

## Index

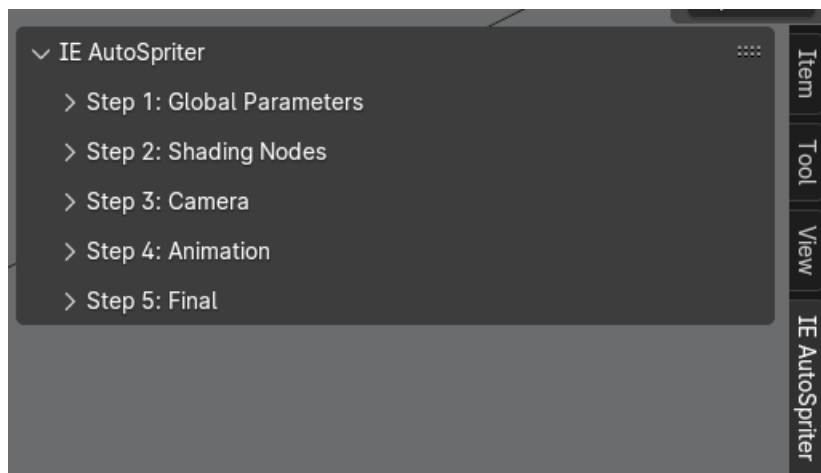
IE Autospriter Interface.....	2
Closed form.....	2
Step 1: Global Parameters(CHANGED).....	2
Step 2: Shading Nodes.....	3
Step 3: Camera (CHANGED).....	4
Step 4: Animation(CHANGED).....	5
Weapon Collection(CHANGED).....	6
Step 5: Final.....	6
Requirements.....	7
Helpful Insights.....	8
Save file Cycles 4.0.....	10
Scene Collections.....	10
Render Engine.....	10
Camera settings.....	11
Sun.....	12
Plane.....	12
Area.....	13
Save file EEVEE 4.0.....	13
Render Engine.....	13
Plane.....	13
Demo files 4.0.....	14
Shape.....	14
Shader Nodes.....	15
Bones.....	16
Scene Collections.....	17
Troubleshooting/FAQ.....	18
RuntimeError - context is incorrect.....	18
OSError – incorrect directory.....	18
„Save at“ path empty, what happens?.....	19
How can time performance be increased?.....	19
Usage of GPU power.....	19
Alter sampling parameters.....	20
Use Decimate Modifier.....	20
Change Render Engine.....	21
Incorrect creature Z rotation in Blender after rendering.....	22

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

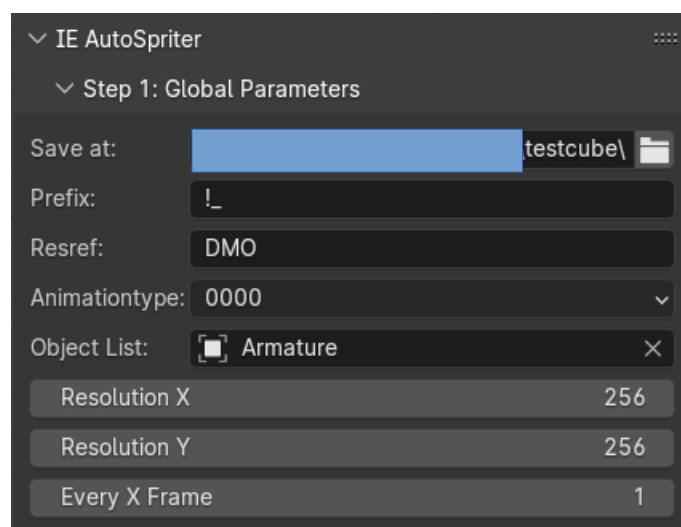
## IE Autospriter Interface

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



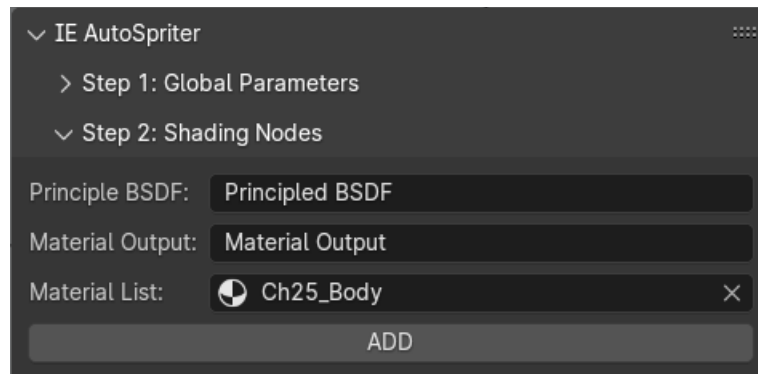
### Step 1: Global Parameters(CHANGED)



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.

