John Montrym

Henry Moreton

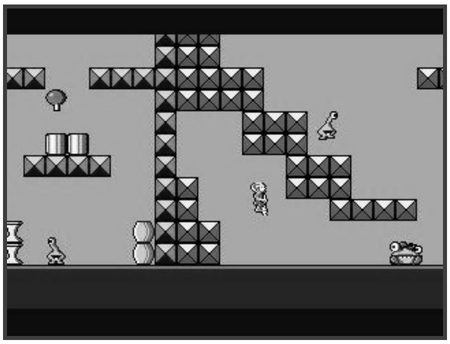
1

GPU Target Applications

(Graphics Processing Unit)

- **Interactive Gaming** (50M units, 10M gamers)
  - Cinematic quality rendering in real time.
- **Digital Content Creation (DCC)** (1M prof, 50M home)
  - Motion Picture and Gaming Industries
- **Computer Aided Design & Manufacturing** (1M)
- **Visual Simulations** (100K units)
  - Pilot / Driver Training
- **General Computing** (a few K sites, 1000s per site)
  - Emerging Area of Research
- **Consumer**
  - handheld(50M), consoles(100M),
    media centers(5M/y), cell phones (600M/y)

Image Quality Evolution



Marooned, 1990

2D gaming

CPU painted pixels BLTs performed by VGA

... First 3D ...

HoverTank3D, 1991

1st 3D PC game

Flat Shading

4

... Doom Series ...

Rudimentary Texturing

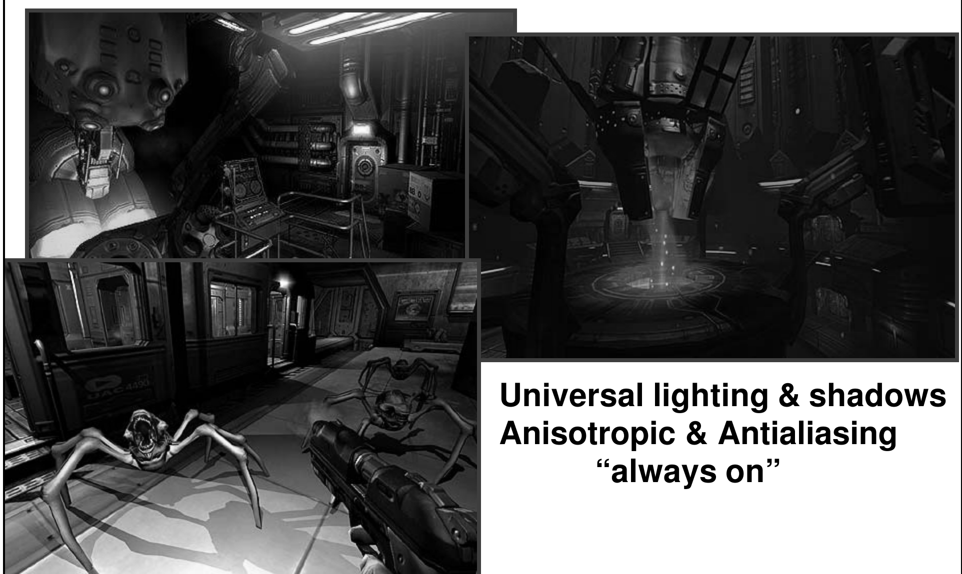
5

... Quake Series ...

Multi-texture &
Color Combiners

6

... *Doom III*, 2004 ...



Universal lighting & shadows
Anisotropic & Antialiasing
"always on"

