

**人工智能导论课程项目论文**

Badnovelists

## Text-Driven Motion Generation With Mtionclip

姓名：

|  |  |
| --- | --- |
| 任政 | 22920212204430 |
| 孙中川 | 22920212204441 |
| 杨磊 | 22920212204476 |
| 熊军睿 | 22920212204465 |
| 谢一帆 | 22920212204464 |

学院： 信息学院

专业： 数字媒体技术年级： 大 三

二〇 二四 年 六 月 二十一 日校内指导教师：姚俊峰 教授

### Abstract

Human motion modeling is crucial for many modern graphics applications, typically requiring professional skills. To remove the skill barriers for laypeople, recent motion generation methods can directly generate human motions based on natural language, images, and even audio inputs. In this course project, we chose natural language input as our research subject. However, it remains challenging to achieve diverse and fine-grained motion generation based on various text inputs. We selected two models from previous related studies: MotionDiffuse and MotionClip. We intend to thoroughly understand and compare these models and ultimately chose MotionClip as our method. Based on this, we implemented the visualization of generated motions using Unity. MotionCLIP, a 3D human motion auto-encoder featuring a disentangled, well-behaved latent embedding that supports highly semantic textual descriptions. The key innovation lies in aligning the human motion manifold with the latent space of the Contrastive Language-Image Pre-training (CLIP) model. By doing so, MotionCLIP leverages the rich semantic knowledge encapsulated in the CLIP model and infuses it into the motion manifold. Specifically, this alignment helps ensure that semantically similar motions are placed close to one another and inherits the disentanglement characteristics from the CLIP space structure. In summary, we compared two important architectures in the text-to-motion field and implemented MotionCLIP ourselves. Based on this, we achieved visualization and provided a platform for motion generation.

### Key Words:

Transformer text2motion SMPL MotionClip MotionDiffuse

**Contents**

1. [Introduction 4](#_bookmark0)
   1. [Background 4](#_bookmark1)
   2. [Research Review 5](#_bookmark2)
      1. [Comparison Between MotionCLIP And MotionClip 5](#_bookmark3)
      2. [Motion Data Visualization Analysis 7](#_bookmark4)
2. [Related Work 9](#_bookmark5)
3. [Methodology 12](#_bookmark6)
   1. [Method of MotionClip 12](#_bookmark7)
      1. [Main Framework 12](#_bookmark8)
      2. [Datasets 15](#_bookmark9)
      3. [Model 17](#_bookmark10)
   2. [Method of Motion Visualization 20](#_bookmark11)
4. [Experiment On MotionClip 46](#_bookmark12)
   1. [Data process 46](#_bookmark13)
   2. [Training 50](#_bookmark14)
   3. [Evaluation 52](#_bookmark15)
   4. [Limitation 56](#_bookmark16)
   5. [Demonstration 57](#_bookmark17)
5. [Evaluation 60](#_bookmark18)
6. [Application 62](#_bookmark19)
7. [Strengths/Limitation and Discussion 64](#_bookmark20)
8. [Conclusion 66](#_bookmark21)

[Reference 67](#_bookmark22)

# Introduction

## Background

Human motion modeling is a critical component in animating virtual characters to imitate vivid and rich human movements, which has been an essential topic for many applications, such as film-making, game development, virtual YouTuber animation, and even the concept of the metaverse. To mimic human motions, virtual characters should be capable of moving naturally, reacting to environmental stimuli, and expressing complex emotions. Despite decades of exciting technological breakthroughs, creating lively and authentic body movements still requires sophisticated equipment and domain experts, or the use of manual keyframing, which demands a significant amount of time. To remove the skill prerequisites for ordinary users and potentially scale to a mass audience, it is crucial to create a versatile human motion generation model that can produce diverse, easily manipulable motion sequences while understanding the hidden semantics in natural language.

Previous methods for human motion generation have utilized various condition signals, including predefined motion categories (Guo et al., 2020; Petrovich et al., 2021; Cervantes et al., 2022), music pieces (Huang et al., 2020; Li et al., 2020, 2021; Zhuang et al., 2020; Siyao et al., 2022), and natural language (Lin et al., 2018; Ahuja and Morency, 2019; Ghosh et al., 2021; Petrovich et al., 2022). Among these condition signals, natural language can be considered the most user-friendly and convenient input format for synthesizing motion sequences. Based on previous research, common models have shown accurate and prominent semantic understanding of natural language, hence we focus on text-driven motion generation in this paper.

Recently, TEMOS (Petrovich et al., 2022) demonstrated fine-grained trajectory synthesis using the KIT Motion-Language MoCap dataset (Plappert et al., 2016). However, it lacks support for stylizing the generated motions, thereby limiting its

diversity. MotionCLIP (Tevet et al., 2022) can generate stylized motions but is still constrained by short text inputs and struggles with handling complex motion descriptions. Additionally, both TEMOS and MotionCLIP typically accept only a single text prompt, significantly restricting user creativity.

In recent years, there have been more motion capture datasets emerging, sometimes categorized by type (Liu et al., 2019; Ji et al., 2018), and even annotated with free text descriptions (Punnakkal et al., 2021; Plappert et al., 2016; KIT Motion-Language MoCap Dataset; HDM05, etc.). We chose HDM05 and the KIT Motion-Language MoCap Dataset as our training datasets.

## Research Review

## Comparison Between MotionCLIP And MotionClip



Fig. Motions generated by MotionCLIP conditioned on different cultural references.

MotionCLIP exploits the rich knowledge encapsulated in pre-trained language-images model (CLIP) and projects the human motion manifold over its latent space.

MotionCLIP leverages the latent space of the CLIP model, allowing it to generate motions that are highly aligned with textual descriptions. This makes it particularly effective for tasks involving natural language descriptions. The model can handle highly abstract text descriptions and generate corresponding motions (e.g., describing "Spiderman" results in web-swinging motions).The latent embeddings of MotionCLIP are disentangled and well-behaved, making it excel in tasks such as motion editing, interpolation, and transfer. Due to alignment with CLIP's latent space, semantically

similar motions are close to each other in the latent space, facilitating continuous generation and editing.

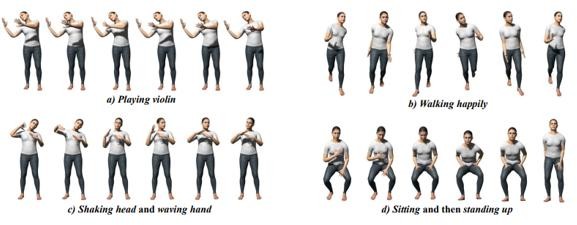
The model can be used for various tasks, including text-to-motion generation, motion style transfer, interpolation, editing, and recognition, demonstrating its high adaptability and flexibility. Aligning with CLIP's latent space makes the training process relatively complex and computationally intensive. It requires extensive pre-training and fine-tuning to achieve optimal performance. Dependence on Visual Inputs: To improve alignment and generation quality, MotionCLIP relies on synthetic visual inputs, increasing the complexity and computational cost of model training.

Fig. MotionDiffuse is a diffusion model-based text-driven motion generation method that features 1) Probabilistic Mapping 2) Realistic Synthesis that results in highly diverse motions with high-fidelity shown in a)-d), and 3) Multi-Level Manipulation that empowers comprehensive motion generation such as that in c) where multiple body parts are involved and d) where

time-varied prompts are given.

Motiondiffuse Based on Diffusion Model: MotionDiffuse utilizes the strong generative capabilities of diffusion models, enabling the generation of high-quality and diverse motion sequences. Diffusion models perform well in handling uncertainty and generating diverse results.

Training Stability: Compared to other generative models, diffusion models typically have a more stable training process, reducing the risk of mode collapse.

Data Efficiency: Diffusion models often perform well with relatively less data, exhibiting good data efficiency.

Weaker Semantic Understanding: Compared to MotionCLIP, MotionDiffuse has

weaker capabilities in handling complex natural language descriptions and abstract semantics.

More task-specific adjustments are needed to achieve high alignment with text descriptions. High Complexity: The structure of diffusion models is relatively complex, resulting in significant training time and resource consumption. High demands on hardware and computational resources may make it unsuitable for resource-constrained environments.

In conclusion, MotionCLIP excels in handling natural language descriptions and generating motions that are highly semantically matched, making it suitable for applications requiring high semantic understanding and editing tasks. However, it has high training complexity and computational resource demands. On the other hand, MotionDiffuse leverages the generative capabilities of diffusion models, offering advantages in generating diverse and stable outputs, making it suitable for scenarios with limited data and a need for high-quality generation. Nevertheless, its semantic understanding capability is relatively weak, requiring more specific adjustments to match complex text descriptions.

The choice of model depends on the specific needs and constraints of the application. For instance, if the goal is to generate complex motions that match highly semantic text, MotionCLIP may be more appropriate. Conversely, if there is a need for stable and diverse motion generation in a resource-limited environment, MotionDiffuse might be the better choice.

## Motion Data Visualization Analysis

Motion data visualization is an essential aspect of understanding, analyzing, and improving human motion generation models. By visualizing motion data, researchers and developers can gain insights into the performance, accuracy, and versatility of models such as MotionCLIP and MotionDiffuse. This analysis includes the examination of motion sequences, comparison of generated motions to ground truth

data, and identification of patterns and anomalies. Visualization Techniques

SMPL (Skinned Multi-Person Linear Model)

SMPL is a 3D human model that parametrizes the shape and pose of a human body, allowing for realistic rendering of human motion. It captures the nuanced details of body deformations, providing a high-fidelity representation of human movements.

By applying motion data to the SMPL model, researchers can create lifelike animations that accurately reflect the generated motion sequences. This helps in assessing the anatomical plausibility and smoothness of the generated motions.

Unity

Unity is a powerful real-time development platform widely used for creating 3D animations, games, and simulations. It provides robust tools for rendering and visualizing motion data in an interactive environment.

By importing SMPL models and motion data into Unity, researchers can render detailed animations, view them from multiple angles, and interact with the motion sequences in real time. This enhances the ability to evaluate the quality and applicability of generated motions in practical scenarios.

By utilizing SMPL and Unity for rendering and visualizing motion data, researchers can effectively analyze the performance of human motion generation models. These visualization techniques not only provide qualitative insights but also support quantitative assessments, enabling a comprehensive evaluation of models like MotionCLIP and MotionDiffuse. Through detailed visualization and analysis, it becomes possible to refine and enhance motion generation models, ultimately leading to more realistic and versatile human motion synthesis.

# Related Work

Motion generation has been studied for decades. Some early works focus on unconditional motion generation (Rose et al., 1998; Ikemoto et al., 2009; Mukai and Kuriyama, 2005). Other works attempt to predict future motions given an initial pose or a starter motion sequence (Futrelle and Speckert, 1978; Gavrila, 1999; O'rourke and Badler, 1980), employing statistical models such as PCA (Ormoneit et al., 2005) and Motion Graph (Min and Chai, 2012).

With the rapid development of Deep Learning (DL) techniques, various generative architectures have emerged and flourished. Previous works can be broadly categorized into four groups: 1) Variational Autoencoder (VAE); 2) Generative Adversarial Networks (GAN); 3) Normalization Flow Network; 4) Implicit Neural Representations.

VAE (Kingma and Welling, 2013) is one of the most commonly used generative models in motion synthesis. Yan et al. (2018) and Aliakbarian et al. (2020) consider the motion generation task as predicting a small future motion sequence given a small current motion sequence. They encode the current and future sequences with VAE and then reconstruct the future sequence. ACTOR (Petrovich et al., 2021) proposes a transformer-based encoder and decoder architecture. Transformer Encoder Layers and Transformer Decoder Layers (Vaswani et al., 2017) form the basic modules to build motion encoders and decoders. This architecture has also been adopted in subsequent works (Tevet et al., 2022; Hong et al., 2022; Petrovich et al., 2022).

GAN (Goodfellow et al., 2014) introduces an auxiliary module, the discriminator network, to assess the quality and validity of generated samples. Some works focus on proposing appropriate discriminator networks for motion generation to enhance synthesis quality (Barsoum et al., 2018; Harvey et al., 2020; Wang et al., 2020). HP-GAN (Barsoum et al., 2018) attempts to supervise motion prediction results without specific ground truth labels. Thus, a data-driven discriminator is employed to

learn motion priors for evaluating prediction quality. Harvey et al. (2020) address the blurriness issue in predicting motion for motion in-between tasks by proposing short-term and long-term critics. Wang et al. (2020) establish a cyclic pipeline where the proposed pipeline can generate both class-specific and mixed-class motion sequences with the aid of a discriminator.

Normalization Flow Network (Dinh et al., 2014) has a long history and has been extensively studied for image synthesis (Dinh et al., 2016; Kingma and Dhariwal, 2018). This architecture establishes a reversible neural network that maps input data into a multi-dimensional Gaussian distribution. Therefore, we can generate an initial random vector from this distribution and input it into the reversed network to generate motion samples. Inspired by the success of GLOW (Kingma and Dhariwal, 2018), MoGlow (Henter et al., 2020) proposes an autoregressive normalization network to model motion sequences. Historical features from an LSTM model (Hochreiter and Schmidhuber, 1997) serve as conditions for the flow network to predict the next pose. Recently, another generative model has gained significant attention due to the remarkable success of NeRF (Mildenhall et al., 2020; Jain et al., 2021) in rendering realistic images. Implicit Neural Representations (INR) are a series of neural networks that optimize their parameters to fit individual samples rather than entire distributions. One major advantage of this technique is its excellent generalization ability in spatial or temporal dimensions. For example, Cervantes et al. (2022) propose an implicit scheme that simultaneously models action categories and timestamps. Similar to the original NeRF, timestamps are represented by sinusoidal values. After supervised training, the proposed method can generate variable-length motion sequences for each action category.

Neural networks have successfully learned powerful latent representations coupling natural images with natural language describing it [He and Peng 2017; Ramesh et al. 2021]. A recent example is CLIP[Radford et al. 2021], a model coupling images and text in deep latent space using a constructive objective[Hadsell et al. 2006; Chen et al. 2020]. By training over hundred millions of images and their captions, CLIP gained a

reach semantic latent representation for visual content. This expressive representation enables high quality image generation and editing, controlled by natural language [Patashnik et al. 2021; Gal et al. 2021; Frans et al. 2021]. Even more so, this model has shown that connecting the visual and textual worlds also benefits purely visual tasks [Vinker et al. 2022], simply by providing a well-behaved, semantically structured, latent space. Closer to our method are works that utilize the richness of CLIP outside the imagery domain. In the 3D domain, CLIP’s latent space provides a useful objective that enables semantic manipulation [Sanghi et al. 2021; Michel et al. 2021; Wang et al. 2021a] where the domain gap is closed by a neural rendering. CLIP is even adopted in temporal domains [Guzhov et al. 2021; Luo et al. 2021; Fang et al. 2021] that utilize large datasets of video sequences that are paired with text and audio. Unlike these works that focus on classification and retrieval, we introduce a generative approach that utilizes limited amount of human motion sequences that are paired with text.

We use a new motion generation pipeline based on Transformer. Transformer is a deep learning model based on attention mechanisms, initially proposed by Vaswani et al. in 2017 for tasks in natural language processing such as machine translation and language modeling. It innovatively introduces self-attention mechanisms, enabling parallel processing of sequence data and significantly enhancing computational efficiency.

# Methodology

## Method of MotionClip

## Main Framework

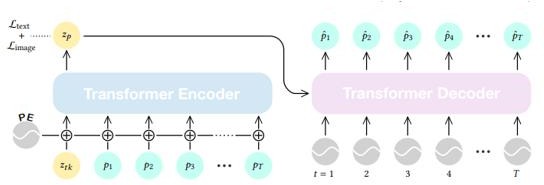


Fig. Motion Auto-Encoder. A transformer encoder is trained to project a motion sequence 1: into a latent vector in CLIP latent space. Simultaneously, a transformer decoder is trained to recover the motion by attending to .

Our goal is to learn a semantic and disentangled representation of motion, serving as the foundation for generation and editing tasks. To achieve this, we must not only learn the encoding of this representation but also its decoding back into explicit motion. Our training process, illustrated in Figure 2, involves training a transformer-based motion auto-encoder while aligning the latent motion manifold with the CLIP joint representation. This alignment is achieved through two main components: (i) a Text Loss, which connects motion representations to the CLIP embeddings of their corresponding text labels, and (ii) an Image Loss, which connects motion representations to the CLIP embeddings of visually rendered images depicting the motion.

During inference, applications such as semantic editing can be performed in the latent space. For instance, style transfer involves finding a latent vector representing the desired style, adding it to the content motion representation, and decoding the result

back into motion. Similarly, action classification can be achieved by encoding the action into the latent space and comparing it with the embeddings of class text labels. Additionally, the CLIP text encoder facilitates text-to-motion translation, where input text is decoded using the text encoder and directly decoded by our motion decoder. Detailed implementation of these applications is provided in Section 4.

Motion sequences are represented using the SMPL body model (Loper et al., 2015). A sequence of length T, denoted as p\_1:T, defines orientations in a 6D representation (Zhou et al., 2019) for global body orientation and 23 SMPL joints at each frame i. The mesh vertices locations v\_1:T are calculated according to SMPL specifications with neutral-gender body models (Petrovich et al., 2021).

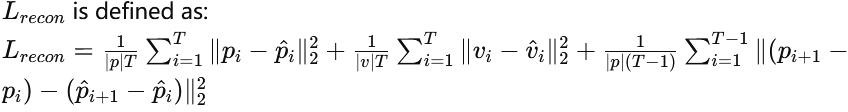
To project the motion manifold into the latent space, we utilize a transformer-based auto-encoder (Vaswani et al., 2017), adapted specifically for the motion domain (Petrovich et al., 2021; Wang et al., 2021b; Li et al., 2021).

Transformer Encoder: Denoted as E, it maps a motion sequence p\_1:T to its latent representation z\_p. Each frame is embedded into the encoder's dimension through linear projection, with standard positional embedding added. The embedded sequence, along with a learned prefix token z\_tk, serves as input to the transformer encoder. The resulting latent representation z\_p is the first output.

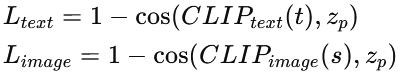
Transformer Decoder: Denoted as D, it predicts a motion sequence p\_hat\_1:T given a latent representation z\_p. This representation is used as key and value in the transformer, while the query sequence is the positional encoding from 1 to T. The transformer outputs a representation for each frame, which is then mapped to pose space using linear projection. Additionally, a differentiable SMPL layer is used to derive the mesh vertices locations v\_hat\_1:T.

