GeForce4

John Montrym

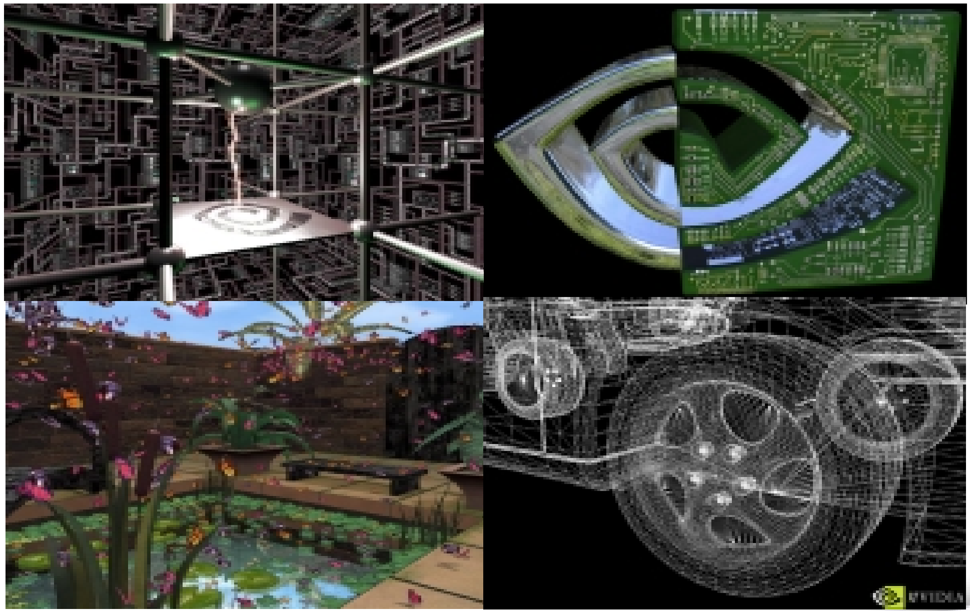
Henry Moreton

Architectural Drivers

- Programmability
- Parallelism
- Memory bandwidth

Recent History: GeForce 1&2

- First integrated geometry engine & 4 pixels/clk
- Fixed-function transform, lighting, and pixel pipelines
- 25M transistors : 0.18um/6LM : 250MHz
- 25M polygons/sec : 1G pixels/sec

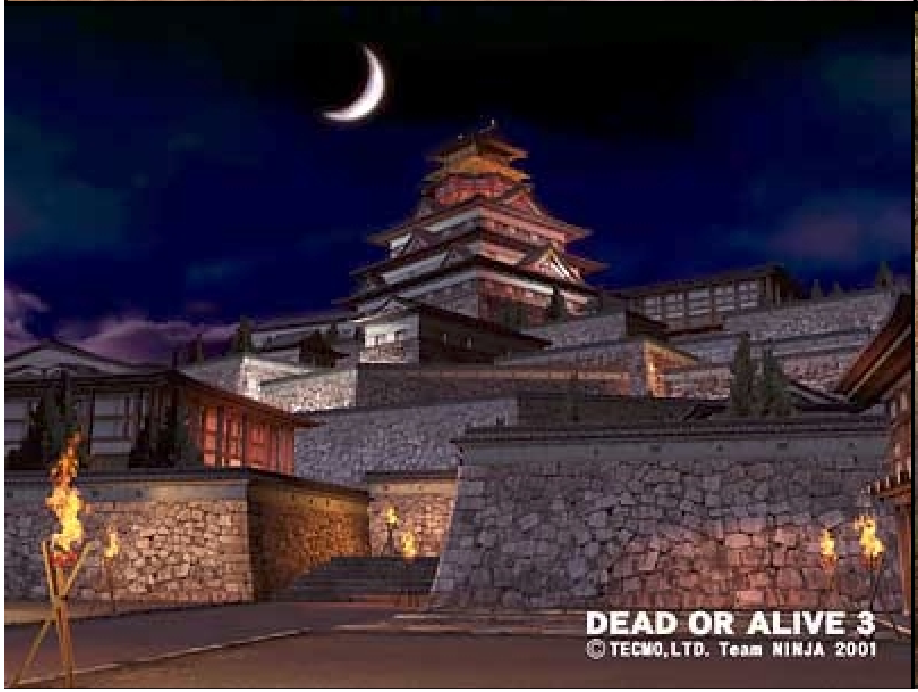


NVIDIA.

