

Paper Code: BCA 303

Paper ID: 20303

**Paper: Computer Graphics &
Multimedia Applications**

UNIT – I

Introduction: The Advantages of Interactive Graphics, Representative Uses of Computer Graphics, Classification of Applications, Development of Hardware and Software for Computer Graphics, Conceptual Framework for Interactive Graphics, Overview, Scan Converting Lines, Scan Converting Circles, Scan Converting Ellipses.

Graphics Hardware

Hardcopy Technologies, Display Technologies, Raster-Scan Display Systems, The Video Controller, Random-Scan Display Processor, Input Devices for Operator Interaction, Image Scanners, working exposure on graphics tools like Dream Weaver, 3D Effects etc.

Clipping

Southland-Cohen Algorithm, Cyrus-Beck Algorithm, Midpoint Subdivision Algorithm

UNIT – II

Geometrical Transformations

2D Transformations, Homogeneous Coordinates and Matrix Representation of 2D Transformations, Composition of 2D Transformations, The Window-to-Viewport Transformation, Efficiency, Matrix Representation of 3D Transformations, Transformations as a Change in Coordinate System.

UNIT – III

Representing Curves & Surfaces

Polygon Meshes, Parametric Cubic Curves, Quadric Surfaces. **Solid Modeling** Representing Solids, Regularized Boolean Set Operations, Primitive Instancing, Sweep Representations, Boundary Representations, Spatial Partitioning Representations, Constructive Solid Geometry, Comparison of Representations, User Interfaces for Solid Modeling.

UNIT – IV

Three Dimensional Viewing: Introduction, Representation of Three-dimensional objects, Projections, Parallel projections: Orthographic Projections, Oblique Projections. Perspective Projection, Three dimensional clipping, Hidden Surface Removal: Depth-Buffer(z-buffer) method, Depth-sorting Method(Painter's algorithm)

UNIT-I

Computer graphics tells us that what the actual workings of graphics are. Computer Graphics remains one of the most existing and repladly growing computer field. Computer Graphics as the pictorial representation or graphical representation of a computer

Application of computer graphics:

- (1) Computer – Aided Design
- (2) Presentation Graphics
- (3) Computer Art
- (4) Entertainment
- (5) Education and Training
- (6) Graphics provides one of the most natural means of communicating with a



तेजस्वि नावधीतमस्तु
ISO 9001:2008 & 14001:2004

computer.

- (7) Interactive computer graphics permits extensive, high-bandwidth user-computer interaction.
-

Representative Uses of Computer Graphics

Computer graphics is used today in many different areas of industry, business, government, education, and entertainment.

- User interfaces: GUI, etc.
- Business, science and technology: histograms, bar and pie charts, etc.
- Office automation and electronic publishing: text, tables, graphs, hypermedia systems, etc.
- Computer-aided design (CAD): structures of building, automobile bodies, etc.
- Simulation and animation for scientific visualization and entertainment: flight simulation, games, movies, virtual reality, etc.
- Art and commerce : terminals in public places such as museums, etc.
- Cartography: map making
- Simulation and animation for scientific visualization and entertainment.
- Multimedia textbooks

Classification of Applications

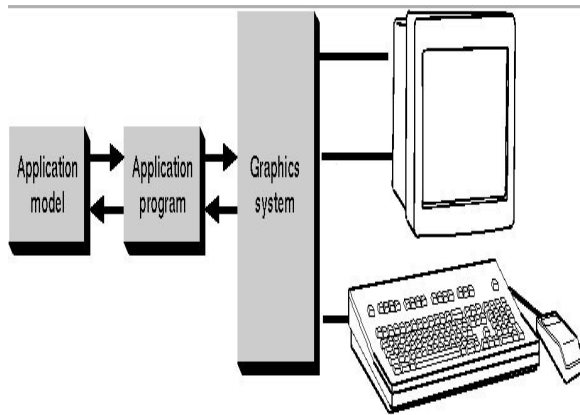
- **Paint programs:** Allow you to create rough freehand drawings.
 - **Animation software:** Enables you to chain and sequence a series of images to simulate movement. Each image is like a frame in a movie
 - **CAD software:** Enables architects and engineers to draft designs. It is the acronym for computer-aided design. A CAD system is a combination of hardware and software that enables engineers and architects to design everything from furniture to airplanes.
 - **Desktop publishing:** Provides a full set of word-processing features as well as fine control over placement of text and graphics, so that you can create newsletters, advertisements, books, and other types of documents
 - **business software:** enables users to create highly stylized images for slide shows and reports. The software includes functions for creating various types of charts and graphs and for inserting text in a variety of fonts.
-

Conceptual Framework for Interactive Graphics.



