Modern Science and Arts University

Faculty of Computer Science

CS406

Graduation Project II

Spring 2020

Rigging Manager for Skeletal Mesh in 3D environment

Name: Yahia Mahmoud Abdelrahman Basyouny

ID: 173269

Supervised by: Dr. Ahmed Farouk

Abstract

In this paper, we describe and explain our project which we will implement. We will talk about computer animation is a process used to generate an animated images, where it draws a lot of its appeal to animate impossible models. Then we will explain the 3D models which are created by rigging 3D figures with virtual skeleton, where 3D figures are the skin and virtual skeleton are the bones that move the 3D figure in 3D imaginary world like realistic world. After the models are built, we will be able to animate models, where IK uses equations of kinematics to set configuration of joints that satisfies a desired position of an end effector and end effector needs a position of a target to reach it, which will help us to animate the 3D models easily. The main objective of this project are: The first level of this project, is to develop and implement the inverse kinematics algorithm. Then we will implement an algorithm for mapping two different models, where they may have not the same deformation system of bones. Then we will implement a smart algorithm that produce better results than the first level for the animation of the 3D model. Our first algorithm is FABRIK which is a heuristic method of inverse kinematics. FABRIK offers a very realistic poses and has low computational cost. Most of the multi component models, like hand, legged bodies like spiders, etc., are actually made up of several kinematic chains, and each chain typically has more than one end effector. We applied and tested FABRIK algorithm on 5 spheres that represent a chain of 4 bones, then we applied and tested it on a complex model like elephant. Then we found that the accuracy of the system on chain of length less than or equal to five is great as the movement of the chain with the existence of the pole is great and smoothly realistic, but with chain of length more than five, it is very good without the pole but with the pole the deformation of the model is damaged and produce unacceptable results. We think that our implemented system could be useful for some users. Users like designers, animators and it may be beneficial for freelancers and for small businesses.

Contents

Table of figures:

Chapter 1
Introduction

1 Introduction

1.1 Introduction

Computer animation is a process used to generate an animated images, where it draws a lot of its appeal to animate impossible models (creatures and objects). While it refers to move and animate images to create an animated graphical scene like in games and movies. (1) So there is two types of computer graphics that are used for animating models, 2D computer graphics and 3D computer graphics. 2D stands for two dimension, where 2D computer graphics consists of digital images which are two dimensional digital images, text, or 2D geometric models. 3D stands for three dimension, where 3D computer graphics consists of three dimensional geometric data such as digital images, text, or 3D geometric models. (2) Famous examples like multi-component characters in feature 3D movies, such as The Incredibles, Minions, and Finding Dory. For 3D animations, the models are created by rigging 3D figures with virtual skeleton, where 3D figures are the skin and virtual skeleton are the bones that move the 3D figure in 3D imaginary world like realistic world. After the models are built, all the frames of the model must be rendered where the rendering process is the key of illustration process. (3) So by using inverse kinematics (IK), we will be able to animate models, where IK uses equations of kinematics to set configuration of joints that satisfies a desired position of an end effector and end effector needs a position of a target to reach it, those joints may be constrained or may not. (4) And Inverse kinematics is important to 3D animation and game programming. Animated models are built with a skeleton of rigid segments connected with joints that is called by kinematic chain. So the equation of the kinematics of a model or a figure describe the connection between pose of the figure and the joint angel of the figure. (5) As Shown in figure 1, the joints are being calculated while the end effector is trying to reach the target, where the joints are connected to each other from base (root) to end effector with parent to child relation. (6) Each child joint moves, it effects its parent and so on. IK is widely used in computer graphics for models animation and in robotics for motion planning. (4)

Figure 1: this figure show how the inverse kinematics work, where the end effector is trying to catch the target. (4)

1.2 Problem Statement

The problem stated in this project is to find 3D model with virtual skeleton, then develop and implement inverse kinematics algorithm to apply this algorithm on the 3D model to be able to animate realistically and smoothly in a 3D environment. IK is widely used in computer graphics for models animation and in robotics for motion planning. (4) Also we will make an algorithm for mapping 2 different types of 3D models, by mapping the 2 models, the first model will be able to animate as the second model. Also there is another problem stated in this project is to develop and implement a smart algorithm by using machine learning to make the 3D model components animate and reach the desired targets smoother.

1.3 Objective

The main objective of this project is to hopefully proceed three levels of this project. The first level of this project, is to develop and implement the inverse kinematics algorithm which will help us to use it to animate a 3D rigged model in 3D world. Where it will be great if it outcomes sufficient results by using inverse kinematics. The second level of the project, is to implement an algorithm for mapping two different models, where they may have not the same deformation system of bones. Then we will make them animate exactly the same, by mapping the first model bones on nearly the same bones of the second model. The third level of this project, is to develop and implement a smart algorithm that produce better results than the first level for the animation of the 3D model. By selecting the nearly best path from different paths to reach the desired target, where those paths will be steps of transition targets will lead to the desired target, so the

objective of the third level objective is how to select nearly the best suitable path to perform the smoother animation for the 3D model by using machine learning.

1.4 Motivation

The main motivation of this project is to create a 3D models that is rigged and animated using algorithms like inverse kinematics that will be useful and helpful for users. Users like designers that wants 3D animated models to test their work with it or to use it in an environment of their own, also the animators need animated 3D models for the production of animated films. It also will be beneficial for freelancers and for small businesses. And at last, computer scientist are the last user that is motivated to create this project, because I am into computer graphics, where I will gain more knowledge about this field. I am into computer animation too and how to animate 3D models with algorithms like inverse kinematics, and I wish to develop and implement all my objectives to reach to my goal.

1.5 Thesis Layout

In the section of chapter 2 which is called Background and literature Review, we will explain what is needed to implement this project. For example: what algorithms are going to be used to reach the objectives of this project. It will also explain the previous works that nearly used the same methodologies that this project will use, and what results these other works achieved. In the section of chapter 3 which is called Materials and Methods, will explain the materials that will be used in this project such as the data (type, size, etc...), the tools that will help in the implementation, and the environment that the project will be implemented and run on it such as (CPU, GPU, Cloud, etc…). It will also explain our methodology and algorithms that are going to be implemented and applied in this project and the system architecture that will describe the project over view. In the section of chapter 4, we will discuss the implementation of the system and what we went through to reach our goal and explain the phases of the system development to show what we went through and what we used, then we will explain our system overview and each component in the system and also we will explain our class diagrams of each algorithm or code used in our system, then we will explain our system running, and what are the inputs and outputs of each component to reach our system goal as planned. In the section of chapter 5, we will explain our results in case studies, then we will explain our evaluation of the system and what is the accuracy of our results, the spaced used, and the performance of the system at

different architectures. At the section of chapter 6, we will explain the conclusion of what we have developed, implemented, and done at our system, and about what we have found during the implementation of the systems, then we will state each problem we faced at our development, then we will state our future work that will improve our system.

Chapter 2

Background

And

Literature Review

2 Literature Review

2.1 Background

To be able to develop and implement this project you need to have background about computer graphics, computer animation, and machine learning. Where computer graphics and computer animation will lead you to know about 3D animated models, also you need to know how 3D animated models which are built with a skeleton of rigid segments connected with joints that is called by kinematic chain. And you need to know about how the 3D model structure is built. By knowing the structure of the 3D model, you will be able to apply one of the many inverse kinematics algorithms that will help you to animate this 3D model. There are many inverse kinematics techniques for modeling and solving the problems of inverse kinematics. Heuristic methods can be used to approximate the inverse kinematics problem. Those methods approximately solve the IK problem, where they are perform simple iterative operations. The heuristic methods return the final poses fast as they have low computational cost, and simply support joint constraints. The techniques of inverse kinematics are The Jacobian Inverse Technique, Cyclic Coordinate Descent (CCD), and Forward and Backward Reaching Inverse Kinematics Technique (FABRIK). (7) (8) CCD and FABRIK are the most popular heuristic algorithms.

2.1.1 Cyclic Coordinate Descent (CCD) technique

CCD refers to cyclic coordinate descent which is a well-known heuristic algorithm used in multi joint chains to provide inverse kinematics solutions. (4) Its viability for controlling and creating very complicated models like human and insects. The reason why cyclic coordinate descent algorithm is well known is because it is simple algorithm that is easy to understand and implement, a computationally fast, and simple computational method that can run at interactive frame rate for generating inverse kinematics solutions. (9) While the development of an Inverse kinematics system using CCD may be completely clear. We examine series of engineering solutions that are necessary to make the CCD method a functional method and realistic for the characters based on environments like games. (9) The advantage of CCD method is that does not need any complex decomposition of matrix mathematics. The method user need to know about the basic vector mathematics like dot project, cross product, and angles. (9) For each joint, the vectors are formed.

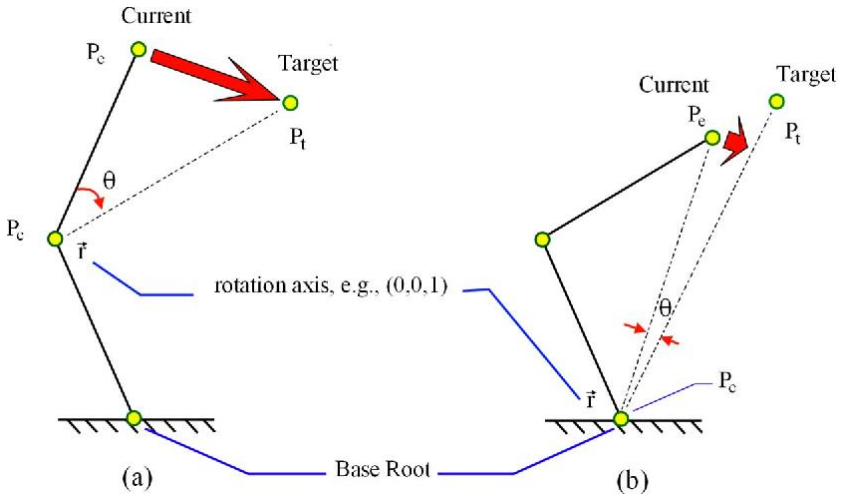

Figure 2: this figure show how the CCD method work. (9)

2.1.2 Forward and Backward Reaching Inverse Kinematics Technique (FABRIK)

FABRIK algorithm is a fast iterative solver for IK problem. IK uses equations of kinematics to set configuration of joints that satisfies a desired position of an end effector and end effector needs a position of a target to reach it as rapidly, smoothly, and accurate as possible while animating, those joints may be constrained or may not. (4) FABRIK uses joint positions that is previously calculated to find the updates in an iterative forward and backward mode. (8)

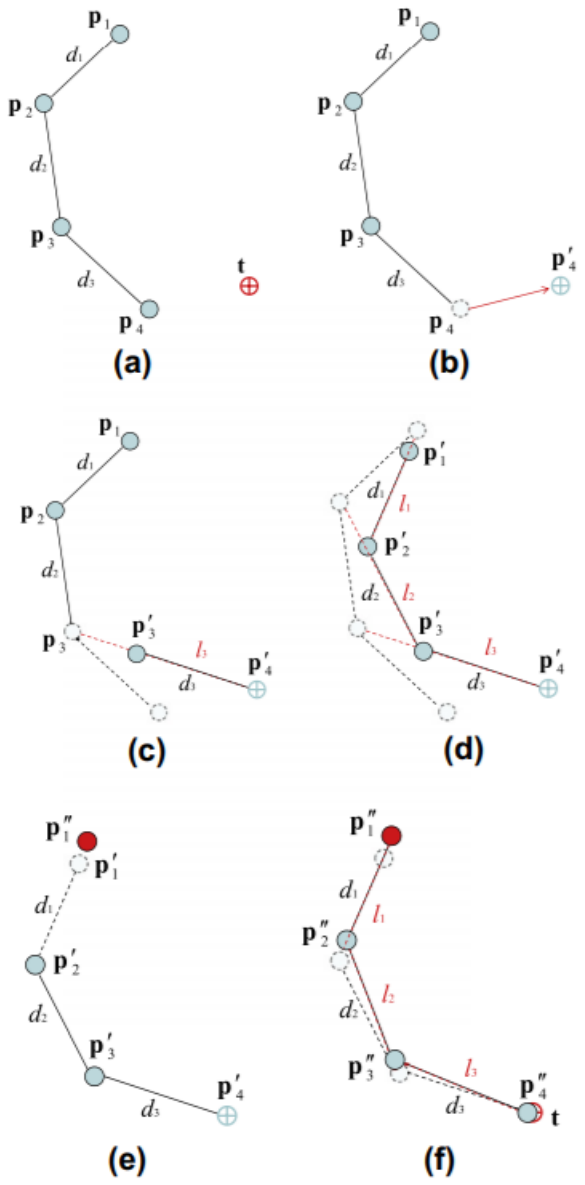

Figure 3: A figure for a full iteration of FABRIK with 1 target and 4 connected joints. (8)

FABRIK also involves in minimizing error that is exist in the system by modifying each angle of a joint at a time. The suggested technique begins from the last chain joint and works forward to adjust each joint along the path. Then in order to complete a full iteration, it will work backward in the same path. Many of the methods that currently available are producing unrealistic poses of animation, and suffering from computational cost that is relatively high. FABRIK is a heuristic method is compared with many of the popular existing methods according to computational cost, reliability, and criteria of conversion. Forward and

Backward Reaching Inverse Kinematics avoids the use of matrices or rotational angles, in state of that it finds each joint position by locating and fixing a point on a line. (8) FABRIK method produces a very realistic poses and has low computational cost, where it converges in few iterations. Most of the multibody models, like human, hands, legged bodies like spiders or elephants, etc., are actually made up of several kinematic chains, and each chain typically has more than one end effector. Hence, the ability to solve problems with multiple end effectors and desired targets is necessary for an IK solver. It is easy to adjust and extend the proposed algorithm to process models that contain multiple end effectors and desired targets. But the number and structure of the chains. And the subbase joints of the model are needed to be known as a prior knowledge. The algorithm is broken down into two stages, as in the case of the single end effector. The standard algorithm is implemented in the first stage but this time it begins from each end effector and it travels inwards to the parent subbase. This will provide as many diverse subbase positions, according to the number of end effector attached to a particular subbase. (8) The subbase's new position will become the centroid of all of those positions. Then the standard algorithm should be extended inwards from the subbase to the root of the manipulator. If more intermediate subbases exist, then the same technique should be used. The standard algorithm is implemented in the second stage and it is starting now from the root, then it is moving outwards to the subbase. (8) For each chain the algorithm should be applied independently till the end effector is reached, the same task is applied, if more subbases exist. The method is repeated until the targets are reached by all the end effectors, or there is not major change between their previous and new positions. The following figure show a model figure example with multiple end effectors and subbases. Models that are complex and sophisticate, they can be also tackled. Extending the proposed algorithm to take the shape, properties, and constraints of the figure into account will reduce and decrease the number of iterations that are needed to reach the desired target and will bring back more realistic postures. (8)

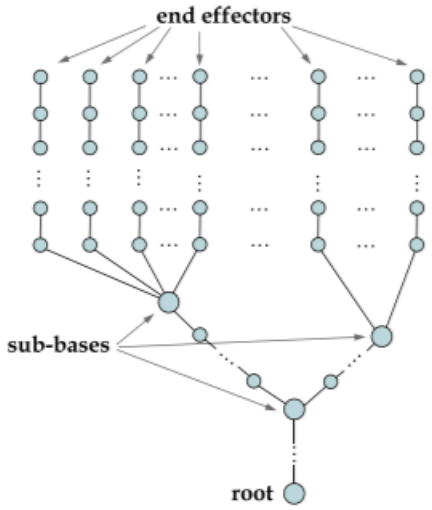

Figure 4: A figure shows a model with multiple end effectors and multiple sub bases. (8)

It is applicable to add constraints to FABRIK method, and it also support multiple chains with multiple effectors for a multi-component model. (8) The constraints of FABRIK method make the animation of the 3D model smoother and more realistic according to the normal approach like shown in the following figure of a human hand.

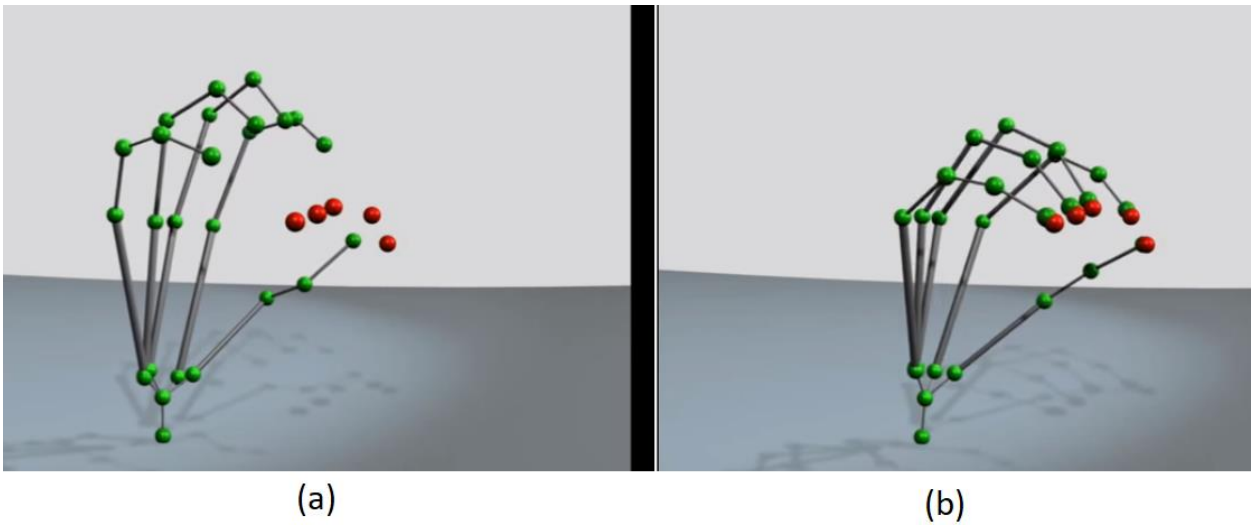

Figure 5: this is a rigged 3D model (Hand) which applies FABRIK with model constraints that supports multiple chains with multiple effectors for multi component model. (10)

2.2 Previous Works

Models are modeled and built on the computer monitor, where 3D models are rigged with a virtual skeleton for animation. While the animation of 2D models are separated objects and

separated transparent layers that are applied with or without the virtual skeleton. Then the clothes, limbs eyes, etc. of the model are being moved by the animator of the key frames. The computer automatically calculates variances in appearance between key frames, where this process is known as morphing or tweening, then the animation is rendered. (11) These are previous works used inverse kinematics techniques as they are very helpful in the development of computer animation, computer graphics, robotics, and more other field.

