

**Development Document for:**

**Murdoch and Monsters**

**M&M**

“Something funny here or not so funny!”™

All work Copyright © 2021 by Team Mysterios

Written by all members of Team Mysterios

Version # 1.0

Friday, October 29, 2021

# Introduction

This document outlines the issues encountered and their solution, if any, during the development of the game. Along with its special features, maintenance issues and solutions, testing and possible improvement for the project.

# Development

This section covers what issues were encountered and how they were resolved, along with how code reuse is utilised and how the assets can be reused within other programs.

|  |  |
| --- | --- |
| Issue | Solution |
| As there was too many abilities to count and we wanted the game to be future proof we couldn’t hard code the abilities | We added a DataHandler and .csv files to input all the data and be able to change the data without needing to change hard code.  In this way and for a lot of other data and data input code, we have made it reusable. |
| When forcing the camera’s vertical angle it would glitch and not set. | We had to change the coding of shay’s and include restrictions which will make further use of the camera easier. |
| We couldn’t add textures to an OpenGL 3D Cube. This was a problem for the door. | To solve this changed the doors into planes which made the code reusable as well for anyone needing rotating or sliding doors. |
| We couldn’t constantly use OpenGL to create objects as it would take a long time and wouldn’t be reusable | To solve this we made an OBJ Reader and therefore made all of our models reusable. |
| The sound code from Shay’s World was heavily bugged and couldn’t work with external sound files properly | We edited the file so that we could add our own sound and play it when needed. |
| Shay’s World didn’t come with any UI. That made it hard to make the game in a pokemon-esk style. | So we added a library and made a customisable UI for all the controls in the game. This code was reusable with small tinkering. |
| We wanted all our data to be able to be accessed and added to if needed | To do this we add code to input the data when needed. So we just made each file and model accessible and therefore easily reusable. |

Reusing code and models came by copying them into the code/file you wanted and altering them slightly to connect to your new code. Most of the time it was that simple but any problems could be resolved fast and easily.

# Special Features

## Externally Customise maps

Maps are read from a csv file that can be modified to include more maps or edit existing ones. The only limitation is that it must follow the current format; level number, number of columns, number of rows, and on a new line start plotting the map with 0’s and 1’s for empty floor and obstacle, respectively. You can also edit the enemies that can appear on a level and how many.

## Externally Customise classes

They are read from a csv file which can be modified with more classes and/or altered existing classes. Also linked with csv of abilities.

## Externally Customise abilities

They are read from a csv file which can be modified with more abilities and/or different functionality (e.g. more damage), linked with classes csv.

## Externally Customise enemies

They are read from a csv file which can be modified with more enemies and/or altered existing classes. Also linked with csv of abilities.

## Adjustable UI

The UI menus can be moved to different positions along with the ability to collapse some.

## Basic pathfinder AI

Enemies find the shortest path to a player and traverse towards them.

## Multiple protagonists

When creating players you can save several to the game giving the user control of a party (multiple protagonists)

## Procedurally generated vegetation

The vegetation in the bush court (pre lobby) area will be different every time the game is launched.

## .obj file loader

Allows the program to read .obj model files which are easier to texture with OpenGL and are more commonly used. Allows players to easily add low poly models into the game.

# Maintenance

What issues arose when you were doing the maintenance part of the assignment? How were these issues resolved?

|  |  |
| --- | --- |
| Issue | Solution |
| Balance | We changed the .csv files to make some of the player's abilities and HP better to make users feel like the game is winnable. |
| UI Glitches | We had to alter the UI to stop the UI from disappearing. |
| Vector out of bound | We made bound checks before using the vectors. |
| Model quality | We increased the texture and model quality for a better experience of the game. |
| AI not moving | We added pathfinding to reduce the chance of AI getting stuck in corridors. |

# Testing

Initial testing was to have a complete run of the entire game from level 1 to 10. This was done via a developer’s cheat character which we named god that would defeat enemies easily for fast progress through the game. This tests the main aspects of the game:

* To find out if it is possible to run through the entire game
* To find out if the enemy AI can function
  + To find out if the AI will move towards the player
  + To find out if the AI would attack the player
* To find out if entities can be damaged
* To find out if the walls are operational
* To find out if enemy/player deaths removes the entity off the board
* To find out if the random room generator works
* To find out if the random enemy spawner works
* To find out if melee, ranged and AOE abilities work.
  + Godpunch, SuperGod and GodAOE
    - Specialized abilities to testability types and to allow for easy clearing

These are run over 10 times in a God run to check if the main aspects of the game stays consistent.

After confirmation that the main game loop and basic implementation are operational, we then test every character for their given ability and if that ability is capable of function. This is run with a party of 4 of the same character type and running the game. Using the same skills tests the consistency of the skill as well as tests if the ability is operational.

After confirmation on all abilities and player archetypes, a regular game is played with 4 different random characters. This is to test difficulty and to test the player’s experience as the focus of this is to see if the information is given to the player easily and to determine how difficult the game is currently and if any changes are needed to help balance the game.

# Possible Improvements

The biggest improvement that should be made is in balancing the stats between the player characters and enemy characters. Due to the lack of enemy progression, earlier enemies are much harder than the level that the player characters can handle. This leads to an early death, and that isn't that fun for anyone, especially when most aspects of the game has an emphasis on RNG due to its similarity to a normal game of DnD. A counter to this is that we have not made a cap for the number of characters in the player’s party, which allows the player to win mostly by having more bodies to sacrifice.

This leads to another problem, in that having a party size larger than the map’s grid leads to the game crashing due to the excess characters having no position to be stored in leading to a crash. This is a problem that can be easily fixed, but the problem is that we don't know what size would be fair enough that the players could have a fair chance to win yet remain a challenge for the player.

Another big improvement in the game would be the addition of light and sound. Due to the lack of time and other responsibilities in life, we were unable to implement light and sound.

There was no issue in the team. We all had our parts to play and we did it well.