Queueing Simulator Functional Specification

Michael Albers

CSCI 6960

University of Colorado Denver

# Revision History

|  |  |
| --- | --- |
| **Date** | **Changes** |
| 2016-06-30 | Initial version |
|  |  |
|  |  |

Contents

[Revision History 2](#_Toc455403906)

[Overview 4](#_Toc455403907)

[What Queueing Simulator Isn’t 4](#_Toc455403908)

[Software 4](#_Toc455403909)

[Use Cases 4](#_Toc455403910)

[Queuing Simulator Startup 5](#_Toc455403911)

[Main Success Scenario 5](#_Toc455403912)

[Extensions 5](#_Toc455403913)

[Simulation Start 5](#_Toc455403914)

[Main Success Scenario 5](#_Toc455403915)

[Extensions 6](#_Toc455403916)

[Simulation Control 6](#_Toc455403917)

[Main Success Scenario 6](#_Toc455403918)

[Extensions 7](#_Toc455403919)

[Camera Control 7](#_Toc455403920)

[Main Success Scenario 7](#_Toc455403921)

[Extensions 7](#_Toc455403922)

[Simulation Termination 8](#_Toc455403923)

[Main Success Scenario 8](#_Toc455403924)

[Extensions 8](#_Toc455403925)

[Screens 8](#_Toc455403926)

[Real-Time Mode 8](#_Toc455403927)

[Screens Not Shown 10](#_Toc455403928)

# Overview

Queueing Simulator is an implementation of a microscopic crowd simulator. It is intended to be a flexible platform which can be used to model somewhat arbitrary crowd behaviors in two dimensions. These behaviors are intended to be based upon the concept of steering behaviors introduced by Craig Reynolds in his paper entitled “Steering Behaviors for Autonomous Characters”.

Overall Queueing Simulator is comprised of two main parts: an engine used to drive some pre-packaged foundation simulation elements (actors, behaviors, etc.) and user defined elements which expand upon the foundation elements. Simulations piece together the necessary elements to define the desired simulation. The simulator is designed to be run on any machine from a personal computer to a supercomputer. Each running instance of Queueing Simulator is a single user application.

A single simulation can be run at a time. Each simulation is defined by a configuration file which the user loads. (Details about the configuration file can be found in the technical specification.) Queueing Simulator supports two modes of operation: real-time and batch. Any simulation can run in either mode. When run in real-time mode the simulation is visualized real-time and the user can control the camera location (x and y directions plus zoom). In batch mode a movie file of the simulation is generated. Aside from presentation, there are no differences in the simulation itself between the different modes. For each mode the user can start, pause and stop the simulation at any point.

At completion, each simulation generates a set of metrics about the simulation including total run time and others. These metrics are then viewable by the user and can be saved to a file.

**[open issue]** – Need to determine what metrics are available. Do they differ per simulation?

As Queueing Simulator is a solo project, this specification is mostly just an exercise in writing a functional specification. The primary benefit is just to have written it. However it should serve to preserve design integrity through the project. This is my first spec, so please bear that in mind while reading this.

Queueing Simulator was originally designed to model how people queue (i.e., form lines like those to enter a concert venue). This was implemented for CSCI 5551. The idea for a more general purpose simulator quickly arose and turned into this project. However, I kept the older name because it was better than some meaningless name for example “eggplant”, “gradle”, “git”) or something boringly generic (“Multipurpose Flexible Crowd Simulator” or some such nonsense).

# What Queueing Simulator Isn’t

Queueing Simulator is a full-scale software design project coupled with parallel computing principles. Traditionally crowd simulators fall into the wheel-house of computer graphics or artificial intelligence. That is not the case here. While Queueing Simulator does support some rudimentary forms of each, they are not the primary purpose. In other words, Queueing Simulator is not intended to experiment in, expand upon or further research either of the fields of computer graphics or artificial intelligence.

# Software

The software will come in two pieces. The first is the Queueing Simulator executable, this is the engine. The second is a prepackaged plugin which contains basic steering behaviors, actors and other “basics” to be used to build simulations. The user of the software is able, and encouraged to, create their own plugins to enhance and expand the simulation capabilities of Queueing Simulator. These user-defined plugins should be built upon the contents of the prepackaged plugin (i.e., inherit from classes contained therein and use some of the behaviors instead of creating duplicates). The details about the plugin system, including how to create one, can be found in the technical specification.

