SIGGRAPH2005

GPU-Accelerated Production Rendering

Larry Gritz
Digital Film Group
NVIDIA

Acknowledgements

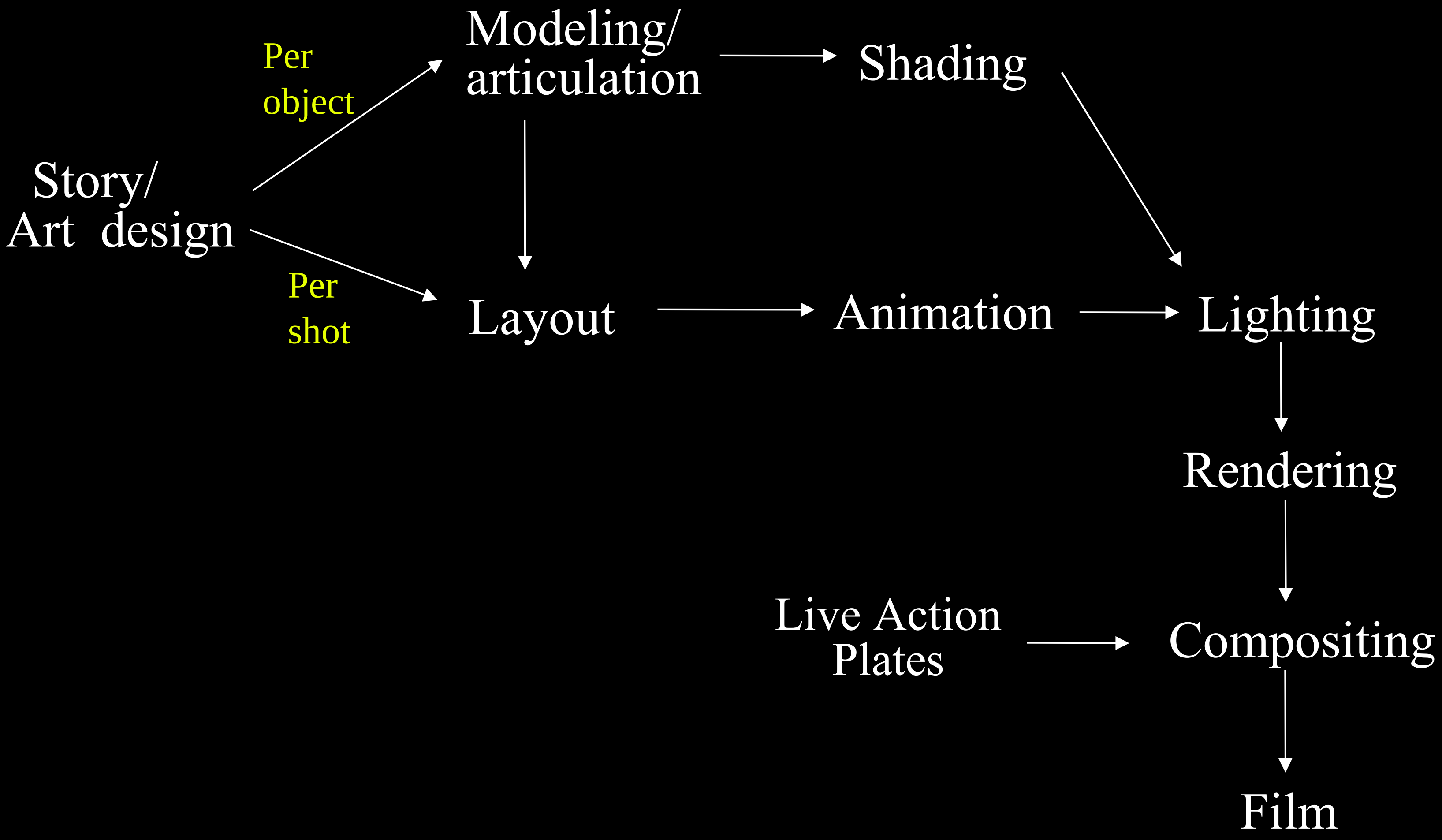
- Gelato development:

  Dan Wexler, Eric Enderton, Philip Nemec, Radomir Mech, John Schlag, Jonathan Rice, Sharif Elcott

- NVIDIA Digital Film Group:

  Beth Loughney, Laura Dohrmann, Cynthia Dueltgen, Dave Wilton, Matt Jefferson

Production pipeline



Story/
Art design

Per
object

Per
shot

Modeling/
articulation

Shading

Layout

Animation

Lighting

Rendering

Live Action
Plates

Compositing

Film

Film != Games

- Games
  - Fixed frame rate – quality negotiable
  - Render 10^5 frames x many games x 10^6 users
  - -> optimize for rendering
  - -> pre-computation at dev time inconsequential
- Film
  - Fixed quality – frame rate unimportant (mostly)
  - Render once, deliver film
  - Humans are bottleneck – maximum flexibility more important than speed -> optimize for development
  - Artist in the loop for every frame

Why Film Rendering Will Never be Realtime

- Games Lie
- Amdahl’s Law
- Blinn’s Law

Production Renderers

- Geometric Primitives & Geometric Complexity
- Texture Complexity & Quality
- Displacement
- Global Illumination
- Flexibility
- Programmable Shading
- Performance
- Image quality
- Robustness

Powerful Shading Language

- Rich data types, including strings, arbitrary-length arrays
- Refer to textures & coord systems by name
- Don’t want users exposed to hardware details (reduced precision, arbitrary limits, lack of loops)
- Need to call user-supplied code (DSO’s, etc.)
- Arbitrary type & number of user-supplied parameters
- Derivs better than fw-bw-difference
- Match “usual” programming model

Image quality

- Careful sampling & filtering
- NO visible aliasing is acceptable, in any dimension.
- No tessellation or other geometric artifacts
- Motion blur
  - Transformation and deformation blur.
  - No visible strobing or excessive noise
- Depth of field
- No excuses for everything looking like plastic
- Eschew all artifacts

Robustness

- Orthogonal feature set
- Must react well and predictably to unexpected input
- Must have vanishingly few bugs
  - —120,000 frames x 3M pels x multiple layers
  - —A 1-in-100,000,000 pixel crasher bug means thousands of unexplained crashes
- Must handle massive complexity
- Must scale gracefully, no hard limits
- Today's unreasonable input is tomorrow's trivial “toy scene”

