Character Creation and Procedural Animation

Danilo Catalan Canales
Daniloc@kth.se

DH2323 - DH2323 Computer Graphics and Interaction

Abstract

This project displays the work of creating a model, rigging up the character with bone constraints and by coding provide an animation sequence. The work concludes with prospects and setbacks during the process. The programs used are blender for modeling and bone-rigging. Unity was used to work out the animated process.

Character Creation

The approach was very straightforward. I started off with a 2D drawn sketch of a character from a front and side view. The sketch was drawn in microsoft paint and later imported to a 3d modelling and animation program called blender. I positioned both sketches along the X and Z axis. From this point on I created a 3D model by stretching out a mesh so it matched the sketch.

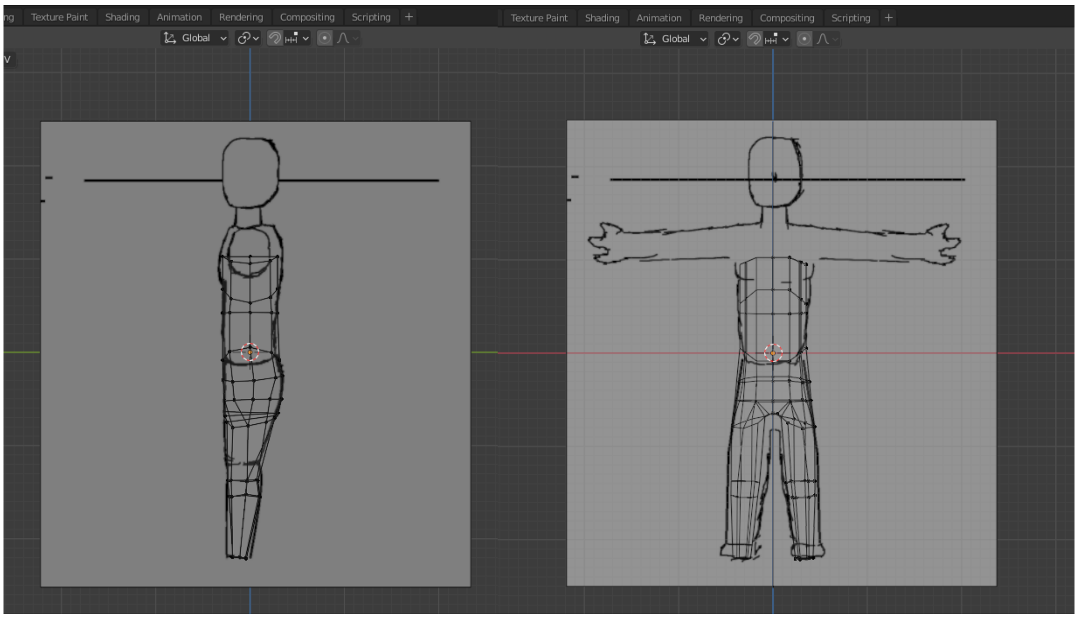
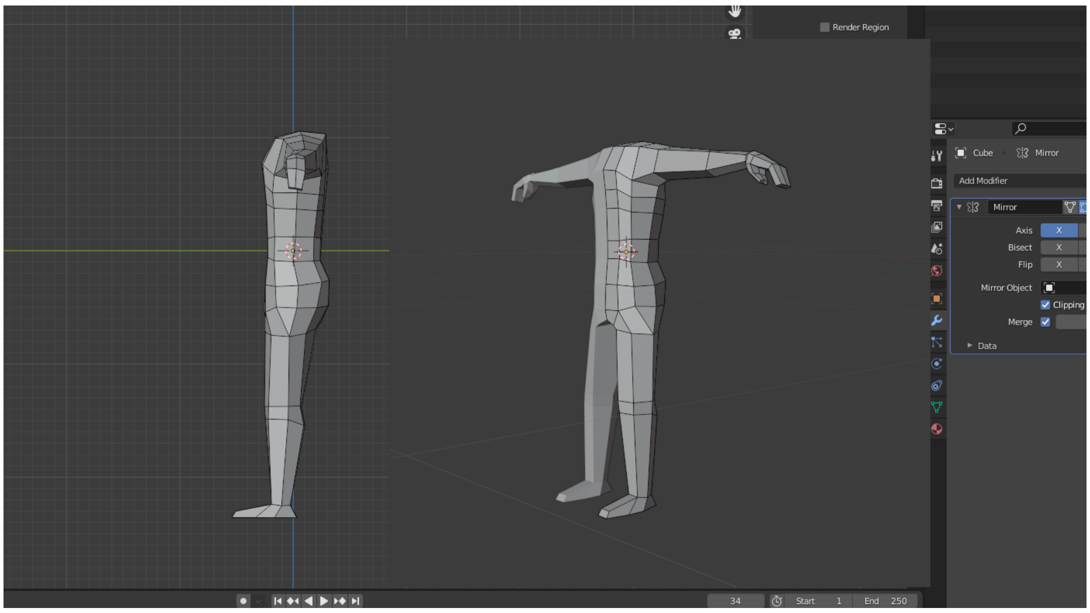


