

CSE 5542 LAB ASSIGNMENT 1

- **How to build and run the code:**

- No external libraries apart from DEVIL is used.
- Corresponding “include” directories and libraries should be added to the project properties.
- Use the DemoFramework.sln file to run the program. Make sure all the dll files are available in the Debug folder

- a. **Pan, Dolly:**

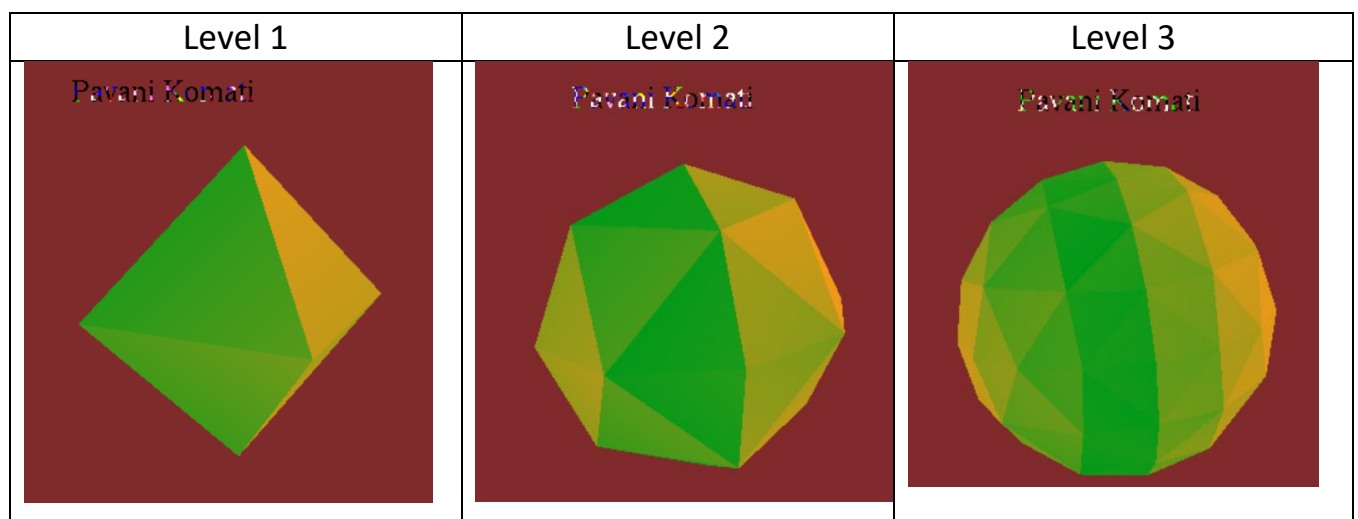
Pan: examinerTransformNode is translated in the corresponding direction of the key pressed.

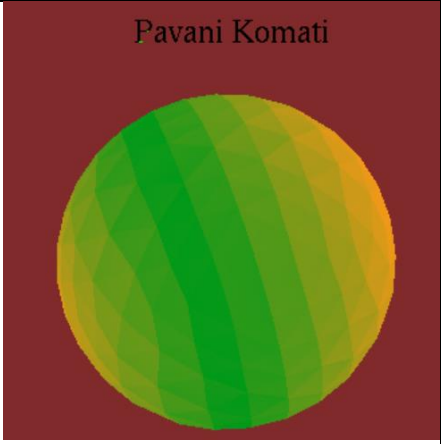
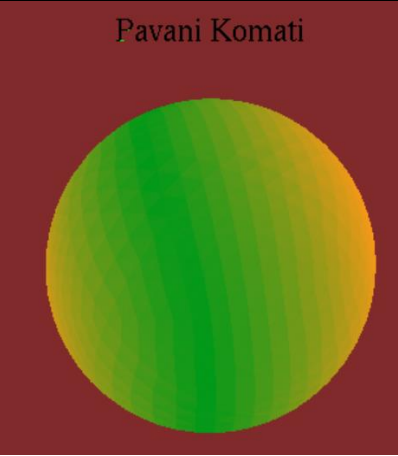
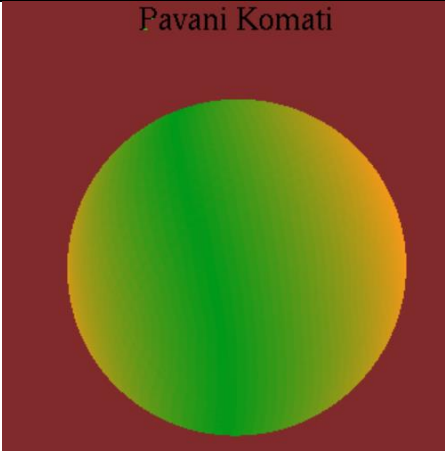
Dolly: dollyNode is translated with a distance of dollyDelta. The scene is moved forward when ‘Z’ is pressed and backward when ‘z’ is pressed. (Given dollyDelta =0.1) dollyNode corresponds to perspective matrix and scene changes can be observed .

