Using Inverse Kinematics in Blender & Unity

Kyle Bowden

April 3, 2022

1 Introduction

1.1 Overview

This paper acts as a small tutorial for using inverse kinematics in Blender 2.8.x and Unity 2019.x. It will discuss how to create bone chains with inverse kinematic targets in Blender to generate constrained and realistic movement. Then it will walk through how to export the bones and mesh from Blender and import them into Unity using FinalIK to create Procedural Animations. Finally, there will be an example of creating a walking creature using Procedural Animations.

1.2 Required Software

- Blender 2.8.x
- Unity 2019.x
- A Unity IK Package such as FastIK or FinalIK.

1.3 Prerequisite Knowledge

- Basic understanding of local transformations vs world transformations.
- Basic understanding of parent-child object hierarchies in regards to transformations.
- Basic understanding of Meshes in Computer Graphics

If either of these are foreign concepts I suggest watching *The Essence of Linear Algebra* series by 3Blue1Brown, which can be found here: Playlist Link

1.4 Skeletal Animation in a Nutshell

Animating characters is usually done using skeletal bones and skinned meshes. Skeletal bones are fairly intuitive as they can be jointed together like normal bones and then have a skin around them. For example, an arm is comprised of an arm bone connected to a forearm bone. These bones can then be attached to a mesh such that the mesh is a skin for the bones. Therefore as the bones are moved so is the mesh. A good interactive example of this is the SkinnedMesh functionality in Three.js: Simple Mesh with Bones Example. Click around in the *Controls* menu and try modifying the bones to see what happens to the mesh.

2 Forward & Inverse Kinematics

2.1 Forward Kinematics

Forward Kinematics utilize bone chains in a top-down fashion. Parent bones are moved prior to children bones in order to generate an animation.

Using the arm example from earlier, this is analogous to moving the arm and then moving the forearm, and finally the hand to reach a desired position.

2.2 Inverse Kinematics

Inverse Kinematics is different from Forward Kinematics as the desired position of the final child bone in the chain is known. Each parent bone in the chain can then be moved within given physical constraints to generate an animation.

For the arm example, this means moving the hand to a desired position such that the forearm and arm are naturally positioned also.

3 Some Blender 2.8 Tips

3.1 Pay Attention to The Current Mode

There are three modes this tutorial will be using: Object Mode, Edit Mode, and Pose Mode.

- **Object Mode** is used to add in new objects to the scene.
- **Edit Mode** is used to change properties such as vertices, faces, and local transformations of the object such as scale, rotation, and translation.
- **Pose Mode** is used to modify the bones once they are ready to be positioned and have other physical constraints applied such as Inverse Kinematic targets and Inverse Kinematic poles.

3.2 Hotkey Cheat Sheet

Blender 2.8 has **A LOT** of hotkeys, even after a year of active use it would still be difficult to understand them all. However, here is a useful link for some common hotkeys that will make life much easier: Blender 2.8 Shortcuts

Otherwise a particularly useful hotkey set is the Numpad keys, these keys orient the view instantly such as looking through each axis to get a view of a given axis plane.

3.3 Initial Setup

When starting up a new *General Project* note how there is a Camera, Cube, and Light in *Scene Collection* over at the right. It should resemble Figure 1.

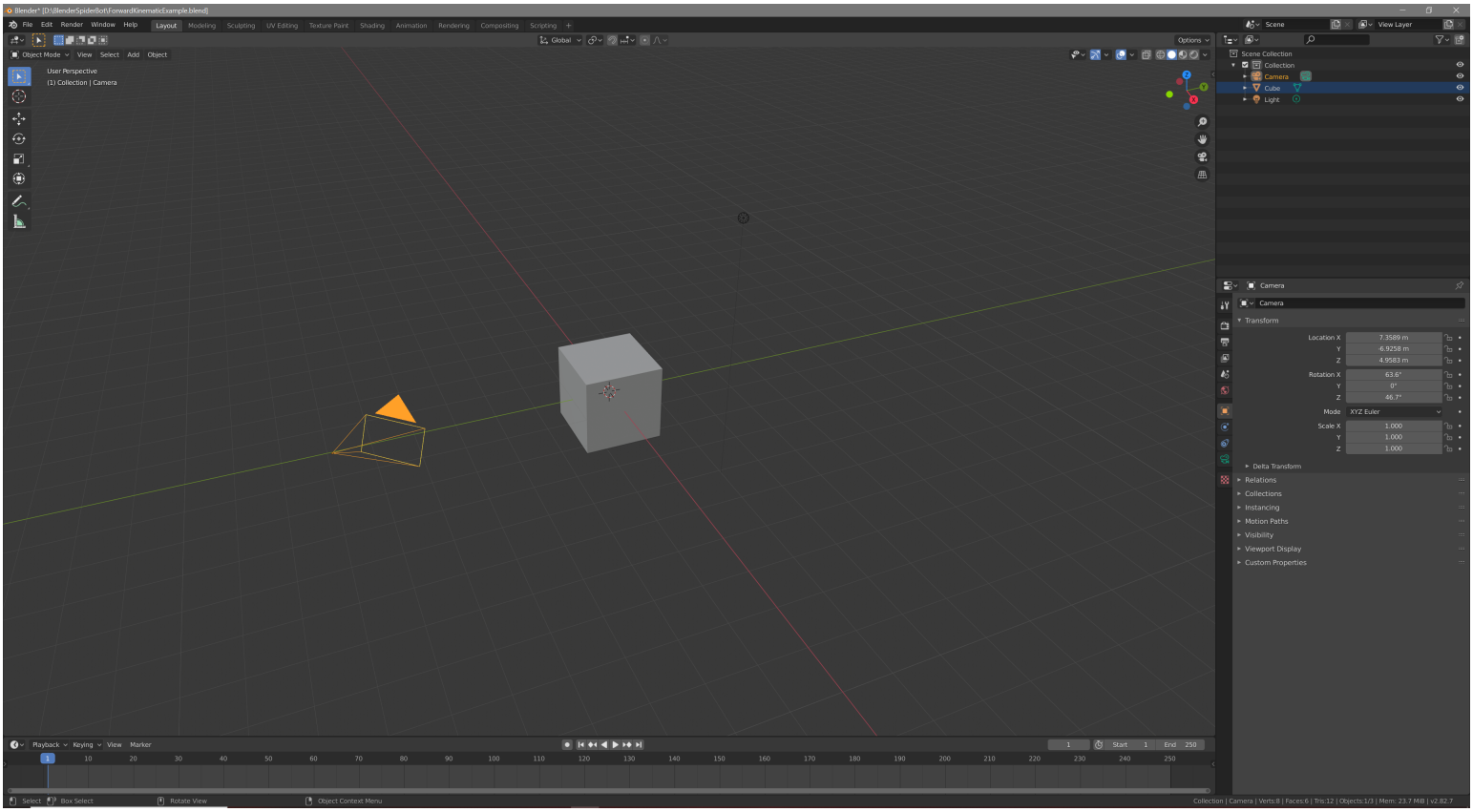

Figure 1: General Project

However the Camera, Cube, and Light are not needed for this tutorial so they should be deleted. The *Scene Collection* should then look like Figure 2.

