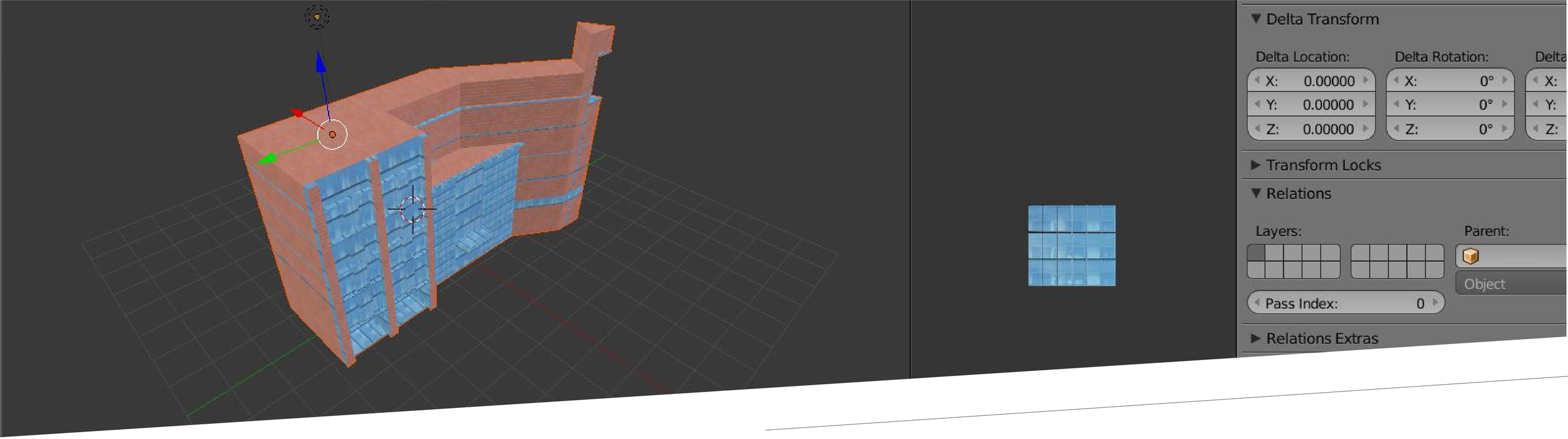
3D VIEW OF ICTC BUILDING

COMPUTER GRAPHICS PROJECT

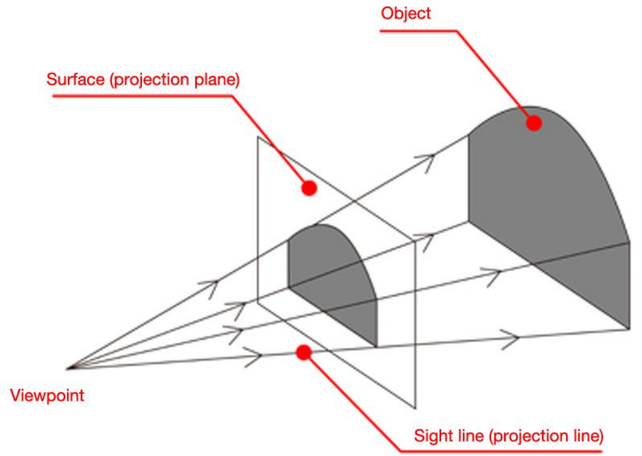


To design the 3D model of ICTC.

To implement the features of computer graphics learnt into the project.

To implement the 3D rotational, scaling, color filling and lightning effects..

Perspective Projection

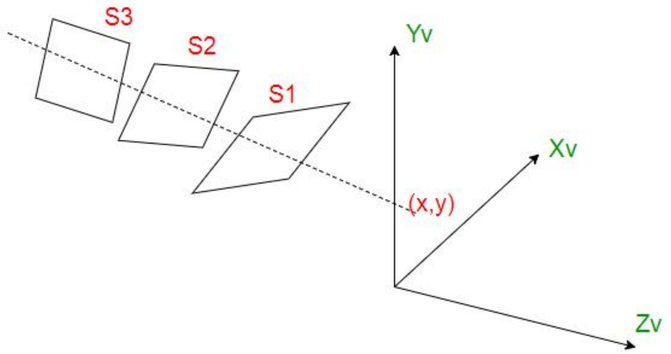


In perspective projection, the distance from the center of projection to project plane is finite and the size of the object varies inversely with distance which looks more realistic.

