MODULE -1

Overview: Computer Graphics and OpenGL

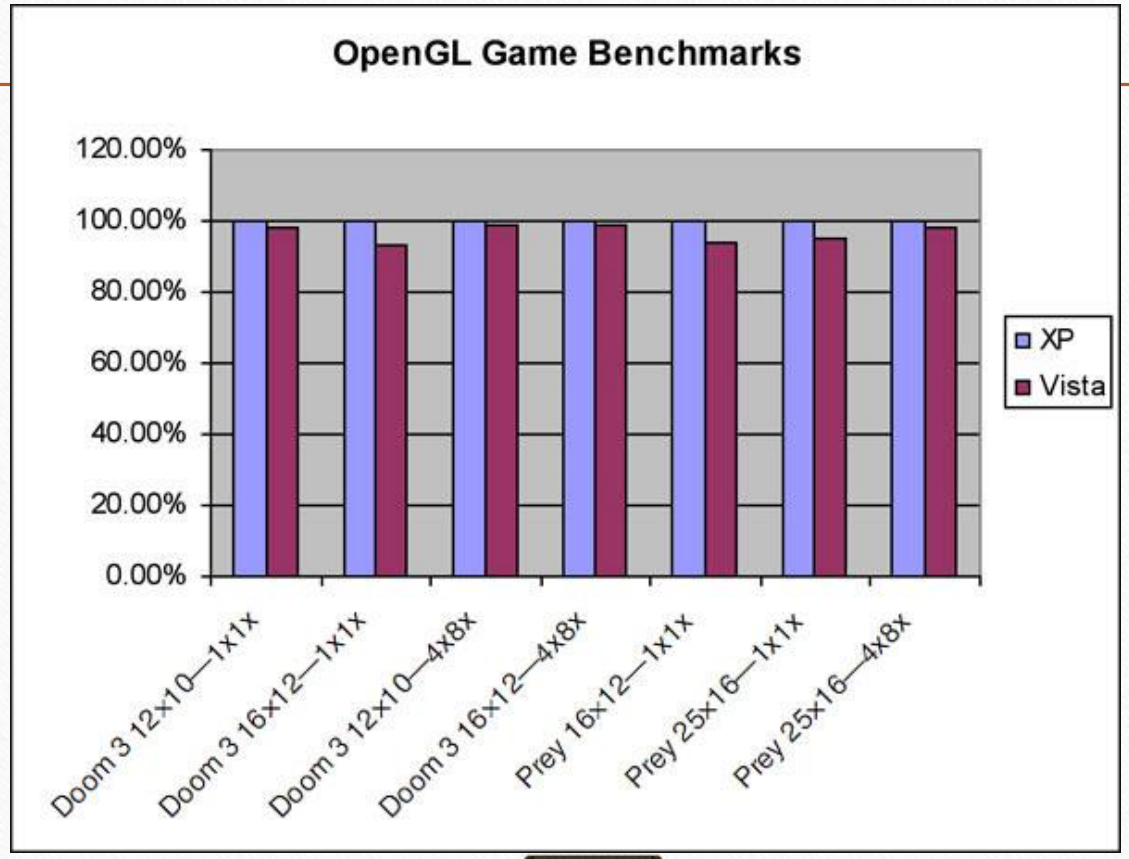
Why computer graphics?

- Visual system offers:
  - Parallel input
  - Parallel processing
- Computer graphics: ideal for human-computer communication

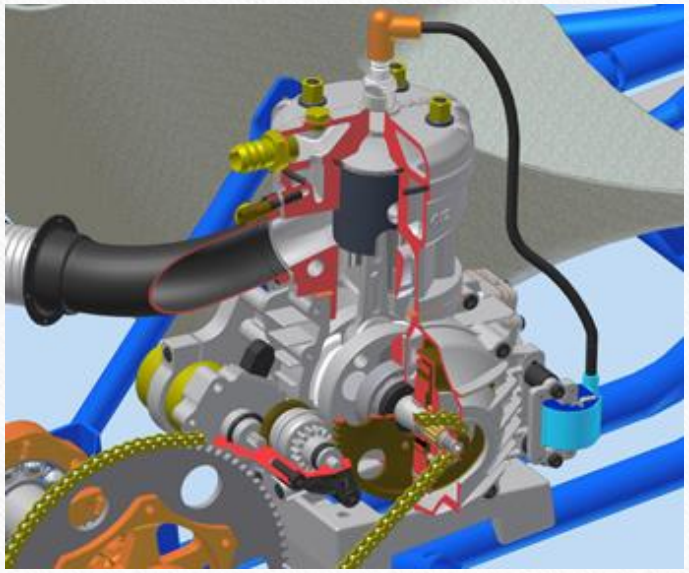
Applications

- Graphs and charts
- Computer-Aided Design
- Virtual Reality
- Data Visualization
- Education and training
- Computer Art
- Movies
- Games
- Graphical User Interfaces

Business graphics

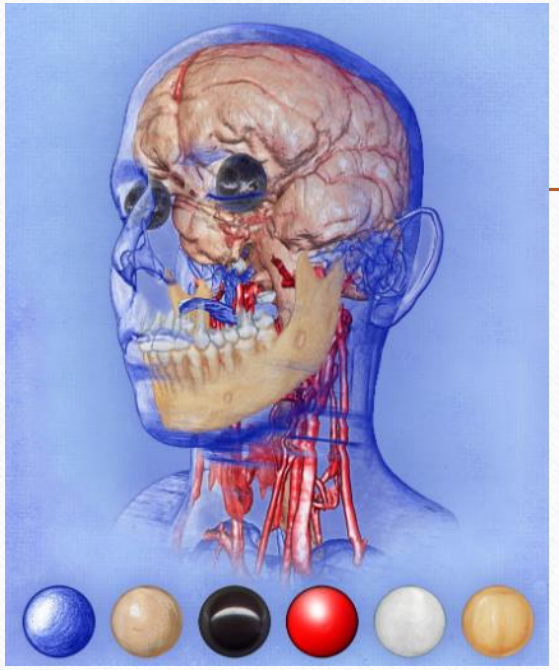


Computer-Aided Design

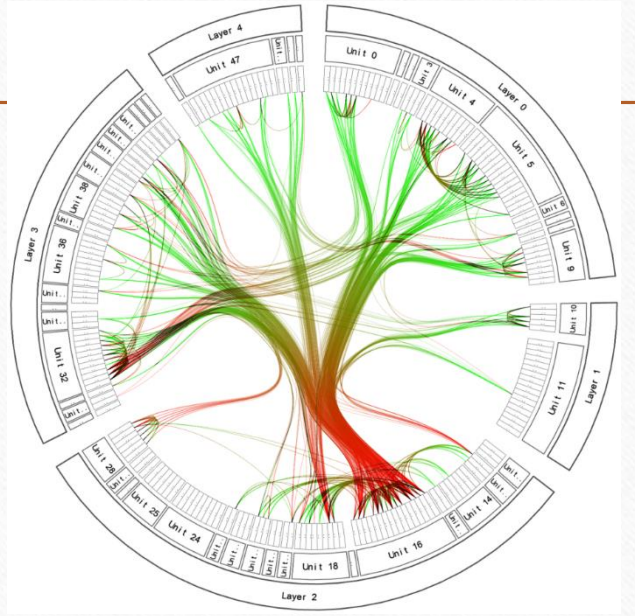


- AutoDesk
- IAME 2-stroke race kart engine

Data Visualization



Bruckner and Groeller,
TU Vienna, 2007



Holten, TU/e, 2007

Gaming

Movies


expression
depth of field
motion
fracture
water
reflection
hair

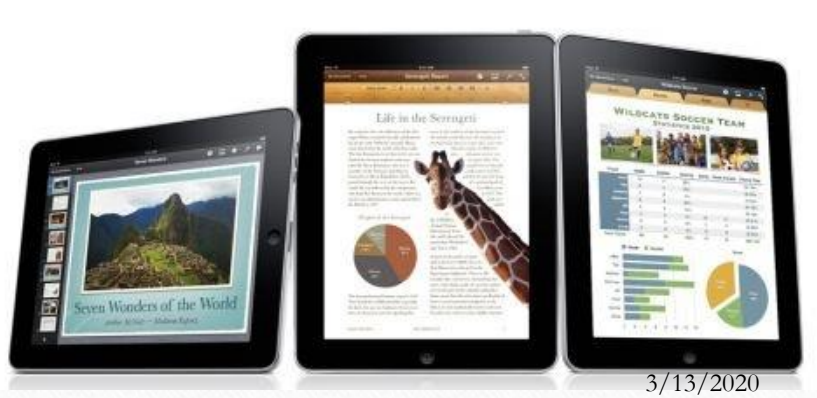
Hardware

- Fast development
- Now: Graphics Processing Unit (GPU), LCD-screen

Beyond the laptop screen

- Microsoft Surface

- Apple iPad

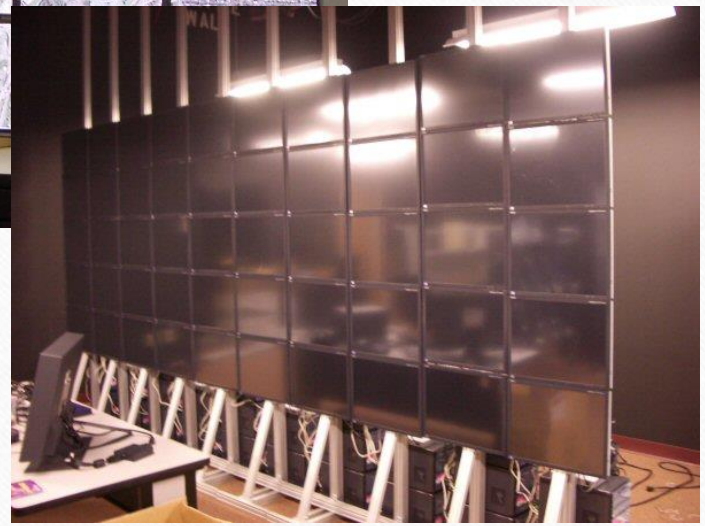


Beyond the laptop screen

- Roll-up screen, Philips

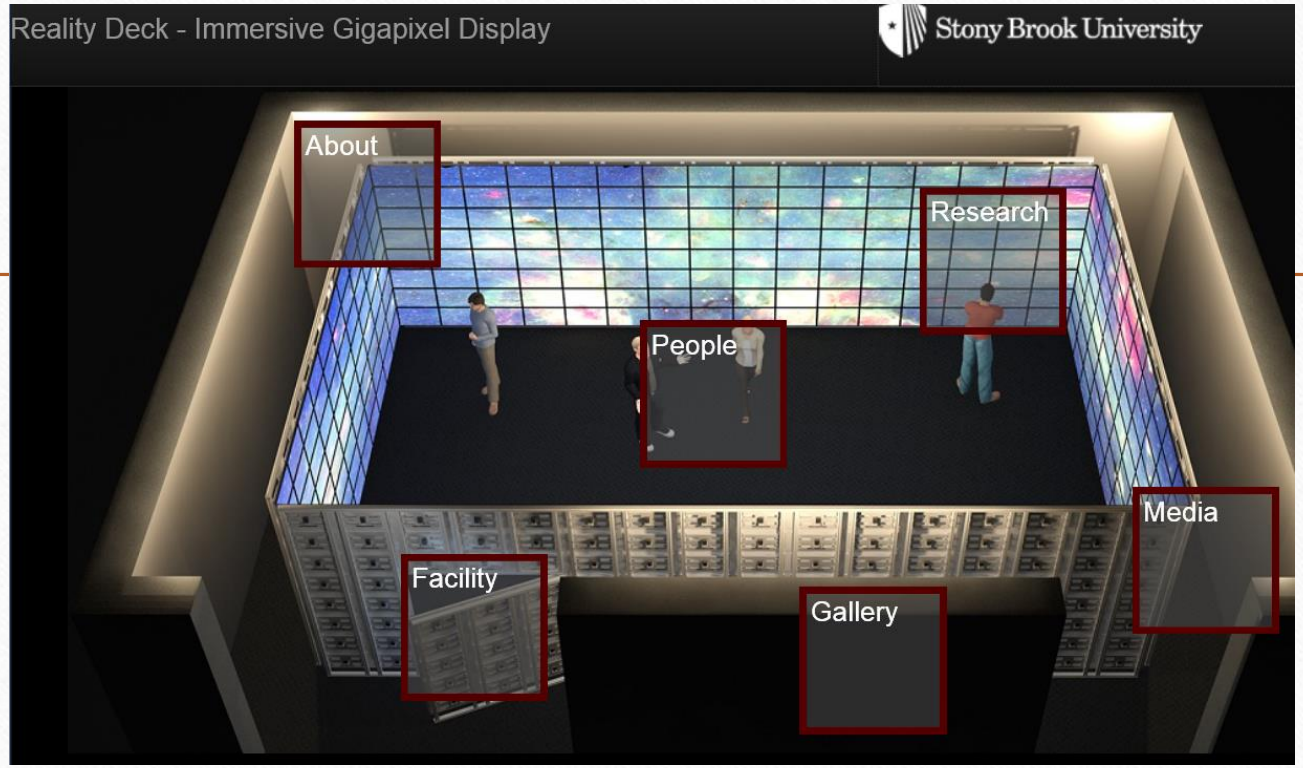
Beyond the laptop screen

- 24 screen configuration, Virginia Tech



- 50 LCD touchscreens

Beyond the laptop screen



- Reality Deck – Stony Brook University
- 416 2560×1440 27” monitors

Beyond the laptop screen

Head mounted displays

Parachute trainer

US Navy

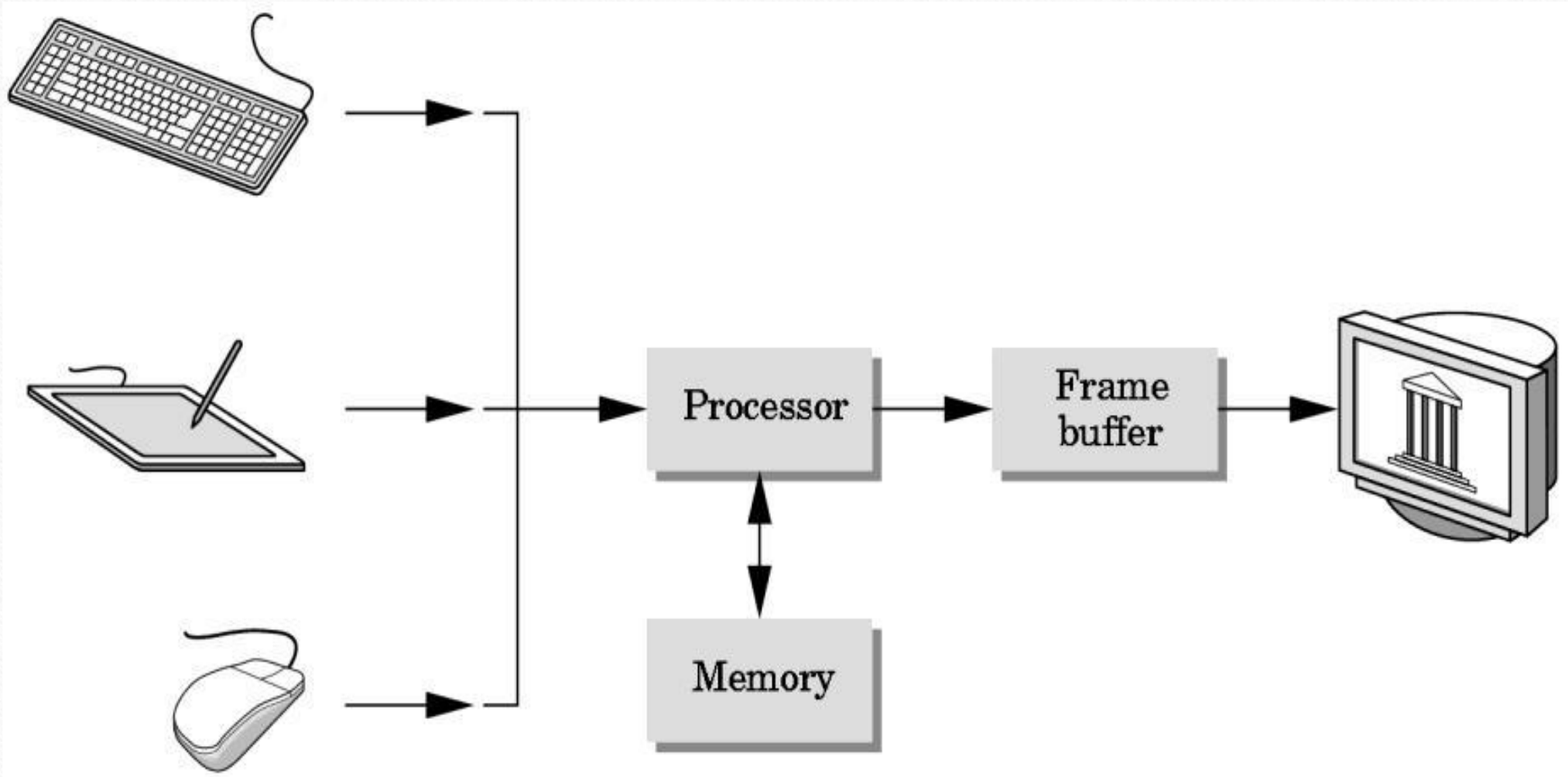
Beyond the laptop screen

- Oculus Rift

Basic **Graphics systems**

A Graphics system has 5 main elements:

- Input Devices
- Processor
- Memory
- Frame Buffer
- Output Devices

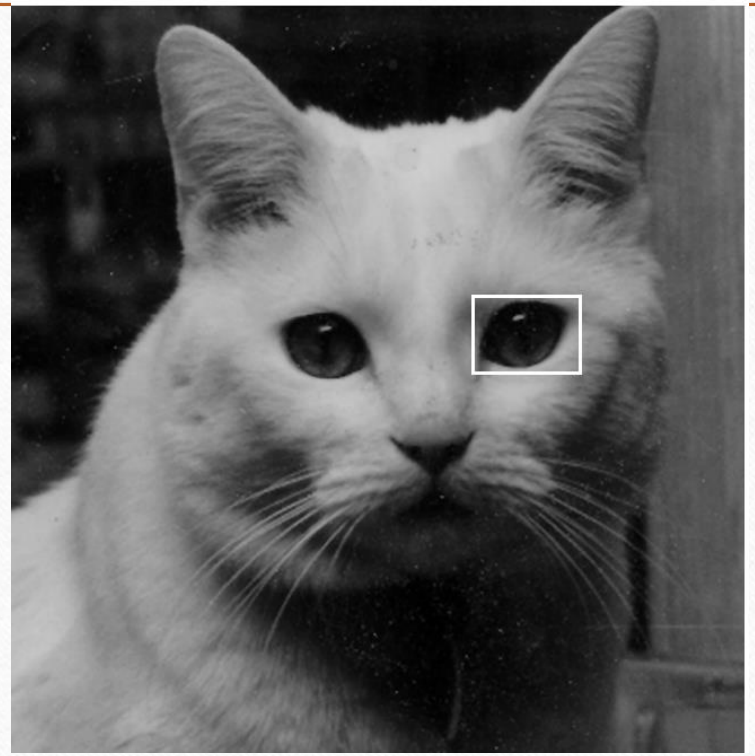

Processor
Frame buffer
Memory

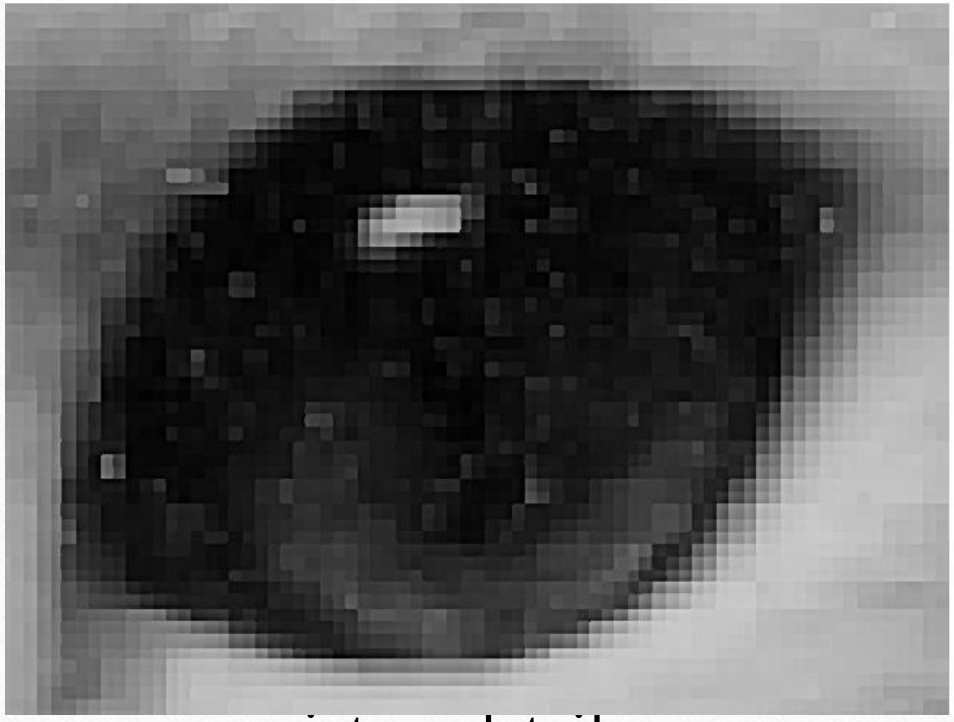
Pixels and Frame Buffer

Most graphics systems are pixel based – need of rasterization or scan conversion;

pixel = picture element

8 bits deep frame – 256 colors; 24 or 32 bits for RGB colors





picture detail

Pixels and the Frame Buffer

- A picture is produced as an array (raster) of picture elements(pixels).
- These pixels are collectively stored in the Frame Buffer.

