**Project Description:** My term project is a game called Buggy Bash. It is similar to an arcade game. In this game, a driver in a buggy must maneuver around a set course, which is based off of the actual buggy course. The driver must avoid crashing into the curb, or running into obstacles, such as pedestrians or potholes. However, if they run into squirrels or flags, they will receive randomly-generated power-ups, such as extra lives, or the ability to clear all potholes or all pedestrians off of the screen. If they can successfully reach the end of the course, they win! My project will run in tkinter, and will use sprites for all objects represented, including the buggy and all obstacles/flags/squirrels on the screen. It will have different modes of difficulty, which will affect the speed of the buggy and the spawn rates of obstacles and squirrels/flags.

**Competitive Analysis:** There are a few games that I have seen similar to mine online. One is called Survivor: Side-Scrolling Video Game, and is described as an “infinite runner game with enemies and power-ups”. Similar to my term project, this game involves navigating a character along a path and avoiding/eliminating obstacles that would cause the user to lose if run into. This game is also done in tkinter. However, this game is an infinite runner which randomly generates a track as it goes. Meanwhile, my game will have a set track, and I plan on randomly initializing all obstacles, flags, and squirrels onto the track before the game actually starts. In addition, the graphics for this game are not the best-looking. I plan on having better graphics by making my game look arcade-y, and by making sprites and the track better designed than those in Survivor: Side-Scrolling Video Game.

Another game that is similar to mine is Procedural Adventure Game. Similarly, in that game, a user explores a world where they try to avoid obstacles and collect items, and the user can eventually win the game. However, this game is more focused on collecting a certain amount of items in order to win, as opposed to just surviving until the end of the course, like my game. Additionally, this game does not have a set world, and instead generates terrain randomly based on the direction that the user is moving. Meanwhile, my game has a set track that the user must follow, or they will lose. Finally, this game utilizes the Panda3D module, while my game is in two dimensions and runs in tkinter.

**Structural Plan:** I intend on having a file for each main object of the game (buggy, general obstacles, and general power-up-bearing objects). In these files, there will be classes for each. In addition, specific obstacles and power-up-bearing objects will have subclasses within their general classes. These classes will contain specific data, including but not limited to information about the object itself, how to draw it, and how collisions/interactions between different objects will work. I also intend on having a file for the track, which will initialize the track object and deal with interactions regarding the track.

I will then have files for different screens the user might be shown. One will be for the start screen, one will be for a “how to play” screen, one will be for the level selection screen, one will be for the actual gameplay, and one will be for a game over/victory screen. Each one of these files will be OOP-based animation files, which will show text/images which the user can interact with. When navigating through all screens besides the gameplay screen, the buttons used to change screens will be treated as an object, and will have a shared “button” class, which will have its own file. Individual buttons will have their own subclasses within that file.