Where GFX HW Succeeds

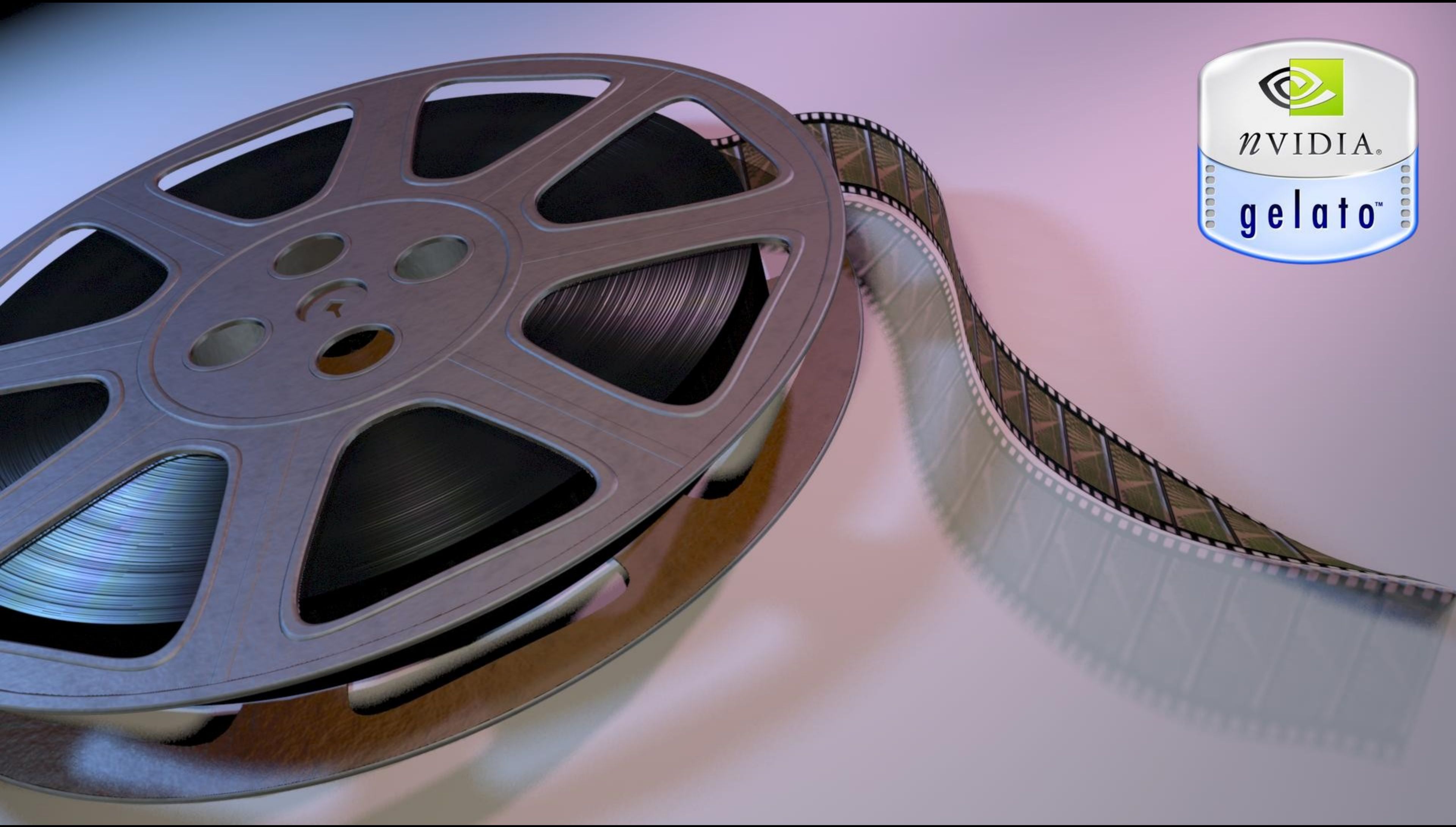
- Floating point throughout (almost)
- Some programmability (a whole lot more than there used to be)
- Rapid speed increase – doubling every 6-12 months

Where GFX HW Falls Short

- No high-order curved surfaces or procedural geometry, no pixel-frequency displacement
- Very inefficient for pixel- or subpixel-sized geom
- Antialiasing
  - Not enough samples
  - Not good enough filtering
  - No motion blur of depth of field
- Texture
  - Limited texture memory, size and number of textures

Where GFX HW Falls Short

- Procedural shading features:
  - Limited memory (especially writable memory)
  - Limited instructions
  - Crude derivatives
  - Too context-dependent
  - Lack of control flow
- Rapid versioning, little stability
- No late/lazy binding
- No back doors


nVIDIA
gelato™

Gelato Goals

- Implement full production rendering features
  - Curved geom, flawless antialiasing, motion blur
  - Ray tracing (GI, ambient occlusion, etc.)
  - No limits (tex, mem, etc.)
- Accelerate with HW wherever possible
  - But NEVER compromise on features, quality, flexibility
- Do not expose users to any HW limitations
  - Especially not require two sets of shaders
  - Stable, high-level interfaces
- Non-goal: real-time

Gelato Status

- 3 years in development
- Tested in studios since September 2003
- 1.0 Released April 04, 1.1 Nov 04, 1.2 Apr 05
- 2.0 beta soon
- Functionality roughly equal to leading SW renderers and diverging
- 2x faster than leading SW renderers and diverging
- Requires Quadro FX
- `film.nvidia.com`

Hard Design Choices

- Realtime vs. offline
- “Preview” vs. final frame
- In memory vs. scene larger than memory
- Limits & fast vs. no limits but higher overhead
- Allowing CPU fallback vs. requiring GPU
- HW shading language (Cg, etc.) vs. custom shading

GPU Programming Model

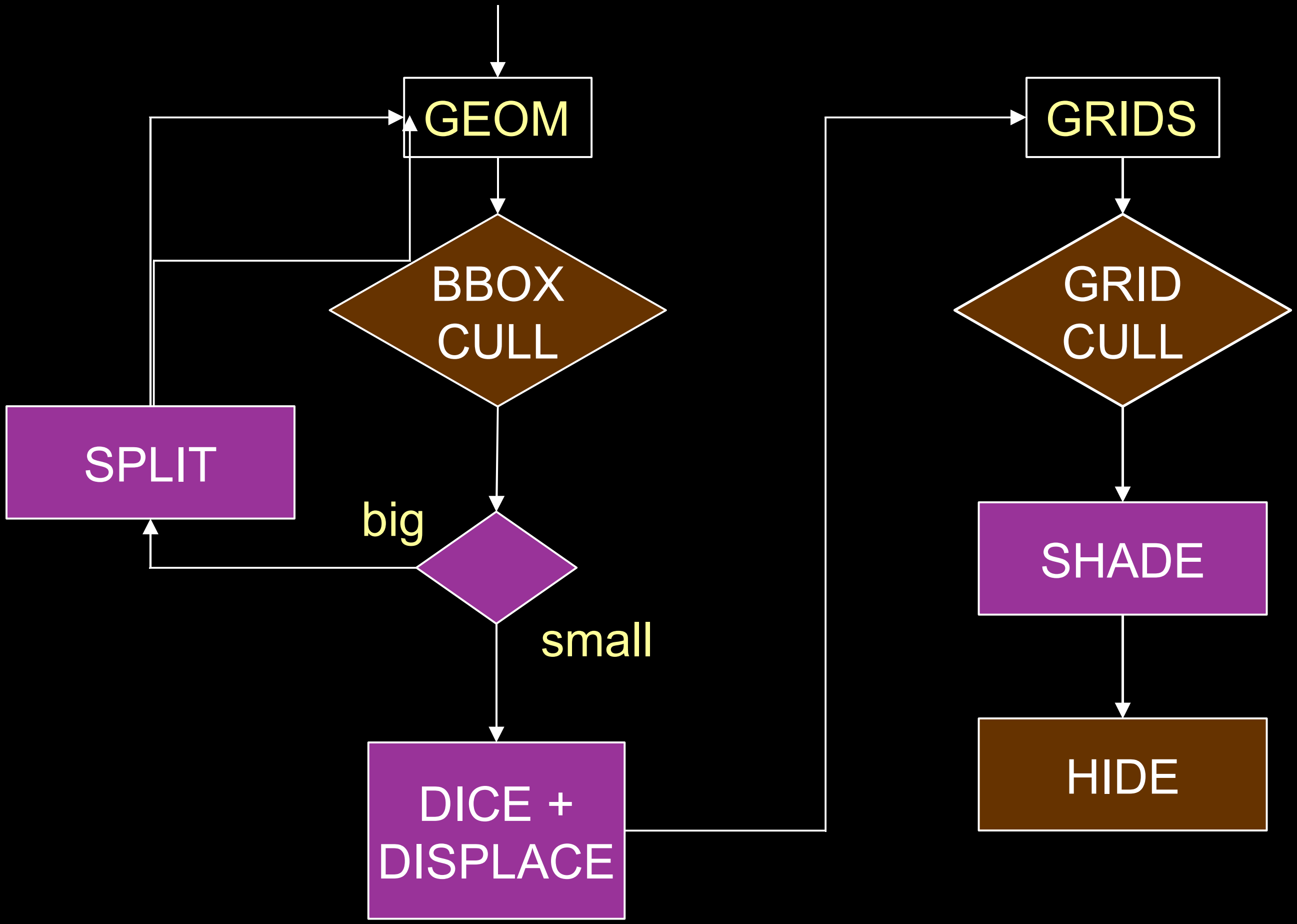
- Send scene geometry to GPU

  OR

- Send single camera-facing quad to GPU
  - Substrate for fragment programs
  - Think of GPU as fast parallel FP accelerator
- GPU programming is hard!
- Programming environment still not mature

