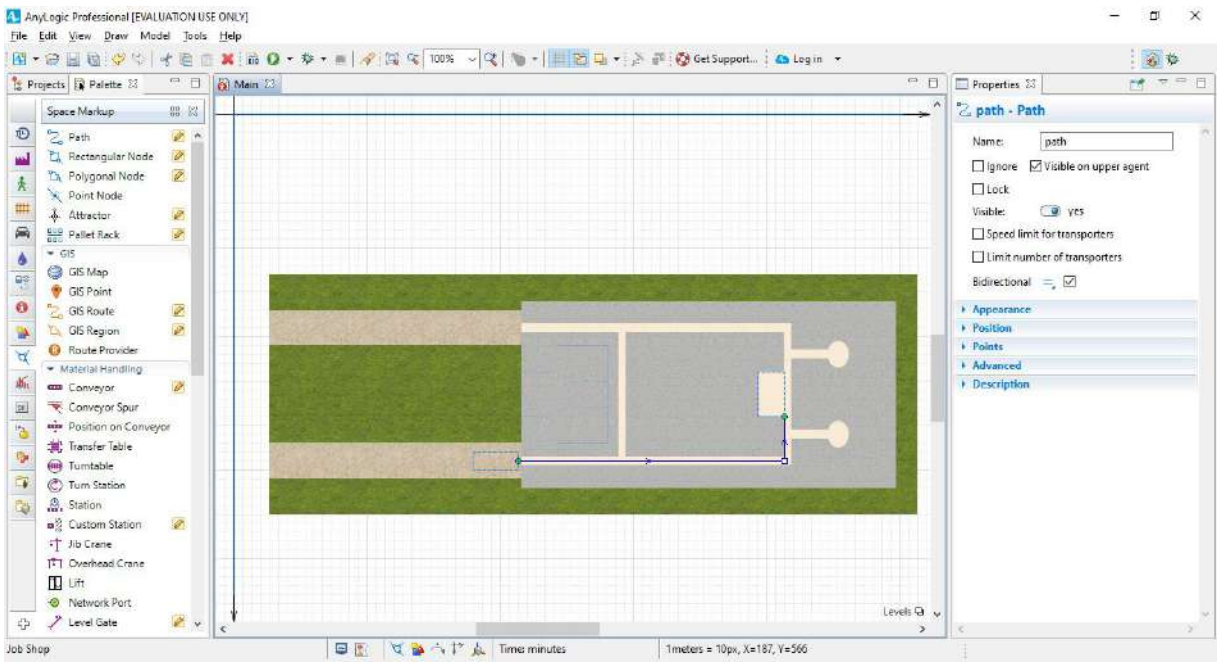
Let's draw a movement path to guide our model's forklift trucks.

Do the following to draw a movement path that will guide our model's forklift trucks:

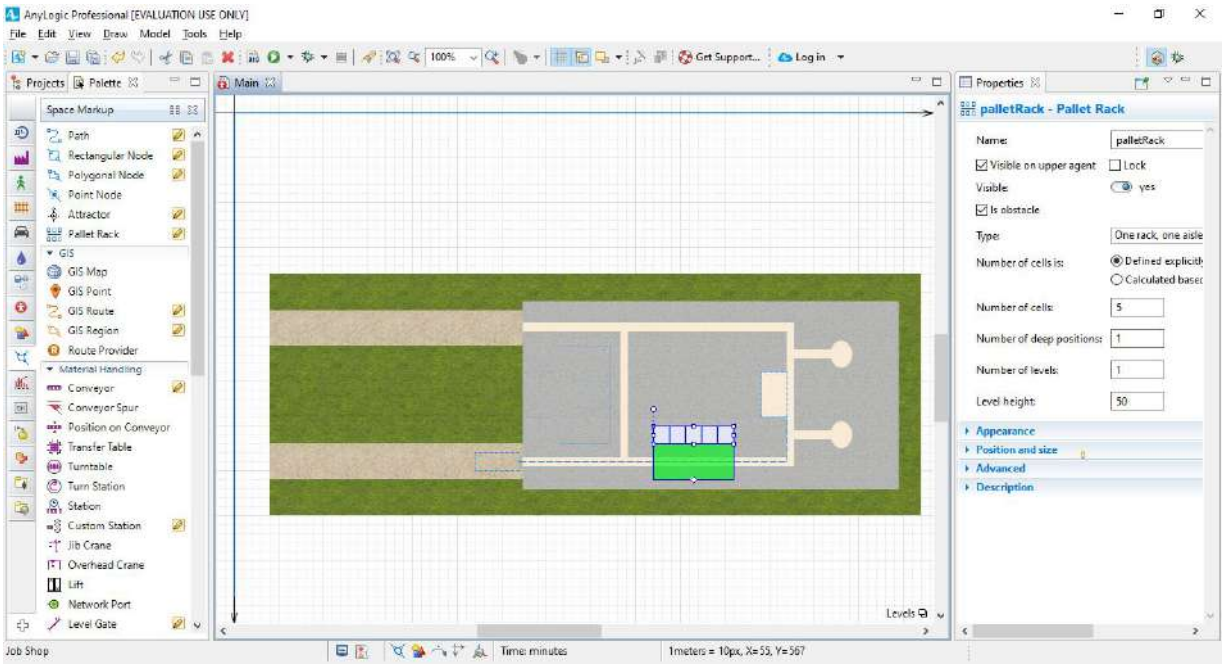
a. In the Space Markup palette, double-click the Path element to activate its drawing mode.

b. Draw the path as shown in the figure below by clicking the receivingDock border, clicking in the diagram to add the path's turning point, and then clicking the forkliftParking node's border.

If you've successfully connected the nodes, the path's connection points will display cyan highlights each time you select the path.



Define your model's warehouse storage by dragging the Pallet Rack element from the Space Markup palette on to the layout and placing its aisle on the path. A correctly - placed pallet rack will display a green highlight that shows it is connected to the network.



In the pallet rack's Properties area, do the following:

a. Set Type to: two racks, one aisle

b. Number of cells:10

c. Level height: 10

In the Position and size section:

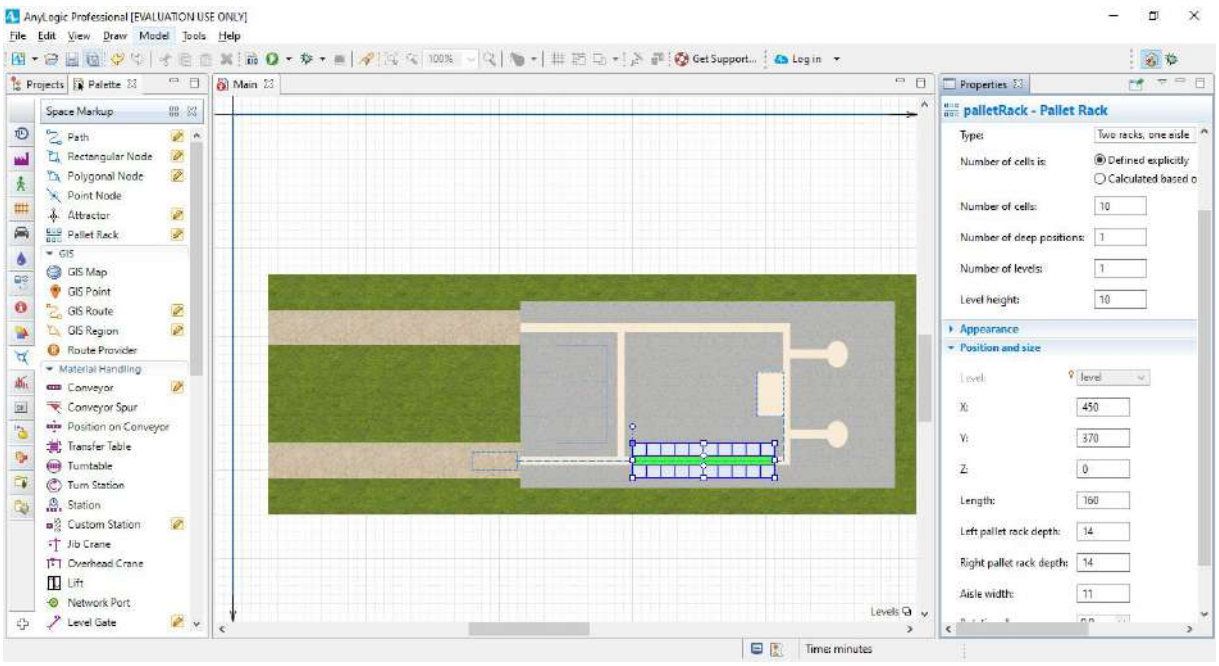
d. Length: 160

e. Left pallet rack depth: 14

f. Right pallet rack depth: 14

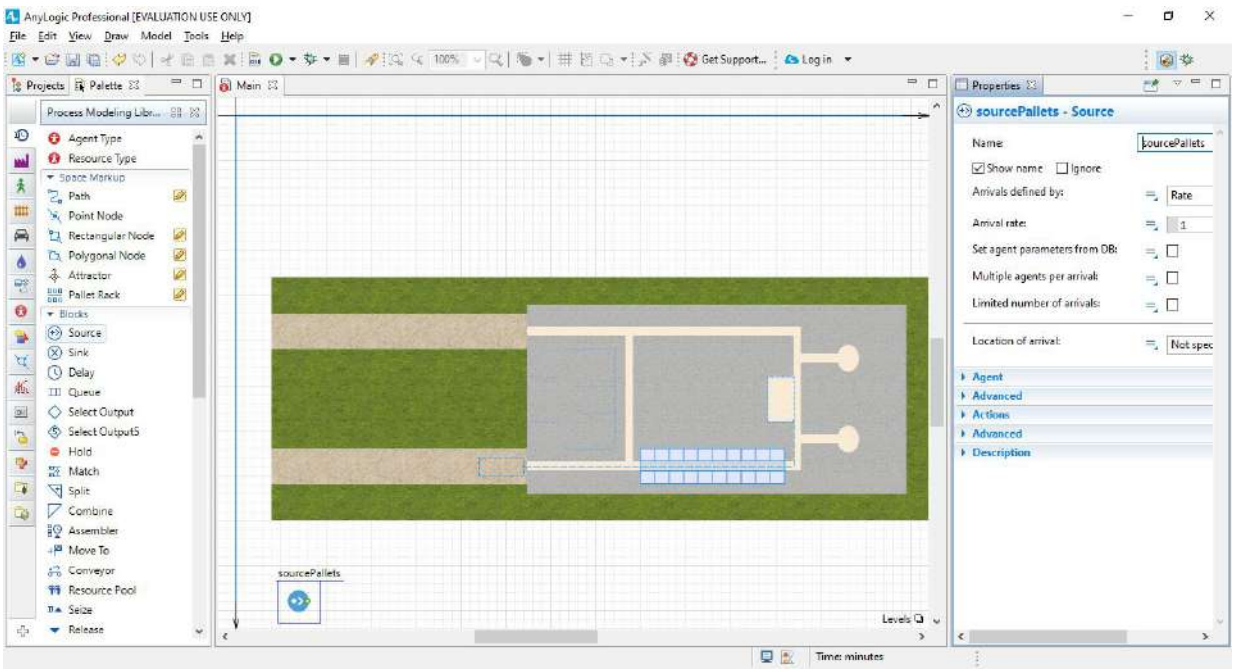
g. Aisle width: 1113.

After you've completed these changes, the pallet rack should resemble the pallet rack shown in the figure below. If necessary, move the pallet rack so that its center aisle lies on the path. Make sure the pallet rack is connected to the network by clicking it twice to select it. Your first click will select the entire network, and the second will select the pallet rack. The pallet rack should display a green highlight that shows it is connected to the network.



Drag the Source element from the Process Modeling Library palette on to the graphical diagram and name the block sourcePallets.

While the Source block usually acts as a process starting point, our model will use it to generate pallets.



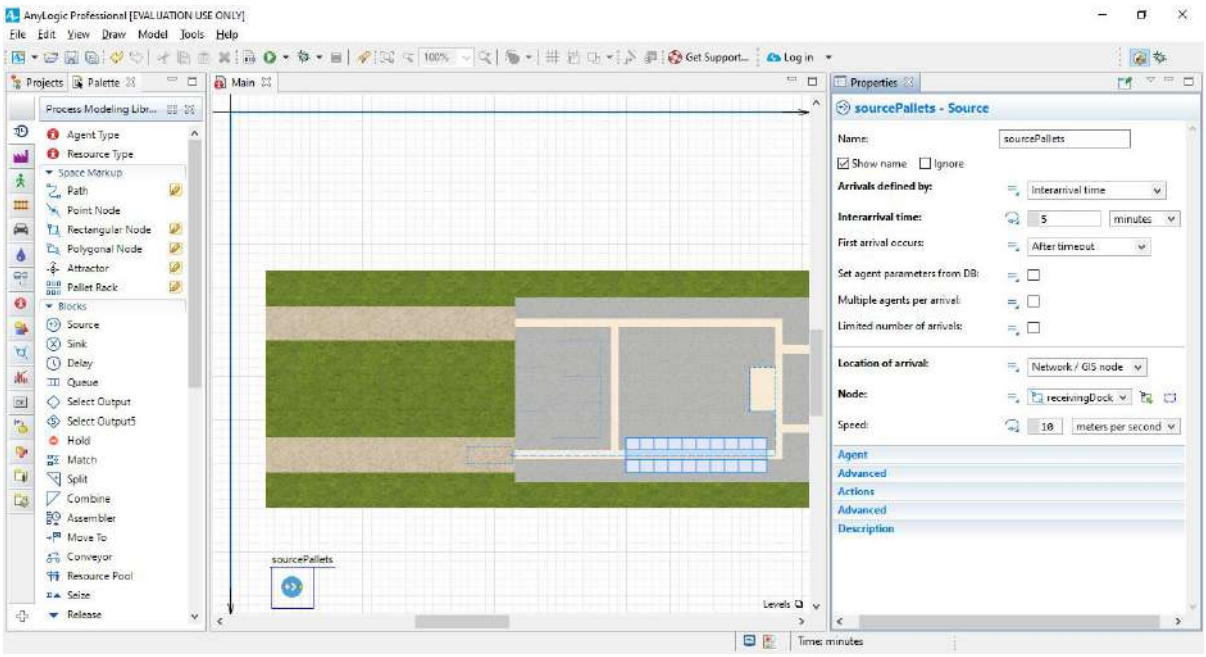
In the sourcePallets block's Properties area, do the following to ensure the model's pallets arrive every five minutes and appear in the receivingDock node.

a. In the Arrivals defined by area, click Interarrival time.

b. In the Interarrival time box, type 5, and select minutes from the list on the right to have pallets arrive every five minutes.

c. In the Location of arrival area, click Network / GIS node in the list.

d. In the Node area, click receivingDock in the list.



Drag the RackStore block from the Process Modeling Library palette on to the diagram and place it near the sourcePallets block so they are automatically connected as shown in the diagram below.

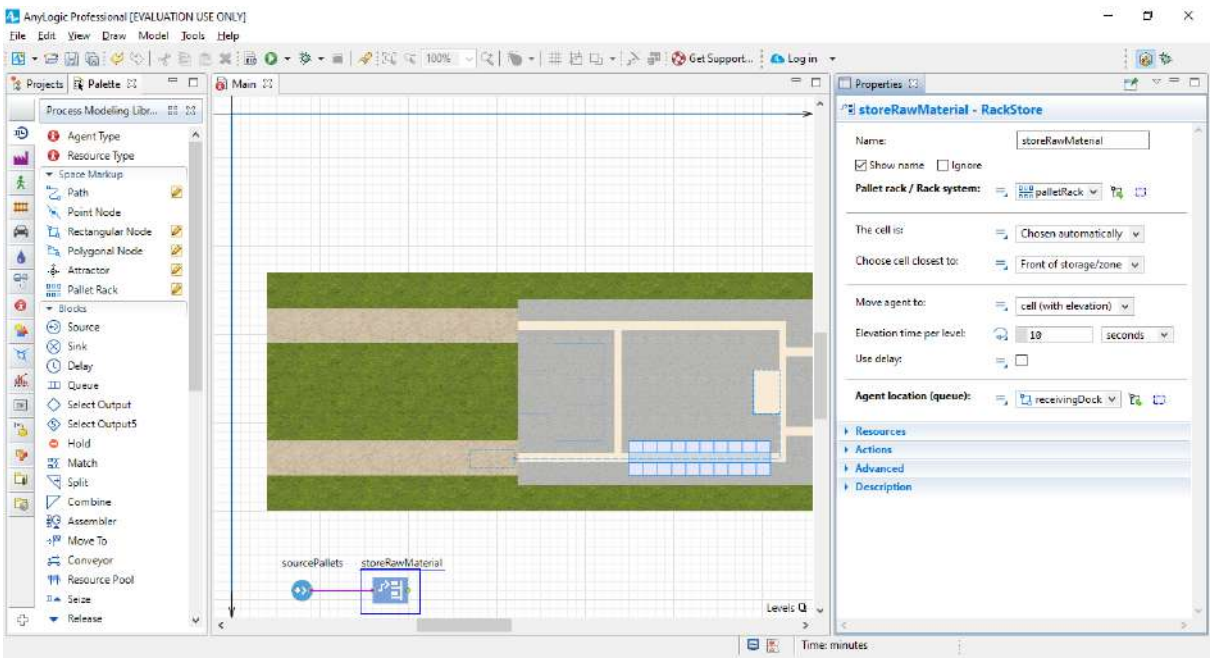
The RackStore block places pallets into a given pallet rack's cells.

In the rackStore block's Properties area, do the following:

a. In the Name box, type storeRawMaterial.

b. In the Pallet rack / Rack system list, click palletRack.

c. In the Agent location (queue)list, click receivingDock to specify the location where agents wait to be stored.

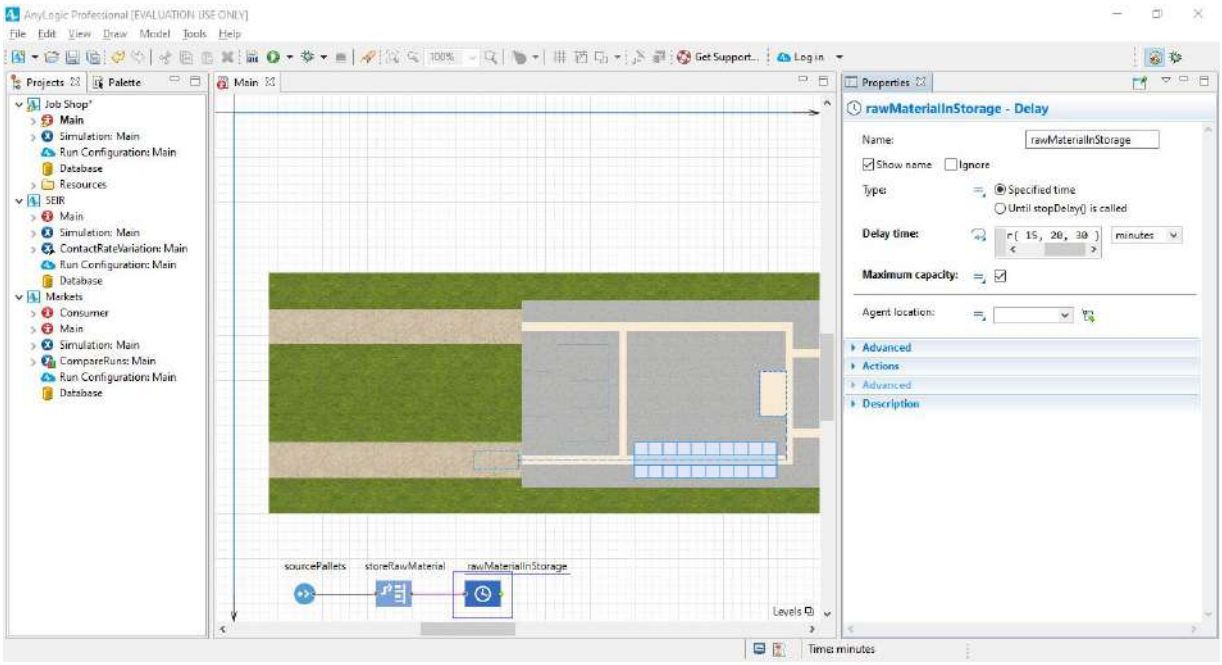


Add a Delay block to simulate how pallets wait in the rack and then name the block rawMaterialInStorage.

In the rawMaterialInStorageblock's Properties area, do the following:

a. In the Delay timebox, type triangular(15, 20, 30) and select minutes from the list.

b. Select the Maximum capacity checkbox to ensure agents will not get stuck as they wait to be picked up from storage.

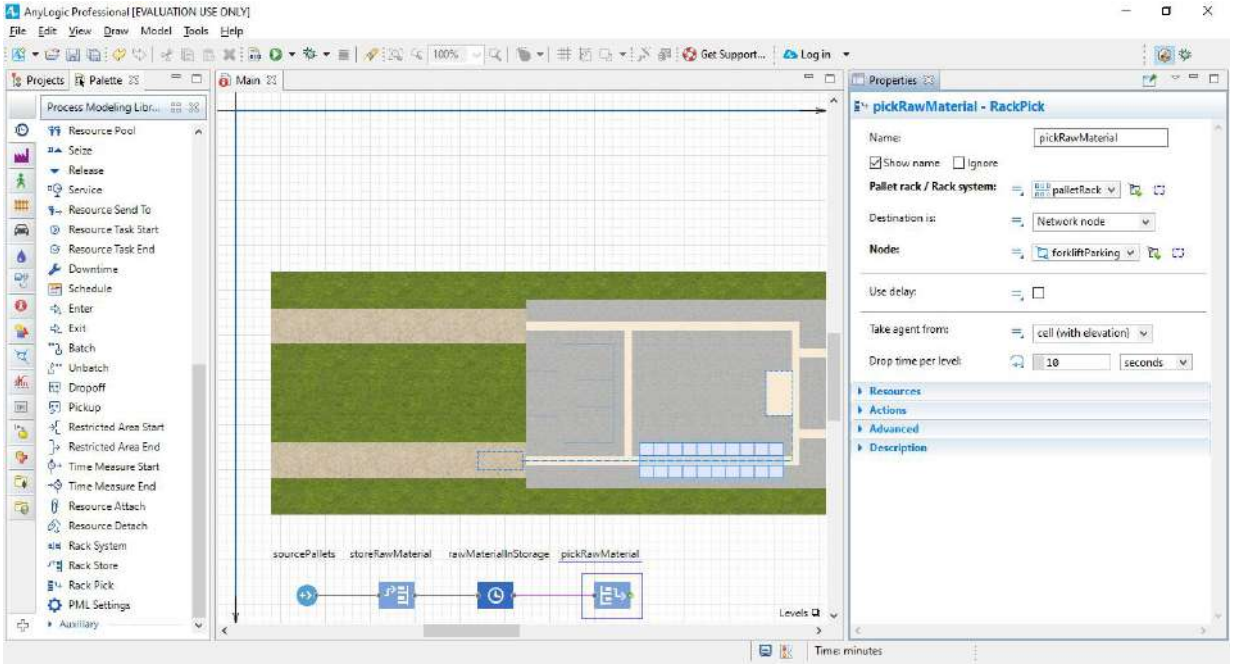


Add a RackPick block, connect it to the flowchart, and then name it pickRawMaterial.

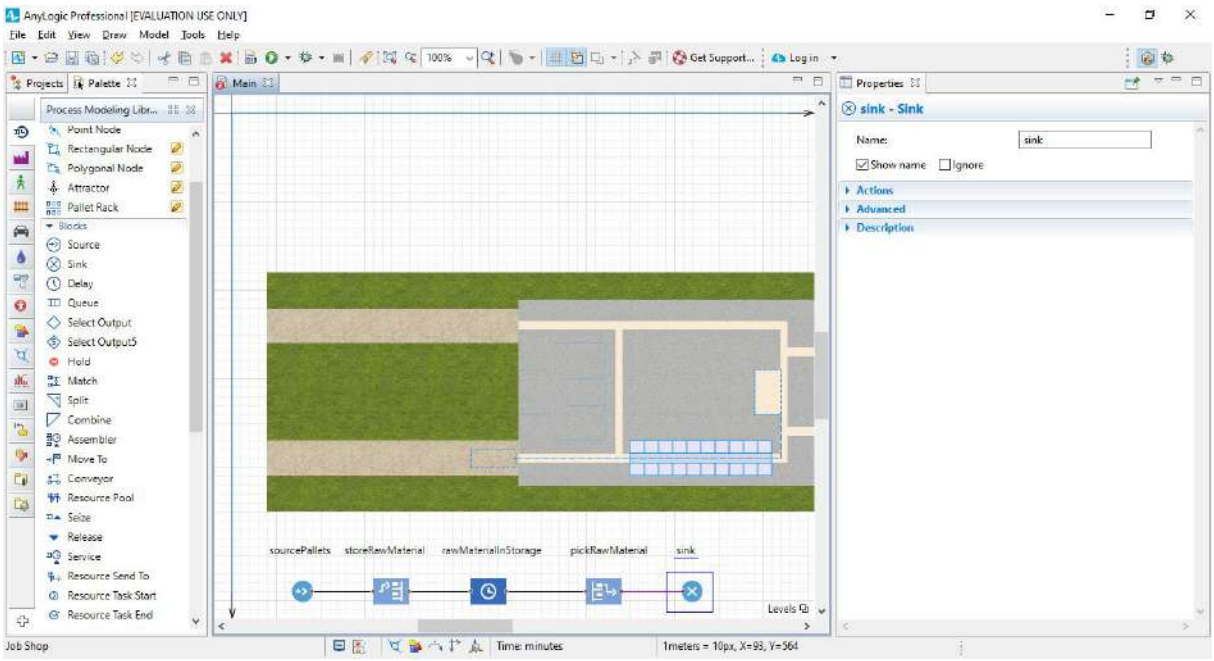
In the pickRawMaterial block's Properties area, do the following:

a. In the Pallet rack / Rack system list, click palletRack to select the pallet rack that will provide pallets to agents.

b. In the Node list, click forkliftParking to specify where the agents should park forklift trucks.



Add a Sink block. The Sink block disposes agents and is usually a flowchart's end point.



We've finished building this simple model, and you can now run it and observe its behavior. Run the model (Job Shop / Simulation experiment).


Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
sourcePallets
storeRawMaterial
rawMaterialInStorage
pickRawMaterial
sink
x1
Running


Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
sourcePallets
storeRawMaterial
rawMaterialInStorage
pickRawMaterial
sink
x1
Running

Adding Resources

Resources

Resources are objects that agents use to perform a given action. An agent must obtain the resource, perform the action, and then release the resource.

Some examples of resources include:

- A hospital model's doctors, nurses, equipment, and wheelchairs

- A supply chain model's vehicles and containers

- A warehouse model's forklift trucks and workers

There are three types of resources: static, moving, and portable.

- Static resources are bound to a specific location, and they cannot move or be moved.

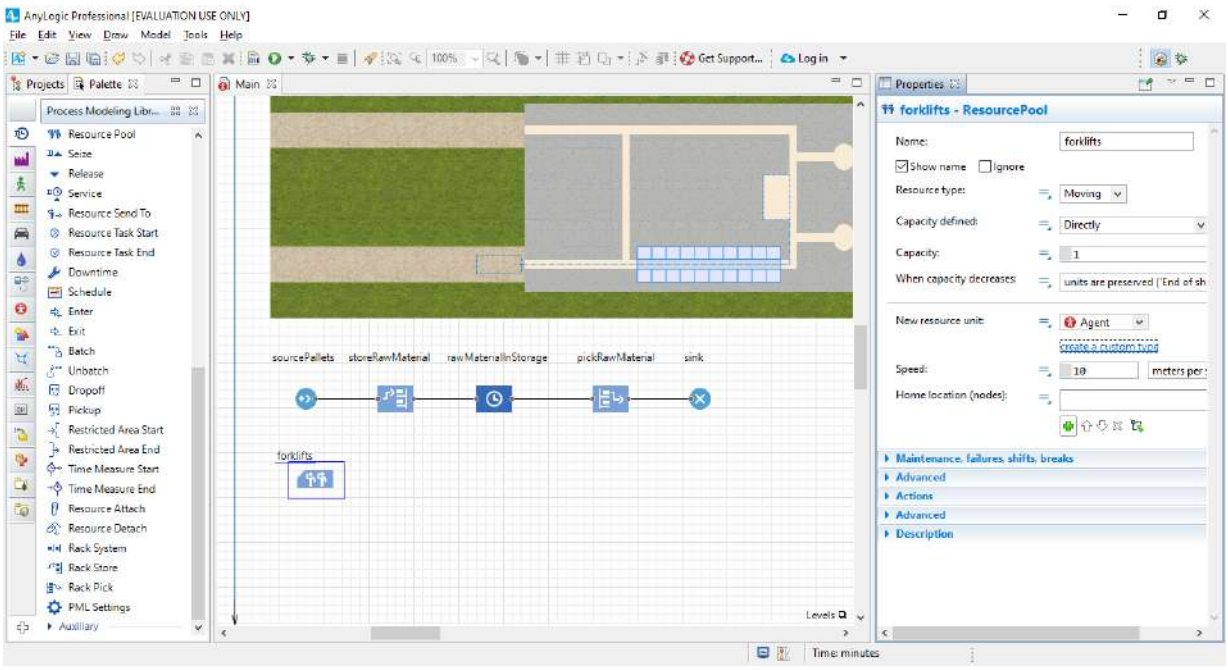
- Moving resources can move independently.

- Portable resources can be moved by agents or by moving resources.

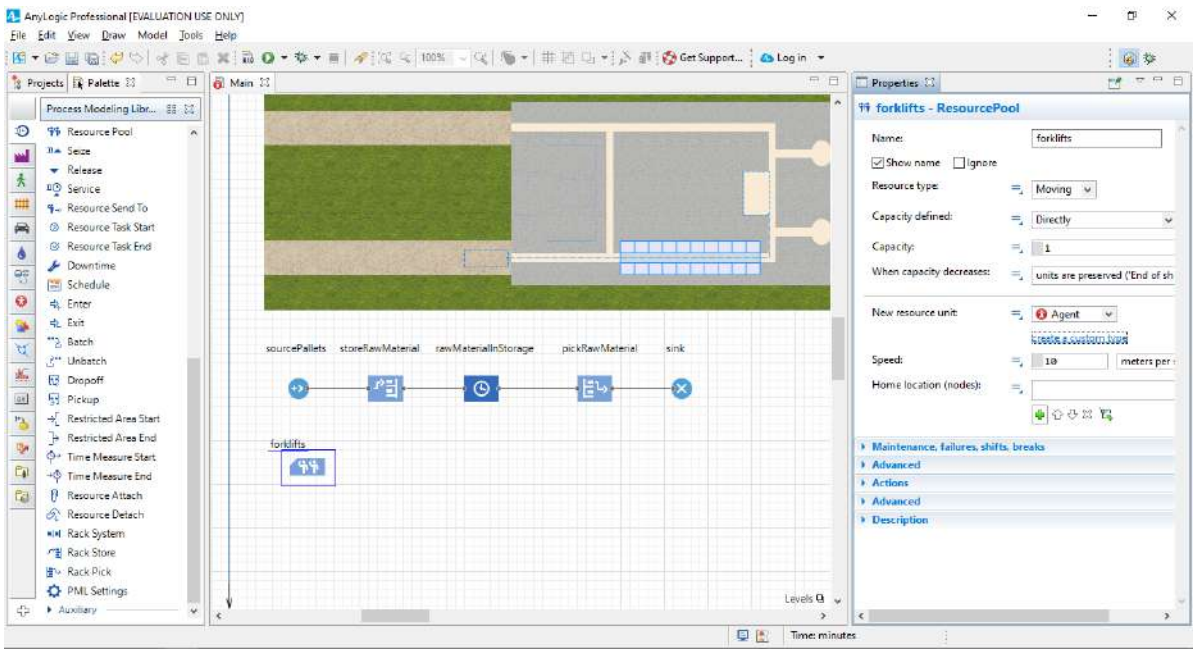
Our model's resources are the forklift trucks that move pallets from the unloading zone to a pallet rack and then deliver pallets from the rack to the production zone.

On the Process Modeling Library palette, drag the ResourcePool block on to the Main diagram. You do not have to connect the block to the flowchart.

Name the block forklifts.



In the forklifts block's Properties area, click the button create a custom type. This way we create a new type of a resource.



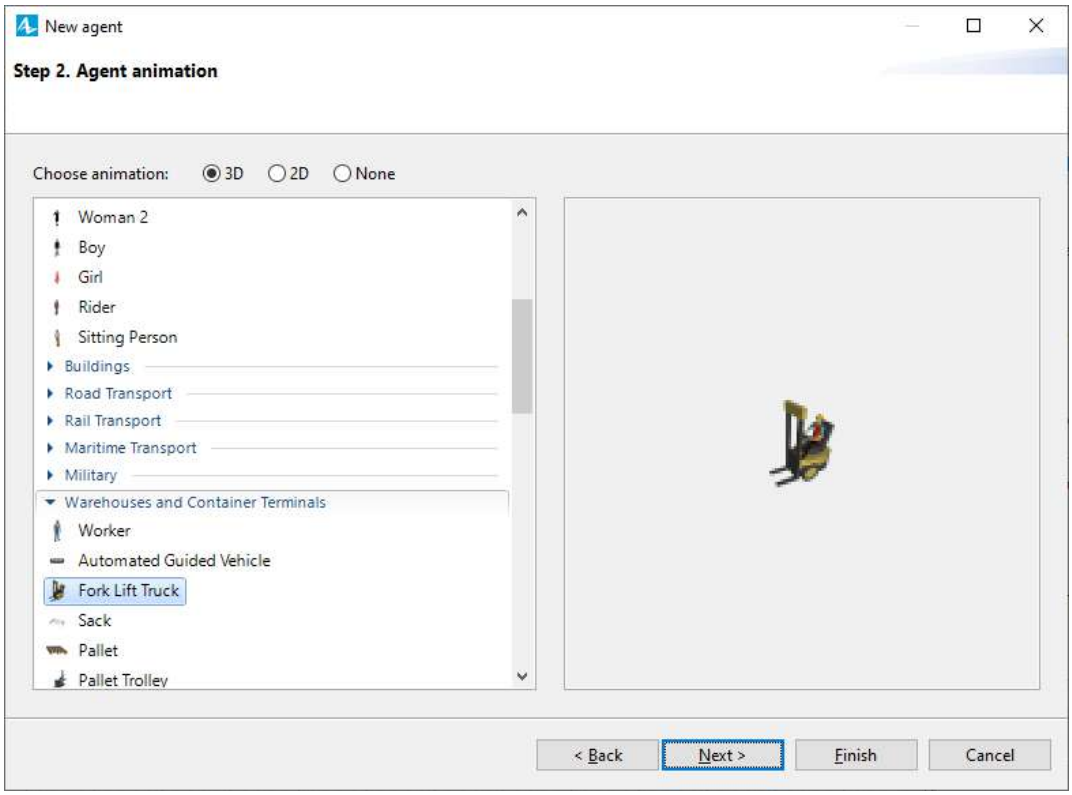
In the New agent wizard:

a. In The name of new type box, type ForkliftTruck.

b. In the list in the left part of the wizard, expand the Warehouses and Container Terminals area, and then click the 3D animation figure Fork Lift Truck.

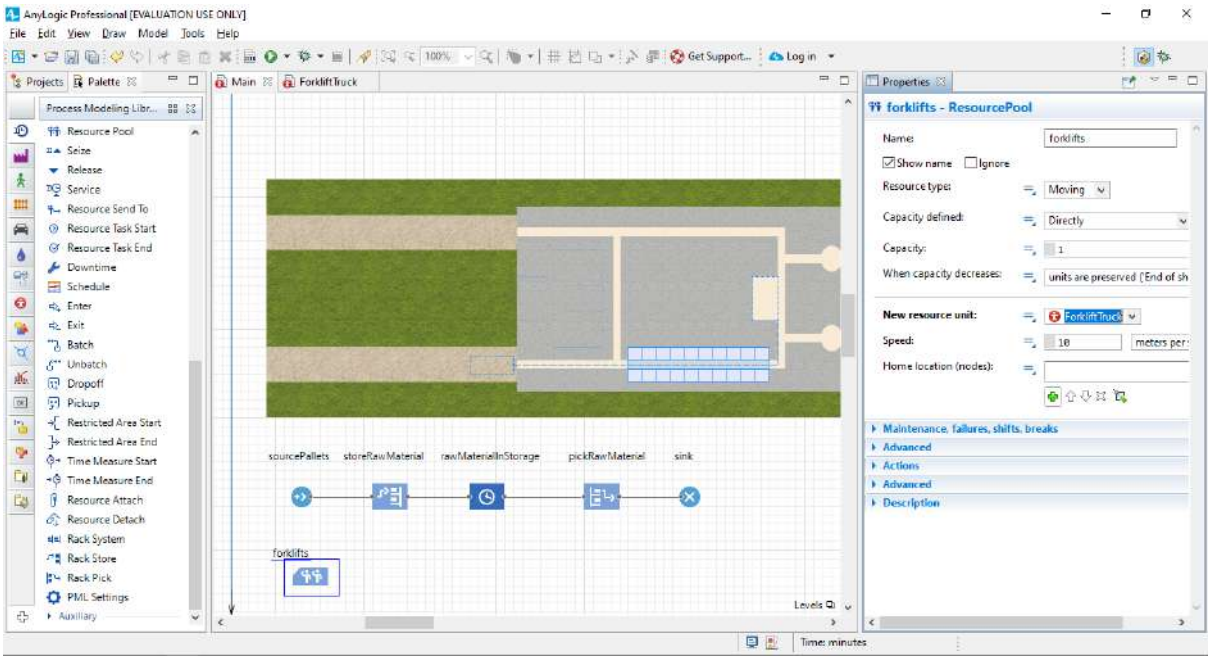
c. Click Finish.

The ForkliftTruck agent type diagram will open and display the animation shape you selected in the wizard.



Click the Main tab to open the Main diagram.

You'll see the ForkliftTruck resource type has been selected in the ResourcePool block's New resource unit parameter.

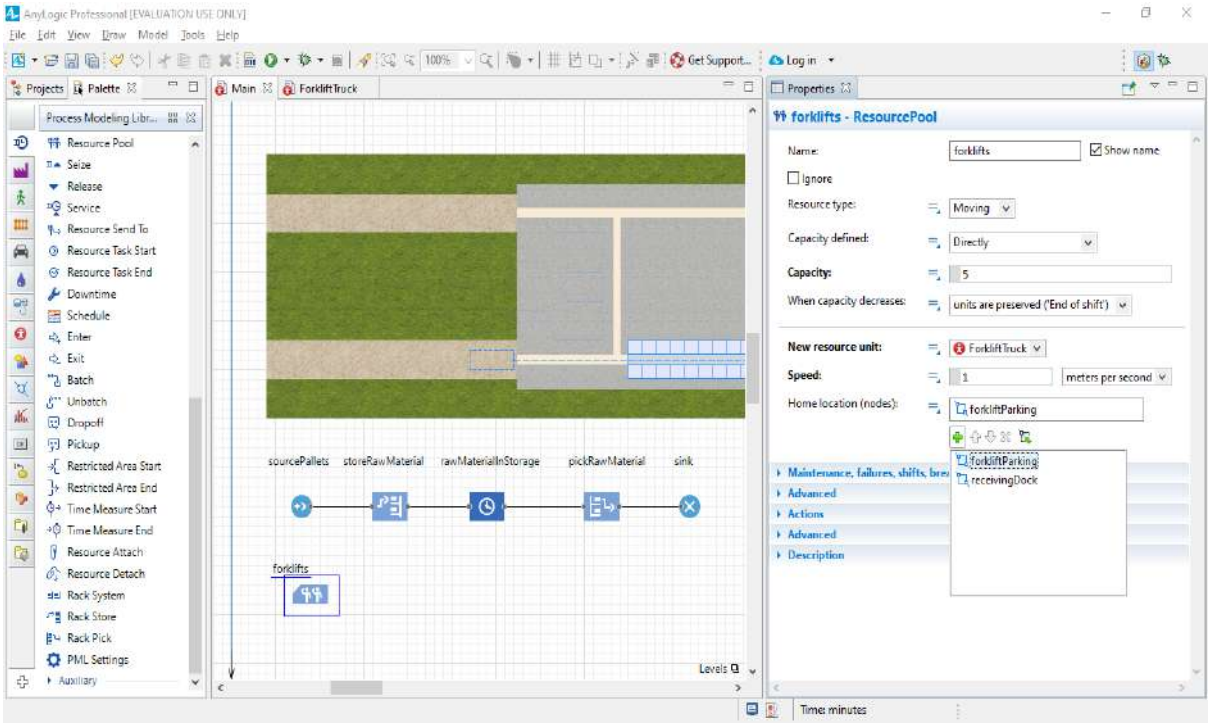


Modify the forklifts resource type's other parameters:

a. In the Capacity box, type 5 to set the number of forklift trucks in our model.

b. In the Speed box, type 1 and choose meters per second from the list on the right.

c. In the Home location (nodes) area, select the forkliftParking node. Click the plus button and then click forkliftParking in the list of the model's nodes.



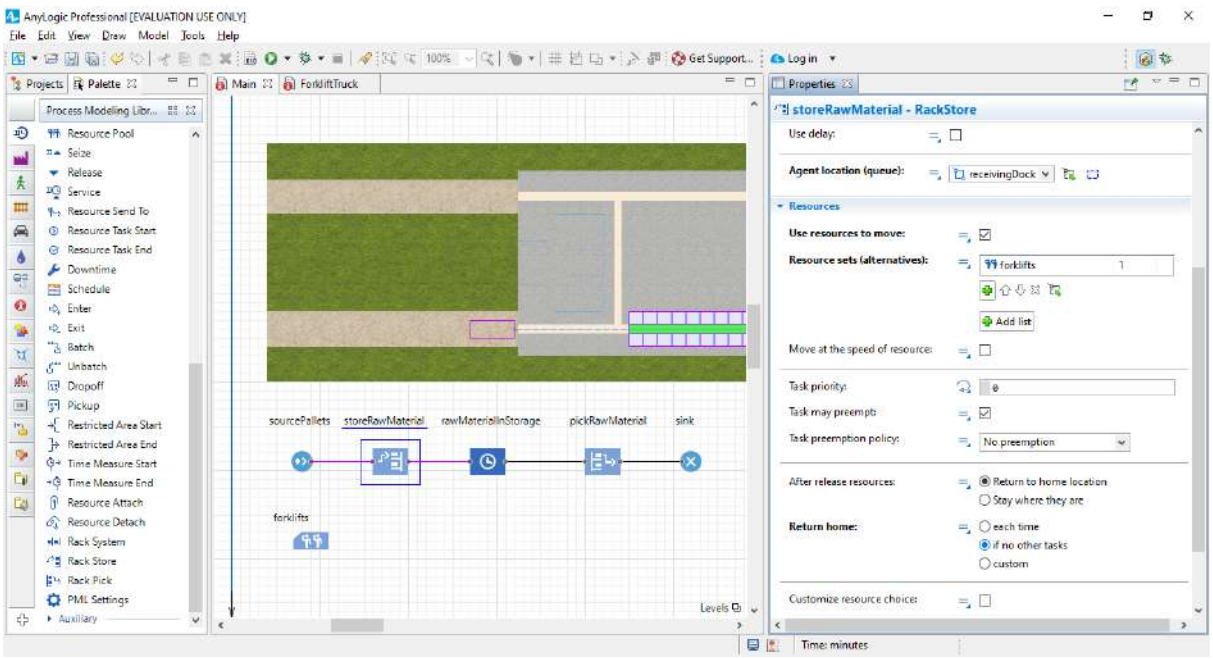
In the storeRawMaterial block's Properties area, do the following:

a. Click the arrow to expand the Resources area.

b. Select the Use resources to move check box.

c. In the Resource sets (alternatives)list, click forklifts to ensure the flowchart block uses the selected resources -- in our case, the forklift trucks -- to move the agents.

d. In the Return home area, select if no other tasks to ensure the forklift trucks return to their home location after they complete their tasks.



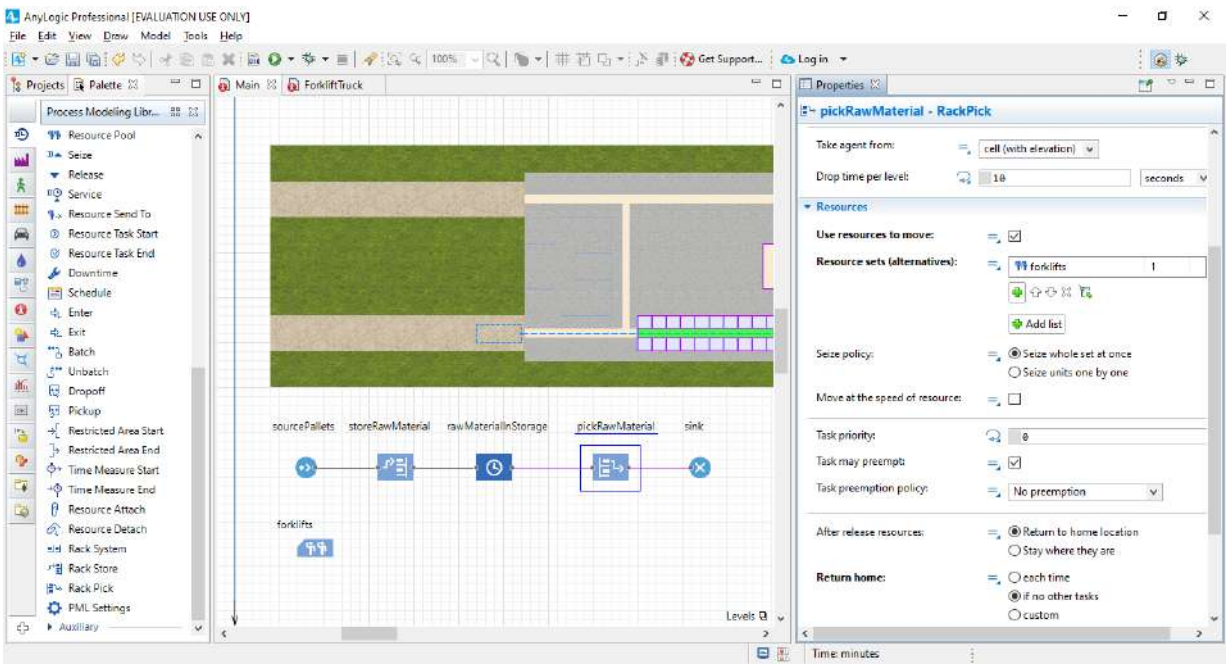
In the pickRawMaterial block's Properties area, do the following:

a. Click the arrow to expand the Resources area.

b. Select the Use resources to move check box.

c. In the Resource sets (alternatives) list, click forklifts to ensure the flowchart block uses the forklift trucks to move the agents.

d. In the Return home area, click if no other tasks to ensure the forklift trucks return to their home location after they complete their tasks.



Run the model.

