Simulation & Modelling

"Education through self-Help is our Motto"-KARMAVEER

RAYAT SHIKSHAN SANSTHA'S

Karmaveer Bhaurao Patil College

VASHI, NAVI MUMBAI – 400703

(Empowered Autonomous College)

DEPARTMENT OF COMPUTER SCIENCE

CERTIFICATE

This is to certify that Mr./Miss. _____

Student Of F.Y./S.Y./T.Y.B.Sc./M.Sc. Part-I/ M.Sc. Part-II-Computer Science Class From Karmaveer Bhaurao Patil College, Vashi, Navi Mumbai has satisfactorily completed the practical Course in Computer Science during the Academic year 2024-2025.

Roll No: _____

Exam No: _____

In charge Faculty | Examiner | Head

Date: / /2024 | Department of Computer Science

Index

4	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 scenario like spread of contagious disease for the purpose]		51	
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 manufacturing and shipping]		68	
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.		104	
7	Verify and validate a model developed like bank model or manufacturing model		138	
8	Create defense model to stimulate aircraft behavior		170	

Practical No: - 1

Aim: - Design and develop agent based model by

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

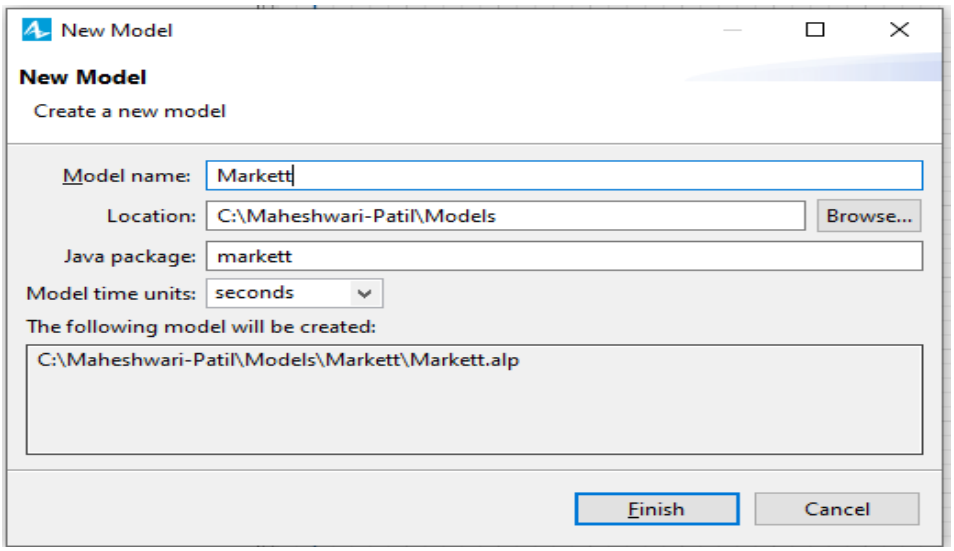
[Use a case scenario like grocery store, telephone call centre etc. for the purpose].

Code:-

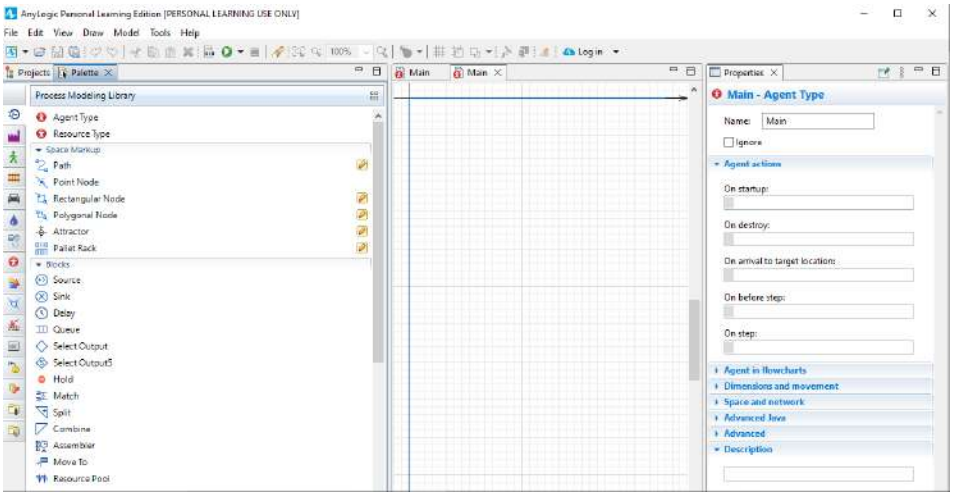
1. Creating the agent population:

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

Give the model's name and choose the location.

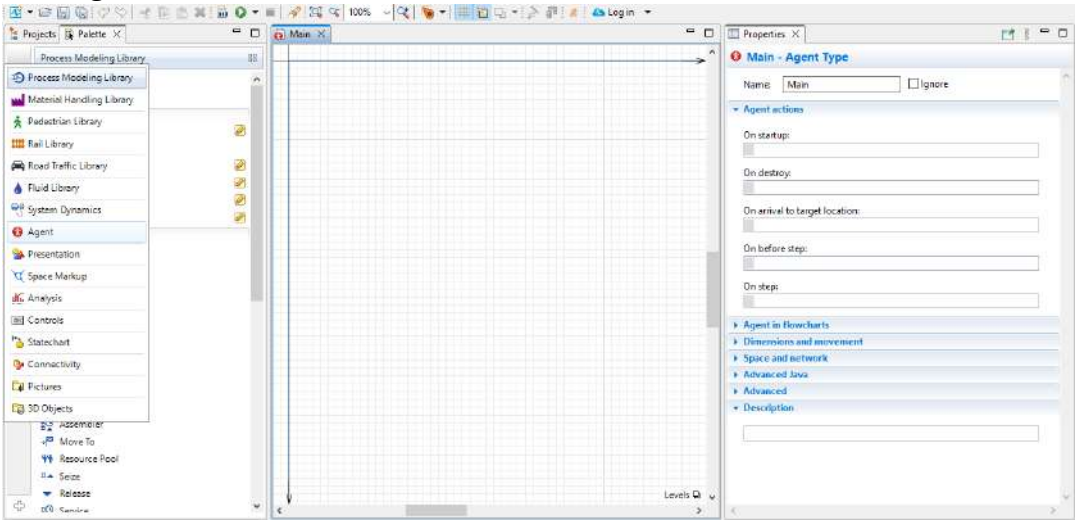


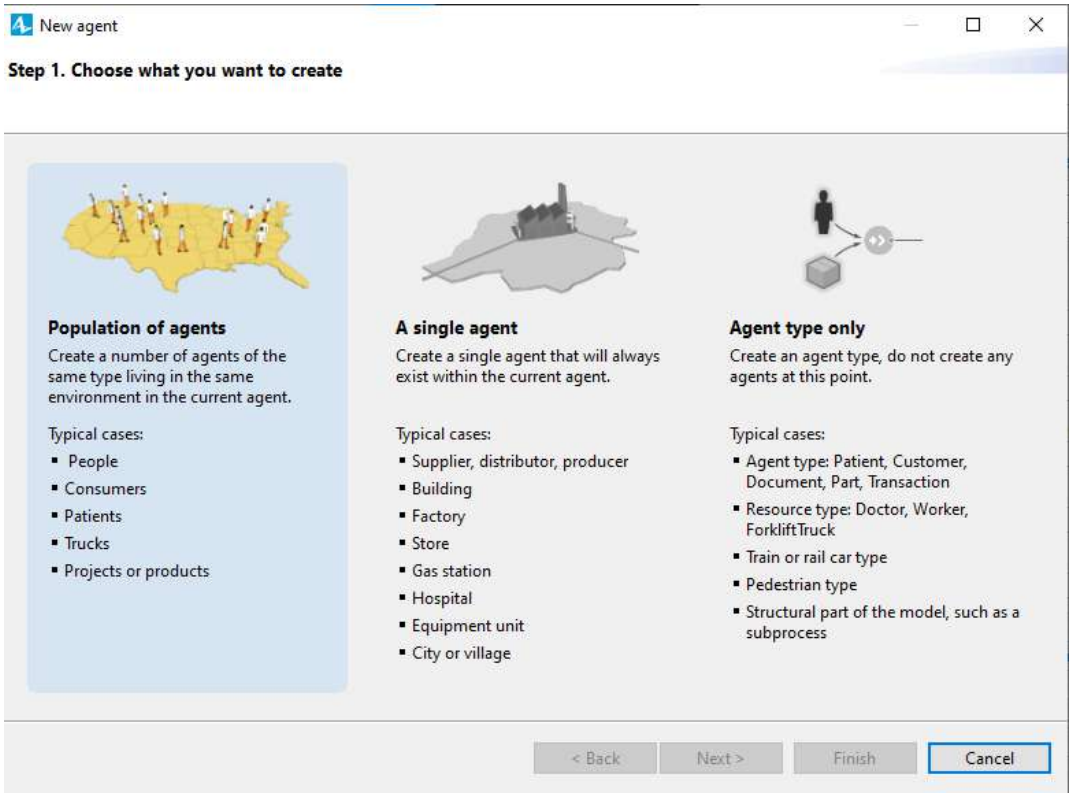
Click on finish.



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.

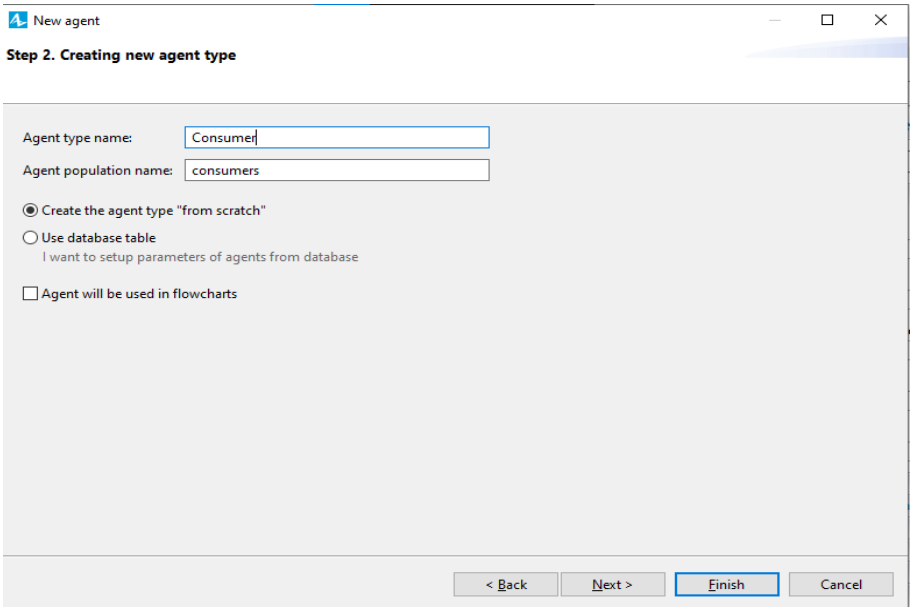
We want to add a new model element, but we first need to switch to the Palette view by clicking the Palette tab.



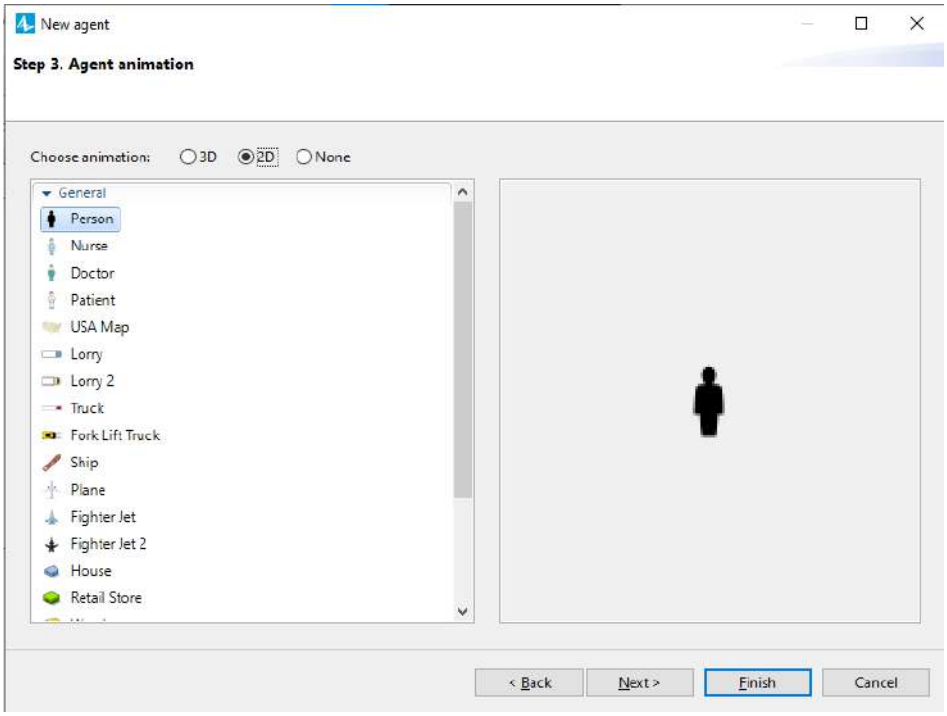


Step 1. Choose what you want to create page, select the option that best meets your needs. Since we want to create multiple agents of the same type, select Population of agents and click Next.10

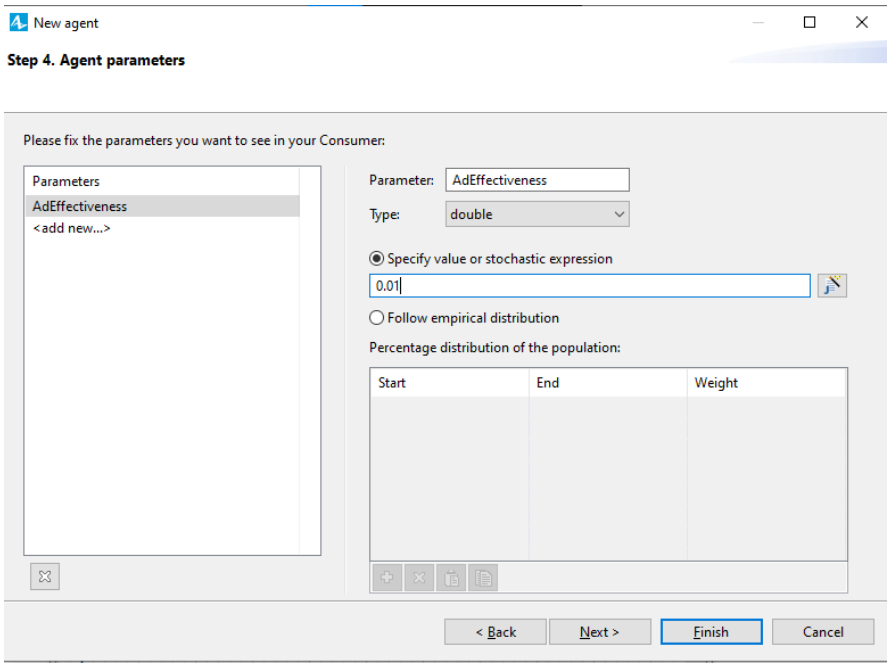
Step 2. Creating new agent type page, in Agent type name box, type Consumer. The information in the Agent population name box will automatically change to consumers



Click on 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.

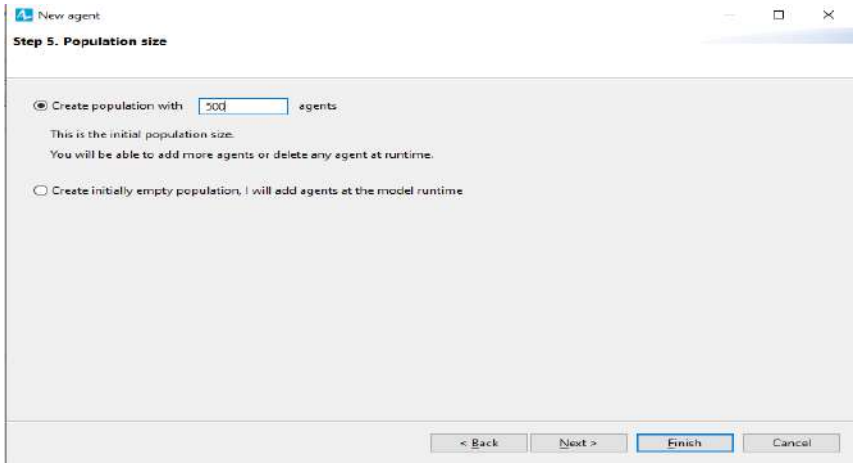


Click on Next.
On the Agent Parameters page, define the agent's parameters or characteristics. Since our model only considers advertising-related product purchases, we'll add a parameter – AdEffectiveness – to define the percentage of potential users who become ready to buy the product during a given day. On the left section, in the Parameters table, click to create a parameter.



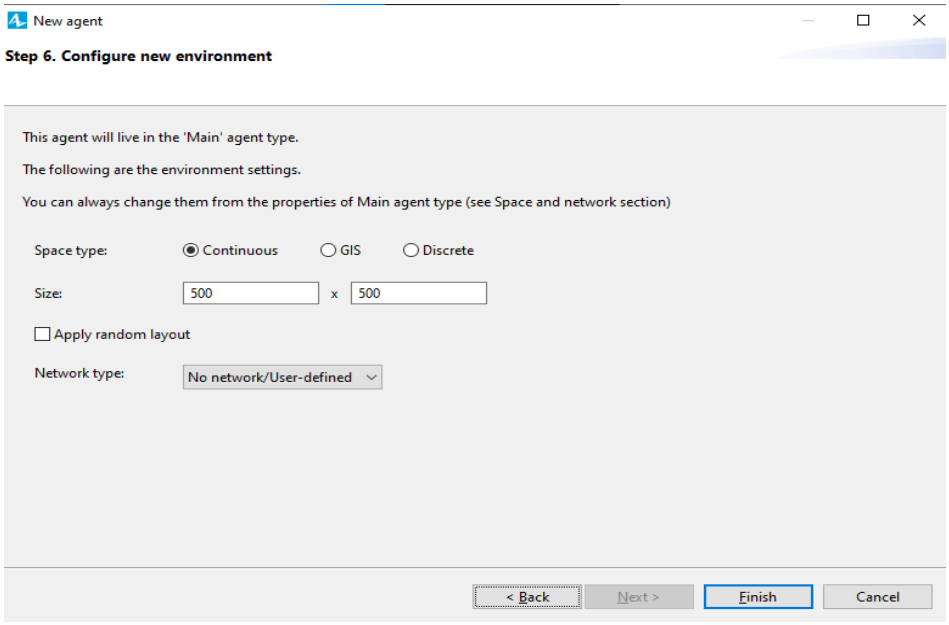
Click on Next. On the Population size page, type 500 in the Create population with …agent's box to create 500 instances of the Consumer type. Each instance in the population will model a specific agent consumer.

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

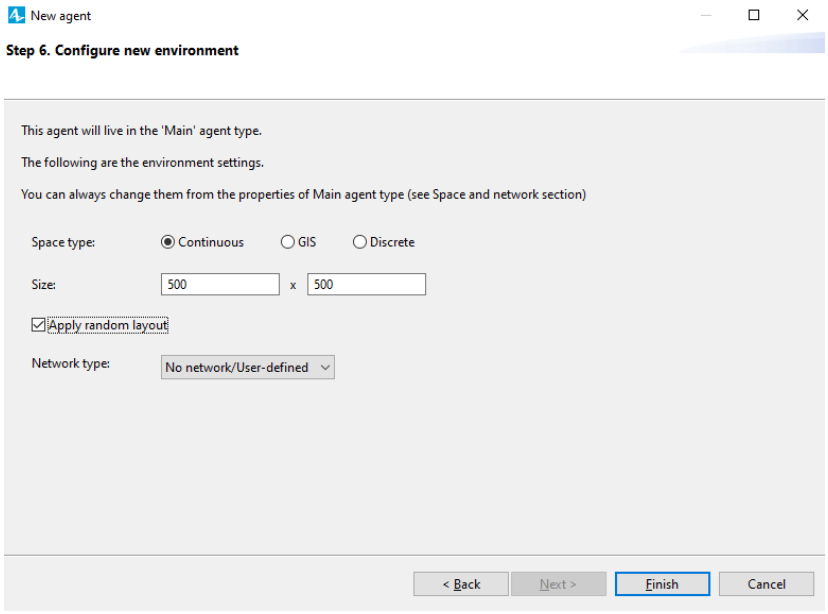


Click Next.

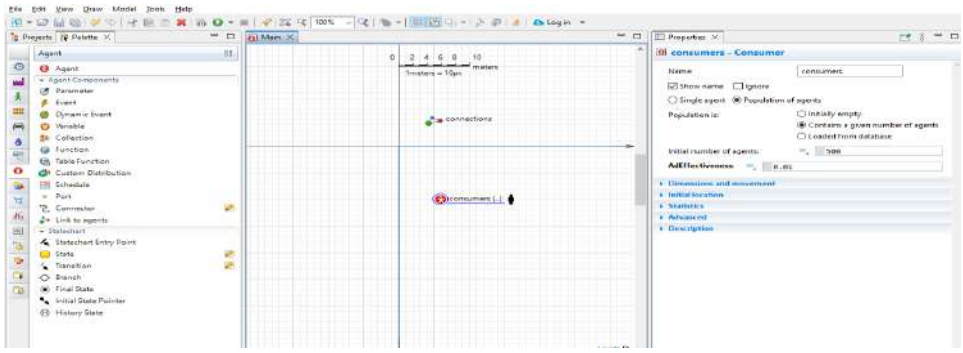
On the Configure new environment page, accept the default values for the environment's space type (Continuous) and both its Width and Height values (500). AnyLogic will display the agents in a 500x500 pixel rectangle.



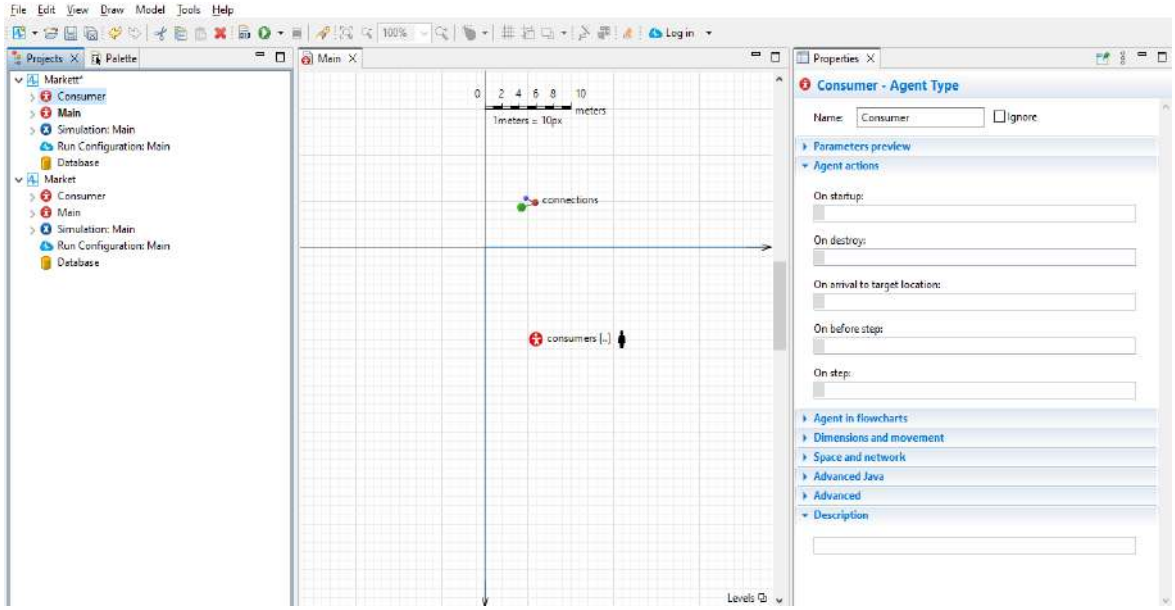
Select the Apply random layout box to randomly distribute the agents across the 500 pixel width and height we've defined. Since we don't want to create an agent network, we'll accept the default No network/User-defined network type.



Click on Finish.



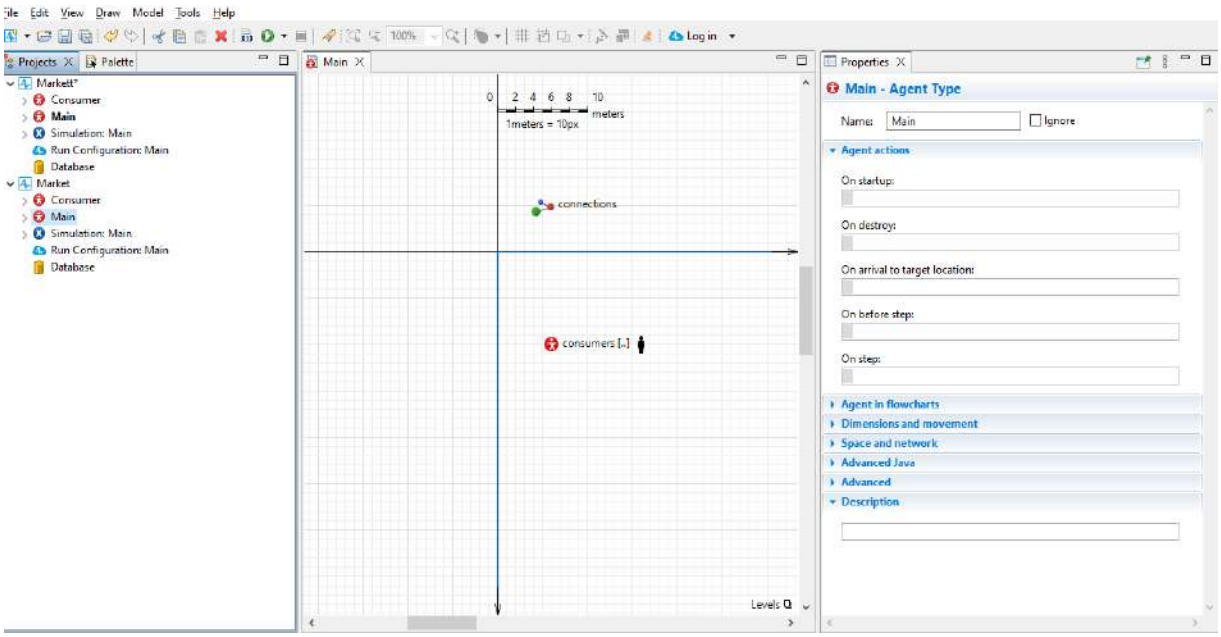
Let's use the Projects view to see the new elements that the wizard created. Expand the model tree branches to see the internals.



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

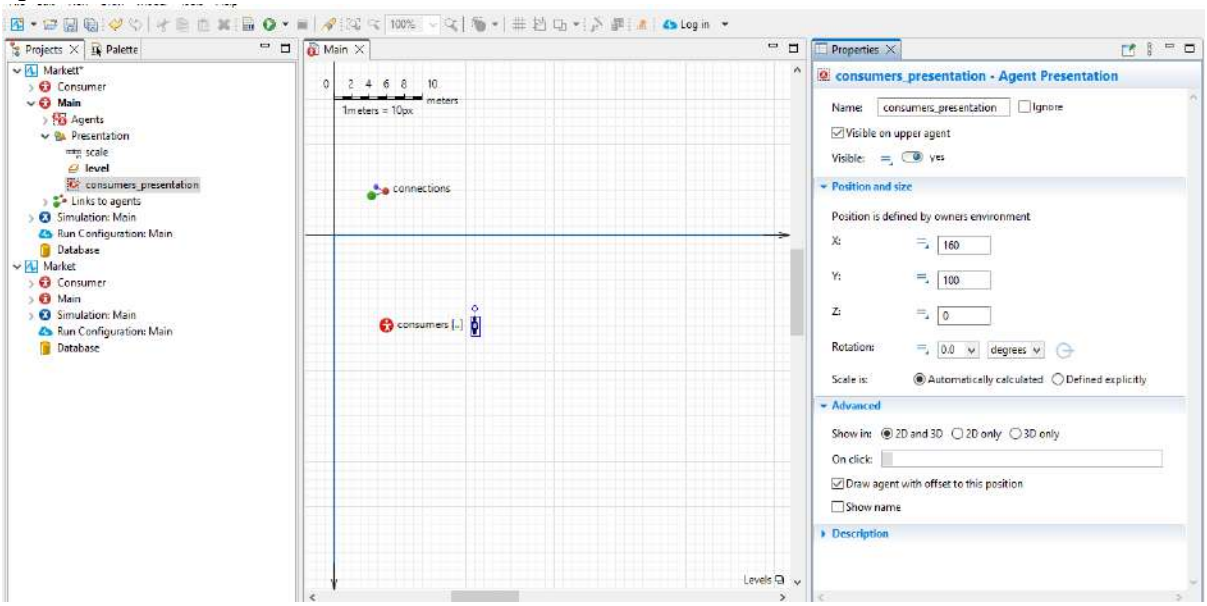
• The Consumer agent type has the agent's animation shape (person, in the Presentation branch) and the parameter AdEffectiveness.

• 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 Any Logic window's right half).



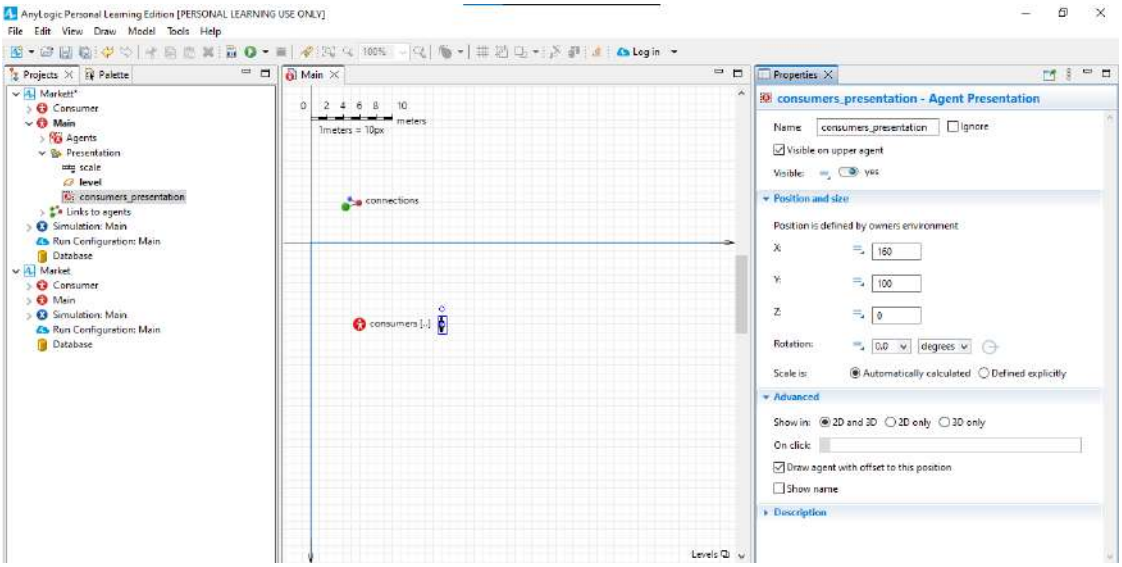
In the Space and network section of Main properties, you can adjust the environment settings for the consumer's agent population.

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.

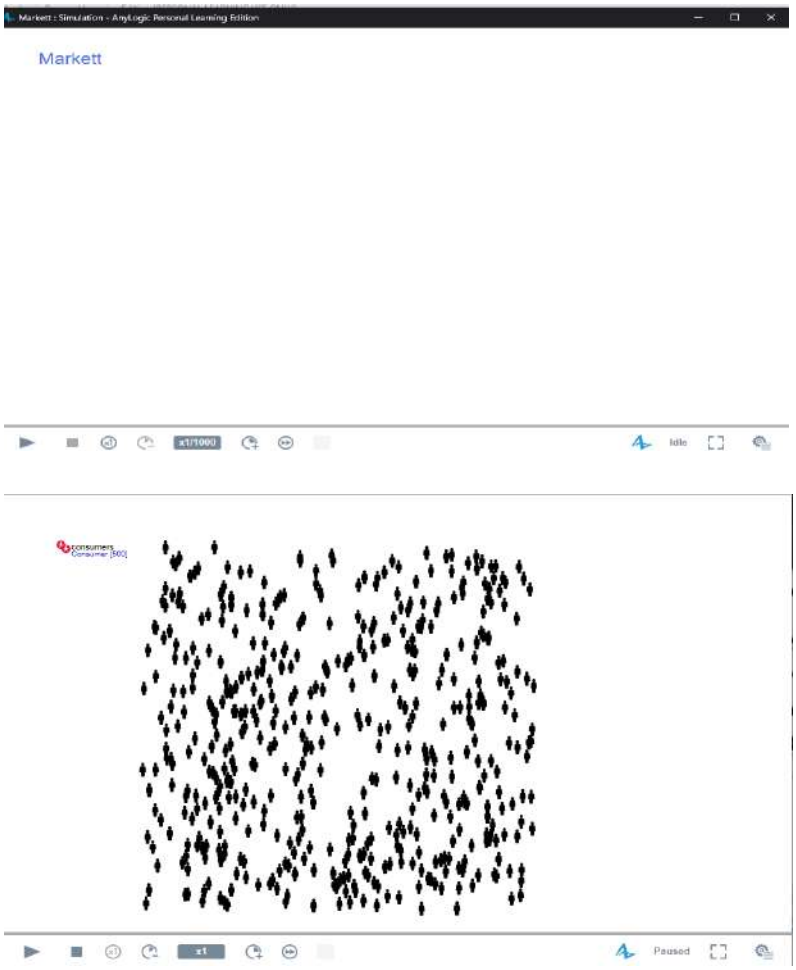


We've finished building this very simple model, and you can now run it and observe its behavior.

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

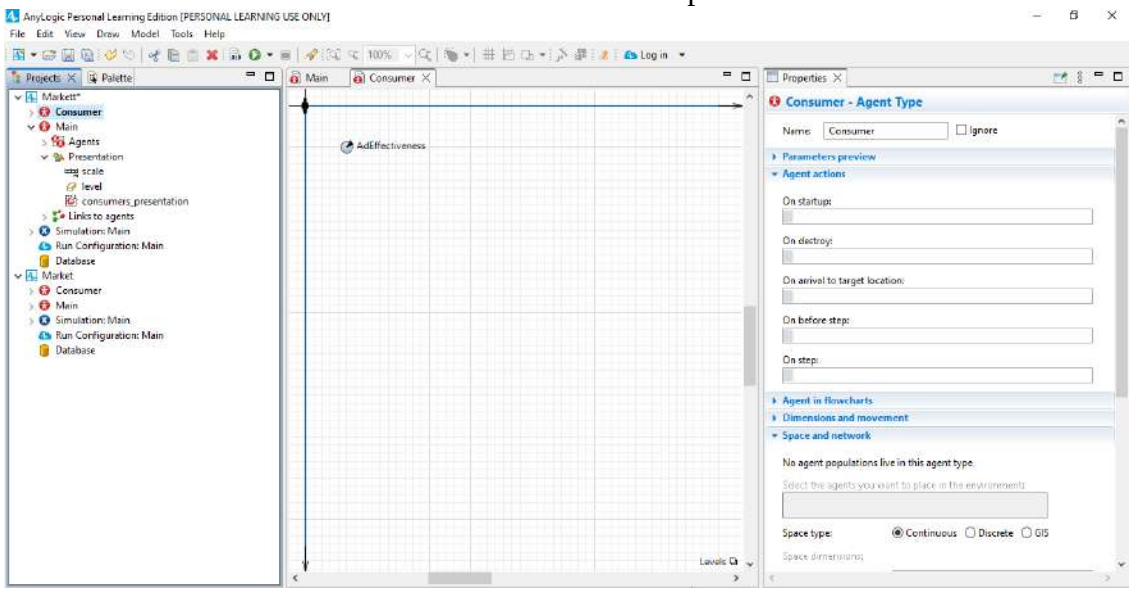


Locate the Run button and click the small triangle to the right. Select the experiment you want to run. Choose Markets / Simulation from the list.

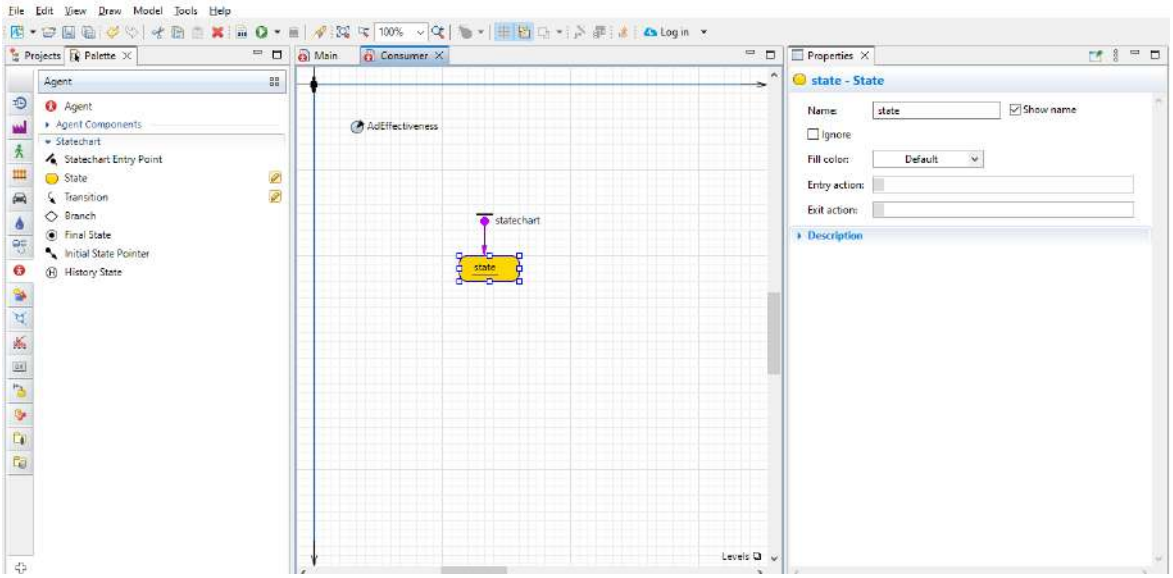


2. Defining Consumer Behaviour

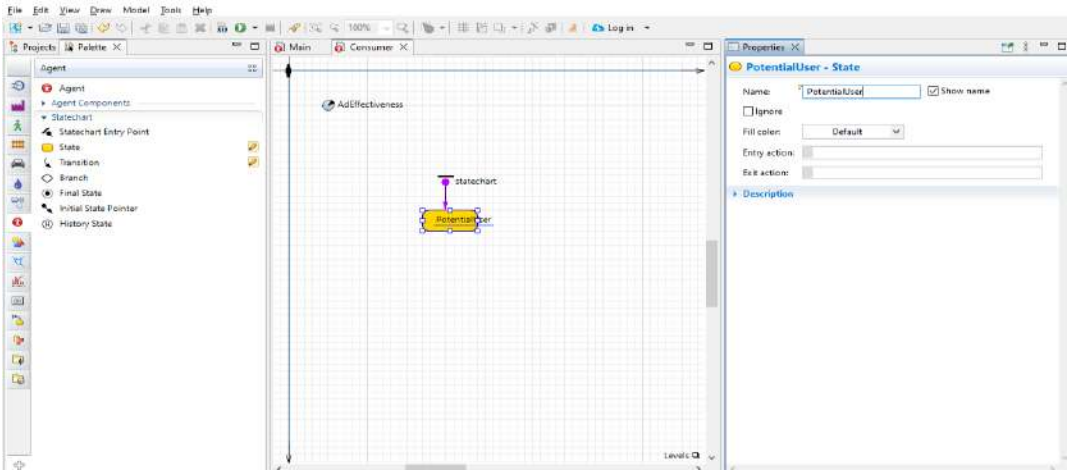
Double click on consumers a new tab of consumers open



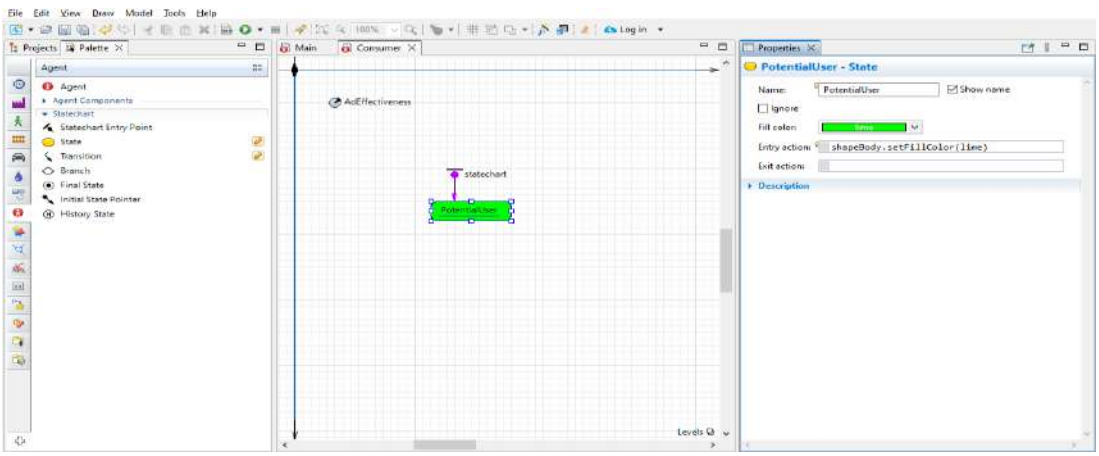
Drag the State and Statechart Entry Point from the Statechart palette on to the graphical diagram and connect it to the statechart entry point.



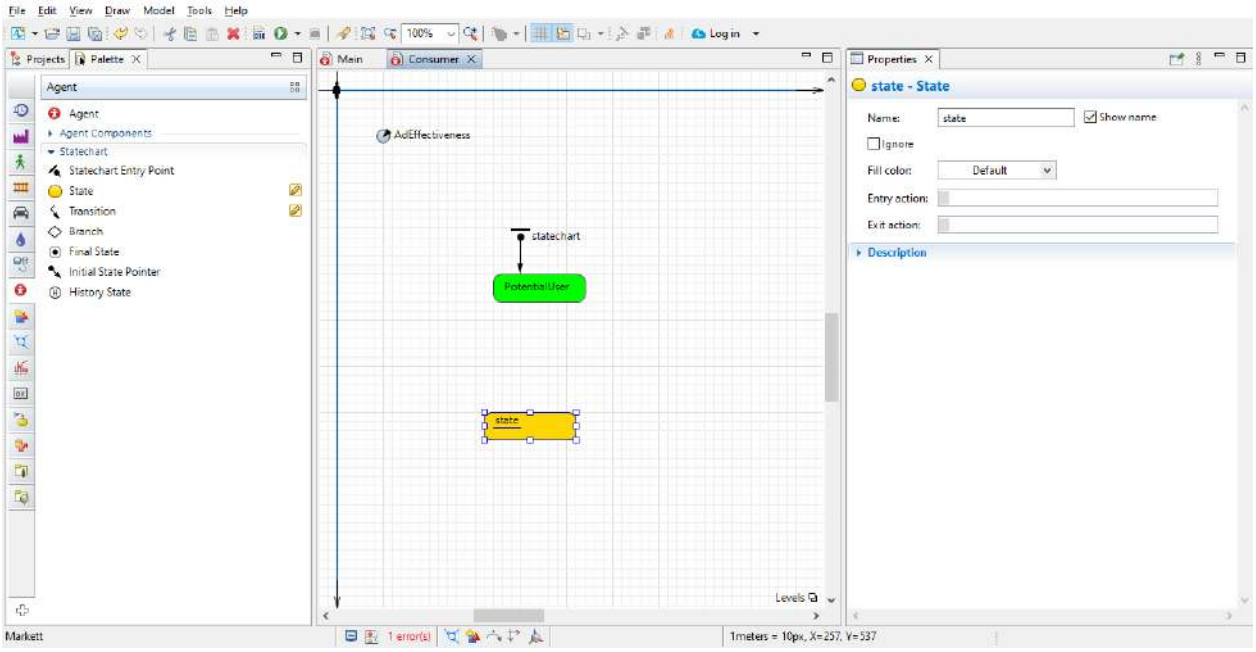
Select the state in the graphical editor and modify its properties. Name the state PotentialUser.



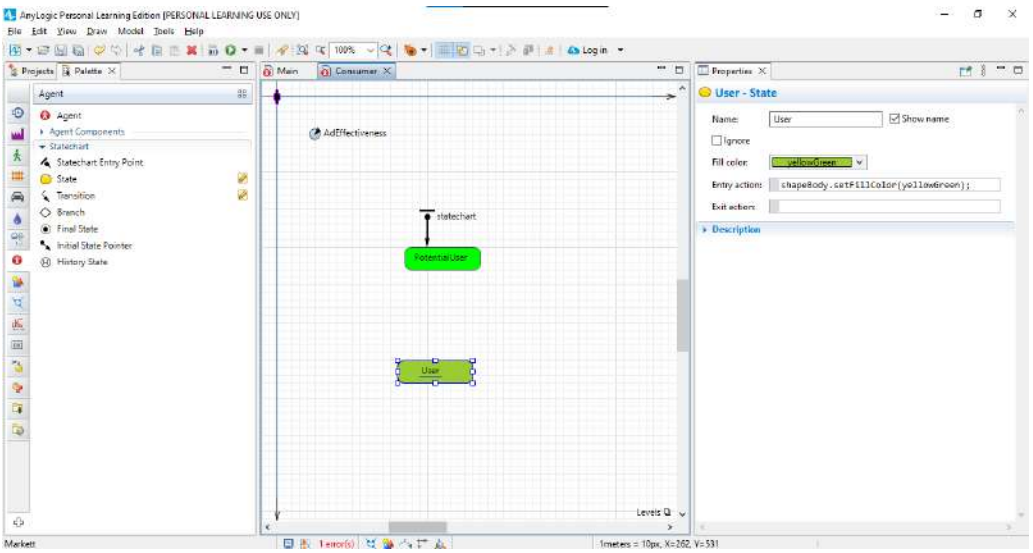
Use the Fill color control to change the state's color to lavender.Type the following Java code in the state's Entry action field:shapeBody.setFillColor(lime)



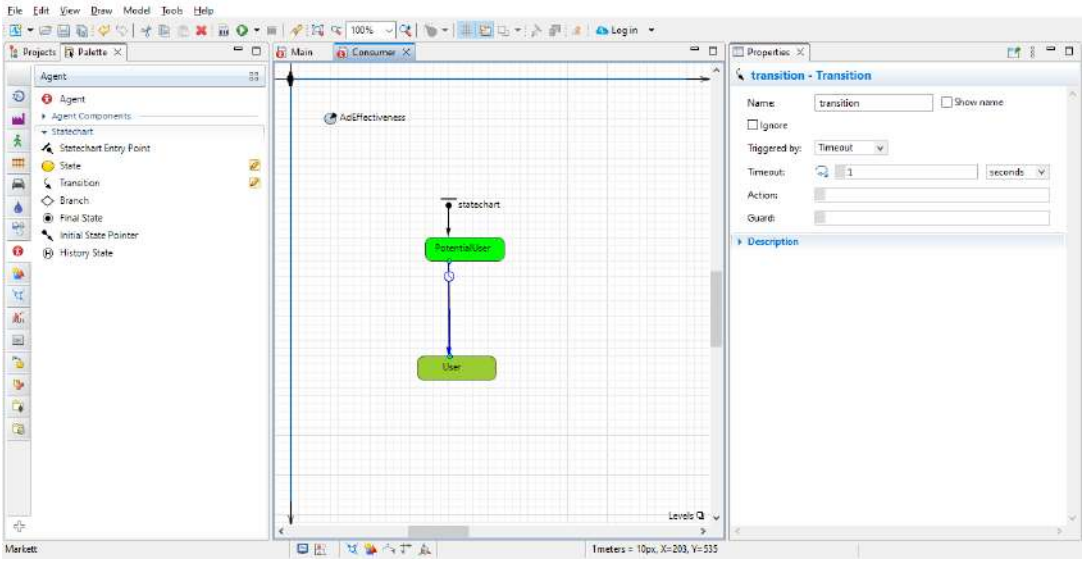
Add another state in the consumer's statechart:



Modify the state's properties like you did earlier:
Name: User
Fill color: yellowGreen
Entry action: shapeBody.setFillColor(yellowGreen);



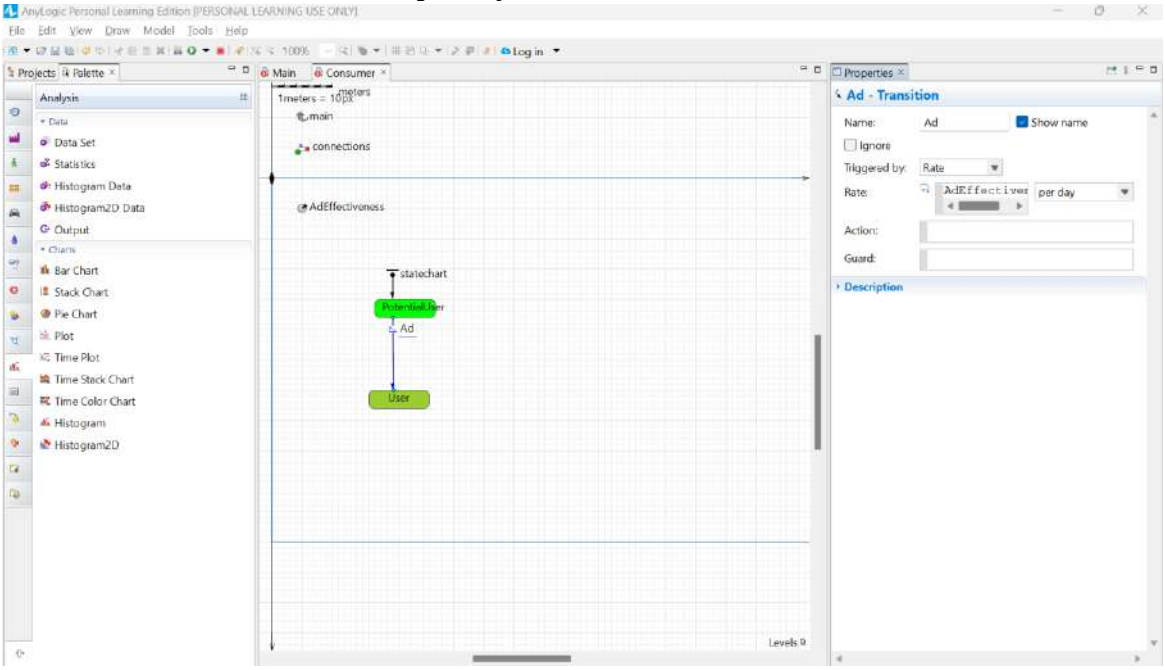
Draw a transition from PotentialUser to User state to model how persons purchase the product and become product users.
To do so, double-click the Statechart palette's Transition element (the element's palette icon should change to), click PotentialUser state, and click User



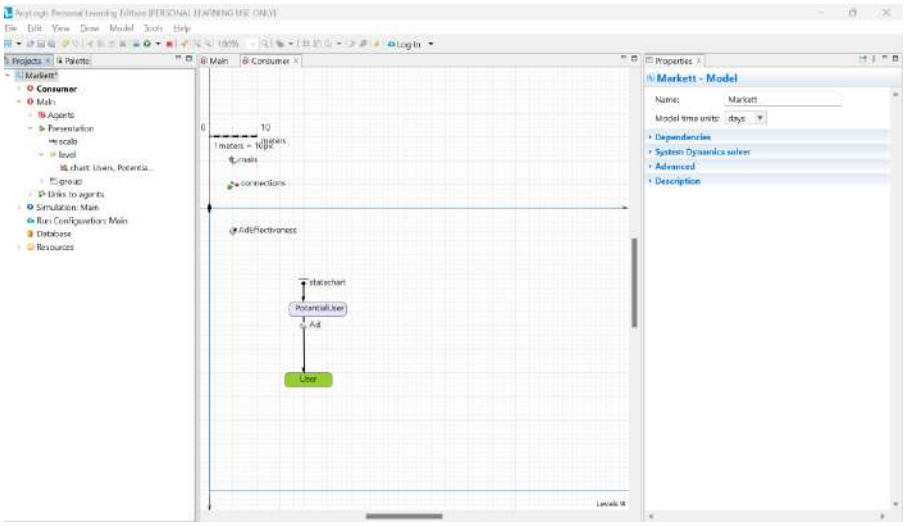
Name the transition Ad to represent “advertising”.

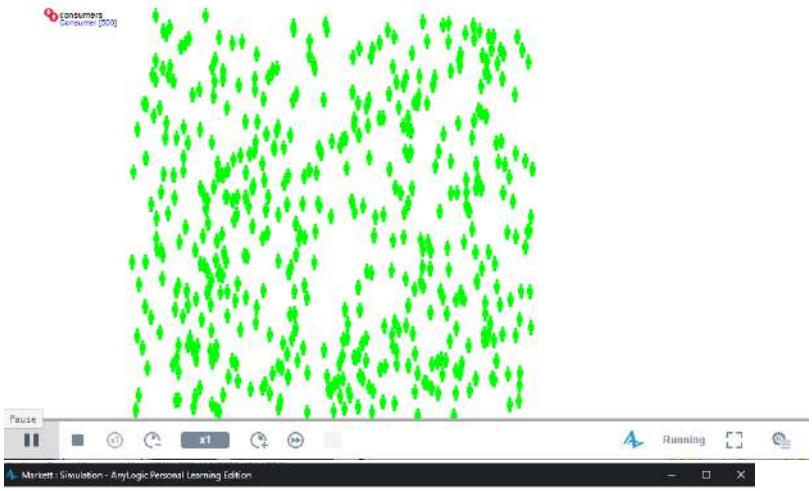
12. Select the Show name checkbox to display the transition’s name on the graphical diagram.

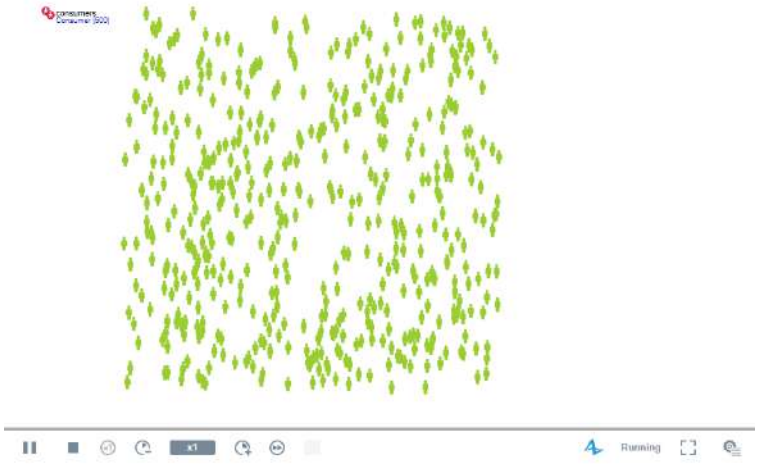
13. The transition from PotentialUser to User state will model how advertising leads the person to buy the product. In the Triggered by list, click Rate. In the Rate field, type AdEffectiveness, and then click per day.



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



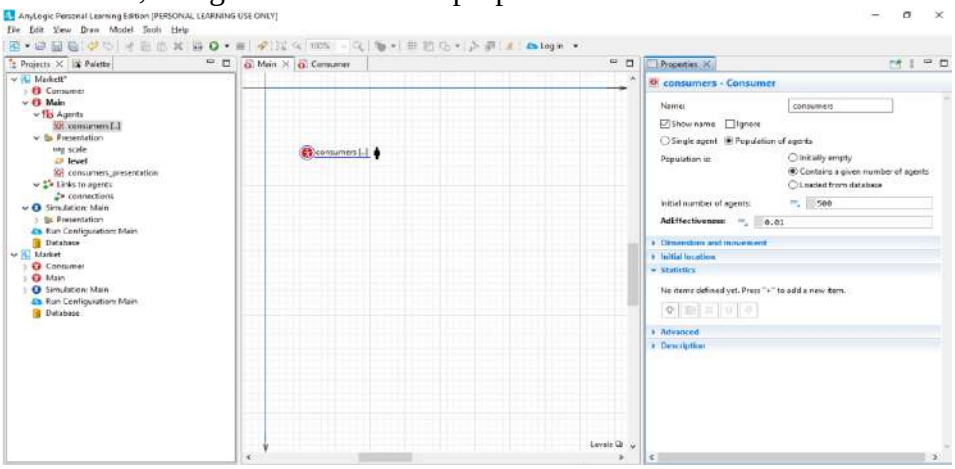




3. Adding a chart to visualize the model

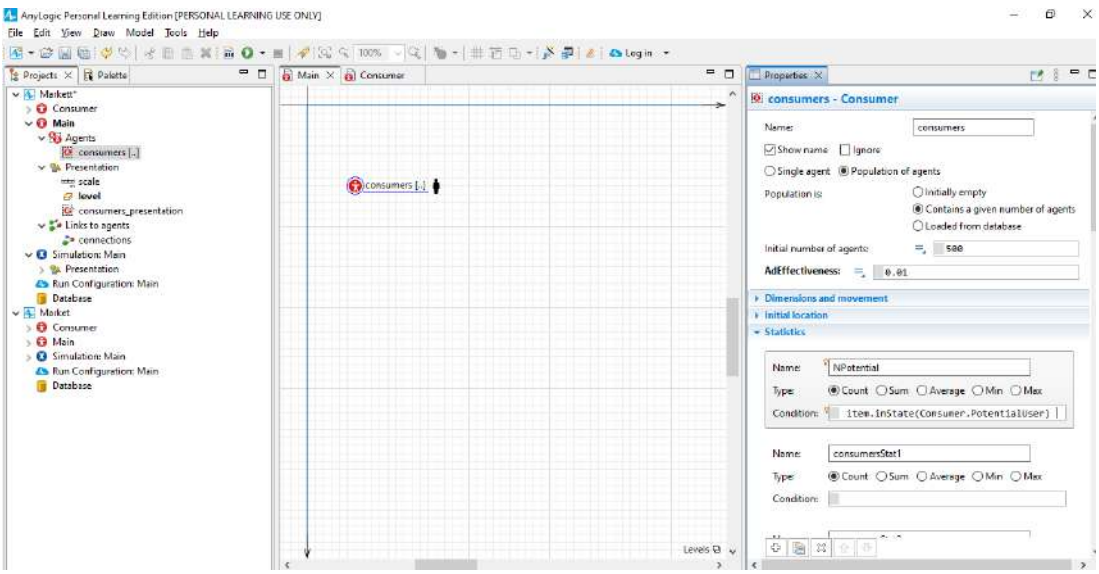
Output First, define a function to count potential users. To add a new function that collects statistics for agents, open the diagram of the agent type Main, select the agent population

consumers, and go to the Statistics properties section.



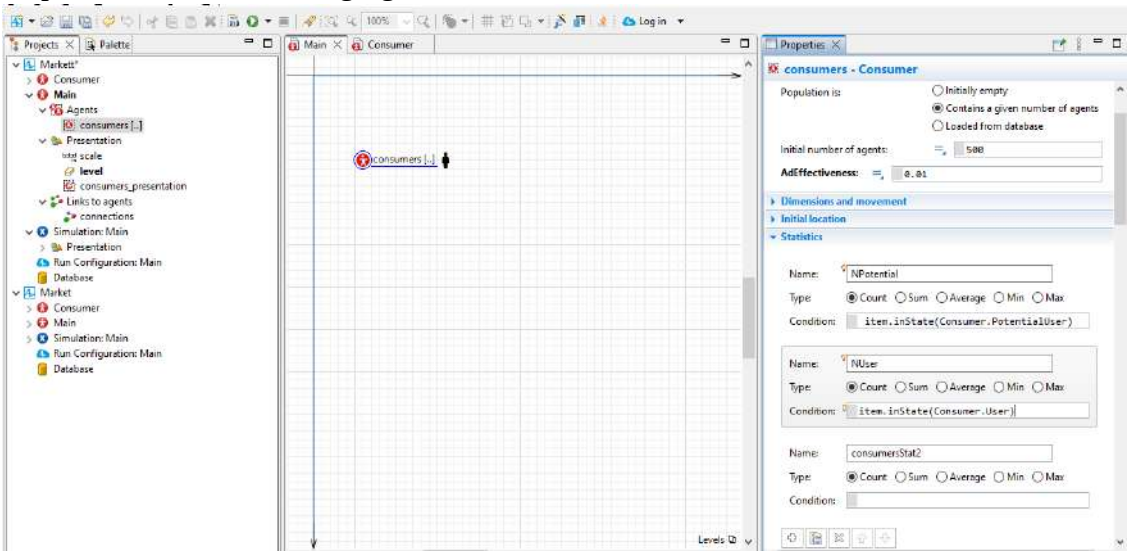
Define the function of type Count with the Name NPotential. The statistics of type count iterates through a given population – in our case, the number of agents – to count those that meet the selected condition.

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



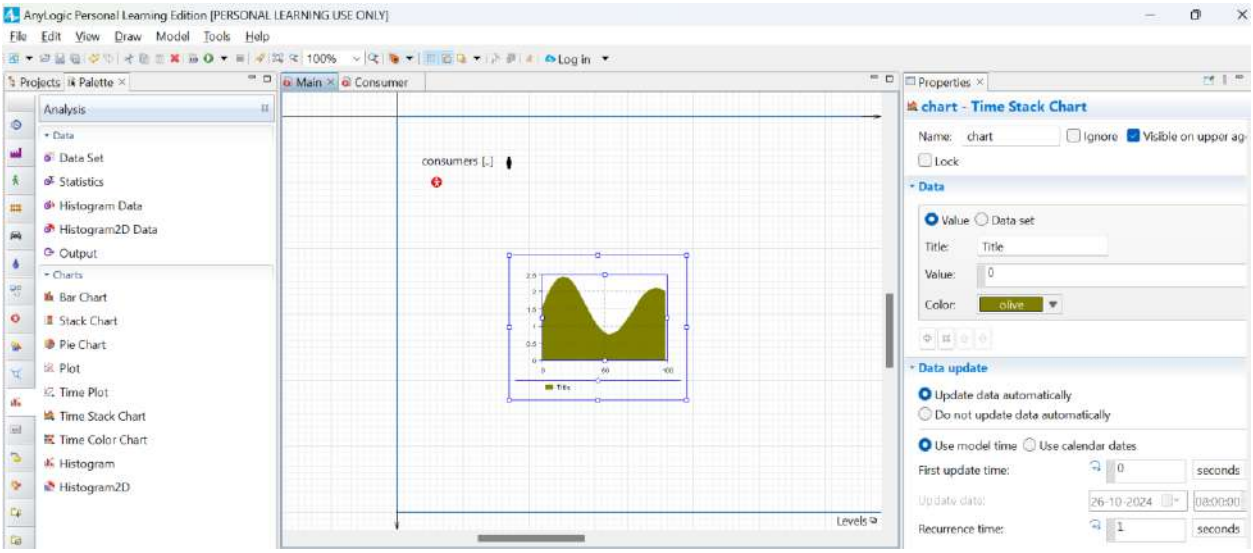
Define a second statistics function to calculate the number of product users. Name it NUser and let it count the number of agents, conforming the Condition item.inState(Consumer.User). You can duplicate the other statistics function by clicking the

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.

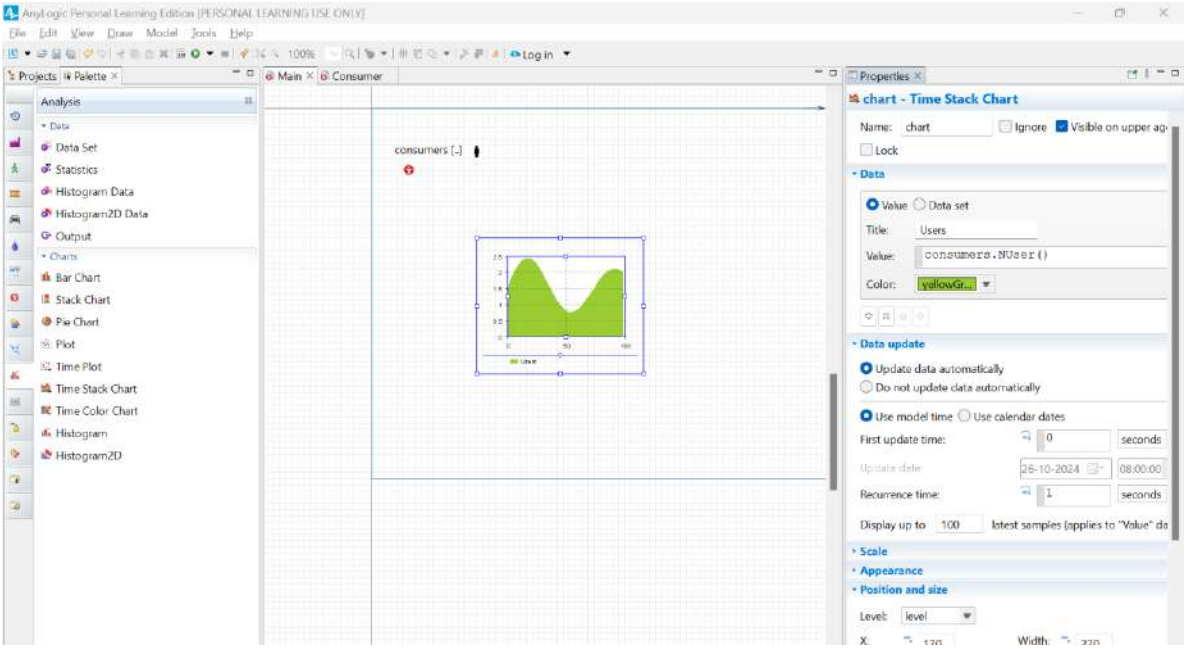
Open the Analysis palette and drag the Time Stack Chart from the Analysis palette on to the Main diagram to create a chart that will display the dynamics of users and potential users. Increase the time stack chart as shown in the figure below:



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 defined for this population.

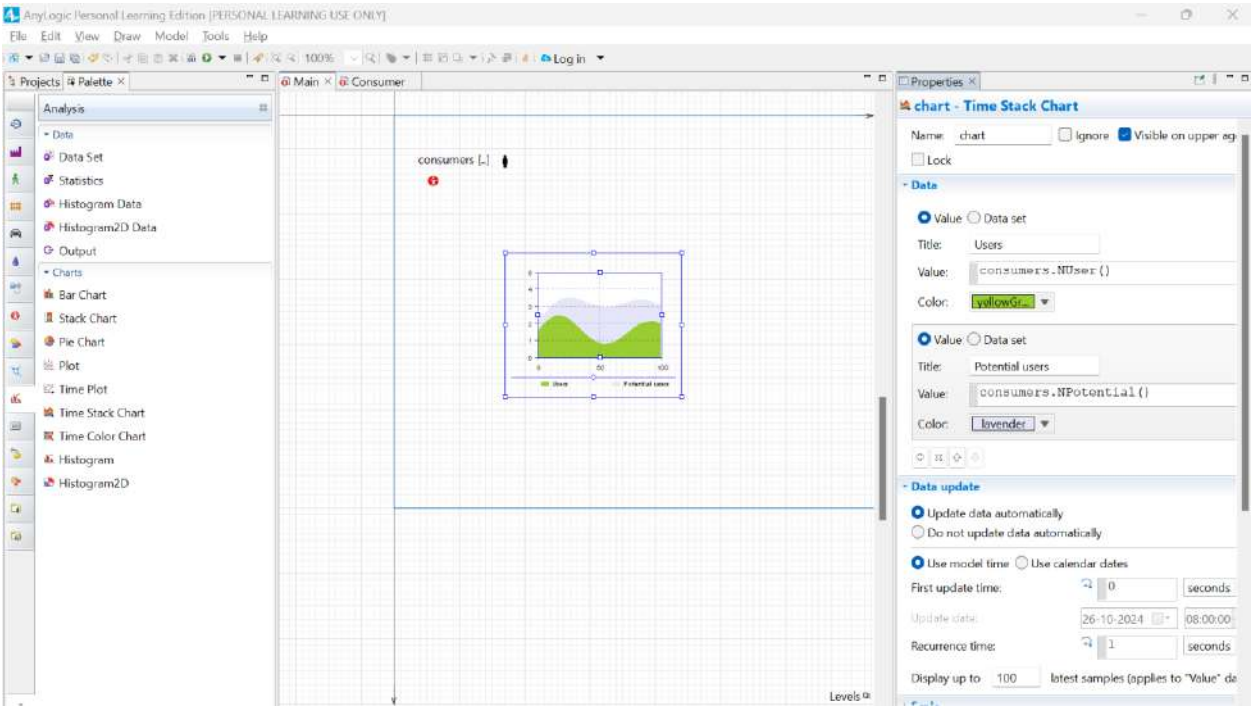


Add one more data item:

• Title: Potential users

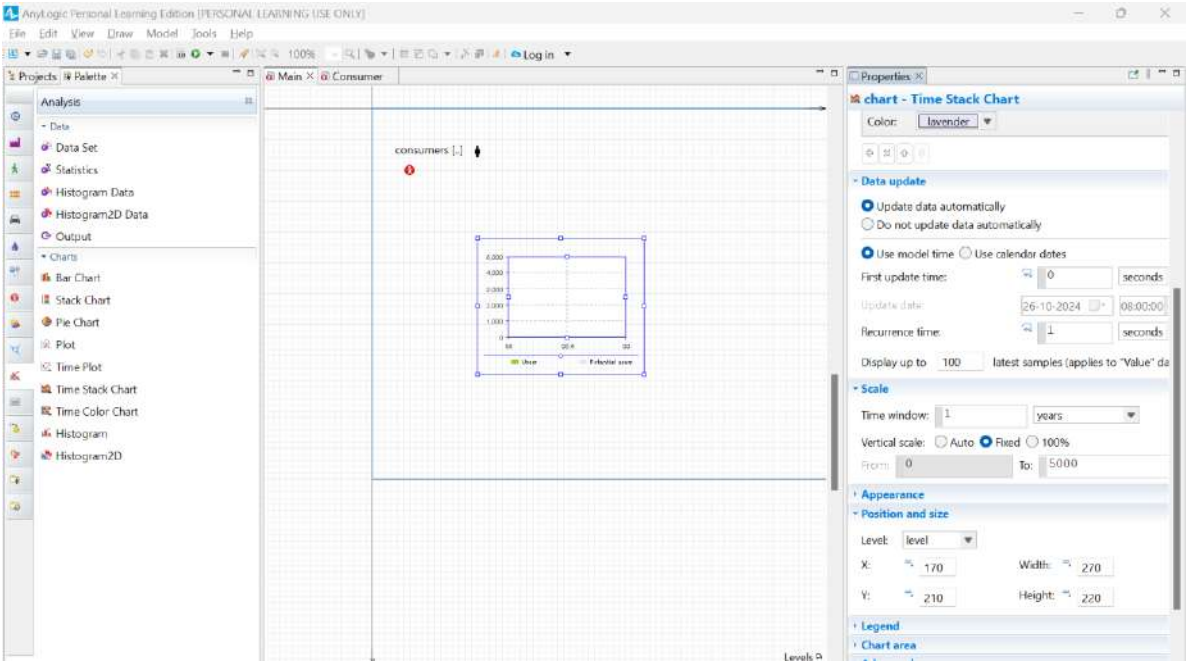
• Color:levender

• Value: consumers.NPotential()

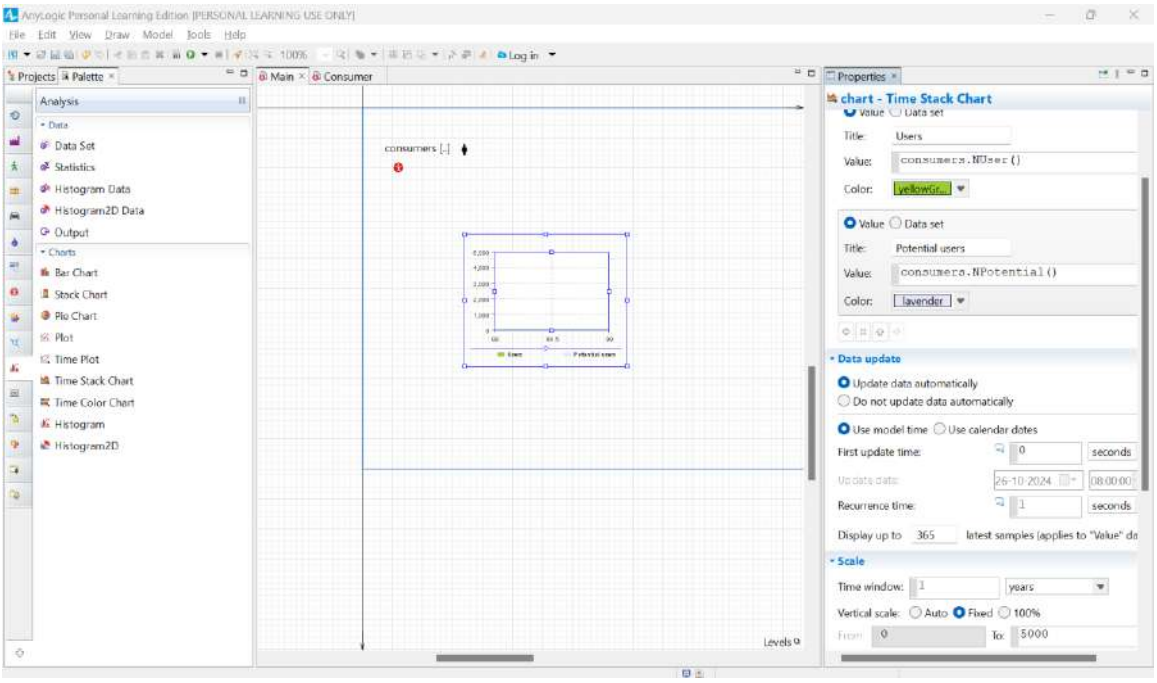


Go to the Scale section and set Time window equal to 1 year. Since our chart will show statistics for consumers population and our model has 500 consumers, set the chart's Vertical

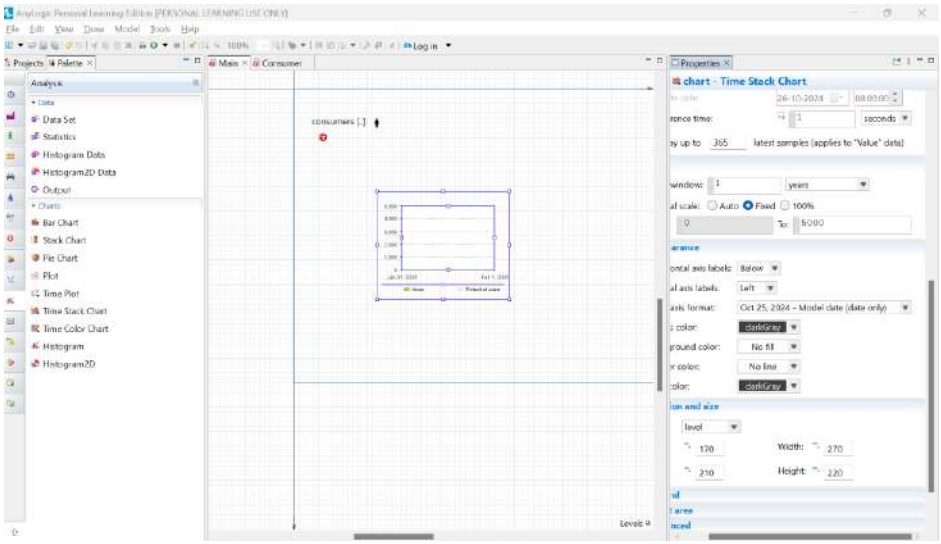
scale to Fixed, and enter 500 in the to: box.



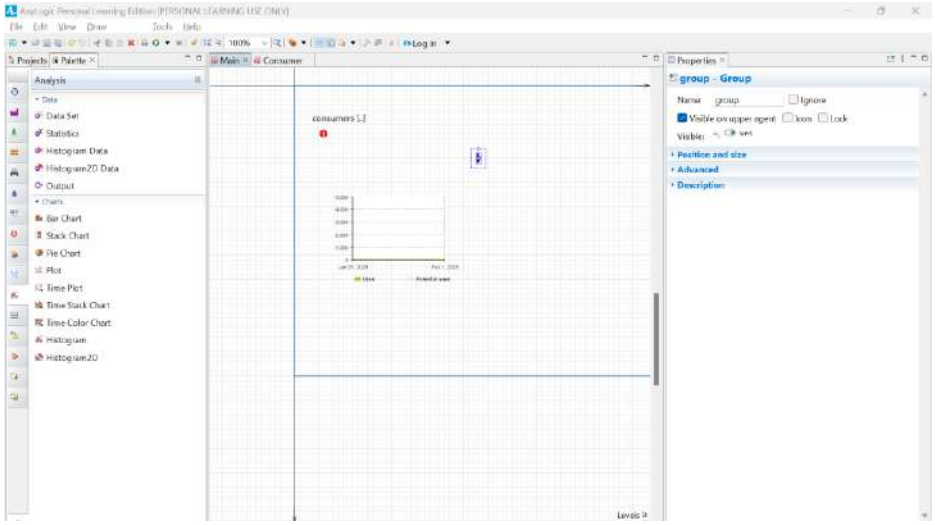
Now that we've set the time window, change the maximum number of data samples that the chart displays by navigating to the section Data update and setting Display up to 365 latest samples. Since we'll add one data sample each day, 365 data sample is an ideal amount for a one year range.



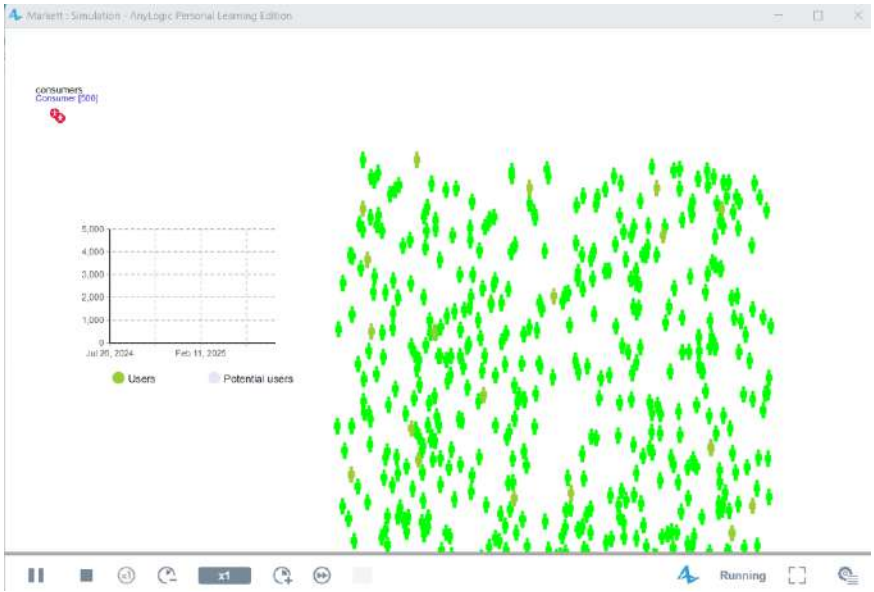
Go to the time stack chart's Appearance properties and set it to display Model date (date only) near the time axis.



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.



Practical No: - 2

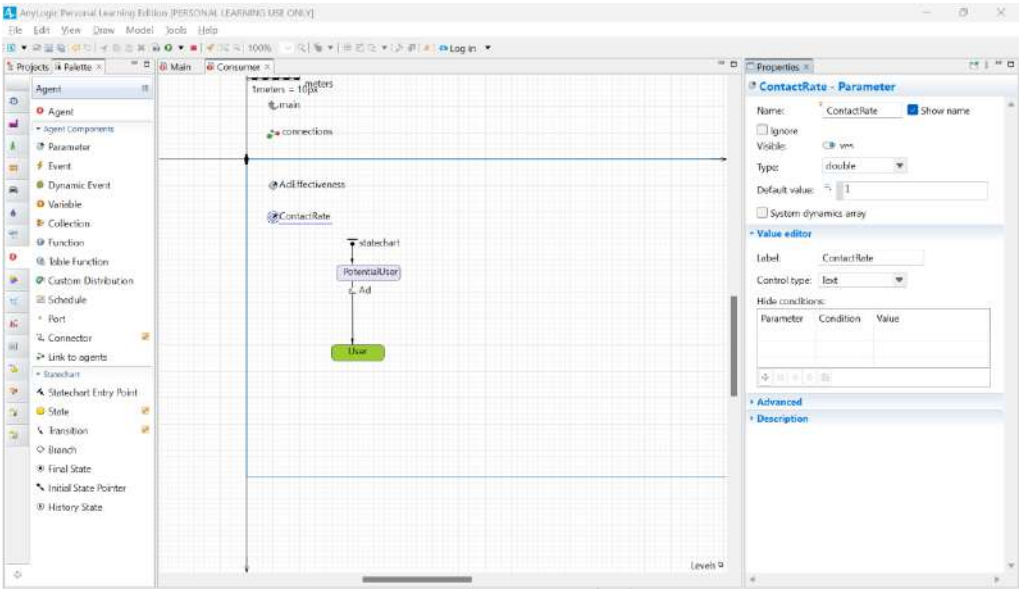
Aim: - Design and develop agent-based model by

- Creating the agent population
- Defining the agent behaviour
- Adding a chart to visualize the model output
- Adding word of mouth effect
- considering product discards
- considering delivery time

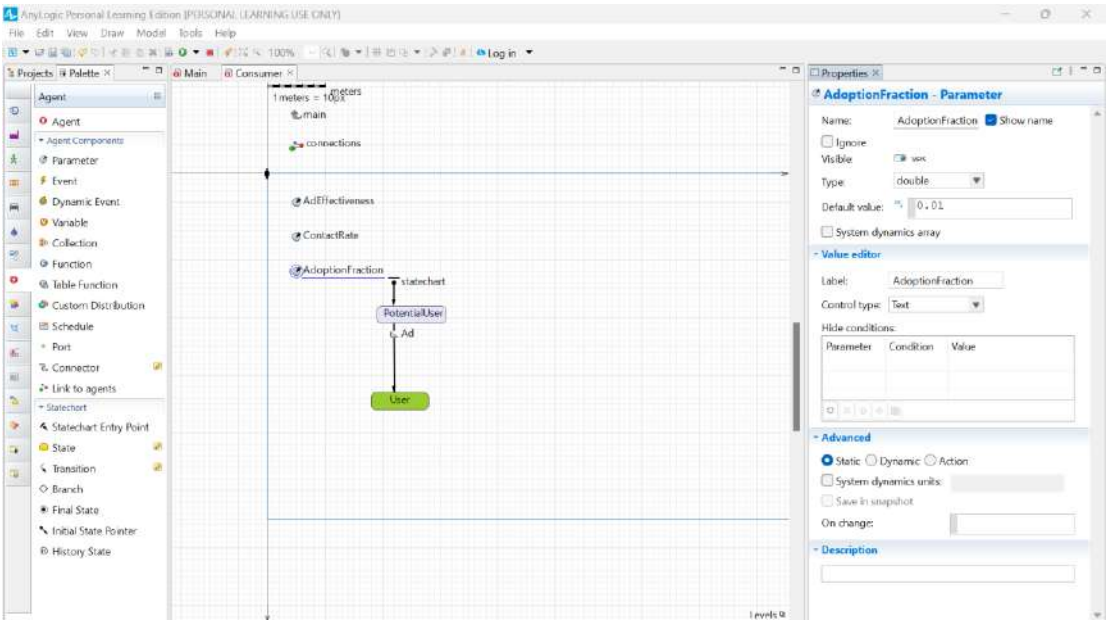
Code:-

Adding word of mouth effect

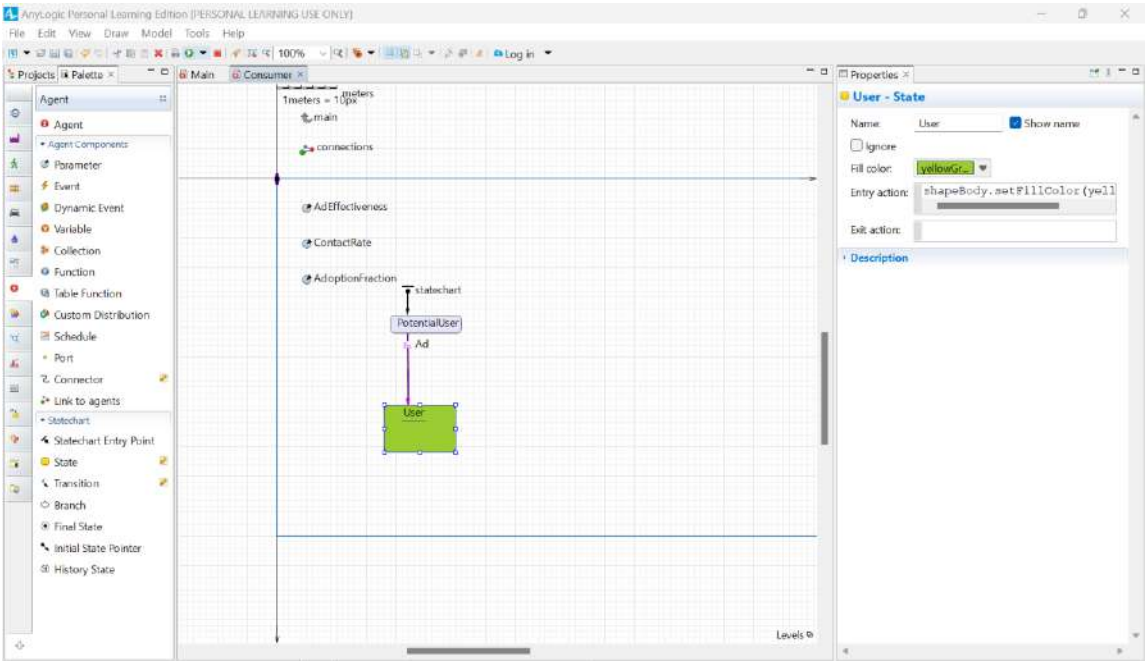
In the Projects tree, open Consumer diagram by double-clicking on Consumer.
Add a parameter to define a consumer's average daily contacts. Drag the Parameter from the Agent palette on to the diagram.
The rate is 1 contact per day, so type 1 as the parameter's diagram.
Name the parameter ContactRate default value.



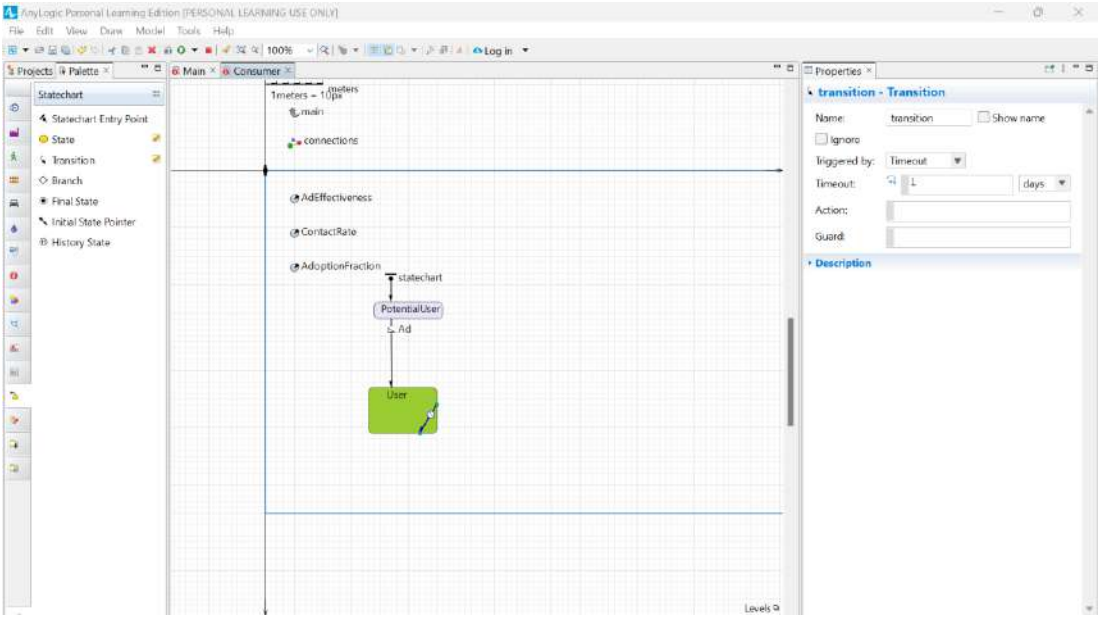
Add another parameter - AdoptionFraction - to define a person's influence on others, a number
that we'll express as the percentage of people who will use the product after they come into contact with the consumer. Leave the default parameter's Type: double and set the Default value: 0.01.The Consumer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



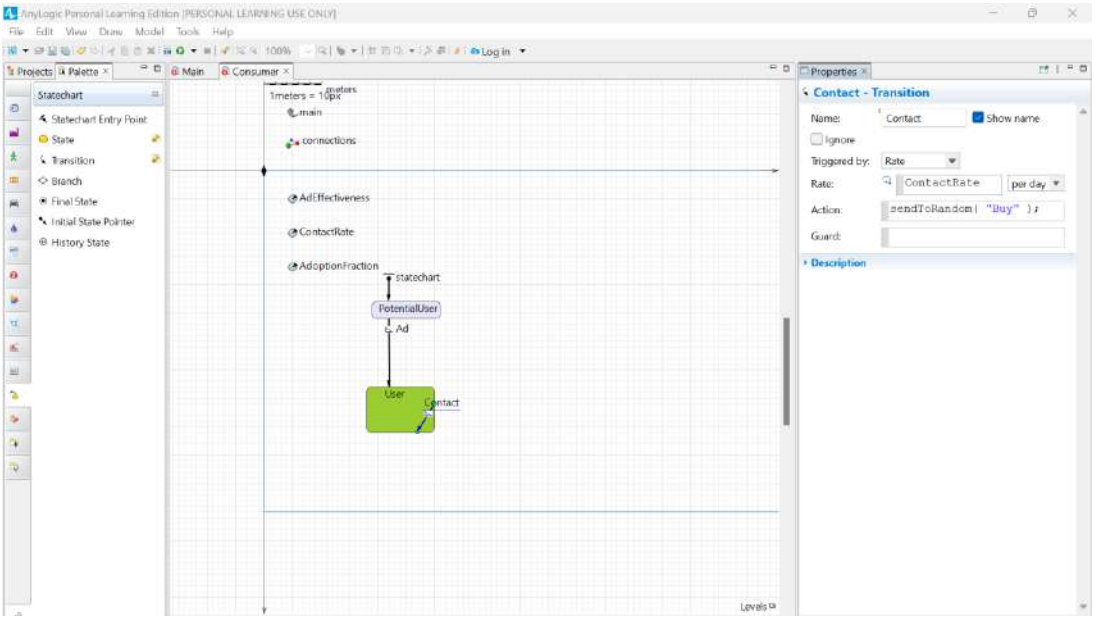
Open the Consumer diagram and increase the User state to fit the internal transition we'll draw
inside the state on the next step.



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:

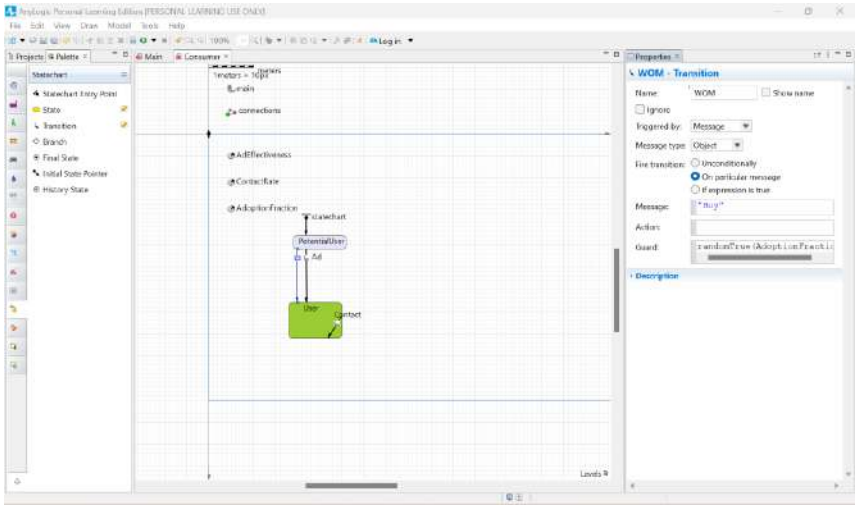
• In the Triggered by list, click Message.

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

• In the Message field, type "Buy"

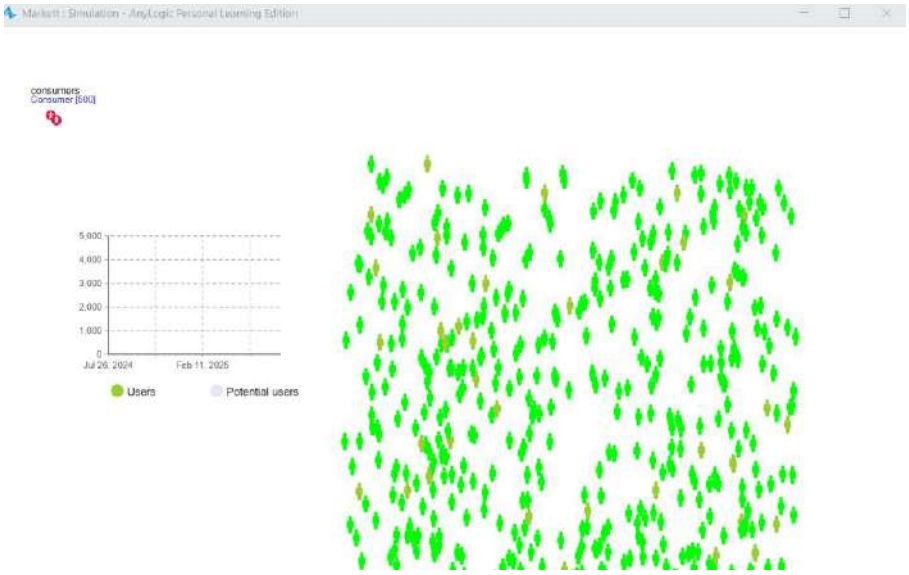
• Since we know not every contact is successful – in other words, a contact may not convince the potential user to buy our product – we'll use AdoptionFraction to make successful contacts

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.

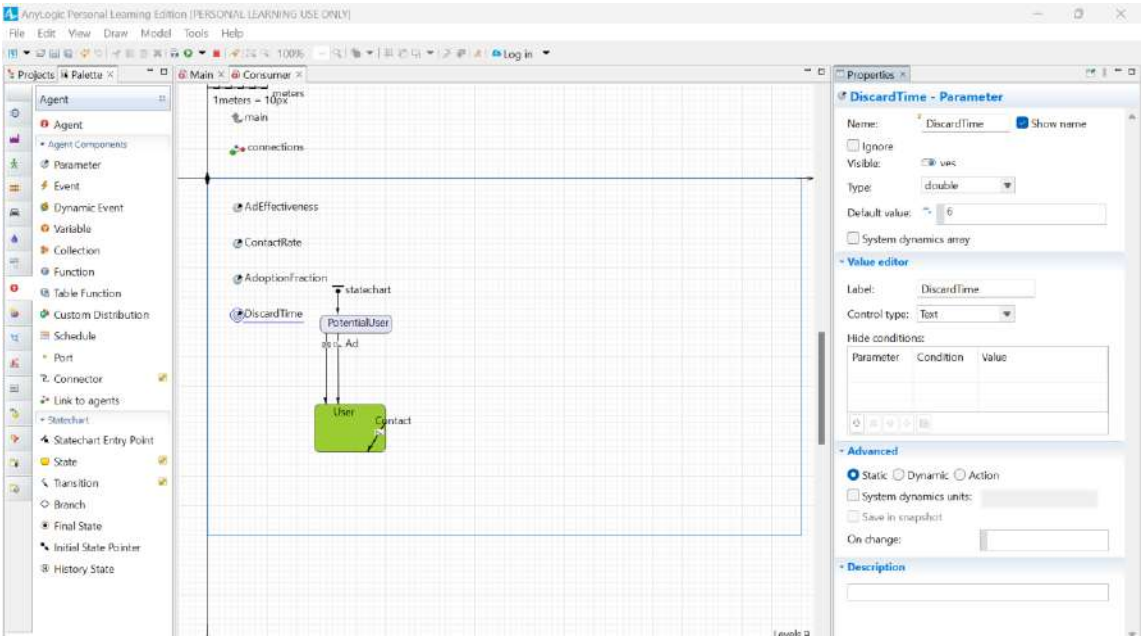


Considering Product Discards

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.



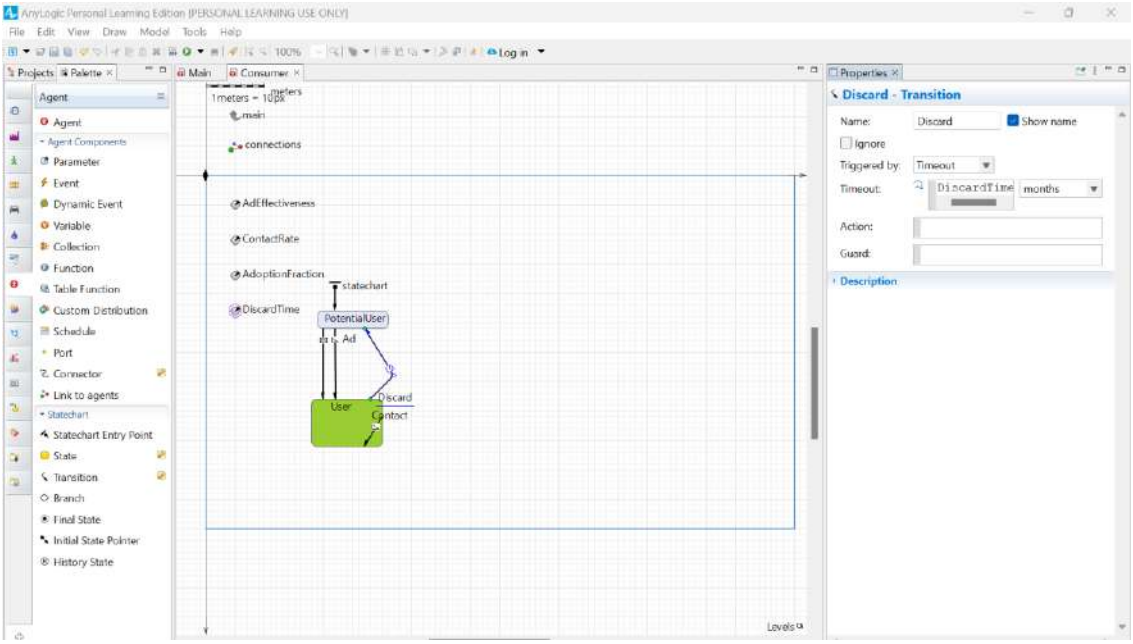
Draw a transition from User to PotentialUser state to model product discards.

To draw a transition with salient points like those shown in the figure, double-click the

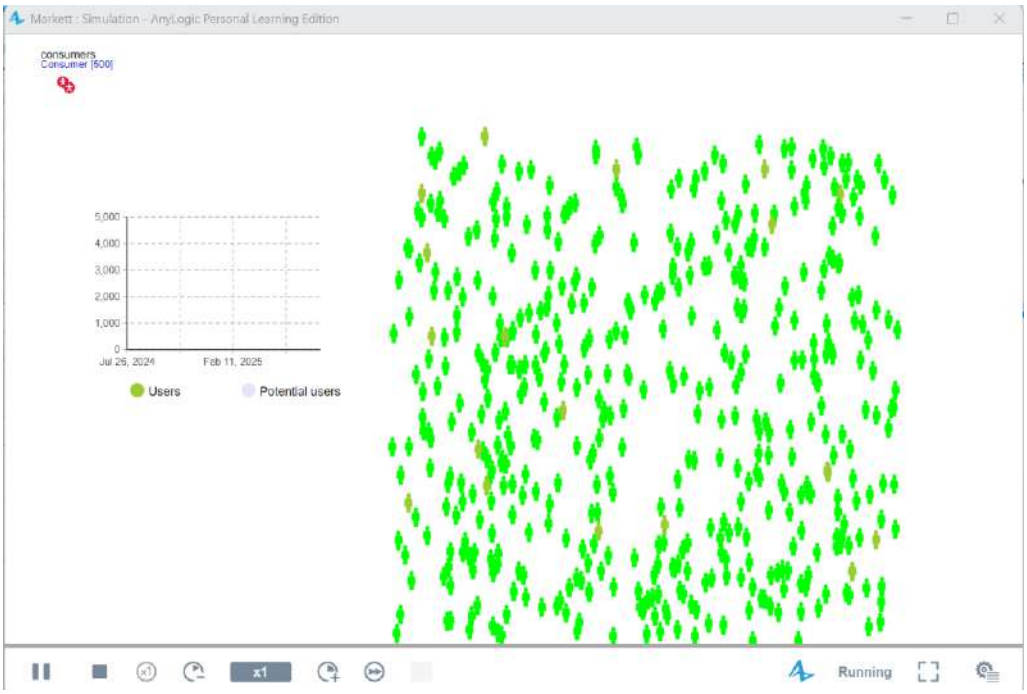
Transition element in the Statechart palette (this should change the element's icon in the palette

to), click the transition's source state User, click at the salient point places, and click the target

state PotentialUser.