Trackball: These operations are handled by TrackBall class. When the scene is rotated, start and end positions are sent to the class. When the mouse is in moving state, manipulations are performed in the trackball class. Rotation around the axis perpendicular to the mouse movement is done from the last point to current point.

- b. **Sphere Tessellation:**

Two pyramids joined together with 6 faces, is tessellated to form a sphere of given radius. Each face is internally divided into 4 triangles by taking midpoints of three sides. This is performed multiple times on every triangle formed for required number of levels. While pushing the vertices of a triangle, face normals are computed and passed as vbo.



Level 4	Level 5	Level 6
		

Above are the tessellations at different levels on the pyramids.

c. DrawableCube VBO:

Cube is constructed by taking 16 vertices, 4 for each side. Indices array is used to create all the required triangles (12). The vertices, normal and colors are passed as DrawElements. Since normal for each face has to be mentioned, we need to consider 16 vertices for indexing. Without normal indexing can be done using 8 vertices also.

d,e. OpenGL 3.3 complaint and matrix operations:

MatrixManager class is created to handle all the push and pop operations. The class consists of two stacks, modelViewStack and projectionStack. Where ever `gl_enable(MODEL_VIEW)` is used, the matrices are pushed in to modelView Stack and `gl_enable(PROJECTION)` is mentioned, matrices are pushed into projectionStack. All the gl functions such as `gl->Lookat`, `gl->perspective`, etc functions are created and implemented in MatrixManager.h similar way as openGL works. The Stored matrices are used onApply of transform matrix node and Model, View, Projection matrices are passed to the shader.

f. Lightening

A Vector List is maintained in LightManager Class to save the list of lights and its properties given in scene.xml. These populated lights along with the material properties are sent to the shaders in onApply of ShadedMaterial.h

Lightening.vert and Lightening.frag files are the corresponding shaders for Lightening application. The fragment shader contains implementation for directional lights as well as point lights.

g. Additional shaders for lightening

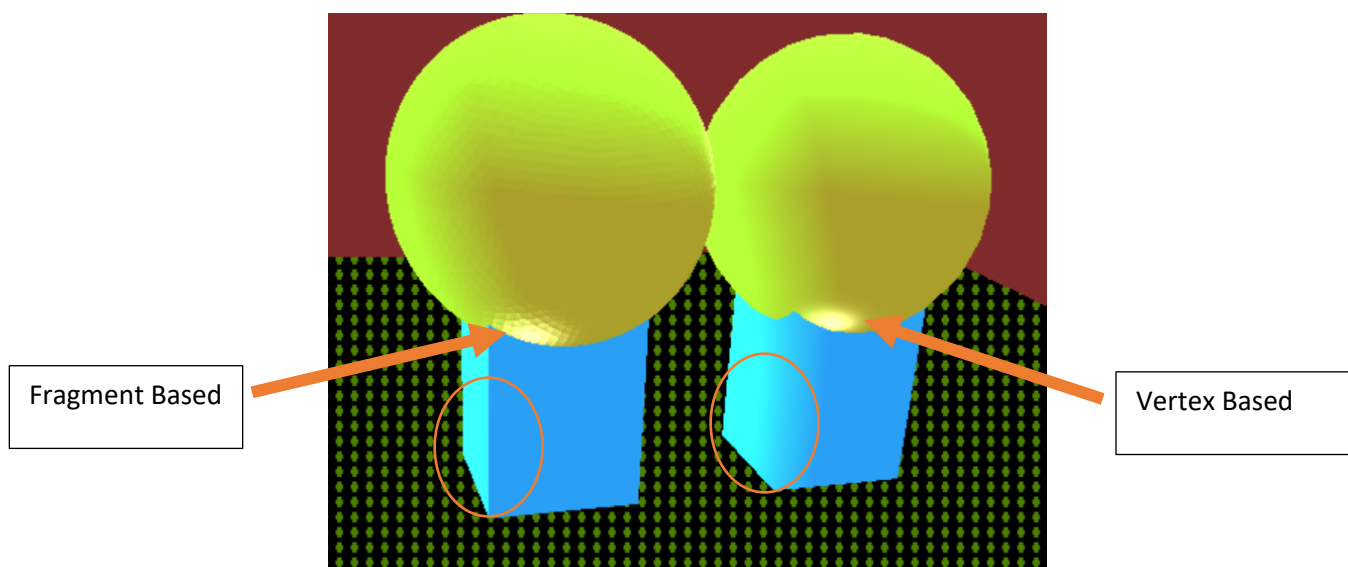


Silhoutte Shading: This shading is done by taking the angle between the surface normal and view direction. If surface normal is perpendicular to the viewing angle, red component of the sphere is changed accordingly to get an effect at the corners of the model.

