Assignment-4

Q1. Explain the issues related to 3D viewing more complex than simple 2D viewing.

→ 3D viewing, which aims to provide a three-dimensional perception of objects on a two-dimensional display, introduces several challenges that make it more complex than simple 2D viewing. Here are some of the key issues related to 3D viewing:

1) Depth Perception:

→ In 2D viewing, depth is conveyed primarily through cues like perspective, size and shading.

→ However in 3D viewing, the challenge lies in accurately reproducing these depth cues to create a realistic sense of depth.

→ Without proper depth perception cues, the viewer may experience difficulties in perceiving the relative distances & positions of objects in the scene.

2) Stereopsis and Binocular Vision:

→ Stereopsis refers to the process of perceiving depth through binocular vision, where each eye receives a slightly different image due to their horizontal seperation.

→ The brain processes these disparate images to generate depth perception.

→ In 3D viewing, the challenge is to present seperate images to each eye in a way that simulates binocular vision.
→ This can be achieved using various techniques like anaglyphic glasses, polarized glasses, or active shutter glasses.

3) Visual Discomfort and Fatigue
→ Extended periods of 3D viewing can lead to visual discomfort and fatigue.
→ Some viewers may experience eyestrain, headaches or dizziness due to the additional cognitive load involved in processing depth cues.

4) Technical limitations
→ Implementing 3D viewing requires specialized display technologies & content creation techniques.
→ These technologies include stereoscopic displays, auto-stereoscopic displays or VR headset.
→ Content creation for 3D viewing often involves capturing or rendering the scene from multiple viewports or using specialized depth-mapping techniques.
→ These technical requirements add complexity to the production & distribution of 3D content.

5) Content Adaption

→ Not all types of content are suitable for 3D viewing.

→ Certain visual elements, such as fast motion, excessive depth disparities, or intense visual effects, can be challenging to perceive or may cause discomfort in 3D.

6) Limited Viewing Zones

→ Some 3D display technologies have limited viewing zones, meaning that viewers must maintain specific positions or angles to perceive the 3D effect correctly.

→ This constraint restricts the freedom of movement & viewing positions compared to traditional 2D displays.

Q2) How does 2D clipping differ from 3D clipping?

→ 2D clipping and 3D clipping are techniques used in CG to determine which parts of a scene or object should be displayed on a screen or viewport.

→ While they share a similar purpose, there are fundamental differences between the two:

1) Dimensionality

→ The most apparent difference is that 2D clipping operates in a two-dimensional space, while

3D clipping operates in a 3-dimensional space.

→ In 2D clipping, the objects or elements being clipped exist in a flat plane, such as an image or a 2D shape.
In contrast, 3D clipping deals with objects or elements that have depth & occupy a 3-D space, such as 3D models or scenes.

2) Geometric Complexity

→ Due to added dimension, 3D clipping involves more geometric complexity than 2D clipping.

3) View Frustum

→ In 3D graphics, the concept of the view frustum becomes crucial for clipping.
The view frustum represents the portion of the 3D world that is visible within the camera's field of view.

→ 3D clipping involves determining which objects or elements intersect or lie within the view frustum & discarding those that fall outside it.

→ The process is essential for optimizing rendering performance by only processing what is visible to the camera.

4) Perspective Transformation

→ Another key difference is the presence of perspective transformation in 3D clipping.

→ In 2D clipping, the transformation is generally limited to simple scaling & translation operations.

(b) Derive the scaling transformation matrix of a point $P = (x, y, z)$ with respect to a fixed point (x_f, y_f, z_f).

Steps:

1) Translate the fixed point to the origin:

ie. $$T = \begin{bmatrix} 1 & 0 & 0 & -x_f \\ 0 & 1 & 0 & -y_f \\ 0 & 0 & 1 & -z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2) Apply the scaling transformation. Let's denote the scaling factors as (S_x, S_y, S_z). The scaling matrix S can be defined as:

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3) Translate back to the original position.
i.e.

$$T' = \begin{bmatrix} 1 & 0 & 0 & x_f \\ 0 & 1 & 0 & y_f \\ 0 & 0 & 1 & z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Composite matrix, $M = T' * S * T$

ie. $M = \begin{bmatrix} 1 & 0 & 0 & x_f \\ 0 & 1 & 0 & y_f \\ 0 & 0 & 1 & z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_f \\ 0 & 1 & 0 & -y_f \\ 0 & 0 & 1 & -z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$

On simplifying the matrix multiplication, we get;

$$M = \begin{bmatrix} s_x & 0 & 0 & x_f(1-s_x) \\ 0 & s_y & 0 & y_f(1-s_y) \\ 0 & 0 & s_z & z_f(1-s_z) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Thus the scaling transformation matrix with respect to the fixed point (x_f, y_f, z_f) is given by M.