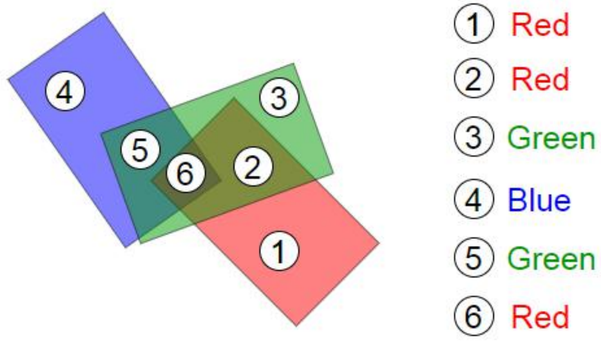
The distance and angles are not preserved and parallel lines do not remain parallel. Instead, they all converge at a single point called center of projection or projection reference point.

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

Z BUFFER



Z-Buffer with S1, S2, S3 Surfaces



1 Red
2 Red
3 Green
4 Blue
5 Green
6 Red

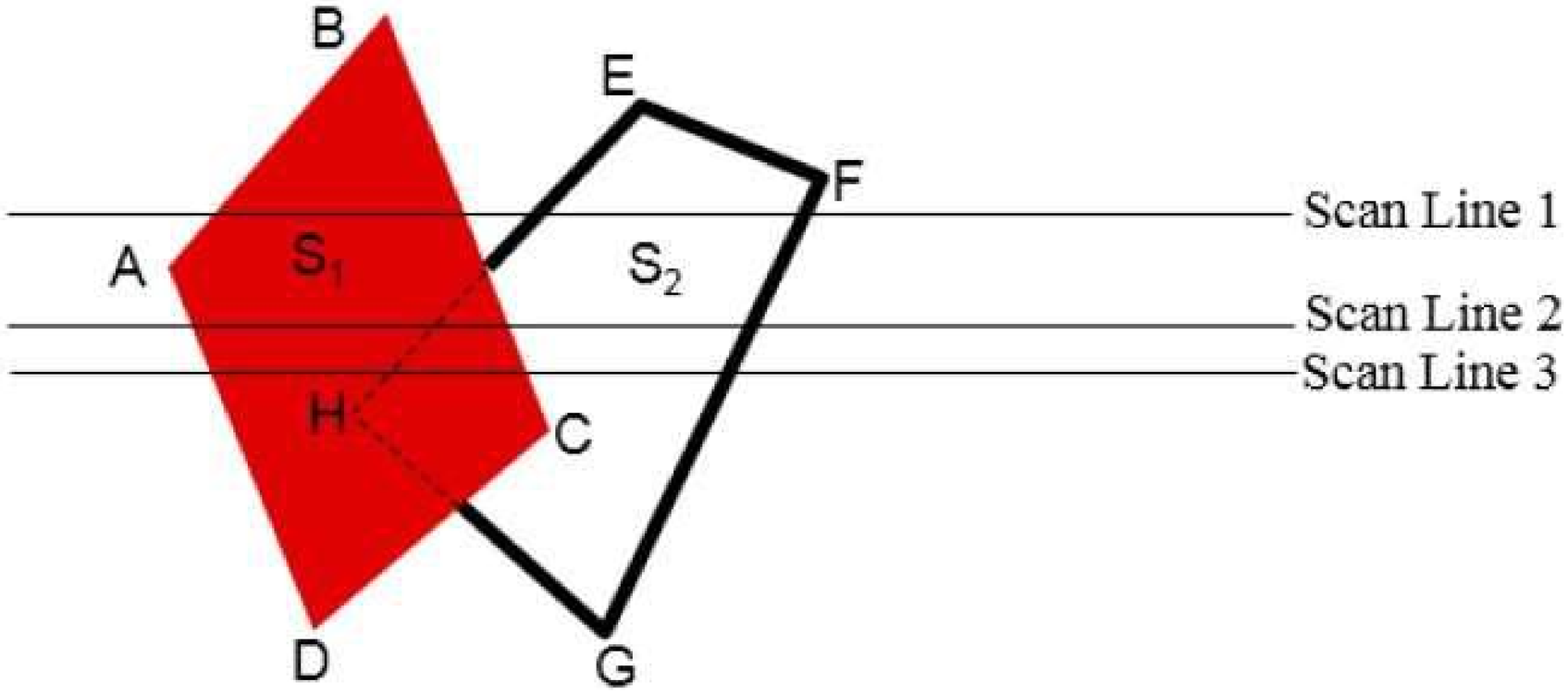
Limitations with transparent surfaces.