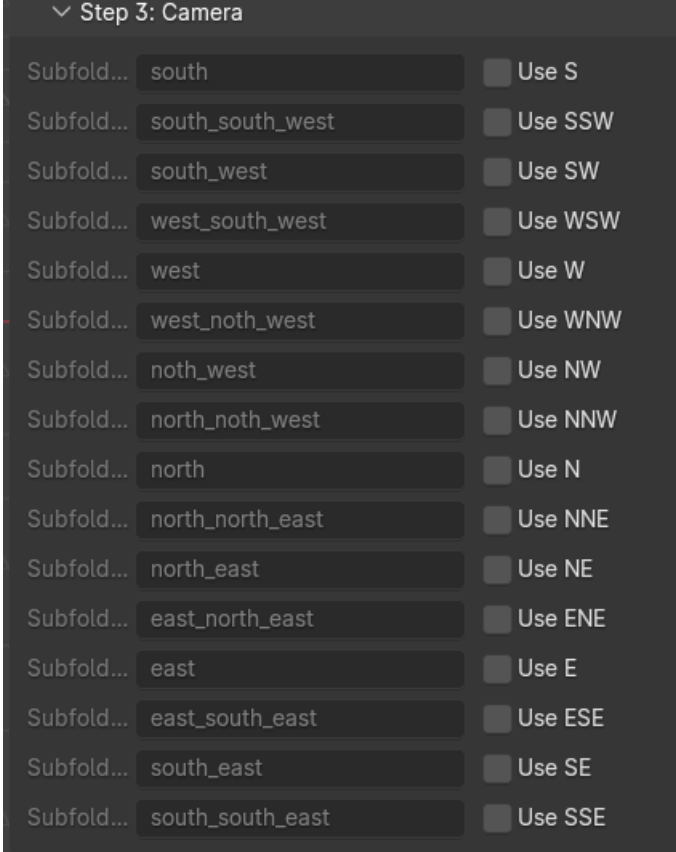
## Step 2: Shading Nodes



**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 (CHANGED)

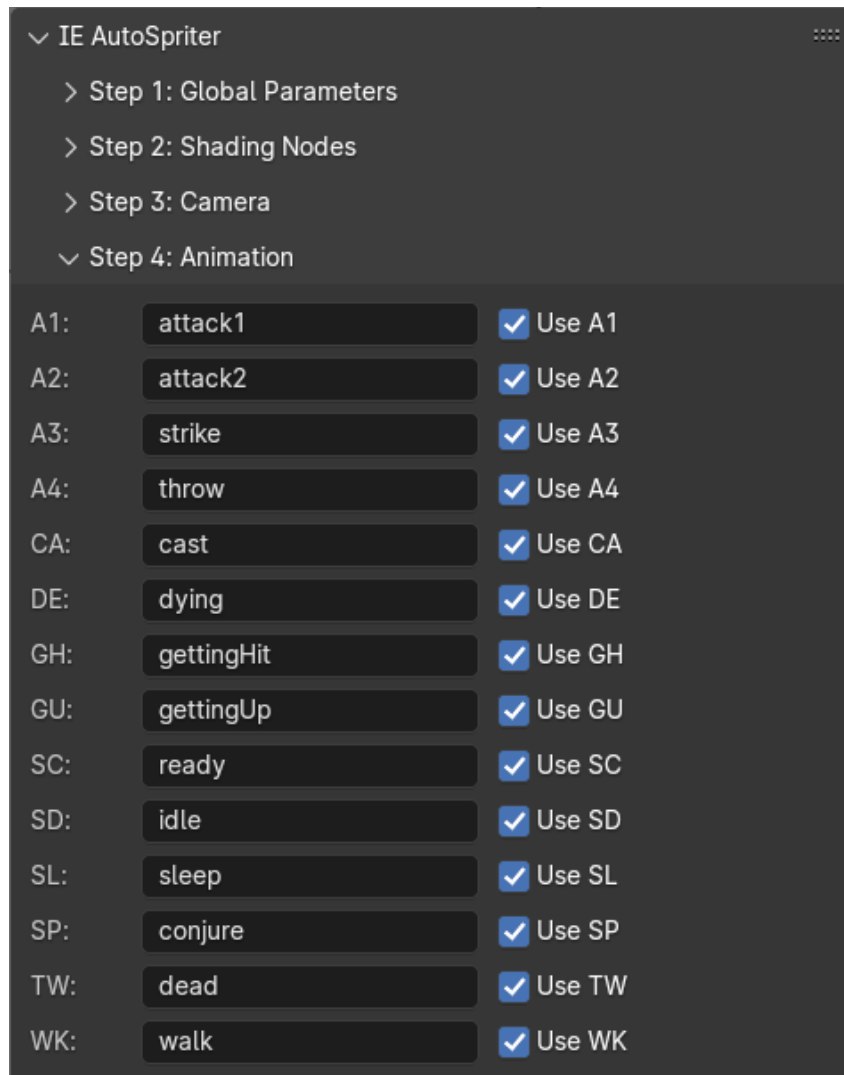


Subfold...	Use
south	S
south_south_west	SSW
south_west	SW
west_south_west	WSW
west	W
west_noth_west	WNW
noth_west	NW
north_noth_west	NNW
north	N
north_north_east	NNE
north_east	NE
east_north_east	ENE
east	E
east_south_east	ESE
south_east	SE
south_south_east	SSE

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 the Animationtype selected in Step 1**. If a checkbox is not activated, sprites for that particular orientation will be ignored and not rendered.

## Step 4: Animation(CHANGED)



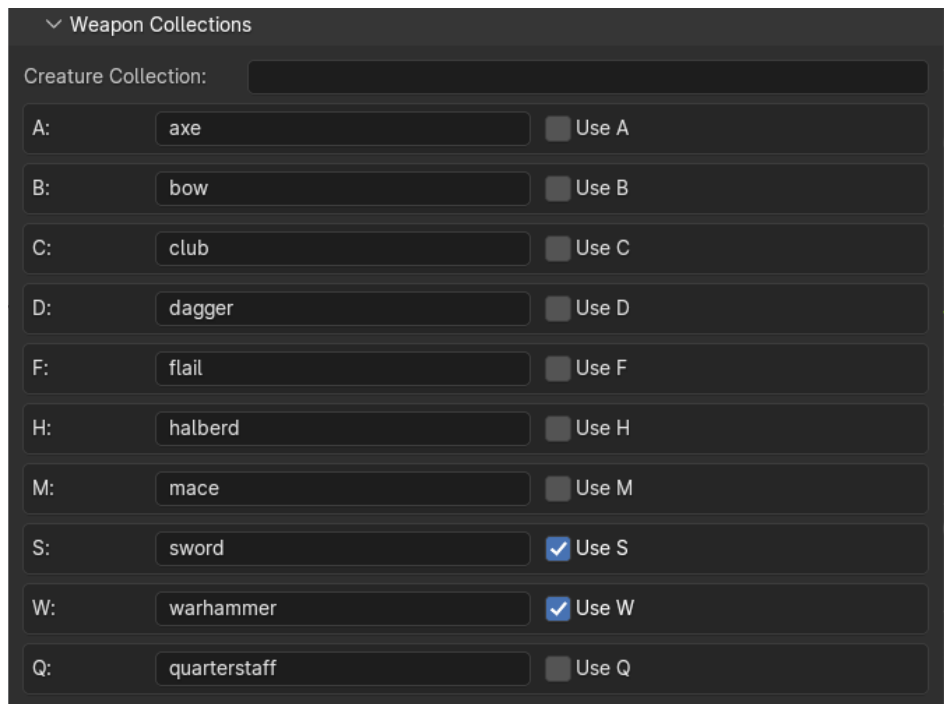
Animation Type	Animation Name	Use
A1:	attack1	<input checked="" type="checkbox"/> Use A1
A2:	attack2	<input checked="" type="checkbox"/> Use A2
A3:	strike	<input checked="" type="checkbox"/> Use A3
A4:	throw	<input checked="" type="checkbox"/> Use A4
CA:	cast	<input checked="" type="checkbox"/> Use CA
DE:	dying	<input checked="" type="checkbox"/> Use DE
GH:	gettingHit	<input checked="" type="checkbox"/> Use GH
GU:	gettingUp	<input checked="" type="checkbox"/> Use GU
SC:	ready	<input checked="" type="checkbox"/> Use SC
SD:	idle	<input checked="" type="checkbox"/> Use SD
SL:	sleep	<input checked="" type="checkbox"/> Use SL
SP:	conjure	<input checked="" type="checkbox"/> Use SP
TW:	dead	<input checked="" type="checkbox"/> Use TW
WK:	walk	<input checked="" type="checkbox"/> Use WK

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 Animation type 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.

## Weapon Collection(CHANGED)

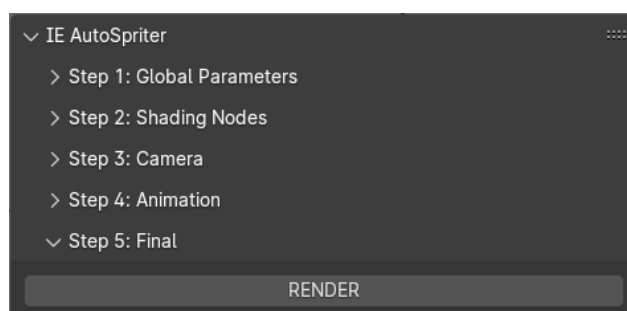


The available weapon animations are **dynamically displayed based on the Animation type selected in Step 1**. If a checkbox is not activated, sprites for that particular orientation will be ignored and not rendered.

