Positional Tracking Based Video Game

Term Project Report
CMPT 732 G-200

Kritu Patel (301464281)
Liam Sparling (301469200)
Abhishek Sashi Nair (301431164)

Supervised By
Ali Mahdavi Amiri

December 13, 2021
School of Computing Science
PMP Visual Computing

Contents

1 Introduction

1.1 About the project

2020 and 2021 have been difficult years. People have been forced to live restricted lives through the countless lock-downs and social distancing mandates needed to help restrict the spread of the Covid-19 virus. These mandates have left quite a toll on the mental and physical health of the population [4]. We have endeavoured to build a solution, that we hope will help alleviate the stress due to these restrictions. Our goal has been to create an immersive movement based gaming experience that can also double up as a fun workout routine. This project aims at making exercise a fun activity that can be done from the comfort of one's home.

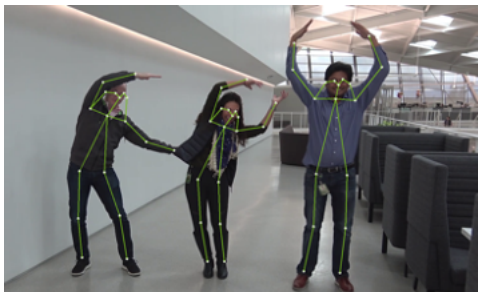

Figure 1: Positional tracking sample output, from [3]

The solution that we have built is a video game that utilises positional tracking. While playing the game, the user will be required to perform various poses and actions - using only their body without the need of additional controllers. An example of possible poses is shown in Figure 1. We have developed a simple game with this type of user interaction given the two month timeline.

1.2 Objectives and scope

Through the discussions held prior to commencing the development, the following were decided:

1. **Controller-free interaction with the game -** Keeping in view the user experience, the team decided that the entire game experience has to be free of controllers. The primary reason being that the game has to be an experience akin to a workout. Use of additional controllers would be degrading to the experience in this regard. Additionally, use of controllers would entail customisation and configuration efforts that would eat into the time required to develop the core logic of the game.

2. **Simple movements only for the gameplay -** The game play will need only simple movements such as shifting sideways, ducking etc. The primary reason

behind this decision is to maintain an easy learning curve for the user. Additionally, a time crunch for delivery of the project means that scope of development has to be kept precise and realistic. To keep things interesting, it was decided that as the gameplay progresses, the difficulty level would increase significantly and would require the users to make quicker movements.

2 Solution Approach

2.1 High Level Architecture

There are two major aspects to this project - user positional tracking and the development of a video game world. An existing open source model was picked as a starting point for positional tracking. For the development of the game world, Unity was chosen, due to ease of use and abundance of community support. The positional tracking module and the Unity modules exist in separate process spaces, and are connected by using a simple client-server architecture, with Unity acting as a server and the positional tracking module acting as the client. The positional tracking module needed significant enhancements to send the user body positions to the Unity modules.

To state it formally, we have developed a video game world that utilizes real-time positional tracking to map user positions to an in-character persona. The positions would be captured in real-time through a video capture device placed in front of the user at a position such that it is possible to track the user's entire body frame. These positions are then utilized to navigate the in-game character through an environment that generates obstacles. The aim of the game would be to avoid the obstacles and gain points.

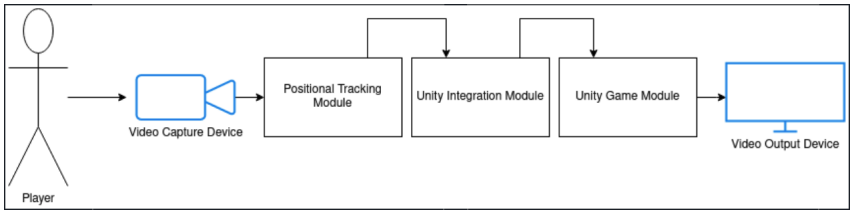

Figure 2: High-level architecture

As seen in Figure 2, the user's positions would be tracked in real-time by use of a video capture device such as a webcam. The positions will be determined using a positional tracking module first, which will be then passed onto the core modules responsible for the in-game behaviour. The resulting actions of the in-game character will be displayed to the user through the output device - typically, the monitor. The user would be able to view the points that they have earned at any point of time during gameplay. We have taken care to include messages and instructions at various points in the game workflow to ensure a smooth experience.

