MSAA-Based Coarse Shading
for Power-Efficient Rendering on High Pixel-Density Displays

Pavlos Mavridis Georgios Papaioannou

Department of Informatics, Athens University of Economics & Business

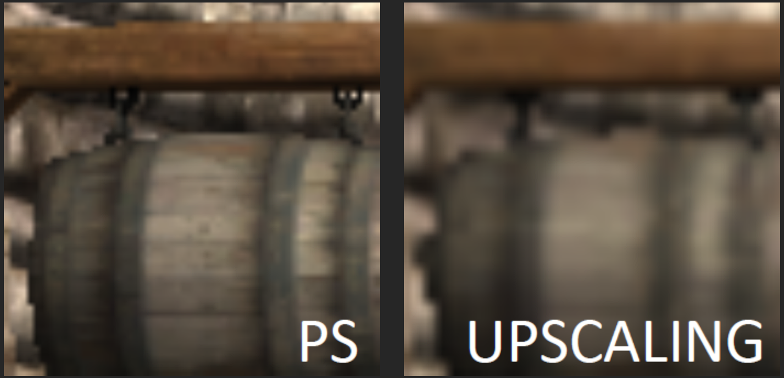
Motivation

- High-pixel-density displays are widely used on many devices
  - Mobile phones and tablets
  - Laptops
  - 4K monitors/TVs are now mainstream
  - 5K displays are introduced in the high-end
- High-pixel-density -> (very) high resolution
- Real-time rendering on such resolutions is challenging
  - The cost of shading all these pixels can be prohibitive, especially on **power-constrained mobile devices**

Typical Approaches

Render at a lower resolution and upscale

- Introduces blurriness
- Thin geometric features are under-sampled



Edge-preserving filters (bilateral) can be used for better results, but under-sampling is still an issue.

Images from [Vaidyanathan et al. 2014]

A Better Alternative

- Use a ***decoupled sampling*** approach, where visibility is sampled at a higher rate than shading.
- In particular, the desired approach will:
  - Sample visibility ***at least*** once per pixel, in order to preserve the clarity of geometric edges
  - Perform shading at a lower (more *coarse*) rate.

Decoupled Sampling: Previous Work

- Parametric-space shading
  - Reyes [Cook et al. 1987], [Andersson et al. 2014], …
  - Caveats: high overhead when implemented on existing GPUs, overshading.
- Deferred shading
  - Inherently decouples visibility from shading
  - Many decoupling opportunities [Lauritzen 2010; Kerzner and Salvi 2014; …]
  - Caveats: memory bandwidth (or tiling overhead), transparency.
- Multi Sample Anti-Aliasing (MSAA) [Akeley 1993]
  - Supported virtually on all shipping GPUs.
  - Caveat: Each covered primitive is shaded **at least once per pixel**
    -> Not directly applicable for coarse shading.

Decoupled Sampling: Previous Work

- Recent extensions of MSAA allow coarse and multi-rate shading [Vaidyanathan et al. 2014; He at al. 2014; Clarberg et al. 2014]
  - Hardware modifications are required.

- Can we perform **coarse shading** efficiently on **existing GPUs** in the context of **forward rendering**?

MSAA-based coarse shading

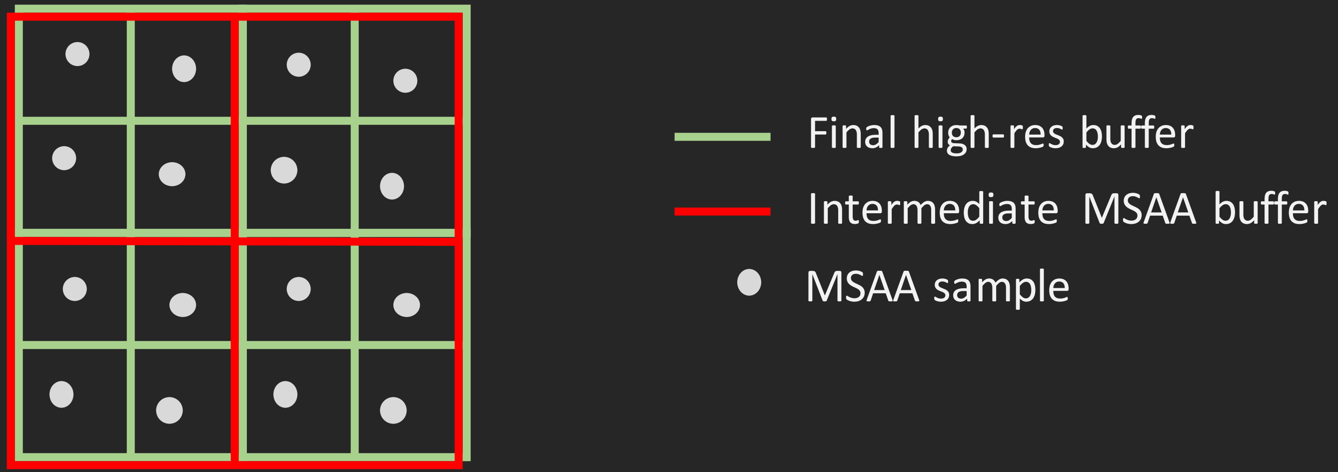
- **Main Idea:**
  - Starting from the render buffer with the final desired resolution.

