Spike: 13
Title: Testing with Unit Tests

Author: Ford Killeen, 9731822

Goals / deliverables:
Goals this spike aims to achieve:
- Write a set of tests for the core functionality of a previous spike (Zorkish or Gridworld)
- Demonstrate that the tests can expose any introduced issues or problems with the evolving solution.

Deliverables required:
- Code showing off the unit tests and their results
- Spike report

Technologies, Tools, and Resources used:
The following is required to complete this spike:
- Visual Studio 2015
- Unit Testing online guides and references
- Online C++ references

Tasks undertaken:
The list below details the steps taken to complete this spike.
- The first thing I did was use the spike recommendations and hints to add the Unit Testing project to my existing Zorkish project.
- With this new testing project added and linked up to reference the Zorkish `.obj` files, I started writing the tests
- I decided to stick with the *CppUnitTest* framework as that is built into Visual Studio, so I didn't need to add anything new or learn any new testing frameworks
- Once I got a basic test working I went through and completed the rest of the tests for my first spike, spike 5 – *Game State Management*. The tests for this spike were aimed towards ensuring that the StateManager class was functioning as it should. They were put into the test class `StateTests`. Some examples are:

```
TEST_METHOD(GameActiveTest)
{
    StateManager* state_man = new StateManager();
    Assert::IsTrue(state_man->gameIsActive(), L"Game not active", LINE_INFO());
}

TEST_METHOD(GameEndedTest)
{
    StateManager* state_man = new StateManager();
    state_man->quitRequested();
    Assert::IsFalse(state_man->gameIsActive(), L"Game did not end", LINE_INFO());
}
```

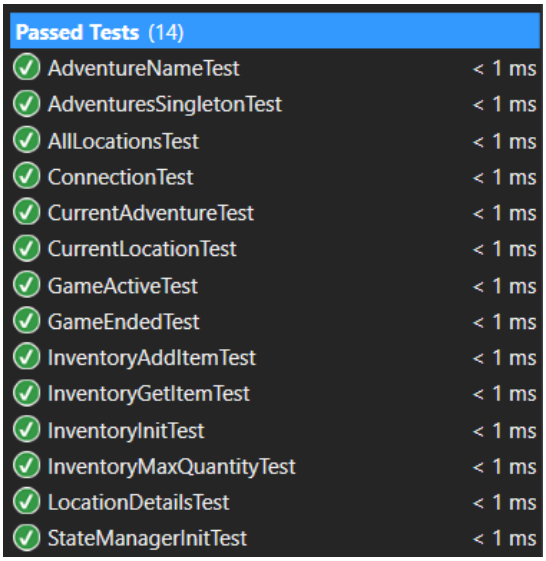
- I then moved onto spike 6 – *Basic Game Data Structures* and an inventory system. In this spike I added the tests from the previous spike and worked on adding another test class to handle all the tests related to the Inventory class, `InventoryTests`. Example tests are:

```
TEST_METHOD(InventoryAddItemTest)
{
    Inventory* inventory = new Inventory();
    Assert::IsTrue(inventory->addItem(new Sword()), L"Sword not added", LINE_INFO());
    Assert::IsTrue(inventory->addItem(new Key()), L"Key not added", LINE_INFO());
    Assert::IsTrue(inventory->addItem(new Sword()), L"Sword not added", LINE_INFO());
}

TEST_METHOD(InventoryMaxQuantityTest)
{
    Inventory* inventory = new Inventory(2);
    Assert::IsTrue(inventory->addItem(new Sword()), L"Sword not added", LINE_INFO());
    Assert::IsTrue(inventory->addItem(new Key()), L"Key not added", LINE_INFO());
    Assert::IsTrue(inventory->addItem(new Sword()), L"Sword not added", LINE_INFO());
    Assert::IsFalse(inventory->addItem(new HealthPotion()), L"Potion was added", LINE_INFO());
}
```

- I then moved onto spike 7 – *Graphs*. In this spike I added all of the tests from the previous two spikes and worked on adding the next test class, `AdventureTests`. Example tests are:

```
TEST_METHOD(LocationDetailsTest)
{
    Location* loc = new Location();
    loc->setId(1);
    loc->setName("TestLocation");
    loc->setDetail("Test details for this location");
    Assert::IsTrue(loc->getId() == 1, L"ID incorrect", LINE_INFO());
    Assert::IsTrue(loc->getName() == "TestLocation", L"Name incorrect", LINE_INFO());
    Assert::IsTrue(loc->getDetail() == "Test details for this location", L"Detail incorrect", LINE_INFO());
}

TEST_METHOD(ConnectionTest)
{
    Location* loc = new Location();
    loc->addConnection("up", 1);
    loc->addConnection("down", 2);
    loc->addConnection("left", 3);
    loc->addConnection("right", 4);
    map<string, int> test_map = loc->getConnections();
    Assert::IsTrue(test_map.at("up") == 1, L"Connection not successful", LINE_INFO());
    Assert::IsTrue(test_map.at("down") == 2, L"Connection not successful", LINE_INFO());
    Assert::IsTrue(test_map.at("left") == 3, L"Connection not successful", LINE_INFO());
    Assert::IsTrue(test_map.at("right") == 4, L"Connection not successful", LINE_INFO());
}
```

- I kept testing and adding, testing and adding until I was happy with the number of tests I had in the project. I then ensured that all tests ran successfully before beginning on the spike report.



What we found out:

By completing this spike we found out how useful unit testing is and how we can easily add unit testing to an existing project. I had briefly touched on unit testing in another subject a few years ago, but never properly implemented it or understood its true potential. This spike has helped re-teach me how to use unit tests and how useful they really are. They even spotted a few minor bugs I hadn't noticed yet in my code. One of which was to do with the inventory system in spike 6. I had a check that would ignore adding new items to the inventory once the maximum count value had been reached, but this was still supposed to allow multiple quantities of an item. For example I could add two swords and one key, and this would be seen by the inventory system as two items as the Sword object would have a quantity of 2. It spotted a bug where the quantity would not be incremented for an existing item once the max count had been reached.