2.2 Work timeline and milestones

Our project solution has had four major milestones to ensure its completion.

- **Implementation of General Positional Tracking into Unity (Oct. 24)**
  This milestone involved finding existing open-source positional tracking code as a starting point, and modifying it for integration into, or with, Unity.
- **Linking of Positional Tracking Outputs into Game Parameters (Nov. 7)**
  The outputs from the positional tracking algorithm were mapped to skeletal movement and interaction with the environment of the video game.
- **Development of a Demonstration Map (Nov. 21)**
  A small demonstration of the game was to be ready by this date. This milestone involved placement and interaction of Unity assets and making a video game world. This demonstration was intended to allow testing and debugging of the interface for finalization for the last milestone.
- **Increasing of Game Size and Completing the User Interface (Dec. 5)**
  Moving from a simple game demonstration to a more complete short video game. User interaction and experience were finalized in this stage.

3 Implementation

3.1 Positional Tracking

The positional tracking module has been implemented in Python. The starting point for our module is the BlazePose python pose detection model that uses the `mediapipe` library.[5][1] It detects various landmarks on the body and is a model matured using machine learning. Corresponding to each landmark, 4 values are returned - the X, Y, Z co-ordinates and a visibility metric. The landmarks that are detected are shown in Figure 3.
The 33 landmark point values are transferred to Unity using a socket, as depicted in Figure 4. The Unity module will open up a socket on the loopback address and act as a server, while the pose detector will remain connected to this socket as a client, and send the information retrieved from each frame of the input from the video capture device.[2]
Once Unity receives the landmark information for a particular frame, an averaging computation is performed for each body part, among the circled points, as shown in Figure 5. The in-game character will have blocks corresponding to each body part, and the averaging computation gives a good estimate on where each block would be positioned.

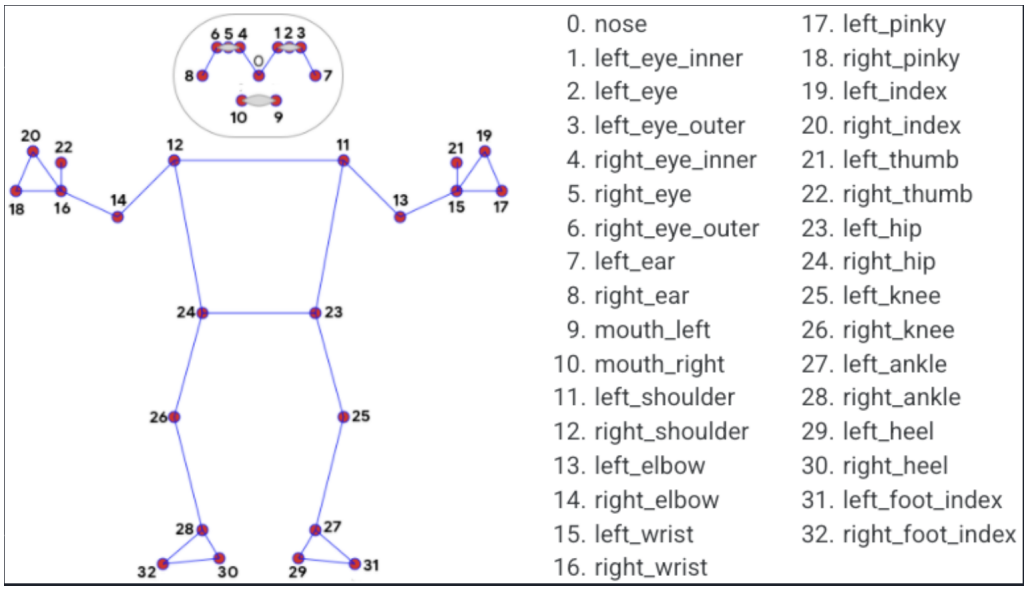

Figure 3: Landmarks detected on the body

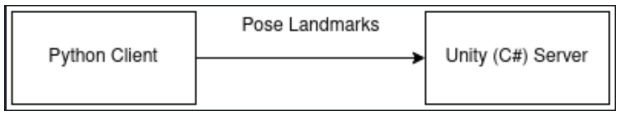

Figure 4: Transfer of information from pose detector to Unity

3.2 Game Workflow

The main objective of the player is to avoid obstacles generated by the video game world, and gain points in the process. Now, we will look at the workflow from the user's perspective, in a step-by-step manner.
(**Note:** The figures in the upcoming sub-sections will also include the positional tracking output on the screen for clarity - this is not intended to be a part of the final game.)

