Graphics Hardware

Display Devices
- Vector Scan
- Raster Scan

Both based historically on CRT (TV)
- Electron beam accelerated toward screen
  - focused
  - deflected
  - strikes phosphorescent material on screen
    --> pixel that glows

A Pixel
- Visible point where electron beam hits screen
- Screen phosphors glow & fade
- Have a finite size
- Not a mathematical point

Resolution
- Maximum number of pixels that can be plotted without overlap
- Expressed as: # horizontal X # vertical pixels
- Depends on:
  - phosphor used
  - focusing system (how small a point)
  - Speed/precision of deflection system
  - video memory size (raster scan)—as we’ll see
Aspect Ratio
- Ratio of # of pixel columns to # of pixel rows
- Examples:
  - SVGA VESA mode 100h: 640 X 400, A.R. = 1.6
  - Standard Windows: 640 X480, A.R. = 1.33
- **Pixel Ratio** (often called Aspect Ratio)
  - Ratio of pixel height to pixel width
  - Ratio of # of horizontal pixels to vertical pixels needed to produce equal length lines
  - For a square screen, A.R. = P.R.
  - If Pixel Ratio \( \neq 1 \), figures are distorted

Dot Pitch
- Minimum distance between centers of adjacent pixels of same color
- Should be less than 0.28 mm for sharp images
- For fixed sized screen
  - Decreasing distance between pixels \( \Rightarrow \) Increase Resolution
  - So dot pitch determines max resolution

Persistence
- After beam leaves a phosphor, it fades
- **Definition of persistence**:
  - Time to reduce initial intensity by 10%
  - Value depends on type of phosphor (10 - 100 msec.)
- Finite persistence \( \Rightarrow \) Screen must be redrawn
  - Refresh rate determined by persistence
- Example: If persistence = 20 msec
  - 1st pixel on screen invisible after that time
  - screen must be refreshed once every 20 msec
  - so refresh rate must be > 50 Hz.

Graphics Hardware Systems
- CPU—Runs program in main memory
  - Specifies what is to be drawn
- CRT—Does the actual display
- Display Controller—Provides analog voltages needed to move beam and vary its intensity
- DPU—Generates digital signals that drive display controller
  - (Offloads task of video control to separate processor)
- VRAM—Stores data needed to draw the picture
  - Dual-ported (written to by CPU, read from by DPU)
  - Fast (e.g., 640X480, 50 Hz \( \Rightarrow \) 65 nsec access time!)
  - Also called Refresh Buffer or Frame Buffer
- I/O devices—Interface CPU with user

A Computer Graphics Hardware System (General)
Flat-Panel Displays
- Technologies to replace CRT monitors
- Reduced volume, weight, power needs
  - Thinner: can hang on a wall
- Two categories
  - Emissive and non-emissive

Flat Panel Displays: Emissive Devices
- Convert electrical energy to light
- Plasma panels (gas-discharge displays)
  - Voltages fired to intersecting vertical/horizontal conductors cause gas to glow at that pixel
  - Resolution determined by density of conductors
  - Pixel selected by x-y coordinates of conductors
  - These are “raster” devices
- Other technologies
  - All require storage of x-y coordinates of pixels
  - Examples:
    - Thin-film electroluminescent displays
    - LEDs
    - Flat CRTs

Flat Panel Displays: Non-emissive Devices
- Use optical effects to convert ambient light to pixel patterns
- Example: LCDs
- Like emissive devices, require storage of x-y coordinates of pixel to be illuminated

Vector Scan Systems
- Also called random, stroke, calligraphic displays
- Images drawn as line segments (vectors)
- Beam can be moved to any position on screen
- Refresh Buffer stores plotting commands
  - So Refresh Buffer often called “Display File”
  - Provides DPU with needed endpoint coordinates
  - Pixel size independent of frame buffer
    - ==> very high resolution

Advantages of Vector Scan
- High resolution (good for detailed line drawings)
- Crisp lines (no “jaggies”) 
- High contrast (beam can dwell on a pixel==>very intense)
- Selective erase (remove commands from display file)
- Animation (change line endpoints slightly after each refresh)
Disadvantages of Vector Scan

- Complex drawings can have flicker
  - Many lines
    - so if time to draw > refresh time ==> flicker
  - High cost--very fast deflection system needed
  - Hard to get colors
  - No area fill
  - so it's difficult to use for realistic (shaded) images
  - 1960s Technology, only used for special purpose stuff today

