

SCS1302 COMPUTER GRAPHICS & MULTIMEDIA SYSTEMS

(Common to CSE & IT)

UNIT III 3D CONCEPTS AND CURVES

3D object representation methods - B-REP, sweep representations, Three dimensional transformations. Curve generation - cubic splines, Beziers, blending of curves- other interpolation techniques, Displaying Curves and Surfaces, Shape description requirement, parametric function. Three dimensional concepts – Introduction - Fractals and self similarity- Successive refinement of curves, Koch curve and peano curves.

3D object representation methods - B-REP, sweep representations

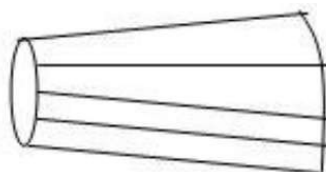
Polygon Surfaces

Objects are represented as a collection of surfaces. 3D object representation is divided into two categories.

- **Boundary Representations (B-reps)** – It describes a 3D object as a set of surfaces that separates the object interior from the environment.
- **Space-partitioning representations** – It is used to describe interior properties, by partitioning the spatial region containing an object into a set of small, non-overlapping, contiguous solids (usually cubes).

The most commonly used boundary representation for a 3D graphics object is a set of surface polygons that enclose the object interior. Many graphics system use this method. Set of polygons are stored for object description. This simplifies and speeds up the surface rendering and display of object since all surfaces can be described with linear equations.

The polygon surfaces are common in design and solid-modeling applications, since their **wireframe display** can be done quickly to give general indication of surface structure. Then realistic scenes are produced by interpolating shading patterns across polygon surface to illuminate.

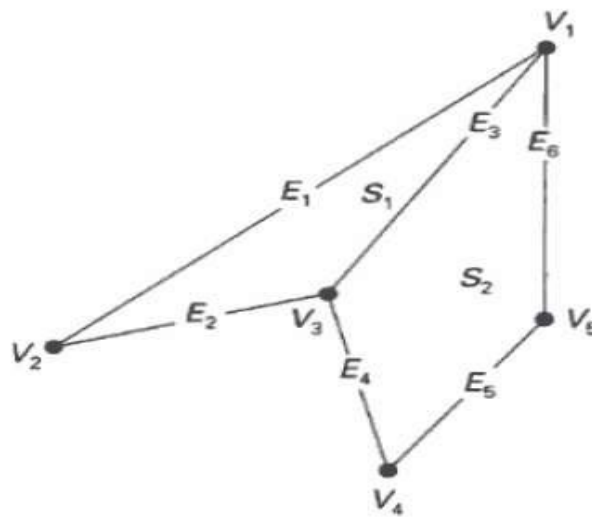


A 3D object represented by polygons

Polygon Tables

In this method, the surface is specified by the set of vertex coordinates and associated attributes. As shown in the following figure, there are five vertices, from v_1 to v_5 .

- Each vertex stores x, y, and z coordinate information which is represented in the table as $v_1: x_1, y_1, z_1$.
- The Edge table is used to store the edge information of polygon. In the following figure, edge E_1 lies between vertex v_1 and v_2 which is represented in the table as $E_1: v_1, v_2$.
- Polygon surface table stores the number of surfaces present in the polygon. From the following figure, surface S_1 is covered by edges E_1, E_2 and E_3 which can be represented in the polygon surface table as $S_1: E_1, E_2, \text{ and } E_3$.



VERTEX TABLE	
$V_1:$	x_1, y_1, z_1
$V_2:$	x_2, y_2, z_2
$V_3:$	x_3, y_3, z_3
$V_4:$	x_4, y_4, z_4
$V_5:$	x_5, y_5, z_5

EDGE TABLE	
$E_1:$	V_1, V_2
$E_2:$	V_2, V_3
$E_3:$	V_3, V_1
$E_4:$	V_3, V_4
$E_5:$	V_4, V_5
$E_6:$	V_5, V_1

POLYGON-SURFACE TABLE	
$S_1:$	E_1, E_2, E_3
$S_2:$	E_3, E_4, E_5, E_6

Plane Equations

The equation for plane surface can be expressed as:

$$Ax + By + Cz + D = 0$$

Where (x, y, z) is any point on the plane, and the coefficients A, B, C , and D are constants describing the spatial properties of the plane. We can obtain the values of A, B, C , and D by

solving a set of three plane equations using the coordinate values for three non collinear points in the plane. Let us assume that three vertices of the plane are (x_1, y_1, z_1) , (x_2, y_2, z_2) and (x_3, y_3, z_3) .

Let us solve the following simultaneous equations for ratios A/D , B/D , and C/D . You get the values of A , B , C , and D .

$$(A/D) x_1 + (B/D) y_1 + (C/D) z_1 = -1$$

$$(A/D) x_2 + (B/D) y_2 + (C/D) z_2 = -1$$

$$(A/D) x_3 + (B/D) y_3 + (C/D) z_3 = -1$$

To obtain the above equations in determinant form, apply Cramer's rule to the above equations.

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

$$D = - \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$

For any point (x, y, z) with parameters A , B , C , and D , we can say that –

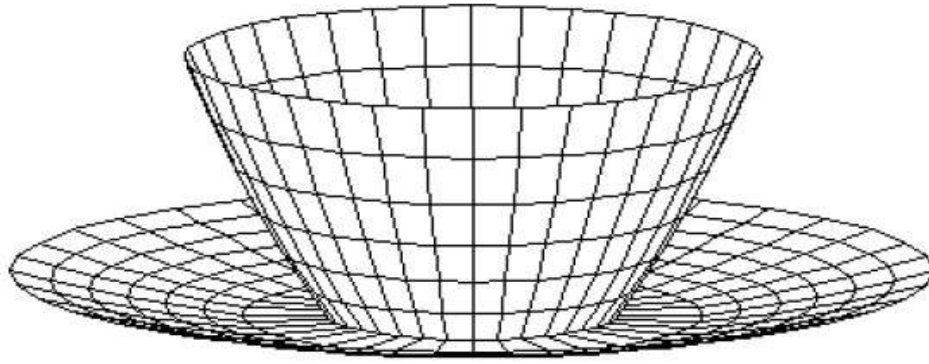
- $Ax + By + Cz + D \neq 0$ means the point is not on the plane.
- $Ax + By + Cz + D < 0$ means the point is inside the surface.
- $Ax + By + Cz + D > 0$ means the point is outside the surface.

Polygon Meshes

3D surfaces and solids can be approximated by a set of polygonal and line elements. Such surfaces are called **polygonal meshes**. In polygon mesh, each edge is shared by at most two polygons. The set of polygons or faces, together form the “skin” of the object.

This method can be used to represent a broad class of solids/surfaces in graphics. A polygonal mesh can be rendered using hidden surface removal algorithms. The polygon mesh can be represented by three ways –

- Explicit representation
- Pointers to a vertex list
- Pointers to an edge list



Advantages

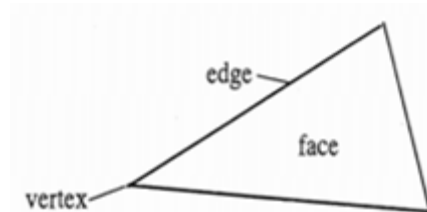
- It can be used to model almost any object.
- They are easy to represent as a collection of vertices.
- They are easy to transform.
- They are easy to draw on computer screen.