Properties of frame buffer:

- Resolution – number of pixels in the frame buffer
- Depth or Precision – number of bits used for each pixel

  E.g.: 1 bit deep frame buffer allows 2 colors

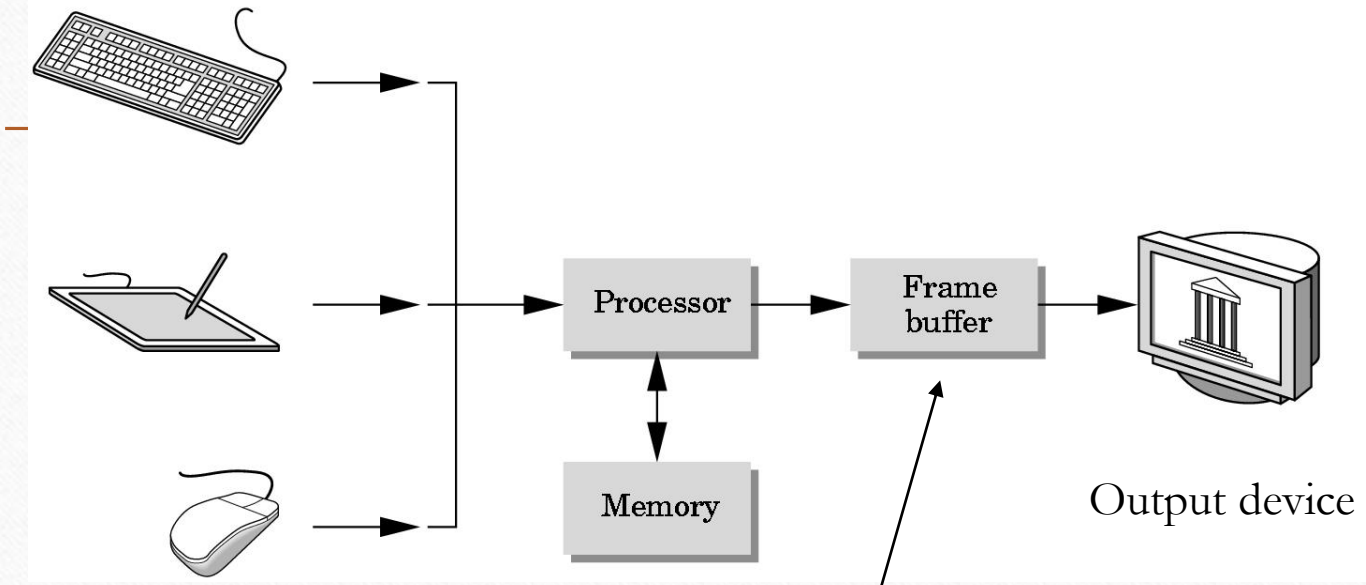
  8 bit deep frame buffer allows 256 colors.

- A Frame buffer is implemented either with special types of memory chips or it can be a part of system memory.

- In simple systems the CPU does both normal and graphical processing.

- Graphics processing - Take specifications of graphical primitives from application program and assign values to the pixels in the frame buffer It is also known as Rasterization or scan conversion.

Basic Graphics System

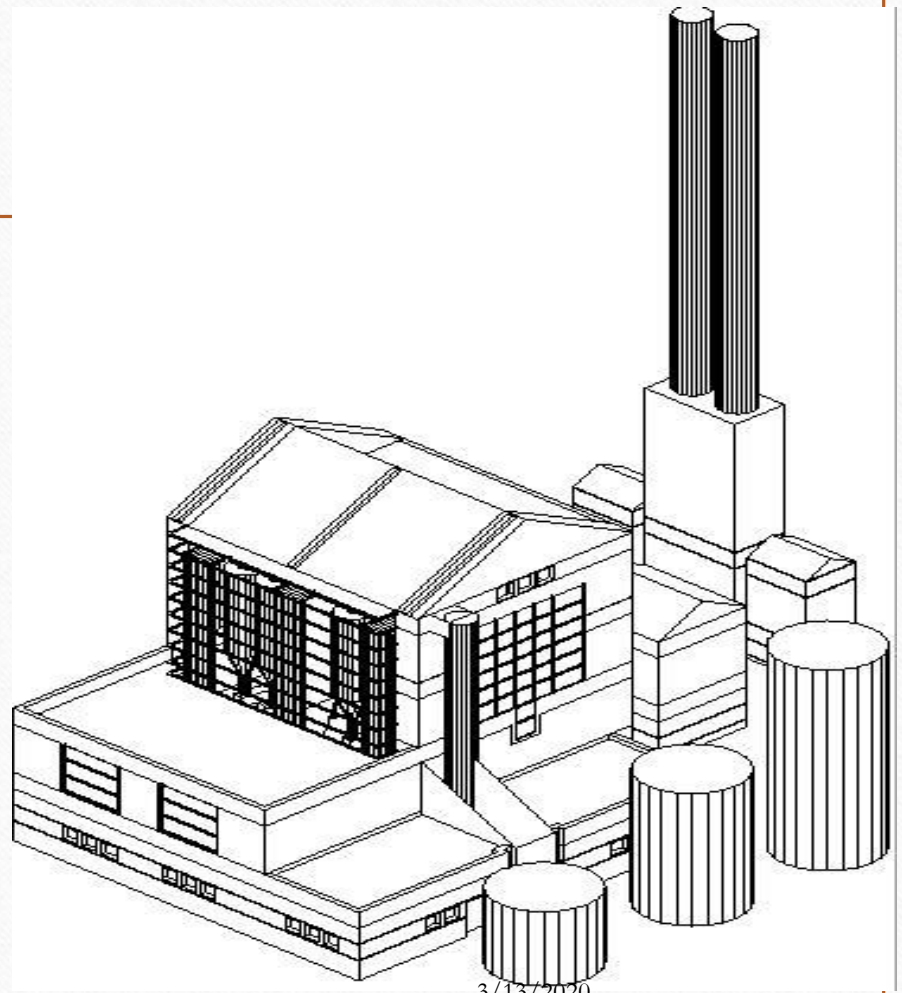


Applications of computer graphics:

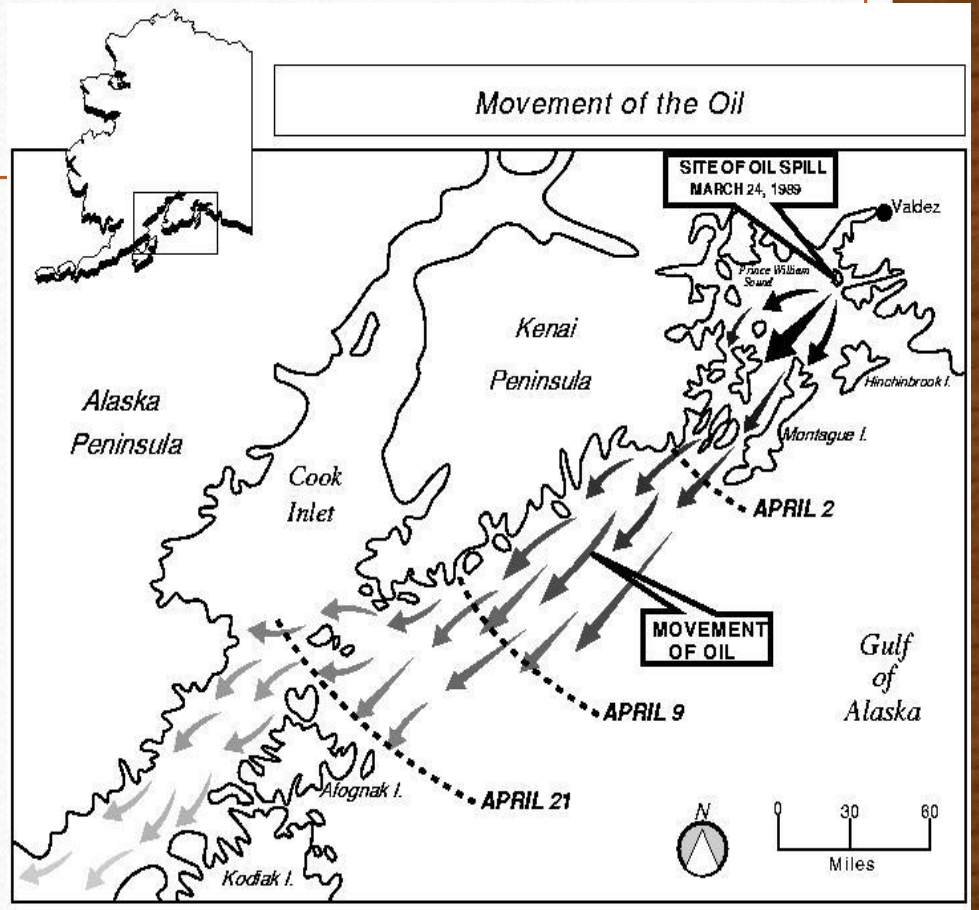
- Display Of Information
- Design
- Simulation & Animation
- User Interfaces

1.1 Display of Information

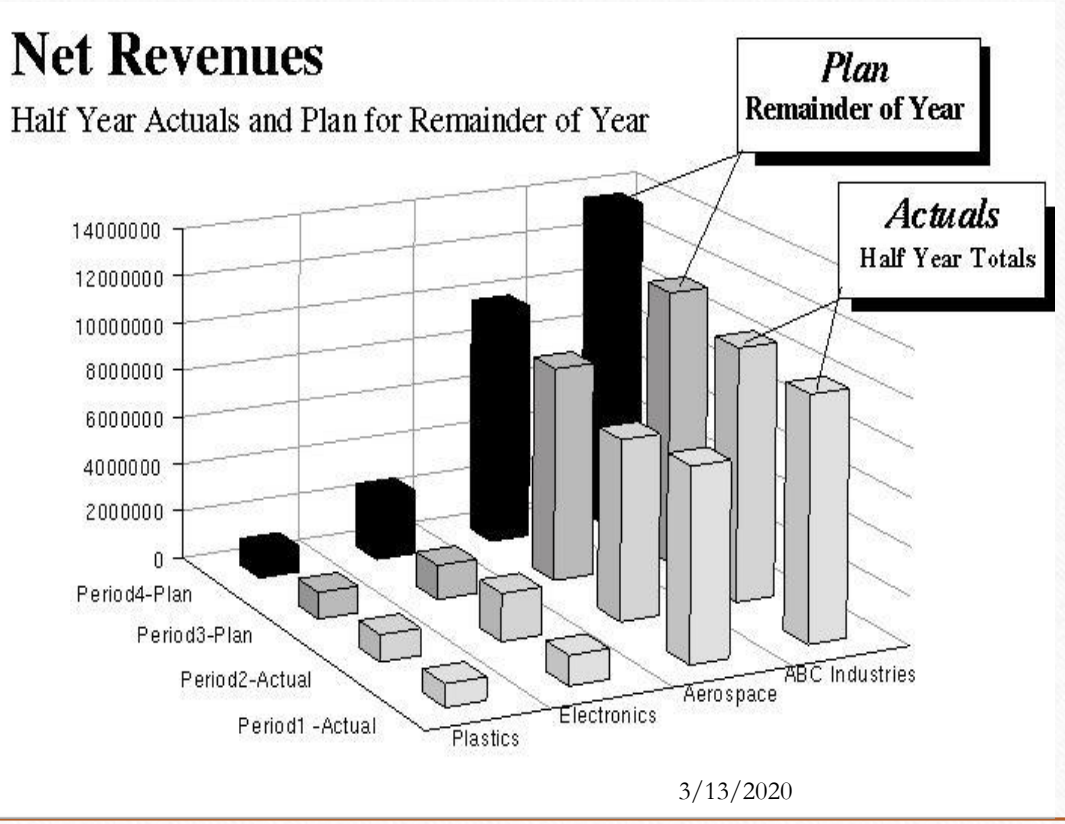
- Classical graphics techniques arose as a medium to convey information among people:
  - 4,000 years ago -- Babylonians: floor plans of buildings on stones
  - 2,000 years ago -- Greeks: Architectural ideas
    - Now we have Computer-based drafting programs



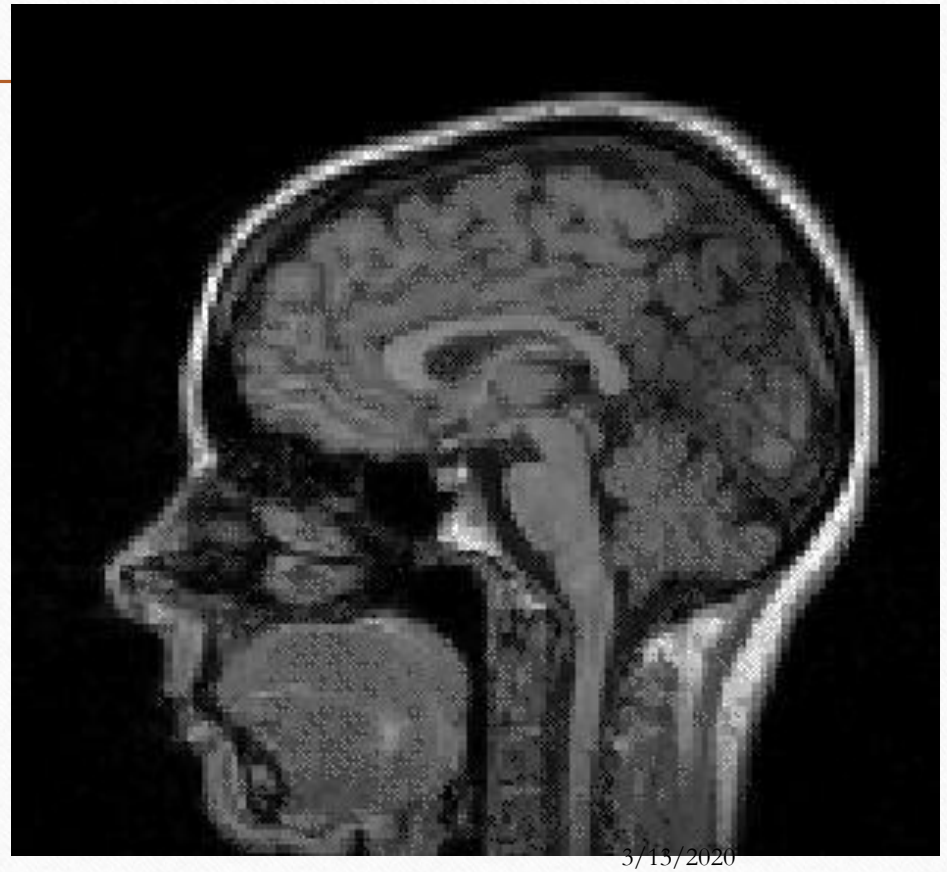
- For centuries -- Cartographers have developed maps to display celestial and geographical information.
  - Now maps can be developed and manipulated in real-time over the internet.



- Over the past 100 years -- workers in statistics have explored techniques for generating plots to convey information
  - Now we have computer plotting packages



- Medicine poses interesting and important data-analysis problems
  - CAT Scans, MRI's ultrasound, and other 3D data producing technologies



- The field of Scientific Visualization provides graphical tools that help these researchers and others interpret the vast quantities of data generated

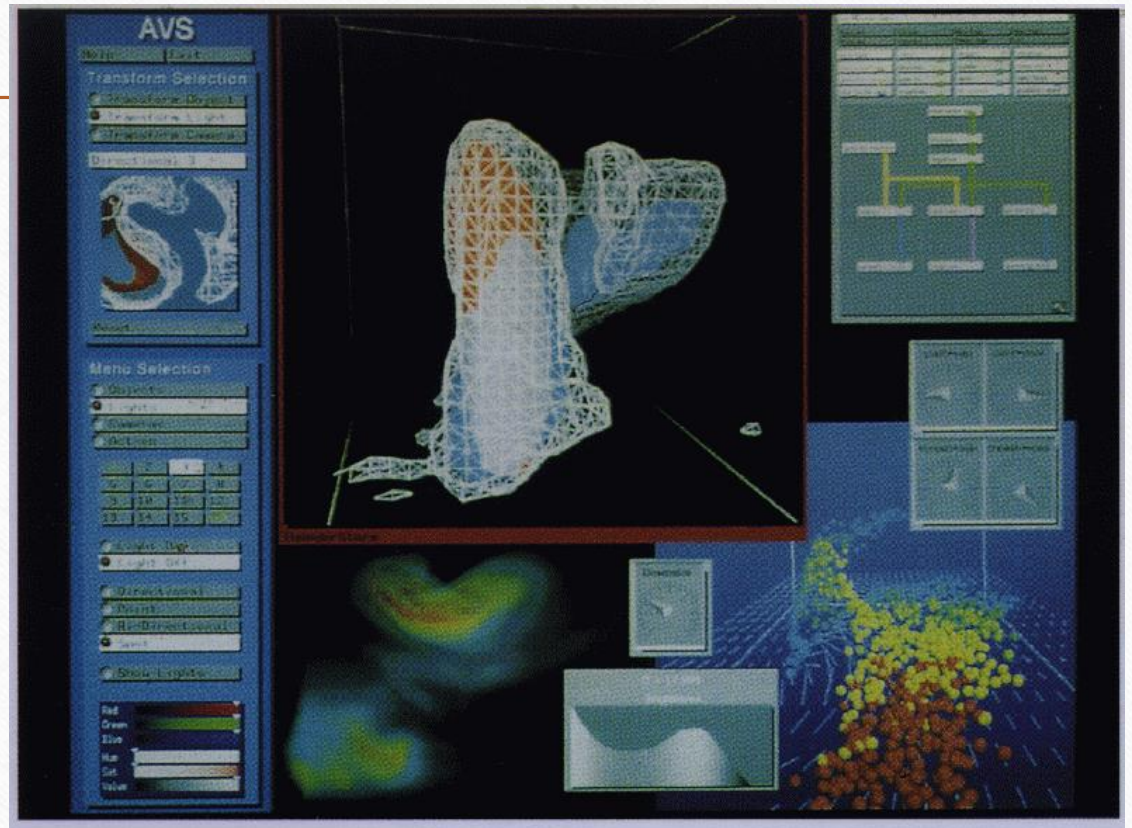
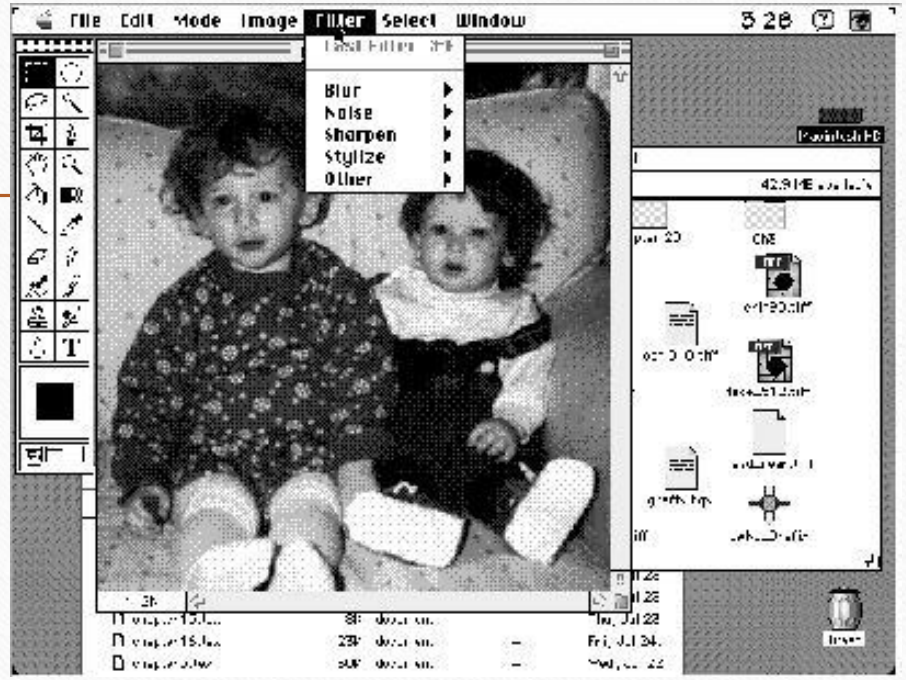


Plate 1. Severe tornadic storm, by R. Wilhelmson, L. Wicker, and C. Shaw, NCSA, University of Illinois. (Application Visualization System by Stardent Computer.)

