Lab Activity 0 - GenAI and Meshlab GUI

Welcome to your first hands-on experience with 3D graphics creation! In this lab, you'll explore how modern generative AI (GenAI) tools can create detailed 3D graphics from something as simple as a text description or a single image — no coding required.

You'll get to try out **Hyper3d-Rodin**, one of the most cutting-edge tools in graphics and GenAI to see just how far technology has come. After generating your own 3D models, we'll walk you through how to open, view, and do some basic editing on these 3D graphics using beginner-friendly software, **Meshlab GUI**. This will give you a first glimpse into how future graphics will look like.

We have some simple tasks at the end of this document, which include generating a few meshes and aligning them.

Hyper3d-Rodin

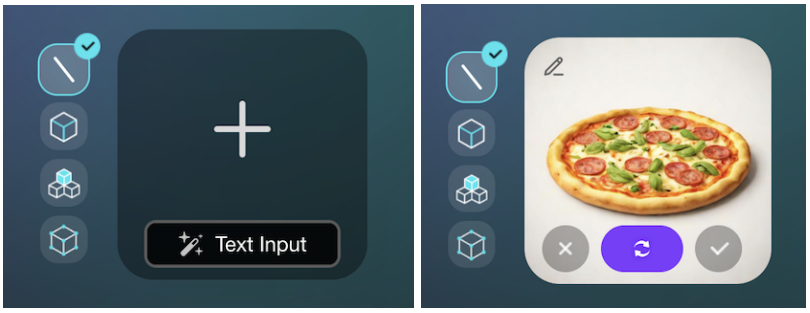
Rodin was recently developed by **Hyper3D.ai** and published at **SIGGRAPH 2024**. It's one of the newest and most powerful GenAI tools for creating 3D models directly from a text prompt or a single image. To begin, head to the following link and sign in — for example, using your UCSC email address.
https://hyper3d.ai/rodin

Rodin so far offers you 5 credits once signed in, which should be sufficient for this lab. If you run out of the free credits quickly and don't wanna pay anything, try using different emails/accounts to sign in again.

There are other exciting GenAI tools out there, such as **Lens Studio** (app by Snapchat), **Meshy.AI**, and **Luma AI – Genie**, which you're welcome to explore if you're curious. However, for this lab, we've chosen Rodin because it's easy to use, runs fully in the browser, and produces high-quality 3D outputs with minimal effort.

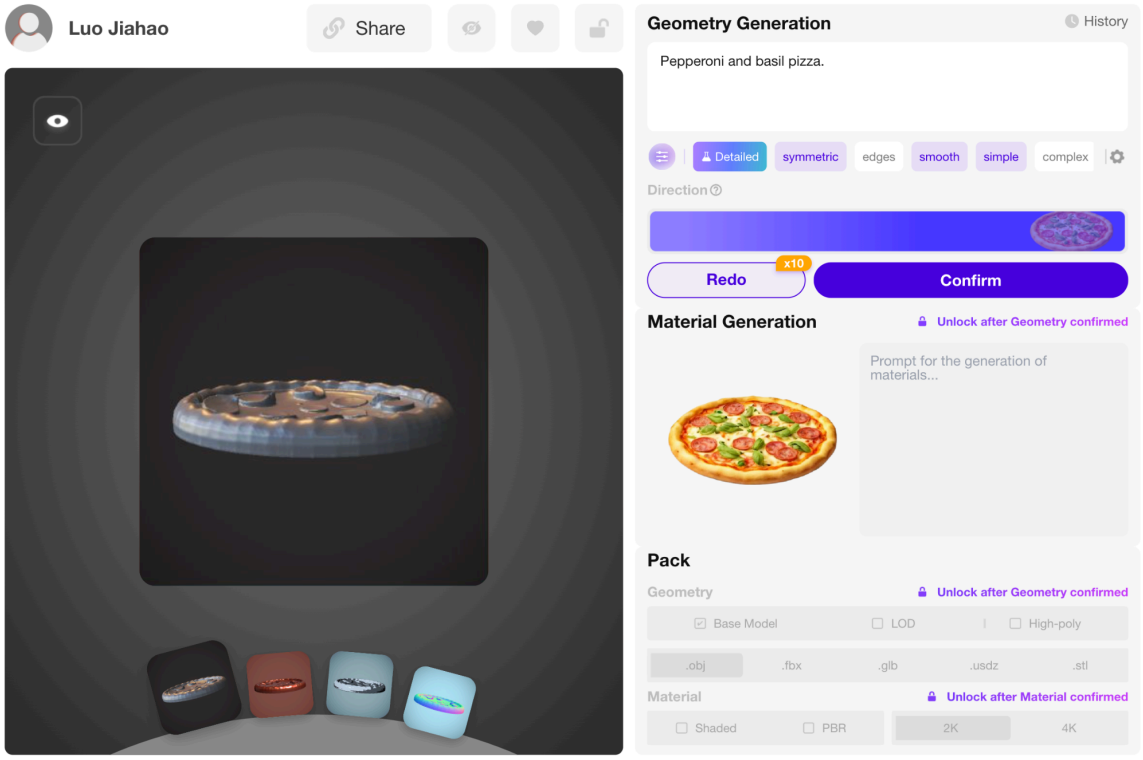
Text-to-image generation & image-to-3D generation

If you've ever used a generative AI tool like **Midjourney** or even **ChatGPT**, you're probably already familiar with the idea of text-to-image generation. To try it out, click on the **"Text Input"** option and type in a description like **"pizza"** or anything else you'd like to see in 3D. You can regenerate the image as many times as you'd like until you're happy with the result. Once you've found one you like, click  to confirm your selection. After that, click **"Generate"** to turn your chosen image into a 3D mesh model.



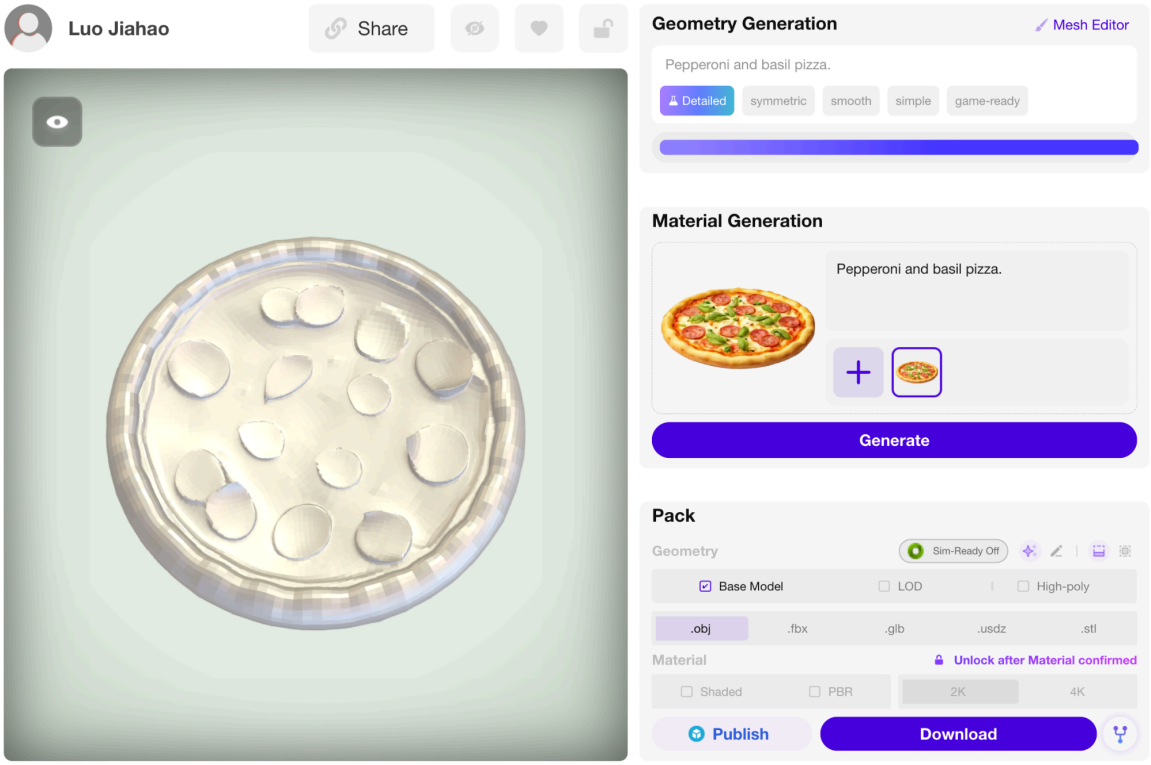
You're also welcome to use your **own images** instead of text prompts. Just keep in mind that **images with a single, clearly visible object and without a complex background** tend to work best. Even though Rodin represents state-of-the-art 3D generation, it still has its limitations — and that's exactly why research in this area is ongoing!