3.2.1 Calibration and choosing game mode

When the game starts, the user will see the menu shown in Figure 6. As the message on the top of the menu suggests, the user needs to calibrate with respect to the camera first, as shown in Figure 7. After 3 seconds, the game character will appear on screen, as shown in Figure 8. The user is now free to choose a game mode by orienting both hands in the direction of the respective mode, as shown in Figure 9.

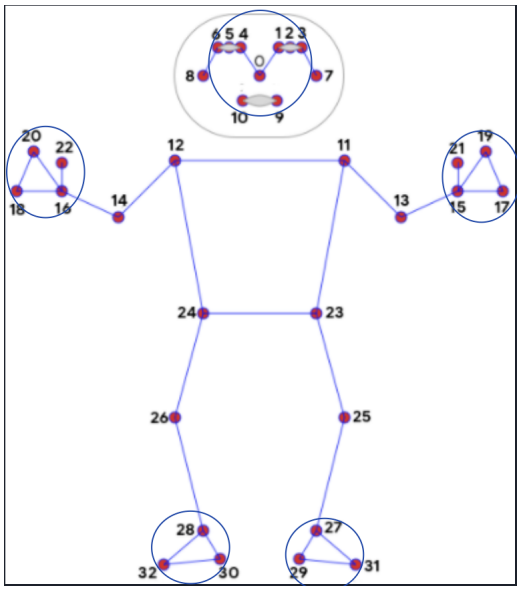

Figure 5: Computation of average positions

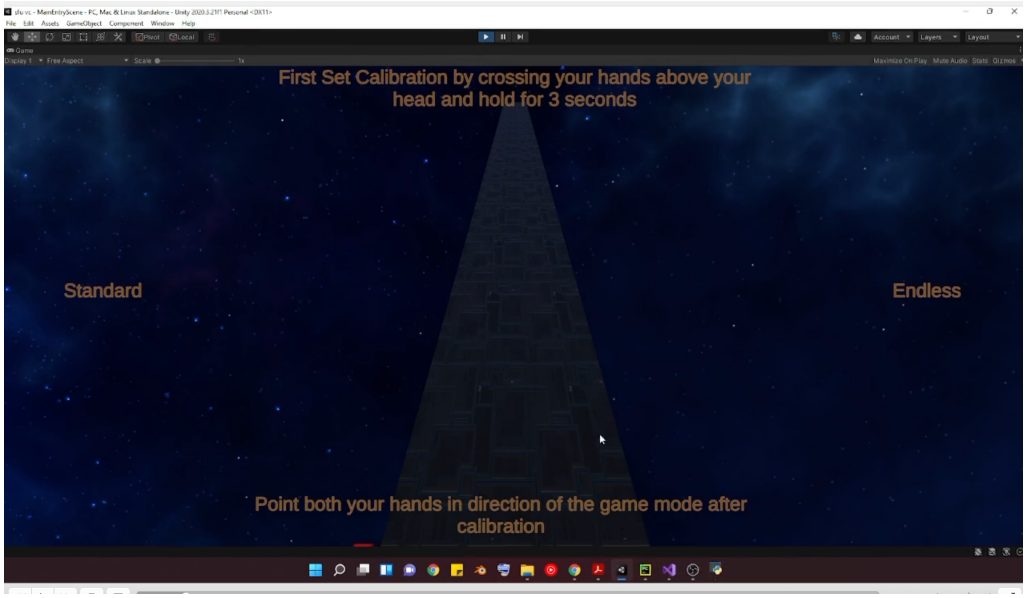

Figure 6: Game Menu

3.2.2 During gameplay

As the game progresses, the user will have to shift sideways or duck to avoid the incoming obstacles. As the game progresses, the speed of the incoming obstacles will

