COMP 4801 Final Year Project [2023/24]

FYP 23008

Interim Report

Exploring the application of machine learning models to achieve advanced character control and improve the naturalness and immersion of virtual characters in games

Yu Ching Lok

Supervised by Dr. Choi, Yi King

Abstract

Currently, AI-based avatar control methods often result in imprecise body movements and restricted emotional expression, thus resulting in a lack of natural interaction between the user and the avatar. To enhance the interaction between the user and the avatar, we propose to utilize multiple AI models for advanced avatar control. Our application combines these models to achieve precise control of avatars. It consists of two programs: one focusing on AI-related tasks such as landmark detection, emotion recognition, and gesture recognition; and the other focusing on controlling avatars using Unity's built-in functionality. During runtime, these programs run in parallel via TCP communication. The results of the AI detection are used to control the avatar's body movements, facial expressions, emotional expressions and trigger specialized actions. We have researched and developed the AI model and avatar control mechanisms into a preliminary pipeline application. Ongoing tests and experiments have identified several difficulties that need to be addressed. The remaining development time will be used to address these difficulties, finalizes the application, and carry out further research and improvements to the AI model. This project aims to overcome current limitations and enable a more natural and interactive user experience.

Table of Contents

Abstract ..... **II**

Abbreviations ..... **IV**

1 Introduction **1**

2 Project Objectives and Deliverables **3**

2.1 Application 4

2.11 AI Model Detection Program 4

2.12 Virtual Character Control Program 4

3 Project Methodology **7**

3.1 Methodology of AI Modeling 7

3.11 Precise Control of Avatar's Body Movement and Facial Expression 7

3.12 Emotion Expression and Specialized Action Triggering of Virtual Avatar 8

3.2 Methodology of Virtual Character Mechanism 14

3.21 Precise Control of Avatar's Body Movement 14

3.22 Precise Control of Avatar's Facial and Emotion Expression 15

3.23 Specialized Action Triggering of Virtual Avatar 16

3.3 Summary of Methodology 17

4 Project Schedule and Progress **18**

5 Difficulties, Problems Encountered and Proposed Solutions **21**

5.1 Fluctuation of Landmark Detection Results 21

5.2 Poor Performance of Self-Developed Emotion Recognition Models 22

6 Conclusions **24**

References ..... **V**

Appendix ..... **VI**

Abbreviations

Abbreviation	Definition
AI	Artificial Intelligence
EAR	Eye Aspect Ratio
HaGRID	Hand Gesture Recognition Image Dataset
MAR	Mouth Aspect Ratio
MLP	Multi-Layer Perceptron
ROI	Region Of Interest
TCP	Transmission Control Protocol
VRM	Virtual Reality Modeling Language

1 Introduction

As Artificial Intelligence (AI) technology continues to advance, its applications have expanded to various fields, including entertainment. One notable application of AI in entertainment is the emergence of virtual YouTubers. These virtual characters are controlled by the user through landmark detection technology that captures facial expressions and body movements through camera input (Singh, 2023).

However, current methods of controlling avatars using artificial intelligence have certain limitations. Existing landmark detection models only allow for basic body movements and simple facial expressions. Expensive motion capture systems are required to enable more advanced control of character movements (Gank Content Team, 2023). Additionally, the process of changing an avatar's emotional expressions currently relies on manual input (Gank Content Team, 2023), resulting in somewhat unnatural expressions. These limitations restricted the interaction and naturalness between users and avatars.

To address these limitations, our project aims to explore how to integrate different AI models to control virtual characters, rather than just relying on landmark detection models. Specifically, we intend to combine emotion detection and gesture recognition models for more advanced control. By combining these AI models, we hope to provide users with a more unique and natural experience when controlling virtual characters.

To achieve our goal, we will develop an application providing advanced control over virtual characters. The backend of the application will be powered by multiple AI models, each responsible for controlling specific features of the virtual character. Importantly, our approach eliminates the need for an expensive motion capture system, as the application only requires a single camera for input. By prioritizing affordability while maintaining a high level of control, we aim to provide a better user experience without significantly increasing costs.

The following sections of the report will be organized as follows. The report begins with an introduction that describes the objectives of the project and outlines the deliverables. Subsequent sections outline the methodology used to develop the application. The report then discusses the progress made to date and outlines upcoming timelines and milestones. Next, the report delves into the challenges and

obstacles encountered during the development process and suggests appropriate solutions. Finally, the report concludes with a short summary highlighting key findings, achievements and concluding remarks.

2 Project Objectives and Deliverables

This project has three primary objectives. First, the project aims to investigate how AI models can be applied to enhance the control of avatars in a natural and unique way. Second, the project aims to evaluate the impact of AI models on avatar control, user interaction and overall user experience. Finally, the project aims to provide advanced control of avatars without the need to rely on expensive motion capture systems, thus providing an economical means of achieving enhanced control of avatars.

To achieve these objectives, the project will employ a variety of AI models to control avatar-specific features in an affordable manner. The impact of these models on avatar control will be thoroughly evaluated, considering their impact on user interaction and overall user experience. By conducting these evaluations, the project aims to determine the effectiveness of utilizing AI models to provide a more natural and unique avatar control experience.

The outcome of this project will be an application that integrates a variety of AI models to enable the user to have advanced control over the avatar and enhance the overall interaction between the avatar and the user. The application is expected to provide the following functionality:

1. Precise control of the avatar's body movements
2. Manipulation of the avatar's facial expressions
3. Express various emotions through the avatar
4. Execution of specialized actions by the avatar

By implementing these features, the application aims to give users a more immersive and engaging experience when interacting with virtual characters.

The following sections will focus on the practical details of the application, first describing the overall structure and workflow of the application, and then detailing what the program does and how it works.

2.1 Application

The application under development consists of two different programs: one for AI model detection and the other for virtual character control. The two programs will run independently in parallel and communicate with each other over a Transmission Control Protocol (TCP) connection. The general workflow is as follows: first, the camera will provide a real-time stream of camera data to the application program. The AI model detection program will listen to the camera data stream and, upon receiving new image inputs, these images will be delivered as inputs to the AI model for detection. After the detection process, the results will be stored and transferred to the virtual character control program via TCP. Once the update results are received, the virtual character control program will update its control signals to influence and trigger different features of the virtual character (Figure 1).

2.11 AI Model Detection Program

The AI model detection program will be implemented in Python and will focus only on tasks related to AI model detection. Python was chosen as the programming language because it is widely used and has good support for AI models development with extensive libraries and frameworks (Beklemysheva, 2022). Once the program is executed, it should run continuously and actively listen for incoming images. After receiving the image, the program will start the detection process, carrying out emotion recognition, landmark detection, and gesture recognition. The detection results will be stored for subsequent communication with the avatar control program.

2.12 Virtual Character Control Program

The virtual character control program will be written in C# using Unity as the main engine. It will be responsible for controlling and displaying the 3D avatar. The decision to use Unity as the platform and C# as the programming language was made primarily because of Unity's advanced support for 3D avatar control with its built-in animation system, character control system and physic engine (NeuroSYS, 2023). The program consists of three main parts, the first is the TCP communication with the AI model detection program, the second is the calculation of control signals based on the AI model detection results, and the last is the control of the virtual avatar.

The program sets up a TCP server that acts as a client receiver for the Python program. This server continuously listens for the latest results sent by the AI model detection program. The control signals of the avatar are then computed and adjusted based on these received results. The modification of the control signals relies heavily on the results detected by the AI model, which help determine the adjustments required

by the avatar. For example, the rotation vector responsible for the rotation of the avatar's skeleton is derived from landmark data results. In addition, emotion IDs were mapped to specific modifications of facial features, while gesture IDs trigger corresponding animations. These aspects will be further elaborated later in this report. This iterative process ensures that captured actions and intentions are accurately transferred from the AI model to the virtual character program. By continually adjusting control values and updating the avatar, the body movements, facial expressions, and emotions of the avatar can be accurately and efficiently reflected in the virtual character, ensuring consistency with the intended image.

To implement virtual character control, we first need to obtain a 3D humanoid model. In our program, we use an application called “Vroid Studio”, which allows the user to create custom 3D models of humanoid avatars. The main reason we chose to use “Vroid Studio” is that it is free, easy to use, and allows us to customize and export the 3D humanoid models we want. It has a user-friendly interface that allows us to easily modify and create the 3D humanoid models (Pixiv Inc.). The generated model will be exported in the Virtual Reality Modeling Language (VRM) format, a platform-independent file format designed for 3D characters and avatars in modern VR environments (VRM consortium Inc.) (Figure 5).

To import the humanoid model into Unity, we use a specialized extension package called UNI-VRM. This package helps to import VRM models into Unity, converting them into Unity assets equipped with existing functionality and scripts. Using this package, we were able to control the movement, animation, and facial expressions of the humanoid models within the Unity environment, utilizing the built-in functionality. (GitHub - VRM-C/UNIVRM) (Figure 6).

After importing the 3D humanoid model into Unity, we will utilize Unity’s built-in features to control the virtual avatar. For skeletal movement, we use the Animated Rigging package to apply rigging to the humanoid model to achieve realistic skeletal movement (Figure 7).

Additionally, for facial and emotional expression, we utilized the skin mesh rendering functionality inherent in humanoid modeling assets. Utilizing this feature of Unity, we were able to dynamically deform the skin mesh of the avatar at runtime. Unity then renders the modified mesh, allowing us to effectively control the avatar’s facial movements and emotions (Figure 8).

By using these extensions and taking full advantage of their functionality, we should be able to effectively control the movements, facial expressions, and emotion expressions of our avatars to create immersive and realistic virtual character experiences.

