Spike: 3
Title: Game State Management

Author: Ryan Chessum, 102564760

Goals / deliverables:

- Code, see /08 – spike - Game State Management/Task 8 Zorkish States/
- Simple plan/design outline for the program
- Spike Report

Technologies, Tools, and Resources used:
List of information needed by someone trying to reproduce this work

- Visual Studio 2019
- C plus plus reference (https://www.cplusplus.com/reference/)
- Zorkish game specifications

Tasks undertaken:

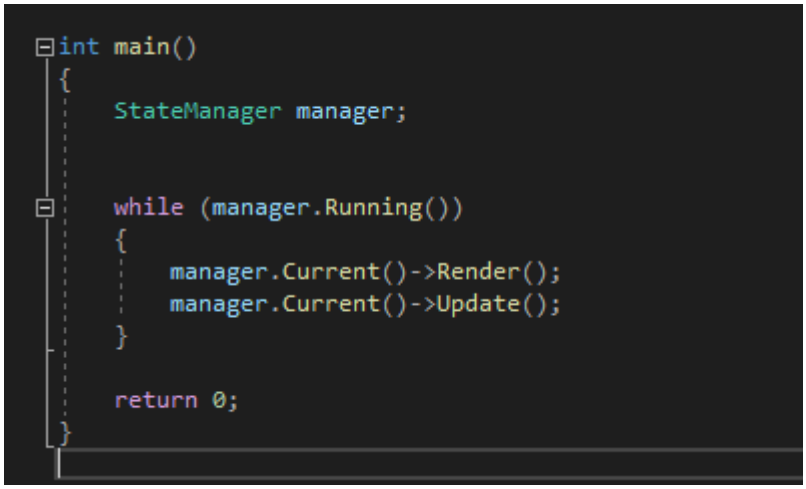
- Download and install Visual Studio
- Create a new C++ project
- Create a basic design for the code
- Create a basic game loop structure
- Create a state class
- Create an enum for all the states
- Create a state manager to change the behaviour of the program by calling the update and render functions of different states.
- Create state subclasses for every state and define the behaviour of their update and render functions

What we found out:

A game state manager is a great way of managing the states or stages of a game. States can help us determine what a game should be doing. For example is the player in a menu, playing the main game, etc. A class based state manager system is an efficient approach for implementing a state manager.

To determine the current state of the game we can create a state manager class. Which will get the current state and call update and render functions that we need for each state/stage. This also can help us with code organisation and abstraction.

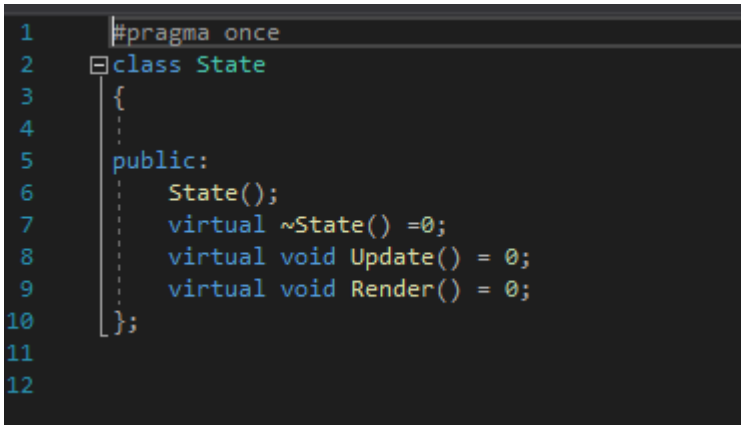
Here is what the main function looks like using a state manager.



```
int main()
{
    StateManager manager;


    while (manager.Running())
    {
        manager.Current()->Render();
        manager.Current()->Update();
    }

    return 0;
}
```

It’s a very simple looking game loop. What’s happening is we have an instance of a state manager which is filled with state objects. Update and render are called by getting the current state of the state manager and calling the member function for update/render of that state.

For this to work we need a base state class. It should be very basic having only update and render functions. We are going to use this as a parent class and the children of this class will have definitions for Update and Render. So, each function should be virtual so they can be overridden in the child states.

