INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY BANGALORE

Assignment 2
3D Rendering Using MVC Architecture

Vaibhav Kansagara
March 17, 2019

1 PROBLEM STATEMENT

Create a 2D graphical application for rendering a touch-screen calculator.

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 incircle 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 like 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.

2 BRIEF EXPLANATION OF THE CLASSES AND DESIGN(MVC ARCHITECTURE)

Point - stores the x,y,z coordinates of the point.

Color - stores the r,g,b values of the color

Parser - parses the file format(.ply) and returns the model object.

Model - class representing the model object.

Controller - this class handles all the mouse and keyboad related events and takes appropriate action. Additionally this class has information about all the model objects and in case of user action it determines which model object is clicked on and calls appropriate methods on the selected model.

View - this class has a display method which handles how the model is to displayed to the user.

Shader - this class handles the compiling and linking of vertex and fragment shaders and also handles passing uniforms to the shader.

3 APPROACH

1. Approach for task1: Create the parser object and load the model object. Code is in Parser.cpp file.
2. Approach for task2: View object stores the view and projection matrix.Here the view matrix is identity matrix and the camera is at origin pointing towards the negative z-direction.Projection matrix is also identity(orthographic projection).
    a) First we need to identify where the cursor is placed and convert the screen coordinates to window coordinates. I am enclosing each model inside a cuboid and checking if the clicked point is inside the cuboid or not. After identifying which model is selected the difference of the cursor positions will be the displacement vector and i am translating the model to that displacement vector.
    b) For scaling just use glm::scale.

c) For rotation we rotate the selected model with axis as cross product of the final and the old position and with angle between those two vectors.

3. Approach for task3: We first calculate the normals of all the triangles and then use adjacency list to calculate the average of all the normals in which a particular vertex is associated and pass that information as color of that vertex.

4. Approach for task4: For this task we first need to calculate the vertices of the circle and divide into many triangles. We then need to rotate that triangle with axis as cross product of triangle normal and circle normal with angle as angle between the two normals and then translate it to the incentre of the triangle and scale it to the inradius of the triangle.

5. Approach for task5: For this task we need to create lighting vertex and fragment shader. We use sphere as light source scaled to a very small size so that it looks like a point object.Lighting gives us the perception that the object is 3-D also provides depth to the model object.We add ambient,diffuse and speculative components and multiply that with the fragment color to give the output color of that particualar fragment.

4 How would you optimize your code to minimize I/O operations and subsequent storage requirements to render multiple copies of the same object, as shown in Figure 3?

For drawing multiple copies of the same object we use a concept called instancing. It is technique in which we draw multiple objects with a single render call saving us all the CPU -> GPU communications each time we need to render an object. This way we only need to send the data to the shader once and specify the GPU of how those instances need to be rendered. Thus GPU renders them without having to communicate to the CPU continuously. We need to change the render calls from **glDrawArrays and glDrawElements** to **glDrawArraysInstanced and glDrawElementsInstanced** respectively. This calls take an extra parameter as number of instances you want to draw.We then need to set the attributes of position and color of each instance for which we need to tell opengl to advance to the next value not for each vertex but for each instance. We use **glVertexAttribDivisor**() for doing that.

5 How would you test if your lighting is working correctly

For ambient light we can test if the color of all vertices are same because it doesn't depend on the position of the vertices. Note that the model vertices should have the same color. For diffuse lighting we can check by varying the angle between the normal and directed light ray. Lesser the angle more bright it should appear and vice versa. For specular lighting we can apply the same technique,vary the angle and try seeing it from different viewpoints.

6 USUALLY SPLATS ARE IMPLEMENTED ON VERTICES AND NOT SURFACE ELEMENTS. NOW THAT YOU HAVE IMPLEMENTED IT FOR A SURFACE ELEMENT, HOW WILL YOU IMPLEMENT THE SAME FOR A VERTEX

In this case the radius of the circle will be half of the minimum distance of a particular vertex with all it's neighbours with centre as the vertex coordinate.everything else remains the same.