Fragment Based Shading: This shading is performed by sending the fragment normal of a surface to the vertex shader.

Two Color Hemisphere: The surface color is computed based on the angle between the surface normal and the shading direction given. Depending on the angle between these two, color of the surface is decided.

h. & 4 :PerVertex and PerFragment lightening difference:



Fragment based lightening is performed by sending the surface normal of cube and sphere. Vertex based lightening is performed by sending the vertex normal to the shaders.

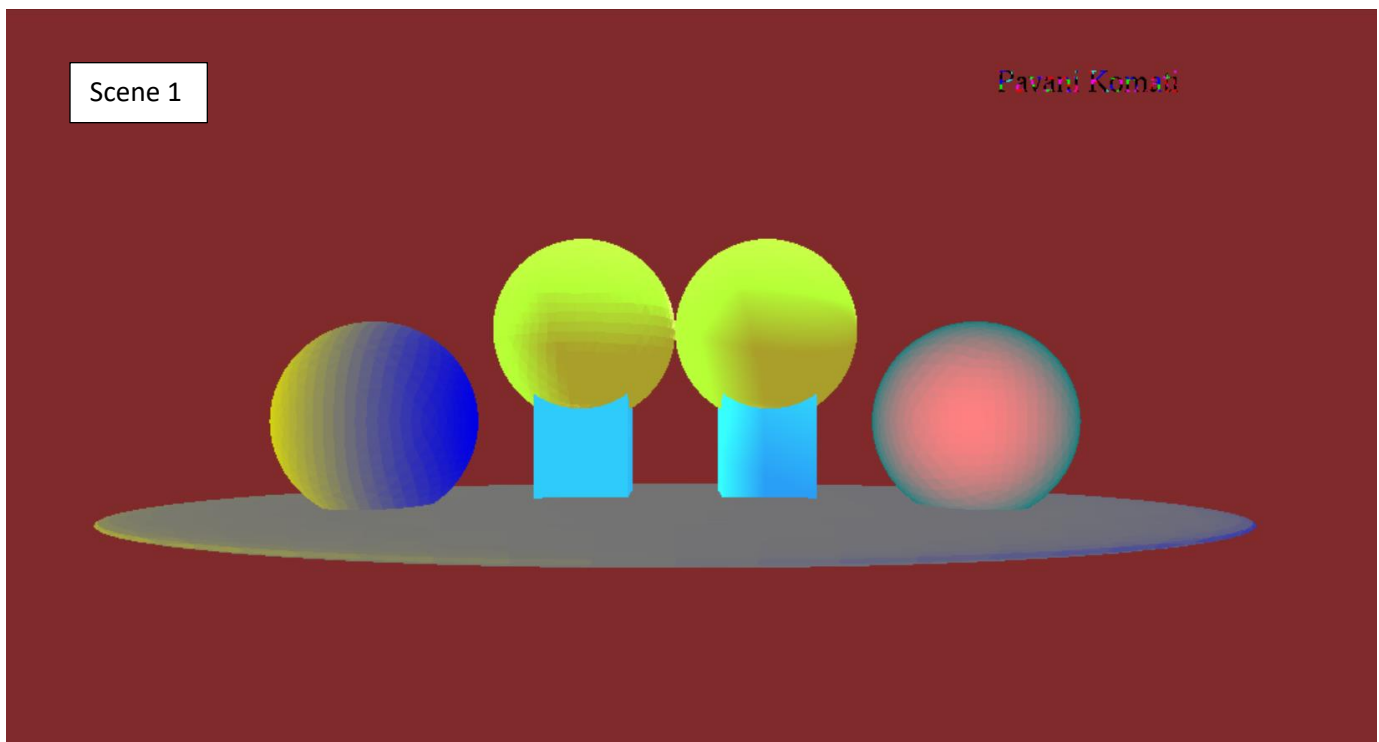
Here we can clearly observe at the arrows pointed that vertex based lightening concentrates at the position unlike the fragment based where it is equally spread over.