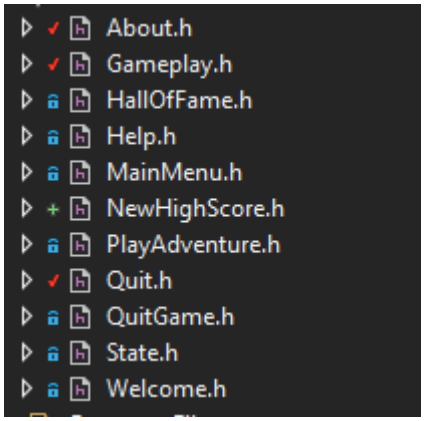


```
1   #pragma once
2   class State
3   {
4
5   public:
6       State();
7       virtual ~State() =0;
8       virtual void Update() = 0;
9       virtual void Render() = 0;
10  };
11
12
```

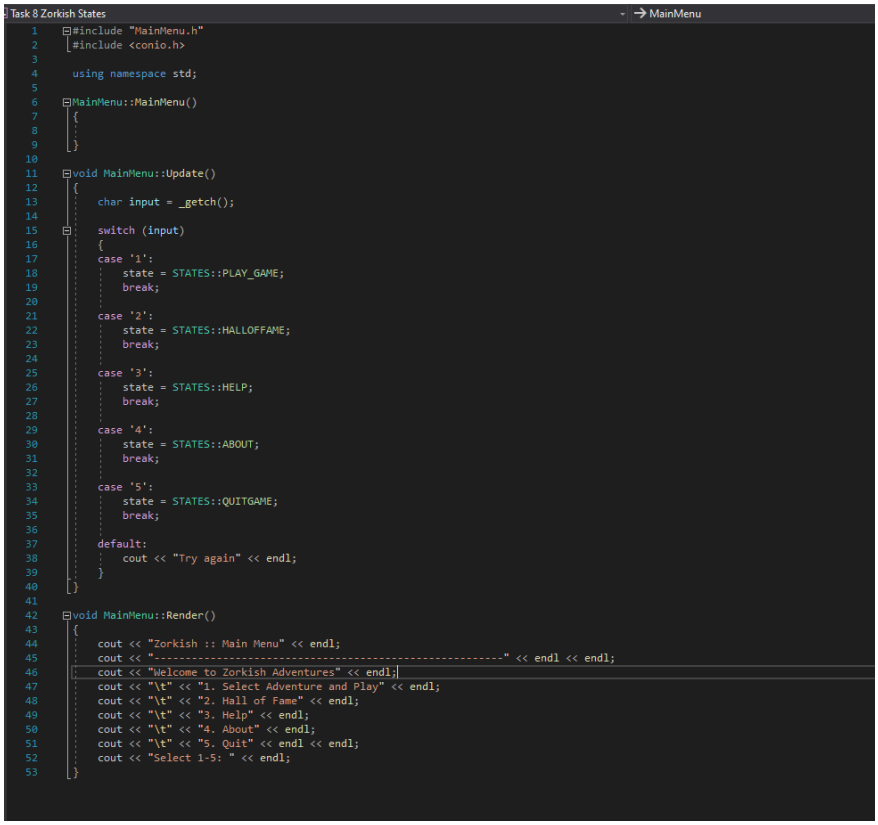
Next create an enum containing each state we want to add. As well as a global variable of the type of your states enum for the current state. This is what is used to change/get the game state. Note that the state variable has to be defined as an external variable in each file so that each file knows that it exists.

```
#pragma once

enum class STATES {WELCOME, MAIN_MENU, PLAY_GAME, HELP, HALLOFFAME, ABOUT, GAMEPLAY, NEWHIGHSCORE, QUIT, QUITGAME, DONE};

```

Then we create a child class for each state we would like to have in our game. Override the update and render functions and define the behaviour for each one.



As an example, here is the main menu state. Render lists the options the player has on screen while update collects an input from the player and updates the global state based on the input. You can define whatever you want in there but keep in mind you probably want to have a way of going back to the main menu, even if it’s going to a state that can take you there.

