# MODULE 1

## Basics of Computer Graphics

Computer graphics is an art of drawing pictures, lines, charts, etc. using computers with the help of programming. Computer graphics image is made up of number of pixels.

Pixel is the smallest addressable graphical unit represented on the computer screen.

## Applications of Computer Graphics

**a. Graphs and Charts**

**Usage**: Displaying simple data graphs (line graphs, bar charts, pie charts, etc.) on character printers and in publications.

**Purpose**: Summarizing data for research reports, managerial summaries, etc.

**b. Computer-Aided Design (CAD)**

**Usage**: Designing engineering and architectural systems.

**Applications**: Automobiles, aircraft, spacecraft, home appliances, circuits, and utility networks.

**Features**: Use of wire-frame shapes and animations for testing designs.

**c. Virtual-Reality Environments**

**Usage**: Training for heavy-equipment operators and analyzing cabin configurations.

**Features**: Interaction with objects, simulated walkthroughs of architectural designs, and object manipulation with special gloves.

**d. Data Visualizations**

**Types**: Scientific visualization (for scientific data) and business visualization (for commercial data).

**Techniques**: Effective visualization schemes depend on data characteristics, such as scalar values, vectors, or tensors.

**e. Education and Training**

**Usage**: Models of physical, financial, political, social, and economic systems as educational aids.

**Applications**: Simulators for training pilots, air traffic controllers, and other specialized training equipment.

**f. Computer Art**

**Techniques**: Electronic painting on graphics tablets, 3D modeling, texture mapping, and CAD software.

**Applications**: Commercial art, logos, page layouts, TV advertising, and morphing in commercials.

**g. Entertainment**

**Usage**: Television production, motion pictures, music videos.

**Techniques**: Combining live actors with CGI, fully computer-generated films, and animation techniques for special effects.

**h. Image Processing**

**Definition**: Modification or interpretation of existing pictures.

**Techniques**: Enhancing image quality, analyzing images, recognizing visual patterns, and medical imaging applications (CT, PET, CAT scans).

**i. Graphical User Interfaces (GUIs)**

**Components**: Window manager for displaying multiple rectangular screen areas (display windows).

**Interaction**: Using interactive devices like a mouse to activate display windows.

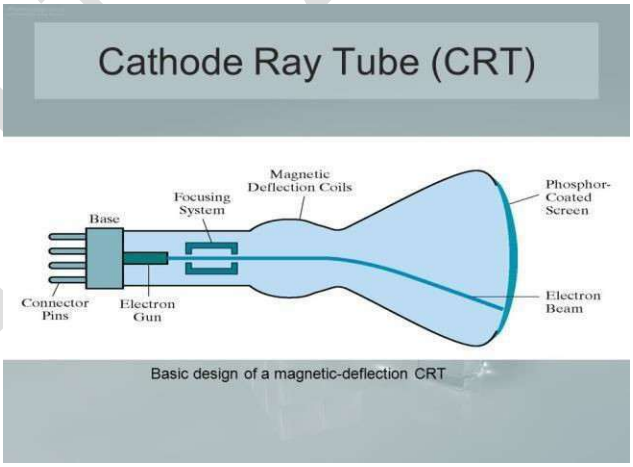
**Video Display Devices**

✓ The primary output device in a graphics system is a video monitor.

✓ Historically, the operation of most video monitors was based on the standard cathoderay tube (CRT) design, but several other technologies exist.

✓ In recent years, flat-panel displays have become significantly more popular due to their reduced power consumption and thinner designs.

**Refresh Cathode Ray Tubes**

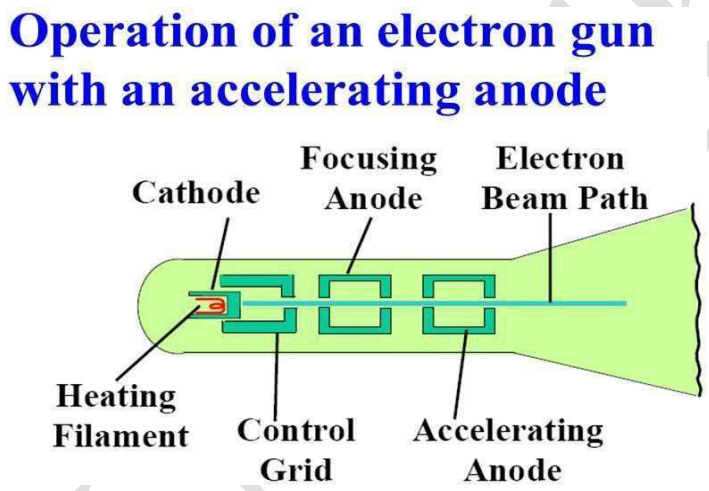



1. **Electron Beam Emission**: A beam of electrons is emitted by an electron gun.
2. **Focusing and Deflection Systems**: These systems direct the electron beam to specific positions on the screen.
3. **Phosphor-Coated Screen**: When the electron beam contacts the phosphor-coated screen, the phosphor emits light, creating a visible spot.

4. **Phosphor Fade**: The emitted light from the phosphor fades very rapidly.
5. **Maintaining the Screen Picture**: To keep the phosphors activated and maintain the picture:
    - **Charge Distribution**: Store picture information as a charge distribution within the CRT.
    - **Refresh CRT**: The most common method is to redraw the picture repeatedly by directing the electron beam back over the same screen points.
6. **Refresh Rate**: The frequency at which the picture is redrawn on the screen is known as the refresh rate.

This process ensures that the image on the screen remains visible and stable to the human eye by continually refreshing the display.

**Operation of an electron gun with an accelarating anode**




**Primary Components of an Electron Gun in a CRT**

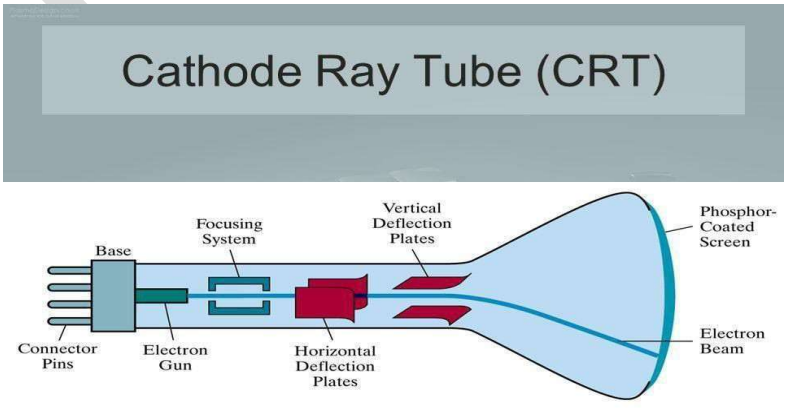
1. **Heated Metal Cathode and Control Grid**:
    - **Cathode Heating**: A current is directed through a coil of wire (filament) inside the cylindrical cathode structure, causing the cathode to heat up.
    - **Electron Emission**: The heat causes electrons to be "boiled off" the hot cathode surface.
2. **Electron Acceleration**:
    - **Acceleration Mechanism**: Free, negatively charged electrons are accelerated toward the phosphor coating by a high positive voltage inside the CRT envelope.
3. **Beam Intensity Control**:
    - **Control Grid Voltage**: The intensity of the electron beam is controlled by the voltage at the control grid.

