FUTURE VISION BIE

One Stop for All Study Materials

& Lab Programs

Future Vision

By K B Hemanth Raj

Scan the QR Code to Visit the Web Page

Or

Visit : https://hemanthrajhemu.github.io

Gain Access to All Study Materials according to VTU,
CSE – Computer Science Engineering,
ISE – Information Science Engineering,
ECE - Electronics and Communication Engineering
& MORE...

Join Telegram to get Instant Updates: https://bit.ly/VTU_TELEGRAM

Contact: MAIL: futurevisionbie@gmail.com

INSTAGRAM: www.instagram.com/hemanthraj_hemu/

INSTAGRAM: www.instagram.com/futurevisionbie/

WHATSAPP SHARE: https://bit.ly/FVBIESHARE

66. What is affine transformation? → 159
67. List the 3D OpenGL geometric transformations. → 168
68. What is color model? Explain the RGB color model. → 171
69. Explain the CMY and CMYK color models. → 173
70. What is light source? Explain the types of light source. → 175
71. Explain the PHONG model. → 182

MODULE 4

72. What is projection plane, parallel and perspective projection? → 184
73. What is depth cueing? → 188
74. Explain the 3D viewing pipeline with diagram. → 189
75. Explain the transformation from world to viewing coordinates. → 190
76. Explain the orthogonal projections. → 192
77. Explain the perspective projection transformation coordinates. → 194
78. Explain the OpenGL 3D viewing functions. → 197
79. Classify the visible surface detection algorithms. → 200
80. Explain the back-face detection algorithm. → 201
81. Explain the z-buffer/depth-buffer algorithm. → 203
82. Explain the OpenGL visibility detection functions. → 206
83. Explain in detail, Oblique and Symmetric perspective projection frustum. → 209
84. Explain vanishing points for perspective projections. → 212
85. Explain briefly the following: → 214
    a) Projections
    b) Depth Cueing
    c) Identifying visible lines and surfaces
    d) Surface rendering
    e) Exploded and cutaway views
    f) 3D and stereoscopic viewing
86. Explain viewup vector and uvn viewing coordinate reference frame → 220
87. Write short notes on axonometric and isometric orthogonal projections → 222.
88. Explain OpenGL functions with respect to: → 223
    a. Viewing Transformation functions
    b. Orthogonal Projection functions
    c. Symmetric Perspective Projection functions
    d. General Perspective Projection functions
    e. Viewport and Display Window
89. Imagine you have a 3D object in front of you. Illustrate how to Normalize the transformation for an Orthogonal Projection? → 227

MODULE 5

90. Explain how an event driven input can be performed for → 229
    (a) window events (b) pointing devices
91. Explain how an event driven input can be programmed for a keyboard device. → 232
92. List out any four characteristics of good interactive program. → 234
93. What are the major characteristics that describe the logical behavior of an input device? → 236
94. Explain how OpenGL provides the functionality of each of the classes of logical input devices. → 238

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

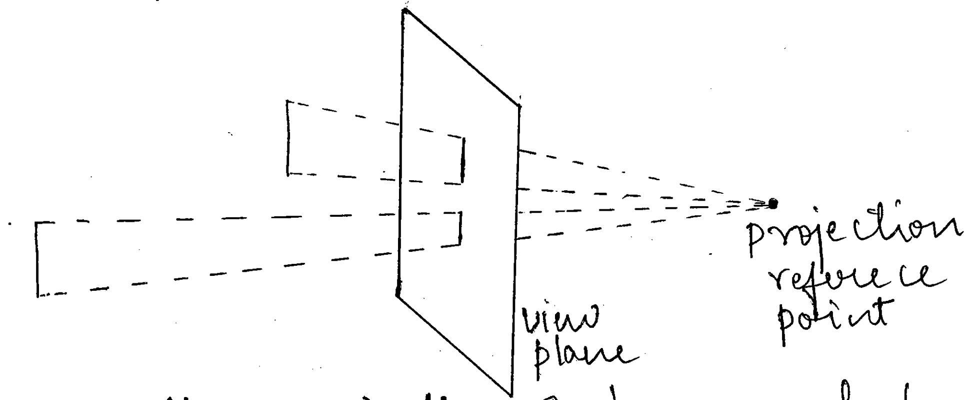
Shankar R
Asst Professor,
CSE, BMSIT&M

72. What is perspective and parallel projection, projection plane?

ans:

Perspective Projection:

To approximate the geometric-optics effect by projecting objects to view plane along converging paths to position called projection reference point. You can sometime select projection reference point as another viewing parameter in graphics package, but some systems place this convergence point a fixed position, such as at view point.



Perspective projection of two equal-length line segments at different distances from view plane.

Figure below shows projection path of a spatial position (x, y, z) to general projection reference point at $(x_{prp}, y_{prp}, z_{vp})$, where z_{vp} is some selected position for view plane on z_{view} axis.

Equations describing coordinate positions along perspective-projection line in parametric forms

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

$$x' = x - (x - x_{prp})u$$
$$y' = y - (y - y_{prp})u \qquad 0 \le u \le 1$$
$$z' = z - (z - z_{prp})u$$

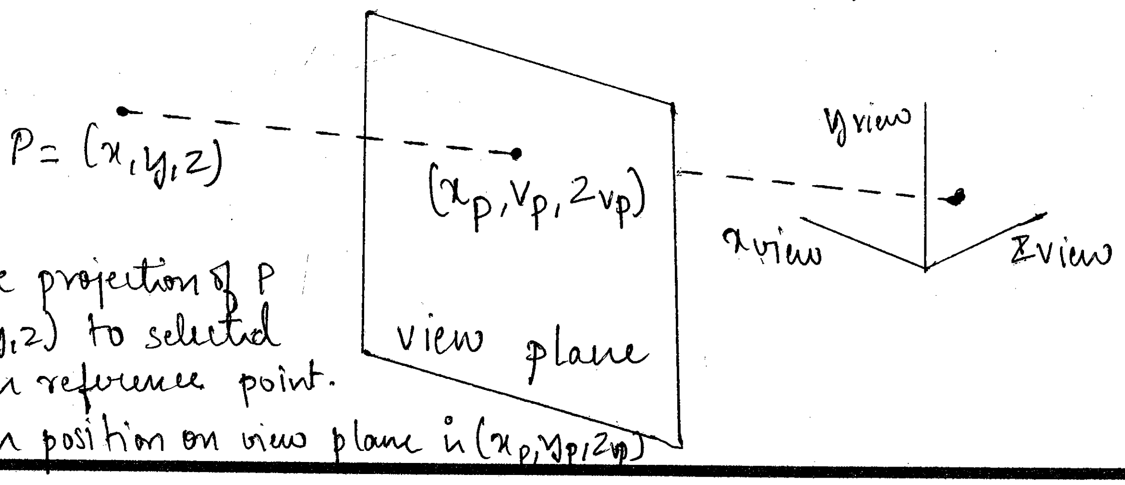
Coordinate position (x', y', z') represents any point along projection line. When $u=0$, you are at position $P=(x, y, z)$. At other end of line, $u=1$, and you have projection reference-point coordinates $(x_{prp}, y_{prp}, z_{prp})$. On view plane, $z' = z_{vp}$ and we can solve z' equation for parameter u at this position along projection line:

$$u = \frac{z_{vp} - z}{z_{prp} - z}$$

Substituting this value of u into equations for x' and y', you obtain general perspective-transformation equations

$$x_p = x\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) + x_{prp}\left(\frac{z_{vp} - z}{z_{prp} - z}\right)$$

$$y_p = y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) + y_{prp}\left(\frac{z_{vp} - z}{z_{prp} - z}\right)$$



Perspective projection of P with (x, y, z) to selected projection reference point. Intersection position on view plane is (x_p, y_p, z_{vp})

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

Projection plane:

A projection plane, or plane of projection, is a type a view in which graphical projections from an object intersect. Projection planes are used often in descriptive geometry and graphical representation. A picture plane in perspective drawing is a type of projection plane.

An orthographic plan view can be any one of six principal plan views (top, bottom, front, rear, left and right) derived from three principal planes of projection.

Types of projection in computer Graphics are

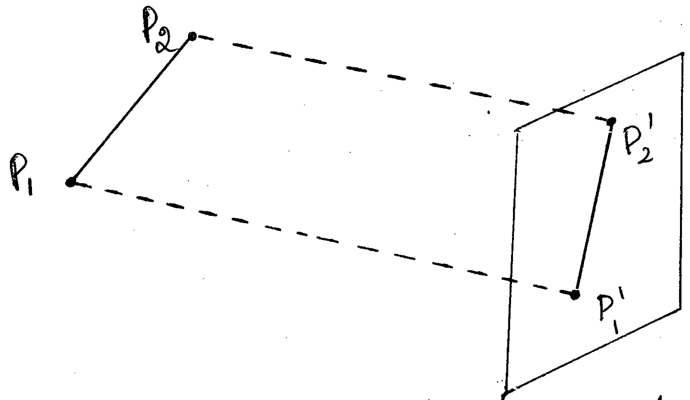
i. Perspective projection
ii. Parallel projection.

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

Parallel Projection :

In a parallel projection, coordinate positions are transferred to a view plane along parallel lines. A parallel projection preserves relative proportions of objects, and this method is used in computer aided drafting and design to produce scale drawings of 3-D objects. All parallel lines in a scene are displayed as parallel when viewed with a parallel projection. There are two general methods for obtaining parallel-projection view of an object: We can project along lines that are perpendicular to view plane, or we can project at an oblique angle to view plane.



parallel projection of a line segment onto a view plane

Above figure illustrates a parallel projection for a straight line segment defined with endpoint coordinates P_1 and P_2.

