




[Tutorials](#) > [Supply Chain GIS \(AB + DE\)](#)

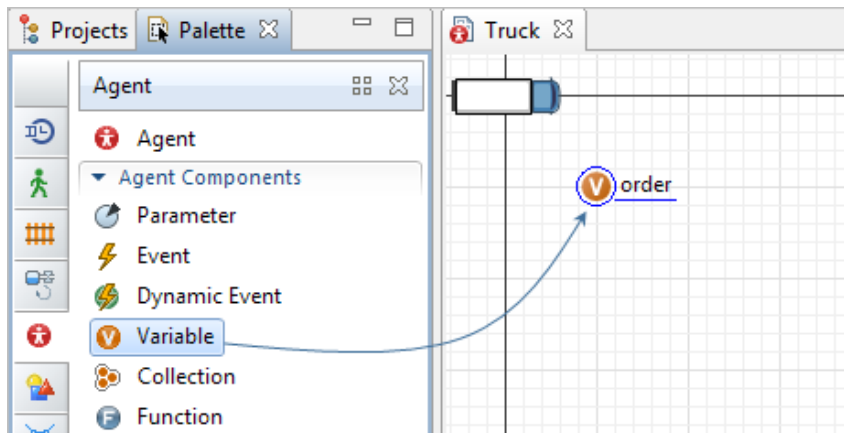
Phase 6. Defining trucks' movement


This is the last phase of our Supply Chain GIS model development process. We are ready now to define the trucks' movement between the points of the supply chain on the map.

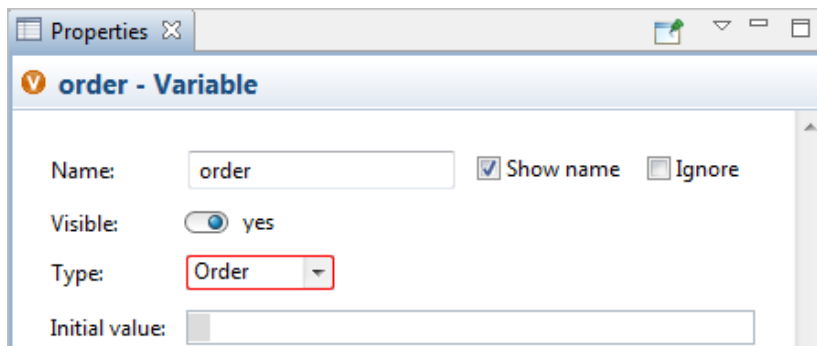
We only need to specify the destination points for the trucks. All the routes are always available from the AnyLogic routing server, and by routes we mean real roads on the map.

To enable access to orders

1. Open the  **Truck** agent type diagram from the **Projects** view.
2. Drag the  **Variable** object from the  **Agent** palette. Name the variable `order`.





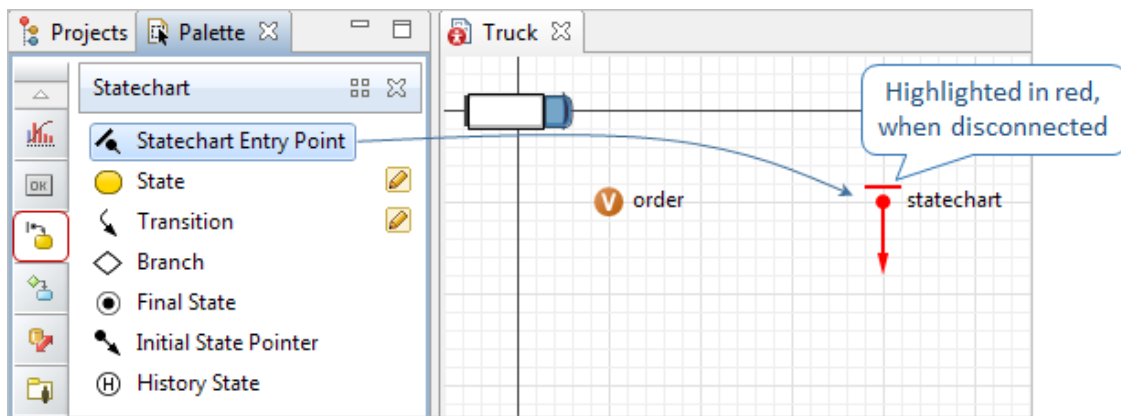
3. Choose its **Type**: `Order`. We will use this variable to access orders and the information they contain (as they are creating on  **Retailer**) to enable trucks' movement in the statechart.



