3D Modelling and Rendering of Dharahara and Ghantaghar

Sandeep Acharya (075BCT074)
Sangam Chaulagain (075BCT078)
Saujan Tiwari (075BCT083)

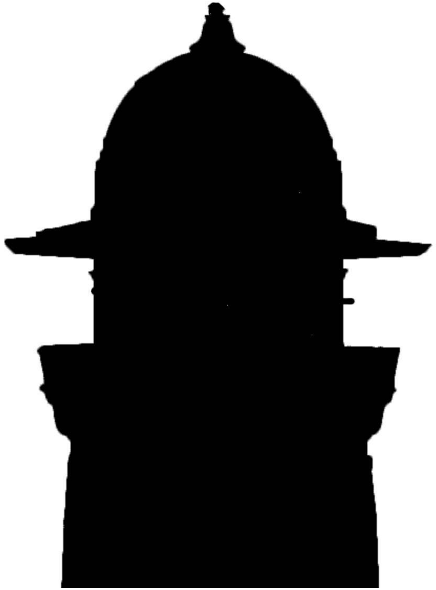
Introduction

Our graphics application features a street scene with historical monuments Dharahara and Ghantaghar rendered using OpenGL.

The graphics algorithms for 3D translation, rotation, scaling, camera for viewing transformation, perspective projection and phong illumination model are implemented in our project

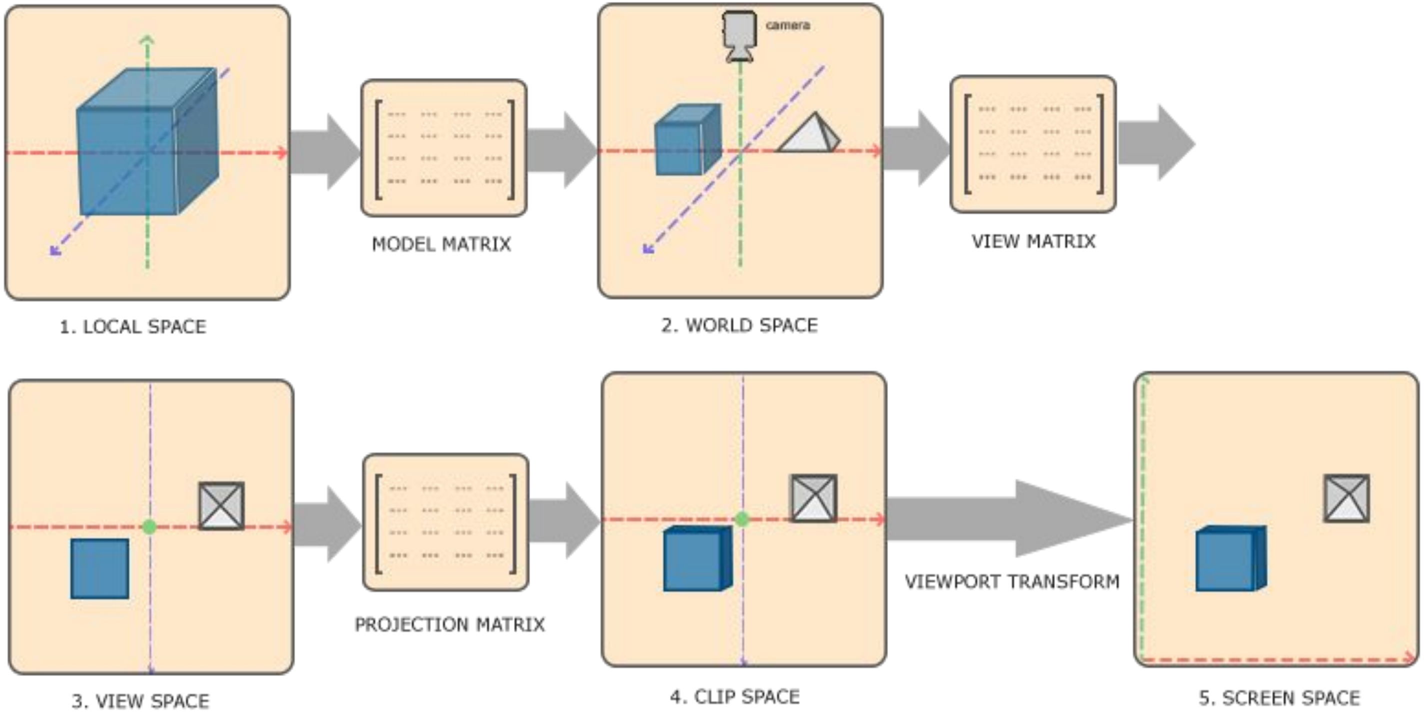
Objectives

- To learn 3D modeling in blender
- To learn about OpenGL graphics API
- To implement the Graphics algorithms
- To understand and implement the various stages of 3D graphics pipeline



