Assignment 1

Kaustubh Nair kaustubh.nair@iiitb.org
IMT2017025

1. Model loading

The scene consists of three mesh objects, loaded from the given dataset (Fig. 1). A custom parser was written to load the vertices and indices of the faces from the ply files. The parser uses min-max normalization to scale the vertices between -0.5 and 0.5. The meshes can be rendered with two options: wireframe mode and filled mode.(Fig. 2) The mode can be toggled with the w key.

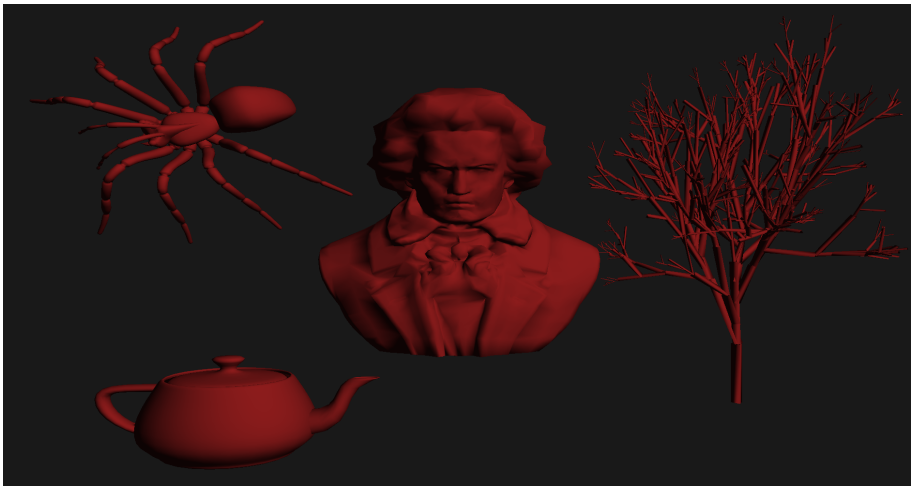

Figure 1: 3D Scene

2. Transformations

Basic transformations can be applied to the objects in the scene. Objects can be selected using the number keys. Once an object is selected, any of the following transformations can be applied:

- Scale - The + or - keys can be used for increasing or decreasing the size of the object. The scaling matrix was generated using $glm :: scale()$ and multiplied to the position vector.

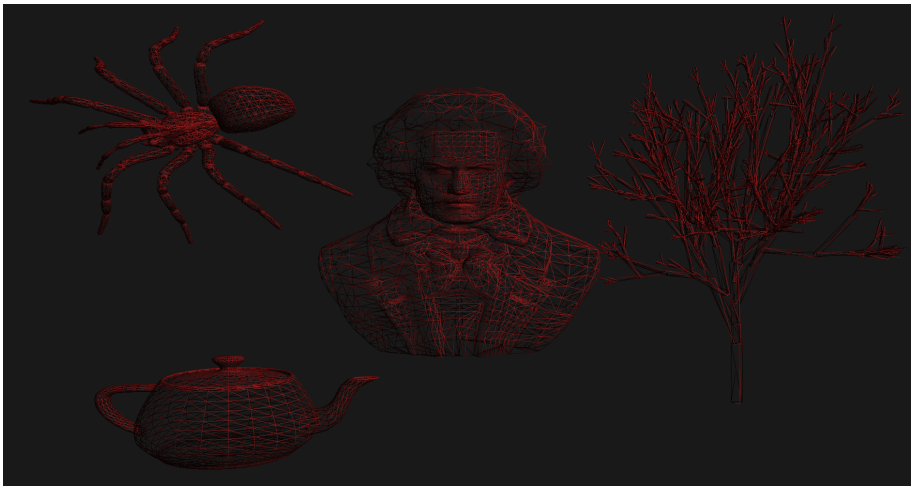

Figure 2: Wireframe mode

- Translate - The left mouse button can used to move an object around the scene. The translation matrix was generated using $glm :: translate()$ and multiplied to the position vector.
- Rotate - The right mouse button can be used to rotate an object. Rotation has been implemented using quaternions.

