COMPUTER AIDED DESIGN
(BME-42)

Unit-II: Computer Graphics Software

(2 Lectures)

- **Graphics Software,**
- **Software Configuration,**
- **Coordinate system,**
- **Graphics software functions**,
- Viewing transformations: windowing and clipping,
- Graphics software standards

Lecture 10

Topics Covered

Graphics Software

General Programming Packages

Special Purpose Application Packages

Rules for Designing Graphics Software

Graphics System Software Configuration

Coordinate Representation System

Functions of Graphics Software

Creation of Graphics Elements

Geometric Transformations

Prepared By

Prof. S. K. SRIVASTAVA

MED, MMMUT, Gorakhpur (UP)

sksme@mmmut.ac.in

GRAPHICS SOFTWARE

The graphics software is the collection of **commands or programs** written to make it convenient for user to operate the graphics system.

There are Two categories of graphics systems software:

General Programming Packages

The general programming software provides an extensive set of graphics functions written in high level programming languages such as **C** or **C++.** It includes commands to perform the following activities:

1. To *generate* the images on CRT displays using 2D and 3D output primitives such as straight lines, polygons, circles, cylinders, spheres, etc. and setting *color* and *intensity* values to these primitives

2. To *manipulate* the graphics images on CRT displays by applying various types of geometric transformations

3. To achieve various types of *interactions* between the user and computer graphics system

GRAPHICS SOFTWARE…

Special Purpose Application Packages

- For *non-programmers*, the special purpose application packages can produce the graphics images without worrying about the operations of application packages.
- The user *communicates* with graphics packages in their own ways.
- The *Application packages* are available for graphics applications such as architectural work, aerospace engineering, chemical engineering, etc.

In addition to graphics software, there may be additional application programs specialized functions related to CAD/CAM activities. They are:

Design Analysis Programs: Finite Element Analysis, Kinematic synthesis, Dynamic analysis, Pipe routing, Simulation and modeling, etc.

Manufacturing Planning Programs: Automated process planning, Numerical control part programming, Materials requirement planning, etc.

GRAPHICS SOFTWARE…

Rules for Designing Good Graphics Software

- The graphics software system is highly *dependent* on the types of hardware used.
- The software must *exploit* the features of graphics input devices and graphics display devices.
- The *commands* for vector displays would be different from raster displays.
- Similarly, the use of *refresh* display or *storage* tube display may affect the commands of graphics software.
- Good CAD software must follow certain ***Rules and Standards***. It is essential to know the requirements of good CAD software.

GRAPHICS SOFTWARE…

Rules for Designing Good Graphics Software…

Simplicity: Graphics software must be simple to use, easy to understand and be user friendly.

Consistency: The graphics software should operate in a consistent and predictable way to the user.

Completeness: There should be no omissions in the set of graphics functions. The graphics software should be complete in itself.

Robustness: The graphics software should be tolerant of minor misuse by the user. Software must always warn for any type of software damage instead of crashing. After removing the error, it should continue to work smoothly without hindrance.

Flexibility: The software must be able to incorporate the modifications without much problem. This leads to easy maintenance of software.

GRAPHICS SOFTWARE…

Rules for Designing Good Graphics Software…

Portability: The software must be portable. It should have capability of operating on any graphics system, i.e., it should be independent of graphics system. The use of Graphics Standards provides portability to graphics software.

Performance: The software should exploit as much as possible the performance of system hardware. Graphics software should be efficient, and speed should be fast and consistent. Efficient software effectively utilizes the performance of CPU in terms of computation time and storage.

Economy: Graphics software should not be very large and expensive in terms of *money* and *computation time*.

GRAPHICS SYSTEM SOFTWARE CONFIGURATION

The user performs following activities during the operations of graphics system software:

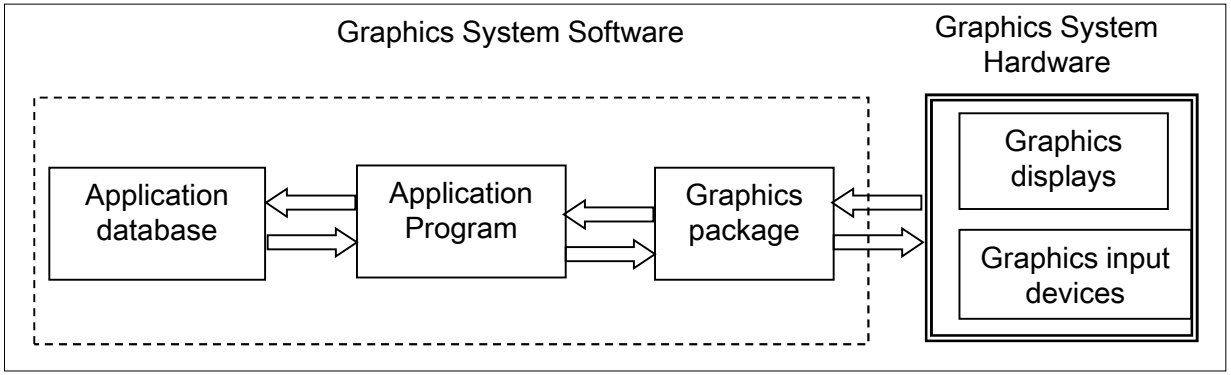
(i) *Creation* and *modification* of 2D and 3D entities generated on the displays

(ii) *Construction* of application models using these entities

(iii) *Storage of application models* into the computer memory or secondary storage.

These activities performed in combination rather than sequentially, and they are the part of any Interactive Computer Graphics (ICG) packages.

The graphics software consists of following *three* modules, interconnected as shown below



Interconnectivity of various modules of graphics system software

GRAPHICS SYSTEM SOFTWARE CONFIGURATION

Graphics Package

- The interaction between the user and graphics system is achieved through the graphics package.
- It provides an *interface* between the users and the application programs.
- The graphics software *accepts* the input command and data from the user through input subroutines, *forwards* them to the application program, and finally *presents* the graphics image of application model on the displays through the output subroutines.

Application Program

- Application program, the central module of any graphics software, constructs the physical model of particular design problem on the display devices.
- The application program controls the data transmission and retrieval from the application database for the image creation for the specific major areas.

GRAPHICS SYSTEM SOFTWARE CONFIGURATION

Application Program…

- There are following major areas such as
  - *architecture,*
  - *construction,*
  - *mechanical and electrical engineering*
  - *aerospace engineering,*
  - *flight simulation,*
  - *electronic circuit design,*
  - *pipe routing, etc.*
- Each application program utilizes the standard images and international conventions related to their specific areas.

Application Database

- The contents of the database can be displayed either on the screen or obtained in hard copy form through different graphics output devices.

GRAPHICS SYSTEM SOFTWARE CONFIGURATION

Application Database…

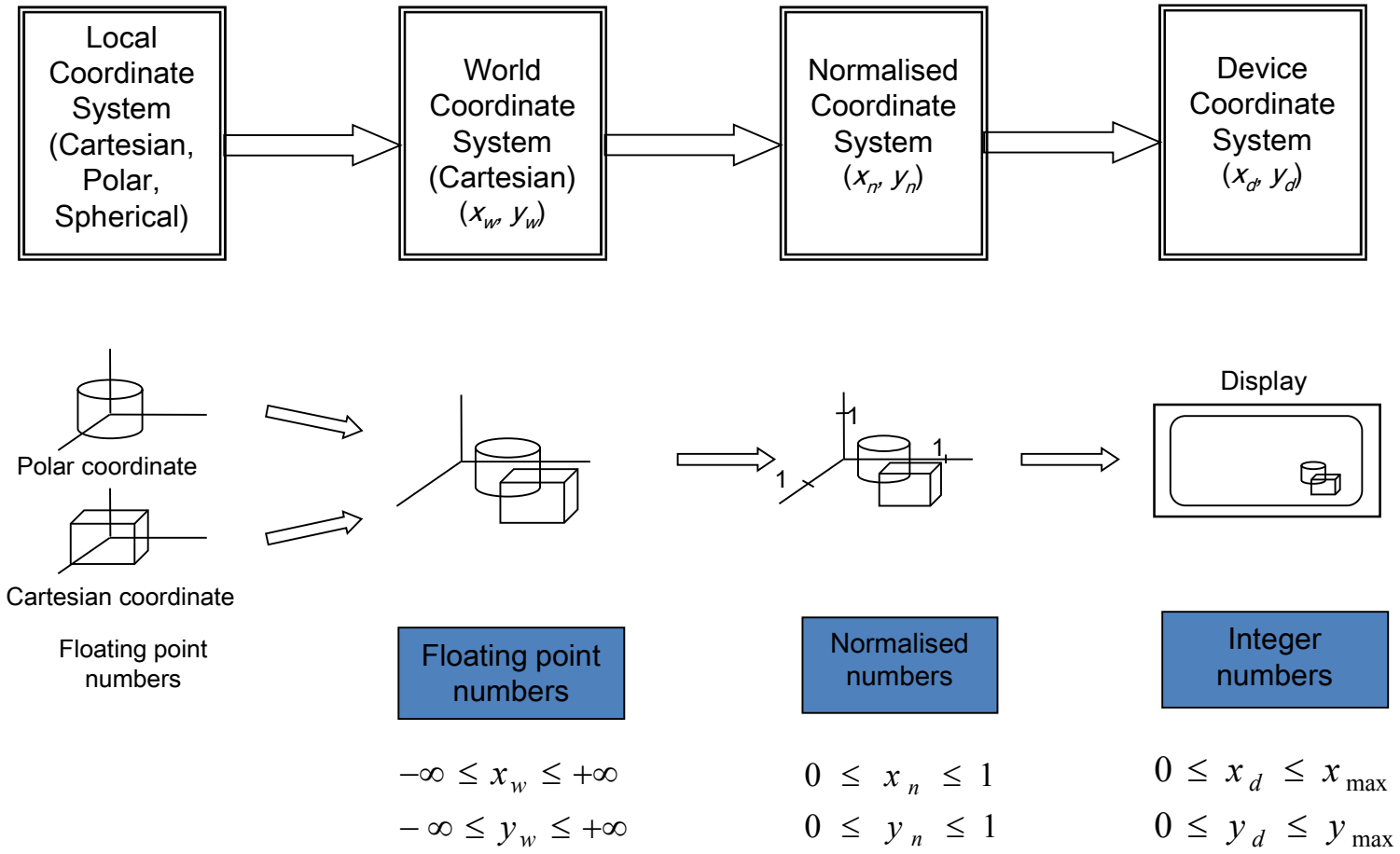
The database of graphics software system contains the following information:

- Basic 2D (point, lines, polygons, curves, conic sections, etc.) and 3D (sphere, box, pipes, ellipsoid, springs, gears, etc.) graphics elements appropriate to the application model.
- Logical, mathematical and numerical definitions of application models, e.g., surface definition of automobile bodies, design information of electronic components used in electronic circuits, mathematical definition of mechanical components, etc.
- Alphanumeric information associated with the components/models such as mechanical and chemical properties, bill of materials, manufacturing information, mass properties, etc.
- Topology of models, i.e., how various components are connected to form the assembly of application model
- Applications specific analysis programs, e.g., *finite element stress analysis, kinematic synthesis, dynamic analysis, vibration analysis, fluid flow analysis, thermal analysis, structural analysis, etc.*

COORDINATE REPRESENTATION SYSTEM

- Every CAD/CAM system follows certain type of coordinate representation system.
- During display of an image, the *mapping of coordinates* of the object consisting of 2D and 3D primitives occurs onto the display device or workstation.
- This is obtained through the *coordinate transformations*, also referred to as viewing transformations.
- Mostly, the graphics software design follows the *Cartesian* coordinate system.
- If coordinates of an image is defined in other coordinate system (e.g., cylindrical or spherical coordinate system), they must be *converted into the Cartesian* coordinates before using in the graphics software.
- Figure shows the viewing transformation sequence from *local* (i.e., modeling or master) coordinates to the *device* coordinates.

COORDINATE REPRESENTATION SYSTEM



Viewing transformation sequence from local coordinates to device coordinates

COORDINATE REPRESENTATION SYSTEM

World Coordinate System

- World Coordinate System (WCS) is the *working* or *user* coordinate system, which describes the image in *Cartesian* coordinates.
- At first, the shape of object is created in the form of graphics image, using separate coordinate reference frames, known as *local* coordinate system.
- The local coordinate systems may be *Cartesian, Polar or Spherical*.
- In world coordinate system, the units are the user units, which can be anything like mm, m, km, foot, etc.
- Once, all the objects in graphics image are described by their individual local coordinate systems, they can be placed in the graphics image with reference to one single reference frame, i.e., *Cartesian coordinates*.

COORDINATE REPRESENTATION SYSTEM

World Coordinate System…

- The WCS may
  - *have range from +∞ to -∞ in both the directions*
  - *Have positive values or negative values*
  - *have numerical values that depends on the type of problem*
  - *be represented by floating point numbers (e.g., 0.134 x 10^7 ; mantissa = 0.134 and exponent = 7)*

Normalized Coordinate System

- During modeling, each graphics output device may follow different coordinates.
- For some images, we might want to specify object dimensions in fraction of a foot, while for some other applications it may be millimeters, kilometers.

COORDINATE REPRESENTATION SYSTEM

Normalized Coordinate System…

- It is, therefore, desirable to convert the *World* coordinates into the *Normalized* coordinates, i.e., Normalized Coordinate System (NCS), to make the coordinate system *independent of several graphics output devices*.

- Normalization may be done from (0, 0) to (1, 1) with origin at (0, 0) in the lower left corner, and coordinate (1, 1) on the right top corner of the display devices.

- To accommodate the differences in scales and aspect ratios, the mapping of normalized coordinates into square area of the displays is required to maintain the proper proportions of various images.

COORDINATE REPRESENTATION SYSTEM

Device Coordinate System

- The image in normalized coordinate system has to be displayed on an output device like monitor (soft device), printer/plotter (hard device) in Device Coordinate System (DCS).
- A graphics device understands the device coordinate system in terms of *pixels, cm, inch,* etc.
- Depending upon the *pixel density*, the DCS would vary from one graphics system to another.

The DCS has following features:

- *Maximum size depends upon the pixel density (e.g., 1024x1024) of display devices*
- *Always consider the positive values*
- *Always fixed in size (i.e., size of display surface) irrespective of the problem*
- *Represented by an integer number*

FUNCTIONS OF GRAPHICS SOFTWARE

- Graphics software performs variety of functions for creating and manipulating the graphics images.
- These functions are grouped into the function sets.
- Each set performs certain kind of interaction between the user and the graphics system to deal with input, output, attributes, transformations, viewing or general control of the graphics image.

In general, graphics software is equipped with the following function sets:

1. Creation of Graphics Elements

- This function set contains basic 2D and 3D graphics elements (output primitives) such as dots, lines, curves, surfaces, polygons, conic sections forming 2D image entities; and spheres, cubes, cylinders, cones, etc. as 3D entities.

FUNCTIONS OF GRAPHICS SOFTWARE…

1. Creation of Graphics Elements…

- Sometimes, a special *hardware chip*, provided with the graphics software, contains basic graphics elements for the development of application models.
- The 2D and 3D graphics elements may be associated with certain kind of attributes that describe how a particular graphics element appears on the screen.
- The attributes for graphics elements may include intensity, color specifications, line and text style, area filling pattern, etc., which can be used to modify the basic 2D and 3D images whenever needed.

2. Geometric Transformations

- After the development of output primitives with suitable attributes, it is now possible to develop a variety of images for the application models, with the facility of *modifying* it, for the better understanding and visualization.

FUNCTIONS OF GRAPHICS SOFTWARE…

2. Geometric Transformations…

- The components of applications models are capable of modifying in *different sizes, shapes, positions and orientations.*
- Geometric transformations incorporate such modifications in the graphics elements, during *construction* of the application model as well as to *reposition the model* in the database by modifying its coordinate's description.

The geometric transformations include:

- Translation (*repositioning* the image from one place to other)
- Rotation (*orientation* of image in 2D plane)
- Reflection (*orientation* of image in 3D plane)
- Scaling (*enlargement/reduction* in the size of image)
- Shear (*controlled distortion* of image in 2D and 3D planes)

COMPUTER AIDED DESIGN
(BME-42)

Unit-II: Computer Graphics Software

(2 Lectures)

- Graphics Software,
- Software Configuration,
- Coordinate system,
- **Graphics software functions,**
- **Viewing transformations: windowing and clipping,**
- **Graphics software standards**

Lecture 11

Topics Covered

Functions of Graphics Software

Viewing Transformation

Segmenting Functions

User Input Functions

Graphics Software Standards

General Programming Software

Application based Software

Prepared By

Prof. S. K. SRIVASTAVA

MED, MMMUT, Gorakhpur (UP)

sksme@mmmut.ac.in

FUNCTIONS OF GRAPHICS SOFTWARE…

3. Viewing Transformations

It is very difficult to visualize *critically the complex drawings* of objects in world coordinate system. Therefore,

- It is required to display only those portions of drawing on the displays that are of immediate interest; thus, a *window* is required to visualize portions of the drawing on the display surface such that
  - The selected rectangular area in the world space, with sides parallel to directions of the world coordinate system, is called a *window*;
  - and method of selecting and enlarging portions of drawing on the displays is termed as *windowing*.
  - The image, which lies within this window, is projected onto the display surface.

The viewing transformation generates images the way the user wants to present.

FUNCTIONS OF GRAPHICS SOFTWARE…

3. Viewing Transformations…

- This function set provides the user with the ability to view the images from the desired angle and at the desired magnification.

Windowing Transformation

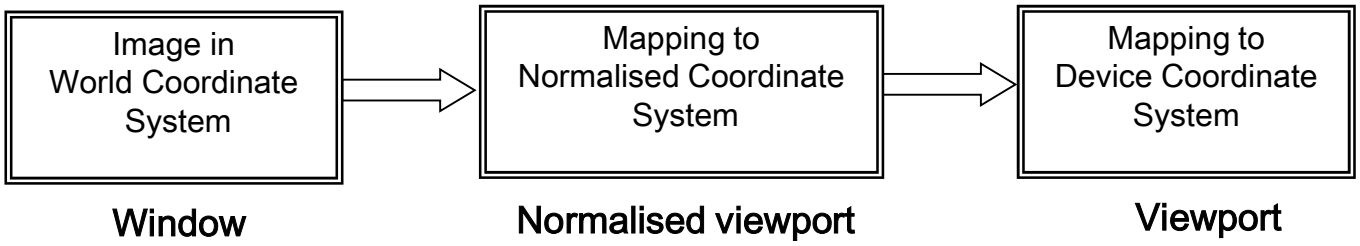
- This is a two-dimensional viewing transformation.
- The graphics screen behaves as a window to display the application model.
- The magnified view of the display area in windowing is added advantage to the user.
- In windowing process, we see the area of interest in a box on the screen. Such a box on the screen is termed *viewport*.

Window refers to a particular area in WCS whereas viewport is the area on the device, which displays window contents.

FUNCTIONS OF GRAPHICS SOFTWARE…

3. Viewing Transformations…

Windowing transformation…



Viewing transformation of image

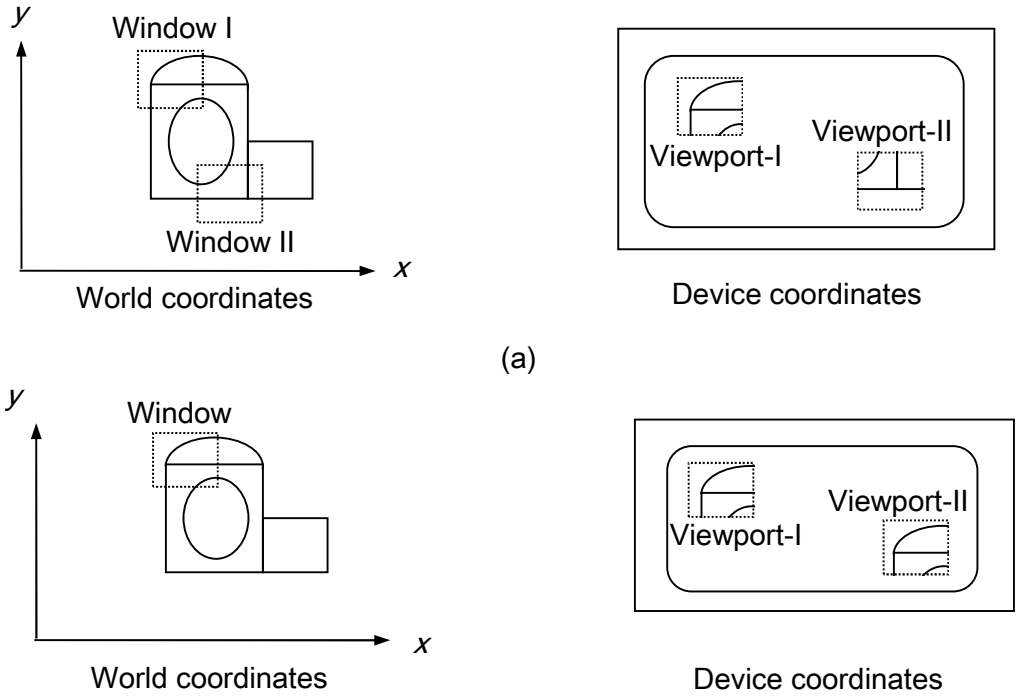
- Figure shows different stages of the viewing transformation from window to viewport.
- The viewing transformation is the process of mapping coordinates/points from window (i.e., object in WCS) onto the viewport (i.e., object in DCS).

Mathematically, viewing transformation is the operation of a transformation matrix on the coordinates of every point of the object.

FUNCTIONS OF GRAPHICS SOFTWARE…

3. Viewing Transformations

Figure shows windowing operation to display certain areas of the images (in world coordinates) on different viewports.



