Computer Graphics & Multimedia (5CS4-04)

Computer Science and Engineering Department

Vision of the Department

To become renowned Centre of excellence in computer science and engineering and make competent engineers & professionals with high ethical values prepared for lifelong learning.

Mission of the Department

- To impart outcome based education for emerging technologies in the field of computer science and engineering.
- To provide opportunities for interaction between academia and industry.
- To provide platform for lifelong learning by accepting the change in technologies.
- To develop aptitude of fulfilling social responsibilities.

Program Outcomes (PO):

- **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- **Problem analysis**: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

- **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

- **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

- **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

- **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

- **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Educational Objectives (PEO):

PEO1: To provide students with the fundamentals of Engineering Sciences with more emphasis in computer science and engineering by way of analyzing and exploiting engineering challenges.

PEO2: To train students with good scientific and engineering knowledge so as to comprehend, analyze, design, and create novel products and solutions for the real life problems.

PEO3: To inculcate professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach, entrepreneurial thinking and an ability to relate engineering issues with social issues.

PEO4: To provide students with an academic environment aware of excellence,leadership, written ethical codes and guidelines, and the self-motivated life-long learning needed for a successful professional career.

PEO5: To prepare students to excel in Industry and Higher education by educating Students along with High moral values and Knowledge.

Program Specific Outcome (PSO):

PSO: Ability to interpret and analyze network specific and cyber security issues, automation in real word environment.
PSO2: Ability to Design and Develop Mobile and Web-based applications under realistic constraints.

Course Outcome (CO):

CO1: Implement geometric images using graphical input techniques

CO2: Design and develop images with the help of 2D & 3D transformations.

CO3: Identify visible surfaces for generation of realistic graphics display and curves representation.

CO4: Analyse multimedia and animation techniques.

CO-PO Mapping

Computer Graphics & Multimedia Techniques 5CS4-04

	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1 : Implement geometric images using graphical input techniques	3	2	3	2	2	1	1	1	1	2	1	3
CO2: Design and develop images with the help of 2D & 3D transformations.	3	2	3	2	2	2	1	1	1	2	1	3
CO3: Identify visible surfaces for generation of realistic graphics display and curves representation.	3	3	3	3	2	2	1	2	1	2	1	3
CO4: Analyse multimedia and animation techniques.	3	3	3	3	3	2	2	1	1	2	2	3

CO-PSO Mapping

CO's	**PSO1**	**PSO2**
CO1: Implement geometric images using graphical input techniques	2	2
CO2: Design and develop images with the help of 2D & 3D transformations.	2	2
CO3: Identify visible surfaces for generation of realistic graphics display and curves representation.	2	2
CO4: Analyse multimedia and animation techniques.	2	2

RAJASTHAN TECHNICAL UNIVERSITY, KOTA

Syllabus

III Year-V Semester: B.Tech. Computer Science and Engineering

5CS4-04: Computer Graphics & Multimedia

Credit: 3 | **Max. Marks: 150(IA:30, ETE:120)**

3L+0T+0P | **End Term Exam: 3 Hours**

SN	Contents	Hours
1	**Introduction:** Objective, scope and outcome of the course.	01
2	**Basic of Computer Graphics:**Basic of Computer Graphics, Applications of computer graphics, Display devices, Random and Raster scan systems, Graphics input devices, Graphics software and standards	06
3	**Graphics Primitives:**Points, lines, circles and ellipses as primitives, scan conversion algorithms for primitives, Fill area primitives including scan-line polygon filling, inside-outside test, boundary and flood-fill, character generation, line attributes, area-fill attributes, character attributers. Aliasing, and introduction to Anti Aliasing (No anti aliasing algorithm).	07
4	**Two Dimensional Graphics:**Transformations (translation, rotation, scaling), matrix representation, homogeneous coordinates, composite transformations, reflection and shearing, viewing pipeline and coordinates system, window-to-viewport transformation, clipping including point clipping, line clipping (cohen-sutherland, liang- bersky, NLN), polygon clipping	08
5	**Three Dimensional Graphics:**3D display methods, polygon surfaces, tables, equations, meshes, curved lies and surfaces, quadric surfaces, spline representation, cubic spline interpolation methods, Bazier curves and surfaces, B-spline curves and surfaces.3D scaling, rotation and translation, composite transformation, viewing pipeline and coordinates, parallel and perspective transformation, view volume and general (parallel and perspective) projection transformations.	08
6	**Illumination and Colour Models:**Light sources – basic illumination models – halftone patterns and dithering techniques; Properties of light – Standard primaries and chromaticity diagram; Intuitive colour concepts – RGB colour model – YIQ colour model – CMY colour model – HSV colour model – HLS colour model; Colour selection.	06
7	**Animations &Realism:**Design of Animation sequences – animation function – raster animation – key frame systems – motion specification – morphing – tweening. **ComputerGraphics Realism:** Tiling the plane – Recursively defined curves – Koch curves – C curves – Dragons – space filling curves – fractals – Grammar based models – fractals – turtle graphics – ray tracing.	06
	Total	**42**

Lecture Plan of Computer Graphics & Multimedia Techniques (CGMT) 5CS4-04

Unit No./ Total lec. Req.	**Topics**	**Lect. Req.**
Unit-1(1)	Introduction: Objective, scope and outcome of the course.	1
Unit-2 (6)	Basic of Computer Graphics	1
	Applications of computer graphic, Display devices	2
	Random and Raster scan systems	1
	Graphics input devices, Graphics software and standards	2
Unit-3(7)	Points,lines,circles and ellipses as primitives, scan conversion algorithms for primitives	3
	Fill area primitives including scan- line polygon filling, inside-outside test, boundary and flood-fill	2
	character generation, line attributes, area-fill attributes, character attributers. Aliasing, and introduction to Anti Aliasing (No anti aliasing algorithm)	2
Unit-4(8)-	Transformations (translation, rotation, scaling), matrix representation	1
	homogeneous coordinates, composite transformations, reflection and shearing	1
	viewing pipeline and coordinates system	2
	window-to-viewport transformation, clipping including point clipping	2
	line clipping (cohen-sutherland,liang- bersky, NLN), polygon clipping	2
Unit-5(8)	3D display methods, polygon surfaces, tables, equations, meshes, curved lies and surfaces	2
	quadric surfaces, spline representation, cubic spline interpolation methods, Bazier curves and surfaces	2
	B-spline curves and surfaces.3D scaling, rotation and translation, composite transformation	2
	viewing pipeline and coordinates, parallel and perspective transformation	1
	view volume and general (parallel and perspective) projection transformations.	1
Unit-6(6)	Light sources – basic illumination models – halftone patterns and dithering techniques	1

	Properties of light – Standard primaries and chromaticity diagram	2
	Intuitive colour concepts – RGB colour model – YIQ colour model – CMY colour model – HSV colour model – HLS colour model	2
	Colour selection.	1
Unit-7(6)	Design of Animation sequences – animation function – raster animation – key frame systems – motion specification – morphing – tweening	3
	Tiling the plane – Recursively defined curves – Koch curves – C curves – Dragons – space filling curves –fractals – Grammar based models – fractals – turtle graphics – ray tracing.	3
	Total No. of Lecture	42

This schedule is tentative and is subject to minimal changes during teaching.

Three Dimensional Display Methods

Parallel Projection

- This method generates view from solid object by projecting parallel lines onto the display plane.
- By changing viewing position we can get different views of 3D object onto 2D display screen.

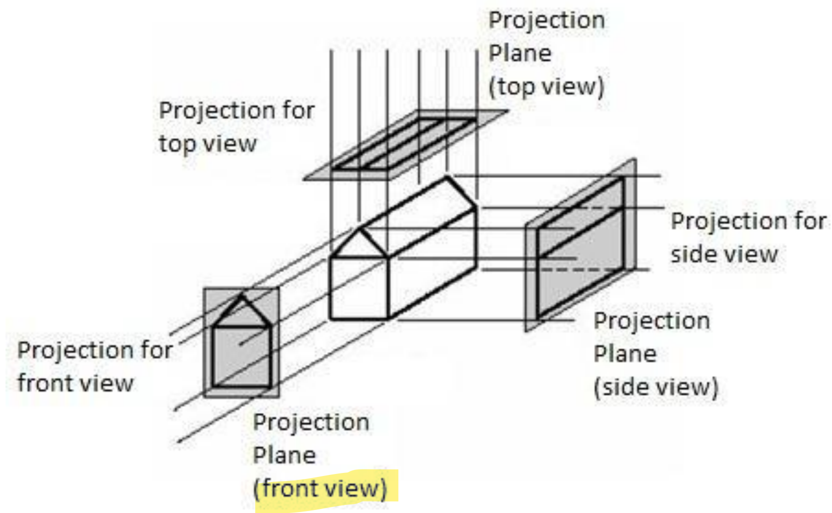


Fig. 4.1: - different views object by changing viewing plane position.

- Above figure shows different views of objects.
- This technique is used in Engineering & Architecture drawing to represent an object with a set of views that maintain relative properties of the object e.g.:- orthographic projection.

Perspective projection

- This method generating view of 3D object by projecting point on the display plane along converging paths.

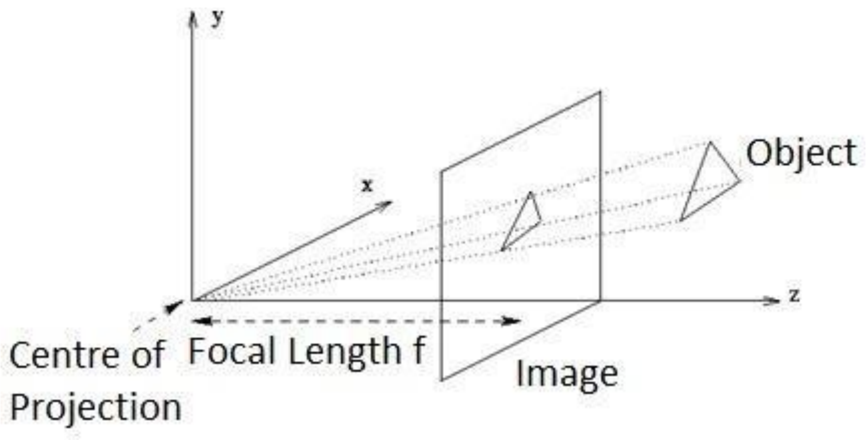


Fig. 4.2: - perspective projection

- This will display object smaller when it is away from the view plane and of nearly same size when closer to view plane.
- It will produce more realistic view as it is the way our eye is forming image.

Depth cueing

- Many times depth information is important so that we can identify for a particular viewing direction which are the front surfaces and which are the back surfaces of display object.

- Simple method to do this is depth cueing in which assign higher intensity to closer object & lower intensity to the far objects.
- Depth cuing is applied by choosing maximum and minimum intensity values and a range of distance over which the intensities are to vary.
- Another application is to modeling effect of atmosphere.

Visible line and surface Identification

- In this method we first identify visible lines or surfaces by some method.
- Then display visible lines with highlighting or with some different color.
- Other way is to display hidden lines with dashed lines or simply not display hidden lines.
- But not drawing hidden lines will loss the some information.
- Similar method we can apply for the surface also by displaying shaded surface or color surface.
- Some visible surface algorithm establishes visibility pixel by pixel across the view plane. Other determines visibility of object surface as a whole.

Surface Rendering

- More realistic image is produce by setting surface intensity according to light reflect from that surface & the characteristics of that surface.
- It will give more intensity to the shiny surface and less to dull surface.
- It also applies high intensity where light is more & less where light falls is less.

Exploded and Cutaway views

- Many times internal structure of the object is need to store. For ex., in machine drawing internal assembly is important.
- For displaying such views it will remove (cutaway) upper cover of body so that internal part's can be visible.

Three dimensional stereoscopic views

- This method display using computer generated scenes.
- It may display object by three dimensional views.
- The graphics monitor which are display three dimensional scenes are devised using a technique that reflects a CRT image from a vibrating flexible mirror.

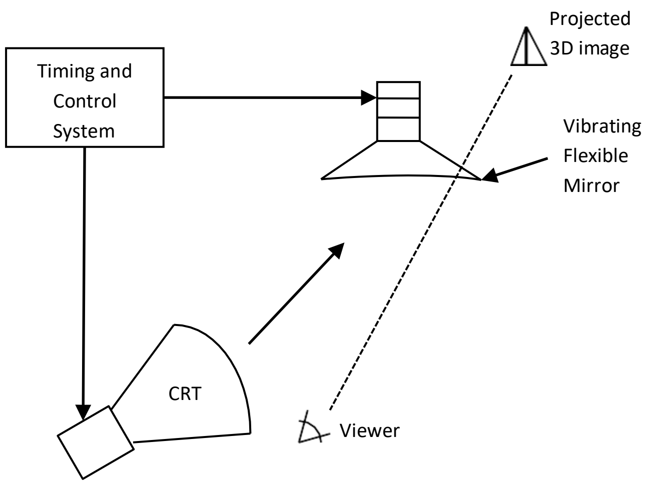


Fig. 4.3: - 3D display system uses a vibrating mirror.