Raster Scan Systems (TV Technology)

- Beam continually traces a raster pattern
- Intensity adjusted as raster scan takes place
  - In synchronization with beam
  - Beam focuses on each pixel
  - Each pixel's intensity stored in frame buffer
  - So resolution determined by size of frame buffer
  - Each pixel on screen visited during each scan
  - Scan rate must be $\geq$ 30 Hz to avoid flicker

Disadvantages of Raster Scan

- Low cost (TV technology)
- Area fill (entire screen painted on each scan)
- Colors
- Selective erase (just change parts of bitmap)
- Bright display, good contrast
  - but not as good as vector scan can be:
  - can't make beam dwell on a pixel
Disadvantages

- Large memory requirement for high resolution
  - (but cost of VRAM has decreased a lot!)
- Aliasing (due to finite size of frame buffer)
  - Finite pixel size
  - Jagged lines (staircase effect)
  - Moire patterns, scintillation, "creep" in animations
- Raster scan is the principal “now” technology for graphics displays!

Tektronix Direct View Storage Tube

- 1st “inexpensive” graphics display device
- Extension of vector scan technique
- Two electron guns
  - writing gun
  - flood gun

Advantages to DVST

- No refresh needed
  - unlimited image complexity possible
- High resolution
- Crisp lines
- Low cost
  - no fast refresh circuitry needed

Disadvantages to DVST

- No selective erase
  - whole image or nothing
- No animation
- Low light output
  - poor contrast
  - must use in subdued light
- No color
- No area fill

Erasure of DVST image

1. Plus charge applied to entire grid
   - Attracts electrons to entire grid
   - Entire screen flashes (Image gone)
2. Minus charge applied to entire grid
   - Provides electrons that can be knocked out by writing gun
   - Ready to draw next image with writing gun

Writing gun beam knocks electrons out leaves + charges behind (constitute image)
Flood gun supplies continuous source of unfocused electrons
  - migrate toward the + charges on grid
  - pass through grid and strike screen phosphors
  -> lighted dots
  - electrons continue to hit + charges
  - continuous light (Up to an hour)
Interlaced Displays

- All even then all odd screen lines scanned
- Typically 1/60 second each
  - Same image presented twice in 1/30 second
  - Image changed at 1/2 non-interlaced frequency
  - less demands on image generation system
  - can be less expensive
  - 30 Hz is borderline for flicker
  - lower quality image (seeing half the image at a time)

Color Display Hardware (raster)

- Each pixel composed of 3 phosphors
  - glow red, green, and blue
- 3 electron guns shoot their beams through a shadow mask
  - so beams hit the sensitive phosphors
  - intensity of 3 beams determines how bright each phosphor glows
  - human eye detects an additive color mix
  - e.g., max red, green, & blue perceived as white

Direct color systems

- Frame buffer divided into bit planes
- A bit plane contributes one bit to the color of each pixel on the screen
- If resolution of the screen is W x H pixels:
  - a bit plane is a W x H x 1 bit memory
- Bit planes can be organized into 3 sets
  - Each called a color channel: (R, G, B)
  - Bit planes of a color channel provide the intensity values fed to that channel’s electron gun
- A system with N bit planes per color channel:
  - $2^N$ red, $2^N$ green, & $2^N$ blue shades
  - $2^{3N}$ different colors displayable simultaneously

True Color & High Color Systems

- True color: direct color system with:
  - N=8
  - so $2^{24} = 16,777,216$ different colors possible for each pixel on screen
  - more colors than discernable by human eye
- High color: direct color system with:
  - Nr=5, Ng=6, Nb=5
  - $2^{16} = 65,536$ different colors possible

Indirect Color Systems

- Values stored in bit planes are indices into one or more color lookup tables (CLUTs)
  - CLUT stores R, G, B intensity values
  - # of bit planes determines # of colors displayable simultaneously on screen
  - width of CLUTs determines # of possible colors.
Indirect Color Systems, continued

- If system has N bit planes per color channel
- And each set of bit planes indexes a CLUT of width w,
  - Then number of entries in each CLUT = \(2^w\)
  - We say there are \(2^N\) colors displayable chosen from a total of \(2^w\) possible colors
  - Each set of \(2^N\) colors often called a palette
  - CLUTs often called palette registers