(a) Different windows on different viewports (b) Same windows on different viewports

FUNCTIONS OF GRAPHICS SOFTWARE…

3. Viewing Transformations…

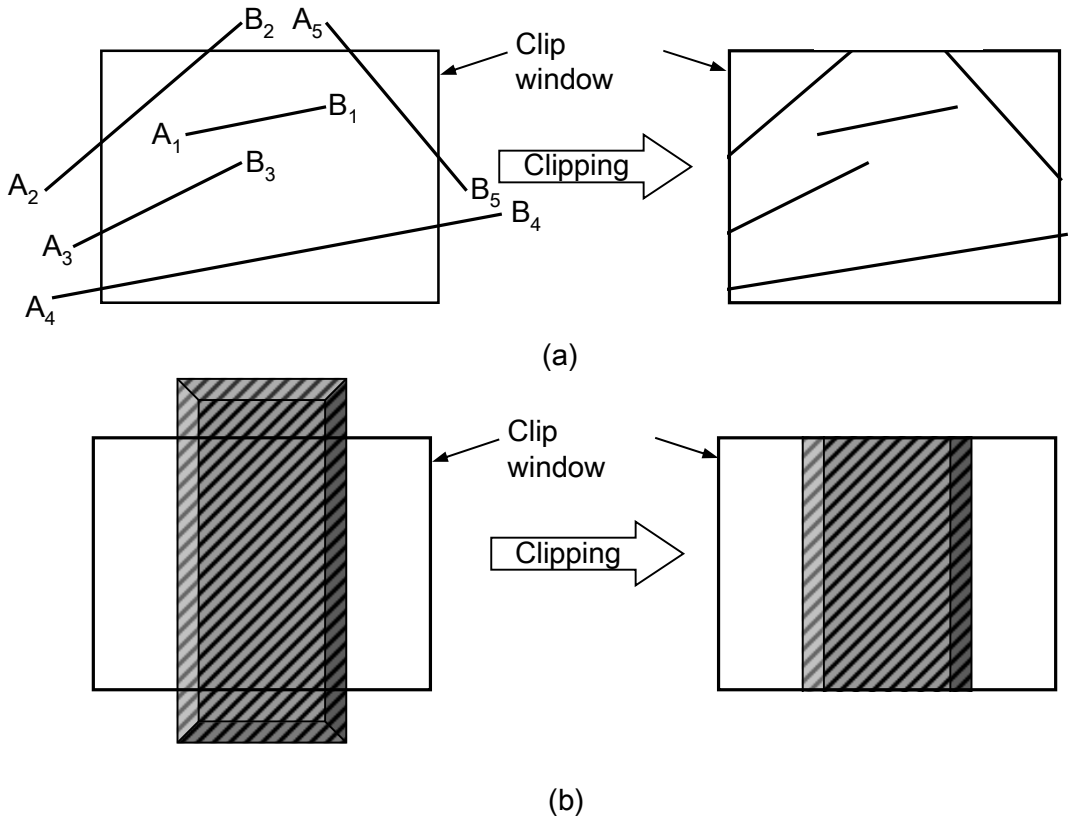
Clipping Transformation

- Sometimes, a portion of drawing may lie outside the rectangular boundary of the window that is not desirable.
- The process of *removing those parts of the images lying outside the window* is called *clipping*. This is another important feature of the viewing transformation.
- The process of hiding a portion of the image, which is outside the viewport boundary, is termed clipping.
- Alternatively, it is the process of determining the portion of an image lying within a region called clip region.
- Figure shows primitives being clipped (1D, e.g., lines or 2D, e.g., filled polygons).
- Since complex images consist of thousands of small line segments, efficient clipping is necessary to reduce the time required for generating the images.

FUNCTIONS OF GRAPHICS SOFTWARE…

3. Viewing Transformations…

Clipping Transformation…



(a) Clipping of one-dimensional primitives (b) Clipping of two-dimensional primitive

FUNCTIONS OF GRAPHICS SOFTWARE…

4. Segmenting Functions

- In complex drawings, it is very difficult to modify a portion of the image because of rapid scan conversion of graphics images.
- After modifications, the entire complex image regenerates in each cycle from image information received from the refresh buffer.
- The term segment in the function set refers to a portion of image that requires modification. The segment may be either a single element or groups of elements.
- This function set provides the capability to selectively replace, modify or delete a portion of the image.
- For example, on a machine drawing, one view of a bolt is required to draw at many places. It is easier to treat this view of the bolt as single entity, and reproduce the entity, wherever necessary.

FUNCTIONS OF GRAPHICS SOFTWARE…

4. Segmenting Functions…

- This process of grouping primitives together and treating it as a single entity for the purpose of further manipulation is called *segmentation*.
- During operation, first, a name is assigned to the segment portion; thereafter, the contents of that portion of display file is modified or deleted, to execute the specific segmenting function.
- Since the segmenting process involves the redrawing of the entire image at a refresh rate of 30 or more times per second; therefore, storage type of CRT displays are not suitable.
- Raster scan displays are ideally suitable for the segmentation process.

FUNCTIONS OF GRAPHICS SOFTWARE…

5. User Input Functions

- The user input functions are used to input the command/data in the graphics system.
- The software commands are written specifically for the types of graphics input devices.
- The user input functions should be organized in a systematic manner, to maximize the benefits of interactive computer graphics, by exploiting the facilities of graphics system.
- The software should be user friendly, i.e., it should be easy to work with the given graphics input devices.
- The software should be designed in such a way that it takes care of all the data entry situations in an easy manner without any confusion, and a designer with little or no programming experience can work *efficiently and effectively* with the graphics system.

GRAPHICS SOFTWARE STANDARDS

Graphics software performs various activities to display the graphics images for the following CAD applications:

- *Display of drawings*
- *Solid model and its components*
- *Wireframe geometry of the model*
- *Animation of assembly*
- *Art and paint applications*
- *Generation of documents, reports, etc.*

The computer program for CAD application is either developed in any one of the high-level programming languages or based on the application software.

GRAPHICS SOFTWARE STANDARDS…

There are Two types of graphics software:

1. General Programming Software

- A high-level programming language such as C & C++ is used for the development of general programming software.
- The set of graphics functions, available in the Graphics Library (GL), are used in software for generating the output primitives such as line, ellipse, spline, polygon, etc.

2. Application based Software

- The drafting software, analysis software and complete CAD software are extensively used for the problems related with mechanical engineering applications.
- These are icon-based software and user never bothers about the software commands required in the execution of certain operations.

GRAPHICS SOFTWARE STANDARDS…

2. Application based Software…

- The graphics system performs all the graphics related activities when it receives the commands from the source code of application program.
- The graphics system is embedded in source code in the form of subroutines; therefore, making the software dependent on the type of hardware used.
- If input/output devices change, its related software becomes obsolete unless we modify the software or go for the new software as per the new hardware configuration.
- This is very costly approach for the user as well as for the vendor.

Thus, graphics software needs **standards** to reduce the further cost of modifying their existing software so that it could run on the graphics system with new/modified configurations.

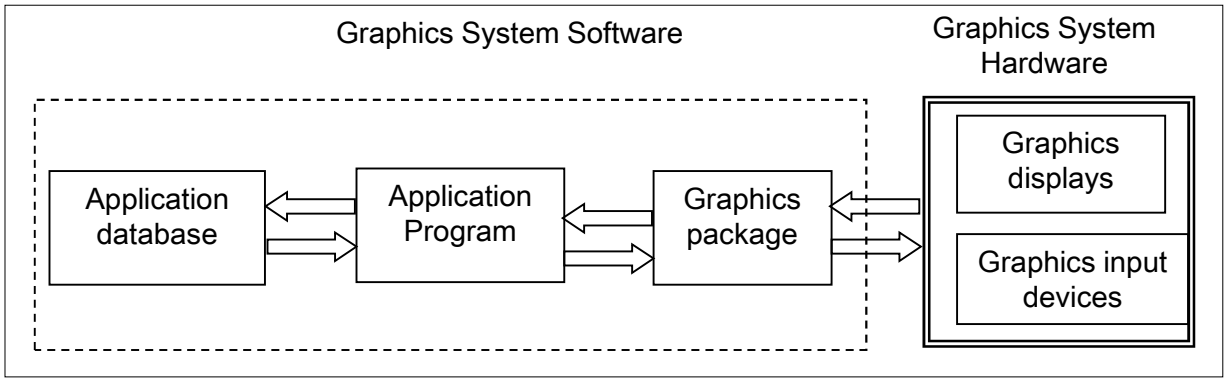
GRAPHICS SOFTWARE STANDARDS…

The use of graphics standards provides the following benefits to the user/vendor:

- Application program portability to any graphics system; thus, eliminates the hardware dependency of the software.
- Programmer portability to new graphics system is possible without learning the new set of related graphics commands.
- Text portability ensures that text associated is independent of the graphics hardware system.
- Design and manufacturing database portability from one graphics hardware to another.

GRAPHICS SOFTWARE STANDARDS…

Figure shows interconnectivity of various modules of a graphics software prior to the development of graphics standards

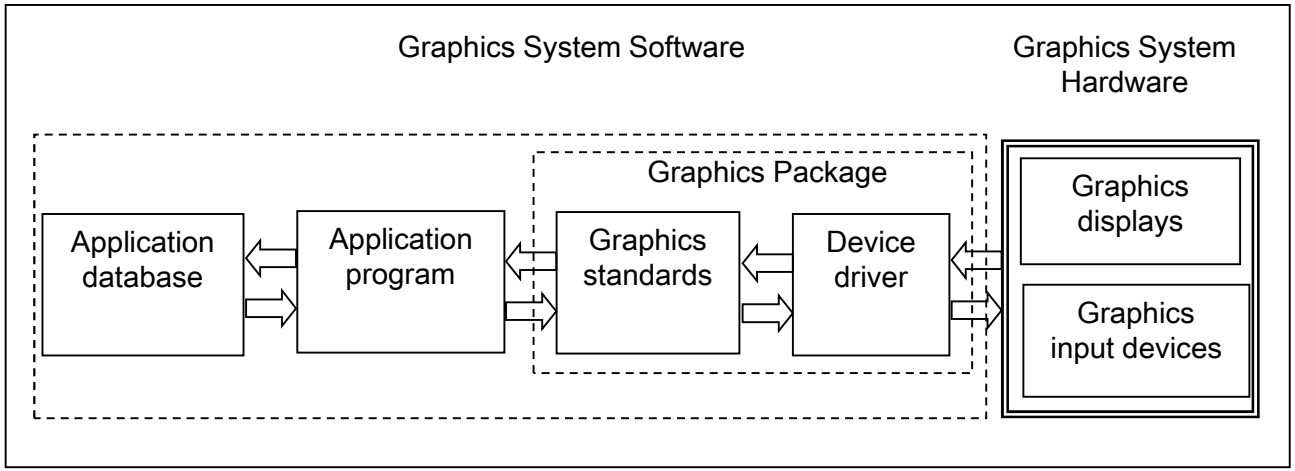


- In this system, the portability of application program affects if we change any one of the graphics input/output devices.
- We cannot transport the application program to the new/modified configuration of graphics hardware system (input/output devices) unless the graphics software is changed.
- This requires the standardization of graphics software.

GRAPHICS SOFTWARE STANDARDS…

The use of graphics standards, in the development of graphics software, makes the application program *independent* of graphics input/output devices. The use of Graphics standards makes the application program device independent.

The graphics software standardization was achieved by dividing the graphics package into two parts as shown below:



Graphics software system with graphics standards and device driver

GRAPHICS SOFTWARE STANDARDS…

- The first part is Graphics Standards, which is hardware independent and second part as device driver being hardware dependent.
- This led to the development of first graphics standard known as Graphics Kernel (core) System (GKS) adopted by International Standards Organization (ISO).
- The graphics standards act as a *buffer* between the application program and hardware device and ensures the portability of application programs.
- In this software system, the application program interacts with the graphics standards, which in turn, interacts with the device driver, and interaction ends at the input/output devices.

GRAPHICS SOFTWARE STANDARDS…

- If there is any change in the configuration of input/output hardware device, only device driver is replaced; and graphics standards as well as application programs remain unaffected.

- Therefore, graphics software, system programmer and CAD/CAM database become portable by incorporating the graphics standards in the graphics software system.

The CAD/CAM software community, all over the world, is using the following graphics standards at various levels:

GKS

- Graphics Kernel System (GKS) developed in 1981 by the American National Standards Institute (ANSI) and ISO for 2D modeling and GKS-3D for the 3D modeling.

- GKS provides six output primitives: Polyline (line type, width scale factor, etc.), Polymarker (marker type, size scale factor, color, etc.), Text, Fill area, Cell array and generalized drawing primitives.

GRAPHICS SOFTWARE STANDARDS…

IGES

- Initial Graphics Exchange Specifications (IGES) adopted in 1981 as ANSI standard Y14.26M, for functioning at the level of database or application data structure.
- IGES standard contains geometric entities such as curves, surfaces, solid primitives and Boolean operations.
- Wireframe, surface and solid modeling software can be developed around IGES.

PHIGS

Programmer's Hierarchical Interface for Graphics (PHIGS) adopted for additional capabilities such as color specification, segmentation, image manipulations for dynamic ability, etc. Later, PHIGS+ was developed for 3D surface shading of images.

DIN

Ditches Institute for Normung (DIN) developed GKS in 1985 which was accepted as ISO-7926 for 2D modeling

GRAPHICS SOFTWARE STANDARDS…

DMIS

Dimensional Measurement Interface Specifications

VDM

Virtual Device Metafile (VDM) works at the level just above the device driver and defines the functions required to describe an image. The image definition can be stored and/or transmitted from existing graphics hardware to the new/modified graphics hardware.

VDI

Virtual Device Interface (VDI) is the lowest graphics interface lies in between GKS/PHIGS and the device driver

GKSM

GKS Metafile

COMPUTER AIDED DESIGN
(BME-42)

Unit-II: Output Primitives

(3 Lectures)

- **Scan conversion of primitives**
- **Line generation algorithms**
- **DDA Line Drawing algorithm**
- Bresenham's line drawing algorithm,
- Circle generating algorithm-Cartesian coordinates, Polar coordinates
- Bresenham's circle generating algorithm

Lecture 12

Topics Covered

Output Primitives

Image Representation in Computers

Scan Conversion of Lines

Digital Differential Analyzer (DDA) Algorithm

Limitations

Prepared By

Prof. S. K. SRIVASTAVA

MED, MMMUT, Gorakhpur (UP)

sksme@mmmut.ac.in

Image Representations in Computers

- In computer graphics, the images are created on the display devices.
- For better understanding of image generation, it is necessary to know more about the image representation mechanism.
- It has been observed that an image is finally represented and stored in the frame buffer in the form of *binary digits*, i.e. bits.

Analysis of Image Transfer Mechanism in Computer Graphics

1. Image Conversion in Binary Form

- An image is visualized in the form of collection of straight-line segments, curves and text matter. If we go insight, these entities are composed of *collection of dots*.
- For example, a line can be represented by the collection of dots. Similarly, an arc is represented by the collection of dots lying on the trajectory of an arc.

IMAGE REPRESENTATIONS IN COMPUTERS

1. Image Conversion in Binary Form…

- The exact positions of these dots depend on the radius, initial/end angles, endpoints coordinates, etc.
- Most of the modern CAD systems use only *raster displays* for the generation and representation of graphics images.

2. Frame Buffer

- In computer graphics, the binary image information is stored in the memory, called as *frame buffer or refresh buffer*.
- The display surface for raster displays is divided into grids of horizontal and vertical lines. Each line consists of series of small picture elements, i.e. *pixels*.
- A pixel is made bright or dark depending upon the two states of *binary digits* 1 or 0, respectively.

IMAGE REPRESENTATIONS IN COMPUTERS…

2. Frame Buffer…

- This is achieved by assigning 1 bit memory corresponding to each pixel of the viewing screen.

3. Scan Conversion

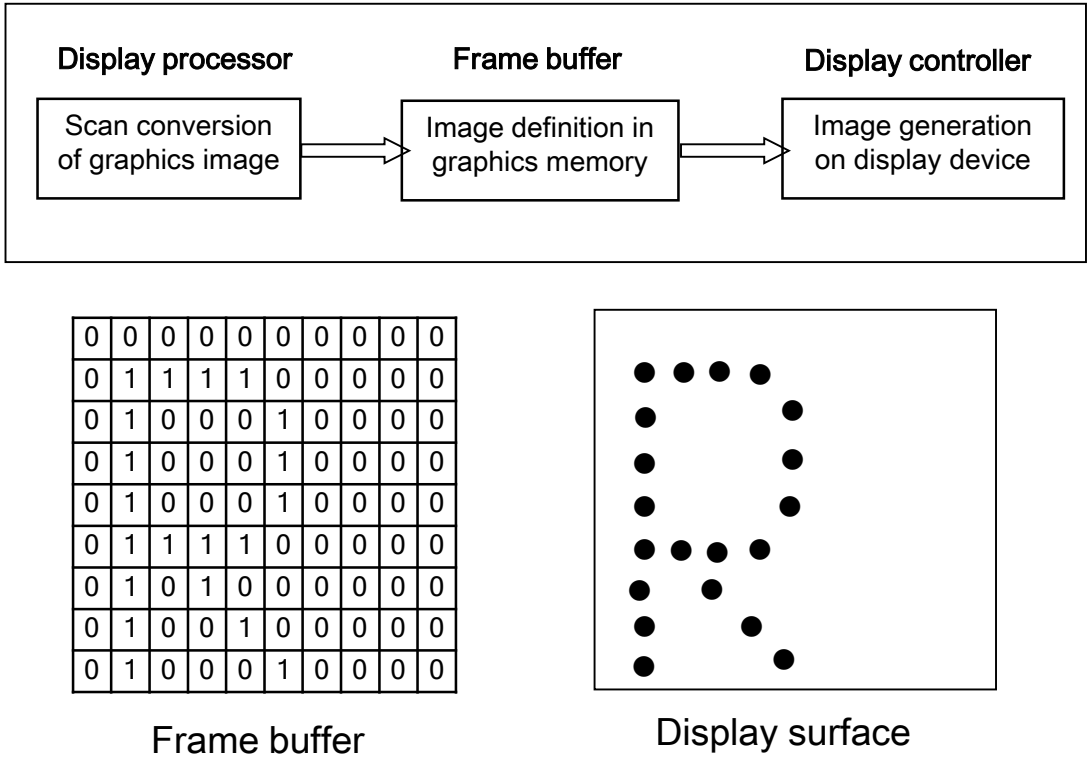
- A drawing can be displayed either with the help of pixel arrays stored into the frame buffer (graphics memory) or by the scan conversion of structural objects forming the pixel patterns.
- During scan conversion, it is required to find out those pixel locations, which are bright.
- For *monochrome* device, the image is displayed by assigning the state of bits corresponding to these pixels equal to 1 (*on* position).

IMAGE REPRESENTATIONS IN COMPUTERS…

3. Scan Conversion…

- Once an image is scan converted, a display controller scans the frame buffer and makes those pixels bright corresponding to the bits value 1 in the frame buffer.
- For example, for a line representation, it is required to determine those pixels, which are bright. This is achieved by assigning the state of bits corresponding to those pixels in the buffer memory to 1.
- Thus, scan conversion investigates those locations of pixels, which are made bright and then assigning the value of corresponding bits in the graphics memory to 1.
- Figure shows the steps of a typical scan conversion process for generating a graphics image on the display surface.
- There is *one-to-one* correspondence between the display surface and memory location in the frame buffer, designated by the coordinates pair (x, y).

IMAGE REPRESENTATIONS IN COMPUTERS…



Scan conversion process for generating graphics image

IMAGE REPRESENTATIONS IN COMPUTERS…

3. Scan Conversion…

- The coordinates (x, y) refer to the pixel locations; hence, they are integers with origin at bottom-left corner of the display devices.
- However, some graphics systems locate the *origin at the top-left corner*. The application program or device driver should take care of such difficulties.
- Every pixel on the display surface requires 1-bit memory; thus, 1024 x 1024 pixels require 1,048,576 bits memory in the frame buffer, which is equivalent to 131 kilobytes (8 bits = 1 byte).
- Each pixel can be made *bright* or *dark* by assigning the binary value 1 or 0, respectively.
- More number of bits is associated to each pixel if *colored* images, with varying intensity levels, are required.

IMAGE REPRESENTATIONS IN COMPUTERS…

3. Scan Conversion…

- For highly specialized graphics applications, the present-day technology offers graphics system with 4096 x 4096 pixels, in which each primary color (*Red, Blue or Green*) is associated with a group of 8-bit planes.

- Personal computers offer different graphics adapters with varying graphics facilities. A typical Enhanced Graphics Adapter (EGA) on personal computers gives a resolution of 640 x 350 pixels with 16 colors.

- EGA adapters with better capabilities are also available which gives very good *tonal* and *spatial* resolutions in the graphics image.

SCAN CONVERSION OF PRIMITIVE OBJECTS

- The primitive objects such as points, lines, circles, curves, surfaces, text, etc., used for the generation of *complex images* on the displays.
- Each output primitive is defined by the input coordinates and other related information about the object that is to be displayed, referred to as *attributes*.
- *Points* and *straight lines* are the simplest 2D output primitives.
- Additional outputs primitives used for constructing the complex graphics image are:
  - ***Conic sections***: Circle, ellipse, parabola and hyperbola
  - ***Quadric surfaces***: sphere, ellipsoid, paraboloids, hyperboloid
  - ***Spline curves***: Hermite curve, Bezier curve, β-spline
  - ***Spline surfaces***: Bezier surface, β-spline surface
  - ***Polygons***: Areas with linear boundaries filled with colors/patterns

SCAN CONVERSION OF POINTS

- The graphics elements such as lines, circles, surfaces, polygons, etc. consist of points. These elements are locus of points under certain constraints.
- A point has no dimensions rather represented as a dot on the paper and phosphor dot on the CRT displays.

A point in a drawing may be obtained as follows:

1. In *monochrome* (black/white) raster displays, a point plotted by setting the bit value equal to '1' (*on* position of pixel) at the specified location within the frame buffer.

   When electron beam scans across a horizontal scan line, it energizes the pixel at the location whenever a bit value equal to '1' is encountered in the frame buffer.

2. In *colored* display, different RGB bit values are loaded in the frame buffer at the specified location. During sweep, electron beam energizes the RGB dots (either as *triads* or *inline*), emitting RGB colors of varying intensities, produces the color dot.

SCAN CONVERSION OF LINES

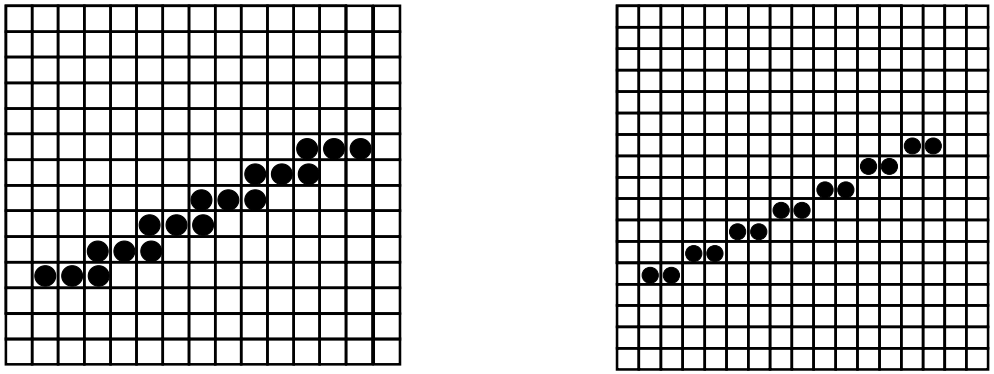
A good line drawing algorithm must satisfy the following characteristics:

- Line should appear straight on the display device
- Line should terminate accurately at the endpoints
- Line should have constant density throughout the length
- Line should be drawn accurately

- For analog device such as random scan display or pen plotter, a smooth line can be drawn between the endpoints.
- This is achieved by applying linearly varying horizontal and vertical deflection voltages to electron beam in such a way that applied voltages are proportional to the required deflection of electron beam along the x and y directions.
- For raster scan display, a line segment can be drawn by calculating the discrete coordinate points (from equation of a line) and plot them between the endpoints.

