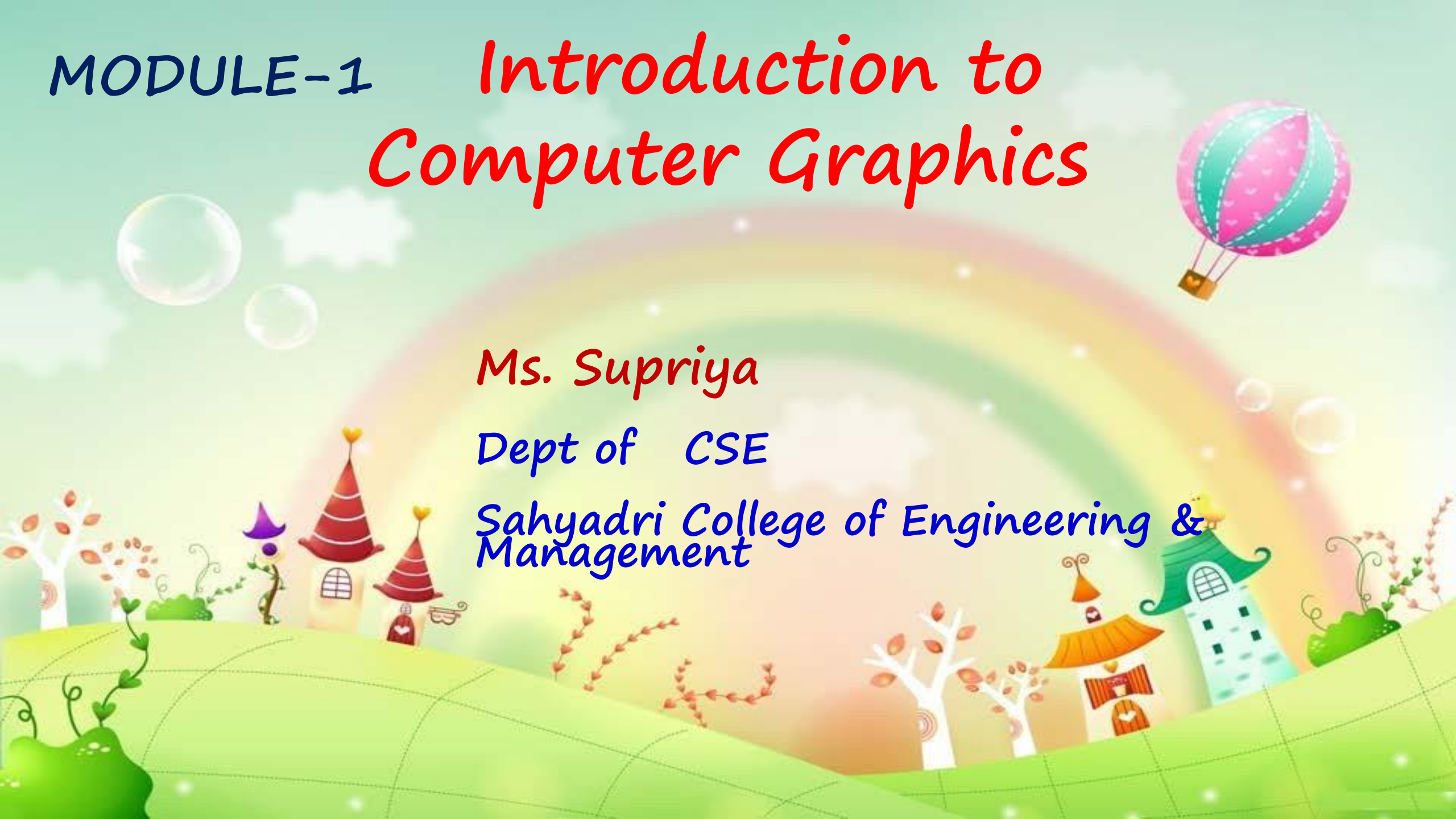


MODULE-1 *Introduction to Computer Graphics*

Ms. Supriya

Dept of CSE

*Sahyadri College of Engineering &
Management*



TOPICS



Overview: Computer Graphics and OpenGL

Computer Graphics: Basics of computer graphics, Application of Computer Graphics, Video Display Devices: Random Scan and Raster Scan displays, graphics software.

OpenGL: Introduction to OpenGL, coordinate reference frames, specifying two-dimensional world coordinate reference frames in OpenGL, OpenGL point functions, OpenGL line functions, point attributes, line attributes, curve attributes, OpenGL point attribute functions, OpenGL line attribute functions, Line drawing algorithms(DDA, Bresenham's), circle generation algorithms (Bresenham's).



What is Computer Graphics?

- Creation, Manipulation and Storage of geometric objects (modelling) & their images (rendering).
- Display those images on screens or hardcopy devices.



Computer Graphics

- Computer graphics deals with all aspects of creating images with a computer
- Applications
 - ❑ Display of information
 - ❑ Design
 - ❑ Simulation and animation
 - ❑ User interfaces



Display of Information



- Graphics techniques are the medium to convey information among people.
- Computer graphics enable the user to interpret the digital information using pictures
- Many fields such as fluid flow, molecular biology, and mathematics, the information has been represented as images using graphical tools



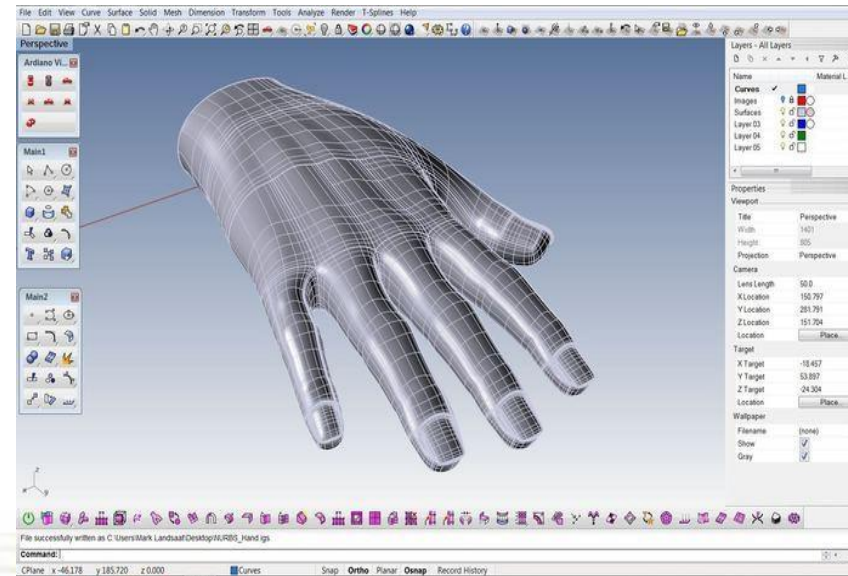
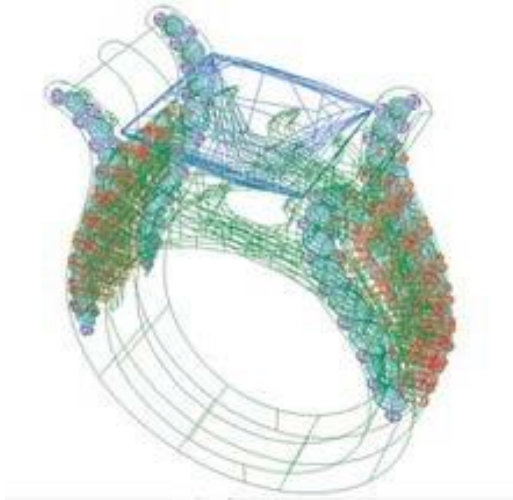


Design



- The computer graphics helped professionals in the area of medicine, engineering, entertainment, mathematics, architecture etc.
- Graphical tools provide the pictorial representation for the designers to analyze and represent the design.
 - ❑ *Cost-effective and efficient process*
 - ❑ *Eg: Computer Aided Design (CAD) tool helps to analyze and design mechanical parts, VLSI circuits etc.*



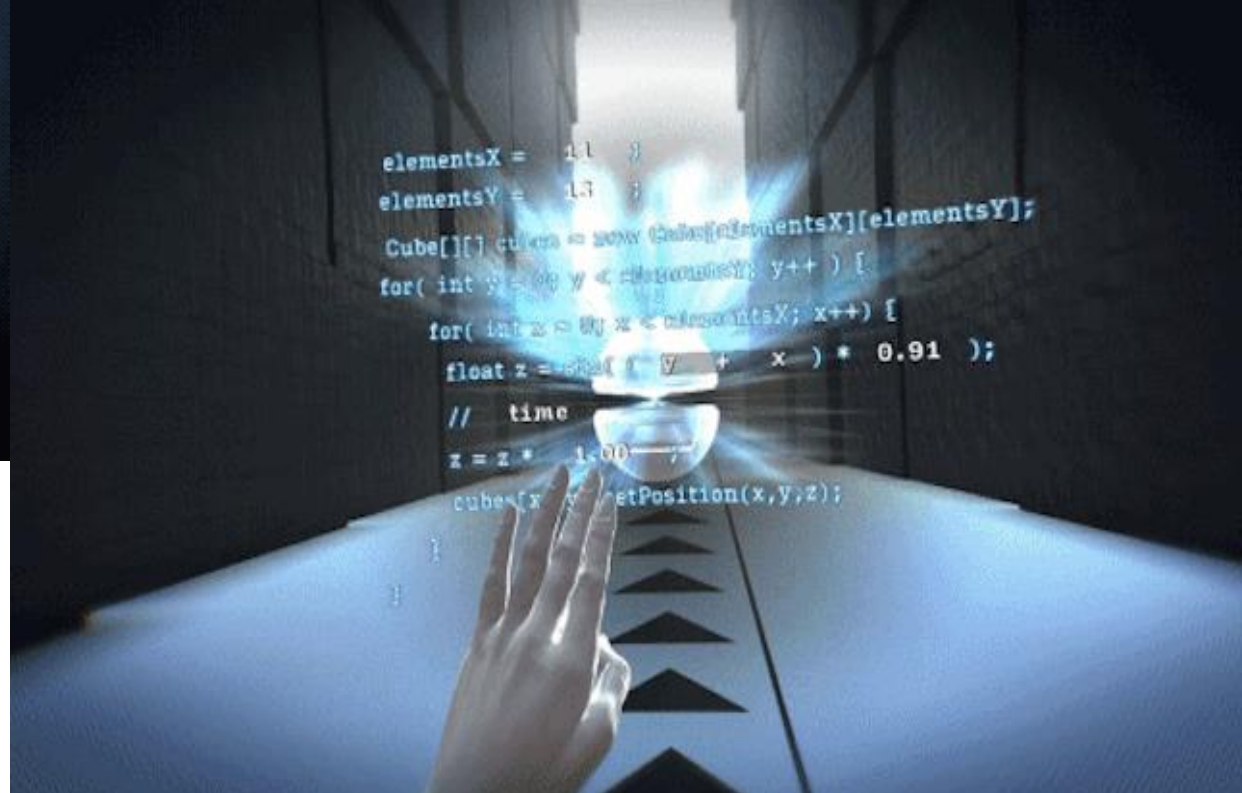
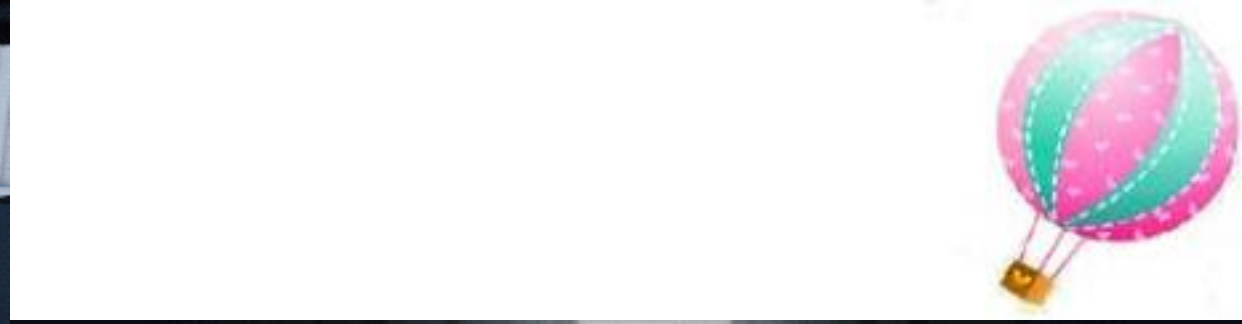


Simulation and Animation



- Computer graphics can be used as simulators
 - ❑ Flight simulators for pilot training , Training vehicle drivers
- Computer graphics are used for animation in the television, motion-picture, and advertising industries
 - ❑ Cost-effective
- Computer graphics led to Virtual reality simulations and Video Games.
 - ❑ Virtual reality simulations helps to train a surgical intern might to do an operation





User Interfaces



- Computer graphics enabled the creation of simple, powerful, understandable and easily usable GUI's (Graphical User Interface) for the user.
 - ❑ Most of operating systems such as Microsoft Windows, the Macintosh OS, Ubuntu provide the GUI with icons, menus, lists etc.
 - ❑ Web browsers, such as Firefox, Chrome, Safari, and Internet Explorer provide the simple and portable GUI.