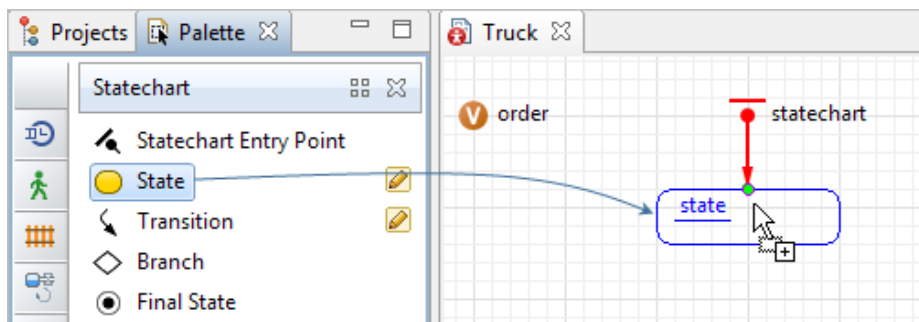
Any truck which received an order, should spend some time to load the product, then it moves to the retailer that requested this product, unloads the goods, and then it has to return back to the distributor. Each of these activities can be modeled as a state: truck is in the state of driving to a retailer. The transitions between the states switch a truck from one activity to the following action.

To define the truck movement

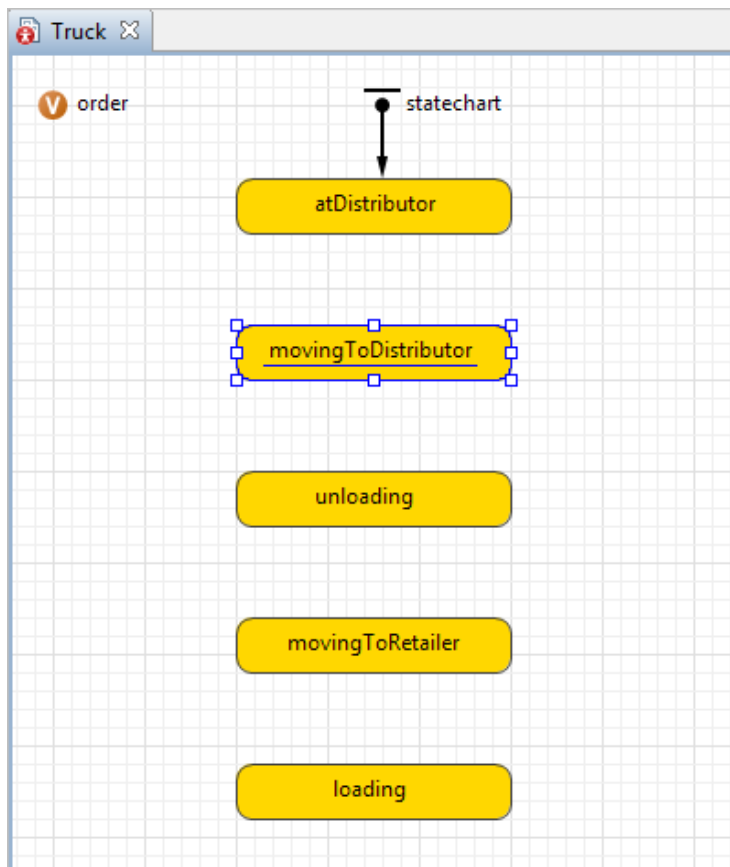
1. Open the  **Statechart** palette in the **Palette** view. We will start building a statechart with the  **Statechart Entry Point** object.



