Queueing Simulator

Software Requirements Specification (SRS) Document

Author: Michael Albers, under advisement of Dr. Gita Alaghband

University of Colorado, Denver

Computer Science Department

Document Control

|  |  |  |
| --- | --- | --- |
| Revision | Date | Changes |
| 1.0 | 2016-02-16 | Initial version. |
|  |  |  |
|  |  |  |
|  |  |  |

[1 Introduction 4](#_Toc443389753)

[1.1 Purpose 4](#_Toc443389754)

[1.2 Scope 4](#_Toc443389755)

[1.3 Definitions, Abbreviation and Acronyms 4](#_Toc443389756)

[1.4 Overview 4](#_Toc443389757)

[2 Overall Description 4](#_Toc443389758)

[2.1 Product Perspective 4](#_Toc443389759)

[2.1.1 User Interfaces 4](#_Toc443389760)

[2.1.2 Software Interfaces 4](#_Toc443389761)

[2.2 Product Functions 5](#_Toc443389762)

[2.2.1 Graphical User Interface 5](#_Toc443389763)

[2.2.2 Simulation Engine 5](#_Toc443389764)

[2.2.3 Plugins 5](#_Toc443389765)

[2.3 User Characteristics 5](#_Toc443389766)

[2.4 Constraints 5](#_Toc443389767)

[2.5 Assumptions and Dependencies 6](#_Toc443389768)

[3 Specific Requirements 6](#_Toc443389769)

[3.1 Functional Requirements 6](#_Toc443389770)

[3.1.1 Simulation Engine 6](#_Toc443389771)

[3.1.2 Real-time Mode 7](#_Toc443389772)

[3.1.3 Batch Mode 7](#_Toc443389773)

[3.1.4 Plugin System 7](#_Toc443389774)

[4 References 9](#_Toc443389775)

# Introduction

## Purpose

This document contains the high level description and functional requirements for the Queueing Simulator. This document is primarily for use by myself (as sole designer, developer and tester). It is also intended for parties involved in the evaluation of this software application.

This document is based off the IEEE830-1998 SRS template [1]. It does not fully conform due to my own discretion. I believe that certain detail are not suitable for a requirements specification (i.e., user interface details) as they are implementation dependent. Also, some information is simply not known at this time such as version numbers of software interfaces.

## Scope

This software product will be called Queueing Simulator. It is a crowd simulator used to model the behavior of individuals when forming a queue. Beyond this it will be able to model general movement of simulated intelligent actors within a two dimensional environment. Queueing Simulator will visually represent this modelled movement.

The primary purpose of Queueing Simulator is to create a full-featured, though not commercial grade, crowd simulator. This program is being developed as the final project for obtaining my Master’s Degree in the Computer Science from the University of Colorado, Denver.

## Definitions, Abbreviation and Acronyms

All definitions, abbreviations and acronyms are either “well known” or listed inline within the document.

## Overview

The next section of this document presents an overall description of Queueing Simulator. Following that is the complete list of functional requirements of the program. This document follows the template laid out by the IEEE 830 standard for software requirements specifications.

# Overall Description

## Product Perspective

Queueing Simulator is a wholly standalone application. While dependent on other software packages, it is an end within itself. It does not interact with any other applications nor is it a component within another system.

### User Interfaces

No user interfaces defined at this point.

### Software Interfaces

Refer to section 2.5 for software dependencies.

## Product Functions

### Graphical User Interface

The sole means of control and visualization lies within the graphical user interface. From here the simulation to be run can be selected and controlled. The GUI has two distinct modes of operation, detailed below.

#### Real-time Mode

Simulations run in this mode allow the user to see the results of the simulation as the simulation is running (i.e., real-time). This mode provides controls to start, stop and pause the simulation. Other configuration controls for the simulation are provided. This mode displays the actors within the simulation as they move throughout the world.

#### Batch Mode

Batch mode removes all user control and real-time visualization from the simulation. The simulation is run and the resulting visualizations are written to a file for later viewing. The main motivation for this mode is to increase performance by removing the real-time visualization. Long-running simulations or those which contain significant numbers of actors are intended to be run in this mode.

### Simulation Engine

The simulation engine runs the simulation; it executes the behaviors of each actor within the system. The engine handles the concurrency necessary for the simulation. The engine uses two concepts which need defining. The first is the update rate, this is rate at which the engine runs the simulation, that is, how many discrete updates to the simulation are made within a given period. The second is the simulation rate, this rate is how much time is being modeled during each discrete update.

### Plugins

All actor behaviors are defined by plug-ins. The plugins are essentially user generated software libraries which take a certain set of inputs and provide a resulting vector for actor movement. The software will come with a pre-defined set of basic behaviors. Plug-ins are loaded real-time and do not cause a rebuild of the software.

## User Characteristics

There are two types of users of Queueing Simulator. The first set are simply those that use the simulator. They load simulations and run them. These users need very little, if any, knowledge of crowd simulation, coding or the software in general. Ideally, anyone with some computer literacy could start the software and run a simulation without any assistance.

