Rigging & Armatures - Full explanation/A deeper dive

In this book I'll attempt to fully explain armatures, bones & transforms

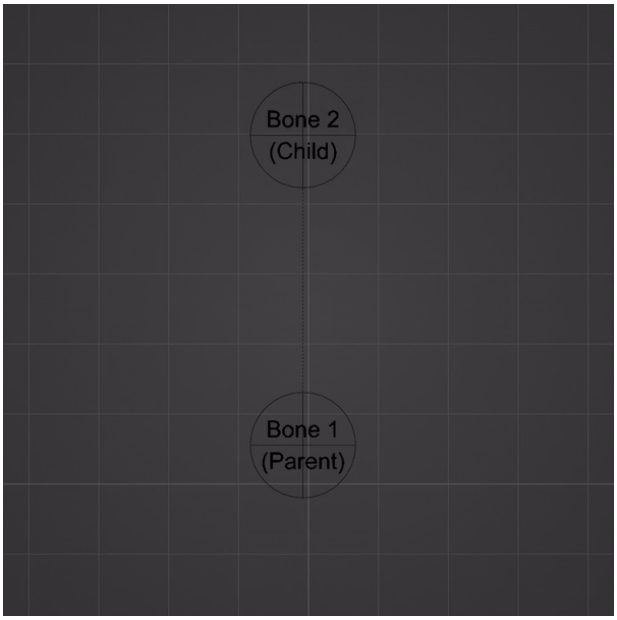
- Armatures, Barebones
  - Armatures / Rigs, what are they & how do they work?
  - Orientation (Rotation) Euler angles
- Deformation, Skinning / Weighting (and Blendshapes)
  - Armature deformation, what is it and how does it work? (WIP)

Armatures, Barebones

This chapter explains how the armature works and behaves on its own.

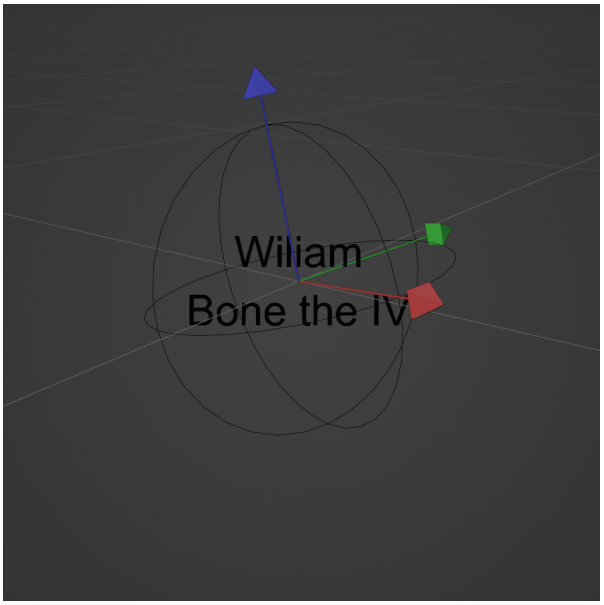
Armatures / Rigs, what are they & how do they work?

What is an armature made of?
An Armature / Rig is made up of bones.
Bones can be parented to other bones, which will make them inherit movement from their parent bone.



The order in which all of the different bones are parented to each other, is called the hierarchy.

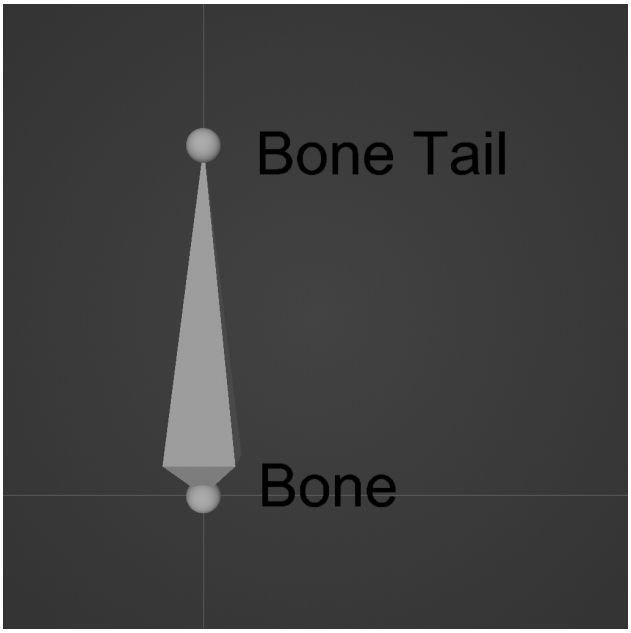
What are bones?
Technically a bone is simply a point (pivot) in space.
These pivots have a name an orientation (rotation) & as previously mentioned a location.



