

# F3D, Fast and minimalist 3D viewer

Mathieu Westphal and Michael Migliore (F3D-APP) • 01/02/2026



# Who are we?

## F3D-APP FOUNDATION

- A french non-profit dedicated to F3D and open source development
- Funded by F3D sponsors as well as NGI 0 grant (NLnet)
- Fund F3D infrastructure and communication costs



## Mathieu Westphal (Président)

- F3D Maintainer and co-creator
- Software Engineer @ Kitware
- 10y open source experience
- Expertise:
  - Programming
  - Scientific Visualisation
  - Software design

## Michael Migliore (Tresorier)

- F3D Maintainer and co-creator
- R&D Team Leader @ z-emotion
- 16y experience
- Expertise:
  - Programming
  - Graphics Rendering
  - Fabric simulation



# Quick test: raise your hand if...

...you have files of the following formats on your computer:



# Quick test: raise your hand if...

...you have files of the following formats on your computer:

- gITF



# Quick test: raise your hand if...

...you have files of the following formats on your computer:

- glTF
- FBX



# Quick test: raise your hand if...

...you have files of the following formats on your computer:

- glTF
- FBX
- USD



# Quick test: raise your hand if...

...you have files of the following formats on your computer:

- gITF
- FBX
- USD
- STL



# Quick test: raise your hand if...

...you have files of the following formats on your computer:

- glTF
- FBX
- USD
- STL
- OBJ



# Quick test: raise your hand if...

...you have files of the following formats on your computer:

- gITF
- FBX
- USD
- STL
- OBJ
- STEP



# Quick test: raise your hand if...

...you have files of the following formats on your computer:

- glTF
- FBX
- USD
- STL
- OBJ
- STEP
- Alembic



# Quick test: raise your hand if...

...you have files of the following formats on your computer:

- glTF
- FBX
- USD
- STL
- OBJ
- STEP
- Alembic
- OpenVDB



# Quick test: raise your hand if...

...you have files of the following formats on your computer:

- glTF
- FBX
- USD
- STL
- OBJ
- STEP
- Alembic
- OpenVDB
- Gaussian Splating (PLY, spz, splat)



# Quick test: raise your hand if...

...you have files of the following formats on your computer:

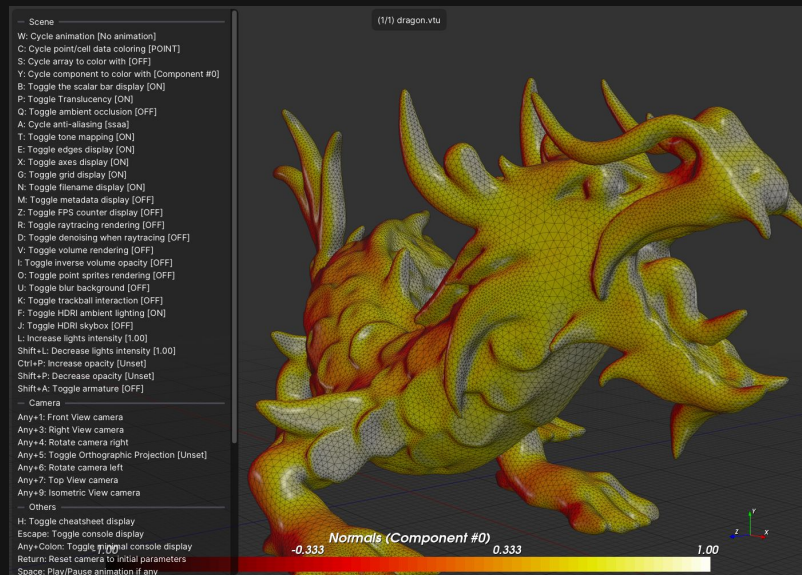
- glTF
- FBX
- USD
- STL
- OBJ
- STEP
- Alembic
- OpenVDB
- Gaussian Splating (PLY, spz, splat)

Then you may want to use F3D!

# Project Focus: F3D

F3D is a fast and minimalist 3D viewer

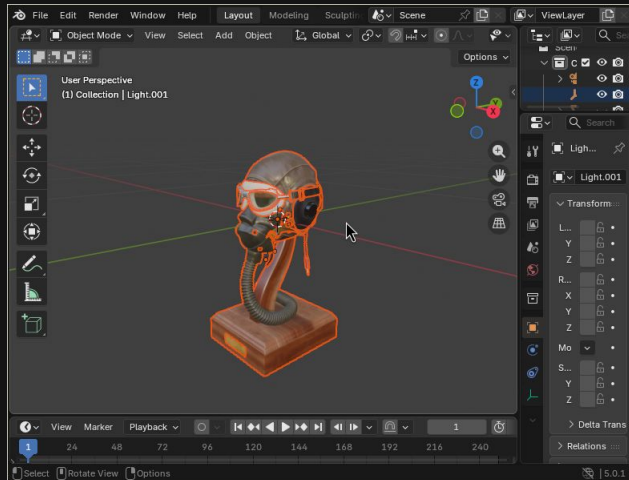
- Github hosted: <https://github.com/f3d-app/f3d>
- Open source (BSD-3-Clause)
- C++ with C, python, Java, javascript bindings
- VTK-based
- Community driven
- Cross-platform and ship binaries
- 3-months release schedule
- Website: <https://f3d.app>
- Available in many package managers



# Scenario 1: I want to quickly visualize a 3D file ?

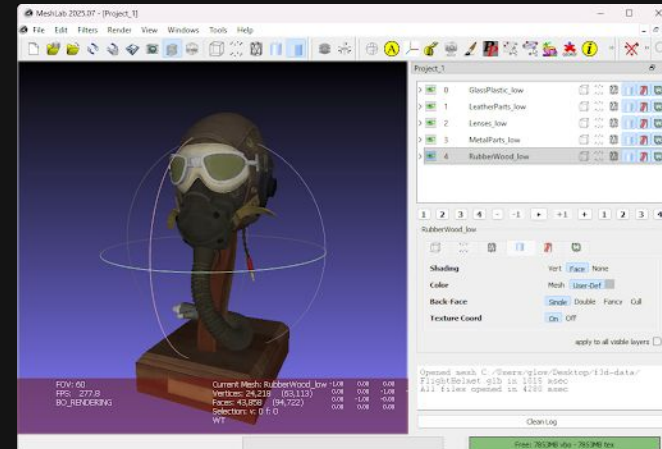
In the FOSS side:

- Go online with <https://3dviewer.net/> ?
- Open blender and import ?
- Open meshlab and import ?



On the closed side:

- Use microsoft 3D viewer ?
- Use a random freeware with ads ?



# Scenario 1: I want to quickly visualize a 3D file ?

Just use F3D!

Command Line:

```
f3d /path/to/supported_file.ext
```

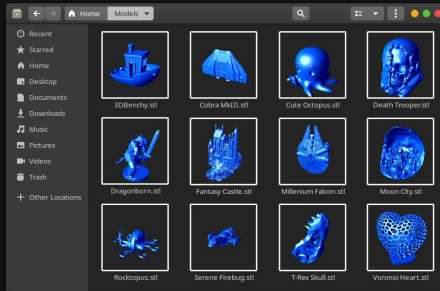
Desktop Environment:

Just double click on any (supported) file!