2.2.1 Inverse kinematics using dynamic parameters

2.2.1.1 Strategy

Over the past 30 years, various inverse kinematics techniques has been suggested (12). Various solutions have greatly improved stability, performance of the speed, and precision of the algorithms. (8) Many algorithms have been developed and implemented in the past few years to solve inverse kinematics specific problems. Some strategies have been suggested for identifying IK constraints and then solve them by using different priority levels. Many other work are aiming to create efficient natural human poses modeling. (13)

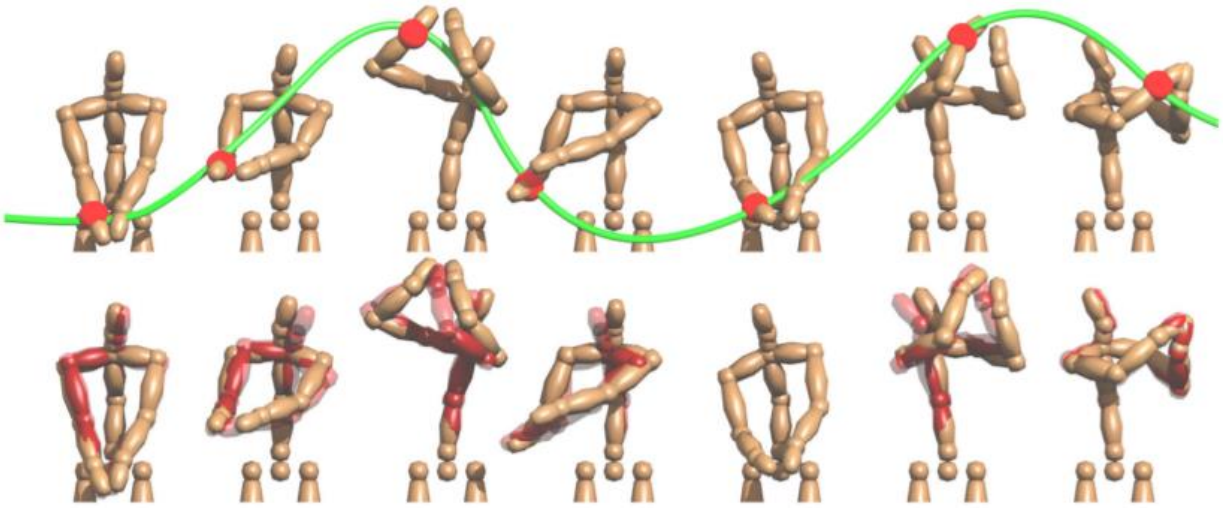

Figure 6: sequential poses of Golf swing. And comparing between wood texture (poses results) and transparent red (reference mocap data). (13)

There is no established work for our knowledge to tell us number of constraints that act on joints for any motion form. By defining the limits of the joint angular across the 3 dimension axes (x, y, z), the IK constraints are defined in 3D modeling applications like

Autodesk Maya and Blender. (13) Artists have to define these values then they can apply IK algorithm to create animation. According to a specific pose, the constraints are changeable. So for specific pose, the artists have to adjust the angle values. (13) The skeleton of the human is very complex structure, so the process can be long and the joints limits have dependencies hierarchy. Many solution may result for the skeleton by using the same end effector and not all of them are the acceptable poses as shown in figure 7. (13)

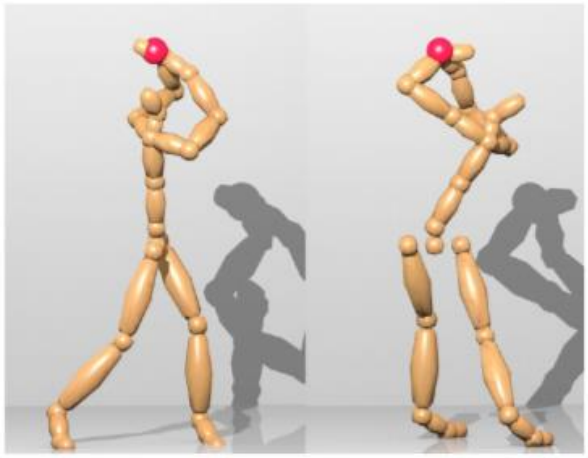

Figure 7: Different poses for the skeleton using the same end effector. (13)

To fix joint limits and achieve the acceptable desired posture, we need one acceptable solution. The efficiency of IK solutions appear on the simple structures, but it may appear on complex structures and impact computational cost. (13) Through parallelizing algorithms on multi core CPUs or GPUs, performance improvements can be achieved. Each pose is depending on the previous pose as shown in figure 8, since the process of IK is sequential process not straightforward. The trivial parallel inverse kinematics solver would independently calculate and estimate skeletal pose for each target position, if the target positions complete path is known. (13) The resulting sequence that is produced would probably be discontinuous and might not fulfill the predefined joint constraints. The contributions of this paper introduces a technique for inverse kinematics animation that is creatively constrained. Our technique conveys parameters automatically from data of input mocap. The parameters represent a human joint model that is behave dynamically natural, after the parameters are stored in an octree. Parameters are retrieved from the octree during the animation, and the parameters are used to solve the inverse

kinematics problem and also to move end effector to the target position. (13) We have developed and implemented our methodology in a parallel pipeline to improve and enhance the performance when the end effector complete trajectory is given, or the positions of the target in the streaming data formation such as “3D modeling software, real time video games and, SAIBA-like Embodied Conversational Agent platform”. [19] All generated poses are assembled using parallel filtering and retargeting processes to generate a sequence of smooth animation. Our solution is able to produce a visual motion of a high quality with a huge improvement in performance. (13)

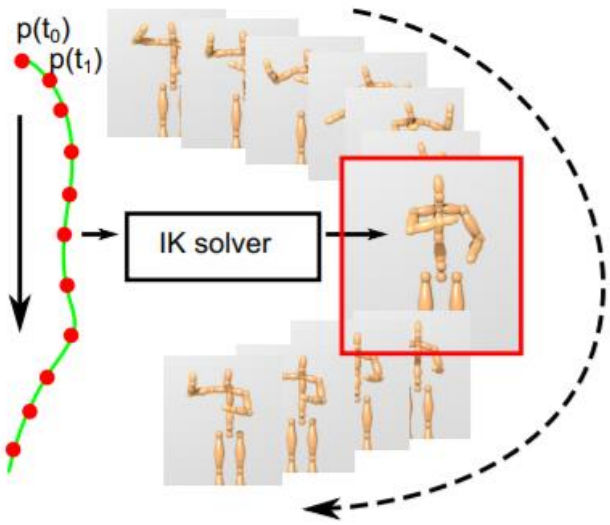

Figure 8: this figure show the process of IK is sequential process, where each pose is depending on the previous pose. (13)

2.2.1.2 Structure

We used a modified Selective Damped Least Squares inverse kinematics algorithm (SDLS) in our system, where SDLS is characterized by its stability, so that we chose it as our main inverse kinematics solver. (14) We will use octree because it is able to cluster, compute, and store the model parameters. We calculate the final animation in 4 parallel loops, as a set of target points which define the constraints octree and a trajectory. (13) Figure 9 explain the parallel pipeline of our IK framework. At the start each target point is assigned to different thread that calculates pose skeleton using the constraint motion parameters that are stored in the octree. Then it creates the animation first crude approximation, then it is refined in the following steps in a realistic smooth natural sequence. In the first step which is called Crude IK parallel pass, we seek to obtain

natural looking poses without being very coherent at the first. (13) In the following optimized steps during the time, realistic and smoothly changing poses will be obtained. In the second step which is called temporal alignment filtering pass, all the computed poses are aligned in a realistic continuous sequence during time. The last 2 steps are working in parallel, where they are used only to optimize the sequence of animation. These 2 steps called IK retargeting and correction passes. (13) IK retargeting is basically repeat the first step, as each target position is assigned to different thread and run the IK solver for each one of them separately. IK solver is supported with the previous step output, so this is the different between the current step and the first step. The end effector reaches the desired positions of the target and the skeleton poses are coherent and different from its previous pose (the first step). (13) Final correction pass although the position of the end effector and the coherency are fulfilled, there might be situation where the joint angular speeding up causes sudden changes in motion. (13)

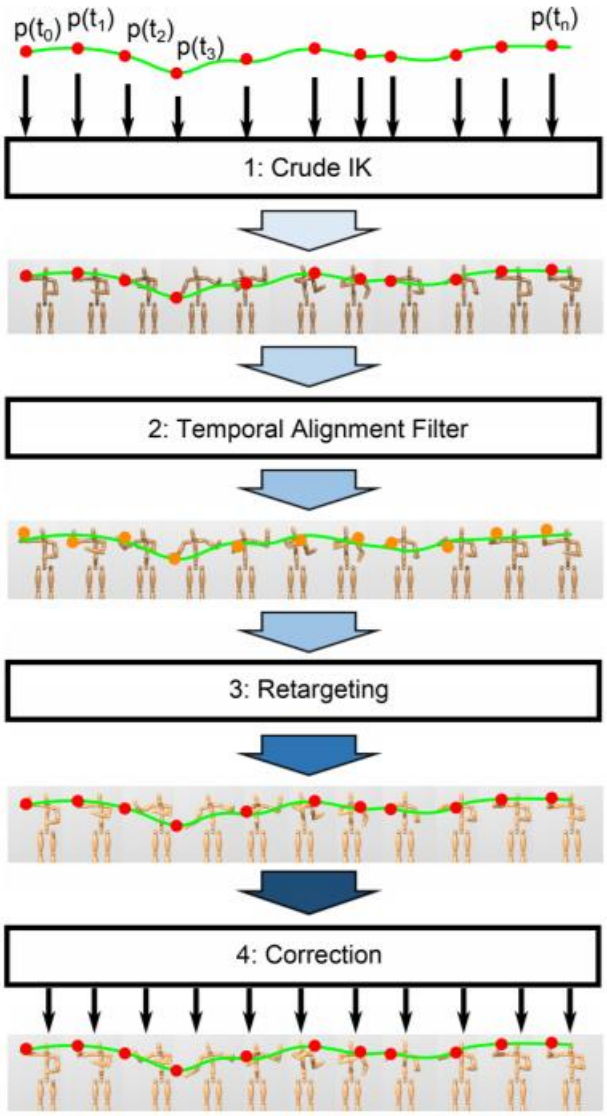

Figure 9: Parallel pipeline of our IK framework. (13)

2.2.1.3 Method Evaluation

Advantage: their method generate animation with high performance and high quality visual motion, where the animation is realistic and smooth.

Disadvantage:

1) They do not use GPU to speed up the performance
2) The input is trajectory, so without the temporal information, it is hard to apply the filtering process.

2.2.1.4 Data

The used input is the macop animation sequences in the Emilya data set. (15)

2.2.1.5 Results Evaluation

Their method are implemented and tested the performance on “an Intel Xeon CPU (2.93 GHZ) that contains 2 processors, where each processor has 4 cores. (13) To parallelize the code, the implementation uses Intel TBB (TBB means Threading Building Blocks). As shown in Table 1, we tested multiple examples and reported the result of performance, then we matched the computational time with the sequential implementation of our code. Notice that the sequential implementation first step equal to method performance of SDLS IK (identical to dls2 that is shown in Table 1 in fig.10). (13)

Table 1 Time performance comparison in milliseconds of different solutions when giving different trajectory frames (fs) from motion capture data: **sq** (our sequential solution), **pl** (our parallelized implementation), **dls1** (dls solution taking original pose as initial state), **dls2** (dls solution taking previous frame pose as initial state).(13)

Motion pass	sq/pl	CIP 1st	TAFP 2nd	IKRP 3rd	CFP 4th	Sum
Angry	sq	578.8	3.1	454.7	0.6	1037.3
1500fs	pl	78.8	1.2	62.6	0.3	143.0
	dls1					753.2
	dls2					461.3
Throw	sq	16.9	0.3	23.2	0.1	40.4
150fs	pl	2.8	0.2	3.6	0.1	6.7
	dls1					42.3
	dls2					15.5
Move	sq	110.3	1.0	50.1	0.3	161.8
600fs	pl	19.9	0.6	6.1	0.3	27.0
	dls1					191.3
	dls2					109.4
Golf	sq	265.1	3.7	200.2	0.7	469.7
2500fs	pl	36.6	1.7	38.5	0.3	77.1
	dls1					847.1
	dls2					304.5
Tennis	sq	171.0	3.7	153.1	0.6	328.4
2000fs	pl	24.1	1.4	23.4	0.4	49.3
	dls1					673.3
	dls2					211.7

Figure 10: Time performance comparison of different solution of SDLS IK. (13)

The amount of computing time isn't linear with the frames number, it also takes into cons ideration the number of iterations that is required to reach the desied target location, just l ike sdls ik and any iterative IK method. (13) We are also calculating the mean errors using both error metric in joint space and the distance space of the desired target to

evaluate the different techniques as shown in Table 2 in fig.11. Our solution will produce acceptable human postures nearly similar to the referenced mocap information matched with original dls solutions. (13)

Table 2 The error metric of comparing different solutions with original mocap in both joint space and target distance.(13)

Methods	Joint mse	Distance mse
Ours (sq/pl)	0.0015	0.00783
dls1	0.261	0.00861
dls2	0.326	0.00761

Figure 11: The error metric of comparing different solution with original mocap. (13)

The used input is the mocap animation sequences that exist in the Emilya data set. (15) It is discovered that the parallel version is 7 times faster than the sequential version, and it is also 4 times faster than SDLS that is not constraint, when generating the entire animation sequence. (13) It worth referencing that our strategy is scalable, implying that the performance is able to be improved, if we increased the number of cores for the implementation and test. We apply the joint of the right hand wrist as an end effector in all the test cases. (13) We use the hand trajectory from mocap sequences in the original data, to verify the produced animation. By the resulted animation, we contrasted the mocap data (the ground truth). We demonstrate normalized errors with various levels of depth with taking in consideration the octree as shown in the following figure. (13)

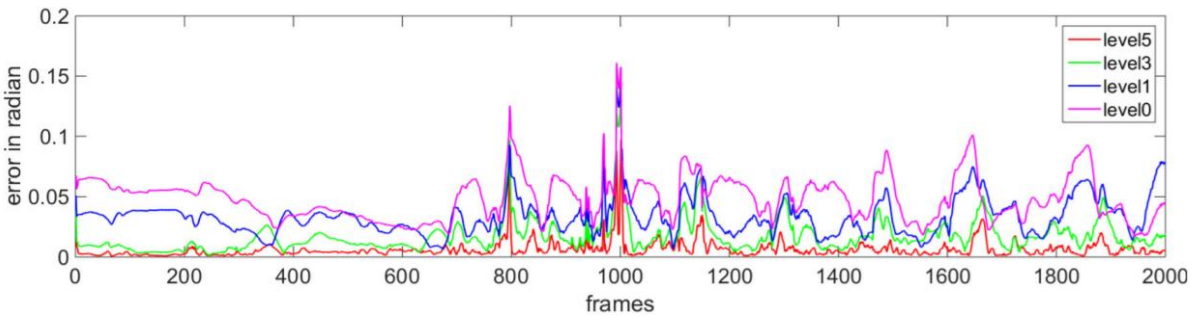

Figure 12: Comparison of normalized mean square errors (rotations over all joints). (13)

With various maximum depths, we have evaluated our method performance using the octrees. When the maximum depth value is greater than 2, the performance does not change suddenly, as revealed in Table 3 in fig.13. (13)

Table 3 Time to compute an animation using octrees with different maximal depth .(13)

Level	0	1	2	3	4	5
Golf	227.6	92.0	75.9	78.6	77.7	76.2
Tennis	115.7	72.5	65.6	51.1	58.1	51.4
Throw	12.1	8.2	7.6	6.8	7.8	8.3

Figure 13: Time to compute an animation using octrees with different maximal depth. (13)

The result of smoothing process is shown in the fig.14, the smoothness of the motion after temporal alignment and retargeting has already been greatly improved. (13) A continuous sequence is generated by the final filtering step.

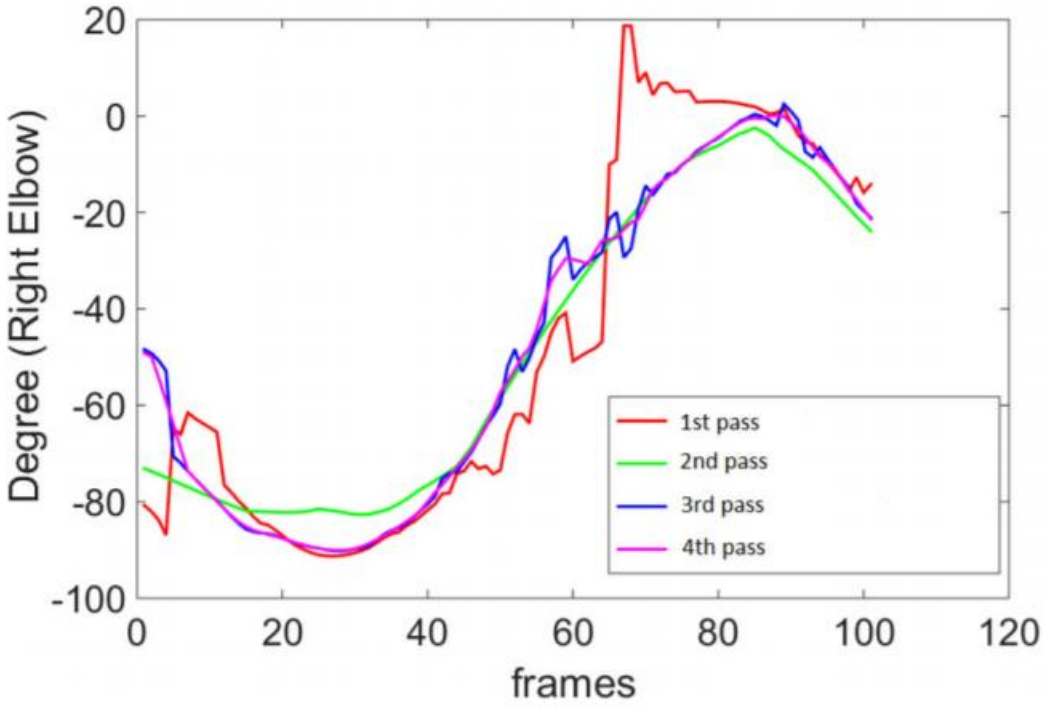

Figure 14: Elbow joint rotation in sequential frames. (13)

The trajectory target locations are not in the domain of leaned motion, our solution can still produce realistic and smooth motion as shown in figure 15 without specified learned joint limits in serious circumstances as shown in figure 16. (13)

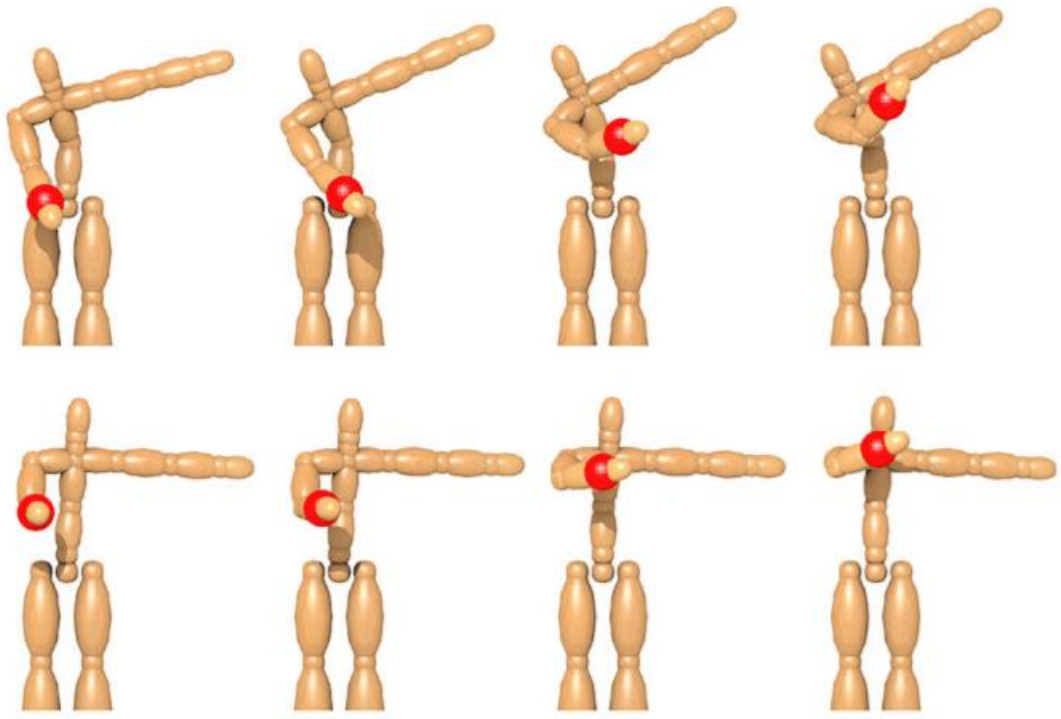

Figure 15: Trajectory of golf swing and tennis joint constraints, where we are generating tennis like golf swing. (13)

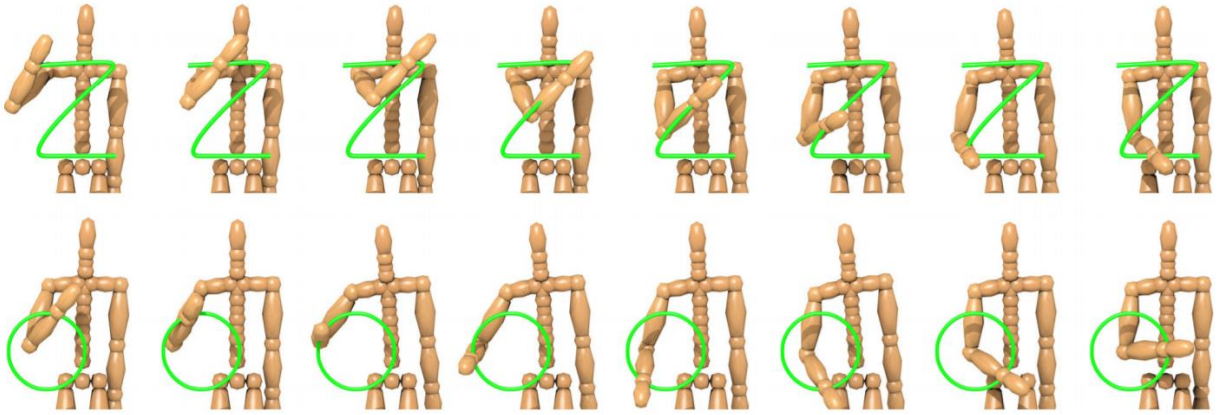

Figure 16: we are using octree to create novel motion does not exist in the input mocap data. (13)

Chapter 3

Materials and Methods

3 Materials and Methods

3.1 Materials

3.1.1 Data

The used data in this project are Assets of rigged 3D models by applying FABRIK method, where FABRIK method is a technique of inverse kinematics and FABRIK will help in animating this 3D models. The Assets contains its bones, textures, and materials. The model is download from a web site called turbosquid.

Format:

The used format in this project is .FBX or a full unity package. FBX means Filmbox which is a proprietary file format, where it is owned by Autodesk since 2006 and developed by Kaydara. FBX is used to applications unity, blender, and more. It is also a part of GameWare of Autodesk. FBX format will be easy to apply and use in this project.

Size:

The size of the assets will be less than 15 MB. Where the number of polygons are 3000 and the number of vertices are 3000.

Preprocessing:

The assets format was .ma (Autodesk Maya format) so it is needed to be converted into .fbx to be able to edit and use it on unity easily.

3.1.2 Tools

Software:

1) Unity: Unity is developed by Unity Technologies, where it is an engine for cross plate form game. It is preferable to use unity as it has easy graphical user interface, support physics, and the code is stable in unity as unity packed with a great architecture that improves the performance of the project. We used Unity platform for the implementation of our system because, Unity is an easy and a practical tool to be used, Unity also decrease the complexity of code and increase the capability and make the developer comfortable because it is based on a primarily high level programming language which is

C#, where C# programming language is easy to learn and make the development process of coding and gaming for developers easy. Unity editor provides to their users a drag and drop environment, Unity also provides an easy animator control to create and control animations without writing a lot of codes. Unity provides a development tool that is completely free for small company's (maximum 3 participants). Last of all, Unity also provides a lot of ready-mades and technologies to build systems and games.

