Appendix A: AnyLogic Quick Guide

In this section we will explain how to use the AnyLogic program from start to a complete model simulation using an example along the way.

We will cover the following: Creating a project from scratch and explaining how to set up a workspace, working with and creating *Agents* and *Agent Types*, how to use specific building blocks and how to eventually run a simulation.

Each section will be colored based on the following legend:

Blue – Technical section.

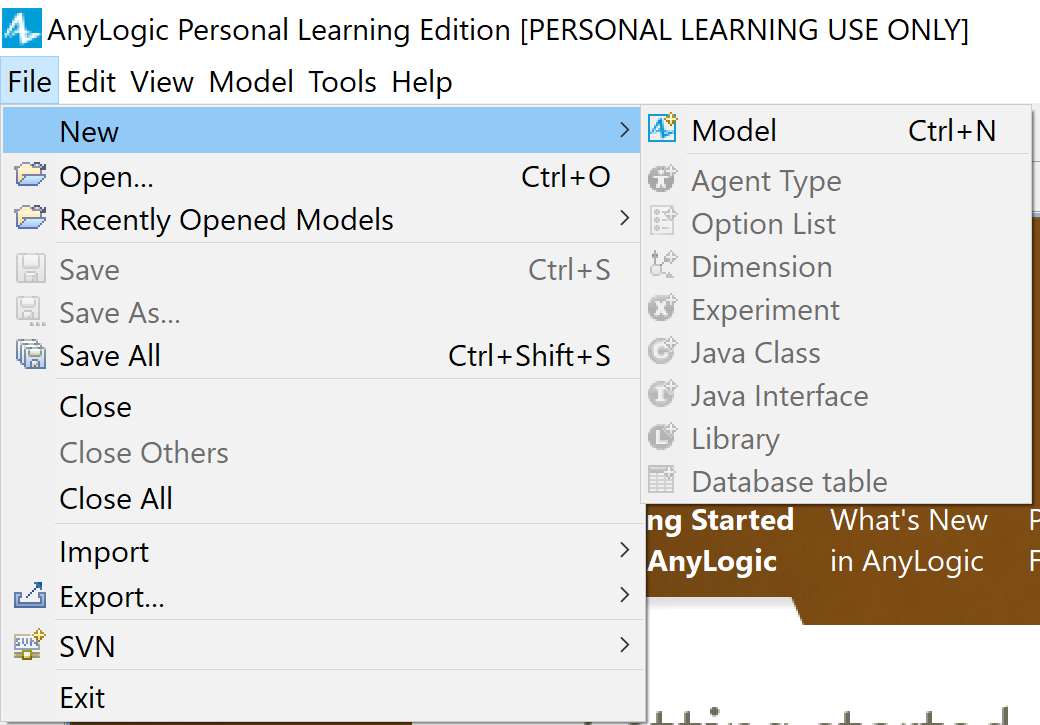
Red – Simulation relevance.

Green – Enhancing visibility.

At any time, you can refer to the references provided for the sections as follows:

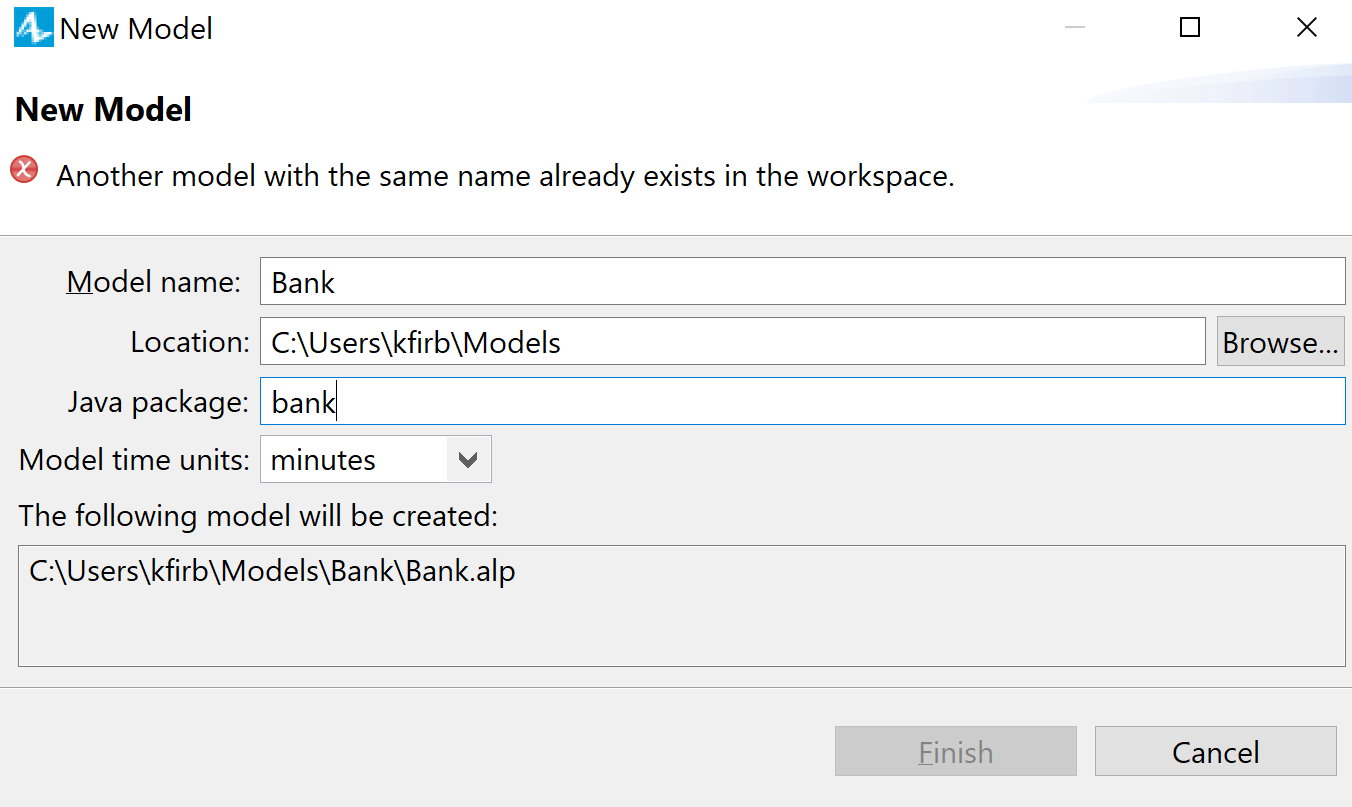
1. Sections 1-4 (inclusive): BankRef1.
2. Section 5: BankRef2.
3. Section 6-7: BankRef3 (final reference model).
4. **Creating a project**

In the program, press CTRL+N (Or in the menu press the *File > New > Model)*.



Give your model a name and the location in which you wish to save it, and choose the *Model time units* most fitting to your model, then press *Finish*.

In our example we will name the model “*Bank*” and our *Model time units* will be *minutes* as shown in the picture below.

