Akshai Srinivasan

Period 5

**Advanced Software Projects Project 1 Design Document**

**Overview:**

I will be designing a system that will allow me to play my own version of the Google “Dino Run” game, except using my own hand gestures instead of the keyboard. In the game, there is a dinosaur running on a track while obstacles like birds and cacti come towards it. The objective of the game is to jump and duck to avoid the obstacles as long as possible, while a score continues to tally up the longer the dinosaur is alive. In addition, as time goes on, the speed at which the obstacles will come towards the dinosaur will also speed up.

My variation of the game will involve changing the way input is passed into the game. Traditionally, the “up arrow” or spacebar is used to make the dinosaur jump, and the “down arrow” is used to make the dinosaur duck. In my version, I will map a hand gesture to each movement, so that I can move the dinosaur simply by holding up a different hand gesture to the camera. For instance, in order to make the dino jump, I might show the onscreen video a thumbs up and to make the dino duck, I might show the onscreen video a thumbs down.

I imagine this project having three main components or chunks: (1) the dinosaur game itself where movements are mapped to the keyboard, (2) the creation and implementation of a machine learning model that can detect and identify hand gestures, and (3) the mapping of the hand gestures to movements, and the creation of a system to run both tasks simultaneously.

I will also be using Python, due to it being the language I am most familiar with, and its versatility with the extensive libraries it offers.

**Dinosaur Game:**

To create the dinosaur game, I will be using Python’s PyGame library, which will allow me to create a user interface. The game itself is fairly simple to make. I will first load in all my sprites, (which I downloaded online, credits will be in citations). Since this is an endless game, I will be making it so that my dinosaur stays in place while the track, which is double the length of the screen, moves towards the left, getting faster and faster as time progresses. The same will be true with my obstacles (small cacti, big cacti, and birds).

I will have objects for the dino, cloud (to make it move to the left with landscape), small cactus, big cactus, and bird. The last three will all go under a separate object called Obstacle, which will be inherited by the other obstacles.

Dino:

My dino object has the following fields: the current image of the dino ducking, the current image of the dino running, the current image of the dino jumping, Booleans that represent each states of the dino (whether it is running, jumping, or ducking), the rectangle of the dino (which lets us access the dino’s x and y coordinates), and the dino’s jump velocity (I want the dino’s jumps to get faster as time goes on).

It will also have the following methods: update, run, jump, duck, and draw. Update will call the respective jump, duck, or run functions based on the current states, and will update the states based on the detected gesture. Ex: if I make a “thumbs up”, the dino will update my jump state to True, and set my run and duck states to false. The run, jump, and duck methods will each move the dino. The jump function changes the dino’s y coordinates, and the duck function will change the dino’s x and y coordinates, since the ducking sprite is horizontal. The draw function will simply replace the current image of the dino with a new one in its new location.

Cloud/Obstacle:

Cloud and Obstacle only have update and draw methods. Update adjusts the x coordinate, and draw replaces the cloud to its new location.

Small Cactus/Large Cactus/Bird:

These objects are all similar and inherit the obstacle object. They will each only have a constructor to change the image and set its y position.

Main:

Finally, in my main function, I will have all my initializations, like my dino and clouds, the functions that will manage game over, the scoring, the movement of the track to seem endless, etc. In addition, I will have a loop that will be set to true while the dino is running, and inside, the dino will constantly be updated, then drawn, so that it can seem like it is moving, and can easily respond to inputs.

**Machine Learning Model for Gesture Recognition:**

Originally, I was going to train my own neural network capable of detecting gestures, but I was a bit worried by the amount of time this would take due to our two-week deadline. But after doing some research, I discovered an API called MediaPipe, which was made by Google, that I can access in Python. Using the API and downloading the pretrained model from their website, I should be able to detect gestures using my desktop camera without training anything myself.

In the end, I ended up going with this API because of its accessibility (it is compatible with Python) and because of how accurate it is (on the website, it said it was trained using 30,000 images). I referenced the documentation on the website in making this project. In addition, it is pretrained to detect 8 common gestures (including thumbs up and thumbs down), which means I likely won’t need to train the model to detect any additional gestures.

It uses a live camera stream, which I can use OpenCV for. Basically, my code will open and initialize my desktop camera, pull up a video feed on the screen, capture each frame, treat the frame as an image, apply the model, and return the gesture as a giant block of text. I will likely have to parse through this in order to get the gesture alone as a single stream. Because of its frame rate (~30 frames per second), it should be able to react and detect my gestures quickly enough to make my game playable.

**Putting the Components Together:**

To put the components together, I will have a small live stream playing in the top left-hand corner of my game screen. This will allow me to see my gestures to ensure that they get detected by my system. I will also create a function that will return the gesture from my model upon inputting a frame. I will then pass this gesture as a parameter into my function in the dino object that moves the dino according to inputs. Then, with a bit of fine tuning, the project should be complete.

Note: I later realized that sometimes, it can be difficult line up your hands to the point of detection, so I will likely add a function to not start the game until a valid gesture is detected. That way, the player knows where to place their hands to be detected.

**Citations:**

<https://www.youtube.com/watch?v=wnBGG7JLrkg>

<https://developers.google.com/mediapipe/>

<https://docs.opencv.org/3.4/index.html>