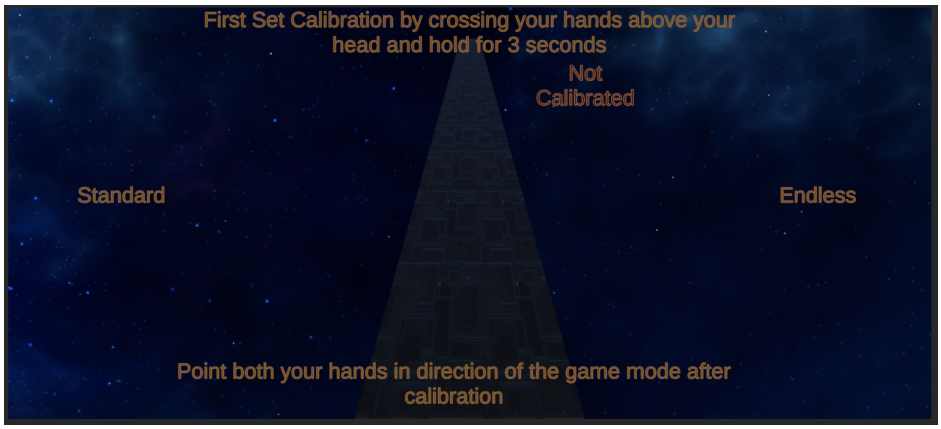

Figure 7: Prior to calibration

Figure 8: Post calibration

increase significantly. Examples of the movements are shown in Figures 10 and 11. Avoiding the biggest obstacle with result in a gain of 60 points, whereas for the smaller obstacles, the gain is 30 points. A warning message is shown if the user goes too far to right or left, as shown in Figure 12.

3.2.3 End of the game

Once the character collides with an obstacle, the game will end and the final points tally will be shown to the user, as seen in Figure 13. The user go back to the menu by raising both hands, as shown in Figure 14.