तेजस्वि नावधीतमस्तु
ISO 9001:2008 & 14001:2004

The high-level conceptual framework shown here can be used to describe almost any interactive graphics system



The three major parts of the framework are:

Application Modeling

Calculating what is to be displayed

Displaying the Model

Calling the graphics API routines

Interaction Handling

Handling user interaction, which will change the model, and therefore the display.

typically an event driven loop

Scan Converting Lines

Converting the geometric definition of a primitive form into a set of pixels that make up the primitive in the image space. This conversion task is scan conversion.

Types of Scan Conversion

1. Digital Differential (DDA) Algorithm
2. Bresenham's Line Algorithm

DDA algorithm is an incremental scan conversion method.

- Incremental scan-conversion method
- Faster than the direct use of the line equation
- a floating point operation is still required
- The line drifts away from the original line when the line is relatively long

AN ALGORITHM TO DRAW A LINE

1. Compute
 $dx = x_2 - x_1$ $dy = y_2 - y_1$
2. If $abs(dx) > abs(dy)$ then $steps = abs(dx)$
3. Else $steps = abs(dy)$
4. Plot a point at (x, y)



तेजस्वि नावधीतमस्तु
ISO 9001:2008 & 14001:2004

the **Bresenham's line-drawing algorithm**. This algorithm was developed by Jack E. Bresenham in 1962 at IBM.

5. $x_{inc} = dx / steps;$
6. $y_{inc} = dy/steps;$
7. $x = x_1$ and $y = y_1$
8. Plot a point at (x, y)
9. $k=1$
10. if $k = steps$, stop
11. $x = x + x_{inc}$
12. $y = y + y_{inc}$
13. Plot a point at (x, y)
14. $k = k+1$
15. Go to step 7

1. Highly efficient incremental method
2. Produces mathematically correct results using simple calculations

Bresenham's Line Drawing Algorithm for $m < 1$:

(1) Input the two line endpoints & store the left end point in (x_0, y_0) .

(2) Load (x_0, y_0) into frame buffer that is plot the first point.

(3) Calculate constants Δx , Δy , $2\Delta y$ and $2\Delta y - 2\Delta x$ and obtain the

starting value for the decision parameter as : $P_0 = 2\Delta y - \Delta x$.

(4) At each x_k along the line starting at $k = 0$, perform the following test if $P_k < 0$ the next point to plot is (x_{k+1}, y_k) and $P_{k+1} = P_k + 2\Delta y$

Otherwise the next point to plot is (x_{k+1}, y_{k+1}) and $P_{k+1} = P_k + 2\Delta y - 2\Delta x$.

(5) Repeat step 4 Δx times.

BRESENHAM LINE ALGORITHM

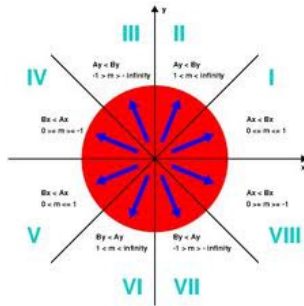
An accurate and efficient raster line generating algorithm,



तेजस्वि नावधीतमस्तु
ISO 9001:2008 & 14001:2004

Scan Converting Circles

A circle is a symmetrical figure, eight points can be plotted for each value that the algorithm calculates



A circle is a set of points that are at a given distance r from the center position (x_c, y_c) . This distance relationship is given as :

$(x - x_c)^2 + (y - y_c)^2 - r^2 = 0$ This equation is used to calculate the position of points along the circle path by moving in the x direction from $(x_c - r)$ to $(x_c + r)$ and determining the corresponding y values as :

$$y = y_c \pm \sqrt{r^2 - (x - x_c)^2}$$

Algorithm :

(1) Input radius r and circle center (x_c, y_c) and obtain the first point on

circumference of a circle centered on origin $(x_0, y_0) = (0, r)$

(2) Calculate the initial value of the decision parameter as : $P_0 = 5/4 - r$

(3) At each x_k position, starting at $k = 0$ if $P_k < 0$ the next point along the circle is (x_{k+1}, y_k) and $P_{k+1} = P_k + 2x_{k+1} + 1$, otherwise the next point along the circle is $(x_k + 1, y_k - 1)$ and $P_{k+1} = P_k + 2x_{k+1} + 1 - 2y_{k+1}$ where $2x_{k+1} = 2x_k + 2$ & $2y_{k+1} = 2y_k - 2$.

