Basic user guide for the Asset Editor

The Asset Editor builds on top of PFM and will gradually remove or replace the original parts based on need. At this stage the program is in a very early stage and not everything can be expected to work. Users are free to try out whatever they want, but should be aware that a lot of the original FPM features are broken or disabled. I have done my best to disable any saving, but to be safe users are recommended to not save anything inside the AssEditor as there is a small chance that the save will actually change data in the pack files in a way that is not intended.

The current build is a debug build, which is a lot slower, but makes it easier to find errors. There are also still some debug features enabled, such as the paladin showing up when the program starts. This is to make it easy to verify that the basics are working.

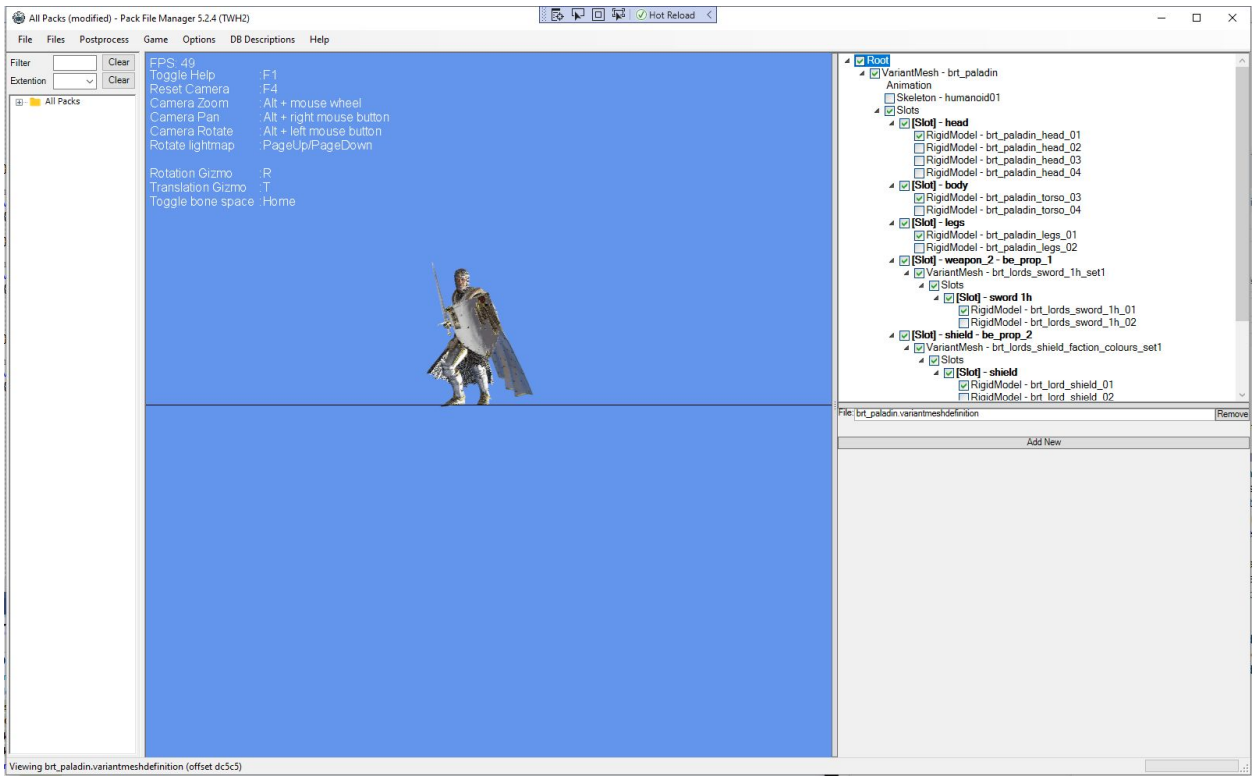
There is also a fair amount of rendering bugs that will be removed later.

If the program crashes, please send me a log, the logs can be found here:
C:\Users\{userName}\FPM\Logs

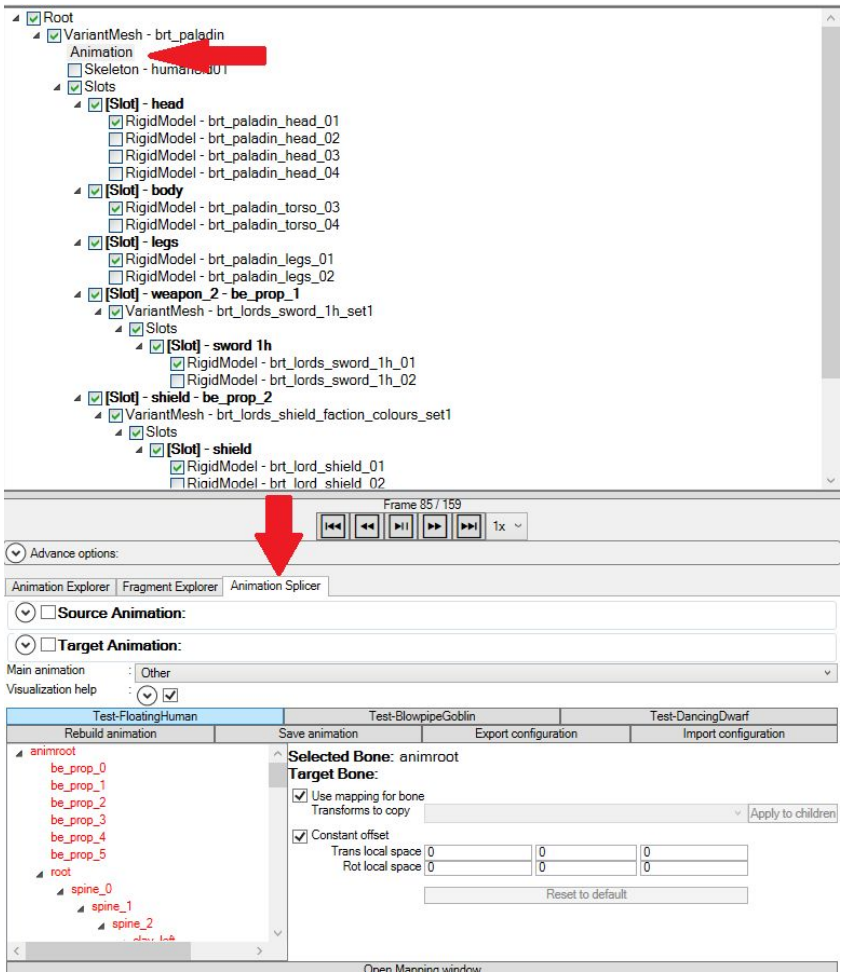
First start - Basic verification

You will be prompted to enter the game directory, only the directory for Warhammer 2 is needed, the others can be aborted. My Path is **"C:\\Program Files (x86)\\Steam\\steamapps\\common\\Total War WARHAMMER II"**
You will be prompted to update schema files, this can be aborted.

