

CS 401 COMPUTER GRAPHICS

Module 4

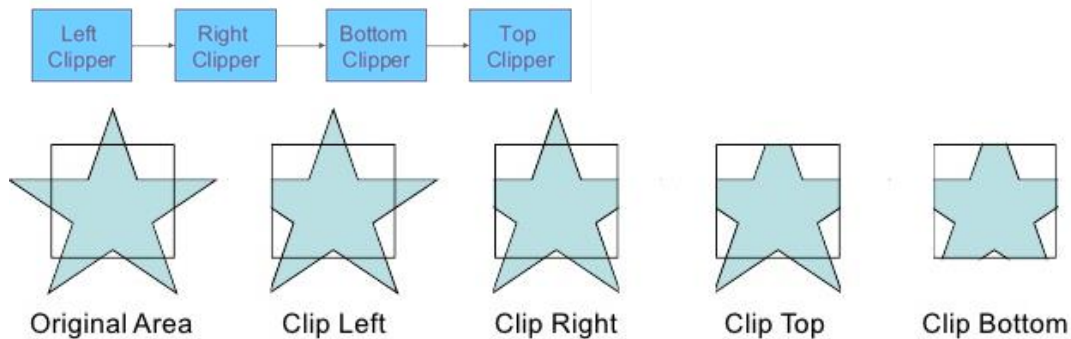
Polygon clipping-Sutherland Hodgeman algorithm, Weiler- Atherton algorithm.

Three dimensional object representation- Polygon surfaces, Quadric surfaces – Basic 3D transformations.

Polygon clipping -Sutherland Hodgeman polygon clipping algorithm

A polygon can also be clipped by specifying the clipping window. Sutherland Hodgeman polygon clipping algorithm is used for polygon clipping. In this algorithm, all the vertices of the polygon are clipped against each edge of the clipping window.

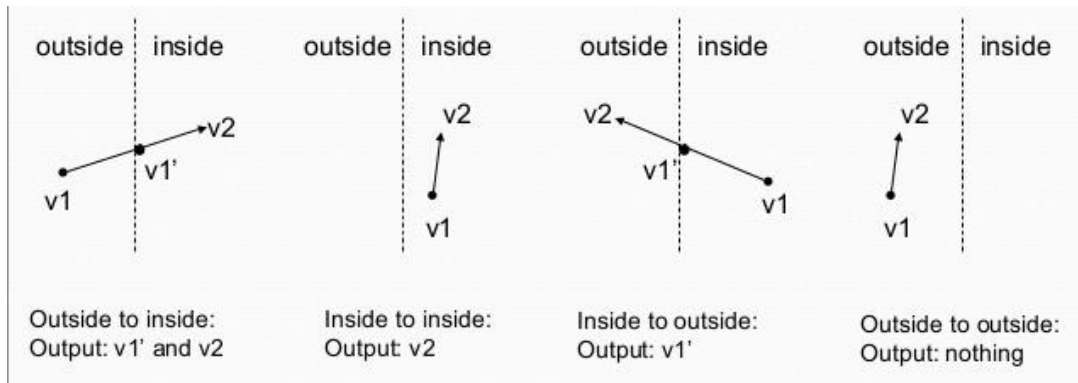
First the polygon is clipped against the left edge of the polygon window to get new vertices of the polygon. These new vertices are used to clip the polygon against right edge, top edge, bottom edge, of the clipping window as shown in the following figure. While processing an edge of a polygon with clipping window, an intersection point is found if edge is not completely inside clipping window and the a partial edge from the intersection point to the outside edge is clipped. The following figures show left, right, top and bottom edge clippings.



Beginning with the initial set of polygon vertices, we first clip the polygon against left boundary to produce new sequence of vertices. The new set of vertices is then successively passed to a right boundary, a bottom boundary, and a top boundary clipper. At each step, a new set of vertices is generated & passed to the next window clipper.

There are 4 possible cases when processing vertices in sequence around the polygon edges. For each pair of vertices, we make the following tests

1. **Both vertices are inside** : Only the second vertex is added to the output list
2. **First vertex is outside while second one is inside** : Both the point of intersection of the edge with the clip boundary and the second vertex are added to the output list
3. **First vertex is inside while second one is outside** : Only the point of intersection of the edge with the clip boundary is added to the output list
4. **Both vertices are outside** : No vertices are added to the output list



Example:

*We illustrate this algorithm by processing the area in figure against the **left** window boundary.*

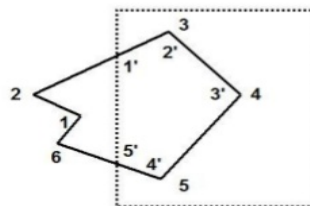
*Vertices **1** and **2** are **outside** of the boundary.*

*Vertex **3**, which is **inside**, **1'** and vertex **3** are saved.*

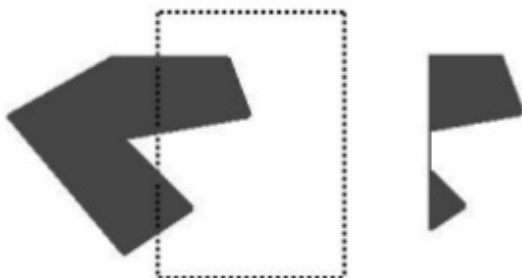
*Vertex **4** and **5** are **inside**, and they also saved.*

*Vertex **6** is **outside**, **5'** is saved.*

Using the five saved points, we would repeat the process for the next window boundary.



Disadvantage: The algorithm correctly clips convex polygons, but may display extraneous lines for concave polygons. So this algorithm is not suited for concave polygons.

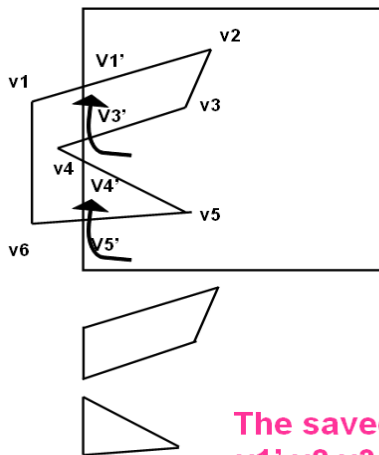


Weiler Atherton polygon clipping algorithm

The vertex-processing procedures for window boundaries are modified so that concave polygons are displayed correctly. This clipping procedure was developed as a method for identifying visible surfaces, and so it can be applied with arbitrary polygon-clipping regions.

The basic idea in this algorithm is that instead of always proceeding around the polygon edges as vertices are processed, we sometimes want to follow the window boundaries. Which path we follow depends on the polygon-processing direction (clockwise or counterclockwise) and whether pair of polygon vertices currently being processed represents an outside-to-inside pair or an inside-to-outside pair. For clockwise processing of polygon vertices, we use the following rules:

- For an outside-to-inside pair of vertices, follow the polygon boundary.
- For an inside-to-outside pair of vertices, follow the window boundary in a clockwise direction.

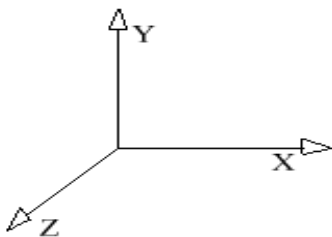


- outside to inside pair follow the polygon boundary.
- inside to outside pair follow the window boundary in clockwise direction

The saved vertices are
v1' v2 v3 v3' v4' v5 v5'

Introduction to graphics in three dimension

In the 2D system, we use only two coordinates X and Y but in 3D, an extra coordinate Z is added. 3D graphics techniques and their application are fundamental to the entertainment, games, and computer-aided design industries.



3D transformations

Methods for geometric transformations and object modeling in three dimensions are extended from two-dimensional methods by including considerations for the z coordinate.

3D Translation

We now translate an object by specifying a three-dimensional translation vector, which determines how much the object is to be moved in each of the three coordinate directions.

