 ***MSIM 603: Simulation Design***

***Department of Modeling, Simulation and Visualization Engineering***

***Old Dominion University***

**Programming Assignment Six**

March 27, 2017

Mark Brosche

Major: MSVE - Master of Engineering

Online Distance Graduate - Outside Virginia (Saratoga Springs, NY)

[mbros006@odu.edu](mailto:mbros006@odu.edu)

407-325-2818

**A*SSIGNMENT:***

Your assignment is to develop a discrete event simulation of an elevator system. The system consists of one elevator. Service calls (events) for the elevator occur with a Poisson distribution having a mean of 2 and a delay for elevator travel and discharge of passengers (event) with an exponential distribution of 1. There is no need to model each floor. The travel and discharge distribution will take care of the variance of travel times representing each floor. The initial event for the system is a discharge event at event time 0.0. The discharge event is scheduled immediately after the elevator travel event. The next travel event is triggered by the execution of the discharge event. If it is event based, you must provide an event graph diagram, properly annotated, of your system. For your event set management, you may use the simulation engine we developed as part of our previous programming assignments. This approach assumes a linked event list. You should analyze the performance of your system per the criteria we set up in programming assignment five. This analysis should be documented in the results section of your report. Your simulation should be run with a total run time of 500.0 time units.

Submit the following as part of this assignment:

1. A report with the following sections
   1. Introduction to the purpose of your simulation
   2. Your simulation methodology
   3. Simulation results
   4. Simulation analysis
2. Your Visual C++ project files
3. Any data files generated as part of your simulation

