 ***MSIM 603: Simulation Design***

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

***Old Dominion University***

**Programming Assignment Five**

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 provide the C++ code to extend our discrete event simulation engine and out SSSQ example to include performance parameters for our SSSQ simulation. This extension must include computing and displaying the following performance parameters:

* Number of entities processed completely through the simulation
* Average delay in the queue
* Maximum delay in the queue
* Average flow time
* Maximum flow time
* Server utilization

You should compute these performance parameters for two separate SSSQ runs. The first run should be with 100 entities. The second run should be with an end time of 200.0. Use the following theoretical distributions:

* Triangle (1, 3, 5) for the interarrival time
* Triangle (2, 3, 4) for the service time

**Grading:**

Design: 10 pts

Implementation: 20 pts

Results: 10 pts

***INTRODUCTION***

The goal of the assignment is to provide analysis methods in the given code base that show how the SSSQ application performs. I expect to see performance that is dependent on the distributions selected for interarrival time and how backed up the queue becomes during the application.

***DESIGN***

The methods to enable the analysis of system performance are provided within the Sink class. To enable those methods to work, the times for arrival, service, and departure need to be stored by each entity moving through the simulation. The Entity class already contained some of the needed storage methods that simply needed to be called in the provided code base.

For this assignment, I have made edits to the following files:

Sink.h, Sink.cpp:

* Contains performance measurement functions required for this assignment
* Depart() has been modified to record and store the departure time of each entity as it departs. It also calls the performance monitoring methods defined within Sink.

SSSQ.cpp:

* Implemented the setServiceStartTime() methods to store the time each entity is served, for later access by the performance measures in the Sink.

Source.cpp:

* Implemented the Entity(Time arrivalTime) overload to record and store the arrival time of each Entity for later access by the performance measures in the Sink.

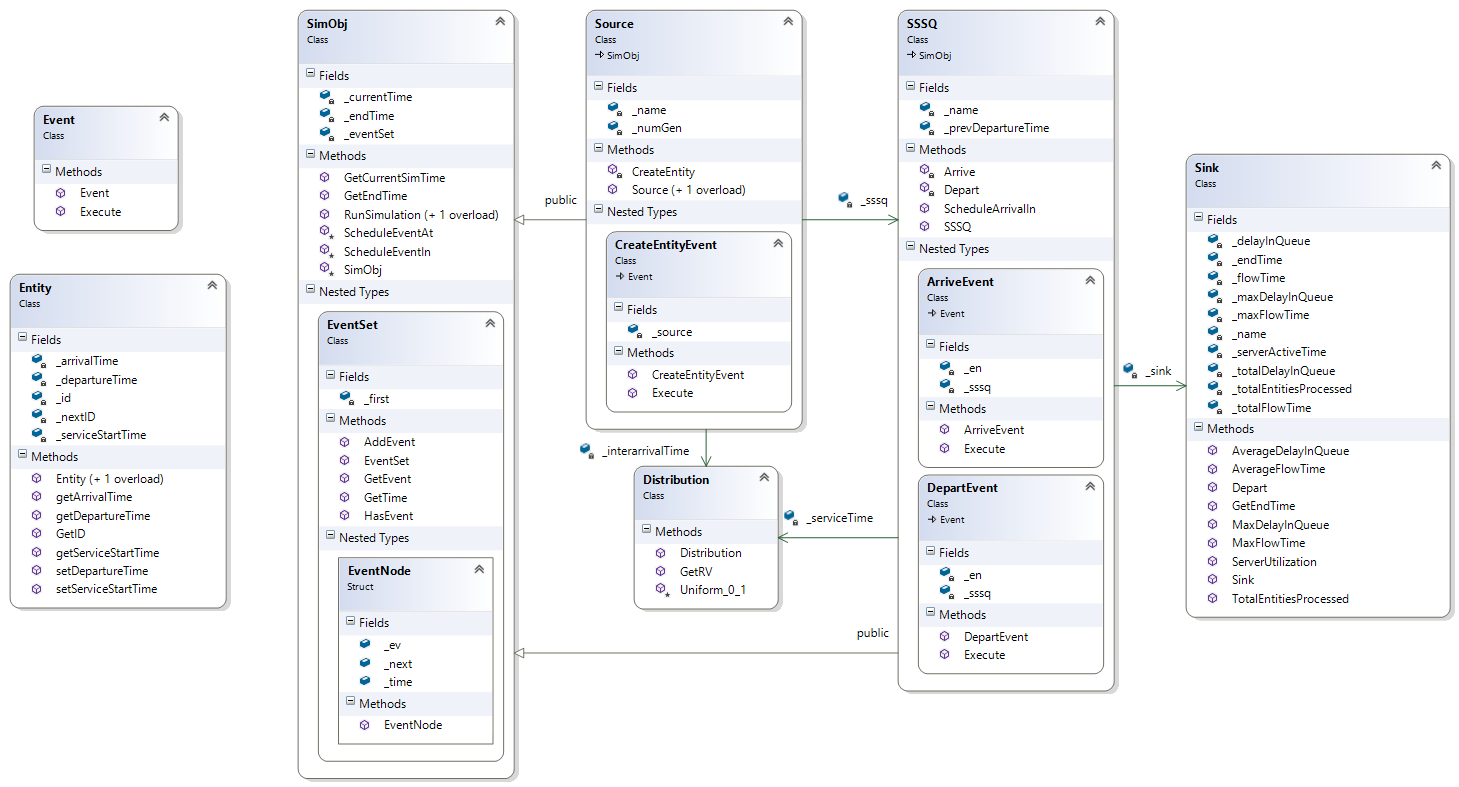
Simulation.h, Simulation.cpp:

* Added a new method to record and store the end time of the simulation if RunSimulation(Time endTime) is called for later access by the performance measures in the Sink.

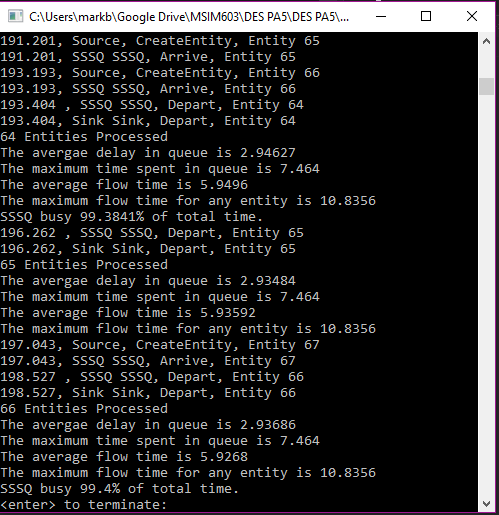
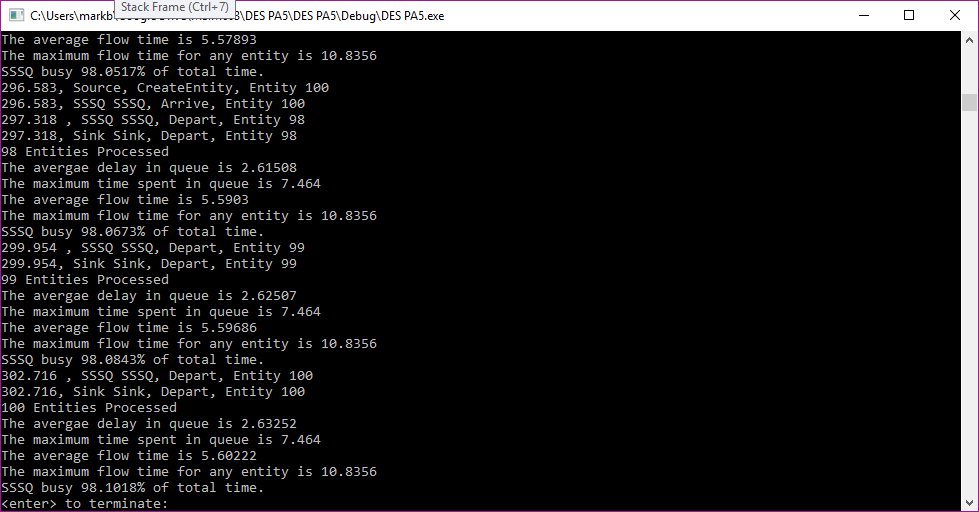
Main.cpp:

* Swapped the distribution parameters for Source and SSSQ to reflect the assignment requirement.

Class Diagram:



***RESULTS:***

**Left:** Console output for endTime = 200.0, **Right:** Console output for 100 entities.

The results show the following for the required methods:

For 100 entities:

* Total Number of Entities processed = 100
* Average delay in the queue = 2.63252
* Maximum delay in the queue = 7.464
* Average flow time = 5.60222
* Maximum flow time = 10.8356
* Server utilization = 98.1018%

For endTime = 200.0

* Total Number of Entities processed = 66
* Average delay in the queue = 2.93686
* Maximum delay in the queue = 7.464
* Average flow time = 5.9268
* Maximum flow time = 10.8356
* Server utilization = 99.4%

***CONCLUSION:***

With more time, I would have organized the output to be easily plottable and that would have made patterns in system behavior easier to discern. Regardless, the maximum delay and maximum flow time for both required approaches were each reached by the time the 20th entity was served. The average flow time was roughly equivalent to the sum of the expected interarrival and service times, and the maximum flow time was roughly equivalent to the sum of the max interarrival and service times.