Real-Time Graphics Architecture

Kurt Akeley

Pat Hanrahan

http://www.graphics.stanford.edu/courses/cs448a-01-fall

Display and Framebuffer

Displays

- Key properties
- Bandwidth

Framebuffers

- Definitions and key properties
- Bandwidth
- Architecture

Required reading

- *Frame-Buffer Display Architectures*, Sproull, Annual Review of Computer Science, '86

Terminology

CRT

- Cathode Ray Tube

LCD

- Liquid Crystal Display (flat panel)

DLP

- Digital Light Processing
- Texas Instruments technology
  - Clever adaptation of IC / photo lithography

Raster vs. Calligraphic

Raster (image order)

- dominant choice

Calligraphic (object order)

- Earliest choice (Sketchpad)
- E&S terminals in the 70s and 80s
- Works with light pens
- Scene complexity affects frame rate
- Monitors are expensive
- Still required for FAA simulation
  - Increases absolute brightness of light points

Display Sequence Issues

Raster video signal takes a full frame to deliver

- Adds almost one frame of latency (worst-case)

Persistence

- Flying dot: CRT, scanning Laser
- Skewed full-frame: LCD panel, DLP ?
- Field sequential: consumer DLP, head-mount CRT

Visual artifacts

- Tearing in tiled displays
- Color separation in field sequential displays
- Motion blur of moving objects?

Display Sequence Issues (Cont.)

Interlace (vs. progressive)

- Two interlaced fields per frame
- Makes no sense for MPEG compression
- Included in HDTV spec!

Visual artifacts

- Flicker if image is poorly filtered
- Image doubling if render rate <= frame rate
- Disappearing objects

Display Resolution History

Rate of increase is low (1.1 compound overall)

LCD display has peak foveal pixel density at 3-feet

Date	Format and Technology	Bandwidth	Rate
1980	1024 x 768 x 60Hz, CRT	0.14 GB	
1988	1280 x 1024 x 72Hz, CRT	0.29 GB	1.1
1996	1920 x 1080 x 72Hz, HD CRT	0.60 GB	1.1
2001	3840 x 2400 x 56Hz, active LCD	1.55 GB	1.2

All figures are the author's estimates!

IBM's Bertha LCD Display

3840 x 2400 resolution, 22" diagonal 16:10 screen

Video Signal Generation

Implemented on GPU

Analog and digital streams

- Analog: complex waveform, critical timing
- Digital: emerging standards and capabilities

Typically supported:

- Gamma correction
- Different resolution displays

Optionally supported:

- Multiple signals / displays
- Genlock synchronization

Display Summary

RGB raster displays are prevalent

- Calligraphics as a pedagogical tool
- Ignore 3D displays

Video bandwidth

- Is a steady load on an operating GPU
- Is increasing slowly

Framebuffer Definitions

What is a framebuffer?

What can we learn by considering different definitions?

Framebuffer Definition #1

Storage for commands that are executed to refresh the display

Allows for raster or calligraphic display (e.g. Megatech)

"Framebuffer" for calligraphic display is a "display list"

- OpenGL "render list"?

Key point: framebuffer contents are ***interpreted***

- Color mapping
- Image scaling, warping
- Window system (overlay, separate windows, …)
- Address Recalculation Pipeline

Framebuffer Definition #2

Image memory used to decouple the render frame rate from the display frame rate

Meets common understanding of framebuffer as image

Leads naturally to *double buffering*

- One render buffer, one display buffer, swap
- *n*-buffering also possible, can control latency

Key idea: decoupling enables general-purpose GPU

- Visual simulation has high render frame rate
- MCAD has low render frame rate
- Window manager has no frame rate

Framebuffer Definition #3

All pixel-assigned memory used to assemble and display the images being rendered

Key point: framebuffer is active participant in rendering

Leads to non-color buffers: depth, stencil, window control

- OpenGL treats these buffers as part of framebuffer
- Some reserve "framebuffer" for color images
- Should be *n*-buffered in some cases (sort last)
- RealityEngine framebuffer can be deeper than wide or high