**Grading:**

Design: 20 pts

Implementation: 30 pts

Results: 20 pts

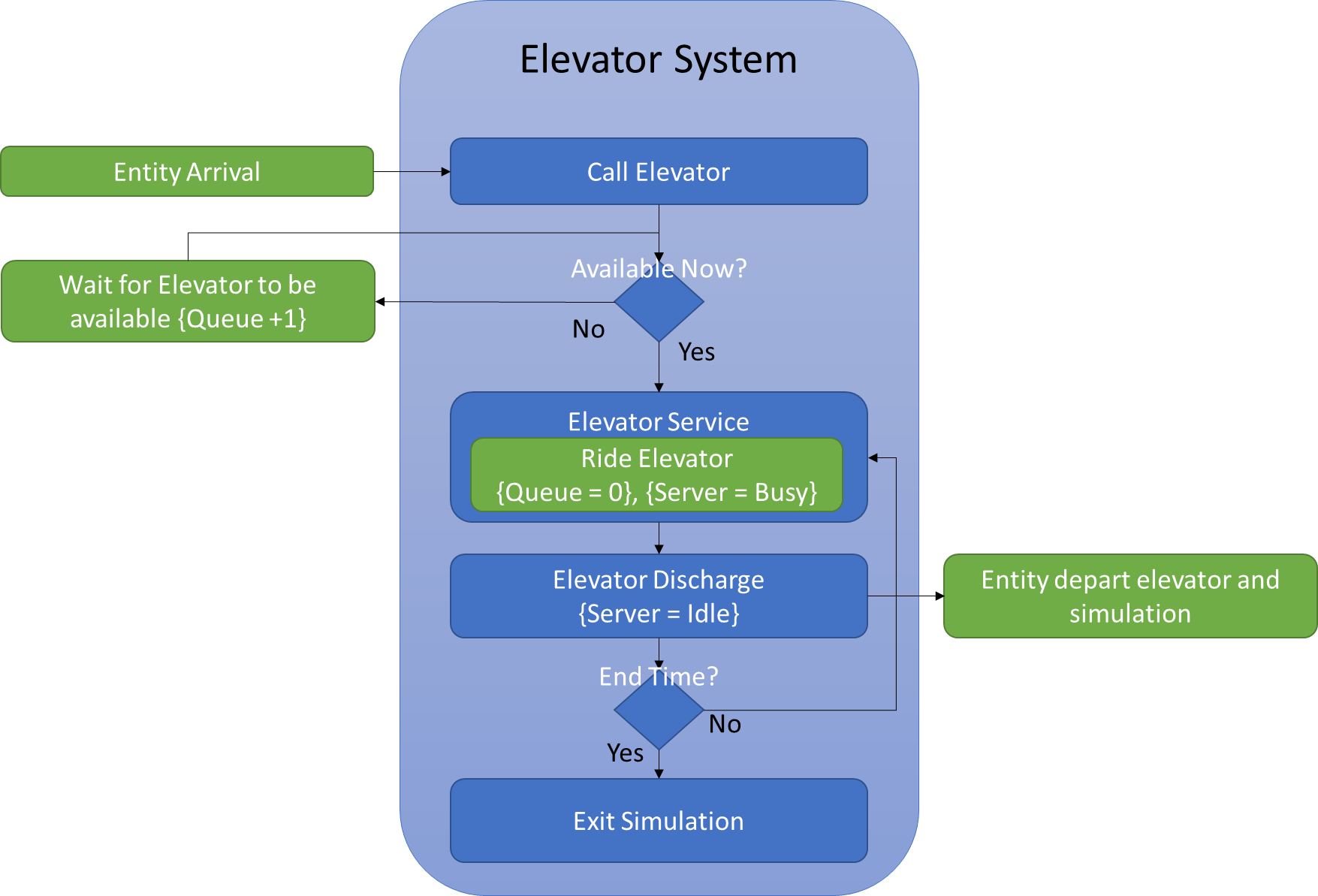
***INTRODUCTION***

The goal of the assignment is to model a rudimentary elevator event system using the SSSQ discrete event engine used in previous assignments.

Elevators in the real world serve multiple queues of entities at the same time, i.e. an elevator might pick up 3 people on floor 1 going to floors 3, 5, and 10 whilst passing by a 4th person trying to go from floor 5 to floor 1 and pick up a 5th person on floor 6 going to floor 9 before delivering the last of the initial 3 people to floor 10 and returning to the 4th person on floor 5. This code will assume far less complexity and simplify many characteristics of actual elevators out of consideration.

***DESIGN***

In this rudimentary elevator model, the system will be modeled such that entities are modeled to arrive with a Poisson distribution, mean of 2, and with service times (can be thought of as time to reach destination floor) modeled with an Exponential Distribution, lambda of 1. All entities are assumed to arrive at the same floor and travel in the same direction. If the elevator is in service at the time an entity arrives and calls for it, the entity must wait in a queue. Once the elevator has discharged its passengers, it returns to the origin floor, which takes the same amount of time as it took to deliver its last passenger. If there are multiple entities in the queue when the elevator is available, they will all begin service together and depart at their prearranged departure times, similar to multiple riders that start at the same floor and ride to different floors. The event diagram below shows the flow of events in the system.

