**Model Description:**

This model is used to show a method of how to select correct vehicle to pick multiple model entities by a user defined order. In addition, this model provide a way to show how the logic of vehicle handling transportation request and the logic of Global.VisitRequestQueue.

If you want to know how to retrieve a certain request from Global.VisitRequestQueue. Please refer to example “**VehicleRequestRejectionAndRetreive”**

**Model Assumptions:**

There is two type of vehicles: Car1 and Car2. Each of the vehicle has a population of one. We have five model entities which is created by a data table. The data table contains several information like the release time, which vehicle it rides on, the sequence that it follows to ride the vehicles and so on.

**Details for Building the Model:**

Data Table Preparation:

* Create a data table in the Data window and change the table name to “Orders”
* Create several columns:

1. Name “Priority”(Integer Property): which is used to define the sequence that the model entity rides on the vehicle.
2. Name “ReleaseTime”(Real Property): which is used to define the release time of the model entity.
3. Name ”Symbol”(Integer Property): which is used to define the symbol of the model entity.
4. Name “Color”(String Property): which is used to show the color of the model entity.
5. Name “Rider”(Transporter Property): which is used select which type of vehicle the model entity should ride.

* The detailed value setting you can find in the model.

System Setup:

* Place one Source object, one Sink object, one TransferNode object, one BasicNode and one ModelEntity in the Facility window.
* Change the Source object name to “OrderSource”; change the Sink object name to “Depot”; change the TransferNode object name to “PickLocation”; change the BasicNode name to “Home”; change the ModelEntity name to “Order”。
* Connect the OrderSource and PickLocation with TimePath and connect the PickLocation and Depot with Path. Connect the PickLocation and Home with Path.
* Select the Path and change *Type* to ‘Bidirectional’.
* Select the Order and click “Add Additional Symbols” in the “Symbols” ribbon. Add four additional symbols and change the color based on the data table.
* Select the OrderSource and change the *Arrival Mode* to ‘Arrival Table’ and *Arrival Time Property* to ‘Orders.ReleaseTime’.
* Within the Facility window, click Detached Queue button under Animation ribbon. Draw a queue and change Queue State to ‘Global.VisitRequestQueue’.

Defining the Vehicle:

* In the Facility window, within the Libraries, right click the Vehicle and left click subclass. In the navigation window, you will have a customize Transporter object: MyVehicle.
* Within the Project Library window, drag MyVehicle and place it to the Facility window twice and change their name to “Car1” and “Car2” respectively.
* Select the Car1, Car2 and change the *Initial Desired Speed* to 0.3 (Meters per Second).
* Change *Initial Node(Home)* to ‘Home’
* Within the Navigation window, select MyVehicle. And in the Definitions window, create one state variables:

1. Name “CurretTaskIndex”(Integer State Variable): which is used to match the Priority in the data table to ensure the vehicle follows a correct order to pick the model entities.

Defining the TransferNode:

* Select PickLocation and change *Entity Destination Type* to ‘Specific’. Change *Node Name* to ‘Input@Depot’. Also Change the *Ride On Transporter* to ‘True’, *Transporter Name* to ‘Orders.Rider’
* Click the PickLocation, click the Draw Queue in the Appearance ribbon, find and click RidePickupQueue. Then draw a queue under the PickLocation.

Defining Events:

* In the Definitions window, create an event named “SystemStart”. This is used to indicate that the Cars can start to pick the ModelEntities.
* In the Facility window, place a button by clicking the button in Animation ribbon. Change the *Button Text* to ‘SystemStart’ and change *Event Name* to ‘SystemStart’.

Defining ModelEntity:

* Within the Navigation window, select ModelEntity. Then in the Definitions window, create an Integer State Variable named “Priorty” which is used to store the ModelEntity Priority.