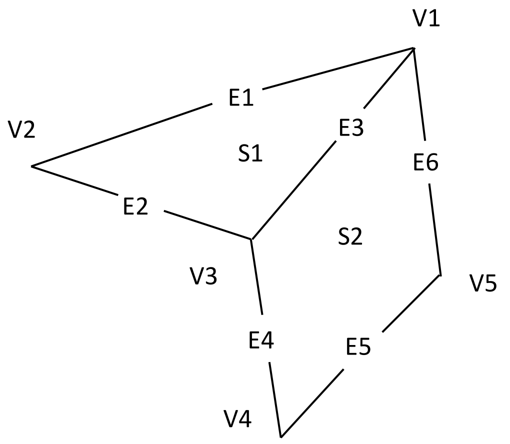
- Vibrating mirror changes its focal length due to vibration which is synchronized with the display of an object on CRT.
- The each point on the object is reflected from the mirror into spatial position corresponding to distance of that point from a viewing position.
- Very good example of this system is GENISCO SPACE GRAPH system, which use vibrating mirror to project 3D objects into a 25 cm by 25 cm by 25 cm volume. This system is also capable to show 2D cross section at different depth.
- Another way is stereoscopic views.
- Stereoscopic views does not produce three dimensional images, but it produce 3D effects by presenting different view to each eye of an observer so that it appears to have depth.
- To obtain this we first need to obtain two views of object generated from viewing direction corresponding to each eye.
- We can contract the two views as computer generated scenes with different viewing positions or we can use stereo camera pair to photograph some object or scene.
- When we see simultaneously both the view as left view with left eye and right view with right eye then two views is merge and produce image which appears to have depth.
- One way to produce stereoscopic effect is to display each of the two views with raster system on alternate refresh cycles.
- The screen is viewed through glasses with each lance design such a way that it act as a rapidly alternating shutter that is synchronized to block out one of the views.

Polygon Surfaces

- A polygonal surface can be thought of as a surface composed of polygonal faces.
- The most commonly used boundary representation for a three dimensional object is a set of polygon surfaces that enclose the object interior

Polygon Tables

- Representation of vertex coordinates, edges and other property of polygon into table form is called polygon table.
- Polygon data tables can be organized into two groups: geometric table and attributes table.
- Geometric table contains vertex coordinate and the other parameter which specify geometry of polygon.
- Attributes table stores other information like Color, transparency etc.
- Convenient way to represent geometric table into three different table namely vertex table, edge table, and polygon table.



Vertex Table	
V1:	X1, Y1, Z1
V2:	X2, Y2, Z2
V3:	X3, Y3, Z3
V4:	X4, Y4, Z4
V5:	X5, Y5, Z5

Edge Table	
E1:	V1, V2
E2:	V2, V3
E3:	V3, V1
E4:	V3, V4
E5:	V4, V5
E6:	V5, V1

Polygon Surface Table	
S1:	E1, E2, E3
S2:	E3, E4, E5, E6

Fig. 4.4: - Geometric Data Table representation.

- Vertex table stores each vertex included into polygon.
- Edge table stores each edge with the two endpoint vertex pointers back to vertex table.
- Polygon table stores each surface of polygon with edge pointer for the each edge of the surface.
- This three table representation stores each vertex one time only and similarly each edge is also one time. So it will avoid representation of storing common vertex and edge so it is memory efficient method.
- Another method to represent with two table vertex table and polygon table but it is inefficient as it will store common edge multiple times.
- Since tables have many entries for large number of polygon we need to check for consistency as it may be possible that errors may occurs during input.
- For dealing with that we add extra information into tables. For example figure below shows edge table of above example with information of the surface in which particular edge is present.

E1:	V1, V2, S1
E2:	V2, V3, S1
E3:	V3, V1, S1, S2
E4:	V3, V4, S2
E5:	V4, V5, S2
E6:	V5, V1, S2

Fig. 4.5: - Edge table of above example with extra information as surface pointer.

- Now if any surface entry in polygon table will find edge in edge table it will verify whether this edge is of particular surface's edge or not if not it will detect errors and may be correct if sufficient information is added.

Plane Equations

- For producing display of 3D objects we must process the input data representation for the object through several procedures.
- For this processing we sometimes need to find orientation and it can be obtained by vertex coordinate values and the equation of polygon plane.
- Equation of plane is given as
  $Ax + By + Cz + D = 0$
- Where (x, y, z) is any point on the plane and A, B, C, D are constants by solving three plane equation for three non collinear points. And solve simultaneous equation for ratio A/D, B/D, and C/D as follows
  $$\frac{A}{D}x_1 + \frac{B}{D}y_1 + \frac{C}{D}z_1 = -1$$
  $$\frac{A}{D}x_2 + \frac{B}{D}y_2 + \frac{C}{D}z_2 = -1$$
  $$\frac{A}{D}x_3 + \frac{B}{D}y_3 + \frac{C}{D}z_3 = -1$$
- Solving by determinant
  $$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \quad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix} \quad C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \quad D = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$
- By expanding a determinant we get
  $A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$
  $B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$
  $C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$
  $D = -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1)$
- This values of A, B, C, D are then store in polygon data structure with other polygon data.
- Orientation of plane is described with normal vector to the plane.

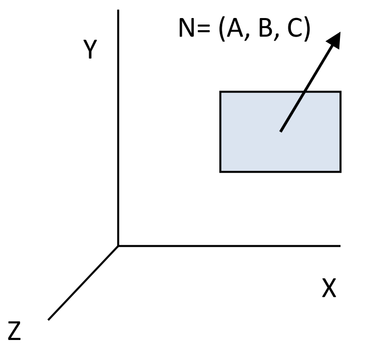


Fig. 4.6: - the vector N normal to the surface.

- Here N= (A,B,C) where A, B, C are the plane coefficient.
- When we are dealing with the polygon surfaces that enclose object interior we define the side of the faces towards object interior is as inside face and outward side as outside face.
- We can calculate normal vector N for any particular surface by cross product of two vectors in counter clockwise direction in right handed system then.
  $N = (v2 - v1)X(v3 - v1)$
- Now N gives values A, B, C for that plane and D can be obtained by putting these values in plane equation for one of the vertices and solving for D.
- Using plane equation in vector form we can obtain D as
  $N \cdot P = -D$
- Plane equation is also used to find position of any point compare to plane surface as follows

If $Ax + By + Cz + D \neq 0$ the point (x,y,z) is not on that plane.
If $Ax + By + Cz + D < 0$ the point (x,y,z) is inside the surface.
If $Ax + By + Cz + D > 0$ the point (x,y,z) is outside the surface.

- These equation are valid for right handed system provides plane parameter A, B, C, and D were calculated using vertices selected in a counter clockwise order when viewing the surface in an outside to inside direction.

Polygon Meshes

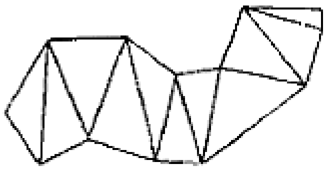


Fig. 4.7: -A triangle strip formed with 11 triangle connecting 13 vertices

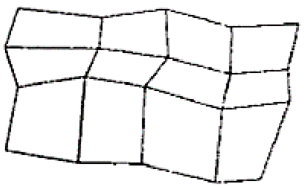


Fig. 4.8: -A quadrilateral mesh containing 12 quadrilaterals constructed from a 5 by 4 input vertex array

- Polygon mesh is a collection of edges, vertices and faces that defines the shape of the polyhedral object in 3D computer graphics and solid modeling.
- An edge can be shared by two or more polygons and vertex is shared by at least two edges.
- Polygon mesh is represented in following ways
  - Explicit representation
  - Pointer to vertex list
  - Pointer to edge list

Explicit Representation

- In explicit representation each polygon stores all the vertices in order in the memory as,

  $P = (((x_1, y_1, z_1), (x_2, y_2, z_2)), \dots , ((x_m, y_m, z_m), (x_n, y_n, z_n)))$
- It process fast but requires more memory for storing.

Pointer to Vertex list

- In this method each vertex stores in vertex list and then polygon contains pointer to the required vertex.
  $V = ((x_1, y_1, z_1), (x_2, y_2, z_2), \dots , (x_n, y_n, z_n))$
- And now polygon of vertices 3, 4, 5 is represented as $P = ((v_3, v_4), (v_4, v_5), (v_5, v_3))$.
- It is considerably space saving but common edges is difficult to find.

Pointer to Edge List

- In this polygon have pointers to the edge list and edge list have pointer to vertex list for each edge two vertex pointer is required which points in vertex list.
  $V = ((x_1, y_1, z_1), (x_2, y_2, z_2), \dots , (x_n, y_n, z_n))$
  $E = ((v_1, v_2), (v_2, v_3), \dots , (v_n, v_m))$
  $P = (E_1, E_2, E_n)$
- This approach is more memory efficient and easy to find common edges.

Spline Representations

- Spline is flexible strip used to produce a smooth curve through a designated set of points.
- Several small weights are attached to spline to hold in particular position.
- Spline curve is a curve drawn with this method.

- The term spline curve now referred to any composite curve formed with polynomial sections satisfying specified continuity condition at the boundary of the pieces.
- A spline surface can be described with two sets of orthogonal spline curves.

Interpolation and approximation splines

- We specify spline curve by giving a set of coordinate positions called control points. This indicates the general shape of the curve.
- **Interpolation Spline**: - When curve section passes through each control point, the curve is said to interpolate the set of control points and that spline is known as Interpolation Spline.

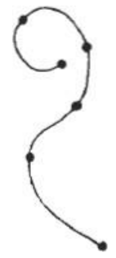


Fig. 4.9: -interpolation spline.

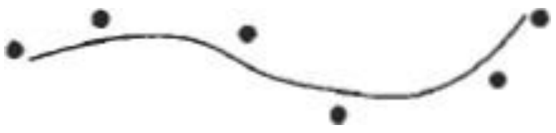


Fig. 4.10: -Approximation spline.

- **Approximation Spline**: - When curve section follows general control point path without necessarily passing through any control point, the resulting curve is said to approximate the set of control points and that curve is known as Approximation Spline.
- Spline curve can be modified by selecting different control point position.
- We can apply transformation on the curve according to need like translation scaling etc.
- The convex polygon boundary that encloses a set of control points is called **convex hull**.

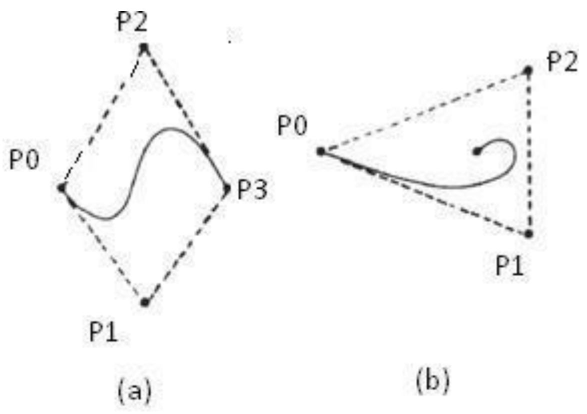


Fig. 4.11: -convex hull shapes for two sets of control points.

- A poly line connecting the sequence of control points for an approximation spline is usually displayed to remind a designer of the control point ordering. This set of connected line segment is often referred as control graph of the curve.
- Control graph is also referred as control polygon or characteristic polygon.

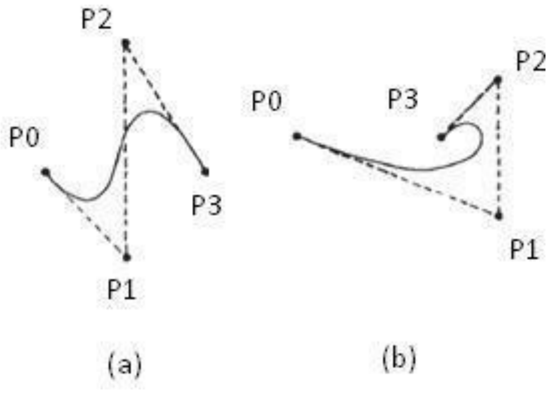


Fig. 4.12: -Control-graph shapes for two different sets of control points.

Parametric continuity condition

- For smooth transition from one curve section on to next curve section we put various continuity conditions at connection points.
- Let parametric coordinate functions as
  $x = x(u),\ y = y(u),\ z = z(u)$ $\qquad \because u_1 \ll u \ll u_2$
- Then **zero order parametric continuity (c^0)** means simply curves meets i.e. last point of first curve section & first points of second curve section are same.
- **First order parametric continuity (c^1)** means first parametric derivatives are same for both curve section at intersection points.
- **Second order parametric continuity (c^2)** means both the first & second parametric derivative of two curve section are same at intersection.
- Higher order parametric continuity is can be obtain similarly.

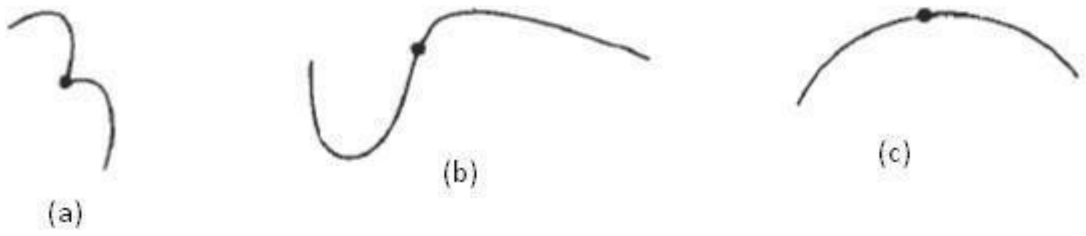


Fig. 4.13: - Piecewise construction of a curve by joining two curve segments uses different orders of continuity: (a) zero-order continuity only, (b) first-order continuity, and (c) second-order continuity.

- First order continuity is often sufficient for general application but some graphics package like cad requires second order continuity for accuracy.

Geometric continuity condition

- Another method for joining two successive curve sections is to specify condition for geometric continuity.
- **Zero order geometric continuity (g^0)** is same as parametric zero order continuity that two curve section meets.
- **First order geometric continuity (g^1)** means that the parametric first derivatives are proportional at the intersection of two successive sections but does not necessary Its magnitude will be equal.
- **Second order geometric continuity (g^2)** means that the both parametric first & second derivatives are proportional at the intersection of two successive sections but does not necessarily magnitude will be equal.

Cubic Spline Interpolation Methods

- Cubic splines are mostly used for representing path of moving object or existing object shape or drawing.
- Sometimes it also used for design the object shapes.
- Cubic spline gives reasonable computation on as compared to higher order spline and more stable compare to lower order polynomial spline. So it is often used for modeling curve shape.
- Cubic interpolation splines obtained by fitting the input points with piecewise cubic polynomial curve that passes through every control point.