Disadvantages

- Curved surfaces can only be approximately described.
- It is difficult to simulate some type of objects like hair or liquid.

B-Rep:

1. B-Rep stands for Boundary Representation.
2. It is an extension to the wireframe model.
3. B-Rep describes the solid in terms of its surface boundaries: Vertices, edges and faces as shown below.



1. It is a method for representing shapes using the limits.
2. A solid is represented as a collection of connected surface elements, the boundary between solid and non-solid.
3. There are 2 types of information in a B – rep topological and geometric.
4. Topological information provides the relationships among vertices, edges and faces similar to that used in a wireframe model.
5. In addition to connectivity, topological information also includes orientation of edges and faces.
6. Geometric information is usually equations of the edges and faces.

7. The B-rep of 2 manifolds that have faces with holes satisfies the generalized Euler's formula:

$$V - E + F - H = 2(C - G)$$

Where, V = Number of vertices.

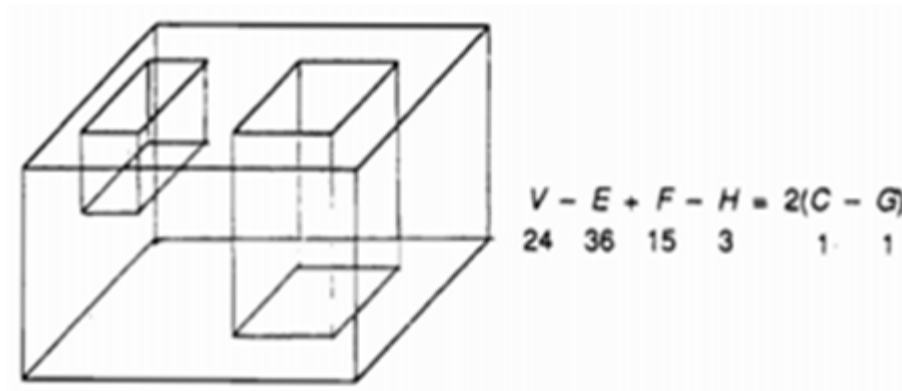
E = Number of edges.

F = Number of faces.

H = Number of holes in the faces.

C is the number of separate components (parts).

G is the genus (for a torus $G = 1$)



Sweep Representations:

Sweep representations are used to construct 3D object from 2D shape that have some kind of symmetry.

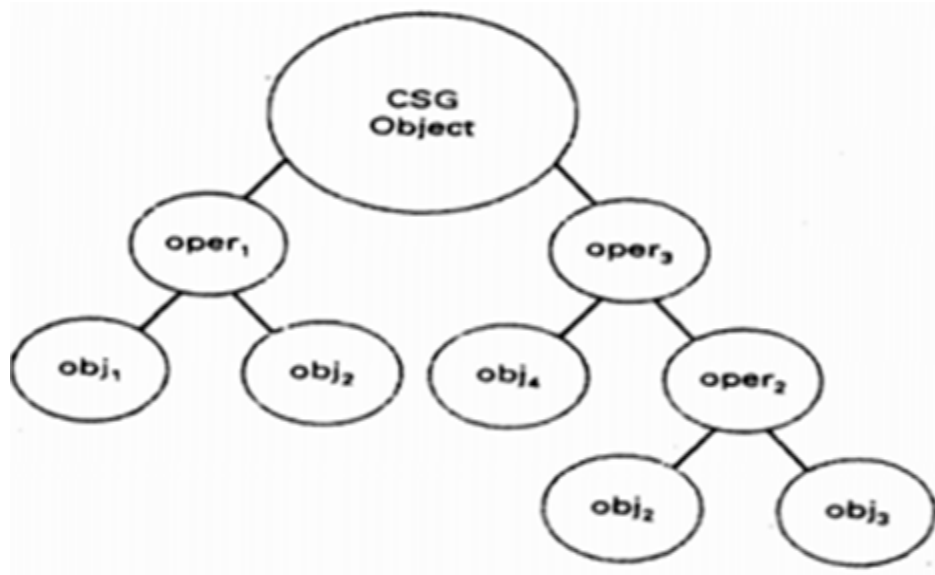
For example, a prism can be generated using a translational sweep and rotational sweeps can be used to create curved surfaces like an ellipsoid or a torus.

In both cases, you start with a cross-section and generate more vertices by applying the appropriate transformation.

More complex objects can be formed by using more complex transformations. Also, we can define a path for a sweep that allows you to create more interesting shapes.

CSG:

1. CSG stands for Constructive Solid Geometry.
2. It is based on set of 3D solid primitives and regularized set theoretic operations.
3. Traditional primitives are: Block, cones, sphere, cylinder and torus.
4. Operations: union, intersection, difference + translation and rotation.
5. A complex solid is represented using with a binary tree usually called as CSG tree.
6. CSG tree is shown below.



1. Ray casting is a method used for determining boundaries of the resulting object if you start with a boundary representation.
2. Octree representations are designed to make this process easier.

Example: Consider the below figure, say Figure 1.

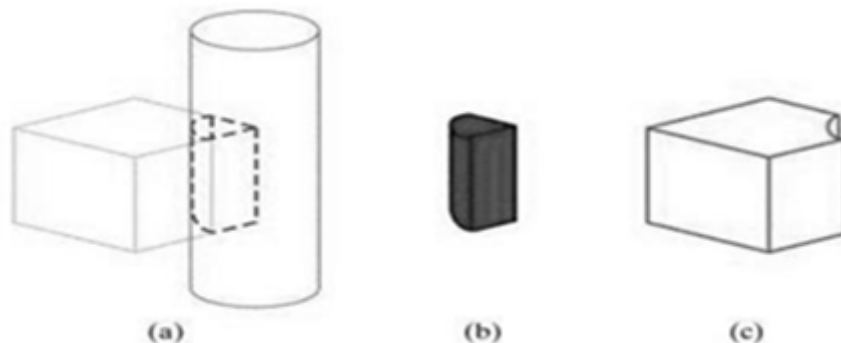


Fig. 1

Figure 1 (b) is the intersection of original solid.

Figure 1 (c) is the difference between 2 original solid.

Sweep representations are used to construct 3D object from 2D shape that have some kind of symmetry.

For example, a prism can be generated using a translational sweep and rotational sweeps can be used to create curved surfaces like an ellipsoid or a torus.

In both cases, you start with a cross-section and generate more vertices by applying the appropriate transformation.

More complex objects can be formed by using more complex transformations. Also, we can define a path for a sweep that allows you to create more interesting shapes.

Translational sweep:

- i. Define a shape as a polygon vertex table as shown in figure 2 (a).
- ii. Define a sweep path as a sequence of translation vectors figure 2 (b).
- iii. Translate the shape; continue building a vertex table figure 2 (c).
- iv. Define a surface table figure 2 (d).