REYES-like Architecture

SIGGRAPH2005



Hider Requirements

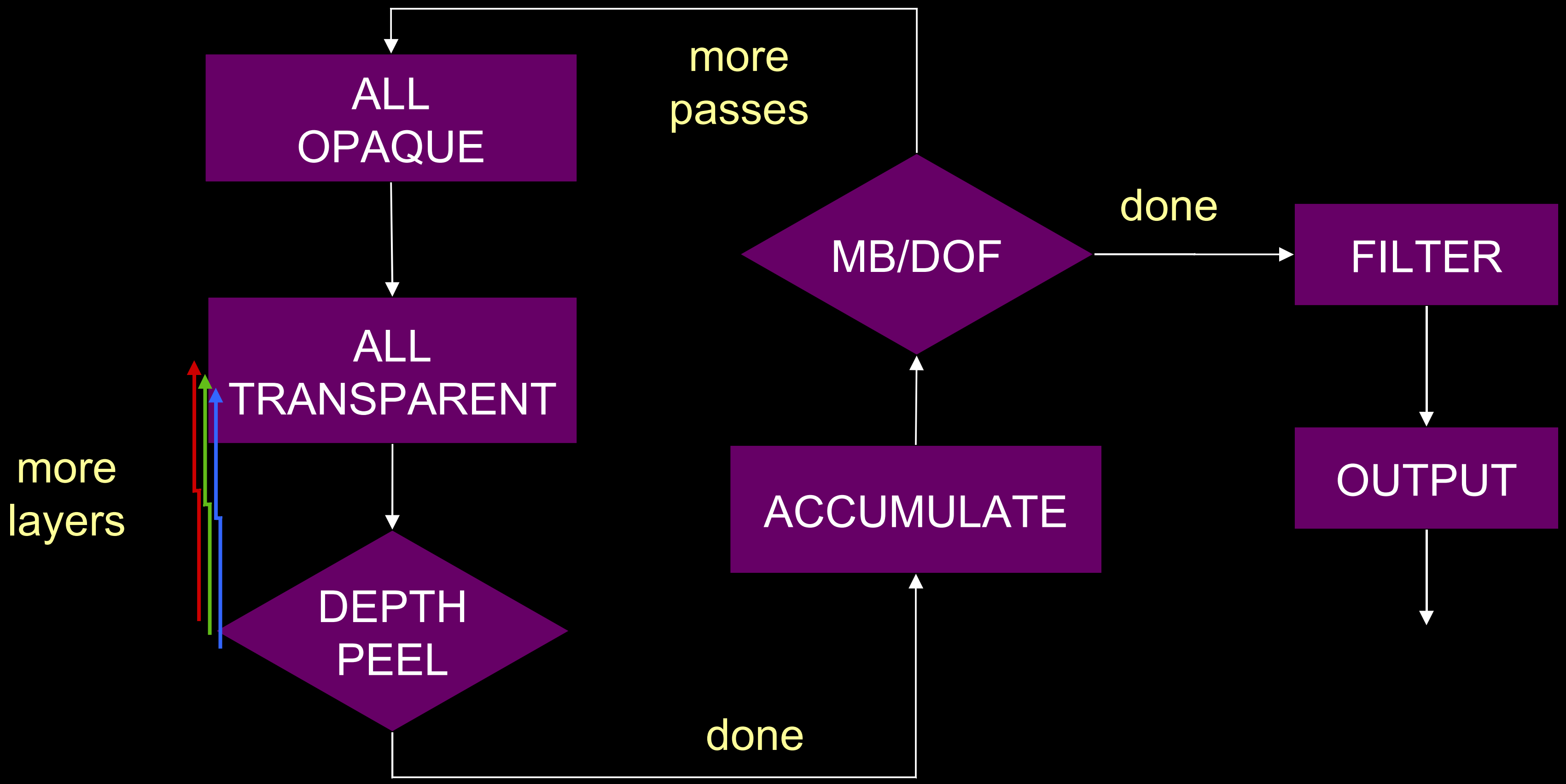
Comparable quality to CPU-only renderers:

- Depth of field and motion blur
- Transparency
- Filters with wide support
- Robustness for real production scenes
- Occlusion culling to avoid excess shading
- Spectral opacity and arbitrary outputs
- Complete feature set (shadows, GI, RT…)

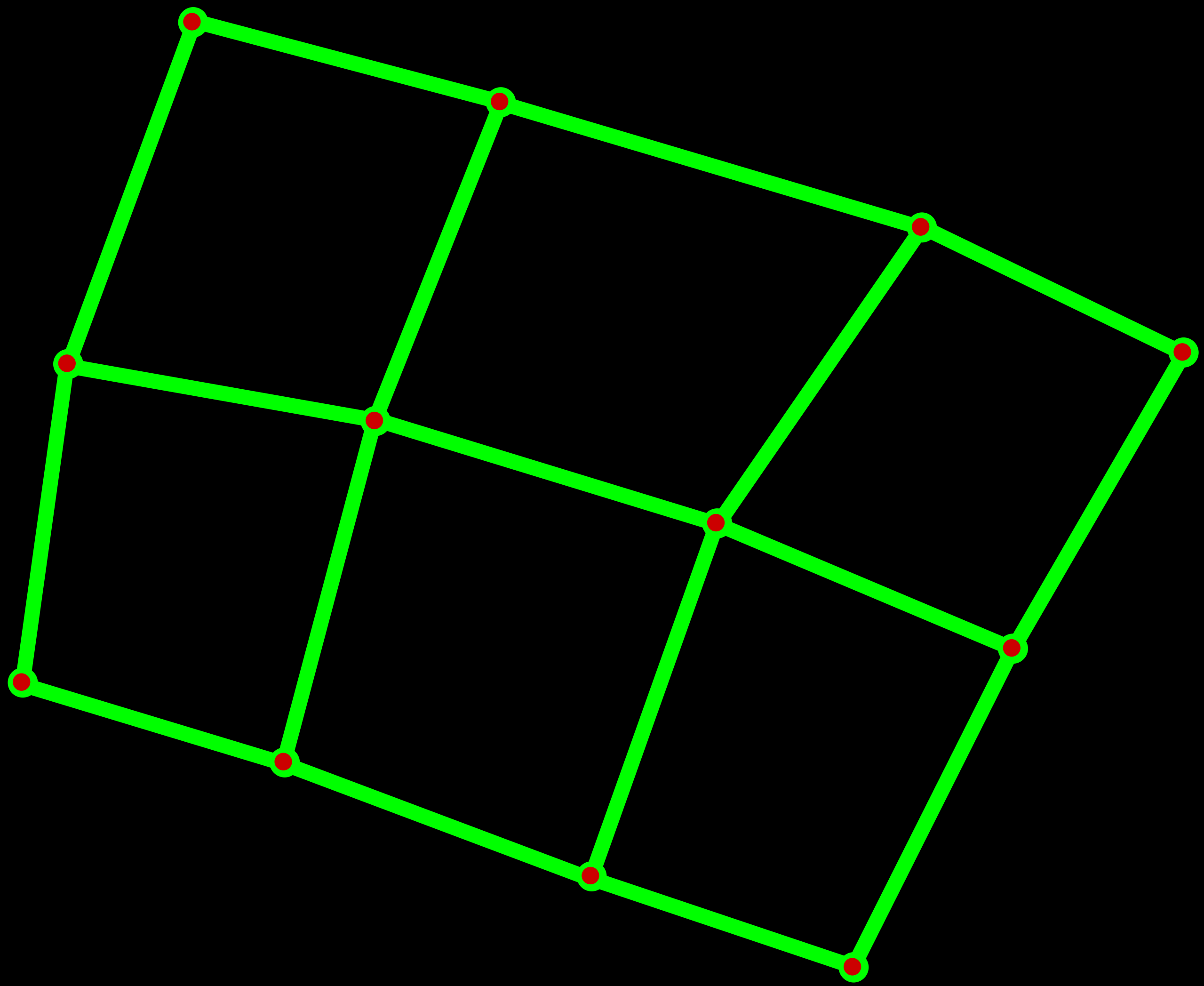
Hider Architectural Overview

- REYES-style geometry processing
- Supersampling for anti-aliasing
- Accumulation buffer for MB & DOF
- Enhanced depth peeling for transparency
- Two-pass downsampling for filtering
- Occlusion query for culling