Areas of Computer Graphics Applications

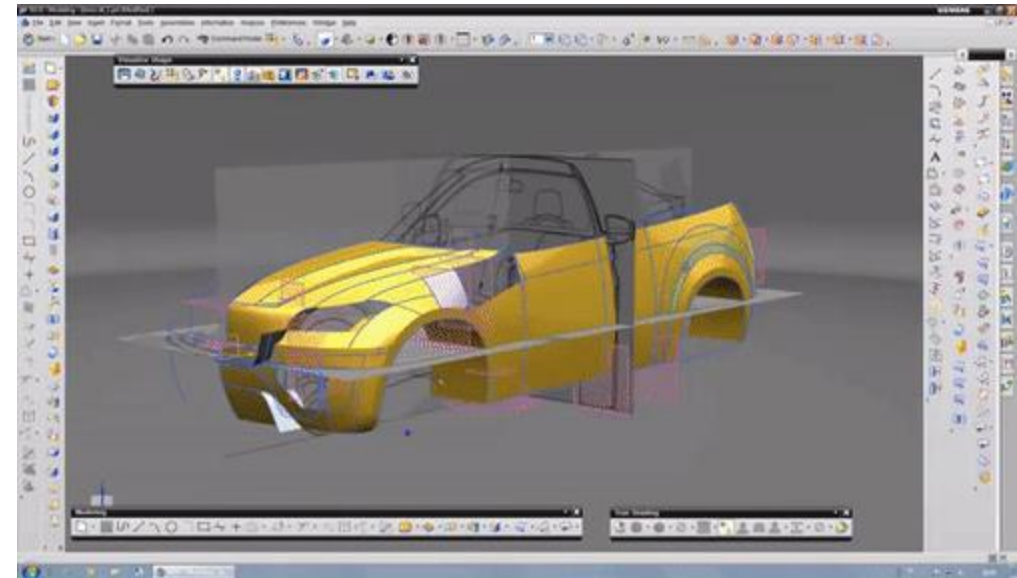
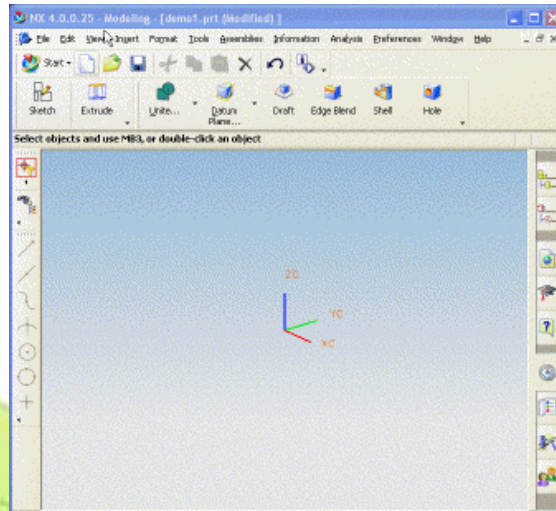
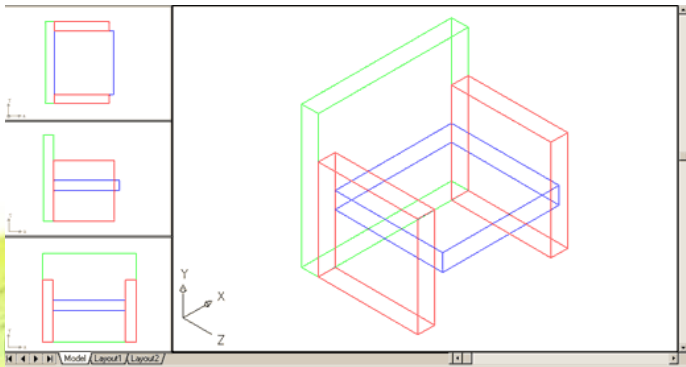


- Computer Aided Design (CAD)
- Presentation Graphics
- Computer Art
- Entertainment (animation, games, ...)
- Education & Training
- Visualization (scientific & business)
- Image Processing
- Graphical User Interfaces

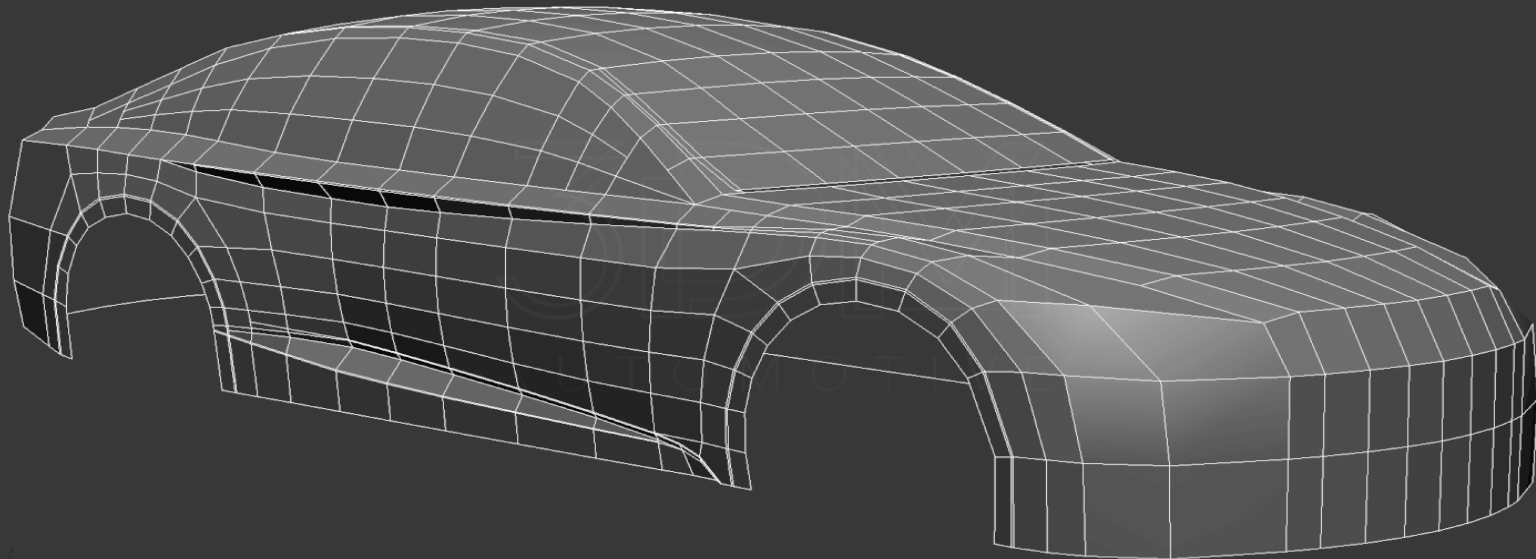
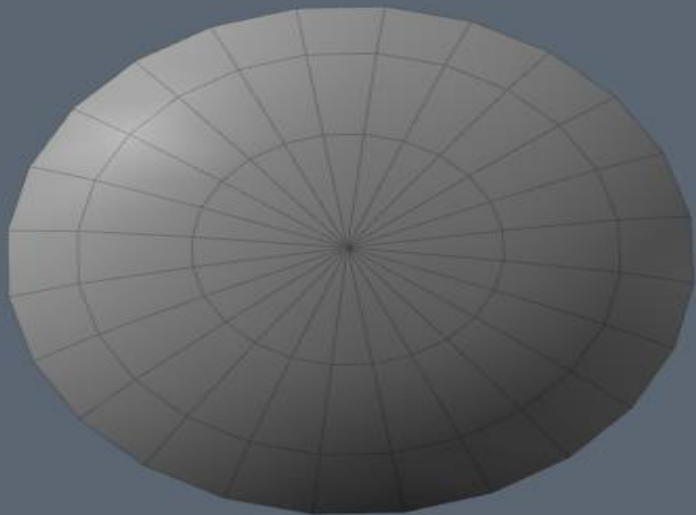


1. Computer Aided Design (CAD)

- Used in design of buildings, automobiles, aircraft, watercraft, spacecraft, computers, textiles & many other products
- Objects are displayed in wire frame outline form
- Software packages provide multi-window environment



- Graphics design package provides standard shapes (useful for repeated placements)
- Animations are also used in CAD applications
- Realistic displays of architectural design permits simulated “walk” through the rooms (virtual -reality systems)



2.Presentation Graphics



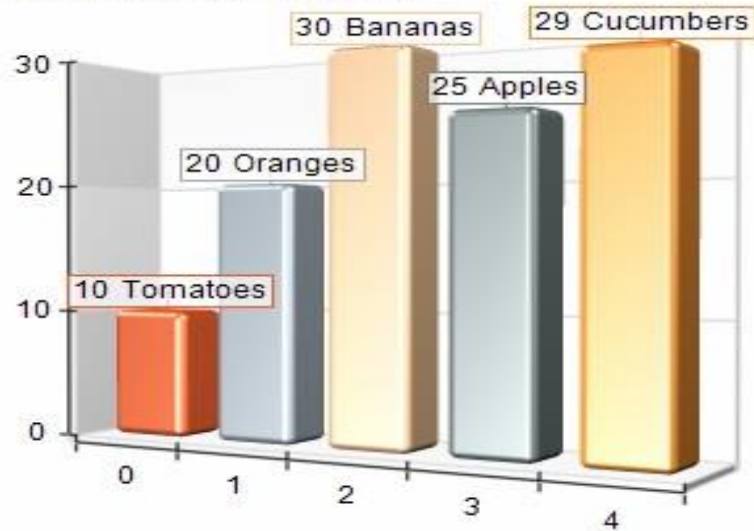
- Used to produce illustrations for reports or generate slides for use with projectors
- Commonly used to summarize financial, statistical, mathematical, scientific, economic data for research reports, managerial reports & customer information bulletins
- Examples : Bar charts, line graphs, pie charts, surface graphs, time chart



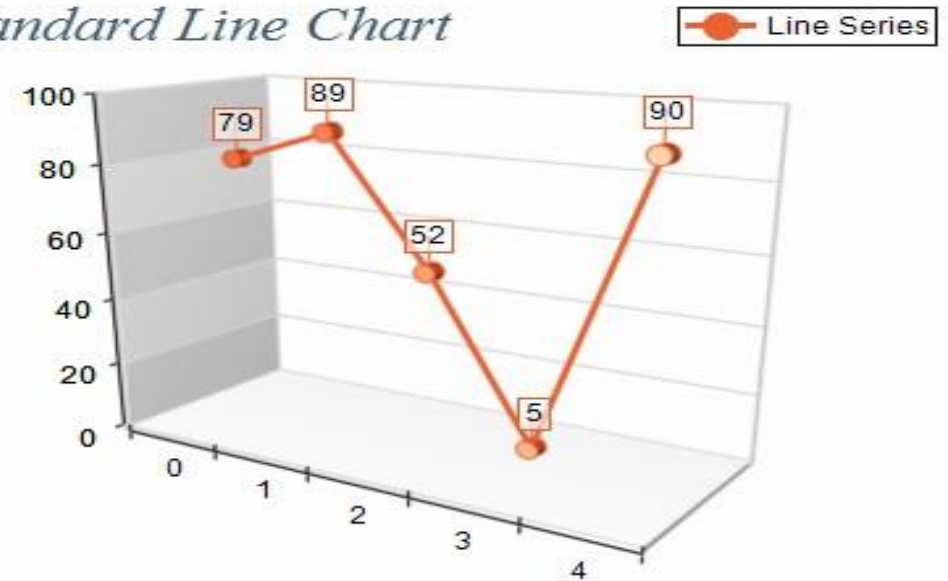
Examples of presentation graphics



Standard Bar Chart



Standard Line Chart



Examples of presentation graphics



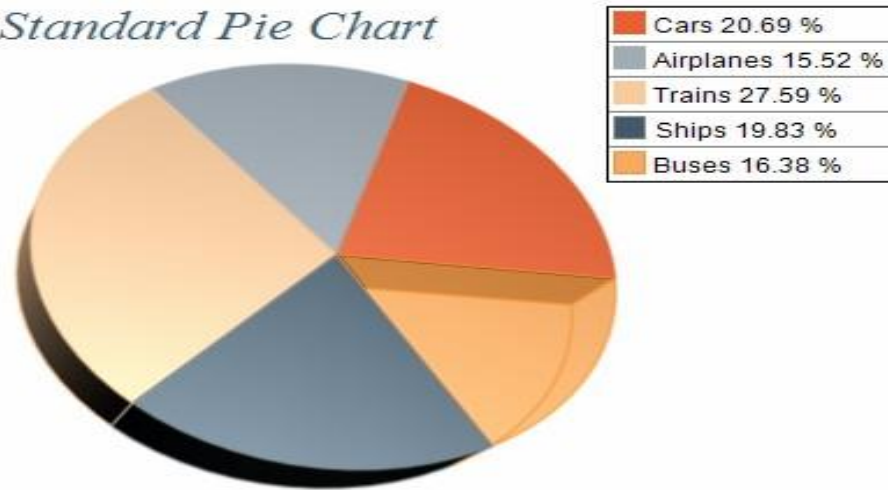
Geologic Time Chart

Era	Period	Epoch	Informal Geologic Time Terms		Mountain Glaciations	Estimated Age* (years before present)
Cenozoic (part)	Quaternary	Holocene				
		Pleistocene	late Pleistocene		_____ ? _____ Pinedale glaciation	_____ ≈11,680 _____
					_____ ? _____ _____ ≈35,000 _____	
			middle Pleistocene	late	Bull Lake glaciation	_____ ≈128,000 _____
				middle	Pre-Bull Lake glaciation	_____ ≈310,000 _____
				early		_____ ≈640,000 _____
	early Pleistocene		_____ ≈778,000 _____			
	Tertiary (part)	Pliocene				_____ ≈1,806,000 _____
Miocene (part)					_____ ≈5,300,000 _____	

