University of Technology
الجامعة التكنولوجية

Computer Science Department
قسم علوم الحاسوب

Visualization
مرئية افتراضية

Teaching by: Dr. Farah Tawfiq ,
أ.م.د. فرح توفيق عبد الحسين
Prepared by: Lecturer Ali Hassan &
Dr. Dr. Farah Tawfiq
م. علي حسن حمادي و أ.م.د. فرح توفيق عبد الحسين

cs.uotechnology.edu.iq

Visualization
2nd Semester

Part one
(3D Geometry and vectors)

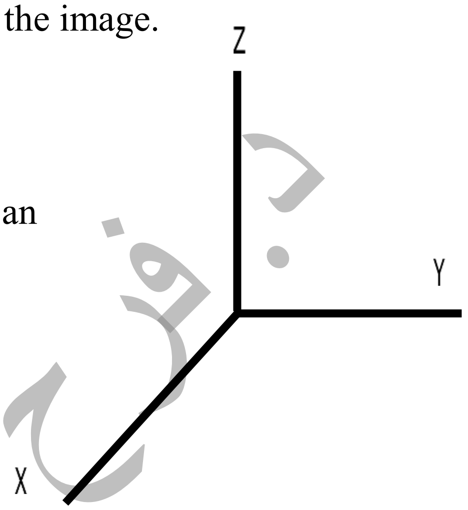
Three-dimensional Transformation

- The world composed of three-dimensional images.
- Objects have height, width, and depth.
- The computer uses a mathematical model to create the image.

1-:Coordinate System:

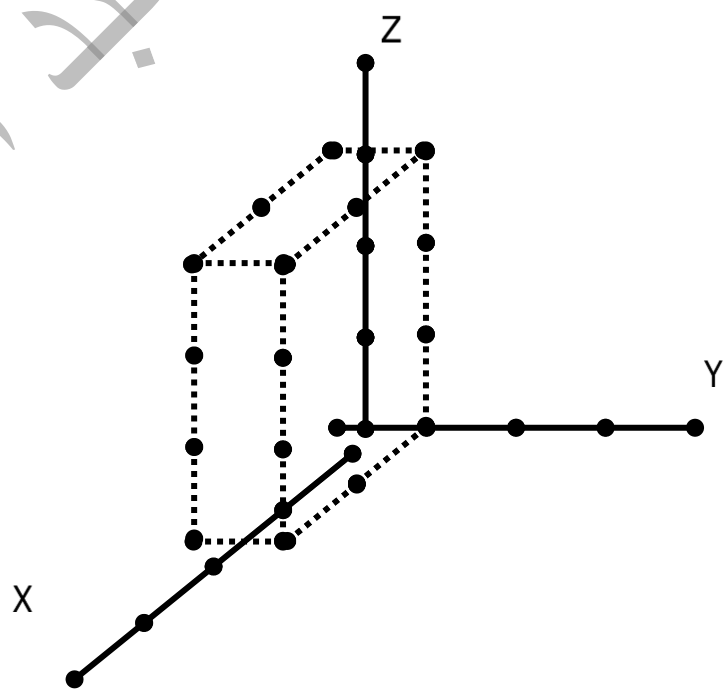
A three dimensional coordinate systemcan be view as an extension of the two dimensional coordinate system.

The third-dimension depth is represented by the Z-axis which is at right angle to the x, y coordinate plane.



A point can be described by triple (x, y, z) of coordinate values

Ex./ Draw the figure: (0,0,3), (0,1,3), (2,0,3), (2,1,3), (0,1,0) (2,0,0), (2,1,0)



2-Vectors in 3D: Vectors can represents as V(X, Y, Z) ≡ V=[x y z] ≡ V=Xi+Yj+Zk

2.1 Modules of vectors: the modules of a vector is given by length of the arrow by using length of line from (0,0,0) to (x, y, z) & term the modules of vector P is |P|.

Where $|P| = \sqrt{Px^2 + Py^2 + Pz^2}$

Ex/ if p(5,-2,3) and Q(2,-4,-4),find |P| and |Q|

Sol/$|P| = \sqrt{5^2 + (-2)^2 + 3^2} = \sqrt{38}$, $|Q| = \sqrt{2^2 + (-4)^2 + (-4)^2} = \sqrt{36}$

2.2 Unit vectors: the unit vectors in direction of vectors P is written as $\hat{P}$, which is calculated as following : $\hat{P} = \frac{P}{|P|}$, in apply of vector P on example p=5i-2j+3k, $|p| = \sqrt{38}$

$\hat{P} = \frac{5i}{\sqrt{38}} - \frac{2j}{\sqrt{38}} + \frac{3k}{\sqrt{38}}$ ➔ $\hat{P}$ = 0.8111i - 0.3244j +0.4867k

2.3 Angles Vector about axis:- using Direction Cosine where = $\frac{\text{Direct in axis}}{|\text{vector}|}$

A. **About X-axis ➔ α =Cos^{-1}(V_i / |V|)**
B. **About Y-axis ➔ β =Cos^{-1}(V_j / |V|)**
C. **About Z-axis ➔ η =Cos^{-1}(V_k / |V|)**

Note: A unit vector is direction cosine for all axes depend of components.

2.4 Add of vectors: let $P=P_i+P_j+P_k$, $Q=Q_i+Q_j+Q_k$ →

$P+Q \equiv Q+P = (P_i + Q_i)i + (P_j + Q_j)j + (P_k + Q_k)k$

2.4 Subtraction of vectors: let $P=P_i+P_j+P_k$, $Q=Q_i+Q_j+Q_k$ →

$P-Q = (P_i - Q_i)i + (P_j - Q_j)j + (P_k - Q_k)k$ ➔ $P-Q \neq Q-P$

2.5 <u>Scalar of vectors</u>: let $P=P_i+P_j+P_k$, $n>1$ then $nP= nP_i+nP_j+nP_k$ but Keep direction

But if $n=-1$ change only direction & $n<0$ then change both components

2.6 <u>multiply of vectors by using Dot product</u>: let $P=P_i+P_j+P_k$, $Q=Q_i+Q_j+Q_k$ →

$P.Q \equiv Q.P = (P_i + Q_i) + (P_j + Q_j) + (P_k + Q_k) = M$

The dot product is useful to find angle between on two vectors by

$P.Q = |P|*|Q|*\cos\Theta \rightarrow \Theta = \cos^{-1}\left(\frac{P.Q}{|P|*|Q|}\right)$

2.7 <u>multiply of vectors by using Cross product</u>: let $P=P_i+P_j+P_k$, $Q=Q_i+Q_j+Q_k$ →

$$P\times Q = \begin{pmatrix} +i & -j & +k \\ Pi & Pj & Pk \\ Qi & Qj & Qk \end{pmatrix} \rightarrow P\times Q \neq Q\times P$$

$[(P_j * Q_k) - (P_k * Q_j)]\, i - [(P_i * Q_k) - (P_k * Q_j)]\, j + [(P_i * Q_j) - (P_j * Q_i)]\, k$

OR $|P\times Q| = |P|*|Q|*\sin\Theta$

OR $P\times Q = |P|*|Q|*\boldsymbol{\eta}*\sin\Theta$ where $\boldsymbol{\eta}$ is unit normal vector

Therefore **i × j = k then j × i = -k**

J × k = i then k × j = -i

Finally/ i × k = j then k × I = -j

Ex/ if p = [5 -2 3], A= -2i+6j -7k find A×P, angle for two P,A

Sol/ A×P=(4 , -29, -26) why?

P×A (H.W)

Angle ? (H.W)

Ex/ if p = [5 -2 3], A= -2i+6j -7k find angle A-P in main axes.

Visualization
2nd Semester

Part two
(3D Transformation)

2: Transformation:

Transformations of 3 dimensions are simply extension of two dimension transformation.A three-dimensional point (x, y, z) will be associated with homogeneous row vector [x, y, z, 1]. We can represent all three-dimensional linear transformation by multiplication of 4*4 matrixes.

2.1 Translate (shift, Move)

The new coordinate of a translate point can be calculate by using transformation.

$$T: \begin{aligned} \underline{X} &= X + a \\ \underline{Y} &= Y + b \\ \underline{Z} &= Z + c \end{aligned}$$

$$[\,\underline{X}, \underline{Y}, \underline{Z}\,] = [X\ Y\ Z\ 1] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & c & 1 \end{bmatrix}$$

2.2:Scaling:

- Allows for a contraction or stretching in any of the x, y, or z direction. To scale an object:
    1. Translate the fixed point to the origin.
    2. Scale the object.
    3. Perform the inverse of the original translation.
- The scaling matrix with scale factors Sx, Sy, Sz in x, y, z direction is given by the matrix

And see that matrices are as follows. The window shift is given by

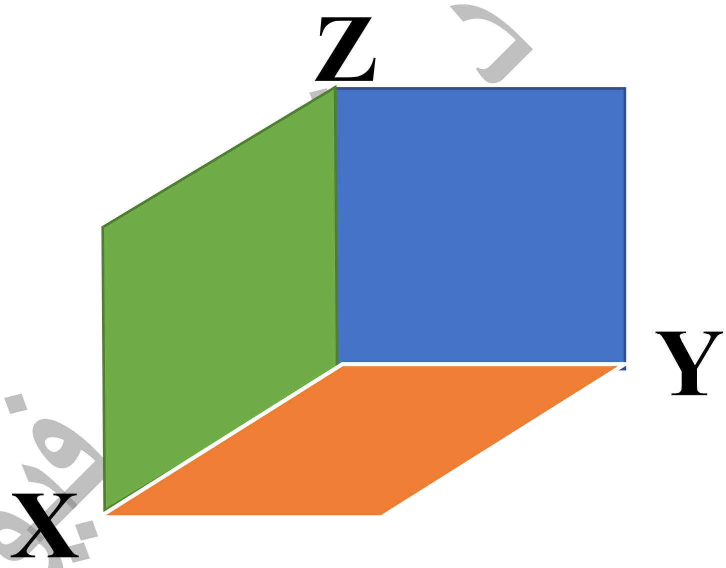
$$S: \begin{aligned} \underline{X} &= Sx * X \\ \underline{Y} &= Sy * Y \\ \underline{Z} &= Sz * Z \end{aligned}$$

$$[\,\underline{X}\,,\underline{Y}\,,\underline{Z}\,] = [X\;Y\;Z\;1] \times \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.3 Mirror 3D

