Simon Corpuz

scorpuz

Project Proposal: Shoot Everything

“In the deep void of space, a lone fighter rocket is all that stands between Earth and invading enemy forces. Do you have what it takes to survive and destroy them all? It doesn’t take much thought – just Shoot Everything!”

Project Description:

Shoot Everything is meant to be a simple shoot-’em-up game with increasing difficulty and complexity in its enemy generation, as well as the inclusion of Boids in its execution of certain enemy movements.

Competitive Analysis:

The only game I had in mind when making this was the already existing Galaga arcade game from 1981. Like Galaga, enemy entities approach the player from the top of the screen to the bottom. The player’s goal is to shoot down all enemies before they reach the bottom row. The enemies themselves have a variety of patterns in their approach towards the player.

Unlike Galaga, Shoot Everything does not incorporate any self-contained levels in its design. Everything is endless, with the game only ending once the player dies. Additionally, Shoot Everything can be seen as somewhat easier than Galaga due to the inclusion of a health meter and no penalty for letting enemy forces get behind you.

Structural Plan:

At this moment in time, the only files that are deemed necessary are a main file for running the program, a GameModes file for setting up different scenarios, and a GameObjects file for defining various entities in the game proper.

The GameObjects file holds the most importance, as it defines everything in terms of a superclass called GameObject. The rocket, bullets, and enemies are all subclasses of GameObject, which are then further divided into different enemy and bullet types.

The GameModes file is roughly the same as that shown in the 15-112 notes for subclassing ModalApp and Mode. There are separate classes for the title screen, the help screen, and the main game. It calls the GameObject class from GameObjects, especially the several static functions that are meant to move and remove each GameObject. Both Boids and line collision are also dependent on these static functions.

Algorithmic Plan:

There are two main parts to this project that I would consider complex: Boids and line collision. Boids is an algorithm that keeps track of the direction of all objects of a certain type, and alters them accordingly thanks to each object’s surroundings. Objects will usually head towards a common direction, move to stay away from one another, and approach a center of mass. I plan on incorporating this with a certain type of enemy in my game called a Swarm. There will be a function that takes each Swarm instance, calculates the three above parameters, and applies those parameters to its changing direction.

Line collision is a type of collision that assumes both objects being compared are made up of lines that form some sort of shape. If any lines from each object intersect, then the objects are considered to have collided with one another. I plan to have a function that takes both objects involved and sets the lines accordingly to their coordinates and types. The function will then check each of one object’s lines individually for an intersection with one of the other object’s lines.

Timeline Plan:

Before TP0, I intend to have the basic framework of the project finished. By that I mean the general structure of the GameModes and GameObjects files, so I will have an easier time working on the rest of the game.

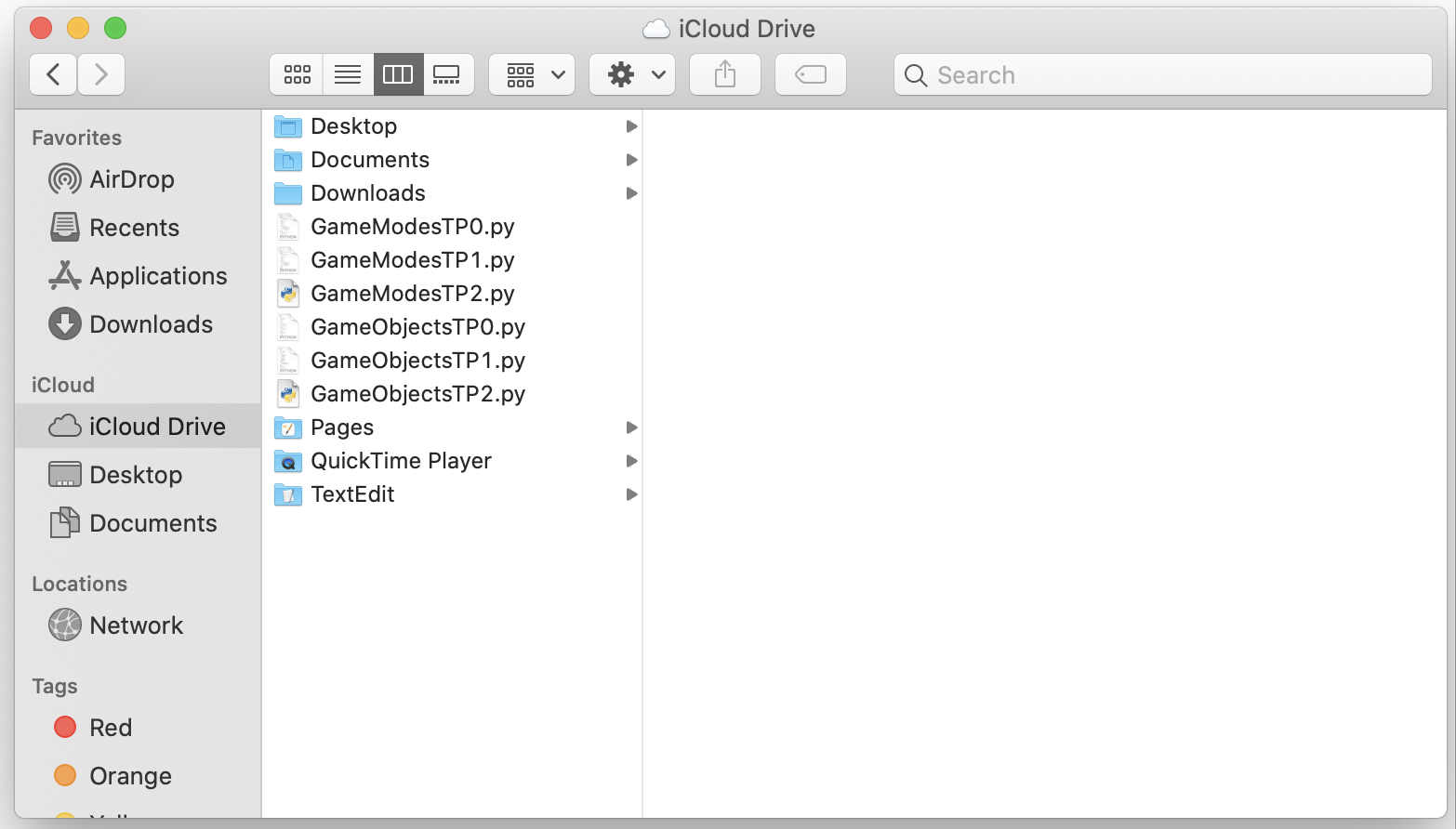
Before TP1, I will hopefully have a basic game setting finished. In addition, I will have most of the Boids work done so that I can debug it in this game setting.

Before TP2, the entire base game should be finished, including line collision and changing difficulty based on the current score/level.

Before TP3, I could possibly entertain the idea of extra features, as well as any modules that I was unable to incorporate before the tech demo.

Version Control Plan:

Each time a TP approaches, I make a copy of both the GameModes and GameObjects files. Considering them as a “checkpoint,” I save them onto the cloud in case I need to revisit them later.



Module List: PyAudio (that’s basically it)

TP2 Update:

At this moment, there have not been any major changes to the above plan. The only thing that may need to be altered is the inclusion of PyAudio, since I have still not found time to begin learning about it.

TP3 Update:

Ultimately, I was unable to incorporate PyAudio in my final product. Otherwise, nothing else has changed from the original design.