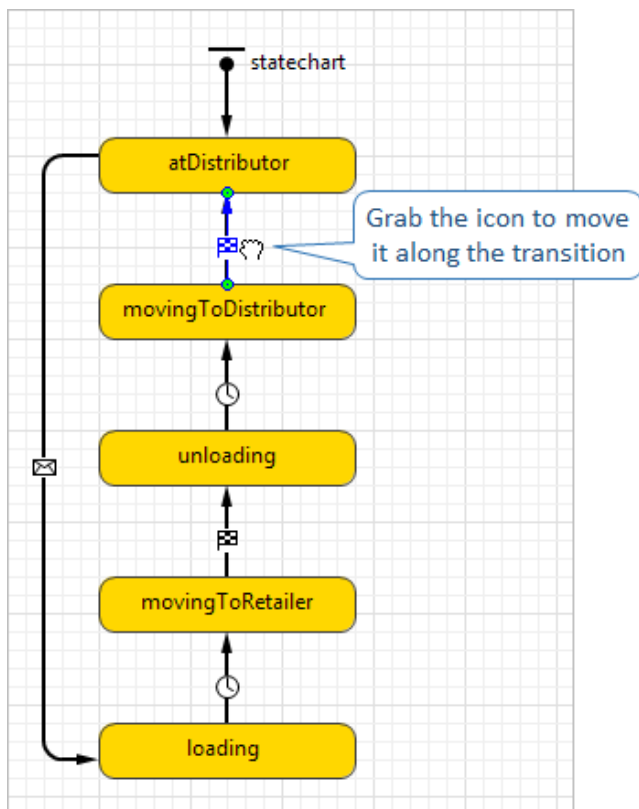
2. Next, drag from the palette the **State** object and connect it to the statechart entry point. When the elements are connected successfully, the point of connection is highlighted in the cyan color.



3. Building a statechart, we connect states with transitions. Add five states first on the **Truck** diagram and name them in the following way. Simply put them on the diagram one under another leaving some space for the transitions between them.



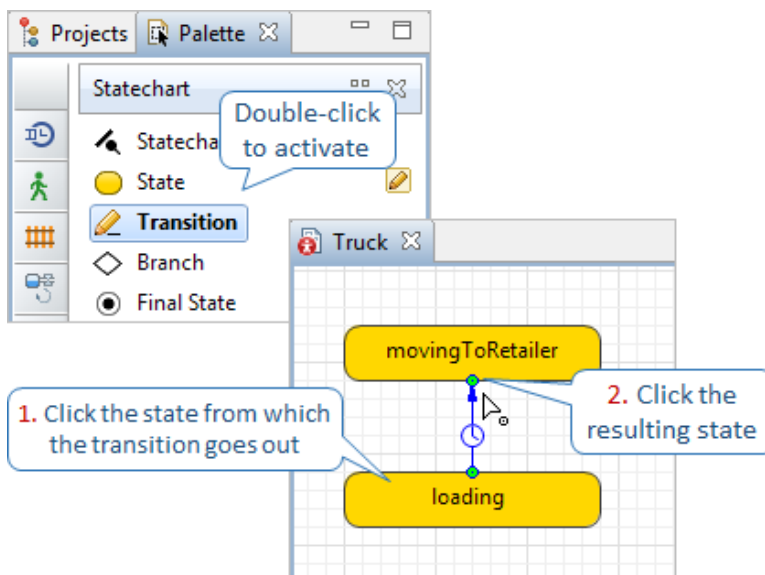
4. We will define the logic of the statechart in the properties of [transitions](#). When we finish this statechart, it will look like in the figure below. You can see there several types of transitions. Let us draw the transitions first, and then we will take a close look at each of them.



5. To draw a transition, double-click the **Transition** element in the **Palette** to activate its drawing mode. Then you can draw a transition by clicking first in the state that it goes from, then the state that it leads to.

Be careful with the transitions' direction: one transition goes "down" from `atDistributor` to `loading`, and the transitions from `loading` to `atDistributor` go "up".

