Project Description [2.5 pts]

Name: Crossy Code

Description: Crossy will be a multiplayer infinite runner game similar to the popular mobile game Crossy Road. The primary objective of this game is to defeat your opponent and stay alive while gradually making your way upwards. The challenge of this game comes in the form of obstacles. From rivers, and cars, to trains, the landscape of Crossy Code is a dangerous one that requires players to maneuver through randomly generating lands achieving the highest possible score. The game will end when the player either gets hit by a car or train or falls into the river which will cause the score to be logged if it manages to beat the high score.

Similar projects [2.5 pts]

Being such a popular game, Crossy Code has many similar projects like itself based on the popular game Crossy Road. Isha Agrawal's term project "Crossy Road" (https://youtu.be/TD2HeH8Q05g) and Winnie Lin's term project "Crossy Road But Not Really" (https://youtu.be/z3O_XI-wRwE) are similar term projects completed by previous 15-112 students. My project will share a similar inspiration with these projects, coming from the popular game Crossy Road. My project will be using the same obstacles and terrain as the other projects as well as sharing the multiplayer feature of Winnie Lin's term project. My version of Crossy Road will be one that is more smooth like Isha's project compared to the choppier movements of Winnie's project.

The primary difference between these projects compared to mine will be the ability of my project to mimic the 3D-like graphics of the original Crossy Road. Rather than the top-down 2D design of the other projects, my project will cause the terrain to be angled slightly so that it allows for depth to be shown to add more dimension to the game itself. This will be done by using multiple different sprites for the same character so that it properly adjusts toward the direction of the player's movement. My version of multiplayer will differ from Winnie's version of multiplayer as it will utilize the same map rather than different maps. This difference will allow players to interact with each other on the map by bumping into each other to disrupt each other's paths during gameplay.

Structural Plan [2.5 pts]

The project will be organized in my existing 15-112 folder as a folder named Term Project. This folder will contain the master file, which will contain the primary code for the project, and a temporary code file, which will be used to write the current code that I am working on, which will later be copied to the master file. This folder will later contain the images for the sprites utilized in the game in addition to the code files. Crossy Code will have three different game statuses, the Start/Gameover status, the Paused status, and Running status. The Start/Gameover status will make use of the onMouseMove and onMousePress functions to interact with the screen including, a single/multiplayer toggle button and quit button. The screen will also include a high score and last score display as well as the ability to move into the

Running game status by pressing any key other than the esc key. The Paused status will temporarily stop the game (stopping the character, moving obstacles, and the screen from moving) and display controls for the game. This status will be able to be toggled on and off by pressing the esc key on the keyboard. The Running status will be where the game interactions will actually occur. It will consist of movement animations for each of the characters, generations of the terrain itself, and the obstacles.

The project will be organized primarily using classes to make the code more readable. A Button class will be used for easier adjustment of the buttons utilized in the start and help screens. The terrain will be separated into the Grass class, Water class, Railroad class, and road Class for map generation itself. The obstacles will be separated into the Tree class, Cars class, Logs class, and Trains class. The final class will be the Player class which will contain the status of the player (dead/alive) and movement. These classes will be incorporated into the MVC model and consist primarily of onKeyPress functions and redrawAll functions for the generation of the animations themselves in addition to the controls of the player's movements.
Algorithmic Plan [2.5 pts]: A plan for how you will approach the trickiest part of the project. Be sure to clearly highlight which part(s) of your project are algorithmically most difficult, and include some details of how you expect to implement these features.
Algorithmic Plan [2.5 pts]

The trickiest part of the project will likely be the infinite random terrain generation in addition to the actual interactions between the map and the player. The infinite random terrain generation will first be achieved by creating a grid of squares that will divide up the entire map using a 2D list and a dictionary mapping the coordinate tuples to the current status of the square. Each row will be randomly assigned a terrain class and generate obstacles depending on the landscape. These terrains will be grouped to take up from 3 to 6 rows and moving obstacles will have varying speeds. The random generation of the terrain and obstacles will be taken using randrange(lo, hi) from a list with the names of the terrain and a separate list with the obstacles in addition to using a dictionary to organize varying speeds for the obstacles. Using redrawAll, the map will be created from the top down so that the foreground correctly covers the background using a for loop for the correct number of rows.

The obstacles will be drawn using sprites of classic Crossy Road and will take up varying numbers of columns. The player will be able to control their sprite using WASD or arrow keys depending on which side they are on and the sprite will move one square at a time using onKeyPress. Depending on whether there are obstacles or not currently in the dictionary of the coordinates, the player's sprite will either lose, move into the space, or be prevented from moving. The sprites will make up a large portion of the animation process and will be implemented using the Pillow library in Python, and the separation of them using classes will allow for faster implementation rather than having to use the same code over and over again.