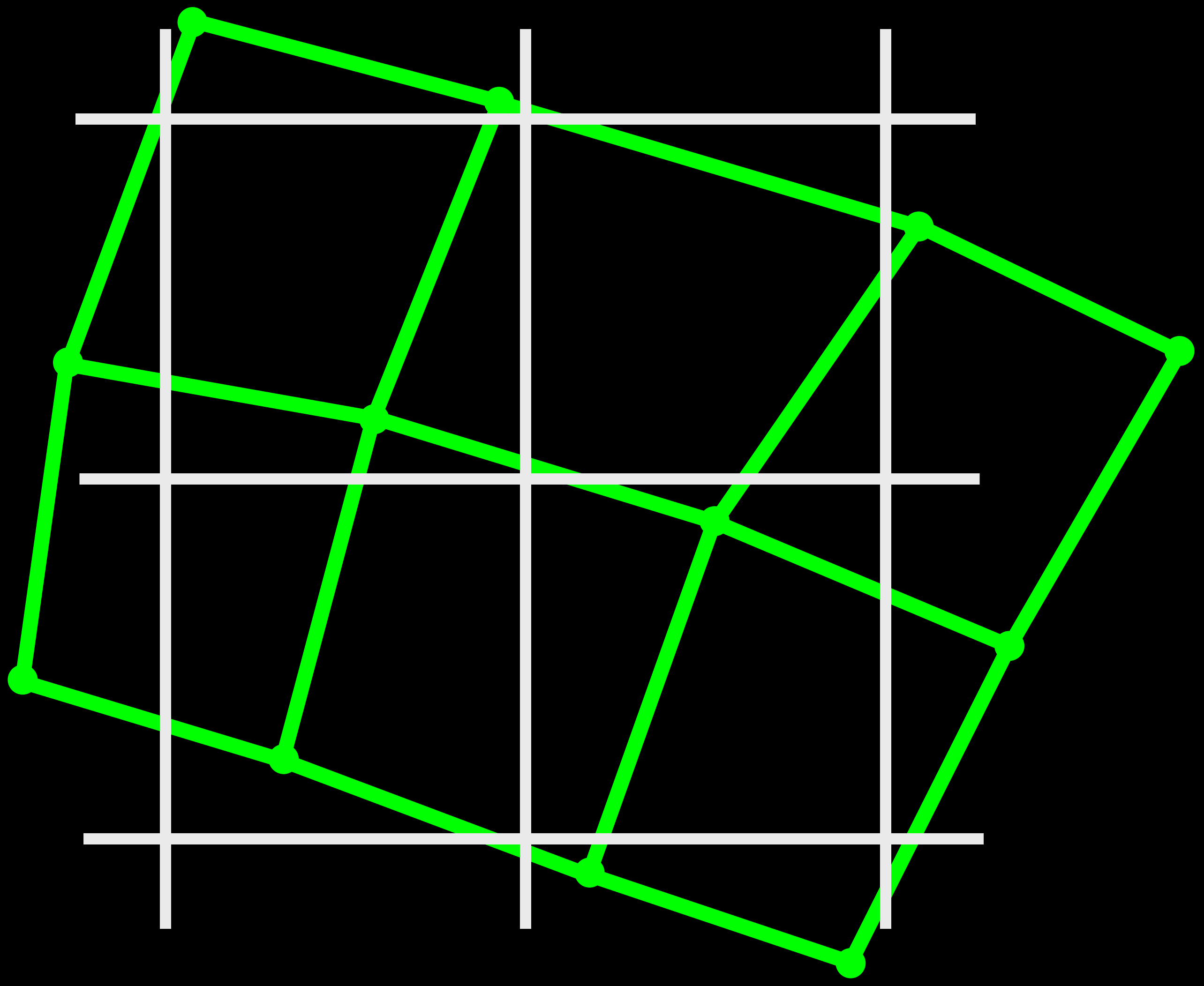
Hiding Algorithm



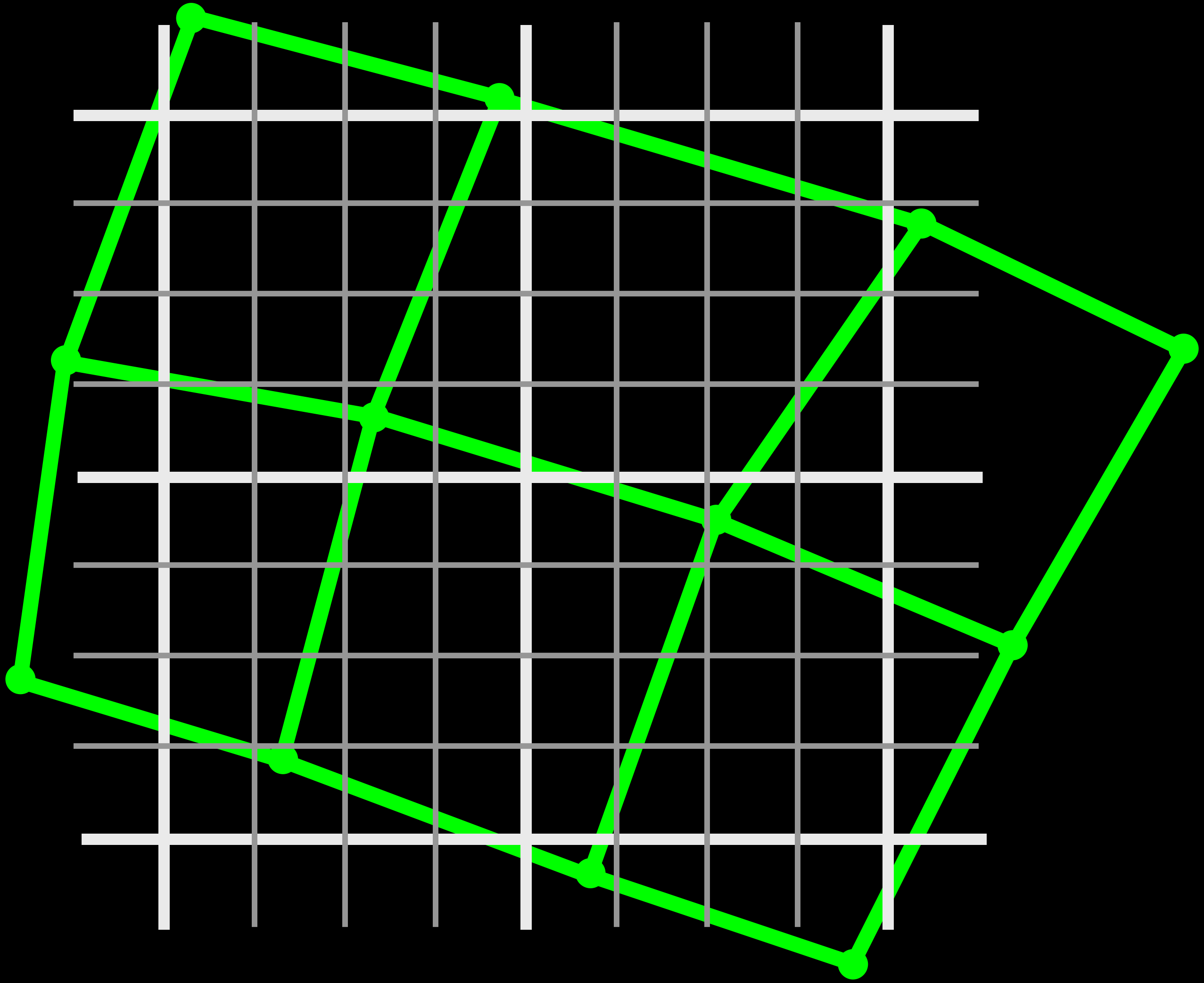
Grids, Pixels & Samples



Grids, Pixels & Samples



Grids, Pixels & Samples



Why Param-space shading?