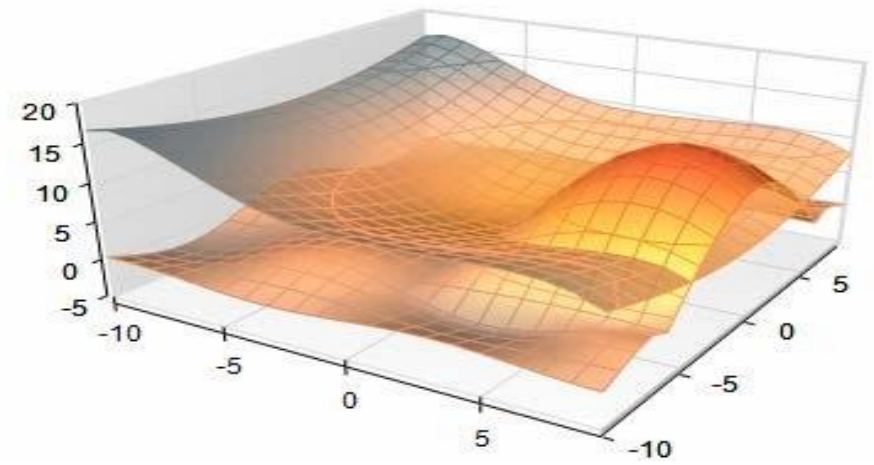
Examples of presentation graphics



Standard Pie Chart



Intersected Surfaces



3.Computer Art

- Used in fine art & commercial art
 - ❑ Includes artist's paintbrush programs, paint packages, CAD packages and animation packages
 - ❑ These packages provides facilities for designing object shapes & specifying object motions.
 - ❑ Examples : Cartoon drawing, paintings, product advertisements, logo design



Computer Art

- Electronic painting
 - ❑ Picture painted electronically on a graphics tablet (digitizer) using a stylus
 - ❑ Cordless, pressure sensitive stylus
- Morphing
 - ❑ A graphics method in which one object is transformed into another



4. Entertainment



➤ *Movie Industry*

- ☐ *Used in motion pictures, music videos, and television shows.*
- ☐ *Used in making of cartoon animation films*

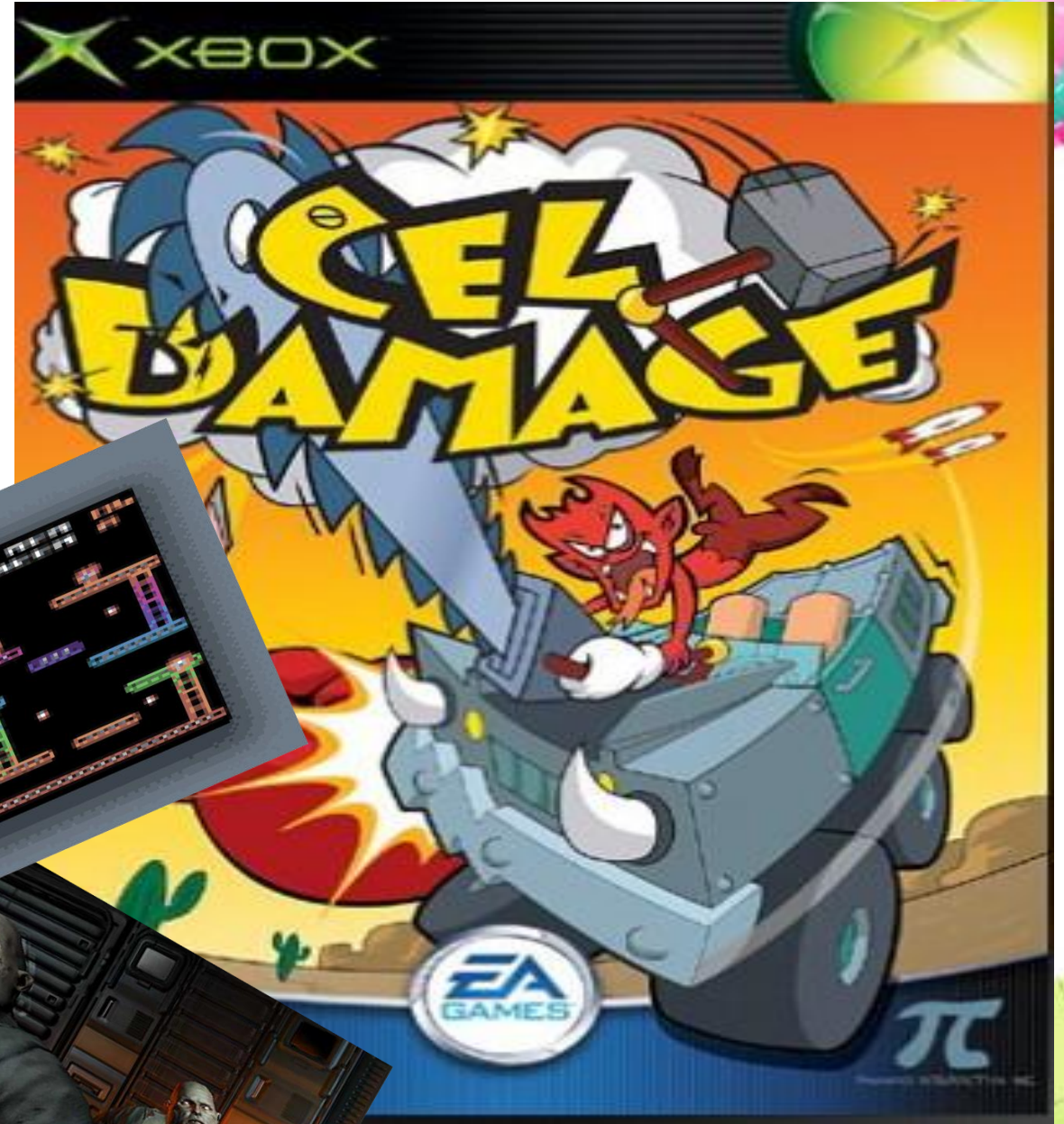


Computer Graphics is about animation (films)



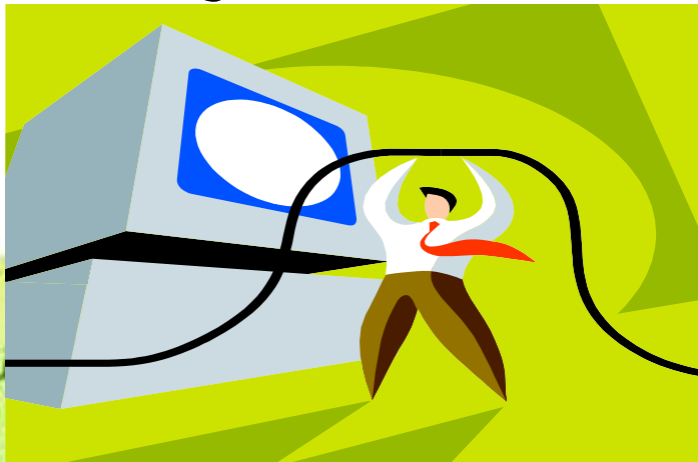
➤ Game Industry

- ❑ Focus on interactivity
- ❑ Cost effective solutions
- ❑ Avoiding computations and other tricks



5. Education & Training

- Computer generated models of physical, financial and economic systems are used as educational aids.
- Models of physical systems, physiological systems, population trends, or equipment such as color-coded diagram help trainees understand the operation of the system



- Specialized systems used for training applications
 - simulators for practice sessions or training of ship captains
 - aircraft pilots
 - heavy equipment operators
 - air traffic-control personnel



6. Visualization

➤ Scientific Visualization

- ❑ Producing graphical representations for scientific, engineering, and medical data sets.
- ❑ To check the behaviour of certain process different fields like engineering, scientists business analysts etc need appropriate visualization.



- Business Visualization is used in connection with data sets related to commerce, industry and other non-scientific areas
- Techniques used- color coding, contour plots, graphs, charts, surface renderings & visualizations of volume interiors.
- Image processing techniques are combined with computer graphics to produce many of the data visualizations



➤ To apply image processing methods

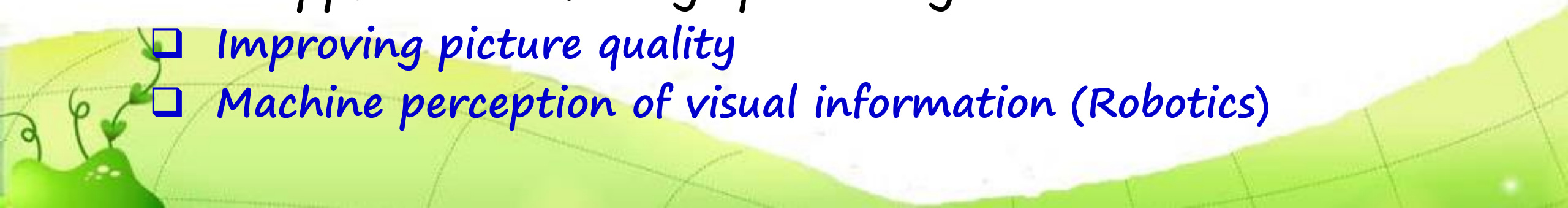
- ❑ Digitize a photograph (or picture) into an image file
- ❑ Apply digital methods to rearrange picture parts to
 - enhance color separations
 - Improve quality of shading
- ❑ Tomography – technique of X-ray photography that allows cross-sectional views of physiological systems to be displayed
- ❑ Computed X-ray tomography (CT) and position emission tomography (PET) use projection methods to reconstruct cross sections from digital data
- ❑ Computer-Aided Surgery is a medical application technique to model and study physical functions to design artificial limbs and to plan & practice surgery



7. Image Processing



- CG- Computer is used to create a picture
- Image Processing – applies techniques to modify or interpret existing pictures such as photographs
- Medical applications
 - ❑ Picture enhancements
 - ❑ Tomography
 - ❑ Simulations of operations
 - ❑ Ultrasonics & nuclear medicine scanners
- Main applications of image processing
 - ❑ Improving picture quality
 - ❑ Machine perception of visual information (Robotics)

