Principles of traditional animation applied to motion capture data

Zhipeng Jia, BAI

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Augmented and Virtual Reality)

Supervisor: Rachel McDonnell

August 2022

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Zhipeng Jia

August 18, 2022

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Zhipeng Jia

August 18, 2022

Principles of traditional animation applied to motion capture data

Zhipeng Jia, Master of Science in Computer Science

University of Dublin, Trinity College, 2022

Supervisor: Rachel McDonnell

Motion capture animation creates high realistic movements for virtual characters. However, when applied to cartoon characters can appear "too realistic" and not stylized or expressive enough. For the convenience of animators to add expressiveness to cartoon characters, this project develops algorithms to alter motion capture data according to the traditional animation principles to add expressiveness.

Acknowledgments

First and foremost, in response to the excellent suggestions and feedback I received from my supervisor, Prof. Rachel McDonnell during the course of this dissertation, I would like to extend my sincere appreciation.

In addition, it is my pleasure to thank all of the authors cited in this paper for their contributions. The results of their fruitful research serve as a solid foundation for this study.

Finally, I sincerely thank my family and friends. Their care and support have made it possible for me to successfully finish my studies. .

Zhipeng Jia

University of Dublin, Trinity College
August 2022

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Presented in this section includes a motivation for developing a tool set that can be used to apply animation principles to motion capture data; the objectives of the development and evaluation process; as well as a description of the dissertation as a whole.

1.1 Motivation

Motion capture(mocap) is sampling and recording motion of humans, animals, and inanimate objects as 3D data and it is a very fast and accurate way to bring human motion into a 3D computer animation (Kitagawa and Windsor (2020)). Since 2001, motion capture technology has been used extensively to make nearly photo-realistic digital character models that simulate or approximate live-action cinema. There are a number of movies that use motion capture, including *Final Fantasy*, *Titanic*, and *Star Wars: Phantom Menace* (Izani et al. (2003)). In some cases, motion capture can replace traditional animation and for totally computer-generated characters, like s Gollum, The Mummy, King Kong, Davy Jones from *Pirates of the Caribbean*, the Na'vi from the film *Avatar* (Koloskova (2012)). The reason why motion capture is widely used benefits from its several advantages over traditional computer animation of a 3D model, including low latency, near real time, advanced movement and realistic physical interactions, as well as the extraordinarily massive number of animation data that can be produced within a given time (Suwetha et al. (2017)). However, after generating animation from motion capture, animators need to add the traditional animation techniques to increase its expressiveness. This could add a huge amount of works to the animators, some of which may be repetitive because the traditional animation principles have always been the same. Thus if there is a tool that can automatically apply animation principles to motion capture data, it can greatly reduce the workload of animators.

1.2 Objectives

This dissertation aims to develop algorithms, which can easily and flexibly apply animation principles to motion capture data, improve the expressiveness of animations. These algorithms must be able to allow animator to work with the results of motion capture without disrupting their artistic workflow or limiting their creative vision, integrating capture data seamlessly into the animation process (Cameron et al. (1997)). Additionally, There are already some plug-ins or add-ons which can achieve similar aim, therefore finding and learning from the excellent work of predecessors is one of the goals of the development process. Finally, these algorithms are integrated into a Python based plug-in on Blender, which provides convenience for animators in the process of using motion capture to make animation, and then improves work efficiency.

Python: Python is an interpreted programming language invented around 1991, which takes ideas from several programming paradigms such as procedural languages (like C), object oriented languages (like Java), functional languages (like Lisp, Haskell)functional languages (like Lisp, Haskell) and is used by many industries (Google, etc.) (Python (2021)).

Blender: Blender is a free Open Source 3D Creation Suite supporting the entire modeling and animation pipeline – modeling, rigging, animation, simulation, rendering, compositing and motion tracking. The program also includes Video Editing and Grease Pencil 2D Animation (Blain (2019)).

1.3 Outline

Throughout this report, a standard pattern is followed and it includes 5 more chapters which are State of Art, Design, Implementation, Evaluation, and Conclusions. Chapter2 details the introduction along with background research that was done both prior and during the implementation of the algorithms. Chapter3 focuses on the design method of the overall structure and the user interface. Chapter4 introduces detailed implementation of each algorithm. This section includes some Mathematics methods which has been used as well. Chapter5 discusses the results and any observations made during the course of this project. Chapter6 is the final chapter that concludes this work with details of contribution and any possible future work as an extension of this project.

Chapter 2

State of the Art

This chapter describes the background and closely-related project. It starts from an introduction to the motion capture and its wide usage in creating animation. Also, the well-known 12 traditional animation principles and related Add-ons are discussed in this section.

2.1 Background

2.1.1 Motion Capture Animation

Looking back at the development of motion capture technology, although the motion capture technology was born in the biological laboratory to serve physical therapy and rehabilitation, it has gradually become an indispensable part of the animation system. *Sinbad: Beyond the Veil of Mists* released in 2000 is supposed to be the first feature-length computer animation film based on motion capture technology. Only 6 years later, out of the three nominees for the 2006 Academy Award for Best Animated Feature, two of the nominees (*Monster House* and the winner *Happy Feet*) used motion capture. In the following years, motion capture technology has enabled more and more excellent

Figure 2.1: Left: Motion captured from human tap dancers on "Happy Feet"; Right: Motion capturing actors on "Monster House."

cartoon films to emerge. This is not only due to the rapid development of motion capture technology, but also because animators are gradually used to combining their creativity with motion capture.

2.1.2 Principle of Animation

Disney's 12 principles of animation were first introduced by Ollie Johnston and Frank Thomas in their book *The Illusion of Life: Disney Animation* (Thomas and Johnston (1995)). Some have called the book the "Bible of animation." Through examining the work of leading Disney animators from the 1930s onwards, Johnston and Thomas boil their approaches down to 12 basic principles of animation. The Figure 2.2 shows an overview of the principles.

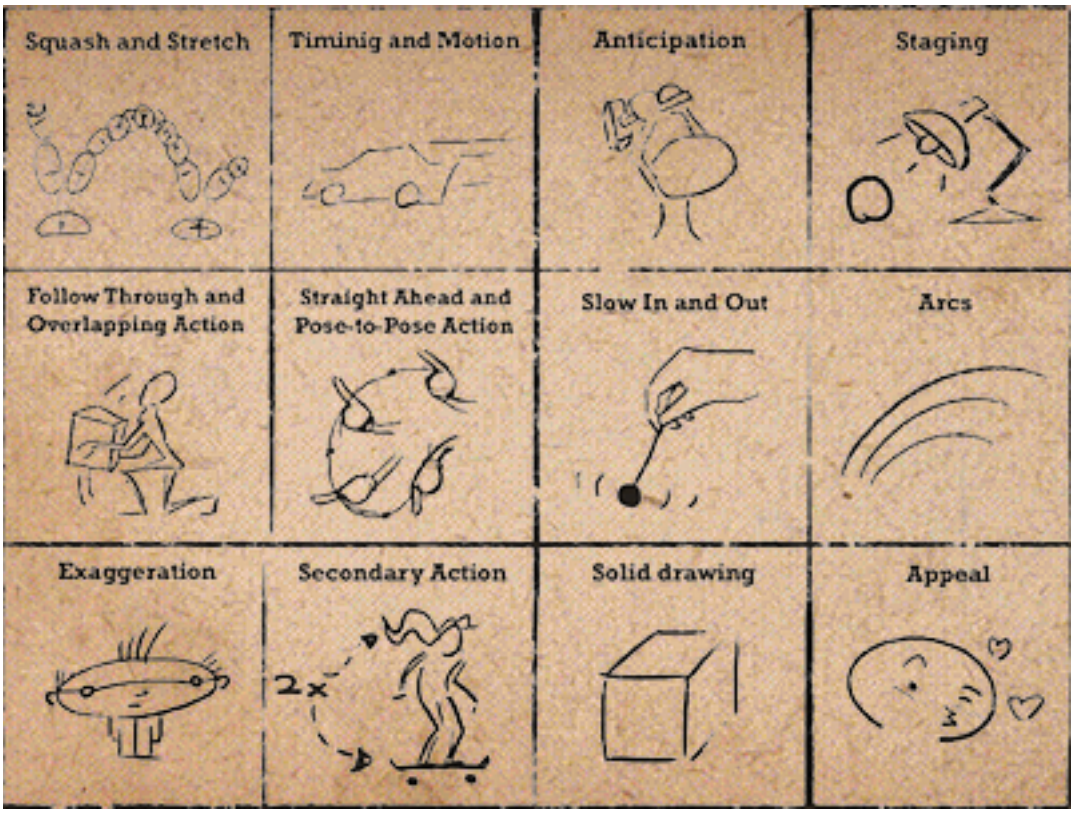

Figure 2.2: Overview of the 12 animation principles

Though originally intended for hand-drawn animation, the principles remain very relevant to modern computer animation as well. As an example, (Lasseter (1987)) demonstrates the importance of these principles to quality 3D computer animation. Obviously, with the continuous development of computer graphics and animation, new technologies such as inverse kinematics and motion capture, have changed the process of creating cartoon movements. However, these principles are still considered as the basis of animation in-

dustry and theoretical research. (Carter (2016)) is an instance for considering the remediation of the 12 principles of animation to fit the evolution of modern 3D CG animation. Then take a closer look at each animation principle and see how they make the animation better.

