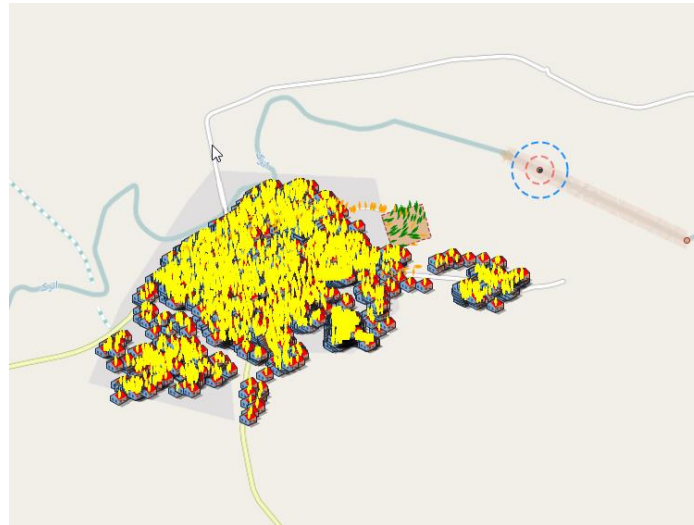


Disaster Simulation: An AnyLogic Agent-Based Approach

Example:
Flood simulation and evacuation
GIS Environment



Ali Asgary
ADERSIM
York University
2017

Lesson 4: Defining Agents' Behavior-Sensor

- Defining agents' behavior using statechart
- Working with transitions and their triggers.

Defining a simple statechart for sensor agent

We want sensor to react to flood agent and pass the flood information to decision makers or residents (in this version of the model to residents).

To do this we use statecharts to define different states of the Sensor agent.

We assume that Sensor has two states:

Normal: when water level is below certain level (no flood).

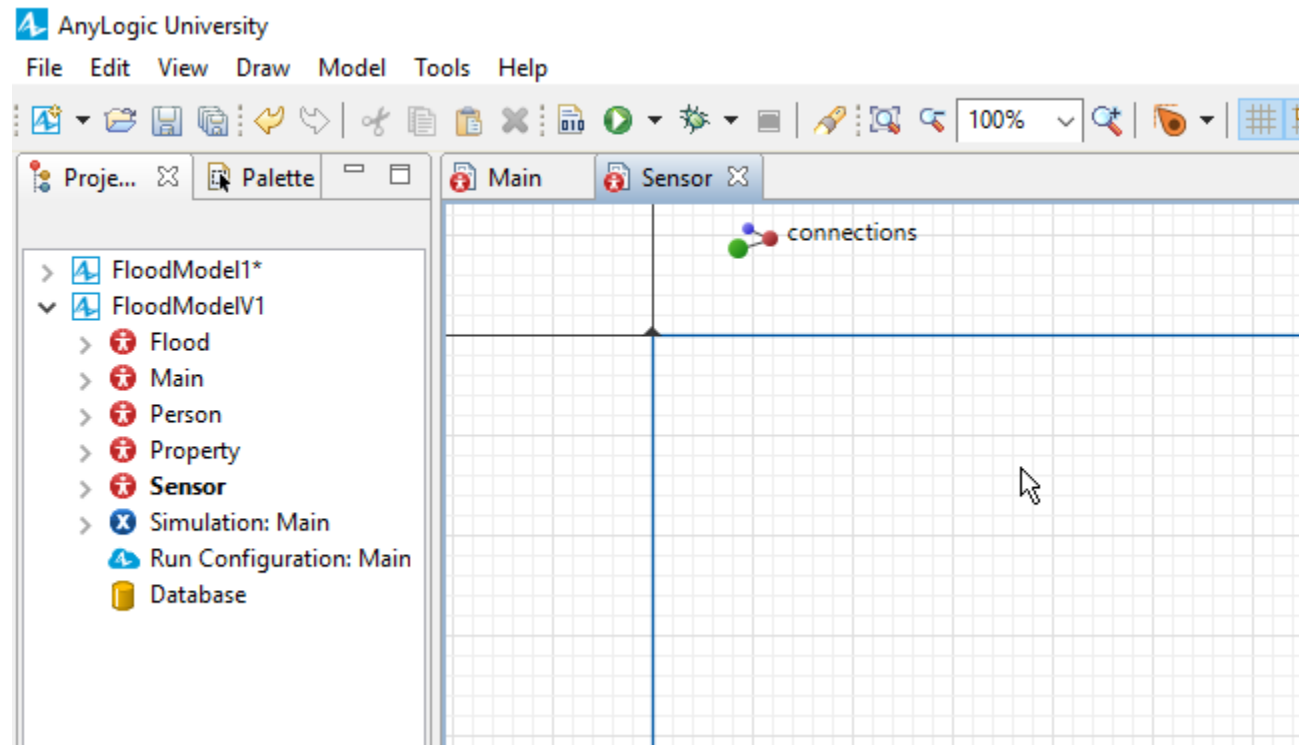
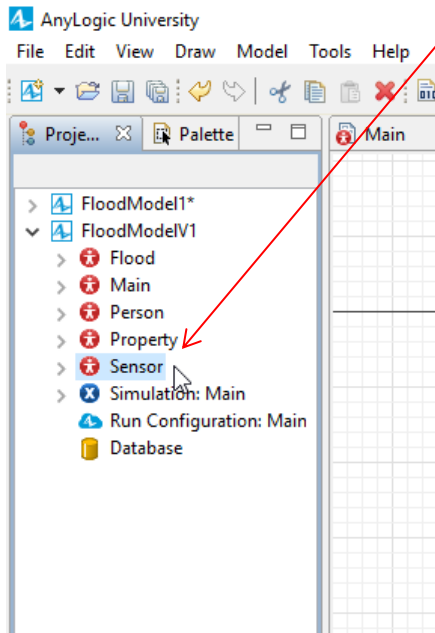
Flooding: when water level is above the normal level (flooding)

Sensor continuously checks the river (once every second) . If flood agent is sensed it changes its state from Normal to Flooding. It then sends a warning to the people in the town.