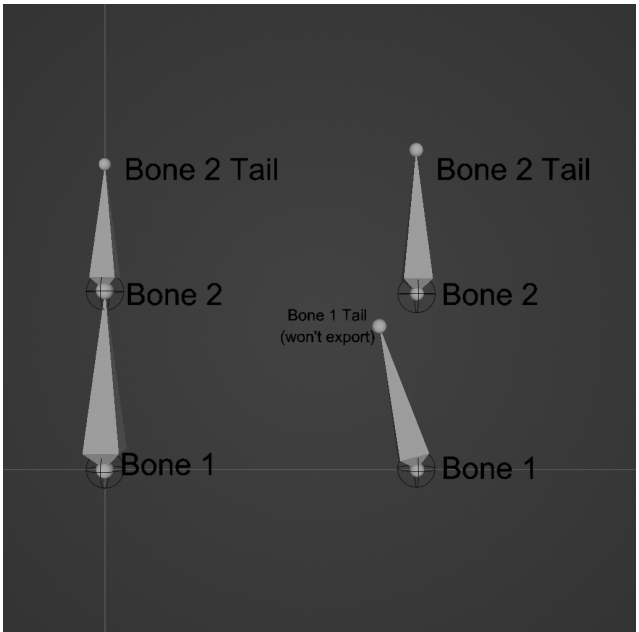
What about scale?
scale isn't actually saved per bone in the armature, all bones have a default scale of 1, and have the ability to scale as soon as they're being posed. if you scale two bones in edit mode, you're just moving the points further apart.

this is also where some confusion begins due to the way blender handles / visualizes bones

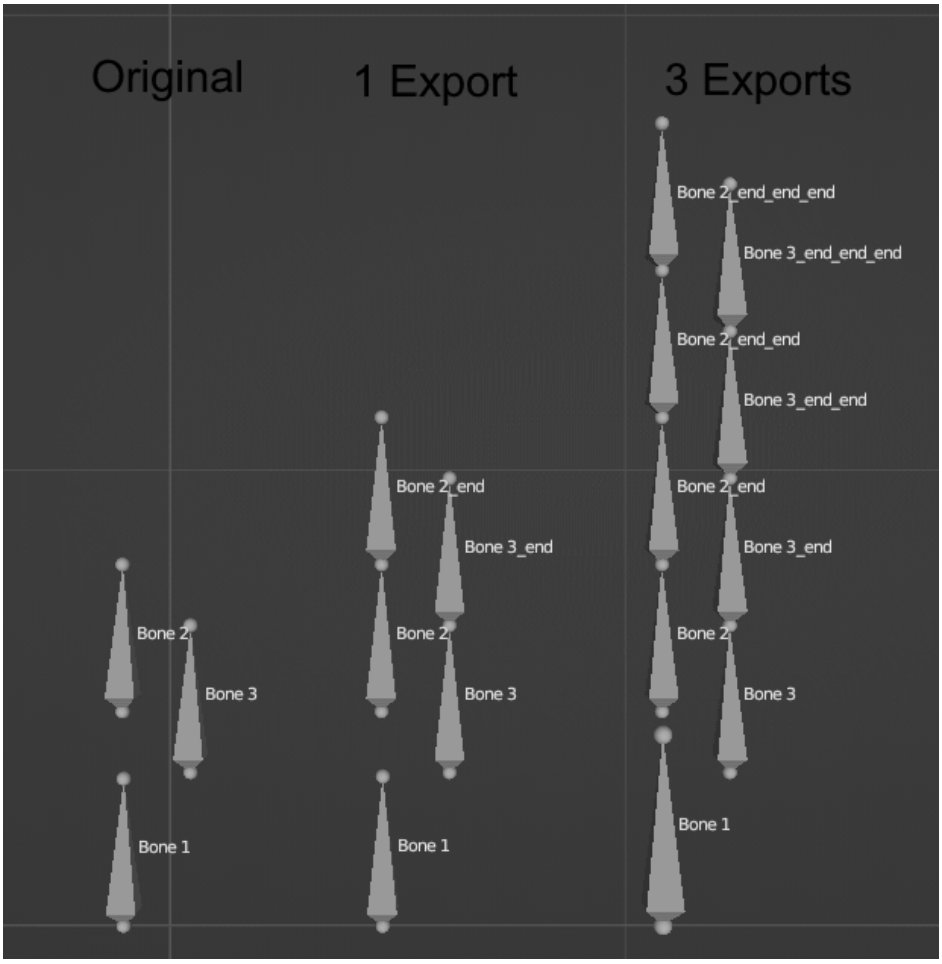
But I thought bones were sticks?
well, no, not really. Unity doesn't treat them as sticks & most 3D file formats can't save bones as sticks.
Blender has bone tails, meaning that you can't have a bone that isn't connected to anything / doesn't have a child. Blender will create another bone at the end to make it a stick (this bone is called the tail, an end bone or a leaf bone).