Process Logic for Assigning Values ModelEntity :

* Within the Processes window, click on the Create Process button on the Process ribbon to add a new process. Specify the Name as “AssignValueAfterCreated”.
* Add an Assign step and change *State Variable Name* to ‘ModelEntity.Picture’ and *New Value* to ‘Orders.Symbol’
* Add additional values to the State Variables of ModelEntity based on the data table.
* In the Facility window, select the output node of OrderSource, change *Entered(Add-On Process Triggers)* to ‘AssignValueAfterCreated’.

Process Logic for Deciding the SystemStart:

* Within the Definition window, create a Boolean State Variable named “StartSymbol”.
* Within the Processes window, click on the Create Process button on the Process ribbon to add a new process. Specify the Name as “StartSystem”.
* Select the created process and change Triggering Event Name to ‘SystemStart’.
* Add an Assign step and change *State Variable Name* to ‘StartSymbol’ and *New Value* to ‘True’.

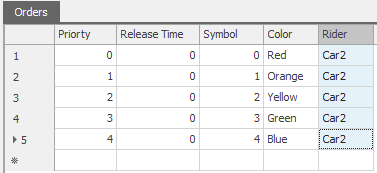
Process Logic for Unloading of Cars:

* Within the Processes window, click on the Create Process button on the Process ribbon to add a new process. Specify the Name as “CarUnload”.
* Add an Assign step and change *State Variable Name* to ‘MyVehicle.CurrentTaskIndex’ and *New Value* to ‘MyVehicle.CurrentTaskIndex+1’.
* In the Facility window, select MainCars and change *Unloaded(Add-On Process Triggers)* to ‘CarUnload’.

Process Logic for Evaluating Transport Request for Cars:

* Within the Processes window, click on the Create Process button on the Process ribbon to add a new process. Specify the Name as “CarEvaluatingTask”.
* First Place a Decide step, change *Condition Or Probability* to ‘StartSymbol == True’.
* Then, place another Decide step at the True exit from the above Decide step. Change *Condition Or Probability* to ‘Orders.Priorty == MyVehicle.CurrentTaskIndex’. This is used to ensure the Vehicle pick the ModelEntity follows the correct order.
* At the True exit from the above Decide step, Add an Assign step and change *State Variable Name* to ‘Token.ReturnValue’ and *New Value* to ‘True’.
* At the False exit from the above Decide step, Add an Assign step and change *State Variable Name* to ‘Token.ReturnValue’ and *New Value* to ‘False’.
* Drag all False exit from the Decide steps to the above Assign step.
* In the Facility window, select Car1 and Car2, change *Evaluating Transport Request(Add-On Process Triggers)* to ‘CarEvaluatingTask’.

**Run the Simulation:**

****

The above table is the datatable used in this example.

Start simulation. **BEFORE** five Orders reach PcikLocation. You need to click the “SystemStart” button. And then, when the five Orders reach PcikLocation , you can find there are five Orders in the Queue under the PickLocation and four Orders in the GlobalVisitQueue. And you can also find that Car2 starts to drive to PickLocations and then pick the Orders by the sequence defined in the column of “Priorty”.

If you click the “SystemStart” button **AFTER** five Orders reach PcikLocation. You will see that there are five Orders in the Queue under the PickLocation and five Orders in the GlobalVisitQueue. No Car drives to PickLocation.

**Interpretation of the Simulation:**

When, the orders reach PickLocation, Each Order will trigger a transportation request to the corresponding Transporter we defined in the TransferNode. If we already let “StartSymbol” equels to “True”, then, four requests are rejected, and they are moved to Global.VisitRequestQueue. Only one request is accepted, here is the request called by Red Order. After the Transporter moving red Order to the Depot (Which means finishing a transportation request), if the Global.VisitRequestQueue is not empty, the Transporter will re-evaluate the requests in the Global.VisitRequestQueue and the request called by Orange Order is accepted.

On the other hand, if we don’t let “StartSymbol” equels to “True” before the orders reach PickLocation , after evaluating the requests, all of them are rejected. The Transporter will re-evaluate those requests unless the Transporter accept and finish another request.