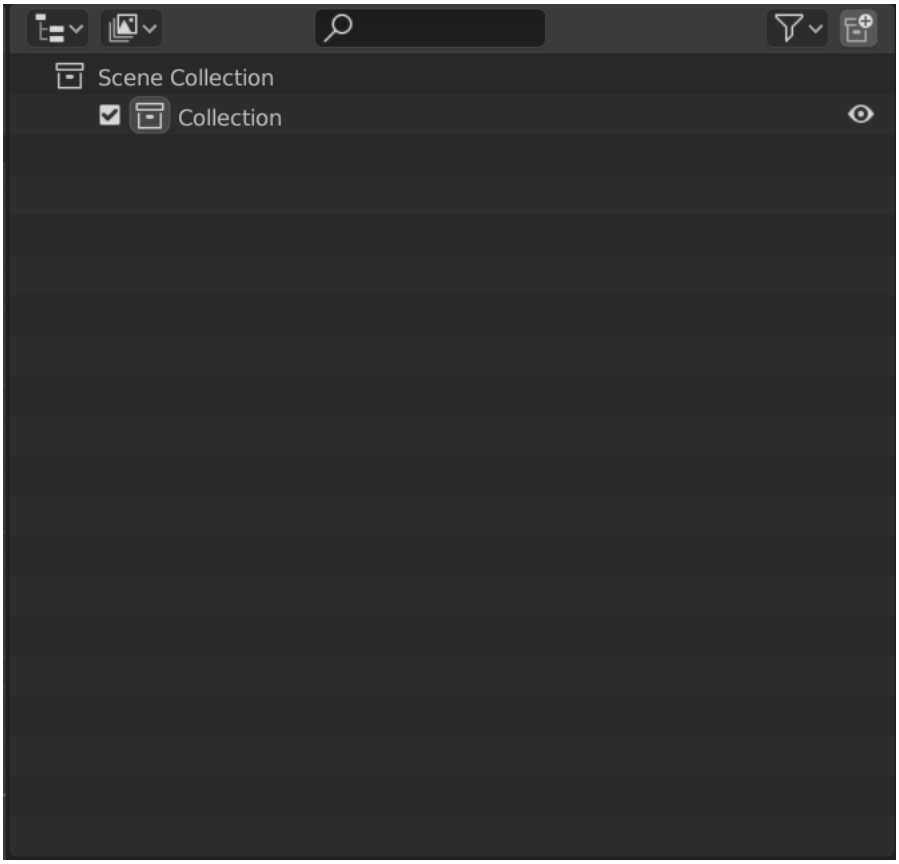

Figure 2: Empty Scene Collection

4 Creating Bones in Blender

4.1 Create a Forward Kinematic Bone Chain

4.1.1 Add a New Armature

In order to create a bone chain, there needs to be an armature. In this example it will start with a single bone. In **Object Mode** go to *Add* → *Armature* → *Single Bone* as seen in Figure 3.

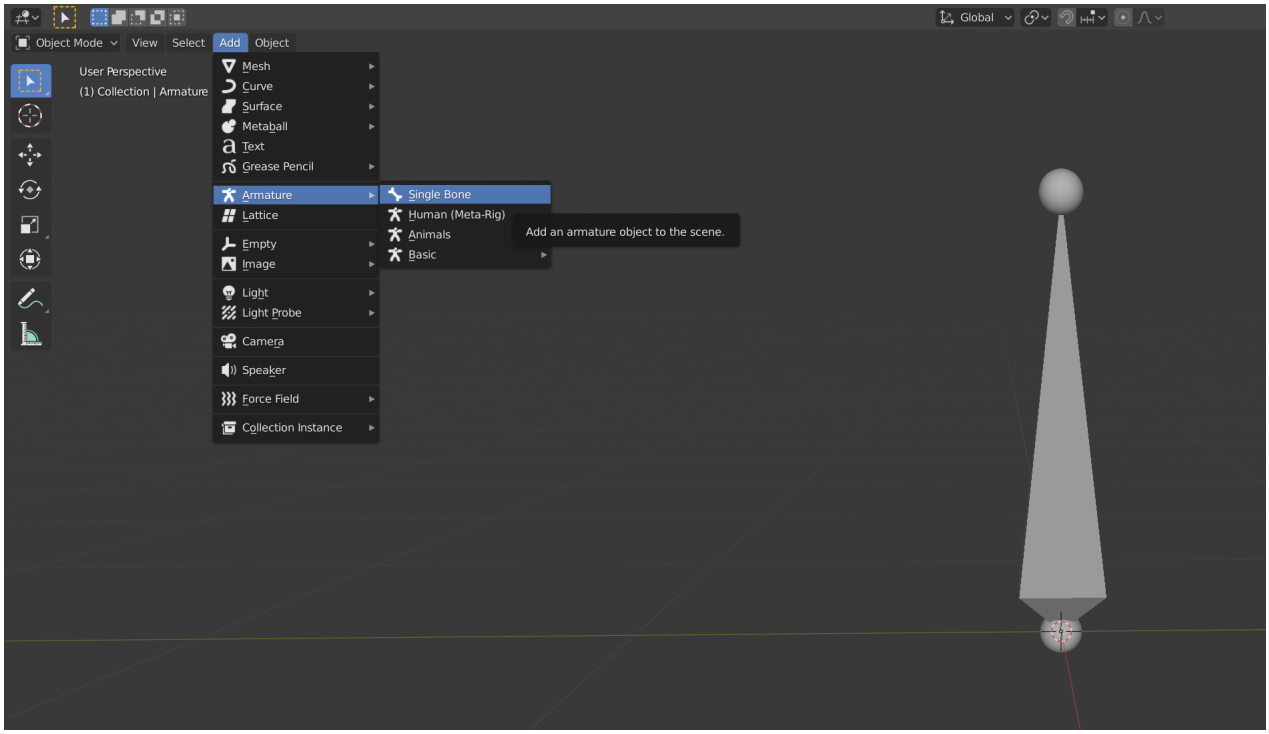

Figure 3: Add Armature

4.1.2 Extrude New Bones Off Single Bone

Extruding new bones off of the single bone automatically parents the newly extruded bones to the single. In order to do this first enter **Edit Mode** by first clicking on the single bone while in **Object Mode** then clicking on the drop down list where **Object Mode** is currently displayed and select **Edit Mode**. This is the same for switching into **Pose Mode** also. Reference Figure 4.

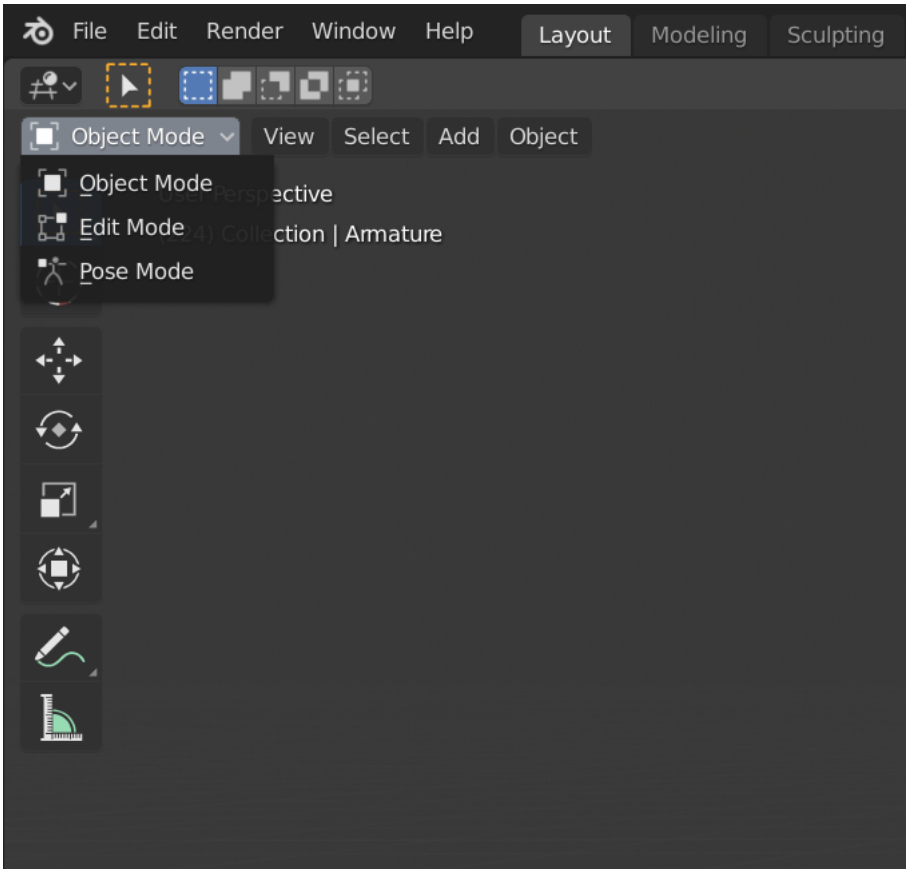

Figure 4: Switch Mode

Once in **Edit Mode** note the extrusion tool on the left, reference the tool highlighted in blue in Figure 5. From here, to create new bones just click and drag the yellow circle with a plus in the middle. This process can be repeated as many times as needed to create the ideal bone chain. An example bone chain can be seen in Figure 6.