- **Brightness Control**: The brightness of a display point is controlled by varying the voltage on the control grid, as the light emitted by the phosphor coating depends on the number of electrons striking the screen.

**Focusing and Deflection Systems**

1. **Focusing System**:
   - **Convergence**: The electron beam is forced to converge to a small cross-section as it strikes the phosphor coating.
   - **Focusing Methods**: This is achieved using either electric or magnetic fields.
   - **Electrostatic Focusing**: The electron beam is passed through a positively charged metal cylinder, keeping electrons along the center line in equilibrium.
2. **Deflection System**:
   - **Deflection Control**: The electron beam can be deflected using electric or magnetic fields.
   - **Magnetic Deflection Coils**:
     - **Coil Pairs**: CRTs commonly have two pairs of magnetic deflection coils.
     - **Placement**: One pair is mounted on the top and bottom of the CRT neck, and the other pair is mounted on opposite sides of the neck.
     - **Deflection Force**: The magnetic field produced by each pair of coils results in a traverse deflection force perpendicular to both the magnetic field direction and the electron beam's travel direction.
   - **Deflection Types**: Horizontal and vertical deflections are accomplished with these pairs of coils.

**<u>Electrostatic deflection of the electron beam in a CRT</u>**




Electrostatic deflection of the electron beam in a CRT

**Electrostatic Deflection in CRT**

1. **Parallel Plates**:
    - **Horizontal Plates**: Control vertical deflection.
    - **Vertical Plates**: Control horizontal deflection.

**Light Production on the Screen**

1. **Energy Transfer**:
    - **Collision**: Electrons in the beam collide with the phosphor coating.
    - **Energy Conversion**:
        - **Heat Energy**: Part of the beam energy is converted to heat through friction.
        - **Quantum Energy Levels**: The remaining energy excites electrons in the phosphor atoms, raising them to higher quantum-energy levels.
    - **Light Emission**:
        - **Photon Emission**: Excited phosphor electrons drop back to their stable ground state, releasing extra energy as photons (light energy).
        - **Visible Effect**: The combined light emissions of all electrons create a glowing spot on the screen, which quickly fades as electrons return to their ground state.
    - **Frequency of Light**: The frequency of emitted light is proportional to the energy difference between the excited state and the ground state.

**Persistence and Refresh Rates**

1. **Phosphor Persistence**:
    - **Low Persistence**: Requires higher refresh rates to maintain a flicker-free image.
    - **High Refresh Rates**: Necessary to continuously redraw the image to prevent flickering.

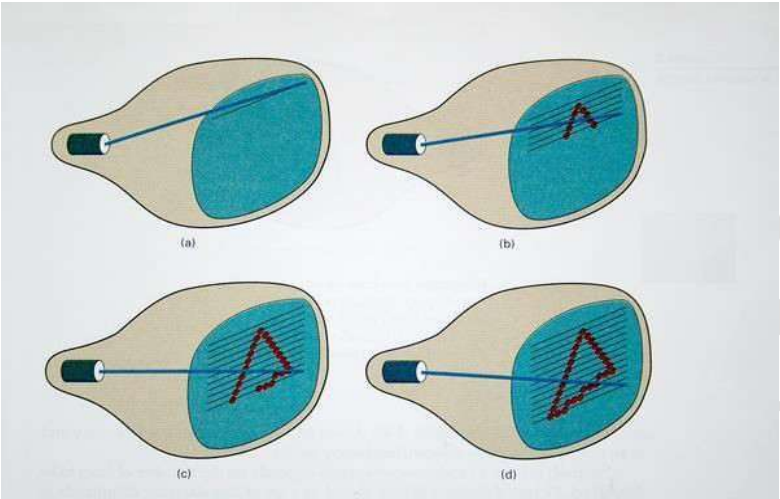
**Resolution**

1. **Maximum Points Displayed**:
    - **Resolution Definition**: The maximum number of points that can be displayed without overlap.
    - **Factors Affecting Resolution**:
        - **Phosphor Type**: Different phosphors affect the display quality.
        - **Intensity**: Brightness levels impact resolution.

- **Focusing and Deflection Systems**: Precision in these systems is crucial for high resolution.

- **High-Resolution Systems**: Often referred to as high-definition systems, providing clearer and more detailed images.

**Raster-Scan Displays**




**Operation and Characteristics**

1. **Electron Beam Scanning**:
   - **Row-by-Row**: The electron beam is swept across the screen one row at a time from top to bottom.
   - **Beam Intensity**: As the beam moves across each row, its intensity is turned on and off to create a pattern of illuminated spots.
2. **Refreshing**:
   - **Scanning Process**: Known as refreshing, this process redraws the screen to maintain the image.
   - **Frame Rate**: The refreshing rate, or frame rate, typically ranges from 60 to 80 frames per second (Hz).
3. **Frame Buffer**:
   - **Storage**: Picture definition is stored in a memory area called the frame buffer.
   - **Pixels**: The frame buffer stores intensity values for all screen points, known as pixels (picture elements).
4. **Aspect Ratio**:
   - **Definition**: Aspect ratio is defined as the number of pixel columns divided by the number of scan lines that the system can display.

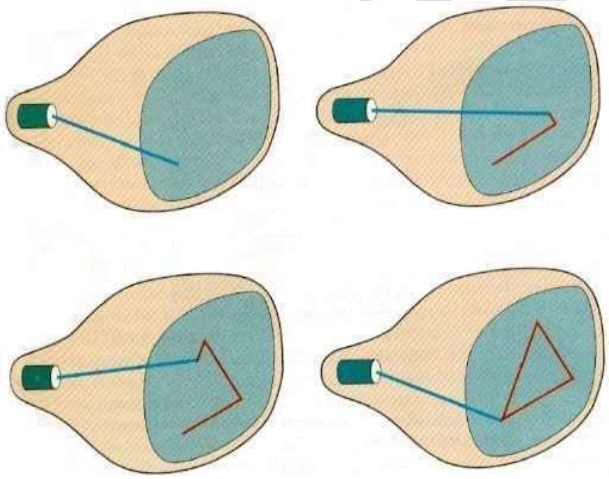
**Black and White Systems**

- **Bitmap**:
    - **Frame Buffer**: On black and white systems, the frame buffer is called a bitmap.
    - **Data Representation**: Each entry in the bitmap is a 1-bit value, determining whether the pixel is on (1) or off (0).