When the program starts you should see this, with the paladin having the idle animation applied.



Then, select the animation tab in the right treeView followed by the Animation Splicer tool:



After this, press the “Test-floatingHuman” button and then Rebuild Animation.
After doing this the new animation should have been applied and it should look like this:

After this, press the Save animation button and save the new animation somewhere. If all of this works, then you can assume that most stuff will work as expected. All of this will be removed later, it's just here to make it easier for me to establish a baseline when trying to help people resolve issues.

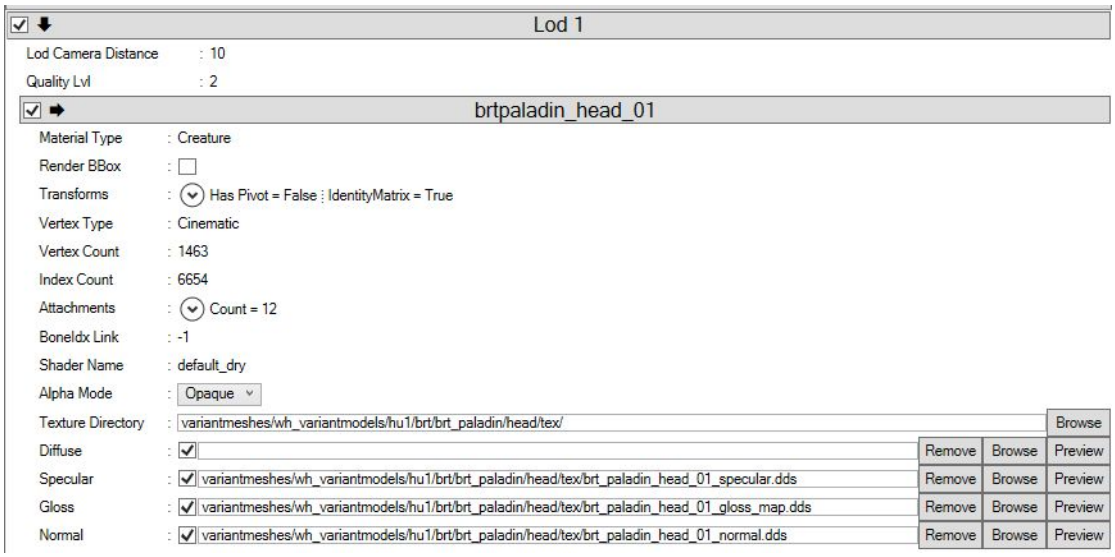
Tool: Variant mesh Editor

Todo - Almost done with the programming

Rigid model explorer

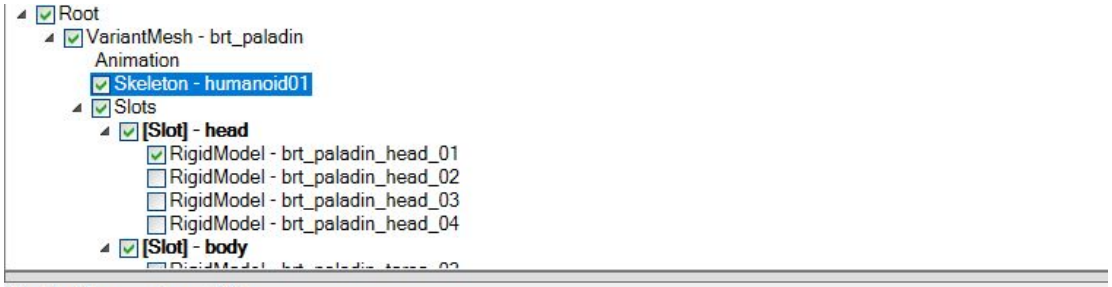
View Textures

