**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 shows how to reject vehicle transportation request and how to retrieve certain rejected request from the Global.VisitRequestQueue.

**Model Assumptions:**

There is two type of vehicles: MainCars and BackupCars. Each of the vehicle has a population of two. 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 “RiderType”(String Property): which is used select which type of vehicle the model entity should ride.
6. Name ”RiderIndex”(Integer Property): which is used to select the particular vehicle in the certain vehicle type.

* 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. Repeat this one time and create two vehicle models: MyVehicle and MyVehicle2.
* Rename the two vehicle models to “MainCar” and “BackupCar”.
* Within the Project Library window, drag MainCar and place it to the Facility window and change its name to “MainCars”.
* Within the Project Library window, drag BackupCar and place it to the Facility window and change its name to “BackupCars”.
* Select the MainCars, BackupCars and change *Initial Number in System* to 2. Also change the *Initial Desired Speed* to 0.3 (Meters per Second).
* Change *Initial Node(Home)* to ‘Home’
* Within the Navigation window, select MainCar. And in the Definitions window, create two state variables:

1. Name “Index” (Integer State Variable): which is used to indicate the vehicle index in its population.
2. 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.

* For the BackupCar, also create the above two state variables.

Defining TransporterList:

* In the Definition Window, create a TransporterList named “MyCars” and it contains two transporters: MainCars and BackupCars.

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 Type* to ‘Select From List’ and *Transporter List* to ‘MyCars’
* Click the PickLocation, click the Draw Queue in the Appearance ribbon, find and click RidePickupQueue. Then draw a queue under the PickLocation.

Defining Tokens:

* Within the Definitions window, click on the Add Token button on the Tokens ribbon to add a token. Specify the name as “EvaluetingTaskToken”. This token is used for processes that help us accept or reject request for vehicles.
* Add a String State Variable named “RiderType” in the created token. Also change its Initial State Value to ‘MainCars’

Defining Events:

* In the Definitions window, create an event named “StartMainCar”. This is used to indicate when the MainCar can start to pick the ModelEntities.
* Create another event named “StartBackupCar”. This is used to indicate when the BackupCar 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 ‘StartMainCar’ and change *Event Name* to ‘StartMainCar’.
* Place another button. Change the *Button Text* to ‘StartBackupCar’ and change *Event Name* to ‘StartBackupCar’.

Defining ModelEntity:

* Within the Navigation window, select ModelEntity. Then in the Definitions window, create a String State Variable named “RiderType” which is used to store the vehicle type the ModelEntity should ride on.
* Create another Integer State Variable named “RiderIndex” which is used to store the index of the reserved vehicle in its population.
* Create another Integer State Variable named “Priorty” which is used to store the ModelEntity Priority.

Process Logic for Setting MainCar Index:

* Within the Processes window, click on the Create Process button on the Process ribbon to add a new process. Specify the Name as “SetMainCarsIndex”.
* Add an Assign step and change *State Variable Name* to ‘MainCar.Index’ and *New Value* to ‘MainCars.Population.IndexOfItem( MainCar )’. So, we can use MainCar.Index to select a certain Vehicle in its population.
* In the Facility window, select MainCars, change the *Run Initialized (Add-On Process Triggers)* to ‘SetMainCarsIndex’

Process Logic for Setting BackupCar Index:

* Within the Processes window, click on the Create Process button on the Process ribbon to add a new process. Specify the Name as “SetBackupCarsIndex”.
* Add an Assign step and change *State Variable Name* to ‘BackupCar.Index’ and *New Value* to ‘BackupCars.Population.IndexOfItem( BackupCar )’. So, we can use BackupCar.Index to select a certain Vehicle in its population.
* In the Facility window, select BackupCars, change the *Run Initialized (Add-On Process Triggers)* to ‘SetBackupCarsIndex’

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 ‘ChangeColor’.

Process Logic for Deciding the Start of Vehicles:

* Within the Definition window, create a Boolean State Variable named “Start”.
* Within the Processes window, click on the Create Process button on the Process ribbon to add a new process. Specify the Name as “StartSystem1”.
* Select the created process and change Triggering Event Name to ‘StartMainCar’.
* Add an Assign step and change *State Variable Name* to ‘Start’ and *New Value* to ‘True’.
* Within the Processes window, click on the Create Process button on the Process ribbon to add a new process. Specify the Name as “StartSystem2”.
* Select the created process and change Triggering Event Name to ‘StartBackupCar’.
* Add an Assign step and change *State Variable Name* to ‘Start’ and *New Value* to ‘True’.

