**6**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Implementation**

* + 1. **Implementation Platform**
    2. **Module specifications**
    3. **Outcomes**
    4. **Result Analysis**

**6.1 Implementation Platform**

**Frontend:** Unity UI

**Backend:** Python & C#

**Code Editor:** Visual Studio & Anaconda(Spyder)

**6.2 Module specifications**

**Modules used in Python**

* **OpenCV**: It is an open-source computer vision library that can be used for detecting and tracking hand gestures. There are Python bindings available for OpenCV, which can be used in conjunction with Unity to implement gesture control.
* **TensorFlow:** It is a popular machine learning library that can be used for gesture recognition. There are Python bindings available for TensorFlow, which can be used to train a model for recognizing specific hand gestures.
* **Mediapipe**: It is a cross-platform, customizable framework for building machine learning pipelines to process audio, video, and other sensor data. It provides a suite of pre-built solutions for tasks such as hand tracking, pose estimation, and facial recognition, which can be integrated with Python and Unity.
* **PyAutoGUI**: It is a Python module for automating GUI tasks, such as moving the mouse, clicking on buttons, and typing text. PyAutoGUI can be used to simulate user input based on the detected hand gestures, which can be sent to the Unity game engine.
* **NumPy**: It is a powerful numerical computing library for Python, which provides support for large, multi-dimensional arrays and matrices. NumPy can be used for processing the image data from the gesture control device, such as cropping, resizing, and normalizing the images before feeding them into a machine learning model.

**Modules used in Unity**

**Input** **System**: This module provides a flexible and powerful system for handling user input in Unity, including support for gamepad, keyboard, and mouse input. It can be used to receive input from gesture control devices and map them to specific game actions.

**Animator** **Controller**: This module provides a tool for creating and managing animation states and transitions in Unity. It can be used to create animations for the game objects that respond to the user's gestures.

**UI** **System**: This module provides a set of components for creating and managing user interfaces in Unity, including buttons, sliders, and text fields. It can be used to create the user interface for the gesture control system.

**Graphics** **Engine**: Unity's graphics engine provides a powerful toolset for creating visually rich and engaging games, including support for 2D and 3D graphics, shaders, and post-processing effects. It can be used to create the visual feedback for the user, such as displaying a hand or an avatar that responds to the user's gestures.

**Scripting** **System**: Unity's scripting system provides support for multiple programming languages, including C#, which can be used to implement the logic for processing the input from the gesture control device or software, and then sending commands to the game engine.

**6.3 Outcomes**

* Below Fig’s 6.3.1 and 6.3.2 represent python outcomes
* While Fig 6.3.3 represent Unity outcomes