Rendering in Transition

- Pre-2001: pixel “painting”
  - Image complexity and richness from LOTS of pixels
  - Each pixel derived from 1-2 textures & blending
  - Detail added by transparency and layers
- Post-2001 fork in the road:
  - Paint more simple pixels, faster - embedded DRAM OR
  - Use Programmable Shading to render “better” pixels - but, must reduce depth complexity

Examples



GPUs vs. CPUs

- More independent calculations
  - Enables wide and deep parallelism
- API churn
  - shorter development cycles -> ASIC
- Blend of general- and special-purpose compute resources
- Both transistor-bound for the foreseeable future

NVIDIA.

Special-Purpose Hardware

- Most efficient implementations of
  - Cube environment map
  - Shadow calculations
  - Anisotropic filtering
  - Clipping
  - Rasterization
  - Log, exp, dot-product
- More programmability won't change this

Managing DRAM Bandwidth

- Very large working sets
- Affordable caches cannot support long-term reuse
- Target is effective streaming with local reuse
- Supercomputer techniques apply
  - Latency hiding
  - Vector operations

Fit the Machine to DRAM Characteristics

- **Page locality**
  - **DRAMs pages are 1-D**
  - **Graphics accesses are 1-D, 2-D, 3-D in memory**
- **Page misses and read/write turns getting more expensive**
- **Granularity**

Unified Memory

- Traffic comprises:
  - Commands --------------- primitives,state
  - Vertex data --------------- {x,y,z,w}
  - Texture samples
  - Depth values
  - Colors
- Relative amounts vary widely
- Powerful programming model

Eliminate Redundant Traffic

- We cache data
  - Textures, vertices
- We cache work
  - Pixel fragments
  - Post-transform vertices
- Designed for 80 - 90% hit rate (not 99.9%)
- Leverage coherence
  - Engines traverse locally

    ex: rasterization order

NVIDIA.

Amplify Peak Bandwidth

- Lossless compression
- Lossy compression
  - Conservative (undetectable)
  - Else application must be given the choice
- Small-grained random access favors fixed compression atoms

