|  |
| --- |
| HALT gAMES PRESENTS |
| Blizzard the Wizzard |
| Technical Specification |
|  |
| **Hamish Carrier, Arran Ford, Liam Doherty, Timothy Veletta** |
| **Version 0.01** |

|  |
| --- |
| [Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.] |

Table of Contents

[Document History 2](#_Toc339434532)

[Engines 3](#_Toc339434533)

[Rendering Engines 3](#_Toc339434534)

[Level Engine 3](#_Toc339434535)

[Object Engine 3](#_Toc339434536)

[Character Engine 3](#_Toc339434537)

[Artificial Intelligence Engine 3](#_Toc339434538)

[Control Loop 3](#_Toc339434539)

[Game Shell 3](#_Toc339434540)

[Multimedia Engines 4](#_Toc339434541)

[Audio Engines 4](#_Toc339434542)

[Tools 4](#_Toc339434543)

[Data Conversion Tools 4](#_Toc339434544)

[Content Creation Tools 4](#_Toc339434545)

# Document History

|  |  |  |
| --- | --- | --- |
| Version | **Date** | **Changes** |
| 0.01 | 29-Oct | Added:   * Engines * Tools |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Engines

## Rendering Engines

Rendering is handled within the classes that inherit the abstract class World. In our program, a world contains all the functions that OpenGL calls back to such as Display, all the keyboard and mouse callbacks and the reshape and idle functions.

Hence Shays code became its own world. The other worlds that we have are the game world, which handles all the game related callbacks and also menu world which handles the menu.

## Level Engine

Since there is only one level it is handled directly in the GameWorld class. The Island model is loaded in and collision is handled by creating invisible objects that when the player or enemy collides with them, it restricts their movement. The outer boundary of the island model is handled by the TerrainModel class which allows other classes to pass in an X and Z value and it returns whether or not that point is on the island.

## Object Engine

Objects are handled by the ObjectManager class. The ObjectManager allows other classes to add game objects (GameObj class) to the World and then it takes care of updating those game objects. The object manager also allows external classes to delete objects given their identification number or the objects themselves can set a flag that they should be deleted before the next update.

## Character Engine

Engines to do with the manipulation of properties of the characters in the game.

## Artificial Intelligence Engine

The Artificial intelligence in game uses a templated finite state machine, each AI object owns an instance of the finite state machine class which handles entering and exiting states. The states are singletons that can be accessed by any AI objects that require that behaviour. All states inherit from an abstract state class that has the enter, exit and execute methods that are used by all states. The AI objects inherit from a parent AI class that contains functions and variables required by all AI objects.

## Control Loop

Control loop is managed primarily in the main.cpp file since it sets all the callbacks for OpenGL and then calls glutMainLoop() to start the program loop. The different worlds are also brought together in this file and are loaded when they are first used. They can be switched by setting the ‘currWorld’ parameter, from the World abstract class, to the desired new world.

## Game Shell

The game by default launches into the menu world and then allows the player to start the game. The game only runs on Windows since the model importer relies on some Windows platform specific libraries to work.

## Multimedia Engines

There are no videos within the game so no Multimedia engine was required.

## Audio Engines

The Audio in the game is run through the FMOD library, Each object that requires sounds owns an instance of the sound class, the sound class consists of custom functions for the adding and playing of sound files this streamlines access to the FMOD functions required by the program..

# Tools

## Data Conversion Tools

The model importer uses a library called GLM for loading in Wavefront .OBJ files which is provided in the examples folder of the GLUT source distribution although the base library does not support model texturing. The version of GLM we are using was modified by Tudor Cerean at [www.3dcodingtutorial.com](http://www.3dcodingtutorial.com) to support texturing on models.

There is also a RAW file importer used for the menu.

## Content Creation Tools

There are no content creation tools available for the game.