- **About origin: (X, Y, Z)➔ (-X, -Y, -Z)**

$$[\,\underline{X}\,,\underline{Y}\,,\underline{Z}\,] = [X\;Y\;Z\;1] \times \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



- **Mirror about Main Axes**
  - **X-axis: (X, Y, Z)➔ (X, -Y, -Z)**

$$[\,\underline{X}\,,\underline{Y}\,,\underline{Z}\,] = [X\;Y\;Z\;1] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - **Y-axis: (X, Y, Z)➔ (-X, Y, -Z)**

$$[\,\underline{X}\,,\underline{Y}\,,\underline{Z}\,] = [X\;Y\;Z\;1] \times \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - **Z-axis: (X, Y, Z)➔ (-X, -Y, Z)**

$$[\,\underline{X}\,,\underline{Y}\,,\underline{Z}\,] = [X\;Y\;Z\;1] \times \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Mirror about Main Plane**
  - **Plane XY: (X, Y, Z)➔ (X, Y, -Z)**

$$[\,\underline{X}\,,\underline{Y}\,,\underline{Z}\,] = [X\;Y\;Z\;1] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  - **Plane YZ: (X, Y, Z)➔ (-X, Y, Z)**

$$[\,\underline{X}\,,\underline{Y}\,,\underline{Z}\,] = [X\;Y\;Z\;1] \times \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Plane XZ: (X, Y, Z)➔ (X, -Y, Z)**

$$[\,\underline{X}\,,\underline{Y}\,,\underline{Z}\,] = [X\ Y\ Z\ 1] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

<u>2.4: Shear 3D</u> about main plane therefore shear 3D are:-

- **Shear XY ➔**

$x^{sh} = x + Shx*z$

$y^{sh} = y + Shy*z$ ➔ $[\,X^{sh}, Y^{sh}, Z^{sh}\,] = [X\ Y\ Z\ 1] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ shx & shy & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$z^{sh} = z$

- **Shear XZ ➔**

$x^{sh} = x + Shx*y$

$y^{sh} = y$ ➔ $[\,X^{sh}, Y^{sh}, Z^{sh}\,] = [X\ Y\ Z\ 1] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ shx & 1 & shz & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$z^{sh} = z + Shz*y$

- **Shear YZ ➔**

$x^{sh} = x$

$y^{sh} = y + Shy*x$ ➔ $[\,X^{sh}, Y^{sh}, Z^{sh}\,] = [X\ Y\ Z\ 1] \times \begin{bmatrix} 1 & shy & shz & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$z^{sh} = z + Shz*x$

<u>Note</u>:

- if shear for example on plane XY is -3**, therefore shx= -3, shy= -3**
- **if** shear on z by -2 and shear on y by 5, therefore this shear at plane YZ and **shy= 5, shz= -2**

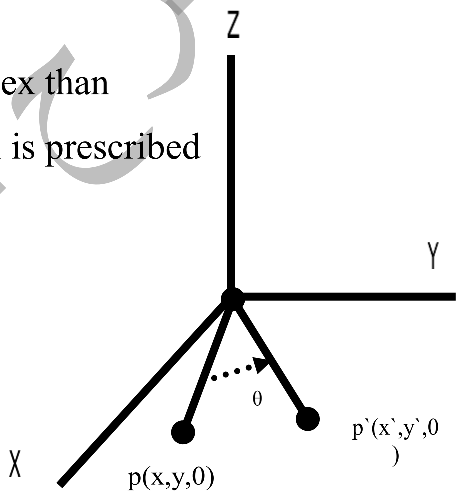
- **if it apply shear directly then center of shearing (0,0,0), but if center shearing not (0,0,0) need**
  a) **Shift center (Xc, Yc, Zc) into (0, 0, 0) by shifting transform.**
  b) **Apply shearing transform (or Scaling transform)**
  c) **Inverse step a (return center in the location (Xc, Yc, Zc))**
  d) **These step (a, b, c) apply in scaling transform.**

2.5 Rotation:

Rotation in three dimensions is considerably more complex than rotation in two dimensions. In two dimensions, a rotation is prescribed by an angle of rotation θ and center of rotation p.

Three dimensional rotations require the prescription of an angle of rotation and an axis of rotation.

The canonical rotations are defined when one of the positive x, y, or z coordinate axes is chosen as the axis of rotation. Then the construction of the rotation transformation proceeds just kike that of a rotation in two dimensions about the origin see figure above.



Rotation about the X-Axis **R(X, θ)**

$$X^r = X$$
$$Y^r = Y\cos(\theta) \quad -Z\sin(\theta)$$
$$Z^r = Z\cos(\theta) \quad +Y\sin(\theta)$$

1	0	0	0
0	Cos(θ)	Sin(θ)	0
0	-Sin(θ)	Cos(θ)	0
0	0	0	1

Rotation about the Y-Axis **R(Y, θ)**

$X^r = X\cos(\theta) - Z\sin(\theta)$

$Y^r = Y$

$Z^r = Z\cos(\theta) + X\sin(\theta)$

Cos(θ)	0	Sin(θ)	0
0	1	0	0
-Sin(θ)	0	Cos(θ)	0
0	0	0	1

Rotation about the Z-Axis **R(Z, θ)**

$X^r = X\cos(\theta) - Y\sin(\theta)$

$Y^r = Y\cos(\theta) + X\sin(\theta)$

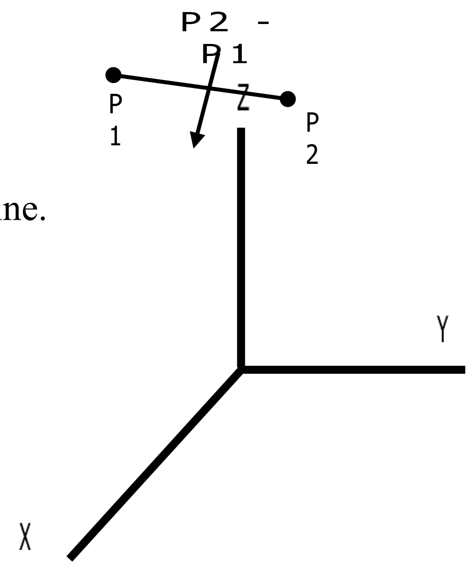
$Z^r = Z$

Cos(θ)	Sin(θ)	0	0
-Sin(θ)	Cos(θ)	0	0
0	0	1	0
0	0	0	1

note that the direction of positive angle of rotation is chosen in accordance to the right-hand rule with respect to the axis of rotation.

The general use of rotation about an axis ***L*** can be built up from these canonical rotations using matrix multiplication in next section.

2.6: <u>Rotation about an arbitrary Axis</u>



- It is like a rotation in the two-dimension about an arbitrary point but it is more complicated.
- Two points P1(x1, y1, z1) and P2(x2, y2, z2) Define a line.

The equation for the line passing through these Point are :

$x = (x2 - x1)\, t + x1$

$y = (y2 - y1)\, t + y1$ t: real value [0 to 1]

$z = (z2 - z1)\, t + z1$

- Let a=(x2 – x1) & b=(y2 – y1) & c=(z2 – z1) then the equation of line becomes

  x=at + x1 & y=bt +y1 & z=ct + z1 the difference P2 – P1 = (x2 – x1) (y2 – y1) (z2 – z1) = (a, b, c) is the direction vector from P1 to P2 along the line through P1 and P2.

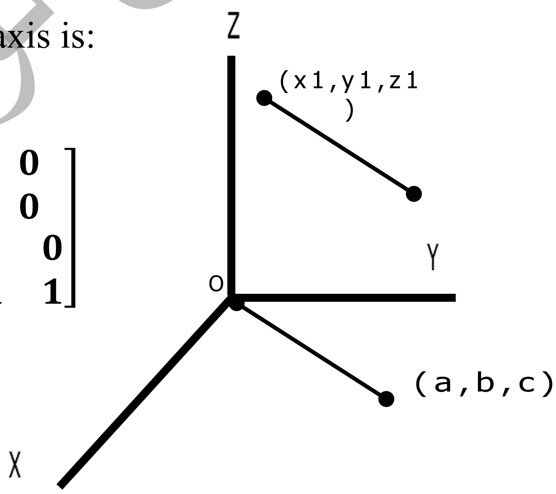
A line can be defined by a point on (x, y, z) and by a direction (a, b, c)

<u>Steps of rotation:</u>

Let (x1, y1, z1) be a point through which the rotation axis passes with (a, b, c) direction. A rotation of angle θ about an arbitrary axis is:

1. ***Translate the point(x1, y1, z1) to origin.***

$$\mathbf{Tr(-x1, -y1, -z1)=} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x1 & -y1 & -z1 & 1 \end{bmatrix}$$



After this translation the direction vector (a, b, c) define the rotation axis as follows.

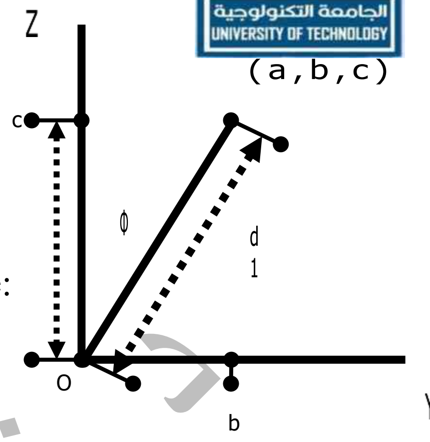
2. Rotate about the x-axis until the rotation axis corresponds to the z-axis.

This can be considering being a rotation about the origin. With the axis coming out of paper

When the rotation axis is projected onto the x,z plane,

any point on it has x coordinate equal to zero. In particular a=0.