When viewing a picture containing non transparent objects and surfaces, it is not possible to see those objects from view which are behind from the objects closer to eye. To get the realistic screen image, removal of these hidden surfaces is must. The identification and removal of these surfaces is called as the Hidden-surface problem.

Z-buffer, which is also known as the Depth-buffer method is one of the commonly used method for hidden surface detection. It is an Image space method. Image space methods are based on the pixel to be drawn on 2D. For these methods, the running time complexity is the number of pixels times number of objects. And the space complexity is two times the number of pixels because two arrays of pixels are required, one for frame buffer and the other for the depth buffer.

The Z-buffer method compares surface depths at each pixel position on the projection plane.

Scan-Line Method



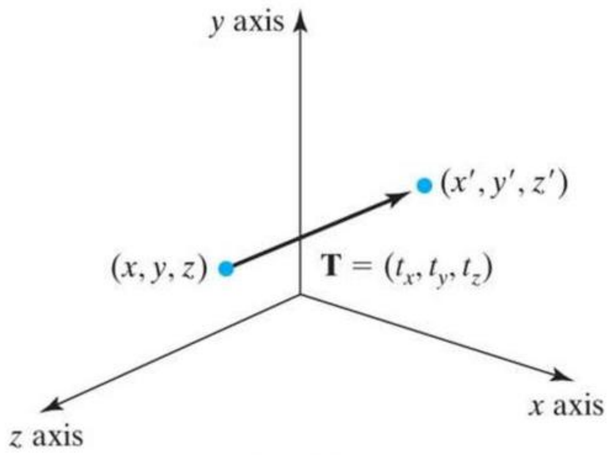
It is an image-space method to identify visible surface. This method has a depth information for only single scan-line. In order to require one scan-line of depth values, we must group and process all polygons intersecting a given scan-line at the same time before processing the next scan-line. Two important tables,edge table and polygon table, are maintained for this.

TRANSLATION

Translation can best be described as linear change in position. This change can be represented by a delta vector [Tx, Ty, Tz], where Tx represents the change in the object's x position, Ty represents the change in its y position, and Tz represents its change in z position.

3D translation

Figure 9-1 Moving a coordinate position with translation vector $T = (t_x, t_y, t_z)$.



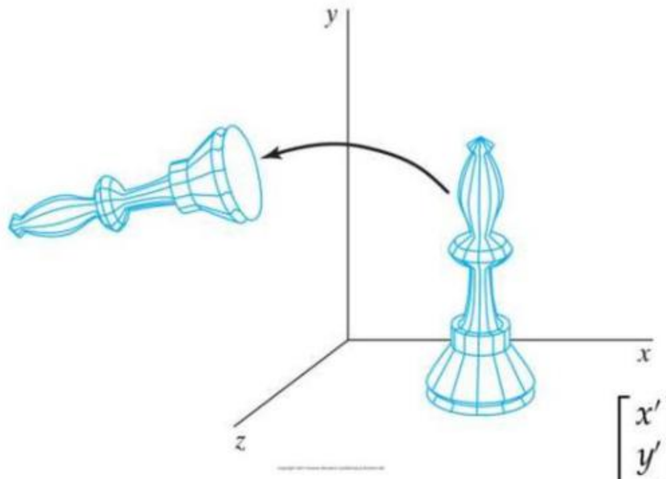
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{T} \cdot \mathbf{P}$$

ROTATION

3D z-axis rotation

Figure 9-4 Rotation of an object about the *z* axis.