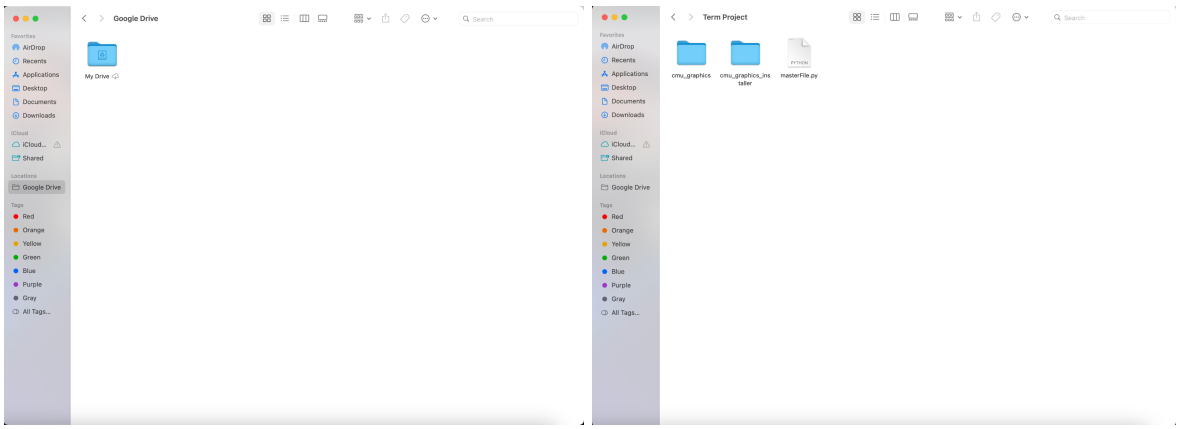
Timeline Plan [2.5 pts]

4/10 - Finish TP0
4/13 - Write terrain classes (grass, water, railroad, road)

4/15 - Implement creating a grid of the map (2D list, dictionary for status) that will correctly shift upwards with a temporary movement key
4/17 - Finish the infinite terrain generation algorithm so that it randomizes the terrain animating from the back forwards
4/19 - Write the character class and the movement capabilities so that the terrain generates properly with the score system
4/20 - Write the obstacle classes (tree, car, train, logs)
4/22 - Implement the obstacles onto the map with correct interactions with the character
4/24 - Implement multiplayer capabilities with bumps between players
4/25 - Finish different the Paused screen and the Start/GameOver screen
4/26 - Finish recordings for TP3

Version Control Plan [1.5 pts]

The backups will be saved first in the Google Drive folder of my computer located in the 15-112 folder which contains the Term Project folder. This Google Drive folder is automatically linked to my andrew.cmu.edu Google Drive and everything that is downloaded in the folder is automatically stored in the Google Drive. This will enable access to my code files and save backup capabilities on my Google without having to worry about constantly uploading different versions of my project to Google Drive. This Google Drive folder will also include the image files that I will be using for the sprites used in this project so that they will not be changed.



Module List [1 pts]

I am planning on not using any additional modules and only planning on using libraries included in Python such as Pillow to import images from sprites used in the game.

Storyboard [5 pts]
Generate a storyboard that demonstrates how a user would interact with your finished project. A storyboard is just a series of sketches showing (roughly) what your project will look like. Your storyboard should have at least six panels, and at least three of those should demonstrate

features within the project. You may scan or take a picture of your storyboard and include it in the directory as the file storyboard.png (other acceptable filetypes include .gif, .jpg, and .pdf).

Include any preliminary code files you have already created at this stage. Some code at this stage would be helpful, but not necessary. It may be prototype code, and so perhaps may not be part of your final project. It also may or may not have any user interface.

TP1 Update

I will be maintaining the same general concepts for the game, such as movement, aesthetics, and the multiplayer feature. I will be changing one of the terrains and obstacles, rather than using a railroad and train, I will be using the same road terrain as used alongside the cars and using a racecar which will move at a much faster speed compared to the cars. This racecar will have a sound queue similar to the sound used at the start of every MarioKart race. The reason for this change was that it was difficult to find a proper isometric railroad block so I will instead be utilizing the same block used for the car roads.

TP2 Update

I will be taking out the racecar from the possible obstacle course currently due to time constraints and not having enough time for proper implementation. I will also be taking out the multiplayer feature due to similar time constraints. The general controls and interface will stay the same from prior to TP2.

TP3 Update

I created a leaderboard using a .txt file and using read and write file functions in order to allow for all scores to be stored in the game and after dying in the game, the largest four scores and the usernames associated with the games in addition to the stored highest score of the username of the current player is displayed in a box. Instead of the player dying when they reach the edge of the screen, the player is brought back to the other side and instead, the difficulty of the car movement was increased to make the game more enjoyable