Antti Ruuskanen

Inverse Kinematics in Game Character Animation

Bachelor of Business Administration

Information Technology

Autumn 2018

KAJAANIN
AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Tiivistelmä

Tekijä(t): Ruuskanen Antti

Työn nimi: Käänteisen kinematiikan käyttö pelihahmojen animoimisessa

Tutkintonimike: Tradenomi (AMK), tietojenkäsittely

Asiasanat: animaatio, käänteinen kinematiikka, peligrafiikka, pelihahmo, rigi, unity

Työn tavoitteena oli tutkia käänteisen kinematiikan käyttötarkoituksia, käyttötapoja, ja tilanteita, joissa sitä käytetään. Käyttötapoja voi soveltaa useilla tavoilla, vuorovaikutuksesta ympäristön kanssa jopa uusiin pelimekaniikkoihin. Käänteisen kinematiikan käyttö animoinnissa on myös tutkittu.

Tämä työ painottuu teoriaan ja tutkintaan. Osaksi tämä työ keskittyy siihen, kuinka käänteistä kinematiikkaa usein käytetään. Työ tutkii tilanteita, joissa käänteisen kinematiikan käyttö parantaa työn tekemistä animaattoreille sekä leivotun animaation tekemisessä että reaaliaikaisessa käytössä. Matemaattinen osio keskittyy käänteisen kinematiikan toteuttamiseen käytännössä. Työssä käsitellään kahta eri toteutustapaa eri vaikeusasteilla. Molempiin liittyvät hyvät ja huonot puolet käydään läpi.

Asiasisältö olettaa lukijan tietävän perusteet rigien tekemisestä sekä hahmojen animoimisesta. Monet luvut käyvät läpi asioita, jotka pohjautuvat muihin 3D grafiikan osa-alueiden, kuten mallintamisen, pohjalle. Perustietämys mallinnusohjelmista ja pelimoottoreista on auttaa lukijaa ymmärtämään syvemmin, kuinka käsiteltyjä ratkaisumalleja käytetään hyväksi.

Abstract

Author(s): Ruuskanen Antti

Title of the Publication: Inverse Kinematics in game character animation

Degree Title: Bachelor's Degree in Business Information Technology

Keywords: animation, inverse kinematics, forward kinematics, game graphics, game character, rig, unity

The aim of this thesis was to explore the methods of implementing inverse kinematics and the situations that they are used in. The methods have multiple applications which can be used from making the environment come alive to even creating new game mechanics. The usage of inverse kinematics as a tool in animating is also explored.

The thesis focuses more on theory and investigation. Part of the thesis focuses on how inverse kinematics is often used. It explores the situations it should be used to improve the workflow of animators in both baked animation and real-time application. There is also a mathematical part that focuses more on how to implement inverse kinematics in practice. It chooses two different approaches with differences in difficulties. There are pros and cons evaluated that come both methods.

The content assumes the reader to know the basics of rigging and character animation. Many of the chapters involve works, that are based on other branches of creating 3D graphics, like modelling. Basic knowledge of modelling software and game engines help the reader to understand more deeply how to use the solution models discussed in practice.

Table of Contents

List of symbols

IK	Short for inverse kinematics.
FK	Short for forward kinematics.
Clip	Often refers to two objects phasing through each other.
Rig	A skeleton made to deform a 3D mesh or 2D sprites.
Baked animation	An animation that is static and does not interact with the environment.
Interpolation	Inbetweening in digital space, mathematically create frames between keyframes.
Constraint	A restriction placed on a bone to control its properties.
End effector	A device that is positioned at the end of an IK chain.
Jacobian matrix	A matrix used in vector calculus. It is one method for calculating inverse kinematics.
DOF	Short for degrees of freedom. Possible movement along an axis, either rotational or positional.
Control bone	A bone that does not deform the mesh but affects other bones.

1 Introduction

The usage of inverse kinematics is popular in the gaming industry. It is used in multiple ways and there are also many ways to implement it. The focus here is more on how it helps character animation, from legs and hands to some more inventive ways. The first thing needed to implement inverse kinematics is to know what it is built on. There are some requirements that need to be met.

The mathematical portion goes through two different ways of inverse kinematics implementation. One of the ways is meant for simple use, simple implementation but it is not quite as powerful. The second way is more for advanced use with more complex implementation. Constraints and limitations are also considered.

Finally, there is a possibility of using inverse kinematics in unusual rigs and game mechanics. There are some games that use inverse kinematics in a game mechanic. Some rigs are also very specific and can benefit from special animation techniques. Sometimes it is almost a requirement as animation done by hand proves to be too difficult or time consuming.

2 Inverse kinematics

The process of inverse kinematics is finding the positions for the joints in the chain to make the chain behave as it is required to (Jenkins 2018). This is usually achieved by rotating the joints to reach the end position for the last link called end effector. Using this technique in game character animation requires a model or object that can be animated. It can be used without joints and chains that are animated, but the benefits become apparent when used with one or more joints. (Richardson 2012)

2.1 Preparing a rig

Animation is often implemented by using a character rig. 2D games often use frame-by-frame animation, which cannot really take advantage of inverse kinematics. However, it is also possible to create a character that consists of multiple sprites and put them in hierarchy. With chains of sprites, implementing inverse kinematics is possible even in 2D. 3D graphics use skeletal rigs most of the time so using inverse kinematics is a popular technique with 3D models.

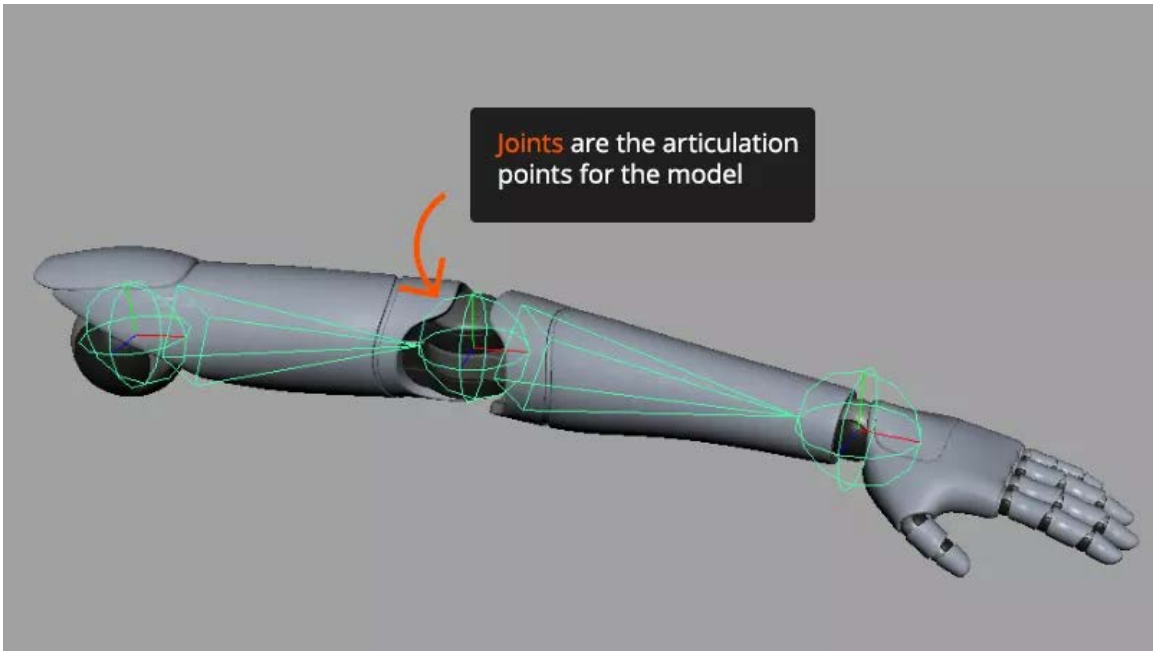

Figure 1. An example of a rigged hand

A rig, also called skeleton, is a chain of bones with its own hierarchy. Bones and joints can be used interchangeably because bones always have pivot point in space. That pivot point is where the bone rotates, thus creating a joint at that point. For example, a leg rig can be created from two bones, separated by a knee. The leg would consist of two bones, two joints, because there are two objects that both have their own transforms. They both can translate, scale or rotate around their own pivot point. (Aristidou, Chrysanthou, Lasenby & Shamir 2017).

The upper part of the leg would be the parent in this hierarchy. Object animation is in “local space”, which means their transform is relative to their parent. Moving the parent would automatically move any child objects down the chain assuming they are connected to each other. Moving the parent causes the child objects all to move in “world space”, but their own transform remain the same. All objects have their own local transforms that are separate from their parent.

There are many kinds of joints that exists but the most used in character animation are hinge joints and ball joints. Most joints only need to rotate in a single direction, like the fingers in a human hand. Some joints, like shoulders, require rotation in multiple directions. Prismatic joints may be used in mechanical contraptions such as luggage bag handles, but they are rarely used in living creatures. The type of constraints the joint has depends on what it needs to achieve, so when creating a rig for a character it is important to know what kind of movements it can make. (Aristidou et al 2017, 12).