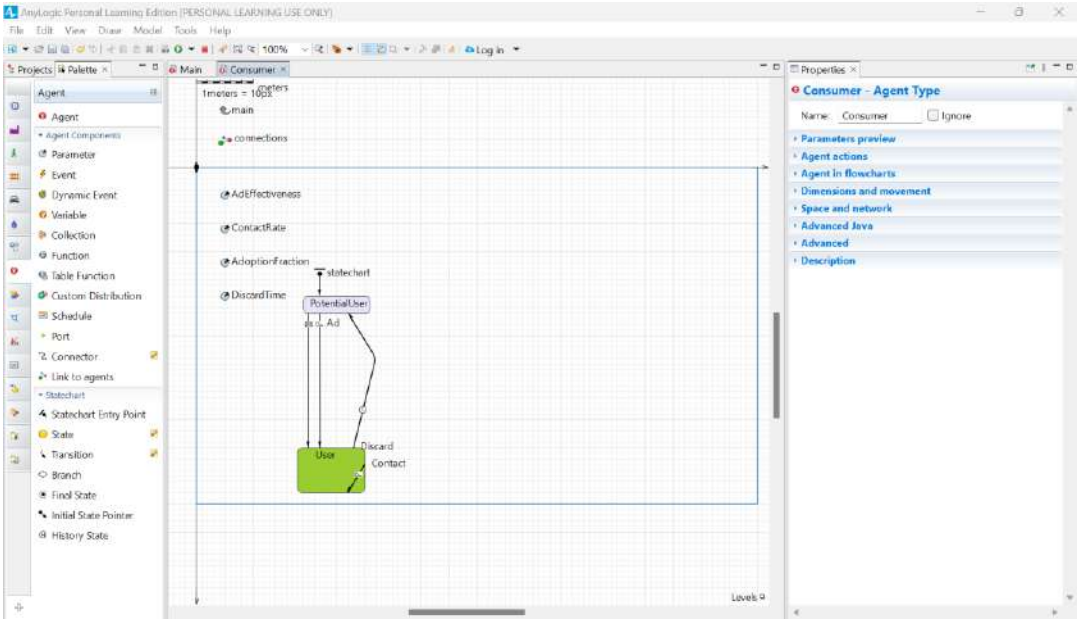


Run the model.

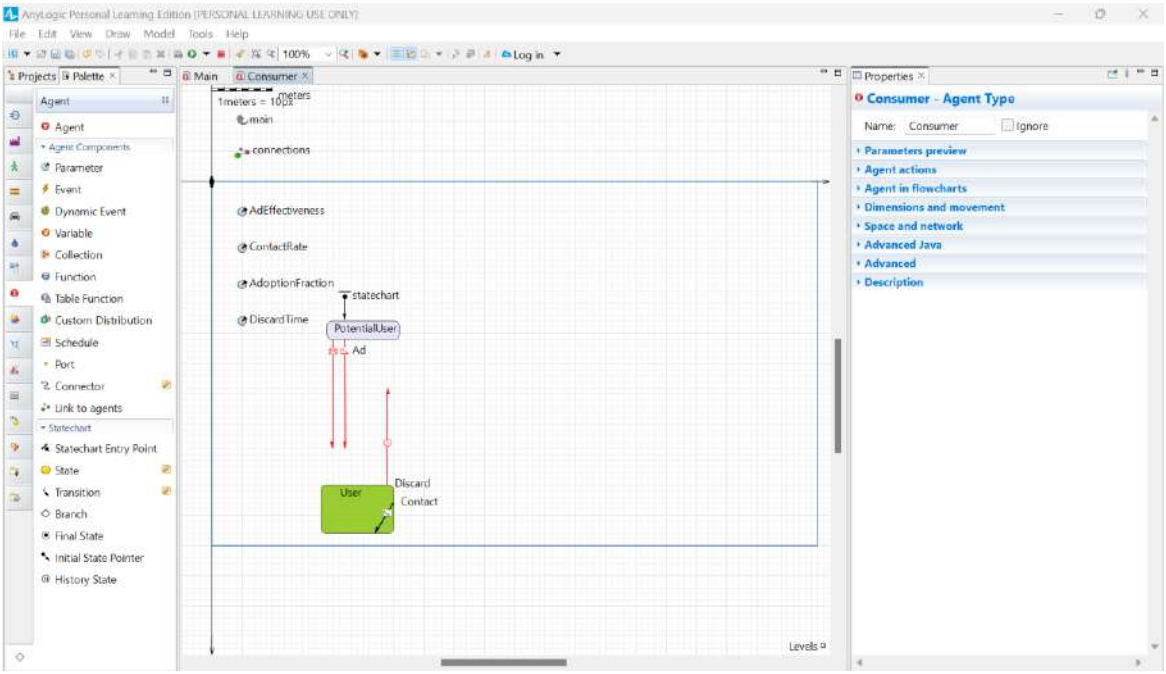


Considering Delivery Time

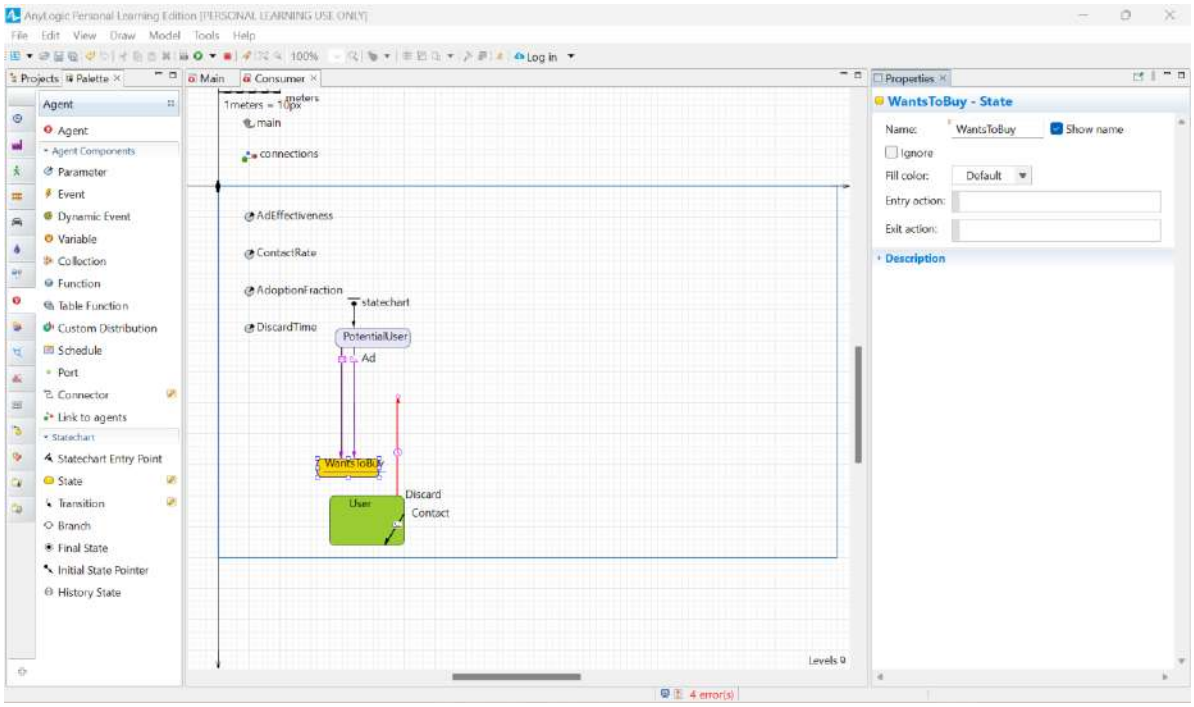
Prepare a place for another state between PotentialUser and User by moving the User state toward the bottom of the screen.



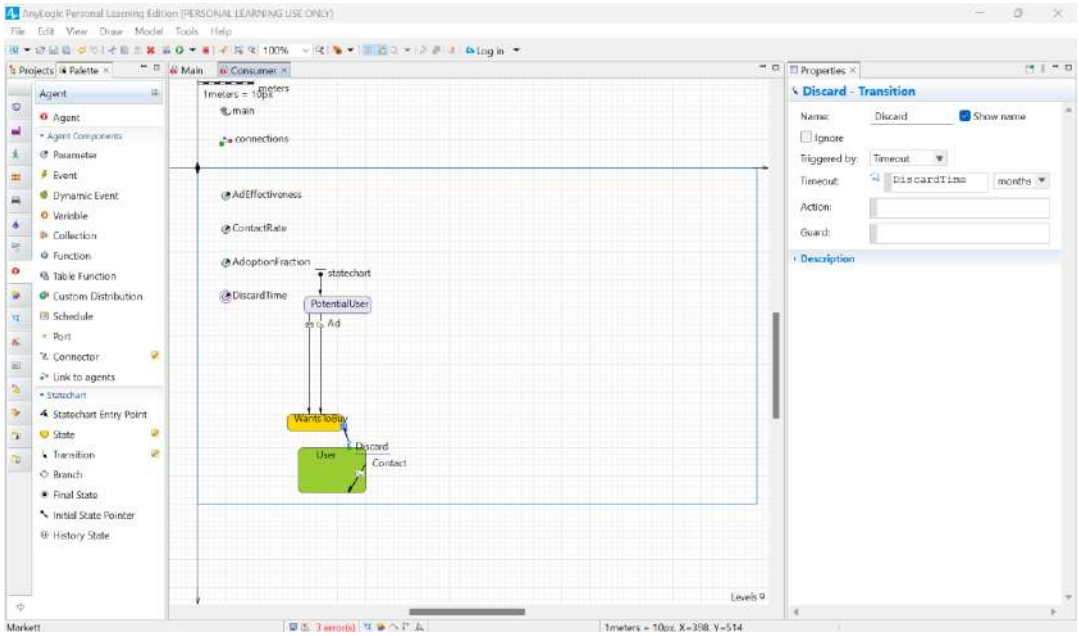
Disconnect the User state from the transitions. Select the WOM and Ad transitions, move their
end points toward the top of the screen, and disconnect the discard transition from PotentialUser. Afterward, you'll notice the disconnected transitions are drawn in red.



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.

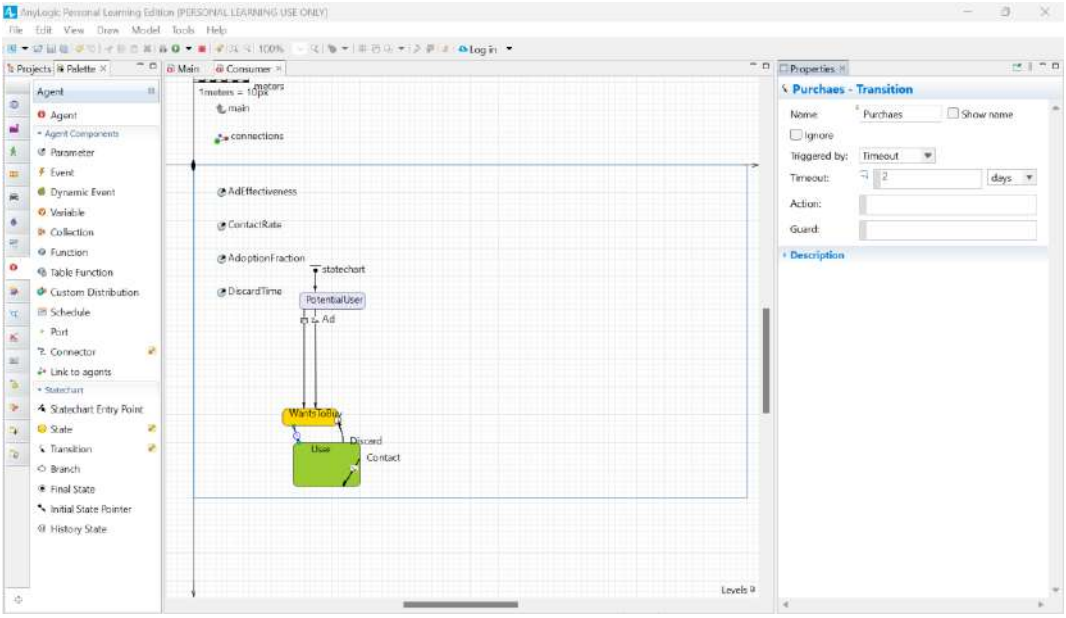


Reconnect transitions to the middle state: the WOM, Ad, and Discard transitions should now end in the WantsToBuy state.

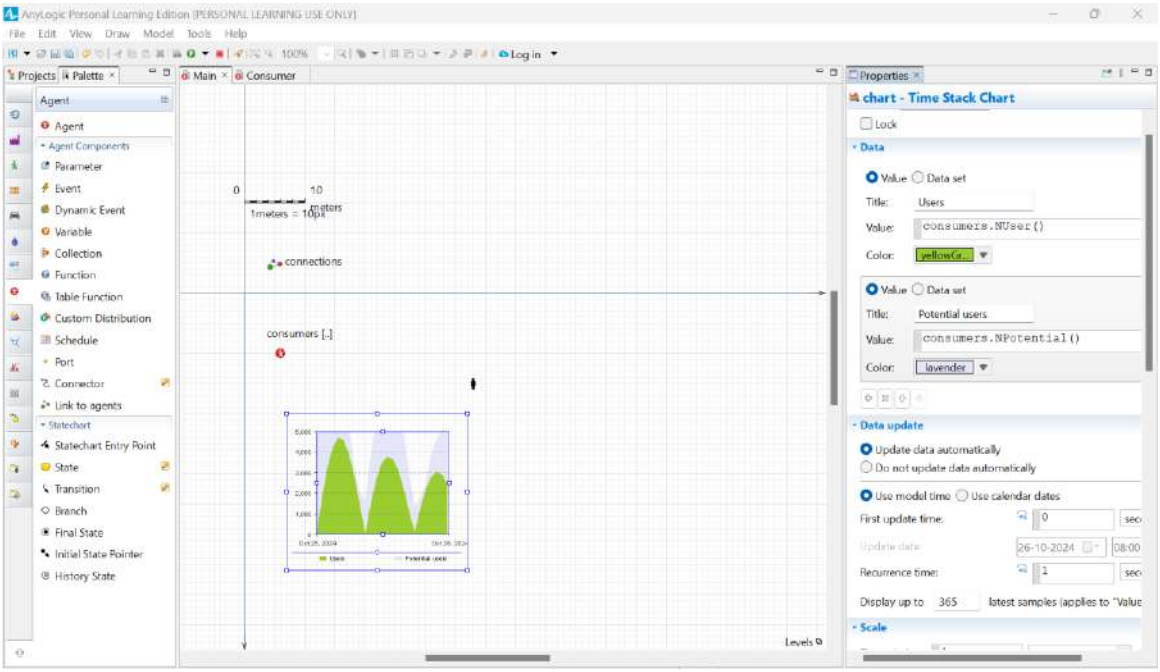


Modify WantsToBuy similar to other states:
Fill color: gold
Entry action: shapeBody.setFillColor(gold);
Add a transition from WantsToBuy to User state to model the product shipment and name it Purchase.
Let's assume it typically takes a user two days to get the product. This means once the consumer's statechart enters the state WantsToBuy, it will proceed to the state User with a two-

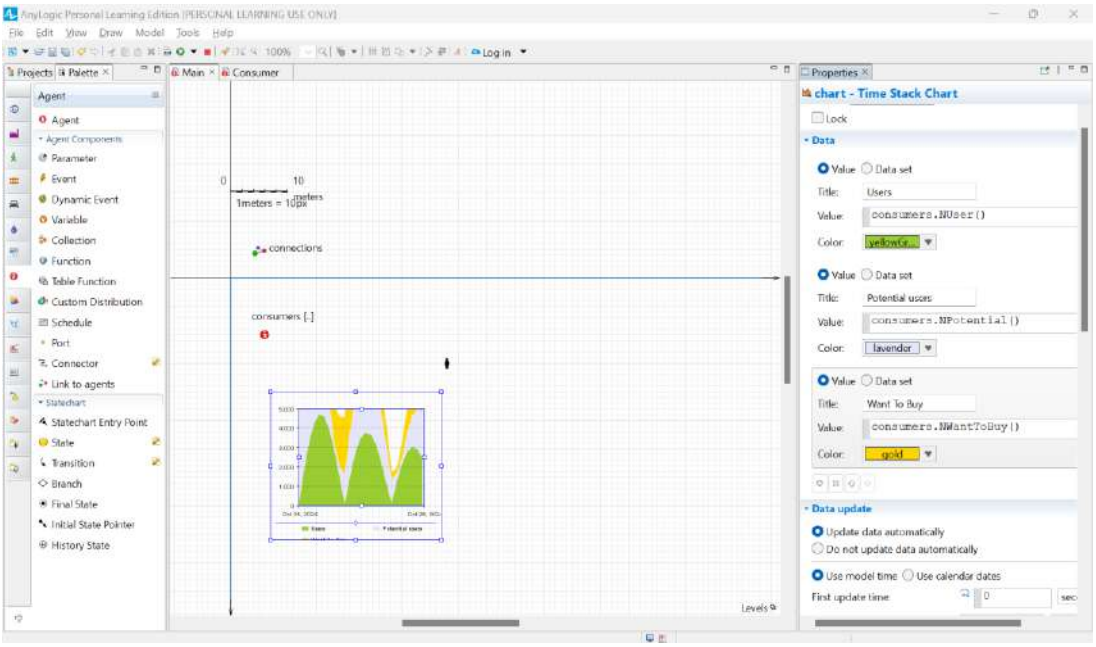
day delay. With this in mind, set 2 days timeout for the Purchase transition:



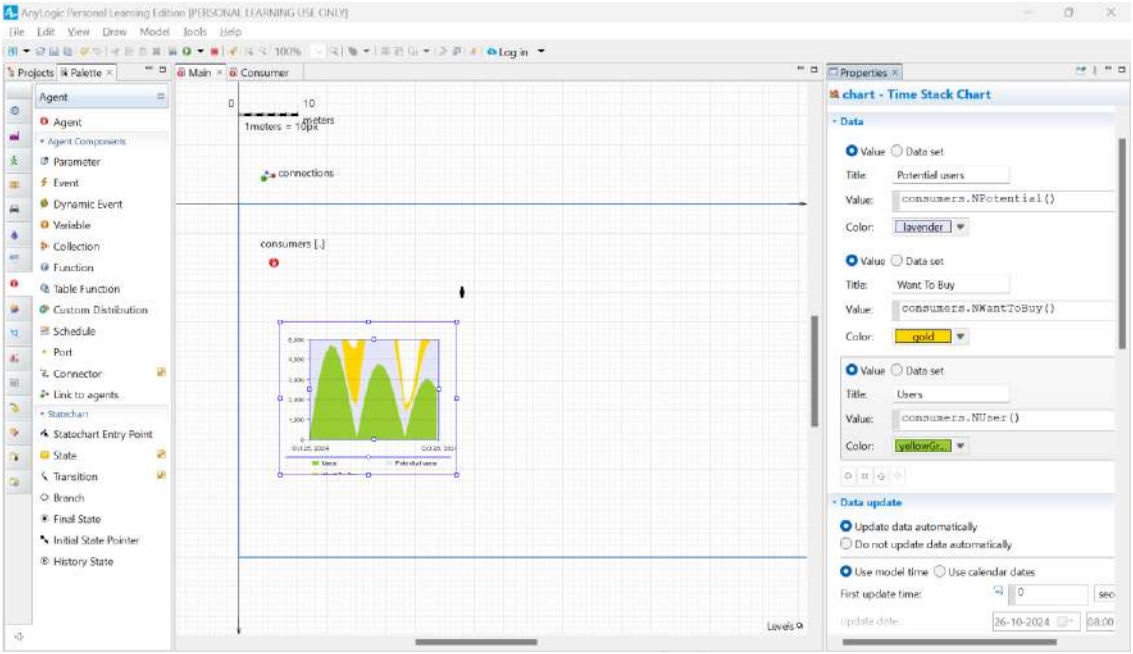
Define one more statistics function to count the product's market-driven demand.
In the editor of Main, click the consumers, go to the Statistics properties section, and add a statistics item: NWantToBuy with condition item.inState(Consumer.WantsToBuy)



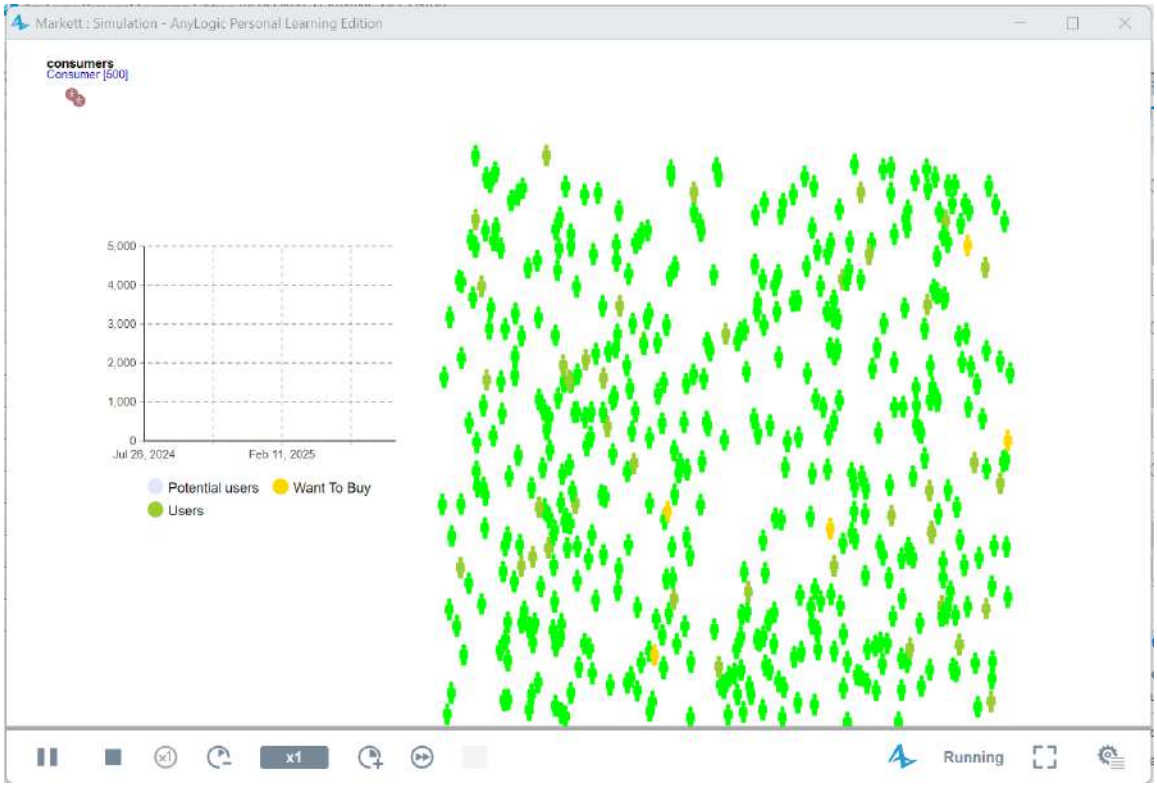
On Main, select the time stack chart, and add another data item to be displayed with the chart: consumers.NWantToBuy() in value with the title Want to buy and color gold.

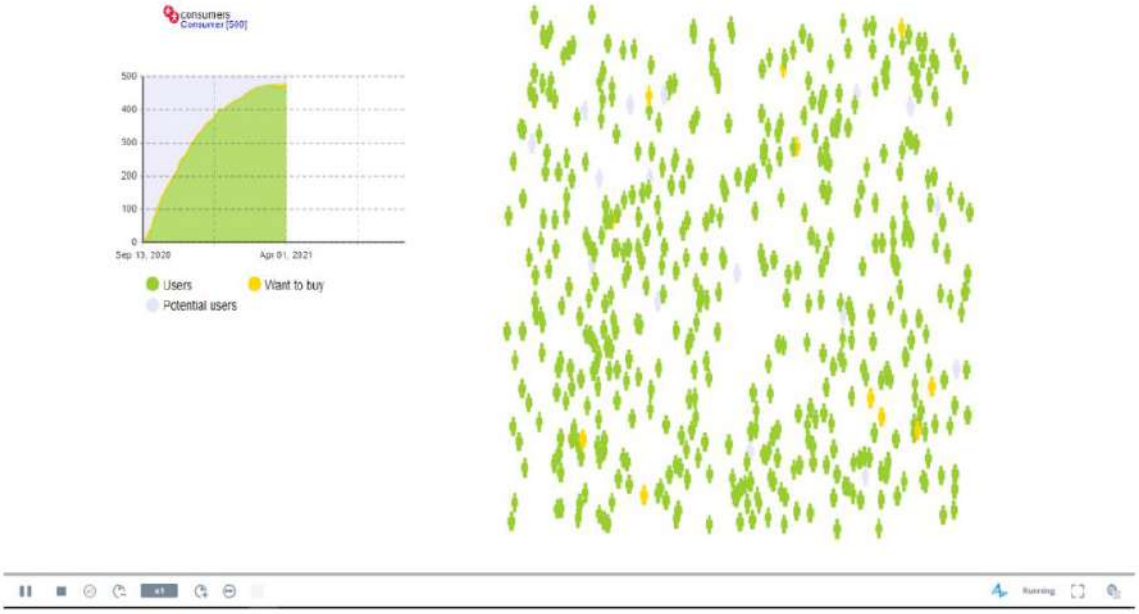


Make the newly-defined data item second in the list by selecting the item's section and clicking
the "up" button.



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


Markett : Simulation - AnyLogic Personal Learning Edition
consumers
Consumer [500]
5,000
4,000
3,000
2,000
1,000
0
Jul 26, 2024
Feb 11, 2025
Potential users
Want To Buy
Users
x1
Running


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

Practical No: - 3

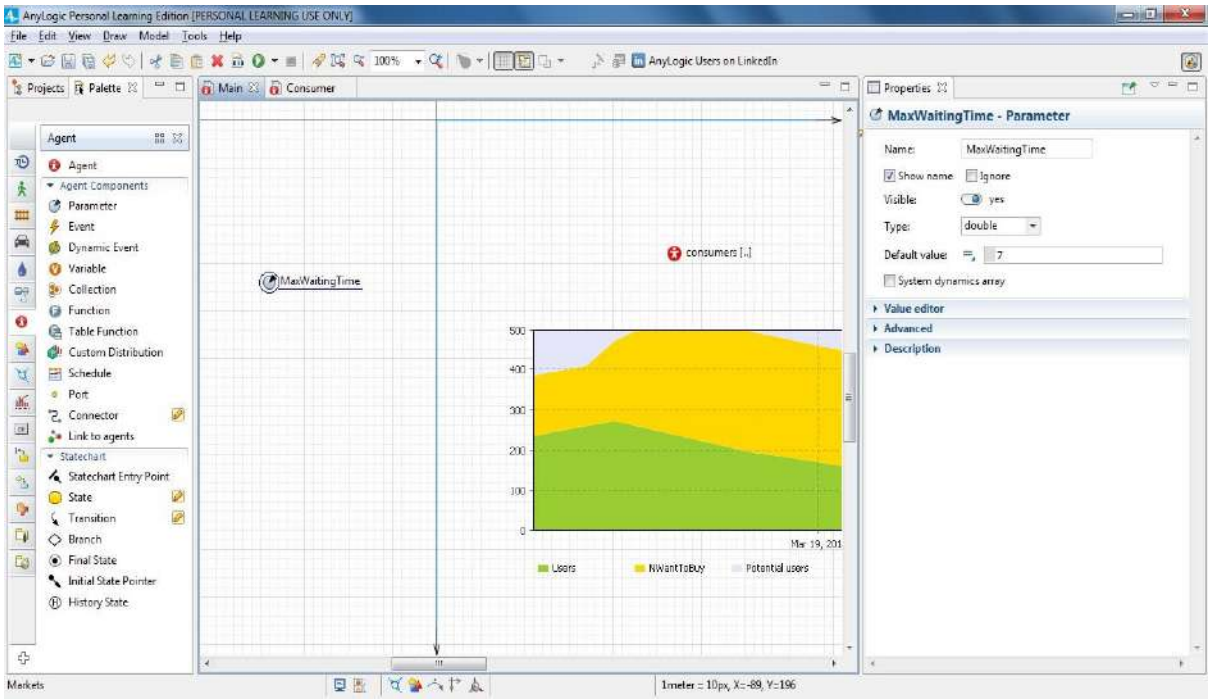
Aim: - Design and develop agent based model by

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

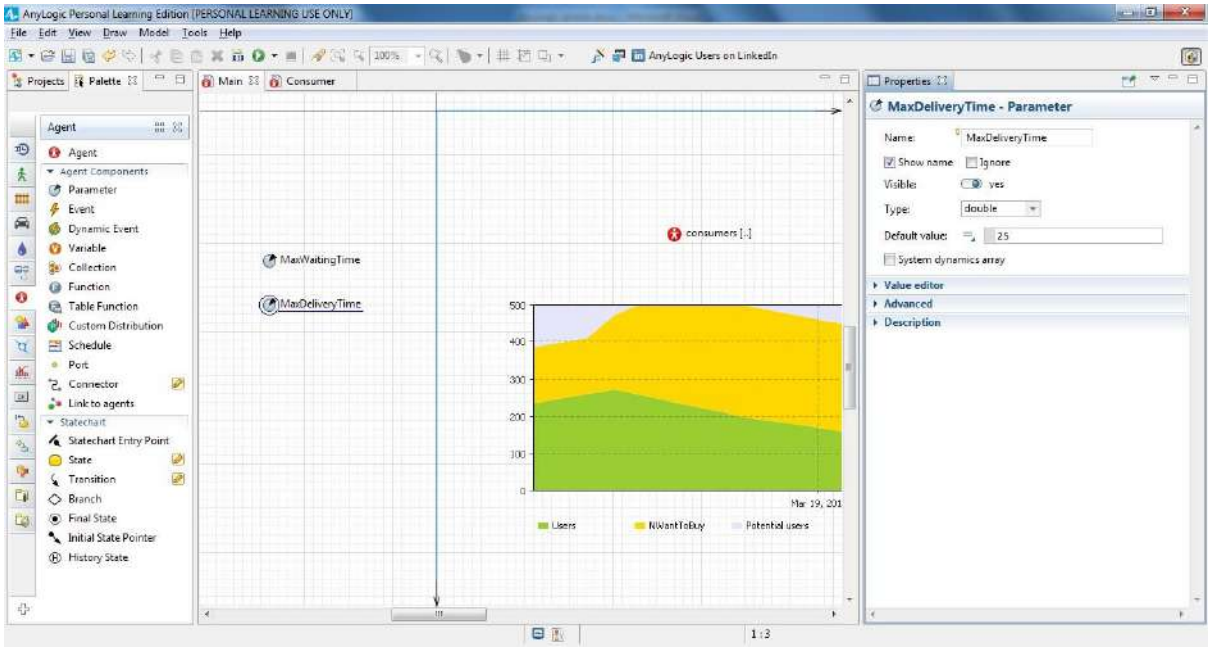
[Use a scenario like market model]

Code:-

Configure the parameters. MaxWaitingTime defines the maximum time a consumer will wait for the product for 7 days



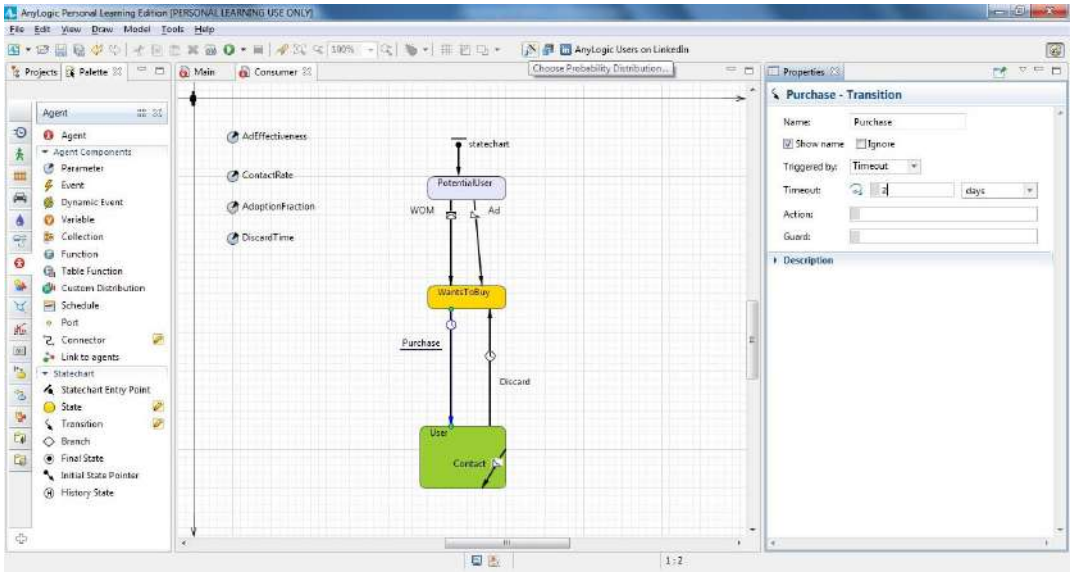
Set the other parameter, MaxDeliveryTime to 25 days to reflect our assumption it may take up to 25 days to deliver a product



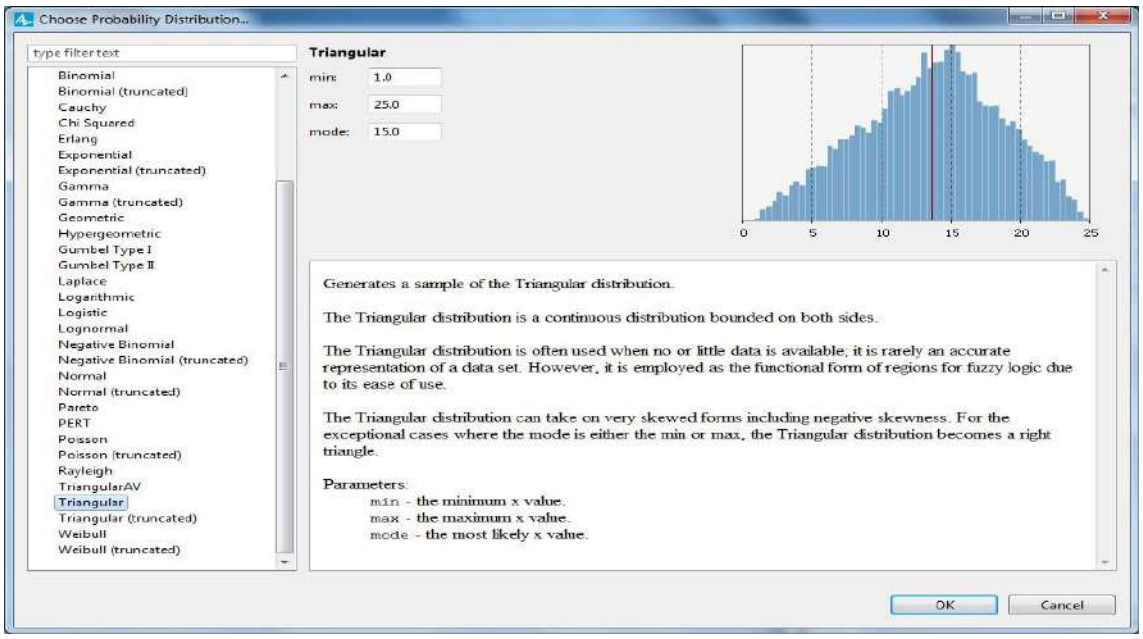
We assume it takes between one and 25 days – with an average of two days – todeliver the product. With that in mind, let's change the delivery time from a fixed two day delivery period to the stochastic expression that describes this pattern.

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…** toolbar button.

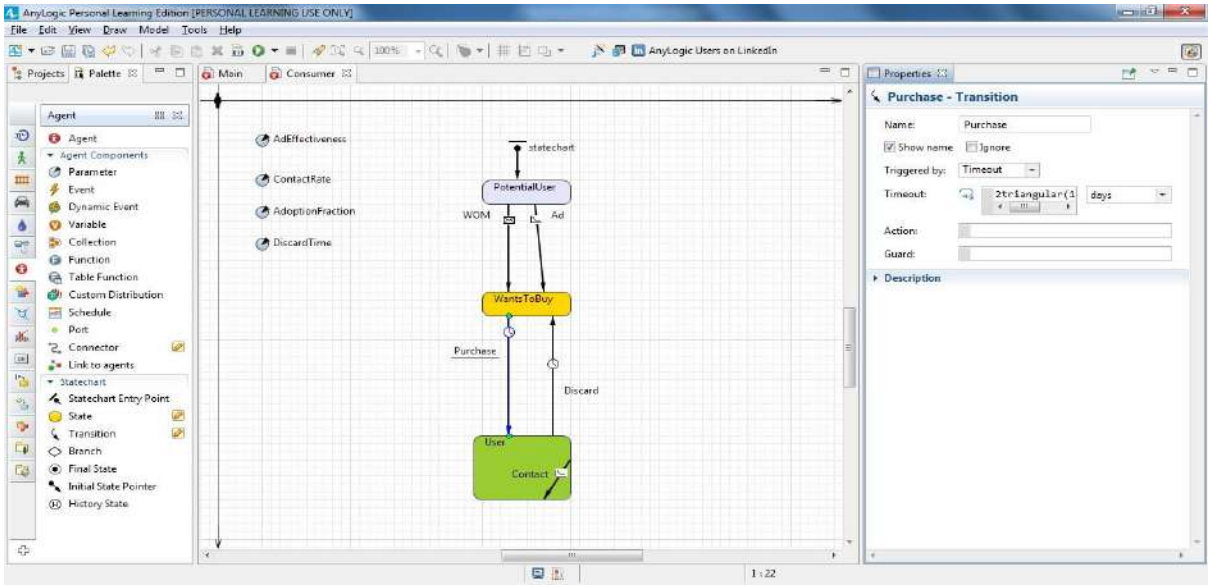


You'll see the **Choose Probability Distribution…** dialog box. **8.** The **Choose Probability Description** screen allows you to view the list of supported distributions, and you can click any name in the list to view the distribution's description.

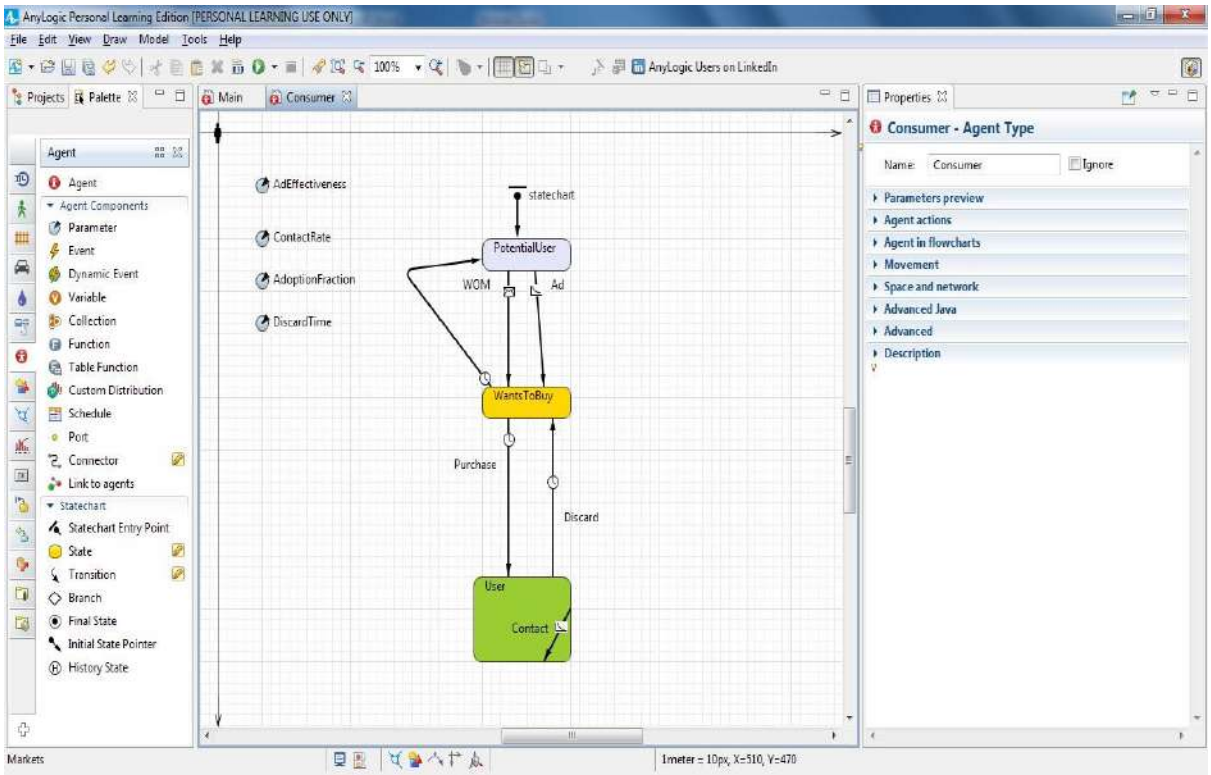


Choose triangular in the list. Set **min**, **max** and **mode** parameters equal to 1, 25, 2 respectively. In the upper right, you'll see PDF instantly built for the distribution with the specified parameters. Click **OK** when finished.

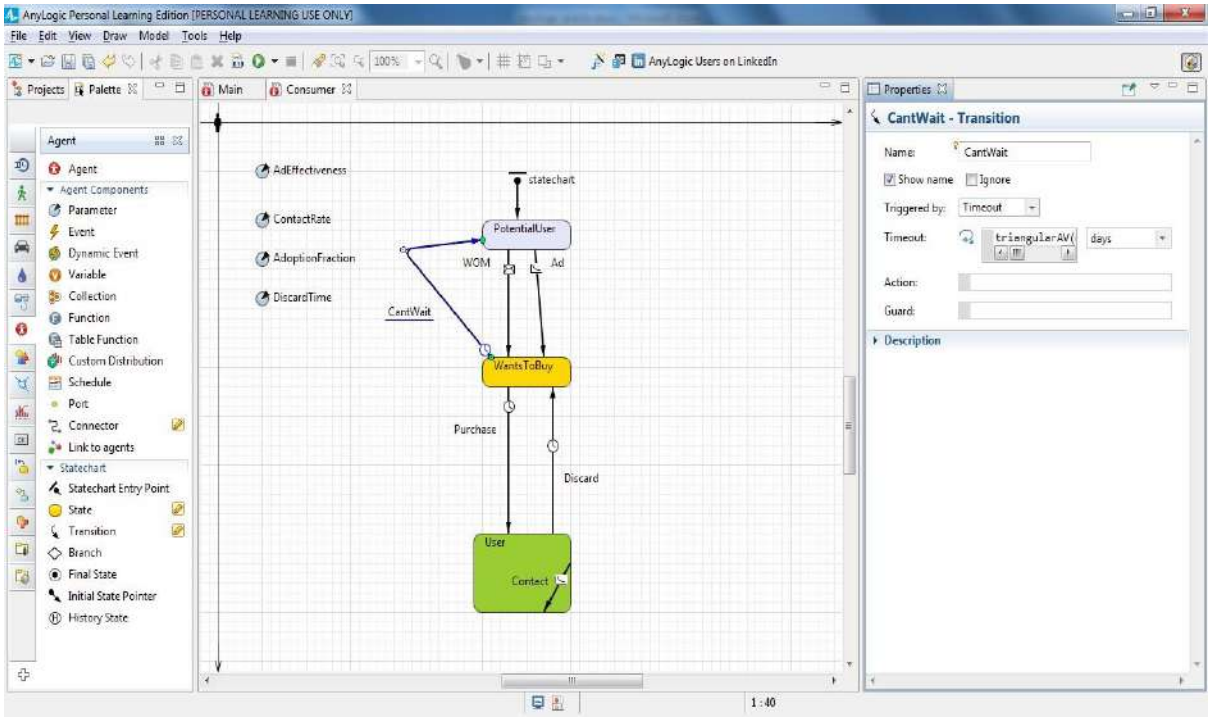
You'll see the expression triangular(1, 25, 2) automatically inserted as the timeout value. Let's modify the line to triangular(1, main.MaxDeliveryTime, 2) Here main is how we access the Main agent from the consumer agent.



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:

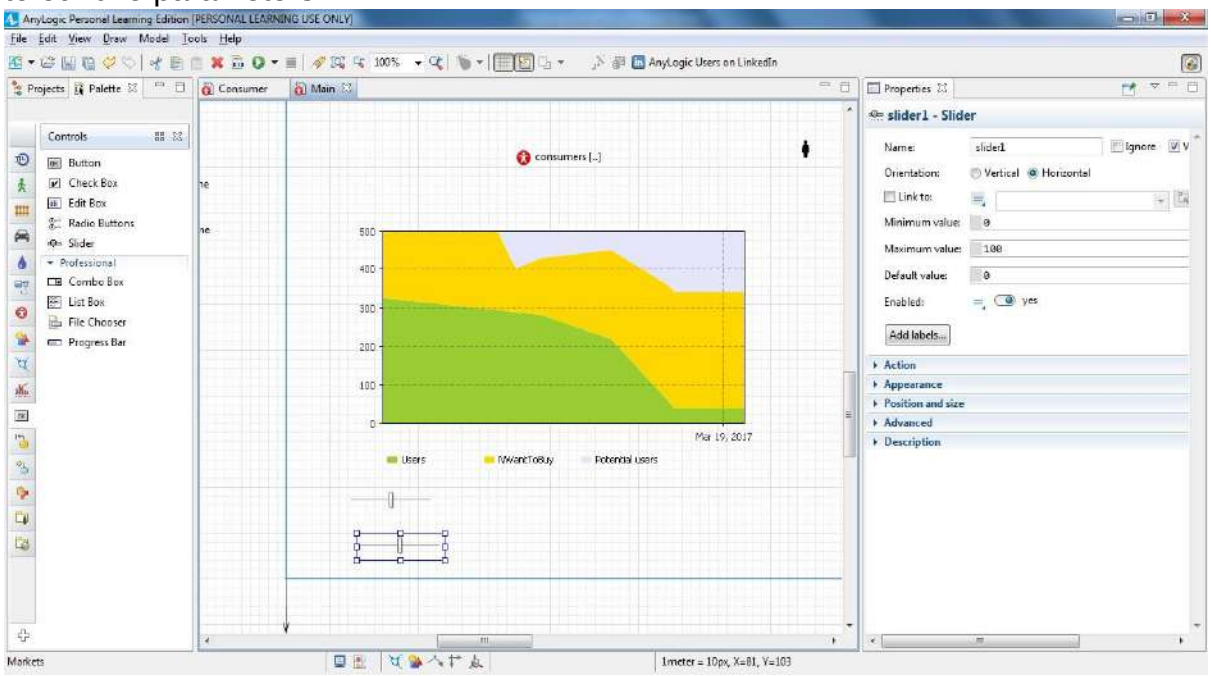


Modify the transition properties so it is triggered by **Timeout** which equals triangularAV(main.MaxWaitingTime, 0.15) **days**

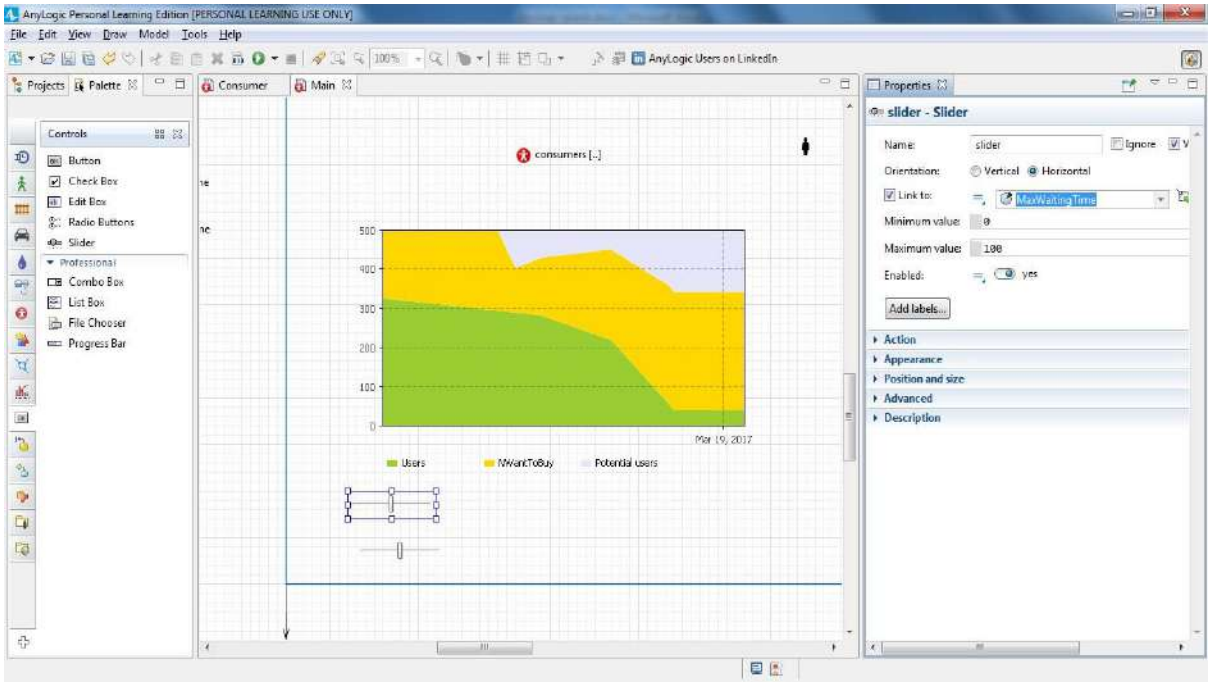


Rather than setting the maximum waiting time equal to constant MaxWaitingTime, we assume it follows a triangular distribution with an average of one week and a possible variation to up to 15 percent.

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.

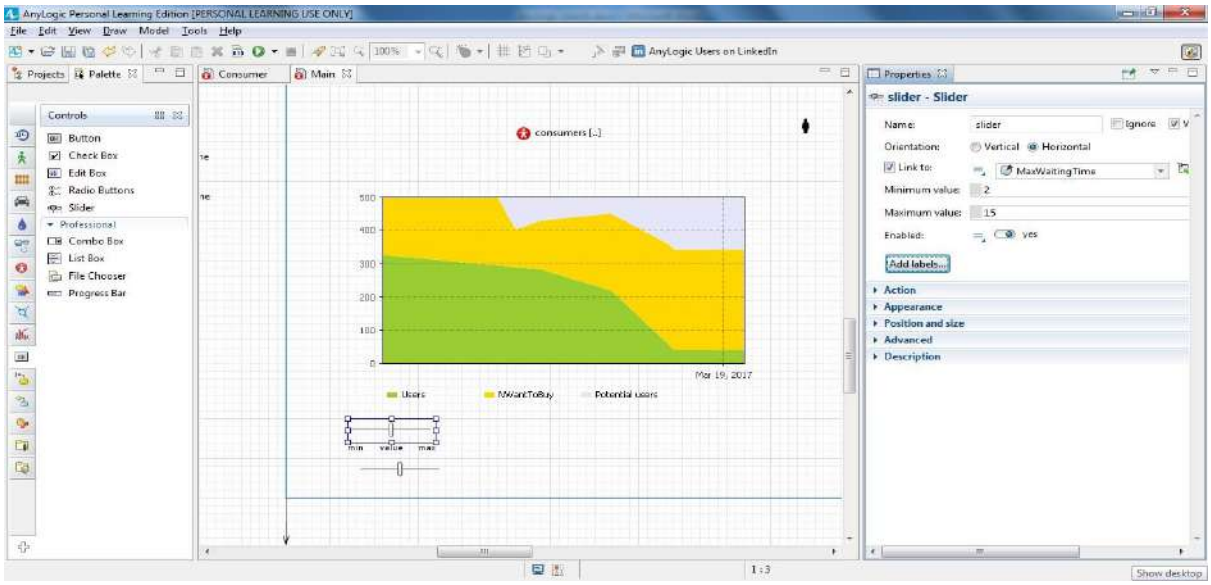


Select the checkbox **Link to** and select the parameter MaxWaitingTime to the right.

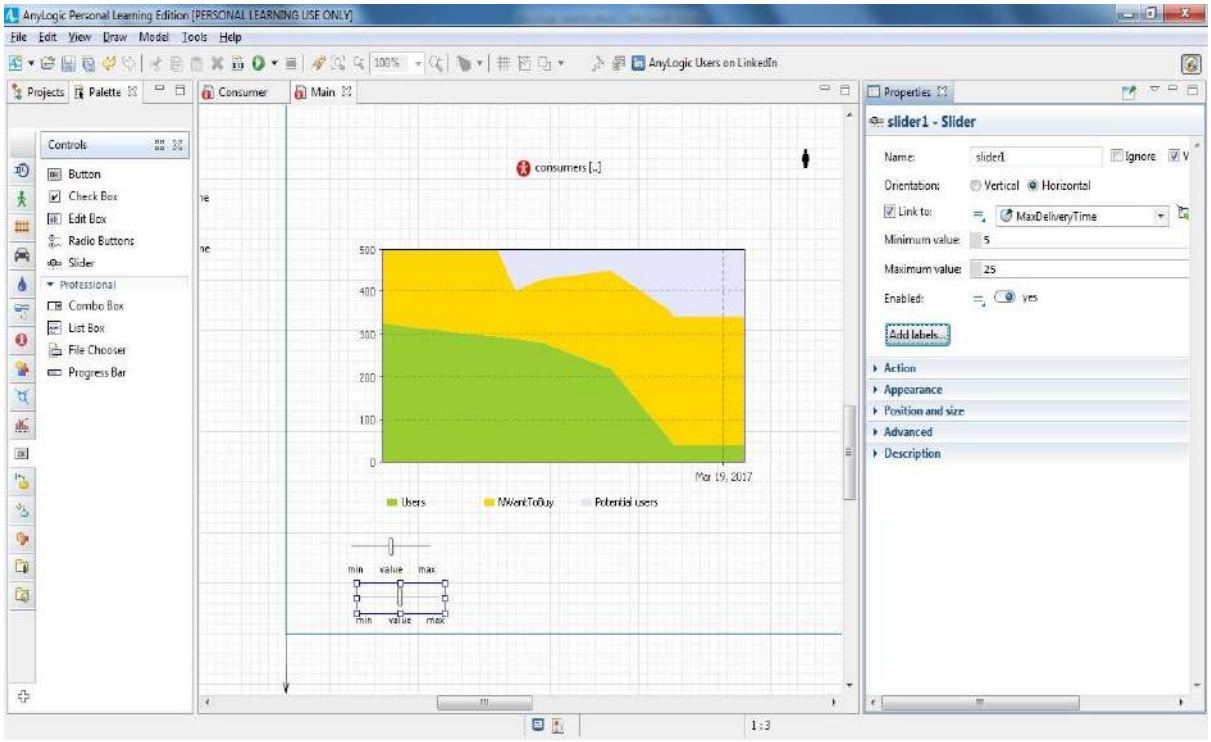


Set the slider's **Minimum** and **Maximum values**. The parameter value can vary within the range you define here, and we'll set 2 as **Minimum** and 15 as **Maximum value**.

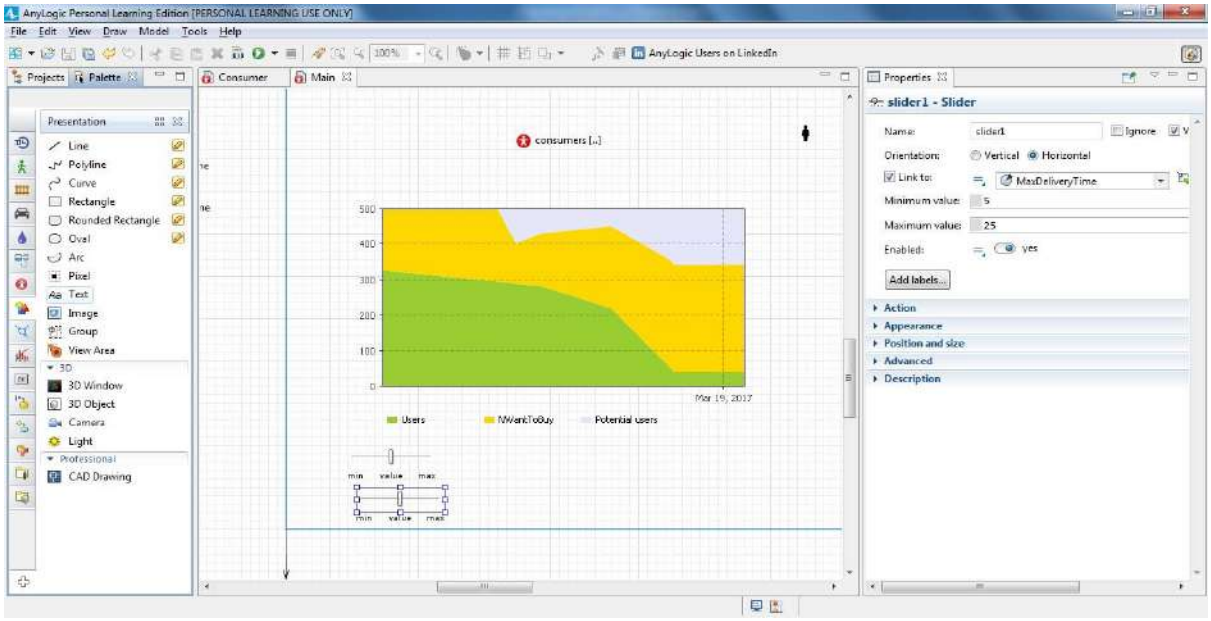
Finally, click the **Add labels…** button to display the slider's minimum, maximum, and current values at runtime (the min, value, and max text shapes will appear beneath the slider).



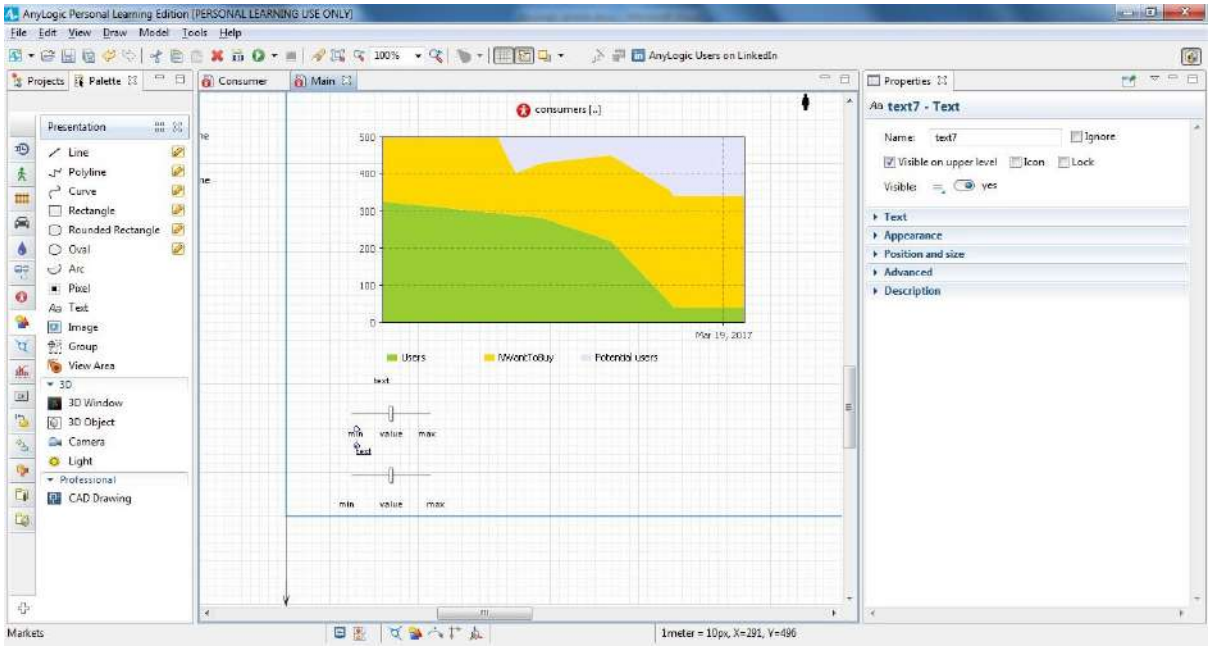
Select the checkbox **Link to** and select the parameter MaxDeliveryTime to the right.

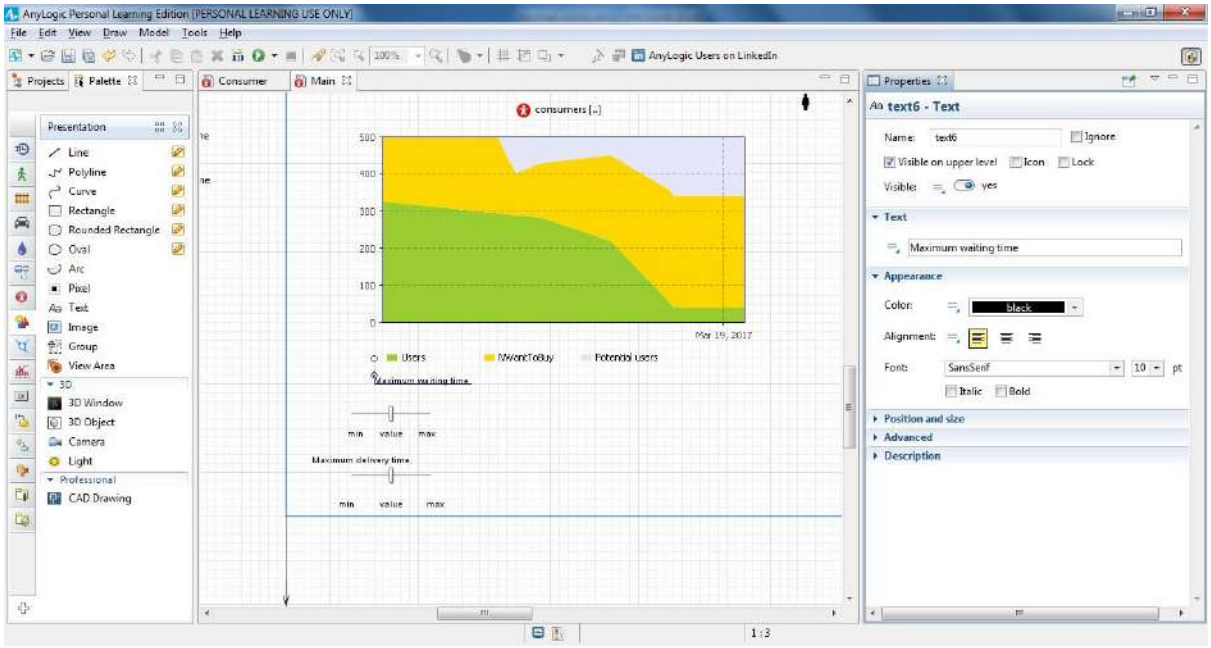


Open the **Presentation** palette, drag two **Text** shapes on to the diagram, and place them above the sliders. Let's configure the titles of these controls.

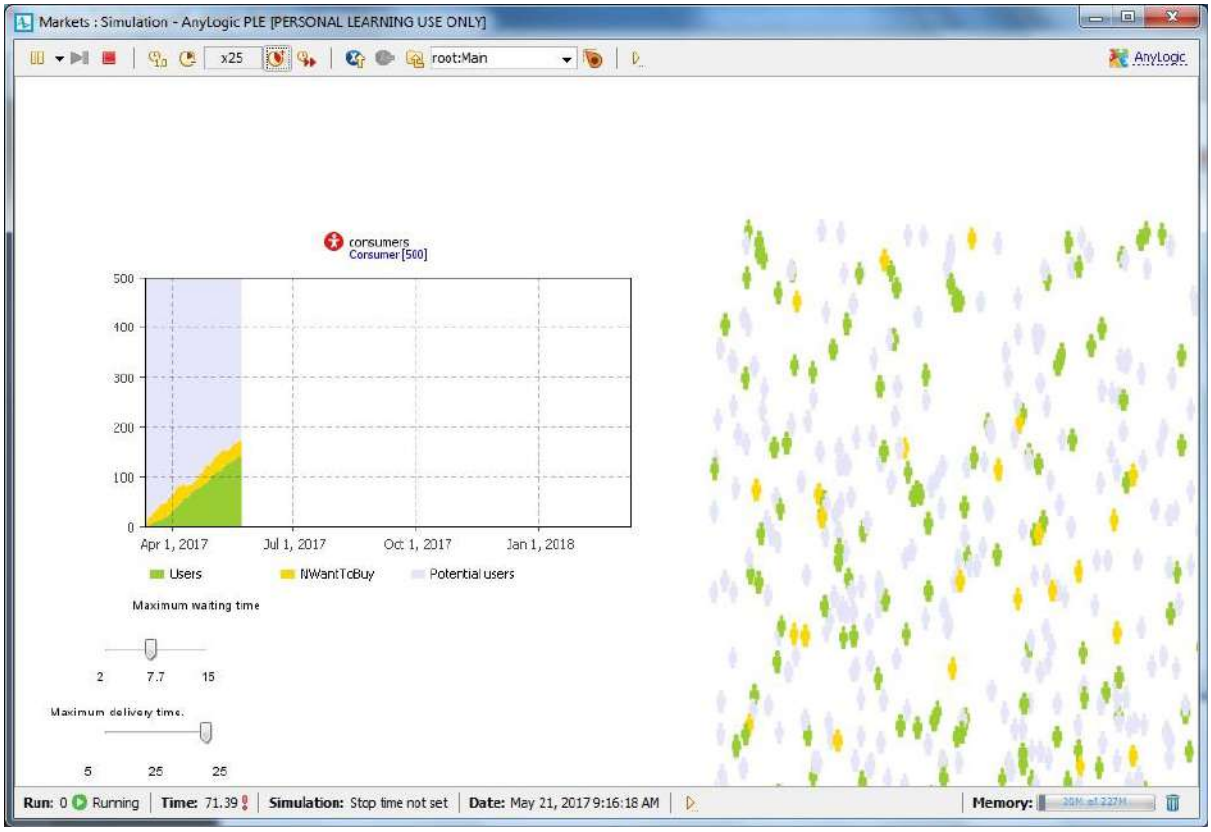


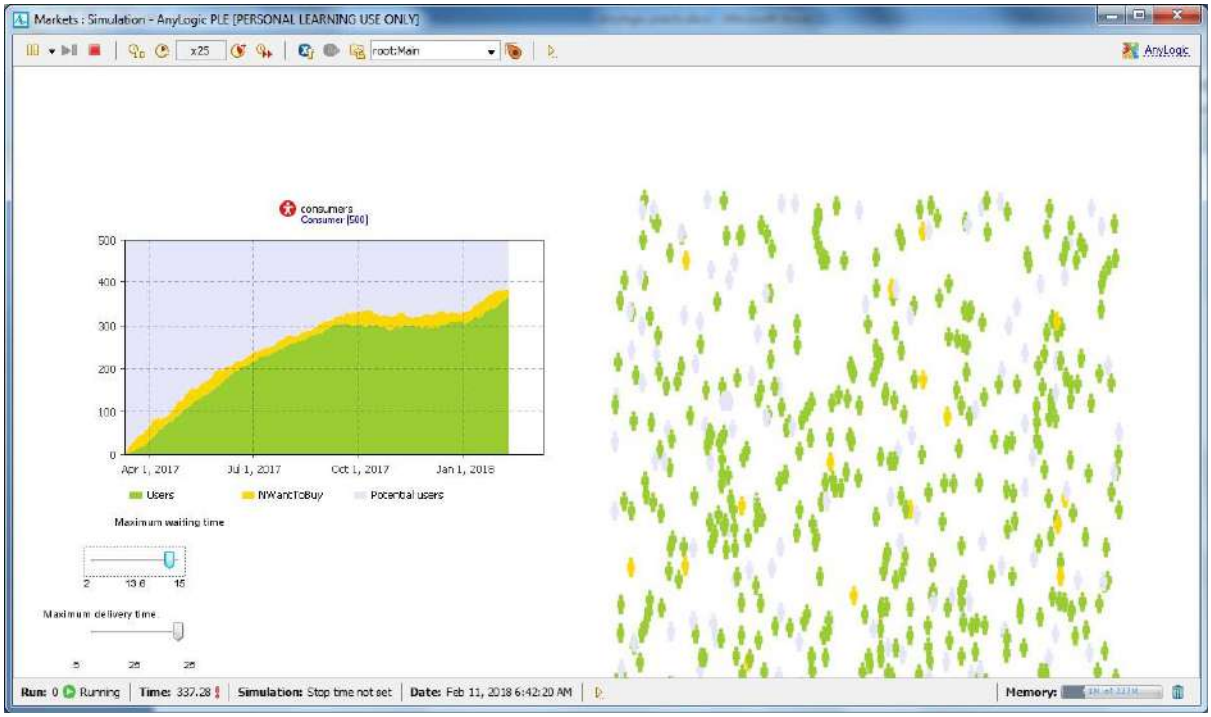
In the properties view, in the **Text** section, enter the text that the model will display. Using text shapes, name one slider Maximum waiting time and the other slider Maximum delivery time.



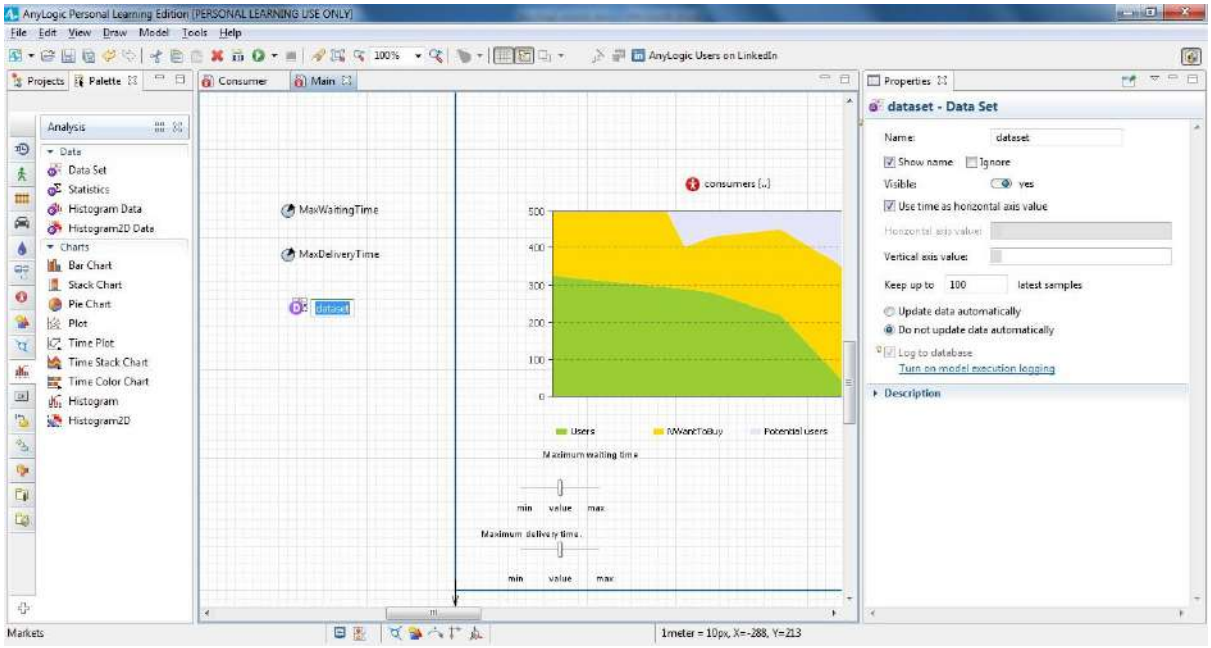


Run the model and observe the behavior. As you use the sliders to change the maximum waiting time or delivery time, you'll see your changes reflected in consumer behaviors and whole adoption dynamics.

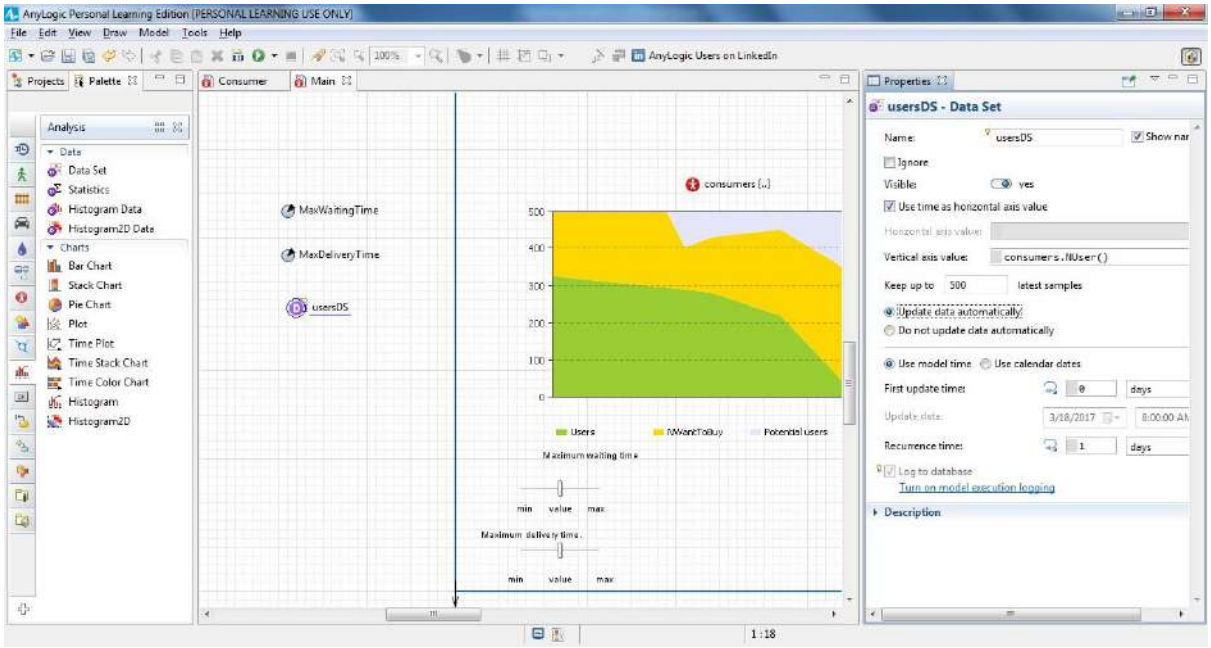




Comparing model runs with different parameter values Open the Main diagram and add a **Data Set** from the **Analysis** palette.Name it usersDS

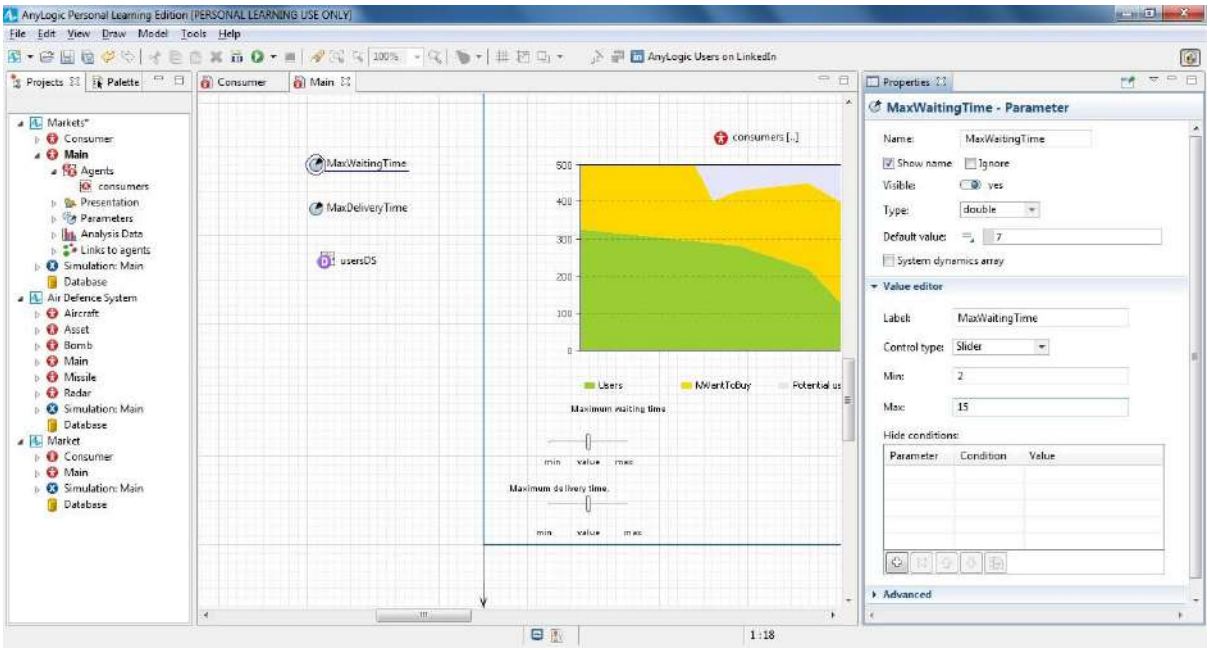


Data Set is capable of storing 2D (X,Y) data of type double. We want this dataset to store the history of product sales dynamics. We'll store data samples, each with a timestamp and the current number of the product users. Set the value that the dataset will store. In the **Vertical axis value** property, 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

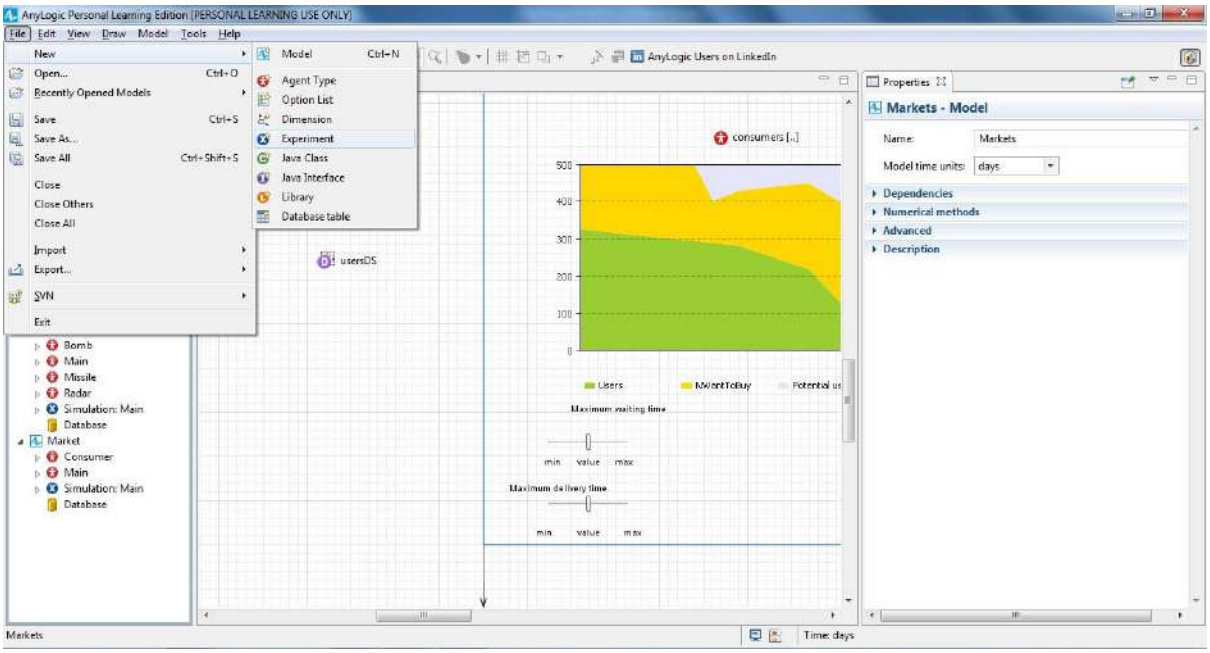


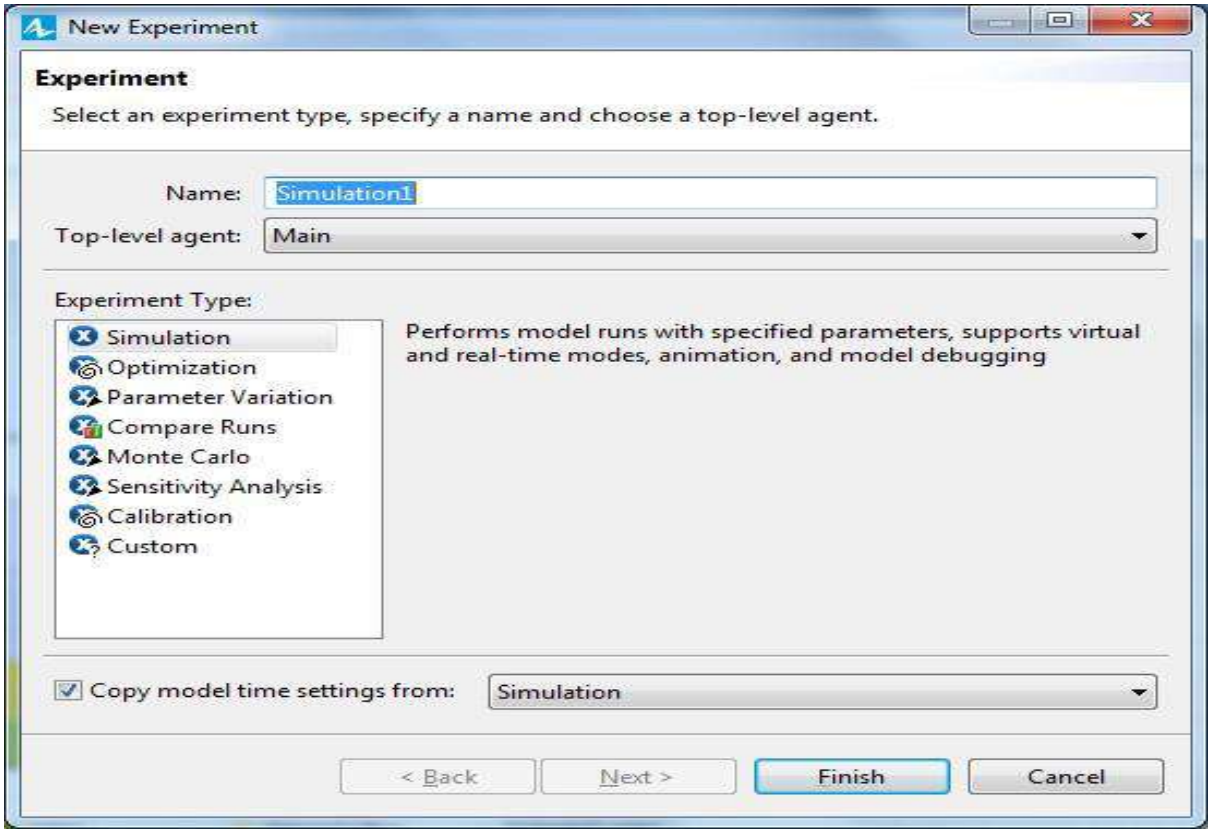
Next, make changes in the **Value editor** section for both parameters on Main diagram (MaxWaitingTime and MaxDeliveryTime). Choose Slider as **Control type**, set **Min** and **Max** values the same as we have in the sliders

on Main, and if you want, change the default label (say, Maximum waiting time).

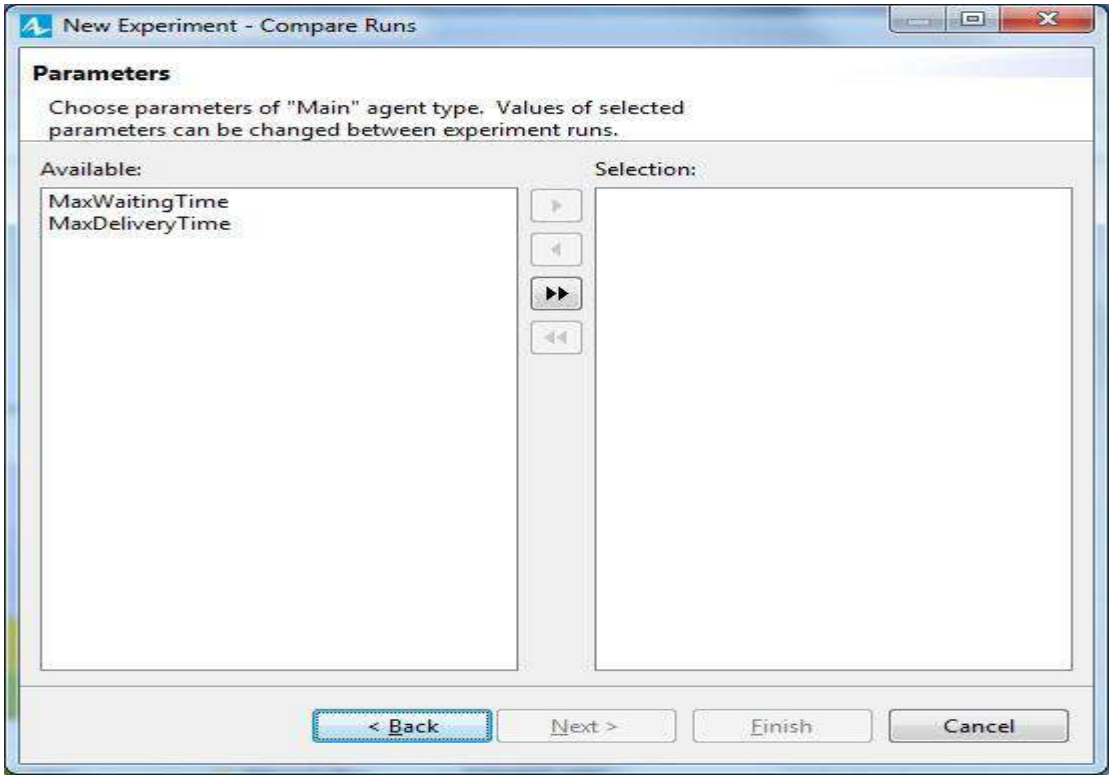


Open the **Projects** view, right-click the model item, and select **New** > **Experiment** from the context menu. The **New experiment** wizard will pop up.



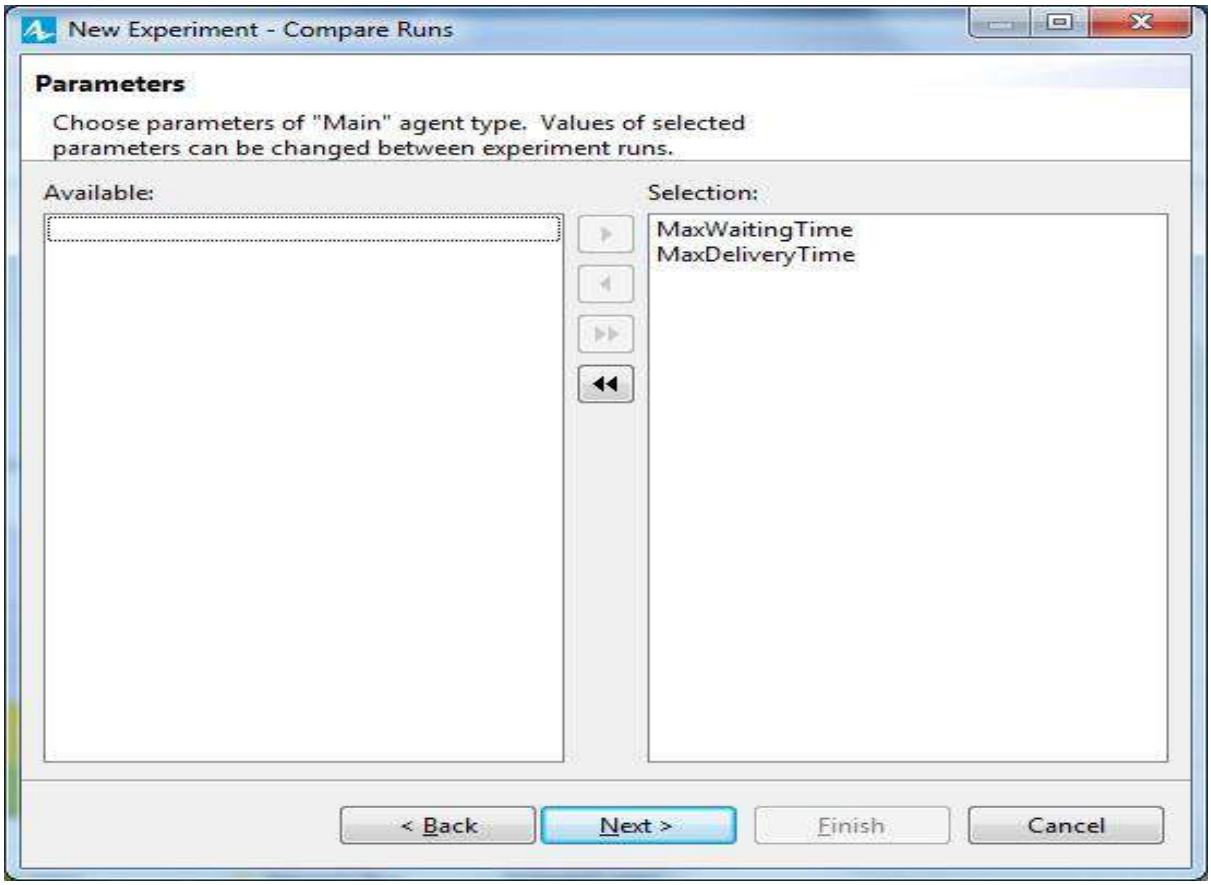


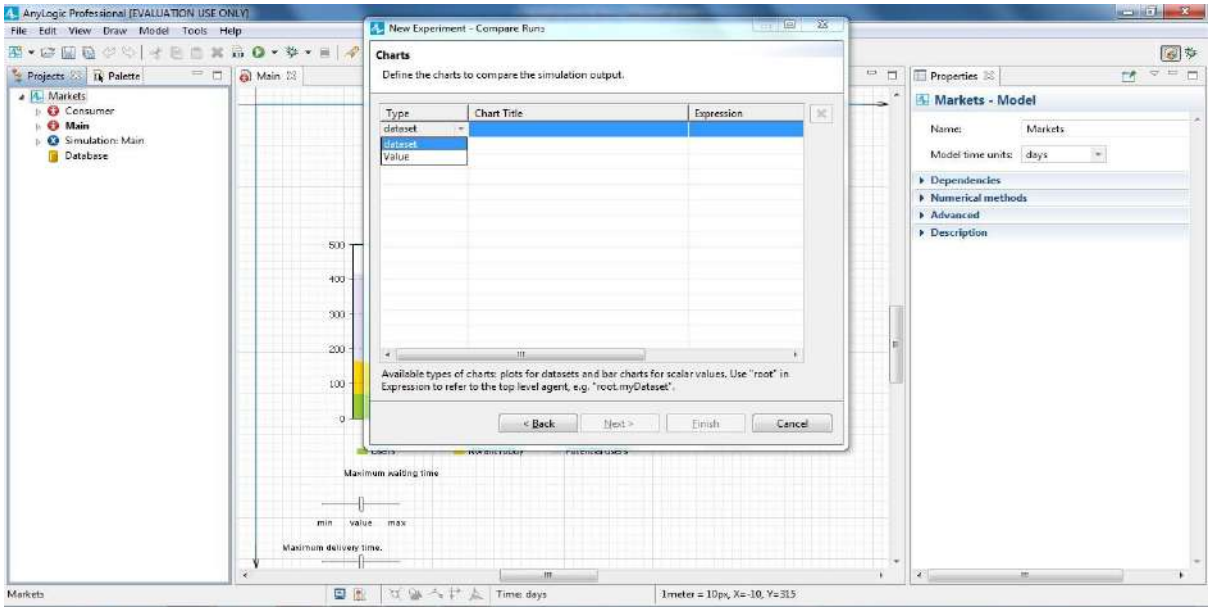
Select **Compare Runs** experiment from the list of experiment types andclick **Next**.



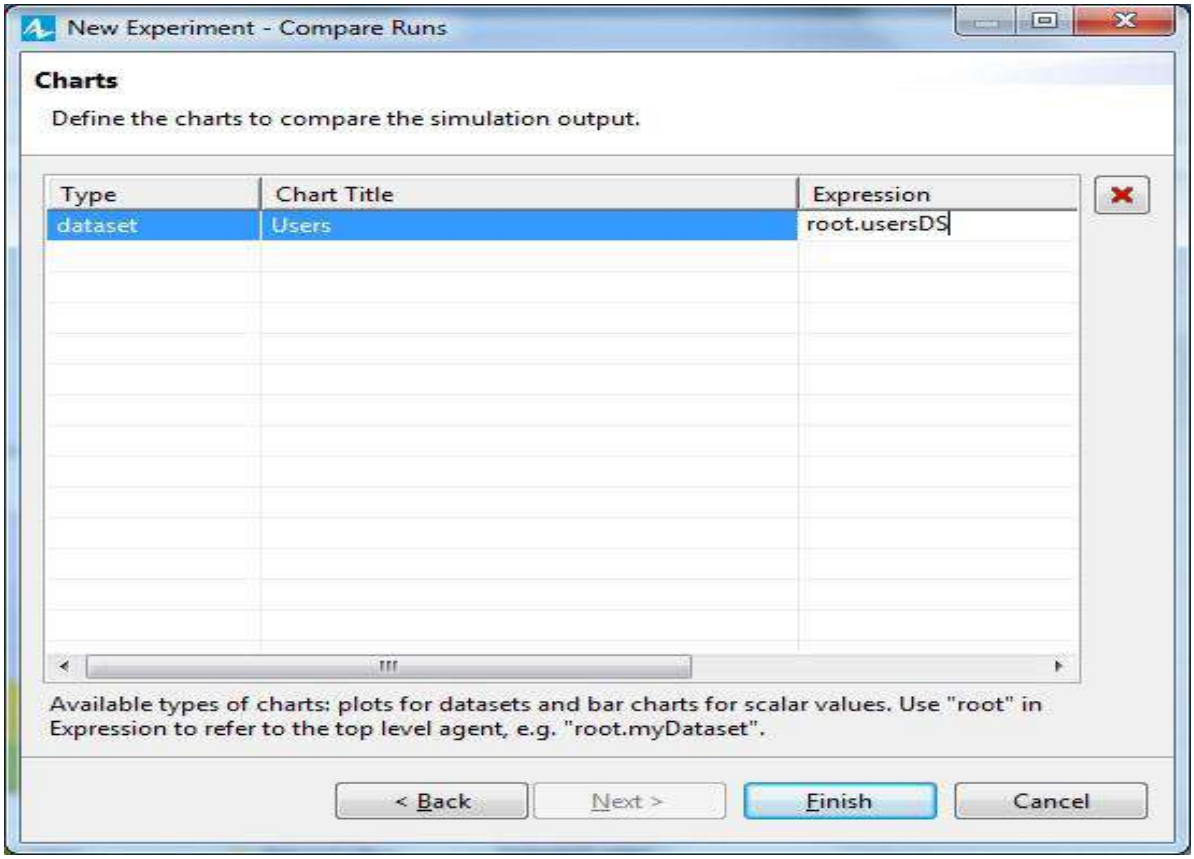
On the **Parameters** page, add both parameters to the **Selection** column. To add a parameter, select it in the **Available** list on the left and click the

arrow. You can also click the button to add all the parameters. Click **Next** after both parameters are in **Selection**.

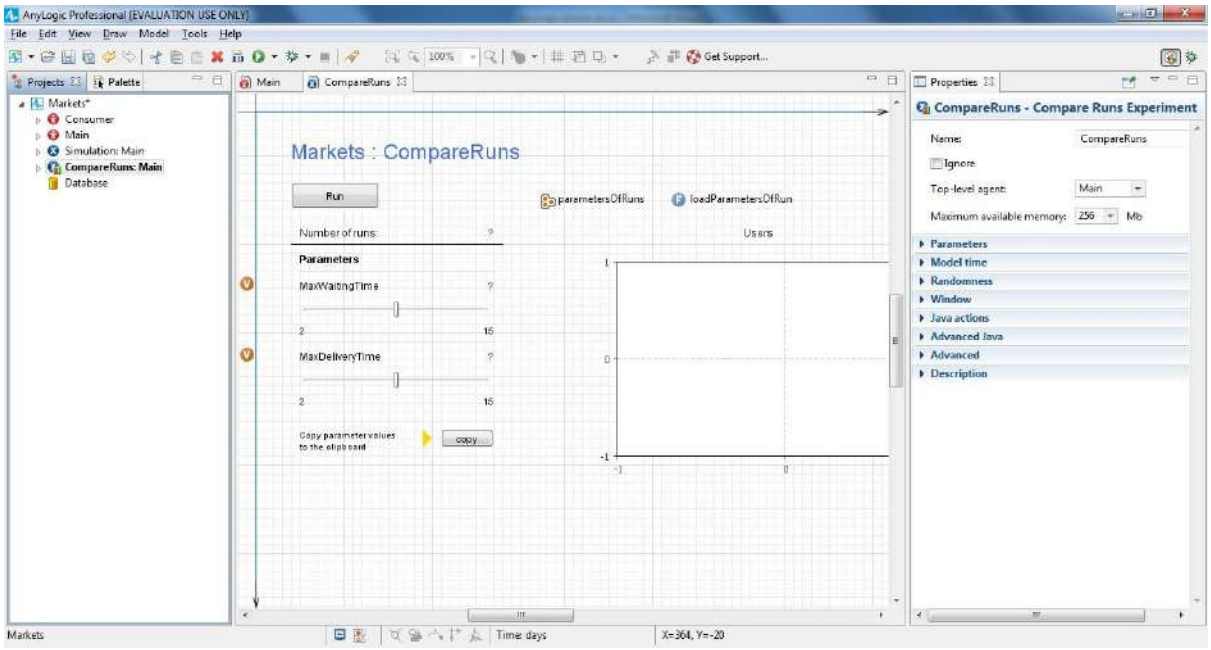




In the **Type** column, select **dataset**. In the **Chart Title** column, type Users. In the **Expression** column, refer to the dataset you defined in Main as root.usersDS where root is the model's top-level agent (Main)

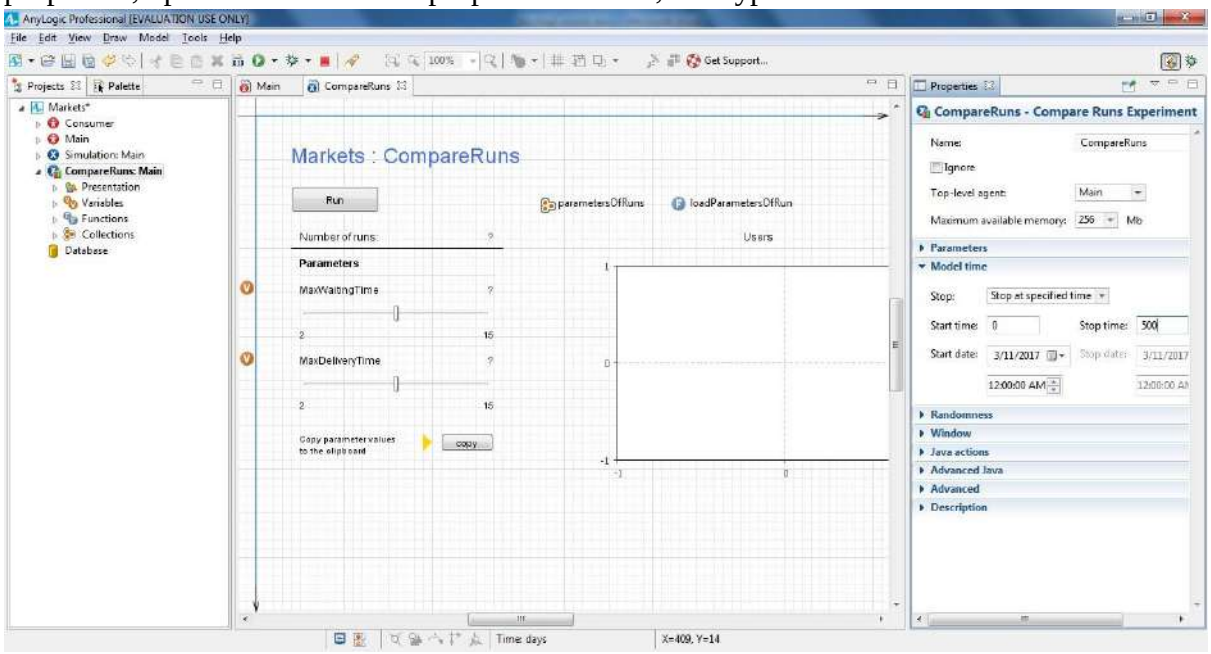


The chart will display the data collected by the dataset usersDS. Click **Finish**. The CompareRuns experiment diagram should open automatically, and you'll see the default user interface we created with the wizard.