(4) Determine symmetry points in other seven octants.

(5) Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) & plot coordinate values $x = x_c + x$ & $y = y_c + y$.

(6) Repeat step (3) through (5) until $x \geq y$.

Scan Converting Ellipses

An ellipse is defined as a set of points such that the sum of distances from two fixed points (foci) is same for all points given a point $P = (x, y)$, distances are d_1 & d_2 , equation is :

$$d_1 + d_2 = \text{constant}$$

In terms of local coordinates

$$F_1 = (x_1, y_1) \text{ \& } F_2 = (x_2, y_2)$$

Display Technologies



तेजस्वि नावधीतमस्तु
ISO 9001:2008 & 14001:2004

In some graphics systems a separate processor is used to interpret the commands in the display file. Such a

Raster Display System:

- Interactive raster graphics employs several processing units
- Apart from CPU, a special purpose processor called video controller or display controller -> control operation of the display device
- Simple raster graphics system

Random Scan Display:

- The random scan display system with display processor.
- It except the frame buffer.
- In random scan display no local memory is provided for scan conversion algorithm.

Video Controller:

- Fixed area of system memory reserved for frame buffer
- Video controller given direct access to frame buffer to refresh the screen
- Coordinator origin is defined at lower left corner
- Scan lines labeled from y_{max} at the top of the screen to 0 at the bottom

1. Cohen-Sutherland Algorithm_(PPT)

Cohen-Sutherland Line Clipping in 2D

- ▶ Divide plane into 9 regions
- ▶ Compute the sign bit of 4 comparisons between a vertex and a clip edge
 - ▶ $(y_{max} - y, y - y_{min}, x_{max} - x, x - x_{min})$, cast the results to 0 or 1
 - ▶ Point lines inside the region: if all four bits are 0
- ▶ For **outcode** records results of four boundary tests
 - ▶ 0 if outside halfplane of top edge (above top edge)
 - ▶ 1 if outside halfplane of bottom edge
 - ▶ 2 if outside halfplane of right edge
 - ▶ 3 if outside halfplane of left edge
- ▶ Compute outcodes for both vertices of the input edge, denoted OC_1 and OC_2
- ▶ If $OC_1 = 1$ and $OC_2 = 0$ (i.e., outcode: 0000) then the input edge is trivially accepted
- ▶ Lines lying entirely in a particular halfplane are trivially rejected. That is: $OC_1 \text{ AND } OC_2 = 0$ (i.e., they share an "outside bit")

15 - Clipping 6/22

Cohen-Sutherland Algorithm

- ▶ If we can neither trivially accept or reject, then we do divide-and-conquer
- ▶ Subdivide line into two segments and test again
- ▶ Use a clip edge to cut line
- ▶ Use outcodes to choose which edge is crossed
 - ▶ The bits that are different between outcodes will tell us which edge to examine
- ▶ Pick an order for checking edges: top - bottom - right - left
- ▶ Compute the intersection point
 - ▶ Clip edge will be axis-aligned, so we can fix either the x or the y
 - ▶ Can substitute into the line equation
- ▶ Iterate for the newly created line segment, might need multiple passes (e.g., E-I at H)

15 - Clipping 7/22



तेजस्वि नावधीतमस्तु
ISO 9001:2008 & 14001:2004

frames of video, the output of Image processing can be either an Image or a set of characteristics or parameters related to the Image.

Cyrus-Beck Algorithm (PPT)

Cyrus-Beck / Liang-Barsky Parametric Line Clipping

- ▶ Use the parametric line formulation:
 - ▶ $P(t) = P_2 + t(P_1 - P_0)$
- ▶ Determine if the line intersects with a clip line (both extended to infinity)
- ▶ Decide if the intersection actually occurs on the polygon
- ▶ This is a very similar strategy for intersection tests in ray-tracing

UNIT-II

2D Transformations

Moving of an object to one place in Window area to another place is called a Transformation.