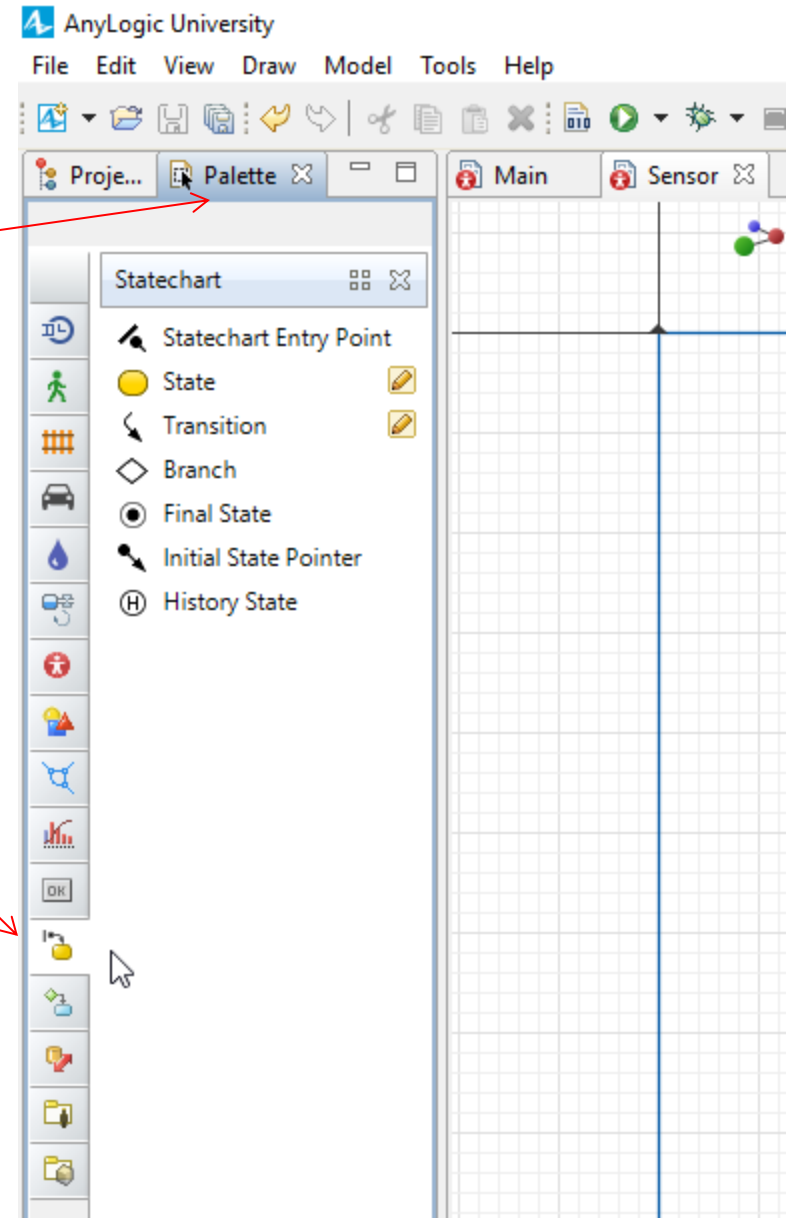
Open Sensors window

Double click on Sensor to open its window.



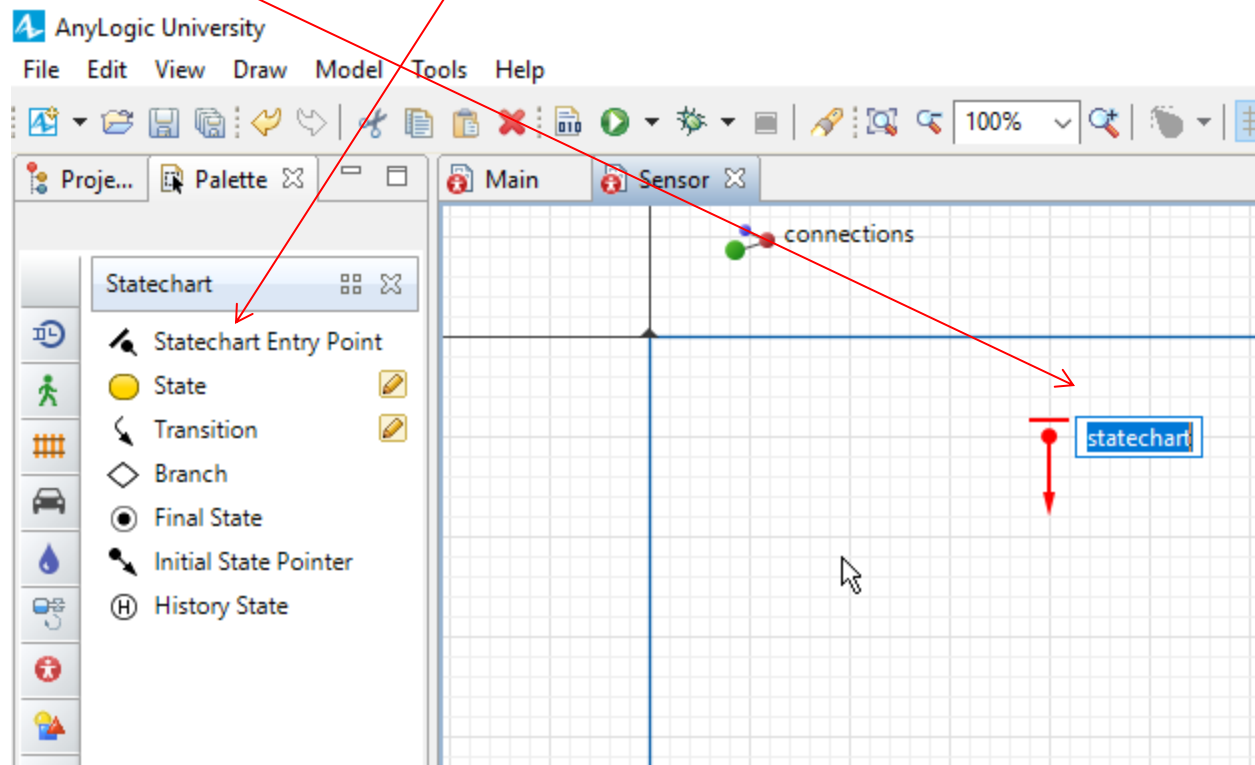
Expand Statechart tools

Click on the Palette tab and click on Statechart tools.