P is translated to P' by:

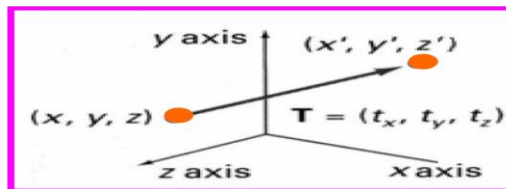
$$\mathbf{P}' = \mathbf{T} \cdot \mathbf{P}$$

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

$$x' = x + t_x$$

$$y' = y + t_y$$

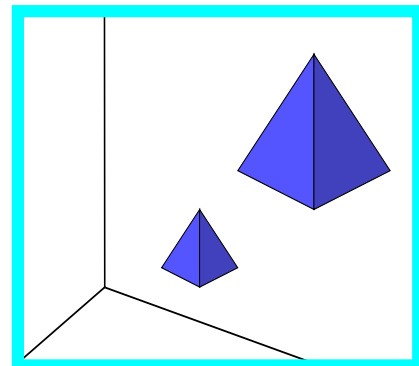
$$z' = z + t_z$$



3D Scaling

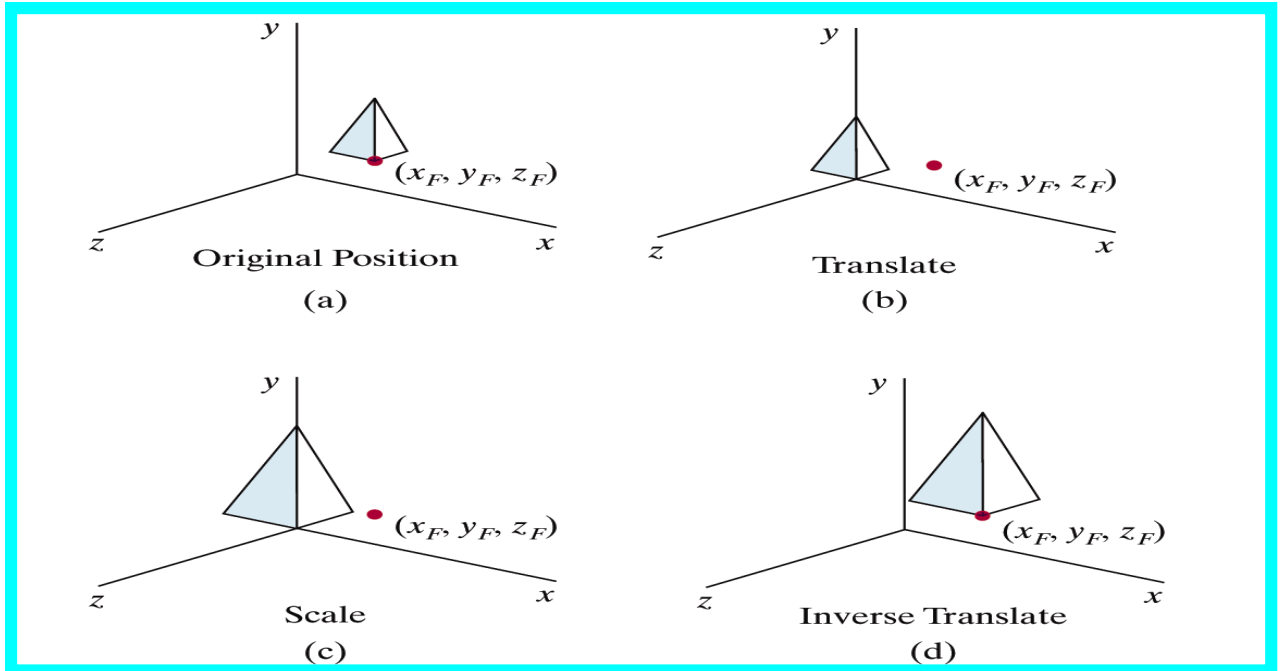
Scaling about origin: Changes the size of the object and repositions the object relative to the coordinate origin

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



Scaling about any fixed point:

- Translate object so that the fixed point coincides with the coordinate origin
- Scale the object with respect to the coordinate origin
- Use the inverse translation of step 1 to return the object to its original position



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

3D Reflections

About an axis: equivalent to 180° rotation about that axis

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

reflection wrt
yz-plane

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

reflection wrt
xz-plane

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

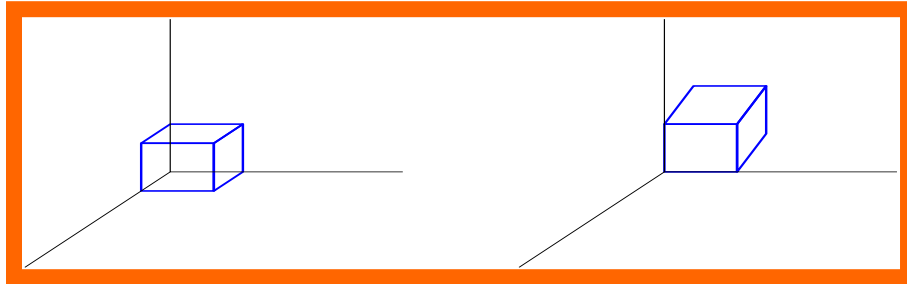
reflection wrt
xy-plane

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

reflection wrt
the origin

3D Shearing

- Modify object shapes
- Useful for perspective projections



X axis shear:

The effect of this transformation matrix is to alter x values by an amount that is proportional to the shearing factors Sh_{x1} and Sh_{x2} value, while leaving the y and z coordinates unchanged.

$$\begin{bmatrix} 1 & sh_{x_1} & sh_{x_2} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ sh_{y_1} & 1 & sh_{y_2} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ sh_{z_1} & sh_{z_2} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

↑ along x-axis,
 ↑ along y-axis,
 ↑ along z-axis

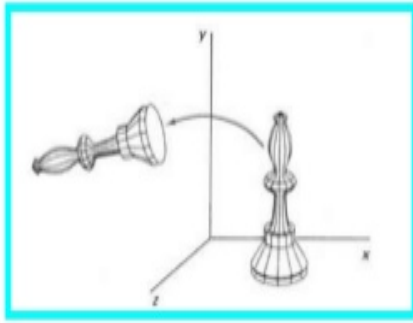
3D Rotations

Coordinate Axis Rotations:

1. Z-axis rotation: For z axis same as 2D rotation.

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

$$\mathbf{P}' = \mathbf{R}_z(\theta) \cdot \mathbf{P}$$



We can obtain rotations around other axes through cyclic permutation of coordinate parameters

