Advanced Raster Graphics
(and other ideas for projects)

Mike Houston (AMD)
Aaron Lefohn (Intel)
Disclaimer

• The examples in this talk do not cover all work in the field---merely a representative subset

• There are relatively few examples in the literature now, but many more this year than last
  – The floodgates are opening...
Advanced Image Processing

• (i.e., Fancy blurry things)
  – Glossy reflections
  – Depth-of-field

• Examples
  – Summed area tables
  – Infinite impulse response (IIR) filters
Dynamic Summed Area Tables

“Fast Summed-Area Table Generation and its Applications,”
Hensley et al., Eurographics 2005
Summed Area Tables

• Goal
  – Filter over arbitrary rectangular regions of a texture in constant time
  – More flexible variable-width filtering than mipmaps
    – Non-square filter regions
    – Filter region sizes not limited to power-of-two LODs

• Idea
  – Create a texture where each texel contains the sum of all elements above and to the left of the original texture
  – Subtract SAT value at lower-right corner from SAT value at upper-left and divide by area

“Summed-area tables for texture mapping,” Crow, SIGGRAPH 1984
Dynamic Summed Area Tables

• Algorithm overview
  – Generate dynamic texture, reflection map, etc.
  – Generate SAT with data-parallel GPGPU computation
  – Use SAT data structure in traditional rendering pass

• Applications
  – Glossy reflections
    – Blurriness depends on distance of reflected object
  – Depth-of-field
    – Blurriness depends on distance from eye
Dynamic Summed Area Tables

• Implementation
  – Step-efficient 2D parallel-prefix “scan” operation

Figure from “Summed Summed-Area Tables Area Tables And Their Application to Dynamic Glossy Environment Reflections,” Scheuermann, Game Developer’s Conference, 2005
Dynamic Summed Area Tables

Glossy reflection

Depth-of-field

*Images from “Fast Summed-Area Table Generation and its Applications,” Hensley et al., Eurographics 2005*
Infinite Impulse Response Filters

• Another method for constant-time, spatially-varying filters
  – Can produce arbitrarily wide blurs in constant cost
  – Also called “recursive filters”

Infinite Impulse Response Filters

• Idea
  – Use value and filtering result of adjacent pixel to determine filtered value of current pixel
    – Enables information to travel across entire image
    – Requires communication between pixels (!)

• Implementation
  – Process each row / column sequentially
    – “Image Processing Tricks in OpenGL,” Green, GDC, 2005
    – Work-efficient parallel-prefix (scan) operator
      – Lefohn, Ph.D. Dissertation, 2006
Work-Efficient Scan Operator

Figure courtesy of Shubho Sengupta at UC Davis
IIR for Depth-of-Field?

• Idea
  – Cast depth-of-field blur problem in terms of anisotropic heat equation
  – Input image is “initial heat distribution”
  – Define “material model” based on CoC
  – Obtain DOF result by allowing heat to diffuse

• Implementation
  – Solve heat equation with separable implicit solver
  – Implicit solver equivalent to IIR filter
    – Build and solve 1000s of tridiagonal linear systems

“Interactive Depth of Field Using Simulated Diffusion on a GPU,”
IIR Depth-of-Field

IIR Depth-of-Field

*Image from “Interactive Depth of Field Using Simulated Diffusion on a GPU,” Kass et al., Pixar Technical Report #06-01, Jan. 2006*
IIR Depth-of-Field

IIR Depth-of-Field

Interactive Cinematic Lighting

• Idea
  – Use offline renderer to compute most of frame
  – Use GPU for “the last mile” to interactively compute lighting
    – Huge win for CG lighting artists
    – Alternative is to re-render entire frame for every parameter change (can takes hours)
  – Increasingly heavy use of GPGPU algorithms to support more complex lighting models and bring interactive quality closer to offline quality
Lpics

• RenderMan computes deep frame buffers
• Use data-parallel “map” operation to reconstruct and light scene
  – Interactive frame rates for 100s of lights, changing one light at a time
  – Works for only direct lighting. Limited anti-aliasing, no motion blur, no transparency


Lpics render $\approx 0.1s$

Final render $\approx 2000s$
Lightspeed

• Compute surface shaders in RenderMan
  – Save all shading (surface) points (much larger than deep frame buffer)
  – Compress data

• Compute light shaders interactively on GPU
  – Execute light shaders on compressed shading samples (parallel ‘map’)
  – Create image by scattering samples to pixel positions and resolve/combine
    – Anti-aliasing, motion blur, depth-of-field

*Images from “The Lightspeed Automatic Interactive Lighting Preview System,”
Ragan-Kelley et al., SIGGRAPH 2007*
Direct-to-Indirect Transfer