The point (0,b,c) is rotated **Φ** degree until the line corresponds to the z-axis. We have find the sin **Φ** and cos **Φ** we find that distance from the origin to (0,b,c) is : $\sqrt{\boldsymbol{b^2 + c^2}}$**= d1**



Sin Φ= b/d1, Cos Φ= c/d1

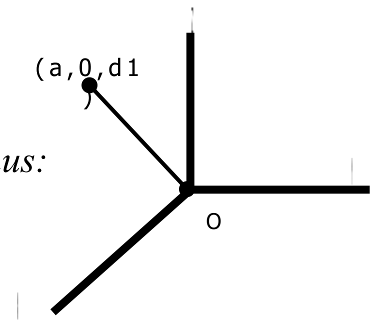
Substituting these values into the x-axis rotation matrix we have:

$$R(X, \boldsymbol{\Phi}) = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{c/d1} & \boldsymbol{b/d1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{-b/d1} & \boldsymbol{c/d1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Now the point(0,b,c)has been transformed to the point (0,0,d1) but since the rotation about the x-axis doesn't change the x coordinate value the point (a, b, c) is now at location (a, 0, d1).

3. ***Rotate about the y-axis until the rotation axis corresponds to the z-axis.***

Since (a, 0, d1) lies in the x, z plane we can visualize this as rotation about the origin with the y-axis coming out of the paper.



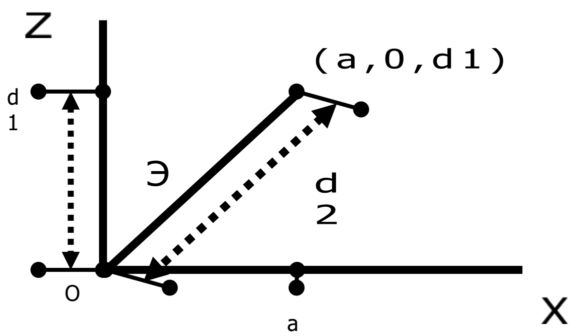
A rotation of angle **Э** in clockwise direction, we need to compute

sin Э, cos Э where: d2 = $\sqrt{\boldsymbol{a^2 + (d1)^2}}$ = $\sqrt{\boldsymbol{a^2 + b^2 + c^2}}$ *thus:*

sin Э = a/d2 ; cos Э = d1/d2

Substituting the value into y rotation matrix given:

$$\mathbf{R(y, Э)} = \begin{bmatrix} \boldsymbol{d1/d2} & \mathbf{0} & \boldsymbol{a/d2} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{-a/d2} & \mathbf{0} & \boldsymbol{d1/d2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$



4. ***Rotate about the z-axis angle* □.** This require the Rz(□) matrix

$$\mathbf{R(Z, \square)} = \begin{bmatrix} cos(\square) & sin(\square) & 0 & 0 \\ -sin(\square) & cos(\square) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5. ***Perform the inverse rotation of step (3)*** **.** requires Ry(-Э)

$$\mathbf{R(y, -Э)} = \begin{bmatrix} d1/d2 & 0 & -a/d2 & 0 \\ 0 & 1 & 0 & 0 \\ +a/d2 & 0 & d1/d2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

6. ***Perform the inverse rotation of step (2).*** Requires Rx(-Φ)

$$R(X, \mathbf{-\Phi}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c/d1 & -b/d1 & 0 \\ 0 & +b/d1 & c/d1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7. ***Perform the inverse translation of step (1).*** Require Tr (x1,y1,z1)

$$\mathbf{Tr(+x1, +y1, +z1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ +x1 & +y1 & +z1 & 1 \end{bmatrix}$$

The composite transformation is:

Tr(-x1,-y1,-z1) * Rx(Φ) * Ry(Э) * Rz(⊖) * Ry(-Э) * Rx(-Φ) * Tr(x1,y1,z1)

Ex/ Rotate figure { W(-1,1,3), U(-3,2,-5), V(5,-2,7), K(-2, -4,-6)….} around line where start (-7,6,-5) and end (4,-3,2) by 56° Clockwise. [In Matrix Form.]

Sol// dx= 11, dy= -9, dz= 7, **d=** $\sqrt{(-9)^2+7^2}=\sqrt{130}$ **,**

➔ **Cos(a)=** $\frac{7}{\sqrt{130}}$, **Sin(a)=** $\frac{-9}{\sqrt{130}}$ {need in step2}

d1= $\sqrt{(11)^2+(-9)^2+7^2}=\sqrt{251}$ ➔ **Cos(b)=** $\frac{\sqrt{130}}{\sqrt{251}}$, **Sin(b)=** $\frac{11}{\sqrt{251}}$ {need in step3}

$$\begin{bmatrix} \frac{\sqrt{130}}{\sqrt{251}} & 0 & \frac{11}{\sqrt{251}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{-11}{\sqrt{251}} & 0 & \frac{\sqrt{130}}{\sqrt{251}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{7}{\sqrt{130}} & \frac{-9}{\sqrt{130}} & 0 \\ 0 & \frac{9}{\sqrt{130}} & \frac{7}{\sqrt{130}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 7 & -6 & 5 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 1 & 3 & 1 \\ -3 & 2 & -5 & 1 \\ 5 & -2 & 7 & 1 \\ -2 & -4 & -6 & 1 \\ . & . & . & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos-56 & \sin-56 & 0 & 0 \\ -\sin-56 & \cos-56 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotate about 56 clockwise in example

$$\begin{bmatrix} \frac{\sqrt{130}}{\sqrt{251}} & 0 & \frac{-11}{\sqrt{251}} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{11}{\sqrt{251}} & 0 & \frac{\sqrt{130}}{\sqrt{251}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{7}{\sqrt{130}} & \frac{9}{\sqrt{130}} & 0 \\ 0 & \frac{-9}{\sqrt{130}} & \frac{7}{\sqrt{130}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -7 & 6 & -5 & 1 \end{bmatrix}$$

Visualization
2nd Semester

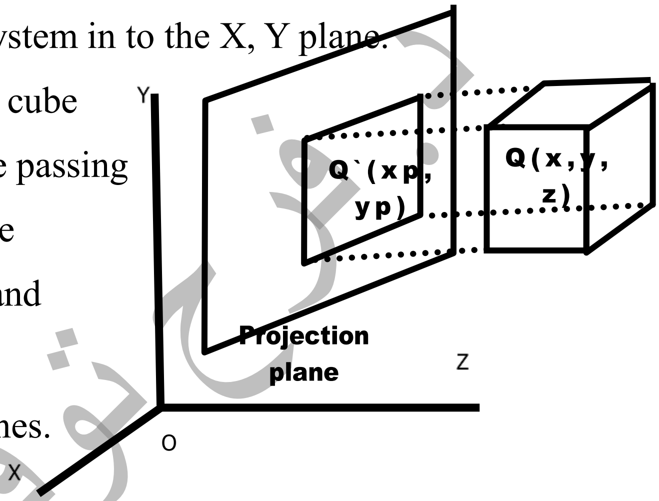
Part three
(3D Projections)

3. Projection

A projection is transformations that perform a conversion from three-dimension representation to a two dimension representation.

3.1 *Parallel (orthogonal) projection:*

A parallel projection is to discard one of the coordinate. Like dropping the Z coordinate and project the X, Y, Z coordinate system in to the X, Y plane.



The projection of a point Q(x, y, z) lying on the cube is point Q`(xp, xy) in the x, y plane where a line passing through Q and parallel to the Z-axis intersect the X, Y plane these parallel line called projectors and we get ***Xp=X ; Yp=Y*.**

- Straight lines are transformed into straight lines.
- Only endpoints of a line in three-dimension are projected and then draw two-dimensional line between these projected points.
- The major disadvantages of parallel projection are its lack of depth information.

Explanation:

- Let [xp yp zp] is a vector of the direction of projection.The image is to be projected onto the x y plane.
- If we have a point on the object at (x1, y1, z1) we wish to determine where the projected point (x2, y2) will lie.The equation for a line passing through the point (x, y, z) and in the direction of projection

X= x1 + xp * u

Y= y1 + yp * u

Z= z1 + zp * u   If Z=0 then u= -z1/zp

Substituting this into the first two equations:

$$X2 = x1 - z1\ (xp / zp)$$

$$[x2\ \ y2\ \ z2\ \ 1] = [x1\ \ y1\ \ z1\ \ 1] * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -xp/zp & -yp/zp & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Y2 = y1 - z1\ (yp / zp)$$

Written in matrix form we set➔

This projection don't care depth object and far near object. it is parallelism of X-axis or y-axis or z-axis and any parallel axis this axis discard in 2D or must be zero in 3D

Parallel Projection	2D – environment	3D – environment
(X,Y,Z)		
Para-X	(y,z)	(0,y,z)
Para-y	(x,z)	(x,0,z)
Para-z	(x,y)	(x,y,0)

Ex// show figure {(7,11,2), (-9, 1,21), (61,19,-2), (17,-31,2), (-72,-18,-22), (4,-11,-92)} that parallel on X-axis and what happen if parallel y-axis ,z-axis in 3D

Sol// Parallel X-axis➔ figure1 {(0,11,2), (0, 1,21), (0,19,-2), (0,-31,2), (0,-18,-22), (0,-11,-92)}

Parallel y-axis➔ figure2 {(7,0,2), (-9, 0,21), (61,0,-2), (17,0,2), (-72,0,-22), (4,0,-92)}

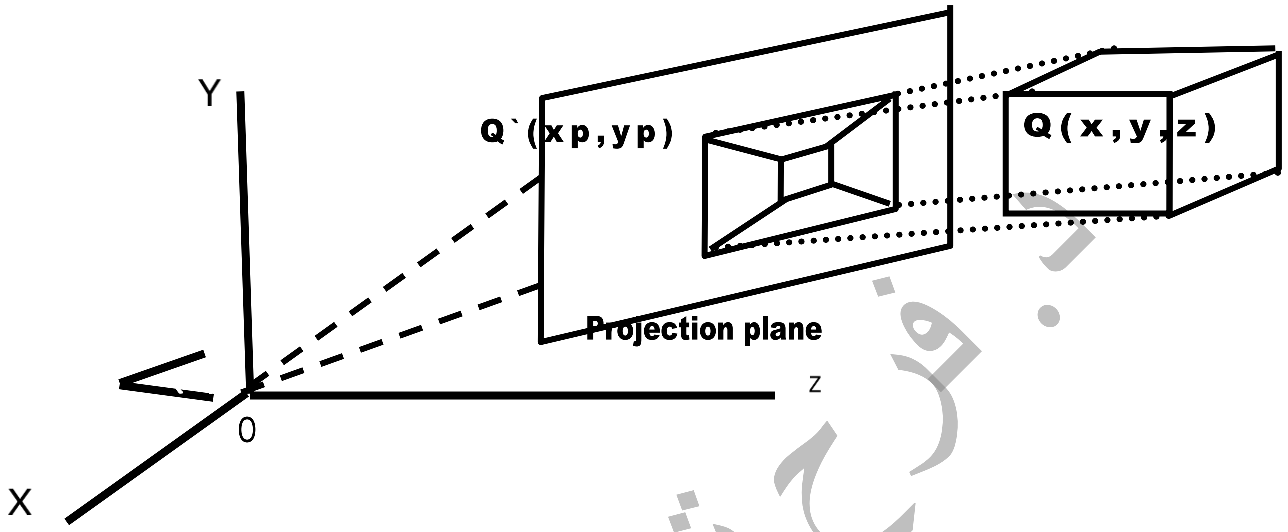
Parallel z-axis➔ figure3 {(7,11,0), (-9, 1,0), (61,19,0), (17,-31,0), (-72,-18,0), (4,-11,0)}

H.W// in 2D Figure1, figure2 and figure3 what happen?