15 - Clipping

11/22

Transformation is to change the object's

- Position (translation)
- Size (scaling)
- Orientation
- rotation)
- Shapes (shear)

Image Scanners

Image Processing is any form of signal, processing for which Input is an Image such as photographs or

Rotation:

- Rotation is applied to an object by repositioning it along a circular path in the XY plane
- Positive values of theta for counter clockwise rotation
- Negative values of theta for Clockwise rotation
- To generate a rotation , we specify

Rotation angle theta

Pivot point (Xr,Yr)

Scaling:

- Scaling alters the size of an object .
- Uniform scaling means this scalar is the same for all components.
- Non –Uniform scaling different per component
- Operation can be carried out by multiplying each of it component by a scalar

Reflection: A reflection is a transformation that produces a mirror image of an object

- Reflection along x axis
- Reflection along y axis
- Reflection relative to an axis perpendicular to the xy plane and passing through the coordinate origin

- Reflection of an object relative to

Shearing: A transformation that distorts the shape of an object such that the transformed object appears as if the object were composed of internal layers that had been caused to slide over each other.

PPT

2D Transformation

- Translation

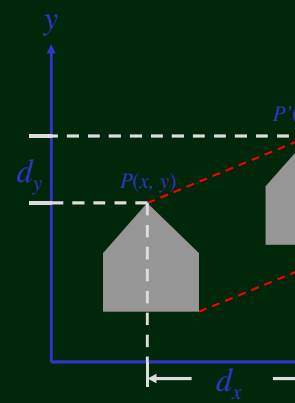
$P(x, y)$ move to $P'(x', y')$

$x' = x + d_x$

$y' = y + d_y$

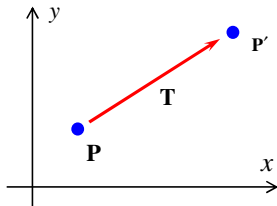
$P = \begin{bmatrix} x \\ y \end{bmatrix}, P' = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} d_x \\ d_y \end{bmatrix}$

$P' = P + T(d_x, d_y)$



The diagram shows a 2D coordinate system with x and y axes. A point P(x, y) is marked with a dashed line to its coordinates. A grey pentagon is shown at this point. A second point P'(x', y') is shown further to the right and up. A dashed red line connects P and P'. Horizontal and vertical dashed lines indicate the displacement distances dx and dy from P to P'.

2D Translation



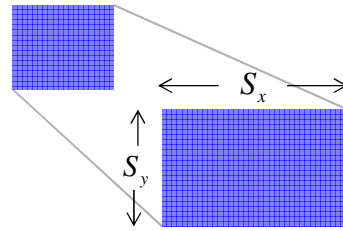
$$x' = x + t_x, \quad y' = y + t_y$$

$$\mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \mathbf{P}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

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

April 2010

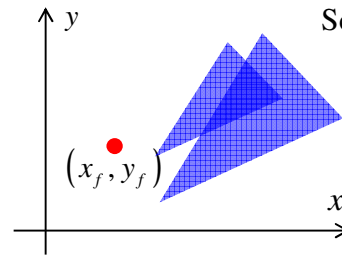
2D Scaling



$$x' = x \cdot s_x, \quad y' = y \cdot s_y$$

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

$$\mathbf{P}' = \mathbf{S} \cdot \mathbf{P}$$



Scaling about a fixed point (x_f, y_f)

$$x' = x \cdot s_x + x_f(1 - s_x)$$

$$y' = y \cdot s_y + y_f(1 - s_y)$$

$$\mathbf{P}' = \mathbf{P} \cdot \mathbf{S} + \mathbf{P}_f \cdot (1 - \mathbf{S})$$

April 2010

2D Transformation

• Scaling

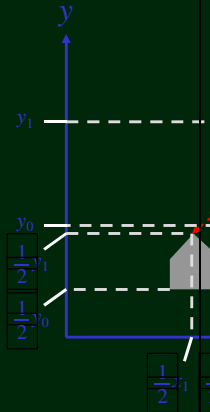
Scale by s_x along the x axis
and by s_y along the y axis

$$x' = s_x \cdot x$$

$$y' = s_y \cdot y$$

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

$$\mathbf{P}' = \mathbf{S}(s_x, s_y) \cdot \mathbf{P}$$



2D Transformation

• Rotation

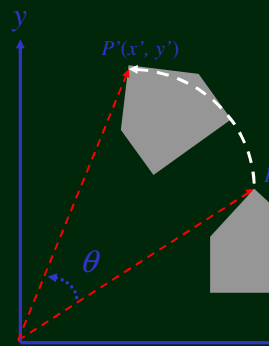
Rotate through an angle θ about the origin

$$x' = x \cdot \cos \theta - y \cdot \sin \theta$$

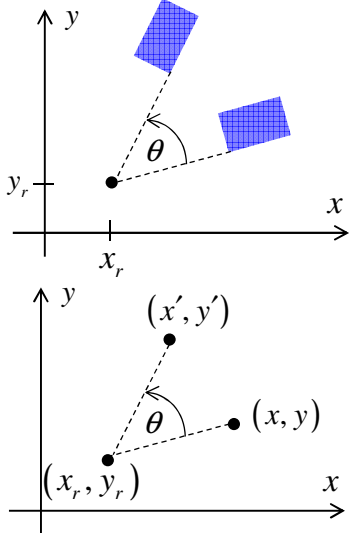
$$y' = x \cdot \sin \theta + y \cdot \cos \theta$$