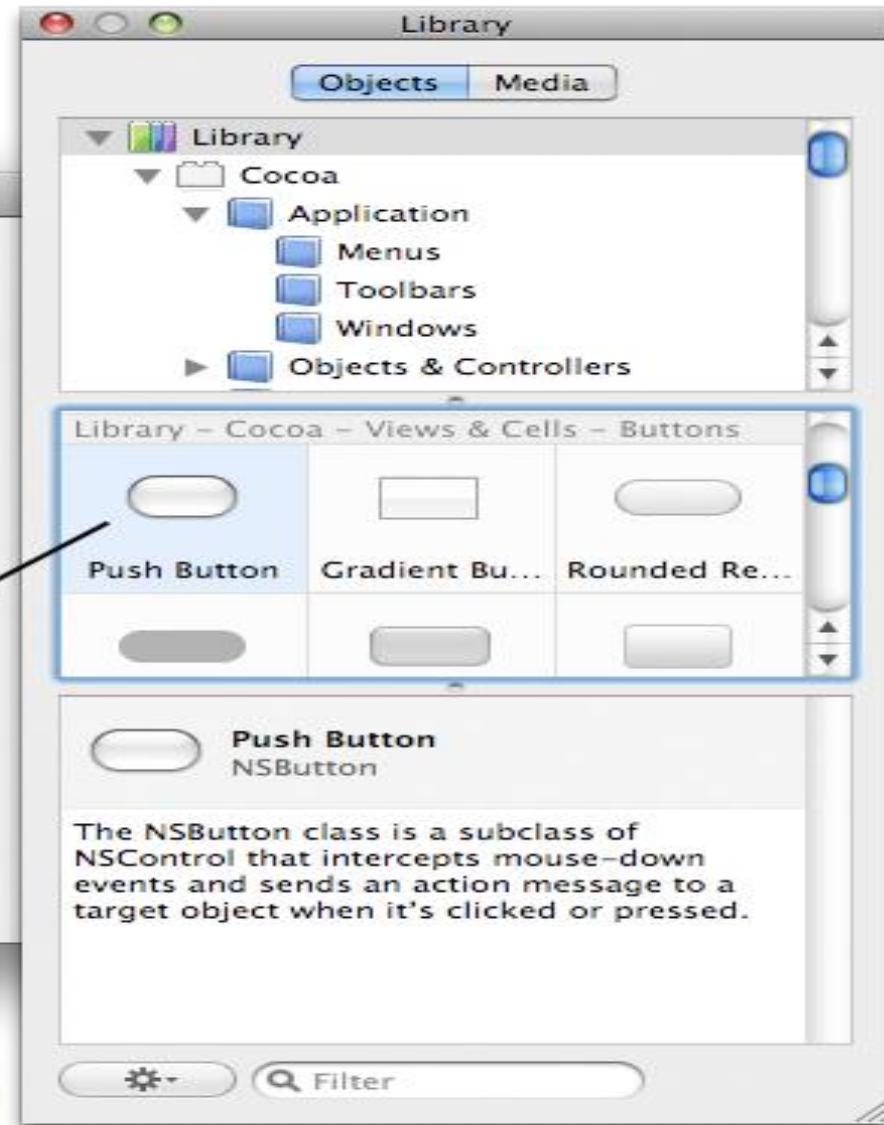
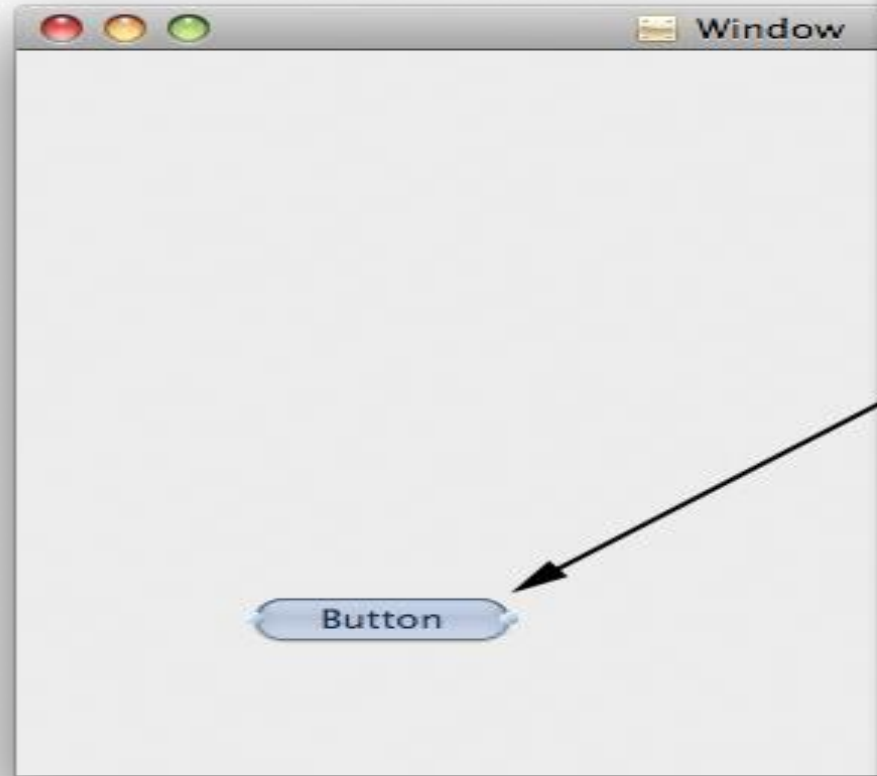


8. Graphical User Interfaces



- Major component – Window manager (multiple-window areas)
- To make a particular window active, click in that window (using an interactive pointing device)
- Interfaces display – menus & icons
- Icons – graphical symbol designed to look like the processing option it represents
- Advantages of icons – less screen space, easily understood
- Menus contain lists of textual descriptions & icons





Raster Scan Display



- Based on the intensity control of pixels in the form of a rectangular box called Raster on the screen
- In this, the electron beam is swept across the screen, one row at a time from top to bottom
- As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots

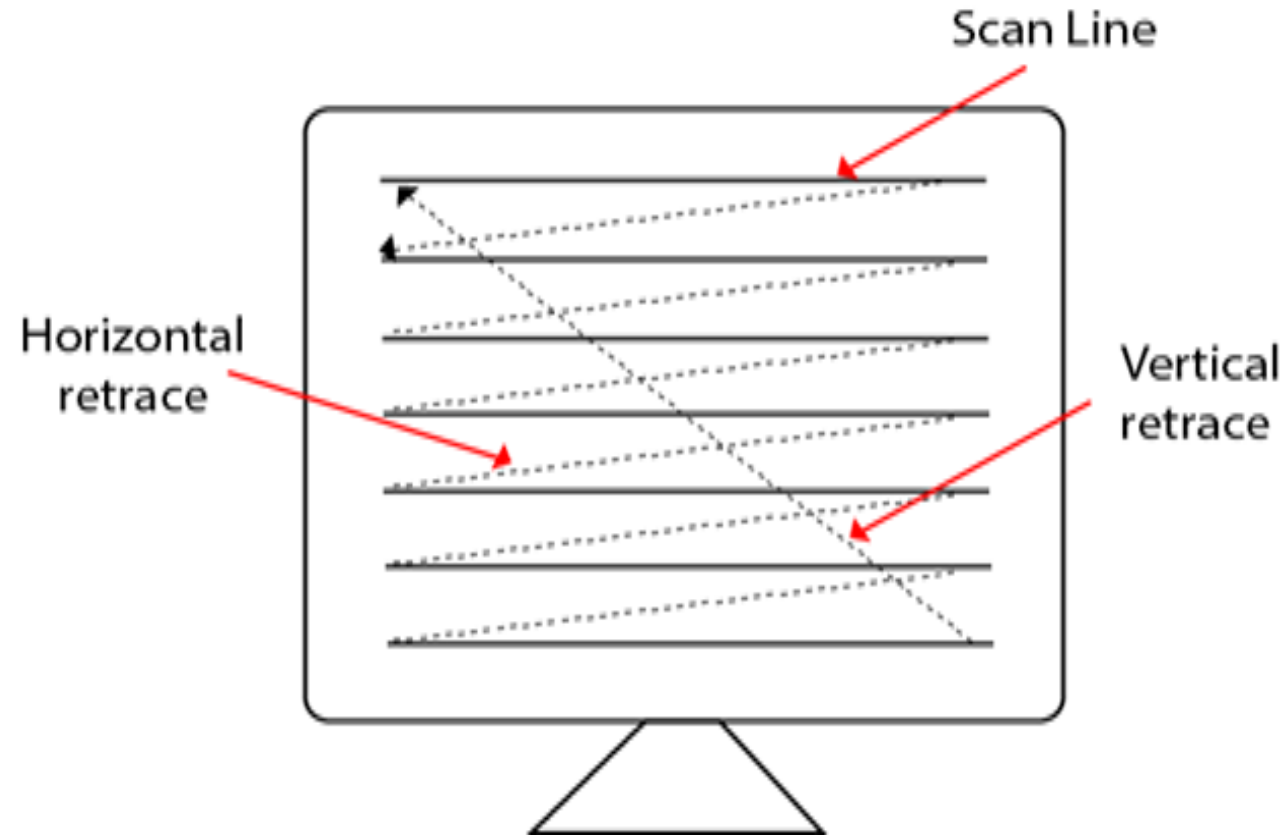


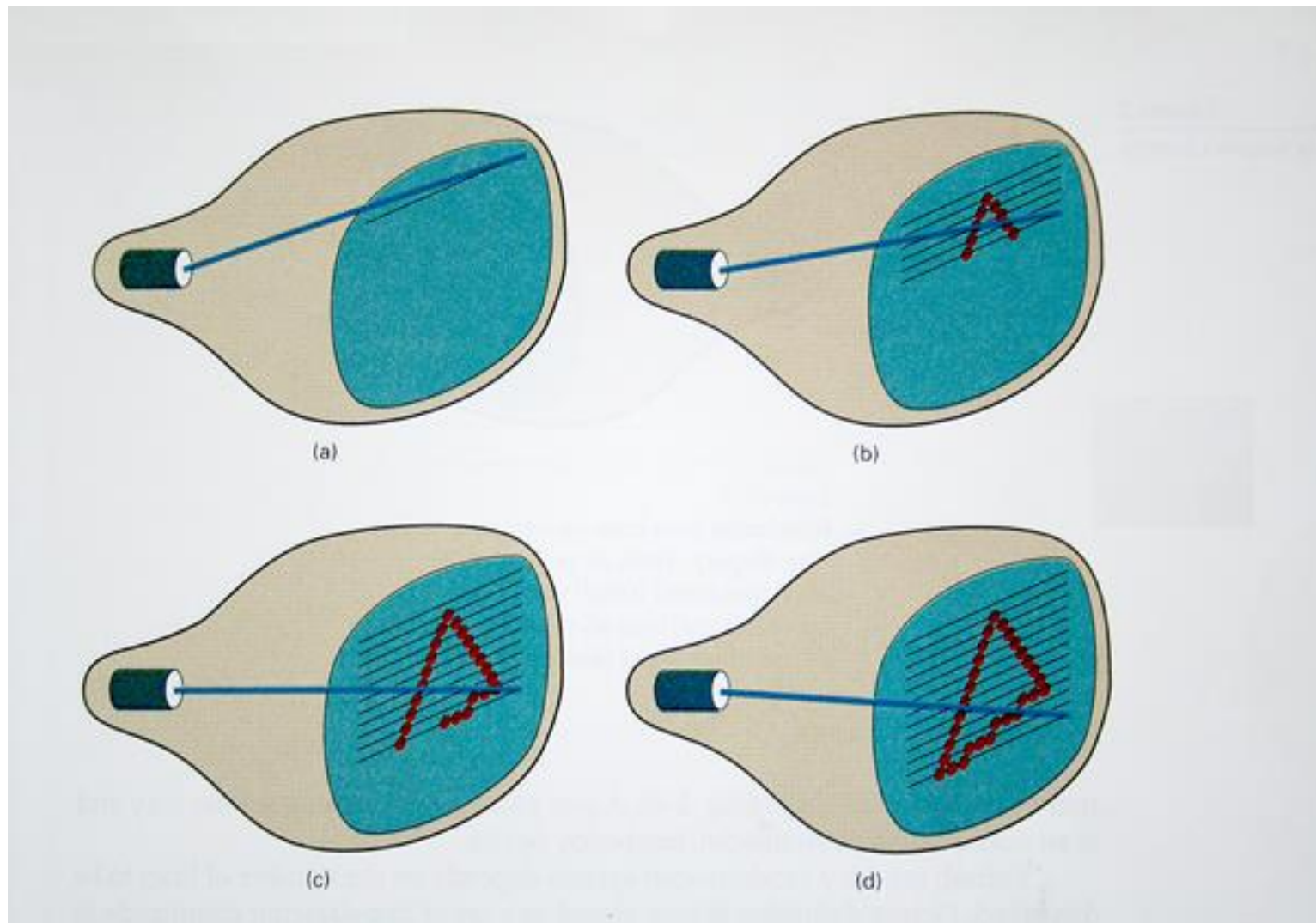
- Picture definition is stored in memory area called the **Refresh Buffer** or **Frame Buffer**
- This memory area holds the set of intensity values for all the screen points
- Stored intensity values are then retrieved from the refresh buffer and “painted” on the screen one row scan line at a time
- Each screen point is referred to as a **pixel**
- At the end of each scan line, the electron beam returns to the left side of the screen to begin displaying the next scan line.



