

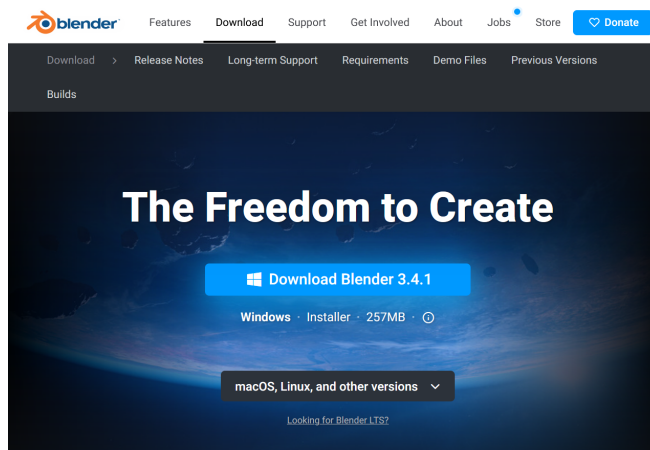
TurboNeRF

Alpha Install Guide

Document Created 2023-03-12 Last Updated: 2023-03-12 08:33 PM (UTC -4)

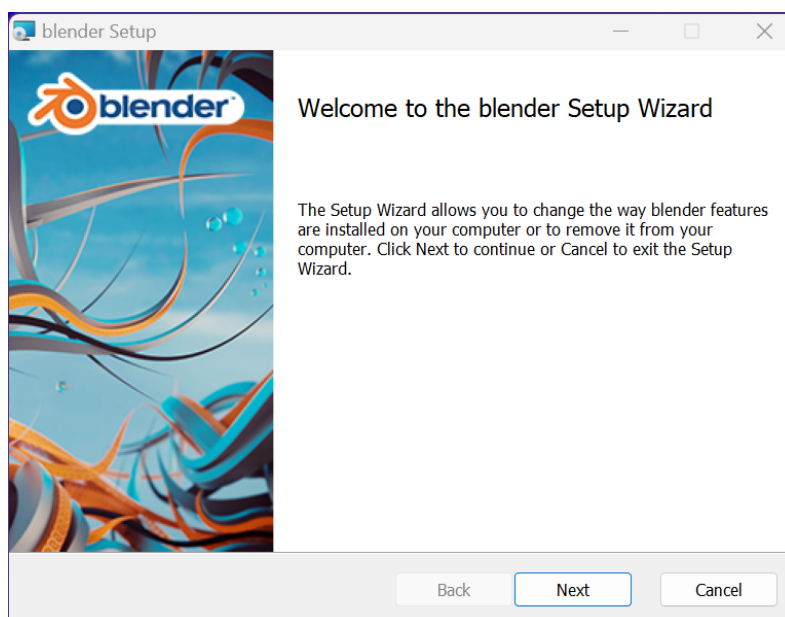
1. Download the current Blender 3.x release for Windows x64:

<https://www.blender.org/download/>



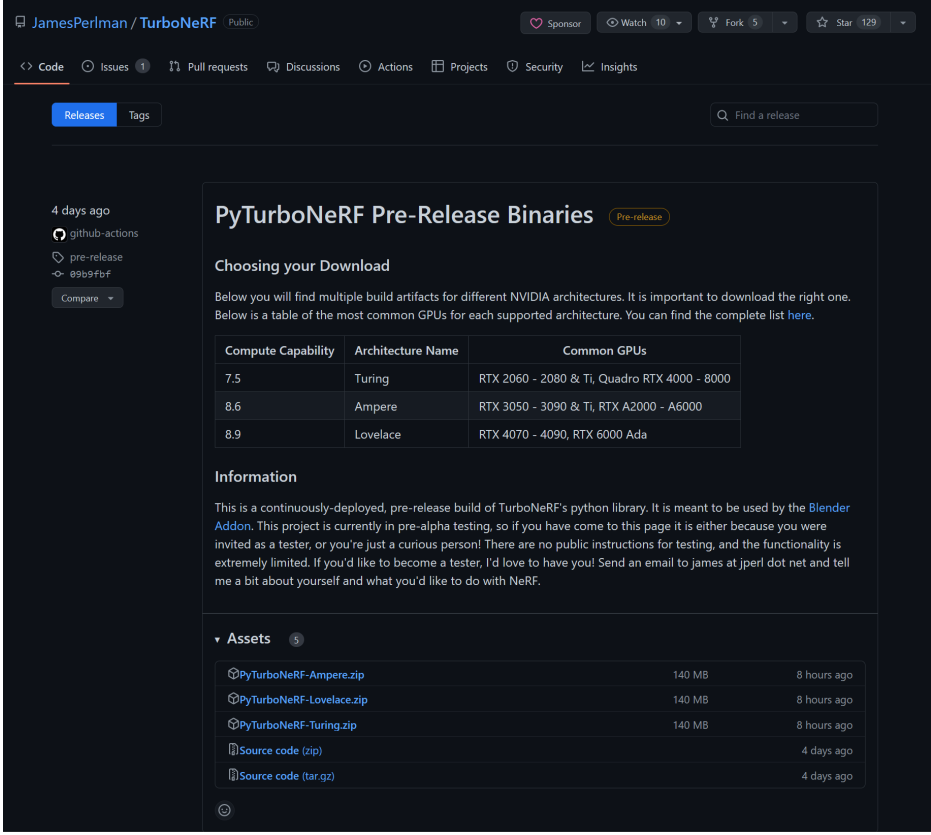
To install the Blender 3D software onto your PC, you need to run the installer program named "blender-<version>-windows-x64.msi".

If you are just getting started with the Blender software for the first time, you can stick with the default settings when you run the Blender setup program.



2. Download the TurboNeRF render engine from the project's GitHub releases page:

<https://github.com/JamesPerlman/TurboNeRF/releases>



The screenshot shows the GitHub releases page for the repository JamesPerlman/TurboNeRF. The page is titled "PyTurboNeRF Pre-Release Binaries" and includes a "Pre-release" badge. It features a section for "Choosing your Download" with a table of GPU architectures and a list of assets.

Compute Capability	Architecture Name	Common GPUs
7.5	Turing	RTX 2060 - 2080 & Ti, Quadro RTX 4000 - 8000
8.6	Ampere	RTX 3050 - 3090 & Ti, RTX A2000 - A6000
8.9	Lovelace	RTX 4070 - 4090, RTX 6000 Ada

The assets section lists the following files:

- PyTurboNeRF-Ampere.zip (140 MB, 8 hours ago)
- PyTurboNeRF-Lovelace.zip (140 MB, 8 hours ago)
- PyTurboNeRF-Turing.zip (140 MB, 8 hours ago)
- Source code (zip) (4 days ago)
- Source code (tar.gz) (4 days ago)

Your NVIDIA GPU's chip "architecture" will define the specific file you need to download from the releases page. Look at the GPU entries listed below to see if you require a "Turing", "Ampere", or "Lovelace" architecture based PyTurboNeRF release.