$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotations

- To obtain rotations about other two axes
  - x ➔ y ➔ z ➔ x
- E.g. x-axis rotation

$$y' = y\cos\theta - z\sin\theta$$
$$z' = y\sin\theta + z\cos\theta$$
$$x' = x$$

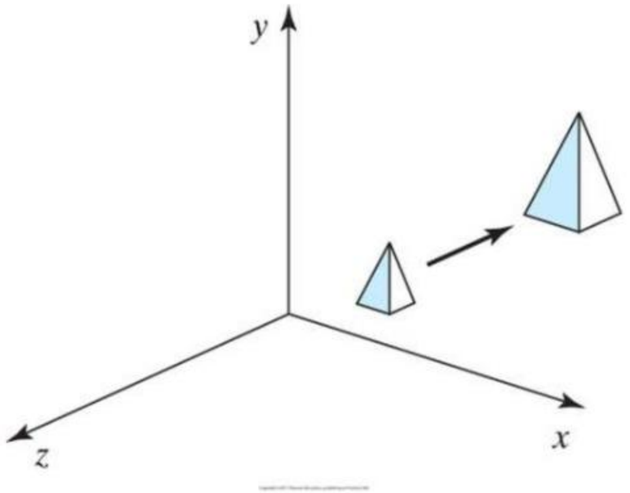
- E.g. y-axis rotation

$$z' = z\cos\theta - x\sin\theta$$
$$x' = z\sin\theta + x\cos\theta$$
$$y' = y$$

SCALING

3D scaling

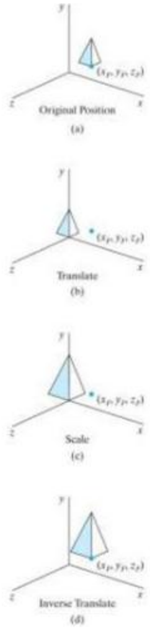
Figure 9-17 Doubling the size of an object with transformation 9-41 also moves the object farther from the origin.



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{S} \cdot \mathbf{P}$$

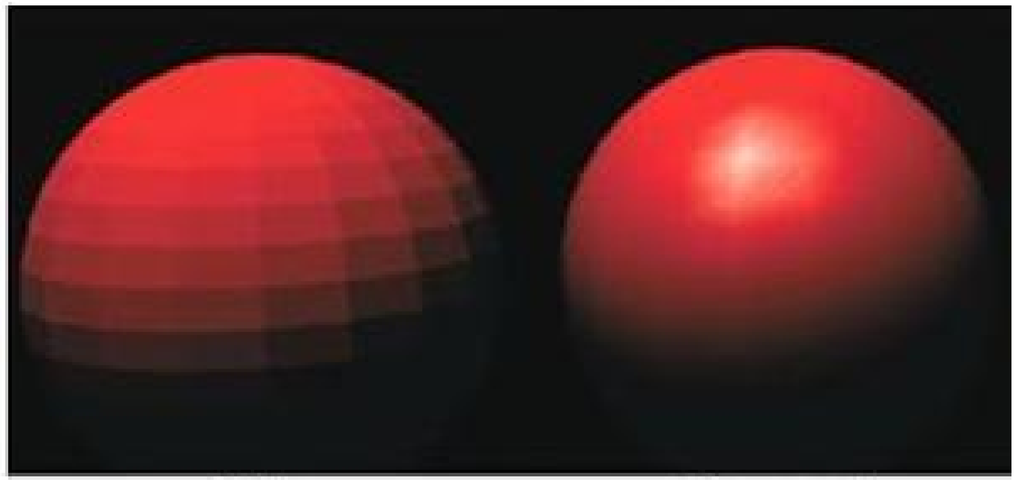
Figure 9-18 A sequence of transformations for scaling an object relative to a selected fixed point, using Equation 9-41.



$$\mathbf{T}(x_f, y_f, z_f) \cdot \mathbf{S}(s_x, s_y, s_z) \cdot \mathbf{T}(-x_f, -y_f, -z_f) = \begin{bmatrix} s_x & 0 & 0 & (1-s_x)x_f \\ 0 & s_y & 0 & (1-s_y)y_f \\ 0 & 0 & s_z & (1-s_z)z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

GOURAUD SHADING

Gouraud shading is considered superior to flat shading and requires significantly less processing than Phong shading, but usually results in a faceted look.