In conclusion, this project aims to explore the use of artificial intelligence models to enhance avatar control in a natural and unique way. By integrating multiple AI models, the application seeks to provide precise control over body movements, manipulation of facial expressions, expression of emotions, and execution of specialized actions. The goal is to create a more immersive and engaging experience for users when interacting with virtual characters.

3 Project Methodology

The implementation of effective control mechanisms is essential to achieve advanced control over virtual avatar and to enhance the immersive user experience. Seamless interaction between the user and the avatar can have a significant impact on the overall experience for both the user and the audience. Furthermore, since the application is real-time, a fast update rate is required to ensure smooth rendering of the avatar's movements. Therefore, the application must accurately capture the user's movements and intentions from the camera and reflect them effectively in the avatar. This emphasizes the key factors that need to be considered when researching and developing AI models and control mechanisms for avatars, such as responsiveness, consistency, and reliability. In this section, we present methods for implementing advanced control of avatars considering the above factors. We will divide our discussion into two main sections: AI modeling and the virtual character control mechanism. Each section will cover the following topics: achieving precise control of the virtual avatar's body movement, facial and emotion expression, and implementing specialized action triggering of the virtual avatar.

3.1 Methodology of AI Modeling

In this section, we will focus on discussing the models we have chosen to achieve advanced control of the virtual avatar, as well as the reasons behind our selection, our approach to development, and how we utilize these models to implement avatar control effectively.

3.11 Precise Control of Avatar's Body Movement and Facial Expression

To achieve advanced control of virtual avatar movements and facial expressions, we rely on hand, pose and facial landmark detection models. In this project, we chose the Mediapipe holistic model as the primary landmark detection model. Several reasons influenced this decision. Firstly, our application relies entirely on image streams from the camera, which requires a landmark detection model that only process images as input, and Mediapipe fulfils this requirement. Second, our project requires efficient, lightweight models for hand, pose, and face landmark detection. Running separate models for each specific landmark detection is not ideal. Fortunately, Mediapipe provides a pipeline detection model that combines all three models into a single lightweight model, significantly reducing device effort and processing time. Finally, Mediapipe proved to be lightweight and efficient, recognizing facial, postural and hand landmarks in real-time, even on average-sized devices and web browsers. For example, it has achieved a frame rate of 20 FPS on devices such as the Samsung S9+

(Grishchenko & Bazarevsky, 2020), demonstrating its consistent and efficient performance in providing accurate and reliable landmark detection results. By utilizing the Mediapipe holistic model, we can efficiently detect facial and body landmarks and map them to the movements and features of the avatar. In this way, the avatar is able to accurately mimic the user's movements, enabling precise control of the avatar's body movements and facial expressions.

For more information on the Mediapipe Holistic Model, the Mediapipe Holistic Model is a comprehensive pipeline that integrates separate models for posture, face, and hand components, each optimized for its specific domain. The detection process consists of the following steps:

First, the model estimates human poses using BlazePose's pose detector and subsequent landmark models. This step captures the human pose, represented by a set of landmarks.

Then based on the inferred pose landmarks, the model identifies three regions of interest (ROIs) for each hand and each face. To improve the accuracy of these ROIs, re-cropping the model was also used.

Once the ROIs are identified, the model crops the full-resolution input frames to these regions and applies task-specific face and hand models to estimate their corresponding landmarks. This results in accurate detection of facial landmarks (468 facial landmarks) and hand landmarks (21 landmarks per hand) (Figure 10).

Finally, all landmarks in the pose model, face model, and hand model were combined to form a comprehensive set of 543 landmarks. This includes 33 pose landmarks, 468 face landmarks and 21 hand landmarks for each hand. We will use these landmark results for post-processing or apply them directly to the control of the avatar (Figure 9).

3.12 Emotion Expression and Specialized Action Triggering of Virtual Avatar

To enable the virtual characters to express emotions and trigger actions efficiently and naturally, we decided to utilize AI models to control the emotional expressions and trigger specialized animations of the virtual characters. These models are responsible for detecting emotions and gestures from image inputs and transferring the results to the virtual character control program for further processing. By integrating these

models into the application, we aim to track the user's emotions and intentions for a more responsive and realistic virtual character control experience.

Given that the application captures the user's body and face primarily through a webcam, the models we use must be able to adapt to this environment. In addition, we may need models to detect specific emotions or gestures, which may vary from situation to situation. Using pre-trained emotion and gesture recognition models may not directly meet our specific needs. Therefore, we are considering developing our own AI models to meet our requirements.

We have evaluated several approaches to build our own AI models. For the first approach, we are exploring the use of classification models that take images as input. Currently, we are testing models with different structures, such as models that include attention mechanisms or residual networks. The chosen model is likely to be a CNN-based classification model. This approach allows us to extract features directly from the image, ensuring that the model has enough information for accurate classification. However, this approach may introduce higher model complexity and potentially increase the processing time for detection.

Another approach we are considering is to use landmarks generated by the Mediapipe model as input to other models that performs emotion and gesture classification. By using only landmark data as input, we can significantly reduce the amount of information that the model needs to extract and classify. This should reduce the overall processing time of the model. However, the performance of this approach is still being evaluated and we are conducting tests and experiments to assess its applicability.

The final approach we are considering involves transfer learning or fine-tuning of pre-trained models. The advantage of this approach is that it reduces development time and training time. It provides a solid base model. However, since our requirements are specific to our situation and environment, most open source or publicly available models may not be directly applicable. We may need to fine-tune or modify the model to suit our requirements, which may require a lot of data and time.

In this phase, we have studied and experimented with different emotion and gesture recognition models. To evaluate these models, we used some well-established evaluation metrics such as F1 score, recall, precision, and accuracy. These metrics help us to evaluate the performance and effectiveness of the models in accurately

recognizing and classifying emotions and gestures. We have encountered a number of difficulties and have drawn our current conclusions based on the results of our research.

We have developed a gesture recognition model that utilizes hand landmark data as input. The model is based on a multilayer perceptron (MLP) classification approach. To enhance its generalization and improve performance on unseen data, we incorporate Dropout layers and batch normalization layers. The Dropout layer prevents overfitting by randomly deactivating neurons during training, while the batch normalization layer reduces internal covariate bias and stabilizes training.

For training, evaluation, and testing, we used the Gesture Recognition Image Dataset (HaGrid). This dataset contains 553,991 gesture images of the right and left hands divided into 18 gesture categories. The dataset includes images taken from 37,563 unique individuals between the ages of 18 and 65. These images were taken under different lighting conditions, including artificial light and natural light, similar to the environment in which our application is used (Kapitanov, 2022). This dataset provides a wide range of gesture images suitable for our requirements, allowing us to train a model tailored to our needs (Figure 11).

After extensive training, evaluation and testing, our model achieved impressive results. When running the gesture recognition model alone, the average processing time per hand landmark detection was approximately 4.67e-05 seconds. The overall accuracy of the test dataset reaches about 99%, with F1 scores, precision and recall values of about 0.99 for both hands (Figure 2 and 3). We also evaluated the real-time detection capability using hand landmark data from actual camera images, and the results show that the model can consistently and reliably recognize gestures at a high update rate. This indicates that our gesture recognition model is lightweight, has a short processing time, reliable and stable performance, and is suitable for our application.

We finally settled on this MLP classification model based on hand landmark data as an input to our gesture recognition system. Overall, the model performed well, but further testing may be required to identify any potential adjustments or improvements.

Initially, for the emotion recognition model, we used a similar approach to the gesture recognition model and built an MLP classification model using facial landmarks as input data. To train, evaluate and test the model, we chose the Facial Expression Recognition 2013 dataset (FER2013). This dataset contains approximately 30,000 facial RGB images and is labelled with seven different emotion categories: anger,

disgust, fear, happiness, sadness, surprise and neutral (PapersWithCode) ((Figure 12).

While FER2013 provides a large number of pure head images, this dataset contains variations in environmental conditions that may not be consistent with the controlled webcam environment we intend to use. Therefore, relying on this dataset alone for training may not be sufficient. However, we can still use it as a base dataset for training and later fine-tune the model using a custom dataset that closely matches the webcam environment we are targeting. By initially training the model on FER2013, we can obtain a baseline model that captures general emotion recognition patterns. Subsequent fine-tuning of this model using our custom dataset will allow us to adapt our model to a specific environment and achieve better performance in real-world webcam-based emotion recognition tasks.

However, the performance of the MLP emotion recognition model did not meet expectations. It has an overall accuracy of about 54%, with precision, recall and F1 score hovering around 0.55 (as shown in Figure 4). The model was able to correctly detect specific emotions, such as happiness, with over 90% accuracy. However, it performs poorly on other emotions (e.g., disgust and fear), showing no true positive results.

The evaluation metrics suggest that the model is overfitting, as although it is proficient at recognizing specific emotions. Overall, it struggles to accurately classify and recognize emotions. This suggests that the model is unable to extract essential features from facial landmark data to achieve reliable classification. Therefore, the inconsistency and lack of reliability of the model prevents it from being used in practical applications. Considering these limitations, using a multilayer perceptron (MLP) for emotion classification based on facial landmark data is not suitable for our task.

Considering that MLP classification based on facial landmark data may not be achievable, we started to develop deep learning models based on Convolutional Neural Networks (CNNs) for emotion classification based on raw image input. We extensively researched various approaches used by other researchers, including “POSTER V2: A Simpler and More Powerful Facial Expression Recognition Network” and “Patt-Lite: A Lightweight Patch and Attention Diverting Network for Challenging Facial Expression Recognition”. Inspired by these research papers, we started to develop our own model, aiming to utilize their proposed model structure and train the model according to our specific requirements.

However, during the development process, we encountered a serious mismatch problem, resulting in an accuracy of only about 40%. Our self-developed model failed to effectively address the classification problem. We attribute this problem to the mismatch in architecture and parameter settings between our implementation and the descriptions in the original papers. The lack of detailed explanations and descriptions in these papers made it challenging to accurately replicate the expected results.