Shankar R
Asst Professor,
CSE, BMSIT&M

Q73 Write a short note on depth cueing

→ Depth information is important in a 3-dimensional scene so that we can easily identify, for a particular viewing direction, which is the front & which is the back of each displayed object.

→ Depth cueing is applied by choosing a maximum & a minimum intensity value and a range of distances over which the intensity is to vary.

→ A simple method to indicate depth in wire frame display is to vary the brightness of line segment based on their distances from the viewing position. Line closest to the viewing position are displayed with highest intensity, line farther away are displayed with decreasing intensity.

→ Another method of depth cueing is modelling the effect of the atmosphere on the percieved intensity of the objects.
Distant objects appear dimmer than near objects due to light scattering by dust particles, haze & smoke.

Scanned by CamScanner

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/
Shankar R
Asst Professor,
CSE, BMSIT&M

75. NAME : SUMAN.R.A
USN : 1BY16CS085
CLASS : VI SEM , 'B'
BRANCH : CSE

QUESTION :- Explain the transformation from the world to viewing coordinate system.

ANSWER :- In the three-dimensional viewing pipeline, the first step after a scene has been constructed is to transfer object descriptions to the viewing coordinate reference frame. This conversion of object description is equivalent to a sequence of transformations that superimposes the viewing reference frame onto the world frame.

1) Translate the viewing coordinates origin to the origin of the world coordinates system.

2) Apply rotations to align the zview, xview, yview and zview axes with the world x_w, y_w, and z_w axes respectively.

The viewing-coordinate origin is at world position $P_0 = (x_0, y_0, z_0)$. Therefore, the matrix for translating the viewing origin to the world origin is :-

$$T = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For the rotation transformation, we can use the unit vectors u, v and n to form the composite rotation matrix that superimposes the viewing axes onto the

Scanned by CamScanner

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

world frame. This transformation matrix is:-

$$R = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

, where the elements of matrix R are the components of uvn axis rotation.

The coordinate transformation matrix is then obtained as the product of the preceding translation and rotation matrices.

$$M_{wc,vc} = R \cdot T = \begin{bmatrix} u_x & u_y & u_z & -u \cdot P_0 \\ v_x & v_y & v_z & -v \cdot P_0 \\ n_x & n_y & n_z & -n \cdot P_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translate factors in this matrix are calculated as the vector dot product of each of the u, v and n unit vectors with P_0, which represents a vector from the world origin to the viewing origin.

These matrix elements are evaluated as:

$$-u \cdot P_0 = -x_0 u_x \cdot y_0 u_y - z_0 u_z$$

$$-v \cdot P_0 = -x_0 v_x - y_0 v_y - z_0 v_z$$

$$-n \cdot P_0 = -x_0 n_x - y_0 n_y - z_0 n_z \;//.$$

Scanned by CamScanner

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

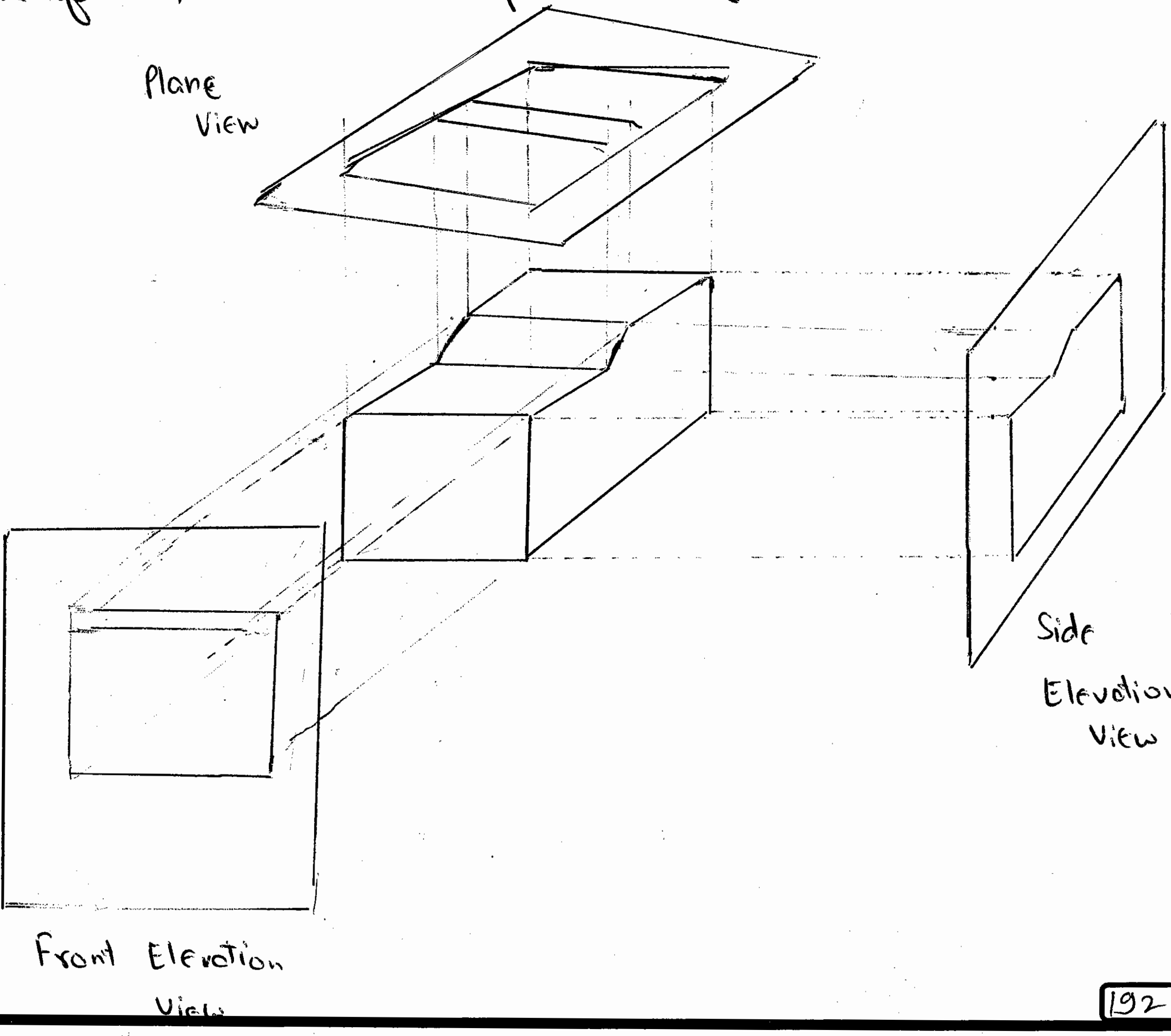
Shankar R
Asst Professor,
CSE, BMSIT&M

Orthogonal Projections

A transformation of Object description to a view plane along lines that are all parallel to the view-plane normal vector N is called an orthogonal projection also termed as orthographic projection.

This produces a parallel-projection transformation in which the projection lines are perpendicular to the view plane.

Orthogonal projections are most often used to produce the front, side and top view of an object.



Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

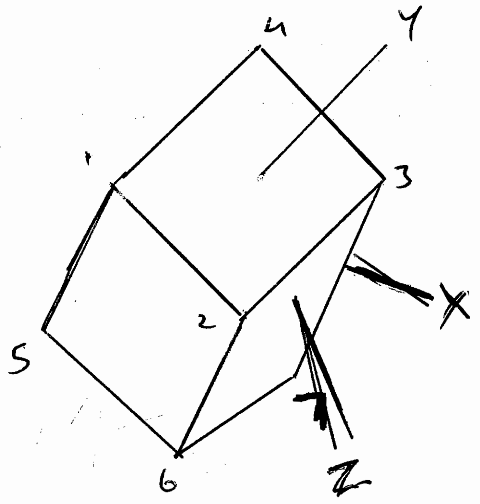
Shankar R
Asst Professor,
CSE, BMSIT&M

Front side and rear ~~orthogonal~~ orthogonal projections of an object are called elevations; and a top orthogonal projection is called a plane view.

Axonometric and Isometric Orthogonal Projections

We can also form orthogonal projections that display more than one face of an object. Such views are called axonometric orthogonal projections.

The most commonly used axonometric projection is the isometric projection, which is generated by aligning the projection plane so that the plane intersects each coordinate axis in which the object is defined, called the principal axes, at the same distance from the origin.



For any position (x, y, z) in viewing coordinates, the projection coordinates are $x_p = x$, $y_p = y$

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

PERSPECTIVE - PROJECTION TRASNFORMATION COORDINATES

ABOUT PERSPECTIVE PROJECTION :

In the real world, reflected light rays from the objects follow converging paths to the view plane. These projections are called perspective projections & the converging point is called the projection reference point / centre of projection.

TRANSFORMATION COORDINATES :

Consider a spatial position (x, y, z) to a general projection reference point (PRP) at the coordinate position $(x_{PRP}, y_{PRP}, z_{PRP})$.

