**The Shining**

**Technical Design Document**

*This technical design* *document describes the technical details and requirements for The Shining, a top down adventure puzzle game where the player escapes a spooky hotel.*

# Contents

[Contents 2](#_Toc31226394)

[Section 1 – Game Overview 3](#_Toc31226395)

[1.1 Game Summary 3](#_Toc31226396)

[1.2 Platform 3](#_Toc31226397)

[Section 2 – Development Overview 4](#_Toc31226398)

[2.1 Development Environment 4](#_Toc31226399)

[2.2.1 Development Hardware 4](#_Toc31226400)

[2.2.2 Development Software 4](#_Toc31226401)

[2.2.3 External Code 4](#_Toc31226402)

[Section 3 – Game Mechanics 5](#_Toc31226403)

[3.1 Main Technical Requirements 5](#_Toc31226404)

[3.1.1 Randomly Generated Floors 5](#_Toc31226405)

[3.1.2 Collision 5](#_Toc31226406)

[3.1.3 Psycho Axe Man 5](#_Toc31226407)

[3.2 Architecture 6](#_Toc31226408)

[3.2.1 Game Object 6](#_Toc31226409)

[3.2.2 Scene Manager 7](#_Toc31226410)

[3.2.3 Main UML 8](#_Toc31226411)

[3.3 Game Flow 9](#_Toc31226412)

[3.4 Game Objects and Logic 10](#_Toc31226413)

[Section 4 – UI and Controls 11](#_Toc31226414)

[4.1 Screen Flow 11](#_Toc31226415)

[4.2 Game Shell 11](#_Toc31226416)

[4.3 Play Screen 13](#_Toc31226417)

[4.4 Controls 13](#_Toc31226418)

[4.4.1 Controller 13](#_Toc31226419)

[4.4.2 Keyboard 13](#_Toc31226420)

# Section 1 – Game Overview

## 1.1 Game Summary

*The Shining is a top-down adventure puzzle game where the player tries to escape a hotel. Players must avoid enemies like the axe wielding psycho and get out of the hotel, moving from the top floor to the lobby. There will be enemies, items and power ups throughout the hotel for the player to fight, collect, use and navigate.*

## 1.2 Platform

*The game will be developed for Windows/Mac/Linux and will be able to be controlled with a keyboard or game controller.*

# Section 2 – Development Overview

## 2.1 Development Environment

*The game will be made using the* [***ASGE Game Framework***](https://huxyuk.github.io/AwesomeSauceGE/getting-started.html) *and C++.*

### 2.2.1 Development Hardware

*The game will be developed on Windows computers.*

* *CLion requires a minimum of Windows 7, 2GB of RAM and 2.5GB of disk space (plus 1GB for caches).*
* *GitKraken requires a minimum of Windows 7, 4GB of RAM and 5GB of disk space.*

### 2.2.2 Development Software

*We will be using* [***CLion***](https://www.jetbrains.com/clion/) *to develop C++ code. Version control will be handled by* [***Git***](https://git-scm.com/) *with the repository hosted by* [***GitHub***](https://github.com/)*. We will also be using*[***GitKraken***](https://www.gitkraken.com/) *to provide a graphical interface for Git.*

*We will be using Trello for project management and to track our progress, our Trello board can be found here:* [**https://trello.com/b/nsPLdJVR/llp-game-2**](https://trello.com/b/nsPLdJVR/llp-game-2)

### 2.2.3 External Code

*We will be using the ASGE Framework for C++ written by James Huxtable to provide basic game functionality such as the game loop, sprite rendering and input events.*

# Section 3 – Game Mechanics

## 3.1 Main Technical Requirements

### 3.1.1 Randomly Generated Floors

*One main technical requirement of this game will be the randomly generated floors. Rooms will be randomly selected from a set of sprites to make up the floor. This will involve lining up doors, so they lead somewhere and stopping the floor from reaching out too far. The algorithm we have for this is:*

* *Generate starting room, with all 4 doors.*
* *Create a queue of rooms to generate based on open doors*
* *For each room in the queue*
  + *Check surrounding rooms for doors*
  + *Select the required doors*
  + *Randomly generate a room with the correct doors*
  + *Add new rooms to generate to queue*
* *Generate mini map*

### 3.1.2 Collision

*Collision will be another big requirement of this game. We will be using simple bounding boxes to determine collision between enemies, projectiles, rooms and items.*