After a short wait, you'll see a window that looks like the one below.

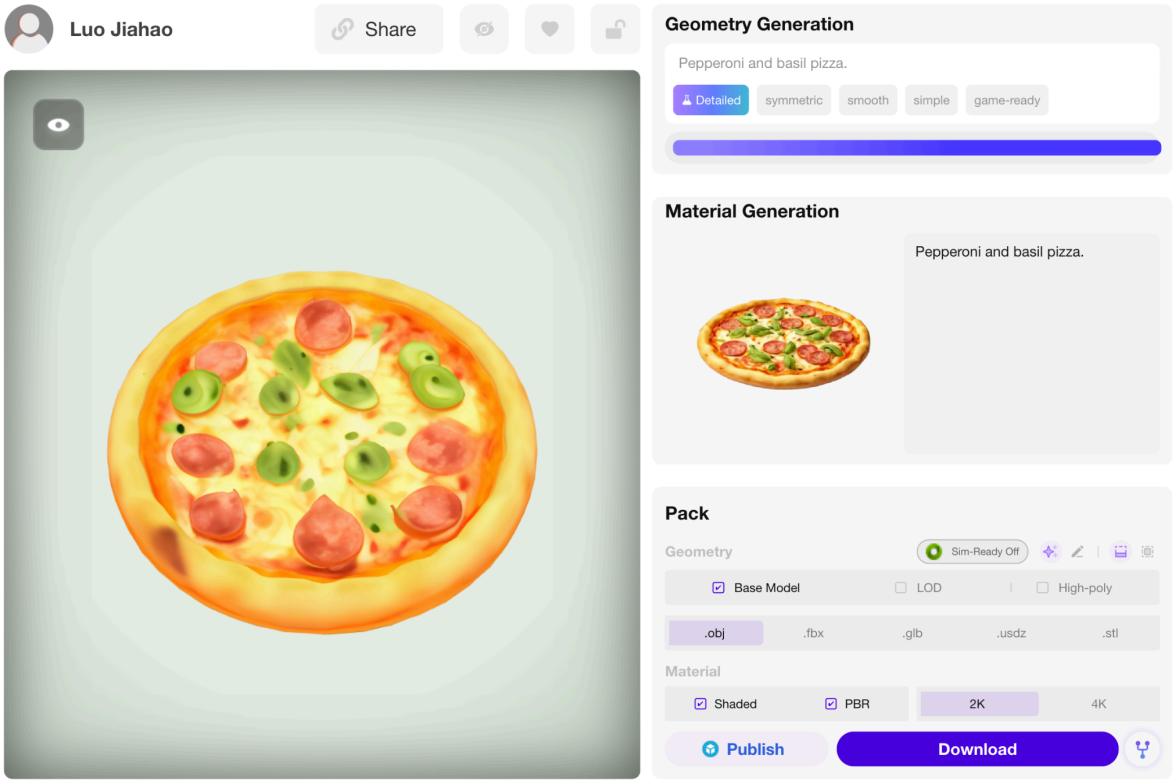


On the **left**, you'll find a **preview of the generated 3D model**, which you can rotate by clicking and dragging with your mouse. In the **top-right corner**, we recommend adding a short **text description under "Geometry Generation"** to help improve the quality of the 3D result — even if you uploaded an image. You can click **"Redo"** multiple times to refine the image-to-3D conversion before moving on. Once you're happy with the result, click **"Confirm"** to proceed to the next stage. Keep in mind that each confirmed generation will **cost 0.5 credits**.

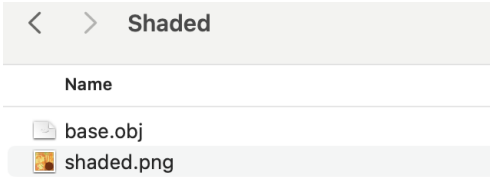
Next, proceed to **generate the material** for your 3D model. For this lab, make sure to select **“Base Model”** as the type and choose the **“.obj” file format**



Once the material is **confirmed**, go ahead and **download the model** to your computer.



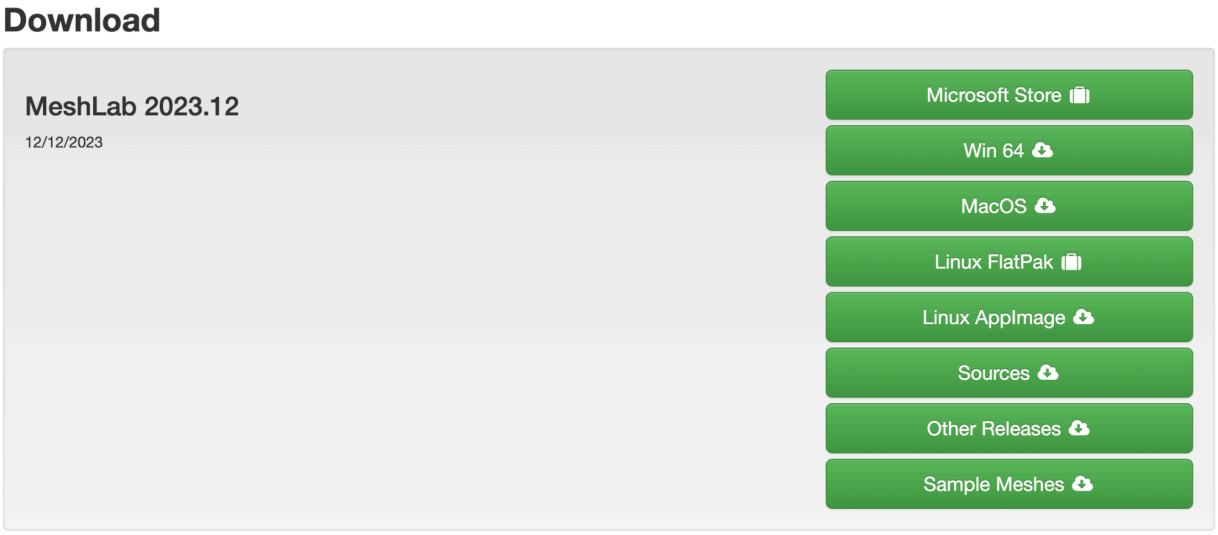
After you unzip the downloaded model, go to **Shaded** and you will see a '.obj' file and an image.



