Project 5

Added by Christopher Hoder, last edited by Christopher Hoder on Oct 23, 2011

The main task of this project was to use the Queue data structure to simulate and compare different strategies for selecting checkout lines (queues) at a store. With this simulation we can directly compare different choices and determine which strategy works the best. In this simulate there are 3 different strategies. The first is to randomly pick a line and join it. The second is to pick the shortest line of them all. The third is to pick the best of two randomly selected lines. Each of these strategies corresponds to different color of shopper on the landscape. The colors are blue, yellow and red respectively. There is also another mode, which will have the shoppers randomly choose one of the 3 strategies so we can compare the different strategies within the same simulation.

In my solution, I reused a lot of code from previous week simulations such as the Landscape class and the SimObject class. The shopper class will take care of selecting a queue, waiting the appropriate time, and storing data such as how many items it has left to queue, how long it has been in the store and if it has paid or not. The CheckoutQueue class will scan the items, and ask the shopper to pay and remove them from the store.

Additionally, all of the shoppers who have left the store are not deleted but instead stored in an ArrayList within the Landscape class. This allows us to compute statistics on the simulation. The minimum and maximum time spent in the store as well as the average it takes the shopper to exit are calculated each turn and printed to the terminal. As part of the extension that will run different strategies within the simulation, the average time in the store for each individual strategy is also determined and printed out to the terminal after each iteration.

Another extension that I undertook in this project was to make the display a bit more interesting. I made it so that each queue was labeled with a number and this displays above the counter. Additionally the transparency of each shopper depends on how many items the shopper has between 0 and the max number of items the class can have.

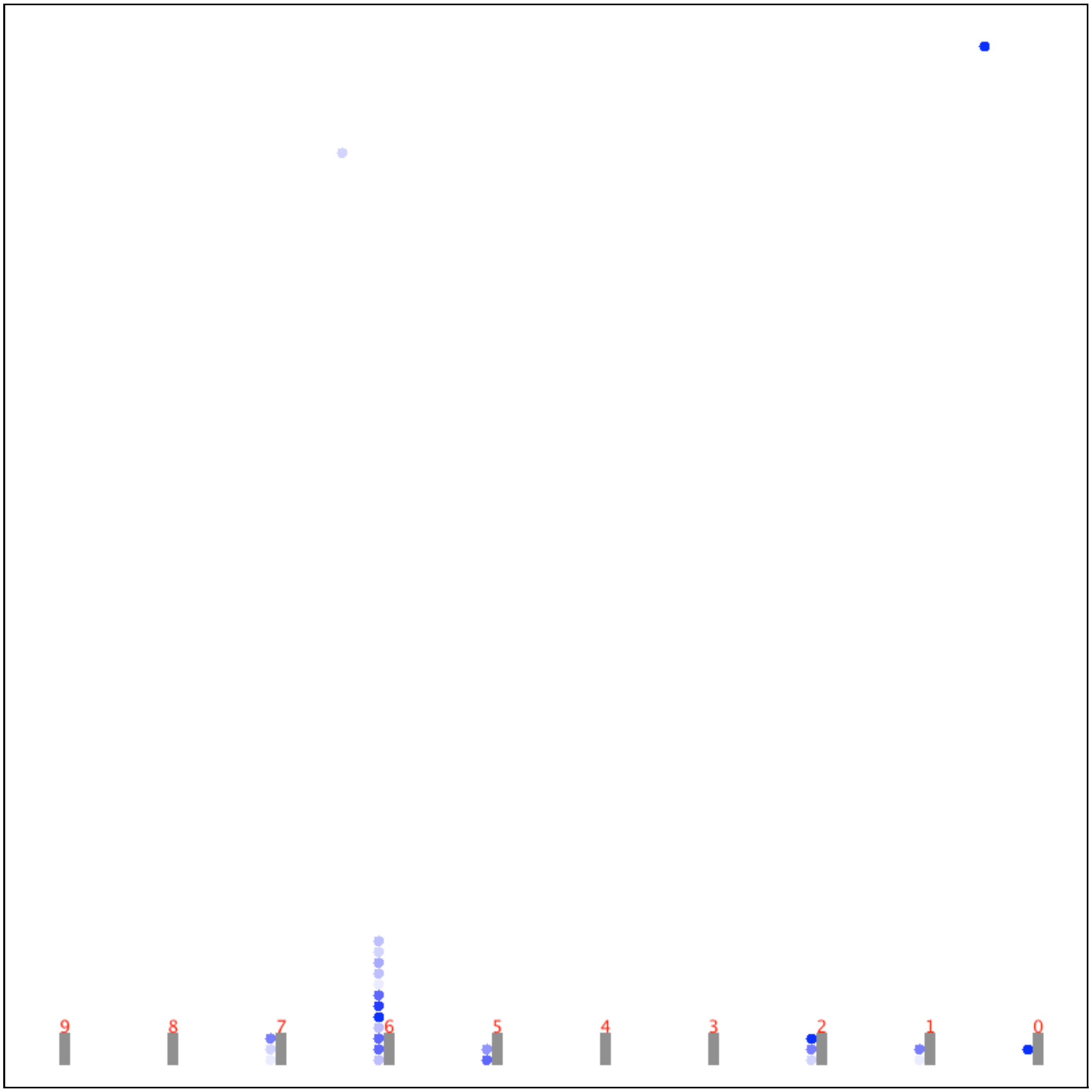
Additionally I created a new simulation mode (gameType = 2) that will simulate the single queue checkout line like the one at Hannaford. This mode will have all of the shoppers queue in a single queue on the left of the landscape. This queue then determines which lines are empty and puts shoppers there.