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

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



2D Rotation



Rotation in
pivot (rotat

$$x' = x_r + (x - x_r) \cos \theta - (y - y_r) \sin \theta$$

$$y' = y_r + (x - x_r) \sin \theta + (y - y_r) \cos \theta$$

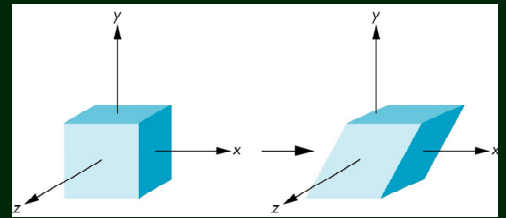
$$\mathbf{P}' = \mathbf{P}_r + \mathbf{R} \cdot (\mathbf{P} - \mathbf{P}_r)$$

$$\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

April 2010

Shear

- Helpful to add one more basic transform
- Equivalent to pulling faces in opposite dire



Angel: Interactive Computer Graphics 3E © Addison-Wesley 2002

2D Transformation

- Derivation of the rotation equation

$$x = r \cdot \cos \phi$$

$$y = r \cdot \sin \phi$$

$$x' = r \cdot \cos(\phi + \theta)$$

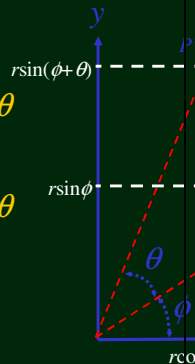
$$= r \cdot \cos \phi \cdot \cos \theta - r \cdot \sin \phi \cdot \sin \theta$$

$$y' = r \cdot \sin(\phi + \theta)$$

$$= r \cdot \cos \phi \cdot \sin \theta + r \cdot \sin \phi \cdot \cos \theta$$

$$x' = x \cdot \cos \theta - y \cdot \sin \theta$$

$$y' = x \cdot \sin \theta + y \cdot \cos \theta$$



Shear Matrix

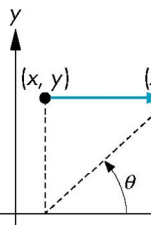
Consider simple shear along x axis

$$x' = x + y \cot \theta$$

$$y' = y$$

$$z' = z$$

$$\mathbf{H}(\theta) = \begin{bmatrix} 1 & \cot \theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Angel: Interactive Computer Graphics 3E © Addison-Wesley 2002

Homogeneous Coordinates and Matrix Representation of 2D Transformations,

PPT

Homogeneous Co

The general form of four dimensional coordinates is
 $p=[x \ y \ z \ w]^T$
 We return to a three dimensional po
 $x \leftarrow x/w$
 $y \leftarrow y/w$
 $z \leftarrow z/w$
 If $w=0$, the representation is that of a
 Note that homogeneous coordinates
 three dimensions by lines through
 dimensions

Angel: Interactive Computer Graphics 3E © Addison-Wesley 200

**Homogeneous Co
and Computer G**

- Homogeneous coordinates
computer graphics systems
 - All standard transformations (translation, scaling) can be in matrix multiplications with 4 x
 - Hardware pipeline works with representations
 - For orthographic viewing, we for vectors and $w=1$ for points
 - For perspective we need a pe

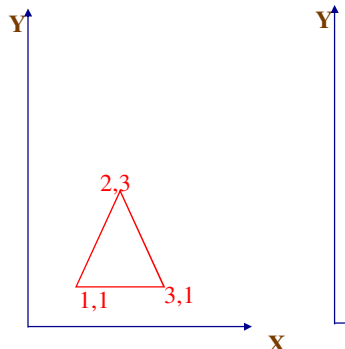
Angel: Interactive Computer Graphics 3E © Addison-Wesley 200

Composite Transformations

1. Sequence of composite transformation matrix and transformations could be setup by the matrix product of the individual transformations
2. Also as Concatenation or Composition of Matrices