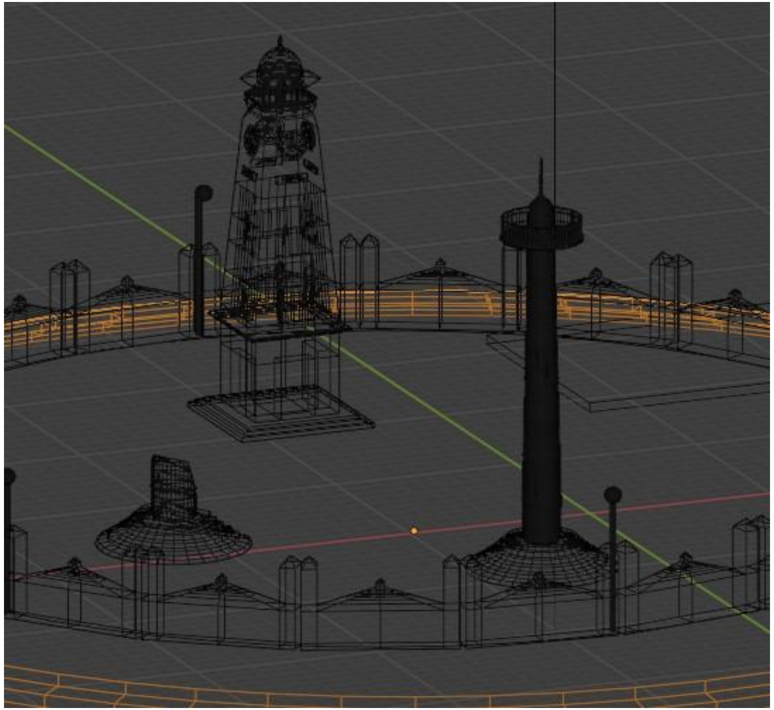
Tools used for development

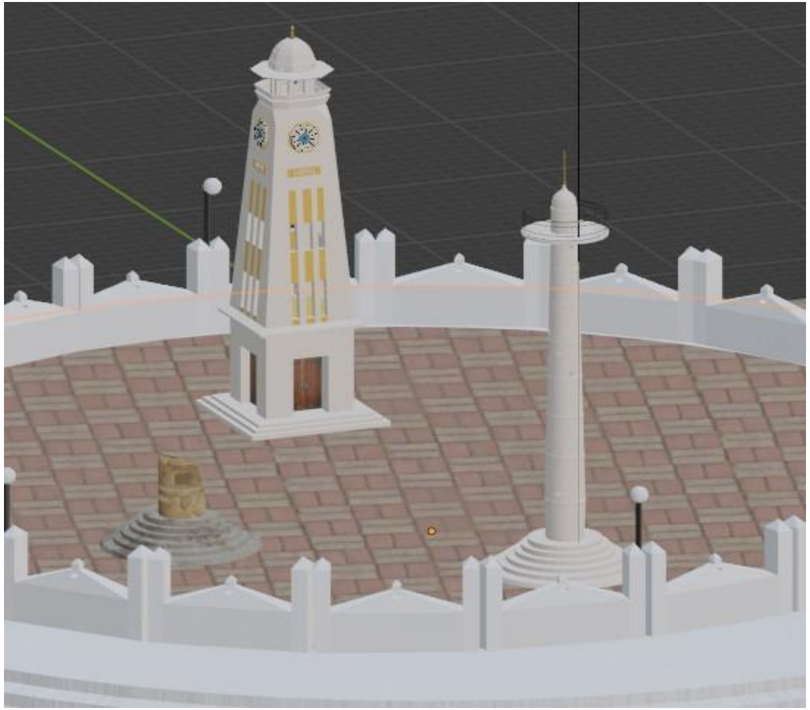
3D Graphics Rendering Pipeline



Object Modeling

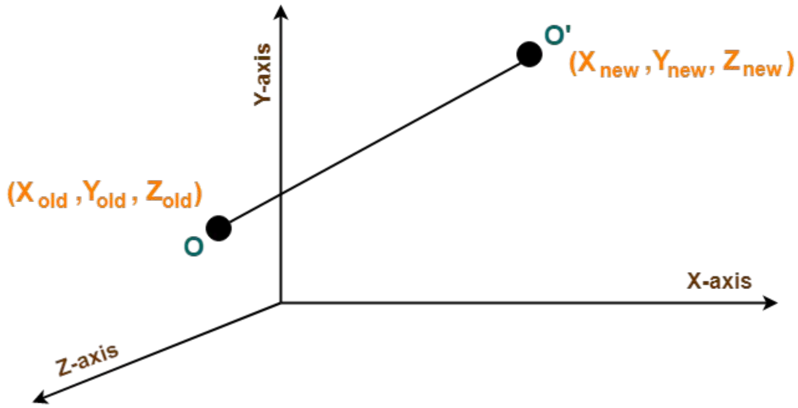
- 3D model of the objects in the scene are created using Blender





3D Translation

- In Computer graphics, 3D Translation is a process of moving an object from one position to another in a three dimensional plane.



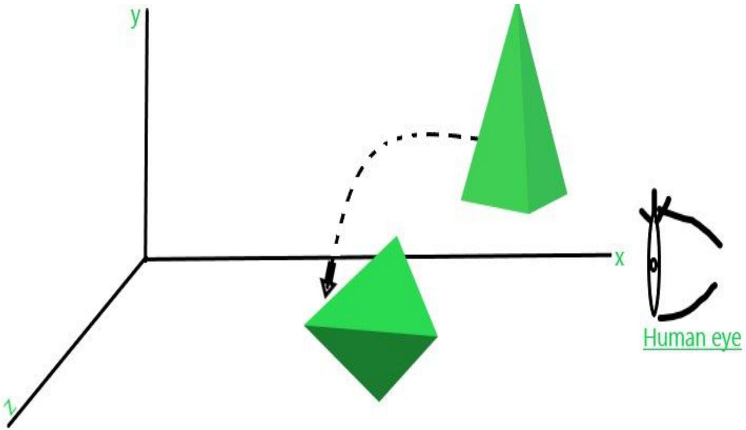
3D Translation in Computer Graphics

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

3D Translation Matrix

3D Rotation