Replicating the models based solely on academic papers proved to be a daunting task, and it became clear that this approach was not suitable for our work at this stage. The complexity and nuances involved in implementing these models require more comprehensive guidance and more detailed explanations, which are lacking in the existing literature.

In addition to building models based on publicly available research papers, we explored the possibility of treating the task as an image classification problem. We applied transfer learning on some well-established CNN-based image classification models (e.g., Mobilenet and ResNet) to the emotion recognition task. However, the performance of these models is still unsatisfactory, and the results are like our previous attempts using the models presented in the research paper.

Although the CNN models show the ability to extract features from facial images, the extracted features may not be suitable for the desired classification task. Based on these extracted features, it seems difficult for the feedforward layer to accurately classify emotions. Currently, we simply feed the extracted features into the linear feedforward layer for classification. Although this approach performs poorly, we believe that the main reason for this is incorrect code implementation or improper parameter tuning. We still believe that these CNN architectures can extract useful features from images. However, inserting the extracted features directly into the linear feedforward layer seems to be an unfavorable approach.

One possible solution is to incorporate other classification layers, such as an attention mechanism, to utilize the extracted features more efficiently and to classify them correctly. However, this approach requires further research, testing and experimentation. We plan to conduct more research on this approach in the future and provide more detailed findings in our next report.

Considering the challenges and limitations encountered, we conclude that further research and development of autonomously developed emotion recognition models would be a difficult and time-consuming task. With this in mind, the most practical approach at present is to utilize pre-trained emotion recognition models. This will allow us to prioritize the development of other important components of the project and avoid any potential schedule delays.

Once all other aspects of the application have been successfully completed, we will continue to explore other emotion recognition models. At this stage, we can assess whether it is feasible to train autonomously developed models to better meet our specific requirements. In addition, we will consider fine-tuning publicly available pre-trained models to meet our needs, as this approach is more likely to allow us to obtain consistent, applicable and well-performing emotion recognition models within the limited research time available.

Considering the complexity and time required to build an autonomously developed model from scratch, dedicating all resources to research and development in this area is not a feasible approach at this time. Therefore, this project will first prioritize the completion of the remaining tasks and objectives. Given sufficient time, the team will reconsider the development of a lightweight, consistent, and reliable emotion recognition model that specifically meets the requirements of our project.

3.2 Methodology of Virtual Character Control Mechanism

In this section, we will focus on discussing about how we calculate the control signals which will be used on controlling the virtual avatar and the approach to implement body movement, facial expression, emotion expression and animation triggering of the virtual avatar with the usage of Unity built-in features.

3.21 Precise Control of Avatar's Body Movement

To achieve control over the body movement of the humanoid model, we utilize the "multi-targeting constraints" feature provided by the Animated Rigging extension. This feature will rotate a specified source target to point at the target game object. Using this feature, we can control the rotation and alignment of the humanoid model's skeleton by aligning specific target game objects.

Our implementation consists of several steps. First, we assign the specified target game objects to each skeletal component of the humanoid rig. Next, we compute rotation vectors for each skeletal component based on pose landmark data obtained from the AI model detection program. These rotation vectors are indicators of desired rotation and movement.

To compute the rotation vectors, we measured the vectors between specific landmarks, such as between the left shoulder and the left elbow, to compute the rotation vectors of the left upper arm bone. By normalizing and scaling these vectors using predetermined coefficients, we obtained the necessary rotation metrics.

These rotation vectors are then applied to the corresponding target game objects. This results in a translation of the target game object relative to its original position, which further results in a rotation of the corresponding bones to point at the target game object, so we can effectively rotate and align the relevant bones of the humanoid model (Figure 13).

By utilizing "Multi-Aim Constraints" features and pose landmark detection results, we can accurately implement the necessary bone rotations and transformations on the actual humanoid model. Through this process, we can accurately transform the captured motions and body postures from the landmark detection model to the virtual humanoid model, ensuring that the virtual humanoid model effectively reflects the detected landmarks and faithfully reproduces the expected body postures and motions.

3.22 Precise Control of Avatar's Facial and Emotion Expression

To accurately control the facial and emotional expression of the avatar, we took advantage of Unity's built-in capabilities. First, we imported the humanoid model into Unity using the "UNI-VRM" extension and converted it into a Unity asset. The asset includes the humanoid 3D model as well as functions and scripts for modifying the model and applying animations or actions. We then use Unity's built-in functionality to control the facial and emotional expression of that Unity asset.

For facial and emotional expression control, we rely on the predefined Skin Mesh Renderer component of the imported humanoid model. Using this built-in feature, we can create deformable meshes and render them. The component sets several blend shape weight values that can be adjusted to modify the facial expression of the humanoid model dynamically. In addition, special blend shape weight value can be used to deform the entire facial mesh to a specific expression. By manipulating these blend shape weight value, we can customize the avatar's facial and emotional expressions (Figure 8).

To automatically control facial and emotional expressions, we utilized artificial intelligence models. For facial expressions, we focus on controlling the opening of eyes and mouth. We measure mouth and eye opening using specific values such as eye aspect ratio (EAR) and mouth aspect ratio (MAR) (Figure 15). Using the facial landmark coordinates generated by the Mediapipe holistic model, we can calculate these ratios to determine how open the user's eyes and mouth are. The calculated EAR and MAR values are then mapped to the corresponding blend shape weights to control the avatar's eye and mouth opening. In this way, we can dynamically deform the skin mesh based on the captured user expression.

Similarly, for emotional expression, we employ an emotion recognition model to detect the user's emotion from the image. The generated emotion IDs are transferred to the virtual character program and mapped to the appropriate blend shape weight values based on corresponding emotions. When a specific emotion is detected, the skin mesh is deformed accordingly to represent the corresponding emotion on the avatar's face.

By continuously calculating the EAR, MAR, and emotion ID for each frame and mapping them to the corresponding blend shape weights, we ensure that the skin mesh accurately reflects the expressions captured by the AI model. This allows for precise control of the avatar's facial and emotional expression.

3.23 Specialized Action Triggering of Virtual Avatar

To achieve natural and immersive control of the avatar, we utilize gestures captured by the AI model to implement specialized actions based on the user's intention. This approach solves the challenge of tracking and detecting complex user movements while allowing the user to trigger actions with simple, specific gestures.

First, we started by acquiring the animation assets for the humanoid models. We acquired these animations from Mixamo, an online platform provided by Adobe that offers a wide range of pre-made 3D character models, animations, and rigging solutions (Adobe Systems Incorporated) (Figure 16). We then applied these downloaded animations to the imported humanoid models in the Unity environment.

To manage the transitions and playback of these animations, we used Unity's built-in animation system, known as "Animator". By using Animator, we built a finite state machine that is responsible for controlling the animation of the avatar. Each state in the state machine corresponds to a specific animation. By defining transitions between states, we can pinpoint when each animation is triggered based on the current state of the avatar. This approach allows us to seamlessly control the animations and synchronize them with the avatar's behavior and movements (Figure 17).

Next, we develop a controller program for the finite state machine that maps the gesture IDs generated by the gesture recognition model to the corresponding animation trigger IDs. When a specific gesture is detected, the controller program receives the animation trigger ID and changes the state of the animator to the corresponding state associated with the desired animation. As a result, the corresponding animation is triggered, and the avatar can perform the specialized action associated with the detected gesture.

By mapping a specific gesture to the corresponding animation, the user can trigger a specialized action by simply executing the defined gesture. This approach provides a more natural and immersive control mechanism as the user can interact with the avatar using intuitive hand movements without having to perform complex physical actions.

3.3 Summary of Methodology

In summary, our approach consists of landmark detection using the Mediapipe holistic model, and emotion and gesture recognition using our self-developed model. Through our research and development efforts, we found that our MLP gesture recognition model performed well, while the emotion recognition model needed further refinement. To ensure the timely progress of the project, we decided to prioritize the development of the remaining tasks at hand. Time permitting, we would then revisit and develop a lightweight, consistent, and reliable emotion recognition model.

For virtual character control mechanisms, it involves computing control signals, including rotation vectors, EAR (eye aspect ratio), MAR (mouth aspect ratio), animation trigger IDs, and gesture IDs, and then applying these control signals to the chosen control mechanism using Unity's various built-in features, such as the Skin Mesh Renderer, the Animation Builder, and the Animation Rigging Package. Using these control mechanisms, we can effectively control the virtual character's body movements, facial expressions, and emotional expressions.

By combining AI models with virtual character control mechanisms, we can develop an application that can skillfully capture the user's body movements, intentions, and emotions. These captured values are then used to autonomously control the virtual character so that it can accurately reflect the user's body movements, intentions, and emotions. This approach creates an efficient, natural and immersive control system for the virtual character, enhancing the overall user experience.

4 Project Schedule, Current Progress and Future Plan

Table 1: Schedule of the project

Date	Task	Status
Early September 2023	- Search and exploration of AI models, Specifically Models for emotion recognition and gesture recognition. - Set up GitHub repository.	Finished
1, October, 2023	- Complete Detail Project Plan. - Set up Project Web Page.	Finished
Mid-October, 2023	- Evaluate performance of AI models. - Pipelining of AI models to evaluate performance.	Finished
Mid-November, 2023	- Exploration of virtual character control mechanism. - Development of virtual character control program.	Finished
Mid December, 2023	- Linkage of Ai model detection program and virtual character control program.	Finished
8-12 January, 2024	- First presentation.	Finished
21 January, 2024	- Complete detail interim report. - Complete preliminary implementation. - Addressing the difficulties encountered.	Pending
Mid-February, 2024	- Further improvement of AI models and virtual character control program.	Pending
Mid-March, 2024	- Finalize development of AI models and virtual character control program.	pending
15 – 19 April, 2024	- Final presentation.	Pending
23 April, 2024	- Complete Final report. - Complete application.	Pending
26 April, 2024	- Project exhibition.	Pending

The project schedule can be divided into five stages: AI models research & development, virtual character control mechanism research & development, Integration of programs testing and fine-tuning, application development, further research, and improvement on AI models. Currently, we are in the Integration of programs stage, focusing on testing and fine-tuning.