• Interactive relighting including multi-bounce global illumination

*Images from “Direct-to-Indirect Transfer For Cinematic Relighting, “Hasan et al., ACM SIGGRAPH 2006*
Direct-to-Indirect Transfer

• Idea
  – Pre-compute surface shading result at sample points distributed in 3D space
  – Compress results into wavelet representation
  – Relight entire on GPU
    – GPU wavelet projections
    – GPU sparse matrix multiplication
    – GPU “map” to compute direct illumination
Direct-to-Indirect Transfer

Images from "Direct-to-Indirect Transfer For Cinematic Relighting, "Hasan et al., ACM SIGGRAPH 2006"
Shadows

• “GPGPU shadows? Isn’t that just rendering?”
  – GPGPU makes it possible to perform per-frame, full-scene analysis of shadow requirements
  – GPGPU algorithms and data structures result in much higher quality shadows than are otherwise possible
Resolution-Matched Shadow Maps

“Build a $32,768^2$ quadtree shadow map from scratch each frame”

“Resolution-Matched Shadow Maps,”
Lefohn et al., ACM Transactions on Graphics, 2007
Shadow Map Overview

• Williams, 1978
  – Depth image rendered from the light position
Shadow Map Overview

• Shadow lookup
Quadtree Virtual Domain

- Shadow map coordinates
Quadtree Address Translator

• Paged 2D texture memory
• Mipmapped page table
Quadtree Shadow Maps
Adaptive Soft Shadows
Adaptive Visibility Sampling

- Render sparse image-space samples
- Categorize samples into occluded, lit, or penumbra and pack into bins
- Compute visibility for each penumbra pixels
- Unpack visibility results into image space

*Image from “High-Quality Adaptive Soft Shadow Mapping,” Guennebaud et al., Eurographics 2007*
Reconstruct Visibility Buffer

• Push-pull reconstruction
  – Respect edges based on normals and depth values
  – Edge detection algorithm (e.g., Sobel)

*Image from “High-Quality Adaptive Soft Shadow Mapping,” Guennebaud et al., Eurographics 2007*
Adaptive Soft Shadows

• Complex data movement required to perform efficient data-parallel conditional computation
  – Implemented with “stream-out” feature of geometry shader
  – A fast “segmented scan” operator* may be a more efficient / general way to perform the operations

*See “Scan Primitives for GPU Computing,” Sengupta et al., Graphics Hardware, 2007
Other Example

• Ambient Occlusion
  – “Dynamic Ambient Occlusion and Indirect Lighting,” Bunnell, GPU Gems II, 2005
  
  – Compute ambient occlusion by traversing hierarchical trees on GPU
“Fast Summed-Area Table Generation and its Applications,”
Hensley et al., Eurographics 2005

“Dynamic Ambient Occlusion and Indirect Lighting,” Bunnell, GPU Gems II, 2005

“Real-Time Approximate Sorting for Self Shadowing and Transparency in Hair Rendering,”
Sintorn et al., I3D 2008

“Resolution Matched Shadow Maps,”
Lefohn et al., ACM Transactions on Graphics 2007
OptiX, NVIDIA 2008

"FreePipe: a Programmable Parallel Rendering Architecture for Efficient Multi-Fragment Effects,”
Liu et al., ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games 2010

"RenderAnts: Interactive Reyes Rendering on GPUs,”
Zhou et al., ACM SIGGRAPH Asia 2009

"Hardware-Accelerated Global Illumination by Image Space Photon Mapping,”
McGuire and Luebke, High Performance Graphics 2009
Order Independent Transparency
Braided Parallelism

- Intermixed task-, data-, and pipeline parallelism

"Braid: Integrating Task and Data Parallelism," West and Grimshaw, 1994

Image from Johan Andersson, DICE
Deferred Lighting (DICE)
Lot’s more stuff (just a short list)

• Hybrid Photon Mapping:  http://graphics.cs.williams.edu/papers/PhotonHPG09/

• Timo Aila's page is a good source of inspiration:  http://www.tml.tkk.fi/~timo/
  – See his HPG 2009 paper and source code for a persistent threads example.

• Efficient sparse voxel trees (Samuli):  http://www.tml.tkk.fi/~samuli/

• Free pipe:  http://sites.google.com/site/hmcen0921/cudarasterizer

• Crytek Real-Time GI:  http://www.crytek.com/technology/presentations/

• Stochastic Transparency:  http://enderton.org/eric/

• Eikonal rendering:
  http://www.mpi-inf.mpg.de/resources/EikonalRendering/index.html

• Irregular Z buffer:
  http://www.cse.chalmers.se/~uffe/softshadows2008EGSR.pdf
General areas to keep in mind

• Lighting
• Sampling
• Rich detail
• Transparency
• Braided parallelism
Draft Proposals Due Next Thursday by end of day! (4/29)