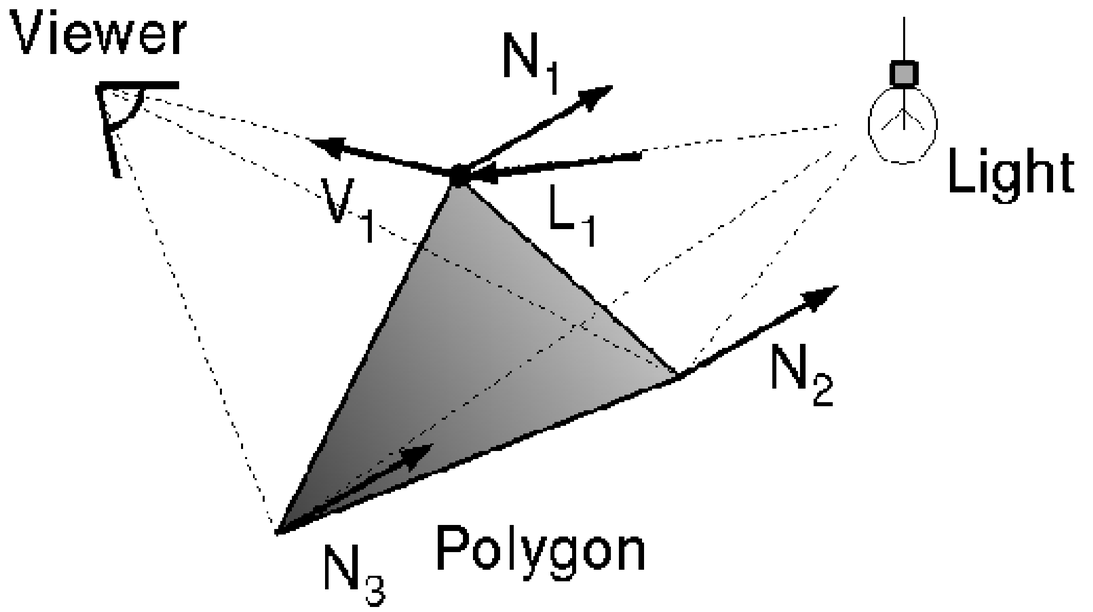


A polygon surface is rendered using Gouraud shading by carrying out the following steps:

- First, the average normal vector is determined at each polygon surface.
- Then, linearly interpolate the intensity over the surface of polygon.
- Apply an illumination model along each scan line to calculate projected pixel intensities for the surface points.

Gouraud Shading

- One lighting calculation per vertex
  - Assign pixels inside polygon by interpolating colors computed at vertices



