Anthony Grant-Cook, Alexis Caldwell, Simon Jones

Professor Mohan

CMPSC 201

2021-12-13

Final Report

Our platformer game project, which we have decided to name "Mohan Jump," in honor of our Professor Mohan, is now completed. We have implemented game illustrations, physics, and collision detection. We have also successfully hosted the game on a python flask server and also have implemented game mechanics such as player loss, player winning, and player restarting.

The motivation for this project was, firstly, our appreciation for Professor Mohan! We wanted to incorporate him into the game since he is the professor of this class. Secondly, we all enjoy playing platformer video games as well as classic arcade style games. Platformer video games are video games in which a two dimensional player jumps on platforms and is governed by basic laws of Newtonian mechanics, meaning that they fall with gravity and take time to accelerate. Super Mario Bros is a prime

example of a two dimensional platformer game, and it served as the main inspiration for this game.

Before getting into the specifics of our game, we first had to do an immense amount of research about what goes into a game and what should be included in a game during the time period we had to make it. In the beginning, our ideas were vast and seemed timely. However, with further research our initial ideas seemed extremely hard to implement and not probable for what we could possibly complete by the due date. Looking into steps on how to approach our game we decided it would be best if we made a character and go from there. So, we wanted to include a sprite that would be our main character of our game which ultimately ended up being Professor Mohan. Taking a picture off the Internet of our Professor as well as a video game body, we had our character. Next, we had to decide what we wanted our character to do. Did we want to have our character fly? It seemed like a good idea at first but what would we do while our character is flying? We ended up deciding on being able to make the character jump as well as move left to right with a goal for the user to reach. While we were making these decisions since we were making a game, we had to familiarize ourselves with working with various programming languages such as: JavaScript, HTML, CSS, and Python. Not everyone in the group knew all of the languages, so it was a learning task for those who

were not familiar with the programming languages. With those goals and ideas in mind, we started to implement our game.

We faced many challenges in the development process of this game. Through trial and error each challenge allowed us to learn more about web development, and how the video game design process works from scratch.

The first major challenge we faced was getting the flask app to work with the directories of our game. Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible framework for new developers since you can build a web application quickly using only a single Python file. We had two directories, one for the static files, and one with the HTML template to be rendered by flask. When flask would serve the templated HTML file, the file would request for assets like images and javascript scripts that were set as relative file paths. This was the wrong way to do it, as we discovered. Because the HTML page was not aware that we were templating it with flask, using relative file paths would be an incorrect approach. We instead had to use a built-in method that came with flask called `send_from_directory()`. This method would allow flask to do a redirect, so when the app sent a request for a certain content asset, the request would be redirected

to the relative file path where we were keeping all of the javascript, css, and image files. After numerous bug fixes and incorrect namespacing, we were able to implement this fully without error.

Another challenge we faced was how to implement collision detection for player movement. Originally, the player element would move through walls and fall through platforms because there was nothing set in place to prevent the player from knowing when it was about to collide with a tile in the game. After careful thought, we developed a way to prevent players from moving into walls that would, at every game loop event, check to see if the player's velocity was going to cause it to move into the bounds of a wall within the next game loop. If it was, then we would prevent the player's movement along that axis in that direction. If it was not, we would simply move on with the game loop. We also had to create ways to ensure that, if a player was not bounded below by a platform, the player would move downwards with a gravitational acceleration. This gravitational acceleration had to be tweaked multiple times to feel natural, but, after many iterative changes, we felt the player's gravity emulated real life as much as possible. This solution worked in tandem with the code that prevented player movement along an axis if there was a collision because it would stop the player from falling if they landed on a tile. We learned from this that player collision detection was much

more complicated than what we thought. It is incredible how much computation goes on many times each second in the process of playing a video game.

Another major challenge involved in this project was finding a way to draw tiles onto the screen. After much research into the language mechanics of javascript, we discovered that the program was attempting to draw the tiles of the game map onto the screen before the images for the respective tiles were loaded. We initially tried to solve this by attaching `onload` parameters onto the images, finally arriving at setting `"load"` event listeners that would draw the image onto the screen when this event was triggered. This ensured that the image would not draw before it was loaded, as the image loading was the event that governed the drawing of the image. The problem wrapped in this was that the images were not being drawn at the correct locations. The underlying issue was that the proper values for the tile coordinates were not being saved in the event listener function correctly. After much research we came to this conclusion. This was solved using a javascript closure, which captured the value of the tile coordinates at the exact time the event listener was created.

For this project what we found most interesting was first the way we designed our levels. We researched many different ways to approach creating our levels. However, we found a way to assign a certain number to an object in our scene. For instance, if the

number equals zero, then it is air. If the number equals two, it is a block and so forth. Creating our levels this way gave us, as programmers, plenty of freedom to make the levels however we wanted. With that creative outlet, we made five completely beatable, yet hard levels.

Another aspect of our project that was fun to work on was creating our sprite. A sprite is a 2-dimensional bitmap that is utilized within a bigger scene. An example of this in action would be for game characters. These characters may have a rigid body or may have other attributes which contribute to its own physics. For the head of the sprite we used Professor Mohan's Face. The body of the sprite is based off of a very famous game called Minecraft. We got inspiration from the main character of the game, Steve, and used his body to fully create our sprite. To continue, making the game utilizing Flask was an appealing idea that, again, did cause quite a few problems yet was very rewarding in the end. Flask in itself is quite interesting considering the fact that it is able to deploy a static website using python. Typically, websites use Javascript and CSS as well as HTML. Python is a language which does not seem to be too common with website deployment. Although, it was quite an experience to work with Flask because it exposed us to something we are not used to. Thus, we were able to learn more considering the challenge. There are many other aspects of our platform game that were exciting and interesting to make.

Running the game through the web application and seeing it work was extremely relieving. Watching our hard work and a crazy idea we just had about a month ago work and have functionality was an accomplishment. Overall, this project was one that as a whole we all enjoyed creating. We were able to set goals for where we wanted our project to be. We also set artificial deadlines for implementing those goals as well as hold each other accountable for the duties that were assigned.

We did fulfill our initial goal in creating a platform game and completing track six of a student led project. We took into account opinions and suggestions from Professor Mohan. These suggestions were implemented into the game. Ultimately, we wanted to create a game which uses some of the core concepts that were presented in class. The idea of “bridging”, in this case, was perfect for our game, as we are using multiple languages to build our game. If we had longer time period for this project and were able to do it again, we would have developed a native app, made the game loop more efficient, made more sprites and tiles, and lastly made a player position tracking method. We would also add more and much harder levels. More time would also allow for focus on finer details of the game. For this project, we have implemented the main core of the game which is a bit bare bones. To reiterate, we were faced with many

challenges and through hard work and effort we were able to complete our project and be successful in our implementation.