2) Visual studio C#: C# is imported in unity that will be used to write and implement algorithm such as IK algorithms and machine learning algorithms.
3) Autodesk Maya: it is used to convert the assets from .ma format to .fbx format.

3.1.3 Environment

This project needs to work in an environment that can handle computer graphics, so the used environment is the GPU of a computer device. Where Graphics processing unit (GPU) is designed for specific use for mass calculation like in real time 3D graphics.

3.2 Methods

In this section we will explain our methodology and algorithms that are going to be implemented and applied in this project. Our first algorithm is FABRIK which is a heuristic method of inverse kinematics. FABRIK offers a very realistic poses and has low computational cost. Most of the multi component models, like hand, legged bodies like spiders, etc., are actually made up of several kinematic chains, and each chain typically has more than one end effector. Hence, the ability to solve problems with multiple end effectors and desired targets is necessary for an IK solver, where FABRIK offer this solution. Our second algorithm will be an algorithm for mapping two different models, where they may have not the same deformation system of bones. Then we will make them animate exactly the same, by mapping the first model bones on nearly the same bones of the second model. Our third algorithm will be a smart algorithm to select nearly the best suitable path to perform the smoother animation for the 3D model by using machine learning. All of these algorithms will help in the rigging of the 3D models.

3.2.1 System Architecture (over views)

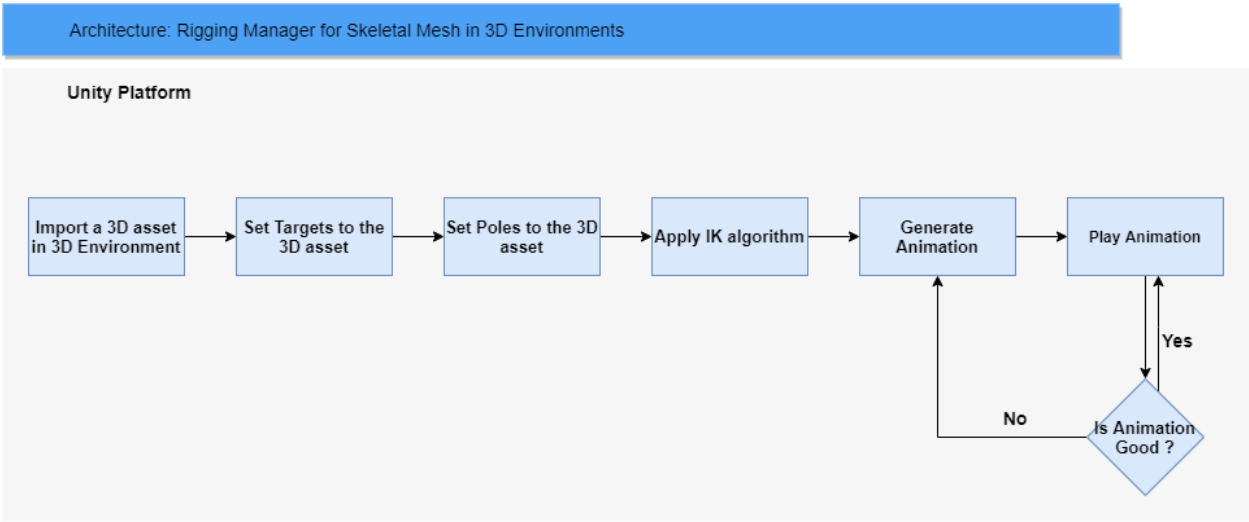

Figure 17: System Architecture.

1) 3D model

As shown in Start by importing the 3D model in unity with its materials and textures then add the 3D model to the scene of the project.

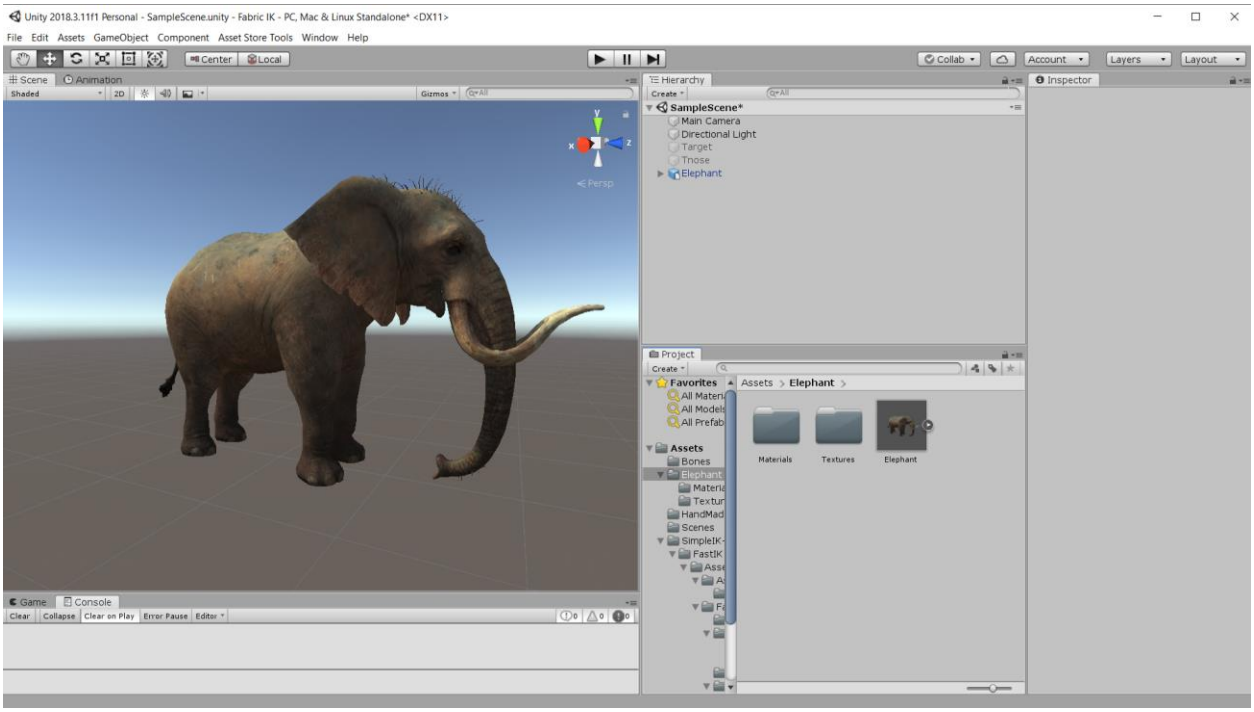

Figure 18: 3D model.

2) set targets

Set targets to the end effectors of the 3D model like shown in the following figure.

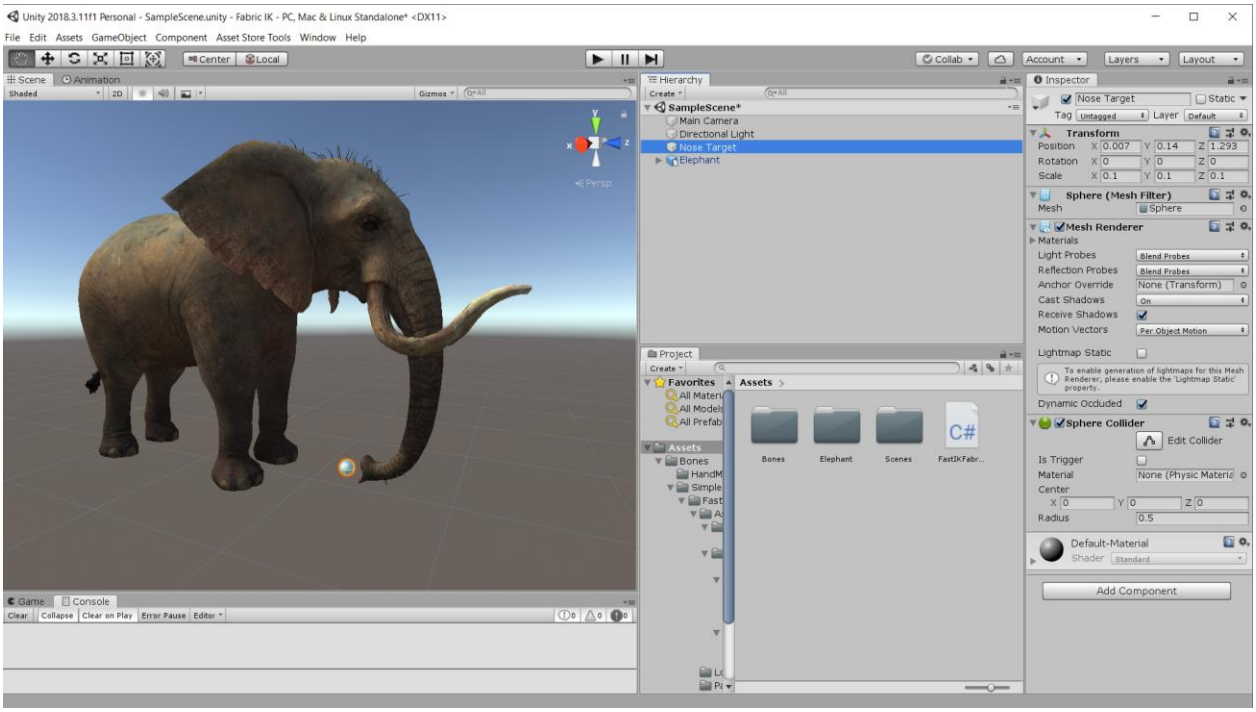

Figure 19: Set target to end-effector in 3D model.

3) set poles

Set poles as each kinematics chain as required in the 3D model as shown in the following figure.

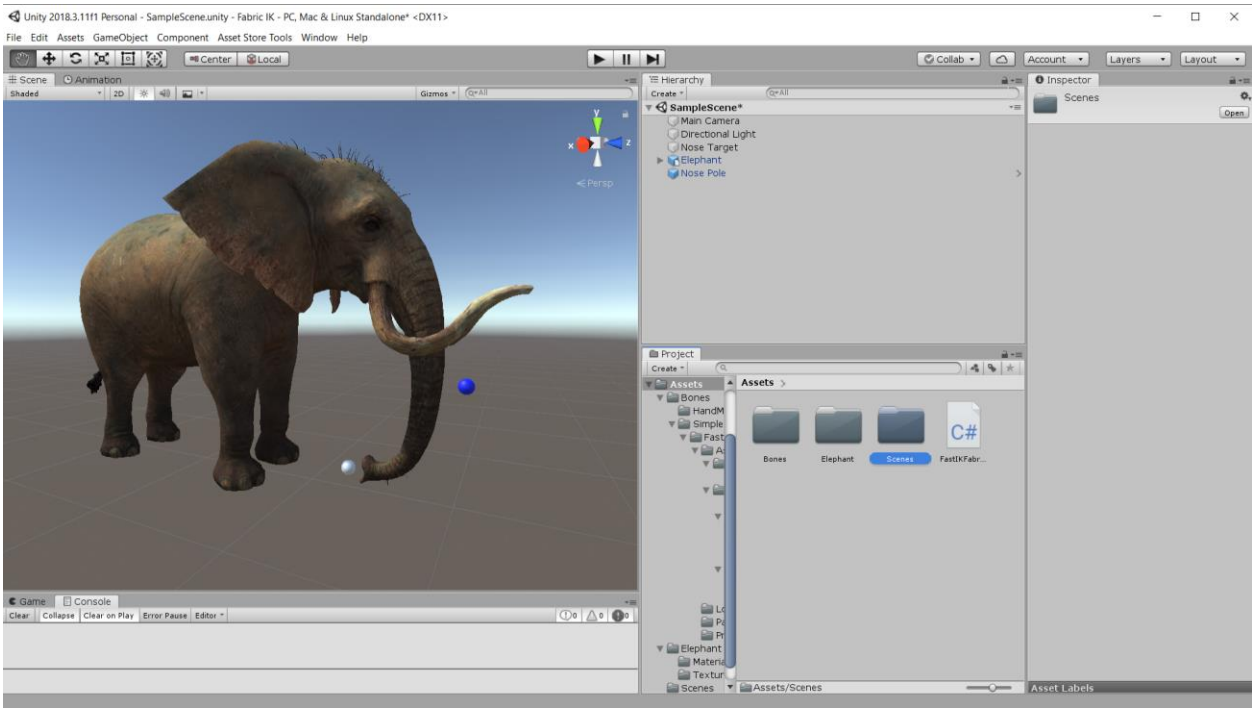

Figure 20: Set target to end-effector in 3D model.

4) apply IK algorithm

Apply inverse kinematics algorithm to each end effector of the 3D model as shown in the following figure.

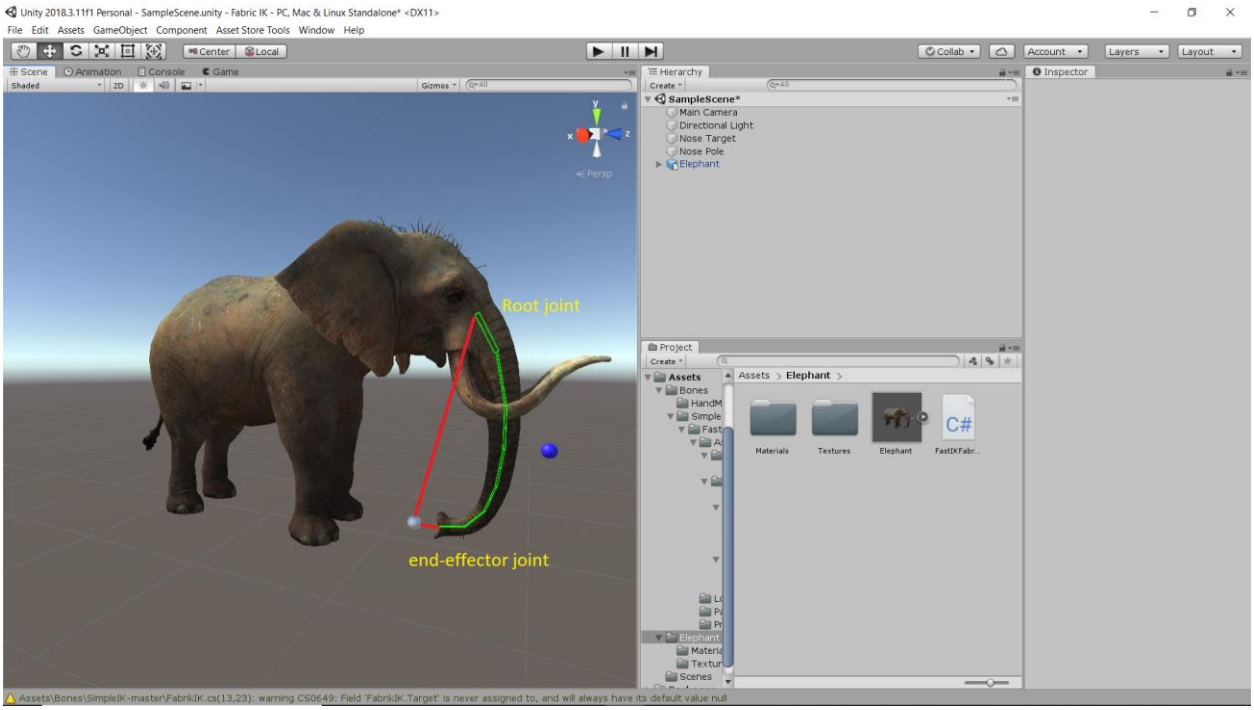

Figure 21: apply IK.

5) generate animation

Generate the animation of the 3D model then run the code as shown in the following figure.

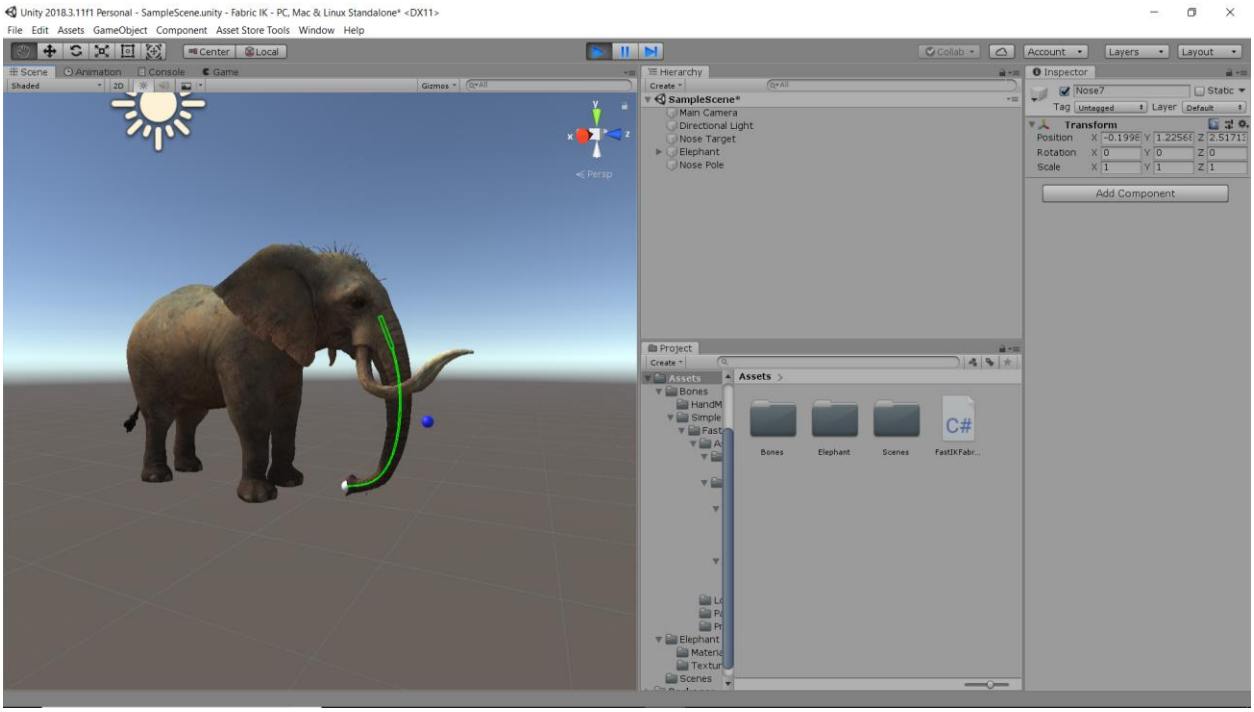

Figure 22: generate animation.

6) play animation

Play the animation to animate the 3D model as required as shown in the following figure.

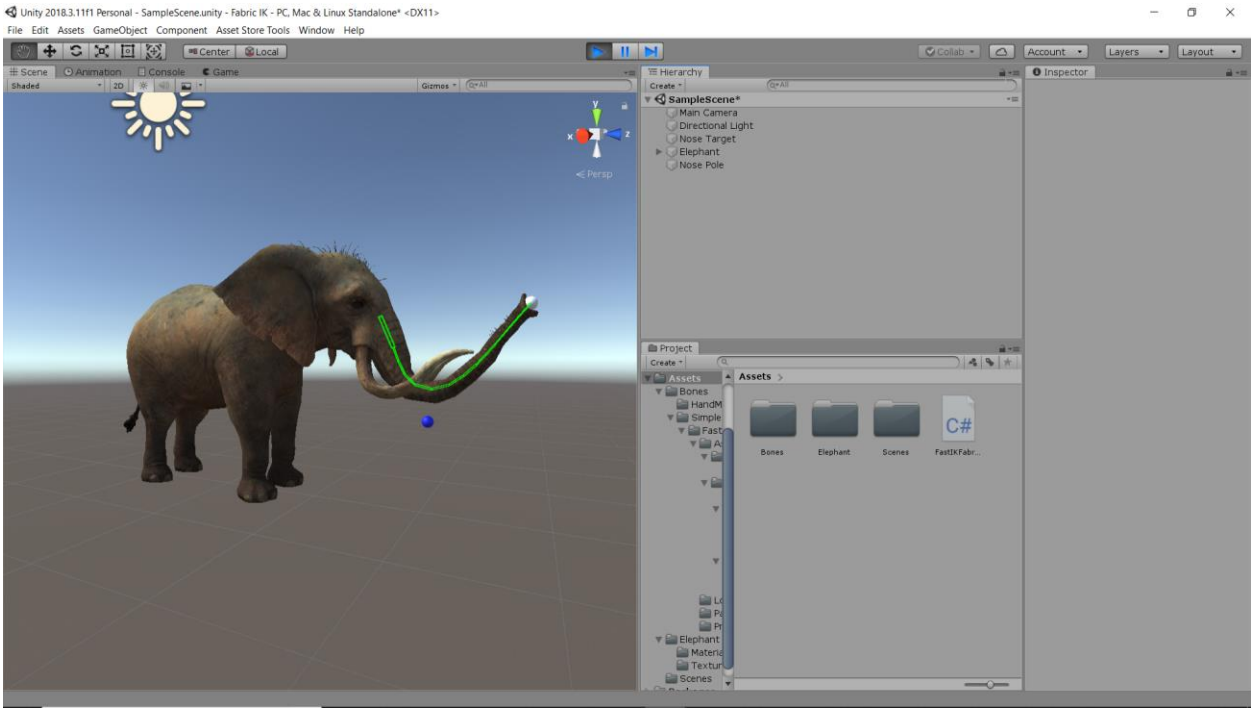

Figure 23: play animation.

7) Check the animation realistic

Check the animation of the 3D model if it is smooth and realistic or not. If animation is realistic keep playing animation else regenerate animation of the 3D model until the animation be as required.