- **Creature Collection:** Specifies the name of the Blender collection containing the main creature model. This is an absolutely necessary input!
- **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

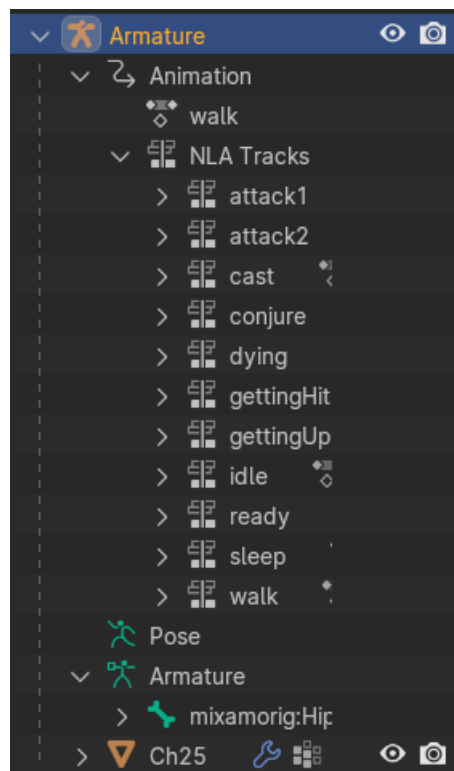


This is the concluding step that triggers the rendering process.

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

## Requirements

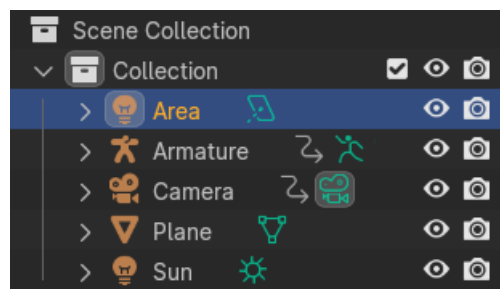
It is necessary that the animation is listed in the **Armature** within the **Animation** object. This means that under **Animation** the actions are collected in the **NLA Tracks** (see example in the image below).



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

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

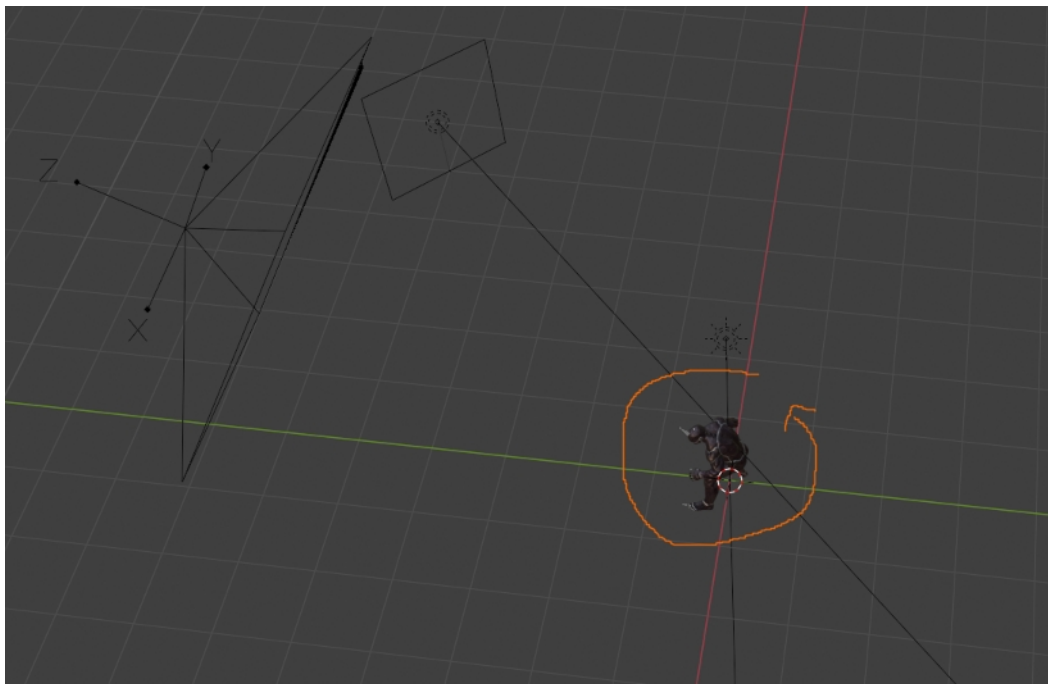
The blend file needs the following setup:



- **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

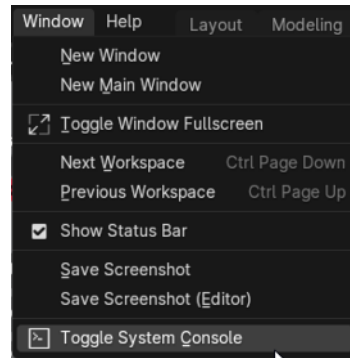
The different angles of the creature model are not created by moving the camera, but by rotating the model while the camera remains in a fixed position.



