Looking Back, Looking Forward, Why and How is Interactive Rendering Changing

Mike Houston
AMD
Welcome to
Beyond Programmable Shading!
A quick history 1960’s to 2000s

1960’s
- ISutherland, Sketchpad, 1961
- S Russel, Spacewar, 1961
- IBM2250, 1965
- Odyssey, Magnavox, 1966-8
- U Utah, 1968
- Hidden surface removal, UU, 1969
- E&S, 1968
- Intel, 1968
- AMD, 1969

1970’s
- Gouraud, 1971
- Pong, Atari, 1972
- “Westworld” PDG, 1973
- Siggraph ACM, 1973
- NYIT, 1974
- Apple II, Apple, 1977
- Ikonas, 1978
- Lucas CGD, 1979

1980’s
- RayTracing, Whitted, 1980
- TRON, Disney 3I, MAGI, NYIT, 1980
- REYES, Lucas, 1981
- SGI, J. Clark, 1982
- Wavefront, 1984
- Radiosity, Cornell, 1984
- 1st MAC wGUI, Apple, 1984
- ATI, 1985
- Pixar, Lucas, 1986
- RenderMan, Pixar, 1988

1990’s
- Win 3.0 wGUI, MS, 1990
- OpenGL 1.0, SGI, 1991
- Toy Story, Pixar, 1992
- Reality Engine, SGI, 1993
- Playstation, Sony, 1995
- Nintendo 64, N, 1996
- Quake, ID, 1996
- Voodoo3D, 3Dfx, 1997
- TNT2/GeF256 NV, 1999
- ArtX, 1997

Interactive rendering techniques are created using an inseparable mix of data- and task-parallel algorithms and graphics pipelines.
How do users write new interactive 3D rendering algorithms?
Fixed Function Pipelines (DX7)

• Writing new rendering algorithms means
  – Tricks with multitexture, stencil buffer, depth buffer, blending …

• Examples
  – Stencil shadow volumes
  – Hidden line removal
  – …
Programmable Shaders (DX8-10)

• Writing new rendering algorithms means
  – Tricks with stencil buffer, depth buffer, blending …
  – Plus: Writing shaders

• Examples
  – User-defined materials
  – User-defined lights
  – User-defined data structures (built in texture memory)
Software Graphics: Part I (DX11)

• Writing new rendering algorithms means
  – Tricks with stencil buffer, depth buffer, blending …
  – Plus: Writing shaders
  – Plus: Writing data- and task-parallel algorithms
    • Analyze results of rendering pipeline
    • Synthesize data structures

• Examples
  – Dynamic summed area table
  – Dynamic quadtree adaptive shadow map
  – Dynamic histogram-analysis shadow map
  – Dynamic ambient occlusion
  – …
“Fast Summed-Area Table Generation and its Applications,”
Hensley et al., Eurographics 2005

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

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

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

Beyond Programmable Shading Course, ACM SIGGRAPH 2010
Software Graphics: Part II (DX11+)

• Writing new rendering algorithms means
  – Tricks with stencil buffer, depth buffer, blending …
  – Plus: Writing shaders
  – Plus: Writing data- and task-parallel algorithms
    • Analyze results of rendering pipeline
    • Synthesize data structures
  – Plus: Creating new and extended rendering pipelines

• Examples
  – Micropolygon rendering
  – Ray tracing pipelines
  – …
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

OptiX, NVIDIA 2008

Hardware-Accelerated Global Illumination by Image Space Photon Mapping,
McGuire and Luebke, High Performance Graphics 2009

RenderAnts
“Render to Data Structures”

- DX11 PixelShader 5
  - Atomics to global memory
  - Gather/scatter to memory ("unordered access views")

- Order independent transparency
  - Capture all rendered fragments
    - Render directly to grid-of-lists data structure instead of framebuffer
    - No framebuffer bound while rendering 😊
    - Increment global counter to get unique address for fragment
    - Scatter {depth, color, prevFrag} to UAV shared by all pixels
    - Result is grid of linked lists
  - Sort and blend lists for final image (pixel shader or compute shader)
Order Independent Transparency
There is no single graphics pipeline

- There is no single workload to optimize

- Moving forward, interactive rendering is an inseparable mix of
  - Task- and data-parallel algorithms
  - Standard, extended and custom graphics pipelines
But Some Food For Thought…
Gradually the processor became more complex. Finally the display processor came to resemble a full-fledged computer with some special graphics features. And then a strange thing happened. We felt compelled to add to the processor a second, subsidiary processor, which, itself, began to grow in complexity. It was then that we discovered a disturbing truth. Designing a display processor can become a never-ending cyclical process. In fact, we found the process so frustrating that we have come to call it the "wheel of reincarnation."

Will There Be Another Turn of The Wheel of Reincarnation?

• Is “the rise of SW graphics” a temporary (5-10) year window as we go around the wheel of reincarnation or has the wheel stopped turning?

• If it has stopped turning, why?

• If it hasn’t stopped turning, what will be the next fixed-function hardware?
Conclusions

• Software + hardware graphics is here today (beginning “for real” in DX11)
  – Graphics programming is no longer simply a single pre-defined pipeline
  – Research is ablaze with software rendering research on GPUs and CPUs

• Future real-time rendering programming will consist of
  – A pre-defined (Direct3D/OpenGL) rendering pipeline
  – User-defined software pipelines
  – User-defined data- and task-parallel code tightly coupled to graphics pipelines

• Is the wheel of reincarnation still turning?
Beyond Programmable Shading I

9:00–9:20 - Mike Houston, AMD
Looking Back, Looking Forward, Why and How is Interactive Rendering Changing

9:20–9:45 - Johan Andersson, DICE
Five Major Challenges in Interactive Rendering

9:45–10:15 - Kayvon Fatahalian, Stanford
Running Code at a Teraflop: How a GPU Shader Core Works

10:15–10:45 - Aaron Lefohn, Intel
Parallel Programming for Real-Time Graphics

10:45–11:15 - Chas Boyd, Microsoft
DirectCompute Use in Real-Time Rendering Products

11:15–11:45 - David Luebke, NVIDIA Research
Surveying Real-Time Beyond Programmable Shading Rendering Algorithms

11:45–12:15 - Johan Andersson, DICE
Bending the Graphics Pipeline
2:00–2:05 - Aaron Lefohn, Intel
Welcome and Re-Introduction

2:05–2:35 - Jonathan Ragan-Kelley, MIT
Keeping Many Cores Busy: Scheduling the Graphics Pipeline

2:35–3:10 - Kayvon Fatahalian, Stanford
Evolving the Direct3D Pipeline for Real-Time Micropolygon Rendering

3:10–3:30 - Jonathan Ragan-Kelley, MIT
Decoupled Sampling for Real-Time Graphics Pipelines

3:30–3:50 - Andrew Lauritzen, Intel
Deferred Rendering for Current and Future Rendering Pipelines

3:50–4:15 - Luca Fascione, WETA
Jacopo Pantaleoni, NVIDIA Research
PantaRay: A Case Study in GPU Ray-Tracing for Movies

4:15–4:30 - Mike Houston, AMD
Wrapup: What’s Next for Interactive Rendering Research?
Panel: What Role Will Fixed-Function Hardware Play in Future Graphics Architectures?

Moderator: Kurt Akeley
Microsoft Research

Panelists:
Steve Molnar, NVIDIA
David Blythe, Intel
Mike Houston, AMD
Johan Andersson, DICE
Kayvon Fatahalian, Stanford
Course webpage and slides:
http://bps10.idav.ucdavis.edu