Advantages to Indirect Color

- Wide CLUTs (large w) \(\implies\) huge number of possible colors
- Modest # of bitplanes (small N) \(\implies\) VRAM not excessive in size
- Also, number CLUT entries is modest
  - So we get lots of possible colors with relatively little memory expense
- Fast animation for certain effects
  - just change CLUTS

Down Side to Indirect Color

- Ultimately number of colors on screen is limited by number of bit planes (N)
  - Even if large number of possible colors (large CLUT w), only a small fraction of them are usable at once
  - So graphics applications must set up CLUTs with values corresponding to most frequently occurring colors in scene
    - Different scenes might require different combinations of colors in the CLUTs
  - Can be slower: 2nd memory access

Color Graphics on a PC

- Graphics capabilities depend on display adapter (video card) in the system
- Common ones:
  - CGA (Color Graphics Adapter)
  - EGA (Enhanced Graphics Adapter)
  - VGA (Video Graphics Array)
  - Many different types of SVGA cards
  - Each display adapter can function in many different text and graphics modes
  - Backwards compatibility

SVGA Adapters

- Many manufacturers
- Each designed differently
  - Each programmed differently at the pixel level
  - No compatibility
  - Most compliant with VESA standards
    - so VESA SVGA modes can be programmed with relative ease
    - often at the expense of performance
Setting the PC Graphics Mode of Operation
- Easiest way: use the BIOS VGA Services
  - via video interrupt 0x10
  - set AH register to 0 (set mode)
  - set AL to desired mode
  - make call to INT 0x10
- INT 0x10 can be used for many other graphics/video functions
  - usually very slow

VGA Graphics Modes
- Support all CGA and EGA modes
- Add 640 X 480 X 16 colors
- Add 320 X 200 X 256 colors (mode 13h)
- Also other modes

VGA Mode 13h–A Simple Example of Indirect Color
- One byte of VRAM controls one pixel
  - Row major ordered
- VRAM starts at address 0xa000:0000
- To set pixel at (x,y) to a given color:
  - Set a segment register (ES) to start of video RAM
  - Compute pixel offset = 320 * y + x
  - Load offset into a pointer register
  - Set pixel by loading location with a color (byte), e.g., MOV ES:[SI], color

VGA Mode 13h, continued
- Indirect color control thru 256 X 18 CLUT
- Color written to VRAM is a byte-size index into this CLUT
- Table entries: 6 bits red, 6 green, 6 blue
  (0=no intensity, 63=maximum intensity)
- To change an entry in the VGA CLUT
  - use the video interrupt (10h):
    - AH=10h, AL=10h, BX=CLUT position (0-255)
    - DH, CH, CL = R, G, B intensity: (0-63 each)

VESPA SVGA BIOS Extension (VBE)
- Using high resolution, high color SVGA display modes in a standard way
- Documentation available at:
  - www.vesa.org/vbe3.pdf
  - entire document with example programs in Adobe pdf format
  - www.faqs.org/faqs/pc-hardware-faq/supervga-programming/

Graphics under Microsoft Windows
- Windows GDI does not permit direct access to Display Adapter
- Must use GDI calls to do graphics
  - SLOW!
- Or Special Libraries
  - OpenGL
  - DirectX
Color Under Windows

- Direct or Indirect
- Direct Modes:
  - 16 bit high color
  - 24 bit true color
    - R, G, B: 8 bits each
    - $2^{24}$ different colors
  - Use RGB() macro to get a COLORREF
    - If used in low color modes, get color dithering

Windows Indirect Color Modes

- 256 entry CLUT (8 bits)
- 16 entry CLUT (4 bits)
- CLUTS called palettes
- Controlled by Windows “Palette Manager”
  - A part of the GDI
- Using a color in the CLUT:
  - PALETTEINDEX(i) instead of RGB()
- We’ll look at the 256-color palette

The System Palette

- Maintained by Palette Manager
- Sort of like the physical CLUT
- Entries contain 8 bits per color channel
- 20 “static” colors initially defined
- Contents determine colors displayed
- Used by all applications
  - Shared between all windows
- Arbitrary changing it could mess up color of other windows

Changing the Palette

- Create a “logical palette”
  - Use CPalette::CreatePalette()
  - Set up with desired colors
- Select into a Device Context
- “Realize” it
  - i.e., map it to the system palette
  - done by calling CDC::RealizePalette()

