Dear Professor Gray,

This is Shawn Fenner from the INF 473 summer 2018 course. I have gathered some thoughts, ideas, and links for resources regarding my area of study as you suggested.

For this semester, I decided I would like to expand on my knowledge of the Python programming language, which I initially studied during your INF 322 course. I’ve decided on Python for several reasons:

* Its popularity means better employment opportunities
* Easy to understand syntax and implicit memory management make writing programs easier and increases productivity
* Numerous modules are available to quickly implement additional functionality and promote code reusability
* Can be used for both game and web development – two of my primary interests

During this semester, I would like to greatly expand upon the game I wrote in JavaScript for my final project in the INF 652 course, “Canvas Knight.”

Since the close of the spring semester, I’ve already begun porting the game over to Python, and in doing so, I feel I’ve already begun to make good progress towards learning features of Python that I was previously unfamiliar with.

**Goals for Summer Semester 2018**

* Code a video game in the Python programming language.
* Gain an introduction to a Python web framework and use it to code a simple website.

**The Current State of the Game**

So far, most of the game logic has been ported over, including the logic for the player and most of the monsters. The 1-player mode is mostly complete. However, the level layouts might be altered once the scrolling feature is implemented or this mode might be renamed “Classic Mode” while a new 1-player mode is created. The 2-player mode is working, albeit with only a test level available. I was able to add some 2-player capabilities by importing two modules – socket and threading, and use them to connect two games over LAN.

From my research and understanding, I’ve concluded that the server is in charge of the game data and ensuring that this data is valid, while the client sends commands, namely keyboard input, to the server. The server then responds with data regarding how the client should draw the screen.

Please read the following directions on how to test the game in its current state (Pygame must be installed):

1. Open “server.py” in IDLE and press F5 to run it. The title screen should appear.
2. Press any key on the title screen to begin the game. The test level should appear.
3. Open “client.py” in another IDLE and press F5 to run it. Another test level should appear and two game windows should be open (you might have to drag the windows so both are visible at once).

If the above steps are followed correctly, you should see two game screens similar to the screen shot below:



Take turns giving each window focus and moving the characters around. Use keys W, A, S, and D to move, J to attack, and K for magic. You’ll see that the games are in-sync!

My goal is to improve upon the game in several ways:

**Game Modes**

There should be three different modes of play: 1-player, 2-player, and battle mode.

1-Player: one player completes each level in turn.

2-Player: two players complete each level together in turn, although the levels in which they must traverse are different than those in the 1-player game, generally featuring more monsters and different layouts.

Battle Mode: two players each choose a magical spell and then the player running the server must choose an arena where both players will do battle.

**Scrolling Feature**

This is a must-have feature. An adventure game must scroll in order to create more expansive levels.

**Additional Magical Spells**

Adding extra spells to the game makes the game more fun and adds replay value.

**Additional Monsters**

There should be a greater array of monsters that populate the levels. Also, there should be more monsters that interact with the player, either in aggressive, defensive, or evasive manners. As of now, most of the monsters simply move about randomly.

More powerful monsters (bosses) should also appear after a certain number of levels are completed.

**Learning Outcomes**

These are the following learning outcomes I wish to obtain during the course:

* Become more proficient with the Python programming language.
* Learn how to structure and implement a larger program.
* Learn how to use networking to create a multiplayer experience.
* Gain an introduction to extensions, Django, and hosting options if time permits. See below.

**Time Constraints and Additional Learning**

I’ve come to the conclusion that projects always take longer than I expect they will. However, if time allows, there are other technologies I would like to experiment with during the semester in regard to Python:

1. I’ve always been somewhat interested in the C programming languages. If I have time, I would like to add a simple DLL (dynamically-linked library) written in C which could be called from Python; I believe this is called an extension in Python’s terminology.
2. I would like to understand how Python can be used towards web development. If I have time, I would like to go through a tutorial or two on the popular Django, Bottle, or Flask web frameworks.
3. If I have time, I would like to look into a hosting option in order to allow others to try my program.

**Semester Deliverables**

Throughout the semester, I will submit several versions of the game as it develops, including source files and resources, and an explanation detailing what feature(s) were implemented.

I would greatly appreciate any feedback and suggestions you can provide during the semester in order to improve the learning experience. I would also appreciate if you have any suggestions for game features that you would like to share that I could try implementing, such as for new magical spells, monsters, or level themes.

**Links:**

<https://bottlepy.org/docs/dev/> - The official Bottle website including user’s guide.

<https://docs.djangoproject.com/en/2.0/intro/tutorial01/> - Another introductory Django tutorial.

<https://gafferongames.com/post/udp_vs_tcp/> – An interesting article comparing the UDP and TCP protocols in regard to game networking and why UDP is the preferred method.

<https://gafferongames.com/post/what_every_programmer_needs_to_know_about_game_networking/> – A short history of networking in video games. Introduces peer-to-peer, client/server and client-prediction methodologies, including the advantages and disadvantages of each.

<https://inventwithpython.com/makinggames.pdf> – A 365-page pdf that teaches how to make video games using Pygame.

<https://www.tutorialspoint.com/django/django_overview.htm> – An introduction to the Django framework followed with a simple tutorial.

<https://www.tutorialspoint.com/python3/python_further_extensions.htm> – A tutorial describing how to implement extensions in Python using C.

<https://sahandsaba.com/thirty-python-language-features-and-tricks-you-may-not-know.html> – Useful tips when working with Python containers.

<https://stackoverflow.com/questions/15869158/python-socket-listening> – Short examples illustrating socket usage in Python.

<https://stackoverflow.com/questions/23876608/how-to-send-the-content-of-a-dictionary-properly-over-sockets-in-python3x> – How to send dictionary data over sockets in Python.