Chapter 4

System Implementation

4 System Implementation

Introduction:

In this chapter, we will discuss the implementation of the system and what we went through to reach our goal. In the first section, we will explain the phases of the system development phase by phase to show what we went through, what we used, and what knowledge we gained to reach the last phase. In the second section, we will explain our system overview and each component in the system and also we will explain our class diagrams of each algorithm or code used in our system. In the third last section, we will explain our system running, and what are the inputs and outputs of each component to reach our system goal as planned.

4.1 System Development

4.1.1 First Phase

At the first phase of the system, we implemented the inverse kinematics in the 2D form to understand the concept of the algorithm and how it works. Where it was a small demo for the inverse kinematics depend on the graphics in multimedia (Ex: draw line and draw circles). This demo will show us the mechanism of inverse kinematics and give us a start on how to implement it for 3D models in 3D worlds. The 2D inverse kinematics is depend on joints that are connected to form a model structure. As shown in the following fig.24, that show use how the joints are connected (the red circles are the joints and the white lines are the bones for example).

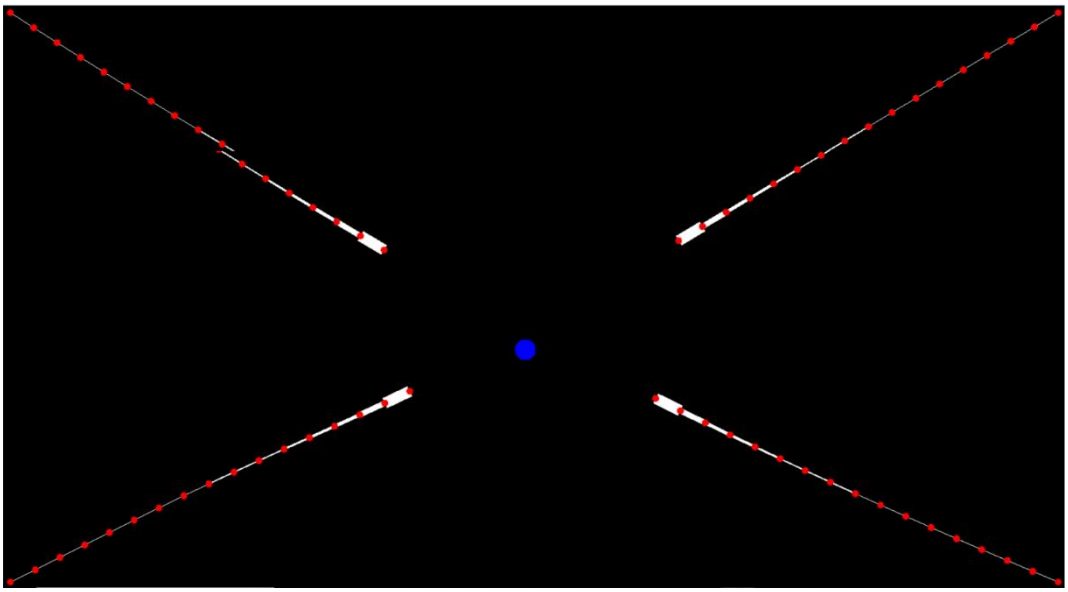

Figure 24: 2D IK algorithm

So by calculating the parameters of this joints, it will consist a kinematic chain such as robot manipulator in a specific orientation and position relative to the start of the chain. The joints has a base and end effector, the base joint is attached joint for a specific position and end effector joint keeps trying to catch the desired at a specific position (the desired is the blue circle). So each 2 joints create segment or armature from the base joint to the end effector joint since it is a parent child relationship (the root is the grandparent and its child is a parent and so on till reach the end effector). Each armature has a position, length, and angle, where the angle is calculated to make the root armature at the base joint points at the target and try to reach it and also the children trying to do the same thing so that our system of ridged length fits and reach the goal which is the target as shown in the following fig.25.

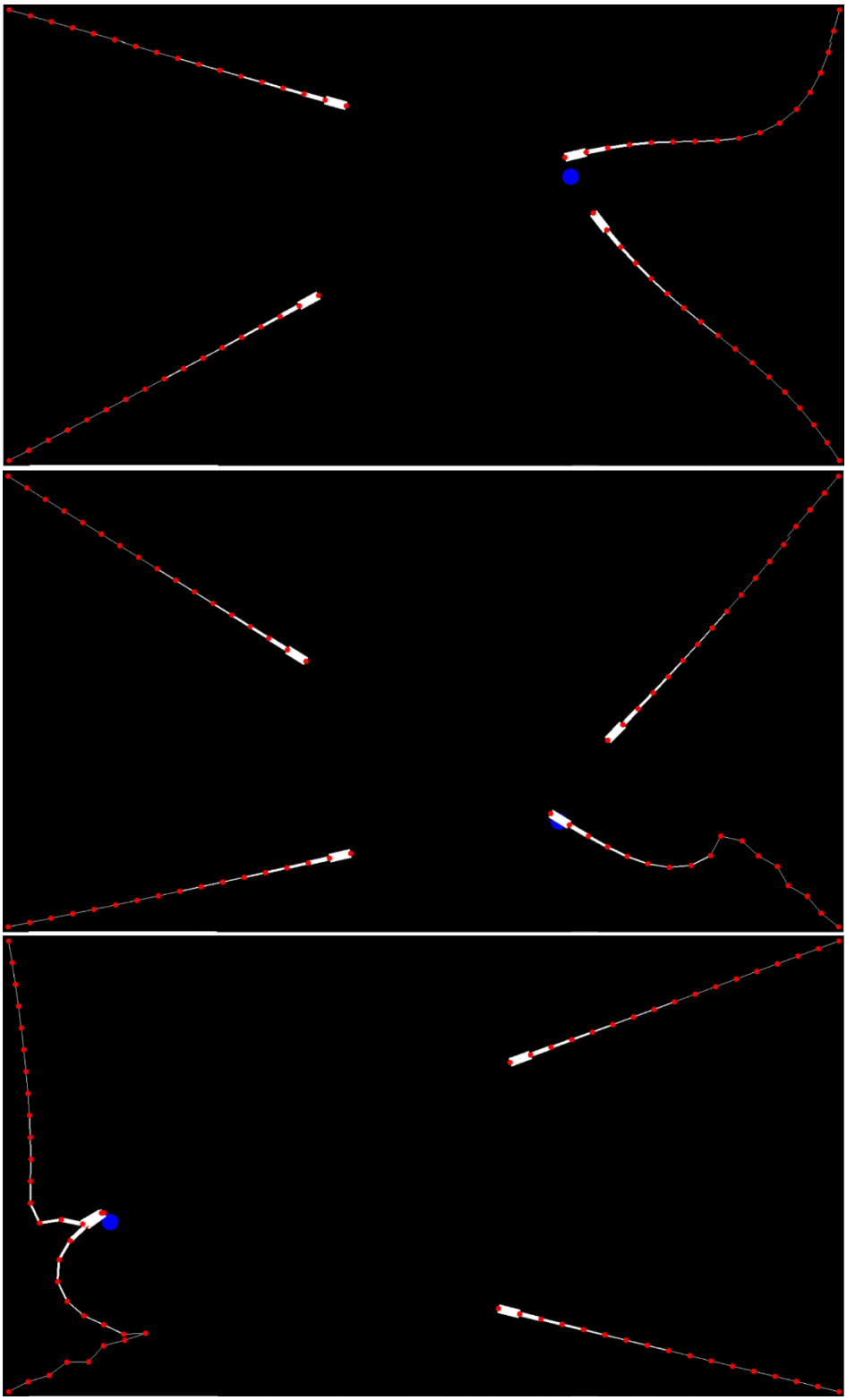

Figure 25: Joints created arms that follow blue circle.

The used platforms: In this phase we used Visual Studio C# to implement the algorithm. The libraries of the Visual Studio C# that we used are C# system, window forms system, drawing system.

4.1.2 Second Phase

By the aid of the first phase, we knew about the inverse kinematics and its solvers (algorithms that helps in DOF problem in the inverse kinematics) and we implemented a 3D inverse kinematic solver which is called "Forward and Backward Reaching Inverse Kinematic" (FABRIK). FABRIK algorithm works on a chain of 3 joints at least (that means the length is at least 2 bones) and the first joint is considered to be the root (grandparent) and the next joint is its child till reach the end effector (this means that it is a parent to child relationship). So FABRIK works with 2 steps the first one is called forward and the second one is called backward, and those 2 steps are iteratively repeated till the end effector joint reach the target. The backward step is starting from the end effector joint and make it reach the target and then get the normal between it and its parent multiplied with the given length between them and do this till the root joint. The forward step is starting from the base (root) and get the normal between it and its child multiplied with the given length between them and do this till the end effector joint. Then complete do forward and backward iterations repeated till we get significantly closer to the target. As shown in the following fig.26, the 5 white spheres (5 joints) are representing 4 bones where the last one on the left is the base (root) joint and the last one on the right is the end effector joint, and the blue sphere is the target, the red sphere is the pole, and the green rectangles are the bones where each rectangle is a bone consist of 2 joints.

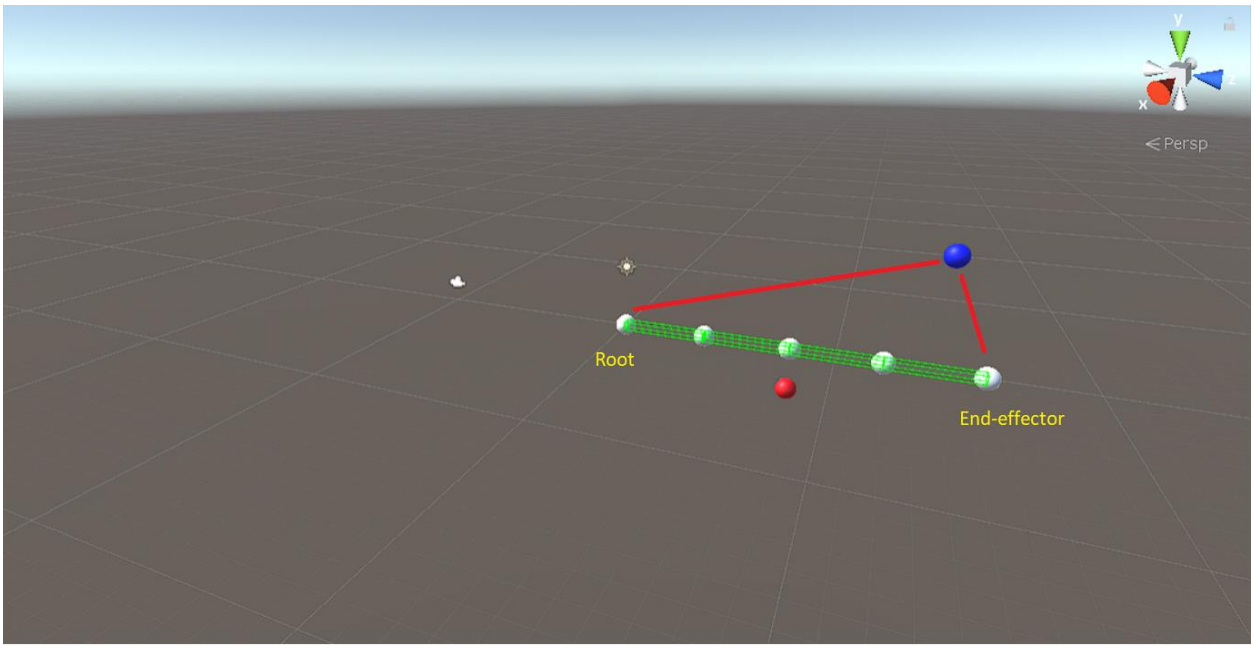

Figure 26: Applying IK on 5 spheres

The end effector keeps trying to reach the target (blue sphere), and the intermediate joints are attracted and biased to the pole (red sphere) as shown in the following fig.27.

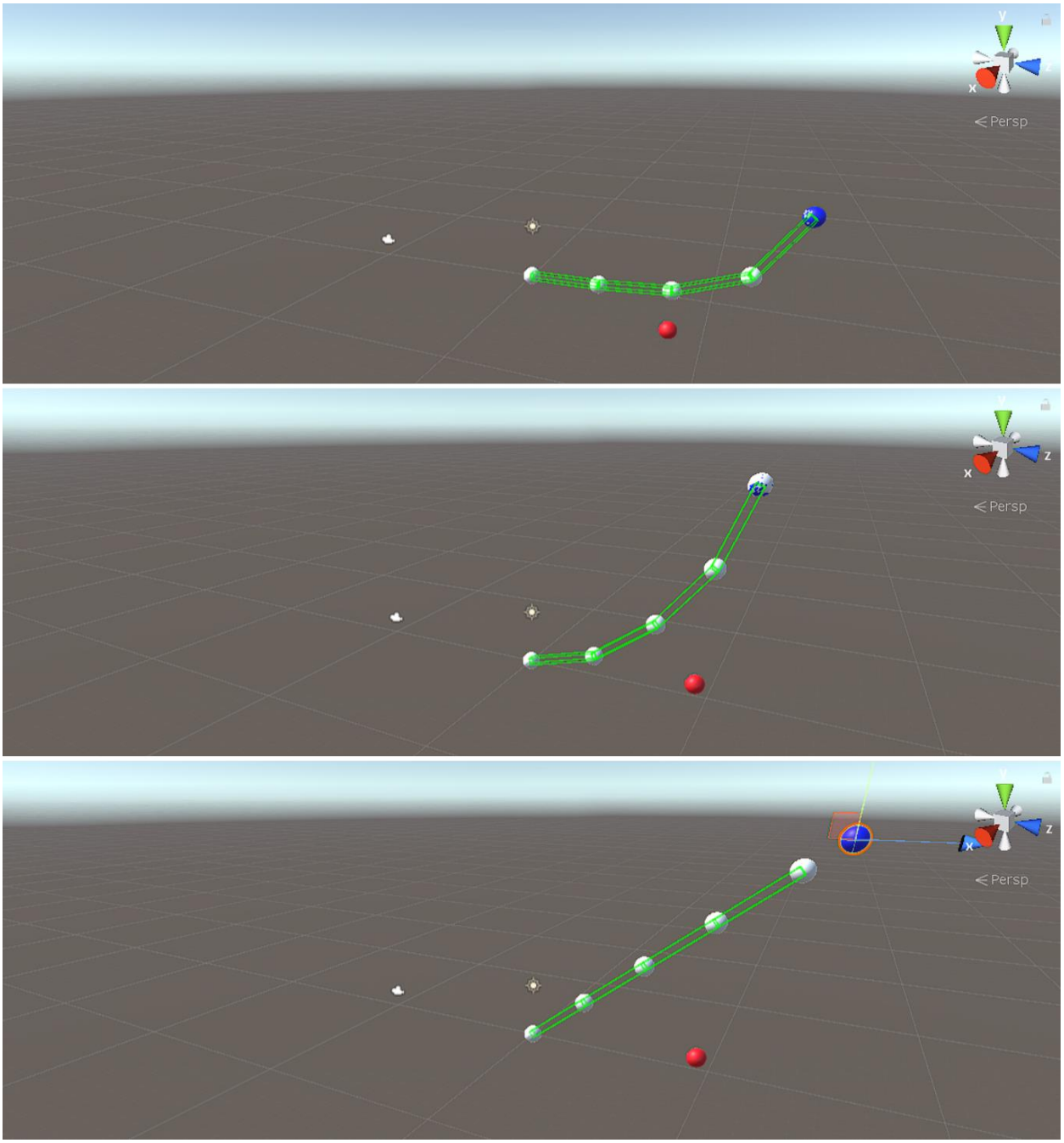

Figure 27: Moving the Target and the End-effector trying to reach it

The used platforms: In this phase we used Unity as a platform and Visual Studio C# in Unity to implement the algorithm. The libraries of the Visual Studio C# in Unity that we used are Unity Editor, Unity Engine, and System Collection.

4.1.3 Third Phase

In the third, phase after we implemented the algorithm and understood its mechanism and how it works, we applied it on a 3D mesh (Ex: Elephant). We applied the script on the end effectors of the mesh and inserted the chain length, the target, and the poles for each desired end-effector. The blue spheres are the targets, the red spheres are the poles, and the green rectangles are the bones where each rectangle is a bone consist of 2 joints.

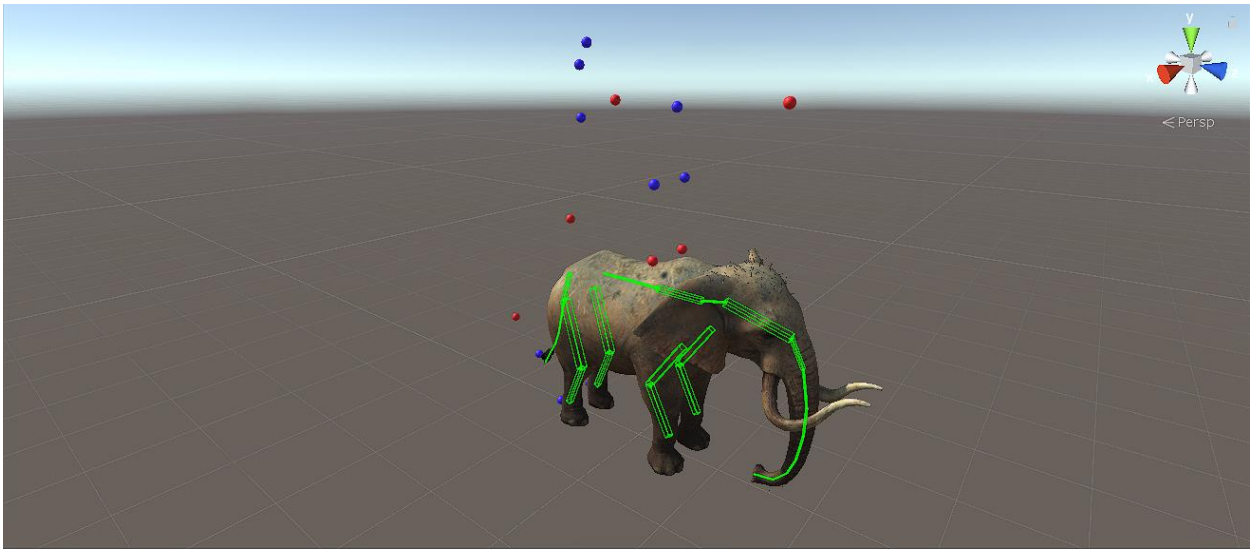

Figure 28: Applying FABRIK IK algorithm on 3D model.

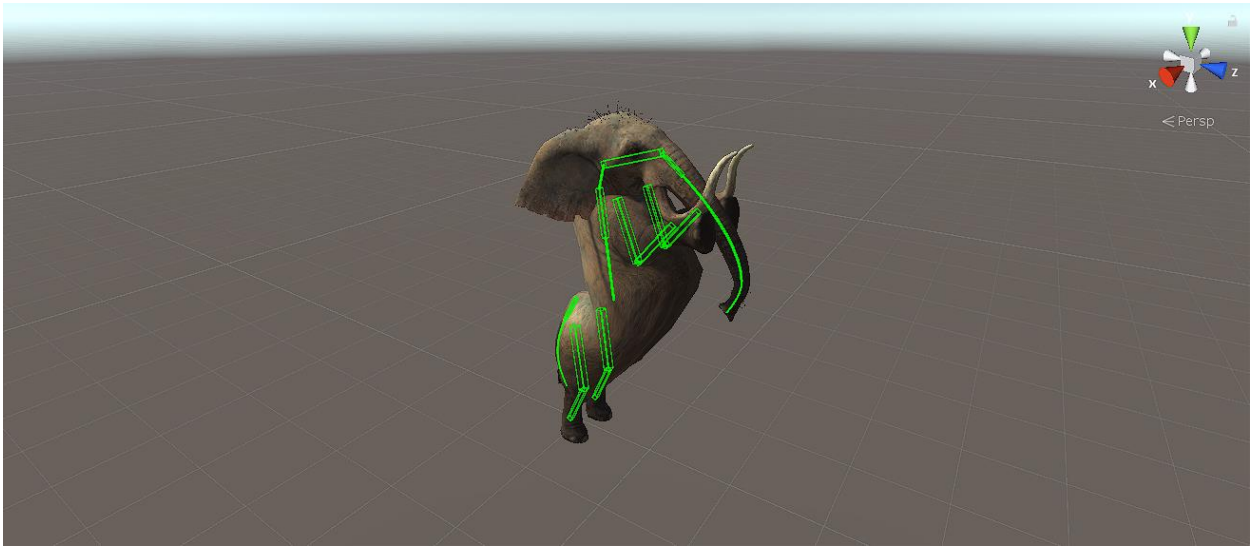

Figure 29: Run Unity to execute the FABRIK script.

The used platforms: In this phase we used the same platform as in the second phase, where we used Unity as a platform and Visual Studio C# in Unity to implement the algorithm. The

libraries of the Visual Studio C# in Unity that we used are Unity Editor, Unity Engine, and System Collection.

4.1.4 Fourth Phase

At this phase, we tried to map 2 same models and 2 different models. At the 2 same models, we mapped two elephants of the same structure as shown in the next figure, the first one is applied with FABRIK algorithm, and the second one without FABRIK algorithm. At the 2 different models, we will us two models with different deformation structure and try to map their joints smartly. By mapping two models, we will be able to make the second model to animate as the first model manually by using script.

Figure 30: Mapping animation of a model use FABRIK IK with another model.

