Decoupled Deferred Shading for Hardware Rasterization

Miklas & Yassen
Content

- Introduction
- The Compact Geometry Buffer
- The Algorithm
- Evaluation
- Conclusion
Introduction

• High Quality Cinematic Effects
  o Distributed Ray tracing
  o Stochastic Rasterization

• Antialiasing
  o MSAA
Introduction

• Deferred Shading
• The problem with multisampling
• Decoupling
  o Visibility Samples
  o Shading Samples
Geometry Buffer

- Naively couples visibility and shading

- Memory consumption grows exponentially
Compact Geometry Buffer

- The solution: Separate shading and visibility
The CG Buffer

- Compact Geometry Buffer
  - Avoids storing redundant shading data
  - Less dependent on sampling density
CG Buffer

• What the CG Buffer Does:
The CG Buffer

- Decoupled Shading
  - Map Visibility samples to a “Shading Domain”
  - Many-to-one mapping
How does the CG buffer work?
How does the CG Buffer work, cont.?

• Assign Unique Id’s to Shading Samples (ssIDs)
  o ssID is unique across the image
  o Map ssIDs to each fragment
  o Cache shading data
Algorithm

- Global cache generation
  - Sampling and shading done in one pass
  - Uses standart fragment shaders

- Local cache
  - Multi Pass Approach
  - Uses GPU Compute Kernels
Global Cache Method

• Replaces G-Buffer generation

1. Determine ssID range for each primitive
   o In Geometry Shader

2. Map each fragment to an ssID
   o Depending on the domain
Global Cache Method cont.

• Check for the address in the cache
  o If it’s there return it in the visibility reference

• What happens when ssID is not in cache?
  o Acquire lock for the ssID - Address pair
  o Generate buffer data in render targets
  o Save address in the cache and return it
Global Cache

• Use cache to store recent ssIDs
  o Hit rate vs. Overhead

• Cache is divided into buckets
  o Buckets are mapped to ssIDs through a hash function

• Each bucket has an array of pairs
  o Each pair is ssID-Address
Global Cache - Problems

• Race conditions

• Compaction
  o Overdraw generates unreferenced data in the CG Buffer
  o Fill z-buffer in a pre pass, or
  o Clean up buffers after sampling
Per-Tile Shading Cache

- Global shading cache
  - Becomes major bandwidth bottleneck
  - Most shading samples are visible in small areas, not globally.
1. z-fill and ssID mapping
2. per-tile caching
3. only visible fragments: interpolate and store
Results

+ Real time
  + Adaptive shading rate

+ Cheap
  + Motion blur
  + FOV
  + AA ($\geq 4x$)

+ Low bandwidth (AA ($\geq 4x$))
Memory Consumption

Storage (MB)

Visibility Supersampling

- Supersampled
- Decoupled
- Decoupled Adaptive
- Vis. References
Results

- Optimization (local cache) is slower
- If no AA and shading rate = 1, double memory usage (than deferred shading)
Results

- Not physically correct
- Possible artifacts on fast-moving objects
Decoupled Deferred Shading for Hardware Rasterization

Gabor Liktor and Carsten Dachsbacher

Computer Graphics Group
Karlsruhe Institute of Technology
Questions?