Press the preview button for a texture



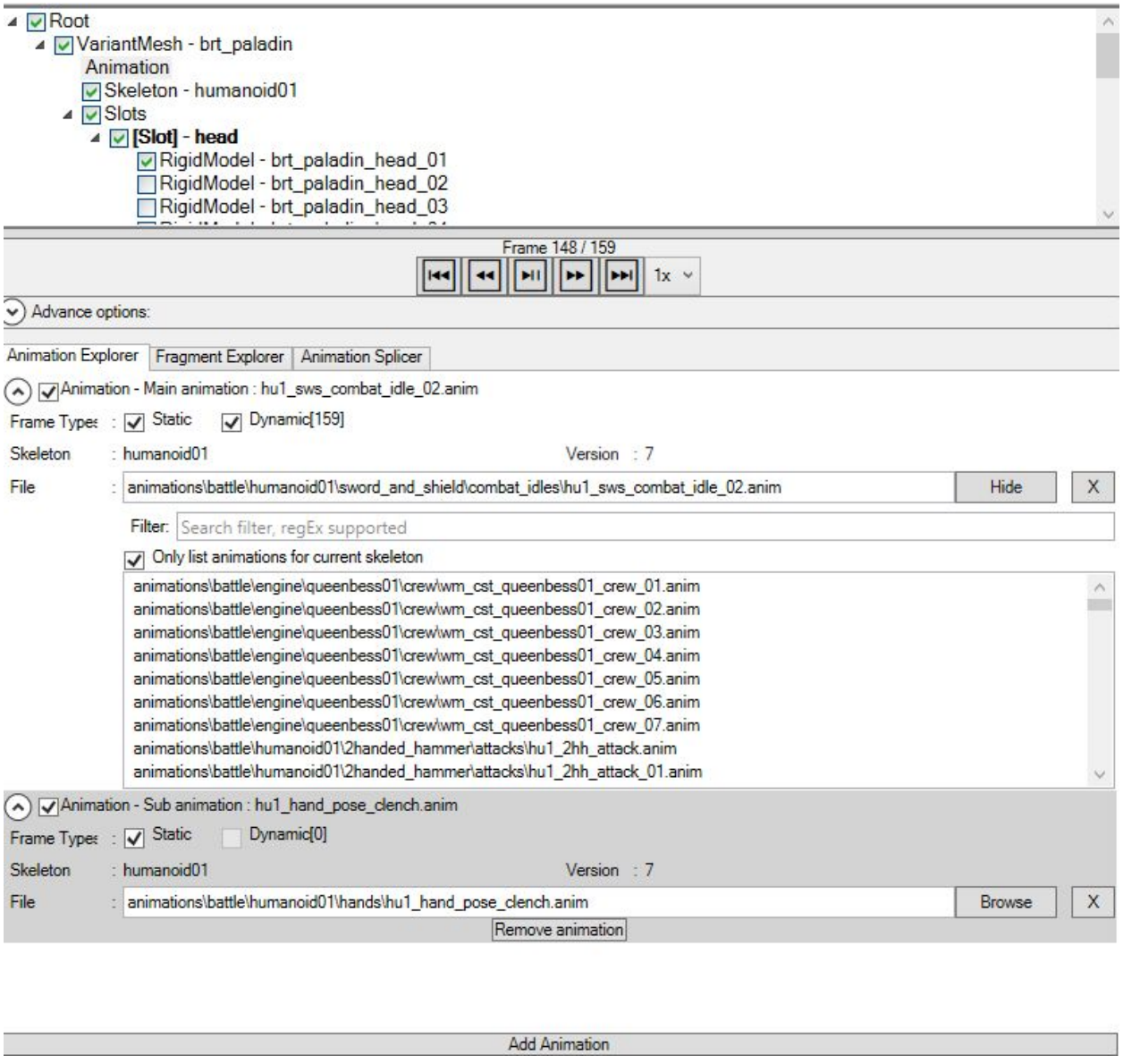
Tool: Skeleton Explorer

Gives basic information about the loaded skeleton. Selecting a node will select the bone in the scene.



Tool: Animation Explorer

It's possible to view all the animation in the game using the Animation Explorer located under the Animation tool in the tree view. Using this you can stack multiple animations on top off each other, the main use for this is to apply the extra animations which the meta files apply.

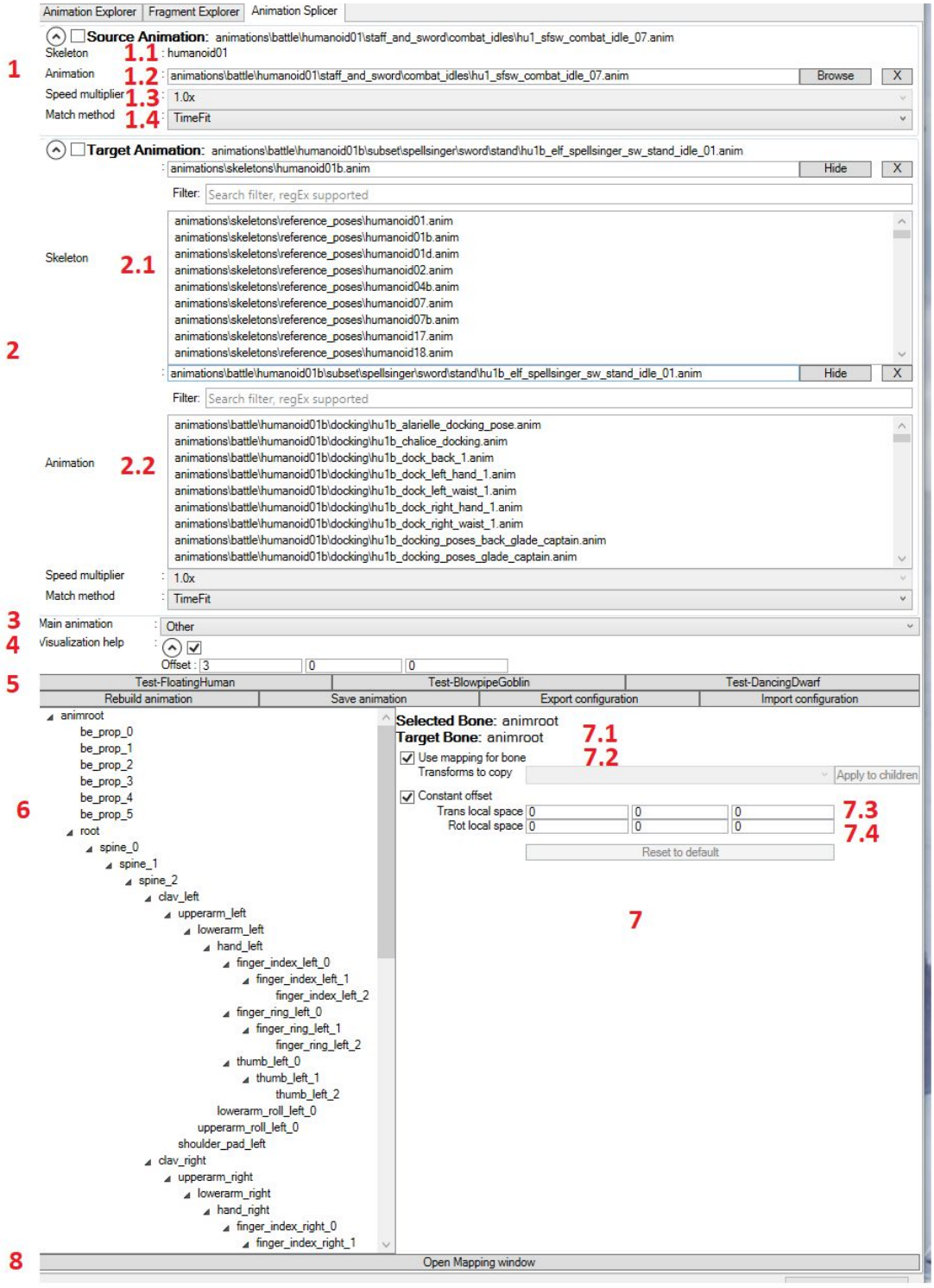


CheckBox next to the animation name: Use this animation.
Frame types: Number frames in the animation. Static indicates if the animation has static keyframes that are constant for each frame. Toggle then on and off to see how they effect the animation.
Browse: Opens the filter explorer
X button: Clears the current animation file