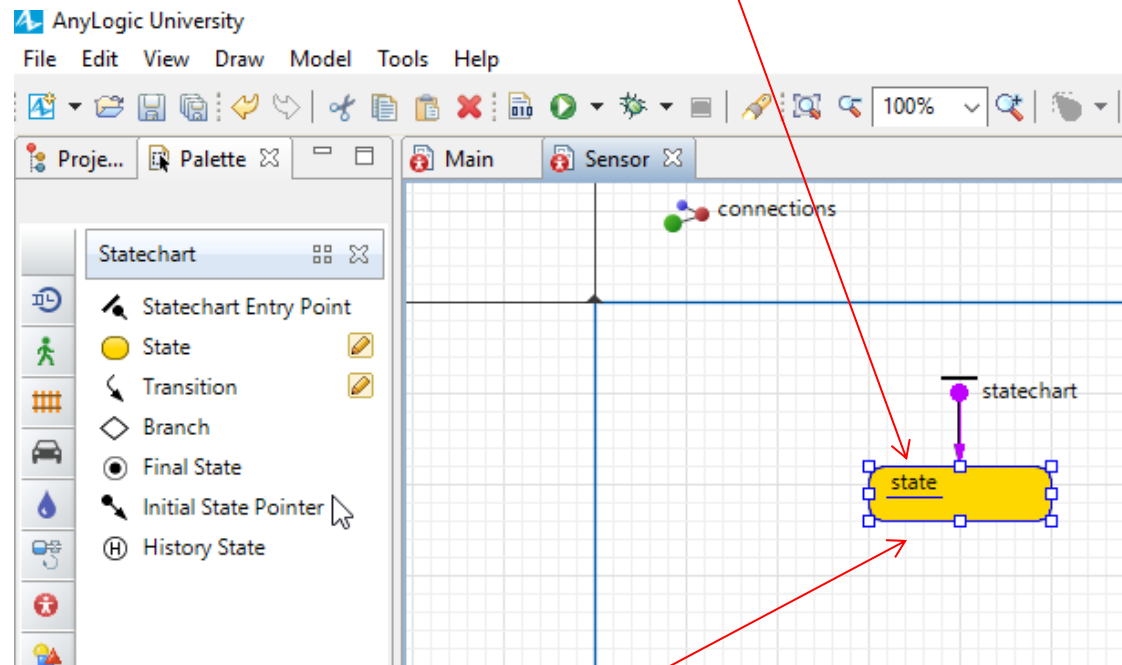
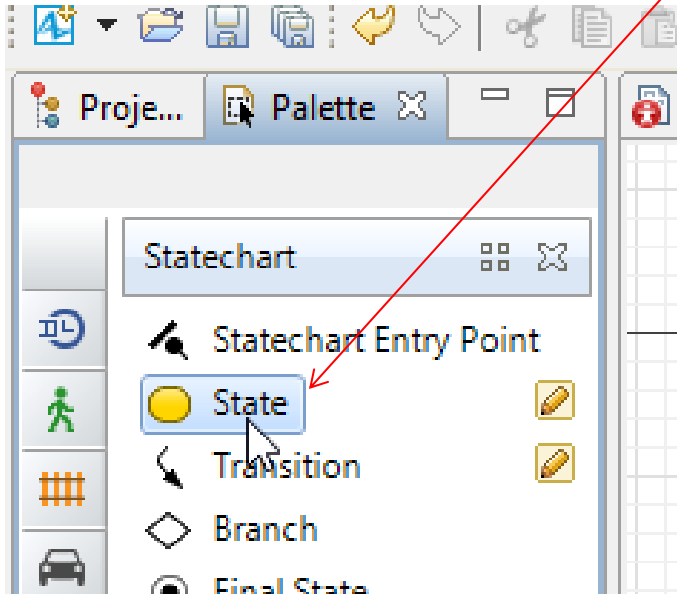
Creating a statechart

To create a statechart for your Sensor you need to drag and drop a Statechart Entry Point to your Sensor window.



Adding states

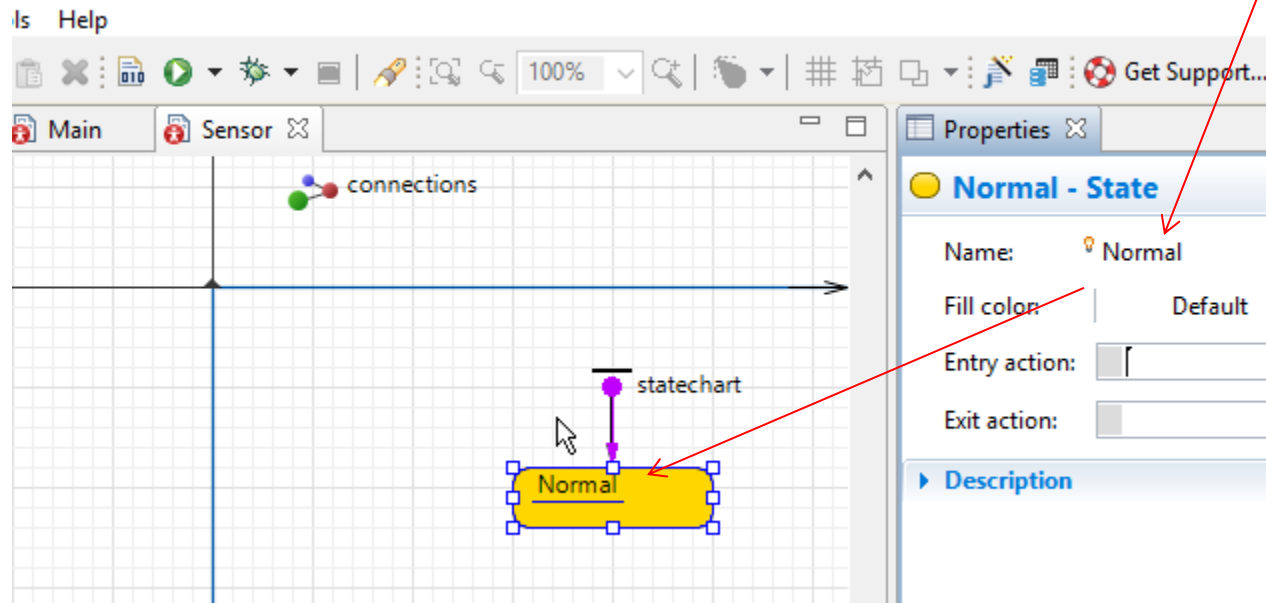
You can add a state by dragging and dropping a state from the Statechart tools to your Sensor space.



