Kent Denver Innovation Scholars Program
Spring, 2020

DISTANCE YOURSELF

EMILY AMSPOKER

TECHNOLOGY AND DESIGN TRACK
Chris Smith

OVERVIEW

This March, I participated in a three-week-long Major League Hacking Hackathon called Hack Quarantine. For this hackathon, I was tasked with creating an app or website to help promote awareness or to support essential workers. Although I was on a team with three others, the coding and design of the game is my own work (minus the assets I used from the Unity Asset Store).

GOALS/TOOLS

I wanted to create a game that emphasized the importance of social distancing and good hygiene habits while being entertaining and "replayable" enough to entertain people stuck in quarantine.

Moreover, I wanted to create a game that was appealing to all ages, but especially kids, as, since they have no jobs and many of them are not even doing remote school, they would be the most likely to be bored during quarantine times.

With this in mind, I designed a game in Adobe XD, as I had had experience designing with it before and found it easy to use. (see Figure 1). It would allow users to play several mini-games in order to earn "cures" that they could use to help people in the game. I decided to use Pygame in order to program the game, due to my prior knowledge of Python.

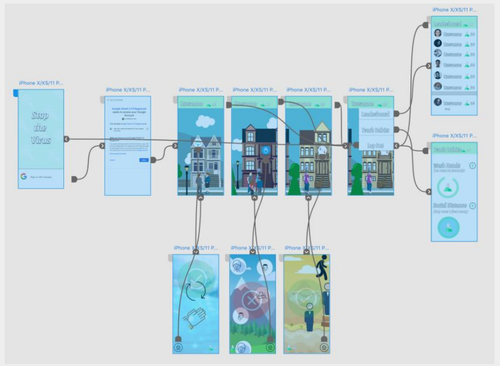

FIGURE 1: Adobe XD Prototype of App

PROCEDURE

Since I did not know how to use Pygame, I started by learning Pygame. I consulted online tutorials, as well as the Python documentation.

```
def spawn():
    #spawning platforms of a random width and set height, must be above the player's head and screen height
    while len(platforms) < 6:
        tall = player.height
        width = random.randrange(70, screen_width//2+100)
        p = Platform(random.randrange(screen_width + 10, screen_width*2), random.randrange(tall, screen_height-100), width, 40)
        platforms.add(p)
        all_sprites.add(p)

    #spawning enemies as time goes on within the game window that pace back and forth from a certain endpoint
    while (len(enemies) < (time//27)//3) or (len(enemies) <= 0):
        x = random.randrange(0, screen_width - 64-20)
        y = random.randrange(0, screen_height - 40)
        end = screen_width
        if x + 30 < screen_width - 128:
            end = random.randrange(x + 30, screen_width - 128)
        elif x + 30 < screen_width - 128:
            end = random.randrange(x + 20, screen_width - 128)
        else:
            end = random.randrange(screen_width-128, x + 30)
        e = Enemy(x,y,end)
        enemies.add(e)
        all_sprites.add(e)
```

FIGURE 2: Enemy/Platform Spawning Code

I started by making simple programs in Pygame, then began to work on one of the mini-games. This game was a platformer where the player needed to dodge people to keep their health up by jumping onto platforms that appear randomly throughout the game.

The most challenging part of creating this game was the physics of making the character jump, as well as spawning the platforms and the enemies (see Figure 2).

Additionally, I could not find free game assets that would work with how my code was set up and were coronavirus themed, so I used assets from the Unity Asset Store that were not coronavirus themed.

FINAL PRODUCT/BUGS

In the end, I could only finish one minigame due to the time constraints of the hackathon. I created the game itself, along with a menu/title screen and instructions (see Figure 3).

Since I had difficulties using the font I wanted to in Pygame, I created the text and buttons in Adobe Illustrator and exported them as .png images.

In the end, the game worked like this: the user would use their arrow keys to move the character left and right and press the spacebar to jump. The user could land on platforms and fall off as well.

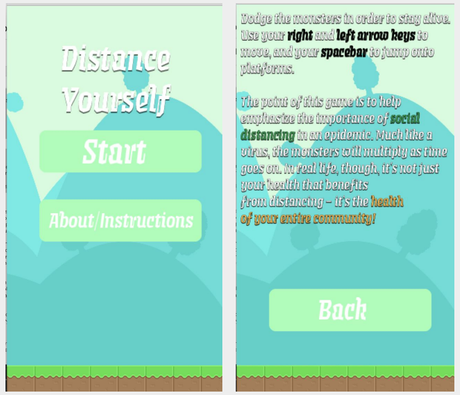

FIGURE 3: Menu and Instructions

The player starts off with 50 health points. They can lose points by colliding with any of the enemies, which increase in number as time goes on. The player accumulates points by staying alive longer. Once the user has 0 health, the game is over (See Figure 4).

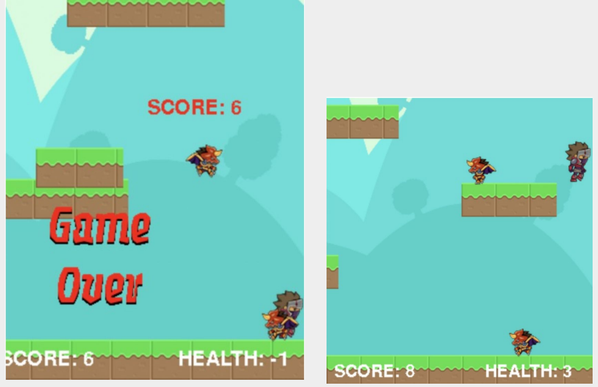

FIGURE 4: Game Over/Gameplay

There are still some issues with the gameplay itself. Since I used the built-in collisions using the Sprite module in Pygame, sometimes the player and platforms collide when they are not visibly touching, leading to occasions where the user seems to teleport.

Additionally, I forgot to account for the fact that two platforms should not be on top of each other while spawning, and, as a result, it spawns a lot of platforms that overlap.

Regardless of these issues, however, the game is still playable and runs without errors. I have played the game many times while testing and it is pretty entertaining (although I am definitely biased).

NEXT STEPS

There are two next steps that I can choose from. The first is building the original app in its complete form as I had designed it originally (see Figure 1). While this would be time consuming, it would result in a more complete app or website. On that note, the other step that I could take is finding a way to showcase/distribute the game I already made. Unfortunately, I couldn't find a way to embed Pygame in a website, but I am looking to find ways to make the game something you could install without needing to have Pygame on your computer.