2.2 Forward and inverse kinematics

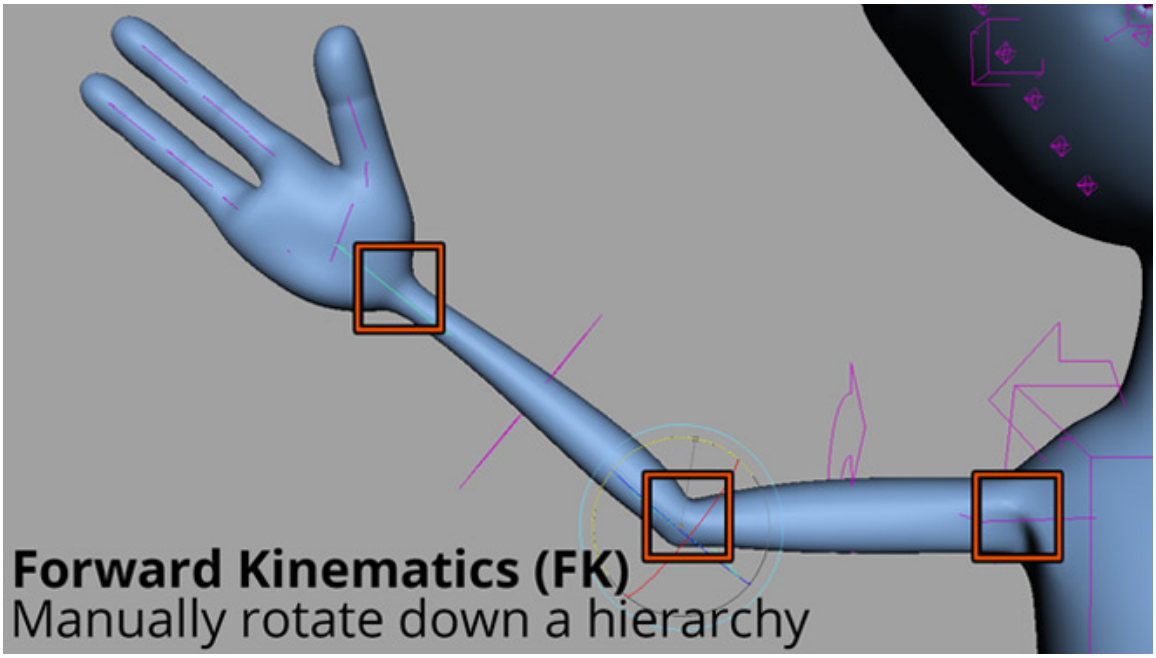

Figure 2. Forward kinematics setup

Forward kinematics is inferring the pose of the end-effector, given joint positions. It is a traditional approach to animating. The animating starts from high up in the hierarchy, moving bones to the desired positions one at a time. This type of animation can be preferred in situations where bones require accuracy in their transform in the local space. Preparing a rig for forward kinematics does not require any specific extra work if the hierarchy is created properly. (Bermudez 2017).

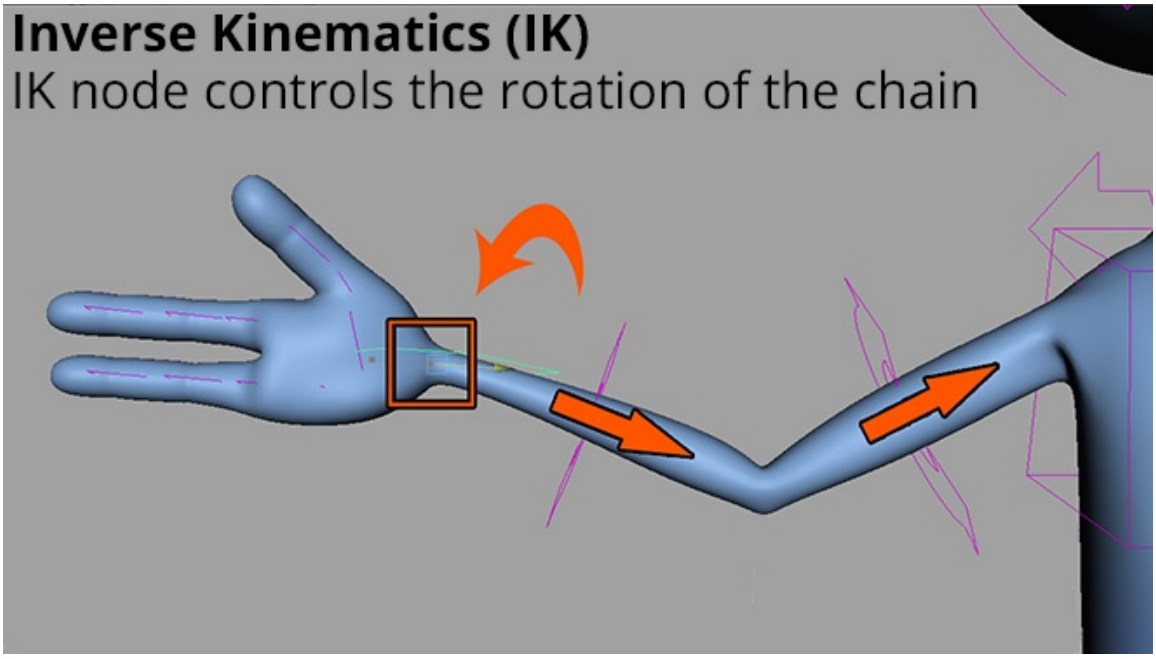

Figure 3. Inverse Kinematics Setup

Inverse kinematics works in the opposite way from forward kinematics. The end effector is the first thing that is set, and the rest of the bone chain is calculated afterwards. The rotations of the parent bones will have to be calculated to accommodate the final bone to reach the desired solution. This is generally more difficult to implement than forward kinematics, and some problems can arise from the multiple solutions that it may give, depending on the method. (Liang 2012). The animator defines the transformation for the last bone in the chain. The program uses mathematical functions to provide the necessary changes to the bones higher up in the chain, improving the animator's workflow. In many cases, it also improves the quality of the animation tremendously. (Wikipedia.org 2018).

Whether it is better to use IK or FK however, depends entirely on the rig, the animator and the situation at hand. In many cases there is a clear purpose for both, and in certain situations it is obvious which method should be used. Oftentimes professional rigs provide the animator with a choice to switch between both methods. Certain animations involving the same bones may require different approaches in two different situations. Sometimes it may come down to simple preference as to how one wants to do their work.

2.3 Advantages in animation

Inverse kinematics can also be used in real time to fix problems when interacting with the environments and the surrounding objects. When animating a character rig, the animations are then baked into keyframes. Those keyframes are static and cannot interact with other objects without some extra work put into the game. IK is a system that can overwrite, or in some situations add into, the keyframes. Giving new positions and rotations to the bones with IK and adding the keyframe animations to the rest of the body can improve the quality of the animation within the game.

The benefits of inverse kinematics in baked animation are similar but can solve even bigger problems. When animating a character, not all body parts necessarily require the use of inverse kinematics but with specific goals in mind it is possible to determine which method is preferred for this specific purpose.

For example, when animating a character that has its feet on the ground, it is important to have the soles stay still. The soles of the feet are low in the hierarchy and are usually parented to the bones higher up in the leg. Moving the upper body and the hips also moves the leg's positions, which in turn move the soles of the feet. Having the feet move is not a problem if the soles stay still. In fact, having some movement in every part of the body is realistic when it comes to living creatures. However, if the soles move it makes the feet feel floaty as if they are not connected to the ground. Ground has friction, so the soles should not move needlessly unless they animator wishes them to. With forward kinematics only, the animator would have to keyframes the soles in the exact place again and again for every frame. This is not a very efficient solution, would cause problems when adjusting the animation due to the number of keyframes that require change, and still have problems when it comes to interpolation. (Autodesk Knowledge Network 2017).

2.4 Interpolation problems

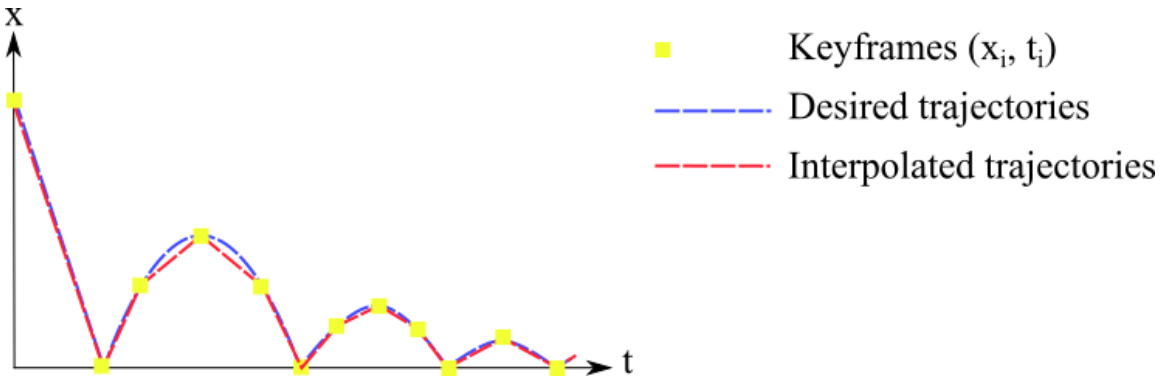

Figure 4. Interpolation between keyframes

Keyframes are created in 3D animation to mark a change in either position, rotation or size. Animating often begins with generating poses by transforming the bones of a rig and then saving them in so called “keyframes”. 3D animation using bones can also take advantage of a feature called interpolation. Interpolation in this case refers to the action of approximating frames in between the keyframes the animator has created. Interpolation can be linear or curved, and the animator generally has full control over it through a graph editor. In fact, a big part of an animator’s job is to adjust these curved to make the animation flow properly. (Ruskeepaa & CTI Reviews 2016).

Transformation in bone chains exists in local space. It means that any change to the transformation of the bone happens in relation to its parent. If a parent bone rotates, its child bones rotate with it. However, the child bones have their own rotation in local space which does not change. When the animator wants a character’s foot to stay in place, it means that the foot should stay still in the overall sense, in world space. Having it stay still in local space means nothing when the parent moves around. (Blender 3D: Noob to Pro, 2018).

Interpolation is helpful when creating smooth animation, but it can also be a source of big problems. Because the animator only specifies the change in transformation of a bone in certain frames (keyframes), the computer determines the change between those frames. Because the change in transformation is measured in local space, having all the bones rotate with their own curves with different speeds and different timings can cause the end of the bone chain, the foot, to move when it is not supposed to. The animator can try to adjust the curves or create more keyframes to combat this small movement in between the original keyframes but this is an inefficient method that creates a lot of extra work and visual cluster in the program. It also makes any changes more difficult because of the number of keyframes one must adjust. A picture depicting a situation where the animator

has created too many keyframes for a bone. Making any significant changes would be tiresome simply because of the number of keyframes that would have to be changed.

