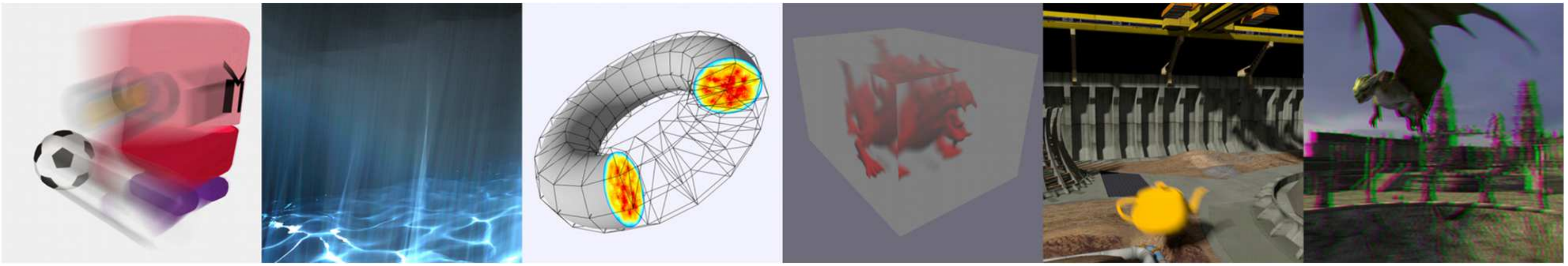


Design and Novel Uses of Higher-Dimensional Rasterization

Jim Nilsson, Petrik Clarberg,
Björn Johnsson, Jacob Munkberg,
Jon Hasselgren, Robert Toth,
Marco Salvi, Tomas Akenine-Möller

Intel Corporation

Lund University

Background & Motivation

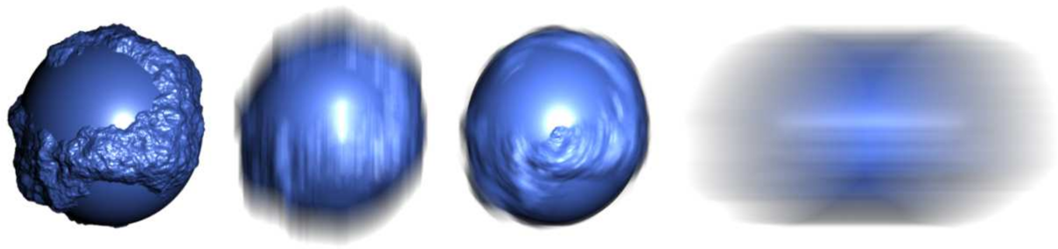
- Brainstorming experiment: what could we do with a fast(!) 5D stochastic rasterizer in HW?
- Apart from the obvious; MB and DOF
- 70+ ideas generated, not all survived
- Limitations in study: lack of detailed performance numbers (no HW numbers!)

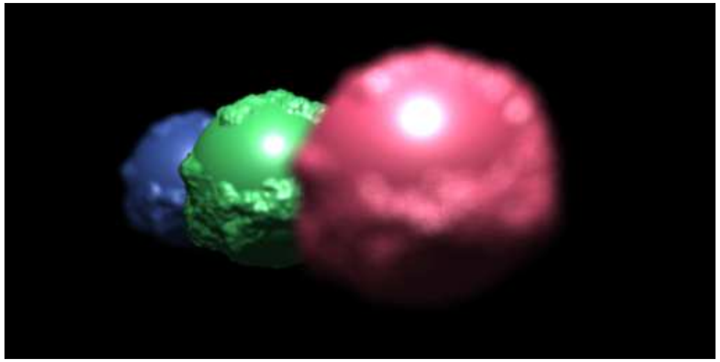
Inspiration and motivation for hardware and API design