Losses: The auto-encoder is trained using reconstruction L\_2 losses on joint

orientations, velocities, and vertices locations. Specifically, the reconstruction loss



Text-motion and image-motion pairs (p\_1:T, t), (p\_1:T, s) respectively are utilized to link motion representations to text and image representations using cosine distance:



Motion-text pairs are derived from labeled motion datasets, while images are rendered from motion sequences to synthetic images s in an unsupervised manner. The overall loss objective of MotionCLIP is:



## Datasets

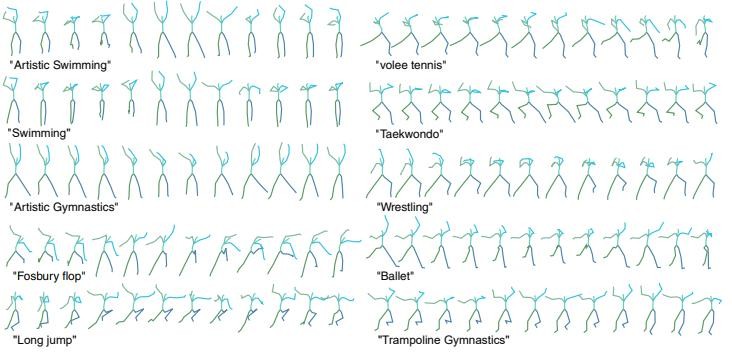
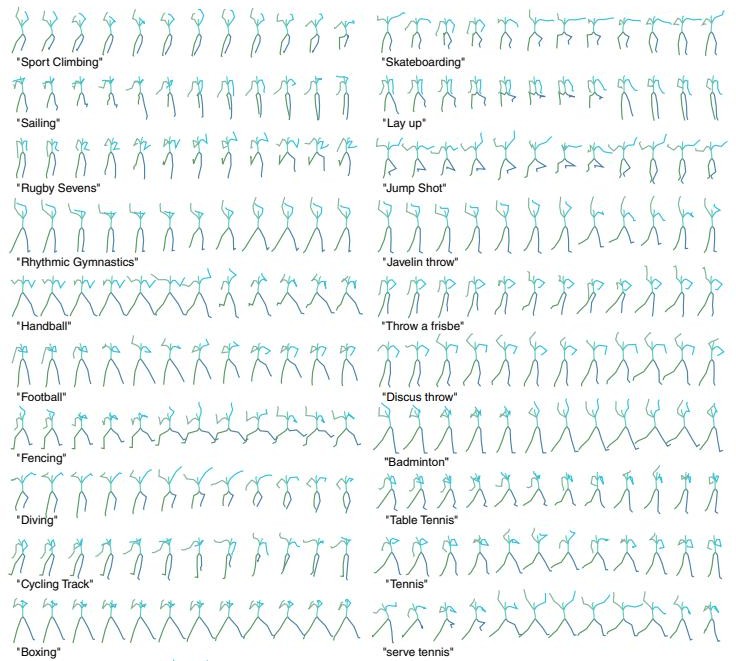


Figure. Sport motions generated by MotionCLIP conditioned on the text beneath each row

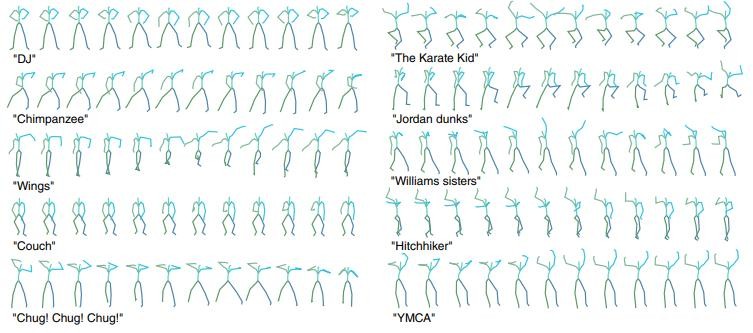


Fig. Abstract language. MotionCLIP generates the signature motions of culture figures and phrases

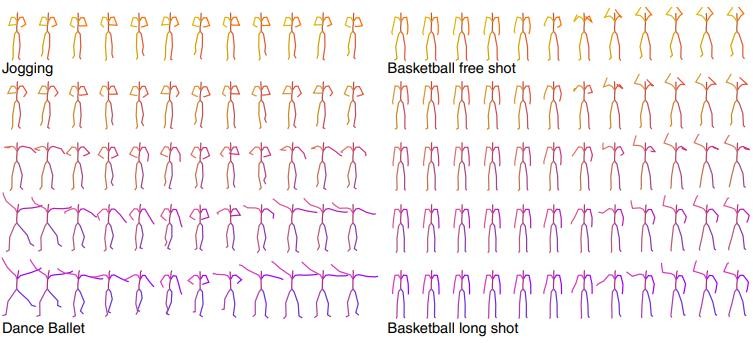


Fig. Latent space motion interpolation. MotionCLIP enables semantic interpolation between two motions

We train our model on the BABEL dataset. It comprises about 40 hours of motion capture data, represented with the SMPL body model. Each frame is annotated with per-frame textual labels, and is categorized into one of 260 action classes. We down sample the data to 30 frames per-second and cut it into sequences of length 60. We get a single textual label per sequence by listing all actions in a given sequence, then concatenating them to a single string. Finally, we choose for each motion sequence a random frame to be rendered using the Blender software and the SMPL-X add-on.

This process outputs triplets of (motion, text, synthetic image) which are

used for training.

We train a transformer auto-encoder with 8 layers for each encoder and decoder as described in Section 3. We align it with the CLIP-ViT-B/32 frozen model. Out of the data triplets, the text-motion pairs are used for the text loss and image-motion pairs for the image loss. Both values are set to 0.01 throughout our experiments.

## Model

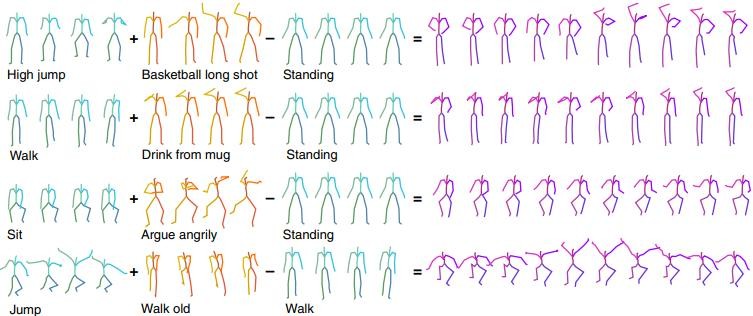
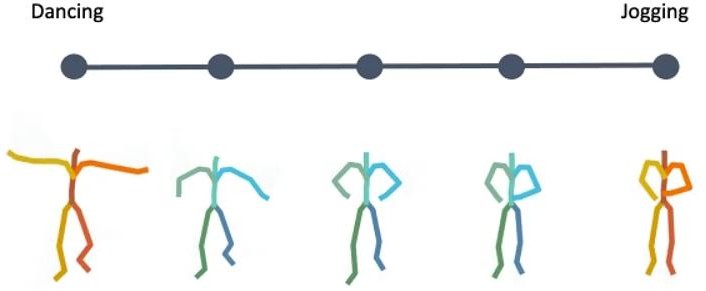


Fig. Latent space motion editing. MotionCLIP enables semantic editing in latent space. Here we demonstrate two applications (1) upper and lower body action compositions (top two examples) and (2) style transfer (the two examples at the bottom).

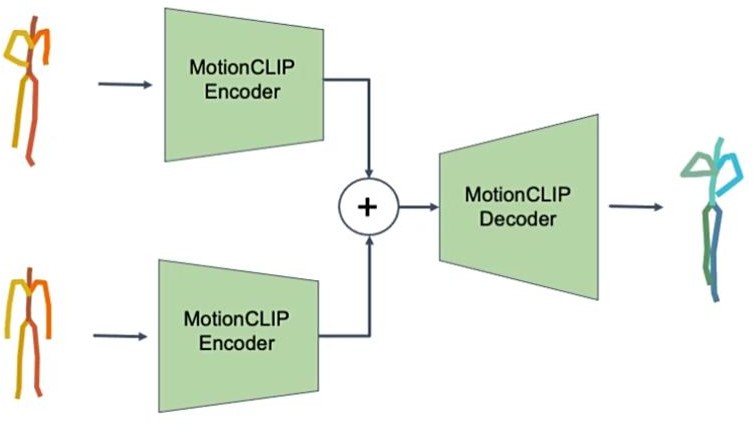
It is already well established that the CLIP space is smooth and expressive. We demonstrate its merits also exist in the aligned motion manifold, through the following experiments.

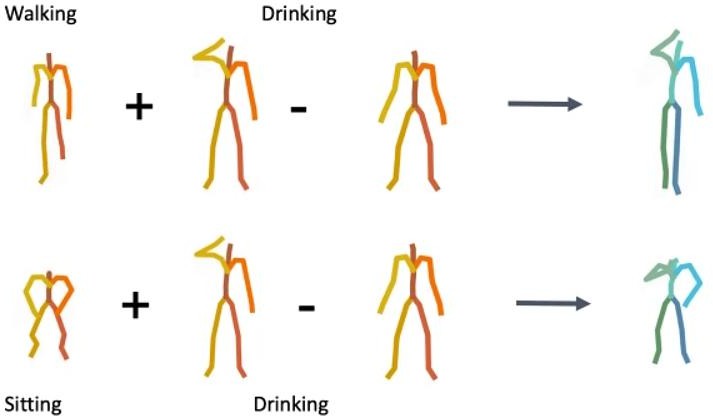
**Interpolation** linear interpolation between two latent codes yields semantic transitions between motions in both time and space. This is a strong indication to the smoothness of this representation. Here, the source and target motions (top and bottom respectively) were sampled from the validation set, and between them are three transitions evenly sampled from the linear trajectory between the two motion representations, then decoded by MotionCLIP.



**Latent-Based Editing** To demonstrate how disentangled and uniform MotionCLIP latent space is, we experiment with latent space arithmetic to edit motion.

As can be seen, these linear operations allow motion compositionality - the upper body action can be decomposed from the lower body one, and recomposed with another lower body performance. In addition, Style can be added by simply adding the vector of the style name embedding. These two properties potentially enable intuitive and semantic editing even for novice users.



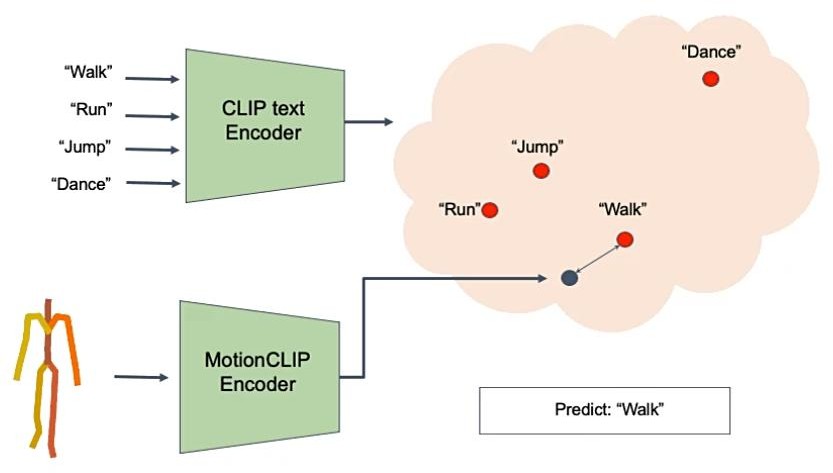


**Action Recognition** Finally, we further demonstrate how well our latent spaces is semantically structured. We show how combined with the CLIP text encoder, MotionCLIP encoder can be used for action recognition. We follow BABEL

60-classes benchmark and

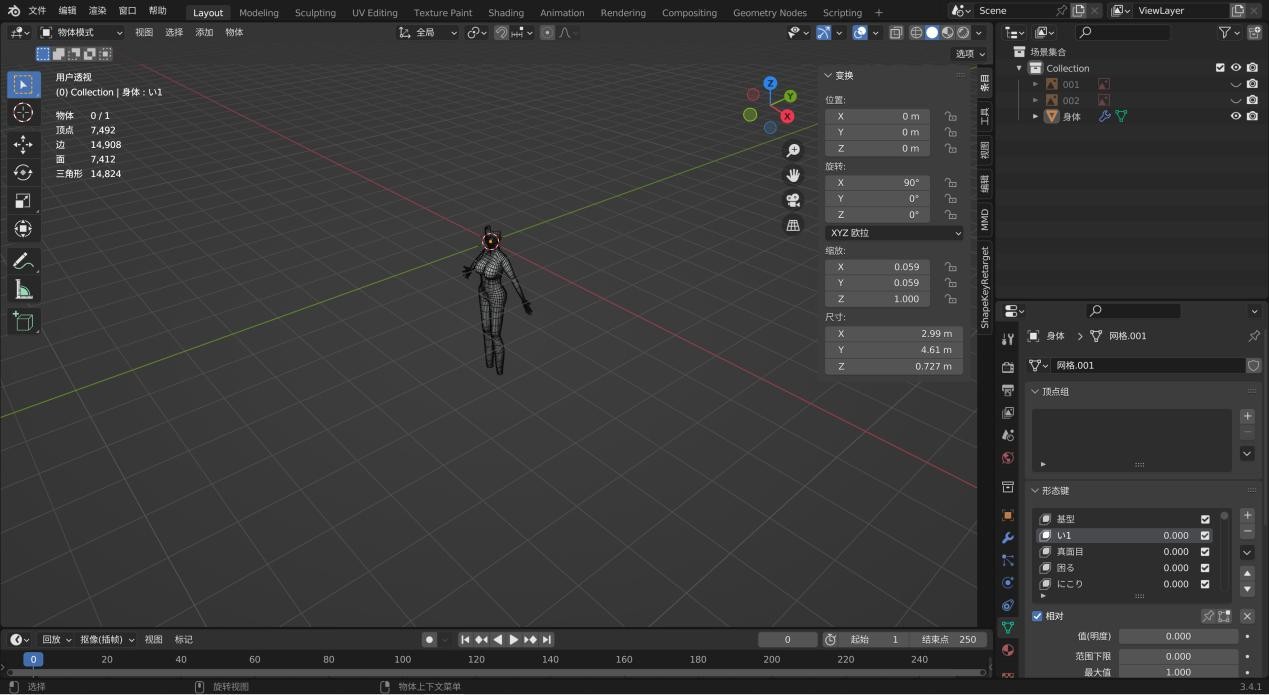
train the model with BABEL class names instead of the raw text. At inference, we measure the cosine distance of a given motion sequence to all 60 class name encodings and apply softmax, as

suggested originally for image classification.

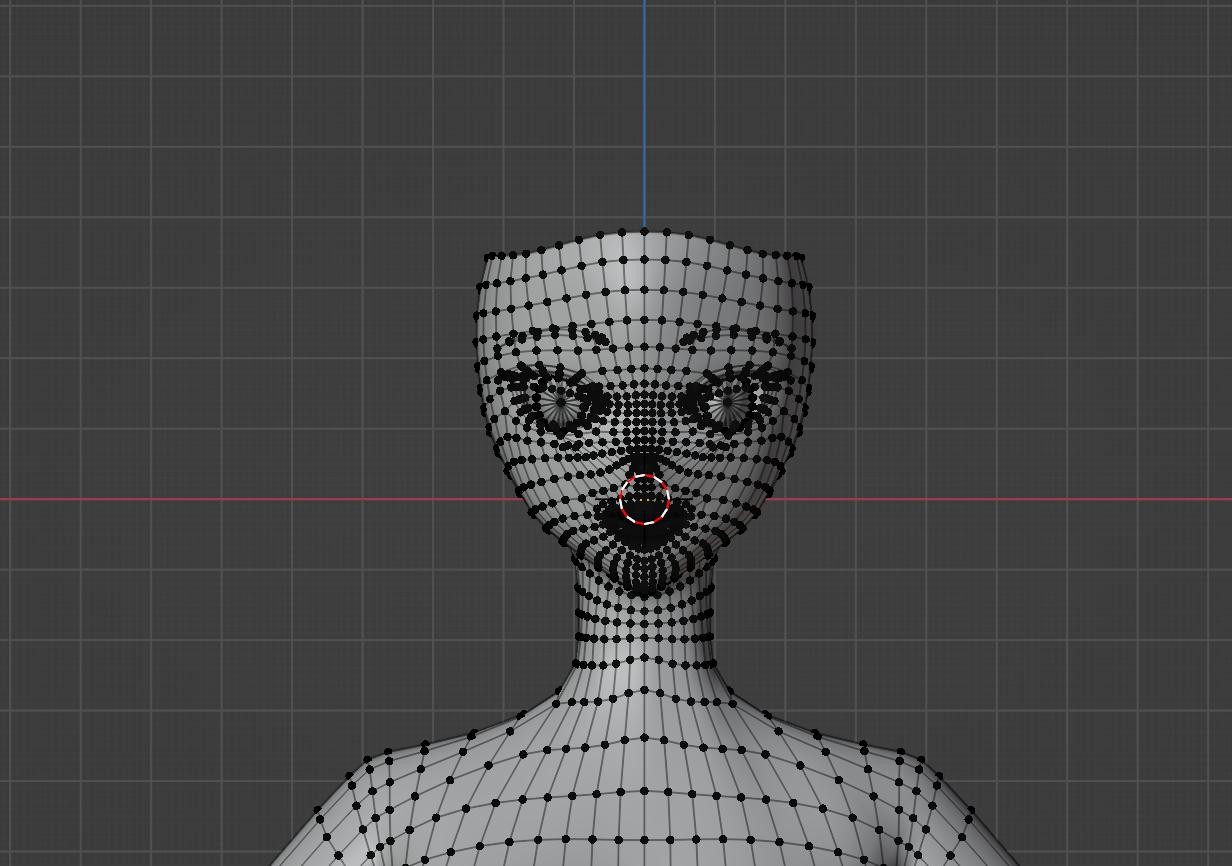


## Method of Motion Visualization

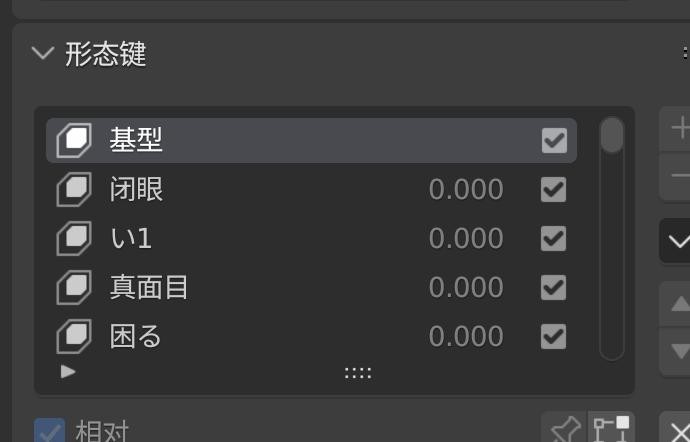
Modeling

Find orthographic projection references to complete the basic body modeling.

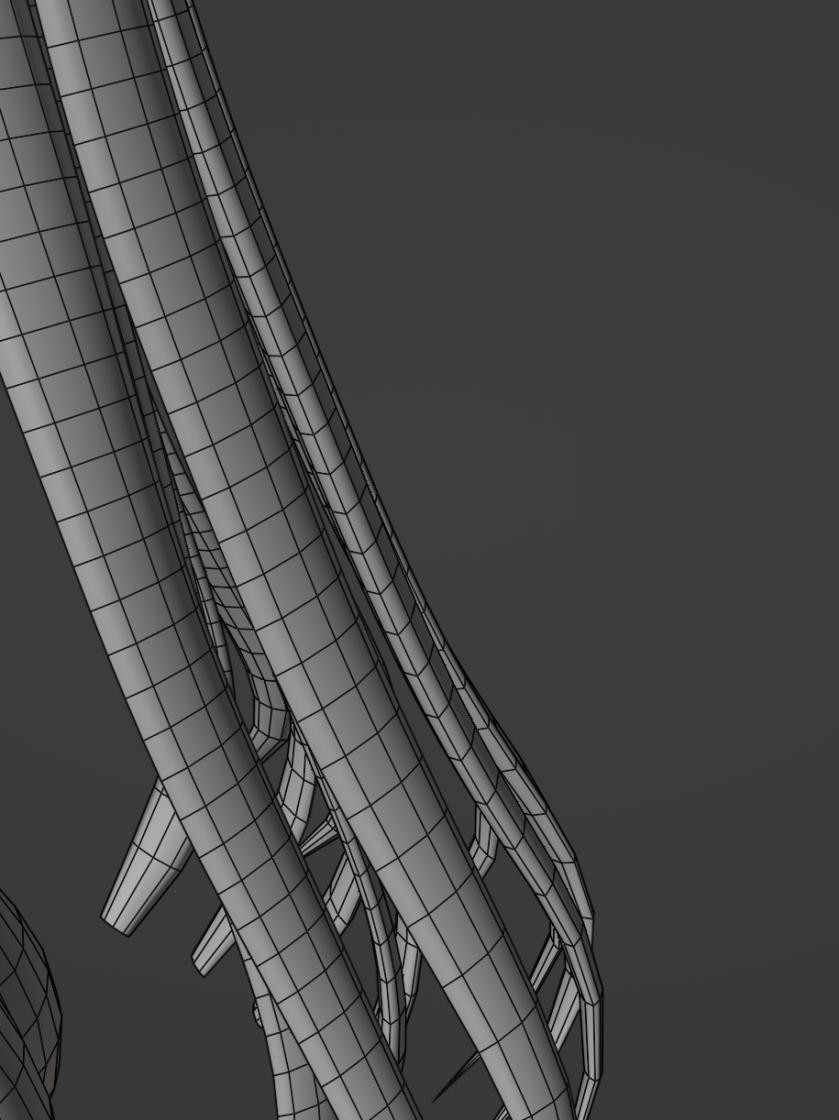
Using blendshapes to handle facial expressions



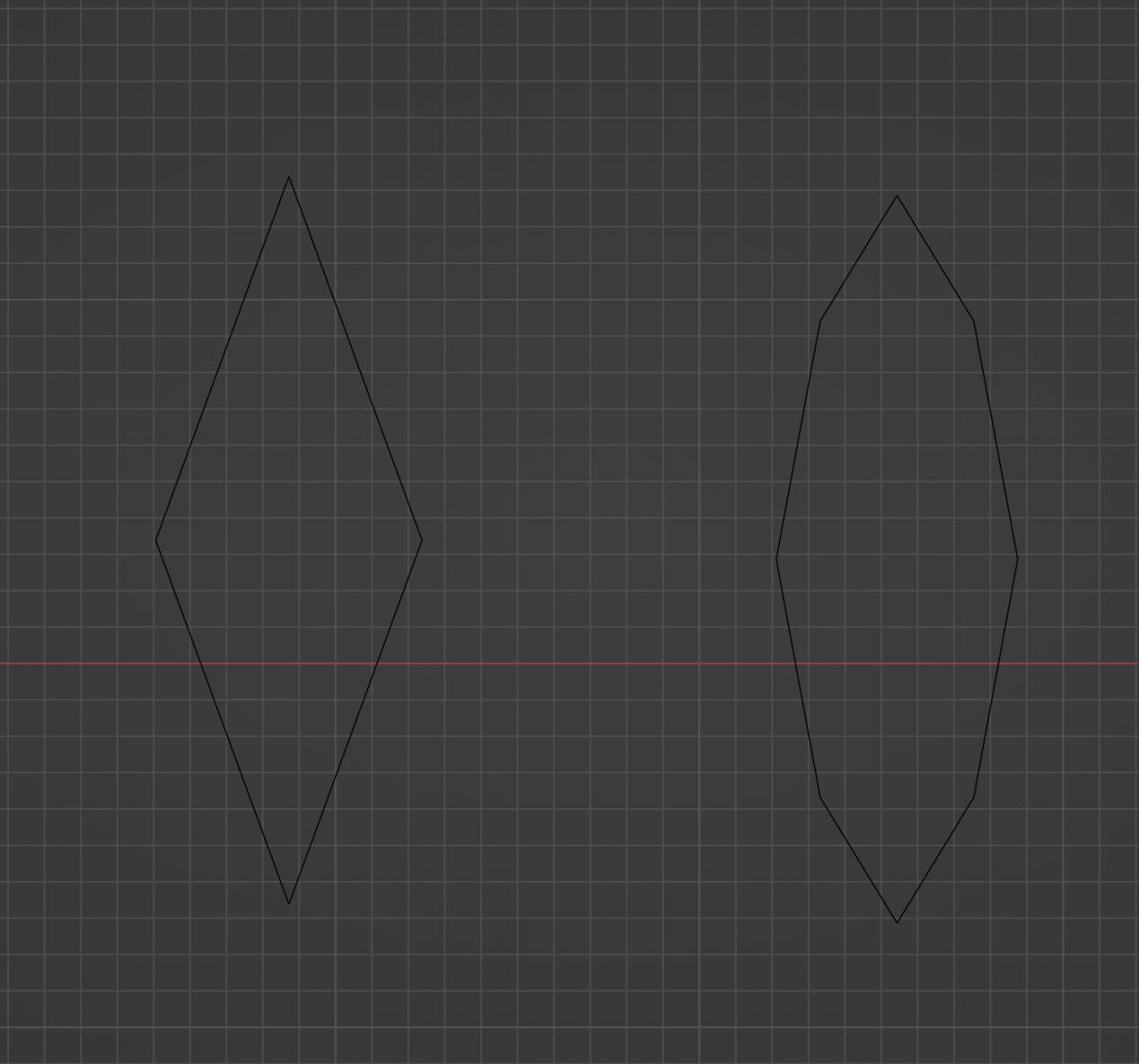
Some of the blendshapes are replicated from a reference model, using lattice deformation to conform the current model to the surface of the reference model. Then, a plugin is used for blendshape transfer. The rest of the blendshapes are created manually.