Although you can rotate and explore the model directly on the **Rodin** website, we'll now introduce you to one of the most commonly used tools by researchers for viewing and doing basic edits on .obj files: **MeshLab**.

Download and Install MeshLab

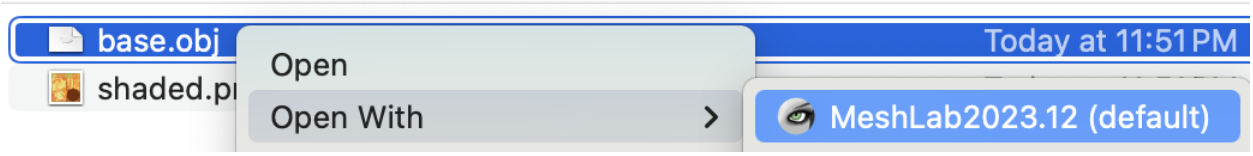
1. Go to the official website: https://www.meshlab.net
2. Click on **Download**, and choose the version that matches your operating system (Windows, macOS, or Linux).
3. Install MeshLab by following the installation instructions.
4. Once installed, launch **MeshLab** — you're ready to go!



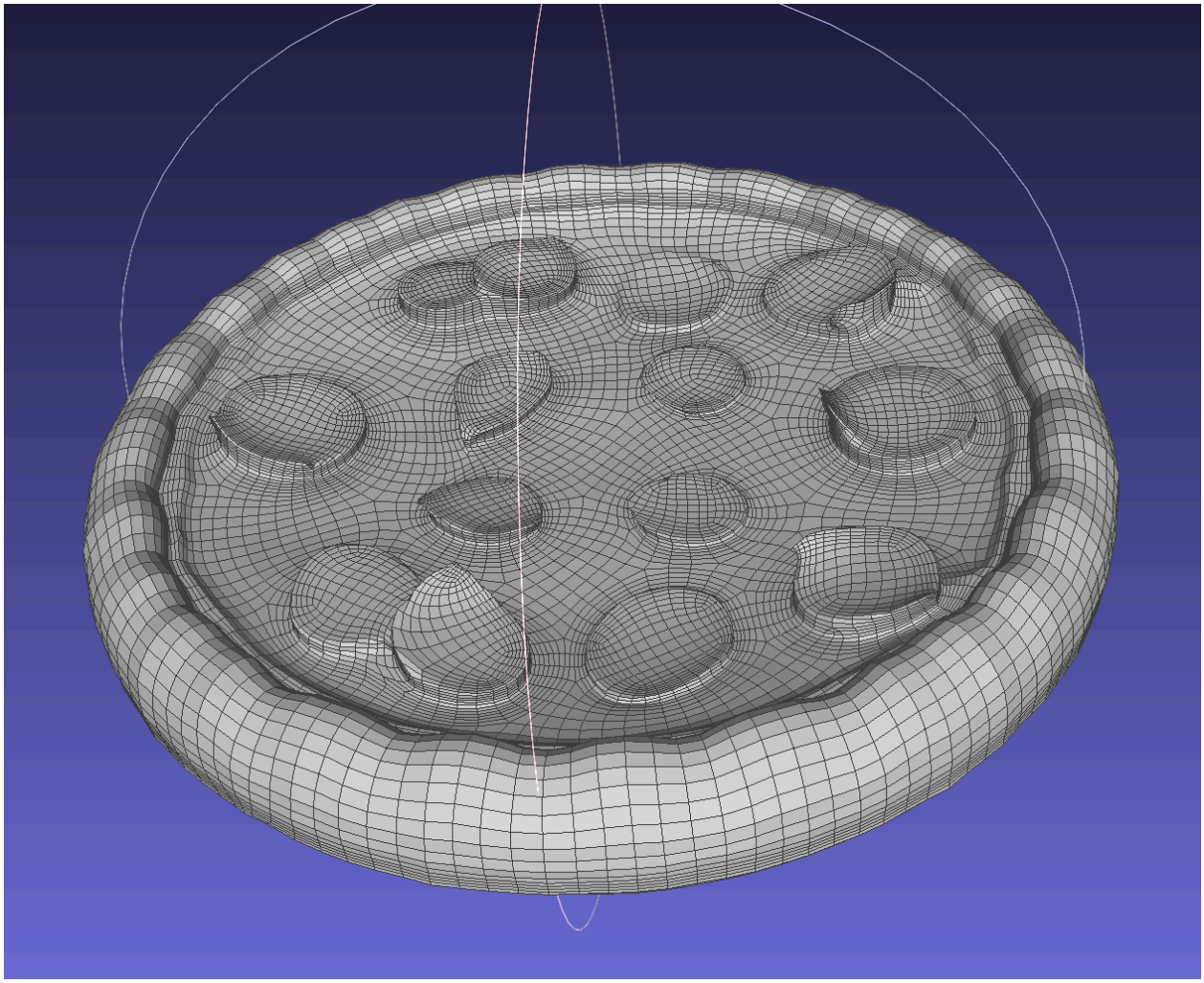
Load a Sample 3D Model

Let's start by opening `'Shaded/base.obj'` from Rodin in Meshlab. You can either

- Open MeshLab, then go to `File` > `Import Mesh`, or
- `'Open with'` Meshlab as the following.



You see your model without color similar to the following



(Right, you will see an error when opening it. Some issues with the color. Just click **ok** for now)

Learn to Navigate Around the Model

Similar to Rodin, in Meshlab you can navigate the model by

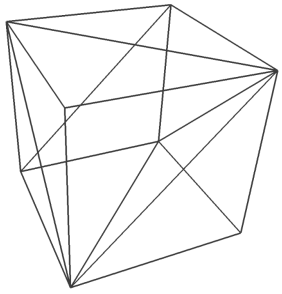
- **Rotate**: Click any point on the mesh and drag with the **left mouse button**.
- **Zoom**:
    - Use the **mouse scroll wheel**

    - hold **right mouse button** and drag up/down
    - Use two fingers on the MacBook's trackpad
    - …
- **Centerize**: Double click a point on the mesh to let the camera center at this point.

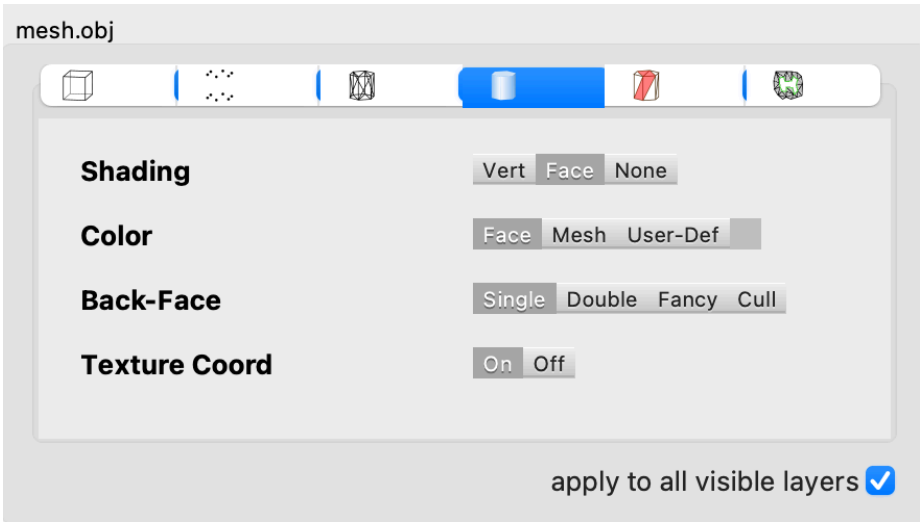
View the Model in Different Modes

You probably have already noticed that the 3D model is represented with a surface. Here you can see the surface is represented by a lot of polygons (Quadrangles if generated by Rodin). More information can be found in **Lecture Object Modeling, triangles and meshes** in week 2.