**Color Systems**

- **Pixmap**:
    - **Frame Buffer**: On color systems, the frame buffer is called a pixmap (though some graphics libraries may still refer to it as a bitmap).
    - **Data Representation**: Each entry in the pixmap uses multiple bits to represent the pixel color.
    - **True Color Display**: Typically uses 24 bits per pixel (8 bits for each of the red, green, and blue channels), allowing 256 intensity levels for each color channel.

**Random-Scan Displays**



**Operation and Characteristics**

1. **Electron Beam Control**:
    - **Drawing**: The electron beam is directed only to the parts of the screen where a picture is to be drawn.
    - **Picture Definition**: Stored as a set of line-drawing commands in a refresh display file or display list.
2. **Vector Display**:
    - **Direct Drawing**: Also known as vector displays, these systems draw images directly by moving the beam to the endpoints of each line to be drawn.

3. **Line Drawing**:
    - **Resolution**: Higher resolution for line drawings compared to raster-scan systems.
    - **Refresh Rate**: Typically lower than raster-scan systems due to the more complex process of drawing images directly.

**Comparison**

- **Raster-Scan**:
    - **Pixel-Based**: Images are composed of pixels, with the frame buffer storing intensity values for each pixel.
    - **Frame Rate**: Usually higher (60-80 Hz), suitable for complex images and motion graphics.
    - **Bitmap/Pixmap**: Frame buffer can be a bitmap for black and white systems or a pixmap for color systems.
- **Random-Scan**:
    - **Line-Based**: Images are drawn using line-drawing commands.
    - **Resolution**: Higher resolution for line drawings, but generally lower refresh rates.
    - **Display List**: Stores line-drawing commands instead of pixel intensity values.

| Base of Difference | Random Scan | Raster Scan |
|---|---|---|
| Resolution | The resolution of random scan is higher than raster scan. | While the resolution of raster scan is lesser or lower than random scan. |
| Cost | It is costlier than raster scan because it requires more sophisticated hardware to control the electron beam directly | While the cost of raster scan is lesser than random scan. |
| Modification | In random scan, any alteration is easy in comparison of raster scan. | While in raster scan, any alteration is not so easy . |
| Interlacing | In random scan, interlacing is not used. | While in raster scan, interlacing is used. |
| Line Drawings | In random scan, mathematical function is used for image or picture rendering. It is suitable for applications requiring polygon drawings. | While in which, for image or picture rendering, raster scan uses pixels. It is suitable for creating realistic scenes. |
| Motion of Electron Beam | Electron Beam is directed to only that part of screen where picture is required to be drawn, one line at a time. | Electron Beam is directed from top to bottom and one row at a time on screen. It is directed to whole screen. |
| Picture Definition | It stores picture definition as a set of line commands in the Refresh buffer. | It stores picture definition as a set of intensity values of the pixels in the frame buffer. |
| Refresh Rate | Refresh rate depends on the number of lines to be displayed i.e. 30 to 60 times per second. | Refresh rate is 60 to 80 frames per second and is independent of picture complexity. |
| Solid Pattern | In random scan, Solid Pattern is tough to fill. | In raster scan, Solid Pattern is easy to fill. |
| Example | Pen Plotter | TV Sets |

## Color CRT Monitors

A CRT monitor displays color pictures using a combination of phosphors that emit different-colored light. It can produce a range of colors by combining the light emitted by different phosphors. There are two basic techniques for color display: Beam-Penetration Technique and Shadow-Mask Technique.

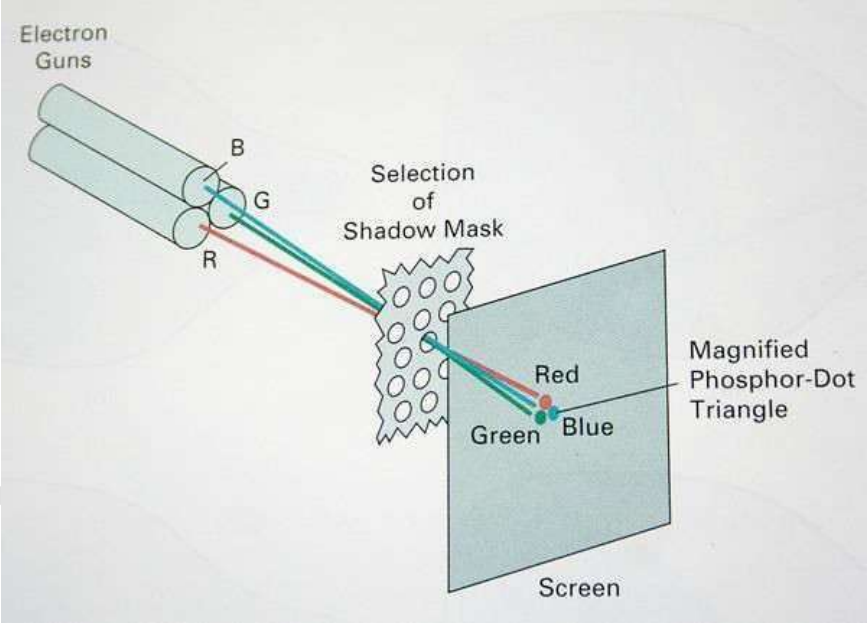
### 1. Beam-Penetration Technique

- **Usage**: This technique is used with random scan monitors.
- **Phosphor Layers**: The inside of the CRT is coated with two phosphor layers, usually red and green.
    - **Layer Arrangement**: The outer layer is red, and the inner layer is green.
- **Color Determination**: The color depends on how far the electron beam penetrates into the phosphor layer.
    - **Electron Penetration**:
        - **Fast Electron Beam**: Penetrates more and excites the inner green layer.
        - **Slow Electron Beam**: Excites the outer red layer.
        - **Intermediate Beam Speed**: Produces a combination of red and green lights, resulting in additional colors such as orange and yellow.
    - **Beam Acceleration Voltage**: Controls the speed of the electrons and hence the color of the pixel.
- **Disadvantages**:
    - **Cost**: It is a low-cost technique to produce color in random scan monitors.
    - **Color Limitation**: Can display only four colors.
    - **Picture Quality**: Not as good compared to other techniques.

### 2. Shadow-Mask Technique

- **Color Range**: Produces a wider range of colors compared to the beam-penetration technique.
- **Usage**: Generally used in raster scan displays, including color TVs.
- **CRT Structure**:
    - **Phosphor Dots**: Each pixel position has three phosphor color dots.
        - **Colors**: One dot for red, one for green, and one for blue light. This is known as a dot triangle.
    - **Electron Guns**: Three electron guns are present, one for each color dot.
    - **Shadow Mask Grid**: Positioned just behind the phosphor-coated screen.

        - **Holes**: The shadow mask grid consists of a series of holes aligned with the phosphor dot pattern.
    - **Electron Beams**: Three electron beams are deflected and focused as a group onto the shadow mask.
        - **Beam Activation**: When beams pass through a hole, they excite a dot triangle.
        - **Dot Triangle Activation**: Each electron beam activates only its corresponding color dot when it passes through the shadow mask.
    - **Screen Display**: A dot triangle, when activated, appears as a small dot on the screen, which is the color combination of the three small dots in the dot triangle.
- **Color Intensity**:
    - **Beam Intensity Adjustment**: By changing the intensity of the three electron beams, different colors can be obtained in the shadow-mask CRT.



