Index

Note: The content of this manual may change and therefore may not be applicable to every version of IE AutoSpriter. It also serves as a preliminary manual. I'm also not a professional Blender user or Blender add-on developer. I learn by doing.

IE Autospriter Interface

This Blender add-on automates the process of rendering sprites specifically for Infinity Engine animations. The workflow is divided into logical steps to guide the user, though the sequence of the

first four steps is flexible. Step 5 initiates the rendering process. Using or knowing the [iesdp](#) documentation is essential to understanding how and why things are designed the way they are.
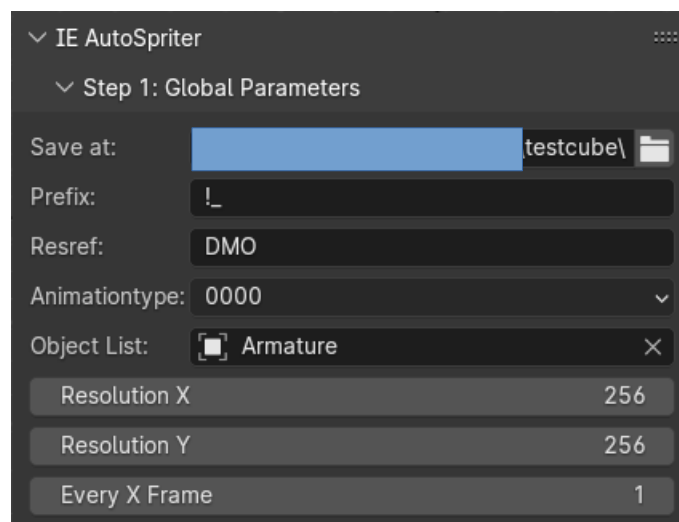
## Closed form
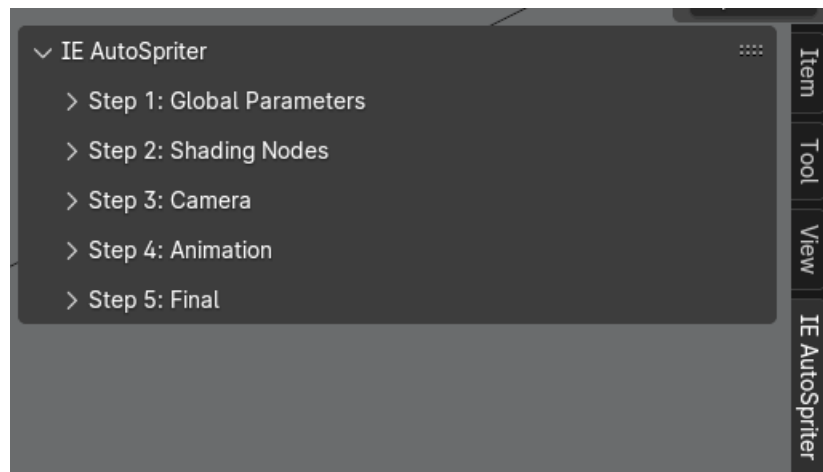
Step 1: Global Parameters

This step defines general settings that apply across the entire sprite rendering process.

- **Save at:** Defines the directory where the generated sprite files will be saved.
- **Prefix:** A user-definable modder prefix that will be added to the beginning of the sprite file names.
- **Resref (Resource Reference):** This is the freely definable part of the sprite file name. Other parts of the file names are fixed and automatically generated by the add-on.
- **Object List:** This must be the armature that contains the NLA tracks. The NLA tracks contain all the animations for the armature (essentially, your creature's animation).
- **Animationtype:** is a dropdown menu that lets you choose the **type of creature or object**

you're rendering based on [iesdp](). Its purpose is to dynamically show or hide the correct set of options in the subsequent steps (*Step 3: Camera* and *Step 4: Animation*), ensuring that the UI only presents relevant choices for the selected animation type.
- **Resolution X:** This refers to the image width resolution
- **Resolution Y:** This refers to the image height resolution
- **Every X frame:** Saves every xth frame as a sprite. This can be useful, for example, when there are too many frames to save as sprites.

Step 2: Shading Nodes
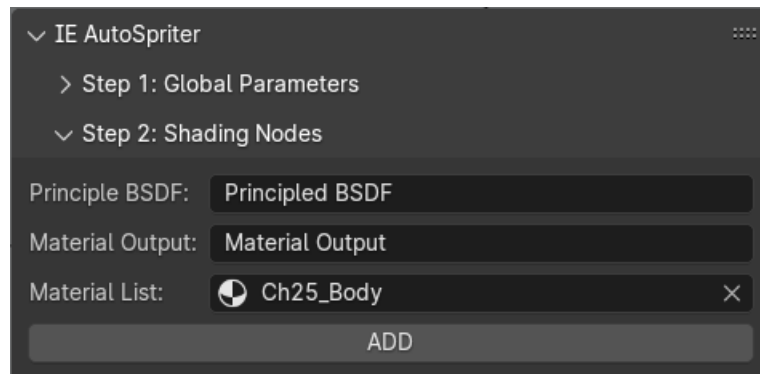
==This optional== step allows for the addition of specific shaders to the material shading setup. This is crucial for Infinity Engine sprites, which require indexed color palettes.

Shader Node String Inputs: These input fields are used to specify the names of existing shader nodes within the material. The new, automatically added shader nodes will be placed and connected between these specified nodes.

Step 3: Camera
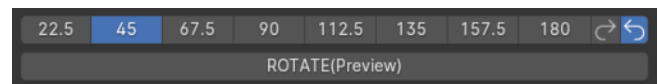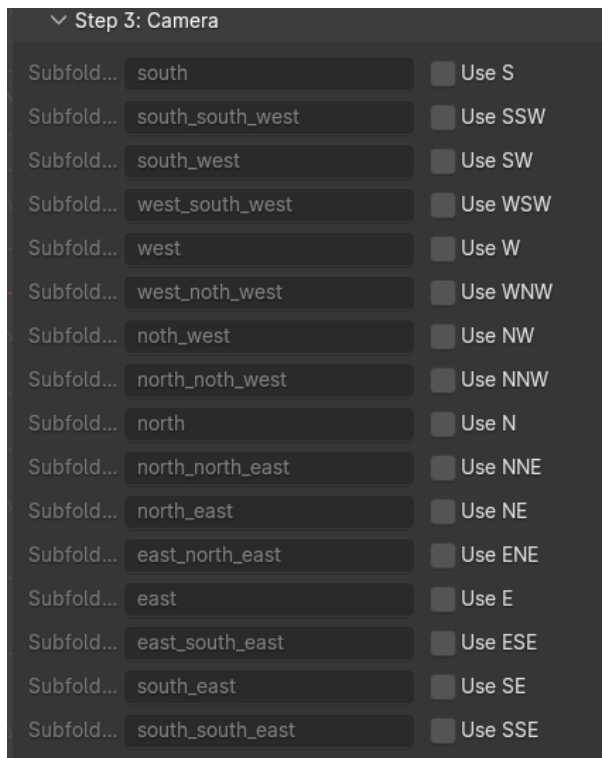
This step manages the output folders and defines which camera orientations will be rendered.