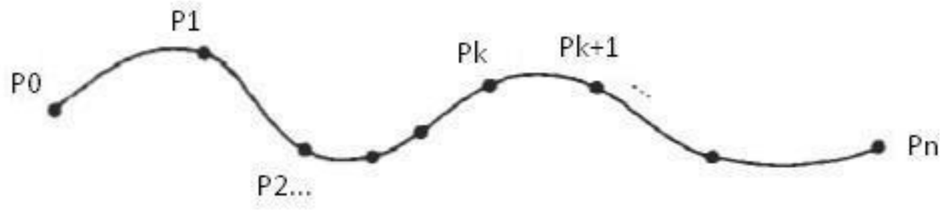


Fig. 4.14: -A piecewise continuous cubic-spline interpolation of n+1 control points.

$p_k = (x_k, y_k, z_k)$ Where, k=0, 1, 2, 3 ..., n

- Parametric cubic polynomial for this curve is given by
  $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$
  $y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$
  $z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$
  $where(0 \leq u \leq 1)$
- For above equation we need to determine for constant a, b, c and d the polynomial representation for each of n curve section.
- This is obtained by settling proper boundary condition at the joints.
- Now we will see common method for settling this condition.

Natural Cubic Splines

- Natural cubic spline is a mathematical representation of the original drafting spline.
- We consider that curve is in c^2 continuity means first and second parametric derivatives of adjacent curve section are same at control point.
- For the ''n+1'' control point we have n curve section and 4n polynomial constants to find.
- For all interior control points we have four boundary conditions. The two curve section on either side of control point must have same first & second order derivative at the control points and each curve passes through that control points.
- We get other two condition as p_0 (first control points) starting & p_n(last control point) is end point of the curve.
- We still required two conditions for obtaining coefficient values.
- One approach is to setup second derivative at p_0 & p_n to be 0. Another approach is to add one extra dummy point at each end. I.e. we add p_{-1} & p_{n+1} then all original control points are interior and we get 4n boundary condition.
- Although it is mathematical model it has major disadvantage is with change in the control point entire curve is changed.
- So it is not allowed for local control and we cannot modify part of the curve.

Hermit Interpolation

- It is named after French mathematician Charles hermit

- It is an interpolating piecewise cubic polynomial with specified tangent at each control points.
- It is adjusted locally because each curve section is depends on it's end points only.
- Parametric cubic point function for any curve section is then given by:
  $p(0) = p_k$
  $p(1) = p_{k+1}$
  $p'(0) = dp_k$
  $p''(1) = dp_{k+1}$
  Where dp_k & dp_{k+1} are values of parametric derivatives at point p_k & p_{k+1} respectively.
- Vector equation of cubic spline is:
  $p(u) = au^3 + bu^2 + cu + d$
- Where x component of p is
- $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$ and similarly y & z components
- Matrix form of above equation is

$$P(u) = [u^3\ u^2\ u\ 1] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

- Now derivatives of p(u) is $p'(u)=3au^2+2bu+c+0$
- Matrix form of p'(u) is

$$P'(u) = [3u^2\ 2u\ 1\ 0] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

- Now substitute end point value of u as 0 & 1 in above equation & combine all four parametric equations in matrix form:

$$\begin{bmatrix} p_k \\ p_{k+1} \\ dp_k \\ dp_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

- Now solving it for polynomial co efficient

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_k \\ p_{k+1} \\ dp_k \\ dp_{k+1} \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = M_H \begin{bmatrix} p_k \\ p_{k+1} \\ dp_k \\ dp_{k+1} \end{bmatrix}$$

- Now Put value of above equation in equation of $p(u)$

$$p(u) = [u^3\ u^2\ u\ 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_k \\ p_{k+1} \\ dp_k \\ dp_{k+1} \end{bmatrix}$$

$$p(u) = [2u^3 - 3u^2 + 1 \quad -2u^3 + 3u^2 \quad u^3 - 2u^2 + u \quad u^3 - u^2] \begin{bmatrix} p_k \\ p_{k+1} \\ dp_k \\ dp_{k+1} \end{bmatrix}$$

$p(u) = p_k(2u^3 - 3u^2 + 1) + p_{k+1}(-2u^3 + 3u^2) + dp_k(u^3 - 2u^2 + u) + dp_{k+1}(u^3 - u^2)$

$p(u) = p_k H_0(u) + p_{k+1} H_1(u) + dp_k H_2(u) + dp_{k+1} H_3(u)$

Where $H_k(u)$ for k=0 , 1 , 2 , 3 are referred to as blending functions because that blend the boundary constraint values for curve section.

- Shape of the four hermit blending function is given below.

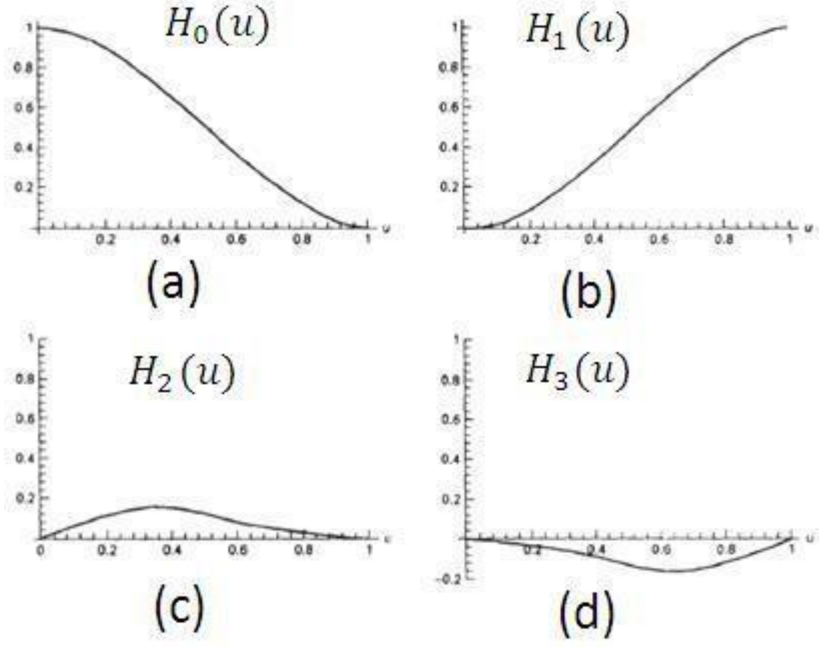


Fig. 4.15: -the hermit blending functions.

- Hermit curves are used in digitizing application where we input the approximate curve slope means DP_k & DP_{k+1}.
- But in application where this input is difficult to approximate at that place we cannot use hermit curve.

Cardinal Splines

- As like hermit spline cardinal splines also interpolating piecewise cubics with specified endpoint tangents at the boundary of each section.
- But in this spline we need not have to input the values of endpoint tangents.
- In cardinal spline values of slope at control point is calculated from two immediate neighbor control points.
- It's spline section is completely specified by the 4-control points.

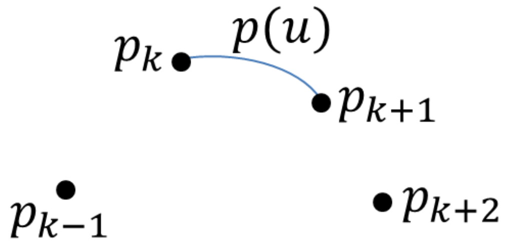


Fig. 4.16: -parametric point function p(u) for a cardinal spline section between control points p_k and p_{k+1}.

- The middle two are two endpoints of curve section and other two are used to calculate slope of endpoints.
- Now parametric equation for cardinal spline is:

$p(0) = p_k$

$p(1) = p_{k+1}$

$p'(0) = \frac{1}{2}(1-t)(p_{k+1} - p_{k-1})$

$p'(1) = \frac{1}{2}(1-t)(p_{k+2} - p_k)$

Where parameter t is called **tension** parameter since it controls how loosely or tightly the cardinal spline fit the control points.

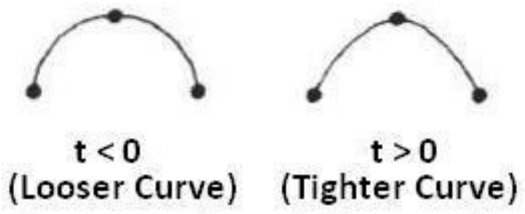


Fig. 4.17: -Effect of the tension parameter on the shape of a cardinal spline section.

- When t = 0 this class of curve is referred to as **catmull-rom spline** or **overhauser splines.**
- Using similar method like hermit we can obtain:

$$p(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot M_c \cdot \begin{bmatrix} p_{k-1} \\ p_k \\ p_{k+1} \\ p_{k+2} \end{bmatrix}$$

- Where the cardinal matrix is

$$M_c = \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

- With $s = (1-t)/2$
- Put value of M_c in equation of p(u)

$$p(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_{k-1} \\ p_k \\ p_{k+1} \\ p_{k+2} \end{bmatrix}$$

$$p(u) = \begin{bmatrix} -su^3 + 2su^2 - su & (2-s)u^3 + (s-3)u^2 + 1 & (s-2)u^3 + (3-s)u^2 + su & su^3 - su^2 \end{bmatrix} \cdot \begin{bmatrix} p_{k-1} \\ p_k \\ p_{k+1} \\ p_{k+2} \end{bmatrix}$$

$$p(u) = p_{k-1}(-su^3 + 2su^2 - su) + p_k((2-s)u^3 + (s-3)u^2 + 1) + p_{k+1}((s-2)u^3 + (3-s)u^2 + su) + p_{k+2}(su^3 - su^2)$$

$$p(u) = p_{k-1}CAR_0(u) + p_kCAR_1(u) + p_{k+1}CAR_2(u) + p_{k+2}CAR_3(u)$$

Where polynomial $CAR_k(u)\ for\ k = 0,1,2,3$ are the cardinals blending functions.

- Figure below shows this blending function shape for t = 0.

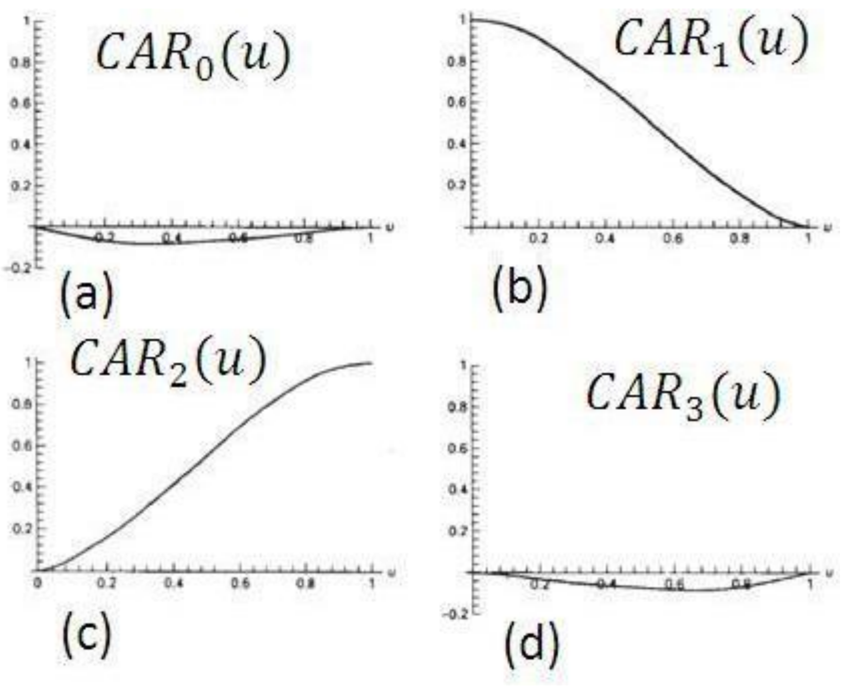


Fig. 4.18: -The cardinal blending function for t=0 and s=0.5.

Kochanek-Bartels spline

- It is extension of cardinal spline
- Two additional parameters are introduced into the constraint equation for defining kochanek-Bartels spline to provide more flexibility in adjusting the shape of curve section.
- For this parametric equations are as follows:

  $p(0) = p_k$

  $p(1) = p_{k+1}$

  $p'(0) = \frac{1}{2}(1-t)[(1+b)(1-c)(p_k - p_{k-1}) + (1-b)(1+c)(p_{k+1} - p_k)]$

  $p'(1) = \frac{1}{2}(1-t)[(1+b)(1+c)(p_{k+1} - p_k) + (1-b)(1-c)(p_{k+2} - p_{k+1})]$

  Where ‘t’ is tension parameter same as used in cardinal spline.
- B is **bias parameter** and C is the **continuity parameter.**
- In this spline parametric derivatives may not be continuous across section boundaries.
- Bias B is used to adjust the amount that the curve bends at each end of section.

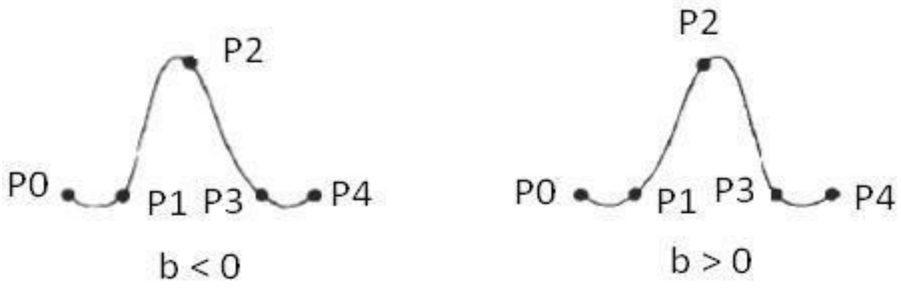


Fig. 4.19: -Effect of bias parameter on the shape of a Kochanek-Bartels spline section.