Reduce Dead Cycles

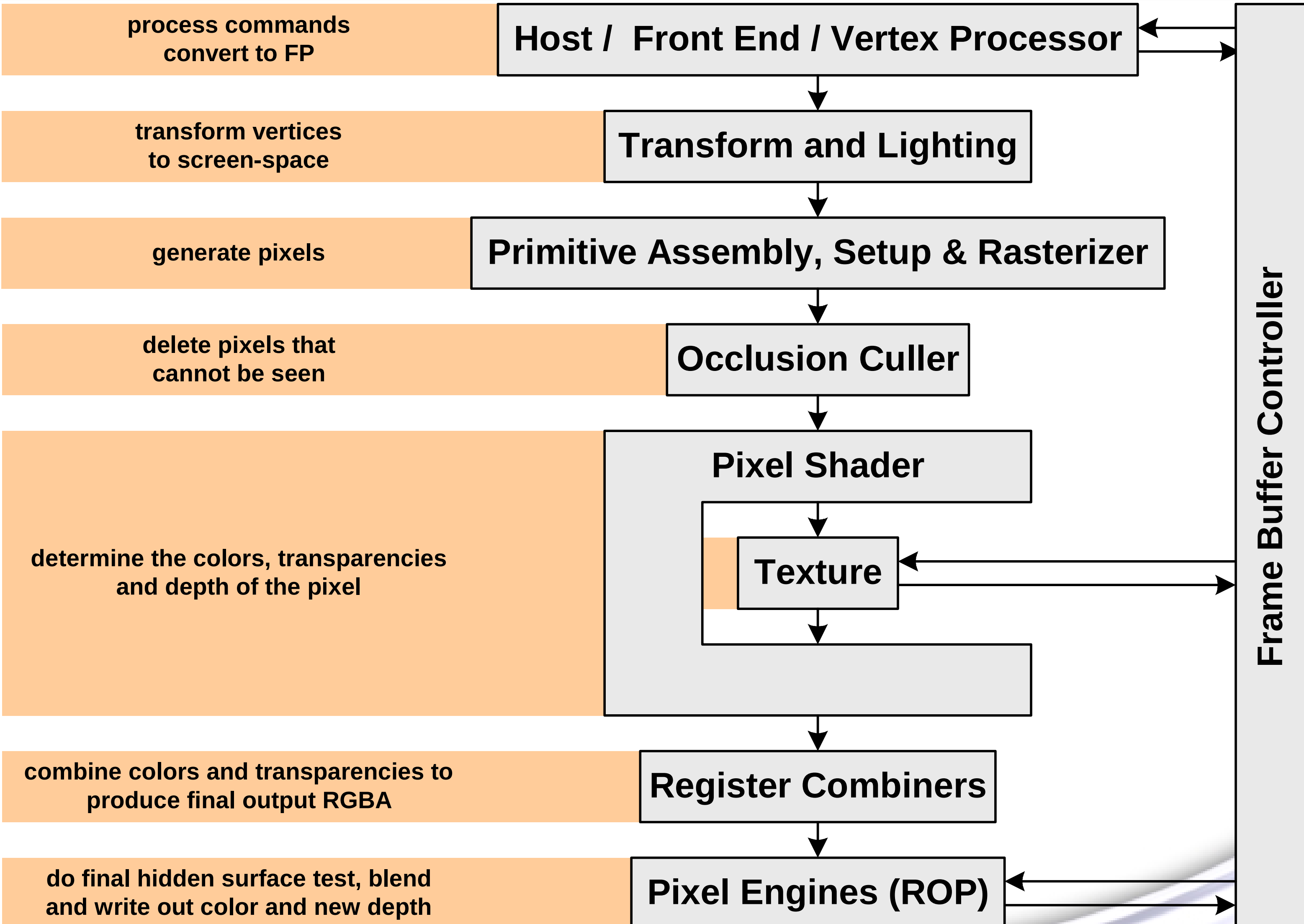
- **Queuing**
  - **Amortize read/write turns over longer runs**
- **Multi-bank DRAM scheduling**

Embedded DRAM

- **Tempting to embed megabytes of DRAM.**
- **But ..**
  - **Cannot fit the whole problem**
  - **Costs are huge**

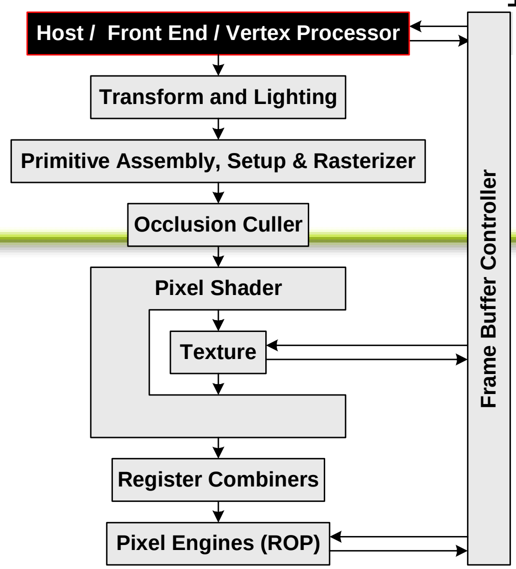
- **Will be "just around the corner" for a long time**

A Tour of the GeForce4



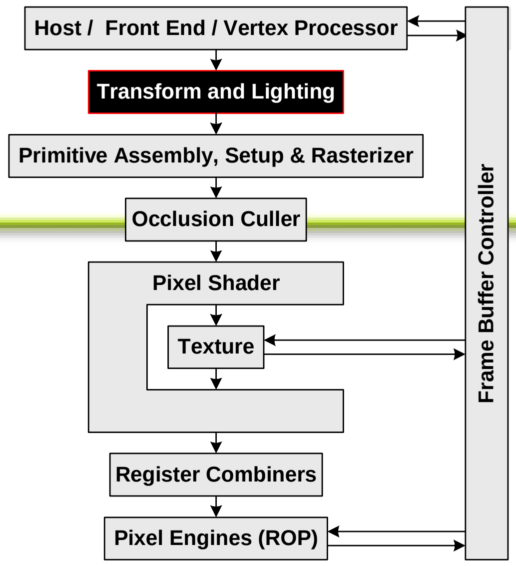
nVIDIA.

