Term Project Design Proposal

**Project Proposal:**

**Project Description:**

Title:

This game is a building and resource collection game. The premise is that the character must collect wood from trees in order to build a bridge to the next island. However, since chopping down trees increases CO2 levels in the atmosphere, the water level is quickly rising! The character must explore as much of the world as possible before the ocean drowns the islands.

**Competitive Analysis:**

This game has similarities with other resource collection games such as Terraria and Minecraft. These depend on the gradual building of a base and collect more specialized resources in order to advance in the game. My term project differs from these in that there is a time pressure element added that comes with rising water. Terraria and Minecraft are largely not time dependent allowing players to continue advancing through the game to their heart’s content without much fear of losing game progress in build modes. Furthermore, there are no physical enemies in this game, and thus no “waves” of difficulty that comes with an onslaught of enemies to defeat; instead, the player must fight nature’s natural progression through time.

Additionally, the 2.5D viewpoint is present in many mobile games such as SimCity and Clash of Clans which urge the user to continue building more advanced buildings. Different blocks are dedicated to different objects, and these objects cannot be placed on top of each other, much like the concept of placing trees and other resources in mine. While these apps do specialize in resource collection, there is also no way to lose the game in its entirety; resources are simply depleted without consistent game play. Furthermore, these games feel more first person without a character to operate, while mine has a character that moves around.

**Structural Plan:**

For each differentiable type of item in my game, I plan for it to be its own class. For example, Block class and Character class hold the attributes for these game elements, and any general functions that fall outside of the class will also go in these files such as island.py and character.py. Blocks will also be stored in arrays for easy identification of position relative to each other, and all game elements will also be stored in sprite groups for drawing. A file called variables.py will store all global variables needed for basic game functionality. The main.py will contain the main while loop for updating sprites and drawing the game and calling functions to get the game started.

**Algorithmic Plan:**

The trickiest part of my project is making my game isometric, or 2.5D. This requires, for most elements in the game, that I keep track of both the cartesian and isometric coordinates. Furthermore, some calculations are easier in cartesian while some are easier in isometric, so maintaining a blend between the two is necessary. As such, this is especially important for the blocks that make up my individual islands, and therefore, my approach is to establish separate isometric and cartesian blockArrays for each in order to keep track of these different realms. Additionally, all blocks must be correctly scaled from their original rectangular shape to be “diamond-like” in order to appear isometric. This is done by rotating each shape by 45 degrees and then doubling the width of the image. Movement across an isometric plane is done by moving half a block height down or up in the y-direction and one block width left or right in the x-direction. Because every isometric motion requires a change in both x and y cartesian coordinates, both these must be tracked.

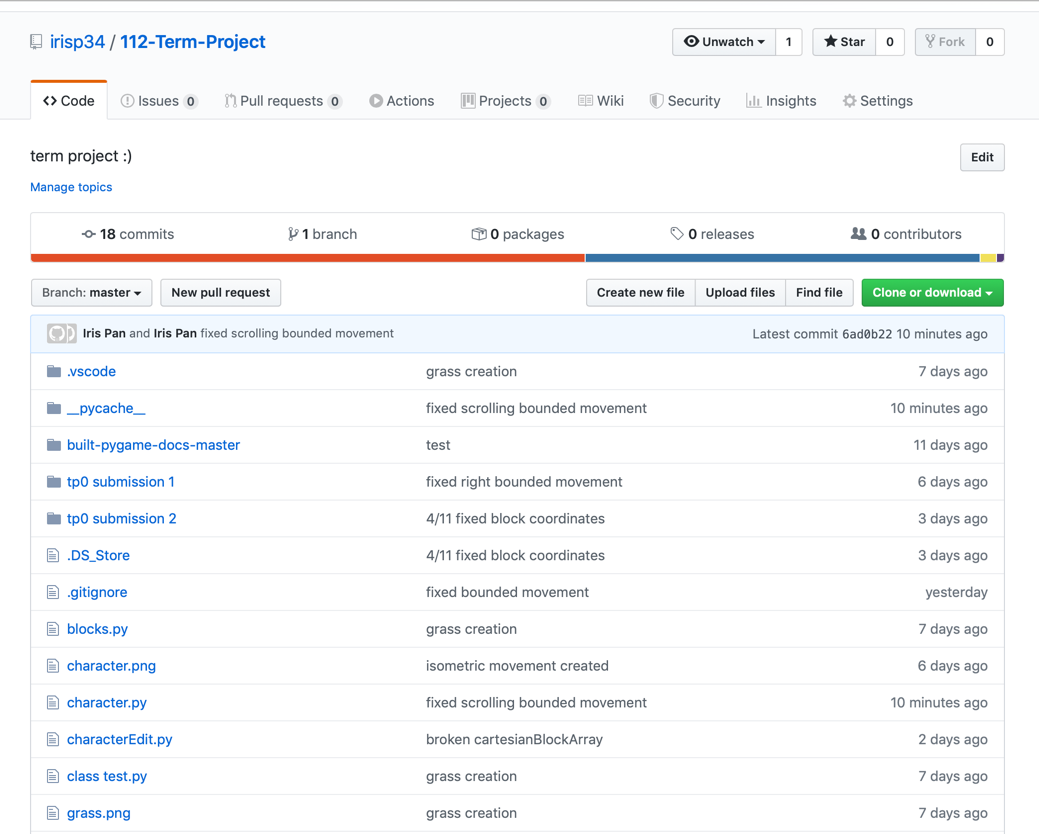
The second most difficult will be to come up with an even balance between how fast the water rises and how quickly the game goes as well as creating the appearance of water rising. Without this balance, it will be difficult to reach other islands before the game is over, so there should be some implementation of being able to slow water rise. This may be modeled off an exponential equation such that water growth starts off very slowly to allow for prolonged game play but then increases as the user grows more experienced.

**Timeline Plan:**

* Tp0/Checkpoint 2: have basic isometric projection of the board and have the character move without bound.
* Checkpoint 3: Add water to the background, and bound the character’s movement to be within the island.
* Tp1/Checkpoint 4: Add side scrolling and make the island appear as if it is sitting in the water with some drawn polygons. Add borders to the individual island blocks.
* Checkpoint 5: Add tree sprites and other elements and modify the board so that the character cannot go through these sprite elements. Be able to remove the sprites from the Group when they are clicked on. Move all clicked on and removed sprite resources to inventory.
* Tp2/Checkpoint 6: Create a screen that allows the player to see what they can build with their resources and then build it on the screen. Expand player movement to be able to move where a bridge is built.
* Checkpoint 7: Create a score counter that tracks how many points the user has scored. Based on how many trees have been cut down, raise the water level. Create a screen saying the player has lost when the water drowns the island and display the score. Display the CO2 amount in the corner.
* Tp3/Checkpoint 8: Add as many different buildings as possible that will enable the player to build certain buildings that will decrease the CO2 levels to a certain extent. Decrease a certain amount per building per unit time passed. Create splash screen at the beginning of the game that will tell the user how to play the game. Add pausing capability to the game.

**Version Control Plan:**

Committing and pushing to a repository in my GitHub account periodically. All pushes will be labeled so that I know what I did in that commit.



**Module List:**

* Pygame
* Numpy (approved by Professor Taylor to use without Tech Demo)