4.1.3 Go Into Pose Mode

Now that the bone chain is created go into **Pose Mode** as described earlier and use the *Move Tool* to try posing the bone chain. Note how moving the last bone in the chain doesn't modify any of the other bones. This is because its a forward kinematic chain, where the last bone is the leaf child of the armature's hierarchy.

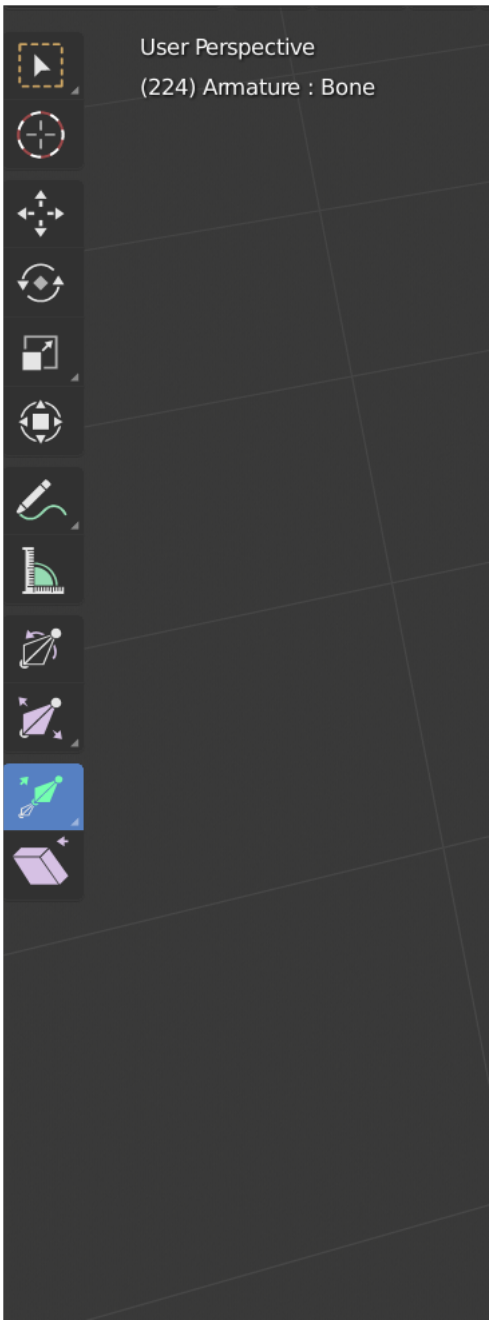

Figure 5: Bone Extrusion Tool

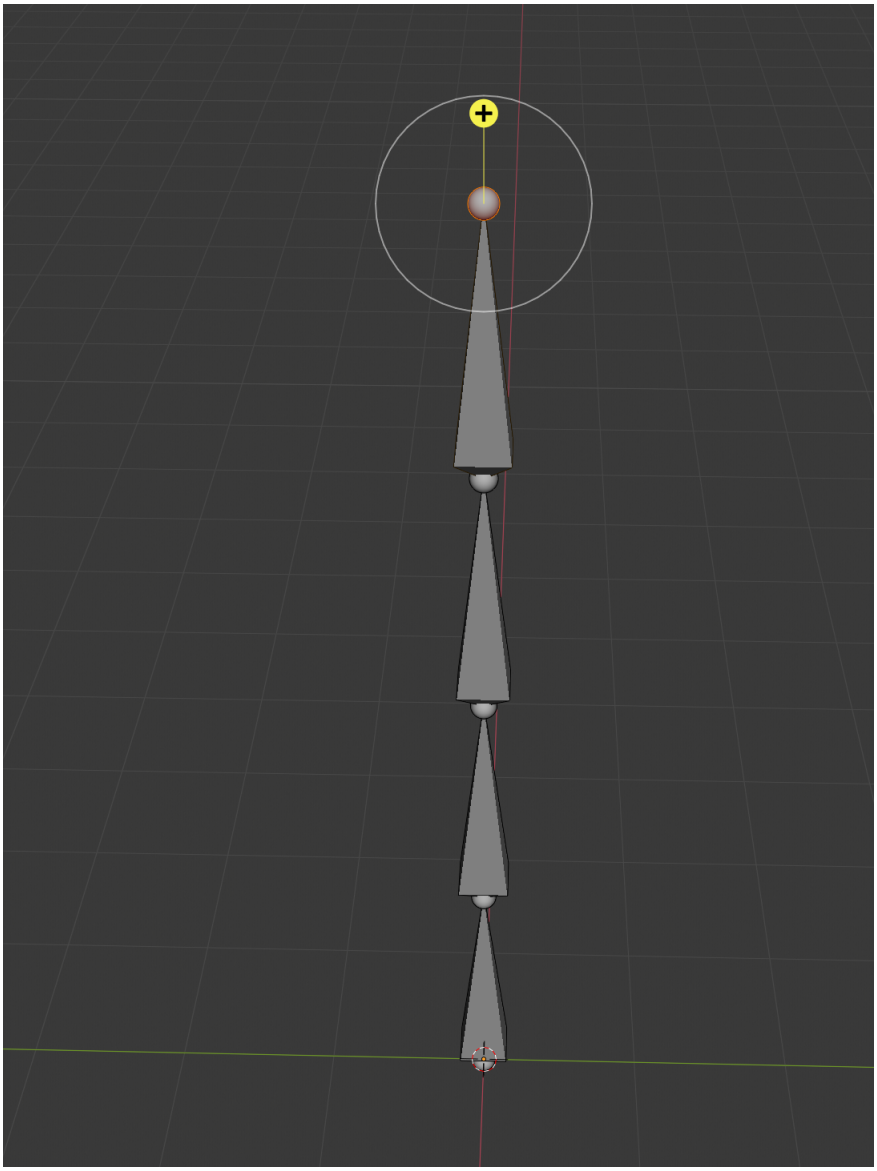

Figure 6: Basic Forward Kinematic Bone Chain

5 Applying Inverse Kinematics

5.1 Natural Movement

Natural body movement usually consists of moving to a targeted position. For example, when reaching for a cup of coffee the first instinct is to move the hand to the handle, not move the arm, then the arm, and the the wrist to grab the handle. This principle is the basis for Inverse Kinematic Targets and Inverse Kinematic Poles.

5.2 Inverse Kinematic Targets

Using the bone chain created to demonstrate Forward Kinematics, an **IK Target** can be created by extruding an additional bone (in **Edit Mode**) from the final child bone of the chain and then clearing the parent of the new bone. This removes the bone's relationship with the chain and makes it easier to create an **IK Target** at a given bone. Reference Figure 7.

Then go into **Pose Mode** and click on the final child bone then go to the *Bone Constraint Properties* tab and add an Inverse Kinematic Bone Constraint, reference Figure 8.