- 1.2 Design
  - Engineering and Architecture are concerned with design
    - starting with a set of specification
    - seek a cost-effective (and esthetic) solution
    - This is an iterative process
  - The power of interacting with images on the screen
    - has been known for at least 40 years.
    - and today the use of interactive tools pervades the CAD field in areas such as architecture and VLSI design

- 1.3 Simulation
  - When did graphics begin to be used?
  - Why?
  - The field of VR has opened up many new horizons.
    - Same (or different) image in each eye
    - position tracking
    - interactive devices
    - This has led to training capabilities
      - NASA research grant

- 1.4 User Interfaces
  - Our interaction with computers has become dominated by a visual paradigm that includes:
    - windows,
    - icons,
    - menus, and
    - a pointing device



  - We have become so accustomed to this style of interface that we often forget that what we are doing is working with computer graphics.

- Video display devices
- Raster-scan systems
- Graphics workstations and viewing systems
- Input devices

Video Display Devices

- Cathode-ray tubes
- Raster-scan displays

- Random-scan displays
- Color CRT displays
- Flat-panel displays
- Three-dimensional viewing devices
- Stereoscopic and virtual-reality systems

Cathode-Ray Tubes

- Classical output device is a monitor.
- Cathode-Ray Tube (CRT)
  - Invented by Karl Ferdinand Braun (1897)
  - Beam of electrons directed from cathode (-)to phosphor-coated (fluorescent) screen (anode (+))
  - Directed by magnetic focusing and deflection coils (anodes) in vacuum filled tube
  - Phosphor emits photon of light, when hit by an electron, of varied persistence (long 15-20 ms for texts / short < 1ms for animation)
  - Refresh rate (50-60 Hz / 72-76 Hz) to avoid flicker / trail
  - Phosphors are organic compounds characterized by their persistence and their color (blue, red, green).

Cathode-Ray Tubes

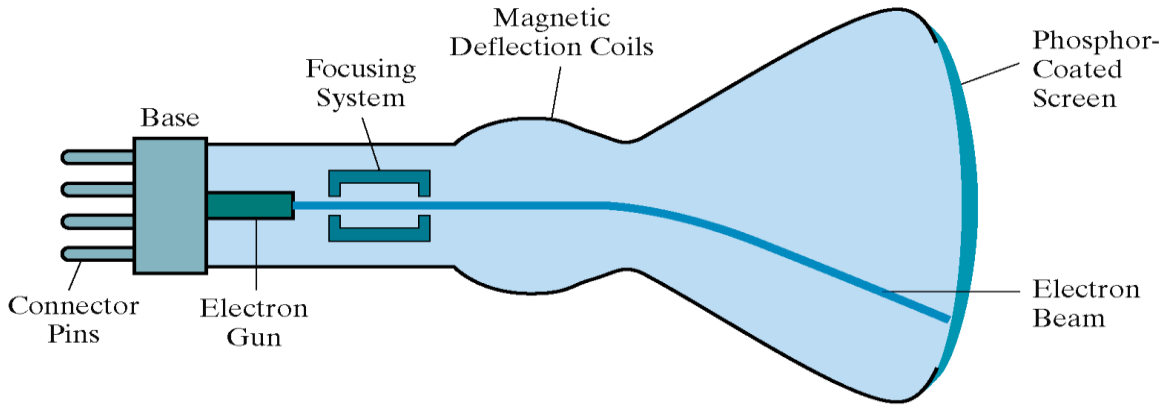


Figure 2-2

Basic design of a magnetic-deflection CRT.

Cathode-Ray Tubes

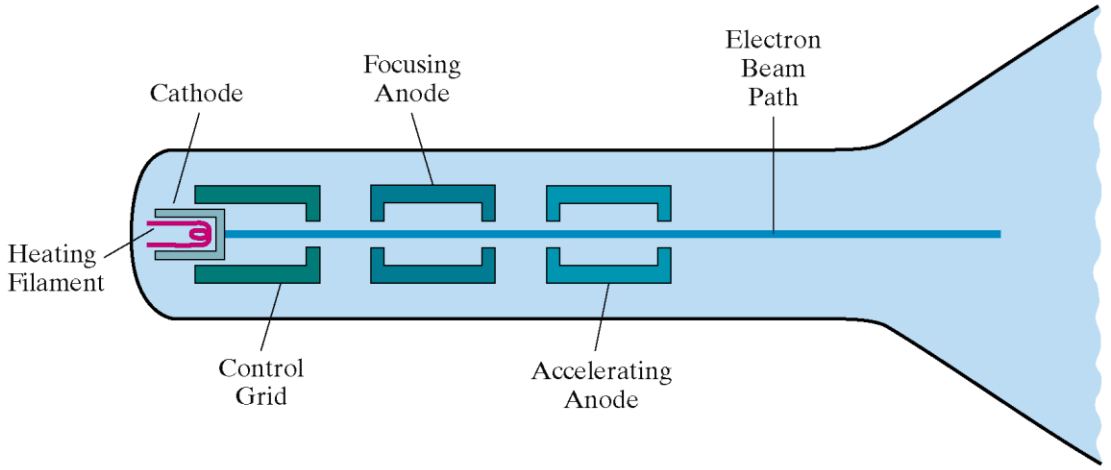


Figure 2-3

Operation of an electron gun with an accelerating anode.

Cathode-Ray Tubes

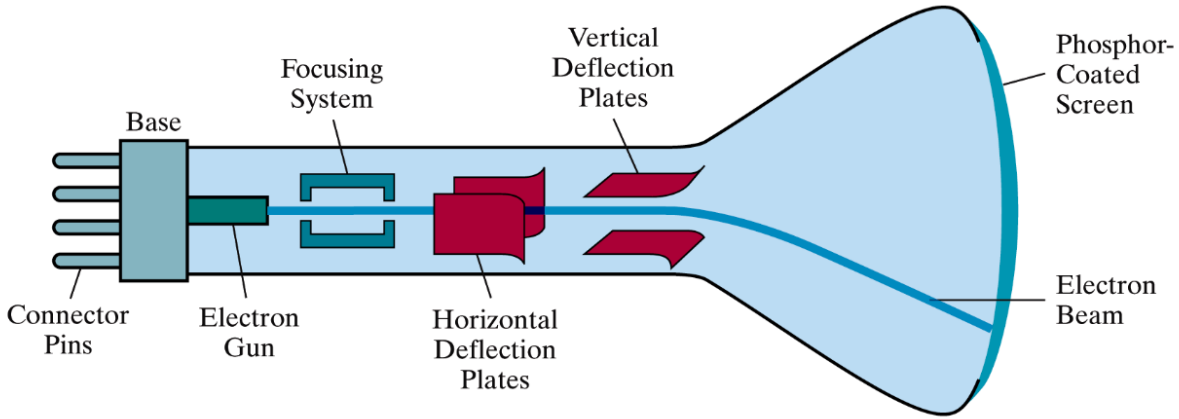


Figure 2-4

Electrostatic deflection of the electron beam in a CRT.

Cathode-Ray Tubes

- Cathode-Ray Tube (CRT)
  - Horizontal deflection and vertical deflection direct the electron beam to any point on the screen
  - Intensity knob: regulates the flow of electrons by controlling the voltage at the control grid (high voltage reduces the electron density and thus brightness)
  - Accelerating voltage from positive coating inside screen (anode screen) or an accelerating anode
- Image maintenance
  - Charge distribution to store picture information
    OR
  - Refresh CRT: refreshes the display constantly to maintain phosphor glow.

Cathode-Ray Tubes

- Characteristics of Cathode-Ray Tube (CRT)
  - ***Intensity*** is proportional to the number of electrons repelled in beam per second (***brightness***)
  - ***Resolution*** is the maximum number of points that can be displayed without overlap; is expressed as number of horizontal points by number of vertical points; points are called pixels (picture elements); example: resolution 1024 x 768 pixels. Typical resolution is 1280 x 1024 pixels.
- High-definition systems: high resolution systems.

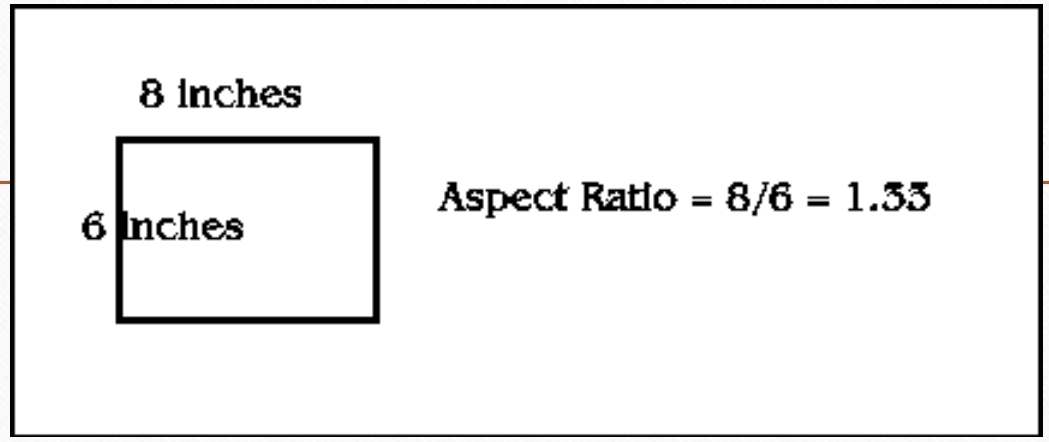
Cathode-Ray Tubes

- Focusing
  - ***Focusing*** forces the electron beam to converge to a point on the monitor screen
  - Can be electrostatic (lens) or magnetic (field)
- Deflection
  - ***Deflection*** directs the electron beam horizontally and/or vertically to any point on the screen
  - Can be controlled by electric (deflection plates, slide 9) or magnetic fields (deflection coils, slide 5)
  - Magnetic coils: two pairs (top/bottom, left/right) of tube neck
  - Electric plates: two pairs (horizontal, vertical)

Cathode-Ray Tubes

- Aspect ratio
  - ***Aspect ratio*** is the ratio of horizontal pixels to vertical pixels for an equal length line.
  - It is the ratio of the horizontal dimension over the vertical dimension.

Cathode-Ray Tubes



(from SIGGRAPH)

If resolution of 640 x 480 pixels:

→Horizontal 640/8 = 80 pixels / inch

→Vertical 480/6 = 80 pixels / inch

Square pixels (no distortion).

Raster-scan Displays

- Video displays can be either raster-scan or random-scan displays.
- Raster-scan display is the most common type of monitor using a CRT.
- The electron beam scans the screen from top to bottom one row at a time. Each row is called a scan line.
- The electron beam is turned on and off to produce a collection of dots painted one row at a time. These will form the image.
- A ***raster*** is a matrix of pixels covering the screen area and is composed of raster lines.

Raster-scan Displays

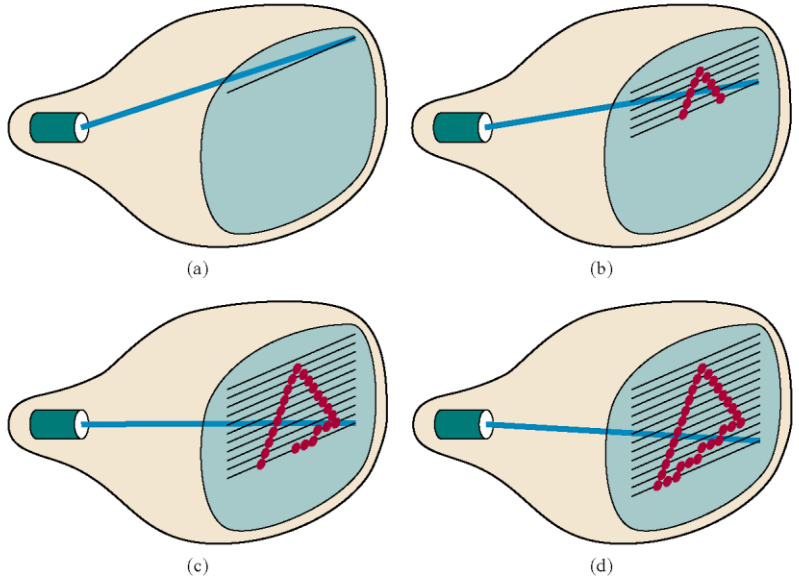


Figure 2-7
A raster-scan system displays an object as a set of discrete points across each scan line.

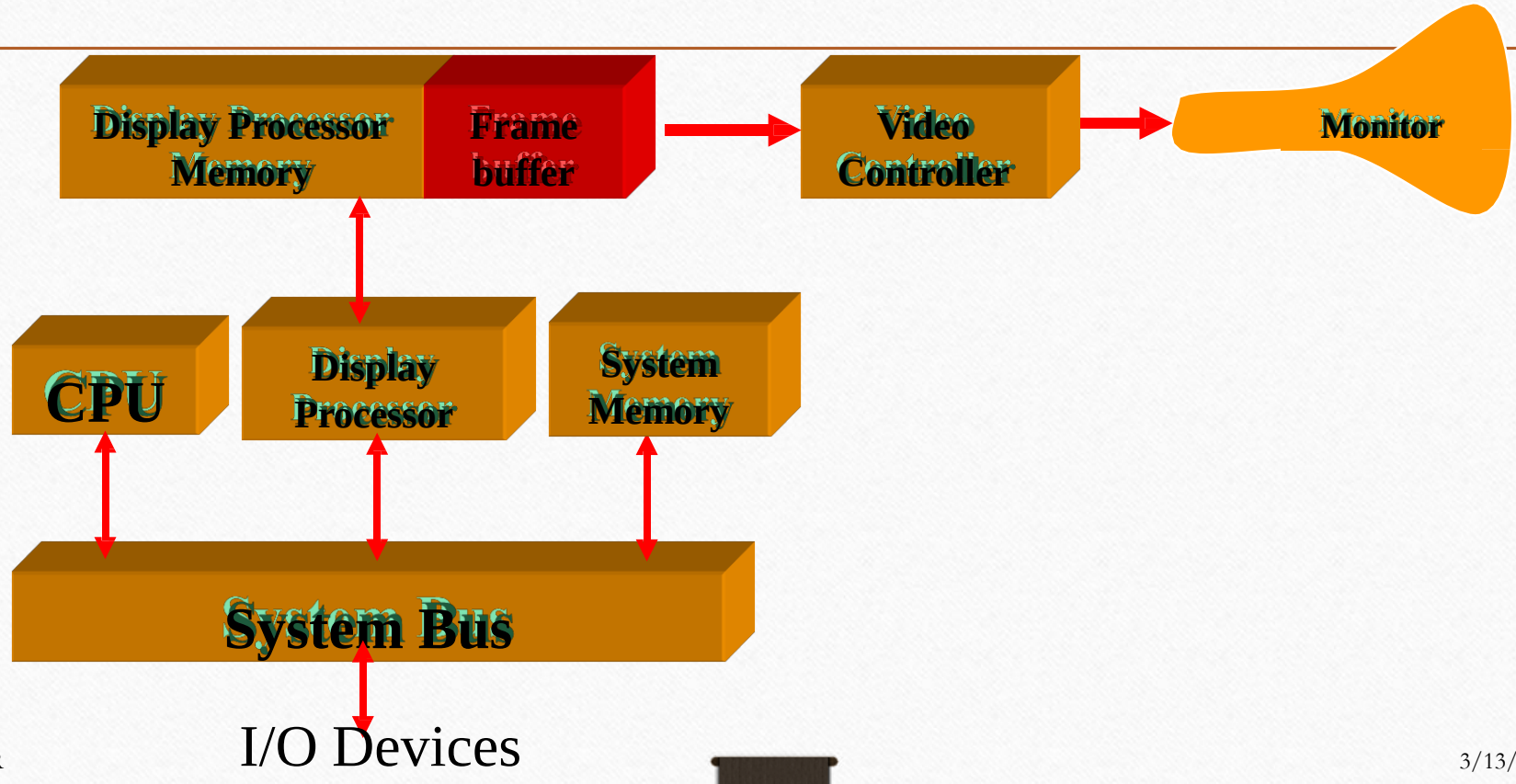
Raster-scan Displays

- The image is stored in a ***frame buffer*** containing the total screen area and where each memory location corresponds to a pixel.
- In a monochrome system, each bit is 1 or 0 for the corresponding pixel to be on or off (bitmap).
- The display processor scans the frame buffer to turn electron beam on/off depending if the bit is 1 or 0.
- For color monitors, the frame buffer also contains the color of each pixel (color buffer) as well as other characteristics of the image (gray scale, …). 8 bits/pixel → 0..255 (pixmap).
- Depth of the buffer area is the number of bits per pixel (bit planes), up to 24.
- Examples: television panels, printers, PC monitors (99% of raster-scan)...

Raster-scan Displays

- Refresh rate: 24 is a minimum to avoid flicker, corresponding to 24 Hz (1 Hz = 1 refresh per second)
- Current raster-scan displays have a refresh rate of at least 60 frames (60 Hz) per second, up to 120 (120 Hz).
- Uses large memory: 640x480 → 307200 bits → 38 kB
- Refresh procedure:
  - Horizontal retrace – beam returns to left of screen
  - Vertical retrace – bean returns to top left corner of screen
  - Interlaced refresh – display first even-numbered lines, then odd-numbered lines
    permits to see the image in half the time
    useful for slow refresh rates (30 Hz shows as 60 Hz).