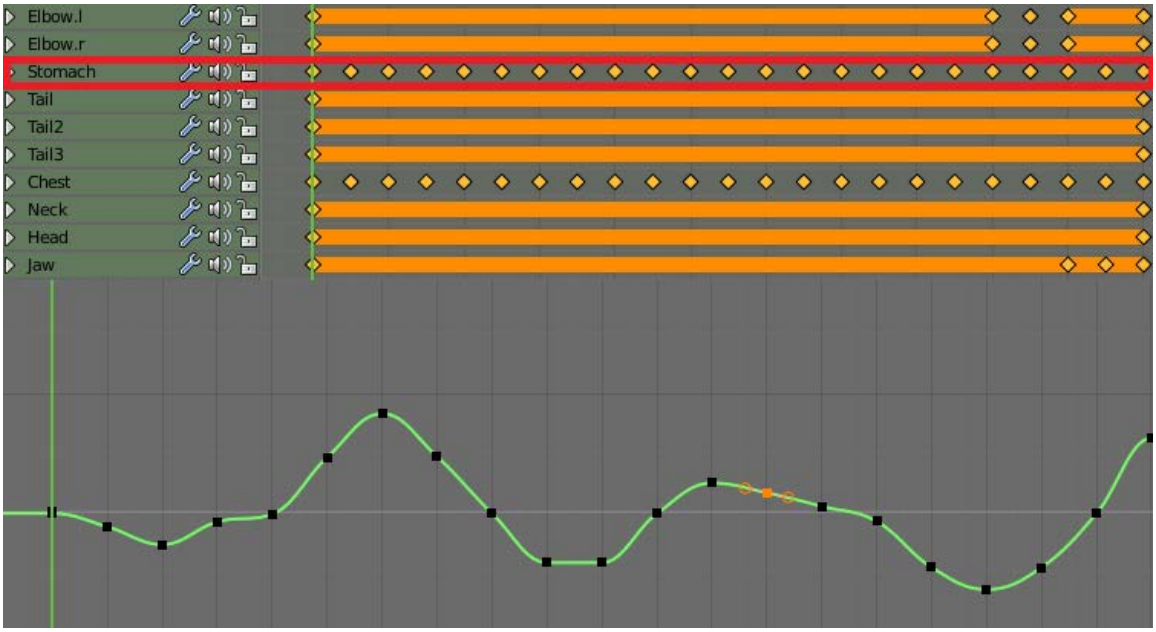

Figure 5. Blender graph editor with keyframes

IK helps with the problems regarding interpolation by doing what the animator would have to do and does it better. If an end effector is specified to be on the ground and the foot chain is targeted to that, the computer can do all the necessary calculations afterward. It doesn't matter how the parent bones change because the foot is always calculated to end up at the position of the end effector. This also makes any adjustments to the animation easy, because all that is required to change is the position of the end effector and that can sometimes be changed with a single keyframe change.

2.5 Constraints and restrictions

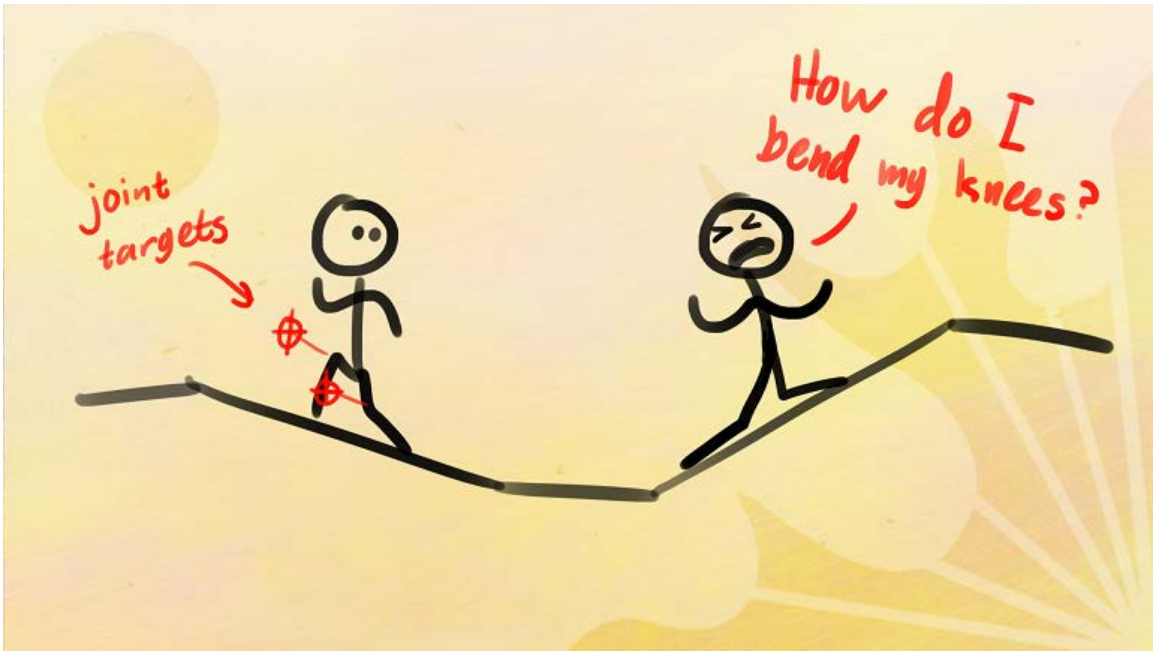

Figure 6. Joints can bend the wrong way without proper setup

Constraints are modifiers on bones that help transform its properties in different ways. It is basically a way to automate certain behaviors on those bones. (Blender 2.80 Manual 2018) Some examples include tracking and limiting rotation. Inverse kinematics is also a constraint in that sense.

Inverse kinematic constraint can also be adjusted for more control. Having an IK set up is the first step, to refine the setup it may need some more limitations. The IK calculations can have many solutions and sometimes the result does not look as expected. The mathematical calculations might have multiple solutions, so the way the bones rotate. Constraints helps to eliminate the solutions that provide correct IK but incorrect looking results. Joints have their own constraints and the IK calculations should also respect them. (Siewert 2012, 5).

A popular method for more accurate IK solutions is to use extra bones to guide the system. The extra bones are not bound to model's vertices, so they do not deform it in any way. The only purpose for them is to be the target of certain bones in the IK chain. It is a simple but effective method for more control over the animations. In addition, these extra bones can even be used in real-time IK when imported to a game engine. The implementation

is not really that different in theory. The settings will have to be modified and constraints created again in the software imported, but the calculations are basically the same.

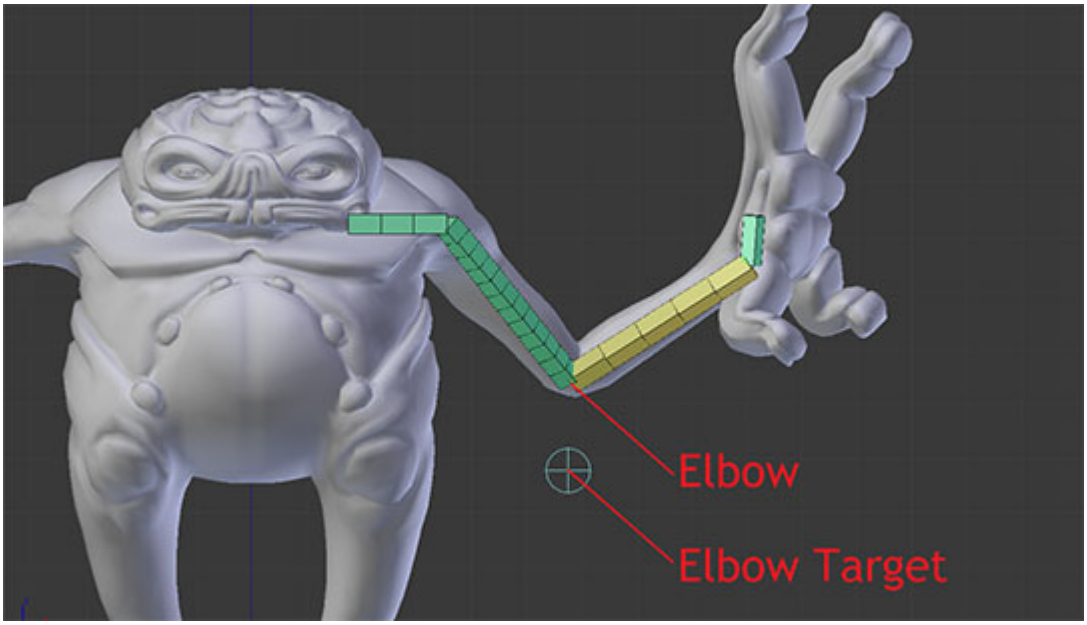

Figure 7. Elbow joint targeted for proper bending

In Figure 6, the elbow joint is targeted to the Elbow Target bone. In this case, it is the upper arm's bone which is constrained. The bone's origin is connected to the shoulder bone and the end of the bone points towards the Elbow Target They purpose of this is to have the IK system rotate appropriately to create the illusion of the arm bending properly. Not all joints can bend in every direction, so bones are rotated to create an angle in which they may point towards the target.

3 The Mathematics of implementation

3.1 The end effector

End Effector is the end position of the chain of bodies, the last link. The main purpose of IK is to adjust the rotations and positions so that the end of the chain ends up at end effector. By having the end effector be at a constant point it is easier to move bones higher up in the hierarchy and still have the chain end up at the same point. This works even with interpolation because the computer is also adjusting the curves between keyframes. (Bouchard 2014).

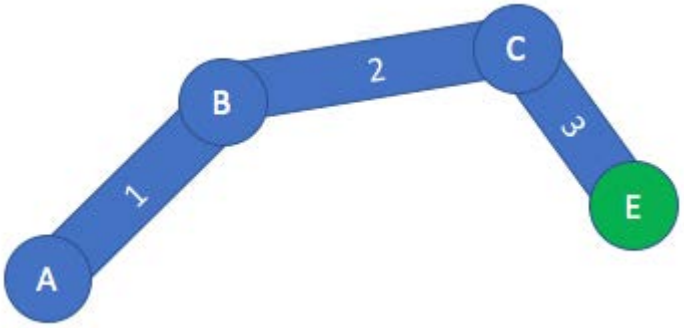

Figure 8. End effector portrayed by the letter R in the chain

Calculating this the path to end effector (EE) is the main goal of inverse kinematics calculations. To be more precise, the goal is calculating the rotations required to make the final bone in the chain end up at EE. There are multiple ways to calculate it, all of which have their own strength and weaknesses. In the end it's up to the user to choose which method suits their purposes the best. (Siewert 2012, 5).