We can visualize the surface in different modes in Meshlab, such as Fill  , Points  , and wireframe  on the top right. We have **Fill** : see the full mesh (default, polygons), **Points**: See just the vertices and **Wireframe**: Shows the mesh edges as lines. Here is points and wireframe visualization of a cube made with triangles.



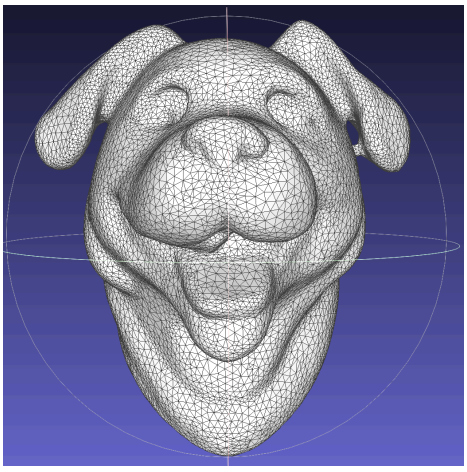
You will learn different shading in **Lecture Shading/Lighting** in week 7. In the lower part selecting  , you can visualize the mesh with

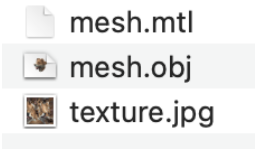


- **Shading**: vertex shading, face shading, and None
- **Color**: Change the color of the mesh
- **Texture**: Turn on and off the texture map

Textured triangled OBJ file from Rodin quadrangles

Note that so far we get an error loading the texture. However, if you open the provided ‘dog/mesh.obj’ in Meshlab, you will see the texture.





If you compare your file from Rodin and this dog example. You will find the key difference is there is an additional ‘.mtl’ file with dog.

Besides, the dog is made of triangles rather than quadrangles generated by Rodin.

The `.obj` format is a very simple, straight-forward text-based format used to represent 3D models. How it defines a mesh is actually very similar to how we define a mesh in WebGL. If you open it with a text reader, you will see something similar to

```
v -0.732315 0.166753 0.023381    vn 0.7607 -0.0590 -0.6465   vt 0.222065 0.397257
v -0.726013 0.159307 0.027947    vn 0.7632 -0.0204 -0.6459   vt 0.218733 0.396282
v -0.728972 0.162883 0.026991    vn 0.7589 0.2057 -0.6178    vt 0.121916 0.347887
v -0.719866 0.146033 0.038452    vn 0.8788 -0.0243 -0.4765   vt 0.124588 0.347336
```

```
f 6577/3817/4874 6569/3816/4873 6572/4476/5690
f 6578/3819/4876 6583/4481/5688 6586/4493/22529
f 6578/3819/4876 6577/3817/4874 6583/4481/5688
f 6588/4484/5698 6391/3820/4877 6580/3821/4878
```

(triangle mesh, e.g. dog/mesh.obj)

It contains definitions for **vertices** (`v x y z`), **vertex normals** (`vn nx ny nz`), **texture coordinates** (`vt u v`), and **faces** (`f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3`) that describe the geometry and appearance of the model.

Normals (vn) define the direction a vertex is facing and are used for lighting and shading.

Texture Coordinates (vt) is used to map a texture image onto the surface of your 3D model. For example, map `dog/texture.jpg` to `dog/mesh.obj`

Faces (f) define the actual **surfaces** of the model by connecting vertices with indices. Indices in here start from 1 (not 0). E.g. ‘f 4800/2814/3632 4801/2818/3640 3969/2757/3548’ means

make a face (here a triangle) using

- vertex (v): 4800, 4801, 3969
- texture coordinate (vt): 2814, 2818, 2757
- vertex normals (vn): 3632, 3640, 3548

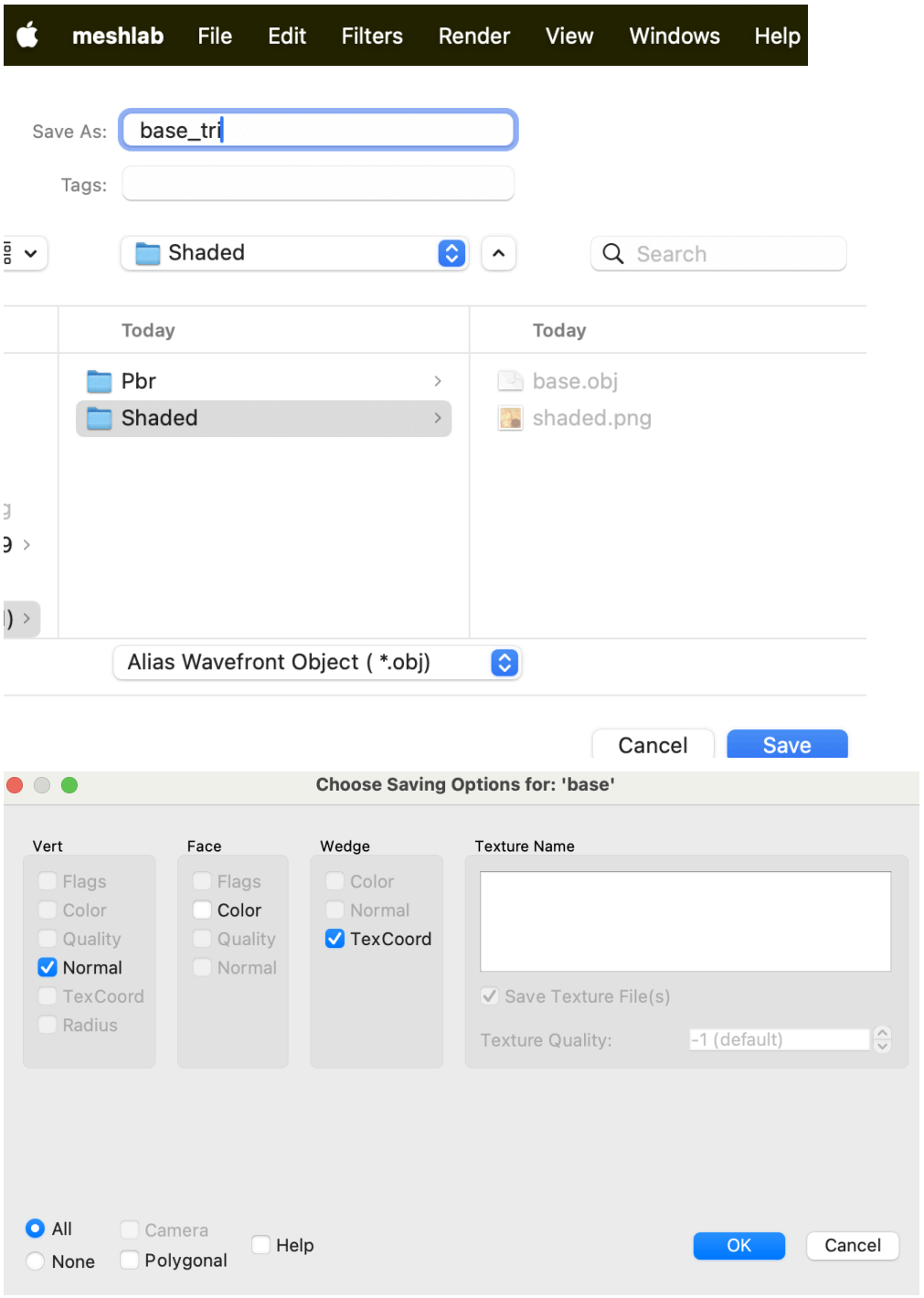
As you can see for the model you get from Rodin, we get one more column as quadrangles:

```
f 2991/3138/2137 2749/2896/1980 2747/2894/1978 2746/2893/1977
f 2994/3141/2140 2993/3140/2139 2995/3142/2141 2996/3143/2142
f 2994/3141/2140 2996/3143/2142 2997/3144/2143 2752/2899/1983
f 2994/3141/2140 2752/2899/1983 2750/2897/1981 2749/2896/1980
```