When you extrude a new bone from the tail of another bone, it becomes a root bone with it's own tail
so basically as soon as a bone has a child of it's own, it doesn't have a tail bone anymore. (even if it's displayed that way)
This also allows you to change the orientation of a parent bone to not directly point at its first child.



End / Leaf bones / Bone tails
Blender will, by default, export leaf bones.
this means that it will create an extra bone where the bone tail is upon export.
I highly advise you don't export with leaf bones, as it can add useless bones that impact the performance of your avatar.
Additionally, if you import and export a handful of times with end bones on, you'll get a giant spaghetti chain of useless bones.

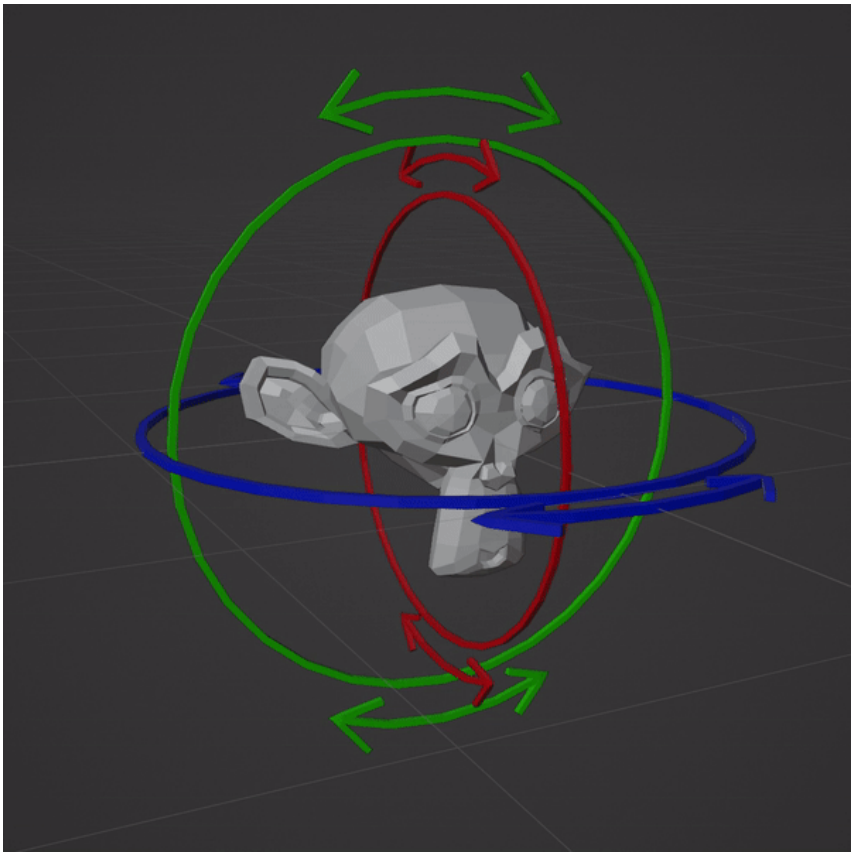


Using Leaf / End Bones for Dynamics
Sometimes you'll actually need end bones, Primarily for dynamics. Unity needs to know where a bone ends to put a collider there, or to know how much the bone needs to move (for ex. a long feather would wiggle more than a short one)
for this I'd recommend adding your own end bones wherever they're actually needed

