Bandwidth –
Gravity of Modern Computer Systems

• The bandwidth between key components ultimately dictates system performance
  – Especially true for massively parallel systems processing massive amount of data
  – Tricks like buffering, reordering, caching can temporarily defy the rules in some cases
  – Ultimately, the performance goes falls back to what the “speeds and feeds” dictate

Interface “feeds and speeds”

• AGP: Advanced Graphics Port – an interface between the computer core logic and the graphics processor
  – AGP 1x: 266 MB/sec – twice as fast as PCI
  – AGP 2x: 533 MB/sec
  – AGP 4x: 1 GB/sec → AGP 8x: 2 GB/sec
  – 256 MB/sec readback from graphics to system

• PCI-E: PCI Express – a faster interface between the computer core logic and the graphics processor
  – PCI-E 1.0: 4 GB/sec each way → 8 GB/sec total
  – PCI-E 2.0: 8 GB/sec each way → 16 GB/sec total

3D Buzzwords

• Fill Rate – how fast the GPU can generate pixels, often a strong predictor for application frame rate

• Performance Metrics
  – Mtris/sec - Triangle Rate
  – Mverts/sec - Vertex Rate
  – Mpixels/sec - Pixel Fill (Write) Rate
  – Mtexels/sec - Texture Fill (Read) Rate
  – Msamples/sec - Antialiasing Fill (Write) Rate
Adding Programmability to the Graphics Pipeline

Specialized Instructions (GeForce 6)

- Dot products
- Exponential instructions:
  - EXP, EXPP, LOG, LOGP
  - LIT (Blinn specular lighting model calculation!)
- Reciprocal instructions:
  - RCP (reciprocal)
  - RSQ (reciprocal square root!)
- Trigonometric functions
  - SIN, COS
- Swizzling (swapping xyzw), write masking (only some xyzw get assigned), and negation is “free”

Easy cross products and normalization

Vector Cross Product
# | i | j | k | into R2.
# | R0.x R0.y R0.z |
# | R1.x R1.y R1.z |
MUL R2, R0.zyxw, R1.yzxw; // swizzle
MAD R2, R0.zyxw, R1.zyxw, -R2; // negation

Vector Normalize
# R1 = (nx,ny,nz)
# | R0.xyz = normalize(R1)
# | R0.w = 1/sqrt(nx^2 + ny^2 + nz^2)
DP3 R0.w, R1, R1;
RSQ R0.w, R0.w; // write-mask
MUL R0.xyz, R1, R0.w; // promotion

Blinn lighting in one instruction

\[
\text{LIT \hspace{1cm} d, s} \\
\hspace{1cm} s.x = N \cdot L \\
\hspace{1cm} s.y = N \cdot H \\
\hspace{1cm} s.z = s \hspace{2.5cm} (-128 < m < 128) \\
\hspace{1cm} d.x = 1.0 \\
\hspace{1cm} d.y = \text{CLAMP}(N \cdot L, 0, 1) \\
\hspace{1cm} d.z = \text{CLAMP}(N \cdot H, 0, 1)^s \\
\hspace{1cm} d.w = 1.0
\]

From Stanford CS448A: Real-Time Graphics Architectures
See graphics.stanford.edu/courses/cs448a-01-fall
**Simple graphics pipeline**

- From Stanford CS448A: Real-Time Graphics Architectures
- See graphics.stanford.edu/courses/cs448a-01-fall

---

**Vertex Cache**
- Temporary store for vertices, used to gain higher efficiency
- Re-using vertices between primitives saves AGP/PCI-E bus bandwidth
- Re-using vertices between primitives saves GPU computational resources
- A vertex cache attempts to exploit "commonality" between triangles to generate vertex reuse
- Unfortunately, many applications do not use efficient triangular ordering

---

**Texture Cache**
- Stores temporally local texel values to reduce bandwidth requirements
- Due to nature of texture filtering high degrees of efficiency are possible
- Efficient texture caches can achieve 75% or better hit rates
- Reduces texture (memory) bandwidth by a factor of four for bilinear filtering
Built-in Texture Filtering (GeForce 6)

- **Pixel texturing**
  - Hardware supports 2D, 3D, and cube map
  - Non power-of-2 textures OK
  - Hardware handles addressing and interpolation for you
    - Bilinear, trilinear (3D or mipmap), anisotropic

- **Vertex texturing**
  - Vertex processors can access texture memory too
  - Only nearest-neighbor filtering supported in G60 hardware

ROP (from Raster Operations)

- **C-ROP** performs frame buffer blending
  - Combinations of colors and transparency
  - Antialiasing
  - Read/Modify/Write the Color Buffer

- **Z-ROP** performs the Z operations
  - Determine the visible pixels
  - Discard the occluded pixels
  - Read/Modify/Write the Z-Buffer

- **ROP on GeForce** also performs
  - “Coalescing” of transactions
  - Z-Buffer compression/decompression

The Frame Buffer

- The primary determinant of graphics performance other than the GPU
- The most expensive component of a graphics product other than the GPU
- Memory bandwidth is the key
- Frame buffer size also determines
  - Local texture storage
  - Maximum resolutions
  - Anisotropic resolution limits

Frame Buffer Interface (FBI)

- Manages reading from and writing to frame buffer
- Perhaps the most performance-critical component of a GPU
- GeForce’s FBI is a crossbar
- Independent memory controllers for 4+ independent memory banks for more efficient access to frame buffer
GeForce 7800 GTX Board Details

- SLI Connector
- Single slot cooling
- sVideo TV Out
- DVI x 2
- 16x PCI-Express
- 256MB/256-bit DDR3
- 600 MHz
- 8 pieces of 8Mx32
- 16x PCI-Express

NVIDIA 7800 GTX - Pixel Processors

- 8 MADD (multiply/add) instructions in a single cycle
- 7800 GTX has 24 of these!

NVIDIA 7800 GTX - Vertex Processors

- 7800 GTX has 8 of these!