To apply materials (like colors or textures) to the geometry, the `.obj` file usually references an external **material file** (`.mtl`) using the `mtllib` directive. The `usemtl` command specifies which material (defined in the `.mtl` file) should be applied to subsequent faces. The material file itself describes surface properties like diffuse color, specular highlights, and texture maps.

Since we already have the texture coordinate (vt) for the file from Rodin. We just need to generate the .mtl file to let Meshlab actually load the texture image ‘shaded.png’.

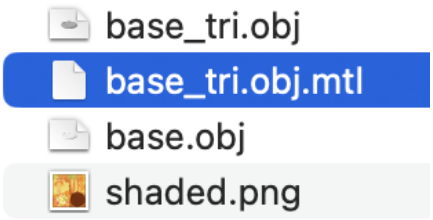
Additionally, since future labs and assignments will involve loading 3D models in **WebGL**, which expects meshes to be composed of **triangles**, we’ll convert any **quadrangle surfaces** to **triangle surfaces** at this stage. To do this on the top left, we select `File` > `Export Mesh As`



Unclick Color and Polygonal.

In MeshLab, **unchecking “Polygonal”** during export tells the program to **convert and save the mesh using triangle surfaces only**, instead of keeping any quadrangle faces.

Then open the newly created .mtl file. Add ‘map_Kd <your_texture_img>’ (here is ‘shaded.png’).



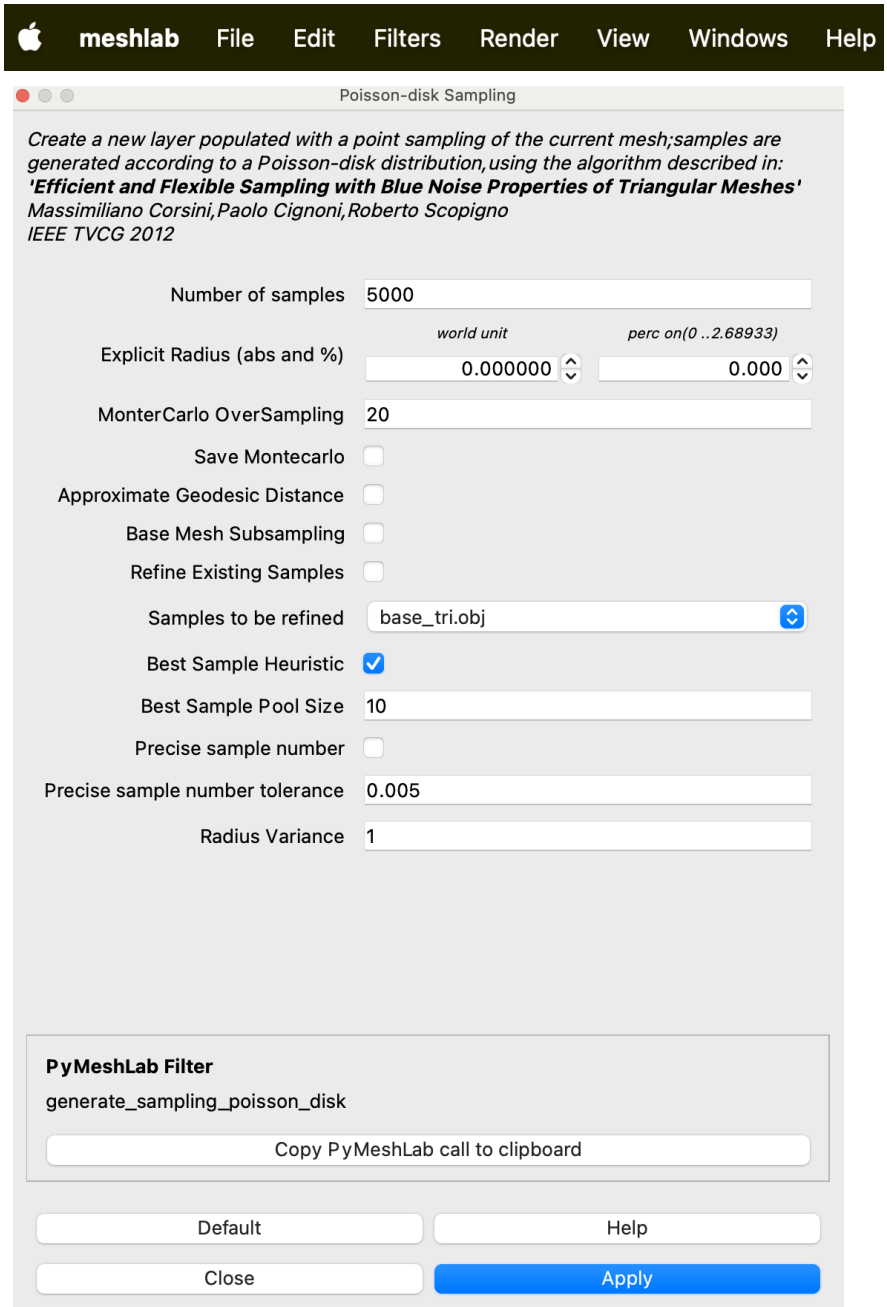

```
newmtl material_0
Ka 0.200000 0.200000 0.200000
Kd 0.752941 0.752941 0.752941
Ks 1.000000 1.000000 1.000000
Tr 0.000000
illum 2
Ns 0.000000
map_Kd shaded.png
```

Open the newly created .obj file. You will see the texture mesh without getting any error!

