Team 57 Report: AnyLogic in Two Hours!

ISYE6644 Summer 2022

Katherine (Katy) Layton, Richie Phan

Abstract:

This project focuses on creating a simple user guide and tutorials for AnyLogic, a competing simulation software in the market with other high-level simulation languages such as Arena, AutoMod, Pro-Model, and many others. AnyLogic was selected due to the multi-method modeling capabilities using three different simulation methodologies (system dynamics, discrete event, and agent-based modeling). All three methodologies were learned over the course of a month with the goal that future readers of this report will be able to save some time and learn AnyLogic in a couple of hours rather than a few days using this guide and tutorials. This report primarily focuses on the usability of AnyLogic while contrasting it with Arena. In comparison to the base student version of Arena, AnyLogic is available on multiple operating systems, relies on the Java programming language, has three-dimensional modeling capabilities, and offers the option to run the simulation in real time as opposed to just simulated time. Overall, AnyLogic has a sharper initial learning curve than Arena. Users more familiar with process flow diagrams and less familiar with the Java programming language may favor the simplistic nature of Arena's user interface over AnyLogic's complex features. However, with enough time to dedicate to learning the software and for those more experienced with simulation principles, AnyLogic's 3-D design features and multi-method modeling offers more flexibility and superior visualizations.

Background/Problem Description:

This project focuses on prompt 11, which requires the learner to create an easy user guide, tutorials, as well as provide commentary and overall opinions on a higher-level simulation language. Several different languages were recommended for this in the prompt including AutoMod, Pro-Model, and AnyLogic which are enterprise software and even some freeware options like Python's SimPy. After reviewing several of the websites, we selected AnyLogic due to its outstanding landing page marketing, including 3D modeling capability, multi-method modeling which sounded intriguing, and the fact that many major industry players such as McDonald's, Volkswagen, and Google use this software. Over the course of a month, this team learned the AnyLogic Software using AnyLogic's "AnyLogic in three days" user guide as well as various how-to videos and content on AnyLogic's page and developed an easy user guide with tutorials so that other ISYE-6644 teams can learn AnyLogic in a couple of hours rather than three days. Of course, it may take a bit longer to be a true expert, but we believe this guide could help someone decide if it is worth pursuing learning this language alongside Arena for other graduate students with less of a time commitment.

Main Findings:

The Basics (Easy User Guide):

This guide is intended for those who have experience with at least one other simulation language or have taken a graduate/undergraduate course in simulation. This guide was written for AnyLogic Personal Learning Edition Version 8.7.12 but may still work for earlier or later versions of this software.

Step 1. Downloading AnyLogic: Navigate to https://www.anylogic.com/ in your favorite browser, select “Purchase”, enter your location. In the “Get a Quick Quote” Box, select “AnyLogic”and then “Free Personal Learning Edition”. Fill out the necessary form information and download the software. Once downloaded, double-click the .exe file and follow the prompts to install on your computer. (Note – AnyLogic is compatible with Windows, Mac, and Linux OS).

Step 2. Launch AnyLogic and create first model: Launch the software and examine the landing page, otherwise known as the *Welcome Menu*. Select *Create a Model* under *Getting Started*. Proceed with current *Model Name* and set *Model time* to “minutes”, then click *finish*. The AnyLogic user interface now displays the model building environment. There are two main tabs that will be accessed during model building, *Projects,* and *Palette*. The *Project*s Tab contains basic high-level information about the model including the model time units, how long the model should run (should it stop at a specified date/time?) and the ability to set the randomness seed for reproducible simulation runs. The *Palette* contains the building blocks for the simulation including various modeling libraries for different types of models, and subsets of blocks/shapes within those libraries for building models. For example, the *Process Modeling Library* contains a *Source* block, a *Queue* block, and a *sink* block. For those familiar with Arena, this corresponds to the “Create”, “Process”, and “Dispose” blocks. See figure 1 for the user interface and common names that will be referenced during the tutorial part of this guide.

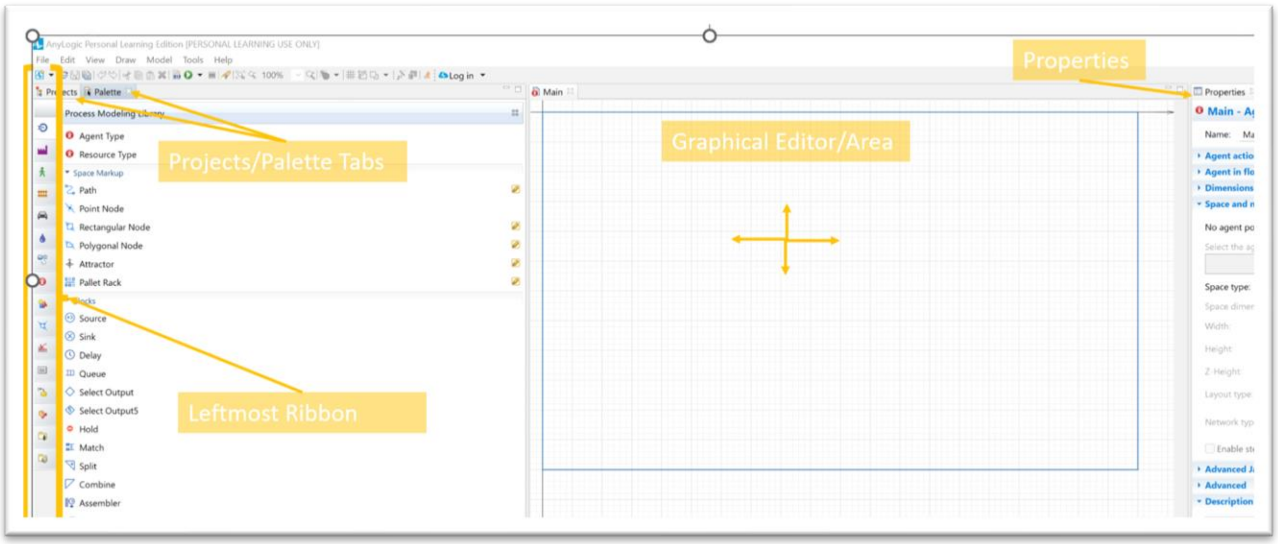


Figure 1. AnyLogic User Interface – Creating a Model.

Step 3. Begin Tutorials: Surprisingly, you are now skilled enough to enter the tutorial phase of this guide! We believe the best way to learn is by gaining hands on experience through model building. Therefore, during the next two sections, we will gain exposure to three different methods of modeling that are available in AnyLogic by creating two models. The first includes an example of a simple toy manufacturing facility highlighting discrete-event modeling. The second tutorial focuses on modeling supply and demand for these toys using the Agent-Based and System Dynamic methods in AnyLogic.