4.2 System Structure

4.2.1 System Overview

As shown in the following fig.28, our system at the 2nd and 3rd phases is composed of 7 main stages, at the 1st stage we import a 3D asset of a FBX format with its textures and materials in Unity 3D world and set the model (asset) in a position in the 3d world. At the 2nd stage we add the IK script (FABRIK algorithm) on each desired end-effector joint in the model as a component in Unity. At the 3rd stage we set targets to each end-effector assigned to the script and give these targets a specific position according to its end-effector. At the 4th stage we set poles to a specific joint effected by end-effector that assigned to our script and give it a specific position according to the joint to act as desired. At the 5th stage we apply the algorithm on the model by execute and run Unity platform, so the model will take the pose according to the targets and the poles that effects the end-effectors in the model. At the 6th stage we generate and create animation by unity animator controller and give the targets and the poles a specific positions in the 3D world that will create a pose for each frame according to the position of the target and poles, so these frames are stored in animation component, where each frame is assigned to a specific time (frame/second). At the 7th stage we play the animation and see if the animation is acceptable or not. So if the animation is acceptable, we keep playing the animation or terminate and stop Unity execution, else we return to 3rd stage and try again to make the animation better.

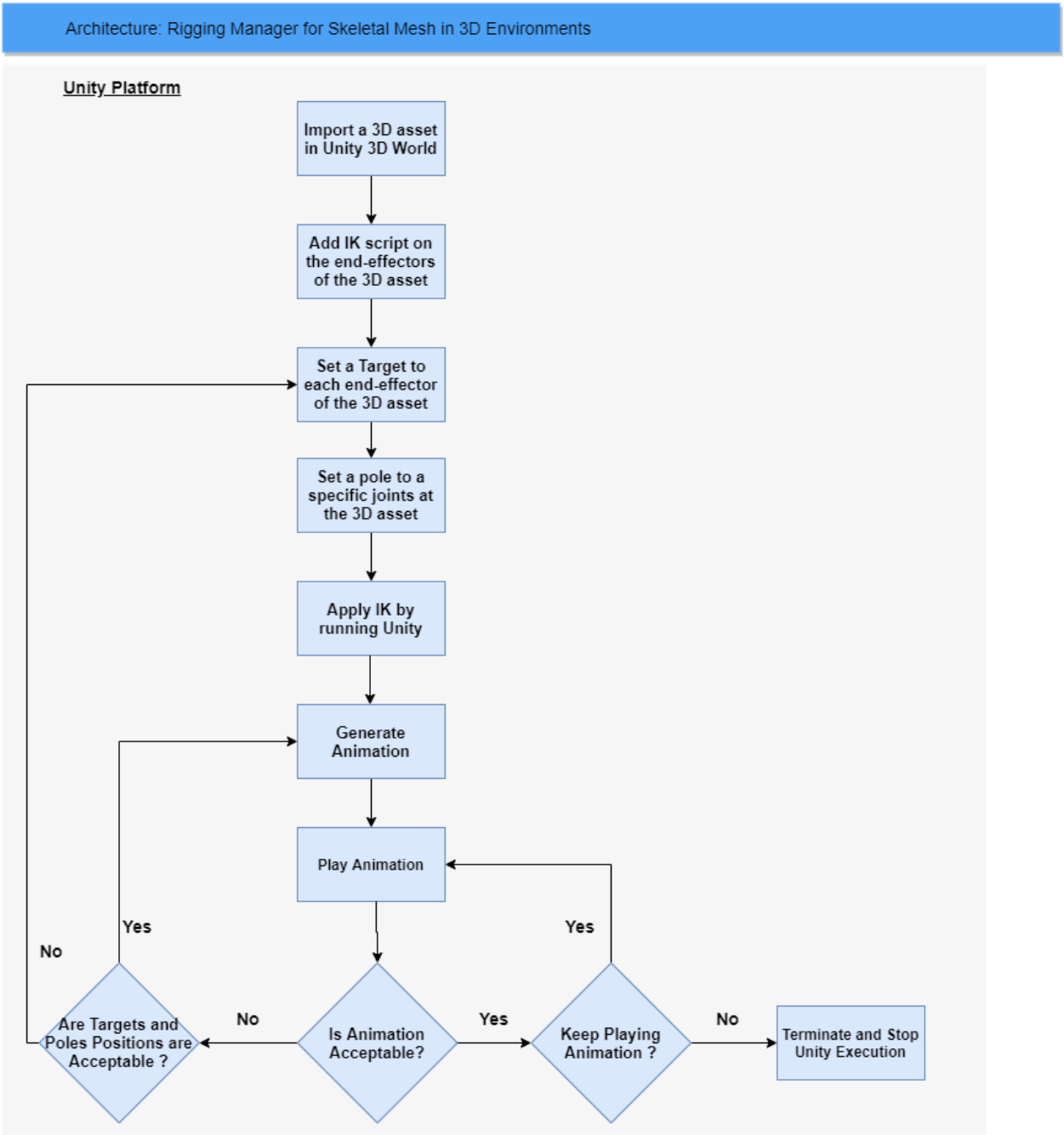

Figure 31: A system overview of how the system works

4.2.2 Class Diagram

FABRIK class diagram:

At the start, our script take from us a Chain Length, a target, and a pole. Then the awake method calls the initializing method to initialize the numbers of Bones (joints), the length of each bone, then search for the parent joint which is the root and check if number of joints equal to the number of chain length plus one and then check if the target is exist. At last it set the transform of the end-effector and intermediate joint to the base (root). After the awake method is called, the late update methods is executed and call two methods every frame, the initializing method and ResolveIK method. In the ResolveIK at each frame, it checks if the target is exist or not, then get the positions (transform) of the joints (bones) every time the method is update as the target or the pole might move, then check if the length from the root to the target less than or greater than the length of the whole chain (bones). So if the length of the chain less than the length from the root to the target just stretch the chain as shown in the last image in figure (27), else the algorithm make two techniques (forward and backward reaching inverse kinematics). Firstly, the backward reaching inverse kinematics technique which permit the end effector to reach the target then the parent joint of the end-effector reach the end-effector position but we minus the saved length between them (bone length) and the intermediate joints do the same thing till the technique reach the joint in front the root joint. Then make the forward reaching inverse kinematics technique which start from the joint which is in front the root joint and let this joint reach the position of the root joint but we plus the saved length between them and the intermediate joints do the same thing till the technique reach the end-effector joint, then check if the end-effector is very close or reached the target according to delta which is the distance between the end effector joint and the target (delta approximately equal to 0) and if the distance between them equal to zero or less than or equal to delta stop completing the iterations else, the two techniques are repeated for a given number of iterations till it reaches the perfect pose according to the target, where number of iteration is 10 (most of the developers approximately choose 10 iterations). Then check if the chain has a pole that is set to a specific position. In the following figure 31, we show how the pole effects the chain. Firstly, we create a normal vector from joint i-1 (for example joint i-1 is the root) to joint i+1 (for example joint i+1 is the end effector), then create a plane where its center is the root, then make a shadow for the pole, the joint I, and

the joint i+1 on the plane. Then the shadow of joint I on the plane is rotating around the shadow of joint i+1 on the plane according to the position of the shadow of the pole on the plane, then set these new values on the real joints according to their position. Then set new positions to the bones according to the calculations made by the two techniques and the pole position.

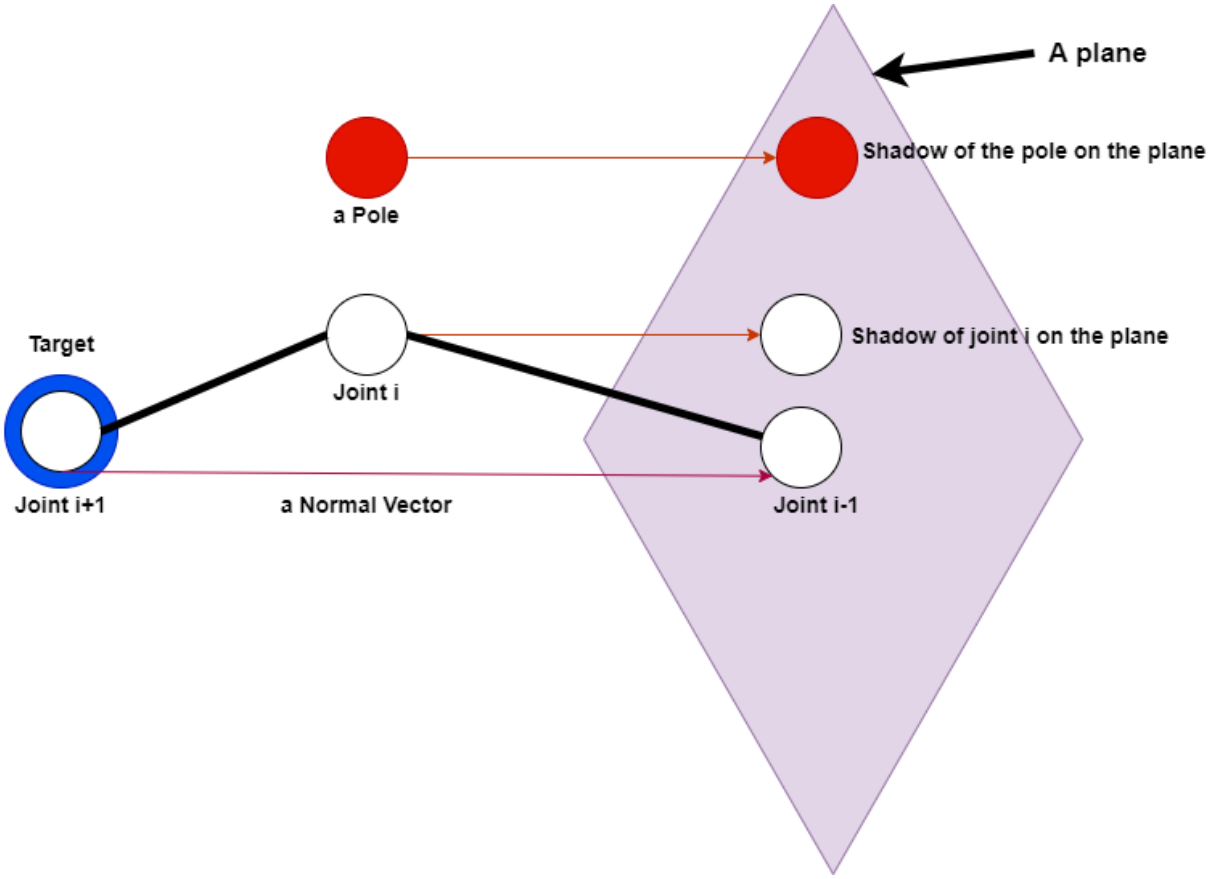

Figure 32: Shows the pole methodology and how to create it and how it works.

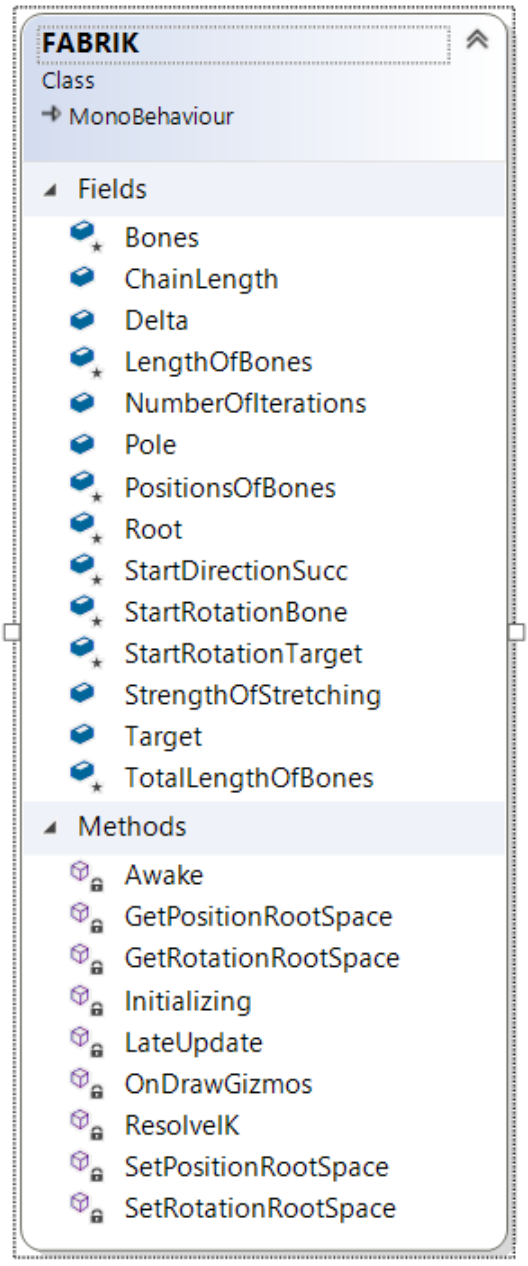

Figure 33: A Class diagram of the FABRIK IK algorithm in the system.

Animate Class Diagram:

When the system starts to run, the animate class starts to run too, where the animate class take animator controller which contains the 3D model animation and play this animation when click on a button declared in the update method, which is called every frame of the runny. And when we click on a specific button the 3D model animation starts to play every time we click that button.

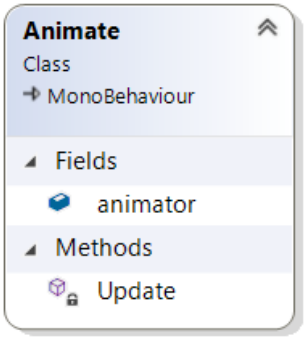

Figure 34: A Class diagram of the animator class in the system.

CollectEAnimation Class Diagram:

This class is used to take joints of the model at each frame, where the model applies the FABRIK IK algorithm, then take these joints and save it in the list E, and then apply it on another model of the same structure (the same elephant model) which is not using FABRIK IK algorithm. After that the another model will animate as the fi

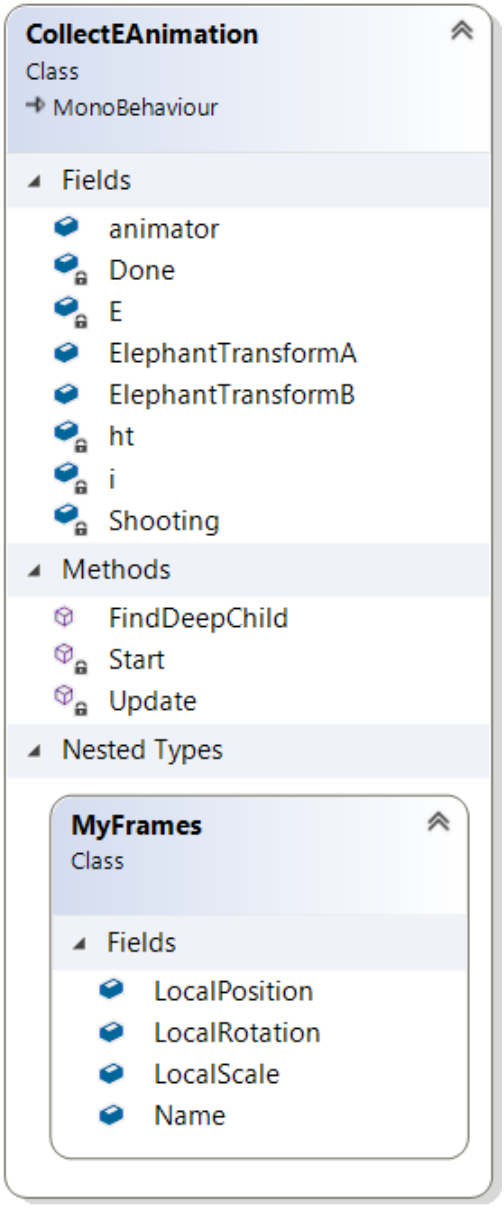

Figure 35: Mapping Class Diagram

4.3 System Running

4.3.1 1st Component

At our 1st component we import a 3D asset or assets to Unity "Assets Folder" and this could happen with 3 basic ways, the first way is to drag and drop the 3D asset inside the "Assets Folder" in the "Project Window", the second way is to download and import the 3D asset from Unity "Asset Store" and it will be automatically added and imported in the "Assets Folder", and the third way is to add a copy of the 3D asset in the "Assets Folder" in the Unity "Project Folder" and it will be also added in the "Assets Folder" automatically. Then we drag and drop the 3D asset to Unity 3D world environment (Scene or Sample Scene) and set it at the position we desired in Unity 3D world as shown in the next figure.

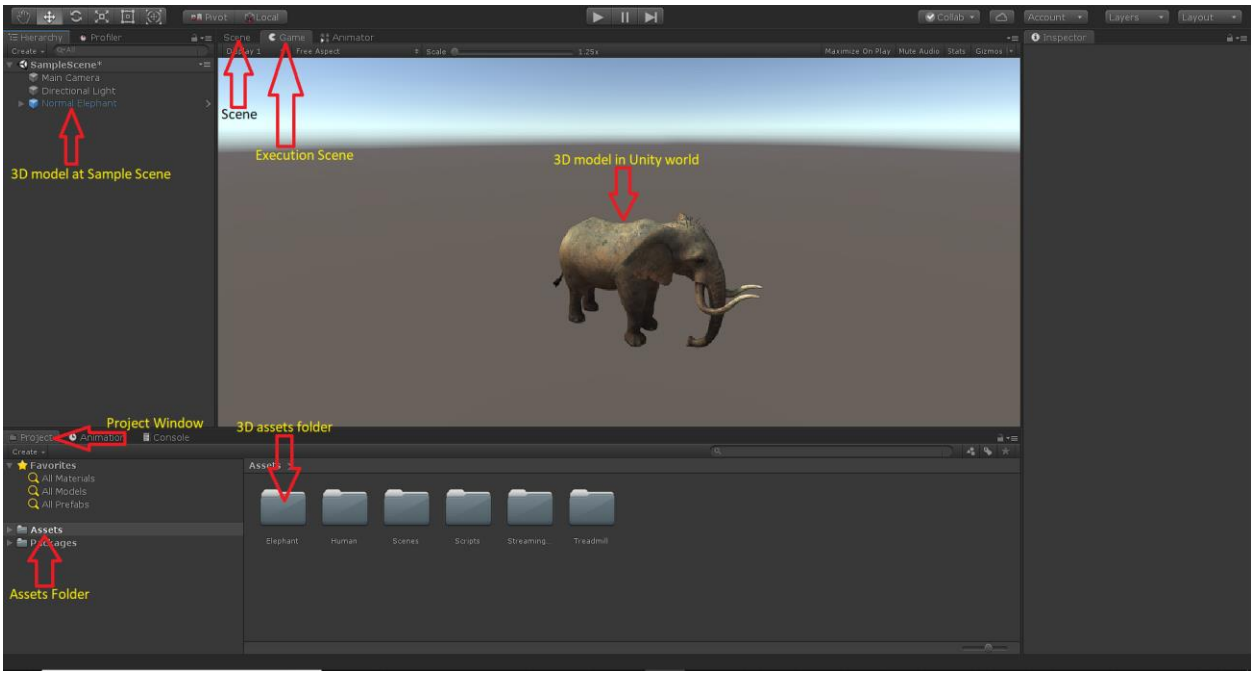

Figure 36: importing 3D asset to Unity platform.

4.3.2 2nd Component

At our 2nd component we add the FABRIK IK algorithm script on each end-effector, which we want to animate or drag and drop the script on each end-effector. Then we give the script at each end-effector the chain length (number of bones). After giving the number of bones, the script will draw rectangles that represents each bone in each end-effector as shown in the next figure.

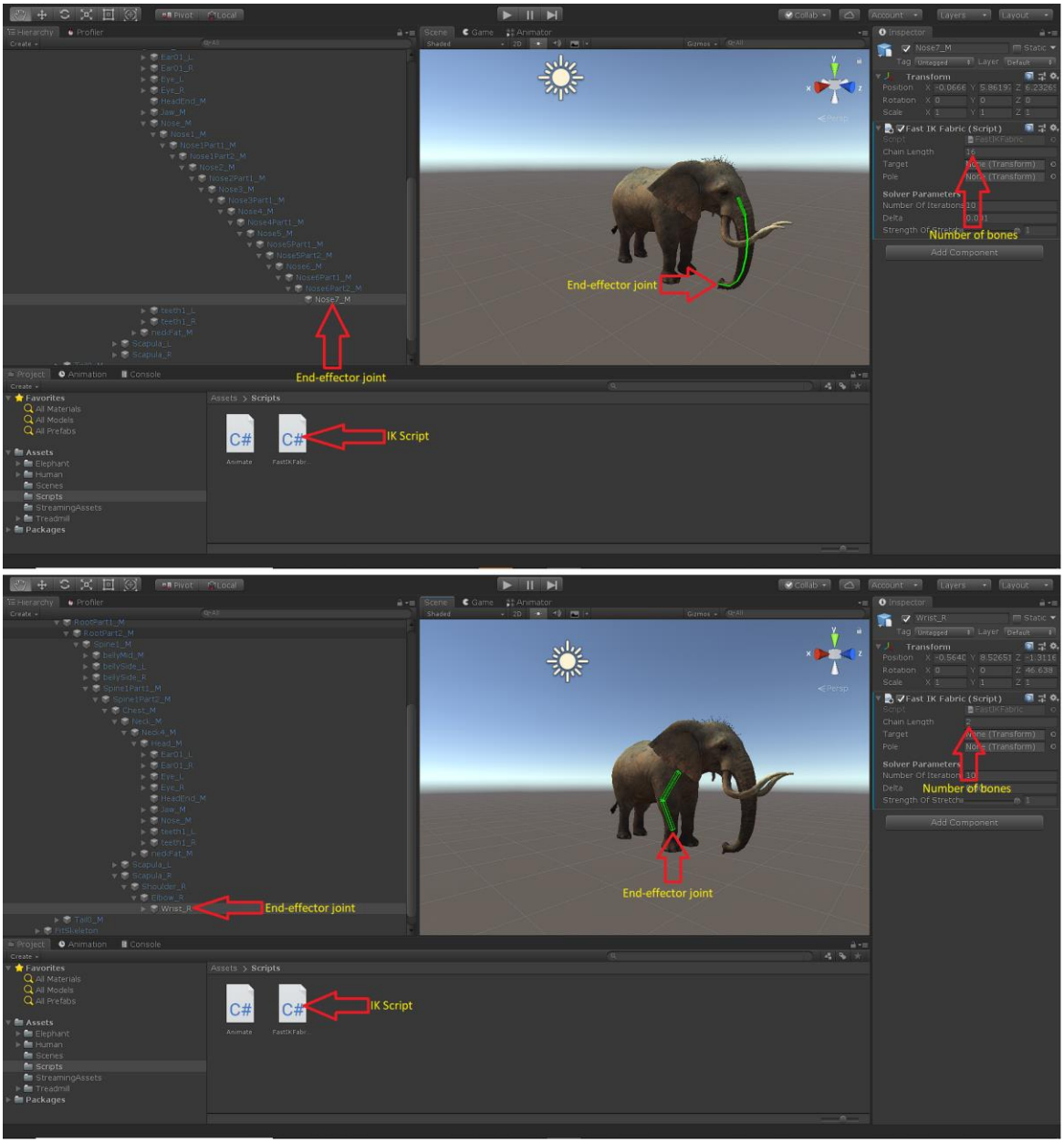

Figure 37: adding the FABRIK script on the end-effctors as a component and setting the chain length.