Q4) List the CM for reflecting an object about an arbitrary line in 3D

$$CM = T_{(x_f, y_f, z_f)} \; R_{inv.} \; R_f \; R_{\theta_{ccw}} \; T_{(-x_f, -y_f, -z_f)}$$

$$= \begin{bmatrix} 1 & 0 & 0 & -x_f \\ 0 & 1 & 0 & -y_f \\ 0 & 0 & 1 & -z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix}$$

Q15. How can fractals be generated? What are its properties?

→ Fractals are generated through iterative mathematical processes or algorithms.

→ The basic concept behind fractals is self-similarity, where a pattern repeats itself at different scales. Here are some common methods for generating fractals:

1) Iterated Function Systems (IFS):

→ In this method, a set of affine transformations is applied repeatedly to an initial point or set of points.

→ Each transformation is associated with a probability, & the chosen transformation determines the next point.

→ As the process continues, a fractal shape emerges.

2) Escape-Time Algorithms

→ These algorithms are based on the concept of complex numbers & the behaviour of iterated functions.

→ One well-known example is the Mandelbrot Set, whose points in the complex plane are iterated through a formula.

→ If a point escapes a certain threshold,

It is considered outside the set & assigned a color based on the number of iterations reqd to escape.

- Properties
1) Self-similarity
2) Infinite complexity
3) Fractional Dimension
4) Repetition & Recursive structure
5) Fractional Dimension & Scaling
6) Natural and Ubiquitous Appearance

Fractals have diverse applications, including CGs, digital art, data compression, terrain generation, modeling natural phenomena, & even understanding complex systems in fields like physics, biology, & finance.

Q16) How is 3D viewing pipeline different from 2D viewing pipeline?

→ The 3D viewing pipeline & the 2D viewing pipeline differ in several key aspects due to the additional dimension of depth in 3D graphics.
Here are the main diff betn the two:

1. Coordinate systems

→ In 2D graphics, the coordinate system typically uses two axes (x and y) to represent positions on the screen.

→ In contrast, 3D graphics use a 3-D coordinate system with 3 axes (x, y & z) to represent positions in a 3D space.

2. Perspective Projection

→ One significant difference in the 3D viewing pipeline is the inclusion of ∿.

→ ∿ accounts for the way objects appear smaller as they move farther away from the viewer.

→ The projection transforms 3D coordinates into 2D coordinates with depth cues to create a realistic sense of depth.

3. Depth Buffering

→ In 3D graphics, ∿ also known as z-buffering is commonly used to handle occlusion & determine which objects or surfaces should be visible.

→ It involves comparing the depth (z-coordinate) of each pixel in the scene & keeping track of the closest visible pixel at each point on the screen.

4) Lighting and shading

→ 3D graphics often involve lighting & shading techniques to simulate the interaction of light with objects in a scene, creating a more realistic appearance.

5) Object Transformation & Viewing Frustum

→ The 3D viewing pipeline includes a series of transformations to position & orient objects in the 3D scene.

→ Additionally, the concept of a viewing frustum is introduced to define the volume of the 3D scene that will be rendered.

6) Clipping and Culling

→ ~ operations are more complex in the 3D viewing pipeline compared to 2D graphics.

→ Clipping involves determining which parts of objects or polygons should be displayed based on their intersection with the viewing frustum.

→ Culling techniques are used to discard objects or surfaces that are entirely outside the viewing frustum, optimizing rendering performance.

7) Perspective correction

→ Another difference in the 3D viewing pipeline is the need for ~. Due to the ~, certain properties such as size & texture coordinates, may need to be corrected to account for the distortion caused by the projection.

→ ~ ensures that objects maintain their intended proportions & textures appear correctly.

Q11) Explain the role of Blending function, Convex Hull & Control points used in Beizer curve.