- *Squash and Stretch*
  - Defining the rigidity and mass of an object by distorting its shape during an action.
  - Consistent volume is necessary when using squash and stretch. In other words, stretching something means it gets thinner, and squashing something means it gets wider. Imagine a rubber ball bouncing in the air: the rubber ball stretches as it travels up and down, and squishes when it his the ground.
- *Anticipation*
  - The preparation for an action.
  - Anticipation helps the Viewers better prepare for what is going to happen. It creates a more realistic effect when applied to objects. Think about how it would look if you jumped in the air with no knee bend, or threw a ball without pulling your arm back first. The movement would appear very unnatural (It might even be impossible to jump without bending your knees!). You'll also come across as awkward, stale, and lifeless if you don't have a flicker of anticipation in your animation.
- *Staging*
  - Presenting an idea so that it is unmistakably clear.
  - To put it simply, you should keep the focus on the scene's most important aspects and minimize the motion of non-important things. For an instance, it is better in a master shot with the camera moving in.
- *Straight Ahead Action and Pose-To-Pose Action*
  - The two contrasting approaches to the creation of movement.
  - In many cases, the two approaches are combined since each has its own advantages. The first is known as *Straight Ahead Action* because the animation literally works frame-by-frame from start to finish. The second is called *Pose-To-Pose Action*. Here, the animator plans his start and end, relate them to each other in size and action, and gives the scene to his assistant to complete the interpolations.

- *Follow Through and Overlapping Action*
  - The termination of an action and establishing its relationship to the next action.
  - Different parts of an object will stop at different rates when it comes to a halt after being in motion. In the same way, not everything belonging to an object moves at the same speed. During a scene in which you have your character running across, their arms and legs should move at a different rate compared to their heads. This is supposed to be *Overlapping Action*. It is also likely that after running, their hair might keep moving for a few frames before stopping, which is a kind of *Follow Through*.
- *Slow In and Out*
  - The spacing of the in-between frames to achieve subtlety of timing and movement.
  - Consider how a car starts up and stops. First, it will move slowly, then gain momentum and accelerate and braking will cause the reverse effect.
- *Arcs*
  - The visual path of action for natural movement.
  - It's should be Kept in mind to follow the laws of physics when working in animation. Objects usually follow a arc path when they move, which should be reflected in animations. In the case of throwing a ball into the air, gravity acts upon it so that it follows a natural arc.
- *Secondary Action*
  - The action of an object resulting from another action.
  - In a scene, *Secondary Action* support or emphasize the main action. Think about things like subtle hair movement as the character walks, or maybe a facial expression or secondary object reacting to them. In any case, the *Secondary Action* shouldn't distract from the primary one.
- *Timing*
  - Spacing actions to define the weight and size of objects and the personality of characters.

  - It is the speed of an action, which can decide whether an audience will be able to comprehend the idea behind the action. For example, there a character with his head that turns left and right. Give different duration for the exact same head turn motions and the result is that it can be 'read' differently based merely on the timing. If the head turns back and forth really slow, it may seem as if the character is stretching his neck. A bit faster it can be seen as saying "no". Really fast, and the character is reacting to getting hit by something.

- *Exaggeration*
  - Accentuating the essence of an idea via the design and the action.
  - It is especially useful for animation, because motions that try to mimic reality can appear static and boring. The classical definition of exaggeration, employed by Disney, was to remain true to reality, just presenting it in a wilder, more extreme form.

- *Solid Drawing*
  - Taking into account forms in three-dimensional space, or giving them volume and weight.
  - The animator needs to be a skilled artist and has to understand the basics of three-dimensional shapes, anatomy, weight, balance, light and shadow, etc. Also, it's important to remain consistent. If your world has wonky doors and a warped perspective, keep that perspective throughout the entire animation. Otherwise, things will fall apart.

- *Appeal*
  - Creating a design or an action that the audience enjoys watching.
  - It's important to note that appeal doesn't necessarily mean good vs. evil. For example, in Disney's animated classic "Peter Pan", Captain Hook is an evil character, but most people would agree that his character and design has appeal.

2.2 Closely-Related Projects

2.2.1 Related Research

Different from the traditional animation, the amount of motion capture data is very large. It is very troublesome and time-consuming to manually modify the motion capture data

to adapt it to the animation principles. In order to solve this problem, some researchers focus on signal processing and propose the concept of animation filter. The following approaches are the most popular solutions that relate to the objectives of this project.

In (Wang et al. (2006)), the *"Cartoon Animation Filter"*, a simple filter that takes an arbitrary input motion signal and modulates it to make the output motion is more "alive" and "animated" is presented. The filter is defined by a equation which is the second derivative of the motion convolved with a Gaussian and then subtracted from the original motion signal.

$$x^*(t) = x(t) \otimes -LoG \tag{2.1}$$

$$\overline{x^{"}}(t) = x^{"}(t) \otimes Ae^{(-(((t/\sigma)\pm\Delta t)^2))} \tag{2.2}$$

where $x^{"}(t)$ is the second derivative of $x(t)$ with respect to time, and the amplitude, A, controls the strength of the filter. A second parameter, σ controls the Gaussian standard deviation, or width, of the smoothing kernel. Δt, can be used to create stretch and squash effects. On the whole, this is a simple, one parameter filter that can simultaneously add exaggeration, anticipation, follow-through, and squash and stretch to a wide variety of motions. In addition, a filter in dealing with single animation principle is also discussed in (White et al. (2006)) and (Kwon and Lee (2009)). The former focuses on the *Slow In and Out*, while the latter focuses on *Squash and Stretch*. However, for a animator, these filter give them less options to choose and alter, which may limit their creativity.

Besides, *Exaggeration* is also a popular direction that researchers often pay attention to. For example, (Kwon and Lee (2007)) have proposed a novel method to convert a captured-motion to a rubber-like exaggerated motion. The key idea of their method is to exaggerate a captured motion even though the resulting motion violates link-length constraints. And there are three limitations of their approach. First, the amount of data required for exaggerated motions is much greater than that required for the original motion. Second, as a result of the length changes in their method, they cannot ensure that the volume of the resulting mesh will be preserved. Finally, a penetration between each link of a character is not considered. (Savoye (2012)) is another research on *Exaggeration*. Their method incorporates motion, shape, and appearance from captured data to generate content-aware animations. In particular, their suggested approach explores two close tools that serve the common theme of Animation-Cartonization. The first consists of realizing articulated-based stretchable cartoon editing from marker-based mocap clips.

The second one generates a video-based character from surface performance capture. The problem of the method is that the current cartoon filter is incapable of achieving more photo-realistic exaggeration for a dynamic surface with an extremely large deformation.

2.2.2 Related Business Project

Computer Animation has become increasingly reliant on skeleton binding and skeleton animation with the arising and development of motion capture technology. In order to meet these demand, various 3D computer graphics applications like Autodesk Maya, 3D Max and Blender have involved such functions into their products. Also, many kinds of Plug–ins or Add-ons have been developed to support and improve the functions. Take the Add-ons used for Blender as an example. The Auto-Rig is a very popular Add–on for doing skeleton generating.

Recently, Blender Market have released a Add-on called GraphKit which could help to apply animation principles to the skeleton animations. The design of the GraphKit is all based on Graph Editor that allows users to adjust animation curves over time for any animatable property. F-Curves in Blender. GraphKit is a collection of 9 Graph Editor features that could greatly speed up animation making in Blender by focusing on editing multiple F-Curves simultaneously. These 9 features includes Amplify, Distribute, Fade, Flatten Holds, Offset, Perfect Loop, Randomize, Range to Loop and Repeat as what is showed in the Figure 2.3.

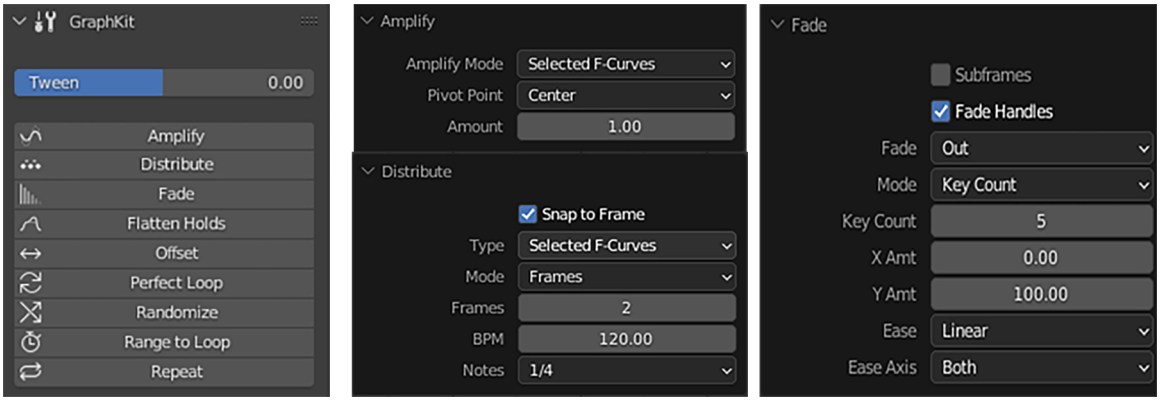

Figure 2.3: Left: main panel; Middle: Details of Amplify and Distribute; Right: Details of Fade

Four of these features can be considered to be aimed at applying animation principles. Firstly, the Amplify, a feature for adjusting the intensity of the animation. By adjusting the features, animation will be exaggerated. Second, the Fade, for fading in and out animations using standard ease functions. Third, the Distribute, a command for creating

even timing between keyframes. Last, the Offset, it allows animators to offset selected keyframes to achieve the principle of *Follow Through and Overlapping Action*.

2.3 Summary

From these studies and the latest development tools, we can see that the animation principles proposed decades ago are not outdated today, but have new applications and vitality. Developers are also committed to integrating more animation principles into development tools to contribute to animation production. This GraphtKit makes good use of these principles, but there are still some problems. This dissertation also aims to optimize and improve its first three features.