Process Logic for Evaluating Transport Request for MainCars:

* Within the Processes window, click on the Create Process button on the Process ribbon to add a new process. Specify the Name as “MainCarEvaluatingTask”.
* Select the process and change *Token Class Name (Advanced Options)* to ‘EvaluetingTaskToken’.
* Then Create one row in *Input Arguments:*

1. Change *Name* to ‘RiderType’; Change *State Variable Name* to ‘EvaluetingTaskToken.RiderType’

* First Place a Decide step, change *Condition Or Probability* to ‘Start == True’.
* Then, place another Decide step at the True exit from the above Decide step. Change *Condition Or Probability* to ‘(Orders.RiderType == EvaluatingTaskToken.RiderType)&&(Orders.RiderIndex == MainCar.Index)’ . This ensures that the ModelEntity ride on the correct vehicle.
* Then, place another Decide step at the True exit from the above Decide step. Change *Condition Or Probability* to ‘Orders.Priorty == MainCar.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 MainCars, change *Evaluating Transport Request(Add-On Process Triggers)* to ‘MainCarEvaluatingTask’ and change *Rider Type* to ‘MainCars’.

Process Logic for Evaluating Transport Request for BackupCars:

* Within the Processes window, click on the Create Process button on the Process ribbon to add a new process. Specify the Name as “BackupCarEvaluatingTask”.
* Select the process and change *Token Class Name (Advanced Options)* to ‘EvaluetingTaskToken’.
* Then Create one row in *Input Arguments:*

1. Change *Name* to ‘RiderType’; Change *State Variable Name* to ‘EvaluetingTaskToken.RiderType’

* First Place a Decide step, change *Condition Or Probability* to ‘Start == True’.
* Then, place another Decide step at the True exit from the above Decide step. Change *Condition Or Probability* to ‘(Orders.RiderType == EvaluatingTaskToken.RiderType)&&(Orders.RiderIndex == BackupCar.Index)’ . This ensures that the ModelEntity ride on the correct vehicle.
* Then, place another Decide step at the True exit from the above Decide step. Change *Condition Or Probability* to ‘Orders.Priorty == BuckupCar.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 BackupCars, change *Evaluating Transport Request* *(Add-On Process Triggers)* to ‘BackupCarEvaluatingTask’ and change *Rider Type* to ‘BackupCars’.

Process Logic for Unloading of MainCar:

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

Process Logic for Unloading of BackupCar:

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

Process Logic for Retreiving the Transport Request for MainCars:

* Within the Processes window, click on the Create Process button on the Process ribbon to add a new process. Specify the Name as “RetreiveTransportRequestForMainCar”.
* Select the created process and change *Triggering Event Name* to ‘StartMainCar’
* First, add a Delay step and change Delay Time to ‘Math.Epsilon’. This will cause the event to be handled last on the calendar of events that occur at the same specific time.
* Then, place a Decide step, change *Condition Or Probability* to ‘Start == True’.
* Then, place a Search step at the True exit from the above Decide step. Change *Entity Type* to ‘MainCars’, *Search Type* to ‘ForwardSumExpression’, *Match Condition* to ‘Candidate.MainCar.ResourceState == 0’ and *Limit* to ‘Infinity’. This will search all idle MainCars in the population.
* At the Found exit from the Search step, add a Decide step and change *Condition Or Probability* to ‘Global.VisitRequestQueue.NumberWaiting>0’. This will check whether there is Rejected request or not.
* In the Definitions window, create an Integer State Variable named “MainCarRequestIndex”. This is used to hold the index in the next Find step.
* At the True exit of the above Decide step, add a Find step. Change *Index Variable Name* to ‘MainCarRequestIndex’, *Ending Index* to ‘Global.VisitRequestQueue.NumberWaiting’ and *Search Condition* to ‘Global.VisitRequestQueue.ItemAtIndex(MainCarRequestIndex).ModelEntity.RiderType == "MainCars"’. This is used to check whether the request is for MainCars or not.
* At the Found exit of the above Find step. Place a PlanVisit step. Change the *Entity Type(Advanced Options)* to ‘SpecificObject’ and *Entity Object* to ‘MainCars[Global.VisitRequestQueue.ItemAtIndex(MainCarRequestIndex).ModelEntity.RiderIndex]’. This will let the correct MainCar to respond for the request.