- Orientation Folders: These checkboxes determine whether sprites for a specific orientation will be rendered and saved into their corresponding designated folders. The available orientations are **dynamically displayed based on sthe `Animationtype` selected in Step 1**. If a checkbox is not activated, sprites for that particular orientation will be ignored and not rendered.
- Rotation: The numbers indicate the degree of rotation, and the curved arrows show clockwise or counterclockwise rotation. Use the "**ROTATE (Preview)**" button to start the rotation. The rotation can be applied during the running object animation. This serves as a preview; the rotation parameters are not applied during rendering.
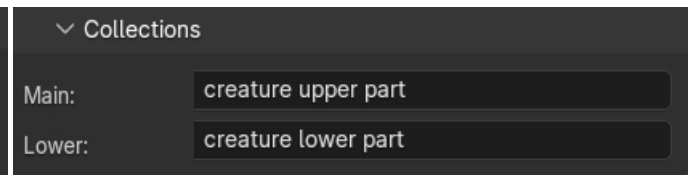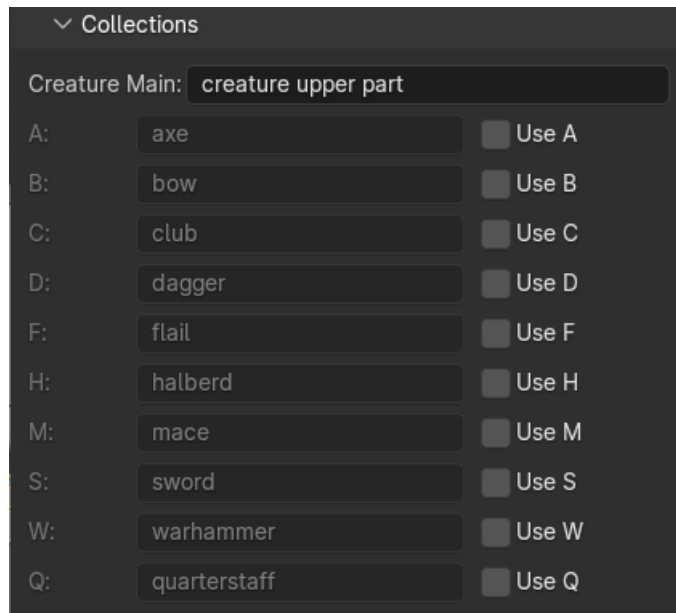
Step 4: Animation
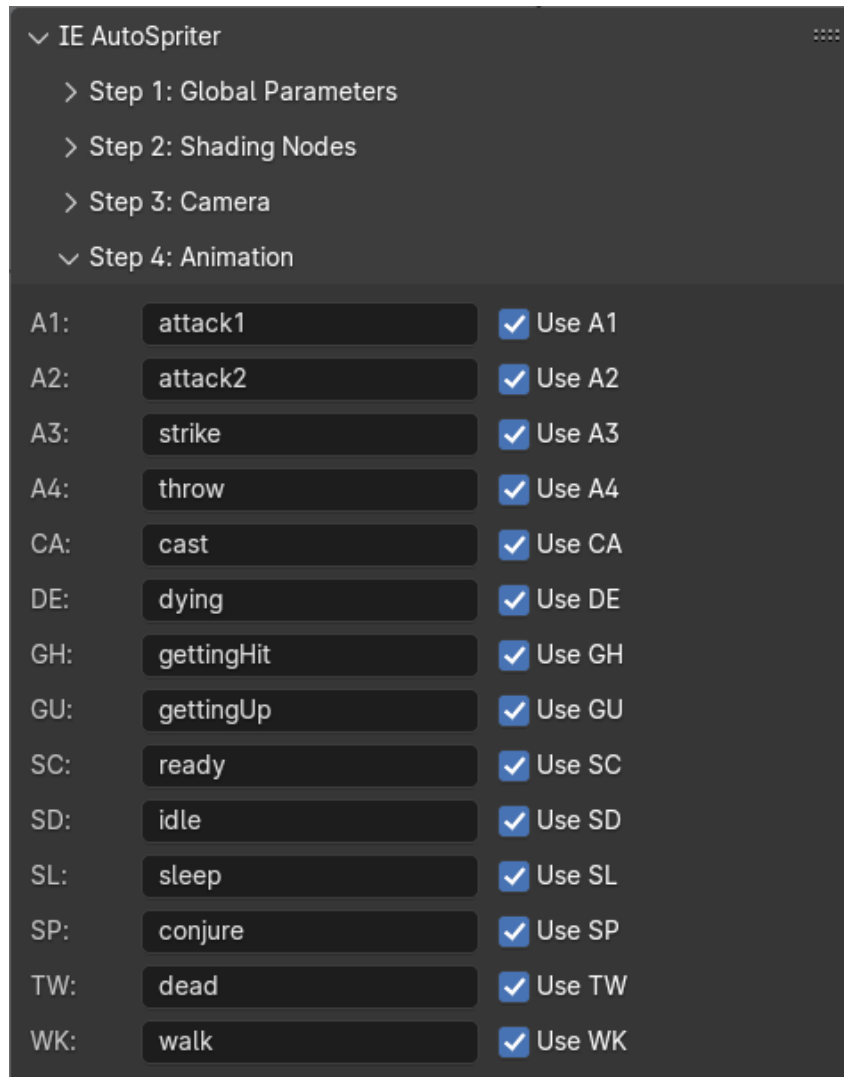
This step defines which animations (Blender Actions) should be rendered and how they are named in the output.

The available animations/actions are **dynamically displayed based on the `Animationtype` selected in Step 1**. If a checkbox is not activated, sprites for that particular animation/action will be ignored and not rendered.

- **Animation(Blender Action):** This refers to the type of animation, corresponding to an "Action" in Blender (e.g., "A1" for Attack Animation 1).
- **Checkbox:** If activated, the specified animation type (e.g., "A1") will be appended to the sprite file name.
- **String Input Field:** The name entered in this input field must precisely match the name of the corresponding Action in Blender.

Collections

The available weapon animations are **dynamically displayed based on the `Animationtype` selected in Step 1**. If a checkbox is not activated, sprites for that particular orientation will be ignored and not rendered.

- **Main:** Specifies the name of the Blender collection containing the main creature model or upper part of effect model(type 0000). This is an absolutely necessary input!
- **Lower:** Specifies the name of the Blender collection containing the creature's lower model or lower part of effect model(type 0000). This is an absolutely necessary input for animation type 3000!
- **A, B, C, etc. (String inputs)**: Define the names of the Blender collections for each weapon type (e.g., "axe", "bow", "club"). The key names(A,B,C, etc.) are also used as wovl (weapon overlay) identifiers in filenames.
- **Use A, Use B, Use C, etc.**: Enable or disable rendering of sprites for each specific weapon collection.