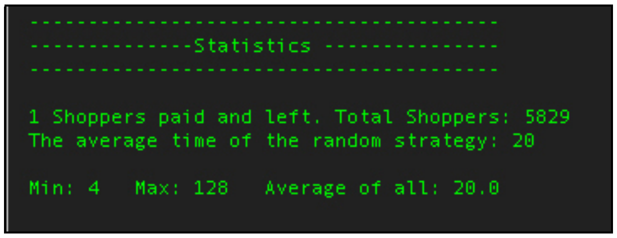
Below are some examples of my solution. Like the last project i used the jargs package for help with command line parsing. This allows me to have a customizable commandline program. The syntax for the program is as follows:

```
java CheckoutSimulation [-g,--gameType] [-w,--width] [-h,--height] [-s,--strategy] [-q,--queues] [-p,--pauseTime]
                        [-c,--SpawnCreate] [-i,--itemsMax] [-t,--shopperStart]
```

Where gameType is how you switch between the two different types of simulation, 1 means the 'classical' version of the checkouts with the shoppers choosing the queue they want to go into. 2 means the single queue set up. width and height set the landscape size. strategy is how you change which strategy is being used. 0 means the shopper will pick a random line. 1 means the shopper will pick the best line of them all. 2 means the shopper will select the best of 2 randomly chosen lines. 3 means that the shopper will randomly pick one of the strategies previously mentioned. The queues sets the number of queues on the landscape. pauseTime is the time between iterations. SpawnCreate is how many shoppers should be created with each iteration. itemsMax is the maximum number of items a shopper may start with. shopperStart sets the number of shoppers in the store at the start.
Using my code i ran several large simulations of several thousand customers. Below are some graphics from the various simulations:

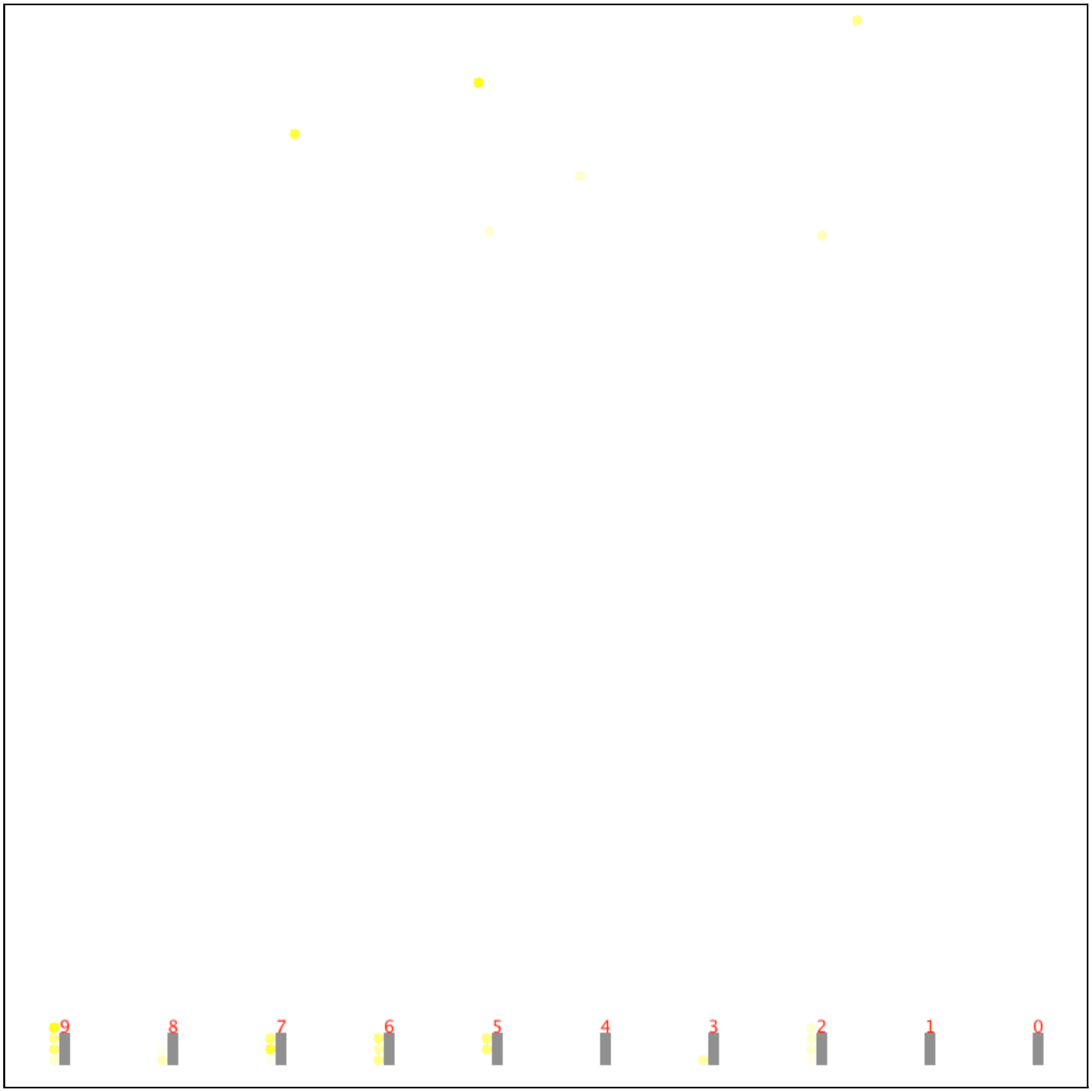
The first is a simulation of shoppers picking random queues. The statistics are also included on the last iteration below.