# Use Cases

The following use cases describe the major ways in which Queueing Simulator is used. They cover starting the application to completion of a simulation.

In all use cases there is only one actor, the user that started the software.

## Queuing Simulator Startup

This use case describes the steps taken on the part of the user to start Queuing Simulator.

### Main Success Scenario

1. User enters program name at command line, providing any options so desired.

**[open issue]** – List options here or technical spec? Will there be any options?

2. Program initializes self

3. Program displays Real-Time GUI for further interaction (see “Screens” section below). No more actions taken until the user proceeds.

The following GUI elements will be enabled (all others disabled):

File->Open, File->Exit, View->Messages, Help->About

### Extensions

1.

1. Invalid option(s) provided

1. Error message is printed to the terminal, program terminates with error code.

2. User enters 'help' option

1. Help information printed to terminal, program terminates with nominal code.

2.

1. Initialization failure

1. Error message(s?) is printed to terminal, program terminates with error code.

## Simulation Start

This use case describes the steps need to start a simulation running. This assumes the “Queueing Simulator Startup” use case has already been successfully.

### Main Success Scenario

1. From the GUI the user selects 'File->Open' to select a simulation configuration file.

2. File system navigation dialog is opened.

3. User finds desired configuration file.

4. User selects 'Ok' (or 'Open', depending on implementation). Dialog closes.

5. File is read in.

6. Simulation name field is populated on GUI.

7. Simulator enables the 'Play' button.

8. User selects Real-Time or Batch mode. GUI shows/hides available controls based on mode selection. When selecting Real-Time mode the Output File GUI elements are hidden and the Camera Control elements are shown. When selecting Batch mode the inverse is done.

9. User clicks the 'Play' button. 'Pause' and 'Stop' buttons enabled, 'Play' button and ‘File->Open’ menu selection are disabled. Mode selection radio buttons are disabled.

10. Simulation starts.

### Extensions

4.

1. Use selected 'Cancel'

1. Navigation dialog closed, use case aborted.

5.

1. File cannot be read.

1. Error dialog displayed, use case aborted.

2. File has invalid format, invalid data, other read/parse error.

1. Error dialog displayed, use case aborted.

## Simulation Control

This use case describes the ability to control the simulation while it is running in either Real-Time or Batch mode. This assumes the “Simulation Start” use case has already been run successfully. This can be run in parallel with the “Camera Control” use case (when running the simulation in Real-Time mode).

### Main Success Scenario

1. User clicks the ‘Pause’ button.

2. Simulator pauses the simulation:

1. Elapsed time count pauses.

2. Simulation state becomes frozen (no movement updates).

3. ‘Pause’ button disabled.

4. ‘Play’ button enabled.

3. User clicks the ‘Play’ button.

4. Simulator continues the simulation:

1. Elapsed time count resumes

2. Simulation state resumes updating (movement continues).

3. ‘Pause’ button enabled.

4. ‘Play’ button disabled.

5. User clicks the ‘Stop’ button:

1. Execute “Simulation Termination” use case.

### Extensions

1.

1. User clicks the ‘Stop’ button

1. Jump to step 5.

4.

1. User clicks the ‘Stop’ button.

1. Jump to step 5.

## Camera Control

This use case describes the steps taken by the user to control the camera while a simulation is in progress. This assumes the “Simulation Start” use case has already been run successfully. This can be run in parallel with “Simulation Control” use case. This use case only applies when running the simulation in Real-Time mode.

### Main Success Scenario

1. User clicks one of the arrow buttons.

2. Simulator repositions camera a fixed distance, in the direction of the button pushed, from the current camera position. Simulator updates displayed camera X & Y coordinates.

3. User clicks one of the zoom buttons.

4. Simulator repositions the camera a fixed distance, in the direction of the button pushed, from the current camera position (i.e., zoom in/out). Simulator updates zoom percentage (100% is the default, 0% is level with simulation "surface").

### Extensions

2.

1. Camera at limit of current direction of movement.

1. Simulator does not move camera any further. No update to camera coordinates is made.

4.