Make sure that the state is connected to the Statechart Entry Point

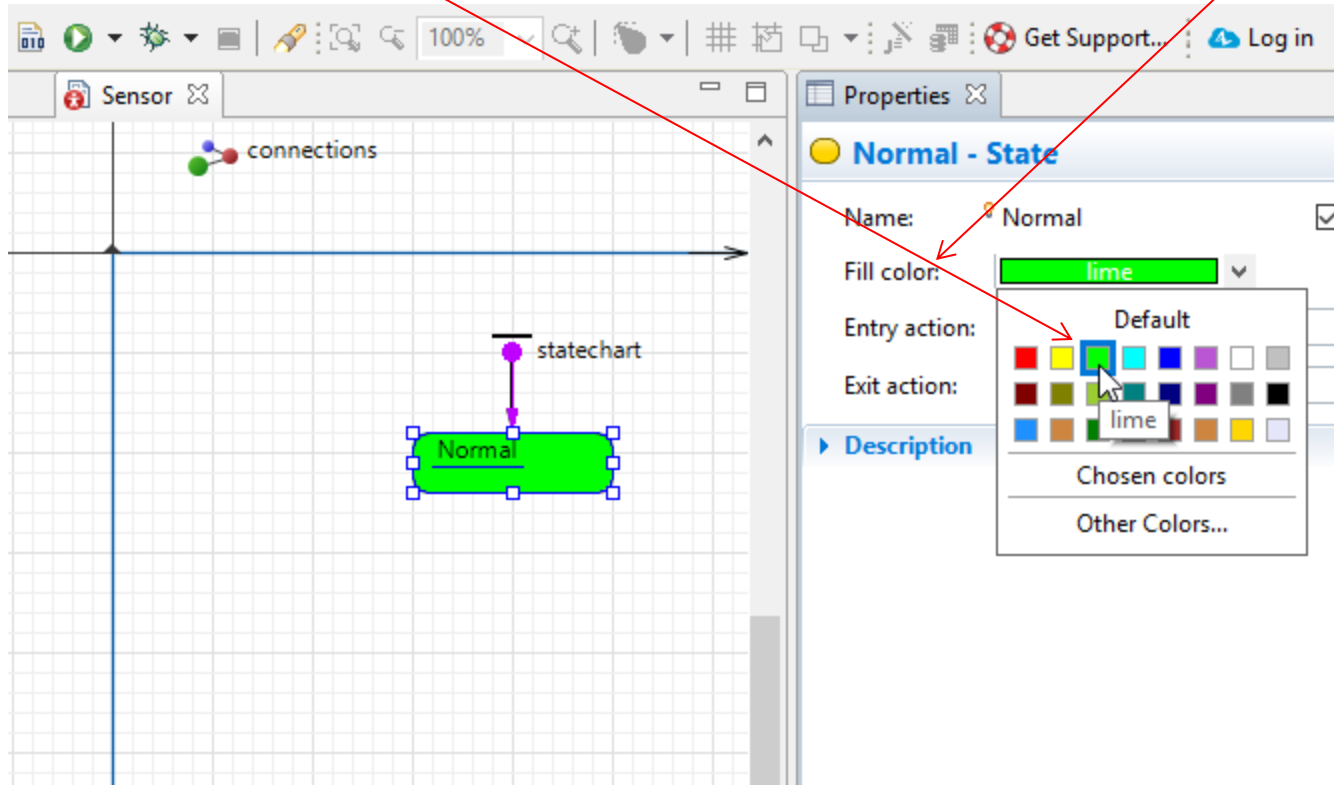
Change the state name to Normal

Click on the state that you just created and in the property window change its name to Normal



Change the state color

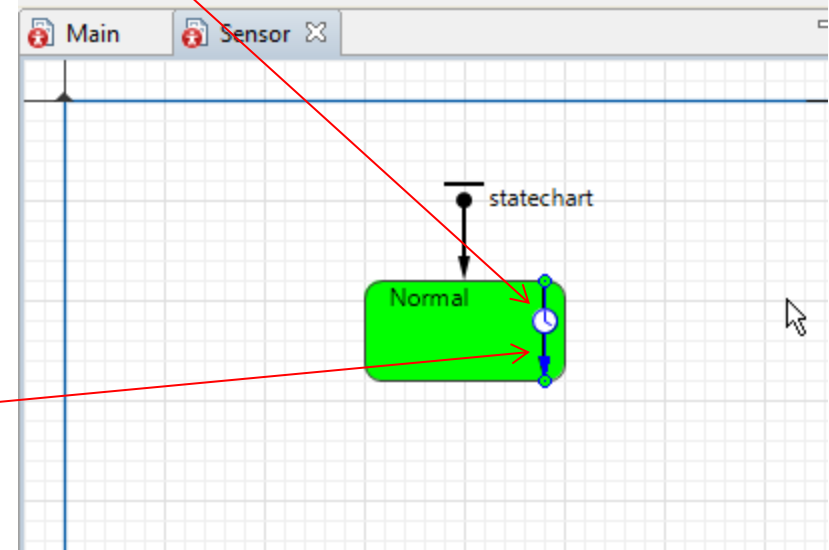
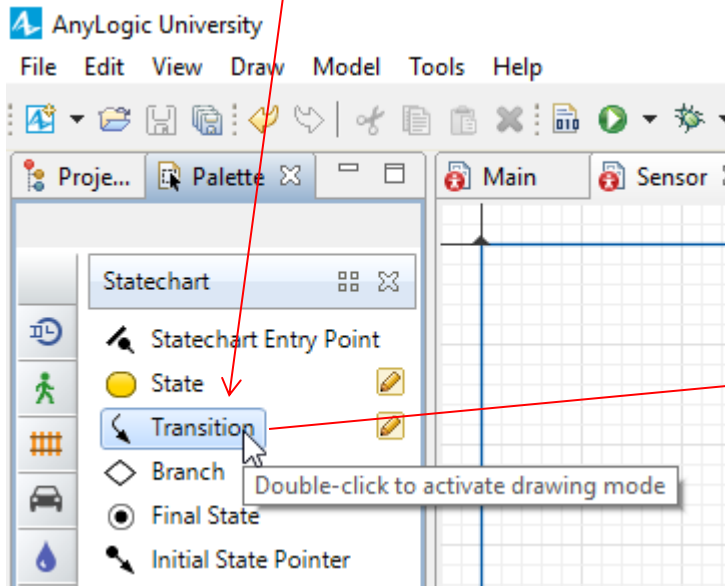
Using the state property window change the Fill color to lime.



Defining some actions

Now that we have created the state we need to add a transition inside it that monitors the flood.

We do this by adding a transition into the Normal state. Drag a Transition and put it inside the Normal state so that its is connected to the borders of the state as shown.



Define Actions for the Transition

Click on the Transition that you just added and open its Properties.

Change the name to sensing.

Select Triggered by: Timeout

Type 1 for the Timeout and choose seconds

The screenshot shows a statechart editor with a statechart on the left and a properties window on the right. The statechart has a state named 'Normal' (green box) with a transition labeled 'sensing' (blue box) that has a clock icon. The properties window is titled 'sensing - Transition' and contains the following fields:

- Name: sensing (with checkboxes for 'Show name' and 'Ignore')
- Triggered by: Timeout (dropdown menu)
- Timeout: 1 (input field) seconds (dropdown menu)
- Action:

```
//calculates distance between the flood and the sensor
distance = distanceTo(main.flood);
// sets the condition. If distance to flood is less than 10 it sets
if (distance < 10)
waterLevel=1;
```
- Guard: (empty input field)

Red arrows point from the text instructions to the corresponding fields in the properties window: 'sensing' to the Name field, 'Timeout' to the Triggered by dropdown, '1' to the Timeout input, and 'seconds' to the Timeout unit dropdown.

Define Transition properties

Make sure that your transition is selected and the property window is open.

Add the following lines of codes into the action section:

```
//calculates distance between the flood and the sensor
```

```
distance = distanceTo(main.flood);
```

```
// sets the condition. If distance to flood is less than 10 it sets the water level from  
0 (no flood) to flood level (1)
```

```
if (distance < 10)
```

```
waterLevel=1;
```

The screenshot displays a software development environment with two windows. The left window, titled 'Sensor', shows a statechart with a green state labeled 'Normal'. A transition labeled 'sensing' is shown on the right side of the state. The right window, titled 'Properties', shows the configuration for the 'sensing - Transition'. The 'Name' is 'sensing', and it is triggered by 'Timeout' with a duration of '1 seconds'. The 'Action' section contains the following code:

```
//calculates distance between the flood and the sensor  
distance = distanceTo(main.flood);  
// sets the condition. If distance to flood is less than 10 it sets  
if (distance < 10)  
waterLevel=1;
```

The 'Guard' section is currently empty. Red lines from the text above point to the code in the 'Action' field of the properties window.

