Shawn Fenner

INF 473 – Problems in Informatics

Progress Report

June 19, 2018

Prepared for

David Gray

Fort Hays State University

Summer Semester 2018

Python Proficiency

So far, I have made good progress during this course in regard to becoming more proficient in the Python programming language. The list below describes several concepts that I have learned and applied to my application:

* Copying objects between containers – when copying elements contained in one Python container to another, the programmer can decide whether to create deep or shallow copies of the elements.
* Overriding methods – super class methods can be overridden to achieve desired functionality. Examples include overriding common methods such as \_\_str\_\_, \_\_lt\_\_, and \_\_new\_\_, as well as custom super class methods.
* List comprehensions – an efficient and concise way to create lists.
* Unpack operators (\* and \*\*) – allow sending a variable number of arguments to a function or method; the latter operator commonly named kwargs (keyword arguments) which accepts keys and values.
* Stack trace – modules *pdb*, *traceback*, and *inspect* all aid the programmer in examining the call stack. pdb in particular pauses the program after a call to pdb.set\_trace() and allows much functionality in viewing stack-based info; however the traceback module offers a simpler, more convenient way of viewing the call stack.
* Generator expressions – similar to list comprehensions except a list is not created in memory; requires less code than and produces the same results as a function containing a yield statement and returning a generator object.
* lambda – similar to anonymous functions in JavaScript. Lambdas have no name, accept an argument list, and contain one line of code.
* JSON format – eases the task of sending and receiving data.

Game Progress

The game has undergone a lot of improvements since the last revision. The most noticeable improvement is that the scrolling feature has been successfully implemented. Now, levels can be much more expansive than a single screen, providing a better, more involved user experience. This was one of the primary features I wished to add and was noted in my initial course document – I’m glad to have this one working. At this time though, the scrolling feature is only available in the one player mode and as such, the two player mode has been temporarily disabled.

Another feature that has been implemented is the primary interface, although currently it is simple in appearance. A user can choose between various game modes and select which mode the user wants to play.

The program is gradually becoming larger and as such, the advantages of object-oriented programming, including modularity and abstraction are becoming more apparent and necessary. The list below is a quick break down of the modules I’ve created:

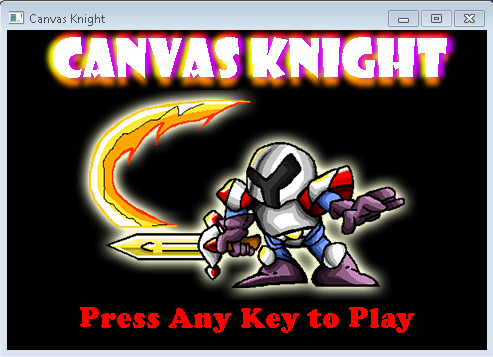
* main.py – the starting point of the application which moderates the order of game modes and initializes the graphics library and sets screen boundaries.
* obj.py – an extensive module that contains game character logic.
* utility.py – contains useful constants and functions other modules can use.
* ui.py – a simple module for working with user interface objects.
* mode.py – runs the various game modes during program execution.
* grid.py – partitions levels to reduce object rendering and update time.

Testing Game Features

In order to demonstrate the game in its current state, two test levels have been created: *Level – Scroll Test* and *Level – Crowded Test*, both of which can be selected via the primary game interface. The first test level focuses primarily on demonstrating the scrolling feature and normal game functionality, such as character controls, enemy movement, and graphics. The second level focuses on testing a level containing many objects and its slowing effect on the game and how such an issue was corrected.

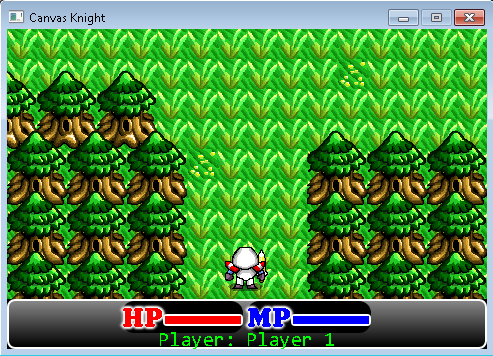
Please follow the steps below in order to test the game:

1. Open the main.py module in IDLE and press the F5 key. The program should start up and the title screen should appear.



2. Press any key to be taken to the options menu. Use the W and S keys to move the cursor up and down the interface, respectively. Use the J key to make a selection.

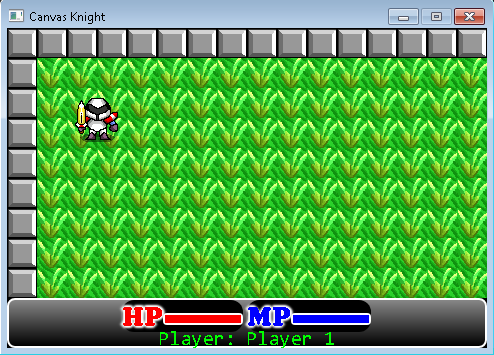
3. For now, select *1 player Level Select* and then select *Level – Scroll Test*. You will be taken into the game world and presented with the level in the below screenshot.



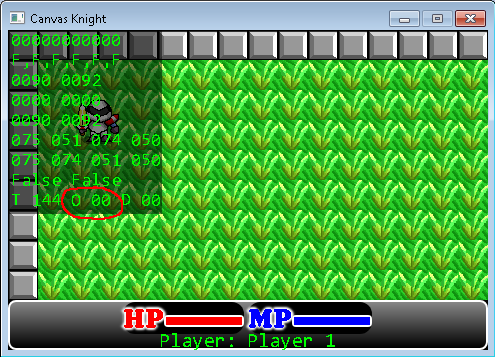
4. Using the W, A, S, and D keys, move the character around the level, and notice how the game world scrolls with him. You will eventually encounter several monsters – use the J (attack) or K (magic) keys to defeat them. Upon defeating all of the monsters in the level, the level will repeat and begin again.

5. Close the program by clicking the red x in the top right corner of the game window.

6. Run the program again except this time select the *Level – Crowded Test* level from the level select menu. You will be taken into the game world and presented with the level in the below screenshot.



7. Press the X key in order to toggle debug information in the top left corner of the screen, as shown in the below screenshot.



8. The highlighted area in the screenshot indicates how many objects, in this case monsters, are being rendered and applied logic to in the level; right now, there are zero monsters.

9. Move the game character to the lower right corner of the level and you will notice the game becoming slower and the number of monsters in the debug information increasing. However, once you move away from the monsters (or defeat enough of them), the number of monsters will decrease and the level will adjust to its normal speed. What this result indicates is that the game only manages objects that it needs to in order to run smoothly.

Continued Learning Objectives

For the remainder of the semester, I will focus my efforts on implementing the two player feature and experiment using the Bottle web framework. By doing so, I hope to learn additional features of the Python programming language and become more proficient in its use.