The path for a weapon sprite is as follows(windows): **<Save at>\<Weapon Collection>\ <Subfolder position>**

Step 5: Final
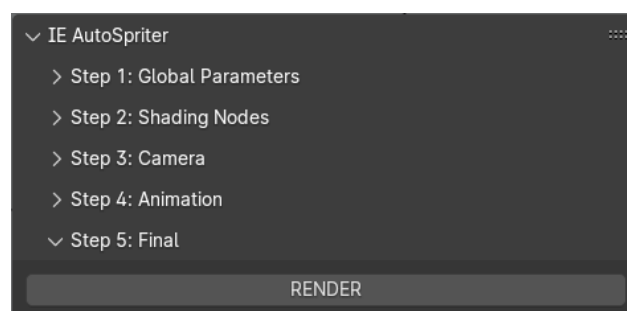
This is the concluding step that triggers the rendering process.

Render: Upon activation, this step initiates the rendering and saving of all sprites according to the settings configured in the preceding steps.

Requirements

It is necessary that the animation is listed in the **Armature** within the **Animation** object. This means that under **Animation** the actions are collected in the **NLA Tracks** (see example in the image below).
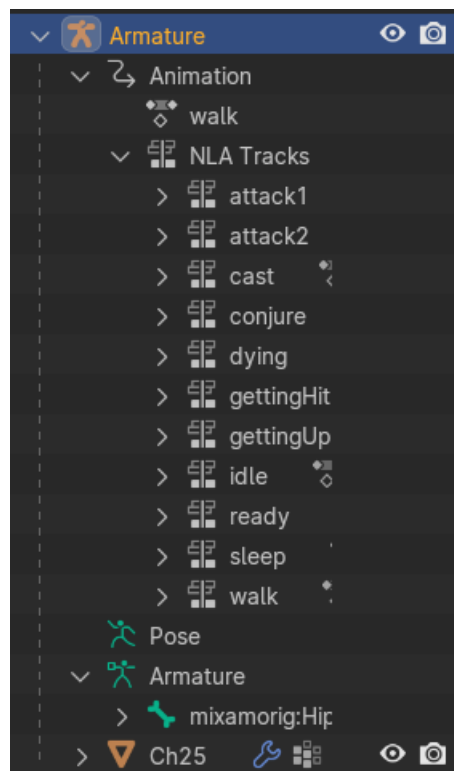

The reason for this is that **IE AutoSpriter uses this list to sequentially specify actions (type of animation) to render**.

The collection containing the main armature for the creature must be activated.

In addition, the names of the animation folders in **step 4** must exactly match the names found in the animation object (see **NLA Tracks**).

The blend file needs the following setup:
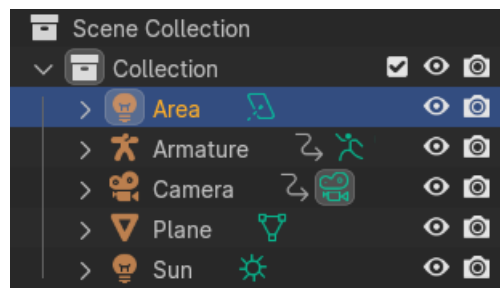
- **Camera**: This is needed to render the creature model.

- **Armature**: This has already been discussed.

- **Plane**: This is needed to create a shadow for the creature model.

- **Area**: This is a light source that should be adjusted as desired.

- **Sun**: This is needed to create a shadow on the **Plane**.