Beam refreshing is of two types

- Horizontal retracing
- Vertical retracing







Types of Scanning or travelling of beam in Raster Scan

- Interlaced Scanning
- Non-Interlaced Scanning





Advantages:

Realistic image

Million Different colors to be generated

Shadow Scenes are possible

Disadvantages:

Low Resolution

Expensive



Random-Scan Display

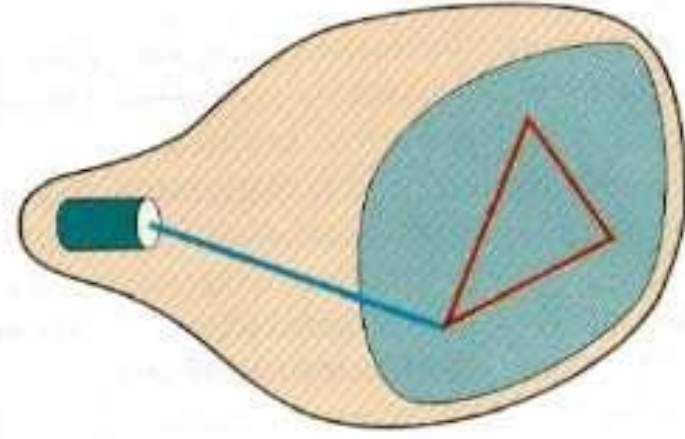
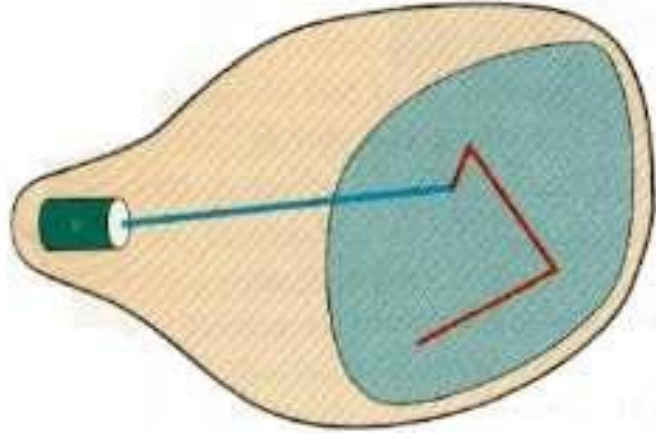
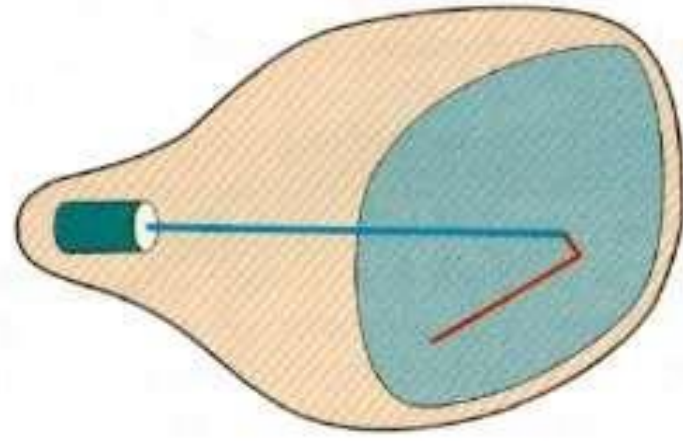
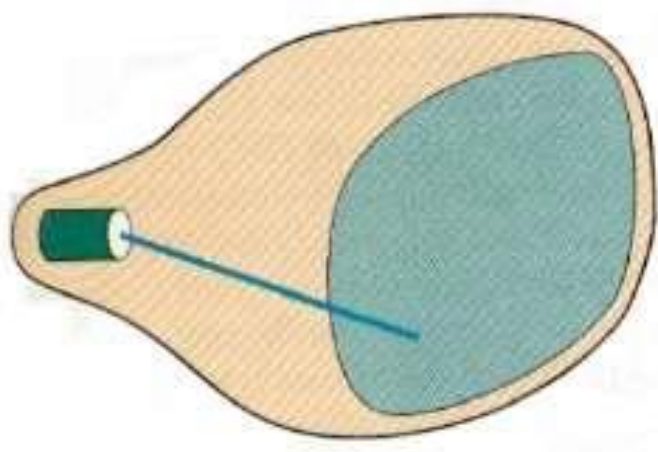
Make use geometrical primitives such as points, lines, curves, and polygons, which are all based upon mathematical equation

A CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn

Random-scan monitors draw a picture one line at a time and for this reason are also referred to as vector displays (or stroke-writing or calligraphic displays).

It can draw and refresh component lines of a picture in any specified sequence





Refresh rate depends on the number of lines to be displayed

Picture definition is now stored as a line-drawing commands an area of memory referred to as refresh display file (display list)

To display a picture, the system cycle through the set of commands in the display file, drawing each component line

The system returns back to first line command in the list, after all the drawing commands have been processed.

Random scan displays are designed to draw all the component lines of a picture 30 to 60 times each second



Random-Scan Display Processors:



Input in the form of an application program is stored in the system memory along with graphics package.

Graphics package translates the graphic commands in application program into a display file stored in system memory.

This display file is then accessed by the display processor to refresh the screen. The display processor cycles through each command in the display file program.

Sometimes the display processor in a random-scan is referred as *Display Processing Unit / Graphics Controller*.



- A Raster system produces jagged lines that are plotted as discrete points sets
- Vector displays produce smooth line drawing
- Random scan displays are designed for line-drawing applications and can not display realistic shaded scenes
- Random scan displays have higher resolution than raster systems.





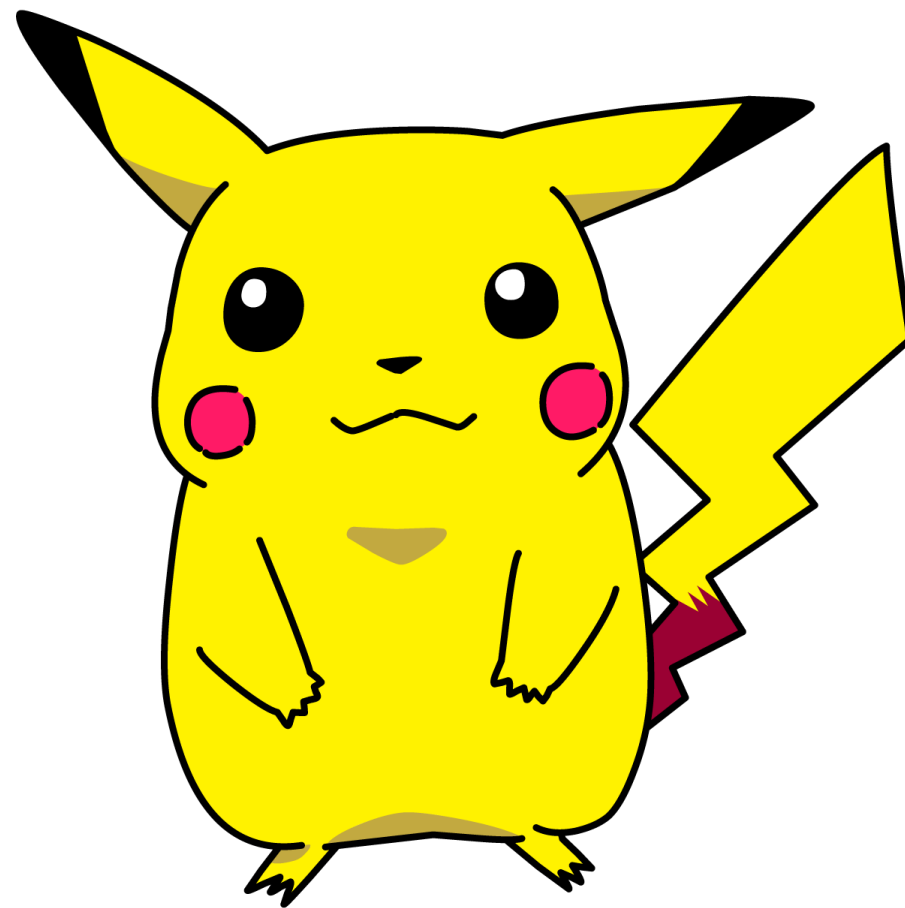
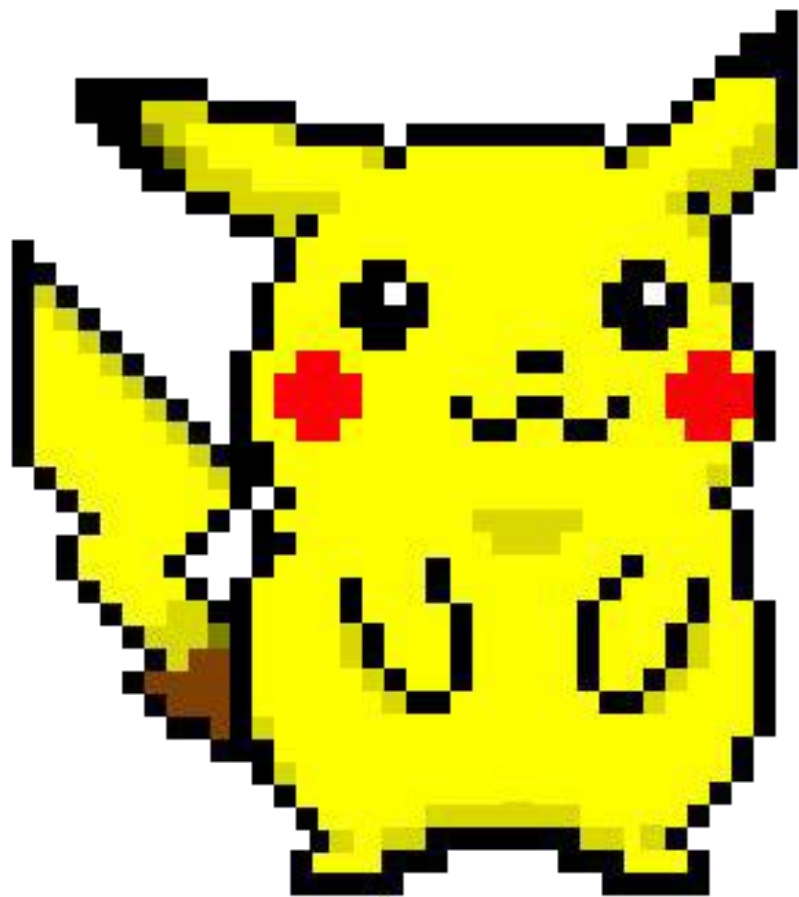
ADVANTAGES:

- Higher resolution as compared to raster scan display.
- Produces smooth line drawing.
- Less Memory required.

DISADVANTAGES:

- Realistic images with different shades cannot be drawn.
- Color limitations.





Difference	Raster scan displays	Random scan displays
resolution	It has poor or less Resolution because picture definition is stored as a intensity value.	It has High Resolution because it stores picture definition as a set of line commands.
Electron beam	Electron Beam is directed from top to bottom and one row at a time on screen, but electron beam is directed to whole screen.	Electron Beam is directed to only that part of screen where picture is required to be drawn, one line at a time so also called Vector Display .
Cost	It is less expensive	It is Costlier than Raster Scan System.
Refresh rate	Refresh rate is 60 to 80 frame per second .	Refresh Rate depends on the number of lines to be displayed i.e 30 to 60/sec
Picture definition	It Stores picture definition in Refresh Buffer also called Frame Buffer .	It Stores picture definition as a set of line commands called Refresh Display File .
Line drawing	Zig – Zag line is produced because plotted value are discrete.	Smooth line is produced because directly the line path is followed by electron beam
Realism in display	It contains shadow, advance shading and hidden surface technique so gives the realistic display of scenes.	It does not contain shadow and hidden surface technique so it can not give realistic display of scenes .
Image drawing	It uses Pixels along scan lines for drawing an image.	It is designed for line drawing applications and uses various mathematical function to draw.



Graphics packages

A set of libraries that provide programmatically access to some kind of graphics 2D functions



➤ Types

- ☐ GKS – Graphical Kernel System – first graphics package – accepted by ISO & ANSI
- ☐ PHIGS – Programmer’s Hierarchical Interactive Graphics Standard – accepted by ISO & ANSI
- ☐ PHIGS + (Expanded package)
- ☐ Silicon Graphics GL (Graphics Library)
- ☐ Open GL
- ☐ Pixar Render Man interface
- ☐ Postscript interpreters
- ☐ Painting, drawing, design packages



Computer Graphics Software



- Overview of graphics software systems
 - two broad classifications for computer-graphics software
 - Special purpose packages
 - General programming packages





➤ special purpose packages

- ❑ Designed for nonprogrammers who want to generate pictures, graphs, or charts in some application area without worrying about the graphics procedures that might be needed to produce such displays.
- ❑ The interface is a set of menus that allows users to communicate with the programs in their own terms.
- ❑ Examples of such applications include artist's painting programs and various architectural, business, medical, and engineering CAD systems





➤ General programming packages

- ❑ Provides a library of graphics functions that can be used in a programming language such as C, C++, Java, or Fortran.
- ❑ Basic functions in a typical graphics library include those for specifying picture components
 - Drawing straight lines, polygons, spheres, and other objects, setting color values, selecting views of a scene, and applying rotations or other transformations.
- ❑ Eg: GL (Graphics Library), OpenGL, VRML (Virtual-Reality Modeling Language), Java 2D, and Java 3D

