Anthony Grant-Cook, Alexis Caldwell, Simon Jones

Professor Mohan

CMPSC 201

2021-12-03

Progress Report

Our platformer game project, which we have decided to name "Mohan Jump," in honor of our Professor Mohan, has come along significantly in the past two weeks. We have implemented game illustrations, physics, and collision detection. The game does not yet have a method by which to tell if the player has won or lost the game. Another missing thing in our game development is involving a second language to incorporate into our project.

We have decided that using Flask, which is python based, to host the web page, would be the best option. Flask is a python library that is used in web application development as well as static website generation. With flask, we can serve the static html files along with their javascript via the Flask API. This will, in theory, allow the game to be deployed to a reverse proxy, which is a secure way to host a static web page. Flask is very capable of doing something such as this.

A major challenge we faced was how to implement collision detection for the player in the tile map of the game. The tile map of the game is a grid of tiles as a string that are reflected as they are drawn onto a javascript canvas object. The javascript canvas approach to level creation will be discussed in the following paragraphs. Originally, we struggled to decide how to best detect when a sprite would collide with a wall or floor when both of the said objects' positions were determined by a single point. The method that we ended up arriving at was based upon the width of the sprite (or the player) and the side width of each tile in the tile map. There were three cases for both the x and the y axis that constituted a collision. These three cases were if the player's outer bounds (either right and left or top and bottom sides) enclose either or both of a tile's outer bounds along the same axis. If a collision was constituted, this would prevent player motion along that axis. We did not bother to implement conservation of momentum, where the player would bounce off of the wall, because we are not using a physics engine. Because we are writing this from scratch, we are skipping over various minute details that would normally be covered by a platformer game.

As for the tile map creation, we originally decided upon drawing html elements with javascript imperatively. This method was migrated over to using a javascript canvas method, in which you create a region for imposing graphics. The imposed graphics are

proportionalized to the pixel dimensions of the canvas. This was close to the original implementation with html div elements, except the graphics were concealed to a single canvas. The graphics were also quicker to be drawn, as the canvas is optimized for drawing two dimensional graphics. The only problem faced with this implementation was trying to draw images at the same time that the images were loaded. The script originally would try to draw each tile onto the canvas before the images were loaded. To fix this, we created anonymous functions that would draw the image onto the canvas upon the loading of the respective image. This would ensure that no images would be drawn prematurely; however, there was a peculiar issue with variable scoping that would stack all of the images at the end of the canvas. This was eventually determined to show that javascript, when creating anonymous functions to be used in an event handler, requires the anonymous function to be created in a closure to capture the current x and y coordinates of the tile. This took roughly three days to solve with little progress across the three days!

The game's physics just forces the player to accelerate downward when unobstructed by any floor obstacles. The physics of the game also have the player decelerate when not being moved left or right. This is similar to how friction would work in the real world. The game was heavily influenced by Mario, which is one of the most popular platformers in existence today.

Our next steps for this game will be minor improvements to enhance player experience, along with backend improvements on content delivery.

The first step involves finishing the game to have mechanisms for victory and loss. This will be relatively simple, involving checking for a player collision with the endpoint block, which constitutes a victory. A loss would be constituted by falling off of the tile map.

Another step we will take will be to possibly track the player movements with a smaller viewport, so the user playing the game will not struggle to see the player on their screen. A step toward this was completed when migrating the drawing of the tile map to the javascript canvas method, which wraps all of the graphics into a manipulatable element.

An additional step towards completing this project will be to use the python library Flask to serve the game as a static html web page. It will be interesting to see how to include the javascript files in this game, as none of us have interacted with Flask to serve static web pages before. This will be a more secure and manipulatable way to display the game over the internet.

Our project is coming along swimmingly; we look forward to seeing what other unexpected turns the project takes, and we will continue to develop until the deadline. We have taken Professor Mohan's advice to implement a new programming language into our project, but have not yet started on that implementation.