GPU: RTX 2060 - 2080 & Ti, Quadro RTX 4000 - 8000

Asset: PyTurboNeRF-Turing.zip

GPU: RTX 3050 - 3090 & Ti, RTX A2000 - A6000

Asset: PyTurboNeRF-Ampere.zip

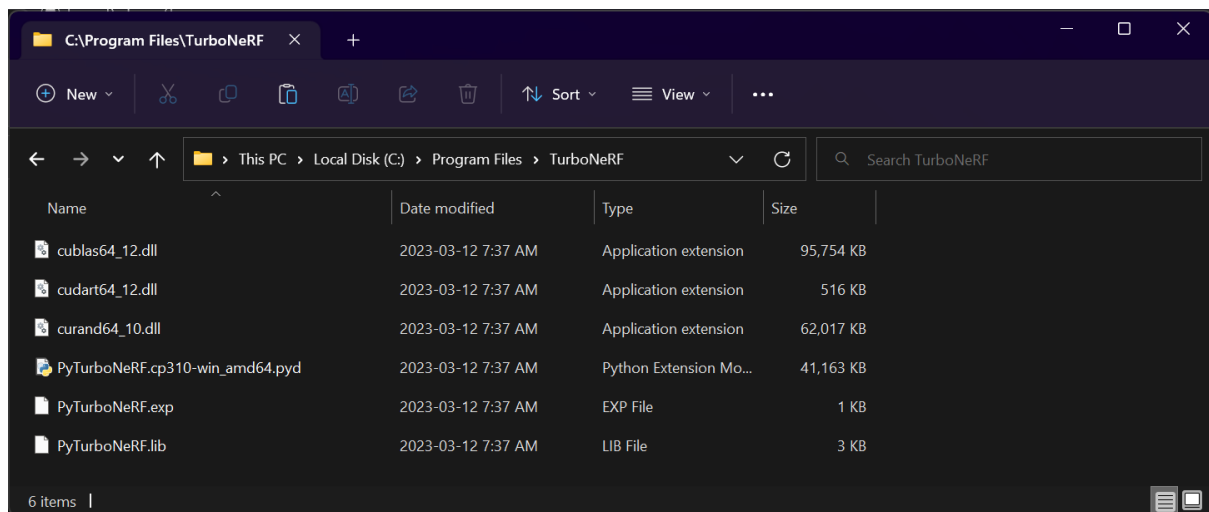
GPU: RTX 4070 - 4090, RTX 6000 Ada

Asset: PyTurboNeRF-Lovelace.zip

Note: You might need to click to expand the "Assets" section on the GitHub releases page if the files are not immediately visible.

3. Create a new "C:\Program Files\TurboNeRF\" folder on your hard disk.

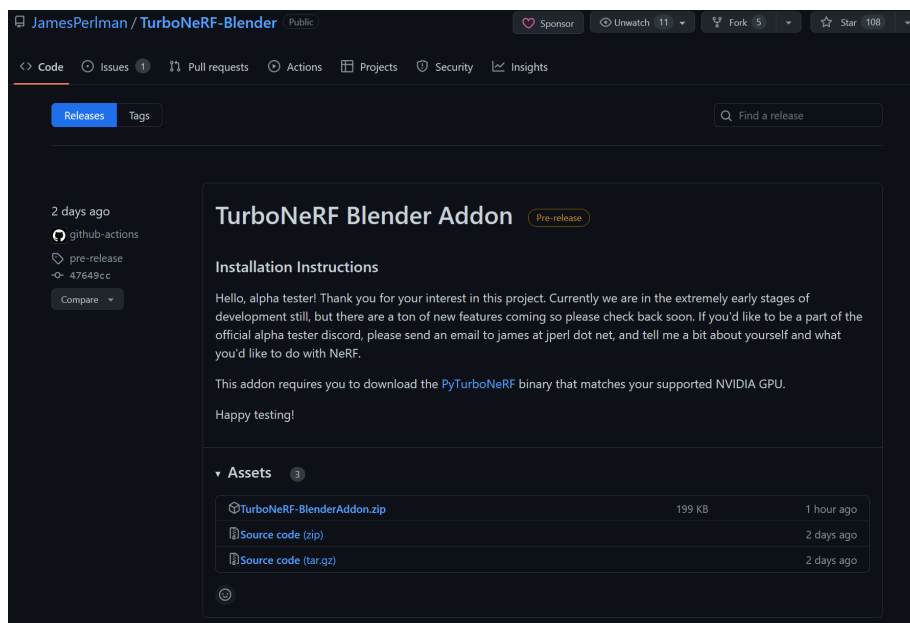
Extract the contents of the "PyTurboNeRF-<release>.zip" file that was downloaded from GitHub. Move those extracted files into the newly created TurboNeRF folder. These libraries are used to power the interactive NeRF render engine.



4. Download the TurboNeRF Blender Addon file named "TurboNeRF-BlenderAddon.zip" from the project's GitHub releases page:

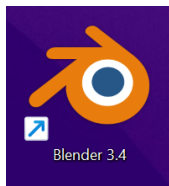
<https://github.com/JamesPerlman/TurboNeRF-Blender/releases>

Note: You might need to click to expand the "Assets" section on the GitHub releases page if the files are not immediately visible.

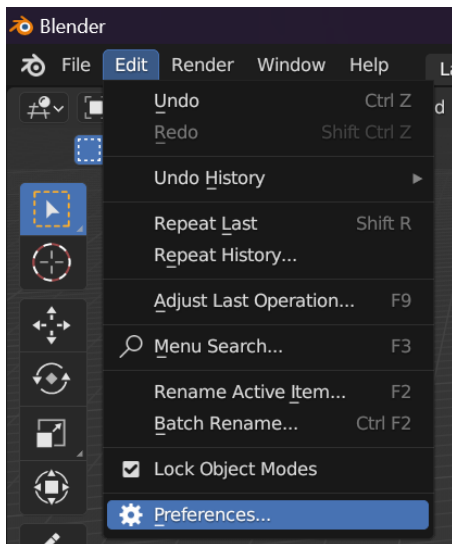


Note: Blender is able to read the addon contents directly from a zip archive. This means you do not have to extract the "TurboNeRF-BlenderAddon.zip" file.

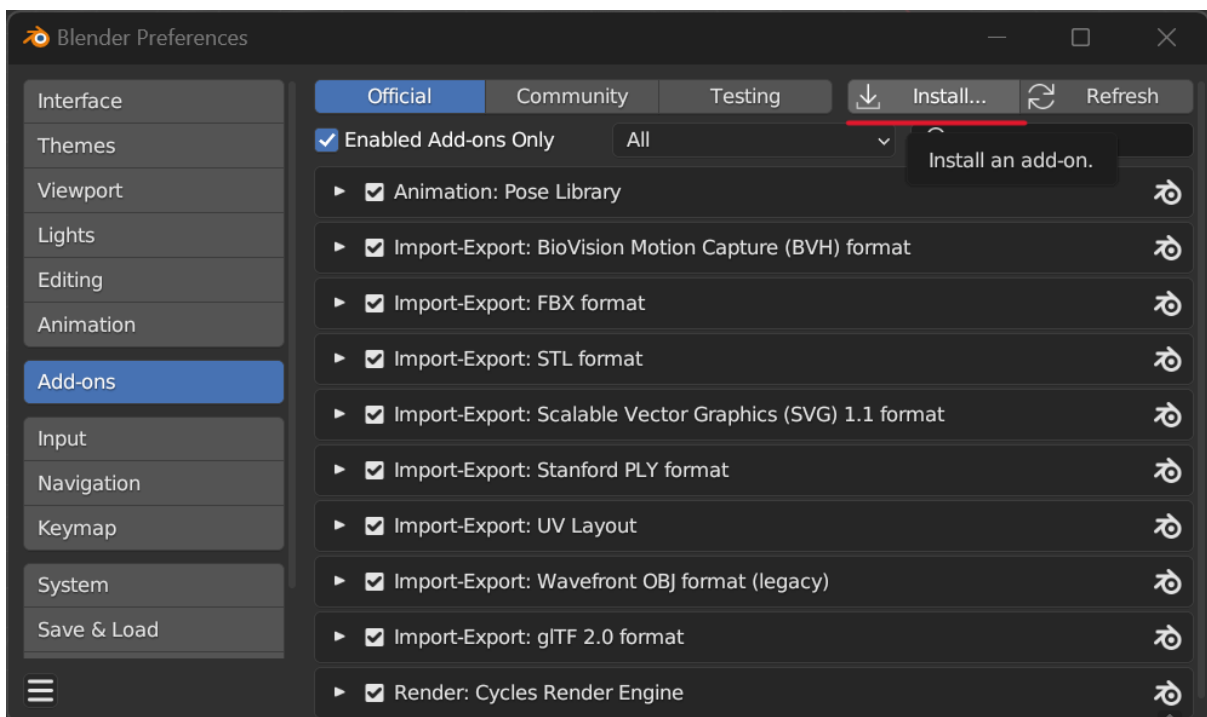
5. Start Blender using the new shortcut on your desktop.