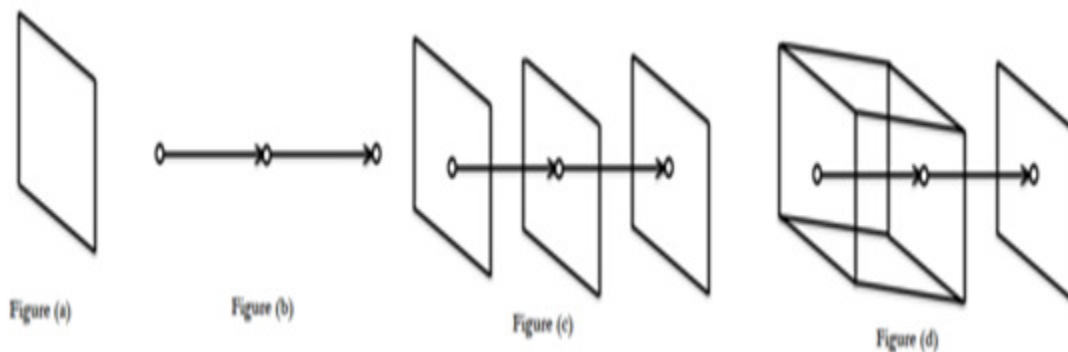


Fig. 2

Rotational sweep:

- i. Define a shape as a polygon vertex table as shown in figure 3 (a).
- ii. Define a sweep path as a sequence of rotations.
- iii. Rotate the shape; continue building a vertex table as shown in figure 3 (b).
- iv. Define a surface table as shown in figure 3 (c).

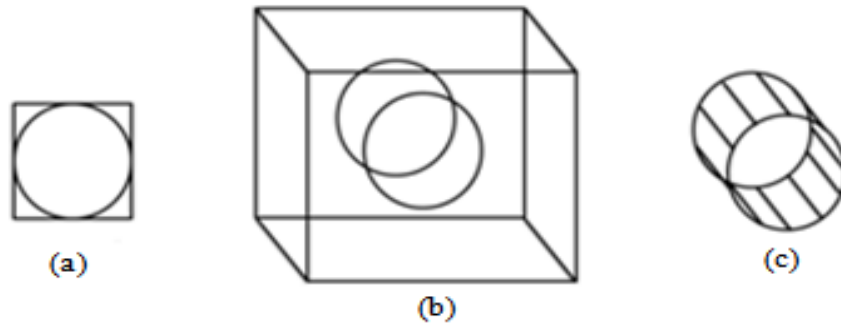


Fig. 3

In computer graphics, we often need to draw different types of objects onto the screen. Objects are not flat all the time and we need to draw curves many times to draw an object.

Three dimensional transformations

Basic Transformations:

1. Translation
2. Rotation
3. Scaling

Other Transformations:

1. Reflection
2. Shearing

Translation

A translation in space is described by t_x , t_y and t_z . It is easy to see that this matrix realizes the equations:

$$\begin{aligned}x_2 &= x_1 + t_x \\ y_2 &= y_1 + t_y \\ z_2 &= z_1 + t_z\end{aligned}$$

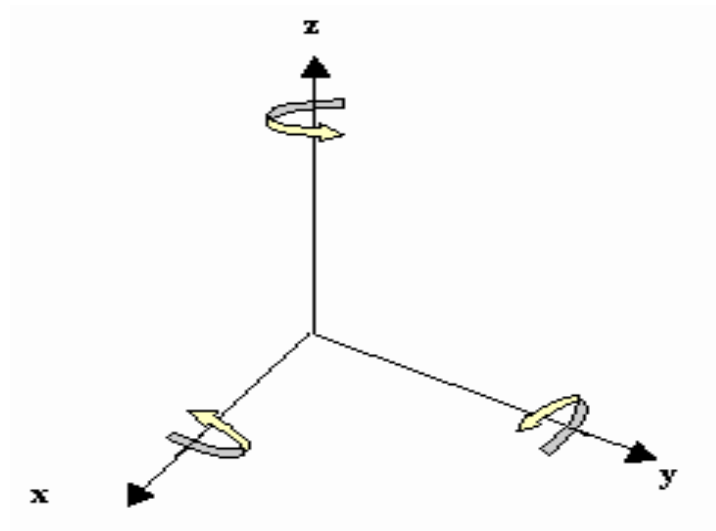
$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

Translation Matrix

Rotation

3D rotation is not same as 2D rotation. In 3D rotation, we have to specify the angle of rotation along with the axis of rotation. We can perform 3D rotation about X, Y, and Z axes. They are represented in the matrix form as below :

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



Scaling

We can change the size of an object using scaling transformation. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result.

In 3D scaling operation, three coordinates are used. Let us assume that the original coordinates are (X, Y, Z), scaling factors are (S_x, S_y, S_z) respectively, and the produced coordinates are (X', Y', Z'). This can be mathematically represented as shown below :

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

$$P' = P \cdot S$$

$$\begin{aligned} [X' \ Y' \ Z' \ 1] &= [X \ Y \ Z \ 1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= [X \cdot S_x \ Y \cdot S_y \ Z \cdot S_z \ 1] \end{aligned}$$

Reflection

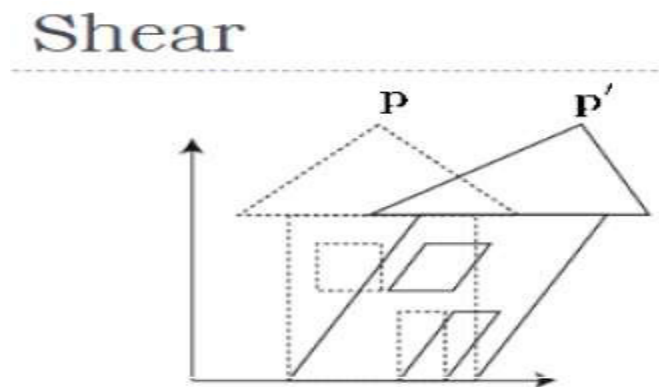
A transformation that gives the mirror image of the object.

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

We can mirror the different planes by using scaling factor -1 on the axis that is placed normally on the plane. Notice the matrix to the left. It mirrors around the xy-plane, and changes the coordinates from a right hand system to a left hand system.

Shear

A transformation that slants the shape of an object is called the shear transformation. Like in 2D shear, we can shear an object along the X-axis, Y-axis, or Z-axis in 3D.



As shown in the above figure, there is a coordinate P. You can shear it to get a new coordinate P', which can be represented in 3D matrix form as below

$$Sh = \begin{bmatrix} 1 & sh_x^y & sh_x^z & 0 \\ sh_y^x & 1 & sh_y^z & 0 \\ sh_z^x & sh_z^y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P' = P \cdot Sh$$

$$X' = X + Sh_x^y Y + Sh_x^z Z$$

$$Y' = Sh_y^x X + Y + sh_y^z Z$$

$$Z' = Sh_z^x X + Sh_z^y Y + Z$$

Curve generation - cubic splines, Beziers, blending of curves- other interpolation techniques

Types of Curves

A curve is an infinitely large set of points. Each point has two neighbors except endpoints. Curves can be broadly classified into three categories – **explicit**, **implicit**, and **parametric curves**.

Implicit Curves

Implicit curve representations define the set of points on a curve by employing a procedure that can test to see if a point is on the curve. Usually, an implicit curve is defined by an implicit function of the form –

$$f(x, y) = 0$$

It can represent multivalued curves (multiple y values for an x value). A common example is the circle, whose implicit representation is

$$x^2 + y^2 - R^2 = 0$$

Explicit Curves

A mathematical function $y = f(x)$ can be plotted as a curve. Such a function is the explicit representation of the curve. The explicit representation is not general, since it cannot represent vertical lines and is also single-valued. For each value of x , only a single value of y is normally computed by the function.