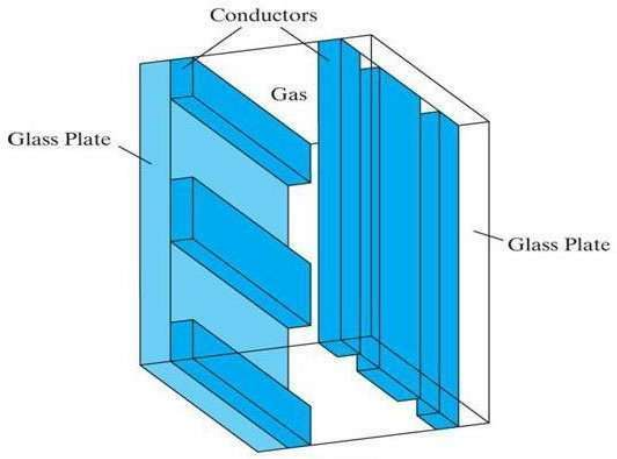

**Flat Panel Displays**

Flat panel displays refer to a class of video devices that have reduced volume, weight, and power requirements compared to CRTs. These displays are thinner, making them suitable for mounting on walls or wearing on wrists. They can also be used as pocket notepads with writeable surfaces. Flat panel displays can be categorized into two types:

**1. Emissive Displays**

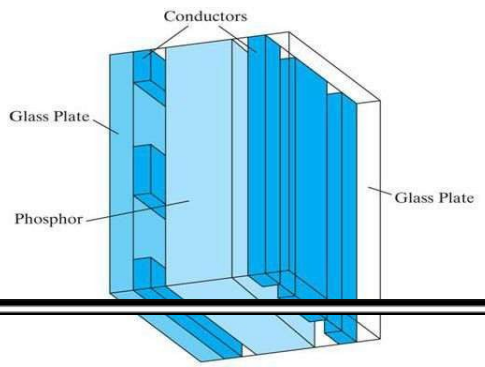
Emissive displays (emitters) are devices that convert electrical energy into light. Examples include:

- **Plasma Panels (Gas Discharge Displays)**:
    - Constructed by filling the region between two glass plates with a mixture of gases, usually including neon.
    - Vertical conducting ribbons are placed on one glass panel, and horizontal ribbons on the other.
    - Applying a firing voltage to a pair of horizontal and vertical conductors causes the gas at their intersection to break down into a glowing plasma of electrons and ions.
    - Picture definition is stored in a refresh buffer, and firing voltages refresh pixel positions 60 times per second.
    - Alternating current methods provide faster application of firing voltages for brighter displays.
    - Disadvantage: Plasma panels are strictly monochromatic (e.g., black and white).




**Thin Film Electroluminescent Displays**:

- Similar to plasma panels, but the region between the glass plates is filled with phosphors doped with magnesium instead of gas.
- When sufficient voltage is applied, the phosphors conduct at the intersection of the two electrodes, absorbing electrical energy and releasing it as a spot of light.
- Requires more power than plasma panels.
- Good color and grayscale are difficult to achieve.



**Light Emitting Diode (LED) Displays**

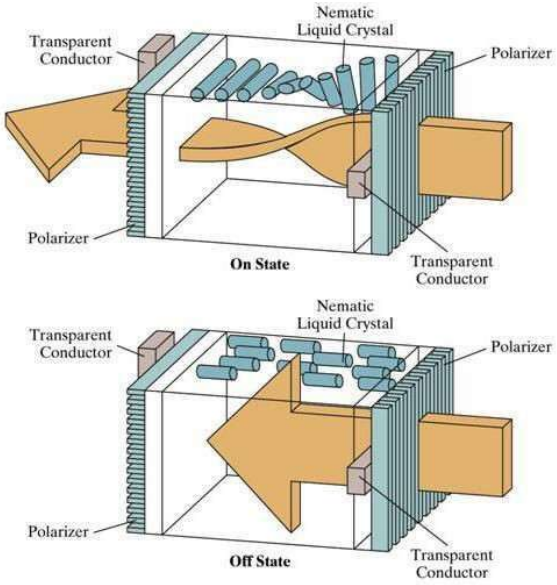
- **Structure and Functionality**:
    - A matrix of multi-color light-emitting diodes (LEDs) forms the pixel positions in the display.
    - Picture definition is stored in a refresh buffer, similar to the scan line refreshing used in CRTs.
    - Information is read from the refresh buffer and converted into voltage levels applied to the diodes, creating the light pattern on the display.

**Liquid Crystal Display (LCD)**

- **Working Principle**:
    - LCDs are non-emissive devices that produce images by passing polarized light through a liquid crystal material that can be aligned to either block or transmit light.
    - The liquid crystal's molecules have a crystalline arrangement but flow like a liquid, allowing them to be manipulated to control light passage.
- **Construction**:
    - **Two Glass Plates**: Each plate has a light polarizer at right angles to each other, sandwiching the liquid crystal material.
    - **Conductors**: Rows of horizontal transparent conductors are embedded in one glass plate, while columns of vertical conductors are embedded in the other plate.
    - **Pixel Definition**: The intersection of these conductors defines each pixel position.
- **States of Operation**:
    - **ON State**: In this state, polarized light passing through the liquid crystal material is twisted, allowing it to pass through the opposite polarizer, thus transmitting light.
    - **OFF State**: In this state, the light reflects back towards the source, blocking the light passage.

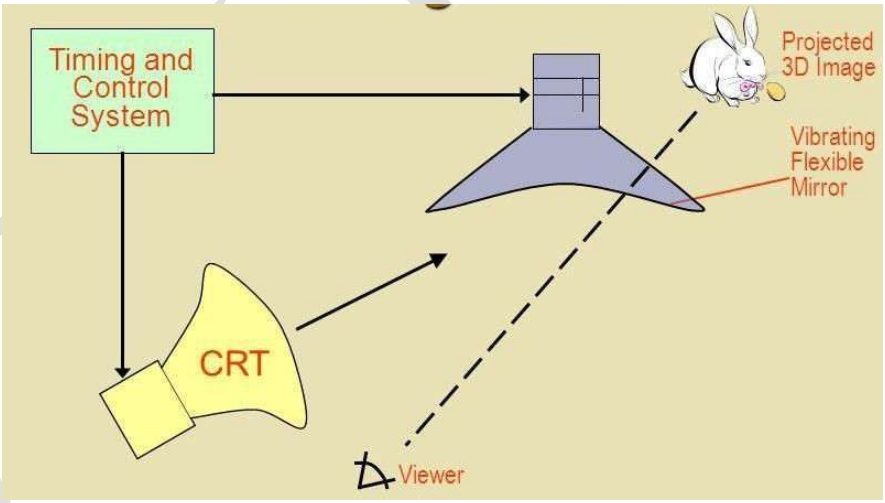
**hree-Dimensional Viewing Devices**

Three-dimensional graphics monitors have been devised using a technique that reflects a CRT image from a vibrating, flexible mirror. As the varifocal mirror vibrates, it changes its focal length. These vibrations are synchronized with the display of an object on a CRT so that each point on the object is reflected into a spatial position corresponding to its distance from a specified viewing location. This allows users to walk around an object or scene and view it from different sides.