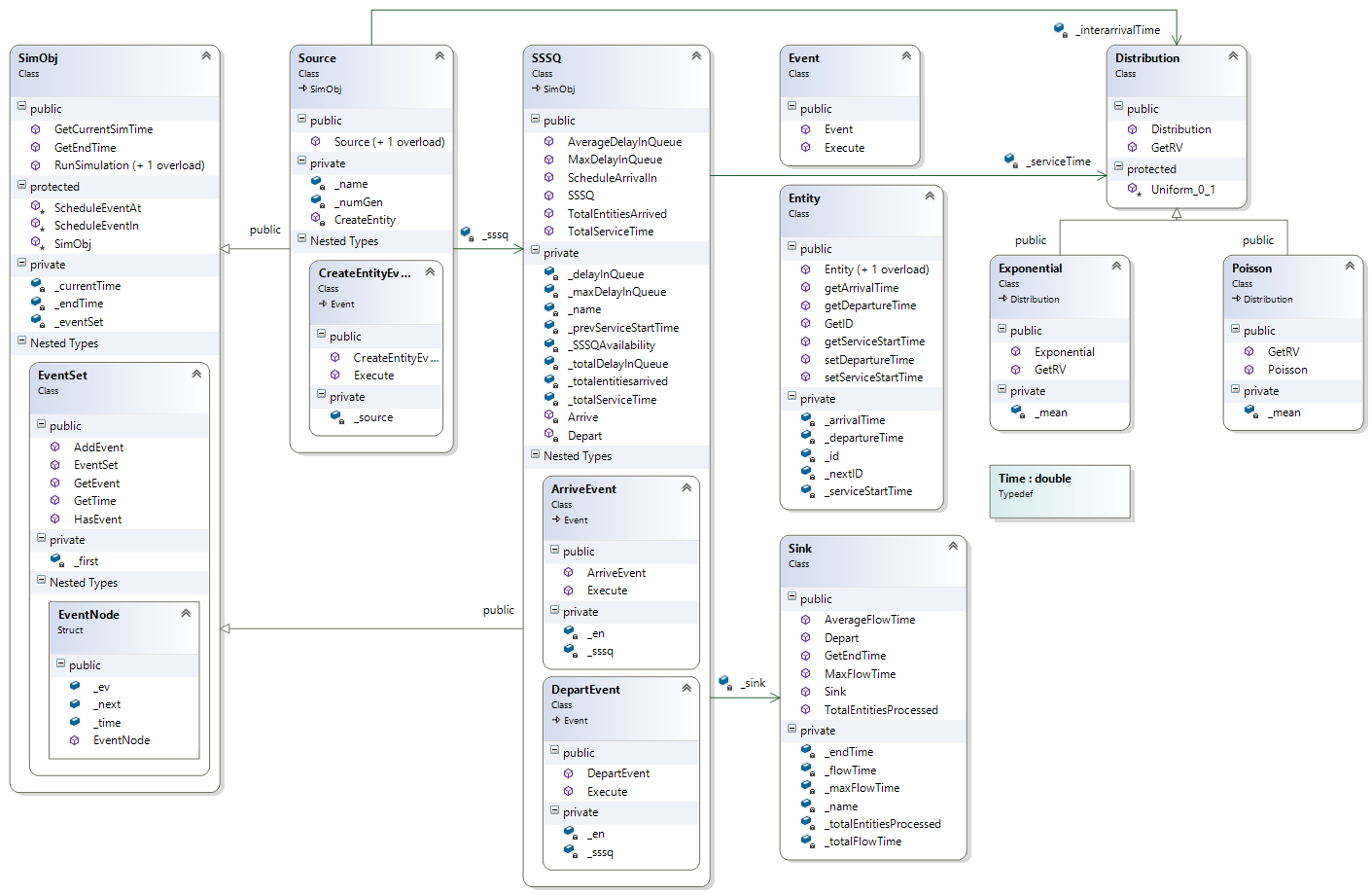


To model this system using the previously assembled discrete event code, I changed the SSSQ method “Arrive()” to check for entities already in the queue (lines 83-87, SSSQ.cpp). To determine this, a new private variable “\_preServiceStartTime” is used to preserve the previous entity’s service start time (line 120, SSSQ.cpp).

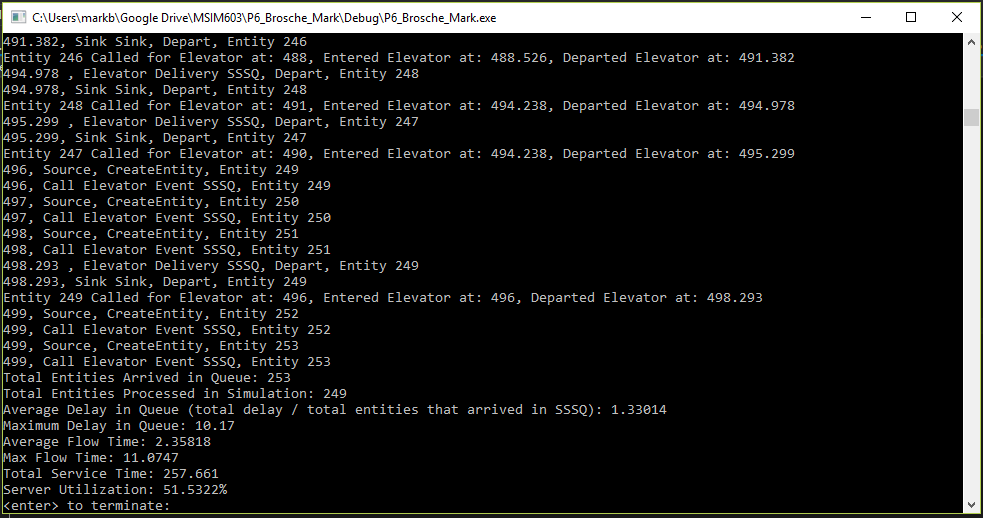
The “\_prevDepartureTime” private variable of SSSQ was also changed to “\_SSSQAvailability” account for the time required for the elevator to return from its last discharge event (line 118, SSSQ.cpp).

Lastly, Main.cpp was changed to populate SSSQ() and Source() with the appropriate distributions and to execute RunSimulation() for 500.0 time units.

Class Diagram:



***RESULTS:***



**Figure 1.** Console output for endTime = 500.0

The results show the following:

Total Entities Arrived in Queue: 253

Total Entities Processed in Simulation: 249

Average Delay in Queue (total delay / total entities that arrived in SSSQ): 1.33014

