Python Project Portfolio

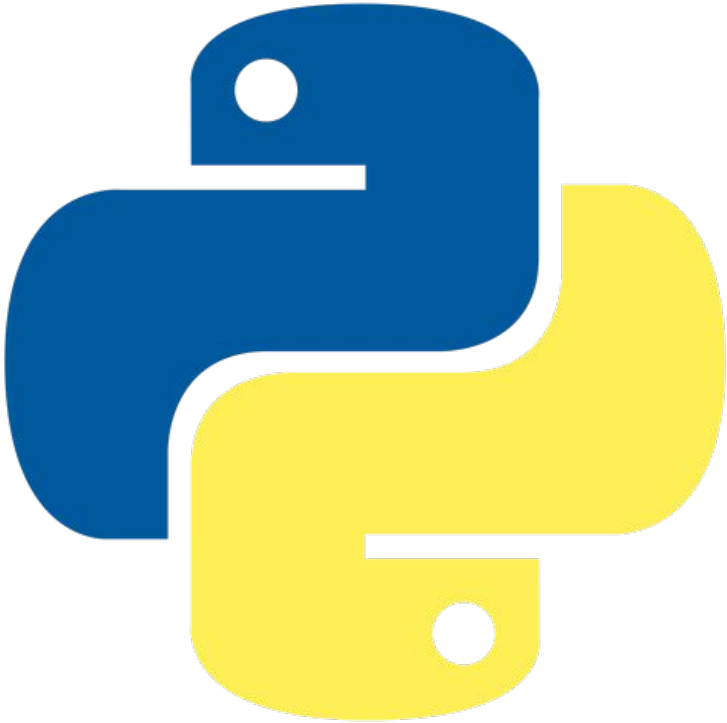
Jack T.

What is my project

My project will be a 2.5d platformer, meaning mechanically you move and interact with the world like you would in a 2d platformer; except all the assets are generated in 3d. The actual game will be a singleplayer (or splitscreen 2p) arena game where you fight waves of increasingly difficult ai endlessly. It’s like a mix between Mario and Castle Crashers. Later on if time permits I will implement a shop and upgrade system, maybe powerups and pickups, or different types of ai and bosses. If possible I might add classes that players can choose to suit different playstyles.

Why I chose Python

My friend does a lot of coding and I remember him saying that although python has it's uses, he wouldn't ever use it very often. He and I have both used php before and we both hated it, so I trusted that python would be difficult. That's mainly why I chose python: to make a game so good he'd be impressed despite saying you couldn't make one. I've only used html and css before so I had no prejudice against python other than what other people have said about it. All the weird things about python people talk about like for example the absence of curly braces has more so become the new normal for me.



Starting the project

After finishing the python course on Udemy, I had to find an engine. I knew I wanted a 2d game, and probably a platformer. Pygame wouldn't install and registration was blocked for months. There were barely any game making engines using python on the internet but I fortunately came upon Panda3D. It was used to make Pirates of the Carribean MMO, ToonTown Online, and other 2000-2010 games. Panda3D was coded on C++ and you program with it in Python. To write I used Visual studio code and github to work on the project at home.

I used the following online Panda3D tutorials to begin:

PANDA 3D TUTORIALS: The Setup Ep. 1 - YouTube , panda3d-tutorial/Panda3D Tutorial.pdf at master · fireclawthefox/panda3d-tutorial (github.com) , ISSUE 1: The one hour 3D world - Page 2 of 2 - My Game Fast (MGF Magazine) , Panda3D Manual — Panda3D Manual

Issue 1: Uploading 3d files

Panda3d was made in the 2000's and hasn't been keeping up with most modern engines. It can upload only .egg files, not the common .obj or .blend 3d files. For this I had to install an extension I found online called Yabee, which can convert .gltf files to .egg files. No other online file converters even recognized .egg files just because of their obscurity. To load files, I usually had to convert the file from .obj to .blend to .gltf to .egg all using blender's import/export method and the online extension Yabee. SimplePBR is another online extension I found for loading the files' original textures as well. These extensions were all installed using pip,

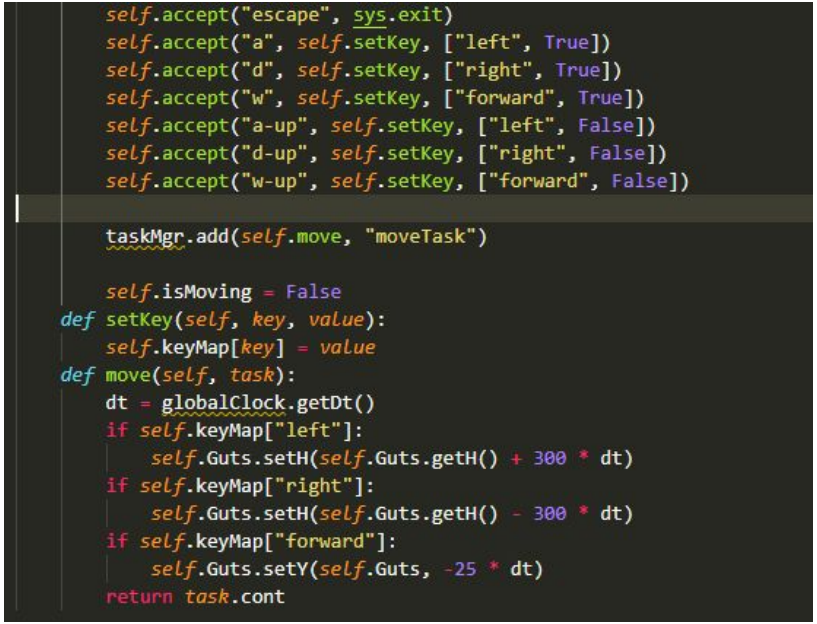
```
self.Guts = self.loader.loadModel("Models/Guts.glb")
self.Guts.reparentTo(self.render)
self.Guts.setScale(0.05)
self.Guts.setPos(0, 5, 1)
```

