Assignment 2 Report

Shobhit Behl (IMT2016024)

- 3D Rendering Using MVC Architecture - Problem Statement

Render a 3D scene using more than two objects in the scene. The objects will be rendered using their surface. Meshes given in .ply file format.

Datasets are available at: https://people.sc.fsu.edu/ jburkardt/data/ply/ply.html.
Larger resolution meshes are available at: http://visionair.ge.imati.cnr.it/ontologies/shapes/viewmodels.jsp

1. Write a parser for the ply files and define the mesh as a model in your implementation. Your scene must contain multiple models which could be from different ply files. Each object can be normalized to be contained inside the unit cube (i.e. x,y,z range of the cube is in range [0,1]). Each mesh must be rendered with the following options (one at a time): (a) wireframe mesh, where the triangles are not filled; (b) filled mesh, where all the triangles are filled with one single colour.

2. Implement object-centric affine transformations, like translation, rotation, scaling. Associate keyboard and mouse inputs for the transformations, where translation and rotation are governed by mouse inputs. e.g. left mouse button may be used for translation and right for rotation; + and - keys may be used for scaling. The mouse button inputs for both press down, drag and release can be used for covering all aspects of translation and rotation. In order to pick an object in the scene, a combination of keyboard and mouse button inputs must be used. The rotations must be implemented using quaternions.

3. Compute vertex normals by averaging the normals of the triangles sharing the vertex. The vertex normals must be unit vectors. Color the vertex using the normal vector, where (R,G,B) correspond to (x,y,z)- coordinates of the normal vector.

4. Implement a splat for each triangle in the mesh, where a splat is a filled circle contained in the triangle, placed at the in-circle of the triangle in the mesh. The circle must be coloured using the normal coordinates, (R,G,B) corresponding to the (x-,y-,z-) coordinates of the normal vector. [There can be different display options for the mesh wireframe , filled triangles, just vertices without the edges, and as splats.]

5. Implement lighting using a point light source. Draw the light source as a relatively small sphere or using GL_POINTS in the scene. Enable lighting as well the point light source, and using appropriate mouse and keyboard actions, move the light.

6. Depending on your code design, the mouse and keyboard interactions and feedback must be handled between a view or a controller object. The code for the scene display must be in the view object.

- Design

1.) *Parser Class* - This class is used to read ply files and return the number of faces, number of vertices, the vertices, the element index array. and certain information pertaining to selection logic.

2.) *Shader Class* - This is used to create, add and compile shaders and get an identifier which can be used to specify the shader programs to be used.

3.) *Vertex Class* - This class is used to store various attributes of each vertex which is used in the shaders i.e. position, colour, normal etc.

4.) *Model Class* - This class represents a geometric model, i.e. a mesh. It has an array of vertices, an array of indexes, transformation matrices, vertex array identifier, vertex buffer identifier and an index buffer identifier. It also has extra information pertaining to selection of the model to check and store whether the model has been clicked or not.
It has a construct method which takes in a file name, creates a parser object and reads the file to get the vertices, indexes etc.
It also does the calculation of normals, splatting and the displaying of the model on the window.
There are methods to handle any user input and perform transformations by modifying the transformation matrices.

5.) *Scene Class* - This class corresponds to the Model in MVC, it has an array of geometric models, methods to add models and perform transformations in the models. It also has the coordinates of the light source.

6.) *View Class*- This class corresponds to View in MVC, it handles the creation of windows and initialization of various OpenGL requirements.

7.) *Controller Class* - This class corresponds to the Controller in MVC, it has a view and a scene object. It handles any interaction between the user and the View/Scene, displaying of the scene and also creates a Shader object to compile shader programs.

- Logic

1.) *Object Selection* - For selection, I am storing the minimum and maximum of x, y and z coordinates of each geometric model. Upon a click, we first get the corresponding z coordinate of the click from the z buffer, apply inverse transformations of the model to be checked for on these coordinates, and then check if the x,y and z coordinates of the click lie in between our ranges.

2.) *Translation* - For the selected model, translate by the difference in the old coordinates and the new mouse coordinates, and then update the old mouse coordinates.

3.) *Scaling* - Scaling is relatively simple, just scale by 1.1, or 0.9 depending upon which key is pressed.

4.) *Rotation* - For rotation, we use old mouse coordinates and new mouse coordinates, we project both onto a hemisphere using x and y coordinates which gives us corresponding p1 and p2. Now, our axis of rotation is $p1 \times p2$ and the angle can be varied according to preference.

5.) *Splatting* - First I take the vertices and index array of a circle of radius 1 centered at $(0, 0, 0)$. Now for each face in my geometric model, I orient this circle along the face by calculating the required transformations and that will be my splat for the face. It's normals will be the same as the face normal.