SCAN CONVERSION OF LINES…

- The *variation in intensity and color* of the line can be visualized by loading the pixel information into the frame buffer and retrieving the same by the video controller.
- In raster displays, the screen locations are always referred with *integer values*; therefore, coordinate (9.43, 20.78) would be converted into the approximate value (9, 21).
- This rounding off coordinate values to integer causes the staircase (jaggies) appearance, which is more pronounced on a low-resolution raster display devices



Line in low resolution device Line in high resolution device

Staircase effect on raster display

DDA ALGORITHM FOR LINE GENERATION

The following algorithms accomplish line drawing in computer graphics:

- Digital Differential Analyzer (DDA) algorithm
- Bresenham's line drawing algorithm

The basic input for a line generation is the endpoints. The pixel locations on the line path are calculated using these algorithms. When electron beam energizes these pixels during the raster scan, line can be visualized on the computer screen.

Digital Differential Analyzer (DDA) Algorithm

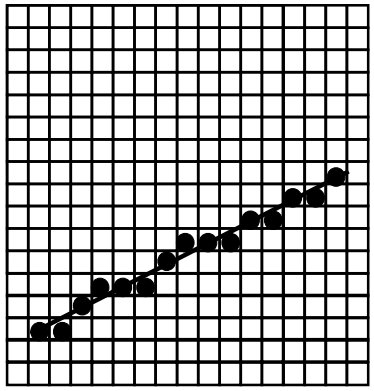
- DDA is a scan conversion line-generating algorithm, based on the calculation of either Δy or Δx differences *along the line path*.
- A line can be generated by increasing unit interval in one coordinate (say, x) and determine the corresponding integer values of interval in other coordinate (say, y) *nearest to the line path*.

DDA ALGORITHM FOR LINE GENERATION...

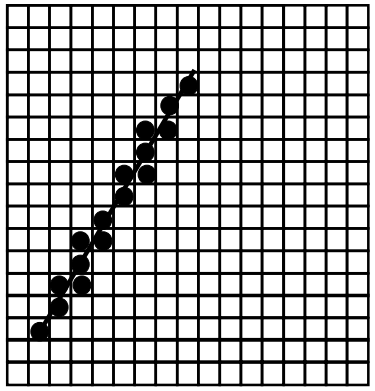
For a line, slope $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x}$

Hence, $\Delta y = m.\Delta x$ or $\Delta x = \frac{\Delta y}{m}$

There are two possibilities of slope, either $m < 1$ or $m > 1$. Figure shows two lines with the positive slopes.



slope less than 1



slope more than 1

Lines in Raster Device with positive slope

DDA ALGORITHM FOR LINE GENERATION

Case I: If $m < 1$ ***and slope is positive***

Let the line starts from left and extends to the right and increment in x coordinate is taken as unity, i.e. $\Delta x = 1$. If (x_k, y_k) is any current point on the line path, then next point will be $x_{k+1} = x_k + 1$ and $y_{k+1} = y_k + m$ because for $\Delta x = 1, \Delta y = m.\Delta x = m.1 = m$. Similarly, if line starts from right, the procedure remains same except that the new coordinate will be $x_{k+1} = x_k - 1$ and $y_{k+1} = y_k - m$.

Case II: If $m > 1$ ***and slope is positive***

Let the line starts from left and extends to the right with unit increment in y coordinate, i.e. $\Delta y = 1$. If (x_k, y_k) is any current point on the line path then the next point will be $y_{k+1} = y_k + 1$ and $x_{k+1} = x_k + (1/m)$ because for $\Delta y = 1$, $\Delta x = \Delta y / m = 1/m$. Similarly, if line starts from right, the procedure remains same except that the new coordinate will be $y_{k+1} = y_k - 1$ and $x_{k+1} = x_k - (1/m)$.

DDA ALGORITHM FOR LINE GENERATION

Case III: If $|m|<1$ and slope is negative

Let the line starts from right, the procedure given for $m<1$ (Case I) follows with the modification that as one coordinate (say, x) decreases, other coordinate (say, y) increases. Therefore, new coordinate will be $x_{k+1}=x_k-1$ and $y_{k+1}=y_k+m$.

Case IV: If $|m|>1$ and slope is negative

Let the line starts from right, the procedure given for $m>1$ (Case II) follows with the modification that as one coordinate (say, y) decreases, other coordinate (say, x) increases. Thus, new coordinate will be $y_{k+1}=y_k-1$ and $x_{k+1}=x_k+(1/m)$.

DDA ALGORITHM FOR LINE GENERATION

Limitations

Although DDA algorithm generates a straight line at a fast speed; however, it suffers from various disadvantages.

- The values of line slope m may be a real number; therefore, the values of new coordinate (x_{k+1}, y_{k+1}) on the line path may also be a real number.
- Hence, every time during the calculations, it becomes necessary to *round off* the real number coordinate values to some *integer values* because the pixels are arranged in an integer fashion.
- The real mathematics calculations for new coordinate values and rounding off operation of real numbers make DDA algorithm a little *slower process*.
- Moreover, in low-resolution displays, *staircase effect* is quite prominent; hence, DDA algorithm is *not preferred* for the CAD applications.

DDA ALGORITHM FOR LINE GENERATION

EXAMPLE: *Digitize the line with endpoints* $A(18,10)$ *and* $B(29,16)$ *using DDA algorithm.*

Solution: $\Delta x = x_B - x_A = 29\text{-}18 = 11$

$\Delta y = y_B - y_A = 16\text{-}10 = 6$

hence, slope $m = \frac{\Delta y}{\Delta x} = \frac{6}{11} = 0.54$

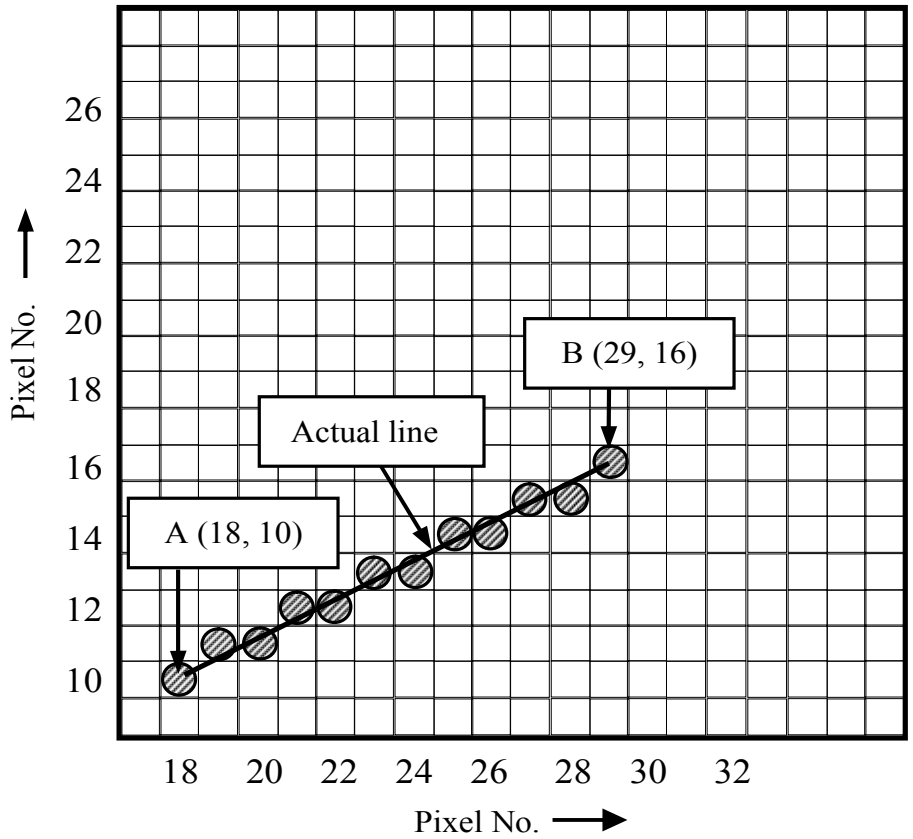
Since slope $m < 1$ and positive (Case I); therefore, new coordinates will be $x_{k+1} = x_k + 1$ and $y_{k+1} = y_k + m$. The intermediate pixels from point A to B are calculated. Table shows the intermediate pixel positions and corresponding raster line in Figure.

DDA ALGORITHM FOR LINE GENERATION

Table Intermediate pixel calculations

k	Integer values $x_{k+1} = x_k + 1$	Real values $y_{k+1} = y_k + m$	Integer values $y_{k+1} = y_k + m$
0	18	-	10
1	19	10.54	11
2	20	11.08	11
3	21	11.62	12
4	22	12.16	12
5	23	12.70	13
6	24	13.24	13
7	25	13.78	14
8	26	14.32	14
9	27	14.86	15
10	28	15.40	15
11	29	15.94	16

DDA ALGORITHM FOR LINE GENERATION



COMPUTER AIDED DESIGN
(BME-42)

Unit-II: Output Primitives

(3 Lectures)

- Scan conversion of primitives
- Line generation algorithms
- **DDA Line Drawing algorithm**
- **Bresenham's line drawing algorithm,**
- Circle generating algorithm-Cartesian coordinates, Polar coordinates
- Bresenham's circle generating algorithm

Lecture 13

Topics Covered

Scan Conversion of Line

Bresenham's Line Drawing Algorithm

Line with Positive Slope

Line with Negative Slope

Example

Prepared By

Prof. S. K. SRIVASTAVA

MED, MMMUT, Gorakhpur (UP)

sksme@mmmut.ac.in

BRESENHAM'S LINE DRAWING ALGORITHM

- The disadvantages associated with DDA algorithm such as *real number coordinate calculations* and *rounding off operation* of real numbers are eliminated in Bresenham's algorithm.

- Bresenham's developed an accurate and efficient method of *raster* generation algorithms to display lines, circles, ellipses and other curves incorporating only *incremental integer calculations*.

- Three important points must be considered in Bresenham's line generating algorithm:

  - Both frame buffer and display surface are defined in two-dimensional coordinate system with *origin at the lower left corner*. A non-negative integer (x, y) coordinate pair accesses each pixel.

BRESENHAM'S LINE DRAWING ALGORITHM…

- Line always starts from left irrespective of the input from the user; therefore, out of the two endpoints, Bresenham's algorithm decides the one, which is lying on the left as staring point. This means that x values start at the origin and *increases from left to right*, while y values start from *bottom and increases towards the top* as in standard Cartesian coordinate system.
- The line generation uses standard equation of the line $(y = mx + c)$

Line with Positive / Negative slope

- A line segment may have *positive* slope or *negative* slope.
- For sampling at unit x intervals, it is required to find out two possible pixel positions in y direction closed to the actual line.

BRESENHAM'S LINE DRAWING ALGORITHM…

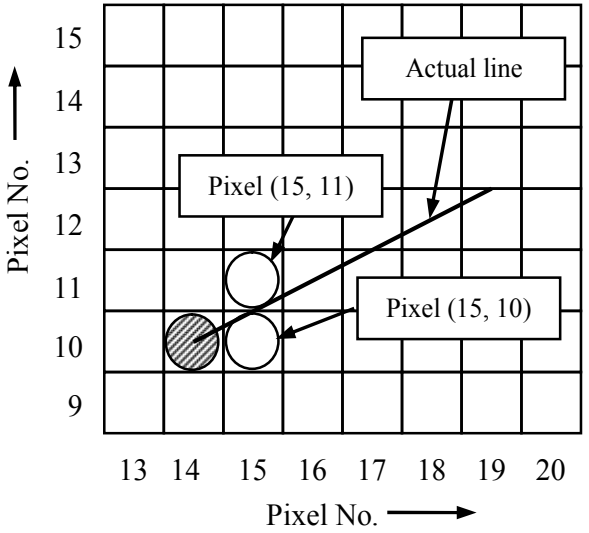
Line with Positive / Negative slope…

- In the line with positive slope, for sample position $x = 15$, whether to select the pixel coordinates positioned at (15, 10) or (15, 11).
- Similarly, for the line with negative slope starting from pixel coordinates (30, 25), for sampling position $x = 31$, whether to select next pixel position at (31, 25) or (31, 24).

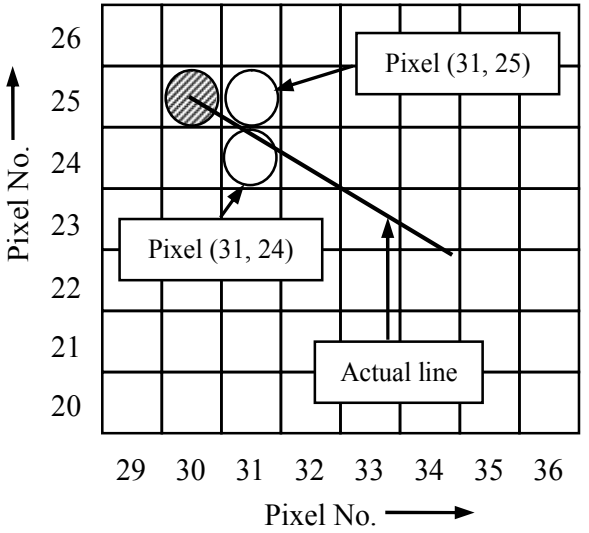
Line segment plotted with

(a) Positive slope

(b) Negative slope



(a)



(b)

BRESENHAM'S LINE DRAWING ALGORITHM…

Line with Positive / Negative slope…

- This is decided in Bresenham's algorithm by testing the sign (positive or negative) of a decision parameter that is taken to be *proportional to the difference between deviations of the two pixel positions from the actual line path*.

- In the following, at first, a line with positive slope with value less than unity (alternatively, when slope lies in between 0^o and 45^o) is considered.

- Similar analysis is carried out for a line with slope value more than unity, i.e., slope lies in between 45^o and 90^o.

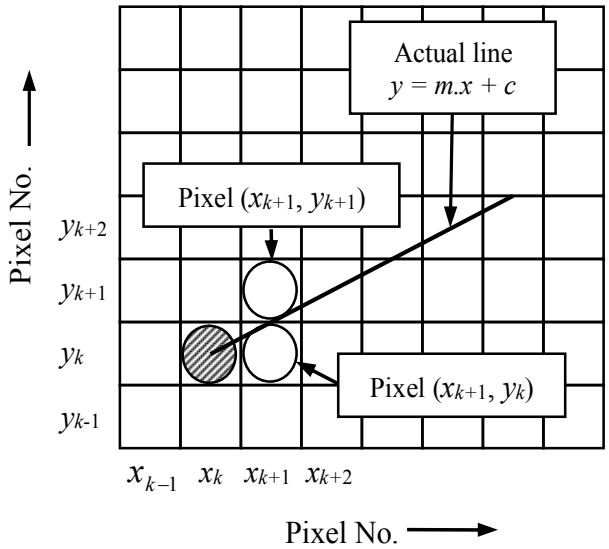
BRESENHAM'S LINE DRAWING ALGORITHM…

Line Generation with Positive slope (m < 1)

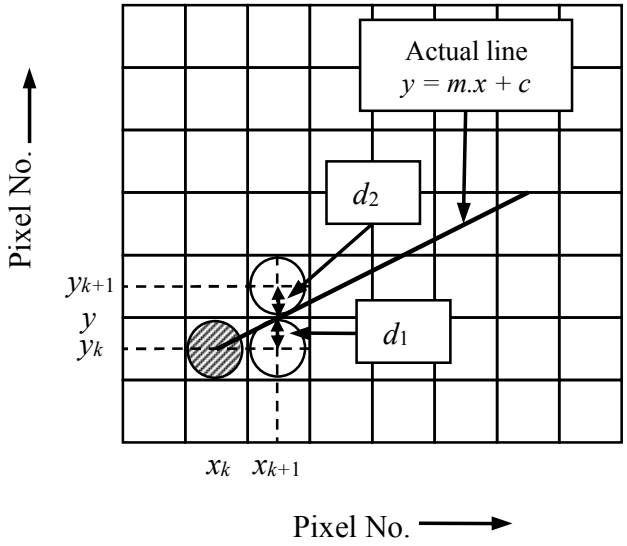
- Let the starting point of the line is (x_A, y_A)
- Suppose we have calculated the current pixel location for the line as (x_k, y_k)
- Now, this is decided that which pixel should be selected for display at the sampling column x_{k+1}
- Alternatively, we have two options for the selection of new pixel location (x_{k+1}, y_k) or (x_{k+1}, y_{k+1})
- Let, at the sampling column x_{k+1} , the vertical separation between the pixels (x_{k+1}, y_k) and (x_{k+1}, y_{k+1}), from the mathematically actual path be d_1 and d_2, respectively.
- At the sampling position x_{k+1} , y coordinate is given as $y = mx_{k+1} + c$.

BRESENHAM'S LINE DRAWING ALGORITHM…

Line Generation with Positive slope (m < 1)…



Choice of pixel in new location at sampling column x_{k+1}



Vertical separations between pixel positions y_{k+1} and y_k, from actual line y coordinate in new location at column x_{k+1}

BRESENHAM'S LINE DRAWING ALGORITHM…

Line Generation with Positive slope (m < 1)…

- Moreover, vertical separations, $d_1 = y - y_k = (mx_{k+1} + c) - y_k$

$$d_2 = y_{k+1} - y = y_{k+1} - (mx_{k+1} + c)$$

- Therefore, $d_1 - d_2 = \{(mx_{k+1} + c) - y_k\} - \{y_{k+1} - (mx_{k+1} + c)\}$

$$= 2mx_{k+1} - y_k - y_{k+1} + 2c$$

$$= 2mx_{k+1} - 2y_k + 2c - 1 \qquad (\text{because } y_{k+1} = y_k + 1)$$

- A decision parameter P_k for the k^{th} step in line drawing algorithm calculated by rearranging the terms so that it incorporates only integer calculations for determining the new pixel locations.
- Let, $\Delta x = x_B - x_A$ and $\Delta y = y_B - y_A$ are the horizontal and vertical separations between the endpoint positions A and B, and slope of the line $m = \Delta y / \Delta x$

BRESENHAM'S LINE DRAWING ALGORITHM…

Line Generation with Positive slope (m < 1)…

- Further, a decision parameter P_k defined as

$$\begin{aligned} P_k &= \Delta x.(d_1 - d_2) \\ &= \Delta x.(2.m.x_{k+1} - 2.y_k + 2.c - 1) \\ &= 2.\Delta y.(x_k + 1) - 2.\Delta x.y_k + 2.c.\Delta x - \Delta x \qquad \text{(because } x_{k+1} = x_k + 1\text{)} \\ &= 2.\Delta y.x_k - 2.\Delta x.y_k + \{2.\Delta y + \Delta x.(2c - 1)\} \end{aligned}$$

or $$P_k = 2.\Delta y.x_k - 2.\Delta x.y_k + b$$

where $$b = 2.\Delta y + \Delta x.(2c - 1)$$

For positive slope ($\Delta x > 0$), the sign of decision parameter P_k will be same as the sign of $(d_1 - d_2)$, and parameter b is independent of the pixel position.

In fact, the parameter b depends upon the incremental values (Δx and Δy), which eliminates during the recursive calculations of decision parameter P_k.

BRESENHAM'S LINE DRAWING ALGORITHM…

Line Generation with Positive slope (m < 1)…

It is obvious that at new pixel position x_{k+1}, if pixel at y_k position is closer to the actual line path (alternatively, $d_1 < d_2$) than the pixel at y_{k+1} location, then decision parameter P_k will be negative and lower pixel y_k will be selected; otherwise, the upper pixel y_{k+1} should be plotted.

- For recursive calculations, at step $k+1$, the decision parameter evaluated as

$$P_{k+1} = 2.\Delta y.x_{k+1} - 2.\Delta x.y_{k+1} + b$$

Thus, $P_{k+1} - P_k = \{2.\Delta y.x_{k+1} - 2.\Delta x.y_{k+1} + b\} - \{2.\Delta y.x_k - 2.\Delta x.y_k + b\}$

$$= 2.\Delta y.(x_k + 1) - 2.\Delta x.y_{k+1} - 2.\Delta y.x_k + 2.\Delta x.y_k \quad (\text{because } x_{k+1} = x_k + 1)$$

or $$P_{k+1} = P_k + 2.\Delta y - 2.\Delta x.(y_{k+1} - y_k)$$

BRESENHAM'S LINE DRAWING ALGORITHM…

Line Generation with Positive slope (m < 1)…

- Depending upon sign of the decision parameter P_k either $y_{k+1} - y_k = 0$ or 1
- When parameter P_k is negative (alternatively, $d_1 < d_2$), $y_{k+1} - y_k = 0$ or $y_{k+1} = y_k$; therefore, pixel at y_k should be brightened.
- However, if P_k is positive (alternatively, $d_1 > d_2$), $y_{k+1} - y_k = 1$ or $y_{k+1} = y_k + 1$; therefore, pixel y_{k+1} becomes closer to the actual line path; hence, this should be energized.

The recursive calculations of decision parameter P_k determine each integer x pixel position, starting from left coordinate endpoint (x_A, y_A) of the line AB. The initial value of decision parameter P_0 may be evaluated for the starting pixel position (x_A, y_A), with the assumption that line passes through the origin (alternatively, $x_k = y_k = 0$ and $c = 0$) as

$$P_0 = 2.\Delta y.0 - 2.\Delta x.0 + \{2.\Delta y + \Delta x.(2.0 - 1)\} = 2.\Delta y - \Delta x$$

BRESENHAM'S LINE DRAWING ALGORITHM…

Line Generation Algorithm when Slope m < 1 (i.e., slope is in between 0^0 and 45^0

Step 1: Input the line endpoints AB as (x_A, y_A) and (x_B, y_B), respectively.

Step 2: Plot the first endpoint (x_A, y_A), i.e., load the point into the frame buffer.