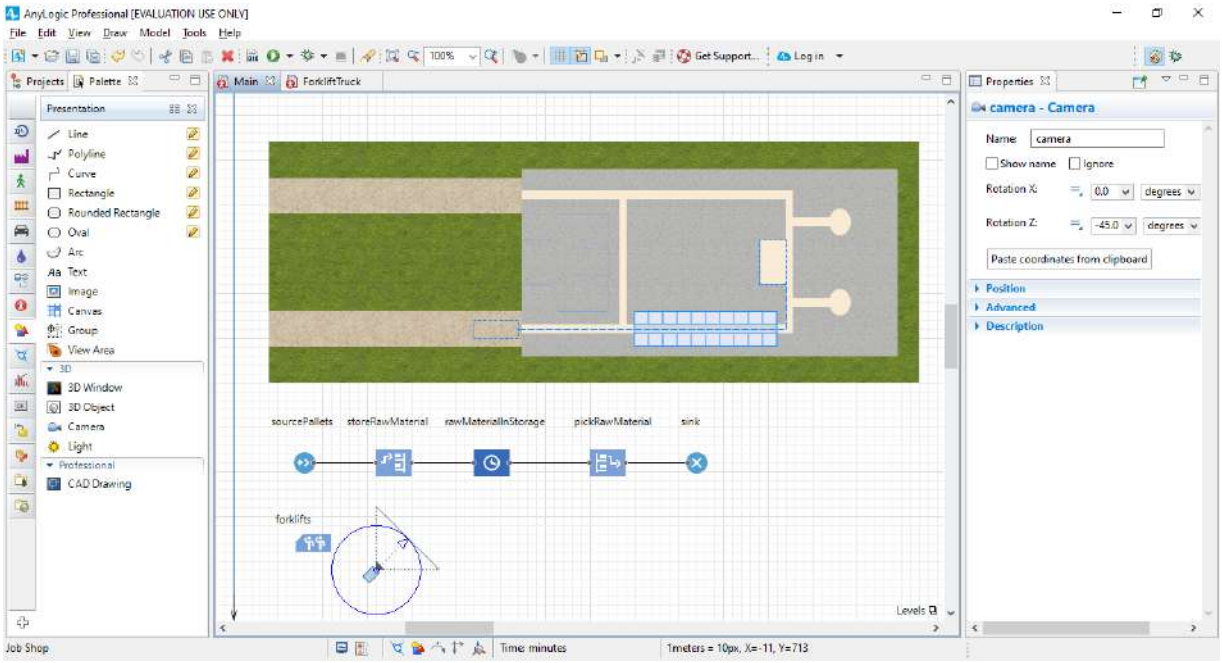

Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
sourcePallets
storeRawMaterial
rawMaterialsInStorage
pickRawMaterial
sink
forklifts
x1
Running


Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
sourcePallets
storeRawMaterial
rawMaterialsInStorage
pickRawMaterial
sink
forklifts
x1
Running

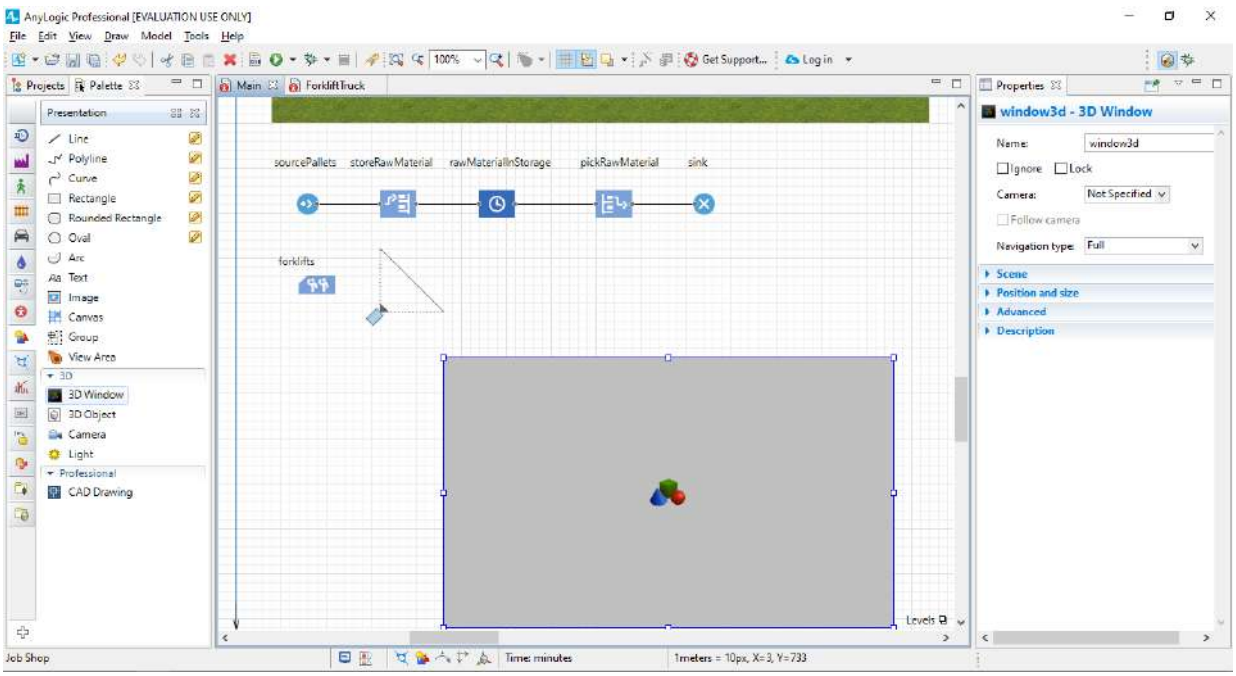

Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
sourcePallets
storeRawMaterial
rawMaterialsInStorage
pickRawMaterial
sink
forklifts
x1
Running

Creating 3D animation

On the Presentation palette, drag the Camera object on to the Main diagram so it faces the job shop layout.

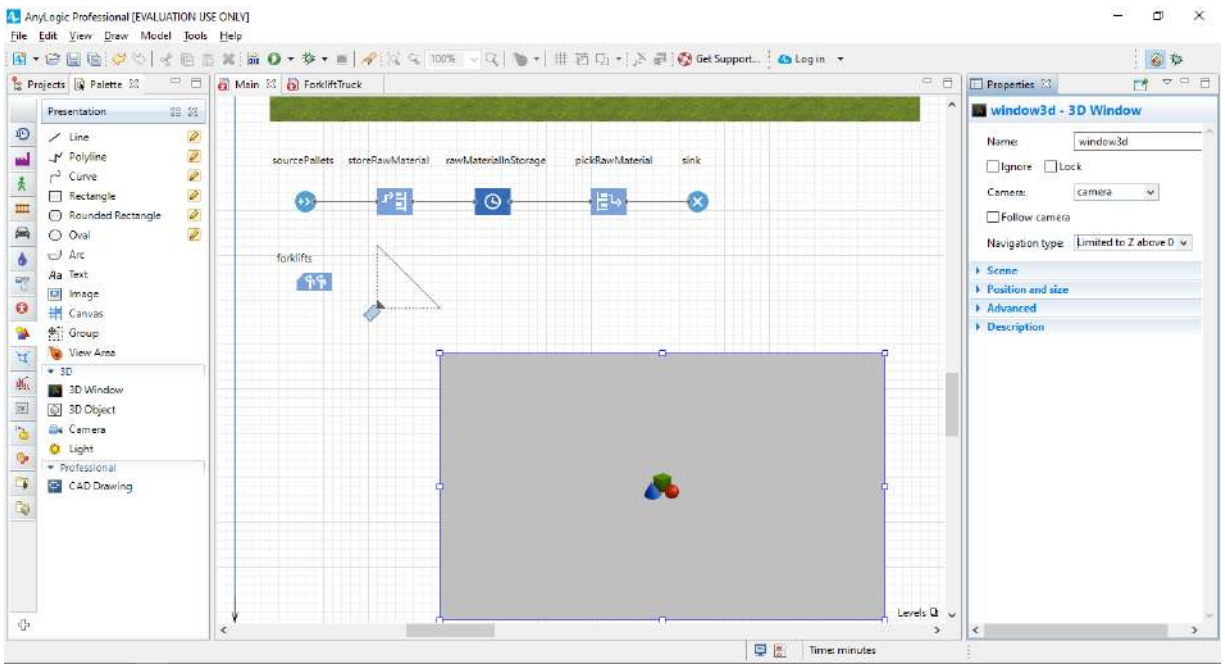


Drag the 3D Window object on to the Main diagram, and then place it below the process flowchart.



Let the camera “shoot” the picture for the 3D window. In the 3D window’s Properties area, click camera in the Camera list.

Prevent the camera from shooting the picture from under the floor by selecting the option Limited to Z above 0 from the Navigation type list.



Run the model.

To switch to this 3D view, click the toolbar's Navigate to view area... button and then click [window3d].


Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
1,128.79 min 09/16/2020 18:48
Console
Events
2 events scheduled by:
T left
1.02 min/sec
Running: 1,217.51 sec
6% of 512M
Running


Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
25.35 min 09/16/2020 00:25
Console
Events
0.76 min/sec
Running: 27.5 sec
10% of 512M
Running


Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
110.45 min 09/16/2020 01:50
Console
Events
0.99 min/sec
Running: 114.51 sec
11% of 512M
Running

Do one or more of the following to navigate in 3D at runtime:

To move the camera left, right, forward or backward, drag the mouse in the selected direction.

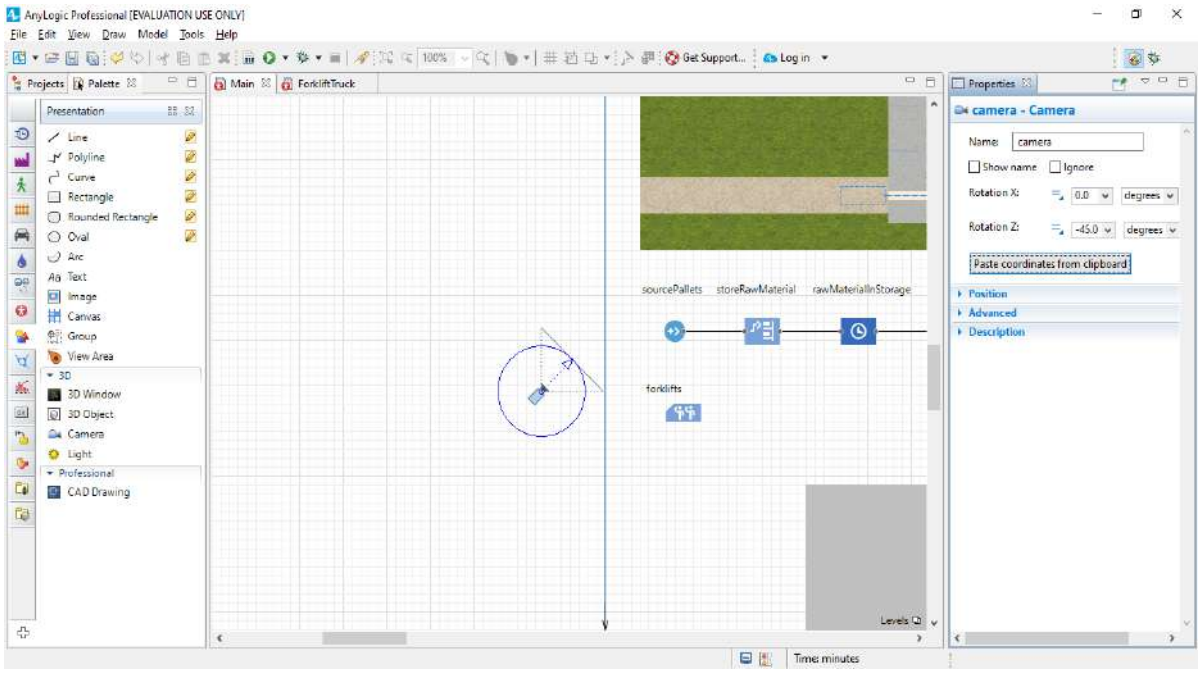
To move the camera closer to or further from the scene's center, rotate the mouse's wheel.

To rotate the scene relative to the camera, drag the mouse while you press and hold ALT and the left mouse button.

Choose the view you want to display at runtime, right-click inside the 3D scene, and then click Copy the camera's location.

Close the model's window.

On the camera's Properties area, apply the camera location you selected during the previous step by clicking Paste coordinates from clipboard.



Run the model to view the 3D view from the new camera position, and then close the model window.

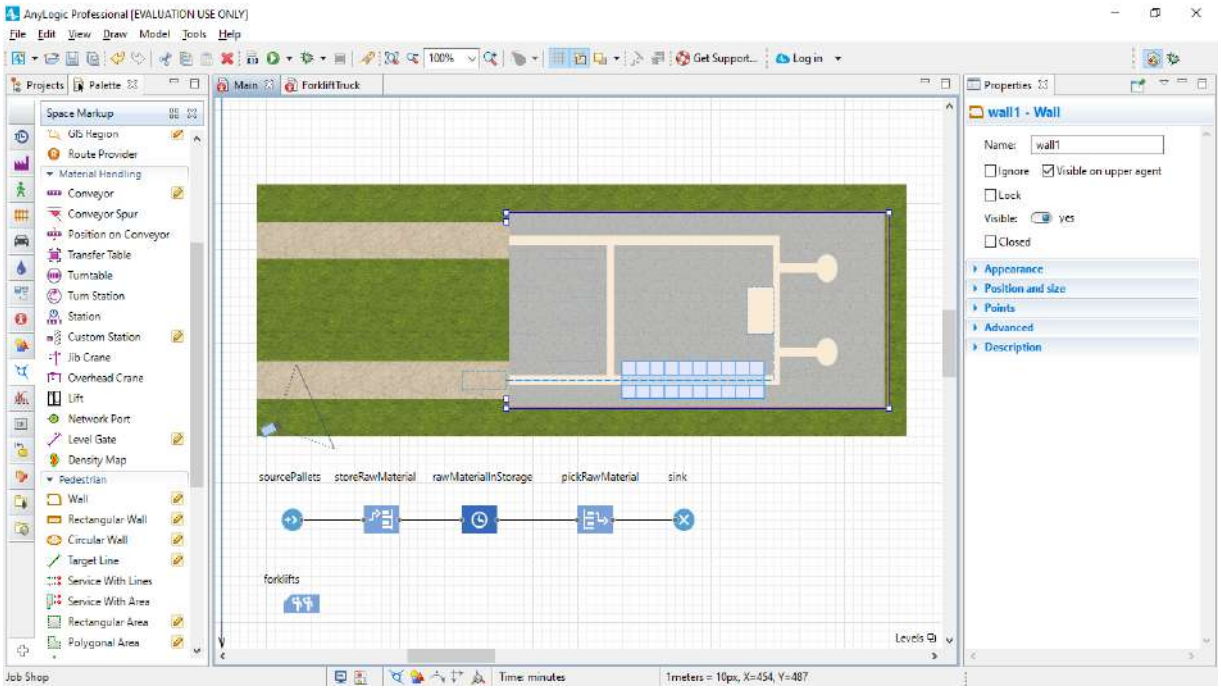
Expand the Space Markup palette's Pedestrian area and then double-click the Wall element's icon to enable AnyLogic's drawing mode.

Do the following to draw walls around the job shop layout's working area:

a. Click the position in the graphical editor where you want to start drawing the wall.

b. Move the pointer in any direction to draw a straight line, and then click at any point where you want to change direction.

c. Double-click at in the point where you want to stop drawing the wall.



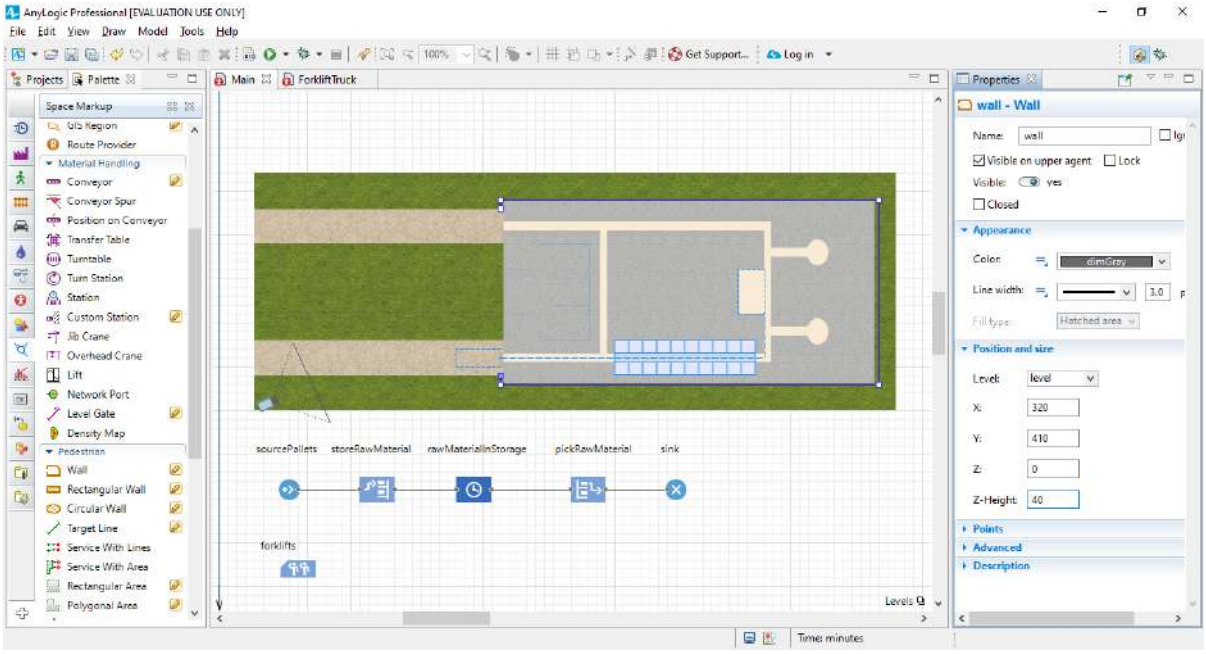
Do the following to change the wall's fill color and texture:

a. On the wall's Properties area, expand the Appearance section.

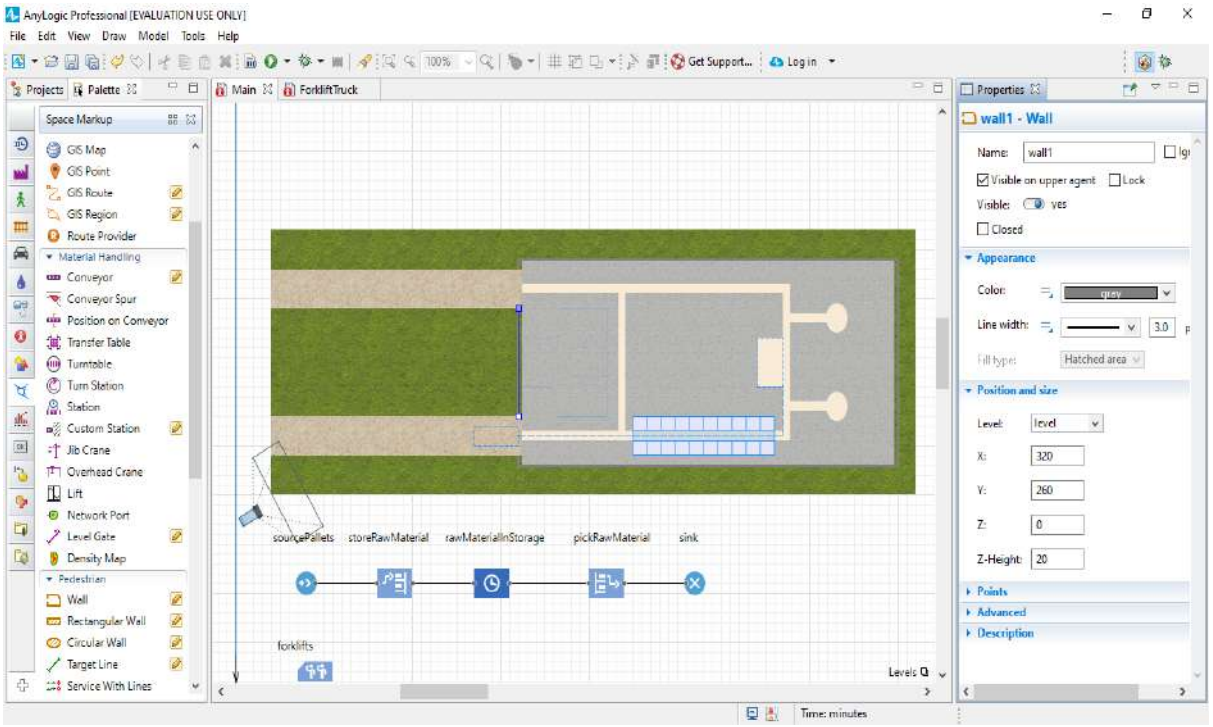
b. In the color menu, click Other colors.

c. In the Colors dialog, select the color that you want to apply to the wall from the palette or the spectrum.

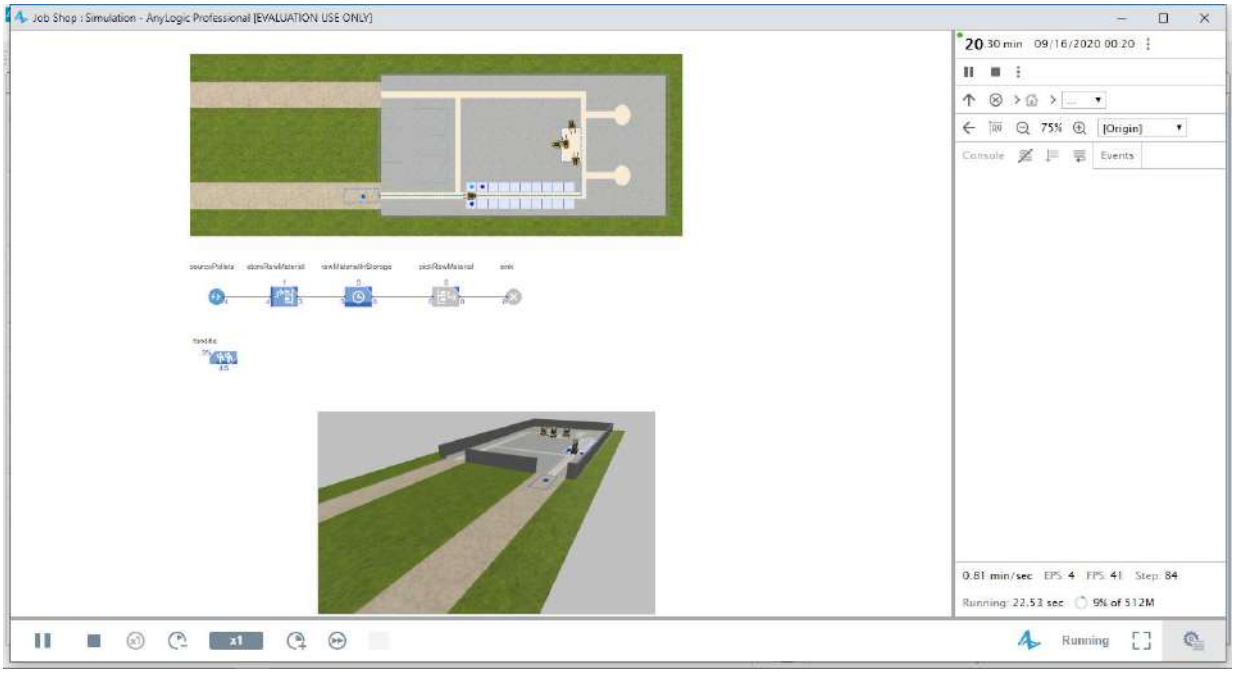
Go to the wall's Position and size section and change the Z-Height to 40.



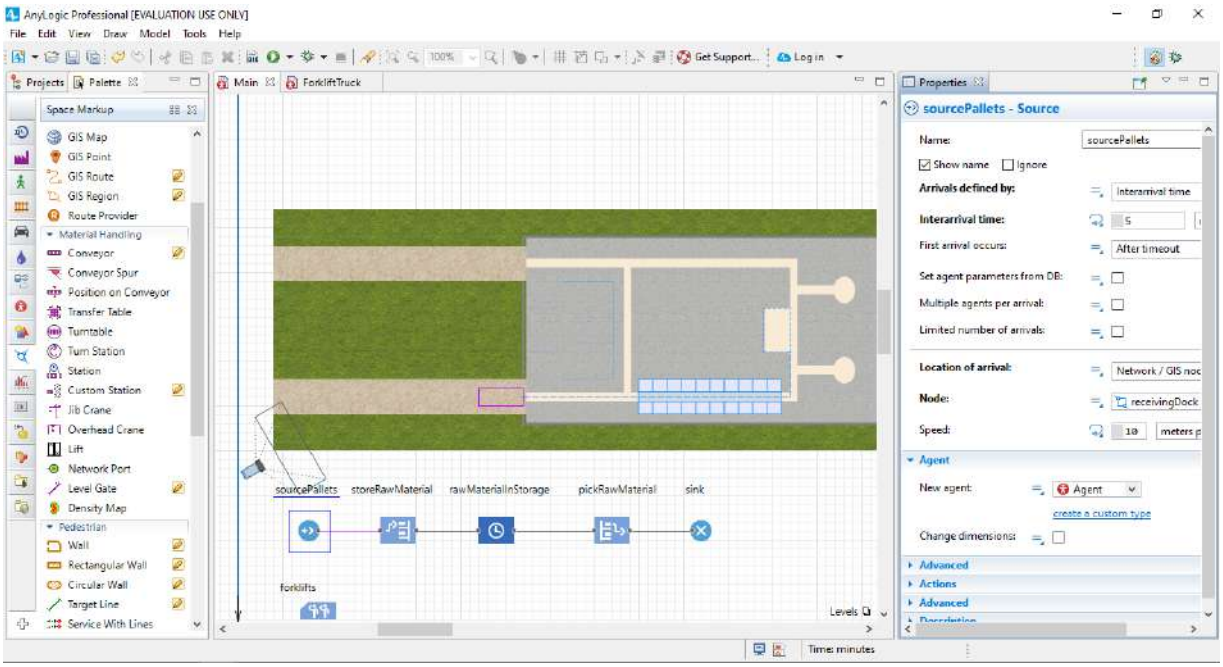
Draw another wall between the exits and then change the settings in the second wall's Properties area to match the first wall.



Run the model and view the 3D animation.



In the sourcePallets block's Properties area, under the New agent list, click the create a custom type link.

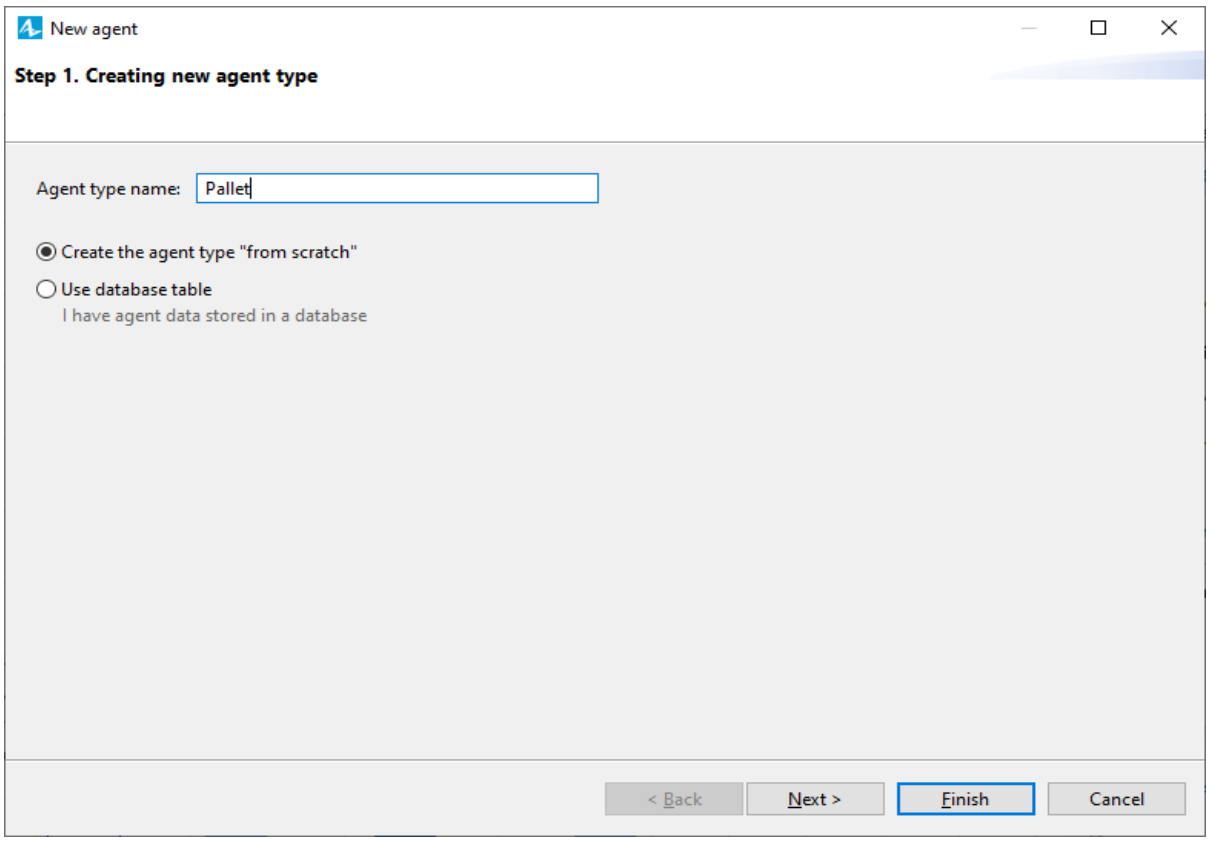


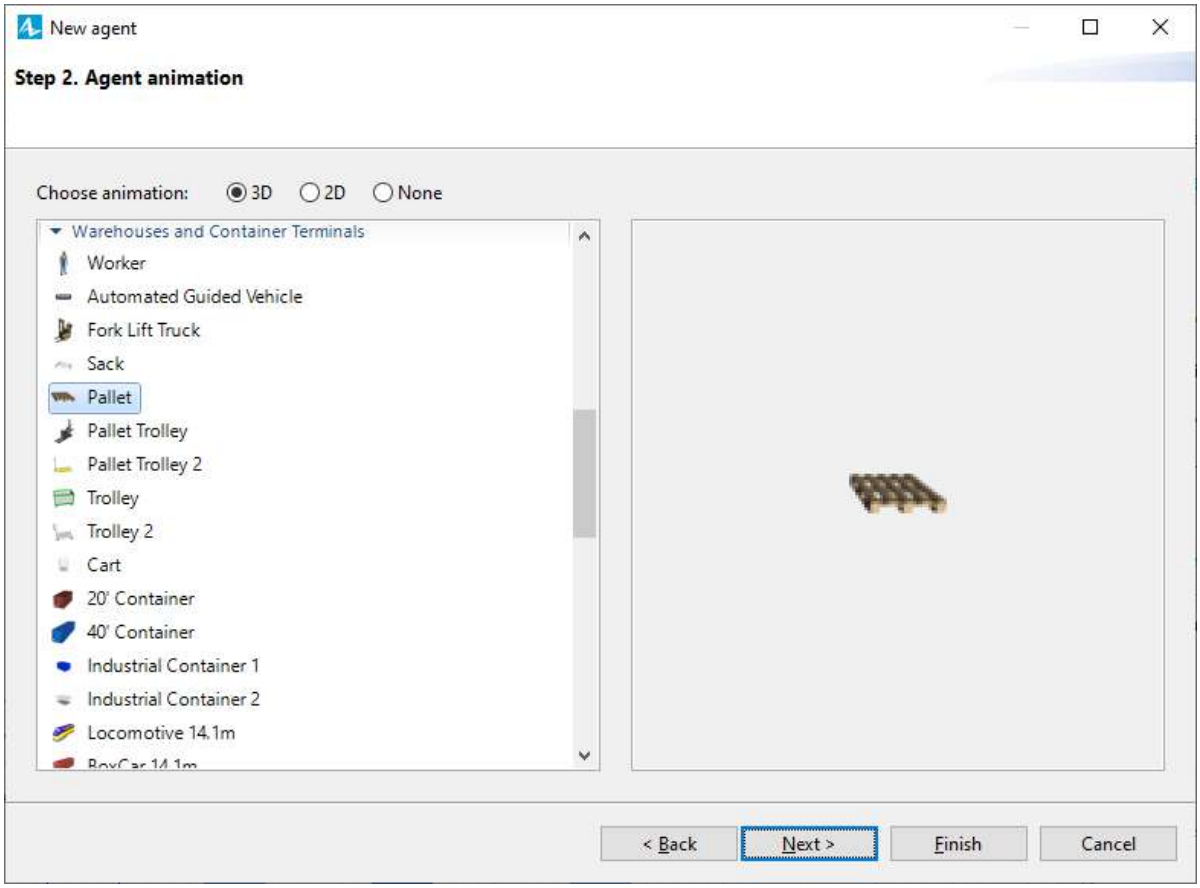
In the New agent wizard, do the following:

a. In the The name of new typelist, type Pallet.

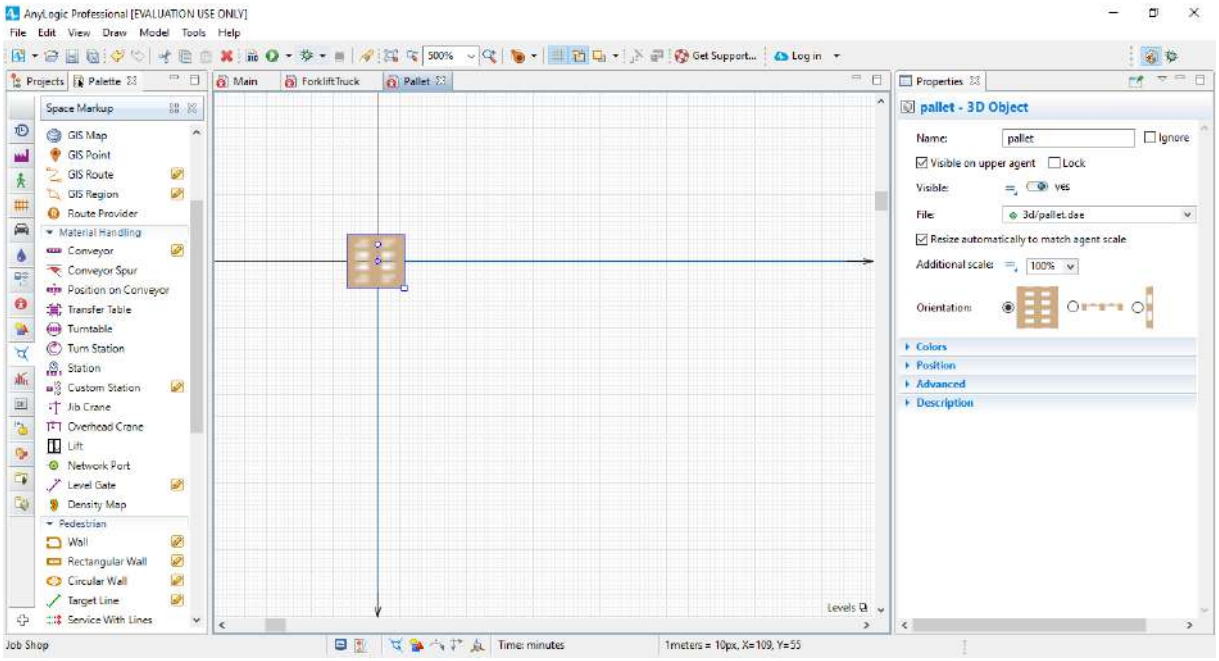
b. In the list in the left part of the wizard, expand the Warehouses and Container Terminals area, and then click the 3D animation figure Pallet.

c. Click Finish


New agent
Step 1. Creating new agent type
Agent type name: Pallet
Create the agent type "from scratch"
Use database table
I have agent data stored in a database
< Back
Next >
Finish
Cancel


New agent
Step 2. Agent animation
Choose animation: 3D 2D None
Warehouses and Container Terminals
Worker
Automated Guided Vehicle
Fork Lift Truck
Sack
Pallet
Pallet Trolley
Pallet Trolley 2
Trolley
Trolley 2
Cart
20' Container
40' Container
Industrial Container 1
Industrial Container 2
Locomotive 14.1m
< Back
Next >
Finish
Cancel

Using the Zoom toolbar, enlarge the Pallet diagram to 500%, and then move the canvas to the right and down to view the axis' origin point and pallet animation shape.

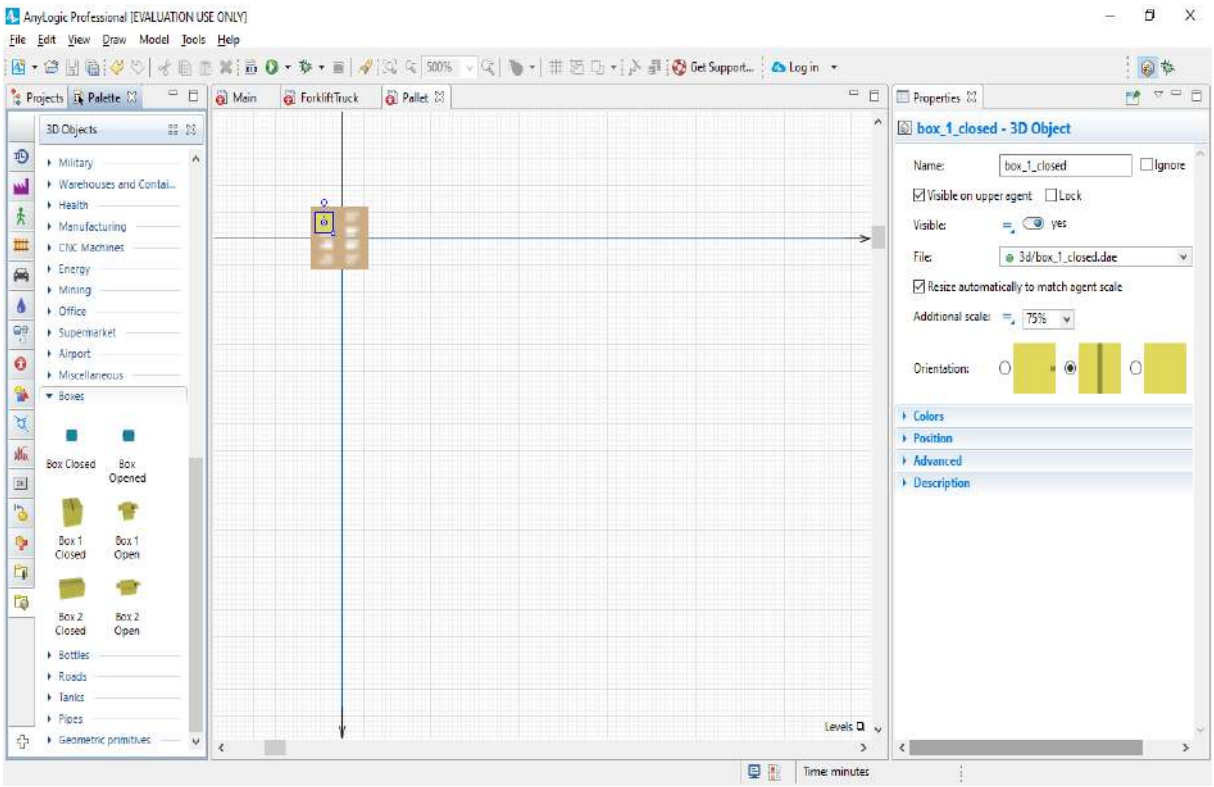


Do the following to start adding product animation on top of the pallet animation.

a. On the 3D Objects palette, expand the Boxes area.

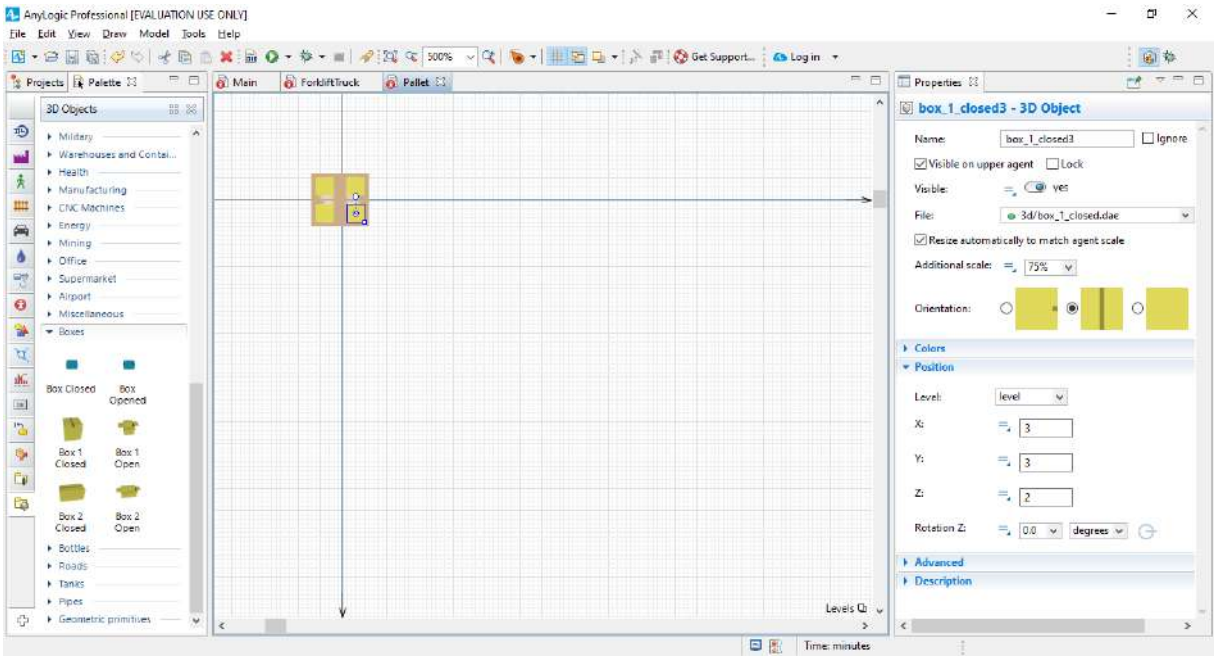
b. Drag the Box 1 Closed object on to the pallet's upper-left corner.

Since this box appears to be too large when compared to the pallet, let's change the box's Scale to 75%.



In the box's Properties area, expand the Position section, and then change the box's Z coordinate to 2. Our change reflects the fact that we want to place boxes on the pallets and each pallet's height is about 2 pixels.

Add three boxes by copying the first box three times. To copy the box, select it and then press and hold CTRL as you drag the box. Our pallet now has four closed boxes, and you can now change the zoom level back to 100% by clicking the toolbar's Zoom to 100% button.



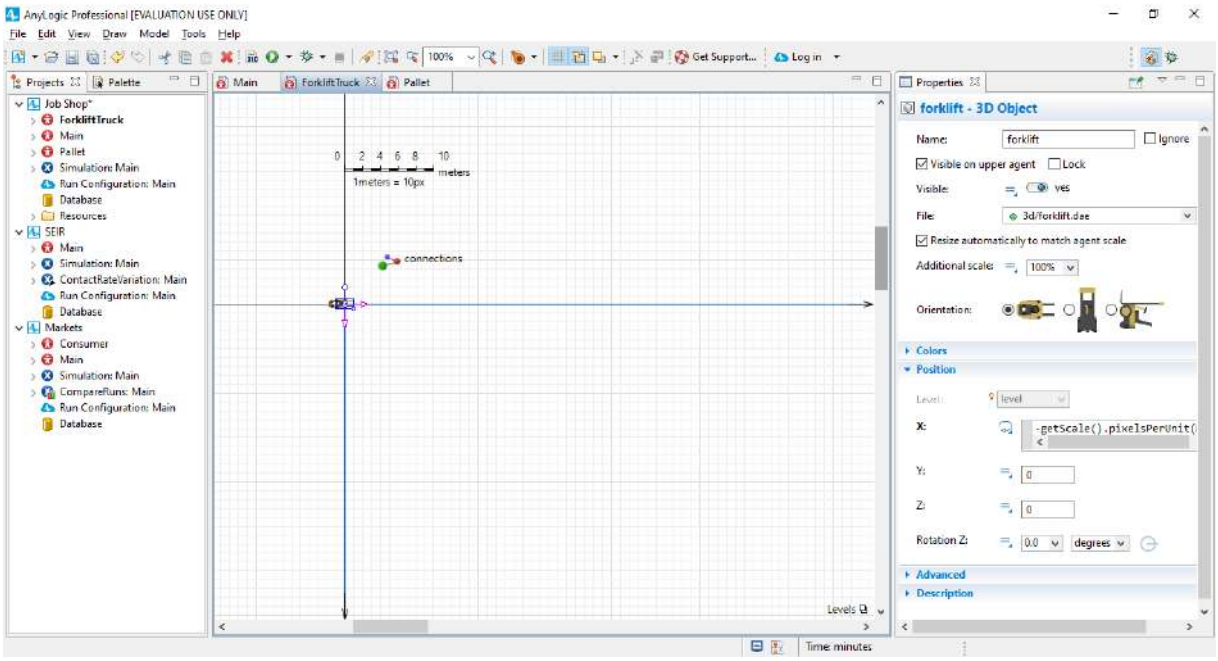
Return to the Main diagram.

If you open the sourcePallets block's Properties area, you'll see Pallet is selected as New agent. This block will generate agents of the Pallet type.

Run the model.

You'll see pallet shapes have replaced the multi colored cylinders. However, if you zoom in the 3D scene, you'll notice that the forklift trucks aren't transporting pallets. We'll correct this problem by moving our model's pallet animation in a way that allows the forklift trucks to pick up the pallets.

In the Projects view, double-click the ForkliftTruck agent type to open its diagram and then move the forkliftWithWorker figure one cell to the right.



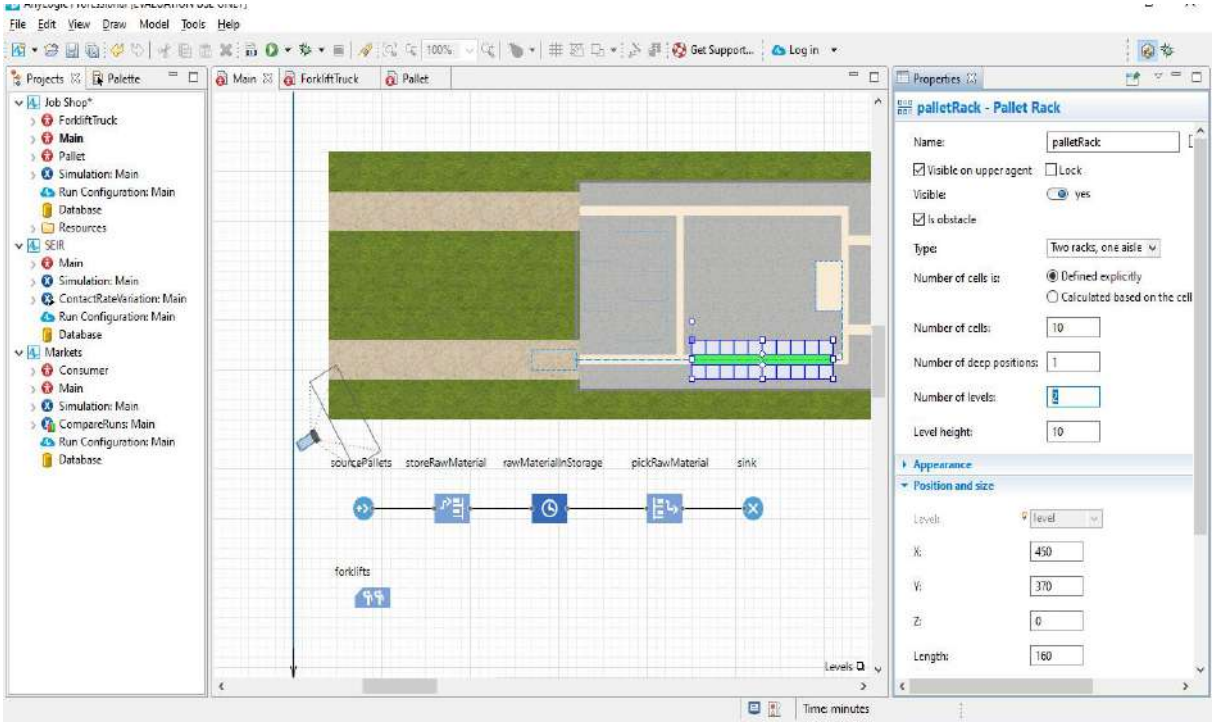
Open Main diagram, and in the pallet rack's Properties area, in the Number of levels box, type 2.

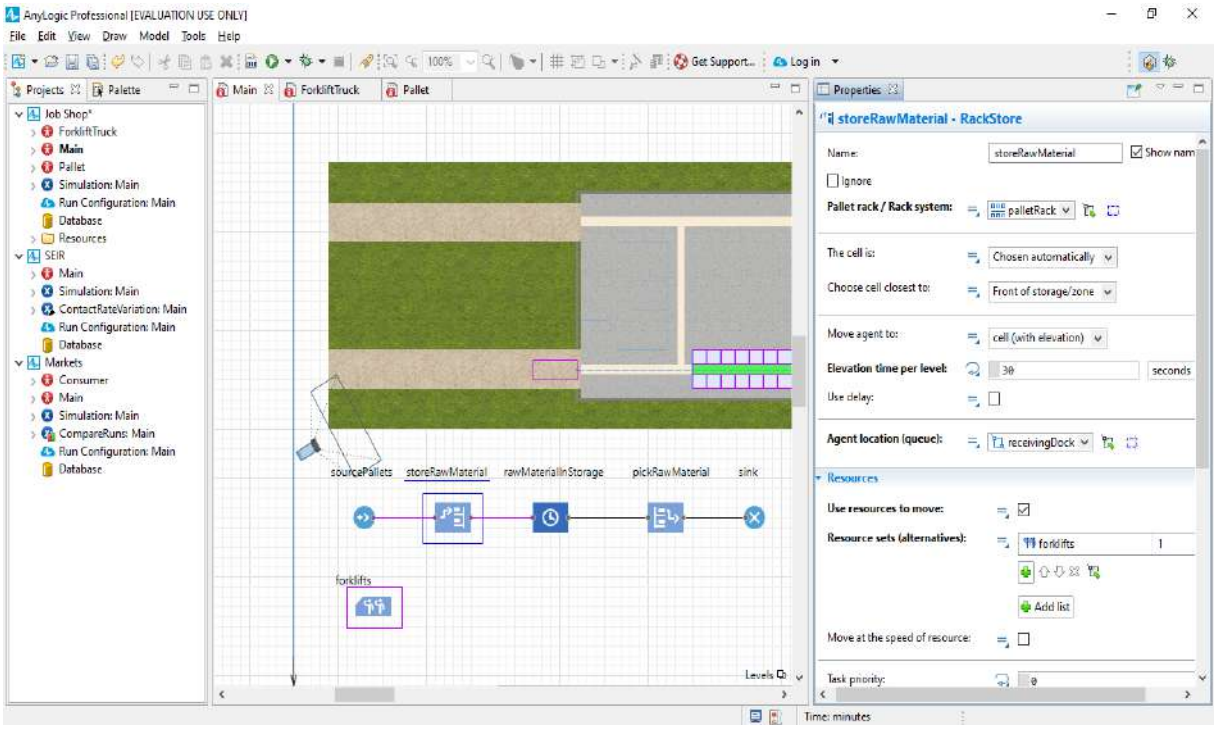
TIP: Remember that your first click will select the network and your second click will select the network element.