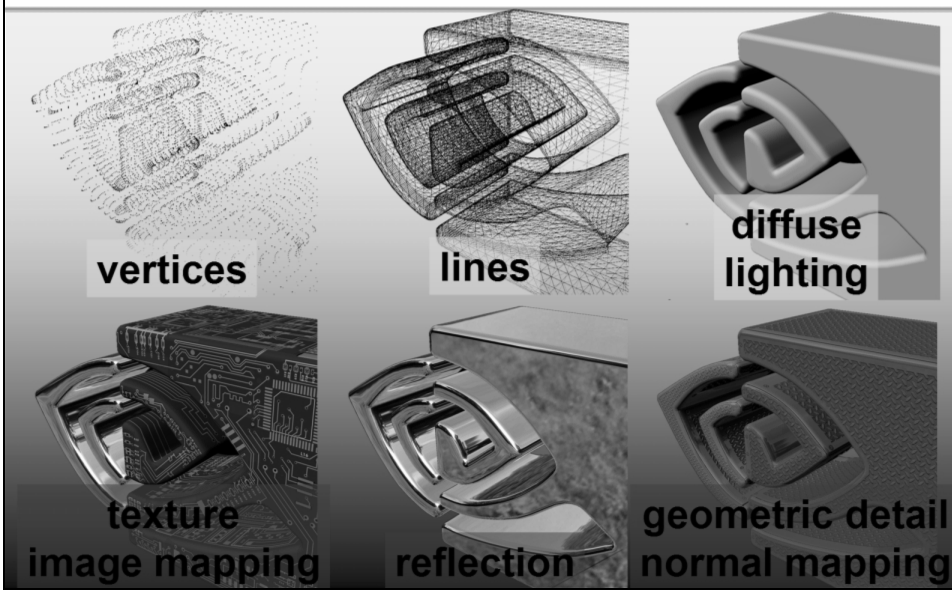
7

Image Synthesis - a sampling problem

- **Scene described by triangles of materials simulated by**
  - **sampled images – textures**
  - **numerically approximated properties**
- **Vertex processing – independent vertex work**
  - **screen position & attributes calculation**
- **Assemble and sample triangles**
  - **generate pixels**
- **Pixel processing – independent pixel work**
  - **texture sampling, color calculation, visibility, and blending**

8

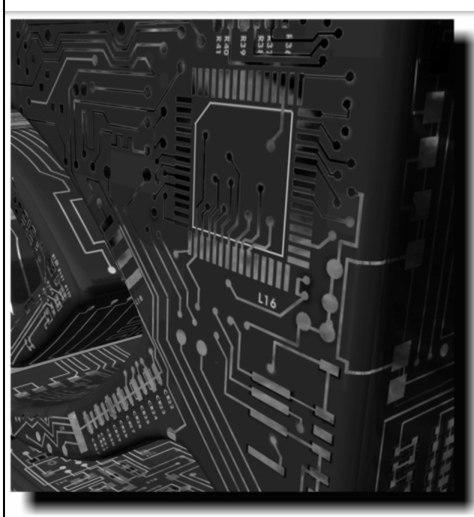
Describing a scene and materials



9

Simple combinations – 2 examples

typically much more complex



reflective traces

diamond plate

nVIDIA.

Fiat Lux – Paul Debevec et al.

SIGGRAPH ’99 Electronic Theatre

- **High Dynamic Range rendering - much greater precision**
- **HDR rendering at work: Light through windows is 100s of times brighter than obelisks, but obelisks aren’t solid black. Glow produces a more cinematic image.**

Typical Rendering Phases

- **Pre-render:**
  - **Shadow maps or shadow volumes for each light source**
  - **Environment maps**
- **Render scene from observer’s viewpoint**
  - **Accumulate contribution from each light source**
- **Post-render:**
  - **Tone-map, flare, depth-of-field**

Light Transport

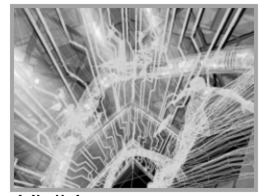
- Inputs: geometry, texture maps, light positions, light radiances, etc.
- Outputs: HDR per-pixel radiance
  - This value can be anything – 1.0×10^{-6} or 9.5×10^{10}
    - Depends on whether scene is a candle-lit cave
    - …or a picnic on the sun.
    - floating point is critical
- Algorithm:
  - Determine which surfaces are visible
  - Compute light reflected (or transmitted) to viewer
  - Based on physics

Light Transport & Color Images

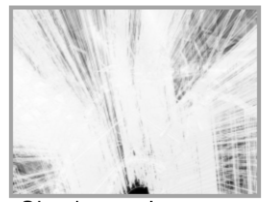
- Light and color are fundamentally different
  - Light Transport: Physics (rendering)
  - Color Images: Perception (image processing)
- High-quality rendering requires handling both
  - Light Transport
    - Huge dynamic range, high precision requirements
    - Multiple lights are additive, objects can be transparent
  - Tone Mapping
    - Compresses dynamic range, but precise
  - Color & Gamma Correction of Images
    - Precision is required near black, but not white

Shadow Volumes Increase Geometric Complexity

Dramatic chase scene with shadows



Visible geometry



Shadow volume geometry

Image courtesy of Contraband Entertainment

Architectural Drivers

- **Programmability**
  - **Evolved from *configurable* logic**
  - **Enables developers' added value**
- **Parallelism**
  - **Independent streams of vertices and pixels**
  - **highly data parallel**
- **Memory Characteristics**