After Blender has finished loading, open the "**Edit > Preferences...**" menu item.

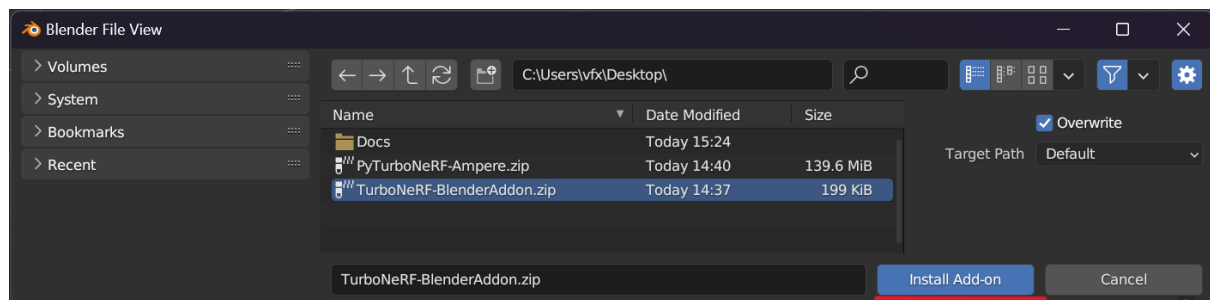


Click on the "Addons" tab on the left side of the Preferences window. This section of the preferences allows us to install and update custom addon packages.

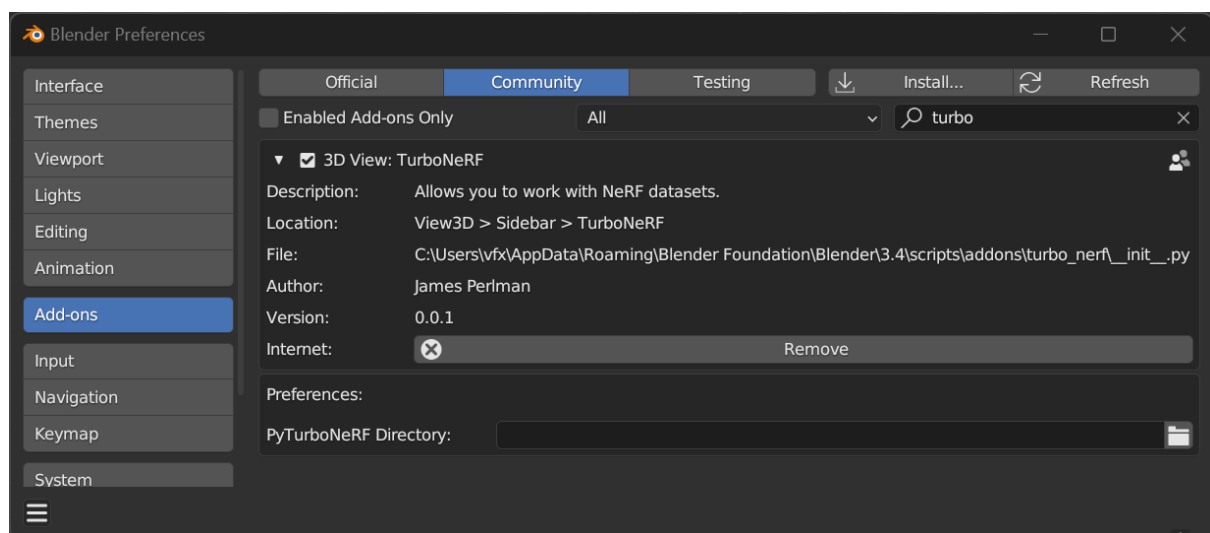


We are going to install the TurboNeRF Blender Addon using this window. Click the "Install...." button at the top right of the preferences window.

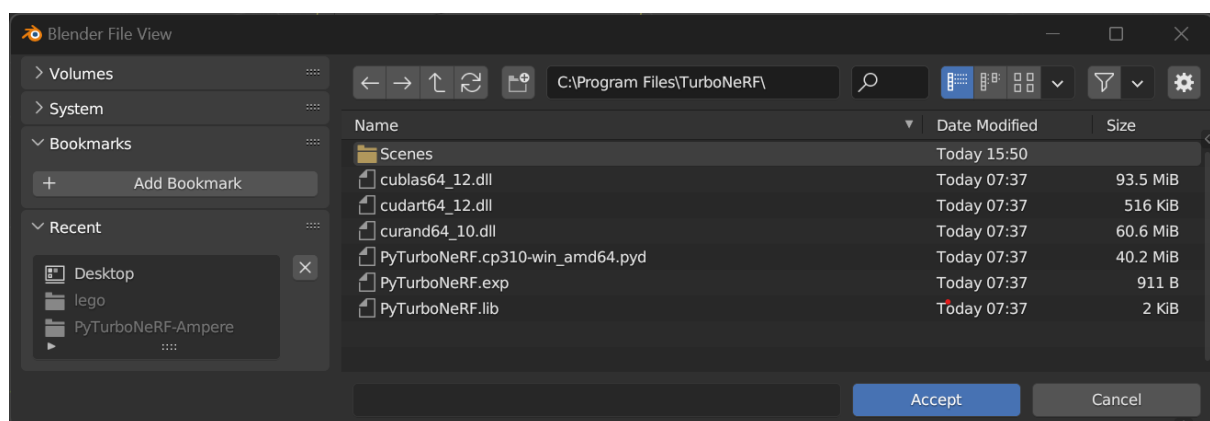
A Blender File View dialog will appear. Select the "TurboNeRF-BlenderAddon.zip" that was recently downloaded. Then press the "Install Addon" button.



In the Blender Preferences window you need to enable the small checkbox, next to the "3D View: TurboNeRF" addon, to enable the new package.