By default, the trigger type of any transition is *Timeout*, and that is why it has an icon of a clock: . First you can draw "empty" timeout transitions, then we will define their properties, and change their appearance too.

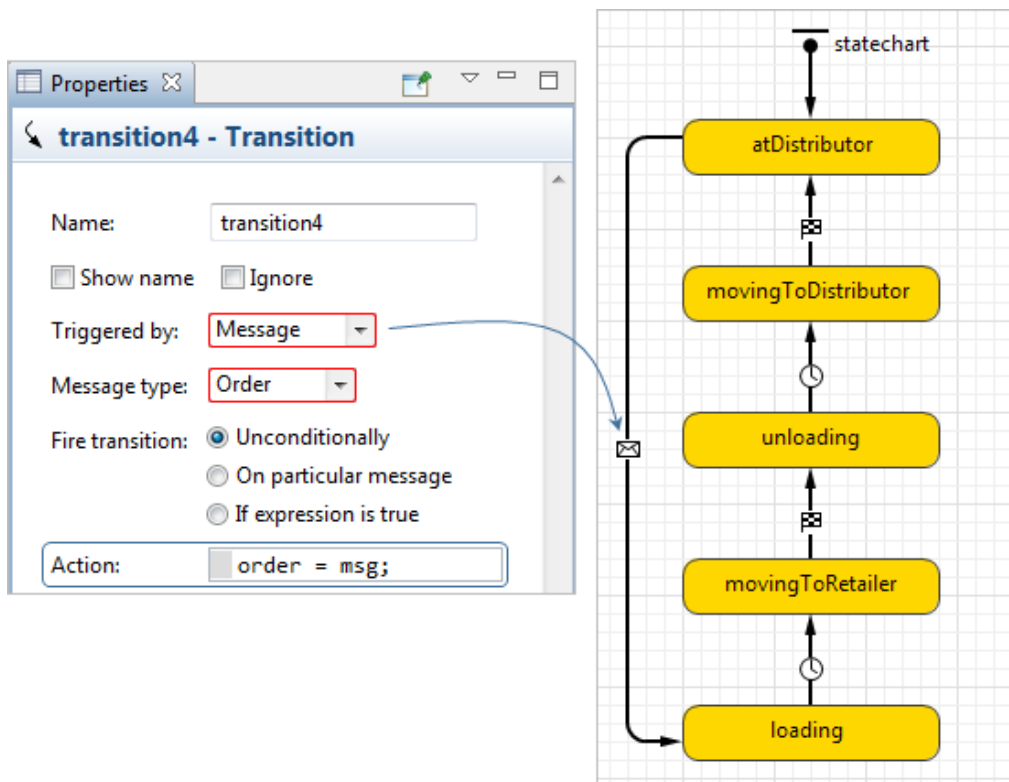


6. Let us have a look at the transition from the state `atDistributor` to `loading`.

`atDistributor` is the initial state in this statechart, we have set the `trucks[...]` initial location in the same GIS point as the distribution center. There they are when they receive an order.

In the previous phase, we have defined the `takeTruck` object to send an `order` message to the trucks. This is where we set up the trucks to receive that message.

