**Spike:** 05

**Title:** Game State Management

**Author:** Ford Killeen, 9731822

**Goals / deliverables:**

*Goals this spike aims to achieve:*

* Create a simple console application that implements Zorkish Phase 1
* Create an implementation with flexible game states and state management using OO State Pattern
* Implement the following states:
  + Main Menu
  + About
  + Help
  + Select Adventure
  + Gameplay
  + New High Score
  + Hall of Fame

*Deliverables required:*

* Code for the Zorkish phase 1 with flexible state management
* Hand written design for a state manager
* Spike report

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

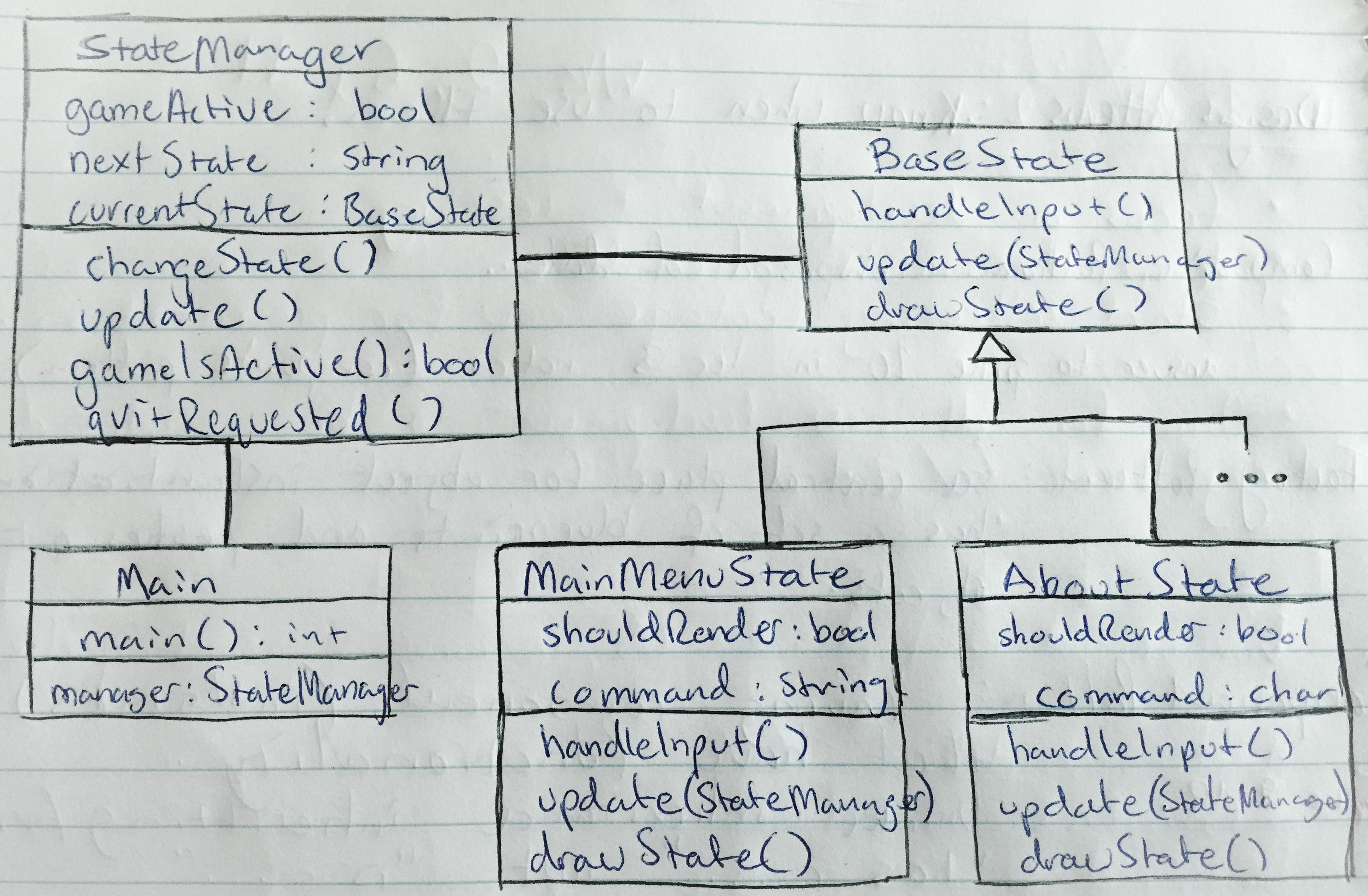
The following is required to complete this spike:

* Visual Studio 2015
* Zorkish game specification
* Online State Pattern references/examples

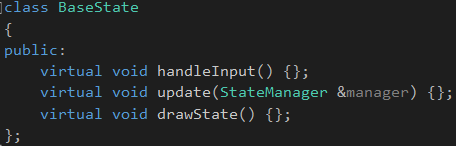
**Tasks undertaken:**

The list below details the steps taken to complete this spike.

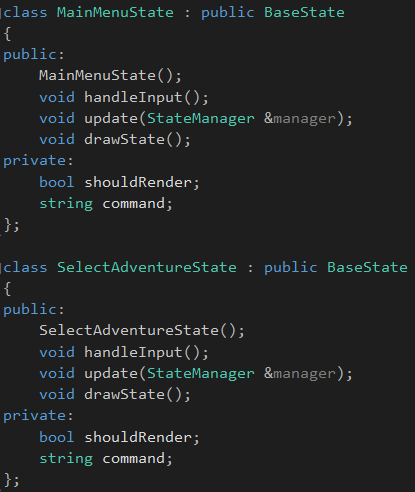
* Grab a pen and paper and make a simple design for the state management system
* Use any state pattern references or UML guides to complete your hand drawn design



* Begin coding your paper design, starting on your state manager and base state first



* With the foundation of your state manager ready, set up and create your MainMenuState, inheriting from your BaseState class
* Be sure to build your project often, fixing any little errors or bugs along the way
* Once happy with the MainMenuState, implement a second state that you can then swap between, checking that the StateManager is working as intended
* Finish off by adding the rest of your states and testing your solution after each addition



**What we found out:**

By completing this spike we found out how to implement a State Manager that allows very flexible state management and scalability. By using a StateManager object that contained a BaseState object, we were able to call the handleInput(), update() and drawState() method on the currentState regardless of what the current state of the game is.