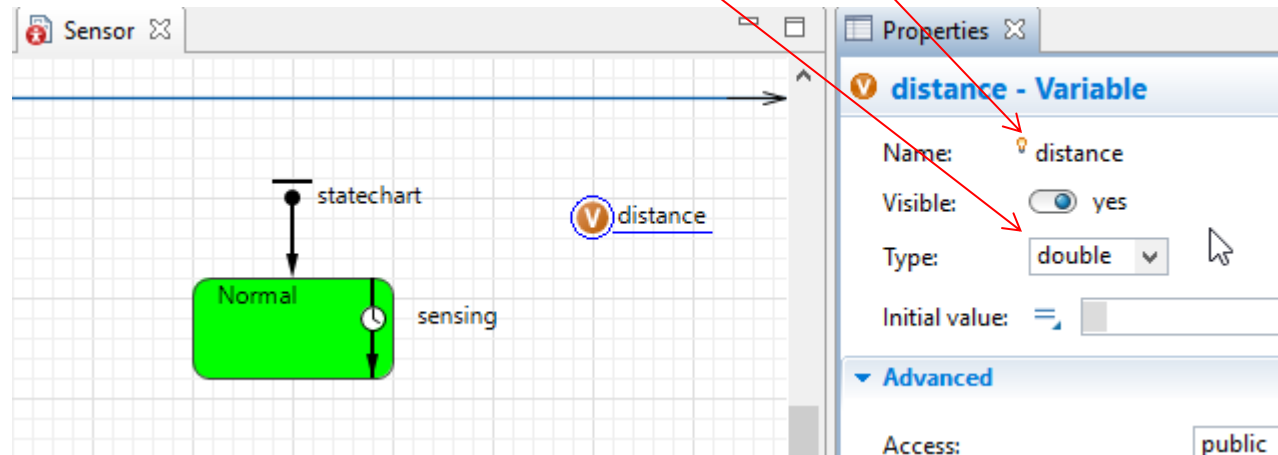
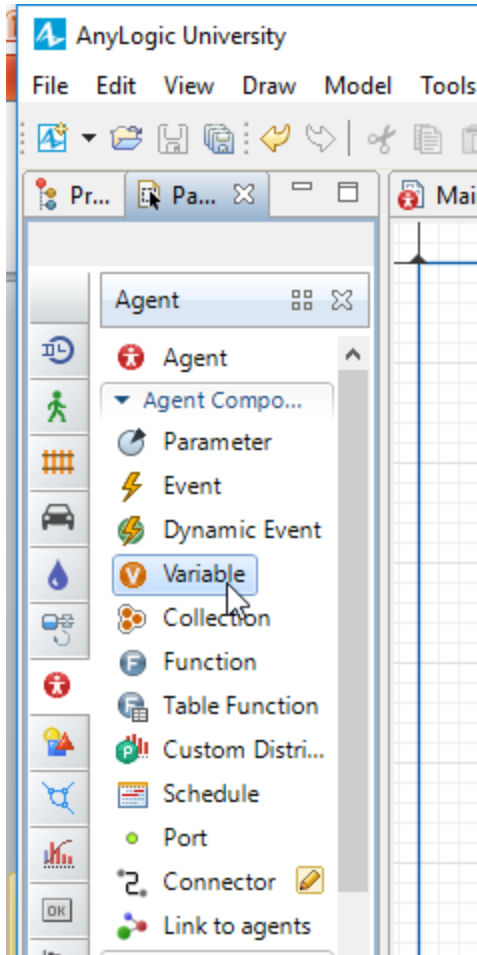
Add distance variable

Since we referred to two variables in our codes in the previous slides, we need to define them.

Drag and drop a variable from the Agent toolset into the Sensor window and change its properties as shown.

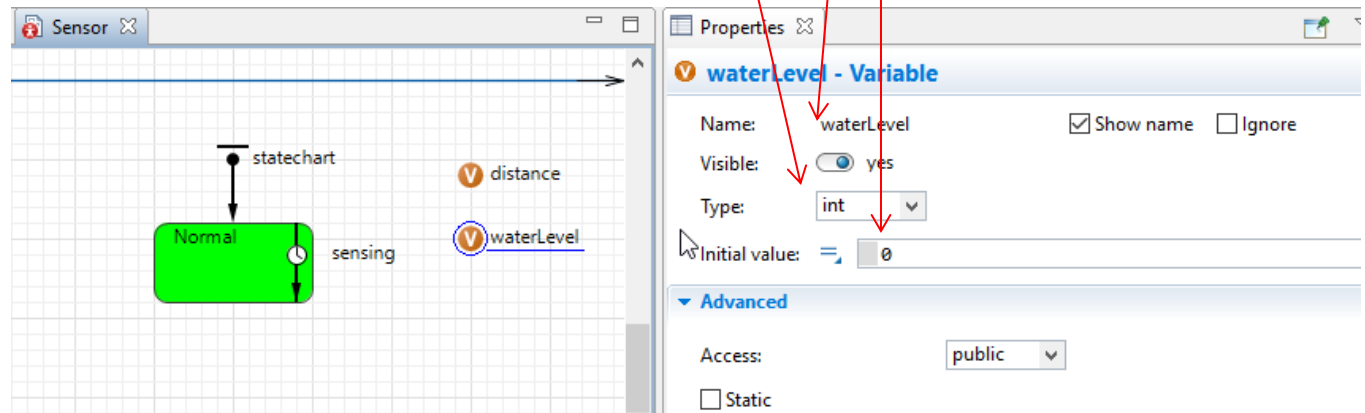
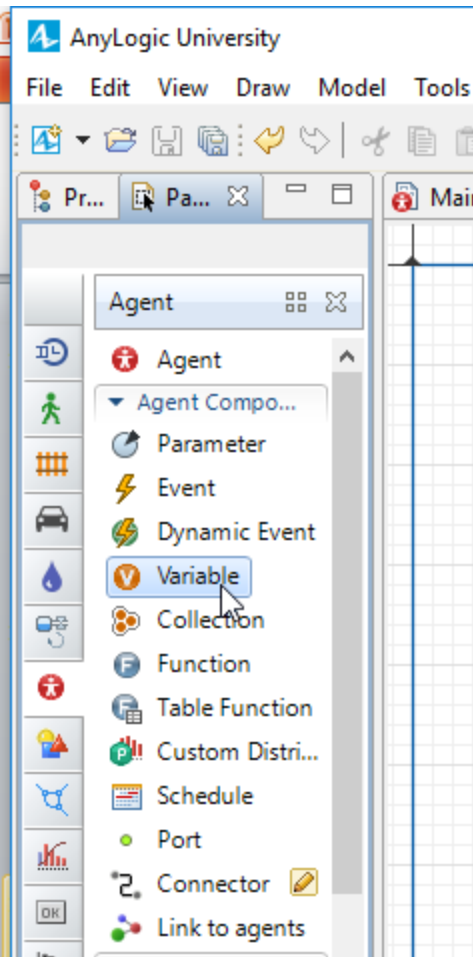
Change its name to: distance