The second type of user is one that creates behavior plugins. This set of users must have a knowledge of software creation, crowd simulation and linear algebra. These skillsets are needed for the creation of behaviors. Naturally, this set of users is a superset of the first set.

## Constraints

The primary constraint upon Queueing Simulator is the ability for the simulation engine to run the behaviors of the actors in parallel. This is critical to being able to simulate large numbers of actors. Thus actor data and the behavior plugins must be completely thread-safe. Additionally, in order to maintain “good” performance, the behaviors must create a minimal (ideally zero) number of critical sections of code. This parallelism cannot extend into the graphical operations as most, if not all, graphical packages and frameworks depend on single threaded operation.

## Assumptions and Dependencies

Queueing Simulator is designed with the following dependencies:

* Linux Operating System, kernel version of 3.14 or later. Distribution Agnostic.
* GNU Compiler Collection (GCC), version 4.9 or later.
* Eigen Linear Algebra Library, version 3.2.8 or later.
* GIMP Tool Kit (GTK) 3

# Specific Requirements

## Functional Requirements

### Simulation Engine

1. The engine shall simulate actor’s movements within a two dimensional environment (also called the “world”).
2. The engine shall maintain basic attributes for each actor. These include: radius, mass, maximum acceleration, maximum velocity, heading/orientation and position.
3. The engine shall use metric units for all applicable values.
4. The engine shall apply steering behaviors [2] to each actor, as long as the actor is still “in” the simulated world at each update.
5. The engine shall support running in real-time mode and batch mode (as described above).
6. When running in real-time mode, the engine shall support a variable simulation rate.
7. When running in batch mode, the engine shall support only a fixed simulation rate.
8. The engine shall support simulations with no defined ending condition, i.e., indefinite simulation.
9. The engine shall support simulations with defined ending conditions.
10. The engine shall support at least 100 actors with no performance degradation, regardless of which mode the engine is operating in (real-time or batch).
11. The engine shall support modelling of fixed environmental objects within the world.
12. The engine shall support modelling of at least 10 fixed environmental objects without performance degradation.
13. The engine shall provide actor data to the GUI component.
14. The engine shall provide the ability to save the current simulation state.
15. The engine shall provide the ability to load previously a saved simulation state.
16. The engine shall provide the ability to use user-defined random number generator parameters. These include initial seed value and the random number generator itself.
17. The engine shall provide the ability to accept user-defined initial simulation state.
18. The engine shall collect and report simulation metrics. Such metrics can include, but are not limited to: total simulation time, total actor travel, etc.
19. The engine shall provide simulation completion percentage to the GUI component.
20. The engine shall support use of code provided by valid plugins.

### Real-time Mode

1. The system shall graphically display the entire two dimensional world and the current location of the actors within the world.
2. The system shall provide graphical controls which allow the user to:

* Start the simulation. This includes initiating the simulation and continuing from a pause
* Pause the simulation. This is a temporary halting of the simulation during execution. The simulation can be continued when paused.
* Stop the simulation. Fully stop the simulation; continuation is not allowed after the simulation has been stopped.

1. The system shall provide graphical controls which allow the user to modify the simulation rate (speed up or slow down the simulation).
2. The system shall provide graphical controls to allow the user to change the camera location (in two dimensions) being used to display the world and actors.
3. The system shall provide graphical controls to change the zoom level of the camera being used to display the world and actors.
4. The system shall display the current run-time of the simulation as measured since the simulation was initiated.
5. The system shall display the current completion percentage of the simulation as reported by the simulation engine.
6. The system shall provide graphical controls to toggle displaying enhanced actor data for each actor. Enhanced data can include, but isn’t limited to, numeric positional data, mass, etc.
7. The system shall provide graphical controls to save simulation metrics from the simulation engine.
8. The GUI shall provide controls to save the current simulation state.
9. The GUI shall provide controls to load simulation state.

### Batch Mode

1. The GUI shall provide controls to load simulation state.
2. The GUI shall provide controls to start and stop the simulation (as defined in the “Real-time” section).
3. The GUI shall display the current completion percentage of the simulation as reported by the simulation engine.
4. The GUI shall provide controls to allow the user to select the file to which visualization data will be written.

### Plugin System

1. All plugins shall be loaded during system startup, and only at system startup.
2. During loading, all plugin data will be validated. Any plugin found to contain invalid data will not be loaded.
3. The plugin system shall support automated discovery of plugins on the filesystem.
4. Plugins shall contain software for any of the following set: actor definition, actor creation, obstacle definition, steering behaviors, random number generation, simulation progress determination and simulation ending condition(s).
5. Plugins shall conform to an API and reference document. These will be provided by Queueing Simulator and defined outside of this document.

# References

|  |  |
| --- | --- |
| [1] | IEEE, "830-1998 - IEEE Recommended Practice for Software Requirements Specifications," 1998. [Online]. Available: https://standards.ieee.org/findstds/standard/830-1998.html. [Accessed 9 8 2015]. |
| [2] | Craig Reynolds, “Steering Behaviors For Autonomous Characters”, in Game Developers Conference, San Jose, CA, 1999 |