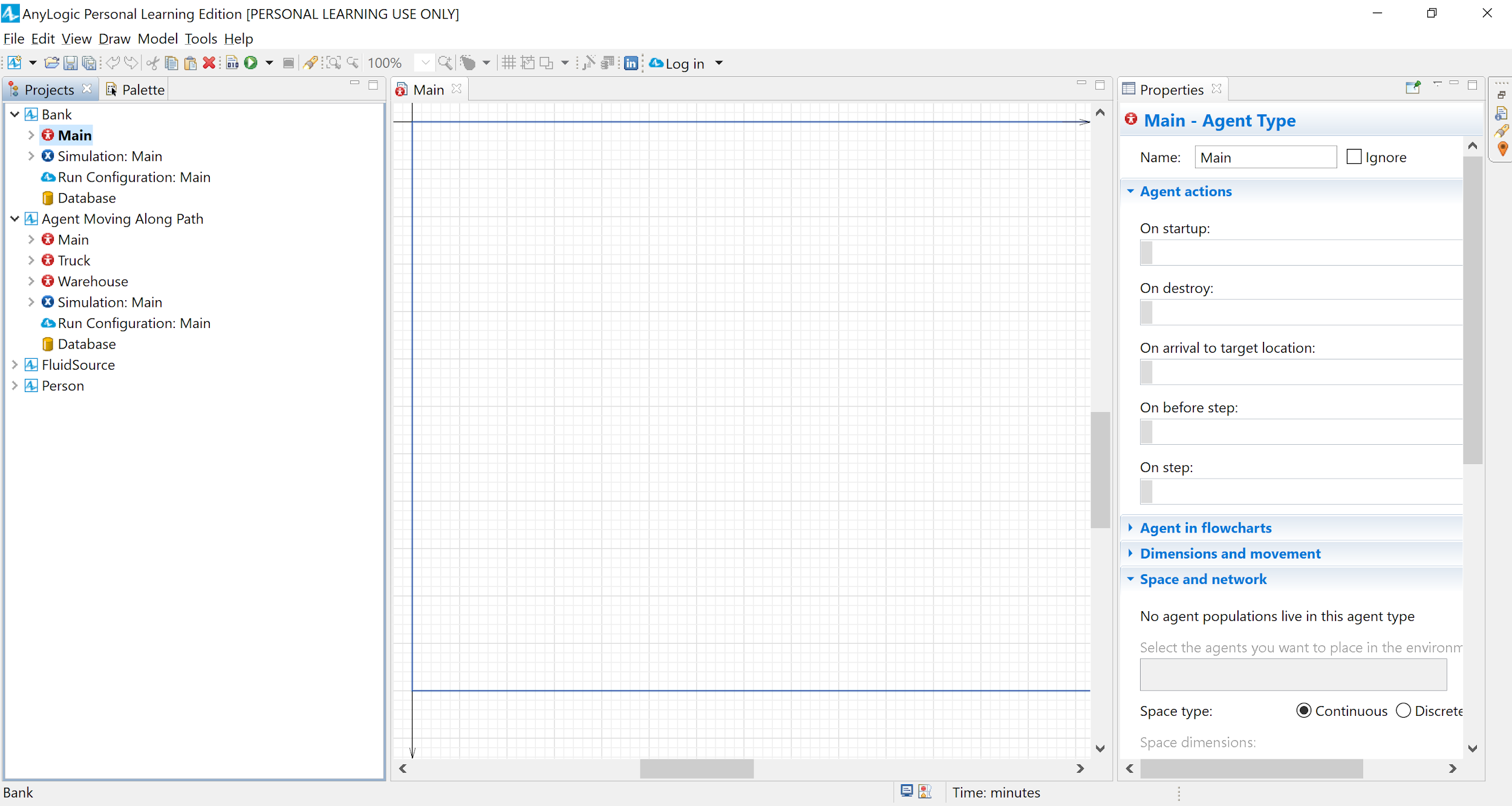


1. **Getting to know your workspace**

The screen that you see is divided to 3 sections: The leftmost section is where you can browse between *Projects* and *Palette*.

In the *Projects* tab you will be able to switch between the projects you’ve created thus far, while in the *Palette* tab you have the different working libraries with which you will create the model.

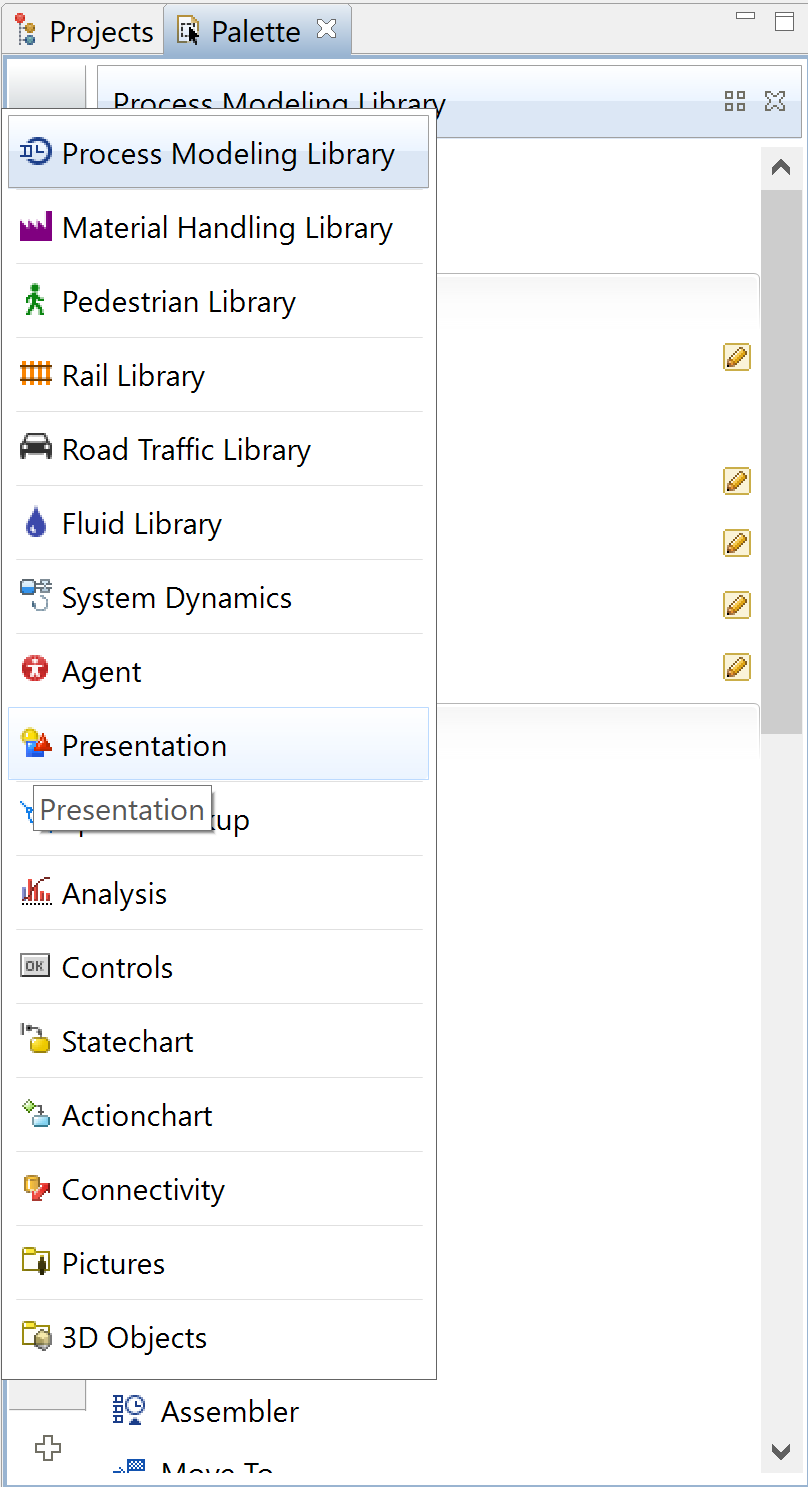
In the middle section you have the *Graphical Editor*, your workspace, and in the rightmost section you have the *Properties* of the currently selected *Object*, in our case it is the *Main* *Agent* (we will elaborate on what does this mean in a later section in this guide).