1) Blending function

→ The ~, also known as the Bernstein polynomial, is a mathematical function used in Beizier curves to calculate the influence or weight of each control point on the curve.

→ It determines how the control points are combined to generate the final curve.

→ The ~ is defined for each control point & depends on a parameter, typically denoted as "t", which ranges from 0 to 1.

→ By varying the value of "t", the ~

calculates the contributions of each control point at different positions along the curve.

2. Convex Hull

→ The ∼ of a set of control points is the smallest convex polygon that encloses all the control points.

→ In Bezier curves, the control points lie within the ∼.

→ The ∼ plays a crucial role in determining the shape & behaviour of the curve.

→ The curve generated by the Bezier alg. remains entirely within the convex Hull, & the control points define the boundary & control the curvature of the curve.

3. Control Points

→ The ∼ are the key elements that define the shape of a Bezier curve.

→ They act as handles that influence the curve's trajectory & curvature.

→ In a quadratic BC, there are 3 control points:

- a starting point
- an ending point
- and a control point that determines the shape.

→ In a cubic BC, there are 4 control points:

- starting point, ending point & two intermediate control points.

→ By adjusting the positions & weights of the control points, different curves can be created, allowing for a wide range of shapes & smooth curves.

Q 18 How can objects be represented in 3D using polygon table? How is the consistency of the geometric data table checked & what are the rules for generating error free polygon tables?

→ Objects in 3D can be represented using a polygon table, also known as a polygon mesh or polygonal representation.

→ A polygon table organizes the geometric data of an object into individual polygons, typically triangles or quadrilaterals which collectively form the object's surface.

To represent an object using a polygon table, the following steps are typically followed:

1. Vertex Data

→ The vertices of the object are defined, each having three-dimensional coordinates (x, y, z). These vertices represent the corners or endpoints of the polygons.

2. Polygon Data

→ The polygons are defined by specifying the indices of the vertices that form each polygon.

3. Vertex Normals

→ Optional vertex normals can be assigned to each vertex to define the orientation or facing direction of the polygon.

→ V are crucial for calculating lighting and shading effects on the objects surface.

4. Texture Coordinates

→ If texture mapping is desired, V can be assigned to each vertex.

→ These coordinates define how a 2D texture image is mapped onto the 3D surface of the object, enabling realistic texture rendering.

→ The consistency of the geometric data table can be checked by ensuring that certain rules are followed. These rules aim to prevent errors & ensure the proper representation of the object:

1) Vertex order
→ The vertices of each polygon should be specified in a consistent order, typically either clockwise or counterclockwise.

2) Polygon connectivity
→ The connectivity bet^n polygons should be consistent & free from errors such as missing or duplicate vertices or incorrect vertex indices.

3) Manifold Geometry
→ Each edge of a polygon should be shared by exactly two polygons.

4) Closed surfaces
→ If the object represents a closed surface, care should be taken to ensure that the polygons collectively enclose a volume without any gaps or holes.

Q.2) Explain the 3D windowing process (window to viewport mapping)

The 3D windowing process, also known as window to viewport mapping or 3D clipping & projection, is a crucial step in CGIs that transforms the 3D world coordinates of objects in a scene (specified within a defined window or view volume) into 2D screen coordinates for display on a computer screen or other O/P devices.

Process

1) Defining the 3D window
2) Viewing Transformation
3) Clipping
4) Projection
5) Normalization
6) Viewport Transformation.

Q.12) How is a non-planar surface formed using splines? Explain Bezier surface with equation explain its properties.

Properties

1) Control Points

→ The shape of the Bezier surface is controlled by its control points. Moving a control point will deform the surface locally, allowing for flexible & smooth surface modeling.

2) Interpolation

→ The Bezier surface passes through its four corner control points (P_{00}, P_{0m}, P_{n0} & P_{nm}) known as "convex hull" property. However, the interior control points do not lie on the surface itself but affect its shape.