History cycles through this definition

- 2D manipulation
- 3D painters algorithm
- 3D depth, stencil, accumulation, multi-pass
- Programmable shading

Framebuffer is Optional

Calligraphic display

- If we don't treat display list as framebuffer

"Follow-the-beam" rendering

- Minimizes latency
- Saves cost if frames are never "dropped"

Talisman-like image assembly (3D sprites)

- Old idea (visual simulation, window systems)

GigaPixel render tile

- Framebuffer stores color images only
- Depth, stencil, etc. in small tile

Dominant Architecture is Consistent

SGI architectures look like

ATI architectures look like

NVIDIA architectures

Details are evolving, but big picture remains the same

Why is this?

- Simplicity of design
- Simplicity of algorithms
- Simplicity of immediate-mode approach

Simplicity of Design

Framebuffer fragment operations

- Blending: merge fragment and pixel color
- Depth Buffering: save nearest fragment
- Stencil Buffering: simple pixel state machine
- Accumulation Buffering: high-resolution color arithmetic
- Antialiasing: (to be covered later)
- ….

Key points:

- All utilize pixel data (not just fragment data)
- All are pixel independent (no neighbor data dependencies)

Why aren't fragment operations programmable?

Simplicity of Algorithms

Framebuffer employs brute-force simplicity

- Hidden surface elimination: Depth-buffer vs. sort/painter
- Capping: Stencil-based vs. object calculations
- Image-space algorithm is efficient
  - Just samples, never "object" information, locality
  - Just-in-time calculation, steady cost function

Accumulation Buffer (high-resolution color arithmetic)

- *The Accumulation Buffer*, Haeberli and Akeley, Proceedings of SIGGRAPH '90
- Volume rendering using 3D textures

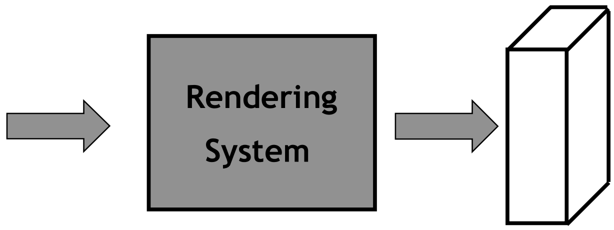
Multi-pass rendering

- *Interactive Multi-pass Programmable Shading*, Peercy, Olano, Airey, and Ungar, Proceedings of SIGGRAPH '00

Simplicity of Immediate-mode

Framebuffer is "context"

Matches 2D/window rendering model



Little graphics state is in rendering hdwr

Most graphics state is in framebuffer

Decreasing Display Bandwidth

Historically display bandwidth was a limiting factor

- Hence "Sproull's Rule": fill rate >= display rate

Now display bandwidth is almost inconsequential

Year	FB Bwth	Disp Bwth	Disp / FB
1984	0.3GB	0.14GB	1/2
1988	1.8GB	0.29GB	1/6 *
1996	12.8GB	0.60GB	1/20
2001	8.0GB	1.55GB	1/5(1/20)**

* VRAM provided separate video bandwidth

** Display requires four separate video signals

Maximize Effective Bandwidth

Display bandwidth is inconsequential, but

Framebuffer bandwidth is still critical, so

- Optimize access locality
- Utilize special purpose memory parts
- Maximize real bandwidth
- Embed framebuffer memory
- Minimize bandwidth needs
- Utilize parallelism
- Pool framebuffer memory

Consider these in more detail ….

Optimize Access Locality

DRAMs run faster when “local” accesses are back-to-back

Imagine that you have a “locality budget”

Allocate it carefully to

- Optimize for display refresh cycles, and/or
  - Scan line locality
- Optimize for triangle fill cycles, and/or
  - Square “tile” of locality
- Optimize for overlay display cycles, and/or
  - Pixel component locality
- ….