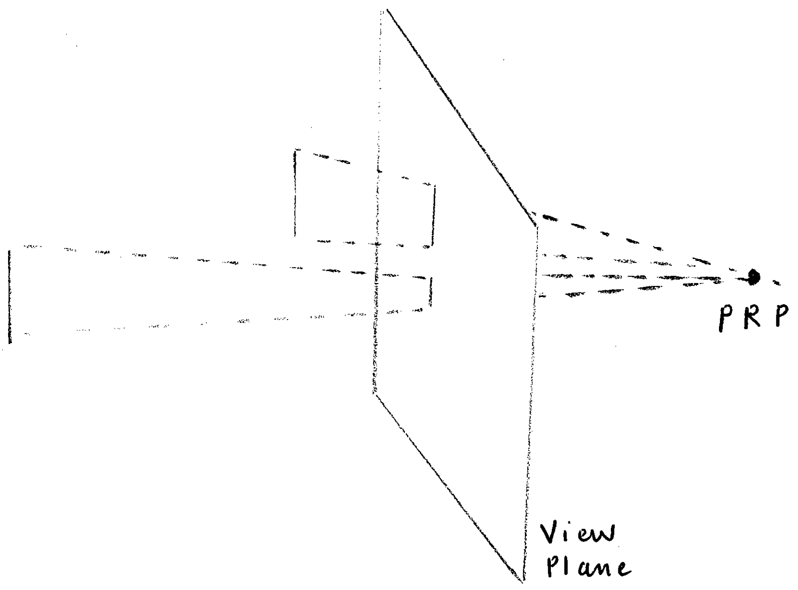
The projection line intersects the view plane at the point (x_p, y_p, z_{vp}) where z_p is some selected position for the view plane.

Thus, we get the parametric equations for an arbitrary point (x', y', z') along the projection line as follows:

$$x' = x - (x - x_{PRP})u$$
$$y' = y - (y - y_{PRP})u$$
$$z' = z - (z - z_{PRP})u$$

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M



when $u=0$ we are position $P=(x,y,z)$

At the other end of the line,

$u=1$ & we have the projection referance point

coordinates $(x_{PRP}, y_{PRP}, z_{PRP})$

We solve for z' on the view plane along the projection line as follows :-

$$u = \frac{z_{vp} - z}{z_{PRP} - z}$$

Subtituiting values of u in eq^n of x' & y' :-

$$x_p = x\left(\frac{z_{PRP} - z_{vp}}{z_{PRP} - z}\right) + x_{PRP}\left(\frac{z_{vp} - z}{z_{PRP} - z}\right)$$

$$y_p = y\left(\frac{z_{PRP} - z_{vp}}{z_{PRP} - z}\right) + y_{PRP}\left(\frac{z_{vp} - z}{z_{PRP} - z}\right)$$

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

SPECIAL CASES:

1. $x_{PRP} = y_{PRP} = 0$:

$$x_p = x\left(\frac{z_{PRP} - z_{vp}}{z_{PRP} - z}\right), \quad y_p = y\left(\frac{z_{PRP} - z_{vp}}{z_{PRP} - z}\right) \rightarrow ①$$

We get ① when the projection reference point ~~could~~ is limited to positions along the z_{view} axis.

2. $(x_{PRP}, y_{PRP}, z_{PRP}) = (0, 0, 0)$:

$$x_p = x\left(\frac{z_{vp}}{z}\right), \quad y_p = y\left(\frac{z_{vp}}{z}\right) \rightarrow ②$$

We get ② when the projection reference point is fixed at coordinate origin.

3. $z_{vp} = 0$:

$$x_p = x\left(\frac{z_{PRP}}{z_{PRP} - z}\right) - x_{PRP}\left(\frac{z}{z_{PRP} - z}\right) \rightarrow ③a$$

$$y_p = y\left(\frac{z_{PRP}}{z_{PRP} - z}\right) - y_{PRP}\left(\frac{z}{z_{PRP} - z}\right) \rightarrow ③b$$

We get 3a & 3b if the view plane is the uv plane & there are no restrictions on the placement of the projection reference point.

4. $x_{PRP} = y_{PRP} = z_{vp} = 0$

$$x_p = x\left(\frac{z_{PRP}}{z_{PRP} - z}\right), \quad y_p = y\left(\frac{z_{PRP}}{z_{PRP} - z}\right) \rightarrow ④$$

We get ④ with the uv plane as the view plane & the projection reference point on the z_{view} axis.

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

78) Explain the OpenGL 3D viewing functions?

→ i) OpenGL Viewing - Transformation Function:

When we designate the viewing parameter in OpenGL, a matrix is formed and concatenated with the current modelview matrix. Consequently, this viewing matrix is combined with any geometric transformations we may have also specified. This composite Matrix is then applied to transform object descriptions in world co-ordinates to viewing co-ordinates.

We set the modelview mode with the statement.

glMatrixMode (GL_MODELVIEW);

Viewing parameter are specified with the following GLU functions, which is in OpenGL Utility library because it invokes the ~~transf~~ translation and rotation routines in the basic OpenGL library.

gluLookAt (x0, y0, z0, xref, yref, zref, Vx, Vy, Vz);

This functions designates the origin of the viewing reference frame as the world-co-ordinate position $P_0 = (x_0, y_0, z_0)$, the reference position as $P_{ref} = (x_{ref}, y_{ref}, z_{ref})$ and the view-up vector as $V = (V_x, V_y, V_z)$.
The positive zview axis for the viewing frame is in the direction $N = P_0 - P_{ref}$ and the unit axis vectors for the viewing reference frame is in the direction $N = P_0 - P_{ref}$. are calculated with equation.
Viewing paramete specified with the gluLookAt function are used to form the viewing transformation matrix. that we derived in Section.
This matrix is formed as a combination of a translation, which shifts viewing origin to the world origin. and a rotation.

Scanned by CamScanner

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

If we do not invoke the gluLookAt function, the default OpenGL viewing parameters are:

$$P_0 = (0, 0, 0)$$
$$P_{ref} = (0, 0, -1)$$
$$V = (0, 1, 0)$$

For these default values viewing reference is same as the world frame, with the viewing direction along negative z_{world} axis.

ii) <u>OpenGL Orthogonal - Projection Function:</u>

→ when we issue any transformation command, the resulting matrix will be concatenated with the current projection matrix.

Orthogonal - projection parameters are choosen with the function

glOrtho (xw_{min}, xw_{max}, yw_{min}, yw_{max}, dnear, dfar);

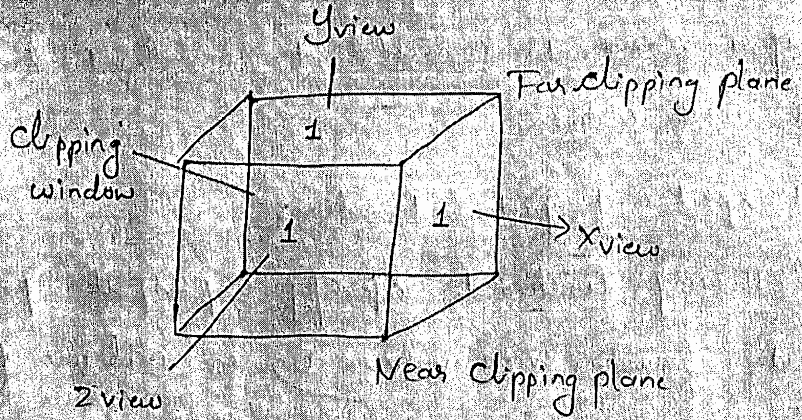
* All parameter values in this function are to be assigned double - precision floating point numbers.
* Parameters dnear and dfar denote distances in the negative z_{view} direction from the viewing ~~-co~~coordinates origin. for ex: if $d_{far} = 55.0$, then the far clipping plane is at the co-ordinates position $z_{far} = -55.0$. A negative value for either parameter denotes a distance "behind" the viewing origin, along the positive z_{view} axis. We can assign any values (positive, negative, or zero) to these parameters as long as ~~dlen~~ $d_{near} < d_{far}$.
* Co-ordinate positions within this view volume are transformed to locations within the symmetric normalized cube in a left-handed reference frame using matrix, with $z_{near} = -d_{near}$ and $z_{far} = -d_{far}$.
* Default parameter values for the OpenGL orthogonal - projection function are ±1, which produce a view volume that is a symmetric normalized cube in the right - handed viewing - co-ordinate system. This default is equivalent to issuing the statement..

Scanned by CamScanner

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

glOrtho (-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);



The default clipping window is thus a symmetric normalized square, and the default view volume is a symmetric normalized cube with $z_{near} = 1.0$ and $z_{far} = -1.0$.

iii) OpenGL Symmetric Perspective - Projection Function:

→ There are two function available for producing a perspective - projection view of a scene. One of these function generates a symmetric frustum view volume about the viewing direction (the negative Zview axis). The other function can be used for either a symmetric - perspective projection or an oblique - perspective projection.

A symmetric perspective - projection, frustum view volume is set up with the GLU function

gluPerspective (theta, aspect, dnear, dfar);

with each of the four - parameter assigned a double - percision floating point number. The first two parameter define the size and position of the clipping window on the near plane, and the second parameter. Specify the distance from the view point to the near and far clipping planes. This angle can be assigned any value from 0° to 180° ~~180~~.

Parameter ascept is assigned a value for the aspect ratio (width / height) of the clipping window

Scanned by CamScanner

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

The frustum view volume set up by the gluPerspective function is symmetric the about the negative zview axis.
And the description of a scene is converted to normalized, homogenous projection co-ordinates with matrix.

iv) <u>OpenGL General Perspective Projection Function:</u>

→ We can use the following function to specify a perspective projection that has either a symmetric frustum view volume or an oblique frustum view volume.

glFrustum (xw_{min}, xw_{max}, yw_{min}, yw_{max}, dnear, dfar);

All the parameter in this functions are assigned double-precision, floating point numbers. This function has the same parameter as the orthogonal parallel-projection function, but now the near and far clipping-plane distance must be positive. The first and the last two parameter specify the distances from the co-ordinate origin to the near and far planes along negative zview axis.

