Assignment – A1
Report

NAME: ANURAG B
ROLL: SMT2018004

LIGHTING:

Default lighting work or did you have to add at least one light for 3D rendering?

By default, In OpenGL lighting color is white and from Z – Direction. Which makes a mesh looks darker.

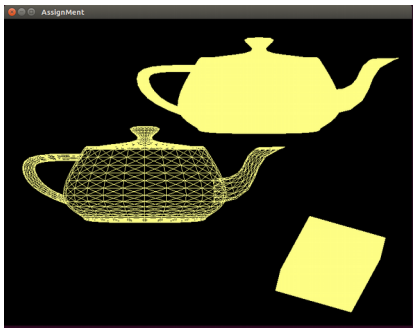


Fig. 1: No Lighting

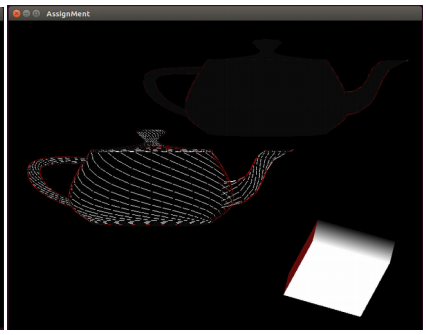


Fig. 2: Default Lighting

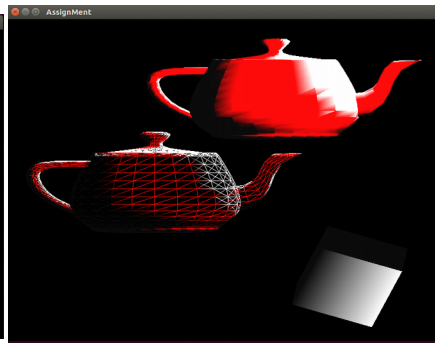


Fig. 3: Lighting with Position

Default Lighting: To enable default Lighting in opengl we can use below two APIs

```
glEnable(GL_LIGHT0);
glEnable(GL_LIGHTING);
```

Code Snippet

By using glEnable(GL_LIGHT0) and glEnable(GL_LIGTHING) APIs lighting gets enabled and result we get as show in Fig. 2. By Default, Light color is white and comming from Z – Direction. When Lighting is enable it needs normal vector to detect color of pixel. If Lighting is disabled color is determined using glColor API which was set to yellow as Fig. 1.

```
glColor4f(1, 1, 0.5f, 1.0f);
```

Code Snippet

Now, Enable Lighting and define light source position and mesh object material properties. As said earlier for Lighting to work in opengl we need to provide normal vector (Polygon/face normal or vertex normal). In Fig. 3. I am using vertex normal, to calculate vertex normal we need to first calculate polygon normal and vertex normal is calculate as a weighted average of these face normal can be used. For setting up lighting source and type we need to use glLightfv API.

```
GLfloat light_amibent[] = {1.0, 0.0, 0.0, 0.0};
GLfloat light_position[] = {1, 1.0, -1.3, 1.0};

glLightfv(GL_LIGHT0, GL_AMBIENT, light_amibent);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
```

Code Snippet

To specify light source position OpenGL uses 4D vector, to specify x,y,z (1.0, 1.0, -1.3) direction of light source and one dimension to define light source as point source in space if value is 1.0 otherwise at infinity. OpenGL normalize the position vector to direction of light source i.e (1.0, 1.0, -1.3) or (10, 10, -13) has same effect.

```
GLfloat white[] = {0.8f, 0.8f, 0.8f, 1.0f};
GLfloat red[] = {1.f, 0.0f, 0.0f, 1.f};
glMaterialfv(GL_FRONT, GL_DIFFUSE, red);
glMaterialfv(GL_FRONT, GL_SPECULAR, white);
GLfloat shininess[] = {50};
glMaterialfv(GL_FRONT, GL_SHININESS, shininess);
```

Code Snippet

Material properties of mesh object is defined as diffused with red in color using GL_DIFFUSE and specular highlight is defined as whitish in color.

User Interaction:

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

In this Application, user interaction happens only through keyboard. In scene there are three objects, rotation, scaling, translation happens only a particular object.

First to particular object user need to press 1 or 2 or 3

1: Wired mesh of teapot (By default this set)

2: Filled mesh of teapot

3: Cube

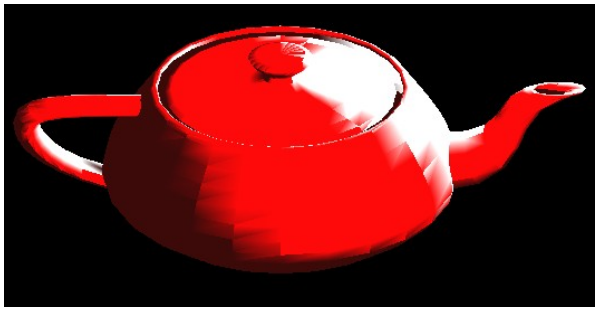
To rotate a mesh object, after selecting a particular object, press x to rotate on x-axis, press y to rotate on y-axis and press z to rotate. For rotation using quaterion, get rotation matrix and multiply it model matrix. Rotation happens over a fix degree of 20°.

```
static float rotation_transform[4][4];
wire_mesh_q.rotationMatrix(rotation_transform);
glMultMatrixf(&rotation_transform[0][0]);
if (rotate != 0) {
        wire_mesh_q.rotateAngle(20, (float)rotate_x, (float)rotate_y, (float)rotate_z);
}
```

Code Snippet

Orignal position

Rotate around Y-Axis



Rotate around X- Axis

Rotate around Z- Axis

For Scaling mesh object, I am using glScale OpenGL API, press '-' to scale down mesh object and '+' to scale up mesh object

```
glScalef(scale, scale, scale);
```

Code Snippet

Scaled down

Scaled Up

Normal Vectors:

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

Non normalized vector

Normalized vector

We have lighting effect we need to know at what angle light ray touches the face. And to calculate that we need to know normal vector point to the face. Normal vector is prependicular to the face. And to calculate angle we can take dot product of normal vector and light ray.

And for dot product to work in this particular vectors needs to be normalized. Impact of light ray on face is in range of 0° - 90° degree. And if vectors are not normalized chances of Θ becoming more that 90 degree is more which leads to suprious details.