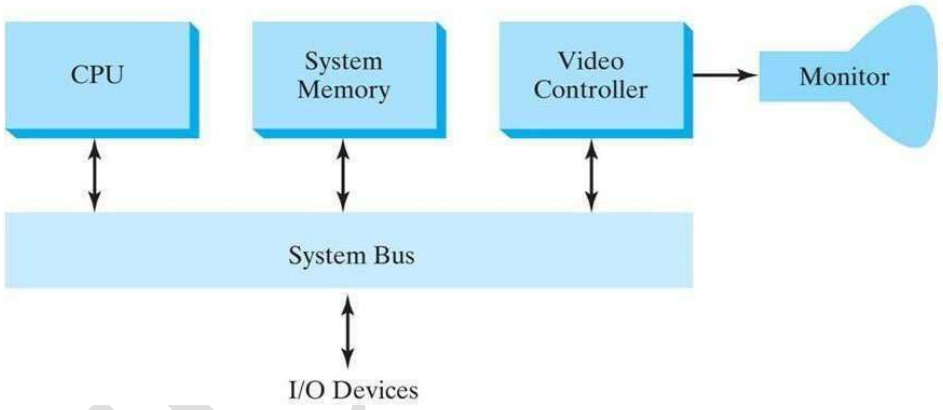

**Three-Dimensional Viewing Devices**

Three-dimensional graphics monitors have been devised using a technique that reflects a CRT image from a vibrating, flexible mirror. As the varifocal mirror vibrates, it changes its focal length. These vibrations are synchronized with the display of an object on a CRT so that each point on the object is reflected into a spatial position corresponding to its distance from a specified viewing location. This allows users to walk around an object or scene and view it from different sides.



**Raster-Scan Systems**

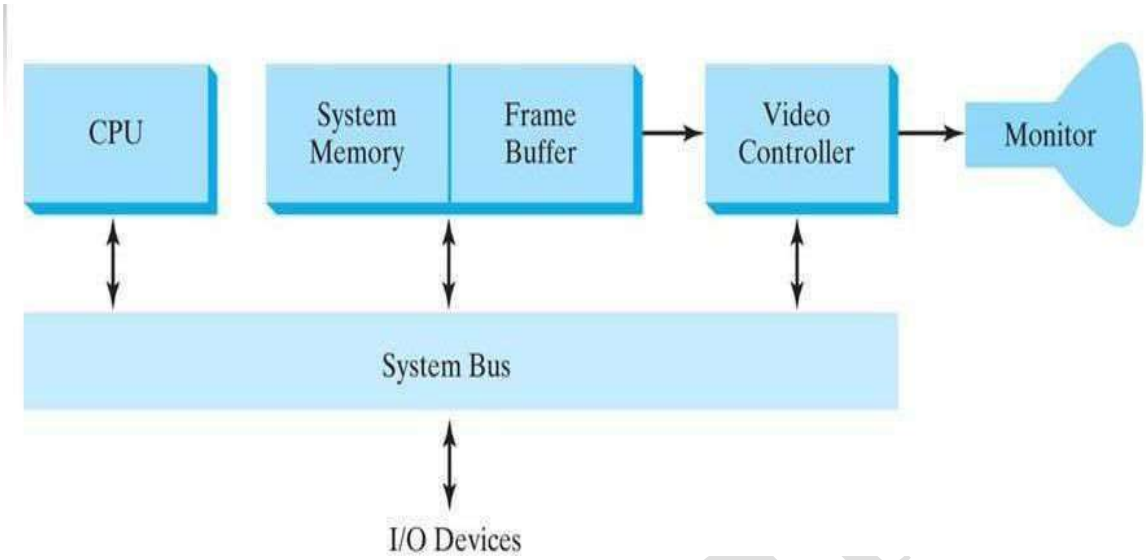
- **Interactive Raster-Graphics Systems**:
    - These systems typically use several processing units to handle various tasks.
    - **Central Processing Unit (CPU)**: The main processor that handles general operations.
    - **Video Controller (Display Controller)**: A special-purpose processor dedicated to controlling the display device's operations.
- **System Organization**:
    - The frame buffer, which stores the image data, can be located anywhere in the system memory.
    - The video controller accesses the frame buffer to refresh the screen.
    - Other processors, such as coprocessors and accelerators, are often used to implement various graphics operations, enhancing performance.




### 1.4.1 Video Controller

- **Role and Function**:
    - The video controller is responsible for accessing the frame buffer and refreshing the screen.
    - It operates in conjunction with the CPU and other specialized processors to manage the display device effectively.
- **Typical Organization**:
    - The frame buffer stores the pixel data for the image to be displayed.
    - The video controller continuously reads data from the frame buffer and sends it to the display device to create the visual output.

- Additional processors, like graphics coprocessors and accelerators, assist in complex graphics computations and rendering tasks.




**Cartesian Reference Frame**

- **Frame-Buffer Locations and Screen Positions**:
  - Locations in the frame buffer and their corresponding screen positions are referenced using Cartesian coordinates.
  - In application programs, commands within a graphics software package are used to set coordinate positions for displayed objects relative to the origin of the coordinate system.
- **Coordinate Origin**:
  - The coordinate origin is typically referenced at the lower-left corner of the screen display area by software commands.
  - However, the origin can be set at any convenient location for a specific application.

**Basic Video Controller Refresh Operations**

**Overview:**

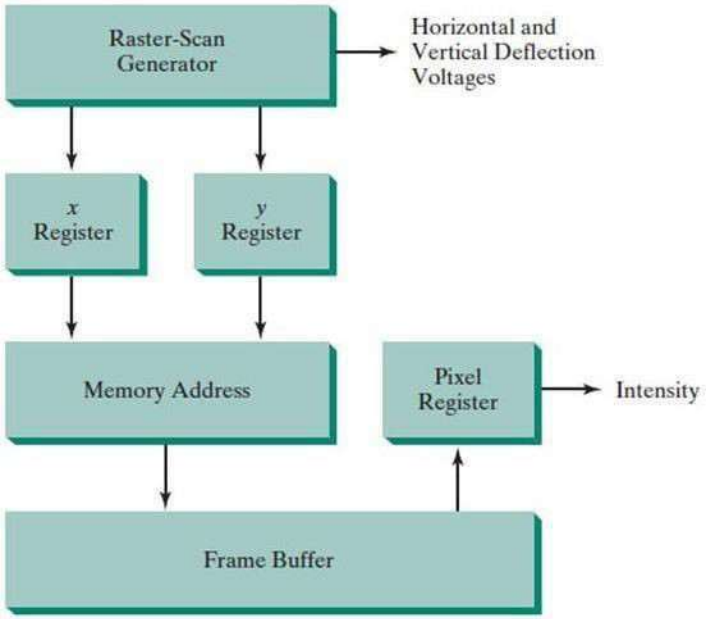
The basic refresh operations of a video controller involve repeatedly updating the screen display by cycling through each pixel in a structured manner. This ensures a stable and continuous display, critical for interactive graphics systems.