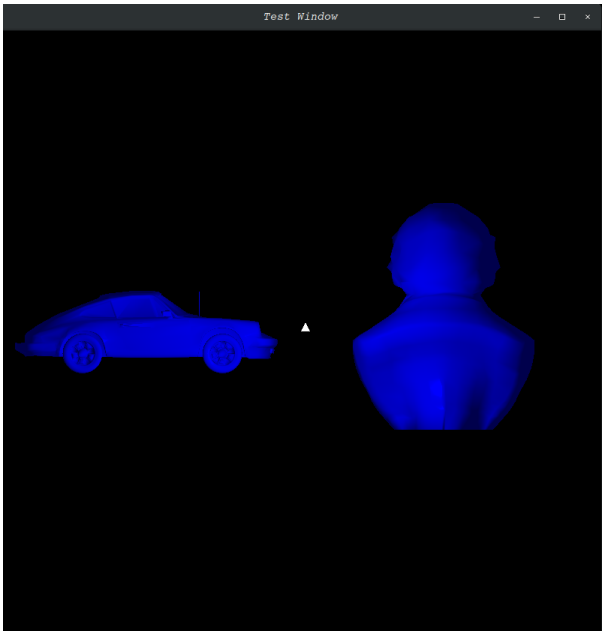
- Questions

1.) To render multiple copies of the same element, we don't need to use multiple buffers for each instance, we can just have different transformation matrices for each object while using the same data after reading the ply file once and storing it in the vertex and index buffer.

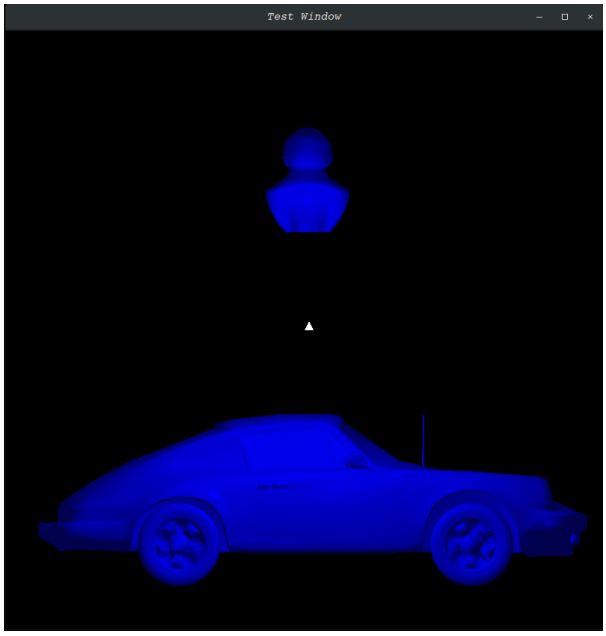
2.) We can visually see if our normals are correct by assigning the normal as the object colour, we can take a basic model for which we know what the normals should be and verify if the colour matches.

3.) We can align the center of each circle with the vertex and take the radius as the half the minimum distance of the vertex with other vertices, this way there shall be no overlaps.