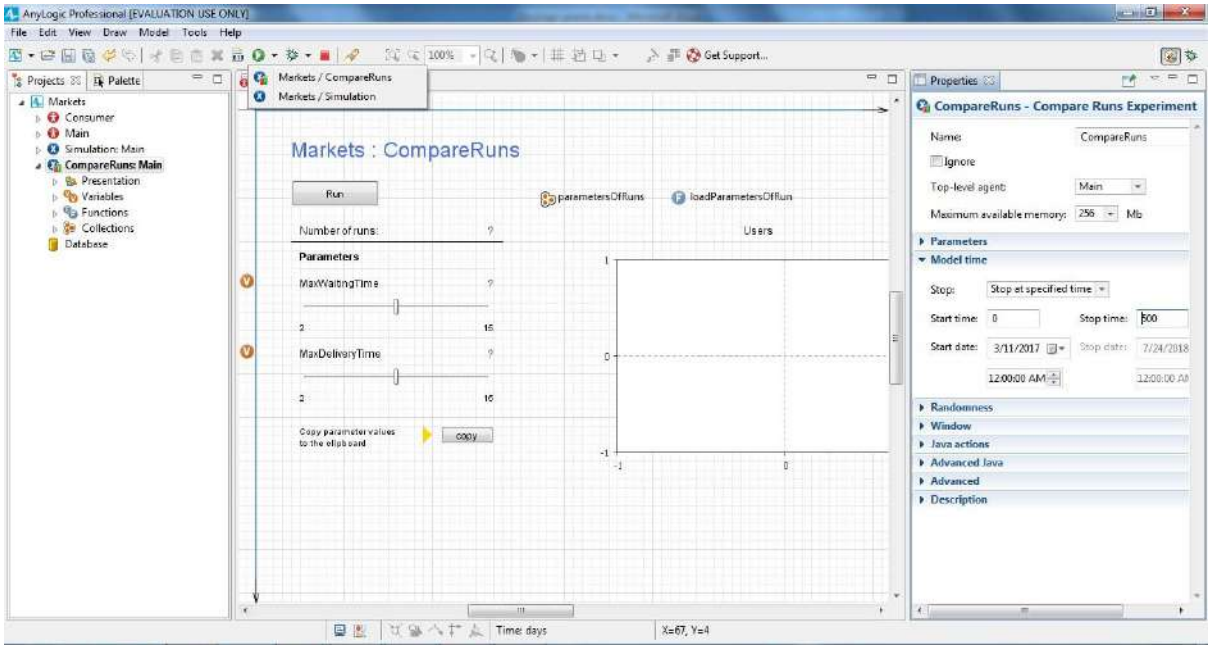


We want our experiment to simulate the model for just 500 days. To do this, select CompareRuns experiment in the **Projects** tree. In the experiment

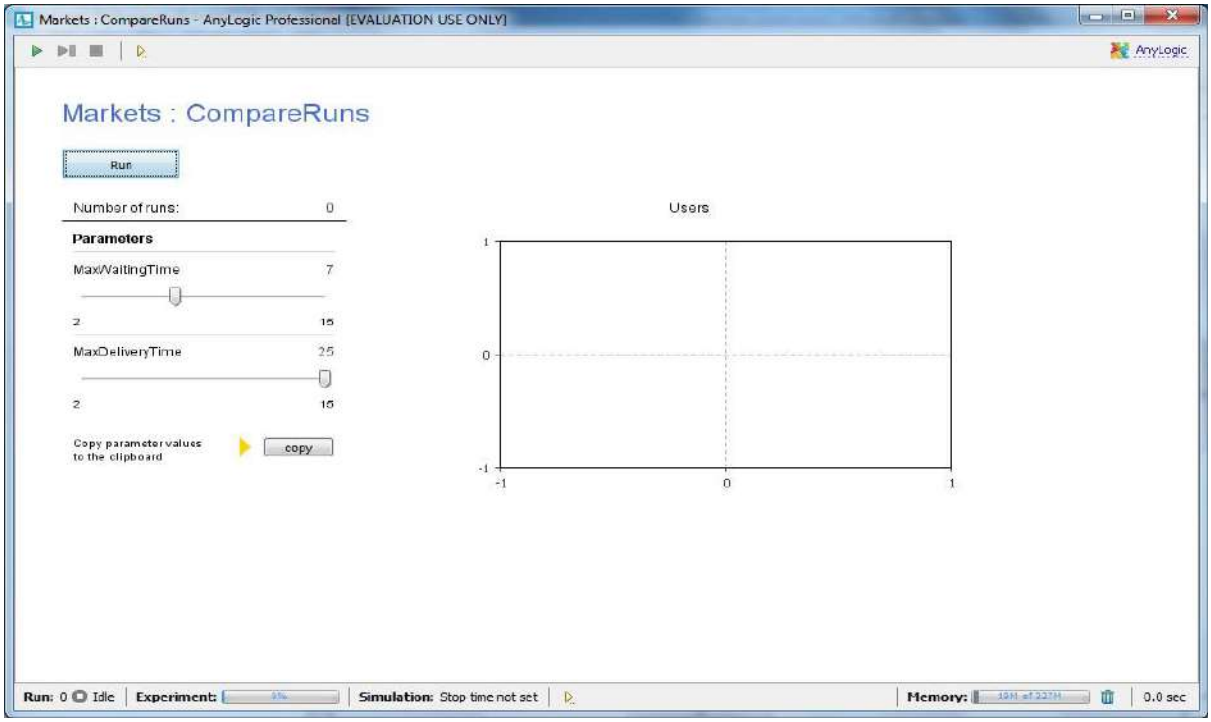
properties, open the **Model time** properties section, and type 500 in the

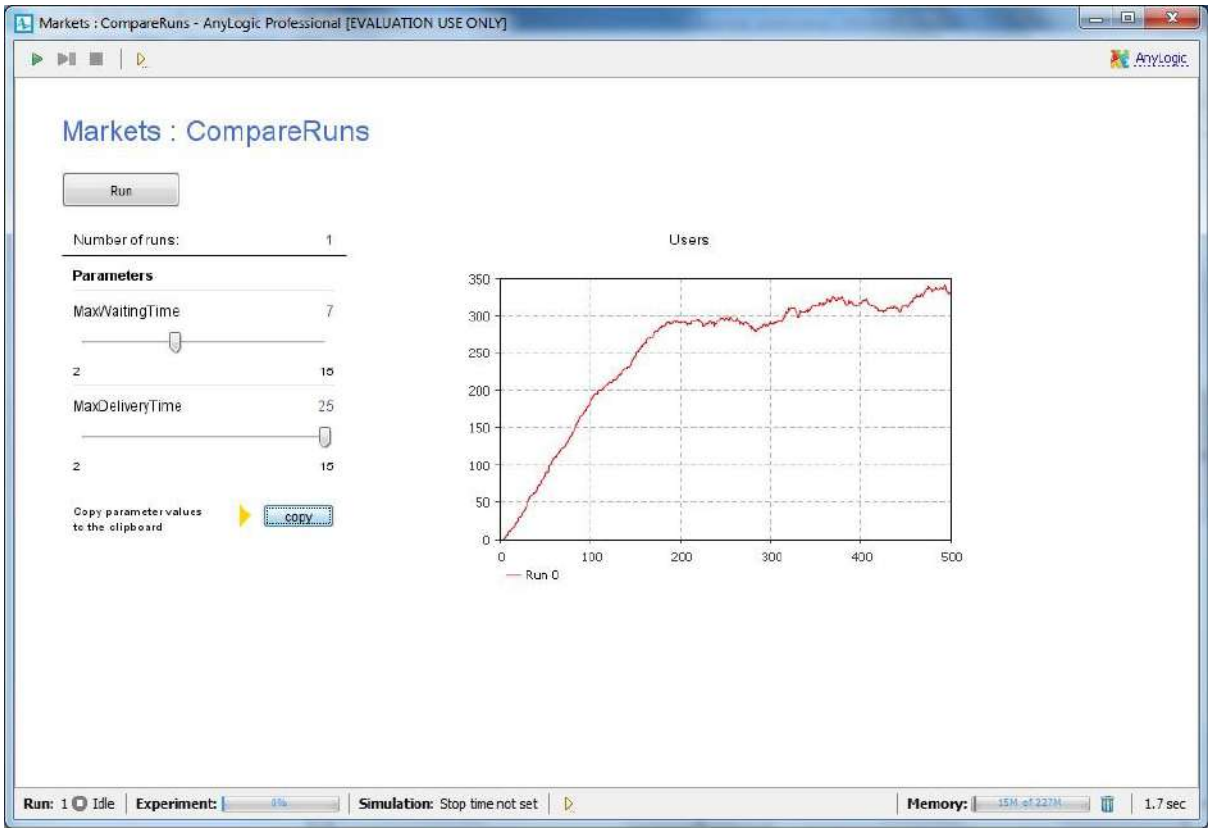


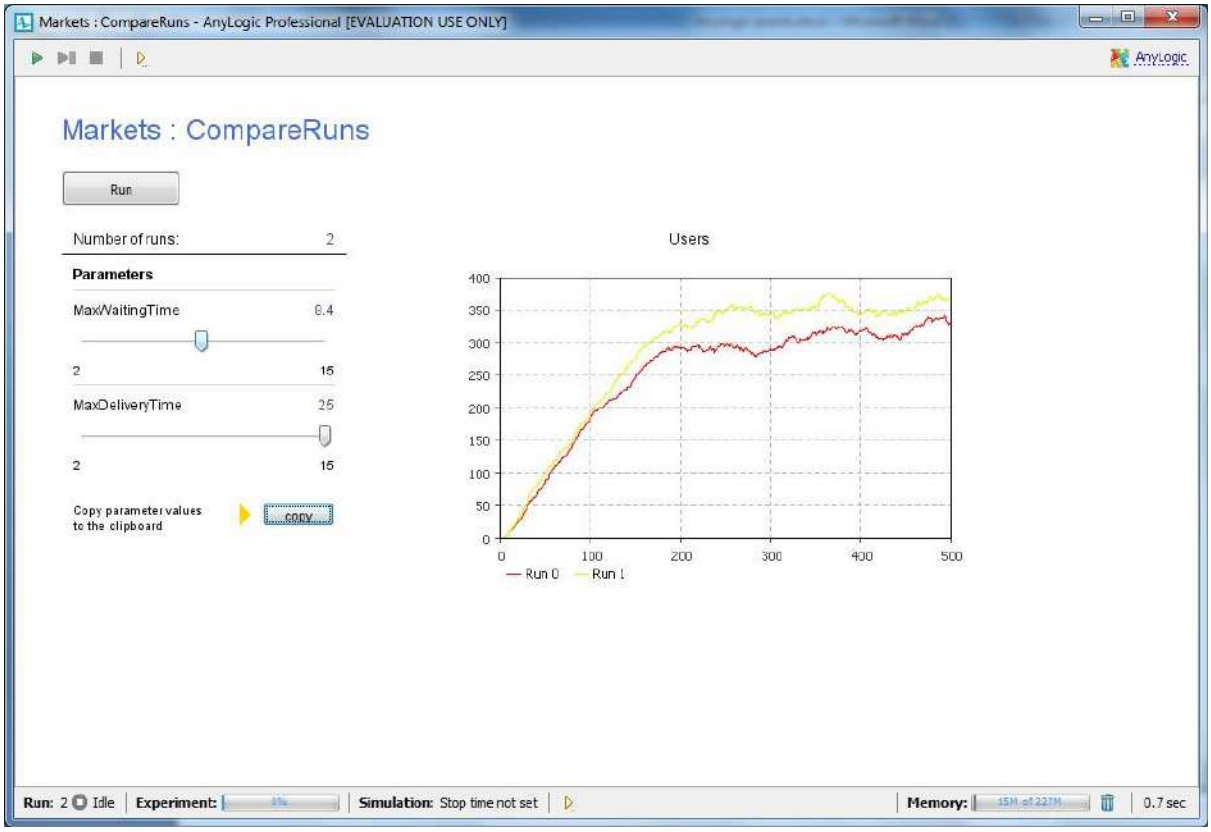
Run the experiment. Select the newly-created experiment from the **Run** list: **Market / CompareRuns**, or right-click the CompareRuns experiment in the **Projects** tree and select **Run** from the context menu.

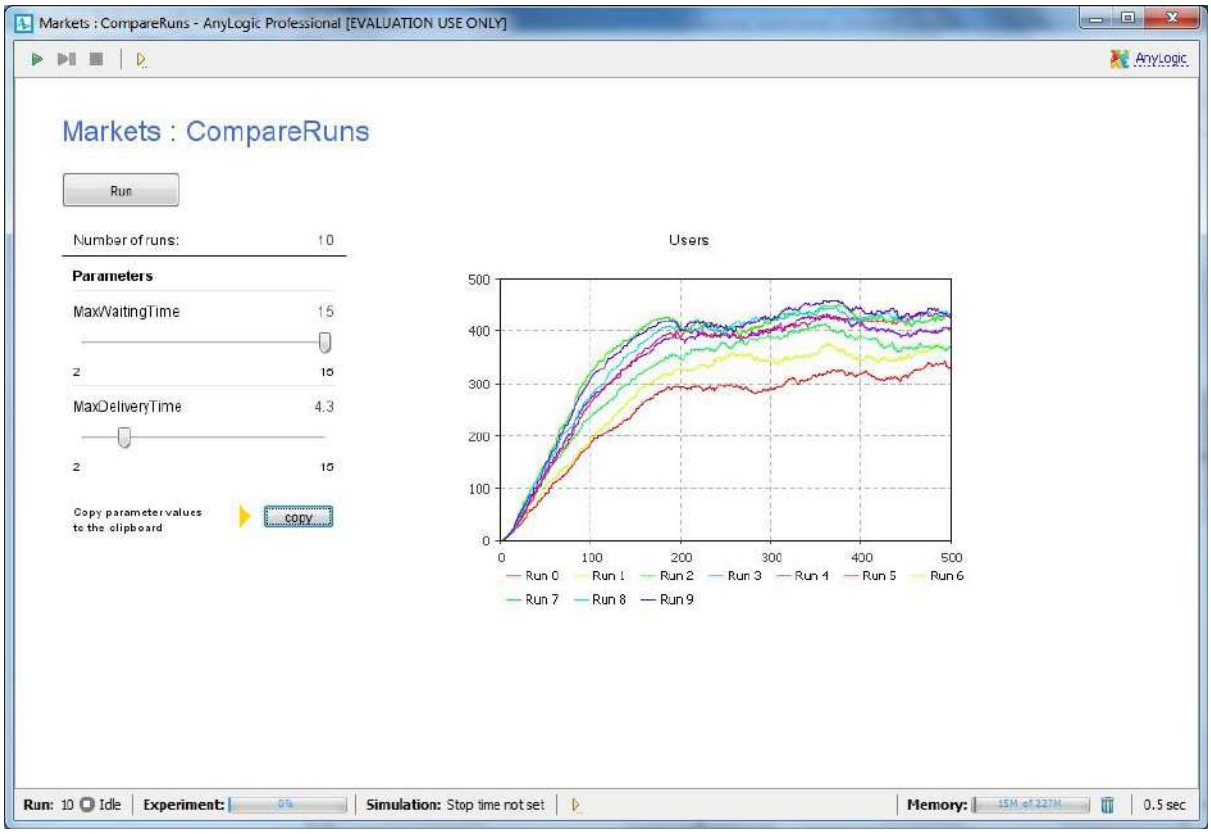


In the model window, click the **Run** button to see the result associated with the default parameter values. Afterward, change the parameter values and click **Run** again to observe the system behavior for the new settings. The chart displays all the results for your review.

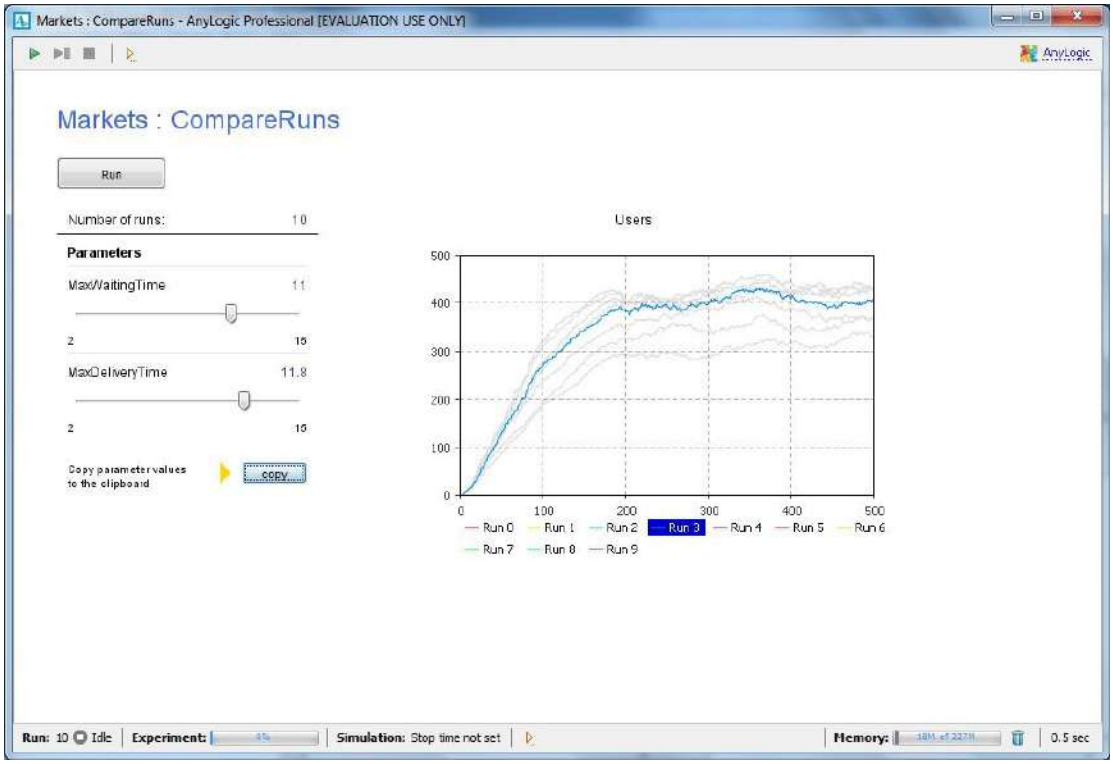

Markets : CompareRuns - AnyLogic Professional [EVALUATION USE ONLY]
AnyLogic
Markets : CompareRuns
Run
Number of runs: 0
Parameters
MaxWaitingTime 7
2 15
MaxDeliveryTime 25
2 15
Copy parameter values to the clipboard
copy
Users
1
0
-1
-1
0
1
Run: 0 Idle
Experiment:
Simulation: Stop time not set
Memory:
0.0 sec

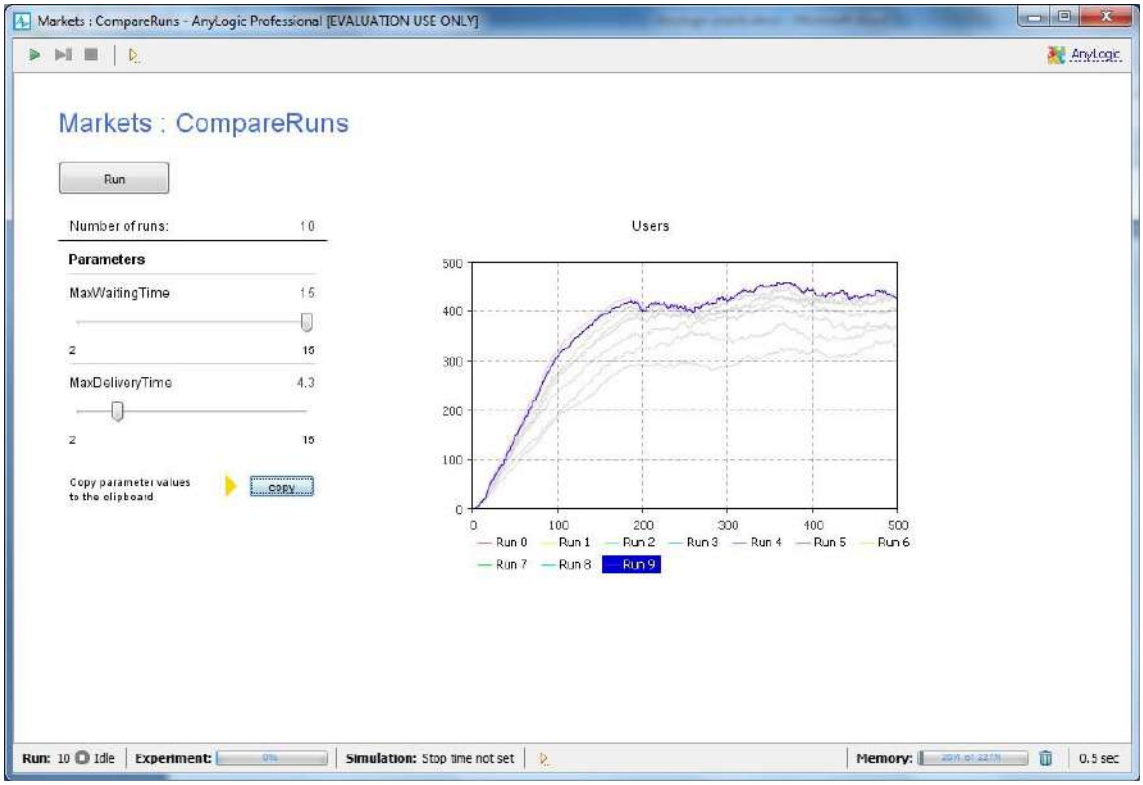

Markets : CompareRuns - AnyLogic Professional [EVALUATION USE ONLY]
AnyLogic
Markets : CompareRuns
Run
Number of runs: 1
Parameters
MaxWaitingTime 7
2 15
MaxDeliveryTime 25
2 15
Copy parameter values to the clipboard
copy
Users
350
300
250
200
150
100
50
0
0
100
200
300
400
500
Run 0
Run: 1 Idle
Experiment:
Simulation: Stop time not set
Memory:
1.7 sec





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: - 4

Aim: - Design and develop System Dynamic model by

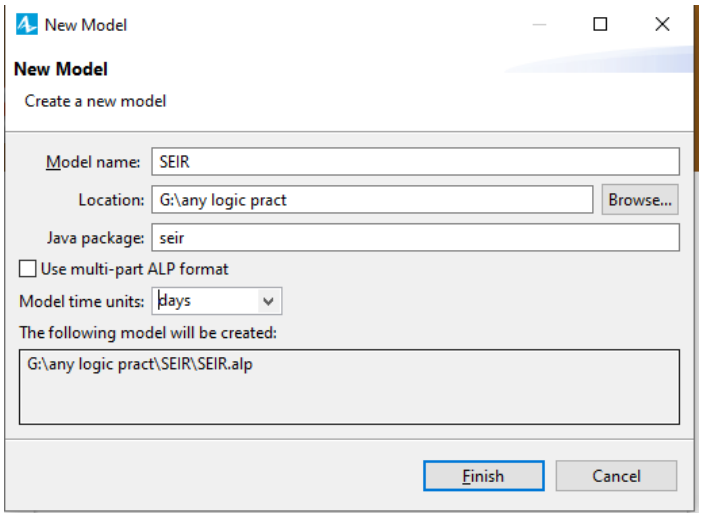
- **Creating a stock and flow diagram**
- **Adding a plot to visualize dynamics**
- **Parameter Variation**
- **Calibration**

[Use a case scenario 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.

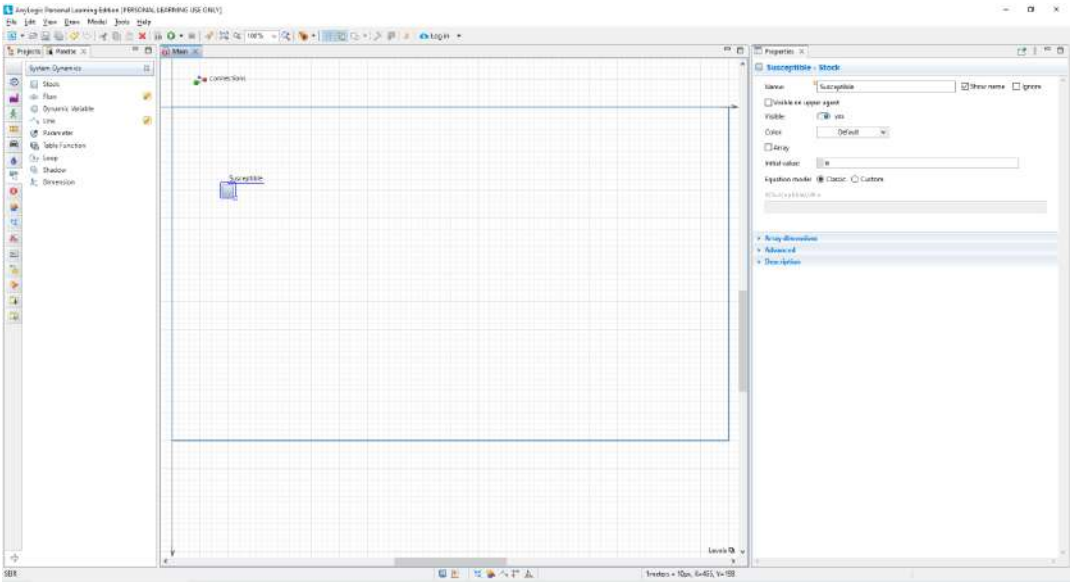
Phase 1. Creating a stock and flow diagram
Create a new model by selecting File > New > Model from the menu, and then name it SEIR. Select days as Model time units.



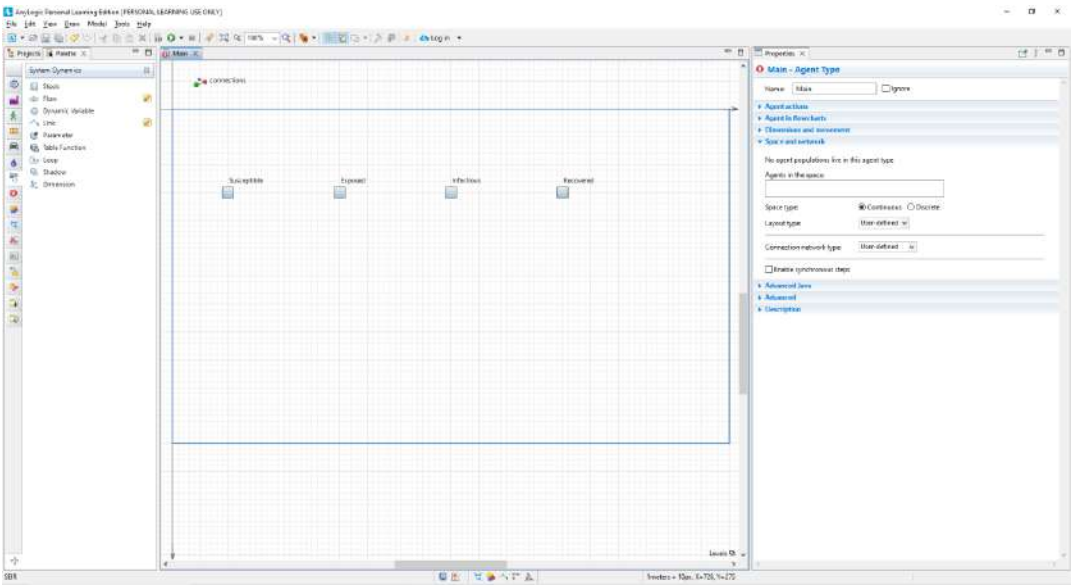
In this example, we'll consider four important characteristics:
• Susceptible - people who are not infected by the virus.

• 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.

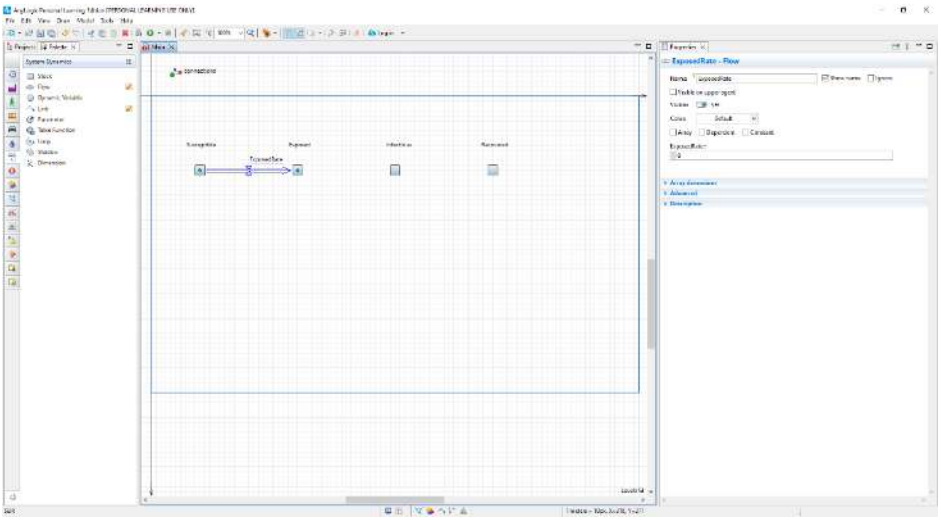


Add three more stocks. Place them as shown in the figure and name them Exposed, Infectious, and Recovered.

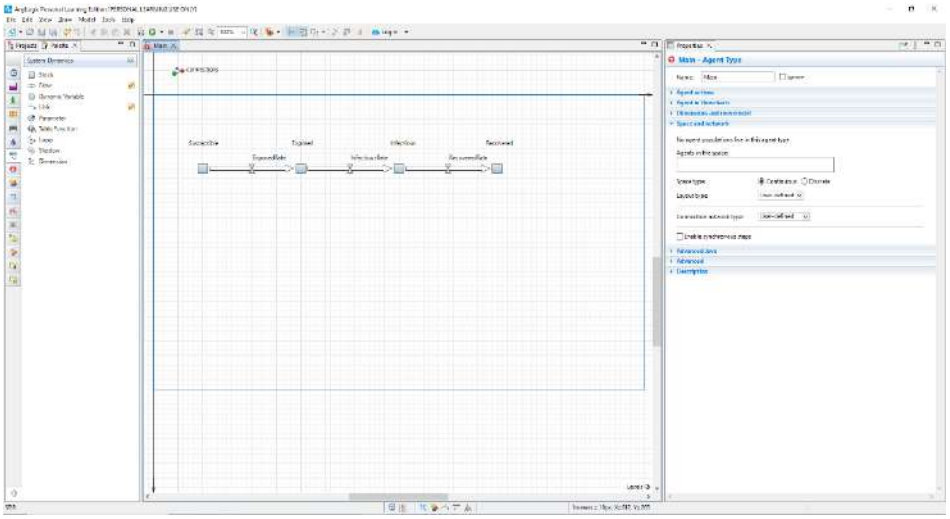


The flow's arrow shows its direction.
In our model, susceptible people are exposed to the virus, become infectious, and then recover. It's a progression that requires our model to use three flows to drive people from one stock to the next.

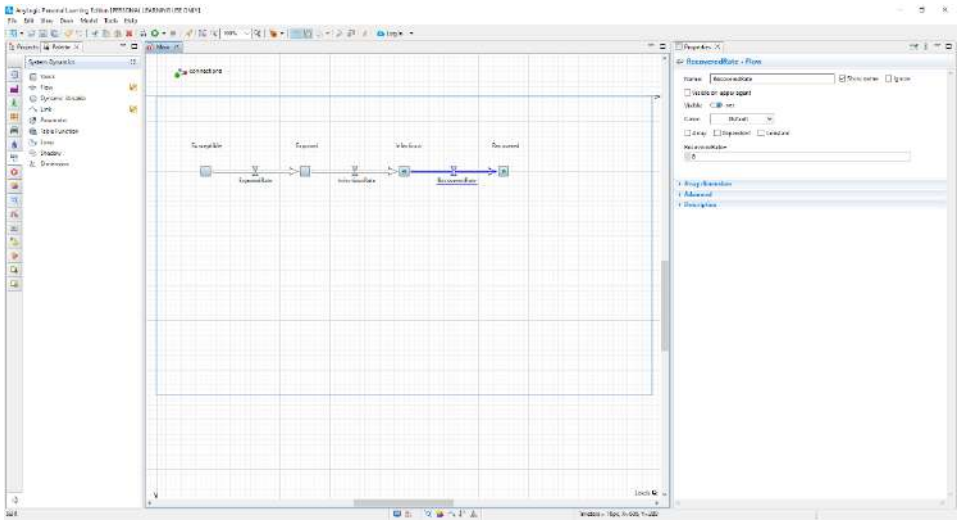
Add the first flow that flows from the stock Susceptible to Exposed. Double-click the stock where the flow flows out (Susceptible), and then click the stock where it flows in (Exposed).
Name the flow ExposedRate.



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

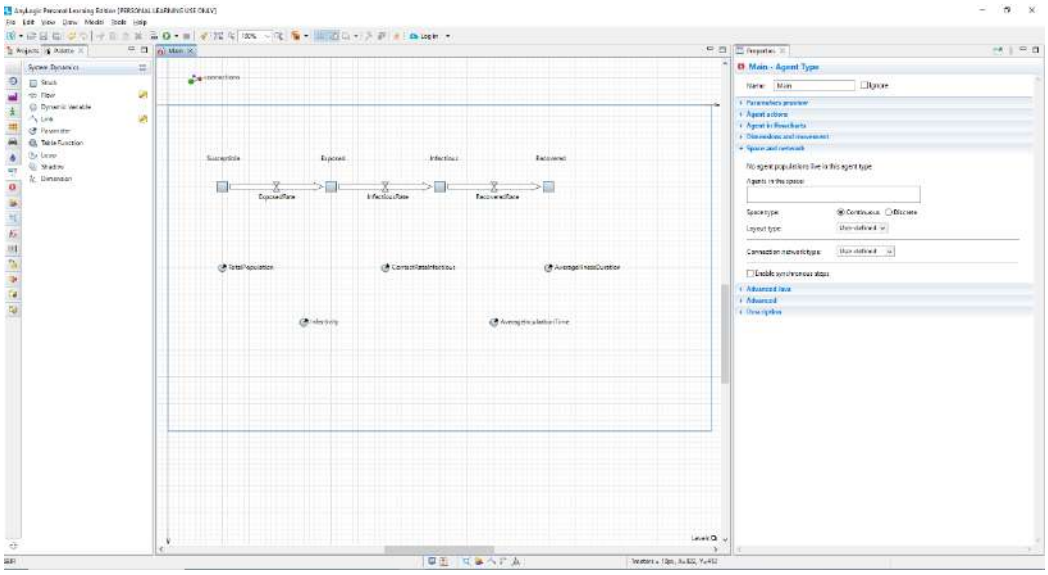


Rearrange the flow names as shown in the figure below. To do this, select a flow and then drag its name.

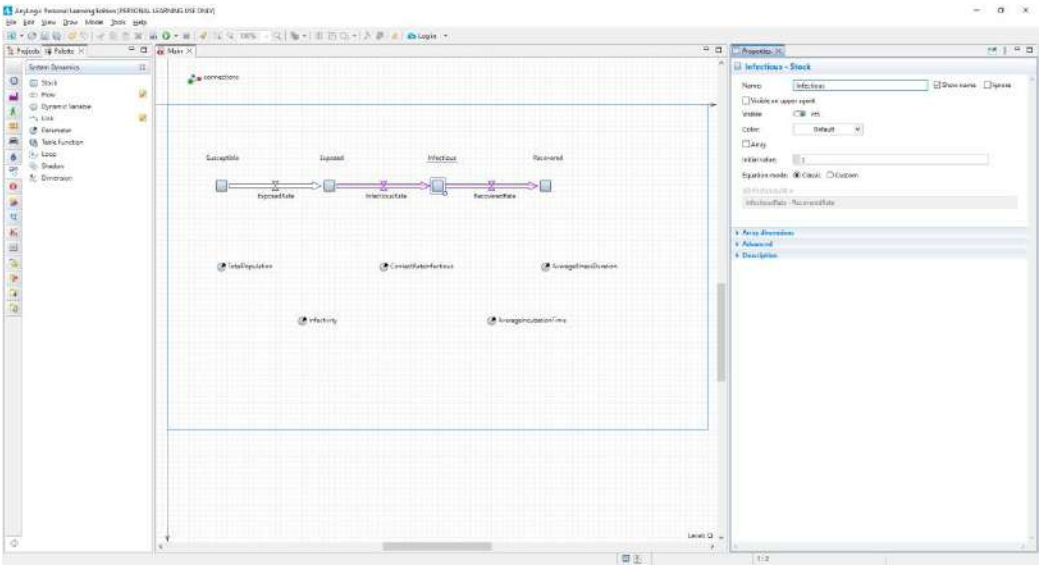


Add five Parameters, name them, and then define their default values according to the information below:

•TotalPopulation = 10000
•Infectivity = 0.6
•ContactRateInfectious = 1.25
•AverageIncubationTime = 10
•AverageIllnessDuration = 15

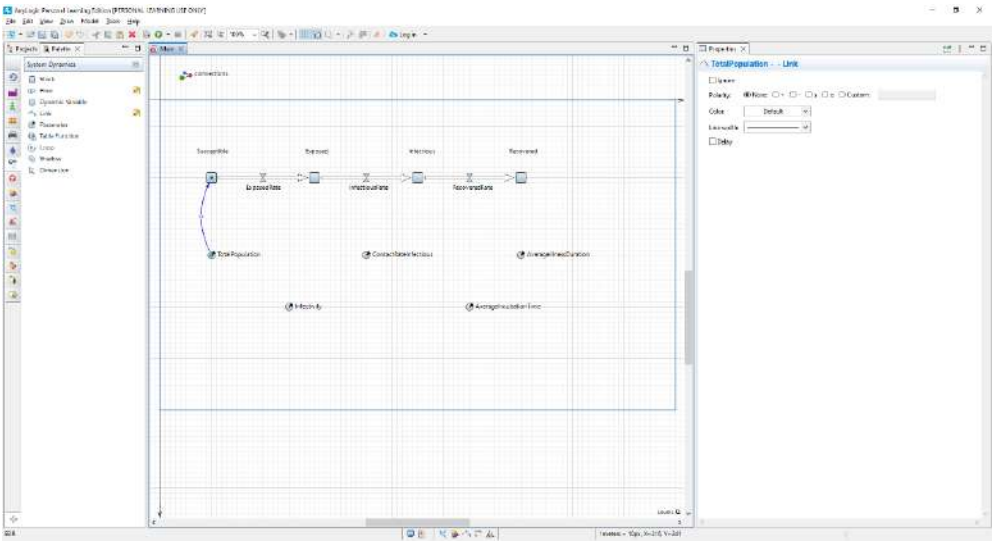


Define the number of infected people by specifying 1 as the Initial Value of the stock Infectious.

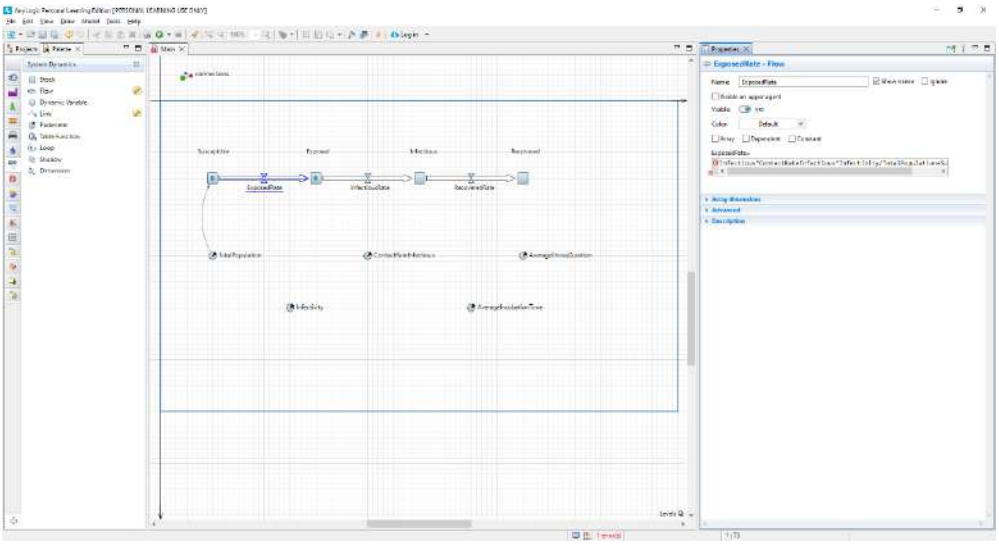


Define the Initial Value for the stock Susceptible: TotalPopulation-1.

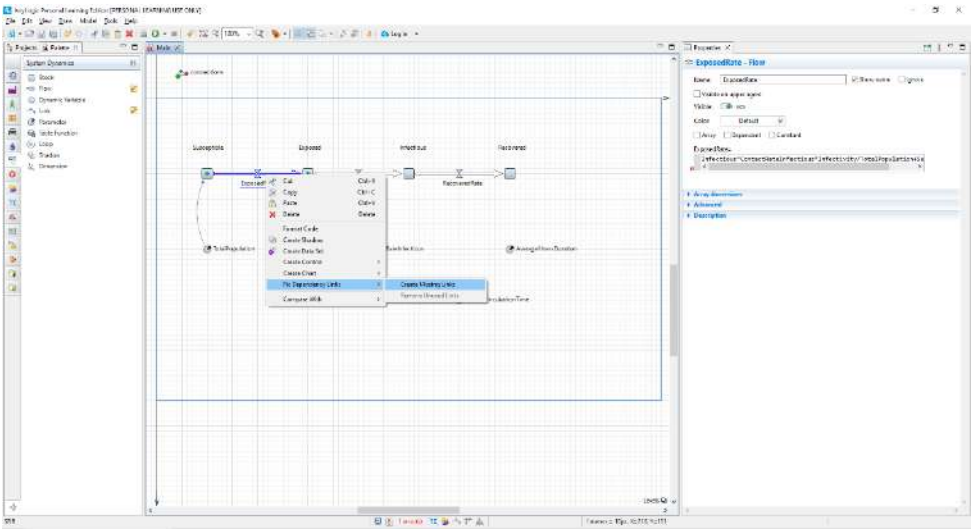
Draw a dependency link from TotalPopulation to Susceptible.
In the System Dynamics palette, double-click the Link element, click TotalPopulation, and then click the stock Susceptible. You should see the link with small circles drawn on its end points:

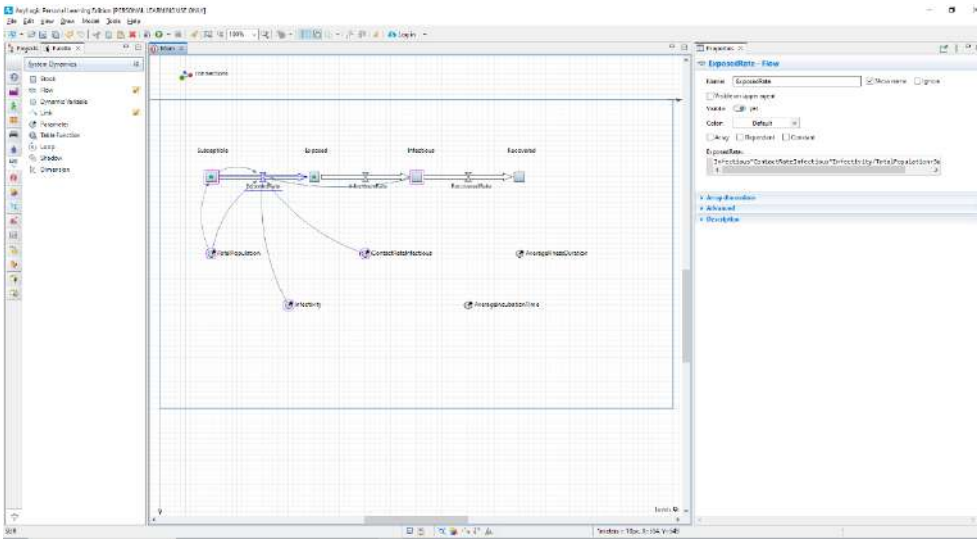


Let's define the formula for the flow ExposedRate.
Click the flow and define the following formula using the Code Completion assistant:
Infectious*ContactRateInfectious*Infectivity/TotalPopulation+Susceptible



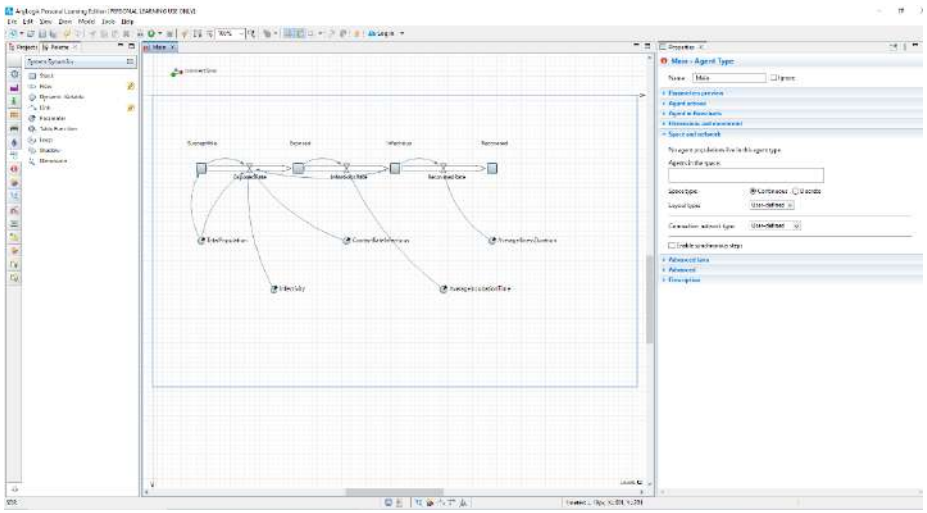
Right-click ExposedRate flow in the graphical diagram and choose Fix Variable Links > Create Missing Links from the context menu. Afterward, you should see the links in the stock and flow diagram:





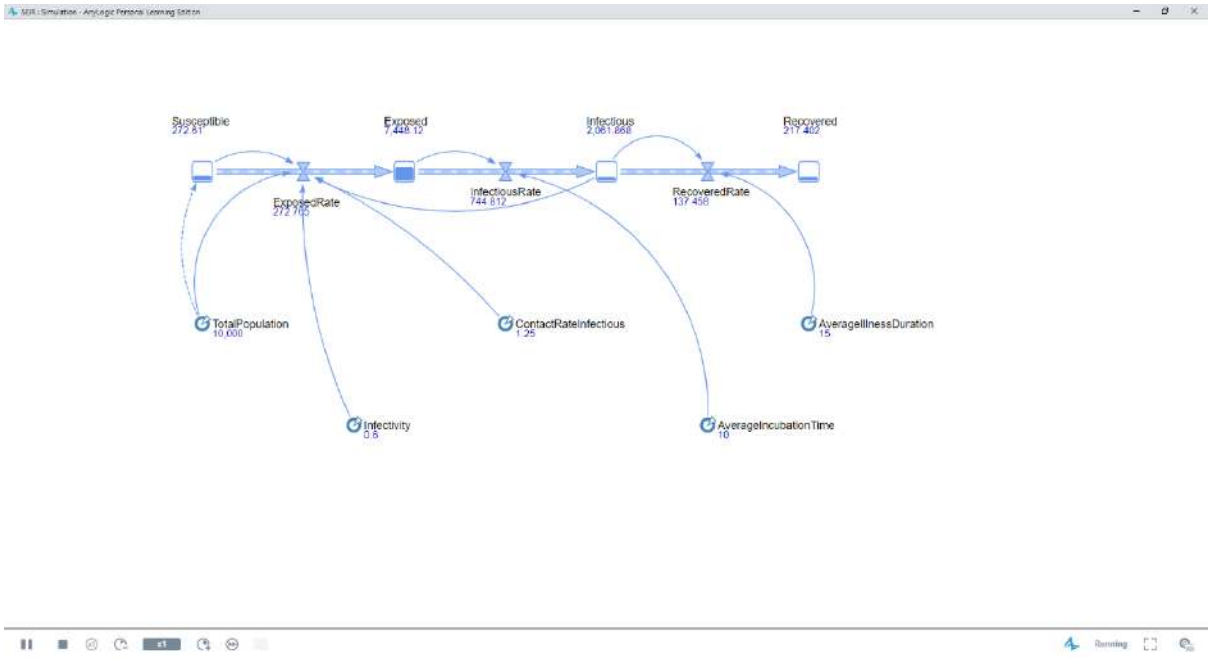
Define the following formula for InfectiousRate:

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.

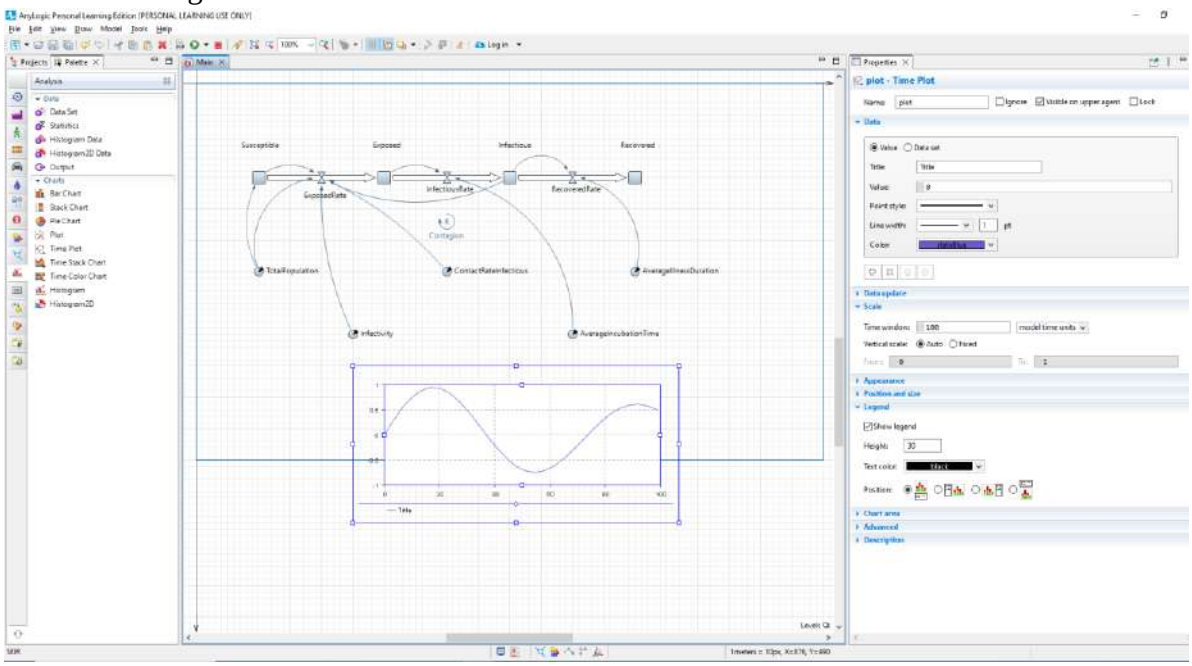


Adding a plot to visualize dynamics

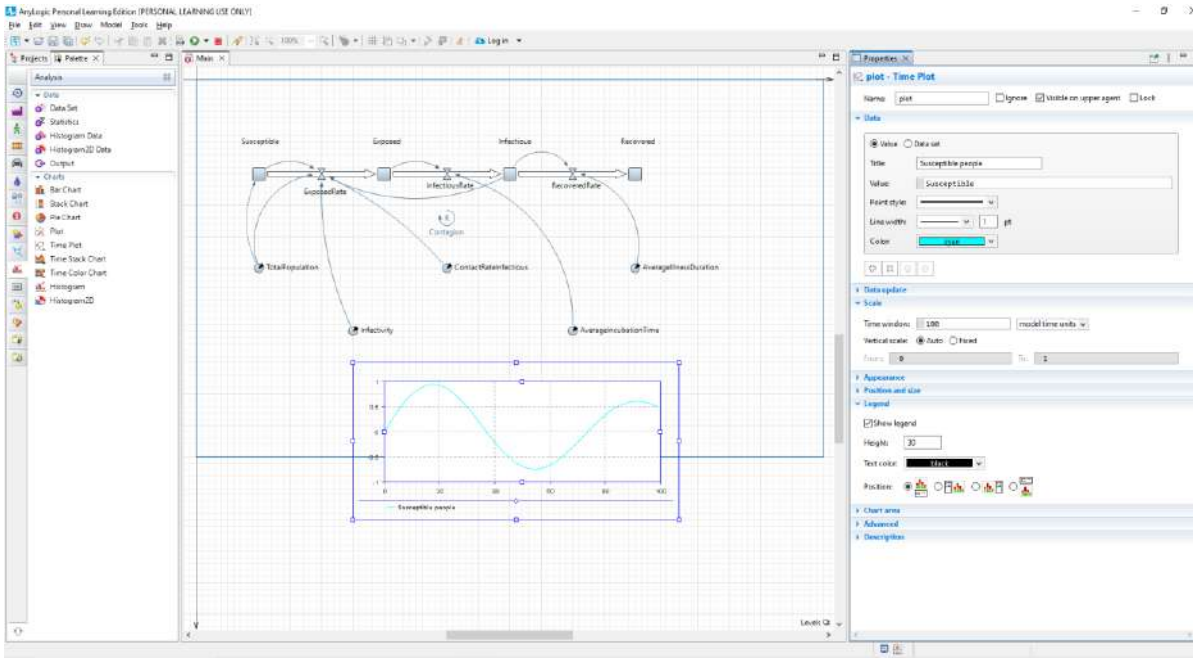
We'll add a loop identifier for one loop to show you.
Drag the Loop element from the System Dynamics palette on to the diagram, and then place it as shown in the figure.

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.

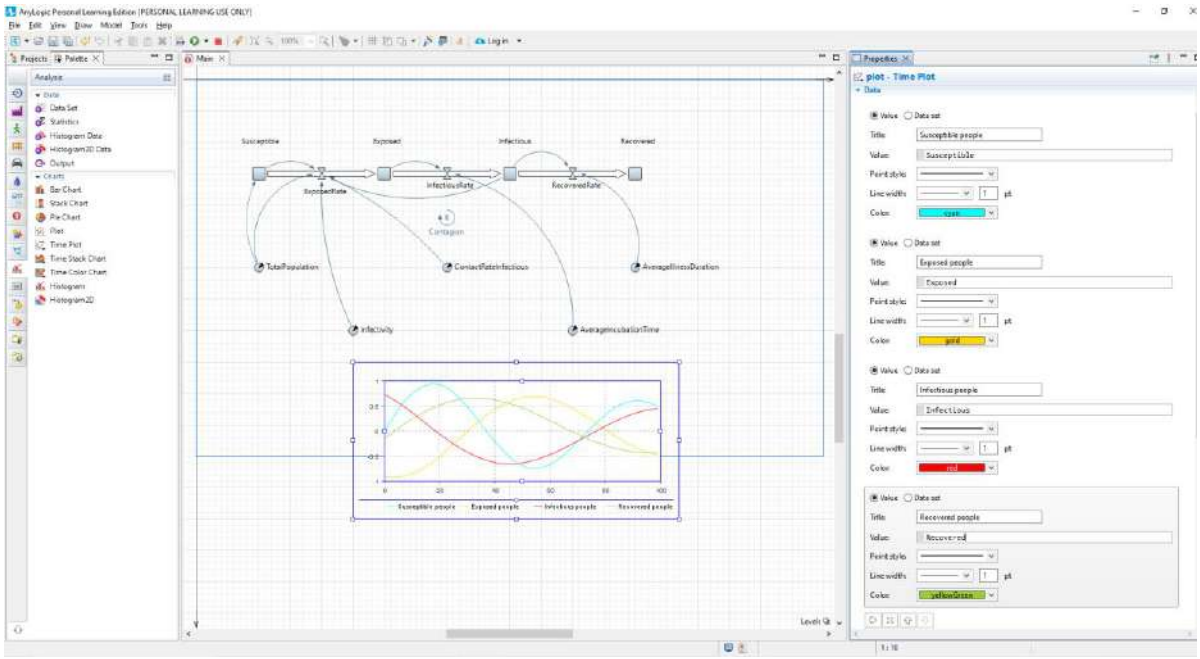
Let’s add a time chart to plot Susceptible, Exposed, Infectious, and Recovered people.
Drag the Time Plot from the Analysis palette on to the diagram, and extend the time plot as shown in the figure below:



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.

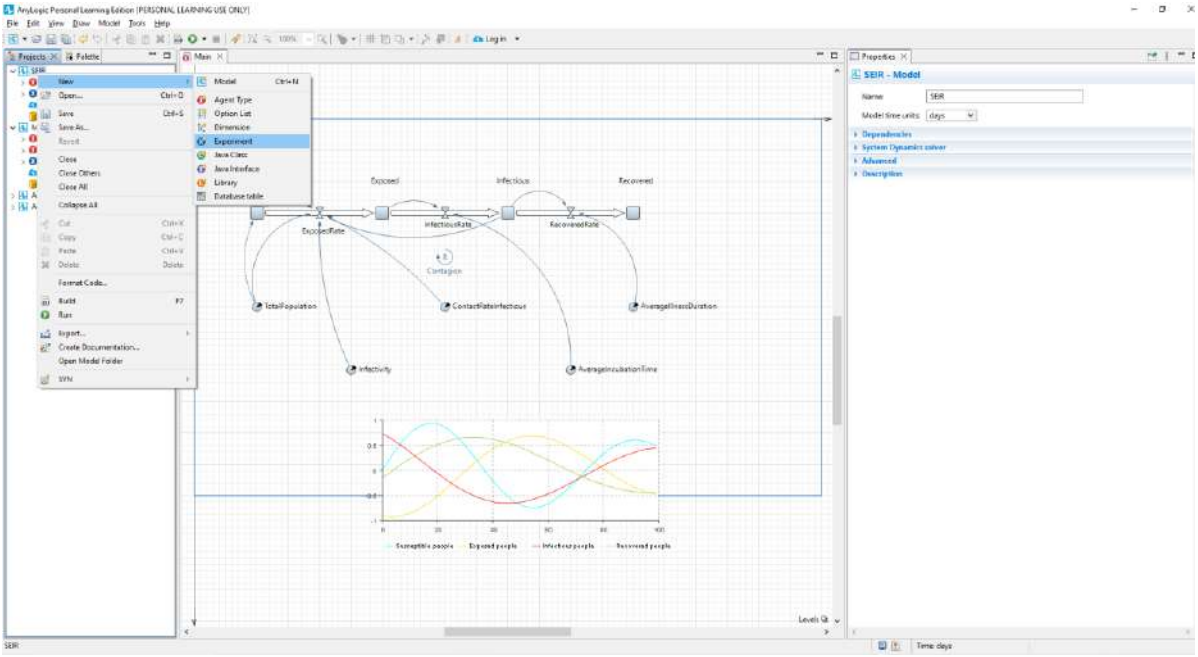


We've finished our last model. Now, run the model and use the chart you added to view its dynamics.

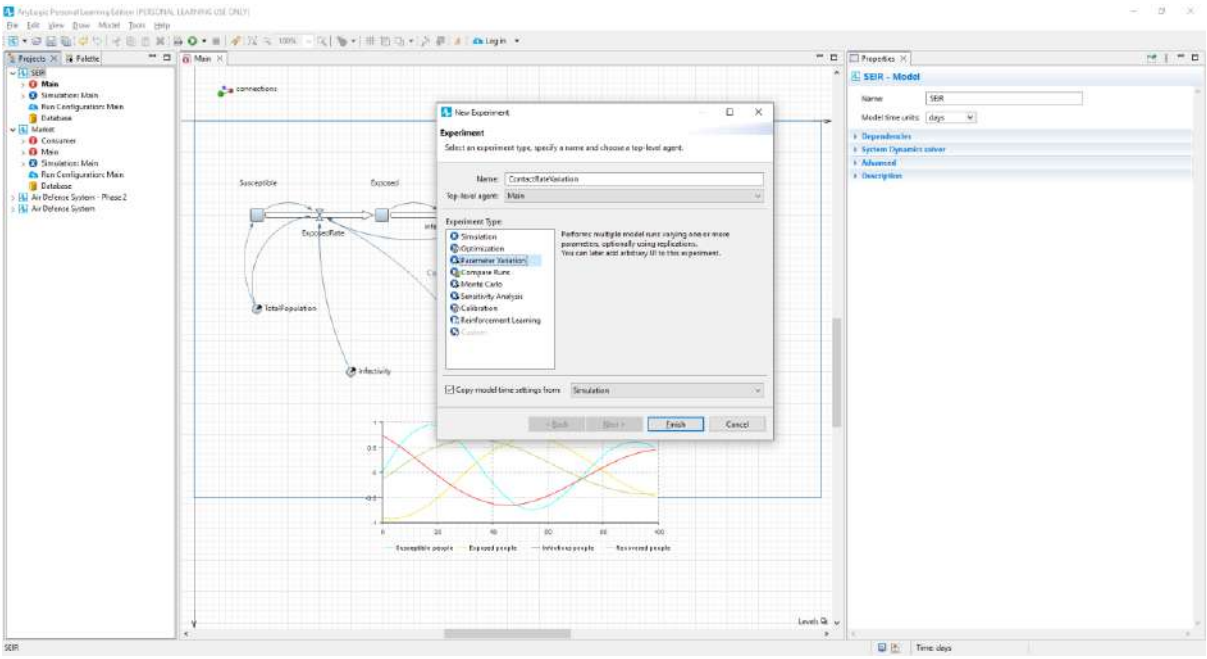
Parameter Variation

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.

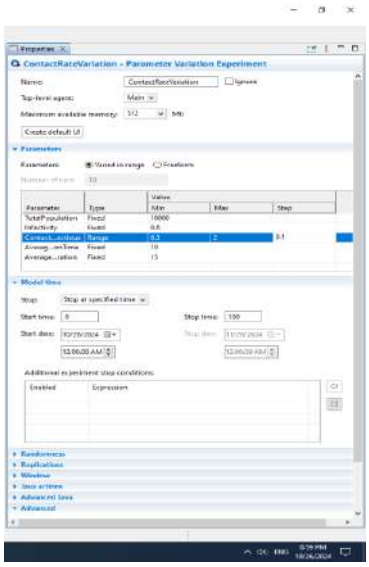
To add an experiment to the model, right-click the model item (SEIR) in the Projects tree, point to New, and then click Experiment.



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.

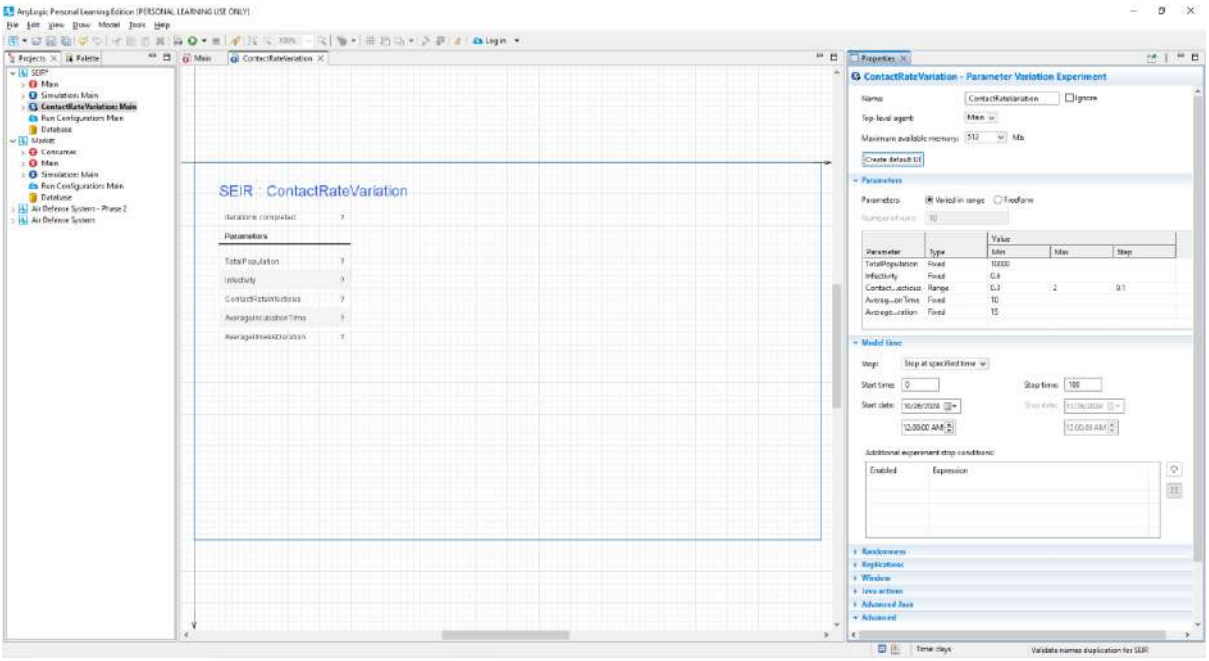


In the experiment's properties, open the Parameters section. The parameters of the experiment's top-level agent (in our case, Main) display.
To ensure our experiment varies the contact rate, locate the table's ContactRateInfectious parameter and change its Type to Range



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

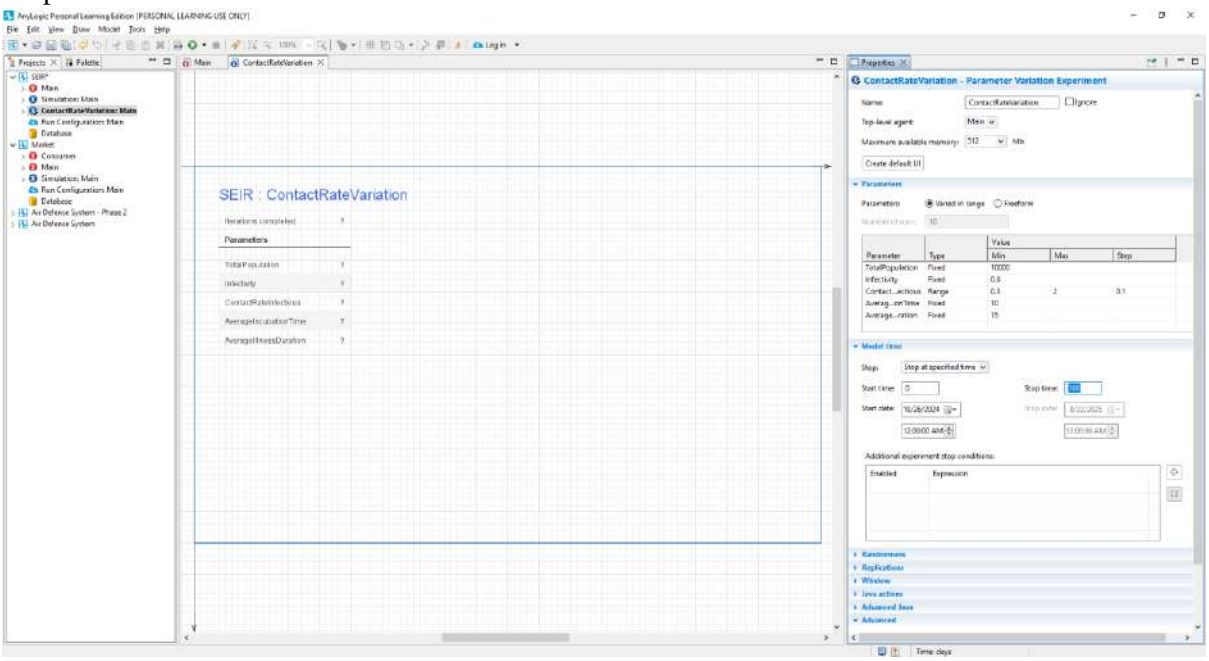
In the Properties area, click Create default UI.



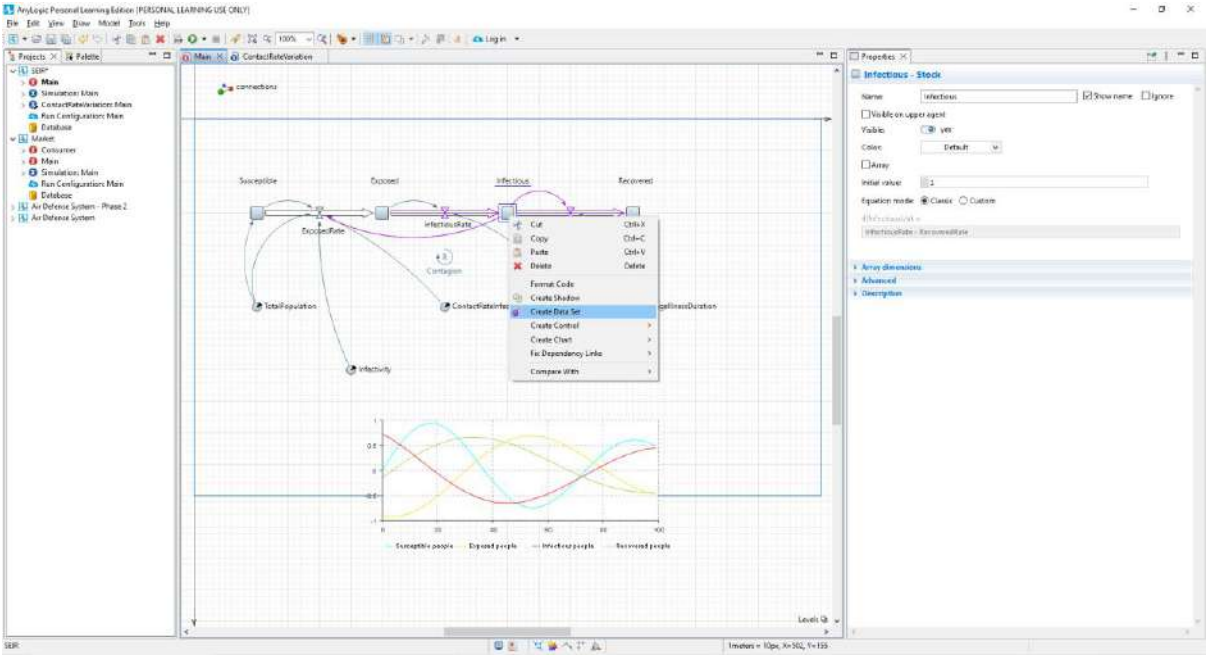
The experiment diagram will display the simple user interface

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

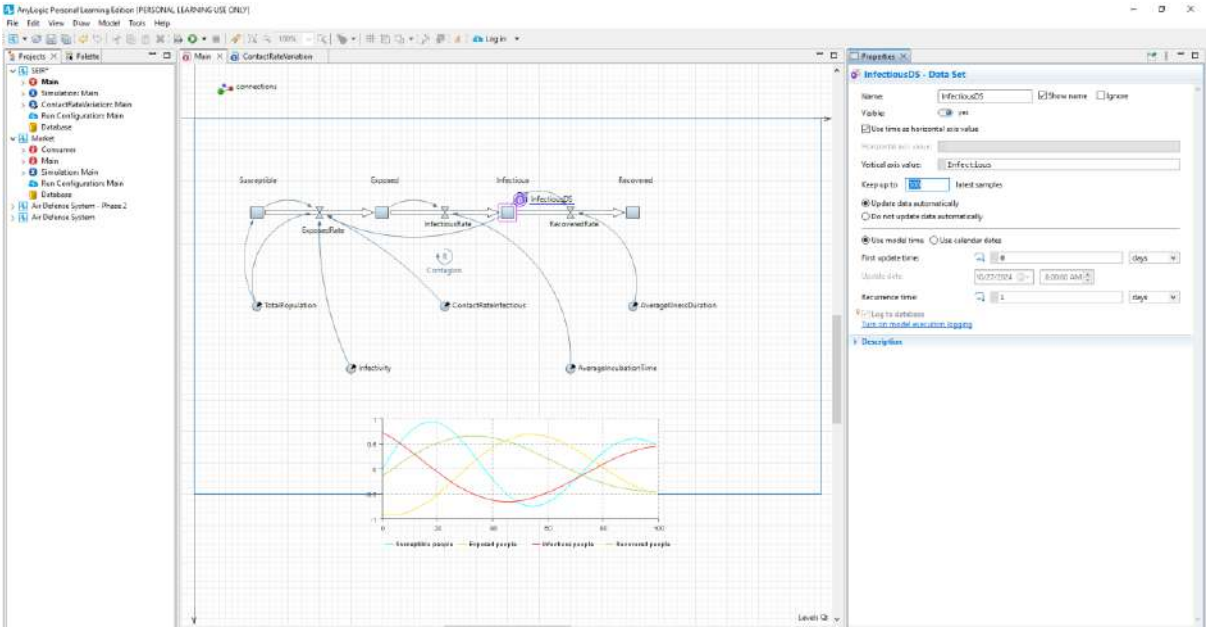


Now, we'll add a time plot to display our experiment's results. Our first step is to gather data about the number of infectious persons.
Open the Main diagram, right-click the Infectious stock and then click Create Data Set.



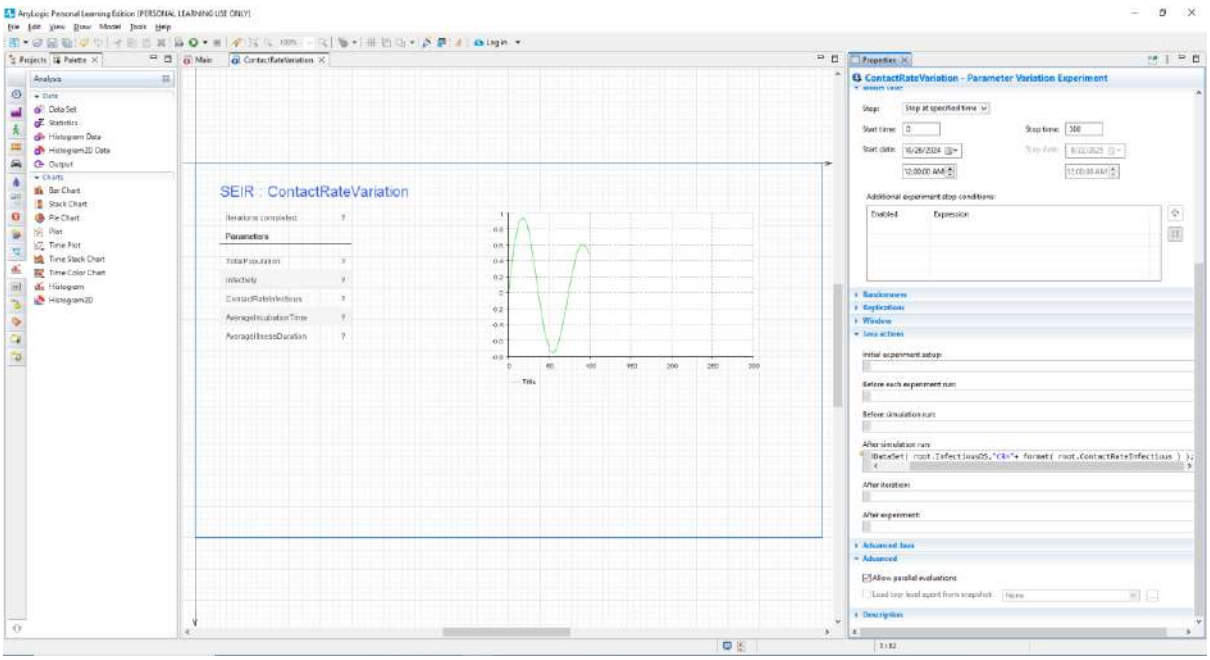
After the InfectiousDS data set displays, navigate to its properties. Since we want to view the infectious diseases dynamics, leave the Use time as horizontal axis value checkbox selected. Select Update data automatically and leave Recurrence time: 1 to add one data sample to our dataset for each model life day.

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

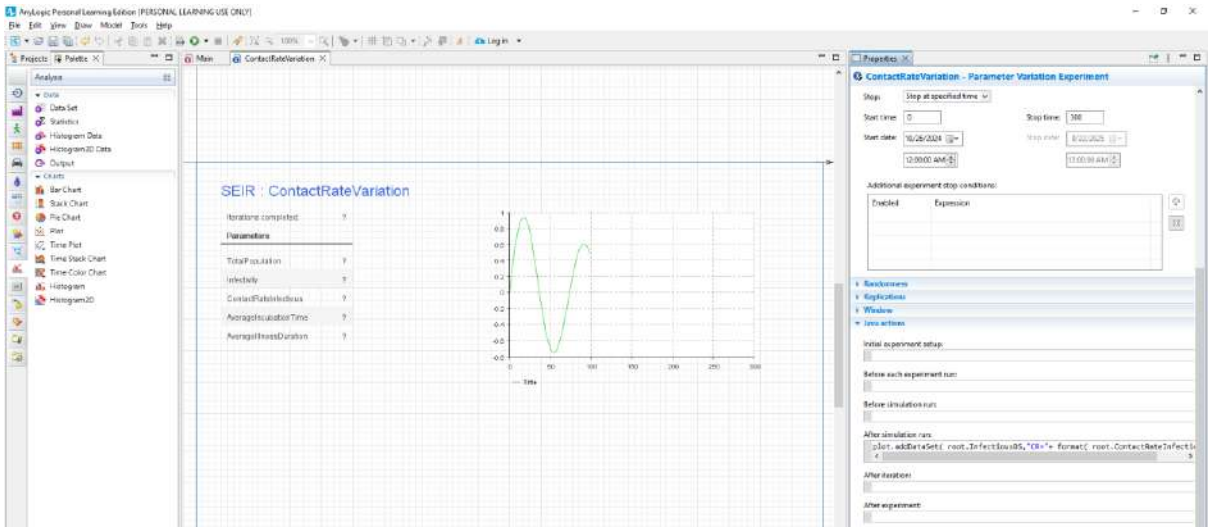


We're ready to add a chart to the ContactRateVariation experiment diagram that will display our results.
Open ContactRateVariation diagram, and drag the Time Plot from the Analysis palette.

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.

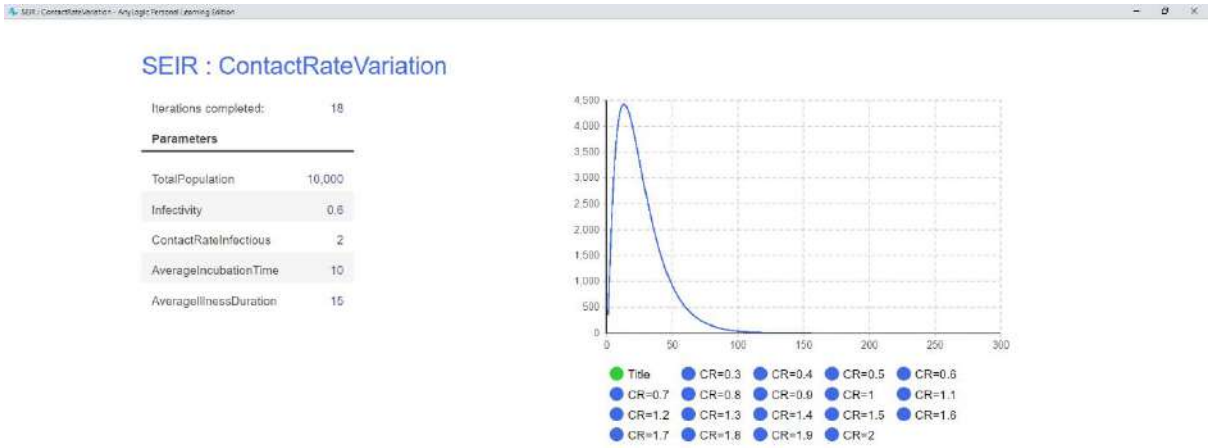


The plot's curves will each display the results from one model run: the history of disease spread for a contact rate collected by the InfectiousDS dataset.
Click ContactRateVariation in the Projects tree to open its properties, and then add data to the plot by navigating to the Java actions section and typing the following code in the
After simulation run field:
plot.addDataSet(root.InfectiousDS,"CR="+ format(root.ContactRateInfectious));



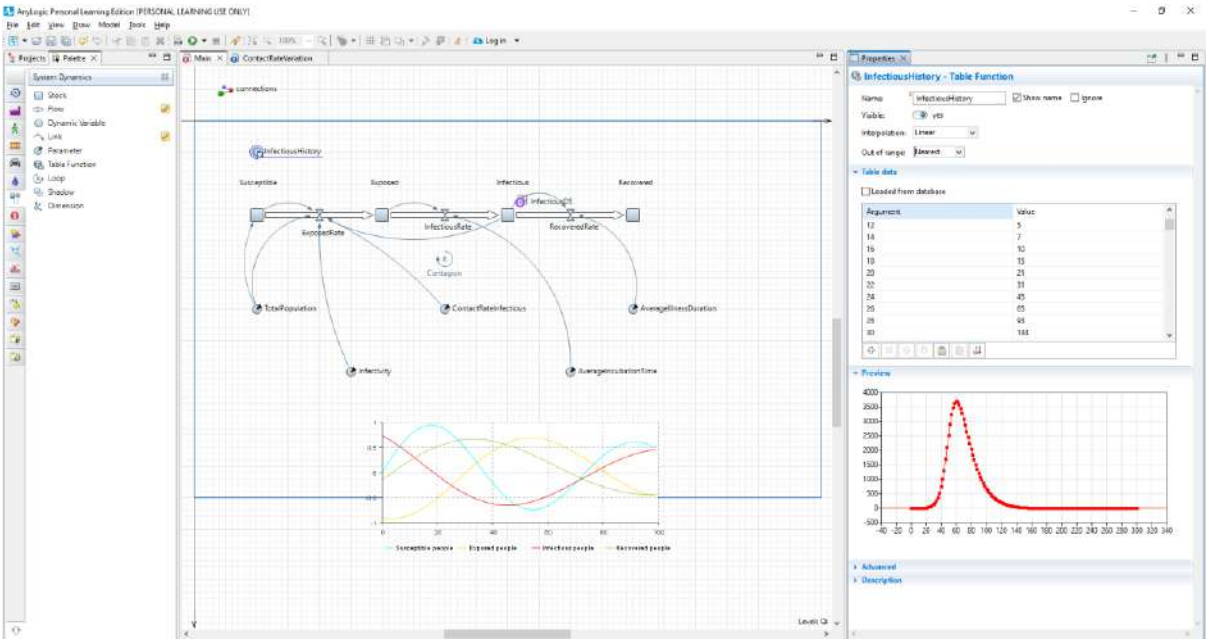
Open the ContactRateVariation experiment properties Advanced section and then clear the Allow parallel evaluations checkbox.

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.



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.

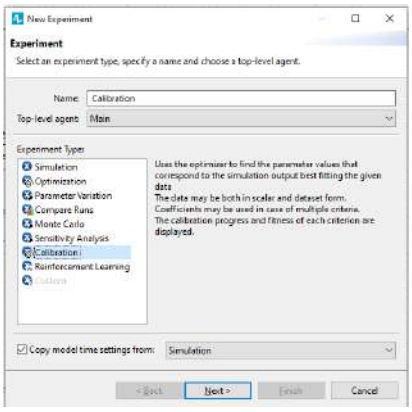


Open the HistoricData.txt file from AnyLogic folder/resources/AnyLogic in 3 days/SEIR. The AnyLogic folder is the location on your computer where you installed AnyLogic, such as Program Files/AnyLogic 7 Professional.
Copy the text file's contents to the Clipboard, go to the table function properties 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.

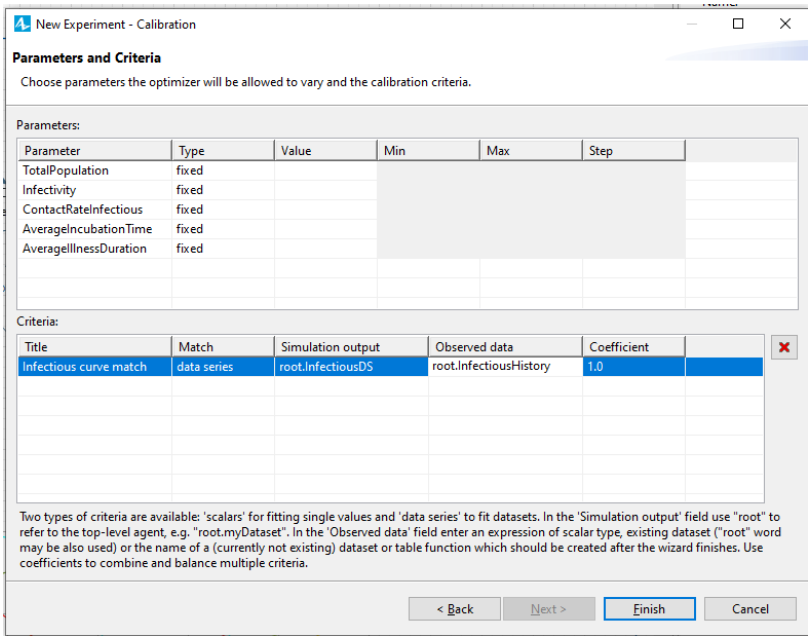
Set the Out of range option to Nearest to ensure the function correctly addresses cases where the function's argument exceeds the value of 300 that we defined in the Table data.

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

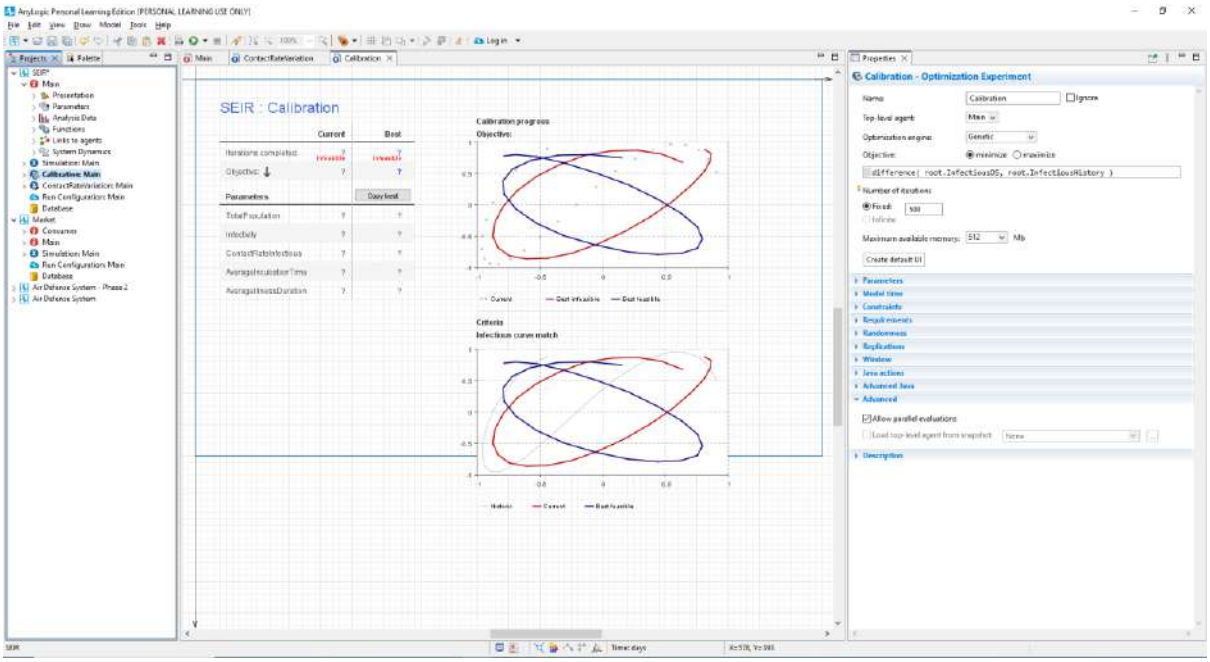


Change the parameter types we want to calibrate (Infectivity and ContactRateInfectious) from fixed to continuous, and then set the range's Min and Max values as shown in the figure below. In the Criteria table shown below, enter the following information.
Title: Infectious curve match

Match: data series
Simulation output: root.InfectiousDS
Observed data: root.InfectiousHistory
Coefficient: 1.0

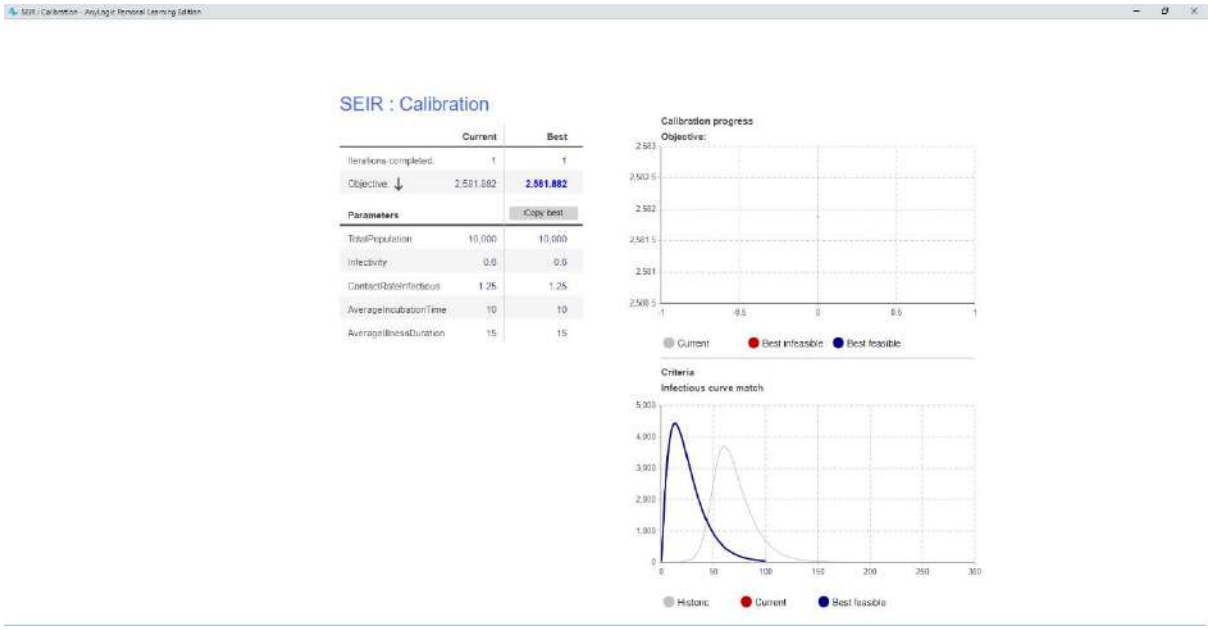


Click Finish. The Calibration experiment diagram will display the configured user interface (UI).



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

Practical No: - 5

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

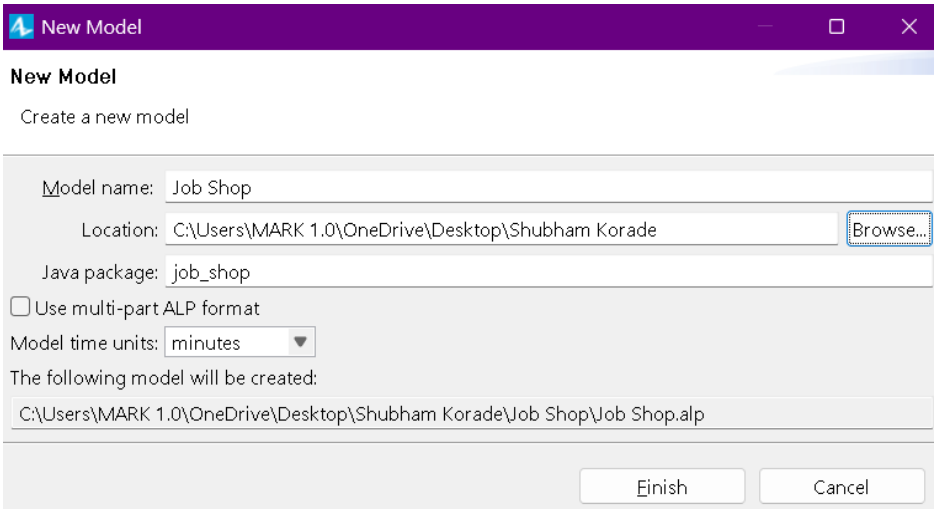
- Creating a simple model
- Adding resources
- Creating 3D animation
- Modelling delivery

[Use a case scenario like a company manufacturing and shipping]

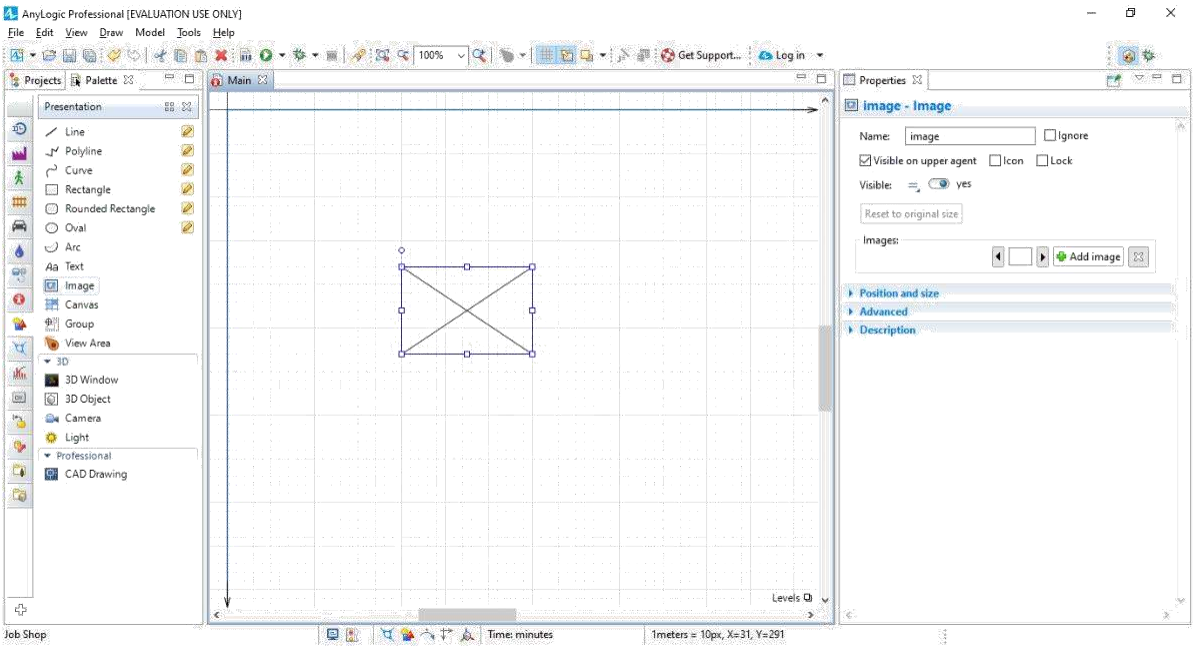
Code:-

Creating Simple Model

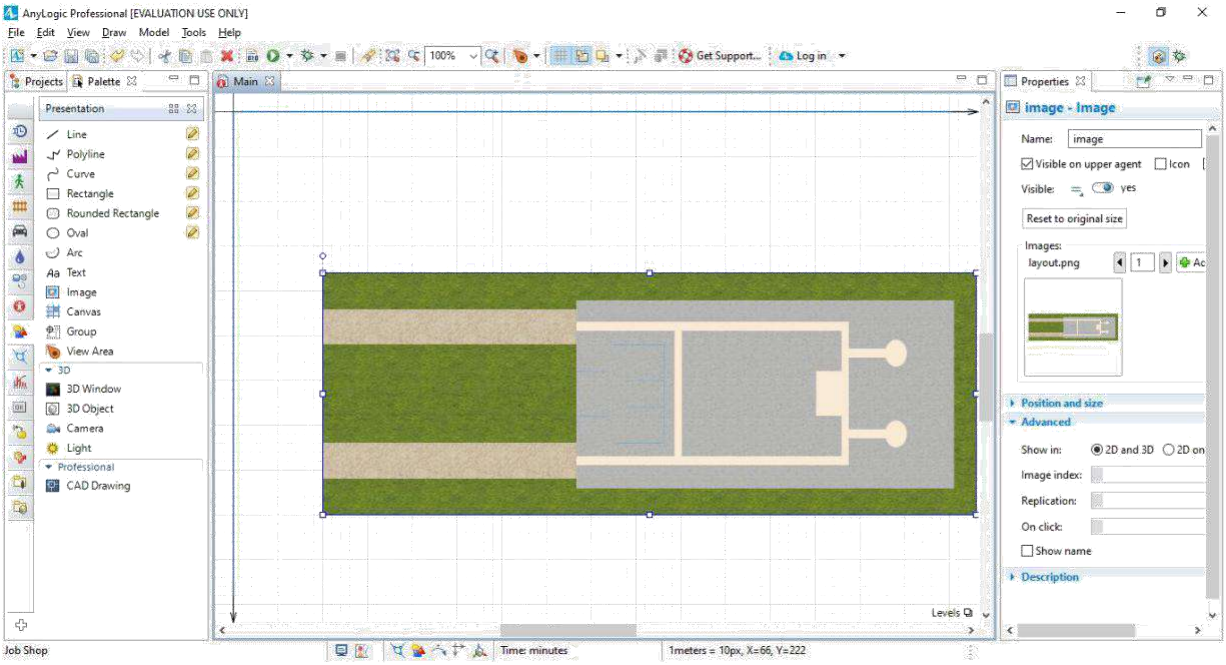
Set the Model name: Job Shop, and Model time



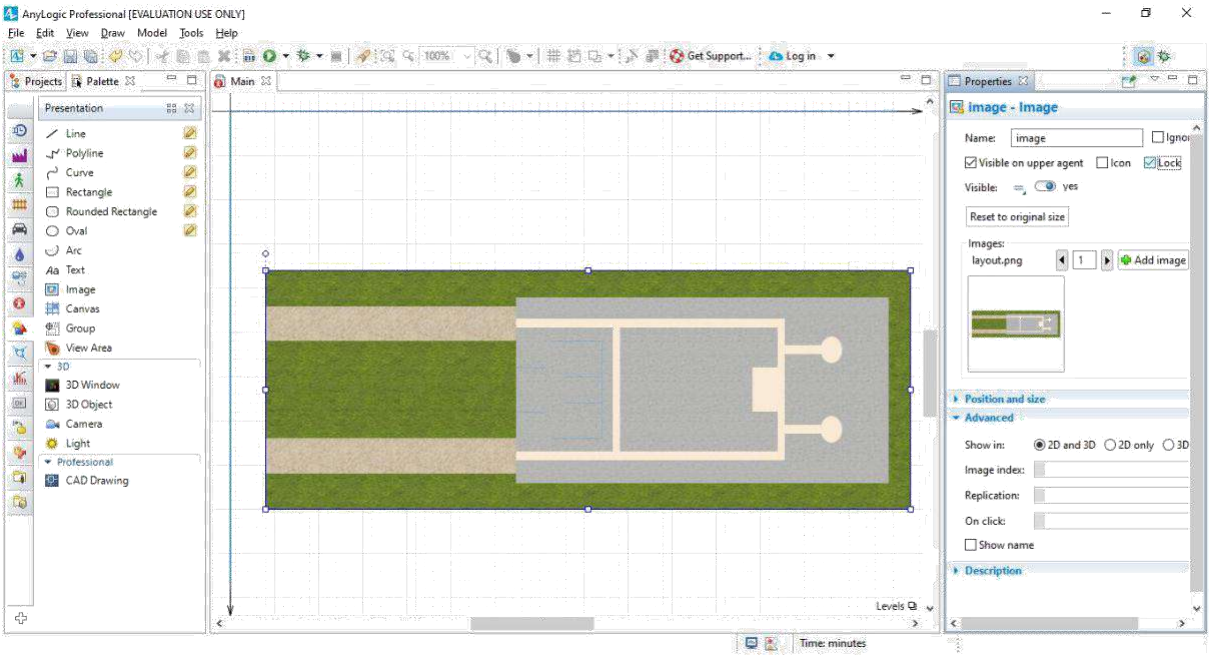
Units: minutes. On the Presentation palette, select the Image shape and then drag it onto the Main diagram. You can use the Image shape to add images in several graphic formats -- including PNG, JPEG, GIF, and BMP – to your presentation.