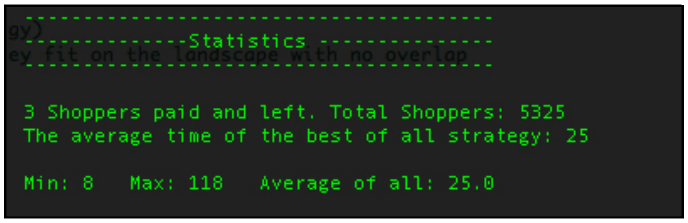




```
----------------------------------------
--------------Statistics ---------------
----------------------------------------

1 Shoppers paid and left. Total Shoppers: 5829
The average time of the random strategy: 20

Min: 4   Max: 128   Average of all: 20.0
```

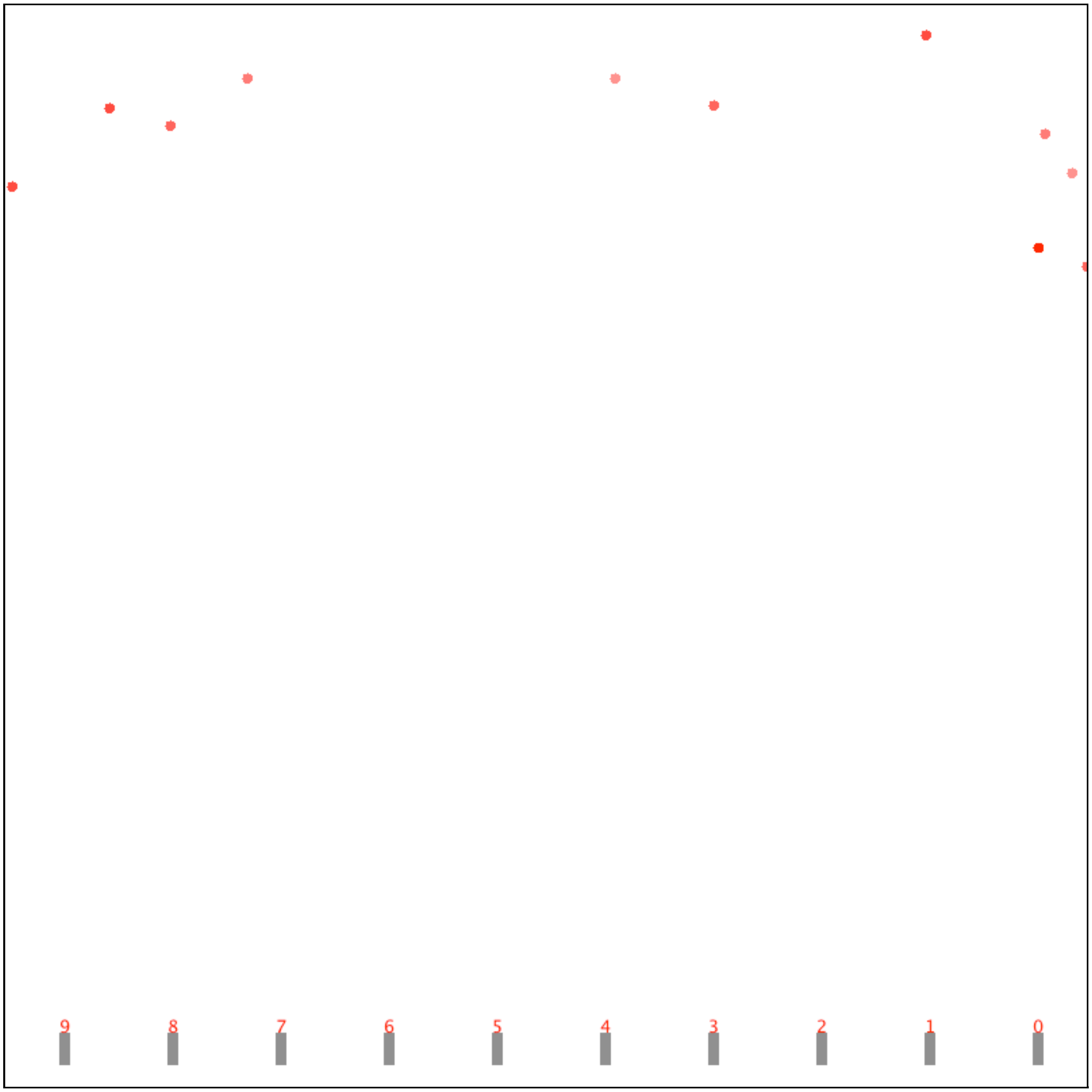
The next is a simulation of the shopper picking the best queue of them all. The statistics are also printed out below