MSAA-based coarse shading

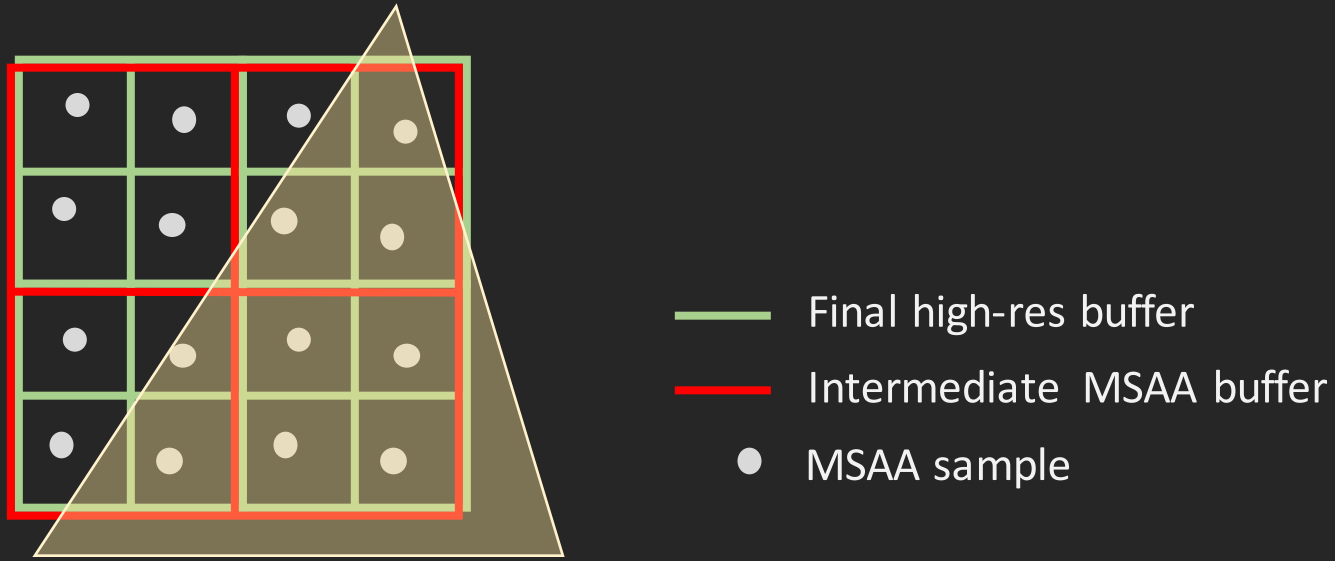
- **Main Idea:**
  - Starting from the render buffer with the final desired resolution.
  - Every NxN block of pixels is replaced with one pixel with **at least** NxN MSAA samples.

MSAA-based coarse shading

- **Main Idea:**
  - Starting from the render buffer with the final desired resolution.
  - Every NxN block of pixels is replaced with one pixel with **at least** NxN MSAA samples.
  - A custom resolve shader maps individual MSAA samples to screen pixels

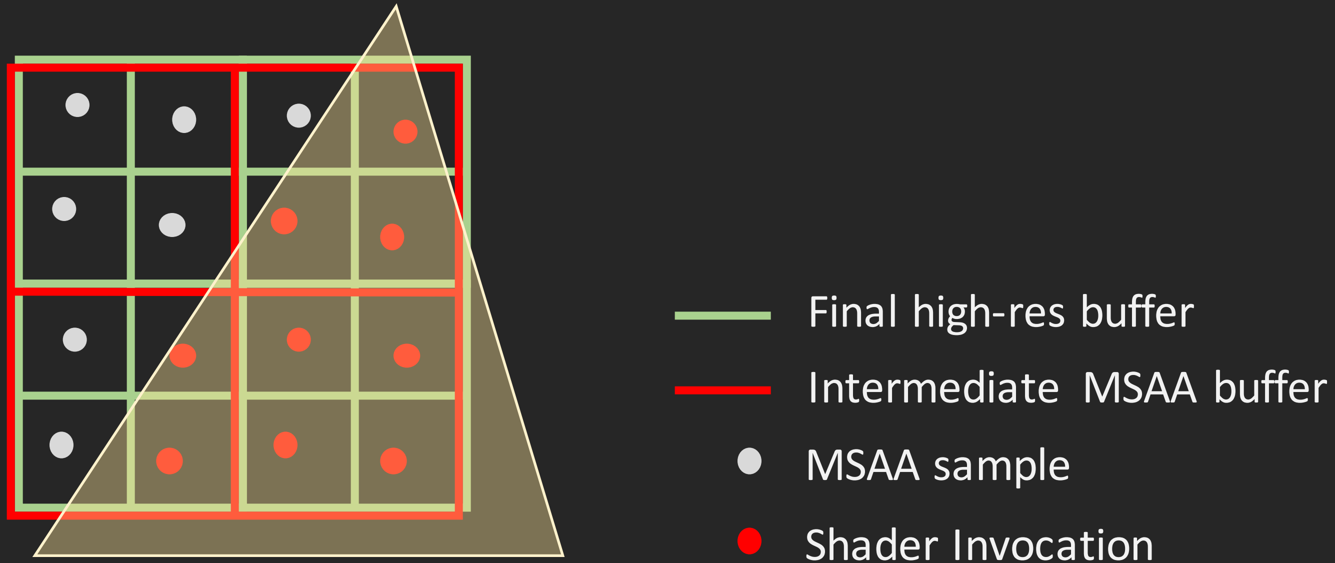


MSAA-based coarse shading



MSAA-based coarse shading

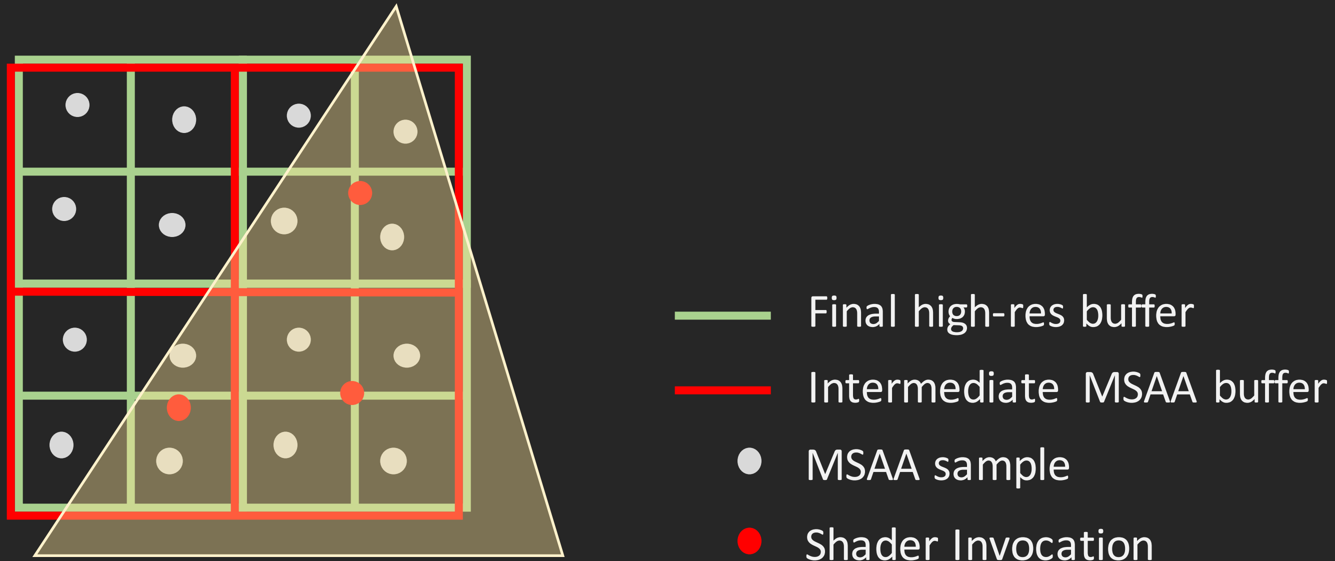
- **Per pixel shading:** Nine fragment shader invocations



