**Spike:** Spike No.8

**Title:** Task 8 – Game State Management

**Author:** Khang Trinh - 102118468

**Goals / Deliverables:**

The goal of this spike is to explore how implement game states to

**Technologies, Tools, and Resources used:**

* Visual Studio 2017

**Useful Links:**

**Tasks undertaken:**

## Step 1: Plan out the transition between states

It’s worth drawing out a diagram showing the relationship between the states so that it can help you figure out how to code their functionalities.

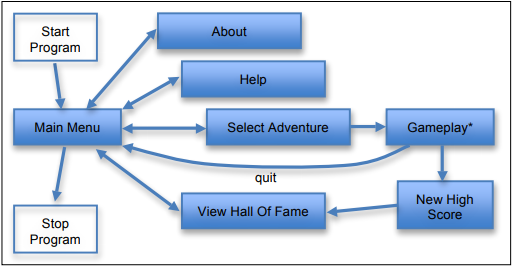


Fig . State diagram for Zorkish Adventures

## Step 2: Create an enum holding all the possible states in the game

If there’s a state in the game that will be doing something, they’ll need to be in this enum so that the game can switch to.



Fig . An enum holding all the states

## Step 3: Create a base class for the states

Every state the game changes to, it’ll do something different, and as a result would need to look different. Therefore, every state should have an update and render function. To do that the most efficient way, we’d need to create a base class with these functions that the states can inherit from. *Note that the update function has a return type of the enum, it’ll be explained.*

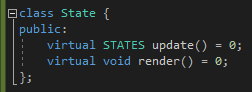


Fig . Base class for states

## Step 4: Create the states

Now that we have a blueprint to create the states, we can use that to add functionalities to all the states.

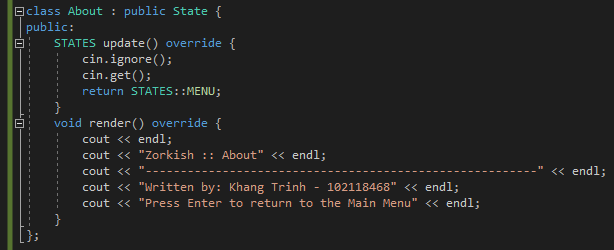


Fig . An example of a state that doesn’t do much other than display info, and proceeded by hitting Enter

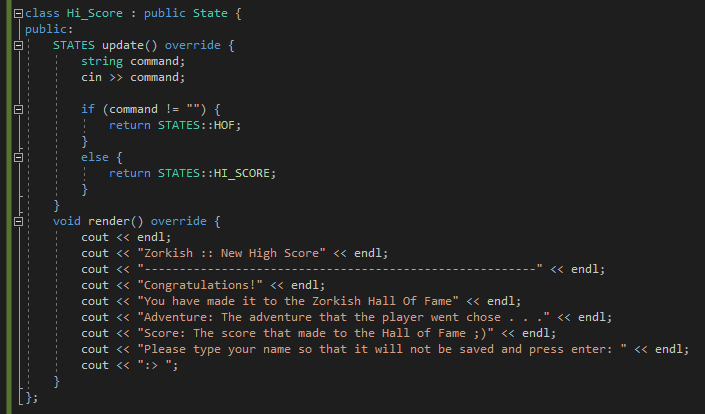


Fig . A state that takes any string input

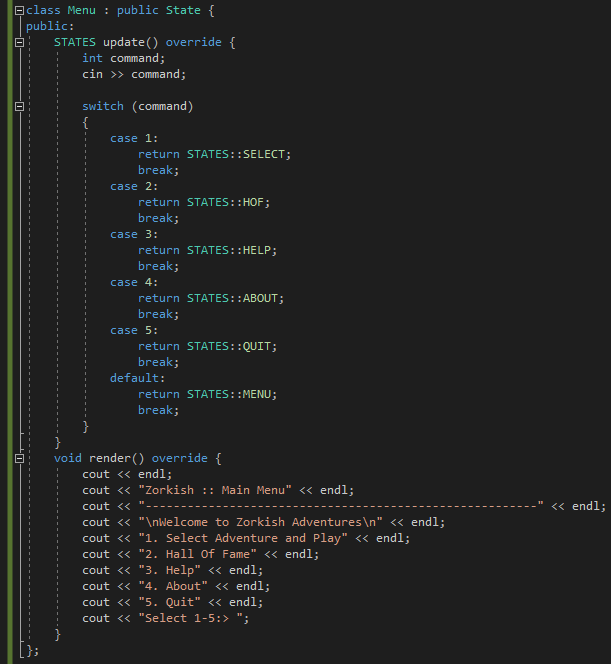


Fig . Sometimes it's worth using a switch statement to save time and effort

## Step 5: Create a state manager class

Now that we’ve created all of the needed states, we need to create a class to manage all of these states (called a manager class) to tell the game when it can switch to what state.

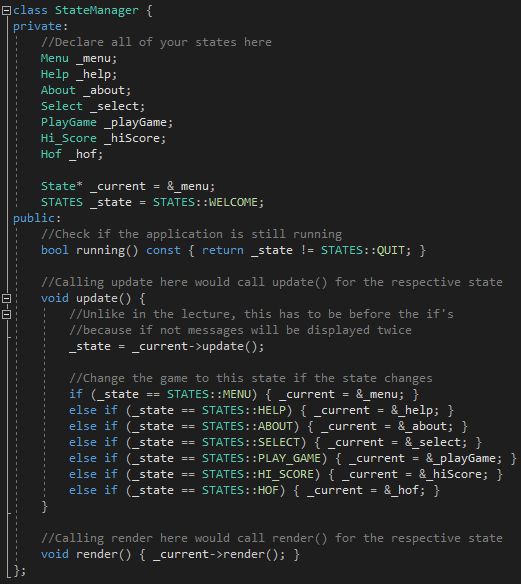
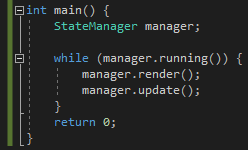


Fig . An example of a state manager class

## Step 6: Tie it all together

If you’ve done all the steps above correctly, main should be trivially easy to implement.



## Example output

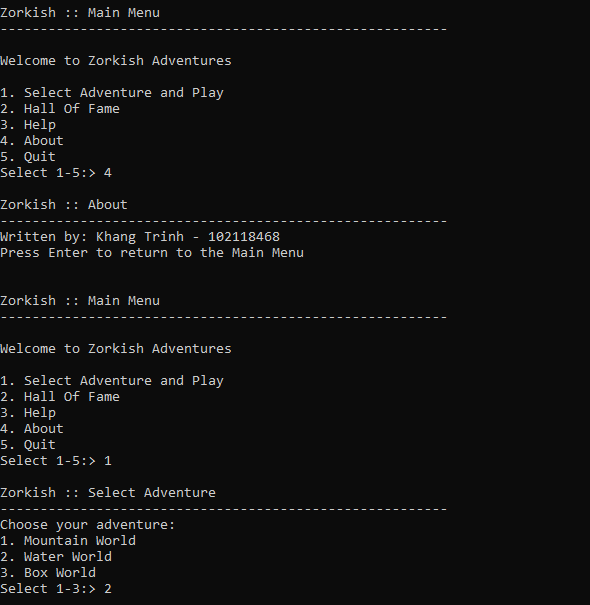


Fig . What the output should look like for Zorkish Adventures