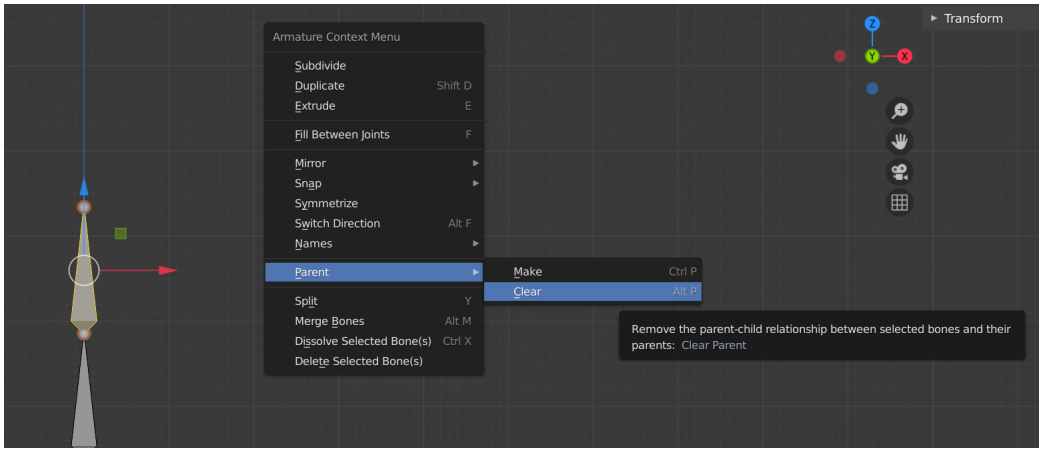

Figure 7: Remove New Bone From Chain

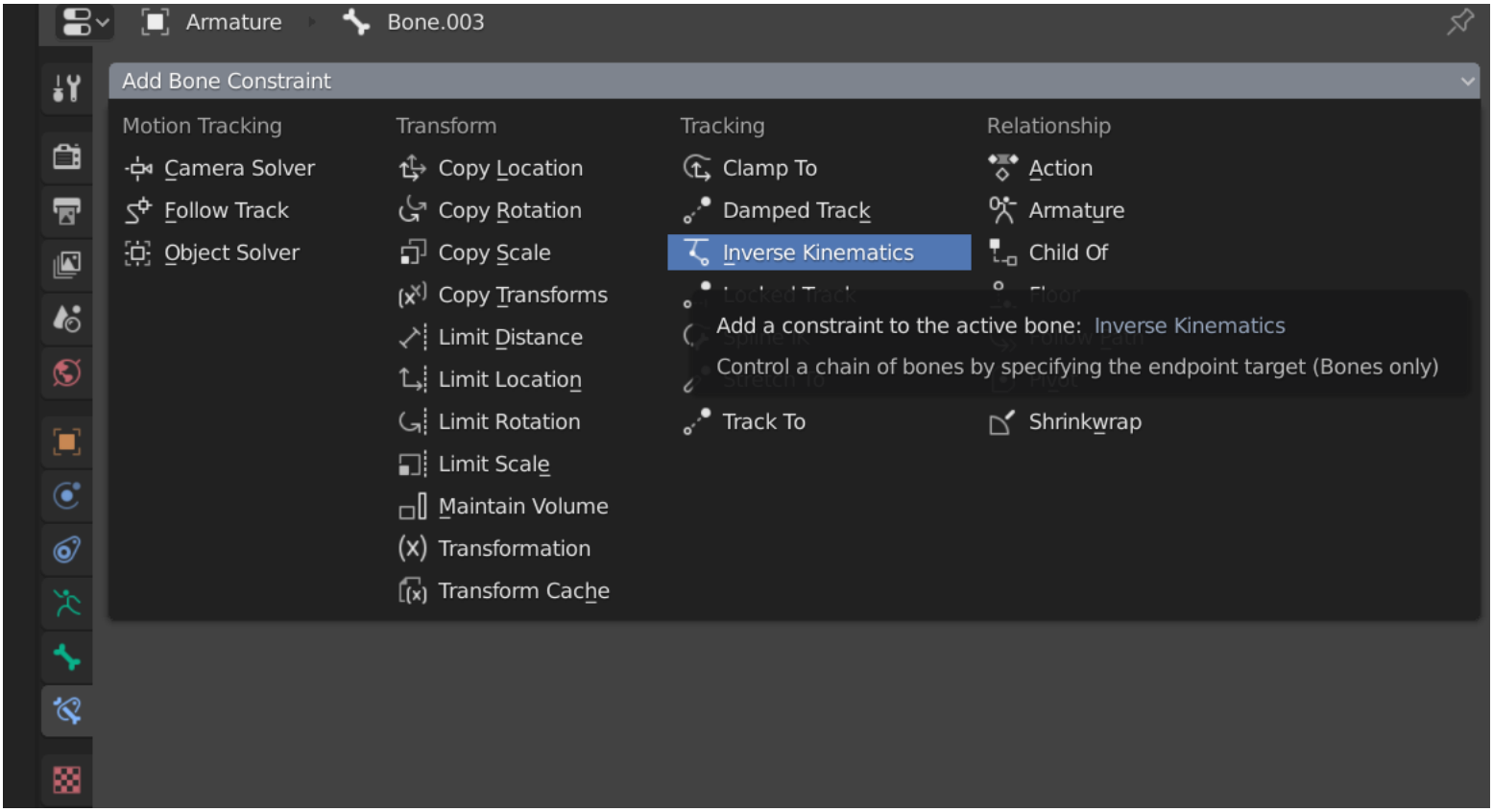

Figure 8: Adding an IK Constraint

5.3 Try to Pose the Bone Chain Now

Now that the child bone has an **IK Bone Constraint** the disconnected bone can now be used to pose the bone chain in a natural way. Reference Figure 9 for how the bone chain may look after moving the target/disconnected bone. While the bone chain is affected by the **IK Target** it still is missing an **IK Pole Target** to generate more ideal constraints.