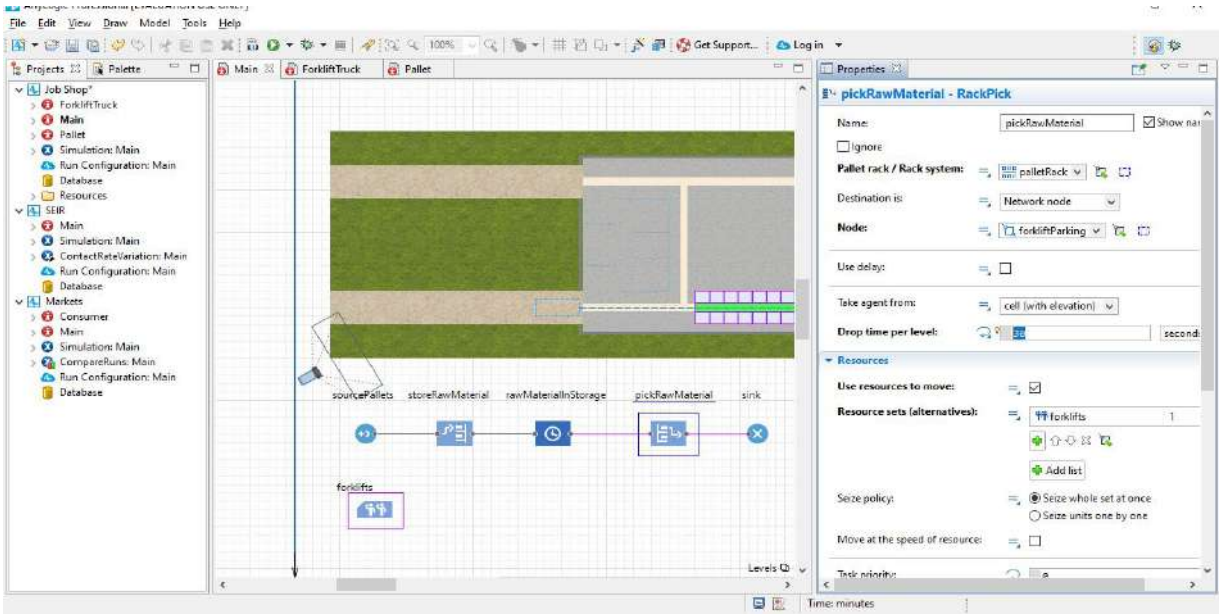
In the storeRawMaterial flowchart block's Properties area, set the Elevation time per level parameter to 30 seconds.

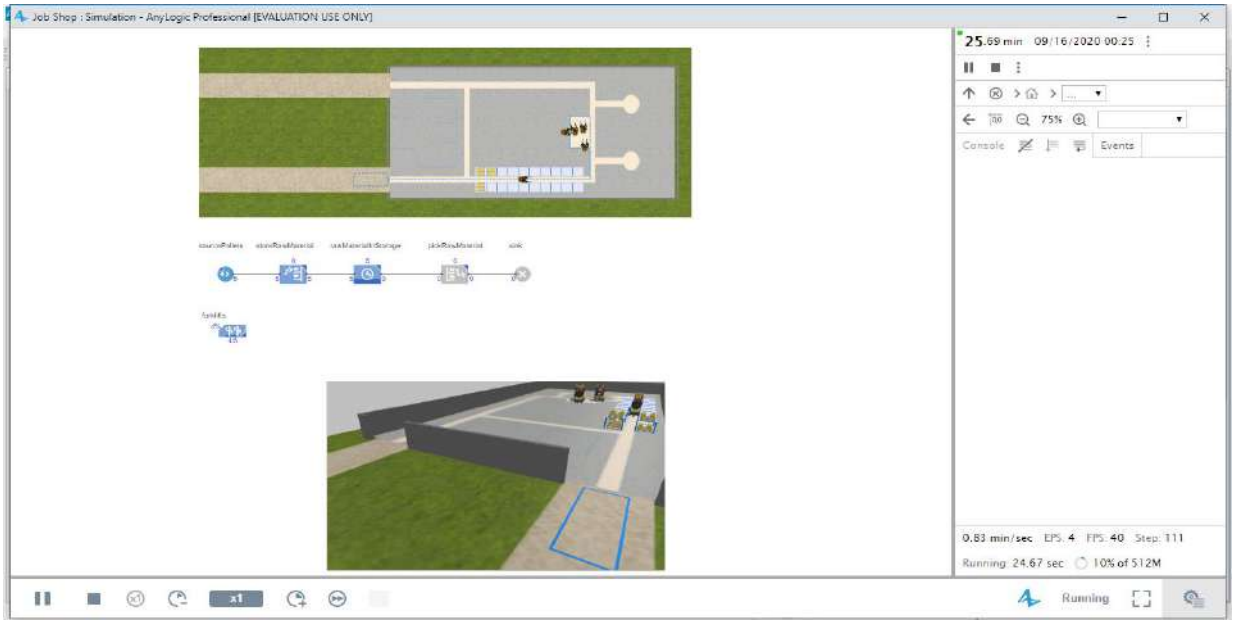
In the pickRawMaterial block's Properties area, set the Drop time per level parameter to 30 seconds.

Run the model and you'll see a pallet rack with two levels.






File Edit View Draw Model Tools Help
Projects
Palette
Main
ForkliftTruck
Pallet
Job Shop*
ForkliftTruck
Main
Pallet
Simulation: Main
Run Configuration: Main
Database
Resources
SEIR
Main
Simulation: Main
ContactRateVariation: Main
Run Configuration: Main
Database
Markets
Consumer
Main
Simulation: Main
CompareRuns: Main
Run Configuration: Main
Database
sourcePallets
storeRawMaterial
rawMaterialInStorage
pickRawMaterial
sink
forklifts
Properties
pickRawMaterial - RackPick
Name: pickRawMaterial
Ignore
Pallet rack / Rack system: palletRack
Destination is: Network node
Node: forkliftParking
Use delay:
Take agent from: cell (with elevation)
Drop time per level:
Resources
Use resources to move:
Resource sets (alternatives): forklifts
Add list
Seize policy: Seize whole set at once
Seize units one by one
Move at the speed of resource:
Levels
Time: minutes


Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
25.69 min 09/16/2020 00:25
75%
Console
Events
0.83 min/sec EPS: 4 FPS: 40 Step: 111
Running: 24.67 sec 10% of 512M
x1
Running


Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
370.43 min 09/16/2020 06:10
75%
Console
Events
0.81 min/sec EPS: 9 FPS: 39 Step: 3,836
Running: 437.65 sec 6% of 512M
x1
Running

Modelling Delivery

we'll add the trucks that deliver the pallets to the job shop. Let's start by creating an agent type to represent them.

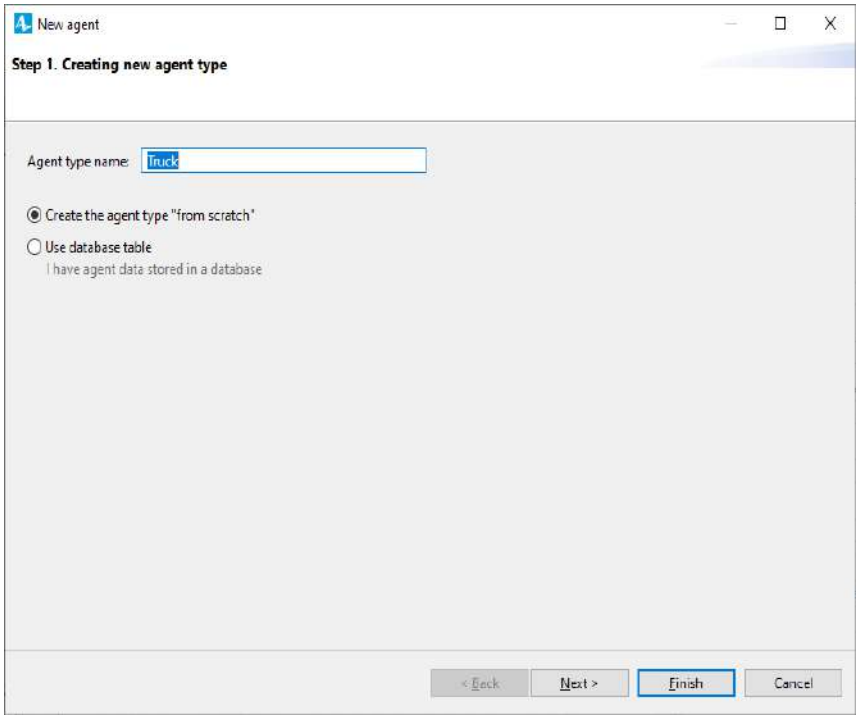
On the Process Modeling Library palette, drag the Agent type element on to the Main diagram.

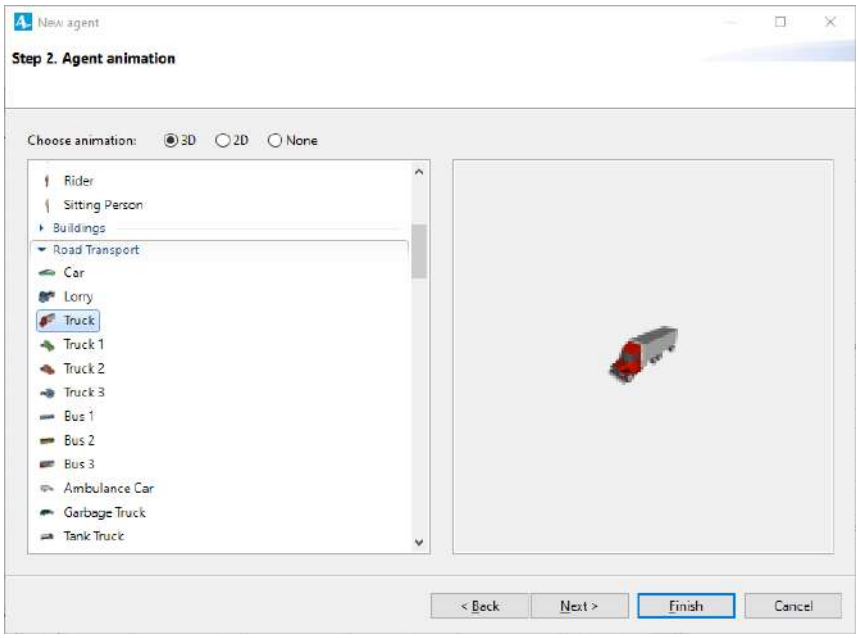
On the New agent wizard's Agent animation screen, do the following:

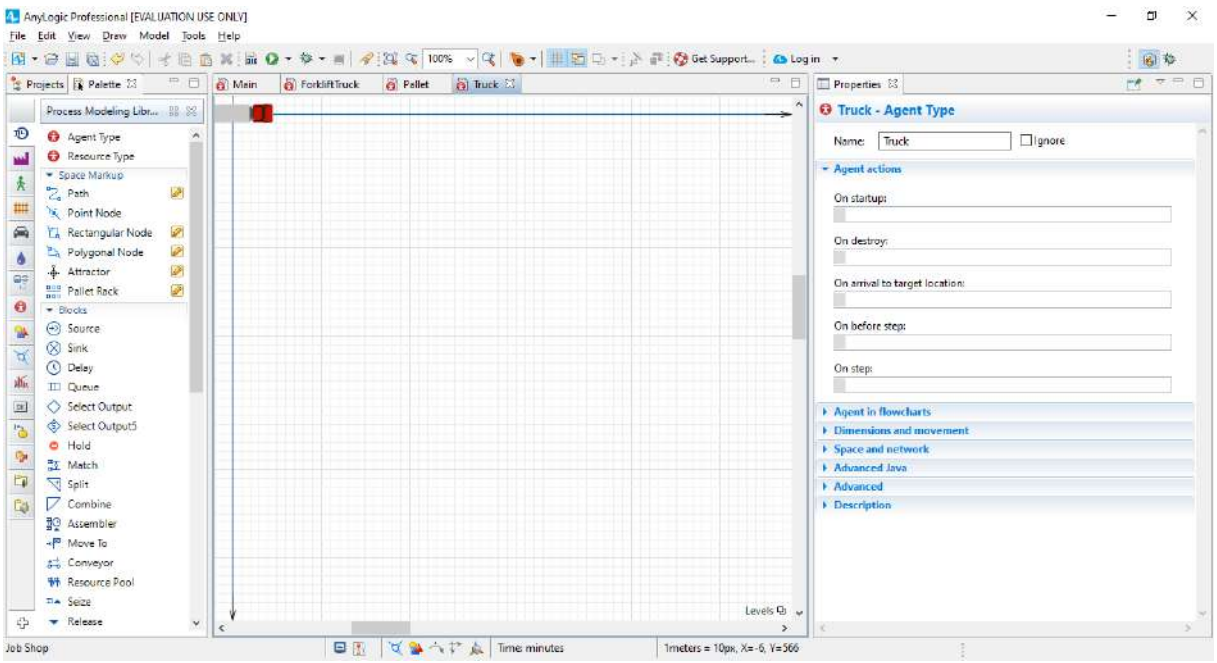
a. In the The name of new type box, type Truck.

b. In the list below, expand the Road Transport area and then click Truck.

c. Click Finish.







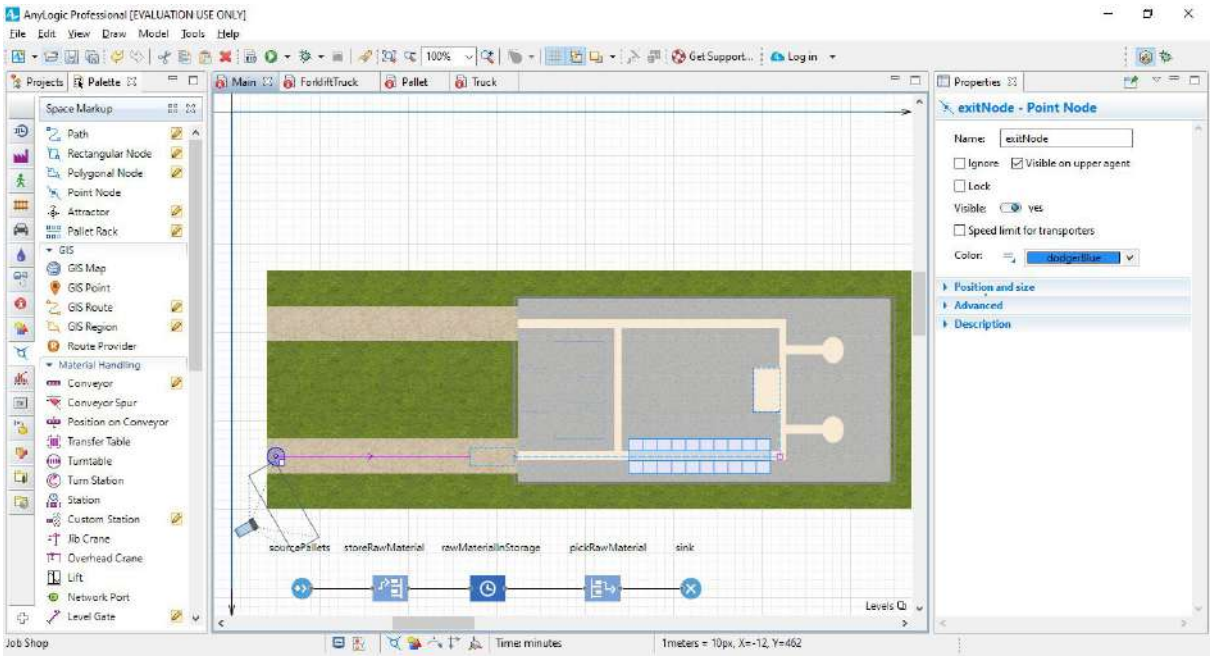
Let's add two more elements to our network: a node where the trucks will appear and the path that they will follow to the receiving dock.

Open the Main diagram,

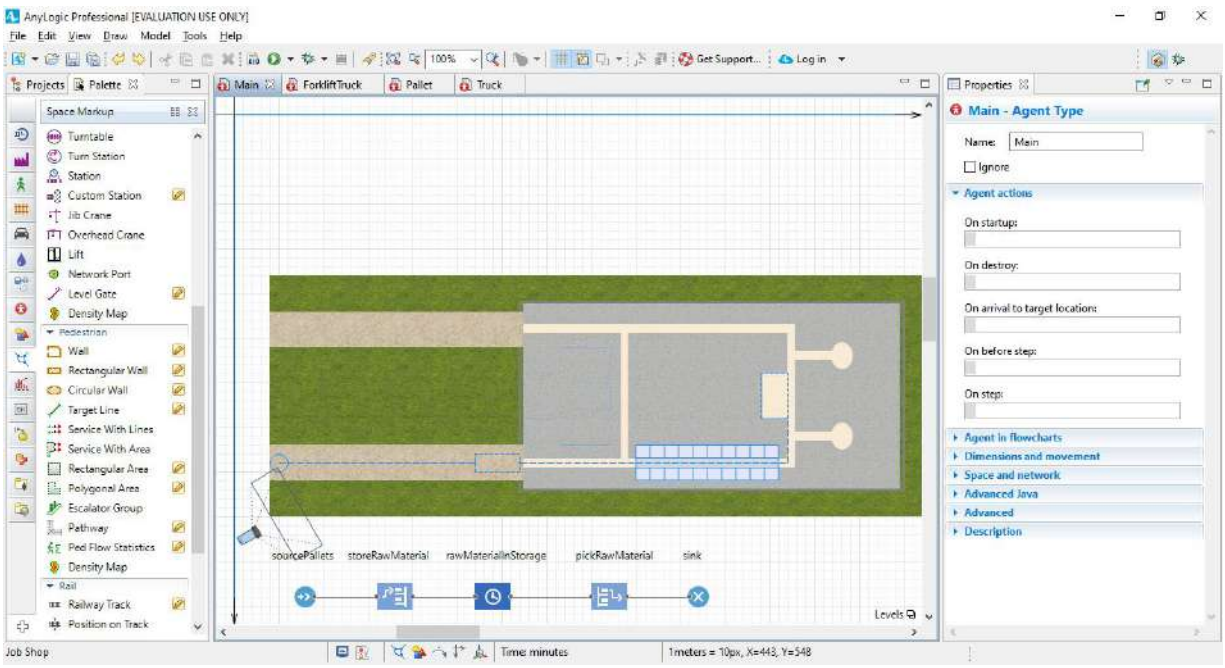
Under the Space Markup palette, click the Point Node element and drag it on to the driveway entry.

Name the node exitNode

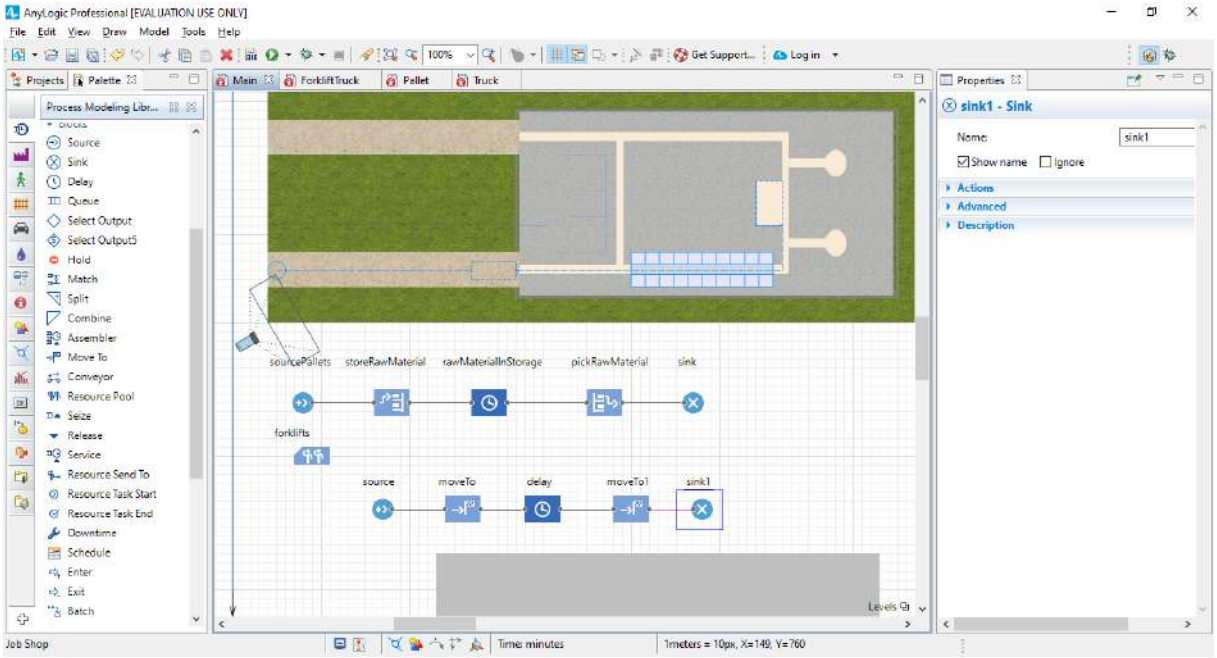
Draw a Path to connect the exitNode to the receivingDock.



Make sure all space markup elements connect to one network.



Create another process flowchart to define the truck movement logic by connecting the Process Modeling Library blocks in the following order:



Name the Source block sourceDeliveryTrucks.

In the sourceDeliveryTrucks block's Properties area, do the following to have a new agent of the custom Truck type arrive to the driveway entry once per hour at a specific speed:

a. In the Arrivals defined by list, click Interarrival time.

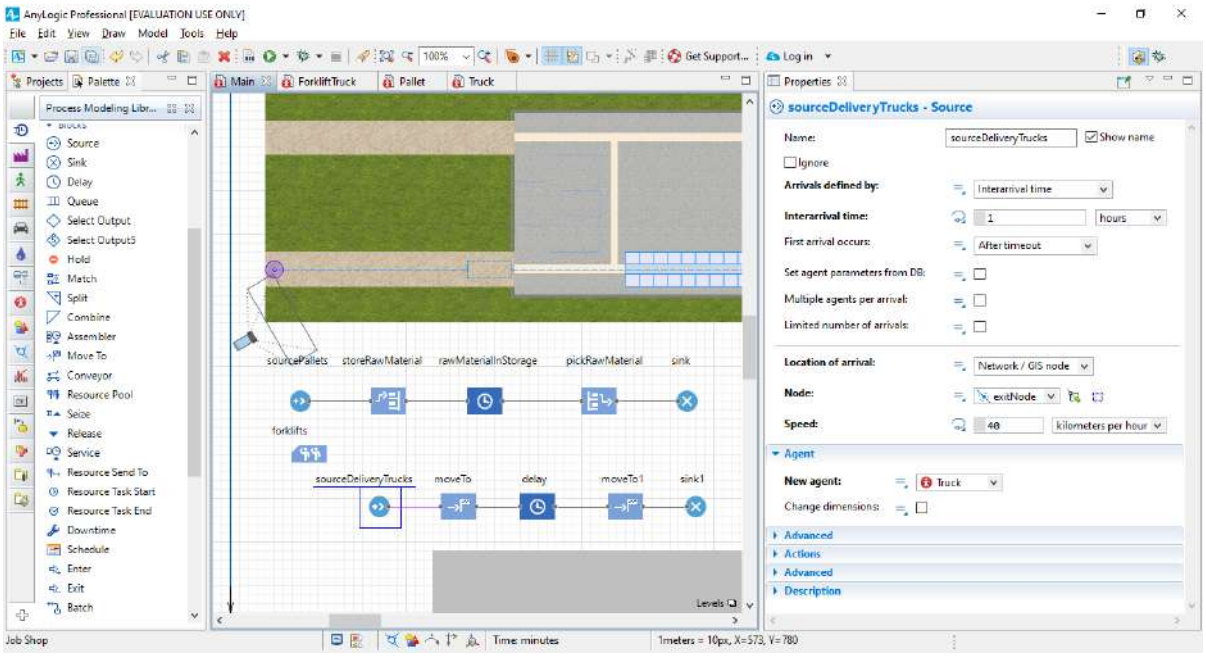
b. In the Interarrival time box, type 1, and select hours from the list on the right.

c. In the New agent list, click Truck.

d. In the Location of arrival list, click Network/GIS node.

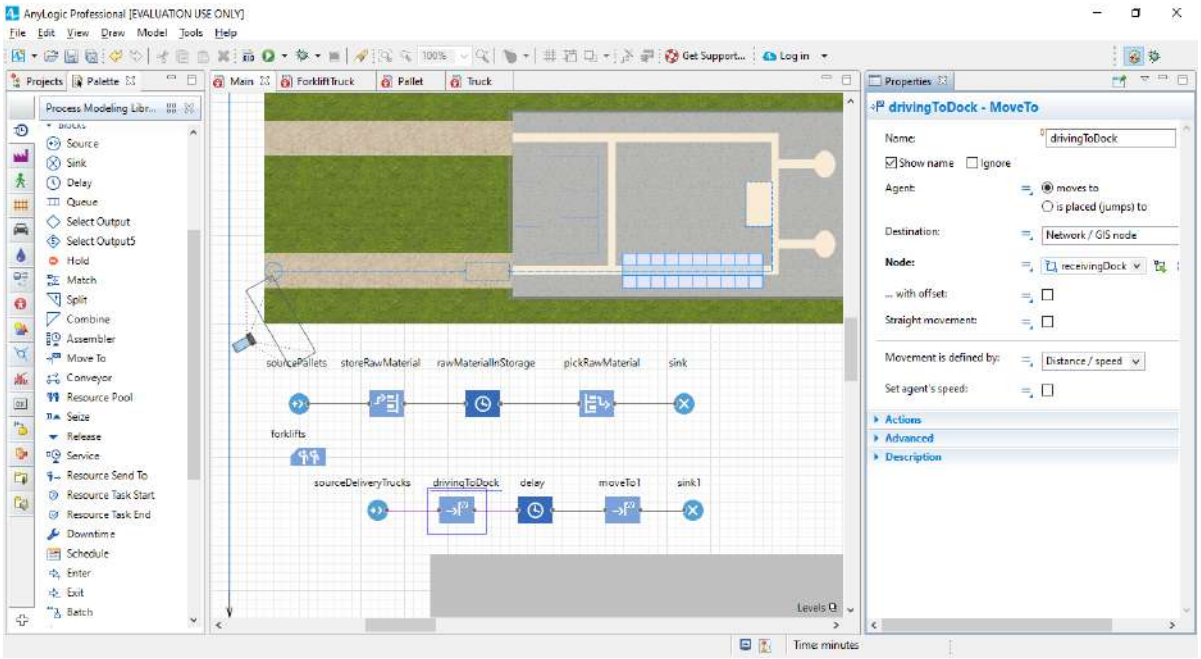
e. In the Node list, click exitNode.

f. In the Speed box, type 40, and select kilometers per hour from the list on the right.



Name the first MoveTo block drivingToDock.

In the drivingToDock block's Properties area, in the Node list, click receivingDock to set the agent's destination.

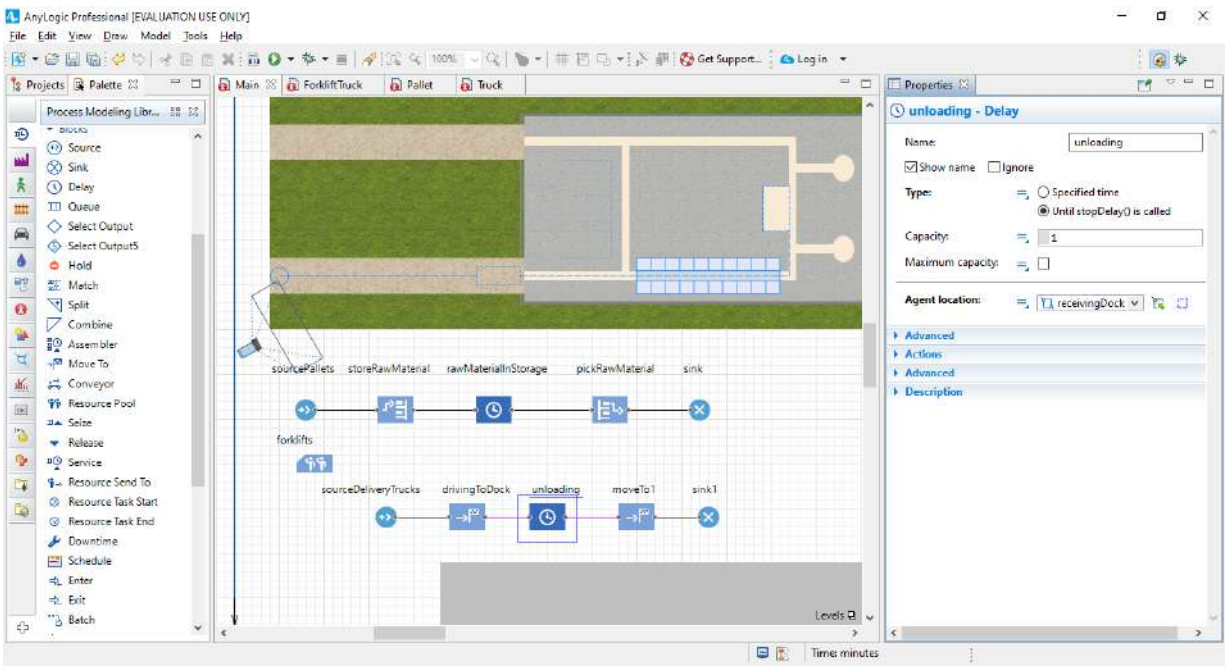


Rename the Delay block to unloading.

In the unloading block's Properties area, do the following:

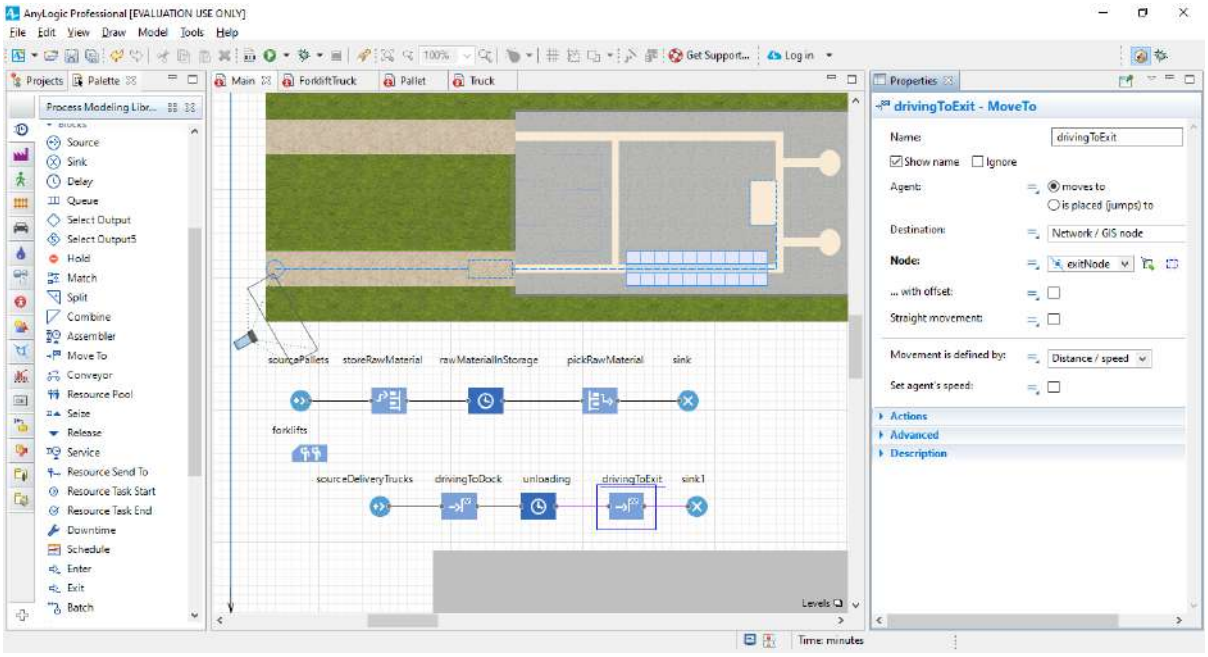
a. In the Type area, click Until stopDelay() is called.

b. In the Agent location list, clickreceivingDock.



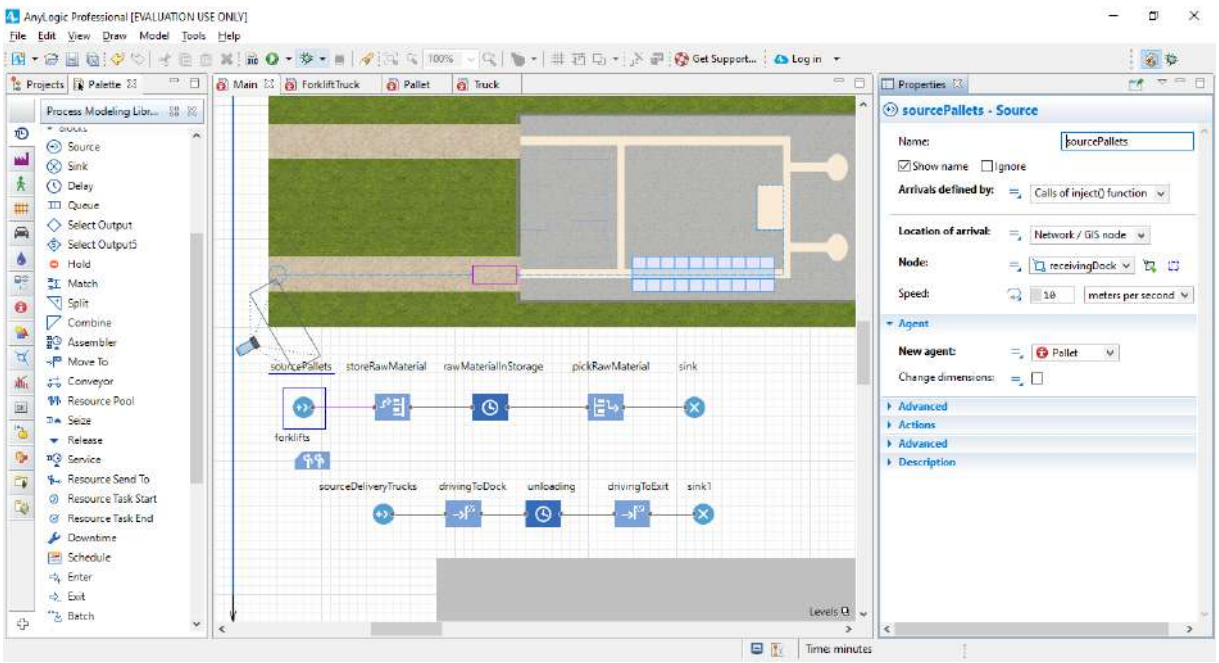
Name the second MoveTo block drivingToExit.

In the drivingToExit. block's Properties area, in the Node list, click exitNode to set the destination node



Our model’s two Source blocks generate two agent types: the trucks that appear each hour and the pallet that is generated every five minutes. Since we want pallets to appear when the truck unloads, we’ll change the arrival mode for the Source block that generates them.

In the sourcePallets block's Properties area, in the Arrivals defined by list, click Calls of inject() function.



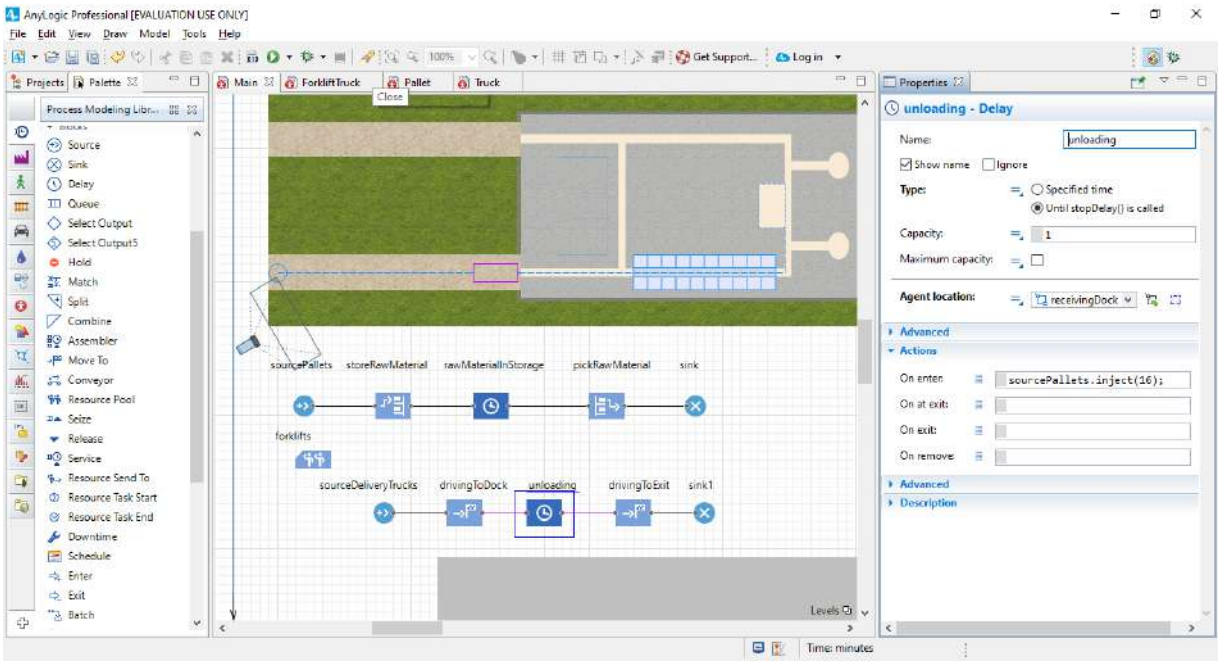
Do the following to have the sourcePallets block generate pallets when a truck enters the unloading block:

a. In the unloading block's Properties area, expand the Actions section.

b. In the On enter box, type the following:

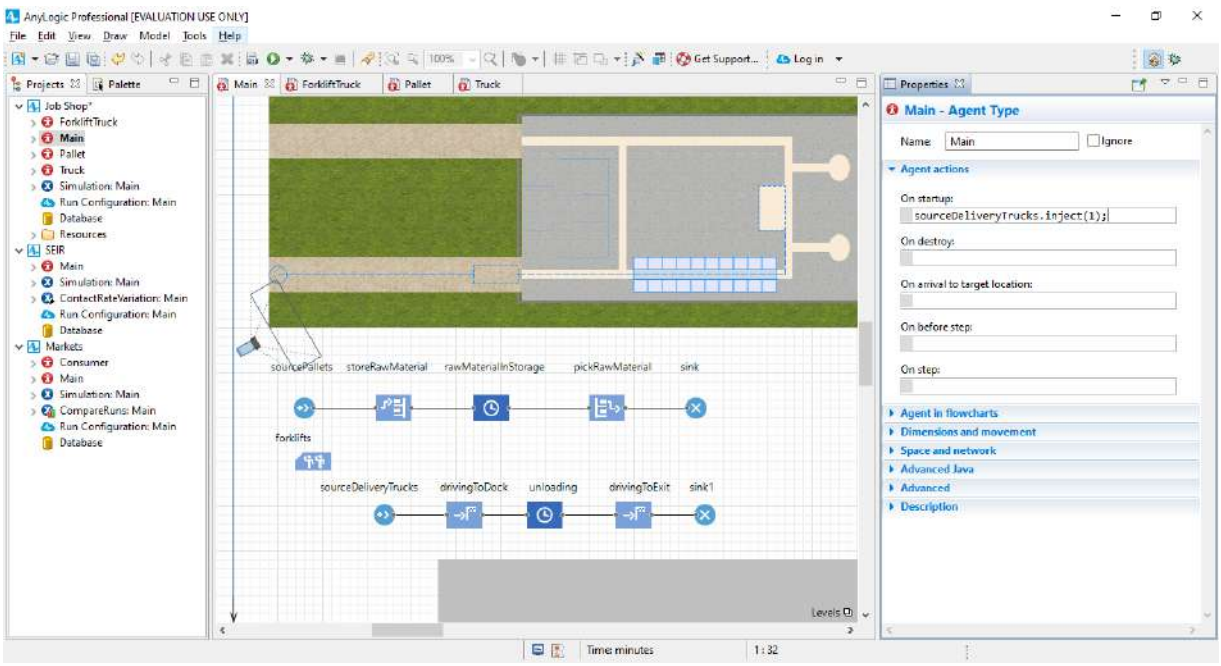
sourcePallets.inject(16);

This Java function will ensure our model generates16 pallets each time a truck starts to unload.



In the Main agent type's Properties area, expand the Agent actions section and then type the following Java function in the On startupbox:

sourceDeliveryTrucks.inject(1);

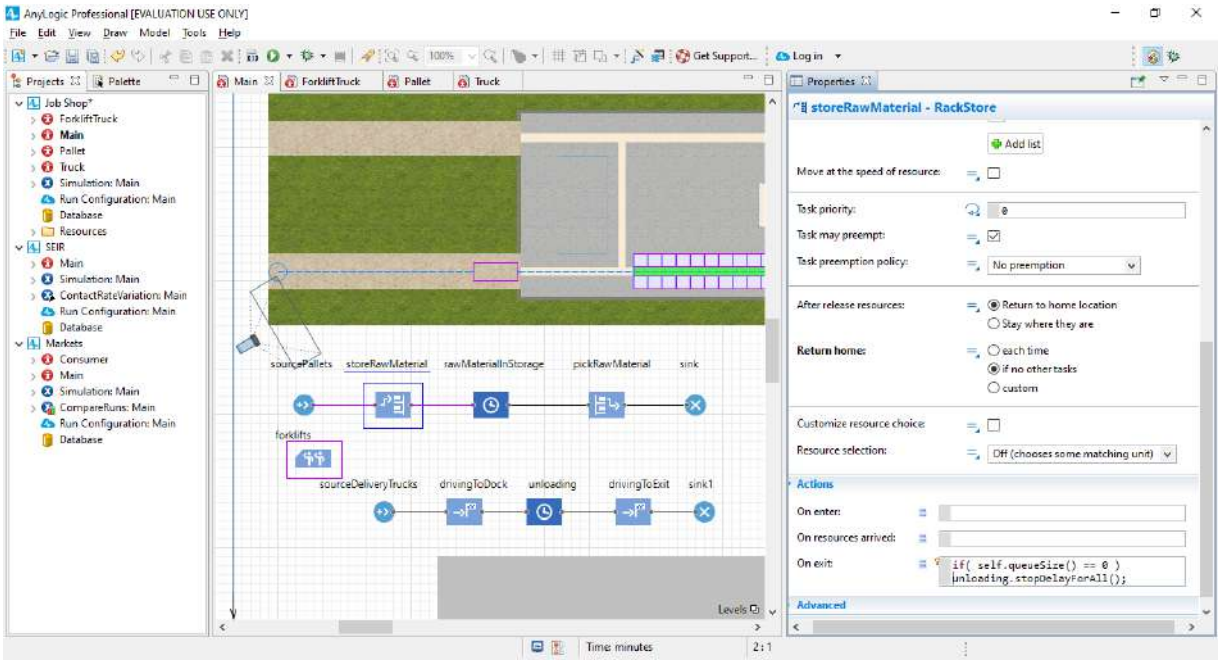


In the storeRawMaterial block's Properties area, expand the Actions section, and in the On exit box, type the following:

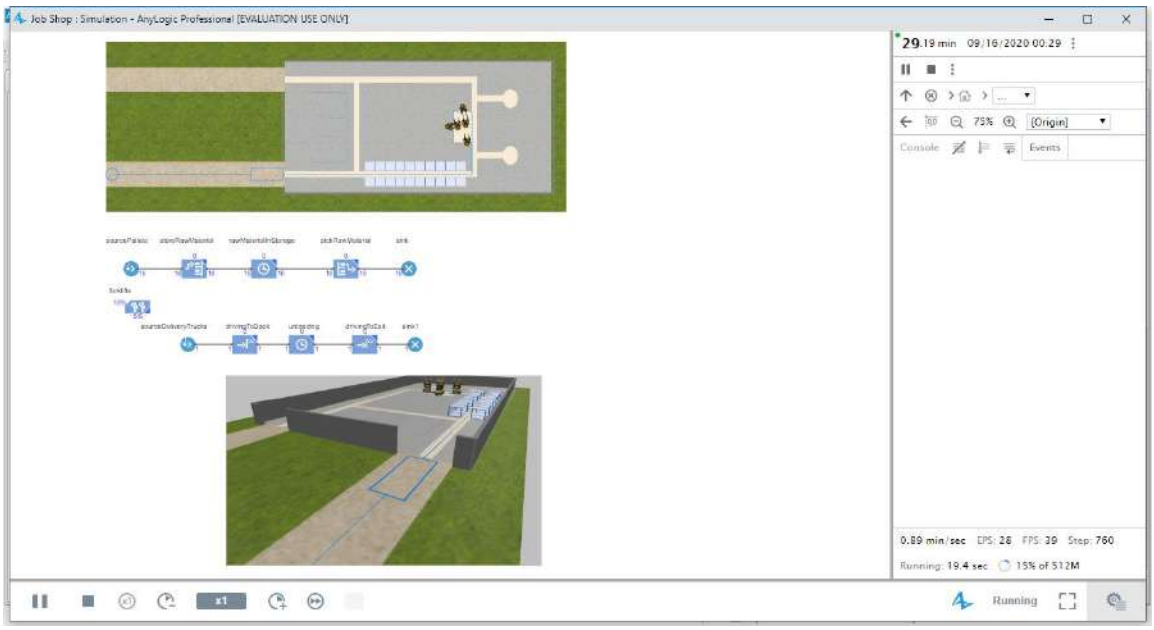
```
if( self.queueSize() == 0 )

unloading.stopDelayForAll();
```

In this example, self is a shortcut we use to refer to the block storeRawMaterial from its own action.



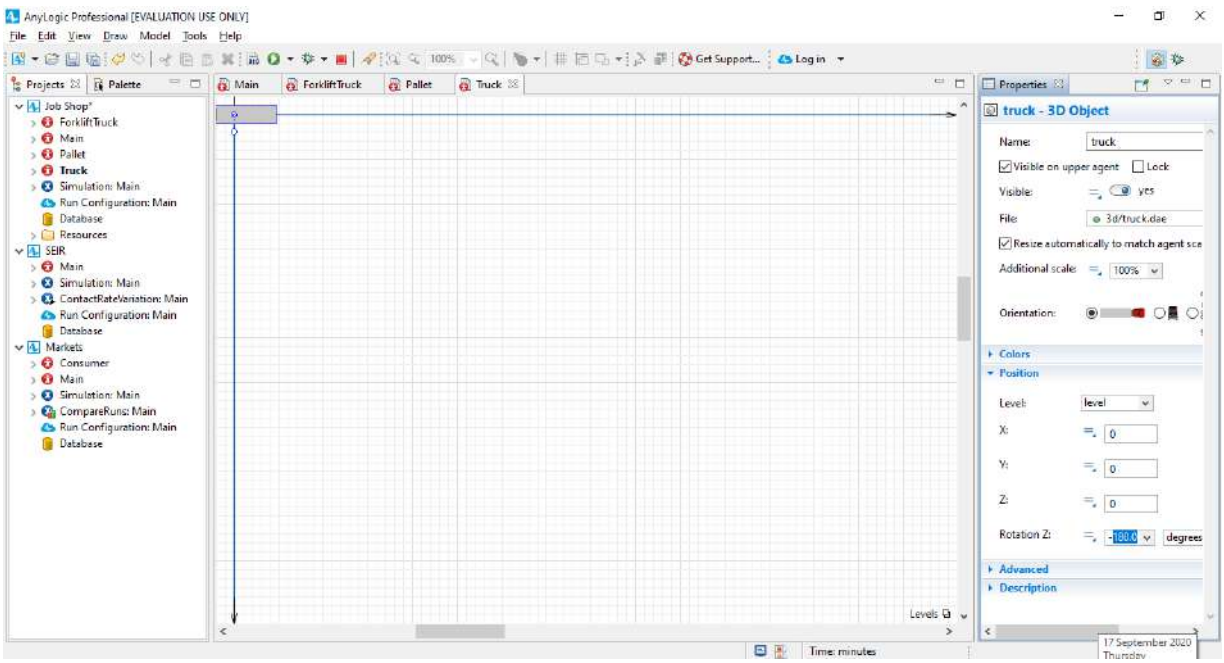
Run the Model



If the truck is aligned incorrectly as in the figure below, do the following to fix it.