Raster-scan Displays - Architecture



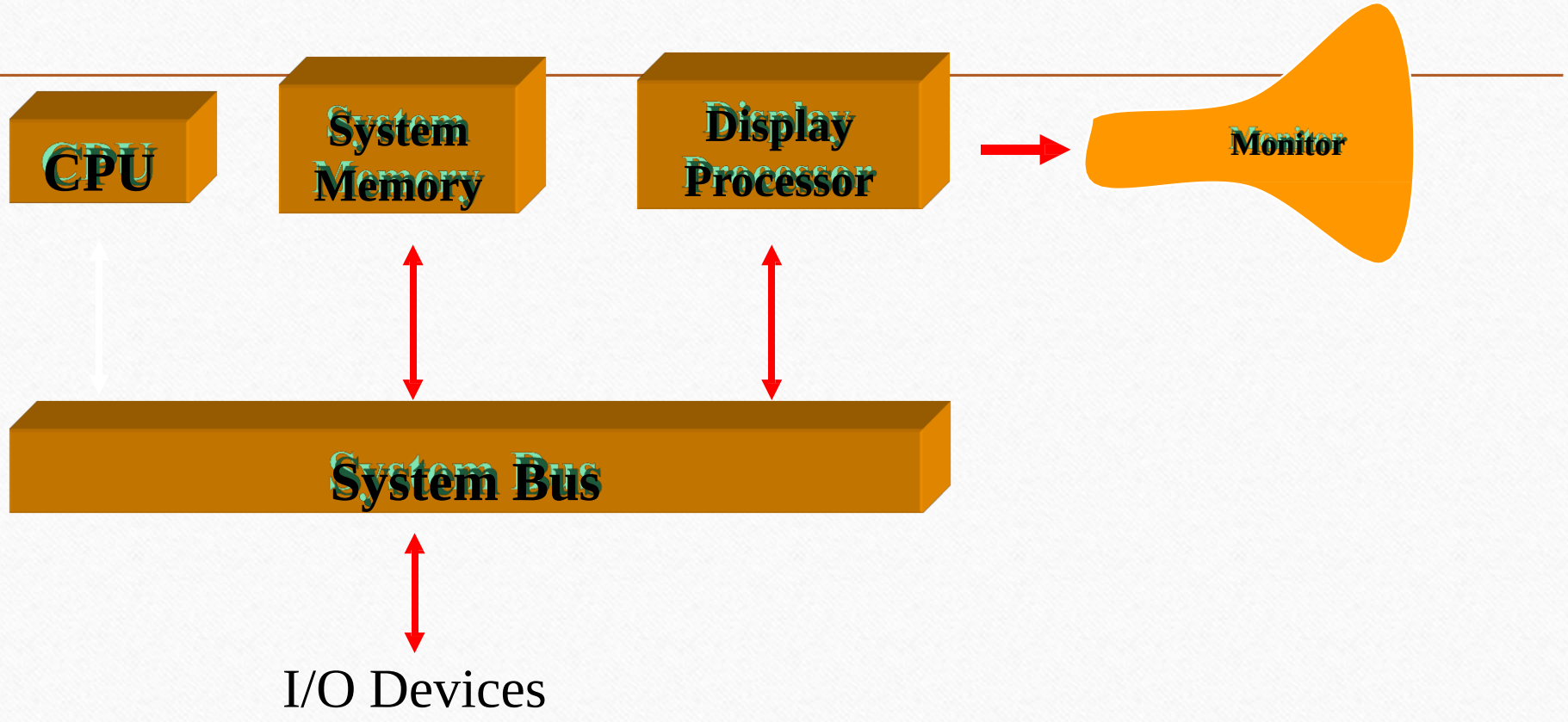
Random-scan Displays

- Random scan systems are also called vector, stroke-writing, or calligraphic displays.
- The electron beam directly draws the picture in any specified order.
- A pen plotter is an example of such a system.
- Picture is stored in a display list, refresh display file, vector file, or display program as a set of line drawing commands.
- Refreshes by scanning the list 30 to 60 times per second.
- More suited for line-drawing applications such as architecture and manufacturing.

Random-scan Displays

- Advantages:
  - High resolution
  - Easy animation
  - Requires little memory
- Disadvantages:
  - Requires intelligent electron beam (processor controlled)
  - Limited screen density, limited to simple, line-based images
  - Limited color capability.
- Improved in the 1960's by the Direct View Storage Tube (DVST) from Tektronix.

Random-scan Displays - Architecture



Color CRT Monitor

- Uses different phosphors, a combination of Red, Green, and Blue, to produce any color.
- Two methods:
  - Random scan: uses beam penetration.
    2 layers (Red, Green) phosphors; low speed electrons excite Red, high speed electrons excite Green, intermediate speed excite both to get yellow and orange. Color is controlled by electron beam voltage.
    Only produces a restricted set of colors.
  - Raster scan: uses a shadow mask with three electron guns: Red, Green, and Blue (RGB color model). Color is produced by adjusting the intensity level of each electron beam.
    Produces a wide range of colors, from 8 to several millions.

Color CRT Monitor

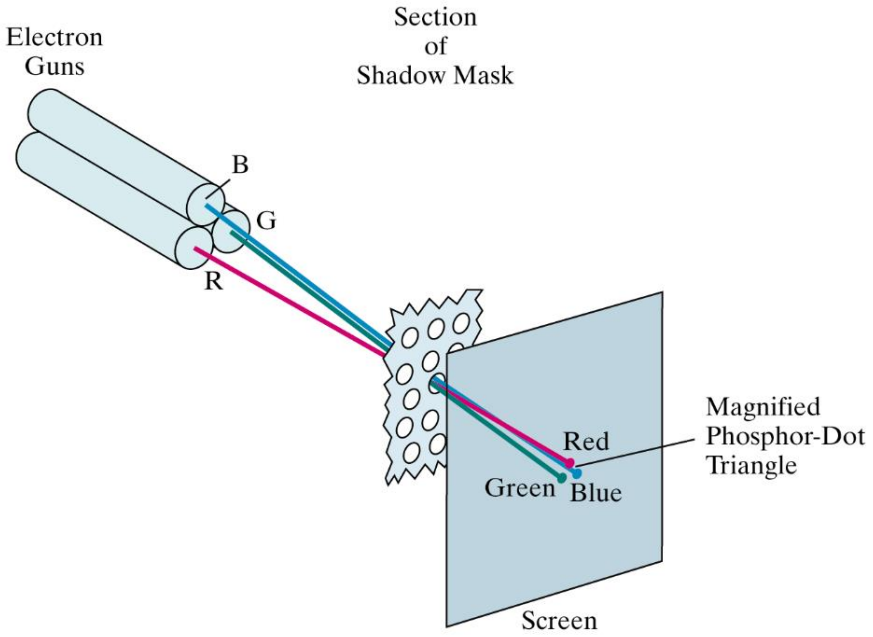


Figure 2-10

Operation of a delta-delta, shadow-mask CRT. Three electron guns, aligned with the triangular color-dot patterns on the screen, are directed to each dot triangle by a shadow mask.

Color CRT Monitor

R G B	color
0 0 0	black
0 0 1	blue
0 1 0	green
0 1 1	cyan
1 0 0	red
1 0 1	magenta
1 1 0	yellow
1 1 1	white

Color CRT Monitor

- Color CRT's are designed as RGB monitors also called full-color system or true-color system.
- Use shadow-mask methods with intensity from each electron gun (red, green, blue) to produce any color directly on the screen without preprocessing.
- Frame buffer contains 24 bits per pixel, for 256 voltage settings to adjust the intensity of each electron beam, thus producing a choice of up to 17 million colors for each pixel (256^3).

Flat Panel Displays

- Flat panel displays are video devices that are thinner, lighter, and require less power than CRT's.
- Examples: wall frames, pocket notepads, laptop computer screens, …
- Emissive versus non-emissive:
  - Emissive panels convert electrical energy into light:
    plasma panels, thin-film electroluminescent display device, light-emitting diodes.
  - Non-emissive convert light into graphics using optical effects:
    liquid-crystal device (LCD).

Flat Panel Displays

- Plasma-panel display:

  a mixture of gases(electrically charged ionized gases) between two plates
  vertical conducting ribbons are placed in one plate, and horizontal conducting ribbons are placed in the other plate
  voltage is applied to the two ribbons to transform gas into glowing plasma of electrons and ions.

Flat Panel Displays

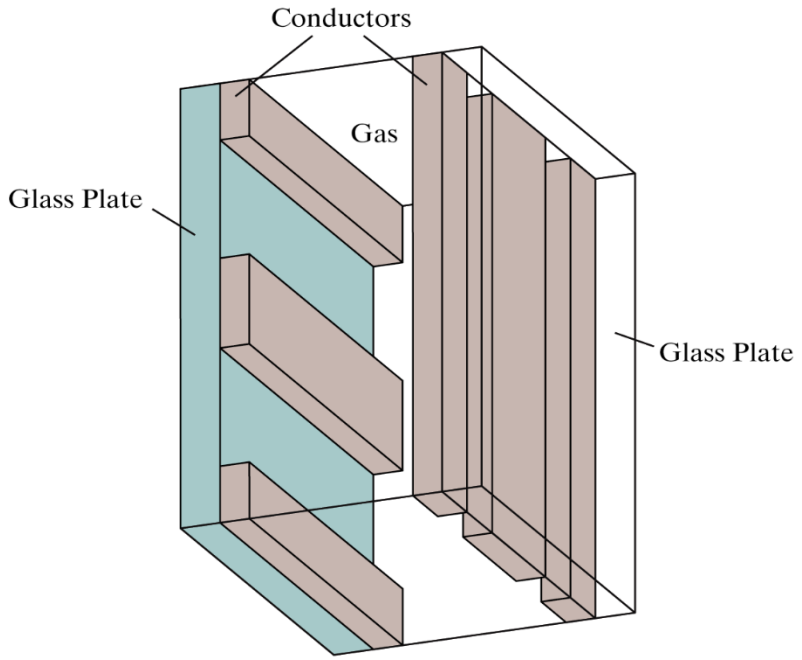


Figure 2-11

Basic design of a plasma-panel display device.

Flat Panel Displays

- Thin-film electroluminescent display:
  similar devices except that the region between the plates is filled with phosphor instead of gas.

  Example: zinc sulfide with manganese
  voltage applied between the plates moves electrons to the manganese atoms that release photons of light.

Flat Panel Displays

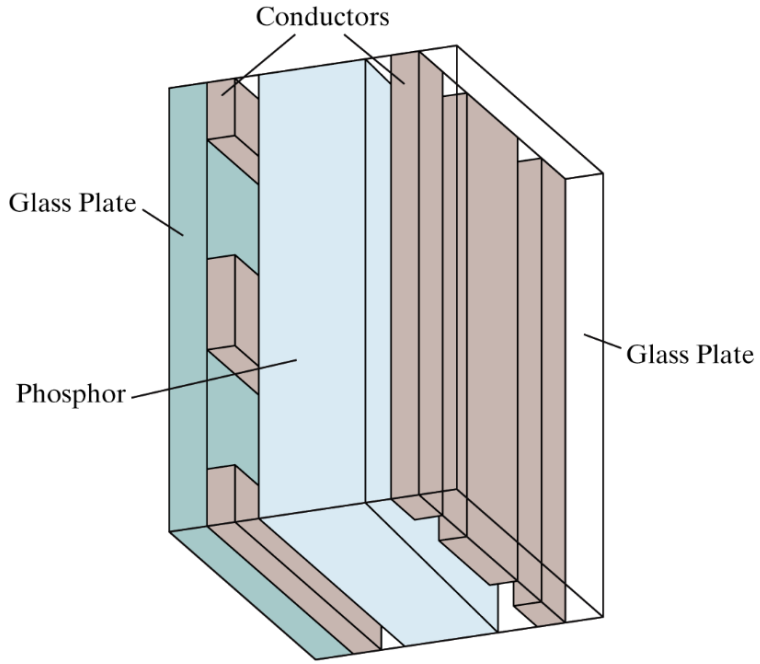


Figure 2-13
Basic design of a thin-film electroluminescent display device.

Flat Panel Displays

- Light-emitting diode:

  a matrix of diodes, one per pixel
  apply voltage stored in the refresh buffer
  convert voltage to produce light in the display.

Flat Panel Displays

- Liquid-crystal displays (LCD):
  LCD screens are often used in small devices such as calculators and laptop monitors.
  non-emissive.
  picture produced by passing light from a light source through liquid-crystal material
  liquid-crystal material can be programmed to either let the light through or not
  liquid-crystal material contains crystals within a liquid
  nematic (thread-like) liquid-crystals have rod shape that can either align to with the light direction or not
  (when voltage is applied to conductors)
  panel made of rows of horizontal, transparent conductors
  apply voltage to two ribbons to make plasma glow
  two polarizers ,two conductors, reflector

Flat Panel Displays

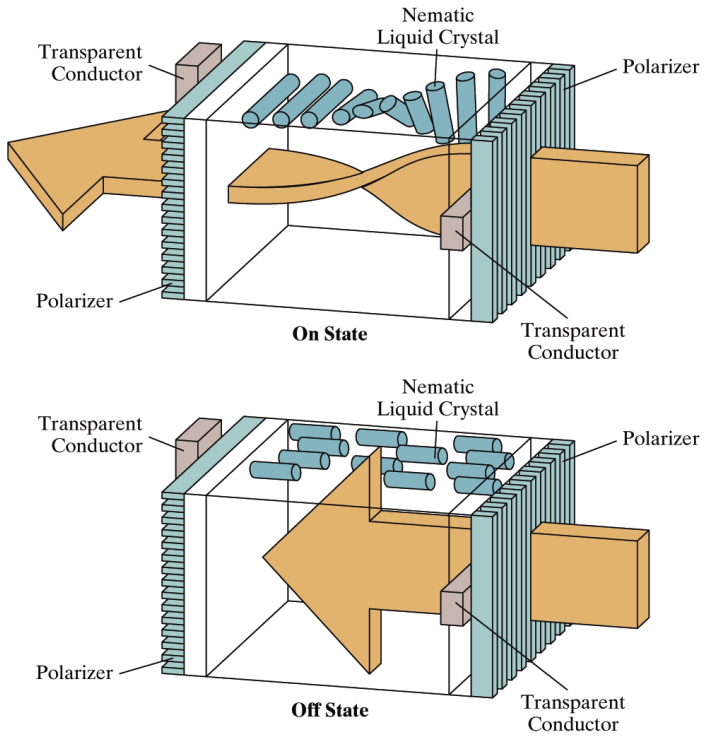


Figure2-15

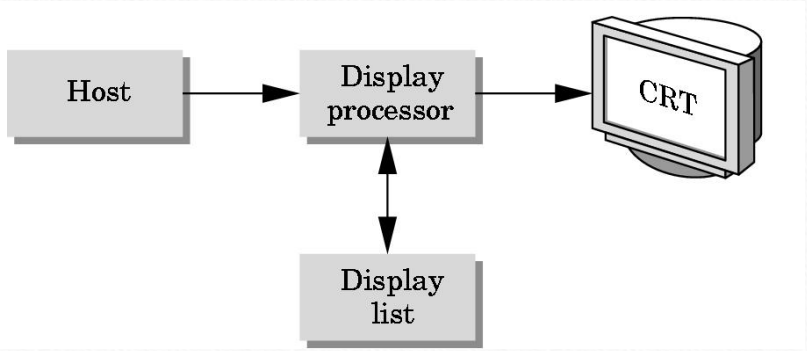
The light-twisting, shutter effect used in the design of most liquid-crystal display devices.

Flat Panel Displays

- Liquid-crystal displays (LCD)
  - Passive matrix LCD
    refresh buffer
    screen refreshed at 60 frames per second
  - Active matrix LCD
    transistor stored at each pixel
    prevents charge from leaking out of liquid-crystals

Display Processor

- Rather than have the host computer try to refresh display use a special purpose computer called a *display processor* (DPU)



- Graphics stored in display list (display file) on display processor
- Host *compiles* display list and sends to DPU

Graphics Workstations

- Graphics monitors use raster-scan displays (CRT or flat-panel monitors).
- Graphics workstations provide more powerful graphics capability:
  - Screen resolution 1280 x 1024 to 1600 x 1200.
  - Screen diagonal > 18 inches.
- Specialized workstations (medical imaging, CAM):
  - Up to 2560 x 2048.
  - Full-color.
- 360 degrees panel viewing systems.

Three-Dimensional Viewing Devices

- For the display of 3D scenes.

- Often using a vibrating, flexible mirror.

- Scan alternate images in alternate frames.

- Multiple stereo images (time multiplexing).

Stereoscopic and Virtual-Reality Systems

- Another technique for the display of 3D scenes.

- Not true 3D images, but provides a 3D effect.

- Uses two views of a scene along the lines of right and left eye. Gives perception of a scene depth when right view is seen from right eye and left scene is seen from left eye (stereoscopic effect). Display each view at alternate refresh cycles.

Stereoscopic and Virtual-Reality Systems

- Stereoscopic systems are used in virtual reality systems:
  - Augmented reality
  - Immersive reality
    - Headset generates stereoscopic views
    - Input devices (gloves, helmet, …) capture motion
    - Sensing system in headset tracks user's position
- Scene projected on an arrangement of walls

Input Devices

- Input devices
  - Keyboards, button boxes, dials
  - Mouse devices
  - Trackballs and space balls
  - Joysticks
  - Data gloves
  - Digitizers
  - Image scanners
  - Touch panels
  - Light pens
  - Voice systems

Input Devices

- Keyboards, button boxes, dials
  - Standard keyboard
    - Alphanumeric
    - Function keys
  - Button box
    set of input dials

Input Devices

- Mouse devices
  - Mechanical mouse
    - One-button
    - Rotating ball
    - Two perpendicular shafts to capture rotation
  - Optical mouse
    - Optical sensor
    - Laser
    - Grid to detect movement
  - Added widgets
    - Buttons
    - Trackball
    - Thumbwheel.

Input Devices

- Trackball
  - A ball device that can be rotated with the fingers or palm of hand
- Space ball
  - Six degrees of freedom
  - Does not move, detects strain placed on the ball by trying to move it.

Input Devices

- Joystick
  - A small, vertical lever mounted on a base
  - Movable joystick measures motion
  - Stationary (isometric) joystick measures strain.
- Data glove
  - Used to grasp a virtual object
  - Measures hand and finger position
  - 2D or 3D
  - Can also be used as input device to detect surface.

Input Devices

- Digitizers
  - Used for drawing, painting, or selecting positions
  - Graphics tablet used to input 2D coordinates by activating a hand cursor or stylus at given positions on a flat surface
  - Used to trace contours, select precise coordinate positions
    - Hand held cursor
    - Stylus
  - Electromagnetic
    - Grid of wires
    - Electromagnetic pulses send an electrical signal in stylus or cursor
  - Acoustic
    - Sound waves to detect stylus position by microphones
    - Can be 3D.

Input Devices

- Image scanners
  - Used to store images on a computer
  - Hand held
  - Flatbed
  - Drum.

Input Devices

- Touch panels
  - Select objects by the touch of a finger
  - Optical
    - Line of infrared light-emitting diodes (LED) along vertical and horizontal edges
    - Interrupted when panel is touched
  - Electrical
    - Two transparent plates of material, one conducting, the other resistive
    - Touch brings the plates to be in contact with one another, causing a voltage drop
    - Measure the voltage drop
  - Acoustical.

Input Devices

- Light pens
  - Pen-shaped device to select screen positions by detecting lights coming from points on the CRT screen
  - Used to capture position of an object or select menu options.

Input Devices

- Voice systems
  - Speech recognition systems to recognize voice commands
  - Used to activate menu options or to enter data
  - Uses a dictionary from a particular user (learning system).