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: Any Logic folder/resources/Any Logic 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:

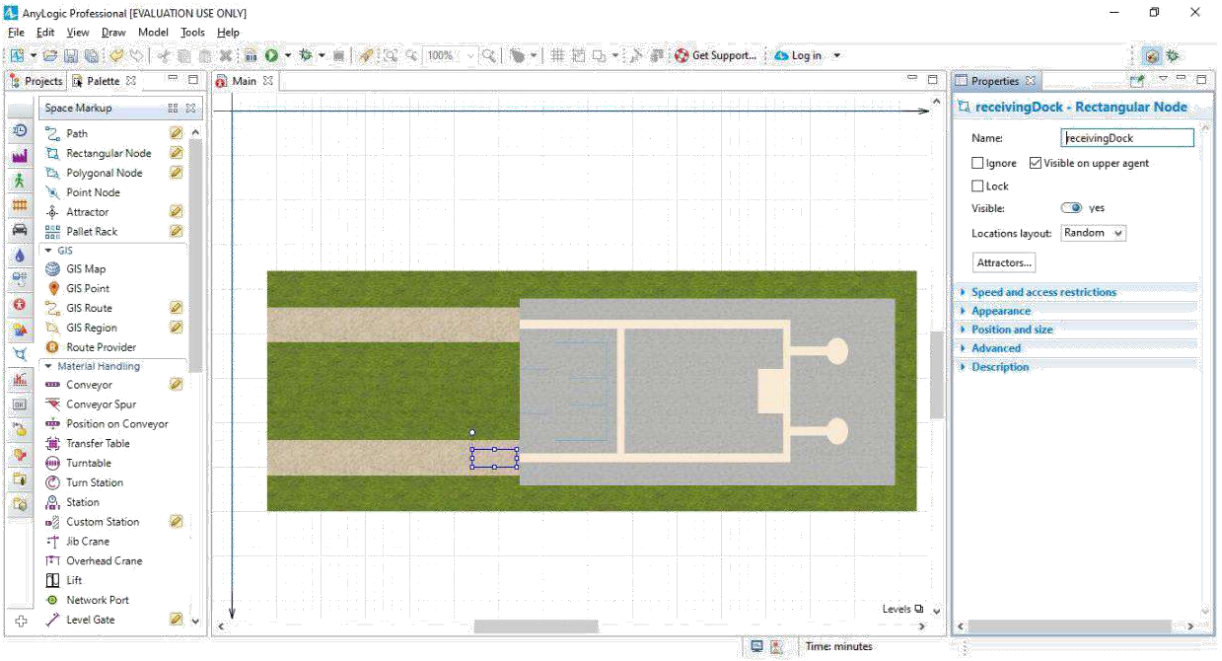


Select the image in the graphical editor. In the Properties view, select the Lock checkbox to lock the image.

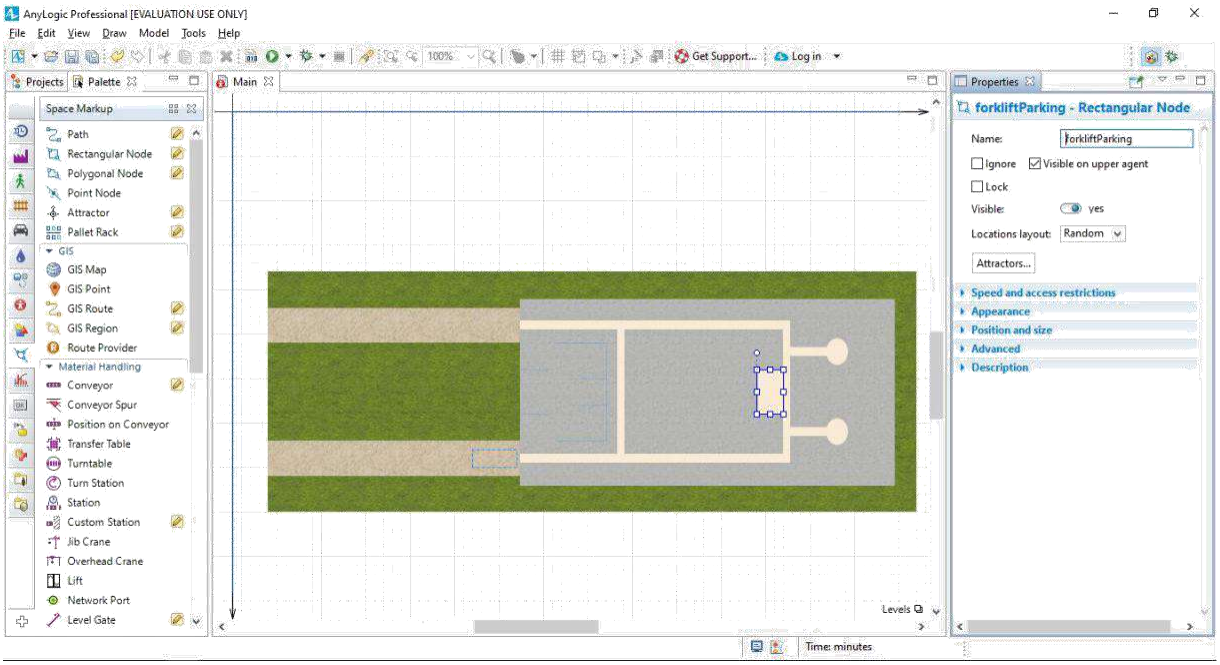


Open the Space Markup palette and drag the Rectangular Node element on to the Main diagram. Resize the node. The node should look as in the figure below.

Name the created node receivingDock.



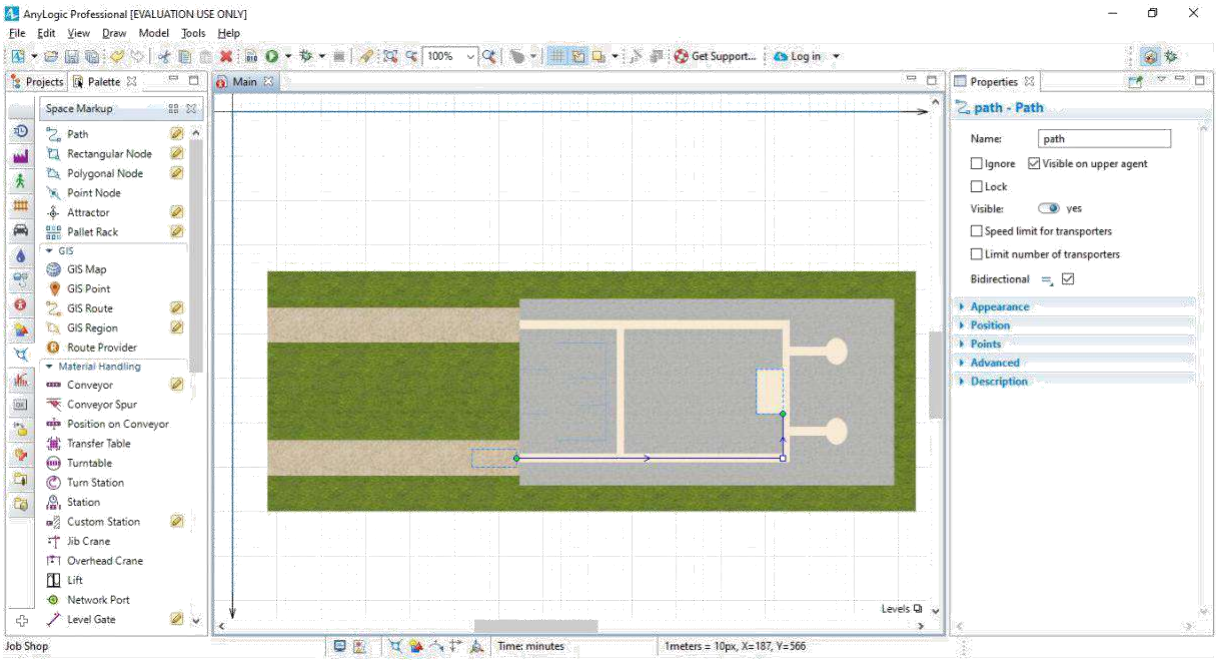
Draw a node to define the location where the model's agents will park forklift trucks once the trucks are idle or the agents no longer need them to complete a task. Use another Rectangular node to draw the parking area as shown in the figure below and then name this node forklift Parking.



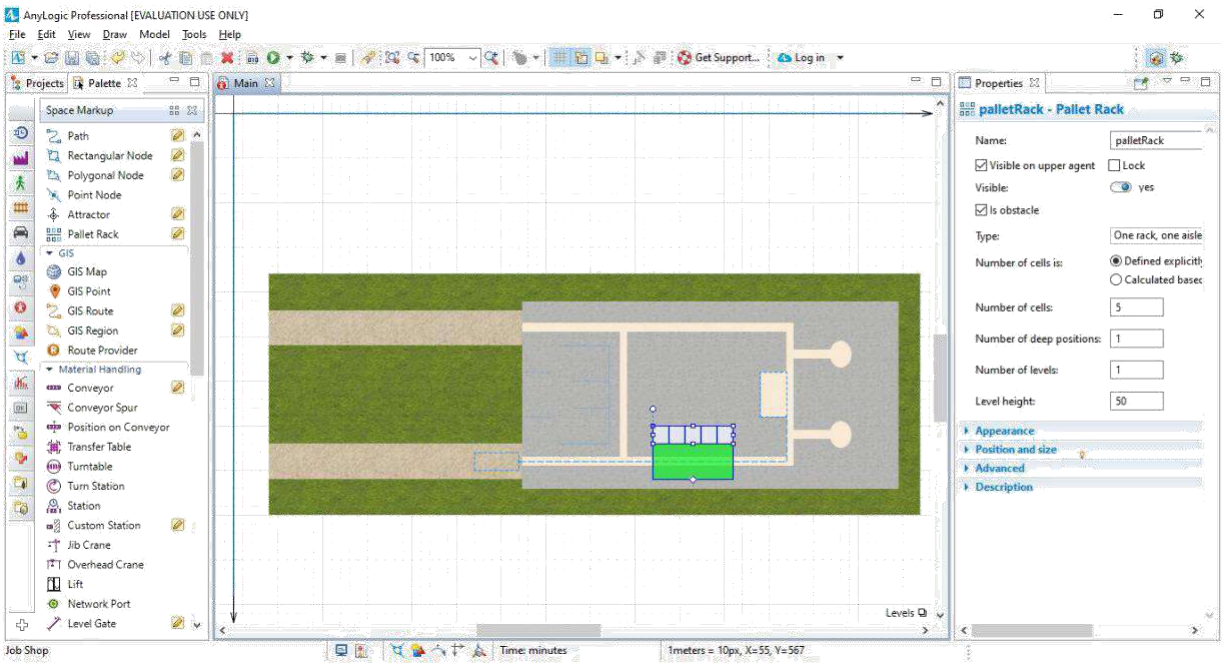
Do the following to draw a movement path that will guide our models forklift trucks?

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

b. Draw the path as shown in the figure below by clicking the receiving Dock border, clicking in the diagram to add the path's turning point, and then clicking the forklift parking 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:

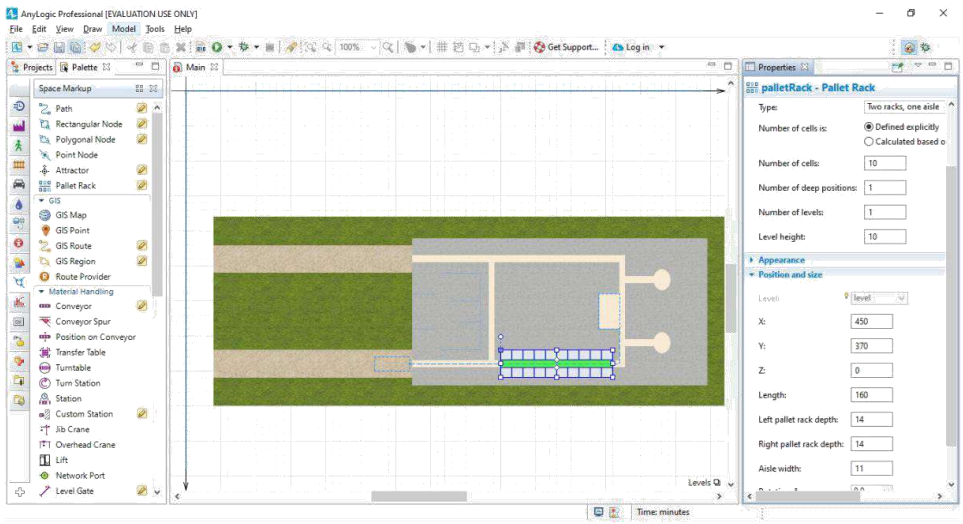
d. Length: 160

e. Left pallet rack depth: 14

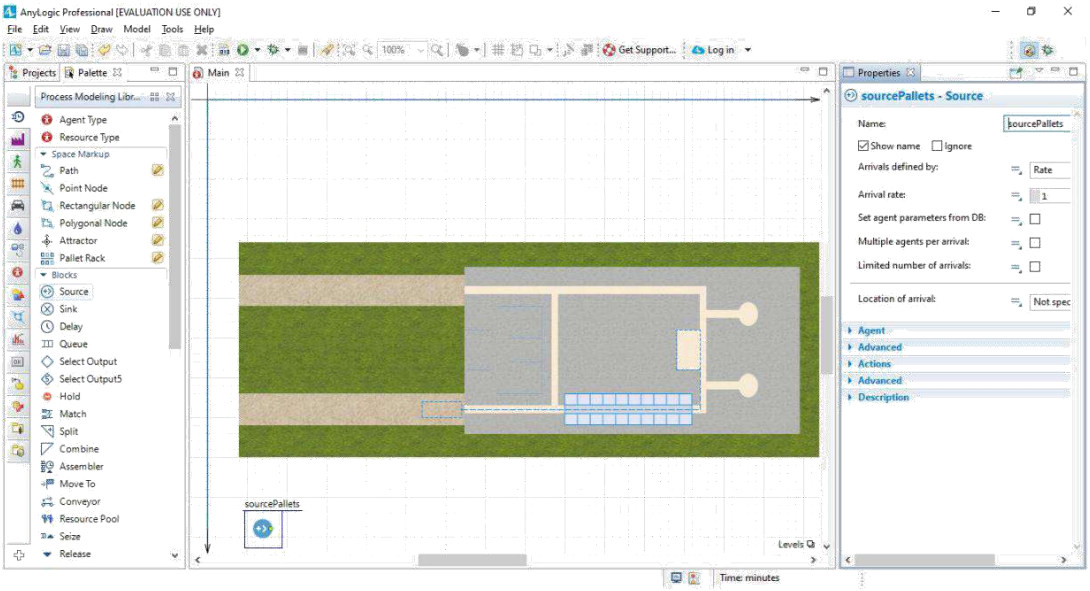
f. Right pallet rack depth: 14

g. Aisle width: 1113.

Move the pallet rack so that its centre 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.



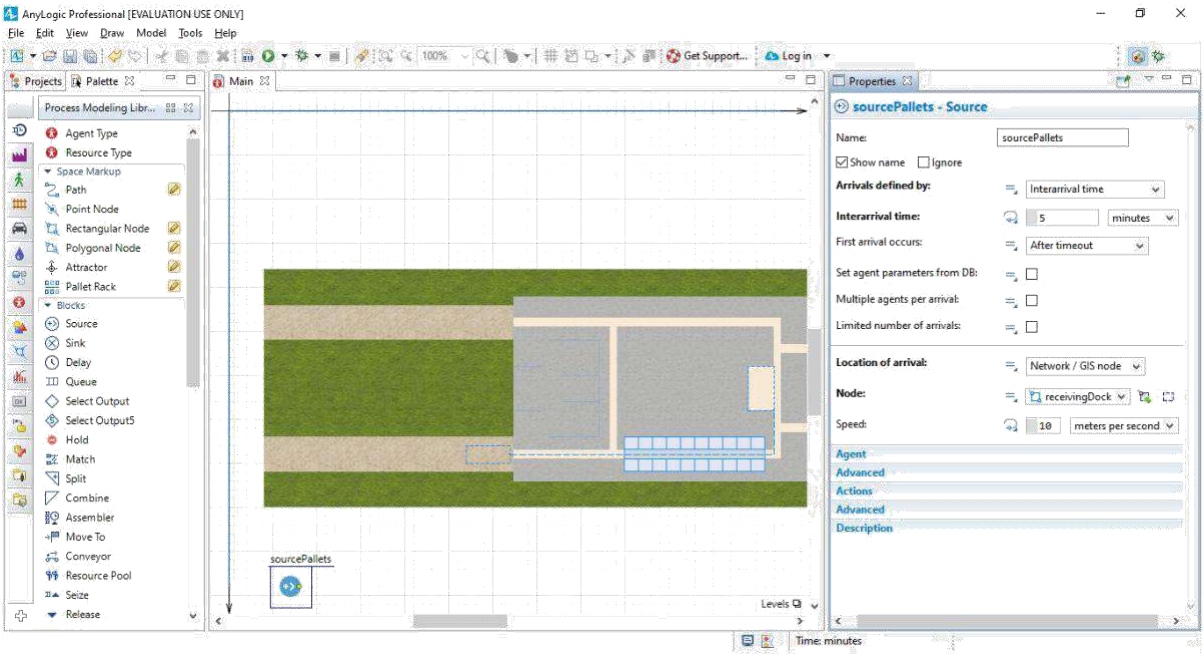
In the sourcePallets block's Properties area, do the following to ensure the model's pallets arrive every five minutes and appear in the receivingDock node.

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.

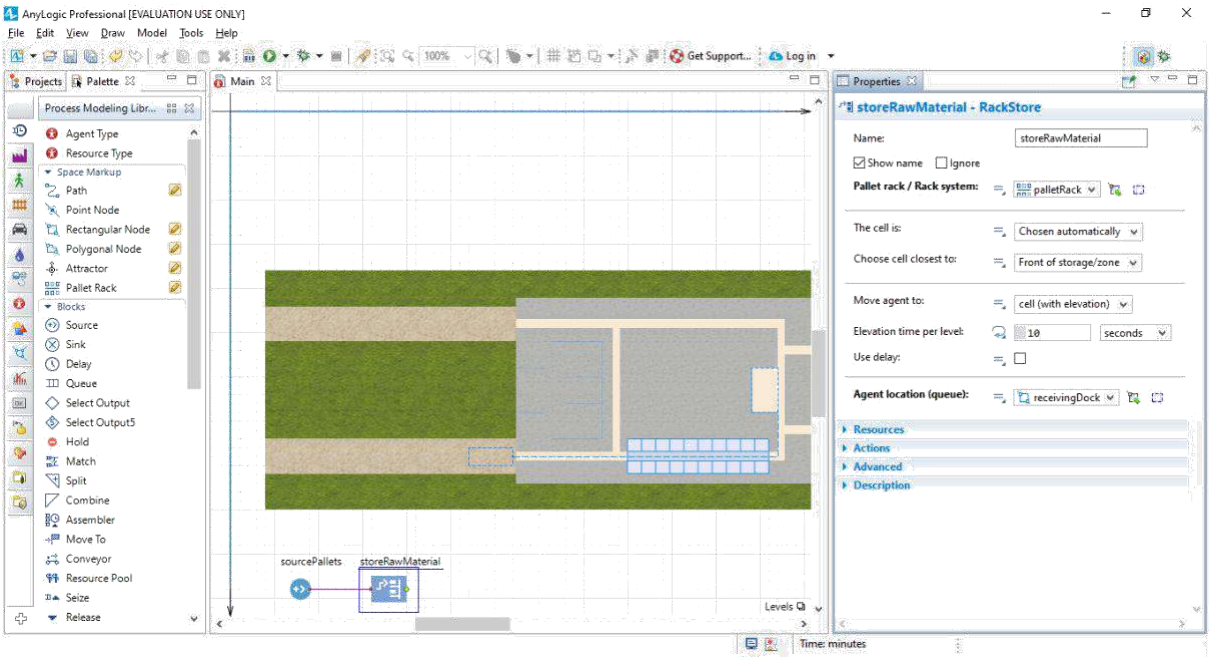
The RackStore block places pallets into a given pallet rack's cell.

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.

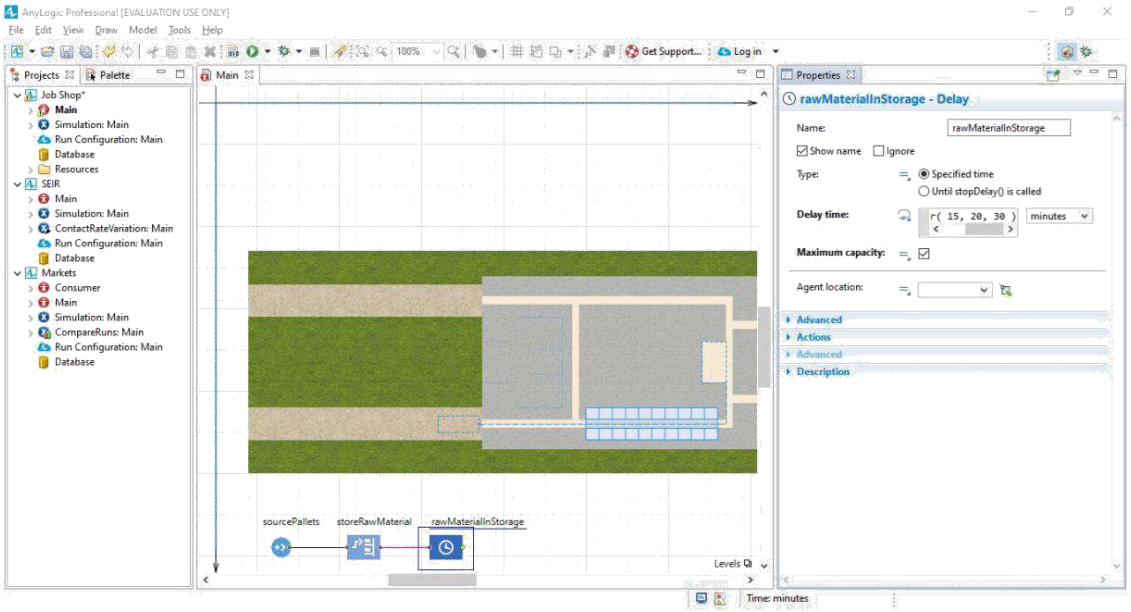


Add a Delay block to simulate how pallets wait in the rack and then name the block raw Material in Storage.

In the raw Material in Storage block's Properties area, do the following:

a. In the Delay time box, 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.

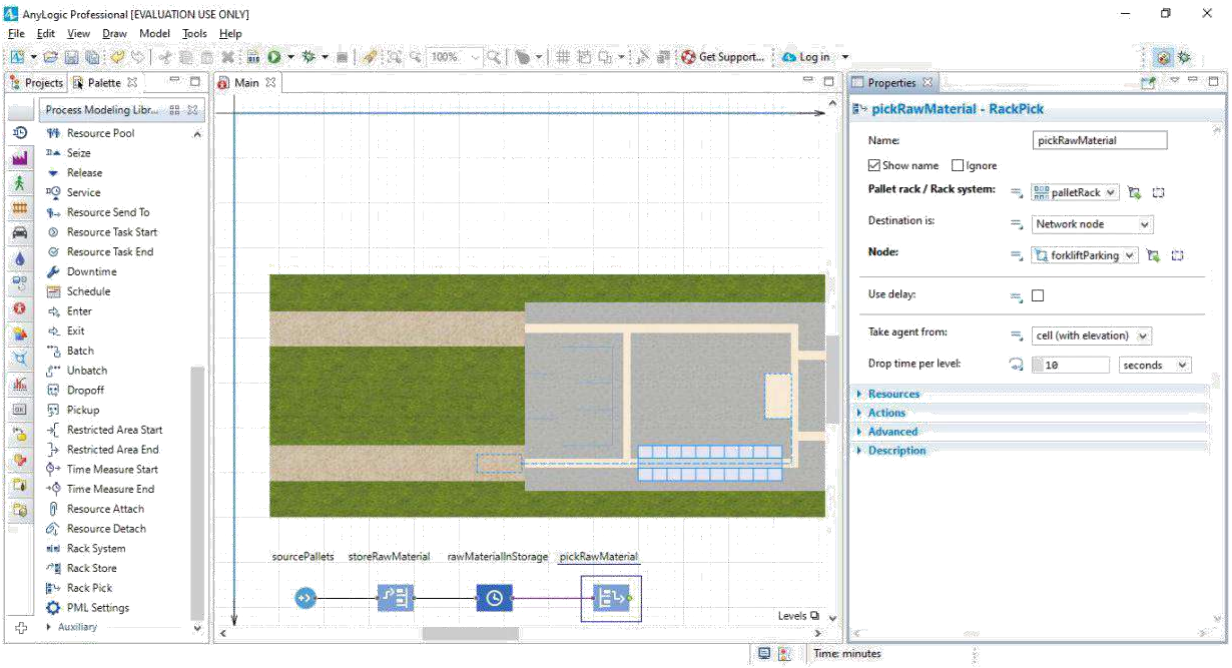


Add a RackPick block, connect it to the flowchart, and then name it pickRawMaterial.

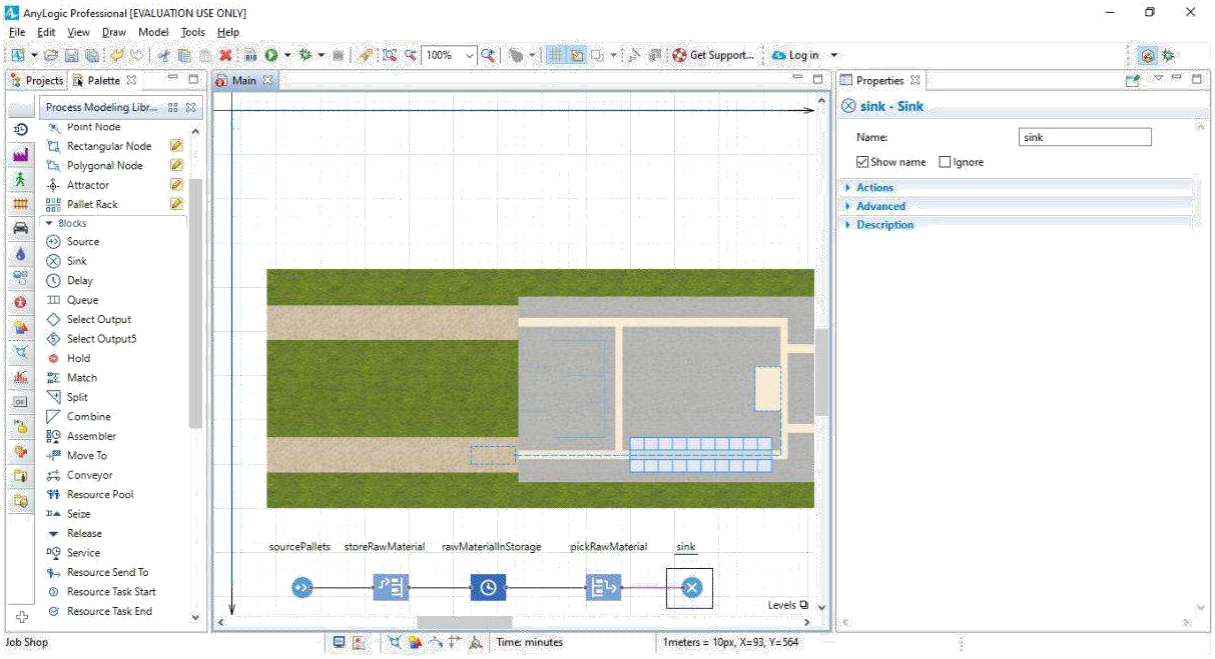
In the pickRawMaterial block's Properties area, do the following:

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 behaviour.

Run the model (Job Shop / Simulation experiment).

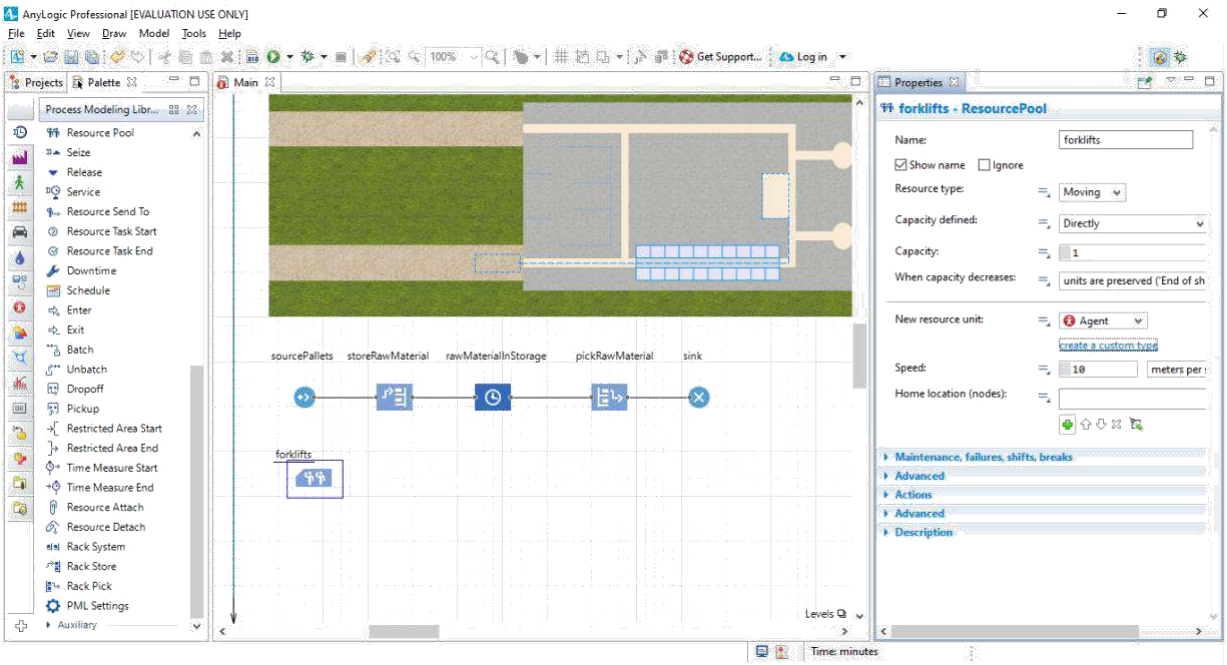
Adding Resources

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.

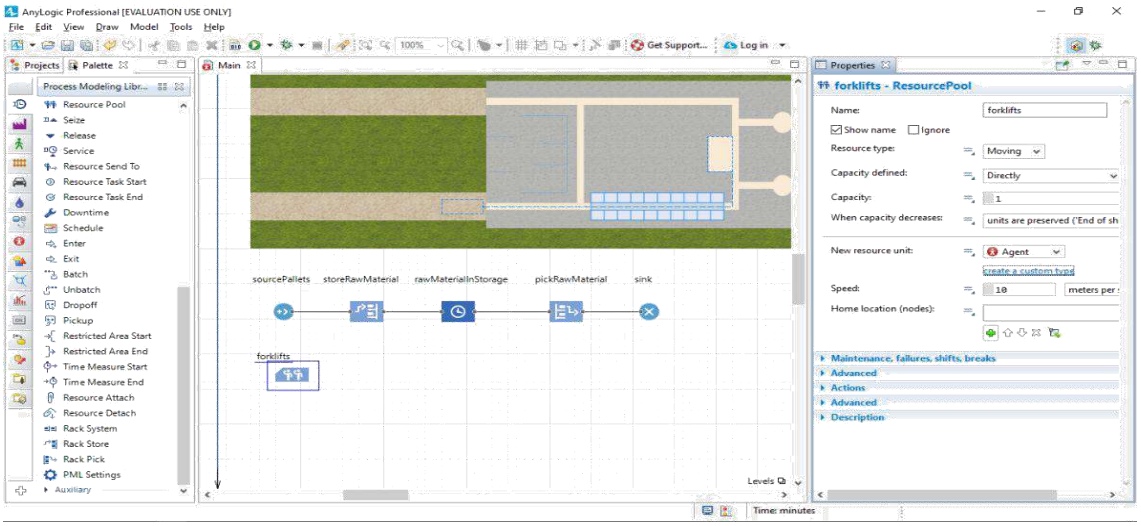
On the Process Modelling Library palette, drag the Resource Pool block on to the Main diagram.

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.



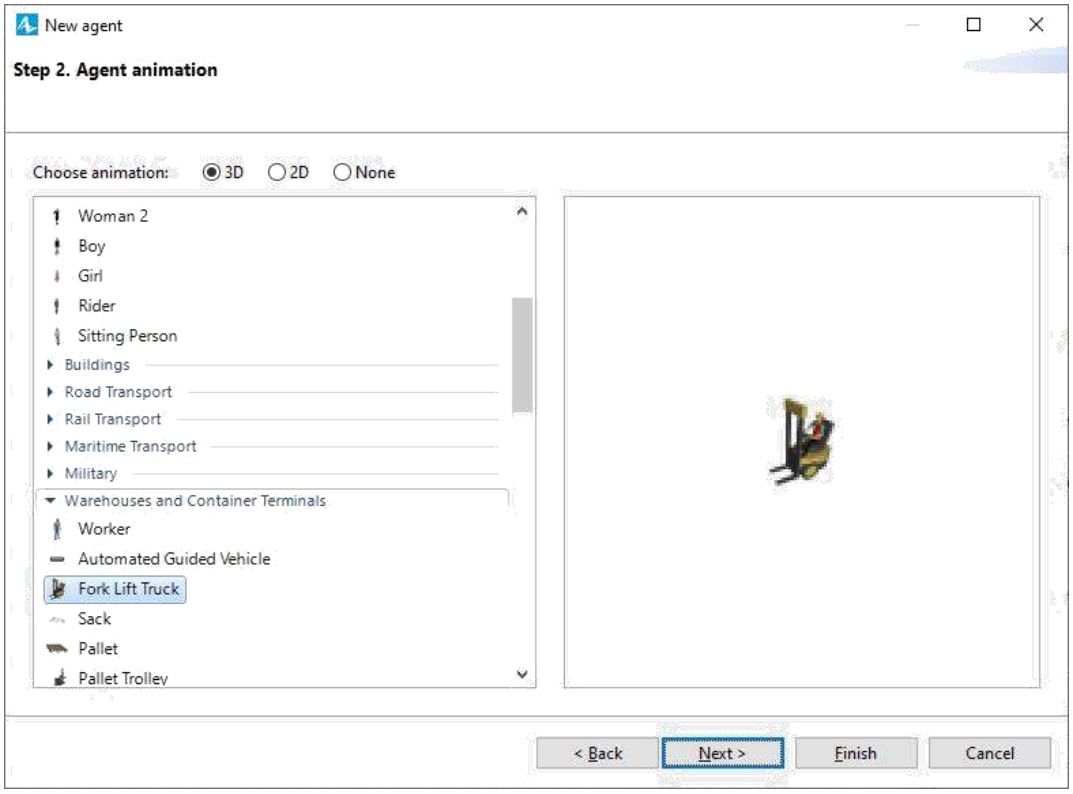
In the New agent wizard:

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

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 Forklift Truck.

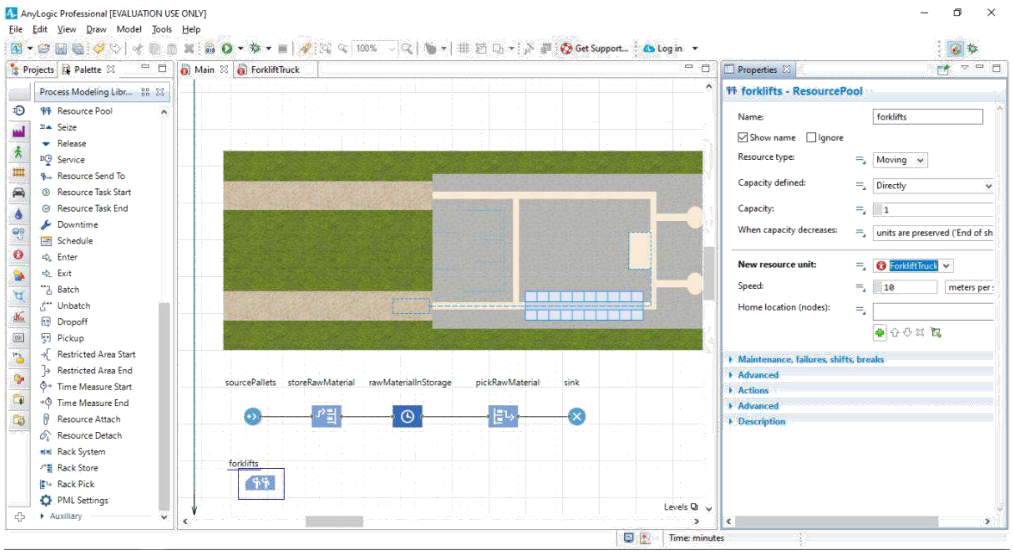
c. Click Finish.

The Forklift Truck 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 Forklift Truck resource type has been selected in the Resource Pool block's new resource unit parameter.

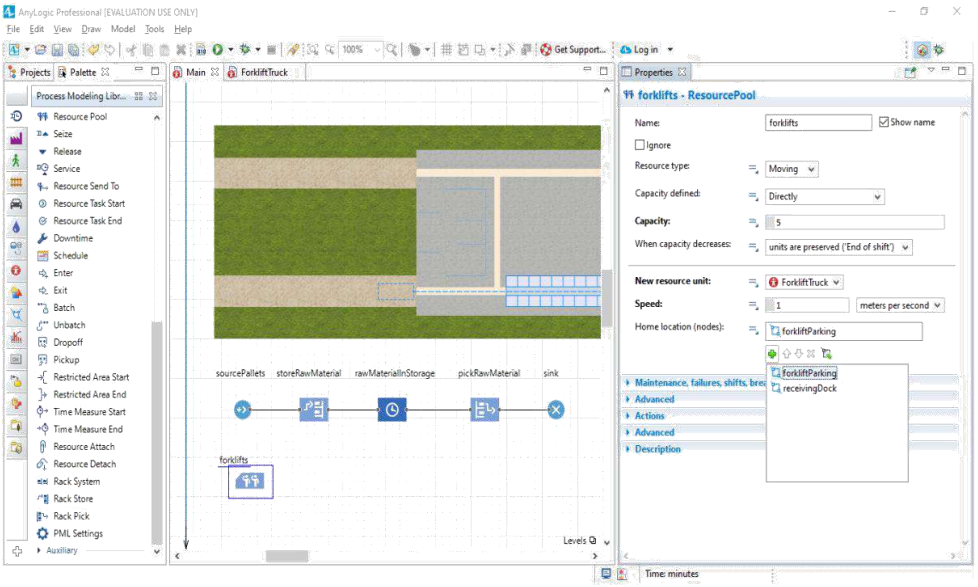


Modify the forklifts resource type's other parameters:

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.



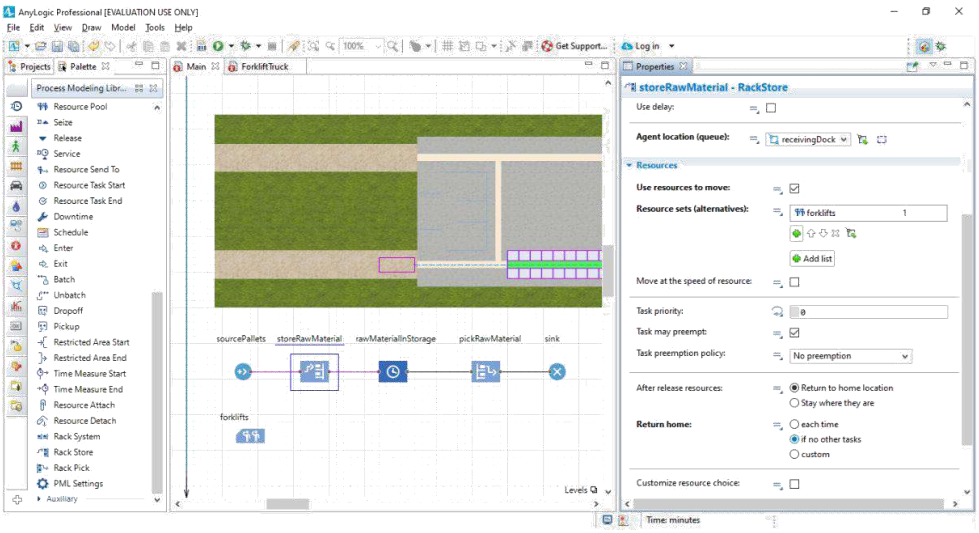
In the storeRawMaterial block's Properties area, do the following:

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.



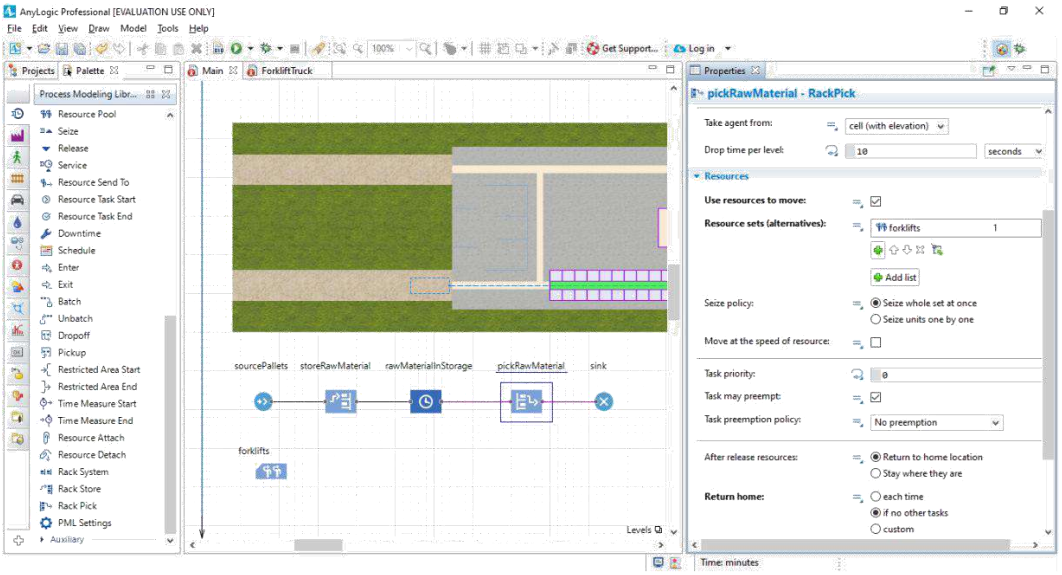
In the pickRawMaterial block's Properties area, do the following:

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.

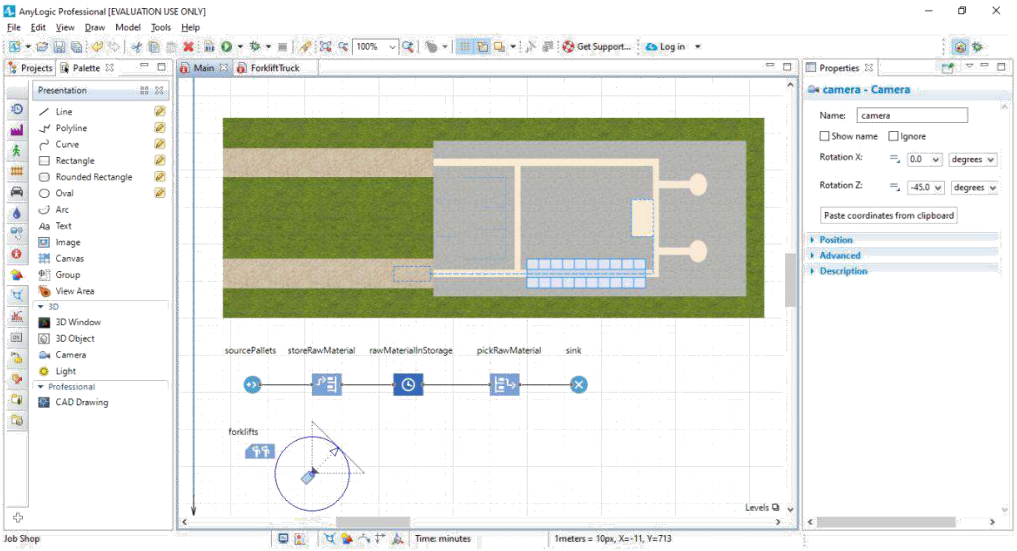
d. In the Return home area, click if no other tasks to ensure the forklift trucks return to their home location after they complete their tasks.



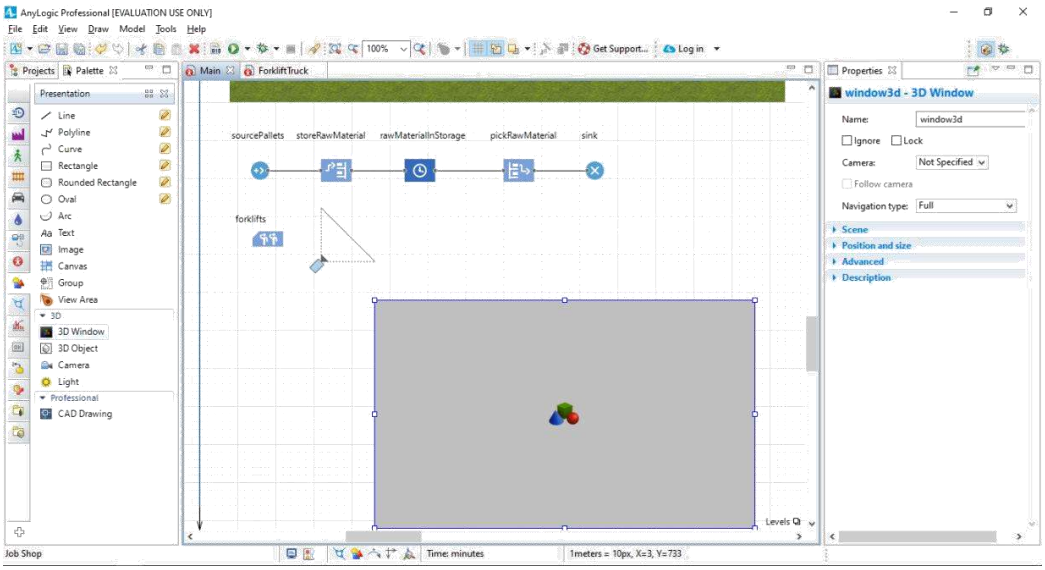
Run the model.

Creating 3D animation

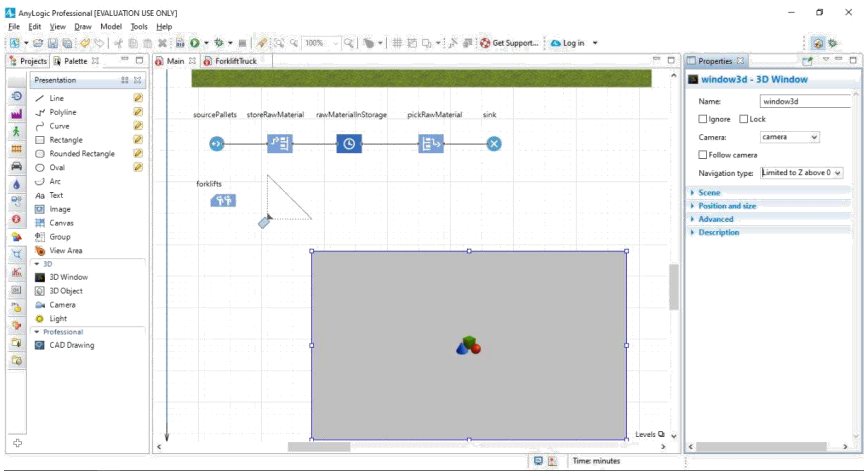
On the Presentation palette, drag the Camera object on to the Main diagram so it faces the job shop layout.



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].

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.

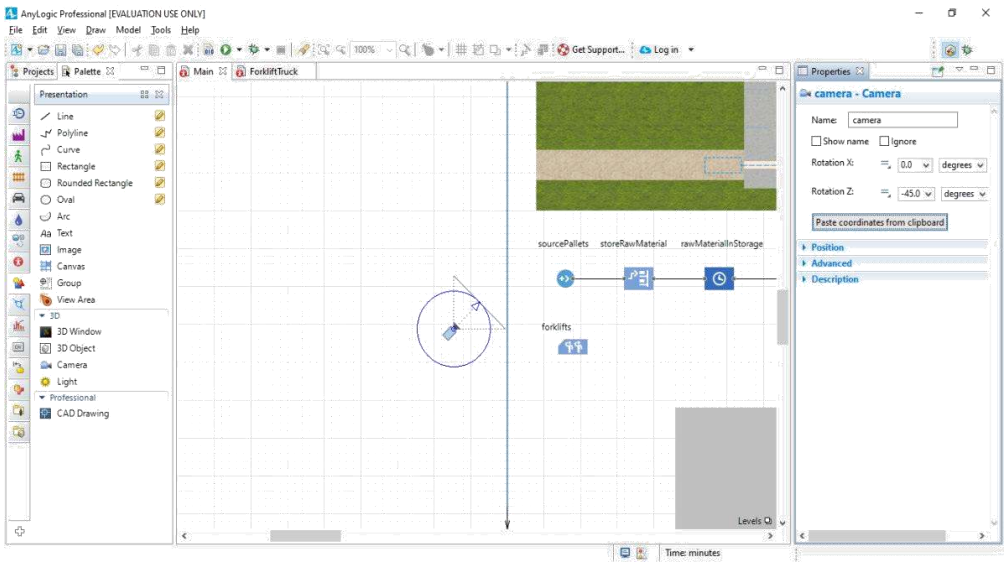
To move the camera closer to or further from the scene's centre, rotate the mouse's wheel.

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.

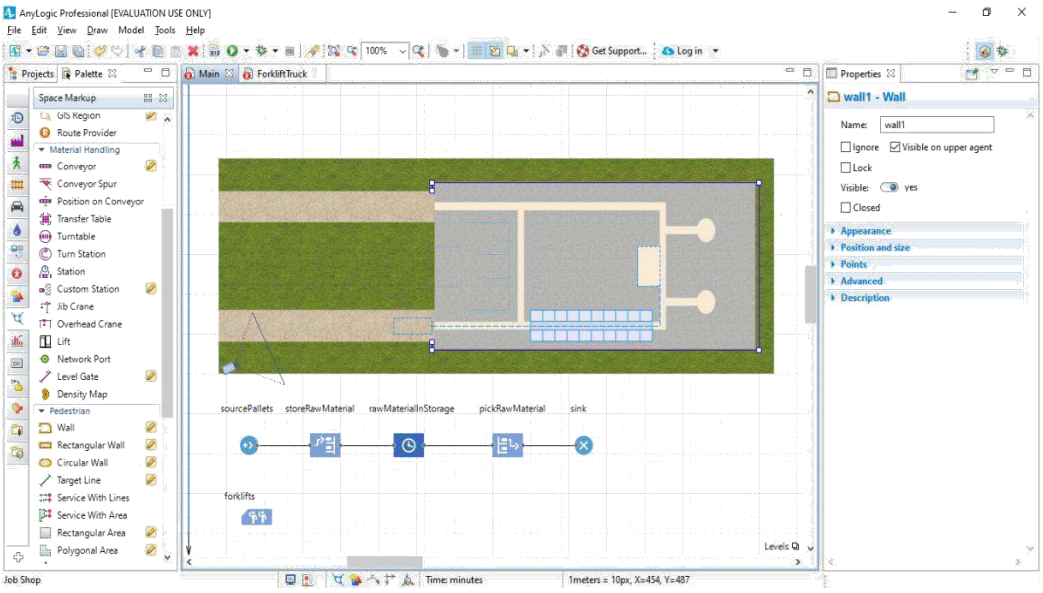
Expand the Space Mark-up palette's Pedestrian area and then double-click the Wall element's icon to enable Any Logic's drawing mode.

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.



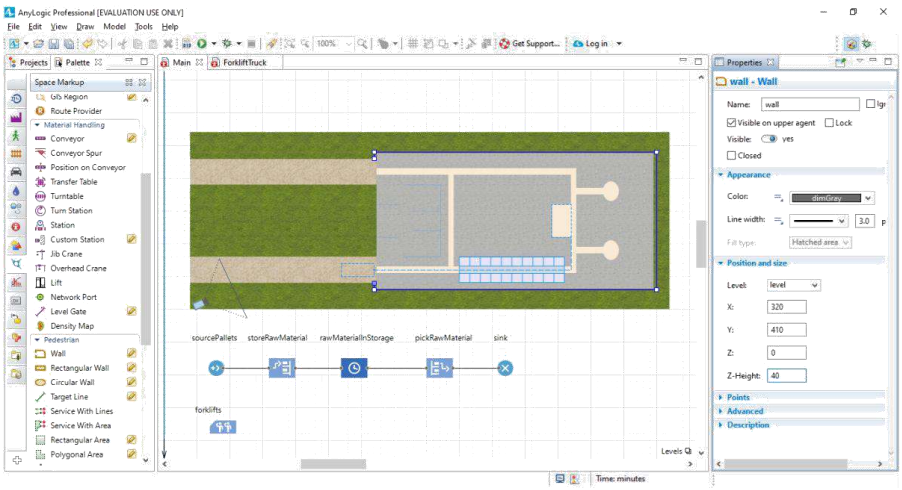
Do the following to change the wall's fill colour and texture:

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

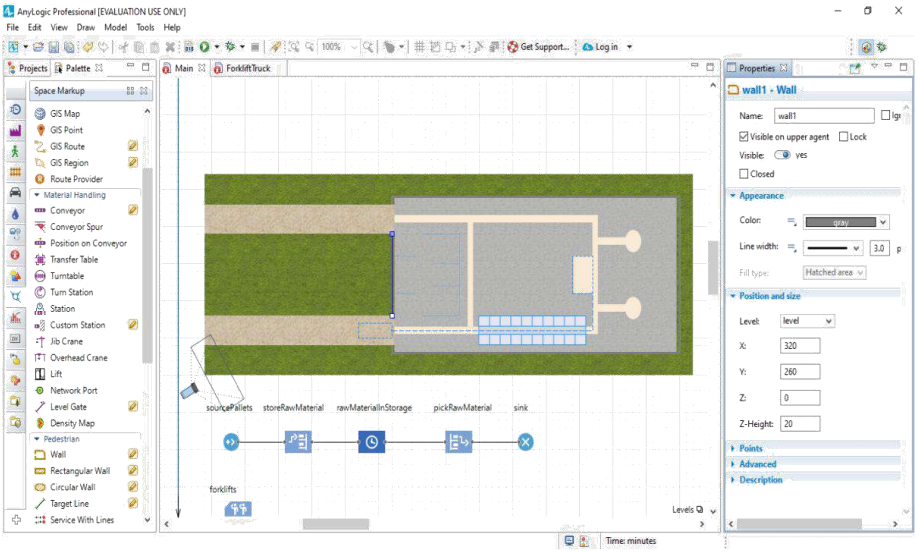
b. In the colour menu, click other colours.

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

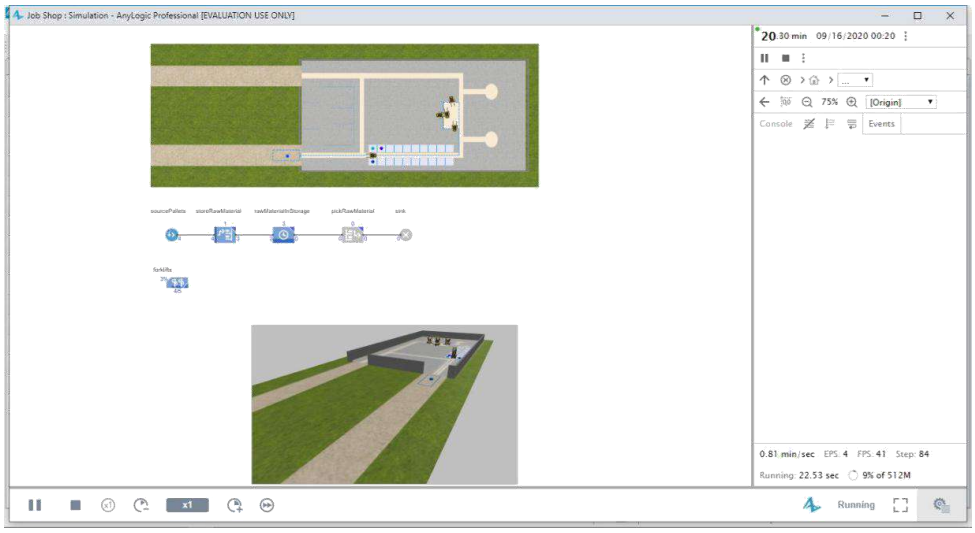
Go to the wall's Position and size section and change the Z-Height to 40.



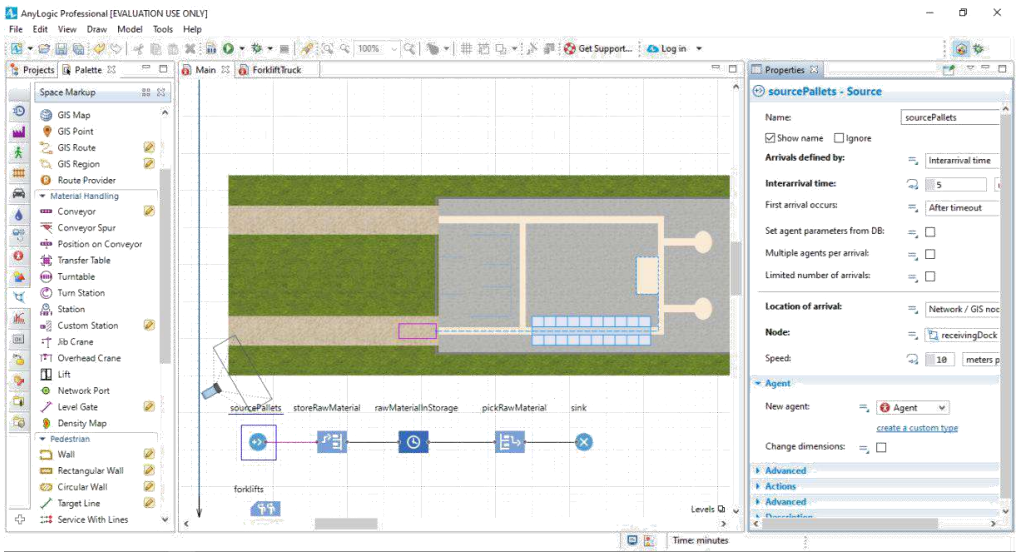
Draw another wall between the exits and then change the settings in the second wall's Properties area to match the first wall.



Run the model and view the 3D animation.



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

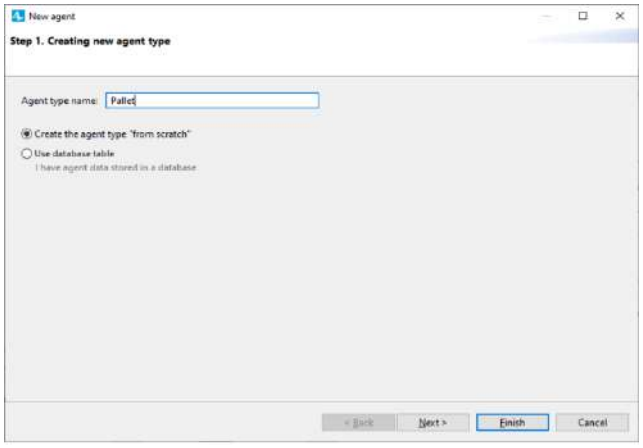


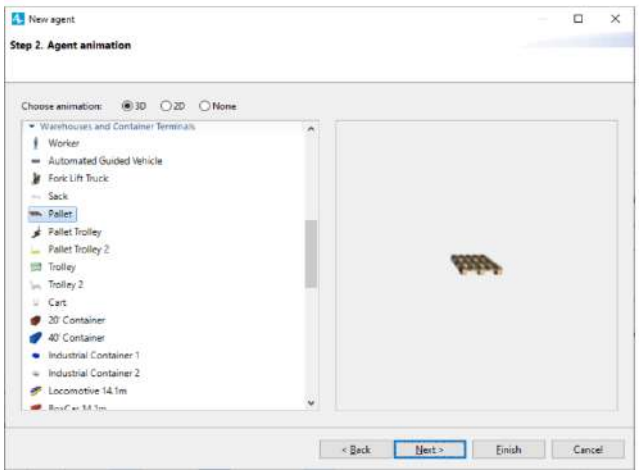
In the New agent wizard, do the following:

a. In the name of new type list, type Pallet.

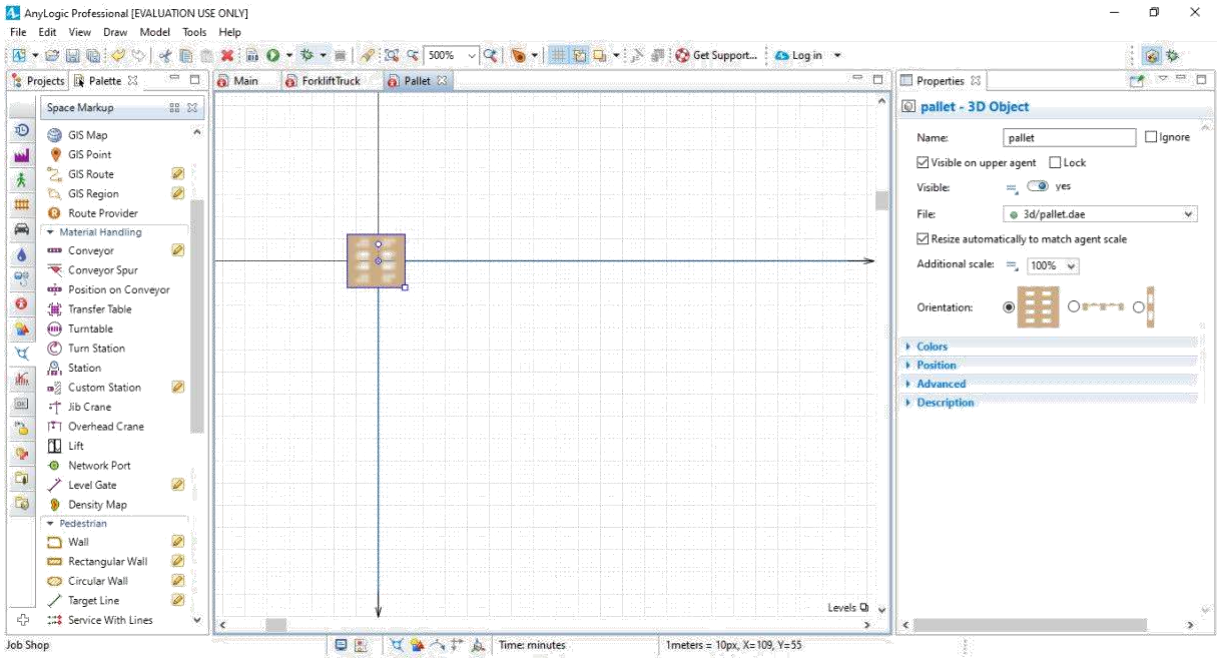
b. In the list in the left part of the wizard, expand the Warehouses and Container Terminals area, and then click the 3D animation figure Pallet.

c. Click Finish





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.

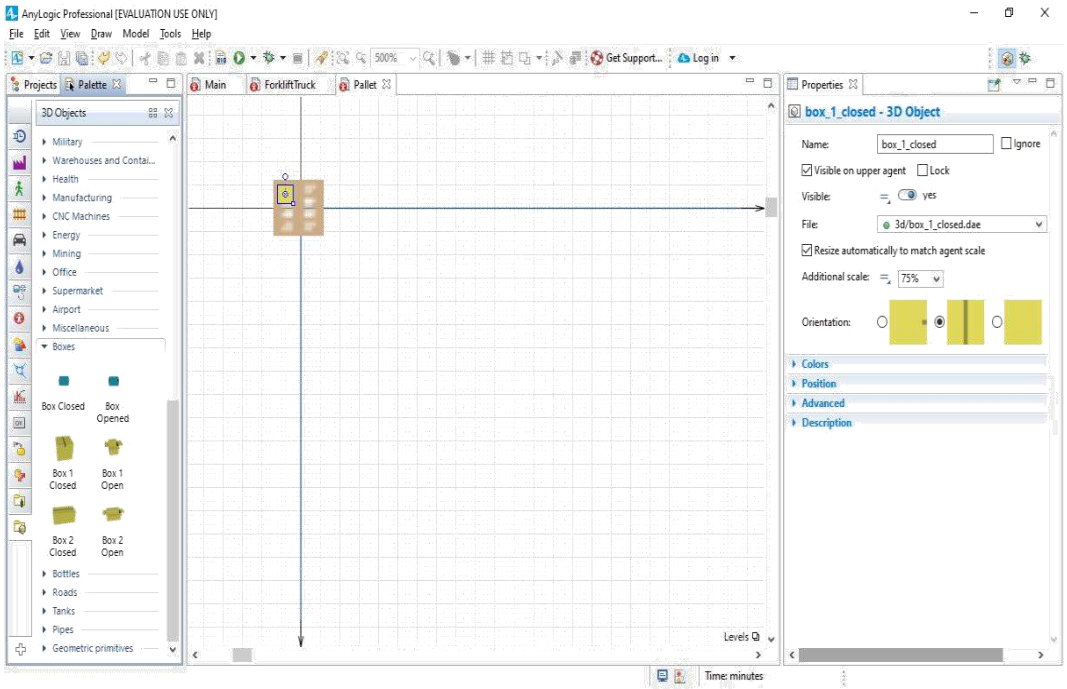


Do the following to start adding product animation on top of the pallet animation.

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

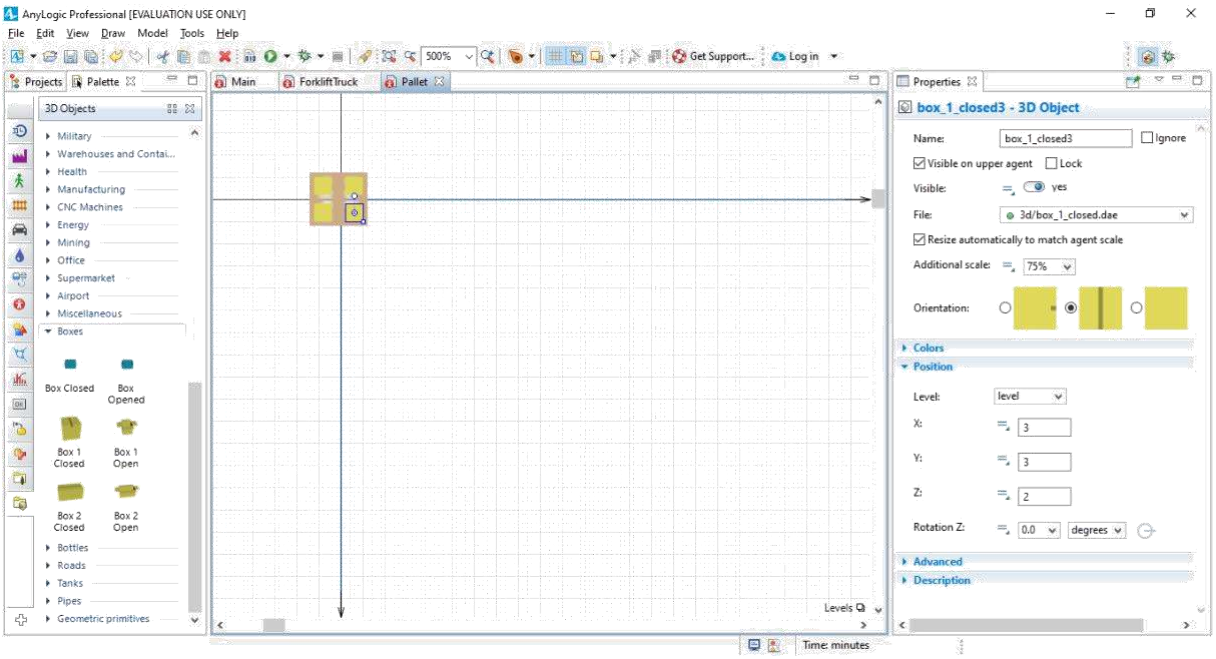
b. Drag the Box 1 Closed object on to the pallet's upper-left corner.

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.

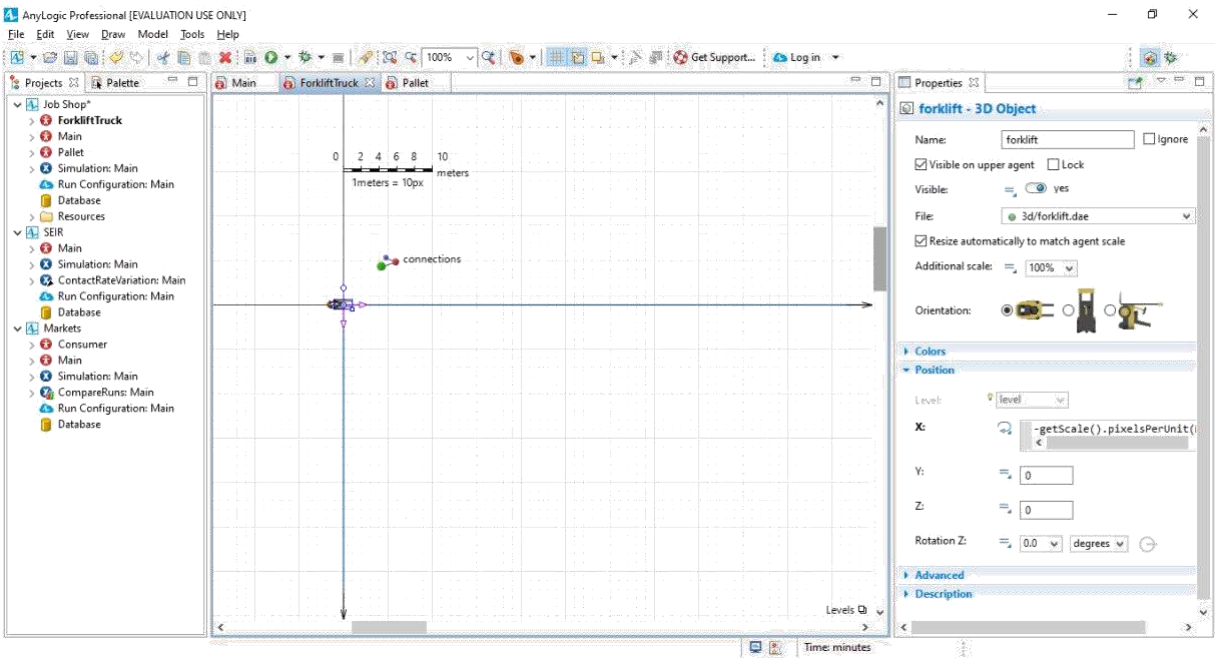
If you open the sourcePallets block's Properties area, you'll see Pallet is selected as new agent.

This block will generate agents of the Pallet type.

Run the model.

You'll see pallet shapes have replaced the multi-coloured 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.

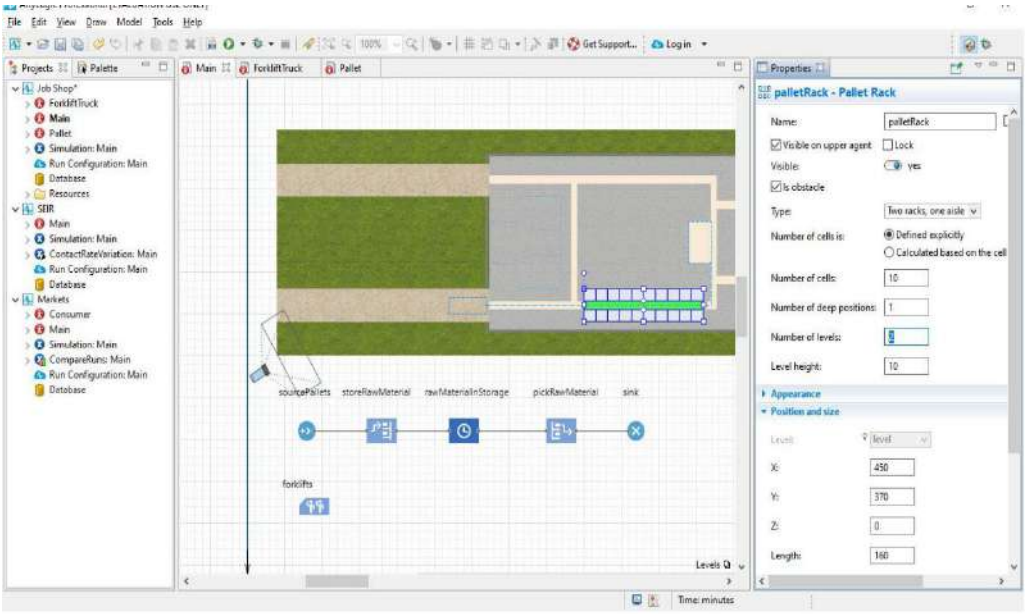
In the Projects view, double-click the Forklift Truck agent type to open its diagram and then move the forkliftWithWorker figure one cell to the right.

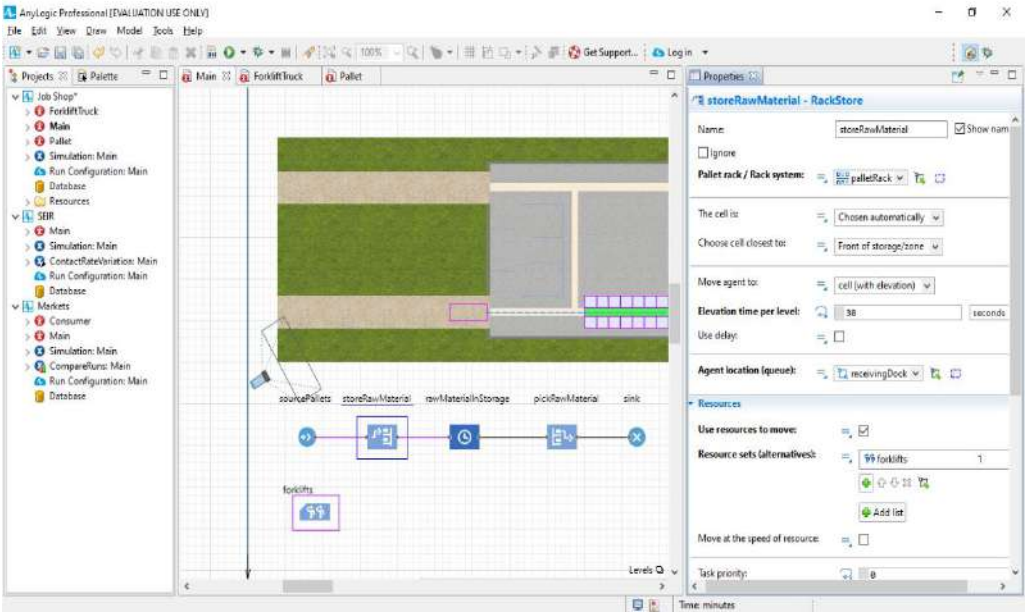


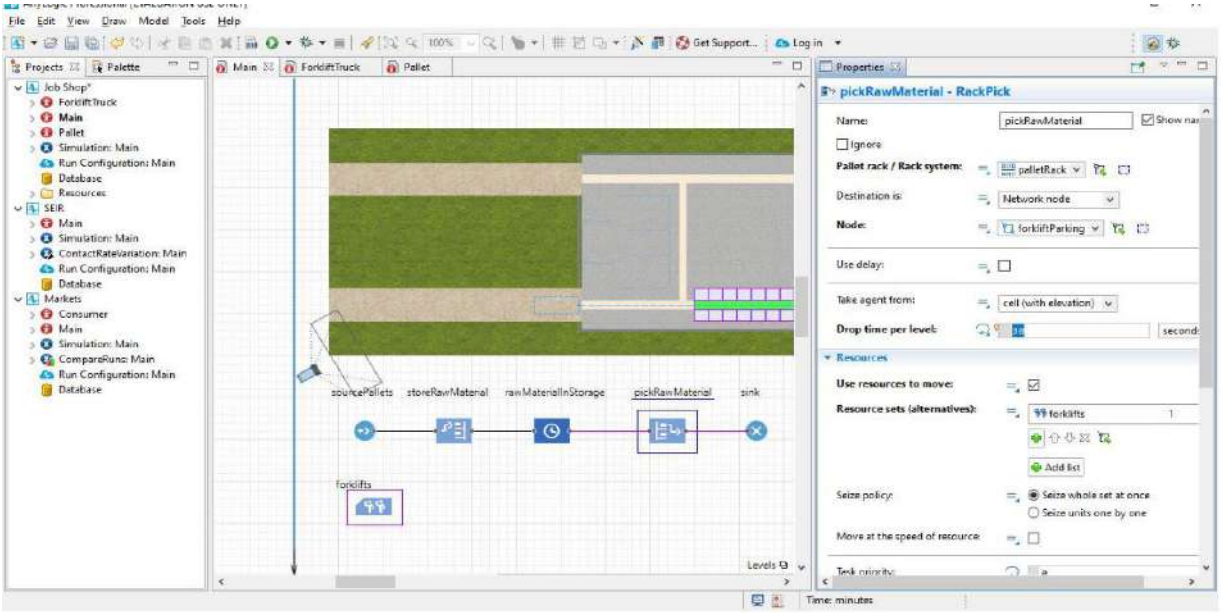
Open Main diagram, and in the pallet rack's Properties area, in the Number of levels box, type 2.

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

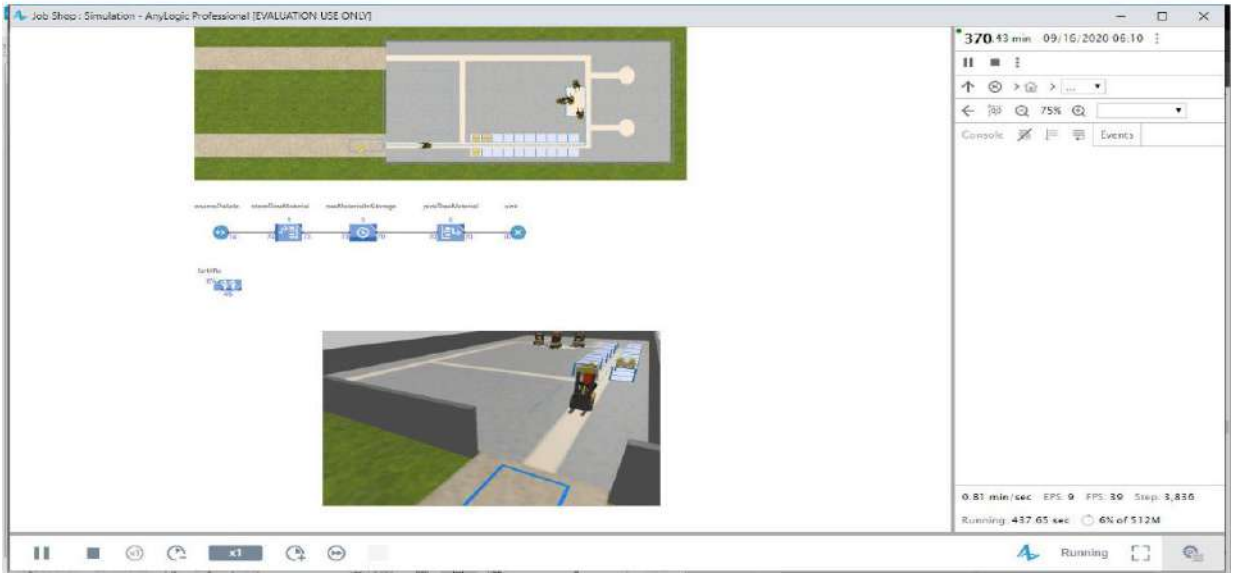
In the storeRawMaterial flowchart block's Properties area, set the Elevation time per level parameter to 30 seconds. In the pickRawMaterial block's Properties area, set the Drop time per level parameter to 30 seconds. Run the model and you'll see a pallet rack with two levels


File Edit View Draw Model Tools Help
Projects Palette
Main ForkliftTruck Pallet
Properties
palletRack - Pallet Rack
Name: palletRack
Visible on upper agent
Lock
Visible: yes
Is obstacle
Type: Two racks, one aisle
Number of cells is: Defined explicitly
Calculated based on the cell
Number of cells: 10
Number of deep positions: 1
Number of levels: 5
Level height: 10
Appearance
Position and size
Level: level
X: 450
Y: 370
Z: 0
Length: 160
sourcePallets storeRawMaterial rawMaterialInStorage pickRawMaterial sink
forklifts
Levels
Time: minutes


AnyLogic Professional [EVALUATION USE ONLY]
File Edit View Draw Model Tools Help
Projects Palette
Main ForkliftTruck Pallet
Properties
storeRawMaterial - RackStore
Name: storeRawMaterial
Ignore
Pallet rack / Rack system: palletRack
The cell is: Chosen automatically
Choose cell closest to: Front of storage zone
Move agent to: cell (with elevation)
Elevation time per level: 38 seconds
Use delay:
Agent location (queue): receivingDock
Resources
Use resources to move:
Resource sets (alternatives): forklifts 1
Add list
Move at the speed of resource:
Task priority: 0
sourcePallets storeRawMaterial rawMaterialInStorage pickRawMaterial sink
forklifts
Levels
Time: minutes


File Edit View Draw Model Tools Help
Projects
Palette
Main
ForkliftTruck
Pallet
Properties
pickRawMaterial - RackPick
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
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:
second
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:
Task priority:
Levels
Time: minutes


Job Shop : Simulation - AnyLogic Professional [EVALUATION USE ONLY]
25.69 min 09/16/2020 00:25
75%
Console
Events
0.82 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.

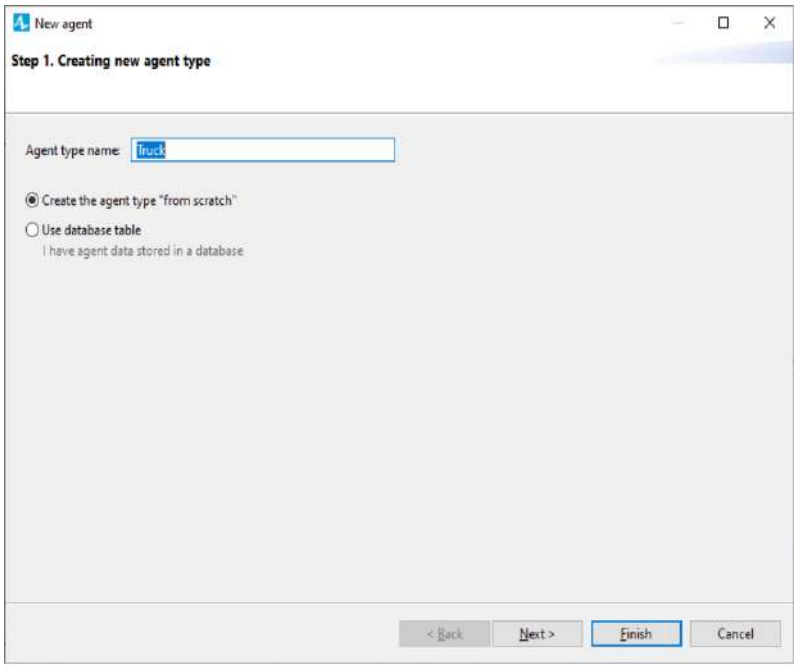
On the Process Modelling Library palette, drag the Agent type element on to the Main diagram.

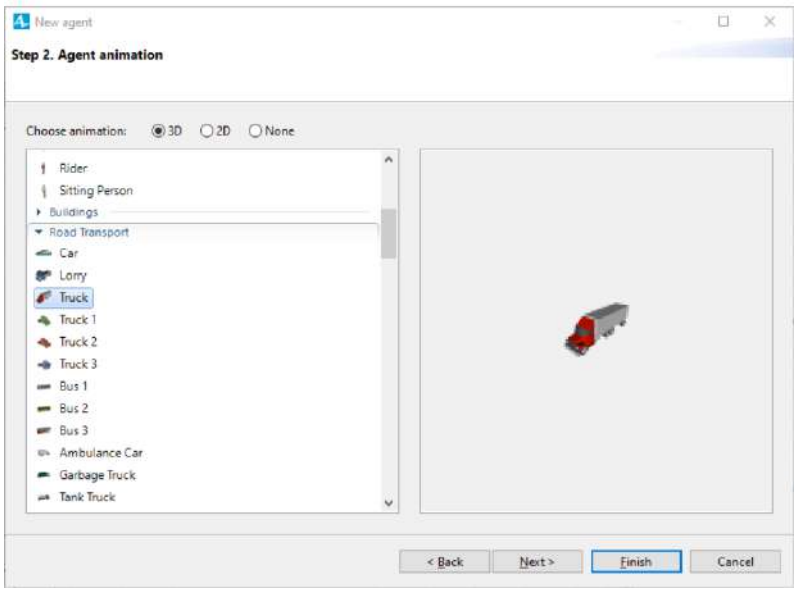
On the New agent wizard's Agent animation screen, do the following:

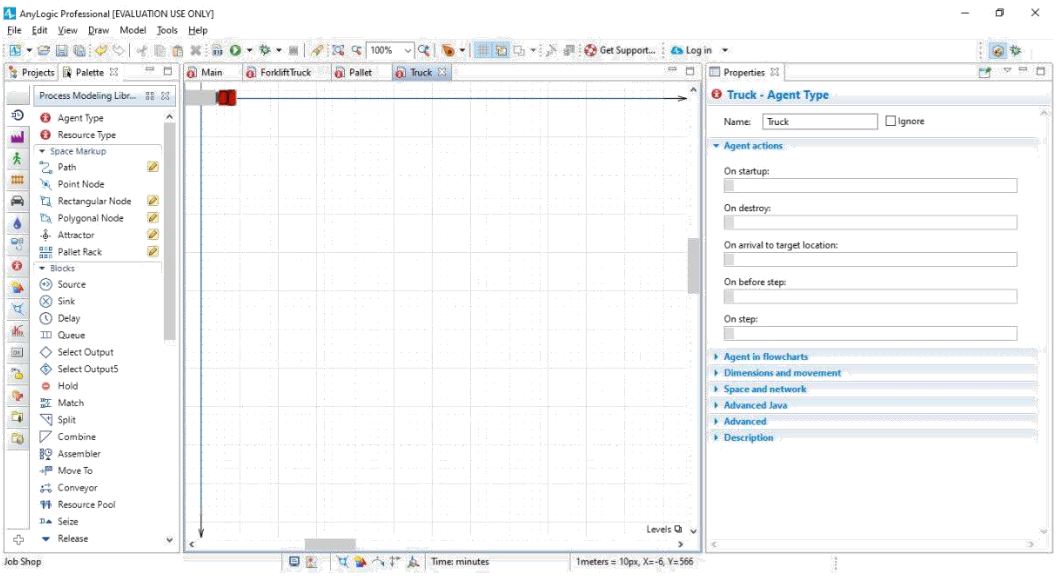
a. In the the name of new type box, type Truck.

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

c. Click Finish.







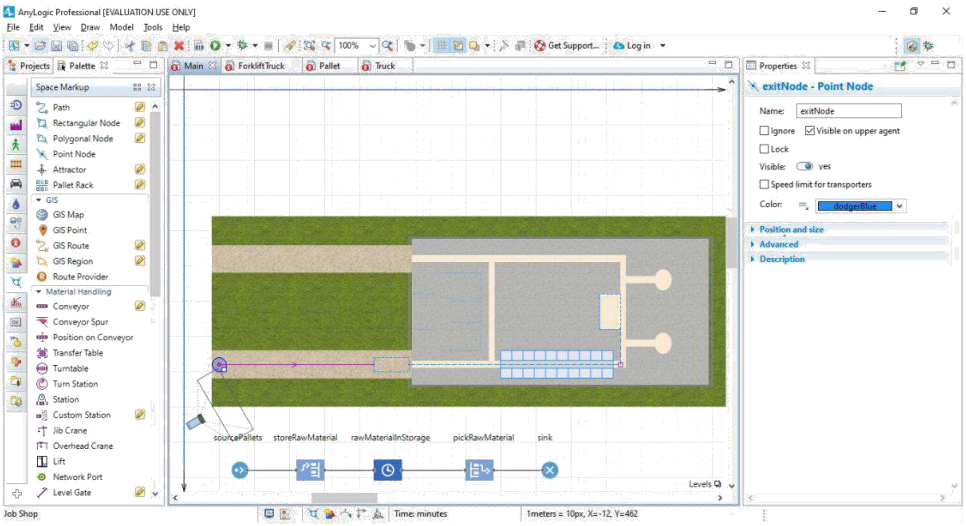
Let's add two more elements to our network: a node where the trucks will appear and the path that they will follow to the receiving dock.

Open the Main diagram,

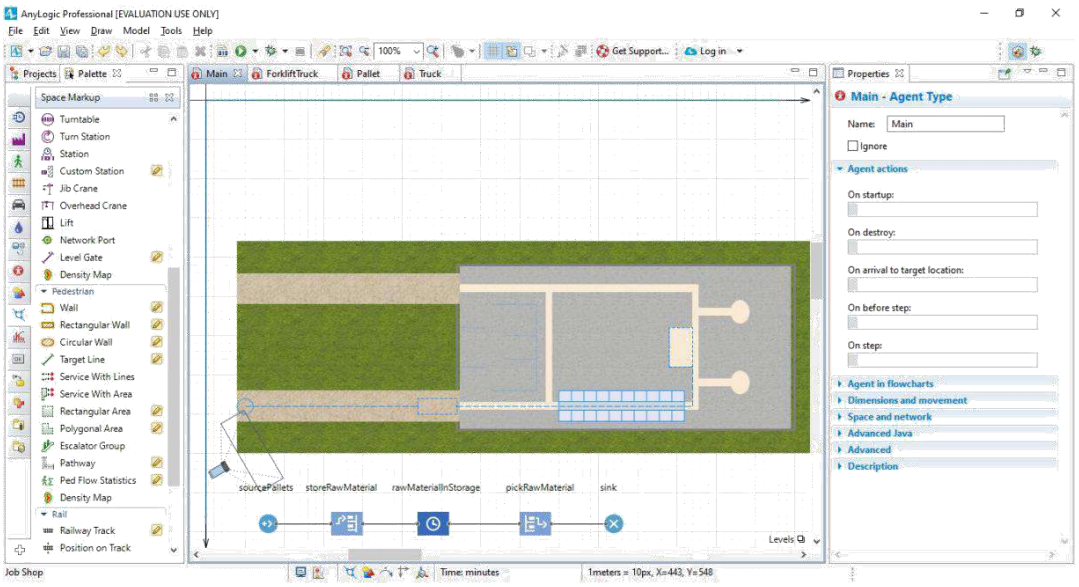
Under the Space Markup palette, click the Point Node element and drag it on to the driveway entry.

Name the node exit Node

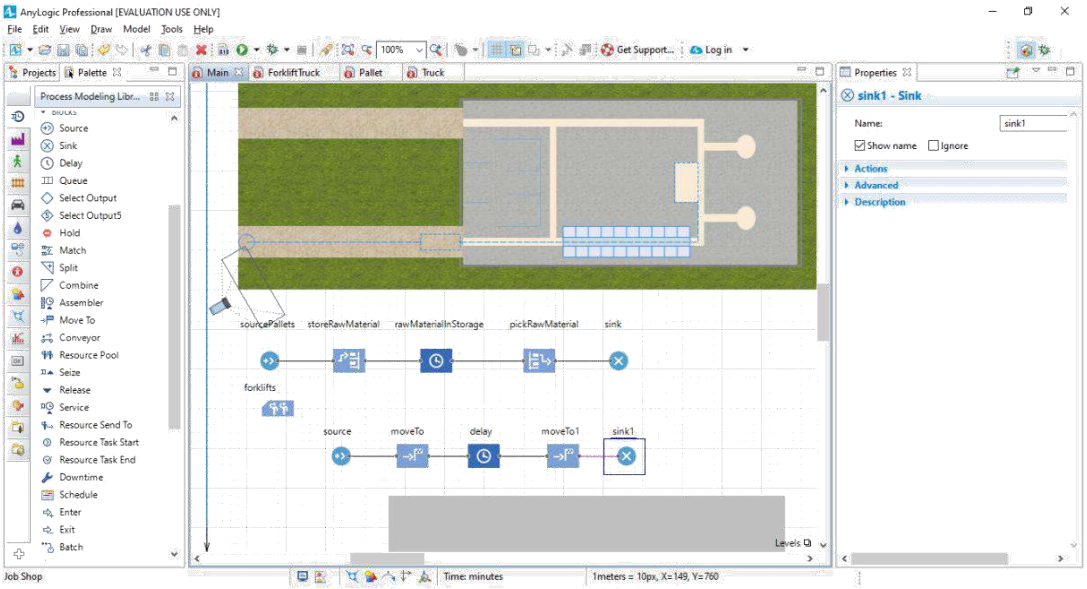
Draw a Path to connect the exitNode to the receivingDock.



Make sure all space markup elements connect to one network.



Create another process flowchart to define the truck movement logic by connecting the Process Modelling 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.

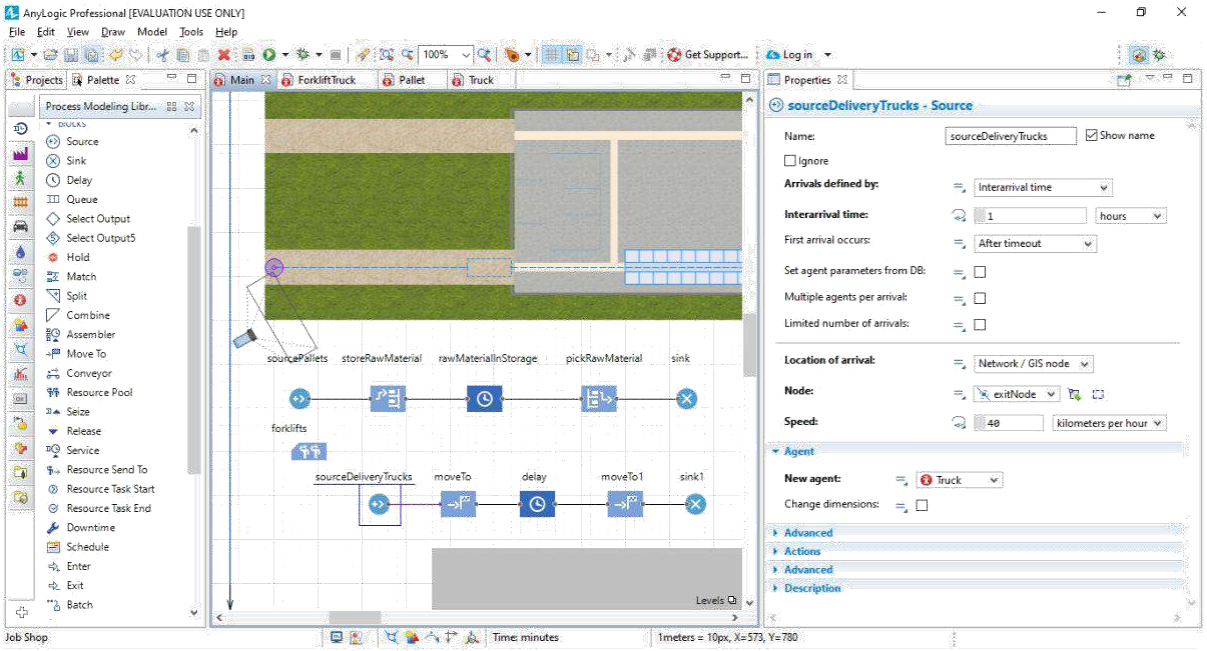
b. In the Interarrival time box, type 1, and select hours from the list on the right.

c. In the New agent list, click Truck.

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

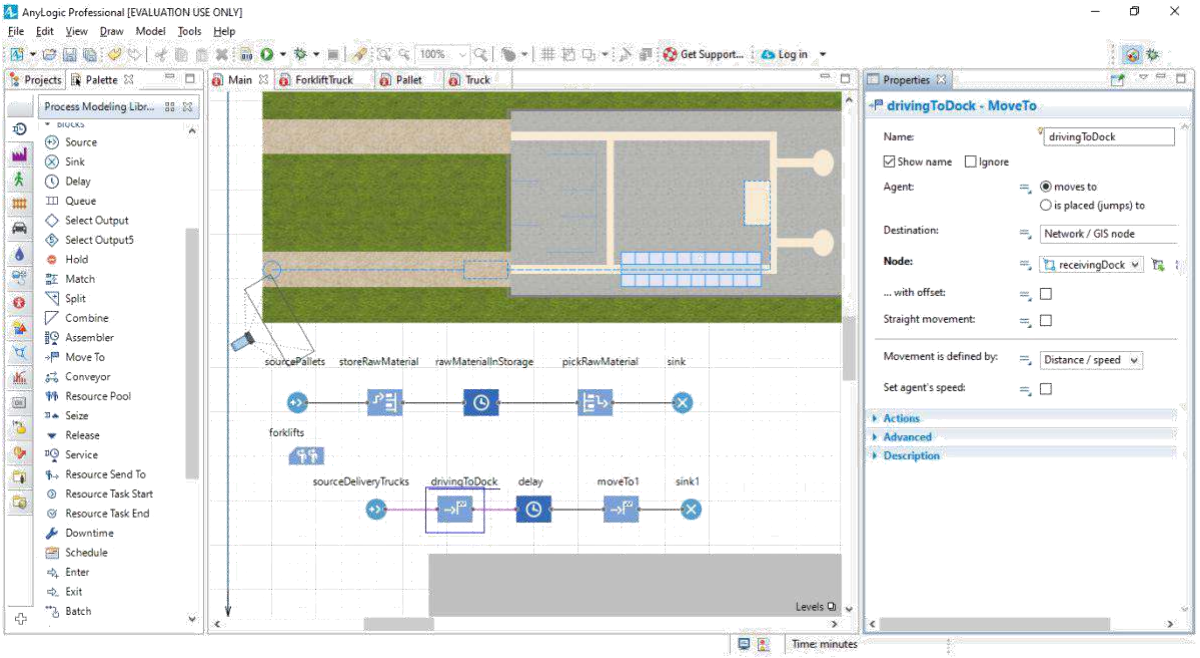
e. In the Node list, click exitNode.

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.

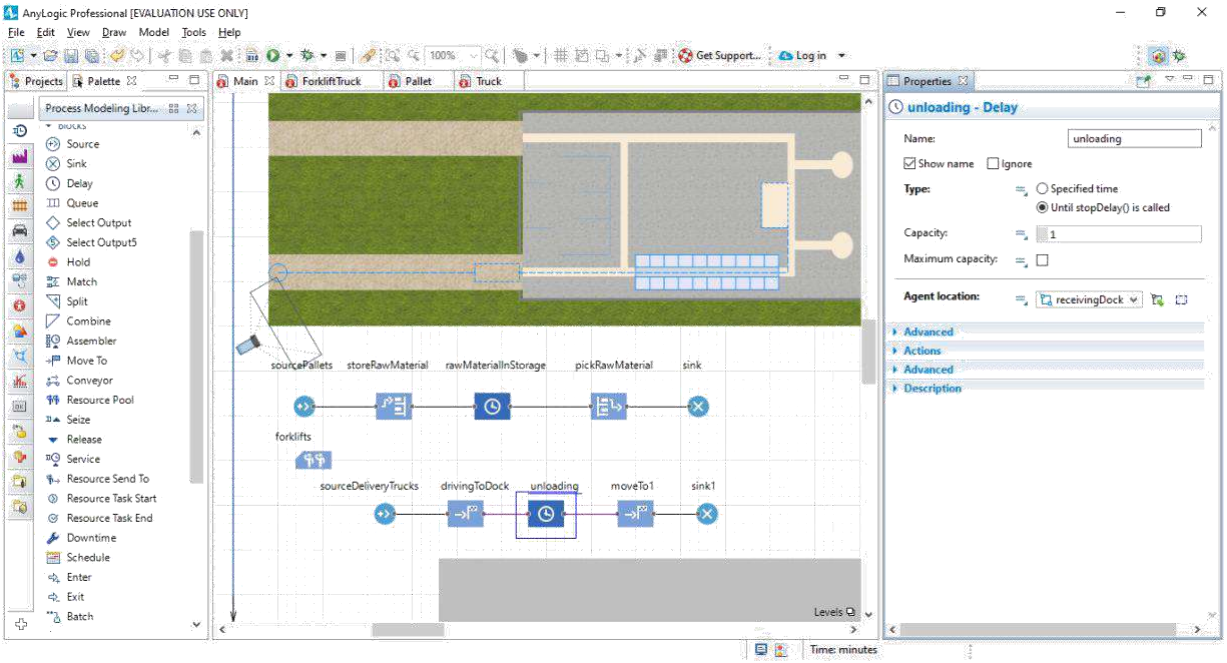


Rename the Delay block to unloading.