1. **Setting up a visual environment**

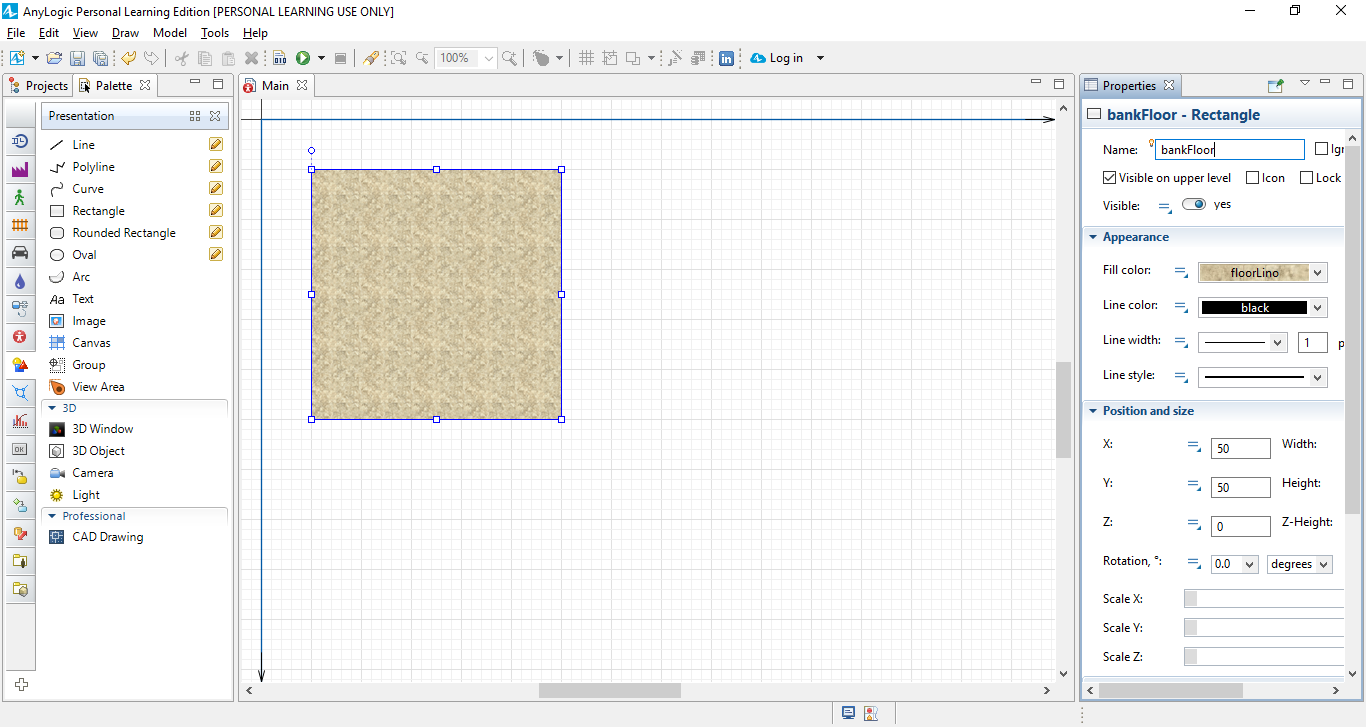
In AnyLogic you can set up a visual environment to work on, which are not interactive components of the simulation, such as images or shapes.

When you hover over a specific palette a pop-up window will appear which will display the names of all the palettes.

To set the environment, choose the *Presentation* palette () by clicking it as shown below.

In the menu that has opened you can choose now between the objects to set up your visual environment. To create any object there are two built-in options depending on the item of choice. If there’s a pencil mark double click the item then drag the component unto the *Graphical Editor*otherwise drag the item to the *Graphical Editor.* 

For our example, let’s set up the bank’s area by using the *Rectangle* element and dragging it in the *Graphical Editor*. After that, choose the rectangle you’ve created, and in the *Properties* tab on the right choose in the *Fill color* the color of your bank’s floor and in the *Name* type “*bankFloor*”.



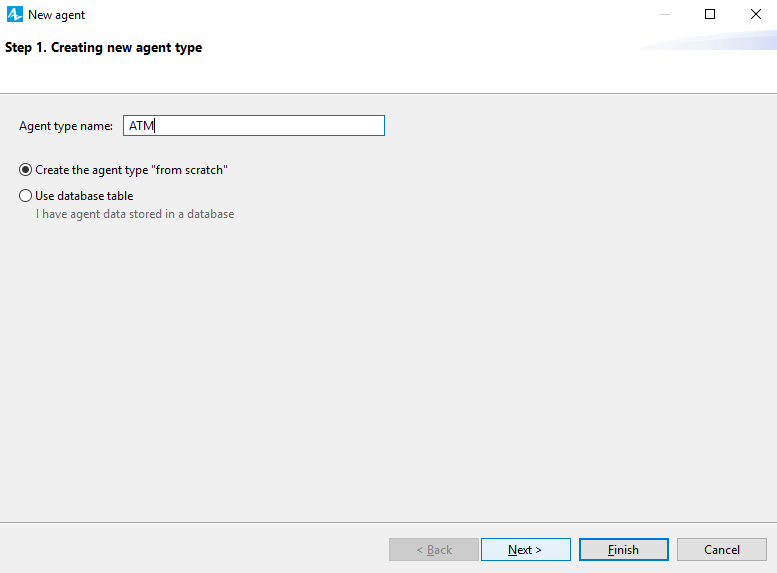
1. **Creating Agent Type**

Since AnyLogic is JAVA based, everything in it is an *Object*, and in AnyLogic an *Object* is called an *Agent*, which is an instantiation of an *Agent Type*, same as creating an instance of an object of a specific *Class* in *JAVA*.

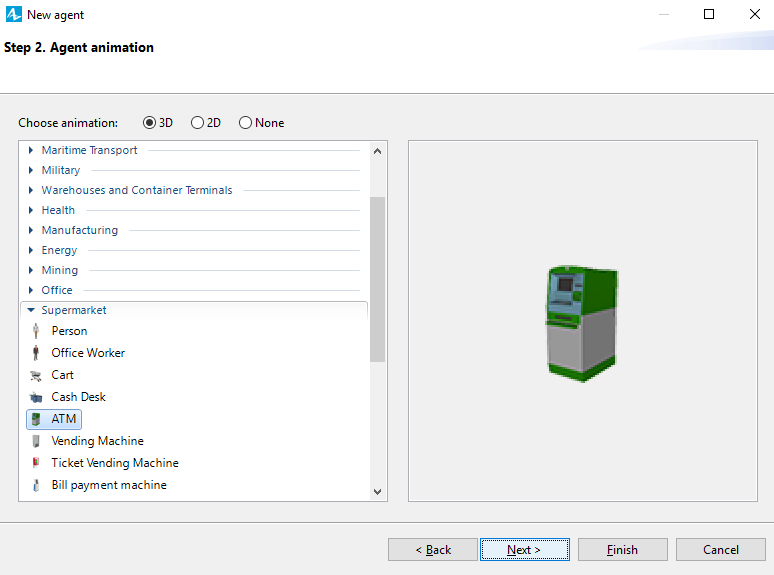
We will create 4 *Agent Type*s in our model: an *ATM*, a *Customer*, a *Teller* and a *Truck* (that will transport the money to the ATMs).

In order to create a new *Agent* you need first to create a new *Agent Type* by clicking the *Process Modeling Library* palette () and dragging the *Agent Type (*) component unto the *Graphical Editor*.

Once in the *New Agent* wizard you need to pick the Agent type name and then click *Next*.