Compositing Transfo

- Does order matter?
 - Case 1: translate by $(-2, 0)$, sc
 - Case 2: scale by $(2, 2)$, transla
 - Begin: red, 1st transform: pur



Case 1(translate then scale)

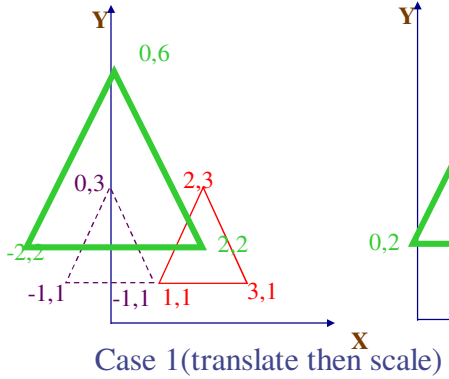


Window-to-Viewport Transformation

Window area on which object to be display and view port is generated in the window area and finally display in window area after selection of a particular object that should be consider as in view port.

Compositing Transfo

- Does order matter?
 - Case 1: translate by (-2, 0), sc
 - Case 2: scale by (2, 2), transla
 - Begin: red, 1st transform: pur



Composition Examp

$$P' = STP \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

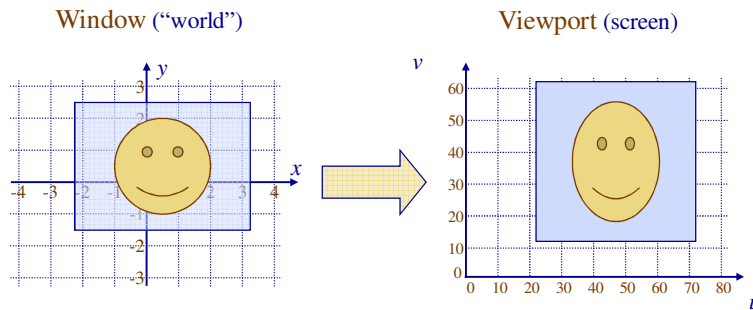
$$P' = TSP \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

In general, transformation

Window-to-Viewport Transform

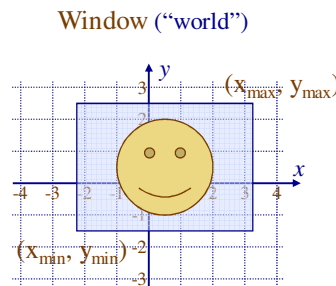
Need to transform points from “world” view (*window*) to the screen view (*viewport*)

- Maintain relative placement of points (usually)
- Can be done with a translate-scale-translate sequence



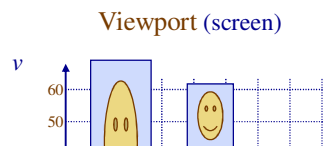
Window

- “Window” refers to the area in “world space” or “world coordinates” that you wish to project onto the screen
- Location, units, size, etc. are all determined by the application, and are convenient for that application
- Units could be inches, feet, meters, kilometers, light years, etc.
- The window is often centered around the origin, but need not be
- Specified as (x,y) coordinates



Viewport (cont)

- You can have multiple viewports
 - They can contain the same view of a window, different views of the same window, or different views of different windows



representations are the most common in computer graphics.

UNIT – III

Representing Curves & Surfaces

Polygon Meshes, Parametric Cubic Curves

Solid Modeling

Representing Solids, Regularized Boolean Set

Operations, Primitive Instancing, Sweep

Representations, Boundary Representations, Spatial

Partitioning Representations, Constructive Solid

Geometry, Comparison of Representations, User

Interfaces for Solid Modeling.

polygon mesh

- it is a collection of vertices
 - it is collection of edges and
 - collection of faces
 - it is a large sub-field of computer graphics and geometric modeling
 - operations performed on meshes may include Boolean logic, smoothing, simplification, and many others
-

Parametric Cubic Curves

Curves and surfaces can have explicit, implicit, and parametric representations. Parametric

- A parametric cubic curve is to be fitted to interpolate four points. The first and last points are to be at $u=0$ and $u=1$. The other two points are at $u=1/3$ and $u=2/3$, respectively. Find the equation of the curve in the form $\mathbf{P}(u)=\mathbf{U}^T [\mathbf{M}_p] \mathbf{B}$.
 - The geometric matrix G of a parametric cubic curve defines a straight line
 - A Bezier cubic curve obtained by a set of points.
 - A set of control points explain what happen to a Bezier segment when two of the control points are coincident.
-