Coordinate System

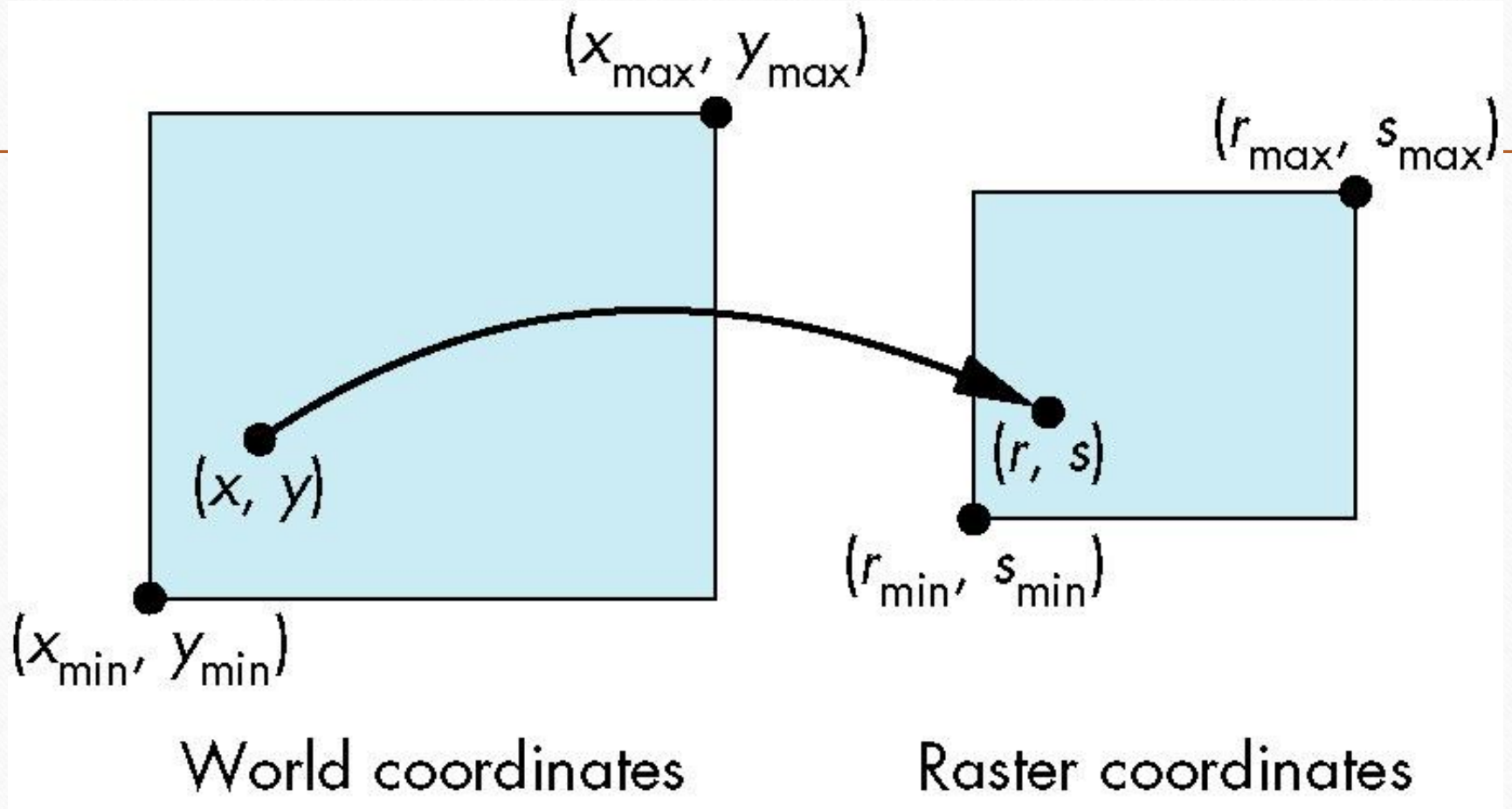
- Early systems depended on specific device mapping
- Device-independent graphics broke link

- Use application or problem coordinate system to define image
- Use device coordinates, raster coordinates, screen coordinates for device

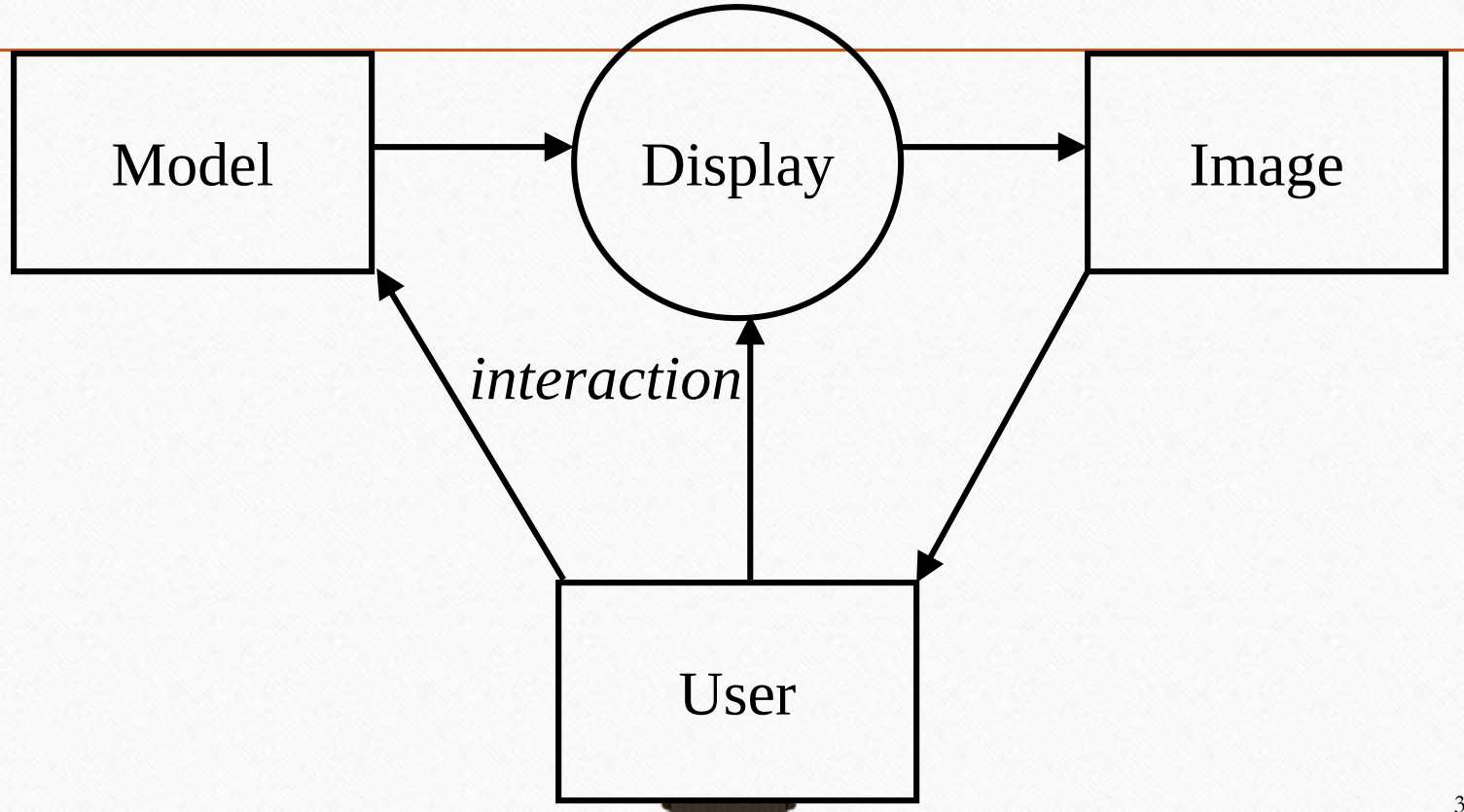
Coordinates

- Application coordinates can be integer or real and multi-dimensional
- Screen or raster coordinates are always integer and essentially 2-dimensional
- Graphics program maps application coordinates onto device coordinates

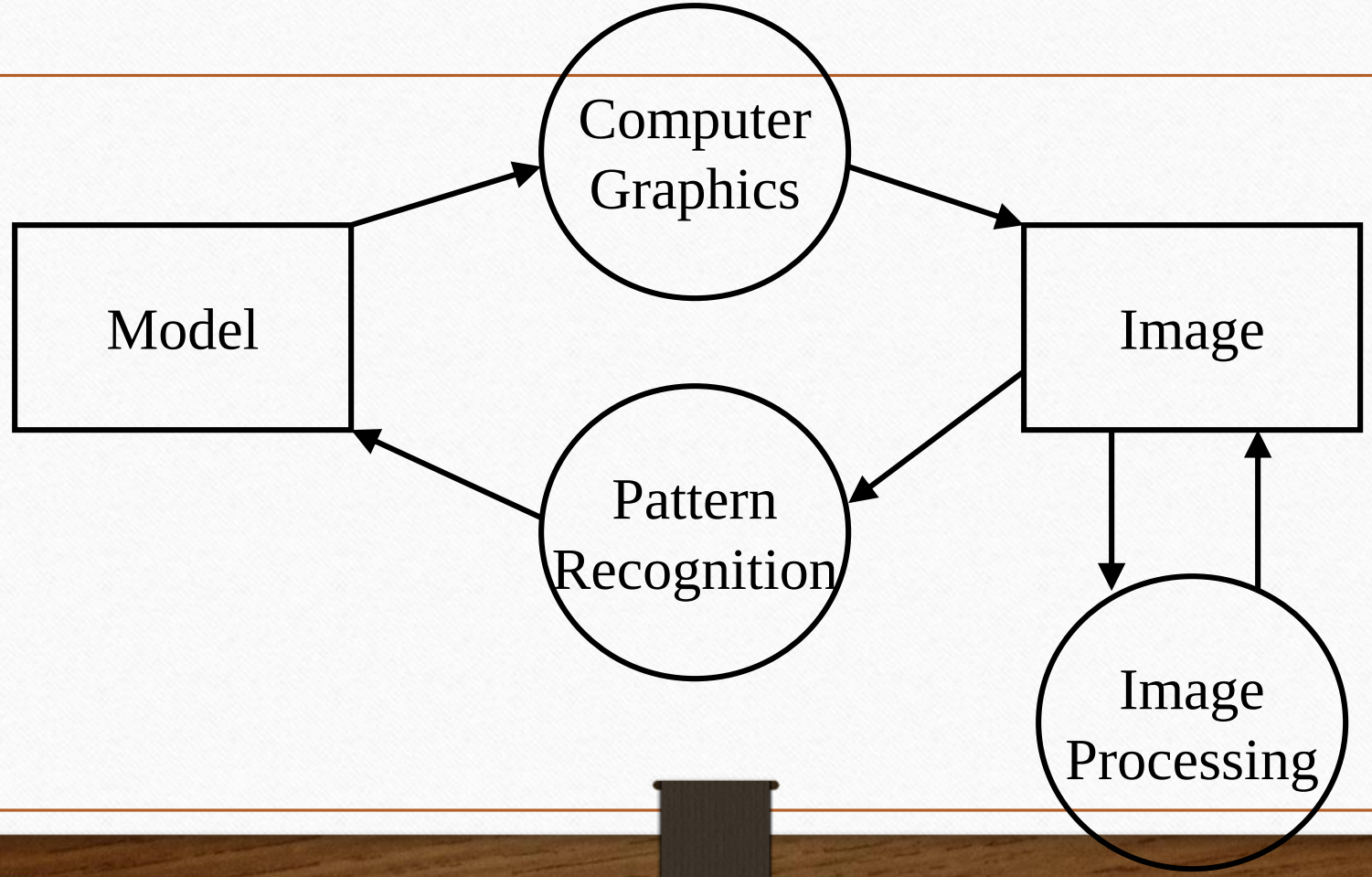
App to Device Mapping



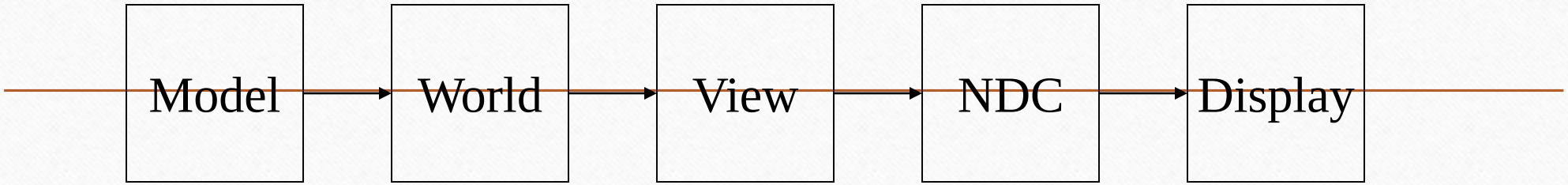
Schematic



Also…


Computer Graphics
Model
Image
Pattern Recognition
Image Processing

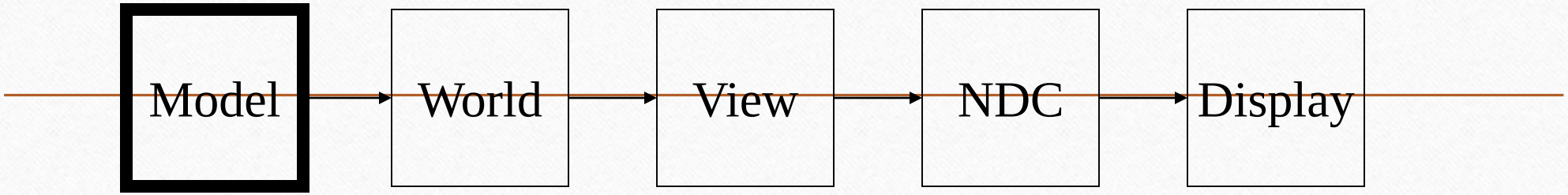
From model to image

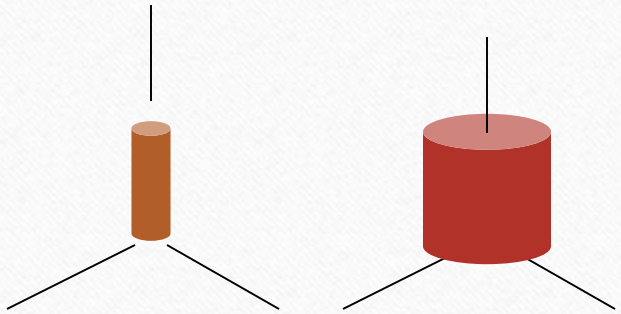


Graphics pipeline

Coordinates and transformations

From model to image





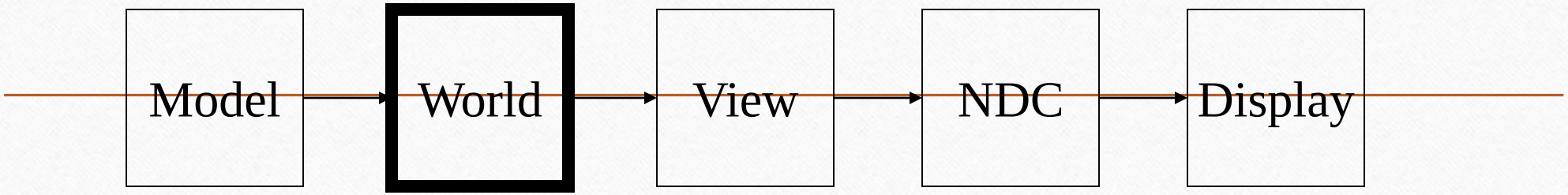
Cylinder:

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

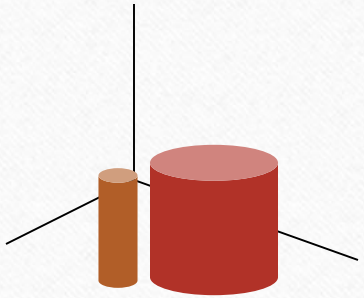
$$0 \le z \le h$$

Local or modeling coordinates

From model to image

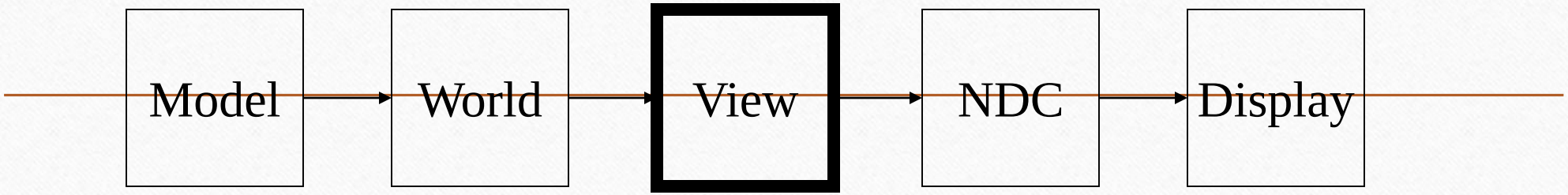


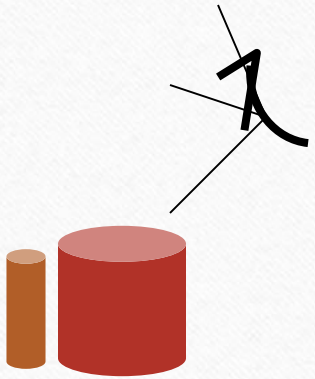
Position cylinders in scene:



World coordinates

From model to image



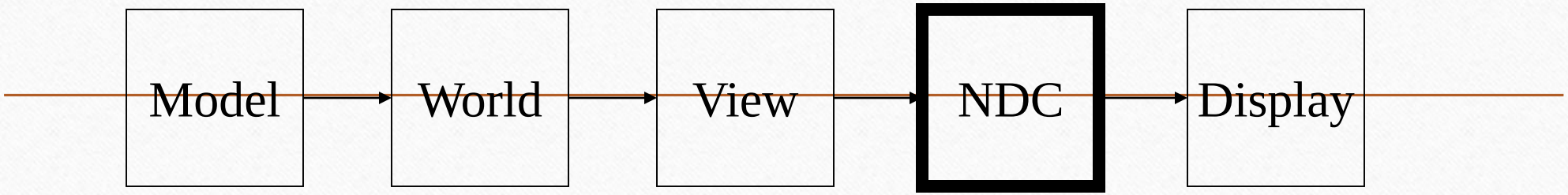


Look at cylinders,
project on virtual screen

Viewing coordinates

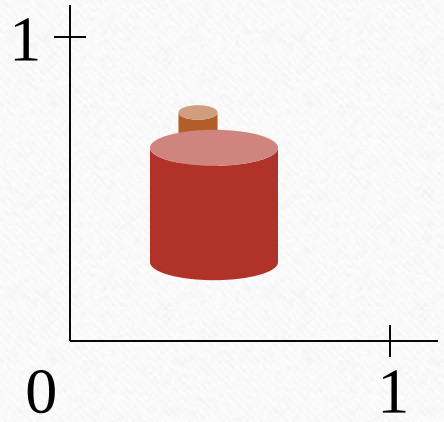
Visible surfaces, shading

From model to image

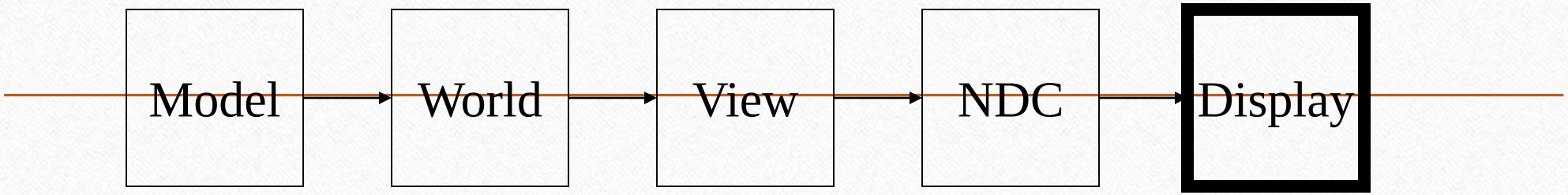


Display:

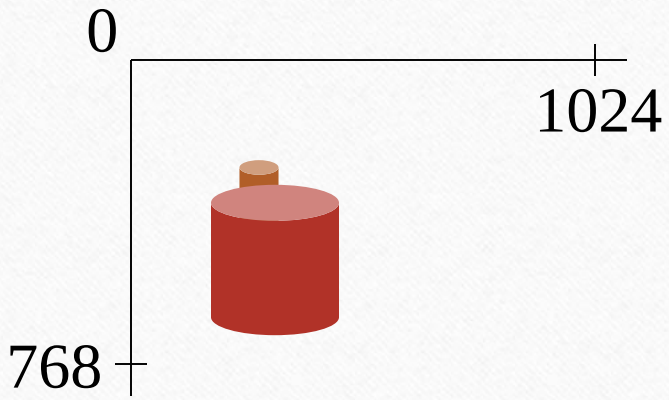
Normalized Device Coordinates



From model to image



Display on screen:



Device Coordinates

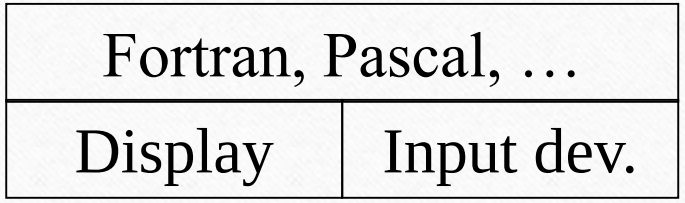
Interaction

Generating graphics

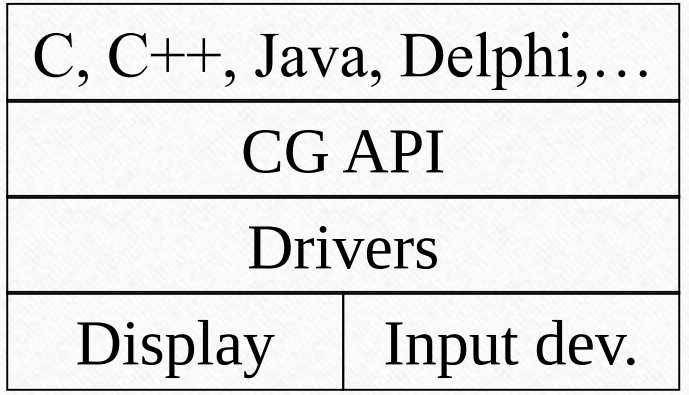
- Special-purpose programs
  - Photoshop, Powerpoint, AutoCAD, StudioMax, Maya, Blender, PovRay, …
- General graphics libraries and standards
  - Windows API, OpenGL, Direct3D,…

CG standards

- Set of graphics functions, to be called from programming language
- Access to and abstract from hardware
- Standardization