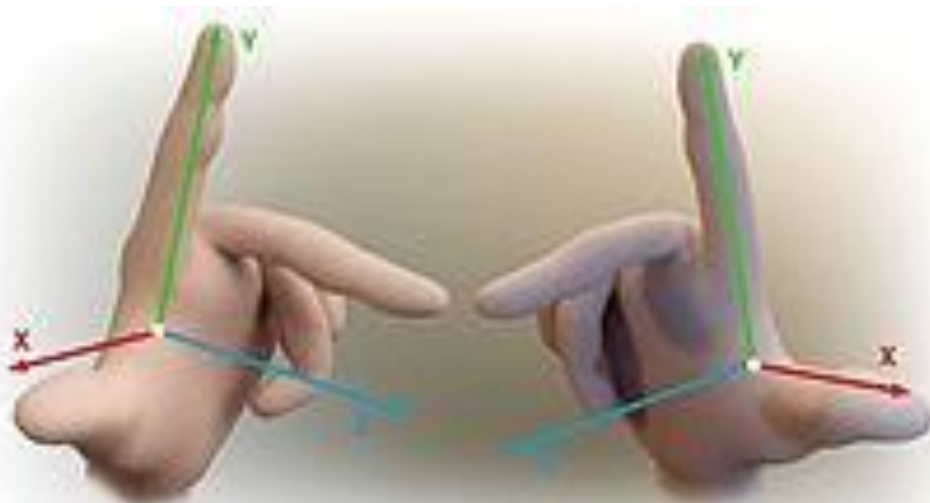


Coordinate Representations



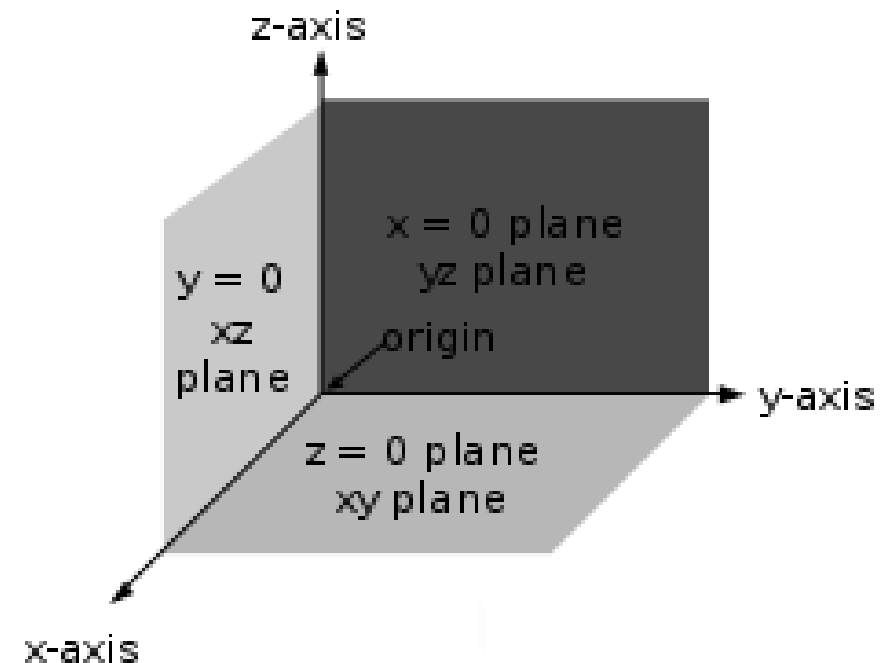
- To generate a picture using a programming package, **geometric descriptions** of the objects are required
- These descriptions determine the **locations** and **shapes** of the objects.
 - ❑ A box is specified by the positions of its corners (vertices)
 - ❑ A sphere is defined by its center position and radius.
- With few exceptions, general graphics packages require geometric descriptions to be specified in a standard, right-hand, **Cartesian-coordinate** reference frame.

- If coordinate values for a picture are given in some other reference frame, they must to be converted to Cartesian-coordinate before they can be input to the graphics package.



Left Handed Coordinates

Right Handed Coordinates



The right-handed Cartesian coordinate system indicating the coordinate plane

➤ Several different Cartesian reference frames are used in the process of constructing and display a scene.



□ Modeling coordinates/local coordinates /master coordinates

- These are used to define the shapes of individual objects, such as tree, within a separate coordinate reference frame for each objects.

□ World coordinates

- The universal coordinate system used to model a scene by placing the objects into appropriate locations within a scene reference frame
- It involves the transformation of the individual modeling-coordinate frames to specified positions and orientations within the world-coordinate frame





➤ Eg: Drawing a Car

- ❑ Initially the all the parts of cars (wheels, frame, seat, steering, gears, mirror, windows) are defined in a separate modeling coordinate frame.
- ❑ Then, the car is constructed by fitting together the component parts in world coordinates.





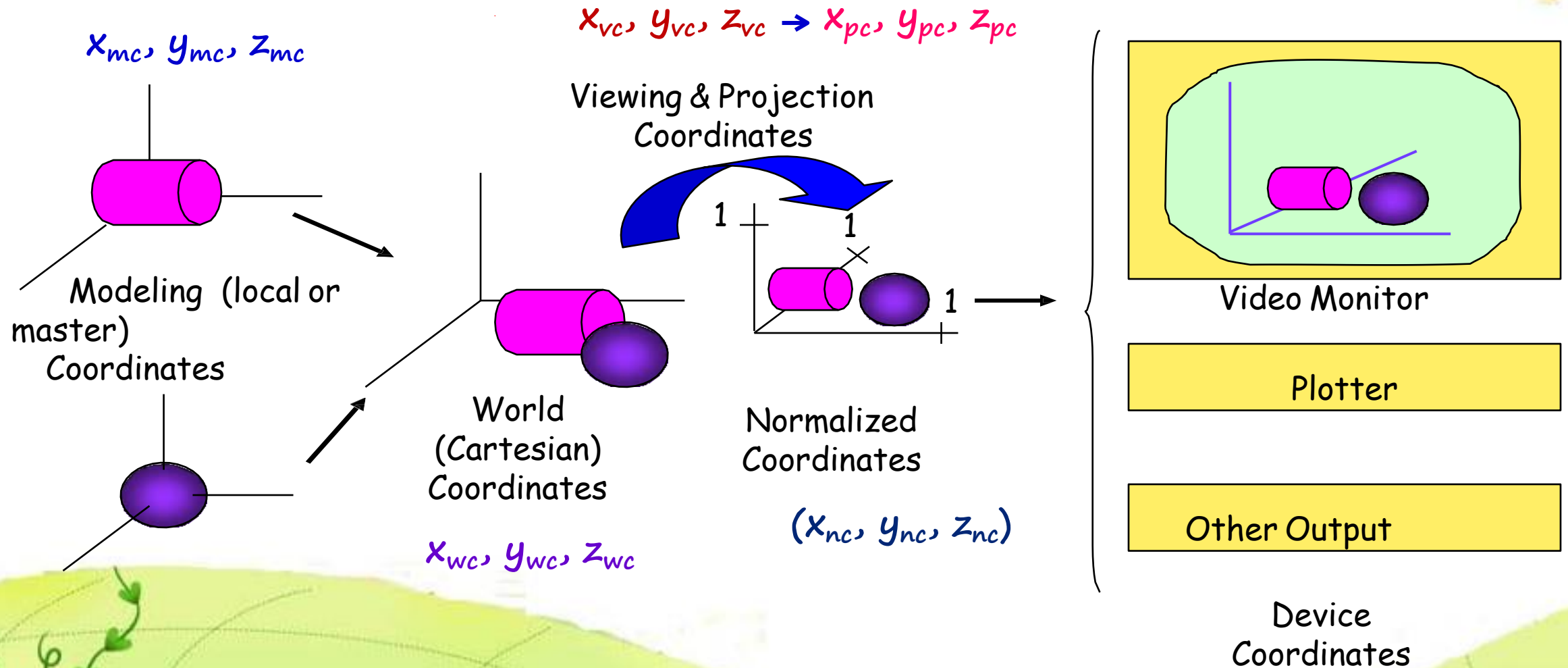
- After all parts of a scene have been specified, the overall world coordinate description is processed through various routines onto one or more **output-device** reference frames for display. This process is called the **viewing pipeline**.
- World coordinate positions are first converted to **viewing coordinates** corresponding to the view **we want** of a scene, based on the position and orientation of a hypothetical camera.





- ❑ Then object locations are transformed to a **2D projection** of the scene, which corresponds to what we will see on the output device.
- ❑ The scene is then stored in **normalized coordinates**, which each coordinate value is in the range from -1 to 1 or in the range from 0 to 1. (**normalize device coordinates**)
- ❑ Finally, the picture is scan converted into the **refresh buffer** of a raster system for display.
- ❑ The coordinate systems for display devices are generally called **device coordinates**, or **screen coordinates** in the case of a video monitor.

Coordinate representations





- An initial **modeling-coordinating** position (x_{mc}, y_{mc}, z_{mc}) is transferred to **world coordinates**, then to **viewing and projection coordinates**, then to **left-handed normalized coordinates**, and finally to a **device-coordinate** position (x_{dc}, y_{dc}) with the sequences:

$$(x_{mc}, y_{mc}, z_{mc}) \rightarrow (x_{wc}, y_{wc}, z_{wc}) \rightarrow (x_{vc}, y_{vc}, z_{vc}) \rightarrow (x_{pc}, y_{pc}, z_{pc}) \\ \rightarrow (x_{nc}, y_{nc}, z_{nc}) \rightarrow (x_{dc}, y_{dc})$$

- Device coordinates (x_{dc}, y_{dc}) are integer within the range $(0,0)$ to (x_{max}, y_{max}) for a particular output device.

Graphics Functions



➤ Graphics output primitives

- ❑ basic building blocks for pictures
- ❑ include character strings and geometric entities, such as points, straight lines, curved lines, filled color areas (usually polygons), and shapes defined with arrays of color points
- ❑ some graphics packages provide functions for displaying more complex shapes such as spheres, cones, and cylinders.





➤ Attributes

- ❑ Properties of the output primitives
- ❑ An attribute describes how a particular primitive is to be displayed.
- ❑ Includes color specifications, line styles, text styles, and area-filling patterns.

➤ geometric transformations

- ❑ Are used to change the size, position, or orientation of an object within a scene





➤ Viewing transformations

- ❑ Used to select a view of the scene, the type of projection to be used, and the location on a video monitor where the view is to be displayed.

➤ Input functions

- ❑ Used to control and process the data flow from various interactive devices such as a mouse, a tablet, and a joystick



Introduction to OpenGL



- Function names in the OpenGL basic library are prefixed with **gl**, and each component word within a function name has its **first letter capitalized**
 - ❑ **glBegin, glClear, glCopyPixels, glPolygonMode**
- Component words within a **constant** name are written in **capital letters**, and the **underscore (_)** is used as a separator between all component words in the name.
 - ❑ **GL_2D, GL_RGB, GL_CCW, GL_POLYGON, GL_AMBIENT_AND_DIFFUSE**



➤ OpenGL data-types

- ❑ GLbyte, GLshort, GLint, GLfloat, GLdouble, GLboolean

➤ Related Libraries

- ❑ OpenGL basic (core) library

- ❑ Associated Libraries

- OpenGL Utility (GLU)

- Contains routines for

- setting up viewing and projection matrices,
- describing complex objects with line and polygon approximations,
- displaying quadrics
- B-splines using linear approximations
- Processing the surface-rendering operations
- other complex tasks.



- Every OpenGL implementation includes the GLU library, and all GLU function names start with the prefix *glu*.
- Basic OpenGL library contains only device independent graphics functions
- Window-management operations depend on the computer used for graphics



OpenGL Library for window management operations



- **OpenGL Extension to the X Window System (GLX)** provides a set of routines for window-management operations that are prefixed with the letters **glX**.
- **Apple** systems can use the **Apple GL (AGL)** interface for window-management operations.
 - ❑ Function names for this library are prefixed with **agl**
- For **Microsoft** Windows systems, the **WGL** routines provide a **Windows-to-OpenGL** interface.
 - ❑ These routines are prefixed with the letters **wgl**
- The Presentation Manager to **OpenGL (PGL)** is an interface for the IBM OS/2, which uses the prefix **pgl** for the library routines.