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

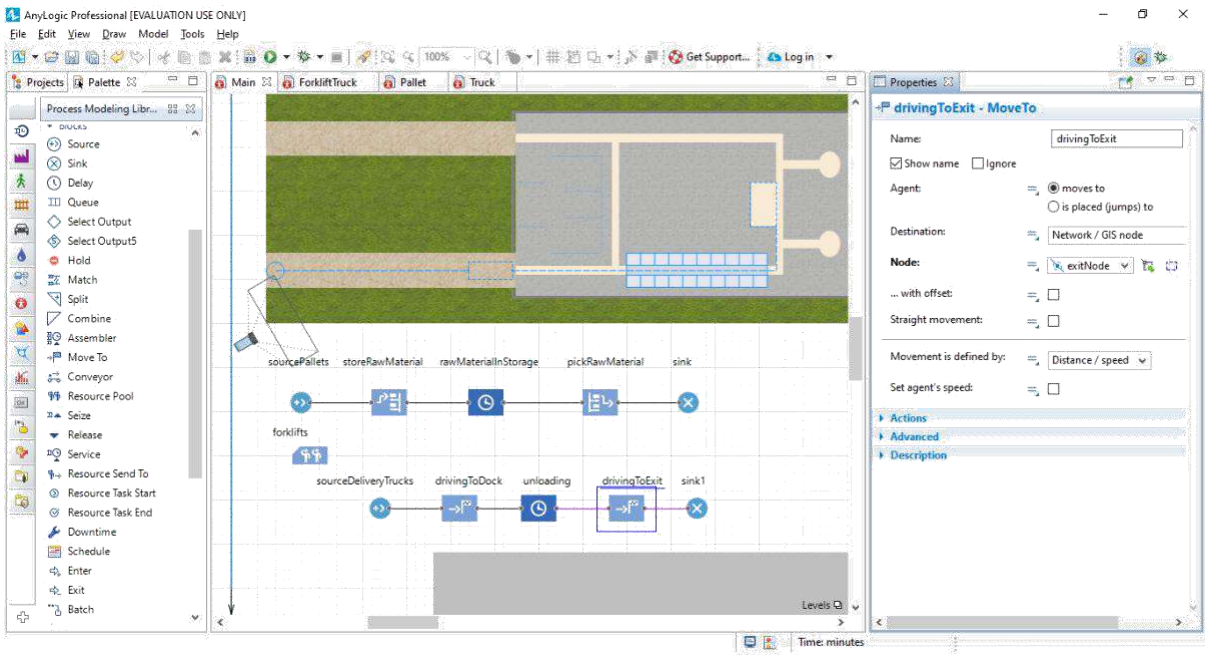
a. In the Type area, click Until stopDelay() is called.

b. In the Agent location list, clickreceivingDock.



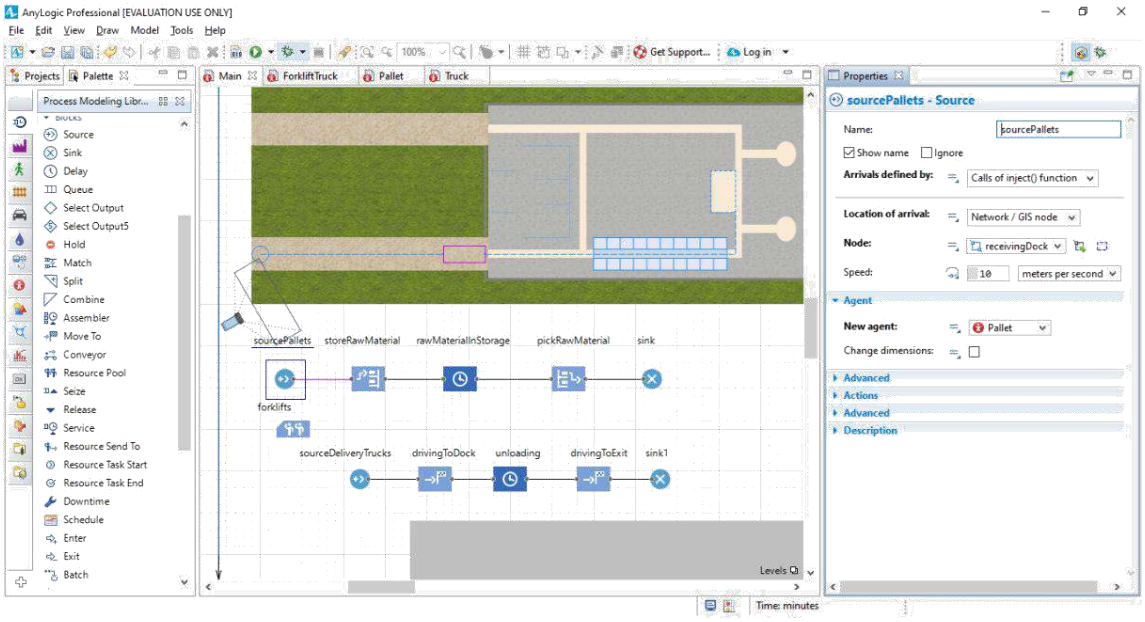
Name the second MoveTo block drivingToExit.

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.



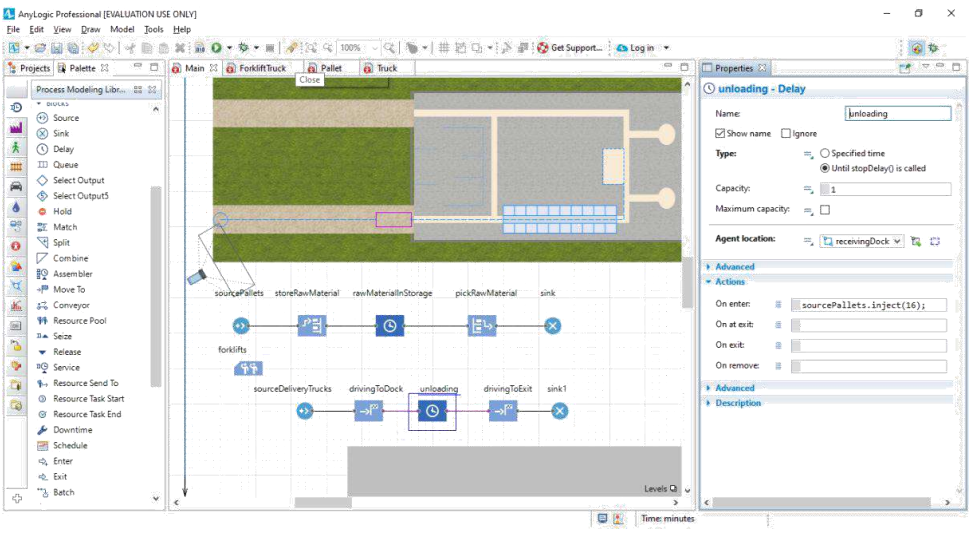
Do the following to have the sourcePallets block generate pallets when a truck enters the unloading block:

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

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

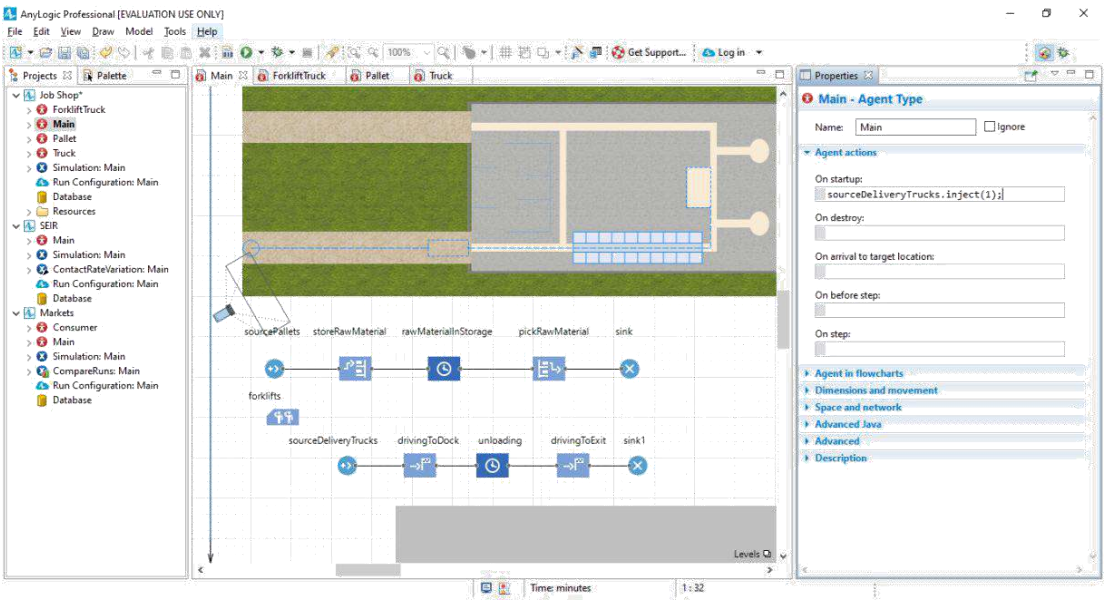
sourcePallets.inject(16).

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);

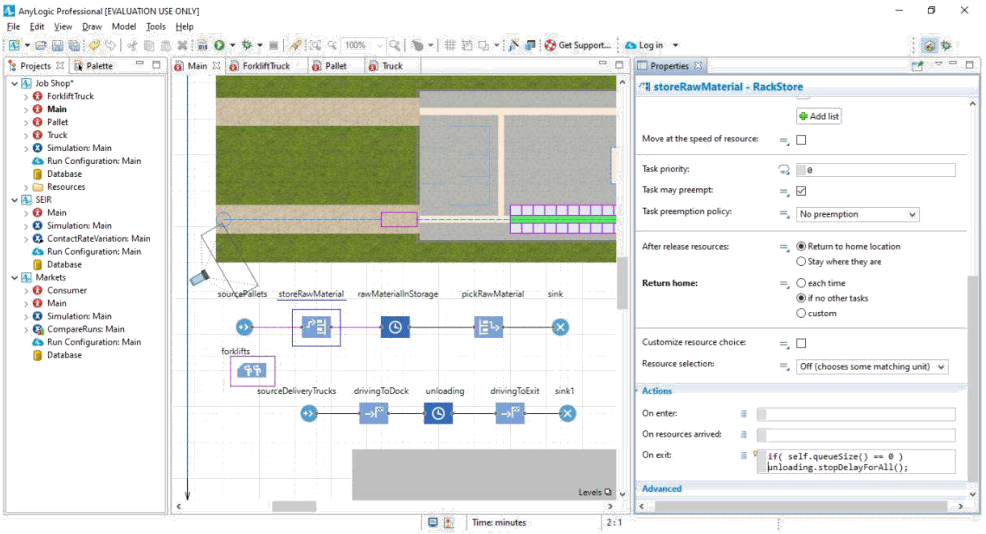


In the storeRawMaterial block's Properties area, expand the Actions section, and in the On exit box, type the following:

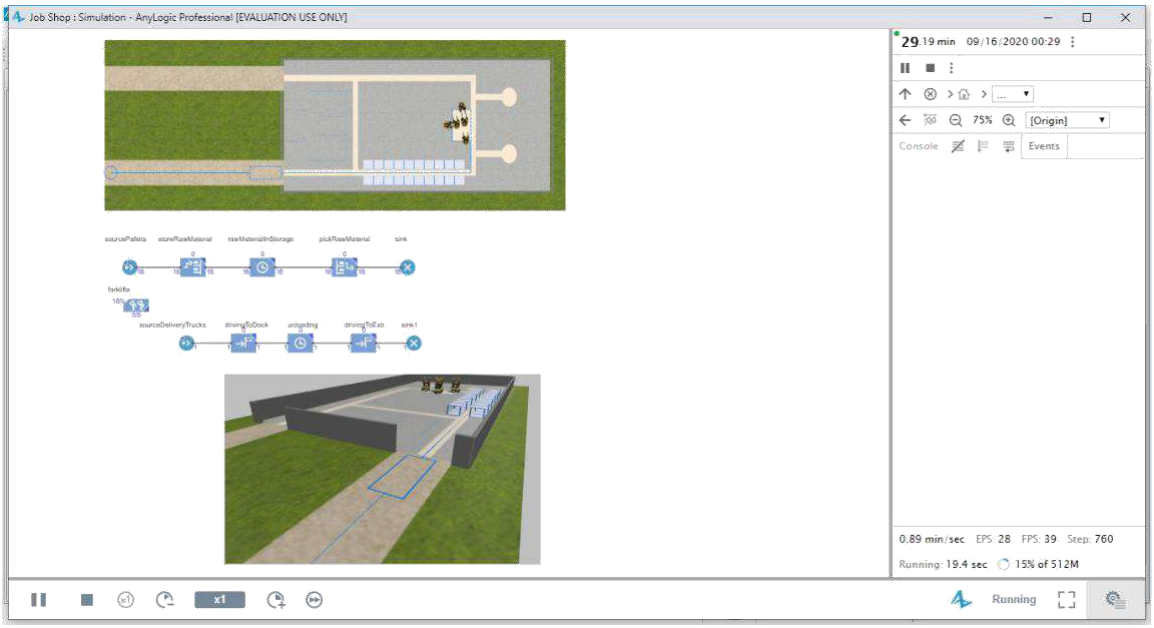
```
if( self.queueSize() == 0 )

unloading.stopDelayForAll();
```

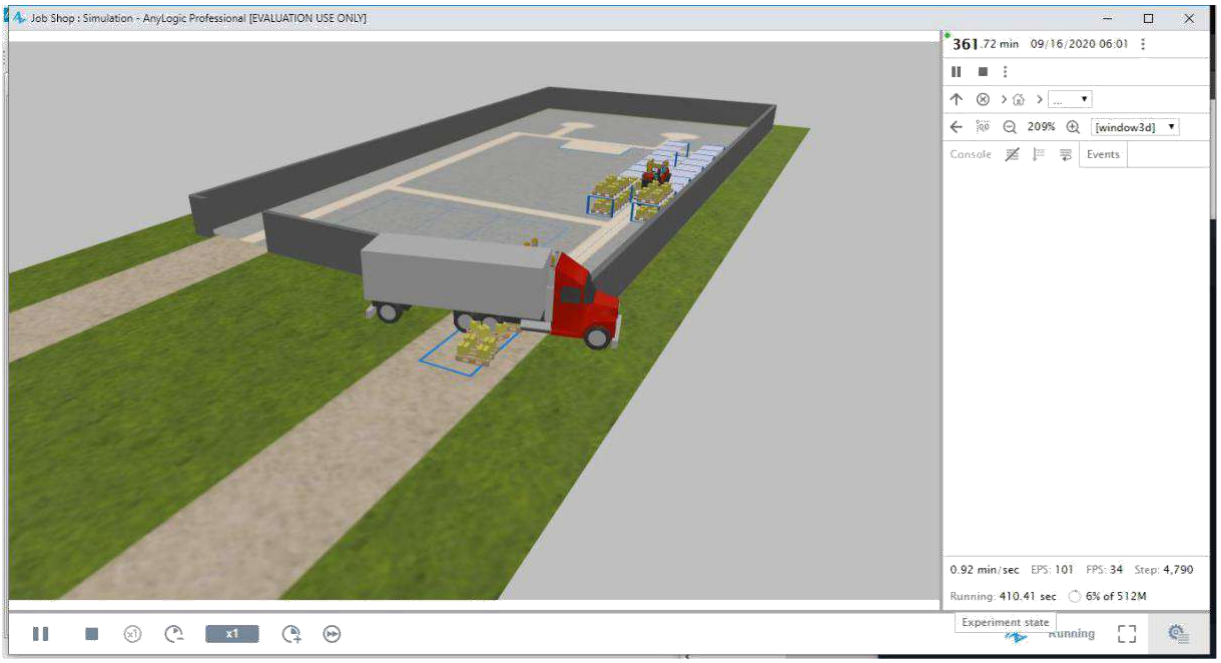
In this example, self is a shortcut we use to refer to the block storeRawMaterial from its own action.



Run the Model

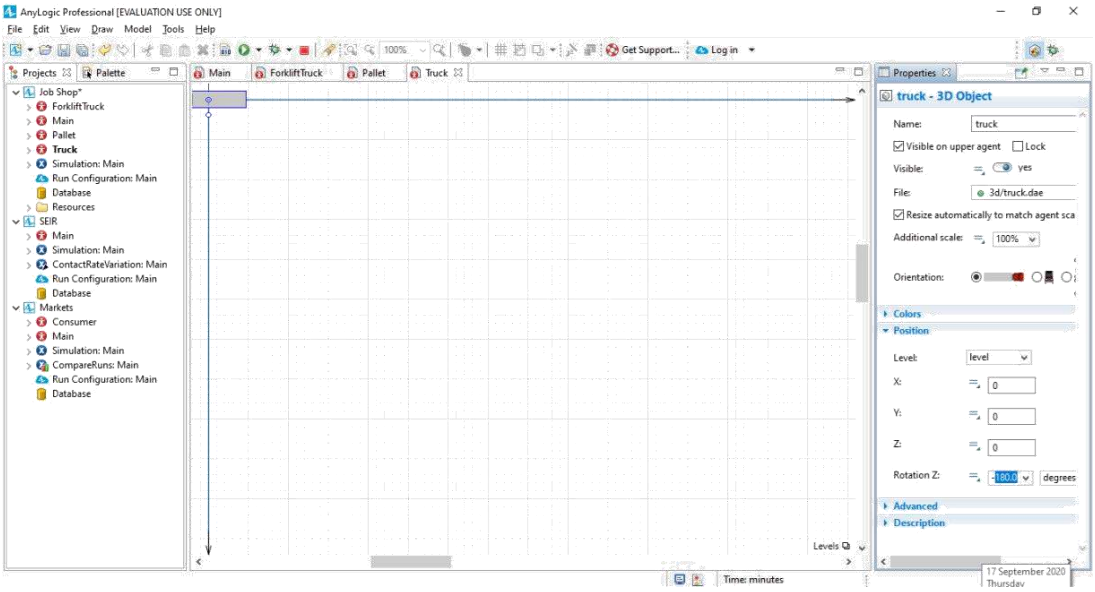


If the truck is aligned incorrectly as in the figure below, do the following to fix it.



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.



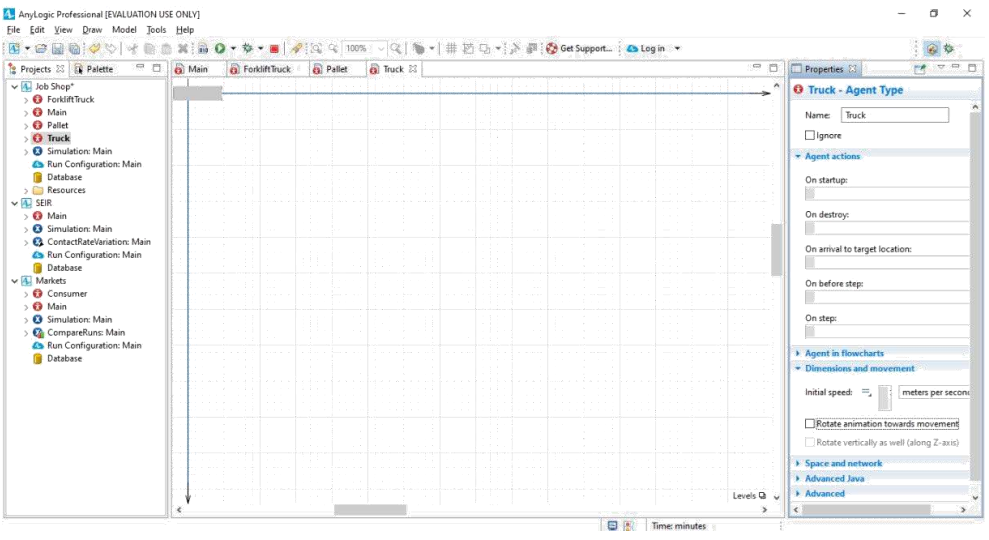
We've changed the truck figure's position, but we'll also need to change AnyLogic's default setting to make sure the program doesn't rotate it a second time.

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.

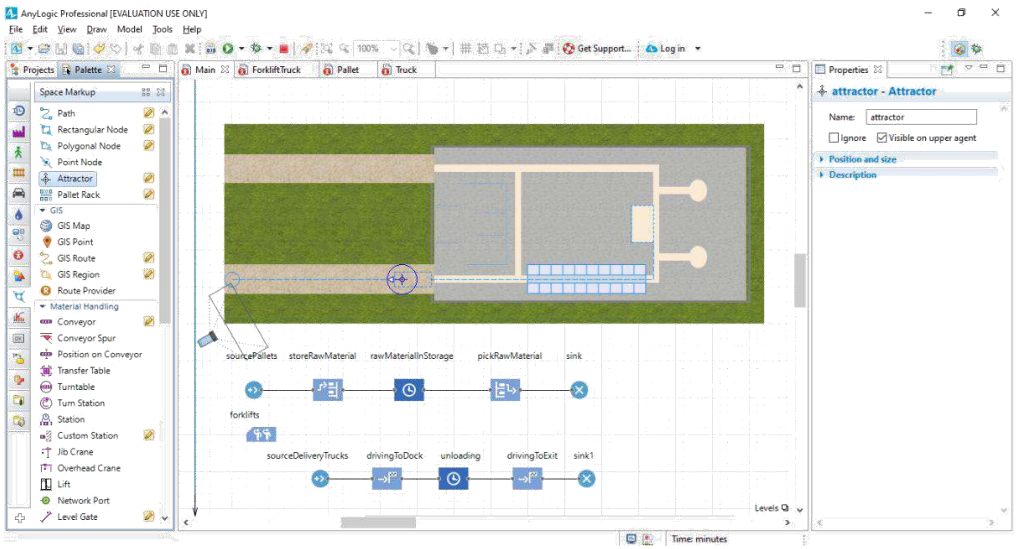
c. Clear the Rotate animation towards movement check box.



Open the Main diagram.

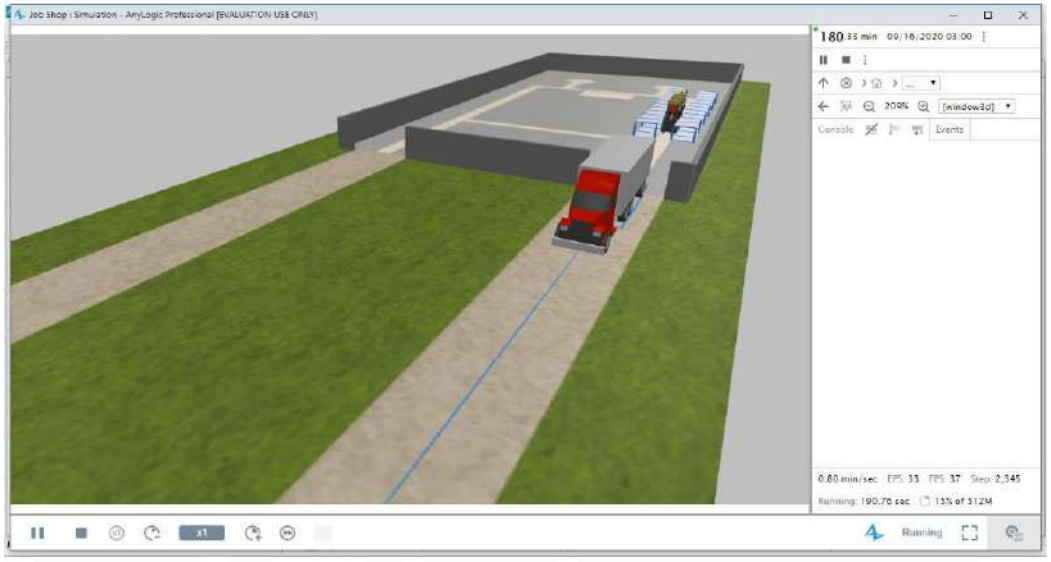
To ensure the pallets are correctly positioned in the receivingDock network node, open the Space Markup palette, and drag an Attractor into receivingDock.

Let it face the entrance.



Run the model to check the truck behaviour.


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

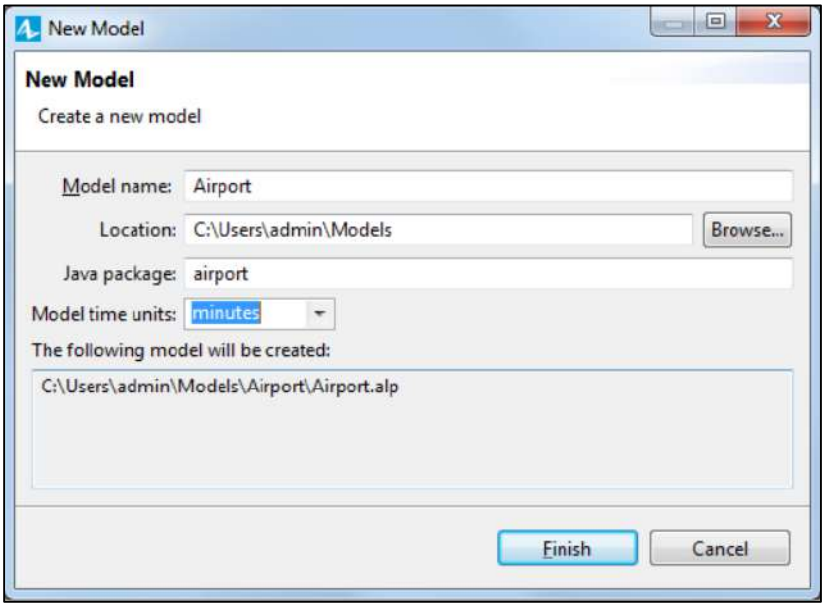
Practical No: - 6

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 passengers' tickets before they allow them to board.

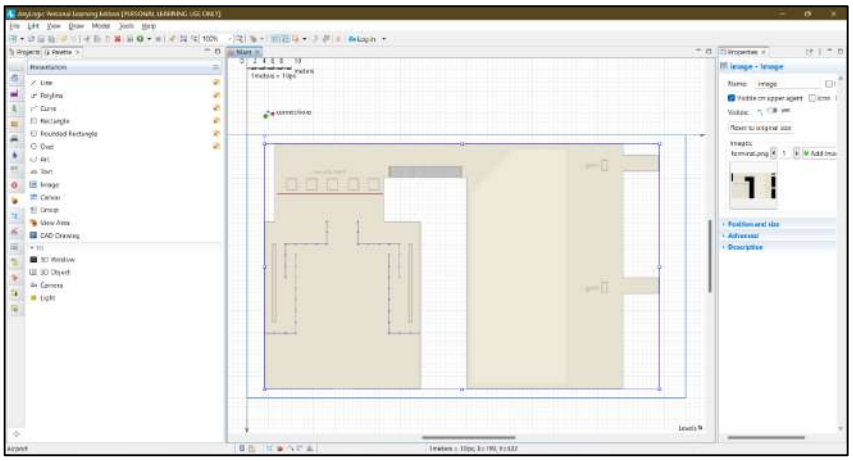
Code:-

Create a new model and name it Airport. Select minutes as the model time units.

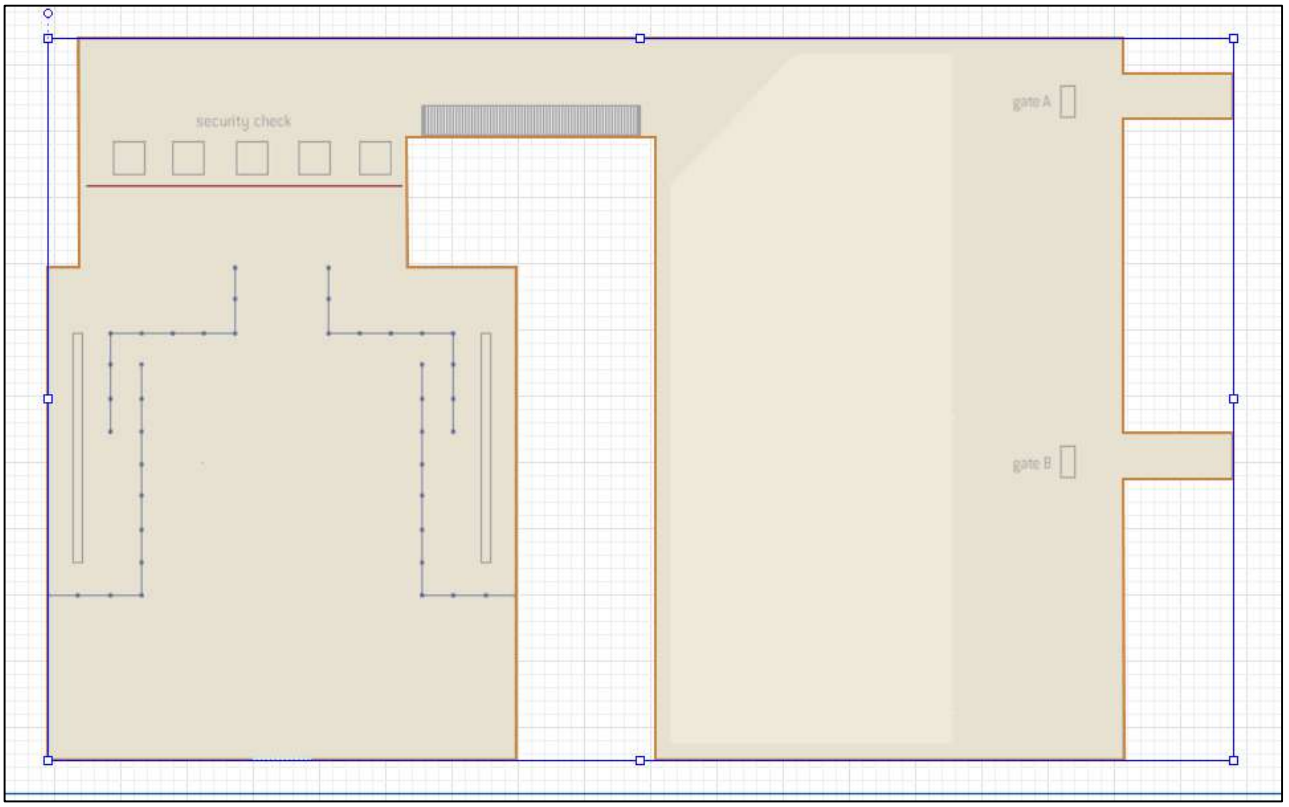


Drag an Image from the Presentation palette on to the Main diagram.Choose terminal.png image file from AnyLogic folder/resources/AnyLogic in 3 days/Airport. On the Main diagram,

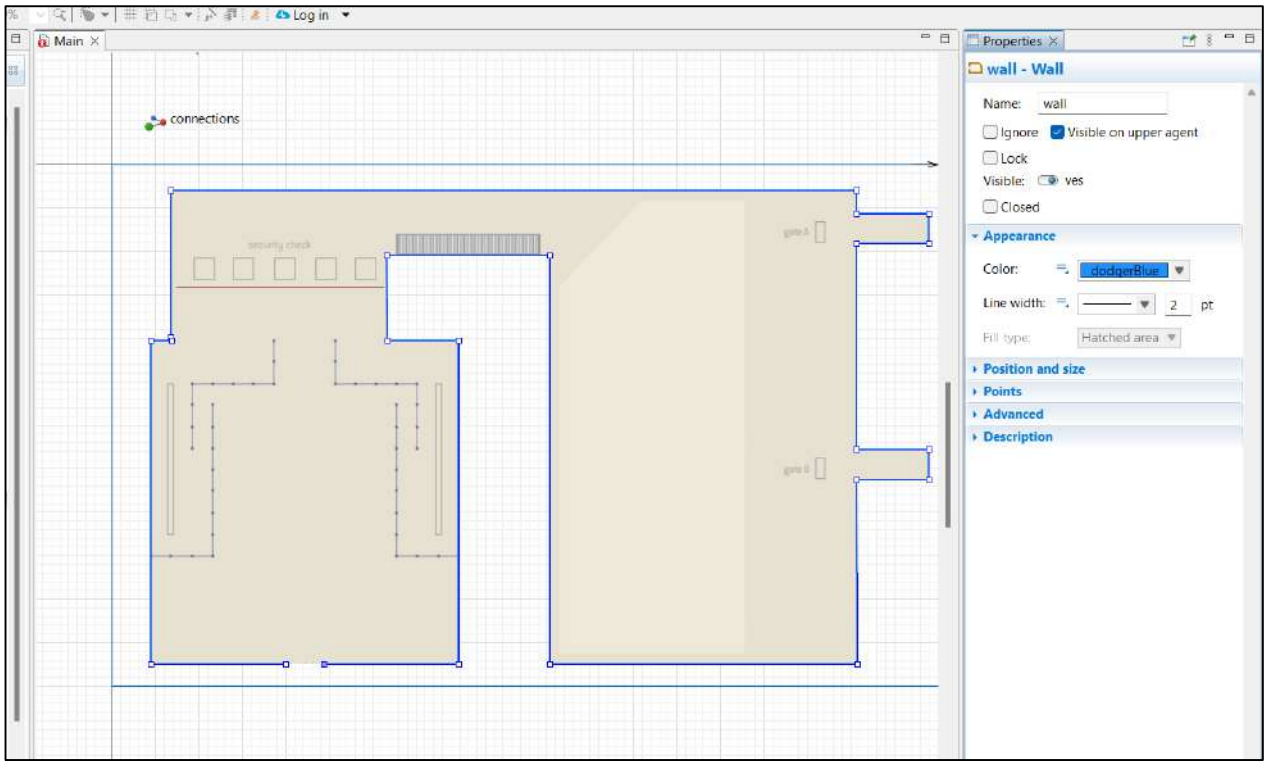
place the image in the blue frame's lower left corner



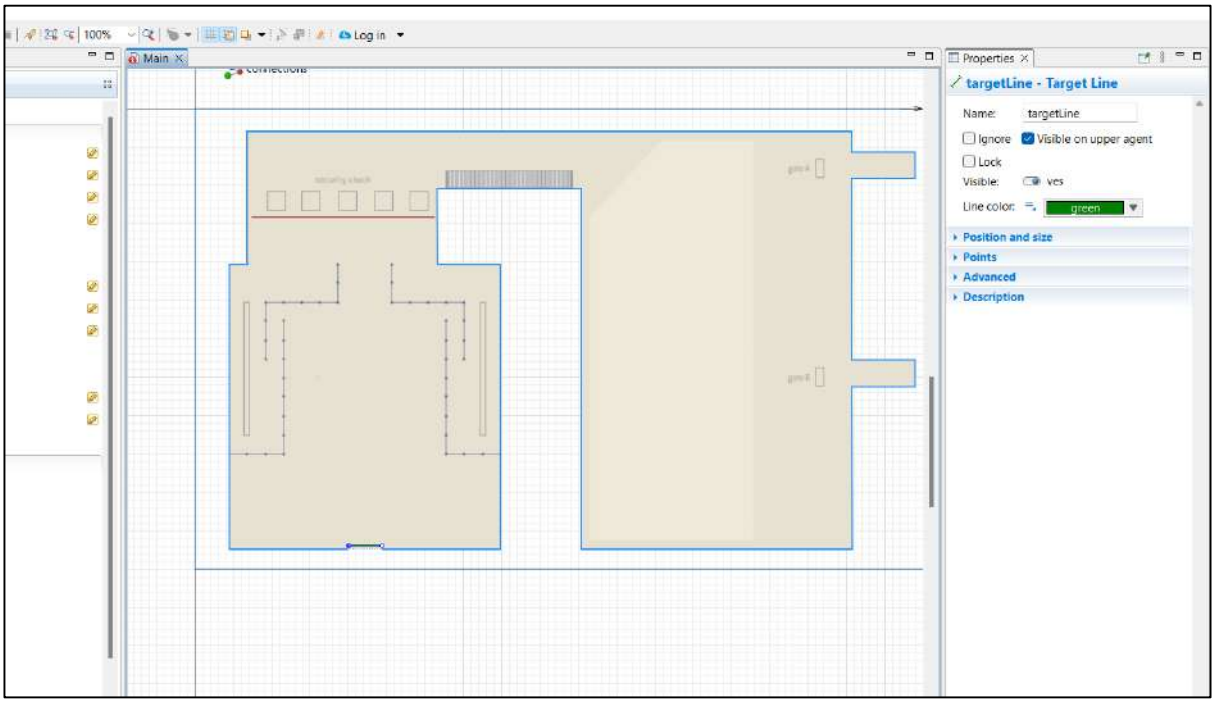
Use the Pedestrian Library palette to draw the airport's walls. Double-click the Wall element you'll find in the Pedestrian Library palette's Space markup section and then draw the wall around the airport building's border



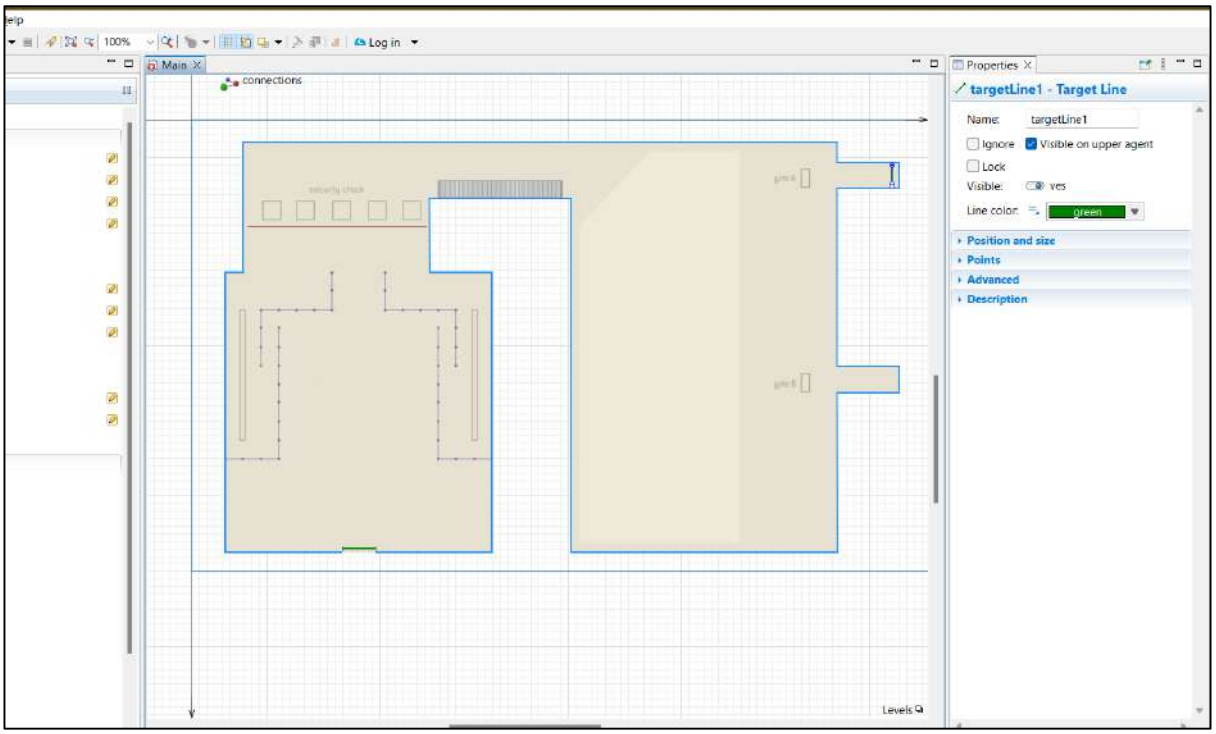
Navigate to the wall's properties and then select the Color: dodgerBlue in the Appearance section.



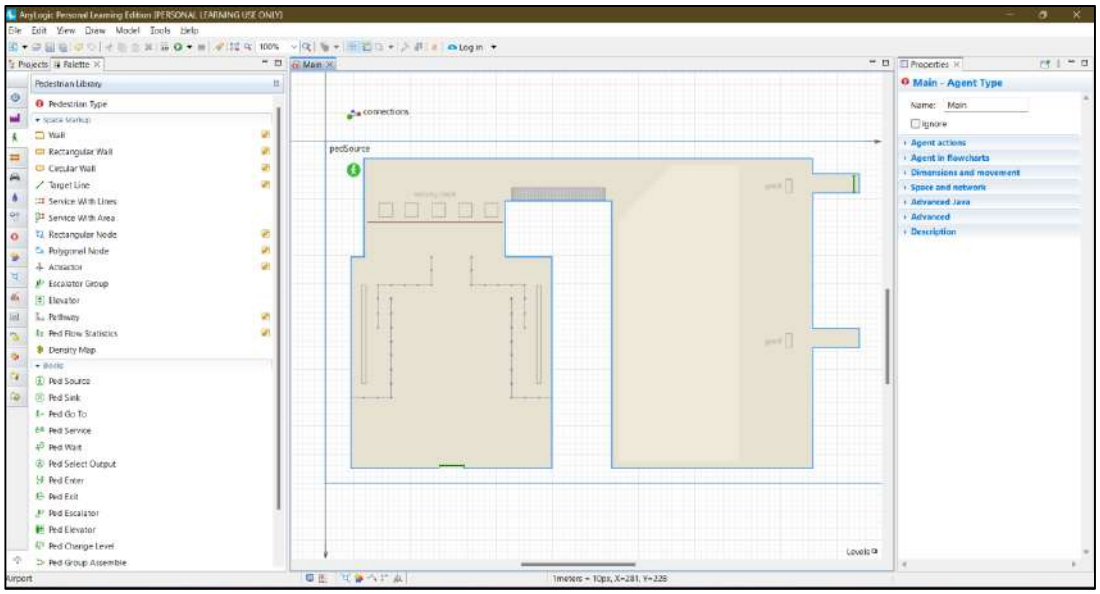
Define the location where your model's passengers appear by dragging the Target Line element from the Pedestrian Library palette on to the graphical diagram, as shown in the figure below.



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 below, and then name it gateLine1.



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

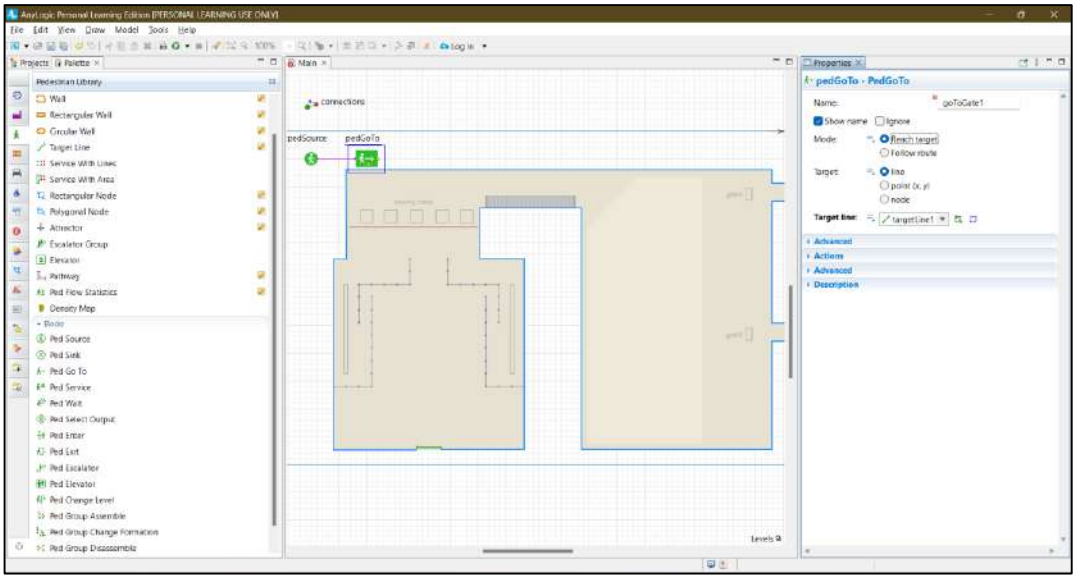


Since we want passengers to arrive randomly at an average rate of 100 passengers per hour, go to the ped Source block properties and then type 100 in the Arrival rate box.

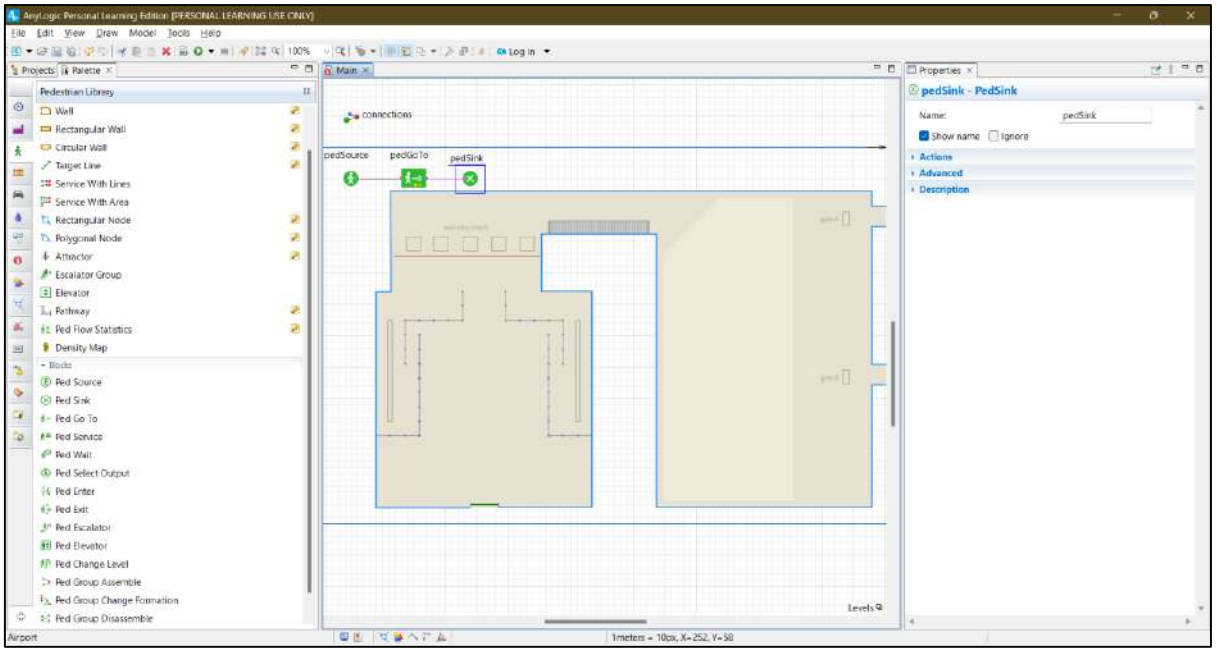
Specify the location where the passengers appear in the simulated system by clicking arrivalLine in the Target line list.

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.

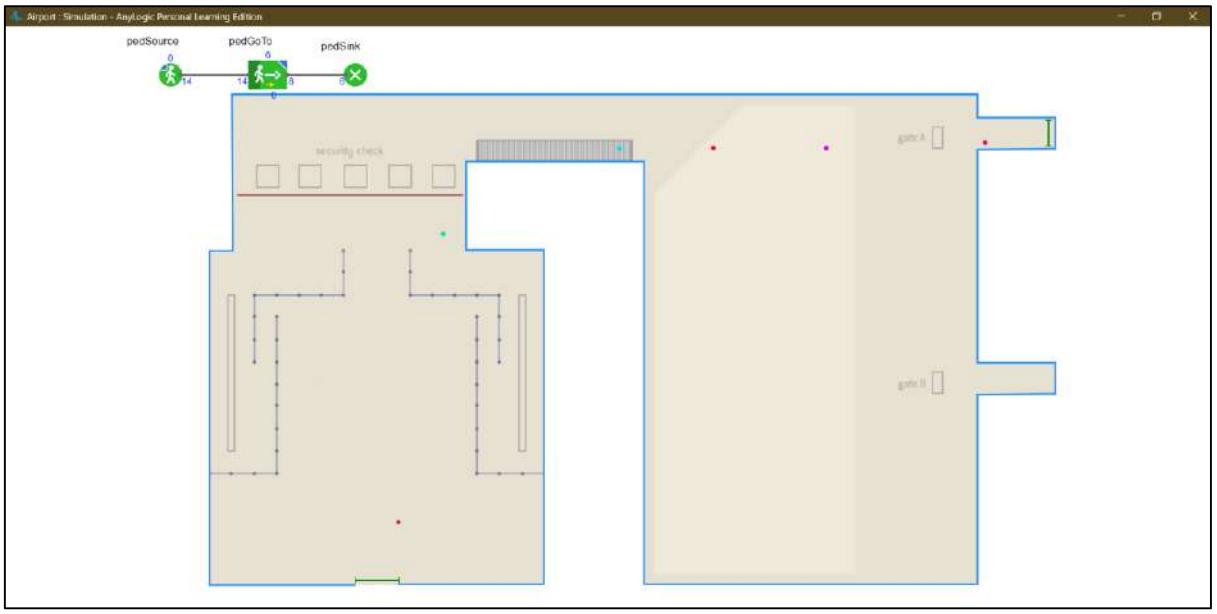
Specify the movement destination by selecting gateLine1 from the Target line combo box.



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

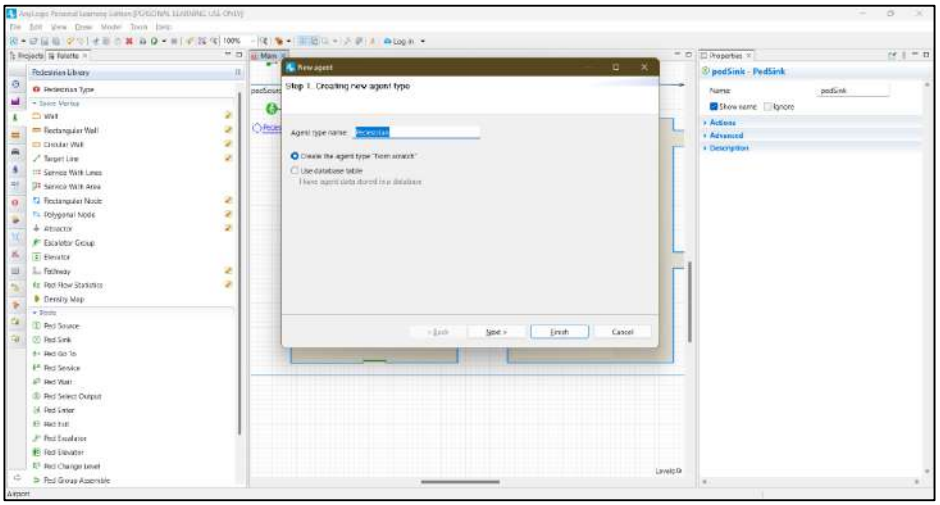


Run the model. In the 2D animation, you'll see the pedestrians move from the airport entrance to the gate.

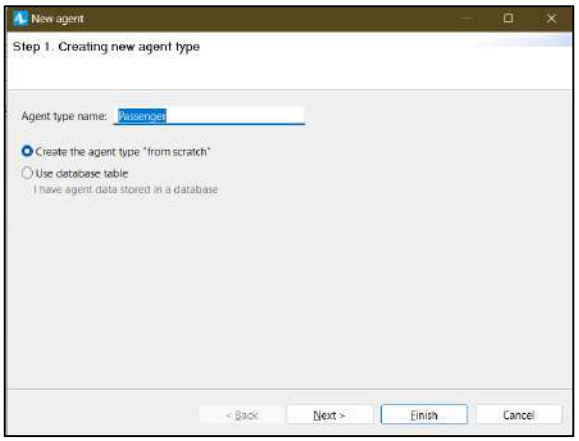


Drawing 3D animation

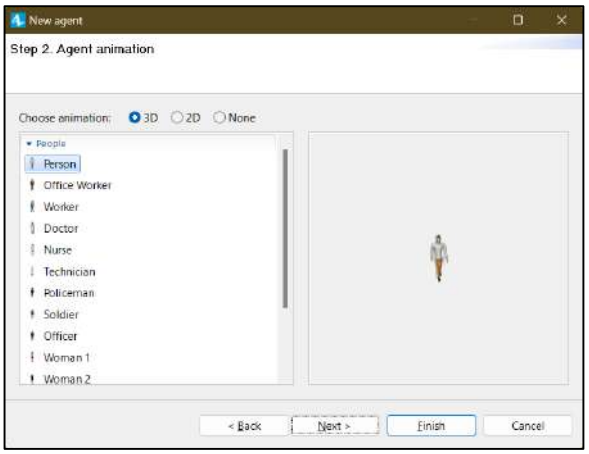
From the Pedestrian Library palette, drag the Pedestrian Type element on to the Main diagram.



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

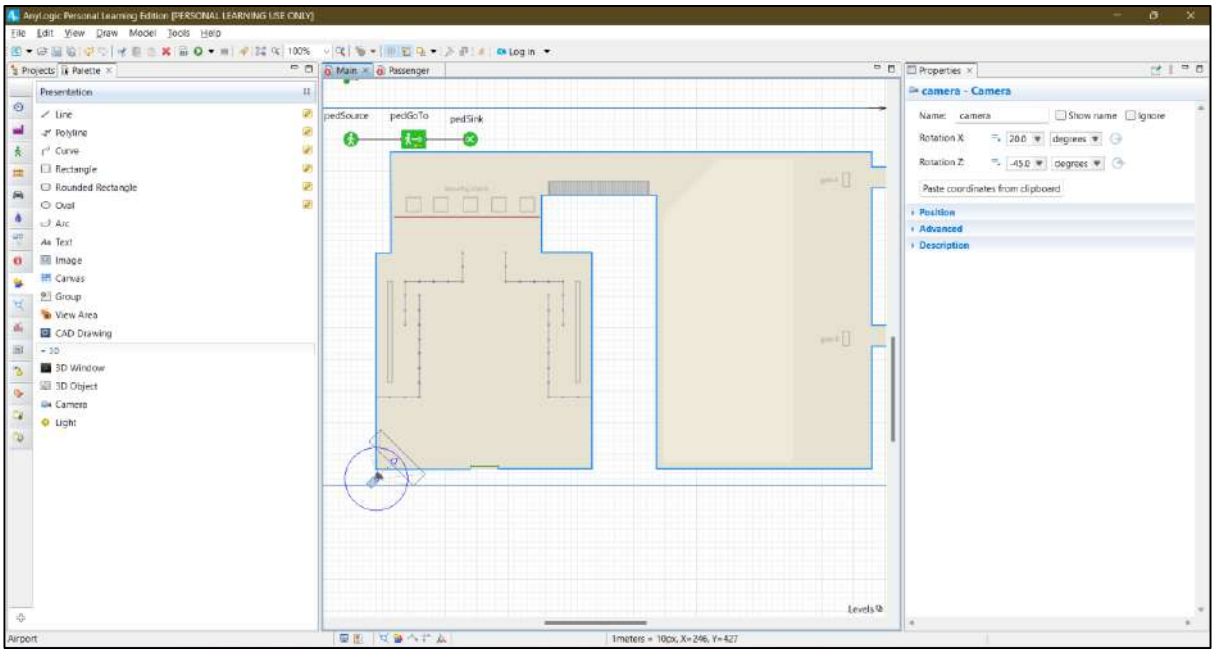


On the Agent animation page, select the General list's first item: Person, and click Finish.

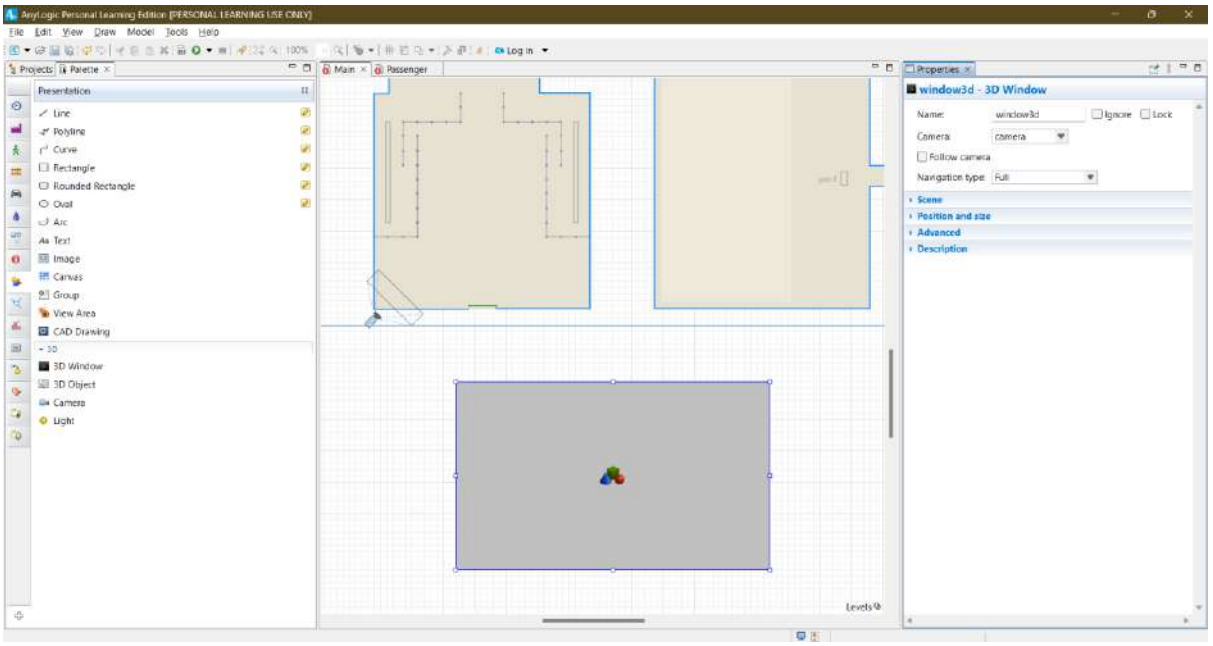


After the Passenger diagram opens, return to the Main diagram.

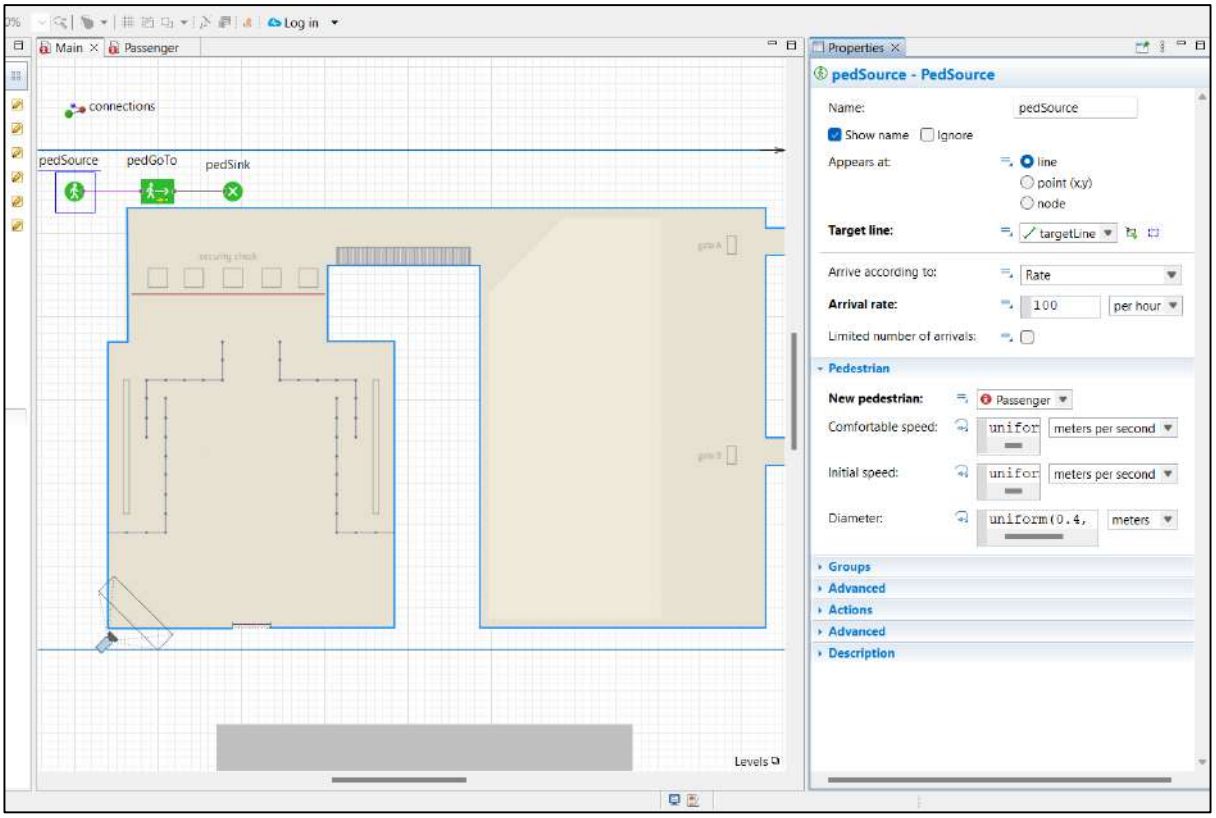
From the Presentation palette, drag the Camera on to the Main diagram and place it so it faces the terminal.



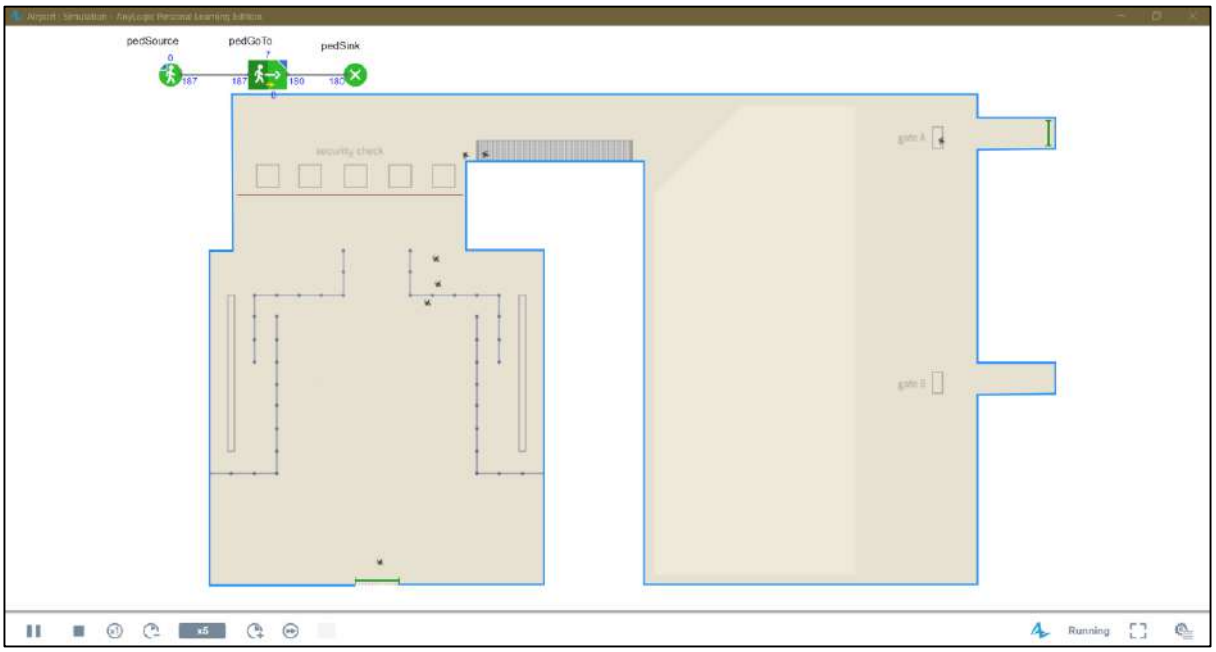
Drag the 3D Window on to the Main diagram and place it below the terminal layout image.

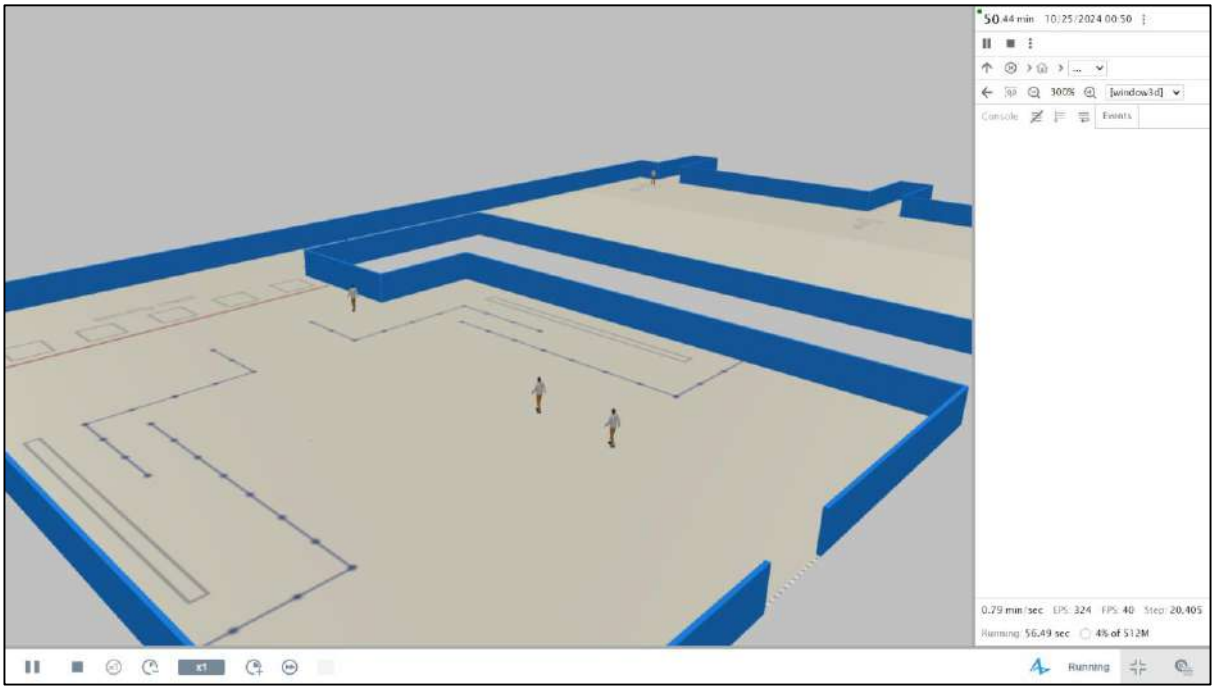


Open the 3D Window properties, and then select camera from the Camera list.We want our flowchart block pedSource to create pedestrians of our custom Passenger type. Open the pedSource properties, and then select Passenger from the New pedestrian box in the Pedestrian section.



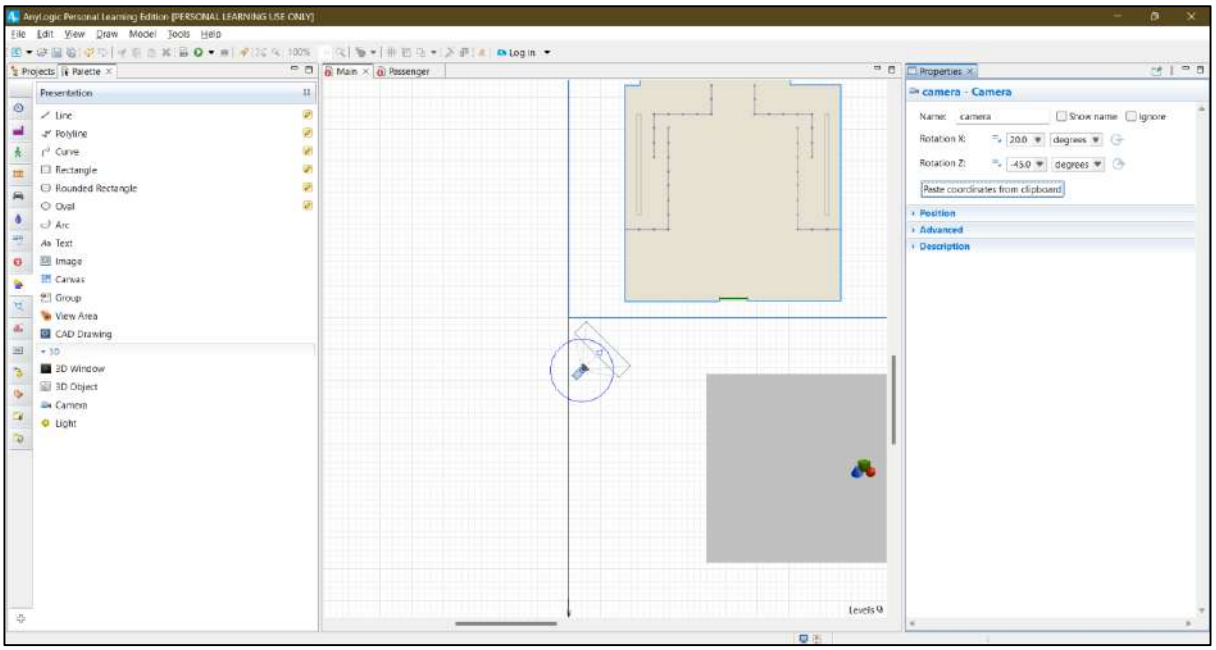
Run the model, and you'll see pedestrians move from the entry to the gate inside the building. You can switch to a 3D view by clicking the toolbar's Navigate to view area… button and then selecting [window3d] from the list.





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

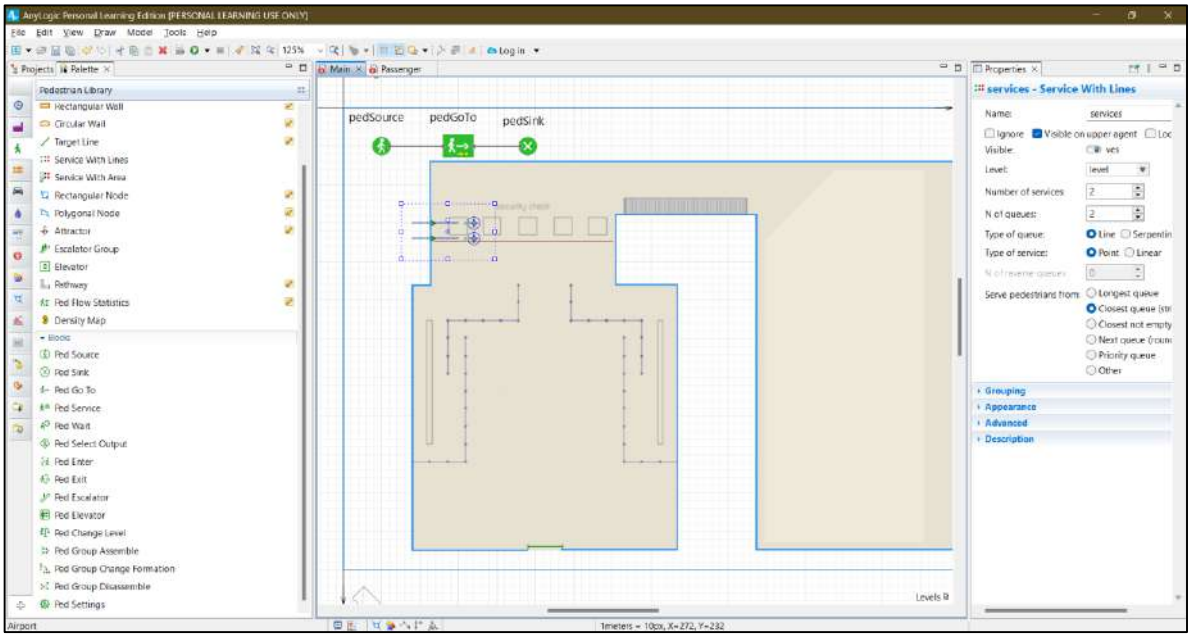
Close the model's window, open the camera's properties, and then apply the optimal camera you selected during the previous step by clicking Paste coordinates from clipboard.



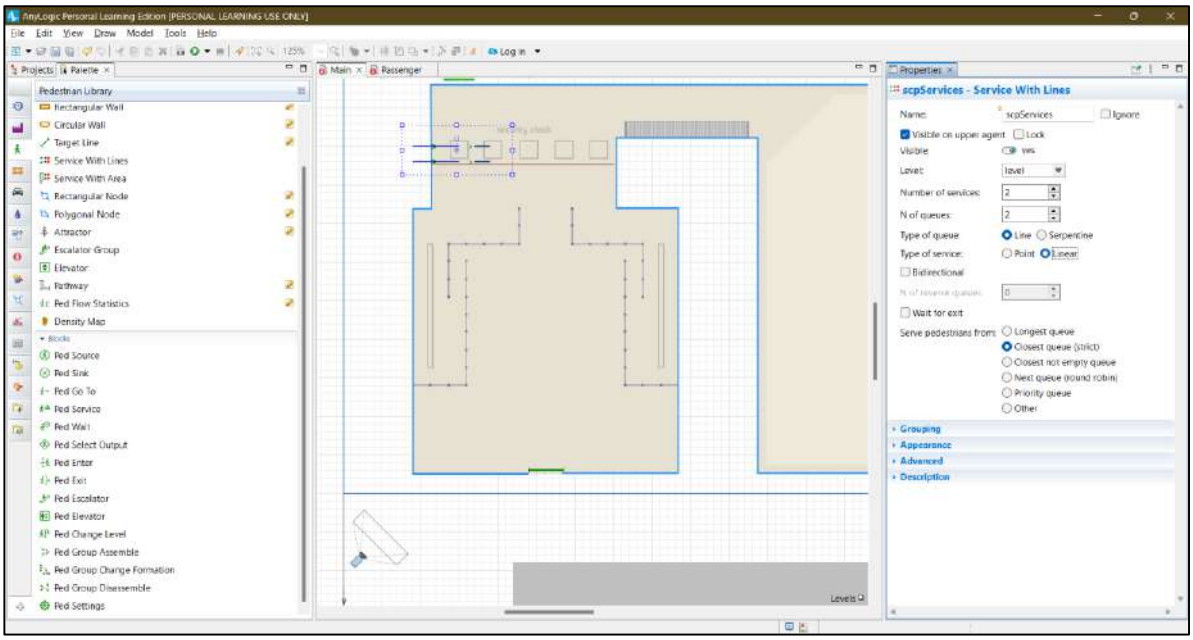
Run the model a second time and view the 3D view that the new camera position provides.

Adding security checkpoints

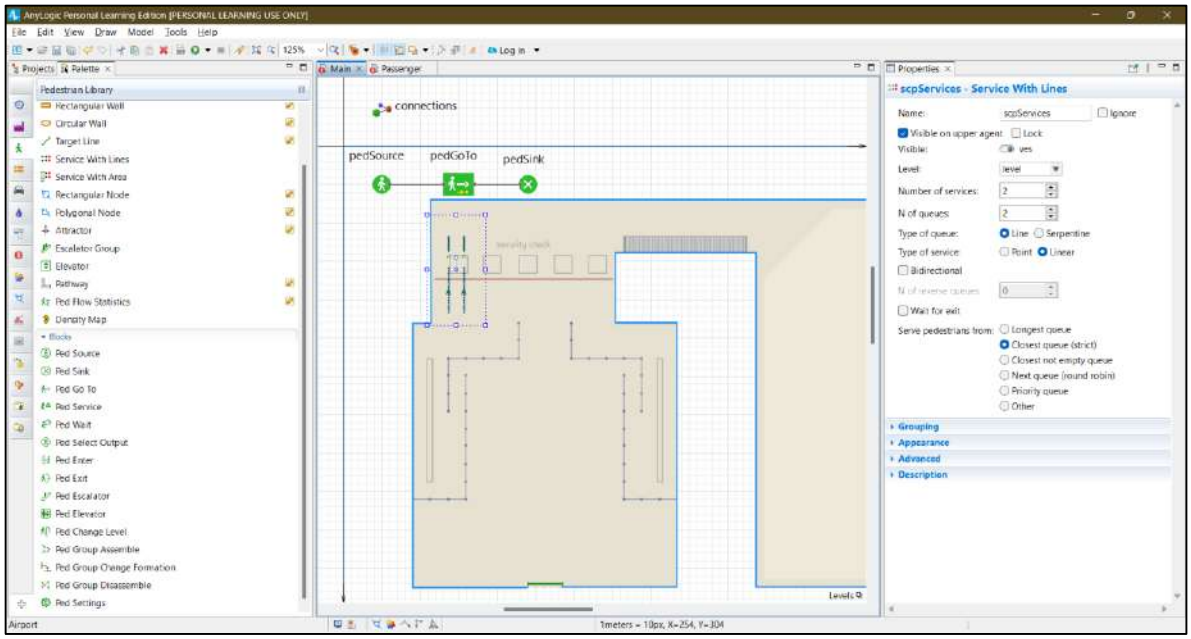
Drag the Service with Lines element from the Pedestrian Library palette on to the terminal layout. By default, a service will have two service points and two queue lines that lead to the service points.



Open the Service With Lines properties area, use the Name box to name the shape scpServices - in this case, “scp” stands for security checkpoints - and then change the Type of service to Linear.



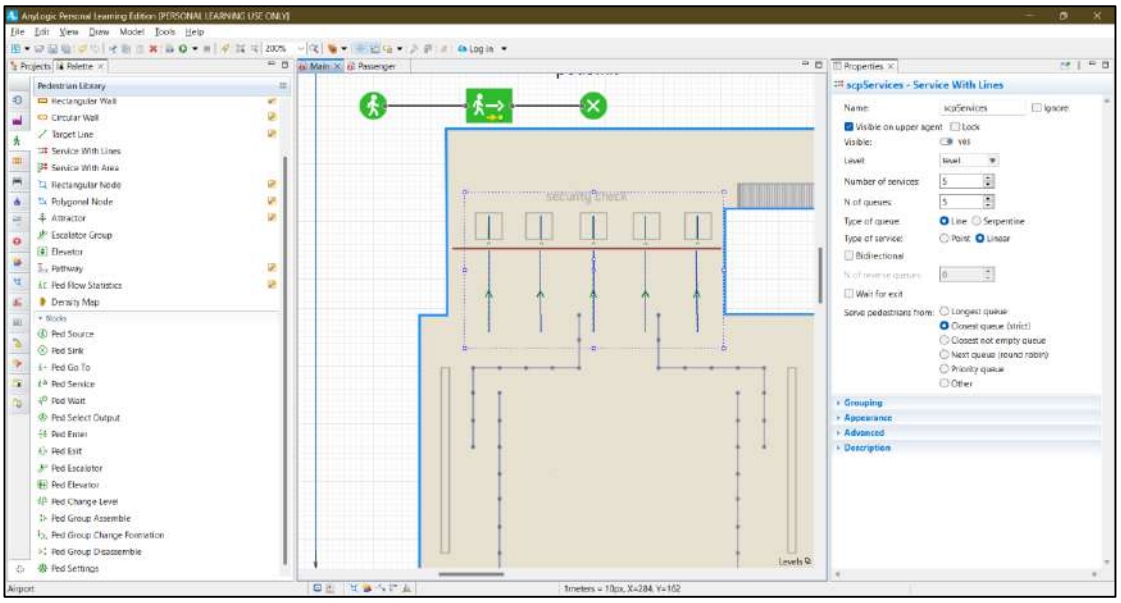
Use the round handle above the shape's center to rotate the service. So that the arrow points upward



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

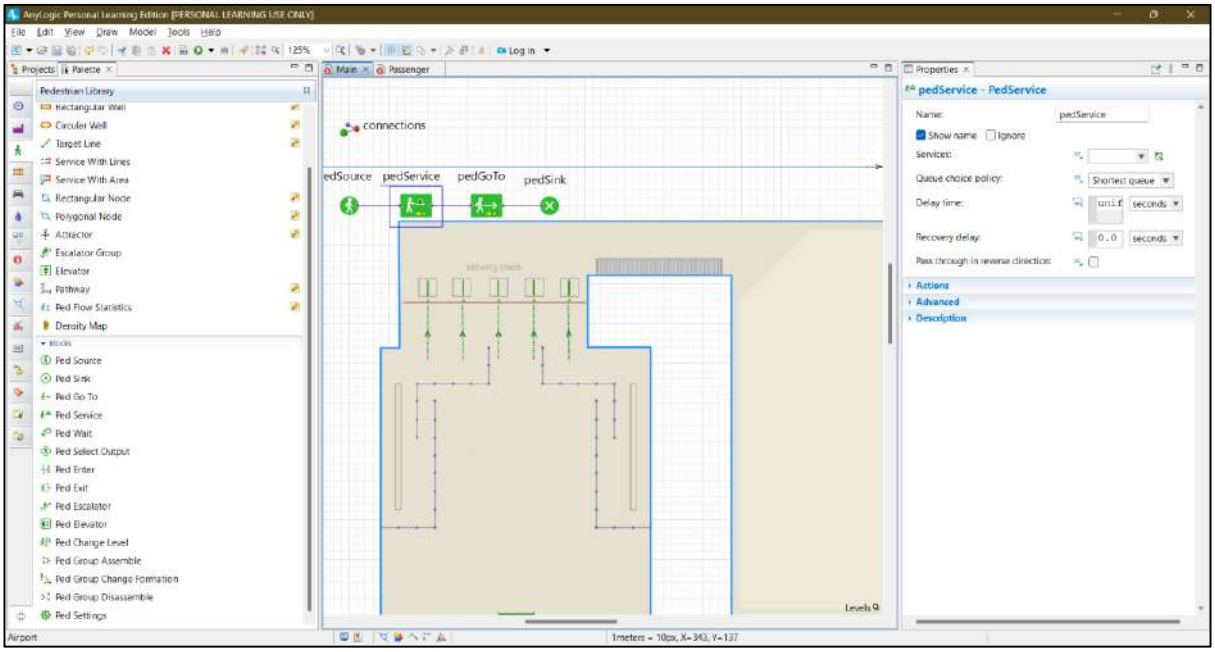
Select the next service line Accurately place the service line on top of the second security checkpoint placeholder and then adjust the queue location.

Navigate to the Service with Lines shape's properties and then change both the Number of services and Number of queue to 5 and then adjust the queue location as before. . After you've completed this step, the service shapes should look like those in the figure below.

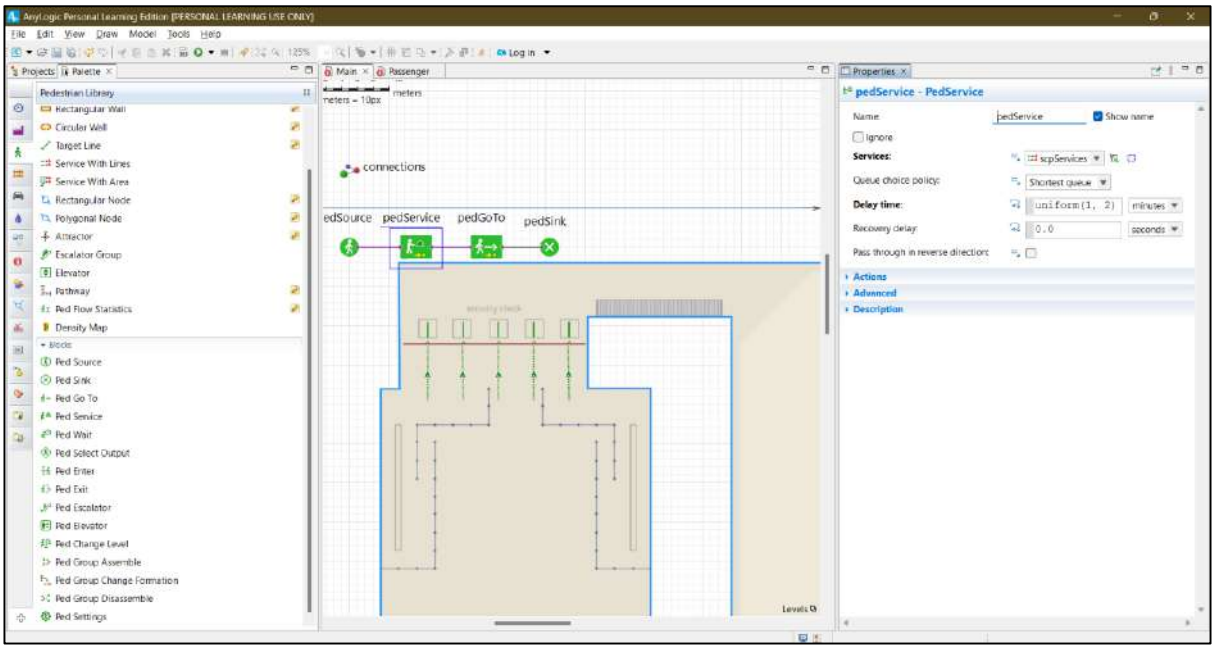


Add the PedService block on to the flowchart between the PedSource and PedGoTo blocks to make pedestrians pass through the service we defined using the referenced Service with Lines shape, and

Then name it securityCheck.



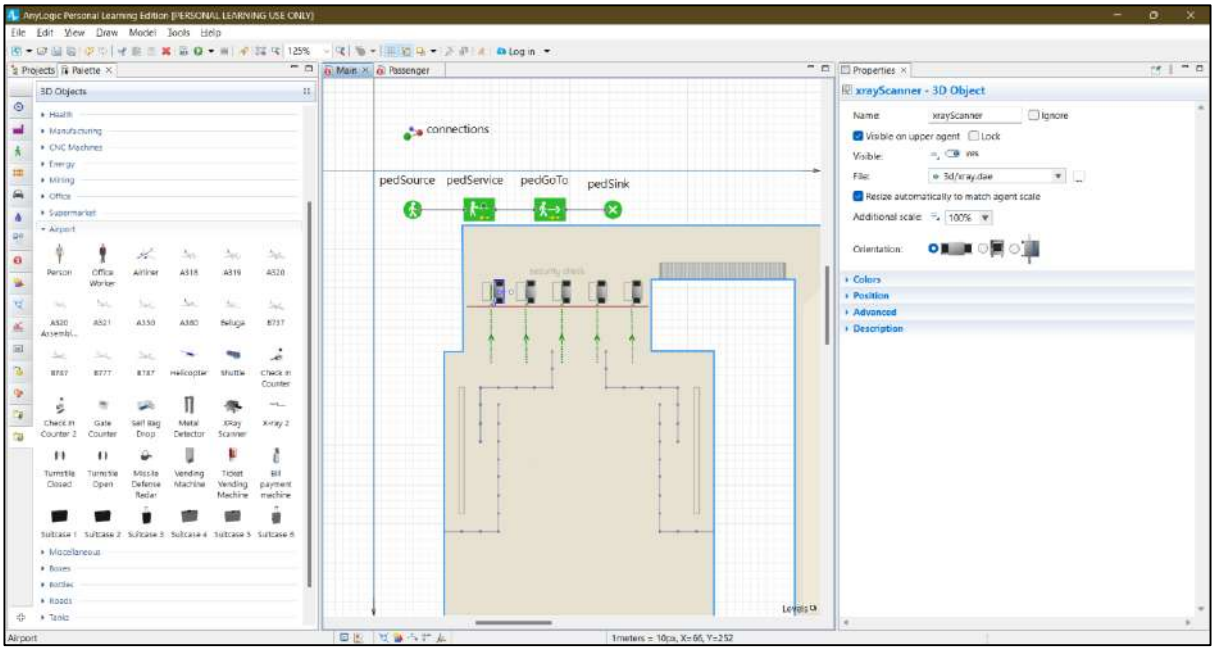
Go to the securityCheck block's properties. Select the services scpServices as Services.



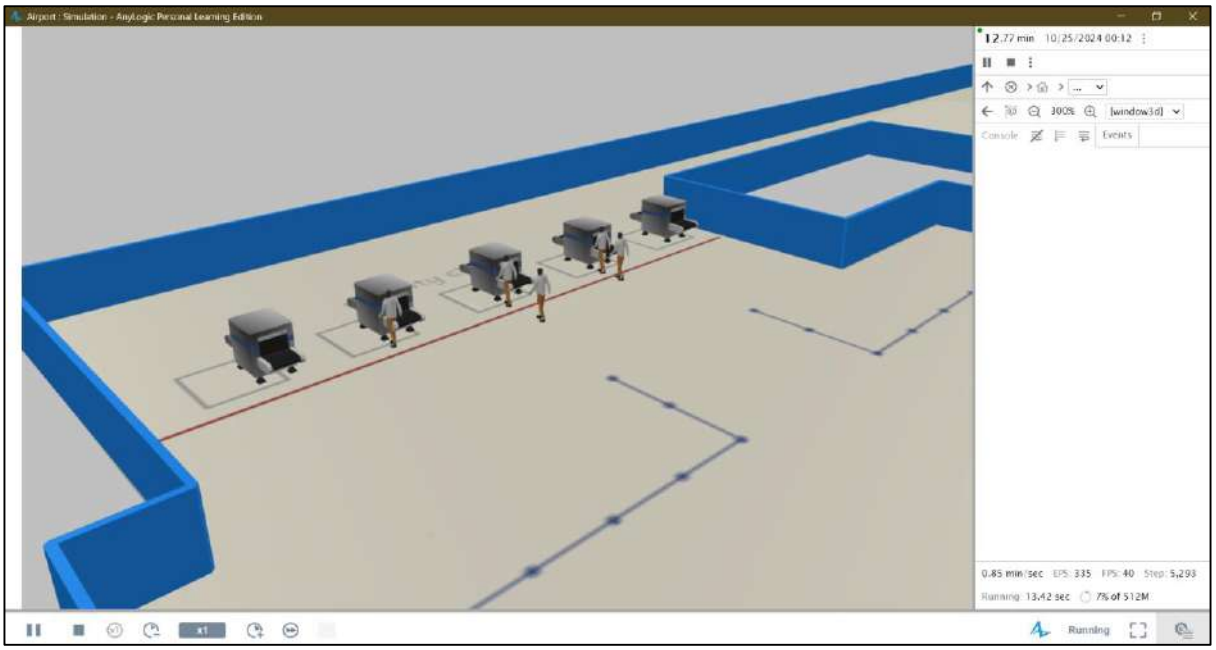
Since we assume it takes between 1 to 2 minutes to pass through the security checkpoint, type uniform (1, 2) minutes as the Delay time.

Now let's add 3D models of the security checkpoints. Using the 3D Objects palette, Airport sections Metal Detector and XRay Scanner elements, draw five security checkpoints.

You will see the message box, prompting you to change the scale of 3D object. Select the option Do not ask me again and click OK.

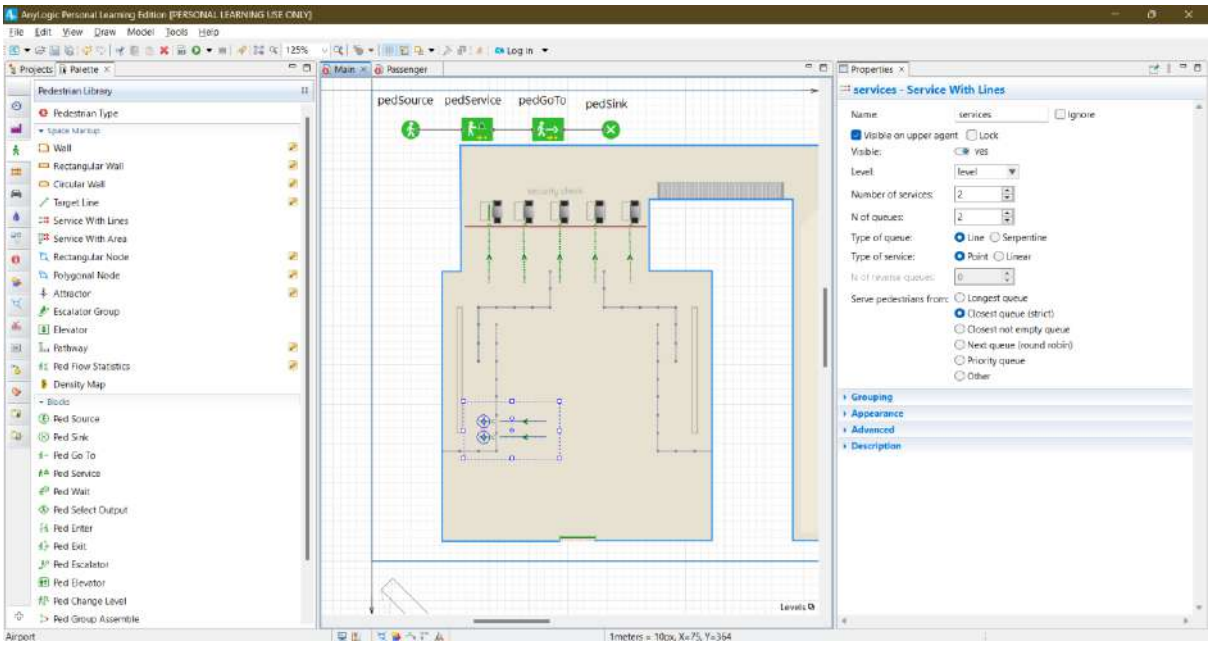


Run the model. You'll see that passengers are now scanned at the security checkpoints.



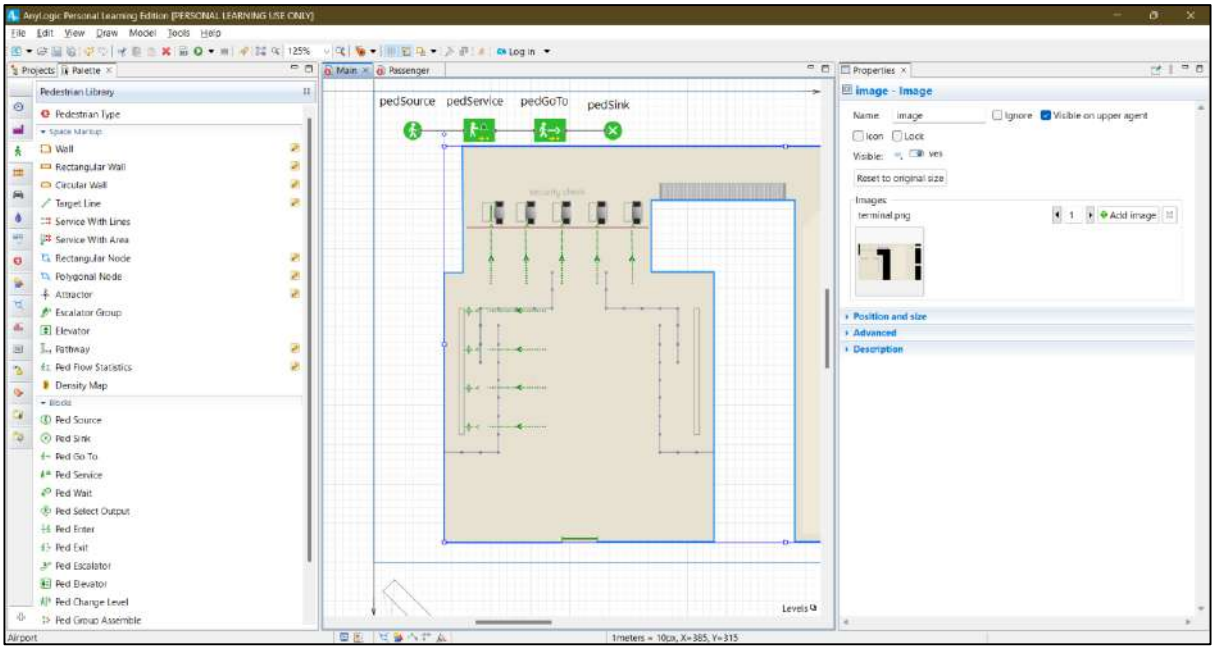
Adding check-in facilities

Draw check-in locations with another Service with Lines shape. Rotate it so that the arrow points towards left.

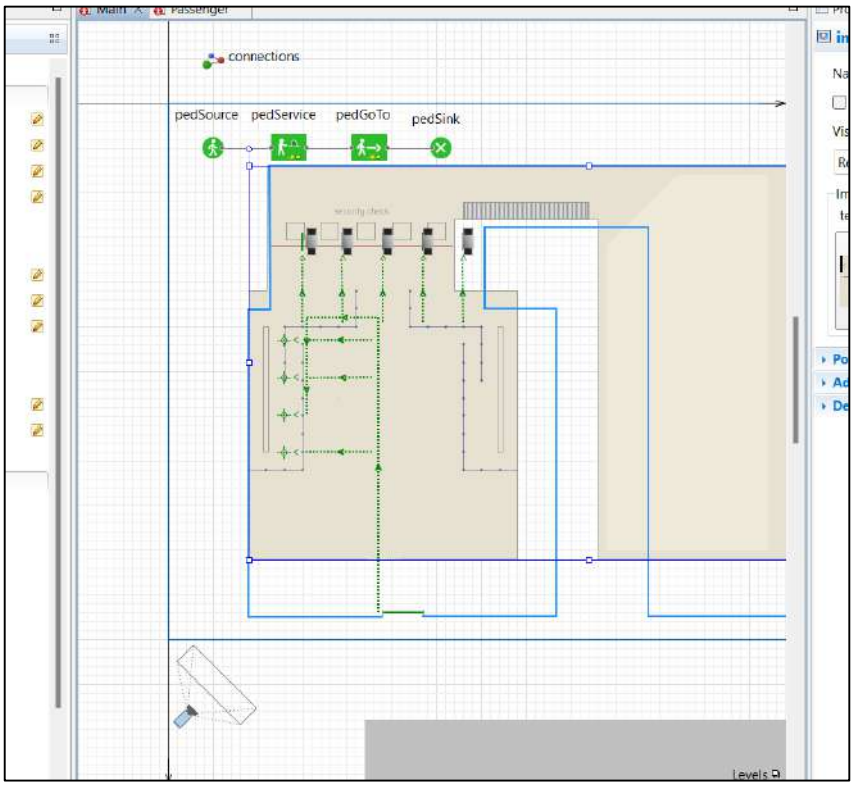


Navigate to the Service with Lines shape's properties and then change both the Number of services and Number of lines to 4.

Name the services checkInServices. Place the shape in the location shown in the figure below.

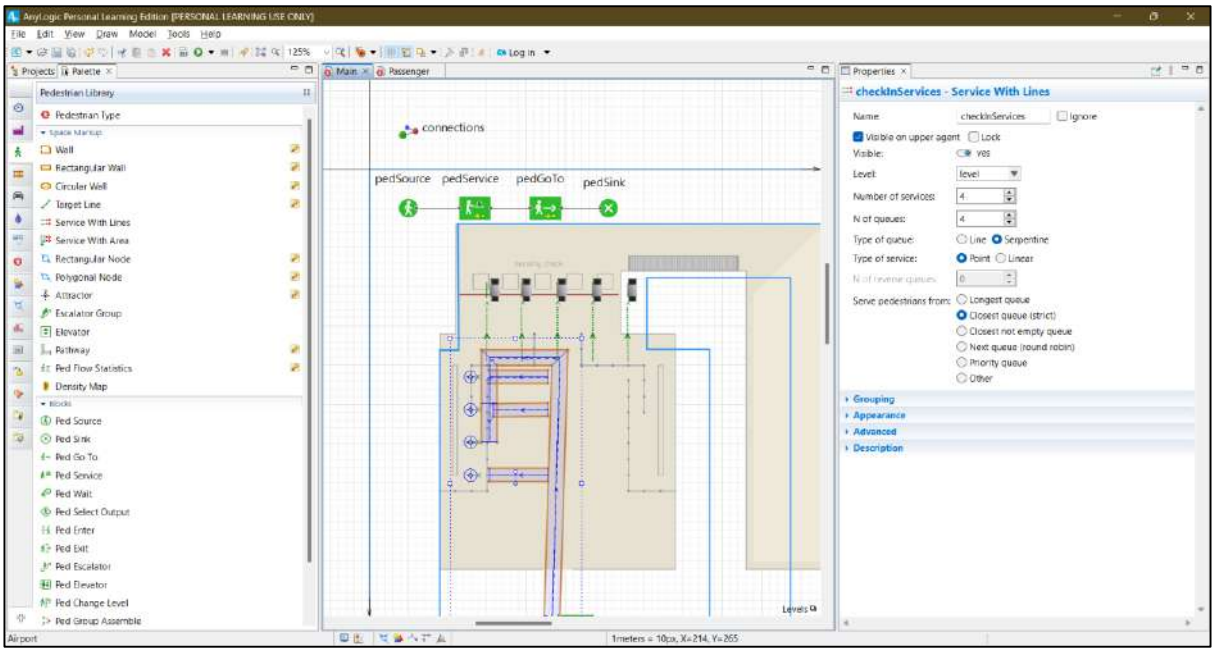


Add more salient points to the line. Right-click the queue line and choose Add points from the pop-up menu. Add more points by clicking where you want to place the line's salient points. Finish drawing the line by double-clicking. Finally you should get the queue line of the following form



In the properties of this Service with Lines shape, change the Type of queue to Serpentine. Use this option to simulate serpentine (also named "zigzag") queues.

