Lantern Game

GAM200 Technical Specification

Sophomore Game Project

Fall 2023

Team Vyv

|  |  |
| --- | --- |
| Programmers: | Designers: |
| Tyler Dean | Zan Moffat |
| Michael Howard | Siddharth Mahesh |
| TayLee Young | Songyang Cai |
| Wang HeTong Louis |  |
| Thomas Stephenson |  |

(contents)

# **Architecture Overview**

(minor header)

(big list)

# **Graphics Overview**

**Graphics API:** *OpenGL (Version 3.3)*

|  |
| --- |
| Graphics Pipeline |
| Load bitmaps |
| Push light sources |
| Draw bitmaps to display |

**Image Loading: The engine is currently capable of loading bitmaps in the PPM format.**

**Animations:**

**Particle System:**

# **Physics Overview**

## *Subsystem 1 (Kinematics for example)*

## *Subsystem 2 (Collision for example)*

# **Player Controls**

Utilizing SDL2, the Inputs class handles and manages various keyboard and mouse inputs. It follows the Singleton design pattern as only one instance exists at any given time and can be considered to be a part of the Observer design pattern as it handles and dispatches input events. It is designed to accommodate a single player experience.

**Input Manager:** SDL2 Manages keyboard and mouse events.

# **Behavior**

**Any Specific Components (ie player):**

**(see sample for more deets)**

## **Debugging**

**ImGUI:** Windowed panels display live information of inputs.

**Assertions:** Asserts are implemented within all core systems with proper error checking and logging to verify everything runs smoothly within the underlying engine.

**Console and File Logging:** Both console and file logging can be utilized and customized to display any needed information.

# **Coding Methods**

**File Naming Conventions:**

**Code Naming Conventions:**

**Styling:**

**Guidelines:**

**Patterns:**

# **Version Control**

**Git:**

**SVN:**

# **Tools**

**Libraries**

**Other Tools (valgrind, cmake, etc)**

# **Editor Implementation**

**In-Game Value Editing:** Input values are monitored. In the future, the editor will be able to track the states and specific values of entities, and will allow for live editing of said specific features for debugging purposes.

# **Scripting Languages**

Scripting will be written in C++, as the engine is currently constructed using the same language. No other languages will be required for this project.

# **Technical Risks**

**TYLER THIS IS YOU BRO**

# Appendices

## Appendix A: Art Requirements

Describe the requirements for all art assets used in the game, including file naming conventions, file format(s), and anything else needed to create the assets.

Describe the process by which new art assets will be incorporated into the game (AKA “art pipeline”). Note, for teams with BFA students, this process ***cannot*** require the intervention of a programmer after Week 6 of the semester.

Describe the source or sources of all the art in your game (programmer art, DigiPen libraries, BFA students, etc.). Remember that you must create all your own art (or use DigiPen libraries). You cannot use your friends, family members, public domain material, etc. You can use art from other students not on your team (but you must give them credit). You can never use outside artists at all.

## Appendix B: Audio Requirements

Describe the requirements for all audio assets used in the game, including file naming conventions, file format(s), and anything else needed to create the assets.

Describe the process by which new audio assets will be incorporated into the game (AKA “audio pipeline”). Note, for teams with BAMSD students, this process ***cannot*** require the intervention of a programmer after Week 6 of the semester.

Describe the source or sources of all the audio in your game (programmer audio, DigiPen libraries, other students, etc.). Remember that you must create all your own audio (or use DigiPen libraries). You cannot use your friends, family members, public domain material, etc. You can use audio from other students not on your team (but you must give them credit). You can never use outside musicians at all.

Our Audio Engine is using Low-Level FMOD. We have created a folder in Assets that is sorely for audio. In the folder, there are 3 different folders, all for different purposes. We have the Music folder, SFX folder and Voice-over. Music is for all the background music, that will be looping until we decided to stop it. SFX are for sound effects like footsteps and interactions. Voice-over is for character voice which we might add in GAM250. All the 3 folders will be loaded into the engine separately. They are in ogg file format rather than mp3 because of the smaller file size and better sound quality. Our audios are sourced from the DigiPen libraries, converted to ogg.