Change its type to : double



Add water level variable

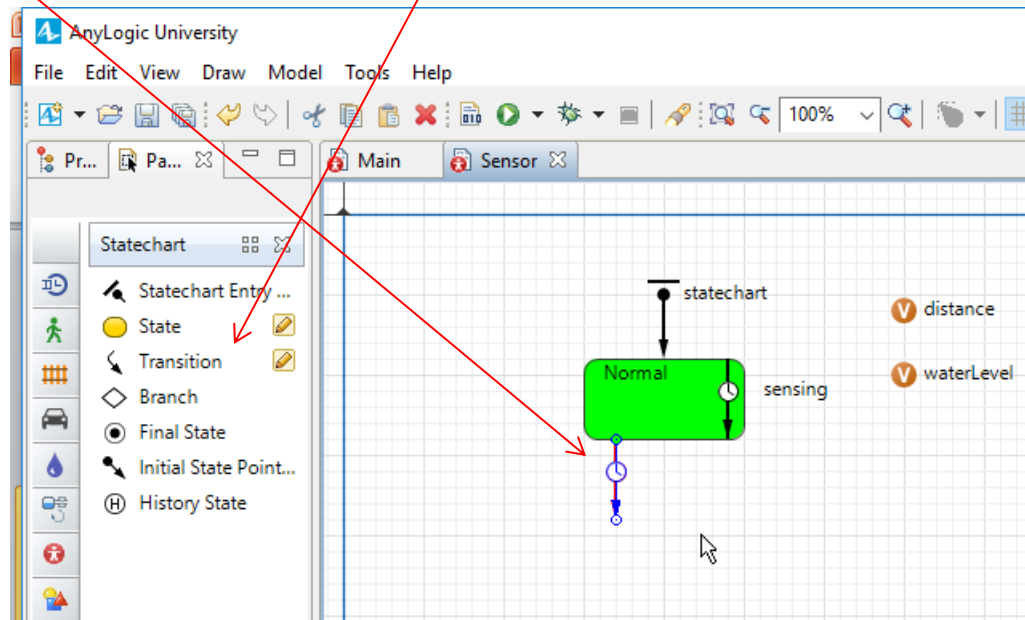
Drag and drop another variable from the Agent toolset into the Sensor window and change its properties as shown.
Change its name to: waterLevel
Change its type to : int
Change the initial value to : 0



Transition from Normal state to Flooding state

So if water level becomes 1 then the state of the sensor changes from Normal to Flooding.

For this to happen we drag and drop a transition to the Normal state as shown:



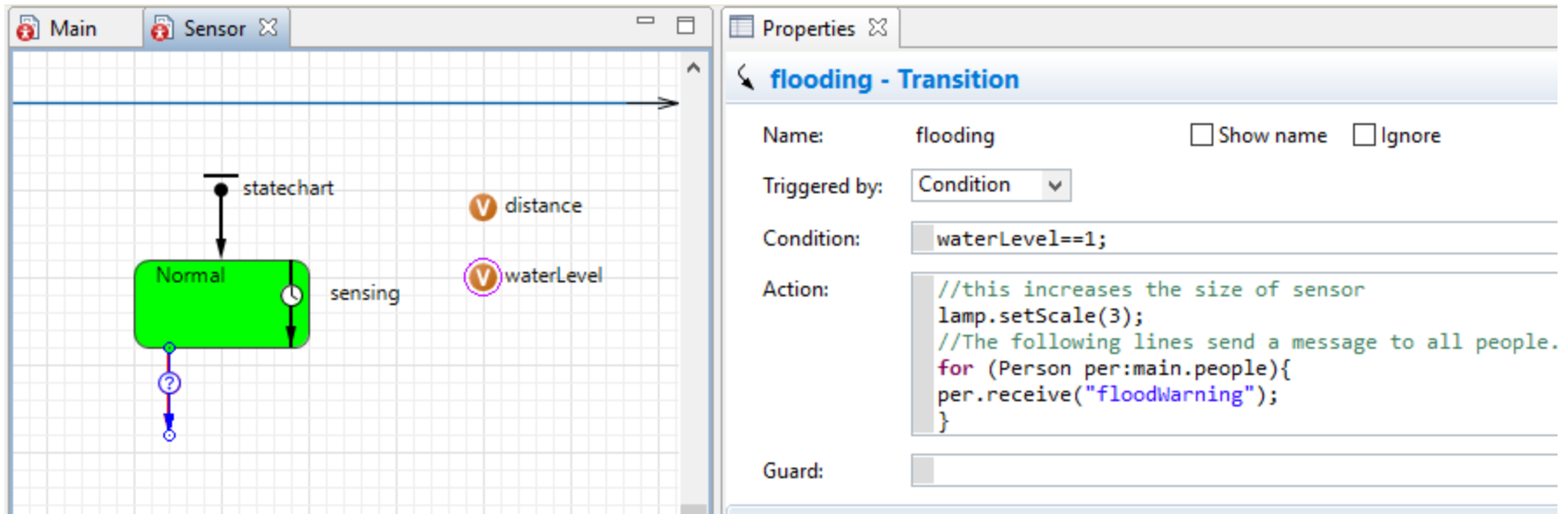
Define properties of the transition

Change the Name to: flooding

Change the Triggered by to: Condition

Add the following line of codes to the Action section:

```
//this increases the size of sensor  
lamp.setScale(3);  
//The following lines send a message to all people.  
for (Person per:main.people){  
per.receive("floodWarning");  
}
```



The screenshot displays a statechart editor interface. On the left, a statechart diagram shows a state named 'Normal' (a green rounded rectangle) with a 'sensing' label. A transition arrow points from the 'Normal' state to the right. The diagram also includes a 'statechart' label, a 'distance' variable (represented by a brown circle with a 'V'), and a 'waterLevel' variable (represented by a purple circle with a 'V').