2. X-axis rotation

The equation for X-axis rotation

$$\mathbf{x}' = \mathbf{x}$$

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

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

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

3. Y-axis rotation

The equation for Y-axis rotation

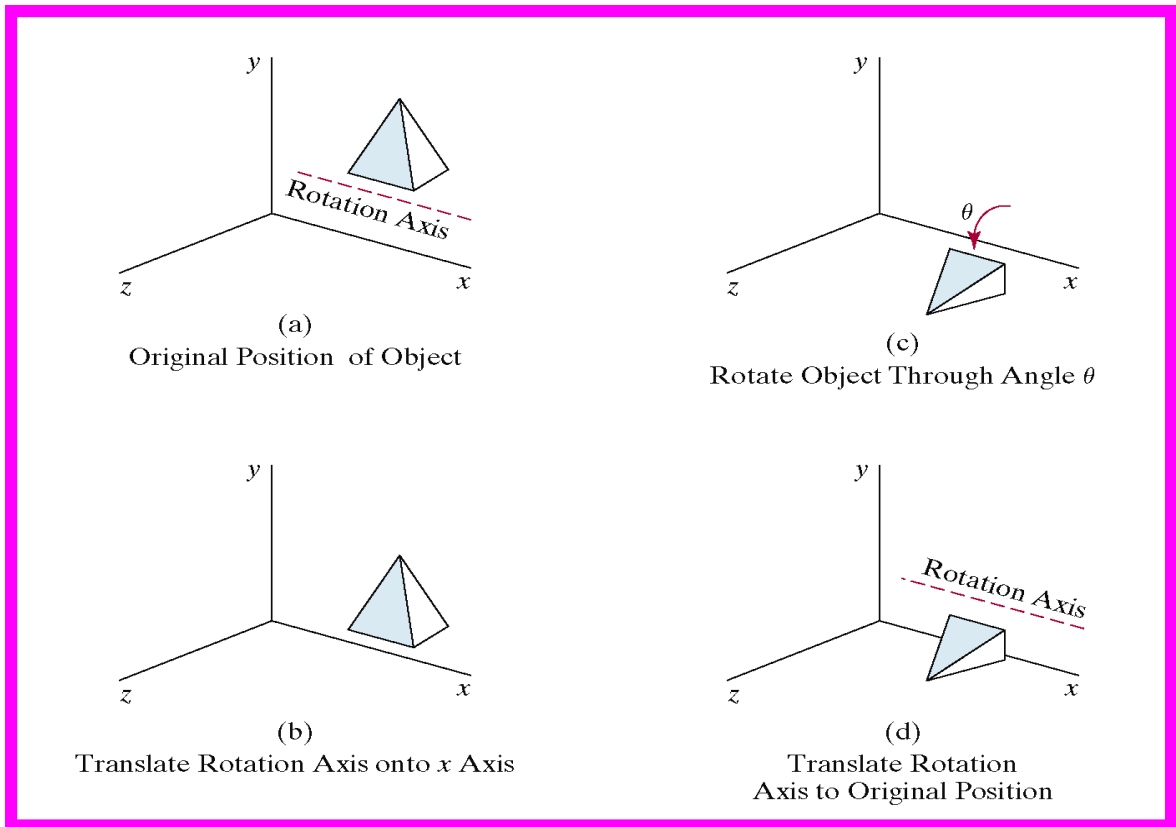
$$\mathbf{x}' = \mathbf{x} \cos \theta + \mathbf{z} \sin \theta$$

$$\mathbf{y}' = \mathbf{y}$$

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

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

4. Rotation about an axis that is parallel to any coordinate axis (Example x axis):



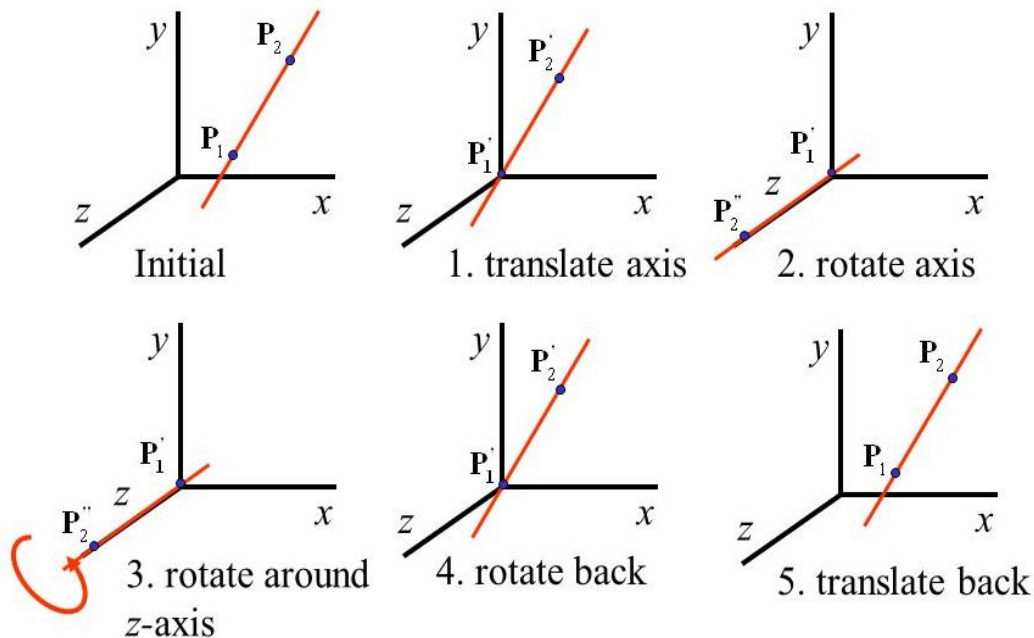
- Translate the object so that the rotation axis coincides with the parallel coordinate axis.
- Perform the specified rotation about that axis.
- Translate the object so that the rotation axis is moved back to its original position

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

5. Rotation about an arbitrary axis that is not parallel to any coordinate axis

Steps:

1. Translate the object so that the rotation axis passes through the coordinate origin
2. Rotate the object so that the axis rotation coincides with one of the coordinate axes
3. Perform the specified rotation about that coordinate axis
4. Apply inverse rotation axis back to its original orientation
5. Apply the inverse translation to bring the rotation axis back to its original position



Composite 3D Transformations

- Like in 2D transformations, we form a composite three-dimensional transformation by multiplying the matrix representations for the individual operations in the transformation sequence.
- This concatenation is carried out from right to left, where the right most matrix is the first transformation to be applied to an object and the leftmost matrix is the last transformation

Three Dimensional display methods

To obtain a display of a three-dimensional scene that has been modeled in world coordinates, we have many display methods.

- Parallel projection
- Perspective projection
- Depth Cueing
- Visible line & surface identification
- Surface rendering
- Exploded and cutaway views
- 3D and stereoscopic views

Parallel projection

Generating a view of a solid object by projecting points on the object surface along parallel lines onto the display plane is called parallel projection. By selecting different viewing positions, we can

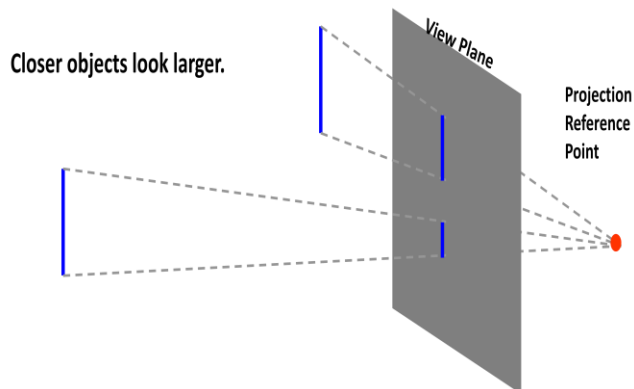
project visible points on the object onto the display plane in different two-dimensional views of the object (front and side elevation, top plan view).



In a parallel projection, parallel lines in the world-coordinate scene project into parallel lines on the two-dimensional display plane. This technique is used in engineering and architectural drawings to represent an object with a set of views that maintain relative proportions of the object.

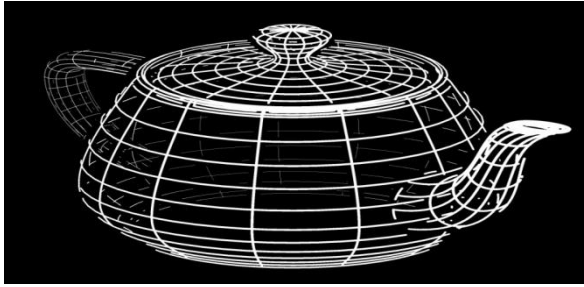
Perspective projection

Generating a view of a 3D scene by projecting points to the display plane along converging paths is called perspective projection. This causes objects farther from the viewing position to be displayed as smaller than objects of the same size that are nearer to the viewing position. Scenes displayed using perspective projections appear more realistic, since this is the way that our eyes and a camera lens form images. Parallel lines appear to converge to a distant point in the background known as Projection reference point. The distant objects appear smaller than objects closer to the viewing position.



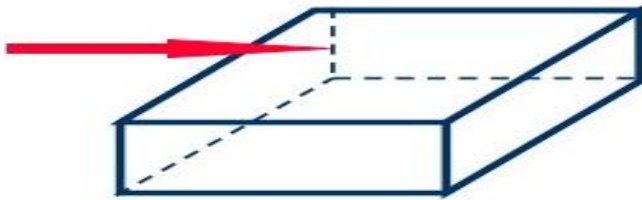
Depth Cueing

Depth information of an object is important so that we can easily identify, for a particular viewing direction, which is the front and which is the back of displayed objects. A simple method for indicating depth with wireframe displays is to vary the intensity of objects according to their distance from the viewing position. The lines closest to the viewing position are displayed with the highest intensities, and lines farther away are displayed with decreasing intensities. Depth cueing is applied by choosing maximum and minimum intensity (or color) values and a range of distances over which the intensities are to vary.