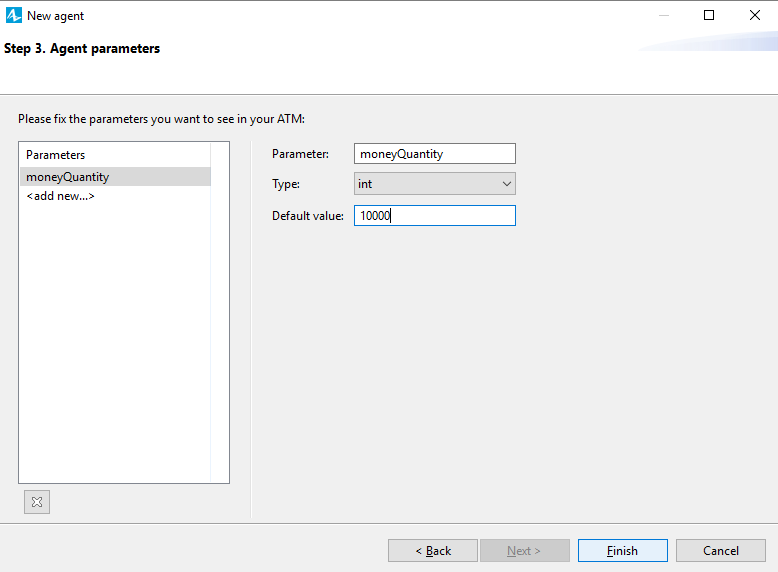


In the second step you can choose a 3D or 2D animation (or none) to represent your new *Agent Type*.

Under the *Supermarket* tree choose *ATM* and then click *Next*.

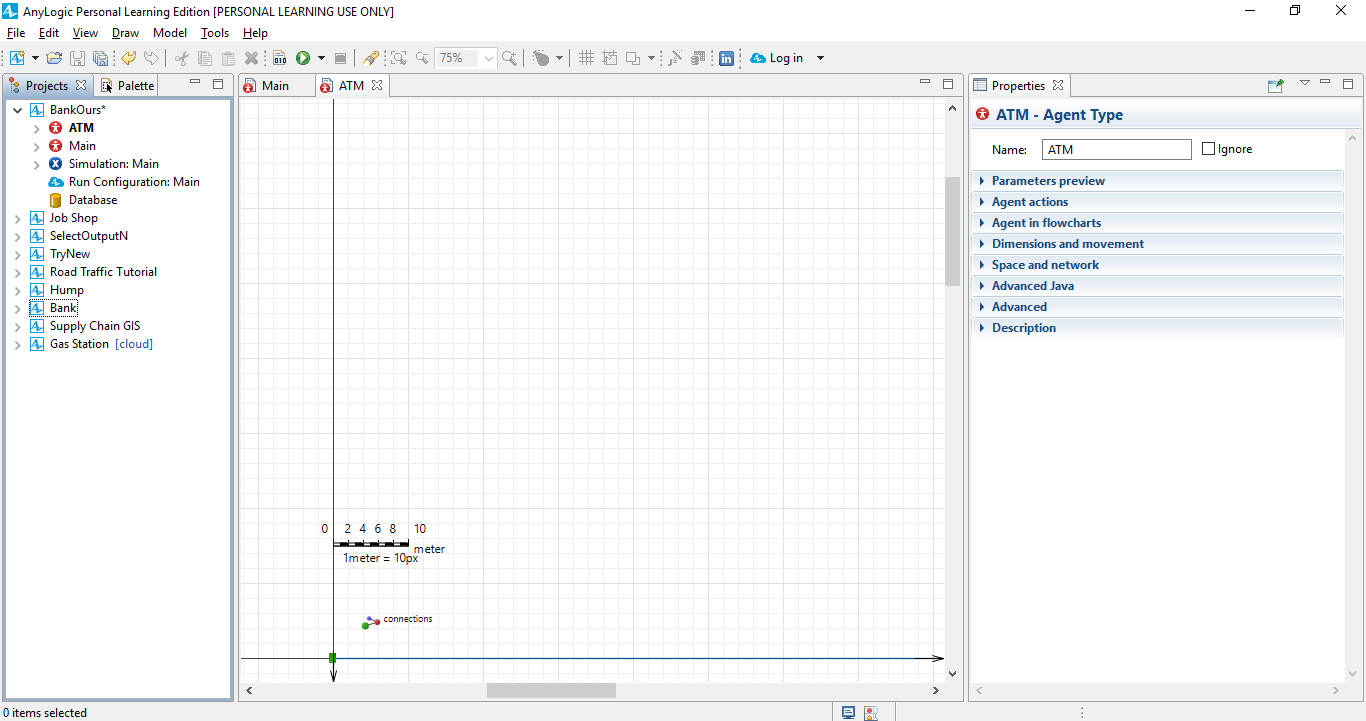
In the third step you can choose to add new parameters to your *Agent Type* by clicking the *<add new..>*.

Once clicked, 3 fields appear on the right where you can choose the new *parameter*’s name, type and defaulted value.

We will add the amount of money in the *ATM* as an integer value with a defaulted value of 10000. Once done, click *Finish*.

A new window will open in the *Graphical Editor* which represents the newly created *Agent Type*. In it you can see the *Parameter* you’ve created and the 3D animation where the axis meet.

If you click on the *Projects* tab on the left you will see the new *Agent Type* under the current model’s tree.



Similarly, create 3 more *Agent Type*s for the *Teller*, *Customer* and *Truck*:

* *Teller* – Choose a 3D animation of an *Office Worker* under the *People* tree, no parameters needed.
* *Customer* – Choose a 3D animation of *Person* under the *People* tree, no parameters needed (we will add parameters later).
* *Truck* – Choose a 3D animation of *Truck* under the *Road Transport* tree, no parameters needed.

Notice that now 3 tabs have been added in the *Graphical Editor* displaying the corresponding *Agent Type* names.

1. **Designing a simulation**
   1. **Building a flowchart**

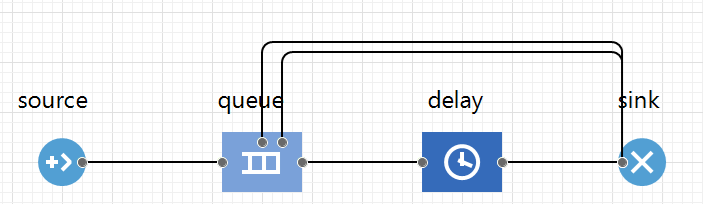
To create a flowchart process in AnyLogic you need to open the *Process Modeling Library* and use the *Blocks* in a flowchart manner.

To do so, simply drag the wanted block unto the *Graphical Editor*.

When dragging another block near an existing block in the *Graphical Editor* they will connect them automatically.

Each block will have its own properties appearing in the *Properties* tab on the right side of the screen.

We will create a flowchart that depicts the process of a customer entering the bank and then going to the ATM.

To do so, drag the *Source, Queue, Delay* and *Sink* from the *Process Modeling Library* unto the *Graphical Editor* as shown below:

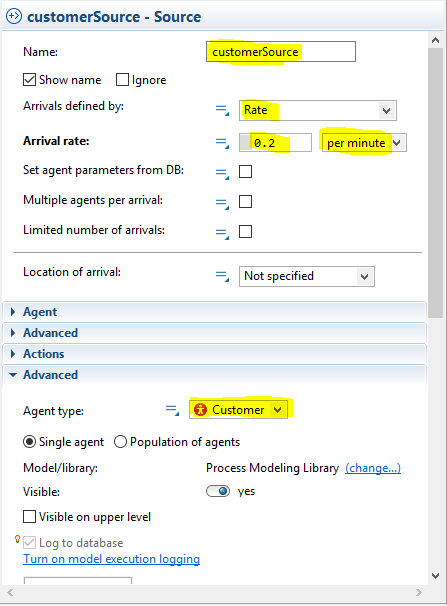
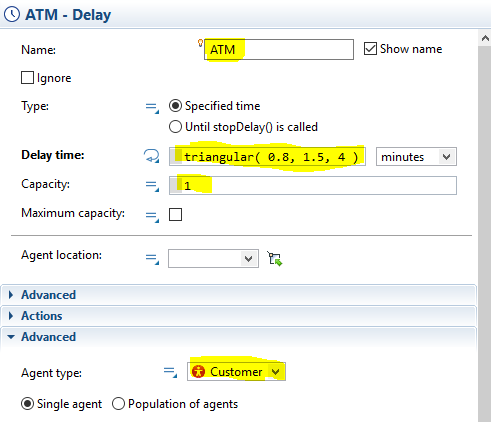
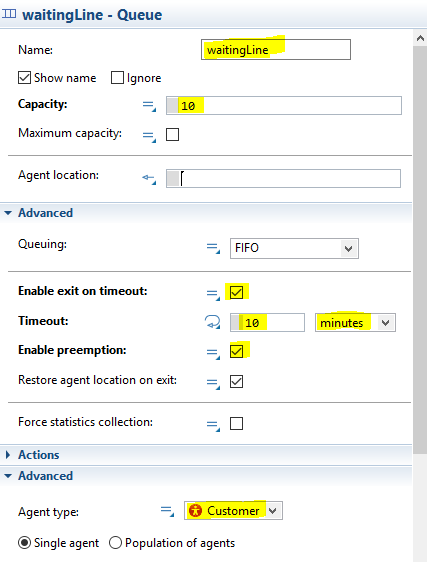
**A brief explanation on the presented blocks:**