3.2 **Perspective projection**

- The further away an object is from the viewer the smaller it appears.
- These provide the viewer with a depth cue.

- All line are converging at a single point called the center of projection.



If the center of projection is at (xc, yc, zc) and the point on the object is (x1, y1, z1) then the projection ray will be the line containing these point and will give by:

$$X = xc + (x1 - xc)\,u$$
$$Y = yc + (y1 - yc)\,u$$
$$Z = zc + (z1 - zc)\,u$$

The projection point (x2, y2) will be the point where this line intersects the xy plane.

The third equation tells us that u for this intersection point (Z=0) is **u= - zc/(z1-zc)**

substituting into the first two equation gives:

$$x2 = xc - zc\ [\,(x1 - xc)/(z1 - zc)\,]$$
$$y2 = yc - zc\ [\,(y1 - yc)/(z1 - zc)\,]$$

this can be written as:

$$x2 = (\,xc * z1 - x1 * zc\,) / (\,z1 - zc\,)$$
$$y2 = (\,yc * z1 - Y1 * zc\,) / (\,z1 - zc\,)$$

This projection can be put into the form of transformation matrix.

$$P=\begin{bmatrix} -Zc & 0 & 0 & 0 \\ 0 & -Zc & 0 & 0 \\ Xc & Yc & 0 & 1 \\ 0 & 0 & 0 & -Zc \end{bmatrix}$$

It is equivalent from of the projection transformations

$$P=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -Xc/Zc & -Yc/Zc & 0 & -1/Zc \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note: If Q(x, y, z) be a point that project to the point Q`(xp, yp) in center of projection (0, 0, D) where is distance from the eye to the projection plane the perspective transformation

xp = (D * x) / (z + D) ; yp = (D * y) / (z + D) ; zp = 0

The perspective transformation matrix

$$P=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/D \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ex// figure { A(-5,8,0), B(7,-9,11) ,C(1,4,-6)} projection at plane XZ where COP(-3,2,-7)

Sol/(x-xc) is dx because x is final, xc is start same as (y-yc) is dy and (z-zc) is dz ➔ Y must be 0

Points	dx	dy	dz	$Uy=\frac{-yc}{(y-yc)}$
A	-5+3➔ -2	8-2➔ 6	0+7➔ 7	$\frac{-2}{6}$ ➔ $\frac{-1}{3}$
B	7+3➔ 10	-9-2➔ -11	11+7➔ 18	$\frac{-2}{-11}$ ➔ $\frac{2}{11}$
C	1+3➔ 4	4-2➔ 2	-6+7➔ 1	$\frac{-2}{2}$ ➔ -1

Points	x	y	z	Result

A	$-2*\frac{-1}{3}$ -3	$6*\frac{-1}{3}$ +2 ➔ 0	$7*\frac{-1}{3}$ -7	(Ax,0,Az)
B	$10*\frac{2}{11}$ -3	$-11*\frac{2}{11}$ +2➔0	$18*\frac{2}{11}$ -7	(Bx,0,Bz)
C	4* -1 -3	2* -1 +2➔0	1* -1 -7	(Cx,0,Cz)

H.W // projection Plane XY and YZ?

Hint projection Plane XY then Z=0, Plane YZ then X=0

Table one only change Filed (U)

3.3 <u>Oblique projection</u>

Remove oblique-axis (slope-axis) and analysis into polar coordinate

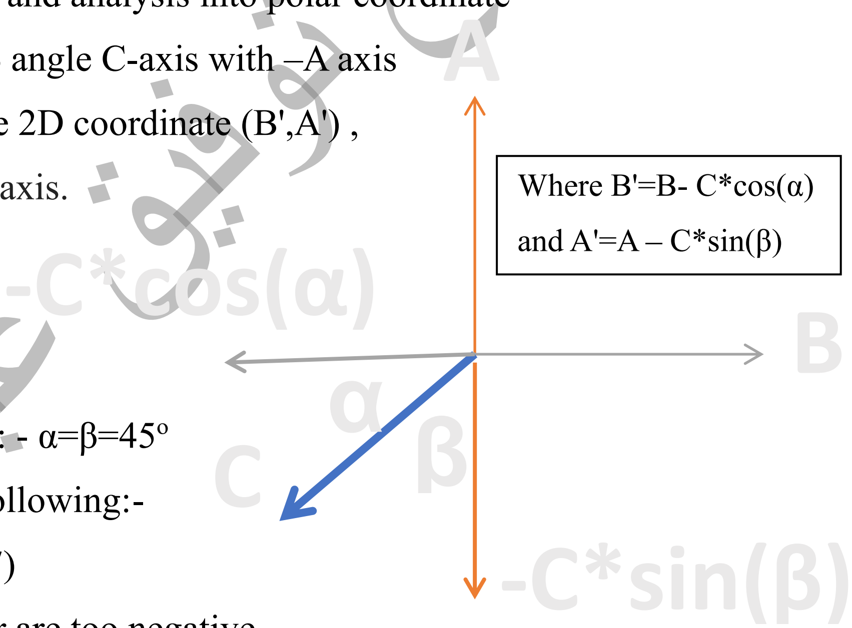
α angle C-axis with –B axis and **β** angle C-axis with –A axis

finally c-axis remove then become 2D coordinate (B',A') ,

B: Horizontal-axis and A vertical-axis.

(Horizontal)➔B'=B- C*cos(α)

(Vertical)➔ A'=A – C*sin(β)



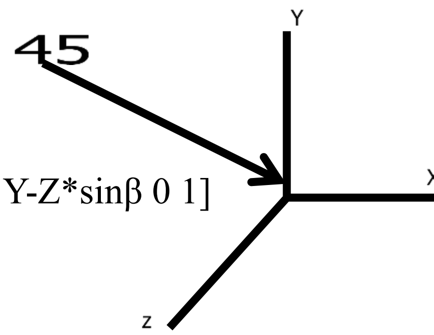
That show 3D reality by equation: - α=β=45°

Z-Axis is oblique coordinate as following:-

X'=X+ (Z*-0.7) & Y'=Y+ (Z*-0.7)

Sin45=cos45 ≈0.7 in three quarter are too negative

Matrix representation



$$[X'\ Y'\ Z'] = [X\ Y\ Z\ 1] * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ Cos(\alpha) & Sin(\beta) & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = [X\text{-}Z*\cos\alpha\ \ Y\text{-}Z*\sin\beta\ \ 0\ \ 1]$$

If you care distance, you add (D: distance in this projection) by

$$[X'\ Y'\ Z']=[X\ Y\ Z\ 1]*\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ D*Cos(\alpha) & D*Sin(\beta) & 0 & 1\\ 0 & 0 & 0 & 1\end{bmatrix}=[X-D*Z*\cos\alpha\ \ Y-D*Z*\sin\beta\ \ 0\ \ 1]$$

x// figure { A(-5,8,0), B(7,-9,11) ,C(1,4,-6)} where X-axis oblique on Vertical by 30°

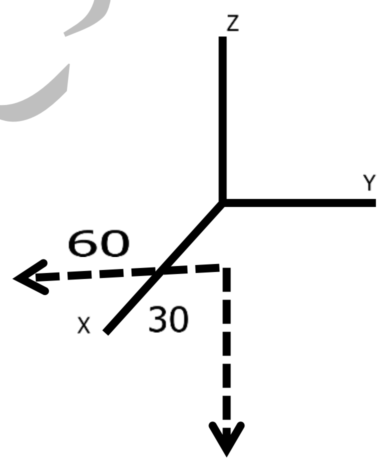
Sol/ X-axis oblique on Vertical by 30°➔ X-axis oblique on horizontal by 90 ° - 30°=60 °

X is Remove then projection on plane YZ

(Horizontal)➔Y'=Y- X*cos(60)

(Vertical)➔ Z'=Z – X*sin(30)

Then apply all figure points (H.W) & draw this figure after



Visualization
2nd Semester

Part four
(3D Shapes)

Line 3D

Line 3D can describe by parametric as following:

x= (x2-x1)*t+x1 where t= [0..1]

y= (y2-y1)*t+y1 in t=0 ➔ x=x1, y=y1, z=z1

z= (z2-z1)*t+z1 in t=1 ➔ x=x2, y=y2, z=z2

To generate line 3D at start(x1, y1, z1) and end(x2, y2, z2)

For t=0 to 1 step 0.01

X= (x2-x1)*t+x1

Y= (y2-y1)*t+y1

Z= (z2-z1)*t+z1

Plot(X ,Y ,Z)

Next t

H.W/ generate line where start (-8, 10, 30) and end (70, -40, -5),find at segment (0.74)

Helix

A cylindrical helix may be described by the following parametric equations:

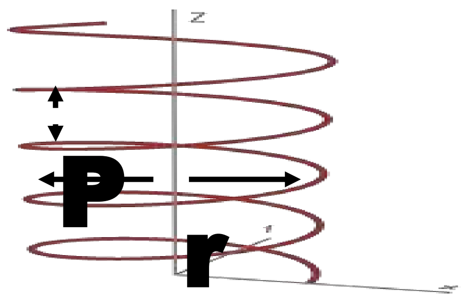
X = Xc + r * Cos(t)

Y =Yc + r * Sin(t)

Z = Zc + p * (t) ' it's round about Z-axis

where t [angle] € (-∞,∞)

(Xc, Yc, Zc) is center of Helix



If cylindrical helix may be round about X-axis therefore:-

X = Xc + p * (t) ' it's round about X-axis

Y =Yc + r * Cos(t)

Z = Zc + r * Sin(t)

same as cylindrical helix may be round about Y-axis therefore:-

X = Xc + r * Cos(t)

Y =Yc + p * (t) ' it's round about Y-axis

Z = Zc + r * Sin(t)

Ex// generate helix where center (-5,11,-8),radius is 56,displace between rings by 33 around x-axis on 76^o into 1112^o.