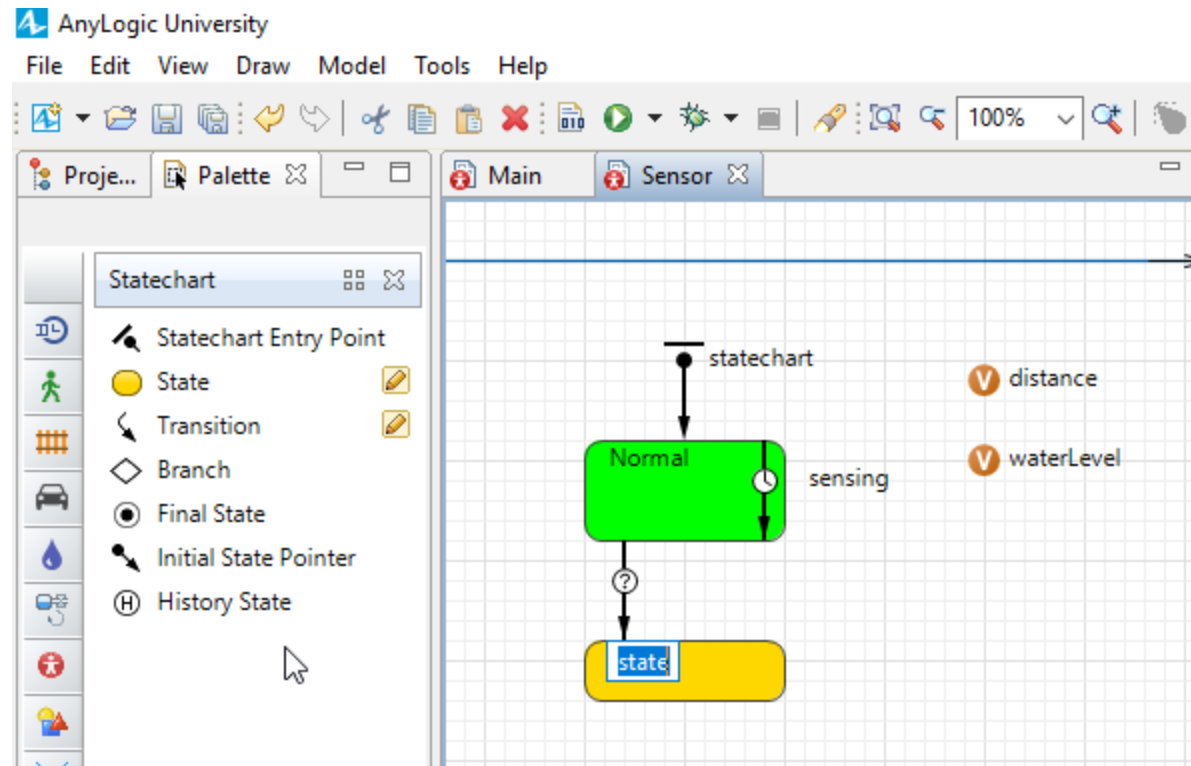
On the right, the 'Properties' panel is open for the 'flooding - Transition'. The configuration is as follows:

- Name: flooding
- Triggered by: Condition
- Condition: `waterLevel==1;`
- Action:

```
//this increases the size of sensor  
lamp.setScale(3);  
//The following lines send a message to all people.  
for (Person per:main.people){  
per.receive("floodWarning");  
}
```
- Guard: (empty)

Ad the Flooding state

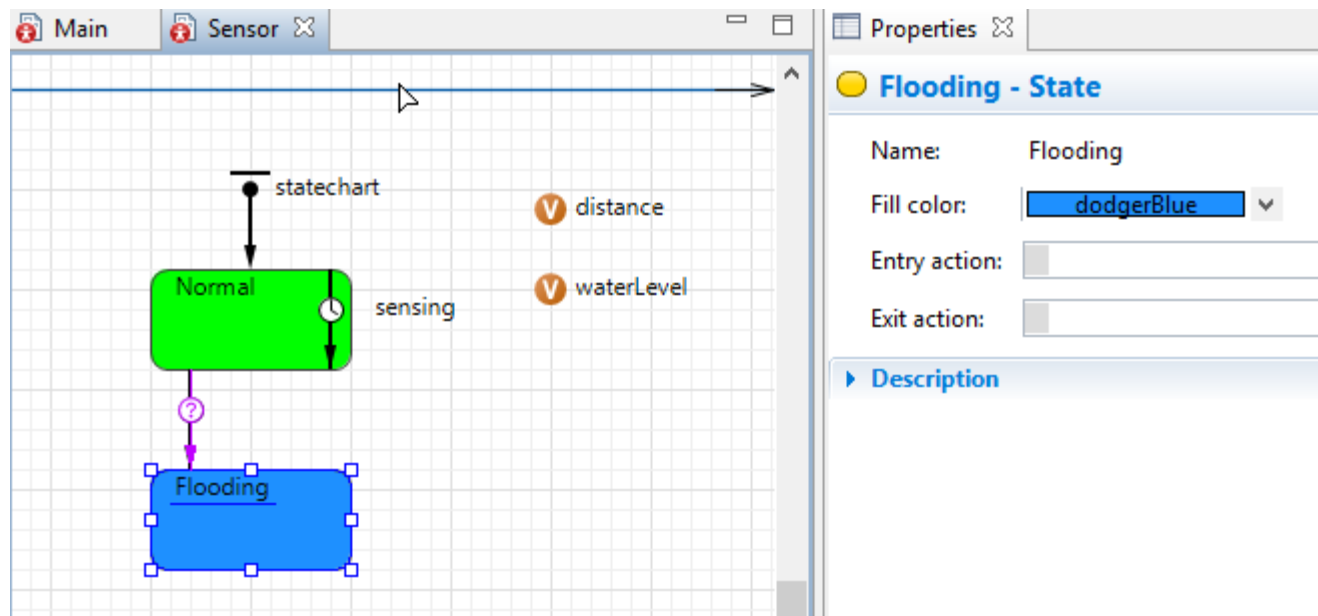
Drag another state from the Statechart tools and drop it into the Sensor window and attach it to the flooding transition as shown



Change the properties of the Flooding state

Change the name t: Flooding

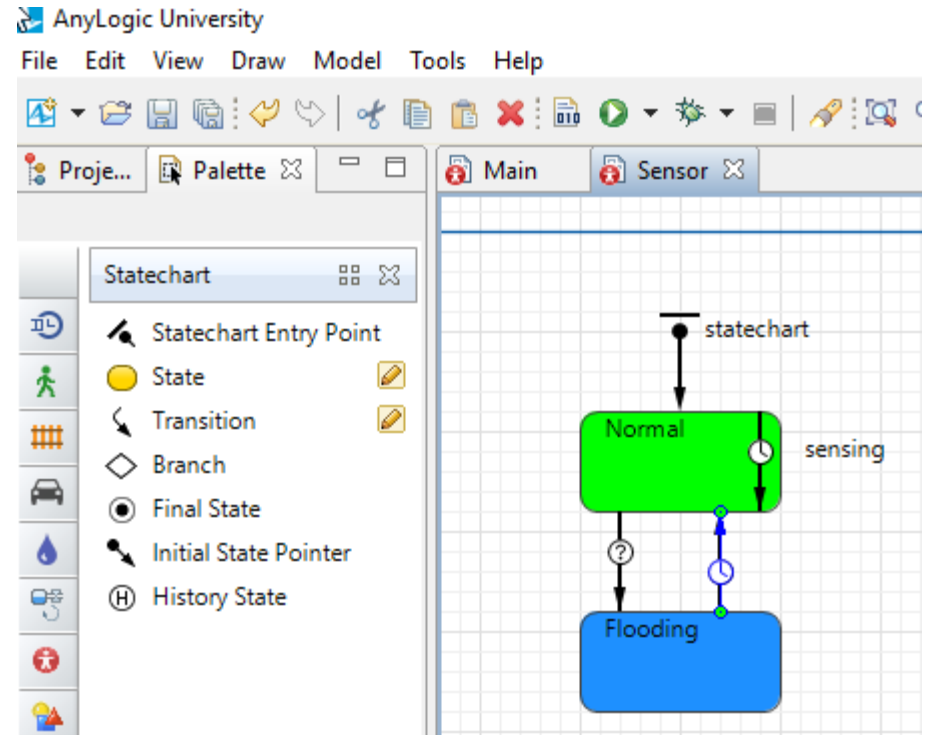
Change the Fill Color to blue



Transition from Flooding state to Normal

We need to add another transition that returns the sensor state from Flooding state to Normal when the water level is no longer at the flooding level.