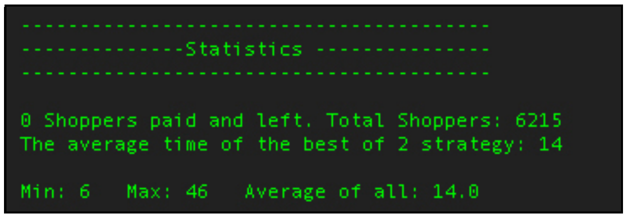




```
 ----------------------------------------
 --------------Statistics ---------------
 ----------------------------------------

 3 Shoppers paid and left. Total Shoppers: 5325
 The average time of the best of all strategy: 25

 Min: 8   Max: 118   Average of all: 25.0
```

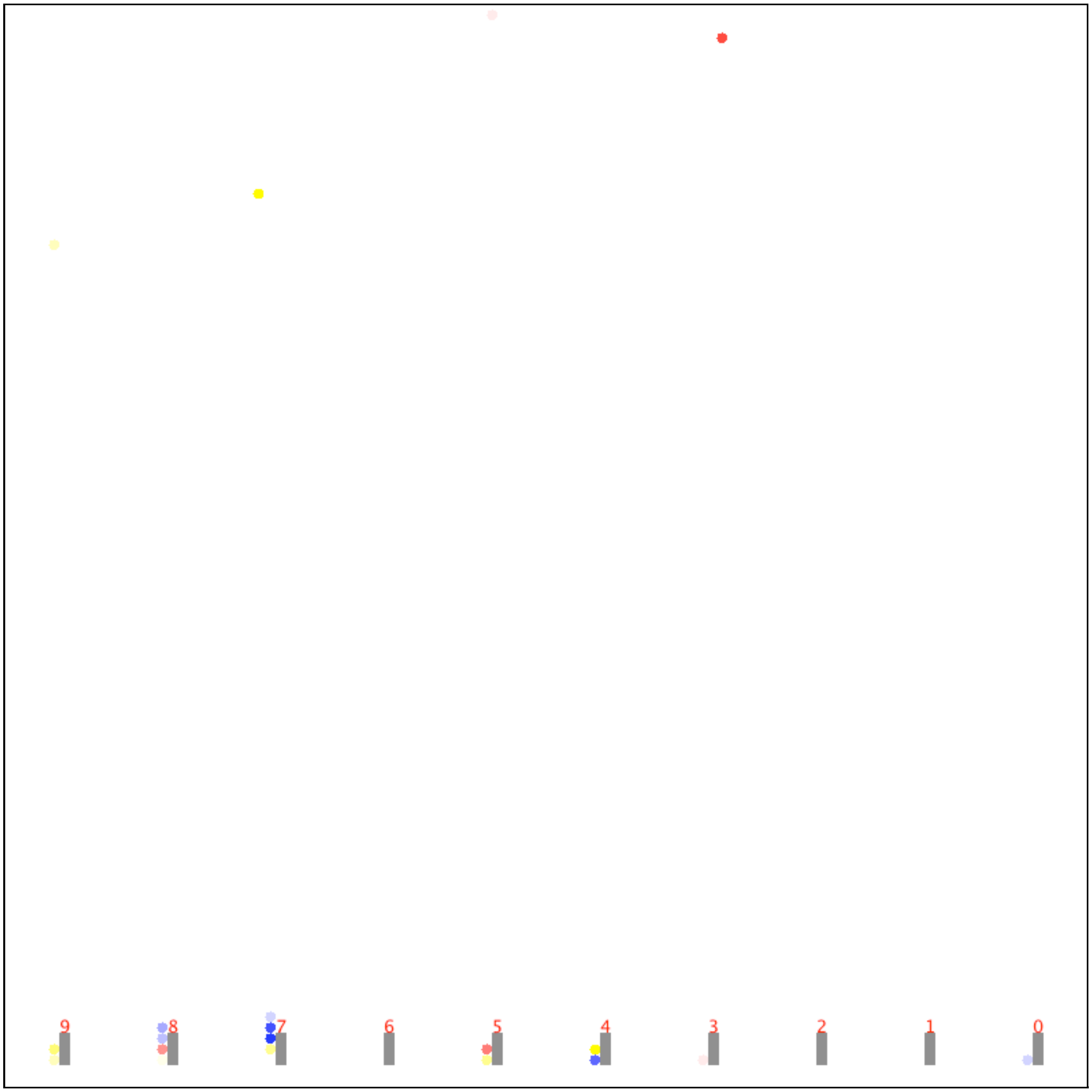
The next one is the best of 2 randomly selected queues. The statistics are below also:

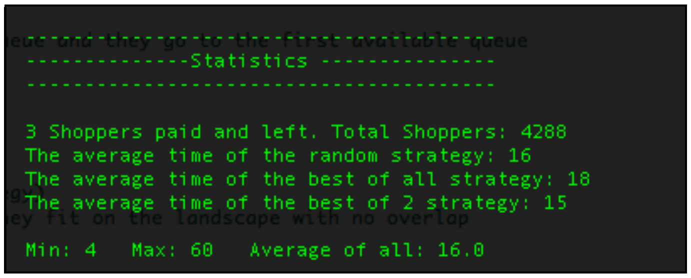




```
----------------------------------------
--------------Statistics ---------------
----------------------------------------

0 Shoppers paid and left. Total Shoppers: 6215
The average time of the best of 2 strategy: 14

Min: 6   Max: 46   Average of all: 14.0
```

From these 3 simulations we can see that the best strategy is to pick two random lines and choose the best one of those two. It is better on average by almost 5 turns. We can also run a simulation where all of these strategies are being run on the same simulation, as shown below:

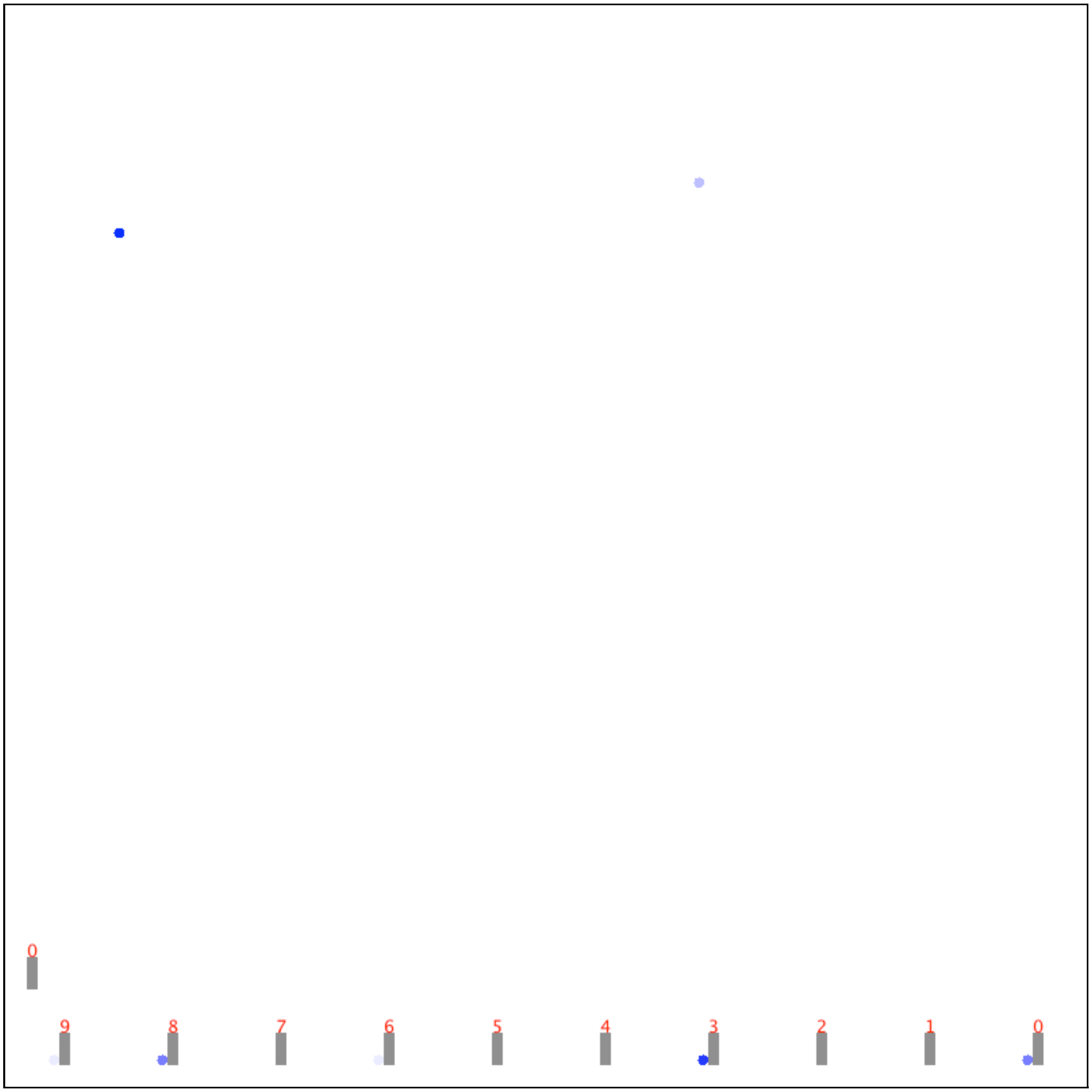


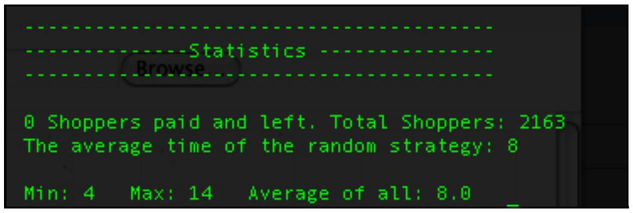


```
 ----------------------------------------
 -------------Statistics ---------------
 ----------------------------------------

 3 Shoppers paid and left. Total Shoppers: 4288
 The average time of the random strategy: 16
 The average time of the best of all strategy: 18
 The average time of the best of 2 strategy: 15

 Min: 4   Max: 60   Average of all: 16.0
```

In this mixed strategy simulation we see that the picking the best of 2 strategy is still better but not by as much as in the individual simulations.