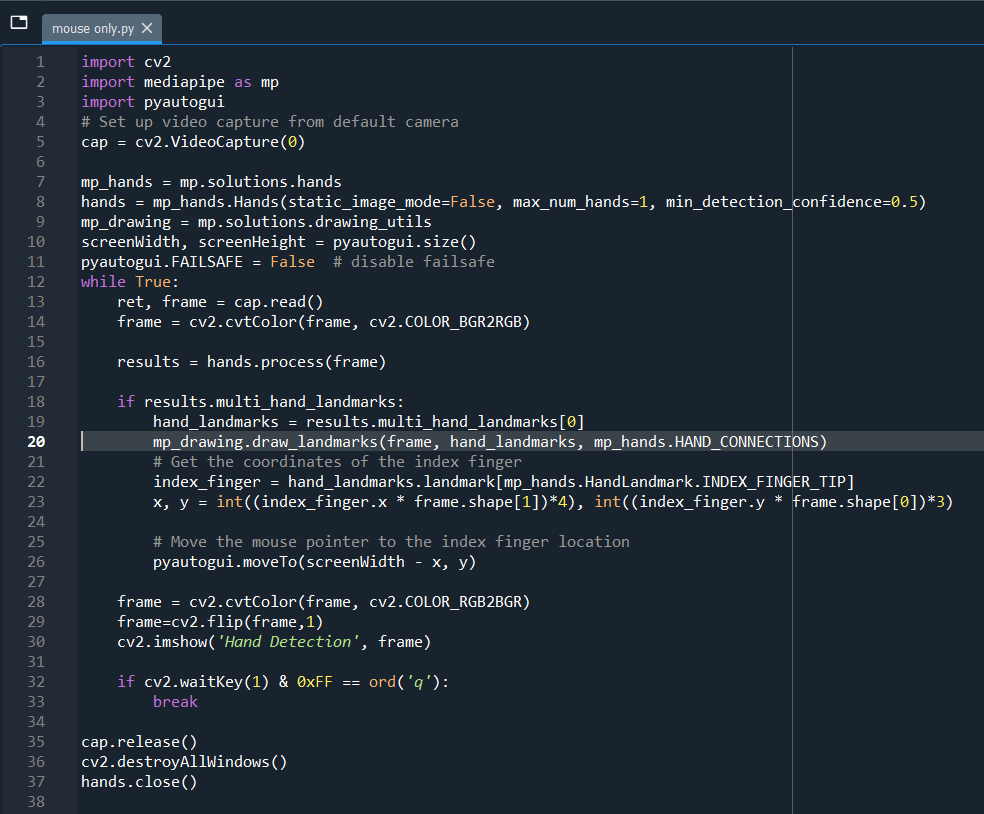
****

Fig 6.3.1 Python Script

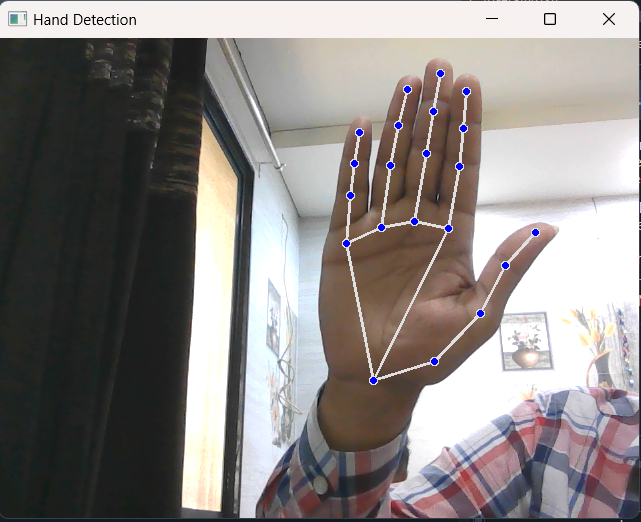
****

Fig 6.3.2 Python Outcome (Hand Track)

****

Fig 6.3.3 Main Screen (Unity)



Fig 6.3.4 Screen short of ongoing game(a)



Fig 6.3.5 Screen short of ongoing game(b)



Fig 6.3.6 Game over Screen (Unity)

**6.4 Result Analysis**

It is also important to gather feedback from users of the gesture control system, and to evaluate the system's usability and user experience. This can be done through user surveys, interviews, or usability testing sessions, where users are asked to perform specific tasks using the gesture control system, and their feedback is recorded and analyzed.

To analyze the results of a gesture recognition system, we can use a variety of metrics, such as:

* Accuracy

This metric measures the percentage of correctly detected gestures. It can be calculated by comparing the predicted gesture with the ground truth gesture for a given input.

* Precision and Recall:

These metrics measure the ability of the system to correctly detect positive (gesture present) and negative (gesture absent) cases. Precision measures the percentage of true positives (correctly detected gestures) out of all the detected gestures, while recall measures the percentage of true positives out of all the actual positive cases.

* F1 Score:

This metric is a combination of precision and recall, and provides a single measure of the overall performance of the gesture control system. It can be calculated as the harmonic mean of precision and recall.

* Confusion Matrix:

This matrix provides a detailed breakdown of the number of true positive, false positive, true negative, and false negative cases for each gesture. It can be used to identify the most commonly misclassified gestures, and to fine-tune the system parameters accordingly.

* Reaction Time:

This metric measures the time taken by the system to respond to a detected gesture. It can be used to evaluate the system's responsiveness and overall performance.