*The collisions we will need to detect will be:*

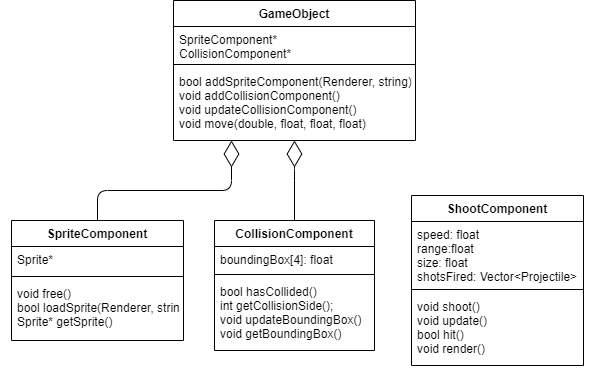
* *Player and Room*
* *Player and Items*
* *Player and Enemy Projectile*
* *Enemy and Player Projectile*
* *Enemy and Room*
* *Enemy and Items*

### 3.1.3 Psycho Axe Man

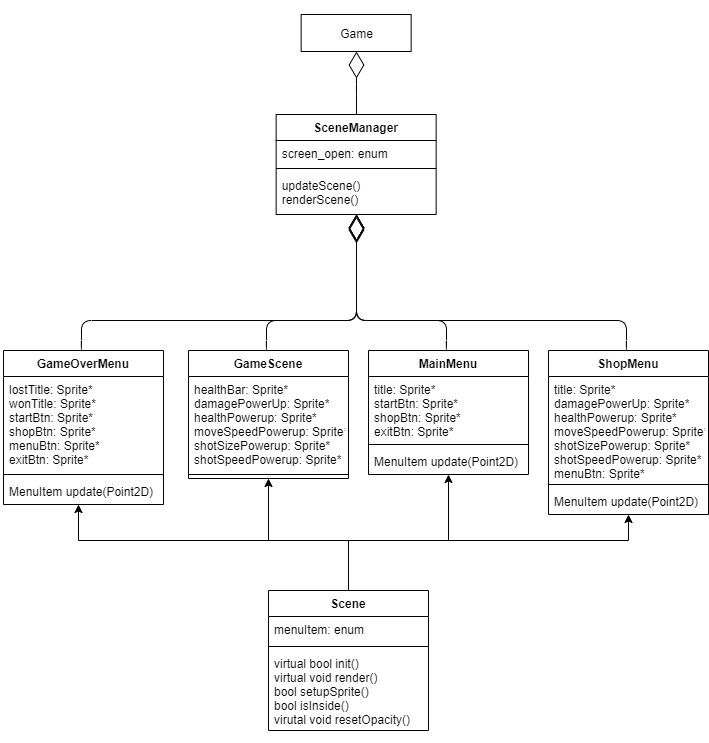
*Another technical requirement of the game is the Psycho Axe man. He will need to randomly appear and attack the player and then retreat when his health is too low. To do this we will need to have a timer, but also check other game values, like the number of enemies to make sure the player is not overwhelmed with enemies on the first level.*