Some other difficulties in inverse kinematics problem are: kinematic equations are coupled, and multiple solutions and singularities may exist. This thesis will mostly deal with two popular mathematical methods of implementing inverse kinematics.

3.2 Iterative method and Jacobian matrix

An iterative method for solving IK means that it is not exact but tries to get as close to the proposed result as possible. As its name implies, an iterative method will loop the calculation multiple times each time become more and more accurate. The number of times can be altered to fit the purpose of the IK. Naturally, fewer loops will result in the IK being less accurate but that might not always be noticeable.

Method to use are various but the Jacobian Inverse IK Method is among the most popular ones. The Jacobian method requires three main steps. First you must find out the rotations of the joints in the mechanism. The second step is to compute/calculate the change in rotations. Finally, the Jacobian is computed. (Bermudez 2017)

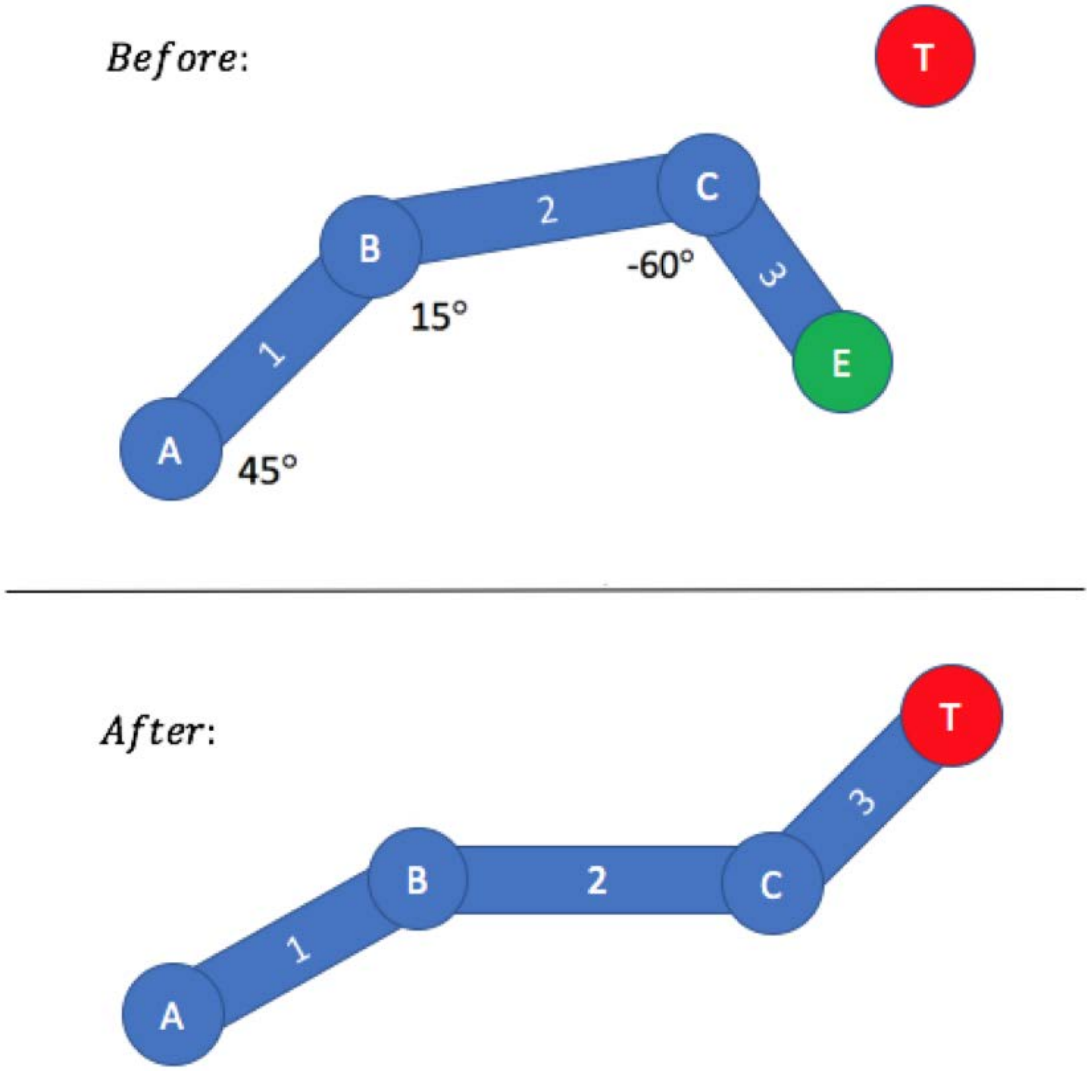

Figure 9. IK chain with three joints

The equation that gives the result T is

$$T = O + dO. \qquad (1)$$

In the equation above, T is the final orientation of the joints. O is the initial orientation of the joints. The symbol dO signifies the change in rotation. By adding the change in rotation to the initial rotation, the final joint rotations values will be determined. Using the equation, it is possible to get the correct joint configurations. (Bermudez 2017).

In the picture above, the initial rotations of the joints are 45°, 15° and -60°. All of these are relative to their parents. Assuming the joint was not rotated, and the bone continues in

the same direction as their parent, the rotation would be 0°. The initial rotation was symbolized with O in the equation, therefore we know that

O = (45°, 15°, -60°). (Bermudez 2017).

The change in rotation is not know from the beginning, therefore it must be calculated. Even if O is known, that still leaves two variables which have unknown values. Knowing only one variable in an equation with three is difficult.

Although dO is what needs to be calculated next, the equation can be modified a little to add a step to it. As stated earlier, this is an iterative method that approximates the result, therefore it is not completely accurate. By adding a step into the calculation, it is possible to get closer and closer to the final answer with more accuracy. That "step" in our equation will be called "h". This value can be tuned according to the user depending on the simulations it is used in. (Bermudez 2017).

$T = O + dO * h.$ (2)

Finding the Jacobian can be done with either the Jacobian inverse or Jacobian transpose. Jacobian inverse and transpose. The Jacobian Transpose always exists, as opposed to the Jacobian Inverse. It is also less computationally expensive, which means that the performance will be faster. Additionally, the Jacobian Transpose is easier to implement than the Jacobian Inverse. (Bermudez 2017). This example will use the Jacobian Transpose further in step three. The equation to calculate the change in rotation using the Jacobian transpose is

$dO = J^T V,$ (3)

where V is the distance between the target position and the end effector. Simply put, end effector position subtracted from the target position. J marks the Jacobian. (Bermudez 2017). The Jacobian is a cross product of each of the joints' rotation values with the value of V.

$$J = \begin{bmatrix} R_A \times (E - A) & R_B \times (E - B) & R_C \times (E - C) \end{bmatrix}$$

Figure 10. Jacobian transpose with three joints

With all the calculations done, it's possible to calculate the final rotation values. The equation (3) can be used to calculate the change in rotation. Using equation (2) it is possible to calculate the resulting rotations for all the angles.

3.3 Analytical, the geometric approach

If the chain only has two joints, then it may be possible to solve IK geometrically. Solving small IK chains with geometry does not require the use of matrices and can be quite simple. However, if the number of joints grows higher, the complexity of the calculations also increases. In that case, another approach might be preferable. (Applied Go 2016).

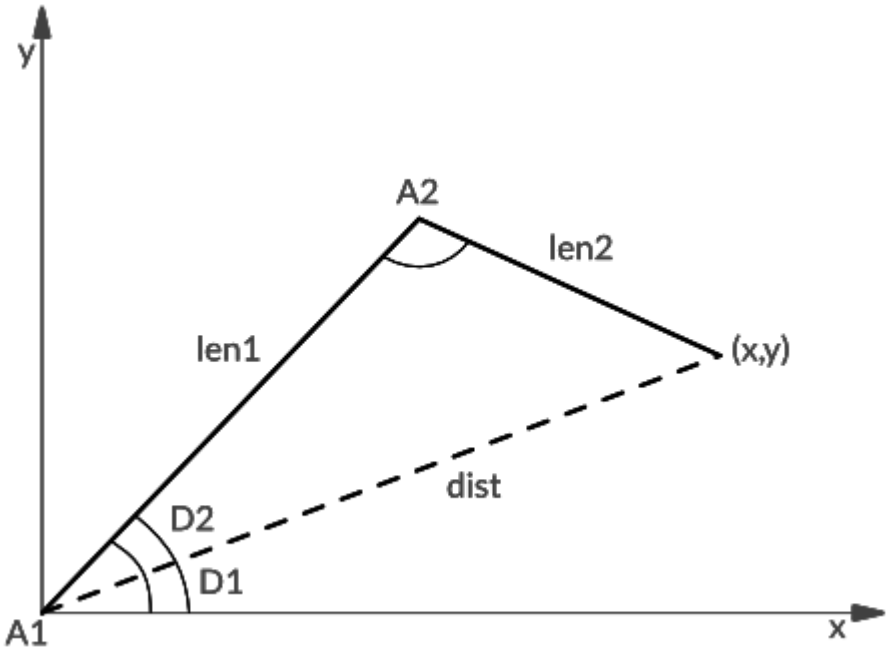

Figure 11. Calculating angles using trigonometry

The picture above depicts a common IK chain with two joints. The joints are represented as A1 and A2. The angle A1 is further divided into D1 and D2. The chain could be depicted as a humanoid hand, as it is very common for a hand IK to look like this. In that case, len1 and len2 would depict the lengths of the two segments of the arm. The elbow joint is placed at A2, right at the end of the first segment. The position (x,y) is the end effector, which is specified by the user. The line called dist represents the distance from the first joint to the end effector. (Applied Go 2016).