Per-pixel Shading

MSAA-based coarse shading

- **Per pixel shading:** Nine fragment shader invocations
- **Coarse shading:** Three fragment shader invocations **(3x reduction)**

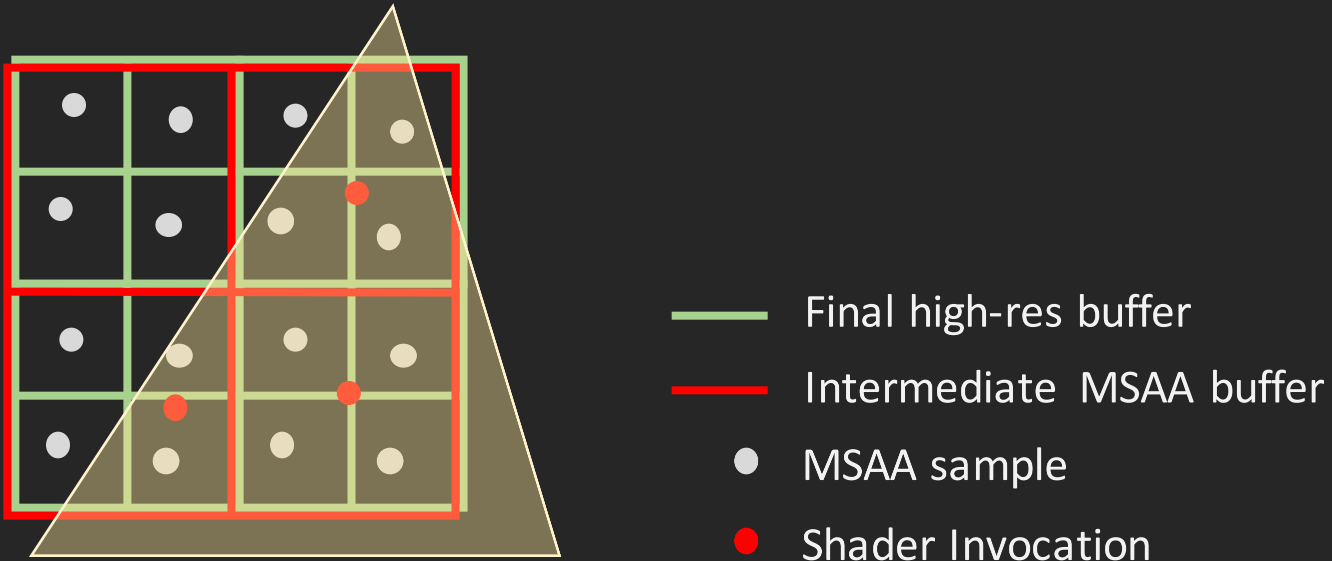


Coarse Shading

MSAA-based coarse shading

- **Per pixel shading:** Nine fragment shader invocations
- **Coarse shading:** Three fragment shader invocations **(3x reduction)**

Centroid sampling should be used in order to evaluate shaders near the covered sample positions.



Coarse Shading

Practical Configurations

Configuration #1

¼ resolution + **4x**MSAA

Visibility: 1 sample/pixel

Shading: *at least* 1 sample per 2x2 block

Resolve: **1:1 mapping** (nearest filtering)

Practical Configurations

Configuration #1

¼ resolution + **4x**MSAA

Visibility: 1 sample/pixel

Shading: *at least* 1 sample per 2x2 block

Resolve: **1:1 mapping** (nearest filtering)

Configuration #2

¼ resolution + **8x**MSAA

Visibility: 2 samples/pixel

Shading: *at least* 1 sample per 2x2 block

Resolve: **2:1 mapping** (weighted average of 2 values)

Practical Configurations

Configuration #1 (alternative)

1. Bind a normal high-res buffer
2. “Pretend” it is a low-res MSAA buffer during rendering

(it could be done on some consoles exploiting non-standard APIs and H/W behavior)

Configuration #2

¼ resolution + **8x**MSAA

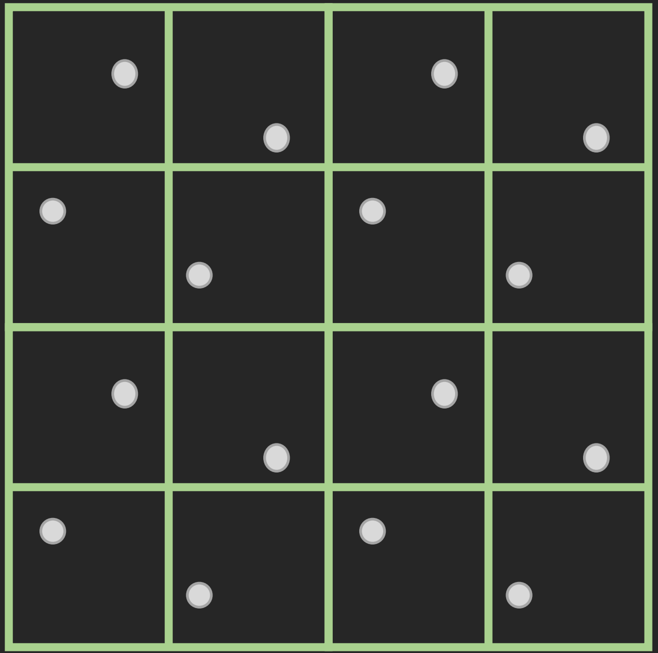
Visibility: **2** samples/pixel

Shading: *at least* 1 sample per 2x2 block

Resolve: **2:1 mapping**
(weighted average of 2 values)