Parametric Curves

Curves having parametric form are called parametric curves. The explicit and implicit curve representations can be used only when the function is known. In practice the parametric curves are used. A two-dimensional parametric curve has the following form –

$$P(t) = f(t), g(t) \text{ or } P(t) = x(t), y(t)$$

The functions f and g become the (x, y) coordinates of any point on the curve, and the points are obtained when the parameter t is varied over a certain interval $[a, b]$, normally $[0, 1]$.

Bezier Curves

Bezier curve is discovered by the French engineer **Pierre Bézier**. These curves can be generated under the control of other points. Approximate tangents by using control points are used to generate curve. The Bezier curve can be represented mathematically as –

$$\sum_{k=0}^n P_i B_i^n(t)$$

Where p_i is the set of points and

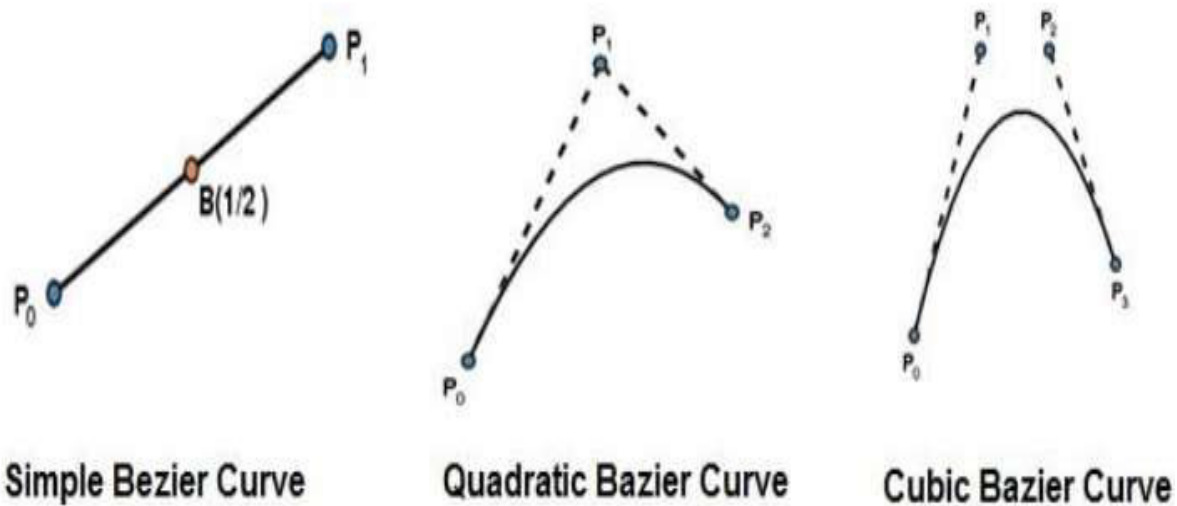
$$B_i^n(t)$$

represents the Bernstein polynomials which are given by –

$$B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i$$

Where **n** is the polynomial degree, **i** is the index, and **t** is the variable.

The simplest Bézier curve is the straight line from the point P_0 to P_1 . A quadratic Bezier curve is determined by three control points. A cubic Bezier curve is determined by four control points.



Properties of Bezier Curves

Bezier curves have the following properties –

- They generally follow the shape of the control polygon, which consists of the segments joining the control points.
- They always pass through the first and last control points.
- They are contained in the convex hull of their defining control points.
- The degree of the polynomial defining the curve segment is one less than the number of defining polygon points. Therefore, for 4 control points, the degree of the polynomial is 3, i.e. cubic polynomial.
- A Bezier curve generally follows the shape of the defining polygon.
- The direction of the tangent vector at the end points is same as that of the vector determined by first and last segments.
- The convex hull property for a Bezier curve ensures that the polynomial smoothly follows the control points.
- No straight line intersects a Bezier curve more times than it intersects its control polygon.
- They are invariant under an affine transformation.
- Bezier curves exhibit global control means moving a control point alters the shape of the whole curve.
- A given Bezier curve can be subdivided at a point $t=t_0$ into two Bezier segments which join together at the point corresponding to the parameter value $t=t_0$.

B-Spline Curves

The Bezier-curve produced by the Bernstein basis function has limited flexibility.

- First, the number of specified polygon vertices fixes the order of the resulting polynomial which defines the curve.

- The second limiting characteristic is that the value of the blending function is nonzero for all parameter values over the entire curve.

The B-spline basis contains the Bernstein basis as the special case. The B-spline basis is non-global. A B-spline curve is defined as a linear combination of control points P_i and B-spline basis function $N_{i,k}(t)$ given by

$$C(t) = \sum_{i=0}^n P_i N_{i,k}(t), \quad n \geq k-1, \quad t \in [t_{k-1}, t_n + 1]$$

Where,

- $\{p_i: i=0, 1, 2, \dots, n\}$ are the control points
- k is the order of the polynomial segments of the B-spline curve. Order k means that the curve is made up of piecewise polynomial segments of degree $k-1$,
- the $N_{i,k}(t)$ are the “normalized B-spline blending functions”. They are described by the order k and by a non-decreasing sequence of real numbers normally called the “knot sequence”.

$$t_i: i=0, \dots, n+K$$

The $N_{i,k}$ functions are described as follows –

$$N_{i,1}(t) = \begin{cases} 1, & \text{if } t \in [t_i, t_{i+1}) \\ 0, & \text{Otherwise} \end{cases}$$

and if $k > 1$,

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

and

$$t \in [t_{k-1}, t_{n+1})$$

Properties of B-spline Curve

B-spline curves have the following properties –

- The sum of the B-spline basis functions for any parameter value is 1.
- Each basis function is positive or zero for all parameter values.

- Each basis function has precisely one maximum value, except for $k=1$.
- The maximum order of the curve is equal to the number of vertices of defining polygon.
- The degree of B-spline polynomial is independent on the number of vertices of defining polygon.
- B-spline allows the local control over the curve surface because each vertex affects the shape of a curve only over a range of parameter values where its associated basis function is nonzero.
- The curve exhibits the variation diminishing property.
- The curve generally follows the shape of defining polygon.
- Any affine transformation can be applied to the curve by applying it to the vertices of defining polygon.
- The curve line within the convex hull of its defining polygon.

Shape description requirement, parametric function

Shape description requirements

Generally shape representations have two different uses, an analytic use and a synthetic use. Representations are used analytically to describe shapes that can be measured; just as a curve can be fitted to a set of data points, a surface can be fitted to the measured properties of some real objects. The requirements of the user and the computer combine to suggest a number of properties that our representations must have. The following are some of the important properties that are used to design the curves. The similar properties can also be used for designing the surfaces as surfaces are made up of curves.

- (a) Control Points: The shape of the curve can be controlled easily with the help of a set of control points, means the points will be marked first and curve will be drawn that intersects each of these points one by one in a particular sequence. The more number of control points makes the curve smoother. The following figure shows the curve with control points.



**Fig : Control points
(Indicated by dots) govern
the shape of the curve**

- (b) Multiple values: In general any curve is not a graph of single valued function of a coordinate, irrespective the choice of coordinate system. Generally single valued functions of a coordinate make the curves or graphs that are dependent on axis. The following figure shows multivalued curve with respect to all coordinate systems.