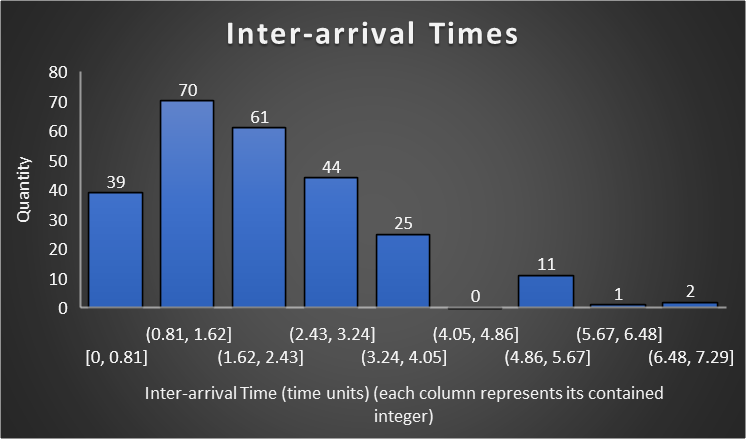
Maximum Delay in Queue: 10.17

Average Flow Time: 2.35818

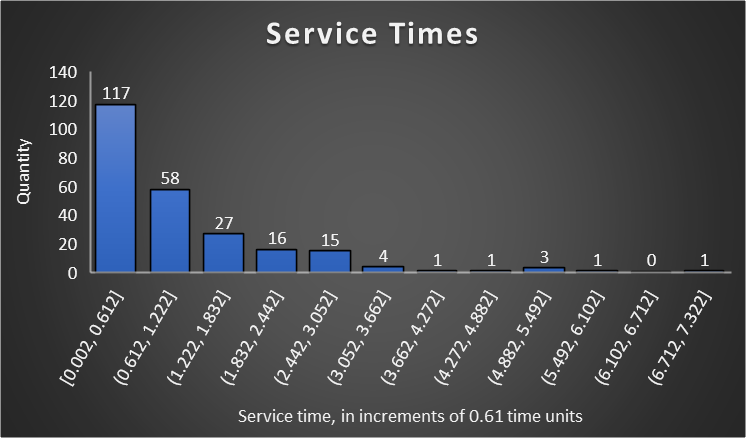
Max Flow Time: 11.0747

Total Service Time: 257.661

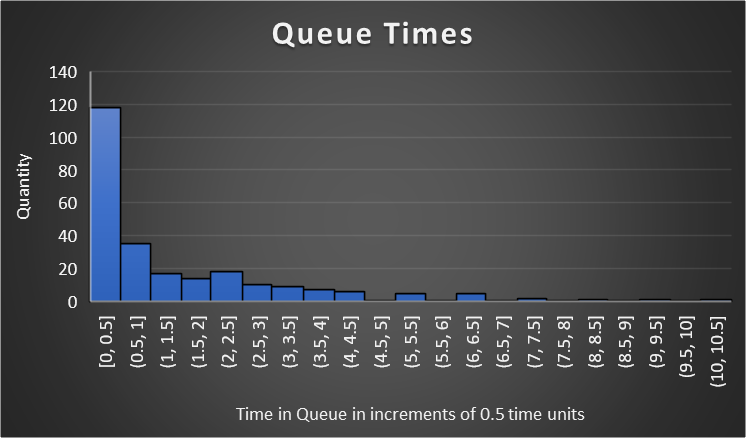
Server Utilization: 51.5322%



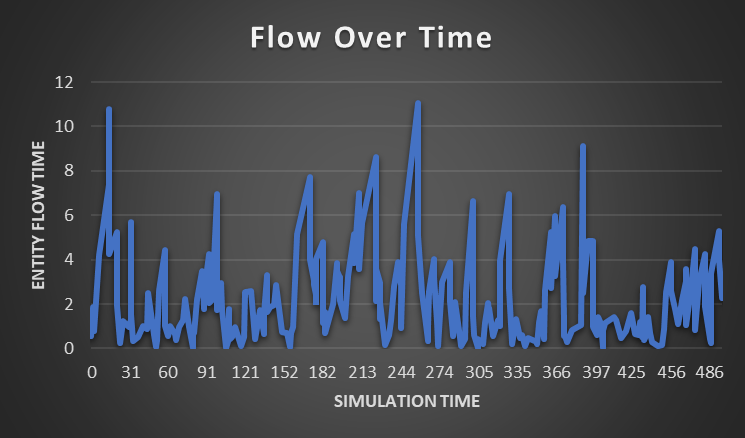
**Figure 2.** Entity inter-arrival times (Poisson Distribution, mean = 2)



**Figure 3.** Entity service times (Exponential Distribution, λ = 1)



**Figure 4.** Distribution of entity queuing times



**Figure 5.** Entity flow through the Elevator over the entire simulation

***CONCLUSION:***

This simplified elevator system shows a roughly 50% utilization and this seems to be due to the time factored into the return of the elevator to the queue from its previous discharge. Interestingly, previous implementations tried (can be found in Book2.xlsx) where the elevator carried a single rider at a time showed a utilization rate of ~13% but processed only 9 fewer entities of the 253 that arrived.

The efficiency gain in queuing and flow times when clearing the queue at each elevator availability would certainly be desirable in any real-world system, however adding in more information (e.g. modeling different starting floors, prioritizing pickups, entity direction, etc.) to be more realistic would introduce more sources of uncertainty. Ultimately, a linked list simulation with precomputed entity event times isn’t necessarily capable of modeling a real elevator.