AutoStore concept model overview

Creating the storage

The shape of the storage is set in the global table “AutoStoreLayout”. The cell represents one cell of the storage. An “X” signifies a storable slot and “PU”, “PI” and “PO” are ports. The second letter defines what the port can be used for: “PI” act only as input ports and “PO” only as output, whereas “PU” (Universal) can do both. Empty cells are just that. The size of the storage must not exceed 99 cells in either direction!

	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9	Col 10	Col 11	Col 12	Col 13	Col 14	Col 15	Col 16	Col 17	Col 18	Col 19	Col 20
Row 1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X				
Row 2	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X				
Row 3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X				
Row 4	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X				
Row 5	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X				
Row 6	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X				
Row 7	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X				
Row 8	PI	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 9	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 10	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 11	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 12	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 13	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 14	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 15	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 16	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 17	PI	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 18	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 19	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 20	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 21	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 22	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 23	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Row 24	X	X	PO	X	X	X	X	PO	X	X	X	X	PO	X	X	X	X	PO	X	X

Figure 1: Example of the layout table

If there is already a storage in place, running the code in the script window toward the bottom of the screen will delete it. This script is saved as a model script under the name “ResetStore” and can be restored by loading it via the folder symbol, should the script window be closed. On the next model reset, a new storage will be created based on the aforementioned table.

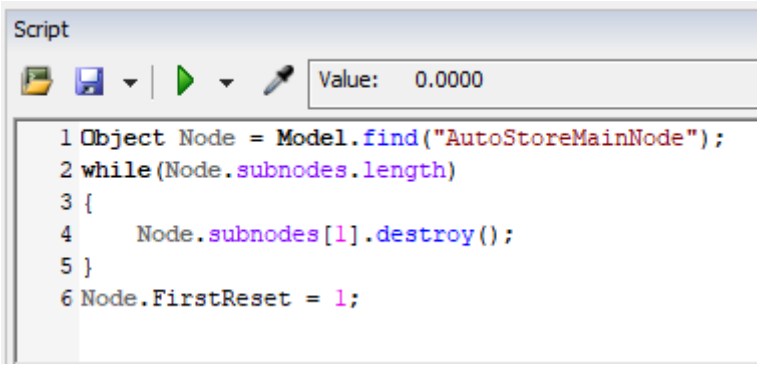

Figure 2: Script window with code to delete the current storage

Three more options are present as labels on the "AutoStoreMainNode" object (MainNode). Here you can control how many boxes can be stacked on top of each other (Height), how high those boxes are (BoxHeight) and how many robots should be servicing the storage (RobotQuantity). Note: When changing the number of robots, the reset button has to be pressed an additional time before running the model to properly update the group they are assigned to. Also, additional robots will start in the same location as "Robot_1" and thus, might not be visible until they start moving.

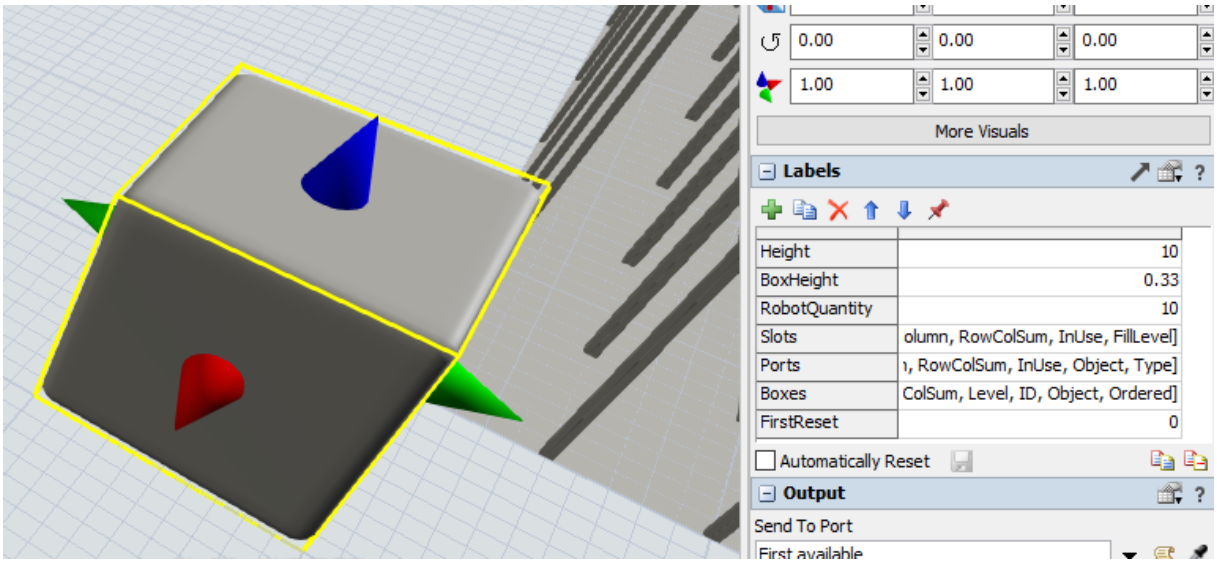

Figure 3: "MainNode" object and its labels

Storing and requesting items

In order to put an item into the storage, simply place it inside an input or universal port. This can be done in any way you might put an item into a queue, processor or other fixed resource. Note that all items put inside the storage have to have a unique identifying value (integer) assigned to them as a label called "ID". When an item enters a port, a task to store it will be dispatched to a robot as soon as one is available.