Spatial Interleaving

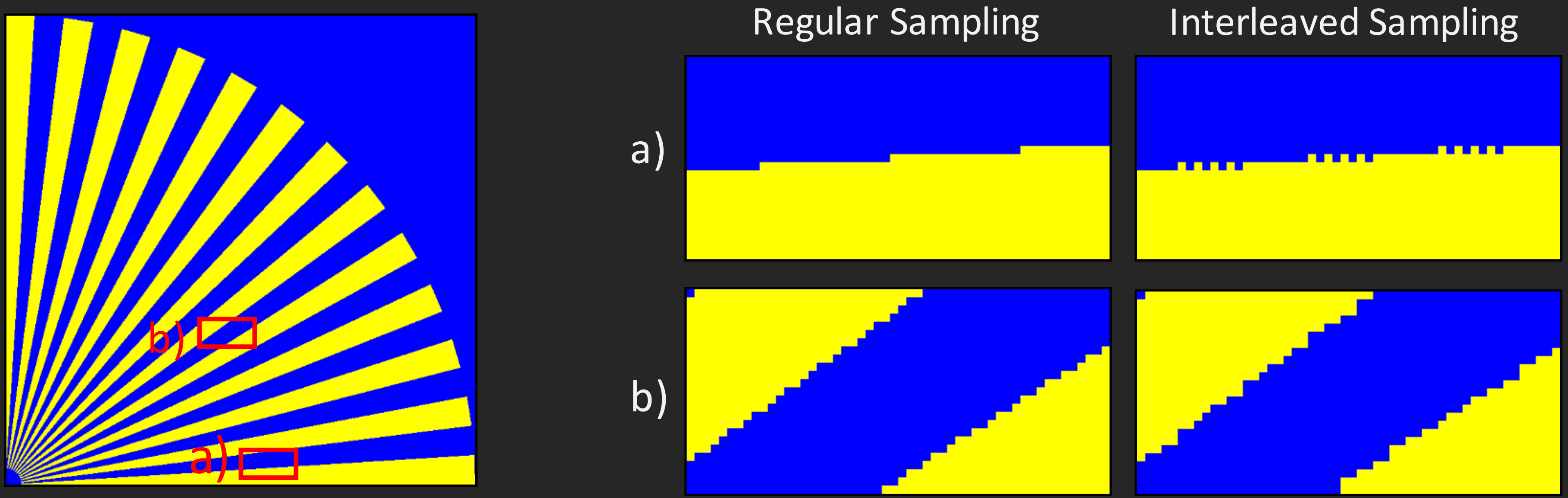
- The method Inherently creates **interleaved sampling** patterns [Keller et al. 2001]



Neighboring pixels are always sampled at different positions.

Spatial Interleaving

- Masks *inter-pixel* aliasing with noise
  (well-known concept from stochastic sampling)



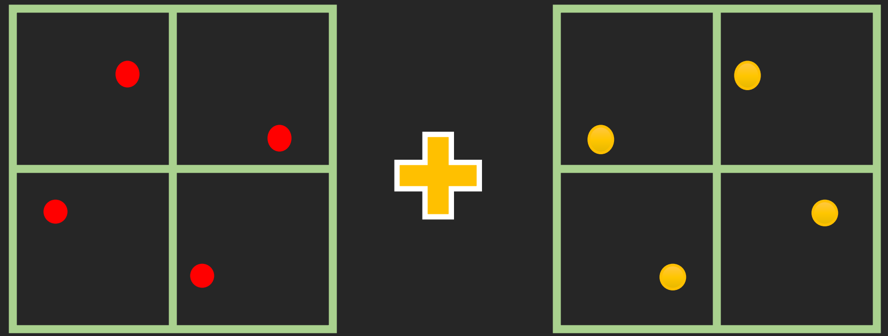
Spatial Interleaving

- Beneficial even when one visibility sample per pixel is used on high-pixel-density displays.
  - But the more samples the better.
  - **Limitation:** Up to two visibility samples per pixel on 8xMSAA hardware (when 2x2 pixel blocks are used for coarse shading)
- On low-pixel-density displays regular sampling + MLAA/FXAA/etc… might be preferable (highly subjective).
  - Revert to regular sampling using **programmable sample positions**.

Note: NVIDIA's Maxwell architecture supports interleaved sampling with MSAA (>1 sample/pixel).

Temporal Interleaving

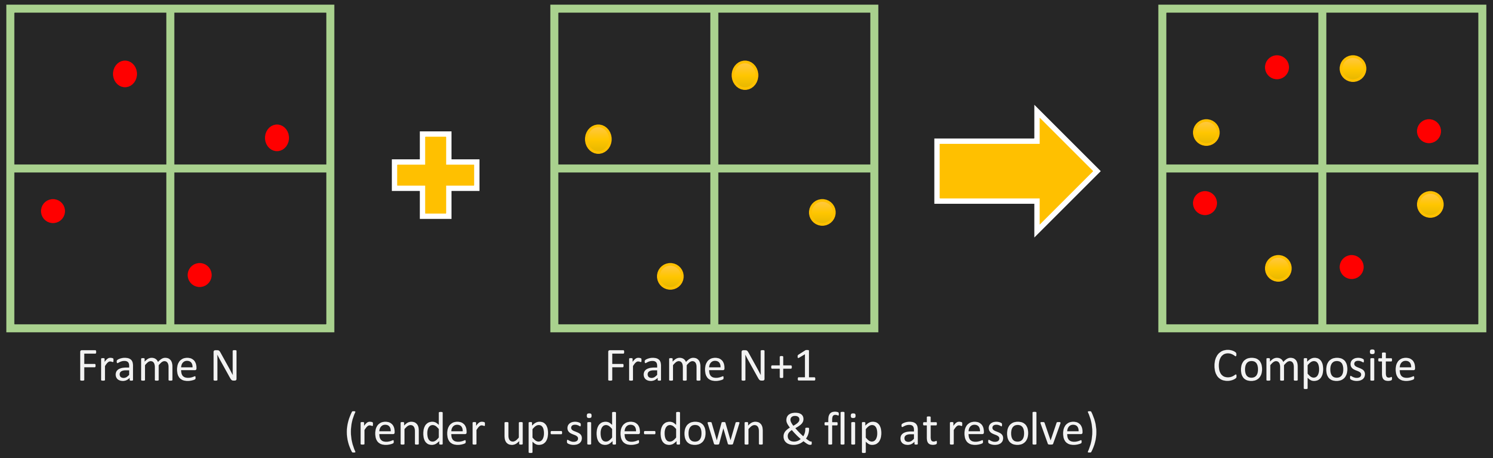
- Alternate between two sampling patterns at successive frames



(render up-side-down & flip at resolve)

Temporal Interleaving

- Alternate between two sampling patterns at successive frames
- **Static content:** Render at a high frame rate (>60Hz **v-sync locked**) and let the human visual system perform the required averaging.
- **Dynamic content:** Temporal sample re-projection…



“Side Effects”

“Side effect”: triangle size

- The pixel footprint gets bigger-> relative triangle size is decreased.
- Can negatively affect the efficiency of rasterization
  (due to increased *“**quad over-shading**”)*
- Techniques like ***Quad-Fragment Merging*** [Fatahalian et al. 2010], ***Pixel Merge Unit*** [Sathe et al. 2015] or similar can be highly beneficial.