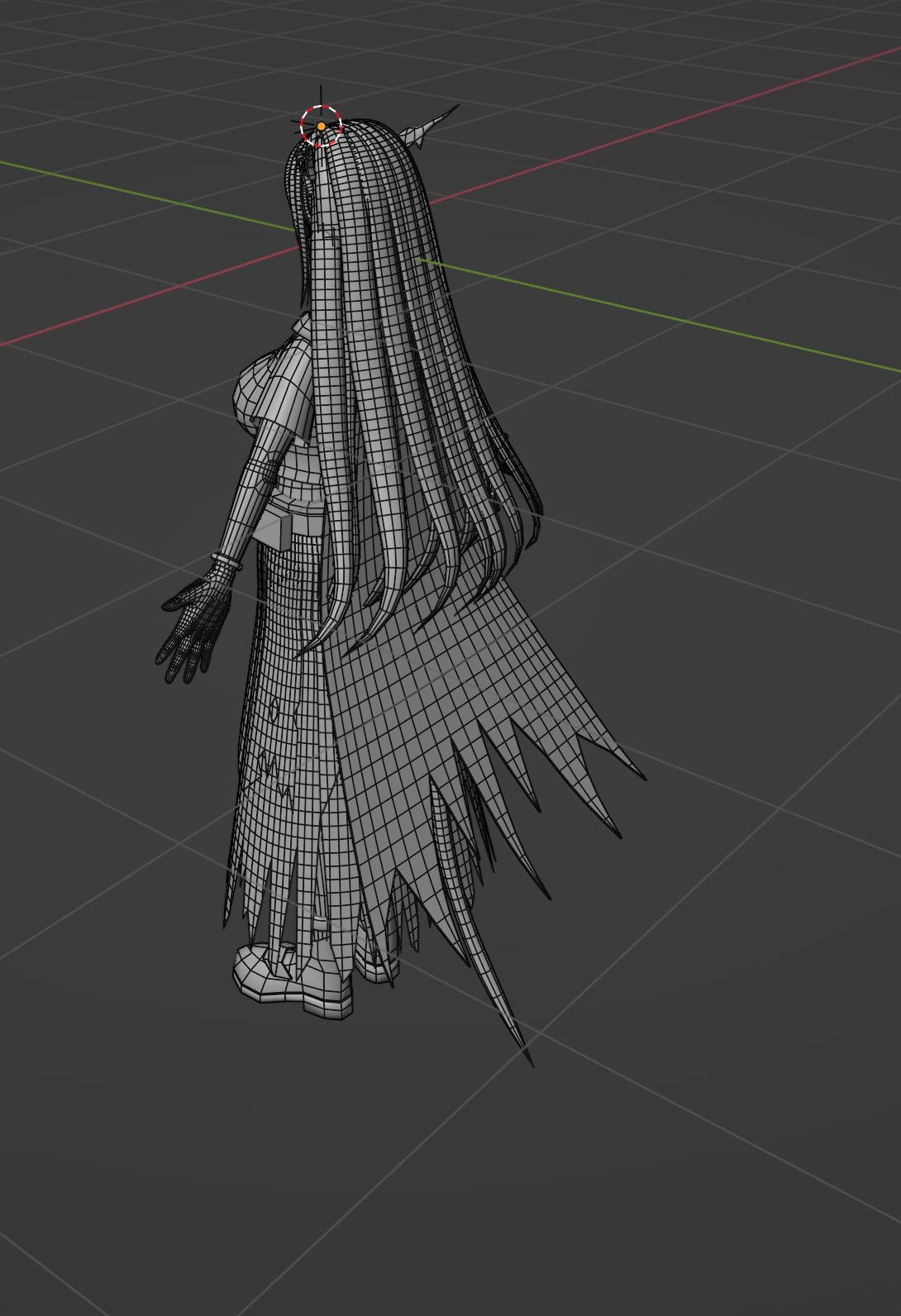
Hair is modeled using curves.



The hair curves have two basic shapes.



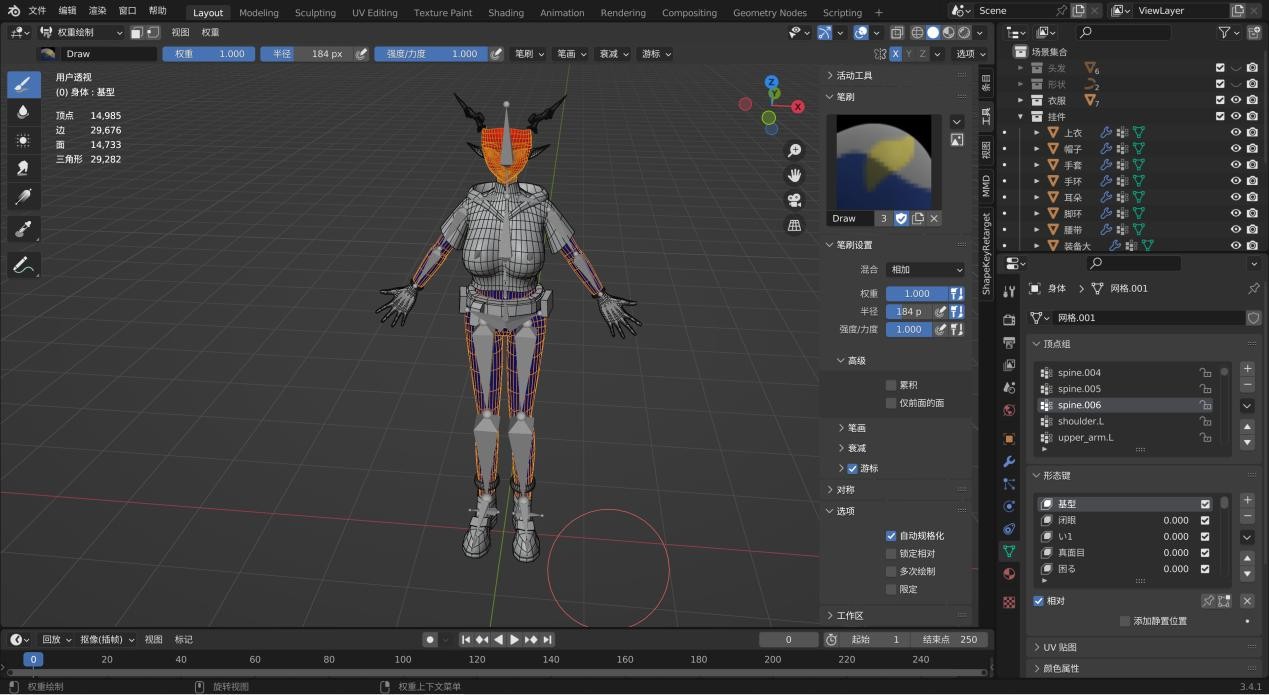
Modeling Clothes and Other Components



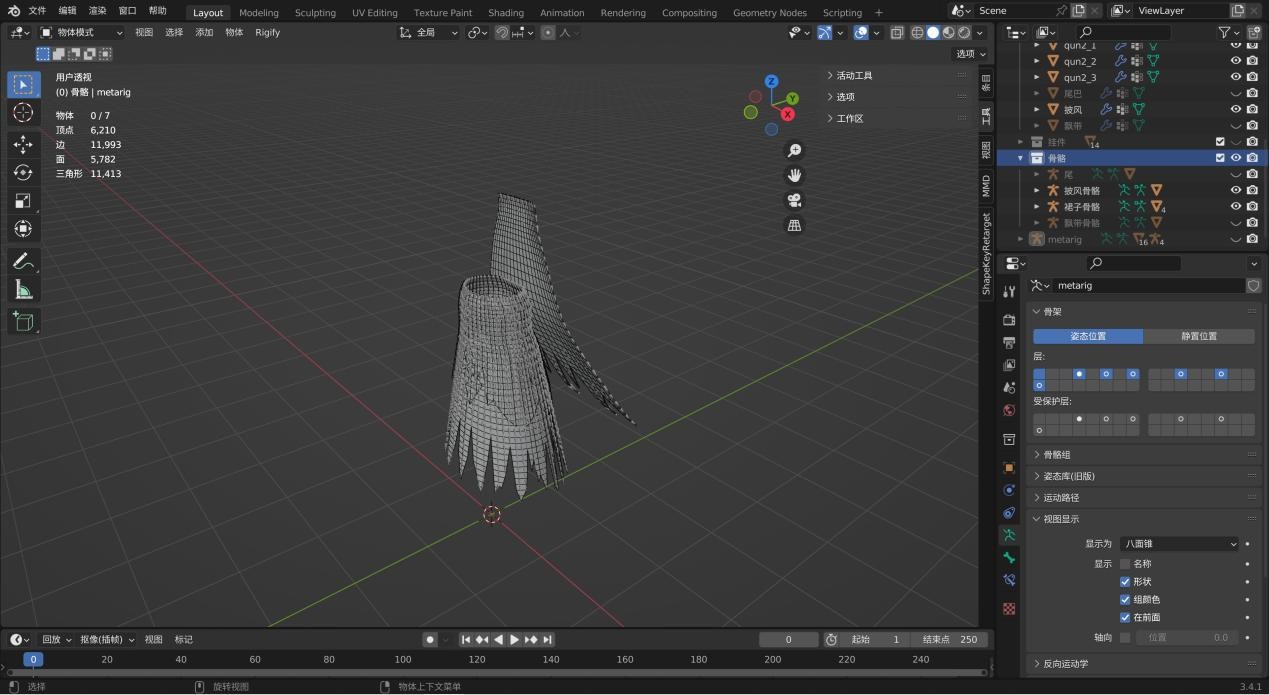
Skeleton Rigging for the human body, based on the Human (meta-rig) plugin.



Assigning vertex groups and painting weights corresponding to bones



Clothing Skeleton



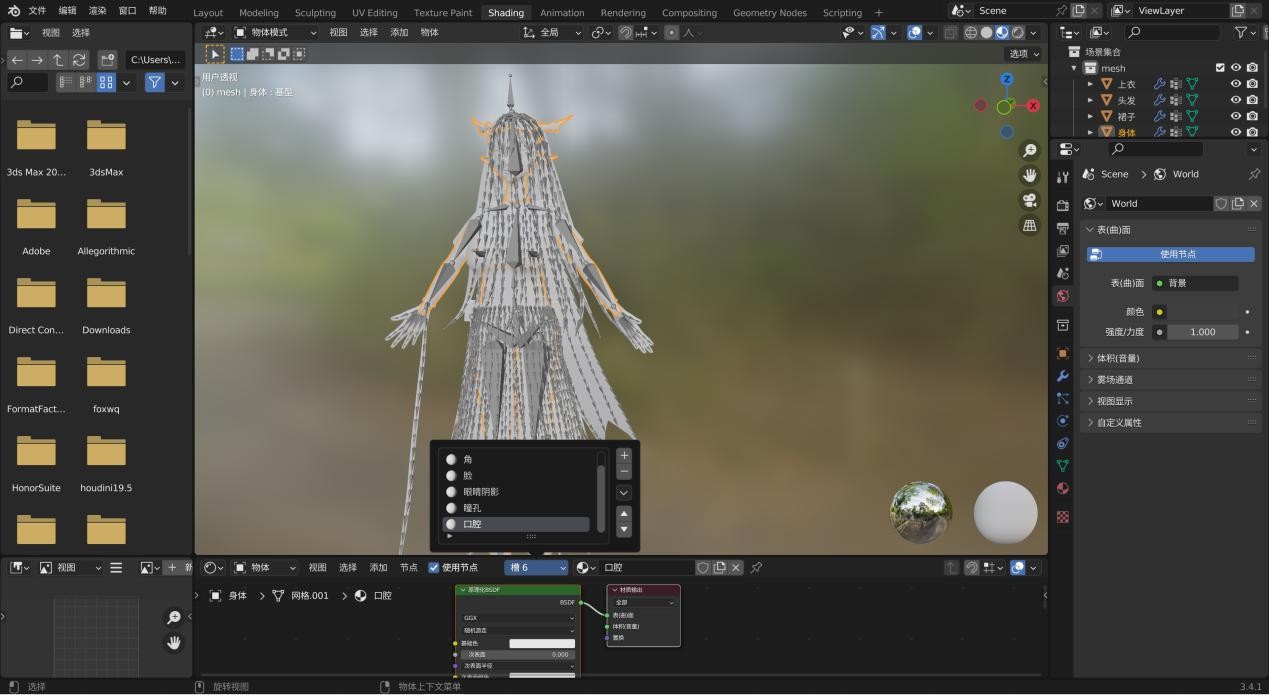
Weight Painting



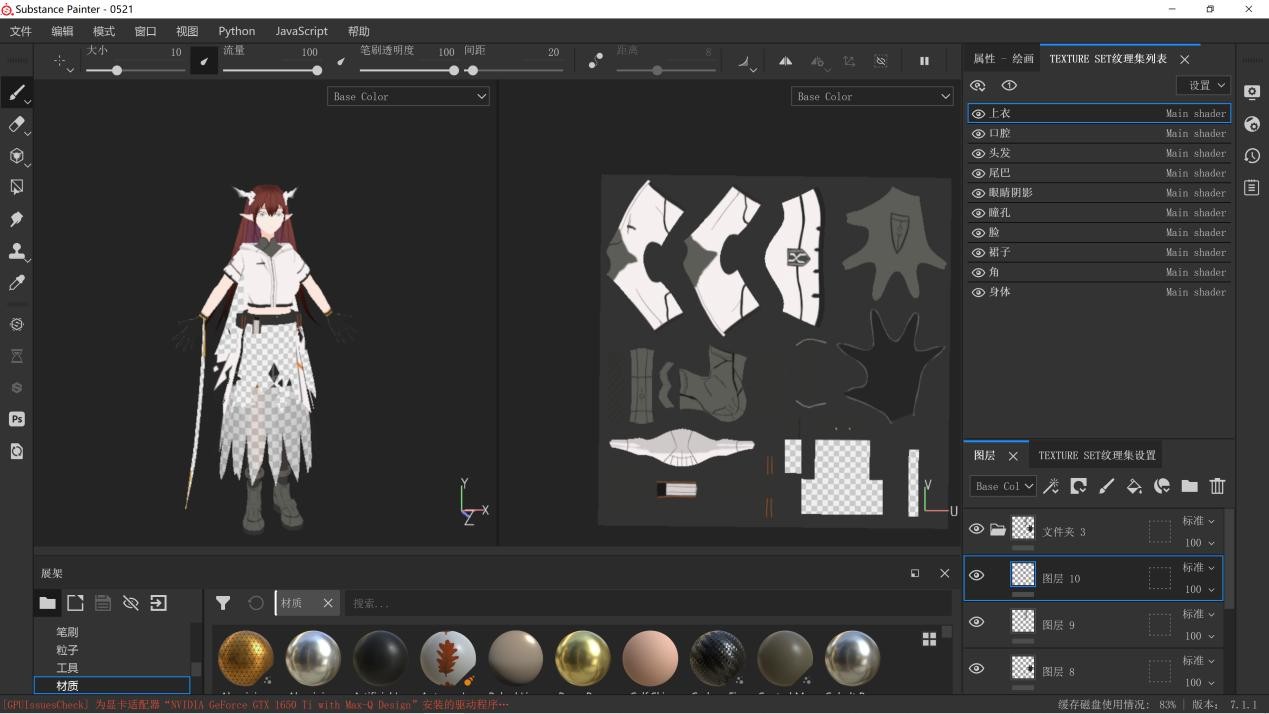
Merge Meshes and Unwrap UVs



Applying Multiple Materials to a Single Mesh



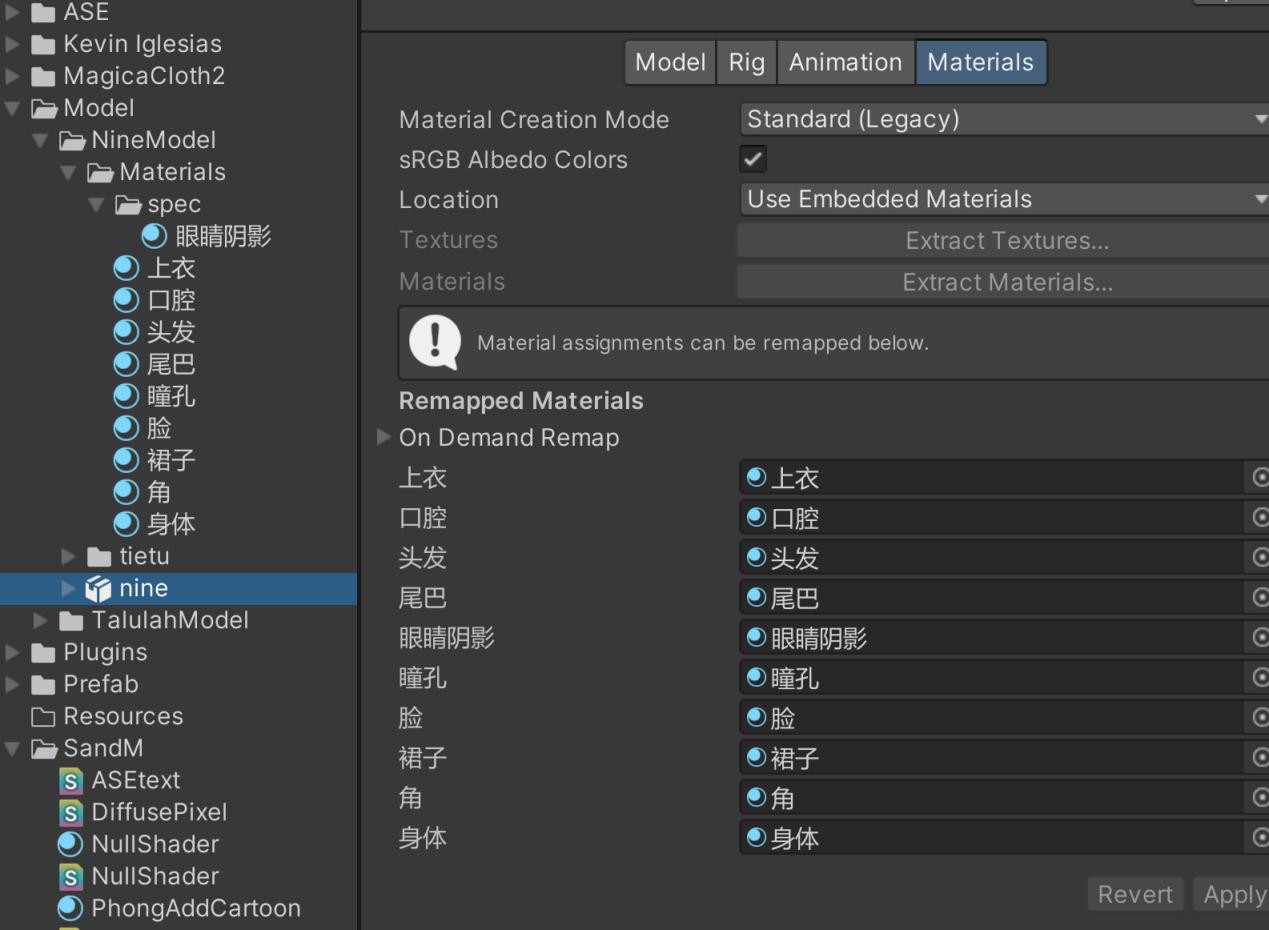
Exporting FBX for Texture Painting in Substance Painter