- In Computer graphics, 3D Rotation is a process of rotating an object with respect to an angle in a three dimensional plane.



$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

3D Rotation Matrix

(For X-Axis Rotation)

$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

3D Rotation Matrix

(For Y-Axis Rotation)

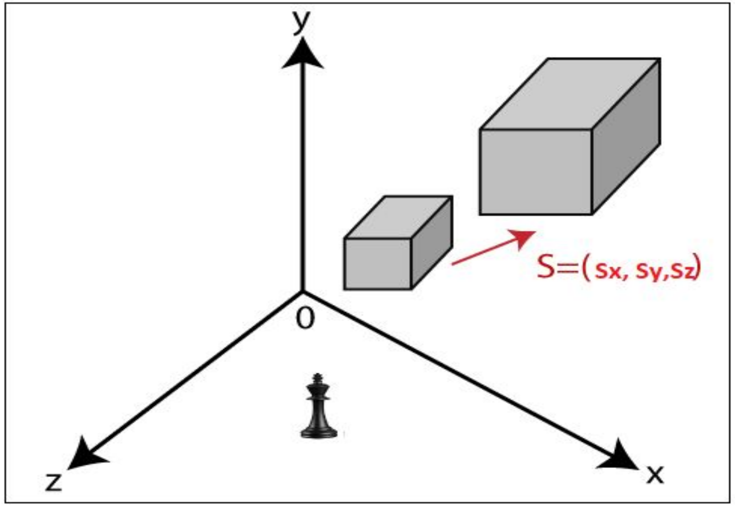
$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

3D Rotation Matrix

(For Z-Axis Rotation)

3D Scaling

- In computer graphics, scaling is a process of modifying or altering the size of objects.
- It may be used to increase or reduce the size of objects depending on the value of scaling factor.