**Initial Setup:**

1. **Registers**:
   - **x Register**: Initially set to 0, representing the starting column of the screen.
   - **y Register**: Set to the value corresponding to the top scan line, representing the starting row of the screen.
2. **Frame Buffer**:

- The frame buffer stores pixel values that define the screen display. The video controller accesses this buffer to refresh the display.

**Refresh Cycle:**




1. **Pixel Intensity Setting**:
   - The video controller retrieves the pixel value from the frame buffer at the coordinates specified by the x and y registers.
   - This pixel value is used to set the intensity of the CRT beam, determining the color and brightness of the pixel displayed on the screen.
2. **Incrementing the x Register**:
   - The x register is incremented by 1, moving to the next pixel on the same scan line (row).
   - This process is repeated for each pixel along the current scan line.
3. **End of Scan Line**:
   - Once the last pixel on the current scan line is processed, the x register is reset to 0.
   - The y register is incremented to move to the next scan line down from the top of the screen.
4. **Next Scan Line**:
   - The video controller repeats the pixel intensity setting and x register incrementing process for each pixel along this new scan line.

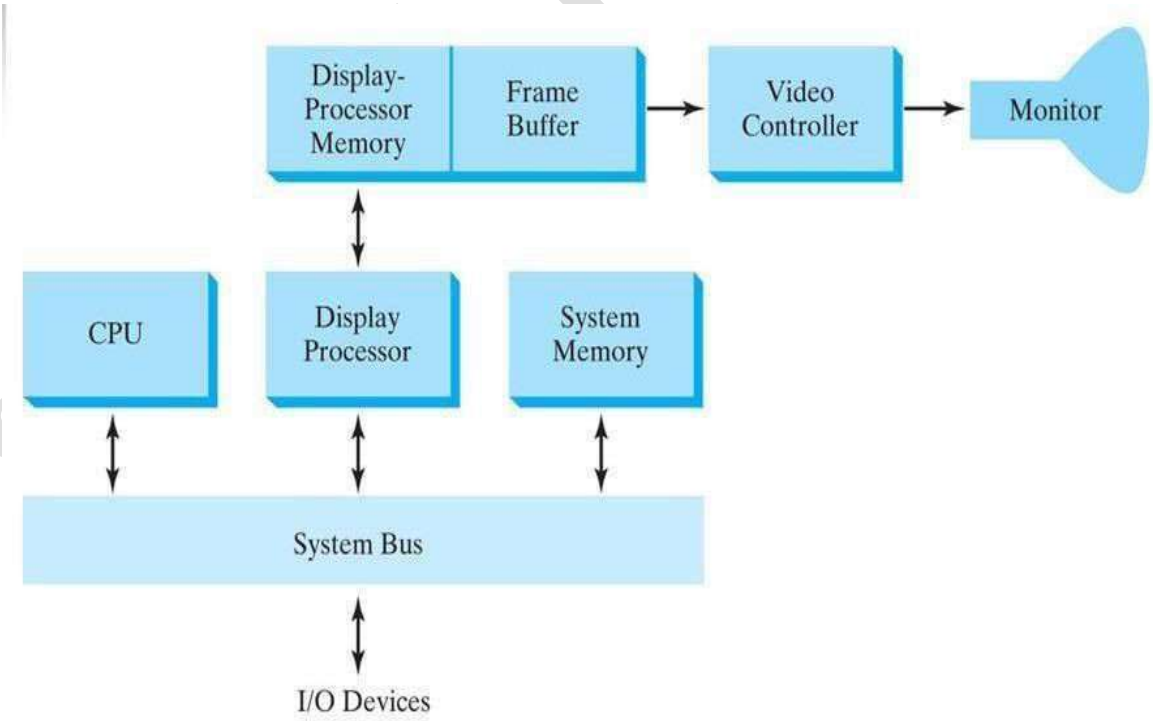
5. **Complete Screen Refresh**:
    - This process continues until all scan lines, from the top to the bottom of the screen, have been processed.
    - After the bottom scan line is completed, the video controller resets both x and y registers to their initial values and starts the refresh process over again.

**Speed Optimization:**

- **Refresh Rate**:
    - The screen needs to be refreshed at a rate of at least 60 frames per second to maintain a stable and flicker-free display.
- **Pixel Processing Speed**:
    - To achieve this refresh rate, video controllers may retrieve and process multiple pixel values in each cycle, instead of one pixel at a time.
    - This batch processing helps in speeding up the refresh process, ensuring that all pixels are updated within the required time frame.

**b) Raster-Scan Display Processor**

✓ Figure shows one way to organize the components of a raster system that contains a separate display processor, sometimes referred to as a graphics controller or a display coprocessor.




✓ The purpose of the display processor is to free the CPU from the graphics chores.

✓ In addition to the system memory, a separate display-processor memory area can be

provided.

***Scan conversion:***

- ✓ A major task of the display processor is digitizing a picture definition given in an application program into a set of pixel values for storage in the frame buffer.
- ✓ This digitization process is called scan conversion.

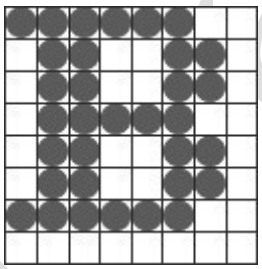
**Example 1: displaying a line**

Graphics commands specifying straight lines and other geometric objects are scanconverted into a set of discrete points, corresponding to screen pixel positions.

Scan converting a straight-line segment

**Example 2: displaying a character**

- Characters can be defined with rectangular pixel grids
- The array size for character grids can vary from about 5 by 7 to 9 by 12 or more for higher-quality displays.
- A character grid is displayed by superimposing the rectangular grid pattern into the frame buffer at a specified coordinate position.



**Using outline:**

□ For characters that are defined as outlines, the shapes are scan-converted into the frame buffer by locating the pixel positions closest to the outline.

**Additional operations of Display processors:**

- Display processors are also designed to perform a number of additional operations.
- These functions include generating various line styles (dashed, dotted, or solid), displaying color areas, and applying transformations to the objects in a scene.
- Display processors are typically designed to interface with interactive input devices, such as a mouse.

**Methods to reduce memory requirements in display processor:**

In an effort to reduce memory requirements in raster systems, methods have been devised for organizing the frame buffer as a linked list and encoding the color information.