Figure 1:
Here we can see the sketches being aligned and the polygon being modelled after the sketch

Some changes were made along the way. The hand was changed to ease the creation of the character. So for the change I made a type of "glove hand" with no separate fingers, and removed the head entirely. Below we see the final product without the bone rig. We can see that the character has kept the integrity of the sketch and is prepared for its' bone structure set up.



Bone Rigging

As for my strategy for the bone rigging I took the conventional human body as reference to set up a similar build for my character. I first started with the main structure from the abdomen up to the chest area. From there I branched off to the legs and arms. The intent was to add as many “main” bones as the human body is made out of. The difference is that the fewer bones the better for the character, as more added bones will put more strain on the deformation of the model. This could later make the animation process more tedious, which is why we will stick to fewer bones with higher importance.

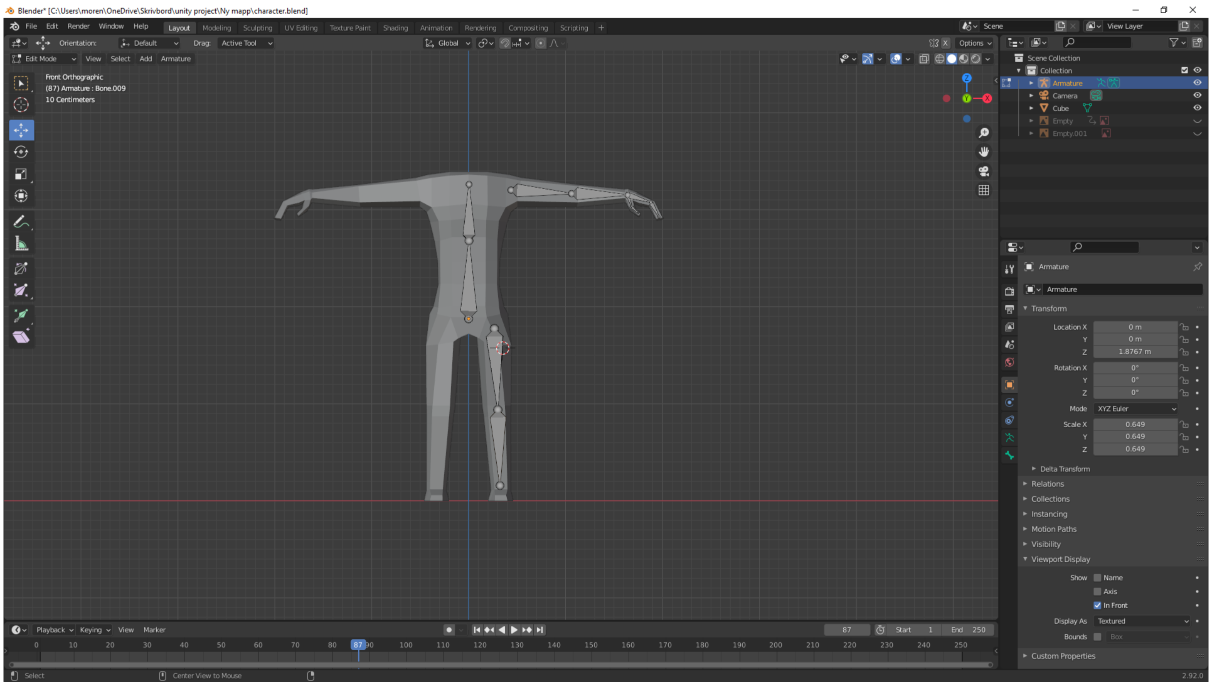