Locations for the near and far planes are calculated as

$z_{near} = -d_{near}$ and $z_{far} = -d_{far}$

If we select the clipping window co-ordinates so that $xw_{min} = -xw_{max}$ and $yw_{min} = -yw_{max}$, we obtain a symmetric frustum about the negative zview axis as its centerline.

Scanned by CamScanner

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

v) OpenGL Viewports and Display windows:

→ After the clipping routines have been applied in normalized co-ordinates, the contents of the normalized clipping window, along with the depth information are transferred to the 3D screen co-ordinates.
The color value for each xy position on the viewport is stored in the refresh buffer and the depth information for each xy position is stored in the depth buffer.
As we noted in, a rectangular viewports is defined with the following OpenGL function

glViewport (xvmin, yvmin, vpwidth, vpHeight);

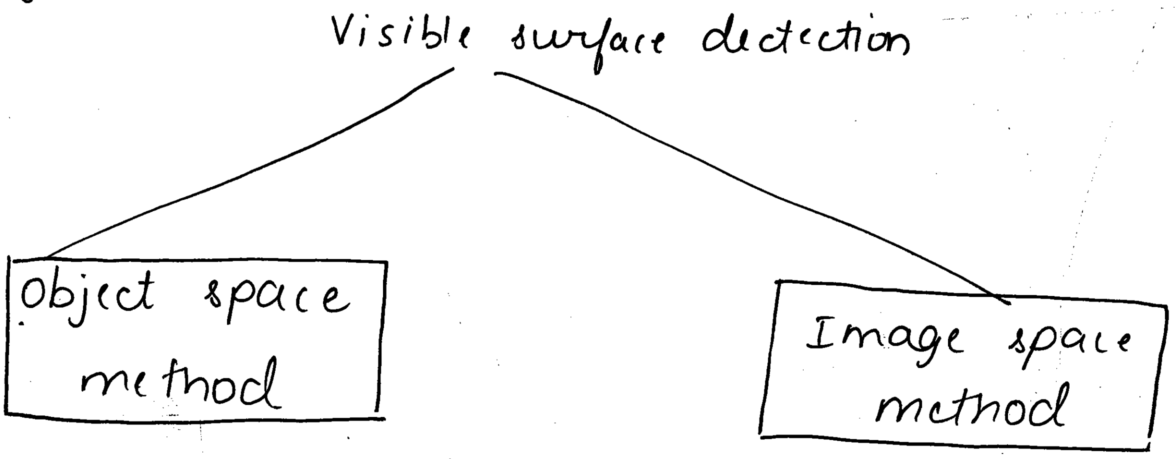
The first two paramder in this function specify the integer screen position of the lower-left corner of the view port relative to the lower-left corner of the display window. And the last two paramder give the integer width and height of the viewport.

• To maintain the proportional of objects in a Scene, we set the ~~ascep~~ ascepts ratio of the Viewport equal to the aspect ratio of the clipping window.

Scanned by CamScanner

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

79 Classify visible surface detection algorithm



→ compares obj & parts of obj to each other with the scene defination to determine which surfaces are vi~~s~~sible

→ Here visibility is decided point by point at each pixel position on the projection plane

→ Eg Back Face Detection algorithm

→ Eg Depth Buffer method.

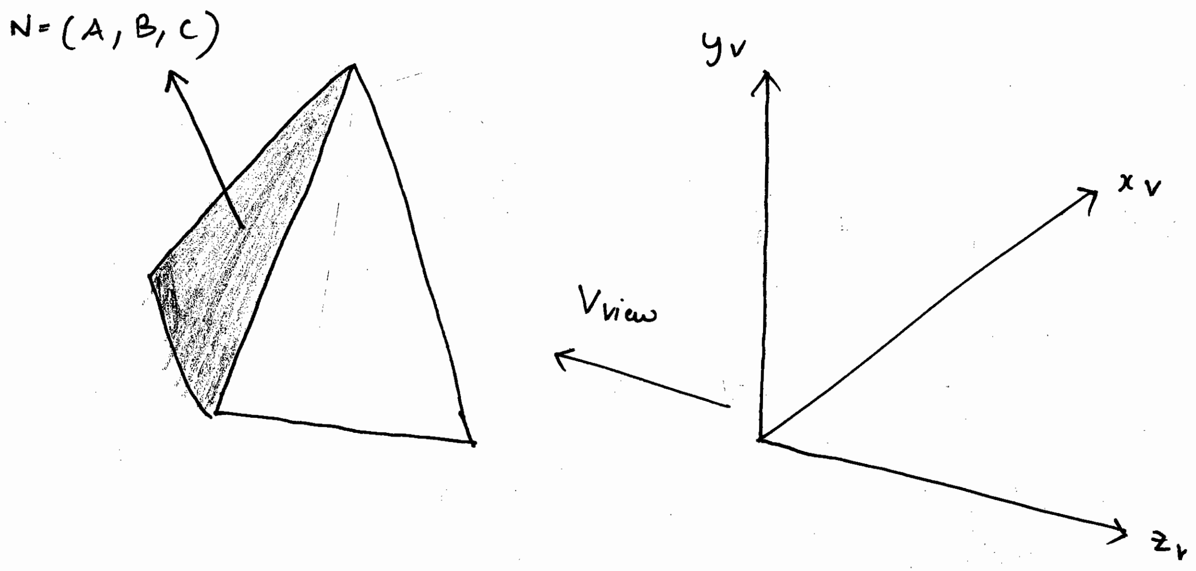
80 Explain Back Face detection algorithm

→ A point (x, y, z) is behind a polygon surface if $\boxed{Ax + By + Cz + D < 0}$

A, B, C, D are plane parameters

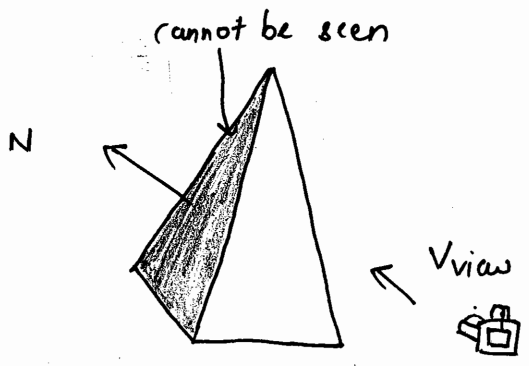
→ We can simplify this test by considering the normal vector N to a polygon surface, which has cartesian component (A, B, C)

→ In general if V is a vector in the viewing direction from the eye position then this polygon is a back face if $\boxed{V_{view} \cdot N > 0}$



→ we will also be unable to see surface with $C = 0$. therefore we can identify a polygon surface as back-face if

$$\boxed{C \leq 0}$$



→ Back-face detection can identify all the hidden surfaces in a scene that contains non-overlapping convex polyhedra

→ But we have to apply more tests that contain overlapping objects along the line of sight to determine which objects obscure with other objects

Shankar R
Asst Professor,
CSE, BMSIT&M

Module-4.

81] Explain the z-buffer / depth-buffer algorithm.

→ A commonly used image-space approach for detecting visible surfaces is the depth-buffer method, which compares surface depth values throughout a scene for each pixel position on the projection plane

→ The algorithm is usually applied to scenes containing only polygon surfaces, because depth values can be computed very quickly and the method is easy to implement.

→ This visibility-detection approach is also frequently alluded to as the z-buffer method because object depth is usually measured along the z-axis of a viewing system.

Depth-Buffer Algorithm

1. Initialize the depth buffer and frame buffer so that for all buffer positions (x, y),

   depthBuff(x, y) = 1.0 , frameBuff(x, y) = backgndColor.

2. Process each polygon in a scene, one at a time, as follows:
   - For each projected (x, y) pixel positions of a polygon, calculate the depth z (if not already known)
   - If z < depthBuff(x, y), compute the surface color at that position and set

     depthBuff(x, y) = z , frameBuff(x, y) = surfColor(x, y)

After all surfaces have been processed, the depth buffer contains depth values for the visible surfaces and the frame buffer contains the corresponding color values for those surfaces

→ Given the depth values for the vertex positions of any polygon in a scene, we can calculate the depth at any other point on the plane containing the polygon.

→ At surface position (x.y) the depth is calculated from the plane equation as

$$z = \frac{-Ax - By - D}{C.}$$

→ If the depth of position (x, y) has been determined to be z, then the depth z' of the next position (x+1, y) along the scan line is obtained as

$$z' = \frac{-A(x+1) - By - D}{C}$$

$$z' = z - \frac{A}{C}$$

→ The ratio -A/C is constant for each surface, so succeeding depth values across a scan line are obtained from preceding values with a single addition.

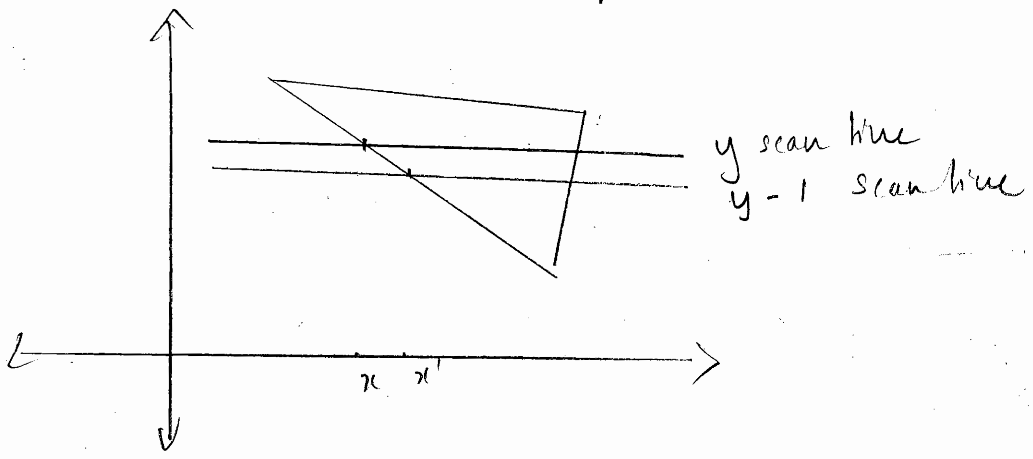
→ We can implement the depth-buffer algorithm by starting at a top vertex of the polygon.