Visible line & surface identification

The simplest method is to highlight the visible lines or to display them in a different color. Another technique, commonly used for engineering drawings, is to display the non visible lines as dashed lines. Another approach is to simply remove the non visible lines, but removing the hidden lines also removes information about the shape of the back surfaces of an object.



Surface rendering

Surface rendering involves the collection of data on a given object in order to create a 3D image of that object on a computer. It is attained by setting the surface intensity of objects according to the lighting conditions in the scene and according to assigned surface characteristics.

Lighting specifications include the intensity and positions of light sources and the general background illumination required for a scene. Surface properties of object include degree of transparency and how rough or smooth the surfaces are to be.



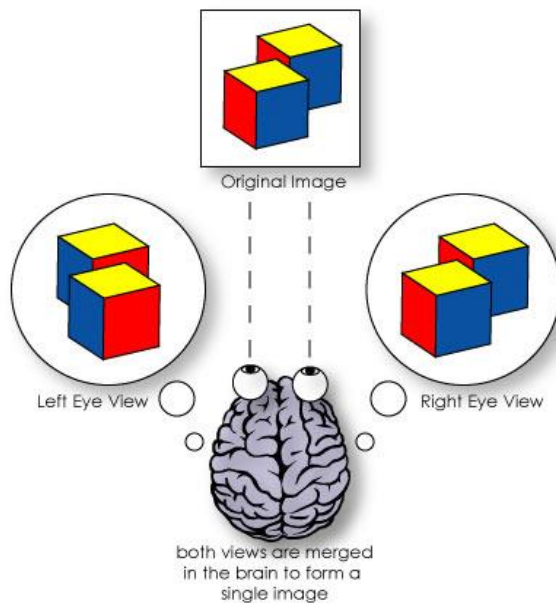
Exploded and cutaway views

Many graphics packages allow objects to be defined as hierarchical structures, so that internal details can be stored. Exploded and cutaway views of such objects can then be used to show the internal structure and relationship of the object parts. Cutaway view removes part of the visible surfaces to show internal structure.



3D and stereoscopic views

Stereoscopic devices **present two views of a scene: one for the left eye and the other for the right eye**. The two views are generated by selecting viewing positions that correspond to the two eye positions of a single viewer. These **two views then can be displayed on alternate refresh cycles of a raster monitor**.

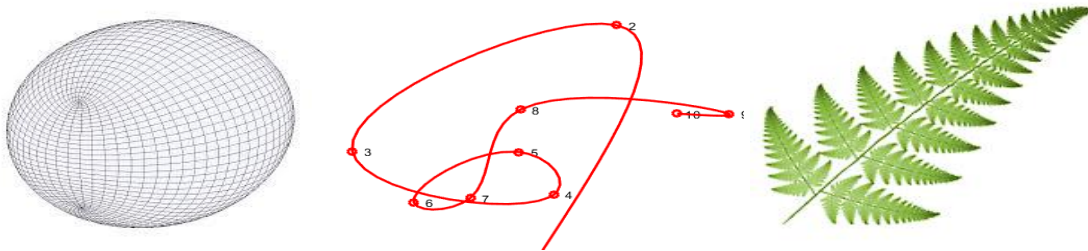


3D Object representations

Graphics scenes can contain many different kinds of objects: trees, flowers, clouds, rocks, water, bricks....there is no one particular method that we can use to describe objects that will include all characteristics of these different materials. To produce realistic displays of scenes, we need to use representations that accurately model object characteristics.

- Polygon and quadric surfaces provide precise description for simple Euclidean objects like polyhedrons, ellipsoids.
- Spline surfaces and construction techniques are useful for designing aircraft wings, gears, and other engineering structures with curved surfaces

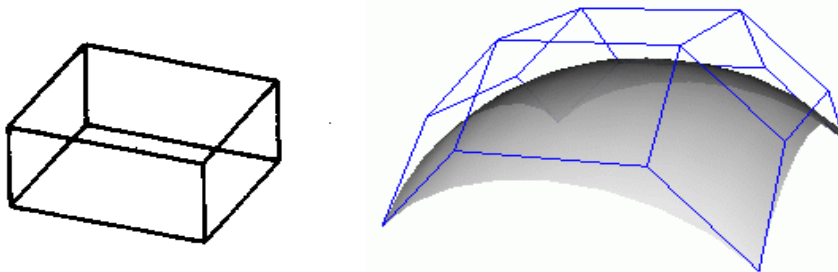
- Procedural methods such as fractals constructions and particle systems give us accurate representations for the natural objects like clouds, trees etc.



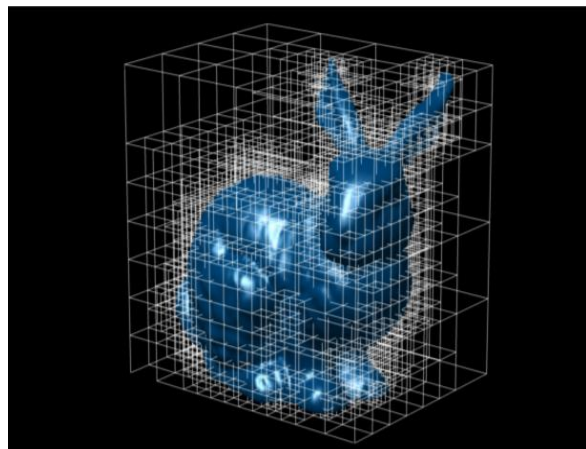
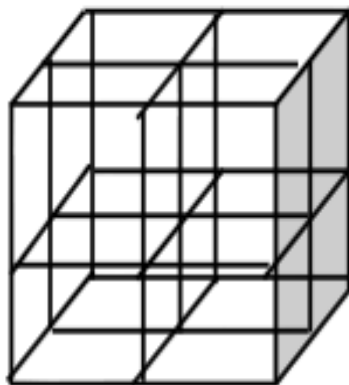
Boundary & Space partitioning representations

The representational schemes of solid objects are divided into two broad categories:

- Boundary representations: Here the 3D object is represented as a set of surfaces that separate the object interior from the environment. Examples are polygonal facets and spline patches.



- Space partitioning representations: These are used to describe the interior properties, by partitioning the spatial regions containing an object into a set of small non-overlapping contiguous solids (usually cubes). Example: Octree (which is the space partitioning description of 3D object).



Polygon Surfaces

The most commonly used boundary representation for a three-dimensional graphics object is a set of surface polygons that enclose the object interior. It simplifies and speeds up the surface rendering and display of objects, since all surfaces are described with linear equations. Realistic renderings are produced by interpolating shading patterns across the polygon surfaces to eliminate or reduce the presence of polygon edge boundaries.

Each graphic object needs some method for its description which could be a polygon table or plane equation etc.

Polygon Tables

It specifies a polygon surface with a set of vertex coordinates and associated attribute parameters. Polygon data tables can be organized into two groups: geometric tables and attribute tables.

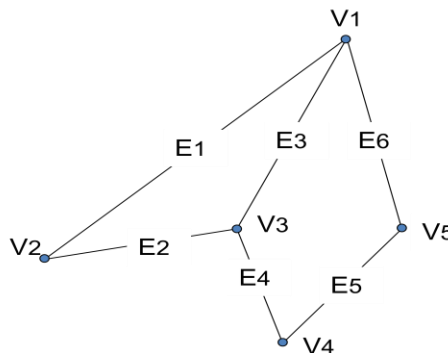
Attribute tables: This table holds object information like transparency, surface reflectivity, texture characteristics, etc., of an object in the scene.

Geometric tables: This table stores the information of vertex coordinates and parameters like slope for each edge, etc. to identify the spatial orientation of polygon surface.