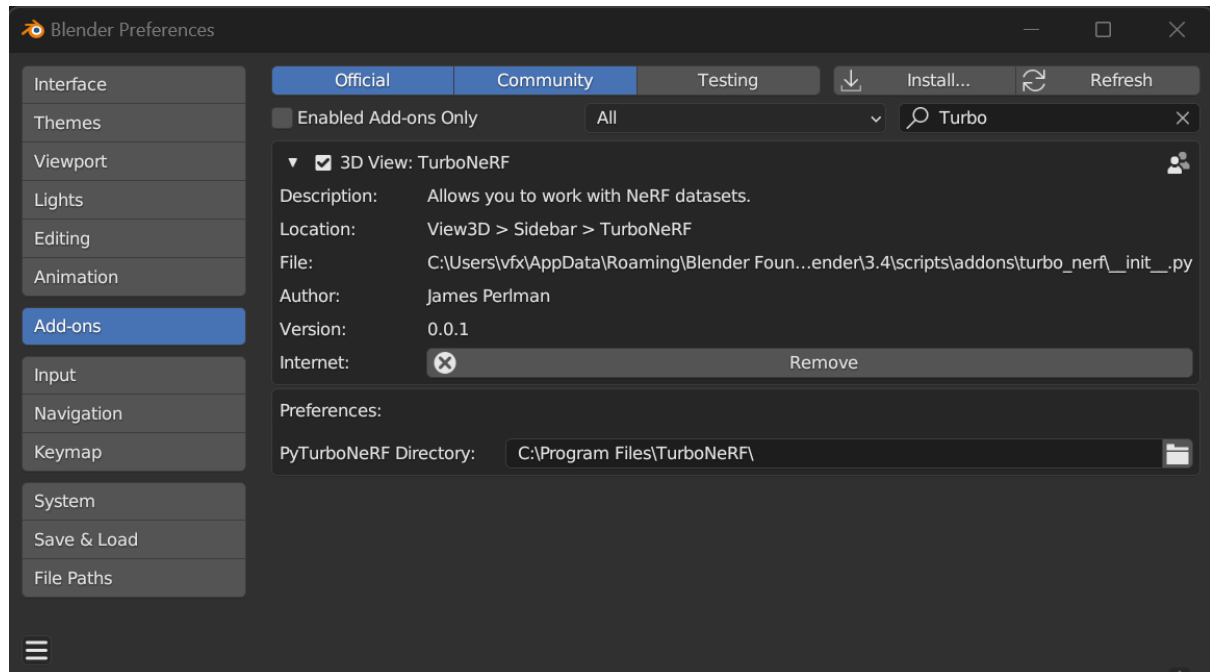
The "PyTurboNeRF Directory" text field is used to specify the location on disk where the PyTurboNeRF render engine files are located. Click on the folder browsing icon to the right of this text field to display a Blender File View dialog.



Point the "PyTurboNeRF Directory" at the following file path:
C:\Program Files\TurboNeRF\

Click the "Accept" button to close the Blender File View dialog.

At this point, the Blender Preferences dialog should look like this:



Close the Blender Preferences window.

Restart Blender once to initialize the TurboNeRF toolset.

Rendering Our First NeRF Scene

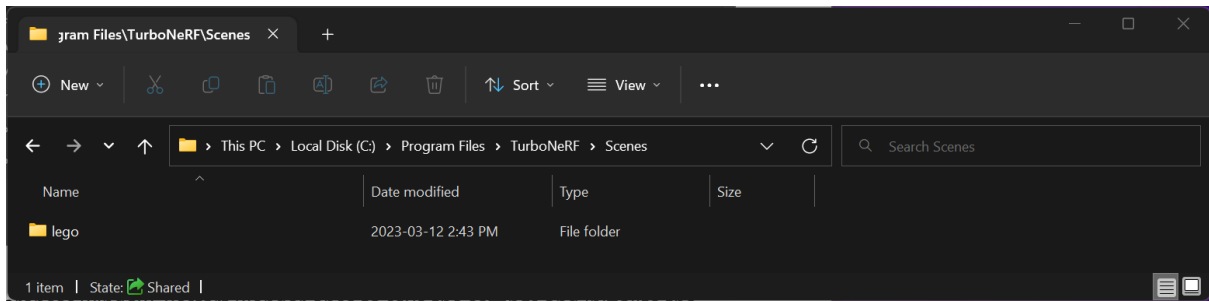
With the TurboNeRF toolset installed, we can now get down to business and start rendering neural radiance field content!

Let's start by downloading the example NeRF scene "Lego.zip" that is linked in the TurboNeRF [readme.md](#) file:
<https://www.dropbox.com/sh/qkt4t1tk1o7pdc6/AAD218LLtAavRZykYl33mO8ia?dl=1>

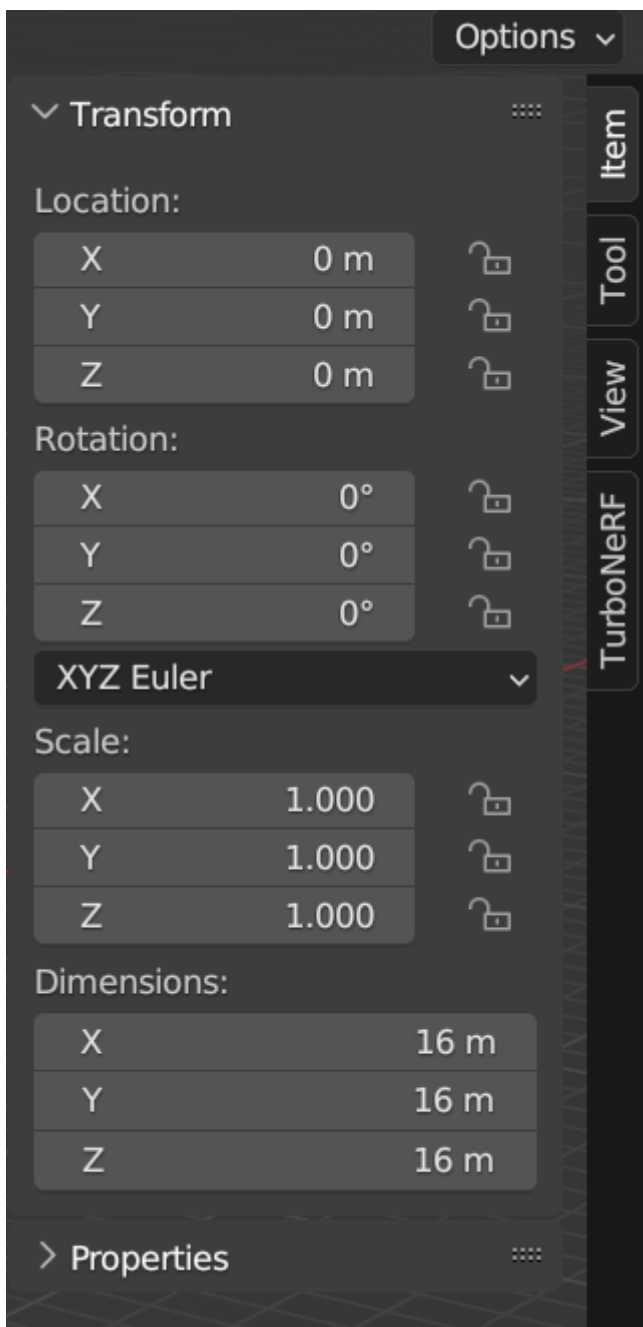
To keep the example content organized, we might find it handy to create a custom TurboNeRF scenes folder named:

C:\Program Files\TurboNeRF\Scenes\

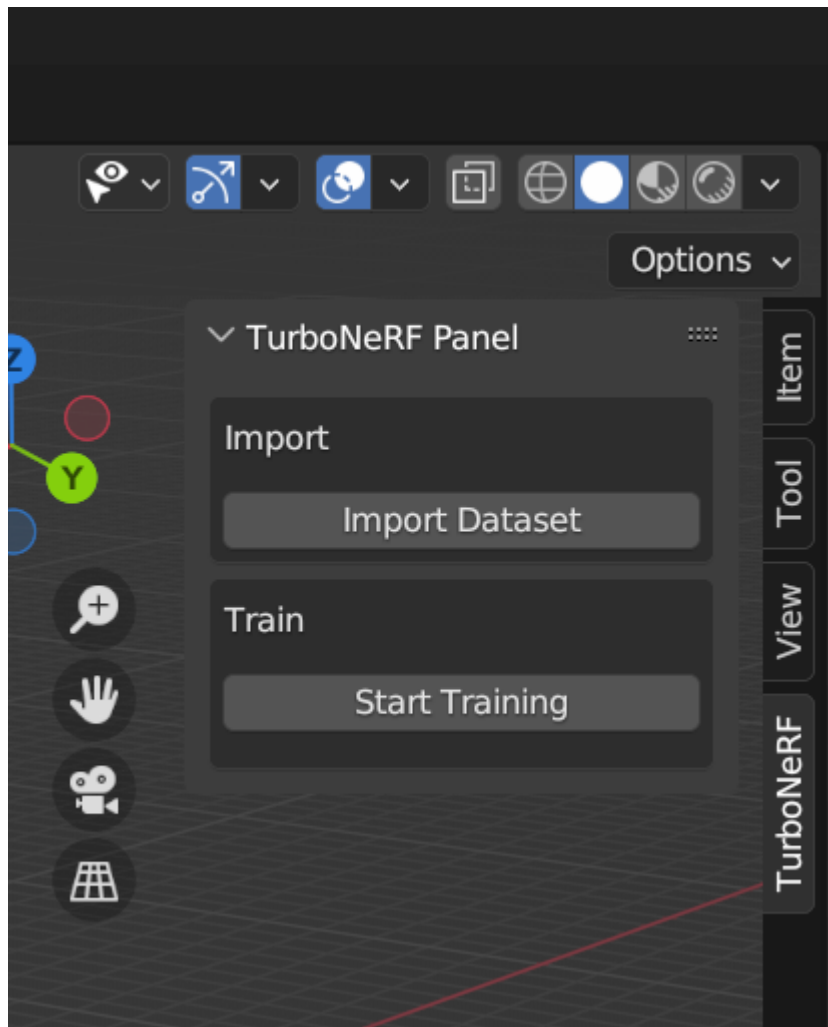
Place the expanded "lego.zip" archive content in the "Scenes" folder.



With Blender open there should be a new "TurboNeRF" panel item on the right edge of the 3D viewport window. Click on the tab labelled "TurboNeRF".



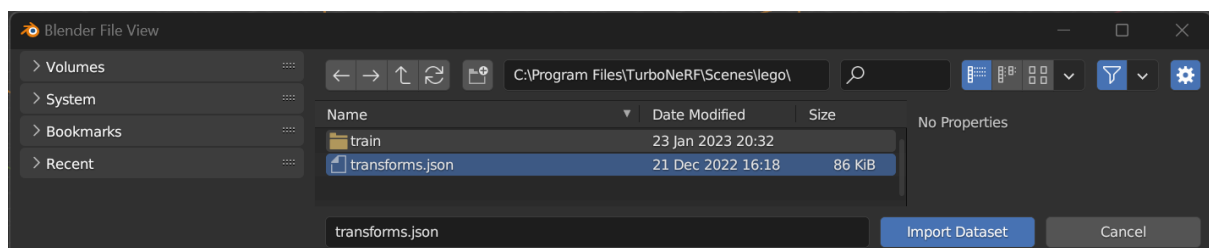
The TurboNeRF panel should be visible. Click the "Import Dataset" button to load in a NeRF scene.



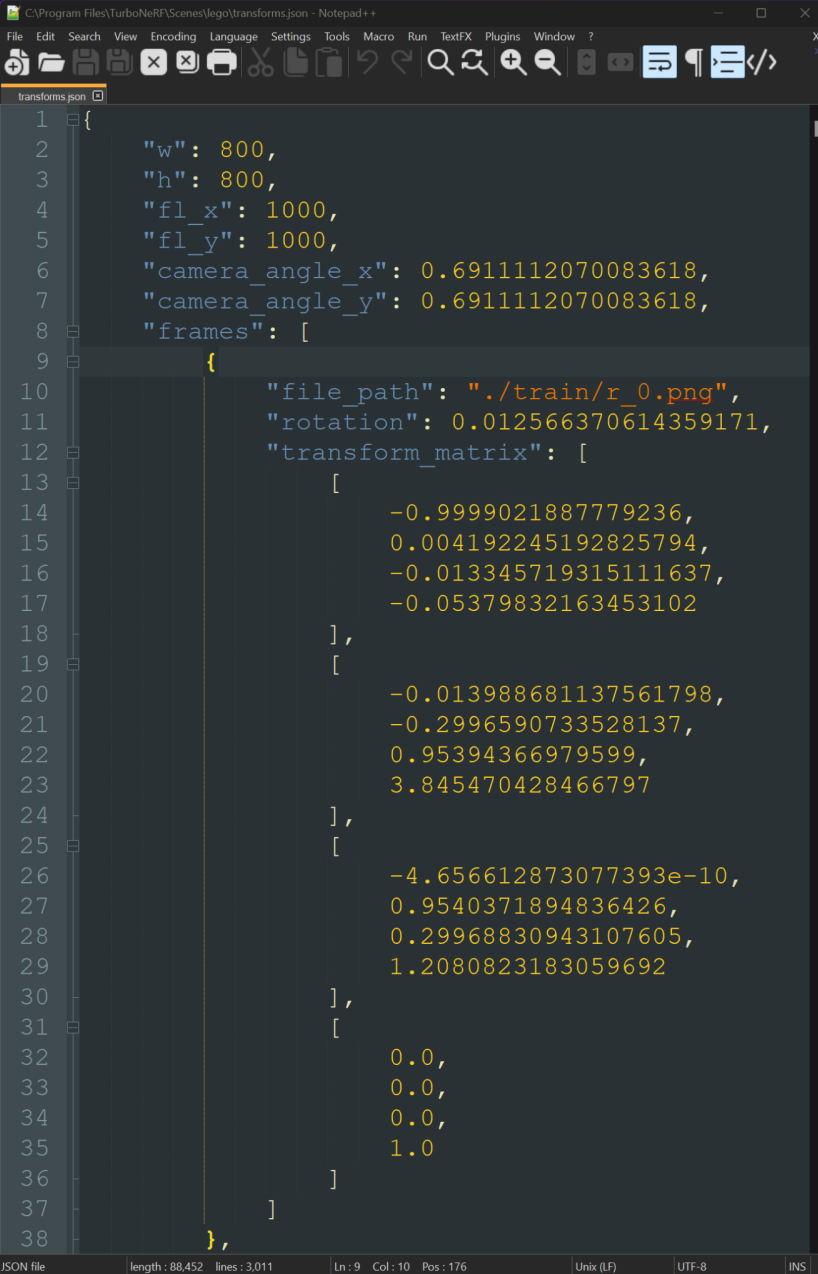
At this point we need to import a "transform.json" file that was created by a camera registration toolset like [COLMAP](https://colmap.github.io/). This JSON document holds our NeRF centric camera pose data, along with the lens parameters, and the image filenames.