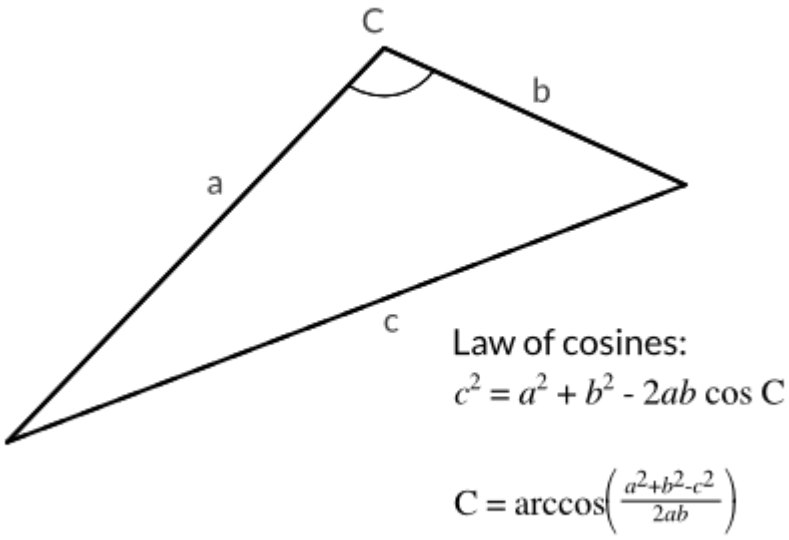

Figure 12. Using law of cosines to calculate C

Using a basic trigonometric formula, the law of cosines, it is possible to calculate the angles of the joints. The symbols in the picture have changed into simpler names for the math equation. The basic formula goes as follows

$c2 = a^2 + b^2 – 2ab \cos C,$ (4)

which can be rearranged into,

$C = \arccos(a^2 + b^2 – c^2 / 2ab).$ (5)

With the second joint angle, A2, calculated, it is time to calculate the A1 angle.

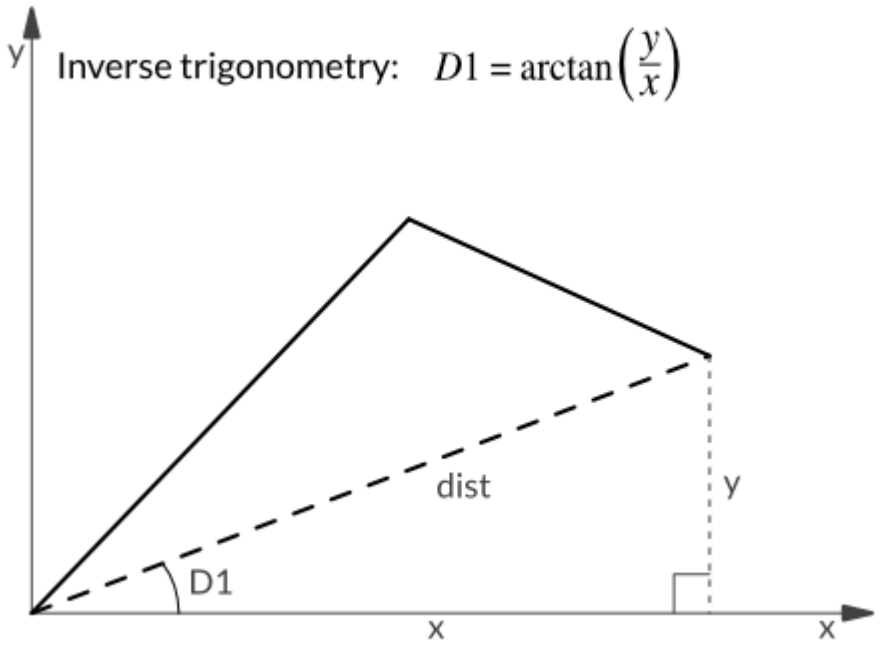

Figure 13. Basic trigonometry to calculate D1

As A1 is divided into two angles, D1 and D2, both must be calculated. The first angle can be calculated with the arctan of D1

D1 = arctan(y/x). (6)

With D1 one complete the only thing left is D2, and it can be calculated by using the law of cosines just like before (Applied Go 2016). Afterwards D1 can be added to D2, which is equal to the angle A1. Now, both A1 and A2 angles are known and can be applied as the joints' rotations to achieve correct result.

3.4 Degrees of freedom and limitations

Degrees of freedom refers to the ability of the rigid body to rotate and move along the three axes; x, y and z. The most basic forms of IK can be created with one degree of freedom. For example, prismatic joints moving along one axis would only have one degree of freedom. As the mechanism becomes more complicated, the axes of freedom often increase.

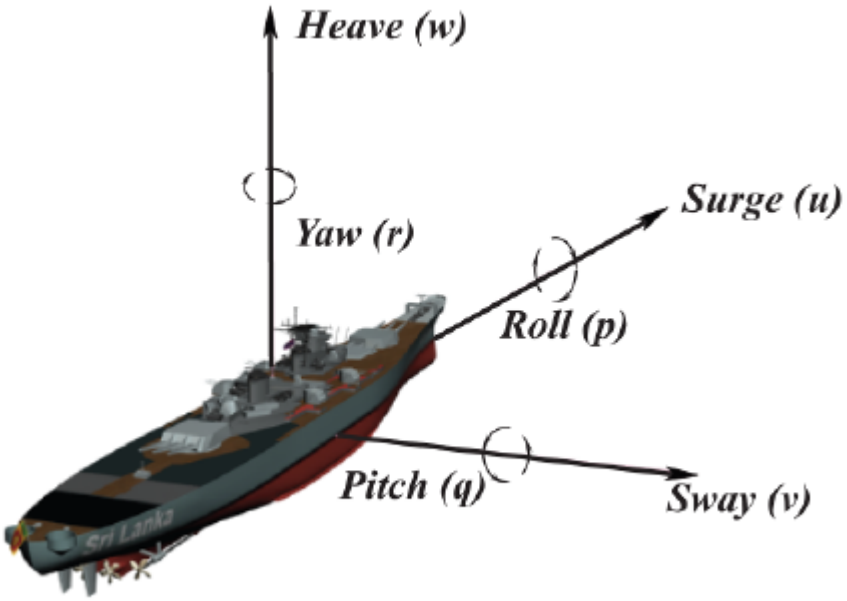

Figure 14. A ship's rotational and positional movement axes' names

Degrees of freedom is often abbreviated as DOF. The rotational and positional motions have multiple names have many names, and the picture above demonstrates some of them. The change in orientation, rotational motions, on axes are often called yaw, roll and pitch. The translational movement of the object has many names, but they are called heave, surge and sway in the picture above about ship motions. The does not have a chain of joints and only moves around a single point. Therefore, it only has six DOF. (Tristan & Fossen 2005)

The combined number of joints in a rigid body gives the total DOF in the system. When dealing with a single body, like the ship, the number is 6. When the number of joints increases, the combined DOF increases as well. In a complex mechanism, such as a human hand, the number can grow surprisingly high.

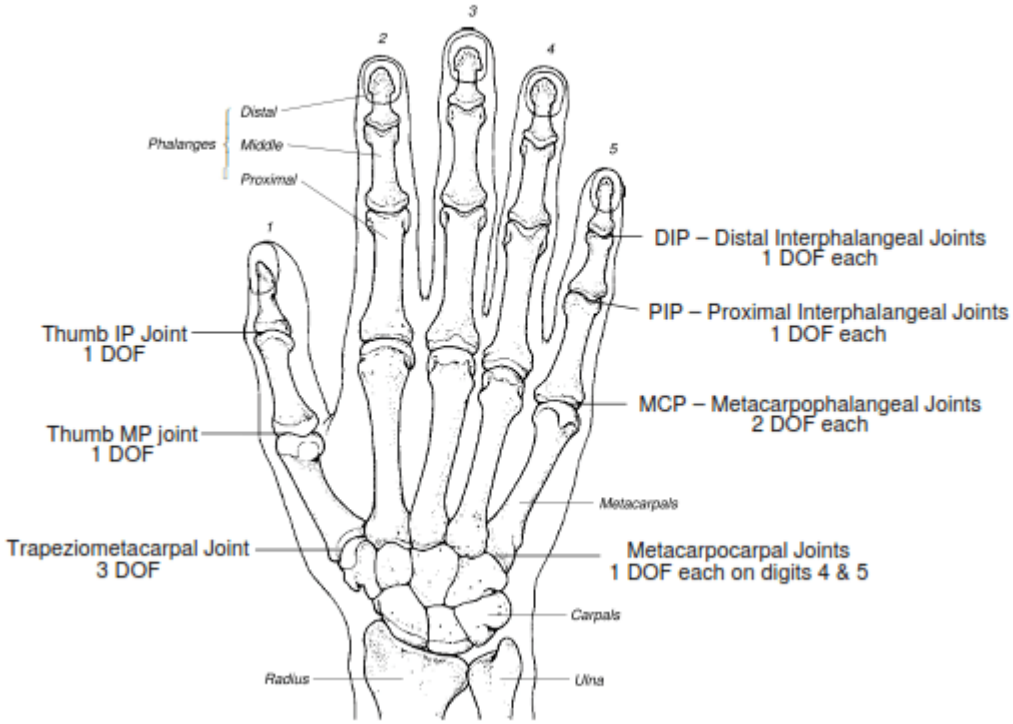

Figure 15. Hand's degrees of freedom

A typical human hand contains various kinds of joints. They all have their own DOF and together form a mechanism with a combined number of DOF. The four fingers, apart from the thumb, all have similar joint configurations, only differing in length. The thumb, however, has a different number of joints and DOF making is behave differently. While degrees of freedom are dependent on the number of joints, the type of joint is also an important matter. Different types of joints may allow a different number of DOF.

4 Role in character animation

While IK can certainly be useful when creating baked animation, it can also be used in real-time in the game engine. The objective might be the same as when used in baked animation, but it is often treated as an extra. Sometimes IK is only used to add immersion by allowing the baked animations to adapt to the surroundings. When looking at older games it is rarer to see such techniques used.

Real-time use of IK can be very inventive, but there are certain purposes more common than others. For humanoid creatures, the most popular application of IK would be feet. Hands also often have the possibility to turn on IK for different animations. Head IK is used to turn the character's head towards points of interest to create more interaction with the environment. Even the rest of the body is often influenced by the other IKs in some way. In the end, even whole characters may be animation by nothing but this method, but the most sensible solution might be to mix it with forward kinematics. (Massoudi 2015).