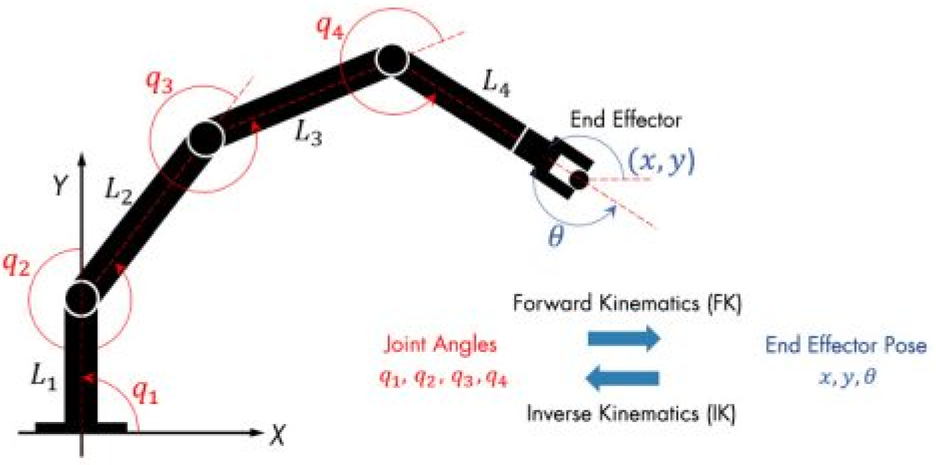
Figure 2:
Displays the bone structure of the character. The right side is later mirrored to the left side.

Inverse Kinematics (IK)

For the next segment I use unity’s own inverse kinematic rigging which will aid me in constructing an algorithm for the characters movement.

Inverse kinematics allow for the movement from the outer limb to control the movement as we go in through the bone hierarchy. The joint angles are calculated as we move the outer limb (the feet in this case), so that a more fluent motion is made.
This can be thought of as a robot arm, intending to move in 3D space. As the end-effector moves towards a target the inverse kinematics will determine how the joints will comply to not obstruct any joint constraints.

Below we can see how the different angles are considered as the end of the arm is moved. This is an optimal way to move the character limbs around, since the legs and arms move roughly the same way.

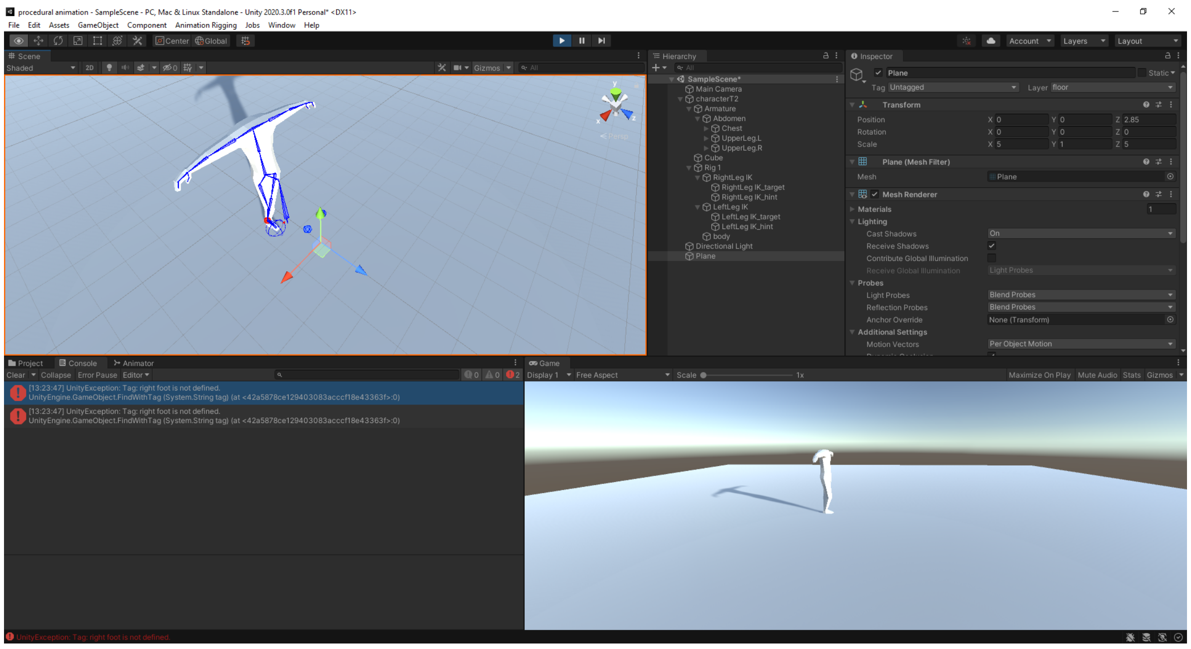


https://www.mathworks.com/discovery/inverse-kinematics/_jcr_content/mainParsys/image.adapt.480.medium.jpg/1620305507858.jpg