4.3.3 3rd Component

At our 3rd component, we will add targets to the end-effectors that the FABRIK IK script is applied on, after we position the targets to the desired positions according to the positions of the end-effectors joint as shown in the next figure.

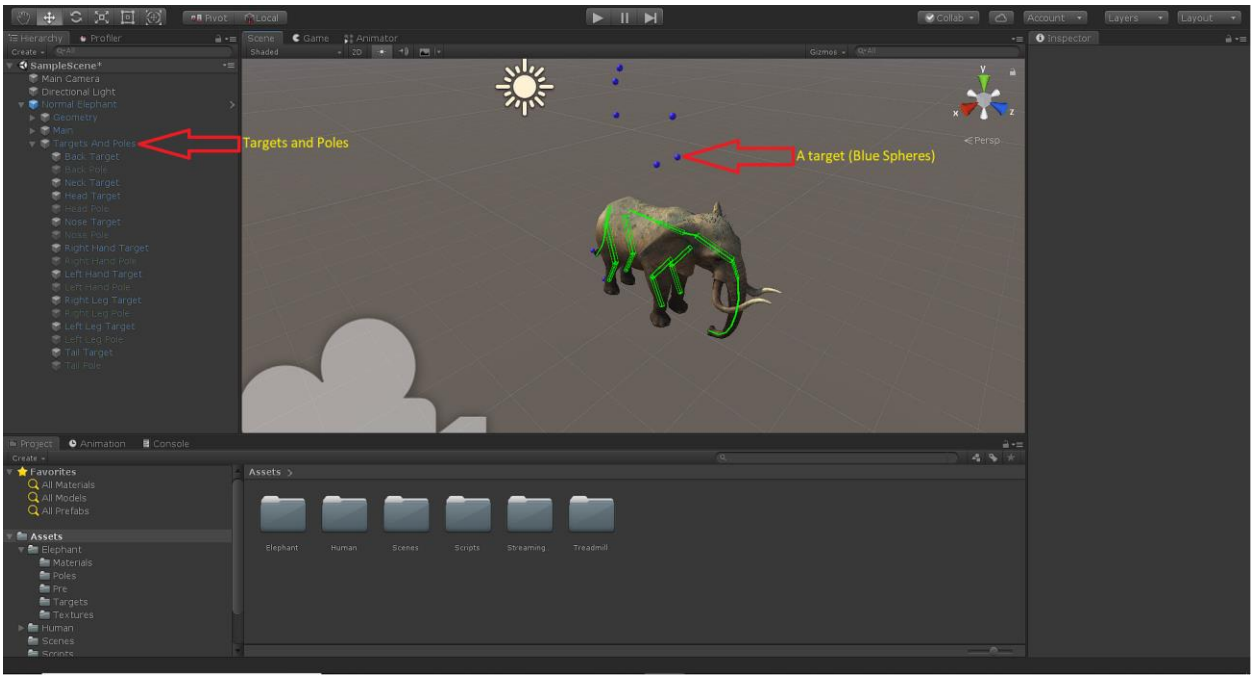

Figure 38: creating and setting the targets.

4.3.4 4th Component

At our 4rd component, we will add poles to the end-effectors that the FABRIK IK script is applied on, after we position the poles to the desired positions according to the positions of intermediate joints and the end-effectors joint as shown in the next figure.

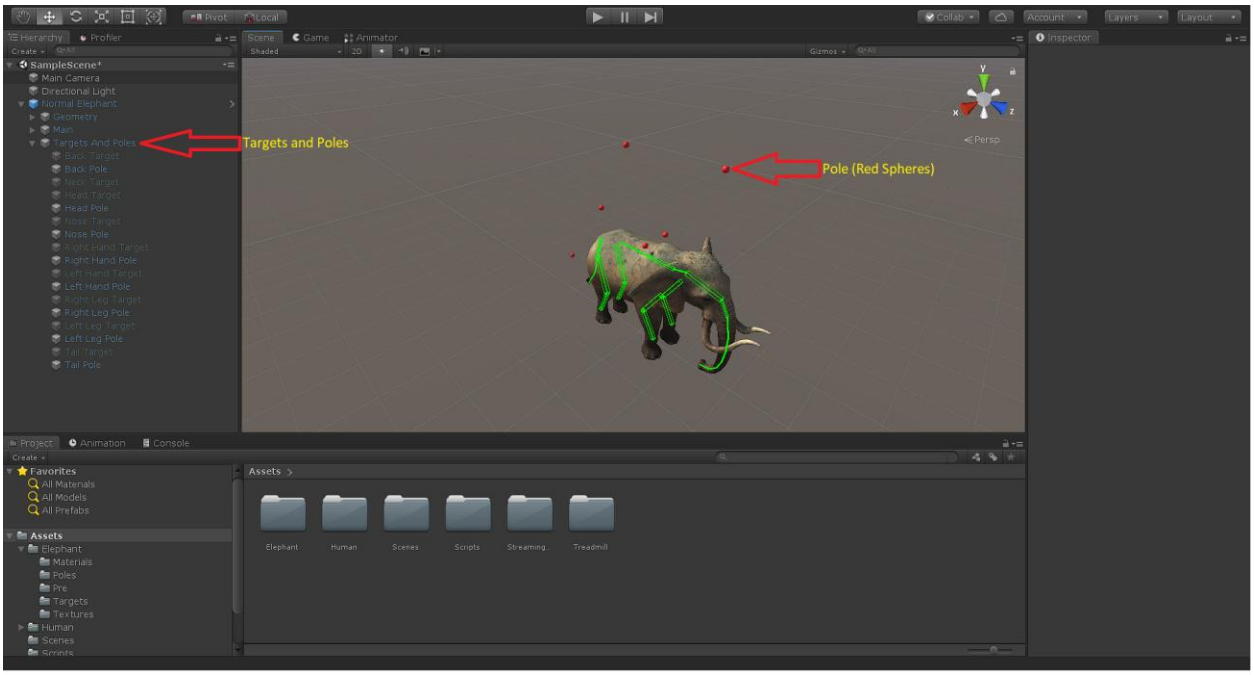

Figure 39: creating and setting the poles.

4.3.5 5th Component

At our 5th component, we run unity to execute the FABRIK IK script which is applied on each end-effector joint, then the 3D model will take a specific pose according to targets and poles positions and number of bones (chain length) as shown in the next figure.

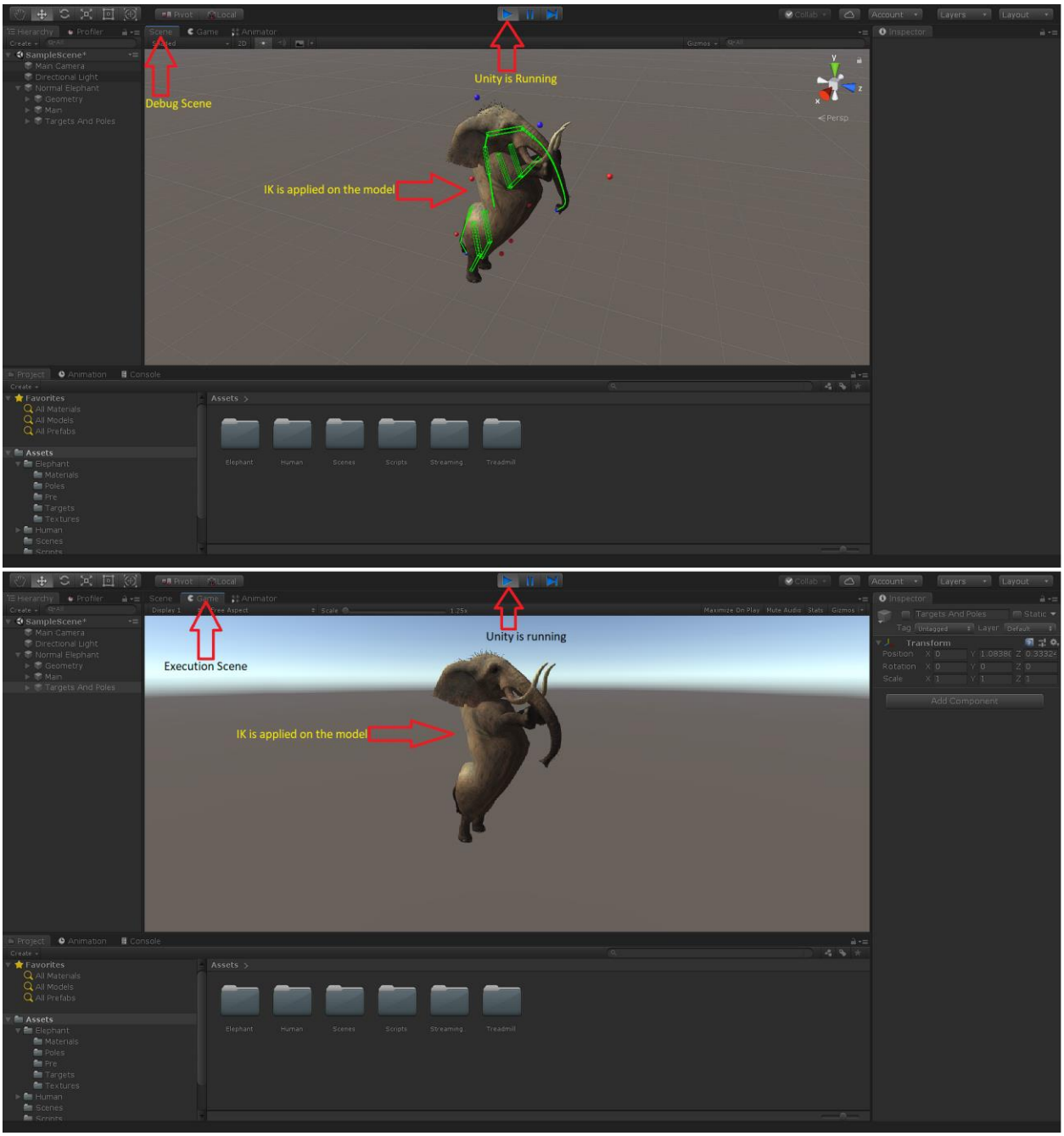

Figure 40: run Unity to apply the algorithm.

4.3.6 6th Component

At our 6th component, we generate our animation by using unity animator control and animation tool. Firstly, we make and create each pose of the 3D model animation and store it in animation component, where we store each pose at specific frame or time (Frame/Second). Then we create animator controller with its parameters as a component in the 3D model to control the animation and play it as desired by a button or key-down by using script that contains the animator controller of the 3D model as a parameter as shown in the next figure.

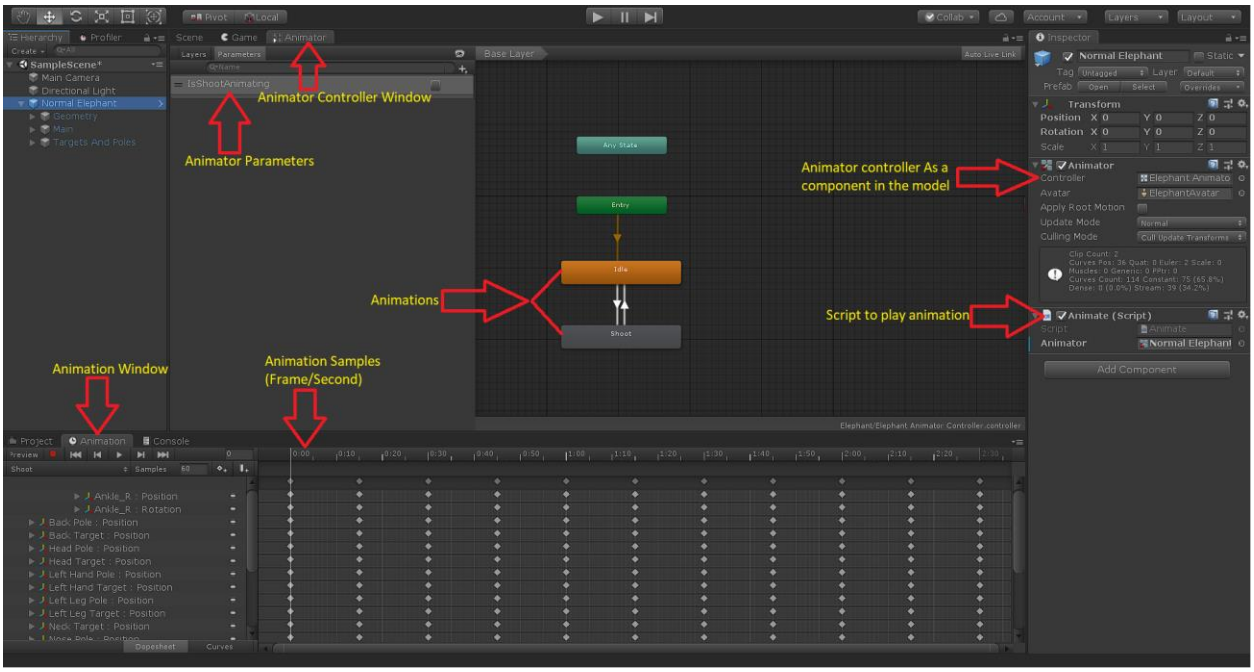

Figure 41: creating an animator controller and animation for the 3D model.

4.3.7 7th Component

At our 7th component, we play the animation by a button or key-down by using script that contains the animator controller of the 3D model as a parameter as shown in the next figure.

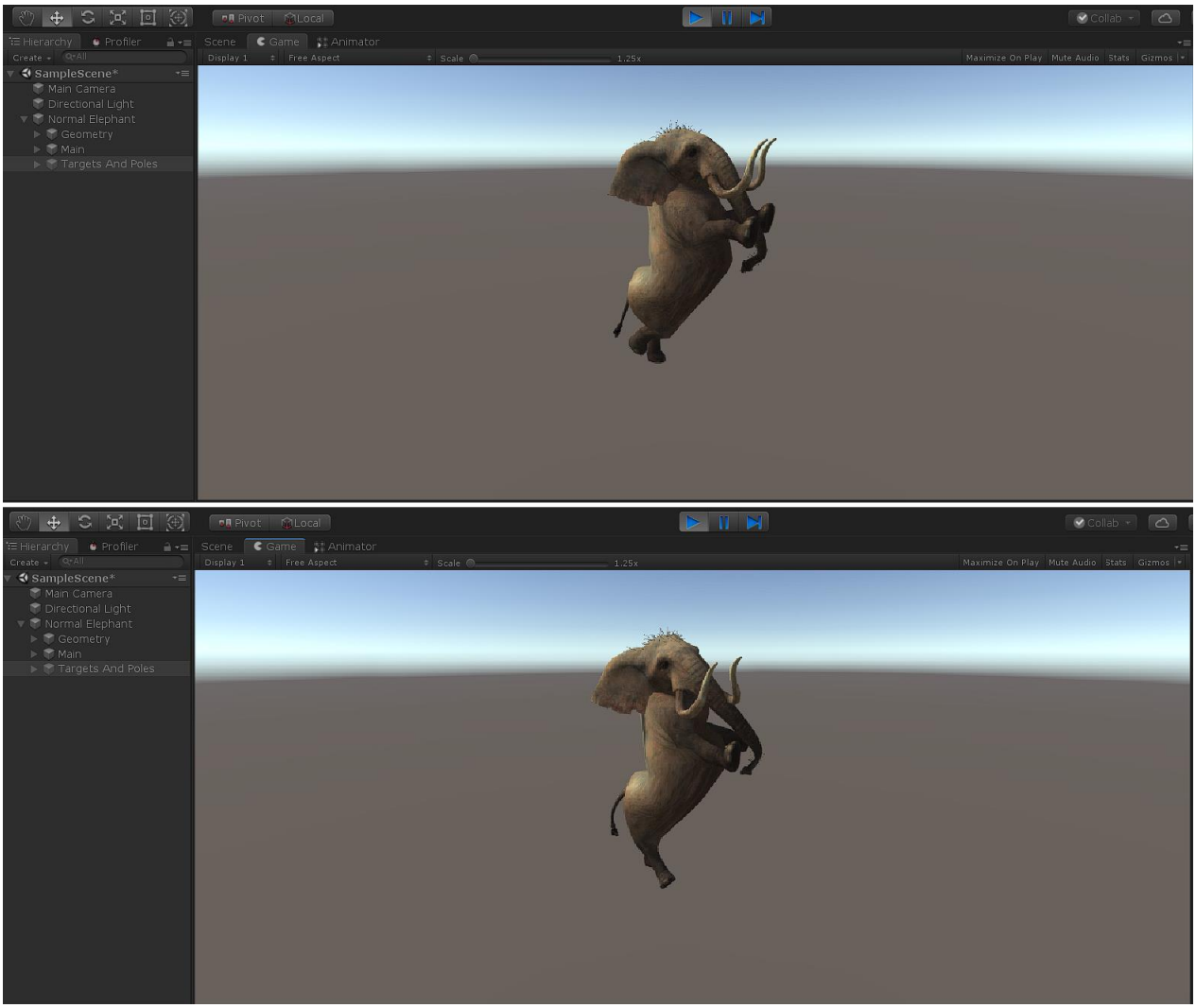

Figure 42: playing the animation of the model.

Chapter 5

Results

And

Evaluation

5 Results and Evaluation

Introduction:

At the previous chapter, we explained our system implementation and how it runs and works. At this chapter we will explain our testing methodology at our system and how we moved through each stage to reach our target and achieve the desired result. Then we will explain our results in case studies (Perfect case, Acceptable case, Worst case). Then we will explain our evaluation of the system and what is the accuracy of our results, the spaced used, the scalability of the data size and the architecture, and the performance of the system at different architectures.

5.1 Testing methodology

Our testing methodology passed through three stages. At the first stage, we applied the FABRIK algorithm on 5 spheres as shown in figure (26) to test if the algorithm works regularly without any errors or bugs, and we tested the movement of the target and how the end-effector will try to reach it correctly as expected. We also added a pole and tested if the intermediated joints are biased to the pole correctly or not to give realistic movement to the rigged spheres as desired. So this will assist and aid us to import this algorithm on a 3D model in our second phase. At the second phase we applied the FABRIK algorithm on the humanoid model with max 3 joint (2 bones) for each rigged body part with end-effector to test if it works well on short chains and we also tested the poles on these joints and it worked perfectly as shown in figure (34). At the third stage, we applied the algorithm on a model consist of more than 3 rigged joints at the part (elephant nose, tail, and back) to test if it works well on long chains, where the end-effectors keep tracking the target smoothly and very realistic and we also tested poles on these joints and it also worked well. After we added the script of the algorithm on each specific desired end-effector that we wanted to move and animated, we gave the end-effector its desired target and a pole for its chain. After we prepared the model, we started to animate the rigged model as desired and if the animation is not smooth and realistic, we keeps trying to reposition the targets and poles of the model before we animate again or at a specific frames, to fix the animation and to make it perfect as wanted and desired.

5.2 Results

5.2.1 Case Studies

1. A perfect result case

   Using a single target and a single Pole for each end-effector:

   When we used a single target and a single pole and positioned them correctly the results where perfect for the 2 bones chain and also for the more than 2 bones chains. As shown in the following figure the pose of the elephant is excellent and perfect and could be animated without any problem but we have to reposition the targets and the poles carefully so we do not ruin the elephant deformation system and keep the elephant model in an excellent shape.

Figure 43: a perfect pose for the model using the FABRIK algrithm.

2. An acceptable result case

   Using a single target and a single Pole for each end-effector:

   In the following figure, we moved target from top to down and let the chain of the elephant nose move through the pole so this ruined the deformation system of the elephant nose and changed the shape of the deformation system of the elephant nose, where this wrong positioning made the shape not perfect but it is acceptable result.

Figure 44: an acceptable pose for the model using the FABRIK algrithm.