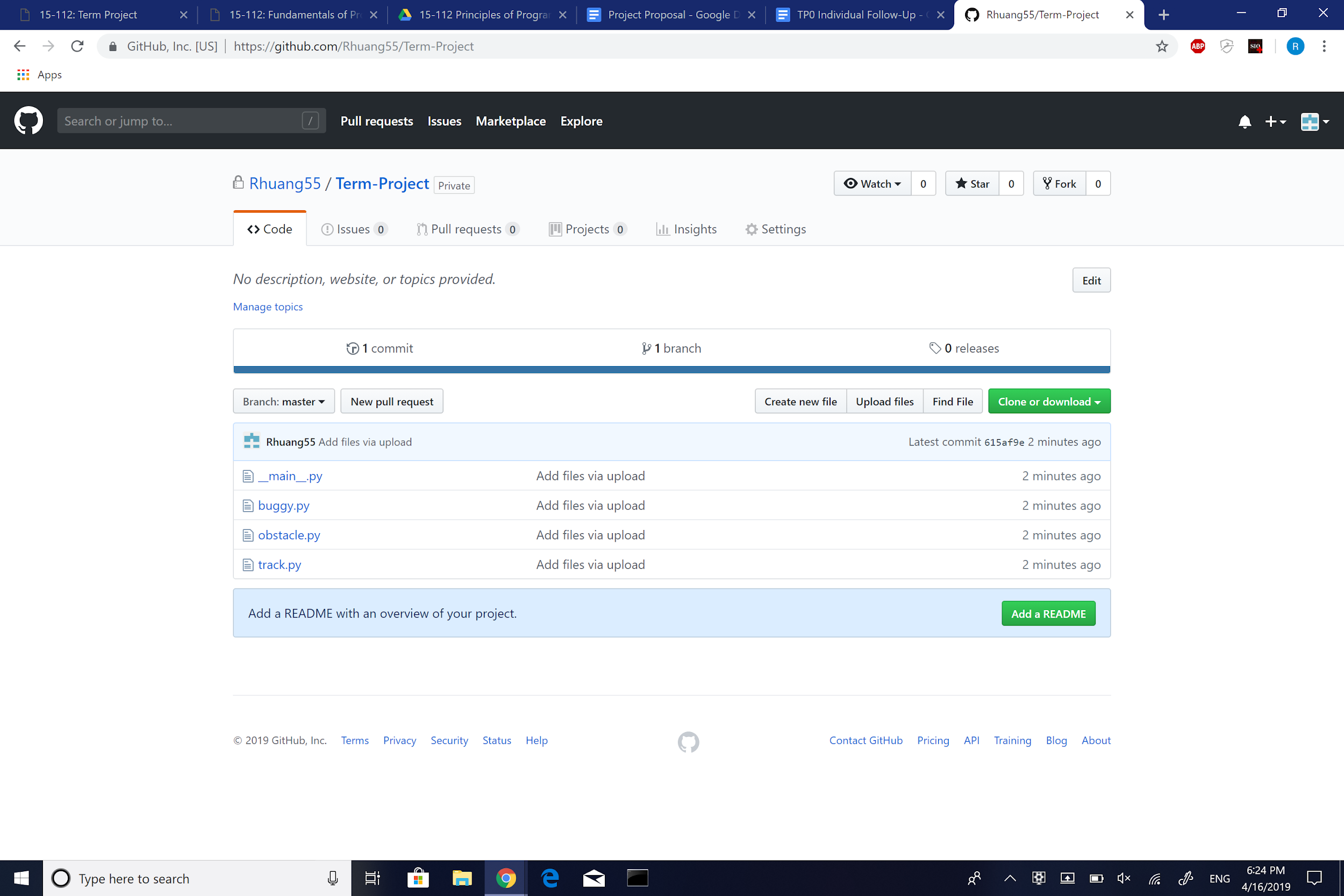
Finally, a \_\_main\_\_ file will bring all of these screen files together, and will handle screen-to-screen interactions. It will be the highest-level file.

**Algorithmic Plan:** One of the trickiest parts of the project will be making collisions between sprites look clean as opposed to messy, so that, when a collision happens, the sprites will be as close to colliding as possible. This will be difficult as these sprites will not be perfectly shaped. I intend on doing this by approximating different sprites as shapes -- for example, the buggy sprite will likely form the shape of an oval, with the pushbar being represented by a couple rectangles. From then, I can use radii from the center at different points of the shape, as well as a lot of Pythagorean Theorem, to determine coordinates on tkinter to use to check for collisions.

In addition, should I choose to implement a feature such as topography near the end of the project (which can be read more about in the timeline plan), I would have to create different sprites for each object with different shadows cast on them, to indicate height.

**Timeline Plan:** I plan on having all of the individual OOP classes created by Friday. Over the weekend, I plan on creating sprites and implementing the collisions between these sprites, and therefore completing the gameplay screen. Before TP2, I will work on creating the non-gameplay navigational screens. At that point, I will have a project that is at least very close to MVP by the TP2 deadline. I will then spend the last week bug fixing, making features such as collisions between sprites cleaner, and adding in extra features which will push my game beyond MVP. I am still deciding on these features, but I may introduce some form of topography that is realistic to the buggy course, which can be shown by changing shadows throughout the course and introducing physics that involves gravity. Once a specific idea is decided on, this timeline will be updated with more specific deadlines. I will create the readme file and the project demo on Wednesday, the day before TP3 is due.

**Version Control Plan:** I plan on backing up all of my code by committing it to Github after every session I spend working on code, or every major change to the code that I make. This will keep every major change that I make backed up, and will allow me to keep track of different changes that I made by looking through file histories.



**Module List:** I only plan on using tkinter, which we learned in class and have experience with using. I completed a tech demo for my TP1 meeting in case I needed one.