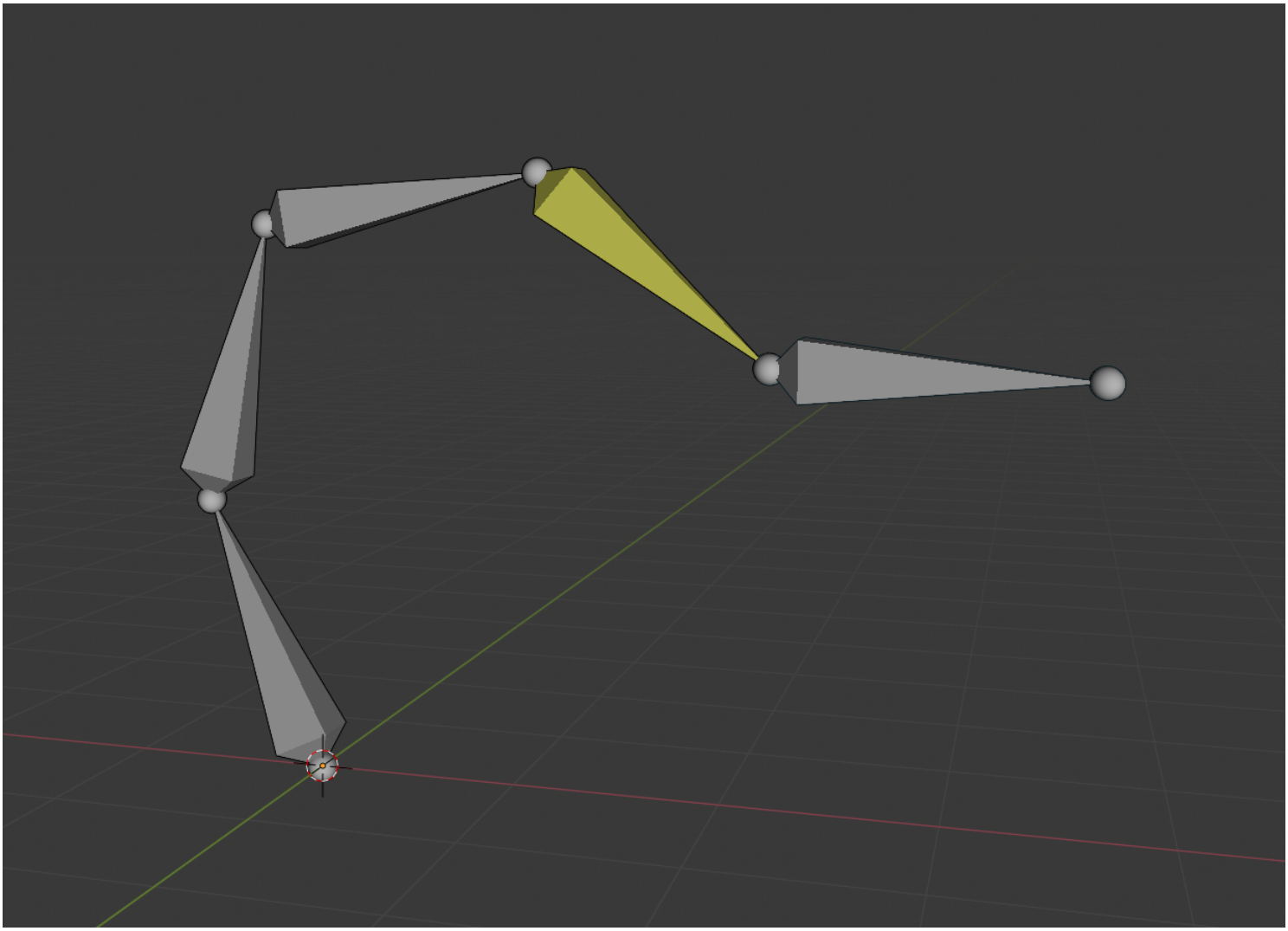

Figure 9: IK Movement Example

5.4 Adding an IK Pole Target

Repeat the process for adding a **IK Target** but instead start from one of the middle bones instead of the last bone. Then go back to the **IK Bone Constraint** tab from earlier and add the pole target as the newly created bone. This is analogous to creating a joint that keeps the middle bone in a constrained rotation and position. Note in the reference Figure 10 that it is

in **Edit Mode** and that the middle bone has been translated towards the **IK Pole Target** as Blender needs to have the middle bone be bent towards the **IK Pole Target** in order to bend properly in **Pose Mode**.

After adding the **IK Pole Target**, the bone chain will know treat the the middle bone as a constrained joint. Reference Figure 11 with Figure 9 for how the pole's position affects the chain.