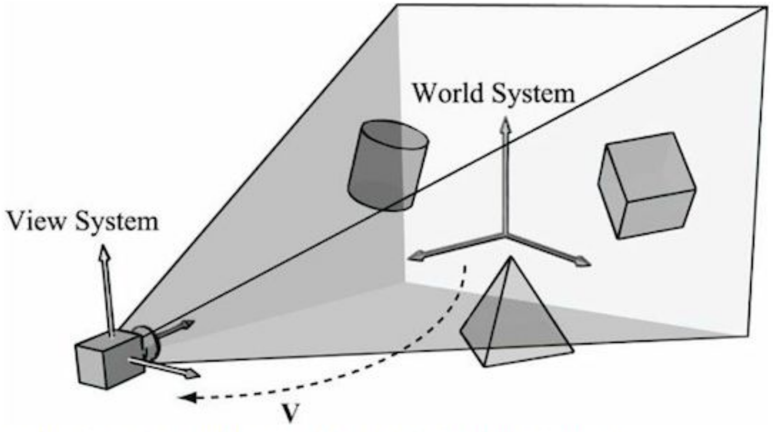
$$\begin{bmatrix} X_{new} \\ Y_{new} \\ Z_{new} \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_{old} \\ Y_{old} \\ Z_{old} \\ 1 \end{bmatrix}$$

3D Scaling Matrix

Viewing Transformation

The viewing transformation is done to transform the world coordinates to view space coordinates in such a way that each object is seen from the viewer's point of view.

The viewing transformation is performed in two steps: Translation followed by Rotation



First, translate the viewing-coordinate origin to the origin of the world coordinate system. Then rotation is performed to align viewing-space coordinate axes with the world coordinate axes.

$$\mathbf{n} = \frac{\mathbf{N}}{|\mathbf{N}|} = (n_x,\ n_y,\ n_z)$$

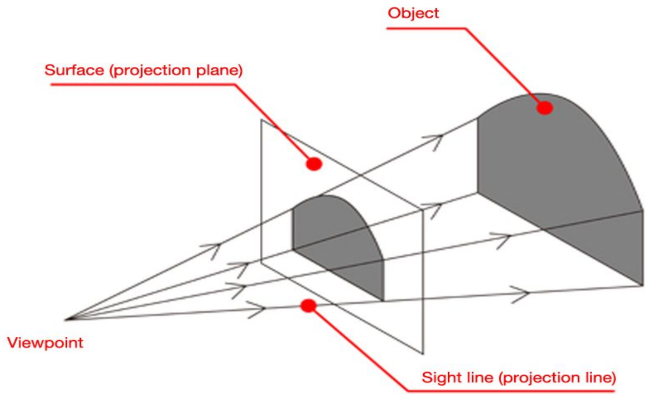
$$\mathbf{u} = \frac{\mathbf{V} \times \mathbf{n}}{|\mathbf{V} \times \mathbf{n}|} = (u_x,\ u_y,\ u_z)$$

$$\mathbf{v} = \mathbf{n} \times \mathbf{u} = (v_x,\ v_y,\ v_z)$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M}_{WC,\,VC} = \mathbf{R} \cdot \mathbf{T}$$

Perspective Projection



In perspective projection, the distance from the center of projection to project plane is finite and the size of the object varies inversely with distance which looks more realistic.

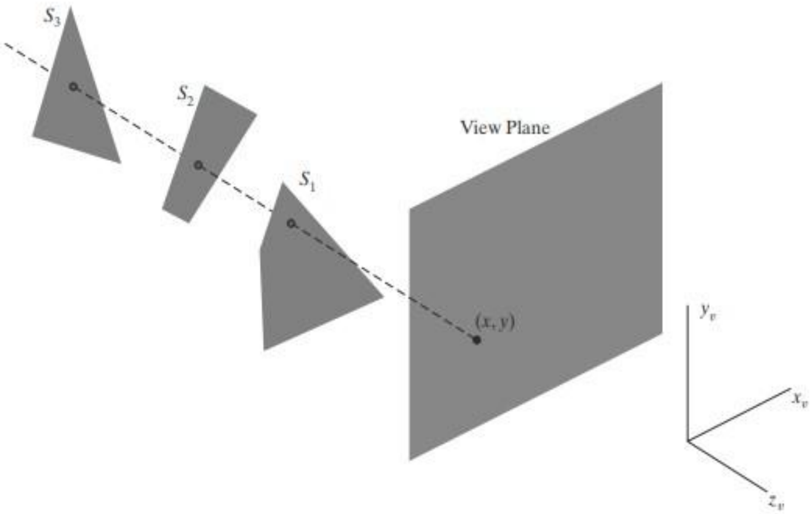
The distance and angles are not preserved and parallel lines do not remain parallel. Instead, they all converge at a single point called center of projection or projection reference point.

$$\begin{bmatrix} \frac{1}{aspect * \tan(\frac{fov}{2})} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(\frac{fov}{2})} & 0 & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & -\frac{2*far*near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

Depth Testing

It is complicated and inefficient to perform depth testing in fragment shader for a scene with about 32,000 vertices.