3. Normal Coloring

The vertices can be colored based on the normal at the given vertex. (Fig. 3 This can be toggled with the n key. The normal for each vertex was computed by taking the average of the normals of the faces at that vertex. This was then normalized and the vertex color was assigned to the normal vector.

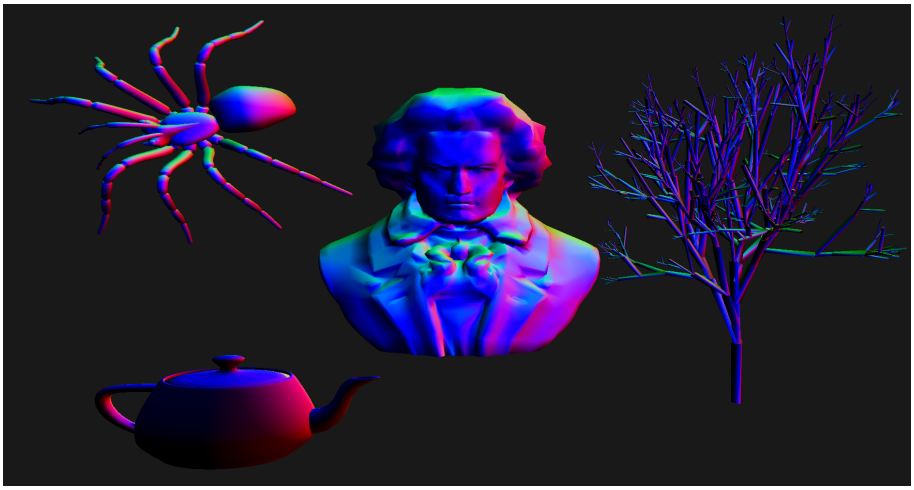

Figure 3: Normal coloring

4. Splats

Splats are incircles rendered in the faces. Splats can be toggled using the s key. The inradius can be modified by j or k keys (Fig. 5. The splats were implemented as follows:

1 During parsing of the ply file, compute the incenter, normal and inradius of each face and store it in an array.

2 Pick any vector, which is not the origin,from the plane perpendicular to the normal. For example, if the position vector of the normal is (x, y, z), a perpendicular vector could be $(0, -z, y)$. Translate the incenter by distance equal to the inradius in the direction of this vector. This gives a point on the circumference of the incircle.

3 Rotate the obtained point by a desired angle and obtain multiple points on the circumference of the circle.

4 Store all the points in an array which will be rendered in the scene.

5 Use the GL_TRIANGLE_FAN primitive to render the points

Though splats work for most meshes, they cannot be used to model objects that have faces with irregularly sized triangles. If one side of a triangle is too small as compared to the others, like for example a tree trunk (Fig. 6), the tree trunks are too small. For such cases, it would be more appropriate to render steiner inellipses.