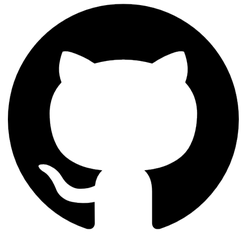
TOOLS USED FOR DEVELOPMENT

IDE USED

Visual Studio 15.7

CLion 2019.1

Version Control



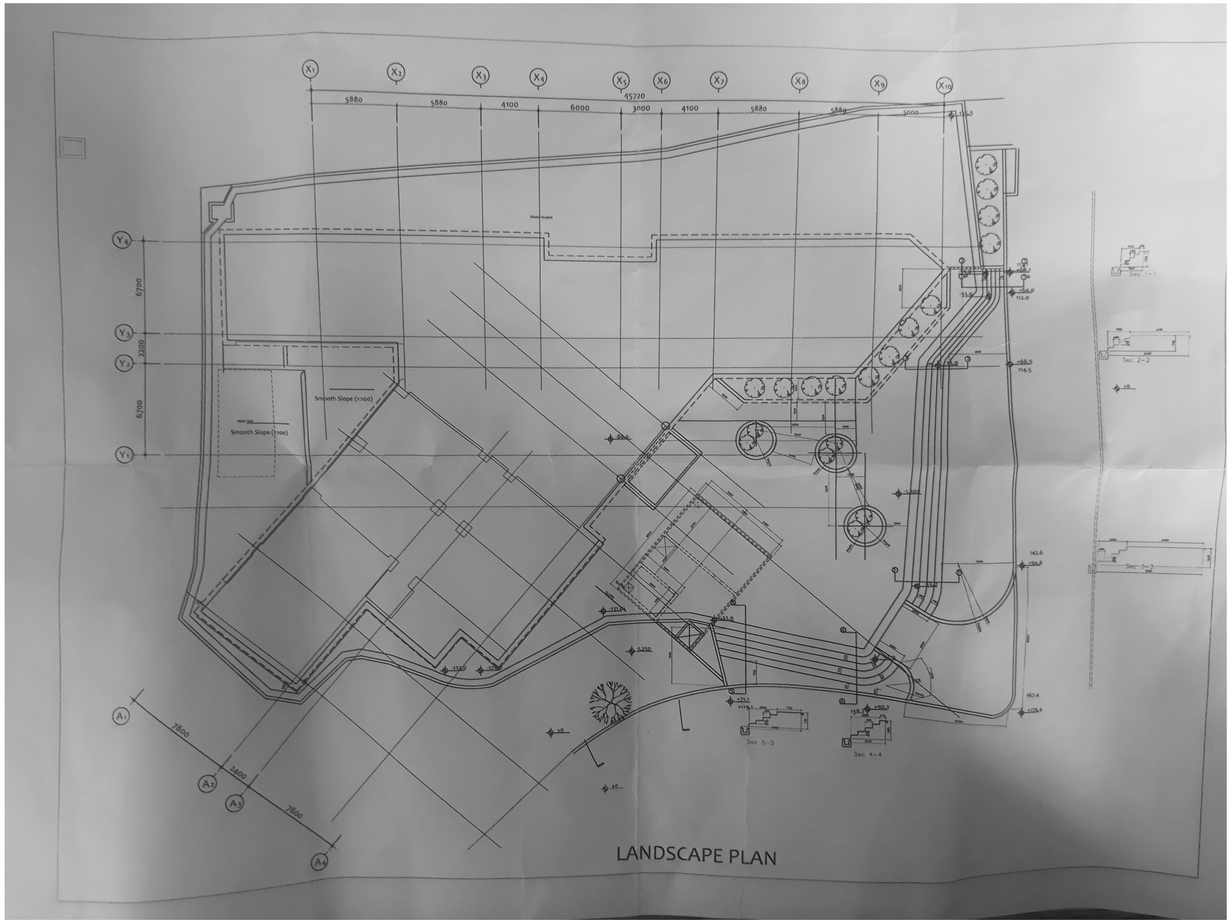
GitHub

Other Tools

CMake

Blender

REFERENCES



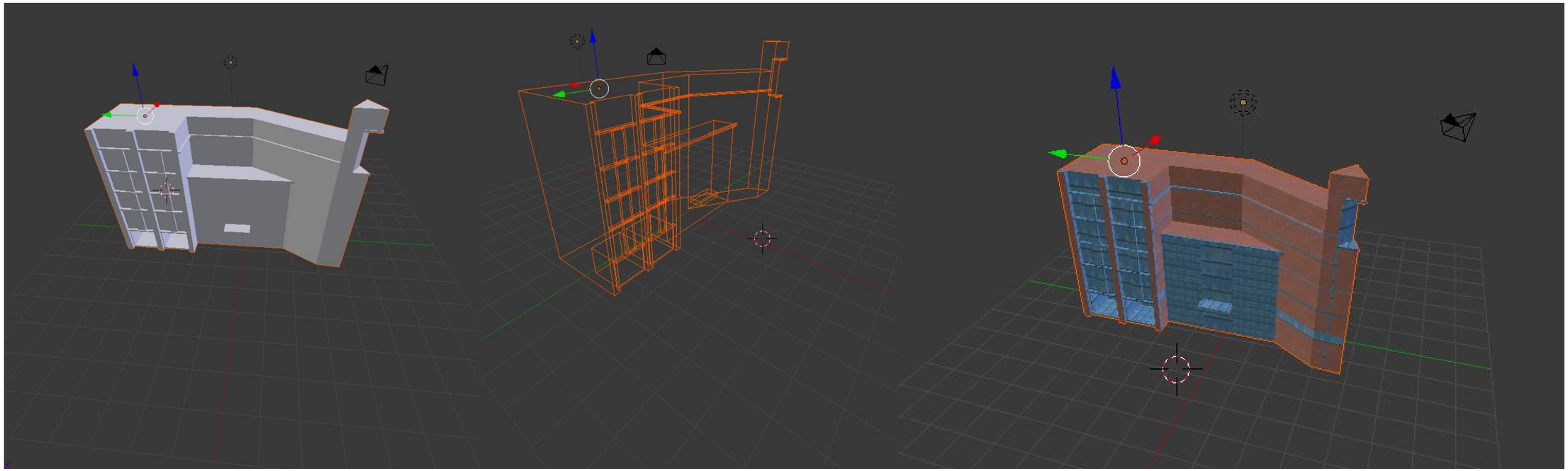
ICTC Building Landscape Plan

https://learnopengl.com/

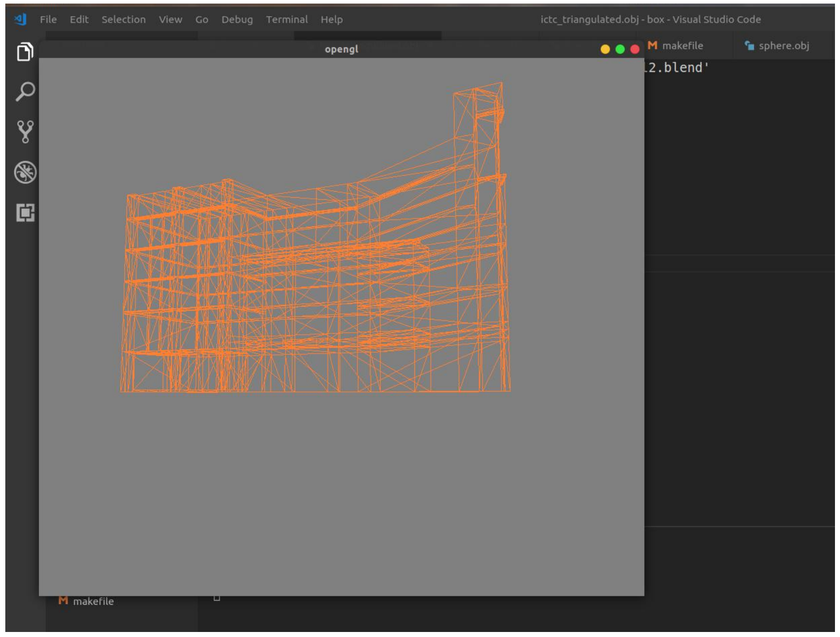