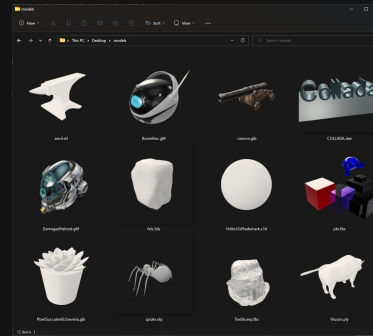


# Scenario 2: I want to generate thumbnails for 3D files ?

- Install stl-thumb? Only STL
- Install space-thumbnails? Unmaintained



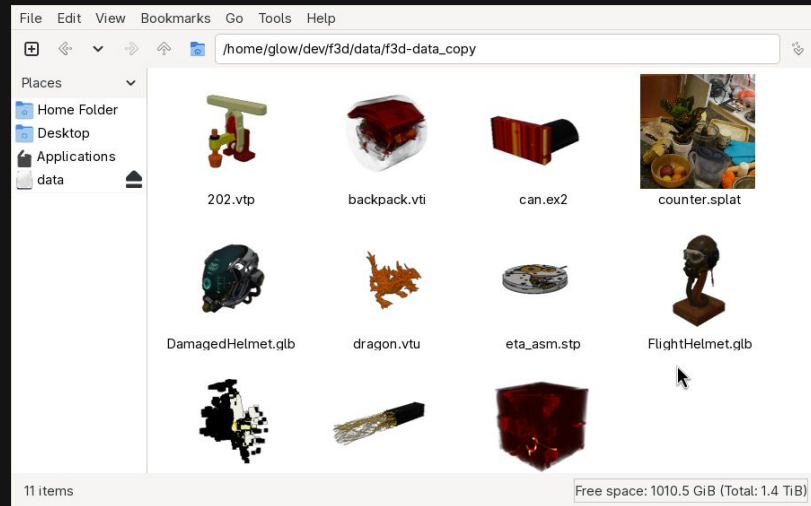
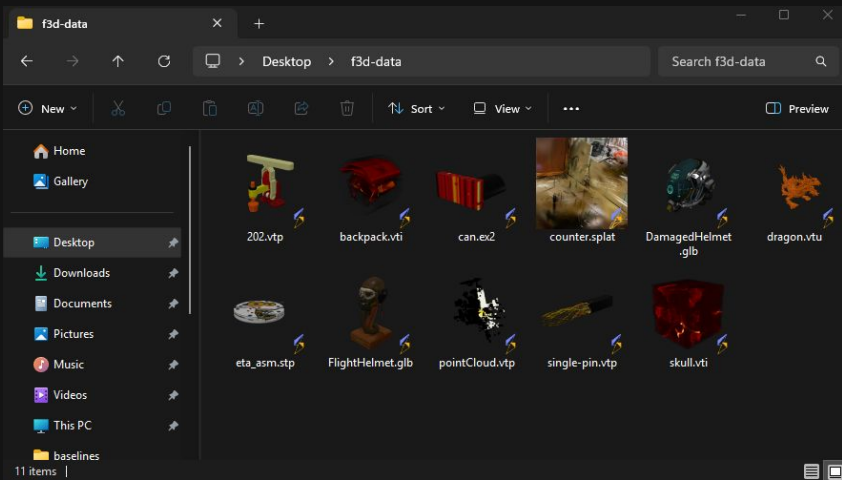
Copyright (c) 2018 Tyler Anderson



Copyright EYHN

# Scenario 2: I want to generate thumbnails for 3D files ?

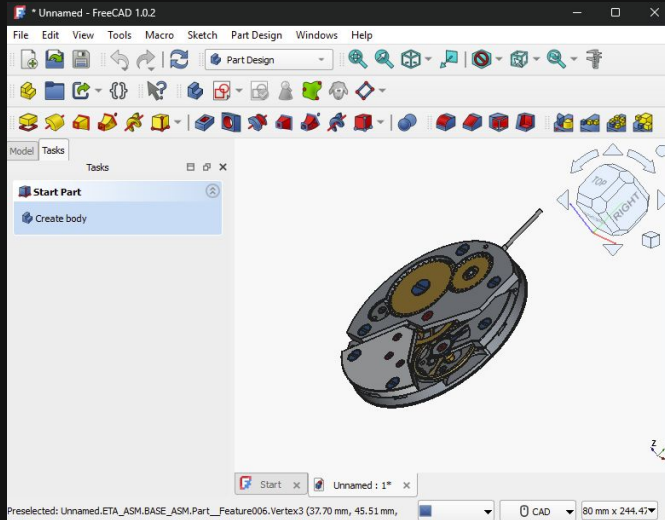
- Install F3D!
  - Native linux support with .thumbnailer files
  - Support sandboxing with egl/osmesa rendering
  - Windows support



# Scenario 3: I want to quickly check out a CAD file ?

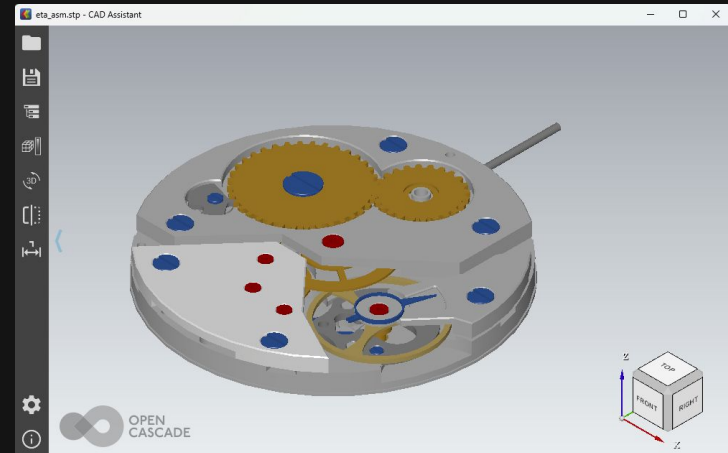
In the FOSS side:

- FreeCAD -> Import Step



In the closed side:

- CAD Assistant
- Autodesk Viewer
- eDrawings



## Scenario 3: I want to quickly check out a CAD file ?

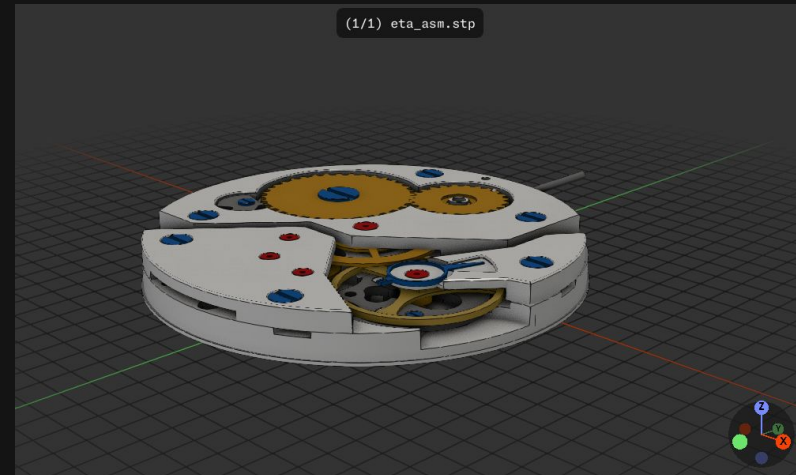
Just use F3D!