Tutorial 1: Discrete Event Modeling of a Toy Manufacturing Facility: Supplies will arrive to make toys; the raw inventory will then sit in the warehouse area while the workers process receiving paperwork. The workers on the manufacturing line will then take the raw inventory and place it inside a machine where the product is created and finally sent off for sale.

Agents are the main building blocks of the AnyLogic model. It is a unit that can have memory, behavior, timing, contacts, etc., allowing you to define events, state charts, system dynamics, flow diagrams, and events. The typical agents have attributes, interface, and behavior with the world environment. We will use agents in this case to design our flow diagram of material turning to product at our manufacturing facility.

Step 1. Create a Model – Set the model time units to “minutes” (or use previously created blank model in Step 2 of the Easy User Guide).

Step 2. Drawing out the facility – the scale of the grid is pictured in the graphical area. Hover over the left panel column with the icon images. Hover over the *presentation* icon. There you will see shapes that you can use to draw out the background for the simulation. Double click the *rectangle* and click on the *grid* to draw in a *rectangle*. See Figure 2 for reference.

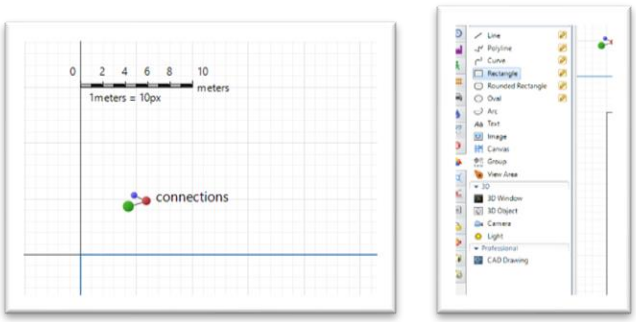


Figure 2. Grid Size (Left) and Rectangle Selection (Right).

Step 2a. You will then hover over the *space markup* tab on the left, go to the *Pedestrian* section, and double click the *Wall* text to create a wall for your facility. It should look like the image below.

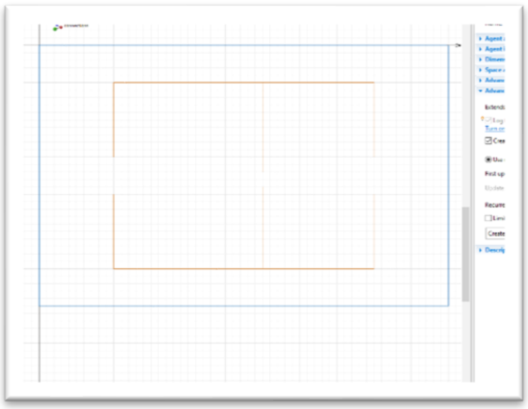
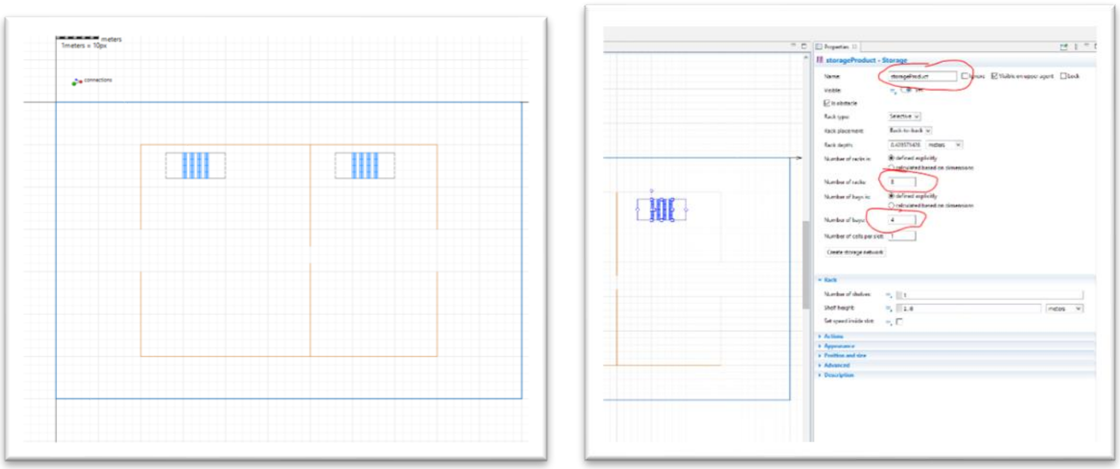


Figure 3. Facility Walls.

Step 2b. Hover over the *Material Handling* Tab, under *Space Markup*, click and drag *Storage* to the left side of the facility, placing it on the grid as shown below. Name the *storage* "storageMaterial" by double clicking the *storage* image and changing the name on the *properties* panel. Create another *storage* on the right side of the facility named "storageProduct." While still on the *Properties* panel change the number of *racks* to "8" and *bays* to "4". See Figure 4 for reference.



storageProduct - Storage

Name:	storageProduct ☐ Ignore ☑ Visible on upper agent ☐ Lock
Visible:	yes
☑ Is obstacle	
Rack type:	Selective
Rack placement:	Back-to-back
Rack depth:	0.428571428 meters
Number of racks is:	(● defined explicitly) ○ calculated based on dimensions
Number of racks:	8
Number of bays is:	(● defined explicitly) ○ calculated based on dimensions
Number of bays:	4
Number of cells per slot:	1
Create storage network	
Number of shelves:	1
Shelf height:	1.8 meters
Set speed inside slot:	☐

Figure 4. Adding Material Storage, Storage (Left) and Properties (Right).

Step 3. Creating the Network—In the *Process Modeling Library* in the *Space Markup* section, observe all the ways to draw up our network. Use the *Rectangular node* to create a *rectangular node* named "receivingDock", then double click *Path* to create a *path* from the receivingDock to our *storage* area. Of note, agents can usually move bidirectionally on a path. Double-click on the *path* or *rectangle node*, the line color should change to cyan, showing what is connected to the network. Then, add another *square node* to the network with a *path* as seen below. Name this area "forkliftParking". See Figure 5 for reference.

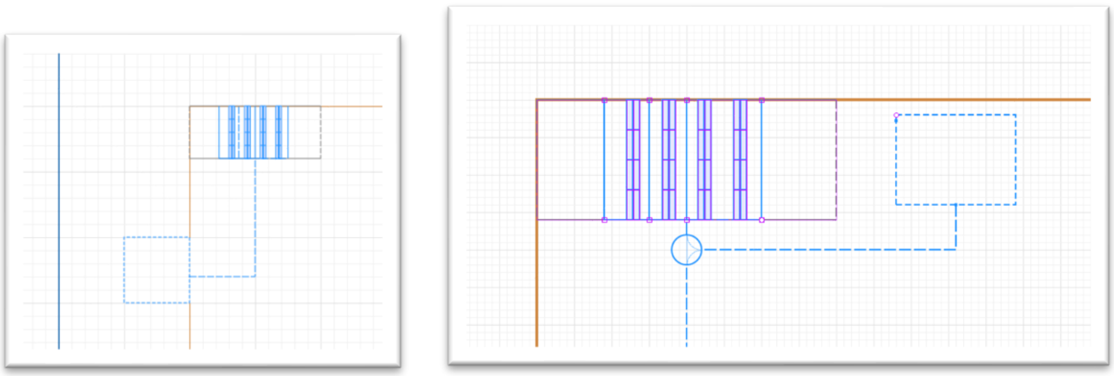


Figure 5. Network Receiving Dock (Left) and ForkliftParking (Right).