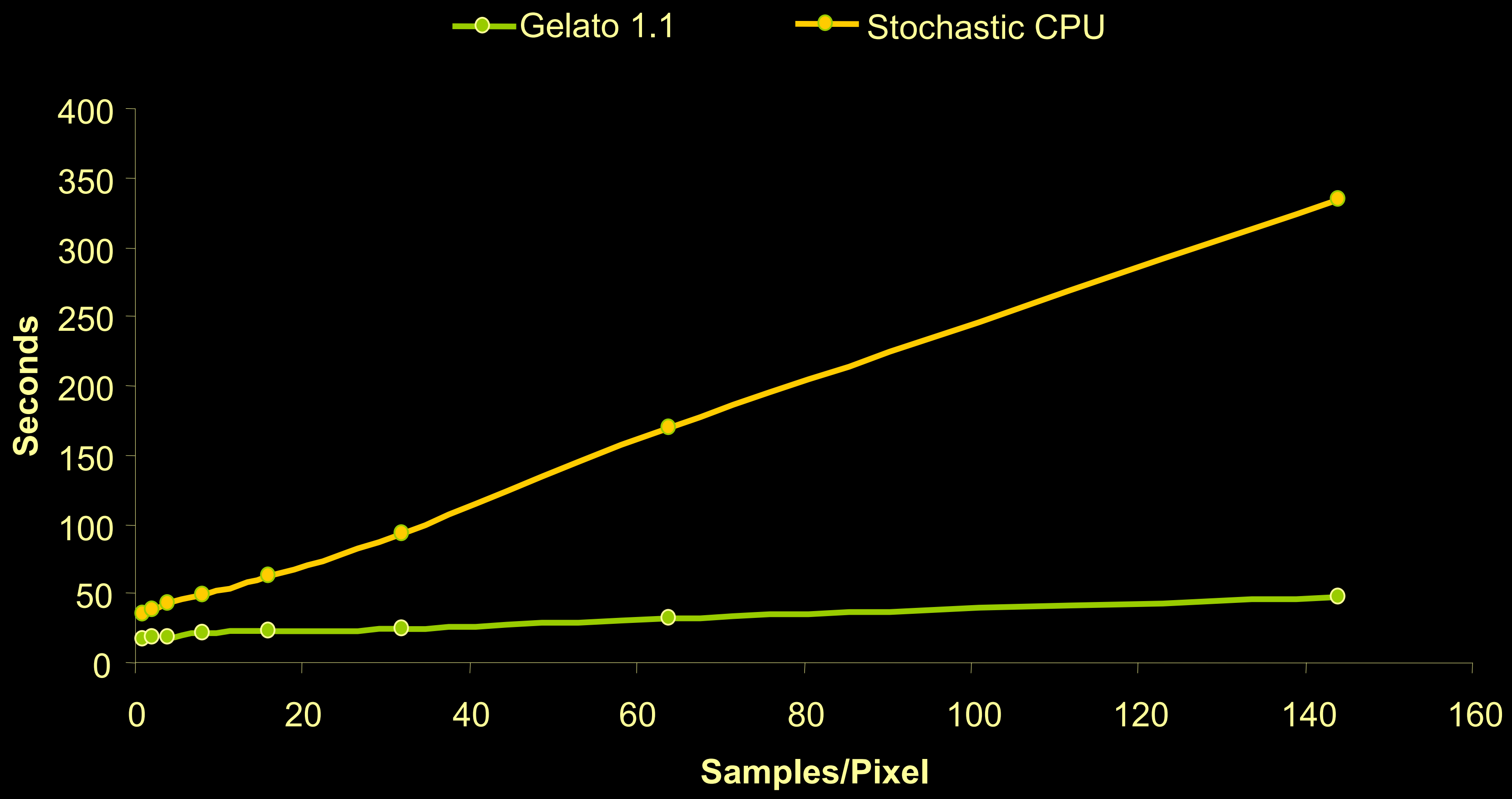
- De-couple shading and hiding complexity
  - Typical 64+ spatial x 64+ temporal samples x DP x SO
  - Marginal shading cost per sample is zero
- Good derivatives
  - Means good texture lookups, shader AA
- Displacement & surface/lights in same space
- More stable in motion
- Can use CPU when GPU unable to do shading operations