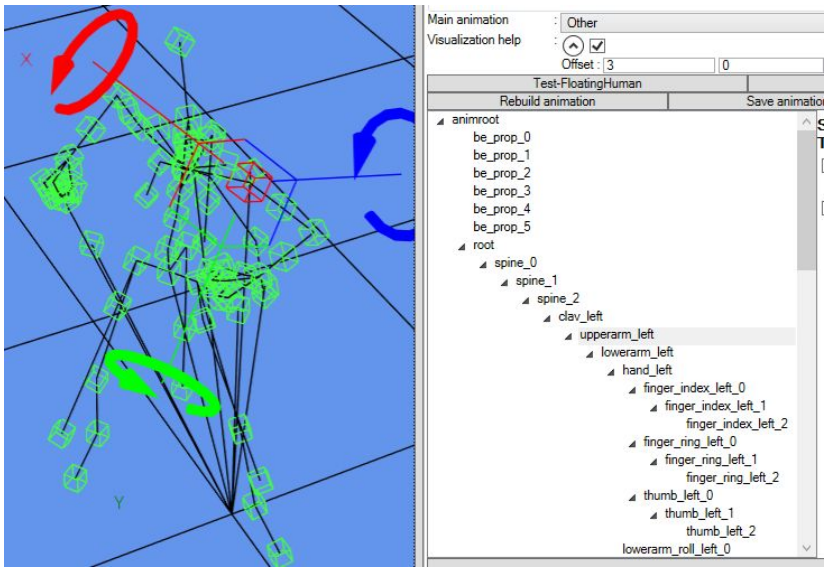
If an animation from a different skeleton is added, there is a high chance of the program crashing.

Tool: Animation Splicer

The animation splicer is the tool for combining animation from different skeletons.



1. The source animation editor. This is where you configure the settings for the skeleton you want to create animations for.
    1.1. The name of the current skeleton, is based on the loaded mesh, can not be edited.
    1.2. The current animation can be changed. Press browse to explorer and X to remove.
    1.3. Speed multiplier, currently not implemented
    1.4. How this animation will match up in time related to the other animation, currently not implemented
2. The Target animation editor. This is where you configure the settings for the skeleton you want to take animations from.
    2.1. The skeleton to use
    2.2. Animation for the selected skeleton
3. Which animation should be used as the longest animation, currently not implemented
4. Show or hide the skeleton you are taking data from. The offset is where in the world it will be placed.
5. The top row of buttons are for testing and should not be pressed
    5.1. Rebuild animation recreates the combined animation. Must manually be pressed when changes are applied. Will be automatic at some point
    5.2. Save the animation to file, this animation can then be imported in RPFM
    5.3. Export the current configuration
    5.4. Import a stored configuration
6. A list of all the bones. Red bones have no mapping. Selecting a bone will display some info about it on the right side. If the animation is not playing, the selected bone can be manipulated using a Gizmo. R to rotate and T to translate. This feature is not completed and has a few bugs which will be ironed out soon.



7. Selected bone information
    7.1. Shows the original and the target bone name
    7.2. Checkbox to temporarily disable this bone mapping

7.3. Translation offset for this bone. The translation is in local space for the current bone, so X,Y and Z will be relative to the rotation of the parent.

7.4. Rotation offset for this bone. The translation is in local space for the current bone, so X,Y and Z will be relative to the rotation of the parent.