## 3.2 Architecture

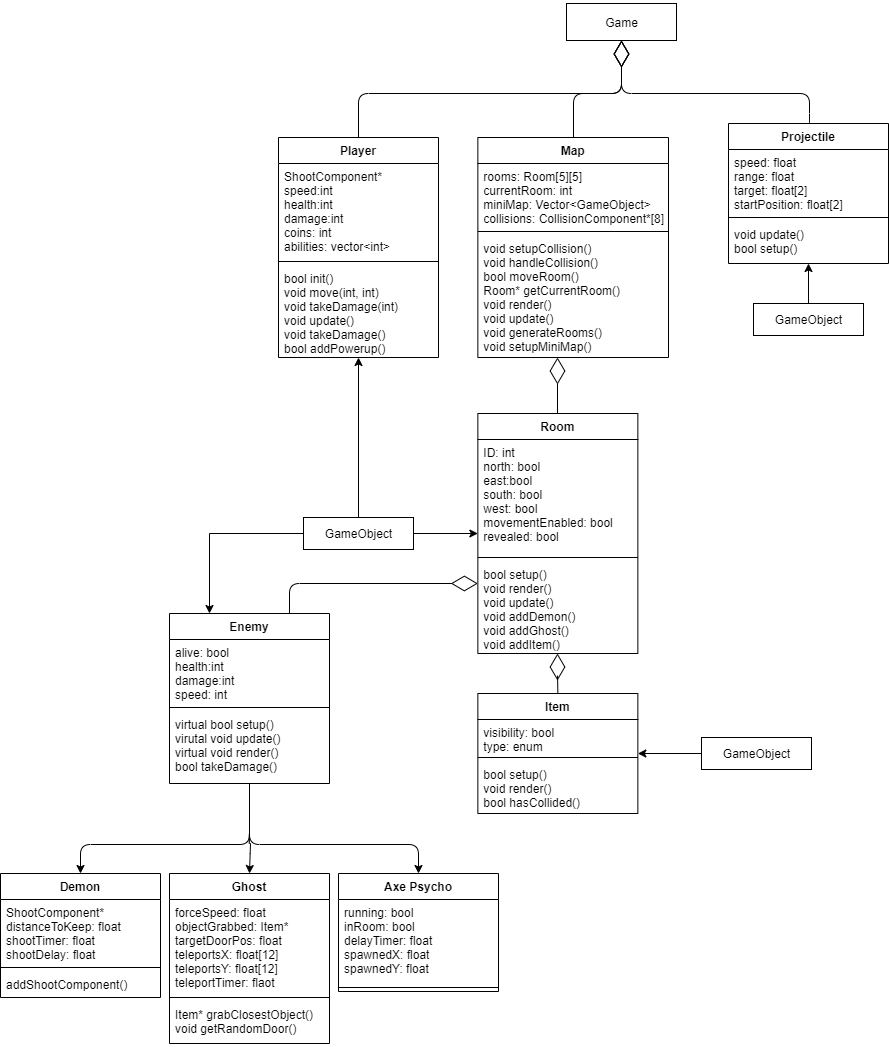
### 3.2.1 Game Object



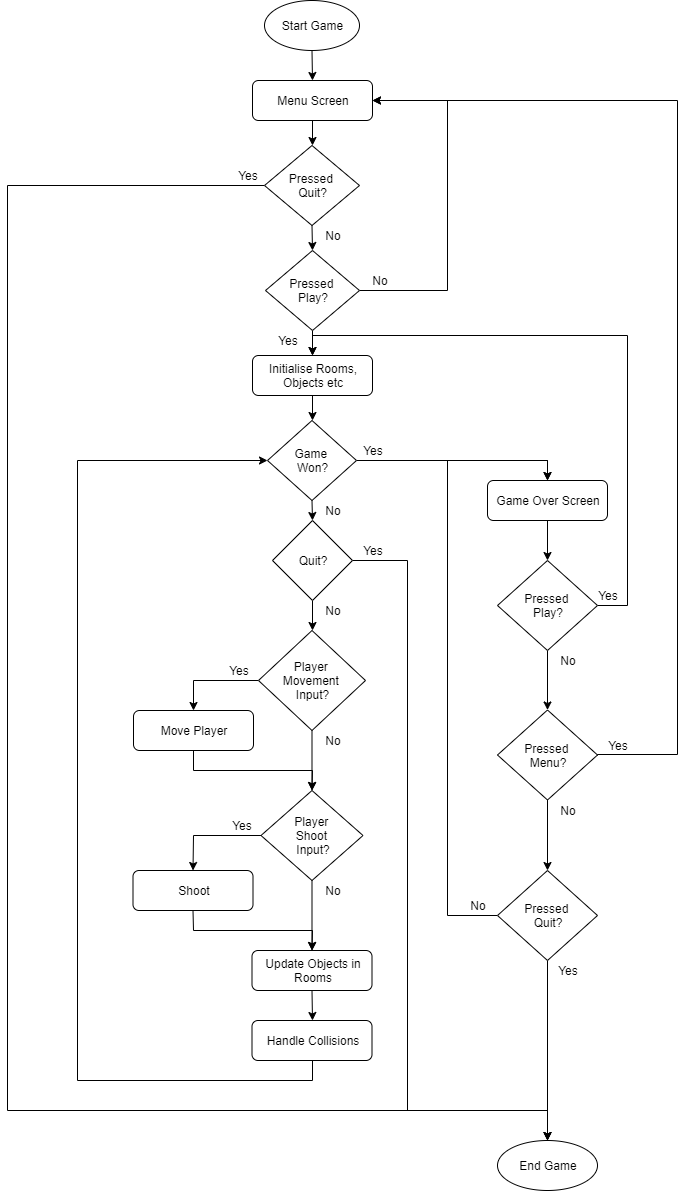
### 3.2.2 Scene Manager



### 3.2.3 Main UML



## 3.3 Game Flow



## 3.4 Game Objects and Logic

***Room***

*All rooms will have the same shape, with different colours and themes, meaning they will all have the same simple collider. Rooms will be randomly generated on each floor, from a set of stored sprites. Items will be randomly generated into each room for the player and the enemies to interact with.*