3) Convex hull property

→ The Bezier surface lies within the convex hull formed by its control points. This means that the surface will not have any holes or self-intersections

4) Boundedness

5) Smoothness

Q4) Compute the necessary coordinates for forming a Bezier curve taking four control points $P_1(10,10)$, $P_2(40,40)$, $P_3(60,60)$ & $P_4(80,10)$ & using 5 line segments

we have,

$$P(u) = P_0 BEZ_{0,3}(u) + P_1 BEZ_{1,3}(u) + P_2 BEZ_{2,3}(u) + P_3 BEZ_{3,3}(u)$$

$$BEZ_{j,n}(u) = \frac{n!}{j!(n-j)!} (u)^j (1-u)^{n-j}$$

So $n = 3$ $\quad nSeg = 5$

$u_0 = \frac{0}{5} = \frac{0}{5} \quad x(0) = \frac{0}{5} = 0$

$u_1 = \frac{1}{nseg} = \frac{1}{5} = 0.2$

$u_2 = \frac{2}{nseg} = \frac{2}{5} = 0.4$

$u_3 = \frac{3}{nseg} = \frac{3}{5} = 0.6$

$u_4 = \frac{4}{nseg} = \frac{4}{5} = 0.8$

$u_5 = \frac{5}{nseg} = \frac{5}{5} = 1$

$x(0) = 10$
$y(0) = 10$

$x(0.2) = 18.75$
$x(0.2) = 18.75$

$x(0.4) = 32.5$
$x(0.4) = 35$

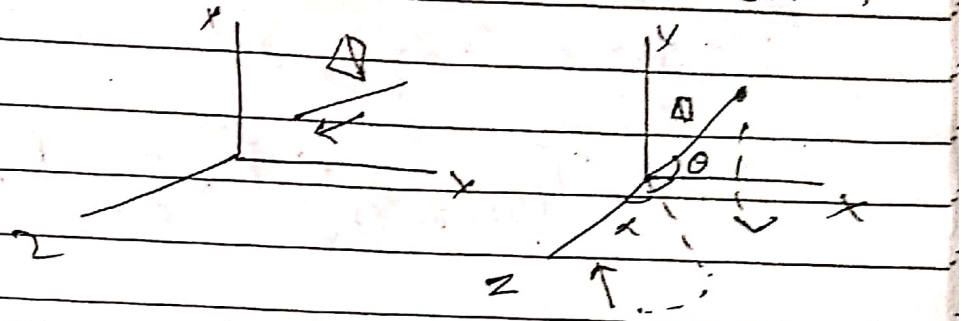
$x(0.6) = 42.5$
$x(0.6) = 42.8$

$x(0.8) = 61.875$
$x(0.8) = 31.875$

$x(1) = 80$
$y(1) = 10$

2. To rotate of an object about any arbitrary axis:

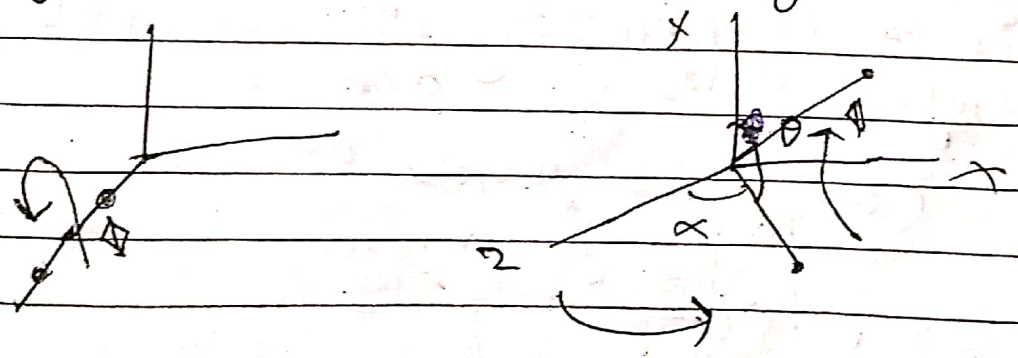
1. Translate the arbitrary axis / line along with the object to be rotated about it so that one of the end points coincides with origin.



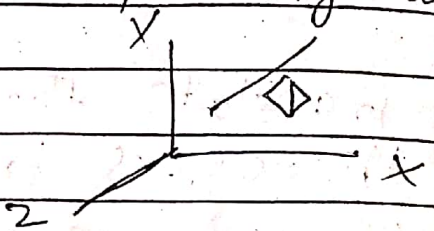
2. Perform rotations about x and y-axis to align the arbitrary axis / line with the positive z-axis.

3. Rotate the object about z-axis by the desired angle.

4. Apply inverse rotations about y then 'x' axis to bring the arbitrary axis / line back to its original position from origin.