Sample and mesh and create a new low-res mesh

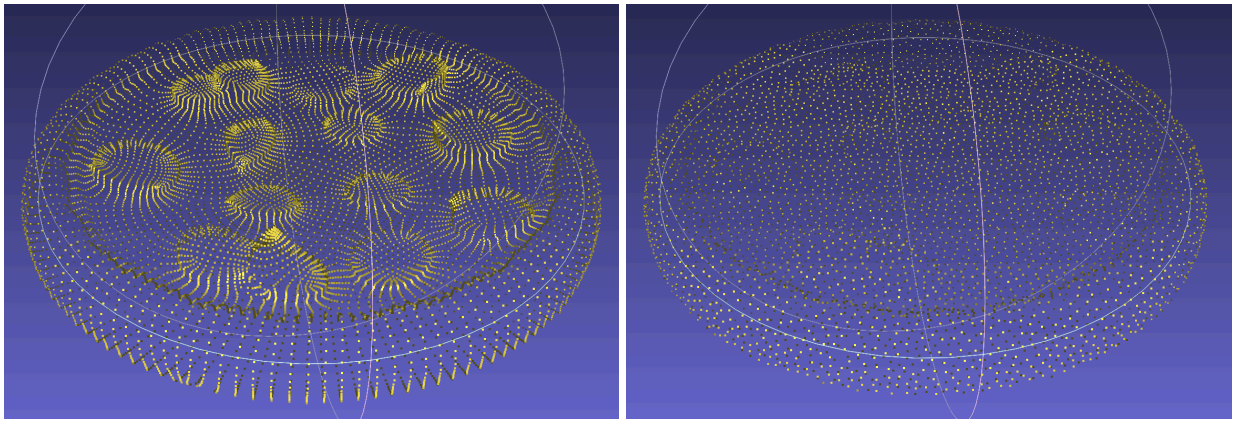
Again, in the later lab/assignment we want you to load this mesh. In many cases you don't need a high resolution mesh with thousands of triangles (probably due to performance issues). One common way is to uniformly sample from the mesh surface and create a lower-resolution mesh. You will need to

- Sample the mesh and specify the target vertex number: `'Filters → Sampling → Posson-disk Sampling Smoothing'`. Specify **Number of samples** and keep other parameters as default.

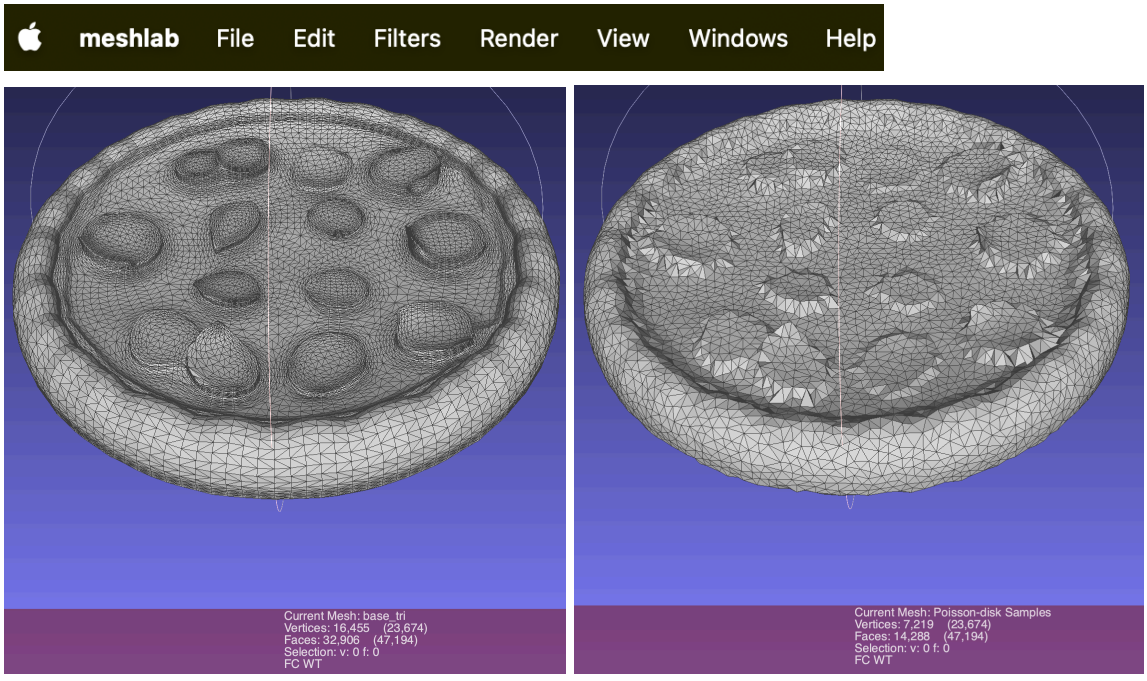


After you click **Apply**, MeshLab will automatically generate a new point cloud — usually named **Poisson-disk Samples** — which often contains **more points than the number you specified**.

To compare it with the original mesh, switch the view mode to **Points**  . This will let you **visually distinguish the sampled points from the full mesh**, and clearly see the difference in structure and density as the following.



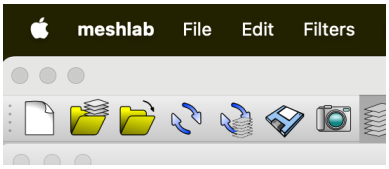
- Select **Poisson-disk Samples,** and **create a new mesh** by triangle reconstruction: `'Filters → Remeshing, Simplification and Reconstruction → Surface Reconstruction: Ball Pivoting'`. Leave the default parameters as they are and click **Apply.** MeshLab will then generate a **triangulated mesh** based on the sampled points you created earlier. When you compare this new mesh to the original as the following, you'll notice a clear difference — especially as the **number of vertices and triangles is significantly reduced**. This gives you a simplified version of the original.

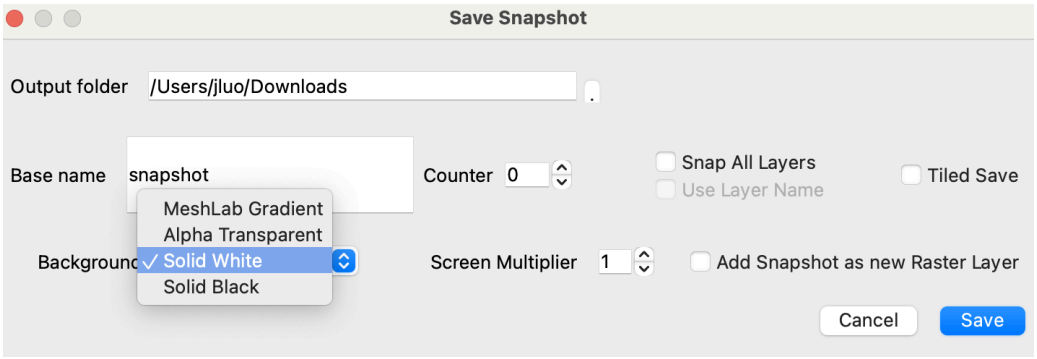


Other commonly used functionalities

Save a Snapshot of the mesh

Click the camera  on the top left, or `File` > `Save snapshot`. Specify the path and the background type. Meshlab will render and save the current view as an image.