***Source* -** Generates agents, usually a starting point for every process.

***Sink* -** Disposes agents, usually at the ending point for every process.

***Delay* -** Delays agents for a given amount of time.

***Queue* -** A queue of agents waiting to be accepted by the next object in the process. The two upper outputs are for preemption and timeout ports.

Now we will modify the properties of each block by clicking on it and adjusting the values as shown below.

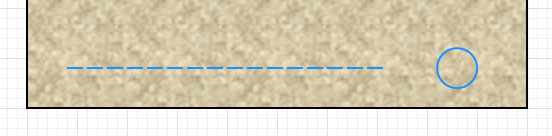
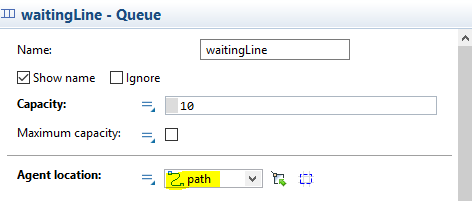
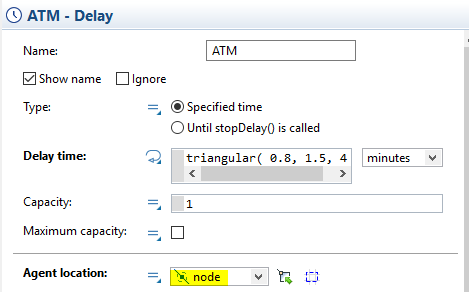
Notice: In the *Delay* block we named it *ATM*, it will represent the time the *Customer* stands at the *ATM* to withdraw money. The time is a number distributed triangularly between the minimum value of 0.8 minutes and maximum value of 4 minutes, with a mean value of 1.5 minutes.

* 1. **Adding active simulation elements - Space Markups**

*Space Markup* elements are active components, contrary to the elements in the *Presentation* palette, which agents can use or reside within them. They contain different types of *nodes*, which is a physical space where agents can reside in, and *paths*, which are routes that agents can use to move between two points.

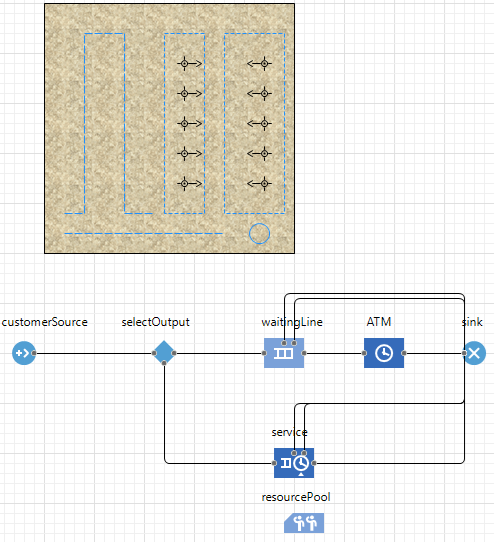
In the *Space Markup* palette () you will see all the possible options for such elements.

Connecting two elements can be made easily. Simply drag one element to an existing one and the connection point will turn green indicating that they are connected.

For our example, draw the *Path* and *Point Node* as shown below and change in the *waitingLine* *Queue* block and the *ATM Delay* block the *Agent Location* in the properties to *Path* and *Point Node* respectively. This will set the *Path* as the waiting line area and the *ATM* at the *Point Node* area.

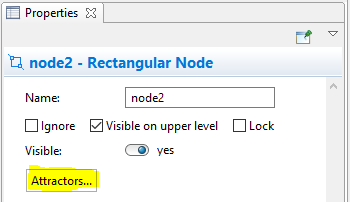
A unique *Space Markup* object is the *Palette Rack* ()*.* This component can be considered as an array that stores different agents. This specific markup is also a path, which is why when connected to a *Path* the areas defined as paths in the element also lit up in green.

* 1. **Completing our flowchart**

Our complete flowchart will look as follows.

Add the elements from the *Space Markup* and *Process Modeling Library* to complete the design.

Notice that we’ve added the following elements: *Rectangular Nodes, Path, Service, Select Output, Resource Pool* and *Attractors*.

To add the *Attractors* quickly and accurately you can click on the *Rectangular Node* and press the  button and choose 5 as the *Number of attractors*. To change the *Attractor*’s orientation, simply click on it and adjust it in the *Position and Size* section in the *Properties* tab.

**A brief explanation on the presented blocks:**

***SelectOutput* -** Monitors the incoming agents to one of 2 outputs. You can also create your own *SelectOutput* block by using the *Select Output In* and *Select Output Out* blocks and defining in them how they should correspond with each other.

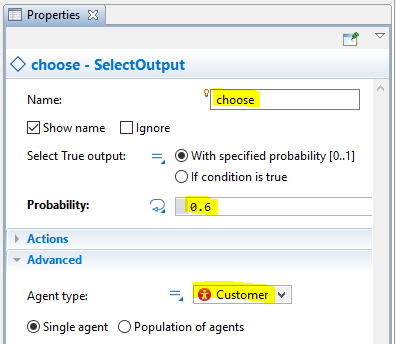
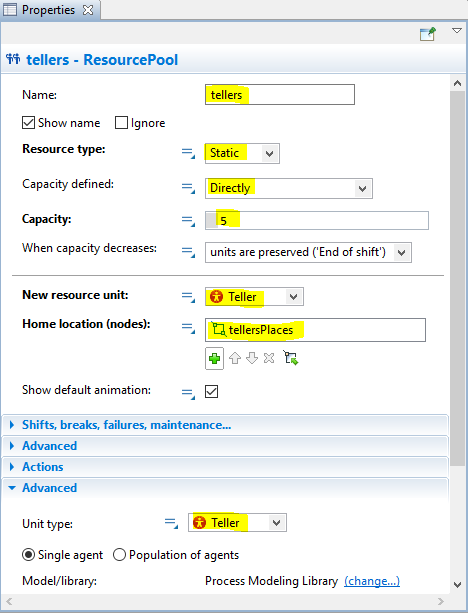
***ResourcePool* -** Defines a set of resources. These resources can then be seized and released by agents using the corresponding blocks.