➤ OpenGL Utility Toolkit (GLUT)

- ❑ Provides a library of functions for interacting with any screen-windowing system.
- ❑ The GLUT library functions are prefixed with **glut**
- ❑ contains methods for describing and rendering quadric curves and surfaces.
- ❑ GLUT is an interface to other device-specific window systems, so the programs will be device-independent





➤ Header Files

```
#include <windows.h>
```

```
#include <GL/gl.h>
```

```
#include <GL/glu.h>
```

OR

```
#include <GL/glut.h>
```

Display-Window Management Using GLUT



- GLUT initialization

`glutInit (&argc, argv);`

- Create window of given size and at fixed position

`glutInitWindowPosition (50, 100);`

`glutInitWindowSize (400, 300);`

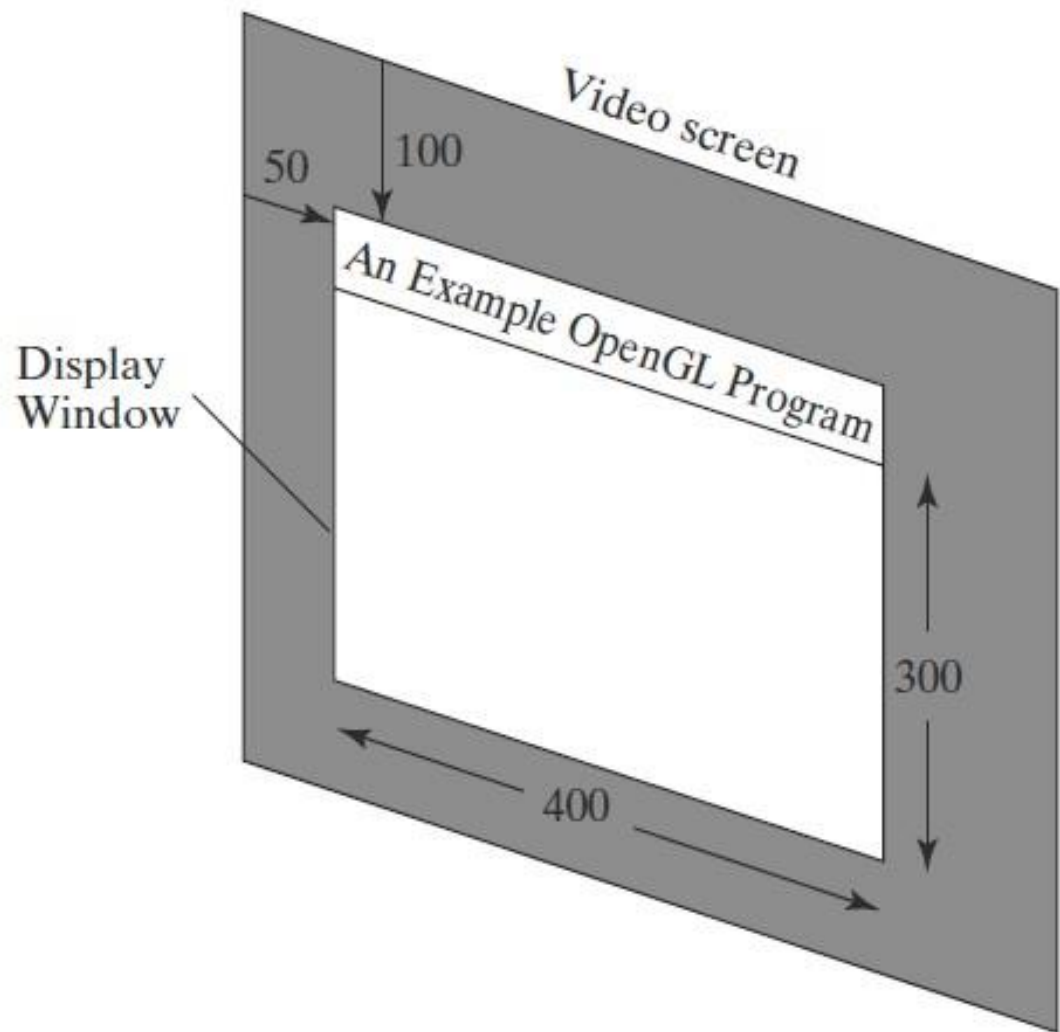
`glutCreateWindow ("An Example OpenGL Program");`

- Specify the contents of display window

`glutDisplayFunc (lineSegment);`

- *lineSegment* is a user defined procedure to display the line segments





A 400 by 300 display window at position (50, 100) relative to the top-left corner of the video display.



- Activate the windows and its contents

`glutMainLoop ();`

- It displays the initial graphics and puts the program into an infinite loop that checks for input from devices such as a mouse or keyboard

- buffering and a choice of color modes

`glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);`

- specifies that a single refresh buffer is to be used for the display window and the color mode is Red, Green, Blue (RGB)



➤ Setting background color

`glClearColor (1.0, 1.0, 1.0, 0.0);`

- The first three arguments are red, green, and blue component
 - The value of RGB components lie between 0-1
- The fourth parameter is the alpha value for the specified color.
 - An alpha value of 0 indicates a transparent object, and an alpha value of 1 indicates an opaque object
 - Alpha value can be used as a “blending” parameter.
 - When the OpenGL blending operation is activated, alpha values can be used to determine the resulting color for two overlapping objects.





`glClear (GL_COLOR_BUFFER_BIT);`

- Activates the assigned window color
- Sets the bit values in the color buffer (refresh buffer) to the values indicated in the `glClearColor` function

➤ Set the foreground color for objects

`glColor3f (0.0, 0.4, 0.2);`

➤ Defining the coordinate reference frame

`glMatrixMode (GL_PROJECTION);`

`gluOrtho2D (0.0, 200.0, 0.0, 150.0);`

- defines the coordinate reference frame within the display window
- Assigns (0.0, 0.0) at the lower-left corner of the display window and (200.0, 150.0) at the upper-right window corner



➤ Drawing line segment

```
glBegin (GL_LINES);
```

```
    glVertex2i (180, 15);
```

```
    glVertex2i (10, 145);
```

```
glEnd ( );
```



An Example OpenGL Program



```
#include <GL/glut.h> // (or others, depending on the system in use)
void init (void)
{
    glClearColor (1.0, 1.0, 1.0, 0.0); // Set display-window color to
    white.
    glMatrixMode (GL_PROJECTION); // Set projection parameters.
    gluOrtho2D (0.0, 200.0, 0.0, 150.0);
}
```





```
void lineSegment (void)
{
```

```
    glClear (GL_COLOR_BUFFER_BIT); // Clear display window.
```

```
    glColor3f (0.0, 0.8, 0.1); // Set line segment color to green.
```

```
    glBegin (GL_LINES);
```

```
        glVertex2i (180, 15); // Specify line-segment geometry.
```

```
        glVertex2i (10, 145);
```

```
    glEnd ( );
```

```
    glFlush ( ); // Process all OpenGL routines as quickly as possible.
```

```
}
```

```
void main (int argc, char** argv)
{
```



```
    glutInit (&argc, argv); // Initialize GLUT.
```

```
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB); // Set display mode.
```

```
    glutInitWindowPosition (50, 100); // Set top-left display-window position.
```

```
    glutInitWindowSize (400, 300); // Set display-window width and height.
```

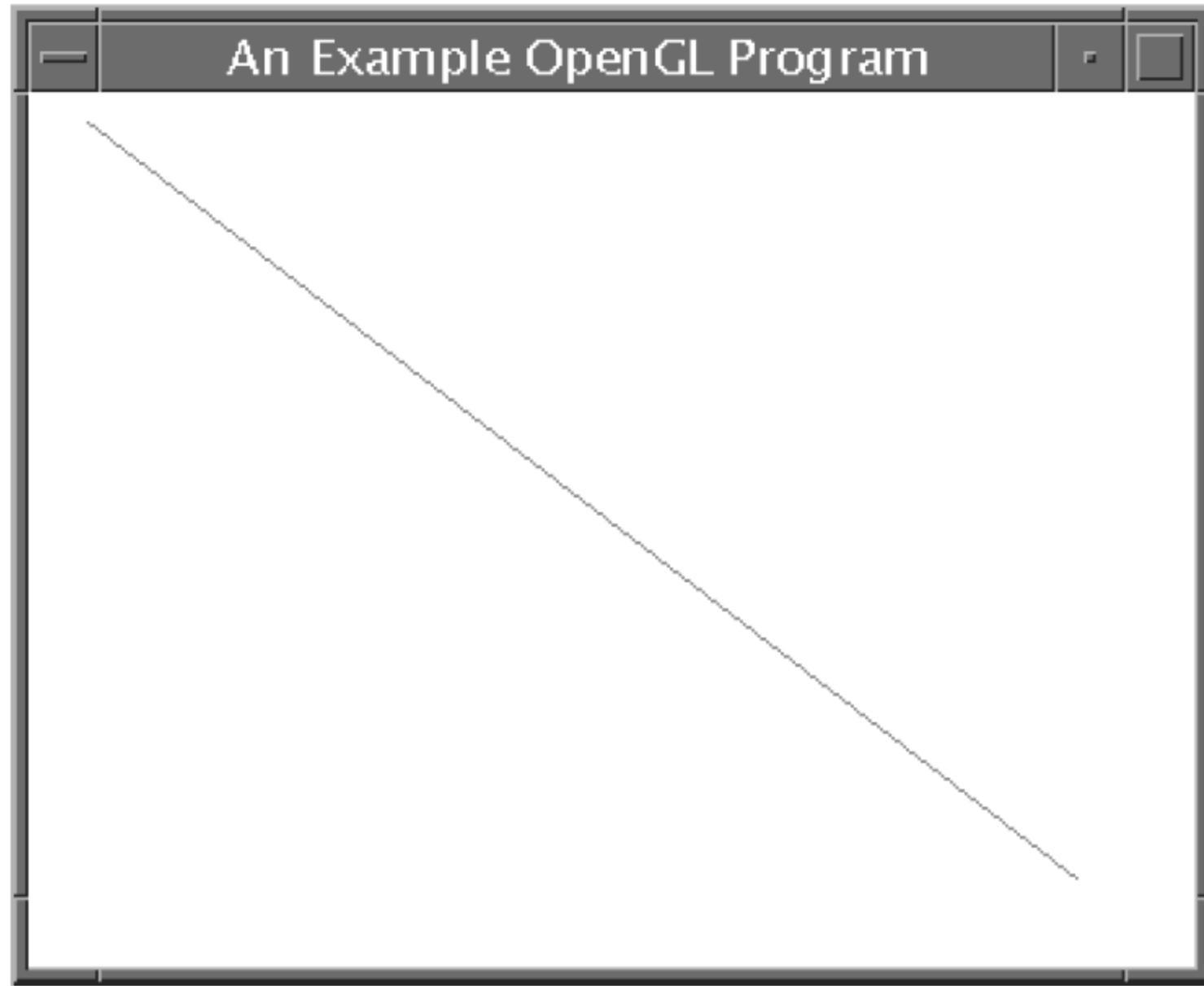
```
    glutCreateWindow ("An Example OpenGL Program"); // Create display window.
```

```
    init ( ); // Execute initialization procedure.
```

```
    glutDisplayFunc (lineSegment); // Send graphics to display window.
```

```
    glutMainLoop ( ); // Display everything and wait.
```

```
}
```



➤ Specifying A Two-Dimensional World-Coordinate Reference Frame in OpenGL

`gluOrtho2D`

- Arguments are the four values defining the x and y coordinate limits for the picture
- ❑ Since the `gluOrtho2D` function specifies an orthogonal projection, the coordinate values are placed in the OpenGL projection matrix.
- ❑ It is required to assign the identity matrix as the projection matrix before defining the world-coordinate range.
 - This ensures that the coordinate values were not accumulated with any values that might have previously set for the projection matrix.



- The coordinate frame for the screen display window is defined as follows:

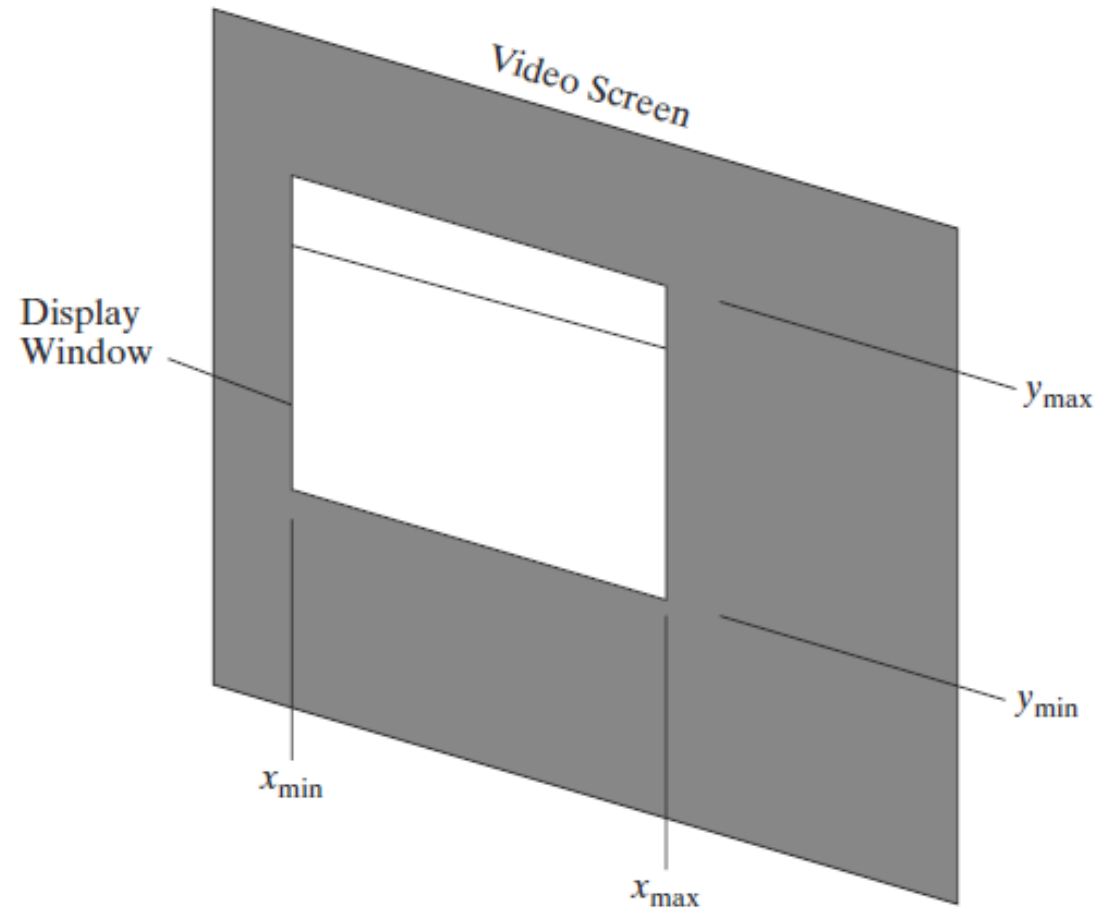
```
glMatrixMode (GL_PROJECTION);
```

```
glLoadIdentity ( );
```

```
gluOrtho2D (xmin, xmax, ymin, ymax);
```