In the Projects tree, double-click the Truck agent type to open its diagram and view the truck animation figure.

b. In the graphical editor, select the truck shape and then use the round handle or the Rotation Z,° property in the shape's Position properties area to rotate the truck to -180 degrees.



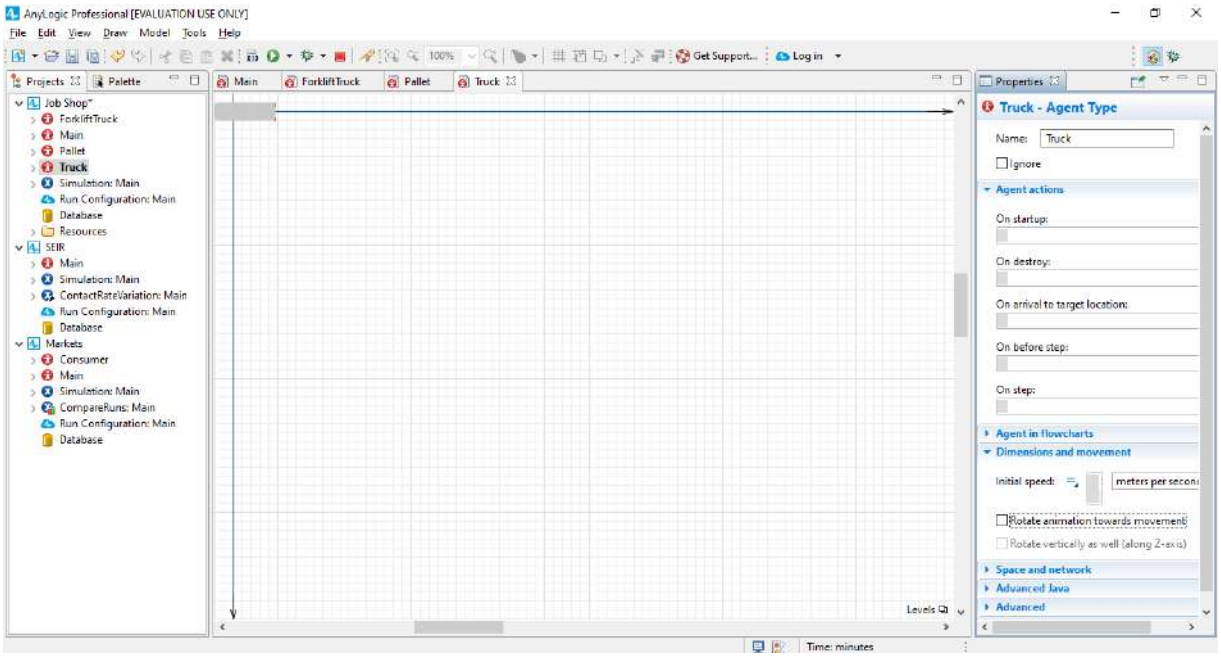
We've changed the truck figure's position, but we'll also need to change AnyLogic's default setting to make sure the program doesn't rotate it a second time.

Do the following to change AnyLogic's default setting:

a. In the Projects area, click Truck.

b. On the Truck agent type's Properties area, click the arrow to expand the Movement area.

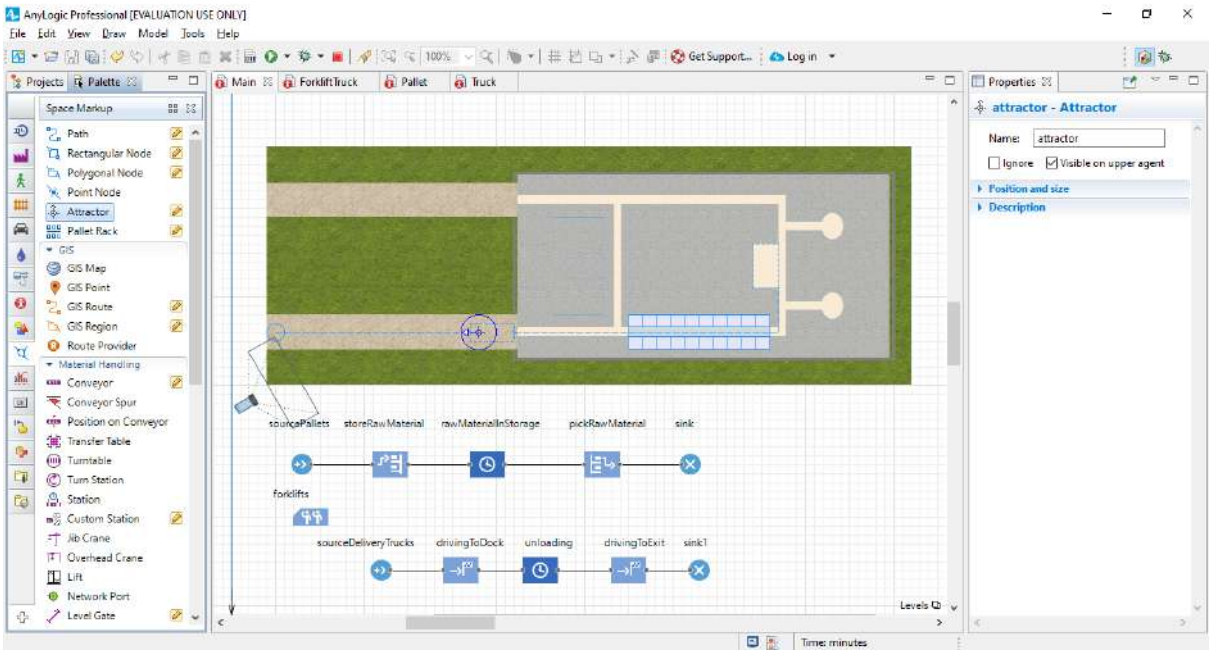
c. Clear the Rotate animation towards movement check box.



Open the Main diagram.

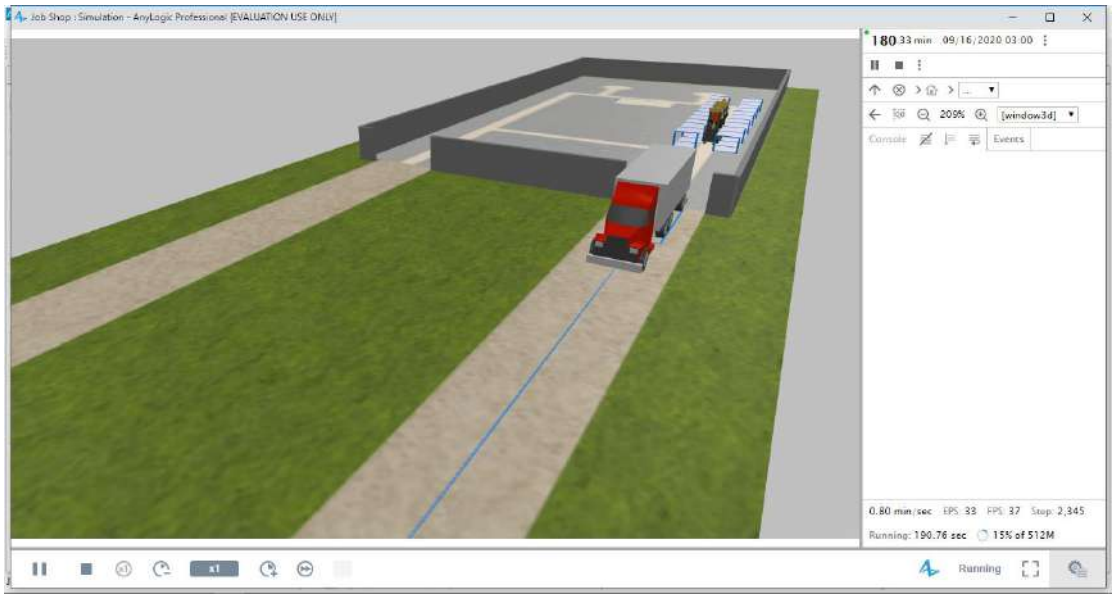
To ensure the pallets are correctly positioned in the receivingDock network node, open the Space Markup palette, and drag an Attractor into receivingDock.

Let it face the entrance.



Run the model to check the truck behavior.


Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
24.69 min 09/16/2020 00:24
Running


Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
180.33 min 09/16/2020 03:00
Running


Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
241.37 min 09/16/2020 04:01
Running

PRACTICAL NO: 06

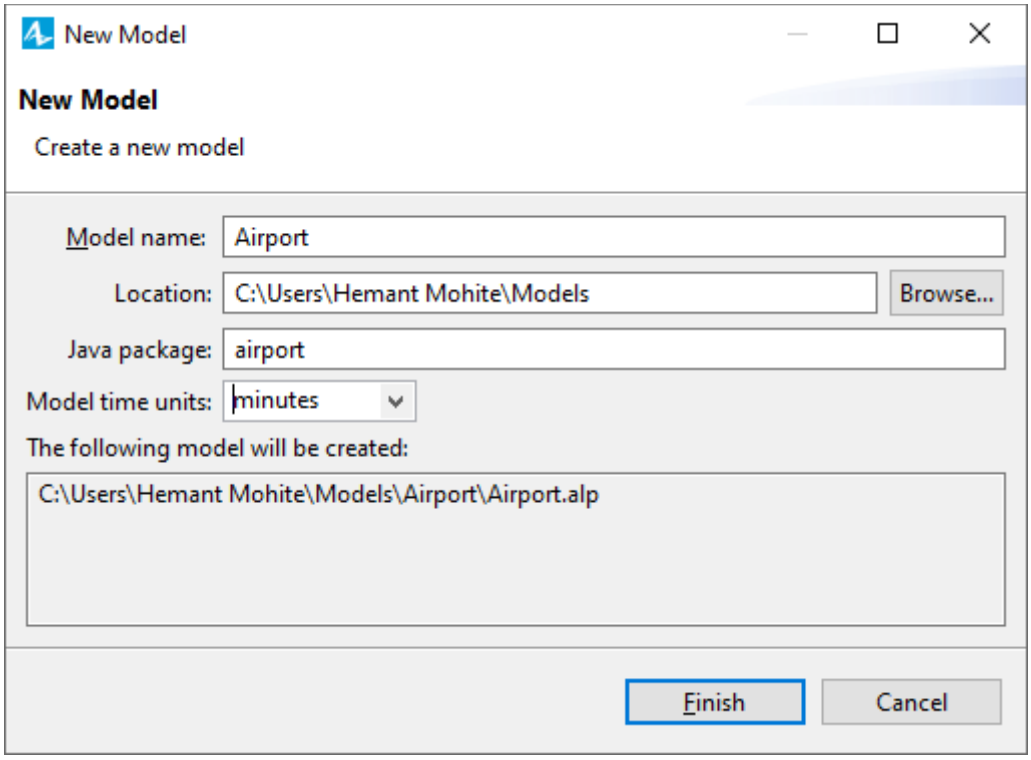
Aim: Design and develop time-slice simulation for a scenario like airport model to design how passengers move within a small airport that hosts two airlines, each with their own gate. Passengers arrive at the airport, check in, pass the security checkpoint and then go to the waiting area. After boarding starts, each airline's representatives check their passenger's tickets before they allow them to board.

Code:

Phase 1. Defining the simple pedestrian flow

In our first phase, we'll use AnyLogic's Pedestrian Library to create a simple model of an airport where passengers arrive and then move to the gate.

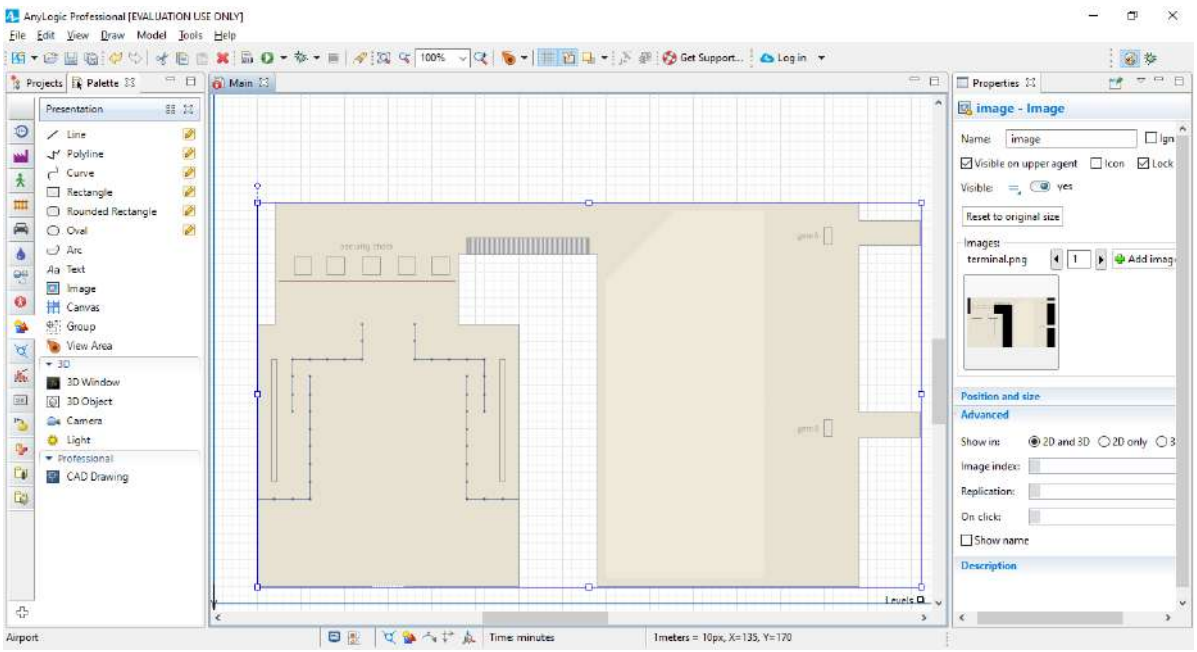
Create a new model and name it Airport.



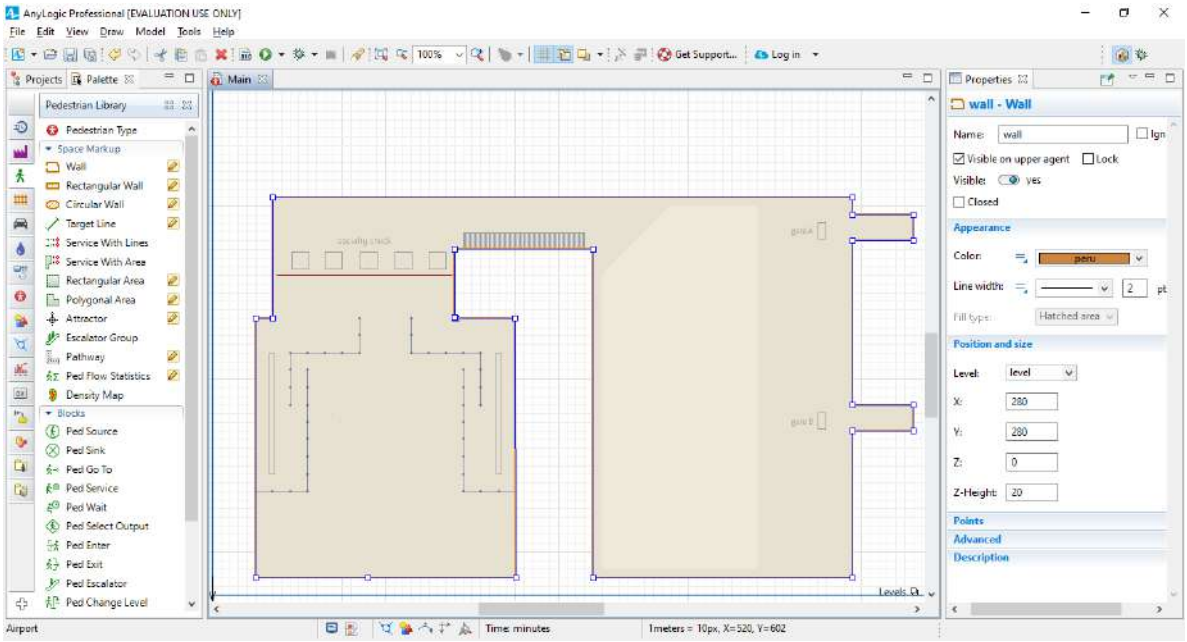
Drag an Image from the Presentation palette on to the Main diagram.

Choose the image file you want to display. In this example, you'll select the terminal.png image file from AnyLogic folder/resources/AnyLogic in 3 days / Airport.

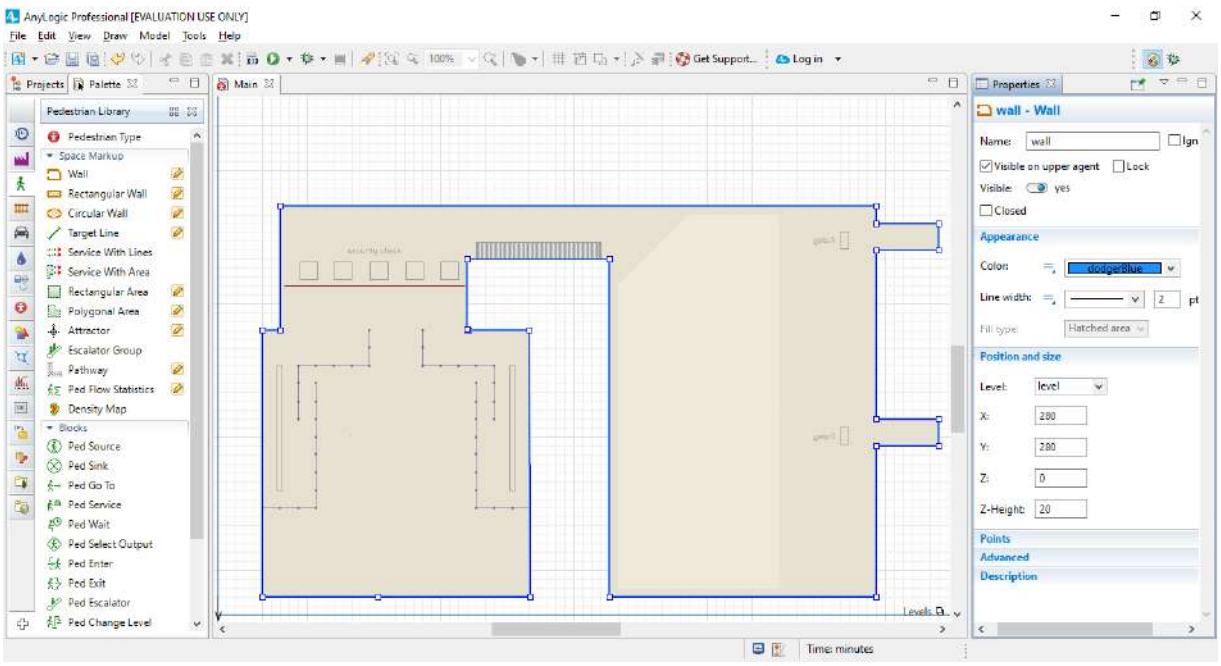
On the Main diagram, place the image in the blue frame's lower left corner. If the image is distorted, click the Reset to original size button and then select the Lock checkbox to lock the image shape



Use the Pedestrian Library palette to draw the airport's walls. Double-click the Wall element you'll find in the Pedestrian Library palette's Space markup section and then draw the wall around the airport building's border by clicking your mouse each time you want to add a point. When you're ready to set the wall's final point, double-click your mouse.

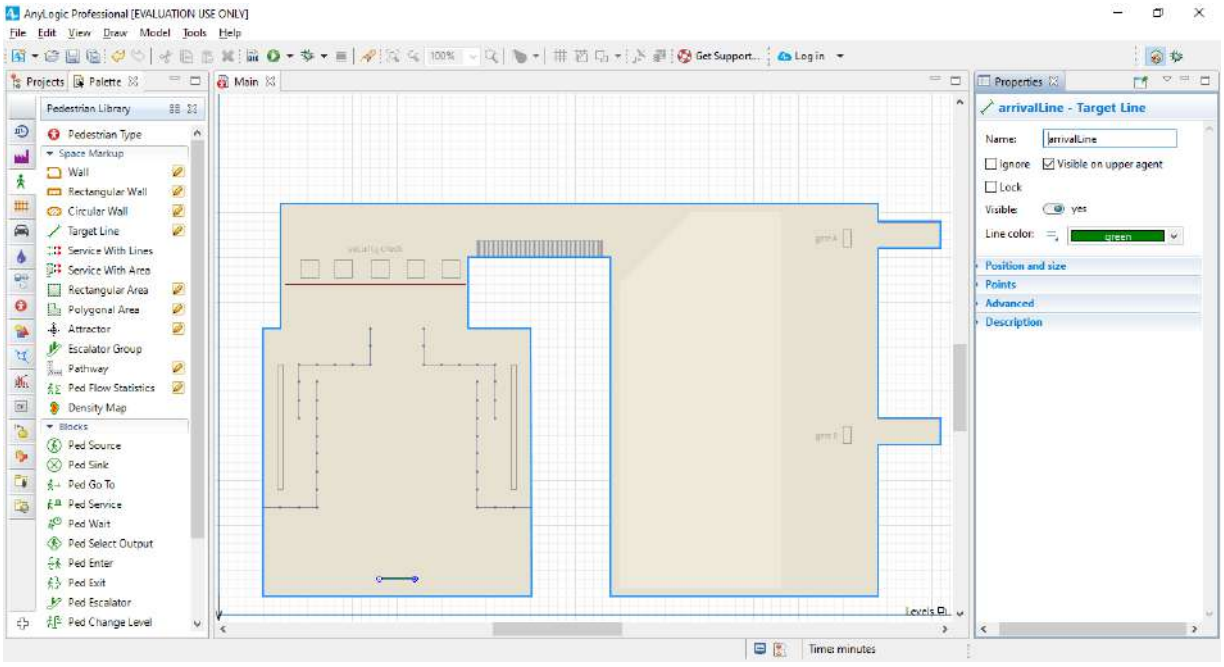


Navigate to the wall's properties and then select the Color: dodgerBlue in the Appearance section.

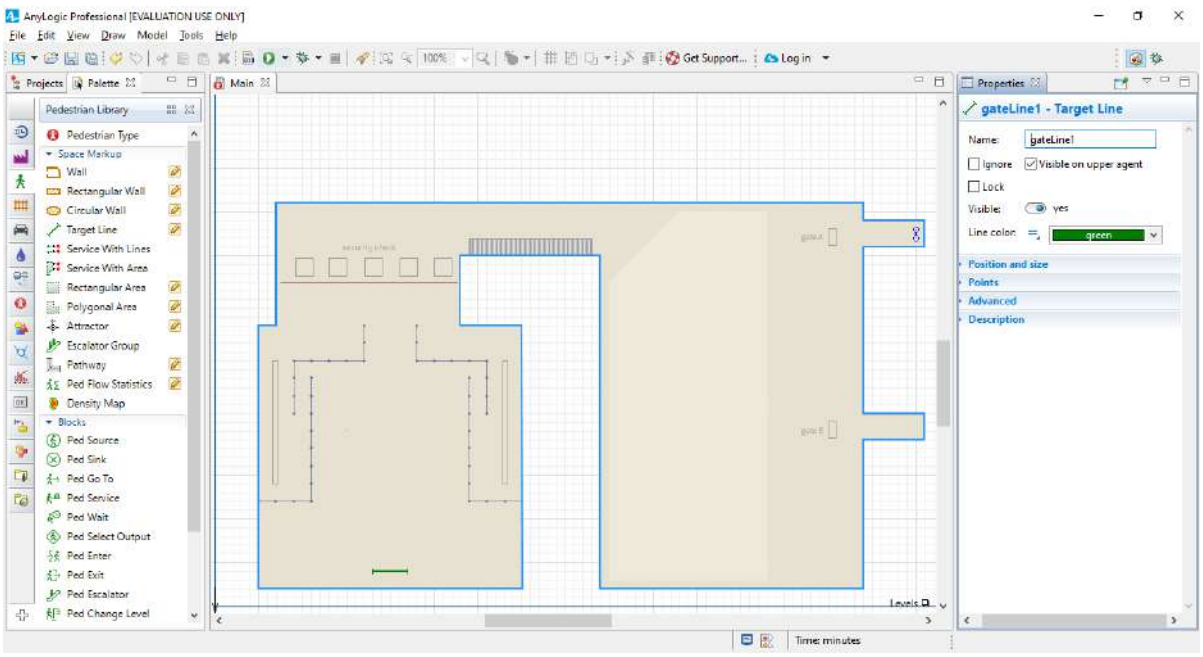


Define the location where your model's passengers appear by dragging the Target Line element from the Pedestrian Library palette on to the graphical diagram, as shown in the figure below.

Name the target line arrivalLine.



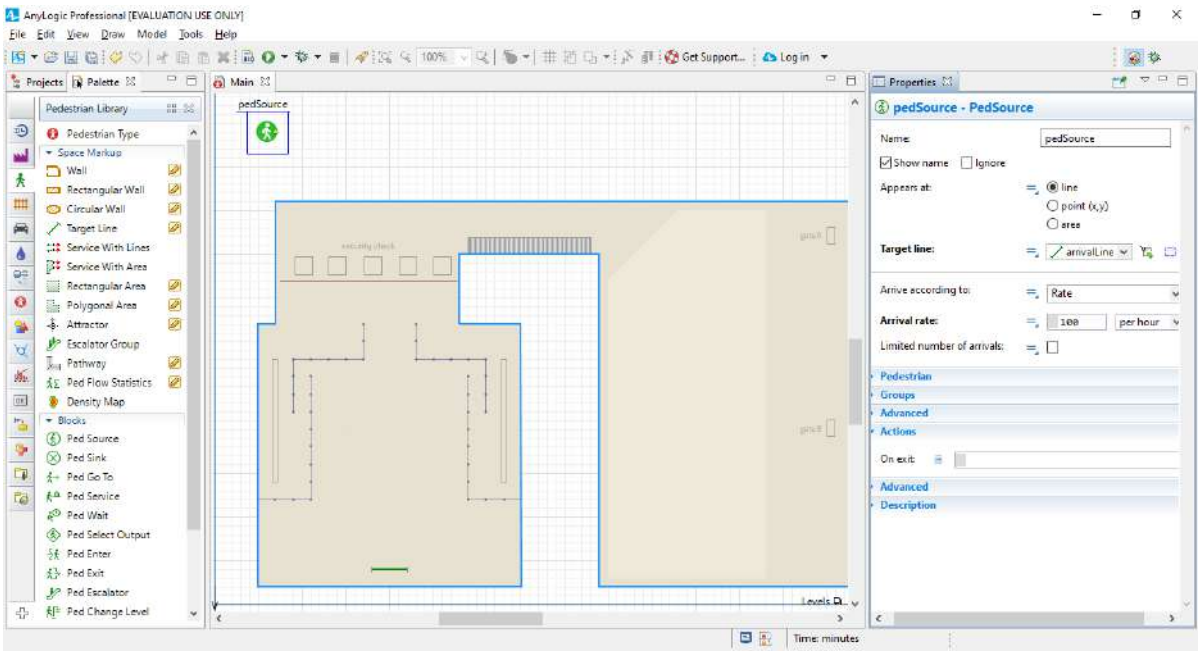
Define a second target line that passengers will move toward after they enter the airport, place it in the gate area as shown in the figure above, and then name it gateLine1.



Start by dragging the PedSource block from the Pedestrian Library palette on to our Main agent type's diagram.

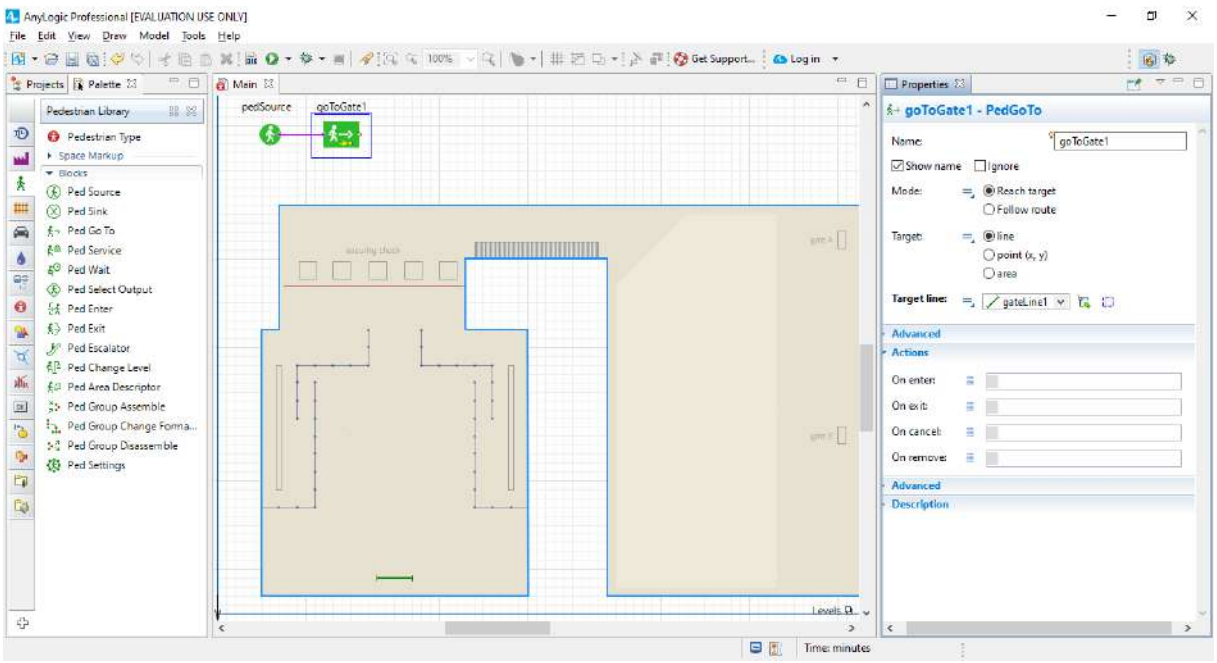
Since we want passengers to arrive randomly at an average rate of 100 passengers per hour, go to the pedSource block properties and then type 100 in the Arrival rate box.

Specify the location where the passengers appear in the simulated system by clicking arrival Line in the Target line list.

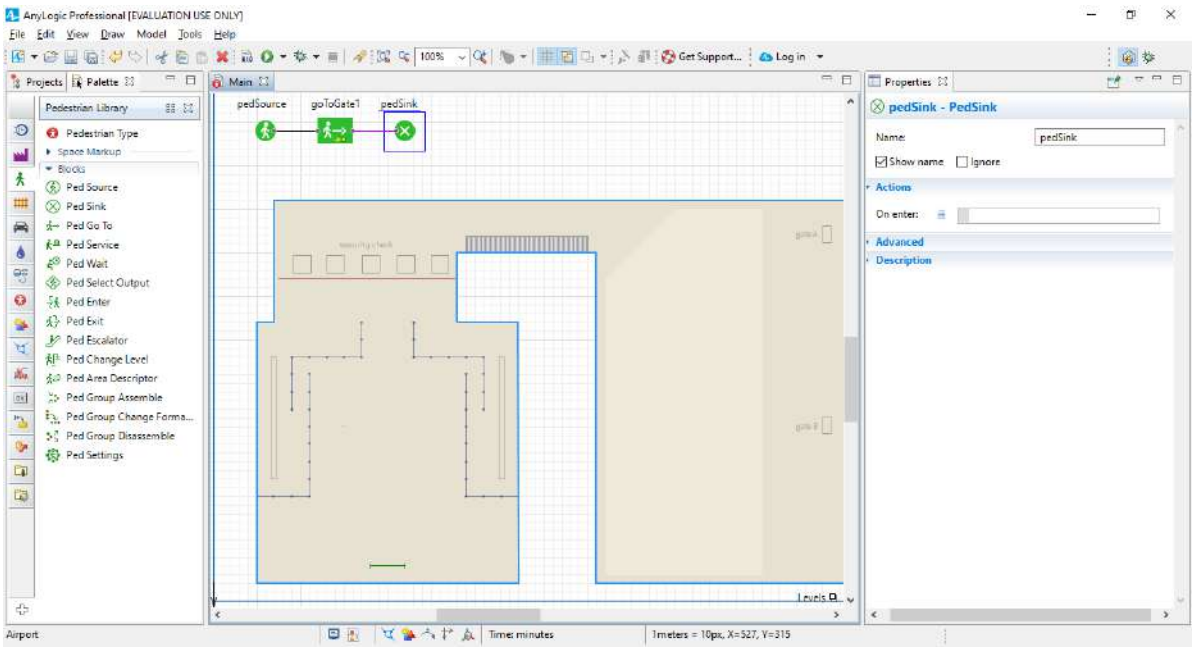


Add a PedGoTo block to simulate pedestrian movement to the specified location and then connect it to pedSource. Since we want our passengers to go to the gate, name the object goToGate1.

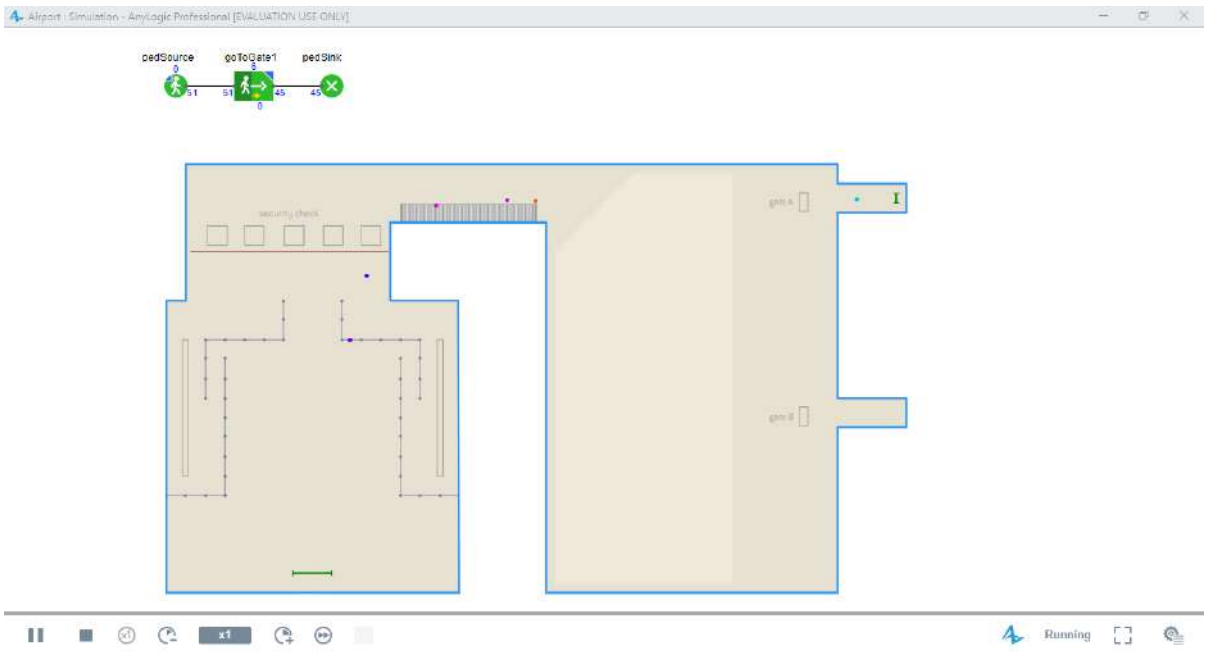
Specify the movement destination by selecting gateLine1from the Target line combo box.



Add a PedSink block to discard incoming pedestrians. Pedestrian flowcharts typically start with a PedSource block and end with a PedSink block.

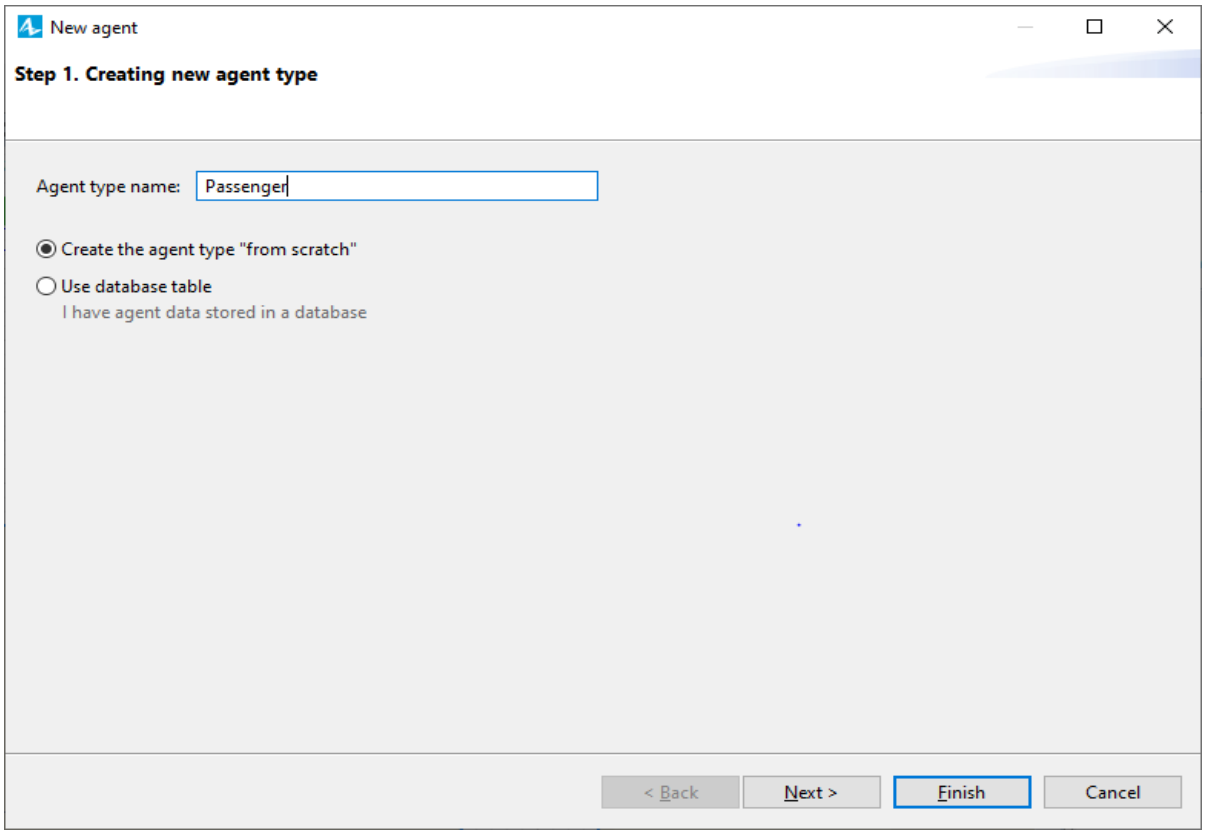


Run the model. In the 2D animation, you'll see the pedestrians move from the airport entrance to the gate.

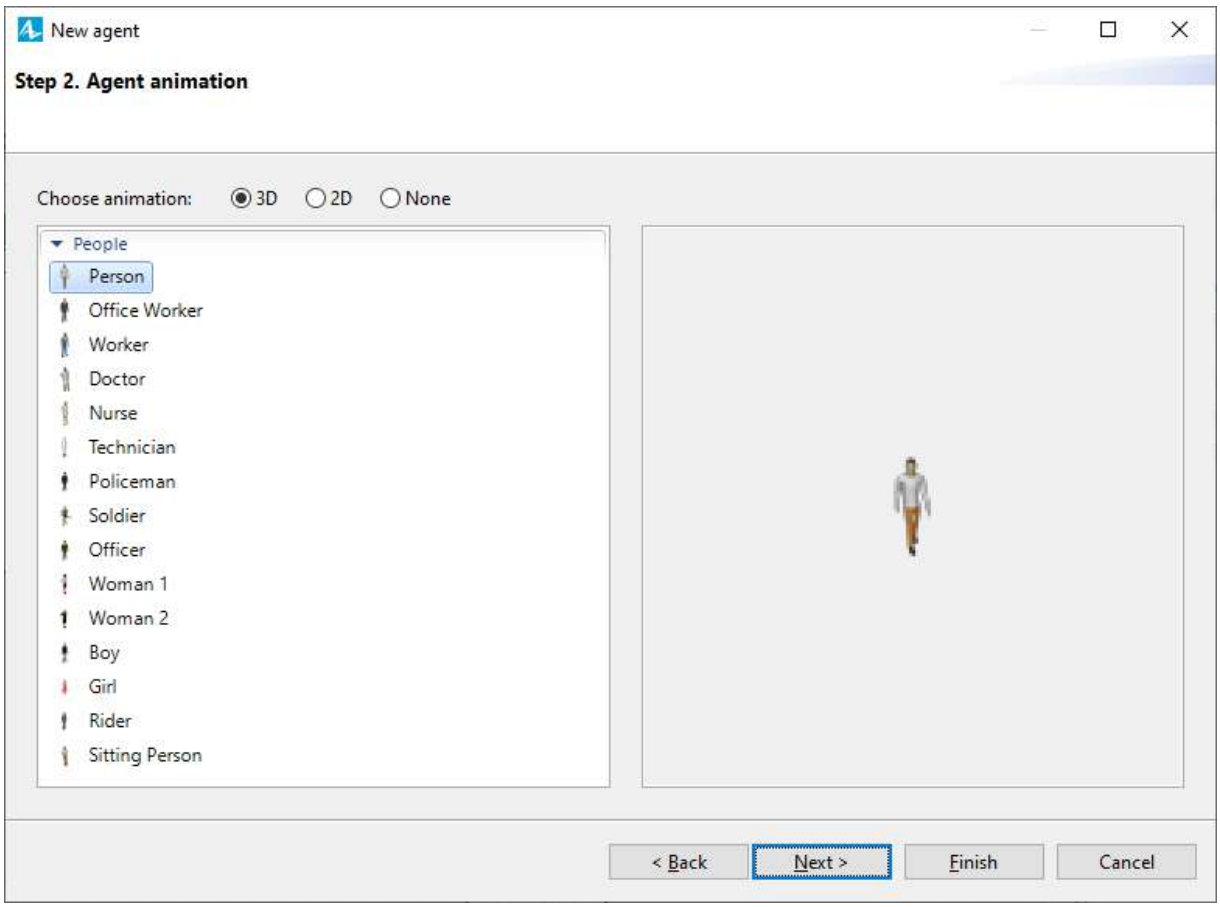


Phase 2. Drawing 3D animation

From the Pedestrian Library palette, drag the Pedestrian Type element on to the Main diagram.

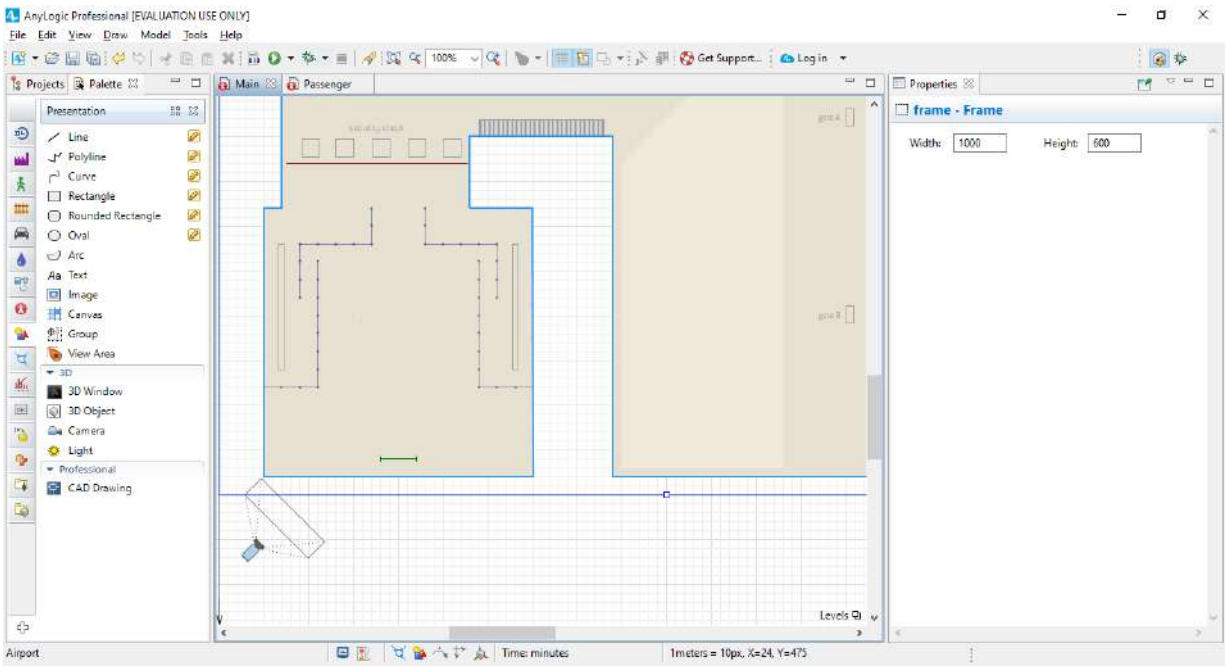


In the New agent wizard, enter the new pedestrian type's name –Passenger–and then click Finish.



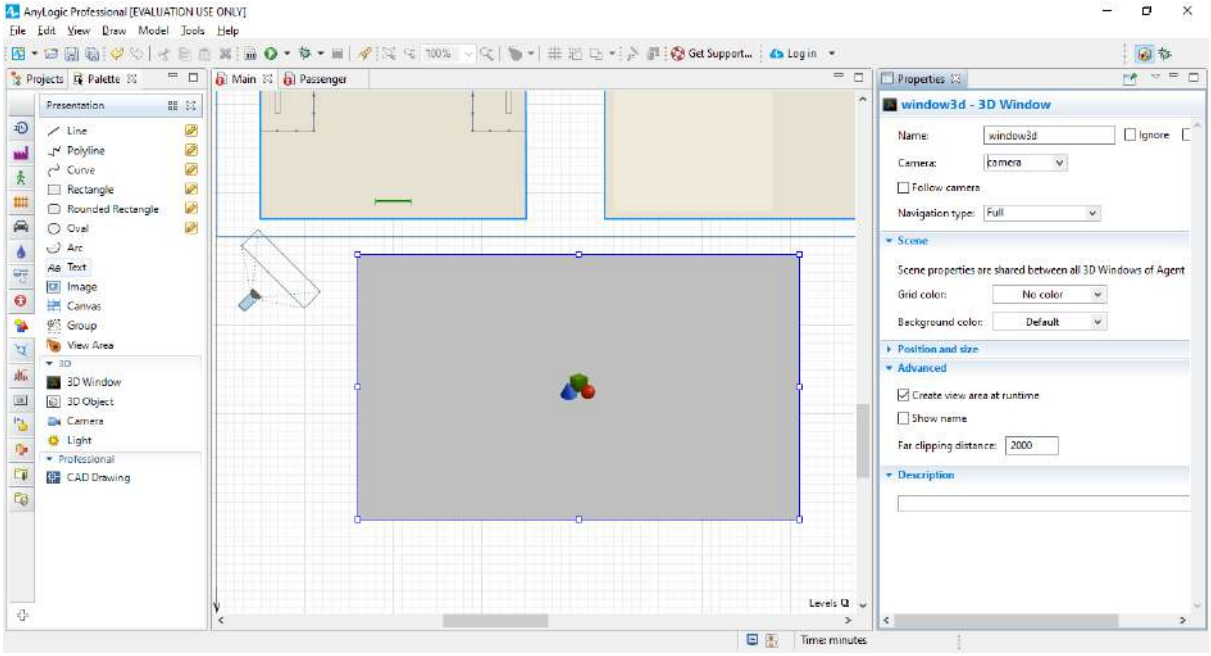
After the Passenger diagram opens, return to the Main diagram.