→ Then, we could recursively calculate the x-coordinate values down a left edge of the polygon.

→ The x value for the begining position on each scan line can be calculated from the beginning (edge) x value of the previous scan line as

$$x' = x - 1/m$$

where m is the slope of the edge



→ Depth values down this edge are obtained recursively as

$$z' = z + \frac{A/m + B}{C}$$

→ If we are processing down a vertical edge, the slope is infinite and the recursive calculations reduce to

$$z' = z + \frac{B}{C}$$

→ One slight complication with this approach is that while pixel positions are at integer (x, y) co-ordinates, the actual point of intersection of a scan line with the edge of a polygon may not be.

Shankar R
Asst Professor,
CSE, BMSIT&M

82] Explain the OpenGL visibility detection functions.

a) OpenGL Polygon - Culling Functions

Back-face removal is accomplished with the functions

glEnable (GL_CULL_FACE);

gl CullFace (mode);

→ where parameter mode is assigned the value GL_BACK, GL_FRONT, GL_FRONT_AND_BACK.

→ By default, parameter mode in glCullFace function has the value GL_BACK.

→ the culling routine is turned off with

glDisable (GL_CULL_FACE);

b) OpenGL Depth-Buffer Functions

→ To use the OpenGL depth-buffer visibility-detection routines, we first need to modify the GL Utility Toolkit (GLUT) initialization function for the display mode to include a request for the depth buffer, as well as for the refresh buffer

glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);

→ Depth buffer values can be initialized with

glClear(GL_DEPTH_BUFFER_BIT);

* By default it is set to 1.0.

→ These routines are activated with the following function:

glEnable (GL_DEPTH_TEST);

And we deactivate these depth-buffer routines with

glDisable (GL_DEPTH_TEST);

→ We can also apply depth-buffer testing using some other initial value for the maximum depth,

glClearDepth (maxDepth);

* It can be set to any value btw 0.0 and 1.0.

→ As an option, we can adjust normalization values with

glDepthRange (nearNormDepth, farNormDepth);

* By default, nearNormDepth = 0.0 and farNormDepth = 1.0.

→ We specify a test condition for the depth buffer routines using the following function.

glDepthFunc (testCondition)

- Parameters that can be used are;
  GL_LESS, GL_GREATER, GL_EQUAL, GL_NOTEQUAL, GL_LEQUAL, GL_GEQUAL, GL_NEVER AND GL_ALWAYS
- default parameter is GL_LESS

→ We can set the status of the depth buffer so that it is in a read-only state or in a read-write state.

glDepthMask (writeStatus);

- When writeStatus = GL_TRUE (default value), we can both read from and write to the buffer. and we can retrieve values from buffer if it is set to GL_FALSE.

Shankar R
Asst Professor,
CSE, BMSIT&M

c] OpenGL Wire-Frame Surface Visibility Methods

→ A wire-frame display of a standard graphics object can be obtained in OpenGL by requesting that only its edges are to be generated.

→ We do this by setting the polygon-mode function as for example:

glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)

But this displays both visible and hidden edges.

d] OpenGL- DEPTH-Cueing Function

→ We can vary the brightness of an object as a function of its distance from the viewing position with

glEnable (GL_FOG);
glFogi (GL_FOG_MODE, GL_LINEAR)

→ This applies the linear depth function to object colors using dmin = 0.0 and dmax = 1.0. We can set different values for dmin and dmax with the following

glFogf (GL_FOG_START, minDepth);
glFogf (GL_FOG_END, maxDepth);

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

Q83) Explain in detail, oblique and symmetric perspective projection frust~~r~~um.

Ans)

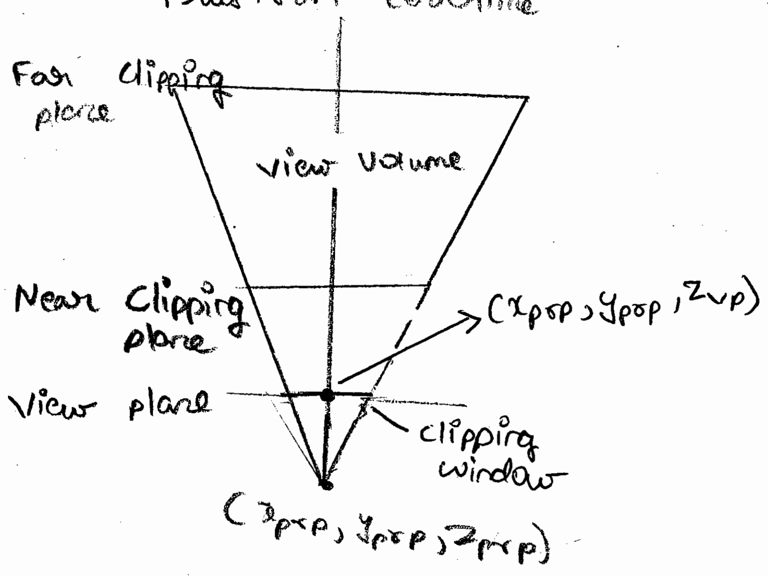
Symmetric perspective-projection frust~~r~~um:

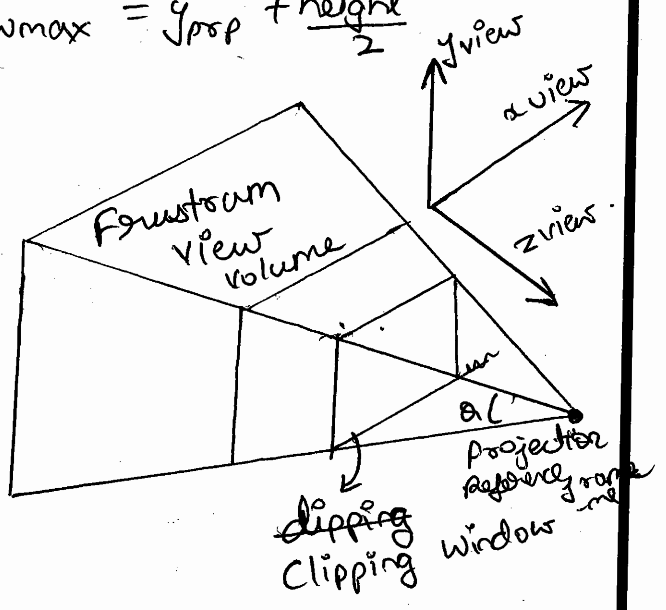
The line from the projection reference point through the center of the clipping window & on through the view volume is the centerline for a perspective projection frust~~r~~um. If this centerline is perpendicular to view plane, the projection is of symmetric frust~~r~~um.

The frustum centerline intersects the viewplane at co-ordinate location $(x_{prp}, y_{prp}, z_{vp})$. The corner positions of clipping window are:

$x_{wmin} = x_{prp} - \frac{width}{2}$ $\quad$ $x_{wmax} = x_{prp} + \frac{width}{2}$

$y_{wmin} = y_{prp} - \frac{height}{2}$ $\quad$ $y_{wmax} = y_{prp} + \frac{height}{2}$

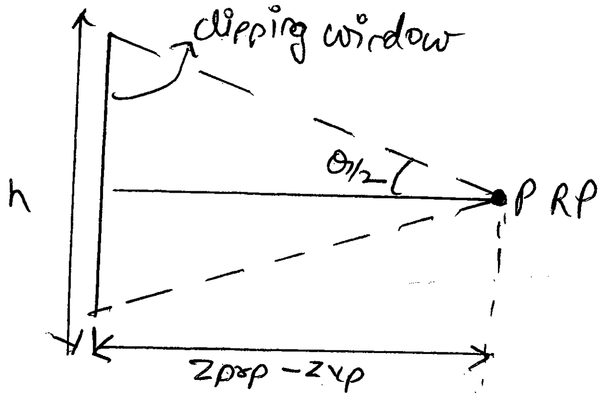




Another way to specify a symmetric perspective projection is to use parameters / properties of camera lens. A photograph is produced with symmetric PP of a scene onto a film plane. Reflected light rays from the objects in a scene are collected on film plane from within the "cone of vision" of the camera. This cone of vision is referenced with ~~a~~ field of view angle - measure of size of camera lens.