Command Line:

```
f3d /path/to/supported_file.ext
```

Desktop Environment:

Just double click on any (supported) file!





## Scenario 4: I want to integrate 3D rendering in a script ?

```
python my_pipeline.py file.gltf  
command file.gltf --output=file.png
```

- Write an custom python script to generate an image ?
- Write your own program in OpenGL ?

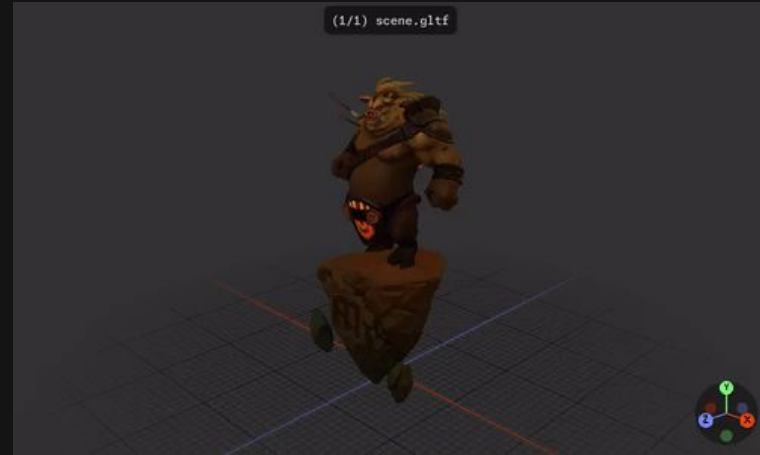
## Scenario 4: I want to integrate 3D rendering in a script ?

```
python my_pipeline.py file.gltf  
f3d file.gltf --output=file.png
```

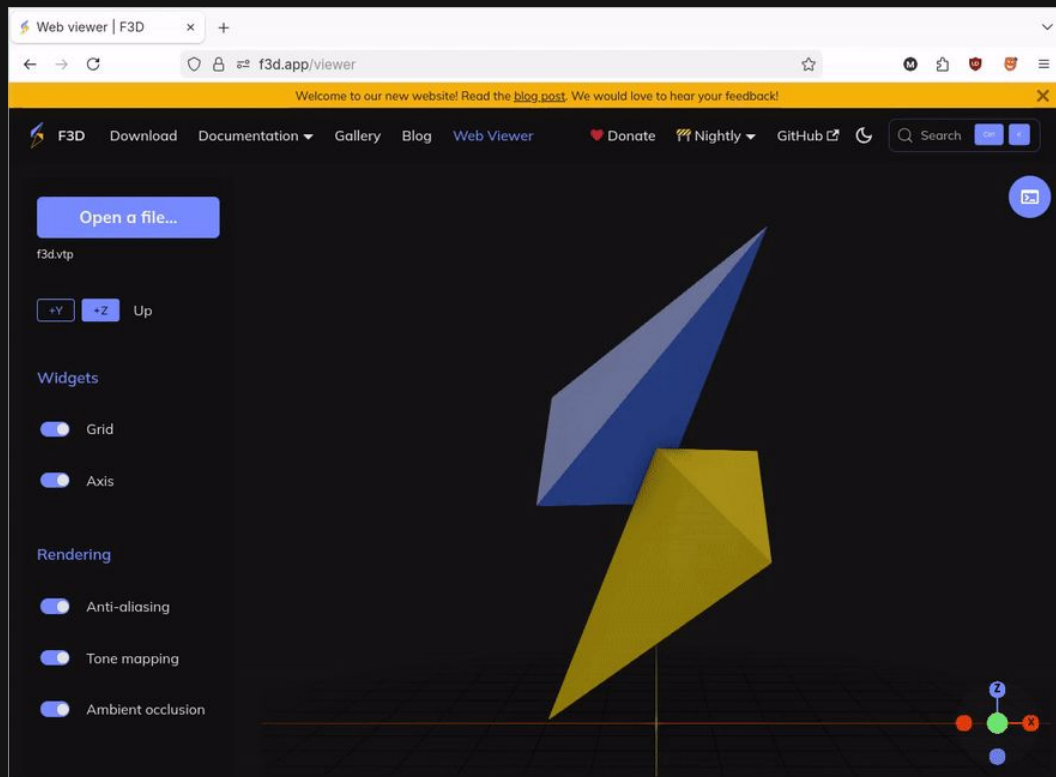
Just Use F3D! It even support animation!

```
f3d animated_file.gltf --output=render_{frame:4}.png  
convert *.png animation.mp4
```

Or even, just use the `libf3d` API in Python!



# Scenario 5 : I want to quickly visualize in the web



<https://f3d.app/viewer>:

Npm package:

<https://www.npmjs.com/package/f3d>

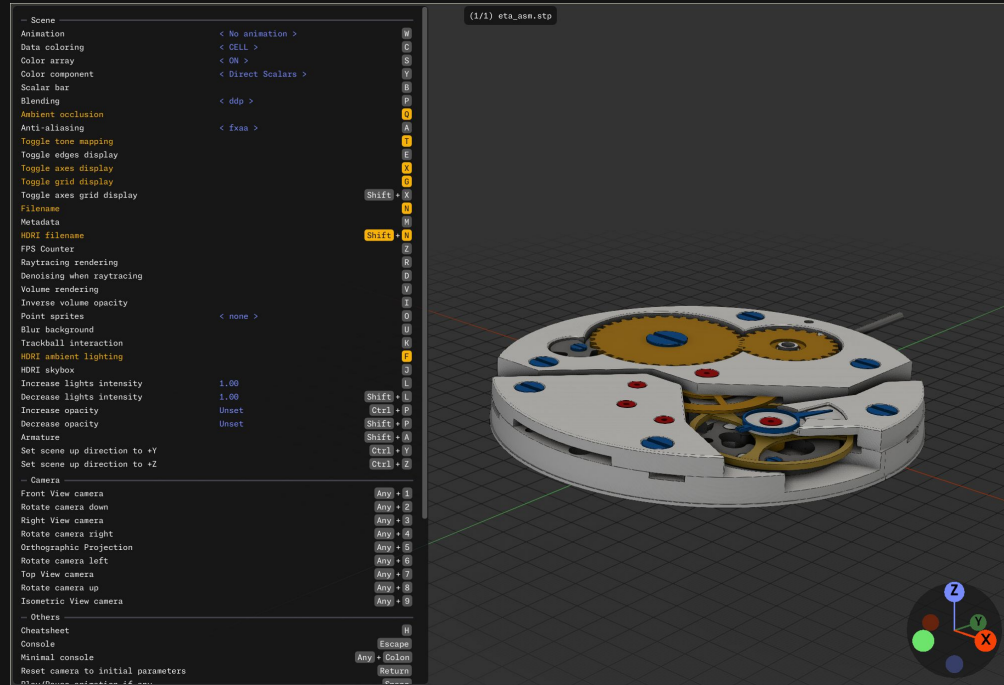


Just a viewer ?