From the Presentation palette, drag the Camera on to the Main diagram and place it so it faces the terminal.

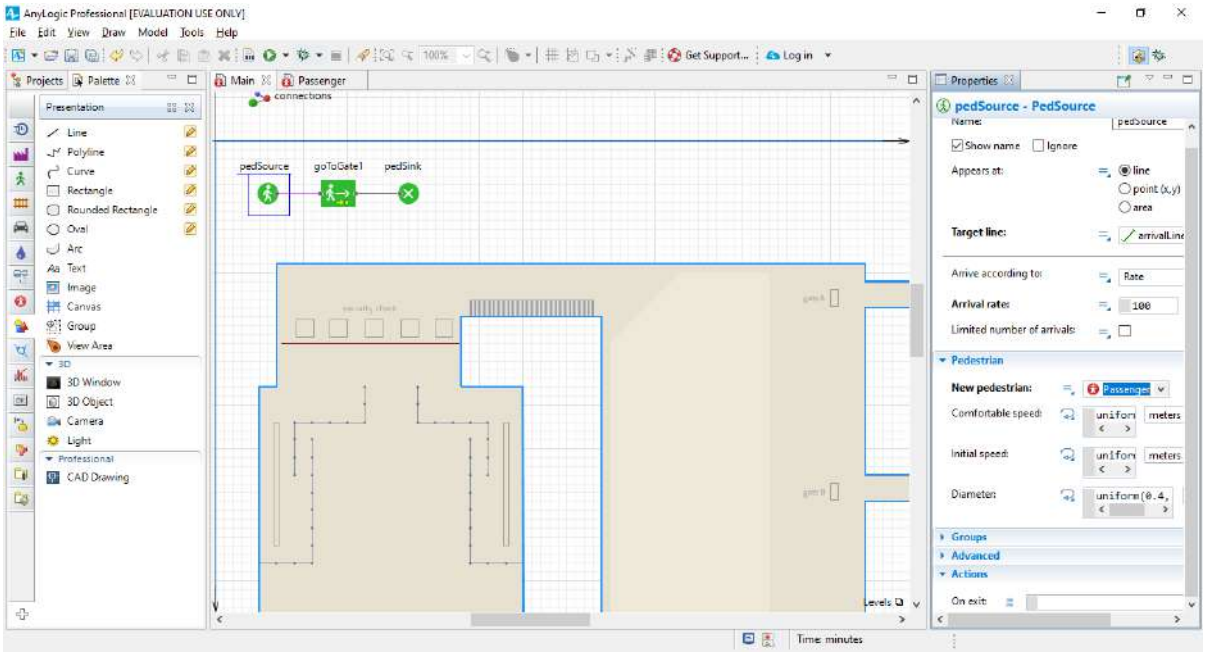


Drag the 3D Window on to the Main diagram and place it below the terminal layout image.

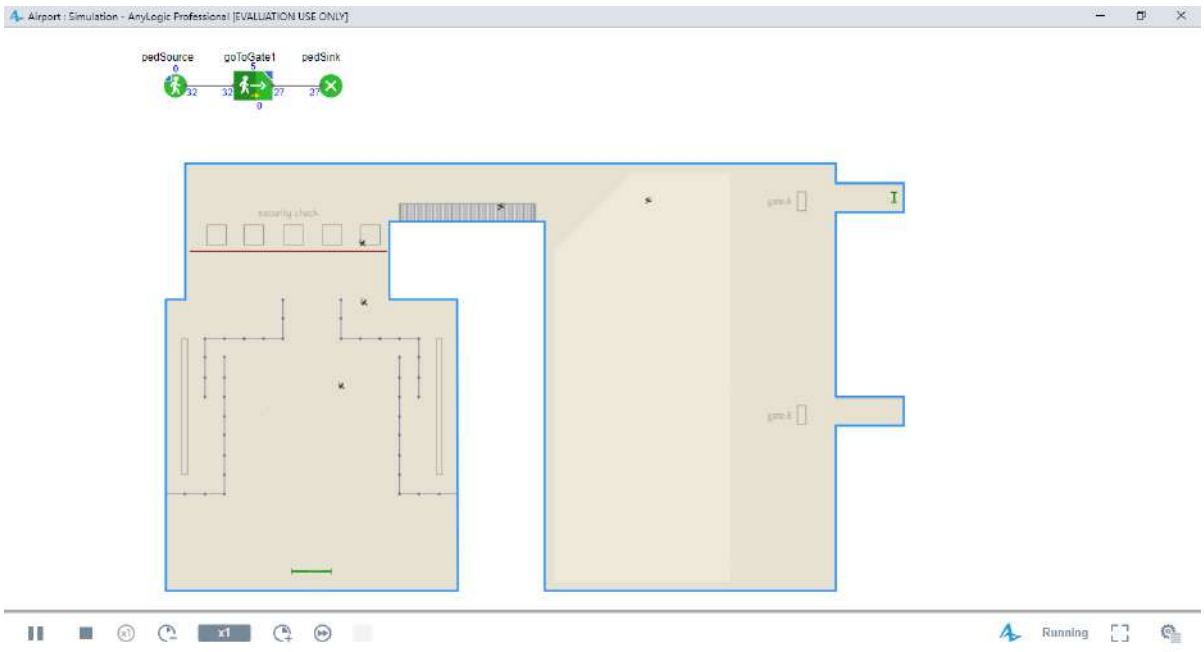
Open the 3D Window properties, and then select camera from the Camera list.

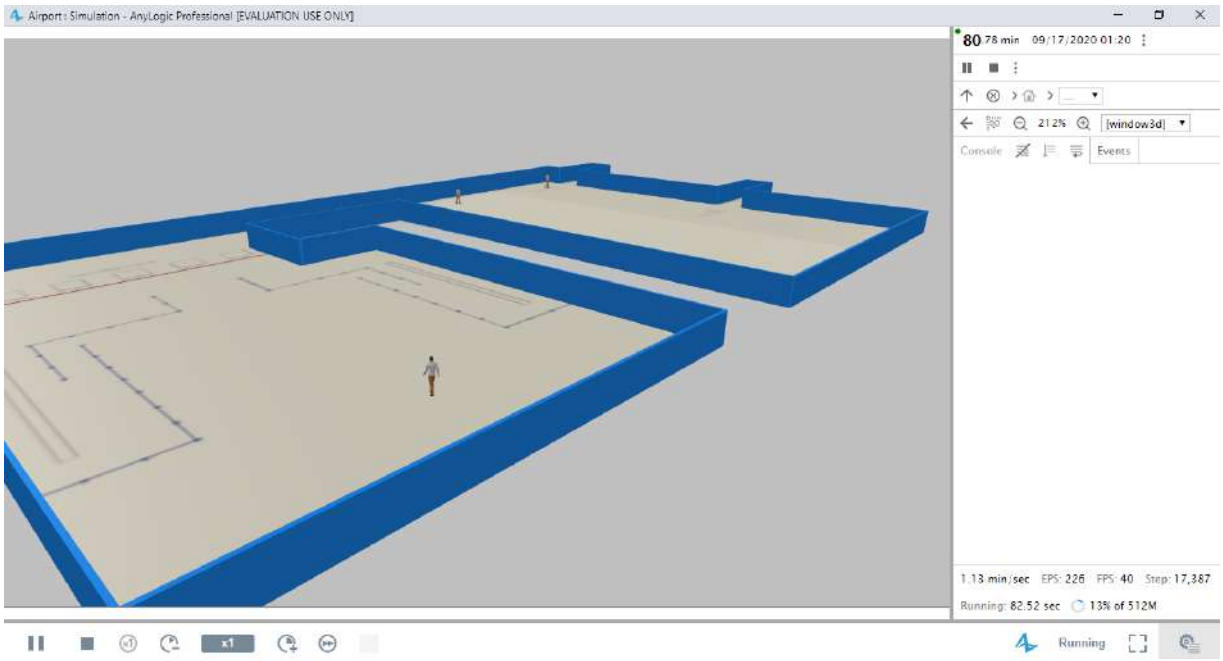


We want our flowchart block pedSource to create pedestrians of our custom Passenger type. Open the pedSource properties, and then select Passenger from the New pedestrian box in the Pedestrian section.

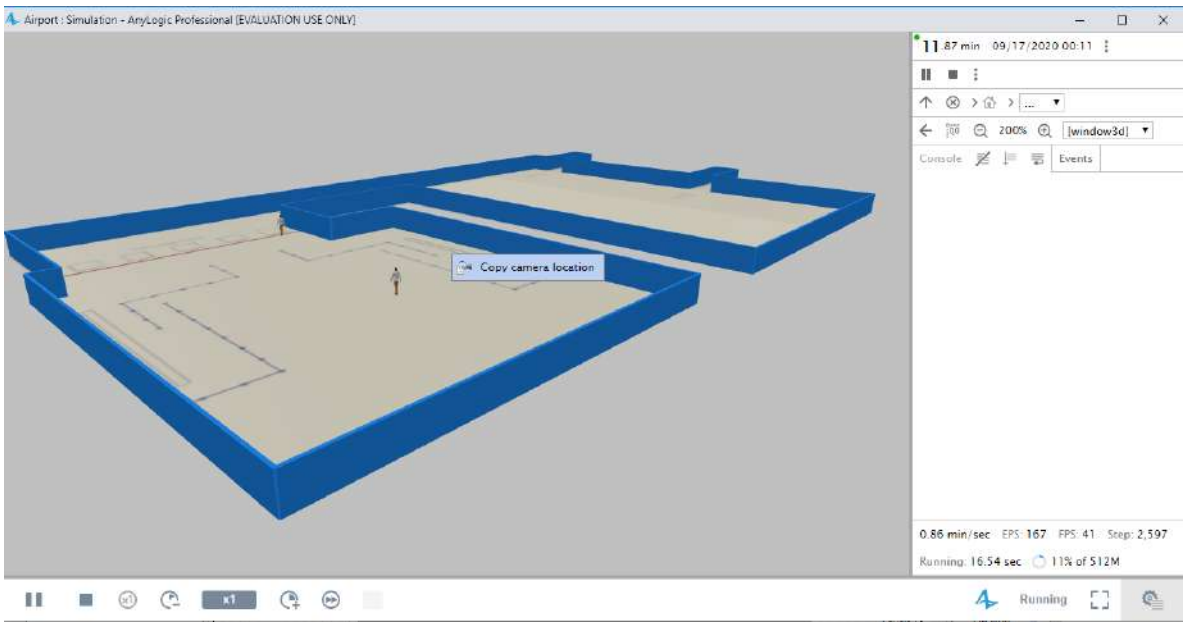


Run the model, and you'll see pedestrians move from the entry to the gate inside the building. You can switch to a 3D view by clicking the toolbar's Navigate to view area... button and then selecting [window3d] from the list.



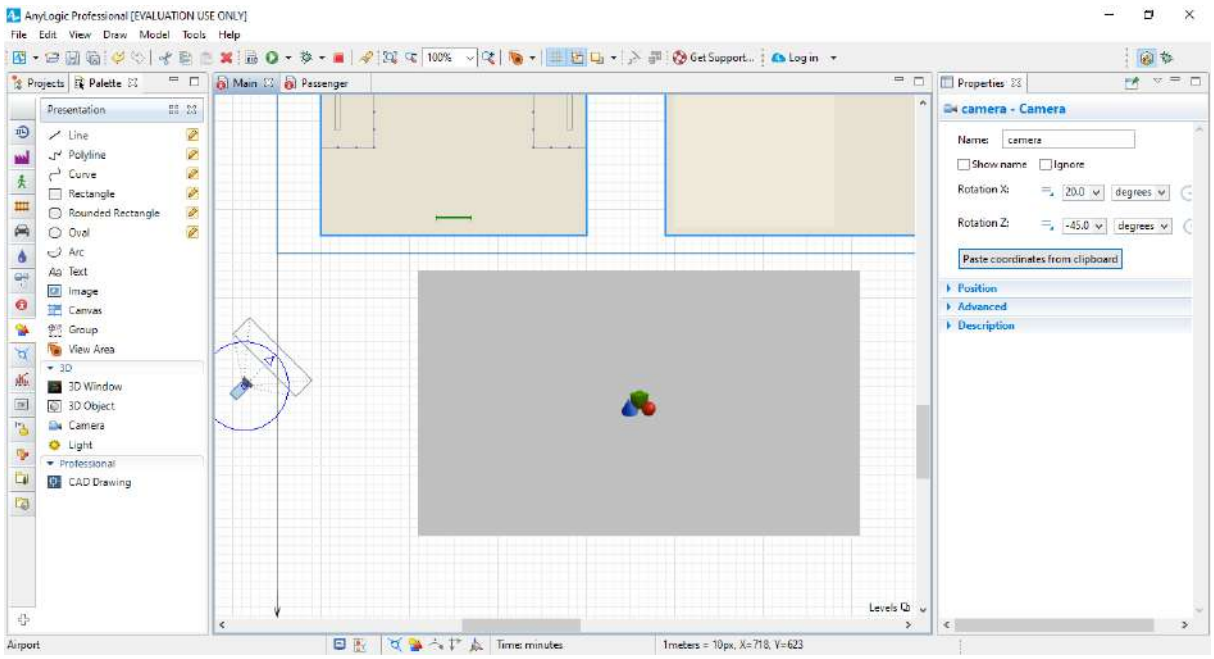


Navigate the scene to get the best view, right-click inside the 3D scene, and then click Copy the camera's location.

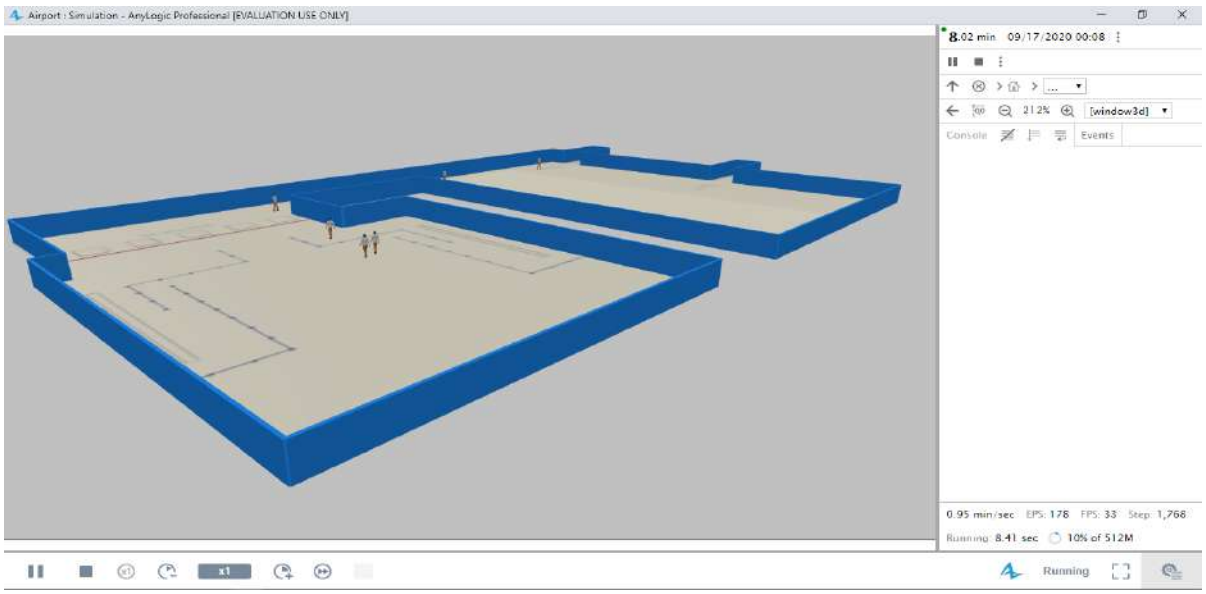


Close the model's window, open the camera's properties, and then apply the optimal camera you selected during the previous step by clicking Paste coordinates from clipboard.

If you can't locate the camera, use the Projects tree to locate it. You'll find it under the Presentation branch of the airport's Main agent.



Run the model a second time and view the 3D view that the new camera position provides.



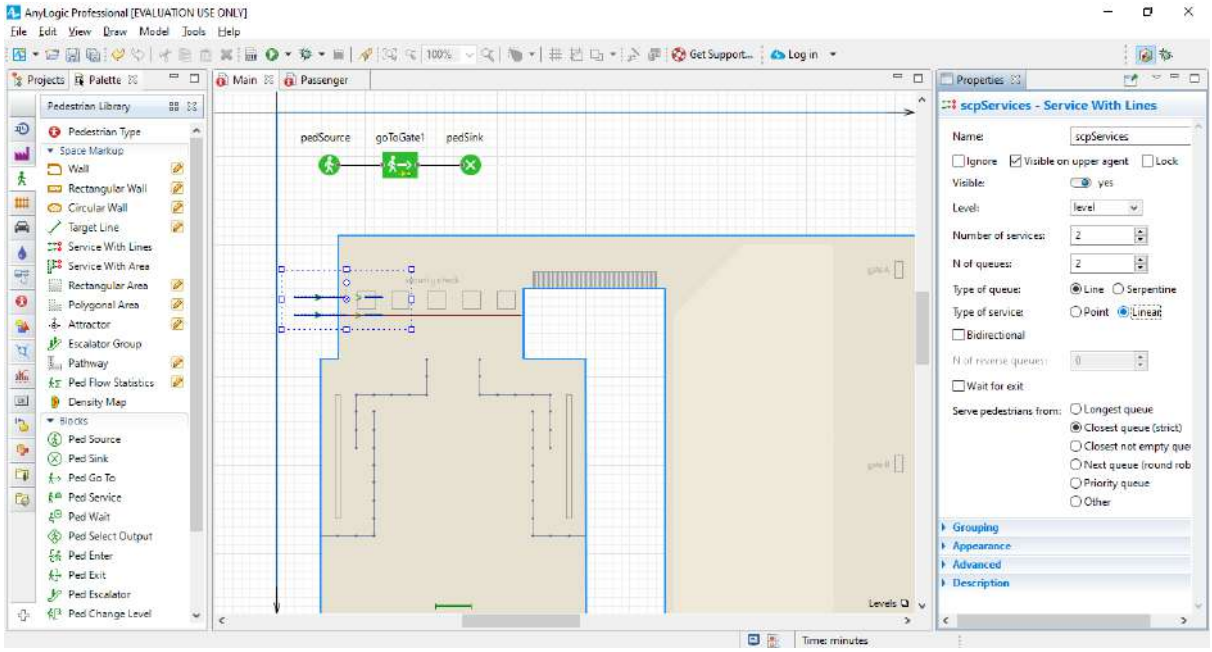
Phase 3. Adding security checkpoints

In this third phase, we'll start modeling the processes that take place inside the airport by adding security checkpoints. All of the security checkpoints are service points.

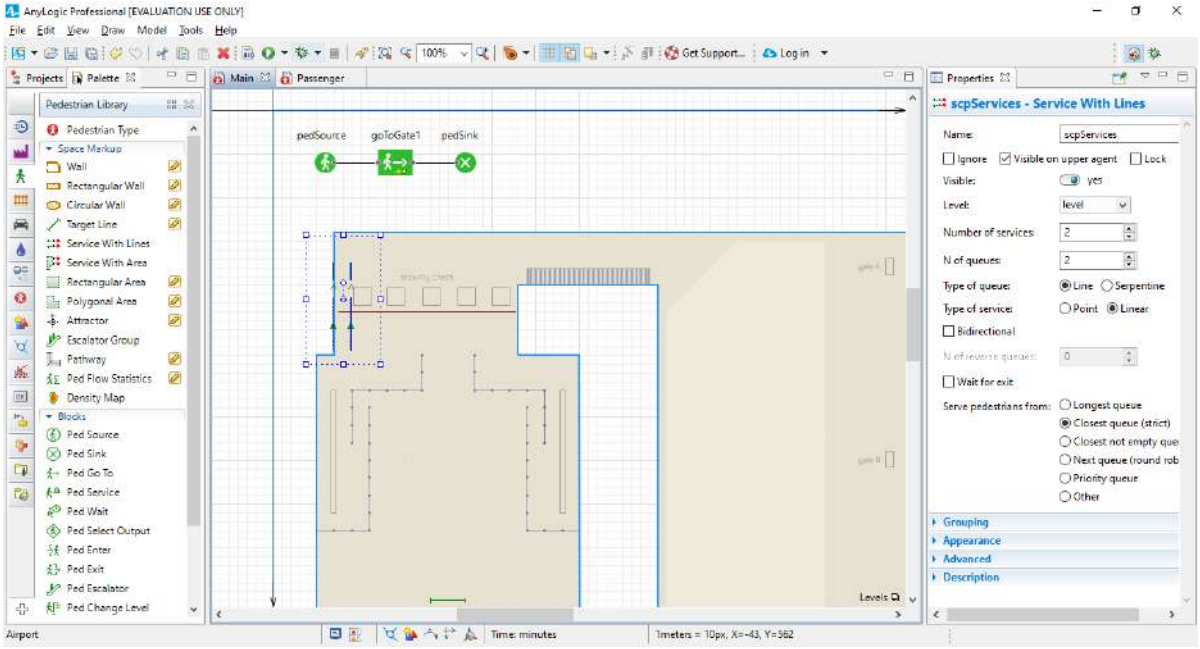
Drag the Service with Lines element from the Pedestrian Library palette on to the terminal layout. By default, a service will have two service points and two queue lines that lead to the service points.

Open the Service With Linesproperties area, use the Namebox to name the shape scpServices- in this case, "scp" stands for security checkpoints -and then change the Type of serviceto Linear.

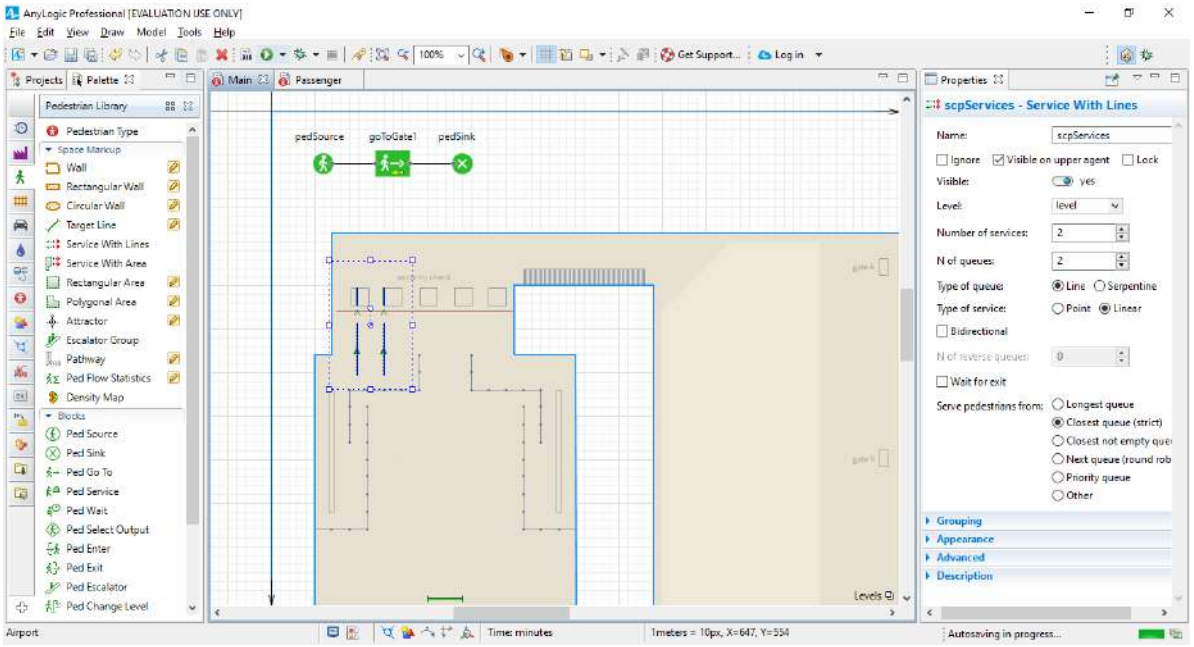
After you change the service type from a point service to a linear service, the service shapes will change from points to lines



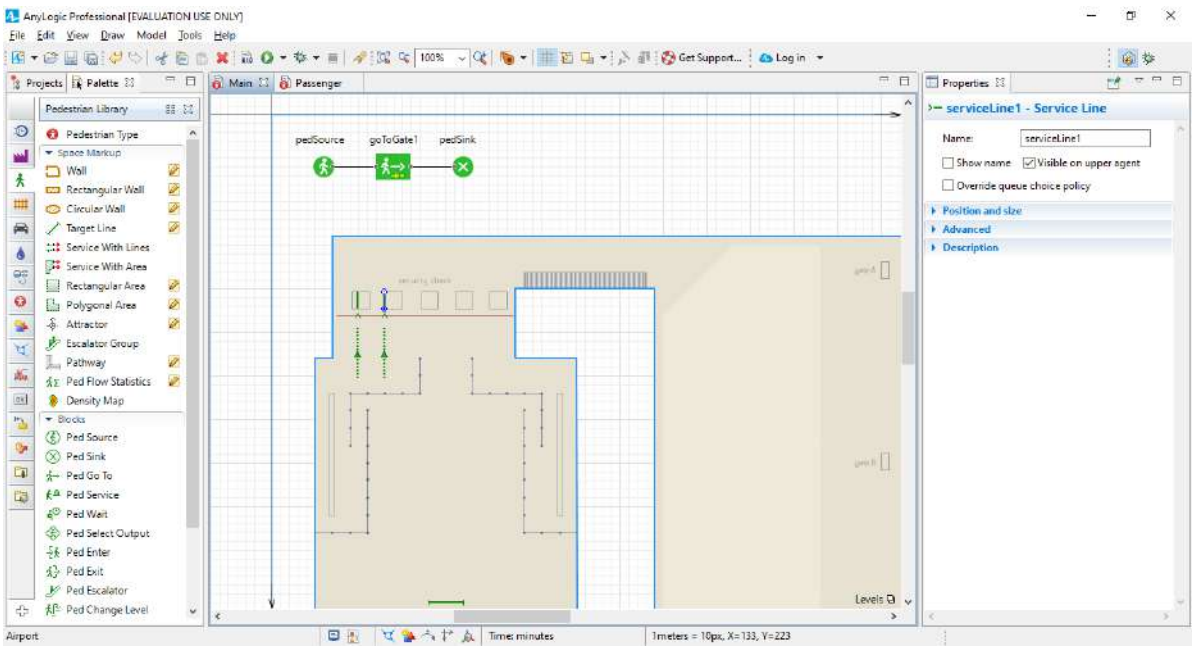
Use the round handle above the shape's center to rotate the service.



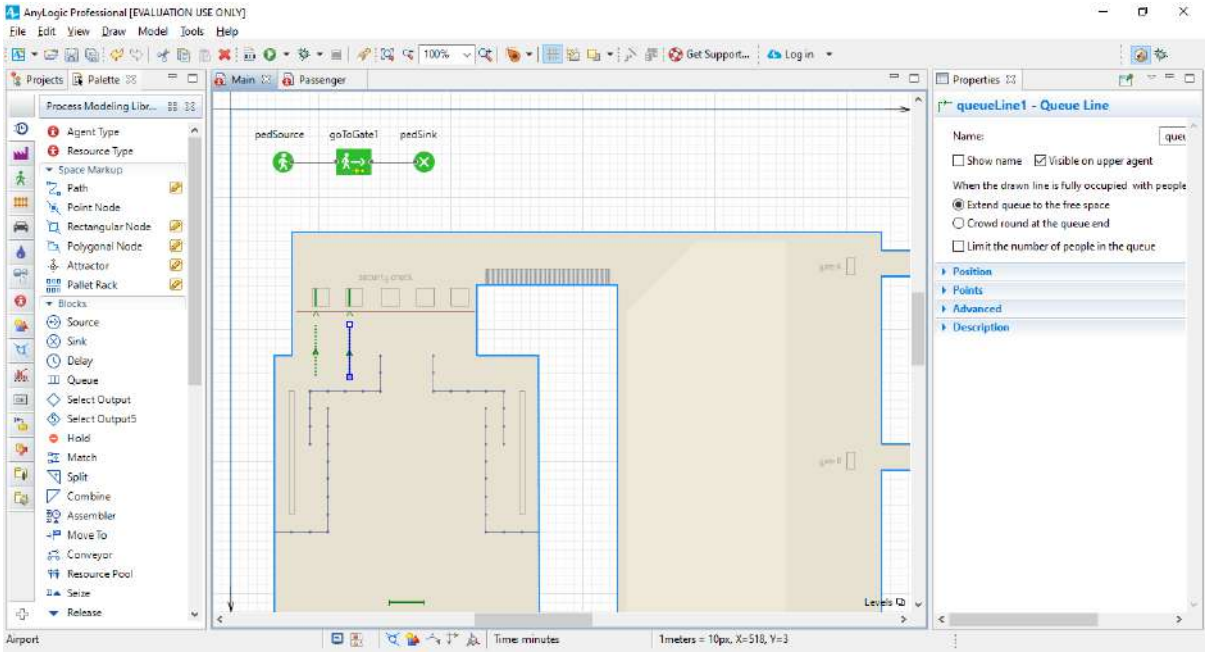
Move the service in a way that ensures the first linear service crosses the rectangle that represents the metal detector frame.



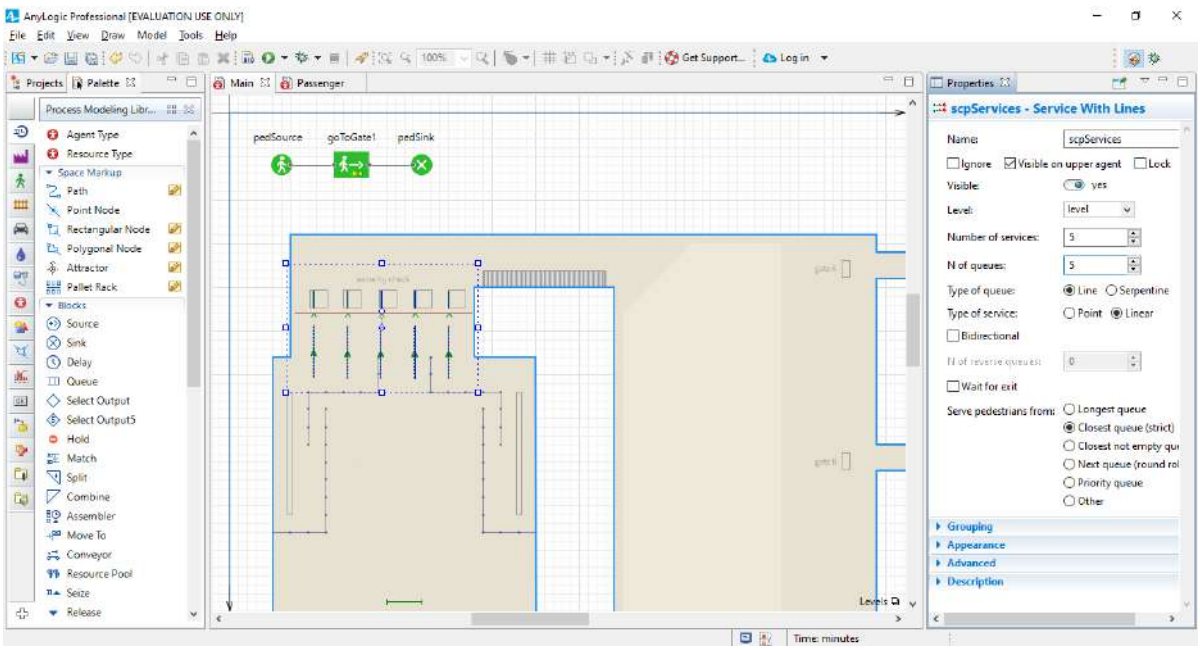
Select the next service line.



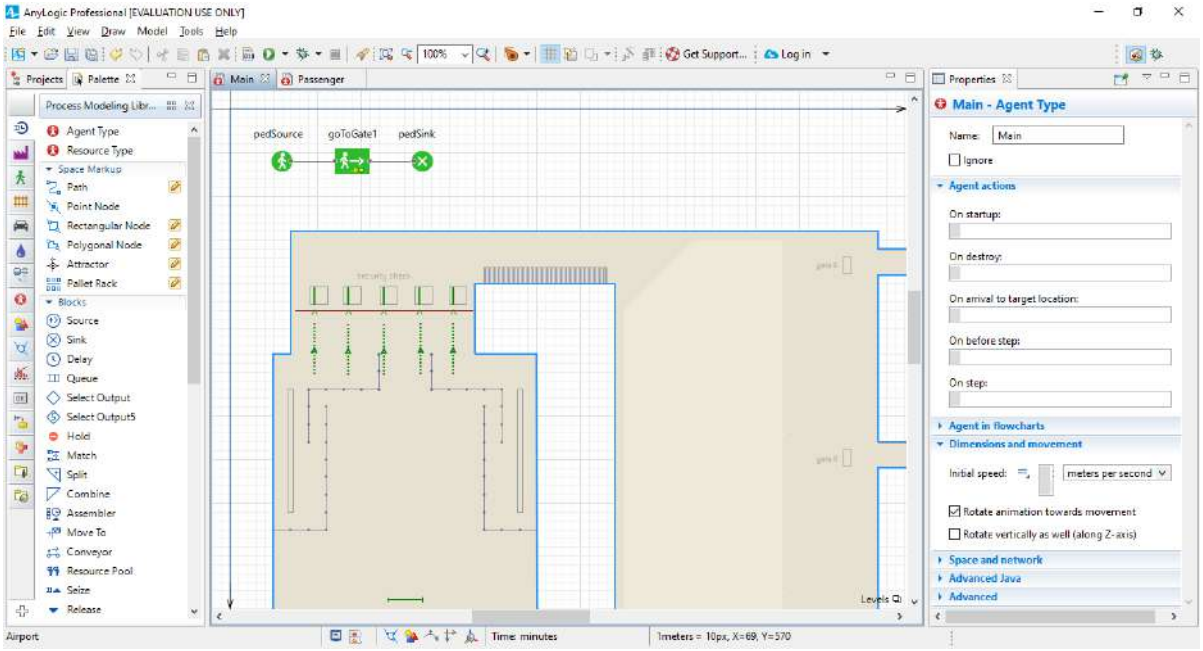
Accurately place the service line on top of the second security checkpoint placeholder and then adjust the queue location.



Navigate to the Service with Lines shape's properties and then change both the Number of services and Number of lines to 5.



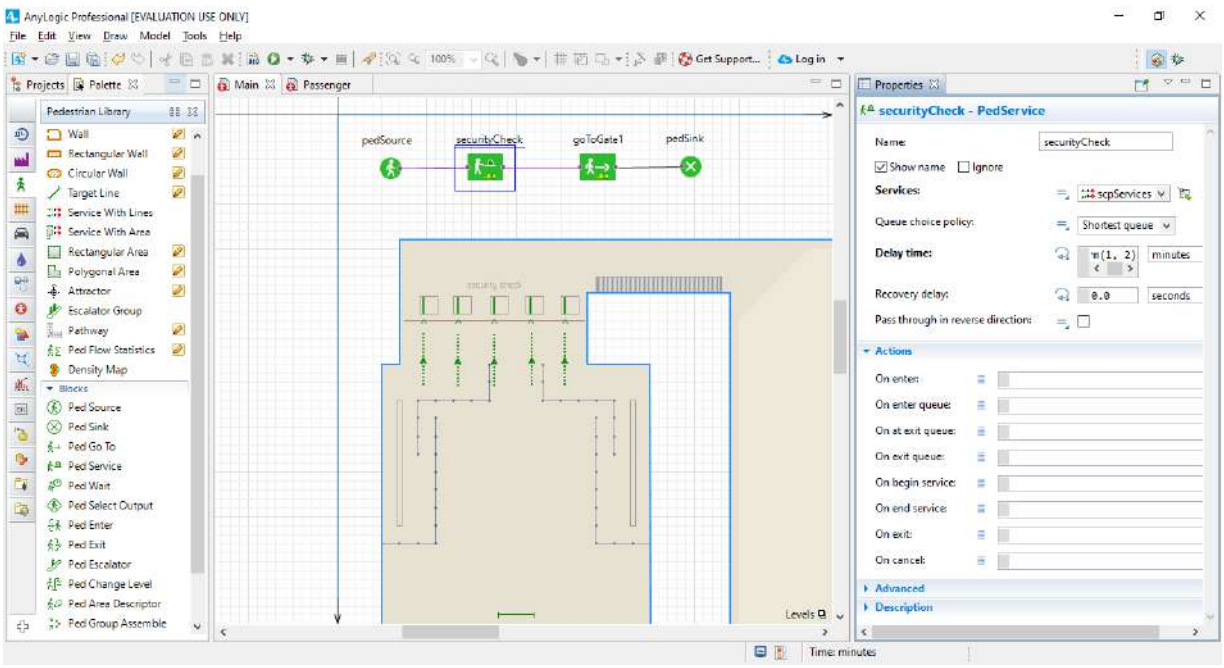
If necessary, adjust the new service and queue lines. After you've completed this step, the service shapes should look like those inthe figure below.



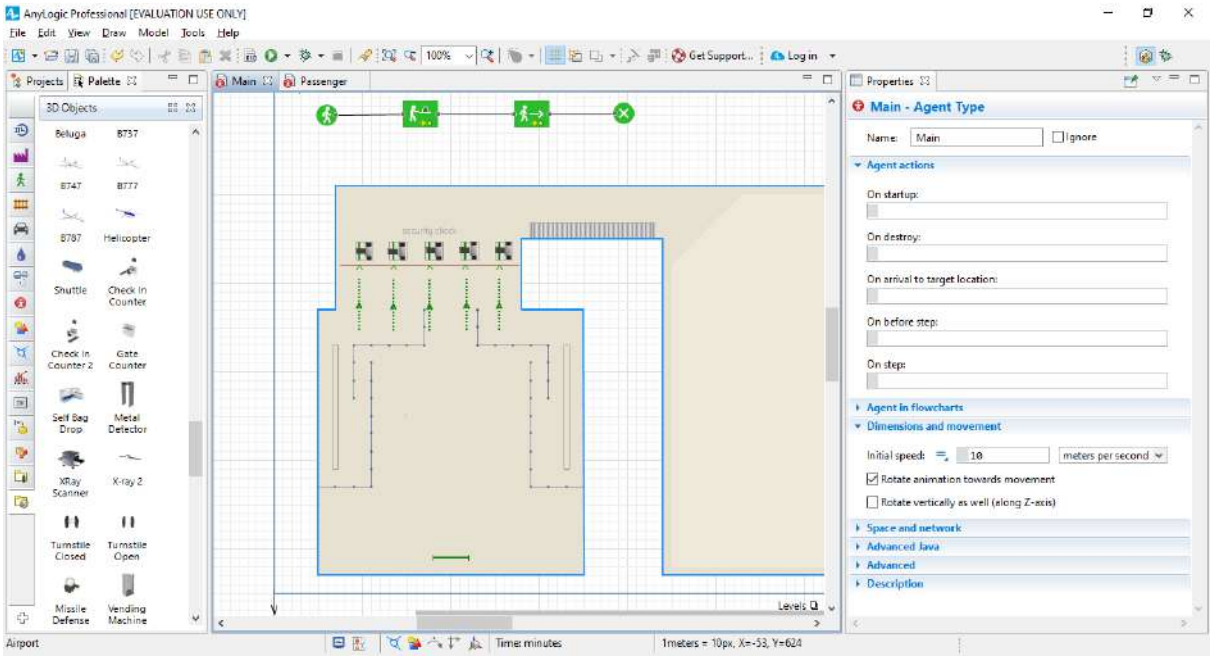
Add the PedService block on to the flowchart between the PedSource and PedGoTo blocks to make pedestrians pass through the service we defined using the referenced Service with Lines shape, and then name it securityCheck.

Go to the securityCheck block's properties. Select the services scpServices as Services.

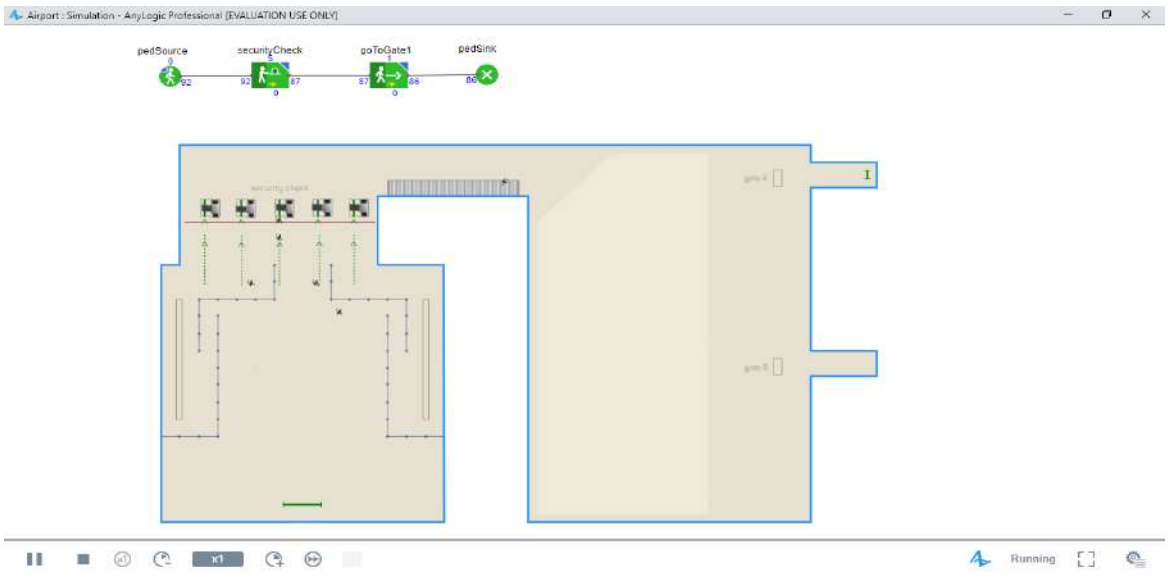
Since we assume it takes between 1 to 2 minutes to pass through the security checkpoint, type uniform(1, 2)minutes as the Delay time.

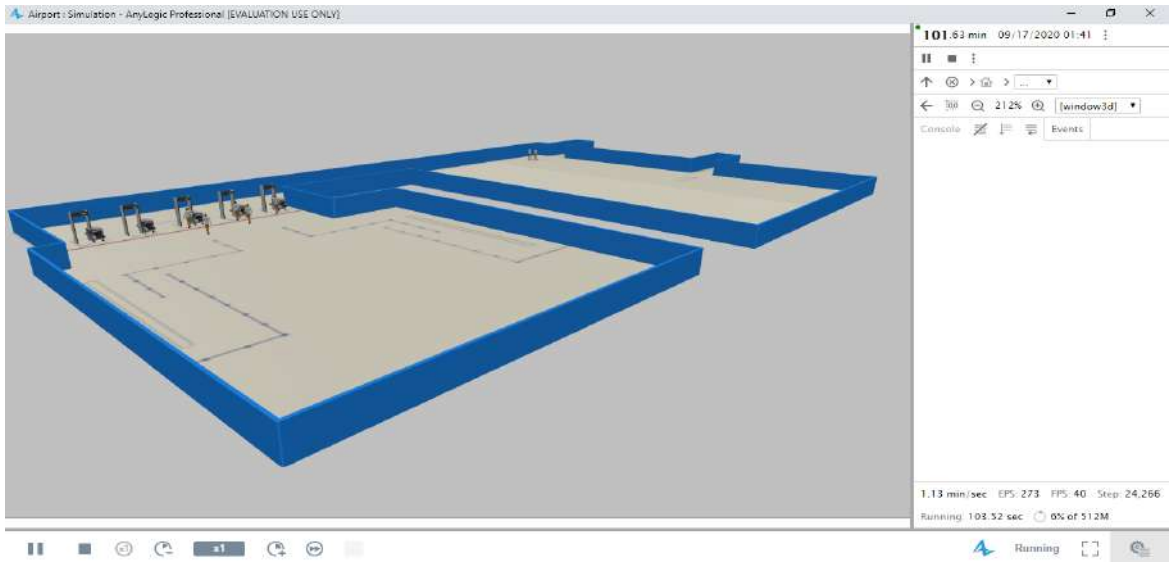


Now let's add 3D models of the security checkpoints. Using the 3D Objects palette, Airport section's Metal Detector and XRay Scanner elements, draw five security checkpoints. Change the Scale of XRay Scanner shapes to 75%.



Run the model. You'll see that passengers are now scanned at the security checkpoints.

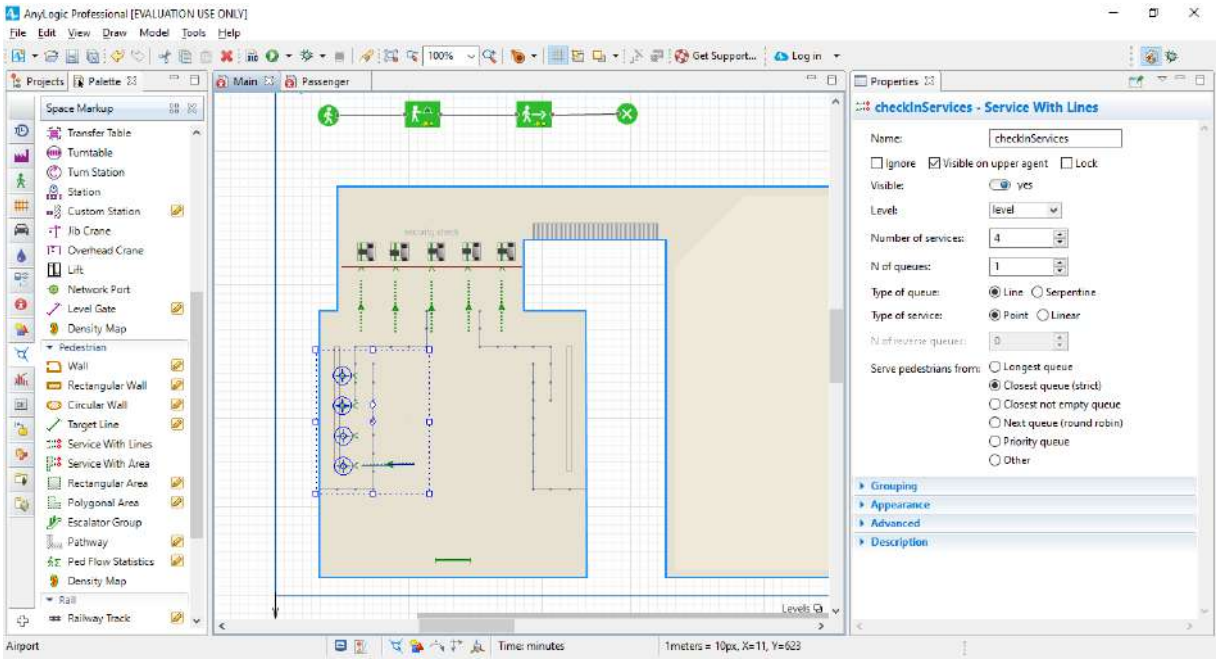

Airport : Simulation - AnyLogic Professional (EVALUATION USE ONLY)
pedSource
securityCheck
goToGate1
pedSink
x1
Running


Airport : Simulation - AnyLogic Professional (EVALUATION USE ONLY)
101.63 min 09/17/2020 01:41
212%
[window3d]
Console
Events
1.13 min/sec EPS: 273 FPS: 40 Step: 24,266
Running: 108.52 sec 6% of 512M
x1
Running

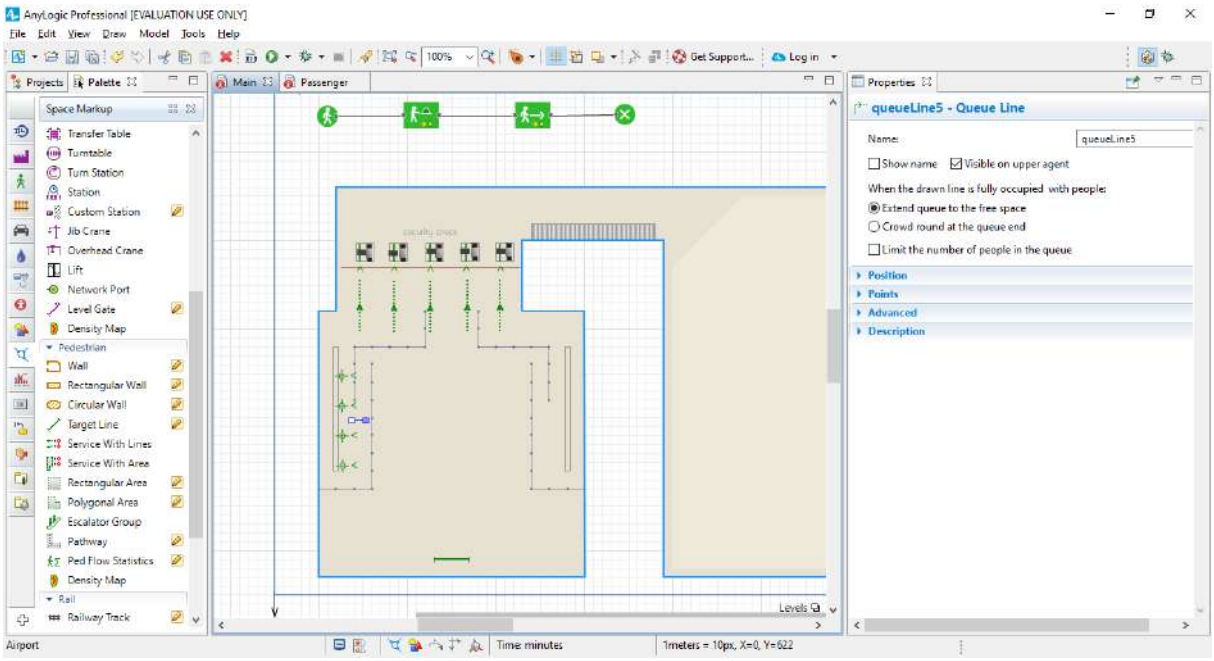

Airport : Simulation - AnyLogic Professional (EVALUATION USE ONLY)
32.11 min 09/17/2020 00:32
196%
[window3d]
Console
Events
7.19 min/sec EPS: 1,709 FPS: 38 Step: 7,895
Running: 16.53 sec 6% of 512M
Virtual
Running

Phase 4. Adding check-in facilities

Draw check-in locations with another Service with Lines shape. This time we'll need four service points, and one queue. Name the services checkInServices.