Step 3: Calculate the initial value of decision parameter P_k, i.e. P_0

$$\Delta x = x_B - x_A \,;\, \Delta y = y_B - y_A \text{ and } P_0 = 2.\Delta y - \Delta x$$

Step 4: At each x_k position, starting from $k = 0$, perform the following test:

If $P_k < 1$ (i.e., negative), the next position to plot the pixel is $(x_k + 1, y_k)$ and $P_{k+1} = P_k + 2.\Delta y$

Otherwise ($P_k > 1$, i.e., positive), the next pixel position is $(x_k + 1, y_k + 1)$

and $P_{k+1} = P_k + 2.\Delta y - 2.\Delta x$

Step 5: Plot a point of current (x, y) position.

Step 6: Repeat Step 4 through Step 5, Δx times until reach the second endpoint, i.e. $x \geq x_B$.

BRESENHAM'S LINE DRAWING ALGORITHM…

Line Generation Algorithm when Slope m > 1 (i.e., slope is in between 45^0 and 90^0

The procedure for line generation algorithm is same except that x and y coordinates are interchanged.

Step 1: Input the line endpoints AB as (x_A, y_A) and (x_B, y_B), respectively.

Step 2: Plot the first endpoint (x_A, y_A), i.e., load the point into the frame buffer.

Step 3: Calculate the initial value of the decision parameter P_k, i.e. P_0

$$\Delta x = x_B - x_A \; ; \Delta y = y_B - y_A \text{ and } P_0 = 2.\Delta x - \Delta y$$

Step 4: At each x_k position, starting from $k = 0$, perform the following test:

If $P_k < 1$ (i.e., negative), the next position to plot the pixel is $(x_k, y_k + 1)$ and $P_{k+1} = P_k + 2.\Delta x$

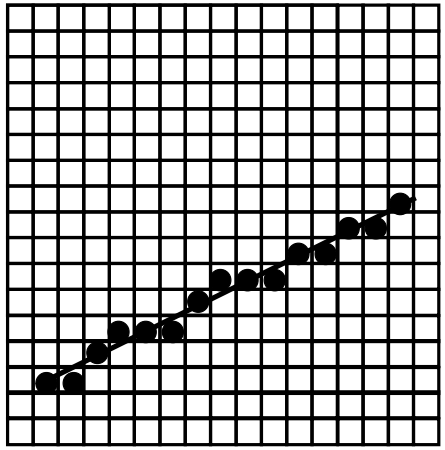
Otherwise ($P_k > 1$, i.e., positive), the next pixel position is $(x_k + 1, y_k + 1)$

and $P_{k+1} = P_k + 2.\Delta x - 2.\Delta y$

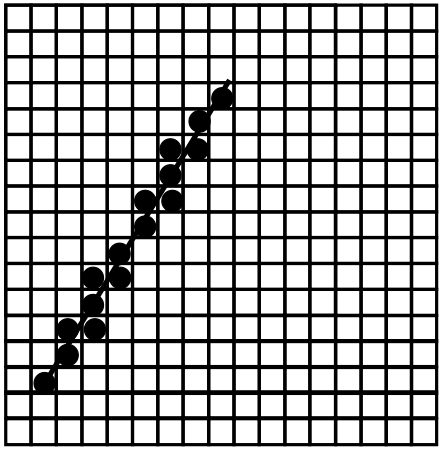
Step 5: Plot a point of current (x, y) position.

Step 6: Repeat Step 4 through Step 5, Δy times until reach the second endpoint, i.e. $y \geq y_B$.

BRESENHAM'S LINE DRAWING ALGORITHM…



slope less than 1



slope more than 1

Lines in Raster Device with positive slope

Line Generation Algorithm when slope m is negative

If slope of the line is negative, the procedure for line generation is same except that one coordinates decreases as other increases.

BRESENHAM'S LINE DRAWING ALGORITHM…

Example: *Digitize the line with endpoints* $A(18,8)$ *and* $B(28,16)$ *using Bresenham's line drawing algorithm and plot the pixel positions.*

Solution: $\Delta x = x_B - x_A = 28\text{-}18 = 10$

$\Delta y = y_B - y_A = 16 - 8 = 8$

hence, slope $m = \frac{\Delta y}{\Delta x} = \frac{8}{10} = 0.80$

Since slope $m < 1$ and positive; therefore, as x pixel position increases, y location also increases. Initial value of the decision parameter, $P_0 = 2.\Delta y - \Delta x$

Successive values of the decision parameter, $P_{k+1} = P_k + 2.\Delta y - 2.\Delta x.(y_{k+1} - y_k)$

If $P_k < 1$ (i.e., negative), the next pixel position to plot is $(x_k + 1, y_k)$ and $P_{k+1} = P_k + 2.\Delta y$

Otherwise ($P_k > 1$, i.e., positive), the next pixel position is $(x_k + 1, y_k + 1)$ and $P_{k+1} = P_k + 2.\Delta y - 2.\Delta x$

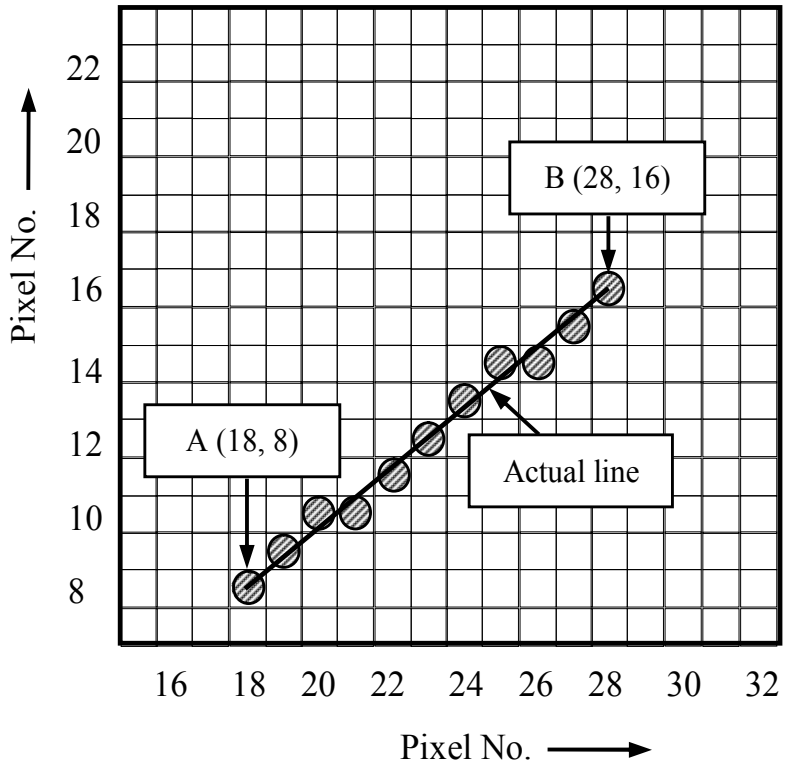
Table shows successive pixel positions until second endpoint $B(28,16)$ reaches. Figure shows the plot of pixel positions representing the line AB.

BRESENHAM'S LINE DRAWING ALGORITHM…

Table *Intermediate pixel calculations*

k	Decision parameter P_k	Value of decision parameter P_k	Next pixel (x_{k+1}, y_{k+1})	Value of next pixel (x_{k+1}, y_{k+1})
0	$P_0 = 2.\Delta y - \Delta x$	2.8 - 10 = 6	$(x_k + 1, y_k + 1)$	(19, 9)
1	$P_1 = P_0 + 2.\Delta y - 2.\Delta x$	6 + 2.8 - 2.10 = 2	$(x_k + 1, y_k + 1)$	(20, 10)
2	$P_2 = P_1 + 2.\Delta y - 2.\Delta x$	2 + 2.8 – 2.10 = -2	$(x_k + 1, y_k)$	(21, 10)
3	$P_3 = P_2 + 2.\Delta y$	-2 + 2.8 = 14	$(x_k + 1, y_k + 1)$	(22, 11)
4	$P_4 = P_3 + 2.\Delta y - 2.\Delta x$	14 + 2.8 – 2.10 = 10	$(x_k + 1, y_k + 1)$	(23, 12)
5	$P_5 = P_4 + 2.\Delta y - 2.\Delta x$	10 + 2.8 – 2.10 = 6	$(x_k + 1, y_k + 1)$	(24, 13)
6	$P_6 = P_5 + 2.\Delta y - 2.\Delta x$	6 + 2.8 – 2.10 = 2	$(x_k + 1, y_k + 1)$	(25, 14)
7	$P_7 = P_6 + 2.\Delta y - 2.\Delta x$	2 + 2.8 – 2.10 = -2	$(x_k + 1, y_k)$	(26, 14)
8	$P_8 = P_7 + 2.\Delta y$	-2 + 2.8 = 14	$(x_k + 1, y_k + 1)$	(27, 15)
9	$P_9 = P_8 + 2.\Delta y - 2.\Delta x$	14 + 2.8 – 2.10 = 10	$(x_k + 1, y_k + 1)$	(28, 16)

BRESENHAM'S LINE DRAWING ALGORITHM…



COMPUTER AIDED DESIGN
(BME-42)

Unit-II: Output Primitives

(3 Lectures)

- Scan conversion of primitives
- Line generation algorithms
- DDA Line Drawing algorithm
- Bresenham's line drawing algorithm
- **Circle generating algorithm- Cartesian coordinates, Polar coordinates**
- **Bresenham's circle generating algorithm**

Lecture 14

Topics Covered

Scan Conversion of Circle

Circle Generation using Cartesian Coordinates

Circle Generation using Polar Coordinates

Bresenham's Circle Generation Algorithm

Example

Prepared By

Prof. S. K. SRIVASTAVA

MED, MMMUT, Gorakhpur (UP)

sksme@mmmut.ac.in

SCAN CONVERSION OF CIRCLE

In computer graphics, the construction of 2D images frequently requires circles and arcs as primitives. A circle generation requires *centre* and *radius* as the basic input parameters. In circle generating algorithm, the pixel positions on the circular path are calculated. There are two types of circle generating algorithms:

I. **Based on conventional equation of circle in Cartesian or Polar coordinates**

II. **Using Bresenham's circle generating algorithm**

Circle Generation using Cartesian Coordinates

If (x_c, y_c) is the centre position and r is the radius of circle then equation of a circle in non-parametric form is

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

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

SCAN CONVERSION OF CIRCLE…

Circle Generation using Cartesian Coordinates…

Equation generates the points on the path of circle where x coordinate varies from $(x_c - r)$ to $(x_c + r)$ in a steps of unity, and corresponding y values are determined and plotted as shown in Figure. This method suffers from the following disadvantages; hence, not preferred for CAD applications.

(i). Equation uses square root calculations; therefore, circle generation becomes a slow process.

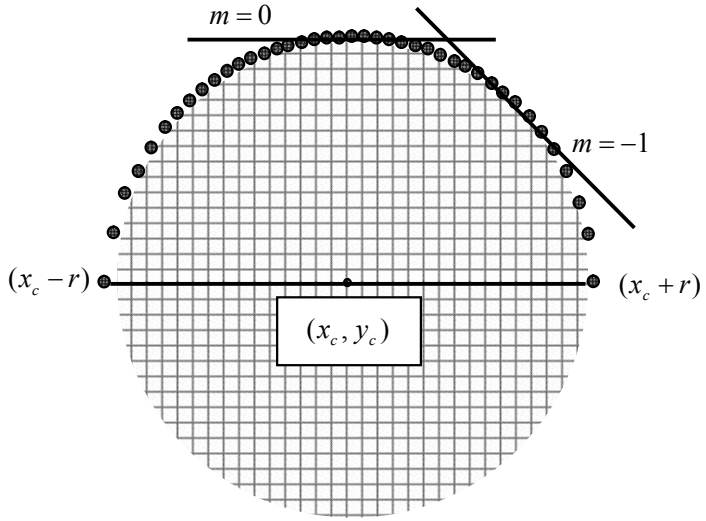
(ii). Spacing between the pixels in y direction is not uniform. For $0 < m < 1$, the pixels are very close whereas for the region $m > 1$, the pixels are farther apart. Although, the results can be improved in the region $m > 1$ by interchanging x and y coordinates (unit increase in y values and calculating x values); however, this increases the complexity of computation and processing time required.

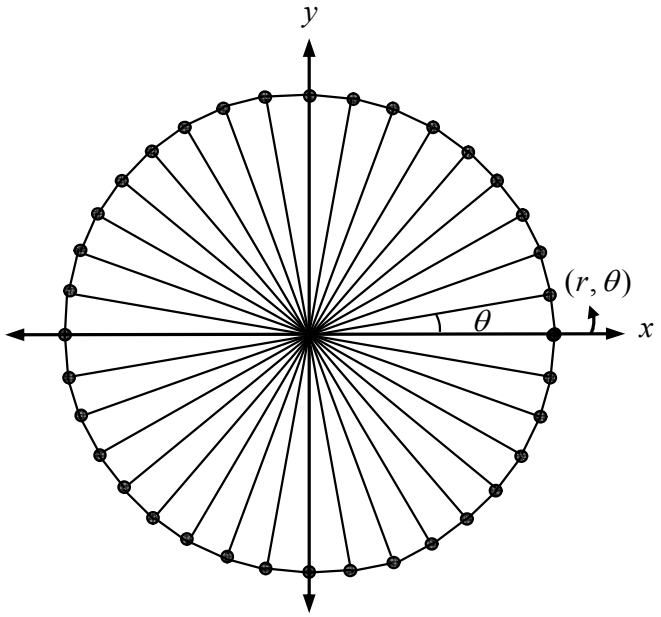
(iii). It is necessary to round off the value of y coordinate (or x coordinate) to the next integer value; therefore, this affects the accuracy of generated circle.

(iv). Results into poor visualization of circle on the display surface.

SCAN CONVERSION OF CIRCLE…

Circle Generation using Cartesian Coordinates…





Circle Generation using Polar Coordinates

The unequal spacing in x and y coordinates can be eliminated by using the Polar coordinates system where pixel positions along the circular path are determined in (r, θ) coordinates. The parametric equation of circle in Polar coordinates may be expressed as

$$x = x_c + r\cos\theta; \quad y = y_c + r\sin\theta$$

SCAN CONVERSION OF CIRCLE…

Circle Generation using Polar Coordinates…

where (x_c, y_c) and r are the centre and radius of the circle, respectively. With origin as a centre and radius equal to unity, the eqn. modified as

$$x = \cos\theta \; ; \; y = \sin\theta$$

The radius vector r makes an angle θ with the x-axis.

- Thus, using a fixed angular step, a circle may be plotted with equally spaced points along the circular path.
- Angular step θ depends on the application and pixel density of the display devices.
- Larger separations between the points, connected with *straight-line segments* to approximate the circumference of circle as shown in Figure.
- For a more continuous circumference, a step size of $1 / r$ unit may be set up on raster displays with pixel positions approximately one unit apart.

SCAN CONVERSION OF CIRCLE…

Circle Generation using Polar Coordinates…

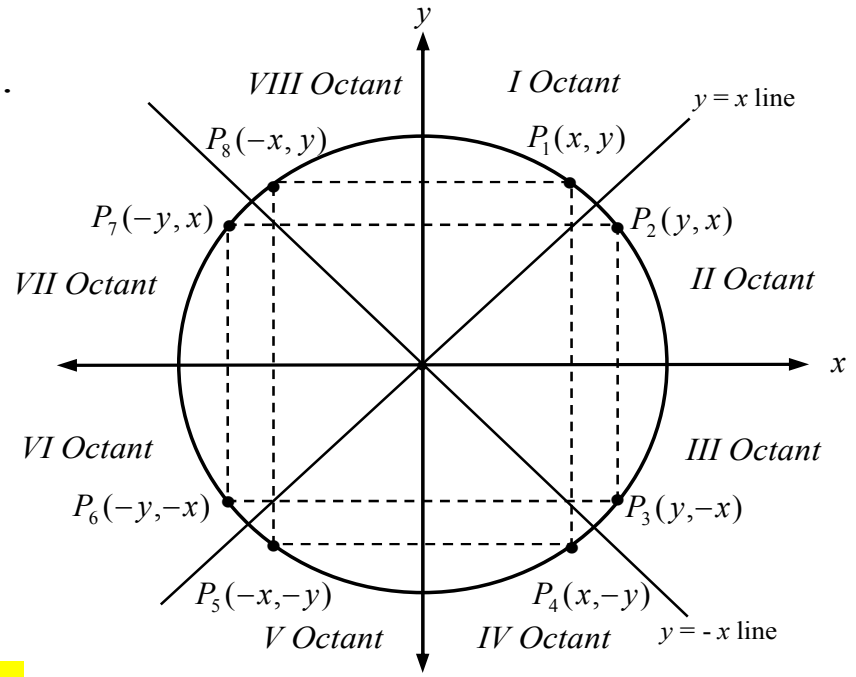
- This algorithm uses trigonometric functions; therefore, *approximate pixel positions*, are calculated in the form of series expansion of trigonometric functions.
- Moreover, this requires *rounding off*, the x and y coordinate values, to the next higher integer.
- Although, the quality of circle is much better than the same obtained using the Cartesian coordinates but algorithm for circle generation is still a slower process; therefore, this method of circle generation is also not preferred for CAD applications.

BRESENHAM'S CIRCLE GENERATION ALGORITHM

- Bresenham's algorithm eliminates the square root calculations and rounding off operations during the pixel calculations. Thus, the accuracy of circle is affected.
- Bresenham's circle generating algorithm uses the *integer calculations*.
- This is the method for direct distance comparison, which tests the midpoint location between the two successive pixels to find whether the midpoint lies inside or outside the circle boundary.
- This is also called *midpoint circle generating algorithm*. Bresenham's algorithm extensively uses symmetry of the circle.
- This is fact that if $\frac{1}{8}$ part of the circle is generated then rest $\frac{7}{8}$ part of the circle can be generated using the symmetry, as shown in Figure.

BRESENHAM'S CIRCLE GENERATION ALGORITHM

- If (x, y) coordinates of a point P_1 in the first octant is calculated, it is easy to find out the remaining seven points in the other seven octants such as $P_2(y,x)$, $P_3(y,-x)$, $P_4(x,-y)$, $P_5(-x,-y)$, $P_6(-y,-x)$, $P_7(-y,x)$ and $P_8(-x, y)$
- Therefore, in Bresenham's midpoint circle generating algorithm, it is required to find out pixel positions only in the $\frac{1}{8}$ region of part of the circle (i.e., one octant), and hence, development of software is very easy.
- Moreover, the circle draws at a faster rate.



Eight-way symmetry of a circle

BRESENHAM'S CIRCLE GENERATION ALGORITHM

- In this method, at first, the topmost point, i.e., (0, r) is plotted if equation of circle is

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

- However, for circle with centre position (x_c, y_c) and radius r, the algorithm can be developed for the calculation of pixel positions around the circle path, centered at origin (0, 0)

- Thereafter, each evaluated pixel position (x, y) is shifted to its proper screen position by adding x_c to x and y_c to y along circle section in the first octant (from $x = 0$ line to $x = y$ line; alternatively, when slope varies from $m = 0$ to -1)

- Therefore, we can go in the positive x-direction by unit steps over the first octant. When x coordinate of a point increases, corresponding y coordinate decreases.

- Similar to other Bresenham's algorithms, a decision parameter P_k determines the two possible pixel positions, which is closer to the actual circle boundary at each step in the first octant.

BRESENHAM'S CIRCLE GENERATION ALGORITHM

- Pixel positions in the other seven octants are determined using the symmetry of circle.
- Let circle function itself is the decision parameter in the Bresenham's algorithm. Thus, the equation of circle in implicit form is

$$P_k = f_{circle}(x,y) = x^2 + y^2 - r^2 \begin{cases} < 0, \text{ if point } (x,y) \text{ lies inside the circle boundary} \\ = 0, \text{ if point } (x,y) \text{ lies on the circle boundary} \\ > 0, \text{ if point } (x,y) \text{ lies outside the circle boundary} \end{cases}$$

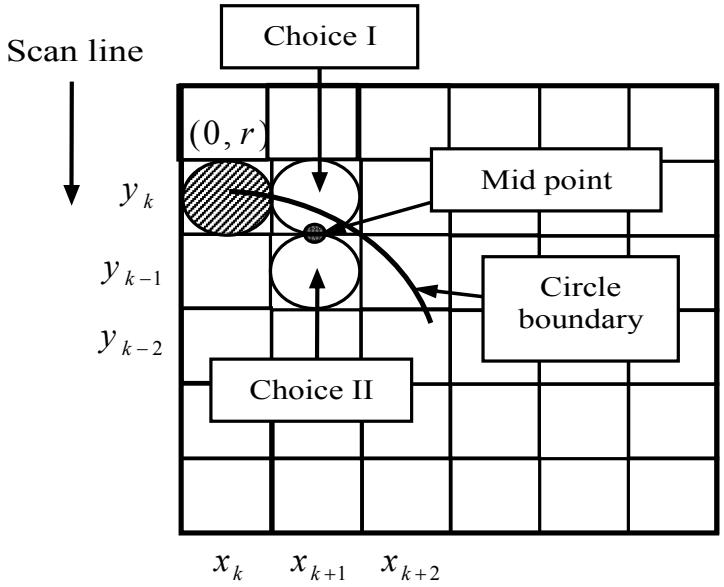
- Figure shows the plot for the current pixel coordinates (x_k, y_k)
- Now, it is to decide that at the sampling position x_{k+1} whether the pixel y_k or one at y_{k-1} is closer to the circle boundary.

BRESENHAM'S CIRCLE GENERATION ALGORITHM

- Alternatively, at position x_{k+1}, we have to select either pixel (x_{k+1}, y_k) as choice I or pixel position (x_{k+1}, y_{k-1}) as choice II, depending upon position of the midpoint between two adjacent pixels y_k and y_{k-1} from the circle boundary.
- The decision parameter P_k at the midpoint between these two pixels will be

$$P_k = f_{circle}(x_{k+1}, \frac{y_k + y_{k-1}}{2}) = f_{circle}(x_k + 1, y_k - \tfrac{1}{2}) = (x_k + 1)^2 + (y_k - \tfrac{1}{2})^2 - r^2$$

Midpoint at sample position x_{k+1} between the candidate's pixels y_k and y_{k-1} along the circular path



BRESENHAM'S CIRCLE GENERATION ALGORITHM

Now, there are two possibilities for the decision parameter P_k :

(i). If $P_k < 0$, the midpoint lies inside the circle; alternatively, pixel on the scan line y_k is closer to the circle boundary and pixel choice will be (x_{k+1}, y_k), i.e., choice I.

(ii). If $P_k \geq 0$, the midpoint lies outside or on the circle boundary; alternatively, pixel on the scan line y_{k-1} becomes closer to the circle boundary and pixel choice will be (x_{k+1}, y_{k-1}), i.e., choice II.