Color Mapping with RealizePalette()

- Causes Palette Manager to compare colors in logical palette with system palette
  - exact match⇒
    - log. palette entry mapped to phys. palette entry
  - no exact match⇒
    - if available free entry, copy and map
    - if not, map to closest existing entry
- Active foreground application mapped first
  - So background window colors can change
Details in Changing System Palette

1. Set up a logical palette structure
   Windows LOGPALETTE structure:
   
   ```
   WORD palVersion;    // 0x300
   WORD palNumEntries; // # colors to change
   PALETTEENTRY palPalEntry[1] //new colors
   - (you may want to define & use your own logical palette struct)
   PALETTEENTRY structure:
   BYTE peRed;       // new color’s red intensity
   BYTE peGreen;    // green intensity
   BYTE peBlue      // blue intensity
   BYTE peFlags;    // usually 0
   ```

2. Create the palette:
   CPalette::CreatePalette(LPLOGPALETTE);
   
   - Member function of CPalette
   - Takes ptr to desired logical palette structure
   - May need to be typecast
   - Returns nonzero if successful

3. Select it into the DC:
   CDC::SelectPalette(pPal,FALSE);

4. Map current log. palette to sys. palette
   CDC::RealizePalette();

5. Use the new palette:
   PALETTEINDEX(i) instead of RGB()
   for example:
   ```
   COLORREF color;
   color = PALETTEINDEX(5);
   SetPixel(x,y,color);
   ```

6. When finished, get rid of it:
   Select it out of the DC with
   CDC::SelectPalette()

II. Computer Graphics Software

1. Lowest Level (earliest)--
   Assembly/machine language
   - Programs drive hardware directly
     - Fast, but non-portable
     - Difficult to program
     - Prone to errors
2. Medium Level (General Programming Packages)

A. Extensions to high level languages—graphics libraries
- e.g., Borland’s BGI, Windows’ GDI, Silicon Graphics GL, Microsoft’s DirectX
- Still have platform dependencies
- Easier to program
- Usually slower, but with optimized compilers, not so bad

B. Standard Graphics Packages
- Sets of specifications
- Supposedly language/platform independent
- Usually with bindings for many high level languages
- Syntax for accessing graphics functions
- Examples: GKS, PHIGS, OpenGL

3. Special-purpose Application packages
- e.g., Corel Draw, 3D Studio, Harvard Graphics, Photoshop
- Good for what they do, but specific uses

We’ll be working at level 2 for most of this course

Describing positions of objects
- Need coordinate systems
- 2-D and 3-D Cartesian systems used universally

2-D Cartesian System
- Measure distance along two mutually perpendicular axes
- Position given by 2 numbers \((x, y)\)
3-D Cartesian System
- Measure distance along three mutually perpendicular axes
- Position given by three numbers (x, y, z)

Other commonly-used systems
- Depend on symmetry
  - Spherical (ρ, θ, φ)
  - Cylindrical (r, θ, z)
- Conversion formulas used

Types of coordinate systems

1. Modeling coordinate system (MCS)
   - System used by programmer (modeler) to describe a single object
   - Can be 2-D or 3-D
   - Depends on object being modeled
   - Origin, scale picked according to object
   - Varies from object to object

2. World Coordinate System (WCS)
   - Reference system used to position objects in a scene
   - Can be 2-D or 3-D
   - Origin, scale, units specific to scene
   - Coordinate transformations map from an MCS to the WCS
     - Effectively positions objects in scene
     - Called the “Modeling Transformation”

3. Device Coordinate System (DCS)
   - Coordinate system used by output device
   - Units usually pixels
   - Always 2-D
   - Varies according to HW platform
   - Graphics SW maps from WCS to DCS
     - Called the “Viewing Transformation”
   - If WCS is 3-D, a projection transformation is also involved
4. Normalized Device Coordinates (NDCS)

- 2-D system
- $0 \leq x \leq 1$; $0 \leq y \leq 1$
- Intermediate between WCS and DCS
- Hardware-independent

Graphics Transformation Pipeline

- MCS $\rightarrow$ WCS $\rightarrow$ NDCS $\rightarrow$ DCS
- Modeling xform, viewing xform, HW-depend. xform
- The first two transformations are device independent