Host / Front End / Vertex Processor



- Protocol and physical interface to PCI/AGP
- Command "ABI" interpreter
- Context switch
- DMA gather

Transform and Lighting

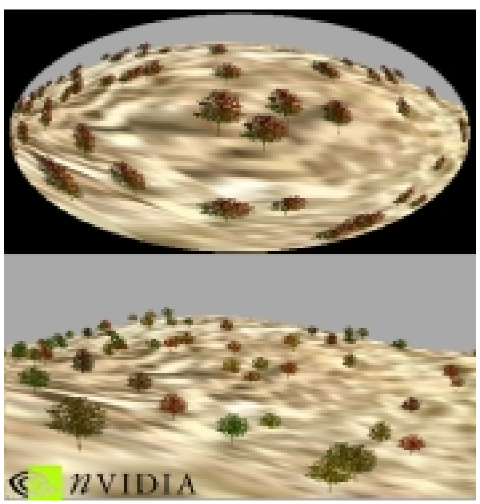


- Handles persistent attributes
- Dispatch
- Hides latency from the programmer
- Fixed-function modes driven by APIs
- Multiple vector floating point processors
  - 256 x 128 context RAM
  - 12 x 128 temp regs
  - 16 x128 input and output

Vertex Program Examples

- Deformation
- Warping
- Procedural Animation

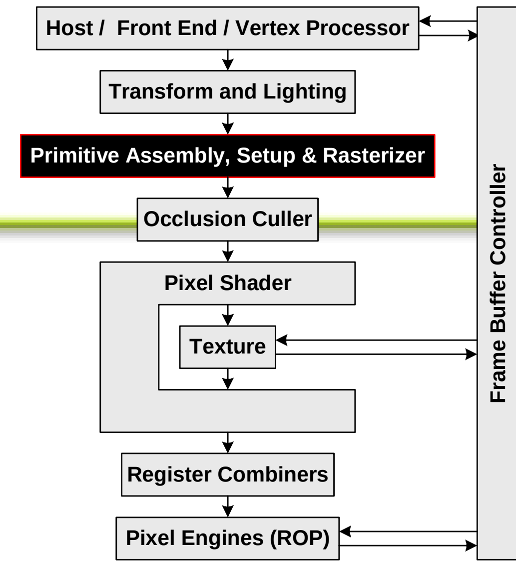
- Range-based Fog
- Elevation-based Fog



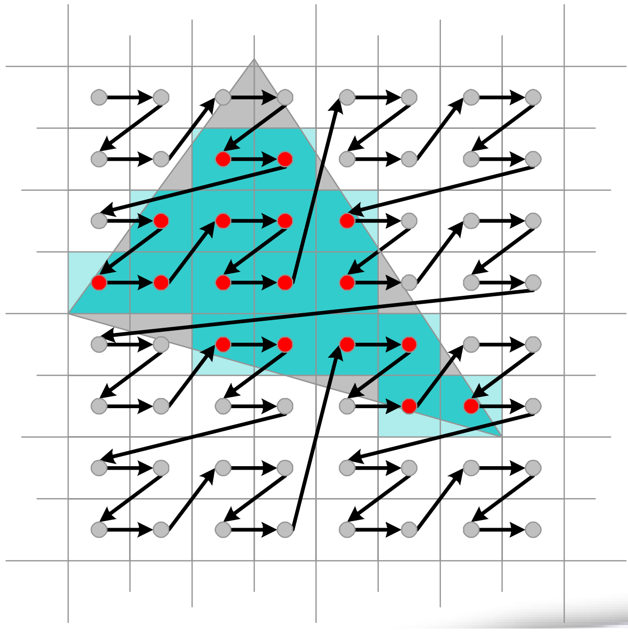
- Lens Effects

- Animation
  - Morphing
  - Interpolation

Primitive Assembly, Setup & Rasterizer



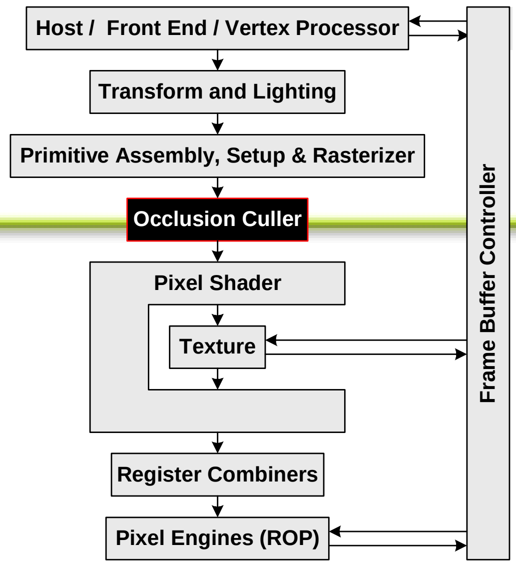
- Per-triangle parameter setup
- Tile walking
- Sample inclusion determination