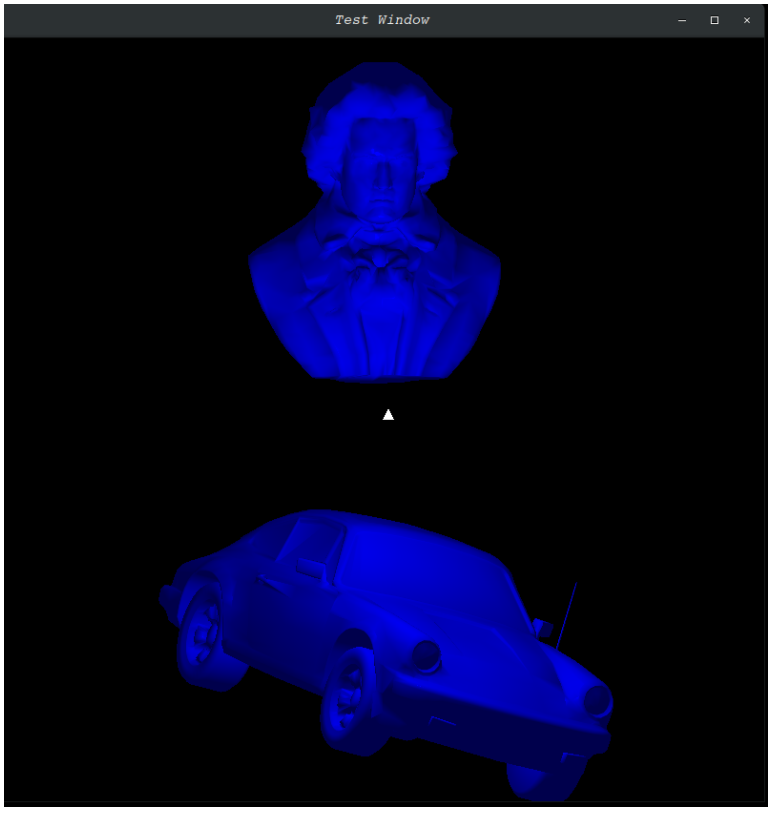
- Images



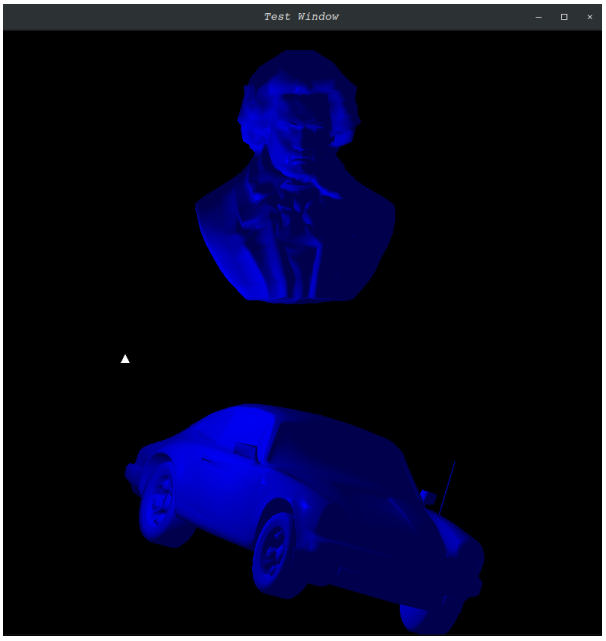
Normal Display



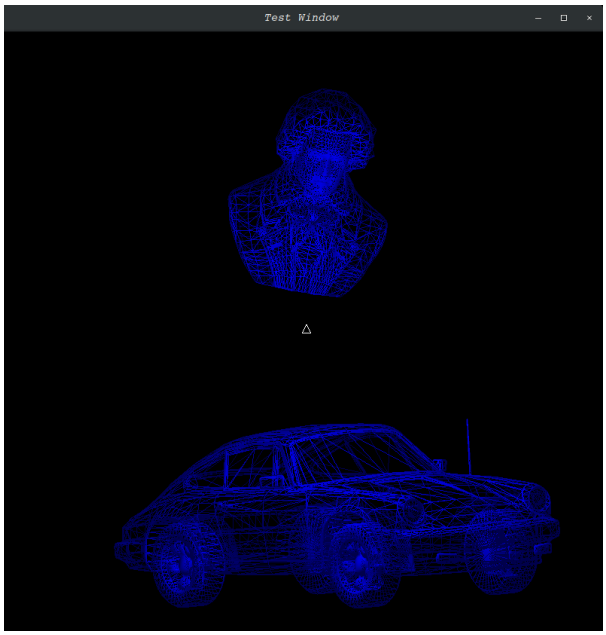
Scale And Translate



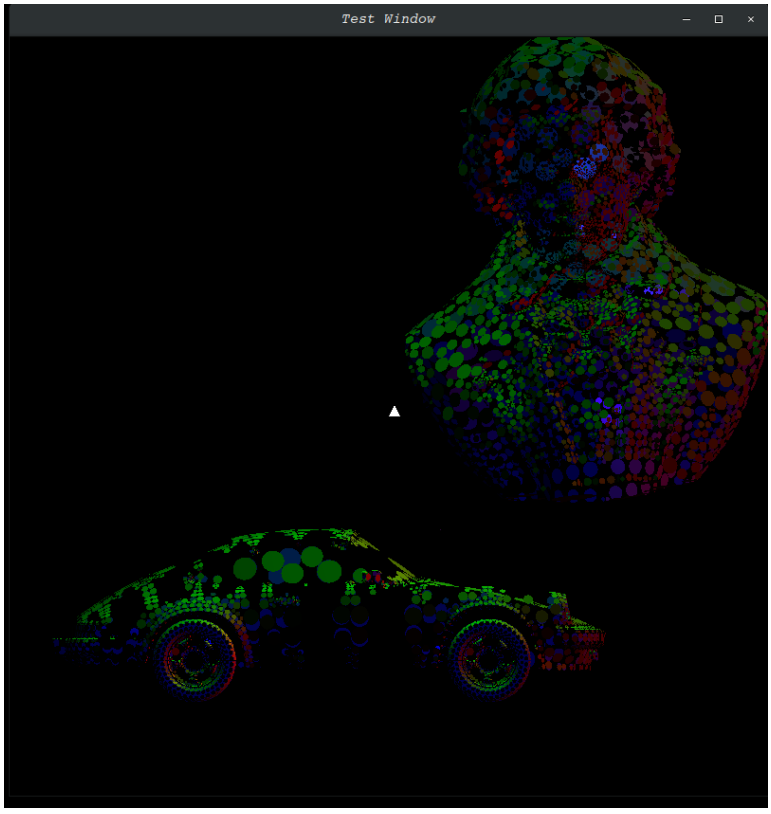
Rotate



Movement In Light



Wireframe



Splatting And Normal Colour