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io

Field of view angle → angle b/w the top clipping plane and bottom clipping plane of frustrum.



$$\text{aspect ratio} = \frac{\text{width}}{\text{height}}$$

$$\tan\left(\frac{\theta}{2}\right) = \frac{\text{height}/2}{z_{prp} - z_{vp}}$$

$$\text{height} = 2(z_{prp} - z_{vp})\tan\left(\frac{\theta}{2}\right)$$

'$z_{prp} - z_{vp}$' can be expressed as

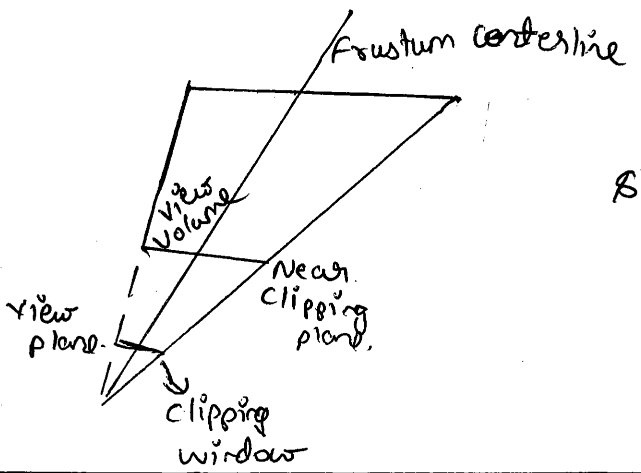
$$z_{prp} - z_{vp} = \frac{\text{height}}{2}\cot\left(\frac{\theta}{2}\right)$$

$$= \frac{\text{width}}{2 \cdot \text{aspect}} \cdot \cot\left(\frac{\theta}{2}\right)$$

Oblique perspective-projection Frustrum

If the center-line of a perspective-projection view volume is not perpendicular to the viewplane - the perspective projection is of oblique frustrum.

An oblique PP view volume can be converted to a symmetric frustrum by applying Z-axis shearing transformation matrix. It shifts all position on any plane that is perpendicular to the z-axis by an amount that is propositional to the distance of the plane from a specified z-axis reference ~~poss~~ position i.e. shift by an amount that will move the center of clipping window to position (x_{prp}, y_{prp}) on the view plane. Hence the centerline is adjusted such that it is perpendicular to the view plane.



Taking PRP as $(x_{prp}, y_{prp}, z_{prp}) = (0,0,$

Shearing matrix is

$$M_{zshear} = \begin{bmatrix} 1 & 0 & sh_{zx} & 0 \\ 0 & 1 & sh_{zy} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If the view plane is at the position of the near clipping plane then $z_{xp} = z_{near}$ & $(x_p, y_p) = (0, 0)$

$$\begin{bmatrix} x_p \to 0 \\ y_p \to 0 \\ z_{xp} \to z_{near} \\ 1 \to 1 \end{bmatrix} = M_{zshear} \cdot \begin{bmatrix} \frac{x_{wmin} + x_{wmax}}{2} \\ \frac{y_{wmin} + y_{wmax}}{2} \\ z_{near} \\ 1 \end{bmatrix} \begin{matrix} \to x \text{ direction of view plane} \\ \to y \text{ direction of view plane} \\ \\ \to z \text{ direction of view plane.} \end{matrix}$$

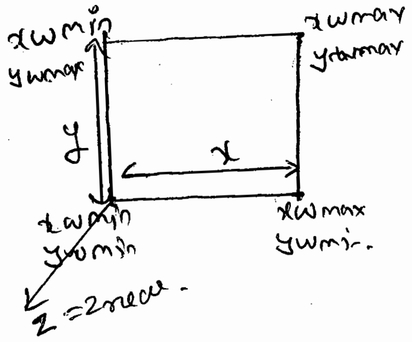
$$\downarrow$$

$$\frac{x_{wmin} + x_{wmax}}{2} + sh_{zx} \cdot z_{near}$$

$$\boxed{sh_{zx} = \frac{-(x_{wmin} + x_{wmax})}{2 \cdot z_{near}}}$$

Similarily,

$$\boxed{sh_{zy} = \frac{-(y_{wmin} + y_{wmax})}{2 \cdot z_{near}}}$$



Similarily, the perspective projection matrix when PRP is at viewing origin $(x_{prp}, y_{prp}, z_{prp}) = (0, 0, 0)$ and view plane is at the near clipping plane $z_{xp} = z_{near}$, the matrix then looks like:

$$M_{pres} = \begin{bmatrix} -z_{near} & 0 & 0 & 0 \\ 0 & -z_{near} & 0 & 0 \\ 0 & 0 & s_z & t_z \\ 0 & 0 & -1 & 0 \end{bmatrix} \Bigg| \begin{bmatrix} -z_{near} & 0 & \frac{x_{wmin} + x_{wmax}}{2} & 0 \\ 0 & -z_{near} & \frac{y_{wmin} + y_{wmax}}{2} & 0 \\ 0 & 0 & s_z & t_z \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

finally, $\boxed{M_{obliquepers} = M_{pres} \cdot M_{zshear}}$

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

8] Explain vanishing Points for Perspective Projections.

* The Point at which a set of projected Parallel lines appears to converge is called vanishing Point

* Each set of Projected Parallel lines has a separate vanishing Point.

* For a set of lines that are parallel to one of the Principle axis of an object, the vanishing Point is referred to as a principle vanishing point.

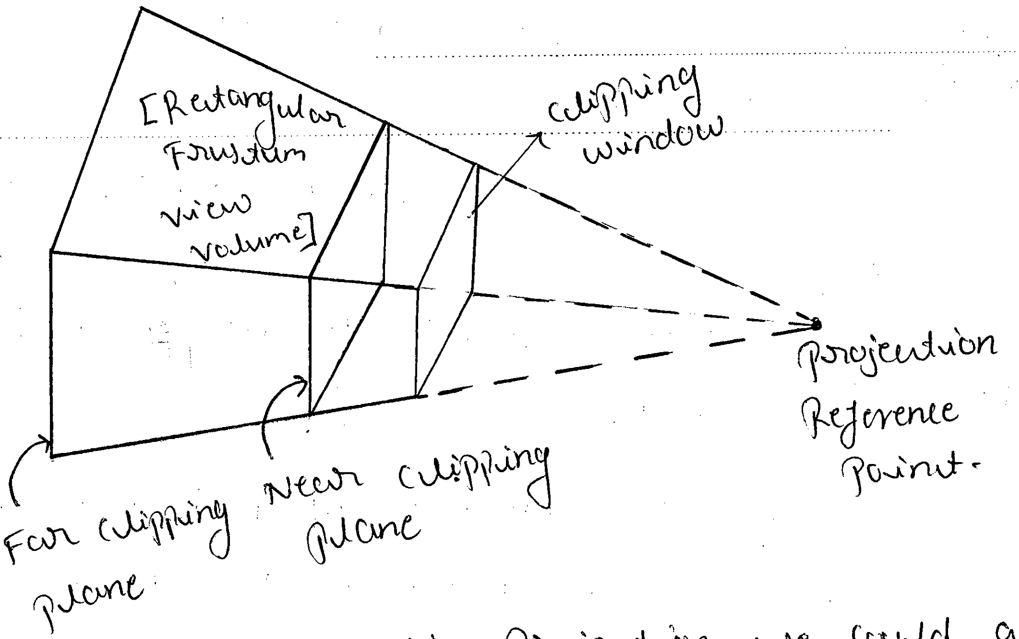
* We control the number of Principal vanishing Points (one, two or three) with the orientation of Projection plane, and prespective Projections are accordingly classified as one-Point two-Point, or Three-Point Projections.

* The displayed view of a scene includes only those objects within the pyramid, just as we cannot see objects beyond our peripheral vision, which are outside the cone of vision.

* By adding near and far clipping planes that are perpendicular to the zview axis (and parallel to view plane) we chop off parts of the infinite, Perspective projection view volume to

Shankar R
Asst Professor,
CSE, BMSIT&M

from a truncated Pyramid or frustum view volume.


[Rectangular Frustum view volume]
clipping window
Projection Reference Point.
Far clipping plane
Near clipping plane

* But with a prespective projection, we could also use the near clipping plane to take out large objects close to the view plane that could project into unrecognized shapes within the clipping window.

* Similarly, the far clipping plane could be used to cut out objects far from the projection references point that might project to small blots on the view plane.

Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

85. Explain briefly the following

b) Depth Cueing:

- Depth information is important in a three-dimensional scene so that we can easily identify, for a particular viewing direction, which is the front and which is the back of each displayed objects.
- There are several ways in which we can include depth information in the two-dimensional representation of solid objects.
- A simple method for indicating depth with wire frame displays is to vary the brightness of line segments according to their distances from the viewing position which is termed as depth cueing.
- A wire frame object displayed with depth cueing so that the brightness of lines decreases from the front of the object to the back.
- The lines closest to the viewing position are displayed with the highest intensity, and lines farther away are displayed with decreasing intensities.
- Depth cueing is applied by choosing a maximum and a minimum intensity value and a range of distances over which the intensity is to

Shruti. P
1BY16CS078

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

vary.

- Another application of depth cueing is modeling the effect of the atmosphere on the perceived intensity of objects.

d) Surface Rendering:

- We set the lighting conditions my specifying the color and location of the light sources, and we can also set background illumination effects
- Surface properties of objects include whether a surface is transparent or opaque and whether the surface is smooth or rough.
- We set values for parameters to model surfaces such as glass, plastic, wood-grain patterns, and the bumpy appearance of an orange.
- Rendering the object surfaces using lighting conditions in scene and assigned surface characteristics.
- Surface rendering is combined with perspective and visible surface identification to generate a degree of realism.