- Parameter c is used to controls continuity of the tangent vectors across the boundaries of section. If C is nonzero there is discontinuity in the slope of the curve across section boundaries.

- It is used in animation paths in particular abrupt change in motion which is simulated with nonzero values for parameter C.

Bezier Curves and Surfaces

- It is developed by French engineer Pierre Bezier for the Renault automobile bodies.
- It has number of properties and easy to implement so it is widely available in various CAD and graphics package.

Bezier Curves

- Bezier curve section can be fitted to any number of control points.
- Number of control points and their relative position gives degree of the Bezier polynomials.
- With the interpolation spline Bezier curve can be specified with boundary condition or blending function.
- Most convenient method is to specify Bezier curve with blending function.
- Consider we are given n+1 control point position from p_0 to p_n where $p_k = (x_k, y_k, z_k)$.
- This is blended to gives position vector p(u) which gives path of the approximate Bezier curve is:

  $$p(u) = \sum_{k=0}^{n} p_k BEZ_{k,n}(u) \qquad 0 \le u \le 1$$

  Where $BEZ_{k,n}(u) = C(n,k)u^k(1-u)^{n-k}$

  And $C(n,k) = {n!}/{k!\,(n-k)!}$
- We can also solve Bezier blending function by recursion as follow:

  $$BEZ_{k,n}(u) = (1-u)BEZ_{k,n-1}(u) + uBEZ_{k-1,n-1}(u) \qquad n > k \ge 1$$

  Here $BEZ_{k,k}(u) = u^k$ and $BEZ_{0,k}(u) = (1-u)^k$
- Parametric equation from vector equation can be obtain as follows.

  $$x(u) = \sum_{k=0}^{n} x_k BEZ_{k,n}(u)$$

  $$y(u) = \sum_{k=0}^{n} y_k BEZ_{k,n}(u)$$

  $$z(u) = \sum_{k=0}^{n} z_k BEZ_{k,n}(u)$$
- Bezier curve is a polynomial of degree one less than the number of control points. degree = n-1
- Below figure shows some possible curve shapes by selecting various control point.

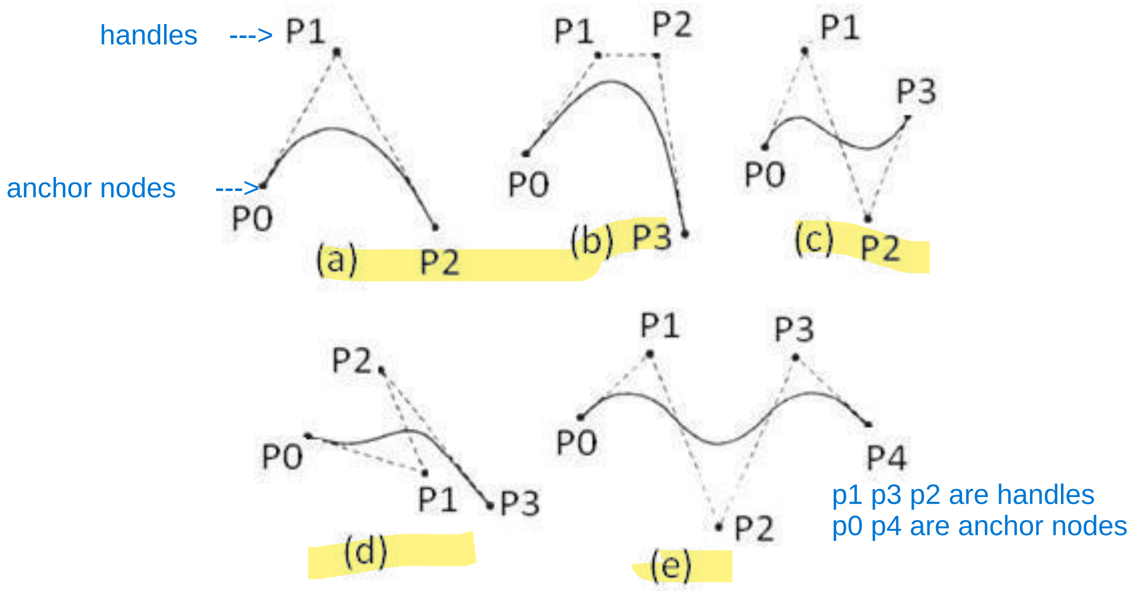


Fig. 4.20: -Example of 2D Bezier curves generated by different number of control points.

- Efficient method for determining coordinate positions along a Bezier curve can be set up using recursive calculation
- For example successive binomial coefficients can be calculated as

$$C(n,k) = \frac{n-k+1}{k} C(n,k-1) \qquad n \geq k$$

Properties of Bezier curves

- It always passes through first control point i.e. $p(0) = p_0$
- It always passes through last control point i.e. $p(1) = p_n$
- Parametric first order derivatives of a Bezier curve at the endpoints can be obtain from control point coordinates as:

$$p'(0) = -np_0 + np_1$$
$$p'(1) = -np_{n-1} + np_n$$

-> the degree of the polynomial defining the curve segment is one less than the no. of defining polygon point.
control point = 4
degree = 4-1 = 3 i.e, cubic polynomial

- Parametric second order derivatives of endpoints are also obtained by control point coordinates as:

$$p''(0) = n(n-1)[(p_2 - p_1) - (p_1 - p_0)]$$
$$p''(1) = n(n-1)[(p_{n-2} - p_{n-1}) - (p_{n-1} - p_n)]$$

- Bezier curve always lies within the convex hull of the control points.
- Bezier blending function is always positive.
- Sum of all Bezier blending function is always 1.

$$\sum_{k=0}^{n} BEZ_{k,n}(u) = 1$$

- So any curve position is simply the weighted sum of the control point positions.
- Bezier curve smoothly follows the control points without erratic oscillations.

Design Technique Using Bezier Curves

- For obtaining closed Bezier curve we specify first and last control point at same position.

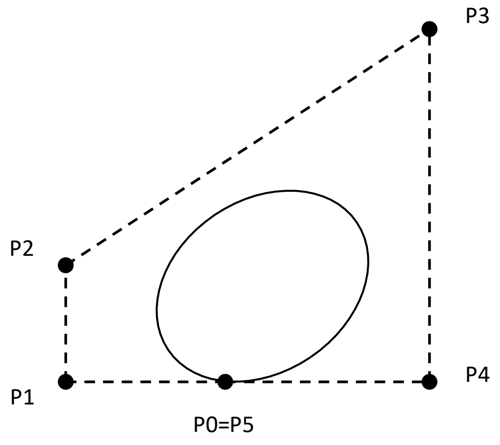


Fig. 4.21: -A closed Bezier Curve generated by specifying the first and last control points at the same location.

- If we specify multiple control point at same position it will get more weight and curve is pull towards that position.

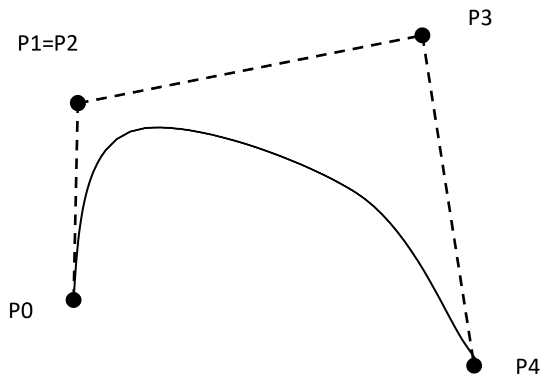


Fig. 4.22: -A Bezier curve can be made to pass closer to a given coordinate position by assigning multiple control point at that position.

- Bezier curve can be fitted for any number of control points but it requires higher order polynomial calculation.
- Complicated Bezier curve can be generated by dividing whole curve into several lower order polynomial curves. So we can get better control over the shape of small region.
- Since Bezier curve passes through first and last control point it is easy to join two curve sections with zero order parametric continuity (C^0).
- For first order continuity we put end point of first curve and start point of second curve at same position and last two points of first curve and first two point of second curve is collinear. And second control point of second curve is at position
  $p_n + (p_n - p_{n-1})$
- So that control points are equally spaced.

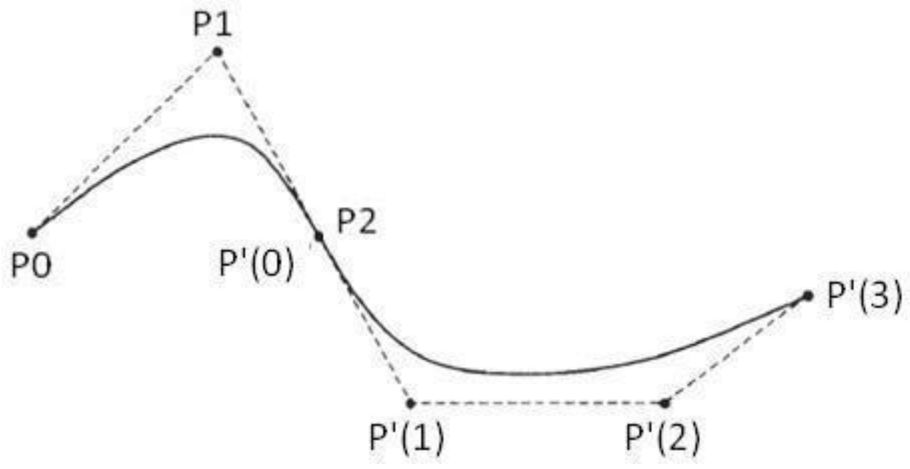


Fig. 4.23: -Zero and first order continuous curve by putting control point at proper place.

- Similarly for second order continuity the third control point of second curve in terms of position of the last three control points of first curve section as
  $p_{n-2} + 4(p_n - p_{n-1})$
- C^2 continuity can be unnecessary restrictive especially for cubic curve we left only one control point for adjust the shape of the curve.

Cubic Bezier Curves

- Many graphics package provides only cubic spline function because this gives reasonable design flexibility in average calculation.
- Cubic Bezier curves are generated using 4 control points.
- 4 blending function obtained by substituting n=3
  $BEZ_{0,3}(u) = (1-u)^3$
  $BEZ_{1,3}(u) = 3u(1-u)^2$
  $BEZ_{2,3}(u) = 3u^2(1-u)$
  $BEZ_{3,3}(u) = u^3$
- Plots of this Bezier blending function are shown in figure below

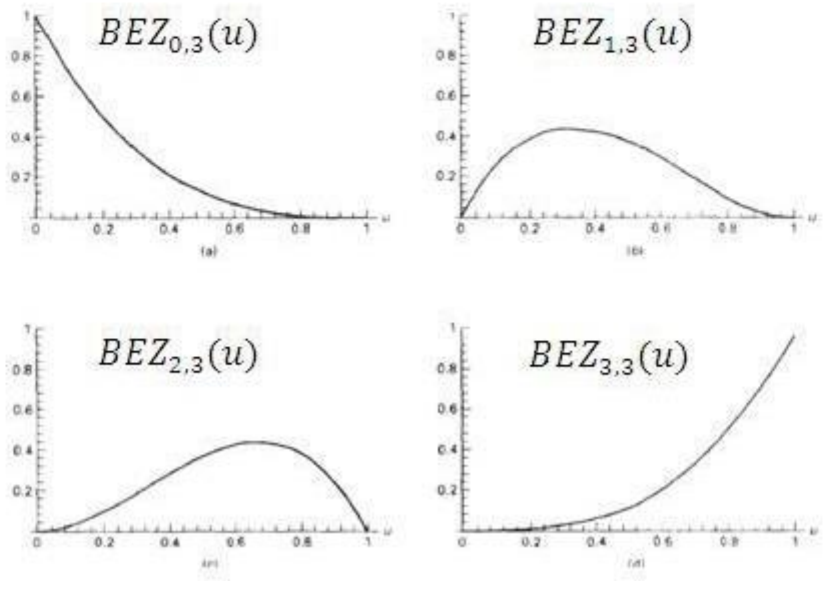


Fig. 4.24: -Four Bezier blending function for cubic curve.

- The form of blending functions determines how control points affect the shape of the curve for values of parameter u over the range from 0 to 1.
  At u = 0 $BEZ_{0,3}(u)$ is only nonzero blending function with values 1.
  At u = 1 $BEZ_{3,3}(u)$ is only nonzero blending function with values 1.
- So the cubic Bezier curve is always pass through p_0 and p_3.
- Other blending function is affecting the shape of the curve in intermediate values of parameter u.
- $BEZ_{1,3}(u)$ is maximum at $u = 1/3$ and $BEZ_{2,3}(u)$ is maximum at $u = 2/3$
- Blending function is always nonzero over the entire range of u so it is not allowed for local control of the curve shape.
- At end point positions parametric first order derivatives are :
  $p'(0) = 3(p_1 - p_0)$
  $p'(1) = 3(p_3 - p_2)$
- And second order parametric derivatives are.
  $p''(0) = 6(p_0 - 2p_1 + p_2)$
  $p''(1) = 6(p_1 - 2p_2 + p_3)$
- This expression can be used to construct piecewise curve with C^1 and C^2 continuity.
- Now we represent polynomial expression for blending function in matrix form:

$$p(u) = [u^3 \quad u^2 \quad u \quad 1] \cdot M_{BEZ} \cdot \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

$$M_{BEZ} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

- We can add additional parameter like tension and bias as we did with the interpolating spline.

Bezier Surfaces

- Two sets of orthogonal Bezier curves can be used to design an object surface by an input mesh of control points.
- By taking Cartesian product of Bezier blending function we obtain parametric vector function as:

$$p(u, v) = \sum_{j=0}^{m} \sum_{k=0}^{n} p_{j,k} BEZ_{j,m}(v) BEZ_{k,n}(u)$$

- $p_{j,k}$ Specifying the location of the (m+1) by (n+1) control points.
- Figure below shows Bezier surfaces plot, control points are connected by dashed line and curve is represented by solid lines.