Programmer's Model

- Two separate shader programs
  - Vertex program;
    executed independently on every vertex.
  - Pixel program;
    executed independently on every pixel.
- Streaming model
  - Inputs delivered to read-only registers
  - Outputs written to write-only registers
- Fixed-size thread-private resources
  - Read/write temporary registers
  - Read-only textures
    - resampling & filtering hardware
  - Read-only constant tables

Programmer's Model
new features of GF6800

- Dual data type
  - IEEE FP32 precision & IEEE-like FP16
- Dynamic pixel branching
- Vertex texture
- Floating Point framebuffer blending
- Microsoft's Direct3D
  - VS/PS shader model 3.0
- languages
  - HLSL and Cg compiled to
    GPU-independent assembly
  - JIT compilation to GPU-specific target

18

GPUs vs. CPUs

- Workload offers more independent calculations
  - Enables wide and deep parallelism
- Simple programming model:
  - Multi-threaded machine with single-threaded programming model
  - Superscalar, up to 6 instructions
- Combination of compute resources
  - programmable processors
  - fixed-function processors
    - rasterization
    - texture filtering
    - transcendental functions, inner product

19

Memory Characteristics

- Affordable caches cannot support long-term reuse
- Effective streaming with local reuse
- Supercomputer techniques
  - Latency hiding & vector operations
- Page locality is important
  - 2D&3D access is mapped to 1D DRAM page
  - Minimize read/write turns & page crossings

Framebuffer Controller

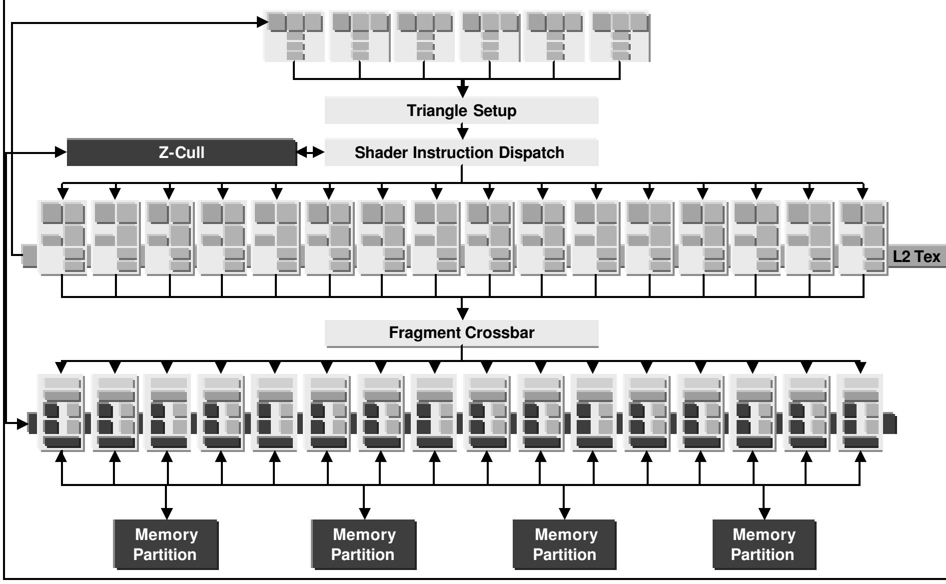
- 256-pin DDR-2 or GDDR-3
- Schedules requests from all engines
- Transparent compress/decompress
  - lossless bandwidth compression
- Maps from pixel-linear address to page & partition tiling
- Flexible in:
  - Width, Depth, Frequency, Banks

Performance Regimes

- No one typical performance mode
- No single balance point
- Seek to maximize usage of the most expensive resource.
- Dozens of regimes that each require "speed of light", limited only by memory bandwidth
  - Z-only (no color) rendering
  - Shadow calculations