“Side effect”: frame buffer compression

- Neighboring pixels share the same color
  - -> good opportunity for better compression.
- But, our implementation binds an MSAA render target for reading
  - this disables compression or triggers a decompression operation on some architectures -> *high overhead.*
- An implementation that avoids this issue *might* be possible
  - Mobile architectures: read directly from the on-chip *tile local storage*.

Expect different behavior on different GPUs
(tests show more gains on Intel GPUS than on NVIDIA or AMD)

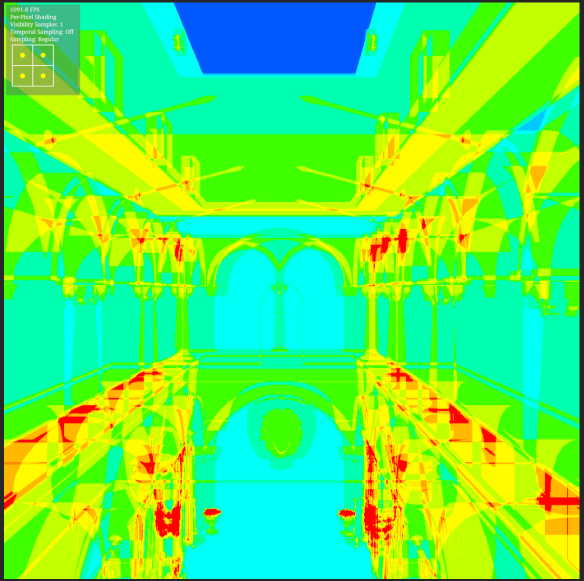
Results

Methodology

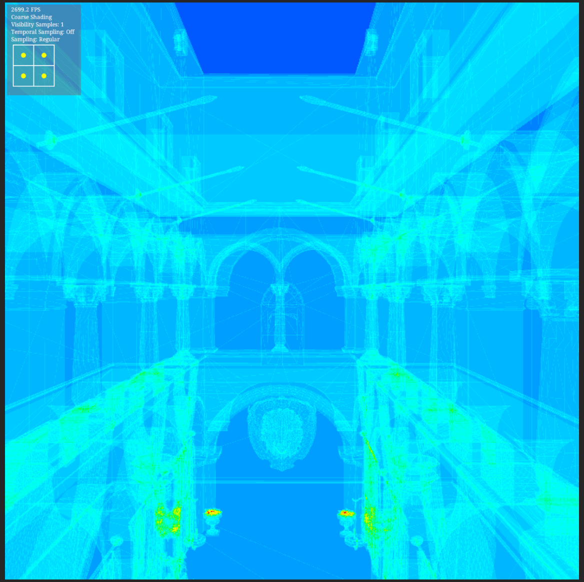
- Test Hardware: Apple MacBook Pro 13’’ / Intel Iris 5100 GPU
- *Intel Power Gadget* tool for energy measurements
  - Measurements include both CPU and GPU consumption.
  - **V-Sync** ensures that all methods perform the same amount of work.
- ARB_pipeline_statistics_query for shader invocation measurements
  - Includes “helper” invocations for derivative calculations

Shader Invocations (per 2x2 pixel block)

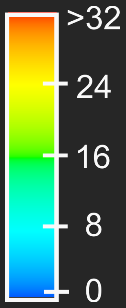
Sponza Atrium
262k Triangles



Per Pixel Shading



Coarse Shading



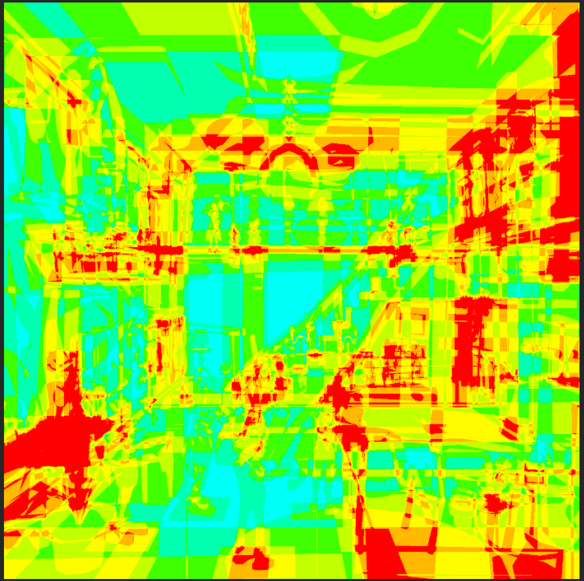
Invocations: 69.3% reduction (3.5M vs 1.1M)

Power usage: 45.7% reduction (32 vs 17 Watts)

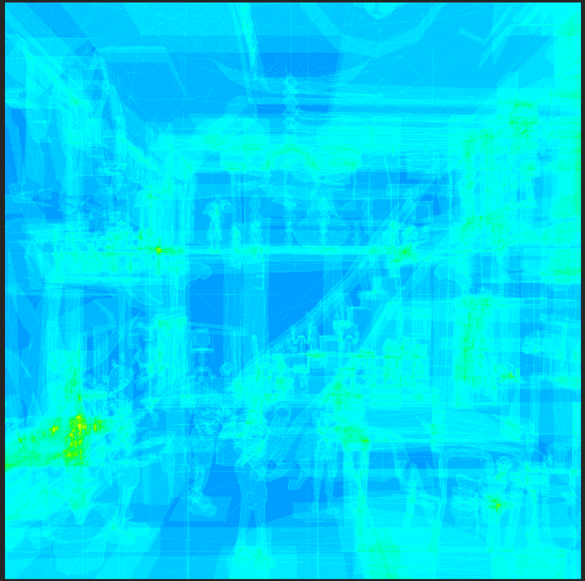
OR 1.73x speedup in rendering

Shader Invocations (per 2x2 pixel block)

