Design Doc – CA1

For CA1 of our gameplay programming module we were required to make a text based game which included character classes inheriting from a base character class with a game class and a main to run the game. This is what I did for this assignment.

## GameCharacter

To begin with the gamecharacter class was made. I used this for the enemy and player classes which I will talk about later. I started with the header file with included all the protected and public variables that were in use for the cpp file and even for some functions in the game file. A computer screen shot of a program

Description automatically generated

These are the public functions I created. Basically what these do is allow the functions to be inherited throughout the project by inheriting this header file. Spawn here passes through all of the core variables for the characters that make them unique like health, id etc. Then render which is used in the game file prints id and the co ordinates for the character made in the game. Update is a virtual function here which is overridden in enemy and player files to make their own unique update function. This basically means that this function isn’t set for all classes, it can be overridden to suit whichever class its being used for. Next was stats which printed all of the characters variables to screen and isAlive was used to check whether the character died in battle or not.

A screen shot of a computer

Description automatically generated

The private variables in this class were made for the characters base stats so their id, health, speed and their co ordinates. These are all changed later on in the game for each spawned character.

A screen shot of a computer program

Description automatically generated

Thes are some new functions not on the doc that I created for the game file. These are all just getters and setters for the objects variables.

A computer screen shot of text

Description automatically generated

In the cpp file then these functions made in the header file are defined. For spawn it sets the variables to allow them to be altered later on. This is the constructer.

A black screen with text

Description automatically generated

The other functions then, render prints out the id and co ordinates, update is virtual so it isn’t defined here its just used for inheritance purposes. Stats prints all of the variables and isAlive checks whether the hp is 0 for the object.

A screen shot of a computer program

Description automatically generated

These are the new functions being defined the getter and setter methods.

## Enemy

A screen shot of a computer

Description automatically generated

Next was the enemy header file all that’s added here is its inheriting the update function to be altered in the cpp file.

A computer screen shot of a program

Description automatically generated

In the cpp file we were tasked with moving the enemy up and down. For this I used a random to see if the enemy would actually move or not. If the random number was not above 8 and not below 0 the enemy would then move. I just used a for loop here to increment the y value for the enemy and printed that the enemy moved or hit a wall.

## Player

The player header looks the exacts same as the enemys header it inherits from gamecharacter and includes the update function.

A screenshot of a computer program

Description automatically generated

In the cpp we made the player so they could move north, south, east or west and depending on the input the player would have their x or y values incremented accordingly and checked if they hit a wall and took damage.

## Game

A screen shot of a computer program

Description automatically generated

For the game header file I inherited from all the previous headers and included the list. In here I created all the objects and also created the list for the objects to be added to. The public functions then included the functions to be altered in the game cpp file. This included some new functions like init, battle and clean.

A computer screen shot of a program code

Description automatically generated

First in the cpp I inherited the header and then I defined the init function. In here it initialised all of the objects, their variables and adds them to the list. I used arrays for the enemys variable to condense the code. Then I used spawn to assign the variables values. Then the pushback was used to add the objects for the list.

A screen shot of a computer

Description automatically generated

Next for render this searches the list created above and uses the render function in gamecharacter to print the id and co oridnates to the screen.

A screenshot of a computer program

Description automatically generated

For battle then first the for loop sorts through the list using the auto function to check for which object using iterators. Then in the second for loop it searches through the objects using 2 iterators. Below in this loop I set the characters to these iterator variables so for the 2 characters that will do battle. Then I had an if to check if the positions are the same using a new get co ordinates function. Under that then I used the isAlive function to stop the battles from looping over and over. The the hp was checked to see who won the battle and it prints the results.

A screen shot of a computer

Description automatically generated

Stats is just used to print all the stats for the objects using similar code to render.

A screen shot of a computer

Description automatically generated

Update is called here implementing that function for the objects

A black screen with green text

Description automatically generated

The clean function is used to erase the objects when they are killed.

## Main

A screen shot of a computer program

Description automatically generated

In main the game object is created and init is called for this object. Then the games functions are called and set to loop 100 times. Finally after that loop it prints the stats.