The built-in function in OpenGL **glEnable(GL_DEPTH_TEST)** is used for performing the depth testing. The Depth value of the fragment gets compared with value in Depth buffer, fragment is rendered accordingly and Depth buffer is updated.

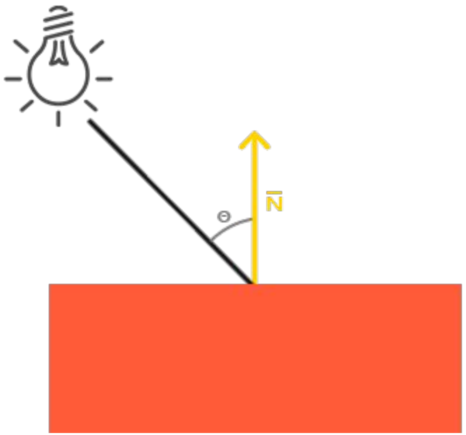


Lighting

- **Ambient Lighting:** Background lighting so that objects are almost never completely dark

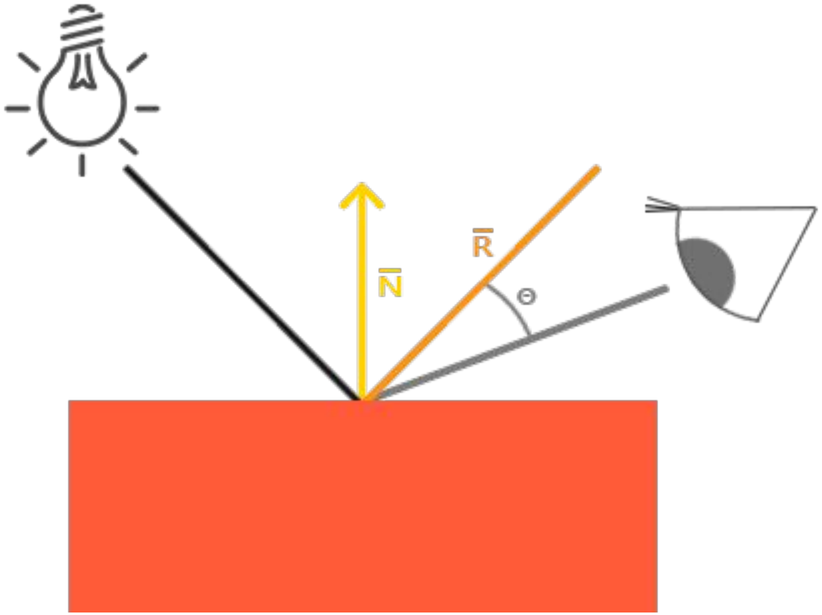
$$I = K_a * I_a$$

- **Diffuse Lighting :** Constant over each surface in a scene, independent of viewing direction.



