**FUNCTIONAL SPECIFICATION**

**Scene Manager for Near Real-Time**

**Concurrent Scene Manipulation**

**2018**

**By James Schmidt**

**Version History**

**Version 1.0 – Draft**

**FUNCTIONAL SPECIFICATION –**

**Scene Manager for Near Real-Time Concurrent Scene Manipulation**

Table of Contents

1. INTRODUCTION, Overview and Purpose ……………………………….. 3
   1. Outcomes and Scope …………………………………………………... 3
   2. Project Requirements ………………………………………………….. 3
   3. Development Requirements …………………………………………… 3
   4. General Constraints ……………………………………………………. 3
2. FUNCTIONAL DESCRIPTION ………………………………………….. 4

2.1 User Functionality ……………………………………………………... 4

1. SPECIFIC REQUIREMENTS …………………………………………….. 4

3.1 External Interface Requirements ………………………………………. 4

3.2 Library Features ……………………………………………………….. 4

3.3 Advanced Functionality……………………………………………….. 4

**1. INTRODUCTION, Overview, and Purpose**

The purpose of this project is to give programmers the ability to concurrently manipulate a scene in real-time to make working on a scene easier and more efficient by allowing for fast iterations, zero merge conflicts, and promoting collaboration amongst developers. The project will incorporate a scene manager that will act as a host to the client developers, allowing for a single scene master to keep track of the scene state. Developers will be able to manipulate game objects freely, though the objects they select will be locked and unusable to other developers, until they deselect them. Changes to game objects will be handled by the scene manager to keep track of updating the master scene state. When the scene has been changed, the scene manager will check the changes for validation, once valid, the new scene state will be updated to any client that is viewing the scene. Unity will be utilized as the editor for this project, however, if issues arise, then the project may end up using a custom engine.

***1.1 Outcomes and Scope***

The planned outcome of this project is to create an embedded plug-in for Unity that will incorporate a scene manager for validating, maintaining the master scene state, and distributing changes to other Unity clients.

***1.2 Project Requirements***

* The project should allow for clients of the same Unity project to join.
* The project should show real-time updating for game objects manipulated by other clients.
* The project should be easy for programmers to understand/use.

***1.3 Development Requirements***

* The project should be developed using Unity’s embedded plug-in architecture.
* The project shouldn’t account for asset management. Any useful source control can handle that.
* Being embedded within Unity will allow for cross-platform integration.

***1.4 General Constraints***

* The plug-in must work Unity version 2017.3.1f1 or higher.
* The plug-in must compile properly and work with Visual Studio 2017.

**2. FUNTIONAL DESCRIPTION**

***2.1 User Functionality***

Users should be able to easily add the plug-in to their existing Unity project. User log-ins may be required given the need for individual account identification, otherwise guest accounts will be utilized with the ability to change their name.

**3. SPECIFIC REQUIREMENTS**

***3.1 External Interface Requirements***

The plug-in will utilize a host-server relationship between the scene manager and its clients.

***3.2 Plug-in Features***

* Connecting to host scene manager
* Client log-in
* Game object locking for selected objects
* Real-Time viewing of scene being manipulated by other clients
* Host scene manager tasked with pushing final build to source control

***3.2 Advanced Functionality***

The advanced functionality list below are stretch goals for the project.

* Highlighting game objects when selected to show locking
* Displaying name and info of who owns selected object
* Detecting time-outs and relinquishing locked objects
* Selecting a client to view their camera (see what they see)