Utilize Special Purpose DRAM

Video DRAM (VRAM) in '80s

- Popular for a short period. E.g. SGI GTX.

Sun 3DRAM in the '90s

- Constrains the architecture
  - Pixel format, fragment operations, etc.
- Expensive

Standard DRAMs have evolved for framebuffer use

- Time-to-fill limits utility of narrow-deep DRAMs
- Wide-shallow parts result (current 32-bit DDRRAM)
- Will DRAMs fall behind? Have they already?

FBRAM

FBRAM is DRAM with video output buffers (as in VRAM) and a cached ALU to perform fragment operations.

This was not a successful product.

FBRAM: A New Form of Memory Optimized for 3D Graphics, Deering, Schlapp, and Lavelle, SIGGRAPH '94 Proceedings

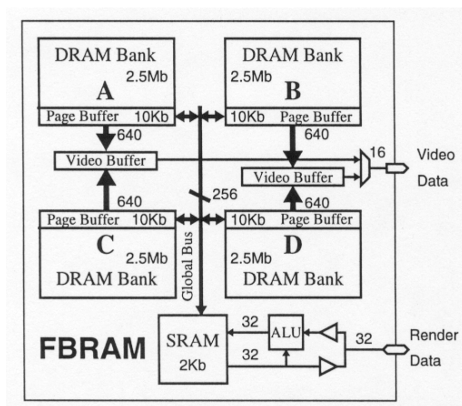


Figure 2. Internal block diagram of a single FBRAM.

Maximize Framebuffer Bandwidth

Use the fastest, widest DRAMs possible

Operate them at the highest possible clock rate

- Separate "pixel" clock and "memory" clock
- Bin memory (and GPU) parts
- Provide elasticity (FIFO) and synchronization

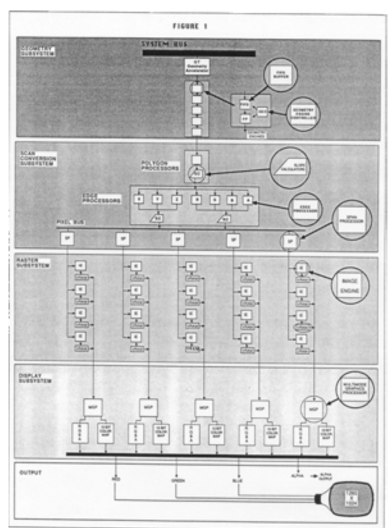
Make all wiring point-to-point

- Optimize signal paths
- Separate memory controller for each DRAM

GTX Block Diagram

Each of the 20 Image Engines was conceived as little more than a stand-alone memory controller with attached VRAM.

***High-Performance Polygon Rendering*, Akeley and Jermoluk, Proceedings of SIGGRAPH '88.**



Embed Framebuffer Memory

Examples

- Pixel Planes (earlier versions)
- Play Station 2

May be the ultimate answer

- When framebuffer memory is inconsequential

But

- It's expensive compared with commodity DRAM
- NVIDIA and ATI have done well without it

Minimize Bandwidth Requirements

Add transistors to make better use of bandwidth

Be frugal, make each memory cycle count

- Aggregate memory transactions
- Cache to get efficient use of memory bandwidth

Compress framebuffer data

- Utilize area redundancy

Optimize occlusion culling

- Backface, early depth test, hierarchical depth

Minimize need for multi-pass rendering

- Programmable shading

SGI Historicals – FB Bandwidth

Bandwidth increases at 1.4, pixel fill rate at 2.2

Year	Product	Zbuf rate	Yr rate	FB Bwth	Yr rate	
1984	Iris 2000	100K	-	0.3GB	-	DRAM*
1988	GTX	40M	4.5	1.8GB	1.6	VRAM**
1992	RealityEngine	380M	1.8	6.4GB	1.4	DRAM
1996	InfiniteReality	1000M	1.3	12.8GB?	1.2	SDRAM
			2.2		1.4	