Also, at the circles marked we can see the edge difference. Vertex normal don't really give real depiction of a model.

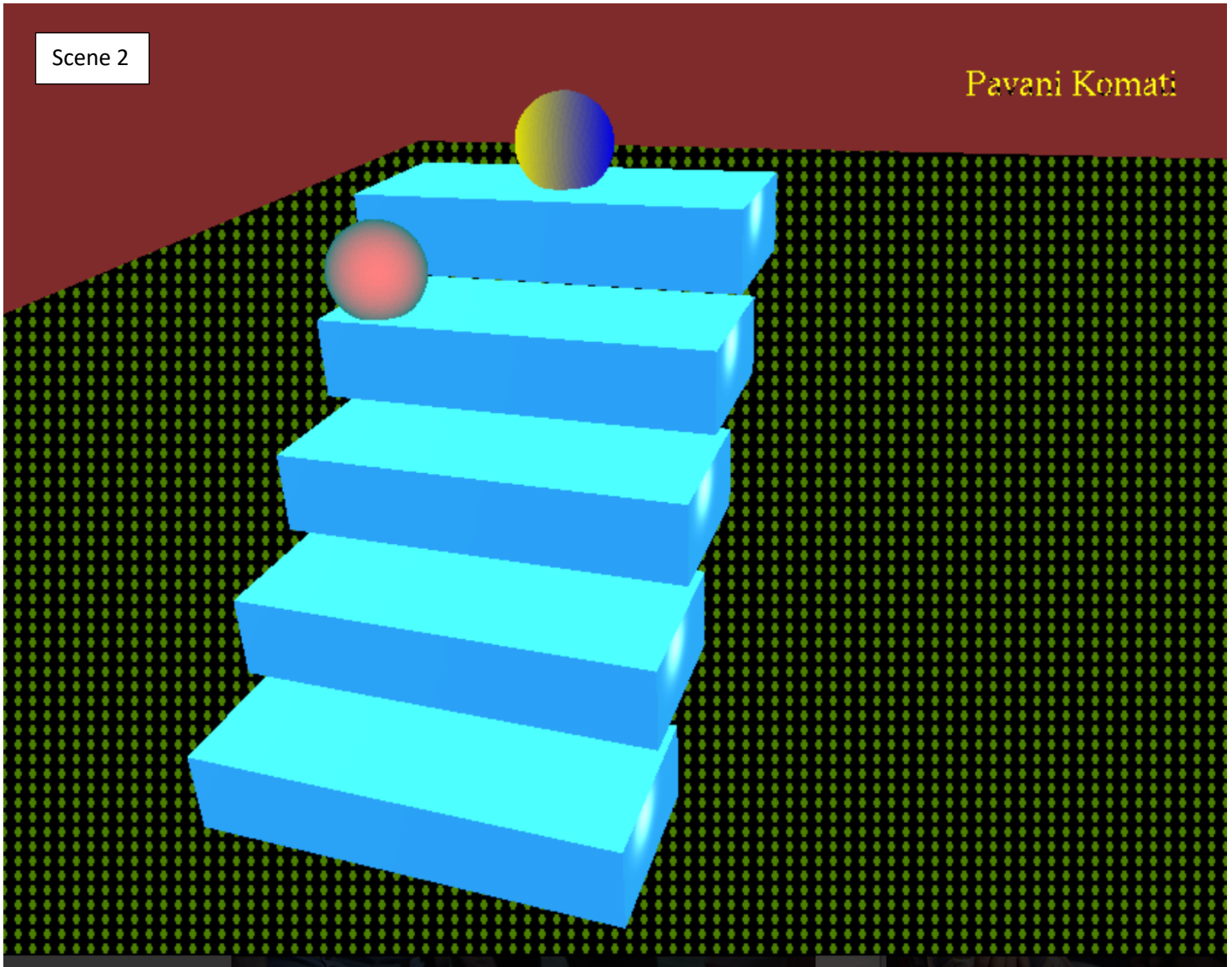
2. Backdrops: Floor of the view is desgined by creating ellipses.



3. scenes



Scene1 shows set of spheres on a plate using all the tasks mentioned in assignment. This is present in "TeapotTrophy.xml".



Scene2: Stairs are created using cubes, spheres with a designed floor. This scene is available in "Stairs.xml".

`uncomment //const std::string sceneFile("../Media/Stairs.xml");` (line 68) in `demoframework.cpp` to see this view.

4.

-Pavani Komati
(komati.1)