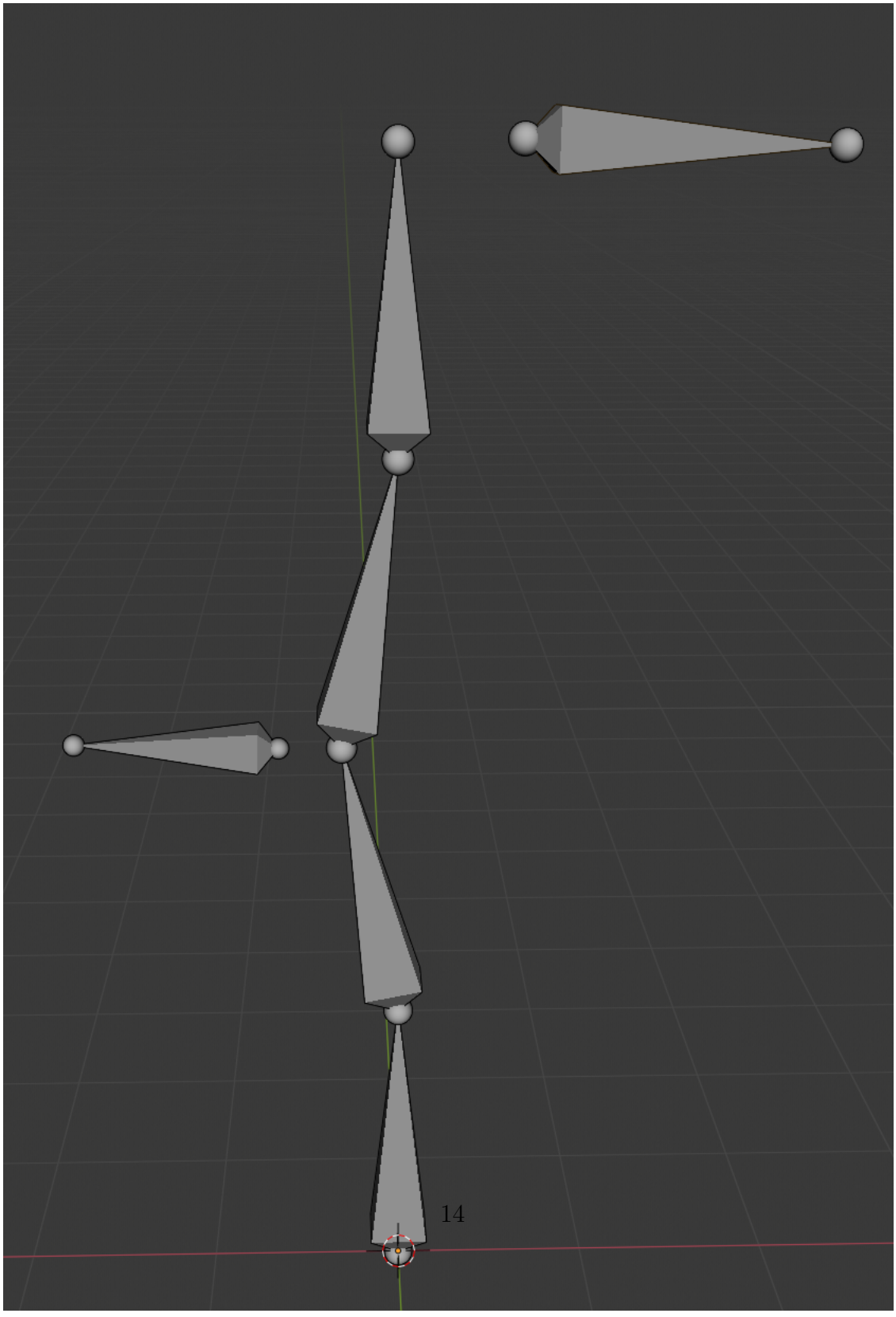

Figure 10: Edit Mode W/ Pole Example

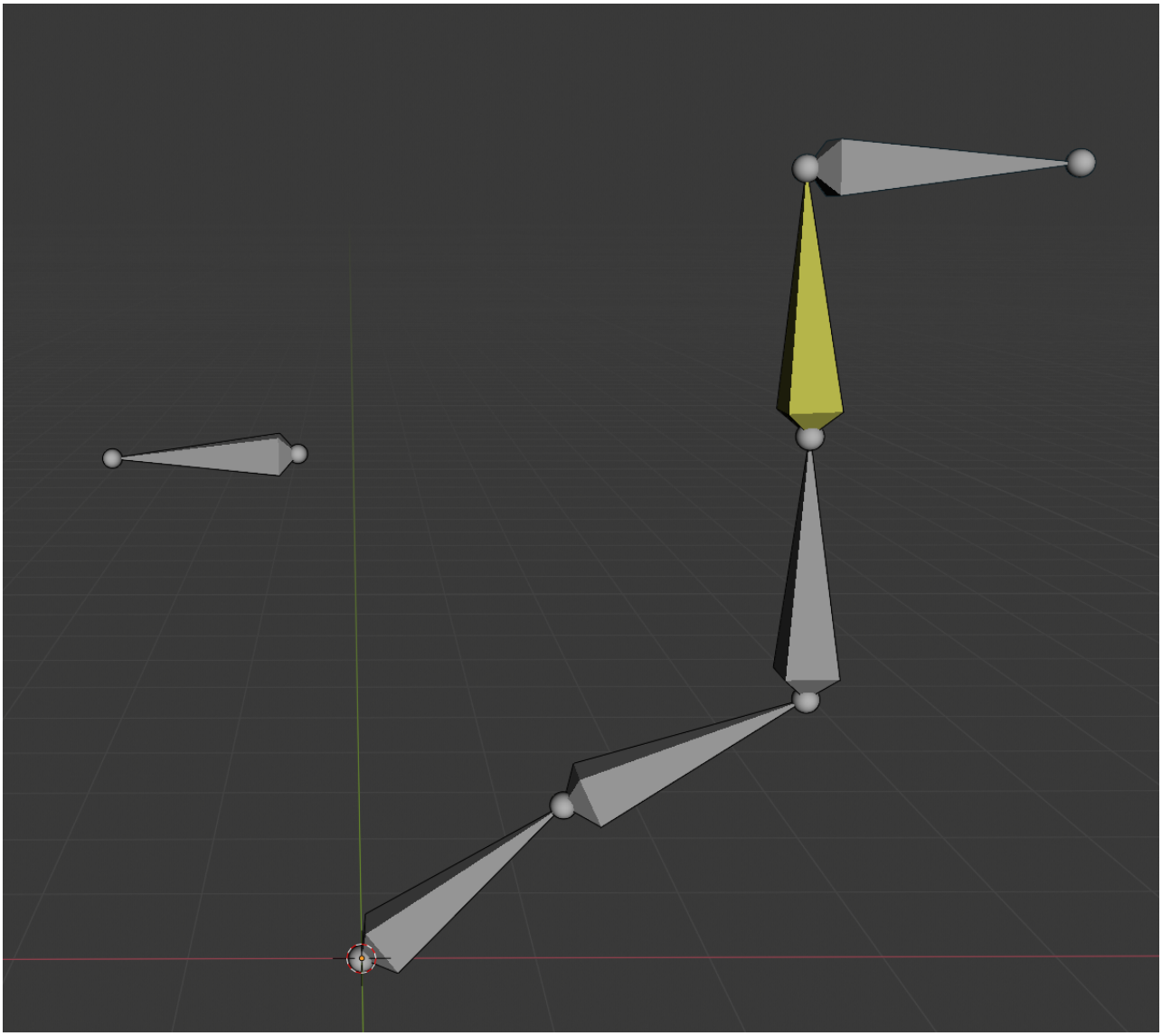

Figure 11: Pose Mod W/ Pole Example

6 A High Level Example In Blender

6.1 Meet SpiderBot

In Figure 12 the bones are see through while the mesh is not. This makes the skinning of the bones apparent. Then in Figure 13 the bones and the mesh are see through to see how they are exactly put together.