Step 3a. Next add a *rectangle node* connected to the storageProduct and add that to the rest of the network. Name this the “breakroom”. Then, go to the *Process Modeling Library* and from the *Space Markup*, drag and drop a *Point Node*, name this nodeCNC in the *properties*s menu.

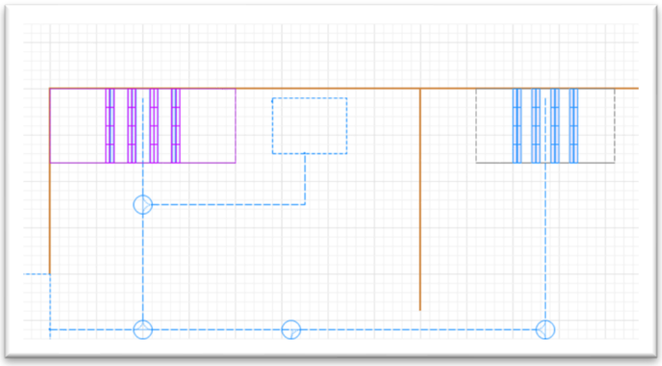
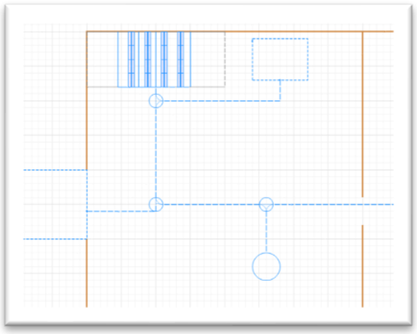
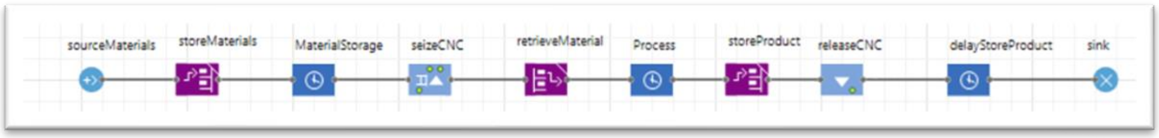



Figure 6. Breakroom (Left) and nodeCNC (Right).

Step 4. Worflow of a System – Figure 7 contains the flowchart of how a material will arrive at the facility, be processed, and then shipped out. The materials will arrive, then they will be picked up by a forklift and be stored in the storageMaterial area. There the materials will be held for a time. Then a CNC resource will be seized, and the materials will be moved to the CNC machine where it will be processed. After the product has been processed, a worker will pick up the product and store it at the storageProduct area. This is where the product will sit for a time before it is shipped and leaves the system. Before starting the flowchart, go to *Materials Handling Library*, under *Process Modeling* is *Resource Pool*. Drag **three** of these onto the *Main* grid. Name **one** “forklifts”, **one** “Workers”, and **one** “CNC”.



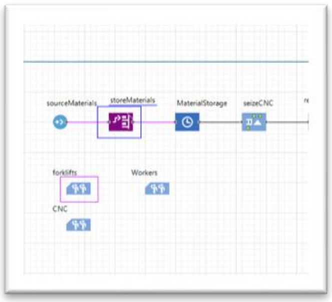
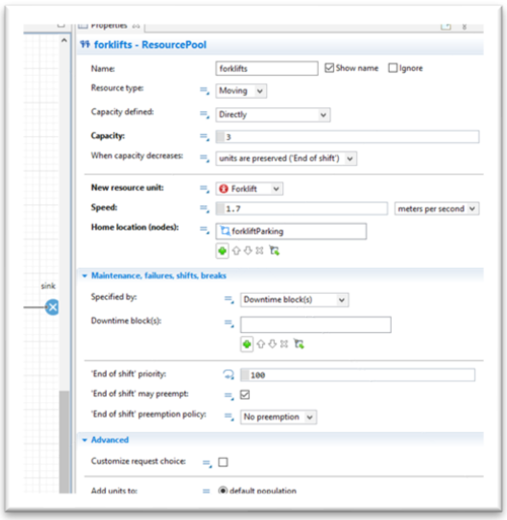


Figure 7. AnyLogic Discrete Event Flow (Material Flow Top) and Agents (Bottom).

Step 4a. For forklifts, under *properties* make sure *resource type* is *Moving*, Capacity is defined as “directly”, and *Speed* is set to “1.7 meters per second”. Set *Home Location* as “forkliftParking”. Leave all other settings as default. See Figure 7 for reference.



forklifts - ResourcePool

Name:	forklifts ☑ Show name ☐ Ignore
Resource type:	Moving
Capacity defined:	Directly
Capacity:	3
When capacity decreases:	units are preserved ('End of shift')
New resource unit:	Forklift
Speed:	1.7 meters per second
Home location (nodes):	forkliftParking

Maintenance, failures, shifts, breaks

Specified by:	Downtime block(s)
Downtime block(s):	
'End of shift' priority:	100
'End of shift' may preempt:	☑
'End of shift' preemption policy:	No preemption

Advanced

Customize request choice:	☐
Add units to:	● default population

Figure 7. Properties of Forklifts.

Step 4b. For Workers set *capacity* to "3", *speed* to "1 meter per second", and *Home location* as "breakroom". Leave all other settings as default.

Step 4c. For CNC set the *Resource type* as "Static", *Capacity* to "1", and *Home location* as "nodeCNC". In the *Process Model Library*, drag and drop a *Source* block from *Blocks* onto the *Main* grid and name it "sourceMaterials". Double-click the block to open *Properties*. Set the *arrival rate* to "3 per hour". Set the *location of arrival* as "Network / GIS node". Set the *Node* to "receivingDock".

Step 4d. Next go to the *Materials Handling Library*. Under *Blocks* drag the *Store* block onto the *Main* grid, placing it to the right of the sourceMaterials so a connection is made onto the left of the block. Name this "storeMaterials" using the *properties* window. In this window also set *Agents Move* to "by resources" and *Resource pool* to "forklifts". Set *Slotting Policy* to "according to storage" and *Storage* to "storageMaterial".