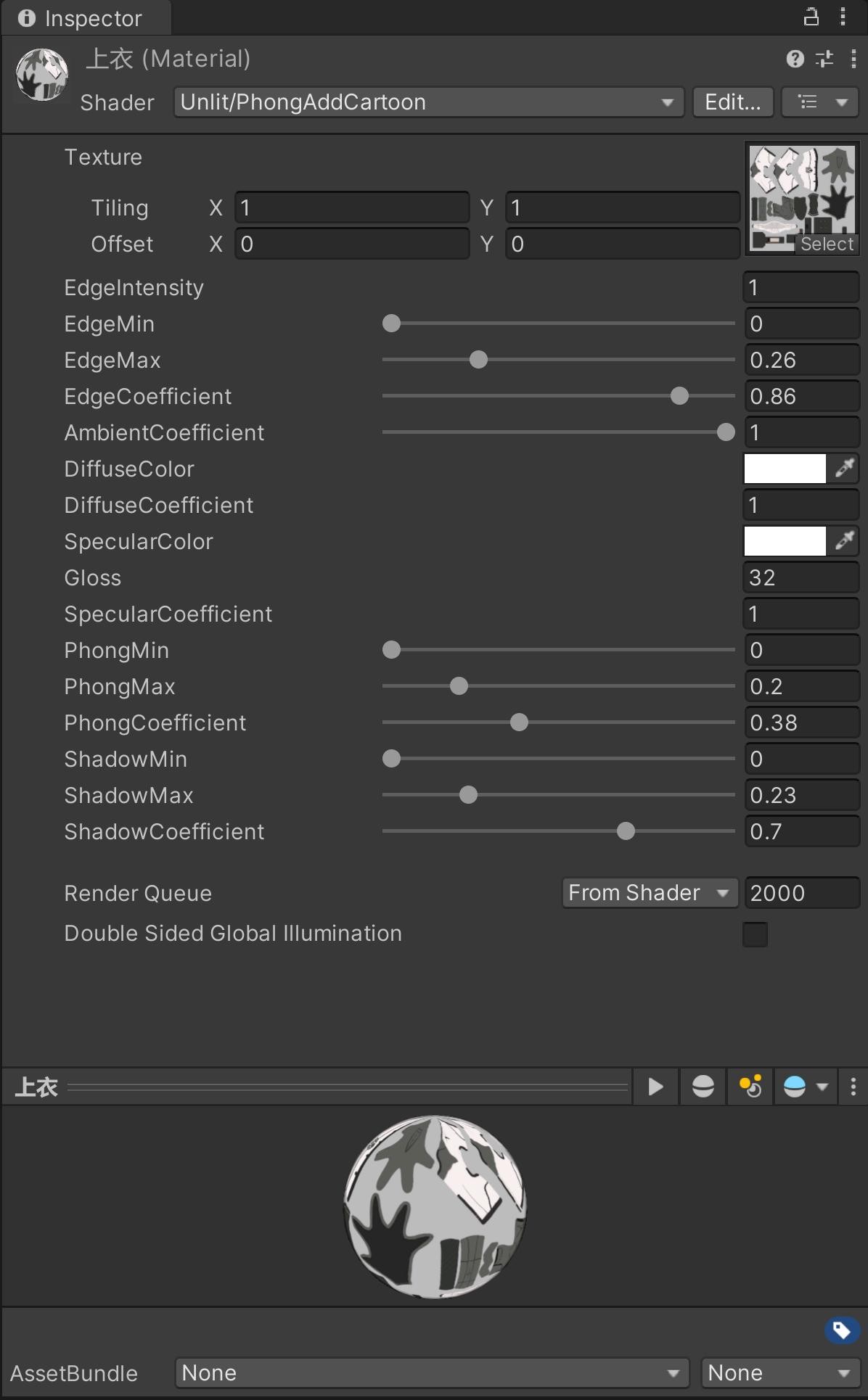


Export the painted textures and merge them in Photoshop (one per mesh).



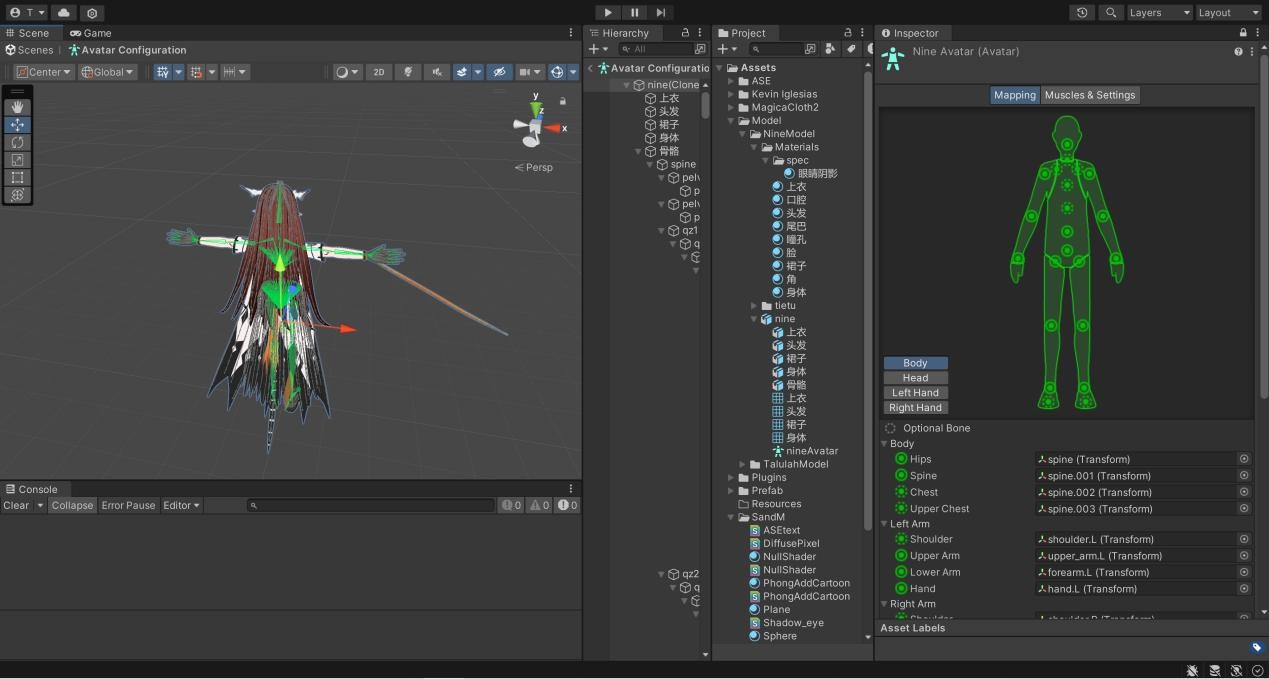
Import the FBX and textures into Unity, then match the materials with their respective textures.



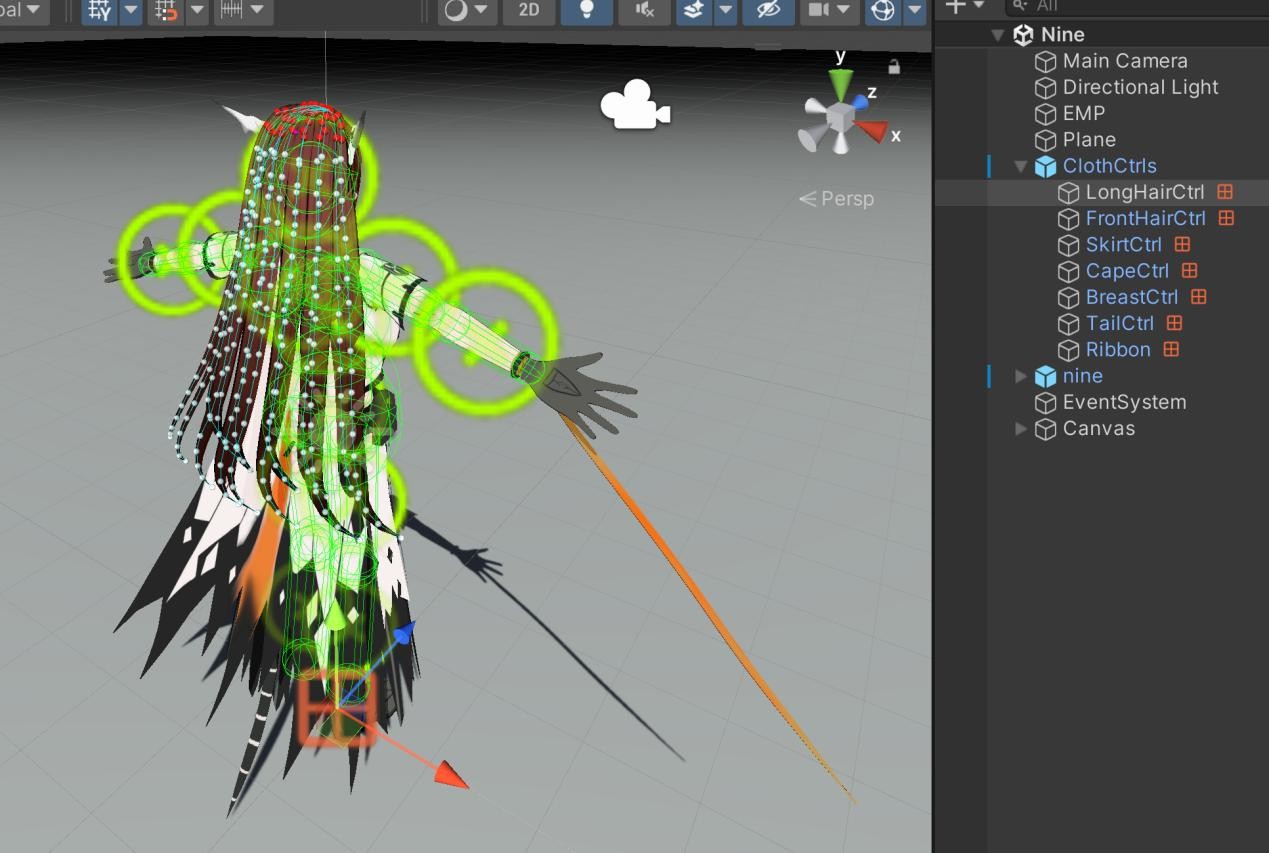


Then proceed with bone matching.

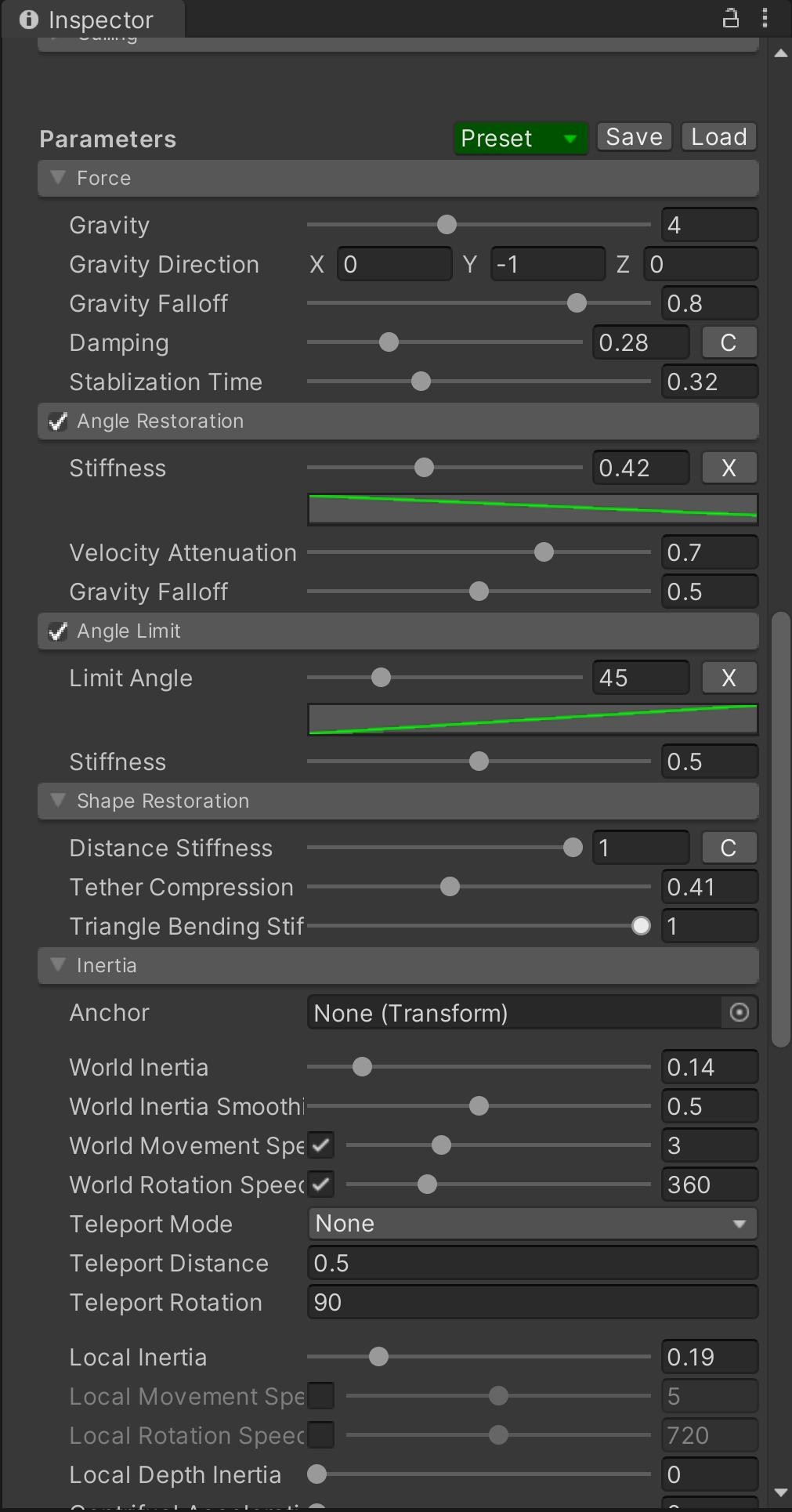




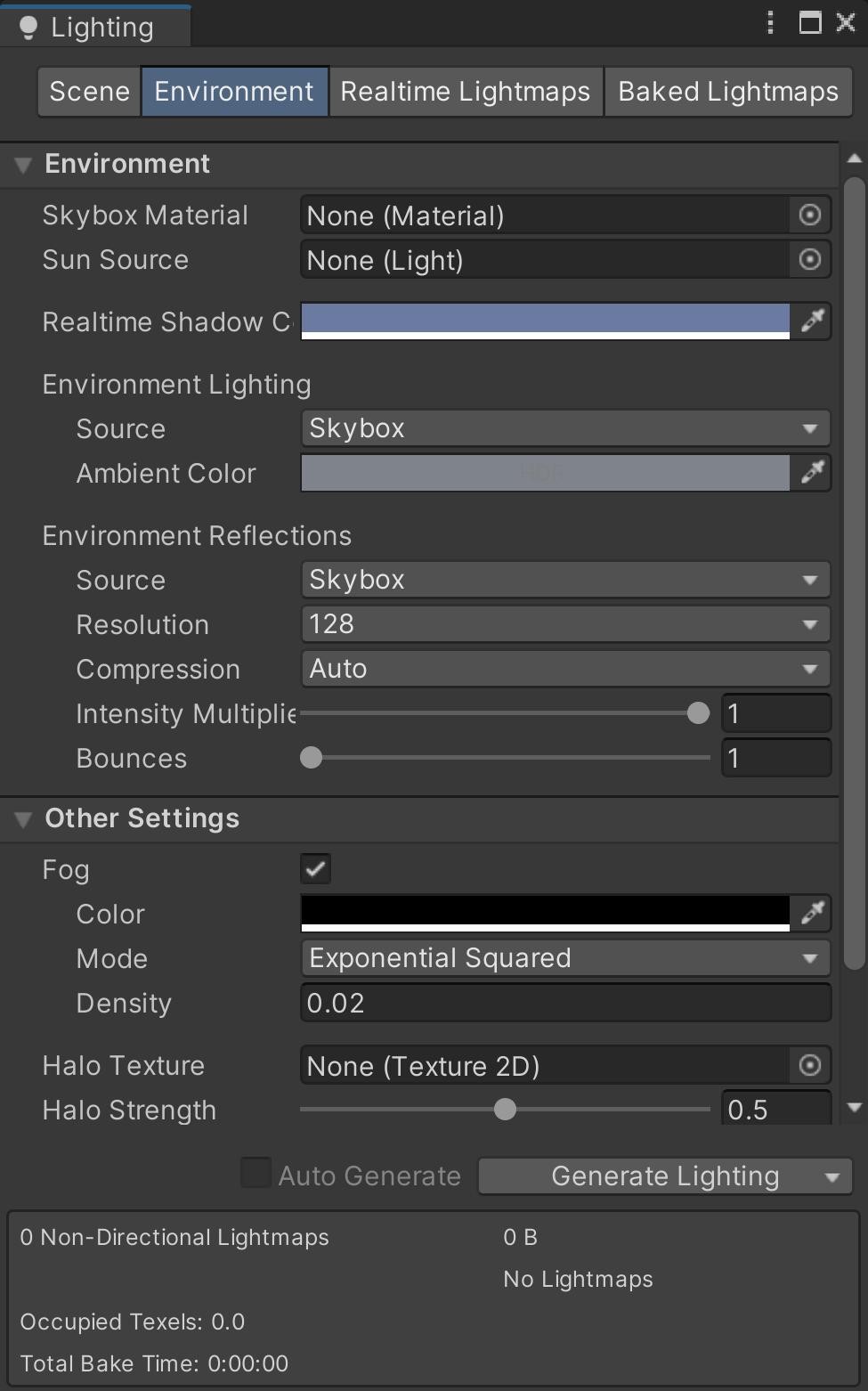
Next, place the model into the scene, add cloth and hair physics effects, apply colliders, based on MagicaCloth2.



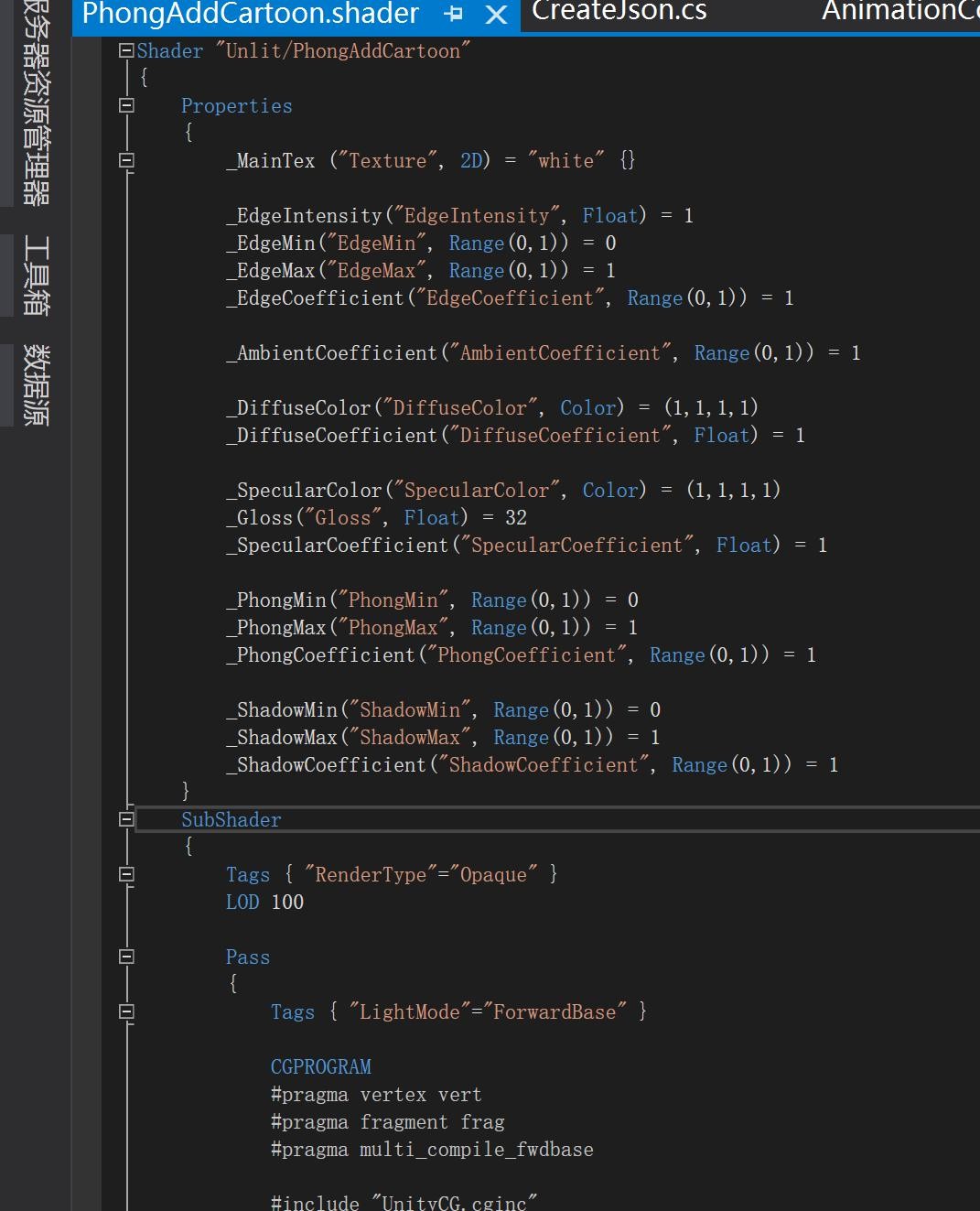
Fix parameter



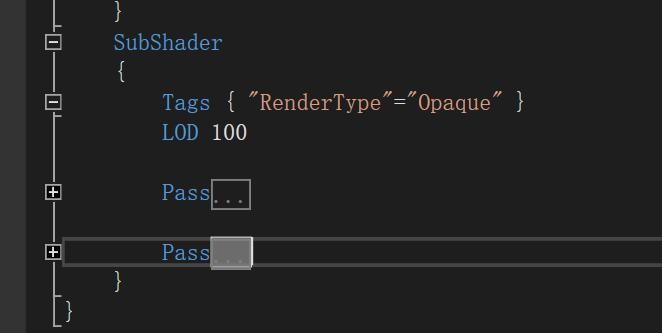
Setting up basic scene and configuring rendering environment.