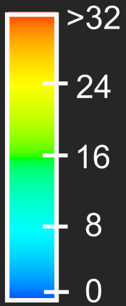
Mansion Scene
53k Triangles



Per Pixel Shading



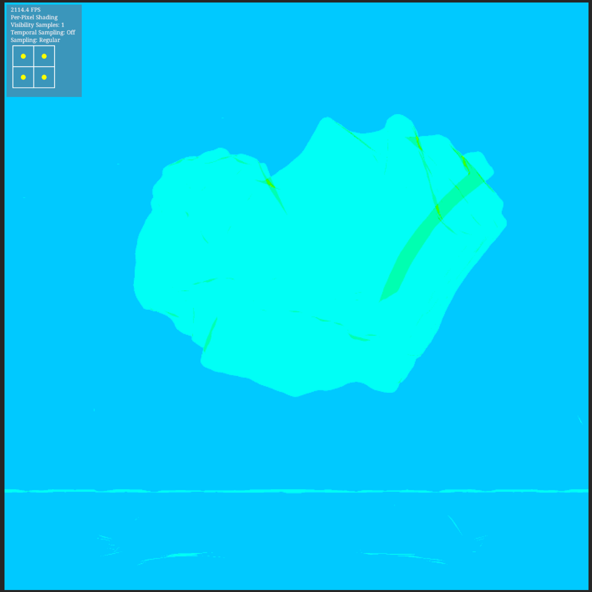
Coarse Shading



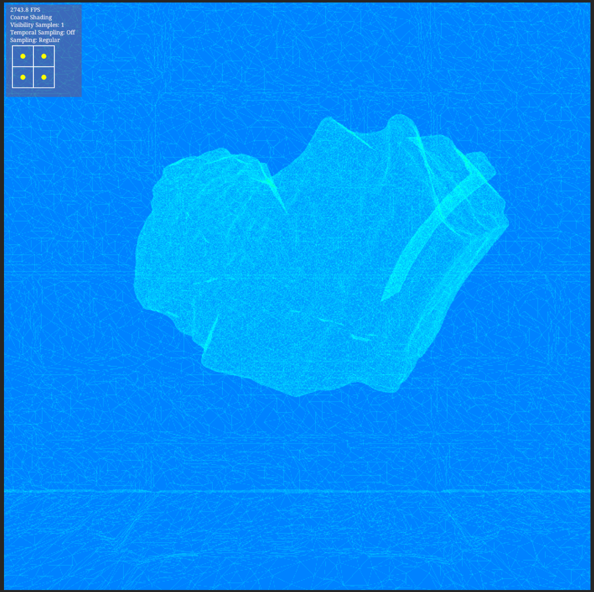
Invocations: 72.5% reduction (10M vs 2.7M)
Power usage: 12.6% reduction (8.7 vs 7.6 Watts)
OR 1.23x speedup in rendering

Shader Invocations (per 2x2 pixel block)

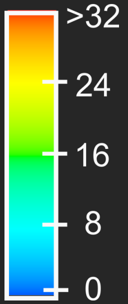
Volumetric Shadows
215k Triangles



Per Pixel Shading



Coarse Shading

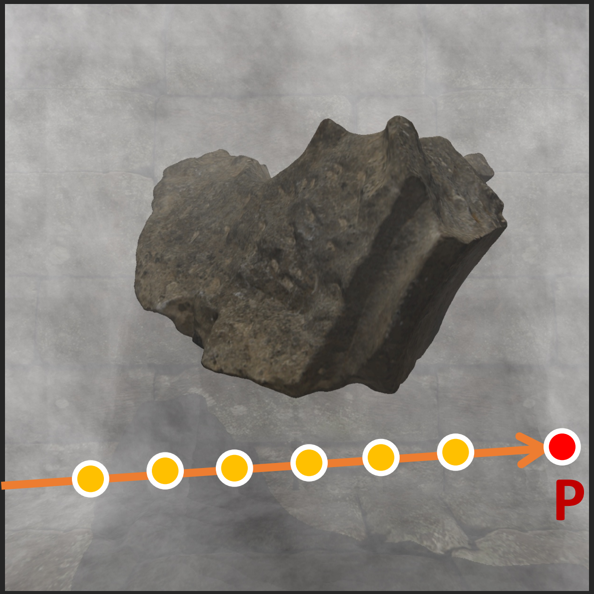


Invocations: **63.2%** reduction (1.9M vs 0.7M)

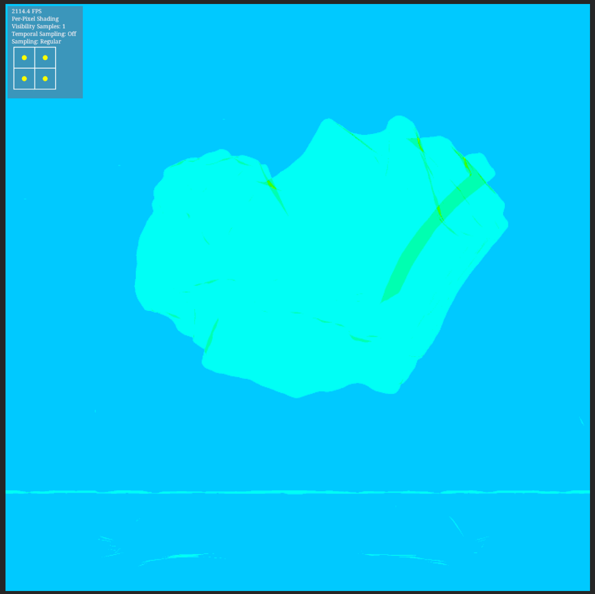
Power usage: **45.8%** reduction (30 vs 16 Watts)