During the exploratory phase, we conducted extensive testing and experimentation to identify the AI models that best fit our project requirements. Based on the evaluation results, we concluded that we would select the Mediapipe Holistic model for body motion capture, an internally developed MLP classification model for gesture recognition, and a self-developed model for emotion recognition. With this conclusion in mind, we set out to develop an AI model detection program that would bring these selected models together to ensure seamless integration and functionality. We have written Python programs for executing the models that are ready to be executed in parallel with an appropriate pipeline design.

Once the AI model detection program is completed, our focus shifted to the research and development of the virtual character control mechanisms. We thoroughly researched the control mechanisms of virtual characters and decided to utilize Unity's built-in package "Animation Rigging" to control limb movements, "Skinned Mesh Renderer" function to manage facial features and emotional expressions, and "animator", Unity's animation system, to control animation playback and blending. We then created virtual character control programs based on these mechanisms to ensure efficient and reliable control of virtual characters.

After successfully developing the AI models and the virtual character control mechanisms, we integrated these two programs together to create a pipeline application. We established a TCP communication connection between the programs and implemented basic functionality to enable the AI model detection program to communicate with the avatar control program and pass the AI detection results to the avatar control program. In this way, the application program is able to utilize the AI model to capture the user's body movements, intentions, and emotions through the camera and effectively reflect them on the virtual character in an efficient and natural way.

We are currently actively testing and experimenting with the pipeline application to identify any issues, potential improvements, or necessary modifications. Throughout this process, we have encountered several difficulties and issues, which we will address in more detail later in this report.

Our next steps include implementing the possible solutions we studied. We will test, pilot, and implement the solution to evaluate its effectiveness in solving the identified problems. If necessary, we will explore alternative solutions and implement them. Once we have conducted thorough testing and made the required changes and improvements, we will proceed to develop the actual application. This includes merging the two programs, refining the code base, and enhancing the overall logic of the application to achieve the final structure.

Once the application is complete, and if there is sufficient time, we will shift our focus to further research and development of the AI model. These additional efforts are intended to improve the efficiency and consistency of the AI model detection program, aiming for optimal performance and accuracy. If any new AI models are developed during this process, we will replace the existing models and apply them to the application. Therefore, for the remainder of the semester, our primary focus will be on problem solving, application development, and subsequent research and development of the AI models. We expect the final product to be available around late-March.

5 Difficulties, Problems Encountered and Proposed Solutions

5.1 Fluctuation Of Landmark Detection Results

During the testing and trial phase of the application, we encountered an issue related to fluctuations in the landmark detection results generated by the Mediapipe holistic model. As described in the methodology section, these landmark detection results are used to control virtual characters, and fluctuations can negatively affect the overall performance of virtual character control.

This problem is particularly evident in tasks involving the control of avatar movements and expressions, especially in cases where rotation vectors are calculated directly from landmark detection results. Variations in the landmark detection results can lead to fluctuations in the rotation vector, which can result in unnatural and abnormal body movements of the avatar (Figure 19). Similarly, when controlling facial expressions such as eye opening and mouth opening, fluctuations in landmark detection results directly affect metrics such as MAR and EAR (Figure 18). This can cause the avatar to constantly blink, open its eyes and mouth, further exacerbating the problem.

Through extensive testing, trials, and experiments, we determined that this problem is primarily caused by the Mediapipe holistic model. While the model performed generally well in correctly recognizing landmark coordinates, the landmark detection results changed slightly each time, even when the actual user remained stationary. These slight variations in subsequent landmark coordinates may not be clearly seen when landmark data is displayed directly on the actual image. However, since the control signals are calculated based on the landmark coordinates, these minor fluctuations are projected onto the control signals, leading to the jitter problem described above when the avatar is moving. This inconsistency conflicts with the natural motion we wish to achieve.

To address this problem, we investigated potential solutions, and one approach we considered was to apply specific filters to the generated landmark detection data. After careful consideration, we concluded that sliding window filters and low-pass filters are viable solutions.

Currently, we are experiencing fluctuations in the landmark detection data due to slight changes in the values of subsequent detections. Therefore, it may not be appropriate to directly apply the original landmark coordinates to calculate the control signal. To solve this problem, we can use a sliding window filter. This technique involves moving a window of specified length over the data, sample by sample, and performing the desired statistical calculation or operation on the data within the window. By averaging successive samples of landmark data, small variations in the data can be smooth out. This can effectively eliminate or minimizes fluctuations and ensures more stable and reliable control signal calculations.

In our real-time application, the AI model is updated very quickly and therefore generates significant fluctuations. These fluctuations are attributed to slight variations in the results of each detection, thus introducing high-frequency noise to the system. To solve this problem, a low-pass filter can be used to mitigate the high-frequency noise.

Low-pass filter allows landmark data with frequencies below a specified cutoff frequency to pass through while attenuating or minimizing the effects of data with frequencies above the cutoff frequency. By using this filter, we can effectively smooth the generated landmark data, thereby reducing the magnitude of the fluctuation error. This approach helps to improve the stability and consistency of the system by mitigating the effects of high-frequency noise.

Based on our study, the application of these two filters may be possible solutions to the fluctuation problem. However, it should be noted that we have not yet tested these filters in an actual program. Therefore, we cannot assert that these filters will solve the problem. Our plan is to apply these filters to the landmark detection model, conduct further tests and experiments, and evaluate the results to determine their effectiveness in mitigating the fluctuation problem. More details will be explained in a later report.

5.2 Poor Performance of Self-Developed Emotion Recognition Models

One of the difficulties we have encountered so far is the poor performance of our self-developed emotion recognition models, as described in the methodology section. As a result of our research, we have identified potential methods for further development of emotion recognition models.

One approach we are considering is to employ transfer learning on pre-trained CNN-based image classification models. In this approach, we will make use of existing CNN models such as ResNet or MobileNet, which are good at extracting valuable

features from images. However, using only linear layers for classification may not be sufficient for our particular task. We observed severe underfitting, suggesting that models using only linear layers as feedforward components lack the necessary ability to accurately classify emotions.

To address this issue, we are exploring the incorporation of more complex structures such as attention mechanisms into the model. By integrating these complex structures, we aim to improve the model's ability to effectively classify emotions based on extracted features.

The second approach is to use both image and landmark data as input to the emotion recognition model. We will use CNN models to extract features from images and combine them with landmark data. The combined feature set will be used as input to another classification model. This approach provides the AI model with a larger and more diverse set of features to analyze and classify emotions, thus providing the model with the possibility to classify emotions with better performance. However, this approach also increases the complexity of the model and may present challenges in accurately classifying emotions based on features. Given the limited development time, this approach may pose additional difficulties. Nonetheless, we remain open to exploring this possible approach if there is sufficient time for testing and experimentation.

We are also exploring an alternative to fine-tuning pre-trained models for our application. Developing AI models from scratch can be a daunting and time-consuming task. Fine-tuning existing models using datasets that match the context of our application increases the likelihood of obtaining a well-performing and consistent model based on our project requirements.

The main challenge was to Identify a suitable lightweight and efficient model that met our needs and then fine-tune it accordingly. While finding a suitable model and fine-tuning it requires an initial investment of time, using a pre-trained base model, we can build a classification model on top of it. And this approach is more likely to produce well-performing sentiment classification models and takes relatively less time than building and training a model from scratch.

In summary, we identified three possible approaches for further development of emotion recognition models: transfer learning of complex structures, combining image and landmark data, and fine-tuning pre-trained models. Each approach has its own

considerations and tradeoffs, and we will continue to test, experiment, and evaluate to determine the most effective solution within the constraints of the project.

6 Conclusions

In conclusion, current methods of controlling avatars using AI models suffer from imprecise body movements and limited emotional expression, resulting in a lack of natural interaction between users and avatars. To overcome these limitations, our project aims to utilize multiple AI models to achieve advanced control of avatars, providing users with a unique and more natural experience.

The main outcome of our project will be an application that utilizes a combination of AI models as the basis for controlling avatars. The application consists of two programs: one written in Python to handle AI-related tasks and the other written in C# on the Unity platform to control the avatar. The workflow consists of transferring live camera data streams to the application. The AI model detector analyzes the camera data stream, detects the images using the AI model, and transmits the results to the virtual character controller over TCP. The virtual character control program then updates its control signals based on the received results, influencing, and triggering various features and animation of the virtual character.

The application will provide several key features. First, it can accurately and efficiently control the avatar's body movements and facial expressions using the Mediapipe holistic model. This ensures that the avatar's movements are consistent with the user's intentions and body movements. In addition, the application uses self-designed, trained and fine-tuned emotion recognition models and gesture recognition models. These models enable the avatar to accurately express emotions and initiate specialized actions based on recognized gestures.

To achieve precise control over the humanoid models, the application utilizes animation rigging extensions as well as Unity's built-in features such as the animator and skin mesh renderer. By integrating these components with the AI model, the application enables precision control of the avatar's movements, resulting in a seamless and realistic user experience.

The project is divided into several phases, including AI models research & development, virtual character control mechanism research & development, integration of programs testing and fine-tuning, application development, further research and

improvement on AI models. Currently, we are testing, experimenting, and trialing the pipeline application and have encountered a number of problems and difficulties. For the remainder of the semester, we will focus on resolving these problems encountered. Subsequently, we will integrate all the components together and start developing the application. Finally, we plan to conduct further research on the AI model to improve its efficiency and performance so that the final application will be able to capture the user's physical movements, intentions, and emotions in a more efficient, consistent, and accurate manner.