Chapter 3

Design & Implementation

The purpose of this chapter is to design algorithms that incorporate three animation principles into motion capture data. In addition, it also includes the overall structure of the Addon and the design and implementation of the graphical user interface. In the meanwhile, it also explains in detail how to draw on the experience of the Addon(Graphkit) mentioned in the previous chapter and modify it to fit the principles.

3.1 Structure

As shown in the Figure 3.1, the tool is divided into 3 parts in terms of function. The 3 parts correspond to the 3 animation principles(*Timing*, *Slow In and Out*, *Exaggeration*) separately. For each part, there is a parameter named Amount designed for controlling change scales. Additionally, there are other options and properties within each part prepared for users to meet different needs and scenarios while applying these 3 principles to the animation.

Both the *Timing* and the *Exaggeration* provide the user with two modes to select the target including the F-curves mode, selecting the entire curves to apply change and the Keyframes mode, selecting specific keyframes to apply change. The *Slow In and Out* only allows the user to select the entire F-curve as the target, but in this part, various mathematics functions are provided to achieve fading in, out and both. A total of 15 different kind of adjustments can be realized through the combination of 3 Types and 5 Functions, which could bring more flexibility and creativity to the animators. And in the *Exaggeration* part, except for the Mode, the exaggeration base can be changed by users as well. There are 4 bases including the Min, Max, First, Center which refer to the baseline of each point while doing the exaggeration.

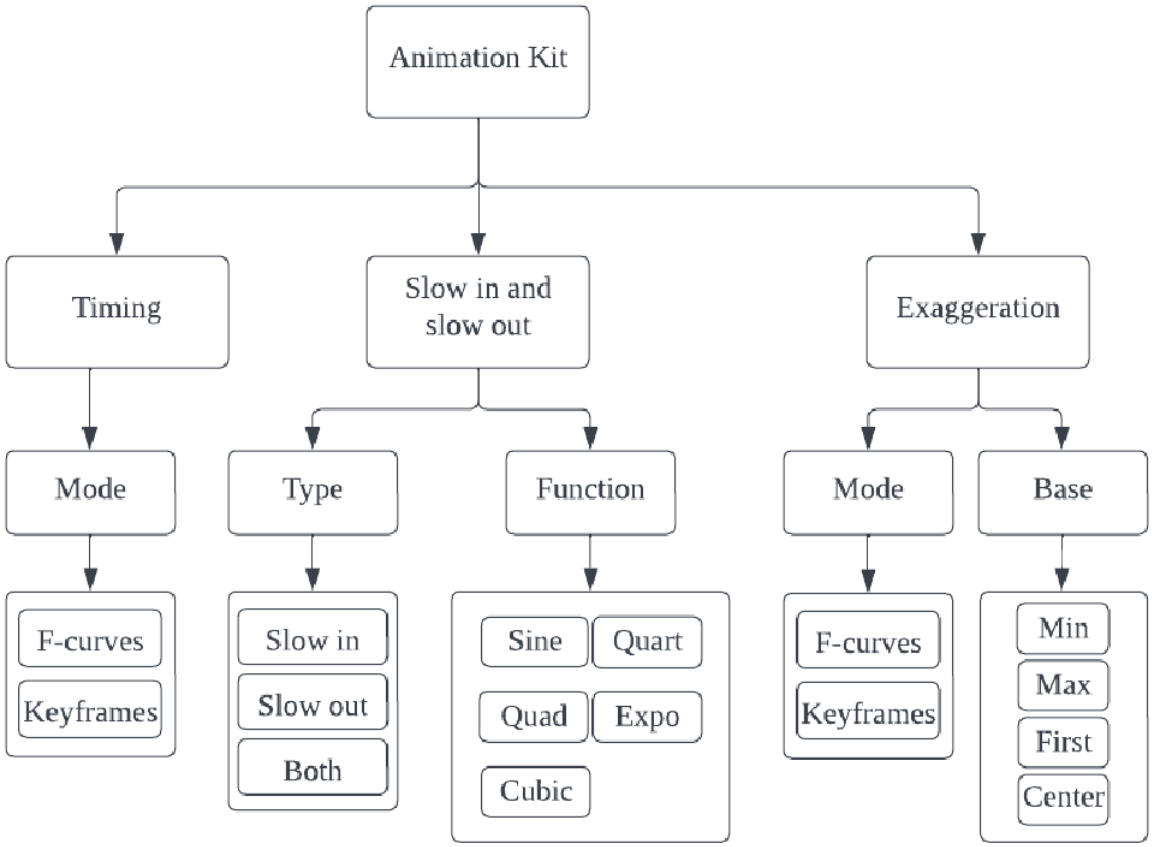

Figure 3.1: Structure of the Animation Kit

3.2 Use Interface & Algorithm Design

3.2.1 Timing

As introduced in previous chapter, different duration may have different means. As a result, the main idea to achieve the principle of *Timing* is to change the duration of target animation. Since there are 2 options in Mode, a enumeration property including F-curves and Keyframes is set in the Timing panel which is shown in the Figure 3.2. And then following it, there is a float property called amount used to determine how much to scale the selected duration.

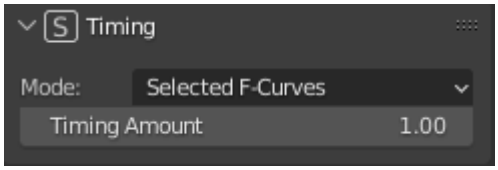

Figure 3.2: Timing UI of the Animation Kit

According to the different modes selected, the design of their corresponding procedures are also different. Then the whole algorithm can be described by 2 conditional branches as the Algorithm 1 shows. It iterates all selected key frames and then enters different branches according to the selected mode. If the mode is "keyframes", it will get into the first branch, calculate and accumulate the offset and then apply the offset to current keyframe and its handles. Otherwise, just multiply all the value of x coordinate by the scale amount. In addition, having at least 2 selected keyframes is an essential prerequisite for executing the algorithm and it is necessary to apply the same change to the handles of each keyframe to avoid unexpected deformation.

Algorithm 1 Algorithm for changing duration between selected keyframes

Require: $keyframes.n \geq 2$
Ensure: Apply the same transform to handles

```
1: procedure TIMING UPDATE(Mode, Amount, Fcurves)
2:     for all Fcurves do
3:         Sum ← 0
4:         for all Keyframes do
5:             if Mode is Keyframes then
6:                 if current keyframe is selected then
7:                     Dist ← k.co[0] − k.left[0]          ▷ This is the original distance
8:                     sDist ← Dist × Amount               ▷ This is the scaled distance
9:                     offset ← Dist − mDist
10:                    Sum ← Sum + offset
11:                end if
12:                k.left[0] ← k.left[0] − Sum
13:                k.right[0] ← k.right[0] − Sum
14:                k.co[0] ← k.co[0] − Sum
15:            else
16:                k.left[0] ← k.left[0] ∗ Amount
17:                k.right[0] ← k.right[0] ∗ Amount
18:                k.co[0] ← k.co[0] ∗ Amount
19:            end if
20:        end for
21:    end for
22: end procedure
```

In the Figure 3.3, there is an graphic explanation of the Algorithm 1. ① shows the original condition and ② represents making a change on the whole F-curve which is achieved by multiply operation only. However, if use the same method to selected keyframes, the result will be like the ③ that the points following the target cannot move together with the target point. And the ④ is the key idea of the solution which can accumulate the change and apply it to its next points.

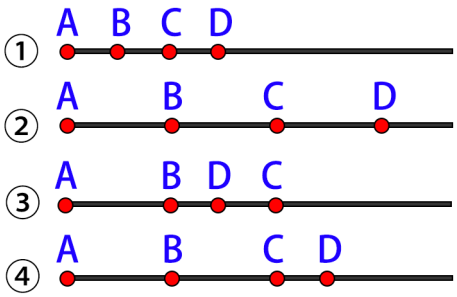

Figure 3.3: Graphic Explanation of Algorithm1

3.2.2 Slow In and Out

The panel for *Slow In and Out* is designed as what has been shown in the Figure 3.4. The first enumeration property is the fading type including in, out and both. The second enumeration property called function is used for choosing the fit curve, which includes Sine, Quad, Cubic, Quart and Expo. Then the float property named Key numbers is used to let animators decide how many keyframes will be applied to the change. The last one, Slowness Amount is responsible for the fading speed.