3. A worst result case

   Using a single target and a single Pole for each end-effector:

   In the following figure, we moved target from top to down and let the chain of the elephant nose move through the pole and also we positioned the nose target and pole in inappropriate position so this ruined the deformation system of the elephant nose and changed the shape of the deformation system of the elephant nose, where this wrong positioning made the shape of the nose very bad and this result is incorrect and unacceptable.

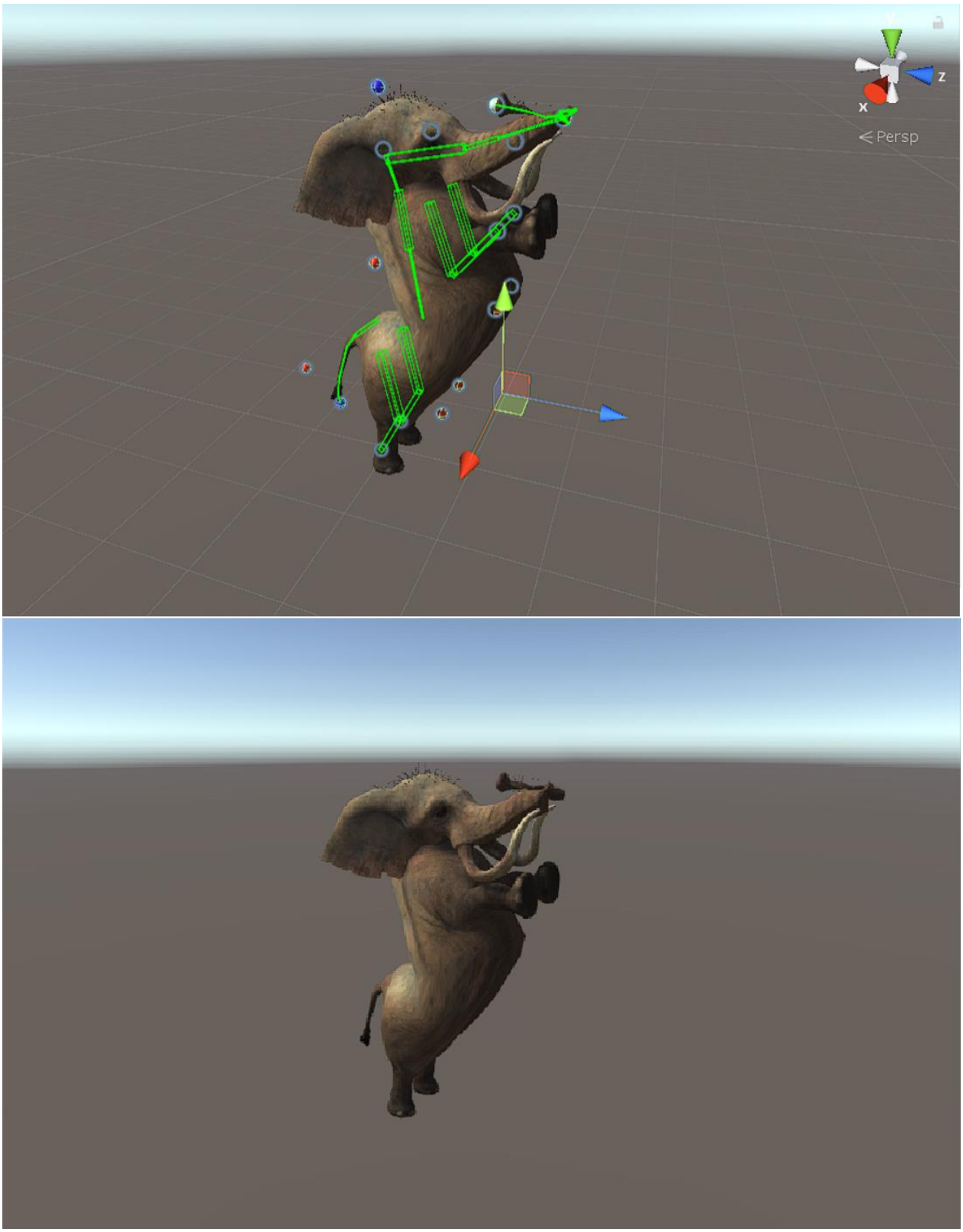

Figure 45: an unacceptable pose for the model using the FABRIK algrithm.

5.2.2 Limitations

1. Tools or Platforms

   Unity:

   a) Unity does not offer a great graphics as other platforms like Unreal

   b) Unity is using C# programing language, which is easily to decompile the code, so we have no safety on our code (low security).

   c) The platform is crashed when a Null references expectations are exist, so this does not allow us to compile the code and we have to reopen the project.

5.3 Evaluation

5.3.1 Accuracy

The accuracy of the system on chain of length less than or equal to five is great as the movement of the chain with the existence of the pole is great and smoothly realistic, but with chain of length more than five, it is very good without the pole but with the pole the deformation of the model is damaged and produce unacceptable results because we need poles equal to the number of intermediate joints or more than one pole to solve this problem.

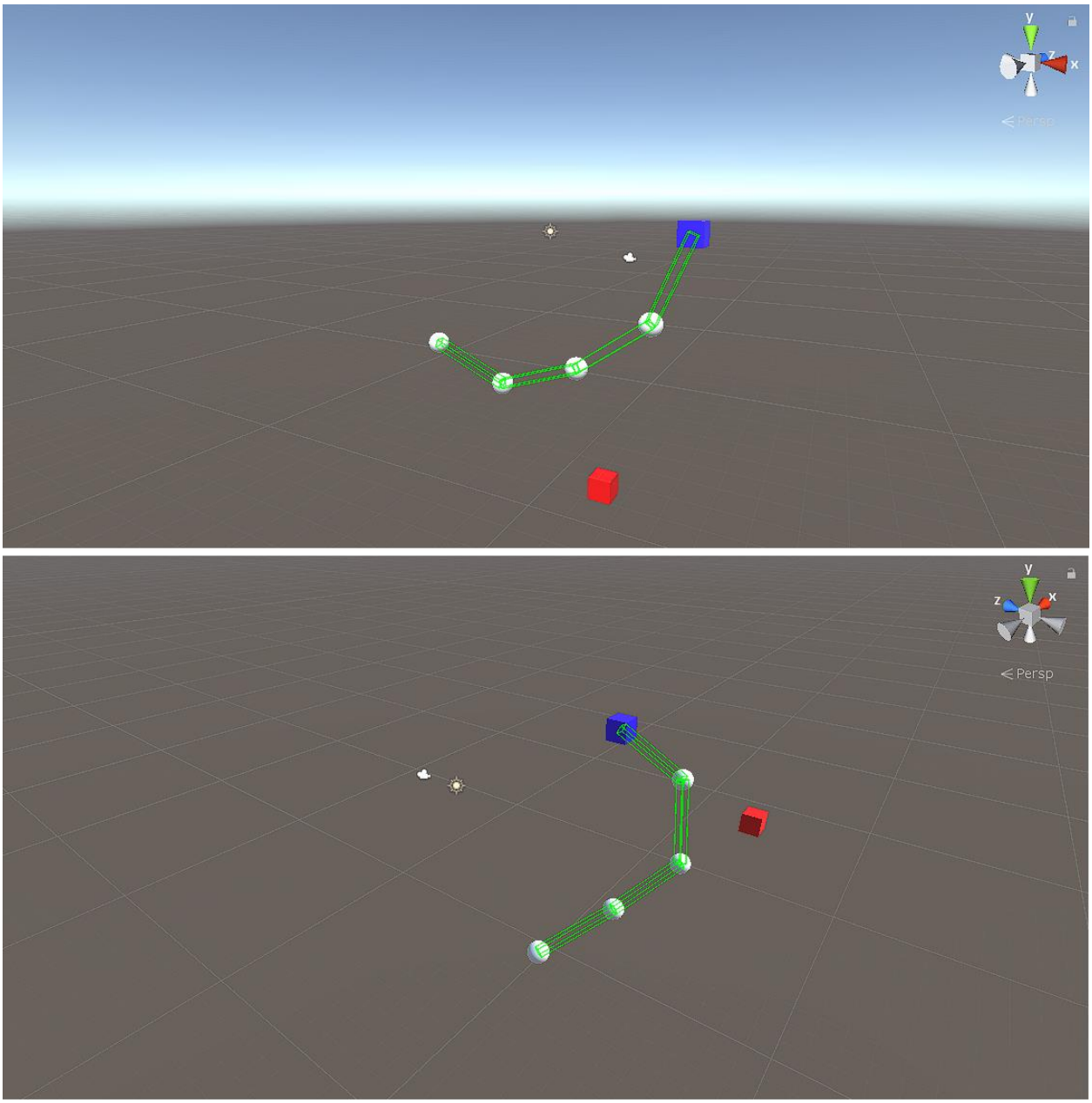

Figure 46: showing the accuracy of FABRIK algorithm on 5 joints.

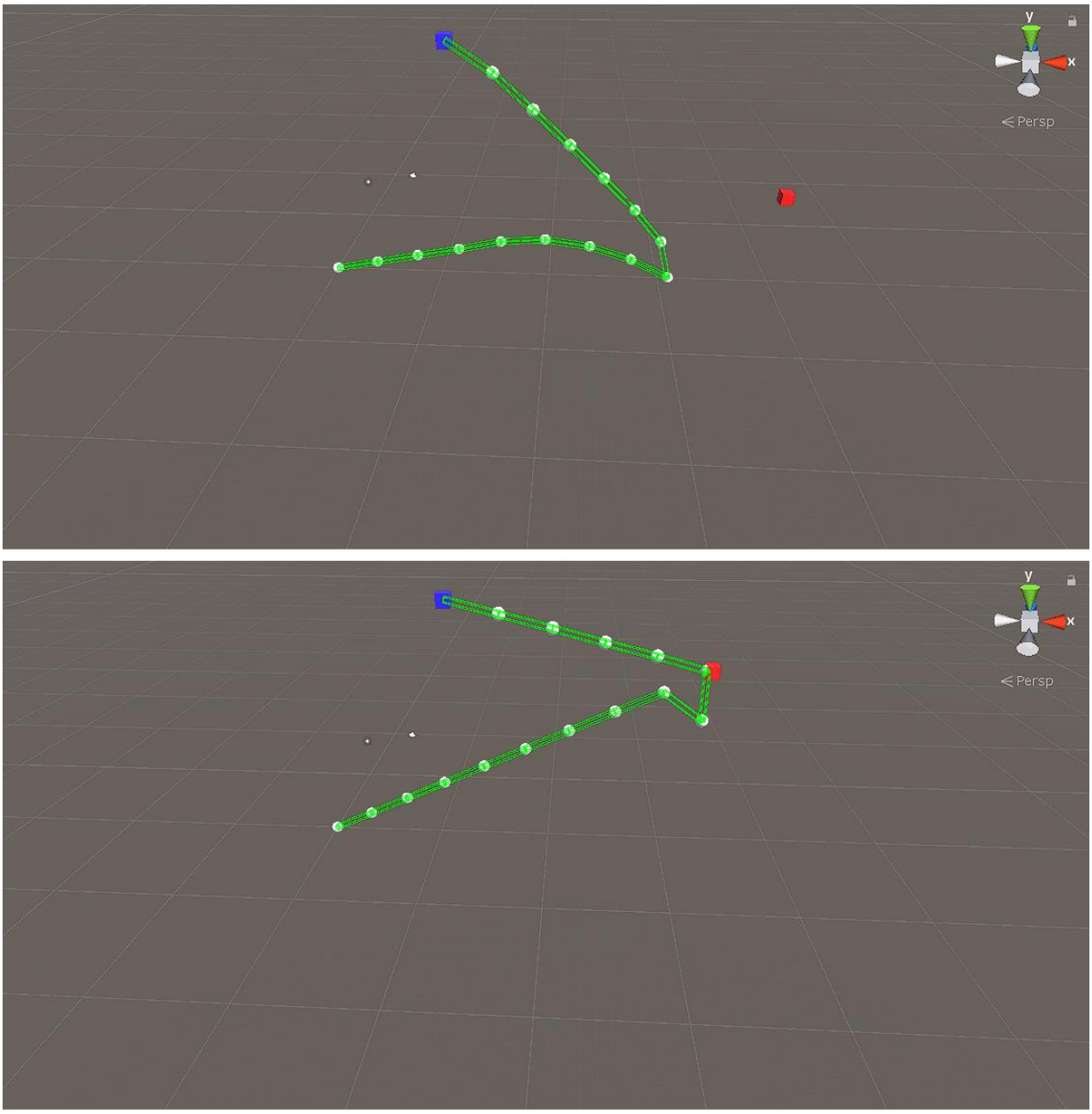

Figure 47: showing the accuracy of FABRIK algorithm on 16 joints.

5.3.2 Space

Degree of scalability for data size:

The FABRIK algorithm and Unity platform can handle any 3D model even if the model size is very huge (5 GB or more).

Degree of scalability for architecture:

Firstly, we tried the scalability of the system on one model (elephant), and the system was fast without any drop in the frame per second, where the system recorded approximately 60 frame per second while the system is running and also while animating and the system was smooth.

Figure 48: show frame per second of the system using one model.

Secondly, we tried the scalability of the system on twenty-two models (elephants), and the system performance is reduced and the frame per second is dropped approximately to the half, where the system recorded approximately 30 frame per second while the system is running and 20 frame per second while animating.

Figure 49: show frame per second of the system using twenty-two model.

5.3.3 Performance

The following two figures show us, the performance of the system on Unity when we used one model at the Unity scene. So the figures shows us the main threads performance, the reading time, CPU usage, Memory usage, UI, and more.

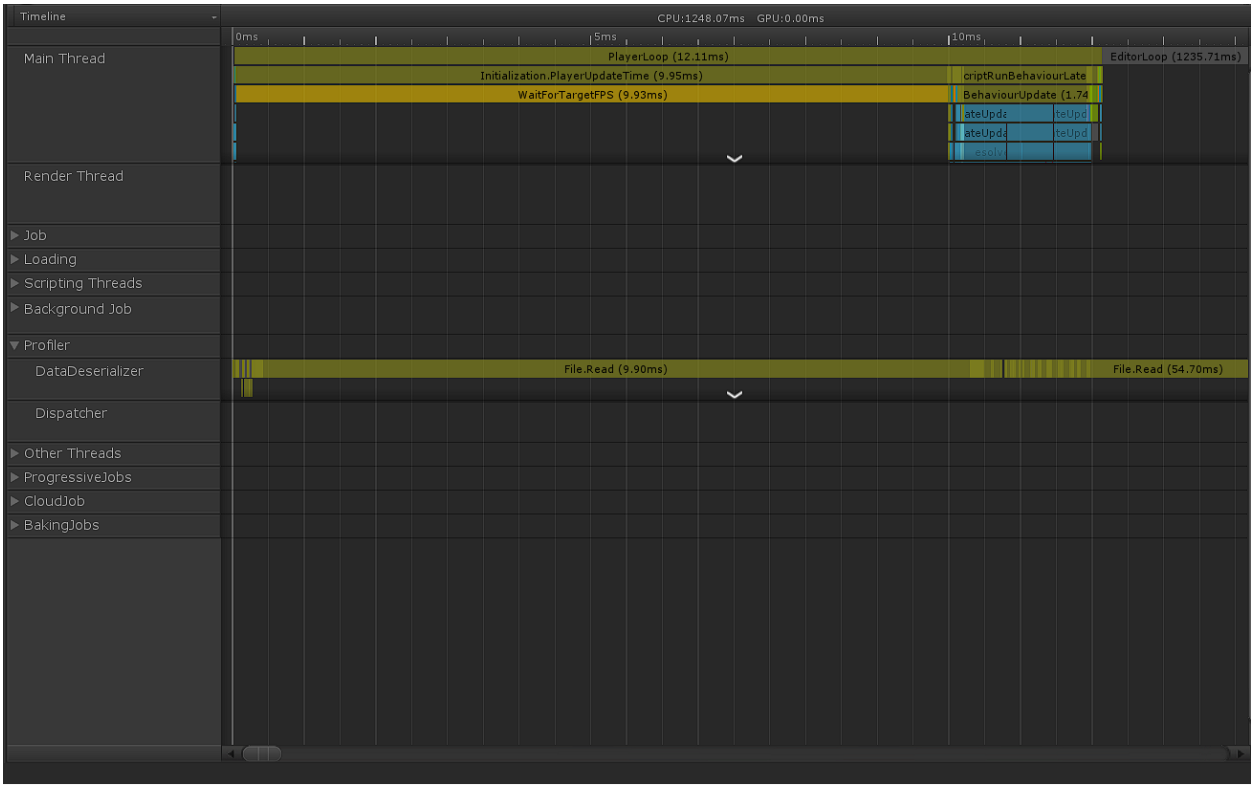

Figure 50: Timeline of the system when we used one model.

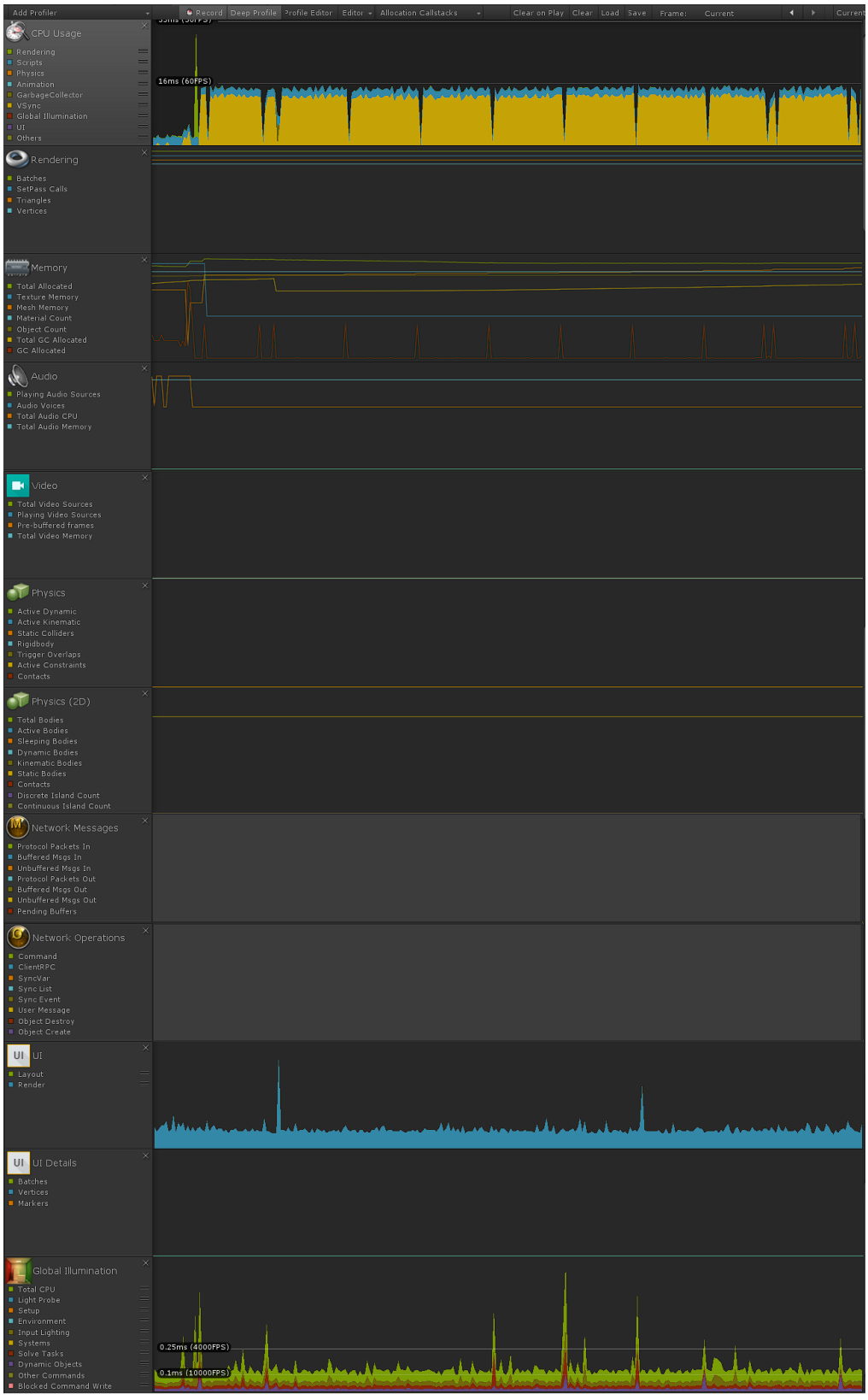

Figure 51: Profiler of the system when we used one model.

The following two figures show us, the performance of the system on Unity when we used twenty-two models at the Unity scene. So the figures shows us the main threads performance, the reading time, CPU usage, Memory usage, UI, and more.