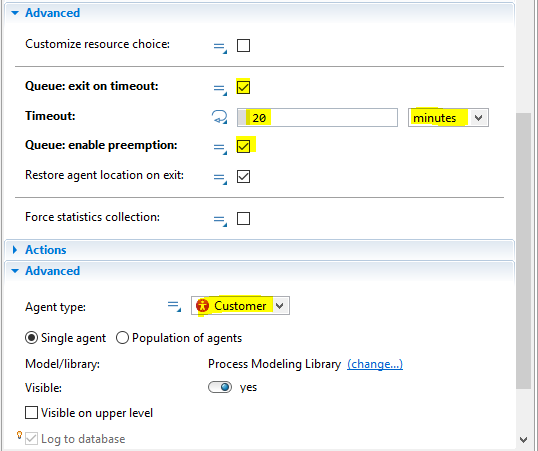
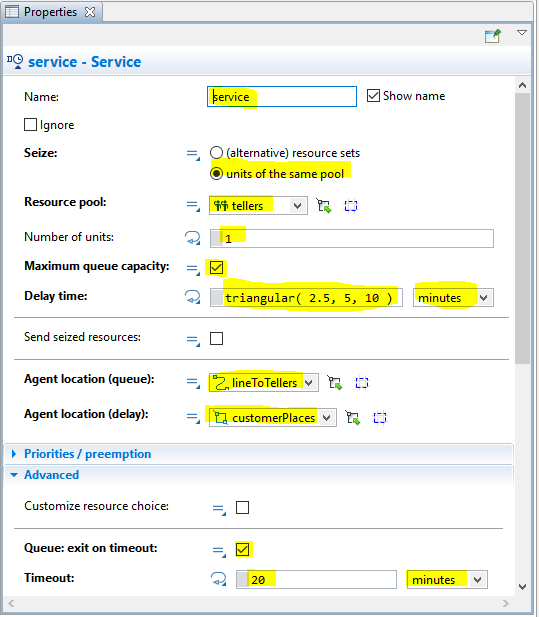
***Service* -** A combination of *Seize*, *Delay* and *Release* objects which does as stated below.

***Seize* -** Seizes a given number of resources from a given *ResourcePool* or several *ResourcePool*s. The *Seize* block can choose from several options of resource sets. Contains a queue where the agents wait for the resources which are currently not being seized. There are options for preemption and timeout outputs as well.

***Release* -** Releases a given number of resource units seized by *Seize* object.

Now let’s adjust the properties for each added element:

First, name the new *Path* “lineToTellers”, the *Rectangular Node* that follows the *Path* “customerPlaces” and the rightmost *Rectangular Node* “tellersPlaces”.



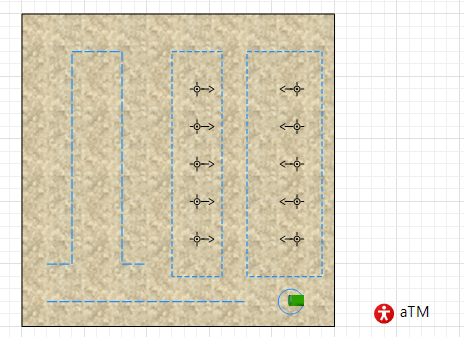
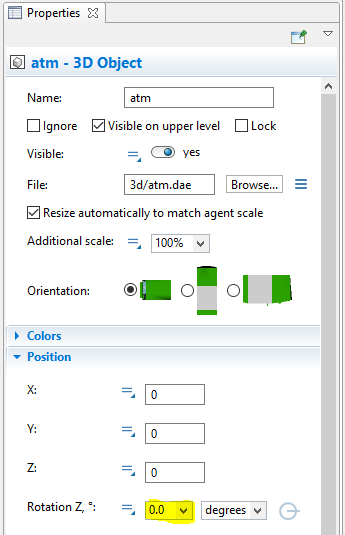
The *Probability* refers to the probability that the customer will proceed to the *ATM*.

* 1. **Adding an ATM Agent**

For now, we’ll add an *ATM* agent only for the visual aid.

Choose the *Agent* element from the *Agent* palette and drag it to the *Graphical Editor*.

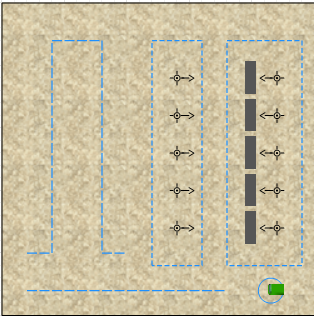
Place the animation in the *Point Node* where it should stand.

Double-click on the *Agent* to open a new tab in the *Graphical Editor*. Click the *3D Object* figure in the axis meeting, and change the *Rotation Z* parameter under the *Position* tree in the *Properties* to 0° so that the *ATM* will face the *Customer* using it.

* 1. **Adding 3D objects**

To add 3D objects for visual aid purposes choose the *3D Objects* palette from the *Palette* tab. Pick the *Table* under the *Office* tree and drag it to the *Graphical Editor* where the teller’s attractors are as seen below. Make sure that you set the table’s *Rotation Z* to -90°.

Notice: You can click CTRL while dragging a table to create a copy of it



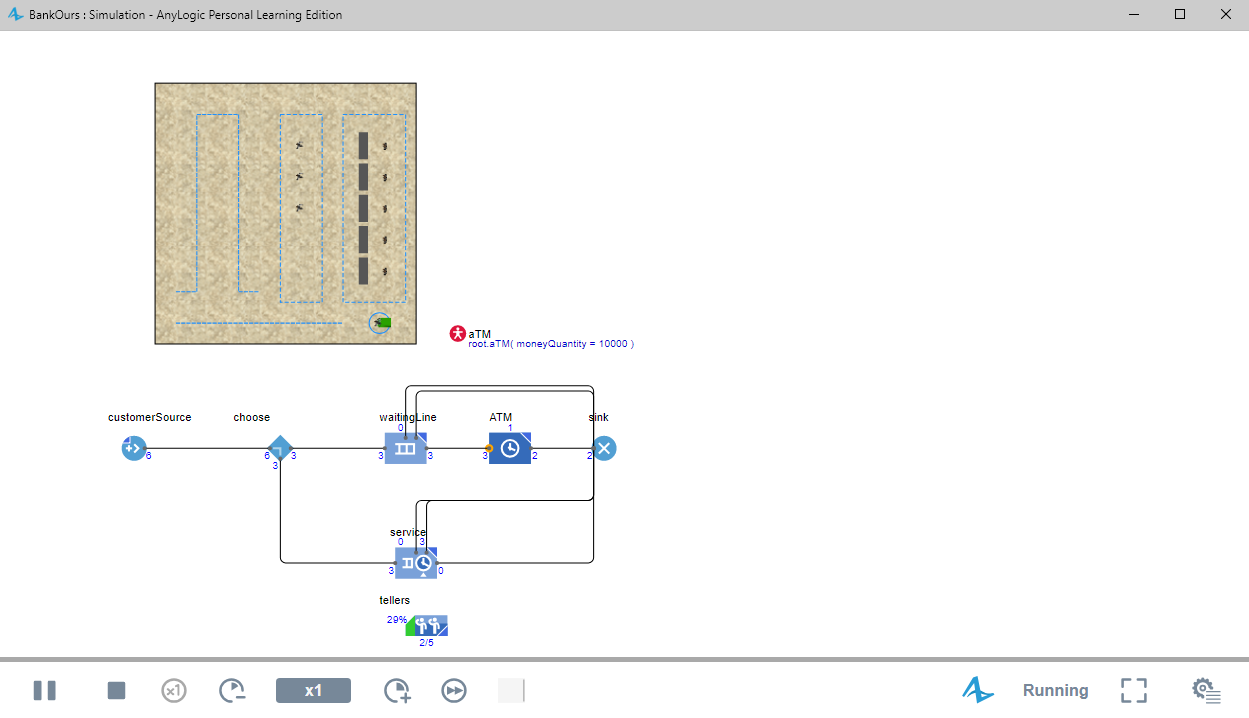
* 1. **Running a first simulation**

We’ve reached a point where we can now run a basic simulation.

Press the *Run* button () in the upper toolbar (or press F5 key).