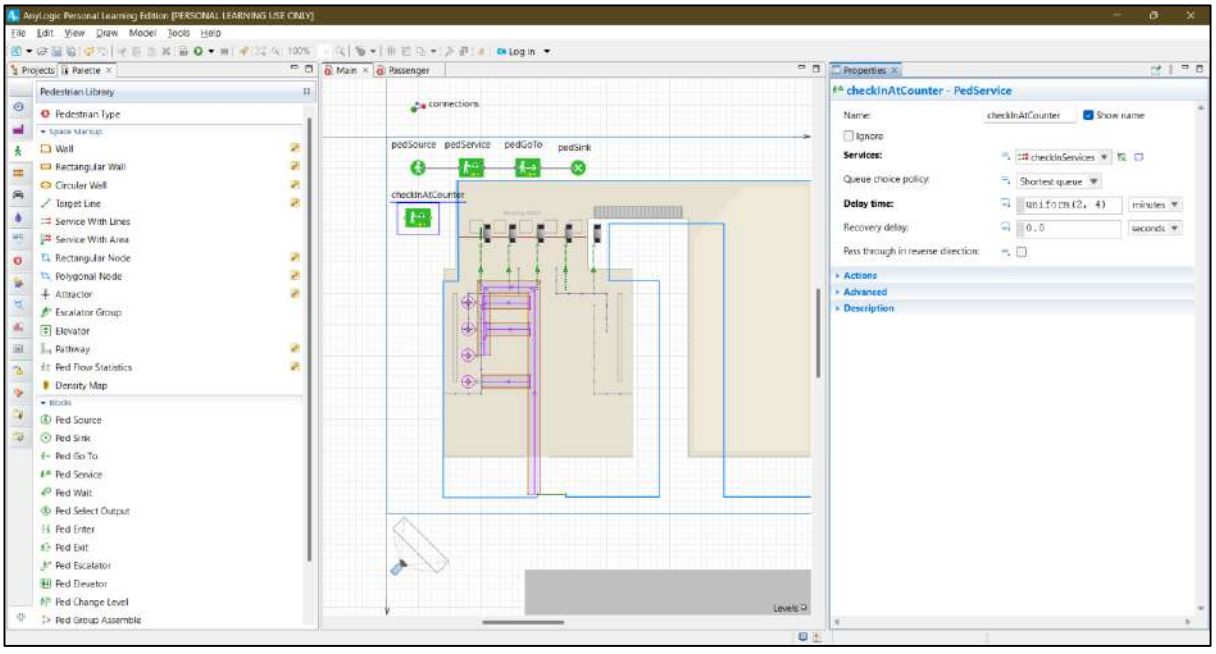
You will see that the queue has borders now. In 3D animation they will appear as belt barriers.



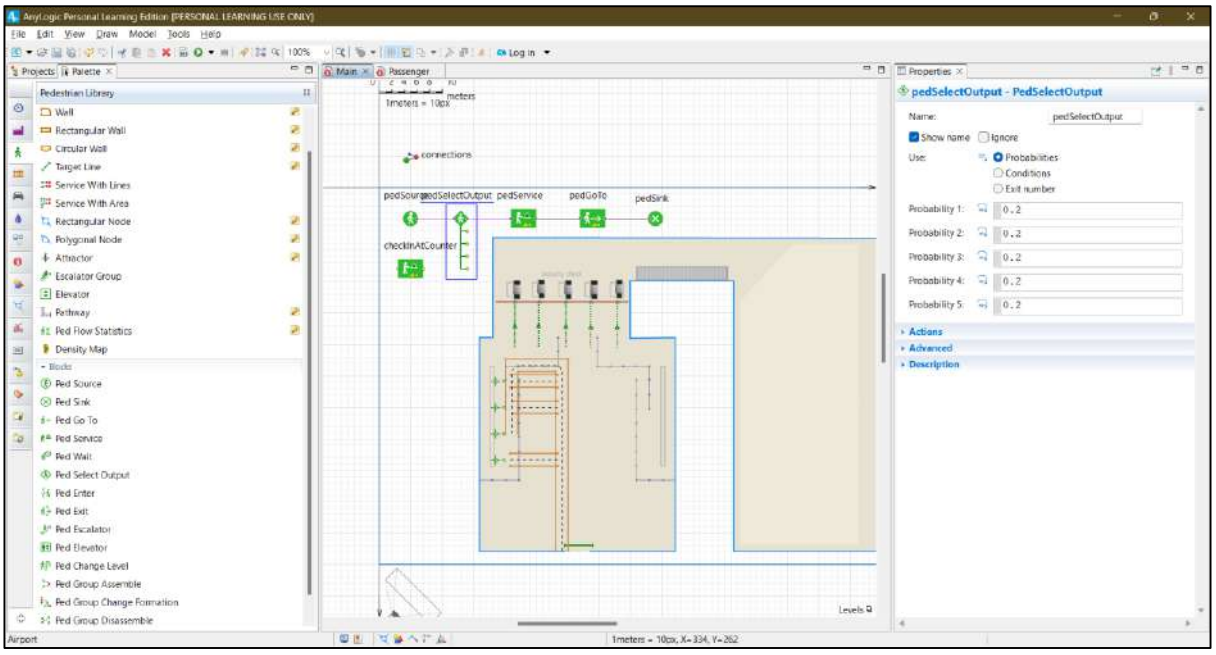
Add another PedService block and name it checkInAtCounter.

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

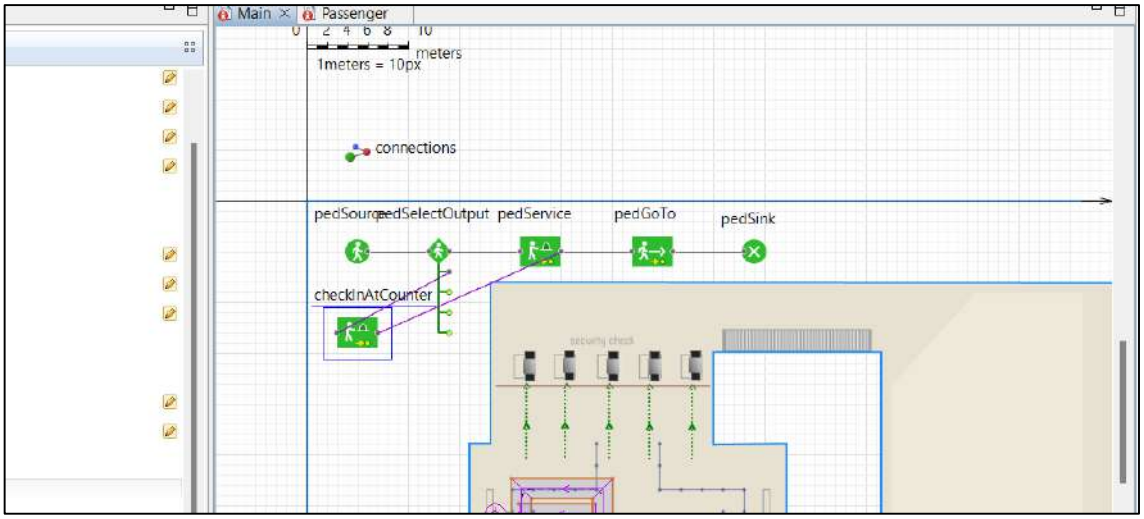
Since we assume it takes between 2 to 4 minutes to check in, type uniform(2, 4) minutes as the Delay time.



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



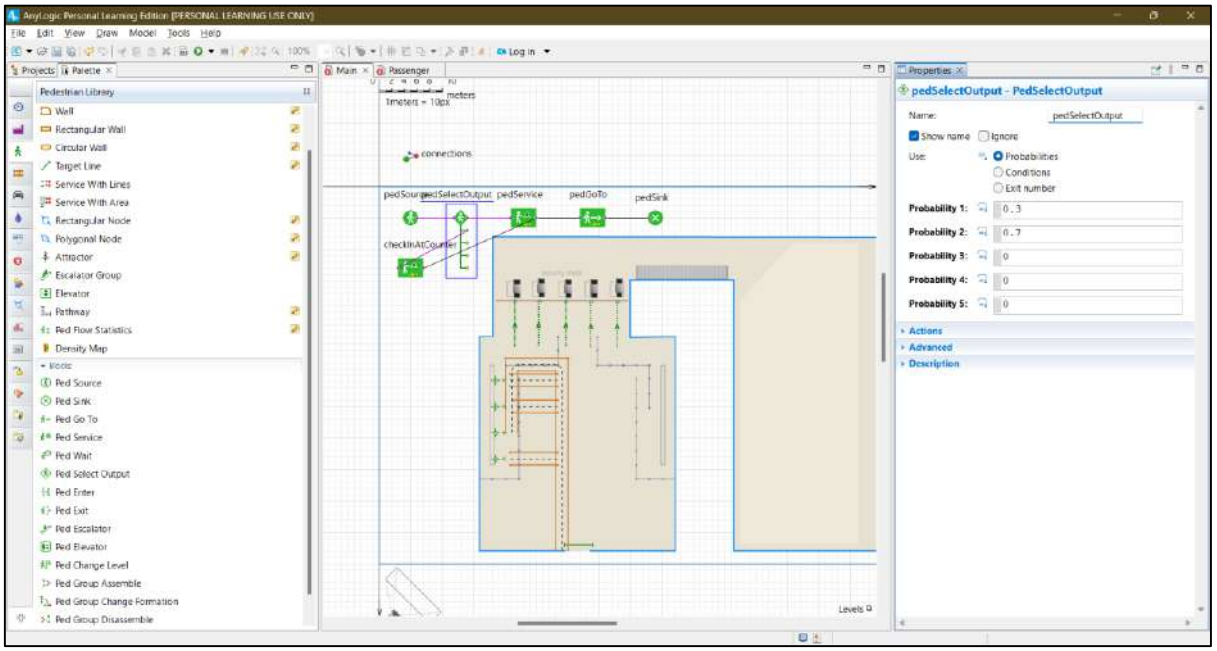
Connect the checkInAtCounter block to the existing flowchart blocks as shown in the figure below.



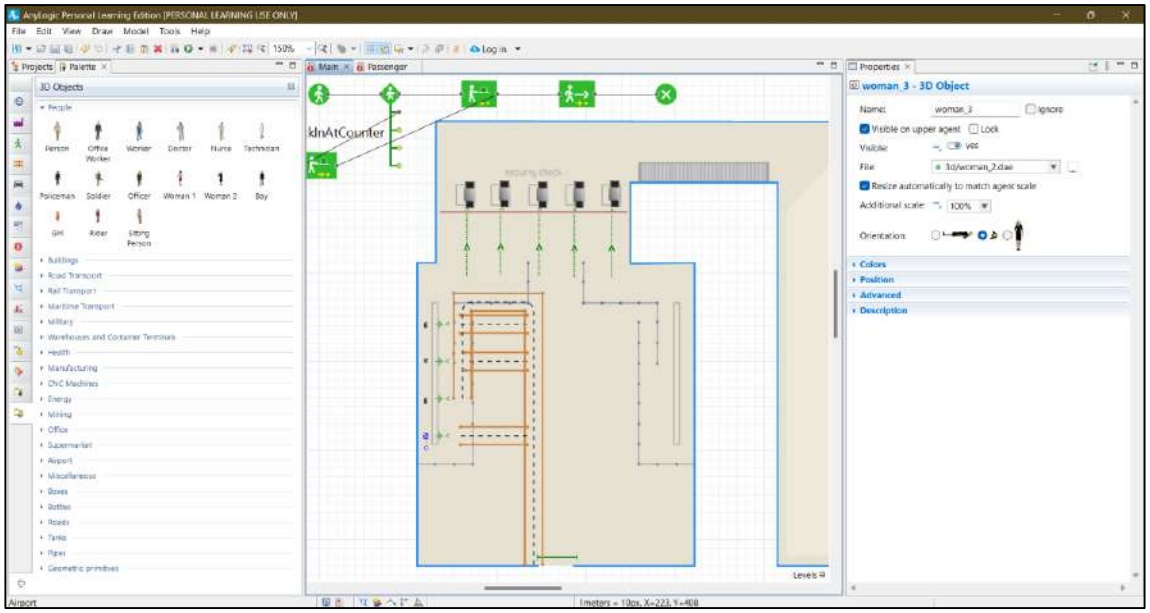
Since we're assuming 30 percent of our passengers will check in online and 70 percent will check in at

the counter, we'll model this behavior by setting pedSelectOutput's Probability 1: 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 AnyLogic 7 in Three Days 219 percent of the passengers to the lower branch. You must set Probability

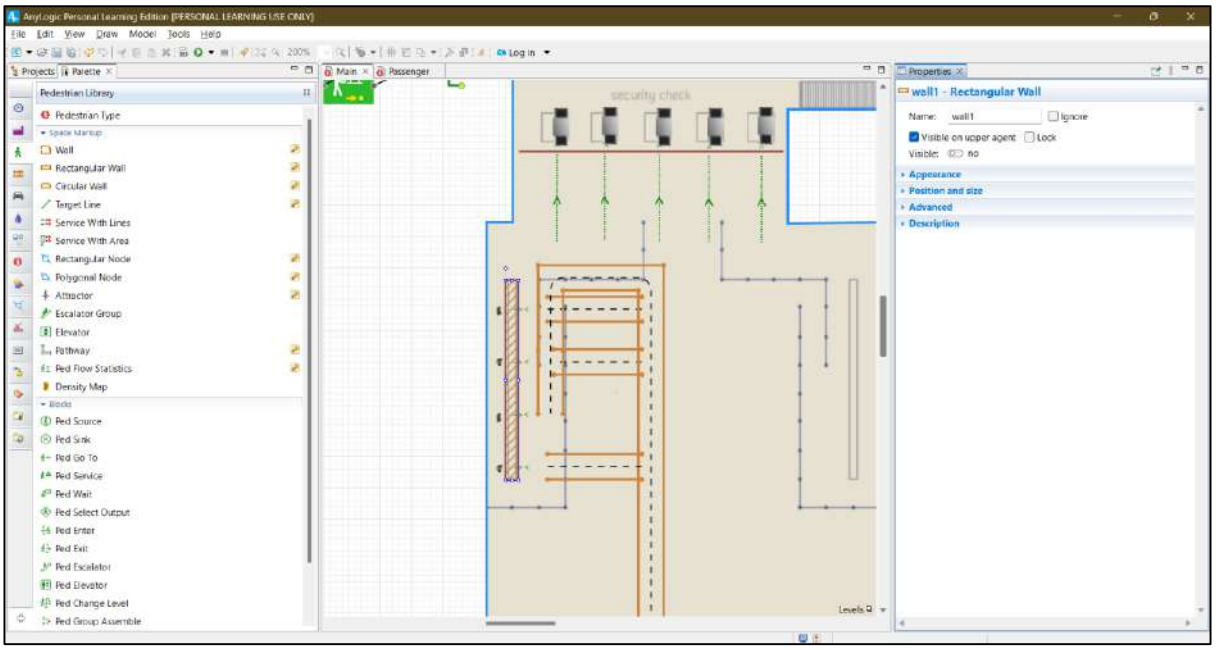
3, Probability 4, and Probability 5 to 0 to prevent AnyLogic from routing passengers to the block's lower three output ports.



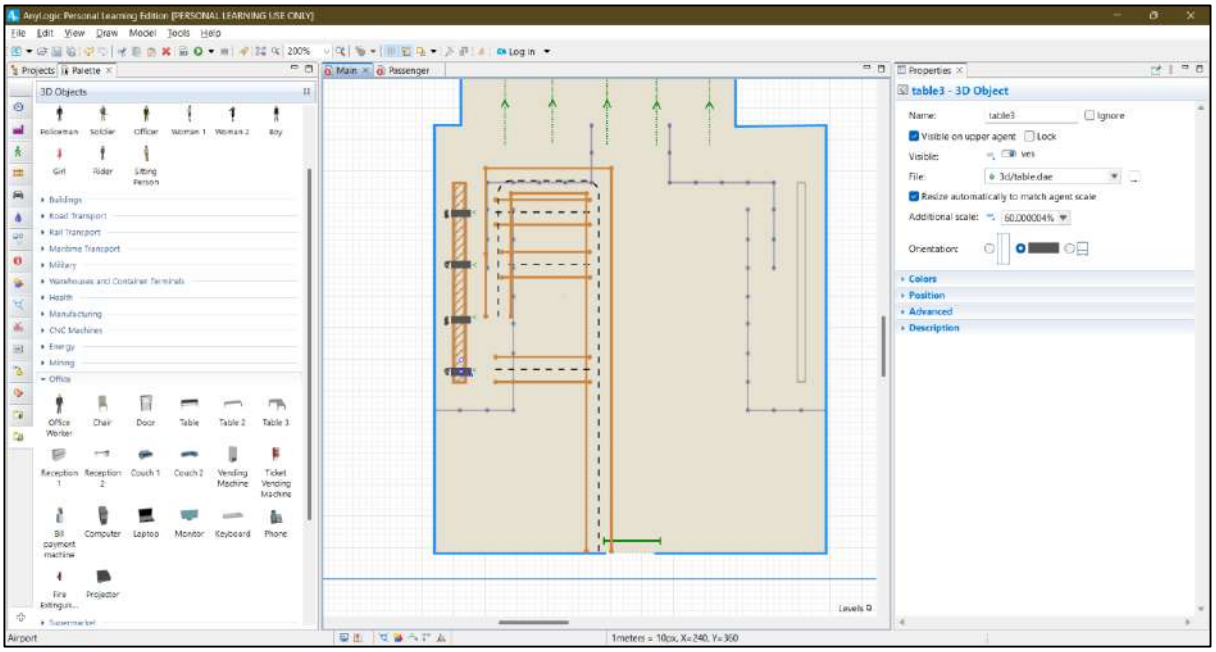
Let’s add ready-to-use 3D models to the airport’s check-in area. On the Palette's 3D Objects tab, expand the Peoplesection, and then add two copies of both Office Worker and Woman 2. to the diagram. As shown below.



Define the area that is not accessible by passengers. Add the Rectangular Wall element from the Space markup section of the Pedestrian Library palette and place it as shown in the figure below. In the wall's properties, set Visible to no to make this wall invisible at the model runtime.

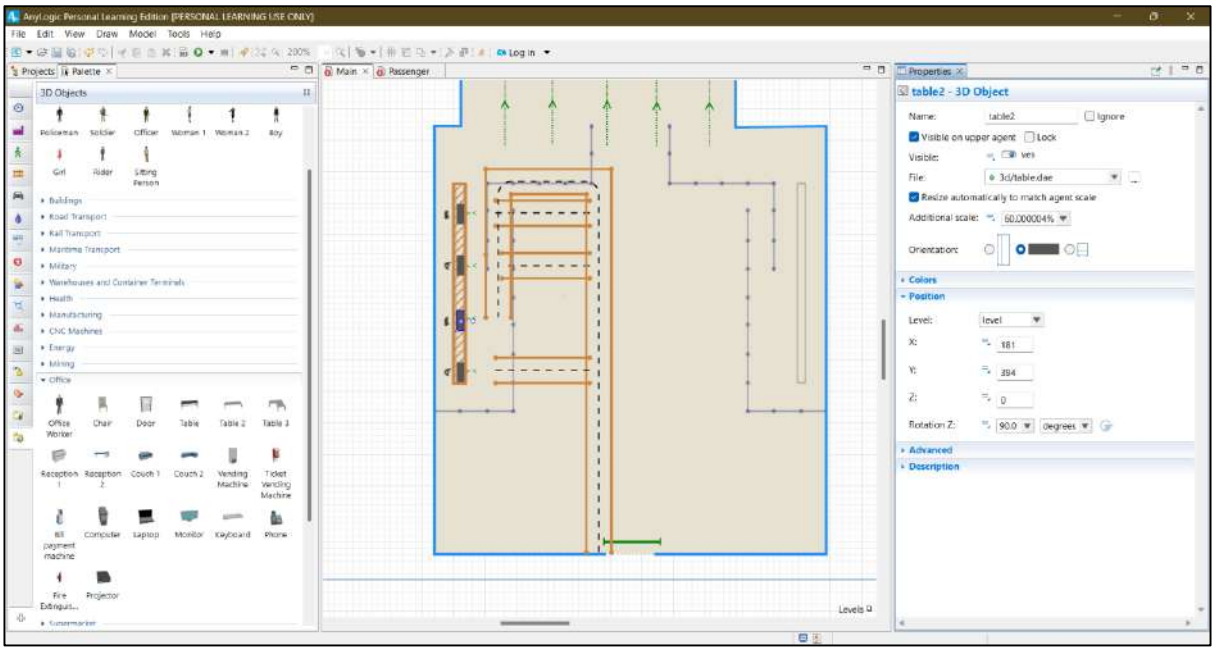


On the Palette's 3D Objects tab, select the Office section, and then drag four copies of the Table object on to the diagram.

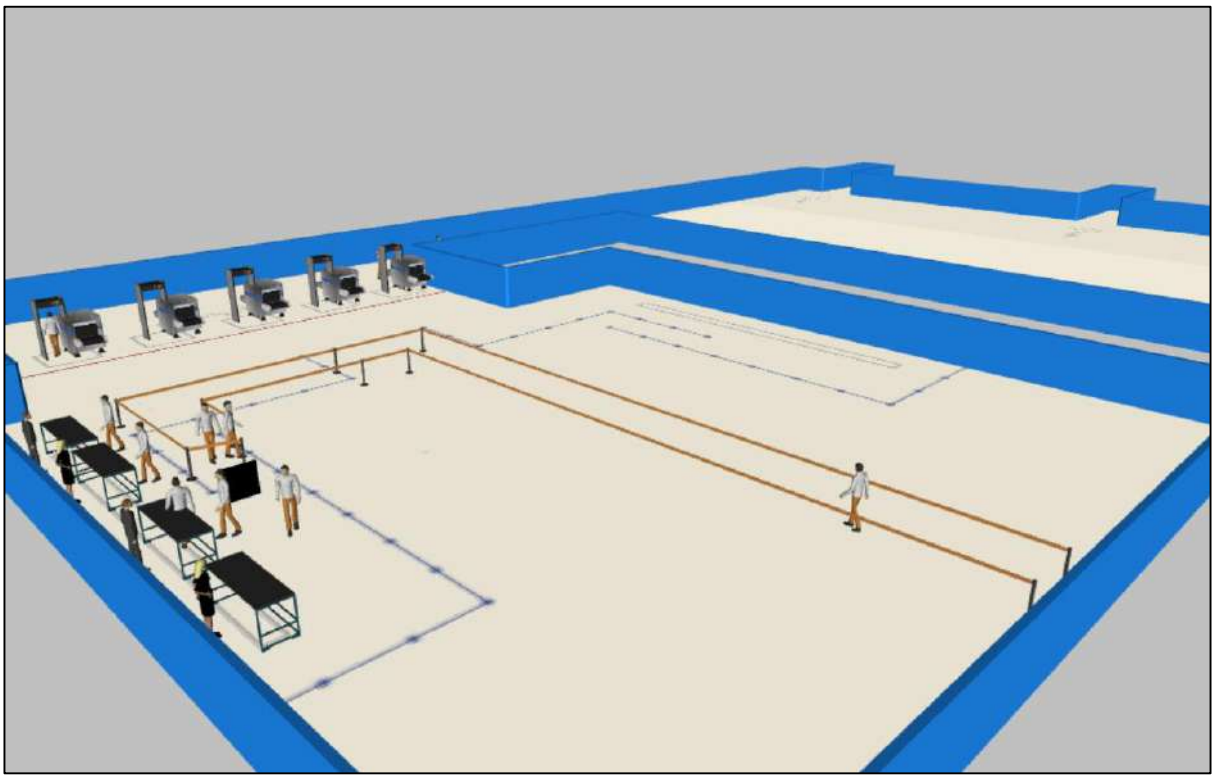


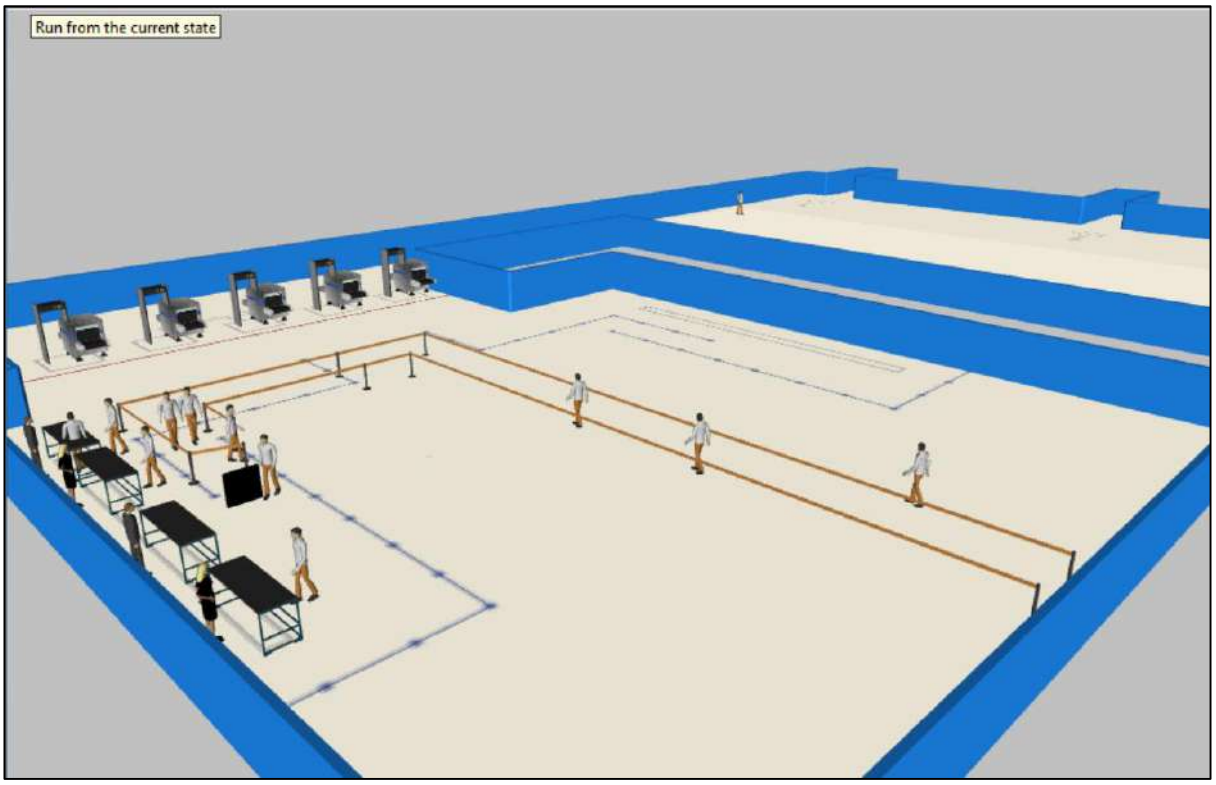
Since the tables aren't facing the correct direction, use their Properties section's Position area to set Rotation,

Z: 90 degrees.

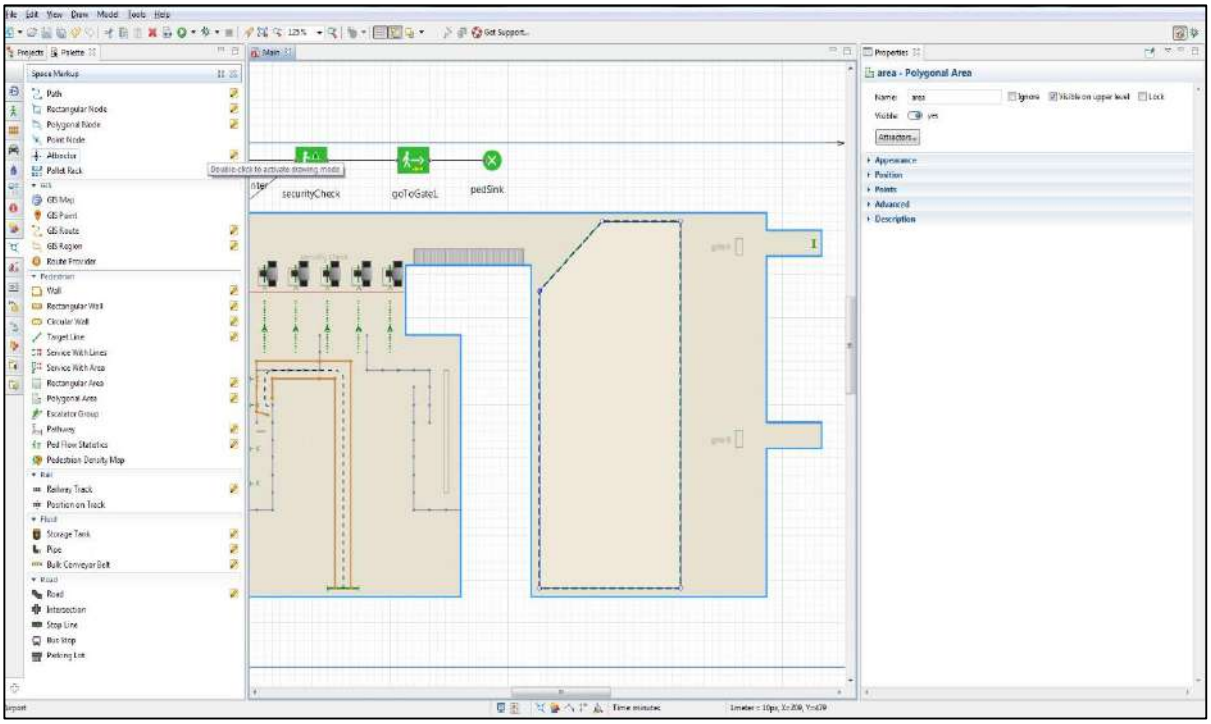


Run the model. You'll see some passengers check in and then go through the metal detector

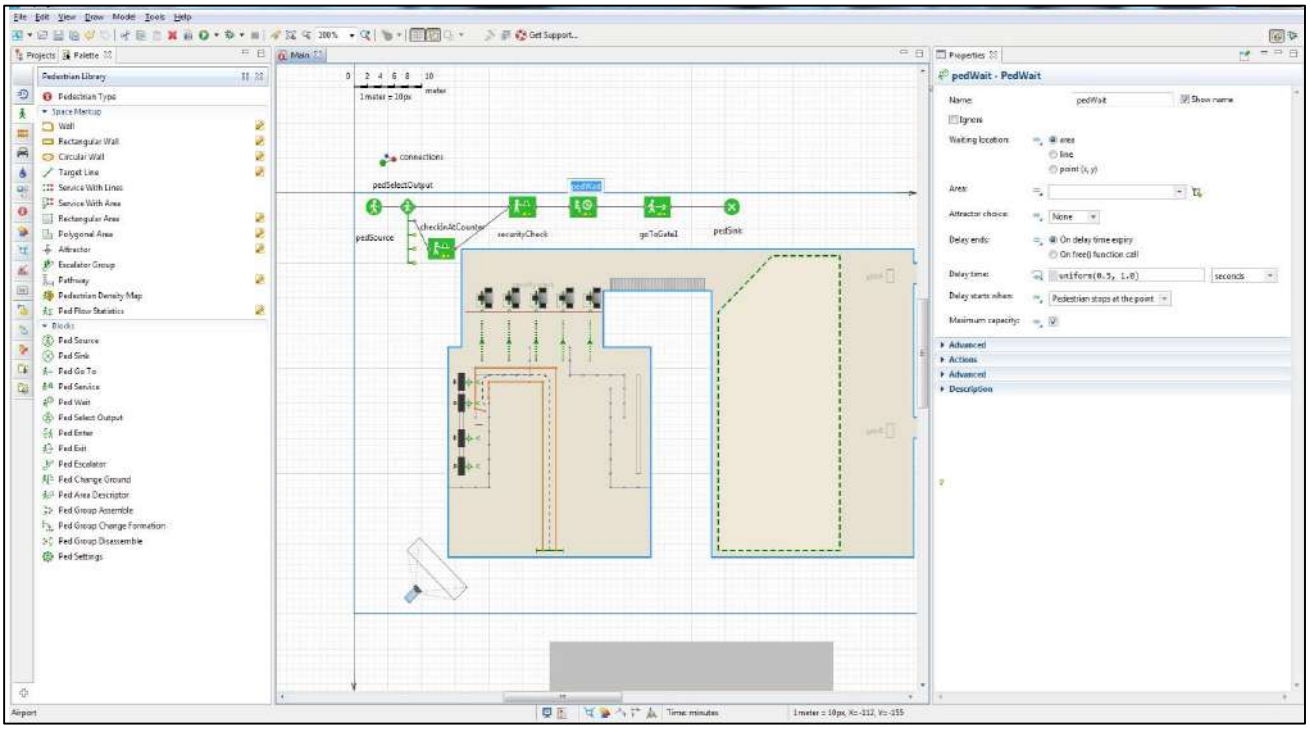




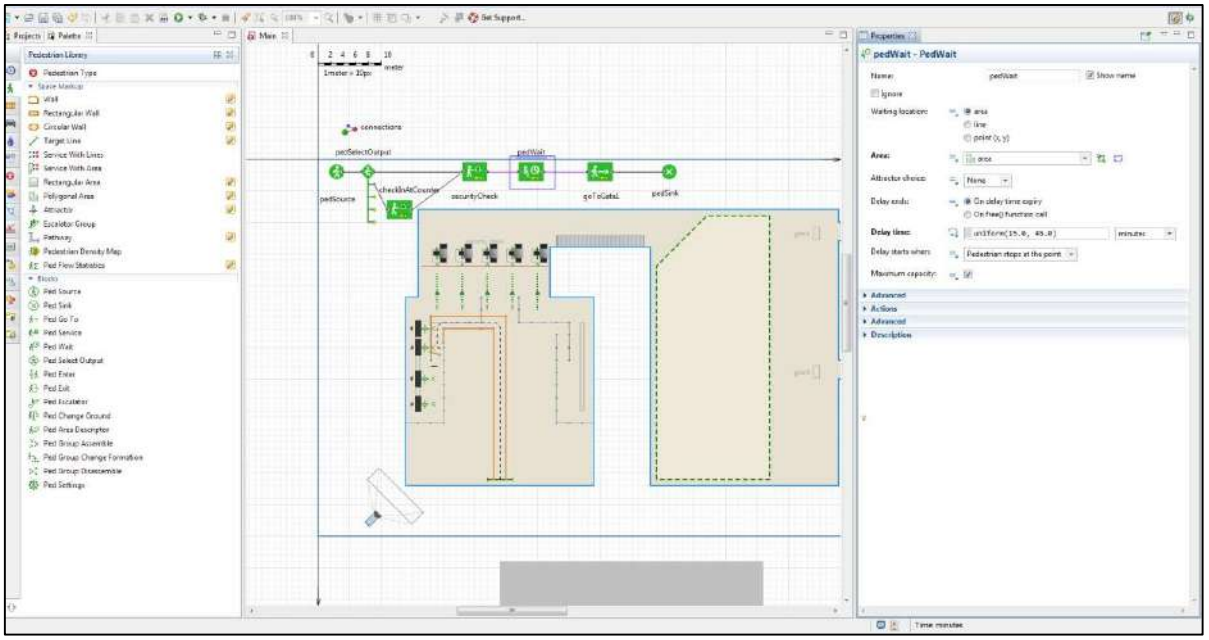
We want the passengers to wait before they go to the gate. To do this, we need to draw the waiting area where the passengers will wait and then add a flowchart block (PedWait) to simulate the waiting. Draw the waiting area before the gates using the Polygonal Area element from the Space Markup section 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, doubleclick your mouse.



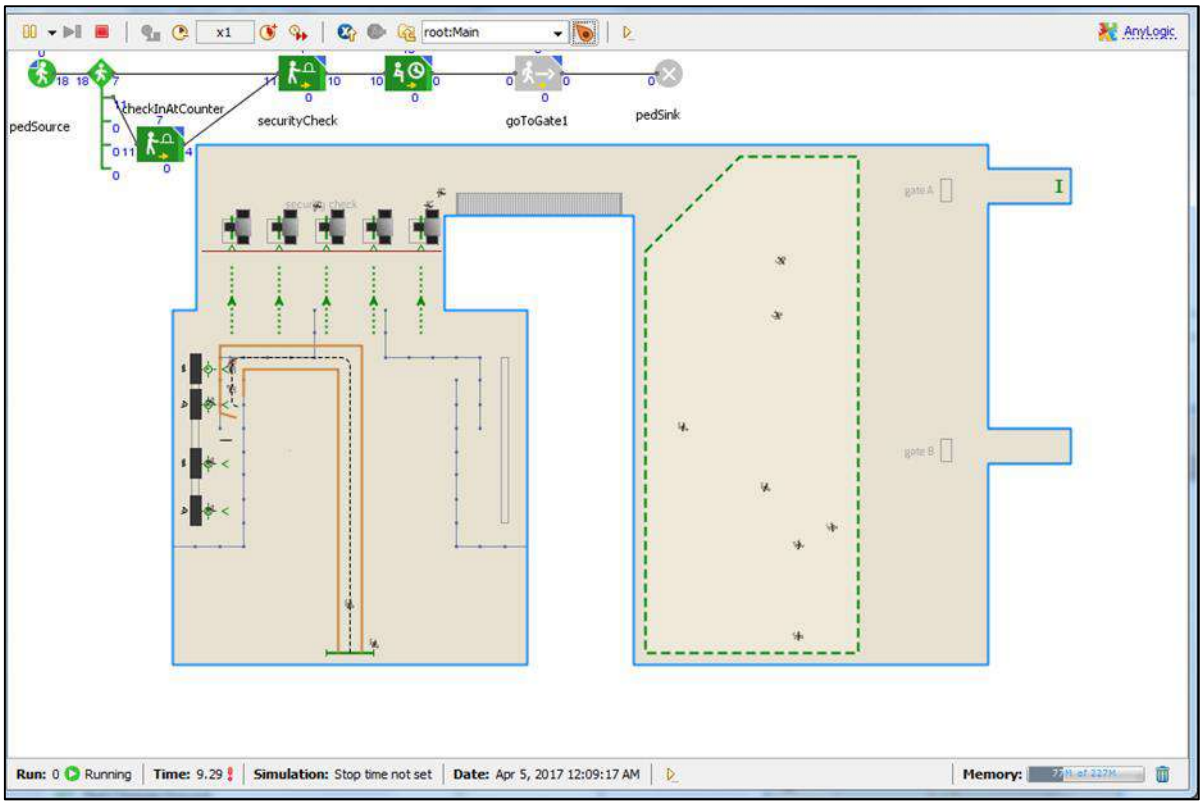
Add the **PedWait** block on to the flowchart between **PedService (securitycheck)** and **PedGoTo**.(gotogate1) as shown below

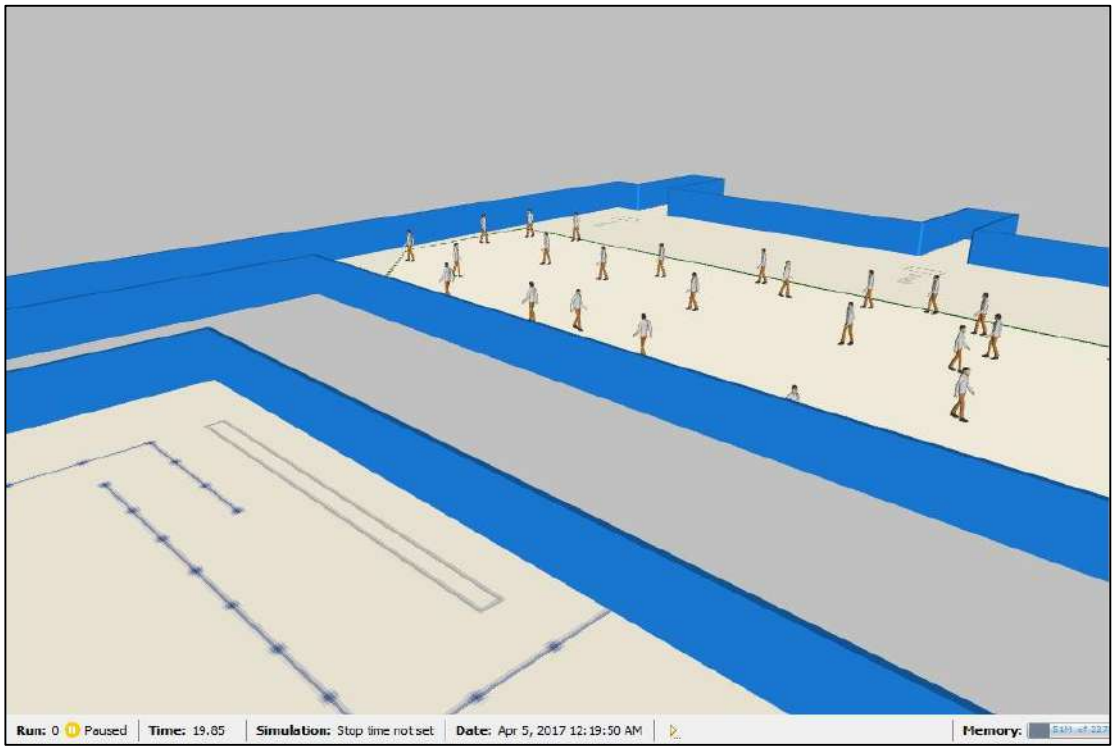


Modify the block's properties by selecting the area from the **Area** list, and then setting the **Delay time** to uniform(15, 45) **minutes**.



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

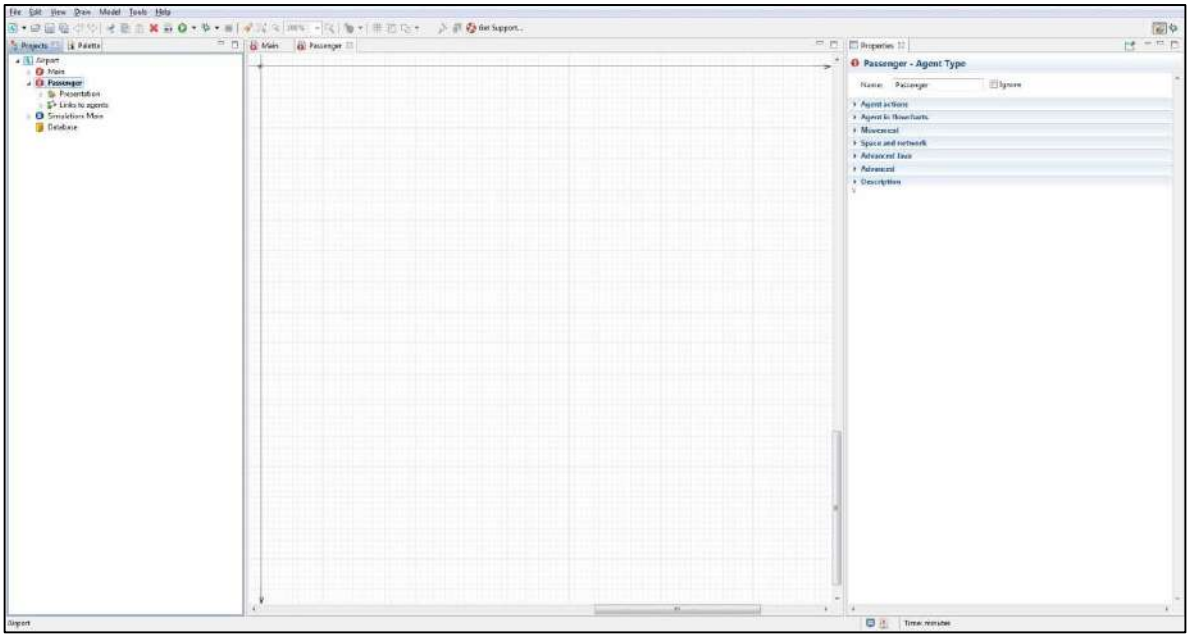




You can add more check-in facilities on the right and configure the **PedSelectOutput** to separate the pedestrian flow to more branches.

Defining the boarding logic

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

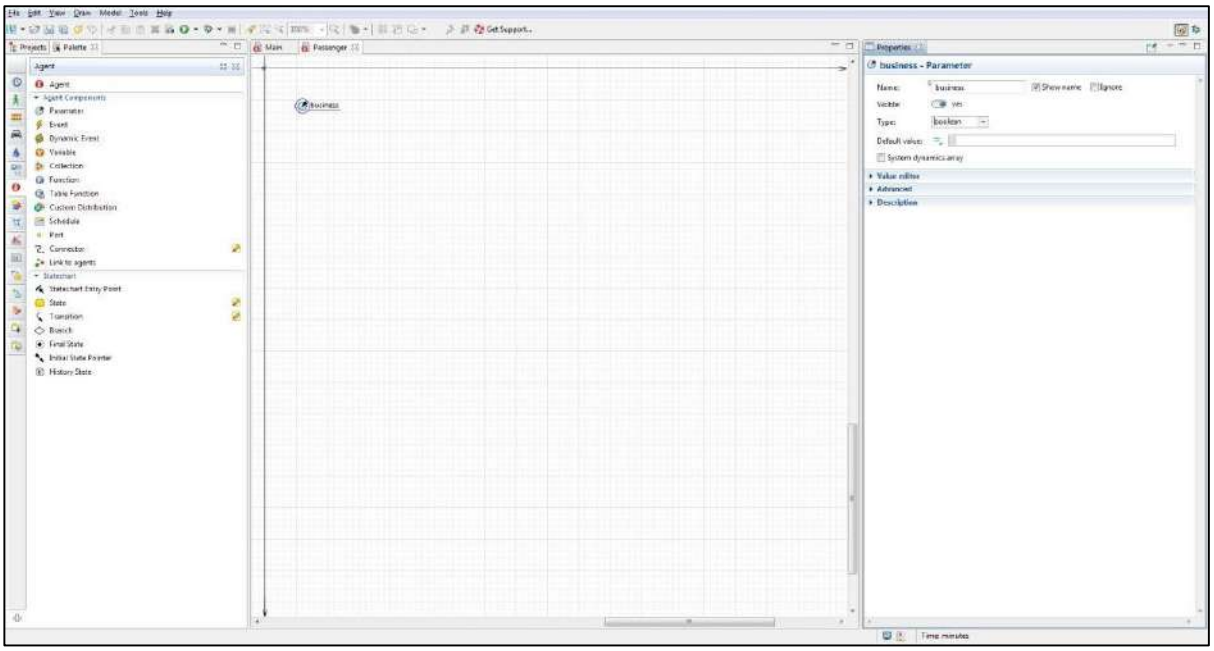


Add a **Parameter** from the **Agent** palette to define the passenger's class.

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 class passenger.

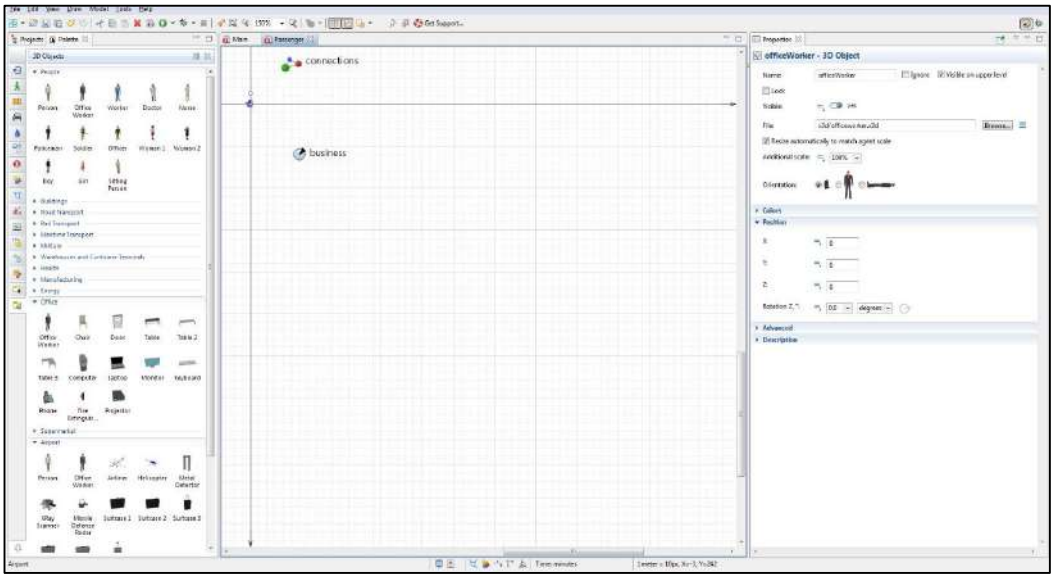
We want to distinguish passengers in 3D animation, namely animate business class and economy passengers with different 3D models. To do this, we'll use the existing Person 3D object to represent economy passengers and add another 3D shape to represent business class passengers.

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.

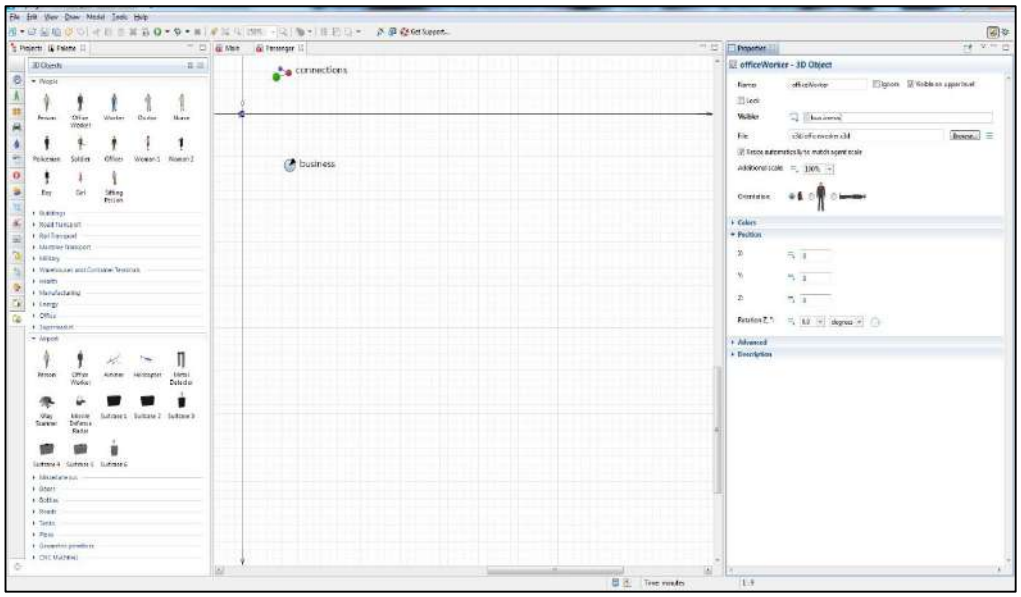


We can distinguish passengers in 3D animation, namely animate business class and economy passengers with different 3D models. To do this, we'll use the existing Person 3D object to represent economy passengers and add another 3D shape to represent business class passengers.

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.

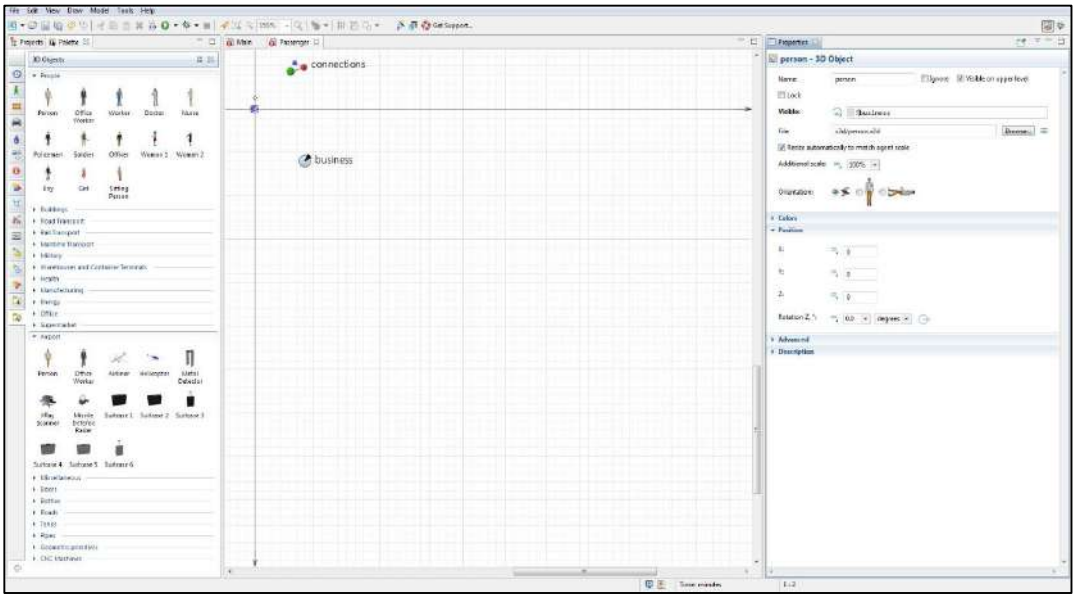


Change the visibility of these objects. First, click the Office worker shape. We want this shape to be visible only when this is a business class passenger, that is, when its business parameter is true. Switch to the **Visible** property's dynamic value editor by clicking the icon to the right of the **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.



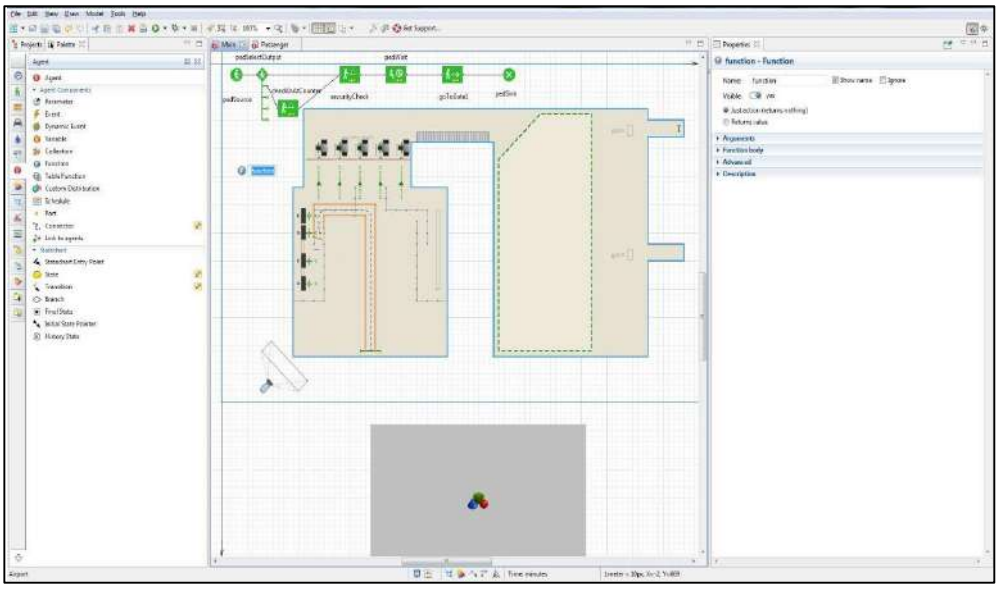
Now select the person 3D object (you can do this from **Projects** tree), and set **Visible**:
! business . This shape will be visible only if the passenger is an economy passenger.

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 setupPassenger.



Configure the function as follows:

- Create one argument in the agument drop down list to pass the newly-created passenger to the function:

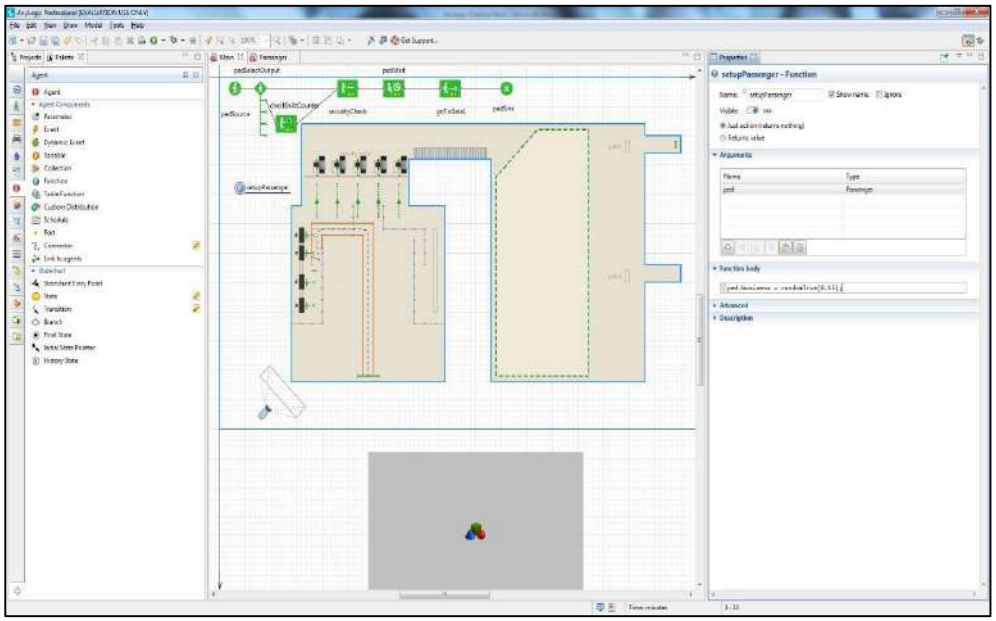
Name: ped

Type: Passenger

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

in the function body list write this

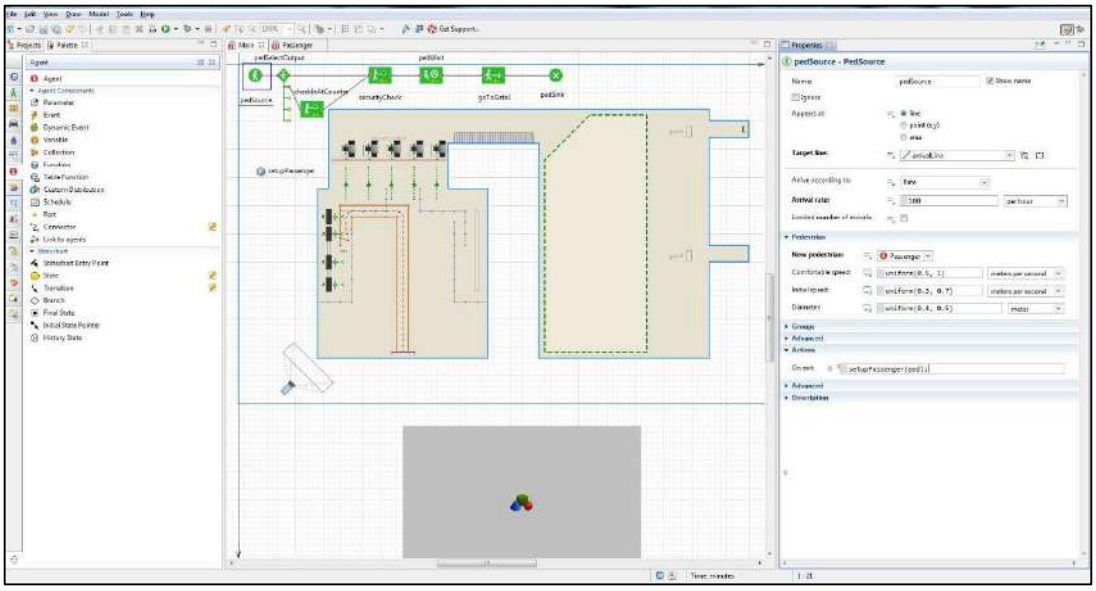
ped.business = randomTrue(0.15);



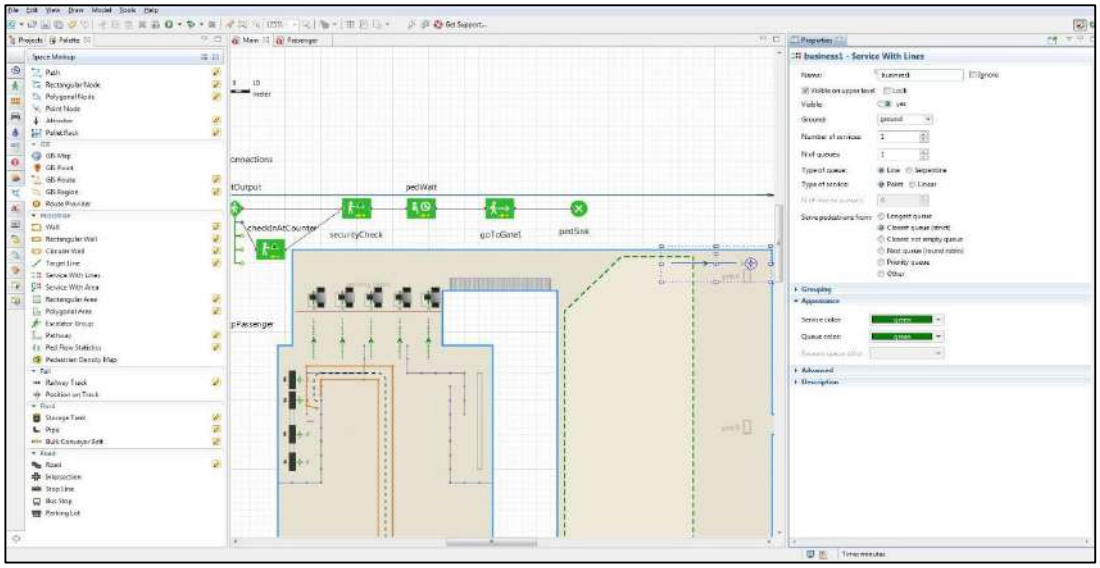
In this case, ped is the function's argument, of type Passenger. Having set up the Passenger as the argument type, we can directly access the custom pedestrian field business simply as ped.business. The function randomTrue(0.15) returns true

in an average of 15 percent of cases, which means an average of 15 percent of our model's passengers will travel in business class.

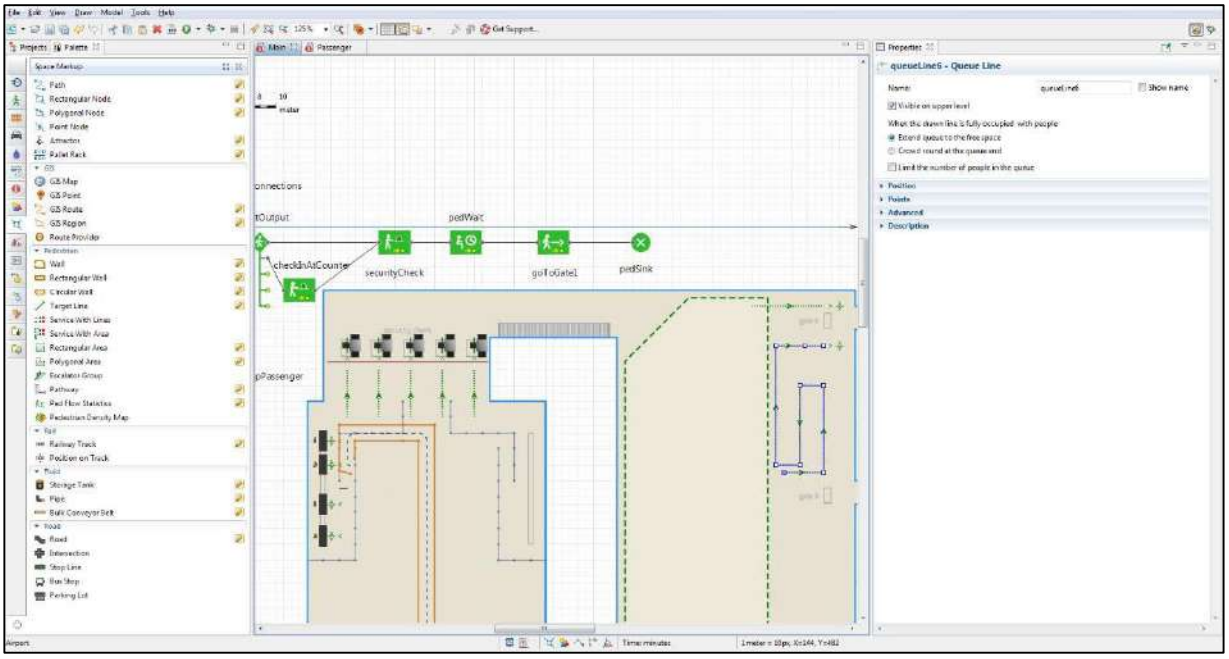
Call this function when a new pedestrian is created by the pedSource block. In the pedSource properties area, click the arrow to expand the **Actions** section, and then enter the following code in the **On exit** box: setupPassenger(ped);



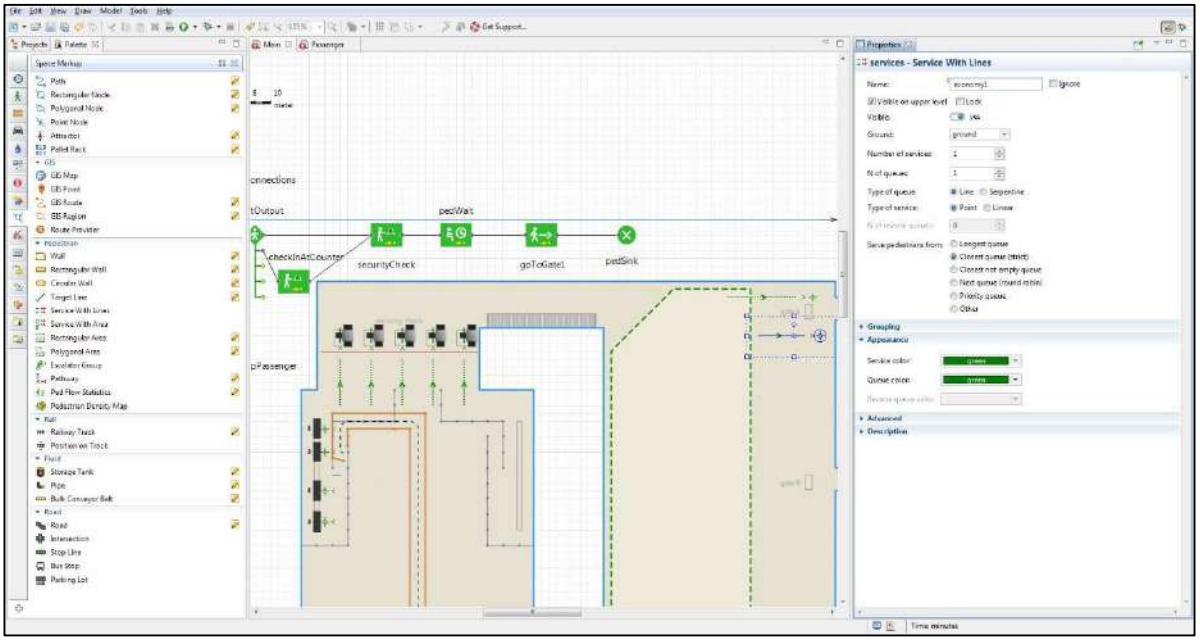
Here we call our function setupPassenger for the newly-created pedestrian. This pedestrian is passed to the function via its argument.
Draw two services with lines for the upper gate, one for business class and one for economy passengers.
Draw a Service with Line, defining the priority line (point service, 1 service, 1 line).
Name this service business1



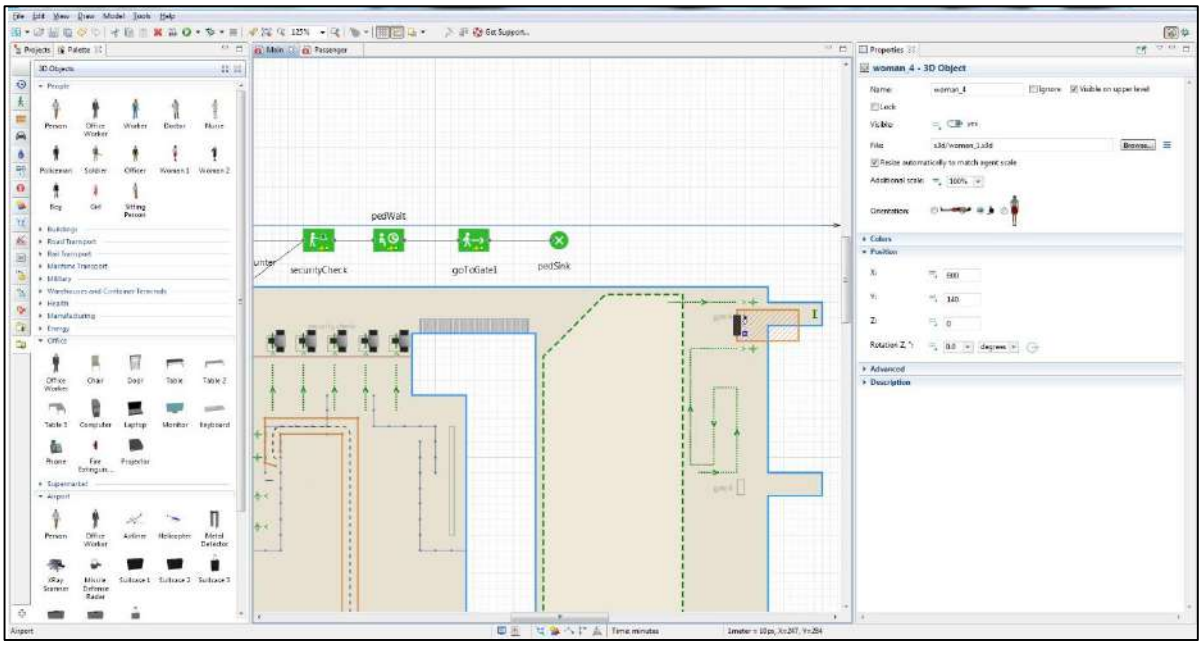
Add one more Service with Line, and name it economy1



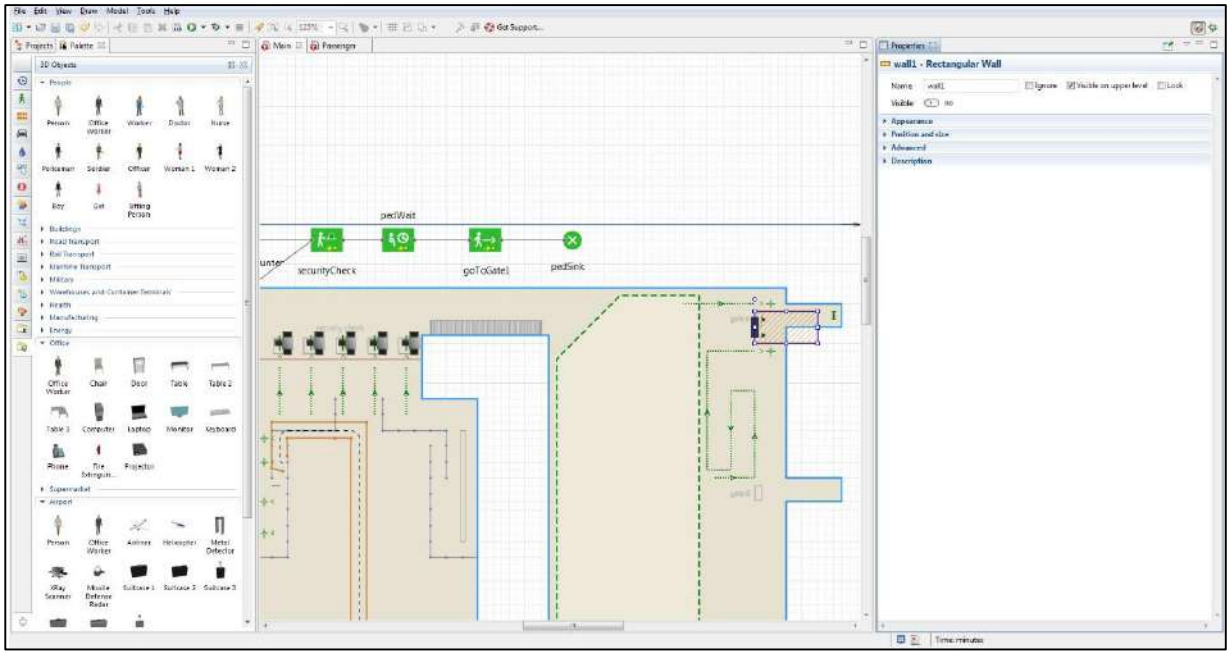
Add more point to the queue



Draw an area at the gate with the **Rectangular Wall** element, from the apce markup and then add a table from 3d office & rotate it 90 degree as per display and add two 3D woman models at the table.

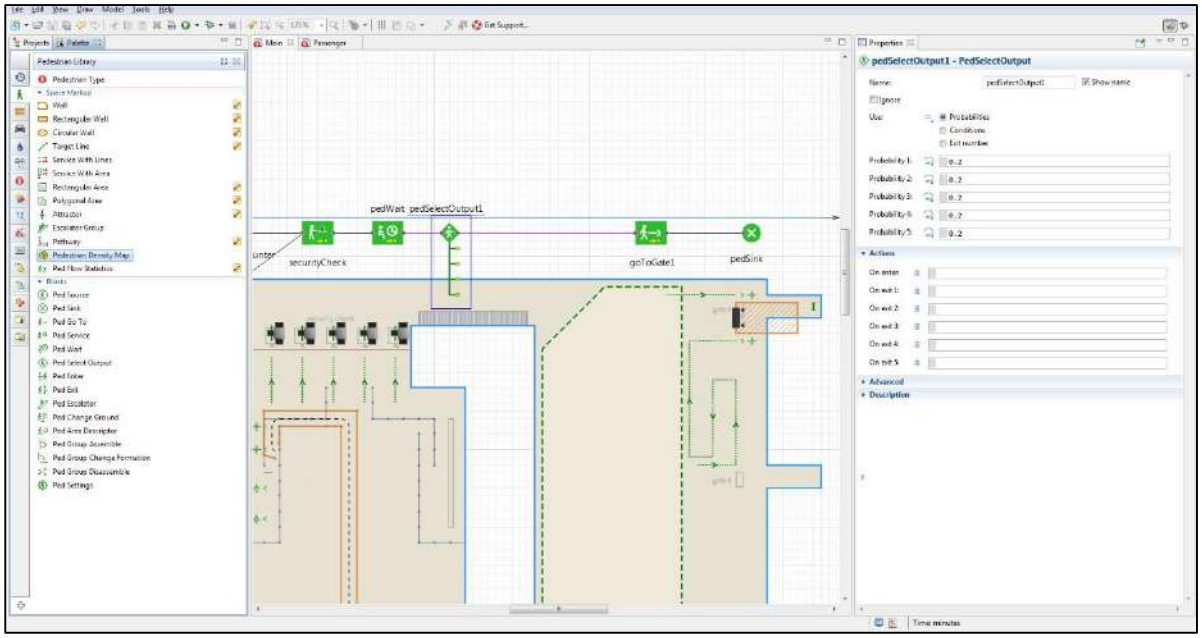

pedWait
securityCheck
goToGate1
pedSink
woman_4 - 3D Object

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

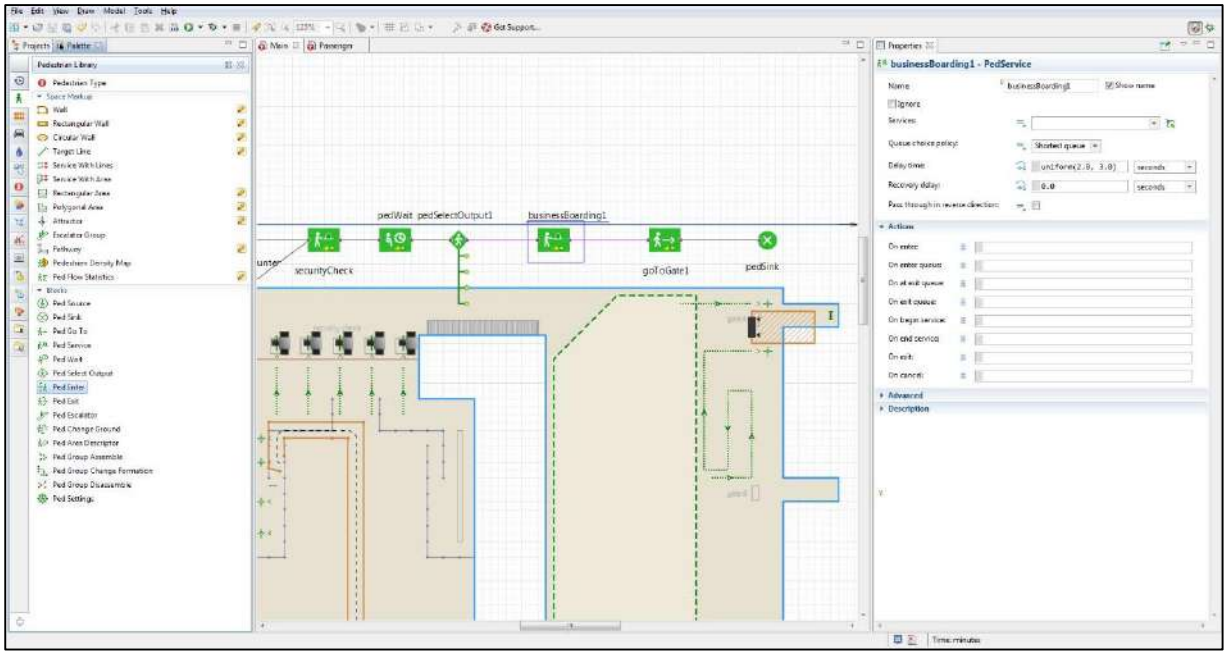


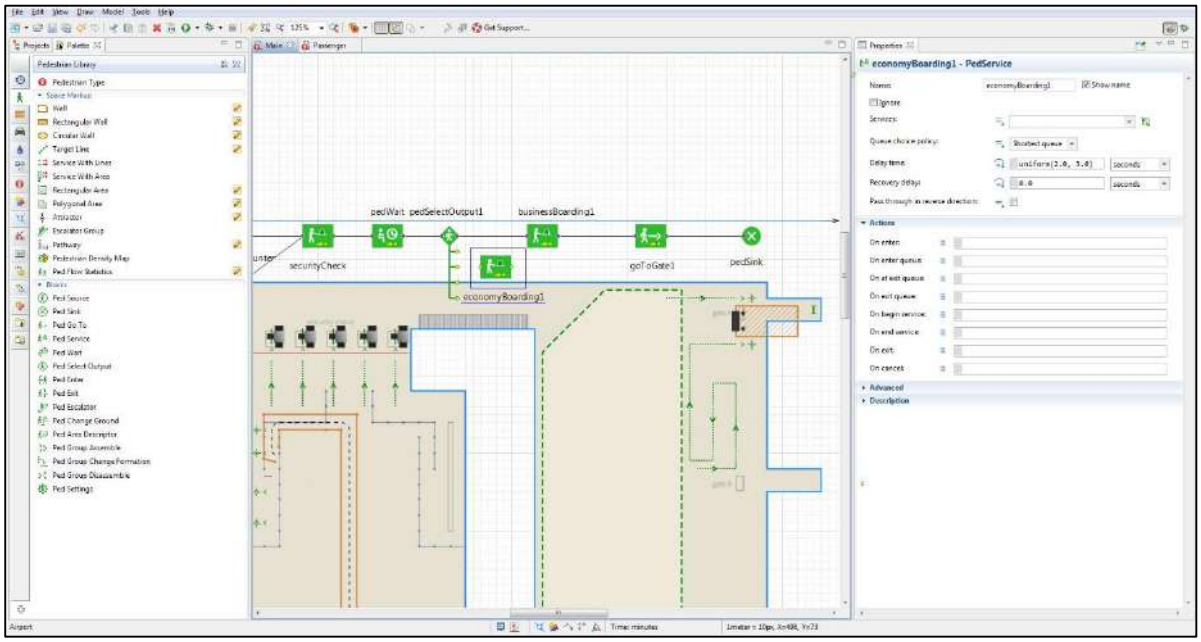
Insert the blocks into the flowchart between the pedWait and goToGate1 objects.

Add **PedSelectOutput** to route business class and economy passengers to different lines. As shown below

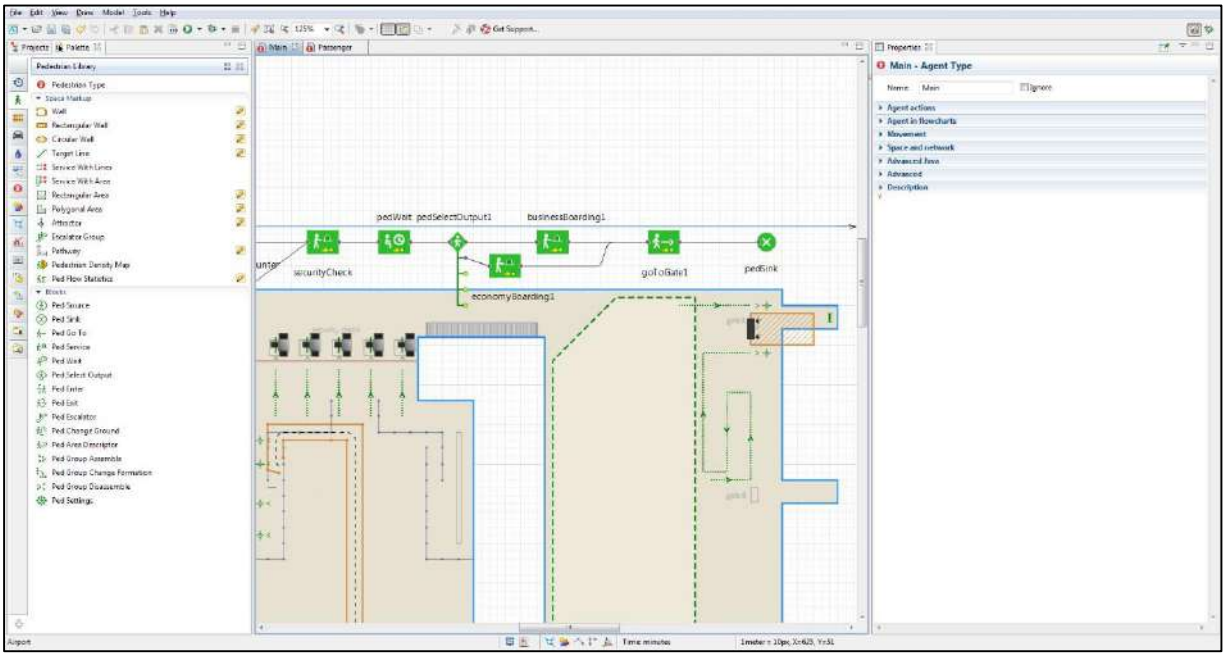


Add two **PedService** blocks: businessBoarding1 and economyBoarding1 to simulate the process of checking passengers' tickets at the gate.



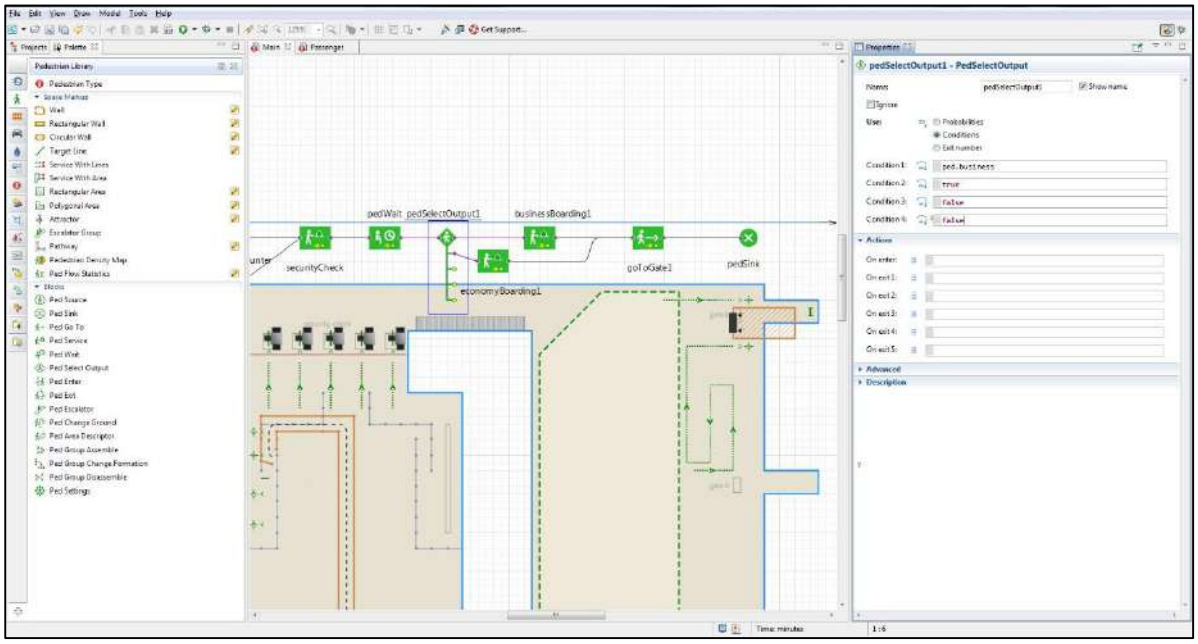


Connect it as shown in the figure below

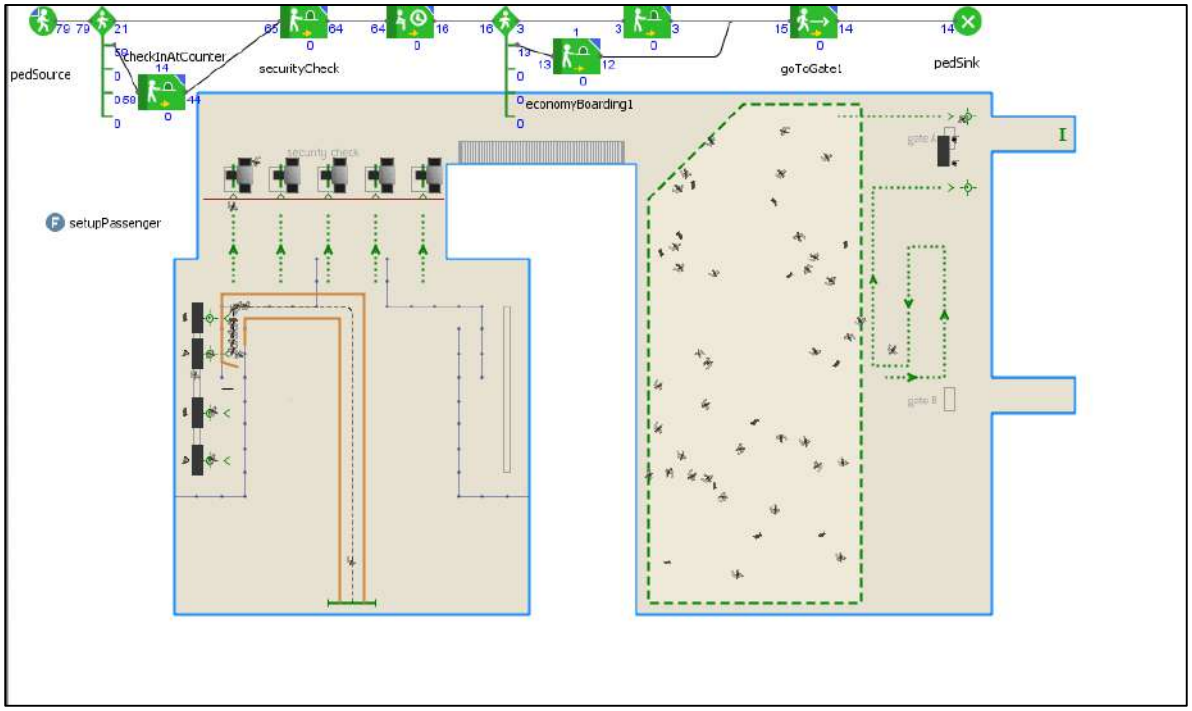


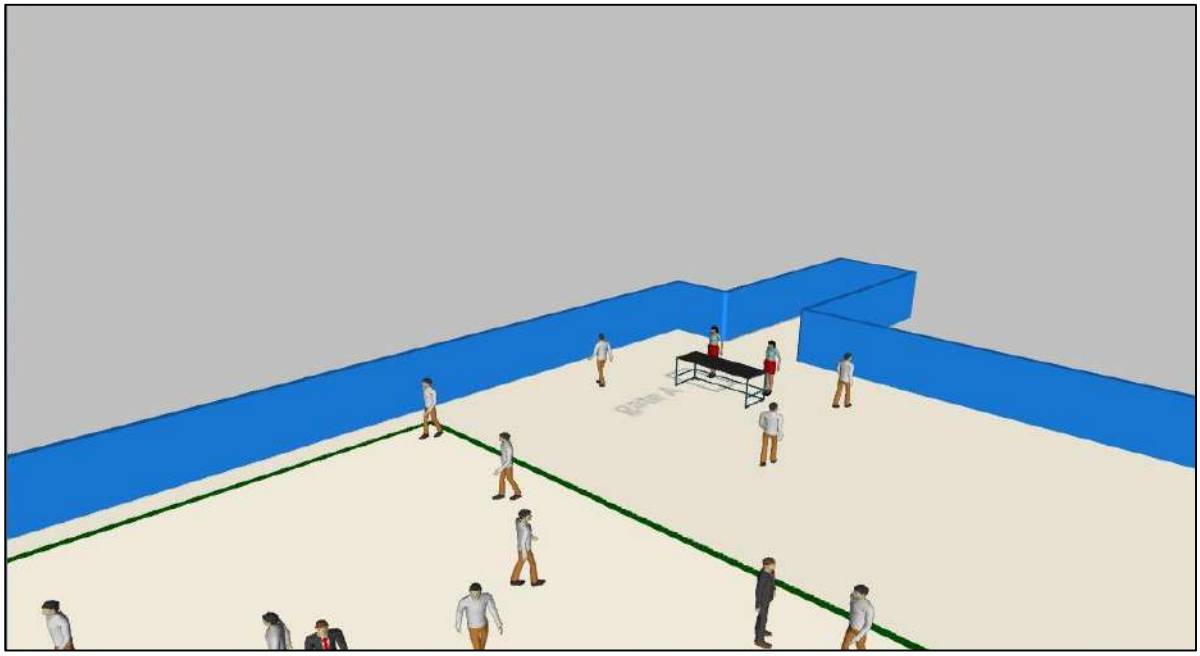
Since the **PedSelectOutput** block routes business class and economy passengers to different lines, select **Use**: **Conditions**, and then type ped.business in the **Condition 1** box. This expression will return true for all business class passengers, which means they will follow the upper flowchart 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.





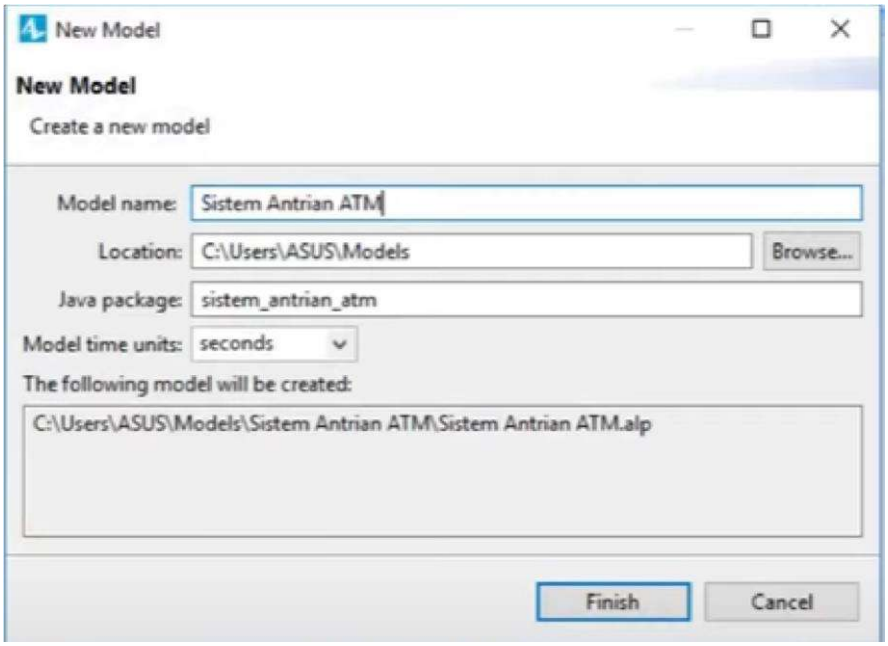
Practical No: - 7

Aim: - Verify and validate a model developed like bank model or manufacturing model

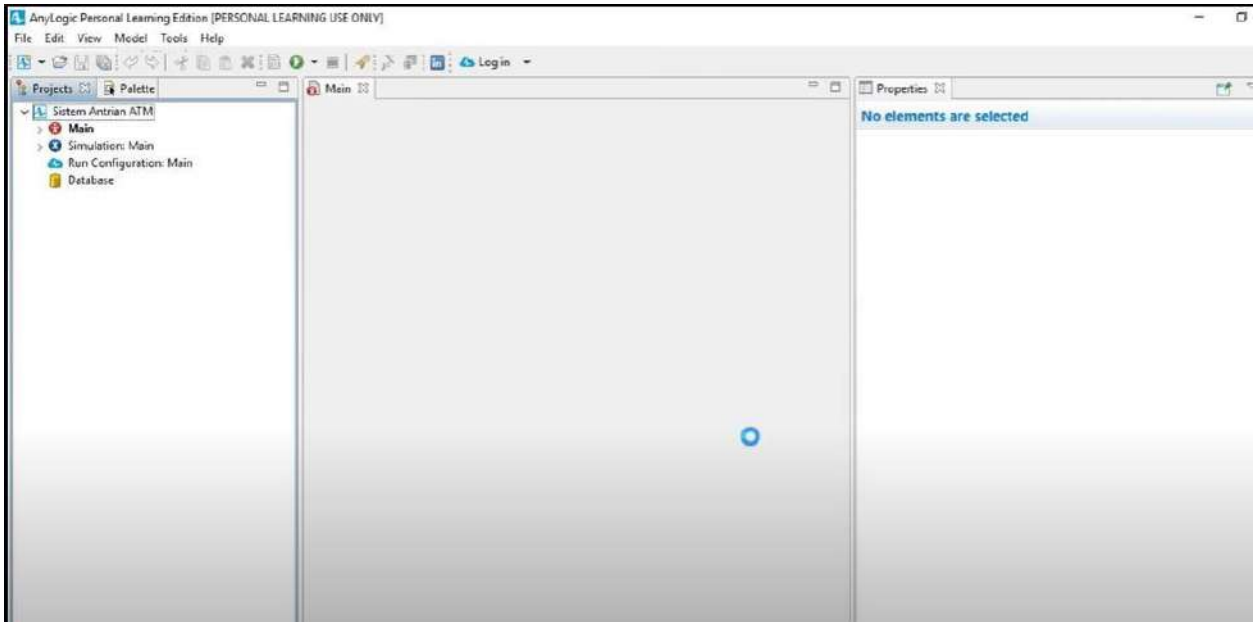
Code:-

Create a new model

1. Click the **New** toolbar button. The **New Model** dialog box is displayed.
2. Specify the name of the model. Type Bank in the **Model name** edit box.



3. Specify the location where you want to store your model files. Browse for the existing folder using the **Browse** button, or type the name of the folder you want to create in the **Location** edit box.
4. Select **minutes** as model time units.
5. Click **Finish** to complete the process.



Creating a process flowchart

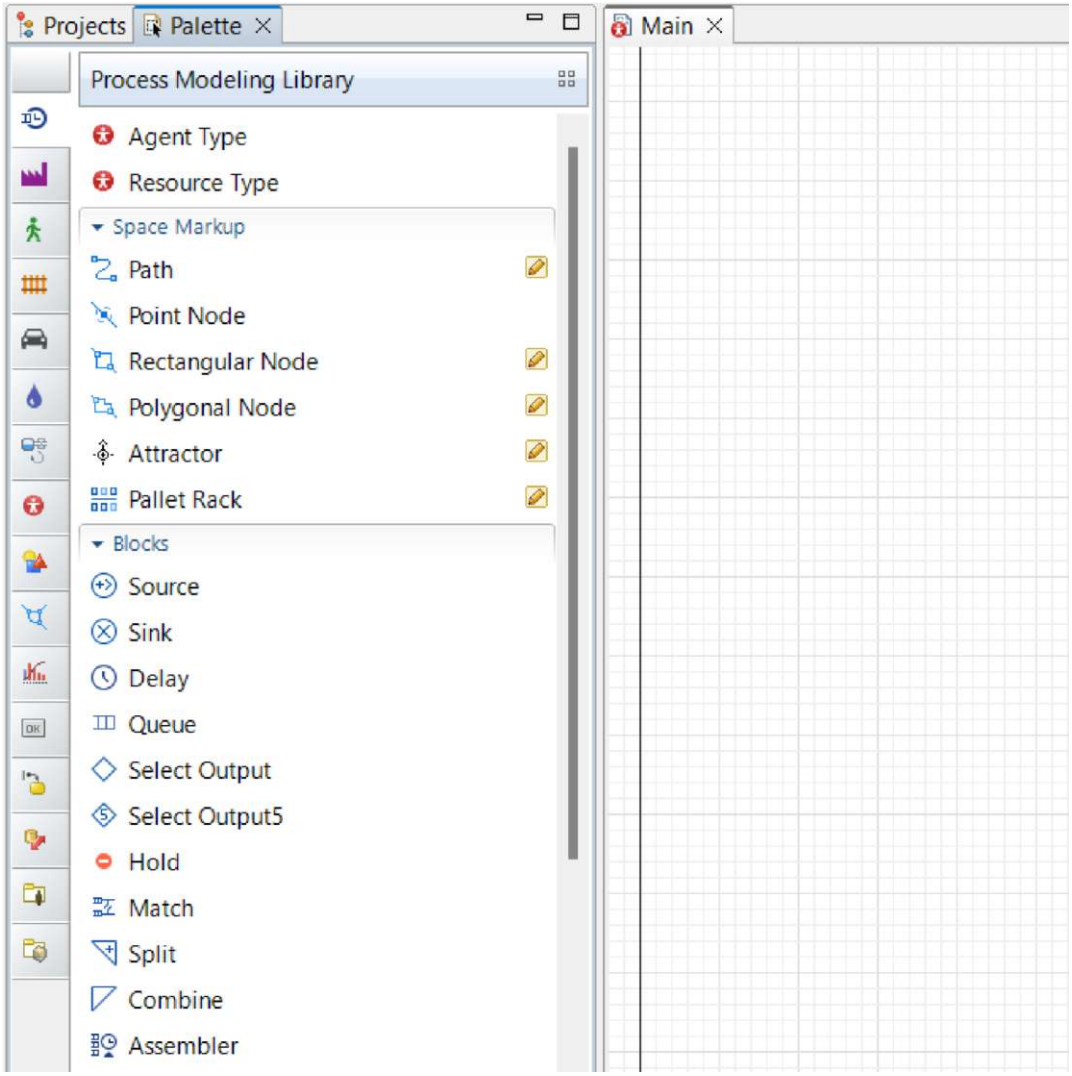
Now we will define the process with a flowchart composed from Process Modeling Library blocks.

Each block here defines some operation that will be performed with agents passing through this block.

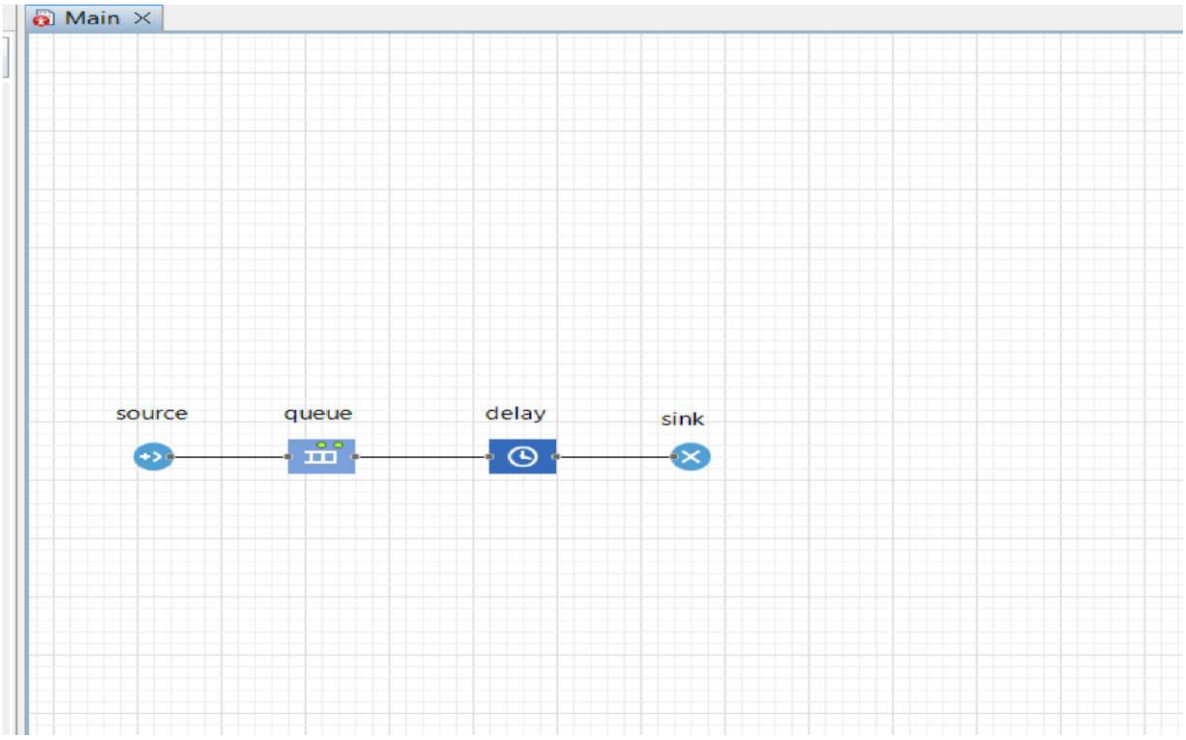
In AnyLogic you create flowcharts by adding the blocks from the library palette to the graphical diagram, connecting blocks together and tuning the parameters of the blocks.