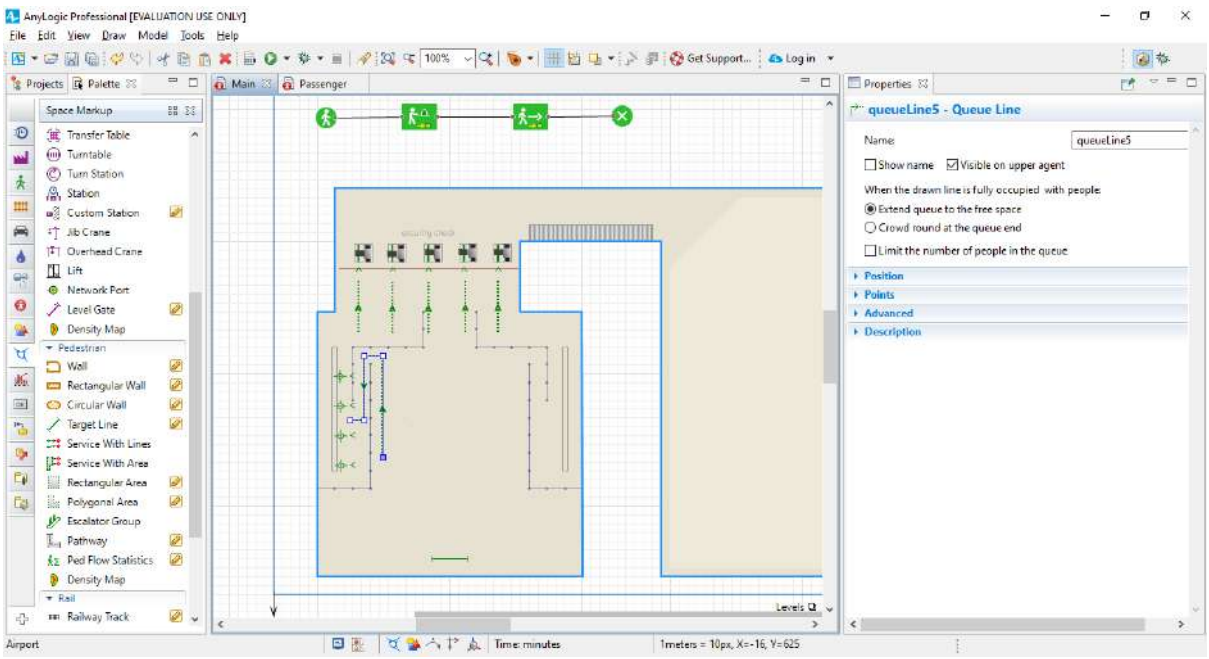


Place the shape in the location shown in the figure below. To make the line look like what you see on the figure, move the line to the required location, and then place the end point where the line starts turning.



Add more salient points to the line. Right-click the queue line, choose Add points from the pop-up menu, and then click the line's end point from where you want to continue drawing the line.

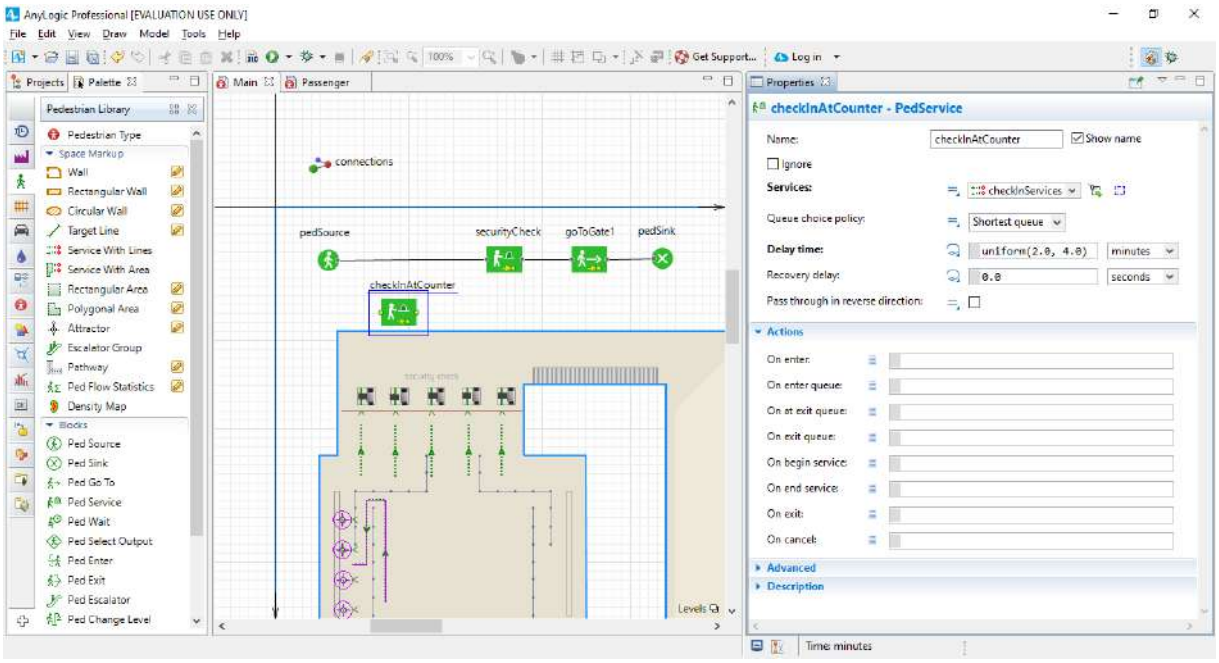
Add more points by clicking where you want to place the line's salient points. Finish drawing the line by double-clicking. Finally, you should get the queue line of the following form:



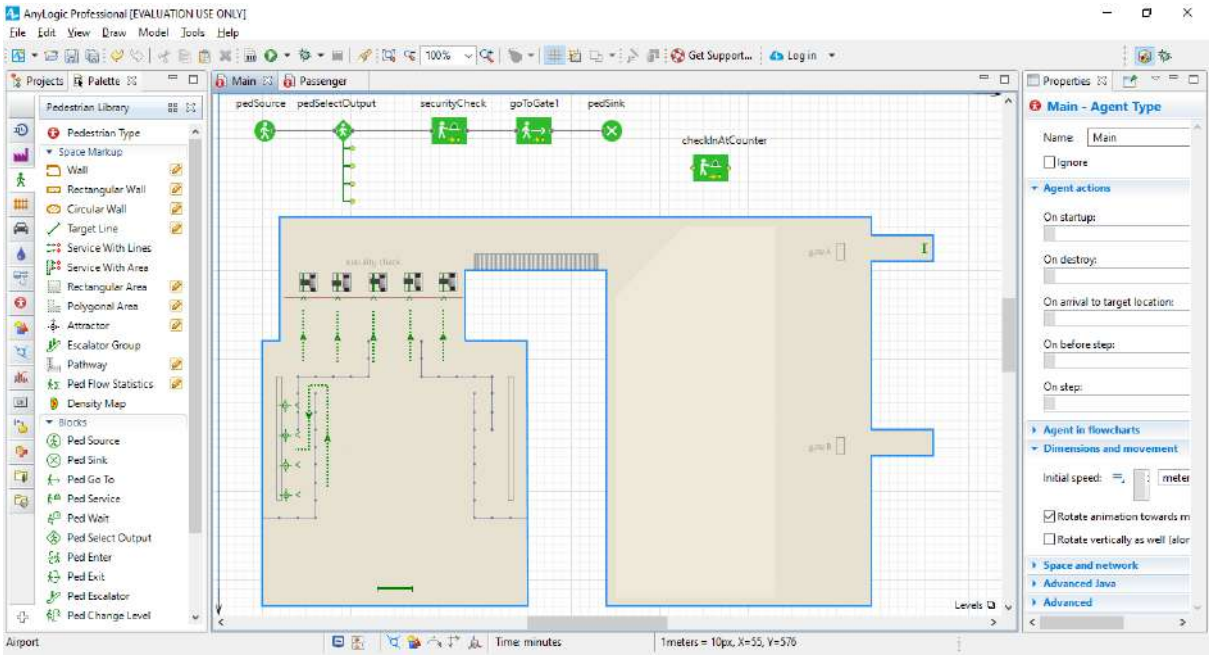
Add another PedService block and name it checkInAtCounter.

In the block's properties, select the space markup shape checkInServices as Services.

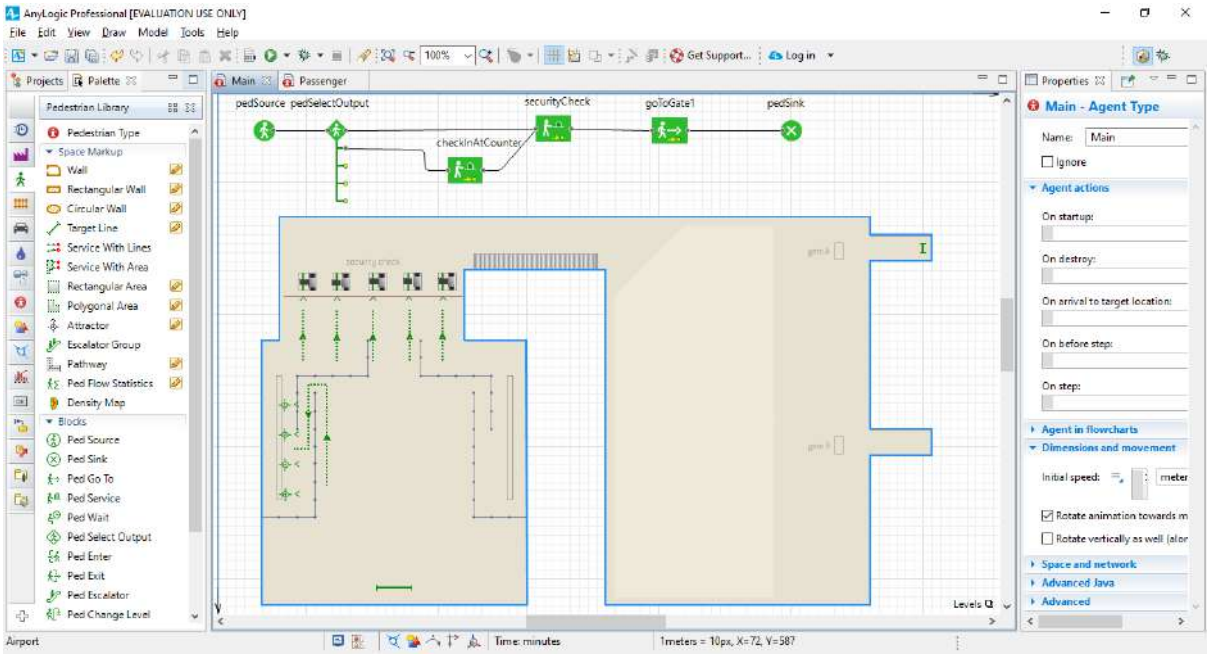
Since we assume it takes between 2 to 4 minutes to check in, type uniform(2, 4)minutes as the Delay time.



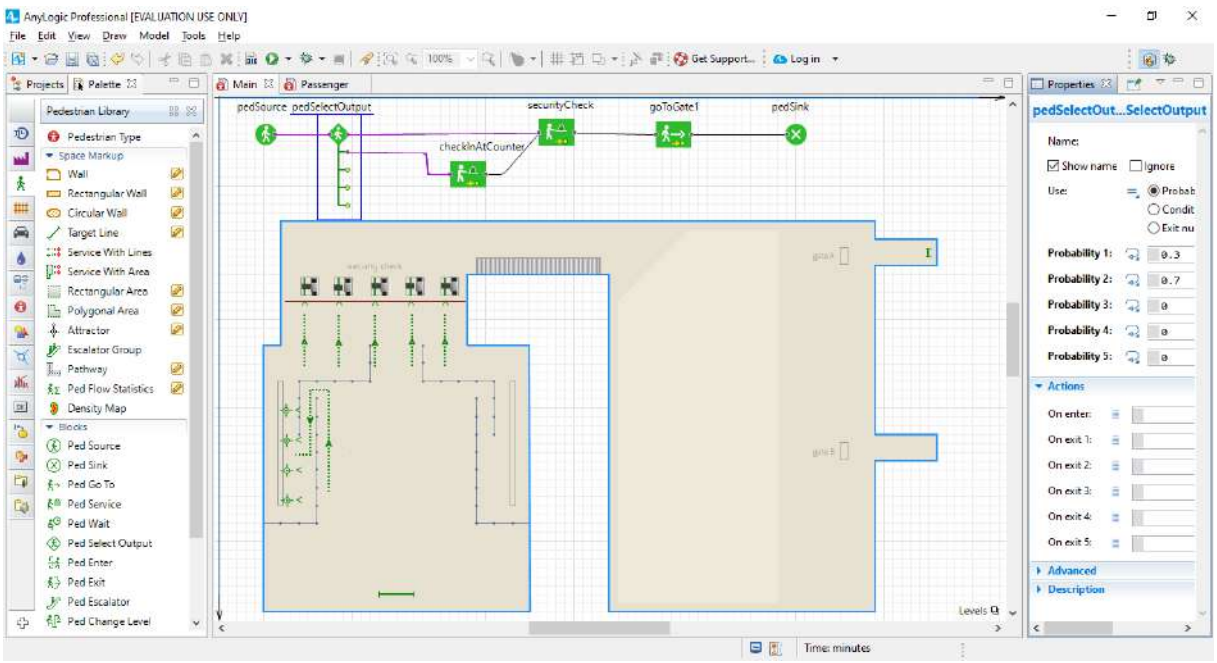
Add the PedSelectOutput block to route passengers to different flowchart branches.



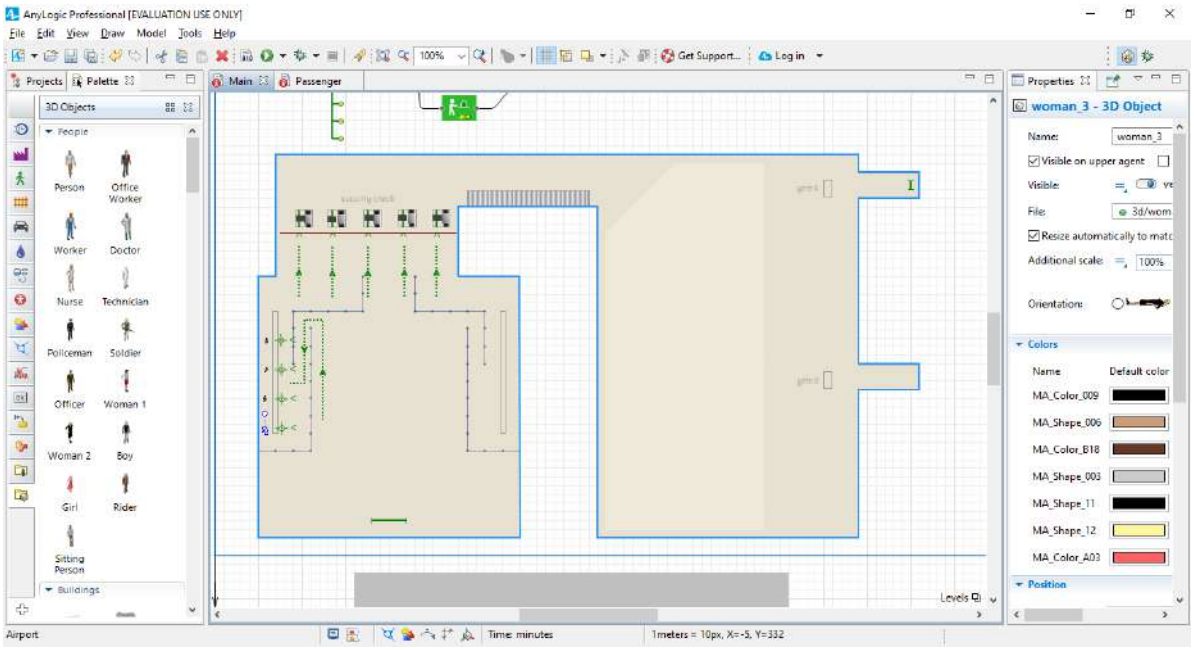
Connect the checkInAtCounter block to the existing flowchart blocks as shown in the figure below.



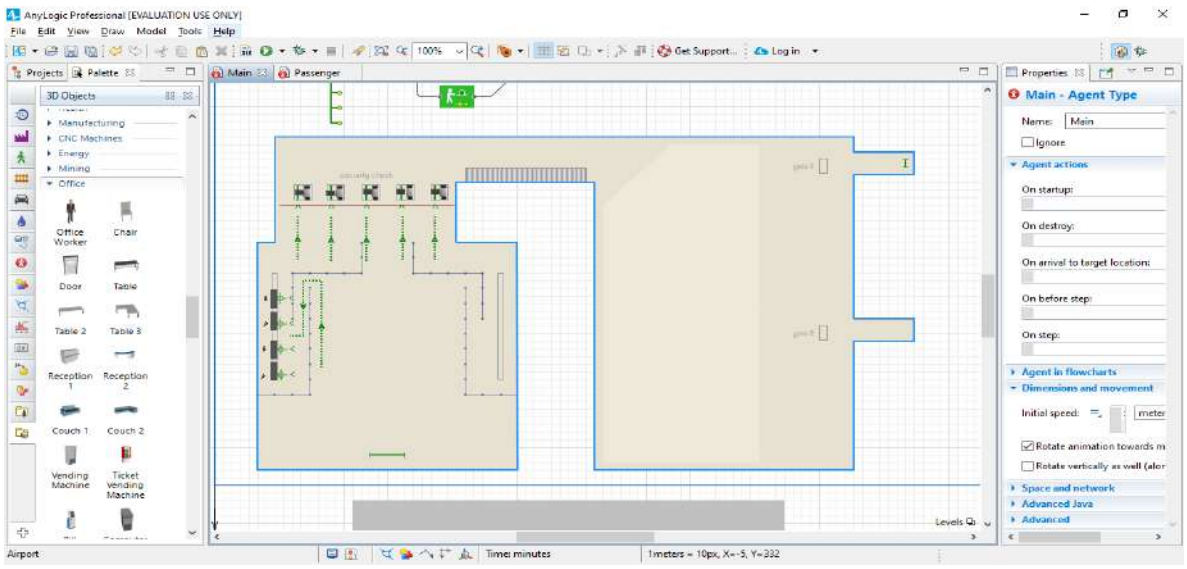
Since we're assuming 30 percent of our passengers will check in online and 70 percent will check in at the counter, we'll model this behavior by setting pedSelectOutput's Probability1: to 0.3 and Probability 2: to 0.7. This action will route 30 percent of the passengers to the upper flowchart branch and 70 percent of the passengers to the lower branch. You must set Probability 3, Probability 4, and Probability 5 to 0 to prevent AnyLogic from routing passengers to the block's lower three output ports.



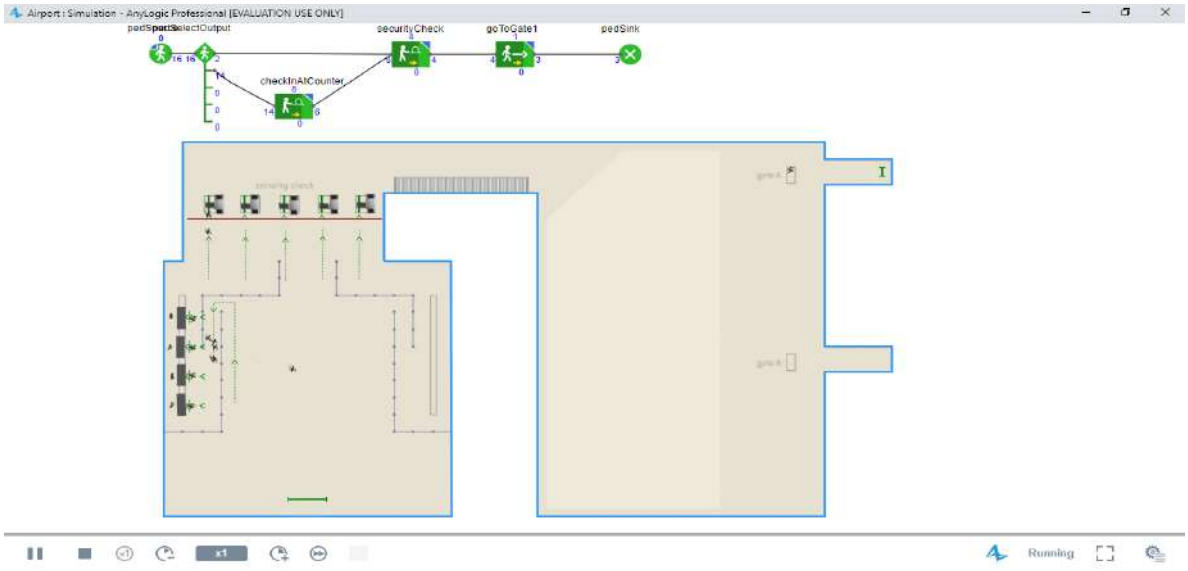
Let’s add ready-to-use 3D models to the airport’s check-in area. On the Palette's 3D Objects tab, expand the People section, and then add two copies of both Office Worker and Woman 2.to the diagram.

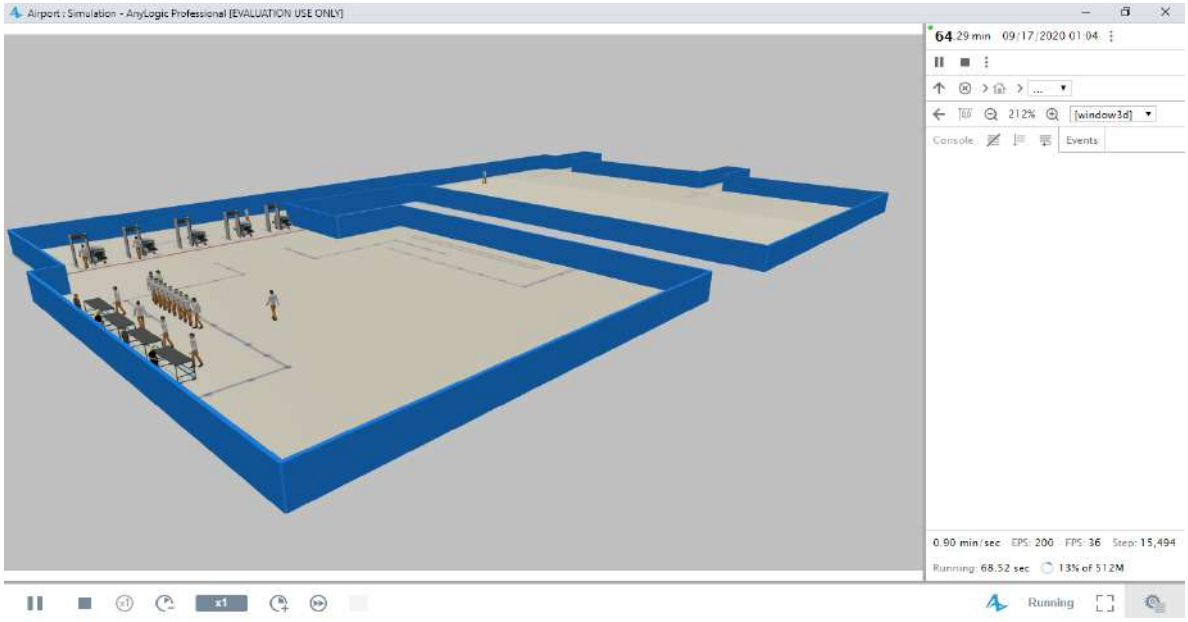


On the Palette’s 3D Objects tab, select the Office section, and then drag four copies of the Table object on to the diagram. Since the tables aren’t facing the correct direction, use their Properties section’s Position area to set Rotation, Z: 90 degrees



Run the model. You'll see some passengers check in and then go through the metal detector.

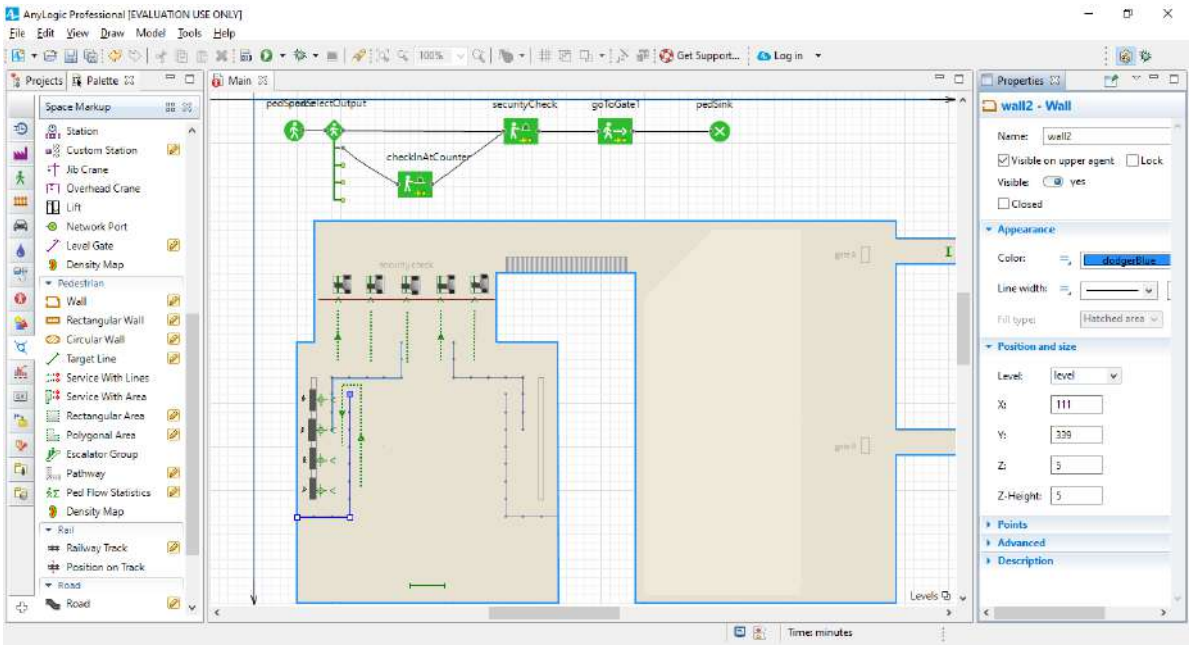




Now we'll add belt barriers that are common in areas where passengers check in for their flights.

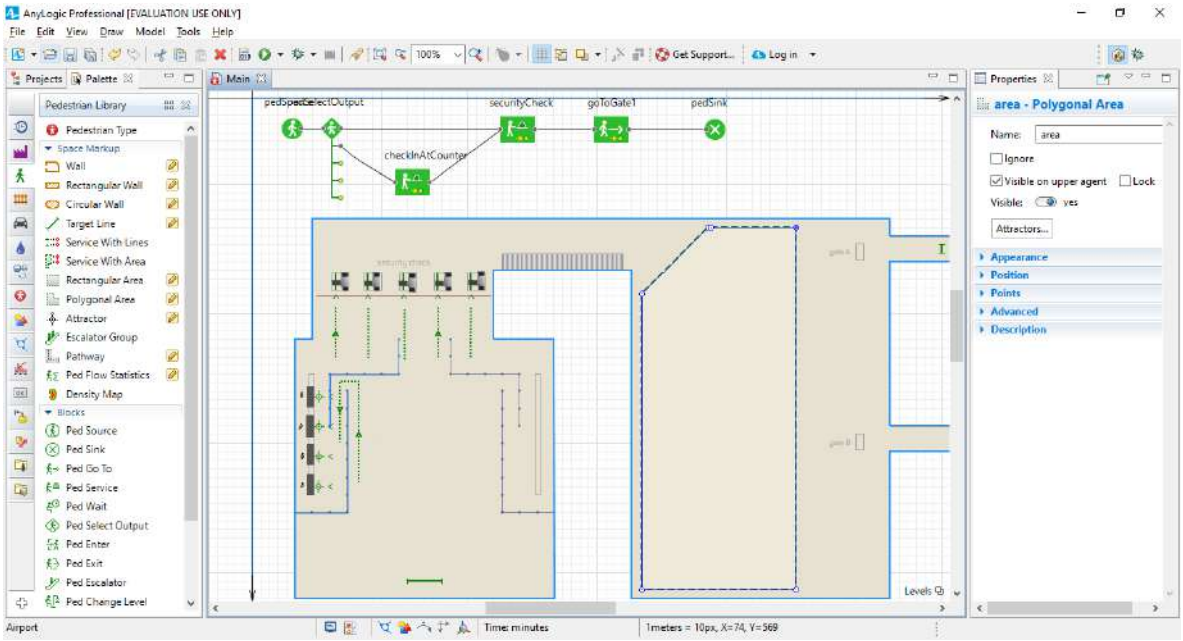
Use the Wall space markup element to draw two belt barriers.

Change the wall's Color to dodgerBlue, the Line width to1, Z: 5 and Z-Height: 5.



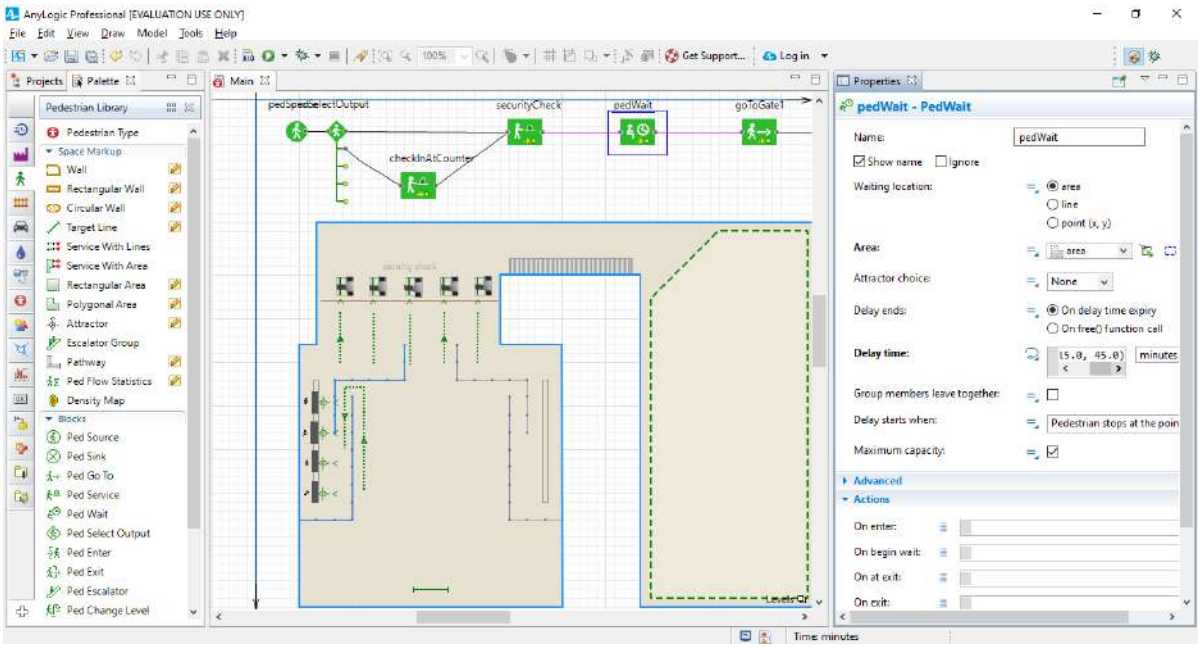
We want the passengers to wait before they go to the gate. To do this, we need to draw the waiting area where the passengers will wait and then add a flowchart block (PedWait) to simulate the waiting.

Draw the waiting area before the gates using the Polygonal Area element from the Space Markup section of the Pedestrian Library palette. Switch to the drawing mode and draw the area as shown in the figure below by clicking your mouse each time you want to add a point. When you're finished, double-click your mouse.

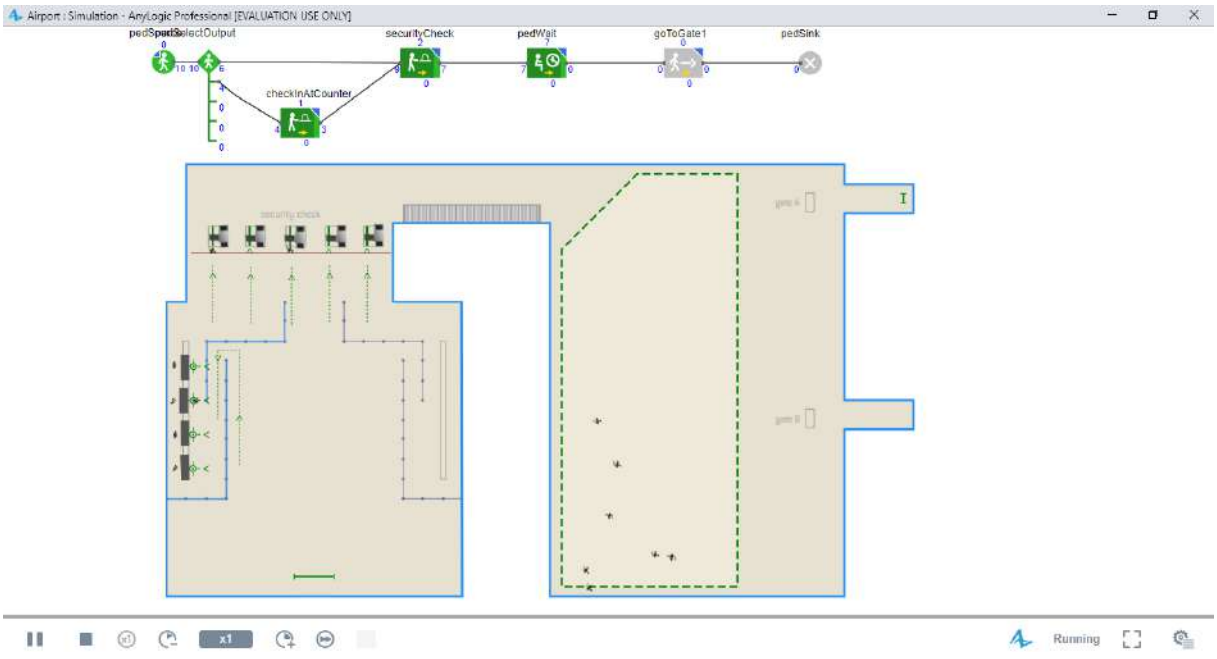


Add the PedWait block on to the flowchart between PedService and PedGoTo.

Modify the block's properties by selecting the area from the Area list, and then setting the Delay time to uniform(15, 45) minutes.



Run the model again, and you'll see the passengers now consider the belt barriers, and wait in the waiting area before they proceed to the gate.

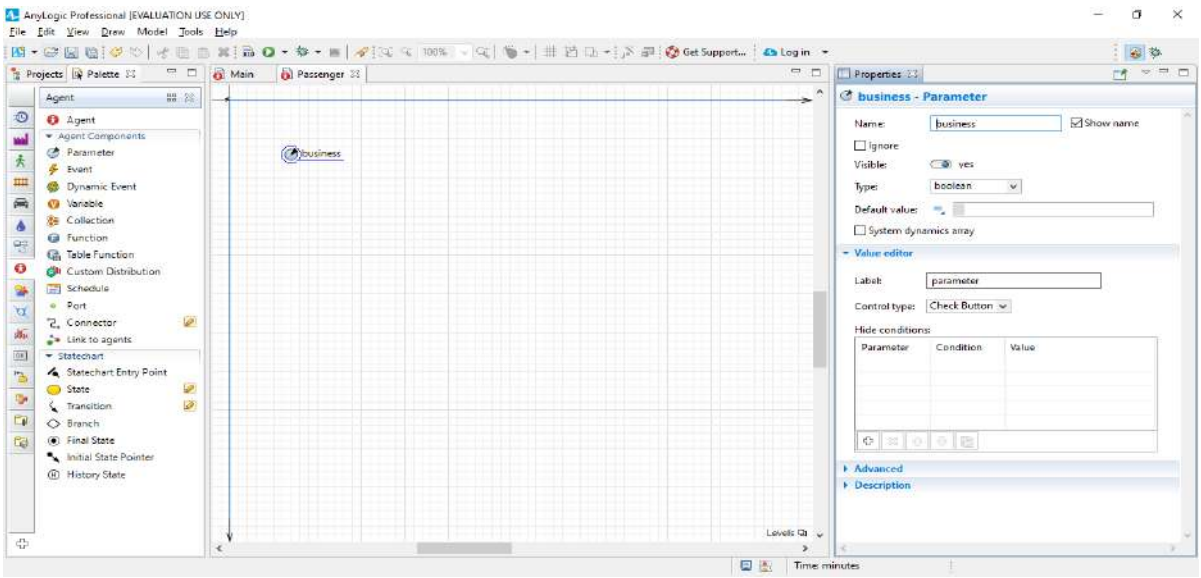


Phase 5. Defining the boarding logic

In this phase, we'll simulate the processes that take place at our airport's gate. The ticket checkpoint that each passenger must pass before they board their plane has one line for business class passengers – who are serviced first – and another for economy passengers. We'll add custom information to pedestrians to distinguish business class passengers from economy passengers.

In the Projects tree, open the Passenger agent type diagram by double-clicking the Passenger item.

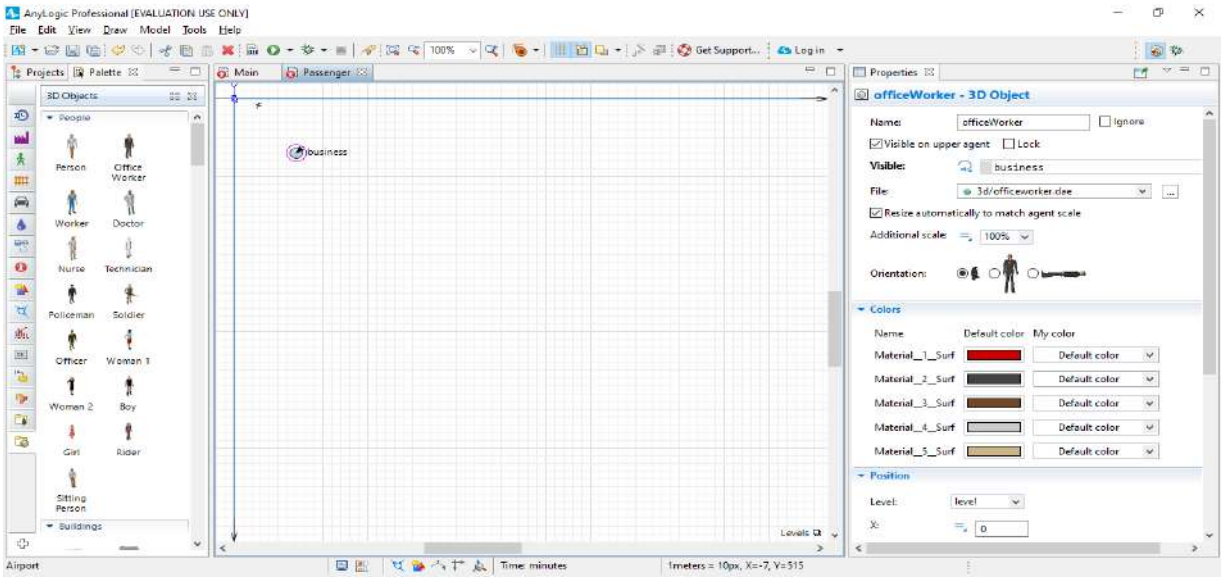
Add a Parameter from the Agent palette to define the passenger's class. Name it business and set Type: Boolean. If the parameter is equal to true, this is a business class passenger, otherwise (if the parameter is false), this is an economy passenger.



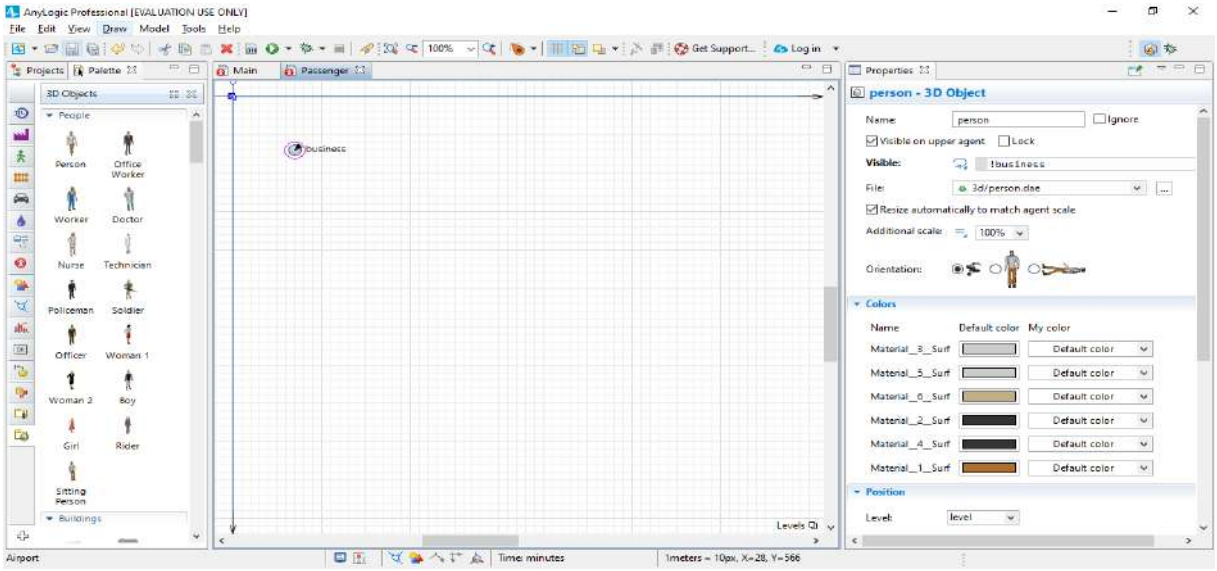
We want to distinguish passengers in 3D animation, namely animate business class and economy passengers with different 3D models. To do this, we'll use the existing Person 3D object to represent economy passengers and add another 3D shape to represent business class passengers.

Add the 3D object Office worker to animate a business class passenger and then place the figure on the axis origin point (0,0), right on the Person shape.

Change the visibility of these objects. First, click the Office worker shape. We want this shape to be visible only when this is a business class passenger, that is, when its business parameter is true. Switch to the Visible property's dynamic value editor by clicking the icon to the right of the Visible label, and then type business in the box. By doing this, we're making this 3D shape visible only when pedestrian's business parameter is true.



Now select the person 3D object (you can do this from Projects tree) and set Visible: !business. This shape will be visible only if the passenger is an economy passenger. The symbol ! is the boolean operand NOT. The expression !business returns true when the business is NOT true – when the passenger is not a business class passenger but is an economy passenger.



We want to set up the passengers class when they arrive to the airport terminal.

Return to the Main diagram and add a Function from the Agent palette. Name it setup Passenger.