# Using F3D: Interactive Key bindings

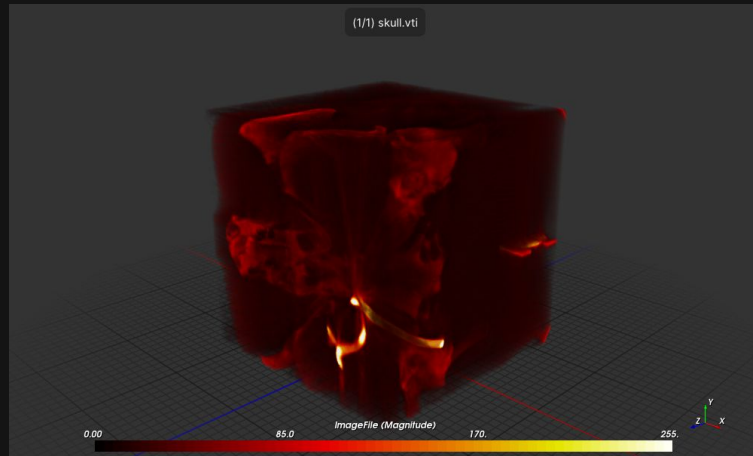
Many options can be changed by pressing a key.

-> Press H for cheatsheet

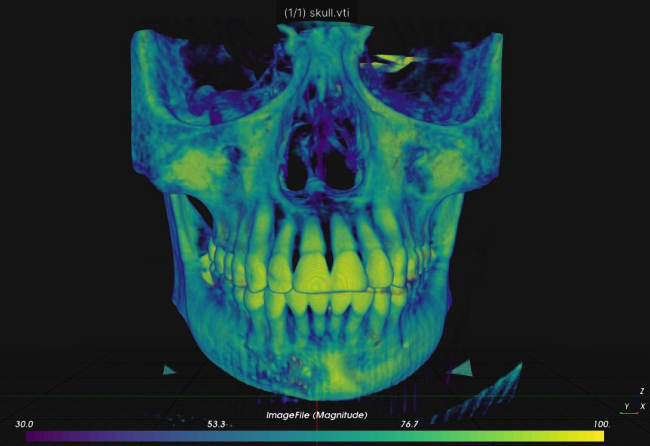


# Using F3D: Powerful CLI options

```
f3d skull.vti
```



```
f3d skull.vti --coloring-range=30,100 --up=+z  
--colormap-file=viridis --no-background
```





# Using F3D: All available in the Config file

- Every option can be enabled in a JSON-based config file
- Options can be enabled based on file name
- We ship “ready to use” config files in our prebuilt packages
- Different config files for interactive and thumbnail usage

```
[
  {
    "match": ".*(step|stp|iges|igs|brep|xbf)",
    "options": {
      "scalar-coloring": true,
      "load-plugins": "occt",
      "up": "+Z",
      "ambient-occlusion": true,
      "coloring-component": "-2",
      "coloring-by-cells": true,
      "camera-direction": "-1,1,-0.5"
    }
  }
]
```

# Using F3D: documentation

We do have a very detailed documentation deployed continuously, please check it out!



<https://f3d.app>

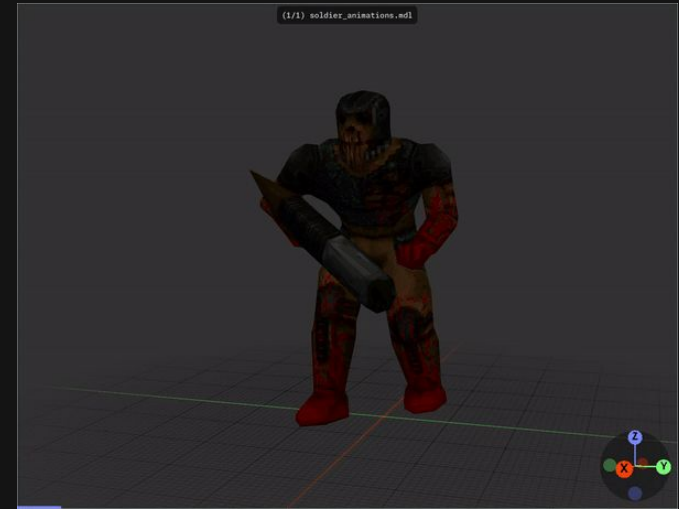
A screenshot of a web browser displaying the F3D Quickstart Guide. The browser's address bar shows the URL 'f3d.app/docs/next/user/QUICKSTART'. The page has a dark theme with a yellow header. A navigation menu on the left lists various sections like 'Quickstart Guide', 'Supported File Formats', and 'Interactions'. The main content area features a yellow banner with a warning message about the 'Nightly' version. Below this, the 'Quickstart Guide' title is followed by an overview paragraph. A 'Prerequisites' section explains that F3D needs to be installed. The 'Running F3D' section provides a list of instructions and terminal commands. The 'Constructing scenes' section is partially visible at the bottom. A right-hand sidebar contains a search bar and a list of links for further reading.



So many options, to do what ?

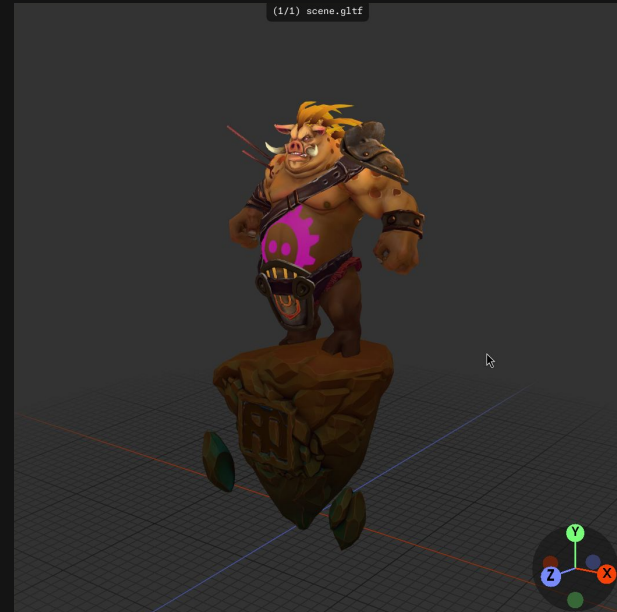
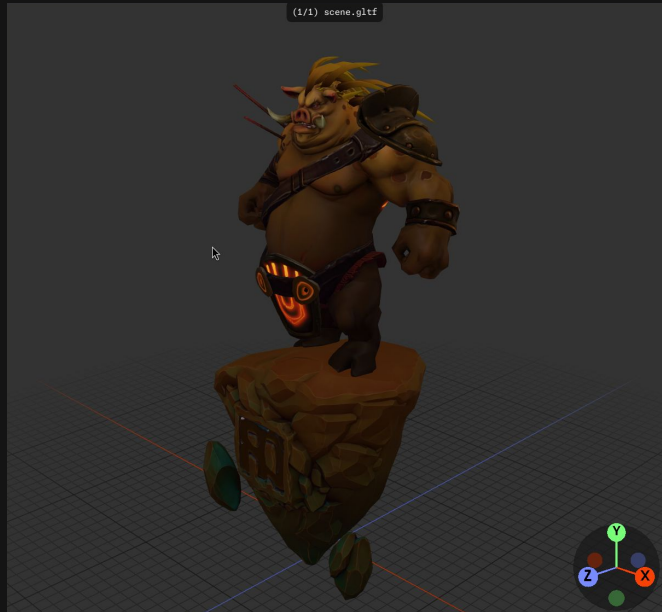
# Features: Animations