Polygon- Geometric Tables

In order to store geometric information of a polygon properly, this table is further bifurcated into three more tables:

- Vertex table:** Holds **coordinate values of vertices** in the object.
- Edge table:** Holds **pointers back in to the vertex table** for identification of the vertices related to **each polygon edge**.
- Polygon table or polygon surface table:** Holds **pointers back into the edge table** for identification of the **edges related to the polygon surface** under construction.



| 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 |

Some basic tests that should be performed before producing a polygon surface by any graphic package:

- 1) every vertex is listed as an endpoint for at least two edges,
- 2) every edge is part of at least one polygon,
- 3) every polygon is closed,
- 4) each polygon has at least one shared edge,
- 5) if the edge table contains pointer to polygons, every edge referenced by a polygon pointer to polygon, every edge referenced by a polygon pointer has a reciprocal pointer back to polygon.

Plane Equation

Any point on the plane should satisfy the equation of a plane to be zero

$$\text{i.e., } Ax + By + Cz + D = 0.$$

This equation means any point (x, y, z) will only lie on the plane if it satisfies the equation to be zero.

For any point (x, y, z) , if $Ax + By + Cz + D < 0$, point lie on the back of the plane i.e point is inside the surface. For any point (x, y, z) , if $Ax + By + Cz + D > 0$, point lie on the front of the plane i.e point is outside the surface

Using Three noncollinear points (x_1, y_1, z_1) , (x_2, y_2, z_2) and (x_3, y_3, z_3) , we can determine A,B,C,D.
by Solving using Cramers rule,

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \quad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix} \quad C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \quad D = - \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

By taking determinant values, We can calculate

$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$$

$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$$

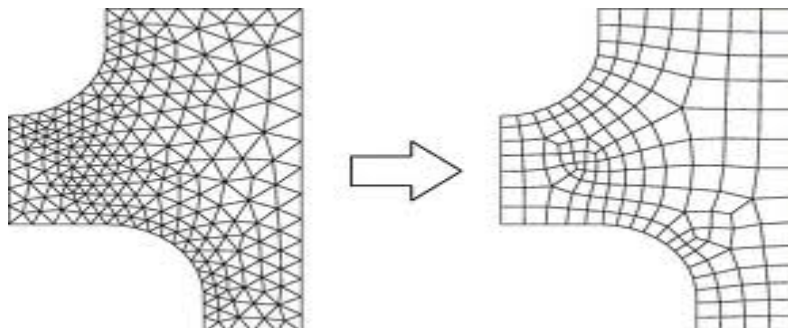
$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$

$$D = -x_1(y_2 z_3 - y_3 z_2) - x_2(y_3 z_1 - y_1 z_3) - x_3(y_1 z_2 - y_2 z_1)$$

As vertex values and other information are entered into the polygon data structure, values for A, B, C and D are computed for each polygon and then can be stored with other polygon data.

Polygon Meshes

It provide several polygon functions for modeling objects. In this method, we specify the surface facets with a mesh function. Main types of polygon mesh is triangle strip and quadrilateral mesh



In this approach, we construct the object from triangles or quadrilaterals by tiling. Most models are constructed this way. This method is easy to work with and perform rendering

Curved lines and surfaces

- **Quadric Surfaces:** This class of objects are described with second degree equations (quadratics)
 - Sphere, Ellipsoid, Torus
- **Superquadrics:** Adding additional parameters to quadric representations to get new object shapes.
 - Superellipse, Superellipsoid

Sphere

A spherical surface with radius r centered on origin is defined as set of points (x,y,z) that satisfy the equation:

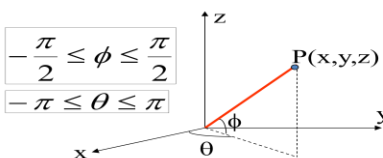
$$x^2 + y^2 + z^2 = r^2$$

- In parametric form,

$$\begin{aligned} x &= r \cos \phi \cos \theta \\ y &= r \cos \phi \sin \theta \\ z &= r \sin \phi \end{aligned}$$

$-\frac{\pi}{2} \leq \phi \leq \frac{\pi}{2}$

$-\pi \leq \theta \leq \pi$



Ellipsoid

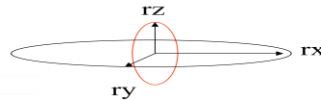
$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 = 1$$

$$x = r_x \cos \phi \cos \theta$$

$$y = r_y \cos \phi \sin \theta$$

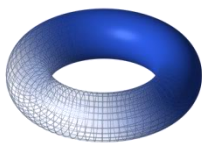
$$z = r_z \sin \phi$$

$$-\pi \leq \theta \leq \pi$$



Torus

Torus is doughnut shaped object generated by rotating a circle around an axis that does not intersect the circle.



$$\left[r - \sqrt{\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2} \right]^2 + \left(\frac{z}{r_z}\right)^2 = 1$$

$$x = r_x (r + \cos \phi) \cos \theta,$$

$$-\pi \leq \phi \leq \pi$$

$$y = r_y (r + \cos \phi) \sin \theta,$$

$$-\pi \leq \theta \leq \pi$$

$$z = r_z \sin \phi$$

Superquadrics

- Adding additional parameters to quadric representations to get new object shapes.
- One additional parameter is added to curve (i.e., 2d) equations and two parameters are added to surface (i.e., 3d) equations.
 - Superellipse
 - Superellipsoid

Superellipse

$$\left(\frac{x}{r_x}\right)^{2/s} + \left(\frac{y}{r_y}\right)^{2/s} = 1 \quad \begin{matrix} x = r_x \cos^s \theta, \\ y = r_y \sin^s \theta \end{matrix} \quad -\pi \leq \theta \leq \pi$$

When s=1, we get normal ellipse.



Super ellipse plotted with different values for s and with condition

Superellipsoid

- Created by adding 2 exponent parameters s1 & s2 to ellipsoid equation.

$$\left[\left(\frac{x}{r_x} \right)^{2/s_2} + \left(\frac{y}{r_y} \right)^{2/s_2} \right]^{s_2/s_1} + \left(\frac{z}{r_z} \right)^{2/s_1} = 1$$

$$x = r_x \cos^{s_1} \phi \cos^{s_2} \theta,$$

$$y = r_y \cos^{s_1} \phi \sin^{s_2} \theta,$$

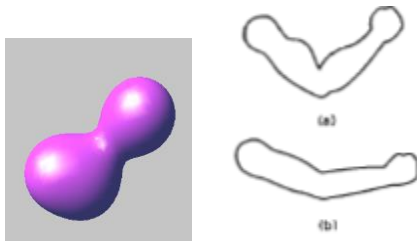
$$z = r_z \sin^{s_1} \phi$$

$$\begin{matrix} -\pi/2 \leq \phi \leq \pi/2 \\ -\pi \leq \theta \leq \pi \end{matrix}$$

- If s1=s2=1, we get ordinary ellipsoid.

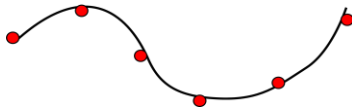
Blobby Objects

Bloppy Objects that do not have a fixed shape. It change their surface characteristics in certain motions or when in proximity to other objects. Eg: Molecular structures, melting objects, muscular movements.



Spline curves

In computer graphics, the term spline curve refers to any composite curve formed with polynomial sections satisfying specified continuity conditions at the boundary of the pieces. A spline surface can be described with two sets of orthogonal spline curves. Spline curves are smooth curves produced from a designated set of points. It is mathematically described using cubic polynomial function. Any composite curve formed with polynomial sections satisfying specified continuity conditions at the boundary . Spline curve is specified by giving a set of coordinate positions called control points.



Applications

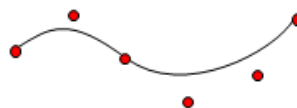
- Used in computer graphics
- to design curve and surface shapes
- to specify animation paths for the objects.

Interpolated & Approximated splines

Spline curves are called as Interpolated splines if curve passes through all control points. It is called as Approximated spline, if it is guided by control points but not necessarily passes through them.