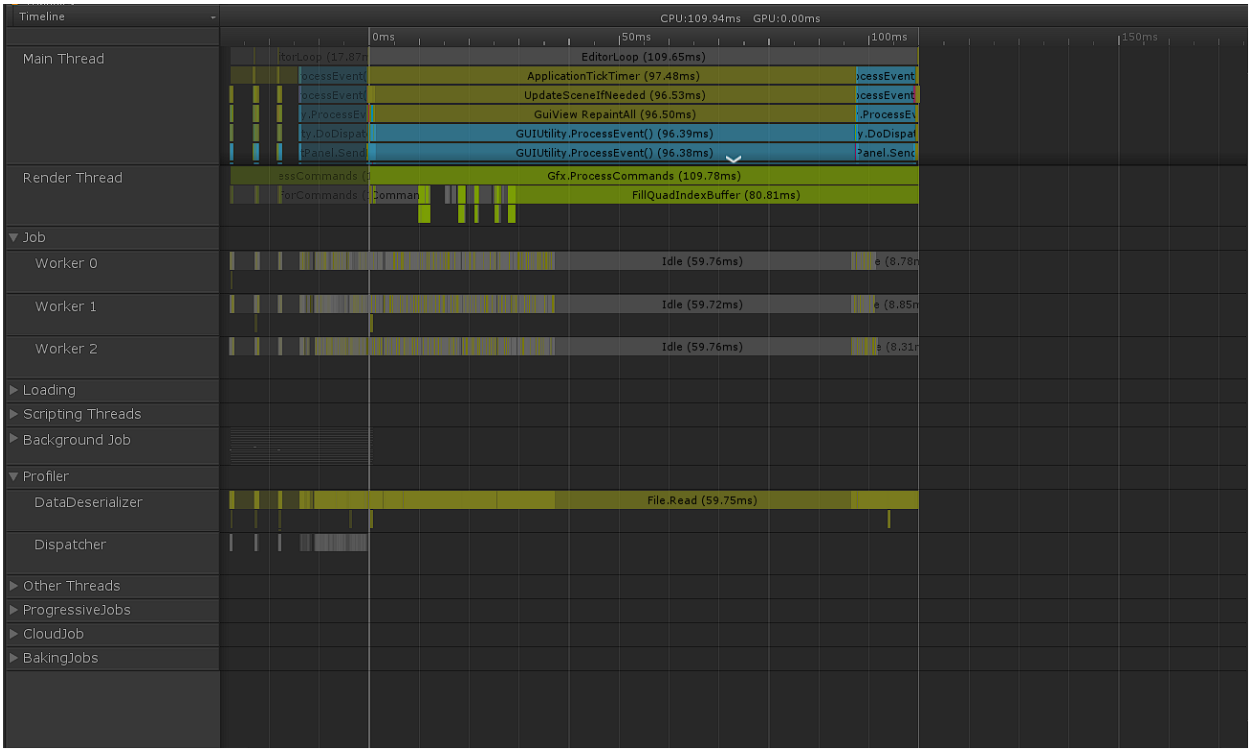

Figure 52: Timeline of the system when we used twenty-two model.

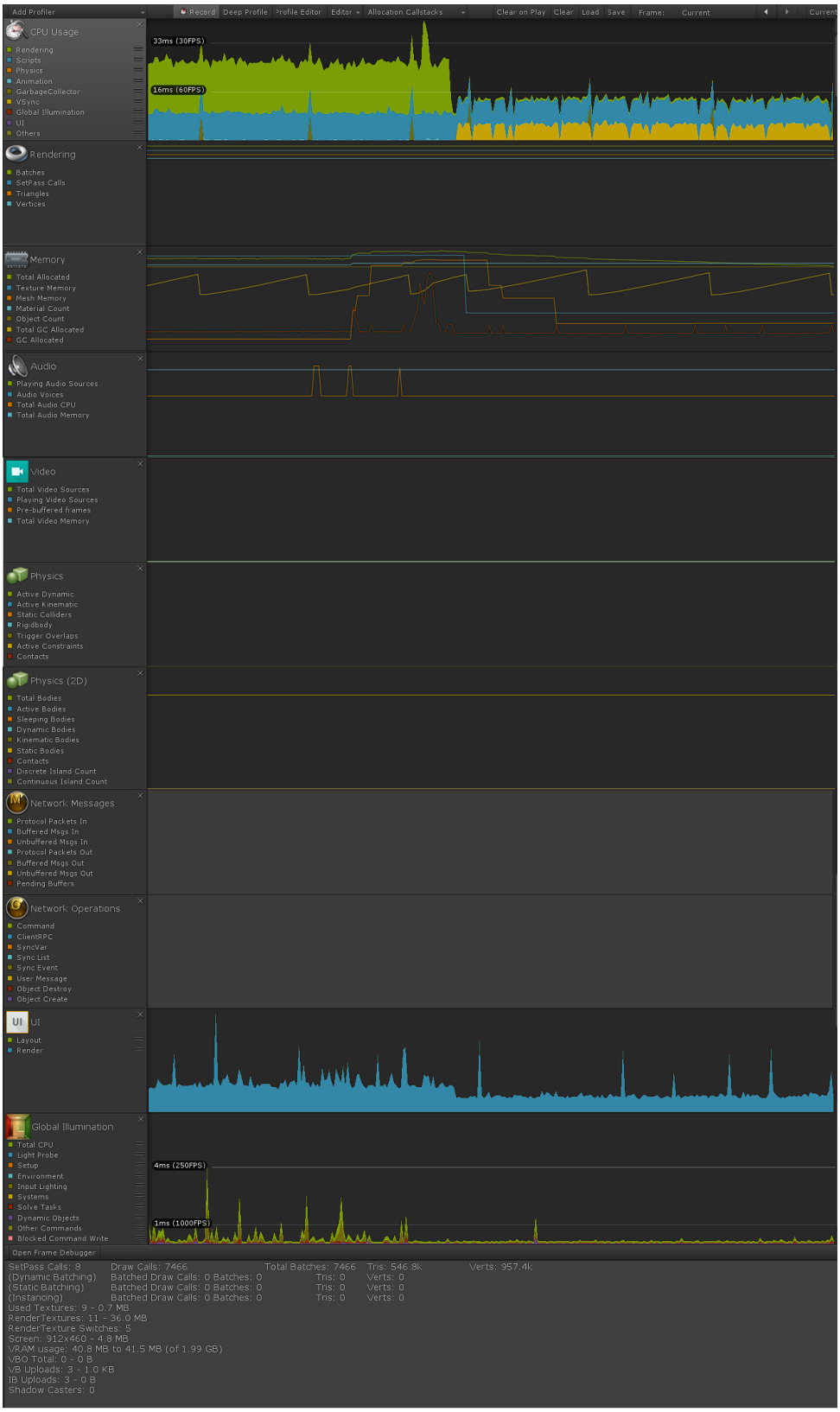

Figure 53: Profiler of the system when we used twenty-two model.

Chapter 6

Conclusions

And

Future Work

6 Conclusion and Future Work

Introduction:

At this section, we will explain the conclusion of what we have developed, implemented, and done at our system, and about what we have found during the implementation of the systems and how the information found helped us. Then we will state and explain each problem or issue we faced at our development either if it was a technical issues or a scientific issues. Then we will state our future work either if it was a problem at the current implemented system which we will solve at the future or if it was a new incremental implementation that will improve our system.

6.1 Conclusion

Our main objective is to develop a complete system, where we will implement the inverse kinematics algorithm which will help us to use it to animate a 3D rigged model in a 3D world environment. This system will be useful and helpful for users. Users like designers that wants 3D animated models to test their work with it or to use it in an environment of their own, also the animators need animated 3D models for the production of animated films. It also will be beneficial for freelancers and for small and large businesses. To implement the system, we need to have background about computer graphics, computer animation, and machine learning. Where computer graphics and computer animation will lead you to know about 3D animated models, also you need to know how 3D animated models are built with a skeleton of rigid segments connected with joints that is called by kinematic chain. And you need to know about how the 3D model structure is built. Then we will be able to apply one of the many inverse kinematics algorithms. There are many inverse kinematics techniques for modeling and solving the problems of inverse kinematics. Some of the techniques of inverse kinematics are The Jacobian Inverse Technique, Cyclic Coordinate Descent (CCD), and Forward and Backward Reaching Inverse Kinematics Technique (FABRIK). We will implement FABRIK algorithm, because it is one of the most popular heuristic algorithms which approximately solve the IK problem and offers a very realistic poses and has low computational cost. Hence, the ability to solve problems with multiple end effectors and desired targets is necessary for an IK solver, where FABRIK offer this solution. So after we implement the system, we test our system under different models, firstly we implemented the inverse kinematics in the form of 2D to understand the concept of the algorithm and how it works. Then we implemented a 3D inverse kinematic solver which is

FABRIK algorithm. Snice FABRIK algorithm works on a chain of 3 joints at least, so we applied and tested it on 5 spheres that represent a chain of 4 bones, then we applied and tested it on a complex model like elephant as we have shown and explained in the previous chapters. Then we found that the accuracy of the system on chain of length less than or equal to five is great as the movement of the chain with the existence of the pole is great and smoothly realistic, but with chain of length more than five, it is very good without the pole but with the pole the deformation of the model is damaged and produce unacceptable results. And we found that our system is slow when increase the number of models which reduce the FPS. At last we reached our objective successfully, where we implemented the FABRIK algorithm and applied it on a 3D model and we mapped two same models and the results was acceptable. But we failed to solve the problem of the multiple poles for the intermediate joints, we also failed to map two different models, and we also failed to do the machine learning implementation that we explained at our objective section at chapter 1. We think that our implemented system could be useful for some users. Users like designers that wants 3D animated models to test them by generating animation using FABRIK algorithm, the animators need animated 3D models that designed by the designers for the production of animated films, at last it will be beneficial for freelancers and for small businesses.

6.2 Problem Issues

6.2.1 Technical Issues

- The 3D model (elephant) was made by Maya platform so the formation type of the model was ".ma". So to solve this problem, we had to download Maya platform and understand how it works and how to use it, then we imported the 3D model on Maya then exported the 3D model as ".FBX" formation to be able to use it on Unity platform to use the 3D model in our system implementation as shown in the next figure.

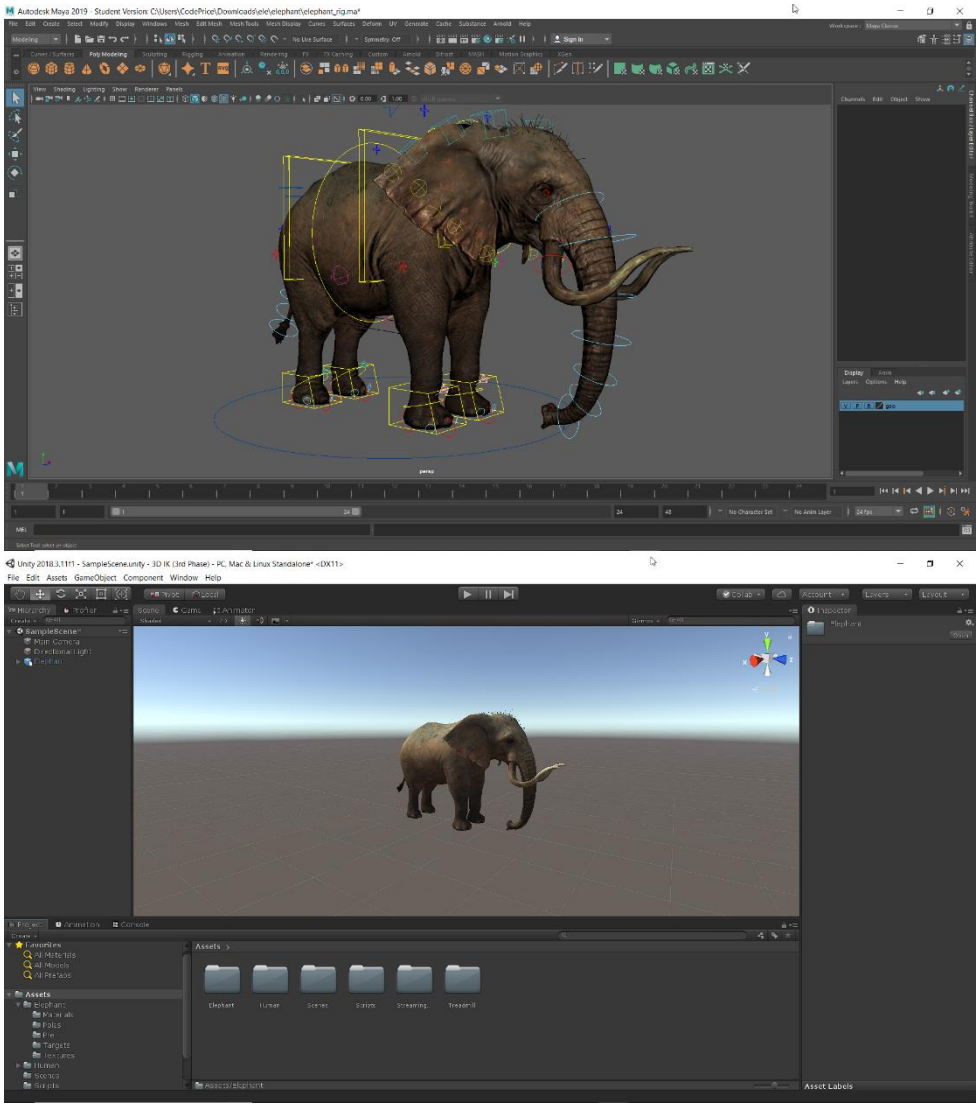

Figure 54: Converting model from Maya platform to Unity platform

- Our System is slow when we increase the number of 3D models (elephants) in Unity 3D world environment because our GPU is low-end GPU and its power is low and this

reduce number of frames per second. So to solve this problem we need to decrease the number of 3D models or to use a high-end GPU that is strong and powerful to keep the number of frames per second of the system high, smooth, and realistic.

- We had a problem to visualize the animation while we are doing each pose at specific frame at the edit mode. So to solve this, Unity provides a line of code that makes any script could be executed at edit mode (the line is "[ExecuteInEditMode]"), by using this line of code we are able to animate at the edit mode and the scripts are executed, so this will make animation creation and generation easier for us as shown in the next figure.

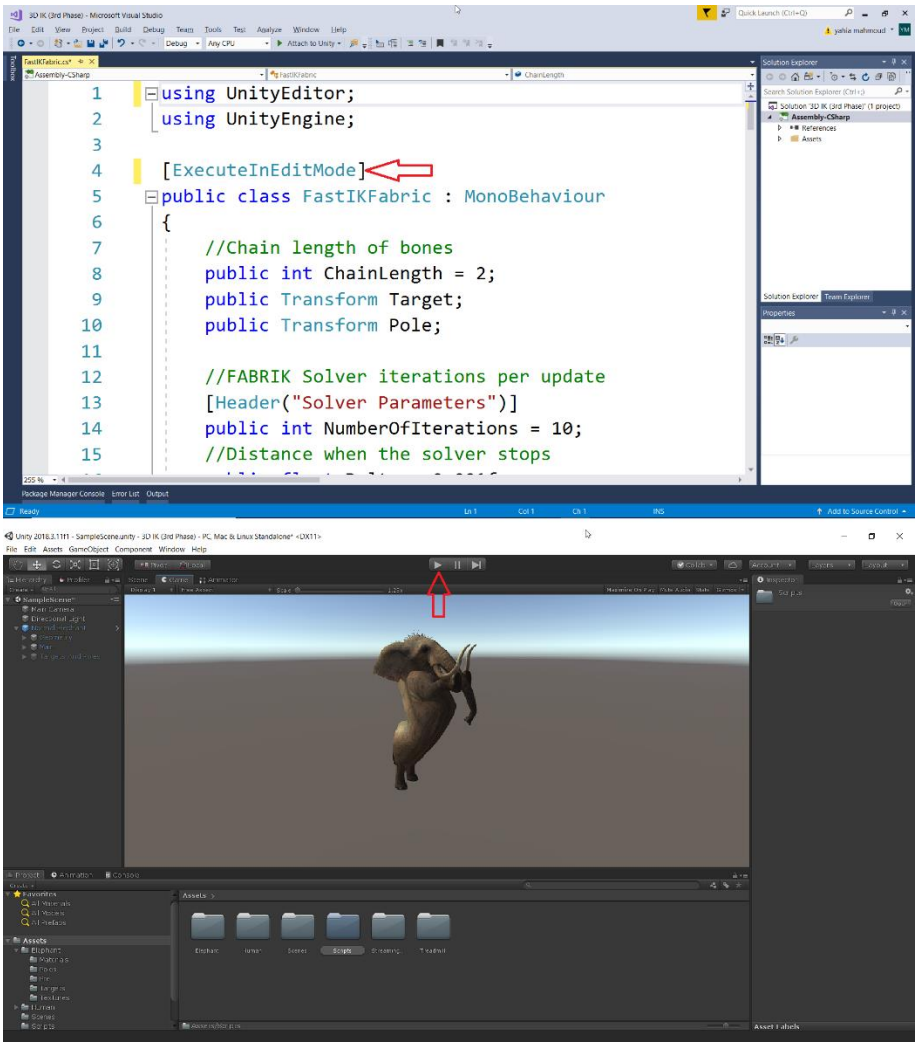

Figure 55: Show how the system run at edit mode using a line of code called "[ExecuteInEditMode]".

- When we run our system on the same scene on unity platform for a long period of time, unity takes a huge storage for caching and consume the whole storage and there won't be

enough memory to handle the system resources and data, so this will led to a crash in the system.

- We did not create more animations or animated more different models because we are not artists, this will be waste of time, and it is not the core work of the system.
- The generated animation by us mightn't be good for some users, because we are not artists and we do not know about the deformation system of all the animals (for example elephant movements). So the artists know the deformation system of models because they study, create, and draw them.

6.2.2 Scientific Issues

- We have an issue with Number of poles, so we use one pole for each chain which have length of more than or equal to two bones (three joints or more). One pole is not enough for long chains (for example 15 joints) because it makes the movement of the chain not smooth and not realistic as desired. So we will solve this problem in our future work as possible as we can.
- We have an issue with Mapping 3d models, because each model has its local world. In details each joint has different six DOF (Degree of freedom, x, y, z for position and rotation), so it is not right to map the six DOF of a joint in a model to a joint in another model. So to solve this problem, we have to collect information about each joint at the two models and map them according to this collected information. For example a model A moves forward with X axis and model B move forward with Z axis so we will give model B the delta X of model A and set it on Z of model B and do the same for the rest of axes of DOF.

6.3 Future Work

At the future:

- Firstly, we can solve the number of poles problem to create more realistic and smoother animation. For example, we will give each intermediate joint a pole as required, where this will increase the accuracy of the animation smoothness.
- Secondly, we can implement forward kinematics (FK) algorithm to make the animation of the 3D models with more functionality movements and create poses for the models that inverse kinematics may not create or it is hard to create or not smooth as the FK algorithm.
- Thirdly, we can make more animations for the same 3D model that we used in our system development or we can make animations for different 3D model, and then we use them at a gaming, video, or any project scene.
- Fourthly, we can use our developed system to implement a machine learning or a neural network. The input of the neural network will be a start pose and an end pose of a specific movement or animation which contain the targets and poles positions of each chain, then each pose will be computed by taking the position of the target of the end-effector in consideration, where each pose (pose i) depend on a previous pose (pose i-1). Then we will train the neural network model till we reach the highest accuracy that we desired according to the output poses that depend on a desired poses which added as an inputs for the neural network model.

References

1. Parent R. Computer Animation: Algorithms and Techniques. 3rd ed. San Francisco: Morgan Kaufmann Publishers Inc.; 2012.

2. Beane A. 3D Animation Essentials. 1st ed. USA: SYBEX Inc.; 2012.

3. Baran I, Popovic J. Automatic rigging and animation of 3D characters. ACM Trans. Graph. 2007 Jul; 26: p. 72.

4. Song W, Hu G. A Fast Inverse Kinematics Algorithm for Joint Animation. Procedia Engineering. 2011 Dec; 24.

5. Aristidou A, Lasenby J, Chrysanthou Y, Shamir A. Inverse Kinematics Techniques in Computer Graphics: A Survey. Computer Graphics Forum. 2018 Sep; 37: p. 35-58.

6. Bhatti Z, Shah A, Shahidi F, Karbasi M. Forward and Inverse Kinematics Seamless Matching Using Jacobian. Sindh University Research Journal (Science Series). 2013 Aug; 45.

7. D. G. L. Linear and Nonlinear Programming: Addison Wesley; 1989.

8. Aristidou A, Lasenby J. FABRIK: A fast, iterative solver for the Inverse Kinematics problem. Graphical Models. 2011 Sep; 73: p. 243-260.

9. Kenwright B. Inverse Kinematics – Cyclic Coordinate Descent (CCD). Journal of Graphics Tools. 2012 Oct; 16.

10. Aristidou A, Chrysanthou Y, Lasenby J. Extending FABRIK with model constraints. Computer Animation and Virtual Worlds. 2016 Feb; 27: p. 35-57.

11. Sito T. Moving Innovation: A History of Computer Animation: The MIT Press; 2013.

12. Baerlocher P, Boulic R. Task-priority formulations for the kinematic control of highly. IEEE IROS'98. 1998 Nov;: p. 323 - 329 vol.1.

13. Huang J, Fratarcangeli M, Ding Y, Pelachaud C. Inverse kinematics using dynamic joint parameters: inverse kinematics animation synthesis learnt from sub-divided motion micro-segments. The Visual Computer. 2016 Aug.

14. Buss S, Kim JS. Selectively Damped Least Squares for Inverse Kinematics. J. Graphics Tools. 2005 Jan; 10: p. 37-49.

15. Fourati N, Pelachaud C. Emilya: Emotional body expression in daily actions database. 2014 May.

16. Bharaj G, Thormählen T, Seidel H, Theobalt C. Automatically Rigging Multi-component Characters. Computer Graphics Forum. 2012 May; 31.

17. Masson T. CG 101: A Computer Graphics Industry Reference: Digital Fauxtography Inc.; 1999.

18. Means S. Pixar founder's Utah-made Hand added to National Film Registry. The Salt Lake Tribune. 2011 Dec.