Create the model flowchart

The **Palette** view and display the **Process Modeling Library** palette:



1. Add **Process Modeling Library** blocks on the diagram and connect them as shown in the figure below. To add a flowchart block on the diagram, drag the required element from the palette into the graphical editor.



While dragging the blocks, and placing them close to each other, you may see the lines connecting the blocks appear. These connectors should connect only the ports lying on right and left borders of the block icons.

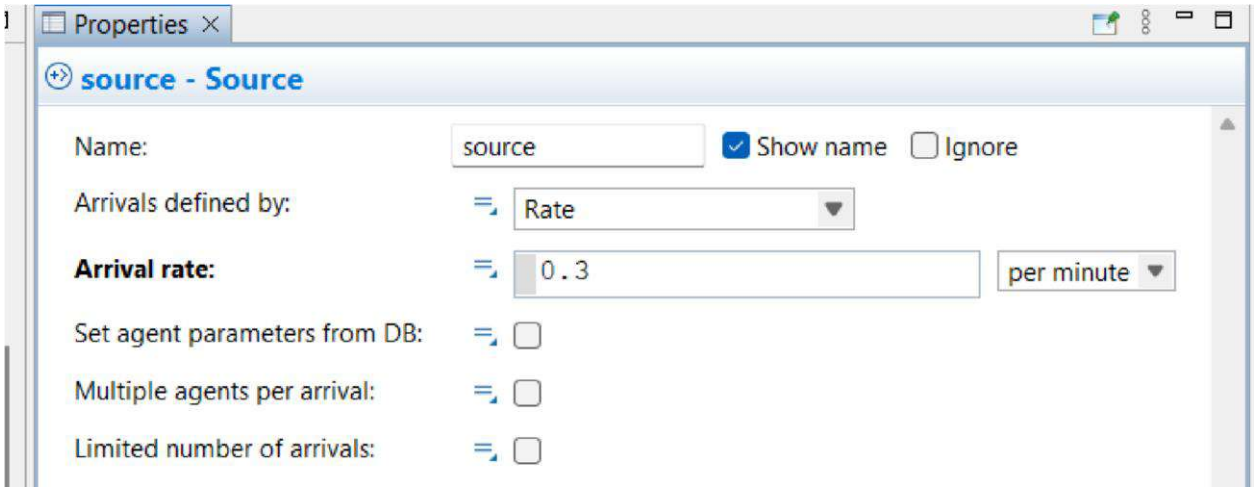
Our flowchart describes the simplest queuing system, consisting of a source of agents, a delay, a queue before this delay, and the final sink block.

Let's say a couple of words about these flowchart blocks.

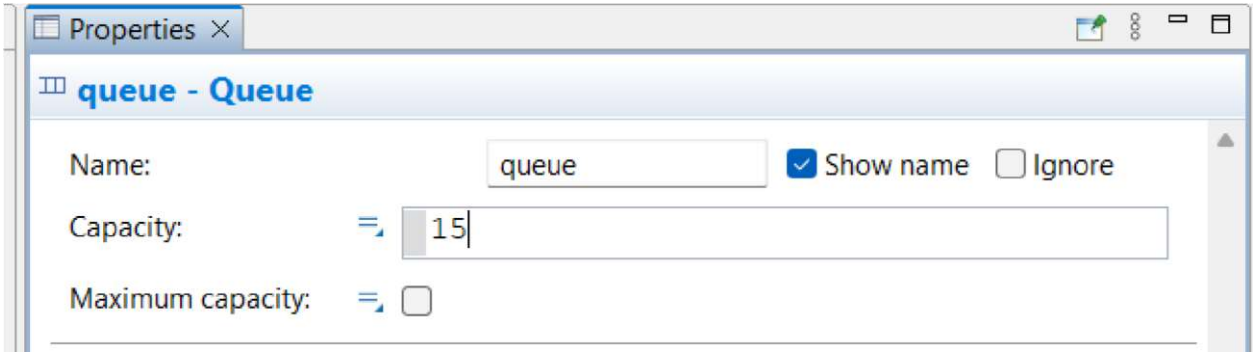
- **Source** block generates agents. It is usually used as a starting point of the process flow. In our example, it models customer arrival.
- **Queue** block models queues. In this model it simulates a queue of customers waiting for the moment they can start accessing ATM services.
- **Delay** here simulates the delay associated with the service at ATM.
- **Sink** block indicates the end of the flowchart and destroys the incoming agents.

Configure the flowchart blocks

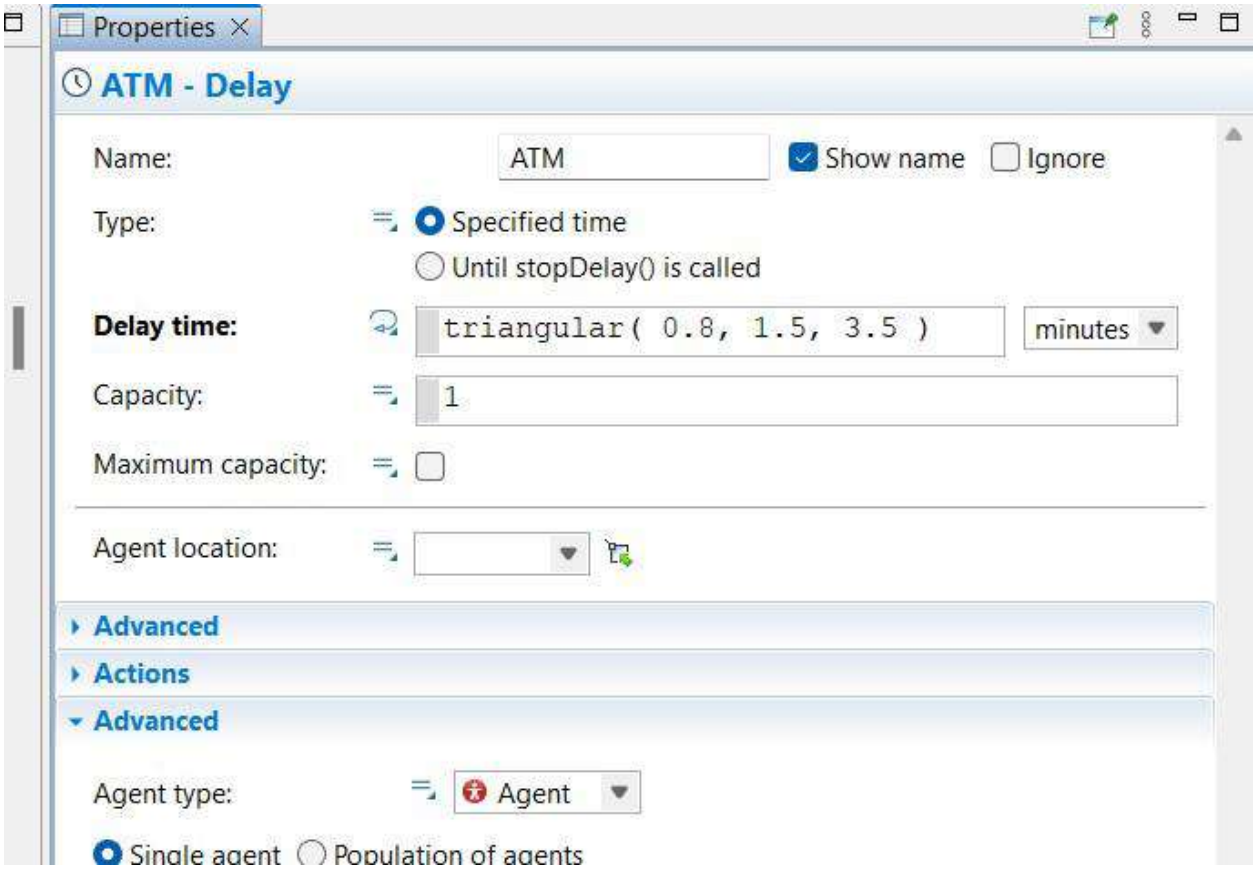
Source block.



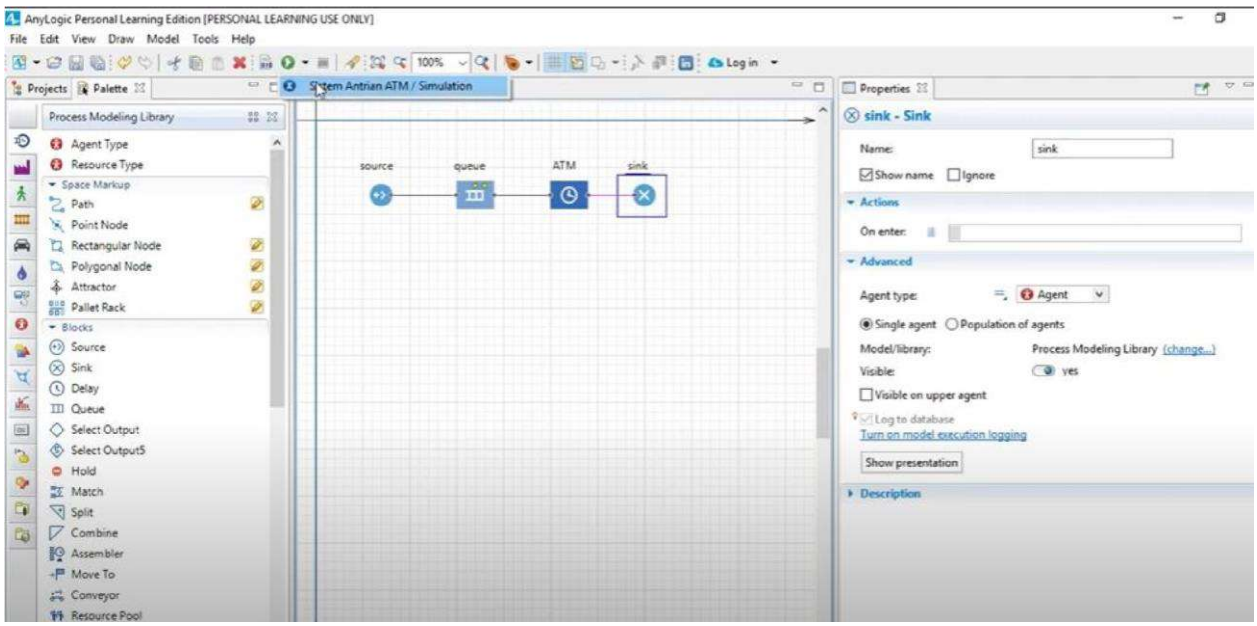
Queue block



Rename the **Delay** block to ATM . Assume that processing time inside **ATM** is triangularly distributed with mean value of 1.5 , min of 0.8 and max value of 3.5 **minutes** .

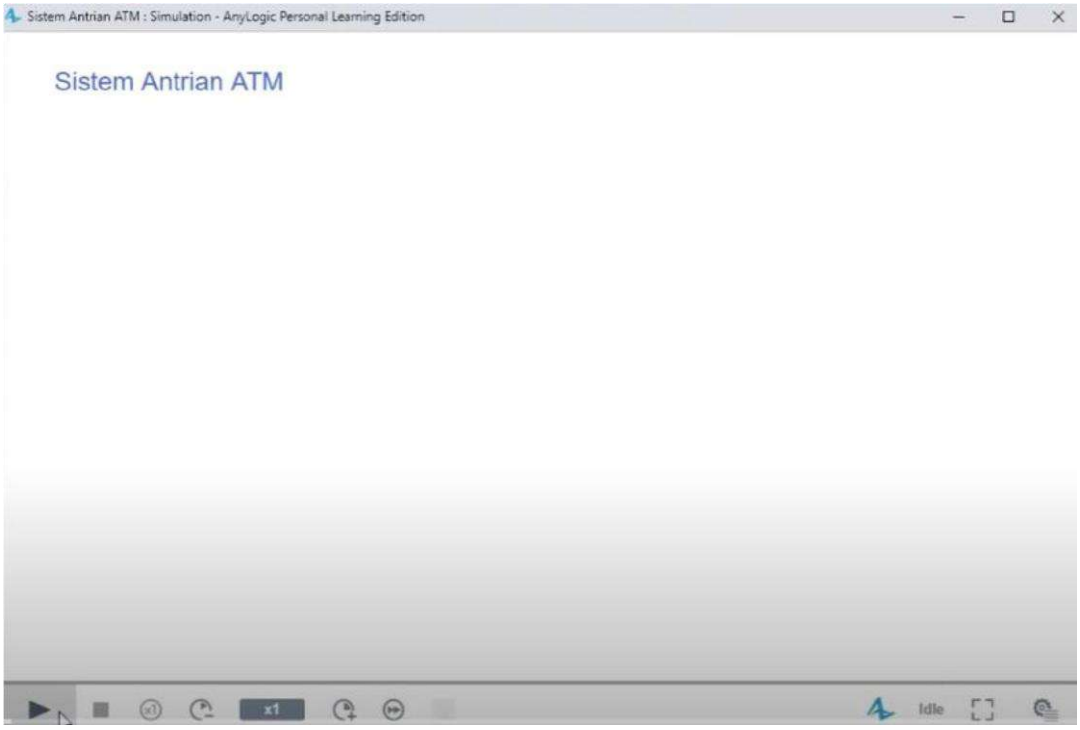


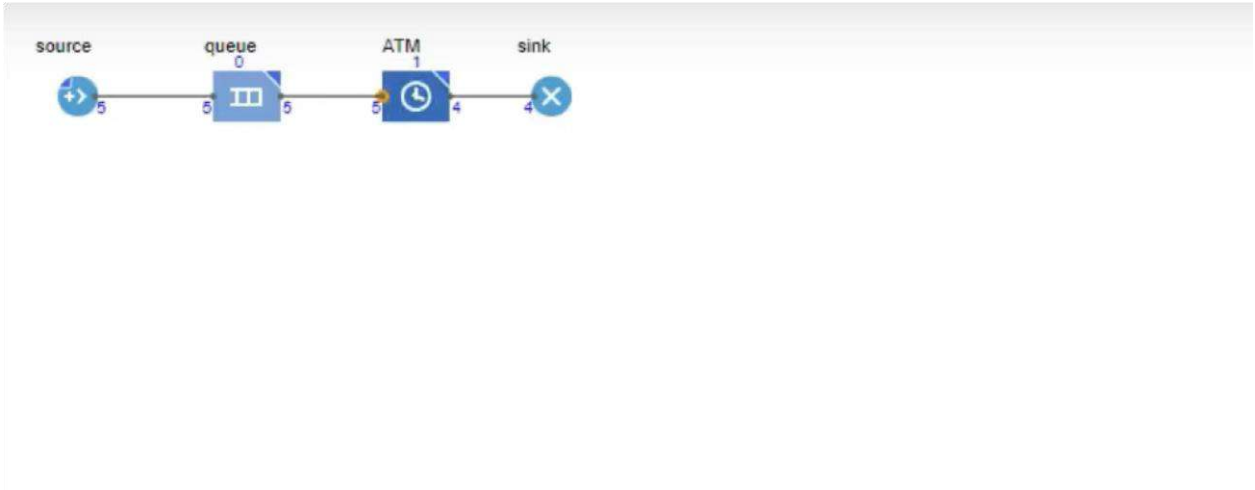
Run the model



The model will launch immediately. You will see animated flowchart. Each model created with **Process Modeling Library** instantly has animated flowchart where you can see detailed current block status, for example queue size, number of agents left and so

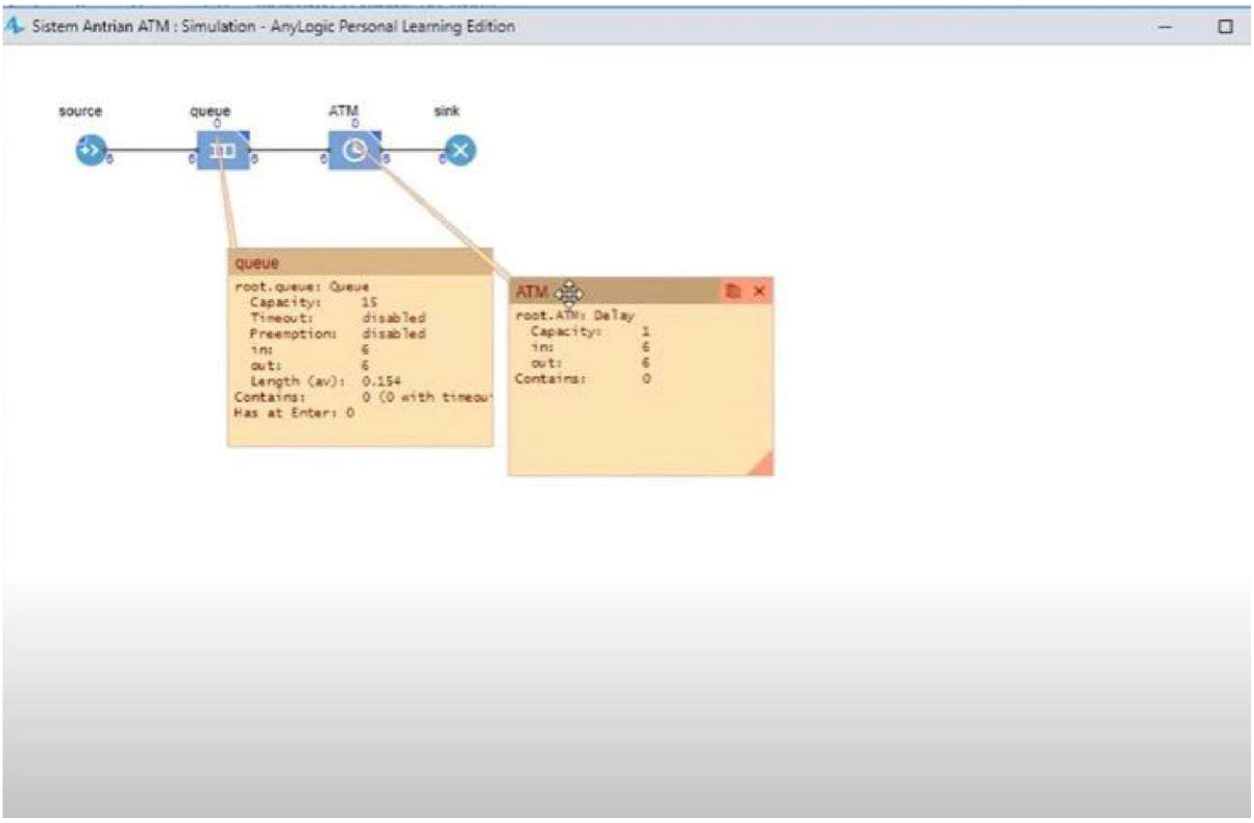
on — completely visualized!





If needed, adjust the execution speed to your needs using **Slow down** and **Speed up** buttons in the control panel.

Click on the block to open it's inspect window. Inspect window shows statistics of the block, e.g. **Queue** blocks inspect shows the queue capacity, the number of agents passed through either port of the block or also whether the timeout option is enabled for this queue. **Contains** string displays the number of agents currently being in the block along with IDs of these agents.

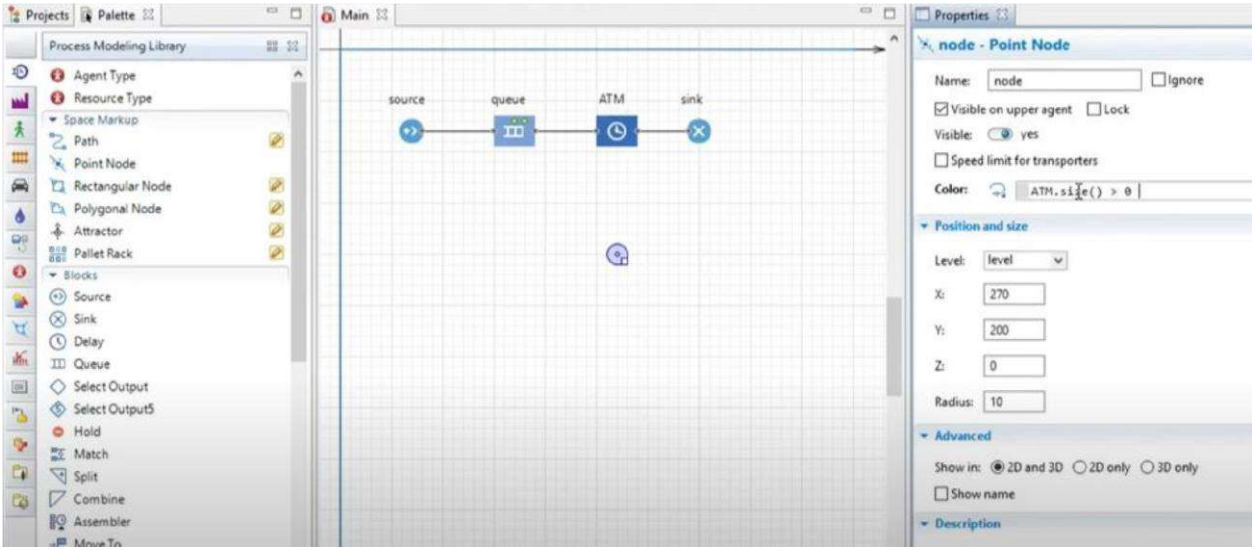


Phase 2: Creating Model Animation

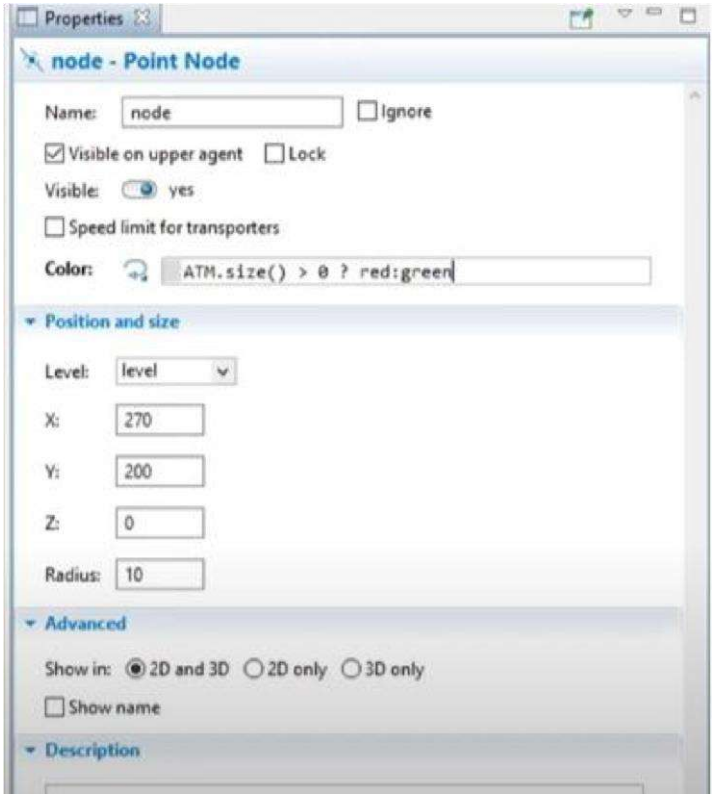
Adding Space Markup shapes

Set up space markup for ATM

1. Draw the ATM as a point node. First, open the **Space Markup** palette in the **Palette** view .
2. Drag the **Point Node** element from the **Space Markup** palette into the graphical editor and place it under the flowchart.

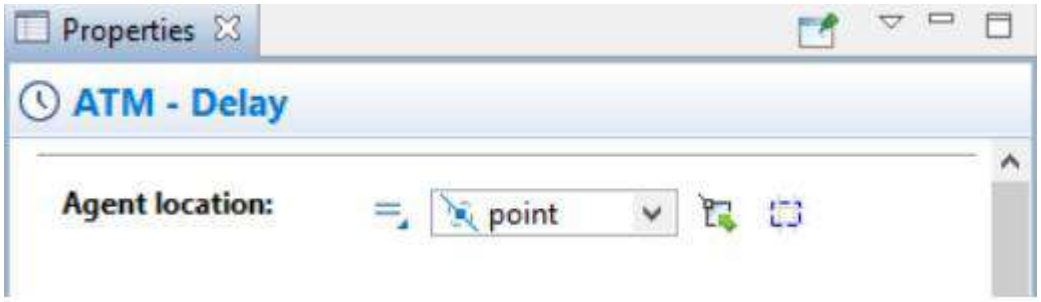


3. Select the point node in the graphical editor to open its **Properties** view . Enter the expression that will allow to change the color of the shape at runtime in the **Color** property: ATM.size() > 0 ? red : green



4. Click the **ATM** block in the flowchart to open its **Properties** view.

5.In the **Agent location** option select the **point** node that you have drawn previously. You can either click the down arrow and select the point node from the list of appropriate space markup elements, or you can click the button, located on the right, to select this space markup shape from the graphical editor (the rest of the elements in the editor will be greyed out).

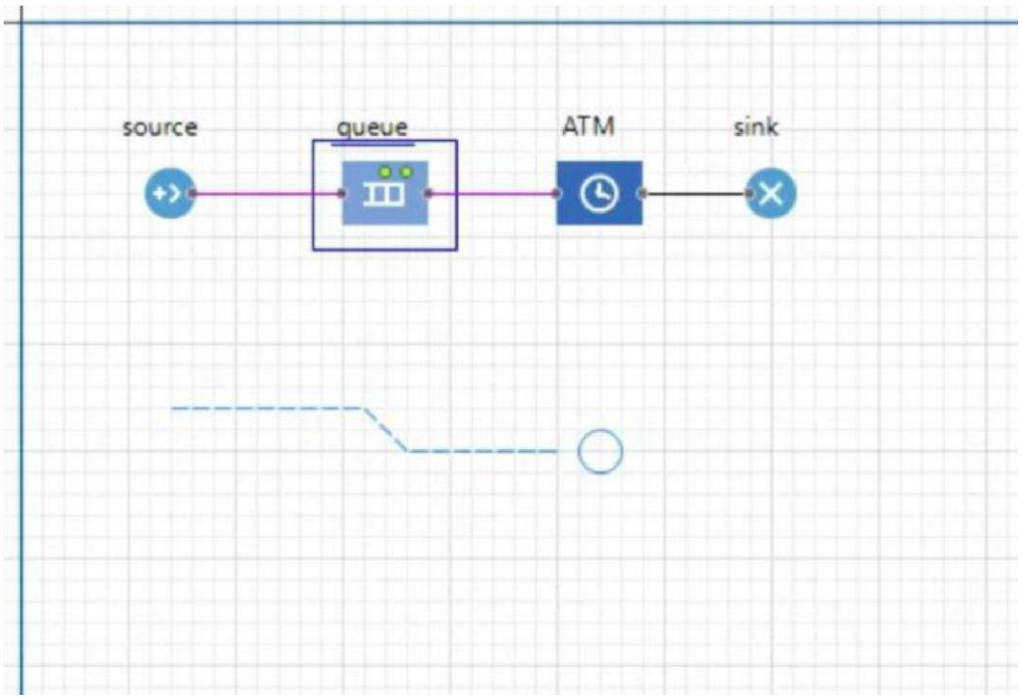


Set up space markup for the queue

1. Draw the queue as a path. First, open the **Space Markup** palette in the **Palette** view.

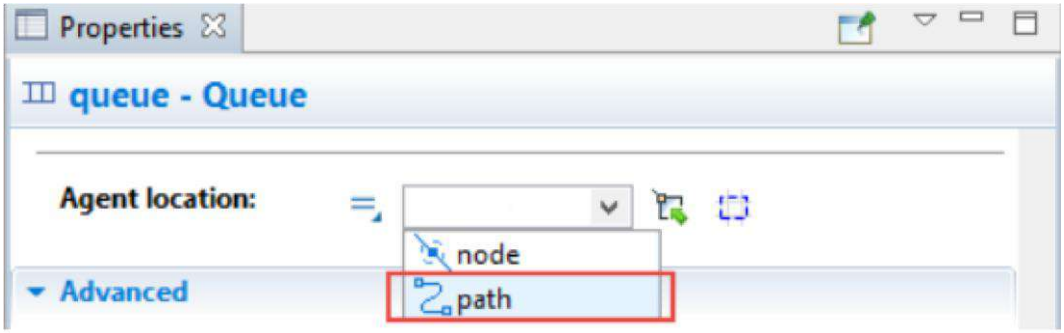
2. Double-click the **Path** element in the palette. A pencil icon will appear next to it, which means that you have switched to the drawing mode.

3. Click in the graphical editor to put the first point of the path. Do more clicks to add
   turning points. Finish drawing with a double -click.

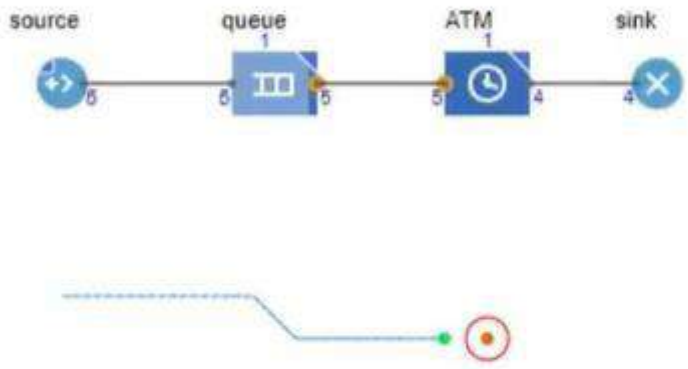


4. Click the **queue** block in the flowchart and go to its **Properties** view.
5. In the **Agent location** option select the **path** you have drawn previously. You can either click the down arrow and select the path from the list of appropriate elements, or you can click the button, located on the right, to select this space markup shape from the graphical editor (the rest of the elements in the editor will

be greyed out).



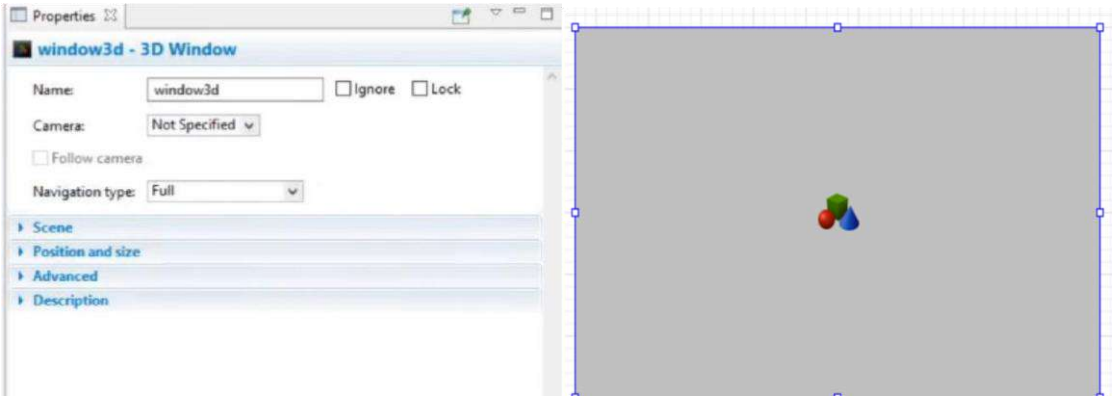
R un the model and observe its behavior. If you want to speed up the simulation significantly, switch to virtual time mode by clicking the **Run as fast as possible (virtual time mode)** control. Switching to virtual time mode allows you to view simulation run at its maximum speed. Therefore, you can simulate a long period of time.



Adding 3D Animation

1. Drag the **3D Window** element from the **3D** section of the **Presentation** palette to the graphical editor.

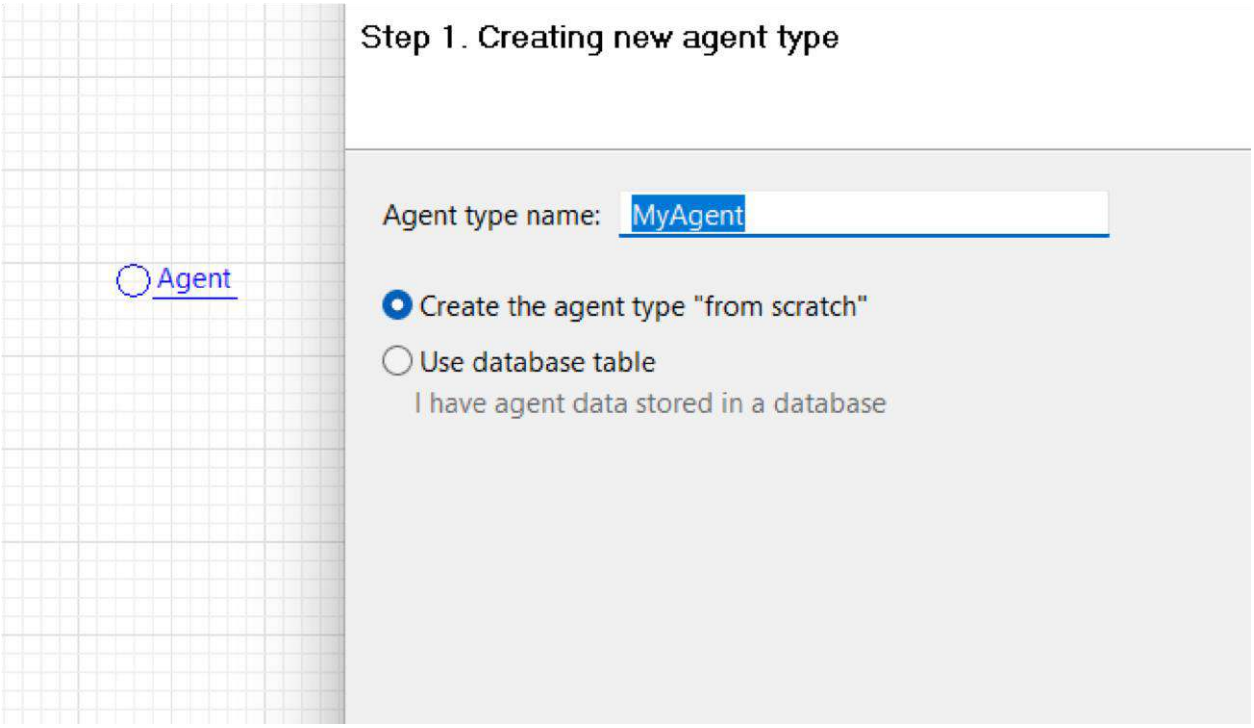
2. The grey area will appear on the screen. Locate it where you want your 3D presentation to be shown at the model runtime:



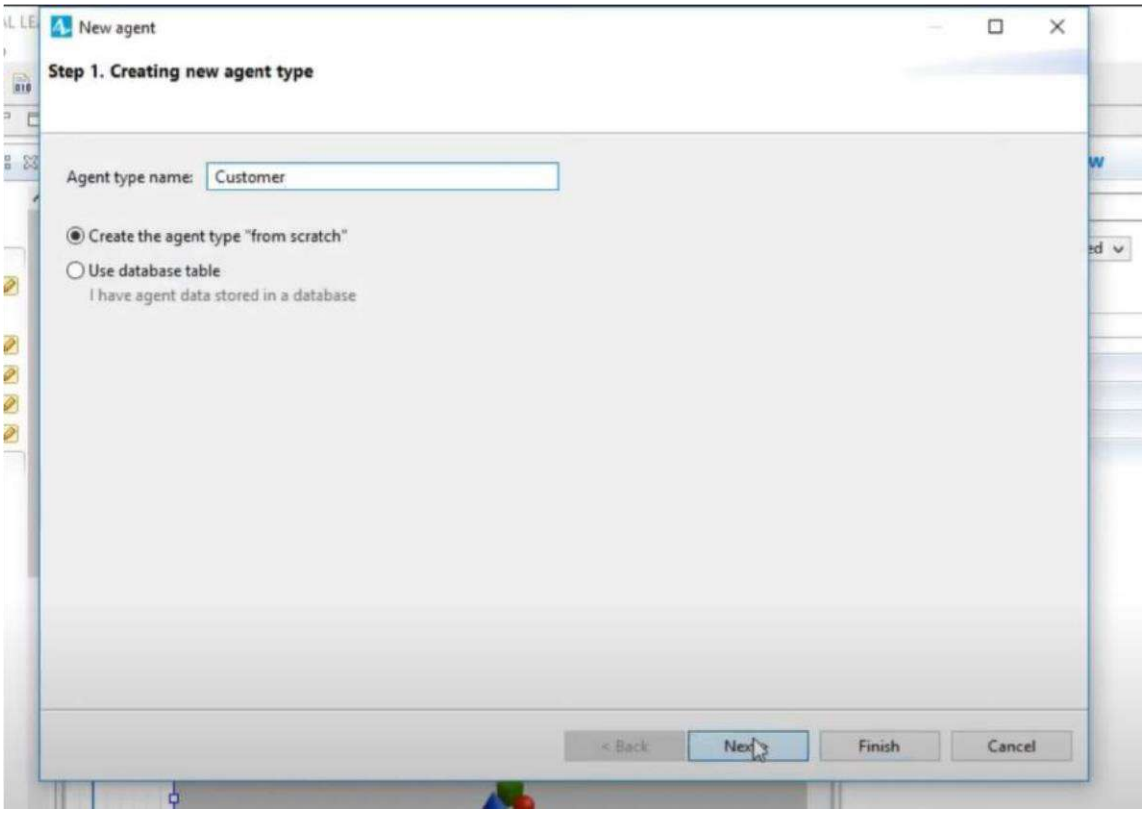
Adding 3D objects

Create a new agent type

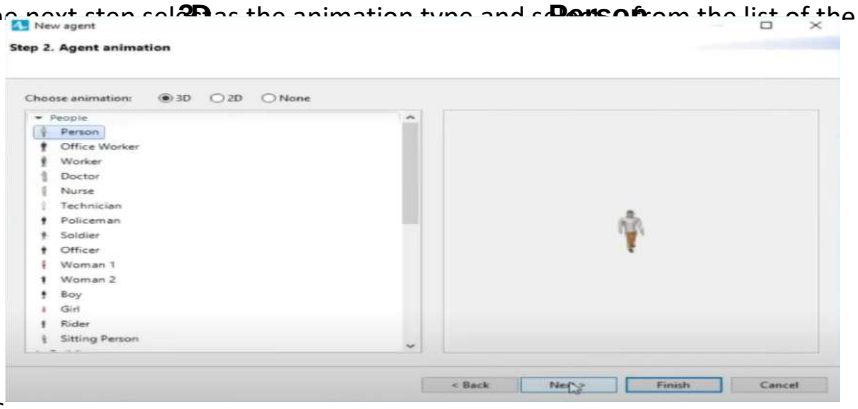
1. Open the **Process Modeling Library** in the **Palette** view.
2. Drag the **Agent type** element into the graphical editor.



3. The **New agent** wizard will open on the **Creating new agent type** step. Enter Customer as the **Agent type name**, and leave the **Create the agent type "from scratch"** selected. Press **Next**.



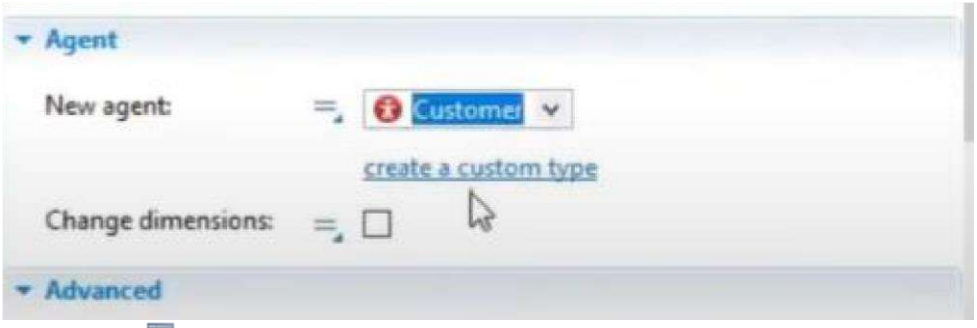
4. In the next step select **2D** as the animation type and select **Person** from the list of the 3D figures



5. Click **Finish**. The new Customer diagram will open. You can find the **Person** 2D figure in the axis origin. Switch back to the **Main** diagram.

Configure flowchart to use the new type

1. On the Main diagram, select the **source** block in the graphical editor.
2. Choose **Customer** from the **New agent** drop-down list.



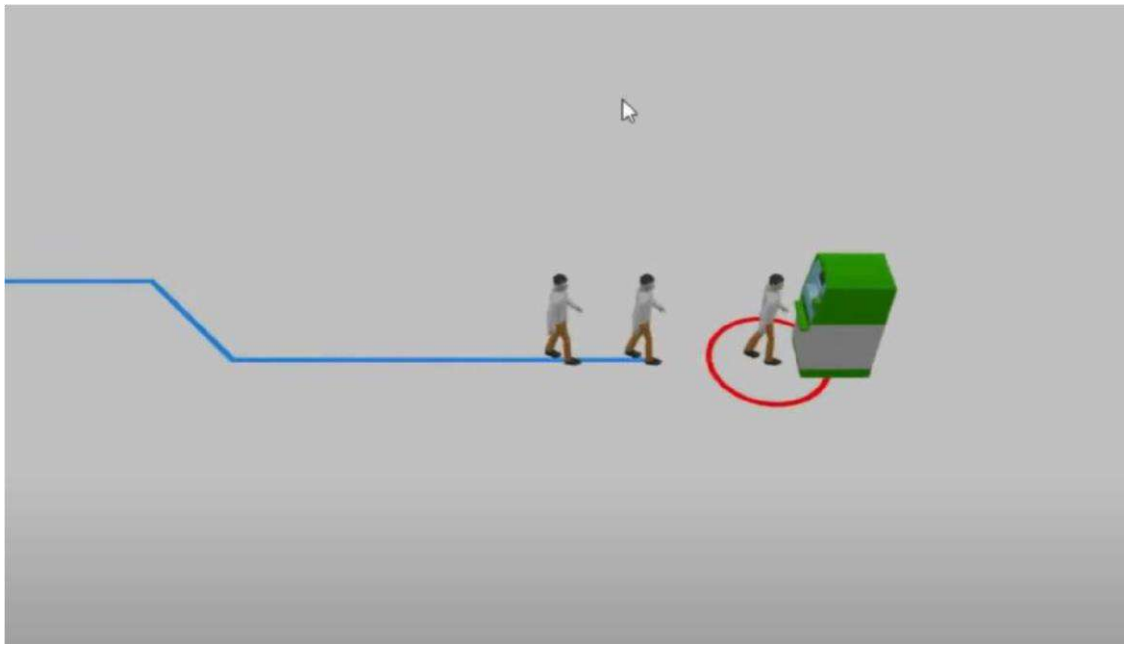
3.Go to the **Properties** view of the **node** element and set **Visible** control to **no**. This way the markup shape will be invisible during animation on model runtime.

4.In the same manner make the **path** element invisible at runtime too.

5. Run the model and switch to 3D view to see our customers moving in the queue.

Add an ATM figure

1. Open the **3D Objects** palette in the **Palette** view.
2. Drag the **ATM** 3D figure from the **Supermarket** section of this palette onto the **node** in the graphical editor. In the displayed **Auto scale 3D object** dialog box click the **Yes** button.
3. If you run the model now and check 3D animation in **window3D** mode, you will notice that our ATM does not face the customers' flow and we need to rotate it.
4. Select the **atm** 3D object in the graphical editor and open the section **Position** in its **Properties** view.
5. Choose 0 degrees from the drop-down list of the **Rotation Z** option.
6. Run the model to double-check that the ATM is facing the customers now.

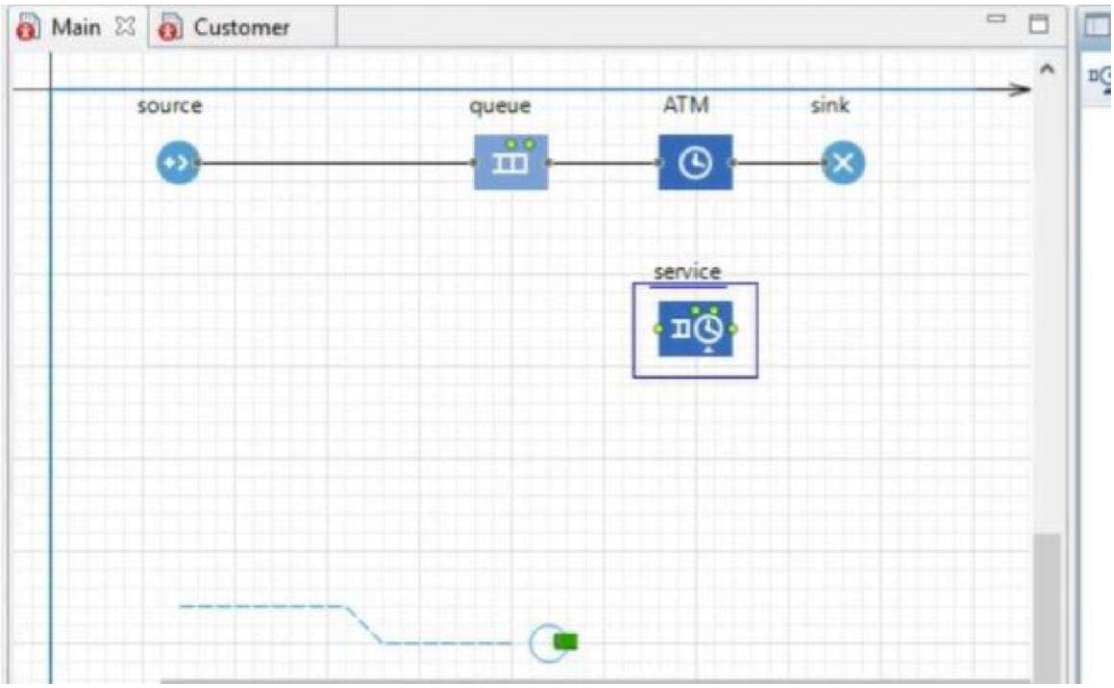


Phase 3: Adding tellers

Modifying the flowchart

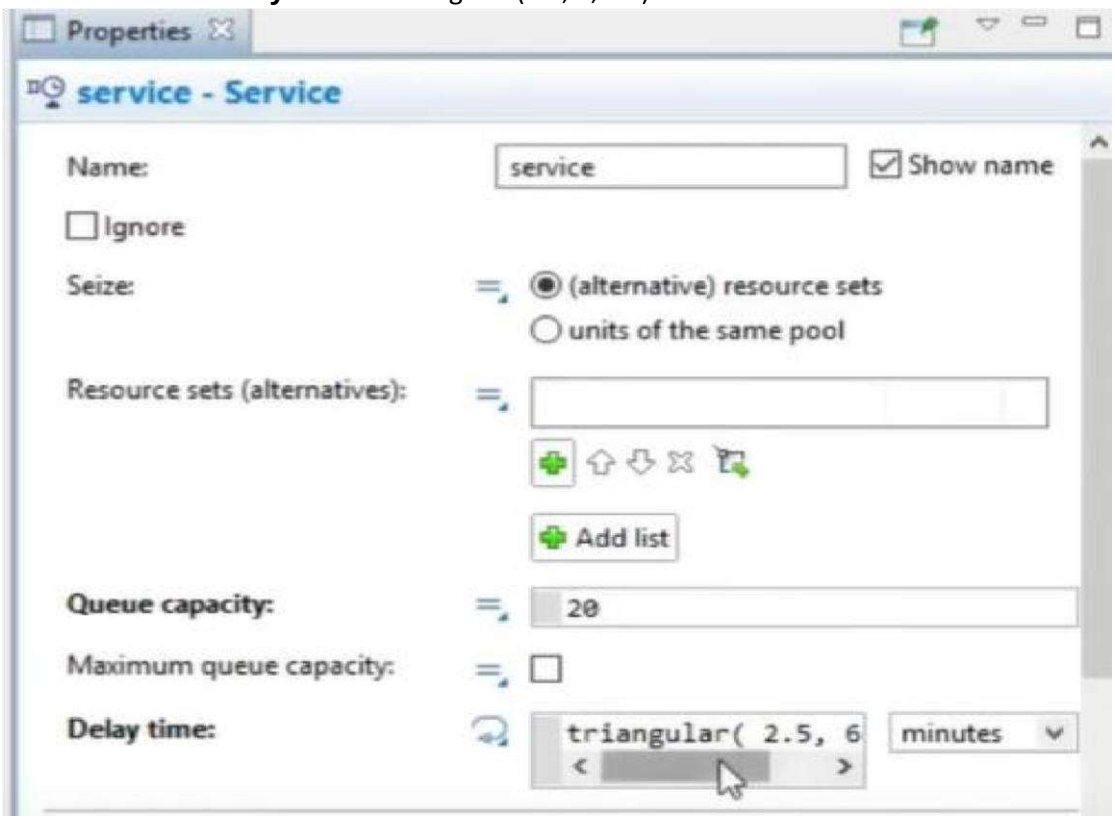
Create a service

1. Open the **Process Modeling Library** in the **Palette** view and drag the **Service** block onto our Main diagram.



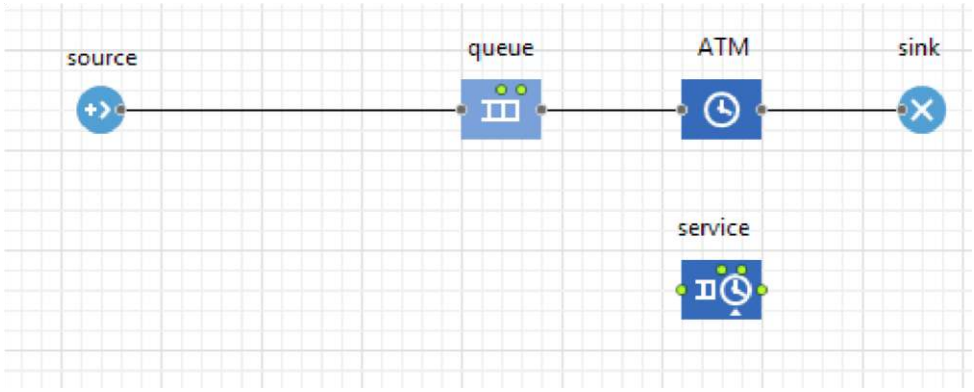
2. Go to the **Properties** view of the **service** block.
3. Modify the block's properties:
    1. There is only one queue for all tellers. Set up **Queue capacity** to 20.

2. We assume that service time is triangularly distributed with the min value of 2.5, average value of 6, and the max value of 11 minutes. Set **Delay time** to triangular (2.5, 6, 11)

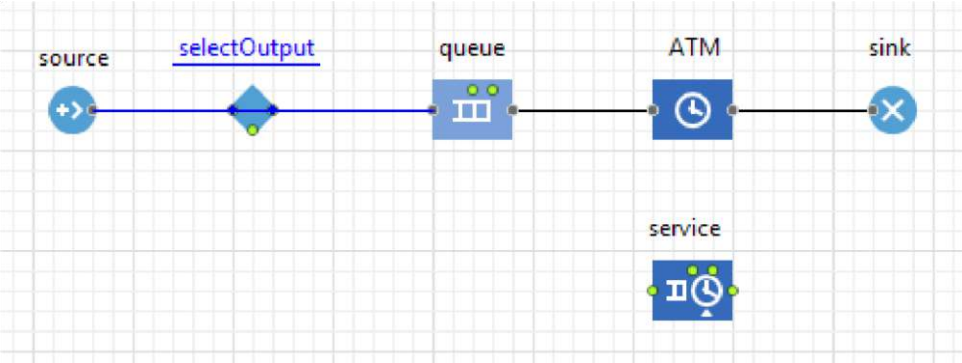


Adjust the process flowchart

1. Move the blocks **queue**, **ATM**, and **sink** to the right to make space for one block between the **source** and the **queue** .

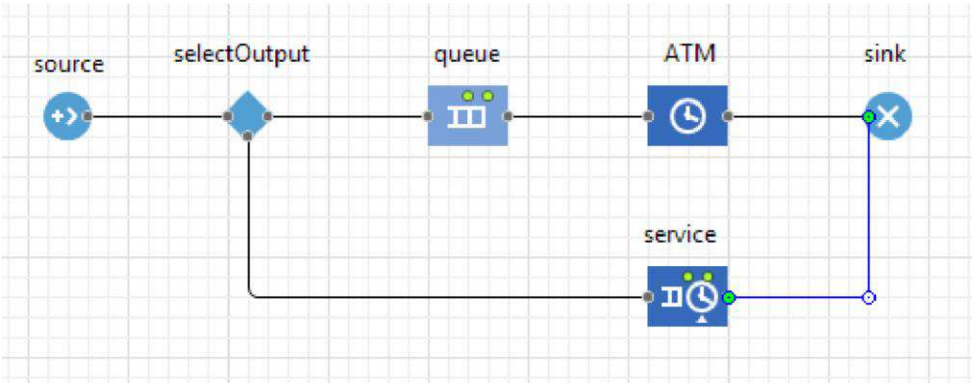


1. Open the **Process Modeling Library** in the **Palette** view and add the **SelectOutput** block in the resulting space. When you place the block on the connector, it will automatically get built in.



SelectOutput is a decision -making block. The agent arrived at the block is forwarded along one of two output ports depending on the user -defined condition.

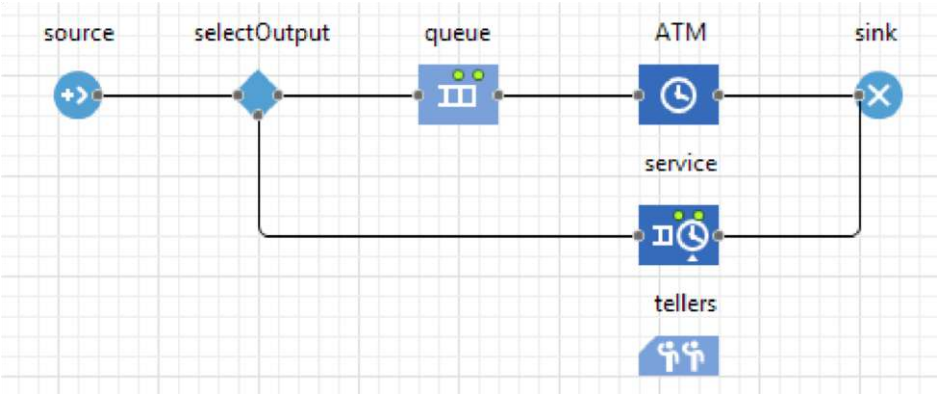
2. Select **selectOutput** in the flowchart and go to its **Properties** view. Choose the option **If condition is true** for the **Select True Output** parameter. Make sure that **Condition** is randomTrue(0.5).
   This agent routing condition defines that the number of customers competing for ATM and teller service will be approximately equal.
3. Connect **selectOutput** and **service** with other blocks as shown in the figure:



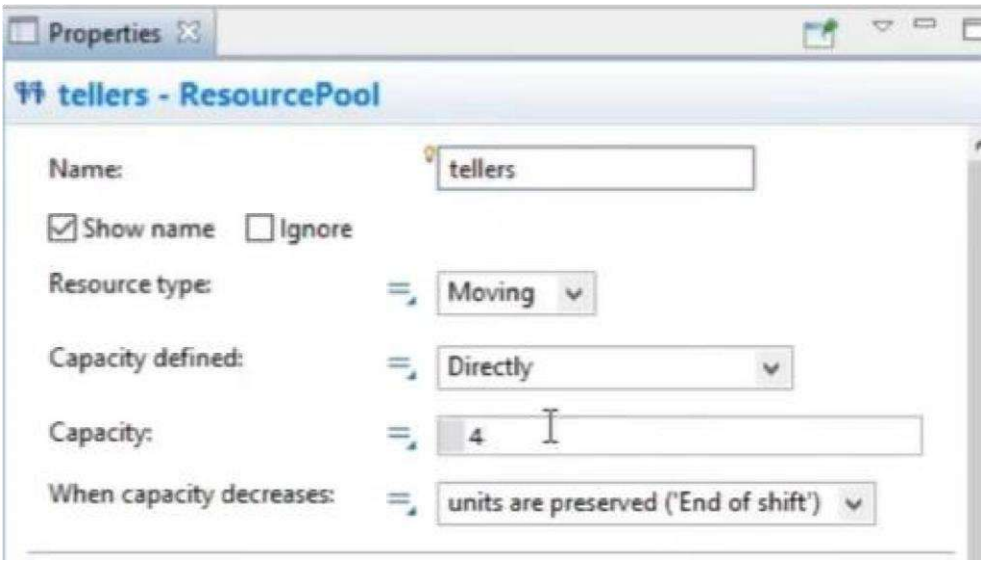
Add resources for the service

1. Open the **Process Modeling Library** in the **Palette** view and drag the **ResourcePool** block onto our Main diagram.
   ResourcePool block is a storage for resource units.
2. Place it under **service** and go to its **Properties** view.
3. Name the block tellers

.

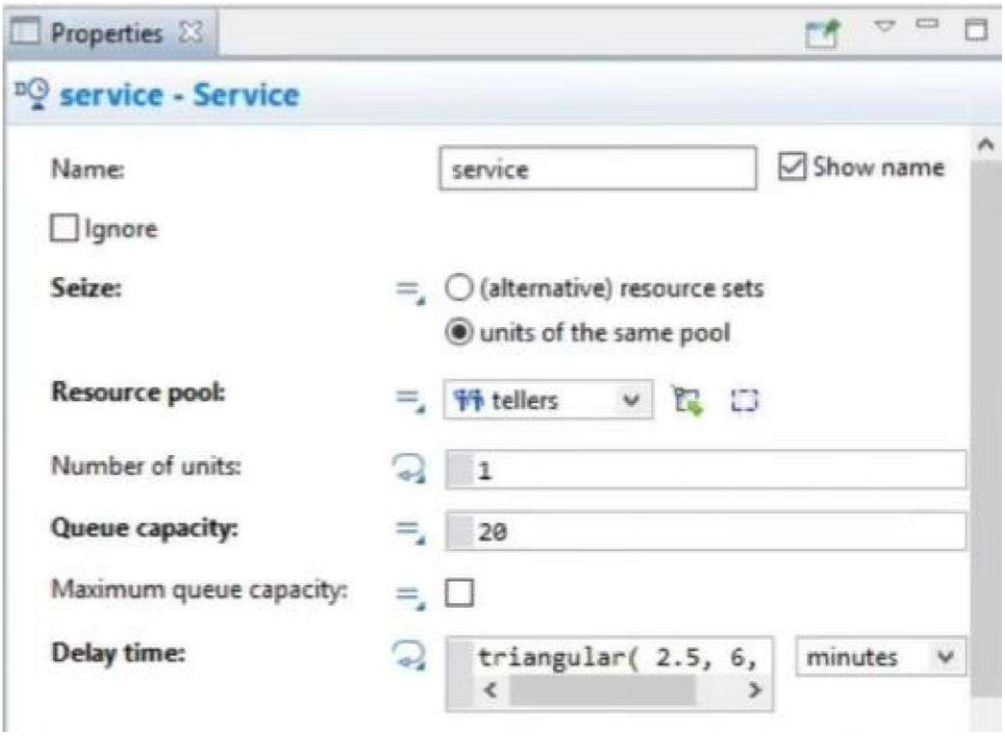


4. Specify that this resource block has only four resource units by setting its **Capacity** to 4.



5. **ResourcePool** block should be connected to resource seizing and releasing block (**Service** in our case). So we need to modify the properties of the **service** block.

6. Select **service** in the flowchart to open its properties. Set the **Seize** parameter to **units of the same pool**. Then specify the resource pool we have created in the option **Resource pool**. You can either click the down arrow to select the resource pool object from the drop-down list, or you can click the button, located on the right, to select the object in the graphical editor (all inappropriate objects will be greyed out).

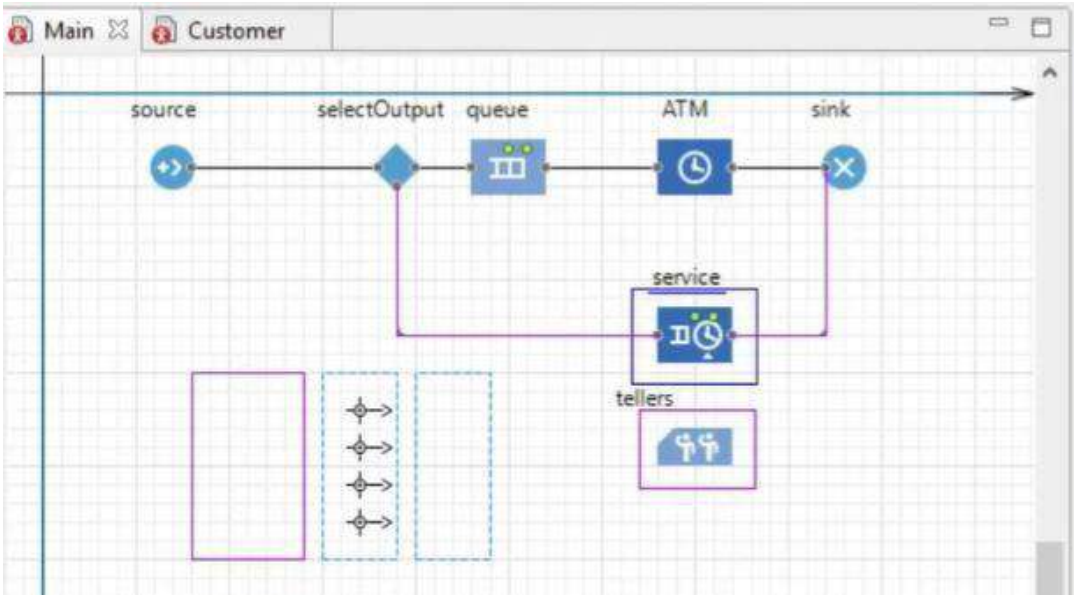


Now since the model has changed, we need to alter the model animation as well.

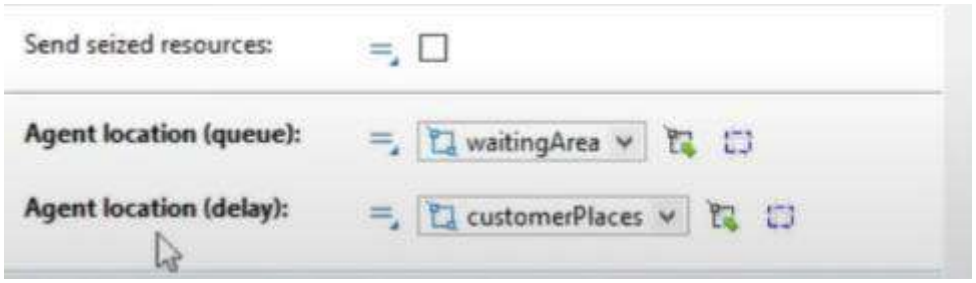
Adding space markup shapes

Set up space markup for the queue to tellers

1. This time we will draw a waiting area using a rectangular node. First, open the **Space Markup** palette in the **Palette** view.
2. Double-click the **Rectangular Node** element to switch to the drawing mode.
3. Click in the graphical editor and drag the rectangle without releasing the mouse button. Release when you have a rectangular node of the required form. You can edit its form later as you need.
4. Name the node waitingArea
5. Switch the **Visible** control to **no**. This way the markup shape will be invisible during animation at model runtime.



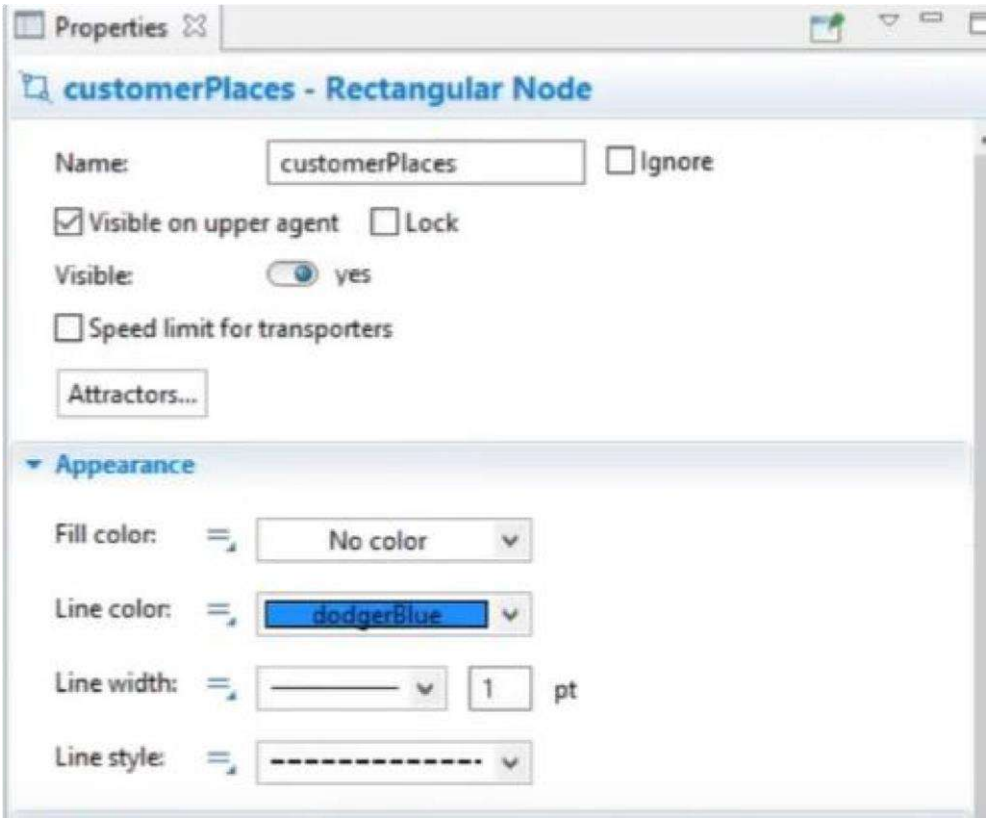
6. Click the **service** block in the flowchart and go to its **Properties** view.
7. Select the **waitingArea** node in the **Agent location (queue)** option.



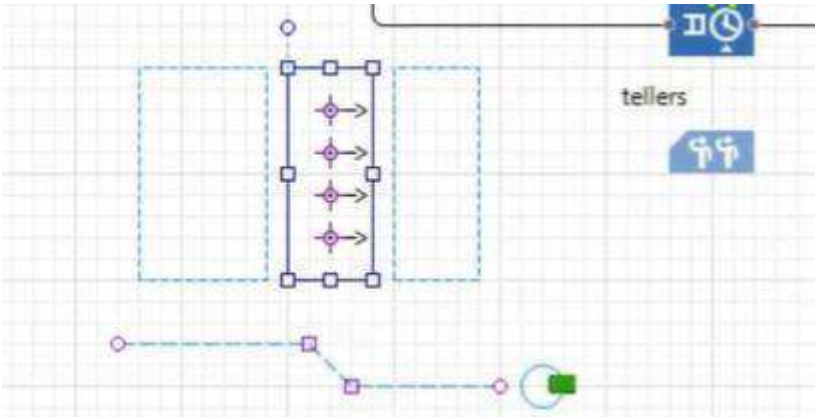
Set up space markup for the customers

The customers need a place to stand somewhere while they are getting serviced by tellers.

1. Draw another rectangular node as displayed in the image below. Name the node customerPlaces and in its properties switch the **Visible** control to **no**.

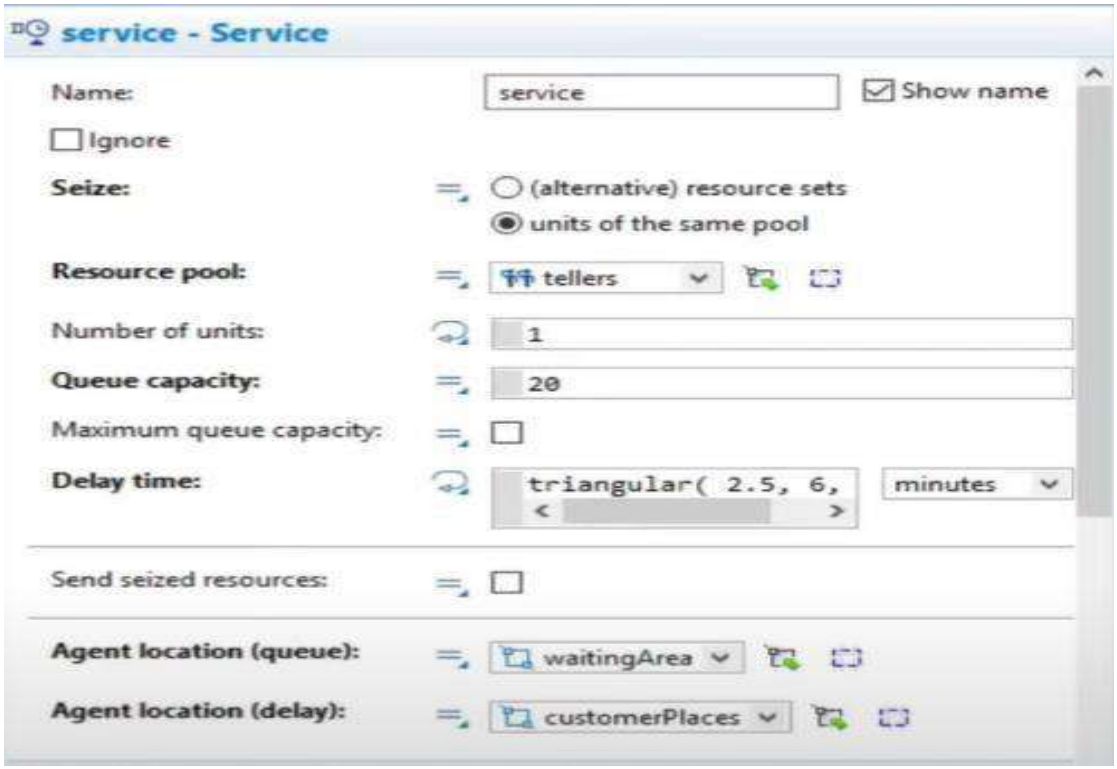


2. In the **Attractors** window that will pop-up, specify 4 for the creation mode **Number of attractors** and click **OK**. You will see that attractors appeared in the **customerPlaces** node with an even offset.



3. Now we need to refer to this area in the flowchart. Click the **service** block in the flowchart and go to its **Properties** view.

4. Select the **customerPlaces** node in the **Agent location (delay)** option.

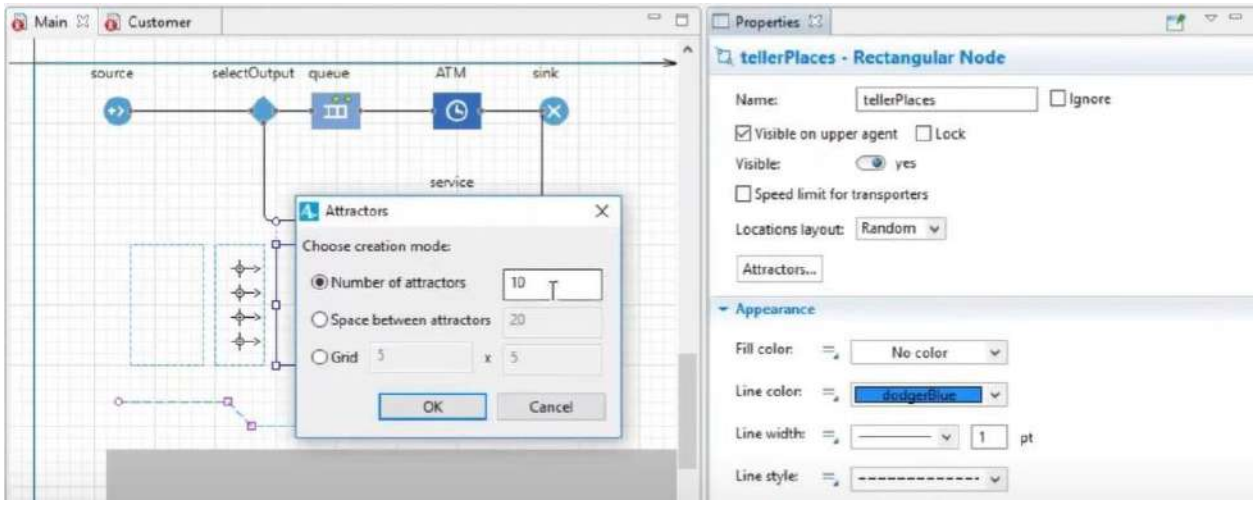


Set up space markup for the tellers

1. Since we have 4 tellers, we will use a rectangular node to draw this service area.

2. Draw another rectangular node as displayed in the image below. Name the node tellerPlaces

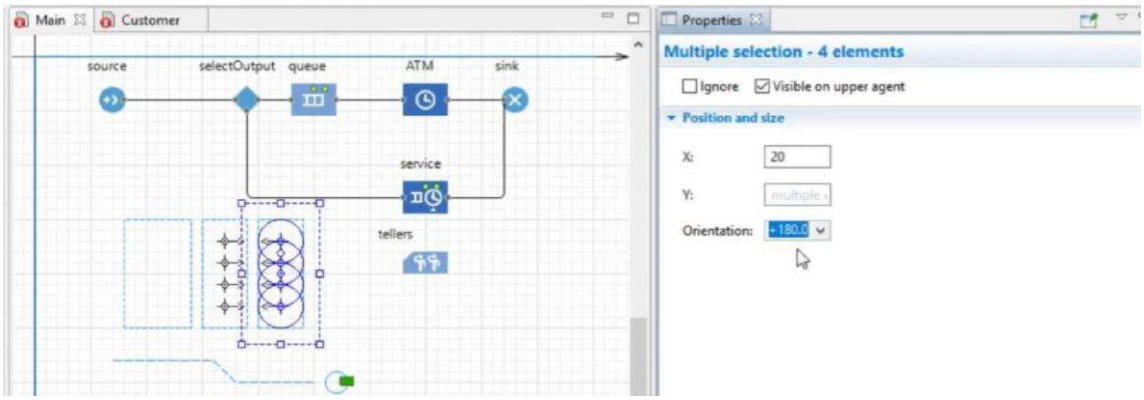
and in its properties switch the **Visible** control to **no**.



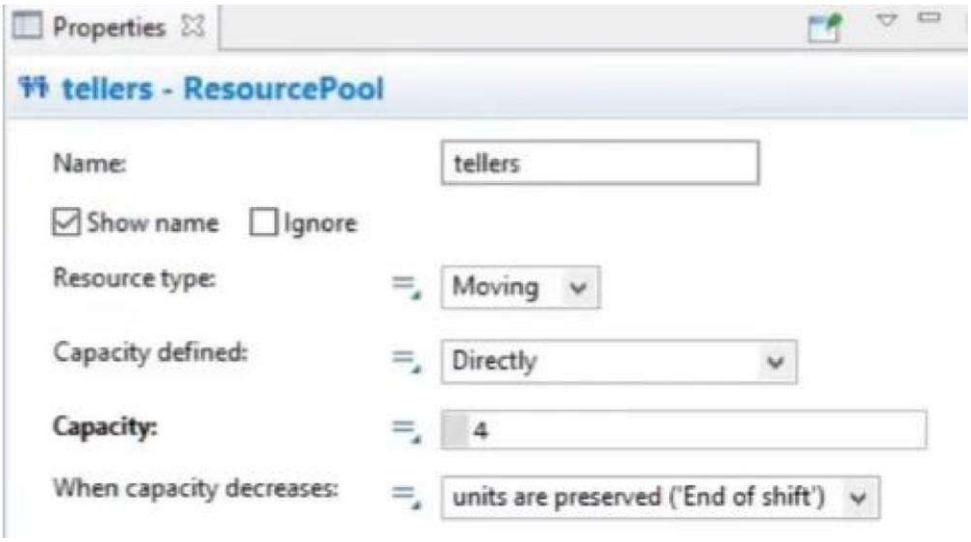
3.Create 4 attractors inside the node to specify tellers in the same manner as you have done before.

4. You will see attractors appear in the **tellerPlaces**node with the even offset, but facing the wrong direction. Select all attractors by clicking each one while

holding Shift key and go to the **Properties** view. In the section **Position and size**, change the **Orientation** parameter to 180.0 .



5. Click the **tellers** block in the flowchart and go to its **Properties** view.
6. Select the **tellerPlaces** in the **Home location (nodes)** parameter by clicking .



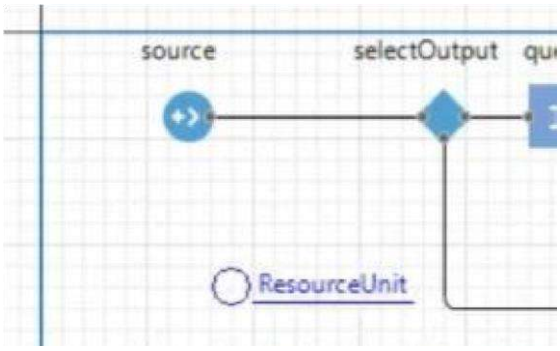
Adding 3D objects

It is time to add teller 3D objects to our model. We will create a new resource type to animate tellers.

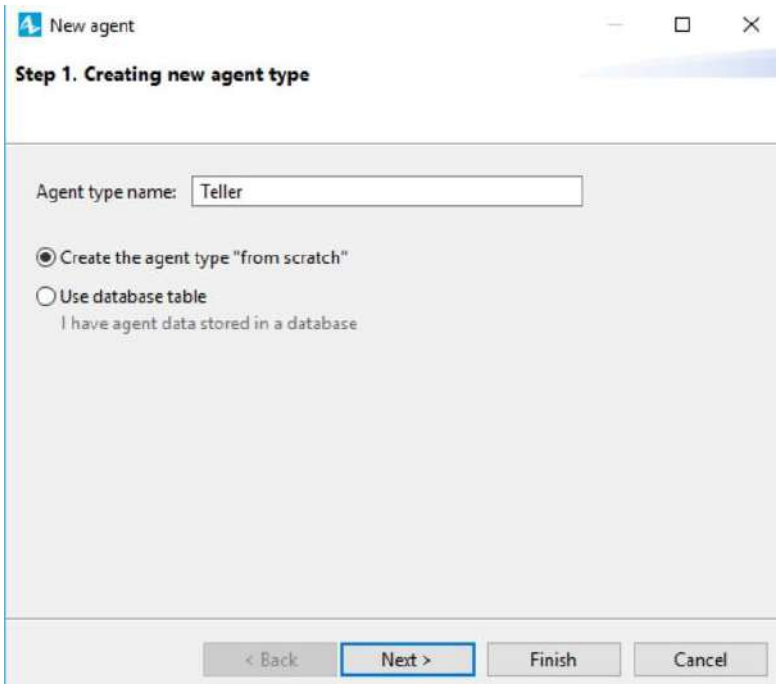
Create a new resource type

1. Open the **Process Modeling Library** palette.

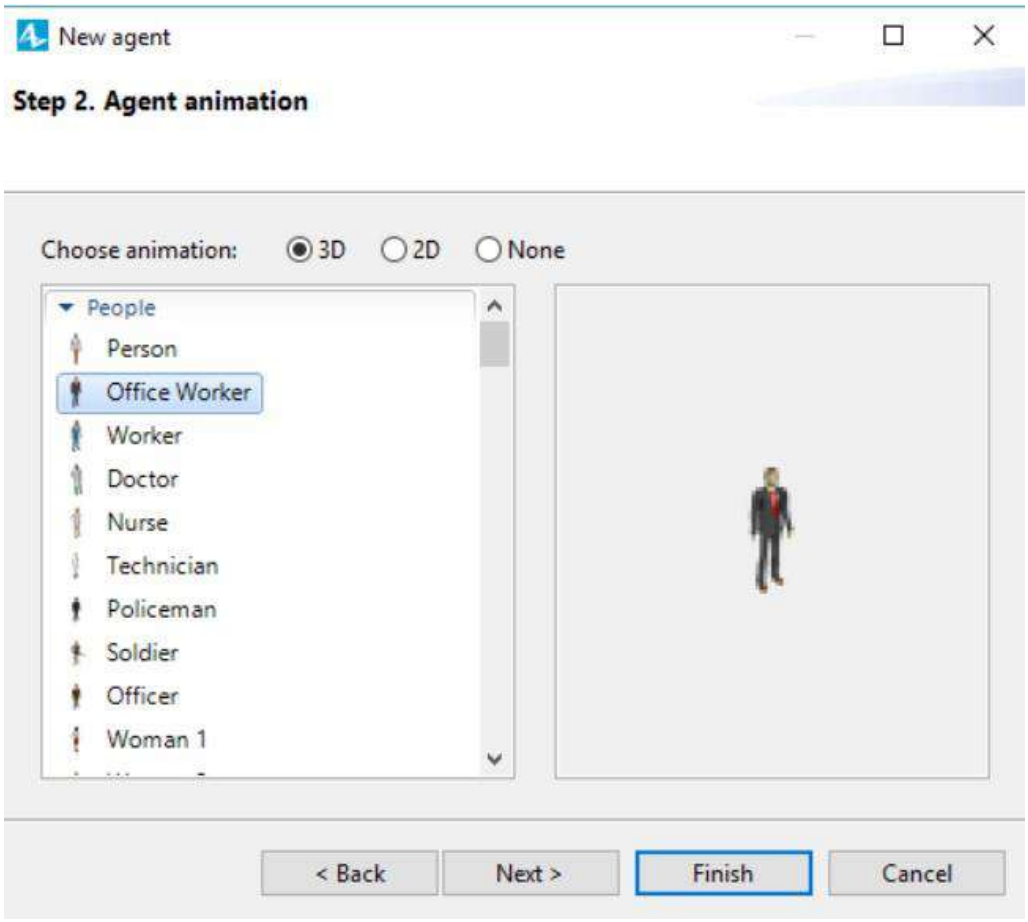
2. Drag the **Resource Type** element into the graphical editor.



3.The **New agent** wizard will open on the **Creating new agent** step. Enter Teller as the **Agent type name** and leave the **Create the agent type "from scratch"** selected. Press **Next**.



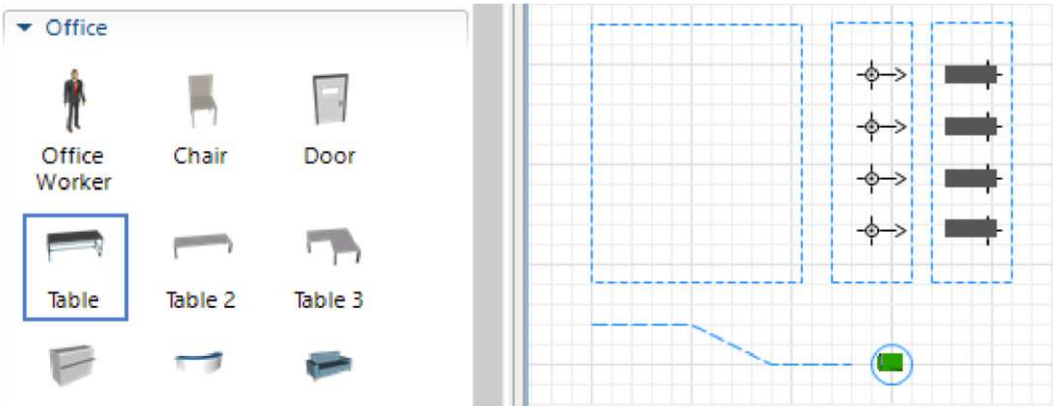
4.In the next step select **3D** as the animation type and select **Office worker** from the list of the 3D figures.



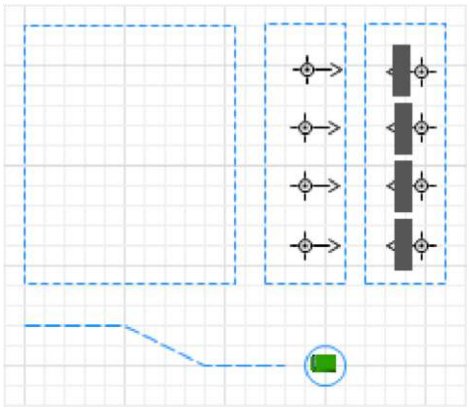
Click **Finish** . The new Teller diagram will open. You can find the **Office worker** 3D figure in the axis origin. Switch back to the Main diagram.

Add tables for the tellers

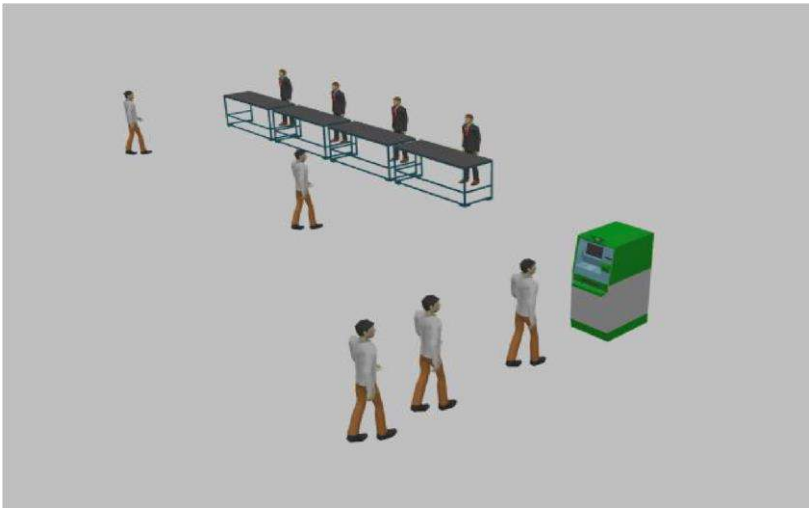
1. Open the **3D Objects** palette.
2. Drag four **Table** 3D figures from the **Office** section of this palette onto the **tellerPlaces** in the graphical editor. In the displayed **Auto scale 3D object** dialog box click the **Yes** button.
3. Place the tables over attractors since attractors are the places where the tellers stand.



4. You can see that their orientation is wrong. Select all tables by clicking them while holding the Shift key and go to the **Properties** view.
5. In the section **Position**, change the parameter **Rotation** to -90.0 degrees.
6. If necessary, rearrange all eight attractors and four tables so that they are reasonably lined up.



Now you can run the model and observe in 3D how some customers go to the ATM and other get service at the teller's tables.



Phase 4. Collecting statistics

Collecting utilization statistics

Add a bar chart to indicate mean ATM utilization

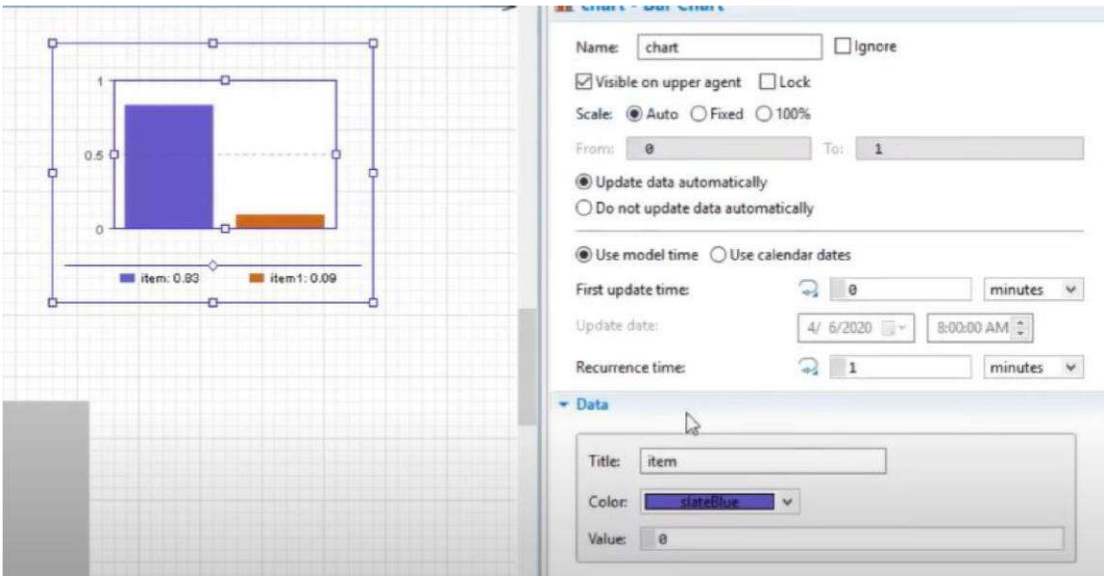
1. Open the **Analysis** palette. This palette contains charts and data objects used for collecting data and performing various statistical analysis on them. Drag the **Bar Chart** element from the stencil into the graphical editor.

2. Go to the **Data** section of the chart's properties. Click the **Add data item** button to add data item to be displayed by this chart.
3. Set the data item's **Title**: ATM utilization

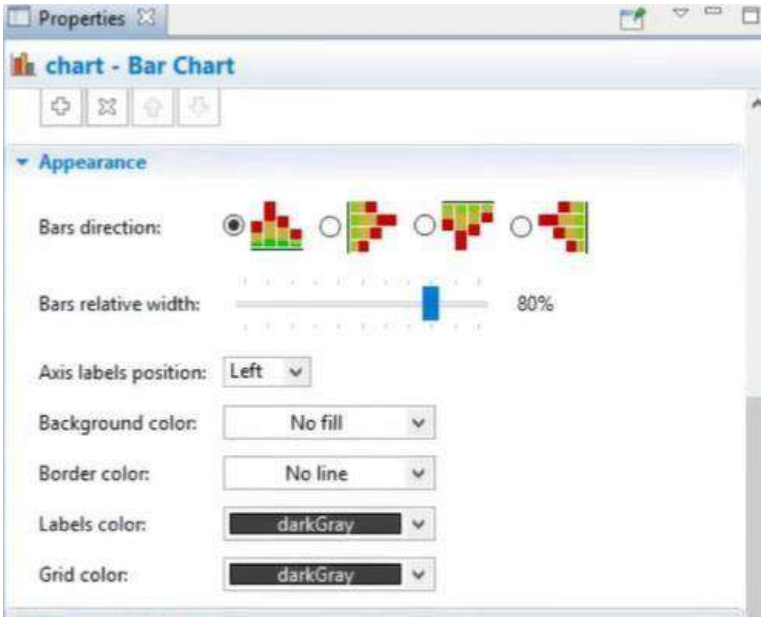
   .

4. Type ATM.statsUtilization.mean()

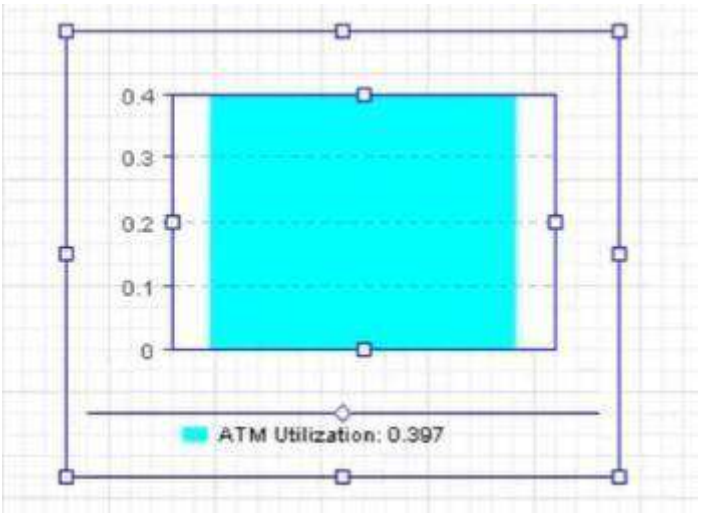
   as the **Value** of the data item. ATM is the name of the **Delay** block we created. Each **Delay** block has a statsUtilization data set that collects statistics on the object utilization. The mean() is the function that returns the mean value measured. You can use other functions to get statistical values, such as min() and max().



5.Change the position of the chart's legend in the **Legend** properties section.

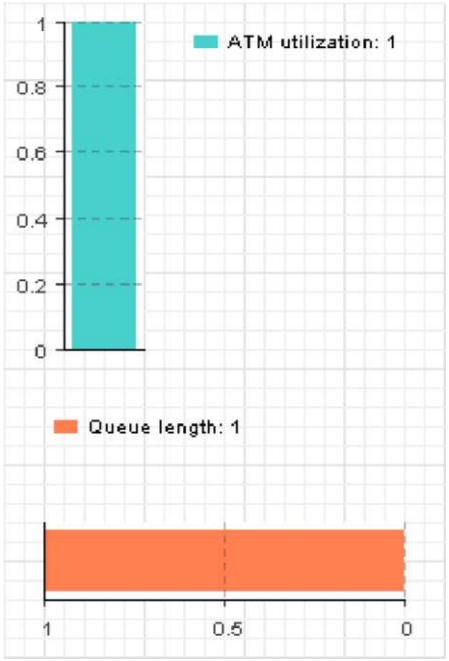


6. Now resize the chart as shown in the figure below:

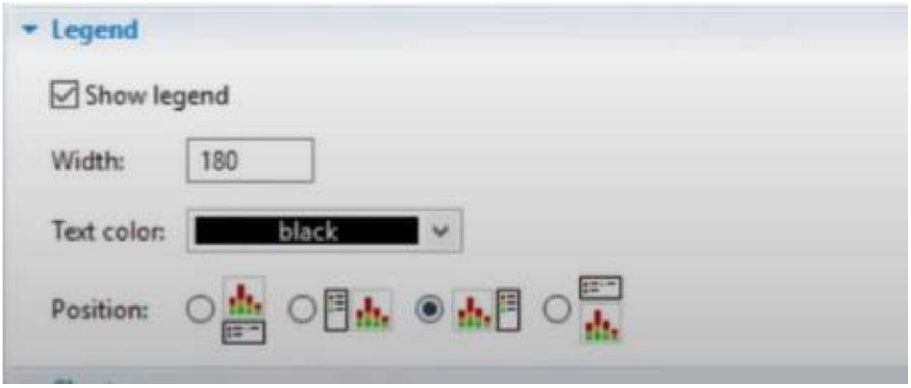


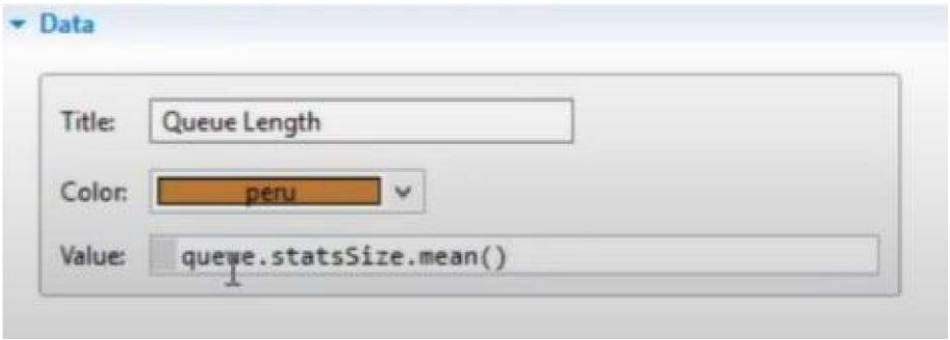
Add a bar chart to indicate mean queue length

1. Add another bar chart in the same way. Resize it to look like the one in the figure below.

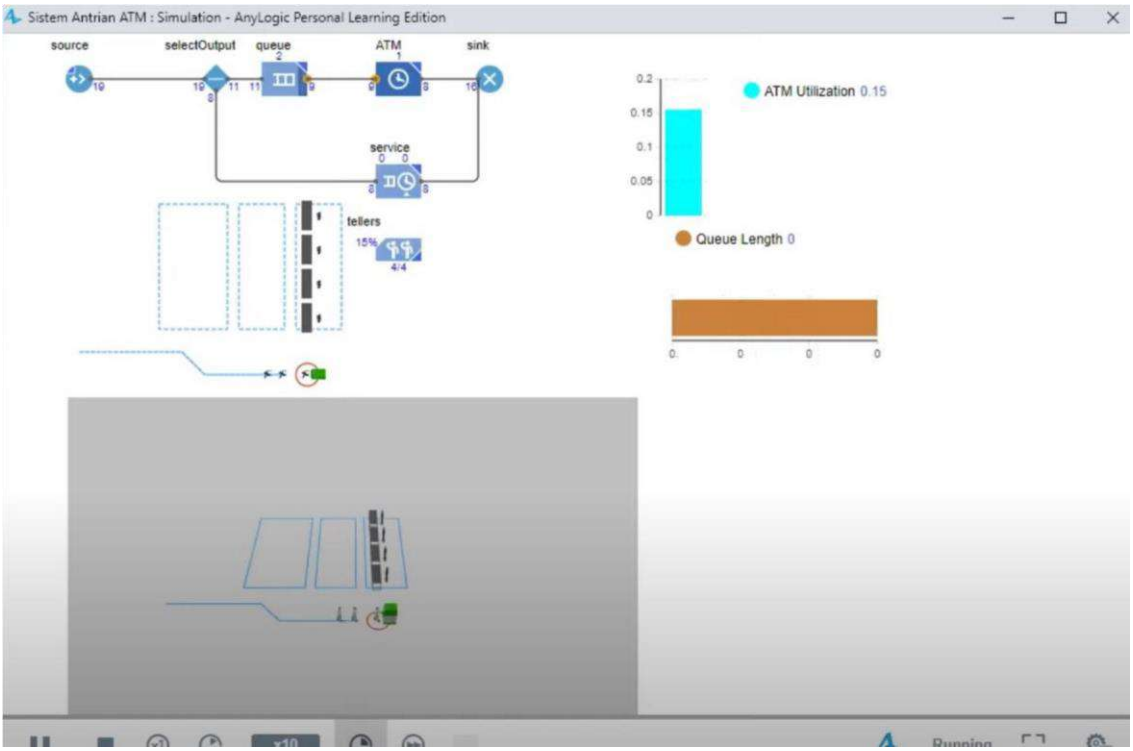


Open the **Appearance** section of the **Properties** view and choose the last option in the **Bars direction** property to make bars grow to the left. Change the position of the chart's legend in the section **Legend** (like it is shown in the figure below).





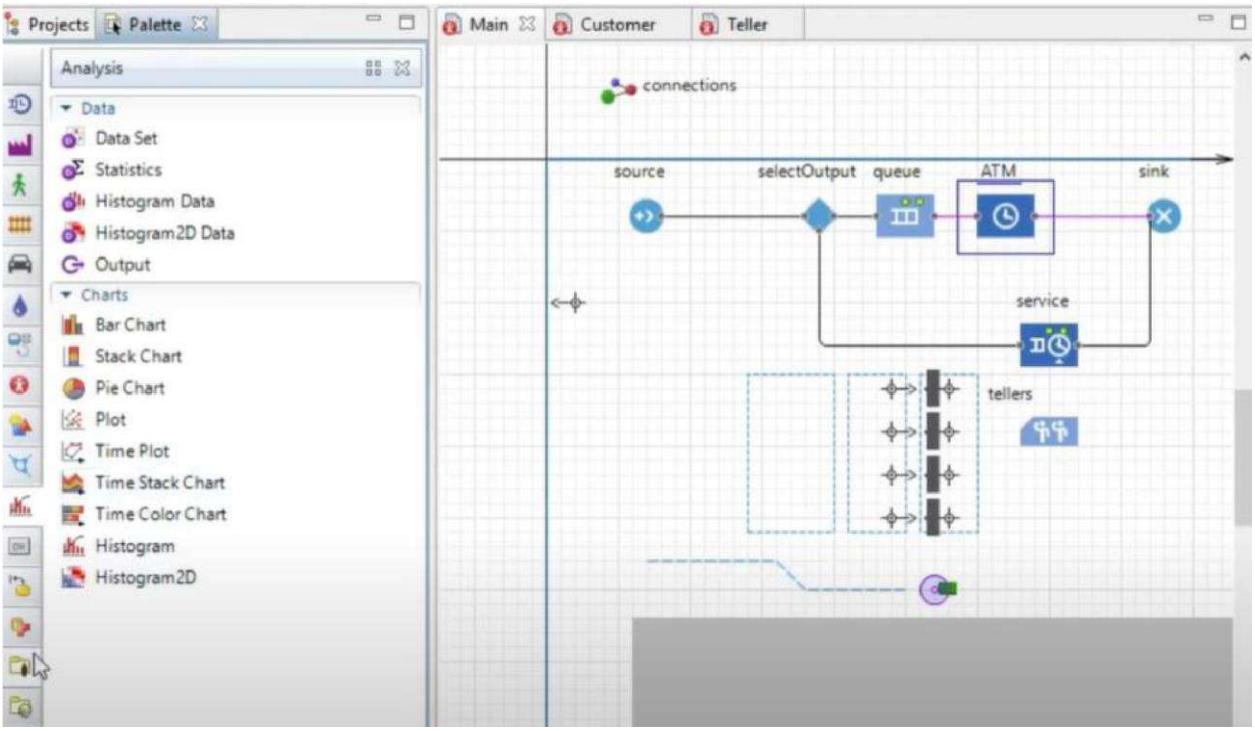
Run the model and observe the ATM utilization and mean queue length with new created charts.