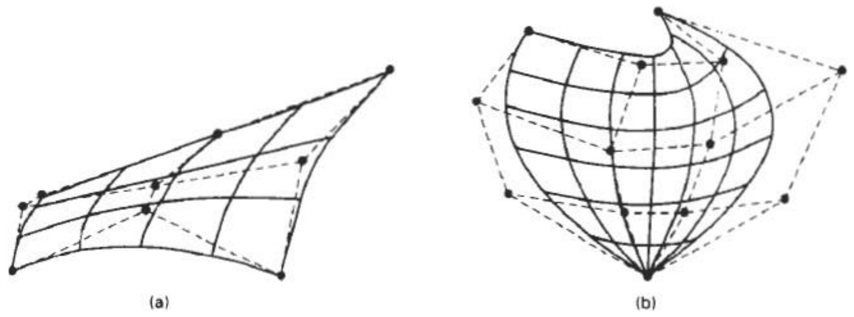


Fig. 4.25: -Bezier surfaces constructed for (a) m=3, n=3, and (b) m=4, n=4. Dashed line connects the control points.

- Each curve of constant u is plotted by varying v over interval 0 to 1. And similarly we can plot for constant v.
- Bezier surfaces have same properties as Bezier curve, so it can be used in interactive design application.
- For each surface patch we first select mesh of control point XY and then select elevation in Z direction.
- We can put two or more surfaces together and form required surfaces using method similar to curve section joining with continuity C^0, C^1, and C^2 as per need.

B-Spline Curves and Surfaces

- B-Spline is most widely used approximation spline.
- It has two advantage over Bezier spline
  1. Degree of a B-Spline polynomial can be set independently of the number of control points (with certain limitation).
  2. B-Spline allows local control.
- Disadvantage of B-Spline curve is more complex then Bezier spline

B-Spline Curves

- General expression for B-Spline curve in terms of blending function is given by:

$$p(u) = \sum_{k=0}^{n} p_k B_{k,d}(u) \qquad u_{min} \le u \le u_{max},\ 2 \le d \le n+1$$

  Where p_k is input set of control points.
- The range of parameter u is now depends on how we choose the B-Spline parameters.
- B-Spline blending function $B_{k,d}$ are polynomials of degree d-1 , where d can be any value in between 2 to n+1.
- We can set d=1 but then curve is only point plot.
- By defining blending function for subintervals of whole range we can achieve local control.
- Blending function of B-Spline is solved by Cox-deBoor recursion formulas as follows.

$$B_{k,1}(u) = \begin{cases} 1 & if\ u_k \le u \le u_{k+1} \\ 0 & otherwise \end{cases}$$

$$B_{k,d}(u) = \frac{u - u_k}{u_{k+d-1} - u_k} B_{k,d-1}(u) + \frac{u_{k+d} - u}{u_{k+d} - u_{k+1}} B_{k+1,d-1}(u)$$

- The selected set of subinterval endpoints u_j is reffered to as a **knot vector**.
- We can set any value as a subinterval end point but it must follow $u_j \le u_{j+1}$
- Values of u_{min} and u_{max} depends on number of control points, degree d, and knot vector.
- Figure below shows local control

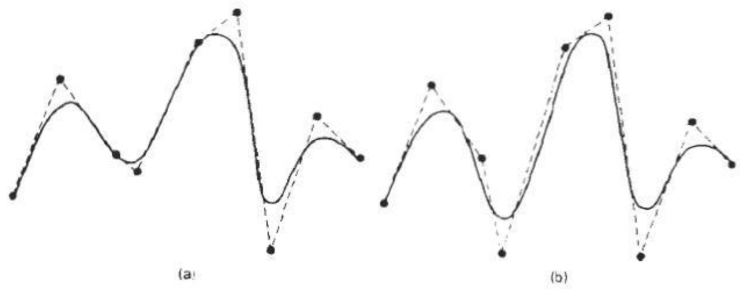


Fig. 4.26: -Local modification of B-Spline curve.

- B-Spline allows adding or removing control points in the curve without changing the degree of curve.
- B-Spline curve lies within the convex hull of at most d+1 control points so that B-Spline is tightly bound to input positions.

- For any u in between u_{d-1} to u_{n+1}, sum of all blending function is 1 i.e. $\sum_{k=0}^{n} B_{k,d}(u) = 1$
- There are three general classification for knot vectors:
  - Uniform
  - Open uniform
  - Non uniform

Properties of B-Spline Curves

- It has degree d-1 and continuity C^{d-2} over range of u.
- For n+1 control point we have n+1 blending function.
- Each blending function $B_{k,d}(u)$ is defined over d subintervals of the total range of u, starting at knot value u_k.
- The range of u is divided into n+d subintervals by the n+d+1 values specified in the knot vector.
- With knot values labeled as $\{u_0, u_1, \dots, u_{n+d}\}$ the resulting B-Spline curve is defined only in interval from knot values u_{d-1} up to knot values u_{n+1}
- Each spline section is influenced by d control points.
- Any one control point can affect at most d curve section.

Uniform Periodic B-Spline

- When spacing between knot values is constant, the resulting curve is called a uniform B-Spline.
- For example $\{0.0, 0.1, 0.2, \dots, 1.0\}$ or $\{0,1,2,3,4,5,6,7\}$
- Uniform B-Spline have periodic blending function. So for given values of n and d all blending function has same shape. And each successive blending function is simply a shifted version of previous function.

  $$B_{k,d}(u) = B_{k+1,d}(u + \Delta u) = B_{k+2,d}(u + 2\Delta u)$$

  Where Δu is interval between adjacent knot vectors.

Cubic Periodic B-Spline

- It commonly used in many graphics packages.
- It is particularly useful for generating closed curve.
- If any three consecutive control points are identical the curve passes through that coordinate position.
- Here for cubic curve d = 4 and n = 3 knot vector spans d+n+1 =4+3+1=8 so it is {0,1,2,3,4,5,6,7}
- Now boundary conditions for cubic B-Spline curve is obtain from equation.

$$p(u) = \sum_{k=0}^{n} p_k B_{k,d}(u) \qquad u_{min} \le u \le u_{max},\ 2 \le d \le n+1$$

- That are

$$p(0) = \frac{1}{6}(p_0 + 4p_1 + p_2)$$
$$p(1) = \frac{1}{6}(p_1 + 4p_2 + p_3)$$
$$p'(0) = \frac{1}{2}(p_2 - p_0)$$
$$p'(1) = \frac{1}{2}(p_3 - p_1)$$

- Matrix formulation for a cubic periodic B-Splines with the four control points can then be written as

$$p(u) = [u^3 \quad u^2 \quad u \quad 1] \cdot M_B \cdot \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

Where

$$M_B = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

- We can also modify the B-Spline equation to include a tension parameter t.
- The periodic cubic B-Spline with tension matrix then has the form:

$$M_{Bt} = \frac{1}{6}\begin{bmatrix} -t & 12-9t & 9t-12 & t \\ 3t & 12t-18 & 18-15t & 0 \\ -3t & 0 & 3t & 0 \\ t & 6-2t & t & 0 \end{bmatrix}$$

When t = 1 $M_{Bt} = M_B$

- We can obtain cubic B-Spline blending function for parametric range from 0 to 1 by converting matrix representation into polynomial form for t = 1 we have

$$B_{0,3}(u) = \frac{1}{6}(1-u)^3$$

$$B_{1,3}(u) = \frac{1}{6}(3u^3 - 6u^2 + 4)$$

$$B_{2,3}(u) = \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1)$$

$$B_{3,3}(u) = \frac{1}{6}u^3$$

Open Uniform B-Splines

- This class is cross between uniform B-Spline and non uniform B-Splines.
- Sometimes it is treated as a special type of uniform B-Spline, and sometimes as non uniform B-Spline
- For open uniform B-Spline (open B-Spline) the knot spacing is uniform except at the ends where knot values are repeated d times.
- For example {0,0,1,2,3,3} for d=2 and n=3, and {0,0,0,0,1,2,2,2,2} for d=4 and n=4.
- For any values of parameter d and n we can generate an open uniform knot vector with integer values using the calculations as follow:

$$u_j = \begin{cases} 0, & for\ 0 \le j < d \\ j - d + 1 & for\ d \le j \le n \\ n - d + 2 & for\ j > n \end{cases}$$

Where $0 \le j \le n + d$

- Open uniform B-Spline is similar to Bezier spline if we take d=n+1 it will reduce to Bezier spline as all knot values are either 0 or 1.
- For example cubic open uniform B-Spline with d=4 have knot vector {0,0,0,0,1,1,1,1}.
- Open uniform B-Spline curve passes through first and last control points.
- Also slope at each end is parallel to line joining two adjacent control points at that end.
- So geometric condition for matching curve sections are same as for Bezier curves.
- For closed curve we specify first and last control point at the same position.

Non Uniform B-Spline

- For this class of spline we can specify any values and interval for knot vector.
- For example {0,1,2,3,3,4}, and {0,0,1,2,2,3,4}
- It will give more flexible shape of curves. Each blending function have different shape when plots and different intervals.
- By increasing knot multiplicity we produce variation in curve shape and also introduce discontinuities.

- Multiple knot value also reduces continuity by 1 for each repeat of particular value.
- We can solve non uniform B-Spline using similar method as we used in uniform B-Spline.
- For set of n+1 control point we set degree d and knot values.
- Then using the recurrence relations we can obtain blending function or evaluate curve position directly for display of the curve.

B-Spline Surfaces

- B-Spline surface formation is also similar to Bezier splines orthogonal set of curves are used and for connecting two surface we use same method which is used in Bezier surfaces.
- Vector equation of B-Spline surface is given by cartesion product of B-Spline blending functions:

$$p(u, v) = \sum_{k1=0}^{n1} \sum_{k2=0}^{n2} p_{k1,k2} B_{k1,d1}(u) B_{k2,d2}(v)$$

- Where $p_{k1,k2}$ specify control point position.
- It has same properties as B-Spline curve.

Translation

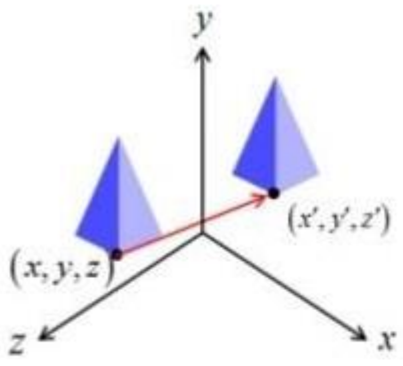


Fig. 5.1: - 3D Translation.

- Similar to 2D translation, which used 3x3 matrices, 3D translation use 4X4 matrices (X, Y, Z, h).
- In 3D translation point (X, Y, Z) is to be translated by amount tx, ty and tz to location (X', Y', Z').

  $\boldsymbol{x' = x + tx}$

  $\boldsymbol{y' = y + ty}$

  $\boldsymbol{z' = z + tz}$
- Let's see matrix equation

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

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

- Example : - Translate the given point P (10,10,10) into 3D space with translation factor T (10,20,5).

  $P' = T \cdot P$

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

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 20 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 10 \\ 10 \\ 10 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 20 \\ 30 \\ 15 \\ 1 \end{bmatrix}$$

Final coordinate after translation is P' (20, 30, 15).

Rotation

- For 3D rotation we need to pick an axis to rotate about.
- The most common choices are the X-axis, the Y-axis, and the Z-axis

Coordinate-Axes Rotations

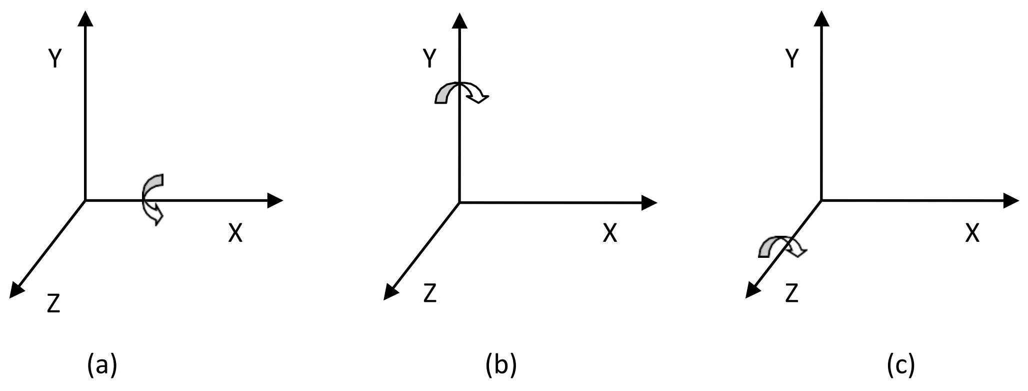


Fig. 5.2: - 3D Rotations.

Z-Axis Rotation

- Two dimension rotation equations can be easily convert into 3D Z-axis rotation equations.
- Rotation about z axis we leave z coordinate unchanged.

  $\boldsymbol{x' = x\cos\theta - y\sin\theta}$

  $\boldsymbol{y' = x\sin\theta + y\cos\theta}$

  $\boldsymbol{z' = z}$

  Where Parameter $\boldsymbol{\theta}$ specify rotation angle.
- Matrix equation is written as:

  $\boldsymbol{P' = R_z(\theta) \cdot P}$

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

X-Axis Rotation

- Transformation equation for x-axis is obtain from equation of z-axis rotation by replacing cyclically as shown here

  $\boldsymbol{x \rightarrow y \rightarrow z \rightarrow x}$
- Rotation about x axis we leave x coordinate unchanged.

  $\boldsymbol{y' = y\cos\theta - z\sin\theta}$

  $\boldsymbol{z' = y\sin\theta + z\cos\theta}$

  $\boldsymbol{x' = x}$

  Where Parameter $\boldsymbol{\theta}$ specify rotation angle.
- Matrix equation is written as:

  $\boldsymbol{P' = R_x(\theta) \cdot P}$

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

Y-A xis Rotation

- Transformation equation for y-axis is obtain from equation of x-axis rotation by replacing cyclically as shown here

  $\boldsymbol{x \rightarrow y \rightarrow z \rightarrow x}$

- Rotation about y axis we leave y coordinate unchanged.