Step 4e. Drag a *delay* block from the *Process Modeling Library*. Place the block to the right of storeMaterials. Name the block "MaterialStorage". Set the *Delay time* to "triangular(15, 20, 30)".

Step 4f. Grab a *seize* block from the *Process Modeling Library* und *Material Handling* and drop it into the grid to the right of MaterialStorage. Name this "seizeCNC". Under *Resource sets*, click on the *green cross* and add "CNC*"* as a *resource*.

Step 4g. Under *Material Handling*, drag a *retrieve* block and add it to the flow chart. Name the block "retrieveMaterial". Set *Agents move* "by resources", *Resource Pool* as "Workers", *Destination* is "Seized resource unit", and *resource* as "CNC".

Step 4h. Drag another *delay* block onto the flow chart and name it "Process". Set the *delay time* to "5 minutes" with a *capacity* of "1".

Step 4i. Drag another *Store* block to the flowchart, name this "storeProduct". *Agents Move* "by resources" and the *Resource pool* is "Workers" this time. *Slotting policy* according to "storage" and set the *Storage* as "storageProduct".

Step 4j. Drag a *Release* block from *Process Modeling Library* and add it to the flow chart. Name it “releaseCNC”.

Step 4k. Add another *delay* block to the flow chart, name it “delayStoreProduct”. The *Delay time* is “triangular(15,20,30)”.

Step 4l. Finally add a *sink* from the *Process Modeling Library* to the end of the flow chart. The *sink* will dispose of the *Agent* at the end of this flow chart and simulate products leaving the system. See Figure 8 for the complete workflow of step 4.

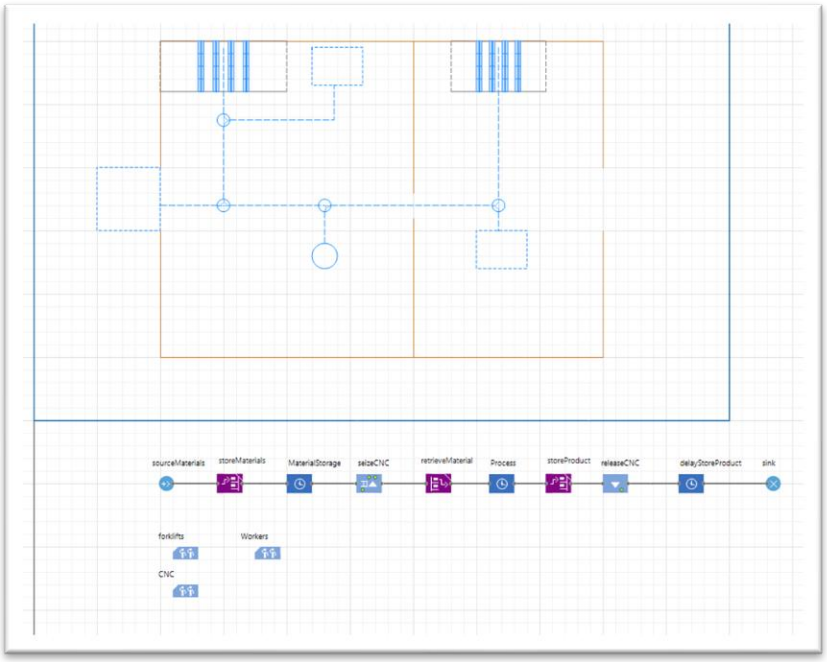
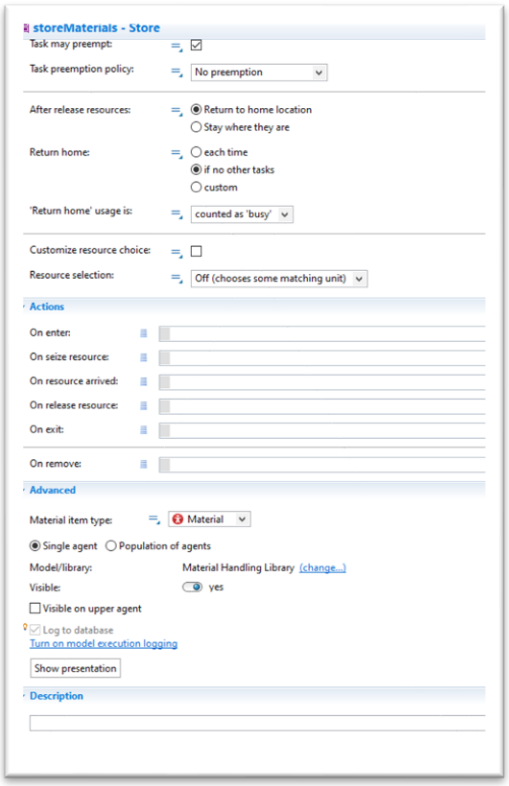


Figure 8. Complete Workflow.

Step 5. Customizing Agents with Images – For this section of the tutorial, we will add images to our *agents* to add to the visualization. First, we will give our forklifts *resource pool* the image of a *forklift*. Go to the *Process Modeling Library* and drag an *Agent Type* onto the grid. Name this “forklift” and click next. Under *Warehouses and Container Terminals*, select “Forklift Truck” and click finish. This will create a new agent tab with your image at the center of your grid. Here you can manipulate the image of the *agent*.

Step 5a. Click on the *forklifts Resource Pool.* Under *Properties*, make new *resource unit* “Forklift”. Do the same for the other *Resource Pools*, browse for images that you would like to be the *Workers* and *CNCmachine* by dragging *Agent Types* to the *Main* Grid.

Step 5b. Now create an *Agent Type* named “Material”, find an image of a box under the *Box* tab. Click on the *source* block on the *main* grid to open the *properties*. Set the new *agent* to “Material” under the *agent* tab. On the *Advanced* tab set the *Agent type* to “Material”. Ensure Material is set on *Material Item Type* as “Material” under the *Advanced* tab in *properties* for each of the following blocks.



storeMaterials - Store

Task may preempt: ☑

Task preemption policy: No preemption

After release resources: [✓ Return to home location] Stay where they are

Return home: each time [✓ if no other tasks] custom

'Return home' usage is: counted as 'busy'

Customize resource choice: ☐

Resource selection: Off (chooses some matching unit)

Actions

On enter:

On seize resource:

On resource arrived:

On release resource:

On exit:

On remove:

Advanced

Material item type: Material

[✓ Single agent] Population of agents

Model/library: Material Handling Library (change...)

Visible: yes

☐ Visible on upper agent

☑ Log to database

Turn on model execution logging

Show presentation

Description

Figure 9. Set Material Item Type to Material for these blocks

Step 5c. Click the green *play* sign to start the simulation.

