Computer Labs: Buffering in Computer Graphics

2º LEIC

Pedro F. Souto (pfs@fe.up.pt)

April 12, 2024

Raster graphics vs. Vector Graphics

Raster graphics An image is represented as "a rectangular matrix or grid of square pixels".

- Each pixel has a numeric value, usually a color
- The position of the pixel in the grid, is determined implicitly
- The size of an image depends on the resolution
- Formats often use compression to reduce size
- File formats: XPM, BMP, PNG, TIFF

Vector graphics Images are created from geometric shapes such as points, lines, curves, on a Cartesian plane.

- It allows a higher degree of geometric precision than rasters graphics
- Its size is independent of the resolution
- File formats: SVG, EPS, PDF

Computer Display Hardware

Raster displays Virtually all displays nowadays, are raster displays, i.e. they have a matrix of pixels

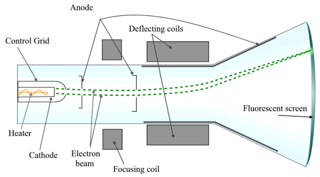
- The image is stored in a frame buffer
- The hardware is simpler
- Takes a constant time to redraw a screen, independently of the number of graphical objects
- Images have jagged edges and geometric shapes are approximated

Vector displays Were the first technology (early 60s), and generate images from paths

- Are able to draw continuous and smooth lines
- Hardware is more complex and expensive
- Refresh rate depends on the number of "paths" in the image
  - May flicker, if the image becomes too large.

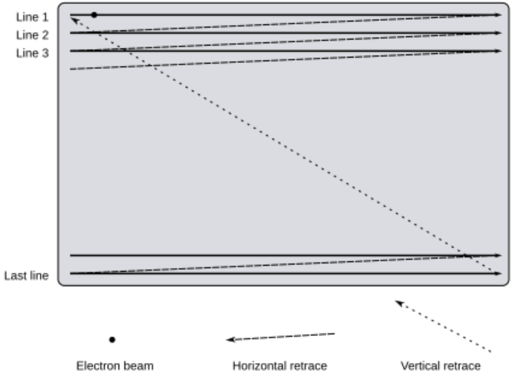
Cathode Ray Tube (CRT) Raster Display Operation

CRT Hardware



src: Theresa Knott

Raster Scan



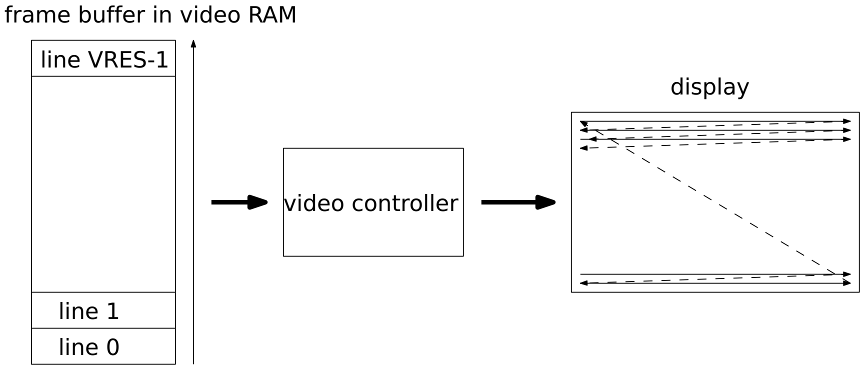
src: Ian Harvey

- The color of each pixel depends on the respective value in VRAM
- The controller refreshes the screen with a frequency $\geq$ 50/60 Hz

LCD Panels

Introduction to graphics and LCD, NXP

Outputing of the Frame Buffer on the Display



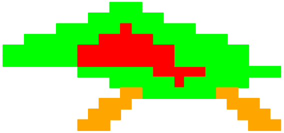
To change the frame displayed we need to change the contents of the frame buffer, in VRAM

There are several alternatives

1. Upon change of a sprite position, change the contents of the frame buffer
2. · · ·

Modify frame buffer, upon change of sprite position

Issue if we modify the frame buffer contents of the region that is being scanned, the picture on the display may show visual artifacts:



- A piece of the display will show what was there before the change, and another piece will show the new contents
- **It there is no new change**, the problem will be fixed the next time the screen is refreshed, i.e. within 16.67 or 20 ms (depending on the vertical refresh frequency)
- How bad is this?
  - It depends on how frequently it occurs (the more dynamic the frames, the worse).
  - The number and size of the sprites also matter, and the colors too

How to avoid these visual artifacts?

No solution Synchronize the application with display refreshment

- If the application modified the frame buffer in the wake of the video controller, there would be no visual artifacts

Double buffering I.e. use a second buffer, **the back buffer** in RAM

1. Create the new frame in the back buffer
2. Copy the new frame from the back buffer to the frame buffer in video RAM (VRAM)

Page Flipping The two buffers are in VRAM

- The video controller has a register that points to the buffer being displayed on the screen, **the front buffer**
- Rather than copying, just flip the two buffers, i.e. change the contents of that register to point to the **back buffer**
  - and swap the role of the buffers: the back buffer becomes the front buffer, and vice-versa
- Use VBE function 0x07 (Set/Get Start of Display), of VBE 2.0 Specification

Vertical Synchronization

Issue Double buffering, regardless of implementation, may not completely eliminate visual artifacts, although it reduces them

- If copy or flip occur in the middle of a scan, the screen will display tearing

Solution Use the vertical retrace interval, i.e. the interval between rendering the right most pixel of the bottom line of one frame, and the rendering of the left most pixel of the top line of the next frame

- For CRTs it is in the order of 500 μs
- For LCDs it is much shorter (Some years ago `xvidtune` reported about 100 μs in my laptop at the time.)

Issue How to synchronize with vertical retrace?

VGA has some registers to configure interrupt generation on vertical retrace

VBE function 0x07 Set/Get Display Start, sub-function 0x80, Set Display Start during Vertical Retrace, allows to swap the frame buffer in the following vertical retrace

Triple Buffering

Issue Vertical synchronization with double buffering may slow down the frame rate

- Assume the video controller is displaying frame n
- The application cannot start the following frame ($n+2$) as soon as it completes the next frame ($n+1$)
  - It must wait until the vertical retrace
    - The back buffer cannot be modified, otherwise one may loose the next frame ($n+1$)

Triple Buffering I.e. use a second back buffer

- After generating frame $n+1$ on the first back buffer, start generating frame $n+2$ on the second back buffer without delay
- Upon a vertical retrace, the back buffer with the most recent completed frame becomes the front buffer, and the previous front buffer a back buffer

Triple Buffering and VBE 2.0

Question Can we use VBE 2.0 function 0x07, sub-function 0x80, Set Display Start during Vertical Retrace, to implement triple buffering?

Answer Depends on the implementation

- In some implementations, Set Display Start during Vertical Retrace is a blocking call, in that case, there is no need for a third buffer
  - When it returns, there is always one free back-buffer, even if there are only two buffers.

For maximum frame rate Many games allow disabling vertical (retrace) synchronization

- Vertical synchronization eliminates visual artifacts
- But introduces a delay between the frame generation and its display on the screen

Further Reading

- John T. Bell, Vector vs Raster Displays, Computer Graphics Course Notes