1975

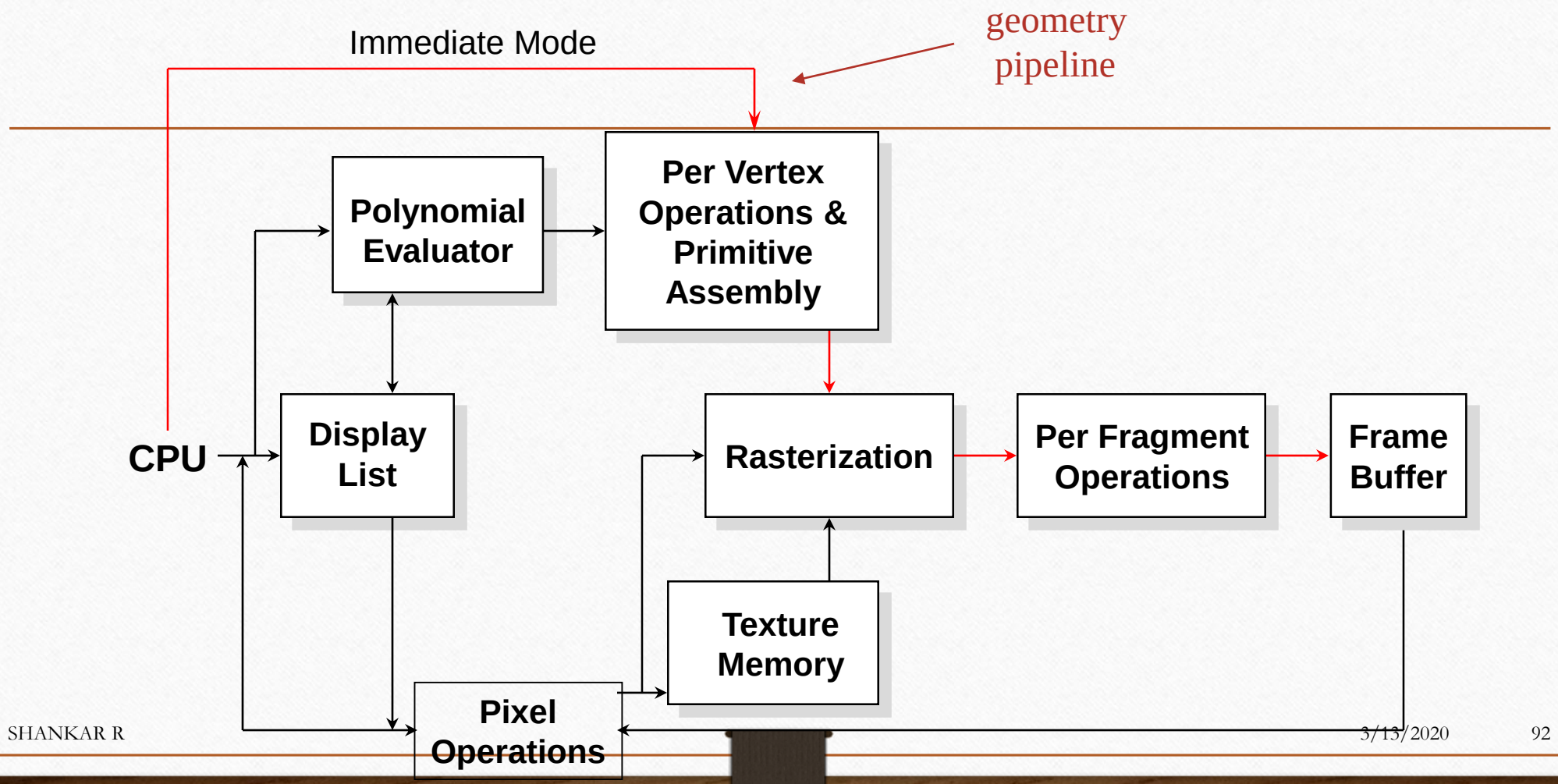


2000

OpenGL

- 3D (and 2D)
- Fast
- Hardware, language, OS, company independent
- OpenGL architecture review board
- Broad support
- Low-level (right level!)
- Standard graphics terminology

OpenGL Architecture



OpenGL, GLU and GLUT

- OpenGL: basic functions
- GLU: OpenGL Utility library:
- GLUT: OpenGL Utility Toolkit library
- GLU and GLUT:
  - Handy functions for viewing and geometry

OpenGL and Java

- C: `glFunction(); gluFunction(); glutFunction();`
- Java: JOGL

```
gl.glFunction();
glu.gluFunction();
glut.glutFunction();
```

- No windowing functions offered by JOGL

OpenGL syntax

- Functions:

  `glFunction: glBegin, glClear, glVertex, …`

- Constants:

  `GL_CONSTANT: GL_2D, GL_LINE`

- Datatypes:

  `GLtype: GLbyte, GLint, GLfloat`

OpenGL State

- OpenGL is a state machine
- OpenGL functions are of two types
  - Primitive generating
    - Can cause output if primitive is visible
    - How vertices are processed and appearance of primitive are controlled by the state
  - State changing
    - Transformation functions
    - Attribute functions

Functions

- Graphics Output Primitives
  - Line, polygon, sphere, …
- Attributes
  - Color, line width, texture, …
- Geometric transformations
  - Modeling, Viewing
- Shading and illumination
- Input functions

Classes for OpenGL Functions

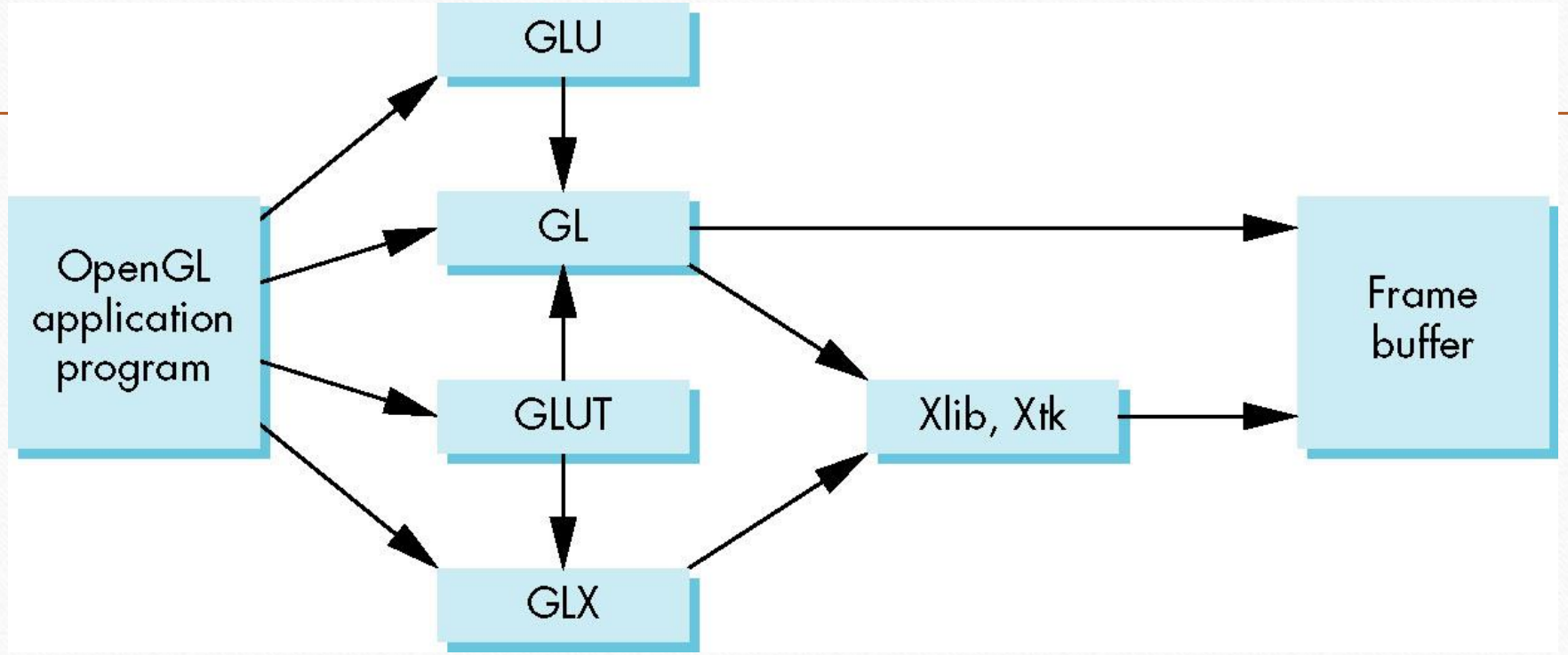
1. Primitives – draw points, line segments, polygons, text, curves, surfaces
2. Attributes – specify display characteristics of objects: color, fill, line width, font
3. Viewing – determine aspects of view: position and angle of camera, view port size, …
4. Transformations – Change Appearance Or Characteristics Of Objects: Rotate, Scale, Translate
5. Input – Handle Keyboard, Mouse, Etc.
6. Control – Communicate With Window System
7. Query – Get Display Information: Size, Raster Value, …

OpenGL Interface

- Graphics Utility Interface (GLU)
  - Creates common objects like spheres
- GL Utility Toolkit (GLUT)
  - Provides generic interface to window system
- GLX for Unix/Linux and

  wgl for Microsoft Windows
  - provide low-level glue to window system

Library Organization



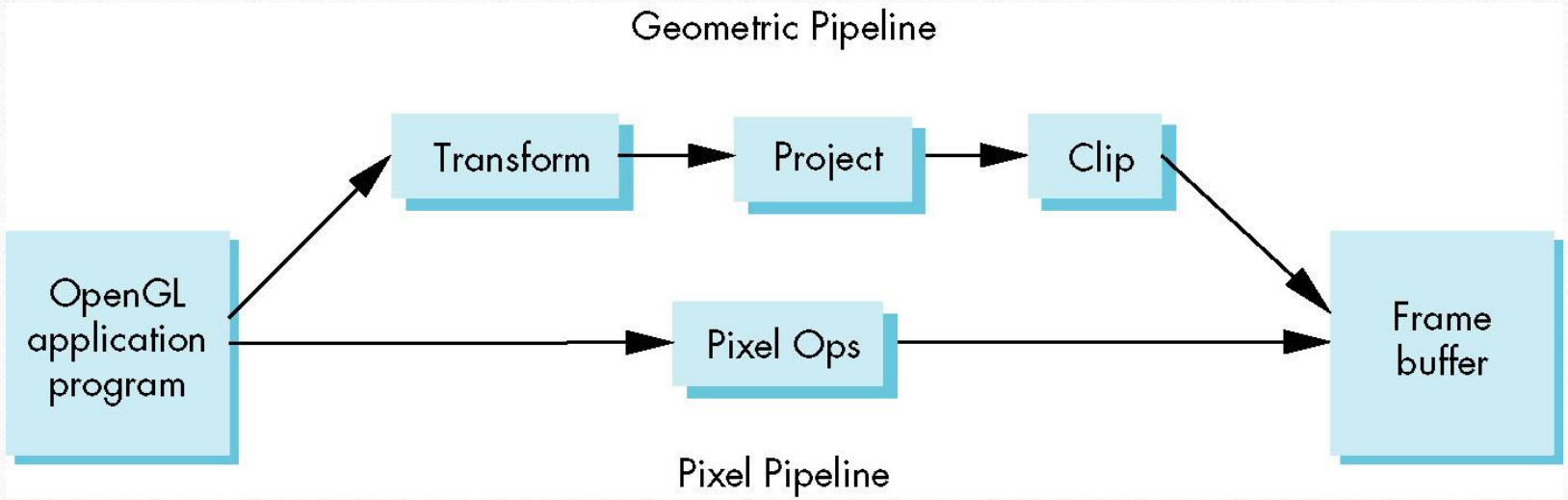
Using Libraries

- Header files
  - #include <GL/glut.h>

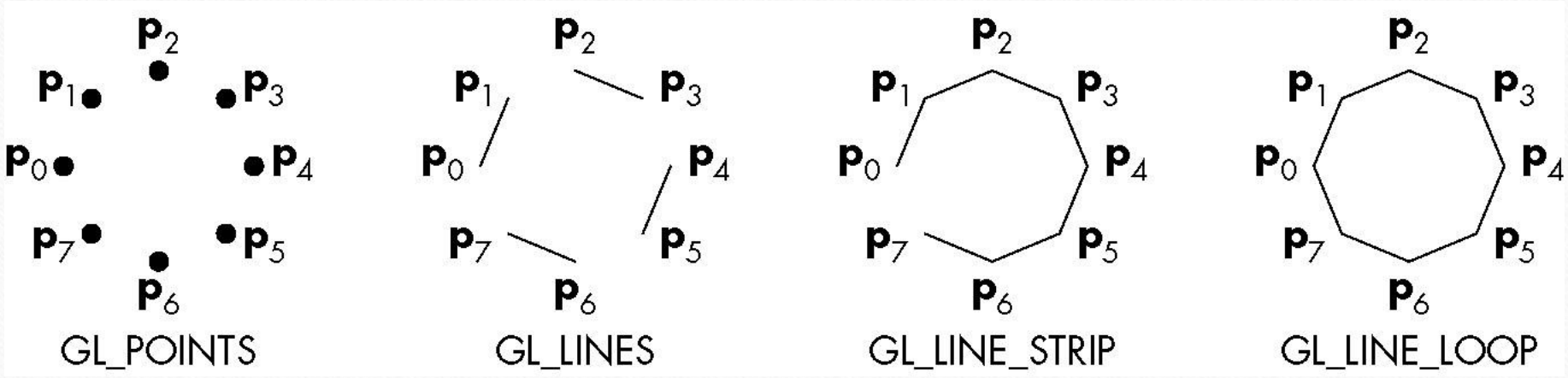
  - #include <GL/gl.h>
  - #include <GL/glu.h>
- On some systems the GL/ is not used

Primitives

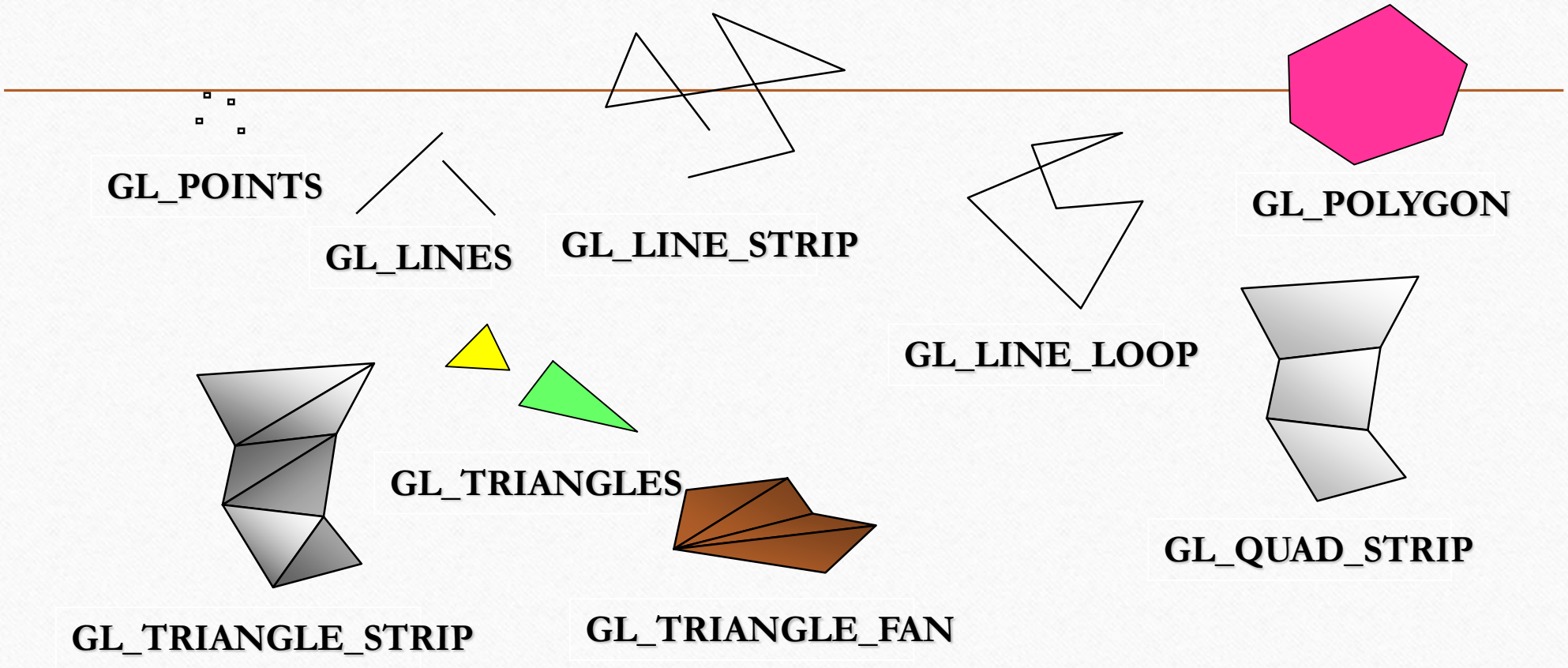
- OpenGL supports both geometric primitives and raster primitives



Primitive Examples

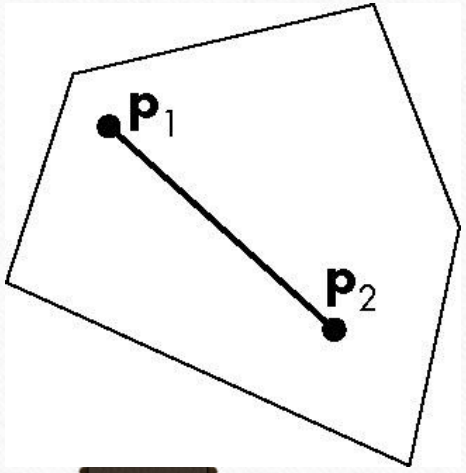


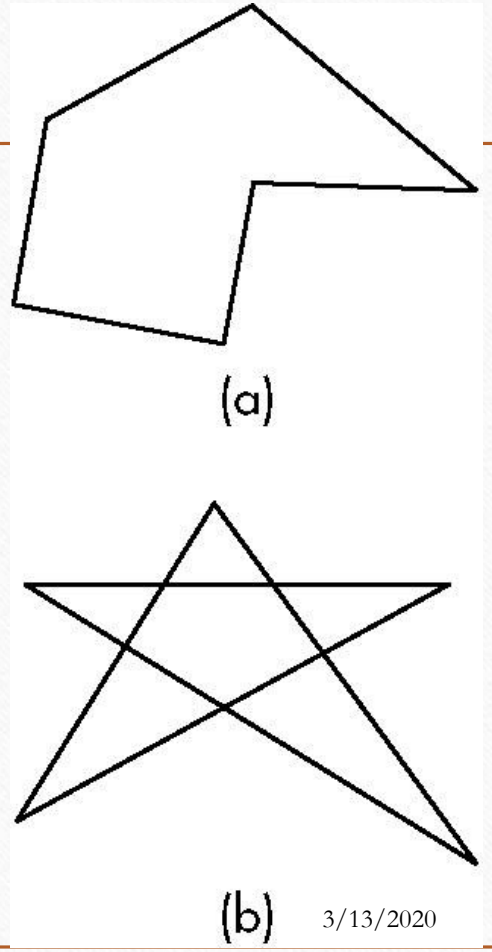
OpenGL Primitives



Properties of Polygons

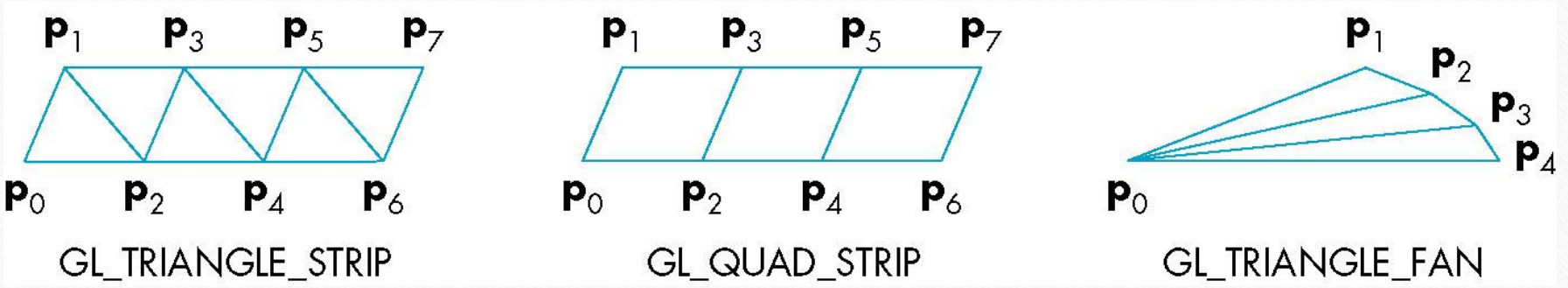
- Defined by line loop border
- Simple if no edges cross
- Convex if every line segment connecting pair of points on boundary or inside lies completely inside