- References the display window by coordinates (xmin, ymin) at the lower-left corner and by coordinates (xmax, ymax) at the upper-right corner



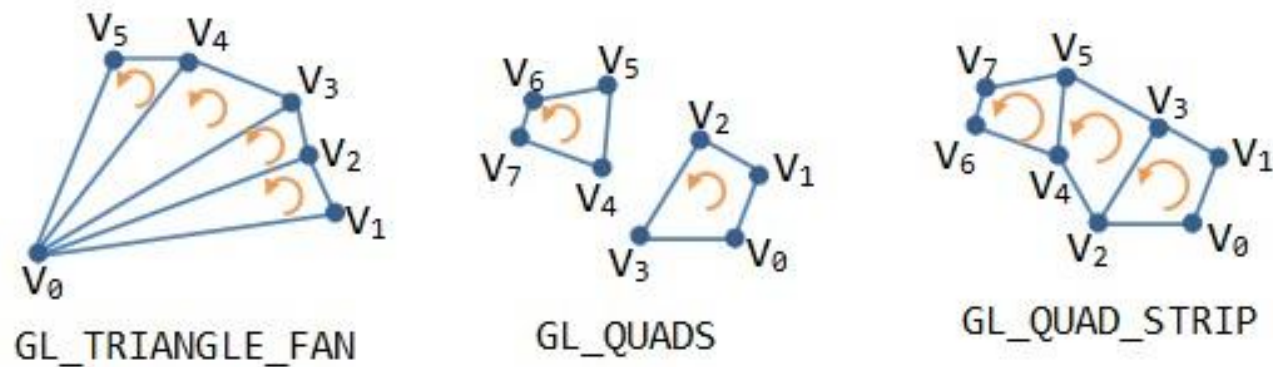
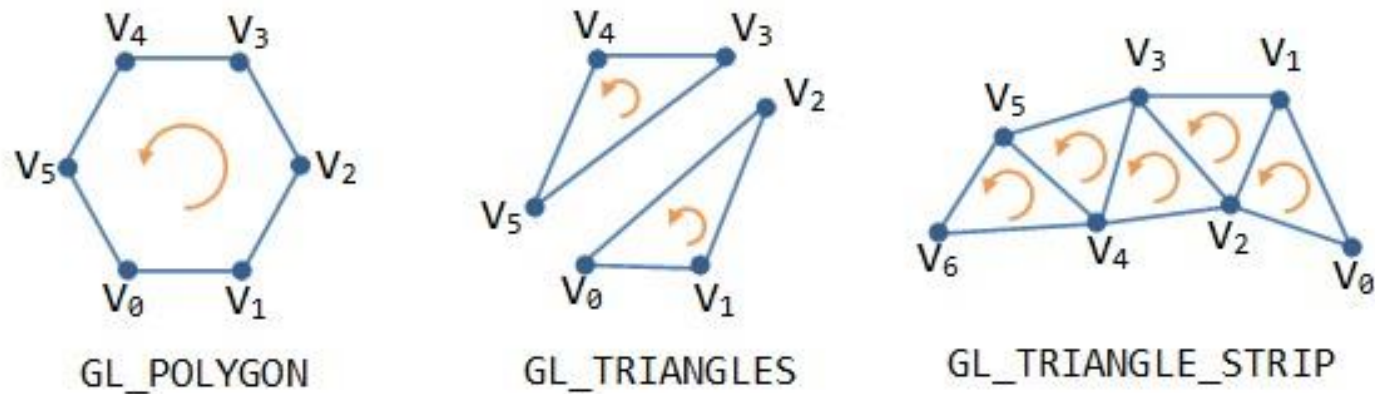
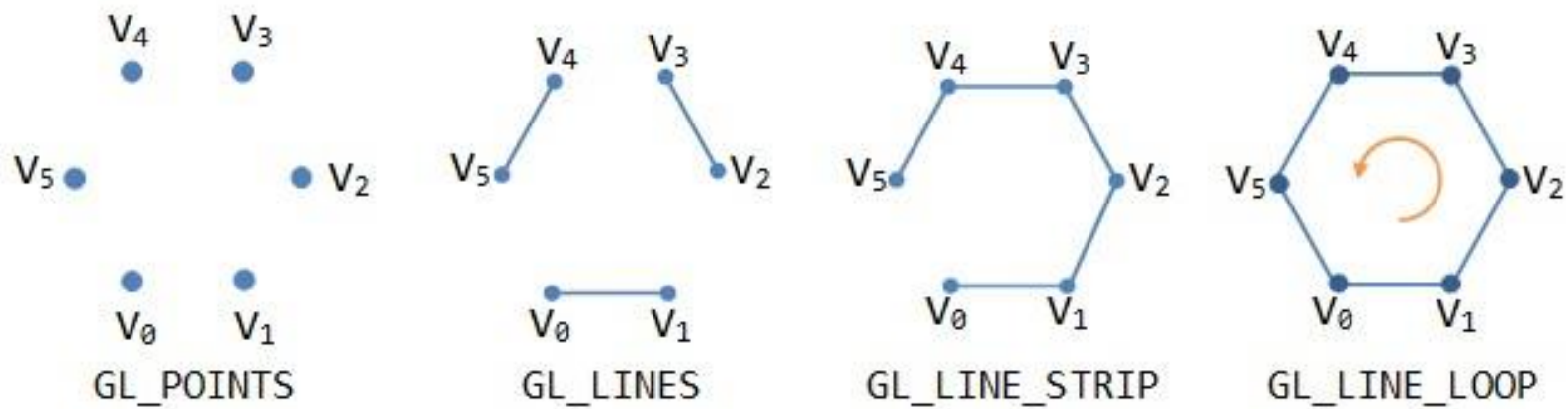


World-coordinate limits for a display window, as specified in the `glOrtho2D` function.

OpenGL primitive constants used with `glBegin` & `glEnd`



<code>GL_POINTS</code>	Individual points
<code>GL_LINES</code>	Pairs of vertices interpreted as individual line segments
<code>GL_LINE_STRIP</code>	Series of connected line segments
<code>GL_LINE_LOOP</code>	Same as above, with a segment added between last and first vertices
<code>GL_TRIANGLES</code>	Triples of vertices interpreted as triangles
<code>GL_TRIANGLE_STRIP</code>	Linked strips of triangles
<code>GL_TRIANGLE_FAN</code>	Linked fan of triangles
<code>GL_QUADS</code>	Quadruples of vertices interpreted as four sided polygons
<code>GL_QUAD_STRIP</code>	Linked strip of quadrilaterals
<code>GL_POLYGON</code>	Boundary of a simple, convex polygon



OpenGL Primitives



OpenGL Point Functions



`glVertex* ();`

- asterisk (*) indicates that suffix codes are required for this function.
- Suffix codes are used to identify
 - the spatial dimension (2,3 or 4)
 - the numerical data type to be used for the coordinate values,
 - *i* (integer), *s* (short), *f* (float), and *d* (double).
 - a possible vector form for the coordinate specification
 - Eg: `glVertex3f(x,y,z)` OR `glVertex3fv(p)`

dimensions

x,y,z are floats


p is a pointer to an array



- Calls to *glVertex* functions must be placed between a *glBegin* function and a *glEnd* function.
 - The argument of the *glBegin* function is used to identify the kind of output primitive that is to be displayed, and *glEnd* takes no arguments.
 - Eg: For point plotting,

```
glBegin (GL_POINTS);  
    glVertex* ( );  
glEnd ( );
```





```
glBegin (GL_POINTS);  
    glVertex2i (50, 100);  
    glVertex2i (75, 150);  
    glVertex2i (100,  
        200);  
glEnd ( );
```

OR

```
int point1 [ ] = {50, 100};  
int point2 [ ] = {75, 150};  
int point3 [ ] = {100, 200};  
glBegin (GL_POINTS);  
    glVertex2iv (point1);  
    glVertex2iv (point2);  
    glVertex2iv (point3);  
glEnd ( );
```



➤ Creating class for points

```
class Pt2D {  
    public:  
        GLfloat x, y;    };  
Pt2D ptPos;  
ptPos .x= 120.75;  
ptPos .y= 45.30;  
glBegin (GL_POINTS);  
    glVertex2f (ptPos.x, ptPos.y);  
glEnd ( );
```

➤ 3D points

```
glBegin (GL_POINTS);  
    glVertex3f (-78.05, 99.7, 4.6);  
    glVertex3f (2.91, -5.7, 8.33);  
glEnd ( );
```

OpenGL Line Functions

- GL LINE STRIP –polyline
- GL LINE LOOP – closed polyline

```
glBegin (GL_LINES);
```

```
glVertex2iv (p1);
```

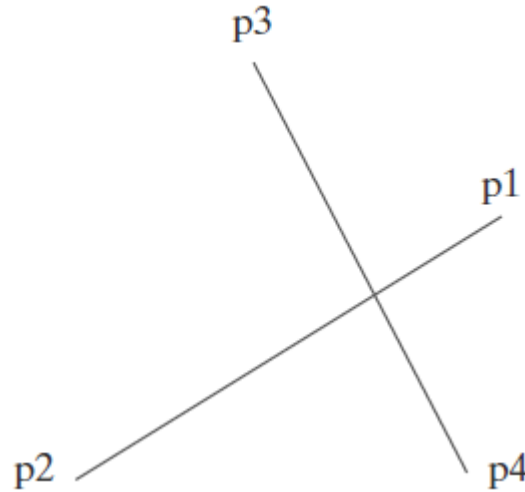
```
glVertex2iv (p2);
```

```
glVertex2iv (p3);
```

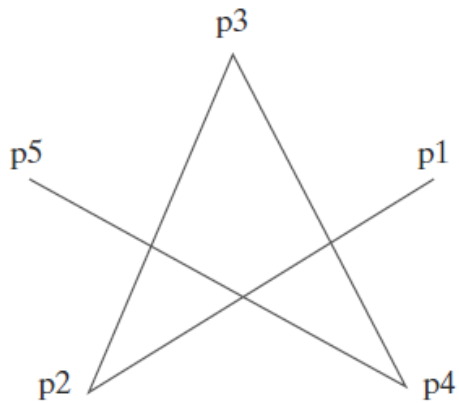
```
glVertex2iv (p4);
```

```
glVertex2iv (p5);
```

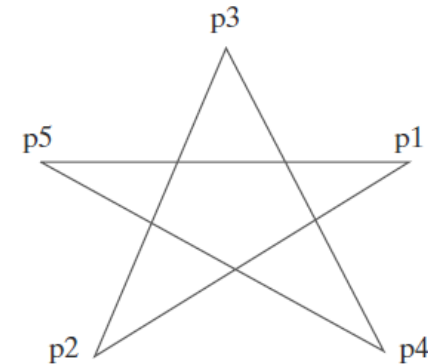
```
glEnd ( );
```



```
glBegin (GL_LINE_STRIP);  
    glVertex2iv (p1);  
    glVertex2iv (p2);  
    glVertex2iv (p3);  
    glVertex2iv (p4);  
    glVertex2iv (p5);  
glEnd ( );
```



```
glBegin (GL_LINE_LOOP);  
    glVertex2iv (p1);  
    glVertex2iv (p2);  
    glVertex2iv (p3);  
    glVertex2iv (p4);  
    glVertex2iv (p5);  
glEnd ( );
```



OpenGL Point-Attribute Functions

`glPointSize (size);`



- ❑ point is displayed as a square block of pixels.
- ❑ Parameter `size` is assigned a positive floating-point value, which is rounded to an integer

```
glColor3f (1.0, 0.0, 0.0);
```

```
glBegin (GL_POINTS);
```

```
    glVertex2i (50, 100);
```

```
    glPointSize (2.0);
```

```
    glColor3f (0.0, 1.0, 0.0);
```

```
    glVertex2i (75, 150);
```

```
    glPointSize (3.0);
```

```
    glColor3f (0.0, 0.0, 1.0);
```

```
    glVertex2i (100, 200);
```

```
glEnd ( );
```