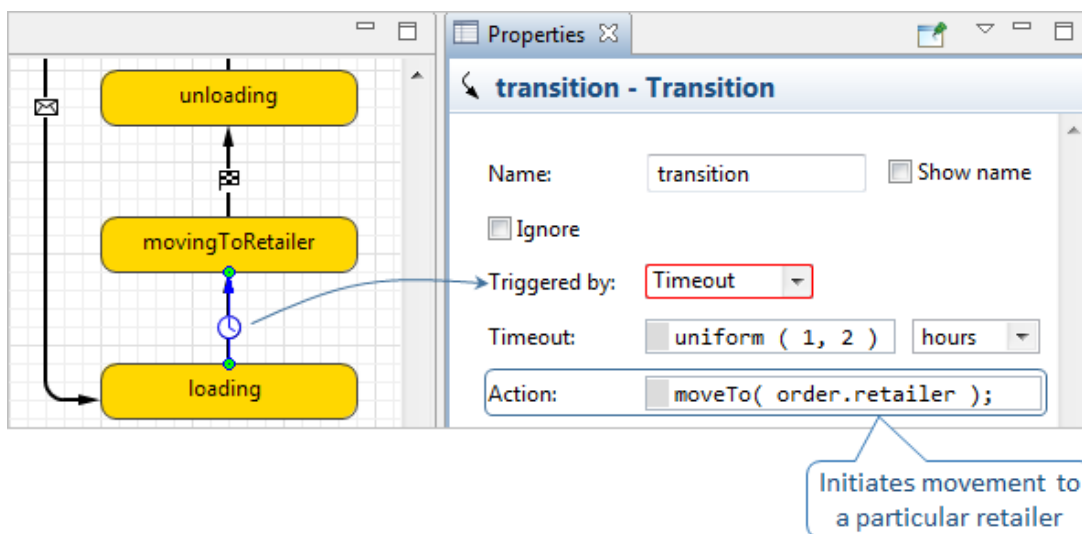
We change the transition's trigger type to *Message* , our message is of *Order* type. In the **Action** edit box, we assign the `order` variable the value of the local variable `msg`.



7. Now a truck is in the `loading` state. Loading takes time, so we will use the *Timeout* ⌚ trigger type for this transition.

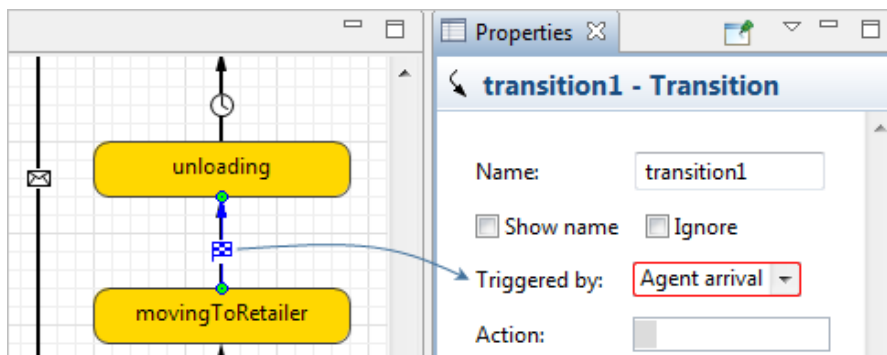
Set the **Timeout** to a couple of hours. Our next state is `movingToRetailer`, and we will switch a loading truck into this state after the specified timeout with the little piece of code in the **Action** edit box: `moveTo(order.retailer);`

`moveTo()` is a popular function to define the agent movement. You can make an agent move to another agent, a node, even a geographic place. Here, we do not know which retailer exactly may order the product this or next time, so we set it to be that particular retailer that has sent the order that triggered the whole statechart in the first transition.



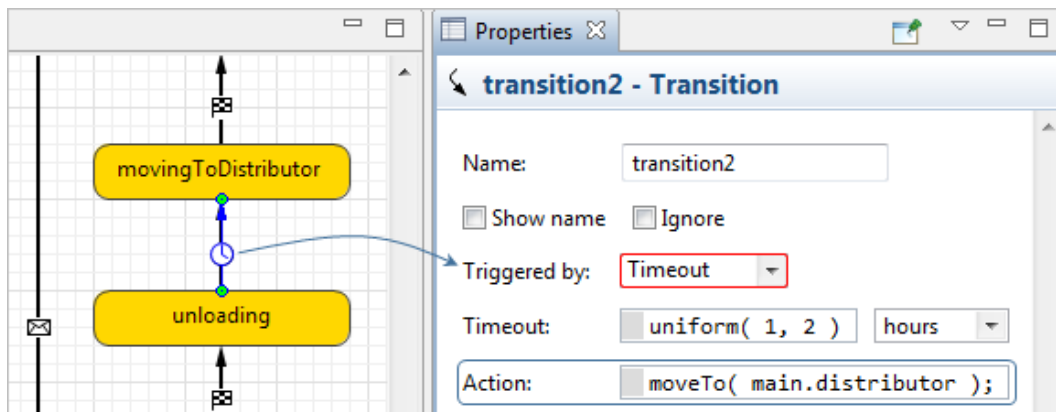
You can read more about movement functions in the [Agent movement](#) help section.

