Final Project Report

Original Plan:

- Our Bare Minimum:
    - Original Super Mario Bros 1-1 level similarity
    - Classes include heroes, enemies, terrain, projectiles, etc.
    - Custom Graphics, Terrain

- High-End Possibilities:
    - Multiple levels, power-ups, boss fight(s?)
    - Different types of enemies
    - Difficulty Levels
    - Different hero types with different abilities
    - Theme music for different aspects
    - Sprites
    - Enemies are weaker to certain weapon classes

Final Project Plan:

- Side-scrolling dodging/maneuvering game with a controllable character and free movement
- Hero, enemy, and terrain classes
- Custom sprites and terrain by JD
- Difficulty ramps up as game continues on
- Start and game over menu screen

What Changed?

Early on we realized the limitations of Pygame and began to restructure our expectations for the amount of work we would be able to do within the four weeks before the deadline. So, we essentially dismissed the possibility of creating multiple levels and power-ups off the bat. We also realized that having multiple complicated enemies in a rather simple and compact space wasn't going to be viable. Since it took a really long time to get the collision mechanics to work, the idea of a jumping puzzle-esque thing to get to the end to win didn't work out. Although since we did get the collision physics to work, maybe we could've done it if we had an extra week or week and a half without finals. Our new bare minimum was a couple of platforms, a playable player class, and an enemy or two.

GROUP MEMBER CONTRIBUTIONS:
(Can you guys write what you did here in your own words I don't wanna discredit you for things I didn't know you did and whatnot)

Nolan

- Programmed movement physics, and player and enemy classes.

Atul

- Programmed the world, collision physics and classes that didn't work (enemy and button). Also created a test file with a very simple playable class as a bare minimum for the presentation in case we couldn't get our main code to work as intended.

JD

- Created graphics for the player, multiple enemies(some unused), background and terrain.

Difficulties/Challenges:

- The Pygame module itself came with a number of limitations, including having to produce every asset from the ground up, such as gravity. It was also difficult to program buttons and to predict how certain changes were going to affect the game before testing it.
- Getting started was one of the hardest parts - it was confusing to try and read the documentation and Nolan often had to resort to watching hours-long video guides and trial-and-error to even get the window running.
- We also ran into just a couple merge conflicts throughout; we learned to periodically save copies of the code and files to Google Drive and other places just in case.
- Sometimes it felt like the code just didn't want to work. We'd add new code, but it wouldn't work for various reasons. We'd check with multiple videos to show how to correctly implement it and it still wouldn't work.
- The split screen/multiple screens to get the start/exit/restart never seemed to work.
- Working on a team was actually super life-saving because your teammates could look over your code if it didn't work or they could tweak it and see if that would fix it.

▶ The ultimate introduction to Pygame

Nolan took a significant amount of the foundation elements from this guide. He used it to program all classes and key inputs, load images, create surfaces for the sprites and background, count score, establish game states, and more.

Atul:

I didn't straight up copy-and-paste any code but chunks of my code such as the Button, Enemy, and Potato class were edited versions of *Coding with Russ*'s tutorials. I basically watched the video, followed along, and edited a couple of

things to fit our game. For example I changed the enemy to make it only move a set amount instead of from end to end; Even though you can't tell because it doesn't work.