- Display any or all animations
- Control speed
- Go frame by frame



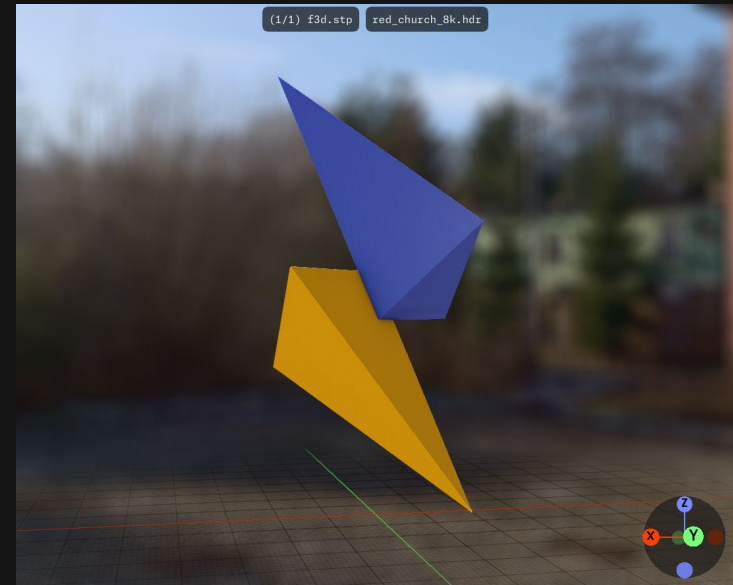
# Features: Textures

Display textures or choose your own using command line: `--texture-base-color` and more



# Feature: HDRIs

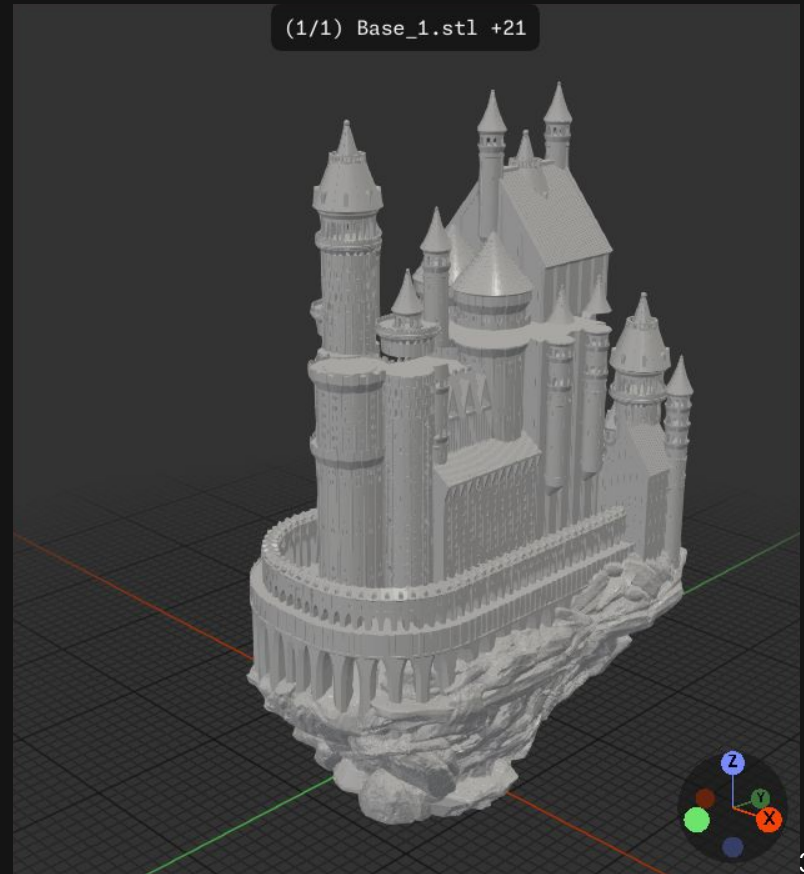
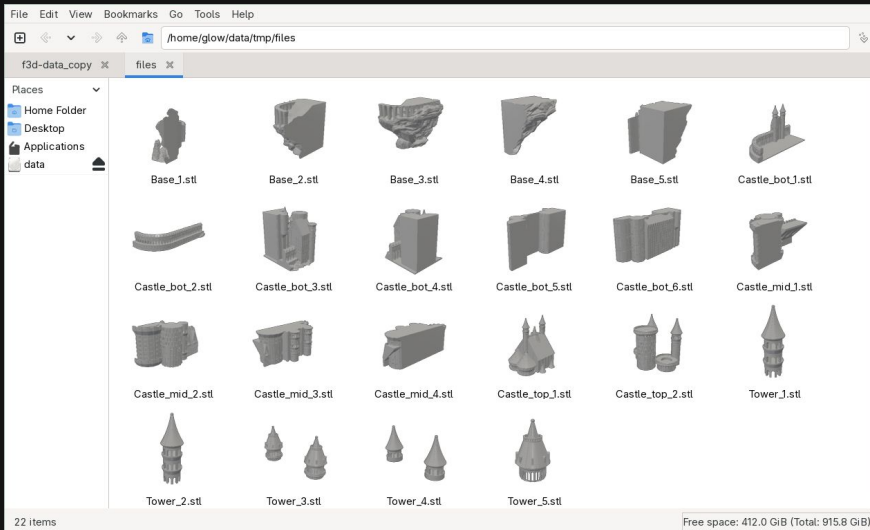
Light with and/or display stunning background HDRIs, with fine control and blur