Variable definition

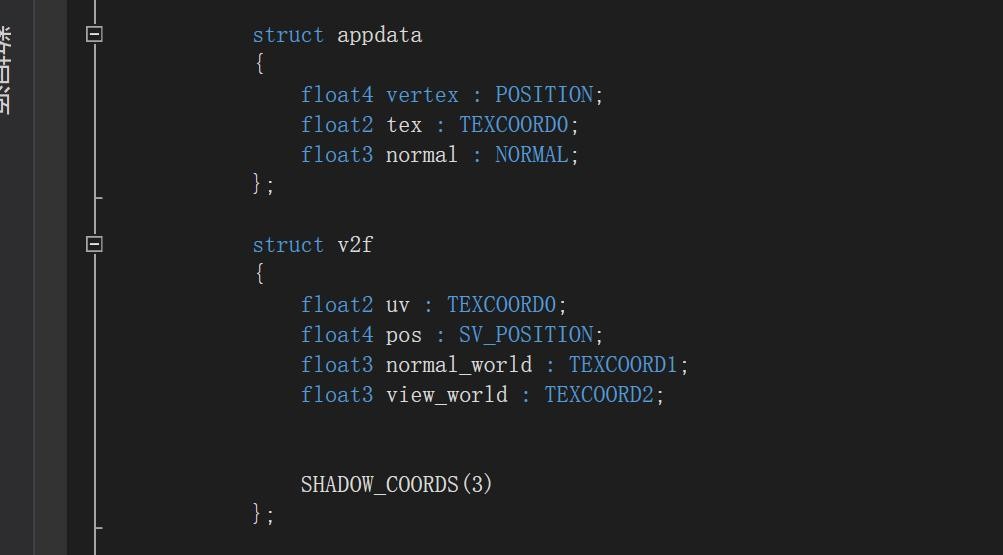


Rendering includes a regular rendering pass and a shadow rendering pass.

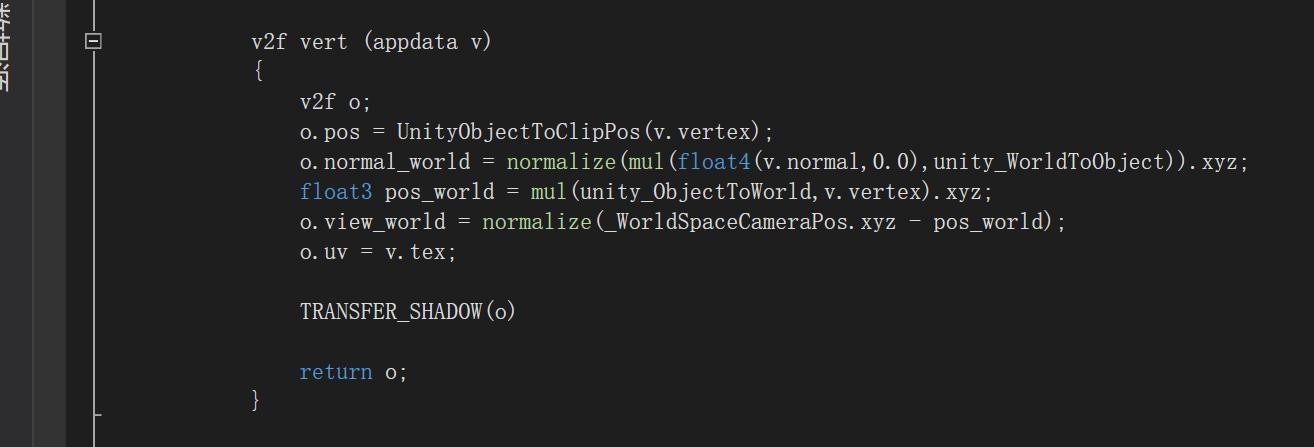


Regular rendering pass:

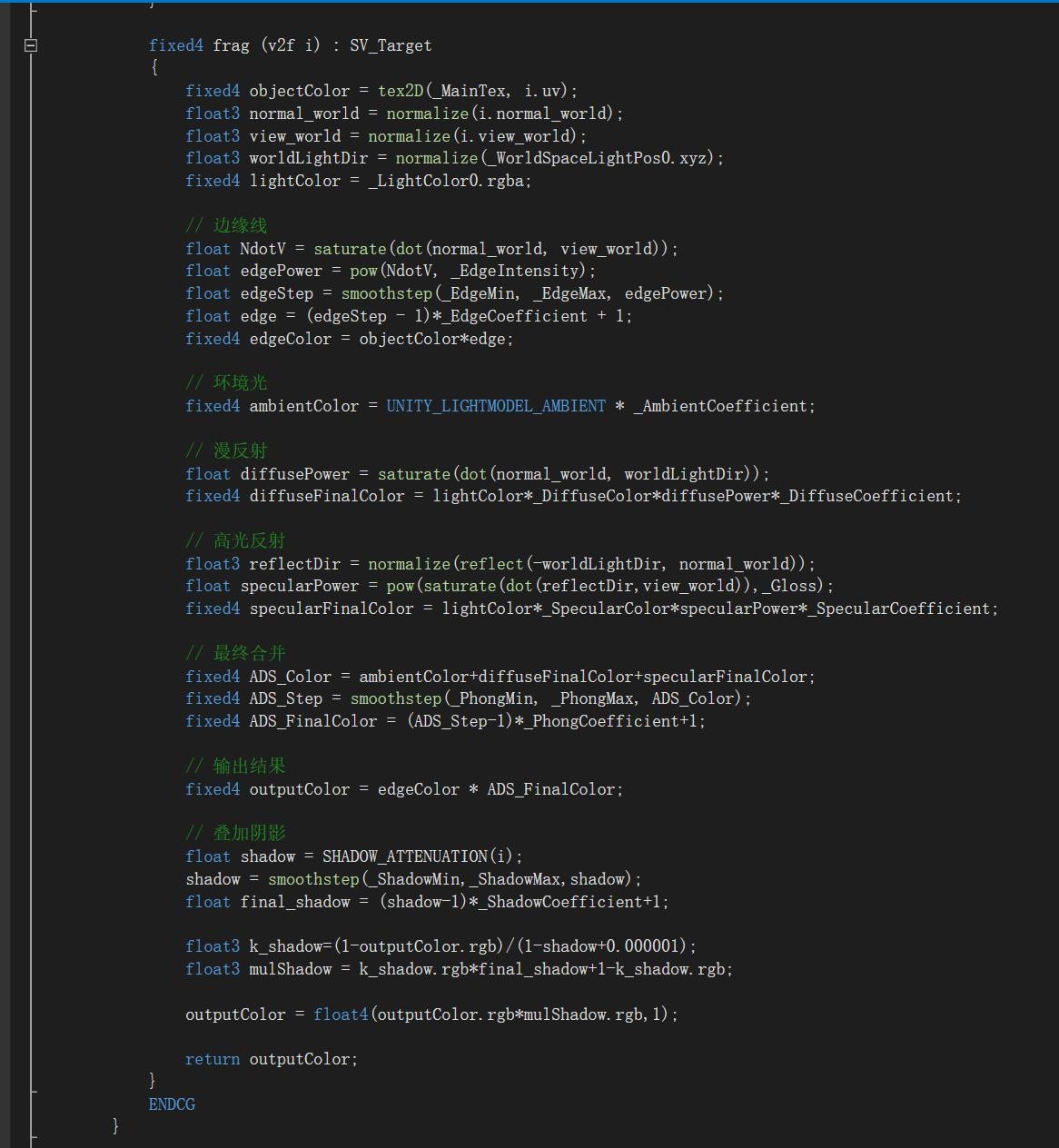
Define the basic structure and obtain the shadow information.



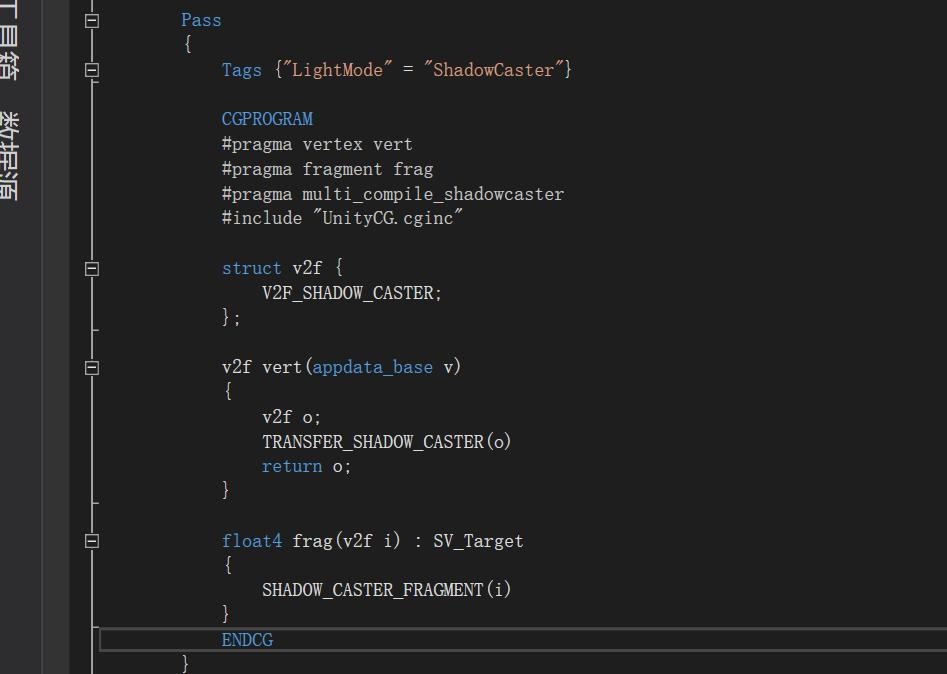
Vertex shader without actual operations.



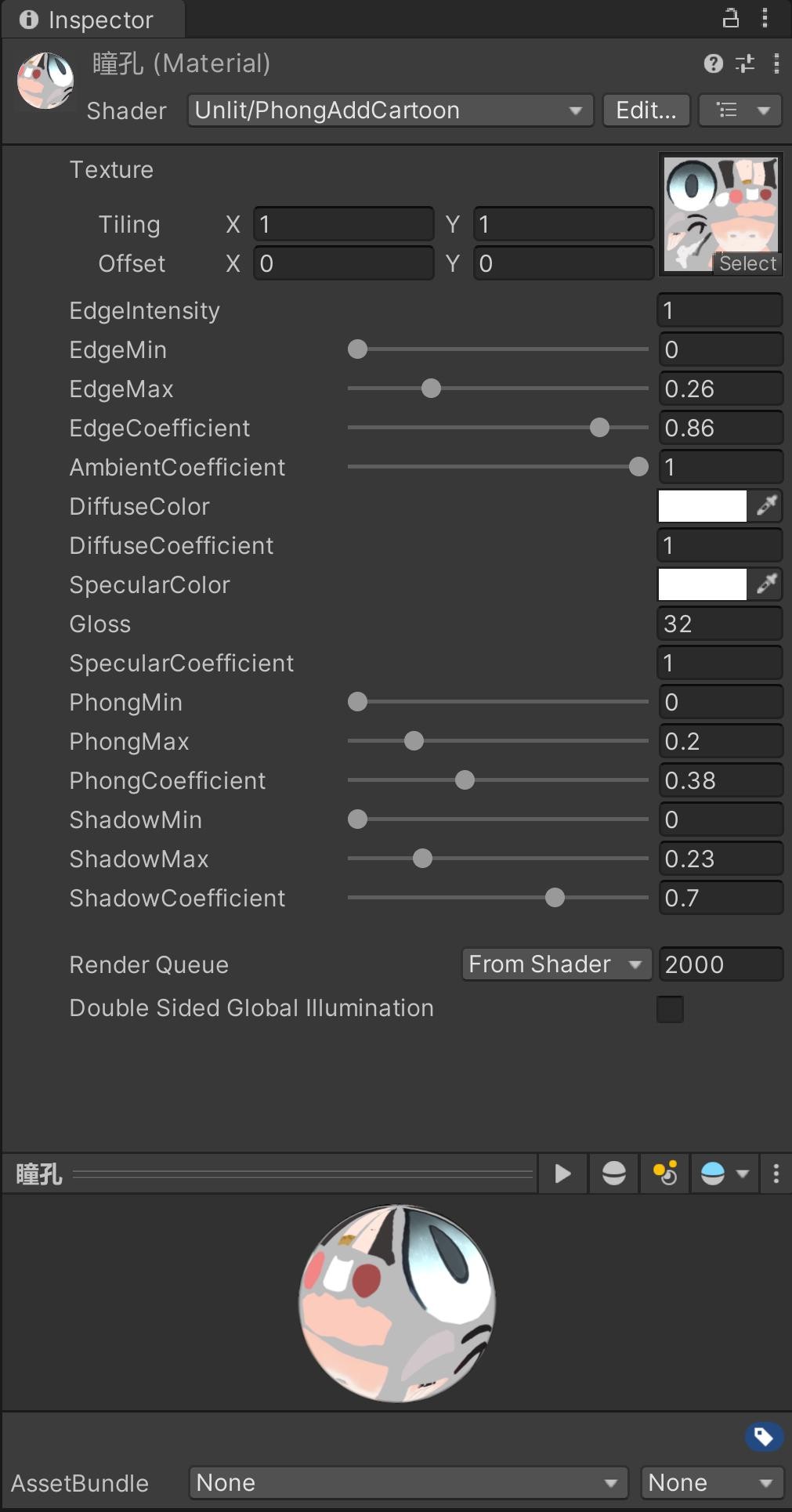
Fragment shader implementing basic lighting model + edge detection + binarization + shadow multiply blend.



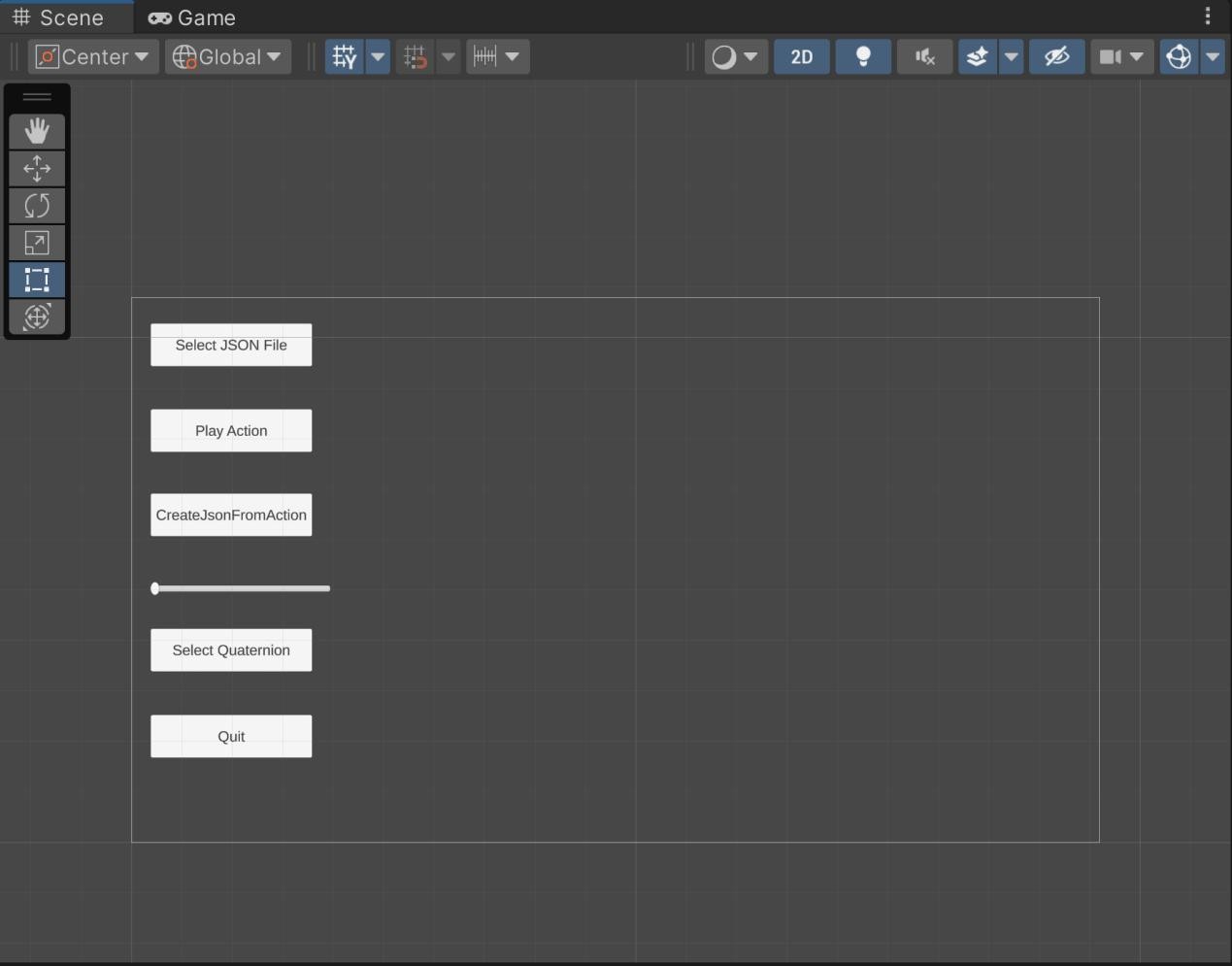
Shadow channel, recording shadow information.