References

1. Adobe Systems Incorporated. (n.d.). Mixamo. https://www.mixamo.com/#/
2. Beklemysheva, A. (2022, December 27). *Why use python for AI and machine learning?*. SteeKiwi. https://steelkiwi.com/blog/python-for-ai-and-machine-learning/
3. Gank Content Team. (2023, July 8). *What is a vtuber? The Ultimate Guide to Virtual YouTubers!*. Gank Blog. https://ganknow.com/blog/vtuber/
4. Grishchenko , I., & Bazarevsky, V. (2020, December 10). *MediaPipe holistic – simultaneous face, hand and pose prediction, on device*. MediaPipe Holistic — Simultaneous Face, Hand and Pose Prediction, on Device. https://blog.research.google/2020/12/mediapipe-holistic-simultaneous-face.html
5. Kapitanov, A. (2022). *Hukenovs/hagrid: Hand gesture recognition image dataset*. GitHub. https://github.com/hukenovs/hagrid
6. NeuroSYS. (2023, October 4). *Unity Engine – is it bad & what is it good for?* https://neurosys.com/blog/why-people-say-unity-engine-is-bad
7. PapersWithCode. (n.d.). *Papers with code – fer2013 dataset*. FER2013 Dataset | Papers With Code. https://paperswithcode.com/dataset/fer2013
8. Pixiv Inc. (n.d.). Vroid Studio. https://vroid.com/en/studio
9. Singh, J. (2023, May 21). *What is a vtuber, and how do you become one?*. Cointelegraph. https://cointelegraph.com/news/what-is-a-v-tuber
10. VRM Consortium Inc. (n.d.). GitHub – VRM-C/UNIVRM. https://github.com/vrm-c/UniVRM

Appendix

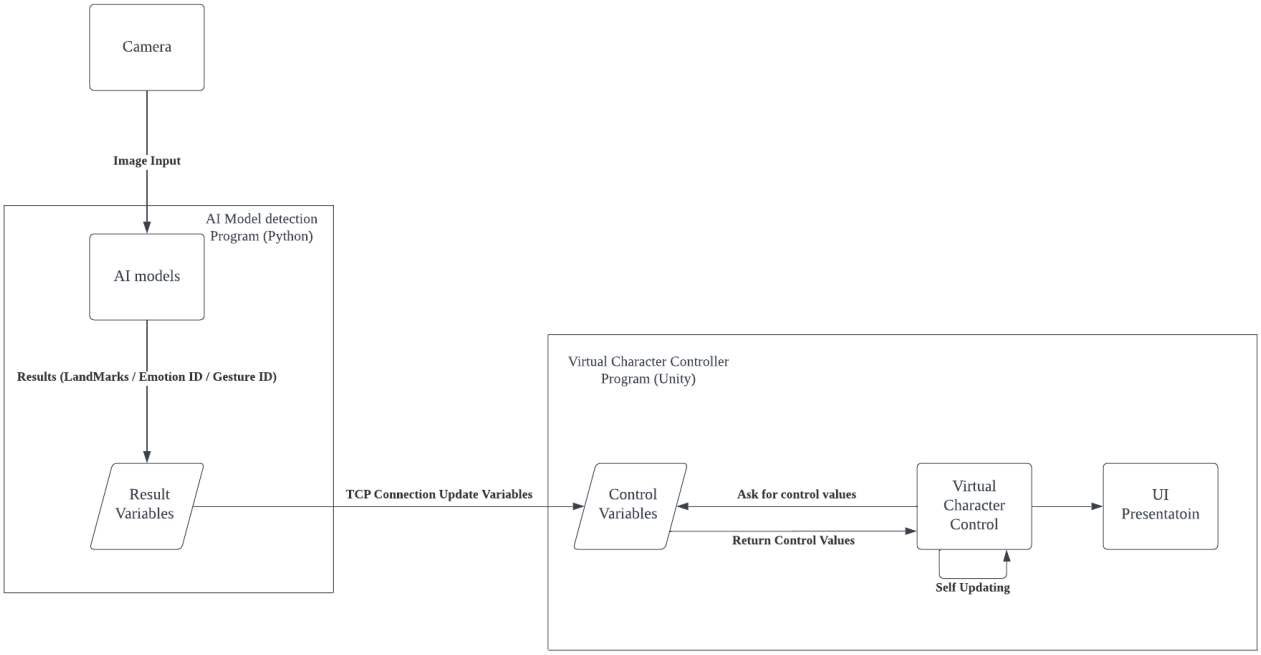

Figure 1: General structure of application.

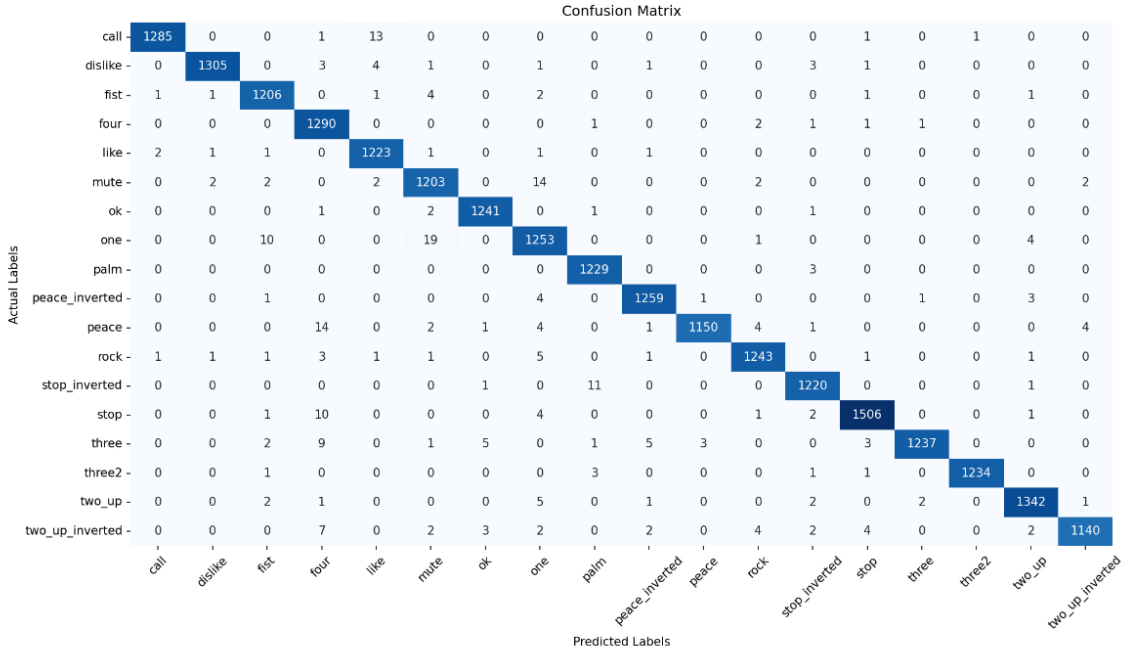

Figure 2: Confusion matrix of lefthand gesture recognition model.

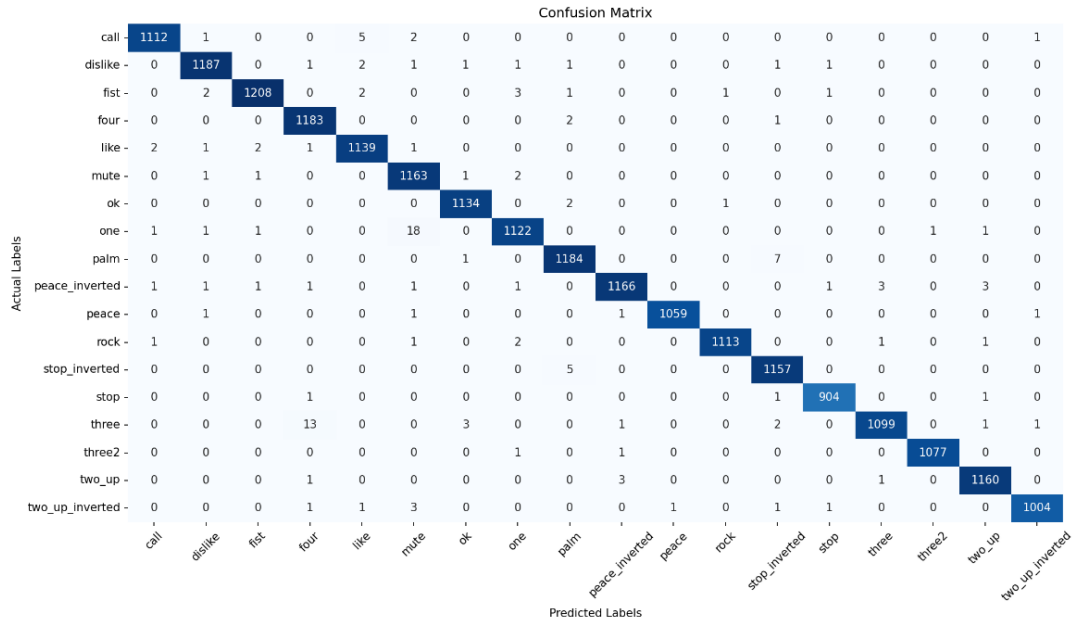

Figure 3: Confusion matrix of righthand gesture recognition model.