Issue 2: learning progress

Since I chose to learn Python and the panda3d engine with no other background knowledge, finding out how to use the program took a long time. Panda3d tutorials are also hard to find as well as extensions. So I watched multiple long tutorials and made a bunch of practice test files. Thankfully panda3d has their own manual including sets of sample programs like rock em sock em robots that taught me alot about python and panda3d as a whole.

Success 1: Keyboard input and character movement

After watching 2 tutorials, I combined what I learned and wrote my own program for moving characters. For reference, self.Guts. represents my playermodel, and dt is for frame by frame input so you won't delay with low end computers. This took awhile since there were some mistakes I made with associating the (class)player with my (class)main script.



```
        self.accept("escape", sys.exit)
        self.accept("a", self.setKey, ["left", True])
        self.accept("d", self.setKey, ["right", True])
        self.accept("w", self.setKey, ["forward", True])
        self.accept("a-up", self.setKey, ["left", False])
        self.accept("d-up", self.setKey, ["right", False])
        self.accept("w-up", self.setKey, ["forward", False])

        taskMgr.add(self.move, "moveTask")

        self.isMoving = False
    def setKey(self, key, value):
        self.keyMap[key] = value
    def move(self, task):
        dt = globalClock.getDt()
        if self.keyMap["left"]:
            self.Guts.setH(self.Guts.getH() + 300 * dt)
        if self.keyMap["right"]:
            self.Guts.setH(self.Guts.getH() - 300 * dt)
        if self.keyMap["forward"]:
            self.Guts.setY(self.Guts, -25 * dt)
        return task.cont
```

Time usage and improvements

Every class I try to either figure out how to do something or start working on the issue. I usually solve at least one bug or misunderstanding per day and I’m beginning to start implementing some of the big changes that make up a game. For improvements I will start working on the actual project with more efficiency and focus on programming rather than finding out how to do certain things. I should also add more python coding since most of my code is utilizing the engine. But that might be more time consuming.

Current state

Currently I just implemented keyboard input and movement into the game and I’m ready to start building more major implementations. Right now when you launch the program you can rotate the model around left and right with the “a” or “d” keys and hit “w” to make him start moving.

Timeline

1. Keyboard input and movement - Done, took 2 classes
2. Lighting and ambient fog - ½ class
3. Player 2 - Maybe 1 or 2 classes
4. Camera - 1 class
5. Environment - ½ class
6. FSM animations for characters - 2 classes
7. Collisions - 3 or 4 classes
8. Menu and GUI - 1 or 2 classes
9. Enemy AI - 2 or 3 classes
10. Finishing touches and beginning of adding extra upgrades

Installation and Tutorial Part 1:

You will need the Panda3d game engine, python, Visual studio code, and all the necessary files. These should be in the files. Open the CastleCatchers.py file using a text editor and run. You should get a new window and the program should launch. The first window is a menu and use the mouse to navigate. Upon hitting play, there will be a new window to play the game on. In the game, there are two players: player 1 who is controlled by “WASD”, and the spacebar, and player 2 who is controlled by “pl;’” in the same fashion as “WASD” and uses the up-arrow in place of the spacebar. In this game your objective is to catch the flying harpy by closing the distance and “catching” it. You use the spacebar or up-arrow depending on your player to attack/catch.

Installation and Tutorial Part 2:

Keep in mind the attacks have a short reach and the harpy can phase through walls to escape. Thus players will have to work together and use the arena to their advantage. The flying harpy has a chance to escape however, if the players cannot catch the harpy in time. Players lose if they cannot catch the harpy in time or if they die. If the players catch the harpy they move onto increasingly difficult rounds in which the harpy is faster, the arena more hostile, and the timer shorter. To exit you can hit “esc” to close the program, hit “m” to enter debug mode, hit “tab” to pause, and hit “`” to close pause menu.

Potential Bugs:

Problems can arise with versions of Panda3d and the interpreter. Make sure the interpreter is the recommended interpreter for your editor. If and error says, “Models cannot load” or anything like that, go to the terminal and paste this in: “`pip install panda3d-gltf`”