One organization scheme is to store each scan line as a set of number pairs

Encoding methods can be useful in the digital storage and transmission of picture information

## 1.1 Input Devices

- Graphics workstations make use of various devices for data input.Most systems have keyboards and mouses,while some other systems have trackball,spaceball,joystick,button boxes,touch panels,image scanners and voice systems.

***Keyboard:***

- Keyboard on graphics system is used for entering text strings,issuing certain commands and selecting menu options.
- Keyboards can also be provided with features for entry of screen coordinates,menu selections or graphics functions.
- General purpose keyboard uses function keys and cursor-control keys.
- Function keys allow user to select frequently accessed operations with a single keystroke.Cursor-control keys are used for selecting a displayed object or a location by positioning the screen cursor.

***Button Boxes and Dials:***

- Buttons are often used to input predefined functions .Dials are common devices for entering scalar values.
- Numerical values within some defined range are selected for input with dial rotations.

***Mouse Devices:***

- Mouse is a hand-held device,usually moved around on a flat surface to position the screen cursor.wheeler or roolers on the bottom of the mouse used to record the amount and direction of movement.
- Some of the mouses uses optical sensors,which detects movement across the horizontal and vertical grid lines.
- Since a mouse can be picked up and put down,it is used for making relative changes in the position of the screen.
- Most general purpose graphics systems now include a mouse and a keyboard as the primary input devices.

***Trackballs and Spaceballs:***

- A trackball is a ball device that can be rotated with the fingers or palm of the hand to produce screen cursor movement.
- Laptop keyboards are equipped with a trackball to eliminate the extra space required by a mouse.
- Spaceball is an extension of two-dimensional trackball concept.
- Spaceballs are used for three-dimensional positioning and selection operations in virtual- reality systems,modeling,animation,CAD and other applications.

***Joysticks:***

- Joystick is used as a positioning device,which uses a small vertical lever(stick) mounded on a base.It is used to steer the screen cursor around and select screen position with the stick movement.
- A push or pull on the stick is measured with strain gauges and converted to movement of the screen cursor in the direction of the applied pressure.

***Data Gloves:***

- Data glove can be used to grasp a virtual object.The glove is constructed with a series of sensors that detect hand and finger motions.
- Input from the glove is used to position or manipulate objects in a virtual scene.

***Digitizers:***

- Digitizer is a common device for drawing,painting or selecting positions.
- Graphics tablet is one type of digitizer,which is used to input 2-dimensional coordinates by activating a hand cursor or stylus at selected positions on a flat surface.
- A hand cursor contains cross hairs for sighting positions and stylus is a pencil-shaped device that is pointed at positions on the tablet.

***Image Scanners:***

- Drawings,graphs,photographs or text can be stored for computer processing with an image scanner by passing an optical scanning mechanism over the information to be stored.
- Once we have the representation of the picture, then we can apply various image- processing method to modify the representation of the picture and various editing operations can be performed on the stored documents.

***Touch Panels:***

- Touch panels allow displayed objects or screen positions to be selected with the touch of a finger.
- Touch panel is used for the selection of processing options that are represented as a menu of graphical icons.
- Optical touch panel-uses LEDs along one vertical and horizontal edge of the frame.
- Acoustical touch panels generates high-frequency sound waves in horizontal and vertical directions across a glass plate.

***Light Pens:***

- Light pens are pencil-shaped devices used to select positions by detecting the light coming from points on the CRT screen.
- To select positions in any screen area with a light pen,we must have some nonzero light intensity emitted from each pixel within that area.
- Light pens sometimes give false readings due to background lighting in a room.

## Coordinate Representations

**Modeling Coordinates:** Define the shapes of individual objects using local coordinates.

**World Coordinates:** Place these objects within a scene using a global reference frame.

**Viewing Pipeline:** Process the scene through various output-device reference frames for display.

**Normalized Coordinates:** Convert the scene to normalized coordinates, typically ranging from -1 to 1 or 0 to 1.

**Device Coordinates:** Finally, transform the scene to device or screen coordinates for display.

## Introduction to OpenGL

### Basic OpenGL Library

- Provides functions for graphics primitives, attributes, transformations, viewing, etc.
- Function names are prefixed with "gl" (e.g., glBegin, glClear).
- Symbolic constants use "GL_" prefix and all caps (e.g., GL_RGB, GL_POLYGON).
- Specific data types (e.g., GLbyte, GLfloat) ensure compatibility across systems.

### Related Libraries

1. **OpenGL Utility (GLU):** Prefixed with "glu", handles complex tasks like setting viewing/projection matrices and rendering complex objects.
2. **Open Inventor:** Provides routines for 3D applications in C++.
3. **Window-System Libraries:** Manage display windows (e.g., AGL, WGL, GLX).
4. **OpenGL Utility Toolkit (GLUT):** Prefixed with "glut", provides device-independent window management.

### Header Files

- Include OpenGL core library headers (gl.h, glu.h) and for GLUT (glut.h).
- On Windows: #include <windows.h> precedes OpenGL headers.
- On Apple OS X: #include <GLUT/glut.h>.

### Display-Window Management Using GLUT

1. **Initialize GLUT:**

glutInit(&argc, argv);

2. **Create Window:**

glutCreateWindow("An Example OpenGL Program");

3. **Define Display Content:**

glutDisplayFunc(lineSegment);

4. **Activate Window:**

glutMainLoop();

**Additional GLUT Functions**

- Set window position: glutInitWindowPosition(x, y).
- Set display mode: glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB).

**Complete OpenGL Program Example**

**Initialize:**

```
glClearColor(1.0, 1.0, 1.0, 0.0);
glMatrixMode(GL_PROJECTION);
gluOrtho2D(0.0, 200.0, 0.0, 150.0);
```

**Define Line Segment:**

```
void lineSegment(void) {
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(0.0, 0.4, 0.2);
   glBegin(GL_LINES);
   glVertex2i(180, 15);
   glVertex2i(10, 145);
   glEnd();
   glFlush();
```

```
}
```

**Main Function:**

```
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(50, 100);
    glutInitWindowSize(400, 300);
    glutCreateWindow("An Example OpenGL Program");
    init();
    glutDisplayFunc(lineSegment);
    glutMainLoop();
}
```

***Screen co-ordinates:***

- ✓ Locations on a video monitor are referenced in integer screen coordinates, which correspond to the integer pixel positions in the frame buffer.
- ✓ Scan-line algorithms for the graphics primitives use the coordinate descriptions to determine the locations of pixels
- ✓ Example: given the endpoint coordinates for a line segment, a display algorithm must calculate the positions for those pixels that lie along the line path between the endpoints.
- ✓ Since a pixel position occupies a finite area of the screen, the finite size of a pixel must be taken into account by the implementation algorithms.
- ✓ For the present, we assume that each integer screen position references the centre of a pixel area.
- ✓ Once pixel positions have been identified the color values must be stored in the frame buffer

Assume we have available a low-level procedure of the form

**i) setPixel (x, y);**

- stores the current color setting into the frame buffer at integer position(x, y), relative to the position of the screen-coordinate origin

**ii) getPixel (x, y, color);**

- Retrieves the current frame-buffer setting for a pixel location;
- Parameter color receives an integer value corresponding to the combined RGB bit codes stored for the specified pixel at position (x,y).
- Additional screen-coordinate information is needed for 3D scenes.
- For a two-dimensional scene, all depth values are 0.