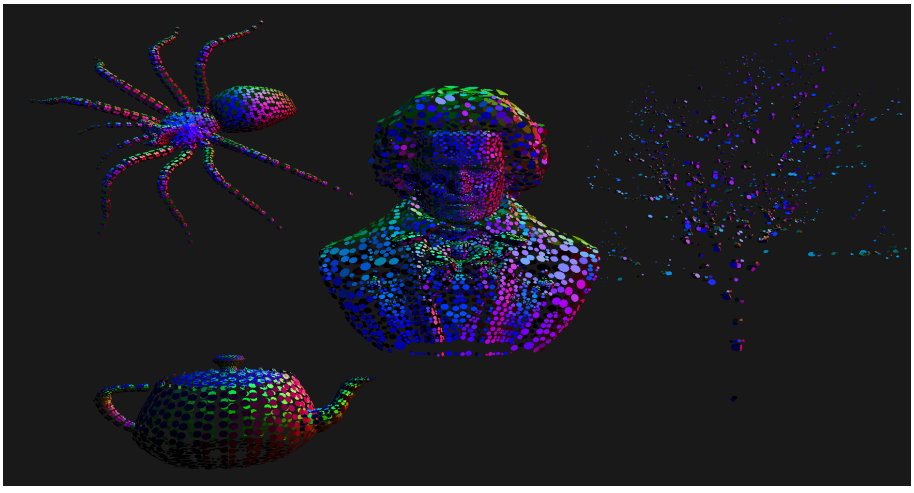

Figure 4: Splats

5. Answers to questions

1. Does default lighting work or did you have to add at least one light for 3D rendering? The scene uses both ambient lighting and one diffuse light source. Since normals were already computed, adding basic diffuse lighting was quite easy.