A Blender File View dialog should be visible.

Select the Lego example's "transform.json" file, then click the "Import Dataset" button to continue.



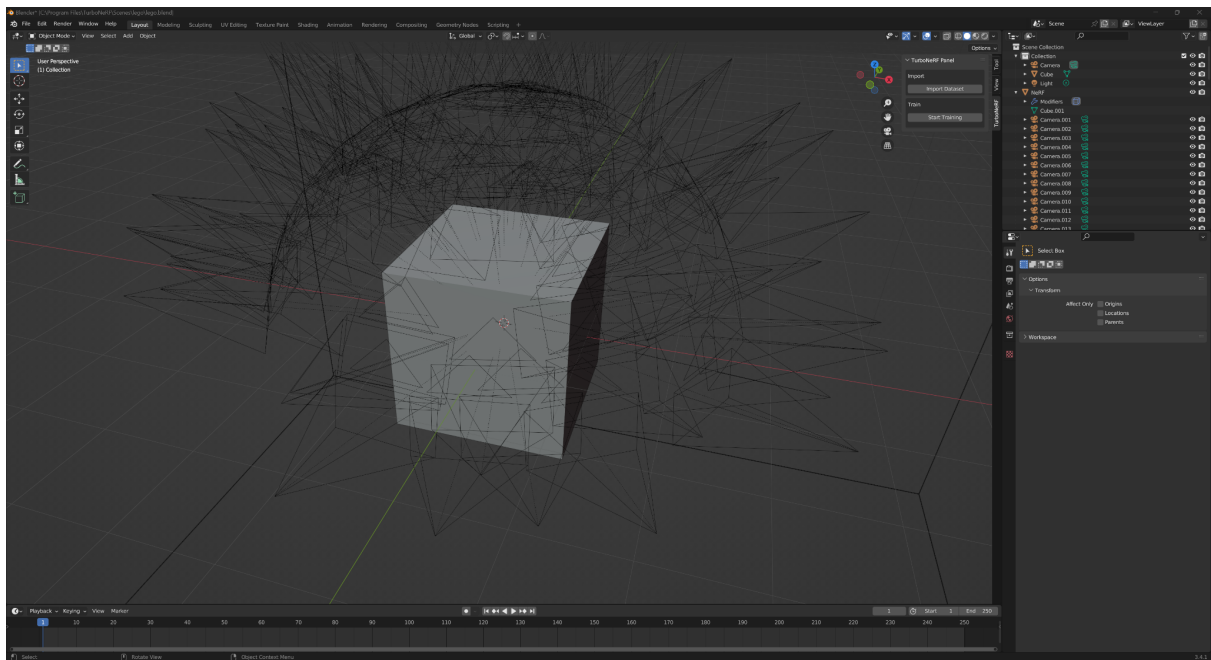
Note: If you open the JSON file in a programmer's text editor like Notepad++ or VSCODE you will see the file has a JSON based scene description structure. The individual images used in the NeRF scene are stored inside an array named "Frames". The position for each camera pose is defined using a 4x4 translation matrix entry named "transform_matrix", and each of the images names are defined using a "file_path" field.



```
1 {
2   "w": 800,
3   "h": 800,
4   "fl_x": 1000,
5   "fl_y": 1000,
6   "camera_angle_x": 0.6911112070083618,
7   "camera_angle_y": 0.6911112070083618,
8   "frames": [
9     {
10      "file_path": "./train/r_0.png",
11      "rotation": 0.012566370614359171,
12      "transform_matrix": [
13        [
14          -0.9999021887779236,
15          0.004192245192825794,
16          -0.013345719315111637,
17          -0.05379832163453102
18        ],
19        [
20          -0.013988681137561798,
21          -0.2996590733528137,
22          0.95394366979599,
23          3.845470428466797
24        ],
25        [
26          -4.656612873077393e-10,
27          0.9540371894836426,
28          0.29968830943107605,
29          1.2080823183059692
30        ],
31        [
32          0.0,
33          0.0,
34          0.0,
35          1.0
36        ]
37      ]
38    },
39  ]
40 }
```

After importing the dataset, a new "NeRF" item is visible in the Scene Collection that is visible at the top right section of the Blender UI.

If you expand this hierarchy you will see the list of the imported cameras. In the perspective view you will also see a large collection of camera locators looking inwards. These cameras were loaded from the transform.json file.