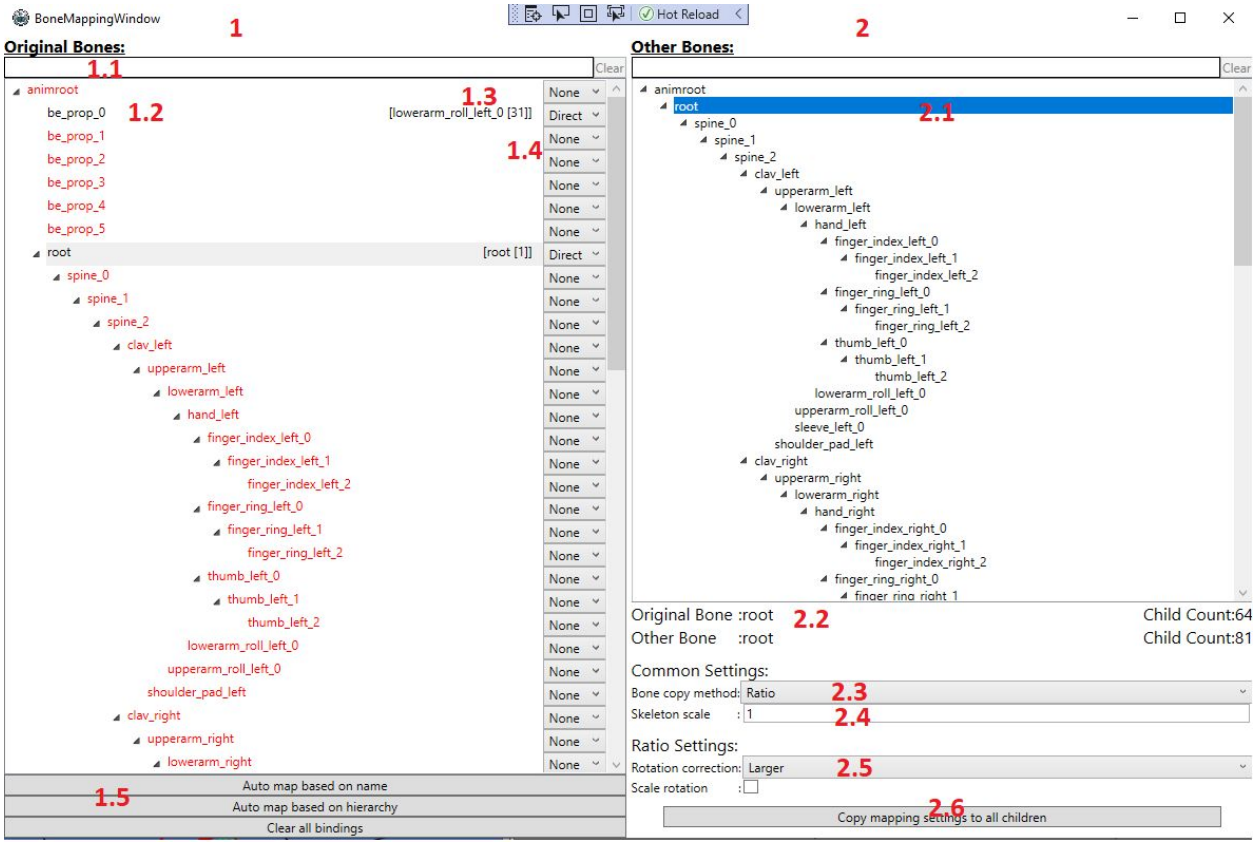
8. Open the mapping window

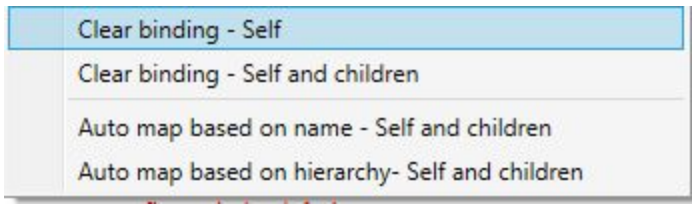
Mapping window

The mapping window is where you configure how animations should be transferred from one skeleton to the other. There are two auto map modes.

Name: The program tries to find bones with the same name in the other skeleton and creates a mapping

Hierarchy: The program creates a mapping based on how the skeleton is structured.





1. The left side contains the bones in the original skeleton. The bones also have a context menu. Red bones have no mapping
    1.1. Filter which bones to show based on name
    1.2. A tree view of all the bones
    1.3. A description of the current mapping
    1.4. The mapping type configured for this bone (currently only direct exists, more will be added later)
    1.5. Buttons for mapping
        1.5.1. Auto map selected bone and children based on name
        1.5.2. Auto map selected bone and children based on hierarchy. It will start at the selected node in the other list.
        1.5.3. Clear all mappings
2. The left side contains the bones in the other skeleton. At the bottom settings for the mapping can be configured.
    2.1. A list of all the bones. Select one to create a mapping for the selected bone in the left column.
    2.2. Shows the name of the mapped to and from bones as well as number of children for this bone
    2.3. Bone copy method. Experiment with them to see what fits best
        2.3.1. Ratio - The program tries to be clever by calculating the difference in size between the two skeletons.
        2.3.2. Relative - The program tries to be bit clever by only moving animation data by subtracting the skeleton pose
        2.3.3. Absolute - The program just moved everything without even trying to be clever
    2.4. Skeleton scale
    2.5. Ratio settings
    2.6. Copy setting to all children.

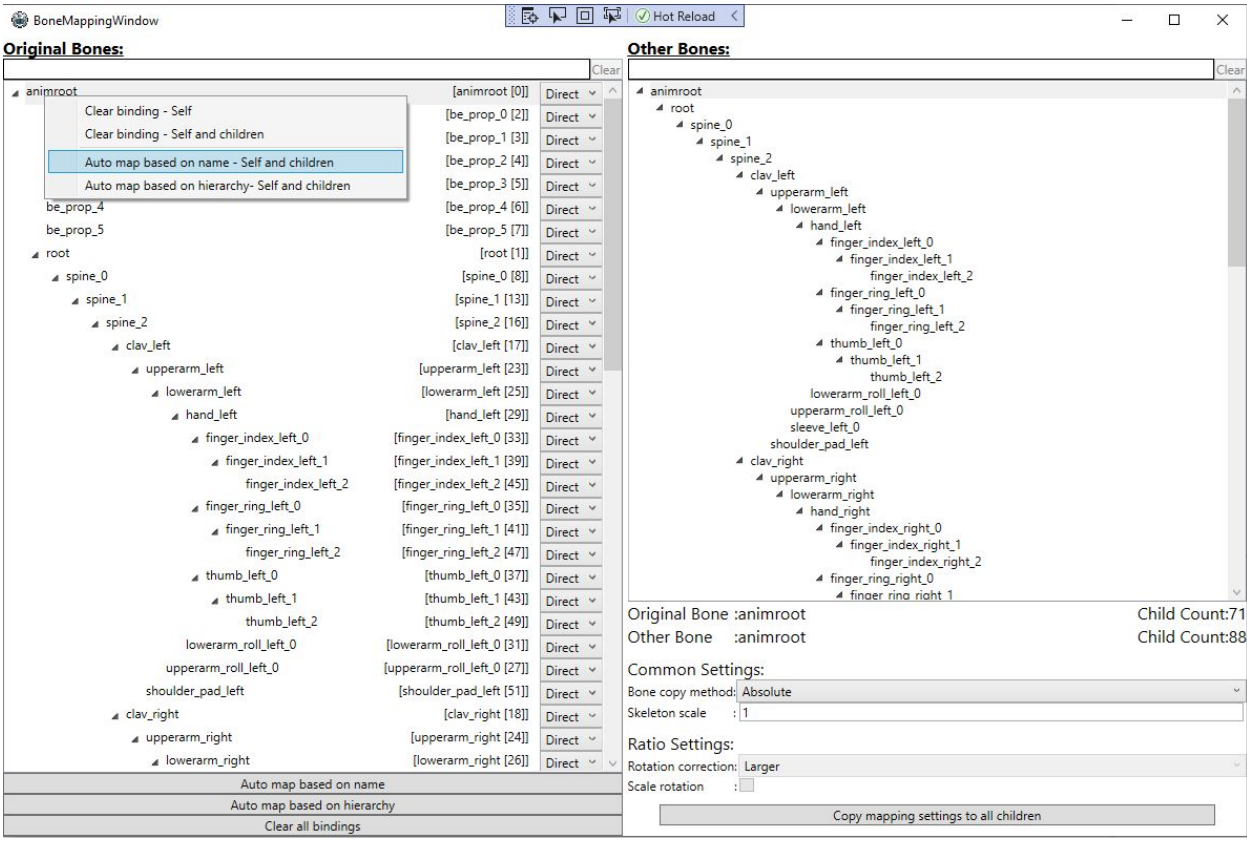
General info and tricks

1. If there is no mapping, the animation in the original skeleton is used.
2. The program works pretty well for skeletons of similar sizes, but for large differences you will notice that contact points drift.
3. Mapping between monsters can look very strange.
4. Remember that all transforms are relative to the parent! Because of this it is often necessary to manually adjust something.