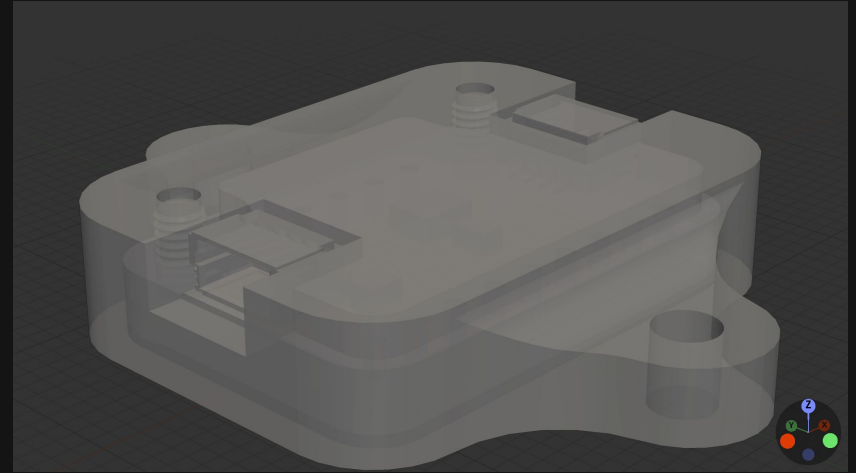
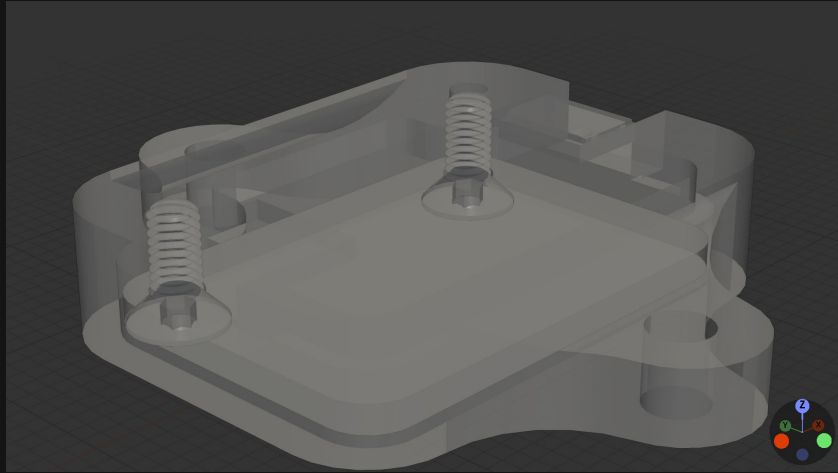
# Features: Multiple files

Combine multi part models easily with:

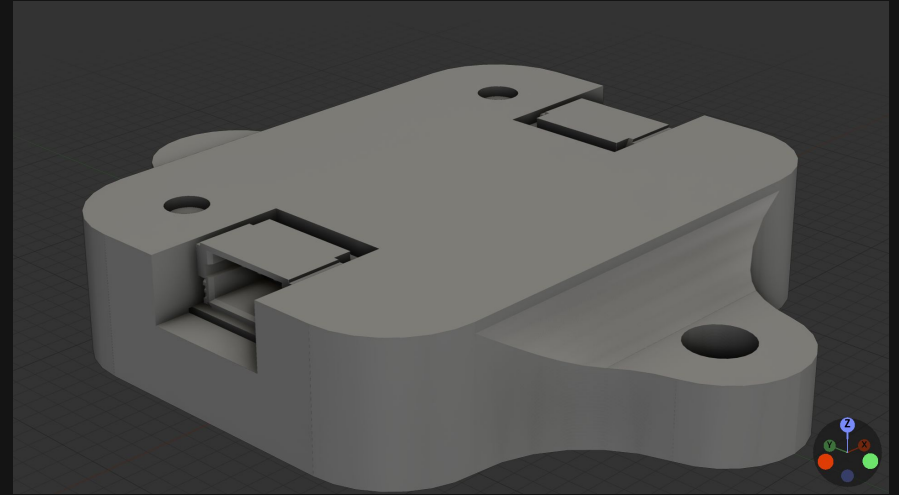
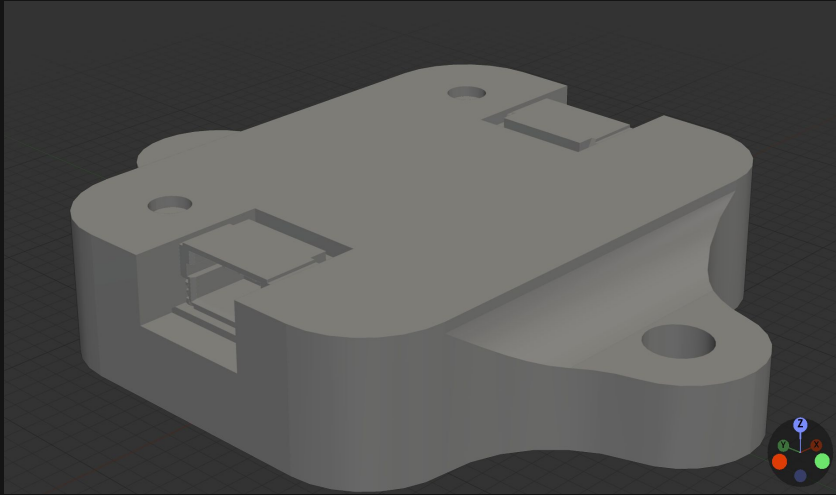
`--multi-file-mode`



# Features: Advanced rendering (blending)



# Features: Advanced rendering (ambient occlusion)



# Features: Gaussian Splatting

Classic GPU sort-based gaussian splatting as well as alternative sort-free stochastic blending