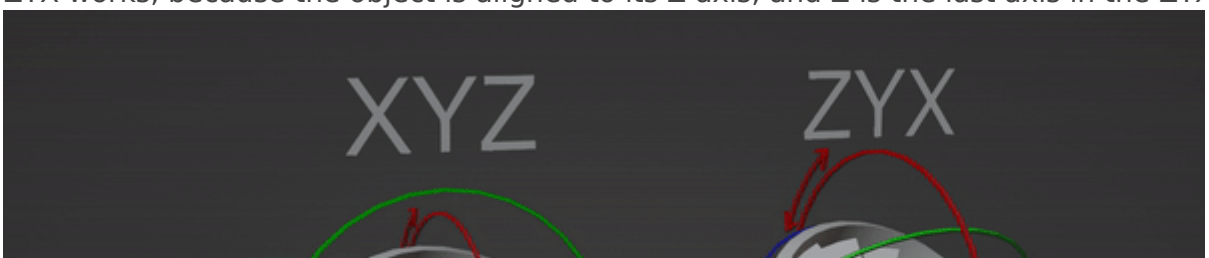
Orientation (Rotation) Euler angles

As mentioned in the previous article, all bones have an orientation (rotation). This orientation mainly affects how constraints behave, but can also have an effect on IK solvers.

How does rotation even work in 3D? (Euler gimbal)
The easiest/most comprehensible way of storing rotation is by having 3 axes, that affect each other in an order.
Unity lets you rotate objects this way, however behind the curtains all rotations are stored as quaternions (another more confusing way of storing rotation).
This GIF shows an order of X Y Z (Red-Green-Blue)

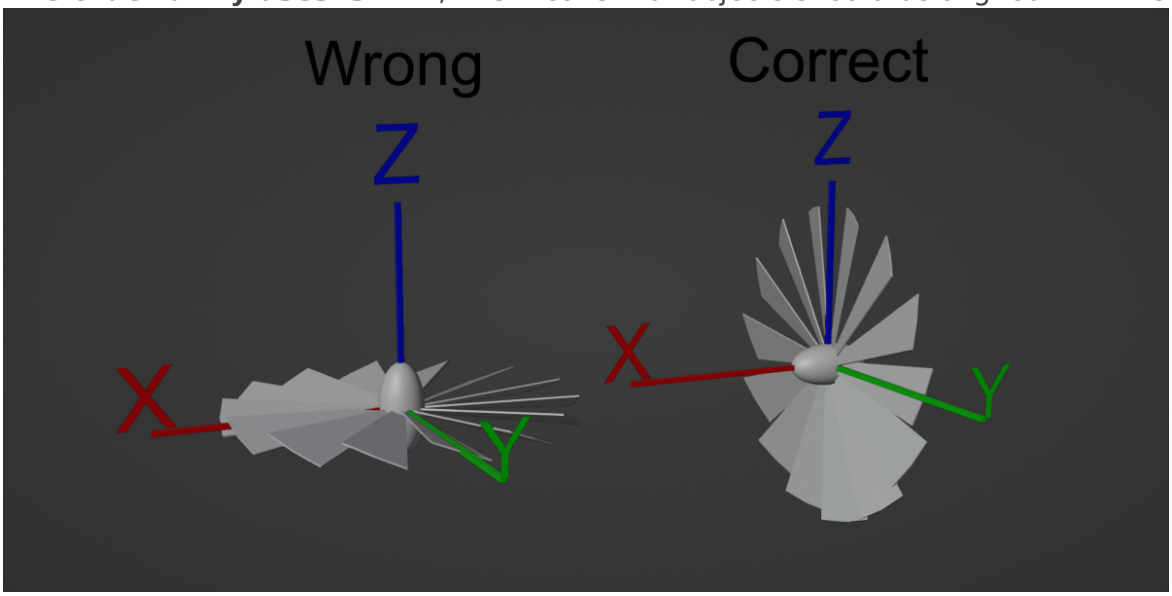


This order heavily affects how objects rotate, this is pretty important to know when you want an object to rotate around one axis.
In the GIF below you can see, that using the order XYZ causes you to have to rotate all 3 axes for the object to appear to rotate around one.
ZYX works, because the object is aligned to its Z axis, and Z is the last axis in the ZYX order