OR **2.14x** speedup in rendering

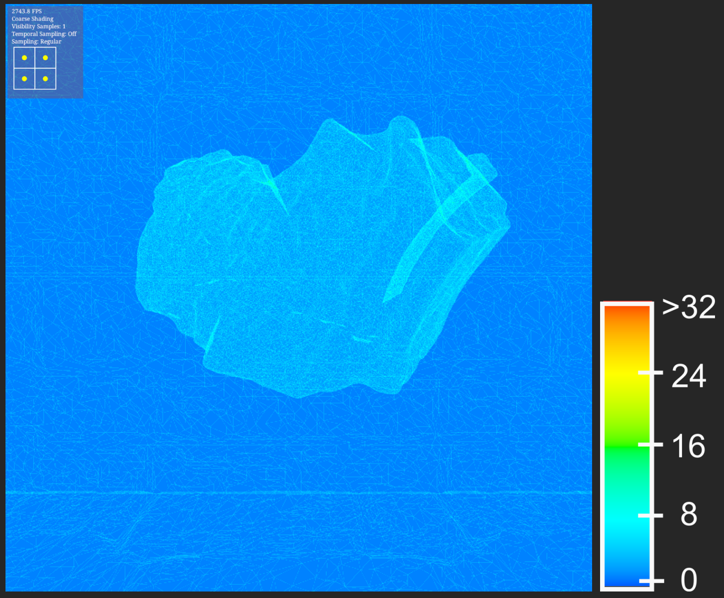
Shader Invocations (per 2x2 pixel block)



Volumetric Shadows
215k Triangles



Per Pixel Shading



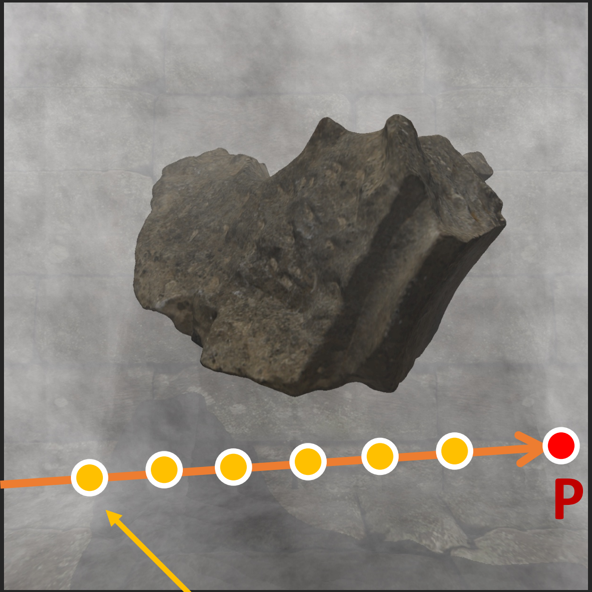
Coarse Shading

Invocations: **63.2%** reduction (1.9M vs 0.7M)

Power usage: **45.8%** reduction (30 vs 16 Watts)

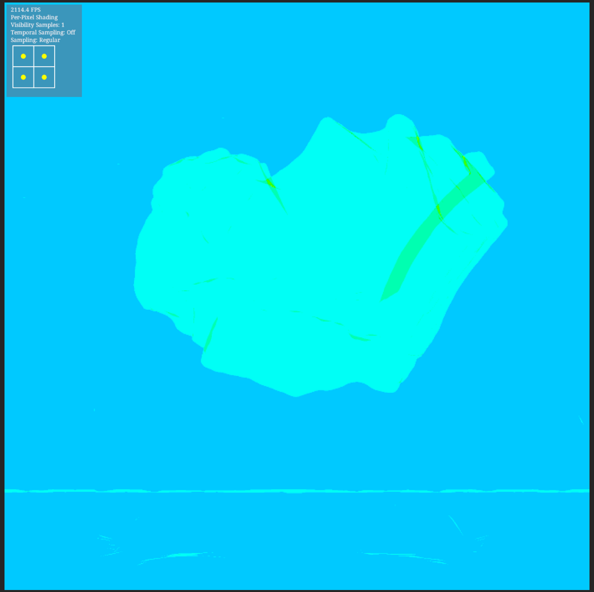
OR **2.14x** speedup in rendering

Shader Invocations (per 2x2 pixel block)

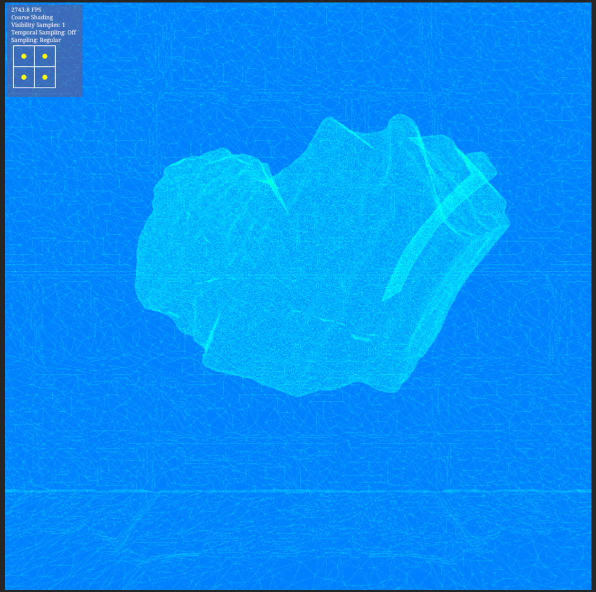


Volumetric Shadows
215k Triangles

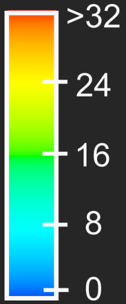
Shadow-map lookup +
4D Perlin Noise + math



Per Pixel Shading



Coarse Shading

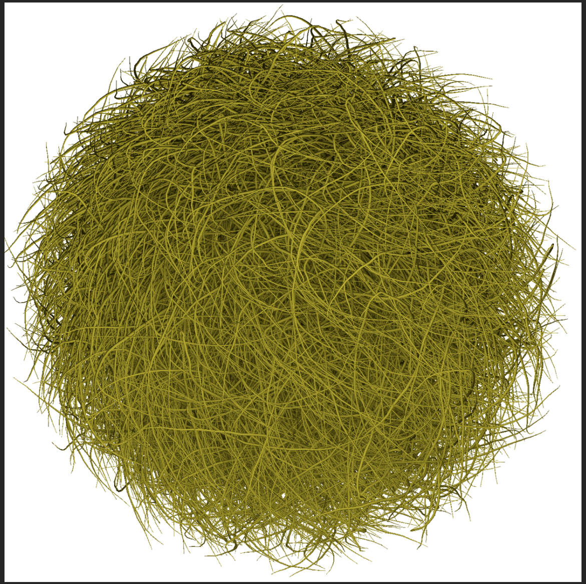


Invocations: **63.2%** reduction (1.9M vs 0.7M)