4.1 Feet and the ground

One of the most widely used applications of IK in game character animations is limbs which touch the ground. In most cases this means the legs of a humanoid being, but the same method also works on beings with any number of legs. This method can be used in both real-time and baked animations. When animating a humanoid being, often the sole of the feet should stay in the same place while the rest of the body moves. Obviously, the feet should also be movable when necessary. (Doody, 2016). If IK is not used, there is a real possibility of the feet sliding along the ground, making it more difficult to have animations where the character is properly standing still. In practice, using only forward kinematics can make it much more difficult to have an animation with non-moving feet.

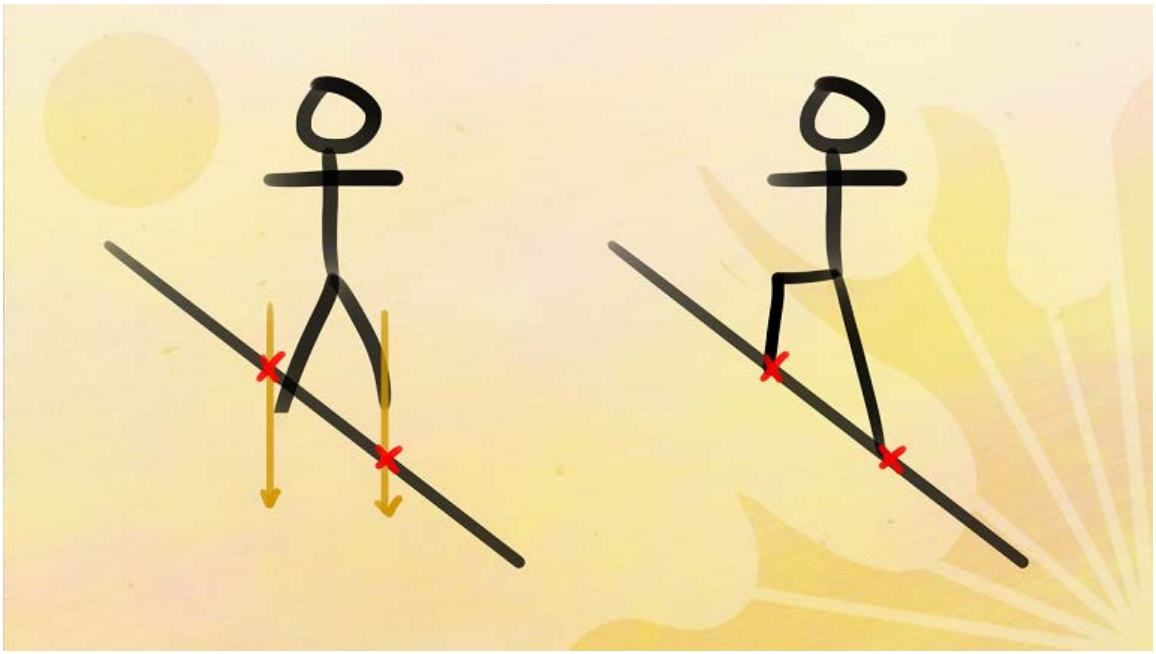

Figure 16. A character with and without foot IK

A very popular method of implementing IK is real-time for feet. Even if it can already be used during the creation of baked animation, adding it real-time can bring in a lot more. In that case, the character's animation plays correctly in the other parts of its body while maintaining the legs position on the surface of the ground.

Most of the time this can mean a simple addition to the immersion to the game. Whether the character has its legs on the ground or clip through it often does not affect the game in any way. If the game's design requires foot IK for its mechanics, a lot more can be done with it. (Massoudi 2015).

In practice real-time IK for feet can be seen on uneven ground. Baked animation is often made to be used either on even ground or with specific set pieces. When taken to a location with uneven ground these animations encounter problems with feet clipping through other objects or the terrain itself. A foot and the environment around it do not have any contact with each other, so IK is needed to add the interaction between the two. With some programming the end effector is placed on the surface of the ground. The baked animation is then often overridden with the new IK calculations in the bones it is implemented. In case of feet, the whole legs are often masked from the rest of the bones in the hierarchy. (Massoudi 2015).

Figure 17. Ingame character with real-time IK

In the picture above, the character is depicted with IK off on the left and IK on the right. With no IK, the left foot of the character goes through the rock. With IK, the foot is properly placed on top of it. This type of system very common to see in modern games. (Lantz n.d) While having the foot clip through objects may not be a huge issue for the gameplay, it is still a good method for creating more immersion and visual polish to the game.

Position is important, but because the surface may not be horizontal rotation should not be forgotten. Even if the foot is placed on the surface of the ground, that only happens at that single point where the bone is placed. That single point is placed in the correct position, but the rest of the foot may still poke inside the ground. To solve this problem the foot must be rotated to match the angle of the surface of the ground. This can be solved by taking the world normal of the ground and then calculating the amount of rotation needed for the foot's bone.

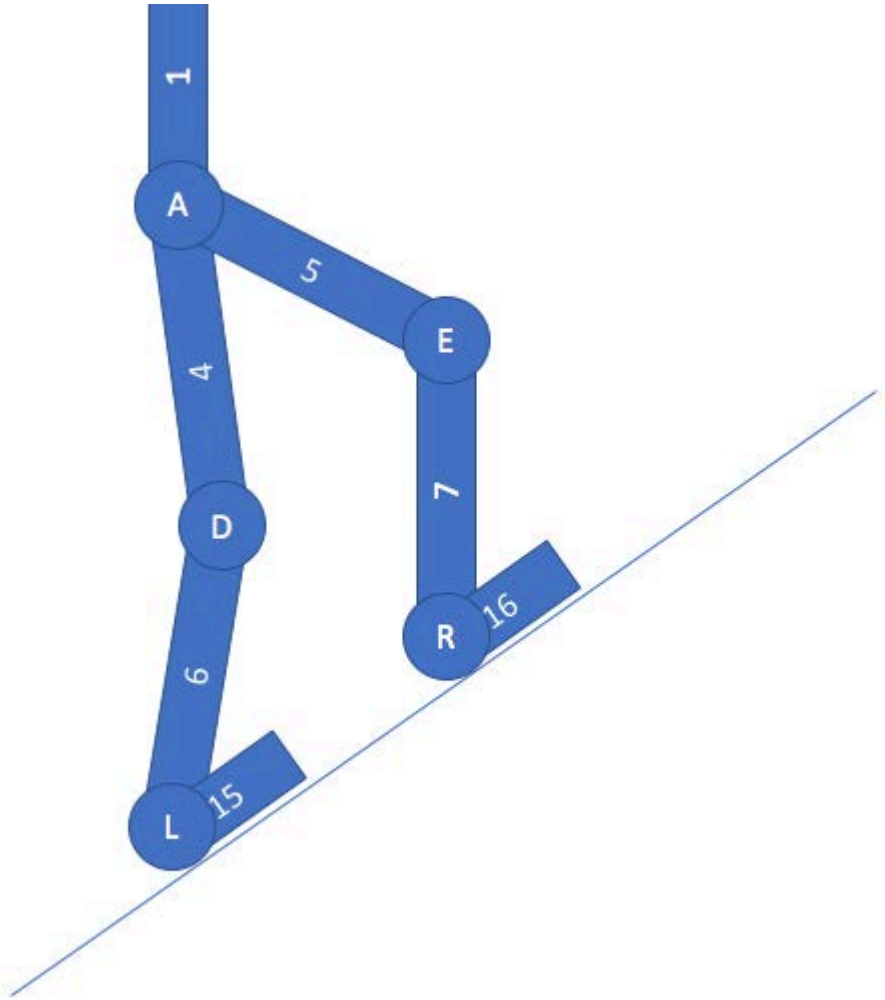

Figure 18. Rotating feet to match the ground's normal

Even if the main purpose of the IK is for feet, it does not necessarily have to be restricted to the bones in the legs. When having your feet on two different surfaces, it affects the whole body. The center of mass moves, and the weight is placed unevenly. One of the most important places this affects is the pelvis. By rotating the pelvis bone to match the height different, even with lower influence, the pose of the character may seem much more realistic. Body parts are not all independent from each other and movement in one

may affect another. Sometimes simply separating legs from the rest of the body might provide weird looking animations.

4.2 Hands and grabbing

Hands can make use of IK just like feet can. In some situations, a hand needs to be placed in a certain position or grab a certain target. One example being a door handle. By placing the end effector on the handle, it is possible to animate the rest of the body while still having the hand in a proper position. Of course, mixing IK with baked animation must be implemented properly so that the change from baked to IK is not noticeable. It can be implemented by changing the influence of IK slowly instead of instantly. (Spyparty.com 2016).

Hands are not always as popular a target for IK as feet. In most games the act of showing a grab animation is enough to convey what the character is doing. Foot IK is often calculated and repositioned with just a single axis, height. Hand IK may require multiple more axes to place it where it needs to be. In addition, the upper body moves with the arms often. The character is probably not even always being grabbing something, unlike feet which are very often placed on the ground. All of this makes IK not a very high priority to implement in hands.