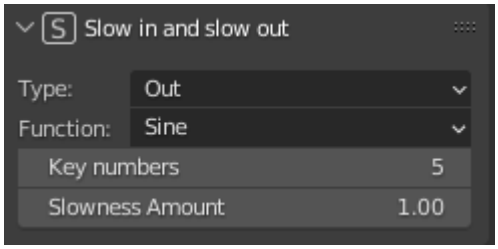

Figure 3.4: Slow In and Out UI of the Animation Kit

The algorithm is consists 3 main parts, fitting curve selection, applying the change to original F-curve as well as the fitting curve function. Different from the algorithm introduced in Related Research, which is based on Laplacian of the Gaussian, this algorithm adapt a group of concave, convex and sigmoid functions got from Graphkit which can be described as equation (3.1) $\sim$ (3.15) to fit F-curve to make it trend to slow in, slow out or both side.

$$F_{\textbf{Concave\&Sine}}(n) = -\cos\left(\frac{\pi}{2}n\right) + 1 \tag{3.1}$$

$$F_{\textbf{Convex\&Sine}}(n) = \sin\left(\frac{\pi}{2}n\right) \tag{3.2}$$

$$F_{\textbf{Sigmoid\&Sine}}(n) = -\frac{1}{2}(cos(\pi n) - 1) \tag{3.3}$$

$$F_{\textbf{Concave\&Quad}}(n) = n^2 \tag{3.4}$$

$$F_{\textbf{Convex\&Quad}}(n) = -n(n-2) \tag{3.5}$$

$$F_{\textbf{Sigmoid\&Quad}}(n) = \begin{cases} 2n^2 & n < \frac{1}{2} \\ -\frac{1}{2}(m(m-2)-1) & n \geq \frac{1}{2}(m = 2n-1) \end{cases} \tag{3.6}$$

$$F_{\textbf{Concave\&Cubic}}(n) = n^3 \tag{3.7}$$

$$F_{\textbf{Convex\&Cubic}}(n) = (n-1)^3 + 1 \tag{3.8}$$

$$F_{\textbf{Sigmoid\&Cubic}}(n) = \begin{cases} \frac{1}{2}m^3 & m < 1(m = 2n) \\ \frac{1}{2}((m-2)^3 + 2) & m \geq 1(m = 2n) \end{cases} \tag{3.9}$$

$$F_{\textbf{Concave\&Quart}}(n) = n^4 \tag{3.10}$$

$$F_{\textbf{Convex\&Quart}}(n) = -(n-1)^4 + 1 \tag{3.11}$$

$$F_{\textbf{Sigmoid\&Quart}}(n) = \begin{cases} \frac{1}{2}m^4 & m < 1(m = 2n) \\ -\frac{1}{2}(m-2)^4 + 1 & m \geq 1(m = 2n) \end{cases} \tag{3.12}$$

$$F_{\textbf{Concave\&Expo}}(n) = \begin{cases} 0 & n = 0 \\ 2^{10(n-1)} & n \neq 0 \end{cases} \tag{3.13}$$

$$F_{\textbf{Convex\&Expo}}(n) = \begin{cases} 1 & n = 1 \\ -2^{-10n} + 1 & n \neq 1 \end{cases} \tag{3.14}$$

$$F_{\textbf{Sigmoid\&Expo}}(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ \frac{1}{2} \cdot 2^{10(m-1)} & n \neq 1 \& n \neq 0 \& m < 1(m = 2n) \\ \frac{1}{2} \cdot -2^{-10m} + 1 & n \neq 1 \& n \neq 0 \& m \geq 1(m = 2n) \end{cases} \tag{3.15}$$

As seen in the Figure 3.5, there are the Function image of the equation above. If the x-axis is regarded as time and the y-axis is regarded as distance, it is not difficult to observe that the object moving according to the convex function is decelerating; the object moving according to concave function is accelerating; The object moving according to the sigmoid function is to accelerate first and then decelerate. Therefore, Slow Out can be simulated by the convex function, Slow In can be simulated by the concave function, and Slow In and Out can be simulated by the sigmoid function.

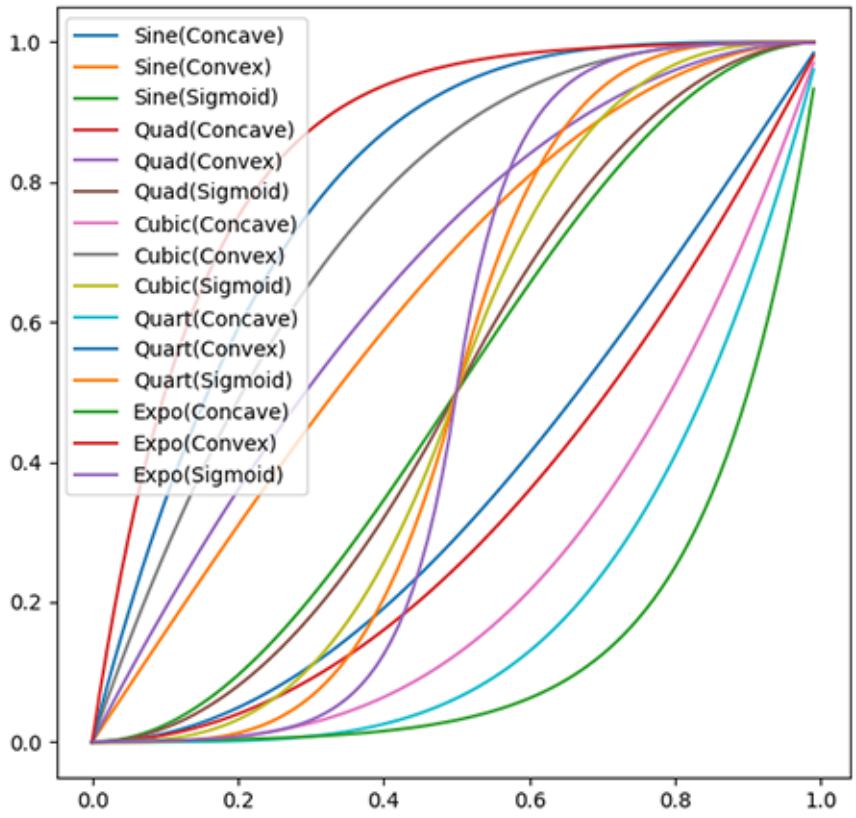

Figure 3.5: Image of Function (3.1) $\sim$ (3.15)

However, taking these into consideration alone isn't enough. The Algorithm 2 is designed to superimpose the change trend of these curves relative to the uniform motion on the original motion curve to achieve the goal. The aim of procedure "GetSlowPerc" is to get the difference between the Slow In/Out motion and the uniform motion and the procedure "SlowInOutUpdate" is responsible for apply the difference to the original curve according to the percentage of current keyframe. As for the procedure "getLinearY" and "getDist", since they are commonly used to determine a line and obtain the distance from two points, they will not be discussed again here.

Algorithm 2 Algorithm for superimposing the Slow In and Out

Require: $keyframes.n \geq 2$
Ensure: Apply the same transform to handles

procedure GETSLOWPERC($function, type, pec$)
  if type is "IN" **then**
    if function is "SINE" **then** return SineConcave(perc) - Linear(perc)
    end if
    if function is "QUAD" **then** return QuadConcave(perc) - Linear(perc)
    end if
    if function is "CUBIC" **then** return CubicConcave(perc) - Linear(perc)
    end if
    if function is "QUART" **then** return QuartConcave(perc) - Linear(perc)
    end if
    if function is "EXPO" **then** return ExpoConcave(perc) - Linear(perc)
    end if
  end if
  if type is "OUT" **then**
    ... ▷ Similar to "IN", just change Concave to Convex
  end if
  if type is "BOTH" **then**
    ... ▷ Similar to "IN", just change Concave to Sigmoid
  end if
end procedure
procedure SLOWINOUT UPDATE($function, type, keyframes, Amount$)
  $x1 \leftarrow keyframes[0].x$ $x2 \leftarrow keyframes[-1].x$
  $y1 \leftarrow keyframes[0].y$ $y2 \leftarrow keyframes[-1].x$
  $max \leftarrow max(keyframes.y)$ $min \leftarrow min(keyframes.y)$
  for all Keyframes **do**
    $x \leftarrow keyframe.x$ $y \leftarrow keyframe.y$
    $lHandleOffset \leftarrow keyframe.y - keyframe.lHandle.y$
    $rHandleOffset \leftarrow keyframe.y - keyframe.rHandle.y$
    $yTarget \leftarrow getLinearY(x1, x2, y1, y2, x)$
    $dist \leftarrow getDist(x1, x2, y1, y2)$
    $yDist \leftarrow getDist(x1, y1, x, yTarget)$
    $perc \leftarrow yDist/dist$
    $slowness \leftarrow getSlowPerc(function, type, perc) \times Ammount$
    $y \leftarrow y - slowness$
    if y $>$ max **then** $y \leftarrow max$
    end if
    if y $<$ min **then** $y \leftarrow min$
    end if
    $keyframe.y \leftarrow y$
    $keyframe.lHandle.y \leftarrow y - lHandleOffset$
    $keyframe.rHandle.y \leftarrow y - rHandleOffset$
  end for
end procedure

3.2.3 Exaggeration

Similar to the *Timing*, there is a enumeration property named "Mode", which includes F-curves and Keyframes options in *Exaggeration* panel as well. Also, in Figure 3.6, a enumeration property named "Base" is used to set the baseline while executing the exaggeration. As mentioned in Structure section, there are 4 options(Center, First, Max and Min) in the list for animators to choose. And at the bottom, there is a float property called amount used to determine how much to exaggerate the keyframes or F-curves.

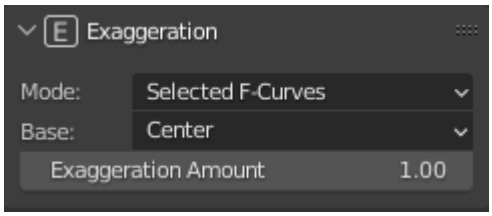

Figure 3.6: Exaggeration UI of the Animation Kit

Algorithm 3 Algorithm for Exaggeration

Require: $keyframes.n \geq 1$
Ensure: Apply the same transform to handles

procedure GETBASE($base, Keyframes$)
  if base is "CENTER" **then** return (max(Keyframes.y) + min(Keyframes.y))/2
  end if
  if base is "FIRST" **then** return Keyframes[0].y
  end if
  if base is "MAX" **then** return max(Keyframes.y)
  end if
  if base is "MIN" **then** return min(Keyframes.y)
  end if
end procedure
procedure EXAGGERATE UPDATE($base, keyframes, Amount$)
  for all Fcurves **do**
    $Base \leftarrow getBase(base, Fcurve.Keyframes)$
    for all Keyframes **do**
      $kDist \leftarrow Base - Keyframe.y)$
      $lDist \leftarrow Base - Keyframe.lHandle.y)$
      $rDist \leftarrow Base - Keyframe.rHandle.y)$
      $kKeyframe.y \leftarrow Base - kDist * Amount$
      $Keyframe.lHandle.y \leftarrow Base - lDist * Amount$
      $Keyframe.rHandle.y \leftarrow Base - rDist * Amount$
    end for
  end for
end procedure

The Algorithm 3 has 3 main steps. First of all, it decides the baseline according to the base selected from the enumeration property.Then it calculates the distance between the baseline and each Keyframe. At the end, it multiplies the distance by Amount and updates the Keyframes as well as its handles by the new values which are the result that subtract scaled distance from baseline. From the Figure 3.7, we can observe that selecting different base will have different effects on the results under the same exaggeration times. For example, if the lowest point of the curve is taken as the baseline, the whole curve will be enlarged to the positive direction with this as the boundary; Conversely, if the highest point of the curve is the baseline, the whole curve will be enlarged to the negative direction.