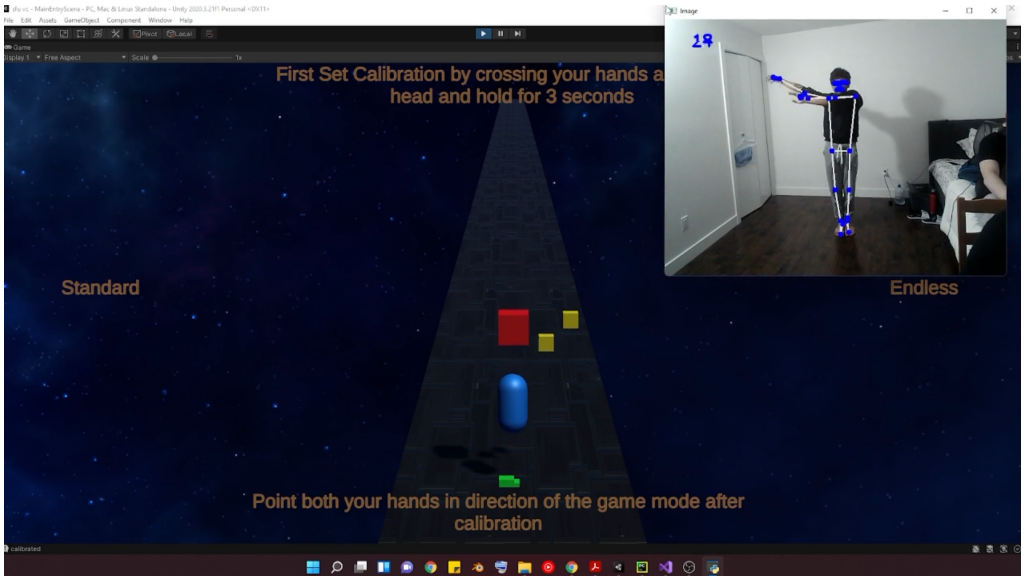

Figure 9: Choose Game Mode

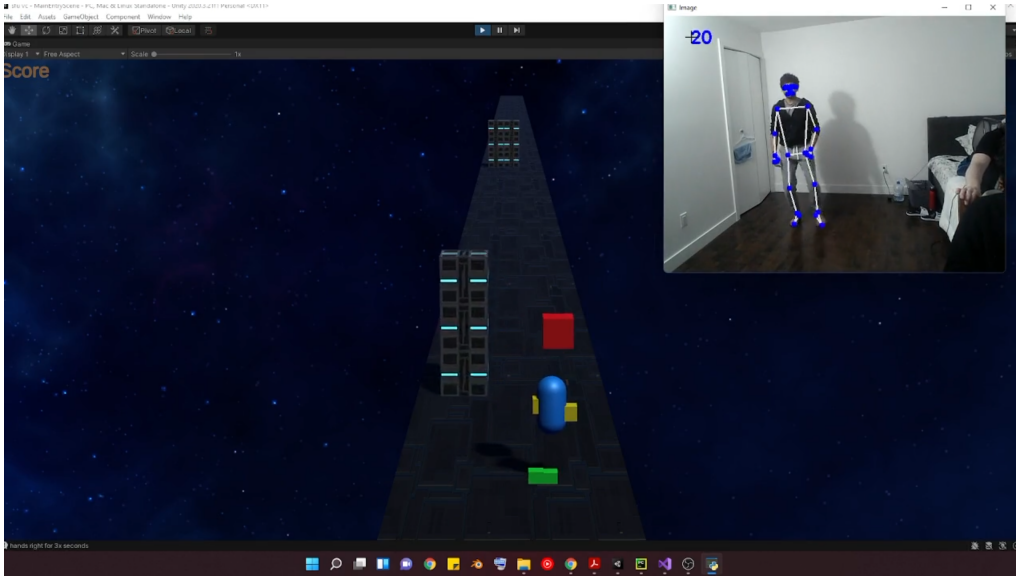

Figure 10: During the game - 1

3.3 In-Game Aspects

In this section, we will explain some of the core in-game aspects that are crucial to the overall experience.

