ECE 531 – Advanced Operating Systems
Lecture 25

Vince Weaver

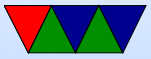
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

5 November 2025

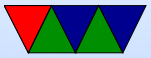
Announcements

- Project topics due today
  Send e-mail with topic and group members



Some Graphics Interface History

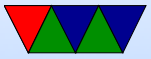
- Teletypes
- Vector Displays
- CRTs
- LCD displays



Video Game Systems – Before Framebuffers

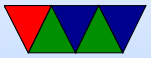
- Atari 2600 – Racing the Beam
  4k ROM, 128 bytes RAM, 40-pixel (5 byte) framebuffer
  3 sprites
  all calculation done during the retraces

- SNES Tile/Sprite Based
  RAM getting cheap enough can have framebuffers, but bandwidth still not that great.
  Use tiles, that let you split the display into tiles, with each large tile specified by a single byte.

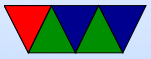


Video Adapters – Framebuffers

- Just an array of bytes that get displayed on the screen.
- Bits per pixel
  - 1 – monochrome
  - 4 – 16 colors
  - 8 – 256 colors (usually palette)
  - 15 – rgb 555
  - 16 – rgb 565 “true color”
  - 24 – rgb 888
  - 32 – rgba

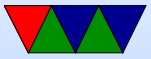


- Can be large: 1024x768x24bpp = 2.4MB, to update at 60Hz = 141MB/s
- Bit-planes
- Palette



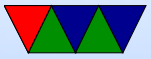
VGA Display example

- VGA text
- Memory mapped, IO ports
- Mode setting
- VESA BIOS
- “Mode X”
- Bitplanes
- Colors
- Loadable fonts



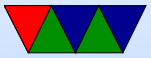
2D Graphics Acceleration

- Once you had nice framebuffer, want to draw to it faster
- GUI interfaces draw a lot of shapes, like rectangles, lines, mouse pointer, etc
- Cards started adding support for that



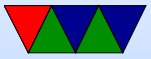
3D Graphics Acceleration

- Major industry change
- Started in expensive workstations (SGI, etc)
- Eventually made it into desktop systems



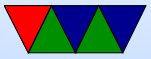
3D Graphics Acceleration – Methods

- Ray-tracing / Ray-casting – calculate scenes by bouncing around light rays. Best graphics but too slow to do in real time
- Rasterization – split scene up into triangles/quads. Calculate normals, base shading/lighting on that. Add textures.
  Can't do things like transparency or shadows.



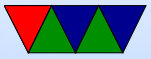
Graphics Processing Units (GPUs)

- Rasterization
- Draw lots of triangles, really fast
- Triangles / Textures / Z-buffers
- Can divide screen into small sections, and calculate massively parallel.
- Shaders



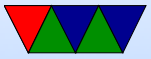
How are these programmed?

- OpenGL
- DirectX (Direct3D)
- Vulkan



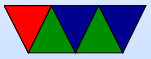
How do you get legacy 2D support?

- Just draw rectangle (two triangles) filling screen
- Draw your desired 2D images/text to a texture
- Apply texture to the rectangle



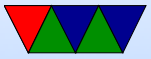
Modern GPUs

- Massively Parallel
- In old days had “fixed” 3D pipeline in hardware
- Modern systems instead it’s a massively parallel SIMD type processing pipeline
- 3D pipeline in software with programmable shaders

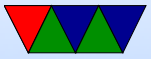


Shaders

- 3D shaders
  - Vertex shader, called on each vertex, can calc texture, convert to 2D, change position
  - Geometry shader, can make new vertices
  - Tesselation – makes meshes
  - Ray Tracing
- 2D shaders
  - Pixel (Fragment) Shaders - called on each pixel, can do things based on co-ords, nearby values



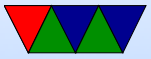
- Compute
  - GPGPU – general purpose code can be offloaded to GPU



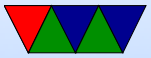
VideoCore GPUs

- In Pi. Old ones IV, newer VI?
- OpenGL ES 2.0 V3D
- Output to HDMI, DSI, DPI, composite
- In theory can do compute shader jobs, but no Linux interface
- QPU – 16-way SIMD (suited for calculating quad (x4) data

  In theory can use mailbox to submit QPU jobs
- Can run general purpose code unlike some GPUs



On Pi runs ThreadX, full OS by itself



Programming VideoCore

- Broadcom eventually released enough info for a full open-source Linux driver
- V3Dlib (?)
- videocoreiv on github, code to write baremetal simple OS running only on GPU