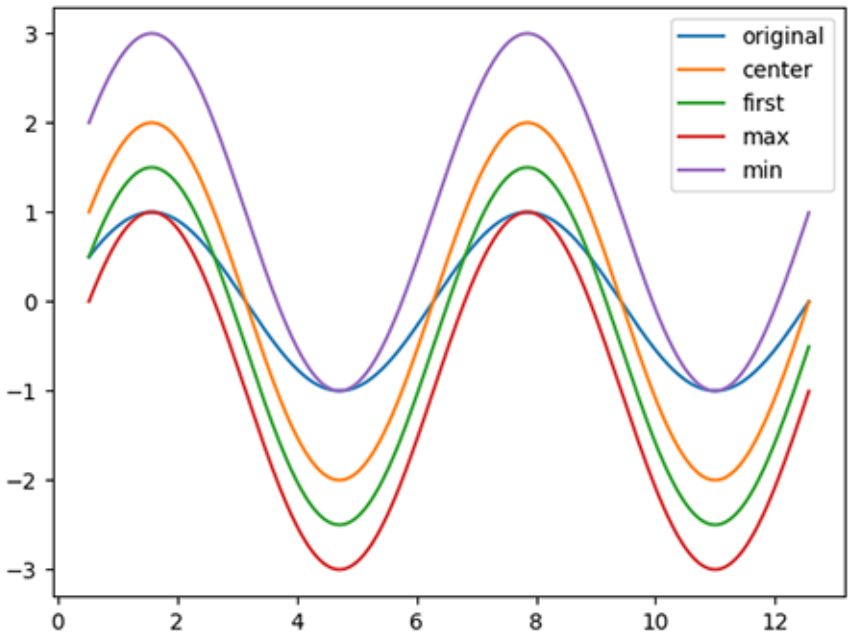

Figure 3.7: Simulation of exaggeration while choosing different base

Chapter 4

Evaluation

This chapter discusses the experiments that is done to test the 3 main algorithms and the relative utils adopted by the Add-on. The basic method of the experiment is to use the Add-on to apply animation principles to the animation of some simple 3D object first and when the expect results show up, import the 3D models with motion capture data and check if the Add-on works as expectation on dealing with motion capture data.

The specification of computer and software used for the experiment are:

Operation System: Windows 11

CPU: Intel® Core™ i9-11900F @ 2.50GHz

RAM: 32.0 GB

GPU: NVIDIA GeForce RTX 3060

Blender: Ver.3.1.2

Unreal Engine: Ver.4.2.7

4.1 Experiments

In this section, a cube will be used to demonstrate the functions. In addition, this section will also analyze the visual impact on the animation and the changes of related parameters after applying the three animation principles using the Add-on. The experiment takes the uniform linear motion as shown in the Figure 4.1 as the initial state and the corresponding F-curve is shown as Figure 4.2. Then through comparing the executing results with the initial state, we can judge whether the Add-on can correctly complete the expected functions according to the animation principles.

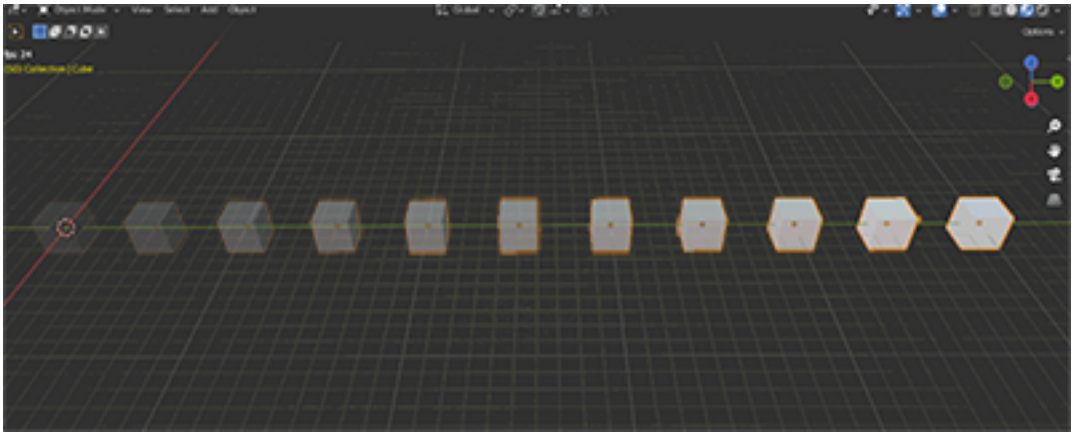

Figure 4.1: Dynamic picture of initial uniform linear motions

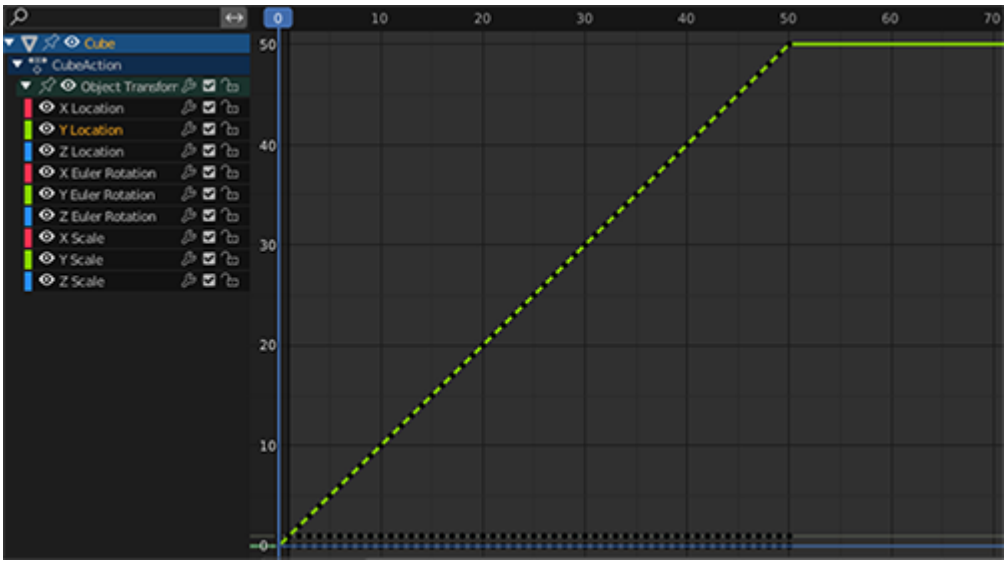

Figure 4.2: F-curve of initial uniform linear motions

The Table 4.1 describes the cases that will be tested. Case 1 is the test for *Timing* and aims to check if it works correctly when several keyframes in the middle is selected. Case 2 is the test for *Slow In and Out* and aims to check if the algorithm can select the corresponding function according to the options and if the proper change can be applied to the F-curve and the motion. Case 3 is the test for *Exaggeration* and aims to check if the exaggeration can be done in terms of the Base selected.

No.	**Function**	**Scenario**
1	*Timing*	Mode: keyframes
2	*Slow In and Out*	Type: Both; function: Quartic
3	*Exaggeration*	Mode: F-curve; Base: Min

Table 4.1: The experimental cases discussed in this section

In the Case 1, 5 keyframes are selected from the 27th Keyframe and the value of Timing Amount was adjusted to 5 and then get the results which are showed in Figure 4.3 and 4.4. First, since the interval of this part is enlarged, the curve of the corresponding part becomes slow down. Then, the motion also slows down during this period. Therefore, the experimental results are in line with expectations.