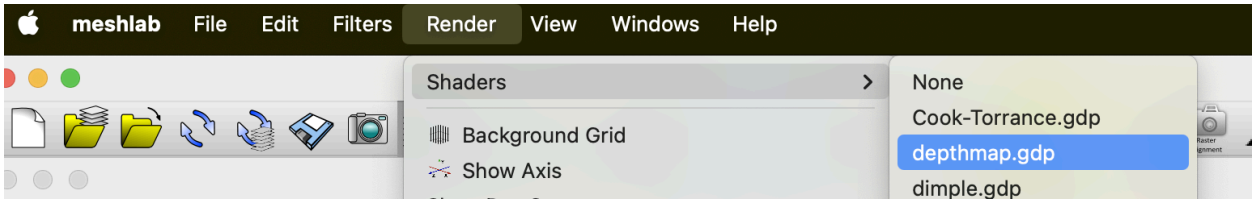


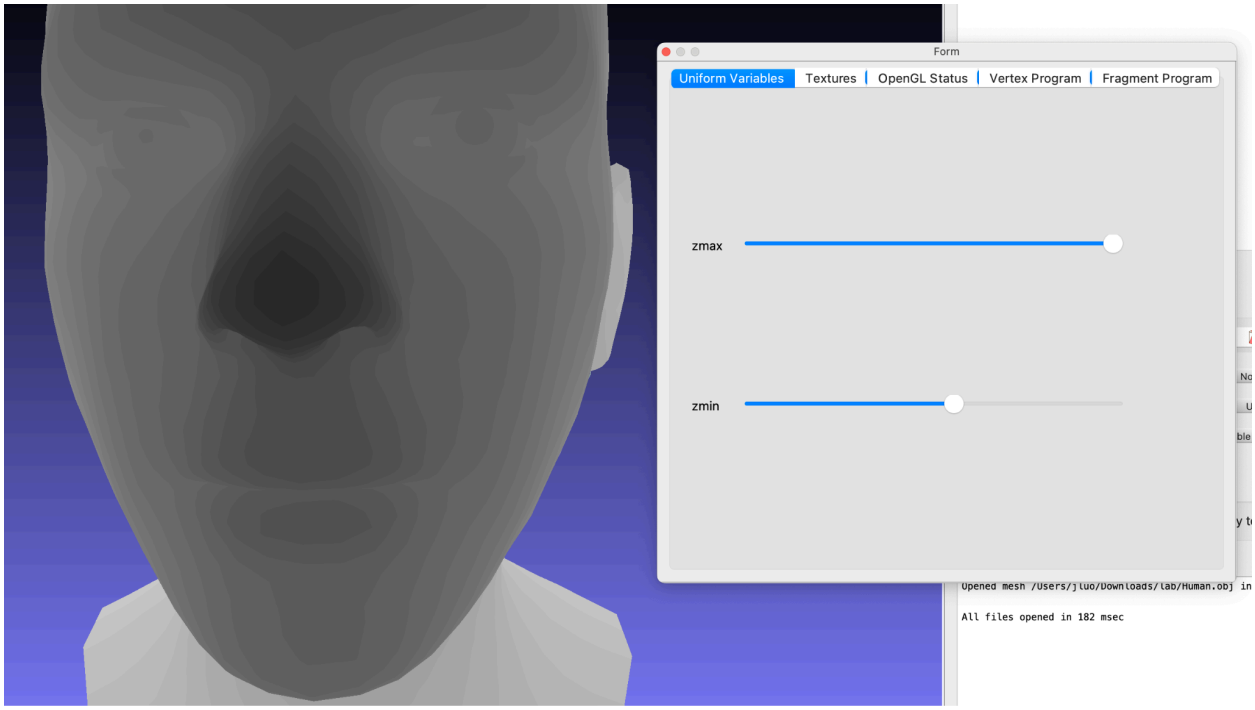


View the Model in Different Shaders

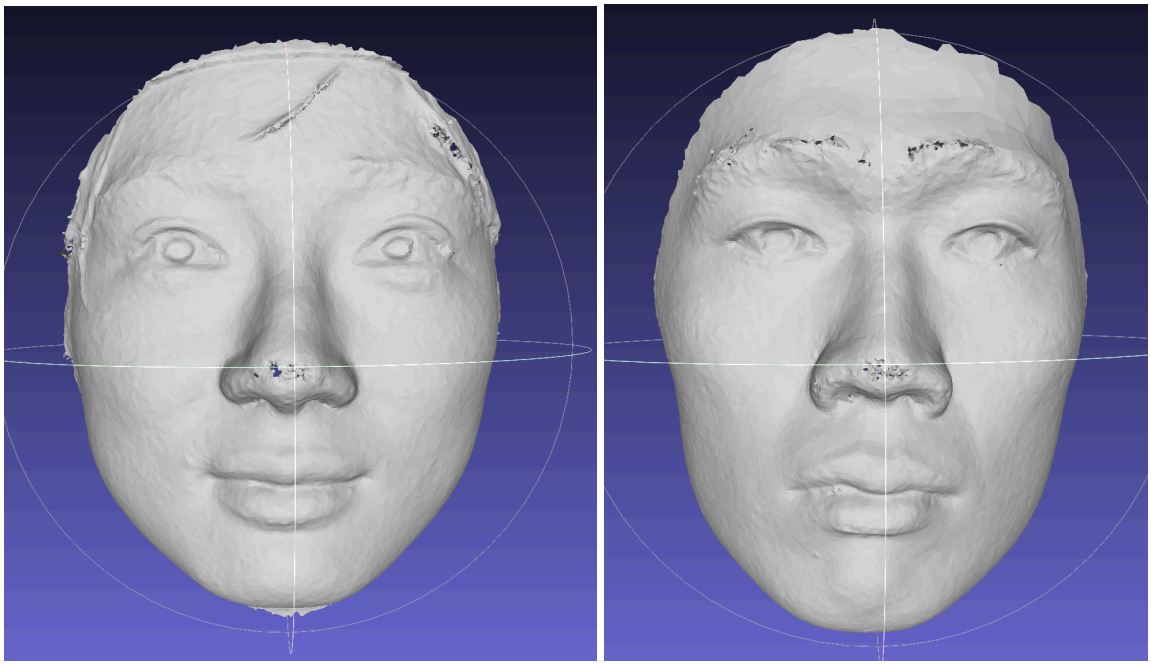
In Meshlab, the default shader when opening a mesh is roughly ambient + diffuse with direction light from the camera, which will be covered in **Lecture `'Shading/Lighting'`**.

One especially useful example is the **`depthmap.gdp` shader**, which visualizes the **distance from each point on the mesh to the camera**. In the following example of a human head model, you'll see that the **nose region appears darker**, since it's **closer to the camera**, while areas like the **ears appear lighter** as they are farther away. The gradual change in grayscale shading follows the natural **depth variation of the face**, giving you an intuitive way to understand the 3D structure at a glance.