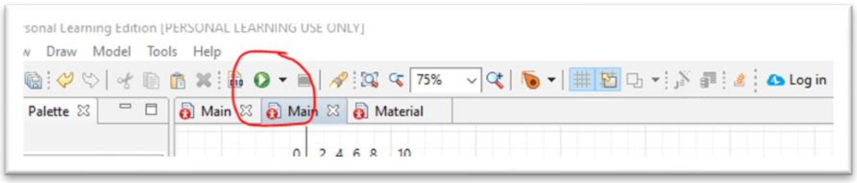


Figure 10. Play the Simulation.

Step 5d. A new window will open with the simulation running, going in real time. Use the controls at the bottom to manipulate the simulation speed. Hover over to each button to see what it does.

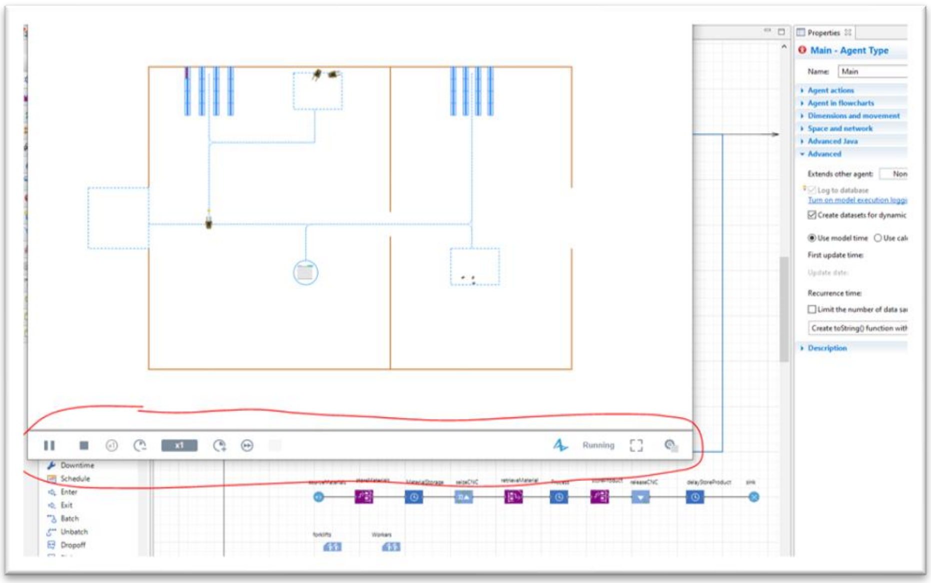


Figure 11. Simulation Window and Controls.

Tutorial 2. Agent-Based and System Dynamics Modeling of Supply and Demand: Suppose you are the new Data Science Intern for the toy factory that we modeled previously, and you are trying to understand the demand for the latest toy your company is about to launch. The toy will be provided to some influencers to market with the hope that it will reach some percentage of the population of potential buyers, and each buyer will love it and tell their friends about it turning into more demand. You also want to make sure that production is considering this demand, that delivery time is sufficient, and that the consumer store has inventory to sell to meet market demand.

Step 1. Create Model: create model using the same steps in the easy guide and name the model whatever you like (e.g., Supply/Demand). Change the *model time* to "days".

Step 2. Begin Building Agent-Based Model: In the *Palette* Tab, select *Agent* from the list of options by hovering your mouse on the left-most ribbon. Drag and drop an *Agent* into the *Main* graphical area. Select "Population of Agents", then name your agent "Person" (Note: *Agent population* automatically provides "people" as the plural version). Leave the radial option as "Create the agent type from scratch" and click *Next*. Choose "2D" and "Person", followed by *Next*. Skip the parameters for now and click *Next.* Enter "2000" as the *population of Agents* (Warning: anything above this may or may not work in the student free edition and you will receive a "Java.nullPointerException" error for trying which can be difficult to troubleshoot)**.** Finally, click finish. See Figure 12 for reference of the *Main* view so far.

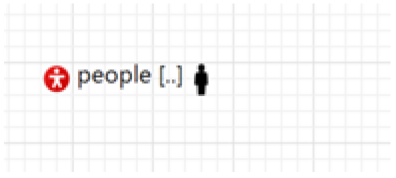
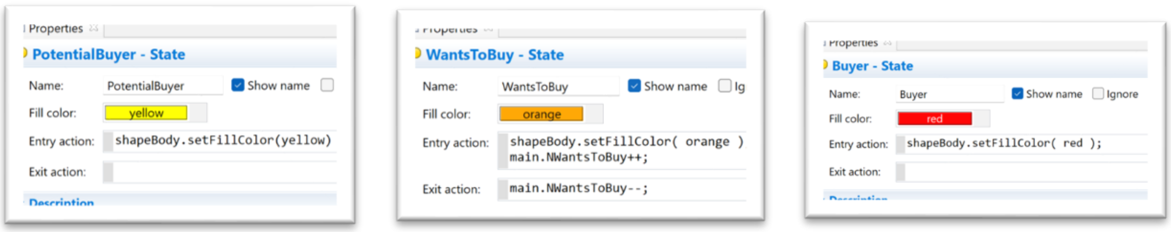


Figure 12. Population of Agents

Step 2a. In the *Projects* tab, double-click on *Person* to open a new graphical area tab called "Person". First, we will focus on the marketing campaign to generate demand for this product. Switch to the *Palette* tab and navigate to the *Statechart* on the leftmost ribbon. Drag and drop *Statechart Entry Point* from the inner left ribbon to the graphical area for *Person*. Then, drag *State* and connect it by aligning the arrow and shape, finally releasing it to attach. (Note: if it did not align properly it will turn red). Now, drag a second *State* block onto the graphical area a short distance below the previous *State* block but do not attach it. Finally, add one more *State* block a short distance below the second (Note: these are free floating for now).

Step 2b. Click on the first *State* block and change the *Name*, *Fill Color*, and *Entry action* in the *Properties* tab per the image below. Complete the other two *State* blocks per the next two images. Type the *Entry action* as is for now and we will explain what this is doing a bit later.