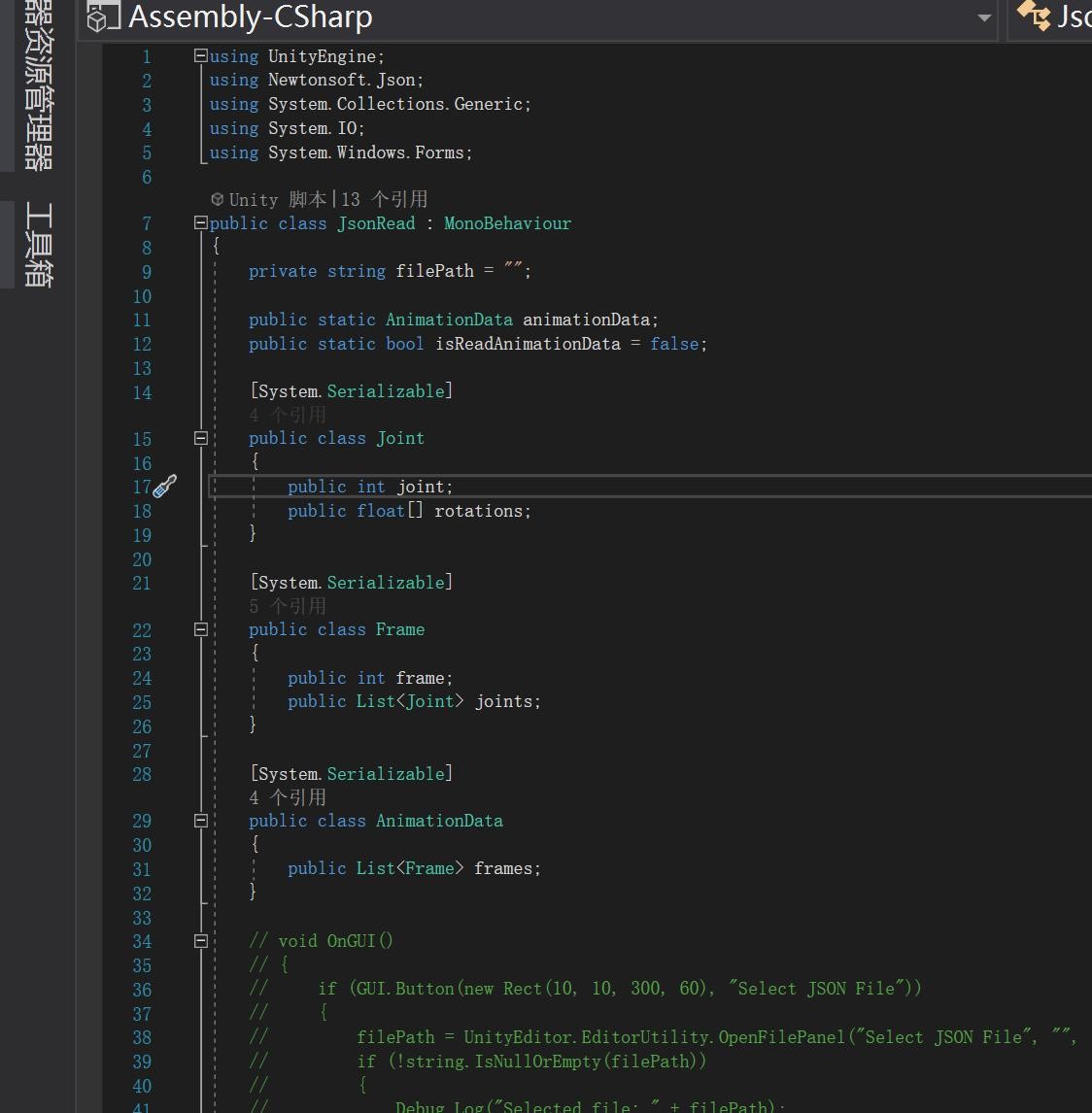
Applying shaders practically in Unity



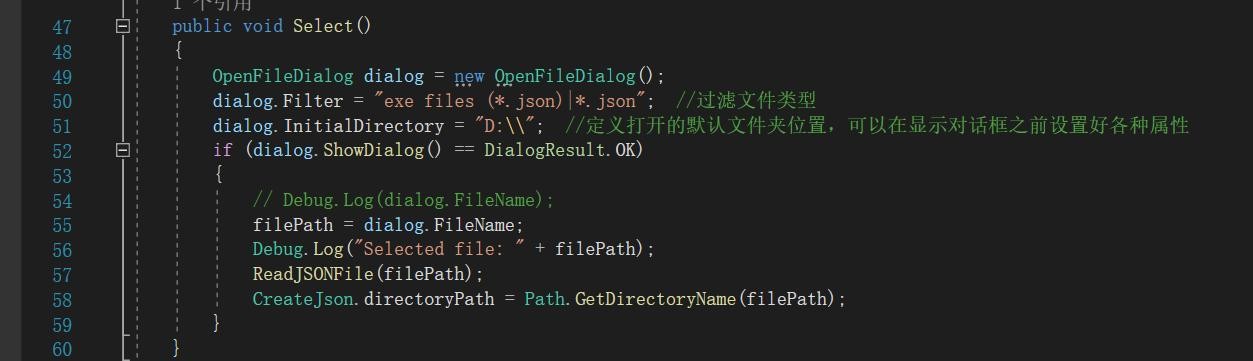
Creating UI interfaces in Unity.



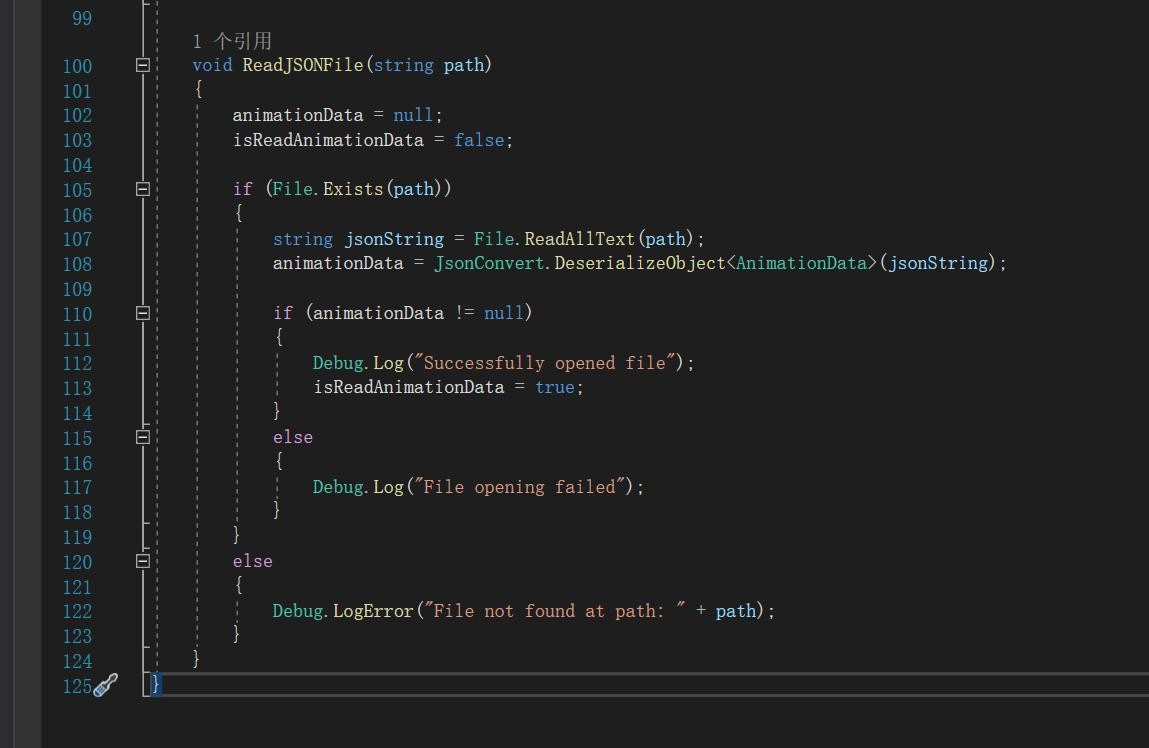
JsonRead.cs read json file.



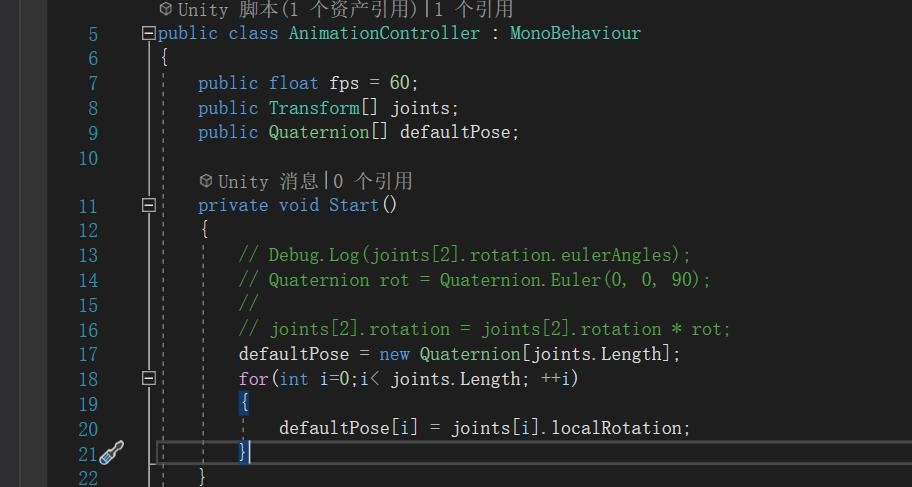
Using `System.Windows.Forms.dll` to create a Windows file selection window (this requires importing the library from another location on the computer into Unity, and is only applicable for Windows platforms).



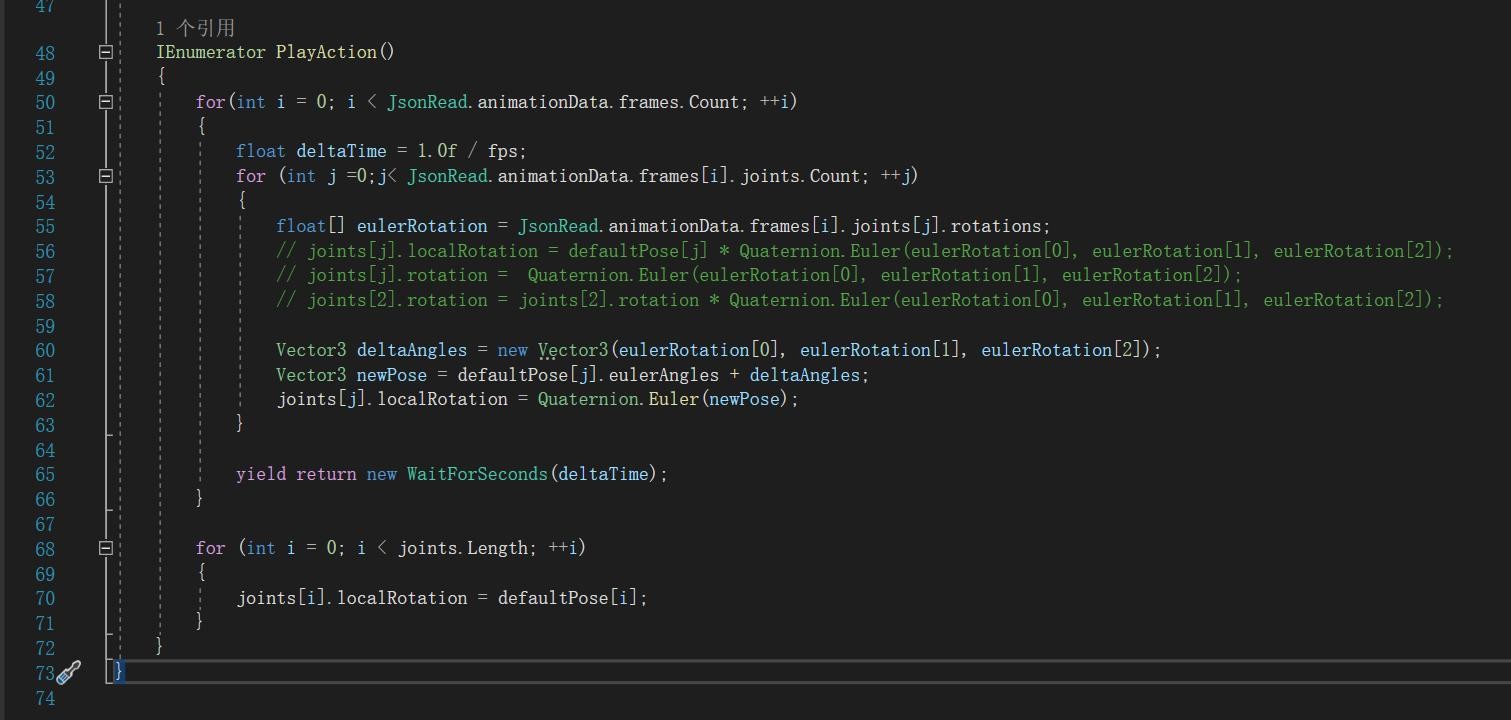
Read and parse JSON file.



AnimationController.cs: Play the JSON file loaded onto the model. First record the initial state of the animation information.

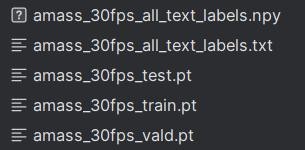


Using a coroutine to periodically apply changes from a JSON file to a model, and finally restoring the model to its initial state.



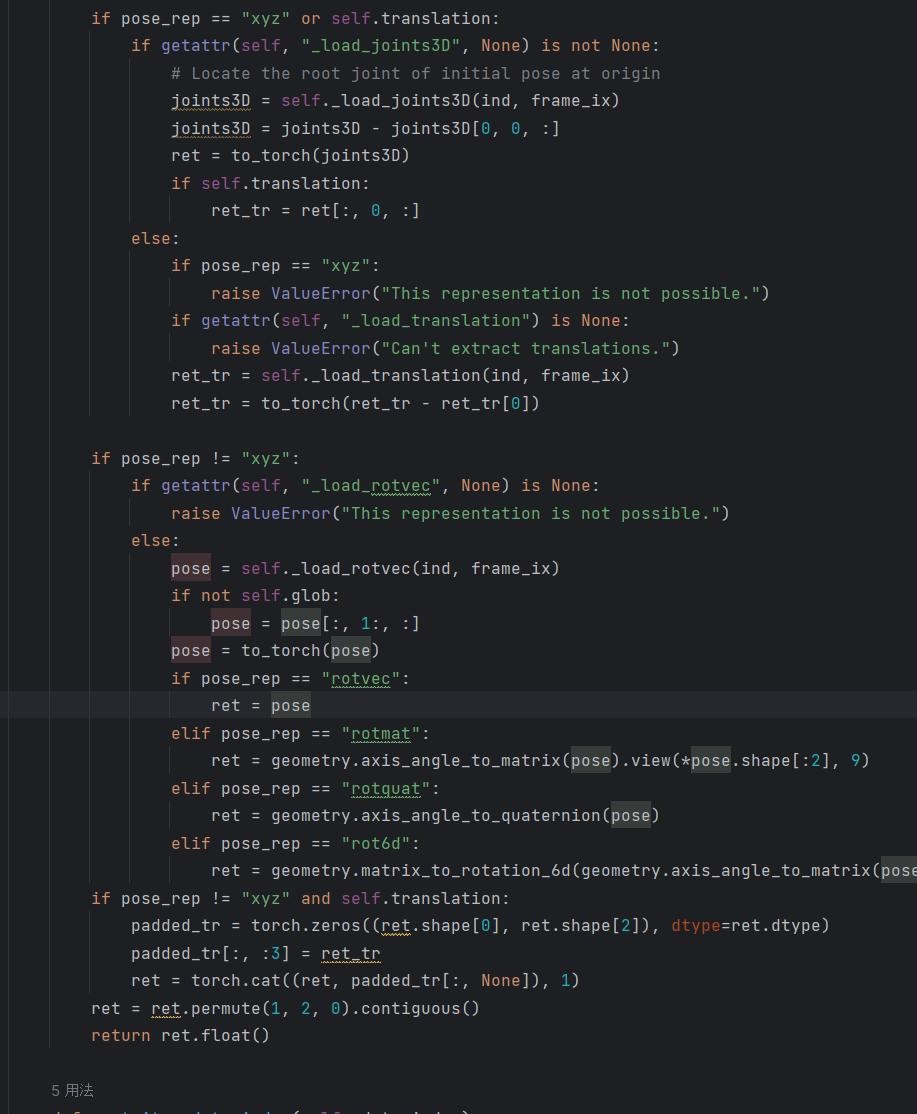
# Experiment On MotionClip

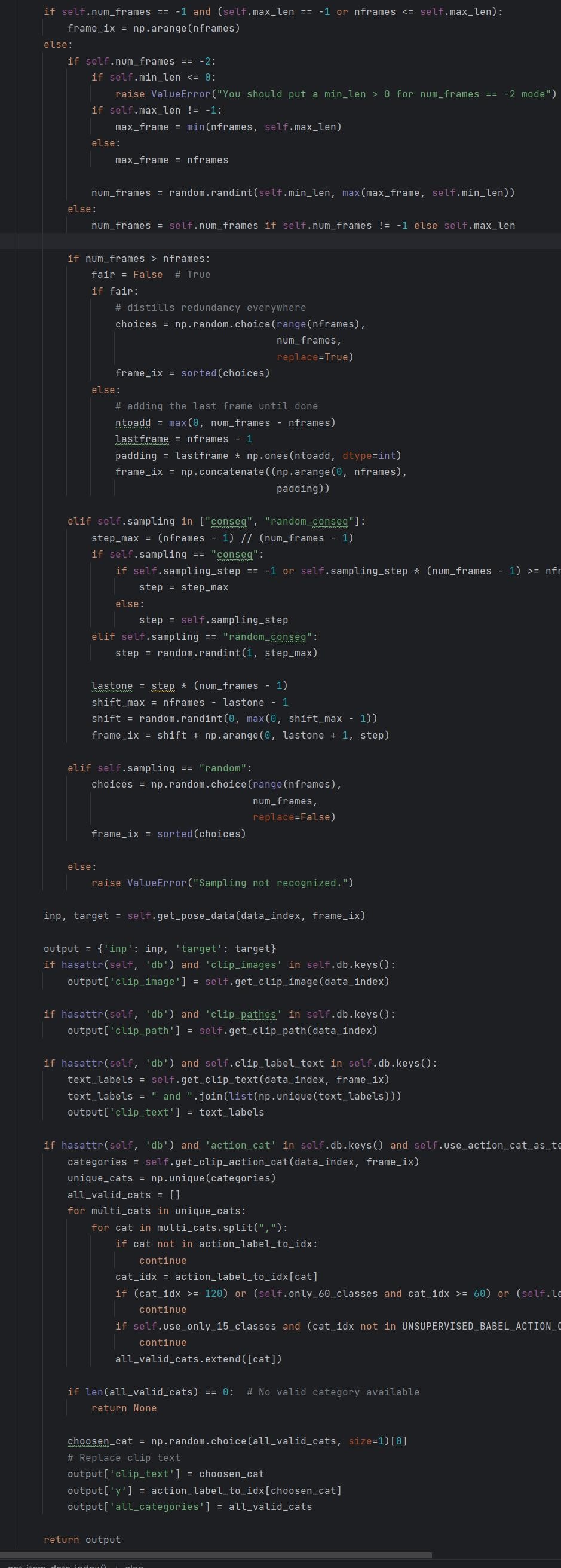
## Data process

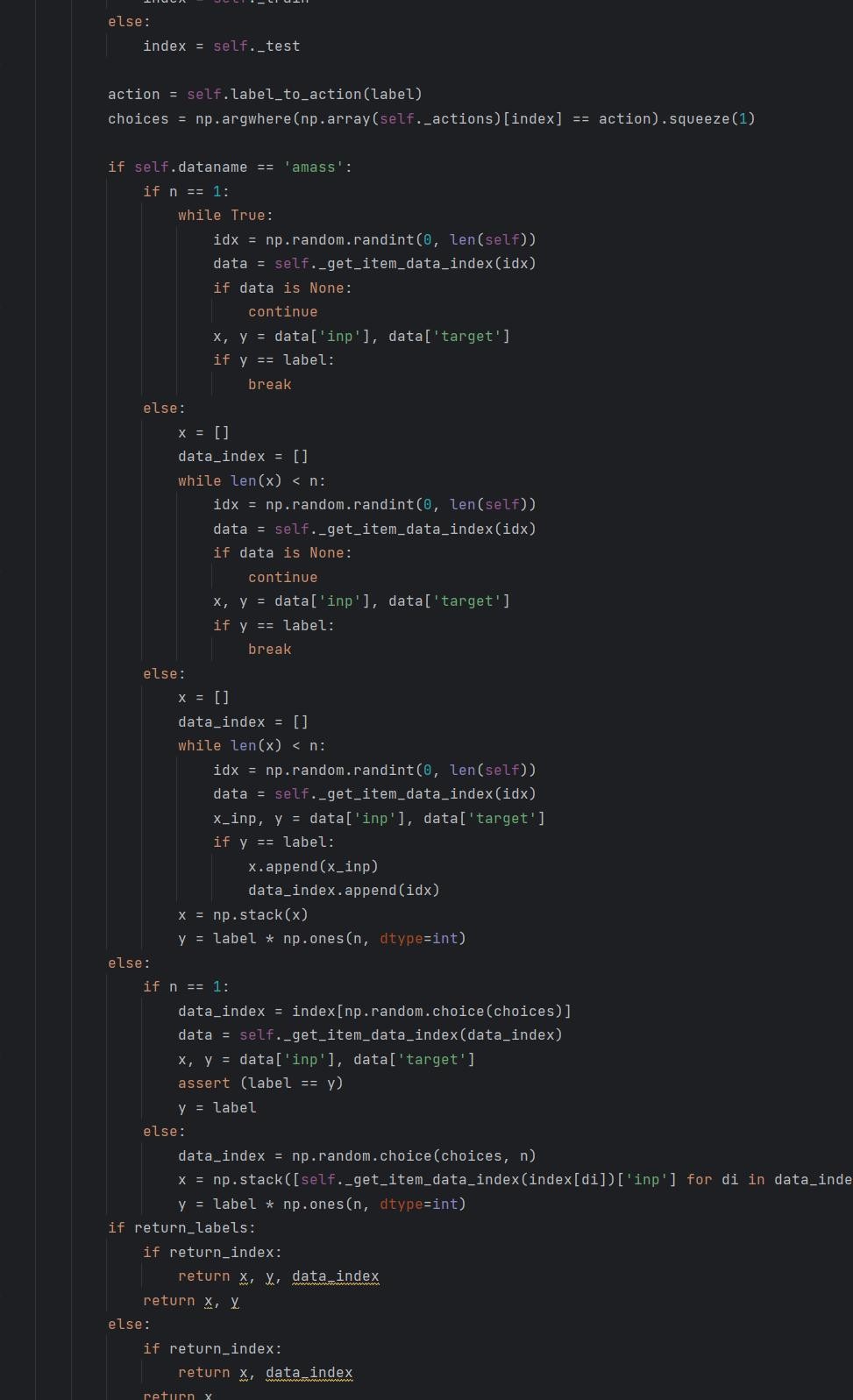


Normalization: Motion data from different clips are often captured under varying conditions. The first step involves normalizing the data to a common reference frame. This includes aligning the root joint position and orientation, ensuring all motion sequences are consistent.