$$I_{diff} = K_a I_a + K_d I_l (\vec{N}.\vec{L})$$

- **Specular Lighting:** Highlights or bright spots seen on shiny surfaces

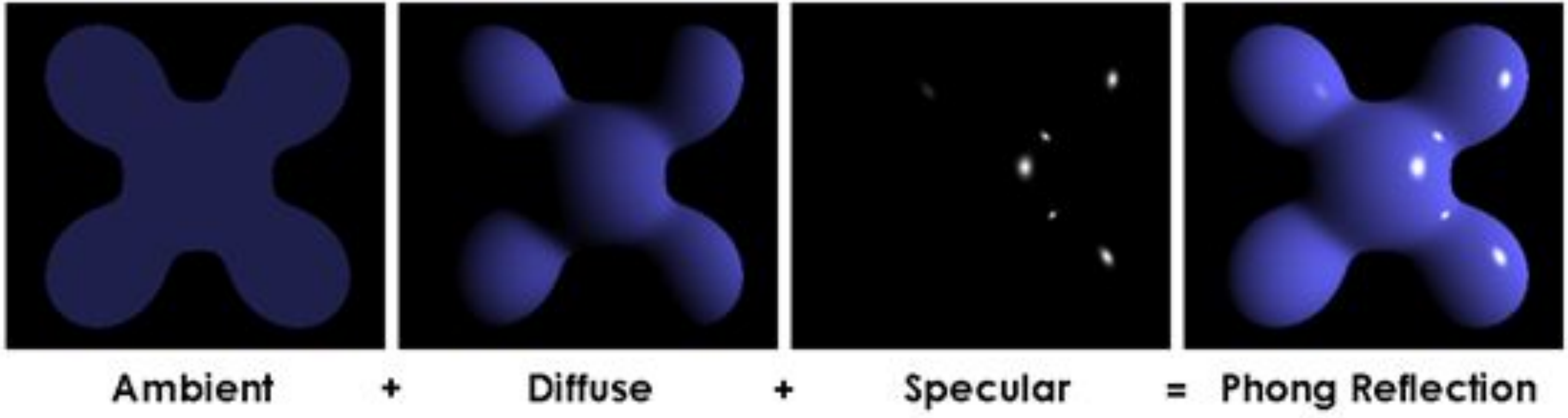


$$I_{spec} = K_s I_l\,(\vec{V}.\vec{R})^{n_s}$$

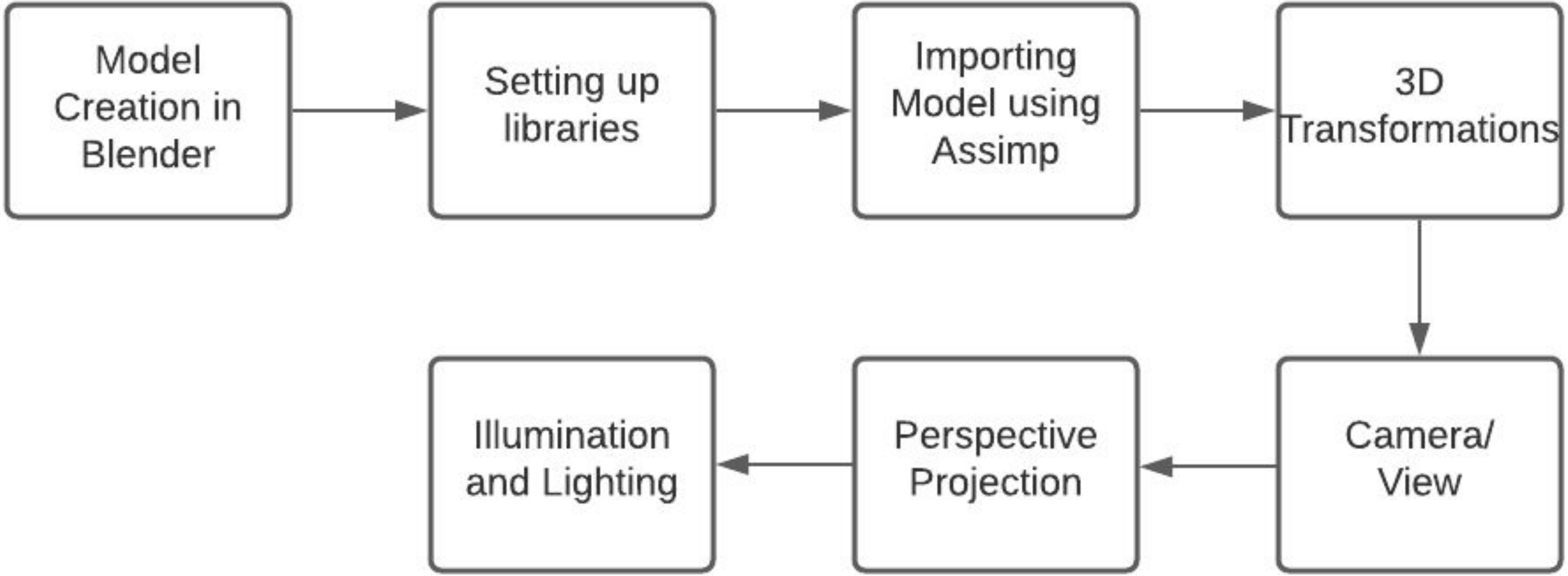
Phong Illumination Model

- Empirical model of the local illumination of points on a surface.
- Designed by the computer graphics researcher Bui Tuong Phong.
- Also called Phong illumination or Phong lighting