8. The `unloading` state should start as soon as the truck reaches its point of destination. The transition to `unloading` is **Triggered by: Agent arrival**.



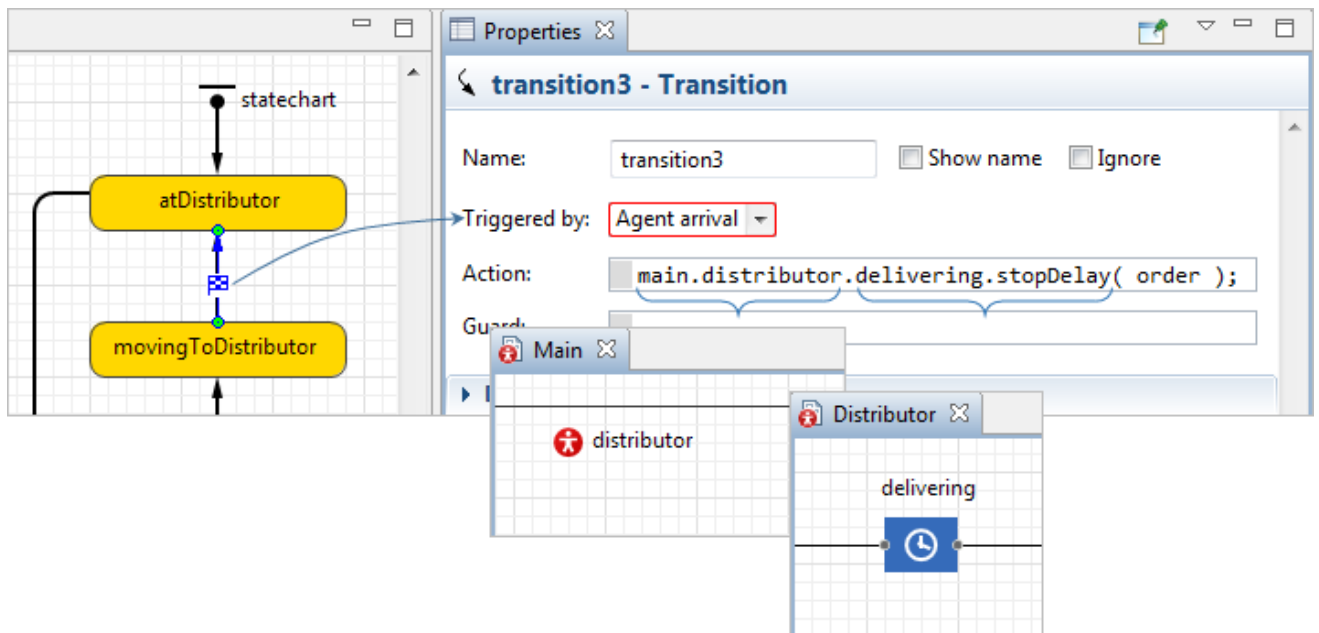
9. Unloading itself takes some time, we trigger the transition from `unloading` to `movingToDistributor` by *Timeout*. Let it again be a couple of hours. And after this timeout, we want the truck to go back to its base, the distributor.

The **Action** defines this movement: `moveTo(main.distributor);`




10. Once the truck arrives to its distributor, it is back in the `atDistributor` state, ready to take and process another order. This transition is triggered by *Agent arrival*.