5. Apply inverse translation to place the arbitrary axis back in its original position in space.



These transformations can be concatenated into a single composite matrix as:

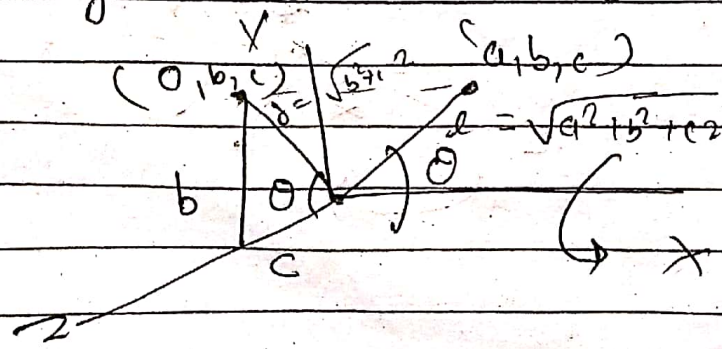
$$M = T_{(x_1, y_1, z_1)} \cdot R_{x_{cw}}^{\theta} \cdot R_{y_{ccw}}^{\phi} \cdot R_z \cdot R_{y_{cw}}^{\phi} \cdot R_{x_{ccw}}^{\theta} \cdot T_{(-x_1, -y_1, -z_1)}$$

The individual matrices in this expression can be found through geometric consideration.

1. Translation is given by $T(-x_1, -y_1, -z_1)$

$$= \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. To establish the rotation matrix about x-axis it is necessary to calculate the angle of rotation 'θ'.



As shown in figure, the line makes an angle 'θ' with xz plane. This angle is found by projecting the arbitrary axis/line onto the yz plane where,

The length of the arbitrary axis/line is $l = \sqrt{a^2+b^2+c^2}$

The length of its shadow $d = \sqrt{b^2+c^2}$, from this projection we get

$$\sin\theta = \frac{b}{\sqrt{b^2+c^2}} = \frac{b}{d} \quad \text{and}$$

$$\cos\theta = \frac{c}{\sqrt{b^2+c^2}} = \frac{c}{d}$$

The matrix for rotating about x-axis in anticlockwise direction by angle 'θ' is given as

$$R_{x\theta}^{ccw} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & -b/d & 0 \\ 0 & b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now this transformation brings the arbitrary axis on the xz plane, the rotation angle 'α' about y-axis is given as $\cos\alpha = \frac{d}{l}$ & $\sin\alpha = \frac{a}{l}$ so the clockwise rotation about 'y' axis is given by the matrix

$$R_{y\alpha_{cw}} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ \sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a/c & 0 & -a/c & 0 \\ 0 & 1 & 0 & 0 \\ a/c & 0 & a/c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. The rotation about arbitrary axis now placed along z-axis is obtained by

$$R_{z\phi_{ccw}} = \begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5. The inverse rotations about y-axis in anticlockwise & about x-axis in clockwise direction is given by:

$$R_{y\alpha_{ccw}} = \begin{bmatrix} a/c & 0 & a/c & 0 \\ 0 & 1 & 0 & 0 \\ -a/c & 0 & a/c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and $R_{x\theta_{cw}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & b/d & 0 \\ 0 & -b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

6. The inverse translation matrix is

$$T(x_1, y_1, z_1) = \begin{bmatrix} 1 & 0 & 0 & x_1 \\ 0 & 1 & 0 & y_1 \\ 0 & 0 & 1 & z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The composite rotation for rotating an object about any arbitrary axis / line is

$$CM = \begin{bmatrix} 1 & 0 & 0 & x_1 \\ 0 & 1 & 0 & y_1 \\ 0 & 0 & 1 & z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & b/d & 0 \\ 0 & -b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} d/e & 0 & a/e & 0 \\ 0 & 1 & 0 & 0 \\ -a/e & 0 & d/e & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

$$\begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d/e & 0 & -a/e & 0 \\ 0 & 1 & 0 & 0 \\ a/e & 0 & d/e & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d & -b/d & 0 \\ 0 & b/d & c/d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projection: The process of transforming points in coordinate system of n dimension into points in a coordinate system of dimension less than 'n'.

Q.6

Projection is required to display object modled in 3D space in a 2D display device.

