**Spike:** Task 9

**Title:** Game State Management

**Author:** Sam Huffer, 101633177

# Goals / deliverables:

* A simple plan of the code design of phase 1 of the “Zorkish Adventure” game
* A console program that implements phase 1 of the “Zorkish Adventure” game, using a flexible and extensible game state management method of some kind, without gameplay yet. Game stages demonstrable at this stage must include:
  + Main Menu (to select other stages)
  + About (including the student’s own details)
  + Help (hard-coded summary of commands)
  + Select Adventure (hard-coded list and the title of the test game)
  + Gameplay (test game only accepting “quit” and “hiscore” commands)
  + View Hall of Fame (view hard-coded list of name/score)

# Technologies, Tools, and Resources used:

* Visual Studio 2017
* Microsoft Word
* Draw.io

# Tasks undertaken:

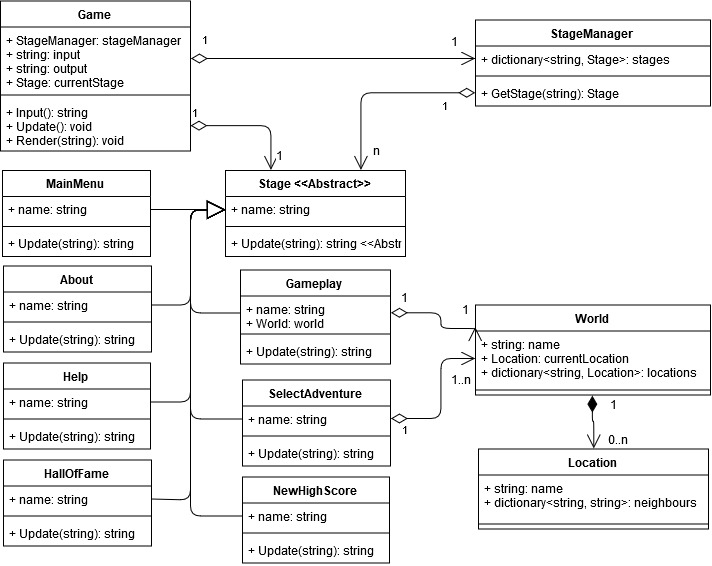
* Read through the spike instructions and the Zorkish phase 1 specifications to get the gist of what the program requires.
* Created a class diagram in draw.io to outline what, at this stage, I figured the program would require:

Figure : preliminary UML class diagram for phase 1 of the Zorkish Adventure game

* Created a Visual Studio project with .cpp files for each class, set up the main() method to create and run the Game class, and set up the basic Input(), Update(), and Render() loop for the game.
* Fleshed out the Game, Stage, MainMenu, and StageManager classes enough for a very basic test of the game’s functionality to ensure I had everything plugged in correctly. Upon running this version, I ran into a number of errors screaming about no or too many includes and definitions of files and classes.
* Reorganised the code to have class definitions in .h files and the method implementations in .cpp files and reworked the #include statements, and ran it again, but it still kept complaining about further errors.
* Did further research to sort out how to organise the files and use #includes and forward declarations properly (<http://www.cplusplus.com/forum/articles/10627/>), and came across other content giving good practice tips (<https://accu.org/index.php/journals/1995>). It took a while and some trial and error, but eventually I sorted out the #includes and forward declarations so that the game ran properly. While doing this, I also cleaned up some interactions so that the code displayed output properly, and did some setup for the implementations of Stage.Update() when it would be called for the first time after being assigned to Game.currentStage. Between errors and other classes, that took me from the Tuesday Tutorial until Saturday morning to complete.

# What we found out

* How to organise C++ code across multiple files, splitting definitions into .h files and implementations into .cpp files, and #include and forward declare classes from other files properly such that the program works and the compiler doesn’t scream at you about errors you have no idea how you caused them.