  $\boldsymbol{z' = z\cos\theta - x\sin\theta}$

  $\boldsymbol{x' = z\sin\theta + x\cos\theta}$

  $\boldsymbol{y' = y}$

  Where Parameter $\boldsymbol{\theta}$ specify rotation angle.
- Matrix equation is written as:

  $\boldsymbol{P' = R_y(\theta)\cdot P}$

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

- Example: - Rotate the point P(5,5,5) 90° about Z axis.

  $P' = R_z(\theta)\cdot P$

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

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos 90 & -\sin 90 & 0 & 0 \\ \sin 90 & \cos 90 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 5 \\ 5 \\ 5 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} -5 \\ 5 \\ 5 \\ 1 \end{bmatrix}$$

  Final coordinate after rotation is P' (-5, 5, 5).

General 3D Rotations when rotation axis is parallel to one of the standard axis

- Three steps require to complete such rotation
  1. Translate the object so that the rotation axis coincides with the parallel coordinate axis.
  2. Perform the specified rotation about that axis.
  3. Translate the object so that the rotation axis is moved back to its original position.
- This can be represented in equation form as:

  $\boldsymbol{P' = T^{-1}\cdot R(\theta)\cdot T\cdot P}$

General 3D Rotations when rotation axis is inclined in arbitrary direction

- When object is to be rotated about an axis that is not parallel to one of the coordinate axes, we need rotations to align the axis with a selected coordinate axis and to bring the axis back to its original orientation.
- Five steps require to complete such rotation.
  1. Translate the object so that the rotation axis passes through the coordinate origin.
  2. Rotate the object so that the axis of rotation coincides with one of the coordinate axes.
  3. Perform the specified rotation about that coordinate axis.
  4. Apply inverse rotations to bring the rotation axis back to its original orientation.
  5. Apply the inverse translation to bring the rotation axis back to its original position.
- We can transform rotation axis onto any of the three coordinate axes. The Z-axis is a reasonable choice.

- We are given line in the form of two end points P1 (x1,y1,z1), and P2 (x2,y2,z2).
- We will see procedure step by step.

1) Translate the object so that the rotation axis passes through the coordinate origin.

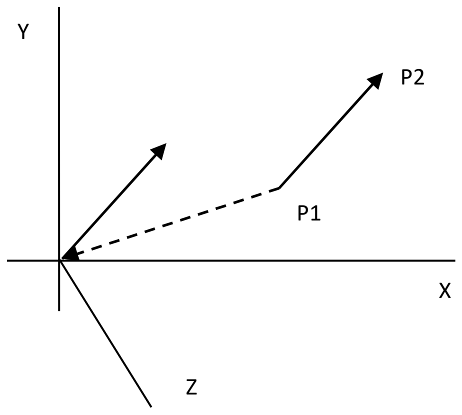


Fig. 5.3: - Translation of vector V.

- For translation of step one we will bring first end point at origin and transformation matrix for the same is as below

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

2) Rotate the object so that the axis of rotation coincides with one of the coordinate axes.

- This task can be completed by two rotations first rotation about x-axis and second rotation about y-axis.
- But here we do not know rotation angle so we will use dot product and vector product.
- Lets write rotation axis in vector form.

$$V = P_2 - P_1 = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$$

- Unit vector along rotation axis is obtained by dividing vector by its magnitude.

$$u = \frac{V}{|V|} = \left(\frac{x_2 - x_1}{|V|}, \frac{y_2 - y_1}{|V|}, \frac{z_2 - z_1}{|V|}\right) = (a, b, c)$$

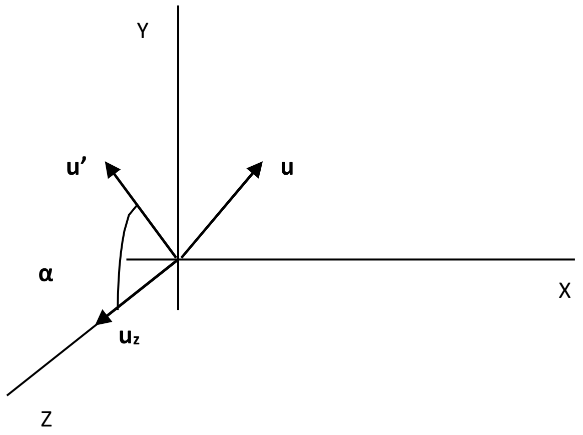


Fig. 5.4: - Projection of u on YZ-Plane.

- Now we need cosine and sin value of angle between unit vector '**u**' and XZ plane and for that we will take projection of u on YZ-plane say '**u''** and then find dot product and cross product of '**u''** and '**u**$_z$' .
- Coordinate of '**u''** is (0,b,c) as we will take projection on YZ-plane x value is zero.

$$u' \cdot u_z = |u'||u_z| \cos\alpha$$

$$\cos\alpha = \frac{u' \cdot u_z}{|u'||u_z|} = \frac{(0,b,c)(0,0,1)}{\sqrt{b^2+c^2}} = \frac{c}{d} \quad where\ d = \sqrt{b^2+c^2}$$

And

$u' \times u_z = u_x|u'||u_z| \sin\alpha = u_x \cdot b$

$u_x|u'||u_z| \sin\alpha = u_x \cdot b$

Comparing magnitude

$|u'||u_z| \sin\alpha = b$

$\sqrt{b^2+c^2} \cdot (1) \sin\alpha = b$

$d \sin\alpha = b$

$\sin\alpha = \frac{b}{d}$

- Now we have $\sin\alpha$ and $\cos\alpha$ so we will write matrix for rotation about X-axis.

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{d} & -\frac{b}{d} & 0 \\ 0 & \frac{b}{d} & \frac{c}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- After performing above rotation '**u**' will rotated into '**u''**' in XZ-plane with coordinates (a, 0, √(b^2+c^2)). As we know rotation about x axis will leave x coordinate unchanged, '**u''**' is in XZ=plane so y coordinate is zero, and z component is same as magnitude of '**u''**.
- Now rotate '**u''**' about Y-axis so that it coincides with Z-axis.

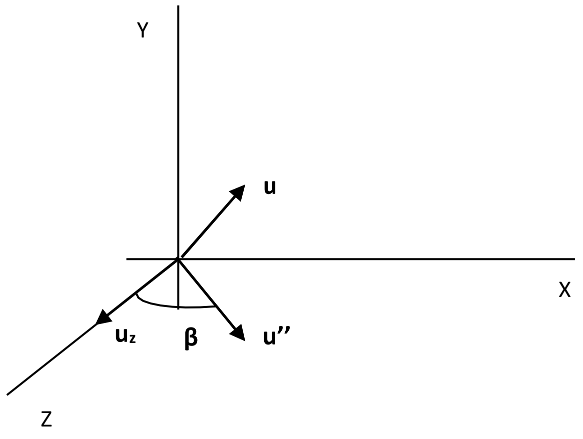


Fig. 5.5: - Rotation of u about X-axis.

- For that we repeat above procedure between '**u''**' and '**u$_z$**' to find matrix for rotation about Y-axis.

$u'' \cdot u_z = |u''||u_z| \cos\beta$

$$\cos\beta = \frac{u' \cdot u_z}{|u'||u_z|} = \frac{(a,0,\sqrt{b^2+c^2})\,(0,0,1)}{1} = \sqrt{b^2+c^2} = d \quad where\ d = \sqrt{b^2+c^2}$$

And

$u'' \times u_z = u_y|u''||u_z| \sin\beta = u_y \cdot (-a)$

$u_y|u''||u_z| \sin\beta = u_y \cdot (-a)$

Comparing magnitude

$|u''||u_z| \sin\beta = (-a)$

$(1)\ \sin\beta = -a$

$\sin\beta = -a$

- Now we have $\sin\beta$ and $\cos\beta$ so we will write matrix for rotation about Y-axis.

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} d & 0 & -a & 0 \\ 0 & 1 & 0 & 0 \\ a & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Now by combining both rotation we can coincides rotation axis with Z-axis

3) Perform the specified rotation about that coordinate axis.

- As we know we align rotation axis with Z axis so now matrix for rotation about z axis

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4) Apply inverse rotations to bring the rotation axis back to its original orientation.

- This step is inverse of step number 2.

5) Apply the inverse translation to bring the rotation axis back to its original position.

6) This step is inverse of step number 1.

So finally sequence of transformation for general 3D rotation is

$$P' = T^{-1} \cdot R_x{}^{-1}(\alpha) \cdot R_y{}^{-1}(\beta) \cdot R_z(\theta) \cdot R_y(\beta) \cdot R_x(\alpha) \cdot T \cdot P$$

Scaling

- It is used to resize the object in 3D space.
- We can apply uniform as well as non uniform scaling by selecting proper scaling factor.
- Scaling in 3D is similar to scaling in 2D. Only one extra coordinate need to consider into it.

Coordinate Axes Scaling

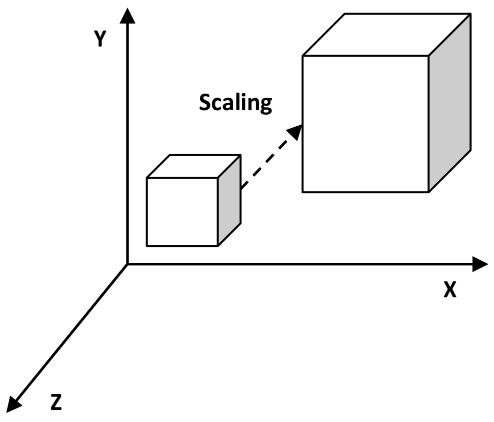


Fig. 5.6: - 3D Scaling.

- Simple coordinate axis scaling can be performed as below.

$P' = S \cdot P$

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

- Example: - Scale the line AB with coordinates (10,20,10) and (20,30,30) respectively with scale factor S(3,2,4).

  $P' = S \cdot P$

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

  $$\begin{bmatrix} A_x' & B_x' \\ A_y' & B_y' \\ A_z' & B_z' \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 10 & 20 \\ 20 & 30 \\ 10 & 30 \\ 1 & 1 \end{bmatrix}$$

  $$\begin{bmatrix} A_x' & B_x' \\ A_y' & B_y' \\ A_z' & B_z' \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 30 & 60 \\ 40 & 60 \\ 40 & 120 \\ 1 & 1 \end{bmatrix}$$

  Final coordinates after scaling are A′ (30, 40, 40) and B′ (60, 60, 120).

Fixed Point Scaling

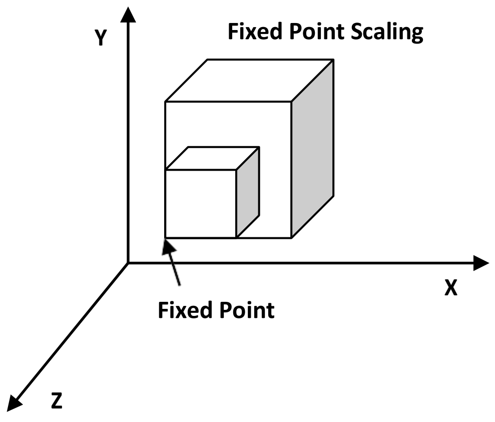


Fig. 5.7: - 3D Fixed point scaling.

- Fixed point scaling is used when we require scaling of object but particular point must be at its original position.
- Fixed point scaling matrix can be obtained in three step procedure.
  1. Translate the fixed point to the origin.
  2. Scale the object relative to the coordinate origin using coordinate axes scaling.
  3. Translate the fixed point back to its original position.
- Let's see its equation.

  $$\boldsymbol{P' = T(x_f, y_f, z_f) \cdot S(s_x, s_y, s_z) \cdot T(-x_f, -y_f, -z_f) \cdot P}$$

  $$\boldsymbol{P'} = \begin{bmatrix} 1 & 0 & 0 & x_f \\ 0 & 1 & 0 & y_f \\ 0 & 0 & 1 & z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -x_f \\ 0 & 1 & 0 & -y_f \\ 0 & 0 & 1 & -z_f \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \boldsymbol{P}$$

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

Other Transformations

Reflections

- Reflection means mirror image produced when mirror is placed at require position.
- When mirror is placed in XY-plane we obtain coordinates of image by just changing the sign of z coordinate.
- Transformation matrix for reflection about XY-plane is given below.

$$RF_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Similarly Transformation matrix for reflection about YZ-plane is.

$$RF_x = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Similarly Transformation matrix for reflection about XZ-plane is.

$$RF_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Shears

- Shearing transformation can be used to modify object shapes.
- They are also useful in 3D viewing for obtaining general projection transformations.
- Here we use shear parameter **'a'** and **'b'**
- Shear matrix for Z-axis is given below

$$SH_z = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Similarly Shear matrix for X-axis is.

$$SH_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ a & 1 & 0 & 0 \\ b & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Similarly Shear matrix for X-axis is.

$$SH_y = \begin{bmatrix} 1 & a & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & b & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Viewing Pipeline

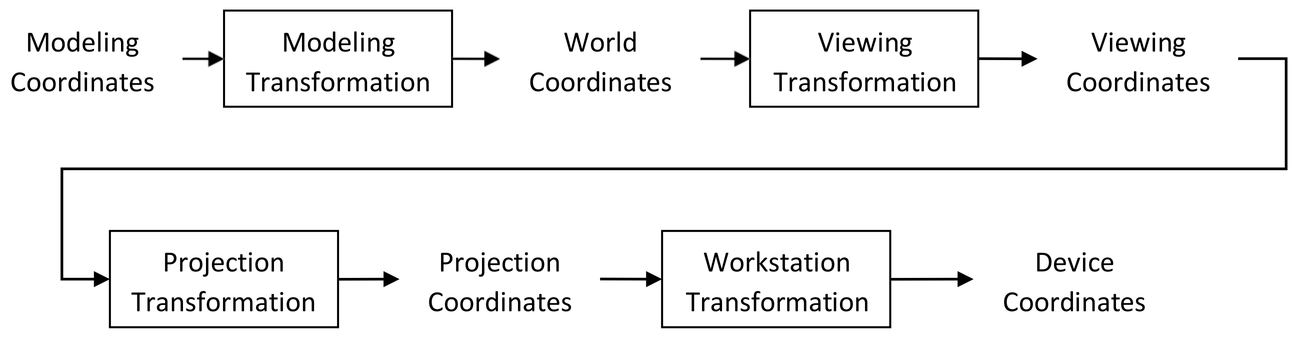


Fig. 5.8: - General 3D viewing pipeline.