22

A Tour of the 6800



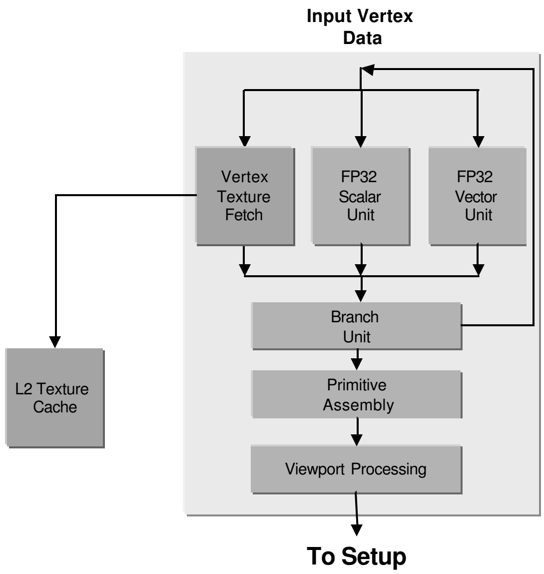
23

Vertex Processing Engine

- ISA compatibility – 2 generations
  - two instruction encodings
  - single merged 123-bit VLIW internal instruction
- Six vector floating point processors
  - 512 x 128 context RAM
  - 32 x 128 temp regs
  - 16 x128 input and output
  - 512 instructions
- Up to 3 threads per processor
  - hides latency from the programmer

24

Vertex Processor Detail



Vertex Processing Engine

- MIMD Architecture
- VLIW issue issues to vector and scalar units
- Penalty free branching
- Shader Model 3.0

Vertex Texture Fetch

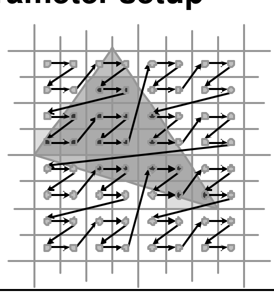
- VPE threads hide latency
- Non-stalling
- Up to 4 textures
- Mip-maps, no filtering

25

Primitive Assembly, Cull Setup & Rasterizer

Fixed function units

- Perspective divide
- Viewport
- Assemble primitives
- Cull
- Per-triangle parameter setup
- Tile walking
- Sample in/out test
- Traversed in page friendly order



26

Pixel Program

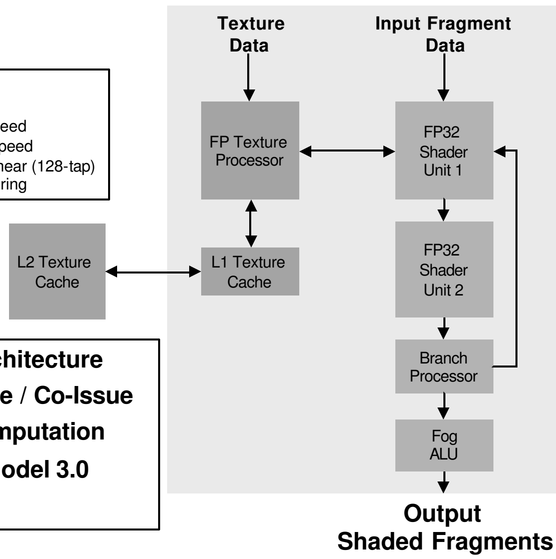
- Program computes resulting color
  - 32-bit Floating point math
  - Texture lookups
  - inputs
    - generic attributes (formerly texture coordinates)
    - colors, Z, fog
  - outputs
    - Multiple Render Targets
- General purpose processor
  - very similar to Vertex Engine
  - constants
  - temporary registers
  - branching...

27

Complex, data-dependent shading

Pixel Processor Detail

<u>Texture Filter</u>
Bi / Tri / Aniso
1 texture @ full speed
4-tap filter @ full speed
16:1 Aniso w/ Trilinear (128-tap)
FP16 Texture Filtering

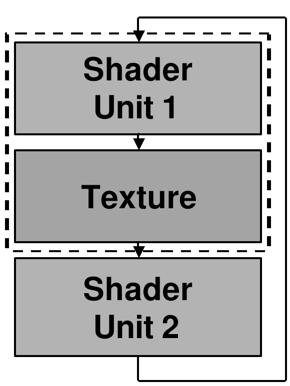


<u>Shader Unit 1</u>
4 FP Ops / pixel
Dual/Co-Issue
Texture Address Calc
Free fp16 normalize
+ mini ALU

<u>Shader Unit 2</u>
4 FP Ops / pixel
Dual/Co-Issue
+ mini ALU

- **SIMD Architecture**
- **Dual Issue / Co-Issue**
- **FP32 Computation**
- **Shader Model 3.0**