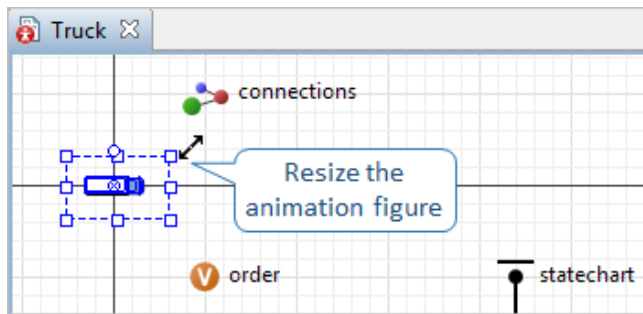
The **Action** here controls the flowchart on the **Distributor** diagram. The delay time lasted as long as a truck needed to deliver the product to the retailer, and now it is time to finish delay and release the resources.



`stopDelay(agent)` is a function of the **Delay** library object (follow the link to read more about **Delay** and its functions). We call this function for the argument `order` which is our variable on the **Truck** diagram. Its type is **Order**, the same agent type that the flowchart processes.

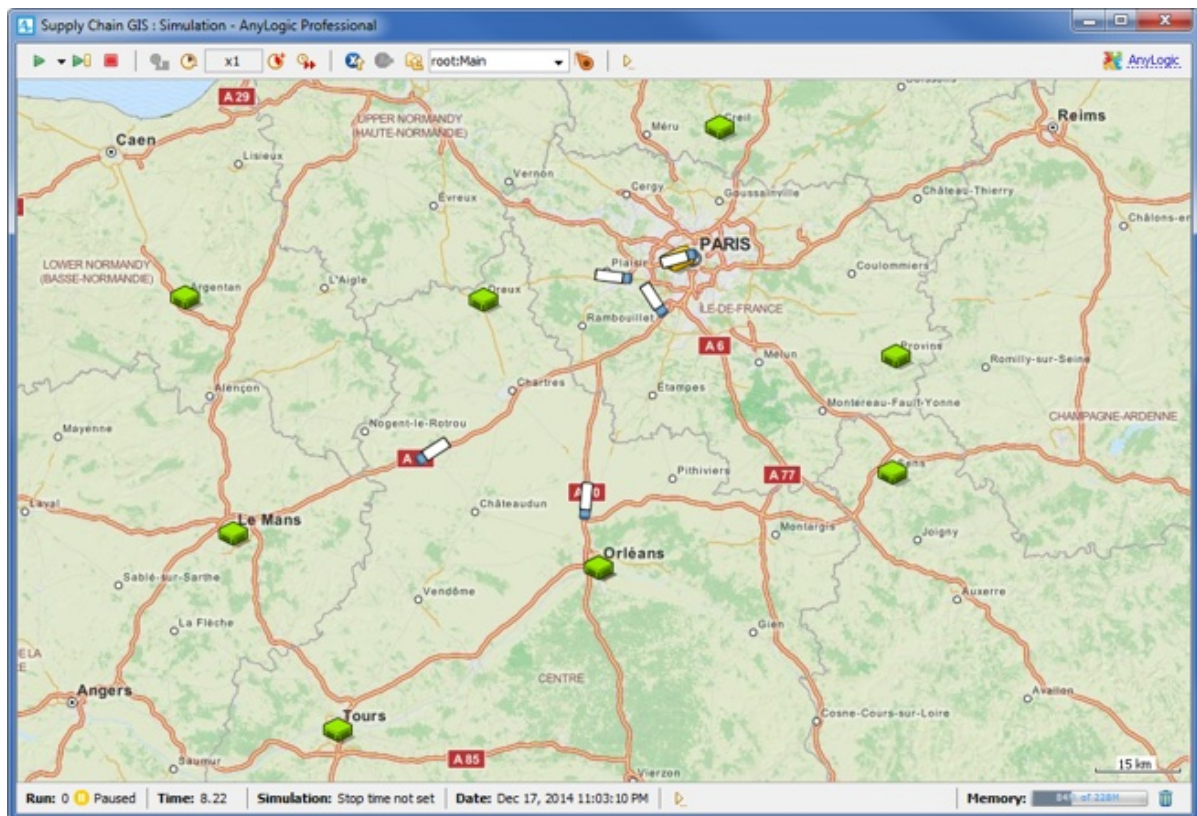
11. Now we are back to the first state. We have finished creating the statechart for the trucks.