# Features: Analysis

- Display meta data interactively
- Recover more info in the terminal

```
Loading files:
/home/glow/dev/f3d/f3d/src/testing/data/soldier_animations.mdl

Current animation is: stand
Current animation time range is: [0, 0.8].

Animation(s) available are:
0: stand
1: dead
2: dead_right
3: reload
4: hit
5: down
6: stumble
7: run
8: shoot
9: walk
No camera available

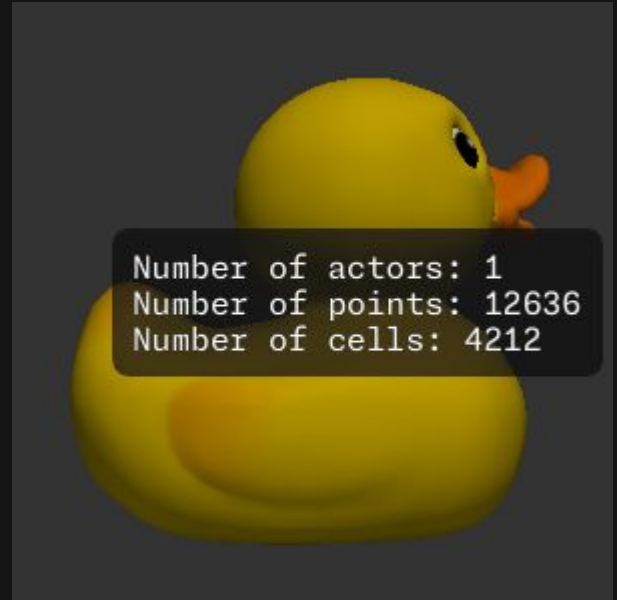
Number of files: 1
Number of actors: 1
-----

Not coloring

Scene bounding box: -10.7819 ≤ x ≤ 13.3392, -14.2737 ≤ y ≤ 20.0918, -24.3789 ≤ z ≤ 23.9942

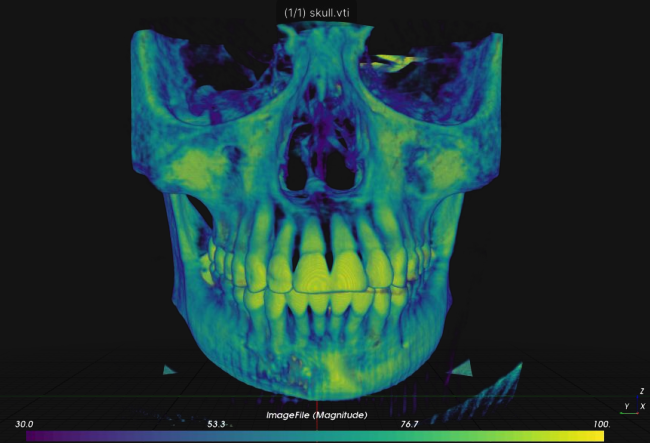
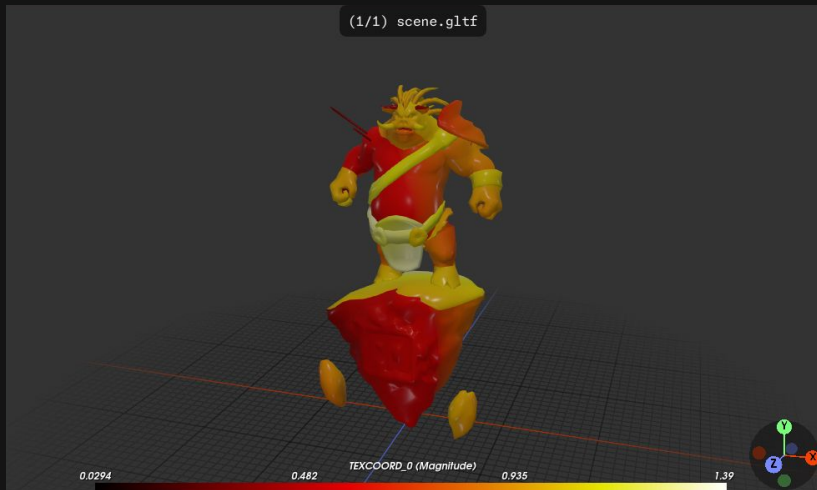
Camera position: 1.27865, -120.832, -0.192358
Camera focal point: 1.27865, 2.90905, -0.192358
Camera view up: 0, 0, 1
Camera view angle: 28.2222

Using grid unit square size = 10
Grid origin set to [1.27865, 2.90905, -24.379]
```



# Features: Scientific Visualisation

- Scalar coloring
- Volume rendering





That's it ?



# Extending F3D: Plugins

- Every file format is handled by a plugin
- F3D officially supports 7 plugins:
  - native: all formats natively supported by VTK or F3D
  - alembic: depends on Alembic (.abc files)
  - draco: depends on Draco (.drc and GLTF Draco extension)
  - hdf: requires VTK HDF module (.vtkhdf and exodus files)
  - occt: depends on OpenCASCADE (.stp, .iges, ...)
  - usd: depends on OpenUSD (.usd related files)
  - vdb: requires VTK VDB module (.vdb files)
- We ship a plugin SDK to extend F3D
- A plugin exists for Abaqus: <https://github.com/yangcshen/F3D-ODB-Reader-Plugin>
- Any existing VTK-based reader/importer can be easily wrapped into a F3D plugin with minimal efforts

# Integrating F3D: C++ API

F3D isn't just an application, it's also a modern C++17 library! Shipped in the binary!

```
#include <f3d/engine.h>
#include <f3d/interactor.h>
#include <f3d/options.h>
#include <f3d/scene.h>

// Load VTK native readers
f3d::engine::autoloadPlugins();

// Create a f3d::engine
f3d::engine eng = f3d::engine::create();

// Recover the options and set the wanted value
f3d::options& opt = eng.getOptions();
opt.render.effect.ambient_occlusion = true;
opt.render.effect.antialiasing.enable = true;

// Standard libf3d usage
eng.getScene().add("path/to/file.ext");
eng.getInteractor().start();
```

```
#include <f3d/engine.h>
#include <f3d/image.h>
#include <f3d/scene.h>
#include <f3d/window.h>

// Load VTK native readers
f3d::engine::autoloadPlugins();

// Create a f3d::engine with a offscreen window
f3d::engine eng = f3d::engine::create(true);

// Load a geometry
eng.getScene().add("path/to/file.ext");

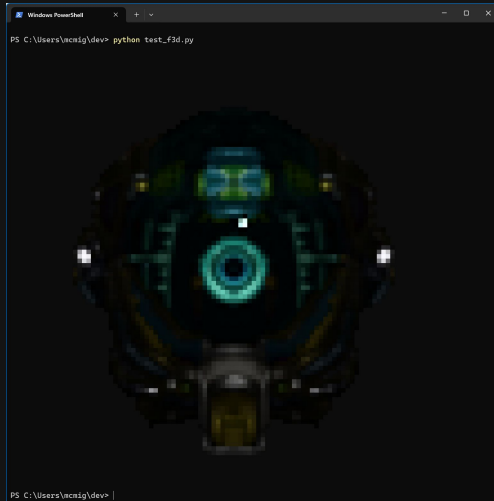
// Set the window size and render to an image
f3d::image img = eng.getWindow().setSize(300, 300).renderToImage();

// Save the image to a file
img.save("/path/to/img.png");
```

# Integrating F3D: Python

Each C++ API is bound to Python!

```
pip install f3d
```



```
import f3d

def main():
    # setup offscreen engine
    engine = f3d.Engine.create(offscreen=True)
    engine.options.update(
        {
            "scene.up_direction": "+Y",
            "render.effect.tone_mapping": True,
            "render.effect.ambient_occlusion": True,
            "render.effect.antialiasing.enable": True,
            "render.effect.antialiasing.mode": "ssaa",
        }
    )
    engine.scene.add("DamagedHelmet.glb")
    engine.window.size = 100, 100

    # render
    image = engine.window.render_to_image(no_background=True)
    print(image.to_terminal_text(), end="")

if __name__ == "__main__":
    main()
```

