Fragments Are Expensive

- Deferred shading: reduces expensive overdraw

(image by Sascha Willems)
Fragments Are Expensive

- **Many** adaptive sampling algorithms in raytracing
  - With us since the beginning!

"I had written a draft of a SIGGRAPH conference submission and was rendering illustrations to be included in the paper. The submission deadline was near, but with 16x super-sampling, the estimated rendering times extended beyond the submission deadline. The spontaneous idea of **adaptively super-sampling** was a life saver because it only added additional samples where needed. It was implemented within a couple of hours and the paper was edited to include this new idea while the illustrations were being rendered."

-J. Turner Whitted

*Figure 6*
(image by J. Turner Whitted)
Fragments Are Expensive

- Checkerboard rendering: reduce shades to < 1 / pixel. Adaptive sampling doesn't map well to GPU!

(Average 2 neighbors) + (Average across diagonals) =

\[
\begin{array}{ccc}
1/16 & 1/8 & 1/16 \\
1/8 & 1/4 & 1/8 \\
1/16 & 1/8 & 1/16
\end{array}
\]

= 

Optimized Bilinear Samples

Weights near to far: 0.375, 0.375, 0.125, 0.125

Weights near to far: 0.5, 0.28125, 0.09375, 0.09375, 0.03125

(slide by Alex Vlachos)
Deferred Adaptive Compute Shading

- Replacement for checkerboard rendering
  - (one-fewer pass, simple, provided code)

- Reduces shading adaptively

- Still GPU-friendly:
  - Typical results: 2–4× better quality/perf
Adaptive Subdivision

• Simple but proven adaptive subdivision scheme (inspired by V-Ray)
  - (Tried some others, but this one works best)

• Elegant rotational pattern
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision

Neighbors Dissimilar?

Shade!
Adaptive Subdivision

• “Similarity” given by user-defined metric

• We suggest:
  – “Dissimilar” if material IDs different
  – “Dissimilar” if final colors differ by threshold or more
  – Look at other G-buffer features?
Adaptive Subdivision

Neighbors Dissimilar?

Shade!
Adaptive Subdivision

Neighbors Dissimilar?
Shade!

Neighbors Similar?
Interpolate!
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision
Adaptive Subdivision

Ground Truth
Adaptive Subdivision

Ground Truth

Shading Rate
(Heuristic: colors < variance threshold)
Adaptive Subdivision

Ground Truth

Shading Rate
(Heuristic: colors $<$ variance threshold)

DACS
Adaptive Subdivision

Ground Truth

Shading Rate
(Heuristic: colors < variance threshold)

DACS

Ground Truth

DACS
Deferred Adaptive Compute Shading
Implementation on Current GPUs
Warp Divergence

- Cannot skip pixels like this!
  - GPU still does the work (it’s just wasted)!

- Solution: warps switch between “search/interpolate” mode and “shade” mode
Mode Switching

Warp

Mode = SEARCH

Shade buffer

Incoming pixels

Written pixels

Thread 0  Thread 1  Thread 2  Thread 3
Mode Switching

Warp

Mode = SEARCH

Shade buffer

Incoming pixels

Written pixels

Thread 0  Thread 1  Thread 2  Thread 3
Mode Switching

Warp

Mode = SEARCH

Shade buffer

Incoming pixels

Written pixels
Mode Switching

Mode = SEARCH

Shade buffer

Incoming pixels

Written pixels
Mode Switching

Warp

Mode = SEARCH

Shade buffer

Thread 0
Thread 1
Thread 2
Thread 3

Incoming pixels

Written pixels
Mode Switching

Warp

Mode = SEARCH

Shade buffer

Thread 0 | Thread 1 | Thread 2 | Thread 3

Incoming pixels

Written pixels
Mode Switching

Warp

Mode = SEARCH

Shade buffer

Incoming pixels

Written pixels
Mode Switching

Warp

Mode = SEARCH

Shade buffer

Incoming pixels

Written pixels
Mode Switching

Warp

Mode = SEARCH

Shade buffer

Incoming pixels

Written pixels
Mode Switching

Mode = SEARCH

Shade buffer

Thread 0
Thread 1
Thread 2
Thread 3

Incoming pixels

Written pixels
Mode Switching

Warp

Mode = SEARCH

Shade buffer

Incoming pixels

Thread 0

Thread 1

Thread 2

Thread 3

Written pixels
Mode Switching

Mode = SHADE

Incoming pixels

Written pixels
Mode Switching

Warp

Mode = SHADE

Shade buffer

Incoming pixels

Written pixels

Thread 0
Thread 1
Thread 2
Thread 3
Mode Switching

Warp

Mode = SHADE

Incoming pixels

Written pixels
Mode Switching

Mode = SHADE

Incoming pixels

Warp

Thread 0
Thread 1
Thread 2
Thread 3

Written pixels
Mode Switching

Warp

Mode = SEARCH

Shade buffer

Incoming pixels

Thread 0

Thread 1

Thread 2

Thread 3

Written pixels
The Code

Download plaintext from https://geometrian.com/research/
Results

- Ran color-only simulated results on synthetic images

- Implemented in my deferred renderer
  - Comparison to simple checkerboard implementation
  - Note: no temporal filtering in any algorithm!

- Ran color-only simulated results on Unreal Engine frames (see video).
  - Timing not meaningful
Results: Synthetic Images

- Perfectly reconstructs gradient and step functions

- Gradient: characteristic of soft shadows, shaded regions

- Step: texture features, depth discontinuities
Results: Thin Features

Considering material ID in user criterion prevents undersampling geometry.
Results: Deferred Renderer

Ground truth
Results: Deferred Renderer

Ground truth
Results: Deferred Renderer

Ground Truth
Results: Deferred Renderer

Checkerboard
- Loss of detail in a single frame
- 1.89× speedup

RMSE: 0.04218
PSNR: 27.53
MSSIM: 0.8954

Speedup: 1.89× to GT
Results: Deferred Renderer

DACS

- "Equal" quality to checkerboard (same MSSIM)
- Better edge resolution
- $4.22 \times$ speedup!

RMSE: 0.02647
PSNR: 31.57
MSSIM: 0.8881

Speedup: $4.22 \times$ to GT
Speedup: $2.24 \times$ to checkerboard
Results: Deferred Renderer

DACS

- Equal time to checkerboard
- Far better quality

RMSE: 0.009076
PSNR: 40.85
MSSIM: 0.9620

Speedup: 1.89× to GT
Speedup: 1.00× to checkerboard
Results: Deferred Renderer

Ground Truth
Extensions and Applications

- Adaptive Supersampling
- Foveated Rendering

- Temporal Filtering
- Framerate stabilization
- More G-Buffer Features
- Perceptual Heuristics
- Energy Tradeoff for Mobile
Conclusion

• Significantly reduces shading complexity
• Adaptive, yet runs efficiently on GPUs
• Simple implementation
Questions
(and video)