Find helix point at ϴ = -177 (t= -177)

xc= -5, yc=11, zc= -8, r=56 ,p=33 ,t=[76 .. 1112] →X

Sol// for t=76 to 1112

X = -5 + 33 * (t) ' it's round about X-axis

Y =11+ 56 * Cos(t)

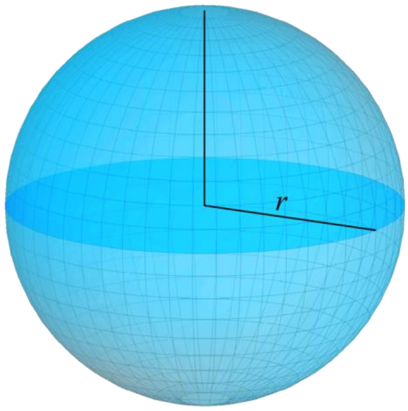
Z = -8 + 56 * Sin(t)

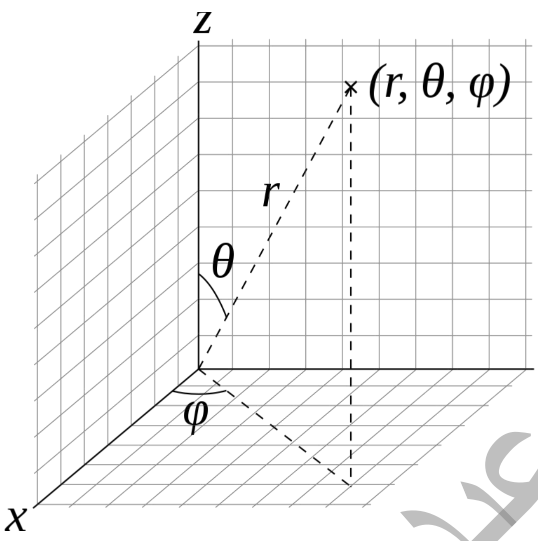
Plot point(X, Y, Z)

Next t

H.W // if you around in Y-axis or Z-axis how to solve it.

Sphere:





Sphere Coordinate has two radius r and p, r is constant but P depend of r where

$X=P*\cos(\omega)$

$Y=P*\sin(\omega)$

$Z=r*\cos(\theta)$

Then $P=r*\sin(\theta)$

- ⇨ ***Substation P on X and Y then***

$X= r*\sin(\theta)*\cos(\omega)$

$Y= r*\sin(\theta)*\sin(\omega)$

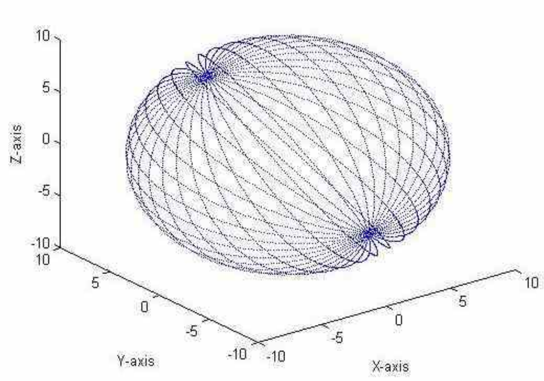
$Z=r*\cos(\theta)$

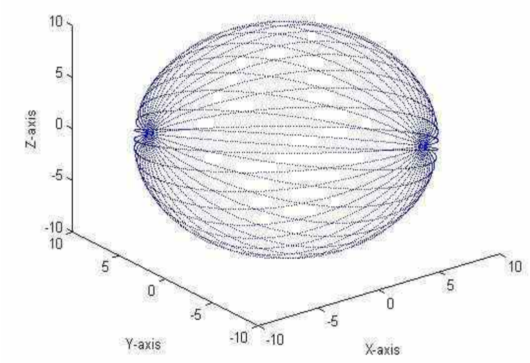
To Draw Sphere by code segment

```
For k = 0 To 360 Step m          ' m is a number circle ball
   For n = 0 To 360 Step v       'v is Texture Ball
     X = r * Sin (n) * Cos (k)
     Y = r * Sin (n) * Sin (k)
     Z = r * Cos (n)

   'Z-rotation
       X2 = X * Cos (az) - Y * Sin (az)        ' az:-angle rotate about Z-axis
      Y2 = X * Sin (az) + Y * Cos (az)

   'X-rotation
     z2 = z * Cos (ax) - Y2 * Sin (ax)        ' ax:- angle rotate about X-axis
     Y1 = z * Sin (ax) + Y2 * Cos (ax)

   'Y-rotation
     X1 = X2 * Cos (ay) - z2 * Sin (ay)       ' ay:- angle rotate about Y-axis
     z1 = X2 * Sin (ay) + z2 * Cos (ay)
     picture1.PSet (X1 + (z1 * -0.7), Y1 + (z1 * -0.7))       ' using oblique
Projection
   Next n
Next k
```

H.W

- Generate ball (sphere) with center (60,-90,-20), size 30 units, rotate about Y-axis by -70 and X-axis by 120 and Z-axis by 30.
- Find location at sphere where (r=11, ϴ=45°, φ= -30)





Sphere ax=120, ay= -70, az=30

Visualization
2nd Semester

Part Five
(3D & 2D curve spline)

Spline Curve

This Part talk's method for curve drawing & curve fitting are {Bezier Curve, B-spline curve, Cubic interpolation curve}

Bezier Curve uses a sequence of control points, P_1, P_2, P_3, P_4 to construct a well defined curve $P(t)$ at each value of t from 0 to 1.This provides a way to generate a curve from a set of points. Changing the points will change the curve. $P(t)$ is defined as:

$$P(t)=(1-t)^3P_1+3(1-t)^2tP_2+3(1-t)t^2P_3+t^3P_4 \ldots (1) \quad \{\text{can apply 2D, 3D}\}$$

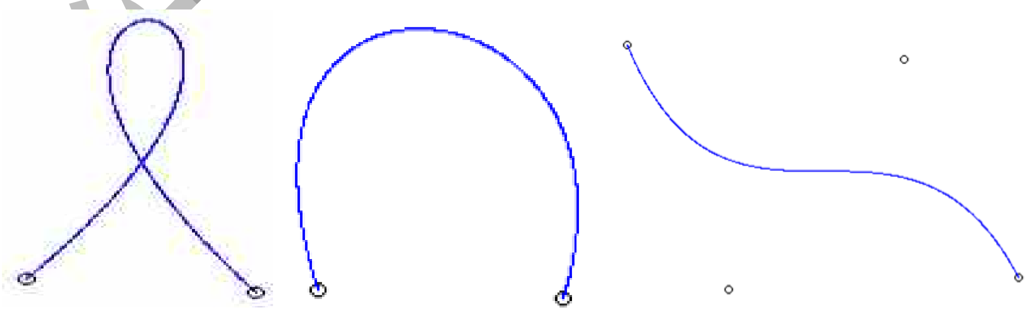
How discover this equ.(1)

T=0 ➔ P(0)=P1 & T=1 ➔P(1)=P4 *therefore* ***equ.(1) Bezier Curve***

Code Segment :- Let X1,X2,x3,X4 & Y1,Y2,Y3,Y4 are control points

```
For t = 0 To 1 Step 0.0001 "to smooth
   x = (1 - t) ^ 3 * X1 + 3 * (1 - t) ^ 2 * t * X2 + 3 * (1 - t) * t ^ 2 * x3 + t ^ 3 * X4
   y = (1 - t) ^ 3 * Y1 + 3 * (1 - t) ^ 2 * t * Y2 + 3 * (1 - t) * t ^ 2 * y3 + t ^ 3 * y4
 plot point (x, y)
Next t
```

Finally: the first and last points are fitting but other are effected not fitting.