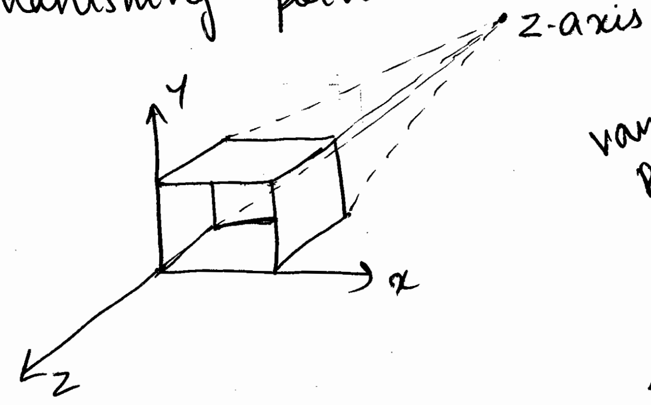
Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

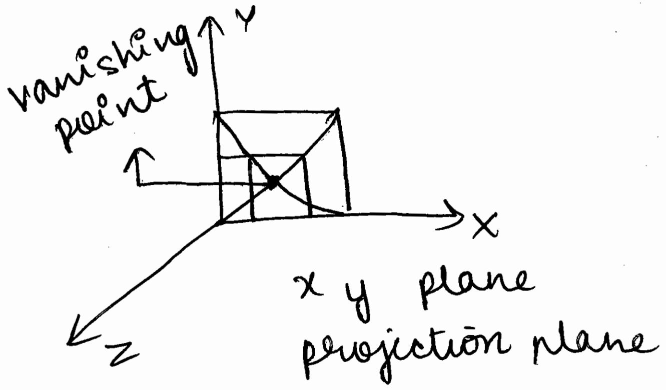
Shankar R
Asst Professor,
CSE, BMSIT&M

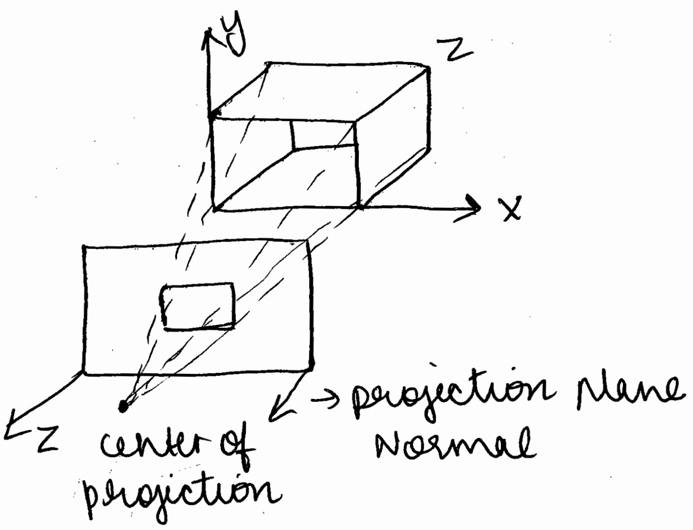
a) Projections :

Projection of a 3D object is defined by straight projection rays elaminating from the center of projection passing through each point of the object and intersecting the projection plane.

Perspective Projections : The perspective projections of any set of parallel lines that are not parallel to the projection plane converge to a vanishing point



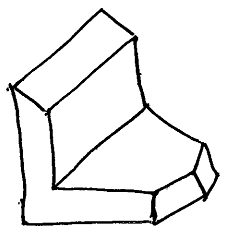


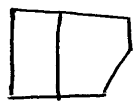


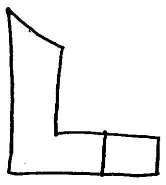
Distant from COP to projection plane is finite. The projections are not parallel and we specify a center of projection. Centre of projection is also called Perspective reference point.

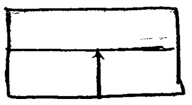
Shruti. P
1BY16CS078

Parallel projection: project the points on the object surface along parallel lines onto a view plane.









c) Identifying visible lines and surfaces:

- One approach is simply to highlight the visible lines or to display them in a different color.
- Another technique, commonly used for engineering drawings, is to display the non visible lines as dashed lines, or we could remove the nonvisible lines from the display.
- When realistic scene is to be produced the back parts of objects are completely eliminated so that only visible surfaces are displayed.

e) Exploded and cutaway views:

- Exploded and cutaway views of such objects

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

can then be used to show the internal structure and relationship of the object parts.

- An alternative to exploding an object into its component parts is a cutaway view, which removes part of the visible surfaces to show internal structure.

f) Three - Dimensional and Stereoscopic Viewing:

- Three-dimensional views can be obtained by reflecting a raster image from a vibrating, flexible mirror.
- The vibrations of the mirror are synchronized with the display of the scene on the cathode ray tube (CRT).
- As the mirror vibrates, the focal length varies so that each point in the scene is reflected to a spatial position corresponding to its depth
- Stereoscopic devices present two views of a scene, one for the left eye and the other for the right eye.
- The viewing positions correspond to the eye positions of the viewer. These two views are typically displayed on alternate refresh cycles of

Shruti . P
1BY16CS078

Prof Shankar R https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/

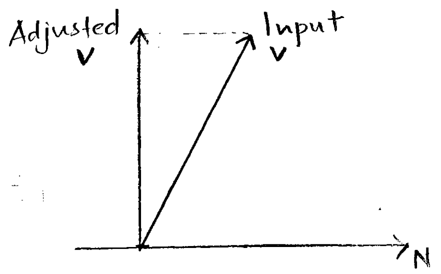
a raster monitor.

Shankar R
Asst Professor,
CSE, BMSIT&M

86. Explain viewup vector and uvn viewing coordinate reference frame.

VIEW UP VECTOR

Ans
- Once we have chosen a view-plane normal vector N, we can set the direction for the view up vector V.
- This vector is used to establish the positive direction for the y_{view} axis.
- Usually, V is defined by selecting a position relative to the world-coordinate origin, so that the direction for the view up vector is from the world origin to this selected position.



- Because the view plane normal vector N defines the direction for the z_{view} axis, vector V should be perpendicular to N.
- But, in general it can be difficult to determine a direction for V that is precisely perpendicular to N.
- Therefore, viewing routines typically adjust the user-defined orientation for vector V.

UVN VIEWING COORDINATE REFERENCE FRAME

- Left handed viewing coordinates are sometimes used in graphics packages, with the viewing direction in the positive z_{view} direction
- with a left handed system, increasing z_{view} values are interpreted as being farther from the viewing position along the line of sight
- But right-handed viewing systems are more common, because they have the same orientation as the world reference frame.
- Because the view plane normal N defines the direction for the z_{view} axis and the view-up vector V is used to obtain the direction for the y_{view} axis, we need only determine the direction for the x_{view} axis.

Shankar R
Asst Professor,
CSE, BMSIT&M

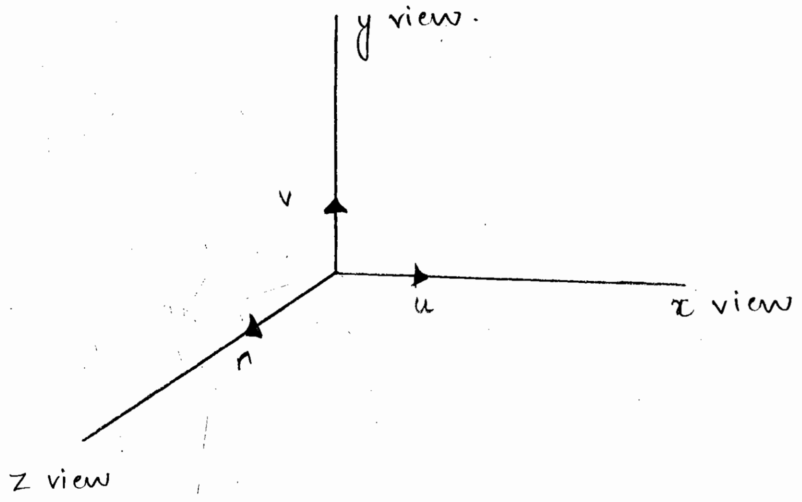
- Using the input values for N and V, we can compute a third vector, U that is perpendicular to both N and V.
- Vector U then defines the direction for positive x_{view} axis.
- We determine the correct direction for U by taking the vector cross product of V and N so as to form a right handed viewing frame.
- The vector cross product of N and U also produces the adjusted value for V perpendicular to both N and U, along positive y_{view} axis.
- We obtain the following set of unit axis vectors for a right handed viewing coordinate system.

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

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

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

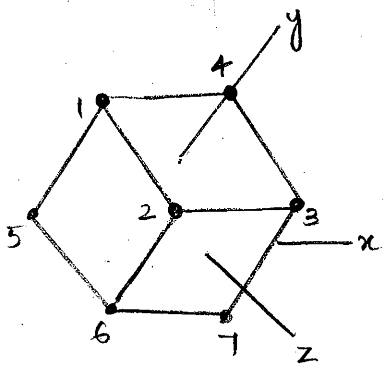
- The coordinate system formed with these unit vectors is often described as an uvn viewing-coordinate reference frame.

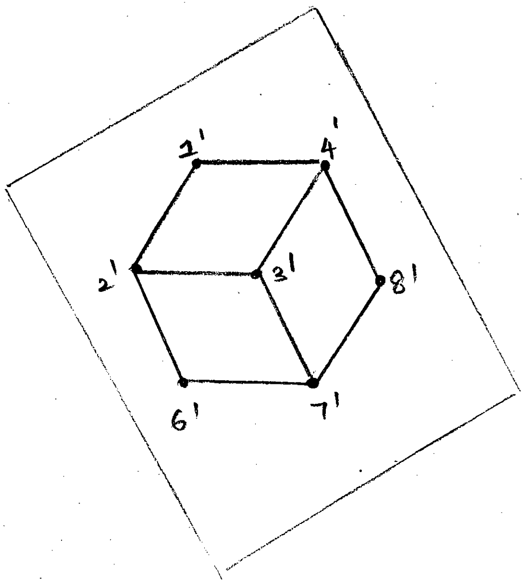


87 Write short notes on axonometric and isometric orthogonal projections.

Ans • We can also form orthogonal projections that display more than one face of an object. Such views are called axonometric orthogonal projections.

