CS4803DGC Design Game Consoles

Spring 2009
Prof. Hyesoon Kim
• Each stage can be also pipelined
• The slowest of the pipeline stage determines the rendering speed.
• Frames per second (fps)
Application Stage

- Executes on the CPU
- Collision detection – may provide the feedback
- Global acceleration algorithms, etc
- Generate rendering primitives, points, lines, triangles ..
- Input from other sources (keyboard, mouse..)
Geometry stage

- The majority of the per-polygon and per-vertex operations (Floating point operations)
- Intel’s MMX/SSE
- Old time: Software implementation.
- Move objects (matrix multiplication)
- Move the camera (matrix multiplication)
- Computing lighting at vertices of triangles (Vertex Shading)
- Project onto screen (Projection)
- Clipping (avoid triangles outside screen)
- Map to window
What’s a Vertex?

• The defining “corners” of a primitive
• Often means a triangle
The RASTERIZER stage

- From GEOMETRY to visible pixels on screen
- Add textures and various other per-pixel operations
- And visibility is resolved here: sorts the primitives in the z-direction
- Per pixel operation
- Mostly integer operations
Application → Geometry → Rasterizer → Frame Buffer
Color Framebuffer

- 2D array of R,G,B color pixel values
- 8 bits (256 levels) per color component
- Three 8-bit components can represent 16 million different colors, including 256 shades of gray
- 4th component: alpha; used for blending
Interfaces between CPU and GPU

- **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
Generation I: 3dfx Voodoo (1996)

- One of the first true 3D game cards
- Worked by supplementing standard 2D video card.
- Did not do vertex transformations: these were done in the CPU
- Did do texture mapping, z-buffering.

http://acceleration.com/?ac.id.123.2
Main innovation: shifting the transformation and lighting calculations to the GPU
- Allowed multi-texturing: giving bump maps, light maps, and others..
- Faster AGP bus instead of PCI

- For the first time, allowed limited amount of programmability in the vertex pipeline
- Also allowed volume texturing and multi-sampling (for antialiasing)
**Generation IV: Radeon 9700/GeForce FX (2002)**

- This generation is the first generation of fully-programmable graphics cards
- Different versions have different resource limits on fragment/vertex programs

![Diagram of the processing pipeline](http://acelenation.com/?ac.id.123.8)

- Vertex Transforms
- Primitive Assembly
- Rasterization and Interpolation
- Raster Operations
- Frame Buffer
- AGP
- Programmable Vertex shader
- Programmable Fragment Processor

[http://acelenation.com/?ac.id.123.8](http://acelenation.com/?ac.id.123.8)
Not exactly a quantum leap, but…

- Simultaneous rendering to multiple buffers
- True conditionals and loops
- Higher precision throughput in the pipeline (64 bits end-to-end, compared to 32 bits earlier.)
- PCIe bus
- More memory/program length/texture accesses
NVIDIA GeForce 7800 Pipeline

Block diagram of the G70 architecture. Source: NVIDIA.
16 highly threaded SM’s, >128 FPU’s, 367 GFLOPS, 768 MB DRAM, 86.4 GB/S Mem BW, 4GB/S BW to CPU
• Xbox 360: Unified shader (ATI/AMD)
• Playstation 3: a modified version of GeForce 78000 (NVIDIA)
• Cuda: unified shader (NVIDIA)
The GEOMETRY stage in more detail

- **The model transform**
- Originally, an object is in "model space"
- Move, orient, and transform geometrical objects into "world space"
- Example, a sphere is defined with origin at (0,0,0) with radius 1
  - Translate, rotate, scale to make it appear elsewhere
- Done per vertex with a 4x4 matrix multiplication!
- The user can apply different matrices over time to animate objects
The view transform

- You can move the camera in the same manner
- But apply inverse transform to objects, so that camera looks down negative z-axis
GEOMETRY Lighting

- Compute “lighting” at vertices

- Try to mimic how light in nature behaves
  - Empirical models and some real theory
Two major ways to do it

- Orthogonal (useful in few applications)
- Perspective (most often used)
  - Mimics how humans perceive the world, i.e., objects’ apparent size decreases with distance
GEOMETRY

Projection

- Also done with a matrix multiplication!
- Pinhole camera (left), analog used in CG (right)
Clipping and Screen Mapping

- Square (cube) after projection
- Clip primitives to square

- Screen mapping, scales and translates square so that it ends up in a rendering window
- These "screen space coordinates" together with Z (depth) are sent to the rasterizer stage
GEOMETRY

Summary

model space -> world space -> world space -> camera space

compute lighting -> projection image space -> clip -> map to screen

Tomas Akenine-Möller © 2002
The RASTERIZER in more detail

• Scan-conversion
  – Find out which pixels are inside the primitive
• Texturing
  – Put images on triangles
• Interpolation over triangle
• Z-buffering
  – Make sure that what is visible from the camera really is displayed
• Double buffering
• And more…
The RASTERIZER
Scan conversion

• Triangle vertices from GEOMETRY is input
• Find pixels inside the triangle
  – Or on a line, or on a point
• Do per-pixel operations on these pixels:
  – Interpolation
  – Texturing
  – Z-buffering
  – And more…
The RASTERIZER
Interpolation

- Interpolate colors over the triangle
  - Called Gouraud interpolation

blue
red
green
The RASTERIZER

Texturing

• One application of texturing is to "glue" images onto geometrical object
• Associate points in an image to points in a geometric object
Examples

1. Without texture mapping
2. With texture mapping

From wikipedia
Another Example: Bump mapping

From wikipedia
The RASTERIZER buffering

- The fixed graphics hardware "just" draws triangles
- However, a triangle that is covered by a more closely located triangle should not be visible
- Assume two equally large tris at different depths

Triangle 1 | Triangle 2 | Draw 1 then 2 | Draw 2 then 1
Tomas Akenine-Möller © 2002
The RASTERIZER
Z-buffering

- Would be nice to avoid sorting...
- The Z-buffer (aka depth buffer) solves this
- Idea:
  - Store z (depth) at each pixel
  - When scan-converting a triangle, compute z at each pixel on triangle
  - Compare triangle’s z to Z-buffer z-value
  - If triangle’s z is smaller, then replace Z-buffer and color buffer
  - Else do nothing
- Can render in any order
The RASTERIZER
Double buffering

- The monitor displays one image at a time
- So if we render the next image to screen, then rendered primitives pop up
- And even worse, we often clear the screen before generating a new image
- A better solution is ”double buffering”
The RASTERIZER

Double buffering

• Use two buffers: one front and one back
• The front buffer is displayed
• The back buffer is rendered to
• When new image has been created in back buffer, swap front and back
Anti-Aliasing

- Aliased rendering: color sample at pixel center is the color of the whole pixel
- Anti-aliasing accounts for the contribution of all the primitives that intersect the pixel

Triangle Geometry  Aliased  Anti-Aliased