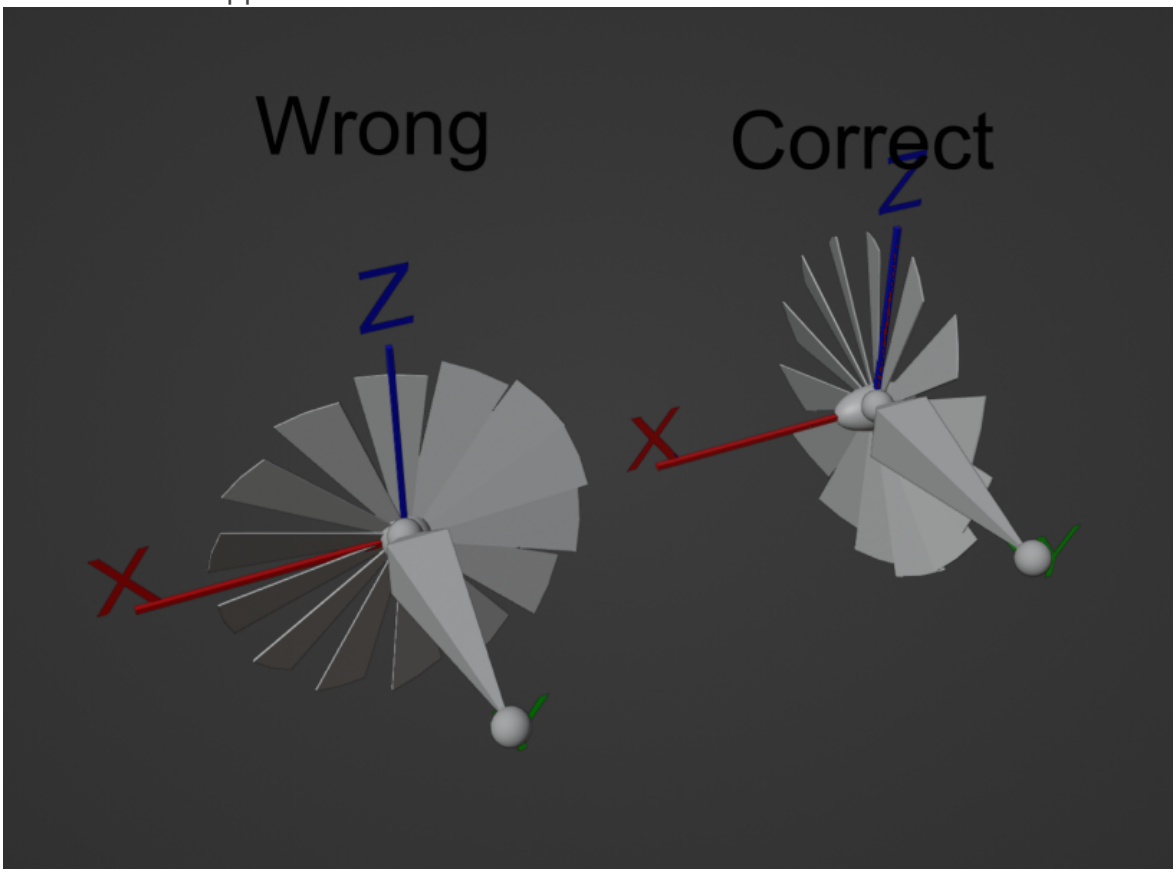


However we cannot change the Euler order in unity, so we need to align the objects at rest (at rest meaning that the rotation is 0,0,0)

The order unity uses is XZY, this means that objects should be aligned with the X axis at rest.