- The recursive formula for successive values of the decision parameter P_k may be developed by using the incremental calculations for the next pixel position at $x_{k+1}+1 = x_k + 2$

,

BRESENHAM'S CIRCLE GENERATION ALGORITHM

- Thus, recursive expression P_{k+1} for may be written as

$$P_{k+1} = f_{circle}(x_{k+1}+1, y_{k+1}-\tfrac{1}{2})$$

$$= \{(x_k+1)^2+1\}^2 + \{y_{k+1}-\tfrac{1}{2}\}^2 - r^2$$

$$= (x_k+1)^2 + 1 + 2.(x_k+1) + y_{k+1}^2 + \tfrac{1}{4} - y_{k+1} - r^2$$

$$= \{(x_k+1)^2 + (y_k^2 + \tfrac{1}{4} - y_k) - r^2\} + 1 + 2.(x_k+1) + y_{k+1}^2 - y_{k+1} - y_k^2 + y_k$$

$$= \{(x_k+1)^2 + (y_k^2 + \tfrac{1}{4} - y_k) - r^2\} + 1 + 2.(x_k+1) + y_{k+1}^2 - y_{k+1} - y_k^2 + y_k$$

or $$P_{k+1} = P_k + 2.(x_k+1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

where $y_{k+1} = y_k$ (if $P_k < 0$, i.e., negative) when pixel (x_{k+1}, y_k) lies inside the circle boundary;

therefore, $$P_{k+1} = P_k + 2x_{k+1} + 1$$

BRESENHAM'S CIRCLE GENERATION ALGORITHM

and $y_{k+1} = y_{k-1}$ (if $P_k \geq 0$, i.e., positive) when pixel (x_{k+1}, y_{k-1}) lies outside the circle boundary; therefore, $y_{k+1} = y_{k-1} = y_k - 1$ gives $y_{k+1} - y_k = -1$ and $y_{k+1}^2 - y_k^2 = 1 - 2y_k$

moreover, the decision parameter may be written as

$$P_{k+1} = P_k + 2.x_{k+1} + 1 - 2.(y_k - 1)$$

$$= P_k + 2.x_{k+1} + 1 - 2.y_{k+1} \qquad \text{(because } y_{k+1} = y_k - 1\text{)}$$

The initial value of decision parameter at topmost point $(0, r)$, calculated as

$$P_0 = f_{circle}(1, r - \tfrac{1}{2}) = 1 + (r - \tfrac{1}{2})^2 - r^2 = \tfrac{5}{4} - r$$

When radius of the circle is specified as integer value, P_0 may be modified as

$$P_0 \cong 1 - r$$

BRESENHAM'S CIRCLE GENERATION ALGORITHM

Step 1: Input radius r and centre (x_c, y_c) and plot the first point on the circumference of circle, centered on origin as $(x_0, y_0) = (0, r)$.

Step 2: Determine the initial value of decision parameter as $P_0 = 1 - r$.

Step 3: Calculate the location of next pixel position along the circular path centered at the origin (0,0)

If $P_k < 0$ (i.e., negative), $x_{k+1} = x_k + 1$, $y_{k+1} = y_k$ and $P_{k+1} = P_k + 2.x_{k+1} + 1$

Otherwise ($P_k \geq 0$, i.e., positive), $x_{k+1} = x_k + 1$, $y_{k+1} = y_k - 1$ and $P_{k+1} = P_k + 2.x_{k+1} + 1 - 2.y_{k+1}$

Step 4: Determine symmetry points in other seven octants.

Step 5: Move each calculated pixel position (x, y) on to the circular path centered at (x_c, y_c) and plot the coordinate values as $x = x + x_c$ and $y = y + y_c$.

Step 6: Repeat step 3 through step 5 until $x \geq y$.

BRESENHAM'S CIRCLE GENERATION ALGORITHM

Example: Using midpoint Bresenham's circle generating algorithm, determine pixel positions along circle octant in the first quadrant from line $x = 0$ to $x = y$. The radius of circle is 10 units. Plot the generated pixel positions.

Solution: In first quadrant from line $x = 0$ to $x = y$, as x pixel position increases y position decreases.

Initial value of decision parameter $P_0 = 1 - r = 1\text{-}10 = \text{-}9$

For circle centered at origin, the initial point is $(x_0, y_0) = (0, 10)$

If $P_k < 1$ (i.e., negative), the next pixel position to plot is $(x_k + 1, y_k)$ and $P_{k+1} = P_k + 2.x_{k+1} + 1$

Otherwise ($P_k > 1$, i.e. positive), the next pixel position is $(x_k + 1, y_k - 1)$ and $P_{k+1} = P_k + 2.x_{k+1} + 1 - 2.y_{k+1}$

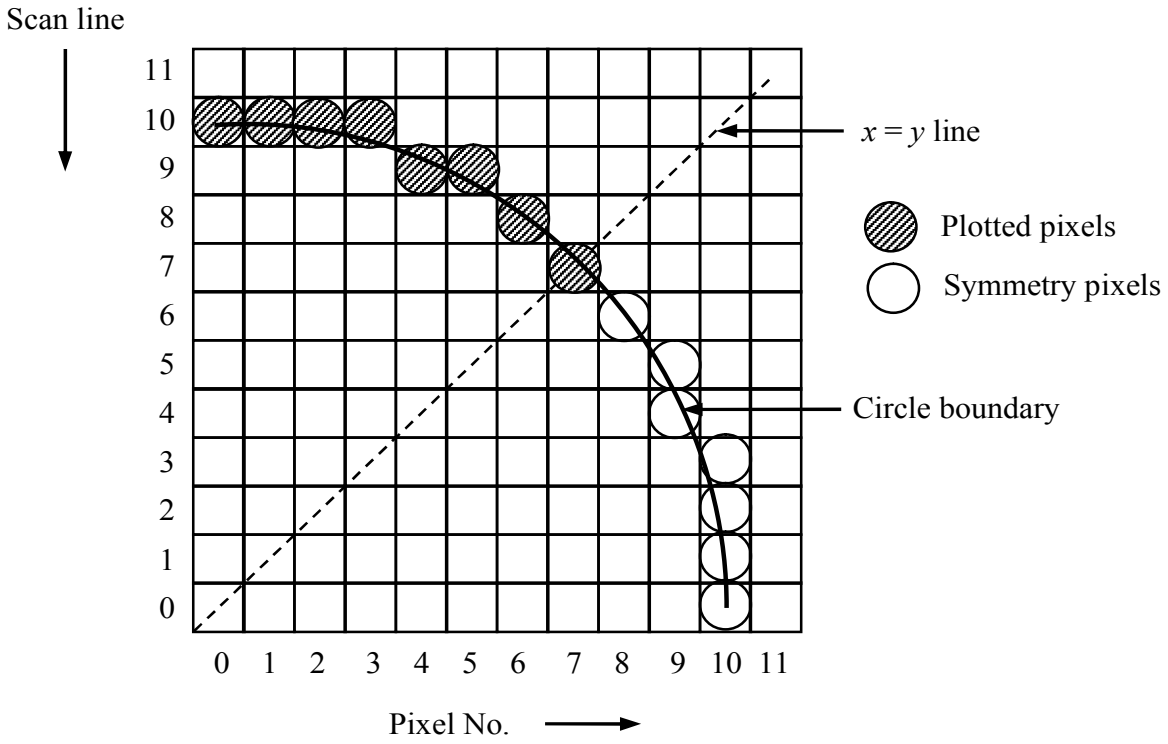
Table shows the successive pixel positions until $x \geq y$ reaches. Figure shows the plot of selected pixel positions along a circle path in first octant, where shadow pixels are plotted pixels and transparent pixels show the corresponding symmetry positions in the second octant.

BRESENHAM'S CIRCLE GENERATION ALGORITHM

Table: Intermediate Pixel calculations

k	Decision parameter P_k	Value of decision parameter P_k	Next pixel (x_{k+1}, y_{k+1})	Value of next pixel (x_{k+1}, y_{k+1})
0	$P_0 = 1 - r$	1-10 = -9	$(x_k + 1, y_k)$	(1, 10)
1	$P_1 = P_0 + 2.x_{k+1} + 1$	-9 + 2.1 + 1 = -6	$(x_k + 1, y_k)$	(2, 10)
2	$P_2 = P_1 + 2.x_{k+1} + 1$	-6 + 2.2 + 1 = -1	$(x_k + 1, y_k)$	(3, 10)
3	$P_3 = P_2 + 2.x_{k+1} + 1$	-1 + 2.3 + 1 = 6	$(x_k + 1, y_k - 1)$	(4, 9)
4	$P_4 = P_3 + 2.x_{k+1} + 1 - 2.y_{k+1}$	6 + 2.4 + 1 – 2.9 = -3	$(x_k + 1, y_k)$	(5, 9)
5	$P_5 = P_4 + 2.x_{k+1} + 1$	-3 + 2.5 +1 = 8	$(x_k + 1, y_k - 1)$	(6, 8)
6	$P_6 = P_5 + 2.x_{k+1} + 1 - 2.y_{k+1}$	8 + 2.6 + 1 – 2.8 = 5	$(x_k + 1, y_k - 1)$	(7, 7)

BRESENHAM'S CIRCLE GENERATION ALGORITHM



Pixel plotted along the circular path using Bresenham's mid point circle generating algorithm

COMPUTER AIDED DESIGN
(BME-42)

Unit-II: Geometric Transformations
(3 Lectures)

- **2D Geometric transformations- Translation, Scaling, Rotation**, Reflection & Shear
- Matrix representation
- homogeneous coordinates
- Rotation and scaling about arbitrary point
- Reflection through arbitrary line
- Composite transformation
- 3 D transformations
- multiple transformation

Lecture 15

Topics Covered

2D Geometric Transformations

Transformation of Geometric Models

Basic Transformations

Translation, Rotation, Scaling

Prepared By

Prof. S. K. SRIVASTAVA

MED, MMMUT, Gorakhpur (UP)
sksme@mmmut.ac.in

2D GEOMETRIC TRANSFORMATIONS

- The output primitives in CAD applications can be used to create variety of pictures, graphs and objects in computer graphics by rearranging the components of varying sizes, shapes and at different orientations.
- The animations are performed by moving the camera/eye or the objects along the animation paths.
- The manipulation in sizes, shapes and orientations are accomplished with the help of geometric transformations.
- Thus, geometric transformations are used to modify the images on the display devices and to reposition the objects in the database by altering its coordinate descriptions.

A set of operations on coordinate system (i.e., object) that results in a change in the position of the coordinate system is known as geometric transformation.

- Geometric transformations play a central role in the modeling and viewing an object on the display devices.

2D GEOMETRIC TRANSFORMATIONS

- Typical CAD commands to translate, rotate, distort, zoom and mirror objects are based on the geometric transformations.
- Different views of the object can be seen onto the proper projection plane by applying various types of *parallel* (orthographic, etc.) and *non-parallel* (perspective) projections.
- Geometric transformations can be used to create *animation* of geometric models to study their motion characteristics.
- For example, the motion of spatial mechanism can be animated by first calculating its motion characteristics (viz., displacement/rotations) using kinematics and dynamics equations.
- Thereafter, different mechanisms are constructed and transformed incrementally, using the motion characteristics.

2D GEOMETRIC TRANSFORMATIONS

- Consequently, the *animation (continuous motion)* of mechanisms results when grouped and redisplayed on the time scale.
- Similarly, geometric transformations help in visualizing the animation effects such as *vibrations*, *deformations*, etc. on the object.
- The display or transformation of a given entity requires the transformations of its key points (*coordinates*).
- When the viewpoint changes rapidly (alternatively, when object moves very fast), the transformations of key points must be carried out rapidly and repeatedly.
- Thus, it is required to find out efficient ways to perform 2D or 3D geometric transformations.

TRANSFORMATIONS OF GEOMETRIC MODELS

- In fact, geometric transformations are the *mappings from one coordinate system to itself*. Alternatively, the model of an object changes within its own device coordinate system.

- The simplest motion of the object is rigid body motion in which the relative distances between the object particles (or coordinates) remain constant; alternatively, the object does not deform during the motion.

- Geometric transformations that describe the rigid body transformations include *translation*, *rotation*, *scaling* and *reflection*.

- In *shear*, the distortion causes sliding of the internal layers over the others.

- These transformations are applied directly to the parametric representations of objects such as points, curves, surfaces and solids.

TRANSFORMATIONS OF GEOMETRIC MODELS

- A point is a basic entity for the object representation. For example, a line is represented by its endpoints.
- Similarly, curves, surfaces and solids are the collection of several points.
- The geometric transformation of a given point $P(x, y, z)$ of a geometric model to the corresponding new point $P_T(x_T, y_T, z_T)$ is given as

$$P_T = f\ (P, \text{transformation parameters})$$

- Thus, transformed coordinates P_T is a function of original coordinates P and the motion parameters corresponding to the given geometric transformation.
- However, if an object consists of several points, all points of the object must transform corresponding to the given geometric transformation.

BASIC TRANSFORMATIONS

- *Translation, rotation and scaling are termed three basic geometric transformations*.
- These transformations are used to reposition and resize two-dimensional objects on the displays (alternatively, in the database).
- In this section, general procedure for applying these transformations to a 2D objects is described.

1. Translation

- Whenever an entity of geometric model remains parallel to its initial positions, the rigid body transformation of the model is termed translation.
- Alternatively, *translation is a rigid body transformation that moves the images without deformation*.
- Translation allows the repositioning of an object from one place to another along a straight line; therefore, every point in the model moves an equal distance in a given direction.

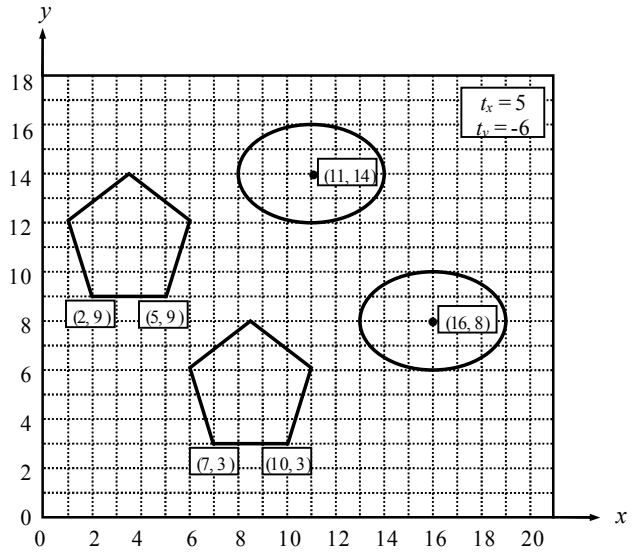
BASIC TRANSFORMATIONS…

1. Translation…

- Translation of an object is obtained by adding the translational distances t_x and t_y to the original coordinate positions (x, y) of the model.
- Thus, a point $P(x, y)$ is transformed to $P_T(x_T, y_T)$ by adding translational distances t_x and t_y as

$$x_T = x + t_x$$
$$y_T = y + t_y$$

- Figure shows the translation of a lamina and ellipse.



Translation of a lamina and an ellipse

2. Rotation

- Rotation is used to view the object models from different angles.
- It also helps to create many geometric models such as creation of entities arranged in a circular pattern by creating the basic entity once and then copying/rotating the same on the circumference of a circle.
- *Axisymmetric models* can be generated using the rotation geometric transformation.
- Rotations are rigid body transformations that move objects without deformation. This means that every point on the object is rotated through the same angle.
- Polygons are rotated by displacing each vertex through the specified rotation angle and polygon is redrawn using the new transformed rotated coordinates.
- Similarly, the curved lines are rotated by repositioning the control points to the new coordinates and redrawing the curves.

BASIC TRANSFORMATIONS…

2. Rotation…

- A two-dimensional rotation can be defined by repositioning the object on a circular path in the *xy*-plane. To achieve the rotation, the following information is required:

- Rotation point or pivot point (x_c, y_c) i.e., reference point about which the object is to be rotated
- Rotation angle α and direction of rotation (clockwise or counterclockwise)

For rotation, the following sign convention is used:

- The right-hand convention is chosen; therefore, counter-clockwise (ccw) rotation about the pivot point, when viewed from a point on the positive portion of the axis towards the origin, is termed as positive and vice-versa.

- Rotations are achieved about an axis perpendicular to the *xy*-plane that passes through the pivot point other than the origin.

2. Rotation…

Rotation about the Origin

Rotating a point through a given angle about the coordinate axes is sometimes referred to as rotation about the origin. Let the initial point $P(x, y)$ is at a distance r from the origin and makes an angle ϕ with the x-axis. The point $P(x, y)$ is rotated by an angle α through the counterclockwise direction in the xy plane (alternatively, about the z-axis) about the origin to new coordinate position $P_T(x_T, y_T)$, as shown in Figure. The angular and coordinate relationships of the transformed rotated coordinate $P_T(x_T, y_T)$ is expressed as

$$x = r.\cos\phi, \quad y = r.\sin\phi$$

and

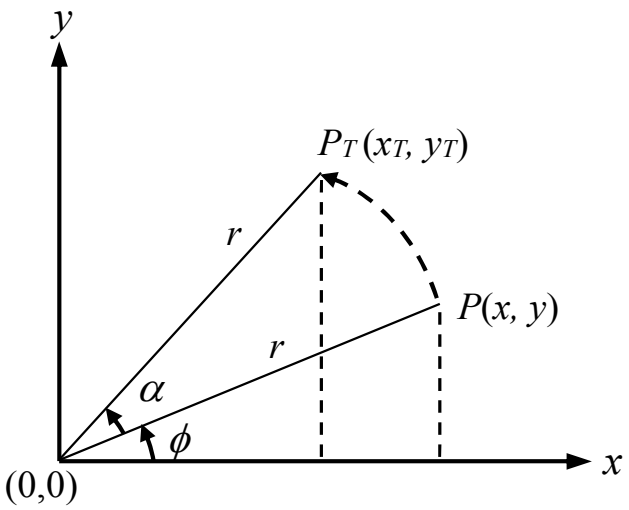
$$x_T = r.\cos(\phi+\alpha) = r.\cos\phi.\cos\alpha - r.\sin\phi.\sin\alpha$$

$$y_T = r.\sin(\phi+\alpha) = r.\sin\phi.\cos\alpha + r.\cos\phi.\sin\alpha$$

Thus, $x_T = x.\cos\alpha - y.\sin\alpha$ and $y_T = x.\sin\alpha + y.\cos\alpha$

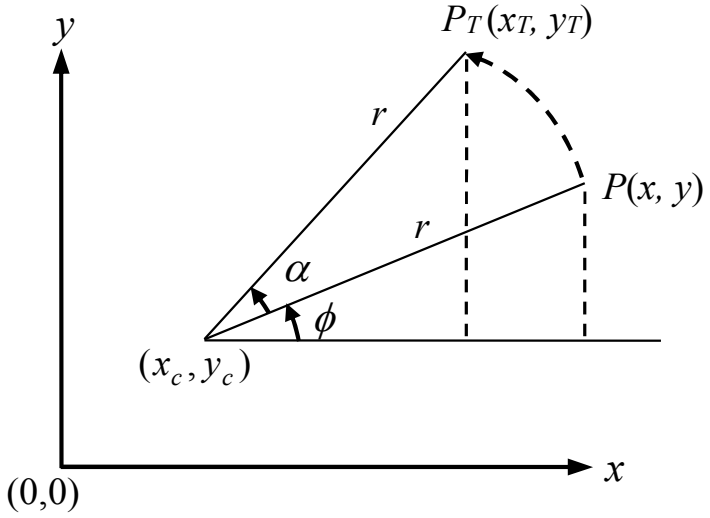
BASIC TRANSFORMATIONS…

2. Rotation…



(a)

Rotation about the Origin



(b)

Rotation about the Pivot

2. Rotation…

Rotation about the Pivot

Let the initial point $P(x,y)$ is at a distance r from the pivot point (x_c, y_c) and makes an angle ϕ with the x-axis. The point $P(x,y)$ is rotated by an angle α through the counterclockwise direction about the pivot point (x_c, y_c) in the xy-plane to the new coordinate position $P_T(x_T, y_T)$, as shown in Figure. Using geometry and trigonometric relations, the angular and coordinate relationships of the transformed rotated coordinates $P_T(x_T, y_T)$ can be expressed as

$$x_T = x_c + (x - x_c).\cos\alpha - (y - y_c).\sin\alpha$$

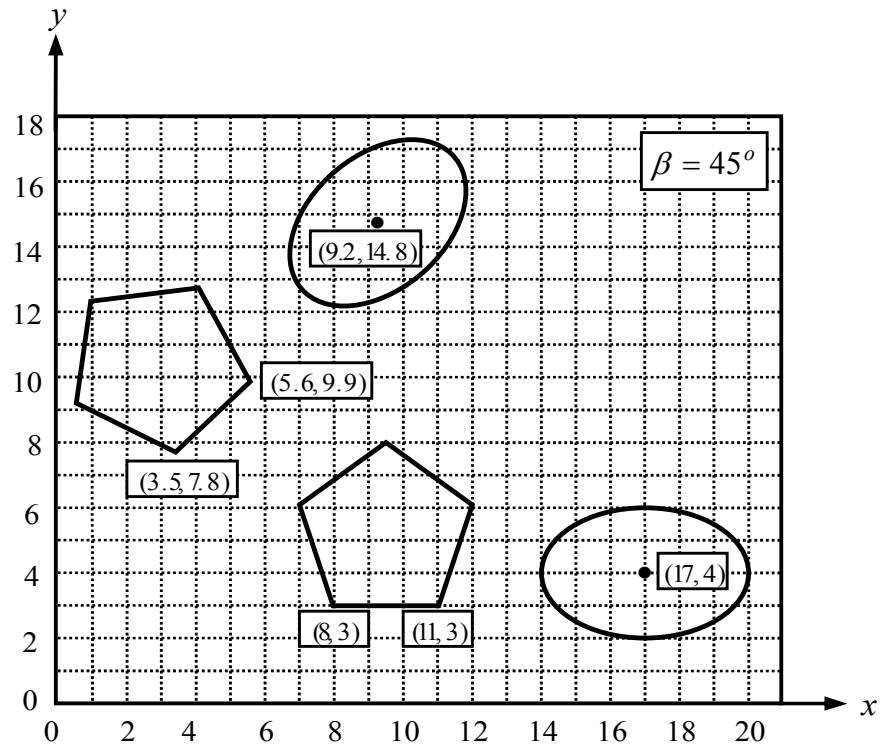
and

$$y_T = y_c + (x - x_c).\sin\alpha + (y - y_c).\cos\alpha$$

where $x - x_c = r.\cos\phi$ and $y - y_c = r.\sin\phi$

2. Rotation…

Figure shows rotation of a lamina and ellipse about the origin by an angle of 45° in the *xy*-plane.