Solid Modeling

- a) Primitive Instancing
 - It is set of Primitive 2D/3D solid shapes
 - Similar to parameterized object
 - A family with few difference in members
 - Relatively complex object
 - Without combing object
- b) Sweep Representations
 - Sweeping a object in 2D and 3D
 - Translational sweep
 - Rotational sweep
 - General sweep
- c) Boundary Representations



तेजस्वि नावधीतमस्तु
ISO 9001:2008 & 14001:2004

- Object description in terms of vertices, faces and edges
 - Some b-reps are restricted to planar , polygon etc...
- d) Spatial Partitioning Representations
- A solid is decomposed into a collection of adjoin nonintersecting solids
 - 1-Cell decomposition
 - 2-spatial occupancy enumeration
 - 3-Octress
 - 4- Binary space –partitioning trees
 - Unambiguous but not necessary unique
- e) Constructive Solid Geometry
- Operators at the internal nodes and easy primitive at the leaves
 - Not Unique
 - Deleting and adding replacing –modifying subtree etc..
- f) Comparison of Representations
- Accuracy
 - Domain
 - Uniqueness
 - Closure
 - Compactness and efficiency
-

Three Dimensional Viewing: Introduction, Representation of Three-dimensional objects, Projections, Parallel projections: Orthographic Projections, Oblique Projections. Perspective Projection, Three dimensional clipping, Three-dimensional Cohen-Sutherland clipping algorithm. **Hidden Surface Removal:** Depth- Buffer(z-buffer) method, Depth-sorting Method(Painter's algorithm)

Projections

Projection is 'formed' on the **view plane** (planar geometric projection) rays (projectors) projected from the **center of projection** pass through each point of the models and intersect projection plane. Since everything is synthetic, the projection plane can be in front of the models, inside the models, or behind the models..

Parallel:

- center of projection infinitely far from view plane
- projectors will be parallel to each other
- need to define the **direction of projection** (vector)
- 2 sub-types
 - orthographic - direction of projection is normal to view plane
 - oblique - direction of projection not normal to view plane
- better for drafting / CAD applications

Orthographic projection (or orthogonal projection)

- Means of representing a three-dimensional object in two dimensions.
- It is a form of parallel projection, where all the projection lines are orthogonal to the projection plane, resulting in every plane of the scene appearing in affine transformation on the viewing surface.
- It is further divided into *multiview orthographic projections* and *axonometric projections* which type of projection is used depends on the needs of the user - whether the goal is the mathematically correct depiction of length and angles, or a realistic looking image of the object

Perspective:

- center of projection finitely far from view plane
- projectors will not be parallel to each other



तेजस्वि नावधीतमस्तु
ISO 9001:2008 & 14001:2004

- need to define the location of the **center of projection** (point)
 - classified into 1, 2, or 3-point perspective
 - more visually realistic - has perspective foreshortening (objects further away appear smaller)
-

Oblique projection

- It is a simple type of technical drawing of graphical projection used for producing pictorial, two-dimensional images of three-dimensional objects.
 - Types
 - it projects an image by intersecting parallel rays (projectors)
 - From the three-dimensional source object with the drawing surface (projection plane).
-

Hidden-Surface Removal

- We now know which pixels contain which objects, however since some pixels may contain two or more objects we must calculate which of these objects is visible and which are hidden
- Hidden surface removal is generally accomplished using the Z-buffer algorithm
- In this algorithm, we set aside a two-dimensional array of memory (the Z-buffer) of the same size as the screen (#rows x #columns)

- This is in addition to the buffer we will use to store the values of pixels which will be displayed (color values)
- The Z-buffer will hold values which are depths (or z-values)
- The buffer is initialized so that each element has the value of the far clipping plane (the largest possible z-value after clipping has been performed)
- The other buffer is initialized so that each element contains a value which is the background color
- Now for each polygon we have a set of pixel values which that polygon covers
- For each one of these pixels, we compare its interpolated depth (z-value) with the value of the corresponding element already stored in the Z-buffer
 - If this value is less than the previously stored value, the pixel is nearer the viewer than previously encountered pixels
 - Replace the old value of the Z- buffer with the new, interpolated value and replace the old value of the other buffer with the value (color) of the pixel



तेजस्वि नावधीतमस्तु
ISO 9001:2008 & 14001:2004

- Repeat for all polygons in the image

Z Buffer

The easiest way to achieve hidden-surface removal is to use the depth buffer (sometimes called a z-buffer). A depth buffer works by associating a depth, or distance from the viewpoint, with each pixel on the window. Initially, the depth values for all pixels are set to the largest possible distance, and then the objects in the scene are drawn in any order.