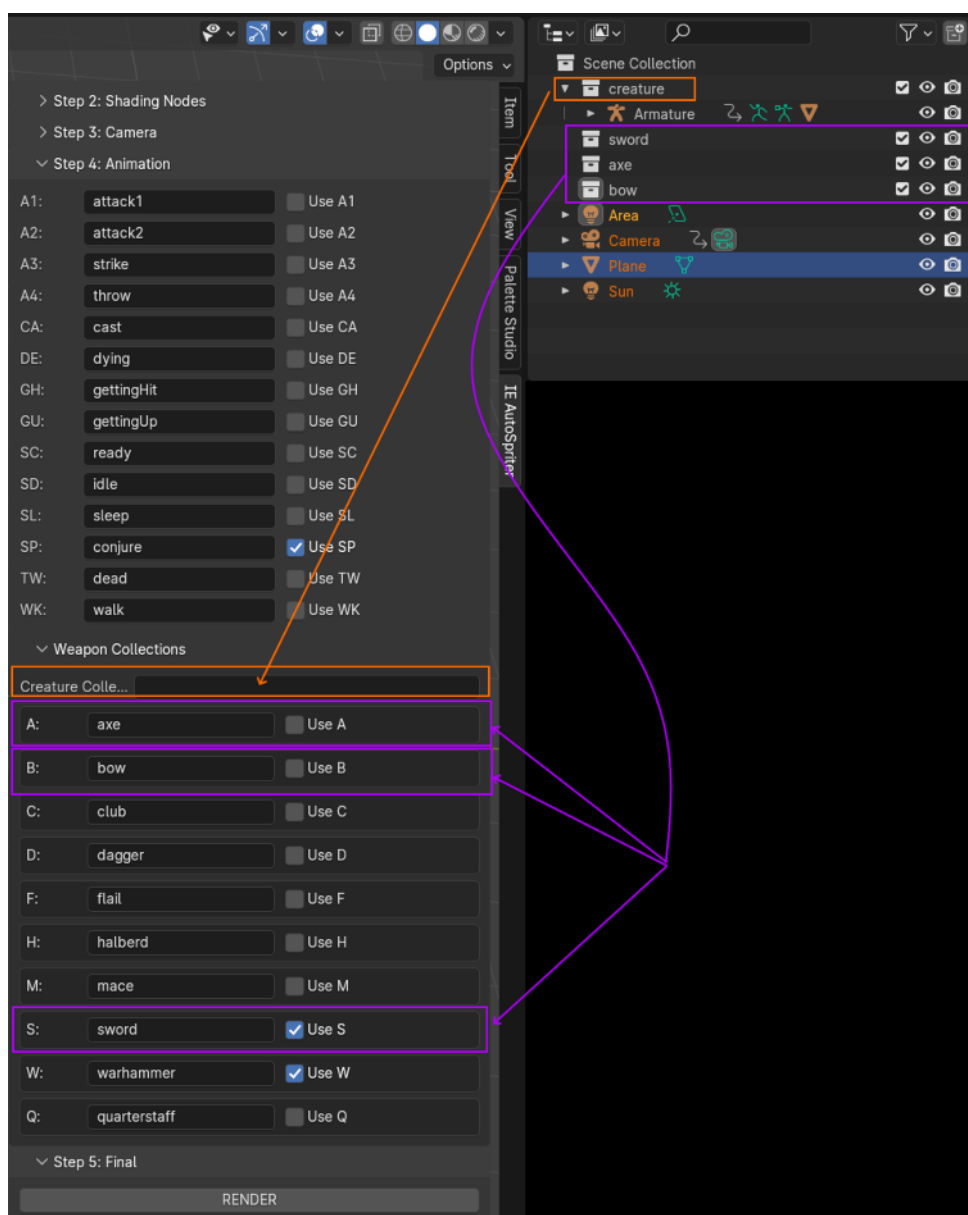
The objects in the collection have a number of specific important parameters that are not covered here, as the details can be found in the save files (these may be covered in later versions of IE AutoSpriter).

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.

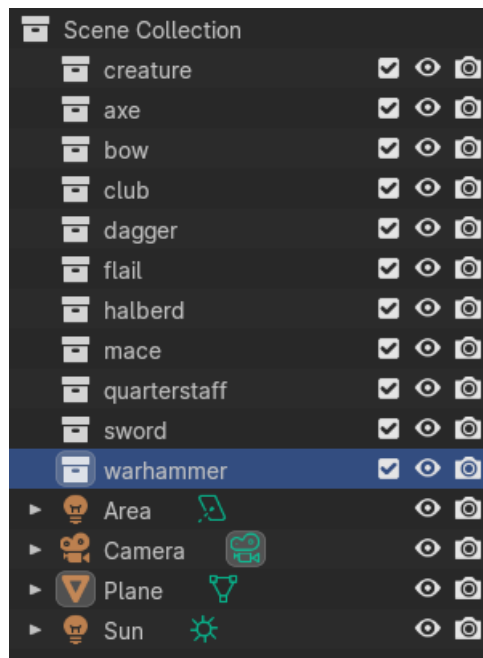


Each weapon to be rendered requires its own collection. This is (at the moment) only required for [type E000](#). Everything else, such as area, camera, plane, and sun, can or should be outside of other subcollections. Depending on the lighting (sun), the weapon rendering may contain shadows of both 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.

## Save file Cycles 4.0

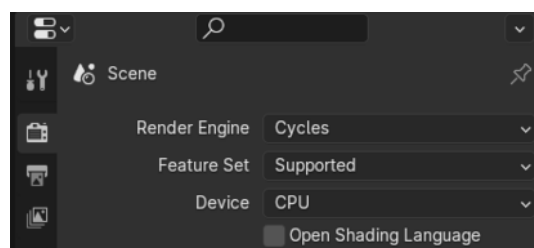
Found in „IE-AutoSpriter-\save files\!\_Template\_CYCLES\_Blender\_4.0.blend“

## Scene Collections



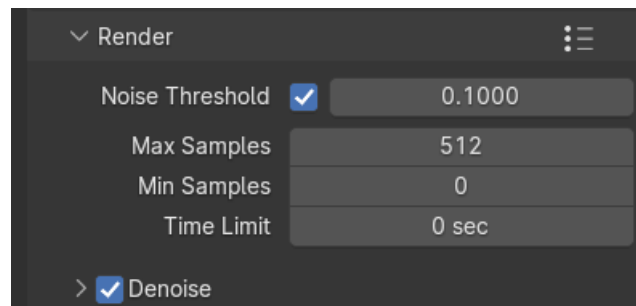
„creature“ is generally the main collection, which should be used for creature animations. The creature collection should also be used if no weapon animations are specified. Empty scenes should not cause rendering issues.

## Render Engine

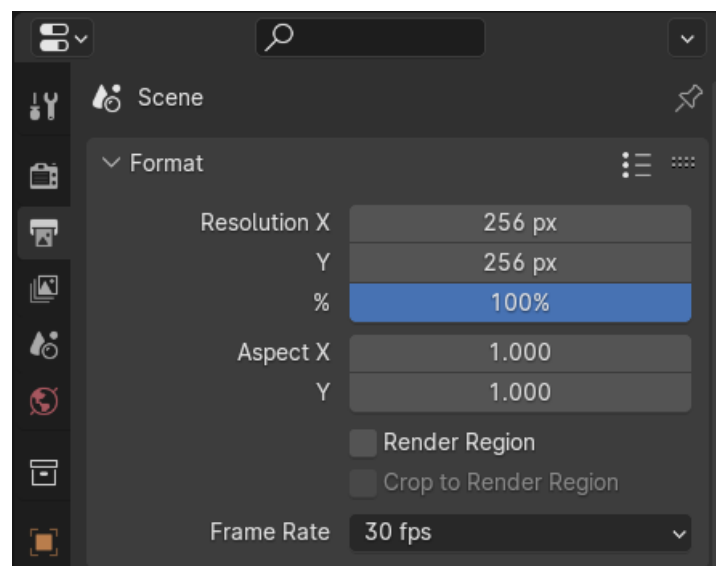
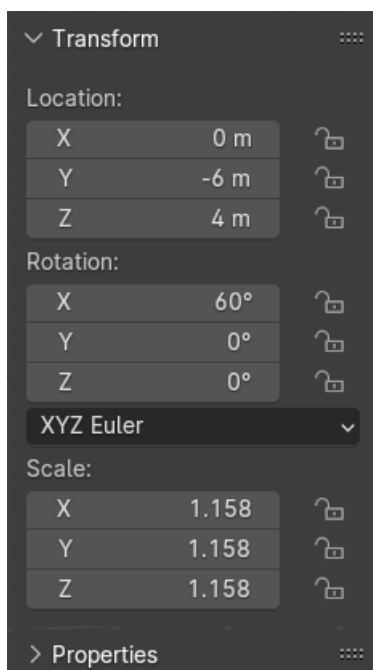


The rendering engine is set to Cycles (instead of Eevee). This allows the "**Plane**" object to be rendered transparently without the "**Sun**" object losing its ability to cast shadows.