Procedural Animation

Character animation is mostly made as a premade asset where, if needed, the character will perform the sequence. However, these sequences can make the character movement look very weird or unrealistic in some situations. These premade animations could also pile up in an extensive library. This is why a more general approach is more than welcome to see the different aspects of animation. An animation that considers geometric deformation of the terrain and responds accordingly. This would hopefully give a more realistic feel to the character which could make, for example a game, feel more realistic.
However, even though the idea is made clear, the behaviour can vary depending on what the creator is looking to achieve.

In my case I would like for the character to make a simple walking sequence where it will consider the floor and some squares as an obstacle course. And for the simplicity of it I would make it go in a straight line to not have rotation as an additional impediment. Also gravity and body upwards force is also discarded, though if time was not of essence this would have been a good addition to the animation.


procedural animation - SampleScene - PC, Mac & Linux Standalone - Unity 2020.3.0f1 Personal* <DX11>

```
SampleScene*
  Main Camera
  characterT2
    Armature
      Abdomen
        Chest
        UpperLeg.L
        UpperLeg.R
    Cube
    Rig 1
      RightLeg IK
        RightLeg IK_target
        RightLeg IK_hint
      LeftLeg IK
        LeftLeg IK_target
        LeftLeg IK_hint
      body
  Directional Light
  Plane
```

```
Plane                                  Static
Tag Untagged                           Layer floor
Transform
  Position    X 0    Y 0    Z 2.85
  Rotation    X 0    Y 0    Z 0
  Scale       X 5    Y 1    Z 5
Plane (Mesh Filter)
  Mesh                          Plane
Mesh Renderer
  Materials                     1
  Lighting
    Cast Shadows                On
    Receive Shadows             ✓
    Contribute Global Illumination
    Receive Global Illumination Light Probes
  Probes
    Light Probes                Blend Probes
    Reflection Probes           Blend Probes
    Anchor Override             None (Transform)
  Additional Settings
    Motion Vectors              Per Object Motion
```

```
[13:23:47] UnityException: Tag: right foot is not defined.
UnityEngine.GameObject.FindWithTag (System.String tag) (at <42a5878ce129403083acccf18e43363f>:0)
[13:23:47] UnityException: Tag: right foot is not defined.
UnityEngine.GameObject.FindWithTag (System.String tag) (at <42a5878ce129403083acccf18e43363f>:0)
```

Figure 3:
Here we can see the character in unity with the bone rig set up.

Method for the sequence

The idea to find a way to make the character move was derived from walking myself and then making up some observations for the code.

<u>Observations:</u>

- A person anticipates the leg position as we lean forward
- If we move the body position just a bit we adjust the legs accordingly
- If we move just a bit we don't lift our knees up to the waist.
- We don't move both legs at the same time.

These observations went on to become the following pseudo code:

```
while body moves:
        Anticipate next step.

        IF distance is far enough from previous position:
                the position of the anticipated step will be the next step.

        If the other foot is not moving:
                take the step.
```

This however resulted in some difficulties. The idea was simple but didn't apply any rules to when the person was standing still. If no leg was moving then both would take a step at the same time. A jumping motion was made.
For the adjustments I made a turn based system where the "foot turn" would be switched. So the character would not have a dominant foot to start with and the turn would switch even though standing still.

```
while body moves:
        Anticipate next step

        IF distance is far enough from previous position:
                the position of the anticipated step will be the next step.

        If the other foot is not moving and myTurnToMove:
                take the step.
```