1. Camera is at 0% zoom.

1. Camera is not moved any closer to the "surface". Zoom percentage is not updated.

## Simulation Termination

This use case describes the steps taken when a simulation is terminated, either by naturally ending or by user-defined termination (i.e., pressing the ‘Stop’ button). This assumes the “Simulation Start” use case has already been run successfully.

### Main Success Scenario

1. Simulator disables the ‘Pause’ and ‘Stop’ buttons.

2. Simulator disables the camera control buttons when running in Real-Time mode.

3. Simulator freezes all time values and % complete.

4. Simulator enables the play button.

5. Simulator enables the ‘View->Results’ menu selection.

6. Simulator enables the ‘File->Open’ menu selection.

7. Simulator enables the Mode radio buttons

8. Simulator closes movie file when running in Batch Mode.

### Extensions

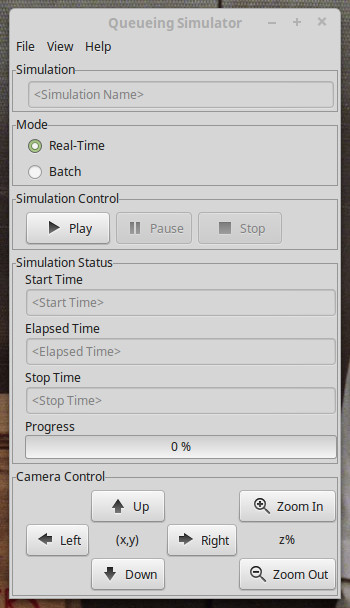
None.

# Screens

Queueing Simulator is rather simple, graphically. Below are the major screens.

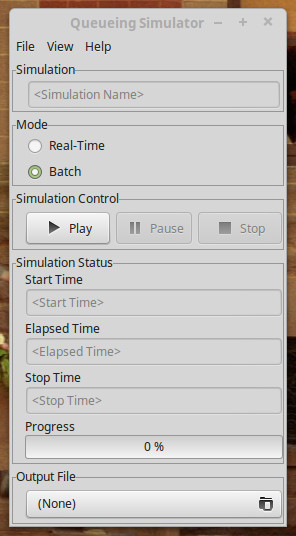
## Real-Time Mode

When running a simulation in real-time mode, the following screen will be the main source of interaction between Queuing Simulator and the user.



Simulation configuration files are loaded via the “File->Open” menu. The name of the simulation will be populated. The only controls active are the mode selection radio buttons (see below for the Batch mode screen), and the play button. Note that the Play button and Camera Control buttons are incorrectly shown as active in this picture. Please pretend they are not.

Once the Play button is clicked, the start time field will be populated and the Camera Control buttons will be enabled. And, of course, the simulation will start. Specifics can be found in the use-cases above.



The Batch mode screen behaves pretty much identically to the Real-Time mode screen. The obvious difference is the lack of camera controls with the Output File elements. The Output File button allows the user to select the location of the generated movie file. As before, it and the Play button are mistakenly shown as active in this screenshot.

## Screens Not Shown

The following screens are not shown. At this point in development they are about as basic as they come. I didn’t deem it worthwhile to generate prototypes. A brief description of each is given.

* Results screen

This screen shows the results of the simulation, namely the metrics mentioned above. It is really just a large text box with a means to save the data to a file. The save components will either be a button or menu. This screen can be opened through the “View->Results” menu on either the Real-Time or Batch GUIs. It will only be available when a simulation has stopped.

* Messages screen

This screen shows any error or log messages generated during execution of Queueing Simulator. This, too, will just be a large text box. This screen can be opened through the “View->Messages” menu on either the Real-Time or Batch GUIs at any point during Queueing Simulator execution.

* Visualization screen

This screen is an OpenGL screen. It shows the visualization when running in Real-Time mode. The Real-Time GUI camera control buttons affect what is seen in this screen. However, on its own, this screen has no user interactions available. This screen will be displayed automatically when a simulation in Real-Time mode starts.

**[open issue]** – How to handle this window being closed during a simulation? Should it just not be able to? Better idea might be to update the “View” menu to add an item to open it again. Also, how will OpenGL react to it being closed?