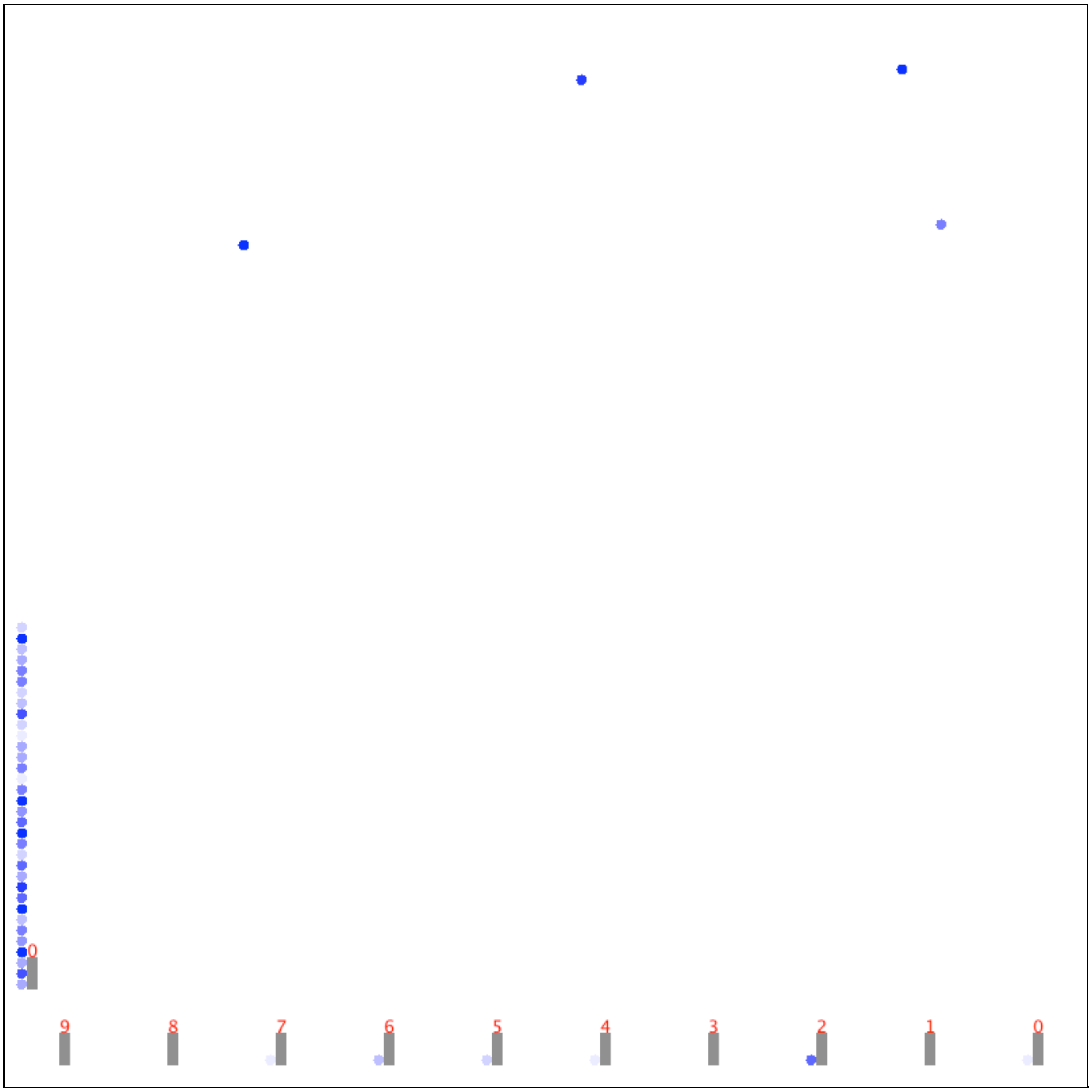
I also created a different simulation mode that has all of the shoppers queue in one line and then get placed in the open checkout counter. A gif of this simulation is below with the statistics printed out below it:





```
-----------------------------------------
---------------Statistics ---------------
-----------------------------------------

0 Shoppers paid and left. Total Shoppers: 2163
The average time of the random strategy: 8

Min: 4   Max: 14   Average of all: 8.0  _
```

We can see that this approach to checkout lines is way more efficient, with shoppers taking on average half as long to checkout. Since this method has so few shoppers it is hard to see how it is working because they never wait in the queue. Below is the same simulation setup but instead there are 2 shoppers spawned with each iteration.



We can see that this quickly becomes unstable but it does show how it is working.

In conclusion this project allowed me to really grasp the type of uses a queue has and apply it to an interesting question about which strategies are the best when it comes to selecting a checkout line. Through running this simulation, I was able to see that picking the best of 2 random lines is the best strategy when it comes to selecting a line. Additionally we can see that a stores should implement the single queue system like Hannaford because it is way more efficient at getting customers out of the store. We were also able to also continue to reuse the classes we defined earlier such as the SimObject class.

Labels

cs231f11project5

Printed by Atlassian Confluence 3.3.1, the Enterprise Wiki.