Proyecto 2 - OpenGL

Piero Marini

June 3 2022

1 Repository

Repository: 3D Reconstruction GITHUB

2 Description

Implemented a basic OpenGL renderer with diffuse, ambient and specular lighting. We can use the keyboard to visualize the point cloud (P), wireframe mode (I), mesh rendering (M), render lights (L) and re-generate the point cloud and triangulation (R). We have a free camera system that allows us to move with W, A, S, D and use the mouse movement to adjust the camera view.
The program can generate a random point cloud by uniformly sampling a parametric cylinder surface. We use this as a basis for the creation of a ”tunnel”. The equation is as follows:

$$f(x) = \begin{cases} \text{x=radius*cos } \theta, & \theta \in (0, 2\pi) \quad (1) \\ \text{y=radius*sin } \theta, & \theta \in (0, 2\pi) \quad (2) \\ \text{z=h}, & h \in (-height/2, height/2) \quad (3) \end{cases}$$

We then implemented the **3D Delaunay Triangulation** that takes the point cloud and outputs a lit of ordered triangles, which can then be sent to the renderer to be drawn onto the screen. Unfortunately, Delaunay Triangulation doesn't handle this type of triangulation correctly, and the results are quite horrible as can be seen in Figure 3.

We then implemented the **Ball-Pivoting Algorithm**, which gives us much better results as can be seen in Figure 5. There are still some holes on the final mesh which we could improve in the future. In Figure 6 we can look a bit more at the work the algorithm did to generate our mesh.

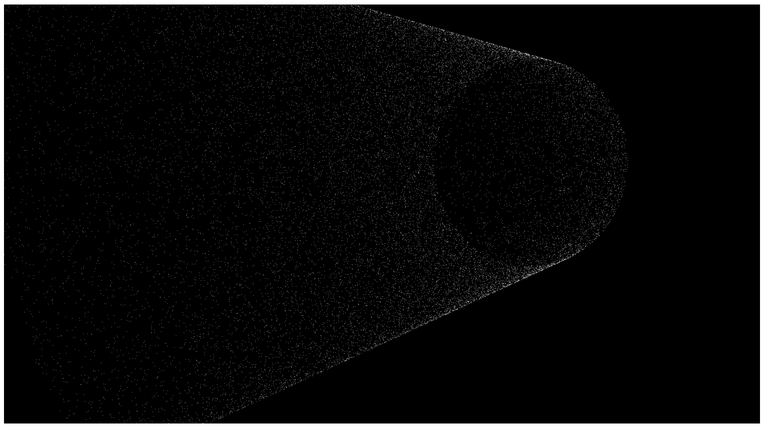

Figure 1: Point Cloud sampling a Cylinder

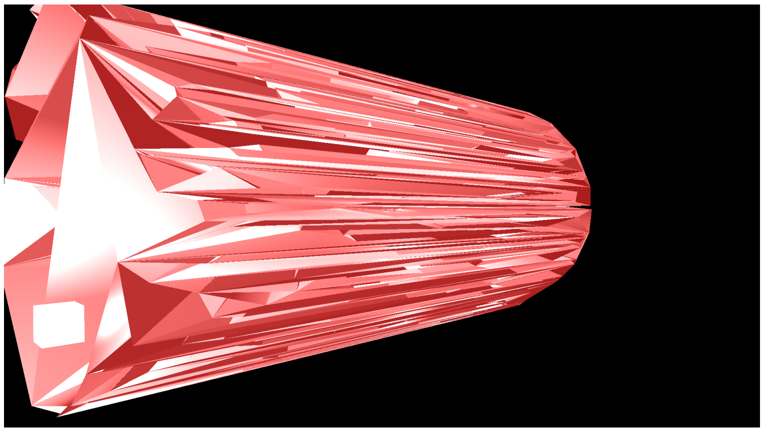

Figure 2: Delaunay Triangulation Results

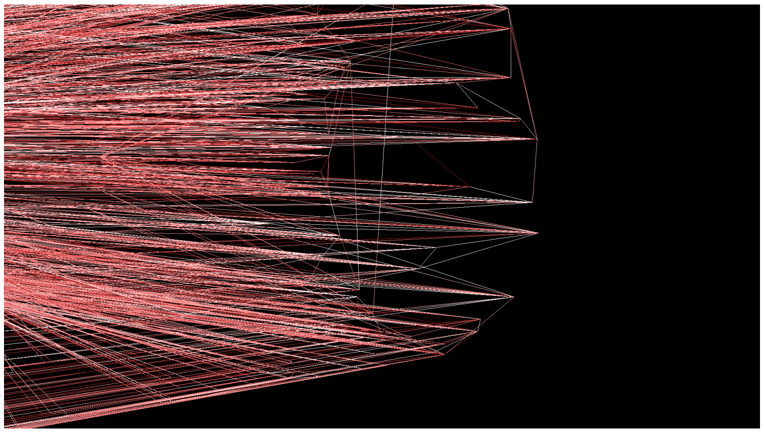

Figure 3: Delaunay Triangulation Wireframe

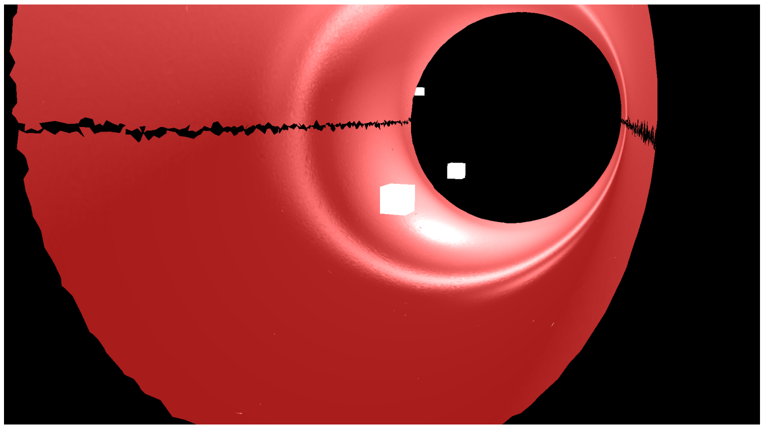

Figure 4: Ball-Pivoting Results

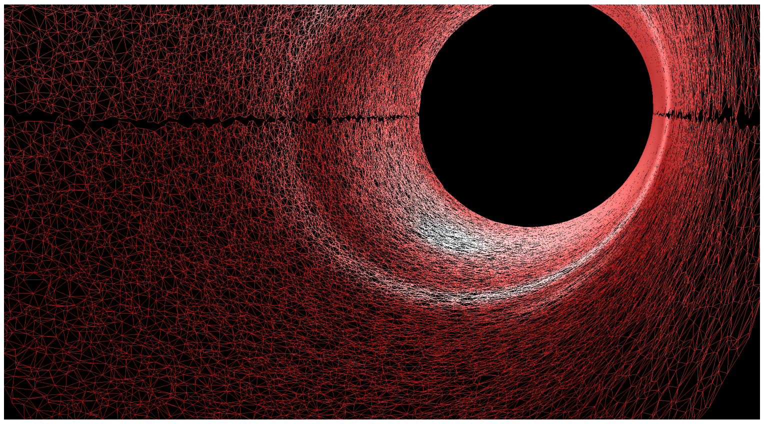

Figure 5: Ball-Pivoting Wireframe