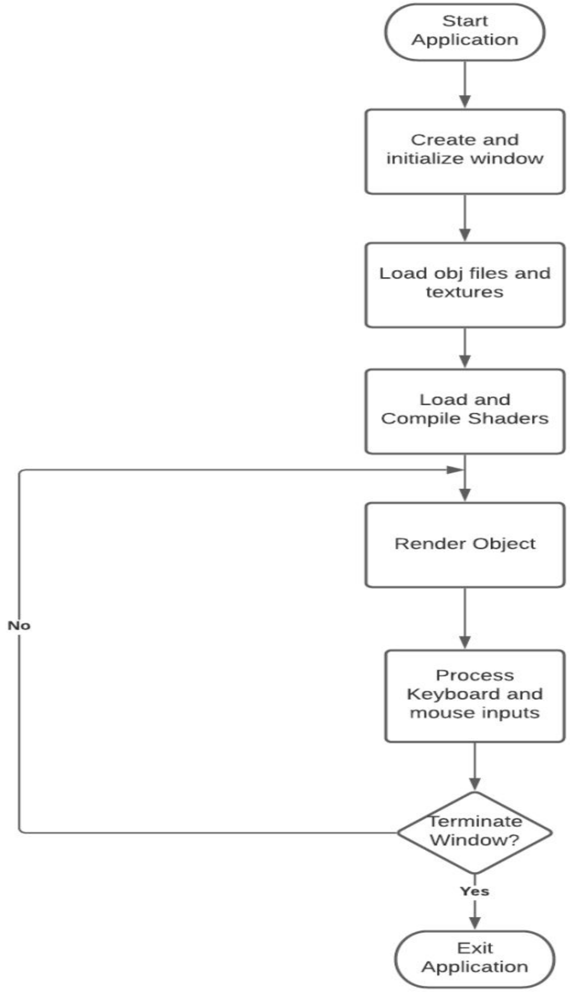
$$I = I_a k_a + f_{att} I_p [k_d (\overline{N}.\overline{L}) + k_s (\overline{V}.\overline{R})^n]$$