- Tiles are traversed in memory page friendly order



Occlusion Culling & Programmable Shading

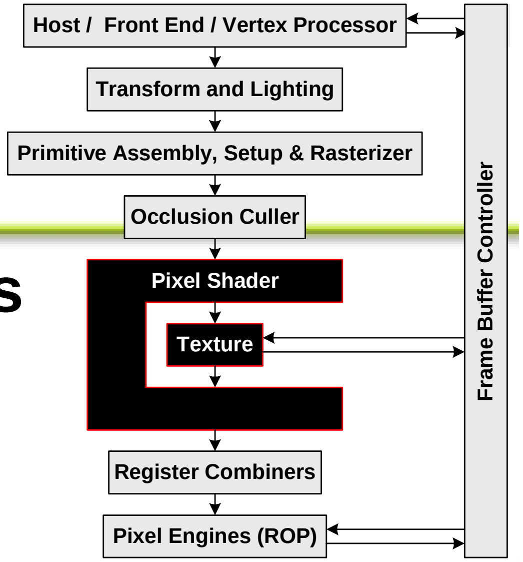
- Occlusion Culling reduces Depth Complexity
  - Calculate Z and determine visible pixels
  - Eliminate invisible pixels
- Programmable Shading enables richer visual quality
  - Accurately model: reflections, shadows, materials
  - More textures/pixel
  - More calculations/pixel – consumes many cycles
- Programmable Shading impractical without Occlusion Culling

Occlusion Strategies



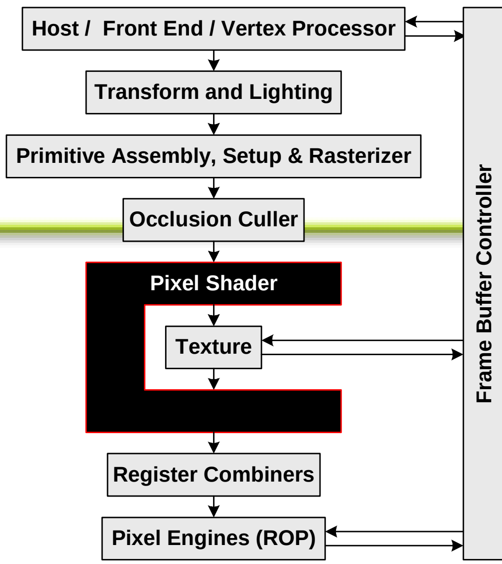
- **Possibilities:**
  - **Maintain local conservative data structure**
  - **Use actual depth buffer data**
  - **Or combine the techniques**
- **A coherence problem no matter how you slice it.**
- **API depth test is at the far end of the pipe!**
  - **Must preserve semantics**