• The most commonly used axonometric projection is the isometric projection which is generated by aligning the projection plane (or the object) so that the plane intersects each coordinate axis in which the object is defined, called principal axes, at the same distance from the origin.





Prof Shankar R
https://hemanthrajhemu.github.io
https://shankarrajagopal.github.io/
Shankar R
Asst Professor,
CSE, BMSIT&M

88. Explain OpenGL functions w.r.t:

a. Viewing transformation functions

⇒ glMatrixMode (GL_MODELVIEW):

→ A matrix is formed and concatenated with the current modelview matrix.

→ This viewing matrix is combined with any geometric transformation.

→ This composite matrix is then applied to transform object descriptions in world coordinates to viewing coordinates.

• gluLookAt (x0, y0, z0, xref, yref, zref, V_x, V_y, V_z):

→ Values for all parameters in this function are to be assigned double precision floating point values.

→ This function designates the origin of the viewing reference frame as world coordinate position $P_0 = (x0, y0, z0)$ the reference position as $P_{ref} = (xref, yref, zref)$ and viewup vector as $V = (V_x, V_y, V_z)$.

→ The positive z_{view} axis for the viewing frame is in the direction $N = P_0 - P_{ref}$.

b. Orthogonal projection functions

⇒ glMatrixMode (GL_PROJECTION):

→ Projection matrices are stored in the OpenGL projection

VARSHA.V

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

mode. To set up a projection transformation matrix, we use this function to invoke the mode.

→ Then when we issue any transformation command, the resulting matrix will be concatenated with the current projection matrix.

- glOrtho (xwmin, xwmax, ywmin, ywmax, dnear, dfar):

→ Orthogonal projection parameters are chosen with this function

→ All parameters should be assigned double-precision, floating point values.

→ We use glOrtho to set the clipping window coordinates and the distance to the near and far clipping planes from the viewing origin.

→ The function generates a parallel projection perpendicular to the view plane.

→ Parameters dnear and dfar are not optional, and they denote distances on the negative z_{view} direction from the viewing coordinate origin.

ii. Symmetric Perspective Projection functions

⇒ gluPerspective (theta, aspect, dnear, dfar):

→ The parameters are assigned double precision floating point numbers.

→ The first two parameters define the size and position of the clipping window in the near plane.

→ The second two parameters specify the distances from

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

Shankar R
Asst Professor,
CSE, BMSIT&M

the viewpoint to the ~~near~~ near and far clipping planes.

→ Parameter theta represents field-of-view angle, which is the angle b/w top and bottom clipping planes (value b/w $0°$ to $180°$)

→ Parameter aspect is assigned a value for the aspect ratio of the clipping window.

→ The near and far clipping planes must always be somewhere along the -ve z_{view} axis.

d. General Perspective Projection function

⇒ glFrustum (xwmin, xwmax, ywmin, ywmax, dnear, dfar):

→ This function is used to specify a perspective projection that has either a symmetric frustum view volume or an oblique frustum view volume.

→ All parameters are assigned double-precision, floating point numbers.

→ The near plane is the view plane and the projection reference point is at the viewing position.

→ This function has the same parameters as the orthogonal, parallel projection function, but the near and far clipping planes distance must be positive.

→ The first four parameters set the coordinates for the clipping window on the near plane, and the last two parameters specify the distance from the coordinate origin from the near and far clipping planes along -ve z_{view} axis.

VARSHA.V

Prof Shankar R https://hemanthrajhemu.github.io https://shankarrajagopal.github.io/

e. Viewport and Display Window

⇒ glViewport (xvmin, yvmin, vpWidth, vpHeight);

→ After the clipping routines have been applied in normalised coordinates, the contents of the normalised clipping window are transferred to 3-D screen coordinates.

→ The color value for each xy position on the viewport is stored in the refresh buffer. and depth information is stored in the depth buffer.

→ First two parameters specify integer screen position of lower left corner of the viewport relative to the lower left corner of the display window.

→ last two parameters specify the integer width and height of the viewport.

→ We set aspect ratio of viewport is equal to aspect ratio of clipping window.

Shankar R
Asst Professor,
CSE, BMSIT&M

89) Imagine you have a 3D object in front of you. Illustrate how to Normalize the transformation for an Orthogonal Projection?

Ans: There are 2 approaches to Normalize a 3D object's transformation for an Orthogonal Projection:-

→ Use a unit cube for the normalized view volume, with each of the x, y and z coordinates normalized in the range from 0 to 1.

→ Another normalization-transformation approach is to use a symmetric cube, with co-ordinates in the range from -1 to 1.

Because screen co-ordinates are specified in a left-handed reference frame, normalized coordinates are also specified in left-handed reference frame, this allows positive distances in the viewing direction to be directly interpreted as distance from the screen.

To illustrate the normalization transformation, we assume that the orthogonal-projection view volume is to be mapped into the symmetric normalization cube within a left-handed reference frame. Also, z-coordinate positions for the near and far planes are denoted as z_{near} and z_{far} respectively. this transformation is illustrated in the fig as shown.

Position $(x_{min}, y_{min}, z_{near})$ is mapped to the normalized position $(-1,-1,-1)$, and position $(x_{max}, y_{max}, z_{far})$ is mapped to $(1,1,1)$.

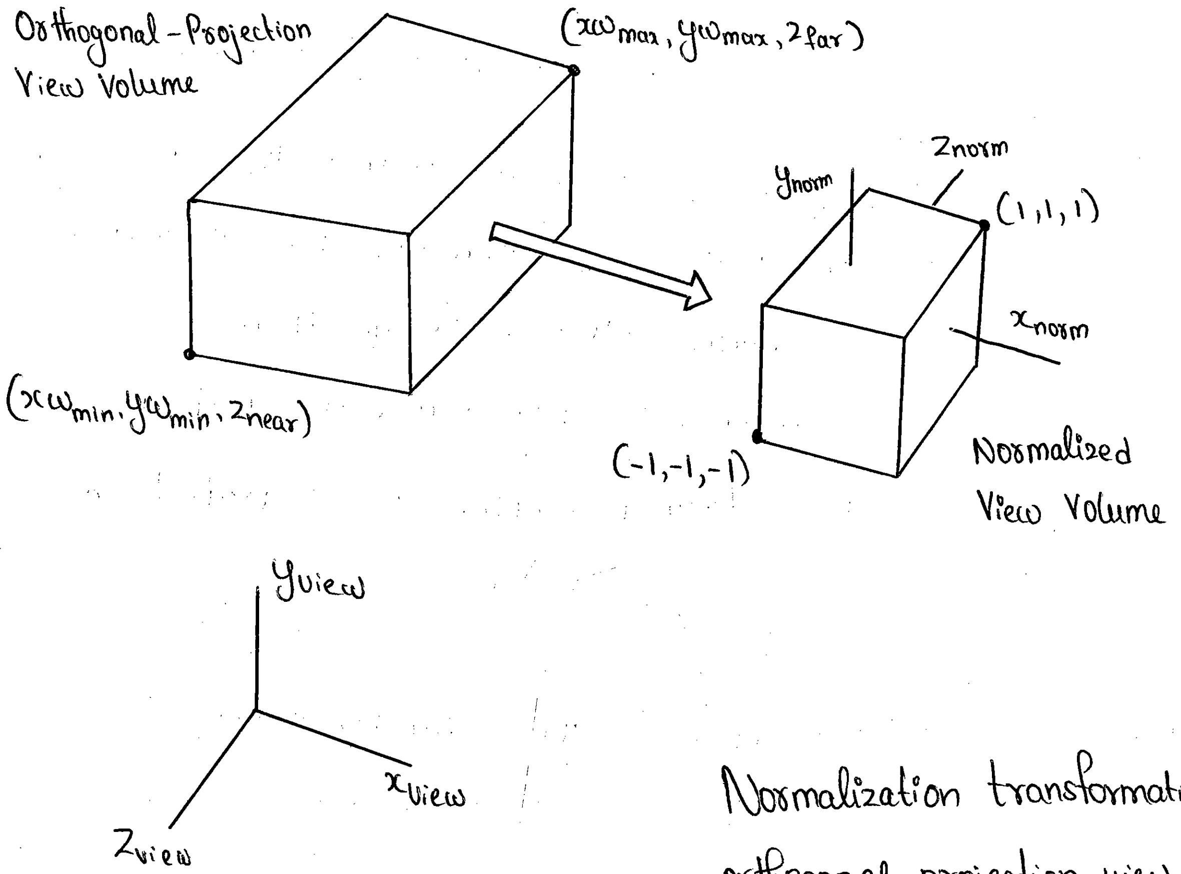
Transforming the rectangular-parallelepiped view volume to a normalized cube is similar to the methods for converting

Shankar R
Asst Professor,
CSE, BMSIT&M

the clipping window into the normalized symmetric square. The normalization transformation for the Orthogonal view volume is

$$M_{ortho,norm} = \begin{bmatrix} \frac{2}{xw_{max}-xw_{min}} & 0 & 0 & -\frac{xw_{max}+xw_{min}}{xw_{max}-xw_{min}} \\ 0 & \frac{2}{yw_{max}-yw_{min}} & 0 & -\frac{yw_{max}+yw_{min}}{yw_{max}-yw_{min}} \\ 0 & 0 & \frac{-2}{z_{near}-z_{far}} & \frac{z_{near}+z_{far}}{z_{near}-z_{far}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This matrix is multiplied on the right by the composite viewing transformation R.T to produce the complete transformation from world coordinates to normalized orthogonal-projection coordinates.



Normalization transformation from an orthogonal-projection view volume to the symmetric normalization cube within a left-handed reference frame.