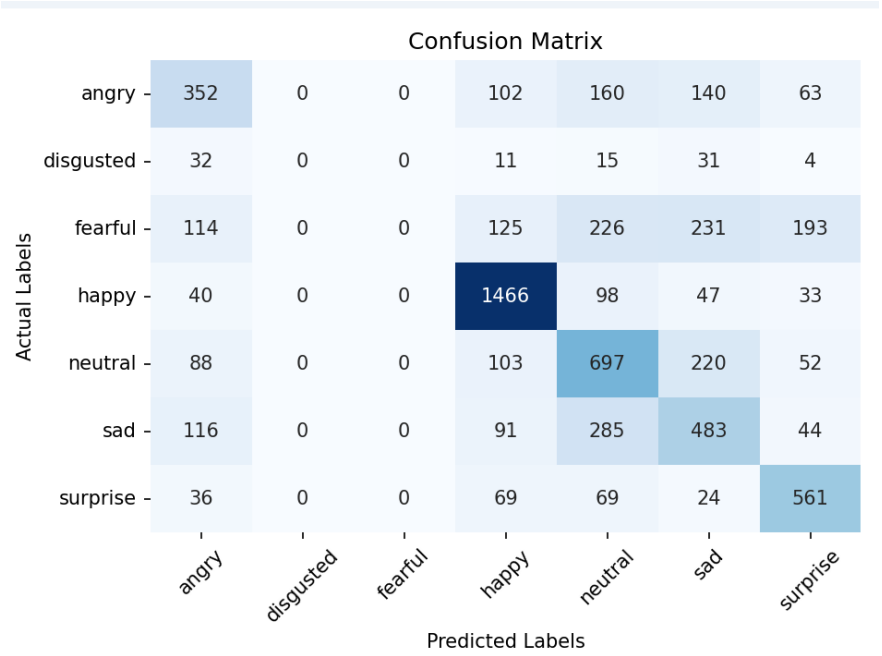

Figure 4: Confusion matrix of MLP classification model of emotion recognition.

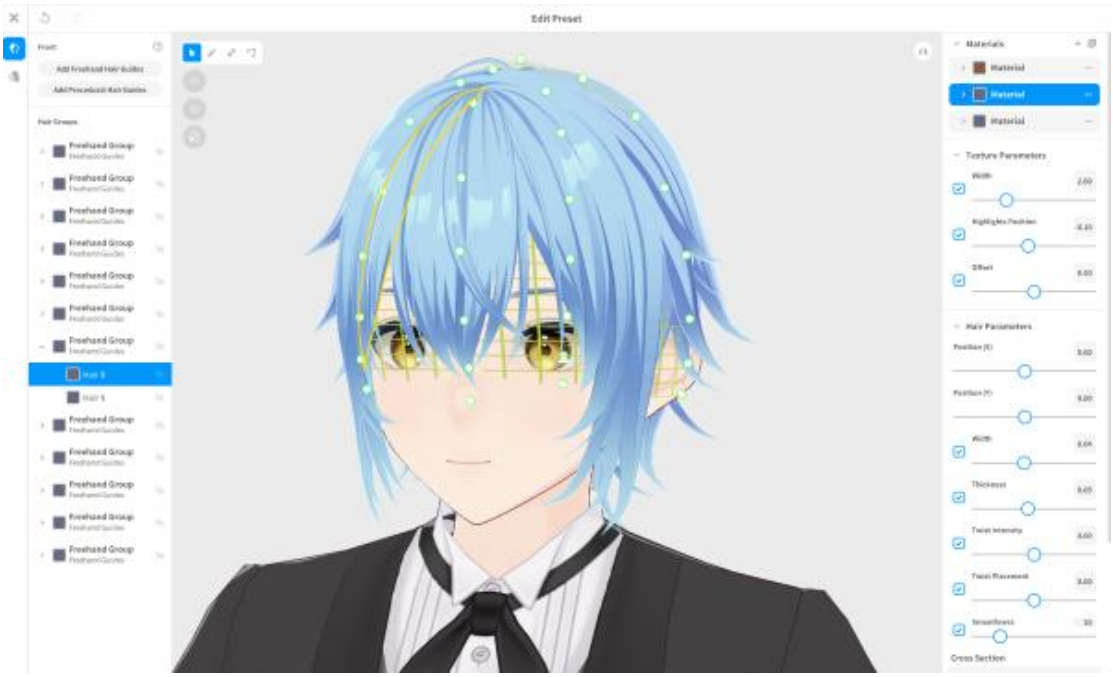

Figure 5: Vroid Studio.

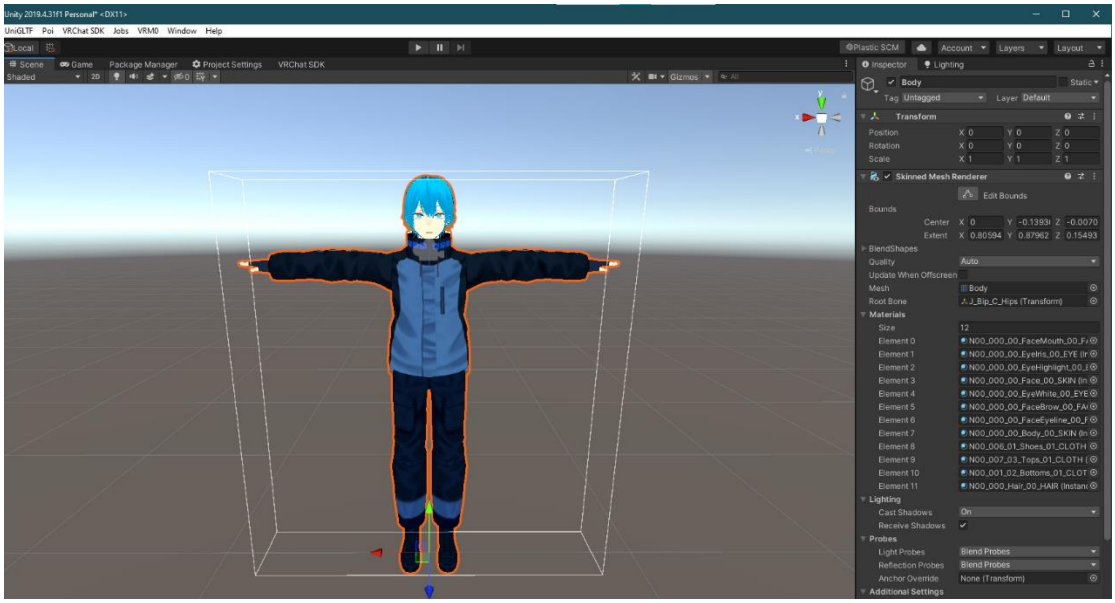

Figure 6: Unity assets imported with UNI-VRM.

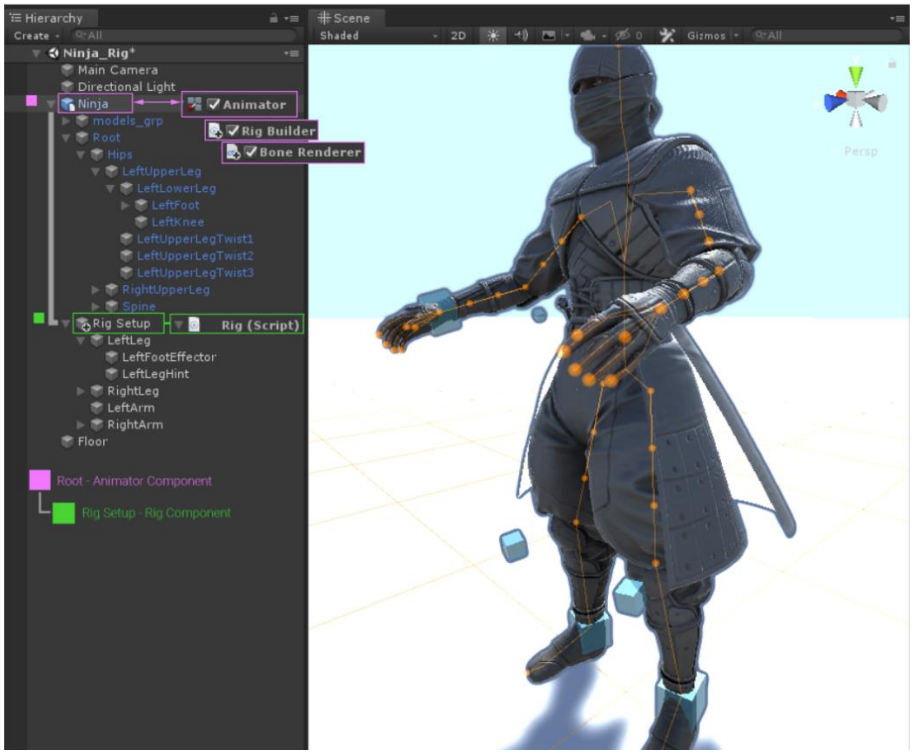

Figure 7: Animation Rigging.