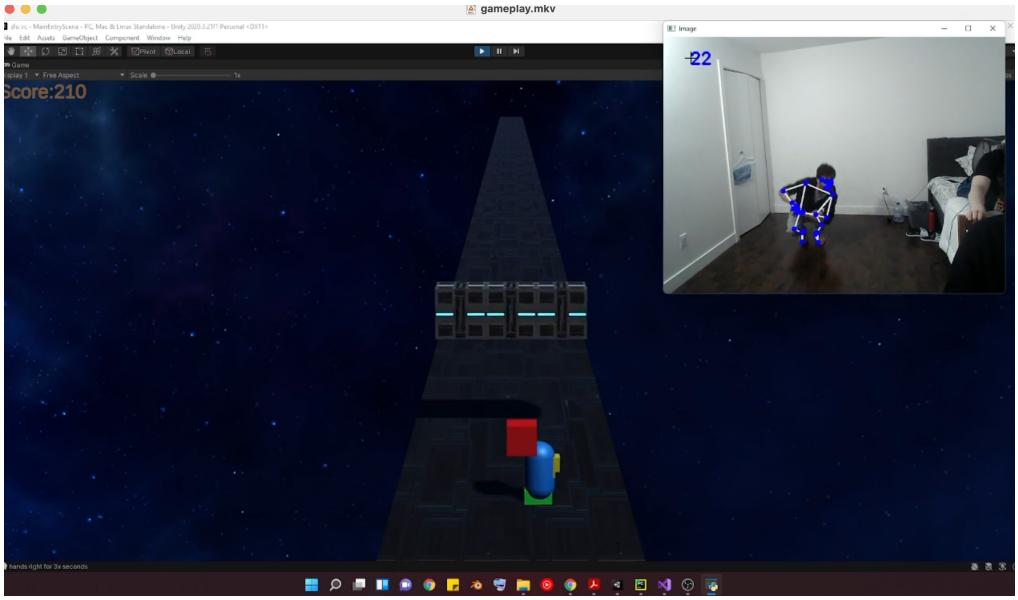

Figure 11: During the game - 2

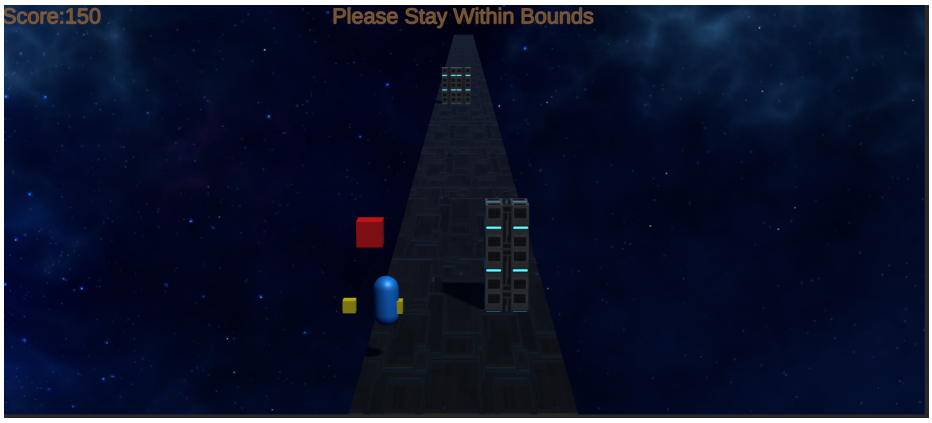

Figure 12: Warning message to stay within path boundary

3.3.1 Obstacle Generation

Depending on the mode of play chosen, the user will face obstacles in a particular fashion. As the number of obstacles faced increases, the speed also increases significantly. For the Standard mode, a fixed sequence of obstacle generation will be followed ie. the type of obstacle and the order in which it will come towards the game character will be the same. Additionally, the mode will have a fixed termination point. In effect, the Standard mode is a practice mode of sorts that is particularly useful for new users.