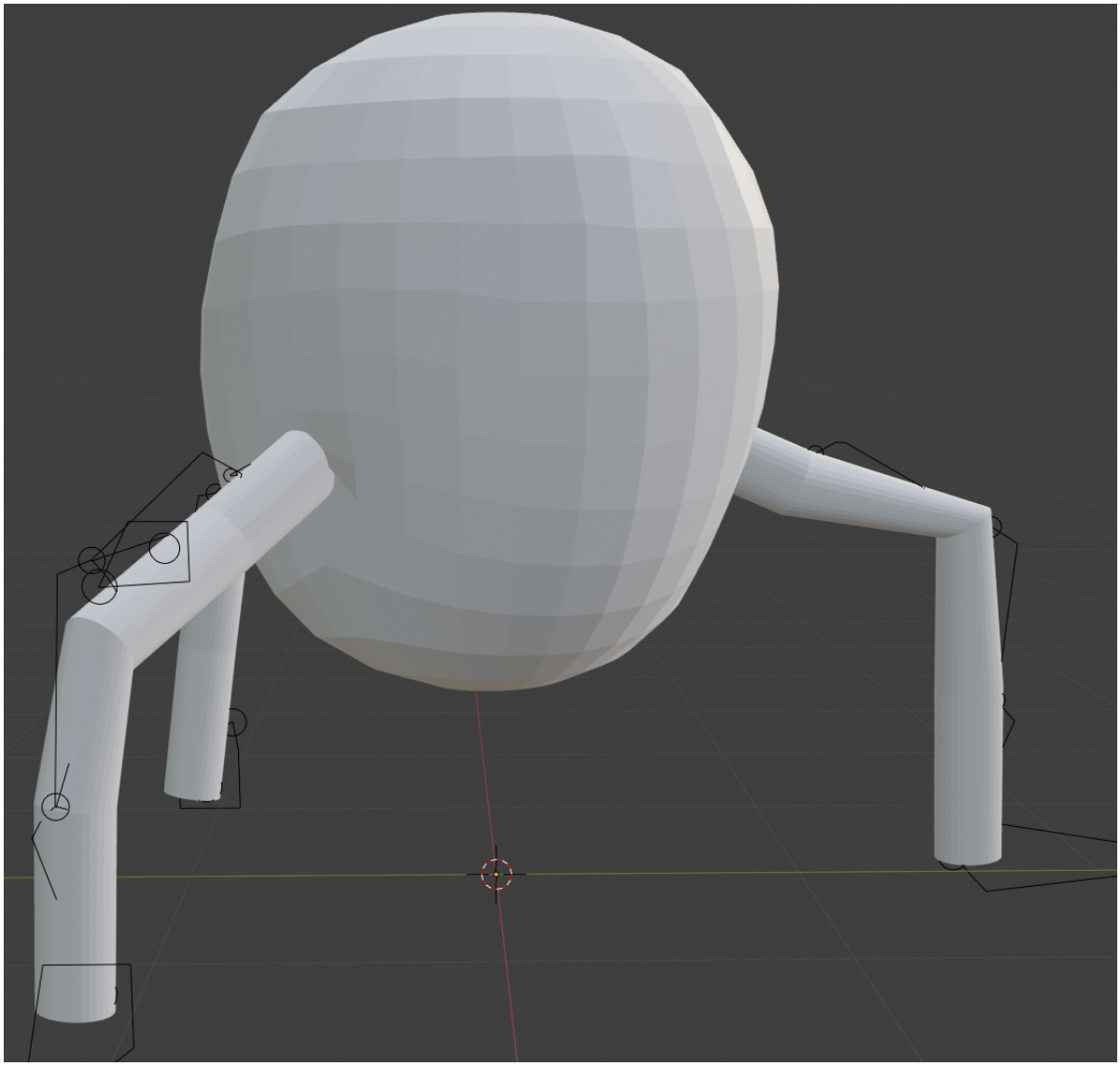

Figure 12: SpiderBot

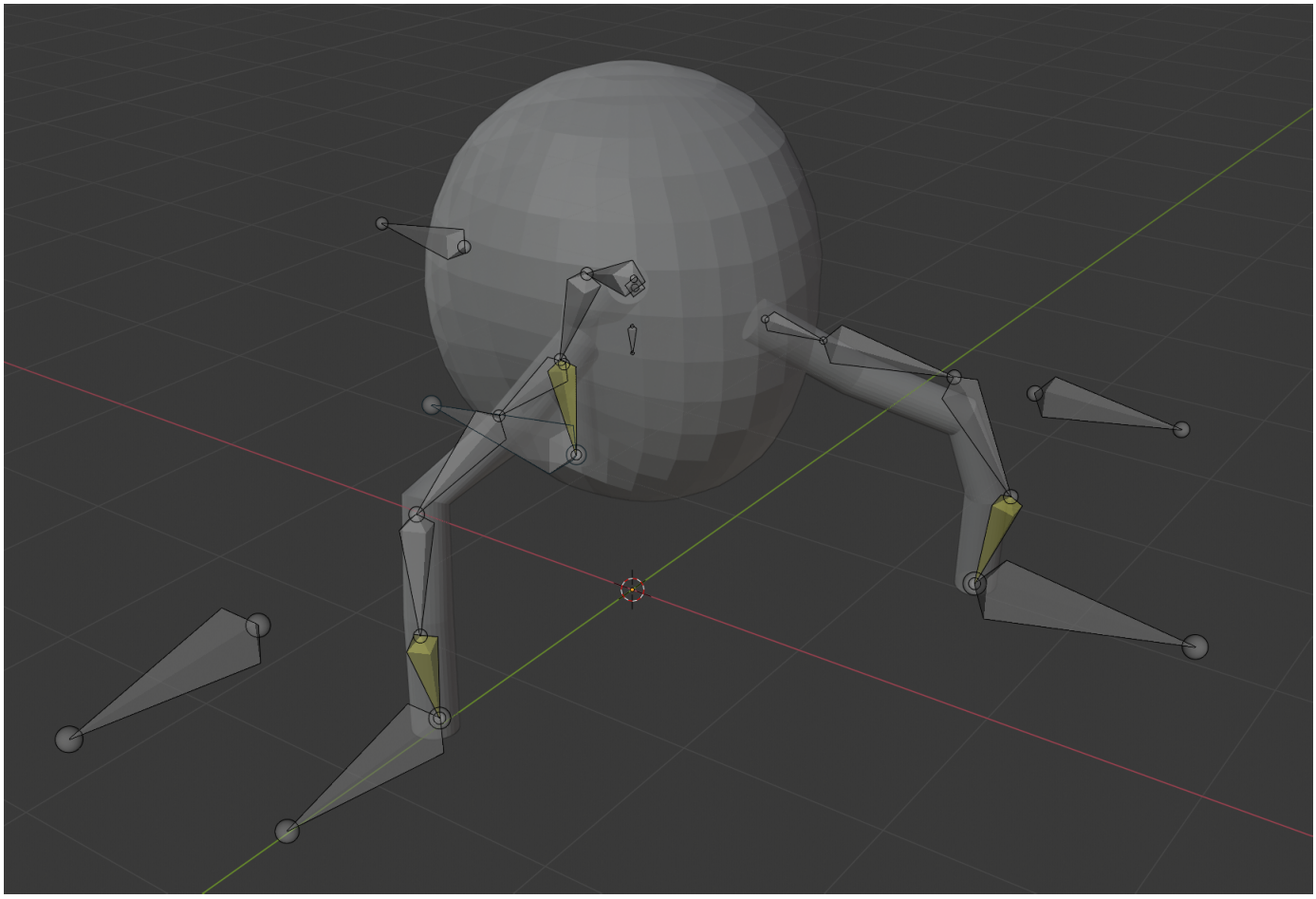

Figure 13: SpiderBot W Xray View

7 Exporting To Unity

7.1 Export as an FBX

- Select *File* → *Export* → *FBX*
- Then *Export FBX*, Blender will handle most materials and textures itself, but some might require importing separately into Unity.