- **Oblique parallel projection:**

→ It is obtained by projecting points along parallel lines that are not perpendicular to the projection plane.

Often specified with 2 angles 'θ' an 'α'. A point P (x, y, z) is projected to position (xp, yp) on the view plane. Orthographic parallel projection line from (x, y, z) to (xp, yp) makes an angle 'α' with the line on the perpendicular projection plane that joins (xp, yp) and (x, y).

This line of length 'L' is at an angle 'θ' with the horizontal direction in the projection plane. Expressing projection coordinates in terms of x, y, L and θ

as $x_p = x + L\cos\theta$

and $y_p = y + L\sin\theta$

L depends on the angle L and z

coordinate of the point to be projected and

$\tan\alpha = \frac{z}{L}$ thus

$L = \frac{z}{\tan\alpha} = zL_1$ where L_1 is inverse of $\tan\alpha$.

So oblique projection equations are

$$x_p = x + zL_1\cos\theta$$
$$y_p = y + zL_1\sin\theta$$

The transformation matrix for producing any parallel projection onto the $x_v y_v$ plane is written as:

$$M_{parallel} = \begin{bmatrix} 1 & 0 & L_1\cos\theta & 0 \\ 0 & 1 & L_1\sin\theta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

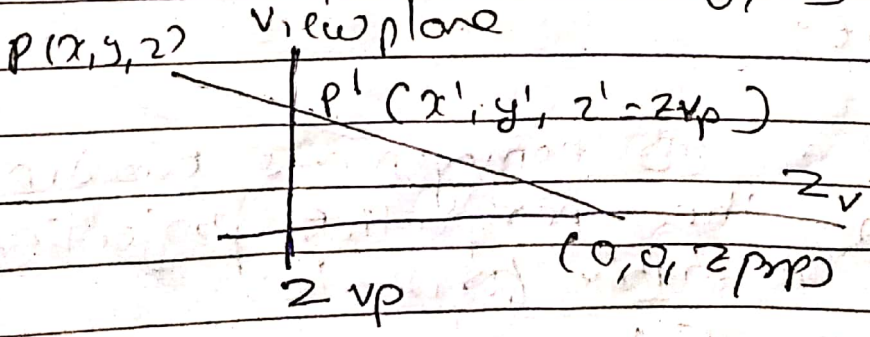
Orthographic projection is obtained when $L_1 = 0$ (projection angle $\alpha = 90°$)

Oblique projection is obtained for non-zero values of L_1 (i.e. $\alpha \neq 90°$)

Generated by projecting points to the display plane along converging path. This causes objects farther from the viewing position to be displayed smaller than the objects of the same size that are nearer to the viewing position.

To obtain a perspective projection of a 3D object, transform points along projection lines that meet at the projection reference point which is placed at z_{prp} along z_v axis and the view plane is situated at z_{vp}. Equations describing coordinate positions along this perspective projection line in parametric form is $x' = x - x \cdot u$, $y' = y - y \cdot u$, $z' = z - (z - z_{prp}) \cdot u$

Parameter 'u' takes values from 0 to 1 the coordinate position $P(x, y, z)$ represents



any point along the projection line. When $u = 0$ we are at position $P(x, y, z)$.

At the other end of the line $U=1$, the projection reference line is at $(0, 0, z_{prp})$, on the view plane $z' = z_{vp}$, solving the z' equation for the parameter U' at this position along the projection line

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

substituting the value of U' in the equations for x', y' we obtain perspective projection equations

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

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

where, $d_p = z_{prp} - z_{vp}$ the distance of viewplane from the projection reference point.

Using a 3D homogeneous coordinate representation, the perspective projection transformation in matrix form is:

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-z_{vp}}{d_p} & z_{vp}\left(\frac{z_{prp}}{d_p}\right) \\ 0 & 0 & -1/d_p & \frac{z_{prp}}{d_p} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Here, the homogeneous factor is $h = \frac{z_{prp} - z}{d_p}$,

the projection coordinates on the view plane are calculated from this homogeneous coordinates as $x_p = \frac{x_h}{h}$, $y_p = \frac{y_h}{h}$