Ex// generate Curve where equation is $P(t)= (1-t)^3P_1+ 3(1-t)^2t\,P_2 +3(1-t)t^2P_3+t^3P_4$ *on Control points (9,-50),(67,13),(4,-8),(-22,-97).(H.W) find curve at section=0.67.*

T=0 ➔ P(0)=P_1 and T=1 ➔ P(1)=P_4

Then X1= 9, Y1= -50, X2= 67, Y2= 13, X3= 4, Y3= -8, X4= -22, Y4= -97

```
For t = 0 To 1 Step 0.0001 "to smooth
   x = (1 - t) ^ 3 * X1 + 3 * (1 - t) ^ 2 * t * X2 + 3 * (1 - t) * t ^ 2 * x3 + t ^ 3 * X4
   y = (1 - t) ^ 3 * Y1 + 3 * (1 - t) ^ 2 * t * Y2 + 3 * (1 - t) * t ^ 2 * y3 + t ^ 3 * y4
 Plot point (x, y)
Next t
```

Or (can apply this values in code segments without assign variables)

B-spline Curve:- uses a sequence of control points, P_1, P_2, P_3, P_4 to construct a well-defined curve of degree three, at each value of ***t*** from *0* to *1*. This provides a way to generate a curve from a set of points. Changing the points will change the curve. *F (t)* defined as

$$F(t) = \frac{1}{6}(1-t)^3 p_1 + \frac{1}{6}\{3t^3 - 6t^2 + 4\} p_2 + \frac{1}{6}\{-3t^3 + 3t^2 + 3t + 1\} p_3 + \frac{1}{6} t^3 p_4 ....(2)$$

How discover this equ.(2) is B-spline

T=0 ➔ P(0)= $\frac{1}{6}P_1 + \frac{4}{6}P_2 + \frac{1}{6}P_3$ ***and T=1 ➔ P(1)=*** $\frac{1}{6}P_2 + \frac{4}{6}P_3 + \frac{1}{6}P_4$ *therefore* ***equ.(2)***

B-spline Curve

Code Segment :- Let X1,X2,x3,X4 & Y1,Y2,Y3,Y4 are control points

```
For t = 0 To 1 Step 0.0001
  x = ((1-t)^3*X1 + (3*t^3 -6*t^2 +4)*X2 + (-3*t^3 +3*t^2 +3*t+1)*x3 + t^3*x4) / 6
  y = ((1-t)^3*Y1 + (3*t^3 - 6*t^2 +4)*Y2 + (-3*t^3 +3*t^2 +3*t +1)*y3 +t^3*y4) / 6
  Plot point (x, y)
Next t
```

Finally: the B-spline curve is not fitting any control point but it inside curve points grouping

***Ex//** generate Curve where on Control points are (9,-50,-1), (67, 13, 66), (4,-8, 99), (-22,-97,-21) by equation is:* $\boldsymbol{P(t) = \frac{1}{6}(1-t)^3 P_1 + \frac{1}{6}\{3t^3 - 6t^2 + 4\}P_2 + \frac{1}{6}\{-3t^3 + 3t^2 + 3t + 1\}P_3 + \frac{1}{6}t^3 P_4}$

Sol/ X1= 9, Y1= -50, Z1= -1, X2= 67, Y2= 13, Z2=66, X3= 4, Y3= -8, Z3=99, X4= -22, Y4= -97, Z4= -21

For t = 0 To 1 Step 0.0001

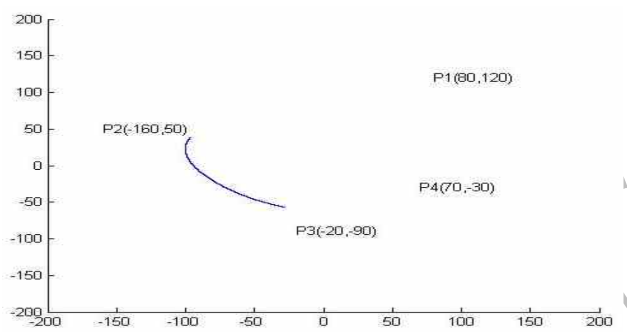
*x =((1-t)^3*X1 + (3*t^3 -6*t^2 +4)*X2 + (-3*t^3 +3*t^2 +3*t+1)*x3 + t^3*x4) / 6*

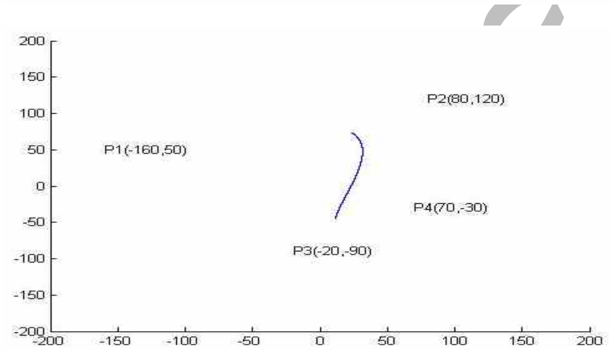
*y =((1-t)^3*Y1 + (3*t^3 - 6*t^2 +4)*Y2 + (-3*t^3 +3*t^2 +3*t +1)*y3 +t^3*y4) / 6*

*z =((1-t)^3*Z1 + (3*t^3 - 6*t^2 +4)*Z2 + (-3*t^3 +3*t^2 +3*t +1)*Z3 +t^3*Z4) / 6*

Plot point (x, z)

Next t . ***(H.W) find curve at section=0.25.***





<u>Cubic Curve interpolation:-</u> *n points curve points that enable fitting all curve points where* $\boldsymbol{F(t)=(t)^3 a_i+(t)^2 b_i+(t) c_i+P_i}$. **<u>where</u>** t=[0..1] and $F(0)= P_i$ but $F(1)= P_{i+1}$

$a_i=(D_{i+1} - D_i)/6$. & $b_i= D_i / 2$. & . $c_i=(x_{i+1} - x_i)-(2D_i + D_{i+1})/6$. Or $c_i=(y_{i+1} - y_i)-(2D_i + D_{i+1})/6$. & $P_i= x_i$ or y_i or z_i

$Dx_i = [\,(x_{i+1} - x_i) - (x_i - x_{i-1})\,] * (3/2)$ where $Dx_{\text{start point}} = 0$ & $Dx_{\text{end point}} = 0$

$Dy_i = [\,(y_{i+1} - y_i) - (y_i - y_{i-1})\,] * (3/2)$ where $Dy_{\text{start point}} = 0$ & $Dy_{\text{end point}} = 0$

$Dz_i = [\,(Z_{i+1} - Z_i) - (Z_i - Z_{i-1})\,] * (3/2)$ where $DZ_{\text{start point}} = 0$ & $DZ_{\text{end point}} = 0$

How can find this للاطلاع

$F(t)=(t)^3 a_i+(t)^2 b_i+(t) c_i+P_i$ …..(1)

$F'(t)=3(t)^2 a_i+2(t) b_i+c_i$ …….(2)

$F''(t)=6(t) a_i+2 b_i$ …. (3) → $F''(0)=D_i$ & $F''(1)=D_{i+1}$

let t=0 in equ.(3) ➔ $D_i=0+2b_i$ ➔ **$b_i=D_i/2$ ..(4)** where $D_i=F''(0)$

let t=1 in equ.(3)➔$D_{i+1}=6a_i+D_i$➔**$a_i=(D_{i+1}-D_i)/6$ …..(5)** where $D_{i+1}=F''(1)$

Apply equ.(4,5) in equ(1) in t=1 then

$$P_{i+1} = \frac{D_{i+1} - D_i}{6} + \frac{D_i}{2} + C_i + P_i \;==>\; (P_{i+1} - P_i) = (\frac{D_{i+1} + 2D_i}{6}) + C_i \;===$$

$$C_i = (P_{i+1} - P_i) - (\frac{D_{i+1} + 2D_i}{6}) \ldots\ldots(6) \;==>\; Ci = (P_{i+1} - P_i) - a_i - b_i$$

Ci=(Pi+1-Pi)-ai-bi

'step 1: WHERE np = number of control points

```
dx(1) = 0: dx(np) = 0: dy(1) = 0: dy(np) = 0
  For i = 2 To np - 1
    dx(i) = ((X(i + 1) - X(i)) - (X(i) - X(i - 1))) * (3 / 2)
    dy(i) = ((Y(i + 1) - Y(i)) - (Y(i) - Y(i - 1))) * (3 / 2)
  Next i
```

'step 2: ' find a,b,c,e for x in all points

```
For j = 1 To np - 1
  ax(j) = (dx(j + 1) - dx(j)) / 6.0                    : bx(j)=dx(j)/2
  cx(j) = ((X(j + 1) - X(j))) + ((-2 * dx(j) - dx(j + 1)) / 6.0) :  ex(j)=X(j)
```

'find a,b,c,e for y for all points

```
  ay(j) = (dy(j + 1) - dy(j)) / 6.0                              : by(j)=dy(j)/2
  cy(j) = ((Y(j + 1) - Y(j))) + ((-2 * dy(j) - dy(j + 1)) / 6.0) : ey(j) = Y(j)
Next j
            'find a,b,c,e for  Z for all points
  az(j) = (dz(j + 1) - dz(j)) / 6.0                              : bZ(j)=dZ(j)/2
  cz(j) = ((Z(j + 1) - Z(j))) + ((-2 * dZ(j) - dZ(j + 1)) / 6.0) : eZ(j) = Z(j)
Next j
'step 3 apply equ.(1)
For P = 1 To np
 For T = 0 To 1 Step 0.0001
   xp = (T ^ 3) * ax(P) + (T ^ 2) * bx(P) + (T) * cx(P) + ex(P)
   yp = (T ^ 3) * ay(P) + (T ^ 2) * by(P) + (T) * cy(P) + ey(P)
   zp = (T ^ 3) * az(P) + (T ^ 2) * bz(P) + (T) * cz(P) + ez(P)
   Plot point (xp, yp, zp) ' draw Curve points or 2D curve
  Next T
Next P
End Sub
```