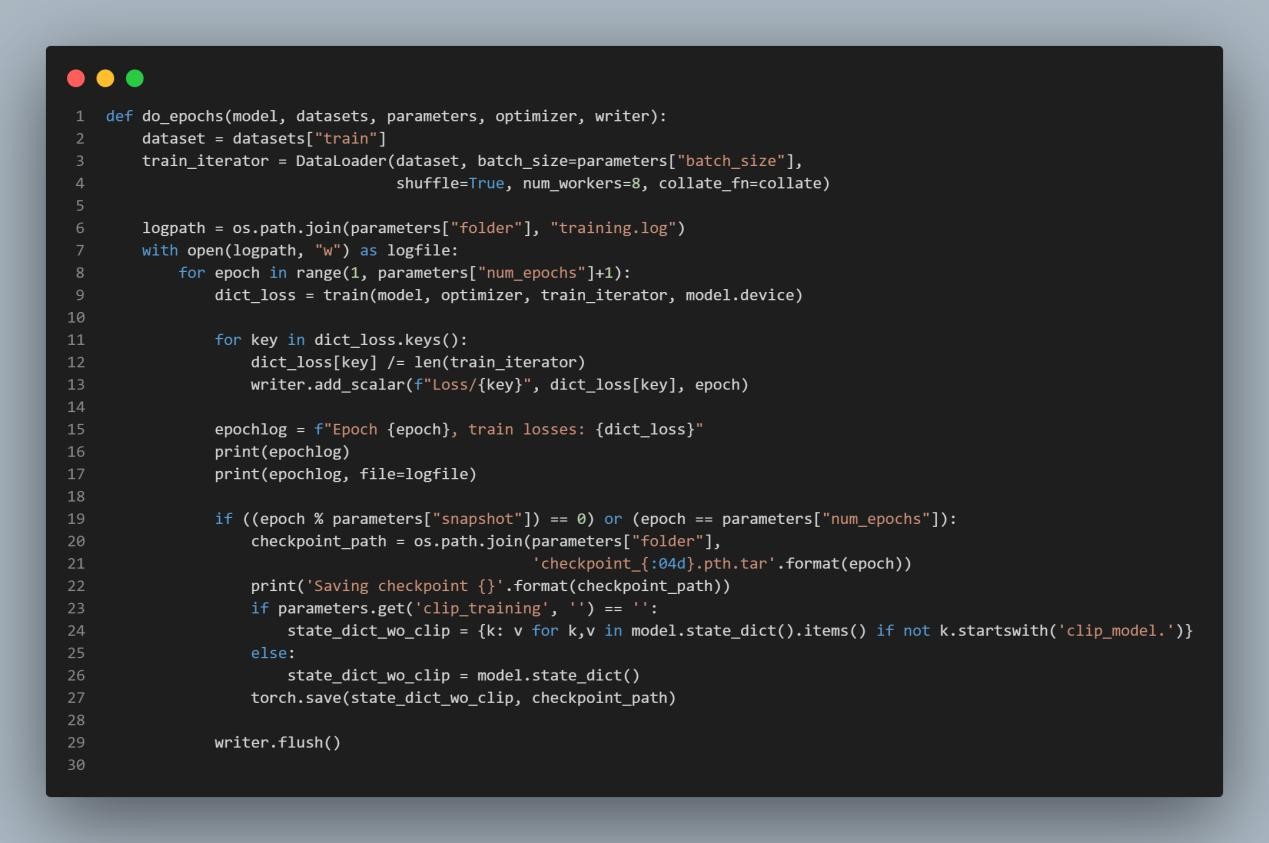
Filtering: To remove noise and ensure smooth motion transitions, the data is filtered using techniques such as low-pass filtering. This step helps in reducing jitter and artifacts that might have been introduced during the capture process.







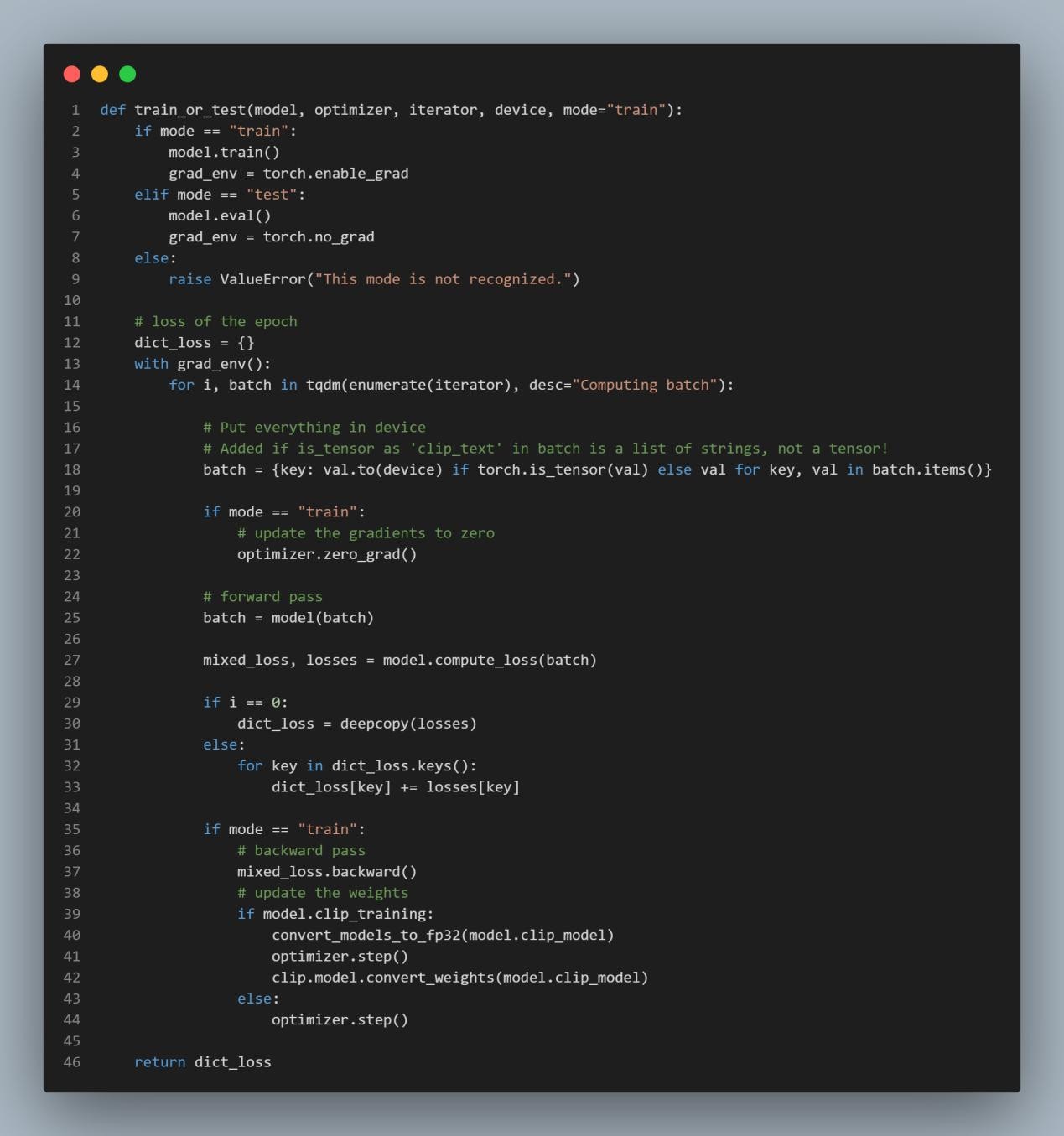
## Training

training a machine learning model using PyTorch. It involves setting up the training loop, managing the dataset, logging training progress with TensorBoard, and saving model checkpoints.

do\_epochs: This function performs the training loop. DataLoader: Creates an iterator for the training dataset. logpath: Path to the log file for recording training progress.

train: Trains the model for one epoch and returns the loss dictionary. writer.add\_scalar: Logs the loss to TensorBoard.

Checkpoint Saving: Saves the model's state at specified intervals. writer.flush: Ensures all pending events are written to disk.



This code showed above provides functions for training and testing a PyTorch model, particularly the CLIP model from OpenAI. It includes utilities to convert model parameters to 32-bit floating point, handles both training and evaluation modes, computes and aggregates losses, updates model parameters, and incorporates specific handling for CLIP's mixed-precision training. The main function train\_or\_test manages the training/testing loop, while train and test are wrappers that set the mode accordingly.

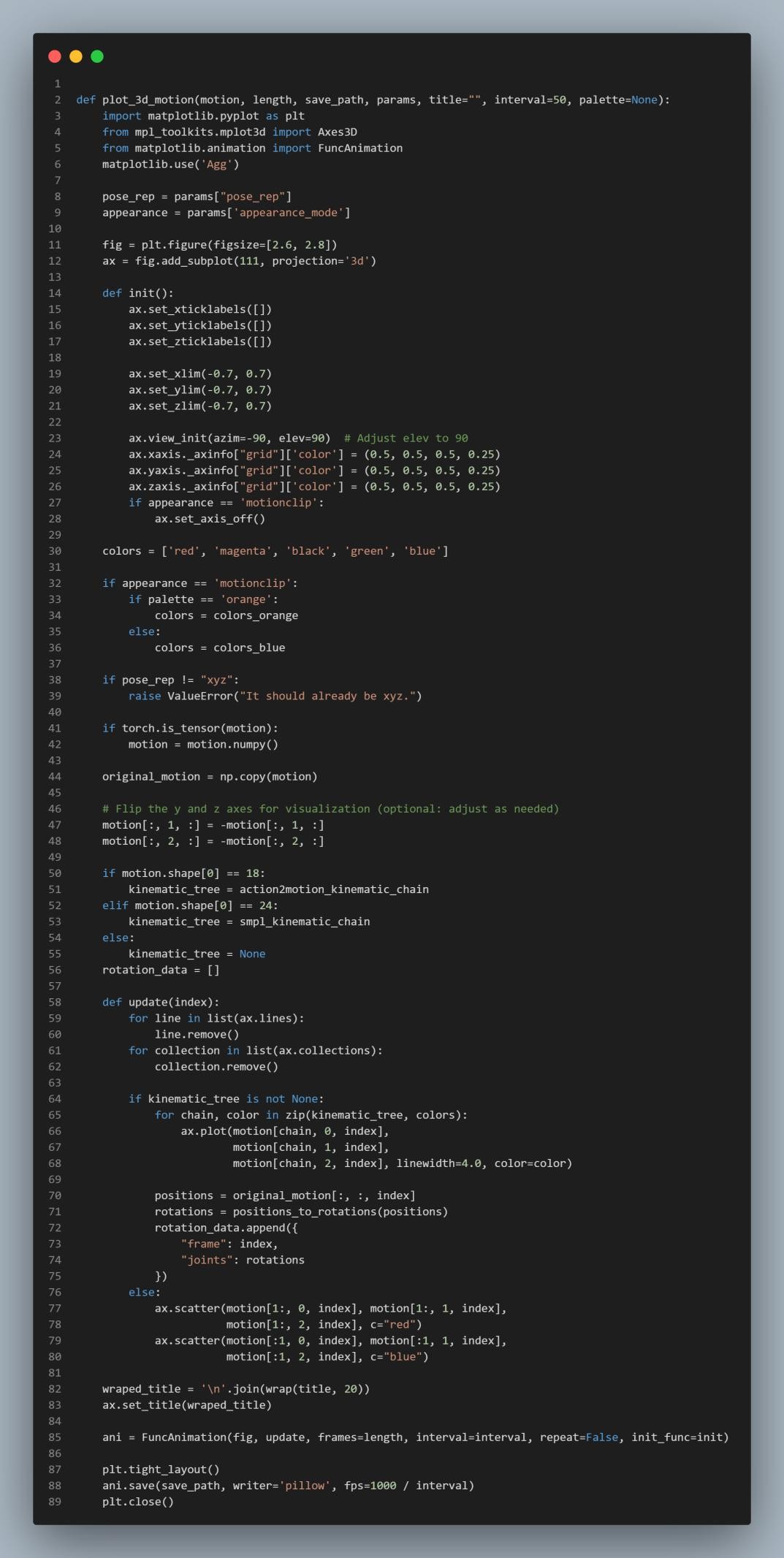
## Evaluation

We will evaluate the action sequence by drawing it into a gif, so below I will give the code for the gif：

This function generates video frames from motion capture data for visualization,

reconstruction, and generation. It handles different modes, uses multiprocessing for efficiency, and can include image overlays. The frames are saved as GIF files and can be stacked together for the final output.



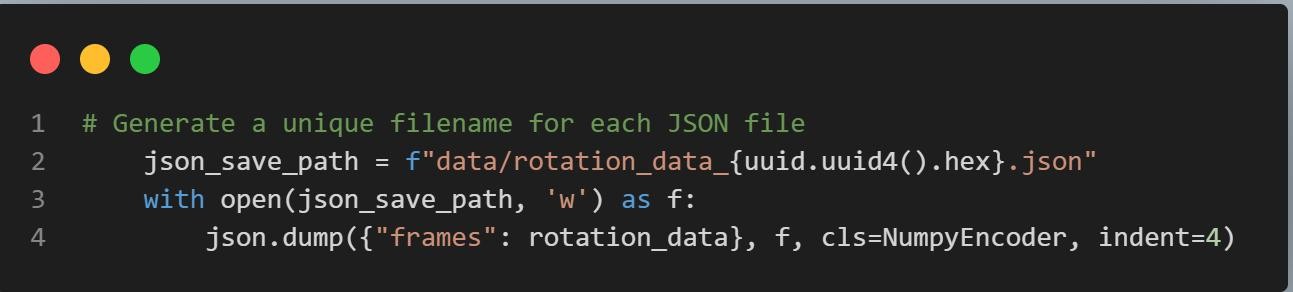


It handles different appearance modes and kinematic chains, processes motion data for each frame.

Position to rotation

Because what we have is the position coordinate information of each joint, but the position coordinate information can't be used by unity's smpl model, so we need to convert the position information of the joints to the rotation angle information, in order to achieve the conversion of the data we need to use a kinematic chain, so as to get the dependency relationship between the joints, and then use this to get the vector information of the two neighbouring joints, and then carry out the conversion to get the rotation angle information.

Write the rotation data into json file：

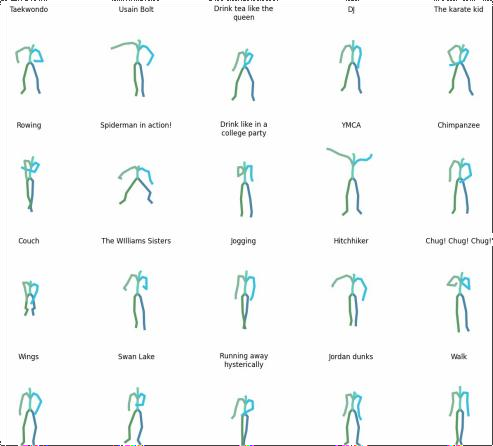




## Limitation

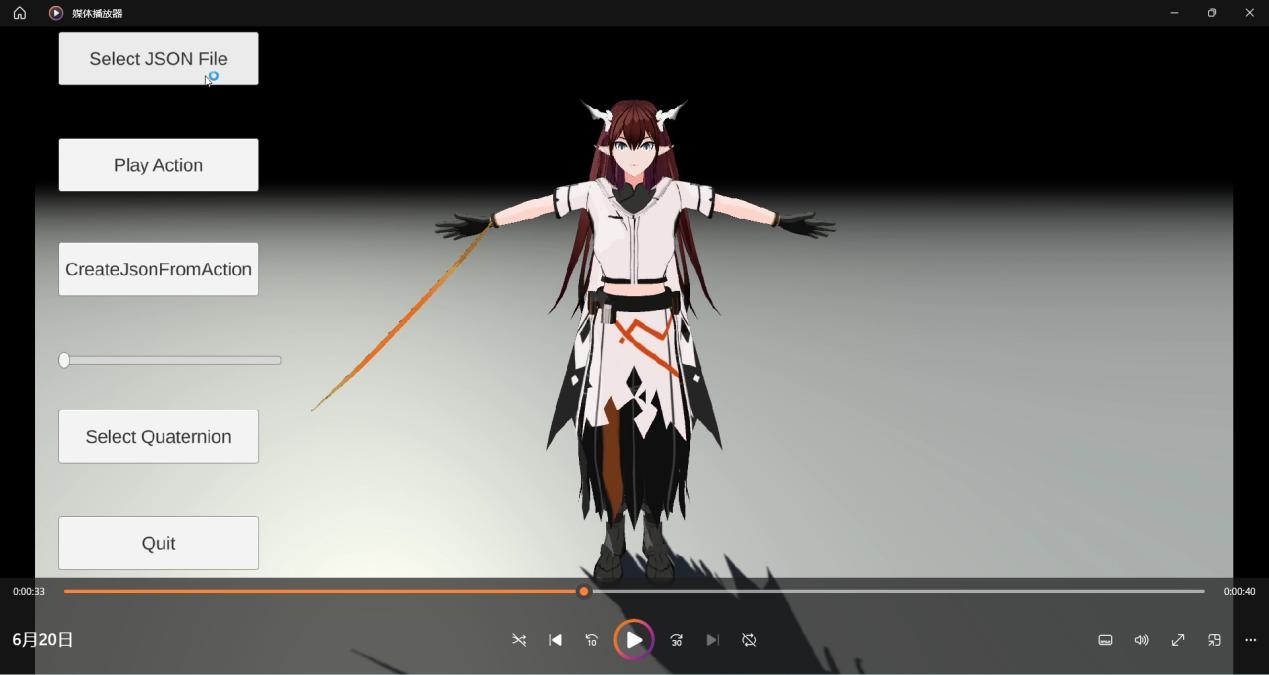
In the training process, when the number of training times increased, the number of traversal of the dataset increased from 100-epoch to 500-epoch, the output motion sequence is not moving, this may be because clip understands the association of images and text, and the multi-frame motion sequence associated with the text may result in the learning of the characteristics of the stationary images after training.