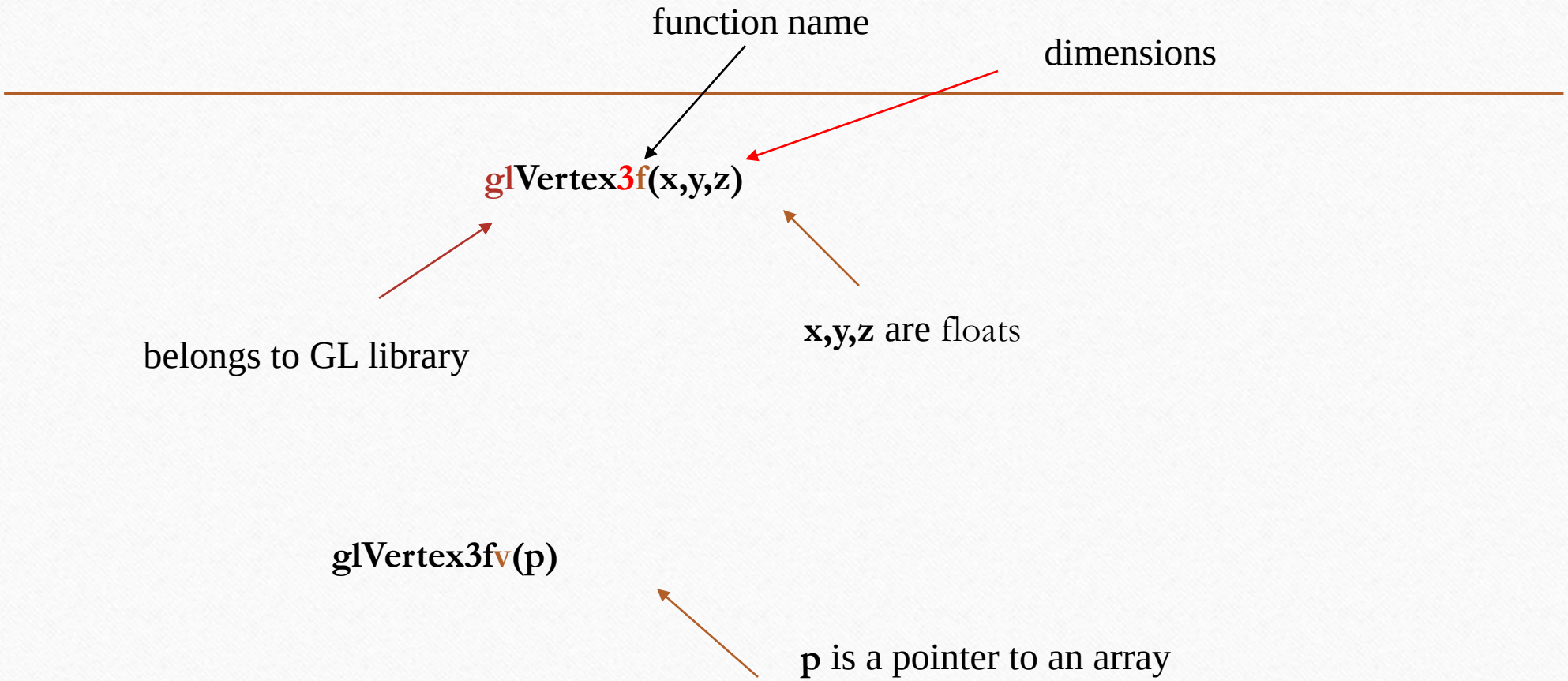


Special Polygons

- GL_TRIANGLE_STRIPE, GL_QUAD_STRIPE, GL_TRIANGEL_FAN
  - Contiguous stripe or fan of triangles or quadrilaterals



OpenGL function format



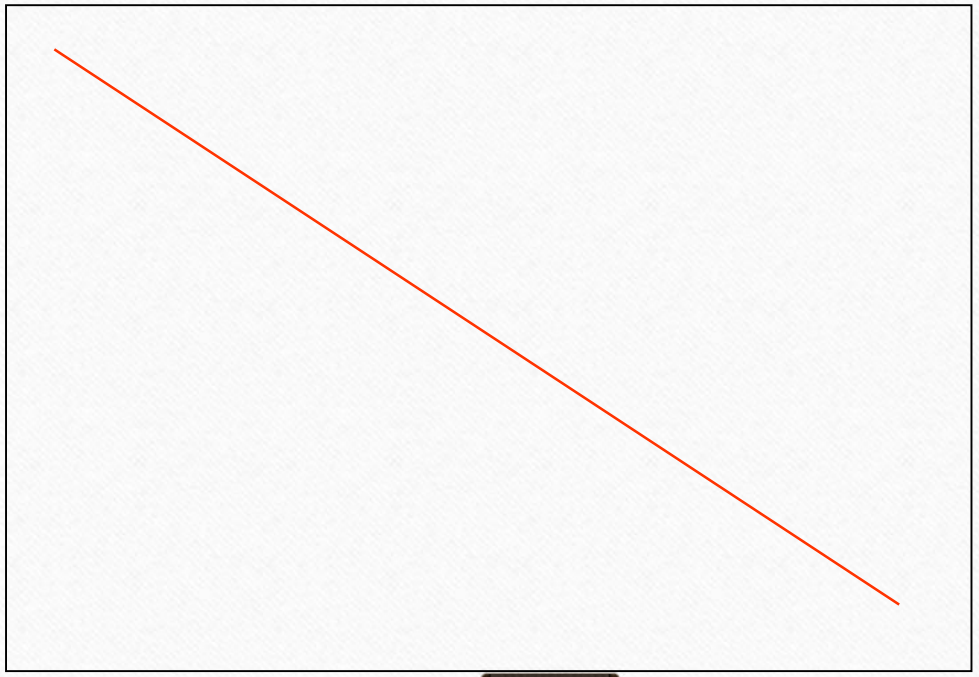
Example

```
glClearColor(1.0,1.0,1.0,0.0);// Background color
glMatrixMode(GL_PROJECTION);  // Set transformation
glLoadIdentity;
gluOrtho2D(0, 200, 0, 150);

glClear(GL_COLOR_BUFFER_BIT); // Clear background

glColor3f(1.0, 0.0, 0.0);     // Set color to red
glBegin(GL_LINES);            // Draw line
   glVertex2i(180, 15);       // - first point
   glVertex2i(10, 145);       // - second point
glEnd;                        // Ready with line
glFlush;                      // Send
```

Output



Example 3D

Aim: Draw two rectangular boxes

1. Set up viewing transformation
2. Specify the colors
3. Draw the objects

Example 3D

```
// Set up viewing transformation

glViewport(0, 0, 500, 500);   // Select part of window

glMatrixMode(GL_PROJECTION);  // Set projection
glLoadIdentity();
glFrustum(-1.0, 1.0, -1.0, 1.0, 4.0, 20.0);

glMatrixMode(GL_MODELVIEW);   // Set camera
glLoadIdentity();
gluLookAt(3.0, 6.0, 5.0,          - eye point
          1.0, 0.0, 0.0,          - center point
          0.0, 0.0, 1.0);         - up axis
```

Example 3D

```
// Clear background
```

```
glClearColor(1.0,1.0,1.0,0.0);// Background color
glClear(GL_COLOR_BUFFER_BIT); // Clear background

// Set color

glColor3f(0.0, 0.0, 0.0);     // Set color to black
```

Example 3D

```
// Draw two rectangular boxes

glutWireCube(1.0); // unit box around origin

glTranslatef(2.0, 0.0, 0.0);  // move in x-direction
glRotatef(30, 0.0, 0.0, 1.0); // rotate 30 degrees
                                 around z-axis
glScalef(1.0, 1.0, 2.0);      // scale in z-direction

glutWireCube(1.0); // translated, rotated, scaled box
```

Example 3D

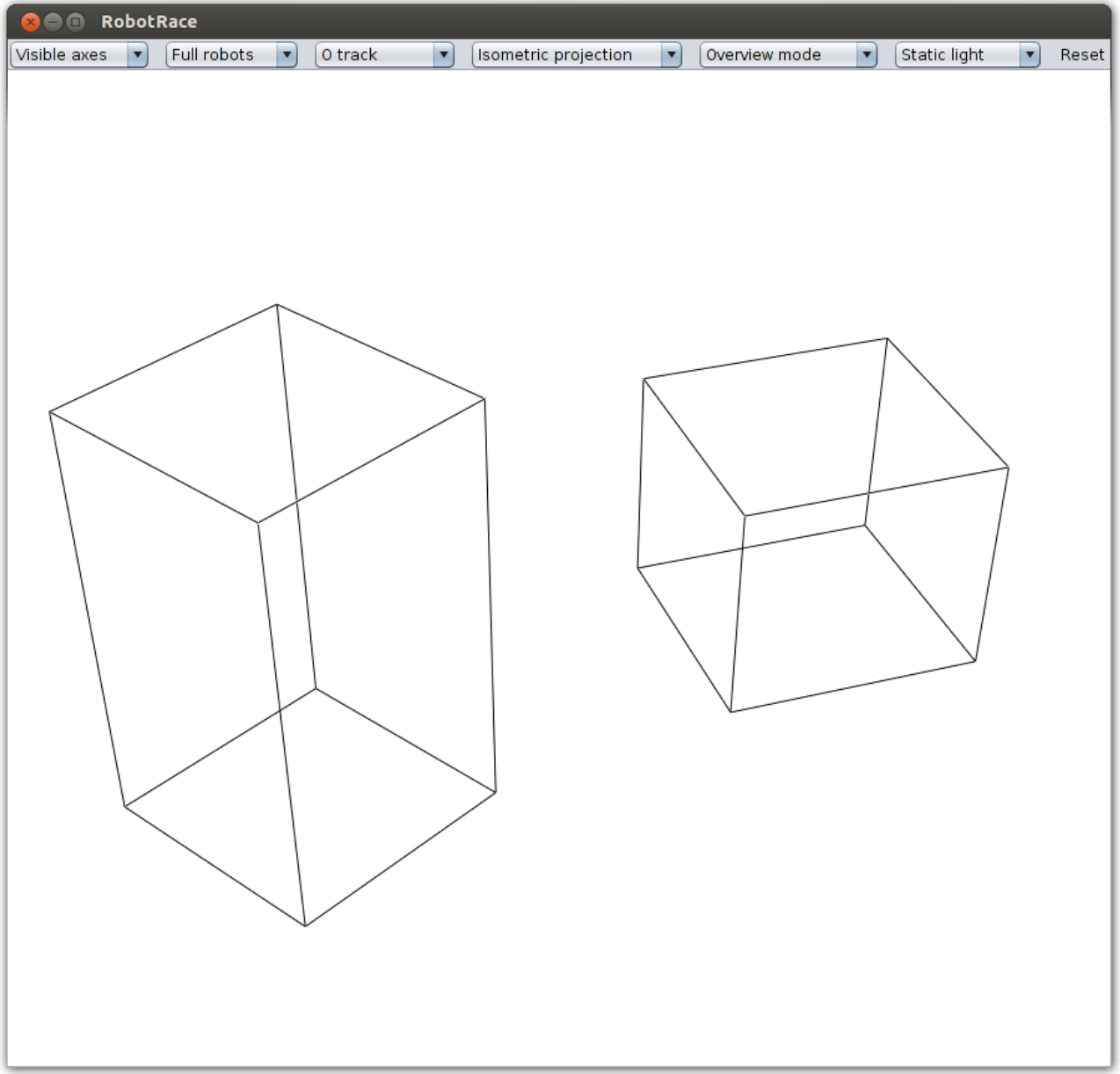
```
glutWireCube(1.0); // unit box around origin
glTranslatef(3.0, 0.0, 0.0);  // move in x-direction
```

```
glRotatef(30, 0.0, 0.0, 1.0); // rotate 30 degrees
                              around z-axis
glScalef(1.0, 1.0, 2.0);      // scale in z-direction


glutWireCube(1.0); // translated, rotated, scaled box
```

Note:

- Objects are drawn in the current local axis-frame;
- With transformations this frame can be changed.


RobotRace
Visible axes
Full robots
0 track
Isometric projection
Overview mode
Static light
Reset

MODULE-1 4-2-19

codeblocks with MinGW setup

1. CG Basics
2. OpenGL Basics
3. Three algorithms - DDA (Digital Differential Analyser), Bresenham's Line or midpoint line, Bressenham's circle or Midpoint circle

6.2.19

* Explain Raster-scan systems & Random-scan Displays.
* Explain CRT systems
Explain basic primitives with an example (points, lines, linestrip, lineloop)

Open GL Basics

gl.h - graphics library ; Primitive fncs how to draw line, circle

glu.h - graphics library utility ; contains fncs of gl.h & windowing fncs.

glut.h - graphics library utility Toolkit ; gl.h + glu.h + other properties

How to draw a point, line, linestrip, lineloop, polygon etc

First OpenGL program.

* Point

```
#include <GL/glut.h>
void display()
{   glPointSize(10);
    glBegin(GL_POINTS);
    glVertex2f(-0.5,-0.5);
    glVertex2f(0.5,-0.5);
    glVertex2f(0.5,0.5);
    glVertex2f(-0.5,-0.5);
    glEnd();
    glFlush();
}
void main(int argc, char **argv)
{   glutInit(&argc,argv);
    glutCreateWindow("Points Demo");
    glutDisplayFunc(display);
    glutMainLoop();
}
```

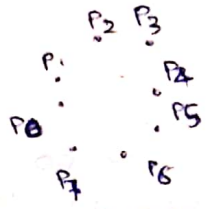
initializing openGL/graphics system so that system understands our graphics applications

OpenGL will create a window with Title 'Points Demo'

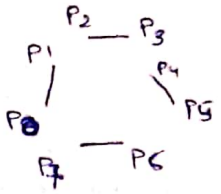
It is a call back function which renders pixels to be drawn on to screen. We need to register a function which actually draws/executes as a parameter to this glutDisplayFunc callback function.

→ Run the output forever

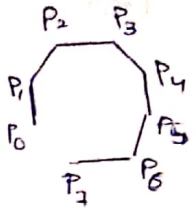
* Primitives



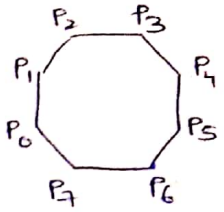
Points



Lines



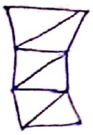
Linesstrip



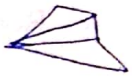
Lineloop

GL_Triangles

Triangle Strip



Triangle fan



GL_Polygon

Polygon with colour

GL quad

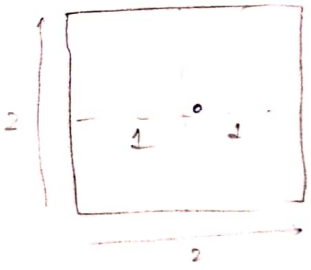
GL QuadStrip

→ In openGL programs default bgcolour is black
→ pixel colour is white by default
→ Entry point for any prog is int main()

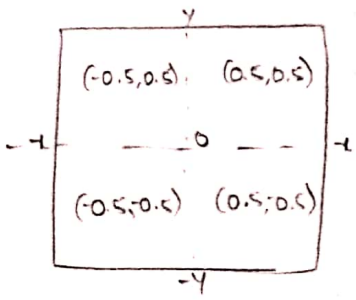
18-2-19

1st Program

→ canvas/window size 2×2 by default



Aim is to draw



* Applying custom background colour

1) glClearColor (0,0,1,1) // setting color
R G B A - Parameter
or
transparency parameter
It can be 1 or 0

2) glClear (Maskedbits); // applying color
↓
constant

In openGL, uppercase : constant

```
void display()
{
    glClearColor(0,0,1,1);
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POINTS);
    =
}
```

} commenting either of the 2 stmts will leave window with black background only.

O/P Blue bg color
white pixel color

* Applying custom font color

```
glColor3f(1,0,1);
```

```
void display()
{
    glClearColor(0,0,1,1);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor(1,1,0);
    =
}
```

O/P Blue Bg color
Yellow pixel color

LINES

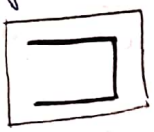
```
glBegin(GL_LINES);
glLineWidth(10); //thickness
```

pg1 O/P

LINE STRIP

```
glBegin(GL_LINE_STRIP);
```

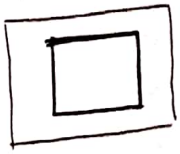
O/P



LINE LOOP

```
glBegin(GL_LINE_LOOP);
```

O/P



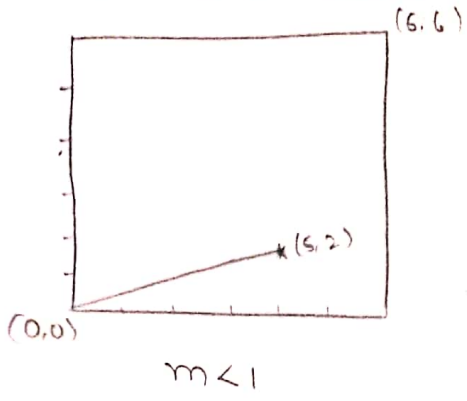
Algorithms

1) DDA
2) Bresenham's line
3) Bresenham's circle

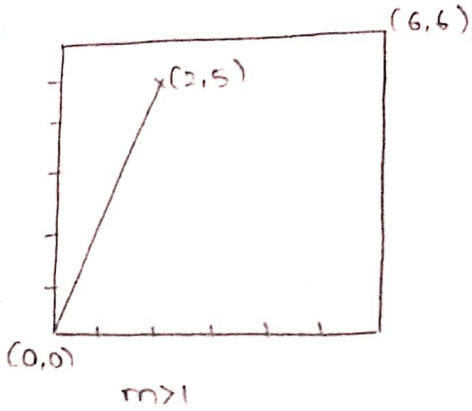
17 DDA Line Digital Differential Analyser

* screen is full of pixels
* aim : to determine intermediate pixels blw start & end point

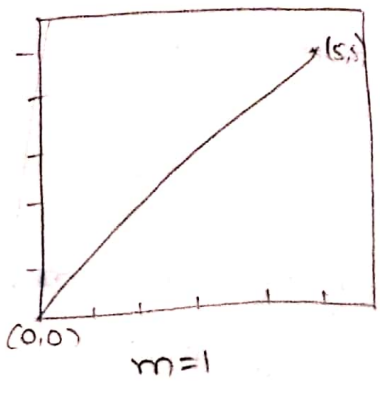
we have 3 cases



$m < 1$



$m > 1$



$m = 1$

we know, Line eq is $y = mx + c$ where $m = \frac{y_2 - y_1}{x_2 - x_1}$ — (1)

for any two points (x_1, y_1) & (x_2, y_2)

In general, let us assume

current pixel = (x_k, y_k)

$\therefore$ Next pixel = (x_{k+1}, y_{k+1})

$$m = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} \quad -(2)$$

we have 3 cases,

case 1 : $m < 1$

x always gets changed unit wise.

$y = ?$

$$\boxed{x_{k+1} = x_k + 1}$$

$y_{k+1} = ?$

(2) $\Rightarrow$ $m = \frac{y_{k+1} - y_k}{1}$

$\therefore$ $\boxed{y_{k+1} = y_k + m}$

case 2 : $m > 1$

y always gets changed unit wise

$x = ?$

$$\boxed{y_{k+1} = y_k + 1}$$

$x_{k+1} = ?$

(2) $\Rightarrow$ $m = \frac{1}{x_{k+1} - x_k}$

$\therefore$ $\boxed{x_{k+1} = x_k + \frac{1}{m}}$

case 3: m=1

y changes unit wise

x changes unit wise

$$x_{k+1} = x_k + 1$$
$$y_{k+1} = y_k + 1$$

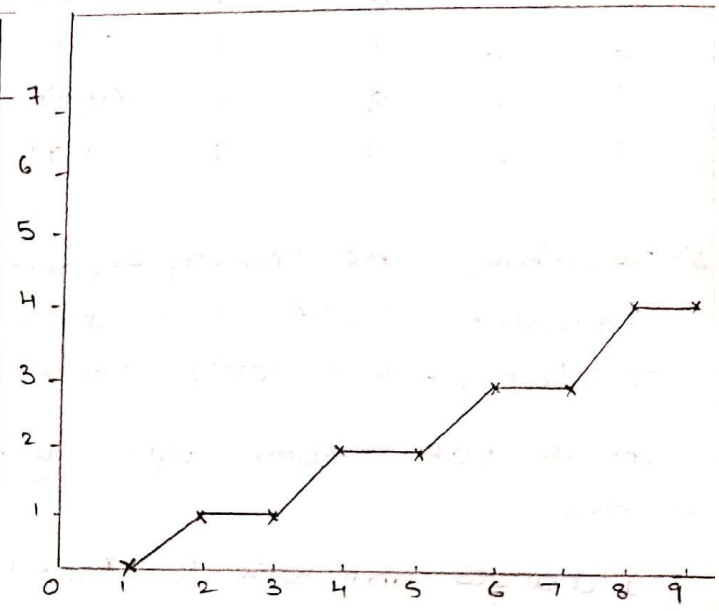
Example 1 : Draw a line between (1,0) to (9,4) using DDA.