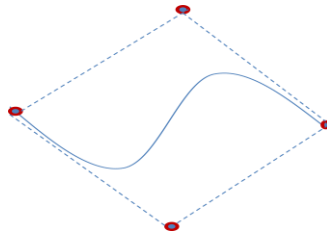
Interpolated



Approximated

Spline curve is defined, modified and manipulated with operations on the control points. After display of curves using these control points, designer can reposition some or all control points to restructure the shape of curve.

Convex polygon boundary that encloses a set of control points is called convex hull. Convex hull provides a measure for the deviation of a curve from the region bounding the control points.



Continuity conditions

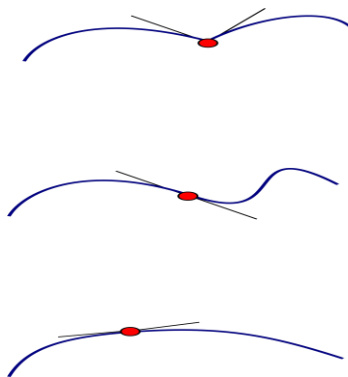
Any composite curve formed with polynomial sections satisfying specified continuity conditions at the boundary .

Parametric continuity conditions

To ensure a smooth transition from one section of parametric curve to the next section ,we set the parametric continuity as C^0 / C^1 / C^2 , If each section of spline is represented using the following Parametric coordinate function:

$$x = x(u), \quad y = y(u), \quad z = z(u), \quad u_1 \leq u \leq u_2$$

- Zero order(C^0): Curve sections intersects at one end-point.
- First order(C^1): C^0 and curve section has same tangent at intersection; i.e 1^{st} derivative of the function at both curve sections are equal at the intersection.
- Second order(C^2): C^0 , C^1 and curves has same second order derivative at the intersection point.



Geometric continuity conditions

- Only require parametric derivatives of two sections of curve to be proportional to each other at the intersection.
- zero order geometric continuity(G^0): same as C^0
- first order geometric continuity(G^1): 1^{st} derivative of the function at both Curve sections are proportional at the intersection
- second order geometric continuity(G^2): 1^{st} and 2^{nd} derivative of the function at both Curve sections are proportional at the intersection

Spline Specifications

In computer graphics, the term spline curve refers to any composite curve formed with polynomial sections satisfying specified continuity conditions at the boundary of the pieces. A spline

surface can be described with two sets of orthogonal spline curves. There are several different kinds of spline specifications that are used in graphics applications. Each individual specification simply refers to a particular type of polynomial with certain specified boundary conditions

There are 3 equivalent methods for specifying a spline representation.

- State the set of boundary conditions that are imposed on the spline.
- State the matrix that characterizes the spline
- State the set of blending functions

As an example : consider the following parametric cubic polynomial representation for x coordinate:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \quad 0 \leq u \leq 1$$

1. Boundary conditions:

- Set the 4 boundary conditions as $x(0), x(1)$ and 1st derivatives $x'(0), x'(1)$
- Using these 4 boundary conditions, find the four coefficients a_x, b_x, c_x and d_x

2. Basis Matrix:

$$x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix} = \mathbf{U} \cdot \mathbf{C}$$

Where U is the row matrix of powers of parameter u and C is the coefficient column matrix.

- To calculate Coefficient column matrix, C

$$\mathbf{C} = \mathbf{M}_{\text{spline}} \cdot \mathbf{M}_{\text{geom}}$$

where \mathbf{M}_{geom} is the column matrix containing geometric constraints and $\mathbf{M}_{\text{spline}}$ is the matrix that transforms the geometric constraints to polynomial coefficients.

$$x(u) = \mathbf{U} \cdot \mathbf{C}$$

$$x(u) = \mathbf{U} \cdot (\mathbf{M}_{\text{spline}} \cdot \mathbf{M}_{\text{geom}})$$

Here $\mathbf{M}_{\text{spline}}$ is called as basis matrix.

3. Blending functions :

$$x(u) = \sum_{k=0}^3 g_k \cdot BF_k(u)$$

Where g_k are the constraint parameters such as control point coordinates & slope of the curve at the control points and BF_k termed as polynomial blending functions.

Cubic interpolation splines

This class of splines is most often used to set up paths for object motions or to provide a representation for an existing object or drawing, but interpolation splines are also used sometimes to design object shapes. Cubic polynomials offer a reasonable compromise between flexibility and speed of computation. Compared to higher-order polynomials, cubic splines require less calculations and memory and they are more stable. Compared to lower-order polynomials, cubic splines are more flexible for modeling arbitrary curve shapes.

Cubic interpolation splines are obtained by fitting the input points with a piecewise cubic polynomial curve that passes through every given control points.

Suppose we have $n+1$ control points specified with coordinates:

$$P_k = (x_k, y_k, z_k) \quad k=0,1,2,\dots,n$$



Parametric cubic polynomial that is to be fitted between each pair of control points is described with the equations:

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$

$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$$

$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z \quad (0 \leq u \leq 1)$$

For each of these three equations, we need to determine the value for a, b, c, d in the polynomial representation for each of the n curve sections between $n+1$ control points.

Natural cubic splines

One of the first spline curves to be developed for graphics applications is the natural cubic spline. We formulate a natural cubic spline by requiring that 2 adjacent curve sections have the same first & second parametric derivatives at their common boundary. i.e. natural cubic splines have C^2 continuity.

For $n+1$ control points to fit, we have n curve sections with a total of $4n$ polynomial coefficients to be determined.

- At each $n-1$ interior control points, we have the boundary conditions.
 - 2 curve sections on either side of control point must have the same 1st & 2nd derivative at that control point; Also curve must pass through the control point.
- This gives $4 \cdot (n-1)$ i.e. $4n-4$ equations to be satisfied by the $4n$ polynomial coefficients.
- We get additional equations from control points, both P_0 and P_n . Now we have $4n-2$ equations to solve.
- To determine the coefficient values, we need 2 more equations.
 - Method 1: set the second derivatives at P_0 and P_n to 0.
 - Method 2: add 2 extra “dummy” control points, one at each end. i.e. we add control points P_{-1} and P_{n+1} .

Thus a total of $4n$ equations obtained to solve the coefficient values.

Disadvantage:

- if the position of any one control points is altered, the entire curve is affected. i.e. natural cubic splines allow for no “local control”.
- Cannot restructure part of the curve without specifying an entirely new set of control points.

Hermite splines

Hermite spline (named after the French mathematician Charles Hermite) is an interpolating piecewise cubic polynomial with a specified tangent at each control point. Unlike the natural cubic splines, Hermite splines can be adjusted locally because each curve section is only dependent on its endpoint constraints.

If $p(u)$ represents a parametric cubic function for the curve section between control points P_k and P_{k+1} , boundary conditions that define hermite curve section are:

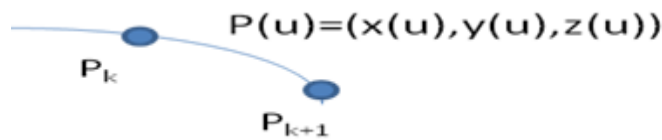
$$P(0) = P_k$$

$$P(1) = P_{k+1}$$

$$P'(0) = DP_k$$

$$P'(1) = DP_{k+1}$$

Where DP_k & DP_{k+1} are Parametric derivatives at control points P_k & P_{k+1}



- Vector equivalent form of Hermite spline can be written as:

$$P(u) = a u^3 + b u^2 + c u + d \quad (0 \leq u \leq 1)$$

$$\text{i.e.} \quad x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$

$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$$

$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

- Matrix equivalent form of Hermite spline can be written as:

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

- Derivative matrix form of Hermite spline can be written as:

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

- Substituting end point values 0 & 1 for u in the previous equations; boundary conditions of hermite curve represented as:

$$\begin{bmatrix} \mathbf{P}_k \\ \mathbf{P}_{k+1} \\ \mathbf{DP}_k \\ \mathbf{DP}_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

Therefore

$$\begin{aligned}
 \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix} \\
 &= \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix} \\
 &= \mathbf{M}_H \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix}
 \end{aligned}$$

Where \mathbf{M}_H is the hermite matrix; inverse of boundary constraint matrix.