- Steps involved in 3D pipeline are similar to the process of taking a photograph.
- As shown in figure that initially we have modeling coordinate of any object which we want to display on the screen.
- By applying modeling transformation we convert modeling coordinates to world coordinates which gives which part or portion is to be display.
- Then by applying viewing transformation we obtain the viewing coordinate which is fitted in viewing coordinate reference frame.
- Then in case of three dimensional objects we have three dimensions coordinate but we need to display that object on two dimensional screens so we apply projection transformation on it which gives projection coordinate.
- Finally projection coordinate is converted into device coordinate by applying workstation transformation which gives coordinates which is specific to particular device.

Viewing Co-ordinates.

- Generating a view of an object is similar to photographing the object.
- We can take photograph from any side with any angle & orientation of camera.
- Similarly we can specify viewing coordinate in ordinary direction.

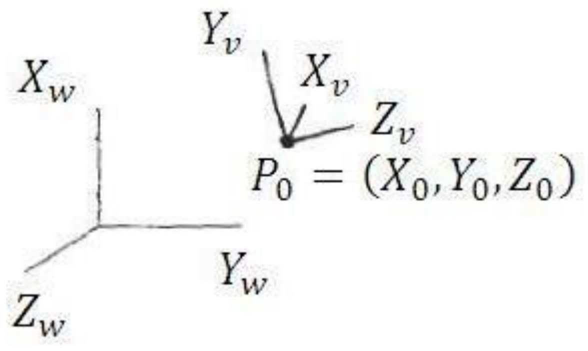


Fig. 5.9: -A right handed viewing coordinate system, with axes Xv, Yv, and Zv, relative to a world-coordinate scene.

Specifying the view plan

- We decide view for a scene by first establishing viewing coordinate system, also referred as view reference coordinate system.
- Then projection plane is setup in perpendicular direction to Zv axis.

- Then projections positions in the scene are transferred to viewing coordinate then viewing coordinate are projected onto the view plane.
- The origin of our viewing coordinate system is called view reference point.
- View reference point is often chosen to be close to or on the surface as same object scene. We can also choose other point also.
- Next we select positive direction for the viewing Zv axis and the orientation of the view plane by specifying the view plane normal vector N.
- Finally we choose the up direction for the view by specifying a vector V called the view up vector. Which specify orientation of camera.
- View up vector is generally selected perpendicular to normal vector but we can select any angle between V & N.
- By fixing view reference point and changing direction of normal vector N we get different views of same object this is illustrated by figure below.

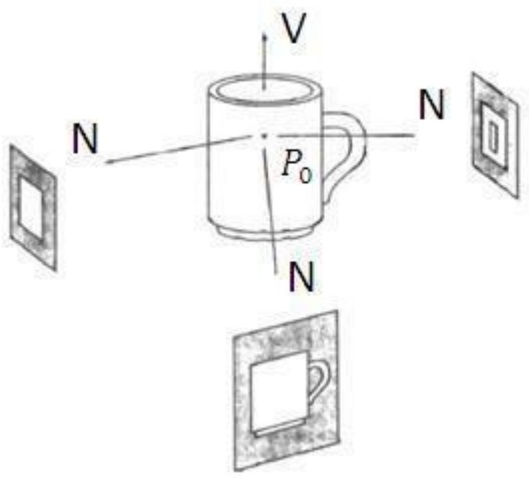


Fig. 5.10: -Viewing scene from different direction with a fixed view-reference point.

Transformation from world to viewing coordinates

- Before taking projection of view plane object description is need to transfer from world to viewing coordinate.
- It is same as transformation that superimposes viewing coordinate system to world coordinate system.
- It requires following basic transformation.
- 1) Translate view reference point to the origin of the world coordinate system.
- 2) Apply rotation to align

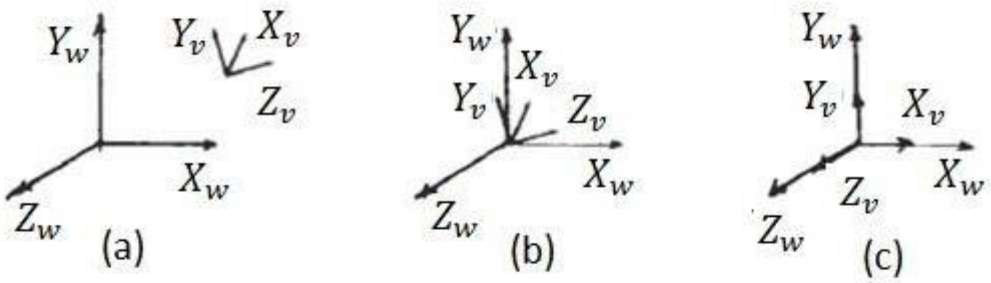


Fig. 5.11: - Aligning a viewing system with the world-coordinate axes using a sequence of translate-rotate transformations.

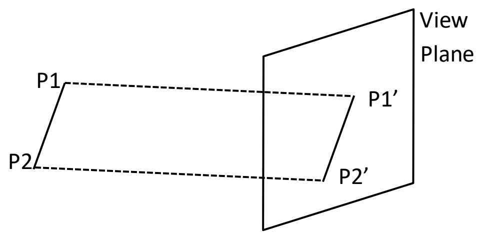
- As shown in figure the steps of transformation

- Consider view reference point in world coordinate system is at position (x_0, y_0, z_0)than for align view reference point to world origin we perform translation with matrix:
- $$T = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- Now we require rotation sequence up-to three coordinate axis rotations depending upon direction we choose for N.
- In general case N is at arbitrary direction then we can align it with word coordinate axes by rotation sequence $Rz \cdot Ry \cdot Rx$.
- Another method for generating the rotation transformation matrix is to calculate unit *uvn* vectors and from the composite rotation matrix directly.
- Here
  $$n = \frac{N}{|N|} = (n_1, n_2 , n_3)$$
  $$u = \frac{V \times N}{|V \times N|} = (u_1, u_2 , u_3)$$
  $$v = n \times u = (v_1, v_2, v_3)$$
- This method also automatically adjusts the direction for *u* so that *v* is perpendicular to *n*.
- Than composite rotation matrix for the viewing transformation is then:
  $$R = \begin{bmatrix} u_1 & u_2 & u_3 & 0 \\ v_1 & v_2 & v_3 & 0 \\ n_1 & n_2 & n_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- This aligns *u* to Xw axis, *v* to Yw axis and *n* to Zw axis.
- Finally composite matrix for world to viewing coordinate transformation is given by:
- $M_{wc,vc} = R \cdot T$
- This transformation is applied to object's coordinate to transfer them to the viewing reference frame.

Projections

- Once world-coordinate descriptions of the objects in a scene are converted to viewing coordinates, we can project the three-dimensional objects onto the two-dimensional view plane.
- Process of converting three-dimensional coordinates into two-dimensional scene is known as **projection**.
- There are two projection methods namely.
  1. Parallel Projection.
  2. Perspective Projection.
- Lets discuss each one.

Parallel Projections

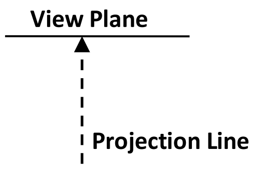


center of projection is at infinite distance

Fig. 5.12: - Parallel projection.

- In a parallel projection, coordinate positions are transformed to the view plane along parallel lines, as shown in the, example of above Figure.
- We can specify a parallel projection with a projection vector that defines the direction for the projection lines.
- It is further divide into two types.
  1. Orthographic parallel projection.
  2. Oblique parallel projection.

Orthographic parallel projection



types

1. front view
2. top view
3. side view

Fig. 5.13: - Orthographic parallel projection.

- When the projection lines are perpendicular to the view plane, we have an orthographic parallel projection.
- Orthographic projections are most often used to produce the front, side, and top views of an object, as shown in Fig.

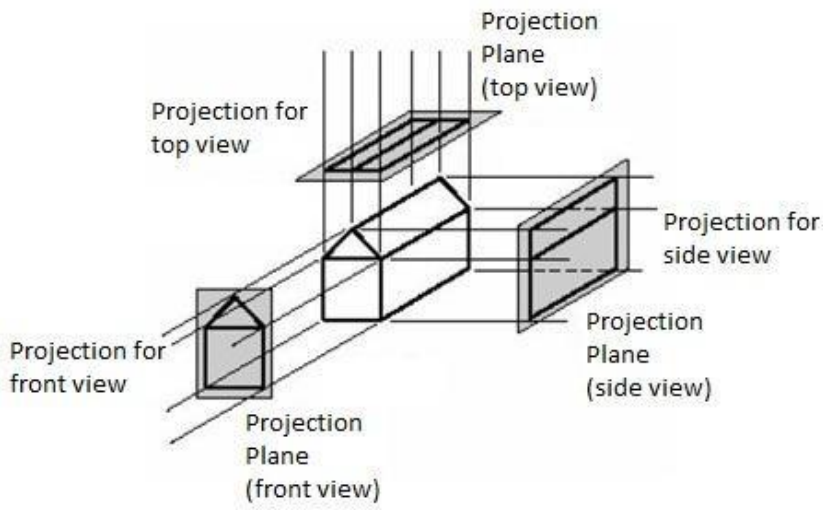

Projection Plane (top view)
Projection for top view
Projection for side view
Projection Plane (side view)
Projection for front view
Projection Plane (front view)

Fig. 5.14: - Orthographic parallel projection.

- Engineering and architectural drawings commonly use orthographic projections, because lengths and angles are accurately depicted and can be measure from the drawings.
- We can also form orthographic projections that display more than one face of an object. Such view are called **axonometric orthographic projections**. Very good example of it is **Isometric** projection.
- Transformation equations for an orthographic parallel projection are straight forward.
- If the view plane is placed at position z_{vp} along the z_v axis, then any point (x, y, z) in viewing coordinates is transformed to projection coordinates as

  $x_p = x, \qquad y_p = y$

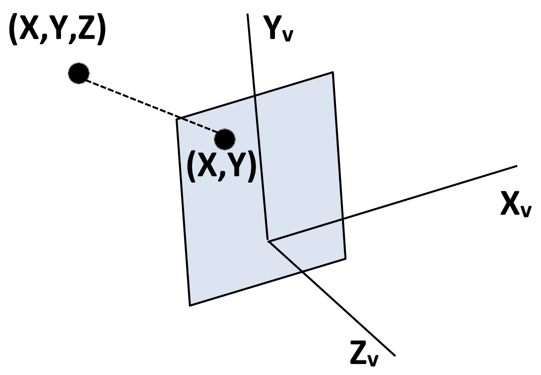


Fig. 5.15: - Orthographic parallel projection.

- Where the original z-coordinate value is preserved for the depth information needed in depth cueing and visible-surface determination procedures.

Oblique parallel projection.

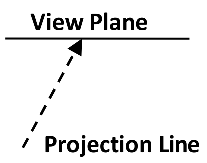


Fig. 5.16: - Oblique parallel projection.

- An oblique projection is obtained by projecting points along parallel lines that are not perpendicular to the projection plane.
- Coordinate of oblique parallel projection can be obtained as below.

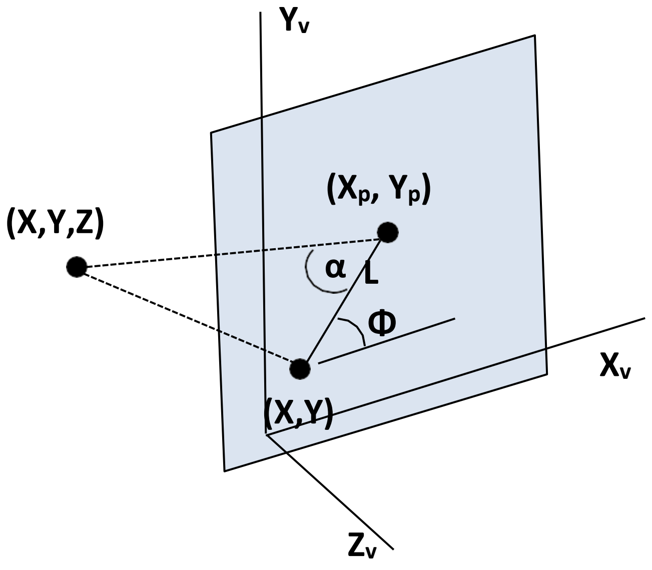


Fig. 5.17: - Oblique parallel projection.

- As shown in the figure (X,Y,Z) is a point of which we are taking oblique projection (Xp,Yp) on the view plane and point (X,Y) on view plane is orthographic projection of (X,Y,Z).
- Now from figure using trigonometric rules we can write

$$x_p = x + L\cos\emptyset$$
$$y_p = y + L\sin\emptyset$$

- Length L depends on the angle α and the z coordinate of the point to be projected:

$$\tan\alpha = \frac{Z}{L}$$
$$L = \frac{Z}{\tan\alpha}$$
$$L = ZL_1, \qquad Where\ L_1 = \frac{1}{\tan\alpha}$$

- Now put the value of L in projection equation.

$$x_p = x + ZL_1\ \cos\emptyset$$
$$y_p = y + ZL_1\ \sin\emptyset$$

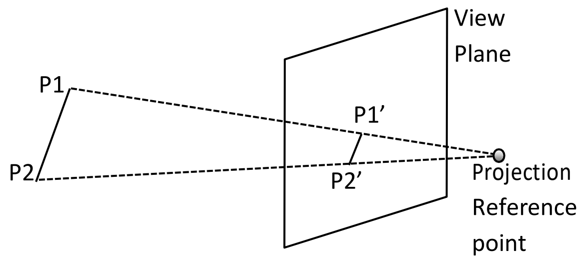
- Now we will write transformation matrix for this equation.