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{4-0}{9-1} = \frac{1}{2} < 1$$

since $m < 1$,

$$x_{k+1} = x_k + 1$$
$$y_{k+1} = y_k + m$$

x_k	y_k	x_{k+1}	y_{k+1}	round (actual point (x,y))
1	0	1	0	(1,0)
1	0	2	0.5	(2,1)
2	0.5	3	1	(3,1)
3	1	4	1.5	(4,2)
4	1.5	5	2	(5,2)
5	2	6	2.5	(6,3)
6	2.5	7	3	(7,3)
7	3	8	3.5	(8,4)
8	3.5	9	4	(9,4)

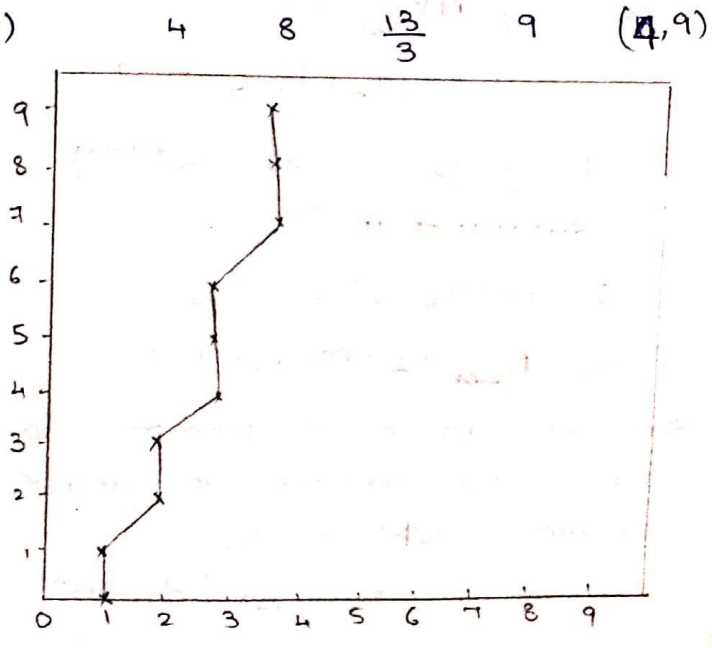


2: (1,0) to (4,9)

$m = 9/3 = 3 > 1$; $x_{k+1} = x_k + \frac{1}{m}$

$y_{k+1} = y_k + 1$

x_k	y_k	x_{k+1}	y_{k+1}	round(x,y)
1	0	1	0	(1,0)
1	0	4/3	1	(1,1)
4/3	1	5/3	2	(2,2)
5/3	2	2	3	(2,3)
2	3	8/3	4	(3,4)
8/3	4	3	5	(3,5)
3	5	10/3	6	(3,6)
10/3	6	11/3	7	(4,7)
11/3	7	4	8	(4,8)

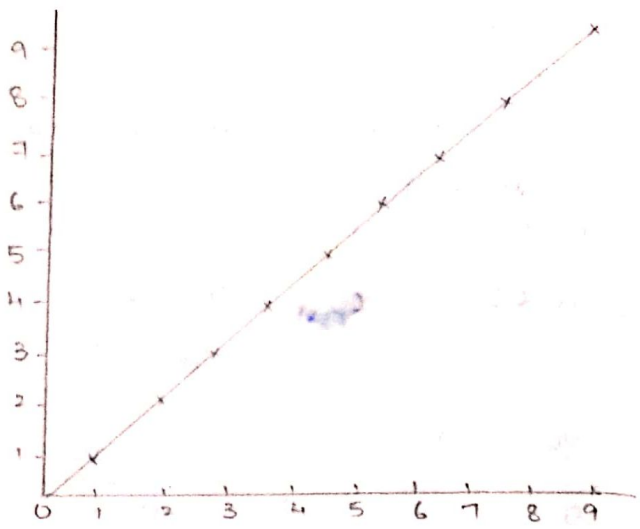


3) (1,1) to (9,9)

$m = 1$ $\quad x_{k+1} = x_k + 1$

$y_{k+1} = y_k + 1$

x_k	y_k	x_{k+1}	y_{k+1}	(x, y)
1	1	1	1	(1,1)
1	1	2	2	(2,2)
2	2	3	3	(3,3)
3	3	4	4	(4,4)
4	4	5	5	(5,5)
5	5	6	6	(6,6)
6	6	7	7	(7,7)
7	7	8	8	(8,8)
8	8	9	9	(9,9)



22/9/18.

2) Bresenham's line Drawing Algorithm (Midpoint Line Algorithm)

Drawback of DDA is it involves rounding operation & floating pt operation which is costly. Hence we have Bresenham's line equation.

Let us take same eqn $y = mx + c$; 3 cases $m<1, m>1$ & $m=1$

i) $m < 1$

x changes unit wise, so $x_{k+1} = x_k + 1$ (no dilema)

$y_{k+1} = ?$ (there is a confusion between to choose y_{k+1} or y_k)

WKT $y = mx + c$, $m = \frac{\Delta y}{\Delta x}$

since it is $m<1$ case, we know $x_{k+1} = x_k + 1$

so., $y = m x_{k+1} + c$

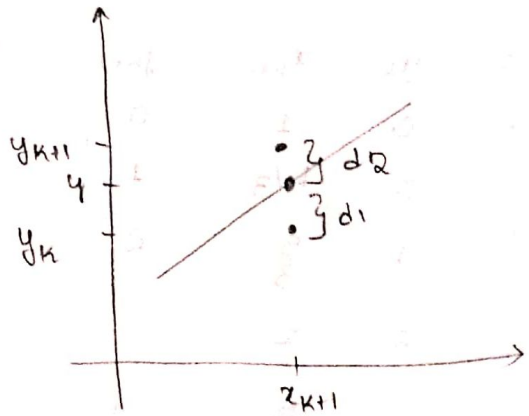
$y = m(x_k + 1) + c$ - ①

$d_1 = y - y_k$ $\quad d_2 = y_k + 1 - y$

substitute in ①

$d_1 = m(x_k + 1) + c - y_k$

$d_2 = y_k + 1 - m(x_k + 1) - c$



Since we are in a dilemma to choose blw $y_k + 1$ or y_k, let us (consider) calculate decision parameter p_k which helps us in deciding y_{k+1} or y_k.

$p_k = \Delta x (d_1 - d_2)$ - ②

Let us calculate $(d_1 - d_2)$

$d_1 - d_2 = m(x_k+1) + c - y_k - y_k + 1 + m(x_k+1) + c$

$= 2m(x_{k+1}) + 2c - 2y_k - 1$

substitute in ②,

$P_k = \Delta x [2m(x_k+1) - 2y_k + 2c - 1]$

$= \Delta x \left[2 \cdot \frac{\Delta y}{\Delta x}(x_k+1) - 2y_k + 2c - 1\right]$

$$\boxed{P_k = 2\Delta y(x_k+1) - 2\Delta x y_k + \Delta x(2c-1)} \quad -③$$

eqn ③ $\Rightarrow$ P_k is initial decision parameter. In order to find the continuous decisions, we have to find the next P_k.

$$P_{k+1} = 2\Delta y(x_{k+1}+1) - 2\Delta x y_{k+1} + \Delta x(2c-1) \quad -④$$

From now the next decision parameter will always be difference b/w P_{k+1} & P_k.

④ − ③ $\Rightarrow P_{k+1} - P_k = 2\Delta y(x_{k+1}+1) - 2\Delta x y_{k+1} + \Delta x(2c-1) - 2\Delta y(x_k+1) + 2\Delta x y_k - \Delta x(2c-1)$

$= 2\Delta y x_{k+1} + 2\Delta y - 2\Delta x y_{k+1} - 2\Delta y x_k - 2\Delta y + 2\Delta x y_k$

$= 2\Delta y(x_k+1) - 2\Delta x y_{k+1} - 2\Delta y(x_k) + 2\Delta x y_k$

$= 2\Delta y - 2\Delta x[y_{k+1} - y_k]$

$$\boxed{P_{k+1} = 2\Delta y - 2\Delta x(y_{k+1} - y_k) + P_k} \quad -⑤$$

The initial point to be plotted is (x_k, y_k)

Let us substitute (x_k, y_k) for (x, y) in initial decision parameter.

Eqn ③ $y = mx + c \Rightarrow c = y - mx$

$P_k = 2\Delta y(x_k+1) - 2\Delta x y_k + \Delta x(2c-1)$

$= 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + \Delta x(2(y - mx) - 1)$

$= 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + \Delta x\left(2\left(y_k - \frac{\Delta y}{\Delta x}x_k\right) - 1\right)$

$= 2\Delta y x_k + 2\Delta y - 2\Delta x y_k + 2\Delta x y_k - 2\Delta y x_k - \Delta x$

$$\boxed{P_k = 2\Delta y - \Delta x} \rightarrow ⑥$$

Eqn ⑥ is is initial decision parameter

(1) $P_K = 2\Delta y - \Delta x$ // apply once initially

(2) $P_{K+1} = P_K + 2\Delta y - 2\Delta x\,(y_{K+1} - y_K)$ // From next iteration

Conclusion: If ($P_K \geq 0$)

{

$x_{K+1} = x_K + 1$

$y_{K+1} = y_K + 1$

}

else

{ $x_{K+1} = x_K + 1$

$y_{K+1} = y_K$

}

eg 1: Draw a line between (1,0), (9,4) using Bresenham's

$m = \frac{4-0}{9-1} = 0.5 < 1.$ $\Delta y = 4$, $\Delta x = 8$, $2\Delta y = 8$, $2\Delta x = 16$

$P_K = 2\Delta y - \Delta x = 8 - 8 = 0$

Let us calculate Initial Decision parameter P_K, since this IDP P_K happens to be zero, see the if loop

x_K	y_K	P_K	x_{K+1}	y_{K+1}	(x_{K+1}, y_{K+1})	
—	—	—	1	0	(1,0)	
1	0	(1): 8-8 0	2	1	(2,1)	
2	1	(2) 0+8-16(1-0) -8	3	1	(3,1)	$y_{K+1} = 1$ $y_K = 0$
3	1	0	4	2	(4,2)	-8+8-16(0)
4	2	-8	5	2	(5,2)	0+8-16(1)
5	2	0	6	3	(6,3)	-8+8-16(0)
6	3	-8	7	3	(7,3)	0+8-16(3-1)
7	3	0	8	4	(8,4)	-8+8-16(3-3)
8	4	-8	9	4	(9,4)	0+8-16(4-3)

2. (3,2) to (9,6)

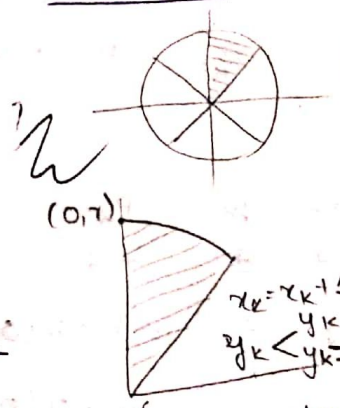
$m = \frac{6-2}{9-3} = \frac{4}{6} = \frac{2}{3} < 1$ $\quad \Delta y = 4, \Delta x = 6, 2\Delta y = 8, 2\Delta x = 12$

$P_K = 2(4) - 6 = 2$

x_k	y_k	P_K	x_{k+1}	y_{k+1}	(x_{k+1}, y_{k+1})	
—	—	—	3	2	(3,2)	
3	2	2	4	3	(4,3)	
4	3	-2	5	3	(5,3)	2+8-12(1)
5	3	+6	6	4	(6,4)	-2+8-12(0)
6	4	2	7	5	(7,5)	6+8-12(1)
7	5	-2	8	5	(8,5)	2+8-12(1)
8	5	6	9	6	(9,6)	-2+8-12(0)

3) Mid point circle [Bresenham's circle Algorithm]

Circle follows symmetrical property. As radius is same for all quadrants any one quadrant calculation is fine. In one quadrant one octant is fine



next coordinates may be (x_k+1, y_k) or (x_k+1, y_k-1)

mid point $= \frac{x_k+1+x_k+1}{2}, \frac{y_k+y_k-1}{2}$

$= \left(x_k+1, y_k - \frac{1}{2}\right)$

we have to apply this circle formula $x^2+y^2=r^2$

$$\boxed{P_K = (x_k+1)^2 + \left(y_k - \frac{1}{2}\right)^2 - r^2} \quad -(1)$$

This is Initial decision parameter

To find next, calculate P_{k+1}

From now on the next dp can be calculated by diff blw P_{k+1} & P_k

$P_{k+1} = (x_{k+1}+1)^2 + (y_{k+1} - \frac{1}{2})^2 - r^2$

$P_{k+1} - P_k = ((x_{k+1}+1)+1)^2 + (y_{k+1} - \frac{1}{2})^2 - (x_k+1)^2 - \left(y_k - \frac{1}{2}\right)^2$

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

$= 2(x_k+1) + y_{k+1}^2 - y_k^2 - (y_{k+1} - y_k) + 1$

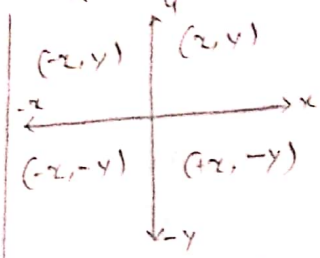
$$\boxed{\star\ P_{k+1} = P_k + 2(x_k+1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1}$$

Initial dp at ① starting from $(0, r)$

$$P_k = (0+1)^2 + \left(r - \frac{1}{2}\right)^2 - r^2 = 1 + r^2 + \frac{1}{4} - r^2 - r$$

$$= \frac{5}{4} - r$$

☆ $\boxed{P_k = 1 - r}$ → Initial dp.



I Octor

Quadrant 1: calculate till (x, 0)
Quadrant 2: write all values with '-' x
3: -x, -y
4: +x, -y

Conclusion:

if $(P_k \geq 0)$

{
$x_{k+1} = x_k + 1$

$y_{k+1} = y_k - 1$
}

else

{
$x_{k+1} = x_k + 1$

$y_{k+1} = y_k$
}

Q. Draw a circle of radius 8 units using MPCircle algorithm.

II Octant

given $r = 8$

Initial dp $P_k = 1 - r = -7$

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

x_k	y_k	P_k	x_{k+1}	y_{k+1}
0	8	-7	1	8
1	8	$-7 + 2(0+1) + (8^2 - 8^2) - (8-8) + 1 = -7 + 2 + 1 = -4$	2	8
2	8	$-4 + 2(2) + (0) - (0) + 1 = -4 + 4 + 1 = 1$	3	7
3	7	$1 + 2(3) + (49 - 64) - (-1) + 1 = 1 + 6 + (-15) + 1 = -7$	4	7
4	7	$-7 + 2(4) + (0) - 0 + 1 = -7 + 8 + 1 = 2$	5	6
5	6	$2 + 2(5) + (36 - 49) - (-1) + 1 = 2 + 10 + 1 - 13 = -9$	6	5
6	5	$-9 + 2(6) + 0 - 0 + 1 = -9 + 12 + 1 = 4$	7	3
7	3	$4 + 2(7) + 9 - 25 - 2 + 1 = 4 + 14 - 16 - 1 = 11$	8	2
8	2	$11 + 2(8) + 4 - 9 - 1 + 1 = 11 + 16 - 5 = 22$	8	1
8	1	$22 + 2(9) + 1 - 4 - 1 + 1 = 22 + 18 - 3 = 37$	8	0

9. $r = 12$

$P_k = 1 - 12 = -11$

x_k	y_k	P_k	x_{k+1}	y_{k+1}
0	12	-11	1	12
1	12	$-11 + 2(1) + (12^2 - 12^2) - (0) + 1 = -11 + 2 + 1 = -8$	2	12
2	12	$-8 + 2(2) + 0 - 0 + 1 = -8 + 4 + 1 = -3$	3	12
3	12	$-3 + 2(3) + 0 - 0 + 1 = -3 + 6 + 1 = 4$	4	11
4	11	$4 + 2(4) + 121 - 144 - (11 - 12) + 1 = 4 + 8 - 23 + 1 + 1 = -9$	5	11
5	11	$-9 + 2(5) + 0 - 0 + 1 = -9 + 10 + 1 = 2$	6	10
6	10	$2 + 2(6) + 100 - 121 - (-1) + 1 = 2 + 12 - 21 + 2 = -5$	7	10
7	10	$-5 + 2(7) + 0 - 0 + 1 = -5 + 14 + 1 = 10$	8	9
8	9	$10 + 2(8) + 81 - 100 + 1 + 1 = 10 + 16 - 19 + 2 = 9$	9	8
			10	7
			10	6
			11	5
			11	4
			12	3
			12	2
			12	1
			12	0

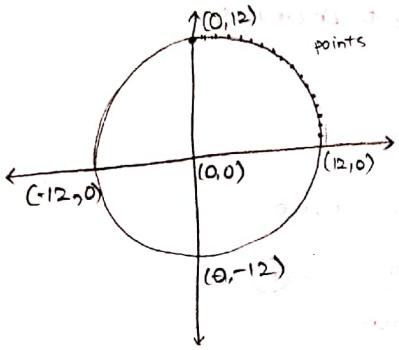
I Octant

II Octant

Quadrant 2: x_{k+1} with $(-)$

3: $-x, -y$

4: $-x, -y$



1. Consider three different raster systems with resolutions of 640 x 480, 1280 x 1024, and 2560 x2048.
   What size is frame buffer (in bytes) for each of these systems to store 12 bits per pixel?

Frame-buffer size for each of the systems is

640 × 480 × 12 bits ÷ 8 bits per byte = 450 KB

1280 × 1024 × 12 bits ÷ 8 bits per byte = 1920 KB

2560 × 2048 × 12 bits ÷ 8 bits per byte = 7680 KB

For 24 bits of storage per pixel, each of the above values is doubled.

2. How much time is spent scanning across each row of pixels during screen refresh on a raster system with a resolution of 1280 x 1024 and a refresh rate of 60 frames/second?

Resolution = 1280 X 1024

That means system contains 1024 scan lines and each scan line contains 1280 pixels

Refresh rate = 60 frame/sec.

1 frame takes = 1/60 sec = 0.01666 sec.

1 frame buffer consist of 1024 scan lines (It means then 1024 scan lines takes 0.01666 sec)

1 scan line takes = 0.01666/1024 = 10.6 μsec

3. Find out the aspect ratio of the raster system using 8 x 10 inches screen and 100 pixel/inch.

Aspect ratio =Width /Height

= 8 x 100/10 x 100

= 4 / 5

Aspect ratio =4 : 5

4. Compute access time per pixel, for systems with resolutions (a) 640*480 and (b) 1280*1024. Assume a refresh rate of 60fps.

The access time per pixel = 1/access rate.

The access time is around 54 nanoseconds/pixel for the 640 x 480 system,

The access time is around 12.7 nanoseconds/pixel for the 1280×1024 system.

5. Let the average time to execute an instruction in the display list be 33.33 μs. If the frame rate is 30fps, obtain the maximum number of instructions that may be present in the display list (for random-scan displays)

6. Implement the DDA algorithm to draw a line from (0,0) to (6,6)

7.Using the DDA algorithm digitize a line with end points (10,15) and (15,30).

8. Digitize a line with end points (20, 10) and (30, 18) using DDA line drawing Algorithm.

9. Starting point of line is (2,2) and ending point of line is (7,4) ,calculate the intermediate points by using bresenham line algorithm

10. Scan convert a line from (1,1) and (8,5) with 0 <m< 1,calculate using bresenham line algorithm

11. Use the midpoint circle algorithm to draw the circle centred at (0,0) with radius 15