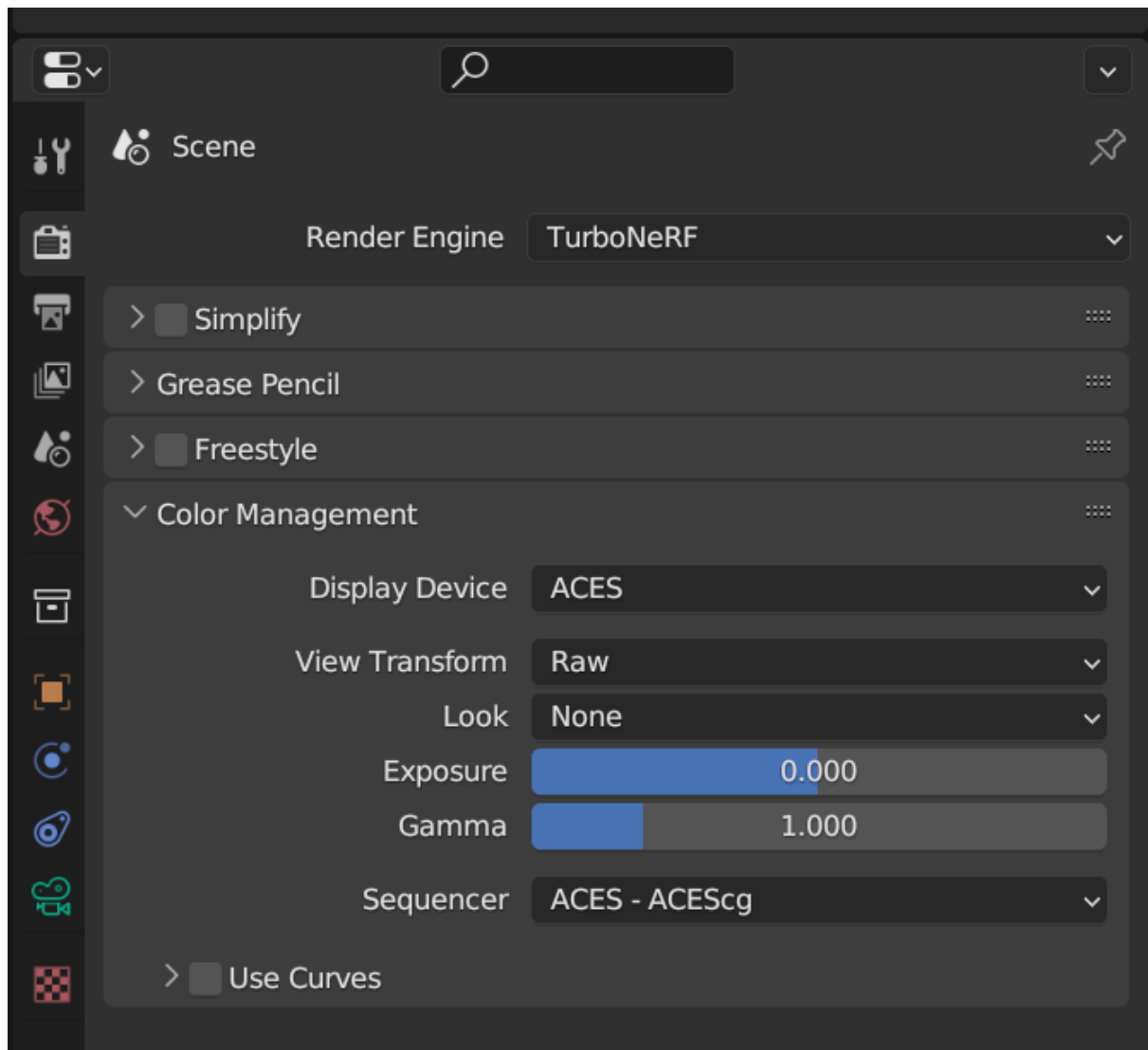


An interesting aspect of NGP (Neural Graphic Primitives) centric NeRF data imports, is the use of a normalized coordinate system with a (0-1) range for scene scale. This means the cameras should be aligned roughly around the size of the default Blender cube.

Let's switch the active renderer for the 3D viewport. Click on the "Camera" icon in the panel view in Blender.

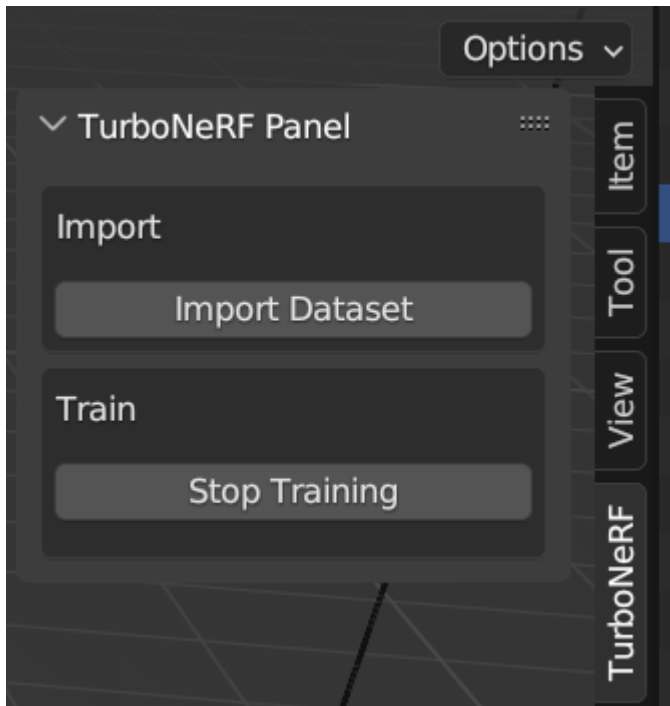
Change the Render Engine from its current value of "Eevee" over to "TurboNeRF".

Expand the Color Management section and switch the View Transform to "Raw". This will use the direct output of the render engine without any additional ACES based color space transforms or tone-mapping effects.

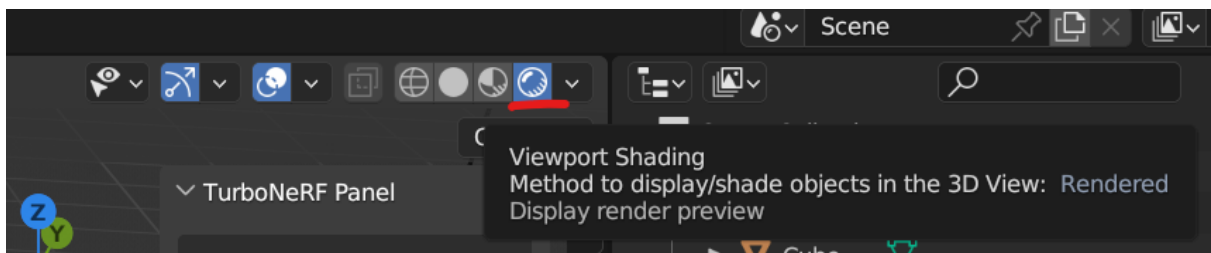


Back in the TurboNeRF panel, click the "Start Training" button. This will begin the machine learning based NeRF model training phase. Your GPU fans will likely spin up at this point to keep the GPU cool as it works its way through a fairly intense compute load.

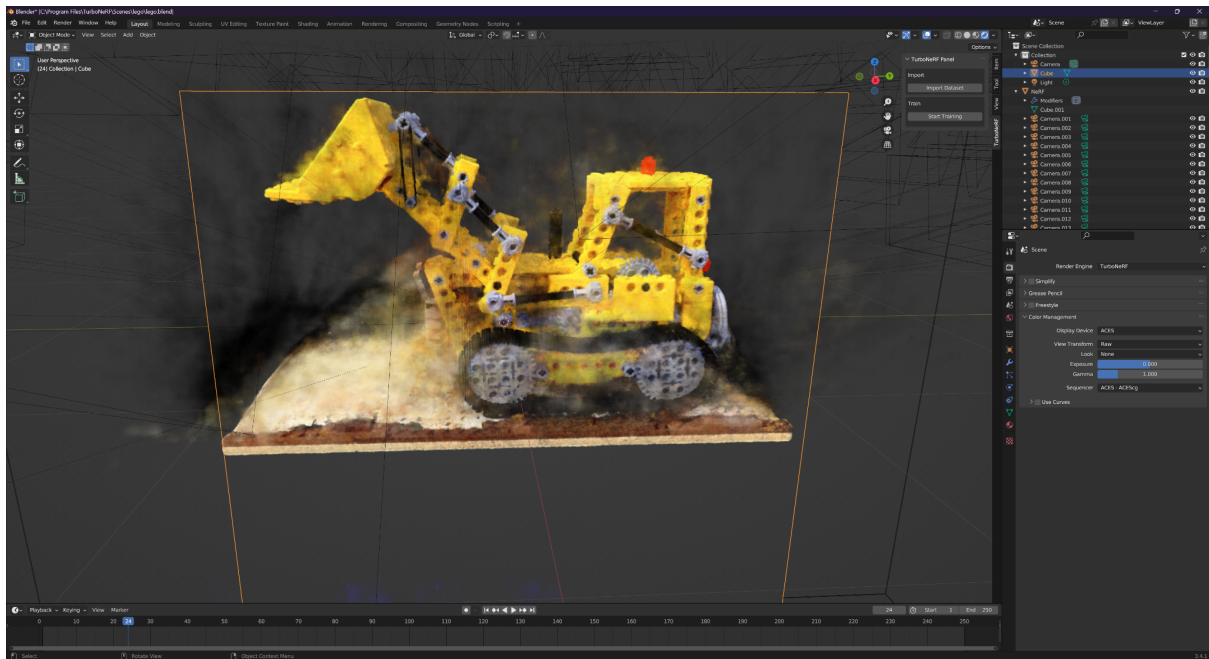
If you need to cancel the training process at any time, click the "Stop Training" button.