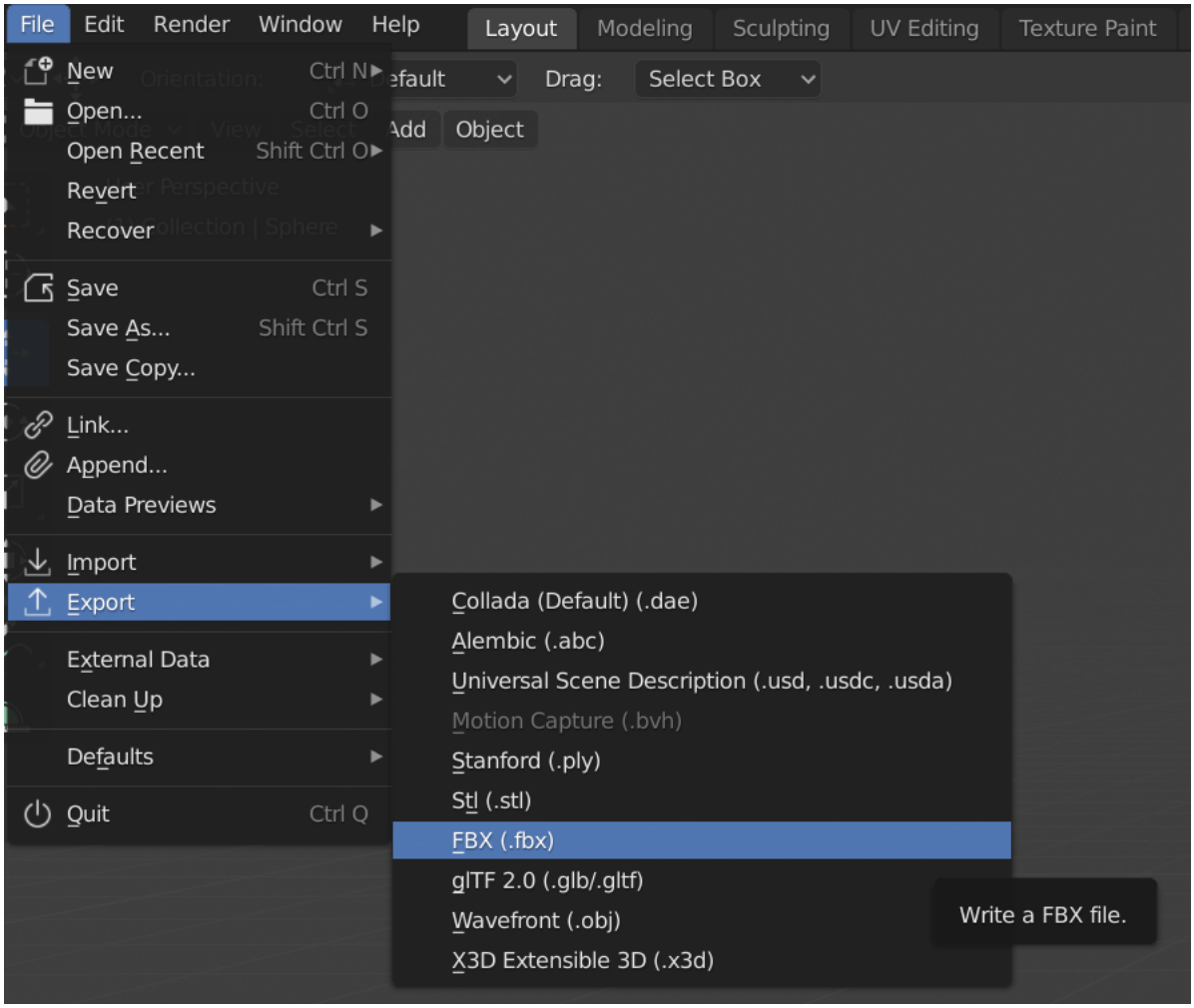

Figure 14: Export Menu Context

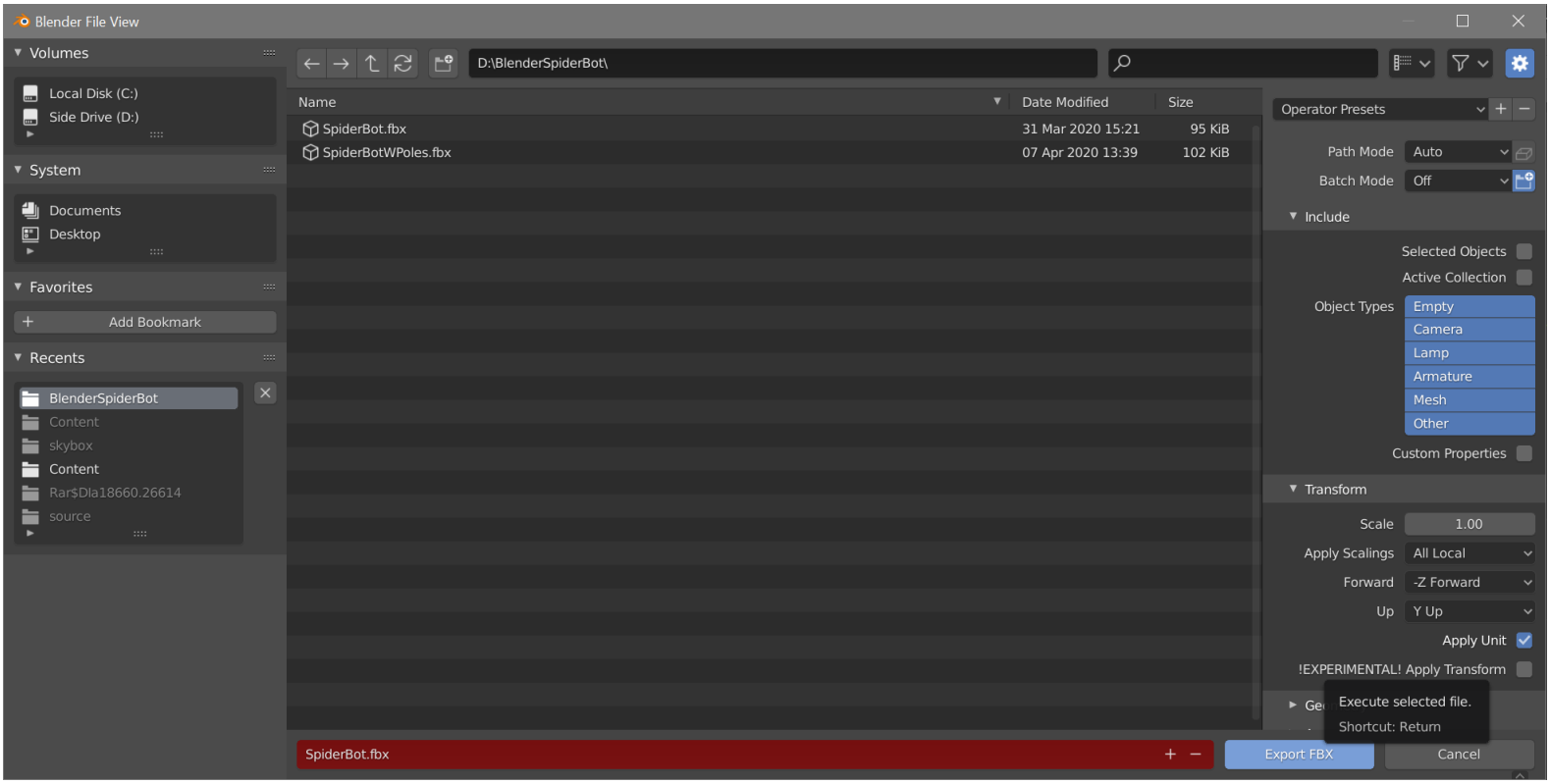

Figure 15: Export Final Menu Context

8 Using IK In Unity Example

8.1 Recommended Asset Packages

- Fast IK: Open Source and Free, but requires more work to setup bones in Unity.
- Final IK: Not Open Source and runs at about $90 USD, but setup is a breeze.
- Unity also has its own Biped IK System within Mecanim. Which is great for VR and foot placement IK.

  The example in this tutorial uses Final IK, and as such the Unity example will not be as in depth as the Blender section.

8.2 Setting Up CCD Bone Chains

8.2.1 CCD: Cyclic Coordinate Descent

Cyclic Coordinate Descent is a common method for creating Inverse Kinematic bone chains while optimizing for performance, it also works best in systems with non-humanoid bones. This system is already built into **Final IK** along with other IK systems.

8.2.2 Setup for a Single Leg

Setup for a single leg consists of going to the base bone of the leg and then adding a *CCD IK Component* to it. From there, the **IK Target** for leg is then added along with other tweakable properties, but most importantly each bone in the leg's bone chain is added in parent-child order. The reference Figure 16, shows the bones for the example SpiderBot's Backleft leg starting from the root bone at *BackLeftLeg.1 through BackLeftLeg.4_end*

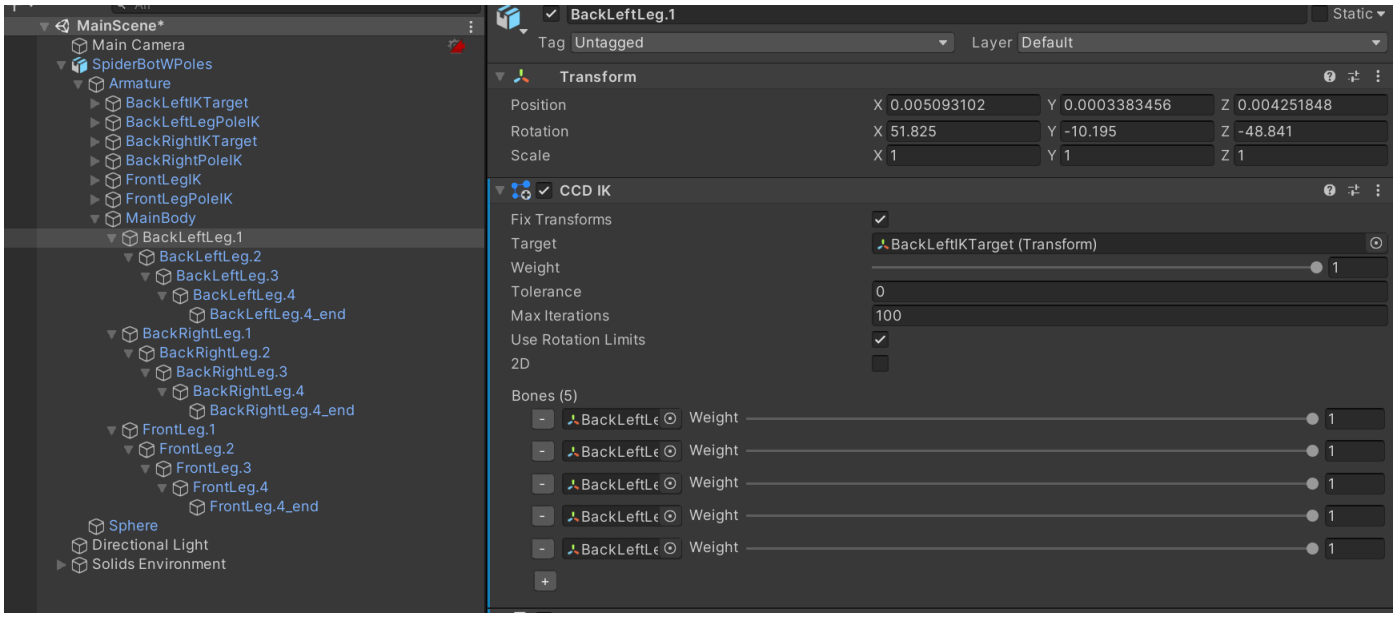

Figure 16: CCD IK Example

8.3 Repeat for Each Leg

After repeating the process for each leg then some of the real magic can be done.

9 Procedural Animation Using IK

9.1 The Main Idea

1. Save the resting position of the legs.
2. Move the main body using standard input.
3. Use the main body's velocity to calculate a new resting position for the legs.
4. Update the IK Target of the leg to be the new position.
5. Use interpolation to smoothly transition the leg between the two points.
6. Repeat.

9.2 Some Auxiliary Additions

- Height of the main body based on leg positions.
- Using Raycasts to determine dynamic leg placement such as foot position based on surface normals.
- Special collision layers for the legs.
- Plus way more than this tutorial could ever cover.

10 References

1. Initial Inspiration and Methodology
2. Another tutorial that conveniently came out in early April 2020. That is a very interesting take on procedural animation, the author's currently WIP game is entirely procedurally animated.