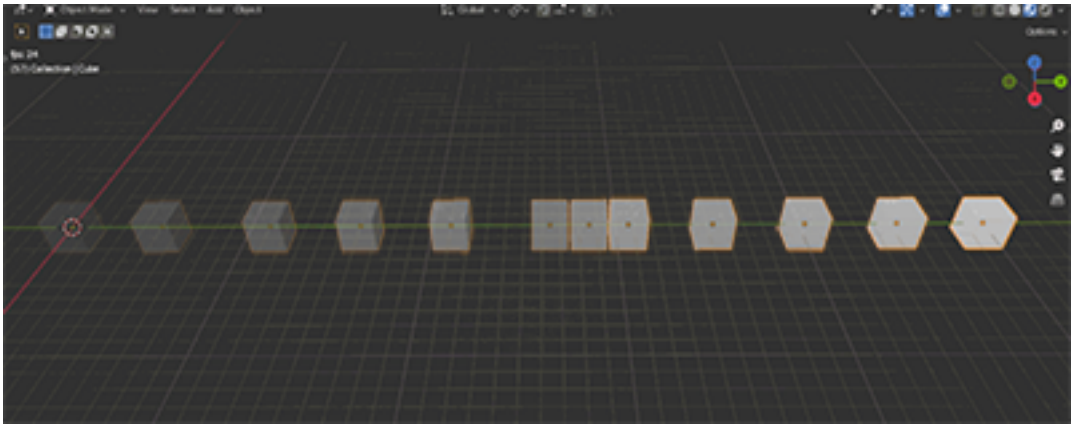

Figure 4.3: Dynamic picture of test Case 1

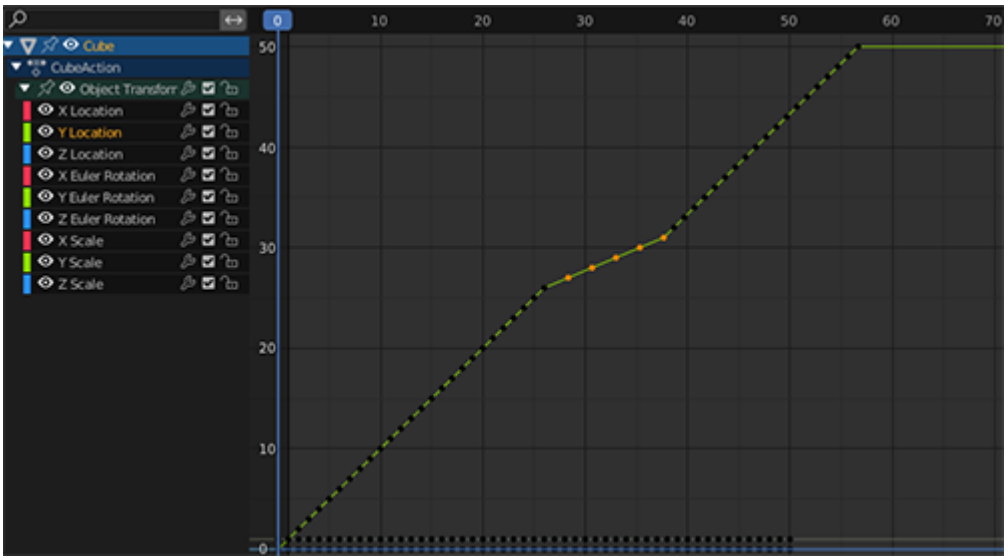

Figure 4.4: F-curve of Case 1

In the Case 2, the whole F-curve selected is taken as the target, and then adjust the Slowness Amount to execute both Slow In and Slow Out with the Quatic Function. After doing in this way, we can observe that the performance of both motion and F-curve can show the feature of *Slow In and Out* in Figure 4.5 and 4.6 and thus the result can be regarded as the same as the animation principle expected.

Here, although the experiment is based on the linear motion, it is supposed to give the similar *Slow In and Out* trend to the motion capture data as well. However, the appearance of its F-curve is fairly complex, thus Obvious effects can be expected to be observed in the actual motion of the 3D model, which will be shown in the next section.

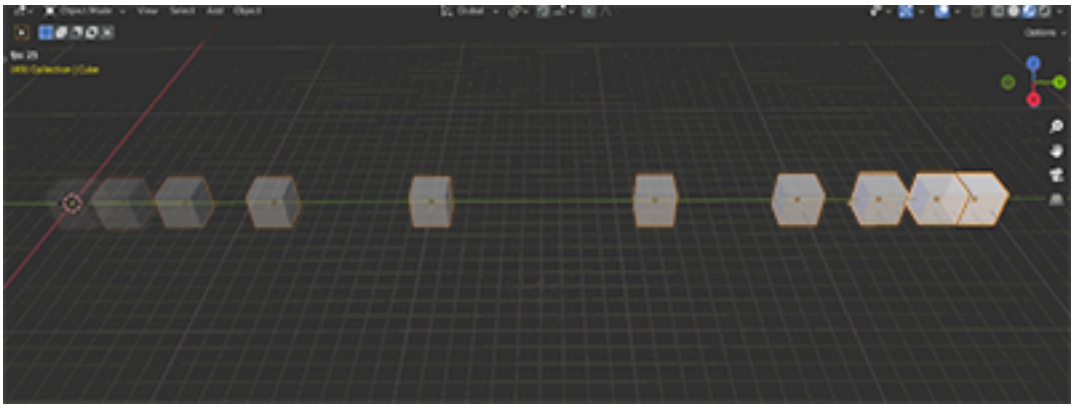

Figure 4.5: Dynamic picture of test case2

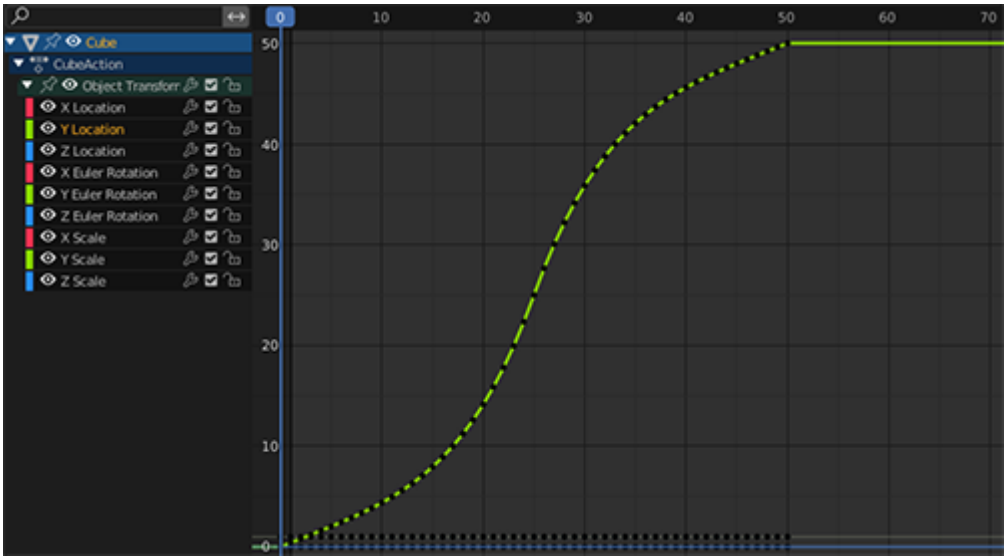

Figure 4.6: F-curve of Case 2

The purpose of Case 3 is to examine whether all transformations, including displacements, rotations, and scaling, can be achieved ideal exaggeration through the method that enlarging the y-axis coordinates, which is used for implementing the exaggeration algorithm in order to enhance the expressiveness of animation.

In this Case, 1 Location F-curve(when the Base is set to Min, the Exaggeration Amount is set to 2.5) and 3 Scale F-curves(when the Base is set to None, the Exaggeration Amount is set to 5) are selected to exaggerate. As shown in Figure 4.7 and 4.7, the displacement

distance becomes 2.5 times the original and the volume becomes 5 times the original, which meets expectations.

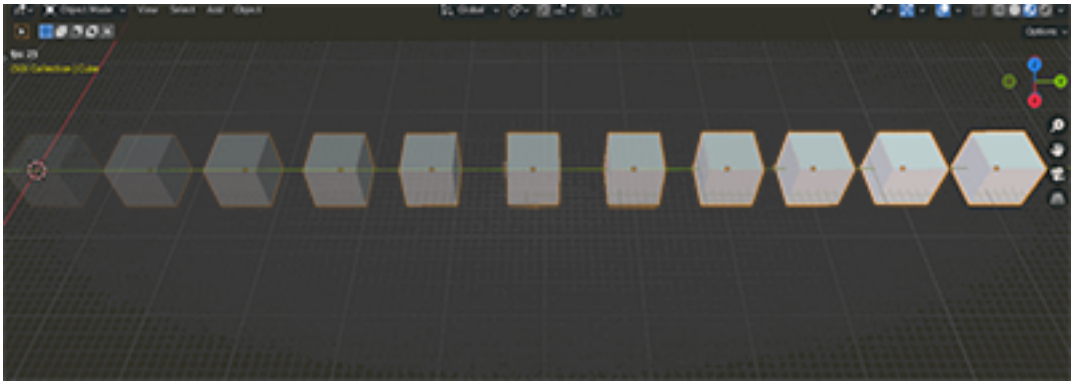

Figure 4.7: Dynamic picture of test case3

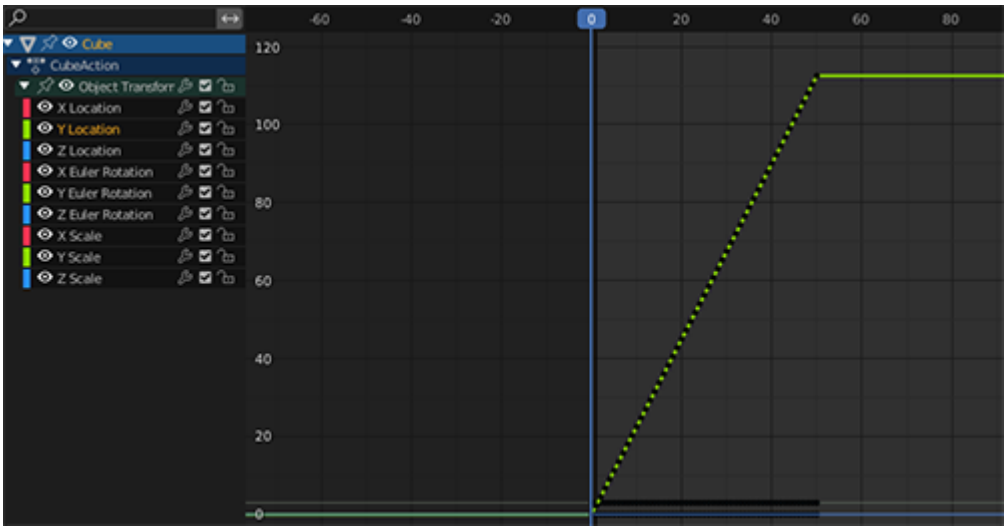

Figure 4.8: F-curve of Case 3

Here is thes Table 4.2 that records the changes in displacement of the object over time as it moves from the initial state to each of the three test cases. The different changes brought to animation by different algorithms can be observed through comparison.

