141 CW 2 Report

What the program is and how to use it

The program is (very loosely) based on Call of Duty zombies, where you have to survive rounds of increasing enemies. Every new level will have more zombies (red squares). You play as a cyan square and shoot blue bullets to defeat enemies. Each enemy will have a different amount of health, size and speed. It will take more bullets to destroy the larger enemies. The health of the enemies are denoted by the darkness of their colour, so the darker the enemy, the more health it has.

The player starts with 100 health, which can be regenerated by touching green squares (supply packages) which spawn in every 5 levels.
Bullets deal 20 damage upon impact and enemies deal 5 damage upon contact.

The primary objective of the game is to survive as many levels as possible.

The player is controlled by the arrow keys and bullets are fired by aiming the mouse and left clicking. The game is started by pressing ‘p’, restarted by pressing ‘r’ and paused by pressing ‘p’.

Any architectural choices made

Files

The program is separated into 3 files:

1. The Main.hs file which contains most of the functions to run the program. These functions all require inputs.
2. The Config.hs file which contains all of the constant functions used in the program. These functions do not require inputs.
3. The Type.hs file which define all of the data structures and types used in the program.

Entities

Entities were designed to be like the Java “abstract class”, so they contain certain values that all entities have, but in the EntityInfo record there is information that relates specifically to certain types of entities. This was so that all entities could be handled in very similar fashions, but could be differentiated when it came to rendering and interractions.

updateFunc

Lots of functions used by updateFunc were defined within the function as it allowed for the world variable to be accessible by all of the functions defined inside updateFunc. This allowed the code that was written to be a lot more flexible and less records had to be passed in to functions, meaning that there would be less overhead.

The primary library used

Gloss was used in this project to render and update the game.

Personal experience when writing the program

I have done some game development in Java before, so I already knew about the processes and requirements of game development before. It was difficult not having access to global variables, but the world record could be treated in a similar way to global variables in the updateFunc.

It was difficult to find any in depth tutorials in developing games in Haskell, so it was useful to look at game development tutorials in other languages and the knowledge could be somewhat translated when writing the program.

Resources used

Python game development tutorial
https://www.youtube.com/watch?v=FfWpgLFMI7w

Blog on making a game using Gloss
https://mmhaskell.com/blog/2019/3/25/making-a-glossy-game-part-1

Hake
https://github.com/dixonary/hake