Helpful Insights

The different angles of the creature model are not created by moving the camera, but by rotating the model while the camera remains in a fixed position.
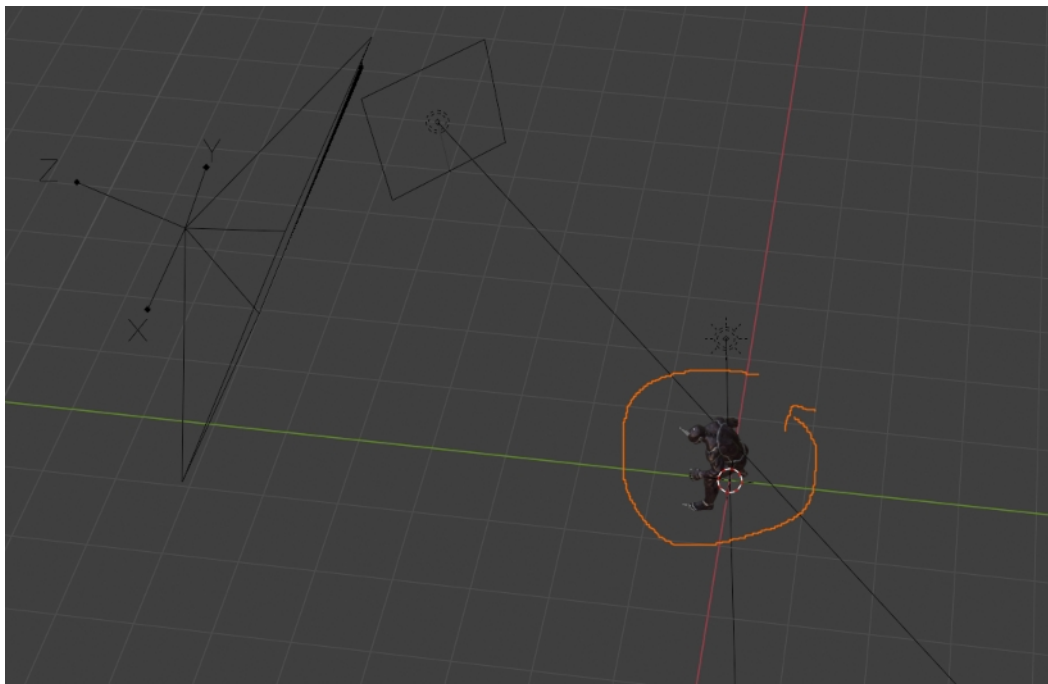
The objects in the collection have a number of specific important parameters that are not covered here, as the details can be found in the save files (<mark>these may be covered in later versions of IE AutoSpriter</mark>).
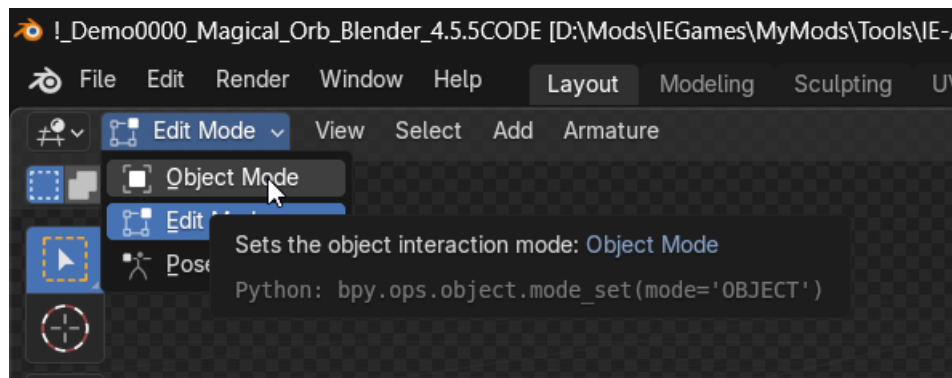
To view the rendering progress and elapsed time, go to "**Toggle System Console**" (see image below). This should be done **BEFORE** rendering, as rendering makes it impossible to interact with Blender.
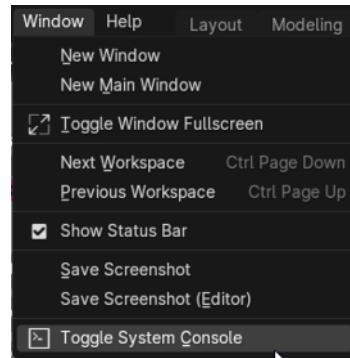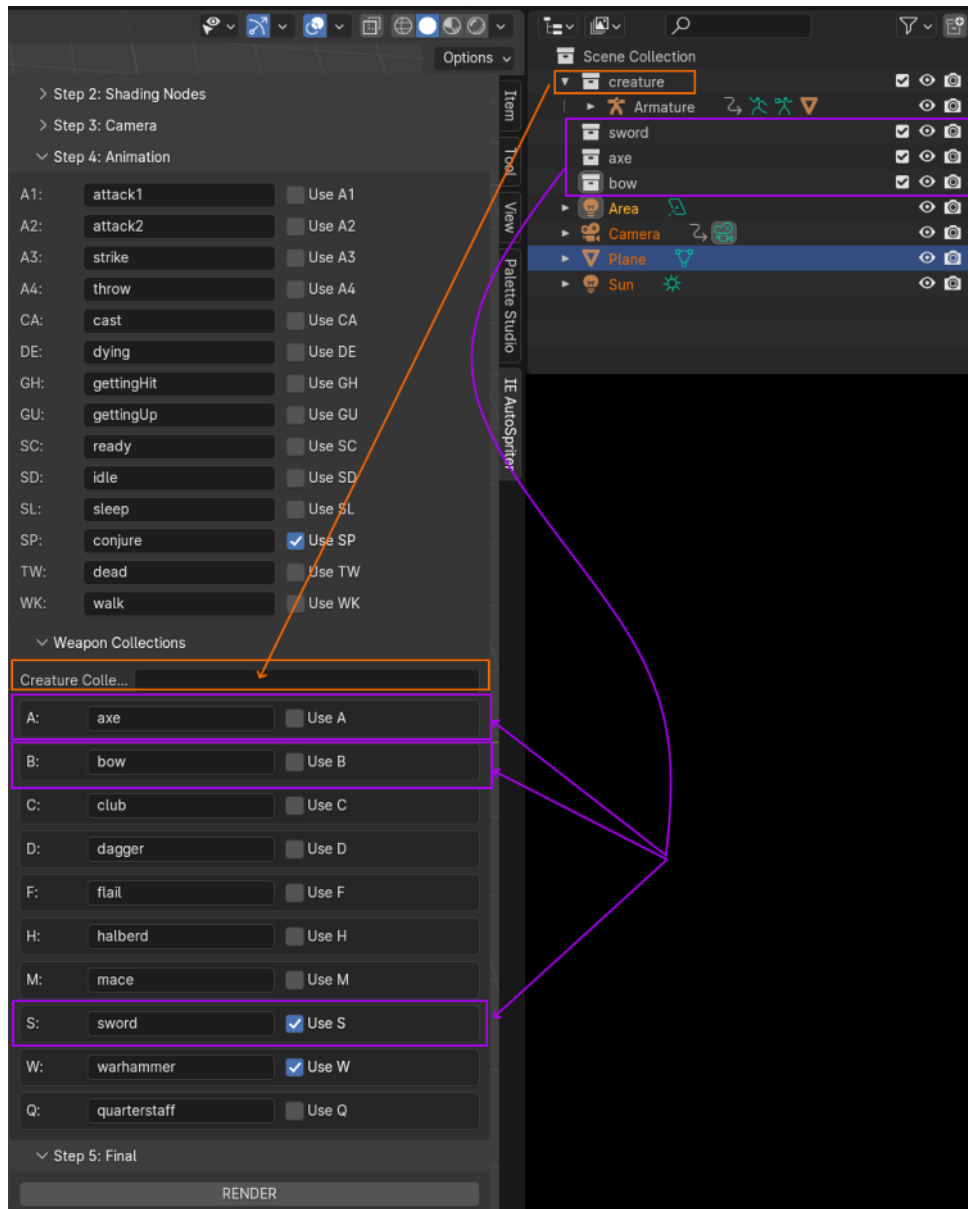
After rendering, the creature's position is reset to the position before rendering started.

Object Mode:

The "3D viewport" must be in "object mode", otherwise a corresponding error message will be displayed when the "RENDER" button is pressed.

Each weapon to be rendered requires its own collection. This is (at the moment) only required for type E000. Everything else, such as area, camera, plane, and sun, can or should be outside of other subcollections. Depending on the lighting (sun), the weapon rendering may contain shadows of both the weapon and the creature. This shouldn't be a problem, since the creature and weapon shadows are merged when the BAM files are combined anyway, and since the weapon and creature shadows (must) be the same black color, the difference shouldn't be visible. This is only theoretical, as no tests have been conducted so far.
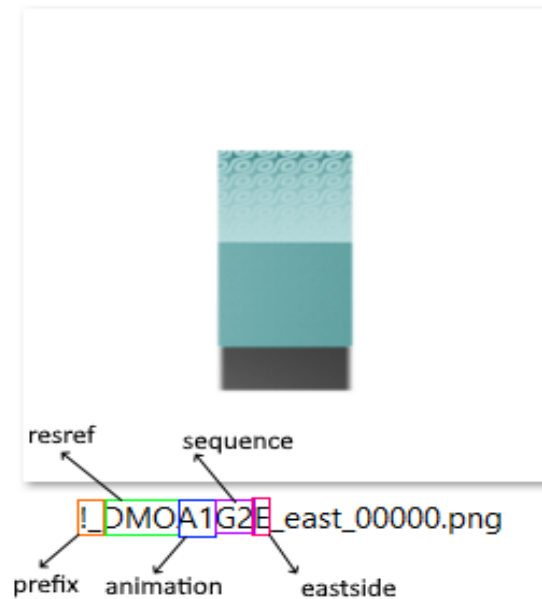
Character Naming

The add-on uses a class called IEAS_AnimationTypes to determine which characters to use for each animation. These characters (e.g., G1, G2, G3, and E) serve to identify the specific sequence a file belongs to. The E character, specifically, is used for "Eastside animations" and is added to the filename depending on the Animationtype. The user doesn't need to manually input these characters; the add-on handles this internally.