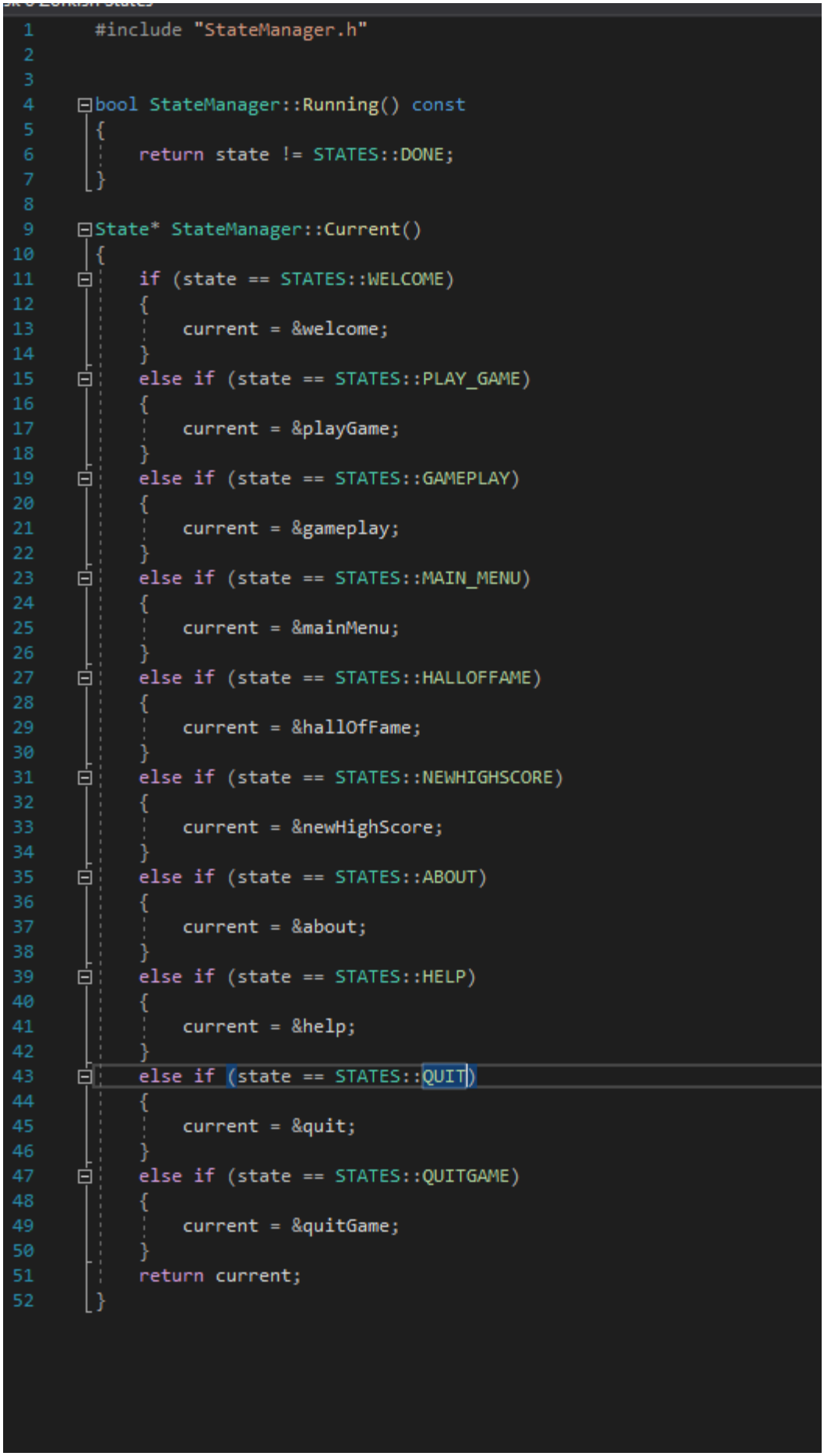


Once you have all of your states, we need a state manager to hold and manage them all. The state manager class should look something like this. It should have instances of all of your states, a state (class not enum) pointer that points to the current state and a function that can give the current state.

Task 8 Zorkish States

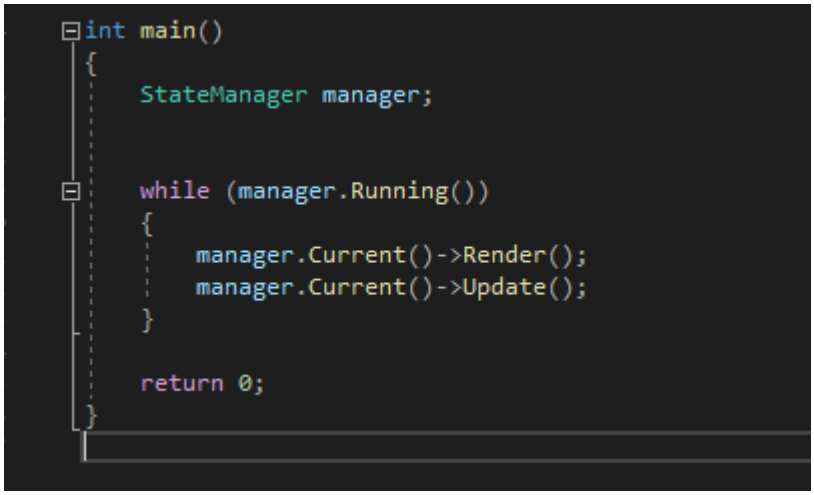
```
1   #pragma once
2   #include "State.h"
3   #include "Welcome.h"
4   #include "MainMenu.h"
5   #include "states.h"
6   #include "Help.h"
7   #include "PlayAdventure.h"
8   #include "Gameplay.h"
9   #include "Quit.h"
10  #include "HallOfFame.h"
11  #include "About.h"
12  #include "QuitGame.h"
13  #include "NewHighScore.h"
14
15  extern STATES state;
16
17  class StateManager
18  {
19  private:
20      //states here
21      Welcome welcome;
22      MainMenu mainMenu;
23      PlayAdventure playGame;
24      Gameplay gameplay;
25      Help help;
26      HallOfFame hallOfFame;
27      About about;
28      Quit quit;
29      QuitGame quitGame;
30      NewHighScore newHighScore;
31
32      State* current;
33      bool running = true;
34
35  public:
36      StateManager() {};
37      ~StateManager() {};
38      bool Running() const;
39      State* Current();
40  };
41
42
```