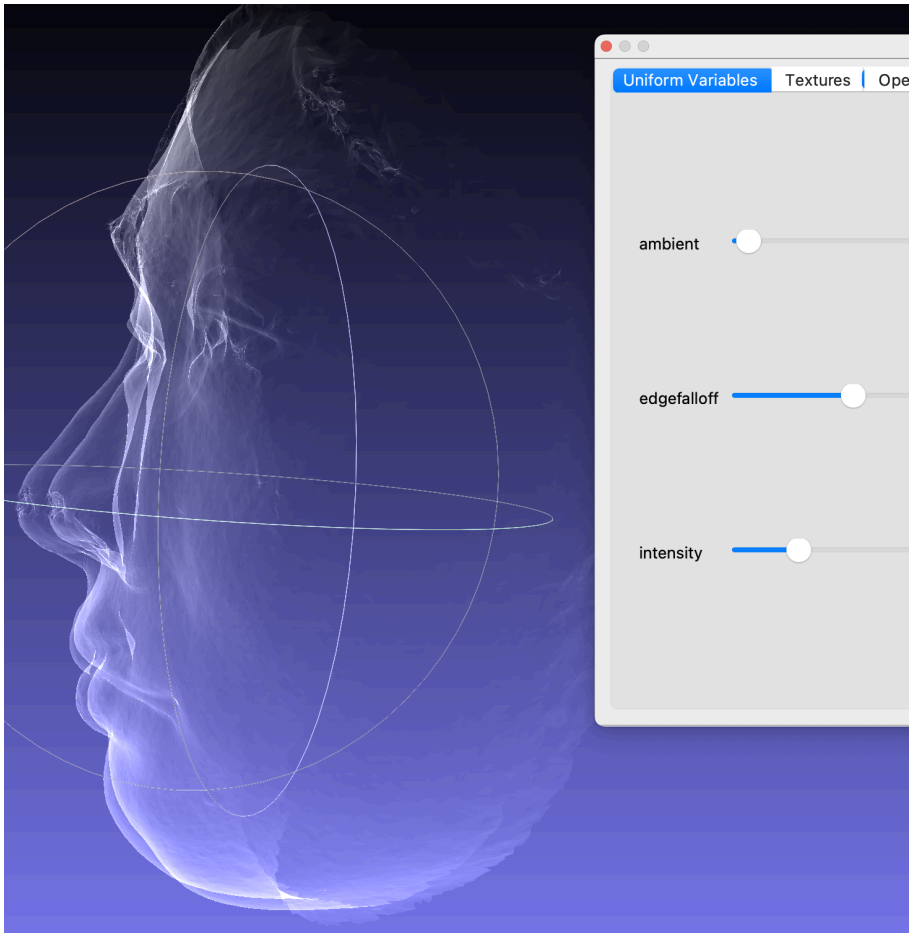




Another useful example is the **`xray.gdp` shader**, which gives the mesh a transparent, **X-ray-like appearance**. This shader is particularly helpful when you want to **compare multiple similar meshes** at once, as it allows you to see overlapping geometry and subtle differences more easily. For example, when viewing **two human face meshes with different identities**, the X-ray shader makes it easier to spot variations in shape — such as changes in nose width, jawline, or eye position — without hiding one mesh behind the other.

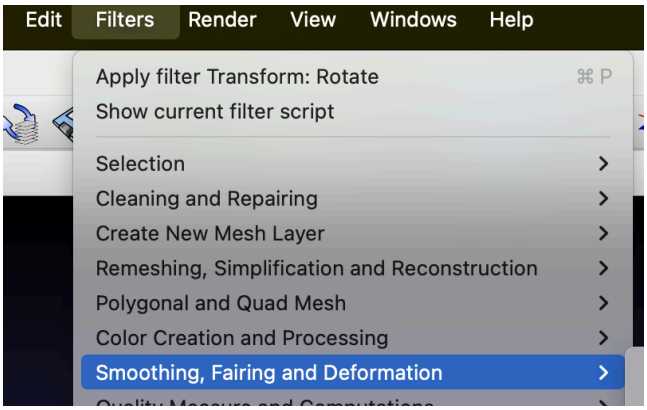


(Open `'Human1.obj'` and `'Human2.obj'` to try it out!)



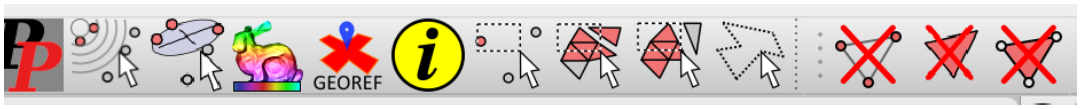
Smooth the mesh

Under `Filters → Smoothing, Fairing and Deformation` there are various built-in functions that smooth the selected mesh.



Select and remove vertices

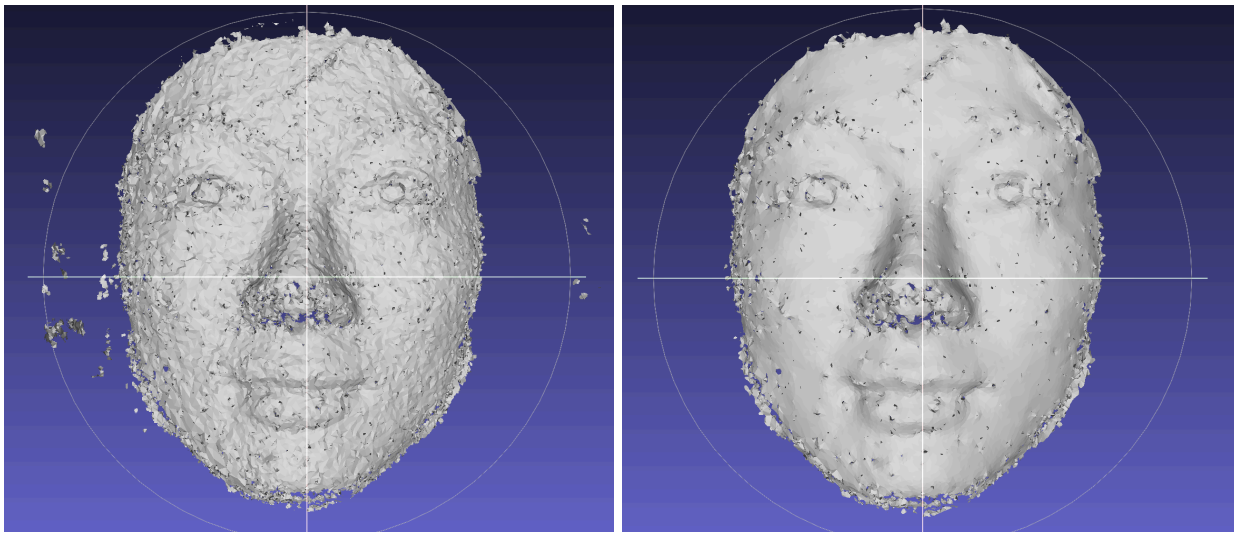
A mesh from real capture is usually noisy and not directly usable. You can select and delete vertices and triangles with the following tools on the top.



Open `'Human-noisy.obj'`. Manually remove the floating vertices with any tool above. Then apply `Filters → Smoothing, Fairing and Deformation → Taubin Smooth` multiple times to denoise the mesh as the following.

Note that like denoising a noisy image, applying **filter-based denoising** (such as Taubin smoothing) on a mesh can help reduce surface noise and irregularities — but it **does not recover the original high-quality geometry** (`'Human1.obj'`).

This is because these filters only operate on the data that's already there. They can smooth out noise, but they **don't add any new information** or guess what the original fine details were. As a result, no matter how many times you smooth the mesh, it **can't fully restore the sharp edges, fine features, or clean topology** of the original model before the noise was added.



In addition to **Taubin Smooth**, we encourage you to **experiment with other smoothing filters** available in MeshLab. Try adjusting their parameters and observe how each one affects the shape and surface of the mesh — it's a great way to develop an intuition for how different algorithms behave!

Tasks

1. **Generate and process 3 Thematically Related Meshes.**

   In this task, you'll use **Rodin** to generate a **set of 3 meshes** that are **semantically connected** — meaning they share a similar shape or concept, but differ in some meaningful way. Here are a few example themes you can explore (but feel free to come up with your own!):

   a. 🍕 Three pizzas with different toppings
   b. 🐱 Three emoji-style cats with different facial expressions
   c. 🐶 Three dogs of different breeds or styles
   d. 👕 Three hoodies with different patterns or logos

   Once you've generated your three 3D models:

   Use Meshlab to convert each mesh to use triangle surfaces (instead of quads) and generate the .mtl files. You will need to open the triangle mesh with texture in Meshlab without getting any error.

   Submit these 3 textured .obj triangle meshes. Also submit a Snapshot of 3 Meshes rendered together with xray.gdp shader with Alpha Transparent background.

2. **Downsample** each of your mesh to fewer than 5000 vertices. They don't have to have exactly the same number of vertices.

   Submit your downsampled .obj mesh. (no texture required)

3. Using the **vertex removal tool and Taubin Smooth**, denoise `'Human-noisy.obj'`.

   Submit your results as a .obj file.

What to turn in?

- a **ZIP file** containing the required files for all tasks above, ensuring that the data is clearly organized for easy identification of each task.
- a readme.txt/readme.md which lists the names of your group members.
- your group in the lab should each turn in the files individually, but it can be identical files for all group members.