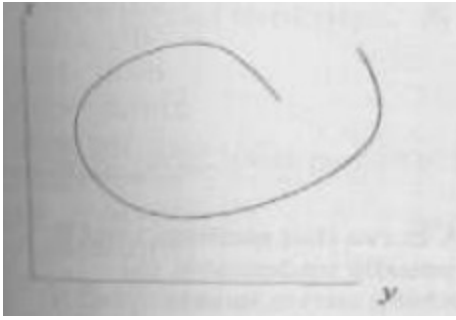


Fig: A curve can be multi-valued with respect to all coordinate systems.

- (c) Axis independence: The shape of an object should not change when the control points are measured in different coordinate systems, that means when an object is rotated to certain angle in any direction (clockwise or anti-clockwise) the shape of the curve should not be affected.
- (d) Global or local control: The control points of a curve must be controlled globally from any function of the same program or it can also be controlled locally by the particular function used to design that curve by calculating the desired control points.
- (e) Variation-diminishing property: Some of the mathematical functions may diminish the curve at particular points and in some other points it may amplify the points. This leads to certain problems for curves appearance at the time of animations, (just as a vehicle looks curved when it is taking turn). This effect must be avoided with the selection of proper mathematical equations with multiple valued functions.
- (f) Versatility: The functions that define the shape of the curve should not be limited to only few varieties of shapes, instead they must provide wide varieties for the designers to make the curves according to their interest.
- (g) Order of continuity: For any complex shapes or curves or surfaces it is essential to maintain continuity in calculating control points. When we are not maintaining the proper continuity of control points it makes a mesh while marking the curve and the complex object.

Parametric functions

The dominant form used to model curves and surfaces is the parametric or vector valued function. A point on a curve is represented as a vector: $P(u)=[x(u) \ y(u) \ z(u)]$.

For surfaces, two parametric are required: $P(u, v)=[(x(u, v) \ y(u, v) \ z(u, v))]$.

As the parametric u and v take on values in a specified range, usually 0 to 1, the parametric functions x , y and z trace out the location of the curve or surface. The parametric functions can themselves take many forms. A single curve be approximated in sever different ways as given below:

$$P(u) = [\cos u \sin u]$$

$$P(u) = [(1 - u^2)/(1 + u^2) \ 2u/(1 + u^2)]$$

$$P(u) = [u \ (1 - u^2)^{1/2}]$$

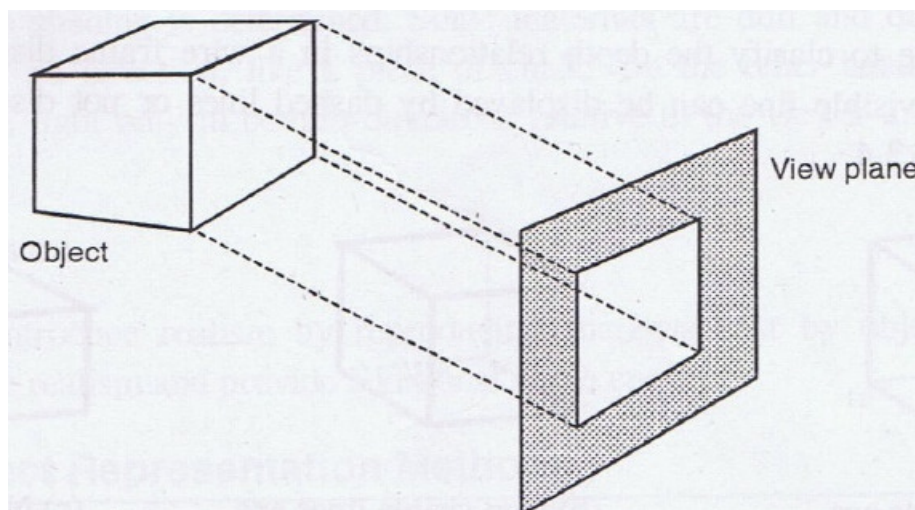
By using simple parametric functions, we cannot expect the designer to achieve a desirable curve by changing coefficients of parametric polynomial functions or of any other functional form. Instead we must find ways to determine the parametric function from the location of control points that are manipulated by the designer.

Three dimensional concepts

- Parallel Projection
- Perspective Projection
- Depth Cueing
- Visible Line and Surface Identification
- Surface Rendering
- Exploded and Cutaway Views
- Three-Dimensional and Stereoscopic Views

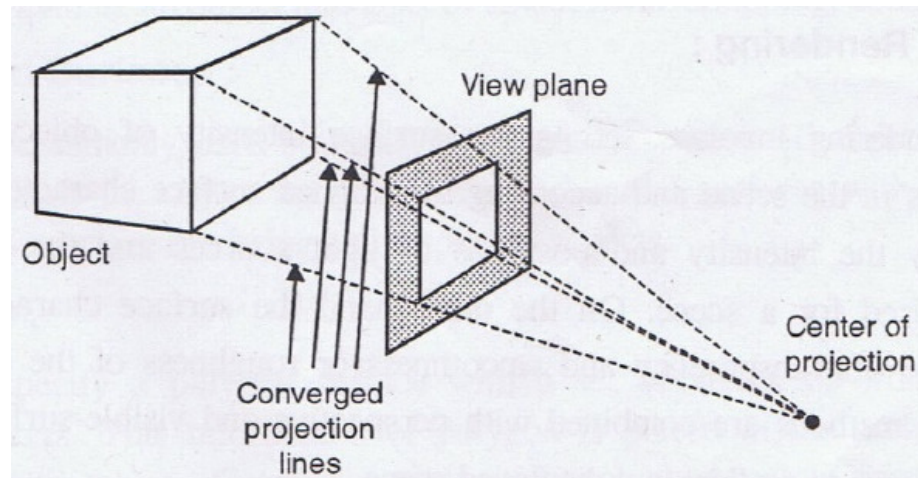
Parallel Projection

In this method a view plane is used. z co-ordinate is discarded. The 3 d view is constructed by extending lines from each vertex on the object until they intersect the view plane. Then connect the projected vertices by line segments which correspond to connections on the original object.



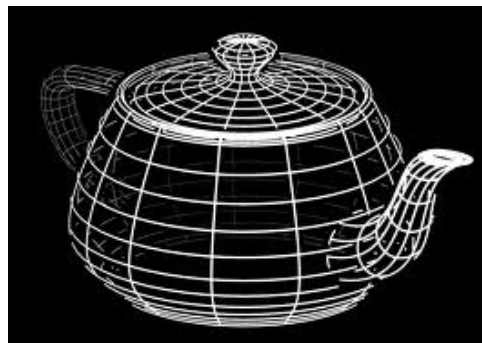
Perspective Projection

Here the lines of projection are not parallel. Instead, they all converge at a single point called the 'center of projection' or 'projection reference point'. The object positions are transformed to the view plane along these converged projection lines. In this method, Objects farther from the viewing position appear smaller.