Pixel Shader Architecture



4 Components
<u>1 Op / component</u>
4 ops/pixel

and/or

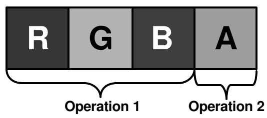
1 Texture/pixel
@ full speed

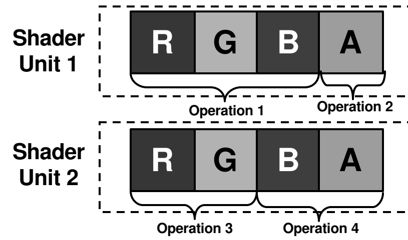
4 Components
<u>1 Op / component</u>
4 ops/pixel

8 Ops/pixel

30

Instruction Processing
Superscalar vs. Split vector





DX9 split vector

- Two independent instructions executing on the same shader unit. (split 4-vector)
- 6800 can group components as 3/1 or 2/2
- **2 instructions/pixel/cycle**

GeForce 6800 Superscalar

- up to 6 instructions executing in the same cycle on different shader units.
- efficient instruction-level parallelism
- **6 instructions/pixel/cycle**

31

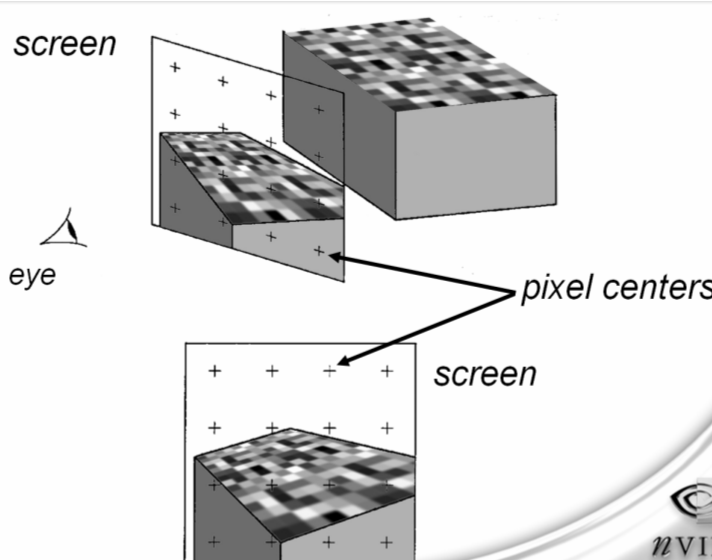
Example Shader Code

```
ps_2_0
def c1, 2.000000, -1.000000, 0.000000, 0.000000
dcl t0.rg
dcl t1
dcl t4.rgb
dcl v0
dcl_2d s0
dcl_2d s1
dcl_cube s2
dcl_2d s3

# clock 1
texld r0, t0, s0;          # tex fetch
madr r0, r0, c1.r, c1.g    # _bx2 in tex
nrm_pp r1.rgb, t4          # nrm in shader 0
dp3 r1.r, r1, r0           # 3D dot product in shader 1
mul r0.a, r0, r0           # dual issue in shader 1

# clock 2
mul r1.a, r0.a, c2.a       # dual issue in shader 0
mul r0.rgb, r1.r, r0       # dual issue in shader 0
add r0.a, r1.r, r1.r       # fx2 in shader 0
mad r0.rg, r0.a, c1, c1.a # mad w/2 const in shader 1
mul r1.ba, r1.a, r0.a, c2  # dual issue in shader 1

# clock 3
 rcp r0.a, r0.a            # reciprocal in shader 0
 mul r0.rg r0, r0.a        # div instruction in shader 0
 mul r0.a, r0.a, r1.a      # dual issue in shader 0
 texld r2, r0, s1          # texture fetch
 mad r2.rgb, r0.a, r2, c5  # mad in shader 1
 abs r0.a, r0.a            # abs in shader 1
 log r0.a, r0.a            # log in shader 1

 # clock 4
 rcp r0.a, t1.a            # reciprocal in shader 0
 mul r0.rg, t1, r0.a       # div instruction in shader 0
 mul r0.a, r0.a, c2.g      # dual issue in shader 0
 texld r1, r0, s3          # tex fetch
 mad r1.rgb, r1, c4, -r2   # mad in shader 1
 exp r0.a, r0.a            # dual issue in shader 1

 # clock 5
 texld r0, r1.bar, s2      # texture coordinates swizzle
 mad r0.rgb, r0, v0, r1    # color calculation in shader 1
 mul r0.a, r1, v0          # dual issue in shader 1

 # clock 6
 mul r1.rgb, r0.a, c5.a    # mul in shader 0
 mad r0.rgb, r1, r0.a, r0  # mad in shader 1
 mov r0.a, c3.a            # move in shader 1
 mov oC0, r0               # move in shader 1
```