To do so drag and drop another transition and connect it's end to Flooding state and its head to Normal state.



Define the properties of the new transition

Change Name to: normal

Change Triggered by to : Condition

Change the Condition to: waterLevel==0

Add the following lines of codes to the action section:

```
//this line reduces the size of the sensor symbole to 1
```

```
lamp.setScale(1);
```

```
//These lines inform people that flood situation is normal.
```

```
for (Person p:main.people){
```

```
p.receive("normal");
```

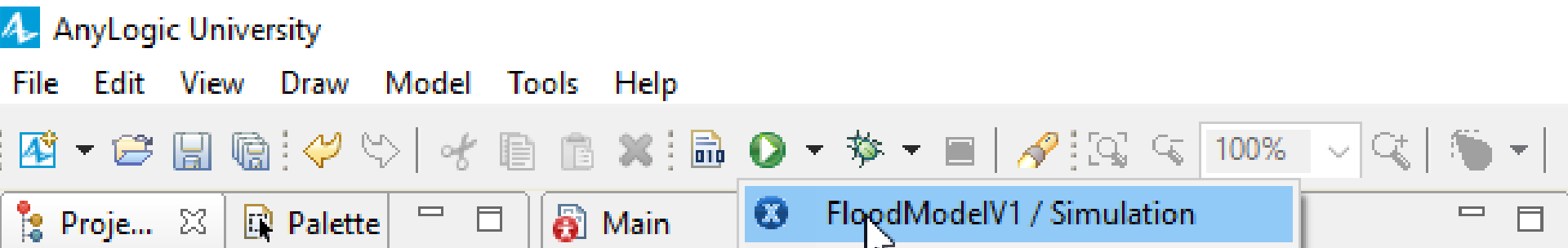
```
}
```

The screenshot displays a statechart editor interface. On the left, a statechart diagram shows a transition from a 'Normal' state (green) to a 'Flooding' state (blue). The 'Normal' state has a 'sensing' label and a clock icon. A transition arrow points from 'Normal' to 'Flooding', with a question mark icon above it. On the right, the 'Properties' panel for the selected transition is shown. The 'Name' is 'normal', 'Triggered by' is 'Condition', and the 'Condition' is 'waterLevel==0'. The 'Action' field contains the following code:

```
//this line reduces the size of the sensor symbole to 1  
lamp.setScale(1);  
//These lines inform people that flood situation is normal.  
for (Person p:main.people){  
p.receive("normal");  
}
```

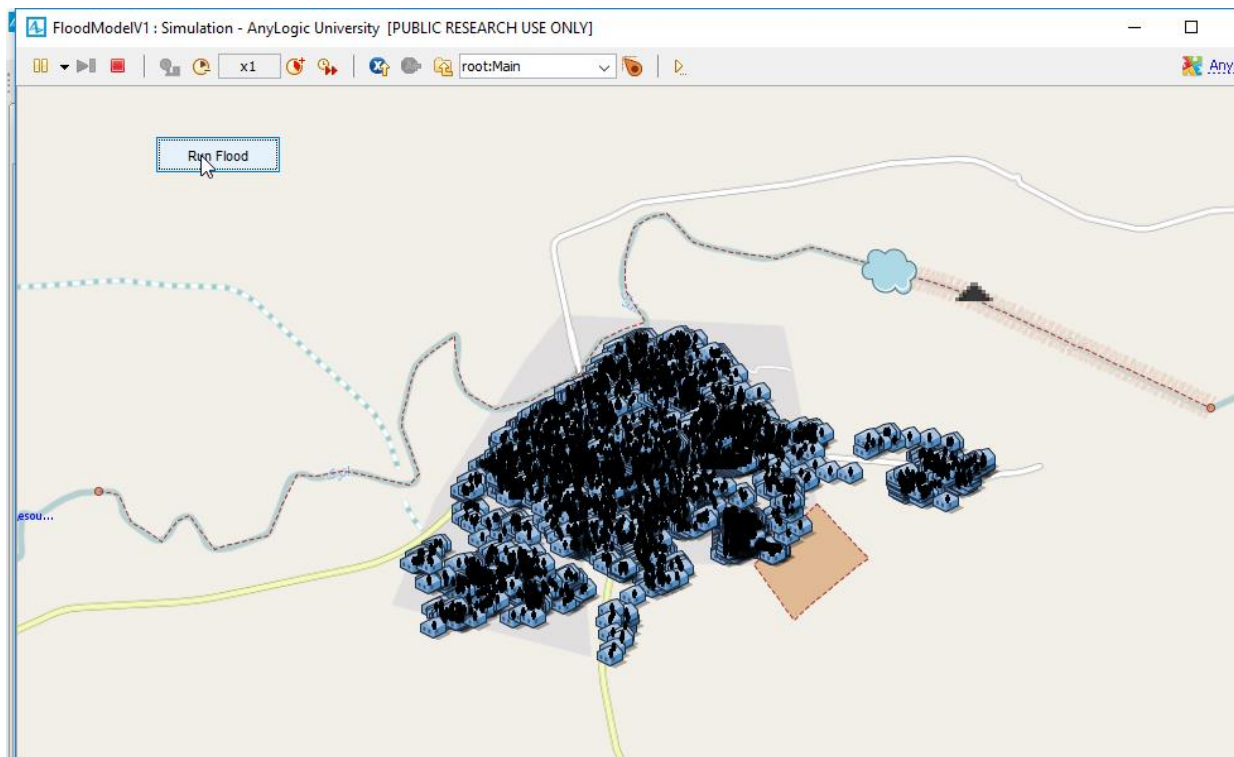
The 'Guard' field is empty. Below the properties panel, there is a 'Description' section.

Run the model



Results

If you do not see any error, you will see that after flood reaches to the Sensor, sensor symbol increases. It also sends the warning to people that we will see them in the next lesson.



- This was a simple way of creating behavior for an agent using statecharts.
- We will create more complicated statecharts in the future lessons.
- We will explain how to define people behavior in lesson 5.
- Save your project.