Ethan Summers & Shiew Yeu Loh

Ethan Summers & Shiew Yeu Loh

Performance

Spatial Samples



Temporal Samples

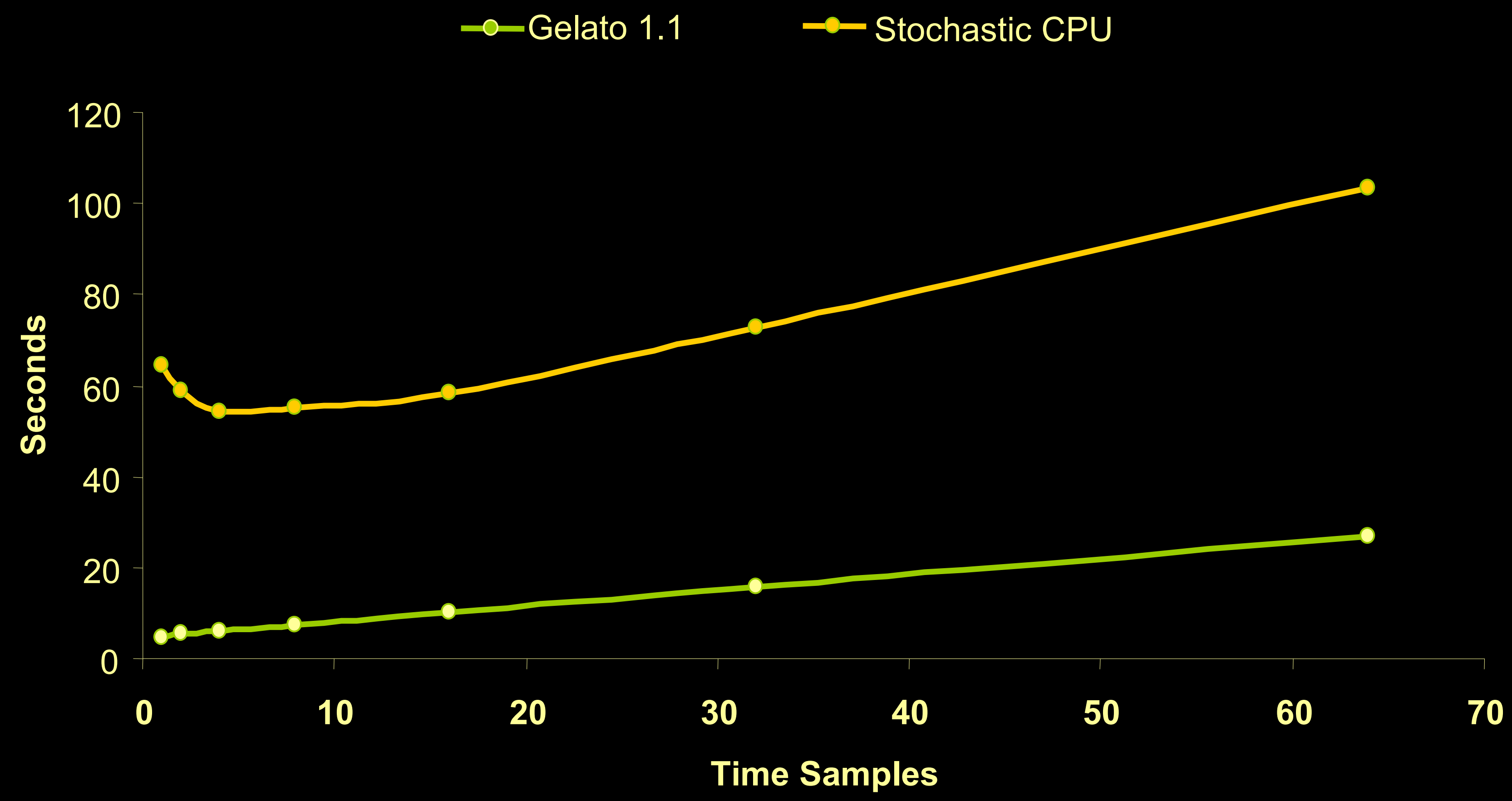


Image quality: spatial

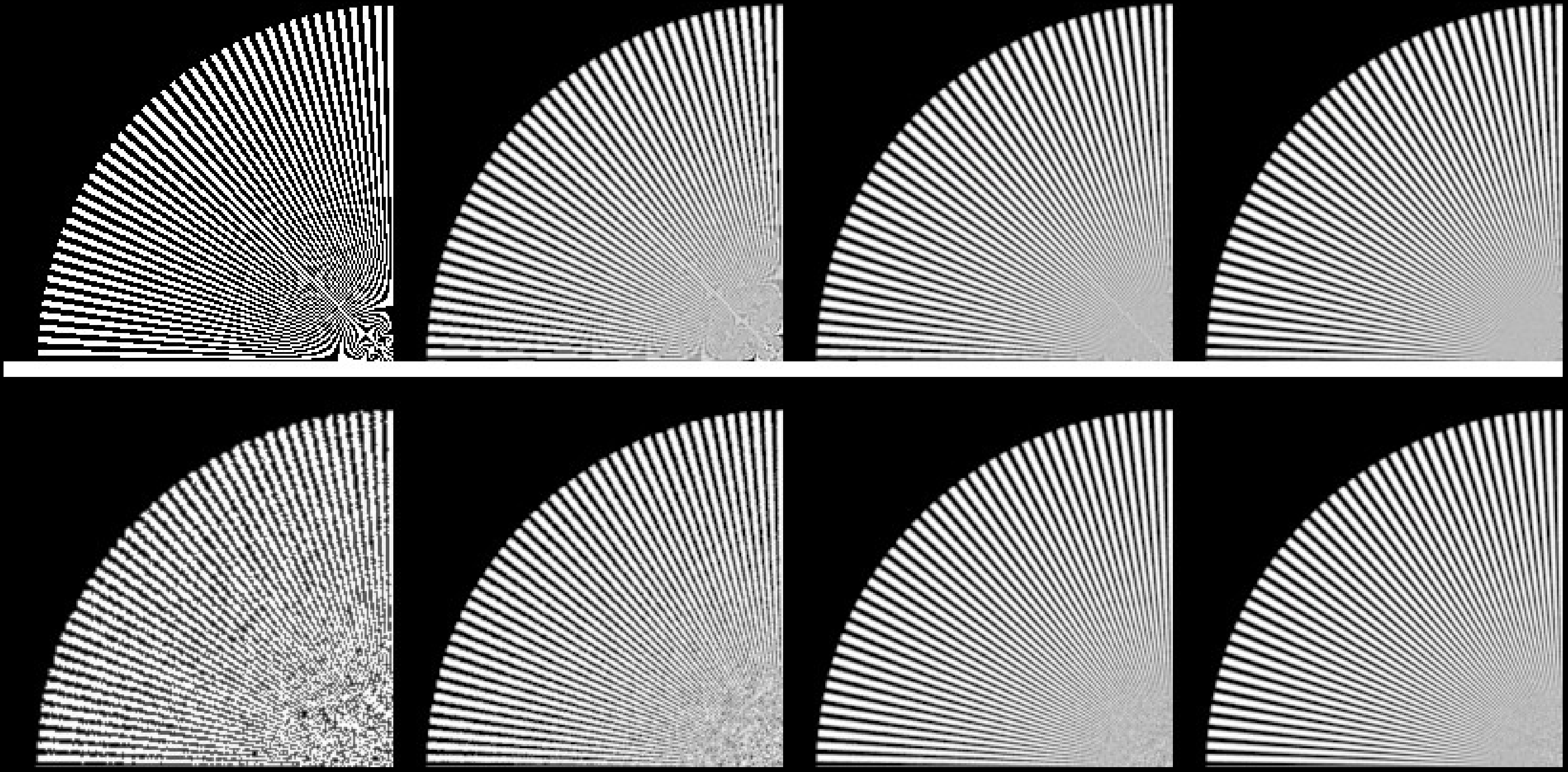
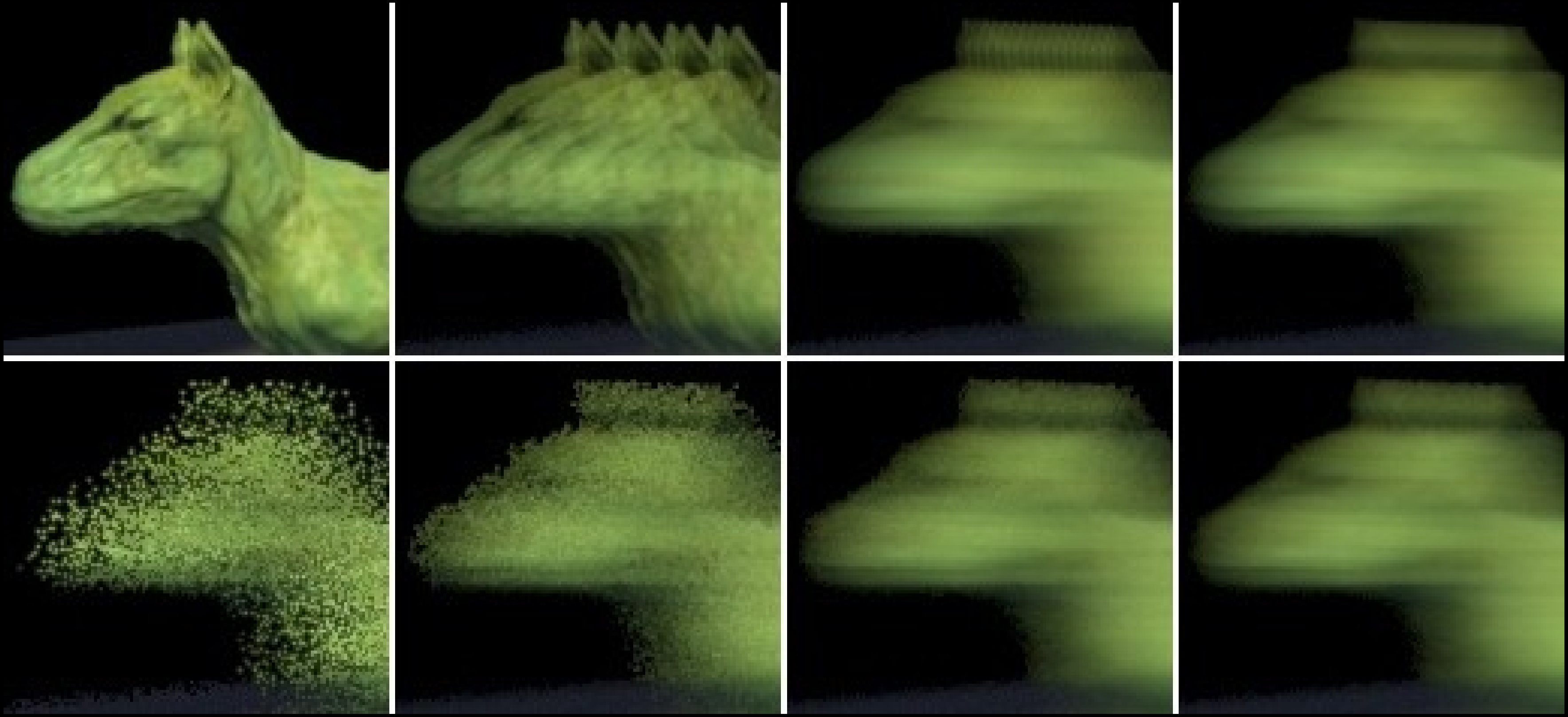