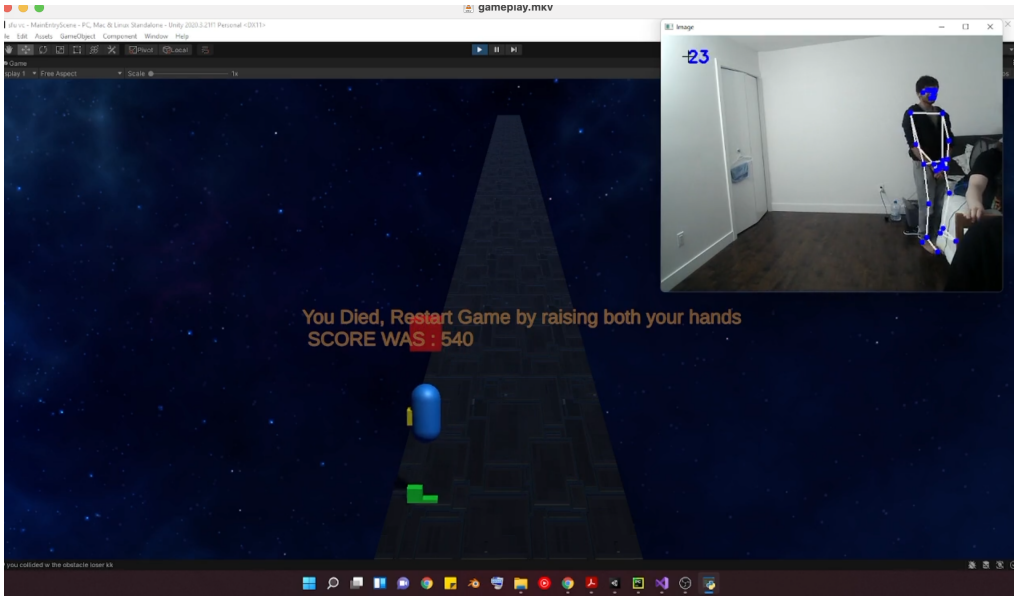

Figure 13: Death screen

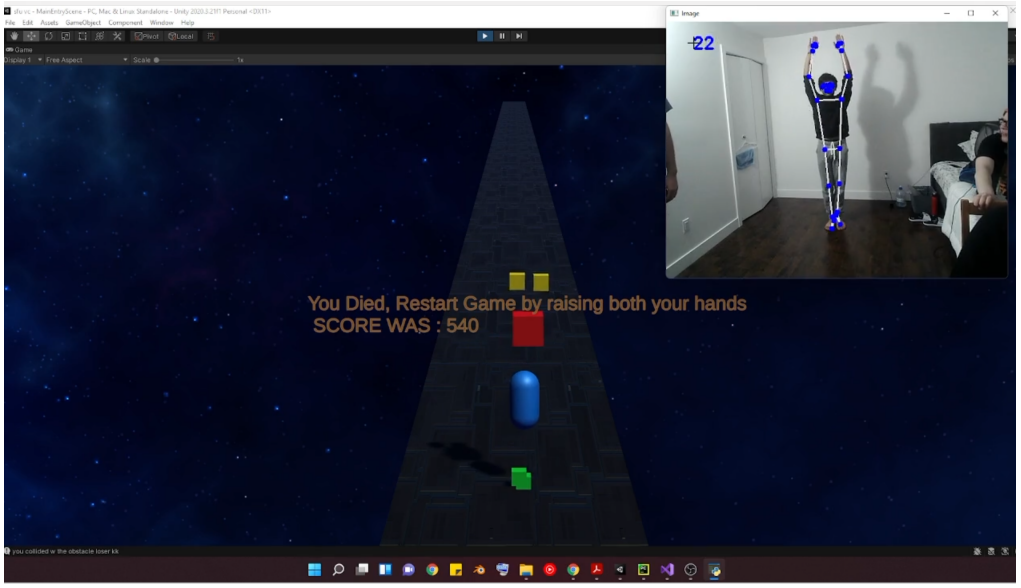

Figure 14: Game restart

In the Endless mode, things are made a little more complex. Obstacles will be generated at random and the game will terminate when the user collides with an obstacle.

This mode is intended to be the true experience of the game, and showcases the true potential of the solution.

3.3.2 User Interface

As shown earlier in Figure 6, the game menu serves as the entry point to the game, and also gives directions on the pre-game calibration. The interface has been built using a variety of open source textures. The user interface design also includes a score component that will be visible while the game is in progress. Additionally, a screen that shows the user the final score has also been added, as shown in Figure 13.

3.3.3 Calibration

The calibration requirement is a necessary step to ensure smooth gameplay. The video capture device used and the user's dimensions will need to be gauged to a certain level of accuracy for the game to function as per expectations. Since the game requires that user puts in a certain amount of physical effort, and given the fact that the surrounding space may vary for each user, the calibration becomes an important step prior to commencing the game.

3.4 Challenges

In this section, we will list out some of the challenging aspects of the implementation:

1. **Support for jump action -** Implementing the jump action support was challenging for a variety reasons - unlike the other actions, jump involved a quick return of the user to the previous spot. Additionally, collision between the game character and the user becomes a complex interaction in the case of jumping. The game character had to be fine-tuned so that when the user executes a jump that would be ordinarily sufficient to clear the obstacle, the game character would follow suit.

2. **Scaling the environment -** Regardless of the ongoing speed, the obstacles must be avoidable to some extent ie. some space must be left on the path for the user to position themselves. Additionally, bounds had to be put in to ensure that the game character would not leave the main path on which obstacles come in. Smaller play areas in the real world may cause issues in terms of dodging the obstacles - this also had to be accounted for, in the implementation.

3.5 Possible enhancements

In this section, we will list out some of the enhancements that are further possible for the solution:

1. **Addition of difficulty levels -** The possibility of choosing a difficulty level with a know range of obstacle speeds would definitely add more value to the game and increase the user-friendliness.

2. **Multiple themes -** In order to accommodate varying tastes, introduction of more environments with alternative themes would be a welcome addition.

3. **Implementation in VR -** Taking things one step further, the game could be enhanced to be a Virtual Reality solution. In this case, the users would themselves be avoiding obstacles as opposed to controlling a character that does. This would be a definitive step-up in the experience.

References

[1] Mediapipe pose.

[2] Canyoucatchme01/csharp-and-python-continuous-communication. 2020. Viewed online; 10th October, 2021.

[3] Training and optimizing a 2d pose estimation model with nvidia tao toolkit, part 1. 2021. Viewed online; 10 October, 2021.

[4] Dai et al. The influence of covid-19 pandemic on physical health–psychological health, physical activity, and overall well-being: The mediating role of emotional regulation. *Frontiers in Psychology*, 12:3005, 2021.

[5] Syed Abdul Gaffar Shakhadri. Pose estimation using opencv. 2021.