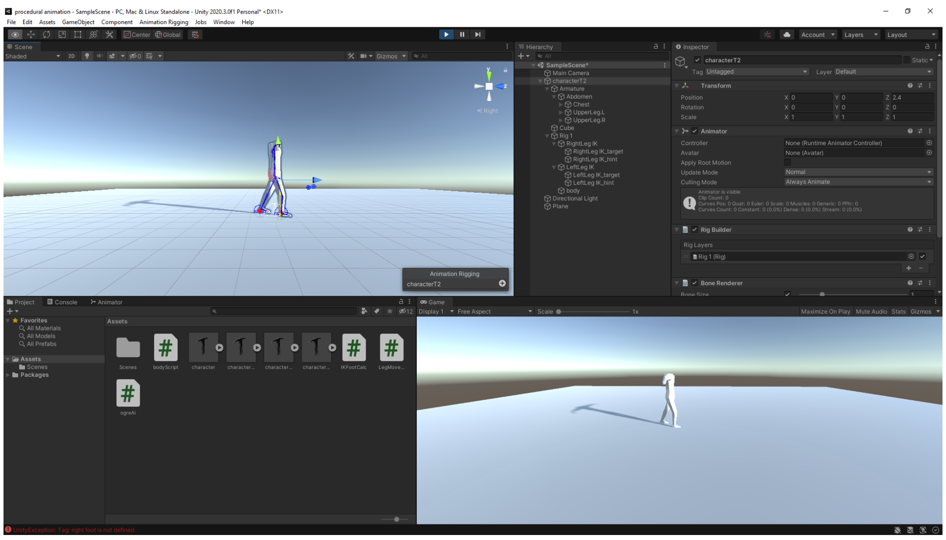

characterT2
Tag Untagged   Layer Default
Transform
Position X 0 Y 0 Z 2.4
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Animator
Controller None (Runtime Animator Controller)
Avatar None (Avatar)
Apply Root Motion
Update Mode Normal
Culling Mode Always Animate
Animator is visible
Clip Count: 0
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0 Generic: 0 PPtr: 0
Curves Count: 0 Constant: 0 (0.0%) Dense: 0 (0.0%) Stream: 0 (0.0%)
Rig Builder
Rig Layers
Rig 1 (Rig)
Bone Renderer

Figure 4: first step for the character

After applying the leg movements I added leg correction where, if not moving, the legs would return to its original position. The following adjustments were made for the pseudocode

```
while body moves:
        1.Anticipate next step

        2.IF distance is far enough from previous position:
                3.the position of the anticipated step will be the next step.

        4.If the other foot is not moving and myTurnToMove:
                5.take the step.

if not moving:
        7.Next step is the position next to the center of mass
        8.Move the feet to next position
```