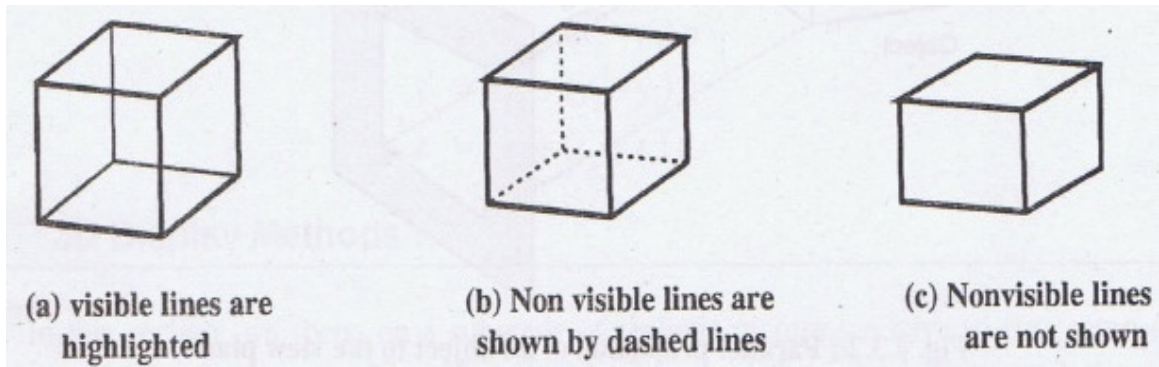
Depth Cueing

Depth information is added. The depth of an object can be represented by the intensity of the image. The parts of the objects closest to the viewing position are displayed with the highest intensities. Objects farther away are displayed with decreasing intensities.



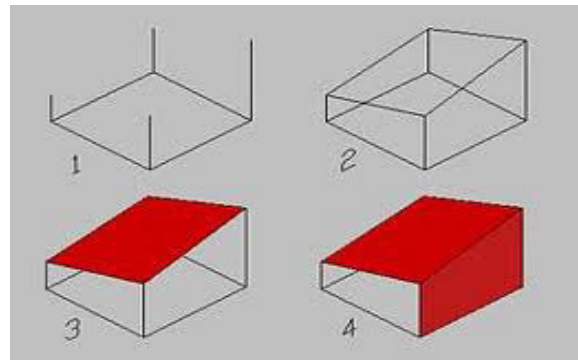
Visible Line and Surface Identification

Visible lines are displayed in different color. Invisible lines either displayed in dashed lines or not at all displayed. Removing invisible lines also removes the info about the backside of the object. Surface rendering can be applied for the visible surfaces, so that hidden surfaces will become obscured.



Surface Rendering

Surface intensity of objects will be according to the lighting conditions in the scene and according to assigned surface characteristics. This method is usually combined with the previous method to attain a degree of realism.



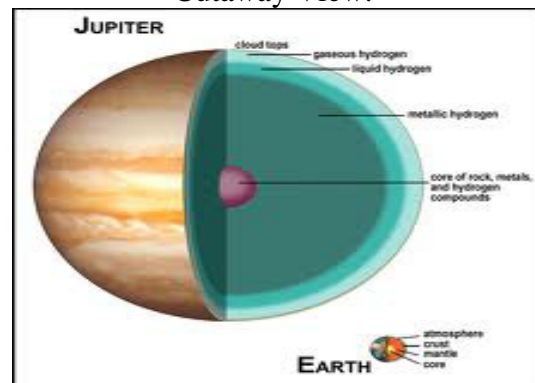
Exploded and Cutaway Views

To show the internal details, we can define the object in hierarchical structures.

Exploded view:



Cutaway View:



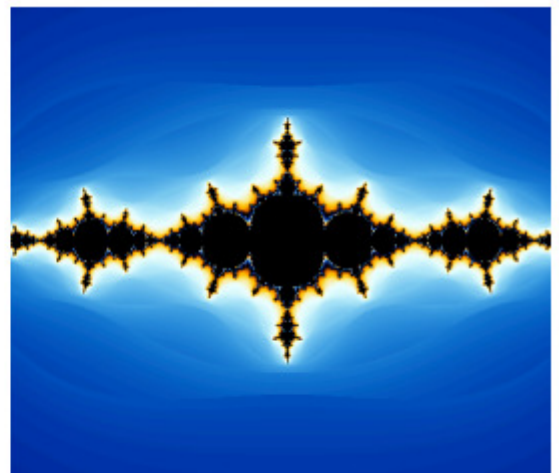
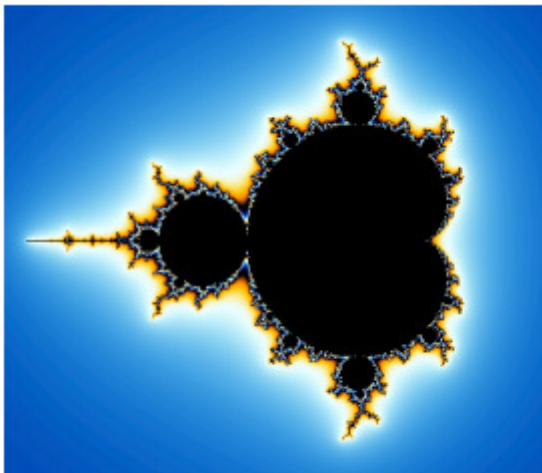
Fractals and self similarity- Successive refinement of curves, Koch curve and peano curves

Fractals

Fractals are very complex pictures generated by a computer from a single formula. They are created using iterations. This means one formula is repeated with slightly different values over and over again, taking into account the results from the previous iteration.

Fractals are used in many areas such as –

- Astronomy – For analyzing galaxies, rings of Saturn, etc.
- Biology/Chemistry – For depicting bacteria cultures, Chemical reactions, human anatomy, molecules, plants,
- Others – For depicting clouds, coastline and borderlines, data compression, diffusion, economy, fractal art, fractal music, landscapes, special effect, etc.



Three common techniques for generating fractals are:

- Iterated function systems — These have a fixed geometric replacement rule. Cantor set, Sierpinski carpet, Sierpinski gasket, Peano curve, Koch snowflake, Harter-Heighway dragon curve, T-Square, Menger sponge, are some examples of such fractals.
- Escape-time fractals — Fractals defined by a recurrence relation at each point in a space (such as the complex plane). Examples of this type are the Mandelbrot set, the Burning Ship fractal and the Lyapunov fractal.
- Random fractals, generated by stochastic rather than deterministic processes, for example, fractal landscapes, Lévy flight and the Brownian tree. The latter yields so-called mass- or dendritic fractals, for example, Diffusion Limited Aggregation or Reaction Limited Aggregation clusters.

Fractals can also be classified according to their self-similarity. There are three types of self-similarity found in fractals:

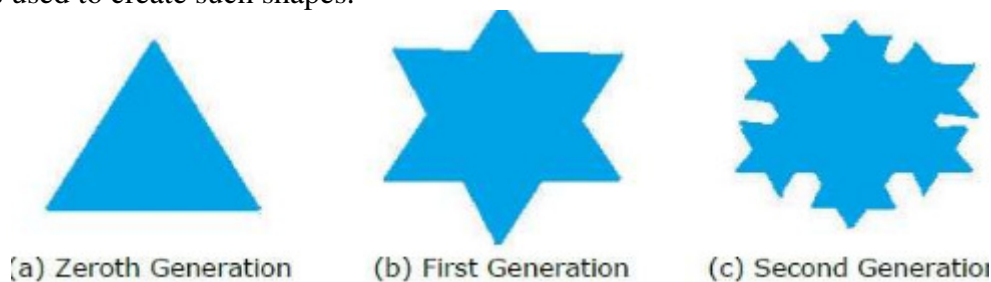
- Exact self-similarity — This is the strongest type of self-similarity; the fractal appears identical at different scales. Fractals defined by iterated function systems often display exact self-similarity.