Time(s)	0.3	0.6	0.9	1.2	1.5	1.8	2.1	2.4	2.7
Dist**I(m)**	9.0	17.0	22.0	30.0	37.0	45.0	50.0	50.0	50.0
Dist**1(m)**	9.0	17.0	22.0	27.25	30.7	37.25	45.3	50.0	50.0
Dist**2(m)**	3.85	10.0	17.8	35.85	43.7	48.0	50.0	50.0	50.0
Dist**3(m)**	20.25	38.25	67.5	83.25	101.25	112.5	125.0	125.0	125.0

Table 4.2: The relationship between displacement and time of 3 Cases Compared to the original uniform motion

4.2 Results

This section shows the effect of the Add-on's practical application in Unreal Engine. Since the animation is a dynamic process, to make the effect to be observed, the Ghost Trails technique is used to record the change of the action. Also, the animation created by the original Mocap data is also essential to show what animation principles has been applied to the animation by the Add-on and how the performance is. We can observe the difference or improvement by comparing them.

In the Figure 4.9, it contains 2 character animations(running and waving) created by the original Mocap data. Here, two expression methods are used to clarify the action. For running, the method of making the phantoms not disappear and always exist is adopted because the action duration of running is relatively long. And for waving, since the action cycle is very short, the method is regularly cleaning up the ghost and keeping the phantoms of one cycle only to avoid too many repeated records interfering with the judgment.

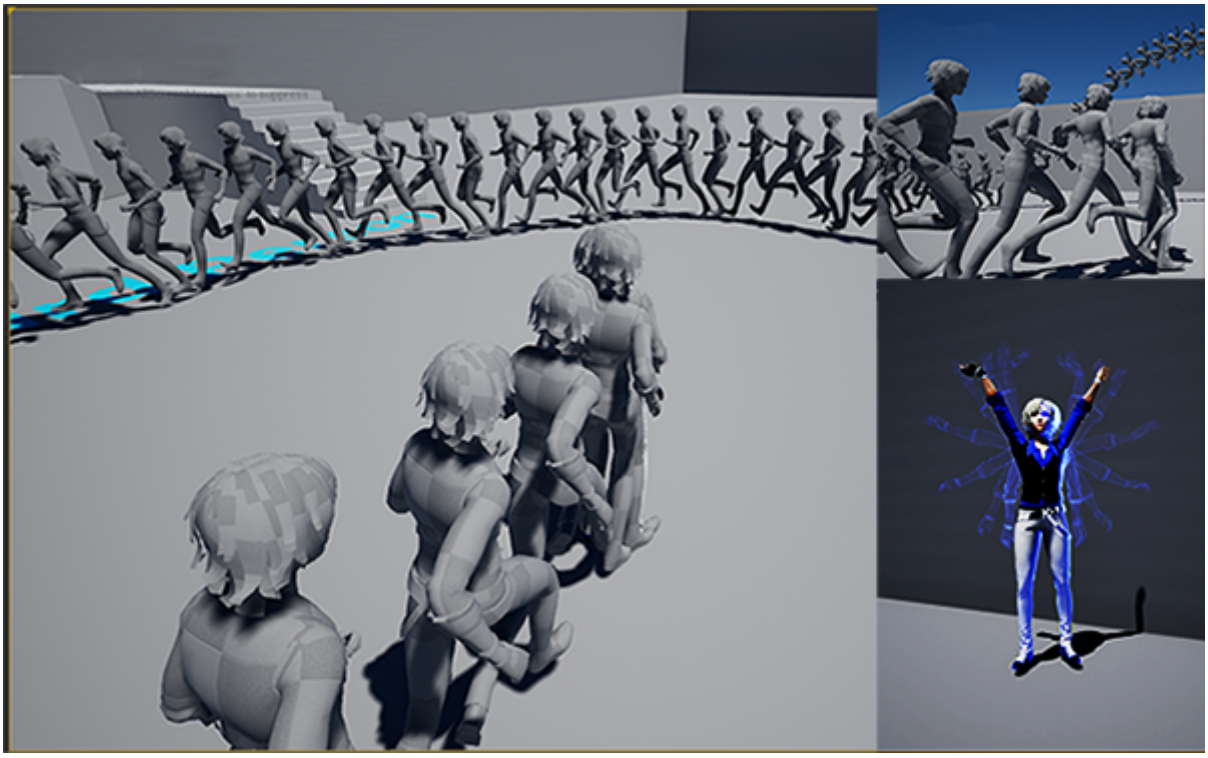

Figure 4.9: Animation in Unreal Engine created by Original Mocap data

Then use the Add-on to apply the animation principles of *Timing* and *Exaggerations* to the walking animation. To be specific, the things done here is increasing the duration and

exaggerate all the F-curves. It can be observed from the Figure 4.10 that the speed of the processed animation is obviously slowed down and the action range is also increased comparing to the original animation.

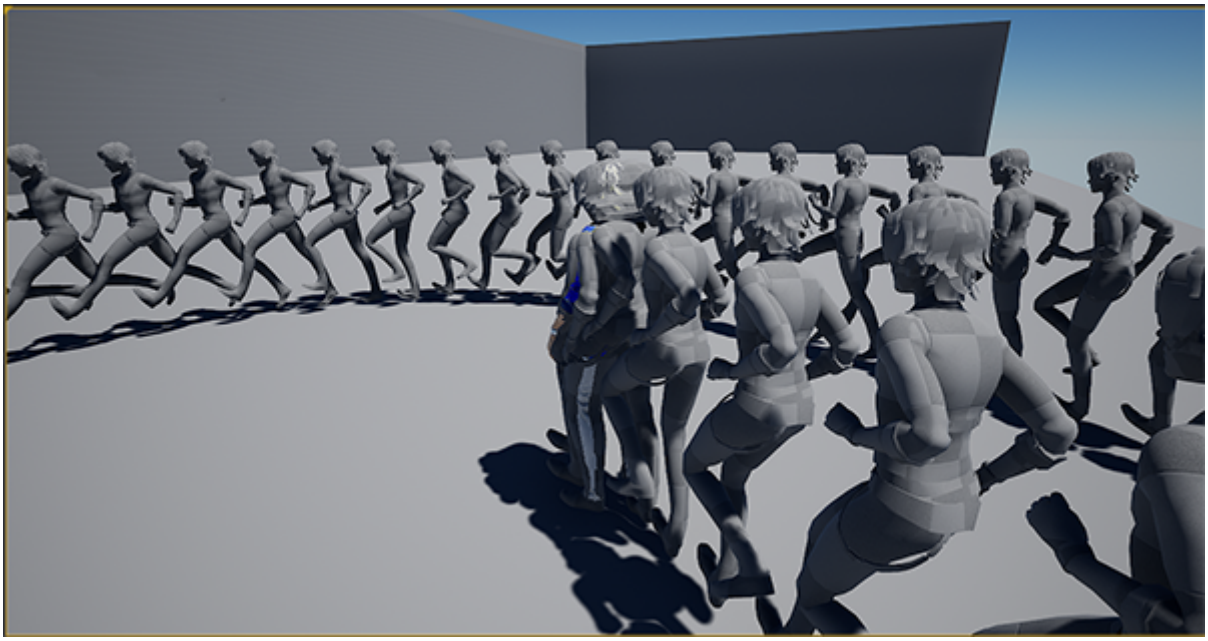

Figure 4.10: Animation in Unreal Engine created by edited running

Different from the previous example, this time we want to edit not all the F-curves but the F-curve of specific transform of 2 skeletons(left and right arm), which can reflect the animation principle of *Slow In and Out* evidently while the character waving his arm. Therefore, the first thing need to be done is to find the target F-curves from the F-curve group in the middle of Figure 4.11. And then, apply the fading to the selected F-curves.

Figure 4.11: Left: The original F-curve; Middle: List of the F-curves; Right: The F-curves edited by the Add-on

Finally, the appearance of the selected F-curve will be changed from the left to the right in the Figure 4.11. Import the processed Mocap data into the Unreal Engine Editor and create the Ghost Trail animation which is shown in Figure 4.12. It is obvious that the action on the right performs the affect of *Slow In and Out* rather than the original one on the left.