Graphical calculations in hardware or software convert each surface that's drawn to a set of pixels on the window where the surface will appear if it isn't obscured by something else. In addition, the distance from the eye is computed. With depth buffering enabled, before each pixel is drawn, a comparison is done with the depth value already stored at the pixel.

If the new pixel is closer to the eye than what's there, the new pixel's colour and depth values replace those that are currently written into the pixel. If the new pixel's depth is greater than what's currently there, the new pixel would be obscured, and the colour and depth information for the incoming pixel is discarded.

Z Buffer in OpenGL

To use depth buffering in OpenGL, you need to enable depth buffering. This has to be done only once. Each time you draw the scene, before drawing you need to clear the depth buffer and then draw the objects in the scene in any order.

Scan-Line Algorithm

The scan-line algorithm is another image-space algorithm. It processes the image one scan-line at a time rather than one pixel at a time. By using area coherence of the polygon, the processing efficiency is improved over the pixel oriented method

Painter's algorithm

The idea behind the Painter's algorithm is to draw polygons far away from the eye first, followed by drawing those that are close to the eye. Hidden surfaces will be written over in the image as the surfaces that obscure them are drawn.

The concept is to map the objects of our scene from the world model to the screen somewhat like an artist creating an oil painting. First she paints the entire canvas with a

background colour. Next, she adds the more distant objects such as mountains, fields, and trees. Finally, she creates the foreground with "near" objects to complete the painting. Our approach will be identical. First we sort the polygons according to their z-depth and then paint them to the screen, starting with the far faces and finishing with the near faces.

The algorithm initially sorts the faces in the object into back to front order. The faces are then scan converted in this order onto the screen. Thus a face near the front will obscure a face at the back by overwriting it at any points where their projections overlap. This accomplishes hidden-surface removal without any complex intersection calculations between the two projected faces.

The algorithm is a hybrid algorithm in that it sorts in object space and does the final rendering in image space.

The basic algorithm :

- Sort all polygons in ascending order of maximum z-values.
- Resolve any ambiguities in this ordering.
- Scan convert each polygon in the order generated by steps (1) and (2).

Reference:

Book Reference

- Himalaya publication: Computer Graphics, Sumit chahan
- D. Hearn & Baker: Computer Graphics with OpenGL, Pearson Education, Third Edition

Web Reference

- <http://www.slideshare.net/KRvEsL/solid-modeling>



तेजस्वि नावधीतमस्तु
ISO 9001:2008 & 14001:2004

FAIRFIELD
Institute of Management & Technology
Managed by 'The Fairfield Foundation'
(Affiliated to GGSIP University, New Delhi)

2. http://web.iitd.ac.in/~pmpandey/RP_html_pdf/assignment%20Parametric%20Cubic%20and%20Bezier%20Curve.pdf
3. <http://www.bcanotes.com/Online/Computer%20Graphics/2D%20Transformation.html>
4. <http://ecomputernotes.com/computer-graphics/two-dimensional-transformations/what-is-transformation-type-of-transformation>
5. <http://askguru.net/d/2d-transformation-in-computer-graphics-ppt-download>
6. <https://sites.google.com/site/assignmentssolved/system/app/pages/search?scope=search-site&q=Antialiasing>
7. <http://www.slideshare.net/KRvEsL/solid-modeling>
<http://www.slideshare.net/KRvEsL/solid-modeling>
8. <http://www.google.co.in/search?q=bresenham+line+drawing+algorithm&source=lnms&tbm=isch&sa=X&ei=mnbSUYWsjYTKrAfvPIHwAQ&ved=0CAcQAUoAQ&biw=1024&bih=587>
9. http://www.cs.uic.edu/~jbell/CourseNotes/ComputerGraphics/Projections_Viewpoints.html
10. <http://www.cs.cityu.edu.hk/~helena/cs31622000B/Chap8Notes.pdf>
11. <http://www.cs.sun.ac.za/~lvzjl/courses/rw778/grafika/OpenGLtuts/Big/graphicsnotes009.html>
12. http://en.wikipedia.org/wiki/Orthographic_projection
13. http://en.wikipedia.org/wiki/Oblique_projection