Texture

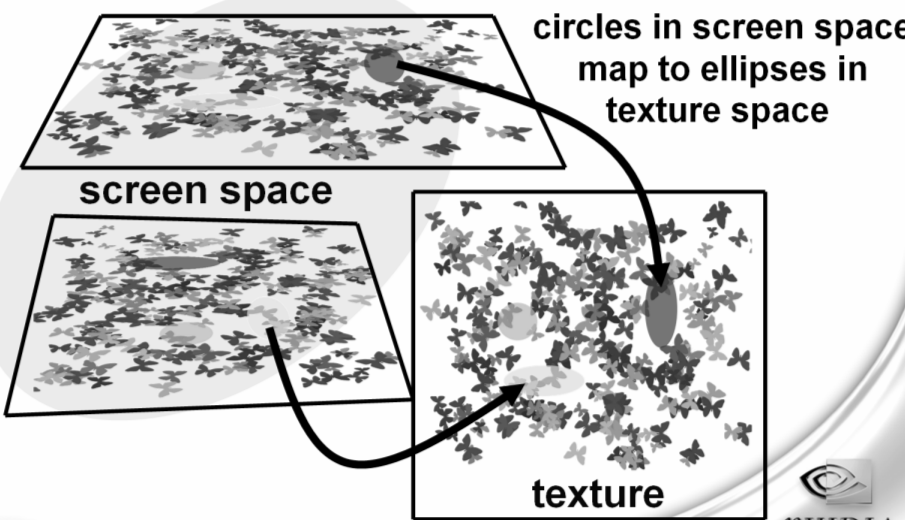
- Deeply pipelined cache
  - Many hits and misses in flight
- Explicit Compression
  - 4:1 ratio
  - Lossy small-grained fixed ratio schemes
- Filtering
  - Bilinear, tri-linear, up to 16:1 anisotropic
  - 4xFP16 texels
- Non-power-of-2 image addressing

Variable sampling density



Anisotropic Footprint

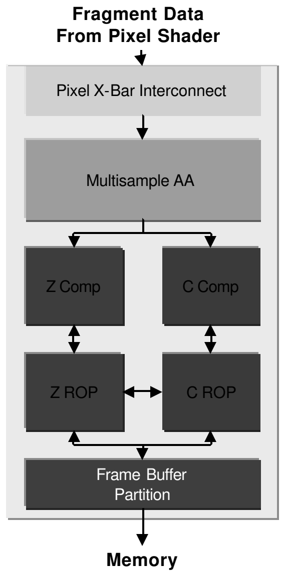
an ellipse



Pixel Engines

- **16 pixels per clock Color & Z**
- **32 pixels per clock Z-only**
- **FP16 x 4 Frame Buffer Blending & Display**
- **Lossless Color & Z-Compression**
- **High Quality Antialiasing (filtering) -** *Rotated Grid*
- **Accelerated Shadow Rendering**

Pixel Pipeline Detail



- FP16 Floating Point Blending
- Double-speed Z
- Lossless Color & Z Compression
- Multiple Render Targets

Anti-aliasing

- Transparent to the application
  - Smooth silhouette and inter-penetration edges
  - Constant color fragments
  - Coverage & depth information per sample
- Multisampling: one color per pixel
- Super-sampling: one color per sample
- 2, 4 or 8 sub-samples per pixel
- Variety of reconstruction filters available

General-Purpose Computing

- Applications
  - NVIDIA Gelato Renderer
  - Protein folding
  - Black-Scholes
  - Image processing
- Dual data types: FP32 & FP16
- Streaming Programming Languages
  - *Brook* (Stanford)
- www.gpgpu.org

Future Directions

- Continue to generalize the programming models
- Virtualization
  - Virtual memory
  - Eliminate fixed limits on intermediate storage

Chip Statistics

- 222 M transistors
- 600 M vertices/sec
- 0.13u IBM process 8LM
- 400+ MHz pipe clock
- 550 MHz DDR mem clock
- 303 mm^2
- >120 GFLOPS peak
  - = 6x a 5GHz Pentium 4

Summary

- Interactive Rendering has reached Cinematic Quality
- The GeForce 6800, built for advanced graphics applications incorporates:
  - A flexible, commodity-to-exotic memory system
  - Rich-featured programmable engines architected for streaming memory
  - Considerable special purpose hardware