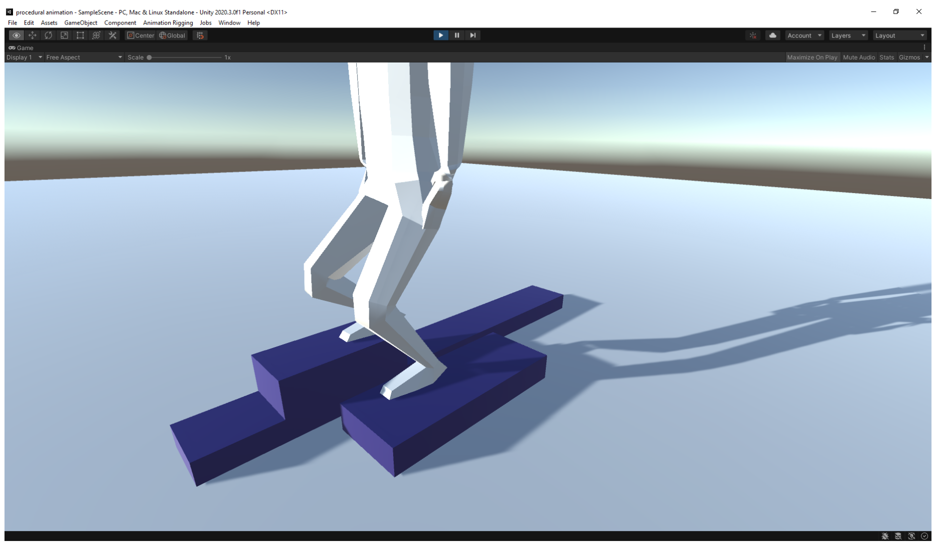
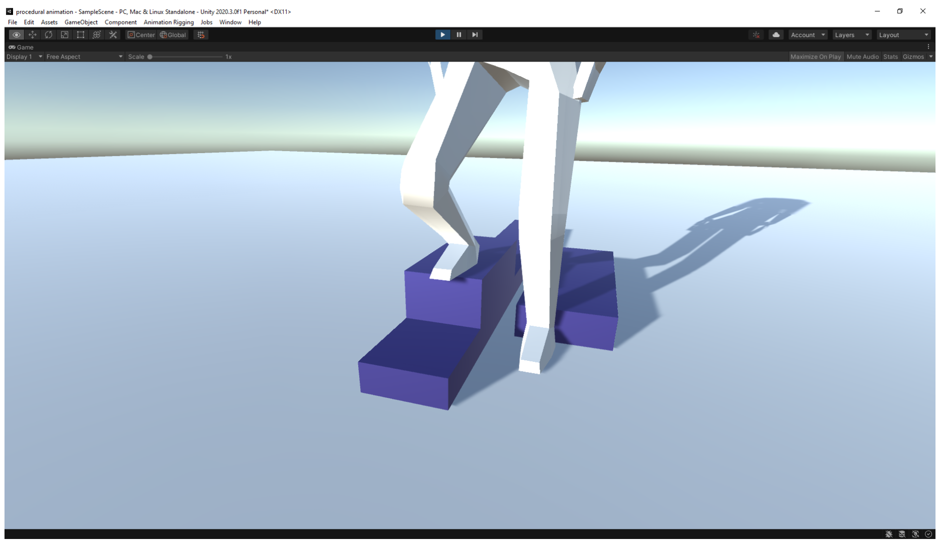
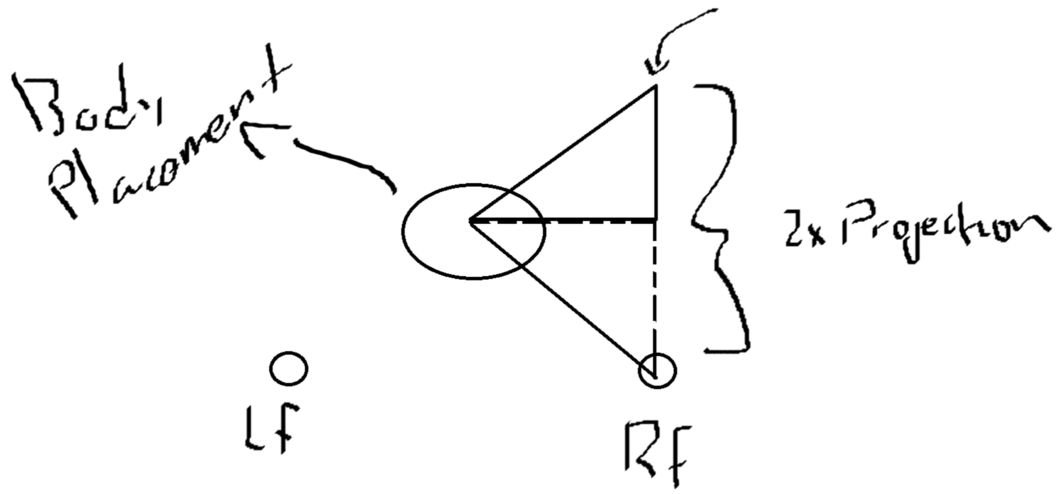



Figure 5 & 6:
After adding some obstacles we see how the character takes the terrain in consideration. The legs are also centered.

Conclusion

The process of the whole project was fairly straight forward. The creation process had more or less a lot of fidgeting. The inverse kinematics could also have been done in the creation process but to save some time I decided to use the unity made IK setup. However, this did not simplify the animation process. The model was not centered at first which made vector calculation very troubling. So to fix this I had to go back to the blender program and fix any issues with the character position, since the program takes the placement into consideration when exporting it.

Further on, I originally thought of using projection to calculate the next position of the feet:



However, this tested the symmetry of the character and when walking further and further the calculation for the next step deviated even more. This separated the feet as the character was walking further and further.
So I decided to just have an offset vector that followed the body position and the next position of the feet would be calculated based on the direction and the distance difference that the body has made.

Since the IK target, that the feet follow, is put on the bone rig, I could see that the feet would sink below the plane or obstacles. This was simply because the feet outer body mesh was not considered when going through the movement. However, I did put a slight offset from the IK target so that it would visibly stay above.

Finally, the end results are not completely realistic but it does resemble the movement to a certain degree and the character was able to pass the obstacle course which was my intended goal.

References

https://robotacademy.net.au/lesson/inverse-kinematics-for-a-2-joint-robot-arm-using-geometry/

https://www.mathworks.com/discovery/inverse-kinematics.html

https://www.diva-portal.org/smash/get/diva2:1018821/FULLTEXT01.pdf

https://www.alanzucconi.com/2017/04/17/procedural-animations/