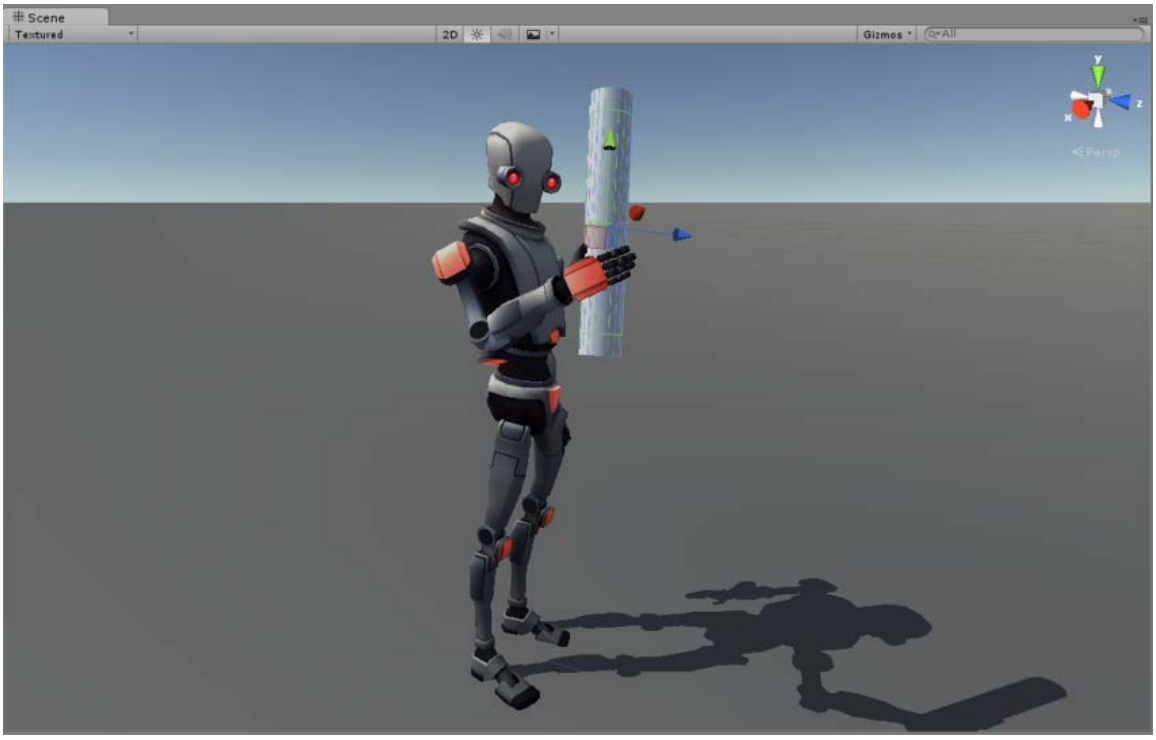

Figure 19. Rig with IK turned towards the target object

Cutscenes that are pre-rendered are different from in-engine. In-engine scenes often play out the same as the game looks, so not is done when it comes to accuracy of animations. Pre-rendered, however, tend to make everything just a little more fluid, more accurate and more realistic. Because they are not necessarily rendered with the same engine, different kinds of setups are used, and the animations may be completely custom, allowing for more specific interactions with animations.

4.3 Clothing and accessories

Clothing can be simulated in multiple ways, but an application of bones with IK can be surprisingly simple.

Using control bones for clothes can be useful. Connecting the cloth to a bone in the body might not provide the desired result. In a situation where the clothes need to rotate around a different central point a bone can be created just for the cloth. The cloth bone can then be connected to the body bone to adjust them both from the parent. The connection does not necessarily have to be using IK, however depending on the situation it can be taken

advantage of. There are other constraints that may be better in certain situations. (Cass-bennett 2014).

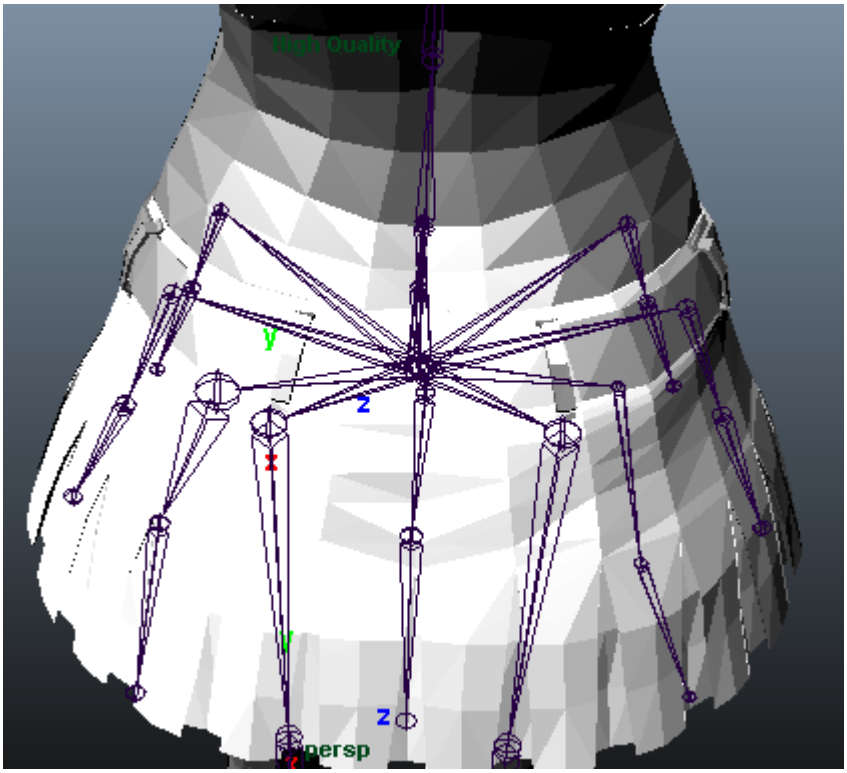

Figure 20. Rigged skirt

Accessories might be a more common target for pure IK on characters besides the body itself. Generally anything hanging or long and bendy can heavily benefit from IK setups in them. For example, handcuffs or chains connected to the character can be simulated using IK. Having the starting position and end effector position known it can be used to simulate the sections in the middle even in real time.

5 Creating new animations in real time

5.1 Procedural animation

When people think about creating animations, a lot of the time they're thinking about baked animations. After all, most games never need anything more than baked animations in the game. As it has been covered before, IK is not actually necessary in many situations. In real time it is often only used as an improvement to give the player a more immersive experience. It is also used in creating the baked in animations, but its functionalities may be lost after baking. There is, however, a case when the point of IK is not to enhance the result, or simplify the animation pipeline, or even be part of the game's mechanics. Procedural animations can fill a totally new type of role.

Procedural animation is a type of animation, that can completely negate the need for an animator to create custom animations for the object. Procedural implies that the animation is generated in real time. While we could call foot IK procedural in that sense, what is usually meant by procedural is that it's not just an enhancement put on top of traditional baked animation, but it is the animation itself. This can also be implemented by mixing up different baked animations together. (Massoudi 2015).

The range of what we call a procedural animation is huge. This thesis mostly focuses on rigs, but procedural is not limited to that. For example, water in games is often created by using shaders. Those shaders then create the animation in real time using textures and shader code.

Creating a water shader often means displacing the vertexes and moving textures across the surface to create an illusion of waves. In real time this is often created by using shaders and directly updating vertex positions. Animations created by these methods are often best for animations that do not require a specific end target. Erratic movement, such as often seen with forces of nature, can be created with this. If we want to have more accurate or mechanical movements as seen in animals, going for animation using rigs and IK might be a better choice.

Procedural animation with IK works well with body parts that work independently. For example, legs of spiders are simple enough to procedurally animate with IK. Their movements are fast and joints simple. There is also no need to worry too much about the center

of gravity so moving them can feel more natural than moving a human's legs. Some games even take advantage of the unnatural feel of full IK animations. There are games built entirely on clumsy and weird movement, which can make the game a fun experience.

Erratic movements such as tentacles can be made much more quickly with IK than with standard forward kinematics. Tentacles are very bendy, so their rigging means lots of small bones to make is flexible enough. Unfortunately, if you were to try to animate such long chains of bones traditionally it would be quite a big load work. Rather than trying to animate it one bone at a time going forward, it's better to have an end effector in place to have a target point for the tentacle. After that, animate the end effector and add movement to the bone chain with other means. (Zucconi 2017).

Figure 21. Tentacles rigged with IK

The movements of the tentacles in the picture above can be animated using creating some randomized movement on top of the IK. The tentacles themselves are made of a long chain of bones in order not to reveal any sharp angles in the joints where it bends. The end effector is clearly created at the ball, which can be moved around as one pleases.

In addition, the tentacles' tips are placed on the surfaces on the ball instead of at the point in the middle of it. They are also rotated, which created the illusion of the tentacles holding the ball.

5.2 A part of game mechanics

Physics based games can take advantage of IK to create important features to the game. Many games have objects like ropes that might behave unrealistically if they had simple baked animations. Objects like that, that are heavily influenced by gravity and swinging, require real-time physics applied to them. Inverse kinematics is one way of implementing that. (Lynn 2017)

Quite a lot of games nowadays have physics-based gameplay. They may be liquid physics or just rigid body physics using gravity. A game like Angry Bids uses physics as a major mechanic with a catapult-like gameplay. Another game called Happy Glass uses fluid physics to create the gameplay. The player needs to guide enough falling water into a glass to win the game. In the end, IK can be part of the main mechanic in a game if the game supports it, or maybe it can even be built around it.

An old example of a game with dynamic IK in real-time is a game that goes by the name "Elasto mania". It is a platformer game that uses physics in its player character to create a lose condition. In the game, the player moves across platforms using a motorbike, and the player driving it sways back and forth according on the momentum. (Bramwell 2008).