In order to request an item be delivered to an output or universal port, send a message to the MainNode, with the parameters set as follows:

- The first parameter signifies what kind of message was just received and should read “Request”
- The second is the ID of the item you want to retrieve
- The third is the “RowColSum” number of the port, the item should be delivered to. This number consists of the row and column of the port, written as one number and be found on each cell in the label of the same name.

Example: A message with the parameters (“Request”, 135, 2403) would have the item with ID 135 delivered to the port at position (24, 03).

Make sure that the item you attempt to retrieve is actually stored, as the model will otherwise display an error message and stop. An item is “stored” as soon as it is lowered to its storage position by a robot.

The MainNode will send a message with the parameters (“BoxAvailable”, >3DBoxObject<, >BoxID<) to itself once the item is placed in the port. You can listen to this event to determine when the box is ready to be picked up.

What the model can’t do

Robots do not avoid each other and will path through each other. Furthermore, the robots are not aware of any gaps in the storage and treat the full extend of the layout table as “driveable”. As such they might travel over gaps or missing corners of the storage.

The ports are a very basic implementation of interface position with the storage and do not possess any internal logic or restrictions. An unlimited number of items may be placed inside each port, as well as a theoretically infinite amount of simultaneous requests made.

As such to measure whether storage is able to keep up with the tasks dispatched to it, the model measures the average number of tasks of active tasks/requests per port and the number of tasks currently waiting for an available robot. Both of these numbers are shown in graphs on a dashboard.

Example/Test ProcessFlow

The model comes with a process flow (FillStorage/RequestExample) that can do two things:

- Fill the storage with a given number of items in a short time
- Place periodical requests for items in the storage

The parameters are given as labels of the ProcessFlow itself. “BoxQuantity” is the number of items that will be placed in storage. “RequestsPerMinute” controls the rate at which requests will be send to the storage once all items are stored. If “HighSpeedFill” is set to a non-zero number, the speed of the robots will be drastically increased until the storage is full, speeding up the filling process. If “StopOnFill” is set to a non-zero value, the model will automatically stop once all items are stored. The items created here are automatically assigned ID numbers ranging from 1 to the total quantity of boxes.

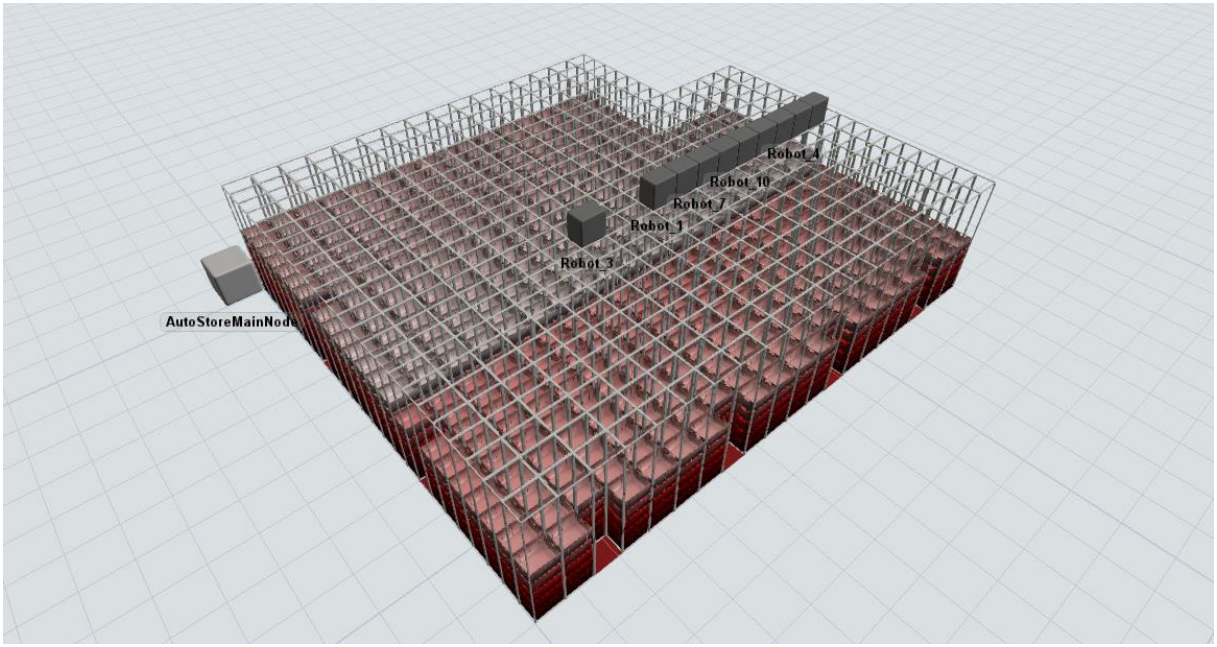

Figure 4: A storage filled with 2500 boxes

The requests are generated using an exponential distribution. Items with a higher ID (lighter color) are much more likely to be requested than once with a low number. As such the storage starts in an "optimal" state, with items with a high ID at the top of the stacks.

This ProcessFlow can be used to get a rough estimate of whether a given layout/number of robots can handle the required amount of requests by looking at the dashboard. The number of tasks waiting for a robot should be close to zero at all times, while the number of simultaneous tasks per port should not rise above what is plausible possible for the real system.