For animation types that handle quadrants, the handling is slightly different. The quadrant number is added it can be identified which quadrant it belongs to. For the 1000 monster quadrant type, the quadrants are divided into 4 equal parts with the resolution of the original image.
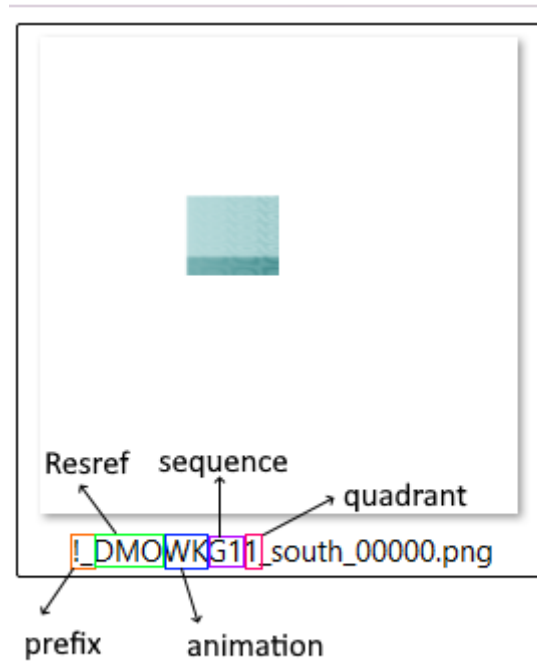
The filename construction depends on the animation type, as some do not have sequences or quadrants.

In general, if an animation type requires sprite splitting (image/animation splitting), the rendering resolution must be divisible by 2 or 3. 1000 monster quadrants require a resolution divisible by 2, and 1000 monster multi must be divisible by 2 for "split_bams=0" and by 3 for "split_bams=1."

Animation Type 5000/6000 character old

This type is special in that its sequences (regarding file naming) are inconsistent. It is strongly recommended that you consult the iesdp documentation to understand the exact reasons for this.

Two points are particularly important: First, this animation type has the W2 and W2E sequences but does not utilize the available orientations, while the other sequences do not use the W2 and W2E orientations. This is relevant because, although IE AutoSpriters offers all 16 orientations when this type is selected, it uses them internally according to the iesdp documentation.

For example, no Walk2 animation sprites are generated for the "South" orientation.

Save file Cycles 4.5.5 LTS

Found in „**IE-AutoSpriter-\save files\!_Template_CYCLES_Blender_4.0.blend**"

Scene Collections
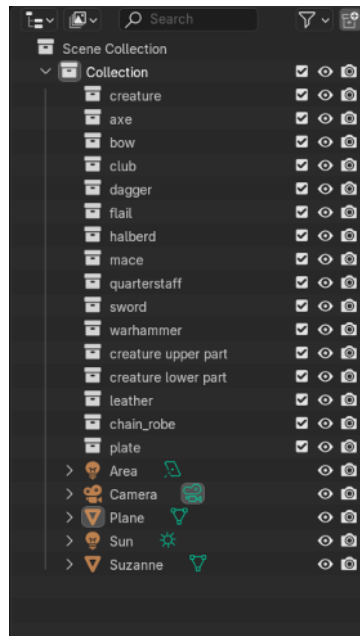
„creature" is generally the main collection, which should be used for creature animations. The creature collection should also be used if no weapon animations are specified. Empty scenes should not cause rendering issues.

For animation type 3000, the "**creature upper part**" and "**creature lower part**" collections are relevant. The others can be deleted.
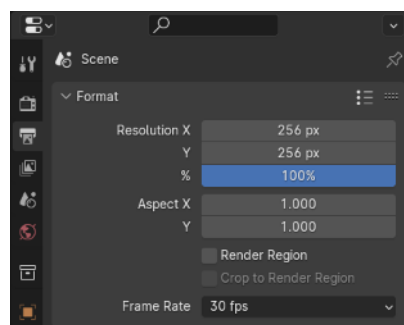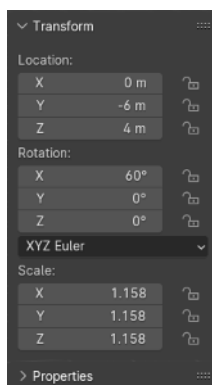
Render Engine
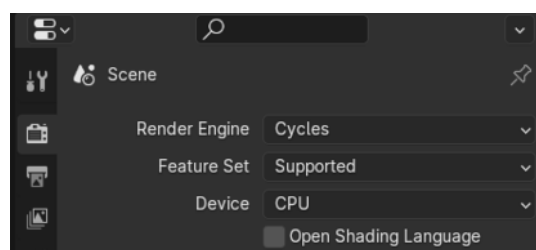
The rendering engine is set to Cycles (instead of EEVEE). This allows the "**Plane**" object to be rendered transparently without the "**Sun**" object losing its ability to cast shadows.

The render settings for the engine is as follows(see next image below):
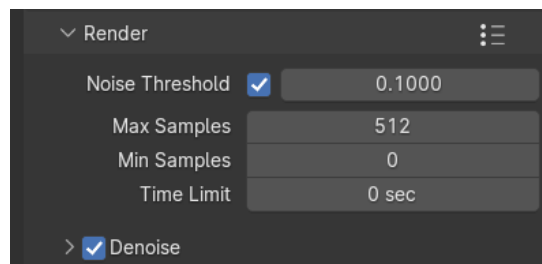

Camera settings

The camera settings can be adjusted to suit your personal preferences if needed. For example, the camera angle is set to 60°, although I've heard arguments for 45°. **After seeing someone else compare an image** of a custom character animation and an in-game animation at 45° and 60°, **I decided to go with 60°**. The rendering resolution is set to **256px to 256px**(px stands for pixels) and frame rate to **30fps**(frames per second).

Sun
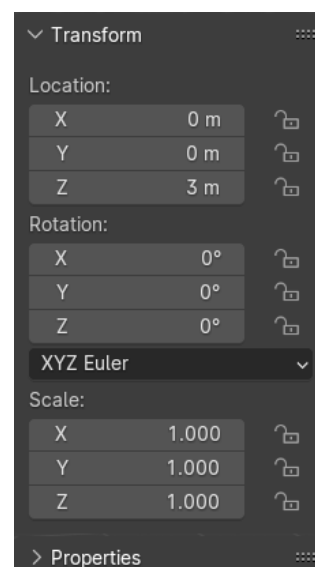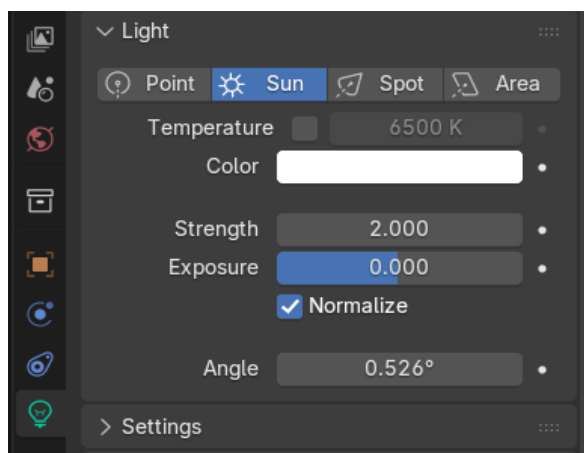
These settings are optional but worked reasonably well (**lumens 2.000 and Z-distance 3m**), but will likely need to be adjusted depending on the model that needs to be rendered.