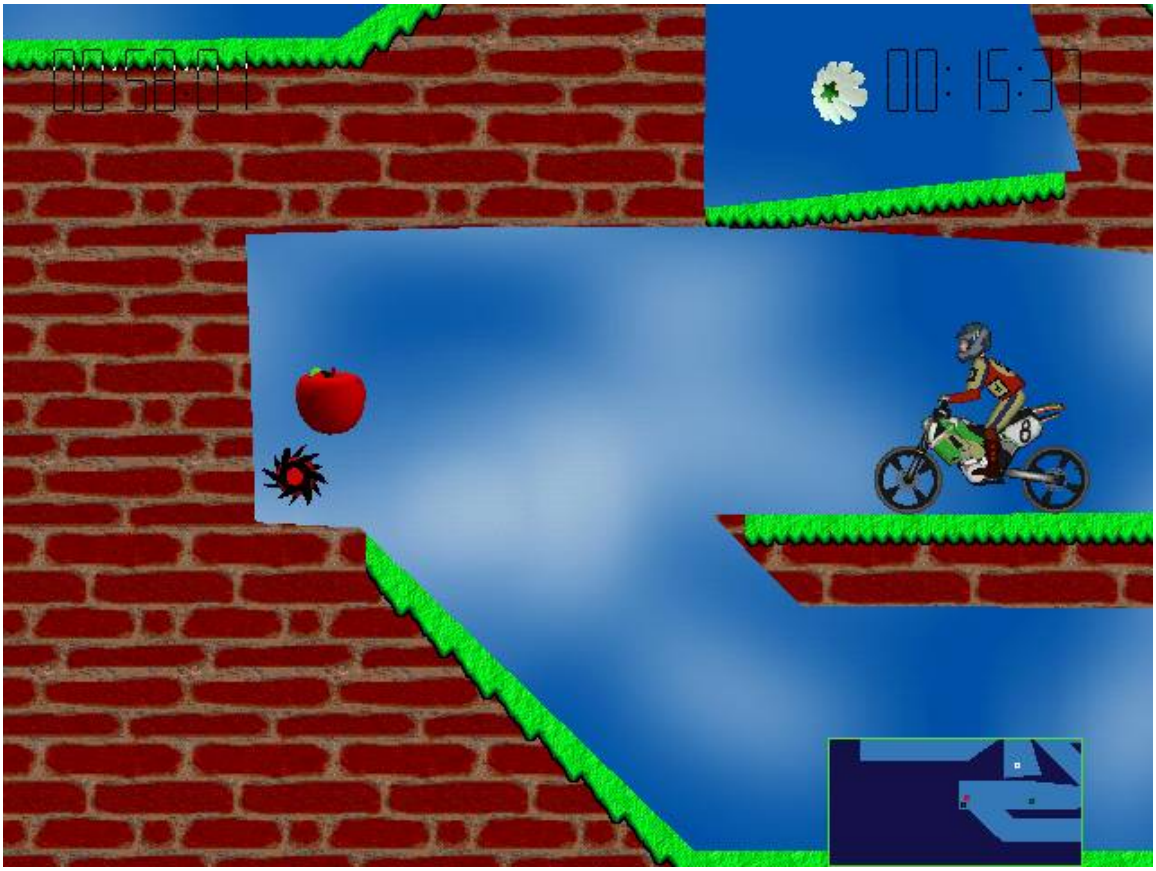

Figure 22: Elasto Mania uses IK in its player as a mechanic

If the player's head touches the ground, the game will be over. If the head lost its sway and simply stayed still the game would lose some of its charm, because everything else about the game follows the momentum.

6 Conclusion

There are many situations that can benefit from the use of IK in different ways. Sometimes IK can simply be used as a tool to help animators create baked animations. Other times it can almost be a requirement to create decent animations. There are some specific uses for IK in most characters that are very popular, notably the character's legs. Creating animations using only forward kinematics can be very restricting. Real-time usage is one of the more modern applications of IK. Simply looking for more immersion or visual polish it can be used on practically every part of the body. Head tracking and foot IK are very often used. Not only are they simple, they are also visible to the player.

The technical side of implementation can be done in multiple ways, but this thesis focused on one simple and one difficult approach. The iterative method consists of using the Jacobian matrix and approximating the end effector through iterative steps. This is a popular method of handling long IK chains that are complex. The analytical approach in this thesis is handled with geometry. The theoretical part of it is easier to understand even with basic trigonometric skills and the implementation is quite simple.

It is best to be aware of the different ways IK can be used, and then select what best fits one's needs. There is no universal answer to what is the best or how it should be done. It is all up to the resources that are available and which advantages and disadvantages are important to the task. This thesis only showcased and examined the possibilities and basic requirements that are possible using inverse kinematics.

7 Sources

Ruskeepaa, Heikki & CTI Reviews. (2016). Mathematica Navigator, Mathematics, Statistics and Graphics. 26.11.2018

Siewert, Sam. (2012). Cloud-based education, Part 3: Cloud-based robotics for education. Retrieved from https://www.researchgate.net/publication/259234959_Cloud-based_education_Part_3_Cloud-based_robotics_for_education 26.11.2018.

Blender 3D: Noob to Pro. (2018). Retrieved from https://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro/Coordinate_Spaces_in_Blender 26.11.2018.

Inverse Kinematics: An approximate real time history. (2018). Retrieved from https://potakudotcom.wordpress.com/2012/11/21/inverse-kinematics-an-approximate-real-time-history/ 26.11.2018.

Aristidou, A.., Chrysanthou, Y., Lasenby, J. & Shamir, A. (2017). Inverse Kinematics Techniques in Computer Graphics: A Survey. Retrieved from http://www.andreasaristidou.com/publications/papers/IK_survey.pdf 26.11.2018.

Doody, Evan. (2016). Advanced Animation Techniques: FK & IK. Retrieved from http://wulverblade.com/advanced-animation-techniques-fk-ik/ 26.11.2018.

Jenkins, Chad. (2018). Inverse Kinematics 1 - Closed-form. Retrieved from http://web.eecs.umich.edu/~ocj/courses/autorob/autorob_10_ik_closedform.pdf 26.11.2018.

Bouchard, Samuel. (2014). Robot End Effector: Definition and Examples. Retrieved from https://blog.robotiq.com/bid/53266/Robot-End-Effector-Definition-and-Examples 26.11.2018.

Wikipedia.org. (2018). Inverse kinematics. Retrieved from https://en.wikipedia.org/wiki/Inverse_kinematics 26.11.2018.

Autodesk Knowledge Network. (2017). Inverse Kinematics (IK). Retrieved from https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2017/ENU/3DSMax/files/GUID-516E301F-E911-429F-9337-9FA7FAD49BB6-htm.html 26.11.2018.

Spyparty.com. (2010). Full Body IK Solver in a Day. Retrieved from http://www.spy-party.com/2010/06/08/full-body-ik-solver-in-a-day/ 27.11.2018.

Massoudi, Peyman. (2015). Foot Placement Using Foot IK. Retrieved from http://peyman-mass.blogspot.com/2015/06/foot-placement-using-foot-ik.html 27.11.2018.

Massoudi, Peyman. (2015). Creating Non-repetitive Randomized Idle Using Animation Blending. Retrieved from http://peyman-mass.blogspot.com/2015/09/creating-non-repetitive-randomized-idle.html 27.11.2018.

Blender 2.80 Manual. (2018). Constraints. Retrieved from https://docs.blender.org/manual/ja/dev/rigging/constraints/introduction.html 16.11.2018.

Liang, Oscar. (2012). Inverse Kinematics Basics Tutorial. Retrieved from: https://oscar-liang.com/inverse-kinematics-and-trigonometry-basics 16.11.2018.

Bermudez, Luis. (2017). 3 Simple Steps to Implement Inverse Kinematics. Retrieved from https://www.gamasutra.com/blogs/LuisBermudez/20170804/303066/3_Simple_Steps_to_Implement_Inverse_Kinematics.php 16.11.2018.

Tristan, Perez & Fossen, Thor. (2005). Kinematics of Ship Motion. Retrieved from https://www.researchgate.net/publication/226128788_Kinematics_of_Ship_Motion 16.11.2018.

Lantz, Peter. Applying IK Constraints in UE4. (n.d.) Retrieved from https://peter-lantz3d.wordpress.com/home-2/problem-solving/applying-ik-constraints-in-ue4/ 16.11.2018.

Applied Go. (2016). Inverse Kinematics: how to move a robotic arm (and why this is harder than it seems). Retrieved from https://appliedgo.net/roboticarm/ 16.11.2018.

Bermudez, Luis. (2017). Overview of Jacobian IK. Retrieved from https://medium.com/unity3danimation/overview-of-jacobian-ik-a33939639ab2 16.11.2018.

Lynn, Zoë. (2017). Custom Rope in Unity3d. Retrieved from http://www.wulcat.com/blog/custom-rope-in-unity3d/ 27.11.2018.

Bramwell, Tom. (2008). Elasto Mania hands on. Retrieved from https://www.eurogamer.net/articles/elasto-mania-hands-on 27.11.2018.

Cassbennett. (2014). Setting Up The Skirt. Retrieved from https://cassbennett.wordpress.com/page/3/ 27.11.2018.

Image Sources

Figure 1: From https://www.pluralsight.com/blog/film-games/key-rigging-terms-get-moving 15.11.2018.

Figure 2: From https://www.pluralsight.com/blog/film-games/key-rigging-terms-get-moving 15.11.2018.

Figure 3: From https://www.pluralsight.com/blog/film-games/key-rigging-terms-get-moving 15.11.2018.

Figure 4: From http://chamilo2.grenet.fr/inp/courses/ENSIMAG4MMG3D6/document/resources/instructions/practical_06.html 16.11.2018.

Figure 6: From https://peterlantz3d.wordpress.com/home-2/problem-solving/applying-ik-constraints-in-ue4/ 15.11.2018.

Figure 7: From https://lesterbanks.com/2013/11/blender-creating-ik-pole-vectors-without-breaking-the-bind-pose/ 15.11.2018.

Figure 8: From https://medium.com/unity3danimation/overview-of-inverse-kinematics-9769a43ba956 16.11.2018.

Figure 9: From https://medium.com/unity3danimation/overview-of-inverse-kinematics-9769a43ba956 16.11.2018.

Figure 10: From https://www.gamasutra.com/blogs/LuisBermudez/20170804/303066/3_Simple_Steps_to_Implement_Inverse_Kinematics.php 16.11.2018.

Figure 11: From https://appliedgo.net/roboticarm/ 16.11.2018.

Figure 12: From https://appliedgo.net/roboticarm/ 16.11.2018.

Figure 13: From https://appliedgo.net/roboticarm/ 16.11.2018.

Figure 14: From https://www.researchgate.net/publication/262046990_A_Ship_Simulation_System_for_Maritime_Education 16.11.2018.

Figure 15: From https://boneandspine.com/degrees-of-freedom-of-upper-limb/ 16.11.2018.

Figure 16: From https://peterlantz3d.wordpress.com/home-2/problem-solving/applying-ik-constraints-in-ue4/ 16.11.2018.

Figure 17: From https://peterlantz3d.wordpress.com/home-2/problem-solving/applying-ik-constraints-in-ue4/ 16.11.2018.

Figure 18: From https://blender.stackexchange.com/questions/71088/what-is-inverse-kinematics 16.11.2018.

Figure 19: From https://docs.unity3d.com/Manual/InverseKinematics.html 16.11.2018.

Figure 20: From https://cassbennett.wordpress.com/page/3/ 16.11.2018.

Figure 21: From https://www.alanzucconi.com/2017/04/12/tentacles/ 16.11.2018.

Figure 22: From http://www.pelikulma.net/pelit/urheilu-auto-ja-tappelupelit/elasto-mania/ 16.11.2018.