Binghamton University

EngiNet™

State University of New York

Thomas J. Watson

School of Engineering
and Applied Science

EngiNet™

WARNING

CS 560

Computer Graphics

Professor Richard Eckert

Lecture # 5

February 5, 2001

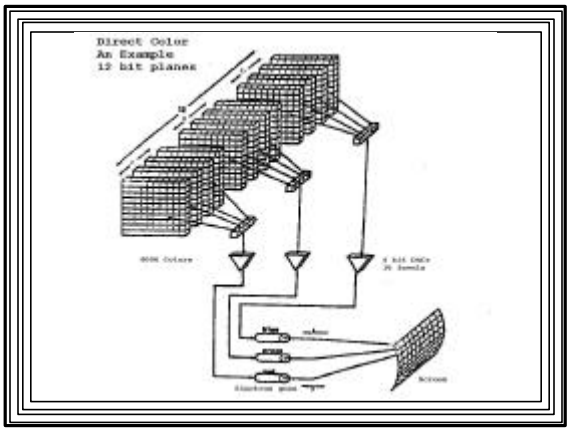
Lecture 5

Monday, 2-5-01

Computer Graphics Hardware, continued

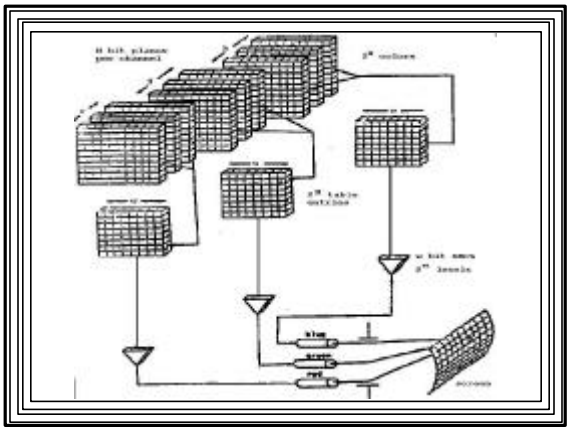
Direct color systems

- Frame buffer divided into bit planes
- A bit plane contributes one bit to color of pixels
- 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



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^N**
  - **We say there are 2^{3N} colors displayable chosen from a total of 2^{3w} possible colors**
  - **Each set of 2^{3N} colors often called a palette**
  - **CLUTs often called palette registers**

Advantages to Indirect Color

- Wide CLUTs (large w) ==>huge number of possible colors
- Modest # of bitplanes (small N) ==> 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 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

II. 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

CGA/EGA/VGA Display Modes

- See Link to Video Modes on a PC
  - On CS-460/560 Notes Web Page
- URL:

http://www.cs.binghamton.edu/~reckert/220/video0.htm

SVGA Adapters

- Many manufacturers
- Each designed differently
  - Each programmed differently at 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 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 pos'n (0-255)**
    - **DH, CH, CL = R, G, B intensity: (0-63 each)**

VESA 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.itsnet.com/home/dragon/Specs/Super.VGA.Adapters/VESA/index.html
  - www.faqs.org/faqs/pc-hardware-faq/supervga-programming/

Windows 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 different 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 256-color palette

The System Palette

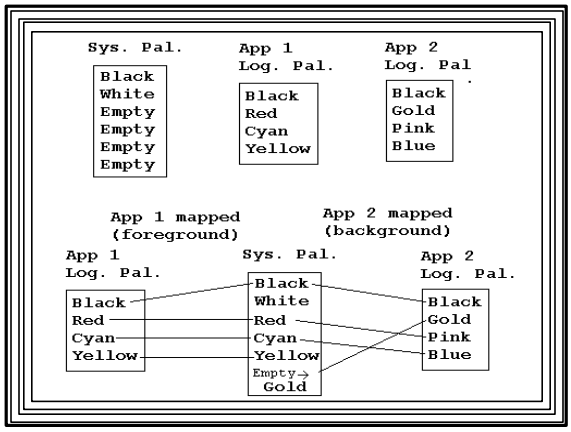
- Maintained by Palette Manager
- Sort of a copy of physical CLUT
- Entries contain 8 bits per color channel
- 20 "static" colors initially defined
- Contents determine colors displayed
- Sharing between windows fairly complex
- Arbitrary changing could mess up color of other windows

Changing the Palette

- Obtain a "logical palette"
  - Set up with desired colors
- Select into a Device Context
- "Realize" it
  - i.e., map it to the system palette
  - done by calling 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 app 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 use your own 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:

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:

RealizePalette();

5. Use the new palette:

PALETTEINDEX(i) instead of RGB()

6. **When finished, get rid of it:**

Select it out of the DC w/ SelectPalette()

III. 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