The Current function should look like this. Depending on the state set on the global variable it will change the current variable to point at the corresponding state.



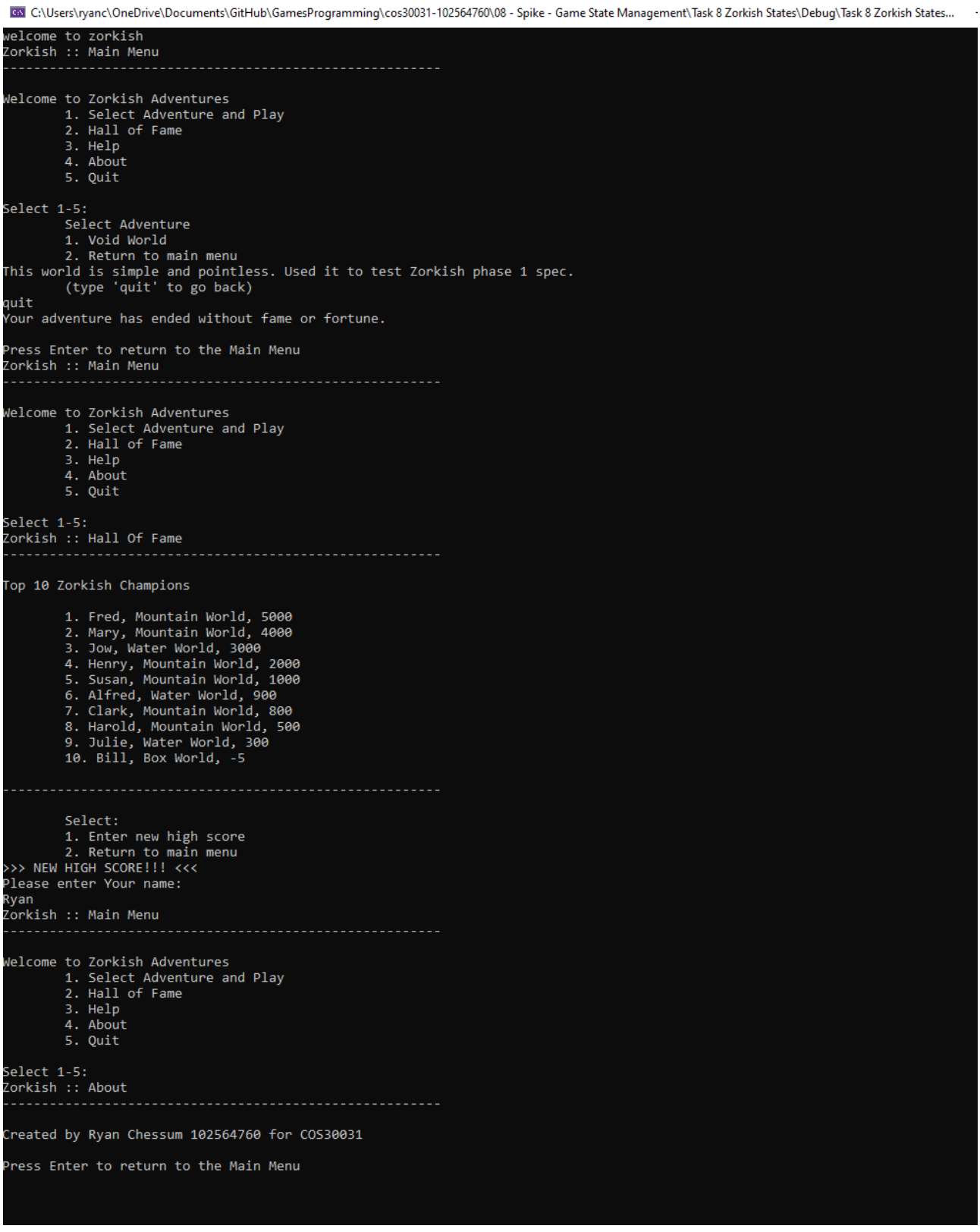
```cpp
#include "StateManager.h"


bool StateManager::Running() const
{
    return state != STATES::DONE;
}

State* StateManager::Current()
{
    if (state == STATES::WELCOME)
    {
        current = &welcome;
    }
    else if (state == STATES::PLAY_GAME)
    {
        current = &playGame;
    }
    else if (state == STATES::GAMEPLAY)
    {
        current = &gameplay;
    }
    else if (state == STATES::MAIN_MENU)
    {
        current = &mainMenu;
    }
    else if (state == STATES::HALLOFFAME)
    {
        current = &hallOfFame;
    }
    else if (state == STATES::NEWHIGHSCORE)
    {
        current = &newHighScore;
    }
    else if (state == STATES::ABOUT)
    {
        current = &about;
    }
    else if (state == STATES::HELP)
    {
        current = &help;
    }
    else if (state == STATES::QUIT)
    {
        current = &quit;
    }
    else if (state == STATES::QUITGAME)
    {
        current = &quitGame;
    }
    return current;
}
```