3. Scaling

- Scaling is used to change, increase or decrease, the size of an object. Thus, it can be applied whenever enlargement/magnification or reduction in size of an object within the image is required.
- Scaling can be done either in x or y direction or in both the directions simultaneously. Scaling factors can be specified as S_x and S_y along x and y coordinate axes, respectively.
- For polygons, scaling is performed by multiplying the coordinate values (x, y) of each vertex by the scaling factors S_x and S_y, respectively. Thus, transformed coordinates will be

$$x_T = S_x.x,\ y_T = S_y.y$$

- Scaling factors are always positive (negative factor produces reflection of the model).

3. Scaling

- If scaling factors are less than unity, the reduction in size of the object occurs, i.e., the geometric model is compressed.

- However, if scaling factors are more than unity, the geometric model is stretched and enlargement or magnification in size occurs.

- Whenever scaling transformation is applied, one point remains the same and this is the fixed point of the scaling transformation.

- There are two types of scaling:

Uniform Scaling

- If $S_x = S_y$, alternatively, the model is uniformly enlarged or reduced, in both the directions and results into changes in the size only and not in the shape. Thus, the shape of the model retains.

- This operation is equivalent to ***zoom*** command in CAD software.

3. Scaling…

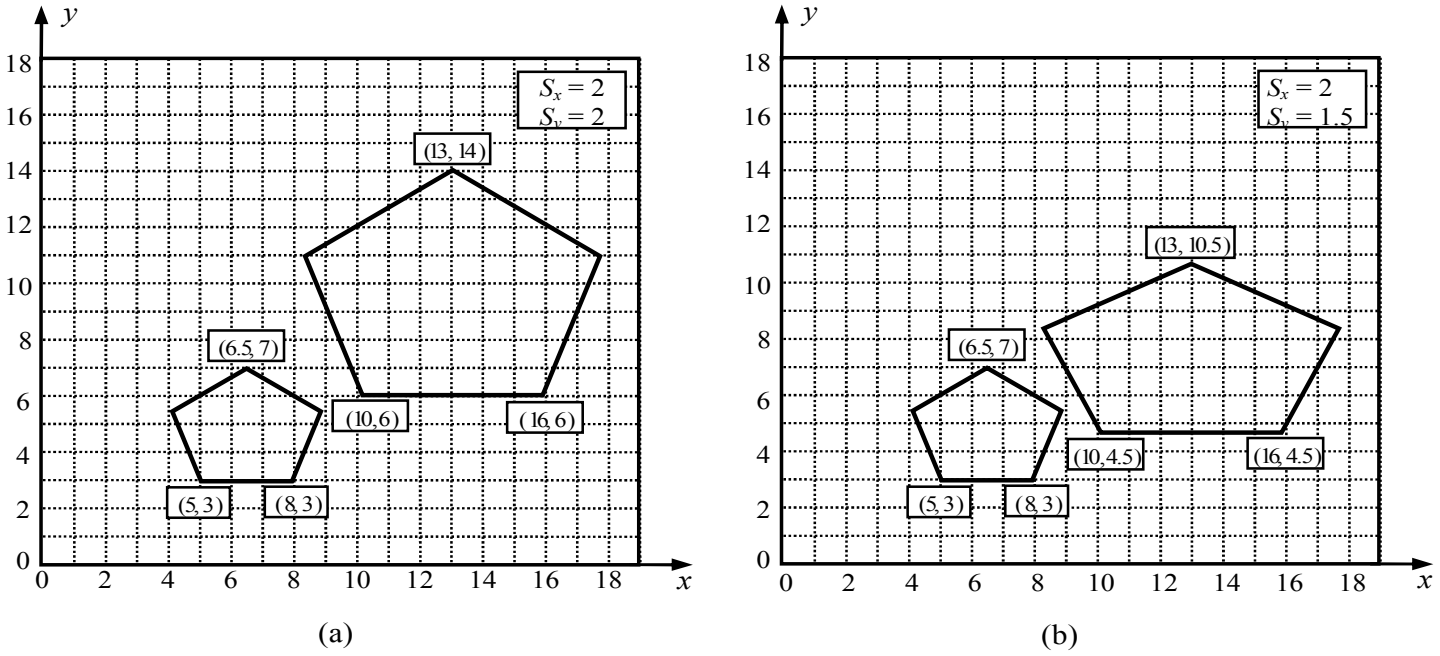
Differential Scaling…

- If $S_x \neq S_y$, the model distorts due to non-uniform enlargement/reduction in coordinate directions.
- Differential scaling changes both size and shape of the geometric model.
- Differential scaling is only of mathematical interest because it is seldom used in practical applications.

BASIC TRANSFORMATIONS…

3. Scaling…

Differential Scaling



Uniform Scaling **Differential Scaling**

3. Scaling…

- Polygon scaling is obtained by applying transformations to each vertex, and polygon redraws with the new transformed coordinates of its vertices.
- An ellipse is resized by scaling the semi-major and semi-minor axes and redrawing the ellipse about the specified centre coordinates.
- Similarly, uniform scaling of a circle occurs by simply adjusting its radius, and circle redraws with the specified radius using the transformed coordinates.

COMPUTER AIDED DESIGN
(BME-42)

Unit-II: Geometric Transformations

(3 Lectures)

- 2D Geometric transformations- Translation, Scaling, Rotation, **Reflection & Shear**
- **Matrix representation**
- **homogeneous coordinates**
- Rotation and scaling about arbitrary point
- Reflection through arbitrary line
- Composite transformation
- 3 D transformations
- multiple transformation

Lecture 16

Topics Covered

2D Geometric Transformations

Transformation of Geometric Models

Complex Transformations

Matrix Representation of 2DTransformations

Homogeneous Coordinate Representation

Homogeneous Transformation Matrices

Prepared By

Prof. S. K. SRIVASTAVA

MED, MMMUT, Gorakhpur (UP)
sksme@mmmut.ac.in

TRANSFORMATIONS OF GEOMETRIC MODELS

Complex Transformations

- Most graphics software includes the basic transformations such as translation, rotation and scaling.
- In these transformations, parallel lines remain parallel, but angles and lengths may change. Such transformations are termed *affine* transformations.
- Some packages provide a few *additional affine transformations* that are useful in certain applications.
- **Reflection** (or mirroring) and **Shear** are the other two modeling transformations.

4. Reflection

- A reflection (or mirroring) is a geometric transformation that produces mirror image of an object.
- Reflection transformation is used for constructing the symmetric models.

TRANSFORMATIONS OF GEOMETRIC MODELS

Complex Transformations...

4. Reflection…

- If a model is symmetric with respect to a plane then one-half of the geometry is created at first, followed by the reflection transformation to generate the full model.
- A geometric model may be reflected through a plane, a line, or a point in space. The mirror image of 2D reflection is generated with respect to an axis of reflection.
- Reflection of a geometric model is achieved by *rotating the object by 180^o about the reflection axis.*
- The axis of reflection may be either in the plane or in perpendicular to the plane.
- When reflection axis is in plane, the rotation path about this axis is in a plane perpendicular to the plane.
- However, the rotation path is in plane if reflection axes are perpendicular to the plane.

TRANSFORMATIONS OF GEOMETRIC MODELS

Complex Transformations...

4. Reflection…

There are different types of two-dimensional reflections:

Reflection of a point about x-axis

- It represents the reflection of a point about the line $y = 0$. Reflection transformation keeps x coordinates same, but y values reverses in sign.
- Thus, the transformed coordinates of a point $P(x, y)$ about the x-axis is given as

$$x_T = x \quad \text{and} \quad y_T = -y$$

Reflection of a point about y-axis

- It represents the reflection of a point about the line $x = 0$. Reflection transformation keeps y coordinates same, but x values reverses its sign.
- Thus, the transformed coordinates of a point $P(x, y)$ about the y-axis is given as

$$x_T = -x \quad \text{and} \quad y_T = y$$

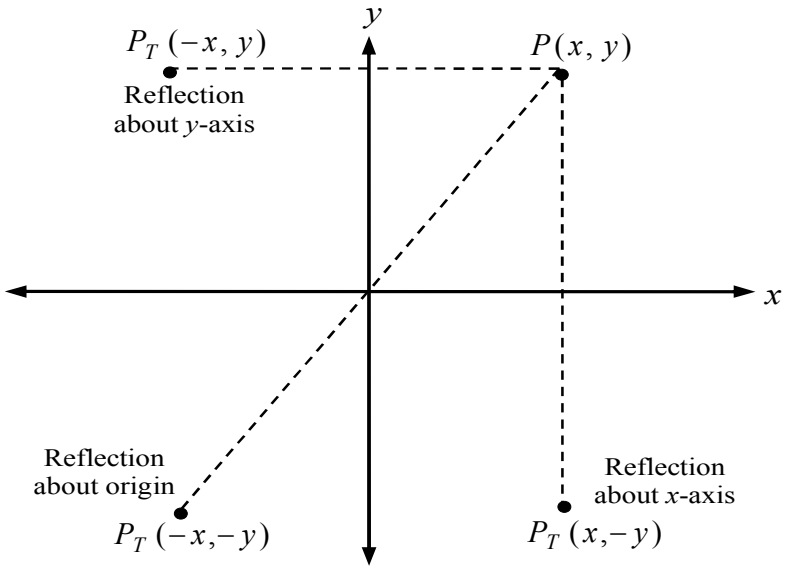
Complex Transformations...

4. Reflection…

Reflection of a point about the Origin

- The transformed coordinates of a point $P(x, y)$ about the *origin* is given as

$$x_T = -x \text{ and } \quad y_T = -y$$



Reflection of a point about the coordinate axes and origin

TRANSFORMATIONS OF GEOMETRIC MODELS

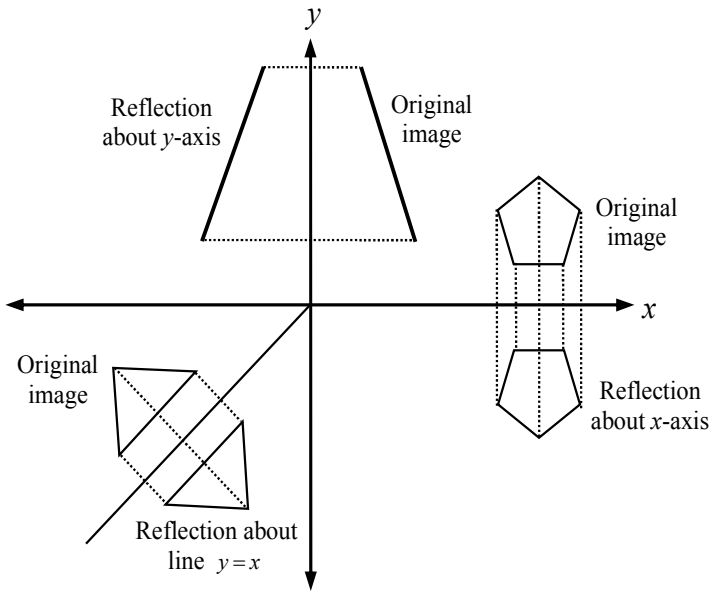
Complex Transformations...

4. Reflection…

Reflection of a point about the line y = x

- The transformed coordinates of a point $P(x, y)$ about the line $y = x$ is given as

$$x_T = y \text{ and } y_T = x$$



Reflection of 2D objects about the coordinate axes and the line $y = x$

TRANSFORMATIONS OF GEOMETRIC MODELS

Complex Transformations…

5. Shear

- The geometric transformation that *distorts* shape of an object such that the transformed shape appears as if the object were composed of internal layers that slide over each other, is termed *shear*.
- In other words, the sliding of internal layers of an object over the other layers to distort its shape is termed shear transformation.
- Shear is the controlled distortion of an object model in x and y coordinates.
- Two common shear transformations are those in which x coordinate depends on y coordinate and vice-versa.
- The transformed coordinates of a point $P(x, y)$, , when shear along the x direction takes place, is given as

$$x_T = x + Sh_x . y \quad \text{and} \quad y_T = y$$

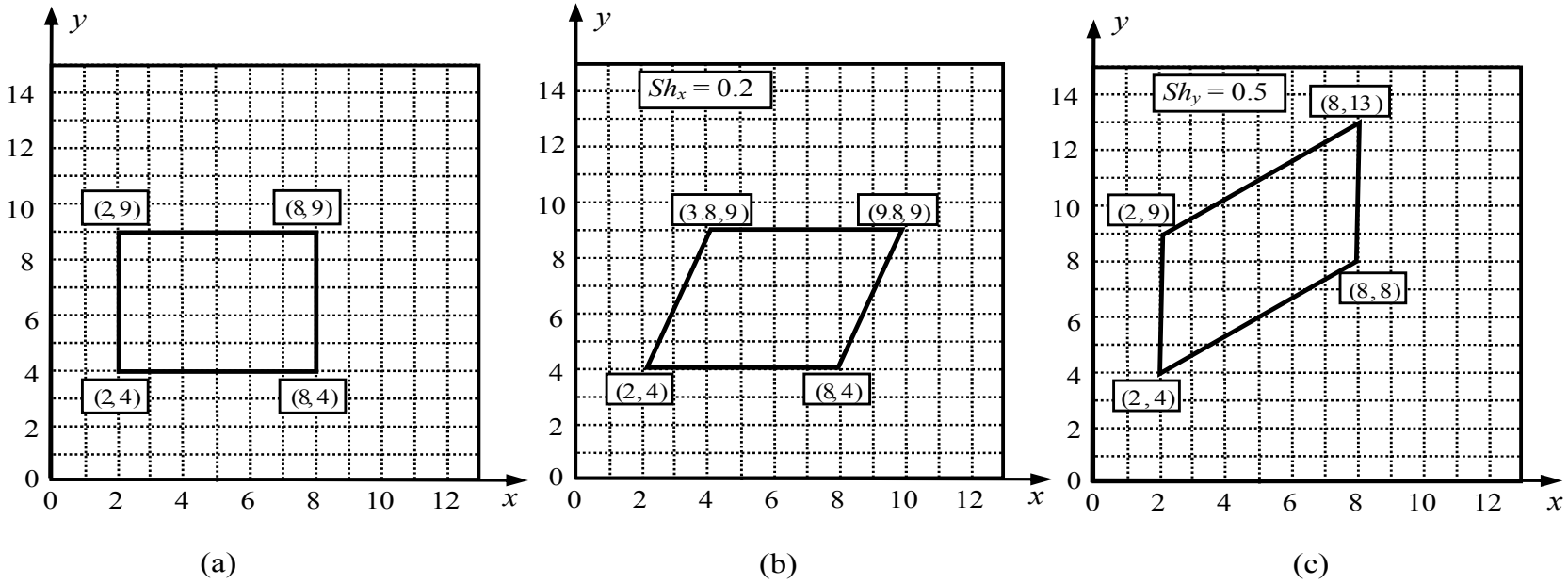
TRANSFORMATIONS OF GEOMETRIC MODELS

Complex Transformations…

5. Shear…

- Moreover, shear along the *y*-direction will be

$$x_T = x \quad \text{and} \quad y_T = y + Sh_y . x$$



Shearing of a lamina (a) original shape (b) shear along x-axis (c) shear along y-axis

MATRIX REPRESENTATION OF 2D TRANSFORMATIONS

- From the expressions of different 2D transformations, it can be observed that in some cases we have *addition* or *multiplication* or both types of terms.
- For complex images, different geometric transformations are applied many times.
- All the geometric transformations can be represented in matrix form.

General matrix equation representing these geometric transformations may be expressed as

$$\begin{Bmatrix} x_T \\ y_T \end{Bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix} = [T] \begin{Bmatrix} x \\ y \end{Bmatrix}$$

Where $[T]$ is the transformation matrix.

Depending upon the values and sign of different elements (A, B, C and D) of transformation matrix $[T]$, different types of geometric transformation can be obtained. The matrices $[T]$ are different to achieve rotation, reflection, scaling and shear transformation of an object.

MATRIX REPRESENTATION OF 2D TRANSFORMATIONS

- Thus, the matrix equations for these transformations may be expressed as

$\{P_T\} = [T_r].\{P\}$ for translation

$\{P_T\} = [R].\{P\}$ for rotation

$\{P_T\} = [S].\{P\}$ for scaling

$\{P_T\} = [Sh].\{P\}$ for shearing

- The right-hand side terms of these matrix equations are the product of two matrices.

- Unfortunately, translation is not possible with this type of matrix representation because of addition of constant terms t_x and t_y associated with x_T and y_T coordinates, i.e.,

$$x_T = A.x + B.y + t_x$$
$$y_T = C.x + D.y + t_y$$

Alternatively, $\{P_T\} = [T_t] + \{P\}$

MATRIX REPRESENTATION OF 2D TRANSFORMATIONS

- In computer graphics, the image manipulation of a model is obtained through the multiple geometric transformations.
- It requires the matrix multiplications corresponding to each geometric transformation.
- The geometric transformations represented by the above matrix equations do not incorporate translation transformation; therefore, it would be impossible to implement the image manipulation through programming when these geometric transformations (including translation) are applied successively to an object model.

MATRIX REPRESENTATION OF 2D TRANSFORMATIONS

Homogeneous Coordinate Representation

- The difficulty in image manipulation, incorporating all the five types of geometric transformations, can be removed if represented by a single matrix equation.
- This is possible if points are represented in *homogeneous coordinates*.
- The homogeneous coordinates are obtained by adding the third coordinate to a point.
- This facilitates the image manipulation with a single transformation matrix for all types of geometric transformations.

Thus, there are mainly three advantages of using homogeneous coordinates representation:

I. It is possible to calculate overall transformation matrix through the matrix multiplications corresponding to each geometric transformation.

MATRIX REPRESENTATION OF 2D TRANSFORMATIONS

Homogeneous Coordinate Representation…

II. It also helps to achieve advanced type of transformation such as *projection*.

III. It removes many anomalies encountered in Cartesian geometry such as representing the points at infinity and non-intersection of parallel lines.

- In homogeneous coordinates system, mapping between the n-dimensional spaces with $(n+1)$-dimensional spaces occurs, if points in n-dimensional coordinates are represented by the corresponding $(n+1)$-dimensional coordinates.
- This is obtained by introducing a scale factor along the Cartesian coordinates.
- In 2D coordinates, instead of being represented by a pair (x, y), each point is represented by triple coordinates,(x', y', h) where $h \neq 0$ is the *scale factor*.

MATRIX REPRESENTATION OF 2D TRANSFORMATIONS

Homogeneous Coordinate Representation…

- The relationship between the Cartesian coordinates and homogeneous coordinates of a point is given by

$$x = x'/h\text{'} \quad y = y'/h$$

or $x' = x.h\ , \quad y' = y.h$

Generally, $h = 1$ represents homogeneous coordinate $(x, y, 1)$ for a point (x, y) in computer graphics.

- At the same time, if two sets of homogeneous coordinates represent the same point if and only if one is multiple of the other.

- Hence, different homogeneous coordinates can represent each point. Therefore, homogeneous coordinates of a point (3, 2) may be expressed by the coordinate triples as (3, 2, 1) or (6, 4, 2) or (9, 6, 3).

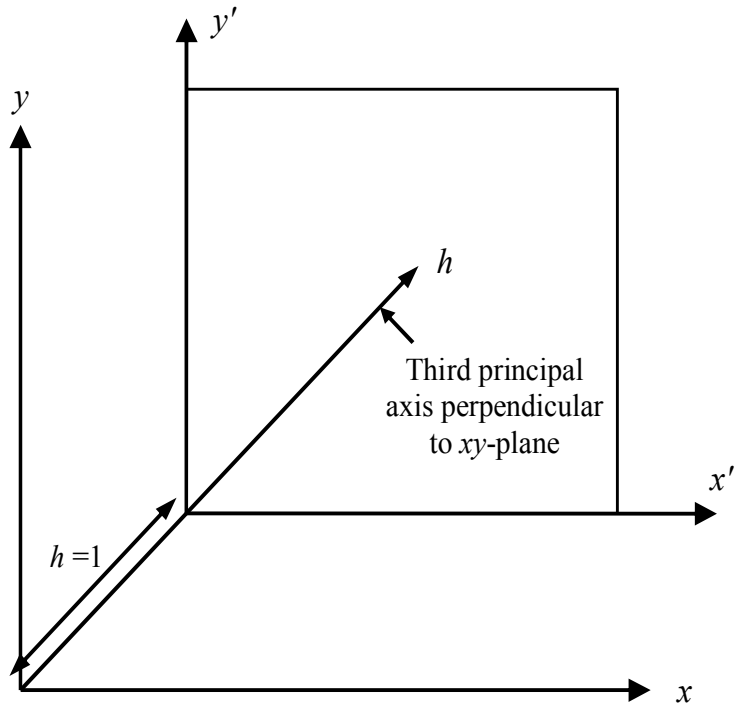
MATRIX REPRESENTATION OF 2D TRANSFORMATIONS

Homogeneous Coordinate Representation…

- If we take all triples representing the same point, we get a line in 3D-space. Thus, each homogeneous point represents a line in 3D-space.
- If these points are homogenized by dividing with scale factor h, it results a point in the form of $(x, y, 1)$.
- Figure shows a homogenized points form the plane, defined by equation $h = 1$, in (x, y, h)-space.
- Graphically, the scale factor h may be interpreted as the Cartesian images on a plane parallel to the xy-plane and at unit distance away from the origin along z-direction.
- This type of visualization (viz. x, y, z, h) is not possible in 3D geometric transformations. However, its mathematical representation is possible.
- In computer graphics, we shift the coordinates of the object model from (x, y) coordinate space to $(x, y, 1)$. For example, (3, 2) as (3, 2, 1).

MATRIX REPRESENTATION OF 2D TRANSFORMATIONS

Homogeneous Coordinate Representation…



The (x, y, h) homogeneous coordinate space, with h = 1 plane

HOMOGENEOUS TRANSFORMATION MATRICES

In generalized form, the matrix equation incorporating all *five* types of geometric transformations may be expressed as

$$\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} A & B & 0 \\ C & D & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix} = [T] . \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$$

where [T] represents a transformation matrix in homogeneous coordinates. Different 2D geometric transformation matrices in homogeneous coordinates are

Translation:

$$[T_t] = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation:

Counterclockwise rotation (ccw) in the *xy*-plane,

$$[T_r] = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Clockwise rotation (cw) in the *xy*-plane,

$$[T_r] = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

HOMOGENEOUS TRANSFORMATION MATRICES

Reflection:

About the x-axis, $[R_x] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

About the y-axis, $[R_y] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

About the origin, $[R] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

About the line $y = x$, $[R] = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Shear:

Along the x-axis, $[Sh_x] = \begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Along the y-axis, $[Sh_y] = \begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Along the x and y-axes, $[Sh] = \begin{bmatrix} 1 & Sh_x & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

HOMOGENEOUS TRANSFORMATION MATRICES

Scaling & Shear:

$$\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} A & B & 0 \\ C & D & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix} = [T] \cdot \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$$

- The effect of elements A, B, C and D in 3 x 3 matrix may be separately identified.
- The terms **B** and **C** causes shear along *x* and *y* directions, respectively.
- The terms **A** and **D** act as scale factors.
- Thus, the general 3 x 3 transformation matrix producing a combination of *scaling and shear*, is expressed as

$$[T] = \begin{bmatrix} S_x & Sh_x & 0 \\ Sh_y & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In general, transformations with a determinant identically equal to +1 gives pure rotations.