Plane

The object „**Plane**" is used to capture the shadow cast by sunlight on an object. To prevent the plane from being included in rendering without losing the shadow, the "**Transparent**" switch is enabled under „**Rend → Film**" checked(see left screenshot). To use the pane as shadow catcher, it needs to be enabled as such(see right screenshot).
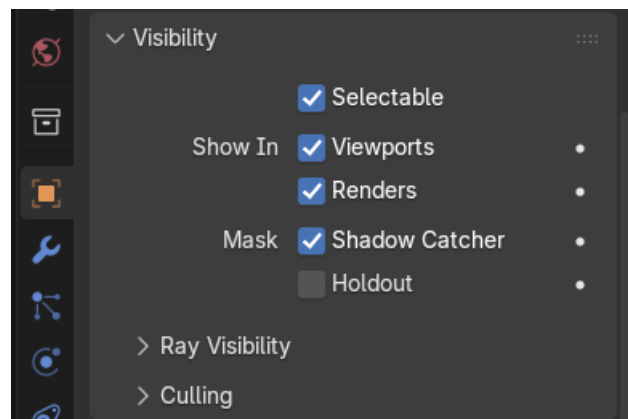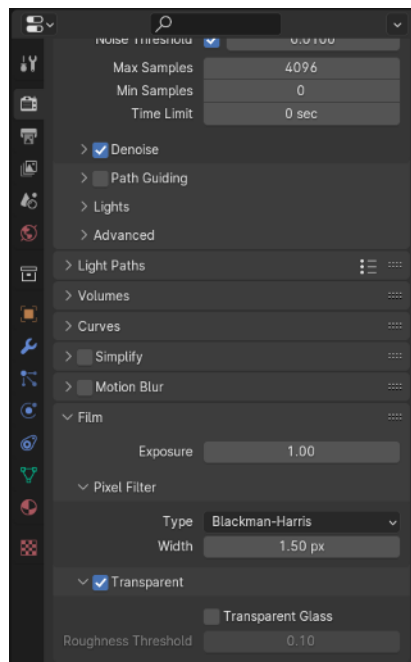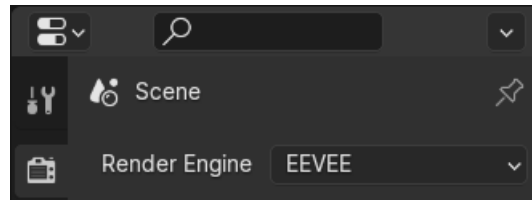

Area

The „**Area**" object is a purely optional light source. Its main purpose is to make the model's texture more visible.

Save file EEVEE 4.5.5 LTS

Found in „**IE-AutoSpriter-\save files\!_Template_EEVEE_Blender_4.5.5.blend**" and is similar to cycles save file.

Render Engine

The rendering engine is set to Cycles (instead of EEVEE). This significantly [reduces rendering time](made with blender 4.0) compared to Cycles. However, it also brings its own challenges for the „**Plane**".

Plane

The object „**Plane**" is used to capture the shadow cast by sunlight on an object. To make everything work, the layer has its own materials and shading nodes within them (see images below).
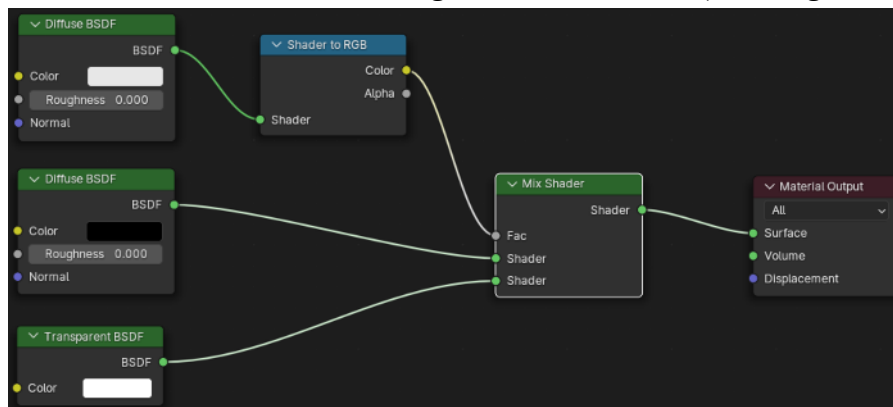
Option 1: Shadow_Catcher2_Material
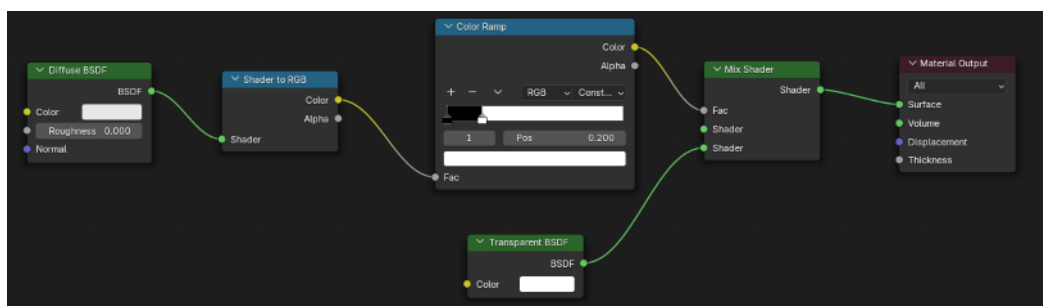
Option 2: Shadow_Catcher1_Material

This was realized with the help of the following video:

- [How to make a shadow catcher in Blender 4.0 Cycles and Eevee by MK Graphics](#).
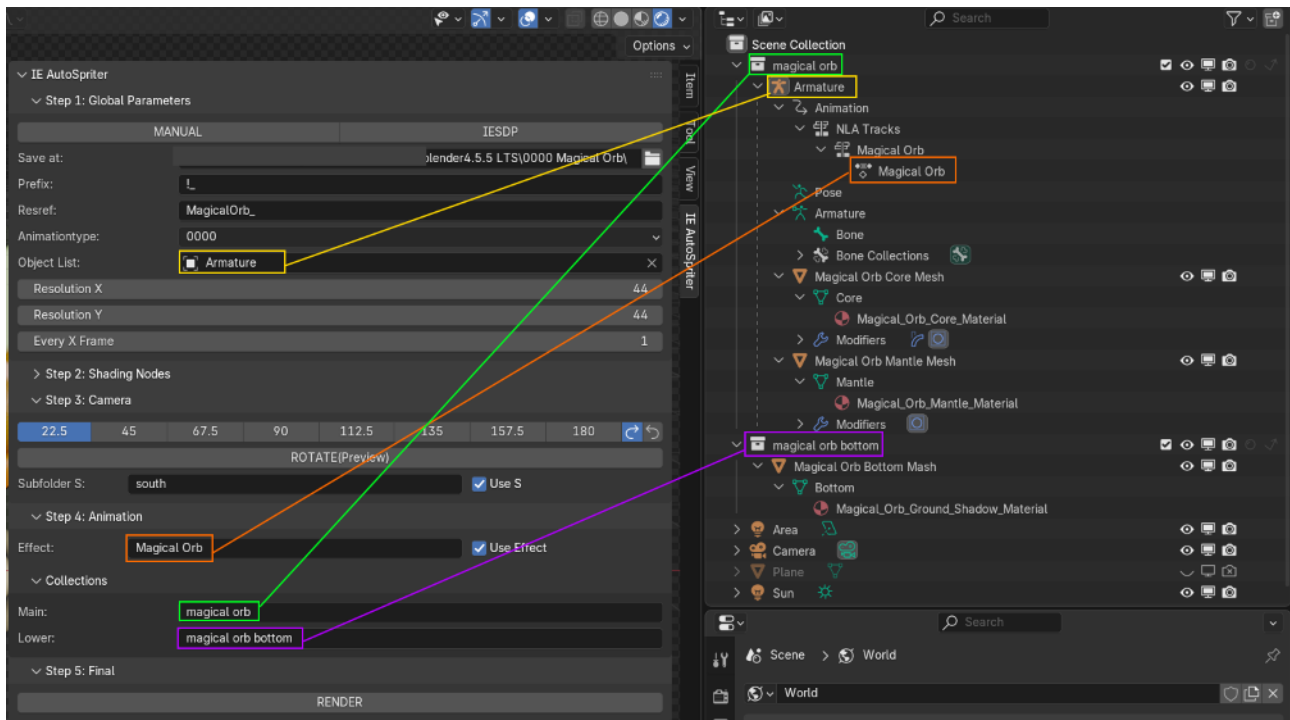
- [Blender 4.4.0 Shadow Catcher Tutorial - Eevee](#)