https://www.tutorialspoint.com/

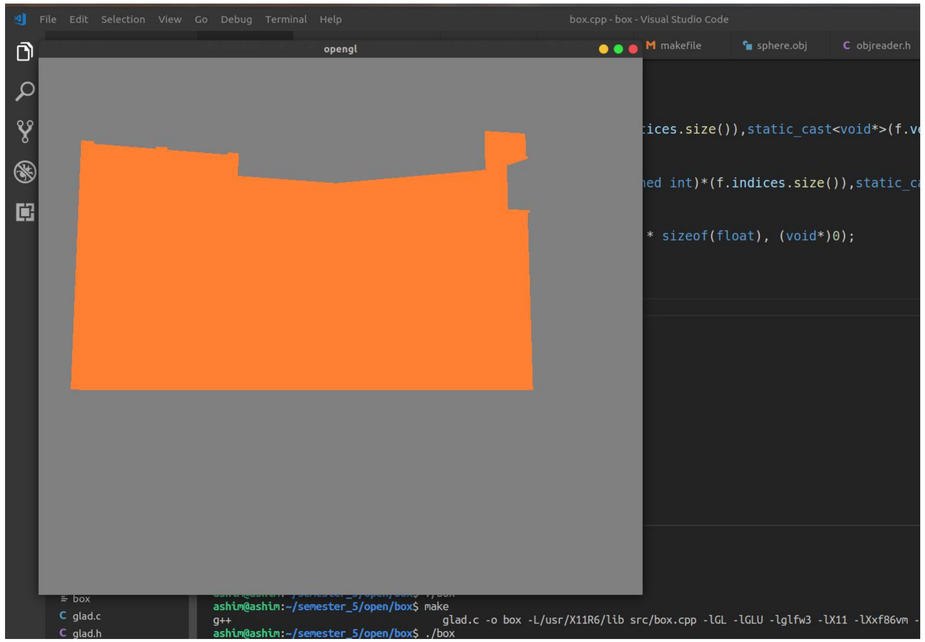
https://www.udemy.com/learn-3d-modeling-blender-basics-in-under-2-hours/

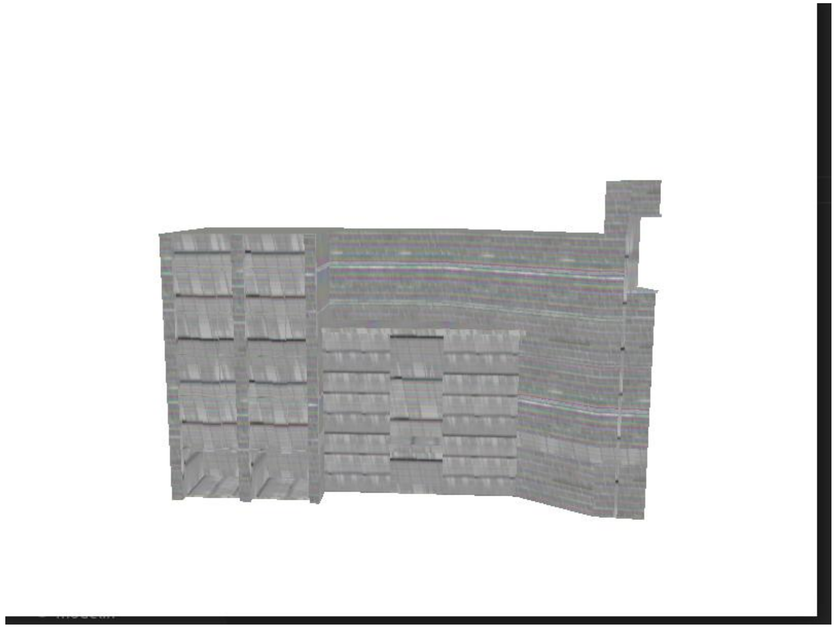
Development Phases

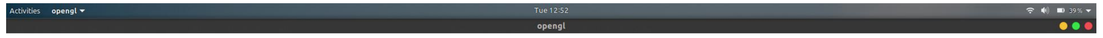


Object Modeling In Blender


File Edit Selection View Go Debug Terminal Help
ictc_triangulated.obj - box - Visual Studio Code
opengl
makefile
sphere.obj
2.blend'
makefile


File Edit Selection View Go Debug Terminal Help
box.cpp - box - Visual Studio Code
opengl
makefile
sphere.obj
objreader.h
box
glad.c
glad.h
ashim@ashim:~/semester_5/open/box$ make
g++ glad.c -o box -L/usr/X11R6/lib src/box.cpp -lGL -lGLU -lglfw3 -lX11 -lXxf86vm -
ashim@ashim:~/semester_5/open/box$ ./box




Activities
opengl
Tue 12:52
opengl