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

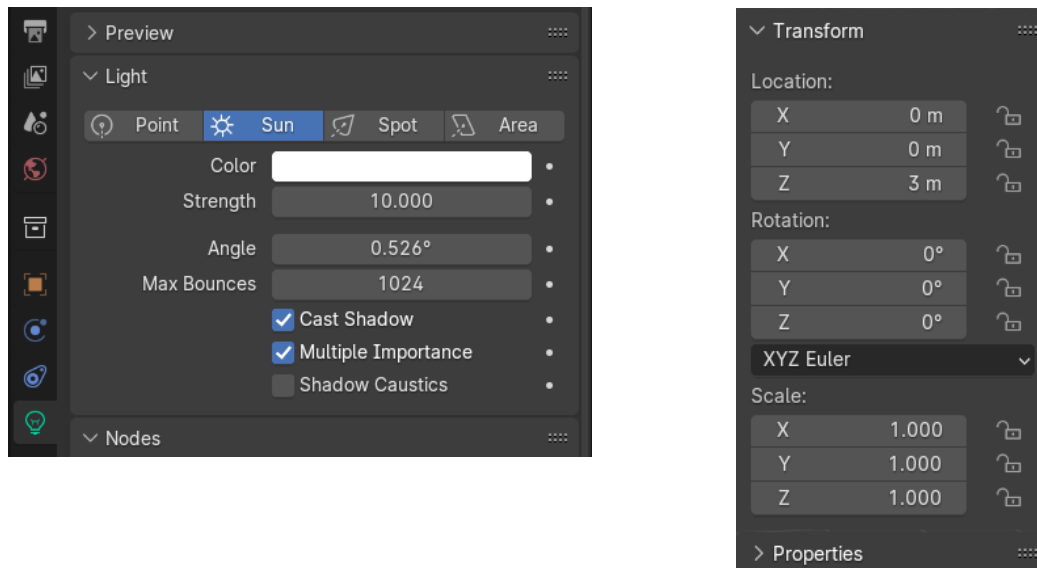


## Camera settings



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).

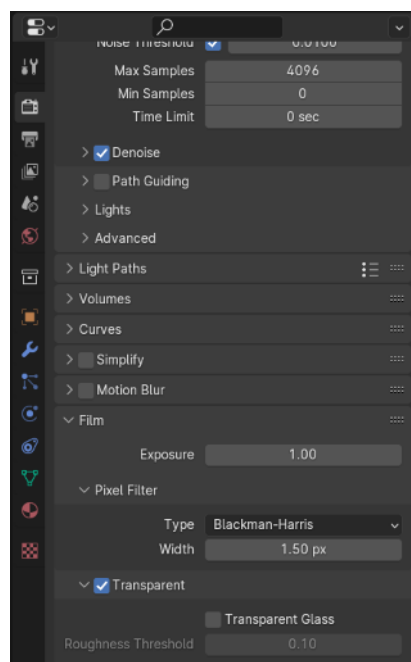
## Sun



These settings are optional but worked reasonably well (**lumens 10,000 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 „**Render** → **Film**“ checked(see screenshot above).



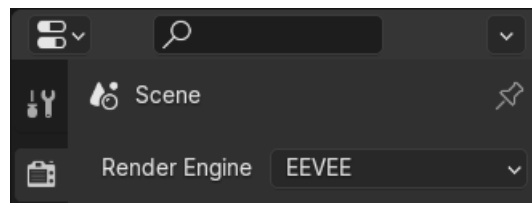
## Area

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

## Save file EEVEE 4.0

Found in „IE-AutoSpriter-\save files\!\_Template\_EEVEE\_Blender\_4.0.blend“ and is similar to cycles save file.

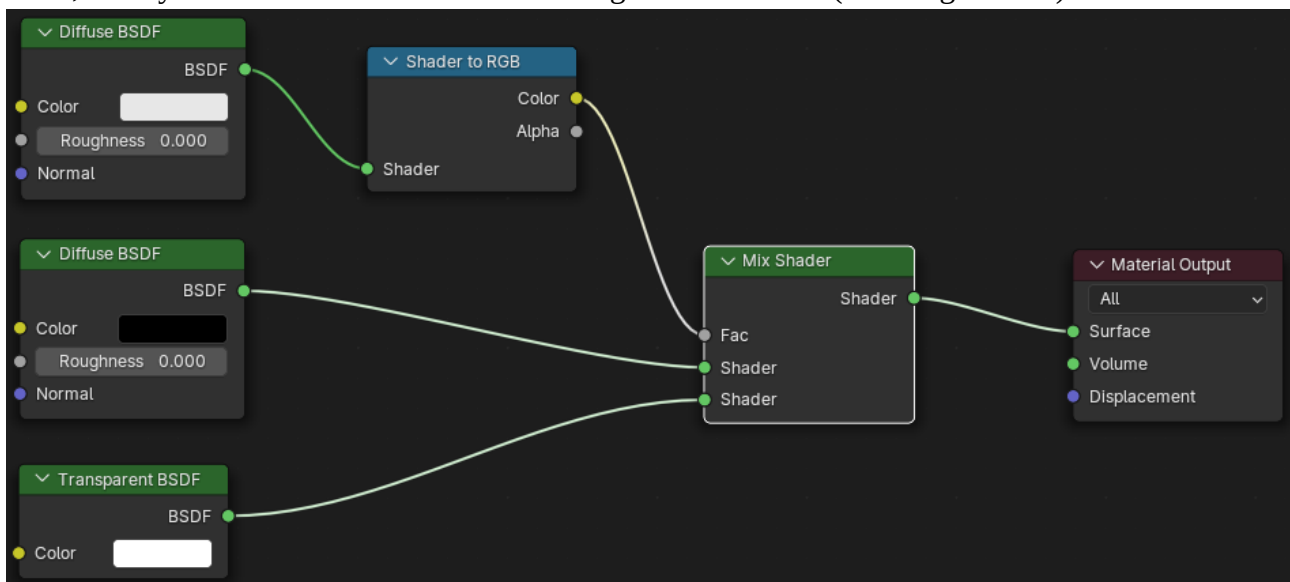
## Render Engine



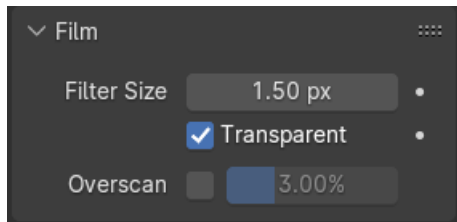
The rendering engine is set to Cycles (instead of EEVEE). This significantly [reduces rendering time](#) compared to Cycles. However, it also brings its own challenges for the „**Plane**“.