Let see figure

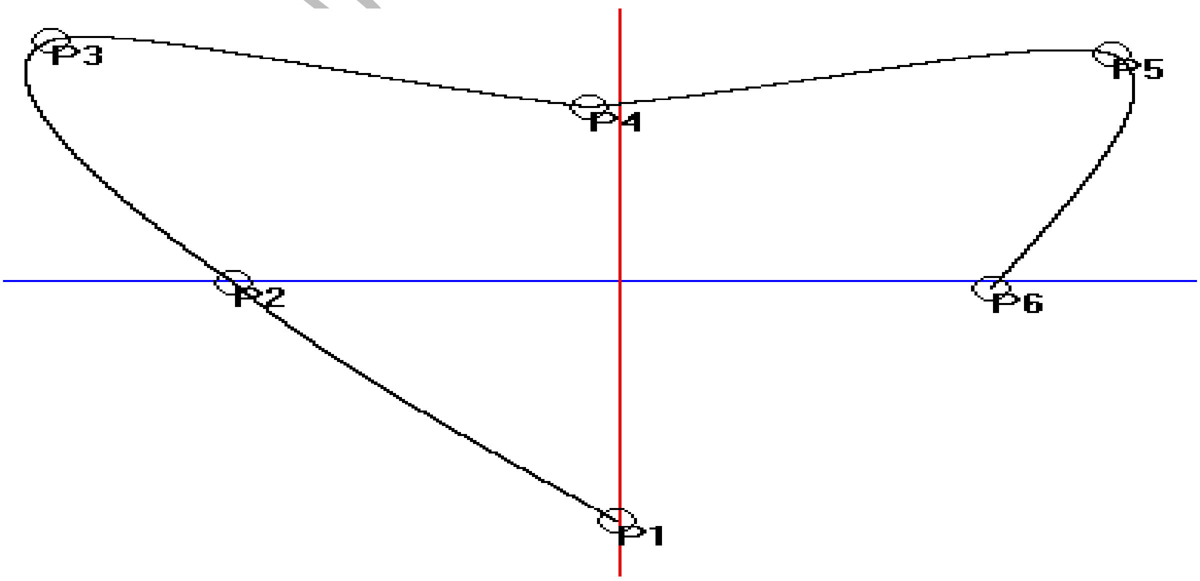


Figure A. design in V.B by L. Ali Hassan Hammadie

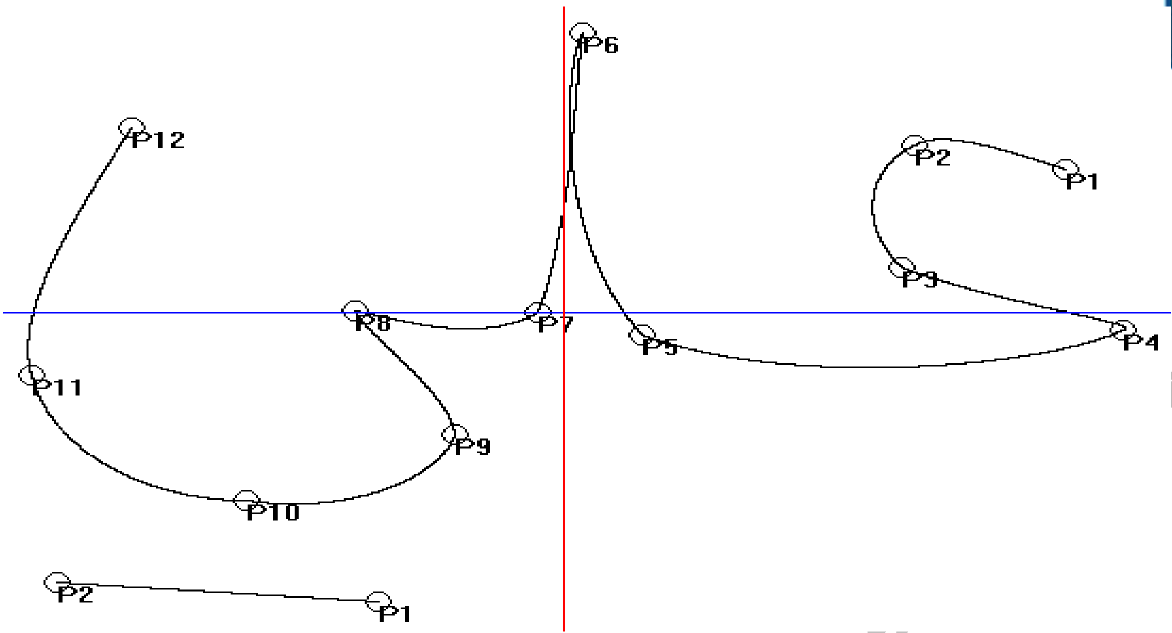


Figure B. design in V.B by L. Ali Hassan Hammadie

Ex// generate Curve where equation is $P(t) = at^3 + bt^2 + ct + P_i$ ***on Control points are (9,-50),(67,13), (4,-8),(-22,-97)***

Sol// T=0 ➔ P(0)= P_i ***and T=1 ➔ P(1)=***P_i***+a+b+c ➔P(1)=*** P_{i+1}

4Point ➔3pieces ➔ pieces (n) = P $_{(n+1)}$ ***- P*** $_{(n)}$

Piece1 {67-9, 13+50)} ➔ Piece1 {58, 63)}

Piece2 {4-67,-8-13)} ➔ Piece2 {-63,-21)}

Piece3 {-22-4,-97+8)} ➔ Piece3 {-26,-89)}

Find Dx_i	*Find* Dy_i	*Find* Dz_i *(if exist)*
$D_1=0$	$D_1=0$	
$D_2=\frac{3}{2}\{-63-58\}=\frac{-363}{2}=-181.5$	$D_2=\frac{3}{2}\{-21-63\}=-126$	
$D_3=\frac{3}{2}\{-26+63\}=\frac{111}{2}=55.5$	$D_3=\frac{3}{2}\{-89+21\}=-102$	
$D_4=0$	$D_4=0$	

Find a_i, b_i, c_i, e_i ***for all pieces***

$$a_i = \frac{D_{i+1} - D_i}{6} \;\;\&\;\; b_i = \frac{D_i}{2} \;\;\&\;\; c_i = (P_{i+1} - P_i) - a_i - b_i$$

Find ax_i	***Find*** bx_i	***Find*** cx_i	***Find*** $ex_i \equiv X_i$
$a_1 = \frac{-181.5 - 0}{6}$	$b_1 = \frac{0}{2}$	$c_1 = 58 - \frac{-181.5}{6} + 0$	***9***
$a_2 = \frac{55.5 + 181.5}{6}$	$b_2 = \frac{-181.5}{2}$	$c_2 = -63 - \left(\frac{237}{6}\right) - \frac{-181.5}{2}$	***67***
$a_3 = \frac{0 - 55.5}{6}$	$b_3 = \frac{55.5}{2}$	$c_3 = -26 - \left(\frac{-55.5}{6}\right) - \frac{55.5}{2}$	***4***

<== للتحقيق الحل $X_{i+1} = a_i + b_i + c_i + X_i$

Piece1(start x1 to x2) X2=ax1+bx1+cx1+ex1 ➔-30.25+0+88.25+9 ➔ X2=67

Piece2(start x2 to x3) X3=ax2+bx2+cx2+ex2 ➔39.5-90.75-11.75+67 ➔ X3=4

Piece2(start x3 to x4) X4=ax3+bx3+cx3+ex3 ➔-9.25+27.75-44.5+4 ➔ X4= -22

Find ay_i	***Find*** by_i	***Find*** cy_i	***Find*** $ey_i \equiv Y_i$
$a_1 = \frac{-126 - 0}{6}$	$b_1 = \frac{0}{2}$	$c_1 = 63 - (-21) + 0$	***-50***
$a_2 = \frac{-102 + 126}{6}$	$b_2 = \frac{-126}{2}$	$c_2 = -21 - (4) - (-63)$	***13***
$a_3 = \frac{0 + 102}{6}$	$b_3 = \frac{-102}{2}$	$c_3 = -89 - (17) - (-51)$	***-8***

<== للتحقيق الحل $Y_{i+1} = a_i + b_i + c_i + Y_i$

Piece1(start y1 to y2) Y2=ay1+by1+cy1+ey1 ➔-21+0+84-50 ➔ Y2=13

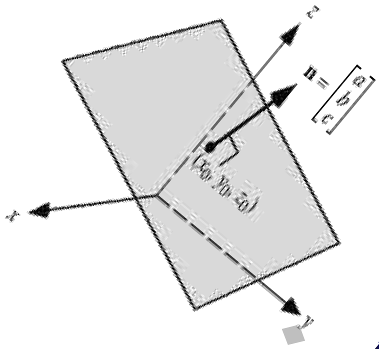
Piece2(start y2 to y3) Y 3=ay2+by2+cy2+ey2 ➔4-63+38+13 ➔ Y3= -8

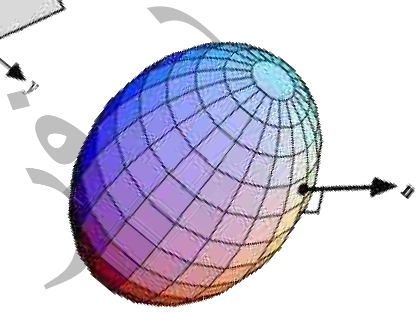
Piece2(start y3 to y4) Y 4=ay3+by3+cy3+ey3 ➔17-51-55-8 ➔ Y4= -97

Visualization
2nd Semester

Part six
(Normal vector & plane equation)

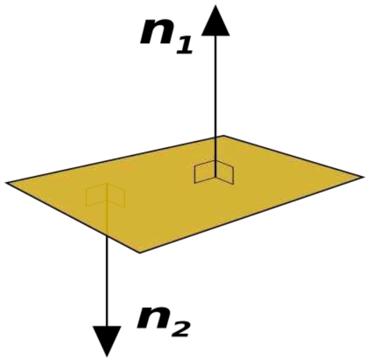
.1Normal Vector





The normal vector, often simply called the "normal," to a surface is a **vector** which is **perpendicular** to the surface at a given point. When normal are considered on closed surfaces, the inward-pointing normal (pointing towards the interior of the surface) and outward-pointing normal are usually distinguished.

How Find Normal Vector at surface or plane?



Let P (3, 1, 4), Q(0, -1, 2), S(5, 3, -2)

➔ P-Q= (3, 2, 2), P-S= (-2, -2, 6)

P-Q×P-S = (16, -22, -2) ➔η1=16i-22j-2k ≡ η1=8i-11j-k

P-S×P-Q = (-16, 22, 2) ➔η2= -16i+22j+2k ≡ η2= -8i+11j+k

Note η1, η2 may be front side surface or back face surface

6.2Plane Equation

In mathematics, a plane is a flat, two-dimensional surface that extends infinitely far. A plane is the two-dimensional analogue of a point (zero dimensions), a line (one dimension) and three-dimensional space. Planes can arise as subspaces of some higher-dimensional space, as with one of a room's walls, infinitely extended, or they may enjoy an independent existence in their own right, as in the setting of Euclidean geometry.

When working exclusively in two-dimensional Euclidean space, the definite article is used, so the plane refers to the whole space. Many fundamental tasks in mathematics, geometry, trigonometry, graph theory, and graphing are performed in a two-dimensional space, or, in other words, in the plane.