Properties

PotentialBuyer - State

Name: PotentialBuyer [✓ Show name]

Fill color: yellow

Entry action:
```
shapeBody.setFillColor(yellow)
```
Exit action:

Description

Properties

WantsToBuy - State

Name: WantsToBuy [✓ Show name] Ig

Fill color: orange

Entry action:
```
shapeBody.setFillColor( orange )
main.NWantsToBuy++;
```
Exit action:
```
main.NWantsToBuy--;
```

Properties

Buyer - State

Name: Buyer [✓ Show name] Ignore

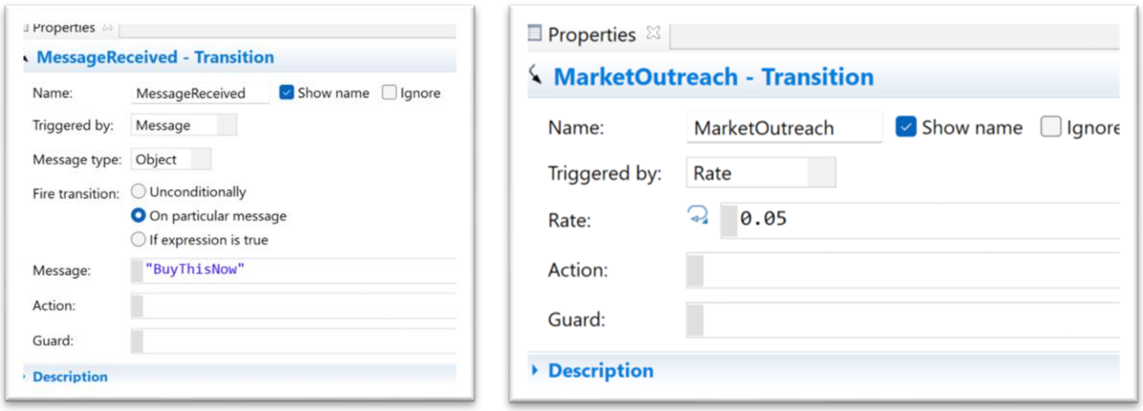
Fill color: red

Entry action:
```
shapeBody.setFillColor( red );
```
Exit action:

Description

Figure 13. Properties of first block (left) second block (middle) and third block (right).

Step 2c. Now, add the *transitions* between blocks. We have 4 *transitions* to add, 1 for the influencer's market outreach campaign, another for the people who bought the product and loved it so much they told their friends, 1 to signify that the product was purchased, and an *internal transition* to generate the message for the buyers to share so their friends will buy it. Double-click on *Transition* and click the yellow *State* block then then orange *State* block (You should see a line with an arrow extending from the yellow to the orange block). Do this one more time on the same blocks. On the left arrow, enter the properties from the below image in Figure 14. This transition represents the *trigger* to make a *Potential Buyer* buy if they receive the message "BuyThisNow". Click the right arrow and add properties per the below right image in the same Figure 14. This represents the influencer hired to reach out to 0.05% of the population of *Potential Buyers* to get them to buy.


Properties

MessageReceived - Transition

Name: MessageReceived [✓ Show name] Ignore

Triggered by: Message

Message type: Object

Fire transition: Unconditionally [✓ On particular message] If expression is true

Message:
```
"BuyThisNow"
```
Action:

Guard:

Description

Properties

MarketOutreach - Transition

Name: MarketOutreach [✓ Show name] Ignore

Triggered by: Rate

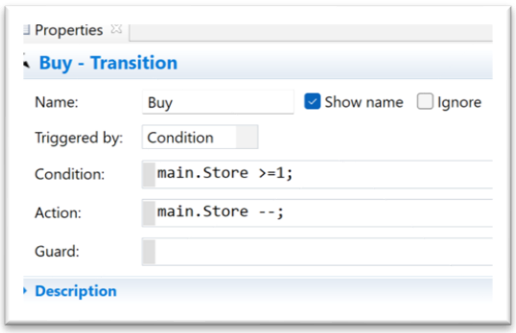
Rate:
```
0.05
```
Action:

Guard:

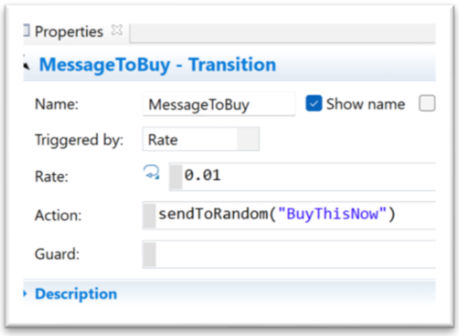
Description

Figure 14. Properties of Messaging a Friend (left) and MarketOutreach Transition (right).

Step 2d. Now add a *Transition* between the orange and the red blocks, and then enter the following information in Figure 15. This creates a *transition* for decreasing the *Store* inventory (Note: We will create *Store* in our System Dynamics portion). The code "main.Store >=1" ensures that the Store block on the Main tab (Main graphical area) has at least 1 item and then decreases that item indicating stock was purchased. Finally, drag and drop the *Transition* inside of the red block (Note, the arrow must be connected to the inside 2 walls of the red block). Enter the properties in the right image in Figure 15. This creates an *Action* where the *Buyer* sends a *message* to a random *Potential Buyer* to "BuyThisNow" because they are obsessed with the new toy. We set the rate to reach .01% of the *Potential Buyers*. The state chart is now completed.



Name:	Buy ☑ Show name ☐ Ignore
Triggered by:	Condition
Condition:	`main.Store >=1;`
Action:	`main.Store --;`
Guard:	



Name:	MessageToBuy ☑ Show name ☐
Triggered by:	Rate
Rate:	`0.01`
Action:	`sendToRandom("BuyThisNow")`
Guard:	

Figure 15. Buy Transition (left) and Internal Transition (right).

Note: We have now finished modeling the Agent-Based method population behavior to the influencer and word of mouth communications used to generate demand for the factory's new toy. What we have done so far is create a *Statechart* that shows the Agent-Based modeling. The population consists of 2000 *Potential Buyers*, who are exposed to both an influencer's marketing tactics and pressured to buy after being bombarded by love of the product from some of their peers. Once they receive word, they likely will want to buy and enter the "WantsToBuy" state and then, eventually they buy the product. See Figure 16 for what the entire model should look like up to this point.

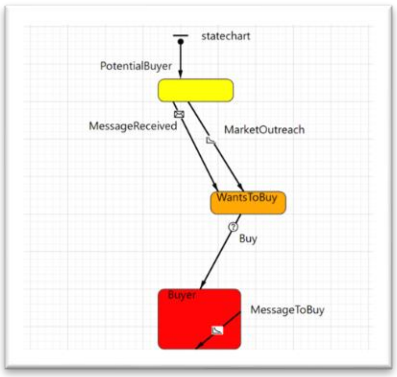
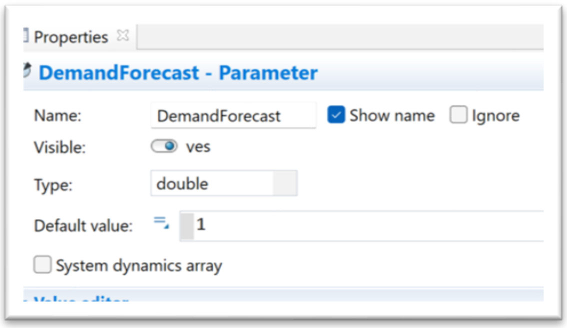


Figure 16. StateChart Agent Based Method.