## Plane

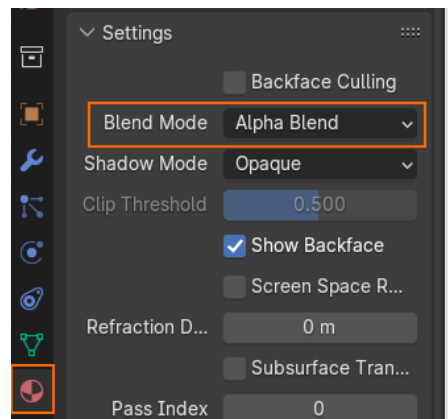
The object „**Plane**“ is used to capture the shadow cast by sunlight on an object. To make everything work, the layer has its own material and shading nodes within it (see image below).



Furthermore the render setting in section „**Film**“ is also set as follows:



Another very important setting is the “**Blend Mode**” in the “**Settings**” section, which is selected as “**Alpha Blend**”.



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](#).

## Demo files 4.0

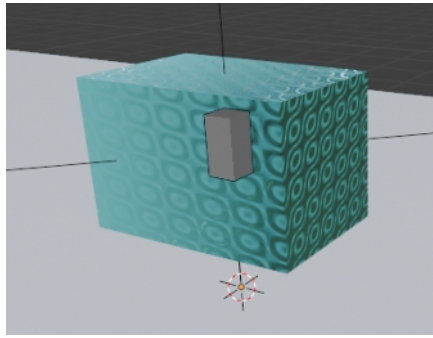
Found in „IE-AutoSpriter-\demos\blend4.0\

- **!\_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.

## Shape

The model is a simple cube (or cuboid, perhaps more accurately)

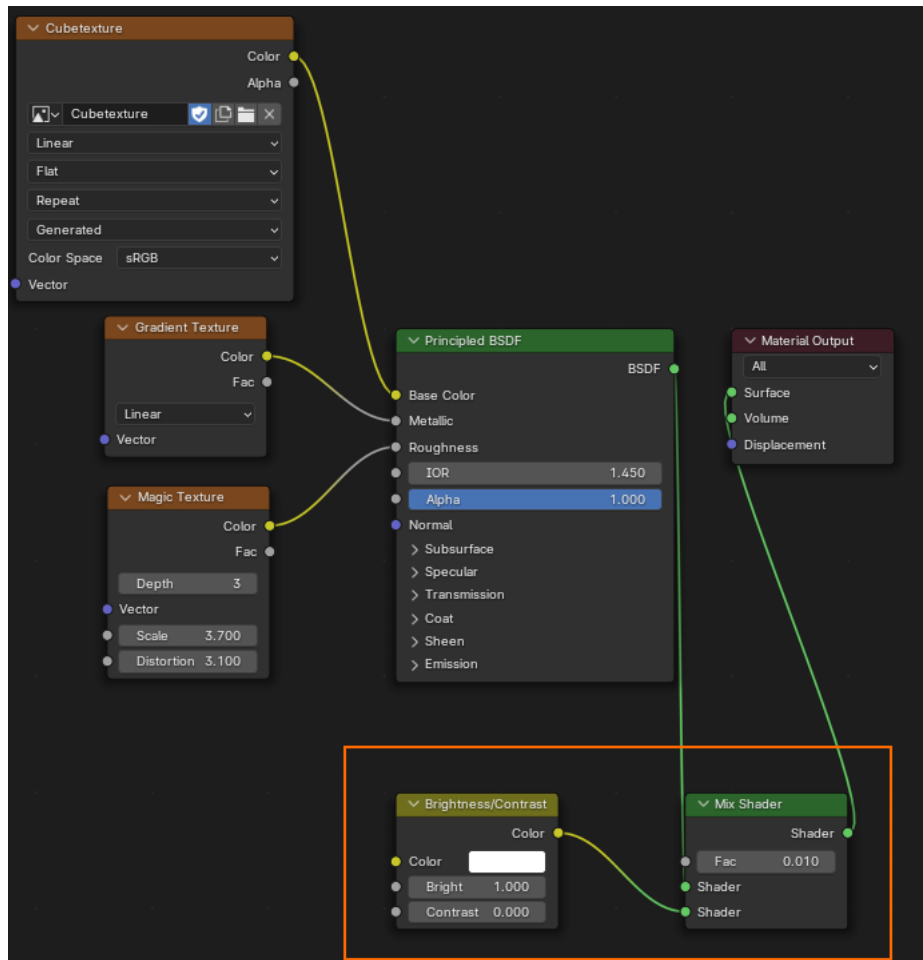


This doesn't mean the small one within the large one, but the large cuboid.

## Shader Nodes

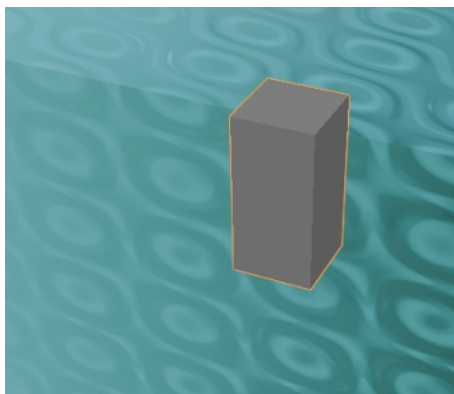
Here, too, the following shading nodes are activated (see next image below):

- Manually added "**Cube Texture**": This is a renamed "**Image Texture**" node and defines the color turquoise.
- "**Gradient Texture**": Creates the dark-to-light gradient.
- "**Magic Texture**": Creates the eye pattern on the surface.
- The "**Principled BSDF**" and "**Material Output**" nodes are created automatically when a new material is created for the cube object.
- The "**Brightness/Contrast**" and "**Mix Shader**" nodes are created connected in the optional step 2 of IE AutoSpriter. They contain selfdefined default values, which can also be seen in the next image below. It is used to prevent the model from containing the color black (rgb(0,0,0)) because this special black is used for transparency in the animation by the Infiniyt Engine



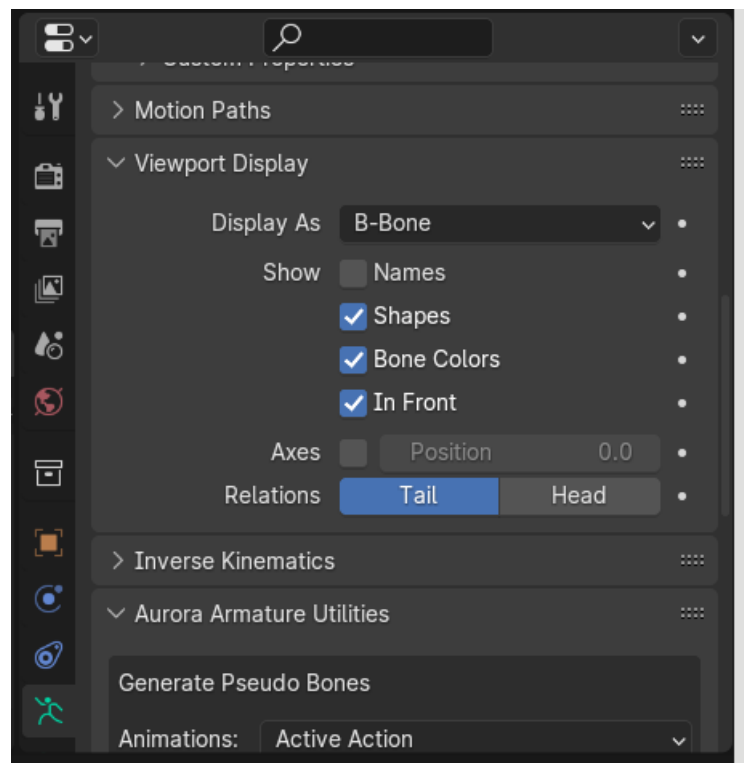
## Bones

The Armature contains only one bone(see image below). It is the the grey cuboid.