Thus we can write

$$\begin{aligned}
 P(u) &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \\
 P(u) &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix}
 \end{aligned}$$

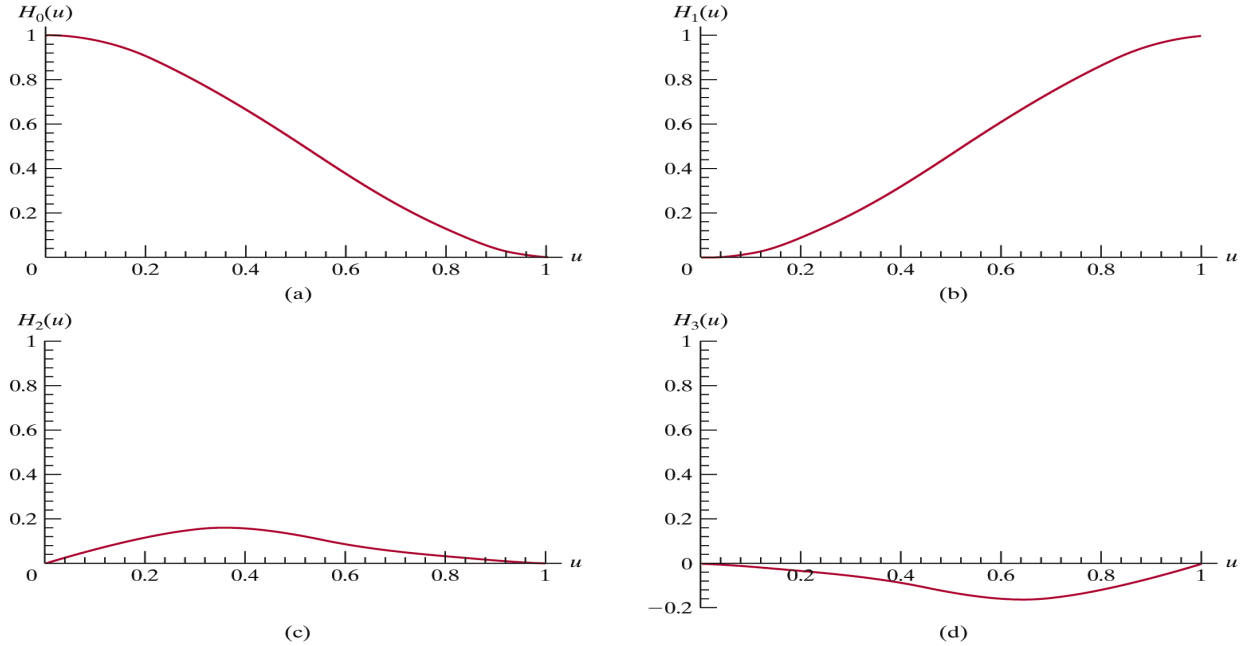
By matrix multiplication we get: Polynomials $H_k(u)$ for $k=0,1,2,3$ referred as blending functions.

$$\mathbf{P}(u) = \mathbf{p}_k (2u^3 - 3u^2 + 1) + \mathbf{p}_{k+1} (-2u^3 + 3u^2) + \mathbf{Dp}_k (u^3 - 2u^2 + u) + \mathbf{Dp}_{k+1} (u^3 - u^2)$$

$$\mathbf{P}(u) = \mathbf{p}_k H_0(u) + \mathbf{p}_{k+1} H_1(u) + \mathbf{Dp}_k H_2(u) + \mathbf{Dp}_{k+1} H_3(u)$$

The polynomials $H_k(u)$ for $k=0,1,2,3$ are referred to as blending functions because they blend the boundary constraint values (endpoint coordinates and slopes) to obtain each coordinate position along the curve.

Figure shows the shape of the four Hermite blending functions.



Cardinal splines and Kochanek-Bartels splines are variations on the Hermite splines that do not require input values for the curve derivatives at the control points. Procedures for these splines compute Parametric derivatives from the coordinate positions of the control points.

Bezier curves & surfaces

This spline approximation method was developed by the French engineer Pierre Bezier for use in the design of Renault automobile bodies. Bezier splines have a number of properties that make them highly useful and convenient for curve and surface design. They are also easy to implement. For these reasons, Bezier splines are widely available in various CAD systems

In general, a Bezier curve section can be fitted to any number of control points. The number of control points to be approximated and their relative position determine the degree of the Bezier polynomial. As with the interpolation splines, a Bezier curve can be specified with boundary conditions, with a characterizing matrix, or with blending functions. For general Bezier curves, the blending-function specification is the most convenient.

Suppose we have $n+1$ control points specified with coordinates: $P_k = (x_k, y_k, z_k)$, where $k=0,1,2,\dots,n$. $P(u)$, position vector describes path of bezier polynomial function between P_0 and P_n .

$$\mathbf{P}(u) = \sum_{k=0}^n \mathbf{p}_k \text{BEZ}_{k,n}(u), \quad 0 \leq u \leq 1$$

Where BEZ denotes the blending function.

- Bezier blending functions $BEZ_{k,n}(u)$ are **Bernstein polynomials**:

$$BEZ_{k,n}(u) = C(n,k)u^k(1-u)^{n-k}, \text{ where } C(n,k) = \frac{n!}{k!(n-k)!}$$

Where $C(n,k)$ is known as binomial coefficient.

- Defining blending function using recursion:
 $BEZ_{k,n}(u) = (1-u)BEZ_{k,n-1}(u) + u \cdot BEZ_{k-1,n-1}(u)$, where $n > k \geq 1$
 $BEZ_{k,k} = u^k$ and $BEZ_{0,k} = (1-u)^k$
- Vector equation:

$$\mathbf{P}(u) = \sum_{k=0}^n \mathbf{p}_k BEZ_{k,n}(u),$$

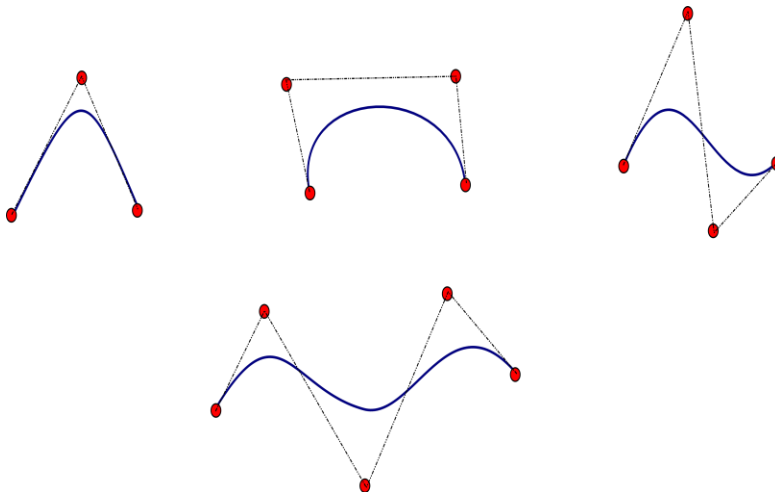
- Represents 3 parametric equations

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

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

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

- Polynomial degree of a Bézier curve is one less than the number of control points.
 3 points : parabola
 4 points : cubic curve
 5 points : fourth order curve



- Bezier curve generated using
 - 3 collinear control points is a straight line segment
 - All points at same coordinate position produce a single point.
- Advantages:
 - Easy to implement
 - Reasonably powerful in curve design

- Efficient method as it uses recursion
- Applications
 - CAD
 - Graphics package
 - Drawing & painting package
- Properties of bezier curves
 1. Bezier curve always passes through first and last control point.
 2. Boundary conditions :

$$P(0)=p_0$$

$$P(1)= p_n$$

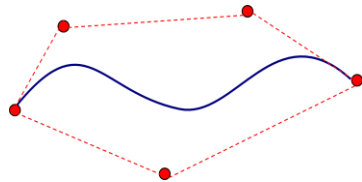
3. parametric first derivative:

$$P'(0)=-np_0+ np_1$$

$$P'(1)=-np_{n-1}+ np_n$$
4. Parametric second derivative:

$$P''(0)=-n(n-1)[(p_2- p_1)-(p_1- p_0)]$$

$$P''(1)=-n(n-1)[(p_{n-2}- p_{n-1}) -(p_{n-1}- p_n)]$$
5. Bezier curve lies with in convex hull