All of this also applies to bones



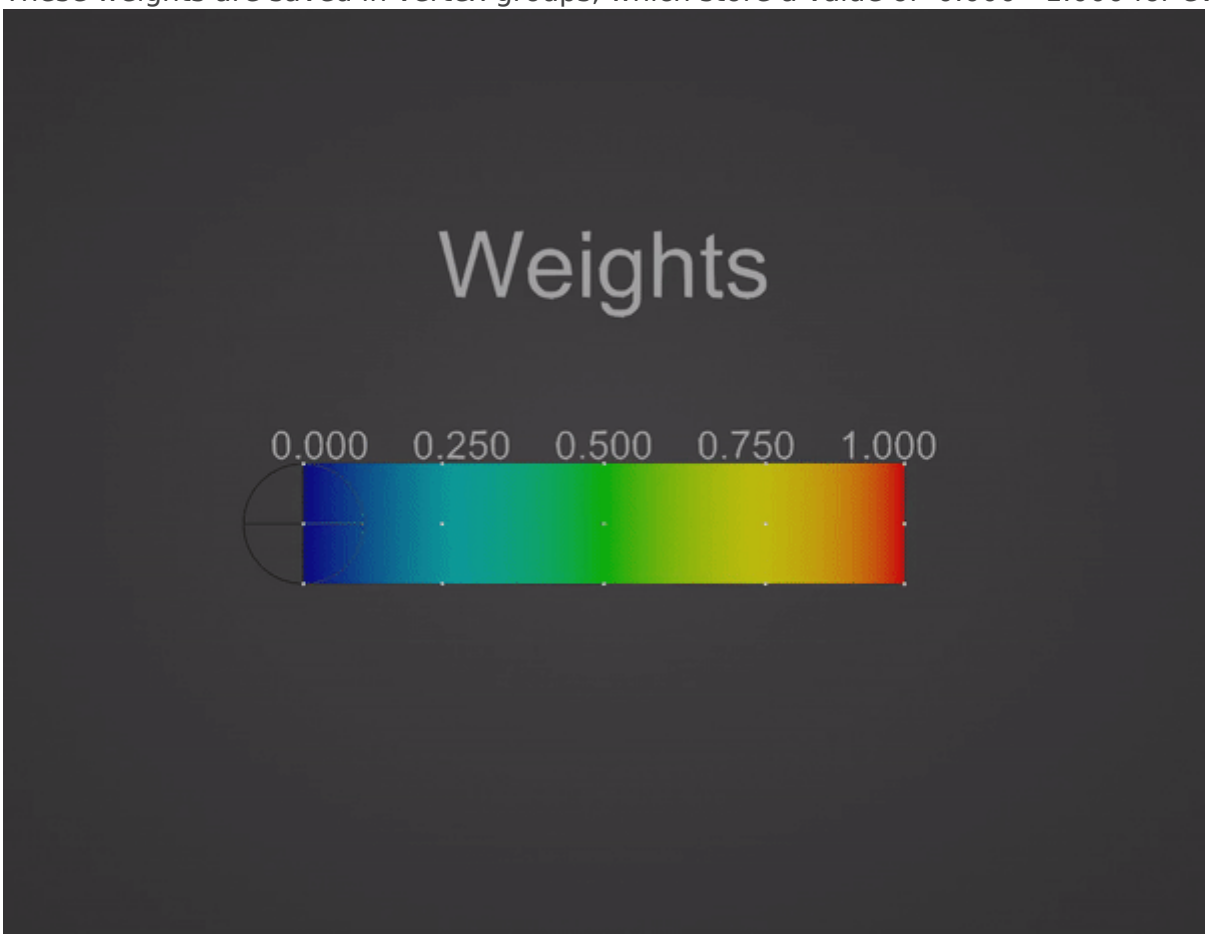
Deformation, Skinning / Weighting (and Blendshapes)

This chapter covers how meshes are deformed by armatures.
It also covers Blendshapes and how they work, since it's also a type of deformation.

Armature deformation, what is it and how does it work? (WIP)

Armature deformation is the reason we built an armature. It allows us to weight/skin vertices to bones, which makes those vertices move along with those bones. This works quite well to simulate the movement of limbs on creatures and robots, while not being very performance intensive.

Weights/Skinning?
As previously stated, weights decide how much a vertex follows a given bone.
These weights are saved in vertex groups, which store a value of 0.000 - 1.000 for every vertex.



Vertex weight groups are assigned to their respective bone via their name, meaning that you shouldn't rename vertex groups for weights unless you also change the bone name to stay identical. This can also be used to change the bone a vertex group is assigned to.