Overview

- System architecture
  - Rendering performance
  - GPU architecture
- Unified shader
- Memory Export
- Texture/Vertex Fetch
- HDR rendering
- Displaced subdivision surfaces
System architecture

CPU

2x 10.8 GB/s

2x PCIE
500MB/s

GDDR3

UNIFIED MEMORY

32GB/s

700MHz
128bit

GPU

Northbridge

22.4GB/s

2x 10.8 GB/s

32GB/s

2x 10.8 GB/s

700MHz
128bit

DAUGHTER DIE

10MB EDRAM

500MB/s

2x 10.8 GB/s

2x 10.8 GB/s

128bit

Rendering performance

• Render surface bandwidth

8pix/clk, 4x MSAA
Stencil and Z test
Alpha blending

256GB/s

10MB EDRAM

22.4GB/s
Rendering performance

- GPU to Daughter Die interface
  - 8 pixels/clk
    - 32BPP color
    - 4 samples Z - Lossless compression
  - 16 pixels/clk – Double Z
    - 4 samples Z - Lossless compression
- Alpha and Z logic to EDRAM interface
  - 256GB/s
  - 32 samples x 32bit color, 24bit Z, 8bit stencil
  - Double Z
    - 64 samples x 24bit Z, 8bit stencil

GPU architecture

- Index Stream Generator
- Tessellator
- Primitive Setup
- Clipper
- Rasterizer
- Hierarchical Z/S
- Unified Shader
- Texture/Vertex Fetch
- Output Buffer
- Memory Export
- EDRAM
- UNIFIED MEMORY
- Vertex Pipeline
- Pixel Pipeline
- Display Pixels
Unified Shader

• A revolutionary step in Graphics Hardware
• One hardware design that performs both Vertex and Pixel shaders
• Vertex processing power

Unified Shader

- GPU based vertex and pixel load balancing
  - Better vertex and pixel resource usage
- Union of features
  - E.g. Control flow, indexable constant, ...
- DX9 Shader Model 3.0+
Memory Export

- Shader output to a computed address
- Virtualize shader resources - multipass
- Shader debug
- Scatter write

Randomly update data structures from Vertex or Pixel Shader
- Ray tracing acceleration structures
- Physical simulation – GPGPU
- Enabling exploration for the future
Texture/Vertex Fetch

Shader fetch can be either:
- Texture fetch (16 units)
  - LOD computation
  - Linear, Bi-linear, Tri-linear Filtering
  - Uses cache optimized for 2D, 3D texture data with varying pixel sizes
  - Unified texture cache
- Vertex fetch (16 units)
  - Uses cache optimized for vertex-style data

Texture Arrays

- Generalization of 6 faced cube maps to 64 faces
- Each face is a 2D mip mapped surface
- Not volume texture
- Applications
  - Animation frames
  - Varying skins for instanced characters / objects
  - Character shadow texture flipbook animations
Texture array application:
Unique seeds for instanced shading

Texture array application:
Hundreds of instanced characters
High Dynamic Range Rendering

- Special compact HDR render target format:
  - Just 32 bits: 7e3 7e3 7e3 2
  - Compatible with multisample antialiasing
  - R, G and B are unsigned floating point numbers
    - 7 bits of mantissa
    - 3 bits of exponent
    - Range of 0..16
  - 2 bits of alpha channel
- 16-bit fixed point at half speed
  - With full blending

Displaced subdivision surfaces

- Prototype algorithm
- Vineet Goel, ATI research Orlando
**Displaced subdivision surface algorithm**

- **Tessellator:**
  - Generates 64 vertices for each patch that are fed into the VS.
- **Vertex Shader:**
  - Reads in one-ring, computes Stam’s method using precomputed table lookup
  - Adds Displacement map
- **Pixel Shader**
  - Adds bump mapping and surface color

---

**Displaced subdivision surface results**

![3D models and images showcasing the displaced subdivision surface results.](image)
Demo

- Ruby: The Assassin