Notice: Once you have several models, you can choose from the different projects which one to run, by clicking the arrow and choosing the one you want.

A pop-up window will appear. Press the *Play* () button to start the simulation.

You can see now the customers entering the bank, some will go to the ATM and some will turn to the tellers for service.

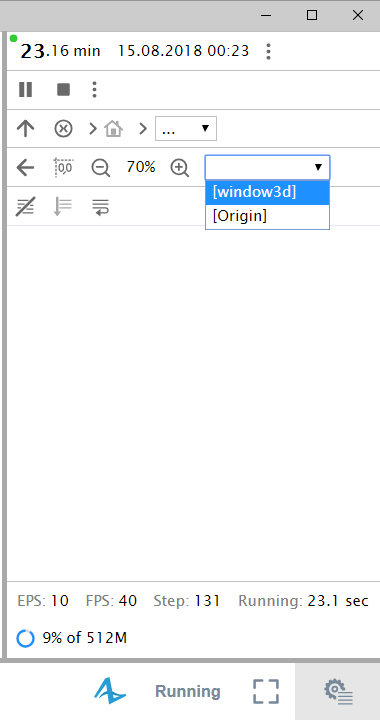
Note: You can modify the *customerSource* block to change the number of customers entering the bank and see how it affects the simulation or change the probability that a customer turns to the ATM and see the affect also.

* 1. **Creating a 3D animation**

In AnyLogic you can see both 2D and 3D animations of your simulation.

To add a 3D simulation all you need to do is drag the *3D Window* from the *Presentation* palette unto the *Graphical Editor* somewhere.

Now when you run the simulation you will see both animations.

If you wish to see only the 3D animation you can choose so in the sub window on the right as shown below.

To control the camera view in the 3D animation or the 2D animation you can use the mouse left button, the scroll button, and a combination of ALT key and the left mouse button to move around and zoom in.

1. **Integrate statecharts**

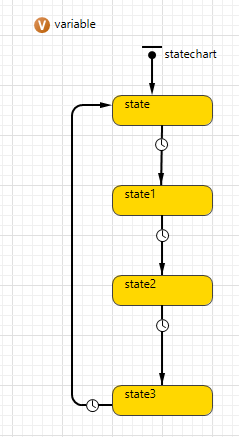
The *Statechart* palette contains elements to design logical stages of interest.

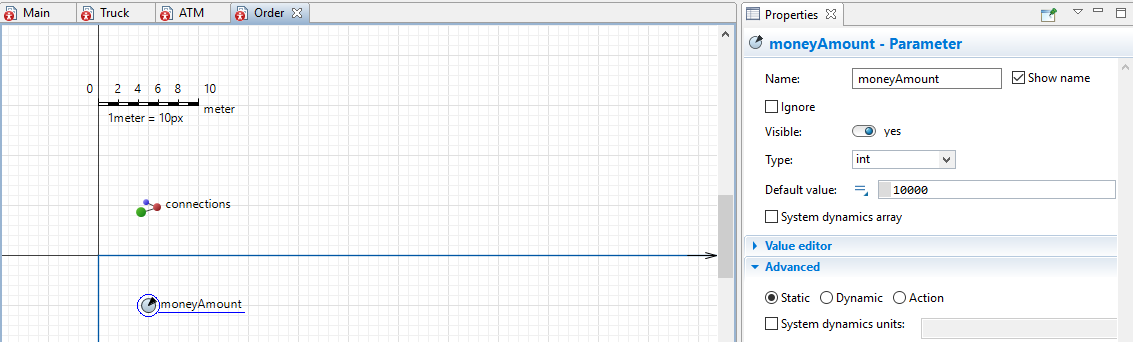
The main two blocks are *State* and *Transition.*

A *Statechart* diagram begins with the element *Entry Point*.

In the *Transition* elements the user can modify the action and the *Trigger* (what causes the action to start) of said *Transition* amongst other properties available.

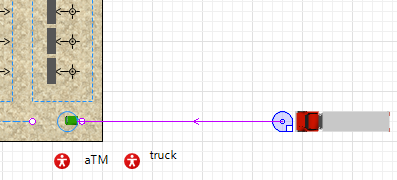
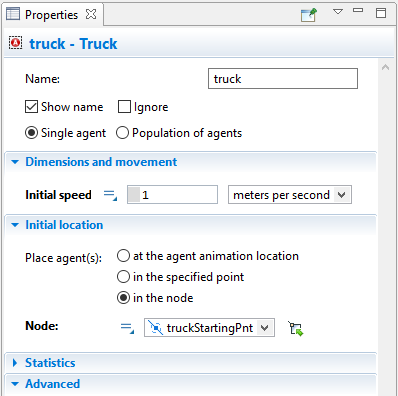
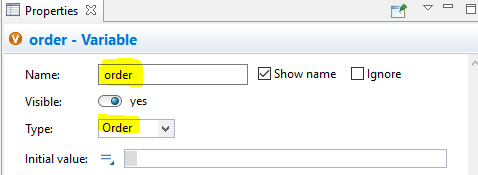
Let’s add a state chart diagram to the truck agent:

Open the *Truck* *Agent* tab and add elements to look as seen below.

* 1. **Adjustments prior to creating the state chart**

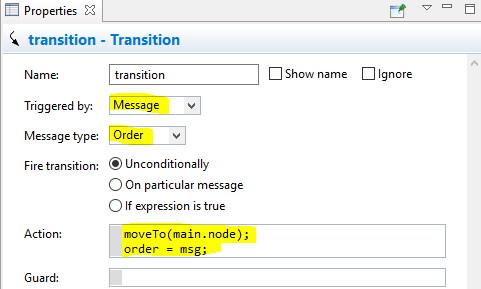
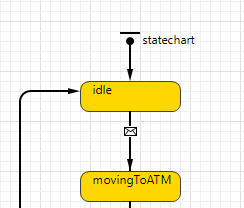
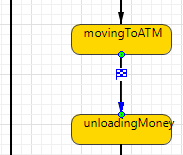
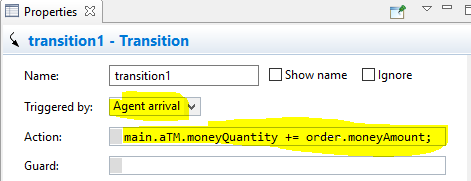
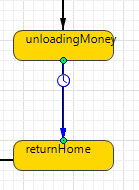
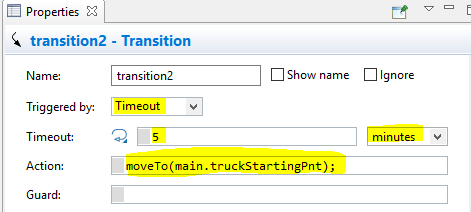
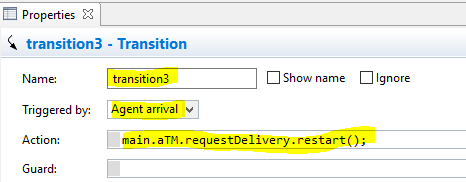
Before we get to change the *States* and *Transitions* we need to create a new *Agent Type* called “*Order*” in the *Main* tab. Give the new *Agent Type* no animation and one parameter called “*moneyAmount*” with a defaulted value of 10000 from *Type* *int*.

Next we’ll add a *Truck* *Agent* to the *Main* *Agent* and locate it in a node called “*truckStartingPnt*”, which will be connected to the *node* where the *ATM* is located by a *Path* as seen below.