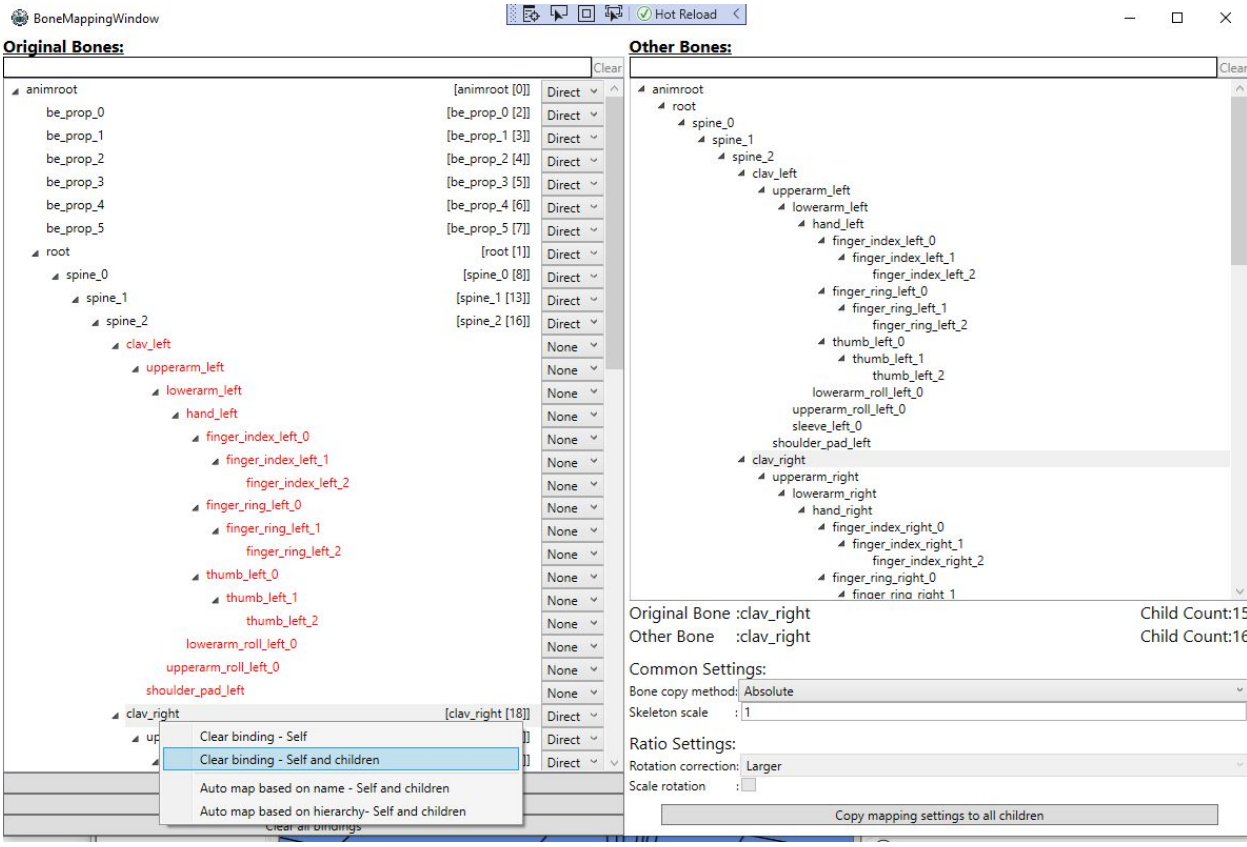
Example - Floating paladin doing spellcasting

1. Start the program from scratch and press the “Test-FloatingHuman” button.

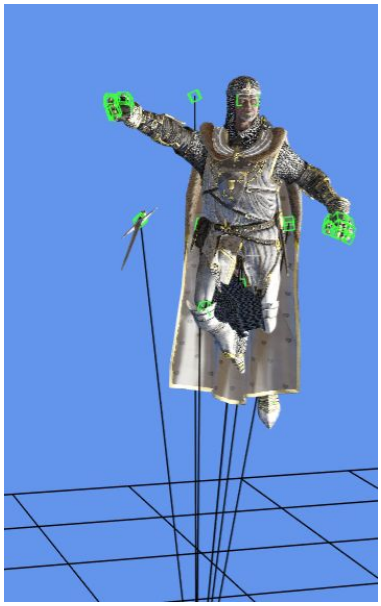
2. Select “animations\battle\humanoid01\staff_and_sword\celebrate\hu1_sfsw_celebrate_01.anim ” as the source animation
3. Open the bone mapping
    a. Select root node and pick auto map based on children.
    b. Pick bone copy method absolute
    c. Press Copy mapping settings to all children



d. Right click clav_left and select clear binding - Self and children. Do the same for clav_right



4. Press rebuild animation. There should now be a mix of the two animations playing

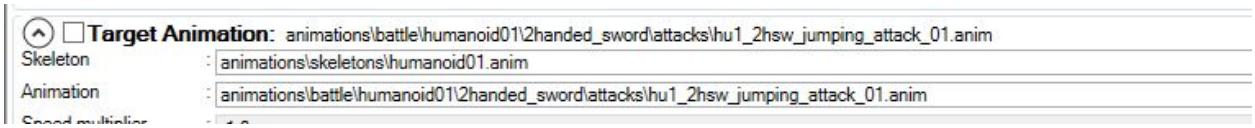


Example - Goblin with pistol animation

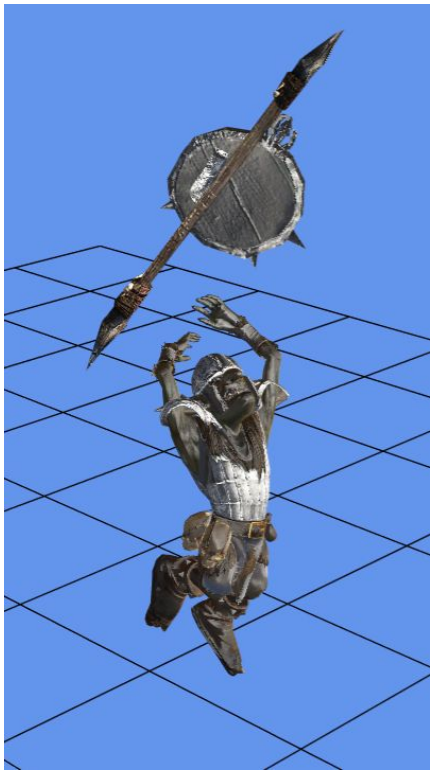
Example - Dwarf with gatling gun

Example - Gobblin with 2 handed human animation (Updated)