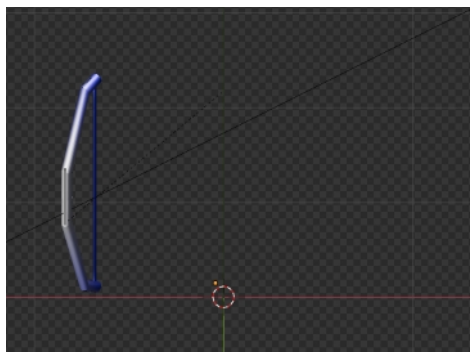
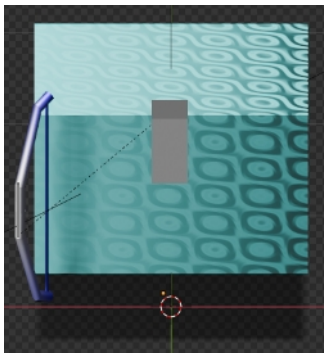
Checking the „In Front“checkbox sets it to always visible. And the shape is "**unusual**" because it is a B-bone (see image below).





## Scene Collections

Contains models for the specific weapons. Here is an example „bow“:



The left image shows a bow with a creature, and the right image shows only the weapon. The weapons have their own bone(s) but no animation other than following the creature's movements.

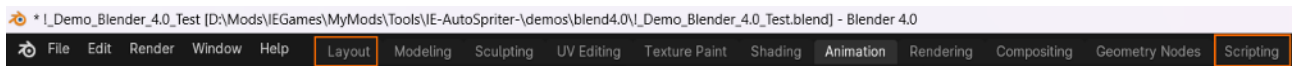
## Troubleshooting/FAQ

### RuntimeError - context is incorrect

If you are experiencing an error very similar to the following after pressing „RENDER“:

```
Python: Traceback (most recent call last):
  File "C:\Users\<username>\AppData\Roaming\Blender Foundation\Blender\4.0\
scripts\addons\ie_autospriter.py", line 246, in execute
    bpy.ops.object.select_all(action='DESELECT')
  File "<Drive letter>:\Blender 4.0\4.0\scripts\modules\bpy\ops.py", line 109,
in __call__
    ret = _op_call(self.idname_py(), kw)
RuntimeError: Operator bpy.ops.object.select_all.poll() failed, context is
incorrect
```

This can then be resolved by switching to the **Layout** or **Script** window. At this point, I'm not entirely sure why the **Animation** window doesn't accept pressing the Render button in step 5, but **it seems to be related to the user's window context**.



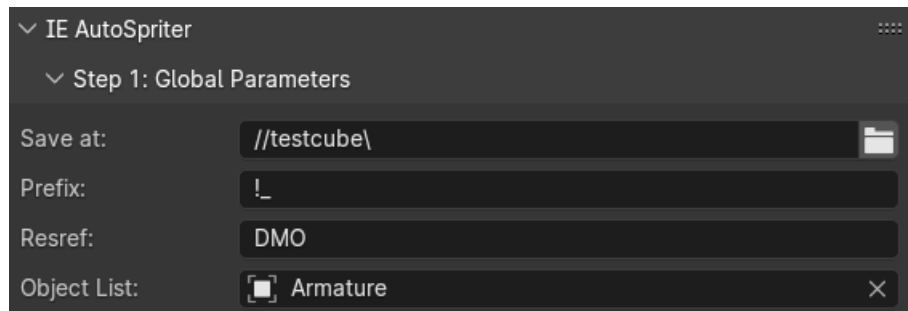
### OSError – incorrect directory

This is actually fixed and shouldn't happen anymore. Please download the latest version!

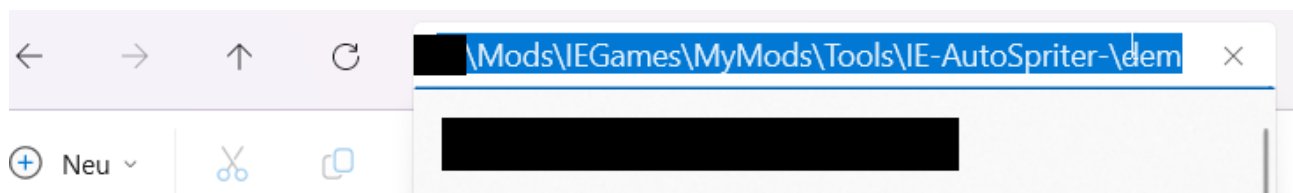
If you are experiencing an error very similar to the following after pressing „RENDER“:

```
Python: Traceback (most recent call last):
File "<Drive letter>:\Blender 4.0\4.0\python\lib\os.py", line 215, in makedirs
    makedirs(head, exist_ok=exist_ok)
File "<Drive letter>:\Blender 4.0\4.0\python\lib\os.py", line 215, in makedirs
    makedirs(head, exist_ok=exist_ok)
File "<Drive letter>:\Blender 4.0\4.0\python\lib\os.py", line 225, in makedirs
    mkdir(name, mode)
OSError: [WinError 123] Die Syntax für den Dateinamen, Verzeichnisnamen oder die
Datenträgerbezeichnung ist falsch: '\\\\'
```

This may be due to the incorrect "Save at" path (see the next image below). This happens especially when the default directory is selected, which is available after pressing the folder icon.



This can be prevented by manually navigating from the local drive to the desired folder one by one, by typing the path manually or, which is quickest, by copying the path from your folder explorer and pasting it into IE AutoSpriters (see next image below).



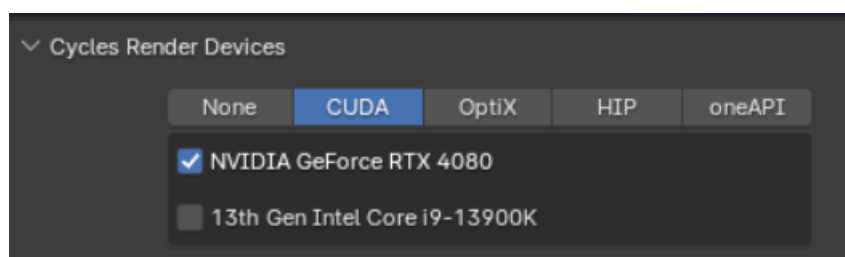
## „Save at“ path empty, what happens?