Configure the function as follows:

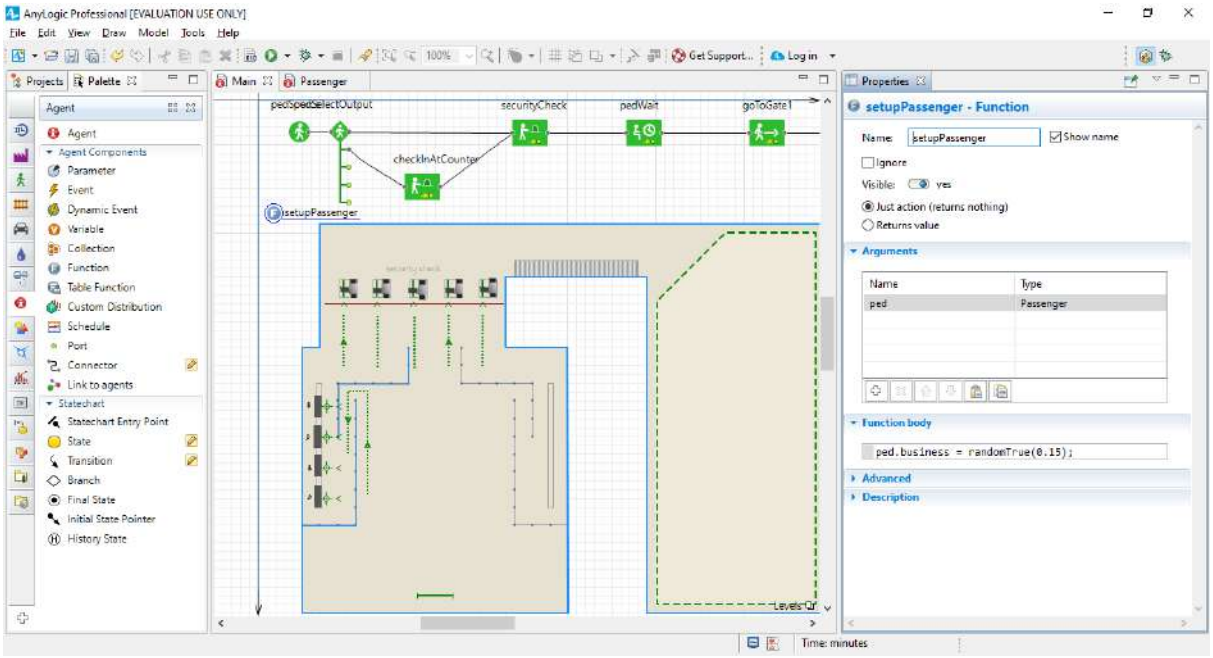
Create one argument to pass the newly-created passenger to the function:

Name: ped

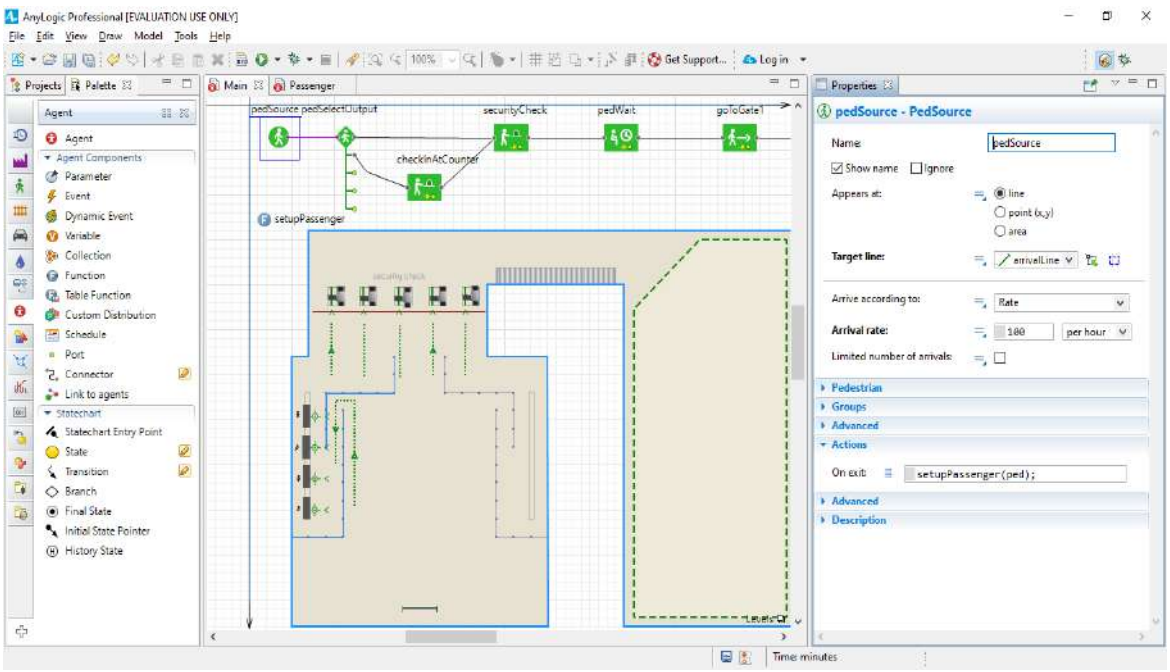
Type: Passenger

The function's code defines the frequency with which the business class passengers appear in the model:

ped.business = randomTrue(0.15);

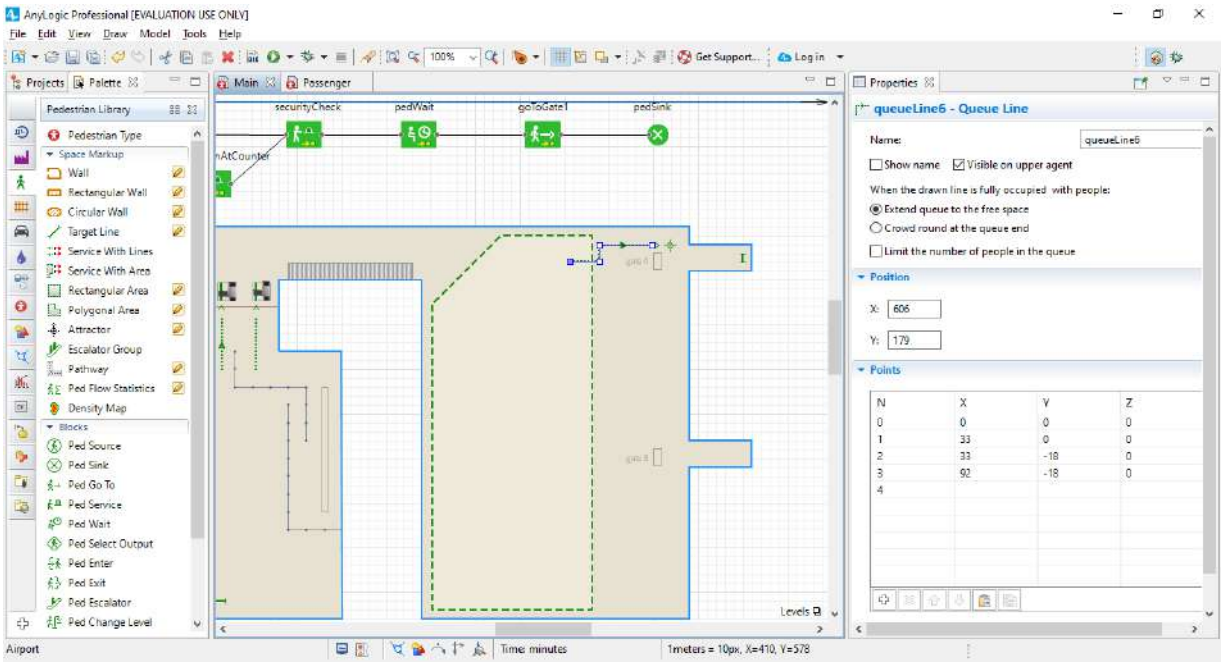


Call this function when a new pedestrian is created by the pedSource block. In the pedSource properties area, click the arrow to expand the Actions section, and then enter the following code in the on exit box: setupPassenger(ped);

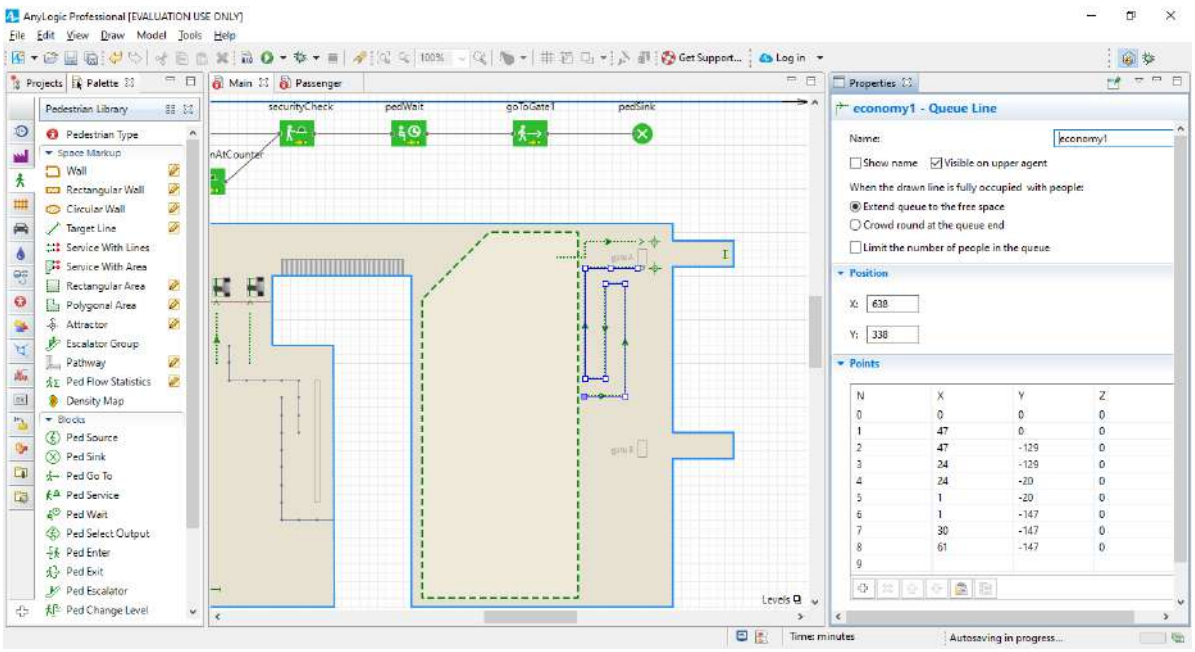


Draw two services with lines for the upper gate, one for business class and one for economy passengers.

Draw a Service with line, defining the priority line (point service, 1 service, 1 line). Name this service business1

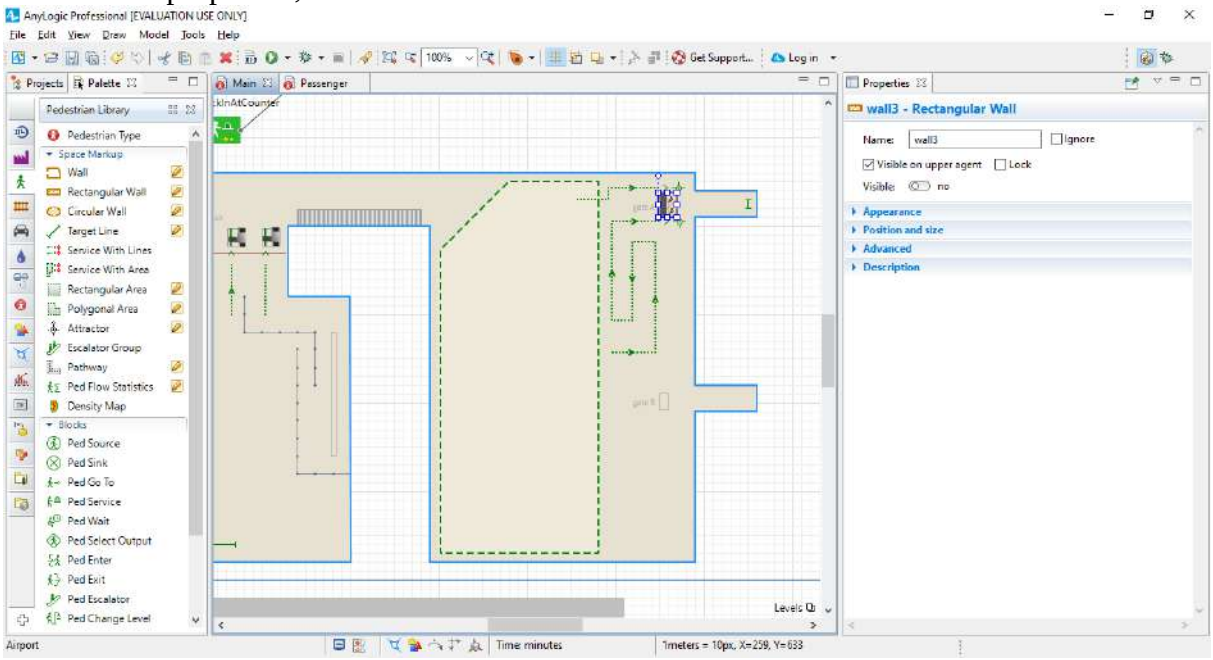


Add one more Service with Line, and name it economy1.



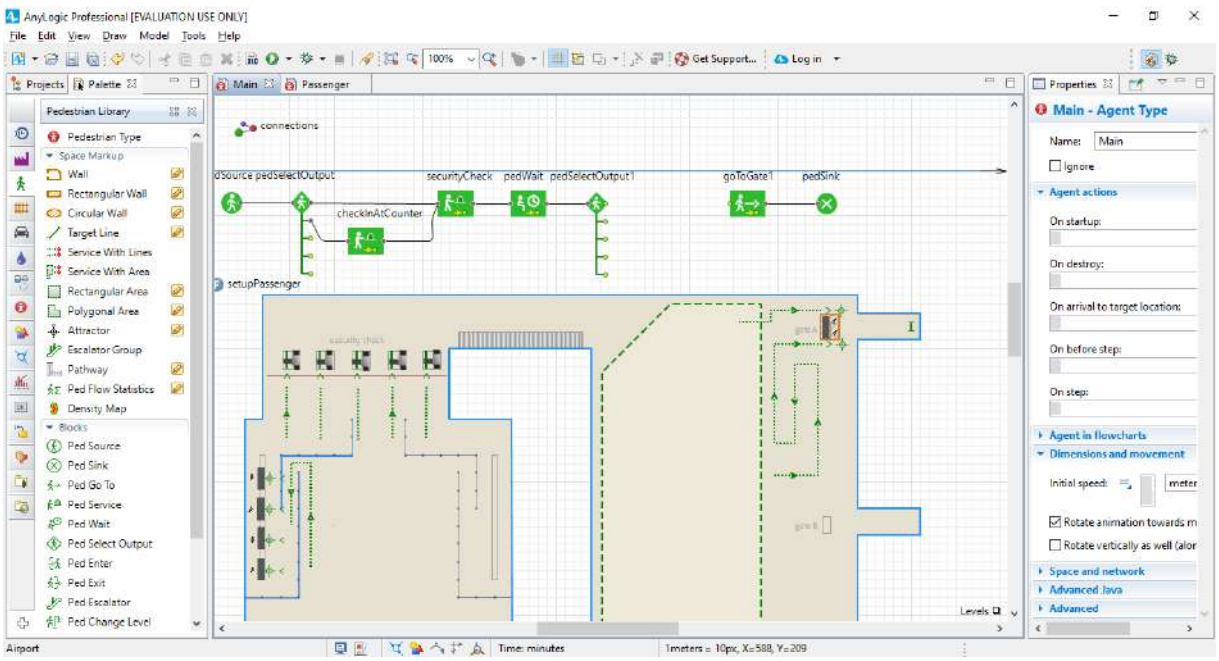
Draw an area at the gate with the Rectangular wall element, and then add a table and two 3D woman models at the table.

In the wall's properties, set Visible to no to make this wall invisible at the model runtime.



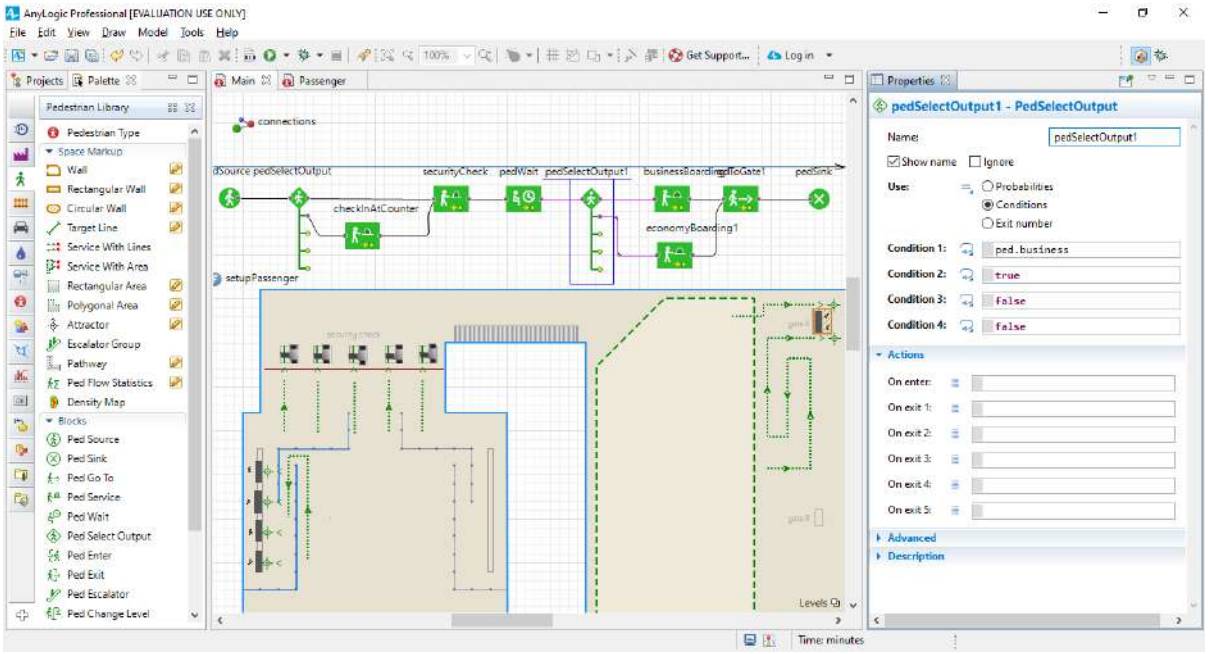
Insert the blocks into the flowchart between the pedWait and goToGate1 objects.

Add PedSelectOutput to route business class and economy passengers to different lines.

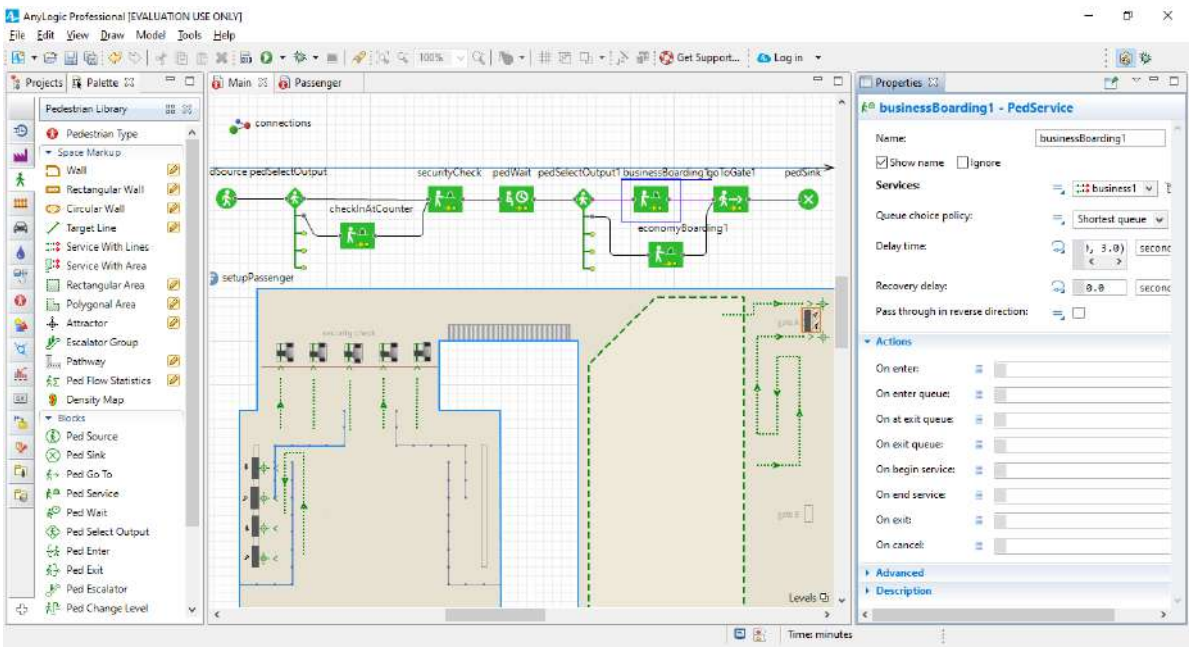


Add two PedService blocks: businessBoarding1 and economyBoarding1 to simulate the process of checking passengers tickets at the gate.

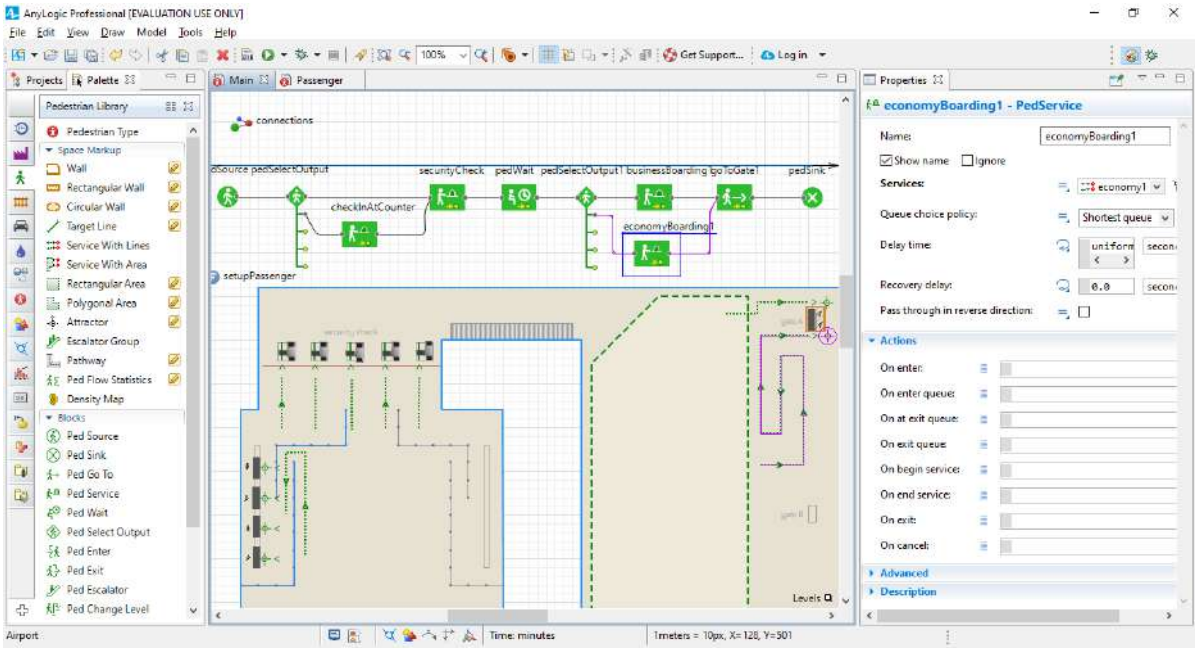
Since the PedSelectOutput block routes business class and economy passengers to different lines, select Use: Conditions, and then type ped.business in the Condition 1 box. This expression will return true for all business class passengers, which means they will follow the upper flowchart branch and join the priority line. After you set up the conditions for the block's next output ports (true, false, false), the model will direct all other passengers to the second output port.



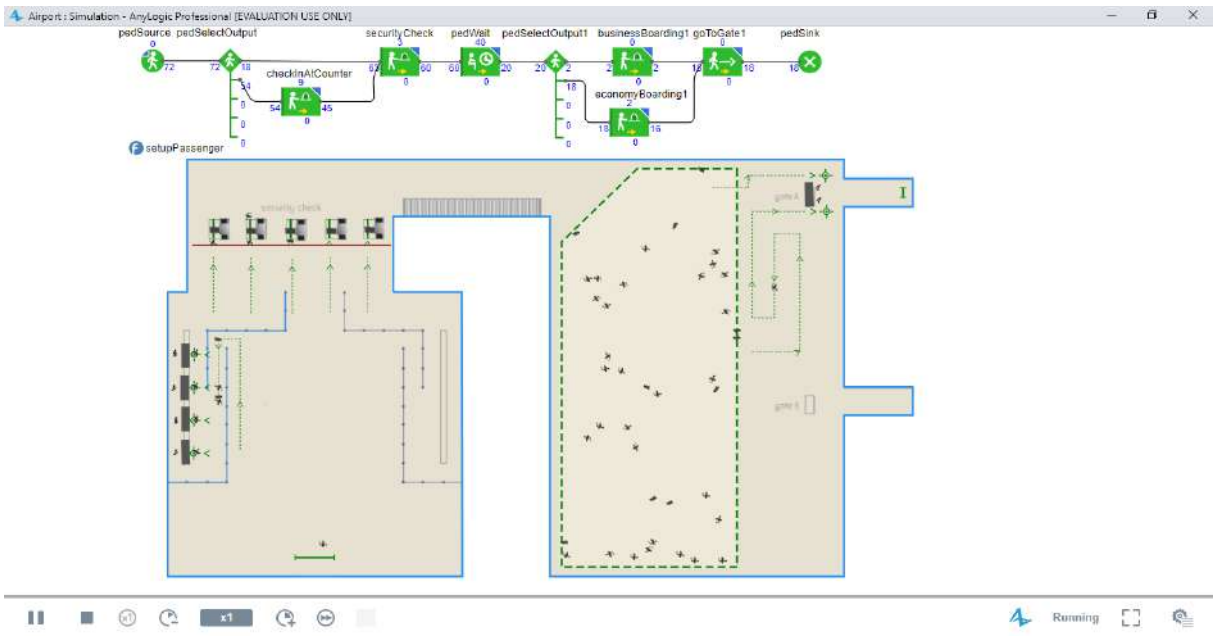
For the PedService block businessBoarding1, choose business1 as Services. Since it takes between two to five seconds to check a passenger's ticket, you can slightly change the Delay time.



For economyBoarding1, set Services: economy1, adjust the Delay time.



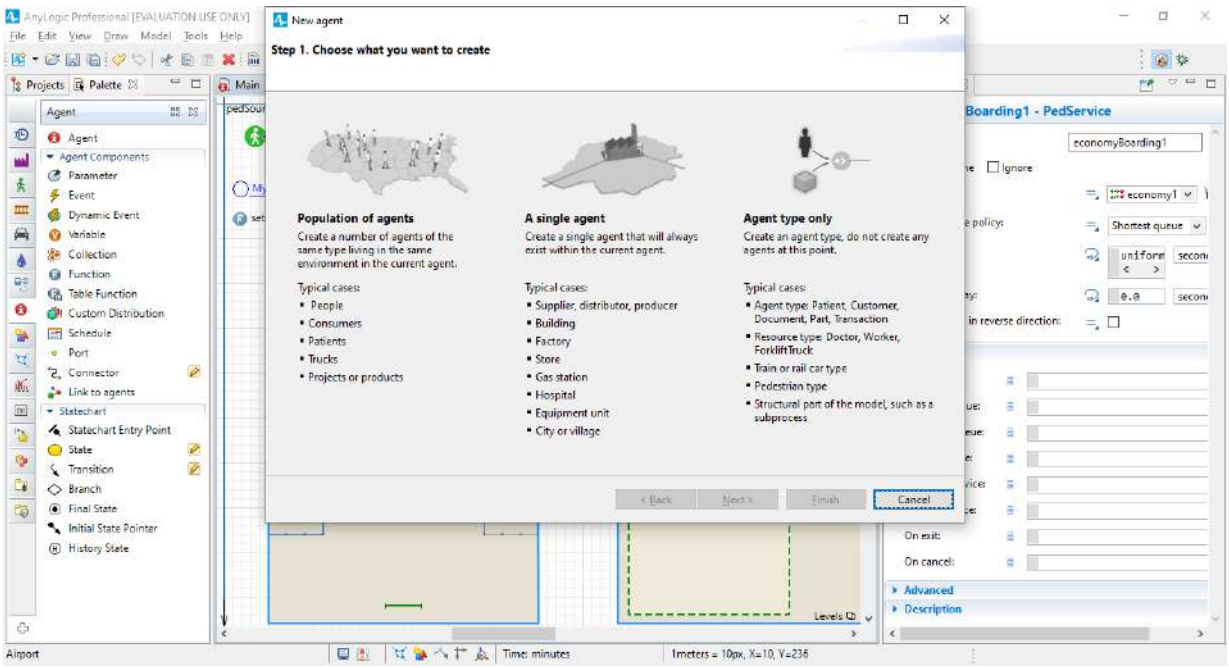
Run the model. You'll see passengers pass the checkpoint, and a small number of them will take the priority line.



Phase 6. Setting up flights from MS Excel spreadsheet

In this phase, we'll model how airplanes take off at specific times according to a timetable stored in a database. We'll start by creating a Flight agent type that will allow our model to store flight information.

Add an empty agent population of the new Flight agent type by dragging the Agent element from the Agent palette on to the Main diagram.



In the New agent wizard, do the following:

a. Select Population of agents.

b. Select the option I want to create a new agent type. Click Next.

c. Specify The name of new type: as Flight. The population name will prefill as flights. Click Next.

d. Since we won't need to animate flights, select None for the animation. Click Next.

e. Skip the parameters creation step. Click Next.

f. Select the Create initially empty population...to allow us to use data from the database to programmatically add flights.

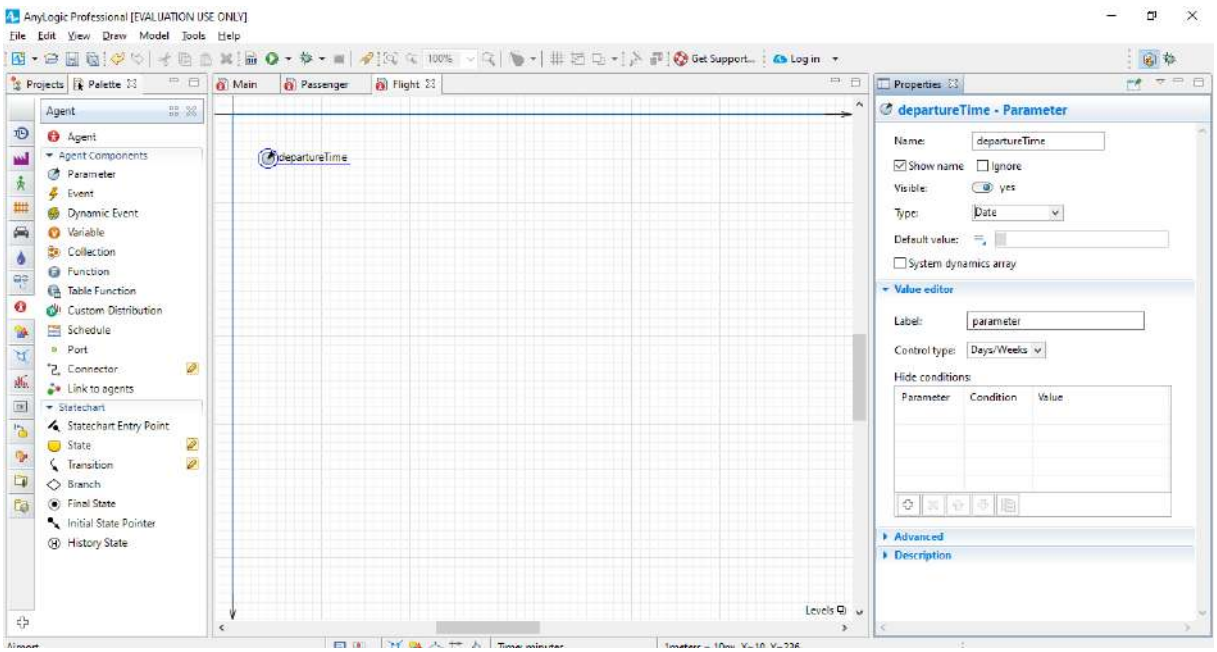
g. Click Finish.

In the Projects view, double-click Flight to open its diagram. On the Flight diagram, create three different parameter types that will store the flight's departure time, destination, and gate number.

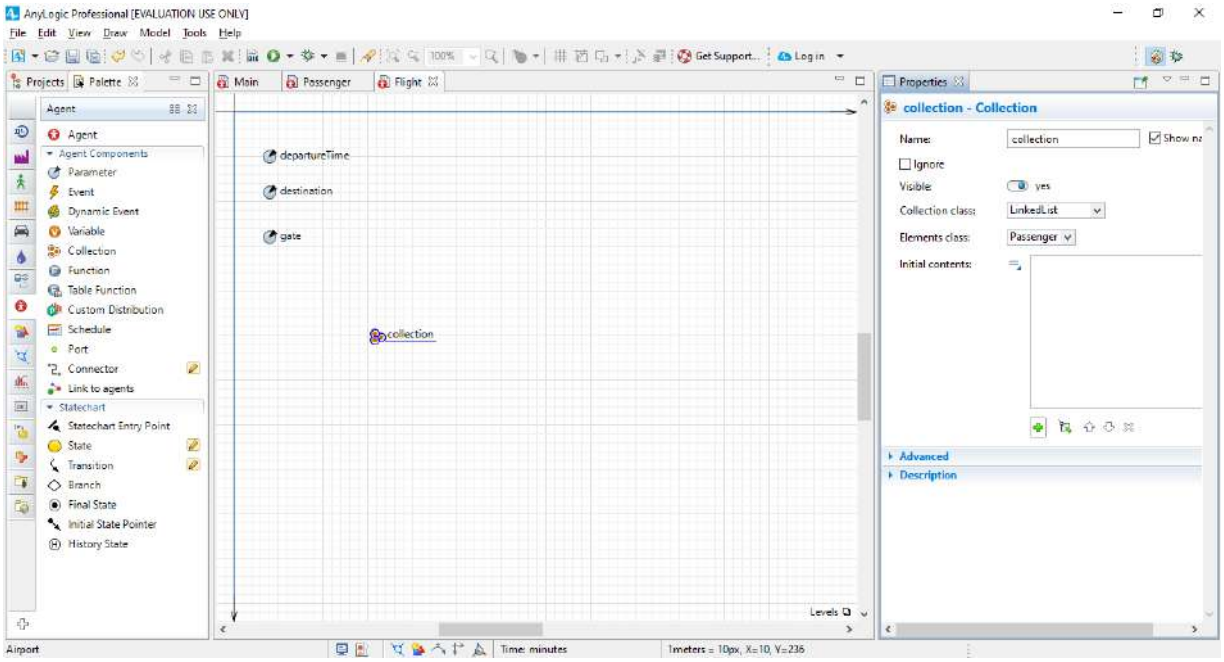
departureTime. In the Type list, click Other..., and then type Date in the field to the right.

destination, In the Type list, click String.

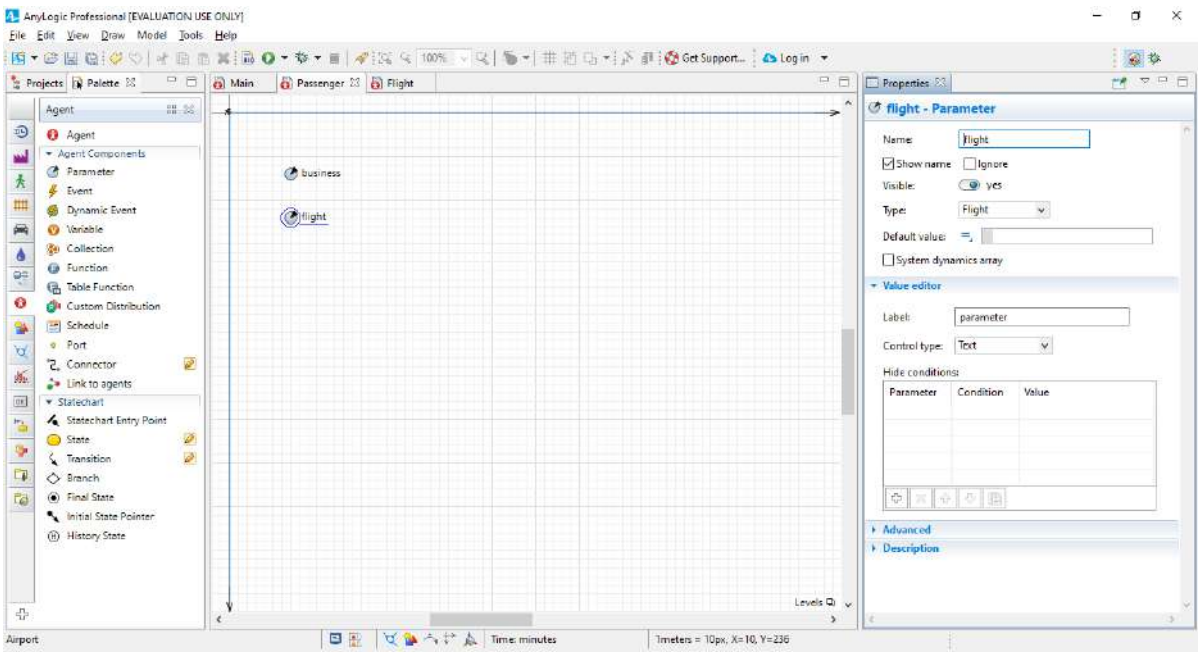
gate, In the Typelist, click int.



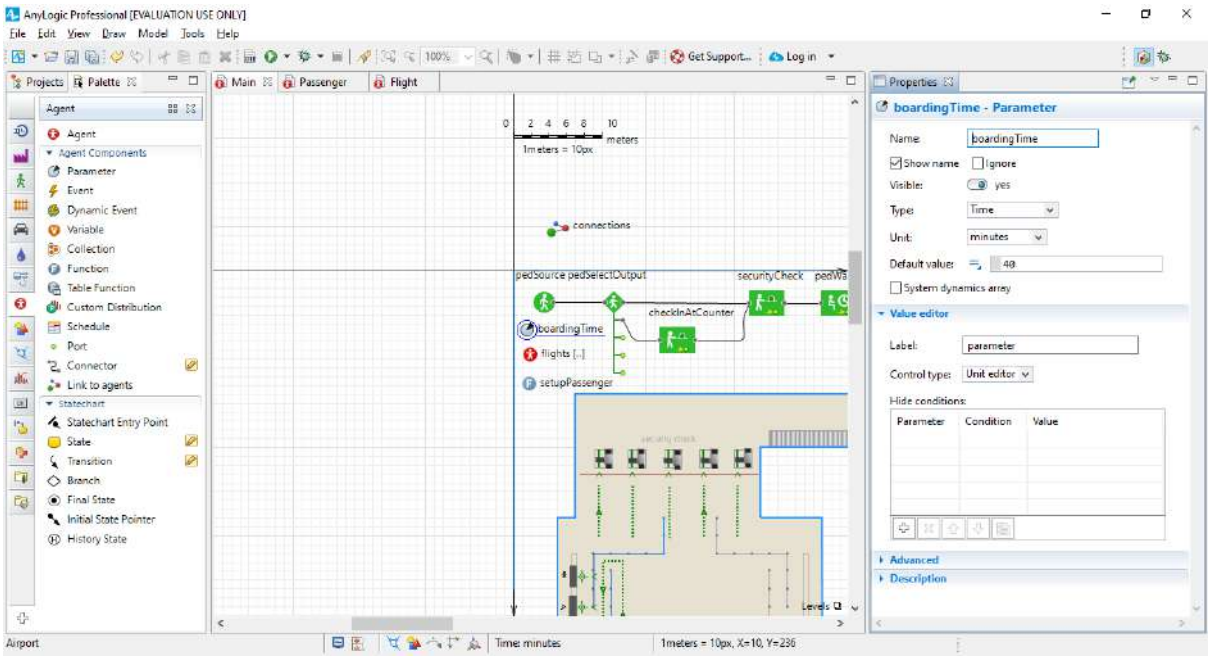
Use the Agent palette to add a Collection, name it passengers, and then set the Collection class to LinkedList, and the Elements class to Passenger. This collection will store the list of passengers that have bought tickets for the flight.



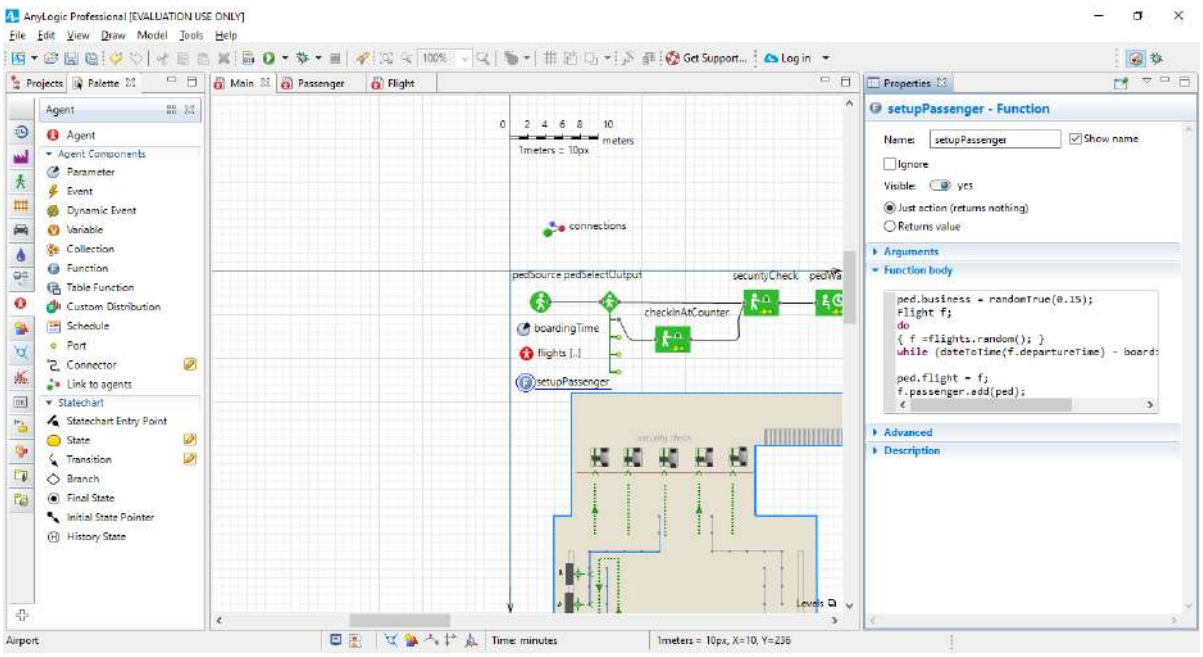
Now that we've created the Flight agent type, we'll add a flight parameter to the Passenger diagram and set the parameter's Type to Flight. This parameter will store the passenger's flight.



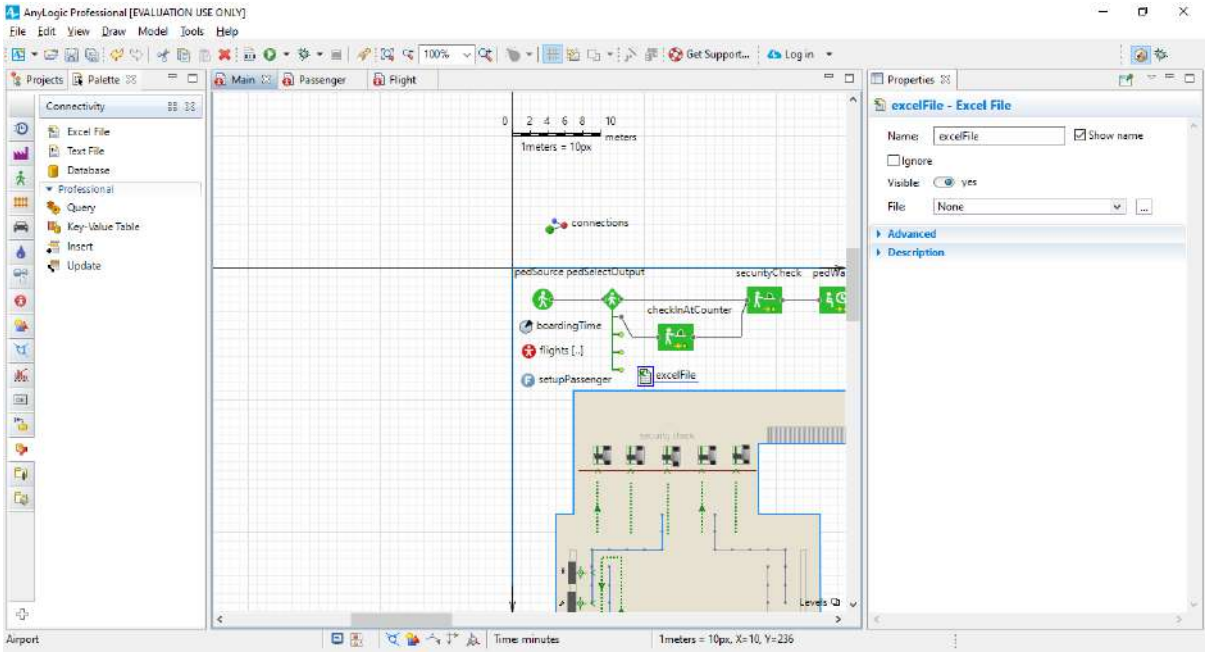
Return to the Main diagram and add a parameter to define the boarding time duration. Name the new parameter boardingTime, and then choose Type: Time, Unit: minutes, and Default value:40



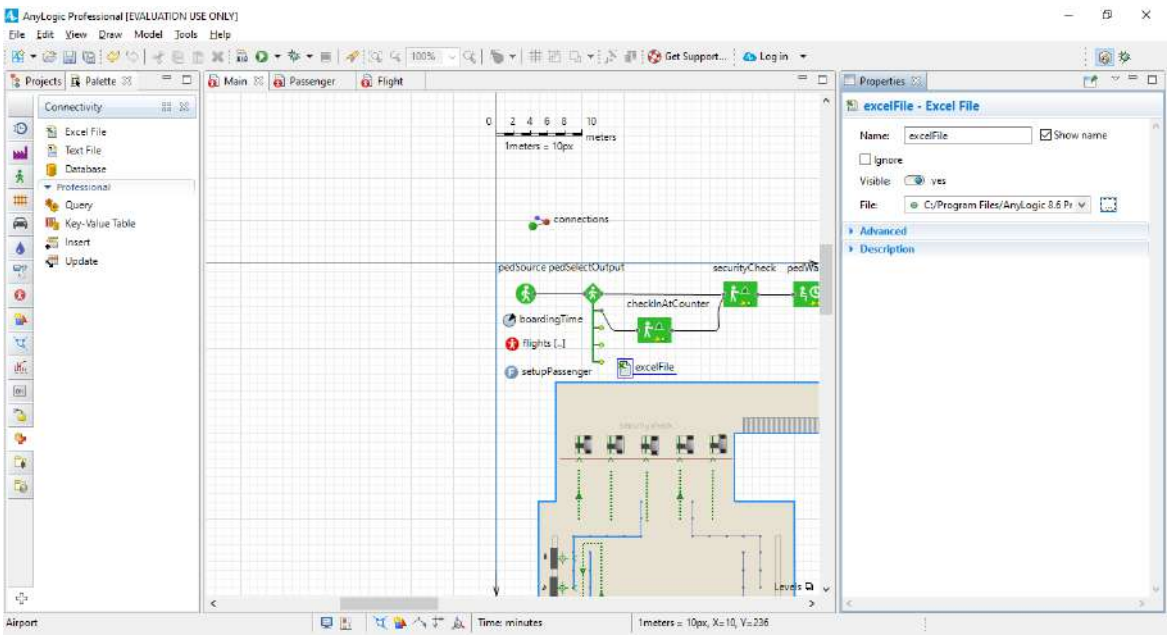
Select the function setupPassenger that we created earlier to complete our setup process. The function now uses the random() function to randomly select the flight that the passenger will take from the list of available flights. The flight is stored in a passenger's parameter flight, and AnyLogic adds the passenger to a collection of passengers who are taking the same flight. Modify the code in the Function body section:



Drag the Excel File element from the Connectivity palette.

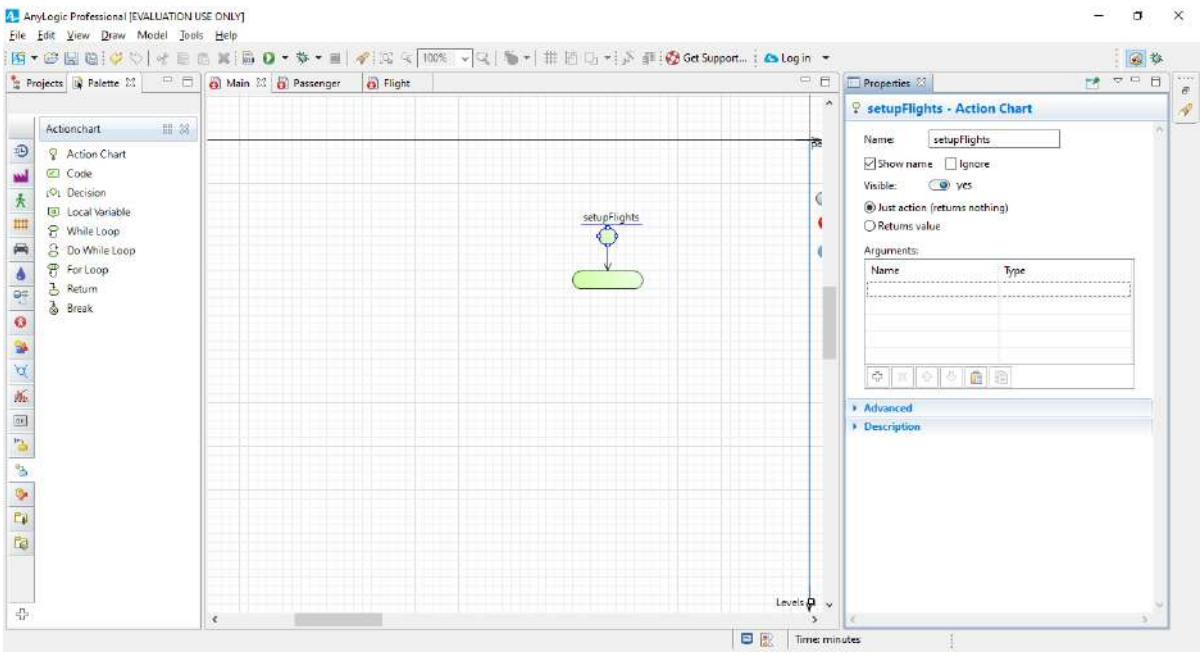


Open excel File properties and then click Browse to add the Flights.xlsx file. You'll select the file from AnyLogic folder/resources/AnyLogic in 3 days/Airport



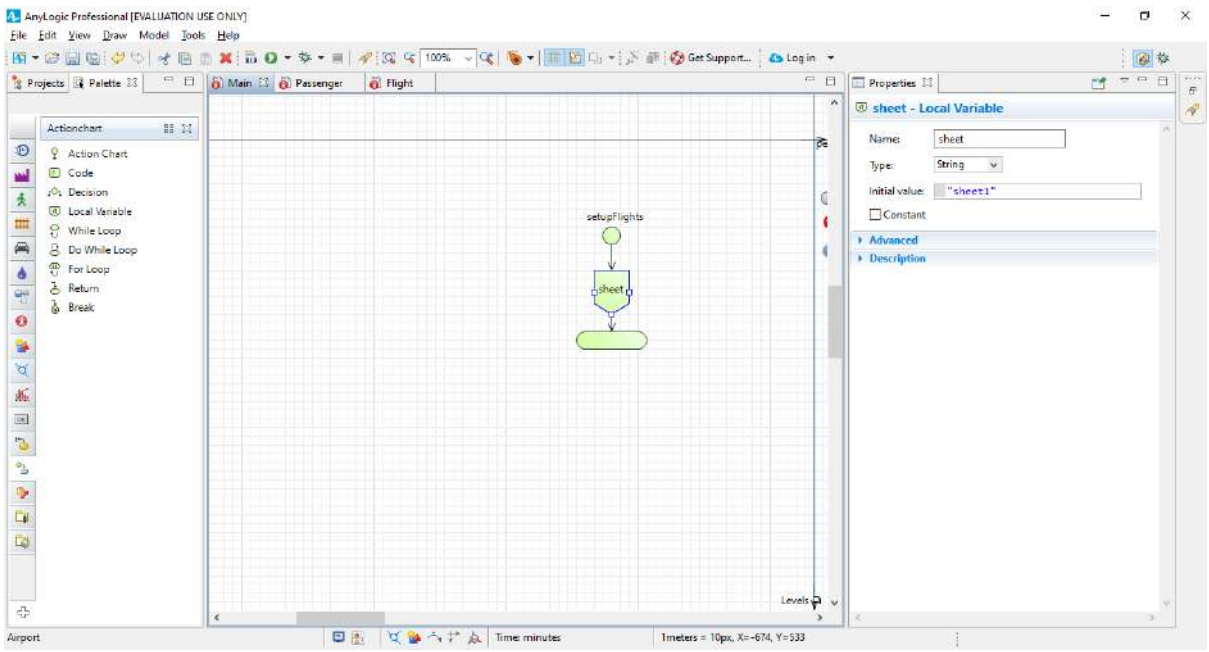
We'll read the data from Excel file to configure flights and use an action chart to define this algorithm.