***Item***

*Items will either be crates in the rooms, which the player cannot interact with, or collectibles to help the player advance through the floors of the hotel, such as coins or hearts.*

***Projectile***

*A projectile will constantly move in it’s given direction for a certain distance, or until it hits an object. After a set distance has been travelled, or if the projectile hits an object, the projectile will be destroyed and the object it hits will be updated.*

***Enemies***

*Enemies will be spawned at random times throughout the game. The number of enemies spawned will be determined by the floor the player is on, increasing difficulty as the game continues.*

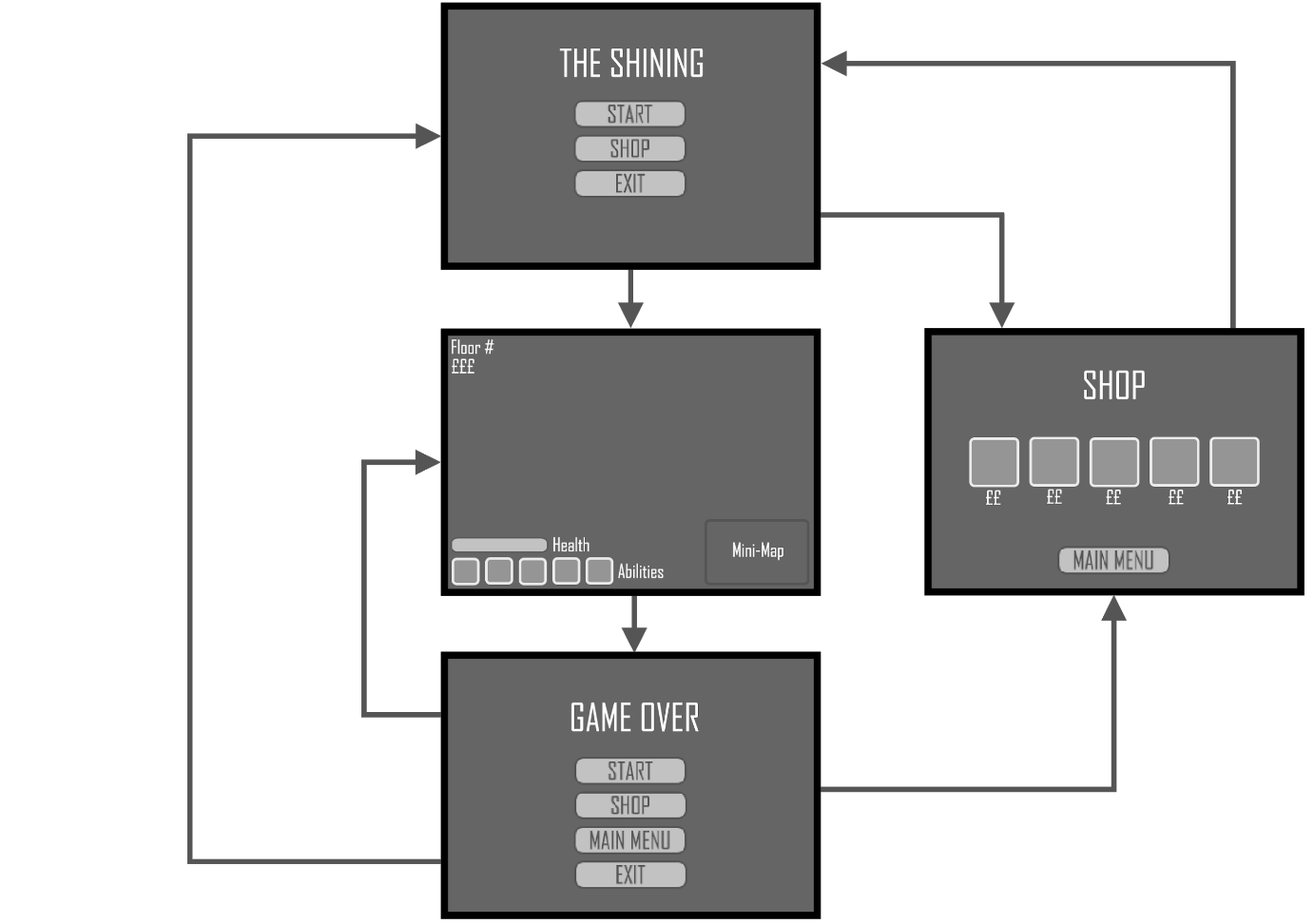
***Scene Manager***

*The scene manager will render the game scenes and will update UI items as needed.*