If the determinant of matrix is identically -1 then the transformation produces pure reflection.

COMPUTER AIDED DESIGN
(BME-42)

Unit-II: Geometric Transformations

(3 Lectures)

- 2D Geometric transformations- Translation, Scaling, Rotation, Reflection & Shear
- Matrix representation
- homogeneous coordinates
- **Rotation and scaling about arbitrary point**
- **Reflection through arbitrary line**
- **Composite transformation**
- **3 D transformations**
- **multiple transformation**

Lecture 17

Topics Covered

Geometric Transformations of 2D Objects

Transformation of Points, Lines, Lamina

Composite Transformations

Rotation About an Arbitrary Point

Scaling About an Arbitrary Point

Reflection Through an Arbitrary Line

Axis of Reflection Passes Through Origin

Axis of Reflection is Arbitrary Line

Prepared By

Prof. S. K. SRIVASTAVA

MED, MMMUT, Gorakhpur (UP)

sksme@mmmut.ac.in

GEOMETRIC TRANSFORMATIONS OF 2D OBJECTS

- An object model consists of large number of points (or vertices). Geometric transformations are applied to these points defining the model.
- After the transformation, the object model redraws with new transformed coordinates.
- A 2D point in homogeneous coordinates is represented as $(x, y, 1)$. The three values are either specified as

1-row, 3-column matrix: $\{x \quad y \quad 1\}$

3-rows, 1-column matrix: $\begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$

Here, the column matrix formulation of position vectors is used.

In geometric transformations, matrix formulation is used for the calculation of transformed coordinates of the object model

GEOMETRIC TRANSFORMATIONS OF 2D OBJECTS

For single point transformation, matrix eqn. is given by $\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = [T].\begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$

When multiple vertices, such as in plane lamina, define the object, matrix eqn. modifies as

$$\begin{Bmatrix} x_{T1} & x_{T2} & x_{T3} & - & - & x_{Tn} \\ y_{T1} & y_{T2} & y_{T3} & - & - & y_{Tn} \\ 1 & 1 & 1 & 1 & 1 & 1 \end{Bmatrix} = [T].\begin{Bmatrix} x_1 & x_2 & x_3 & - & - & x_n \\ y_1 & y_2 & y_3 & - & - & y_n \\ 1 & 1 & 1 & 1 & 1 & 1 \end{Bmatrix}$$

Transformation of Points

The general transformation matrix equation is written as $\{P_T\} = [T].\{P\}$

For *translation* operation, $\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}\begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$ or $\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{Bmatrix} x+t_x \\ y+t_y \\ 1 \end{Bmatrix}$

GEOMETRIC TRANSFORMATIONS OF 2D OBJECTS

Transformation of Points...

For *rotation* operation, $\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$ or $\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{Bmatrix} x.\cos\alpha - y.\sin\alpha \\ x.\sin\alpha + y.\cos\alpha \\ 1 \end{Bmatrix}$

For *scaling* operation, $\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$ or $\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = . \begin{Bmatrix} S_x.x \\ S_y.y \\ 1 \end{Bmatrix}$

For *reflection* about the x-axis, $\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$ or $\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{Bmatrix} x \\ -y \\ 1 \end{Bmatrix}$

Similarly, the transformed coordinates for other type of *reflection* can be obtained.

For *shear* about the x-axis, $\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} 1 & Sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$

Transformation of Points...

For *shear* about the y-axis,

$$\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$$

For *shear* about the x & y -axes,

$$\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} 1 & Sh_x & 0 \\ Sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$$

Transformation of Straight Lines

- Line with different positions and orientations are obtained by changing the position vectors of its endpoints.
- Figure shows a straight line between the endpoints A (1, 2) and B (2, 4) to be transformed through the shear deformation matrix

$$[T] = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

GEOMETRIC TRANSFORMATIONS OF 2D OBJECTS

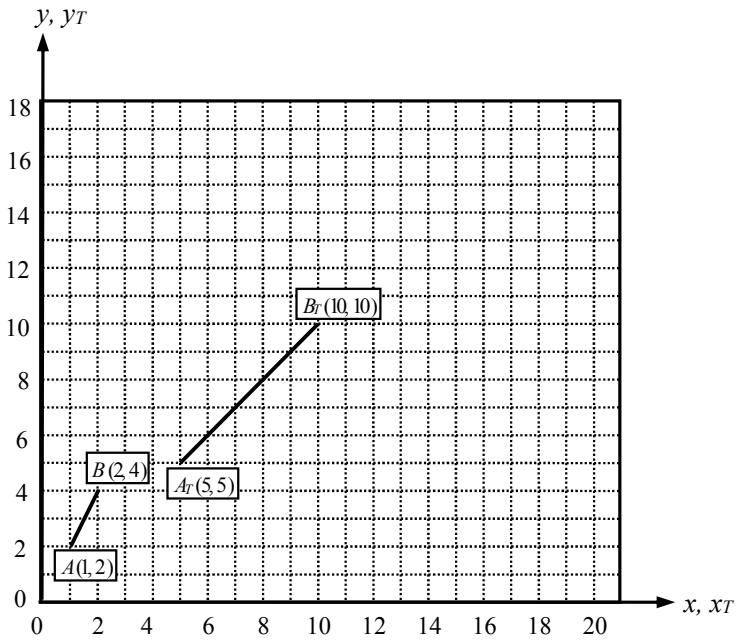
Transformation of Straight Lines…

- Thus, the transformed coordinates can be calculated as

$$\{P_T\} = [T]\{P\} \quad \text{or} \quad \begin{Bmatrix} x_{T1} & x_{T2} \\ y_{T1} & y_{T2} \\ 1 & 1 \end{Bmatrix} = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} 1 & 2 \\ 2 & 4 \\ 1 & 1 \end{Bmatrix} = \begin{Bmatrix} 5 & 10 \\ 5 & 10 \\ 1 & 1 \end{Bmatrix}$$

Thus, transformed coordinates of endpoints are

$A_T(5,5)$ and $B_T(10,10)$



Transformation of Plane Lamina

Example: Determine the transformed coordinates for the following geometric transformations:

Triangular lamina having vertices $A(4,2)$, $B(2,-2)$ *and* $C(6,-2)$ *is subjected to the translation with translational distances* $t_x = 5$ *and* $t_y = 7$ *along the coordinate axes*

The transformed translated coordinates may be calculated as

$$\begin{Bmatrix} x_{T1} & x_{T2} & x_{T3} \\ y_{T1} & y_{T2} & y_{T3} \\ 1 & 1 & 1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 7 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} 4 & 2 & 6 \\ 2 & -2 & -2 \\ 1 & 1 & 1 \end{Bmatrix} = \begin{Bmatrix} 9 & 7 & 11 \\ 9 & 5 & 5 \\ 1 & 1 & 1 \end{Bmatrix}$$

The transformed coordinates is given as $A_T(9,9)$, $B_T(7,5)$, $C_T(11,5)$ shown in Fig. (a)

Triangular lamina having vertices $A(4,2)$, $B(2,-2)$ *and* $C(6,-2)$ *is rotated through* 90^o *in counterclockwise direction about the origin*

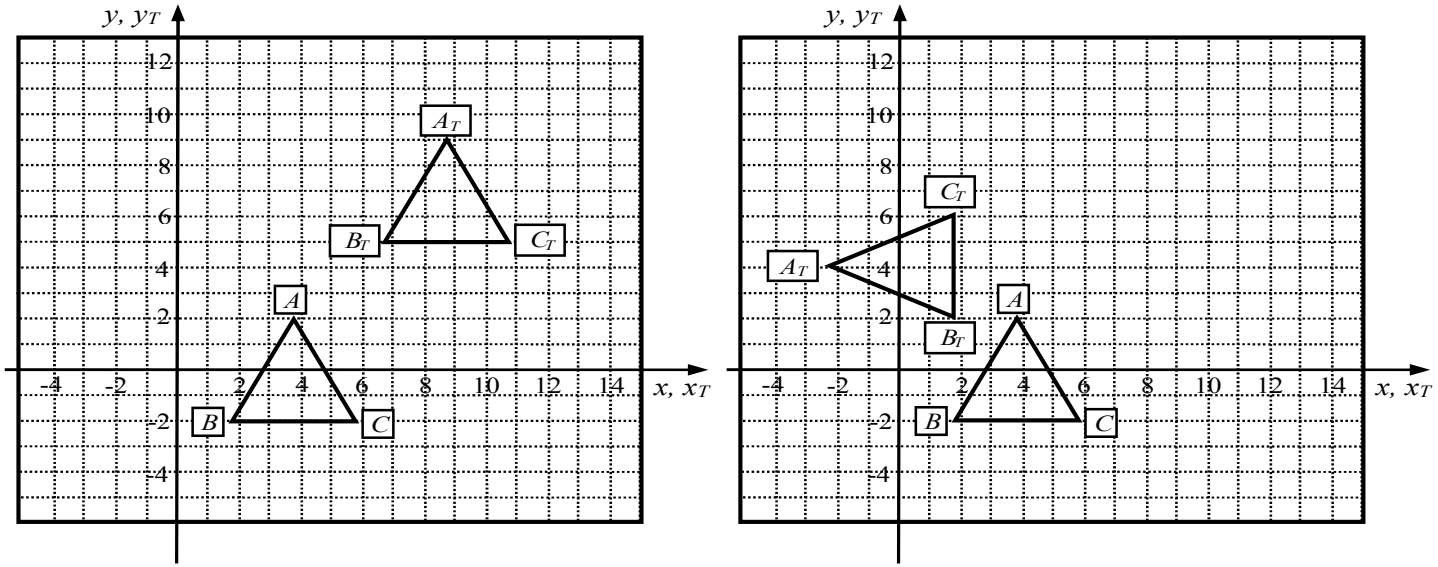
GEOMETRIC TRANSFORMATIONS OF 2D OBJECTS

Transformation of Plane Lamina…

The transformed translated coordinates may be calculated as

$$\begin{Bmatrix} x_{T1} & x_{T2} & x_{T3} \\ y_{T1} & y_{T2} & y_{T3} \\ 1 & 1 & 1 \end{Bmatrix} = \begin{bmatrix} \cos 90^0 & -\sin 90^0 & 0 \\ \sin 90^0 & \cos 90^0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{Bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} 4 & 2 & 6 \\ 2 & -2 & -2 \\ 1 & 1 & 1 \end{Bmatrix} = \begin{Bmatrix} -2 & 2 & 2 \\ 4 & 2 & 6 \\ 1 & 1 & 1 \end{Bmatrix}$$

The transformed coordinates is given as $A_T(-2, 4), B_T(2, 2), C_T(2, 6)$ shown in Fig. (b)



COMPOSITE TRANSFORMATIONS

- When there is arbitrary sequence of rotation, translation and scaling transformations, parallel lines remain parallel but angles and lengths may change.
- Such transformations are termed ***affine*** transformations.
- The shape and position of 2D objects can be controlled by performing the matrix operations on the position vectors defining vertices of the model such as lamina/surface.
- *Any sequence of transformation is termed as composite (or combined) transformations.*
- This can be achieved by multiplying all the transformation matrices in sequence (or order) and overall transformation is represented by a single matrix.
- This is also termed ***concatenation*** of transformations.
- Since matrix multiplication is non-commutative $(A.B \neq B.A)$, the order of applications of transformation is important; therefore, care must be taken to ensure the order of matrix multiplication in the same way as that of the transformations.

COMPOSITE TRANSFORMATIONS

- Another commutative pair of transformations may be uniform scaling and rotation operations.
- The associative property of matrix ($A.B.C = (A.B).C = A.(B.C)$) does not impose any restriction for the evaluation of composite matrix in either order. This can be verified using a suitable example.

If numbers of geometric transformations are taking place successively, e.g., operation sequence is

$$\text{Rotation } [T_r] \rightarrow \text{Scaling } [S] \rightarrow \text{Reflection } [R] \rightarrow \text{Translation } [T_t]$$

Then, overall transformation matrix is expressed as

$$[T] = [T_t].[R].[S].[T_r]$$

Moreover, matrix equation for the transformation of a point with position vector $\{X\}$

$$\{X_T\} = [T].\{X\}$$

COMPOSITE TRANSFORMATIONS

***Example**: Find the transformed position of a rectangular lamina having vertices A(1, 2), B (3, 2), C(4, 3) and D (2, 4) subjected to the following successive transformations*

I. Reflection about the y-axis
II. Translation by -1 unit in x & y directions
III. Rotation by 180^o counterclockwise direction about the z-axis

Reflection matrix, $$[R] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Translation matrix, $$[T_t] = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation matrix, $$[T_r] = \begin{bmatrix} \cos 180^o & -\sin 180^o & 0 \\ \sin 180^o & \cos 180^o & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The resulting concatenated matrix is

$$[T] = [T_r].[T_t].[R]$$

COMPOSITE TRANSFORMATIONS

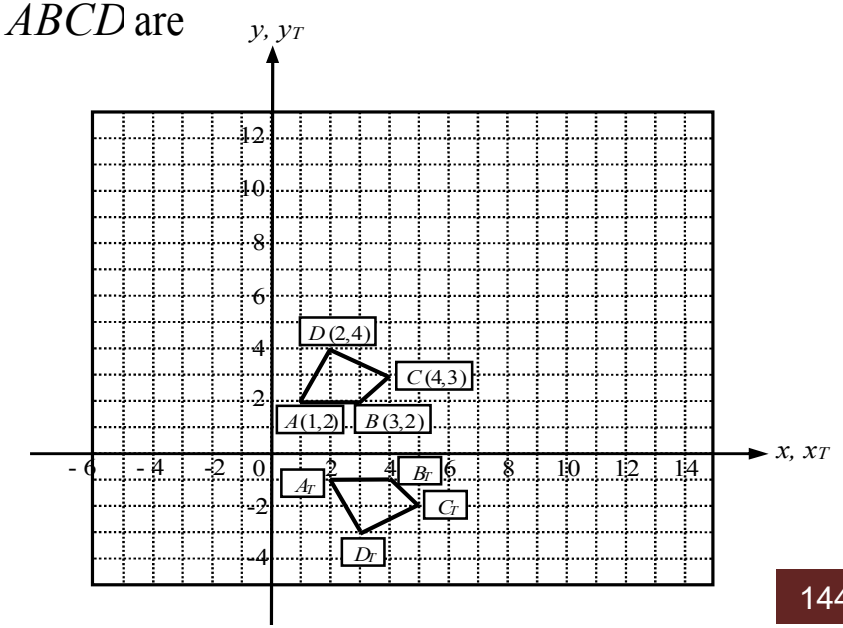
$$[T]=\begin{bmatrix}-1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & -1\\ 0 & 1 & -1\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}-1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & 1\\ 0 & -1 & 1\\ 0 & 0 & 1\end{bmatrix}$$

Hence, transformed position vectors of the lamina $ABCD$ are

$$\begin{Bmatrix}x_{T1} & x_{T2} & x_{T3} & x_{T4}\\ y_{T1} & y_{T2} & y_{T3} & y_{T4}\\ 1 & 1 & 1 & 1\end{Bmatrix}=\begin{bmatrix}1 & 0 & 1\\ 0 & -1 & 1\\ 0 & 0 & 1\end{bmatrix}\begin{Bmatrix}1 & 3 & 4 & 2\\ 2 & 2 & 3 & 4\\ 1 & 1 & 1 & 1\end{Bmatrix}=\begin{Bmatrix}2 & 4 & 5 & 3\\ -1 & -1 & -2 & -3\\ 1 & 1 & 1 & 1\end{Bmatrix}$$

Thus, the transformed coordinates of rectangular lamina $ABCD$ are

$A_T(2,-1)$, $B_T(4,-1)$, $C_T(5,-2)$, $D_T(3,-3)$



COMPOSITE TRANSFORMATIONS

Rotation About an Arbitrary Point

The rotations about an arbitrary point (x_r, y_r) other than the origin, can be accomplished by performing **three** steps taken in sequence:

I. Translate point (x_r, y_r) to the origin

II. Rotate the object by desired angle about the origin (alternatively, about an axis perpendicular to the plane of rotation

III. Translate point (x_r, y_r) back to the origin

Thus, matrix equation for the rotation of position vector $\{X\}$ about the point (x_r, y_r), by an angle α in ccw direction in the *xy*-plane is expressed as

$$\{X_T\} = [T_t]^{-1}.[T_r].[T_t].\{X\}$$

where $[T_t]$ = Translation matrix obtained by adding translational distances $-x_r$ and $-y_r$ to the position vector

Rotation About an Arbitrary Point….

$[T_r]$ = Rotation matrix for rotation of position vector $\{X\}$ by an angle α in ccw direction so that it rotates about the origin in xy-plane

$[T_t]^{-1}$ = Inverse translation matrix obtained by adding translational distances x_r and y_r, to the position vector $\{X\}$ so that it translates back to the original position

The matrix equation yields

$$\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix} . \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$$

By carrying out the products of interior matrices, we have

$$\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & x_r.(1-\cos\alpha)+y_r.\sin\alpha \\ \sin\alpha & \cos\alpha & y_r.(1-\cos\alpha)-y_r.\sin\alpha \\ 0 & 0 & 1 \end{bmatrix} . \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$$

COMPOSITE TRANSFORMATIONS

Scaling About an Arbitrary Point

The scaling about an arbitrary point (x_s, y_s), other than the origin, can be accomplished by performing the three steps in sequence:

I. Translate point (x_s, y_s) to the origin

II. Scale the object along x and y directions about the origin with scaling factors S_x and S_y

III. Translate point (x_s, y_s) back to the original position

Thus, matrix equation for scaling of position vector $\{X\}$ about the point having coordinates (x_s, y_s) is given as

$$\{X_T\} = [T_t]^{-1}.[S].[T_t].\{X\}$$

Where $[T_t]$ = translation matrix obtained by adding translational distances $-x_s$ and $-y_s$ to the position vector $\{X\}$ so that it coincides with the origin

$[S]$ = scaling matrix with scaling factors S_x and S_y about the origin

COMPOSITE TRANSFORMATIONS

Scaling About an Arbitrary Point…

$[T_t]^{-1}$ = inverse translation matrix obtained by adding translational distances x_s and y_s to the position vector $\{X\}$ so that it translates back to the original position

The matrix eqn. yields

$$\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & x_s \\ 0 & 1 & y_s \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & -x_s \\ 0 & 1 & -y_s \\ 0 & 0 & 1 \end{bmatrix} . \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$$

By carrying out the products of interior matrices, we have

$$\begin{Bmatrix} x_T \\ y_T \\ 1 \end{Bmatrix} = \begin{bmatrix} S_x & 0 & x_s.(1-S_x) \\ 0 & S_y & y_s.(1-S_y) \\ 0 & 0 & 1 \end{bmatrix} . \begin{Bmatrix} x \\ y \\ 1 \end{Bmatrix}$$

Reflection Through an Arbitrary Line

- The reflection of a point through the axes/lines that passes through the origin, are considered.
- In general, reflection through a general line that may or may not pass through the origin is required.
- This can be accomplished using a procedure similar to that for the rotation about an arbitrary point.

In general, there are two axes of reflections:

Axis of Reflection Passes through Origin

- The axis of reflection may be a line $(y = mx)$ passing through the origin having a slope $m = \tan\phi$ where ϕ is the angle made by the line with the x-axis.

The overall transformation matrix for the reflection about this line can be achieved by performing the following transformations steps in sequence:

COMPOSITE TRANSFORMATIONS

Reflection Through an Arbitrary Line…

Axis of Reflection Passes through Origin…

I. Rotate the line and the object about the origin until line is coincident with one of the coordinate axes (say, the line is rotated by an angle ϕ in clockwise direction about the z-axis so that it becomes coincident with the x-axis.

II. Reflect the object through the coordinate axis (say, x-axis)

III. Inverse rotate the object about the coordinate axis (say, z-axis)

Thus, the resulting concatenated matrix is expressed as $[T] = [T_r]^{-1}.[R].[T_r]$

The rotations and reflections are also applied to the object to be transformed. The matrix equation for the reflection of position vector $\{X\}$ about the line $(y = mx)$ is given by

$$\{X_T\} = [T].\{X\}$$

COMPOSITE TRANSFORMATIONS

Reflection Through an Arbitrary Line…

Axis of Reflection is Arbitrary Line

When the axis of reflection is a line $(y = mx + c)$, not passes through the origin, having a slope $m = \tan\phi$ where ϕ is the angle, which line makes with the x-axis.

The overall transformation matrix can be obtained by performing the following steps in sequence:

I. Translate the line and the object so that the line passes through the origin.

II. Rotate the line and object about the axis until the line becomes coincident with one of the coordinate axes (say, line is rotated by an angle ϕ in clockwise direction so that it becomes coincident with the x-axis).

III. Reflect the object through the coordinate axis (say, x-axis)

IV. Inverse rotate the object about the coordinate axis (say, z-axis).

V. Translate back the line/object to the original position.

Reflection Through an Arbitrary Line…

Axis of Reflection is Arbitrary Line…

Thus, the resulting concatenated matrix is given by

$$[T] = [T_t]^{-1}.[T_r]^{-1}.[R].[T_r].[T_t]$$

The translations, rotations and reflections are also applied to the object. The matrix equation for the reflection of position vector $\{X\}$ about the line $(y = mx)$ is given as

$$\{X_T\} = [T].\{X\}$$

COMPUTER AIDED DESIGN
(BME-42)

Unit-II: Geometric Transformations

(3 Lectures)

- 2D Geometric transformations- Translation, Scaling, Rotation, Reflection & Shear
- Matrix representation
- homogeneous coordinates
- Rotation and scaling about arbitrary point
- Reflection through arbitrary line
- Composite transformation
- **3 D transformations**
- **multiple transformation**

Lecture 18

Topics Covered

Geometric Transformations of 3D Objects

Translation, Rotation, Scaling

Reflection and Shear

Composite Transformations

Prepared By

Prof. S. K. SRIVASTAVA

MED, MMMUT, Gorakhpur (UP)

sksme@mmmut.ac.in

GEOMETRIC TRANSFORMATIONS OF 3D OBJECTS

- 3D geometric transformations can be obtained by including *z* axis in the modeling.
- In 3D transformations, the object is translated by specifying three-dimensional *translation* vector, which determines how much the object is to be moved along the coordinate axes.
- Similarly, the object can be *scaled* with scaling factors corresponding to the three coordinate's axes.
- However, in case of three-dimensional rotation, it is not the straightforward conversion from two-dimensions.
- In 2D rotation, we rotate the object in the *xy* plane or about the axis that is perpendicular to the *xy* plane, i.e., *z*-axis.
- In three-dimensional rotations, we can rotate the object about any one of the *three coordinate axes*. Mostly, the graphics software provides three options for the rotations, each along the three Cartesian axes.