* Physically separate front and back color buffers

** Not counting shift output bandwidth

NVIDIA Historicals – FB Bandwidth

Bandwidth increases at 1.5, pixel fill rate at 2.5

Season	Product	Fill rate	Yr rate	FB bwth	Yr rate
2H97	Riva 128	20M	-	1.6GB	-
1H98	Riva ZX	31M	2.4	1.6GB	1.0
2H98	Riva TNT	50M	2.6	2.0GB	1.6
1H99	TNT2	75M	2.3	2.9GB	2.1
2H99	GeForce	120M	2.6	4.0GB	1.9
1H00	GeForce2	200M	2.6	6.4GB	2.6
2H00	NV16	250M	1.6	8.0GB	1.3
1H01	NV20	500M	4.0	8.0GB	1.0
			2.5		1.5

Rent's Rule

Rent's rule:

$$\text{Bandwidth} = K_R \text{ Capability}^{0.7}$$

NV series exponent is 0.5 (against 0.46 expected)

NV20 does:

- Transaction aggregation
- Clever depth buffer fragment elimination
- Lossless data compression
- ….

Utilize Parallelism

Single-Instruction, Multiple-Data Parallelism (SIMD)

- Usually tiled rendering stamp (e.g. Stellar)
- Efficiency poor due to "pixel depth complexity"

Multiple-Instruction, Multiple-Data Parallelism (MIMD)

- Fragment operations are independent
- Individual memory controllers are more efficient
- SGI approach, merge them into Image Engines
  - Became massively parallel (hundreds of engines)
- NVIDIA approach also?
  - Parallelism limited to 4 or so, more pipelining

InfiniteReality Block Diagram

Fully-configured InfiniteReality system includes 320 Image Engines. Each combines a fragment processor with a memory controller.

Image Engines are packaged in groups of four.

InfiniteReality: A Real-Time Graphics System, Montrym, Baum, Dignam, and Migdal, Proceedings of SIGGRAPH '97.

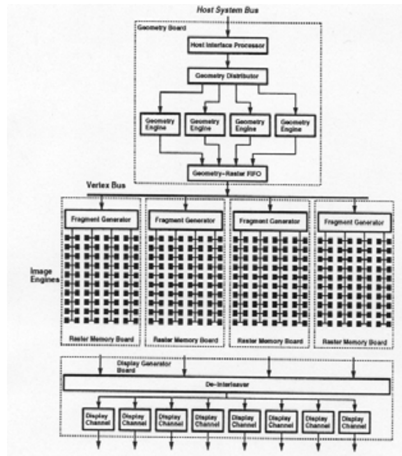


Figure 1: Board-level block diagram of the maximum configuration with 4 Geometry Engines, 4 Raster Memory boards, and a Display Generator board with 8 output channels.

Pool Framebuffer Memory

Single shared memory for all GPU needs

- Framebuffer, texture, "display list"
- Standard GPU solution (including SGI desktop)

Can share CPU memory too

- "System company" solution
- Lots of issues (latency, error correction, locality)
- SGI O^2

Automatically balances bandwidth needs

Addresses time-to-fill issue nicely

Requires crossbar for multiple memory controllers

Other Issues

Coordinate system

- Pixel is a region, not a point sample
- Pixels have integer coordinates, but
- Screen/window coordinates are continuous

Error detection/correction

- No SGI framebuffer has this (even O^2)
- Do others?

Why not map framebuffer into CPU address space?

- Lots of reasons
- DrawPixels/ReadPixels is the right interface

Conclusion

Elegant brute-force is working

- Complexity is localized
- Architecture remains unchanged

More transistors buy lower bandwidth needs

- CPU designers add cache memory
- GPU designers have lots of tools

Real-Time Graphics Architecture

Kurt Akeley

Pat Hanrahan

http://www.graphics.stanford.edu/courses/cs448a-01-fall