Step 3. Begin Building System Dynamics Model: Return to the *Main* tab in the graphical area, and we will start the System Dynamics portion of the model. In the *Palette* tab on the left window, In the leftmost ribbon, select *System Dynamics*, then drag and drop a *flow* into the graphical area of *Main.* Drag a *Stock* into the graphical area and attach to the arrow portion of the *flow* (Note: you can drag the *stock* block around and if the *flow* does not stay attached, you will need to drag the arrow of the *flow* onto the *stock* block). Add another *flow* and another *stock*, ensuring the entire thing is connected by testing dragging the pieces. Drag and drop a *Parameter* under the first *flow*. Then, click the *parameter* and enter the following to initialize the forecast to 1.


Properties

DemandForecast - Parameter

Name: DemandForecast ☑ Show name ☐ Ignore

Visible: yes

Type: double

Default value: 1

☐ System dynamics array

Figure 17. DemandForecast Parameter.

Step 3a. Double-click *link* and then click "DemandForecast"(Parameter you just renamed) and then click the hourglass portion of the first *flow*. This links the *parameter* and *flow* together.

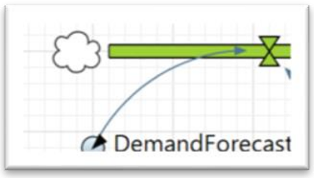
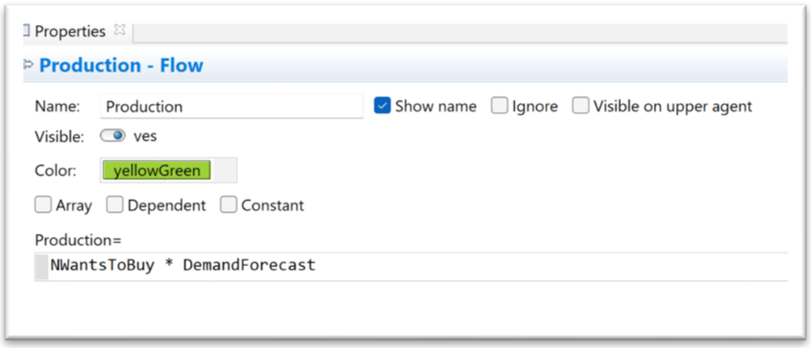


Figure 18. DemandForecast Parameter linked to first *flow*.

Step 3b. drag and drop a second *parameter* under the second *flow.* Click the *Parameter* and update properties with *Name* "DeliveryPeriod", *Type* "Time", *Unit* "days", and *Default value* "3". Here we are setting the Delivery Period (time it takes to ship product from Finished Goods to Store) to 3 days once a product reaches finished goods. Now, double-click *link* and connect "DeliveryPeriod" to the hourglass symbol on the second *flow*.

Step 3c. Click the first flow and enter the info in figure 19 to indicate that *Production* is the product of the *variable* "NWantsToBuy" and the *parameter* "DemandForecast". Click the first stock (square block we added earlier), enter *Name* "FinishedGoods", *Color* "Yellow", and *Initial Value* "0". Click the second *flow* and enter *Name* "Shipping", *Color* "dodgerBlue" and equation Shipping="FinishedGoods/DeliveryPeriod". Then, double-click *link* and connect the *Finished Goods Stock* block to the *Shipping* hourglass *flow* in the graphics (Note: The equation completes itself here once you attach these and will change from just "Production" to "Production-Shipping" to show that they are now linked. This *Stock* represents the number of toys finished by production that are ready for shipping). Finally, click the last *stock* and enter the following properties: *Name* "Store", *Color* "Cyan", *Initial Value*

"150". This sets the initial store stock to 150 toys and defines the retail end of the supply chain as "Store". Recall earlier that we defined an action to deplete this stock in the Agent-Based portion of this model.



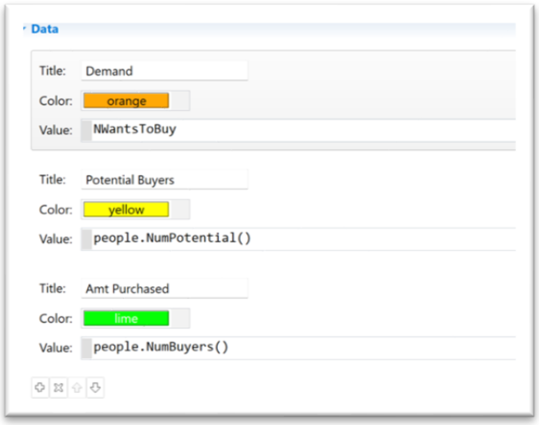
Name: Production
[✓ Show name] Ignore Visible on upper agent
Visible: yes
Color: yellowGreen
Array Dependent Constant
Production=

```
NWantsToBuy * DemandForecast
```

Figure 19. Production *Flow* Properties.

Step 3d. Finally, add the *variable* we referenced earlier. Drag and drop the *Dynamic Variable* into the *Main* graphical area and link it to the *Production* hourglass portion of the *flow*. Rename it to "NWantsToBuy" and change the *type* to "Int" if needed. Recall that earlier in the Agent-Based Modeling portion, we created an *Entry-Action* to update this variable and an *Exit Action* to update the *variable* when a *Potential Buyer* changes from "WantsToBuy" to "Buy".

Step 4. Adding Charts: Lastly, we will want to monitor the demand, potential buyers, and amount purchased in our simulation. A straightforward way to do this is to add in a *bar chart.* In the leftmost ribbon, select *Analysis* and then drag and drop *Bar Chart* into the *Main* graphical area. Name the chart "Demand", then update the following in the *Data* Section of properties (NOTE: Feel free to play around with the appearance and legend as well – not shown below).



Title:	Demand
Color:	orange
Value:	NWantsToBuy
Title:	Potential Buyers
Color:	yellow
Value:	people.NumPotential()
Title:	Amt Purchased
Color:	lime
Value:	people.NumBuyers()

Figure 20. Creating a Bar Chart.

Step 5. Run the model: Compare your *Main* graphical area to the final flowchart we got below. Now that we have finished, let us review the entire System Dynamics portion. Here we have generated our supply chain to meet the demand. *Production* starts, *Finished Goods* inventory builds up, then enters *Shipping* with a delivery period of ~3 days and product reaches the *Store* where the *People* we defined earlier can now buy the product. Click *Save*, then *Build*, then *Run* to execute the model as we did in the first tutorial.