GEOMETRIC TRANSFORMATIONS OF 3D OBJECTS

A point in three-dimensional space (x, y, z), in homogeneous coordinates is represented by the four-dimensional position vector as

$$\begin{Bmatrix} x' \\ y' \\ z' \\ h \end{Bmatrix} = [T].\begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

The matrix equation can be transformed to ordinary coordinates as

$$\begin{Bmatrix} x_T \\ y_T \\ z_T \\ 1 \end{Bmatrix} = [T].\begin{Bmatrix} x'/h \\ y'/h \\ z'/h \\ 1 \end{Bmatrix} = [T].\begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

The general 4x4 matrix for three-dimensional homogeneous coordinates is expressed as

$$[T] = \begin{bmatrix} A & B & C & P \\ D & E & F & Q \\ G & I & J & R \\ L & M & N & S \end{bmatrix}$$

GEOMETRIC TRANSFORMATIONS OF 3D OBJECTS

The above 4x4 transformation matrix can be partitioned into four separate sections as

$$[T] = \left[\begin{array}{ccc:c} A & B & C & P \\ D & E & F & Q \\ G & I & J & R \\ \hdashline L & M & N & S \end{array}\right]$$

Linear Transformation

A linear transformation transforms an initial linear combination of vectors into the same linear combination of transformed vectors. The upper-left 3x3 submatrix produces a linear transformation corresponding to the scaling, rotation, shear and reflection.

Translation

The upper right 3x1 submatrix produces translation transformation

Perspective Transformation

The lower left 1x3 submatrix produces a perspective transformation

Overall Scaling

The lower right hand 1x1 submatrix produces overall scaling, i.e., all components of position vector are equally scaled.

GEOMETRIC TRANSFORMATIONS OF 3D OBJECTS

1. Translation

In three-dimensional homogeneous coordinates, a point translates from $P(x, y, z, 1)$ to $P_T(x_T, y_T, z_T, 1)$ using the matrix equation

$$\begin{Bmatrix} x_T \\ y_T \\ z_T \\ 1 \end{Bmatrix} = [T_t].\begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.\begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

where $[T_t]$ is the translation matrix and t_x, t_y and t_z are the translational distances (any real values) along the coordinate axes, respectively.

The matrix eqn. is equivalent to three equations

$$x_T = x + t_x,\ y_T = y + t_y,\ z_T = z + t_z$$

2. Rotation

- Rotation is a rigid body transformation that moves the object without deformation.
- This means that every point on the object is rotated through the same angle.

2. Rotation…

- we must specify the axis of rotation and angle of rotation. Unlike the two-dimensional rotations, wherein rotation occurs in the *xy* plane, the easiest three-dimensional rotation may be taken about the coordinate axes.

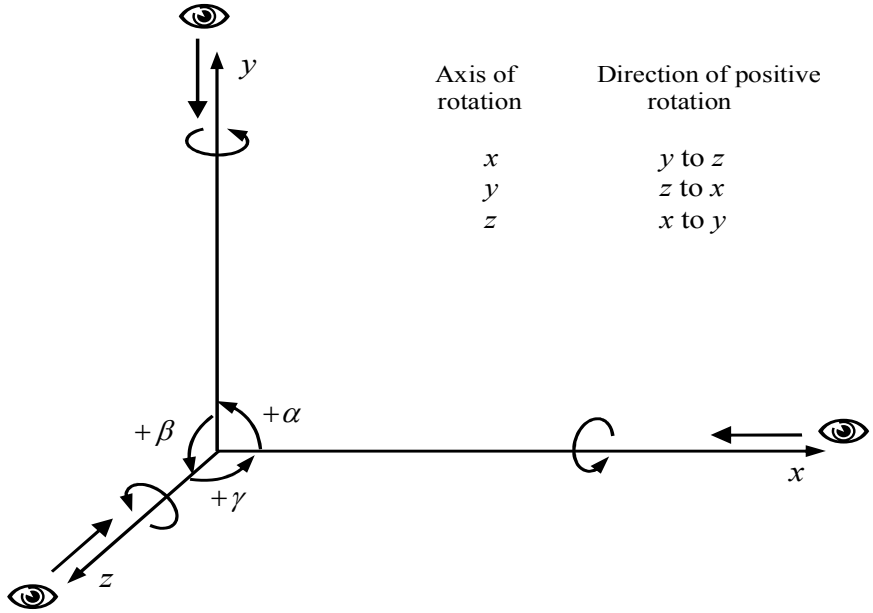
- Rotation of an object about the line, which is parallel to the coordinate axes, is obtained in two steps:
  I. Translate line so that it becomes coincident with any one of the coordinate axes.
  II. Rotate object about the coordinate axes by the desired angle and direction.

- Thus, we can use combinations of coordinate axes rotations (along with appropriate translations) to specify the general rotations of an object.

- In the followings, the most general case of rotation of an object, about any arbitrary line, is considered.

2. Rotation…

Figure shows the right-handed coordinate system to determine the direction of positive rotation.



Axis of rotation	Direction of positive rotation
x	y to z
y	z to x
z	x to y

Positive rotations about the coordinate axes using right hand coordinate system

The two-dimensional z-axis rotation can be extended to the three-dimensions using the equations

$$x_T = x.\cos\alpha - y.\sin\alpha$$

$$y_T = x.\sin\alpha + y.\cos\alpha$$

$$z_T = z$$

Parameter α is the positive rotation angle (ccw) about the z-axis in the xy plane.

2. Rotation…

In homogeneous coordinates, three-dimensional z-axis rotation equation in matrix form is expressed as

$$[T_{rz}] = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation equations for the rotation about the other two axes can be obtained with a cyclic permutation of the coordinate parameters x, y and z in above equation. Thus, we can replace $x \rightarrow y \rightarrow z \rightarrow x$.

For x-axis rotation by an angle β in counterclockwise direction, the eqns. are modified as

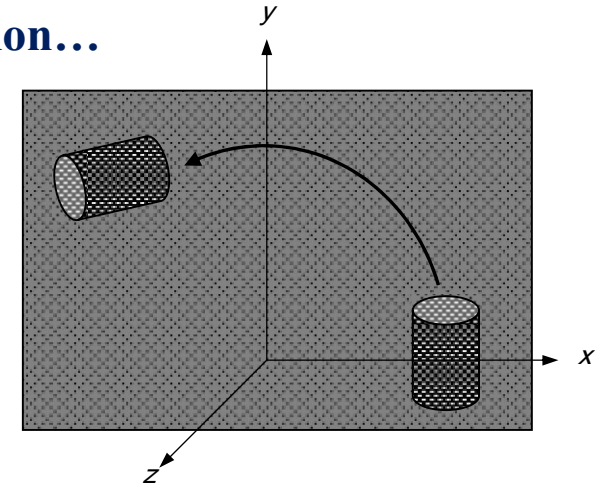
$$y_T = y.\cos\beta - z.\sin\beta$$

$$z_T = y.\sin\beta + z.\cos\beta$$

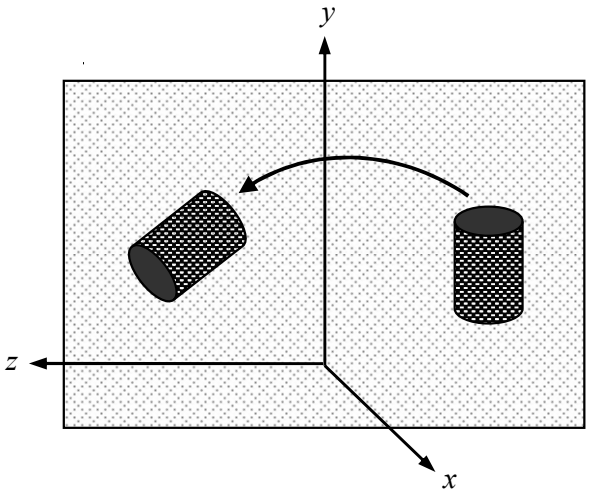
$$x_T = x$$

GEOMETRIC TRANSFORMATIONS OF 3D OBJECTS

2. Rotation…

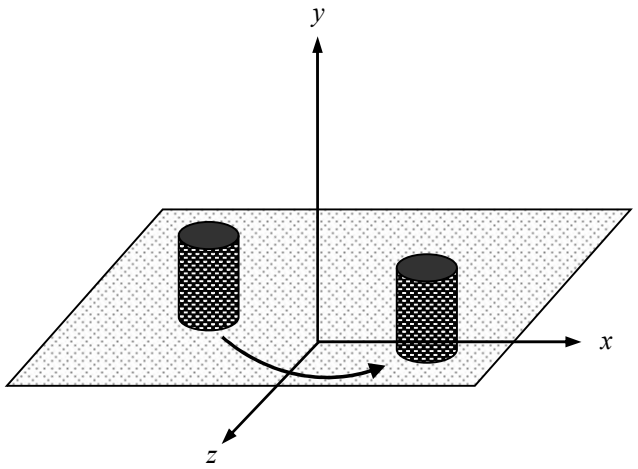


Rotation of an object about the z -axis



Rotation of an object about the x -axis

Rotation of an object about the y -axis



2. Rotation…

In homogeneous coordinates, the matrix equation is expressed as

$$[T_{rx}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For y -axis rotation by an angle θ in counterclockwise direction, the eqns. are modified as

$$z_T = z.\cos\theta - x.\sin\theta$$

$$x_T = z.\sin\theta + x.\cos\theta$$

$$y_T = y$$

In homogeneous coordinates, the matrix equation for y -axis rotation is expressed as

$$[T_{ry}] = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Scaling

Scaling changes the size and position of object relative to origin in the database.

In homogeneous coordinates, the matrix representation for scaling transformation of a point $P(x, y, z)$, relative to the coordinate origin, may be expressed as

$$\begin{Bmatrix} x_T \\ y_T \\ z_T \\ 1 \end{Bmatrix} = [S] \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

where S_x, S_y and S_z are the scaling factors along the coordinate axes, respectively, which can be assigned any positive value.

3. Scaling…

The transformed coordinates for the scaling relative to the coordinate origin may be expressed as

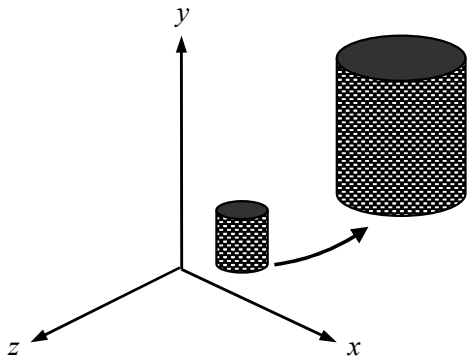
$$x_T = S_x.x,\ y_T = S_y.y,\ z_T = S_z.z$$

Similar to the two-dimensions, the overall scaling for three-dimensional objects can be obtained by assigning the fourth diagonal element, the value other than 1. Hence, matrix equation for the overall scaling may be expressed as

$$\begin{Bmatrix} x' \\ y' \\ z' \\ h \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & S \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix} = \begin{Bmatrix} x \\ y \\ z \\ S \end{Bmatrix}$$

GEOMETRIC TRANSFORMATIONS OF 3D OBJECTS

3. Scaling…



The physical coordinates on $h = 1$ plane, by normalizing the coordinates, is given as

$$\begin{Bmatrix} x_T \\ y_T \\ z_T \\ 1 \end{Bmatrix} = \begin{Bmatrix} x/S \\ y/S \\ z/S \\ 1 \end{Bmatrix}$$

If overall scaling factor $S < 1$, expansion occurs whereas compression occurs if $S > 1$.

4. Reflection

A three-dimensional reflection can be obtained either relative to a reflection axis or through a reflection plane.

Reflection through a given axis is equivalent to 180^o rotation about that axis.

When reflection occurs through a coordinate plane (either xy, yz or xz plane), it can be considered as a conversion between the left-hand and right-hand systems.

The reflection matrix through the xy plane is given as

$$[R_{xy}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similar matrices for the reflections through the xz plane and yz plane, respectively, are

$$[R_{xz}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [R_{yz}] = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5. Shear

- Shear is the controlled distortion of an object model.
- Shear transformations can be used to distort the shape of an object model by sliding of internal layers over the other layers.

In two dimensions, individual and simultaneous shear transformations of x and y coordinates are considered. In three dimensions, shear is extended relative to the z -axis.

If we ignore the shear along the x and y axes, the following transformation produces z -axis shear

$$[Sh_z] = \begin{bmatrix} 1 & 0 & Sh_{13} & 0 \\ 0 & 1 & Sh_{23} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix is equivalent to the following shear equations:

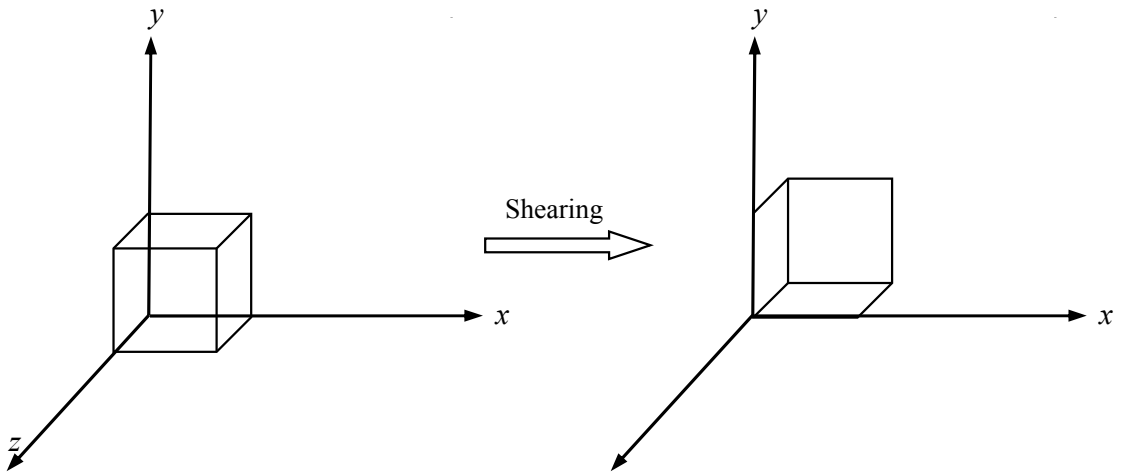
$$x_T = x + Sh_{13}.z$$

$$y_T = y + Sh_{23}.z$$

$$z_T = z$$

5. Shear

Figure shows a unit cube sheared by the z-axis shear with shear parameters $Sh_{13} = Sh_{23} = 1$.



Shearing of unit cube by transformation matrix with $Sh_{13} = Sh_{23} = 1$

Similar matrices for the x-axis shear and the y-axis shear, respectively, may be written as

$$[Sh_x] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ Sh_{21} & 1 & 0 & 0 \\ Sh_{31} & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [Sh_y] = \begin{bmatrix} 1 & Sh_{12} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & Sh_{32} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

COMPOSITE TRANSFORMATIONS

Mathematically, matrix equation for the composite transformation of position vectors $\{X\}$ to $\{X_T\}$ may be expressed as

$$\{X_T\} = [T_4][T_3][T_2][T_1]\{X\} = [T]\{X\}$$

where $\{X_T\}$ = transformed position vectors of the model

$\{X\}$ = original position vectors of the model

$[T_i]$ = scaling, shear, rotation, reflection, translation, projective and perspective transformations matrices

$[T]$ = overall transformation matrix

The transformation matrices will be in the following order of sequence.

Translation / Scaling / Rotation / Reflection / Shear → Perspective → Projective

COMPOSITE TRANSFORMATIONS

If numbers of vertices constituting a three-dimensional object, is subjected to composite transformation then the matrix equation is given as

$$\begin{Bmatrix} x_{T1} & x_{T2} & x_{T3} & - & - & x_{Tn} \\ y_{T1} & y_{T2} & y_{T3} & - & - & y_{Tn} \\ z_{T1} & z_{T2} & z_{T3} & - & - & z_{Tn} \\ 1 & 1 & 1 & 1 & 1 & 1 \end{Bmatrix} = [T] . \begin{Bmatrix} x_1 & x_2 & x_3 & - & - & x_n \\ y_1 & y_2 & y_3 & - & - & y_n \\ z_1 & z_2 & z_3 & - & - & z_n \\ 1 & 1 & 1 & 1 & 1 & 1 \end{Bmatrix}$$

COMPOSITE TRANSFORMATIONS

Example: *Find the transformed coordinates of homogeneous position vector* $(3, 2, 2, 1)$ *subjected to the following successive transformations:*

(i) *Translation by* -1, -1, -1 *along the coordinates axes, and*

30^o *cw rotation about the* x *-axis*

Solution: The translation matrix is expressed as

$$[T_t] = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For $\beta = -30^o$, rotation matrix about the x -axis is expressed as

$$[T_{rx}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.866 & 0.5 & 0 \\ 0 & -0.5 & 0.866 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore, the resulting concatenated matrix is expressed as

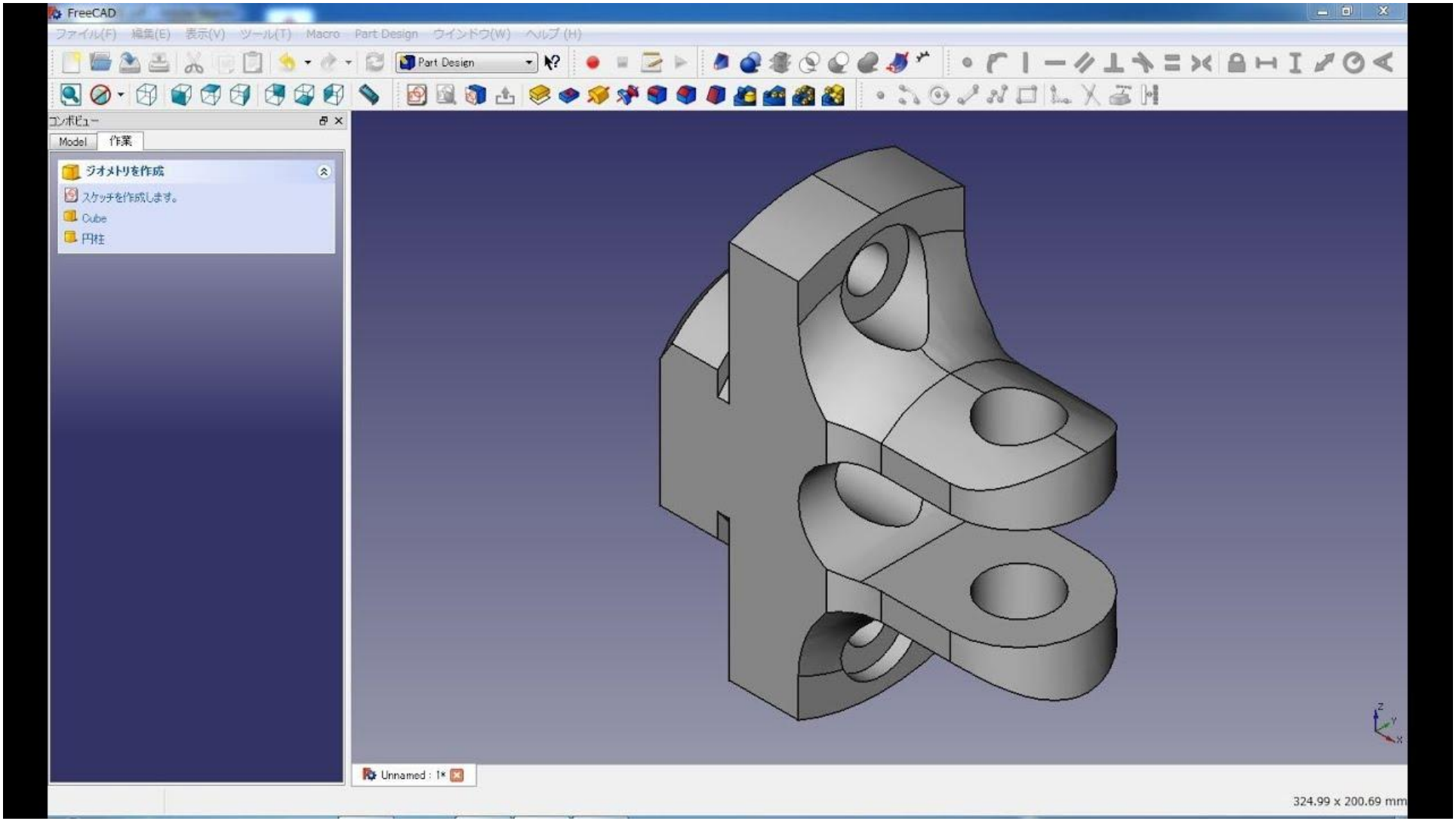
$$[T] = [T_{rx}].[T_t]$$

$$[T]=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0.866 & 0.5 & 0\\ 0 & -0.5 & 0.866 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & -1\\ 0 & 1 & 0 & -1\\ 0 & 0 & 1 & -1\\ 0 & 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}1 & 0 & 0 & -1\\ 0 & 0.866 & 0.5 & -1.366\\ 0 & -0.5 & 0.866 & -0.366\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Hence, transformed position vector is obtained as

$$\begin{Bmatrix}x_T\\ y_T\\ z_T\\ 1\end{Bmatrix}=\begin{bmatrix}1 & 0 & 0 & -1\\ 0 & 0.866 & 0.5 & -1.366\\ 0 & -0.5 & 0.866 & -0.366\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{Bmatrix}3\\ 2\\ 2\\ 1\end{Bmatrix}=\begin{Bmatrix}1\\ 0\\ 0.35\\ 1\end{Bmatrix}$$

Thus, transformed coordinates of homogeneous position vector $(3, 2, 2, 1)$ is $(1, 0, 0.35, 1)$.


FreeCAD
Part Design
Model
作業
ジオメトリを作成
スケッチを作成します。
Cube
円柱
Unnamed : 1*
324.99 x 200.69 mm

THANK YOU