Add time measurement blocks to the flowchart.

1. Adjust the flowchart to make space for a new block between **source** and **selectOutput**.

2. Open the **Process Modeling Library** in the **Palette** view and add the **TimeMeasureStart** block in the resulting space. Make sure that the ports of the block are properly connected.

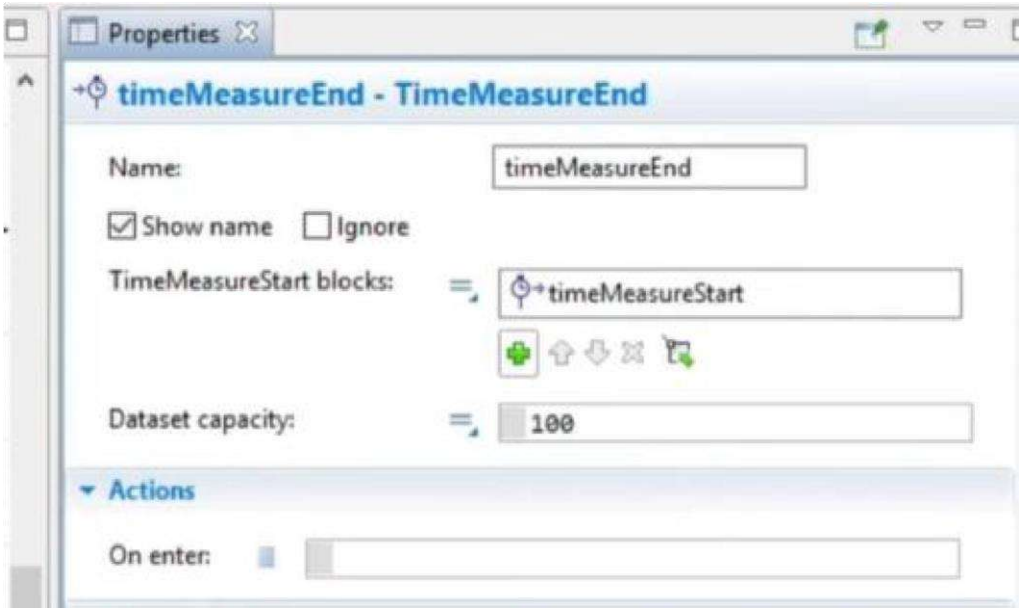


3.Move block **sink** to the right and rearrange the connection between **service** and **sink** to make space for a block after the connector.

4.Drag the **TimeMeasureEnd** block from the **Process Modeling Library** in the **Palette** view into the graphical editor and place it before the **sink** block. Name the block timeTotal

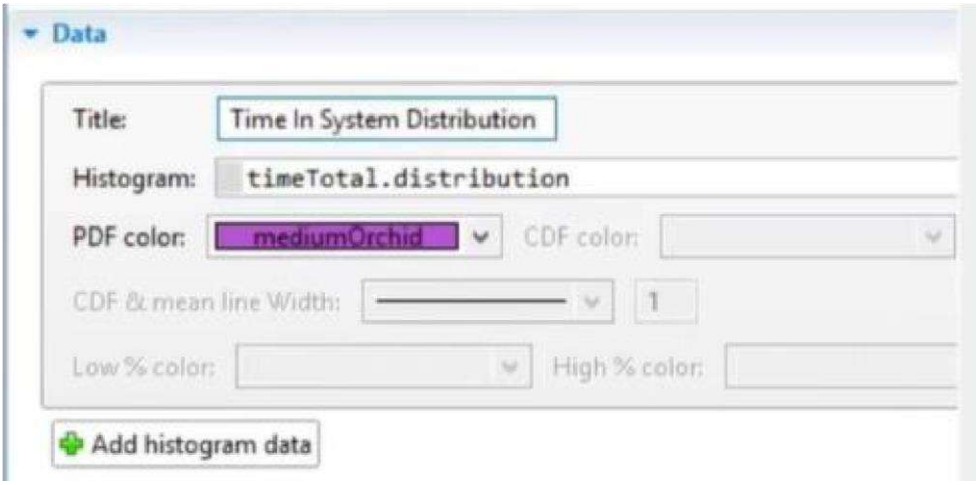
5.Make sure that the ports of the **timeTotal** block are properly connected both to **ATM** and **service** blocks.

6.Every **TimeMeasureEnd** block must have at least one **TimeMeasureStart** block specified in its properties to calculate the time distribution for agents. Specify our **timeMeasureStart** block in its **TimeMeasureStart blocks** parameter.

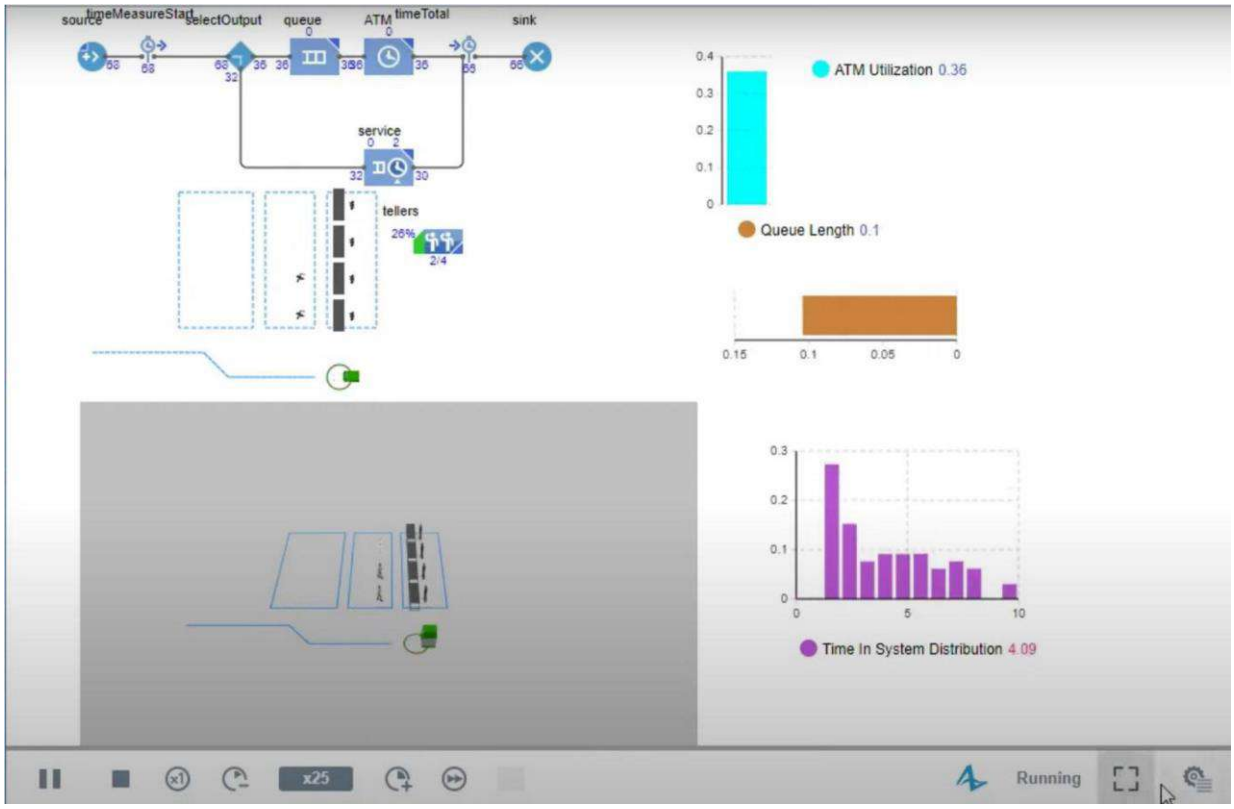


Add a histogram to indicate the collected statistics

1. Open the **Analysis** palette. Drag the **Histogram** element from the palette into the graphical editor.



Run the model. Set virtual time mode and observe distribution of customer's time in system.



Practical No: - 8

Aim: - Create defence model to stimulate aircraft behaviour

Code:-

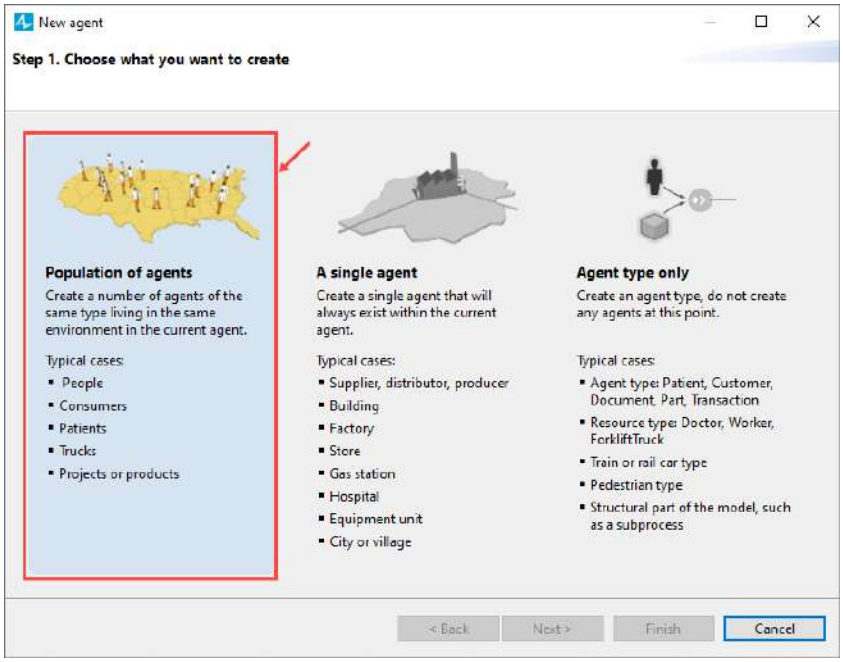
Create a new model

1. Click the **New** toolbar button toolbar button. The **New Model** dialog box will be displayed.
2. Define the parameters of the new model:
   - Specify the name of the model in the **Model name** field. Type Air Defense System
   - Set the new model location if necessary by clicking the **Browse** button and selecting the required folder.
   - Set **Model time units** to **seconds**.
3. Click **Finish** to create a new model.

We'll begin the design our model with creating a new population of agents that will represent 10 facilities.

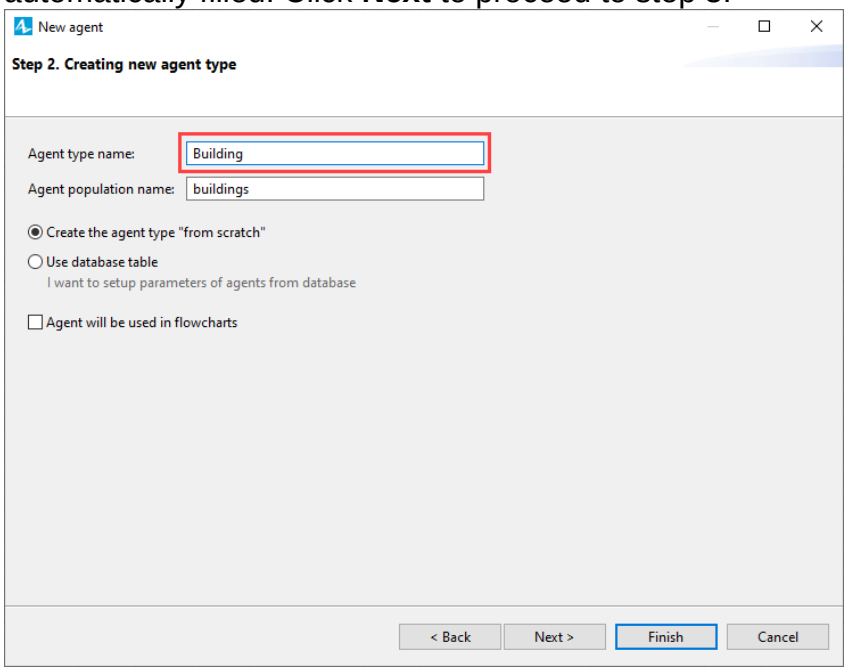
Create a new population of agents

1. Drag the **Agent** element from the **Agent** palette onto the Main diagram. The **New agent** dialog box will open.
2. Click **Population of agents**. You will be taken to the next step of the **New agent** wizard.

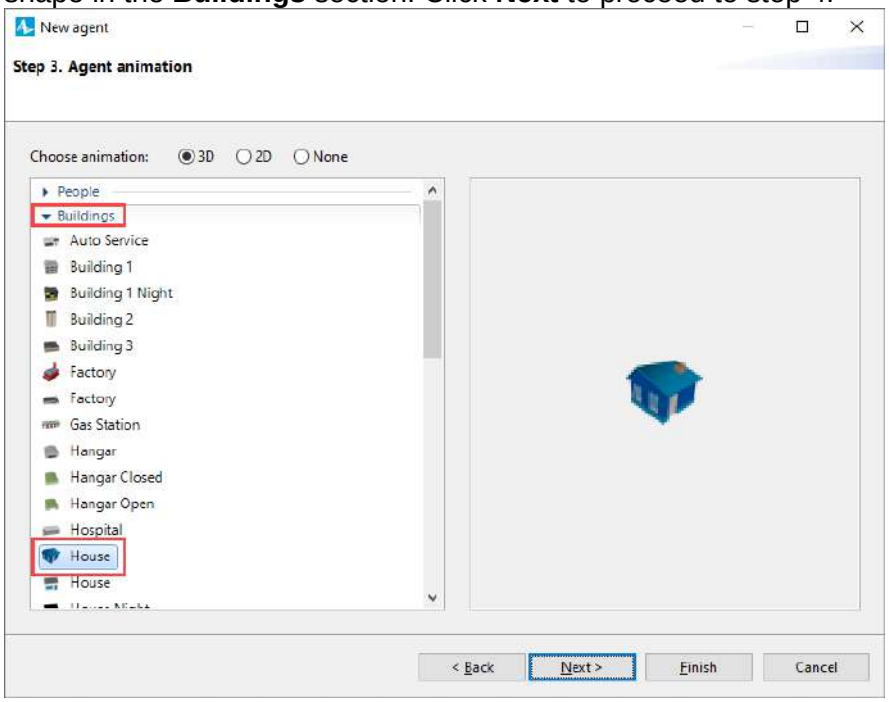


Type Building

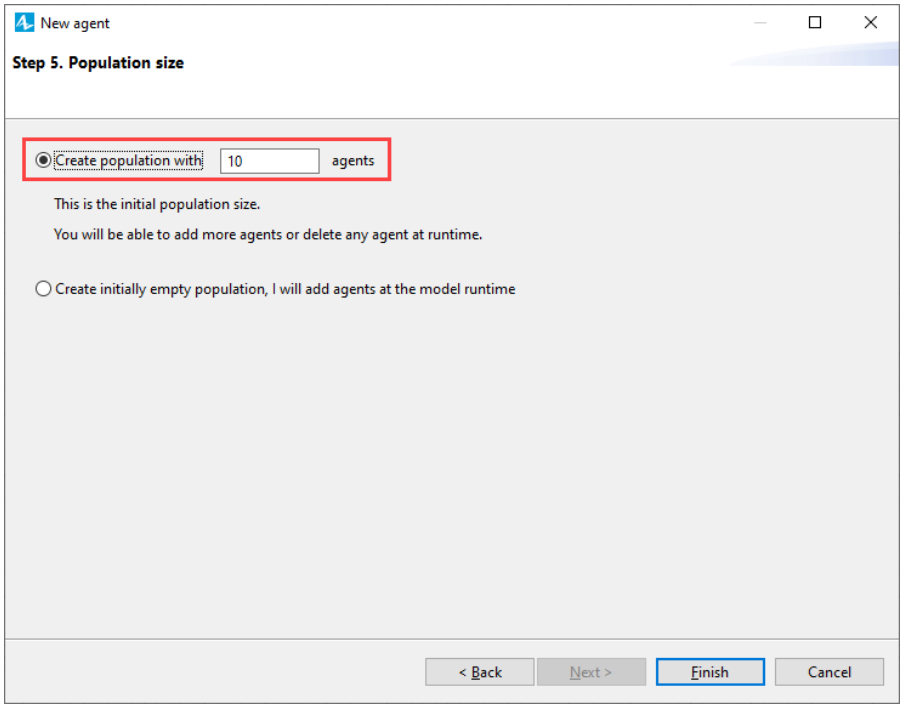
Type `Building` in the **Agent type name** field. The **Agent population name** will be automatically filled. Click **Next** to proceed to step 3.

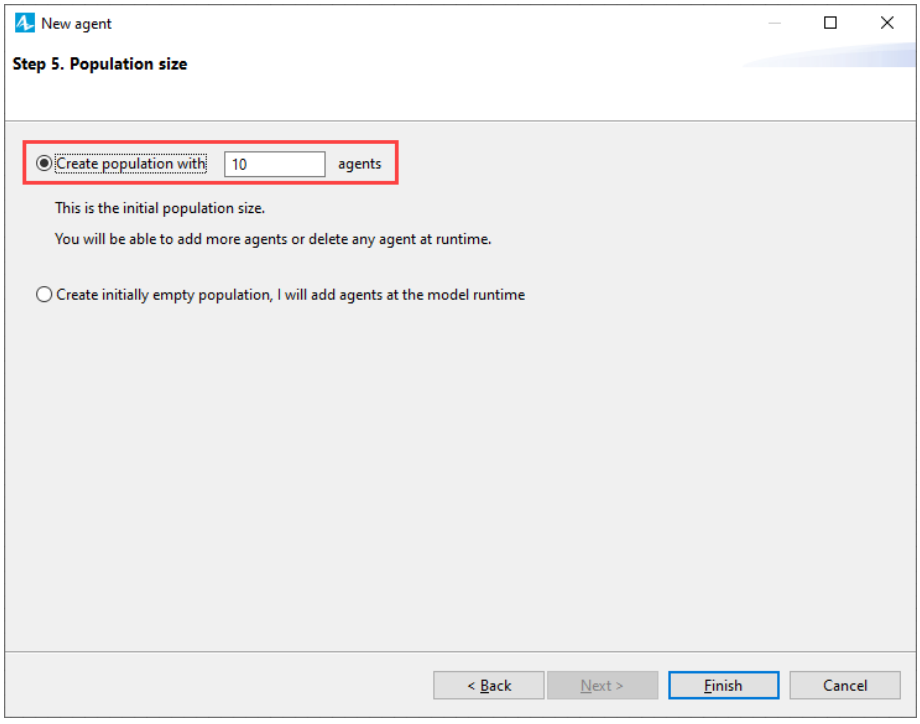


Leave the **Choose animation** parameter set to **3D** mode and choose the **House** animation shape in the **Buildings** section. Click **Next** to proceed to step 4.

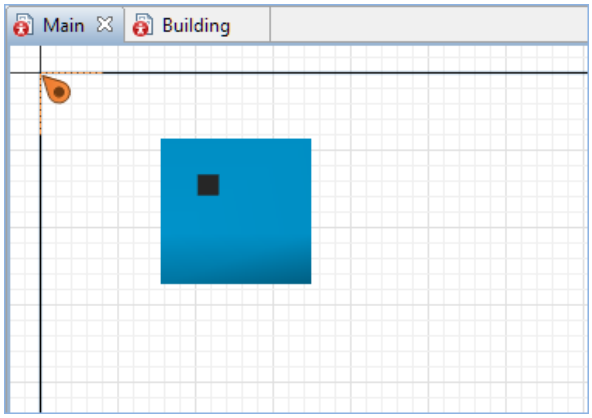


1. We will not set any agent parameters, you may click **Next** to proceed to step 5.
2. Set the **Create population with** parameter to `10`.





Click **Finish** to save the changes and create the customized population. We have created a population of agents and defined animation shape for them.



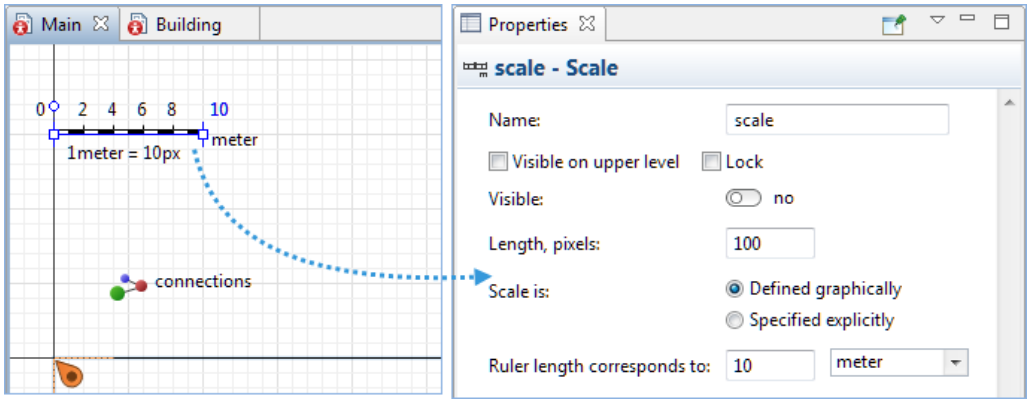
We want to place our agents to randomly selected points within the continuous space. By default the space has the size of 1000x600 pixels.

Define locations for the agents

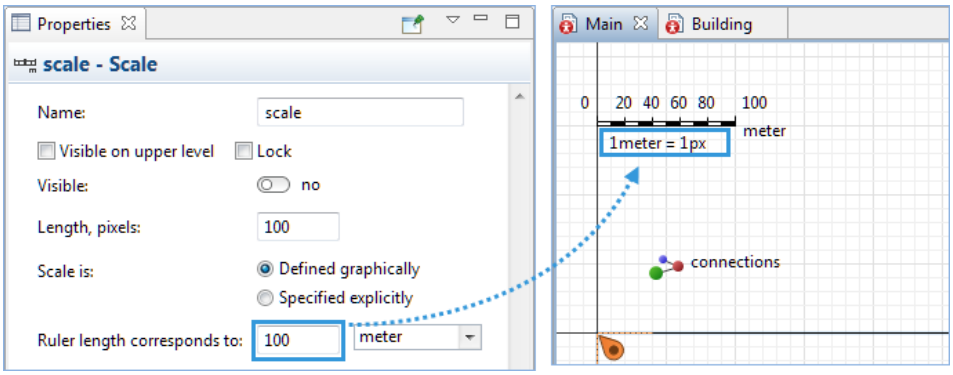
1. Click **Main** in the model tree to open its **Properties**.
2. Expand the **Space and network** section of the Main agent type's properties.
3. Set the **Layout type** to **Random**.

Adjust the scale of the model

1. Click the **Scale** element on the Main diagram to open its **Properties**.



2 Set the **Ruler length corresponds to** parameter to 100. The scale on the ruler will change.

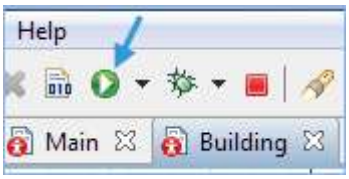


We modified the scale of the model, as a result the building shape changed its size. As you can see in the image above, if we will keep the animations of our agents resized corresponding to the model scale, these agents (such as building here) will become invisible in our model's animation scene. So we will use greater scales for the agents to make them visible during the animation.

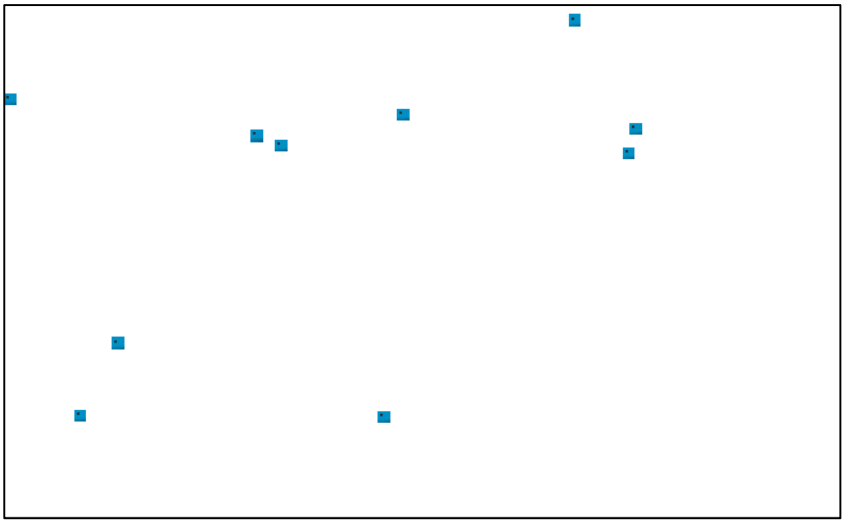
We have completed setting up the population of agents and may now run our model for the first time to observe the model behavior.

Run the model

1. Click the **Run** toolbar button.



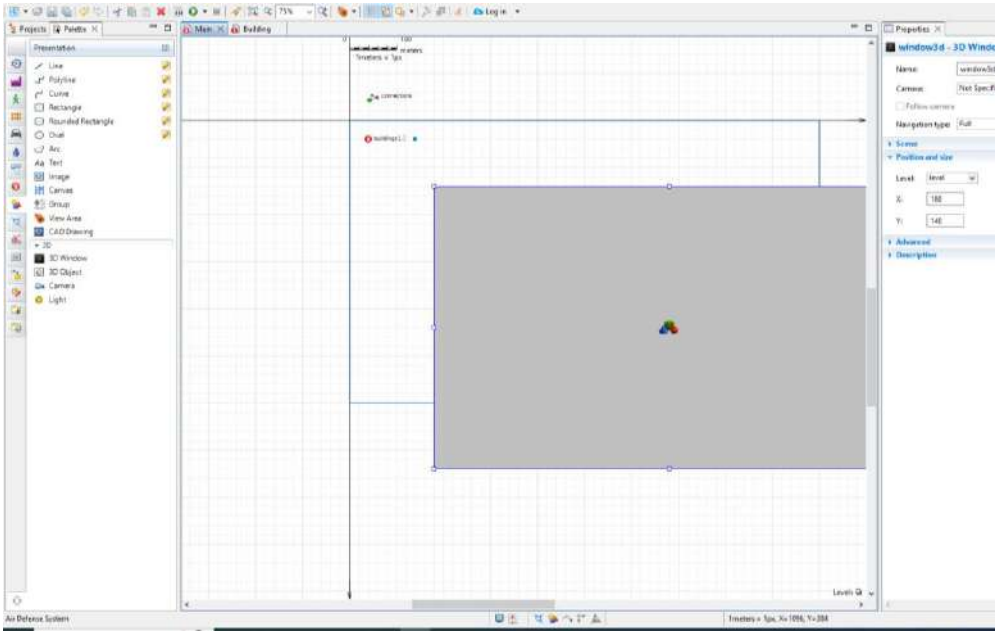
The model window will appear and the model will launch automatically. You will see the 10 building shapes randomly distributed in a 500 by 500 pixel space. This is the space size set by default. We do not need to change it as we will customize the location of the facilities.



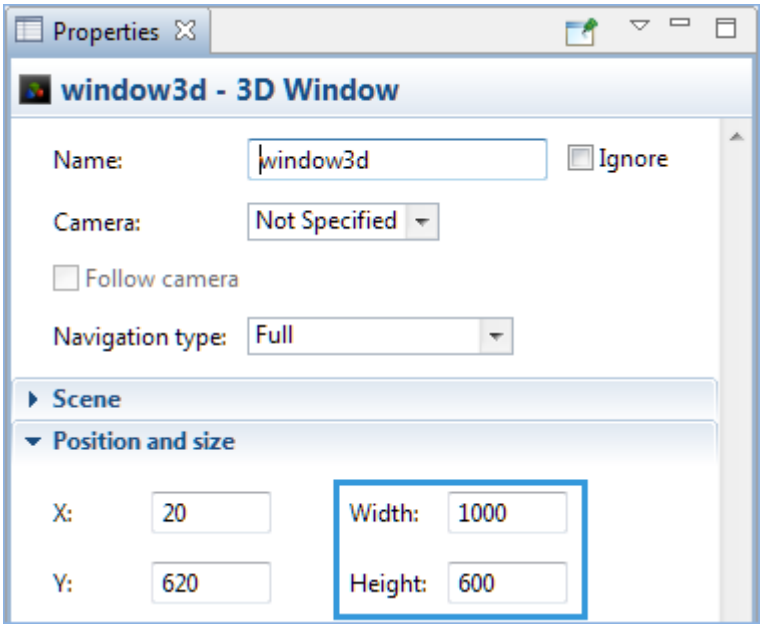
Now we will create 3D animation for our model. First of all we need to add 3D window to our Main diagram. The 3D window plays the role of a placeholder for 3D animation. It defines the area on the presentation diagram where 3D animation will be shown at runtime.

Add 3D window

1. Drag the **3D Window** element from the **3D** section of the **Presentation** palette to the graphical editor. Place the window below the model window frame.



Navigate to the **Properties** view of the **3D window** and set its **Width** and **Height** parameters to `1000` and `600` respectively.



Now we will run the model and switch between the 2D and 3D views in the model window.

Compare 2D and 3D views

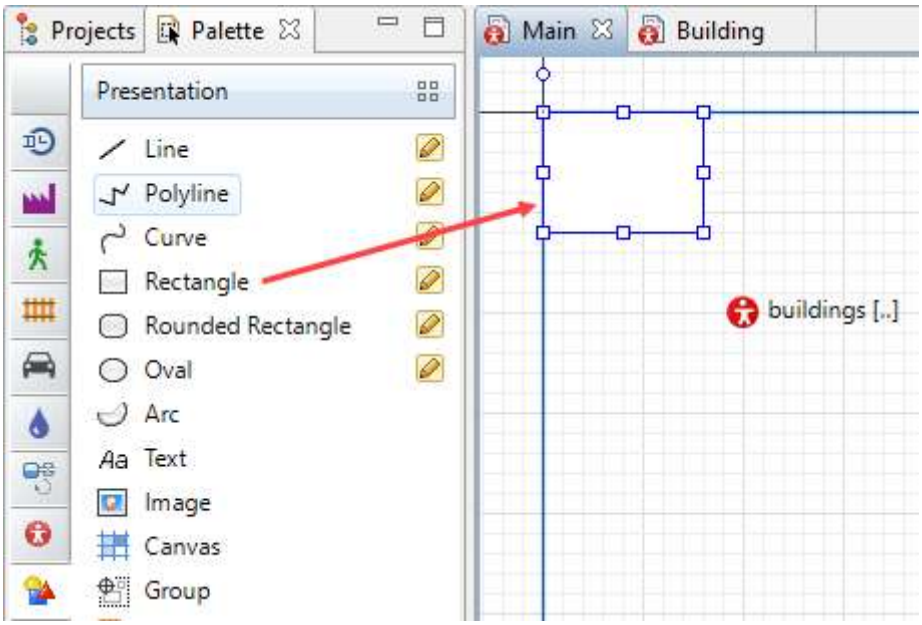
1. Click **Run** in the toolbar to run the model.
2. When you create a 3D window, AnyLogic adds a view area that allows you to easily navigate to the 3D view at runtime. To switch to this 3D view while the model is running, open the developer panel by clicking the **Developer panel** control in the right corner of the control panel. In the developer

panel, expand the [▾] **select view area to navigate** list and select **[window3d]** from the list.

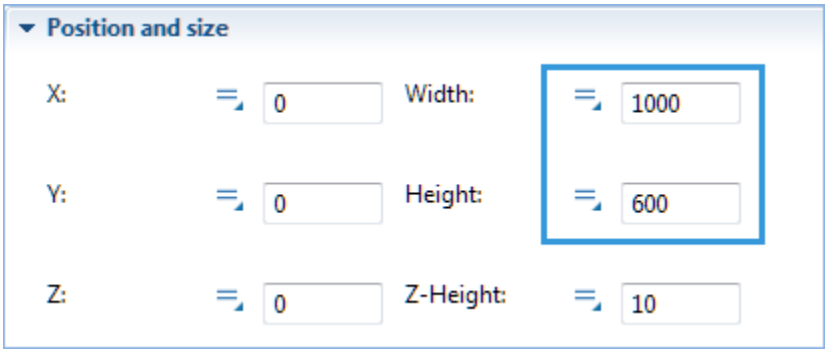
The view mode will change to 3D.

Draw the surface

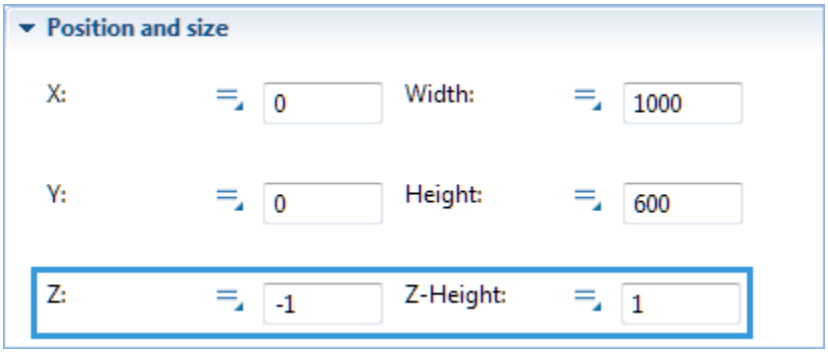
1. Drag a ▭ **Rectangle** shape from the **Presentation** palette onto the editor of Main, place its top left corner at (0, 0).



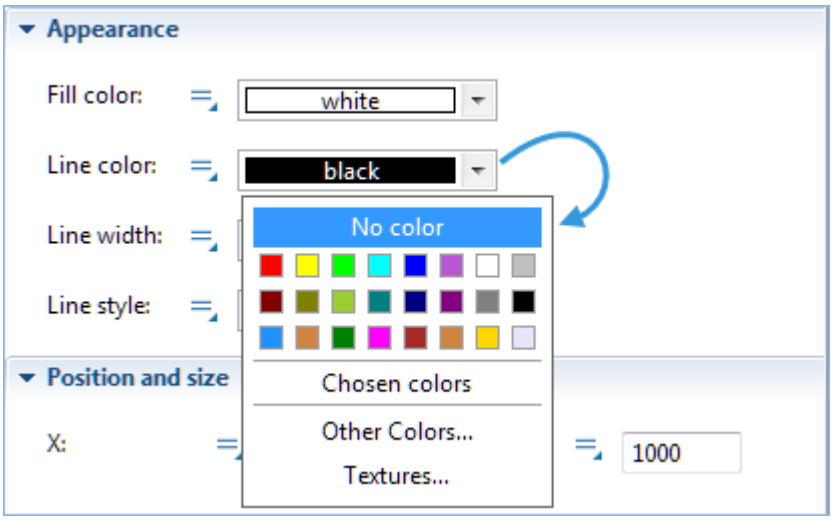
Navigate to the **Position and size** section of the rectangle's **Properties** and set its **Width** and **Height** parameters to 1000 and 600 respectively, hence the area size is 1000*600 meters.



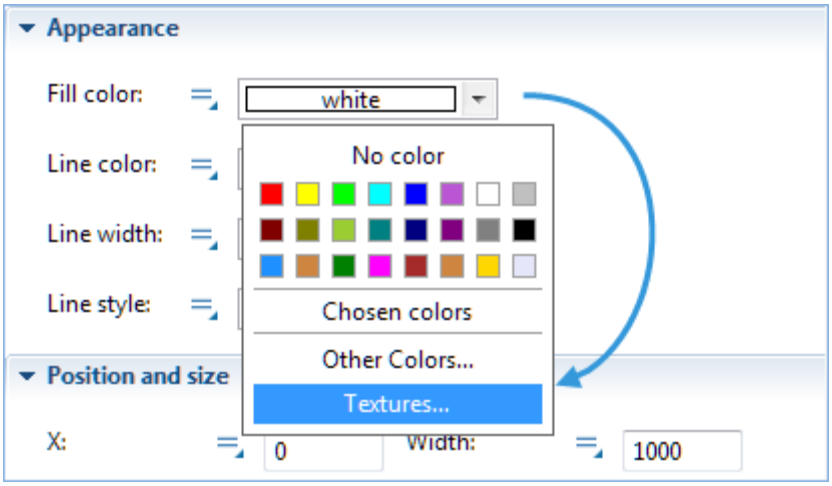
Now set the **Z** and **Z-Height** parameters to `-1` and `1` respectively.



In the **Appearance** section of the rectangle's **Properties** set the **Line color** option to **No color**.



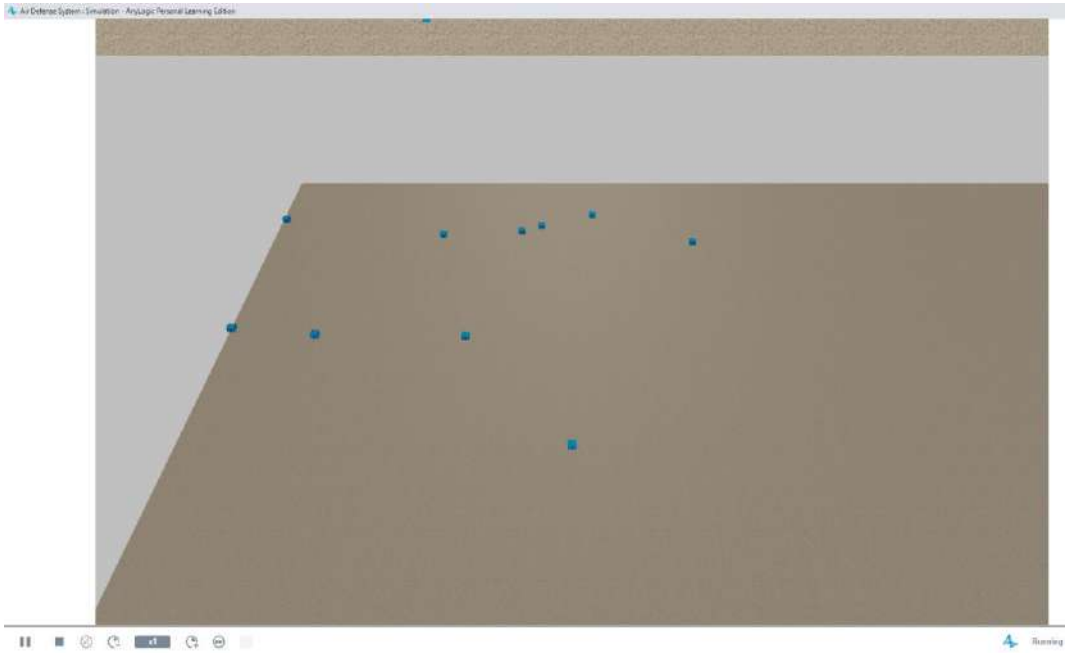
Now we will define the texture of the rectangle area. Click the **Fill color** control and select **Textures** to open a dialog box with available textures.



Click **Earth** texture to set it as the fill colour for the rectangle shape.

1. Click **OK** to close the dialog box. The rectangle shape will change it appearance.

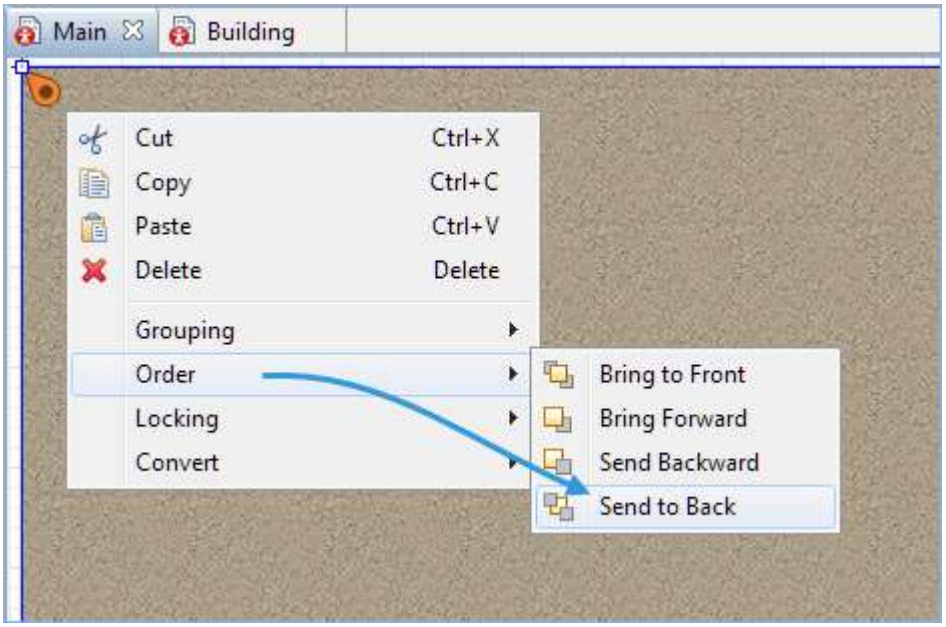
Now run the model and observe the new layout both in 2D and 3D modes. Observe the new layout Run the model. You will observe the model in the 2D view. As you can see, we have the new layout here but no buildings can be seen.



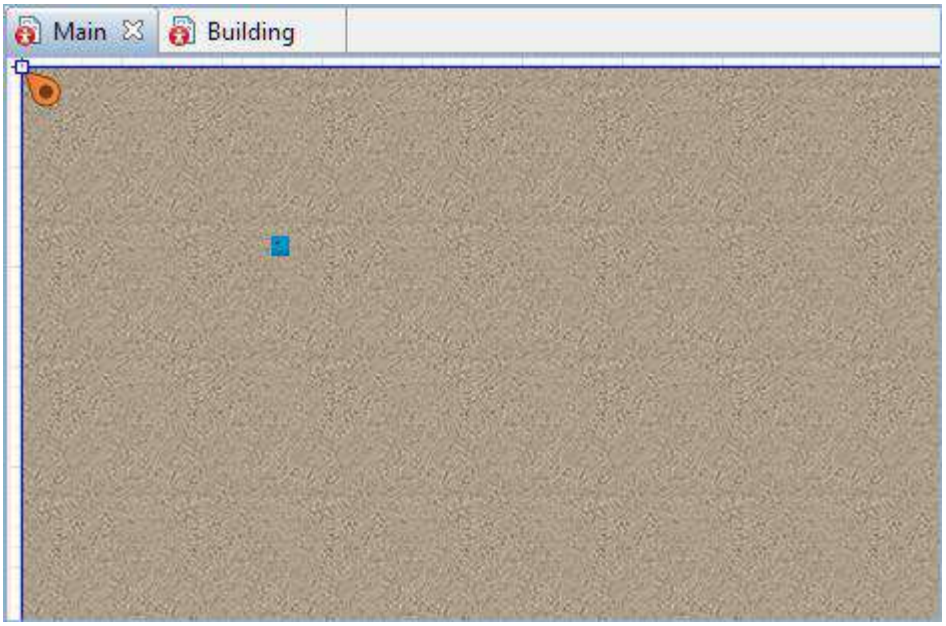
In order to make buildings visible in the 2D view we must reorder the shapes in the design.

Reorder shapes

1. Right-click the rectangle shape to open its context menu. Then click the **Order** menu item and select the **Sent to Back** sub-item from the list of available orders.



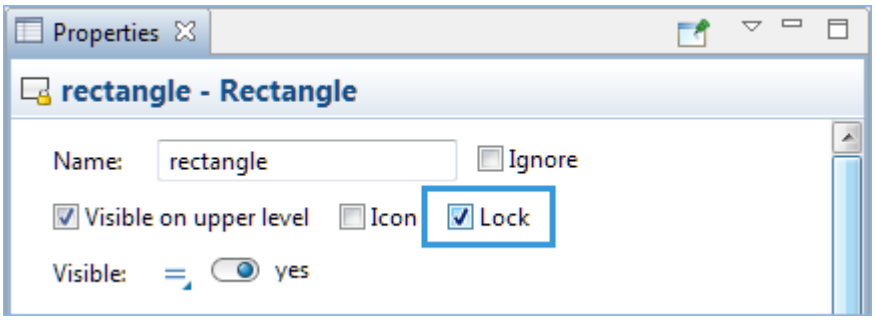
The 2D building shape will now be visible.



Now the model should have correct visualization in both 2D and 3D view. Prior to running the model to check if everything works like it should, we will lock the **Rectangle** shape and add camera.

Lock the rectangle shape

1. In the **Properties** view select the **Lock** option to lock the shape. This will prevent occasional selection of the rectangle while working with other elements on the diagram.



Add a camera

1. Drag the **Camera** element from the **3D** section of the **Presentation** palette onto the graphical editor. You will see the camera icon in the graphical editor.
2. Direct the camera at the buildings presentation shapes (the positioning is described in details in the Camera article).

It is time to run the model and adjust the camera position to the desired one. We will save that position afterwards to have the camera always located at the set position.

Adjust the camera position

1. Run the model and switch to 3D mode.
2. Right-click the 3D model presentation to open a pop-up menu. Navigate to the **Camera** menu item and select **camera** sub-item. You will switch to the camera view.
3. Position the camera to have the best possible picture while observing the model in the runtime (for more details on navigating the 3D scene, see Moving and rotating 3D scene at runtime).
4. When done, right-click the 3D model presentation and click **Copy camera's location**.

You may stop the simulation now. We have set up our camera and copied its position, now we need to apply the new camera position.

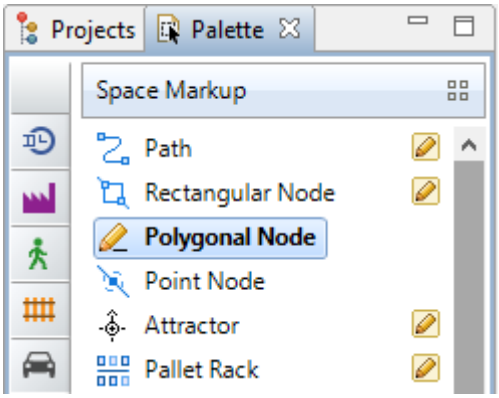
Apply new camera position

1. Click the Camera element to open its **Properties** view.
2. Click the **Paste coordinates from clipboard** button in the **Properties** view. The rotation parameters will change.

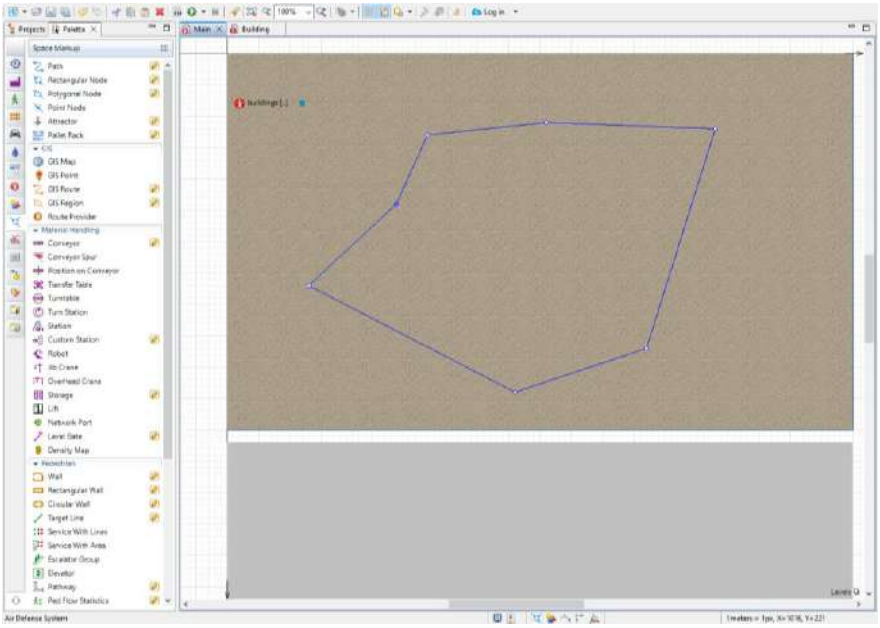
In the last step of this phase we will create a protected area, within which the buildings will be placed. The area will further be equipped with the air defense system, eliminating the air attacks on our facilities.

Define protected area

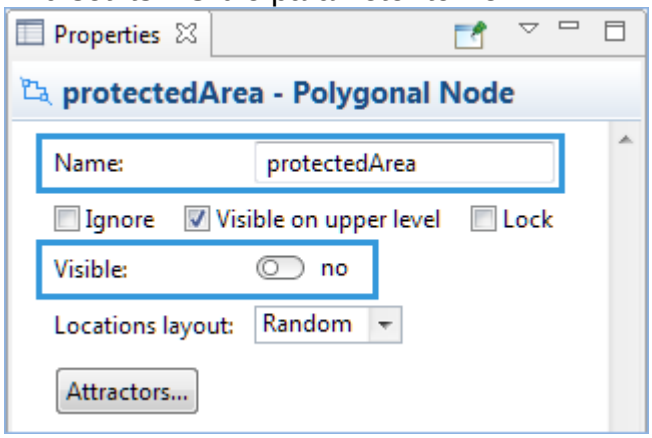
1. Double-click the **Polygonal Node** element in the **Process Modeling Library** palette to activate drawing mode.



1. Click on top of the rectangle to start drawing the protected area.
2. Finish drawing with a double-click on the last node.



Navigate to the element's **Properties**, name it protectedArea
And set its **visible** parameter to **no**.

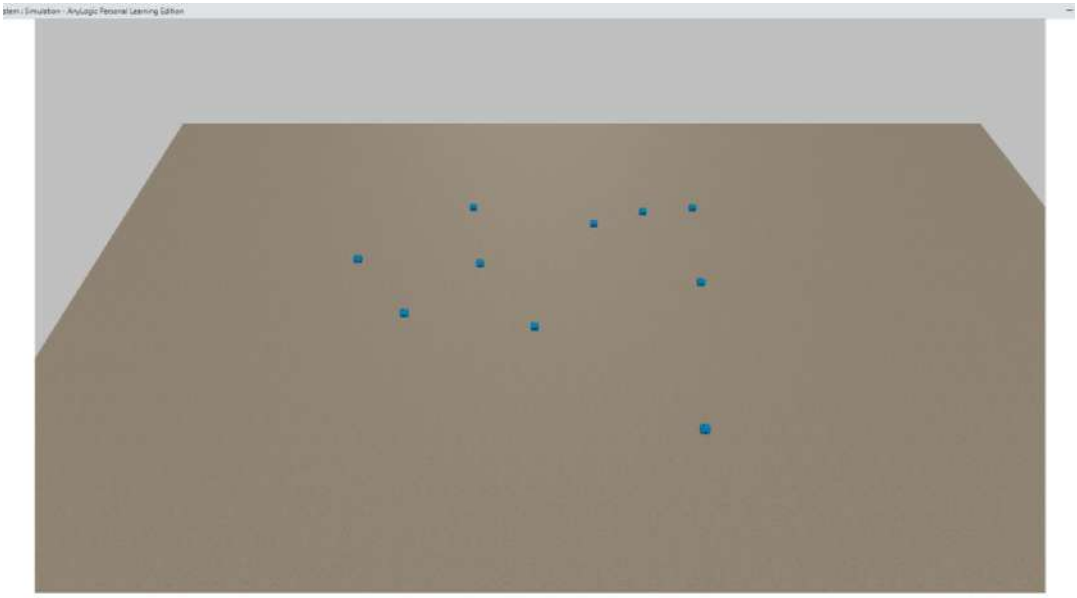


We have created a new area, now we need to locate our facilities within its boundaries.

Relocate the facilities

1. Click the **buildings** population on the Main diagram to open the **Properties** view.
2. Navigate to the **Initial position** section and set the **Place agent(s)** parameter to **in the node**. The **Node** parameter will appear below.
3. Click the set by default **None** value for the **Node** parameter and select **protectedArea** from the drop-down list of available nodes.
4. Now open the properties of Main and navigate to the **Space and network** section.
5. Set the **Layout type** to **User-defined**.

Finally, we may run the model. The buildings will be located within the area defined by the polygonal node



Phase 2. Creating bombers

In the previous phase we created a new model, populated it with facilities, defined animation shapes for the facilities, added 3D view and camera to observe the running model and finally added protected area.

In this phase we will add bomber aircrafts to our model, define their behavior and mission.

Let us start with creating a new population of agents that will represent the bombers.

Create a new agent type

1. Drag the **Agent** element from the **Agent** palette onto the `Main` diagram. Place it nearby the **buildings** population to the left of the model animation.
2. Click **Population of agents**. You will be taken to the next step.
3. Specify `Bomber`

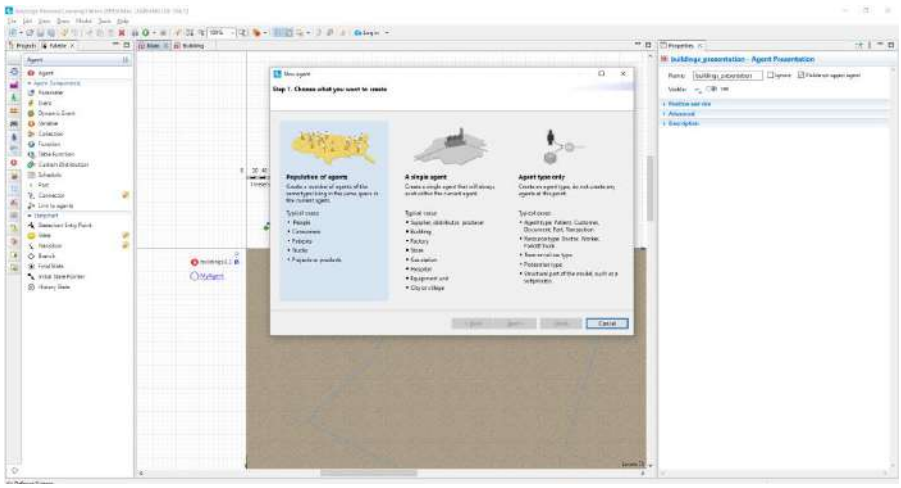
   in the **Agent type name** field on the third step of the agent wizard. Click **Next**.

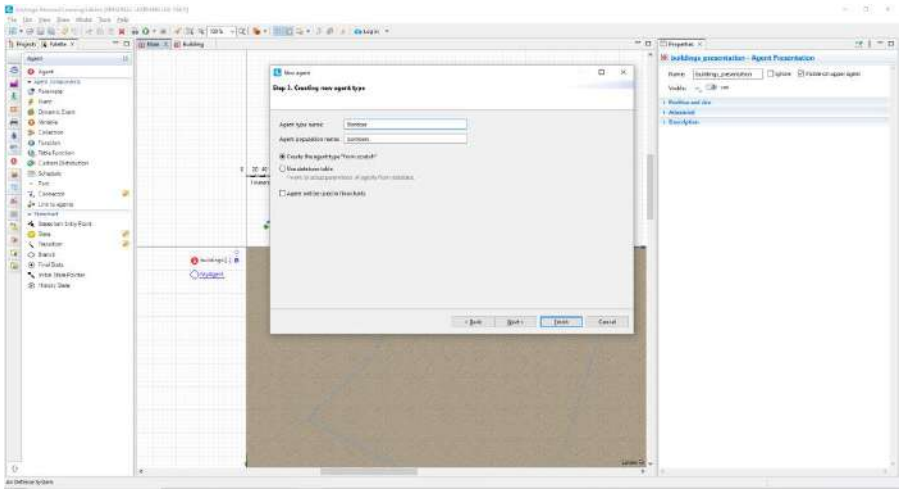
4. Choose the **Bomber** animation shape in the **Military** section. Click **Next**.
5. Define the population's parameter:
   - Click **add new...** to create a parameter.
   - Type `target`

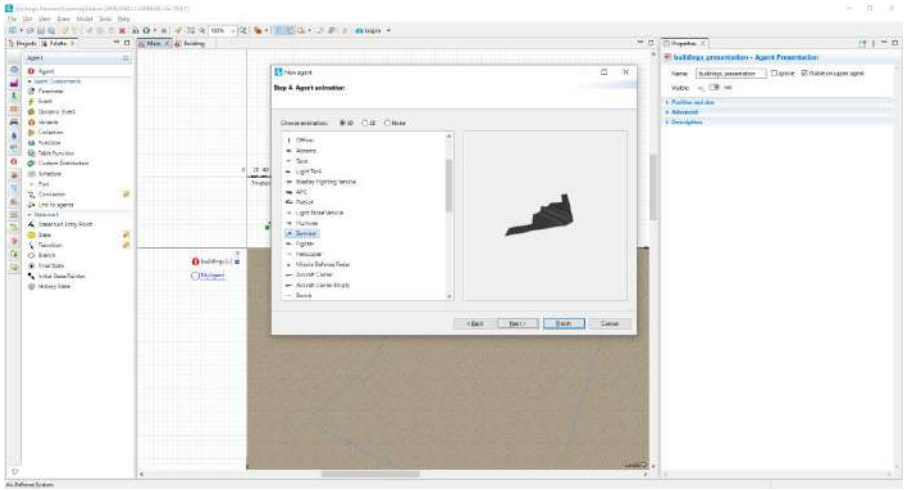
     into the **Parameter** field.

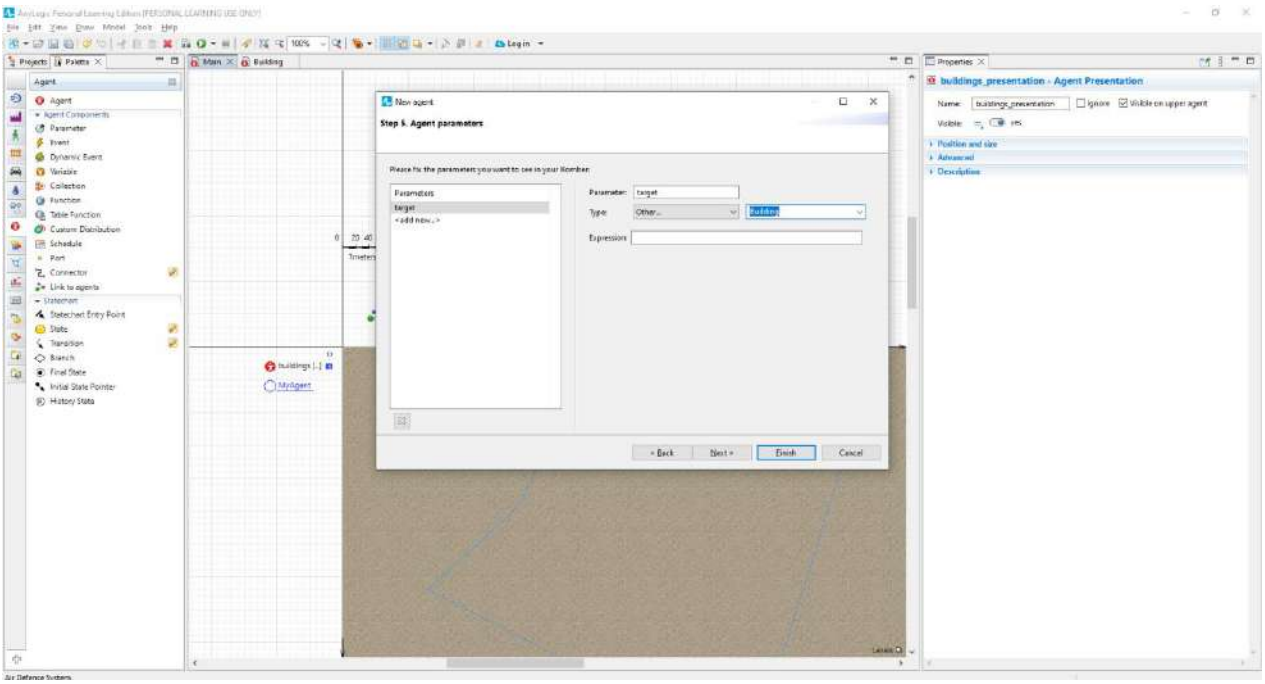
   - Set **Type** to **Other** (additional drop-down list will appear next to it).
   - Select **Building** from the additional drop-down list of the parameter's type (This will be the target building in the bomber mission).

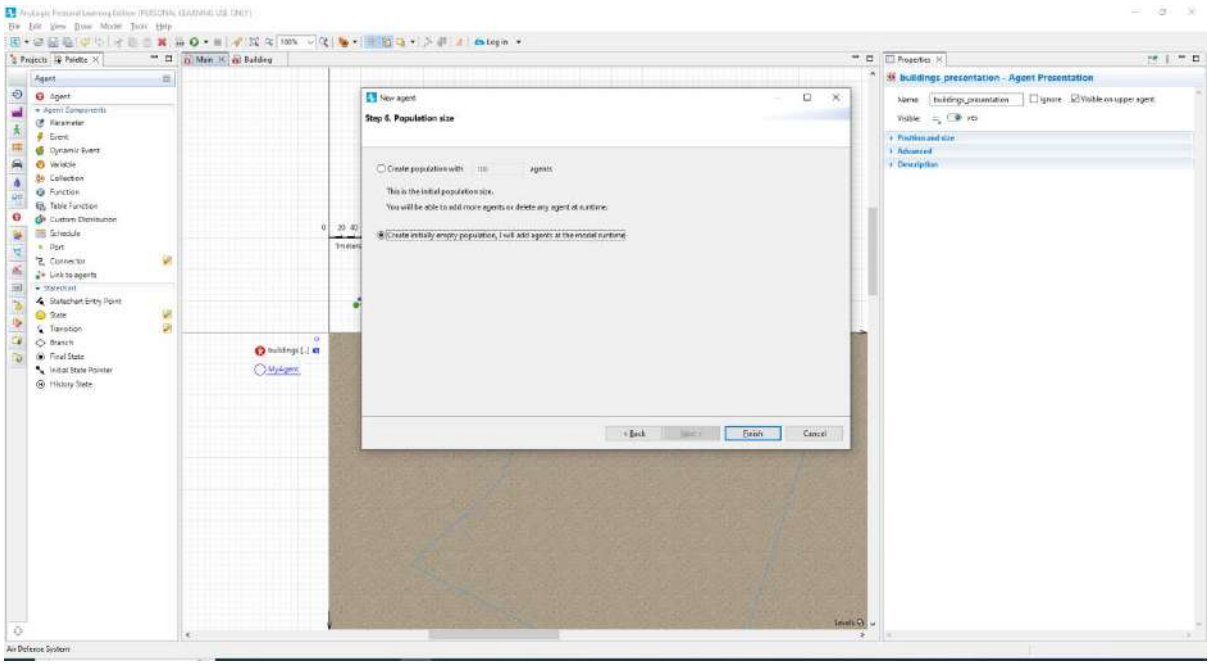
Click **Next**.

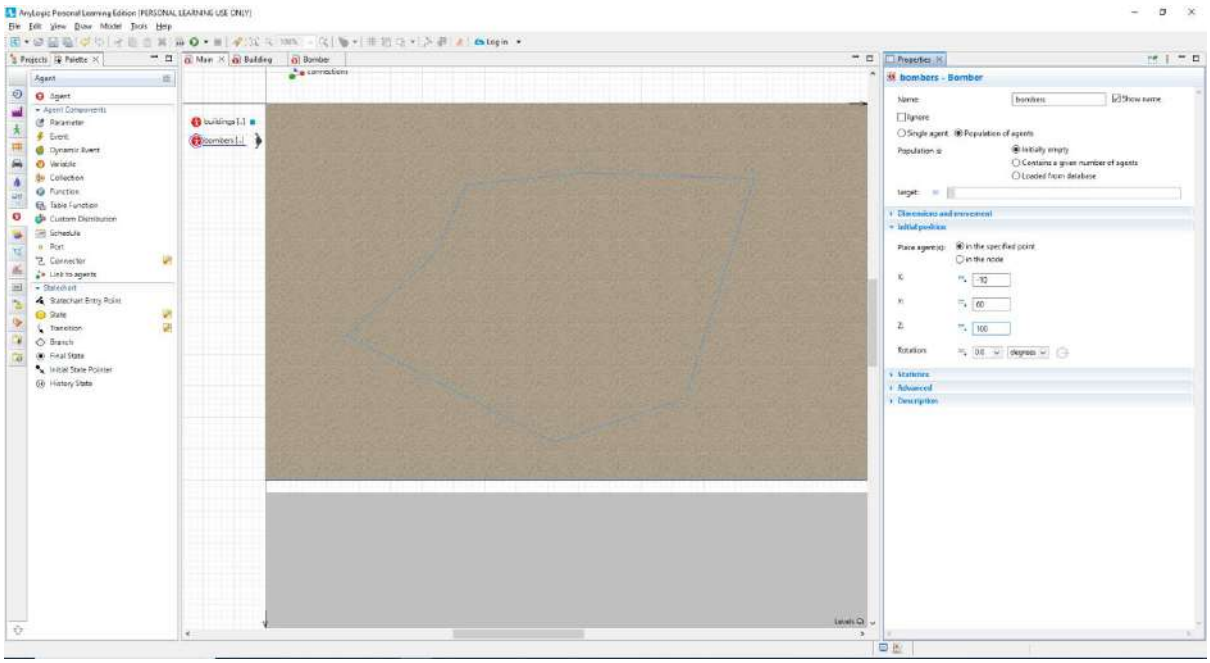






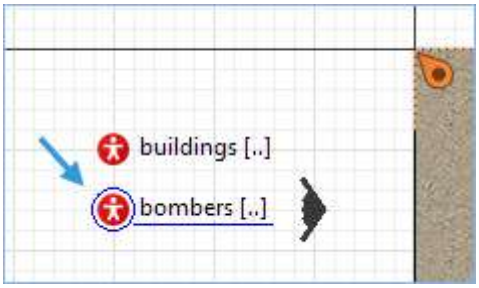

New agent
Step 5. Agent parameters
Parameters
target
< add new... >
Parameter: target
Type: Other... Builder
Expression:
< Back
Next >
Finish
Cancel
buildings_presentation - Agent Presentation
Name: buildings_presentation
Ignore
Visible on upper agent
Visible: yes
Position and size
Advanced
Description
buildings [..]
MyAgent


New agent
Step 6. Population size
Create population with 100 agents
This is the initial population size.
You will be able to add more agents or delete any agent at runtime.
Create initially empty population, I will add agents at the model runtime
< Back
Finish
Cancel
buildings_presentation - Agent Presentation
Name: buildings_presentation
Ignore
Visible on upper agent
Visible: yes
Position and size
Advanced
Description
buildings [..]
MyAgent

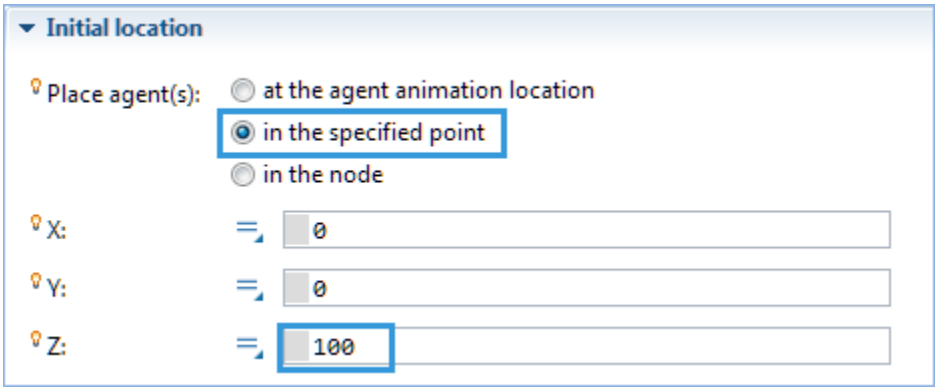


Define initial location

1. Click the **bombers** population element to open its properties.



2. Navigate to the **Initial position** section of the element's **Properties**, set its **Place agent(s)** parameter to **in the specified point** to manually specify the point of appearance and then type `100` into the **Z** coordinate's field to specify the altitude at which the bombers will be flying into the scene.



The next step is to define the velocity of the aircraft. It is done right here, in the same **Properties** view, just above the **Initial position** section.

Define the velocity of the aircraft

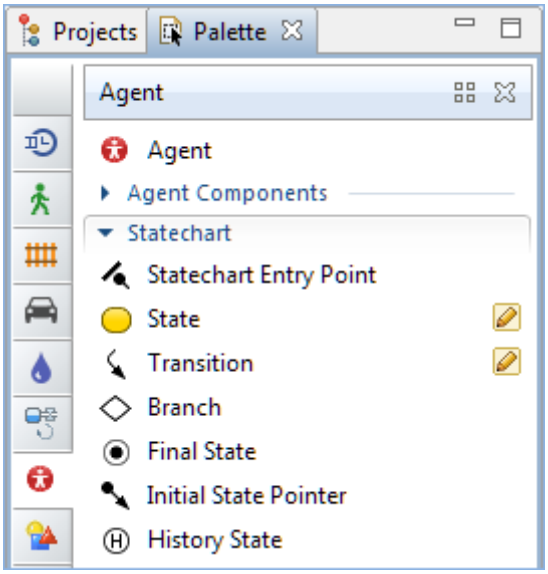
1. Navigate to the **Dimensions and movement** section and set the **Initial speed** parameter to 600, then click the speed units drop-down menu to the right of it and select **kilometers per hour**.

Now that we have created bombers and defined their animation, initial location and target, we can move on to create the initial version of their behavior using the statechart.

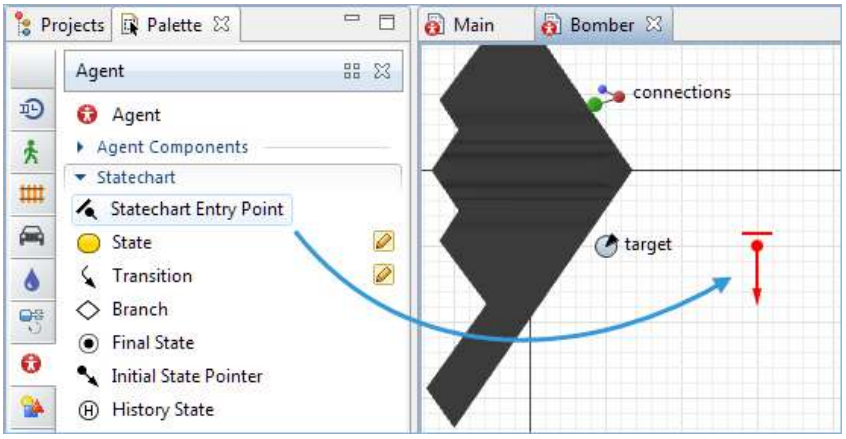
Once created, the bombers will be heading towards the buildings (it is set by the **target** parameter). We will now define the further behavior, which will make the bombers gradually get to a lower altitude and then return to the initial point where they will delete themselves from the model.

Create the behavior defining statechart

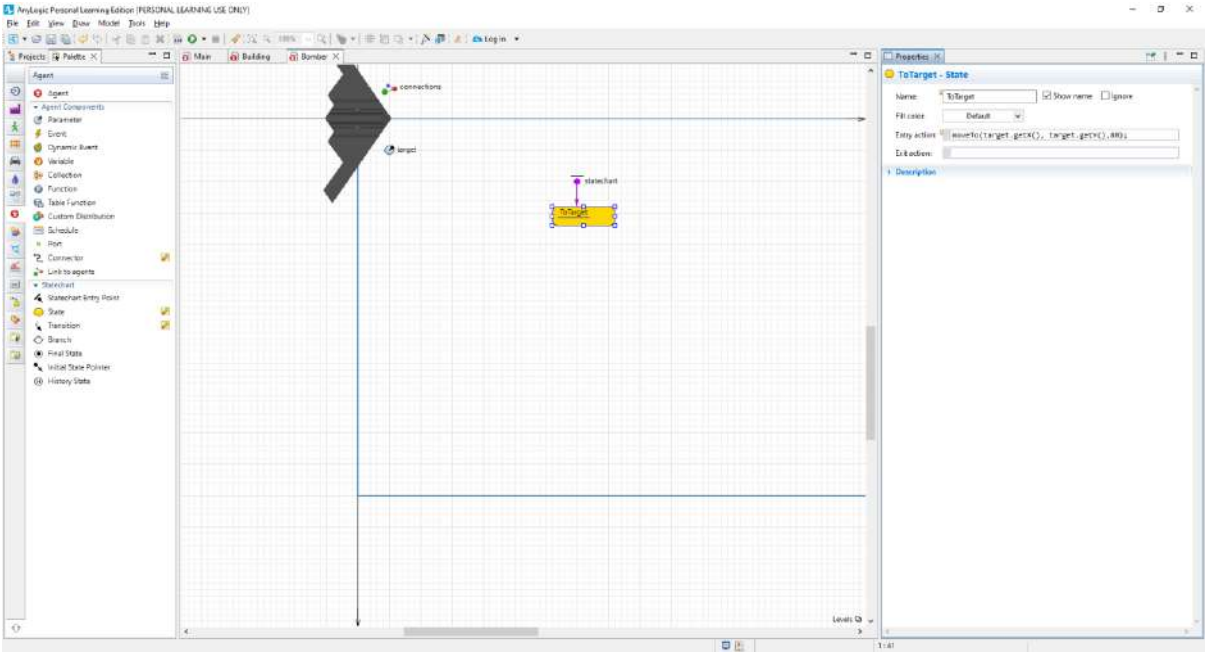
1. Double-click Bomber in the **Projects** view to open its diagram.
2. Switch to the **Agent** palette to use its statechart section.



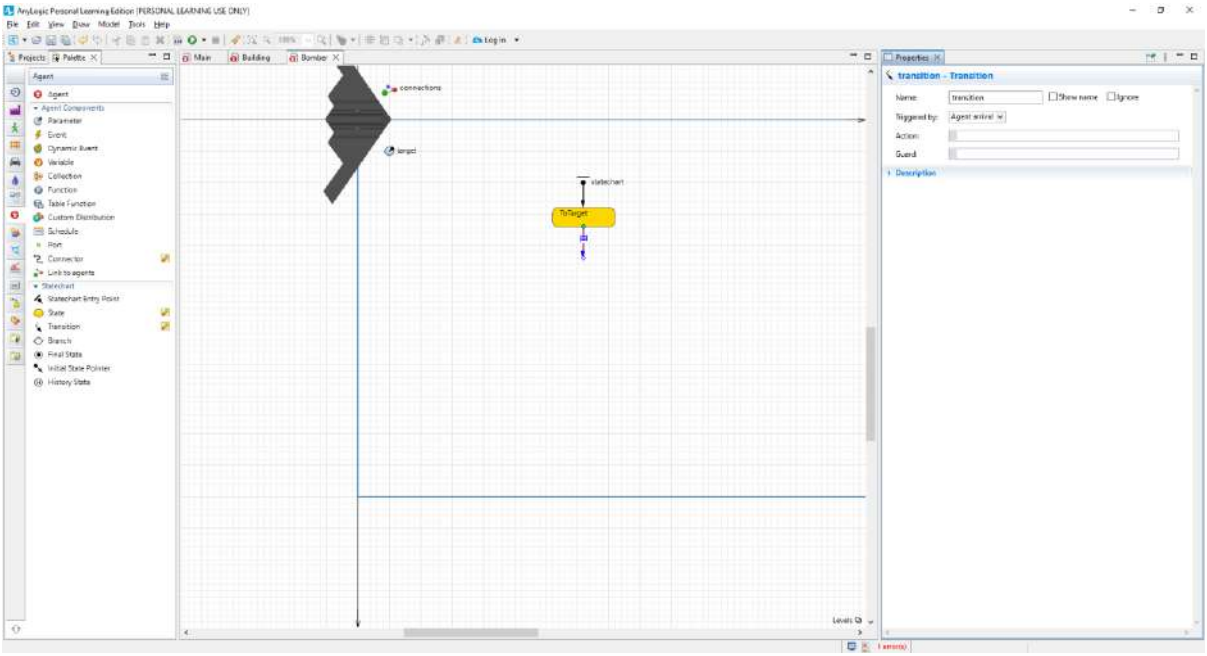
3. Drag the **Statechart Entry Point** element to the graphical editor of the Bomber agent.

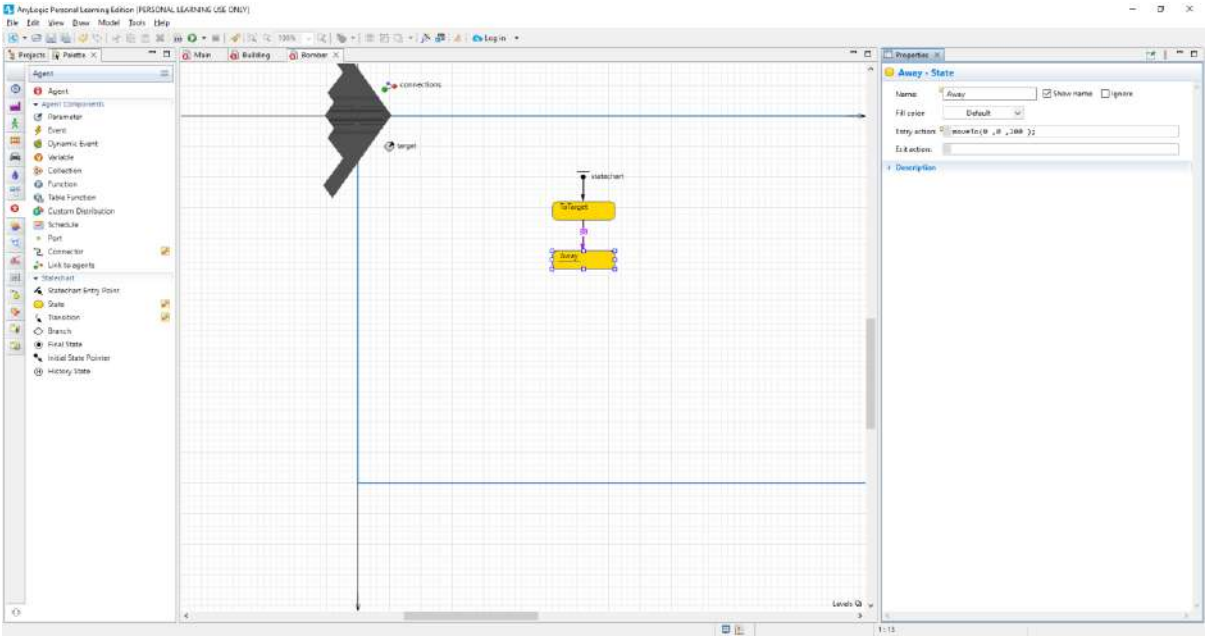


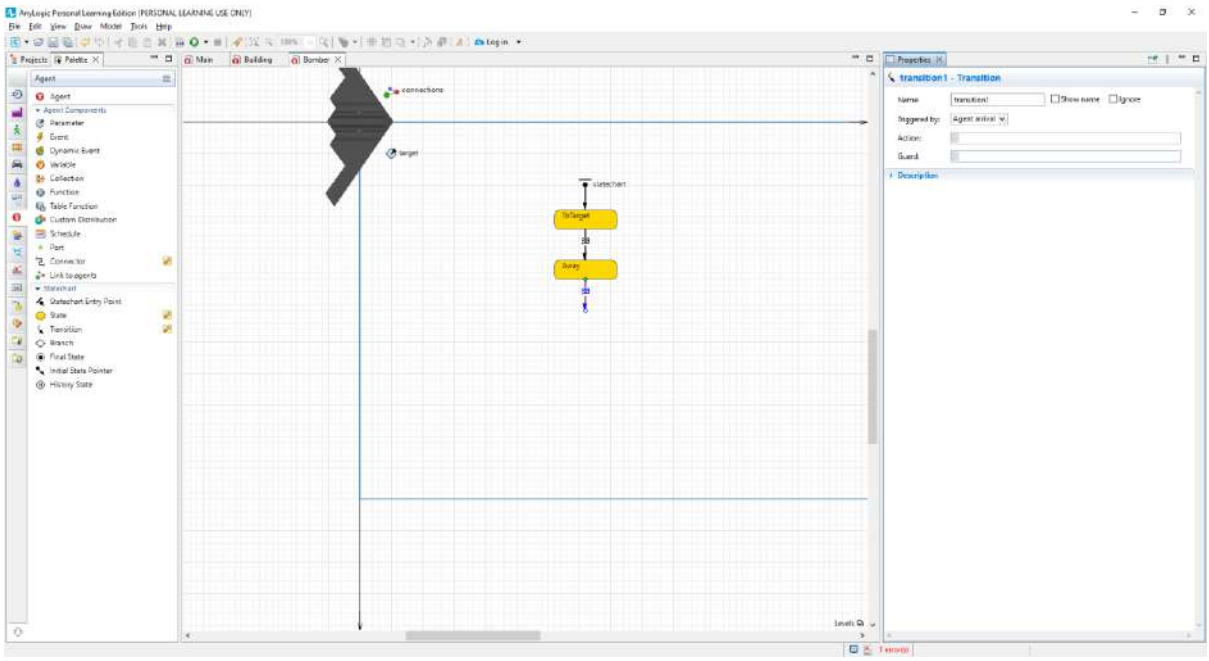
4. Double-click the **State** element to activate the drawing mode and draw a state rectangle below the previously drawn **Statechart Entry Point**.
5. Name it ToTarget and drag it up to the entry point until you see the green dot indicating that the two elements can be connected.

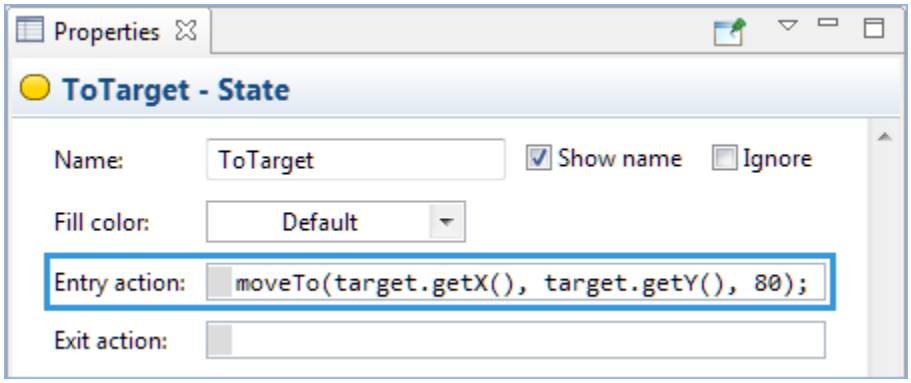


Modify the **Entry action** field in the **Properties** view of the **ToTarget** state. Type in the following code to define the coordinates of the bombers' location. By specifying the code we will define the altitude different from the one specified in the **Initial position** section. It will make the bombers get to a lower altitude on their way to the targets, which are represented in our case by the buildings.

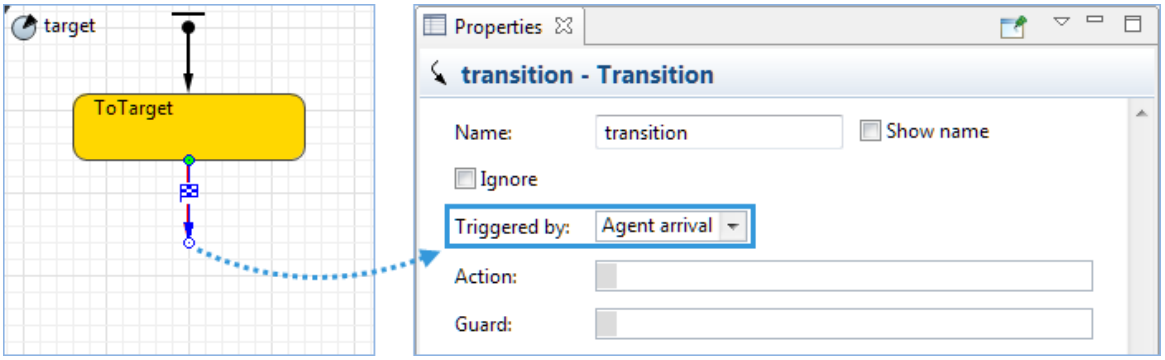

connections
target
statechart
ToTarget
transition - transition
Name:
transition
Triggered by:
Action:
Guard:
Description


connections
target
statechart
ToTarget
Away
Away - State
Name:
Away
Fill color:
Default
Entry action:
Exit action:
Description



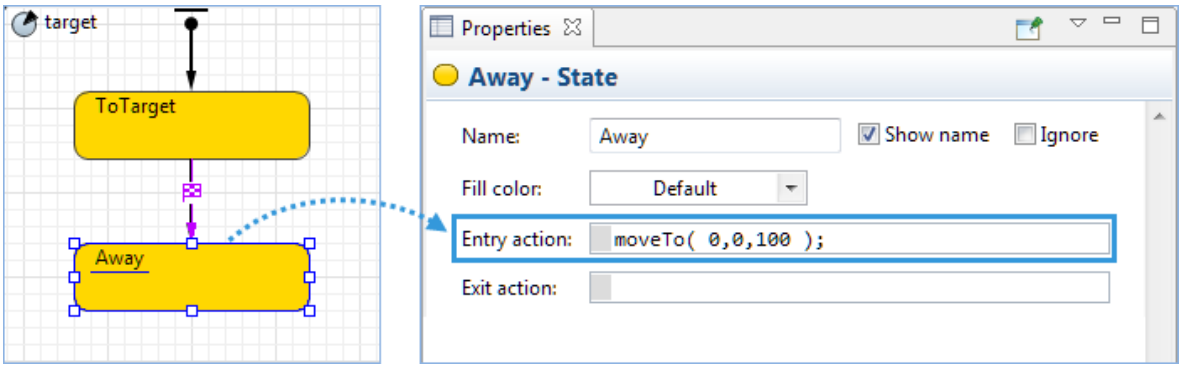


1. Drag the **Transition** element to the **ToTarget** state on the diagram and connect them. Then navigate to its properties and set the **Triggered by** parameter to **Agent arrival**.

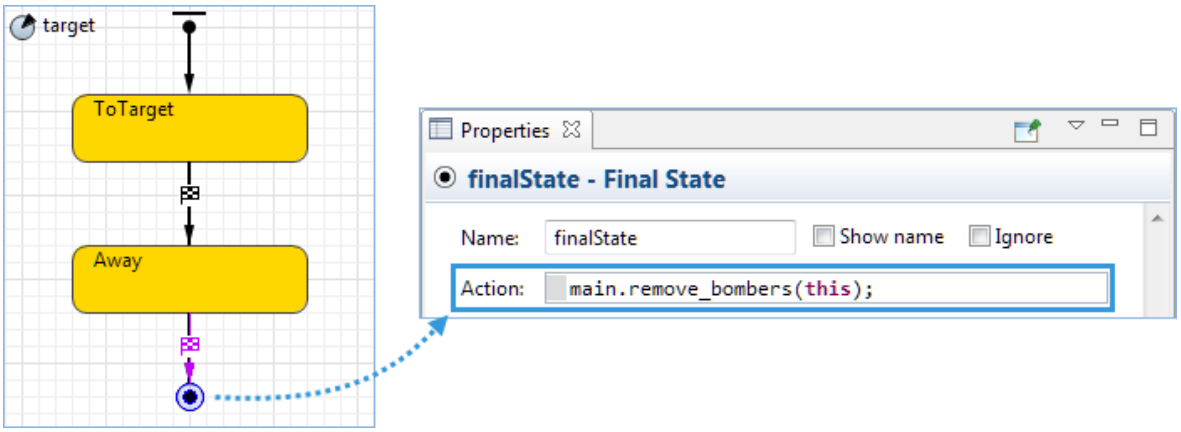


2. Now drag another **State** element to the agent diagram and connect it to the **Transition** element. Name it Away

. Then navigate to its **Properties** view and type the following code into the **Entry action** field, which will make the bombers get back to their previous altitude once they have reached their targets.



3. Drag another **Transition** element to the agent diagram and connect it to the **Away** element. Then navigate to its properties and set the **Triggered by** parameter to **Agent arrival**. The two transition elements on the statechart diagram must be identical.
4. Finally, drag the **Final State** element to the agent diagram and connect it to the last added **Transition** element. Then navigate to its **Properties** view and type the following code into the **Entry action** field, which will remove the bombers that have completed all the steps of this statechart.



We have completed the behavior defining statechart. As you can see we call the moveTo() function in the **On enter** field of both states to initiate movement of the agent. The transition from one state to another is triggered by the agent arrival. This pattern is very common in the agent based models.

The next step is to create the mission assignment. We will do it with the help of the **Event** element.

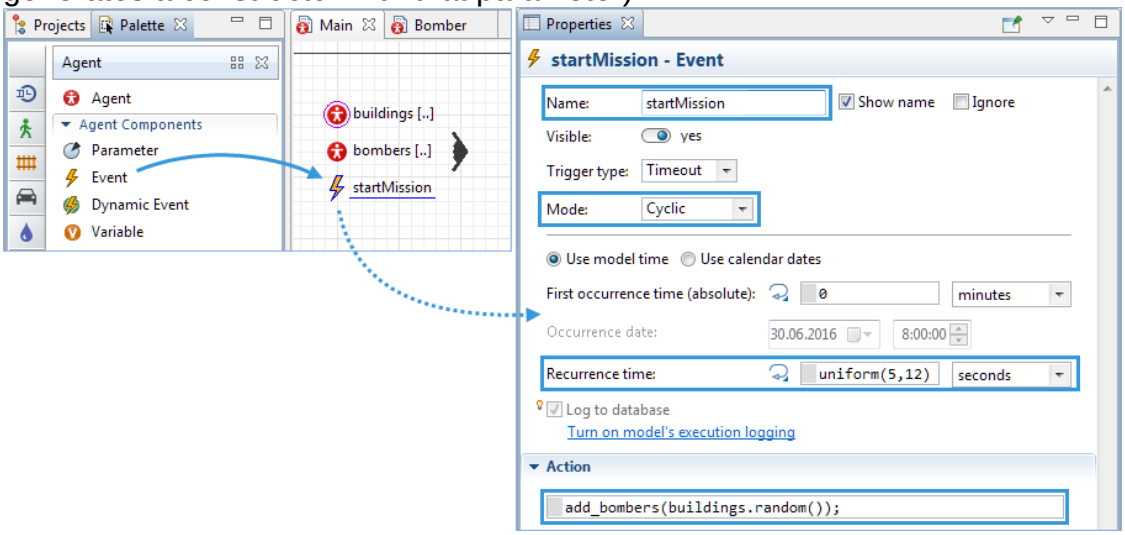
Program mission assignment

1. Switch to the Main diagram of the top level agent.
2. Drag the **Event** element from the **Agent** palette to the Main diagram and place it beside the agents populations.

3. Navigate to its **Properties** and perform the following changes:
   - Name it `startMission`
   - Set the **Mode** parameter to **Cyclic**.
   - Set the **Recurrence time** parameter's time units to **seconds** and type `uniform(5,12)` into the parameter's field.
   - Type `add_bombers(buildings.random());`

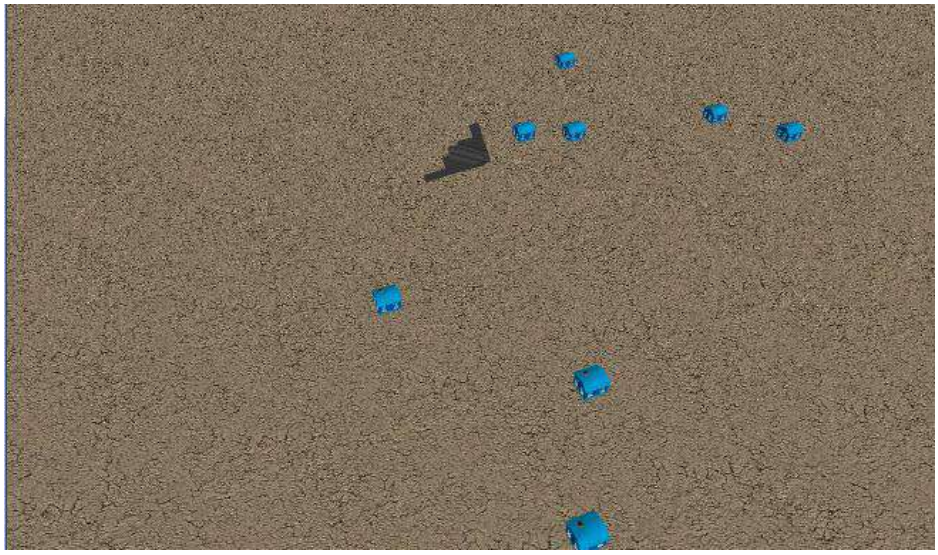
     into the field of the **Action** section.

4. The cyclic event periodically looks for a building without a bomber already flying to it. We use iteration across the agent population twice: in the outer loop we iterate across buildings, and in the inner loop we iterate across bombers. If such a building is found, we create a new bomber agent and assign the building to the bomber as the target (as long as the `Bomber` agent has one parameter **target** of type `Building`, AnyLogic generates a constructor with that parameter).

5. 

Finally, we can run the model to observe the flight of the bombers live.

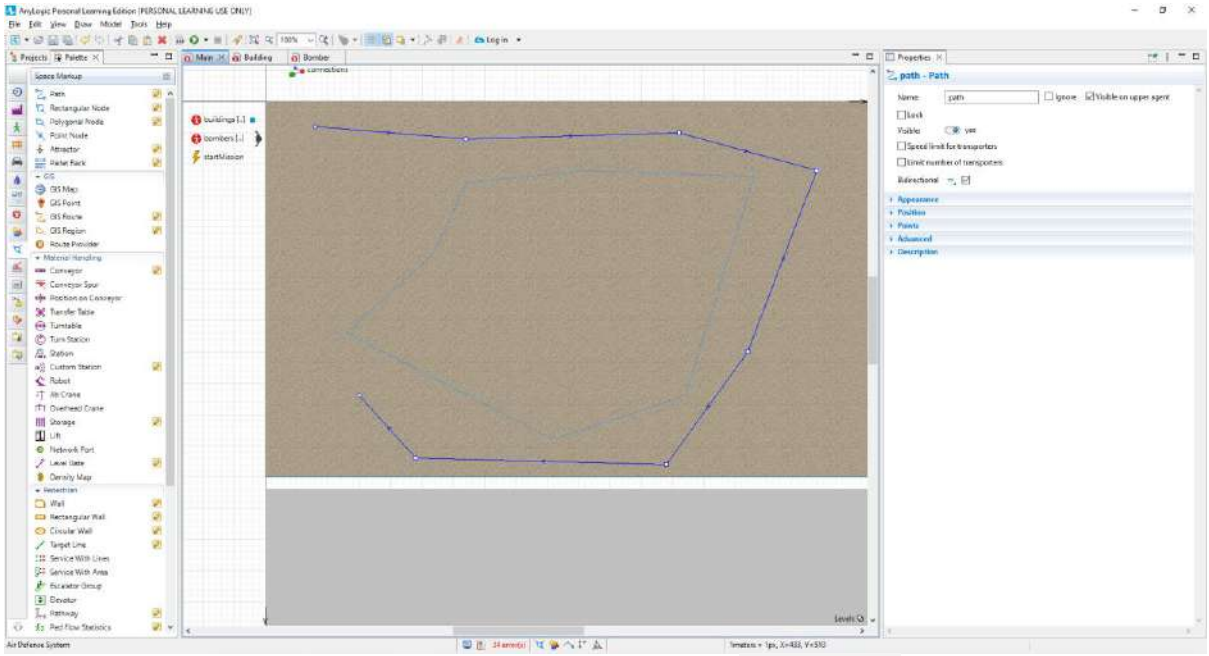
Upon creation, a bomber takes direction to the target building, makes an "instant u-turn", and heads back to (0,0). So far, the bombers use straight line trajectories — this is assumed by the `moveTo( x, y, z )` function. We will further draw the 3D "escape trajectory" for the return route and set the bombers to follow that trajectory on their way back. We will use another version of the method: `moveTo( main.exitNode );`



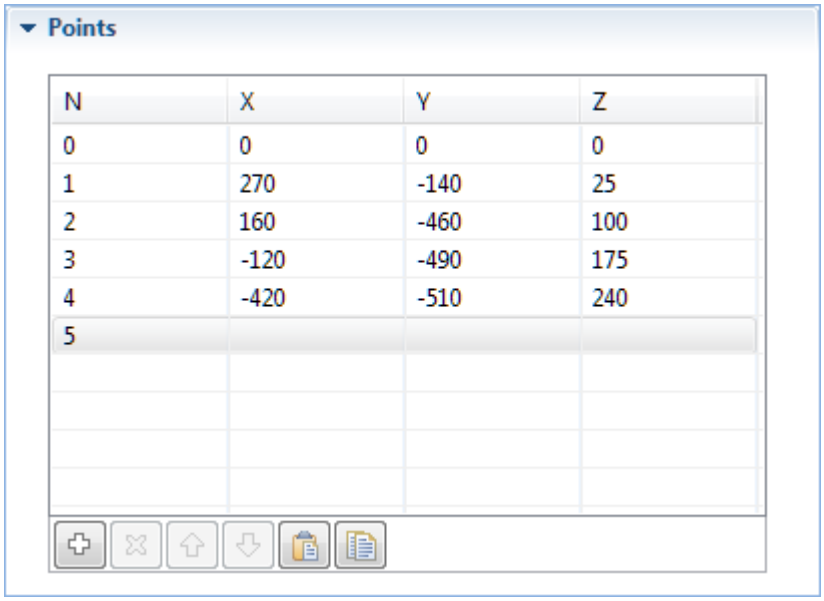
Now we will create escape route that the aircrafts will be following to avoid being hit by a missile.

Draw the 3D escape route

1. Open the **Space Markup** palette, double-click the **Path** element to activate the editing mode and draw a path, as shown in the figure.



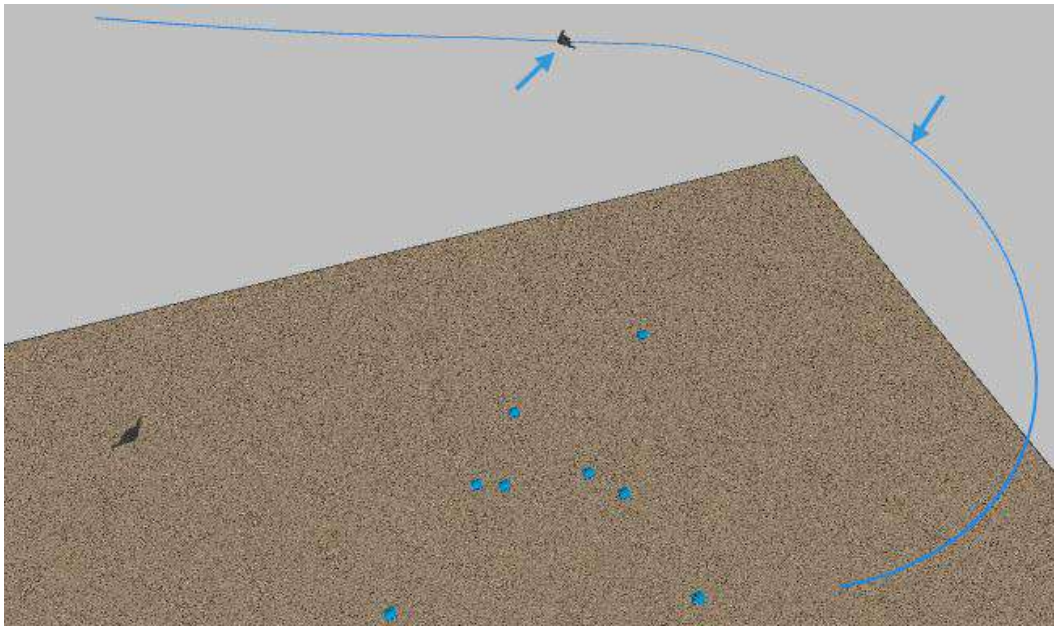
1. Navigate to its properties and name the path `escapeRoute`
2. Set the **Z** parameter of the **Position** section to `100` (this will be the base Z-coordinate of the polyline).
3. Open the **Points** section of the path properties and modify the individual Z coordinates of the points approximately, as shown in the figure. The idea is to have the initial section of the path at about the same altitude as the bomber attack altitude.



1. Add **Point Node** from the **Space Markup** palette to the end of the path, make sure the node connects to the path. Name it exitNode. Set its **Z** parameter of the **Position and size** section to 340.
2. Open the diagram of the Bomber agent, click the **Away** state to navigate to its properties and change the **Entry action** to: moveTo(main.exitNode);
   We need to put the prefix main before the exitNode because this graphical object is located not inside the Bomber agent, but one level up, in Main.

Run the model. See how the bombers return to the base using the defined by the path route.

The **escapeRoute** is visible at runtime. We will now set the **Visible** parameter of both the **escapeRoute** and the **exitNode** to **no**, to hide both of them.



Hide escapeRoute during runtime

1. Click the **escapeRoute** to select it.
2. Navigate to its **Properties** view and click the **Visible** parameter's toggle button to switch the visualization state to **no**. The **escapeRoute** will not be visible during runtime now.

In the same way hide the **exitNode**.

Run the model. No odd elements are present in the scene now.