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

- This equation can be used for any parallel projection. For orthographic projection L_1=0 and so whole term which is multiply with z component is zero.
- When value of $\tan\alpha = 1$ projection is known as **Cavalier projection**.
- When value of $\tan\alpha = 2$ projection is known as **Cabinet projection**.

Perspective Projection

size of object is inverse of distance of obj



center of projection is at finite distance

--> similar to human visual system

Fig. 5.18: - Perspective projection.

- In perspective projection object positions are transformed to the view plane along lines that converge to a point called the **projection reference point** (or **center of projection** or **vanishing point**).

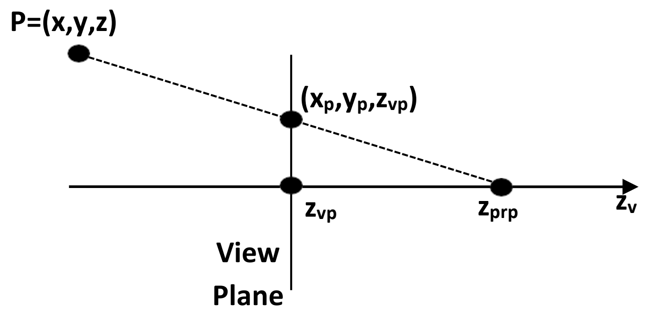


Fig. 5.19: - Perspective projection.

- Suppose we set the projection reference point at position z_{prp} along the z_v axis, and we place the view plane at z_{vp} as shown in Figure above. We can write equations describing coordinate positions along this perspective projection line in parametric form as

$$x' = x - xu$$
$$y' = y - yu$$
$$z' = z - (z - z_{prp})u$$

- Here parameter u takes the value from 0 to 1, which is depends on the position of object, view plane, and projection reference point.
- For obtaining value of u we will put z'=z_{vp} and solve equation of z'.

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

- Now substituting value of u in equation of x' and y' we will obtain.

$$x_p = x\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) = x\left(\frac{d_p}{z_{prp} - z}\right)$$
$$y_p = y\left(\frac{z_{prp} - z_{vp}}{z_{prp} - z}\right) = y\left(\frac{d_p}{z_{prp} - z}\right), \quad Where\ d_p = z_{prp} - z_{vp}$$

- Using three dimensional homogeneous-coordinate representations, we can write the perspective projection transformation matrix form as.

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

- In this representation, the homogeneous factor is.

$$h = \frac{z_{prp} - z}{d_p}\ and$$
$$x_p = x_h/h\ and\ y_p = y_h/h$$

- There are number of special cases for the perspective transformation equations.
- If view plane is taken to be uv plane, then $z_{vp} = 0$ and the projection coordinates are.

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

- If we take projection reference point at origin than $z_{prp} = 0$ and the projection coordinates are.

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

- The vanishing point for any set of lines that are parallel to one of the principal axes of an object is referred to as a principal vanishing point
- We control the number of principal vanishing points (one, two, or three) with the orientation of the projection plane, and perspective projections are accordingly classified as one-point, two-point, or three-point projections.
- The number of principal vanishing points in a projection is determined by the number of principal axes intersecting the view plane.

View Volumes and General Projection Transformations

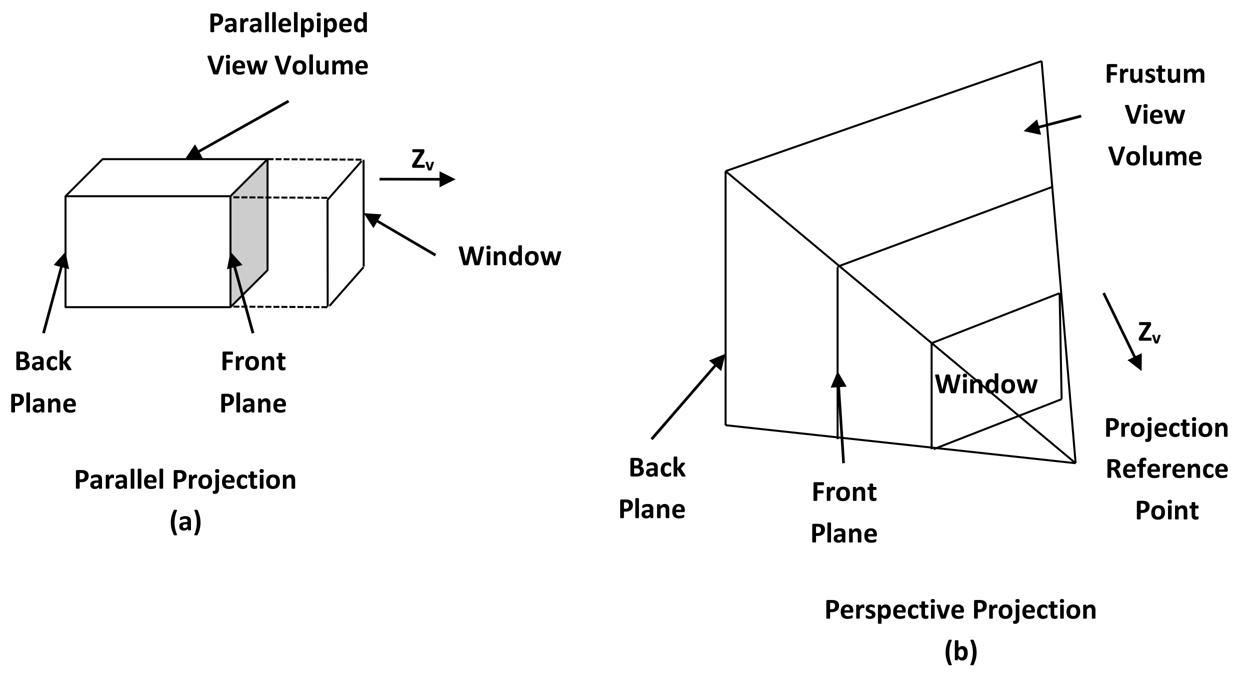


Fig. 5.20: - View volume of parallel and perspective projection.

- Based on view window we can generate different image of the same scene.
- Volume which is appears on the display is known as view volume.
- Given the specification of the view window, we can set up a view volume using the window boundaries.
- Only those objects within the view volume will appear in the generated display on an output device; all others are clipped from the display.
- The size of the view volume depends on the size of the window, while the shape of the view volume depends on the type of projection to be used to generate the display.
- A finite view volume is obtained by limiting the extent of the volume in the z_v direction.
- This is done by specifying positions for one or two additional boundary planes. These z_v-boundary planes are referred to as the **front plane** and **back plane**, or the **near plane** and the **far plane**, of the viewing volume.
- Orthographic parallel projections are not affected by view-plane positioning, because the projection lines are perpendicular to the view plane regardless of its location.
- Oblique projections may be affected by view-plane positioning, depending on how the projection direction is to be specified.

General Parallel-Projection Transformation

- Here we will obtain transformation matrix for parallel projection which is applicable to both orthographic as well as oblique projection.

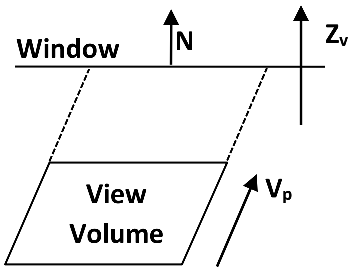


Fig. 5.21: - General parallel projection.

- As shown on figure parallel projection is specified with a projection vector from the projection reference point to the view window.
- Now we will apply shear transformation so that view volume will convert into regular parallelepiped and projection vector will become parallel to normal vector N.

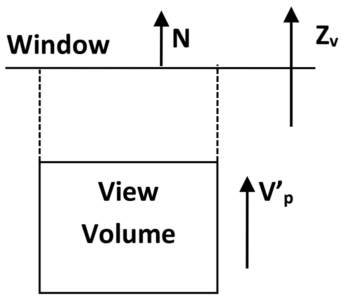


Fig. 5.22: - Shear operation in General parallel projection.

- Let's consider projection vector $\boldsymbol{V_p = (p_x, p_y, p_z)}$.
- We need to determine the elements of a shear matrix that will align the projection vector $\boldsymbol{V_p}$ with the view plane normal vector **N**. This transformation can be expressed as

$$\boldsymbol{V_p' = M_{parallel} \cdot V_p}$$

$$\boldsymbol{V_p' = \begin{bmatrix} 0 \\ 0 \\ p_z \\ 1 \end{bmatrix}}$$

- where $\boldsymbol{M_{parallel}}$ is equivalent to the parallel projection matrix and represents a z-axis shear of the form

$$\boldsymbol{M_{parallel} = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

- Now from above equation we can write

$$\boldsymbol{\begin{bmatrix} 0 \\ 0 \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}}$$

- From matrix we can write.

$$\boldsymbol{0 = p_x + ap_z}$$
$$\boldsymbol{0 = p_y + bp_z}$$

So

$$\boldsymbol{a = \frac{-p_x}{p_z}}, \quad \boldsymbol{b = \frac{-p_y}{p_z}}$$

- Thus, we have the general parallel-projection matrix in terms of the elements of the projection vector as

$$M_{parallel} = \begin{bmatrix} 1 & 0 & \frac{-p_x}{p_z} & 0 \\ 0 & 1 & \frac{p_y}{p_z} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- For an orthographic parallel projection, $p_x = p_y = 0$, and is the identity matrix.

General Perspective-Projection Transformations

- The projection reference point can be located at any position in the viewing system, except on the view plane or between the front and back clipping planes.

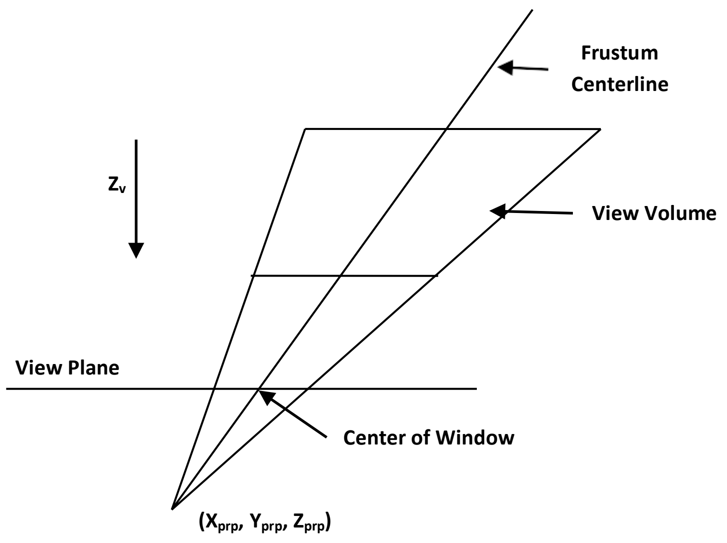


Fig. 5.23: - General perspective projection.

- We can obtain the general perspective-projection transformation with the following two operations:
  1. Shear the view volume so that the center line of the frustum is perpendicular to the view plane.
  2. Scale the view volume with a scaling factor that depends on 1/z .
- A shear operation to align a general perspective view volume with the projection window is shown in Figure.

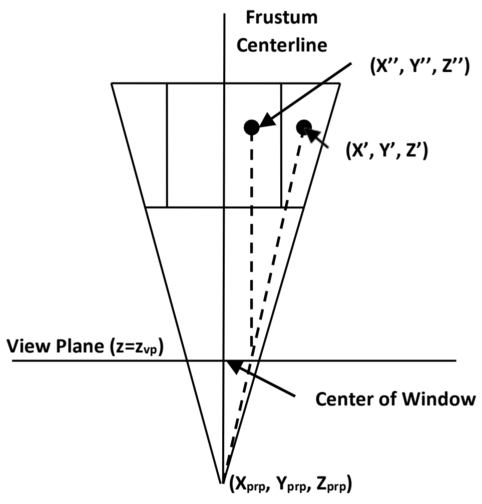


Fig. 5.24: - Shear and scaling operation in general perspective projection.

- With the projection reference point at a general position (X_{prp}, Y_{prp}, Z_{prp}) the transformation involves a combination z-axis shear and a translation:

$$M_{shear} = \begin{bmatrix} 1 & 0 & a & -az_{prp} \\ 0 & 1 & b & -bz_{prp} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where the shear parameters are

$$a = -\frac{x_{prp} - (xw_{min} + xw_{max})/2}{z_{prp}}$$

$$b = -\frac{y_{prp} - \frac{yw_{min}+yw_{max}}{2}}{z_{prp}}$$

- Points within the view volume are transformed by this operation as

$$x' = x + a(z - z_{prp})$$
$$y' = y + b(z - z_{prp})$$
$$z' = z$$

- After shear we apply scaling operation.

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

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

- Homogeneous matrix for this transformation is:

$$M_{scale} = \begin{bmatrix} 1 & 0 & \frac{-x_{prp}}{z_{prp} - z_{vp}} & \frac{x_{prp}z_{vp}}{z_{prp} - z_{vp}} \\ 0 & 1 & \frac{-y_{prp}}{z_{prp} - z_{vp}} & \frac{y_{prp}z_{vp}}{z_{prp} - z_{vp}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{z_{prp} - z_{vp}} & \frac{z_{prp}}{z_{prp} - z_{vp}} \end{bmatrix}$$

- Therefore the general perspective-projection transformation is obtained by equation:

$$M_{perspective} = M_{scale} \cdot M_{shear}$$