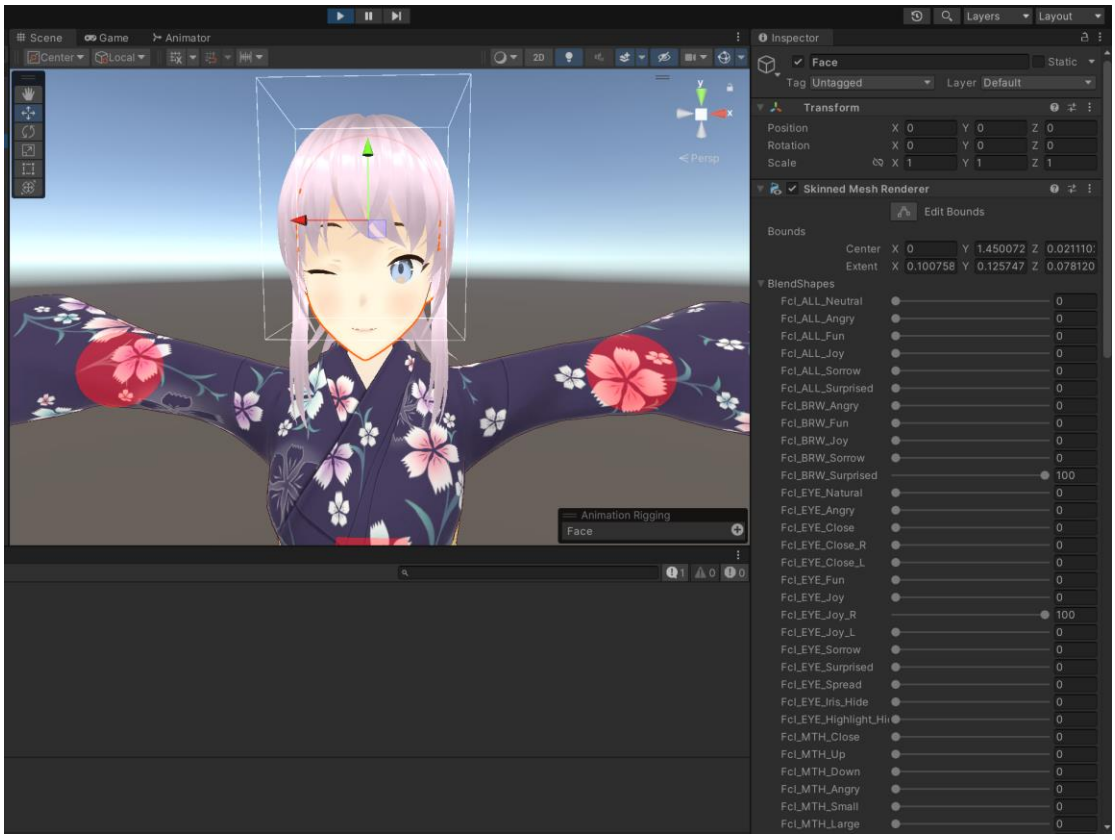

Figure 8: Skinned Mesh Renderer.

Figure 9: Mediapipe Holistic Model detection results

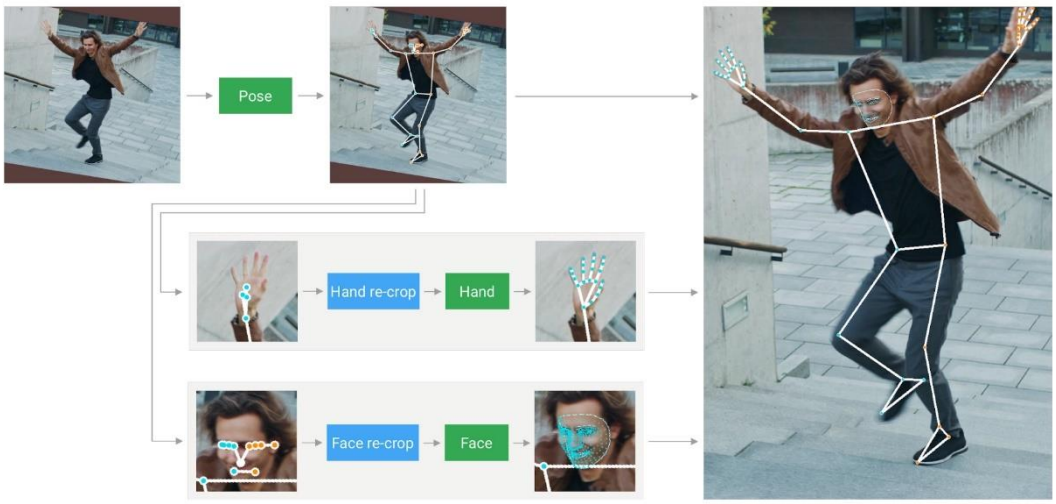

Figure 10: Mediapipe Holistic Pipeline overview.

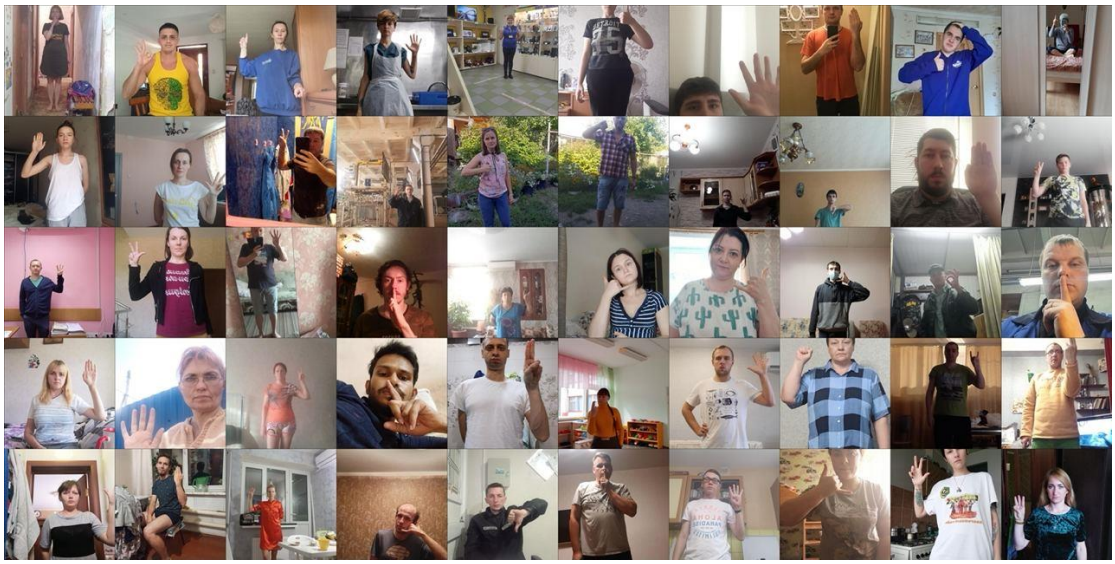

Figure 11: Example of HaGRID.

Figure 12: Example of FER-2013.

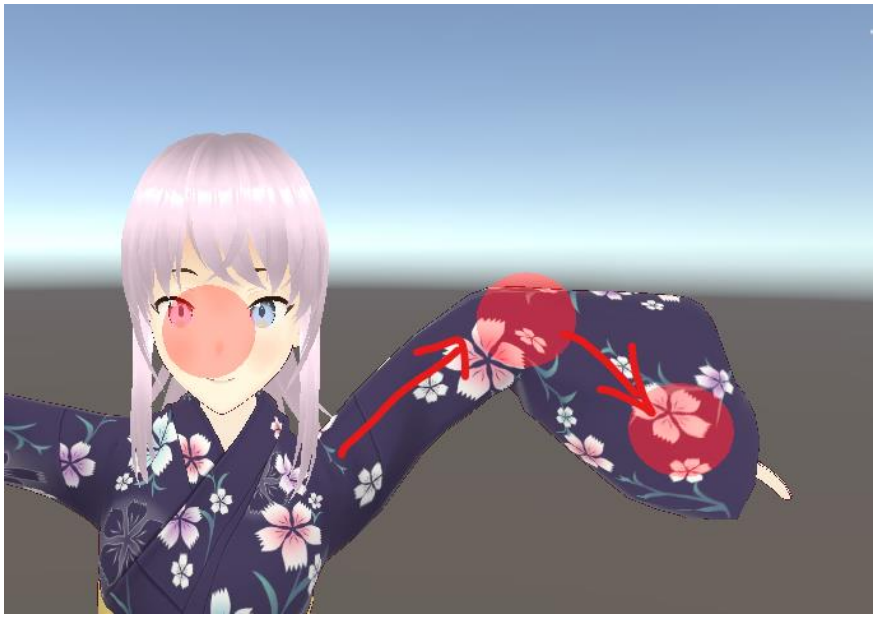

Figure 13: Example of Multi-Aim Constraint applied on avatar’s arm.

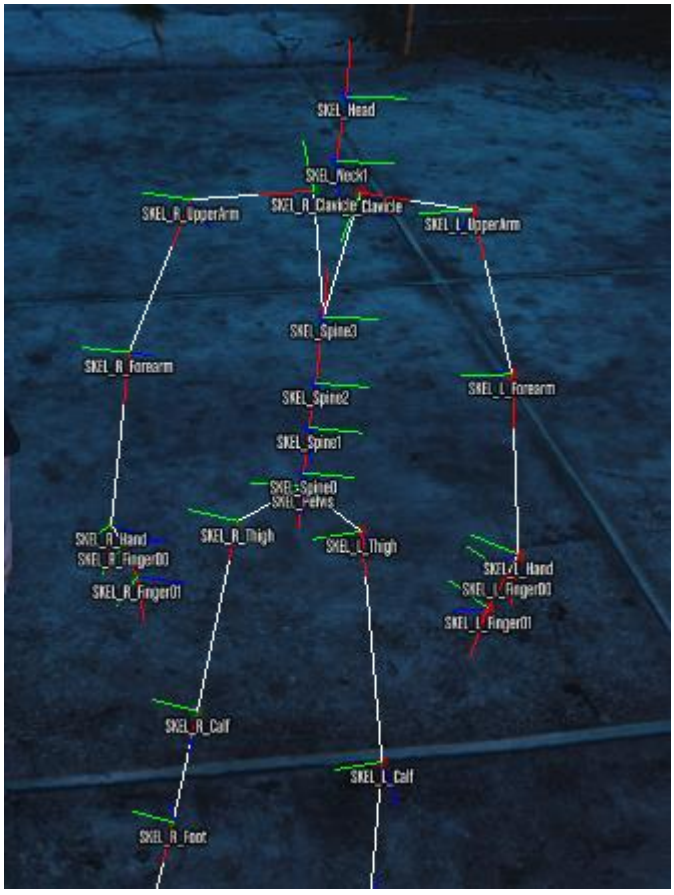

Figure 14: Rotation Vector.