Open the Actionchart palette, drag the Action Chart element on to the Main diagram, place it to the left of the Y-axis, and then name it setupFlights. Since the action chart doesn't return data and doesn't have arguments, you don't have to change the element's default settings.

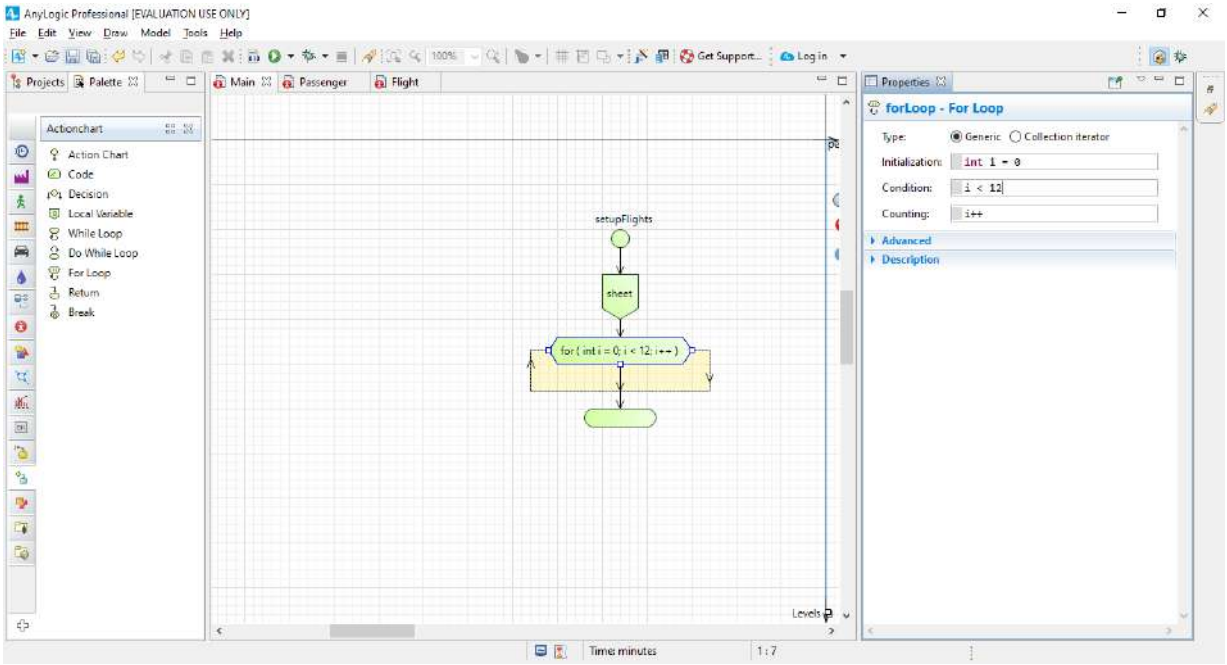


Add the Local Variable. To insert a block, drag it on to the action chart, and then release the mouse button when you see AnyLogic highlights the insertion placeholder in cyan as displayed in the figure below.

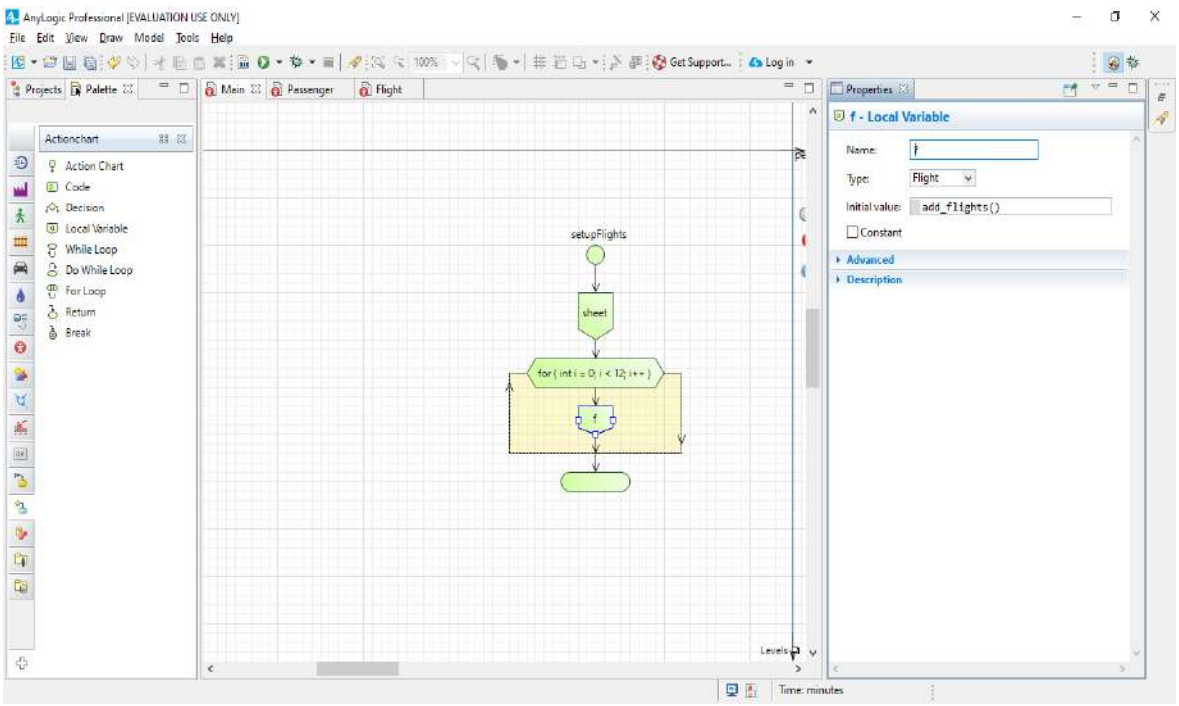
You'll use this local variable to store the Excel file's sheet name.



Insert the For Loop on to the action chart, and then set the loop to count to 12 to reflect the 12 data entries in our spreadsheet.



Add another Local variable inside the for loop to configure flights. By creating an additional flight and using AnyLogic's add_<populationName> function to add it to the flights population, we're allowing the local variable f to store the reference to our newly-created Flight agent.

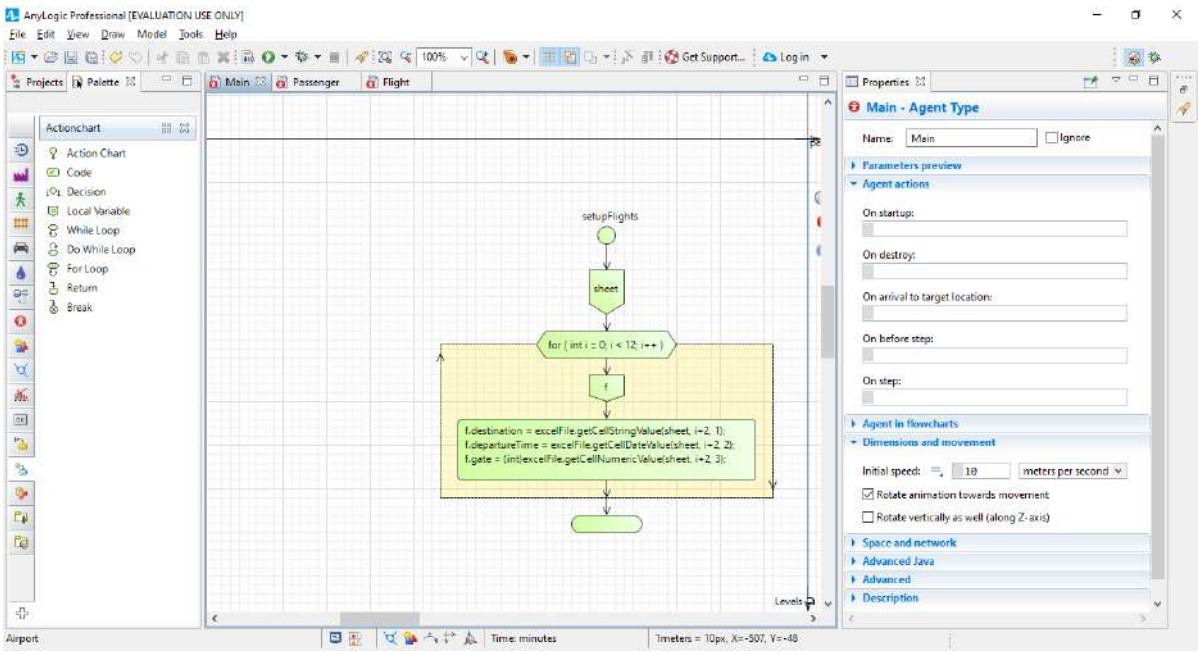


Finally, insert the Code inside the For Loop area to allow us to define code that reads the selected Excel file's data. In the Code field, type the following:

f.destination = excelFile.getCellStringValue(sheet, i+2, 1);

f.departureTime = excelFile.getCellDateValue(sheet, i+2, 2);

f.gate = (int)excelFile.getCellNumericValue(sheet, i+2, 3);

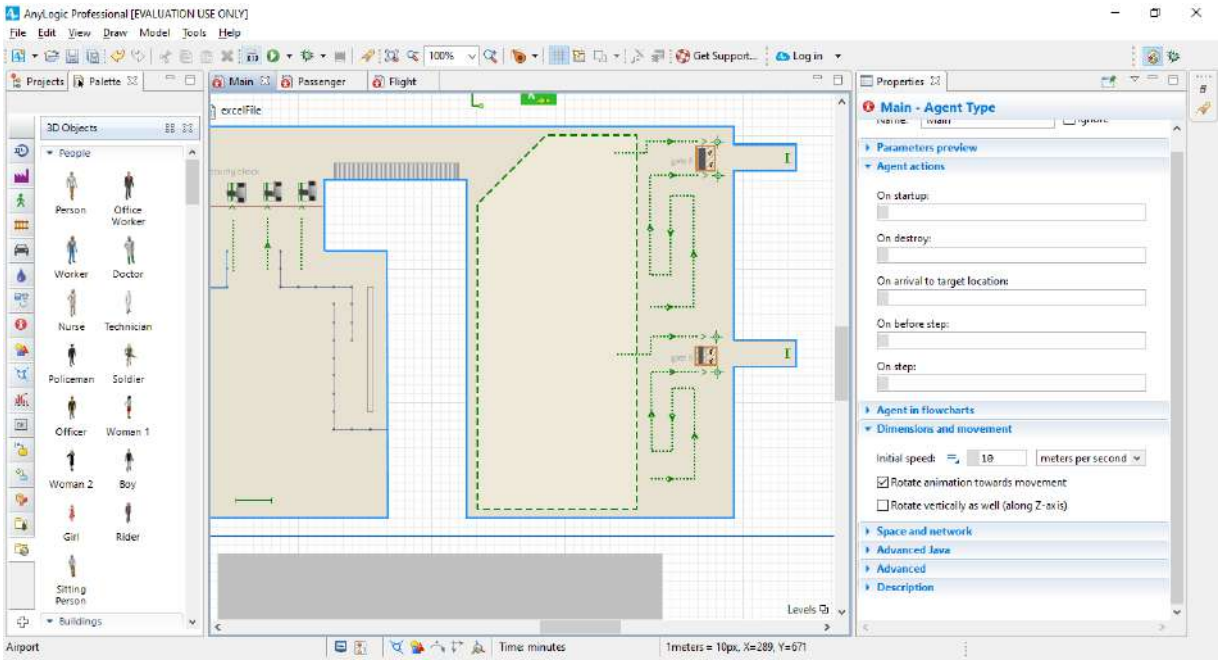


Define the second gate:

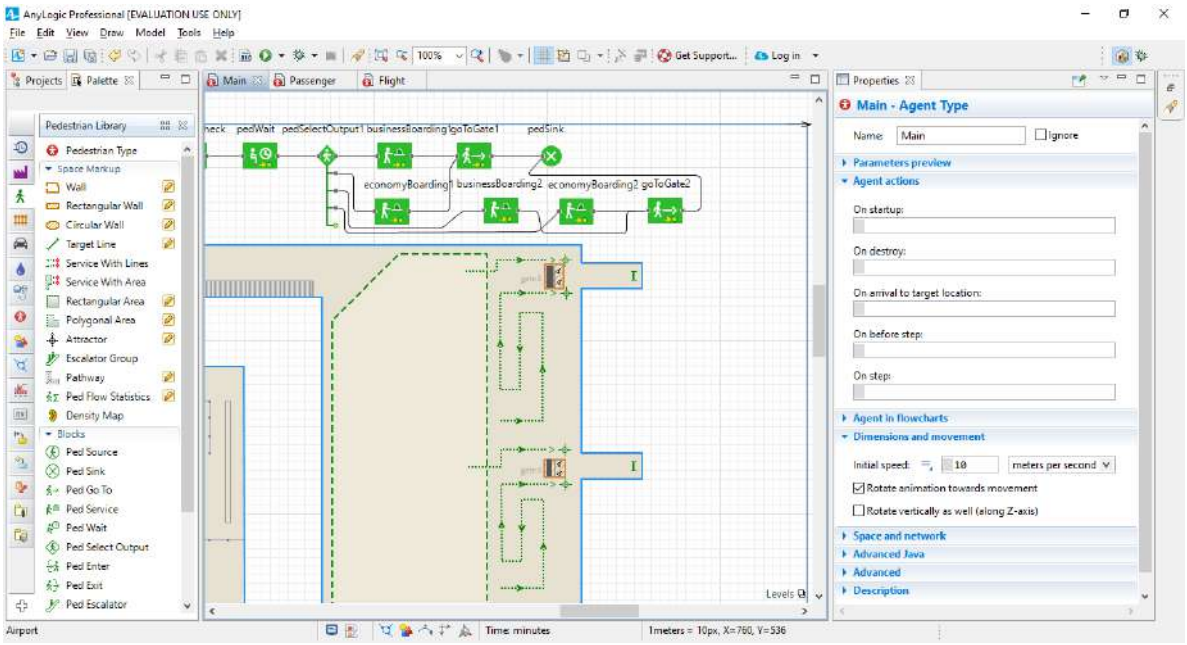
Add two Service with lines elements: business2 and economy2.

Draw the rectangular wall, table and women figures.

Draw the Target line gateLine2



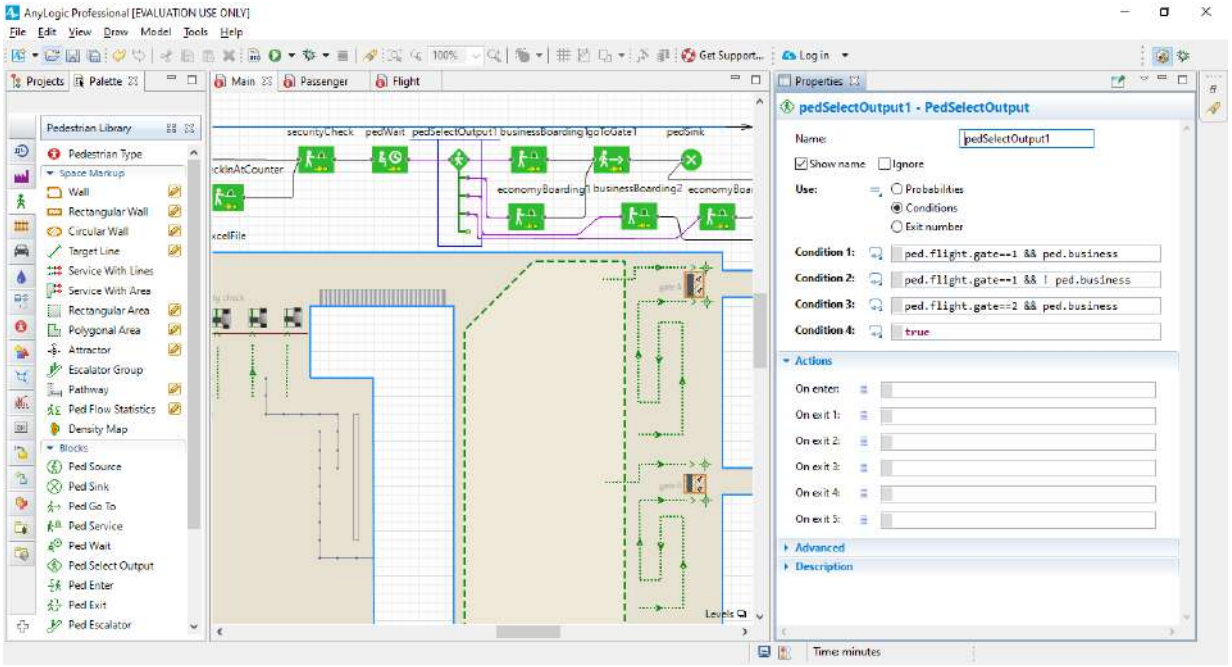
Add two more PedService blocks, businessBoarding2 and economyBoarding2, that go out of PedSelectOutput and go into PedGoTo. Let PedSelectOutput directs passengers to four different ports.



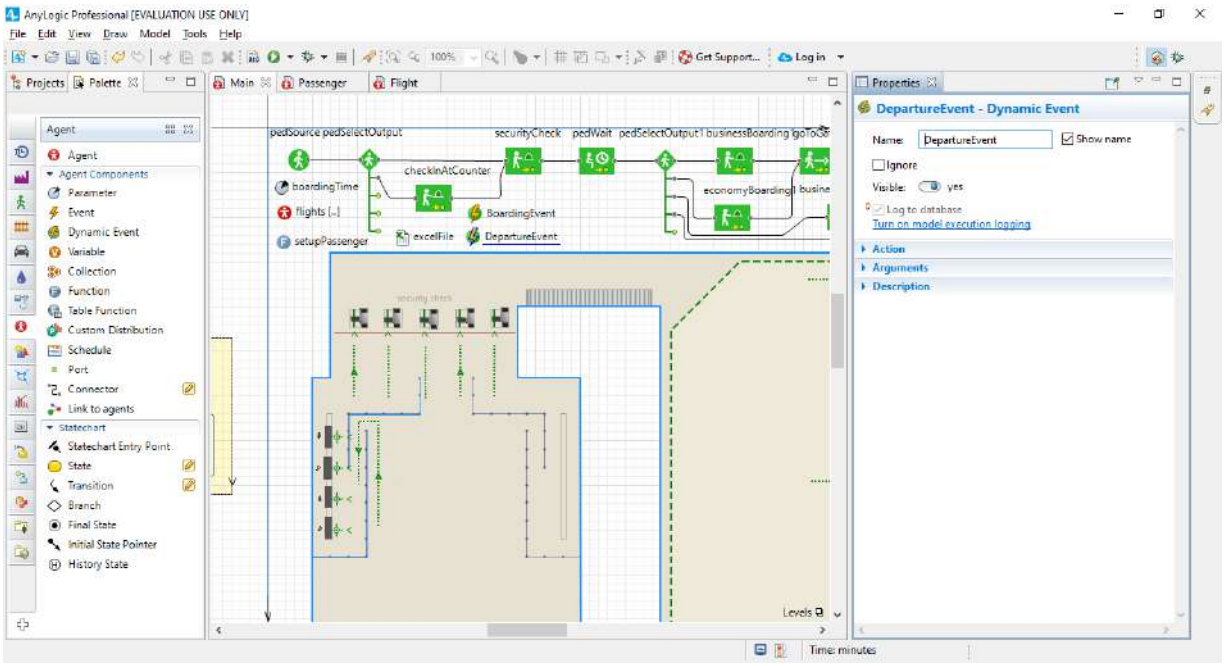
Add another PedGoTo block to model how passengers move to the second gate. Select gateLine2 as the block's Target line.

For businessBoarding2, set Services: business2. For economyBoarding2, set Services: economy2. For both, set Service time: uniform(2, 5)seconds.

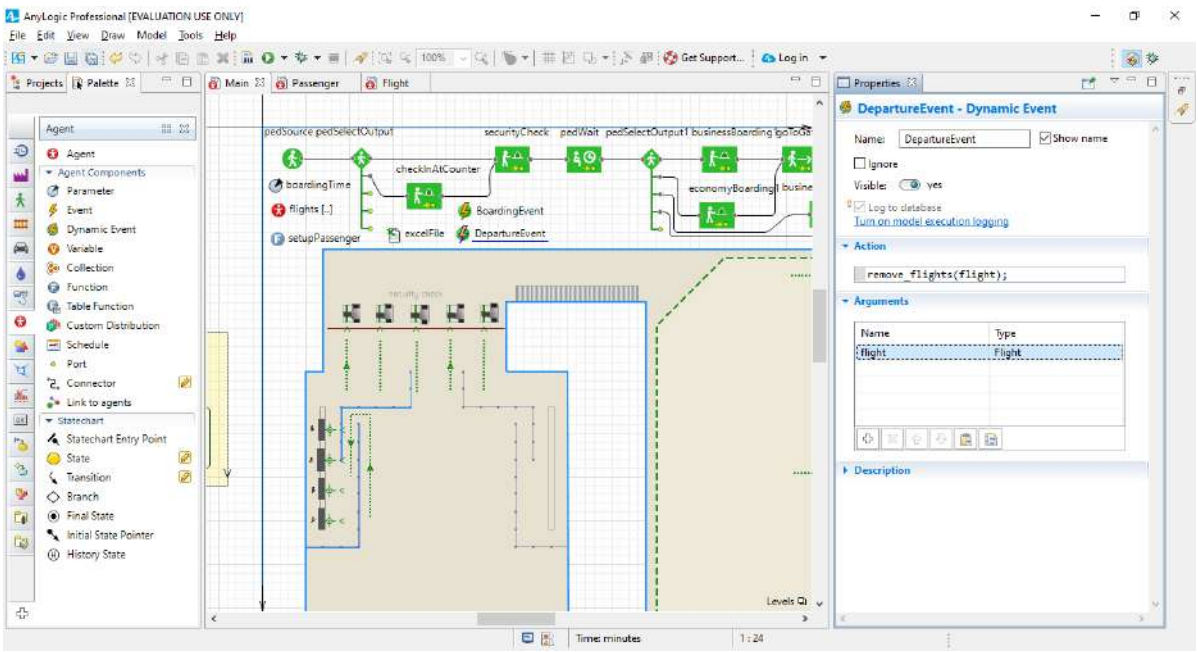
With our flights set up, we can change the pedSelectOutput1 conditions that define which gate our passengers select.



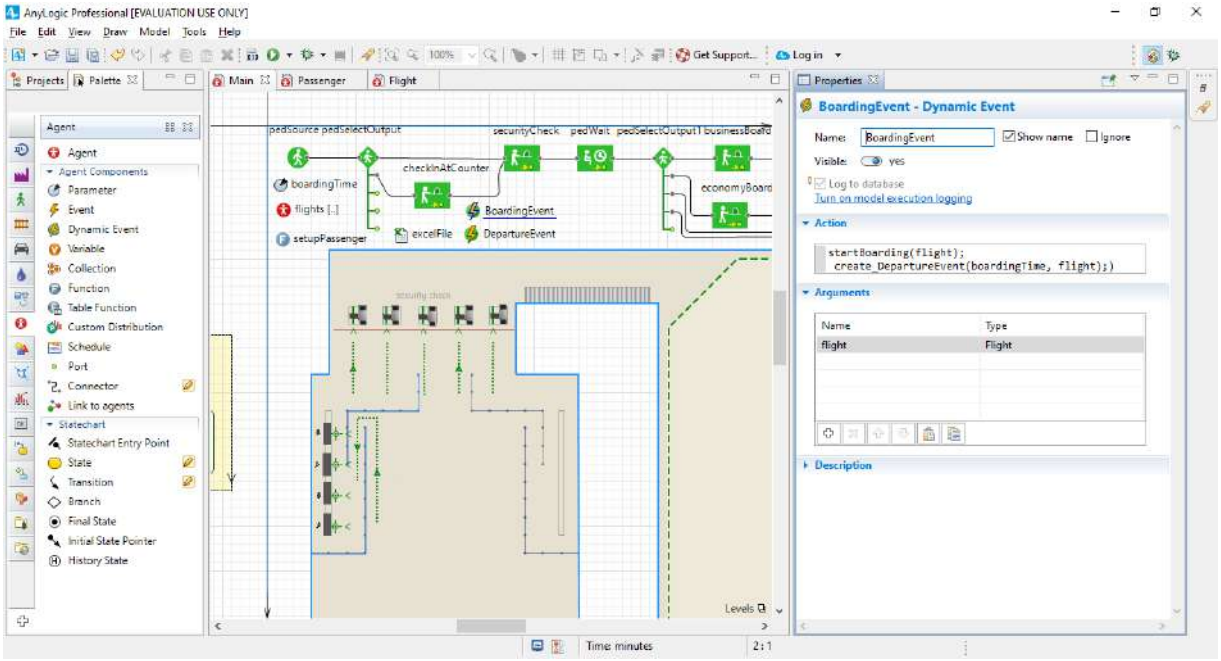
Add two Dynamic Event elements from the Agent palette on to the Main diagram



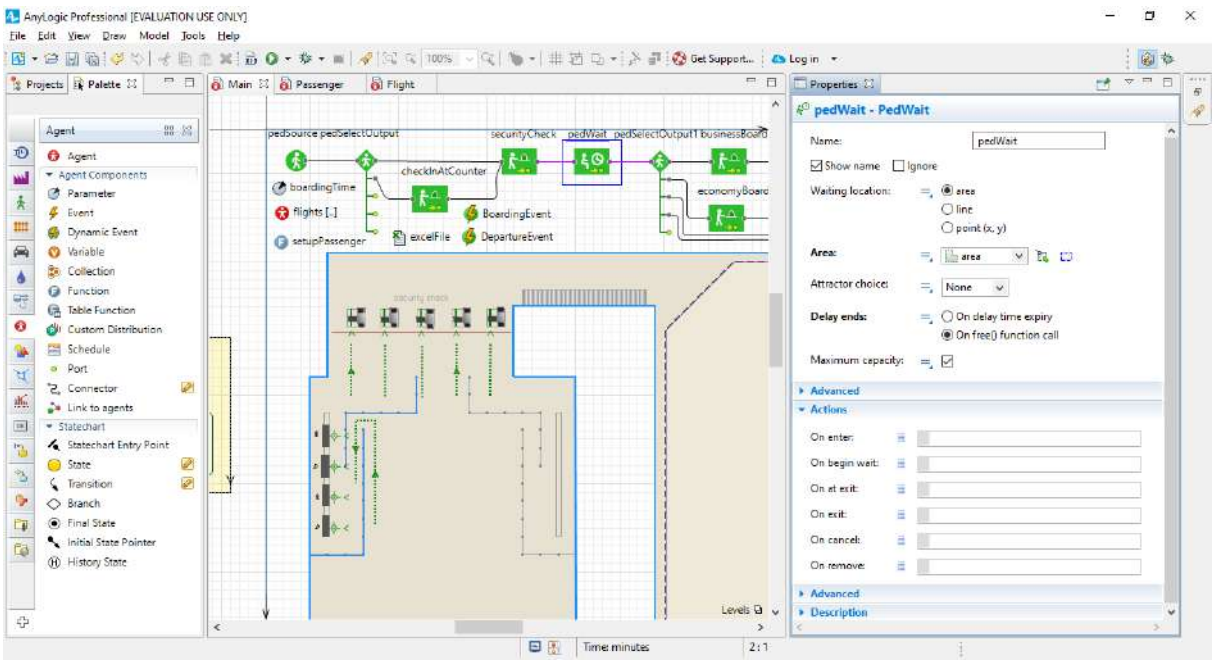
The dynamic event DepartureEvent schedules a plane's departure by removing the flight from the agent population that contains upcoming flights. Use the figure below to help you set up the event.



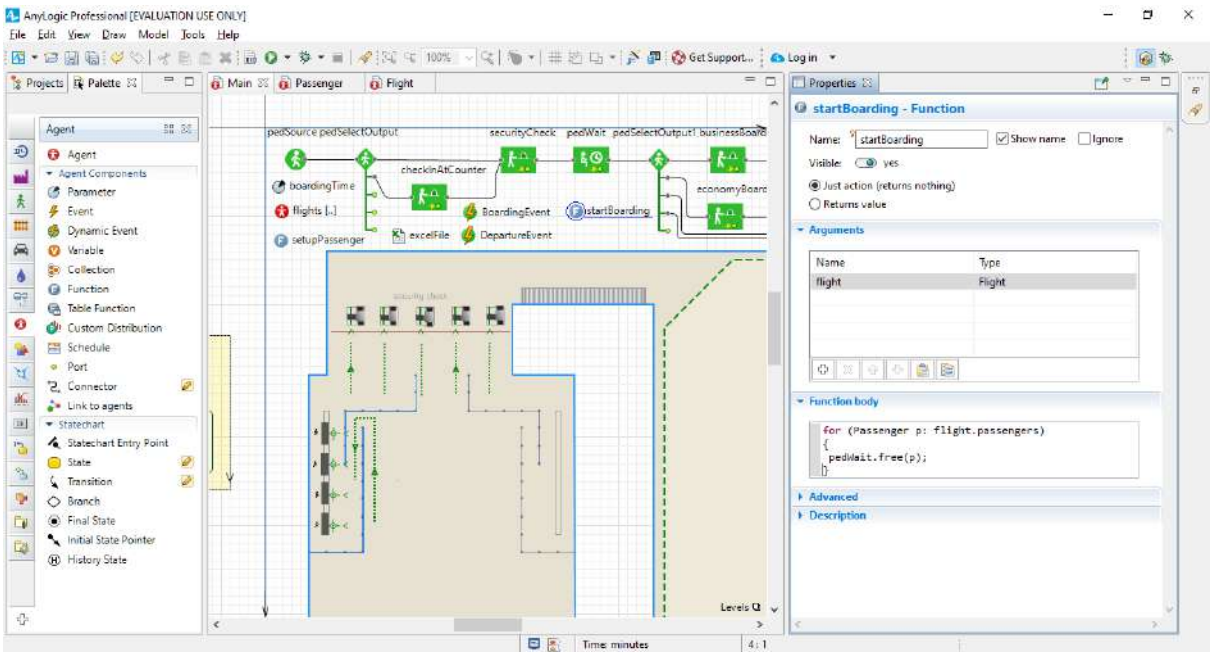
A second dynamic event, BoardingEvent, schedules the plane's boarding and then creates an instance of the dynamic event DepartureEvent that schedules the flight to depart in 40 minutes.



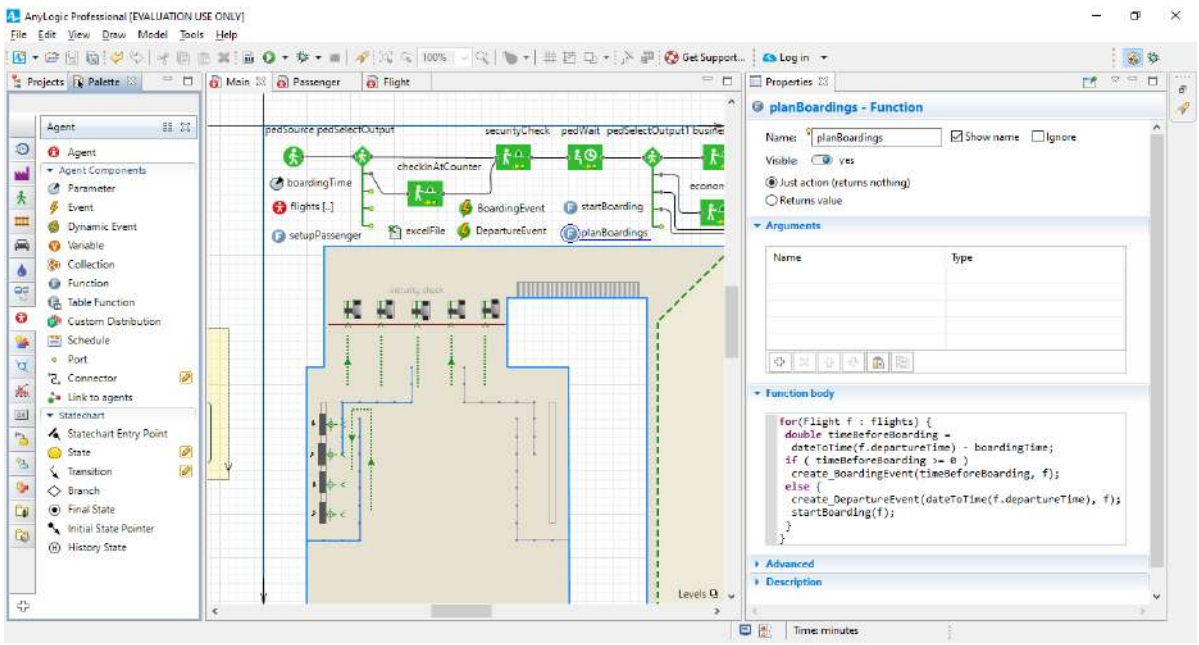
Change the pedWait block's Delay ends parameter from On delay time expiry to On free() function call to ensure passengers who need to wait to board their plane will wait for the announcement in the waiting area.



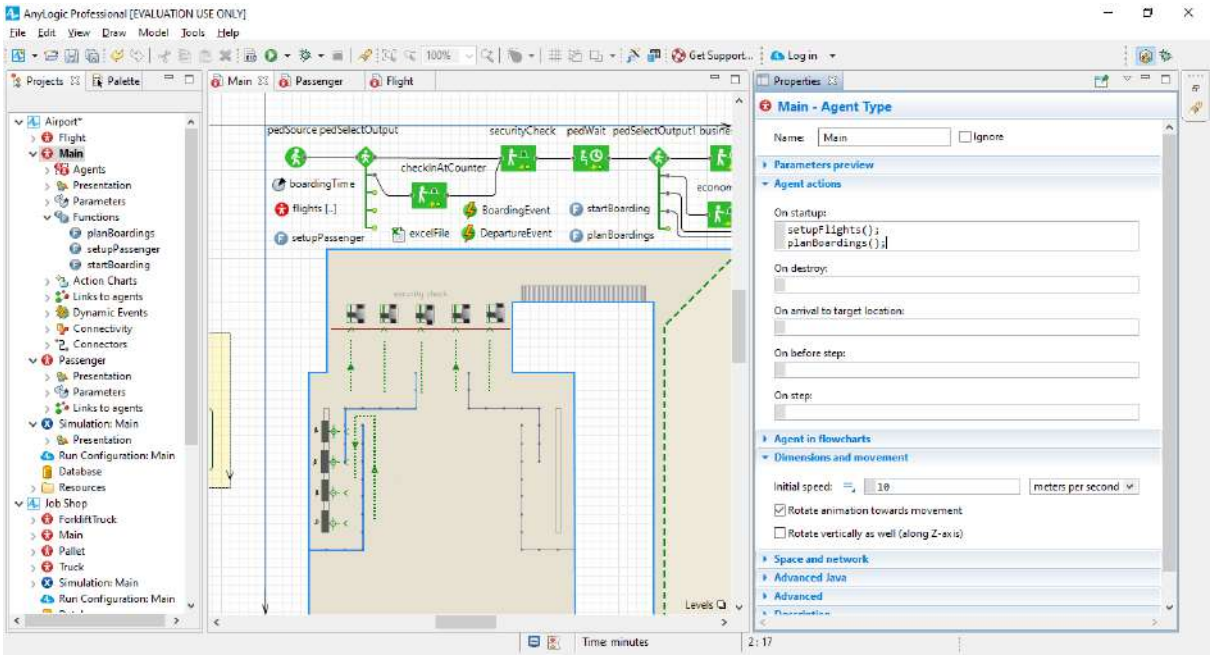
Define a Function start Boarding to model the start of the plane's boarding process. This function iterates through the passengers who are waiting to board for the given flight and allows them to board by ending their delay in the block pedWait with the call of the block's function free()



Define a planBoardings function to schedule boarding for all registered flights. The function iterates through the agent population flights in the For Loop. It allows flights that are set to take off before their boarding time has elapsed to immediately start boarding. Flights that do not meet this condition will start their boarding process 40 minutes before their departure time as we defined in the boardingTime parameter.

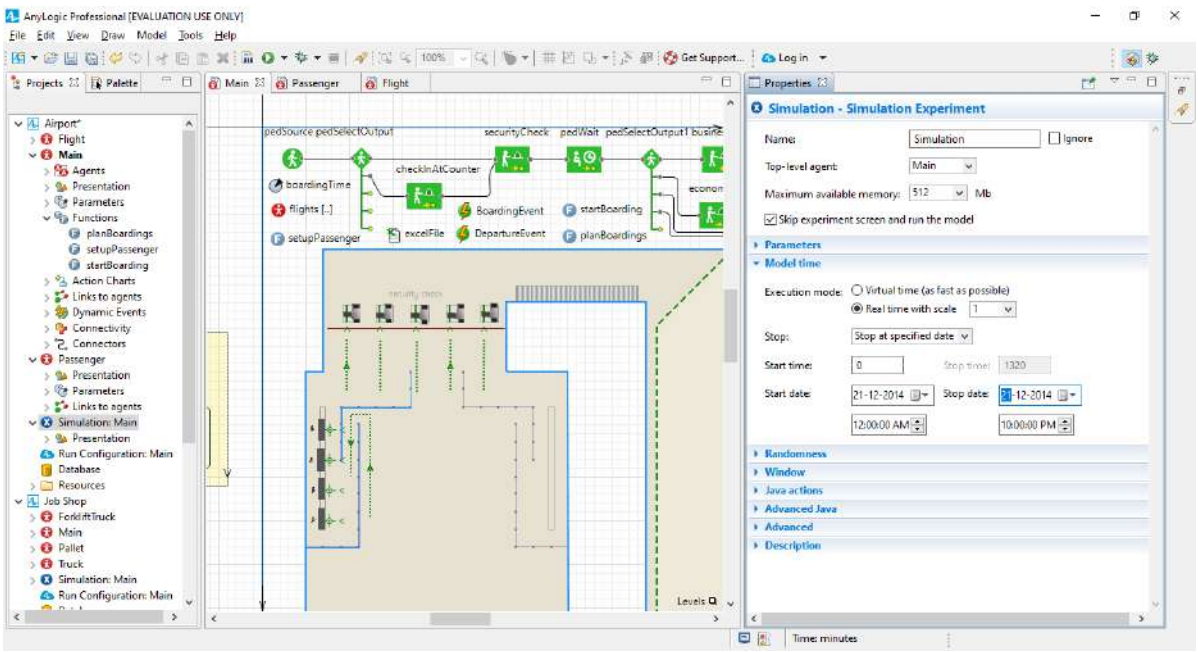


In Main's Agent actions area, in the On startup box, add the calls for the setupFlights() and planBoardings() functions.

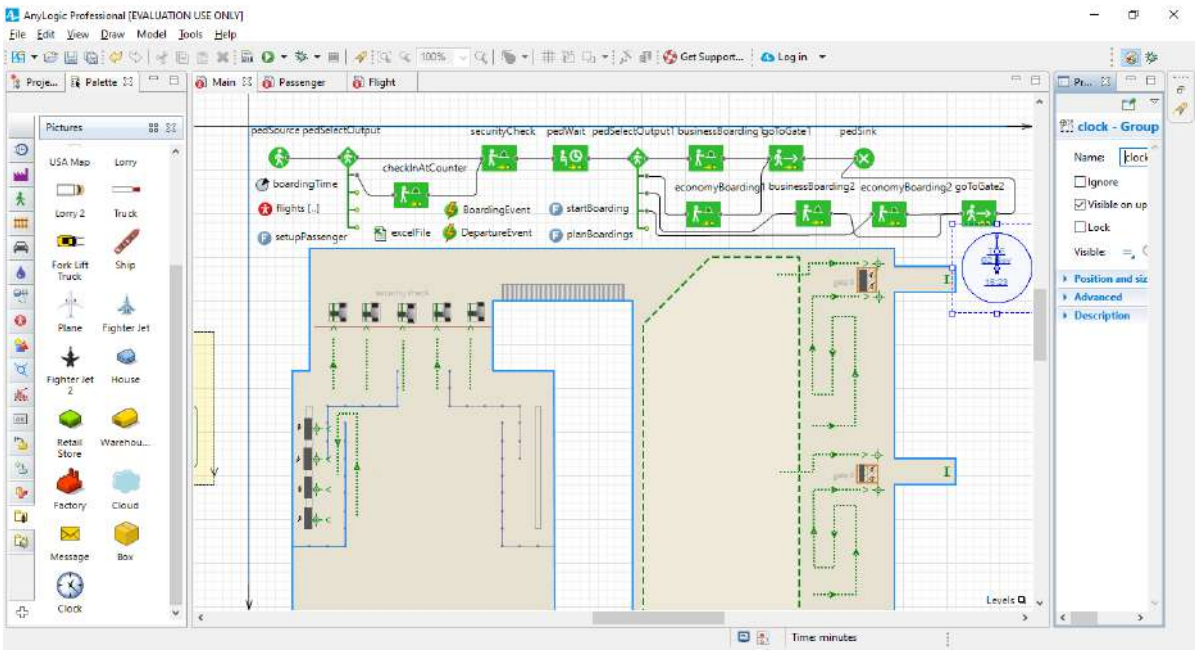


In the Projects, select Simulation. In the Model time section of the experiment properties, select the Use calendar box to ensure our model will work with actual calendar dates rather than abstract time, and then set the database's flight date as the start date.

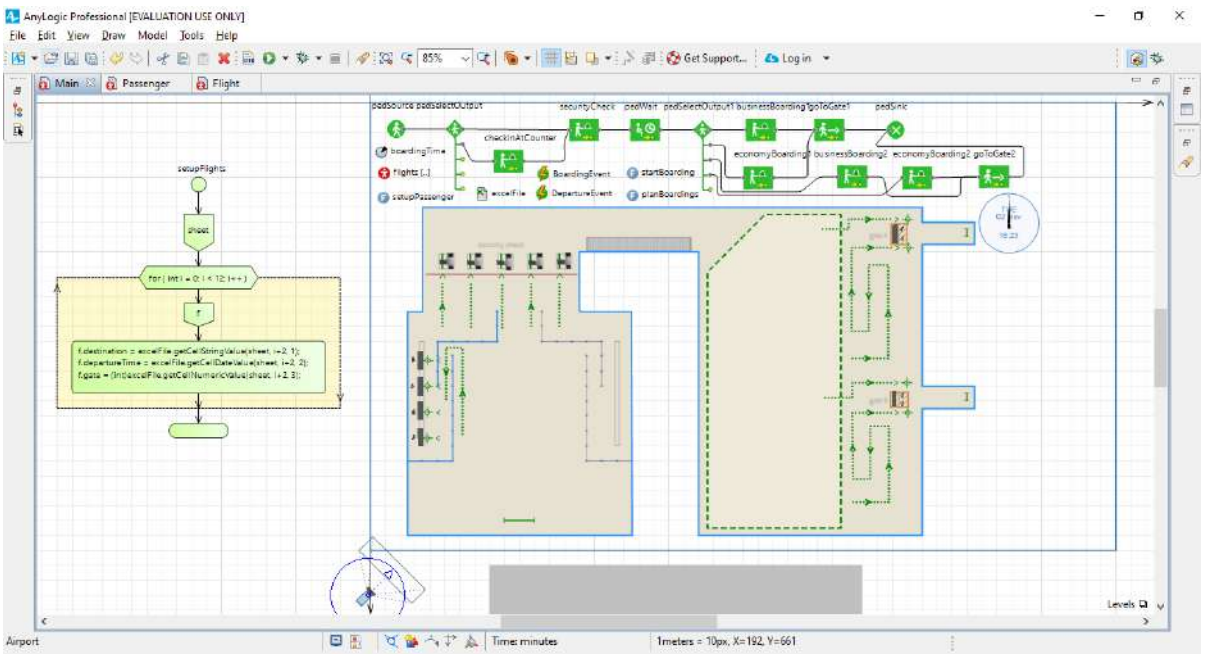
Set the Start date to 21/12/2014, 12:00:00. On the Stop list, click Stop at specified date and then set the Stop date to 21/12/2014, 22:00:00.

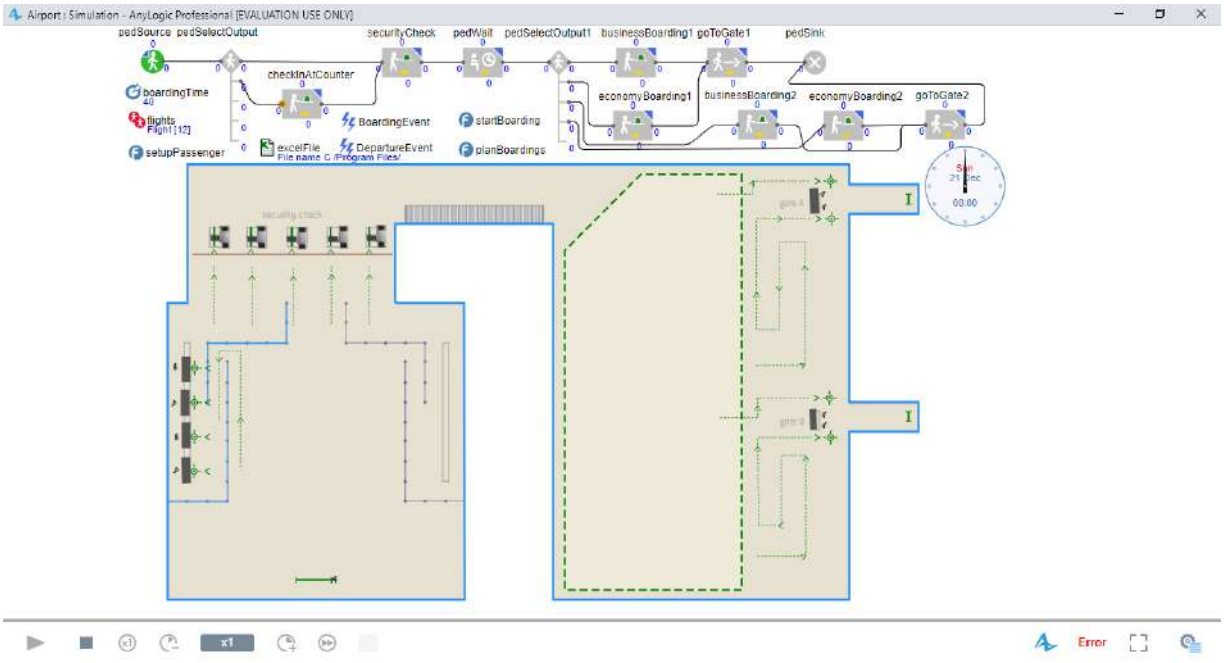


Use the Pictures palette to add a Clock element that will display the model date.



Run the model. You'll see passengers wait for the boarding announcement in the waiting area and then go to their gate.


AnyLogic Professional [EVALUATION USE ONLY]
File Edit View Draw Model Tools Help
Main
Passenger
Flight
setupFlights
checkInAtCounter
securityCheck
pedWait
economyBoarding1
businessBoarding2
economyBoarding2
goToGate2
pedSink
Airport
Time: minutes


Airport : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
pedSource
pedSelectOutput
securityCheck
pedWait
pedSelectOutput1
businessBoarding1
goToGate1
pedSink
checkInAtCounter
economyBoarding1
businessBoarding2
economyBoarding2
goToGate2
boardingTime
flights
setupPassenger
BoardingEvent
DdepartureEvent
startBoarding
planBoardings
excelFile
x1
Error