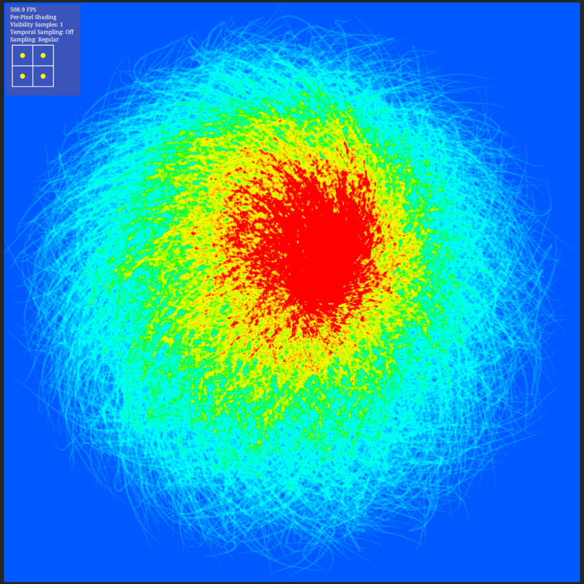
Power usage: **45.8%** reduction (30 vs 16 Watts)

OR **2.14x** speedup in rendering

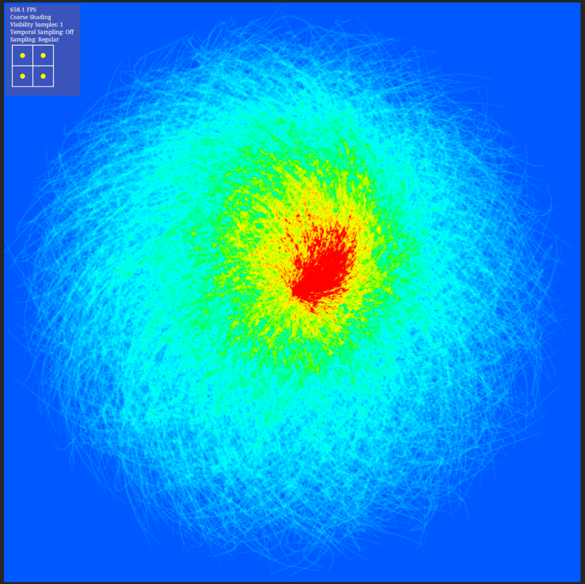
Shader Invocations (per 2x2 pixel block)



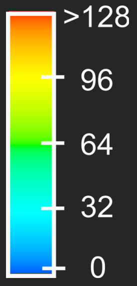
Hairball
2850k Triangles



Per Pixel Shading

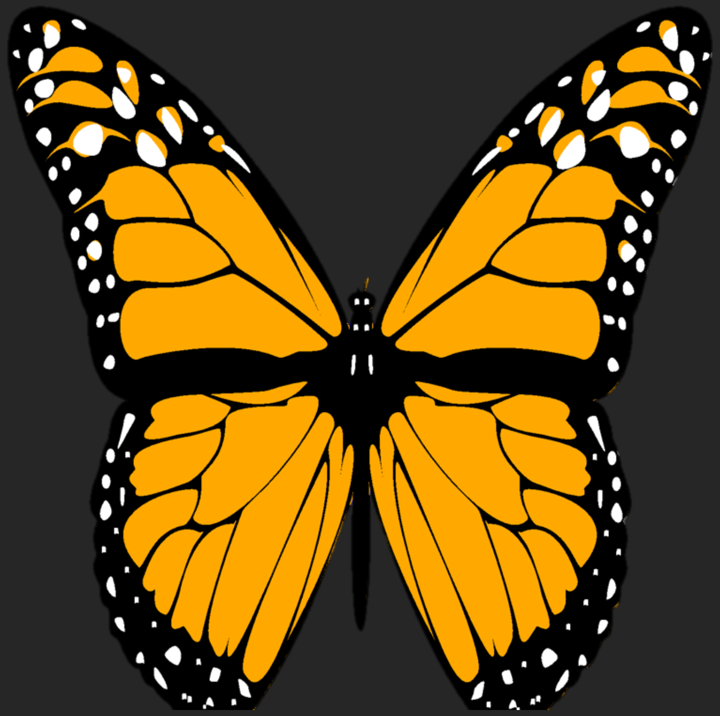


Coarse Shading



Invocations: **34.9%** reduction (3.2M vs 2.0M)
Power usage: **7.4% increase** (29 vs 31 Watts)
OR **0.82x** speedup in rendering

Vector Drawing



Butterfly Scene (SVG)

- Tested using the NanoVG library (vector drawing with OpenGL)
- Important for UI / Maps / 2D games
- Flat or smooth shaded regions exhibit very small loss of image quality.

Invocations: **74%** reduction (439K vs 114K)
Power usage: **14.3%** reduction (7 Watts vs 6 Watts)
OR **1.27x** speedup in rendering

Image Quality

- **Subjective evaluation:** we can switch between per-pixel and coarse shading without noticing any quality degradation on Hi-PPI displays **(see the demo!)**
- **Objective evaluation:**

Scene Name	SSIM (%)
Butterfly (2D SVG)	99.9
Mansion	95.6
Volumetric Shadows	96.5
Sponza	81.2
Hairball	97.6

Coarse shading compared to per-pixel shading

Extension: Selective Coarse Shading

- Apply coarse shading only on specific parts of the scene
  - Out of focus regions (defocus blur)
  - Fast moving objects (motion blur)
  - Distant objects (often covered by thin haze)
  - Peripheral objects in VR applications (foveated rendering)
- On existing hardware we can switch between coarse and fine shading between draw-calls
  - Not as fine-grained control as the proposed hardware implementations

Conclusions

- We have presented a method to perform coarse shading efficiently on commodity graphics hardware
  - Important for practitioners & software developers.
  - Important for IHVs: before designing new hardware extensions it is important to understand the limits of existing architectures.
- Performance and energy-consumption analysis on various scenes.
  - Up to **45% reduction in power consumption**.
- *Interleaved sampling* for improved antialiasing.

Thank you for your attention!

Acknowledgement
This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement No 600533