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?

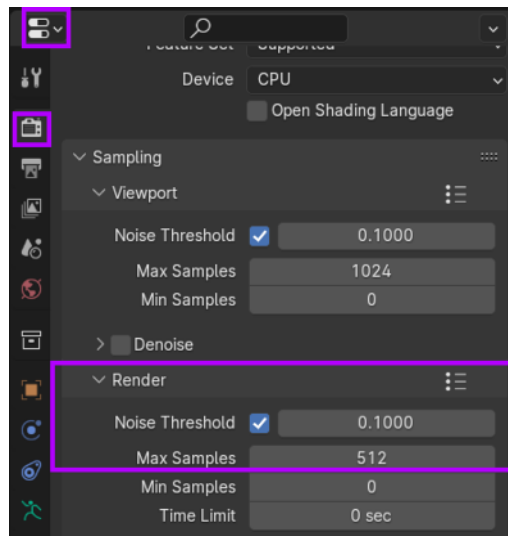
Testresults can be found [here](#).

## 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.

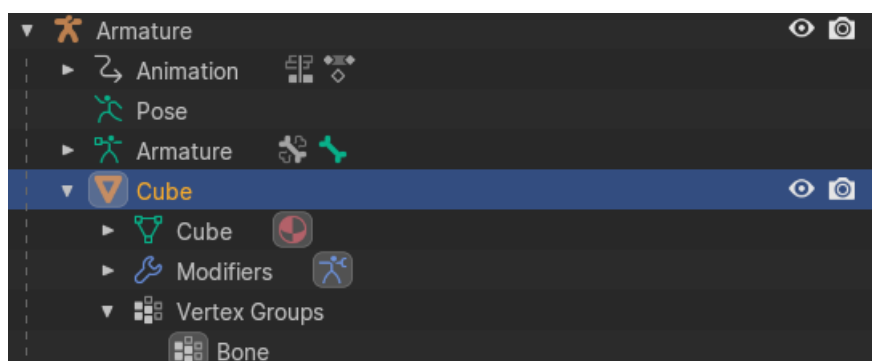
Further reducing the "**Max. Samples**" (e.g., to 256) should certainly be possible without any significant loss in sprite quality, but increasing the "**Noise Threshold**" may not be as effective. Users will have to test for themselves what works best for them.

## Use Decimate Modifier

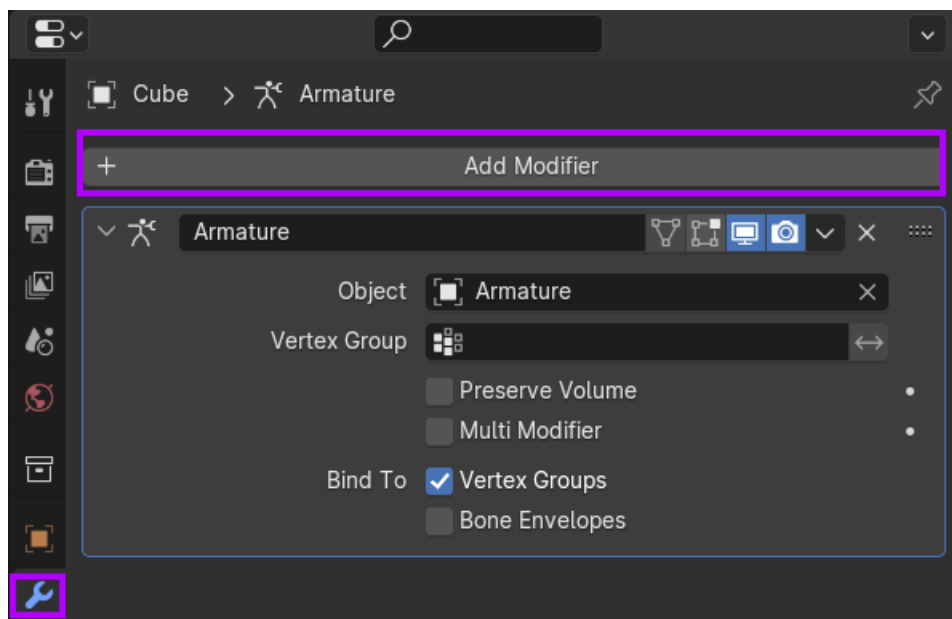
**This should only be considered as a last resort** as it will definitely reduce model quality.

[Decimate Modifier](#) can be used as follows:

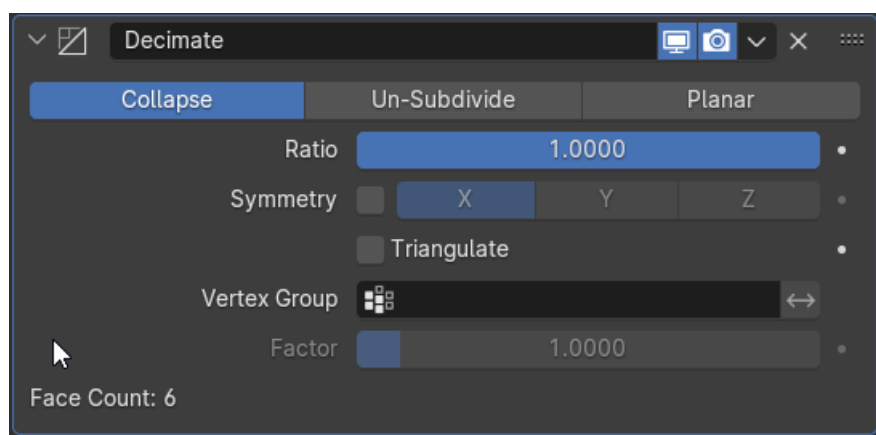
1. Choose your model in this particular case cube



2. Press „**Add Modifier**“ and select **Generate → Decimate**



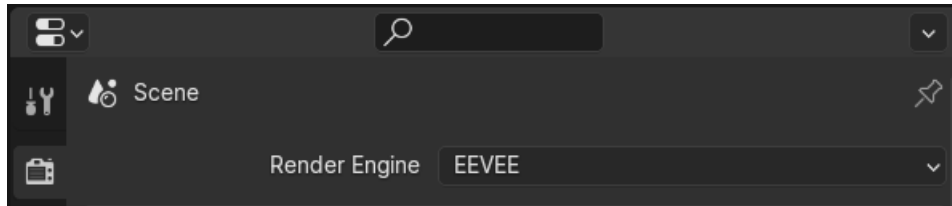
3. **Reducing Ratio** will improve performance time([see testexample](#))



## Change Render Engine

A blend save file and demo regarding EEVEE is available with the latest version.

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](#)

## Incorrect creature Z rotation in Blender after rendering

This should be fixed with the latest version. However, since the value that resets the creature's Z-rotation is a floating-point value, there's a chance the position might still not be exactly the same as before rendering.

This may be due to [floating-point precision](#). The easiest solution is to reset by hand in this case. The value could be hard-coded, but this is not planned at this stage of development (and may result in other issues).