Now that you have your state manager go back to main and create your gameloop by accessing the current state and calling the update and render function through the pointer.



```
int main()
{
    StateManager manager;


    while (manager.Running())
    {
        manager.Current()->Render();
        manager.Current()->Update();
    }

    return 0;
}
```

If done correctly everything should work.


C:\Users\ryanc\OneDrive\Documents\GitHub\GamesProgramming\cos30031-102564760\08 - Spike - Game State Management\Task 8 Zorkish States\Debug\Task 8 Zorkish States...
welcome to zorkish
Zorkish :: Main Menu
Welcome to Zorkish Adventures
1. Select Adventure and Play
2. Hall of Fame
3. Help
4. About
5. Quit
Select 1-5:
Select Adventure
1. Void World
2. Return to main menu
This world is simple and pointless. Used it to test Zorkish phase 1 spec.
(type 'quit' to go back)
quit
Your adventure has ended without fame or fortune.
Press Enter to return to the Main Menu
Zorkish :: Main Menu
Welcome to Zorkish Adventures
1. Select Adventure and Play
2. Hall of Fame
3. Help
4. About
5. Quit
Select 1-5:
Zorkish :: Hall Of Fame
Top 10 Zorkish Champions
1. Fred, Mountain World, 5000
2. Mary, Mountain World, 4000
3. Jow, Water World, 3000
4. Henry, Mountain World, 2000
5. Susan, Mountain World, 1000
6. Alfred, Water World, 900
7. Clark, Mountain World, 800
8. Harold, Mountain World, 500
9. Julie, Water World, 300
10. Bill, Box World, -5
Select:
1. Enter new high score
2. Return to main menu
>>> NEW HIGH SCORE!!! <<<
Please enter Your name:
Ryan
Zorkish :: Main Menu
Welcome to Zorkish Adventures
1. Select Adventure and Play
2. Hall of Fame
3. Help
4. About
5. Quit
Select 1-5:
Zorkish :: About
Created by Ryan Chessum 102564760 for COS30031
Press Enter to return to the Main Menu

Ryan Chessum	a1e9d23	task 8 Quitgame state
Ryan Chessum	52aa45f	Task 8 about state
Ryan Chessum	8e4a488	Task 8 Hall of fame state
Ryan Chessum	6bccc17	Task 8 quit
Ryan Chessum	72c3b69	Task 8 dummy gameplay state
Ryan Chessum	9c3a8dc	Task 8 Choose adventure state
Ryan Chessum	3c9cf7a	Task 8 Help state implemented (state change ok)
Ryan Chessum	d37cea8	Task 8 Main Menu Complete
Ryan Chessum	704b874	Task 8 Basic states working
Ryan Chessum	848cdb4	Task 8 StateManager mostly set up
Ryan Chessum	ccc6a04	Task 8 Class files for states and state manager
Ryan Chessum	631616a	Task 8 Set up basic gameloop
Ryan Chessum	2163cef	Task 8 Project Setup