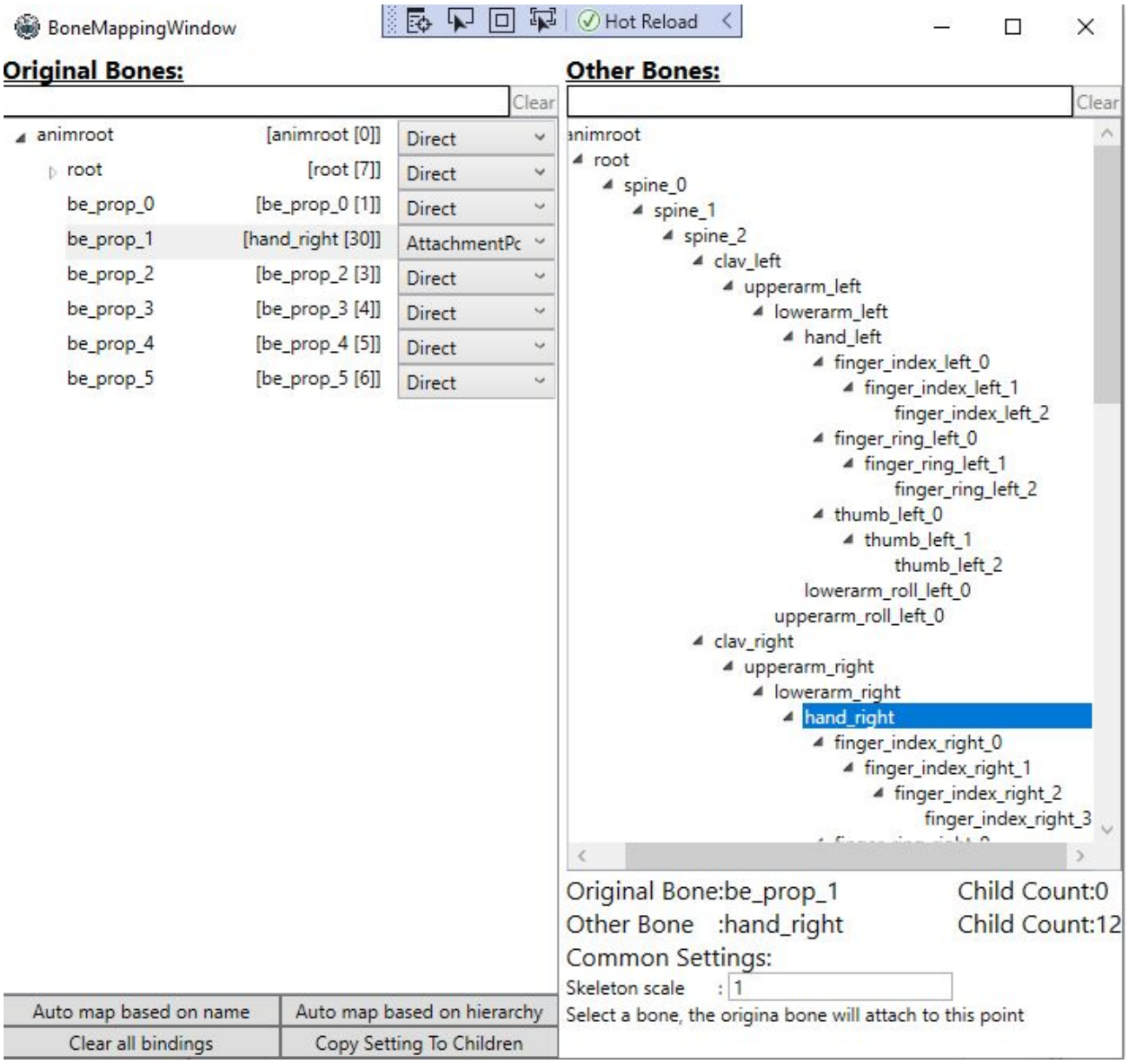
1. Load the **grn_goblin_spearmen** file
2. Load **animations\battle\humanoid01\2handed_sword\attacks\hu1_2hsw_jumping_attack_01.anim** for **animations\skeletons\humanoid01.anim**



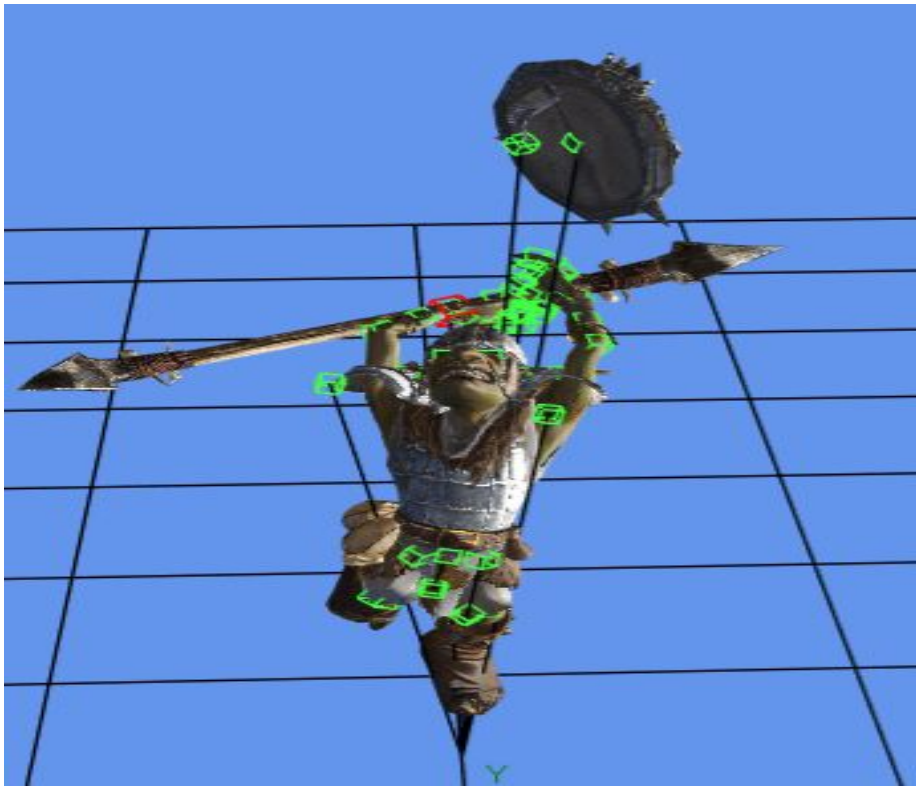
3. The automatic bone mapping does a pretty good job of mapping this skeleton. Press “Rebuild animation” to see how it looks. The animation should look good, but the attachment points are not where we want them. Frame 9 looks like this:



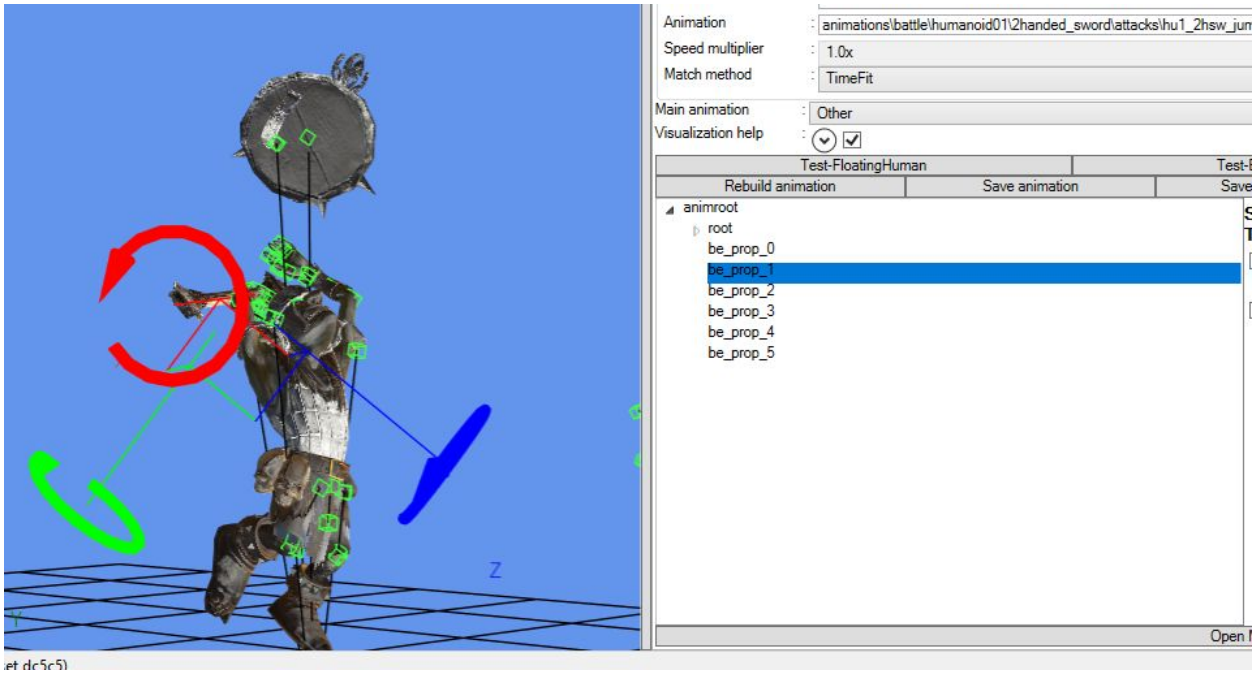
4. To correct this open the “Mapping window” and change **be_prop_01** to attachment and then select **Hand_Right**



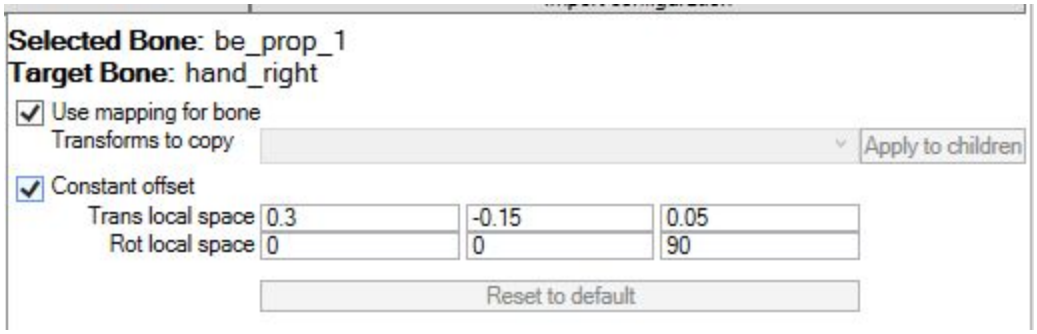
5. Rebuilding the animation should now have the bone attached to the hand, but at the wrong orientation.



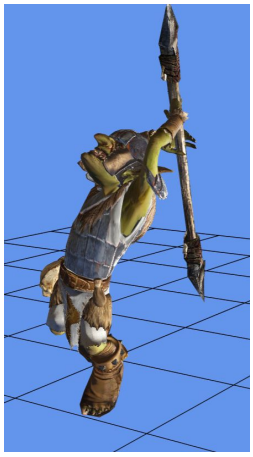
6. Select the bone in the bone list in the splicer tool and a gizmo should appear



7. Rotate the z axis 90 degrees and set the offset



8. Rebuild the animation and it should not look like this (shield hidden manually)



Future work:

More mapping methods that work better when things are different. IK solver and using absolute transformation are my two best ideas so far. If you have any ideas let me know. I am also working on a fix to correctly position attachment points