Demo files 4.5.5(TODO)

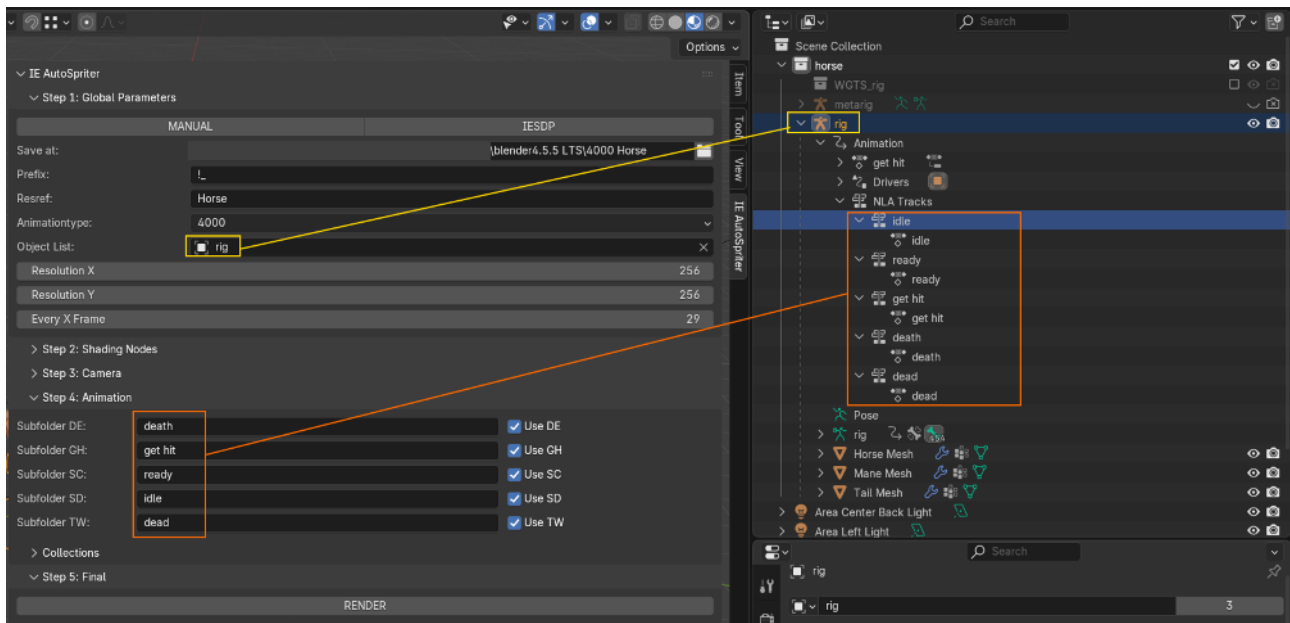~~Found in „**IE-AutoSpriter-\demos\blend4.0**~~

- ~~**!_Demo_CYCLES_Blender_4.0.blend**~~

- ~~**!_Demo_EEVEE_Blender_4.0.blend**~~

~~It serves as a very simple example and is used for performance testing of IE AutoSpriter. They are intended ONLY to demonstrate/test the features of IE Autospriter.~~

Animation Type 0000
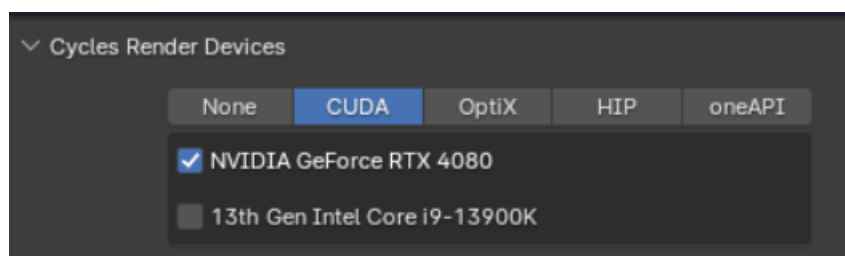
Animation Type 4000(NEW)

Troubleshooting/FAQ

„Save at" path empty, what happens?

This saves the animation folders and their contents directly to the Blender project path. However, it's more human-readable to include this path in the "**Save as**" option.
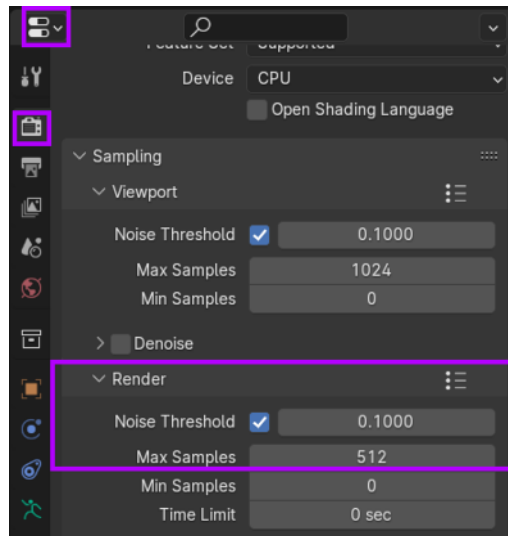
How can time performance be increased?

Testresults can be found [here](#).

Usage of GPU power

This option can be found under Edit→Preferences...→System. However, the change [didn't result in any performance improvements on my PC](#).

Alter sampling parameters

You can find this in the editor window under ==Render (camera icon) → Sampling → Render==.

Increasing the "**Noise Threshold**" and reducing the "**Max. Samples**" should improve the time performance.