The static gif from 500-epoch：

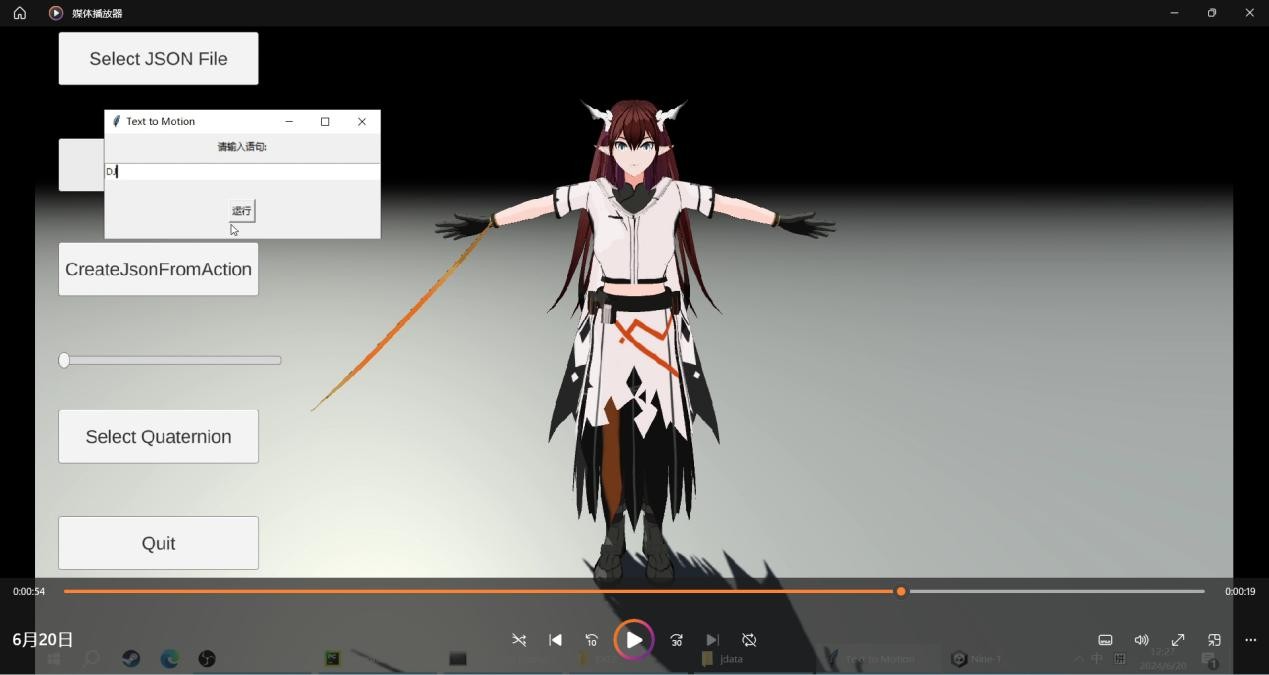


## Demonstration

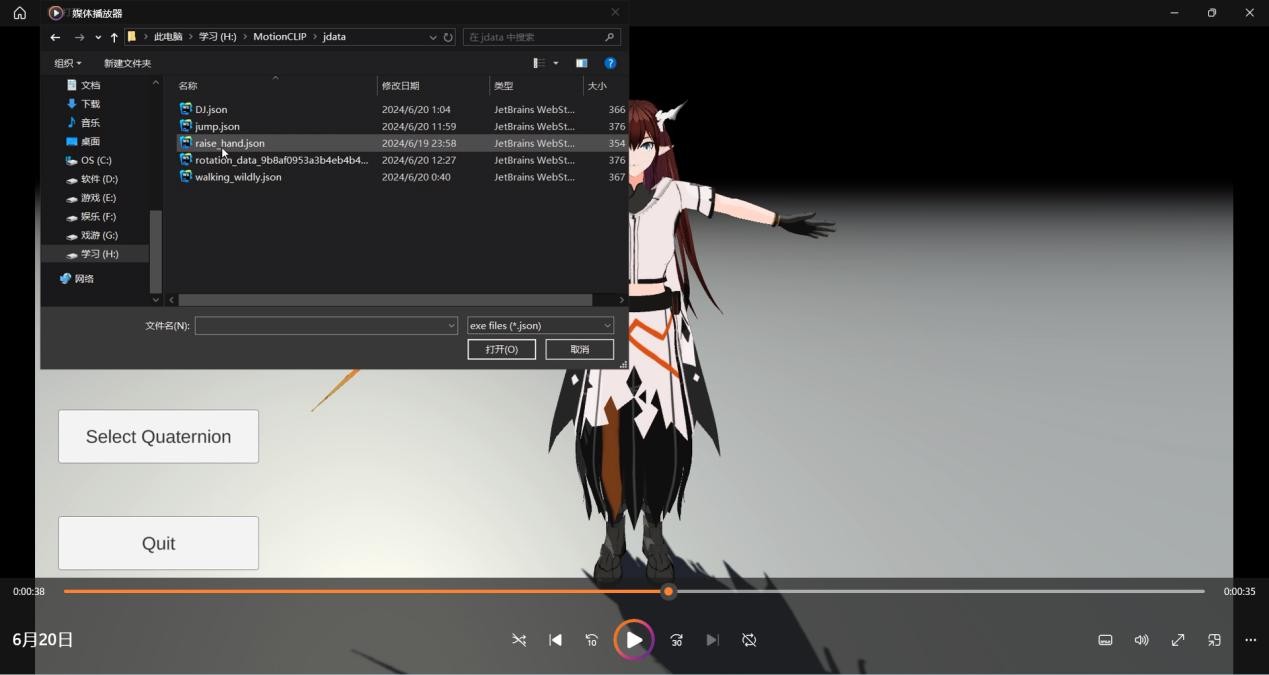
### Main Interface Display



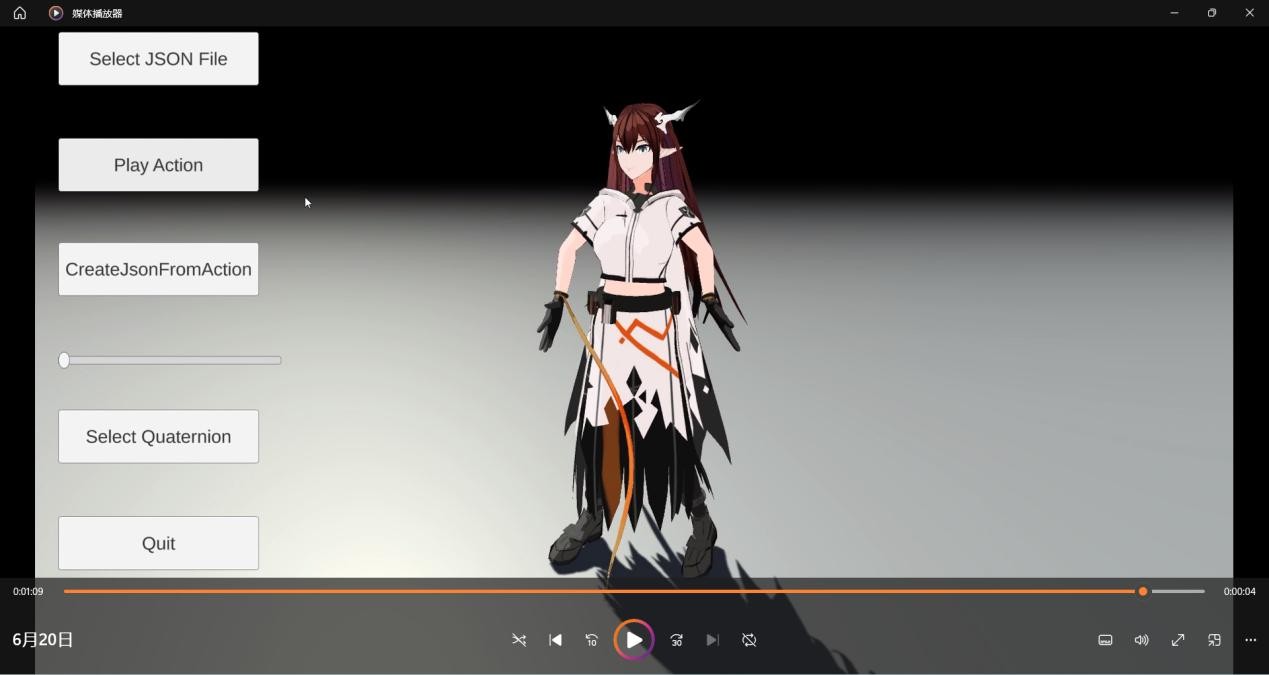
Enter text



Select json



Play



# Evaluation

**Actions.** We start by demonstrating the capabilities of MotionCLIP to generate explicit actions - both seen and unseen in training. We compare our model to JL2P. Since the two models were trained on different datasets, we define a new common ground for evaluation. We define two new sets of samples for a user study: (1)The

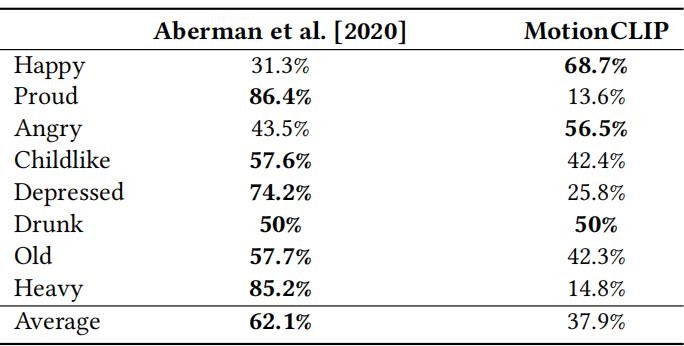
in-domain set comprises actions with textual labels that appear in at least 0.5% of the labels of both datasets, and (2) the Out-of-domain set includes textual labels that do not appear in any of the labels of both datasets, hence, unseen for both models. For fairness, we construct this set from the list of Olympic sports (both summer and winter) that are disjoint to both datasets. We conduct a user study, comparing the generation of each model conditioned on a given textual label. For each example, we then ask users to choose which of the two motions best fits the label. Table 1 shows that MotionCLIP was clearly preferred by the users for both sets. Figure 5 demonstrates a variety of sports performed by MotionCLIP, as used in the user-study. Note how even though this is not a curated list, the motion created according to all 30 depicted text prompts resembles the requested actions.

**Styles.** We investigate MotionCLIP’s ability to represent motion style, without being explicitly trained for it. We compare the results produced by MotionCLIP to the style transfer model by Aberman . The latter receives two input motion sequences, one indicating content and the other style, and combines them through a dedicated architecture, explicitly trained to disentangle style and

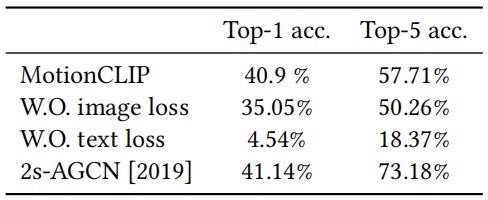
content from a single sequence. In contrast, we simply feed MotionCLIP with the action and style textual names. We show to users the outputs of the two models

side-by-side and ask them to choose which one presents both style and/or action better. Even though Aberman et al. was trained specifically for this task and gets the actual motions as an input, rather then text, Table 2 shows comparable results for the two models, with an expected favor toward Aberman et al.. This, of course, also means

that MotionCLIP allows expressing style with free text, and does

not require an exemplar motion to describe it. Such novel free text style augmentations are demonstrated in.

**Abstract language.** One of the most exciting capabilities of MotionCLIP is generating motion given text that doesn’t explicitly describe motion. This includes obvious linguistic connections, such as the act of sitting down, produced from the input text "couch". Other, more surprising examples include mimicking the signature moves of famous real and fictional figures, like Usain Bolt and The Karate Kid, and other cultural references like the famous ballet performance of Swan Lake and the YMCA dance. These results include motions definitely not seen during training, which strongly indicates how well the motion manifold is aligned to CLIP space.



# Application

Film and Animation Production:

Visual Effects and Animation: Text descriptions can be directly converted into actions for characters in movies, TV shows, and animations. For instance, a text describing "the protagonist sprinting forward" can be transformed into corresponding motion sequences for animation or special effects scenes.

Previsualization and Pre-production: In film production, screenwriters and directors can use text descriptions to preview actions of characters, aiding in scene design and action planning before filming.

Virtual Reality and Augmented Reality:

Virtual Character Control: In VR and AR applications, text input can control the real-time actions of virtual characters. For example, in training or educational applications, text describing how to perform specific actions can directly influence the behavior of virtual characters.

Interactive Experiences: Text-to-motion technology enhances interactivity in games and virtual experiences, enabling players to drive game character actions through verbal commands.

Education and Training:

Sports Training: Coaches can use text instructions to generate sequences of movements that athletes need to perform. This helps athletes understand and learn correct techniques and actions.

Medical Rehabilitation: In rehabilitation therapy, healthcare professionals can generate motion guides tailored to the recovery needs of patients based on text descriptions, facilitating effective rehabilitation training.

Artistic Creation and Performanc:

Dance and Performance: Dancers and performers can utilize text descriptions to generate motion sequences required for dances or performances. This can serve as part of the creative process or guidance during rehearsals.

Creative Expression: Artists and creators can explore and realize their creative ideas using text-to-motion technology, such as applying it in art installations or interactive art projects.

These examples demonstrate the wide-ranging applications of text-to-motion technology across various fields, enhancing efficiency and creativity in creation, training, and interactive experiences by translating verbal descriptions into executable actions.

# Strengths/Limitation and Discussion

MotionCLIP exhibits several notable strengths and limitations within the domain of motion generation:

### Strengths

Semantic Understanding:

MotionCLIP leverages the CLIP model's robust semantic understanding to generate motion sequences that closely align with high-level natural language descriptions. This capability allows for nuanced and contextually relevant motion generation based on textual inputs.

Decoupled and Continuous Embeddings:

The model utilizes decoupled latent embeddings that are well-aligned with desired behaviors, enabling effective motion editing, interpolation, and transfer tasks. This alignment ensures that semantically similar motions are proximal in latent space, facilitating smooth transitions and coherent outputs.

Versatility Across Tasks:

MotionCLIP demonstrates adaptability across a spectrum of motion-related tasks, including text-to-motion synthesis, style transfer, interpolation, editing, and recognition. This versatility underscores its applicability in diverse scenarios requiring precise and expressive motion generation capabilities.

### Limitations

Training Complexity:

Achieving optimal performance with MotionCLIP necessitates intricate alignment

with CLIP's latent space during training. This process is computationally intensive and demands substantial resources for pre-training and fine-tuning, limiting accessibility in resource-constrained environments.

Dependency on Visual Inputs:

To enhance alignment accuracy and generation quality, MotionCLIP relies significantly on synthetic visual inputs. This dependency introduces additional complexities in model training and requires extensive computational resources, potentially posing challenges in practical deployment scenarios.

In summary, while MotionCLIP excels in leveraging semantic understanding for sophisticated motion generation tasks, its effectiveness is tempered by considerable training complexities and dependencies on visual data inputs. These factors warrant careful consideration when deploying the model in real-world applications.

# Conclusion

We have introduced a motion generation network that harnesses the knowledge embedded within CLIP, enabling intuitive functionalities such as text-conditioned motion generation and editing. Our experiments demonstrate that training an auto-encoder solely on available motion data struggles to generalize effectively, potentially due to data quality issues or the inherent complexity of the domain. Nonetheless, by aligning this auto-encoder with a well-structured and information-rich latent space, we observe substantial improvements in understanding the motion manifold and its semantic representations.

It is intriguing to note that despite CLIP never having been exposed to motion data or temporal signals, its latent structure naturally induces semantics and disentanglement. This occurs despite the sparse and sometimes inaccurate textual annotations that connect CLIP's latent space with the motion manifold. Essentially, our alignment strategy facilitates semantic transfer by encouraging the encoder to place semantically similar motion samples closer together. Moreover, it fosters disentanglement within the CLIP space, as evidenced by our experiments in latent-space arithmetic.

This work underscores the powerful implications of integrating state-of-the-art vision-language models like CLIP into motion generation tasks. By leveraging CLIP's latent space, we enhance the interpretability and manipulation capabilities of motion representations, paving the way for more sophisticated and intuitive applications in graphics and animation.

Our motion data visualization plays a crucial role in comprehending, analyzing, and refining human motion generation models such as MotionCLIP and MotionDiffuse. Visualizing motion data allows researchers and developers to gain insights into the performance, accuracy, and versatility of these models through various analytical

techniques. These include examining motion sequences, comparing generated motions with ground truth data, and identifying patterns and anomalies.

# Reference

Kfir Aberman, Yijia Weng, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. 2020. Unpaired motion style transfer from video to animation. ACM Transactions on Graphics (TOG) 39, 4 (2020), 64–1.

Chaitanya Ahuja and Louis-Philippe Morency. 2019. Language2pose: Natural language grounded

pose forecasting. In 2019 International Conference on 3D Vision (3DV). IEEE, 719–728. Andreas Aristidou, Anastasios Yiannakidis, Kfir Aberman, Daniel Cohen-Or, Ariel Shamir, and Yiorgos Chrysanthou. 2021. Rhythm is a Dancer: Music-Driven Motion Synthesis with Global Structure. arXiv preprint arXiv:2111.12159 (2021).

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In International conference on machine learning. PMLR, 1597–1607.

Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai

Arulkumaran, and Murray Shanahan. 2016. Deep unsupervised clustering with gaussian mixture variational autoencoders. arXiv preprint arXiv:1611.02648 (2016).

Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516 (2014).

Pif Edwards, Chris Landreth, Eugene Fiume, and Karan Singh. 2016. JALI: an animatorcentric viseme model for expressive lip synchronization. ACM Transactions on graphics (TOG) 35, 4 (2016), 1–11.

Han Fang, Pengfei Xiong, Luhui Xu, and Yu Chen. 2021. CLIP2Video: Mastering Video-Text

Retrieval via Image CLIP. arXiv preprint arXiv:2106.11097 (2021).

Kevin Frans, LB Soros, and Olaf Witkowski. 2021. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. arXiv preprint arXiv:2106.14843 (2021).

Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. 2021. Stylegan-nada: Clip-guided domain adaptation of image generators. arXiv preprint arXiv:2108.00946 (2021).

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Chuan Guo, Xinxin Zuo, Sen Wang, Shihao Zou, Qingyao Sun, Annan Deng, Minglun Gong, and Li Cheng. 2020. Action2motion: Conditioned generation of 3d human motions. In Proceedings of the 28th ACM International Conference on Multimedia. 2021–2029.

Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel. 2021. Audioclip: Extending clip

to image, text and audio. arXiv preprint arXiv:2106.13043 (2021).

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), Vol. 2. IEEE, 1735–1742.

Xiangteng He and Yuxin Peng. 2017. Fine-grained image classification via combining vision and

language. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 5994–6002.

Daniel Holden, Jun Saito, and Taku Komura. 2016. A deep learning framework for character

motion synthesis and editing. ACM Transactions on Graphics (TOG) 35, 4 (2016), 1–11.

Xun Huang and Serge Belongie. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In Proceedings of the IEEE international conference on computer vision. 1501 – 1510.

Yanli Ji, Feixiang Xu, Yang Yang, Fumin Shen, Heng Tao Shen, and Wei-Shi Zheng. 2018. A large-scale RGB-D database for arbitrary-view human action recognition. In Proceedings of the 26th ACM international Conference on Multimedia. 1510–1518.

Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. 2021. AI Choreographer: Music

Conditioned 3D Dance Generation with AIST++. In The IEEE International Conference on Computer Vision (ICCV).

Angela S Lin, Lemeng Wu, Rodolfo Corona, Kevin Tai, Qixing Huang, and Raymond J Mooney. 2018. Generating animated videos of human activities from natural language descriptions. Learning 2018 (2018), 1.

Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C Kot. 2019. Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding. IEEE transactions on pattern analysis and machine intelligence 42, 10 (2019), 2684–2701.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black.

2015. SMPL: A skinned multi-person linear model. ACM transactions on graphics (TOG) 34, 6 (2015), 1–16.

Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. 2021.

CLIP4Clip: An Empirical Study of CLIP for End to End Video Clip Retrieval. arXiv preprint arXiv:2104.08860 (2021).

Shubh Maheshwari, Debtanu Gupta, and Ravi Kiran Sarvadevabhatla. 2022. MUGL: Large Scale Multi Person Conditional Action Generation with Locomotion. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 257–265.

Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black.

2019. AMASS: Archive of Motion Capture as Surface Shapes. In International Conference on Computer Vision. 5442–5451.

Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. 2021. Text2Mesh:

Text-Driven Neural Stylization for Meshes. arXiv preprint arXiv:2112.03221 (2021).

Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. 2021. Styleclip: Text-driven manipulation of stylegan imagery. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 2085–2094.

Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman,

Dimitrios Tzionas, and Michael J. Black. 2019. Expressive Body Capture: 3D Hands, Face, and Body from a Single Image. In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). 10975–10985.

Mathis Petrovich, Michael J. Black, and Gül Varol. 2021. Action-Conditioned 3D Human Motion

Synthesis with Transformer VAE. In International Conference on Computer Vision (ICCV). 10985

–10995.

Matthias Plappert, Christian Mandery, and Tamim Asfour. 2016. The KIT motion-language dataset. Big data 4, 4 (2016), 236–252.

Matthias Plappert, Christian Mandery, and Tamim Asfour. 2018. Learning a bidirectional mapping between human whole-body motion and natural language using deep recurrent neural networks. Robotics and Autonomous Systems 109 (2018), 13–26.

Abhinanda R. Punnakkal, Arjun Chandrasekaran, Nikos Athanasiou, Alejandra QuirosRamirez,

and Michael J. Black. 2021. BABEL: Bodies, Action and Behavior with English Labels. In Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR). 722–731.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal,

Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In International Conference on Machine Learning. PMLR, 8748–8763.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark

Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In International Conference on Machine Learning. PMLR, 8821–8831.

Aditya Sanghi, Hang Chu, Joseph G Lambourne, Ye Wang,

Chin-Yi Cheng, and Marco Fumero. 2021. Clip-forge: Towards zero-shot text-to-shape generation. arXiv preprint arXiv:2110.02624 (2021).

Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. 2019. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 12026–12035.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation

using deep conditional generative models. Advances in neural information processing systems 28 (2015).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems 30 (2017).

Yael Vinker, Ehsan Pajouheshgar, Jessica Y Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. 2022. CLIPasso: Semantically-Aware Object Sketching. arXiv preprint arXiv:2202.05822 (2022).

Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. 2021a. CLIPNeRF: Text-and-Image Driven Manipulation of Neural Radiance Fields. arXiv preprint arXiv:2112.05139

(2021).

Jiashun Wang, Huazhe Xu, Medhini Narasimhan, and Xiaolong Wang. 2021b. MultiPerson 3D Motion Prediction with Multi-Range Transformers. Advances in Neural Information Processing Systems 34 (2021).

Yu-Hui Wen, Zhipeng Yang, Hongbo Fu, Lin Gao, Yanan Sun, and Yong-Jin Liu. 2021. Autoregressive Stylized Motion Synthesis with Generative Flow. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 13612–13621.

Tatsuro Yamada, Hiroyuki Matsunaga, and Tetsuya Ogata. 2018. Paired recurrent autoencoders for

bidirectional translation between robot actions and linguistic descriptions. IEEE Robotics and Automation Letters 3, 4 (2018), 3441–3448.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2019. On the continuity of

rotation representations in neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 5745–5753.