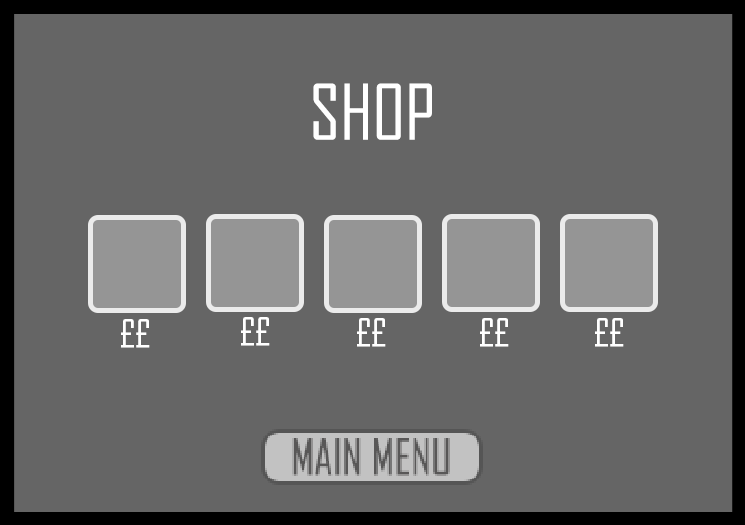
# Section 4 – UI and Controls

*The game will have a resolution of 1056px wide by 672px tall as this is the size of a single room sprite and a suitable size for PC monitors.*

## 4.1 Screen Flow



## 4.2 Game Shell

*  *

*The non-gameplay screens of the game are the Main Menu, Shop and the Game Over/Won screens. Users will select the option they want by moving the left analog stick around the screen and then pressing the A button to confirm their choice. If using a keyboard, the mouse and left click button will be used.*

***Main Menu***

*The main menu displays the game title and 3 options:*

* ***Play:*** *Starts the game*
* ***Shop:*** *Goes to the shop*
* ***Quit:*** *Exits the program*

***Shop Menu***

*The shop menu will display the possible power ups and abilities the player can buy:*

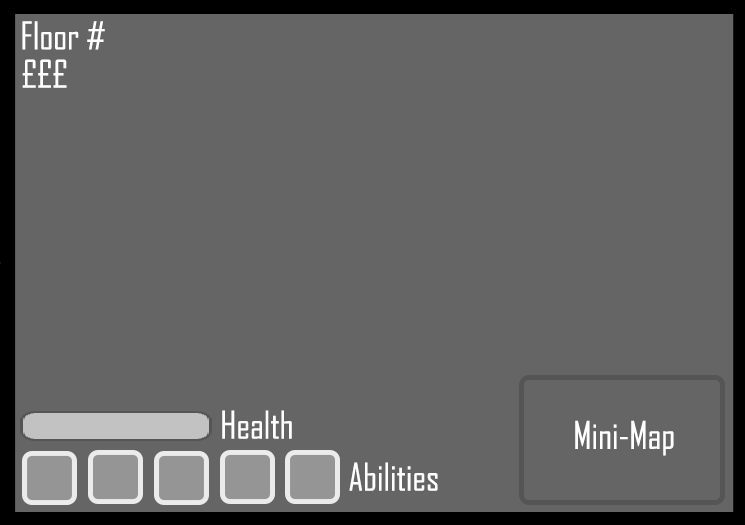
* ***Abilities:*** *The player can select these using the analog stick and can choose to buy one with the A button, if they have the right amount of money.*
* ***Main Menu:*** *Takes the player back to the main menu screen.*

***Game Over***

*The game over menu will have the following options:*

* ***Play Again:*** *Will reset the game, allowing the player to play again*
* ***Shop:*** *Goes to the shop*
* ***Menu:*** *Goes to the main menu*
* ***Quit:*** *Exits the program*

## 4.3 Play Screen

**

***Main Game Screen***

*The main game screen will show the current room the player is in. The UI will include floor number, coins, health, current abilities and the mini map.*

## 4.4 Controls

*The game will use ASGE’s built in controller support to control the player. The game will be compatible with both controller and keyboard.*

### 4.4.1 Controller

***Left Analogue Stick:*** *Move Player*

***Right Analogue Stick:*** *Aim Direction*

***Right Bumper:*** *Shoot*

***A (Xbox) / X (PS):*** *Select*

### 4.4.2 Keyboard

***Arrow Keys:*** *Move Player and Aim Direction*

***Spacebar:*** *Shoot*

***Left Mouse Button:*** *Select*

***Escape:*** *Close Game*