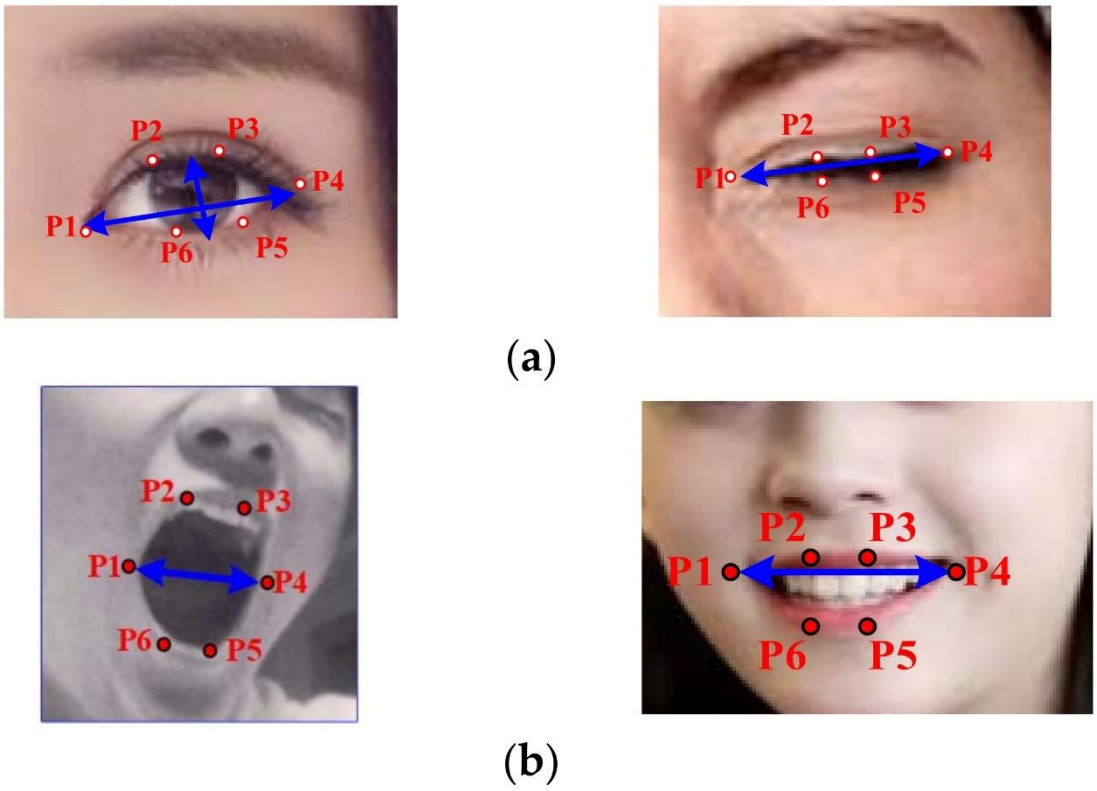

Figure 15: EAR (a) and MAR (b).

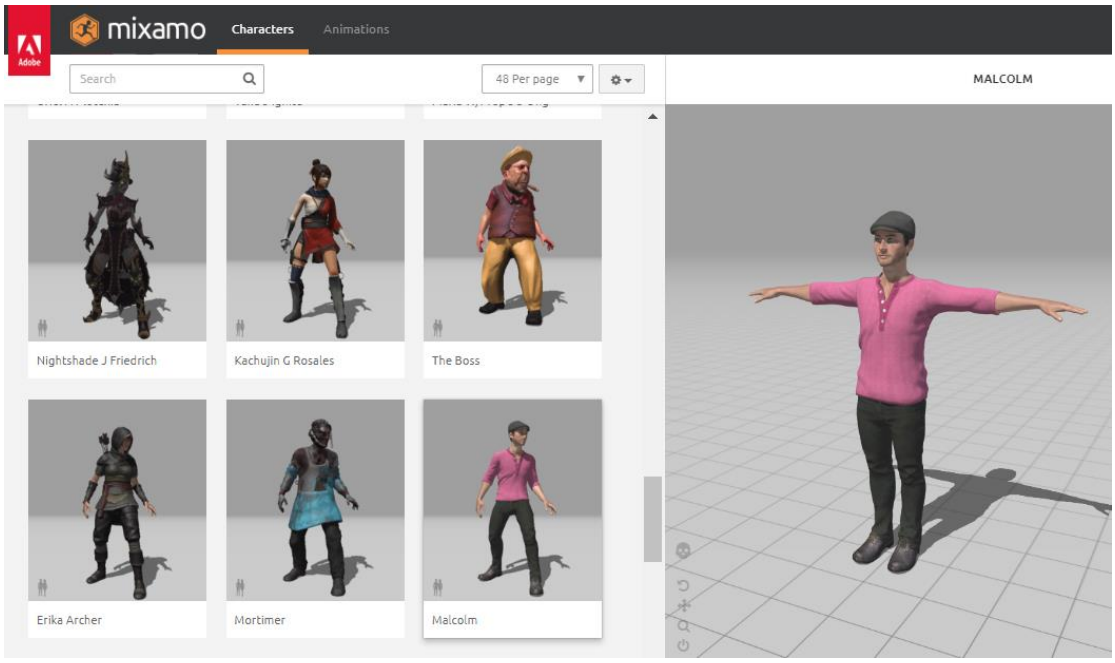

Figure 16: Mixamo.

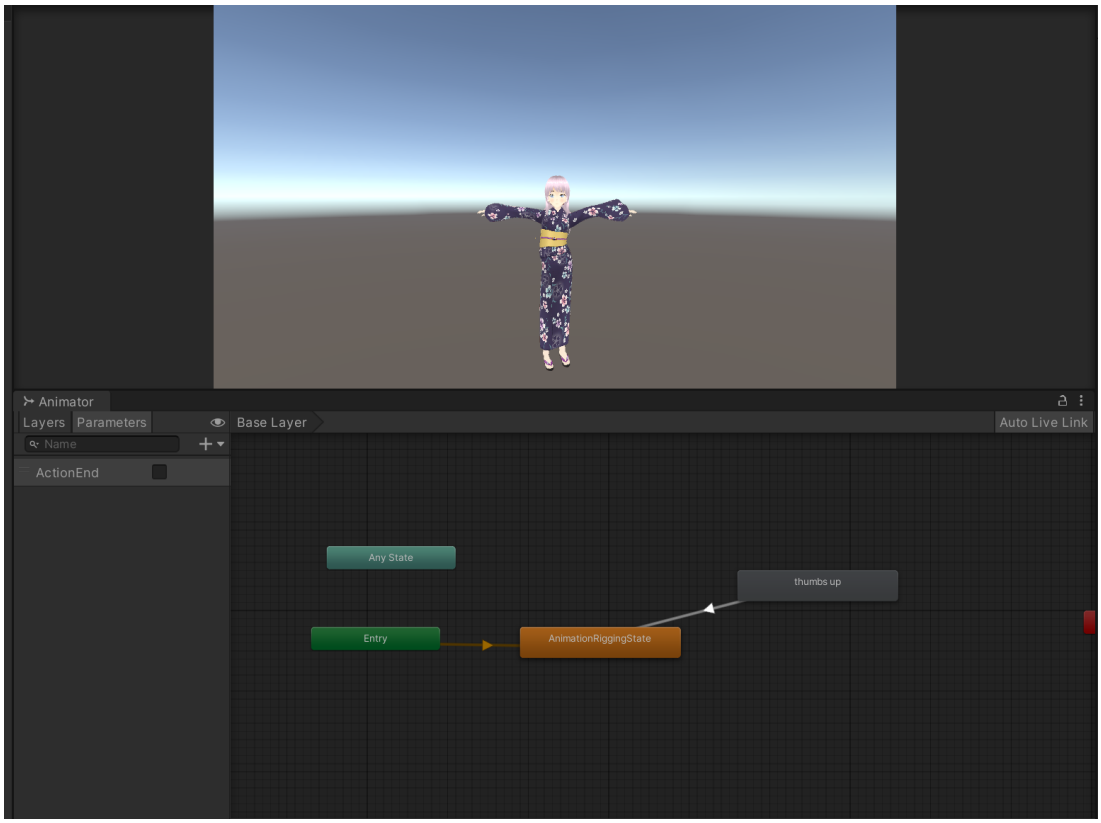

Figure 17: Example of Animator.

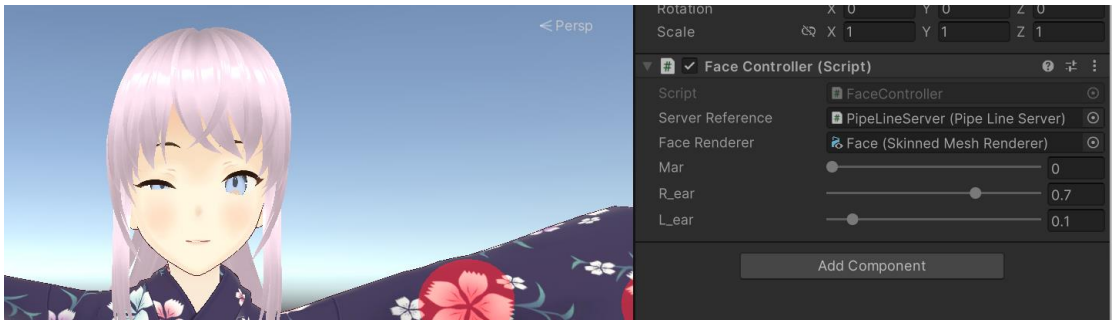

Figure 18: Example of fluctuation of MAR and EAR (blinking problem).

Figure 19: Example of fluctuation of Rotation Vector (shaking problem).