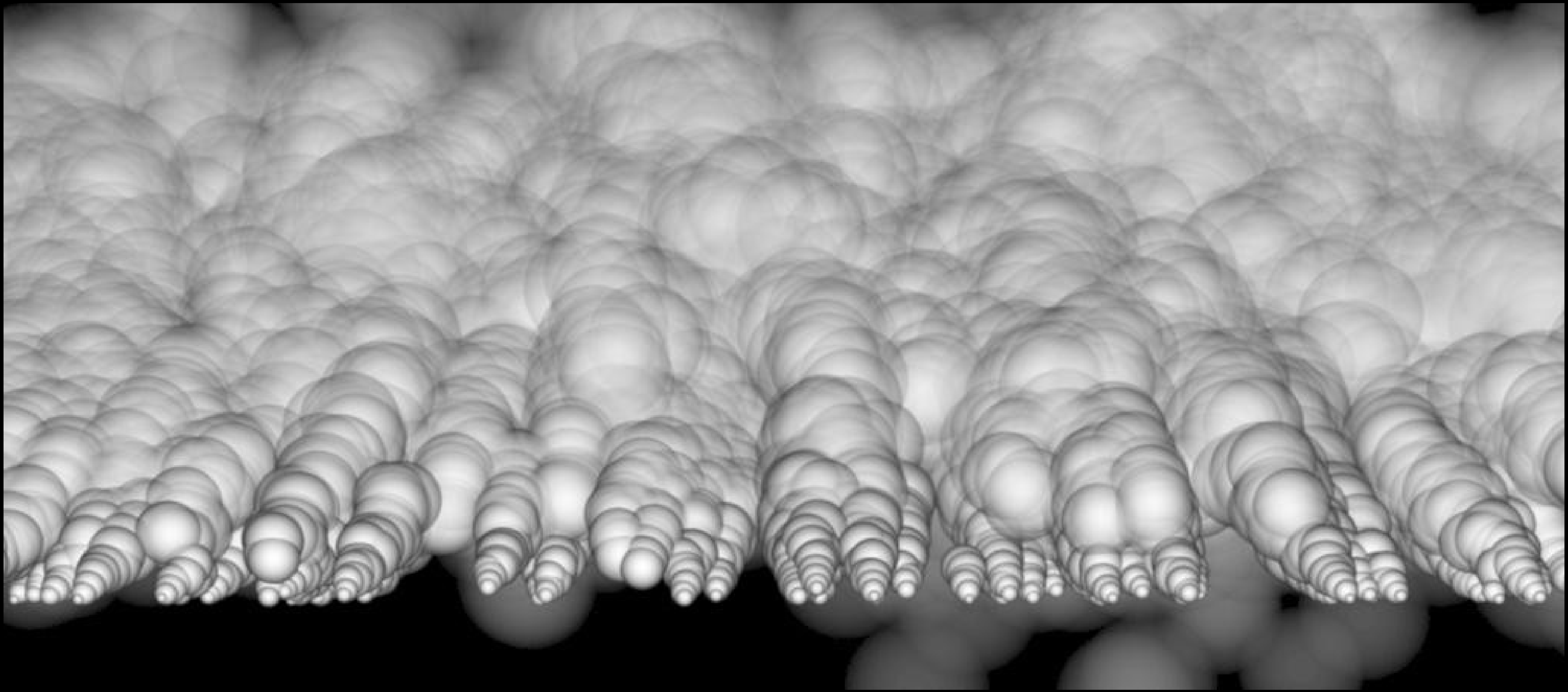
Image quality: temporal



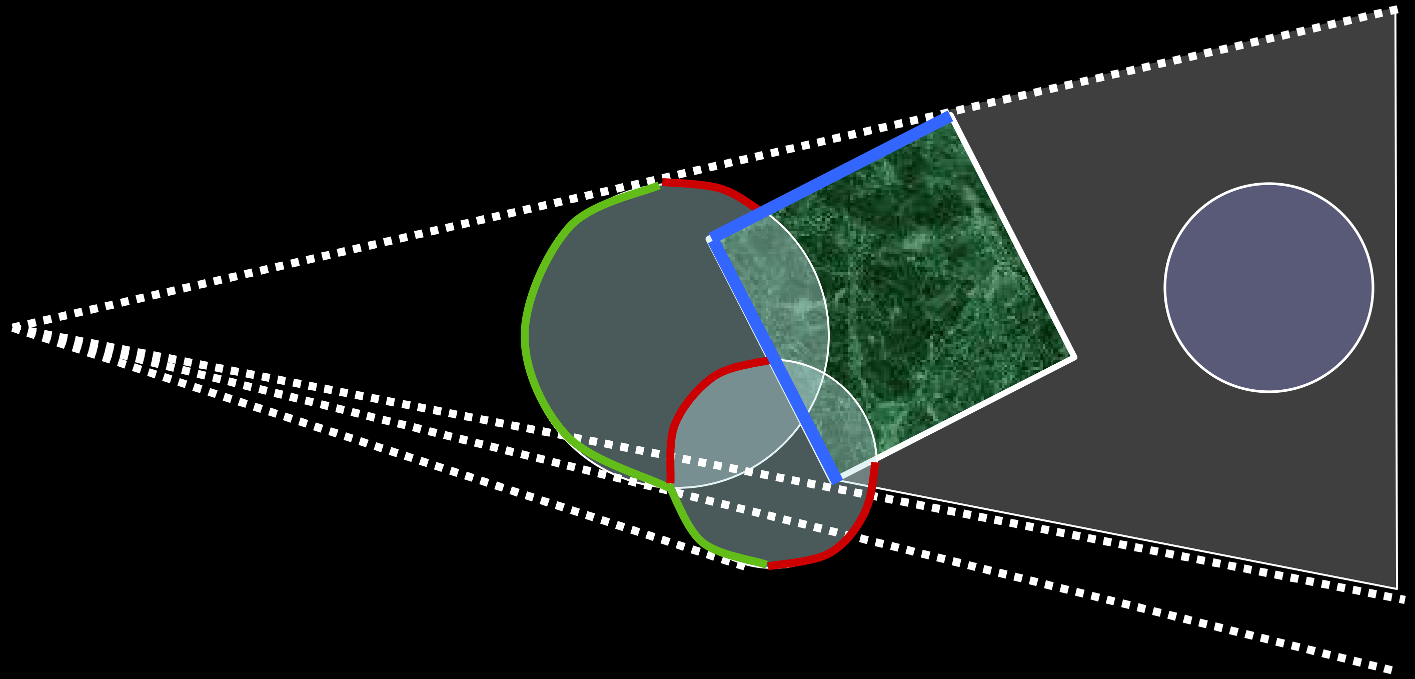
Depth peeling

- Render one pass for each visible depth layer
  - Use previous z buffer to mask closer surfaces
- O(n^2)

Transparency

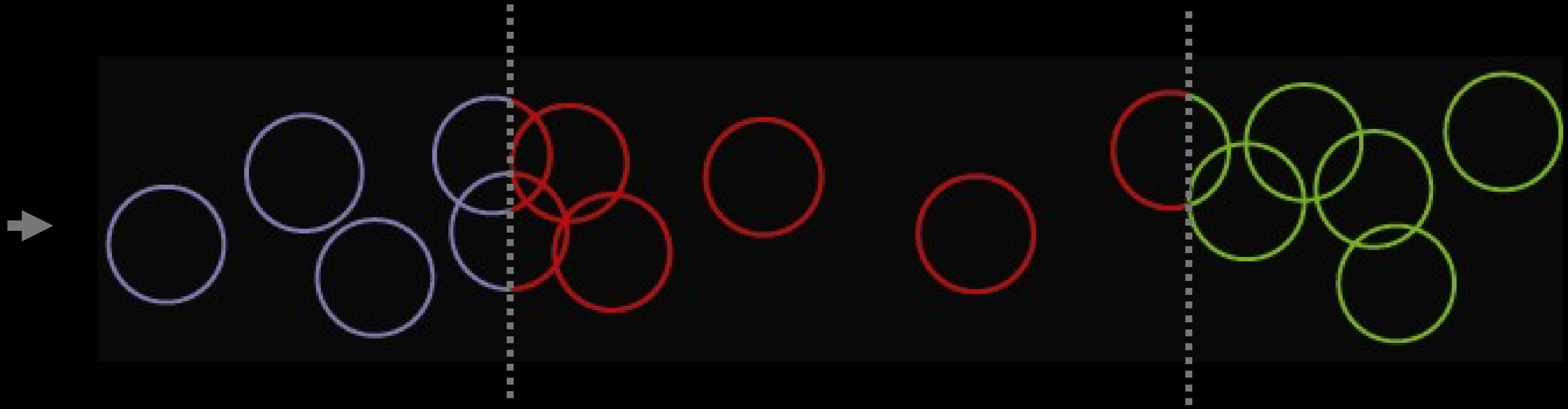


Transparency



- Opaque pre-processing
  - One additional texture-z test
  - Reduces number of depth peeling passes
  - Occlusion culling remove hidden surfaces