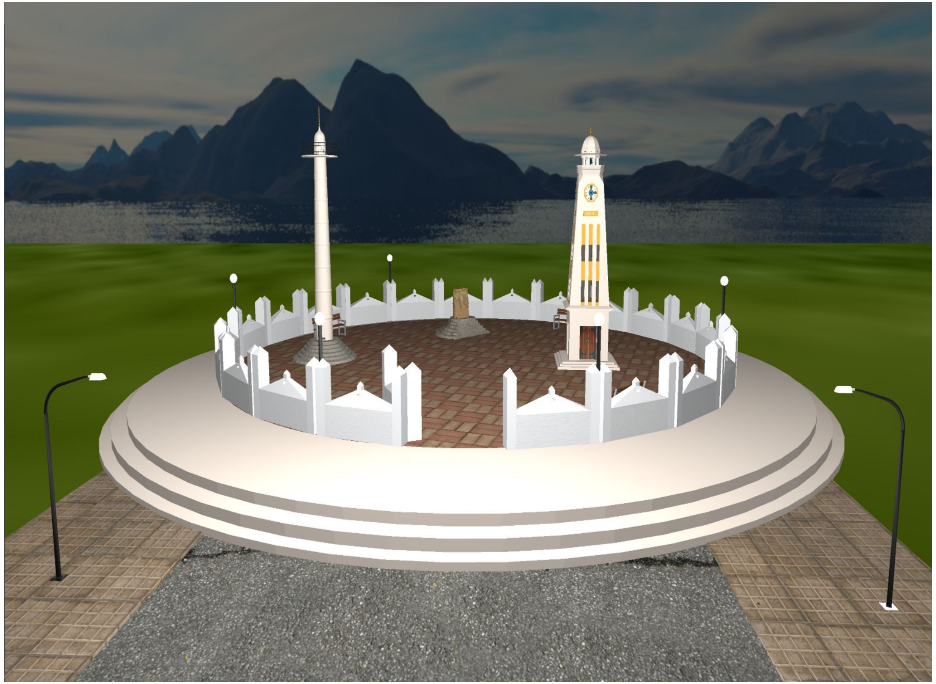
Project Block Diagram

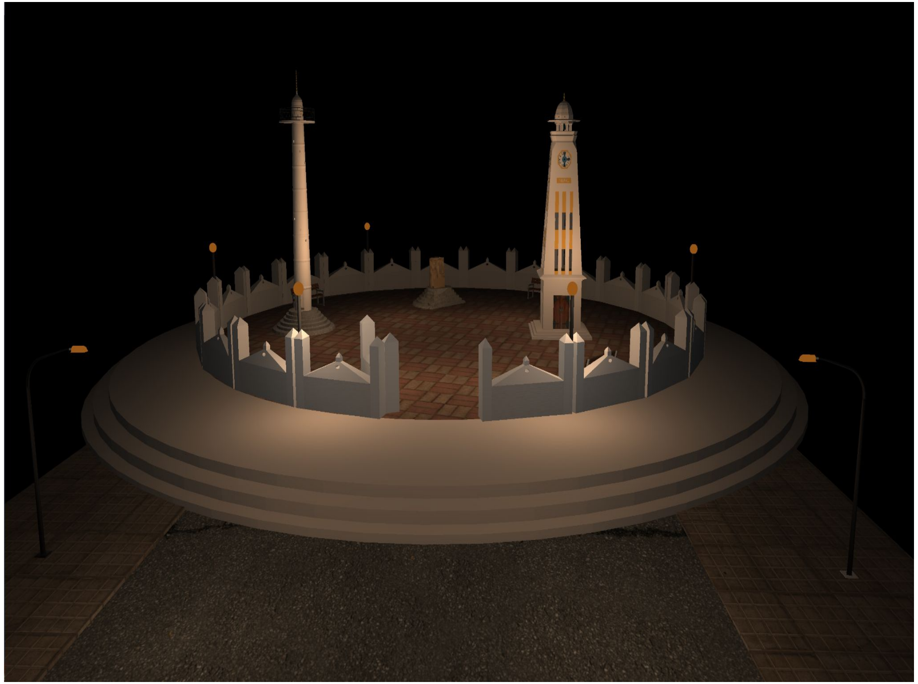


Application Flowchart

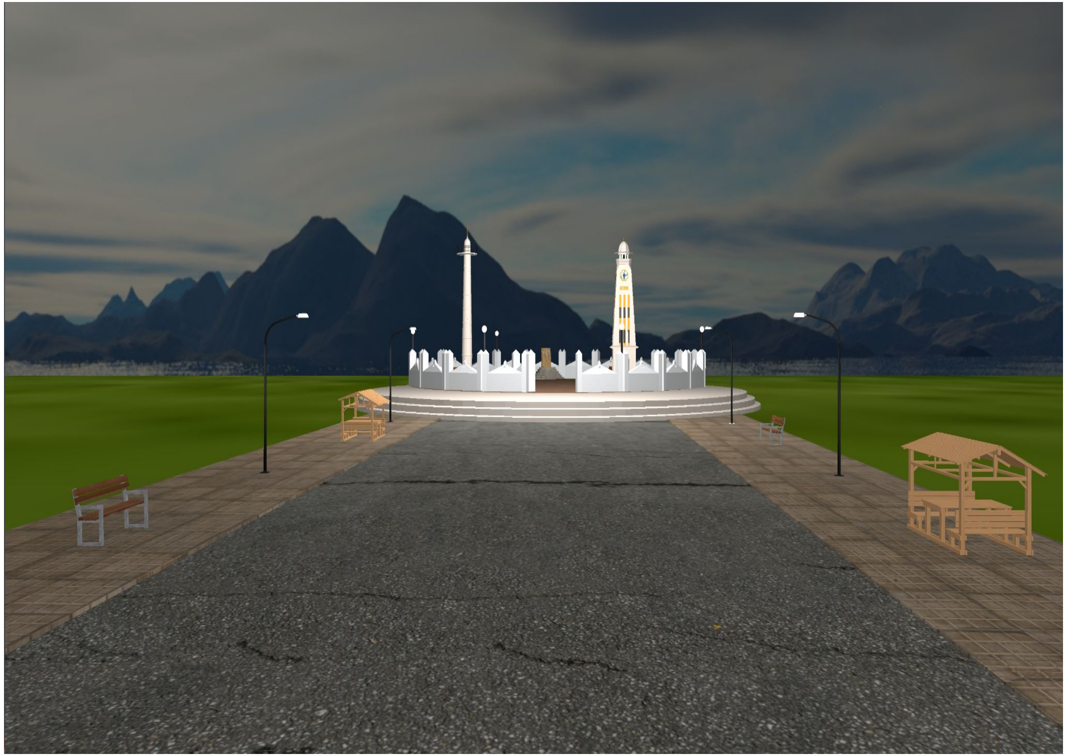


Output Images





Output Images



Limitations and Future Enhancements

The rendered scene is less realistic. No collision detection algorithm is in use. So, camera can move across the objects in the scene.

Lags might be observed between the user input and camera movement in the scene.

Future enhancements include using the scene as a part of open-world 3d games or developing the application itself as a simulation game using physics, audio, ray tracing, first person camera, characters and levels.

Thank You

For Your Patience