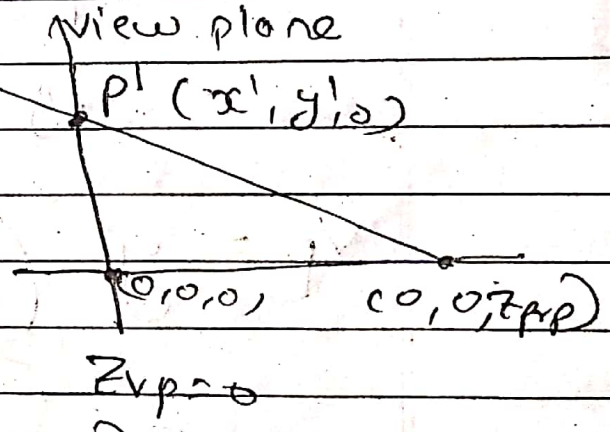
Special cases:

If the view plane is taken to be xy plane or at origin then $z_{vp} = 0$ and Projection coordinates are

$$x_p = x\left(\frac{z_{prp}}{z_{prp} - z}\right) = x\left(\frac{1}{1 - z/z_{prp}}\right)$$

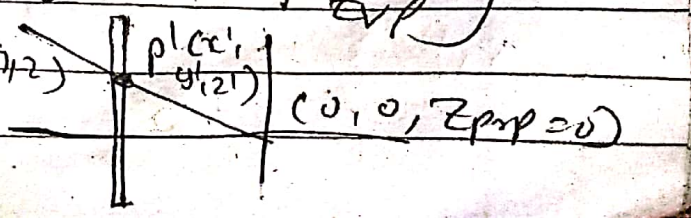
and $$y_p = y\left(\frac{z_{prp}}{z_{prp} - z}\right) = y\left(\frac{1}{1 - \frac{z}{z_{prp}}}\right)$$

If the projection reference point is taken to be at origin then $z_{prp} = 0$ and Projection coordinates on view plane are



$$x_p = x\left(\frac{z_{vp}}{z}\right) = x\left(\frac{1}{z/z_{vp}}\right)$$

and $$y_p = y\left(\frac{z_{vp}}{z}\right) = y\left(\frac{1}{z/z_{vp}}\right)$$



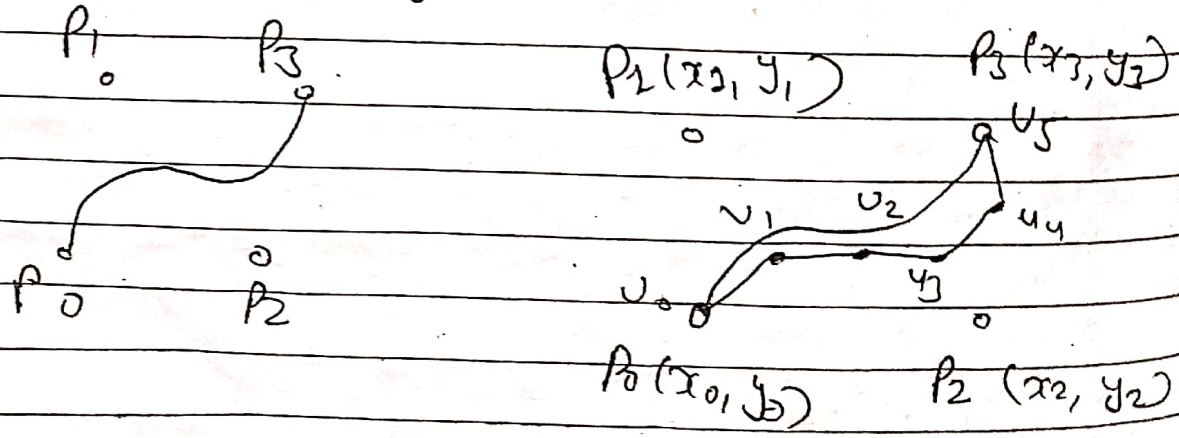
Bezier Curve

Q12 ⇒ A spline approximation technique proposed by Pierre Bezier, used for designing Renault automobile bodies.

The number of control points to be approximated and their relative positions determine the degree of the Bezier polynomial. So degree of curve = no. of control points used - 1

If four control points are used to approximate a curve then we have a cubic curve and if three control points are used then we have a quadratic Bezier curve.