Pixel Shading / Texturing



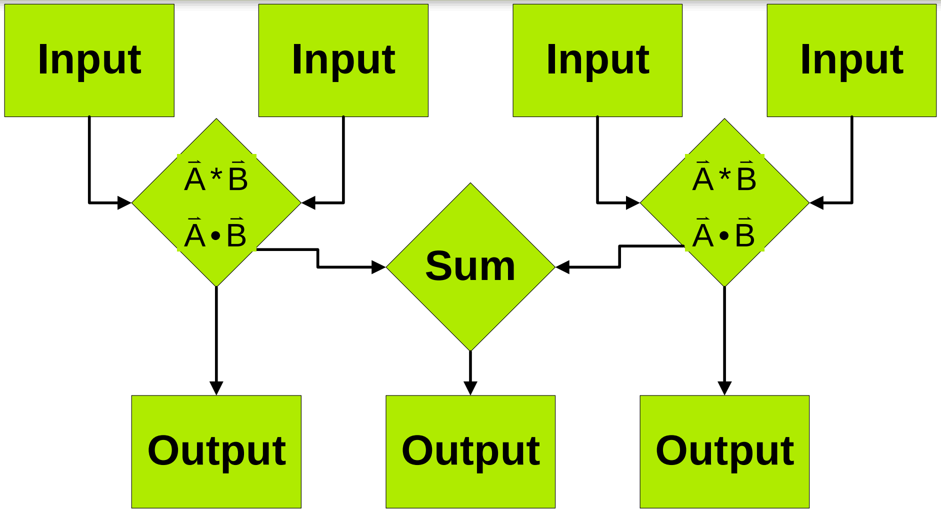
- A pixel shader converts texture coordinates into a color using a shader program.
  - Floating point math
  - Texture lookups
  - Results of previous pixel shaders
- 4 stages, 1 texture address op per stage
  - Compressed, mipmapped 3-D textures
  - True reflective bump mapping
  - True dependent textures (lookup tables)
  - Full 3×3 transform with cubemap or 3-D texture lookup
  - 16-bit-per-component normal maps

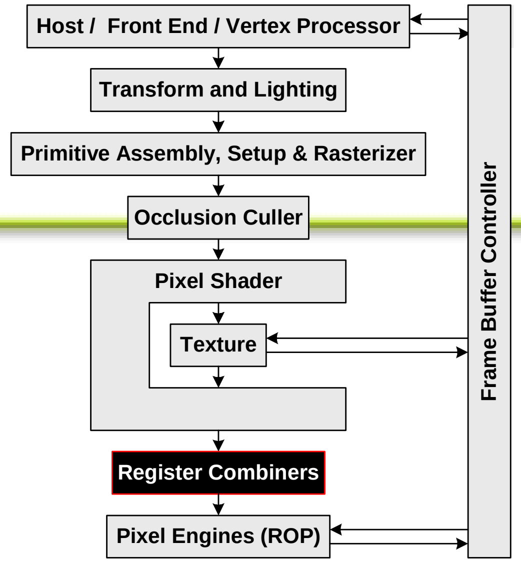
Pixel Shader



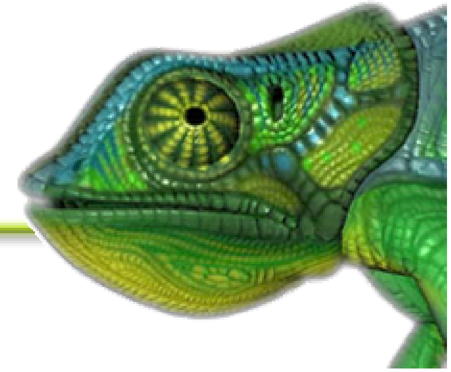
- Input: values interpolated across triangle
- IEEE floating point operations
- Lookup functions using textures
  - Large, multi-dimensional tables
  - Filtered
- Outputs an ARGB value that register combiners can read