Last step before adjusting the flow chart, modify the *Variable* type as seen below

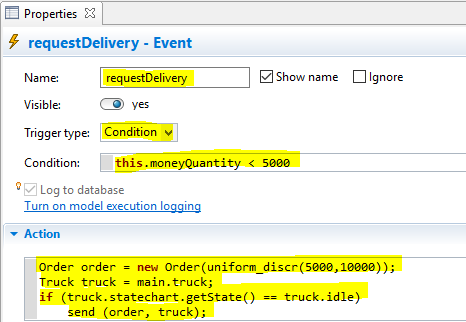
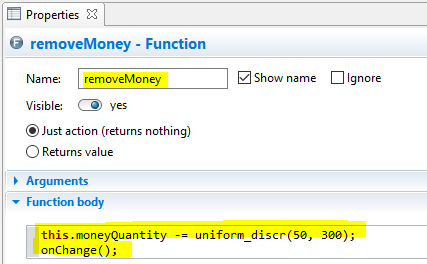
* 1. **Editing the state chart**

Now adjust the names of the *States* and logic of the *Transitions* step by step in the *Truck* *Agent* Type’s tab:

1.   whereas the logic of the *Transition* from *idle* to *movingToATM* is:
2.  whereas the logic of the *Transition* from *movingToATM* to *unloadingMoney* is:
3.  whereas the logic of the *Transition* from *unloadingMoney* to *returnHome* is:
4. The final *Transition* from *returnHome* to *idle* should look like this:

Let’s review what we’ve done:

1. We triggered the transition from the *idle* state to the *movingToATM* when we got a *message* which is *Order* in our case and then the *truck* moves to the *ATM*’s *Node*. The *message* was sent by the *ATM*, we’ll cover that in a bit. We saved the *message* as the *Order Agent*.
2. Once the *truck* arrives at its destination, the money in the *ATM* will rise by the amount specified in the *Order* *Agent*’s parameter *moneyAmount*.
3. After 5 minutes, simulating the unloading of money from the *truck* unto the *ATM*, the truck will head back to its starting node.
4. Once the truck arrives at its destination it will become idle again.
   1. **Initiating the state chart’s process**

Open the *ATM Agent* tab and drag from the *Agent* palette a *Function* element and an *Event* element, and adjust their properties as follows:

The *Function* that we’ve named “*removeMoney*” simulates the amount of money that the *Customer*s withdraw from the *ATM*. Notice that the amount is discrete and uniformly distributed between the values of 50 and 300, and are reduced from the *moneyQuantity* parameter of *ATM*.

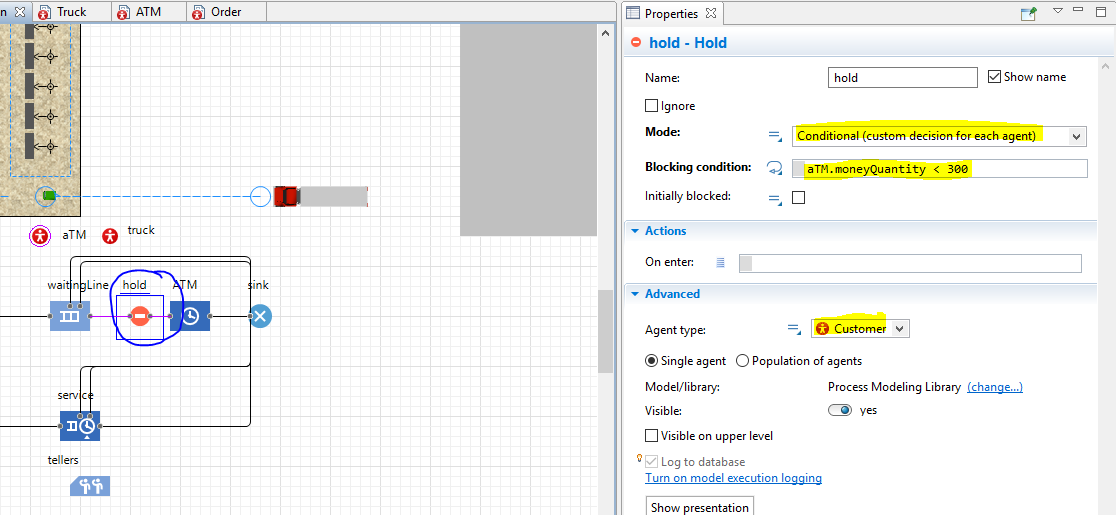
The *Event* is a bit more complicated, it schedules an action in a model. In this case the action is triggered by a rate of 0.7 per hour, and what the action does is creating a new *Order* with an amount of money distributed uniformly between the values of 5000 and 10000 and sending it to the *truck*, only if the *truck* is in an *idle* state.

Activating the entire process is done by the *Delay* block in the *Main* *Agent* since that’s the exact moment in the *Customer*’s process in which the money is withdrawn.

In the *Delay* block, edit the *On enter* edit box under the *Actions* tree to “*aTM.removeMoney();*” which will call the *removeMoney* function of *ATM*.

Now if you run the simulation you will see that a truck has appeared on the right side of the bank and will move to the bank when it is called to fill the ATM. Also, you can see the current amount of money in the ATM next to the ATM item itself in the simulation.

1. **Extras**
   1. ***Hold***

In the *Process Modeling Library* you will find the *Hold* element, which blocks an agents flow in the process, even if it can be accepted by the next object in the process. For example, if there isn’t enough money in the *ATM* then the *Customer* can’t withdraw any money and will stop at the *Hold* element in the process.

To see this happens, you should change the *Rate* of the arrival of the *truck* to at least 0.5. Since we’ve enabled *timeout* in the *queue* the *Customers* might leave if they’re waiting for too long.

* 1. **Elaboration on other important elements**
     1. ***Match* -** Synchronizes two streams of agents by matching pairs according to a given criteria. The streams wait in a queue until a match is found and only then leave the block in pairs. For each stream there are also 2 more output options involving preemption and timeout if specified.
     2. ***Split* -** Splitting an agent coming to the *in* port. The original agent will come out of the *out port* while the copies of that agent come out of the *outCopy* *port*. You can set how many copies will be made and whether they will be of the same *Agent* *Type* or an entirely different *Agent Type*.
     3. ***Combine* -** Combines 2 incoming agents into a single output. The output agent can be either a new *Agent Type*, or either of the incoming agents. Another option is to give a specific statement for how the new agent will be produced (for example with if/else statements regarding the incoming inputs).
     4. ***MoveTo* -** Moves an agent to a new location.
     5. ***ResourceAttach* -** Attaches a seized resource to an agent, and from now on they will move together.
     6. ***ResourceDetach* -** Detaches a previously seized and attached resource (or resources) from an agent so that they will no longer move together.
  2. **Combining several blocks into one black box**

You might want to combine several blocks into one to make your model more readable and cleaner. To do so, you need to select all the blocks you wish to turn into a black box, right-click with your mouse and choose the option “*Create Flowchart Block”*. A new icon will appear which is your new object. You can double-click on it to open the new *Agent* you’ve created and see all the blocks inside.

* 1. **Auto completion**

There’s an option for auto completion based on the available Agents, Methods etc. By pressing ***Ctrl+Space*** a pop-up window will appear displaying all the available options to complete, and by double-clicking an option it will automatically be added to the location chosen.

* 1. **The Fluid Library**

In AnyLogic you can create a model to simulate either discrete or continuous logic. Up until now we talked about the discrete options, and to use the continuous elements and properties available by the program, you need to pick the elements from the *Fluid Library* in the *Palette* tab. You will find there elements such as *Source*, *Dispose* (Like *Sink*), *Tank* (Like *Palette Rack*), *Pipe* (like *Path*) etc.

Special elements in this library include *Agent To Fluid* and *Fluid To Agent* which allows us to go from discrete values to continuous values and vice versa.

* 1. **Statistics**

To add some statistical information to the model, the user can choose add charts, plots and histograms from the *Analysis* palette which will tell the user some statistical analysis of the model.

The user can modify each element from *Analysis* which he’ll use in the *Properties* tab, things like the values to measure, the appearance of the charts etc.