A plane in three-dimensional space has the equation (ax + by + cz +d =0) where at least one of the numbers a, b, and c must be non-zero. A plane in 3D coordinate space is determined by a point and a vector that is perpendicular to the plane.

How find plane equation in the following figure?

Let P (3, 1, 4), Q(0, -1, 2), S(5, 3, -2)

Step1: find normal vector P-Q= (3, 2, 2), P-S= (-2, -2, 6), P-Q×P-S = (16, -22, -2)

➔η1=16i-22j-2k

Step2: plane = 16(x-Xi)-22(y-Yi)-22(k-Ki) ➔ apply on P➔16(x-3)-22(y-1)-2(k-4) = 16x-22y-2k-18=0➔ plane= 8x-11y-k-9 (H.W) apply η with Q and S what happen?

6.3 Test arbitrary point on plane

Plane Equation is Ax+ By+ Cz +D =0 if arbitrary point (x_p, y_p z_p) how detect this point is inside or outside or boundary of plane's.

If Axp + By$_p$ + Cz$_p$ +D =0 ➔ point (x_p, y_p z_p) on boundary plane (edge plane)

If Axp + By$_p$ + Cz$_p$ +D <0 ➔ point (x_p, y_p z_p) is inside on plane

If Axp + By$_p$ + Cz$_p$ +D >0 ➔ point (x_p, y_p z_p) is outside on plane

For example plane=8x-11y-k-9 check (1,-2, 0), (1, 2, 0) belong to plane or not why?

Check (1,-2, 0) ➔ 8*1 -11*-2 -1*0 -9 = 21 ➔outside on plane

Check (1, 2, 0) ➔ 8*1 -11*2 -1*0 -9 = -23 ➔inside on plane

6.4 Detect Front –Back side on plane

How detect front side (Visible Surface Detection) and back face (Hidden Surface Elimination)? If find Normal η (X η, Y η, Z η) of plane and have view point V (Xv, Yv, Zv), therefore find {η .V}

If η .V >0 then Surface back face (Hidden Surface Elimination)

Otherwise if η .V <0 then Surface front face (Visible Surface Detection)

Visualization
2nd Semester

Part seven
(The Boundary Fill Recursive Algorithm)

The Boundary Fill Recursive Algorithm

The Boundary Fill Recursive Algorithm is a technique used in computer graphics to fill a region enclosed by a boundary with a specified color. Here's a detailed explanation of the algorithm:

Steps of the Boundary Fill Recursive Algorithm

Starting Point: Choose a starting pixel inside the area to be filled.
Boundary Condition: Define the boundary color that outlines the region.
Fill Color: Specify the color to fill the region with.

Algorithm Process

Initial Check:
If the current pixel is the boundary color or the fill color, return (do nothing).
Color the Pixel:
Change the color of the current pixel to the fill color.
Recursive Calls:
Recursively apply the algorithm to the four or eight neighboring pixels (depending on whether you use 4-connected or 8-connected approaches).

Pseudocode for 4-Connected Boundary Fill

```
boundaryFill(x, y, fillColor, boundaryColor):
   if getPixel(x, y) is not boundaryColor and getPixel(x, y) is not fillColor:
      setPixel(x, y, fillColor)
      boundaryFill(x + 1, y, fillColor, boundaryColor) // right
      boundaryFill(x - 1, y, fillColor, boundaryColor) // left
      boundaryFill(x, y + 1, fillColor, boundaryColor) // top
      boundaryFill(x, y - 1, fillColor, boundaryColor) // bottom
```

-8-Connected Boundary Fill

In the 8-connected approach, the algorithm also checks and fills the diagonal pixels:

```
boundaryFill(x, y, fillColor, boundaryColor):
   if getPixel(x, y) is not boundaryColor and getPixel(x, y) is not fillColor:
      setPixel(x, y, fillColor)
      boundaryFill(x + 1, y, fillColor, boundaryColor) // right
      boundaryFill(x - 1, y, fillColor, boundaryColor) // left
      boundaryFill(x, y + 1, fillColor, boundaryColor) // top
      boundaryFill(x, y - 1, fillColor, boundaryColor) // bottom
```

boundaryFill(x + 1, y + 1, fillColor, boundaryColor) // top-right
boundaryFill(x - 1, y + 1, fillColor, boundaryColor) // top-left
boundaryFill(x + 1, y - 1, fillColor, boundaryColor) // bottom-right
boundaryFill(x - 1, y - 1, fillColor, boundaryColor) // bottom-left

Key Points

4-Connected vs. 8-Connected: The choice between 4-connected and 8-connected depends on the desired fill pattern and connectivity of pixels.
Stack Overflow: Recursive boundary fill algorithms can lead to stack overflow for large areas due to excessive recursive calls. Iterative methods or stack-based implementations can mitigate this issue.
Applications: Used in graphical editors, games, and image processing to fill bounded areas.
The boundary fill algorithm is simple yet powerful for filling enclosed regions, making it a fundamental technique in raster graphics.

Algorithm Steps for Boundary Fill (4-Connected)

1. **Initialize**:
   - Choose a starting pixel (x,y)(x, y)(x,y) inside the region.
   - Define the boundary color and fill color.
2. **Boundary Fill Algorithm**:
   - If the current pixel color is not the boundary color or the fill color:
     - Set the current pixel to the fill color.
     - Recursively apply the algorithm to the four neighboring pixels.
3. procedure boundaryFill(x, y, fillColor, boundaryColor):
4. if getPixel(x, y) ≠ boundaryColor and getPixel(x, y) ≠ fillColor then
5. setPixel(x, y, fillColor)
6. boundaryFill(x + 1, y, fillColor, boundaryColor) // right
7. boundaryFill(x - 1, y, fillColor, boundaryColor) // left
8. boundaryFill(x, y + 1, fillColor, boundaryColor) // top

- boundaryFill(x, y - 1, fillColor, boundaryColor) // bottom

Mathematical Example
Given:

- A 5x5 grid where the boundary color is 1, and the fill color is 2.
- The initial point (2,2)(2, 2)(2,2) is inside the region to be filled.
- The grid is defined as follows:

1 1 1 1 1
1 0 0 0 1
1 0 1 0 1
1 0 0 0 1
1 1 1 1 1

Step-by-Step Execution:

1. Start at (2, 2):
   - Pixel (2, 2) is 0, not boundary color (1), and not fill color (2).
   - Set (2, 2) to 2.
2. Move to (3, 2):
   - Pixel (3, 2) is 0, not boundary color (1), and not fill color (2).
   - Set (3, 2) to 2.
3. Move to (4, 2):
   - Pixel (4, 2) is boundary color (1), so return.
4. Move to (2, 2) from (3, 2):
   - Already filled (2), so return.
5. Move to (3, 1):
   - Pixel (3, 1) is 0, not boundary color (1), and not fill color (2).
   - Set (3, 1) to 2.
6. Continue similarly for other directions and recursively until all valid pixels are filled.

Final Grid:
1 1 1 1 1
1 2 2 2 1
1 2 1 2 1
1 2 2 2 1
1 1 1 1 1

Equations and Recursive Relation

- **Base Case**:
  - if getPixel(x,y)=boundaryColor or getPixel(x,y)=fillColor then return
- **Recursive Case**:
  - setPixel(x,y,fillColor)
  - boundaryFill(x+1,y,fillColor,boundaryColor)
  - boundaryFill(x−1,y,fillColor,boundaryColor)
  - boundaryFill(x,y+1,fillColor,boundaryColor)
    boundaryFill(x,y−1,fillColor,boundaryColor)

This example demonstrates how the algorithm progresses and fills a bounded region, ensuring no overflow or incorrect coloring.

Given:

- A 7x7 grid where the boundary color is 1, and the fill color is 3.
- The initial point (3,3)(3, 3)(3,3) is inside the region to be filled.
- The grid is defined as follows:

1 1 1 1 1 1 1
1 0 0 0 0 0 1
1 0 1 1 1 0 1
1 0 1 0 1 0 1
1 0 1 1 1 0 1
1 0 0 0 0 0 1
1 1 1 1 1 1 1

Algorithm Steps:

1. . **Start at (3, 3)**:
   - Pixel (3, 3) is 0, not boundary color (1), and not fill color (3).
   - Set (3, 3) to 3.
2. **Move to (4, 3)**:
   - Pixel (4, 3) is 0, not boundary color (1), and not fill color (3).
   - Set (4, 3) to 3.
3. **Move to (5, 3)**:
   - Pixel (5, 3) is 0, not boundary color (1), and not fill color (3).
   - Set (5, 3) to 3.

4. **Move to (6, 3)**:
    - Pixel (6, 3) is boundary color (1), so return.
5. **Move to (4, 4)**:
    - Pixel (4, 4) is 1 (boundary), so return.
6. **Move to (4, 2)**:
    - Pixel (4, 2) is 1 (boundary), so return.
7. **Move to (3, 2)**:
    - Pixel (3, 2) is 0, not boundary color (1), and not fill color (3).
    - Set (3, 2) to 3.
8. **Move to (2, 2)**:
    - Pixel (2, 2) is 0, not boundary color (1), and not fill color (3).
    - Set (2, 2) to 3.

Continue similarly for other directions and recursively until all valid pixels are filled.

```
procedure boundaryFill(x, y, fillColor, boundaryColor):

   if getPixel(x, y) ≠ boundaryColor and getPixel(x, y) ≠ fillColor then

      setPixel(x, y, fillColor)

      boundaryFill(x + 1, y, fillColor, boundaryColor)  // right

      boundaryFill(x - 1, y, fillColor, boundaryColor)  // left

      boundaryFill(x, y + 1, fillColor, boundaryColor)  // top

      boundaryFill(x, y - 1, fillColor, boundaryColor)  // bottom
```

1 1 1 1 1 1 1
1 3 3 3 3 3 1
1 3 1 1 1 3 1
1 3 1 3 1 3 1
1 3 1 1 1 3 1
1 3 3 3 3 3 1
1 1 1 1 1 1 1

The grid shows that all enclosed regions starting from (3, 3) have been filled with the fill color (3). The boundary color (1) remains unchanged, ensuring the boundary is respected.