Z-Batches

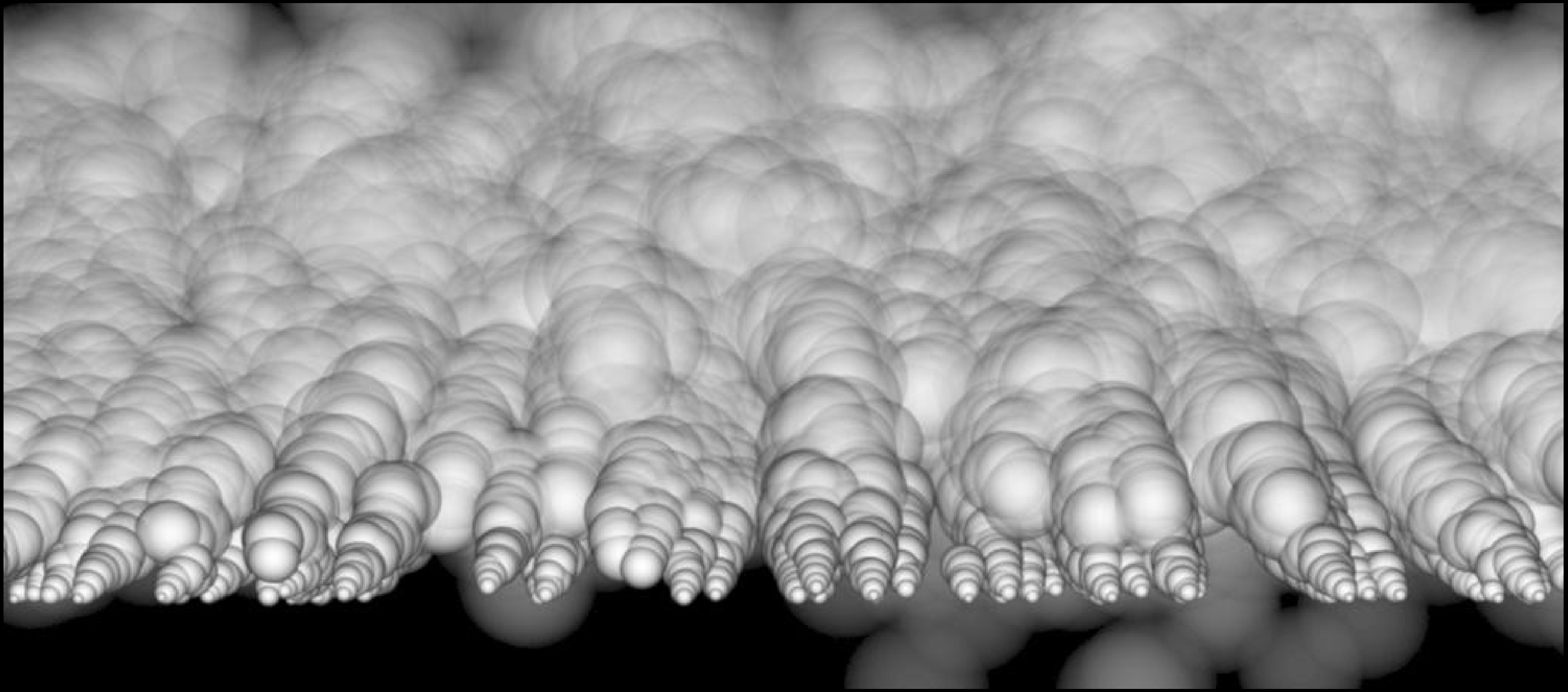


- For N grids processed in batches of B grids:

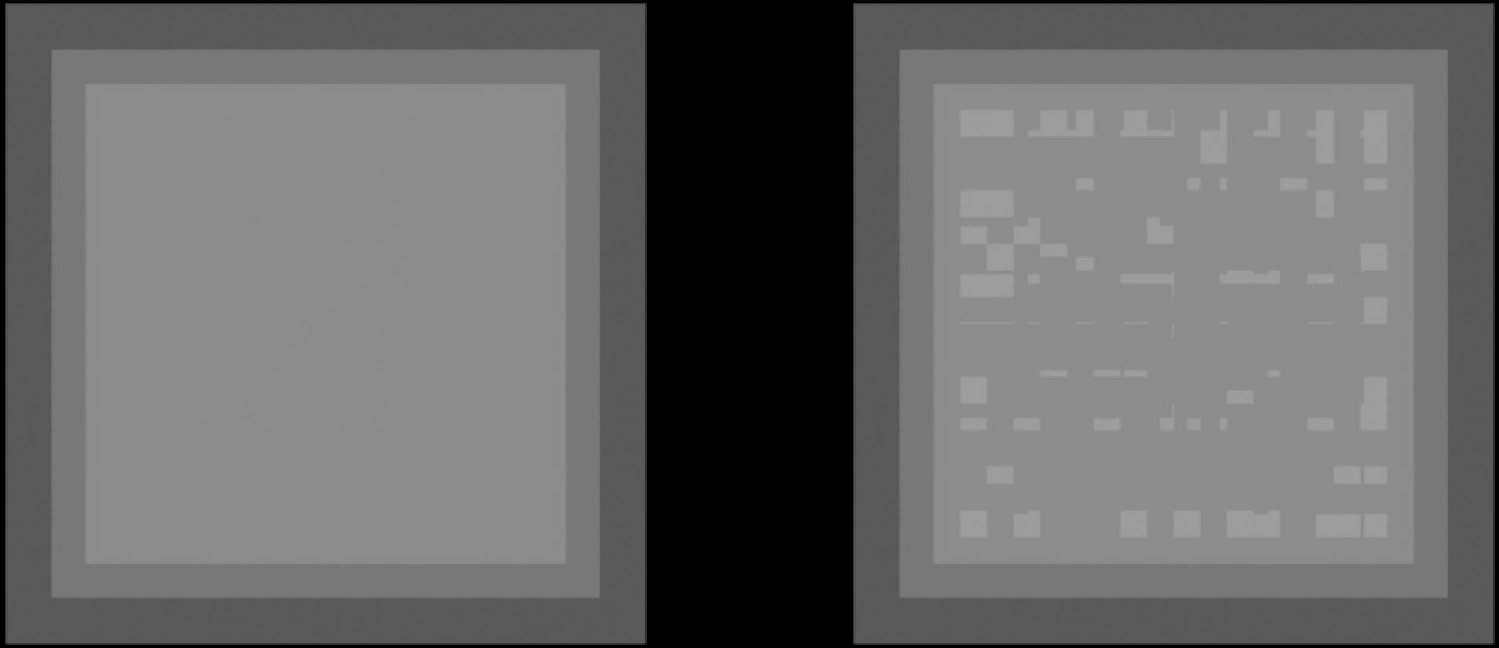
  $O((N/B)B^2) = O(BN) = O(N)$

- Problem: grids overlap into multiple batches
- Opacity thresholding between batches

Transparency



Transparency Artifacts



Poor Performance Cases

Extensions

- Two-pass depth peel for average-z
- Volumetric shadow map generation
- Multiple camera (stereo) rendering
- Workqueue-based latency hiding
- Adaptive motion and DOF sampling
- Fast Relighting

Challenges

- Hiding Latency
  - Occlusion Query
  - Orthogonal computations
- Hybrid Algorithms
  - Batch size vs. excessive computation
  - Starving and Readback
- Programming Environment
  - Debugging and profiling
  - Support and stability

GPU Programming – Sort Values

SIGGRAPH2005

CPU

- Five minutes to write
- Every programmer in the world understands the code
- Complex sorts are well understood

GPU

- Days or weeks to write
- Code is hard to read
- Won't run next year
- Best methods <u>not</u> well understood yet

nVIDIA GELATO

Future

- GPU’s will be standard, even on compute farm
  - Like FPU: once exotic, now essential
- Many apps will use GPU’s
  - Even things by far not interactive graphics
  - Compositing
  - Cloth and fluid simulation
- Stop looking at CPU vs. GPU
  - Think about von Neumann AND Stream processor
  - Divide work naturally, load balance
  - Try to leave no computational resource unused

More info

- Two-Pass Filtered Downsampling
  [Wexler, Enderton ’05] Graphics Gems II
- Gelato at NVIDIA Booth
- `film.nvidia.com`
- `lgritz@nvidia.com`

Q&A

SIGGRAPH2005

nVIDIA
GELATO