Further reducing the "**Max. Samples**" (e.g., to 256) should certainly be possible without any significant loss in sprite quality, but increasing the "**Noise Threshold**" may not be as effective. Users will have to test for themselves what works best for them.
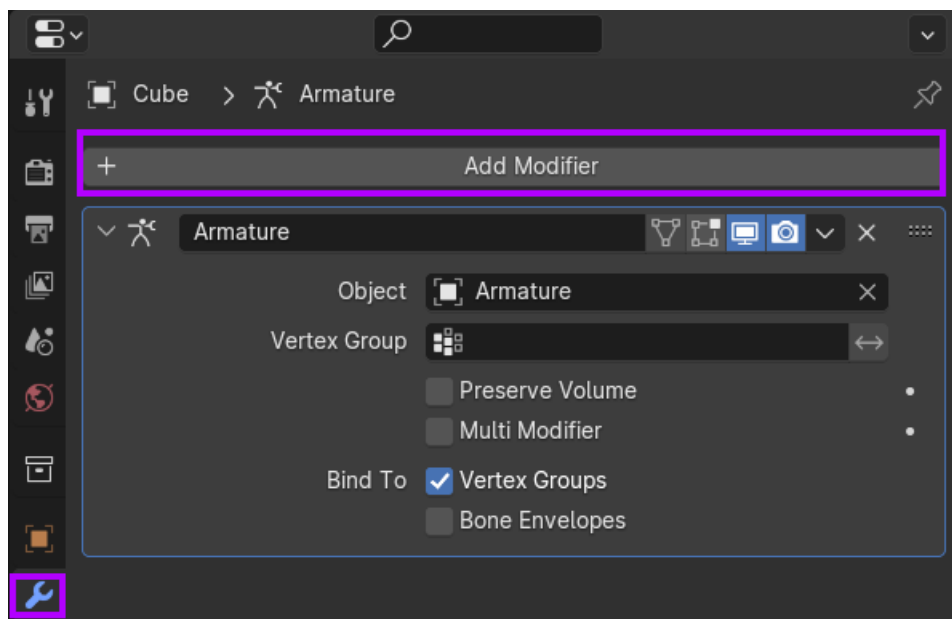
Use Decimate Modifier

This should only be considered as a last resort as it will definitely reduce model quality. Decimate Modifier can be used as follows:
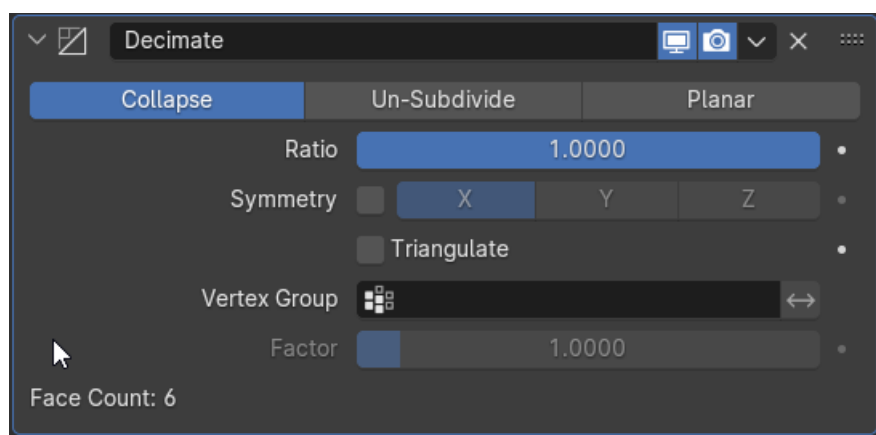
1. Choose your model in this particular case cube
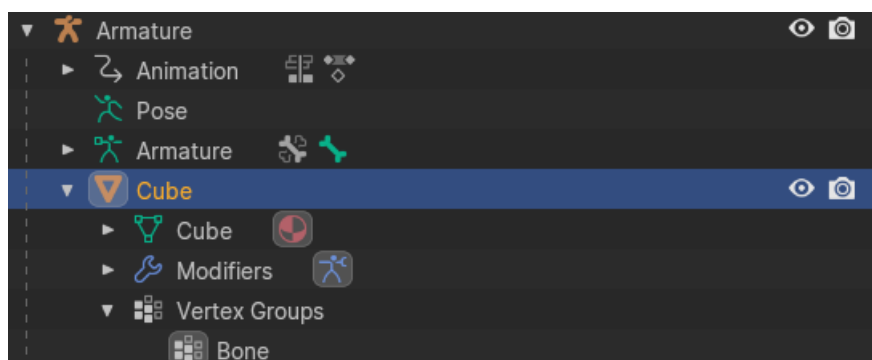
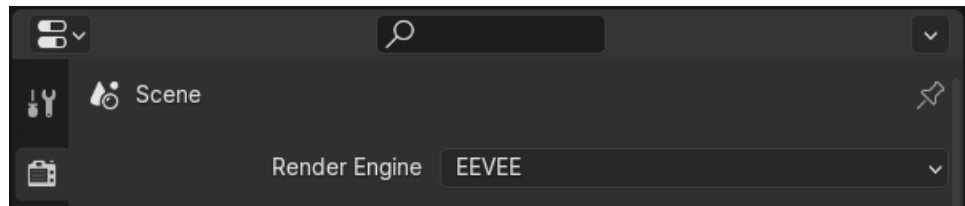2. Press „**Add Modifier**" and select ==Generate → Decimate==

3. **Reducing Ratio** will improve performance time([see testexample](#))

Change Render Engine

Switching to EEVEE will significantly reduce rendering time, but this is not currently supported by the template save files that ship with IE AutoSpriter and may be added later in the development process.

[Here is the specific IE AutoSpriter test result for EEVEE](#)

Specific animation types spam new images in Image Editor

Yes, this is a known problem that should not have any side effects and can be solved as follows

- File → Clean Up → Recursive Unused Data-Blocks

Note this will also delete things that are not used but maybe created for later use, unless „[Fake User](#)" is active for that specific object.

How does the resolution setting work?
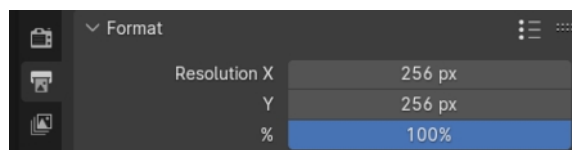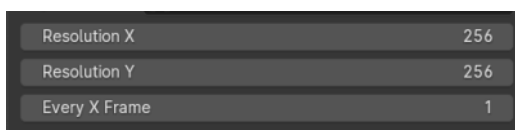
If the resolution setting in IE AutoSpriter is clicked to be manipulated then it will have its values transferred to blender's render settings in "Output Properties". If the values are different in both setting(IE AutoSpriter and "Output Properties") then the one from "Output Properties" will be transferred to the resolution settings in IE AutoSpriter after "RENDER" button in step 5 is pressed. This is to prevent having the IE AutoSpriter resolution setting constantly manipulating the "Output Properties" resolution if the user prefers to use the resolution of "Output Properties" (e.g. for camera view).

Don't use special characters for the string properties

Since some string properties are used to create folders, a special character such as "/" will result in the creation of a subfolder.