- Quasi-self-similarity — This is a loose form of self-similarity; the fractal appears approximately (but not exactly) identical at different scales. Quasi-self-similar fractals contain small copies of the entire fractal in distorted and degenerate forms. Fractals defined by recurrence relations are usually quasi-self-similar but not exactly self-similar.
- Statistical self-similarity — This is the weakest type of self-similarity; the fractal has numerical or statistical measures which are preserved across scales. Most reasonable definitions of "fractal" trivially imply some form of statistical self-similarity. (Fractal dimension itself is a numerical measure which is preserved across scales.) Random fractals are examples of fractals which are statistically self-similar, but neither exactly nor quasi-self-similar.

Generation of Fractals

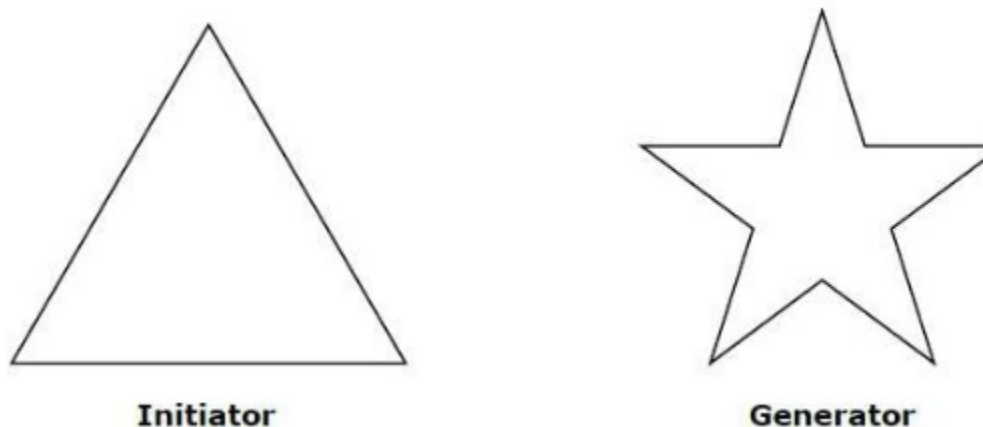
Fractals can be generated by repeating the same shape over and over again as shown in the following figure. In figure (a) shows an equilateral triangle. In figure (b), we can see that the triangle is repeated to create a star-like shape. In figure (c), we can see that the star shape in figure (b) is repeated again and again to create a new shape.

We can do unlimited number of iteration to create a desired shape. In programming terms, recursion is used to create such shapes.




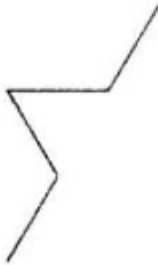

Geometric Fractals

Geometric fractals deal with shapes found in nature that have non-integer or fractal dimensions. 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.



As an example, if we use the initiator and generator shown in the above figure, we can construct good pattern by repeating it. Each straight-line segment in the initiator is replaced with four equal-length line segments at each step. The scaling factor is $1/3$, so the fractal dimension is $D = \ln 4 / \ln 3 \approx 1.2619$.

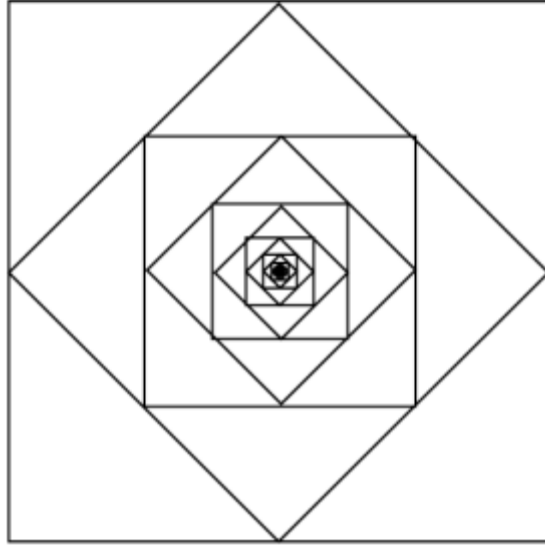
Also, the length of each line segment in the initiator increases by a factor of $4/3$ at each step, so that the length of the fractal curve tends to infinity as more detail is added to the curve as shown in the following figure.

Segment Length = 1	Segment Length = $1/3$	Segment Length = $1/9$
		
Length = 1	Length = $4/3$	Length = $16/9$

Many fractals also have a property of self-similarity – within the fractal lies another copy of the same fractal, smaller but complete. In this project we will study the simplest, and best known, fractals with this property, the strictly self-similar fractals. We will show how to describe and create these fractals, and how to measure their fractal dimension using the similarity dimension.

Strictly Self-Similar Fractals

A geometric figure is self-similar if there is a point where every neighborhood of the point contains a copy of the entire figure. For example, imagine the figure formed by inscribing a square within another square, rotated by 45° . Then inside the inner square, inscribe another square in the same manner, and so on ad infinitum. Of course, we can't really draw this figure, since it contains infinitely many nested squares, but an approximation of the result is shown below:



This figure is self-similar at the center of the squares – every ball around the center, no matter how small, will contain a complete copy of the entire figure. However, this is the only point where the figure is self-similar – at most points, the figure looks, at some magnification, like either a straight line or a corner where four lines meet, so this is not complicated enough to be a fractal. A figure is strictly self-similar if it is self-similar at every point. Equivalently, this means that the figure can be decomposed into some number of disjoint pieces, each of which is an exact copy of the entire figure. What does such a figure look like? Let's look at several classical examples. As with the inscribed squares above, these examples are really the limits of some repeated, or iterative, process, which we imagine repeating infinitely often.

Cantor Set

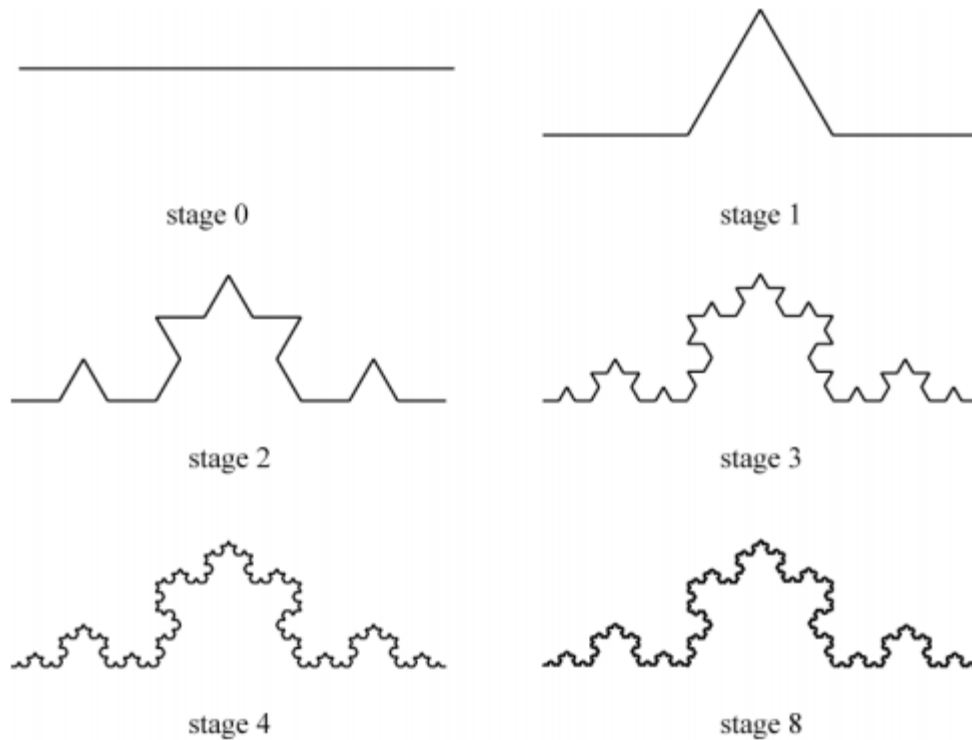
The Cantor set was first described by German mathematician Georg Cantor in 1883, and is the foundation for many important fractals. The description of the set is deceptively easy. Begin with the unit interval $[0,1]$, which is represented below as a line segment (stage 0). Delete the middle third, which is the open interval $(1/3, 2/3)$, leaving the two closed subintervals $[0, 1/3]$ and $[2/3, 1]$. Now repeat this process on each of these subintervals, leaving 4 subintervals, and continue repeating this process on each new set of smaller subintervals forever. The figure below shows the first few stages of this construction:



The Cantor set is the set of points remaining when all of these deletions of subintervals have taken place. This is more than you might expect – for example, all the endpoints of all the closed intervals at every stage of the process are in the Cantor set. In fact, there are just as many points in the Cantor set as there were in the original unit interval. Despite this, any point of the interval $[0, 1]$ that is not in the Cantor set is the center of an open interval which contains no points of the Cantor set – we say that the Cantor set is nowhere dense. So the Cantor set is simultaneously as large as the entire unit interval, and so small that any point not in the set can be separated from it with an open interval. It is truly an amazing mathematical object! However, we are most interested in the fact that the Cantor set is strictly self-similar: the points in the Cantor set are all either in $[0, 1/3]$ or $[2/3, 1]$, and the subset of the Cantor set in either of these intervals is an exact copy of the entire set. So the Cantor set can be decomposed into two disjoint pieces, each of which is itself a Cantor set. In fact, at any stage n , the Cantor set can be decomposed into 2^n disjoint pieces, each an exact copy of the entire set.

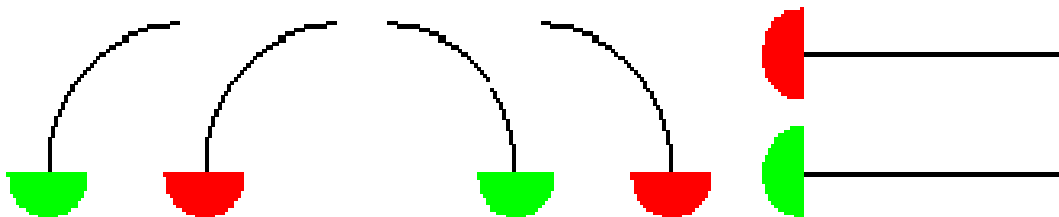
Koch Curve

Swedish mathematician Helge von Koch introduced the Koch curve in 1904, as an example of a curve that is continuous but nowhere differentiable. It is the result of a simple geometric construction that is easily generalized to give a wide variety of examples. Begin with a line segment in the plane – for example, the interval $[0, 1]$ along the x-axis (stage 0). On the middle third of this interval, construct an equilateral triangle upward (so each side of the triangle has length $1/3$, and one side is the interval $[1/3, 2/3]$), and remove the base (the open interval $(1/3, 2/3)$). The result is four line segments of length $1/3$, arranged as shown below (stage 1). For stage 2, perform the same construction on each of these four line segments so that the new segments protrude outward from those of stage 1 (see below) and continue to repeat the process indefinitely on all the smaller line segments formed at each stage. A few stages of this process are shown below:

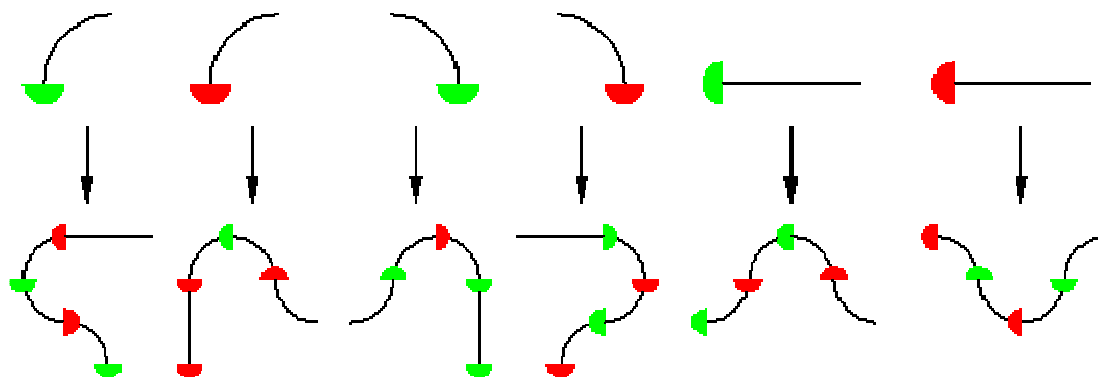


The Peano Curve and Fractal Curves

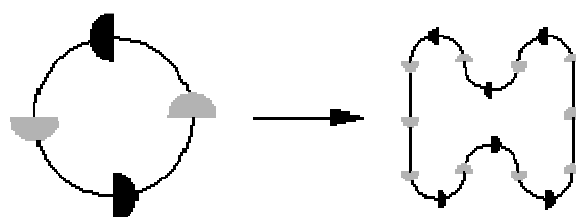
There are examples of curves (in the sense of continuous maps from the real line to the plane) that completely cover a two-dimensional region of the plane. We give a construction of such a Peano curve, adapted from David Hilbert's example. The construction is inductive, and is based on replacement rules. We consider building blocks of six shapes,



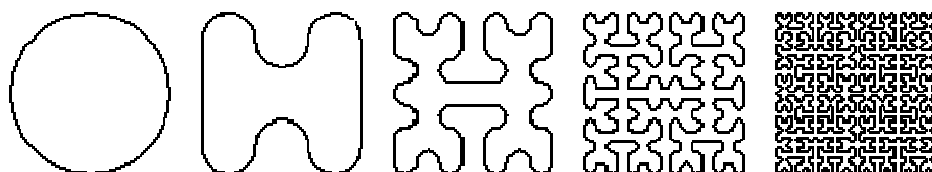
the length of the straight pieces being twice the radius of the curved ones. A sequence of these patterns end-to-end represents a curve, if we disregard the red and green half-disks. The replacement rules are the following:



The rules are applied taking into account the way each piece is turned. Here we apply the replacement rules to a particular initial pattern:



(We scale the result so it has the same size as the original.) Applying the process repeatedly gives, in the limit, the Peano curve. Here are the first five steps:



The same idea of replacement rules leads to many interesting fractal, and often self-similar, curves. For example, the substitution $\text{---} \rightarrow \text{---} \text{---}$ leads to the Koch snowflake when applied to an initial equilateral triangle, like this (the first three stages and the sixth are shown):