As the final step, let us make the animation figure of a truck smaller. There are [advanced scaling properties](#) for the animation figures of agents on  Main designed specially for the GIS map animation management.



To run and explore the model

1. This is the final stage, and we can run the model and observe the trucks going back and forth delivering product according to the requests that the center receives.

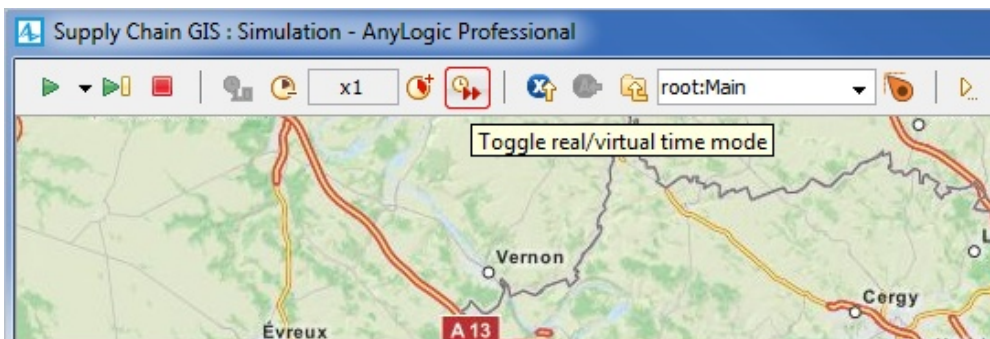


2. If you zoom in to a smaller scale, you will see how the trucks follow real roads on the map:



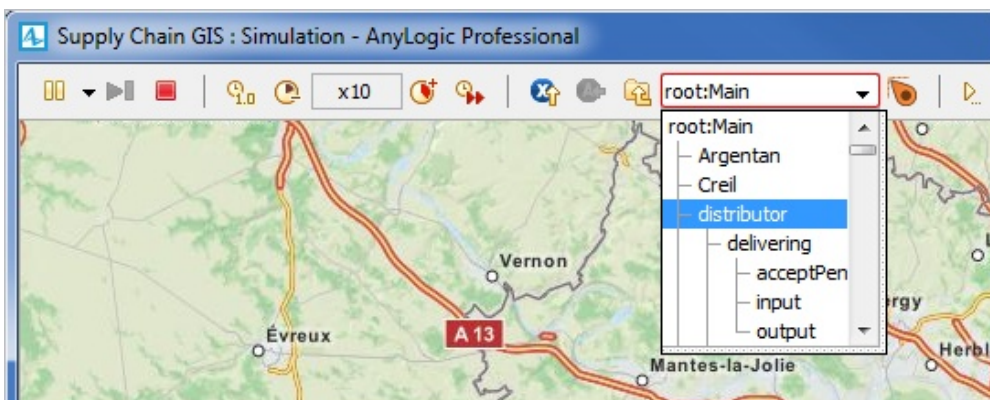
AnyLogic generalizes animation at the run-time for the sake of performance, although it always builds the routes with the best precision, both at the design-time and at runtime of the model, to calculate distances and travel time in compliance with the actual data. You can customize the [generalization](#) settings of the routes to make animation look even more realistic.



3. You can run the model in the virtual time mode, i.e. as fast as possible, to cache the routes quicker. After the routes are cached, the model performance will be higher.



To go back to the real time mode, click the button  on the presentation window toolbar.

4. You can explore the agents in this model and observe their components. Click in the model explorer and select the agent type you would like to check.



5. For instance, we can take a look at the  Distributor to check on the trucks utilization via fleet statistics. To go back to the  Main presentation, click the "level up" button.

If necessary, you can always check the reference model that we provide for your convenience.

Reference model: [Supply Chain GIS - Phase 6](#)



[Phase 5. Processing orders at the distributor](#)