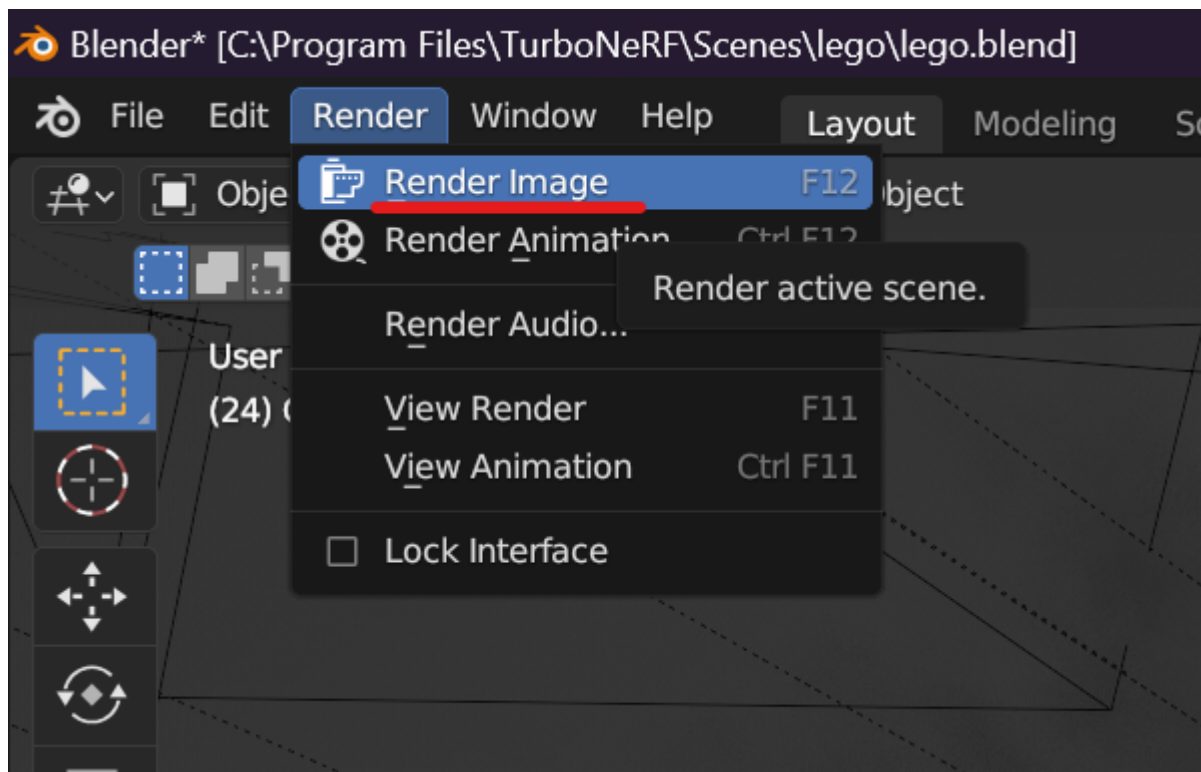
To see the results of the NeRF training and rendering processes, enable the Blender "viewport shading" mode.



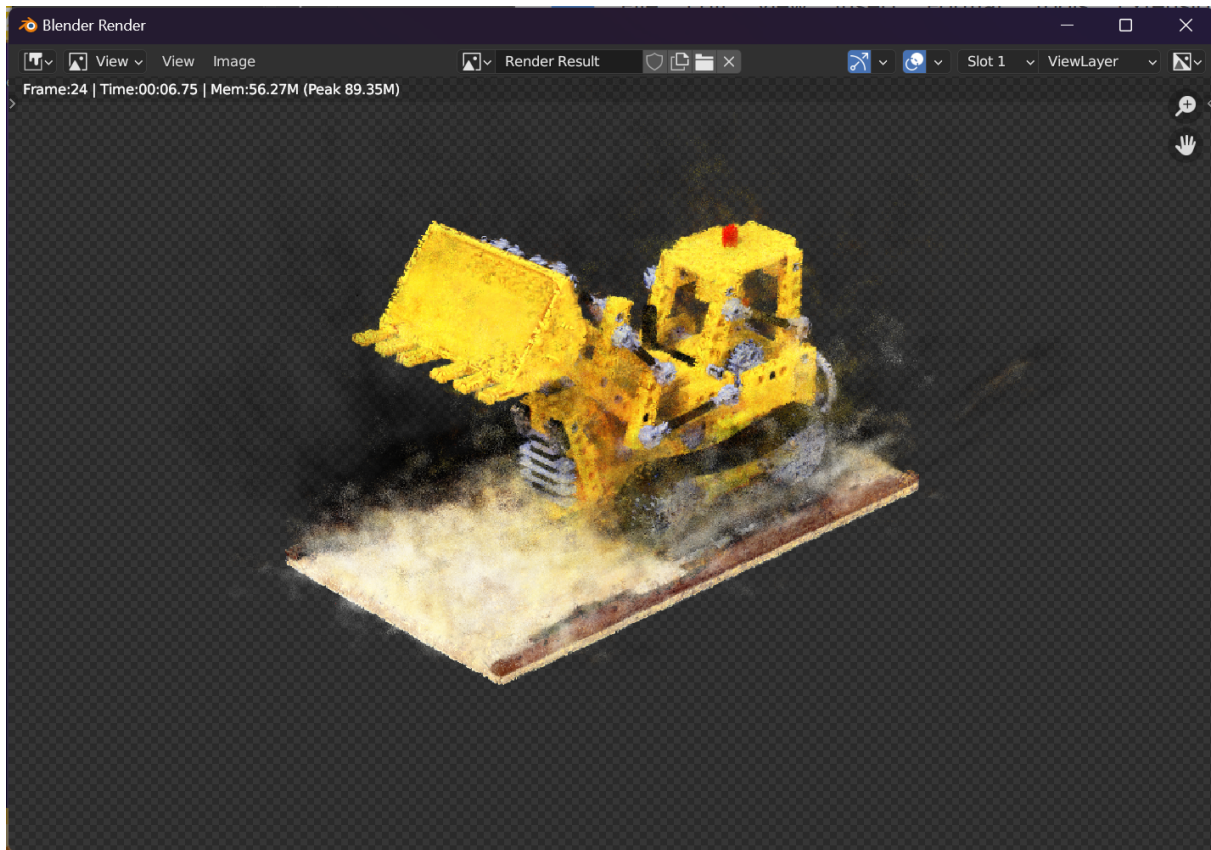
Now you will see an interactive NeRF rendering of the lego model in the Blender viewport window. You can tumble the perspective view to look at the Lego model from any view position.



Blender's offline rendering features can be used to create a higher-quality image output of the current NeRF scene. To create a newly rendered image, select the "Render > Render Image" menu item (or press the F12 hotkey).



The "Blender Render" window will appear. Your GPU fans will likely run at a higher speed during this NeRF based rendering stage. After a few moments a finished image will appear in the window.



Press the "Stop Training" button in the TurboNeRF panel. Then save the current Blender scene to disk as "lego.blend".

Closing Thoughts

At this point you have installed TurboNeRF and rendered your first example scene. Congrats! 🎉

There are lots of additional things you can explore with NeRF workflows, including the process of creating new transform.json files from your own imagery.

Since Blender is a 3D animation toolset, you are able to create your own animated cameras that fly through a NeRF scene.

It is also possible to do "augmented reality" like visual effect scene integration shots that mix NeRF renderings as the background plate footage, with Blender 3D models that are rendered using Eevee or Cycles. You can use Blender's built-in compositing features to combine the imagery, or external compositing tools like Adobe After Effects, Foundry Nuke, or BMD Resolve/Fusion.