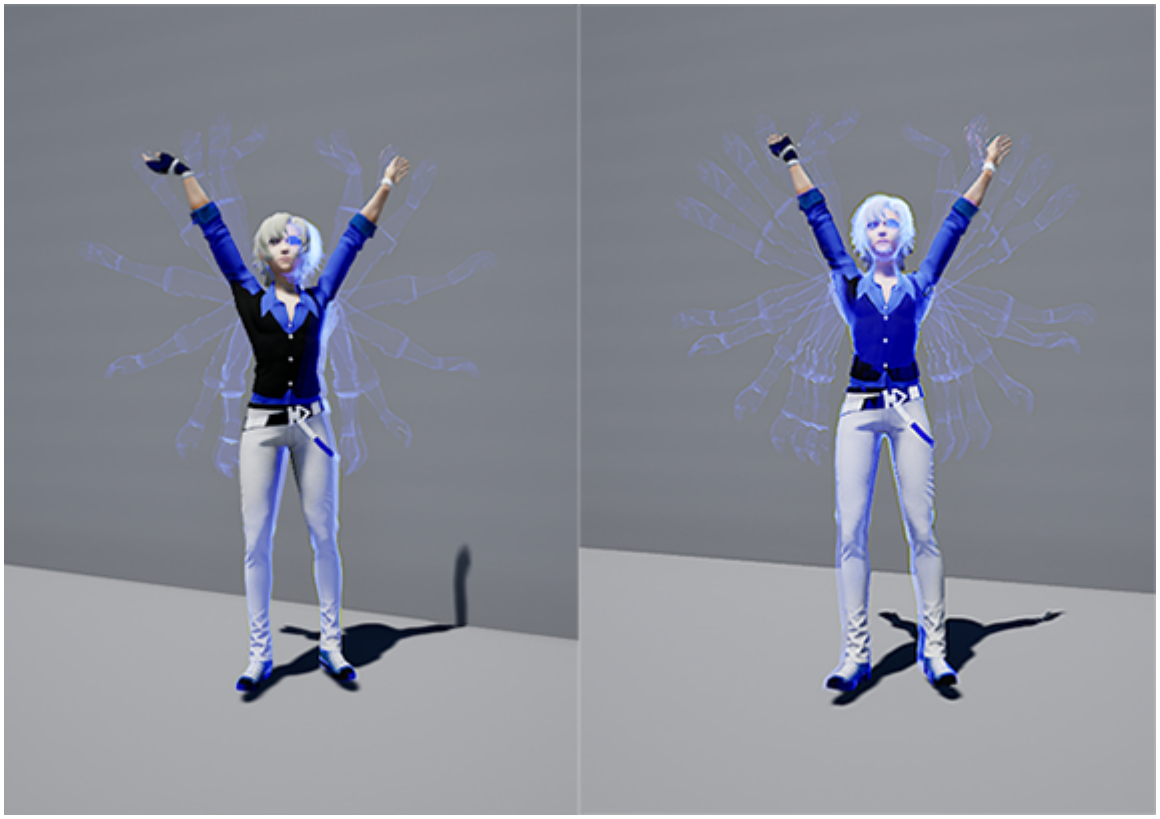

Figure 4.12: Left: Animation created by the original Mocap data; Right: Animation created by the edited Mocap data

Chapter 5

Conclusions & Future Work

This chapter summarizes the work done by this dissertation, as well as the possibilities that could be undertaken to improve or extend the implementation in the future.

5.1 Conclusions

The purpose of this dissertation is to explore algorithms to apply Disney's classical 12 principles to motion capture data, so as to improve the expressiveness of animation. Obviously, it is impossible to design and implement the algorithm of all animation principles in this dissertation. Therefore, as the first step of the dissertation is to learn and understand Disney's 12 classical animation principles and decide which principles should or can be realized for the time being. Then the principle of *Timing*, *Slow In and Out* and *Exaggeration* are chosen as the target because there are some relevant research and projects that can be used as reference. During the design process, some existing Add-on that can complete similar functions are also referred to. In the end, a Add-on based on Blender including the functions of *Timing*, *Slow In and Out* and *Exaggeration* is completed. Although the current implementation still has some shortcomings and room for improvement, the main algorithms and related functions are completed as planned.

5.2 Future Work

The work presented in this dissertation can be improved from several directions. These possible improvement could be optimization of the algorithms, simplification of the user operation, as well as achievement of more animation principles like *Squash and Stretch*, *Secondary Action* and so on.

Optimization of the algorithms: The current algorithm is completely based on F-curve and does not consider the optimization of key frames. For example, while changing the duration in *Timing*, the number of key frames should be controlled dynamically by the algorithm. To be specific, if the interval is reduced, the keyframes in the middle will play a smaller or even useless role. At this time, it is better to delete some appropriately; On the contrary, if the interval increases, the range dominated by the key frames will become larger and the flexibility will be reduced. At this time, some keyframes should be appropriately added in the middle. The same problem also exists in other functions.

Skeleton	Location			Quaternion				Scale		
Hips	x	y	z	w	x	y	z	x	y	z
Spine	x	y	z	w	x	y	z	x	y	z
Spinel	x	y	z	w	x	y	z	x	y	z
Spine2	x	y	z	w	x	y	z	x	y	z
Neck	x	y	z	w	x	y	z	x	y	z
Head	x	y	z	w	x	y	z	x	y	z
LeftShoulder	x	y	z	w	x	y	z	x	y	z
LeftArm	x	y	z	w	x	y	z	x	y	z
LeftForeArm	x	y	z	w	x	y	z	x	y	z
LeftHandLeftHand	x	y	z	w	x	y	z	x	y	z
LeftHandIndex1	x	y	z	w	x	y	z	x	y	z
LeftHandIndex2	x	y	z	w	x	y	z	x	y	z
LeftHandIndex3	x	y	z	w	x	y	z	x	y	z
RightShoulder	x	y	z	w	x	y	z	x	y	z
RightArm	x	y	z	w	x	y	z	x	y	z
RightForeArm	x	y	z	w	x	y	z	x	y	z
RightHandLeftHand	x	y	z	w	x	y	z	x	y	z
RightHandIndex1	x	y	z	w	x	y	z	x	y	z
RightHandIndex2	x	y	z	w	x	y	z	x	y	z
RightHandIndex3	x	y	z	w	x	y	z	x	y	z
LeftUpLeg	x	y	z	w	x	y	z	x	y	z
LeftLeg	x	y	z	w	x	y	z	x	y	z
LeftFoot	x	y	z	w	x	y	z	x	y	z
LeftToeBase	x	y	z	w	x	y	z	x	y	z
RightUpLeg	x	y	z	w	x	y	z	x	y	z
RightLeg	x	y	z	w	x	y	z	x	y	z
RightFoot	x	y	z	w	x	y	z	x	y	z
RightToeBase	x	y	z	w	x	y	z	x	y	z

Table 5.1: F-curves of a Motion capture data

Simplification of the user operation: In the current Add-on, if the user want to edit a specific transform of a skeleton, he has to find the target F-curve from a very long list. As

shown in Table 5.1, it is just a very simple skeleton structure. However, each bone belongs to the structure has the F-curve to be responsible for Location, Quaternion and Scale. As a result, a skeleton structure containing 28 bones will need 280 F-curves to control its transform. This will make users feel confused when using this Add-on. Therefore the possible solution can be add a search function for user to locate the target F-curve before applying the principles.

Other animation principles: This project has only realized 3 of the animation principles, however, the other 9 animation principles are often used in animation production and worth to be developed as well. Of course, some of them may be difficult to achieve using only F-curve. Such as the *Squash and Stretch*, when applying the principle, it is essential to ensure that the volume of the model before and after the change is constant. To achieve the aim, the function related to the mesh may be taken in account.

Currently, motion capture is not only limited to body movements, but also the facial motions. Through the use of cameras or laser scanners, facial motion can be captured electronically and converted into a digital database as well. Also, there is a standard approach called *BlendShape* for making expressive facial animations in the digital production industry. Therefore animation principles can also be supposed to make contribution to add expressiveness to facial motion capture data as well, which is not referred to in this dissertation.

Bibliography

Blain, J. M. (2019). *The complete guide to Blender graphics: computer modeling & animation*. AK Peters/CRC Press.

Cameron, G., Bustanoby, A., Cope, K., Greenberg, S., Hayes, C., and Ozoux, O. (1997). Motion capture and cg character animation (panel). In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 442–445, USA. ACM Press/Addison-Wesley Publishing Co.

Carter, C. P. (2016). *Animated Mise-en-scene and aesthetic harmony: An expansion of the traditional principles of animation to 3D computer animation*. PhD thesis, Queensland University of Technology.

Izani, M., Aishah, Eshaq, A., and Norzaiha (2003). Analysis of the keyframe animation and motion capture case studies. In *Proceedings. Student Conference on Research and Development, 2003. SCORED 2003.*, pages 177–182.

Kitagawa, M. and Windsor, B. (2020). *MoCap for artists: workflow and techniques for motion capture*. Routledge.

Koloskova, O. (2012). *Motion capture*. PhD thesis, Sumy State University.

Kwon, J.-y. and Lee, I.-K. (2007). Rubber-like exaggeration for character animation. pages 18–26.

Kwon, J.-Y. and Lee, I.-K. (2009). The squash-and-stretch filter for character animation. In *ACM SIGGRAPH ASIA 2009 Posters*, SIGGRAPH ASIA '09, New York, NY, USA. Association for Computing Machinery.

Lasseter, J. (1987). Principles of traditional animation applied to 3d computer animation. *SIGGRAPH Comput. Graph.*, 21(4):35–44.

Python, W. (2021). Python. *Python Releases for Windows*, 24.

Savoye, Y. (2012). Content-aware exaggerated editing for life-like captured animations. New York, NY, USA. Association for Computing Machinery.

Suwetha, R., Subedha, V., Kalaichelvi, T., and Hemalatha, S. (2017). Motion capture using 3d. *International Journal for Research in Applied Science and Engineering Technology*, 5(4):1220–7.

Thomas, F. and Johnston, O. (1995). *The Illusion of Life: Disney Animation*. Disney Editions Deluxe. Disney Editions.

Wang, J., Drucker, S. M., Agrawala, M., and Cohen, M. F. (2006). The cartoon animation filter. *ACM Trans. Graph.*, 25(3):1169–1173.

White, D., Loken, K., and van de Panne, M. (2006). Slow in and slow out cartoon animation filter. In *ACM SIGGRAPH 2006 Research Posters*, SIGGRAPH '06, page 3–es, New York, NY, USA. Association for Computing Machinery.

Appendix

Third party Assets:

3D Model from www.cg99.com

Motion capture data from Mixamo.com

Source:

`https://github.com/Kirito-JZP/Dissertation/blob/main/init.py`