Register Combiners

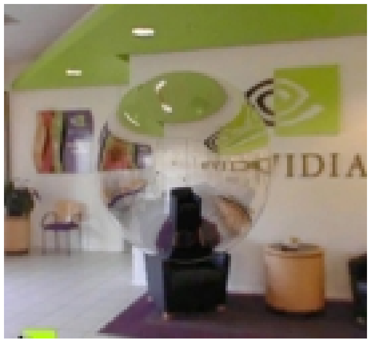




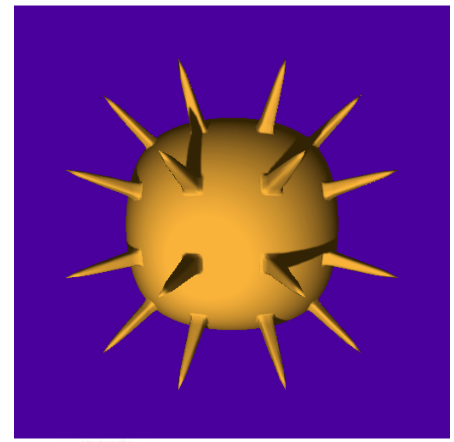
- 1–8 stages, plus a final combiner
- Up to 4 inputs from texture stages, interpolators, constant registers, earlier combiners
- Fixed set of operations:
  - Each stage can evaluate A*B+C*D and output result, along with A*B, C*D
  - Alternatively, each stage can evaluate dot products instead of multiples
  - Can conditionally select A*B or C*D

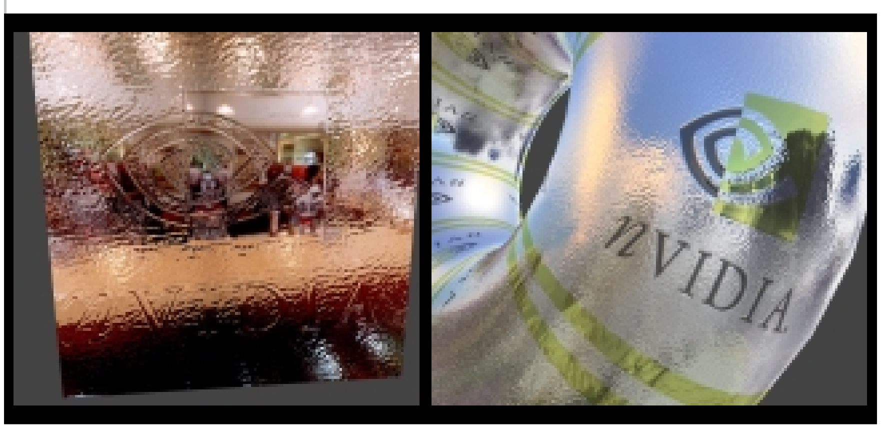


Pixel Shading effects

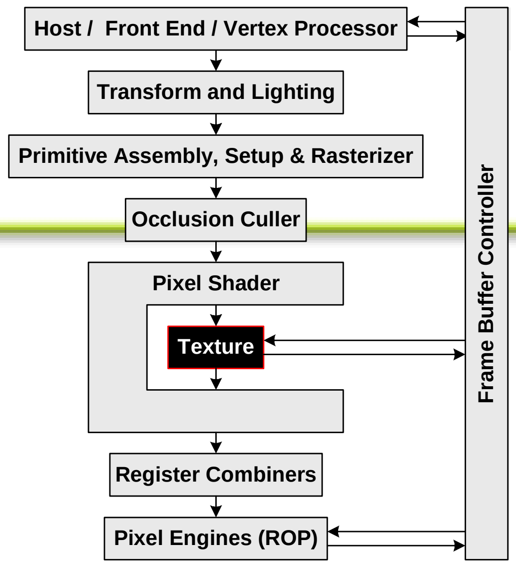


- Multi-texturing
- Dot products for per pixel lighting calculations
- Reflections
- Shadowing
- Custom effects
- Pixel math