A smooth curve is approximated with a number of line segments.



let the number of line segments used to approximate a curve nSeg = 5

Similarly,

$$y(u) = \sum_{j=0}^{n} y_j \, BEZ_{j,n}(u)$$

or, $y(u) = y_0 \, BEZ_{0,3}(u) + y_1 \, BEZ_{1,3}(u) + y_2 \, BEZ_{2,3}(u) + y_3 \, BEZ_{3,3}(u)$

The blending function used is a Bernstein Polynomial defined as $BEZ_{j,n}(u) =$

$$= \frac{n!}{j!\,(n-j)!} u^j (1-u)^{n-j}$$

or, $BEZ_{j,n}(u) = C(n,j)\, u^j \cdot (1-u)^{n-j}$

where $C(n,j)$ is the Binomial coefficient.

For each 'u' the coordinates x, y are computed and the desired curve is produced when the adjacent coordinates (x, y) are connected with line segments. Now the generalize form of the position vector $P(u)$ can be expressed as:

$$P(u) = P_0 \, BEZ_{0,3}(u) + P_1 \, BEZ_{1,3}(u) + P_2 \, BEZ_{2,3}(u) + P_3 \, BEZ_{3,3}(u)$$

Four Blending functions can be found based on Bernstein Polynomials.

$$BEZ_{0,3}(u) = \frac{3!}{0!\,3!}(u)^0(1-u)^3 = (1-u)^3$$

$$BEZ_{1,3}(u) = \frac{3!}{1!\,2!}u^1(1-u)^2 = 3u(1-u)^2$$

$$BEZ_{2,3}(u) = \frac{3!}{2!\,1!}u^2(1-u) = 3u^2(1-u)$$

$$BEZ_{3,3}(u) = \frac{3!}{3!\,0!}u^3(1-u)^0 = u^3$$

Substituting these values of u in above equations

$$P(u) = (1-u)^3P_0 + 3u(1-u)^2P_1 + 3u^2(1-u)P_2 + u^3P_3$$

when $u = 0$ then $P(u) = P_0$ and when $u = 1$ $P(u) = P_3$

In matrix form,

$$P(u) = \begin{bmatrix}(1-u)^3 & 3u(1-u)^2 & 3u^2(1-u) & u^3\end{bmatrix}$$

$$\text{or, } P(u) = \begin{bmatrix}(1-3u+3u^2-u^3) & (3u-3u^2) & (3u^2-u^3) & u^3\end{bmatrix}\begin{bmatrix}P_0\\P_1\\P_2\\P_3\end{bmatrix}$$

$$P(u) = \begin{bmatrix} u^3 & u^2 & u^1 & u^0 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

Properties

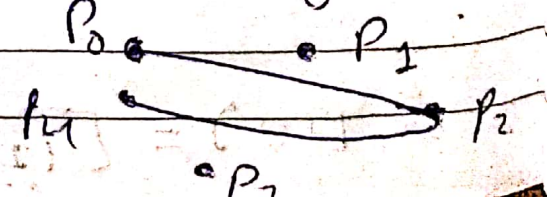
Beizer curve lies within the convex hall of the control points which ensures that the curve smoothly follows the control points.

Four Bezier polynomials are used to construct a curve of 4 control points.

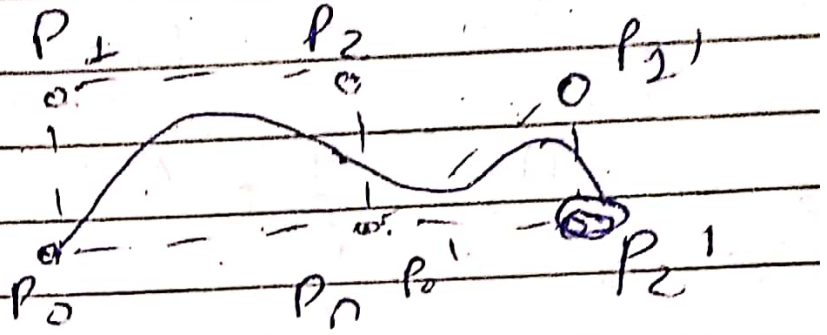
It always passes through the first and the last control points.

A closed curve can be generated by specifying the first and the last control point at the same position.

Specifying multiple control points at the same position gives more weight to that position.



Complicated curves are formed by joining several pieces of curves together (the last control point of the first curve segment and the first control point of the second curve segment must match)



The tangent to the curve at the end point is along the line joining the end point to the adjacent control point.