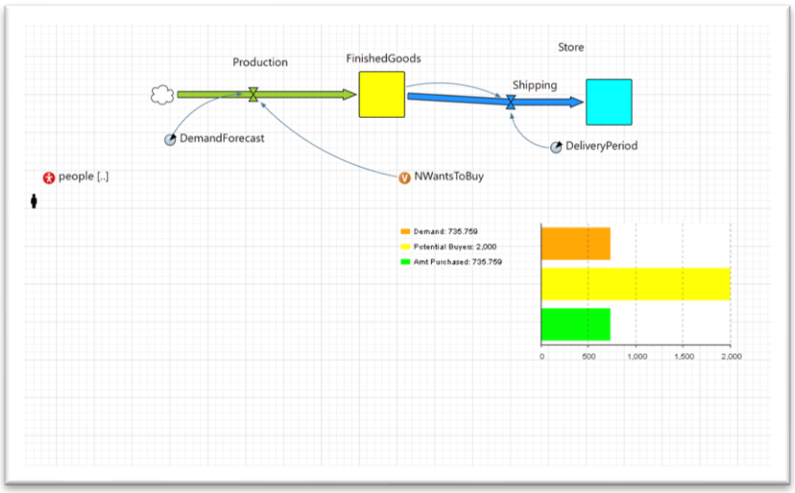


Figure 21. System Dynamics Model Flowchart in *Main*.

Step 6. Observe Simulation: The output below reflects the changing demand, potential buyers, and amount purchased as well as the status throughout the supply chain! The colors we selected correspond to the different states (Note: the population diagram on the bottom left reflects the change in state to green as *People* buy the product). If any errors arise, first ensure that all blocks are connected and there are no free-floating graphics in the supply chain flow or *Statechart*. Then, run the model again.

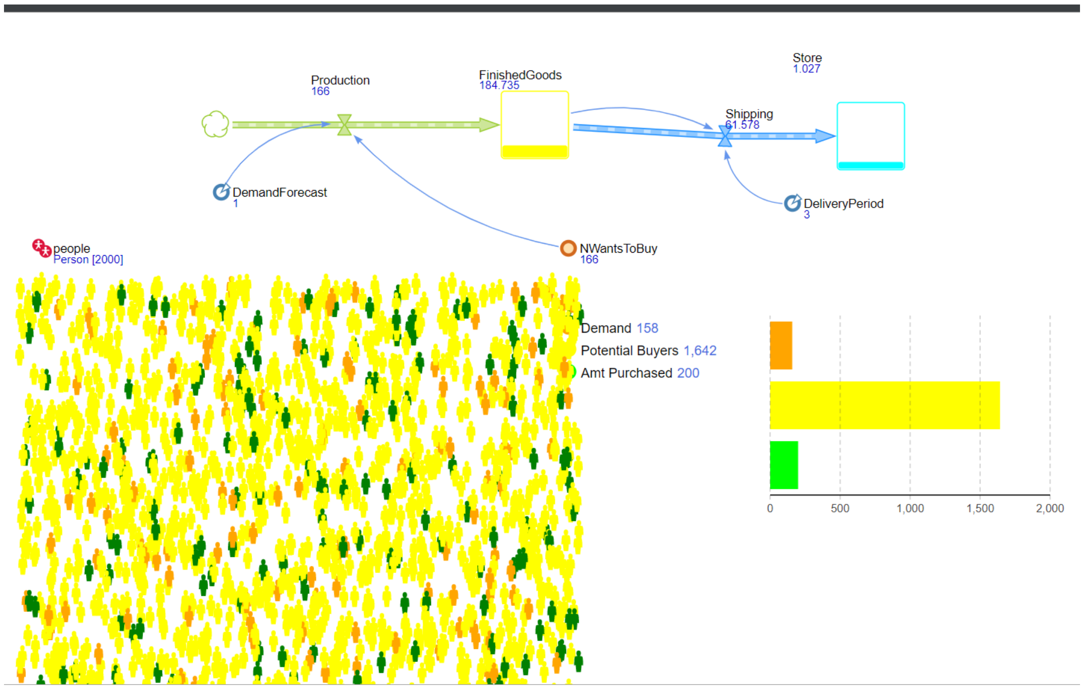


Figure 22. Simulation Output.

The Good, the Bad, and Comparisons to Arena:

Our favorite qualities of AnyLogic are the abundance of training available, the robust design features, and accessibility of multiple projects in the *Project* pane view. As soon as you login, the *Welcome* page provides specific examples to work on and links the user to multiple guides, while in Arena the user must search for the guides. There is an array of design features that allow you to customize your agents and design your model in three dimensions, whereas the base student version of Arena does not offer this. Arena has the modules pane on the lefthand side with the various templates (e.g. basic process template), while AnyLogic has the "Palette"(e.g. Process Modeling Library. In both, you drag the individual blocks you want to use to the graphical editor. In Arena, when you click on the block, the properties come up in a pop-up window. In AnyLogic, the properties pane is already on the right-hand side and changes based on what you have selected. Both allow for multiple models to be open in the graphical area, with each model featured in a different tab. However, in AnyLogic, you can also access each model in the "project folder" which has a nice overview of all the different models you have created. When you run a model in AnyLogic, it opens in a separate window as opposed to everything happening in the same window within Arena. In Arena, the simulation time does not follow real time. In AnyLogic this is also true, however you have the option to run the model in both real time and simulated time. In AnyLogic, the user can easily track multiple projects on the left window and see the different project hierarchies. Arena keeps it simpler and only allows you to switch between tabs. Our least favorite qualities of AnyLogic include the overall complexity of the initial user interface and too many advanced options that could be overwhelming to a novice user. In contrast, Arena starts the user off with basic options and you can add more advanced templates as needed. Overall, AnyLogic is attractive software used by many giant corporations and has some unique, useful features. However, the learning curve is higher for this software over something like Arena. Arena's simplistic use of simple flowchart style processes is more intuitive for someone with training in manufacturing process flow diagrams and is a better starter software. AnyLogic may come more naturally for someone with a Java background (since a lot of the nomenclature, commands, and structure seems to be based on the Java programming language). Regardless, once enough time is dedicated to AnyLogic, it would become the clear winner for any company with the means to purchase software due to its diverse multi-method modeling feature.

Conclusions/Future Ideas:

This team thoroughly enjoyed learning a new simulation language and learning about different simulation methods aside from discrete event modeling that we focused on with Arena. The models we provided in the tutorial are extremely simple models that give the user a world tour of AnyLogic and the three different methods of modeling: Agent-Based, System Dynamics, and Discrete Event. The recommended next steps for anyone looking to learn AnyLogic after reading this report, are to first read the AnyLogic user guide, "AnyLogic in three days" which has far more detailed examples and provides a tour of many more features. For the more adventurous, our tutorial models could be expanded to include a competing product line, where parts are manufactured on a separate line in production. We could also add another area to our S*tatechart* in the Agent-Based model to determine the separate demand for each competing product. Additionally, the parameters in the model were selected for demonstration purposes and were not representative of a real production or supply chain model. More work could be done to bring these examples in line with something closer to reality.

References:

AnyLogic in three days:

The AnyLogic Company. (2018). AnyLogic in 3 days [Online].

The Job Shop:

“Tutorials.” *AnyLogic Help*, Anylogic Company, https://anylogic.help/tutorials/index.html.