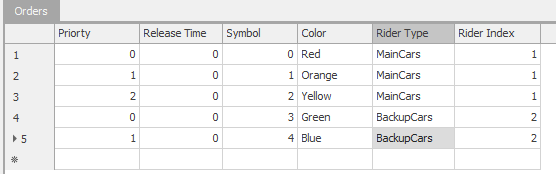
Process Logic for Retreiving the Transport Request for BackupCars:

* Within the Processes window, click on the Create Process button on the Process ribbon to add a new process. Specify the Name as “RetreiveTransportRequestForBackupCar”.
* Select the created process and change *Triggering Event Name* to ‘StartBackupCar’
* First, add a Delay step and change Delay Time to ‘Math.Epsilon’. This will cause the event to be handled last on the calendar of events that occur at the same specific time.
* Then, place a Decide step, change *Condition Or Probability* to ‘Start == True’.
* Then, place a Search step at the True exit from the above Decide step. Change *Entity Type* to ‘BackupCars’, *Search Type* to ‘ForwardSumExpression’, *Match Condition* to ‘Candidate.BackupCar.ResourceState == 0’ and *Limit* to ‘Infinity’. This will search all idle BackupCars in the population.
* At the Found exit from the Search step, add a Decide step and change *Condition Or Probability* to ‘Global.VisitRequestQueue.NumberWaiting>0’. This will check whether there is Rejected request or not.
* In the Definitions window, create an Integer State Variable named “BackupCarRequestIndex”. This is used to hold the index in the next Find step.
* At the True exit of the above Decide step, add a Find step. Change *Index Variable Name* to ‘BackupCarRequestIndex’, *Ending Index* to ‘Global.VisitRequestQueue.NumberWaiting’ and *Search Condition* to ‘Global.VisitRequestQueue.ItemAtIndex(BackupCarRequestIndex).ModelEntity.RiderType == "BackupCars"’. This is used to check whether the request is for BackupCars or not.
* At the Found exit of the above Find step. Place a PlanVisit step. Change the *Entity Type(Advanced Options)* to ‘SpecificObject’ and *Entity Object* to ‘BackupCars[Global.VisitRequestQueue.ItemAtIndex(BackupCarRequestIndex).ModelEntity.RiderIndex]’. This will let the correct BackupCar to respond for the request.

**Run the Simulation:**

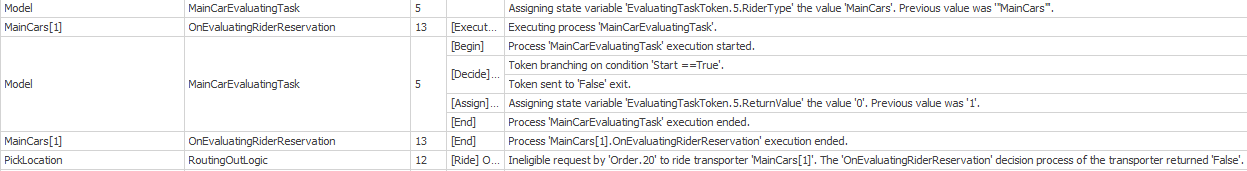
Start simulation. When five Orders reach PcikLocation. You can find there are five Orders in the Queue under the PickLocation and also five Orders in the GlobalVisitQueue. Then by clicking the button, you will find the MainCars or BackupCars pick the Orders based on the information in the data Table.

**Interpretation of the Simulation:**



This is the data table. So you can find The MainCars[1] should pick Red, Orange and Yellow Orders. BackupCars[2] should pick Green and Blue Orders.

When the Orders reach the PickLocation, Based on the Trace, Each Order will try to request and based on the add-on process we created, the request could be accepted or rejected.



The above figure is an example of how the process reject the request.

If a request is not accepted by any of the vehicle. The request will be sent to the GlobalVisitRequestQueue. You can find trace information below.



After Click the Button, The RetreiveTransportRequest process will let the vehicles search the GlobalVisitRequestQueue and Re-do the EvaluatingTransportRequest process. If the request is accepted, Then The vehicle will move to PickLocation and pick the corresponding Order.