MP4 Reflection: Club Penguin Sledding Game

Diego Berny and Rachel Won

Project Overview

This project utilizes pygame to recreate the sledding minigame from the beloved webiverse, Club Penguin. The goal of the game is to navigate the sledding penguin through various obstacles that speed or slow it down in order to get to the finish line in the shortest time possible. The track is randomly generated, so every game played is slightly different.

Results

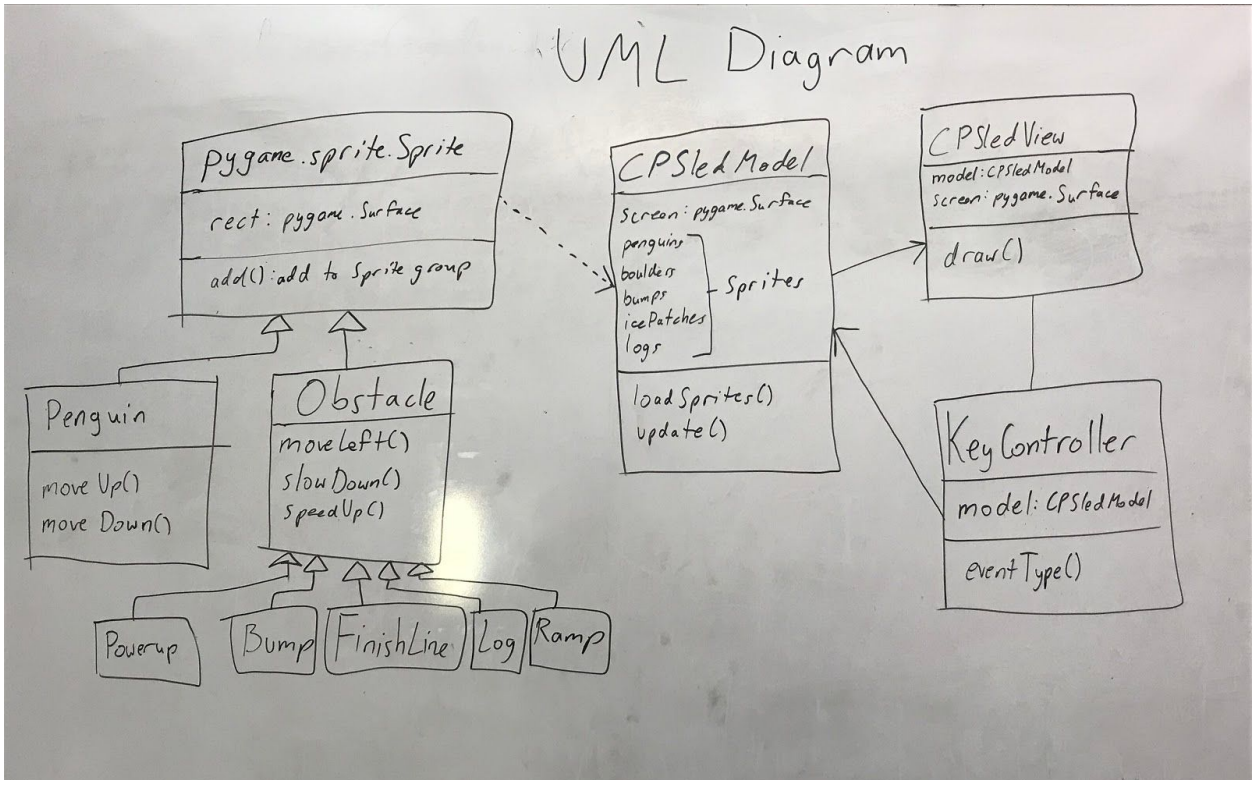
In order to make our game look like its source of inspiration, we used images from the original to display the penguin and all of the objects on the track. We are able to tell when the penguin collides with an obstacle, and also what kind of obstacle it collides with, so the result of the collision varies depending on what it is. For example, the penguin speeds up when sliding on ice, but falls over and slows down significantly when hitting a rock in its path.

The game runs until the player reaches the finish line, at which point the penguin stops sledding. The player can check their time, shown in the top right, to see how fast they were. The timer starts right away, and stops once the player reaches the end.

Implementation

On a large scale, our game utilizes a Model-View-Controller system. The user's input is recorded through the Controller, which then modifies the Model. The View part then displays the updated Model each frame. Specifically for our game, we made use of Pygame's Sprite objects, which have very useful features like detecting collisions with other sprites already built into them. All the objects that interact with each other in the game (the penguin and obstacles) inherit from sprites for this exact reason. We decided to split up the Penguin from Obstacles because although they both inherit from sprites, we wanted them to have different functionality. Similarly, each specific type of obstacle inherits from the Obstacle class so they have the same methods, but we can differentiate between them.

At the start of the game the model loads the sprites onto the track and then updates their position based on the inputs from the user and what collisions have occurred to speed up or slow down the progression of the obstacles across the screen. We also ended up adding a method to minimize the number of obstacles that spawn overlapping each other. At first it was a large and noticeable problem, but most cases of that occurring have been removed with our obstacle interference detection function.



Reflection

We started out unsure of how to approach creating our game, but the initial road block of setting up code that at least did something was possibly the hardest part. After that we knew what kinds of things we wanted to add and had ideas for implementing them, so the code was mostly flowing after that point. We also realized fairly early on that Pygame already has functions for something we wanted to do, so in the future we should make sure that we are not unnecessarily rewriting things that already exist for us to use.

We mostly used pair programming to work together throughout the project, which worked well for both of us. We both had opportunities to be the driver and navigator, so we learned how both roles worked fairly well.