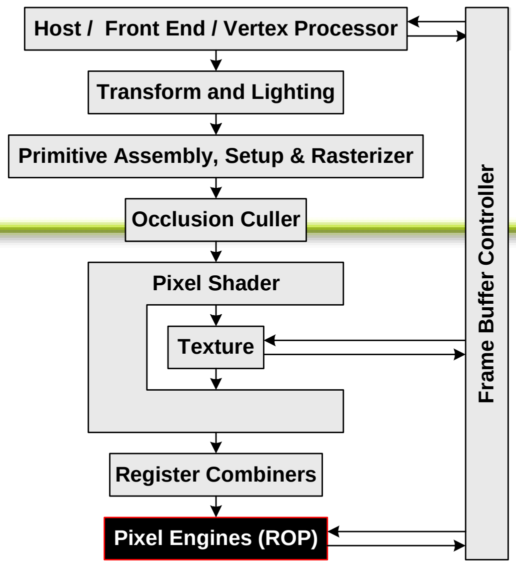


Texture



- Deeply pipelined cache
  - Many hits and misses in flight
- Compression
  - 4:1 ratio
  - Palettes
  - Lossy small-grained fixed ratio scheme
- Filtering
  - Bilinear, tri-linear, 8:1 anisotropic

Pixel Engines (ROP)



- **Coalesces shader pixels into memory access grain**
- **Performs visibility and blending / transparency calculations**
- **Balanced processing power vs. bandwidth**
  - **Bandwidth is amplified by compression**

Multisample Antialiasing

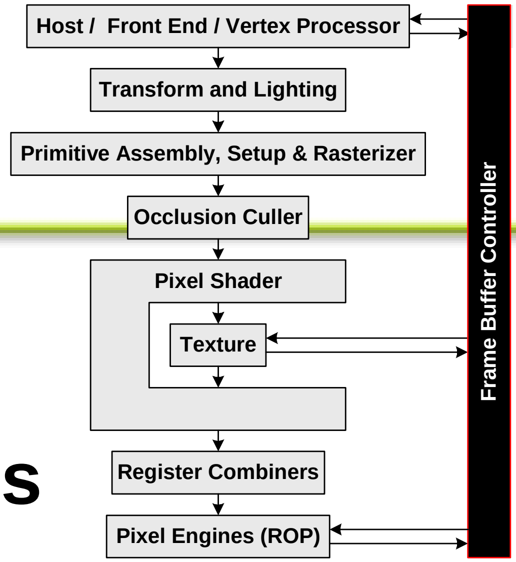
- Transparent to the application
- 2 & 4 subsamples per pixel
- 2, 4, 5, and 9-tap reconstruction filters

nView Display Technology

- **Flexible display combinations**

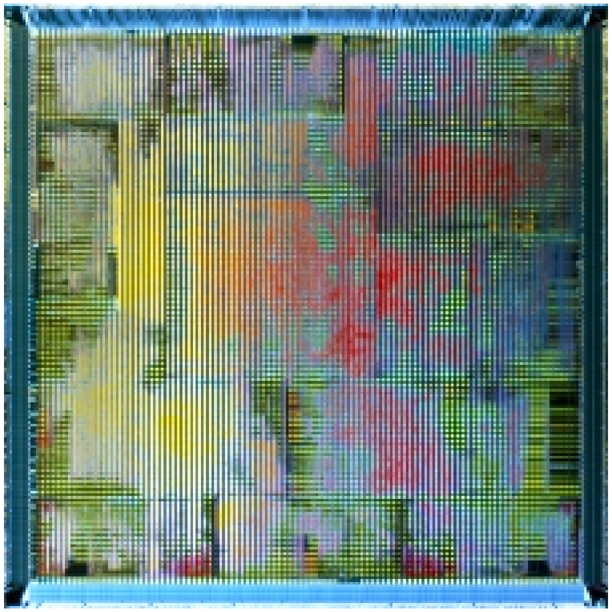
- **Intuitive user interface**
  - **Easy set-up**
  - **Application Management**
  - **Multi Desktop Support**
  - **Window Management**
  - **Window Effects**
  - **Custom User Profiles**

Framebuffer Controller



- 128-pin DDR
- Schedules requests from all engines
- Transparent compress/decompress
- Maps from pixel-linear address to page & partition tiling
- Flexible in:
  - Width
  - Depth
  - Frequency
  - Banks

Statistics



- 136M vertices per second
- 60M triangles per second
- 4.8G samples/sec
- 1.2T ops/sec
- 83.2 GB/sec clear BW
- 63M transistors
- TSMC 0.15u
- 300 MHz pipeline / 325 MHz memory clk

Conclusions

Questions