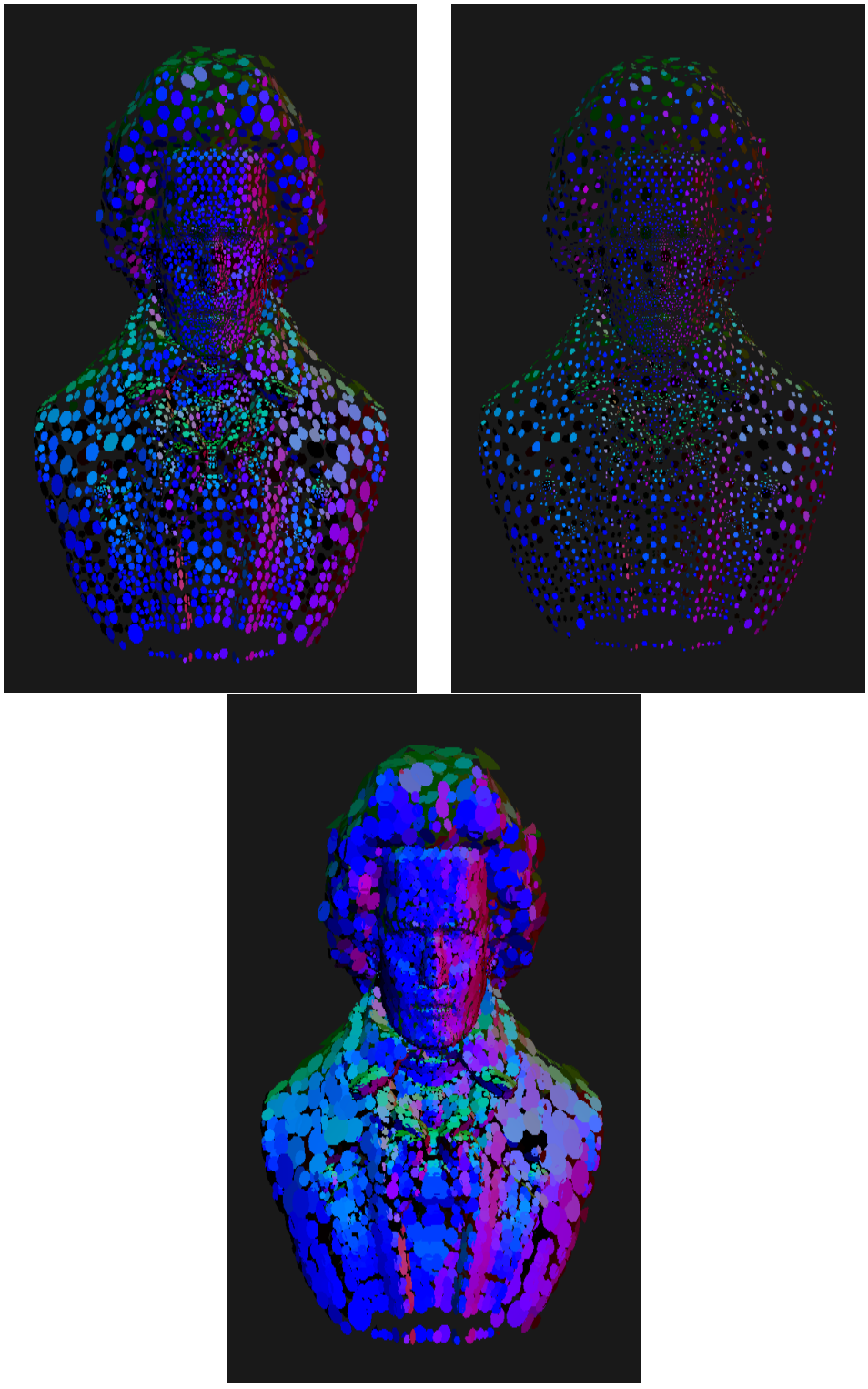

Figure 5: Splats with different radii

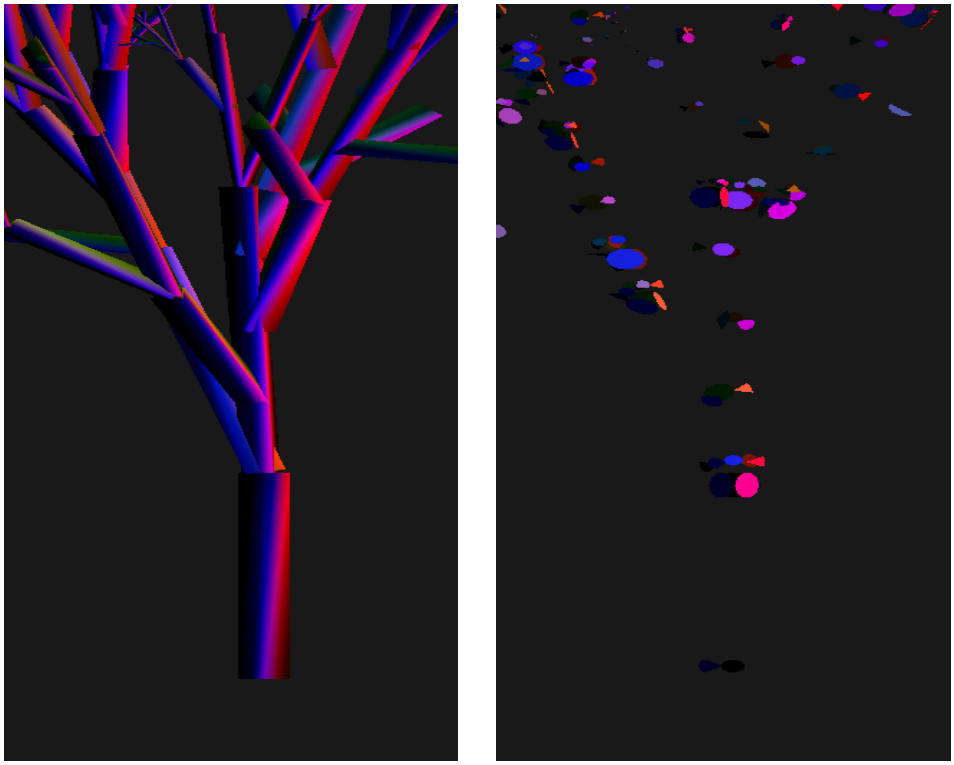

Figure 6: Splats on faces with disproportionate sides

2. What is your choice of user interactions for rotation, translation, scaling in your graphical application?
   Discussed in Transformations

3. How did you implement circular splats? If applicable, how did you implement elliptical splats?
   Discussed in Splats

4. If the normal vector is not normalized at the faces, or the average normal vector at the vertex, what do you anticipate will happen to lighting?
   It would lead to unexpected lighting conditions, like for example it would look similar to ambient lighting.

5. What other properties of the vertex can you capture using its color?
   We can specify properties like transparency using it's color.