## Absolute and Relative Coordinate Specifications

***Absolute coordinate:***

- So far, the coordinate references that we have discussed are stated as absolute coordinate values.
- This means that the values specified are the actual positions within the coordinate system in use.

***Relative coordinates:***

- However, some graphics packages also allow positions to be specified using relative coordinates.
- This method is useful for various graphics applications, such as producing drawings with pen plotters, artist's drawing and painting systems, and graphics packages for publishing and printing applications.
- Taking this approach, we can specify a coordinate position as an offset from the last position that was referenced (called the current position).

## *Specifying a Two-Dimensional World-Coordinate Reference Frame in OpenGL*

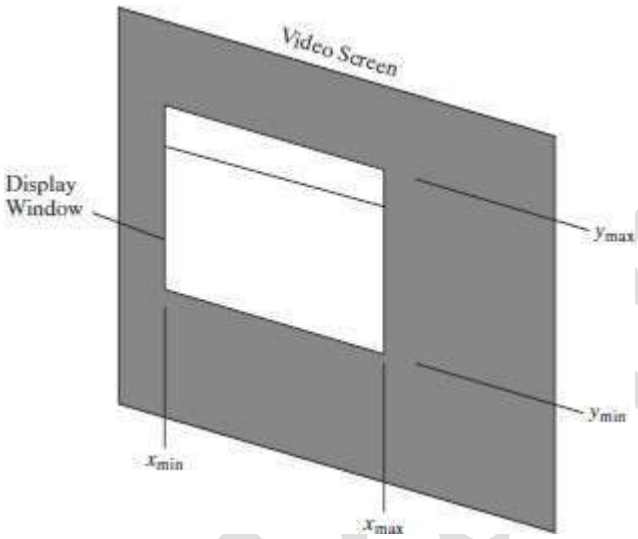
- The gluOrtho2D command is a function we can use to set up any 2D Cartesian reference frames.
- The arguments for this function are the four values defining the x and y coordinate limits for the picture we want to display.
- Since the gluOrtho2D function specifies an orthogonal projection, we need also to be sure that the coordinate values are placed in the OpenGL projection matrix.
- In addition, we could assign the identity matrix as the projection matrix before defining the world-coordinate range.
- This would ensure that the coordinate values were not accumulated with any values we may have previously set for the projection matrix.
- Thus, for our initial two-dimensional examples, we can define the coordinate frame for the screen display window with the following statements

***glMatrixMode (GL_PROJECTION); glLoadIdentity ( );***

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

The display window will then be referenced by coordinates (xmin, ymin) at the lower-left corner and by coordinates (xmax, ymax) at the upper-right corner, as shown in Figure below

- We can then designate one or more graphics primitives for display using the coordinate




reference specified in the gluOrtho2D statement.

- If the coordinate extents of a primitive are within the coordinate range of the display window, all of the primitive will be displayed.
- Otherwise, only those parts of the primitive within the display-window coordinate limits will be shown.
- Also, when we set up the geometry describing a picture, all positions for the OpenGL primitives must be given in absolute coordinates, with respect to the reference frame defined in the gluOrtho2D function.

**OpenGL Geometric Primitives**

- **Points (GL_POINTS)**:
    - Each vertex is displayed as a single point.
    - Size is typically one pixel unless specified otherwise.
    - Coordinates can be specified in 2D (glVertex2i) or 3D (glVertex3f) formats.

**Examples:**

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

**Lines (GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP)**:

- GL_LINES: Connects pairs of vertices as individual line segments.
- GL_LINE_STRIP: Connects successive vertices with lines, not closing the path.
- GL_LINE_LOOP: Connects vertices forming a closed loop.

**Examples:**

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

**Point Attributes**

- **Color (glColor3f)**:

- Sets the color of subsequent primitives (points, lines, etc.).
- Parameters are RGB values between 0.0 and 1.0.

- **Size (glPointSize)**:
  - Sets the size of points to be rendered.
  - Size is specified in floating-point values, with each unit typically representing a pixel.

**Example Program:**

```
glColor3f(1.0, 0.0, 0.0); // Red color
glBegin(GL_POINTS);
glVertex2i(50, 100);
glPointSize(2.0); // Double size
glColor3f(0.0, 1.0, 0.0); // Green color
glVertex2i(75, 150);
glPointSize(3.0); // Triple size
glColor3f(0.0, 0.0, 1.0); // Blue color
glVertex2i(100, 200);
glEnd();
```

**OpenGL Line-Width Function**

- **Setting Line Width**:
  - Use glLineWidth(width) to set the width of lines in OpenGL.
  - width is a floating-point value which is rounded to the nearest nonnegative integer.
  - Default width is 1.0 if width rounds to 0.0.
  - Implementation-dependent: Some systems may support only certain widths.
  -

**OpenGL Line-Style Function**

- **Setting Line Style**:
    - Use glLineStipple(repeatFactor, pattern) to set the line style.
    - pattern is a 16-bit integer where each bit determines whether a pixel is "on" or "off".
    - repeatFactor specifies how many times each bit in the pattern is repeated before moving to the next bit.
    - Default pattern (0xFFFF) creates a solid line.
- **Activating and Deactivating Line Style**:
    - Enable line style with glEnable(GL_LINE_STIPPLE) before drawing lines with stipple patterns.
    - Disable line style with glDisable(GL_LINE_STIPPLE) to revert to solid lines.

**Example Code**

Here's an example demonstrating how to use line width and line style functions in OpenGL:

```
typedef struct {
   float x, y;
} wcPt2D;
void linePlot(wcPt2D dataPts[5]) {
   int k;
   glBegin(GL_LINE_STRIP);
   for (k = 0; k < 5; k++) {
      glVertex2f(dataPts[k].x, dataPts[k].y);
   }
   glEnd();
}
```

```
void drawLinesWithStyles(wcPt2D dataPts[5]) {
    glEnable(GL_LINE_STIPPLE);

    // Plot a dash-dot, standard-width polyline
    glLineStipple(1, 0x1C47);
    linePlot(dataPts);

    // Plot a dashed, double-width polyline
    glLineStipple(1, 0x00FF);
    glLineWidth(2.0);
    linePlot(dataPts);
    // Plot a dotted, triple-width polyline
    glLineStipple(1, 0x0101);
    glLineWidth(3.0);
    linePlot(dataPts);
    glDisable(GL_LINE_STIPPLE);
}
/* Example usage */
wcPt2D dataPoints[5] = { {50, 100}, {75, 150}, {100, 200}, {125, 250}, {150, 300} };

// Call function to draw lines with different styles
drawLinesWithStyles(dataPoints);
```