6. Bezier blending functions are all positive and their sum is always 1

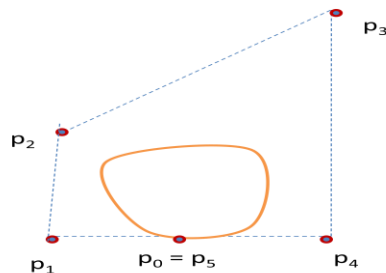
i.e

$$\sum_{k=0}^n \text{BEZ}_{k,n}(u) = 1$$

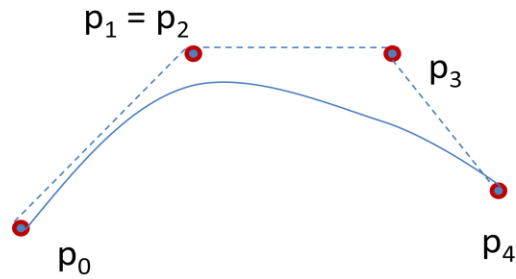
7. Tangent to the curve at end point is along the line joining that end point and adjacent control point.

- Design techniques using bezier curves

- Closed bezier curves: First and last point at same position



- Specifying multiple control points at a single position gives more weight to that position,i.e curve will be pulled more towards that point



Cubic bezier curves:

- Generated using 4 control points.
- Blending functions:

$$\text{BEZ}_{k,n}(u) = \left[\frac{n!}{k!(n-k)!} \right] u^k (1-u)^{n-k}$$

$$\text{BEZ}_{0,3} = (1-u)^3 \quad \text{BEZ}_{1,3} = 3u(1-u)^2$$

$$\text{BEZ}_{2,3} = 3u^2(1-u) \quad \text{BEZ}_{3,3} = u^3$$

- Matrix form:

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \mathbf{M}_{\text{Bez}} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

- Bezier matrix:

$$\mathbf{M}_{\text{Bez}} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Bezier surfaces

- Set of bezier curves using a mesh of control points are used to design a bezier surface.

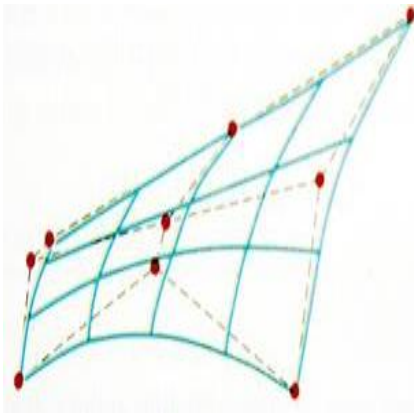
$$\mathbf{P}(u, v) = \sum_{j=0}^m \sum_{k=0}^n \mathbf{p}_{j,k} \text{BEZ}_{j,m}(v) \text{BEZ}_{k,n}(u)$$

$$0 \leq v \leq 1 \text{ \& } 0 \leq u \leq 1$$

Where $\mathbf{p}_{j,k}$ specify the location of $(m+1)$ by $(n+1)$ control points.

- Control points are connected using dashed lines and solid lines show curves of constant u and v

- Curve of constant u is plotted by varying v over the interval from 0 to 1 with u fixed at one of its values in the interval
- Similarly curves of constant v is also plotted



Important questions

1. Explain Sutherland Hodgeman polygon clipping algorithm
2. Explain Weiler Atherton Polygon clipping algorithm
3. Explain a clipping algorithm that works with concave polygons.
4. Write a note on 3D viewing devices
5. Explain the steps of transforming a 3D object about an axis that is parallel to one of the coordinate axes
6. Describe quadratic surface .
7. Explain 3D display methods
8. Explain Bezier curve representation and their properties
9. Write a note on 3D viewing.
10. Explain how fractals are classified.
11. List the transformations required for 3D scaling with respect to a selected fixed point
12. Explain the representation of polygon surfaces using polygon tables and meshes
13. Explain the rotation of a 3D object about an axis that is not parallel to one of the coordinate axes
14. What is a polygon surface.
15. Explain spline curves
16. Explain cubic splines
17. Explain 3D transforms and derive their equation and matrix
18. How can u rotate an object with respect to Z axis
19. Explain the continuity conditions to be satisfied for a spline representation.
20. What are the advantages and disadvantages of B-spline curve over Bezier curves
 - Advantages
 1. Degree of B-spline polynomial can be set independently of the no: of control points
 2. B-splines allow local control over the shape of a spline curve or surface
 - Disadvantages
 - B-splines are more complex than Bezier splines.

21. Compare the features of Bezier curves and B-spline curve

| BEZIER CURVE | B SPLINES |
|---|--|
| <ul style="list-style-type: none"> ❖ Approximation spline ❖ The degree polynomial is one less than the no of control points ❖ Has local control ❖ General representation $p(u) = \sum_{k=1}^n (p_k) bez_k(u), d^u \{0 \leq u < 1\}$ ❖ $Bez_{k,n}(u)$ is the blending function used. is also known as Bernstein polynomial | <ul style="list-style-type: none"> ❖ Approximation spline ❖ The degree is independent of the no of control points ❖ Has local control over the surface of the spline ❖ General representation $p(u) = \sum_{k=1}^n (p_k) b^k, d^u \{2 \leq d < n+1\}$ ❖ $B_{k,d}(u)$ is the blending function used |

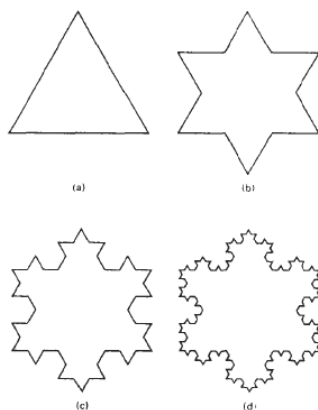
22. Explain how fractals are classified

fractals : Natural objects can be realistically described with fractal-geometry methods, where procedures rather than equations are used to model objects. A fractal object has two basic characteristics: infinite detail at every point and a certain self-similarity between the object parts and the overall features of the object .

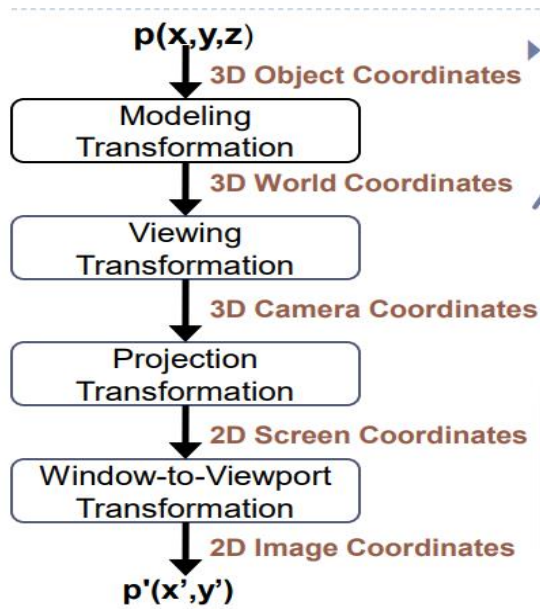
Types of fractals

1. Self-similar fractals have parts that are scaled-down versions of the entire object.
2. Self-affine fractals have parts that are formed with different scaling parameters, S_x, S_y, S_z , in different coordinate directions.
3. Invariant fractal sets are formed with nonlinear transformations .

To geometrically construct a deterministic (nonrandom) self-similar fractal, we start with a given geometric shape, called the initiator. Subparts of the initiator are then replaced with a pattern, called the generator.



23. Write the steps in 3D viewing.



- ▶ **Transform** in 3D word coordinate system
- ▶ Illuminate according to lighting and reflectance
- ▶ **Transform** into 3D camera coordinate system
- ▶ **Transform** into 2D screen coordinate system
- ▶ Clip primitives outside camera's view
- ▶ Draw Pixels (also texturing, hidden surface, ...)