# Integrating F3D: Python showcase

Some results made using Python scripts



# Integrating F3D: Other bindings

- C bindings
- Java bindings
- JavaScript (wasm) bindings

```
f3d_engine_autoload_plugins();
f3d_engine_t* engine = f3d_engine_create(0);
f3d_options_t* options = f3d_engine_get_options(engine);

f3d_options_set_as_bool(options, "render.grid.enable", 1);
f3d_options_set_as_bool(options, "render.show_edges", 1);
f3d_options_set_as_bool(options, "ui.axis", 1);
f3d_options_set_as_bool(options, "ui.fps", 1);
f3d_options_set_as_bool(options, "ui.animation_progress", 1);
f3d_options_set_as_bool(options, "ui.filename", 1);

f3d_scene_t* scene = f3d_engine_get_scene(engine);
f3d_scene_add(scene, "f3d/testing/data/dragon.vtu");

f3d_interactor_t* interactor = f3d_engine_get_interactor(engine);
f3d_interactor_start(interactor, 1.0 / 30.0);

// ...

f3d_engine_delete(engine);
```

```
import app.f3d.F3D.*;

public class F3DExample {
    public static void main(String[] args) {

        Engine.autoloadPlugins();

        // Always use try-with-resources idiom to ensure the no
        try (Engine engine = new Engine(Window.Type.NATIVE)) {
            Scene scene = engine.getScene();
            scene.add("f3d/testing/data/dragon.vtu");

            engine.getWindow().render();
        }
    }
}
```

```
f3d(settings)
    .then(async (Module) => {
        // write a 3D file located on the server to the internal filesystem
        const modelFile = await fetch("/assets/example.obj").then((b) =>
            b.arrayBuffer(),
        );
        Module.FS.writeFile("example.obj", new Uint8Array(modelFile));

        // automatically load all supported file format readers
        Module.Engine.autoloadPlugins();

        // create an engine
        Module.engineInstance = Module.Engine.create();

        Module.setupOptions(Module.engineInstance.getOptions());

        // setup the window size based on the canvas size
        const scale = window.devicePixelRatio;
        Module.engineInstance
            .getWindow()
            .setSize(Module.canvas.clientWidth, scale * Module.canvas.clientHeight);

        const scene = Module.engineInstance.getScene();
        scene.add("/example.obj");

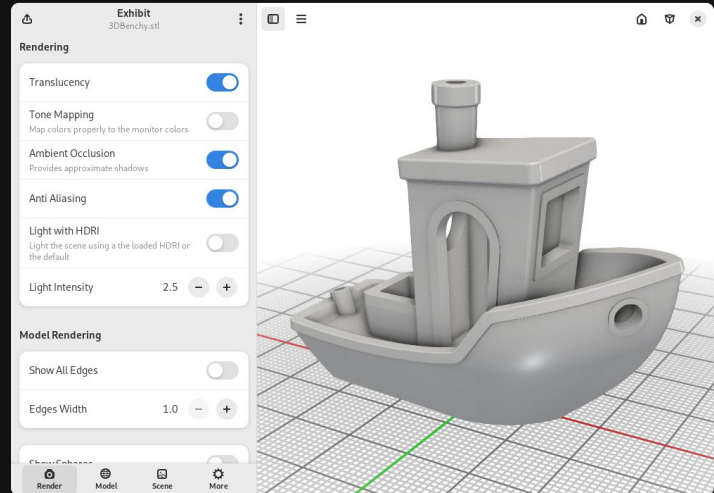
        // do a first render and start the interactor
        Module.engineInstance.getWindow().render();
        Module.engineInstance.getInteractor().start();
    })
    .catch((error) => {
        console.error("Internal exception: " + error);
    });
```

# Integrating F3D: External window

- F3D can be integrated into an existing application by hooking the OpenGL context handled at the application level.
- We successfully manage to integrate it in QT, QML, FLTK, GLFW applications using C++
- Exhibit uses GTK+Python

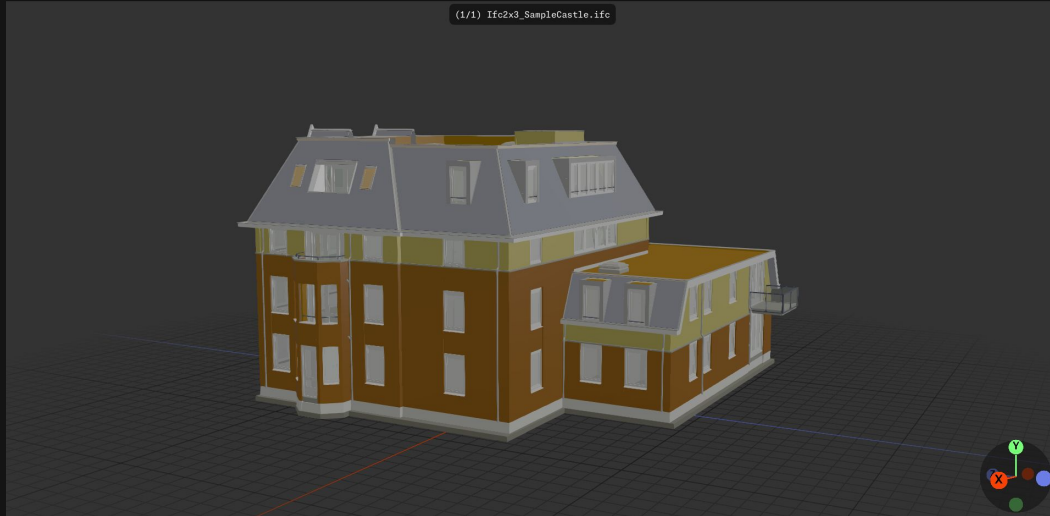
Exhibit  
GNOME application

Credits: <https://github.com/Nokse22/Exhibit>



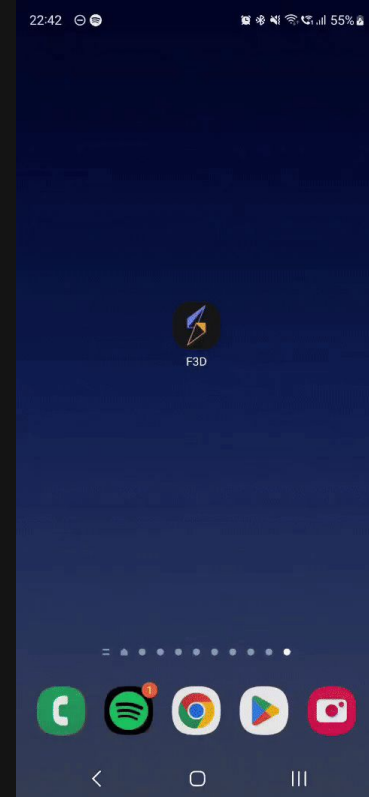
# Perspectives

.ifc Support



Android: <https://github.com/f3d-app/f3d-android>

Linux mobile planned too



# Conclusion



**F3D APP**  
FOUNDATION

- Non-profit foundation
- Donations
- Community support on Discord
- Contacts:
  - [f3d.app](https://f3d.app)
  - [discord.f3d.app](https://discord.f3d.app)
  - [contact@f3d.app](mailto:contact@f3d.app)

## Credits:

*bristleback model* by Nikolay\_Tsys  
*Sir Frog - Chrono Trigger* by Adrian Carter  
*DamagedHelmet* by ctxwing and theblueturtle\_  
*soldier\_animations.mdl*: LibreQuake Project  
*Watch movement* by Greg Brown  
*Medieval Castle* by boldmachines  
*Duck*: assimp test models  
*SampleCastle*: ThatOpenEngine  
*Adafruit case*: Hawkry Zhang (GrabCAD)