Contributions

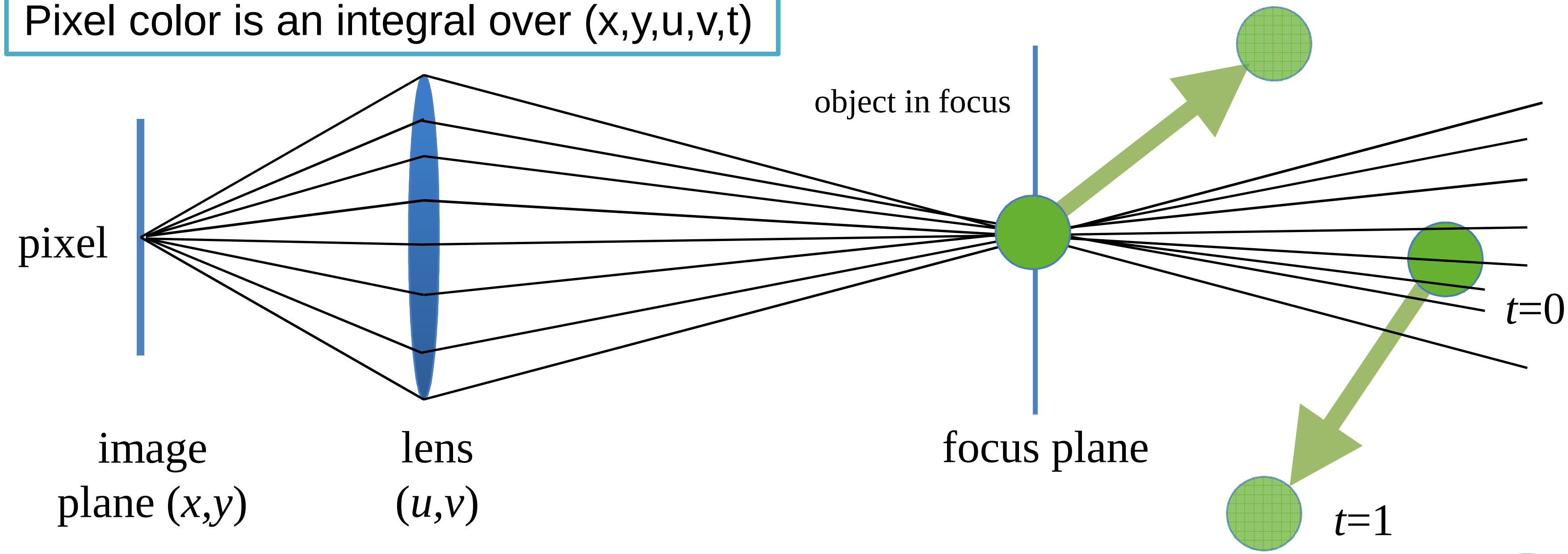
- Design space of new applications for higher-dimensional rasterization
- A coherent model for efficient stochastic rasterization
- Pin-pointing practical design aspects, e.g., importance of conservative rasterization and flexible sampling
- Motivation for further research in uses, hardware, and API support, for stochastic and higher dimensional rasterization

5D Rasterization

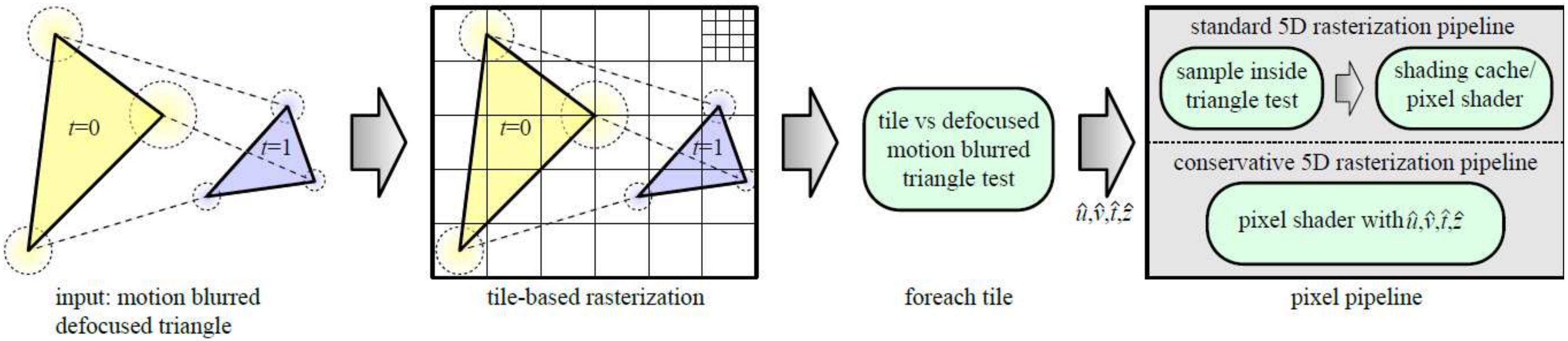




Pixel color is an integral over (x,y,u,v,t)



A Five-Dimensional Rasterization Pipeline



- Two pipelines
  - Standard: each sample inside-tested and shaded
  - Conservative: intervals sent to pixel shader (per tile)
- Research areas, all important:
  - Efficient stochastic rasterization
  - Efficient reuse of shaded points
  - Filtering/reconstruction to reduce noise
  - Bandwidth reduction

Applications

Standard pipeline	Conservative pipeline
Sampling	5D occlusion queries and continuous collision detection
Motion blurred shadows	Caustics
Stereo and multi-view rendering	Glossy reflections and refractions

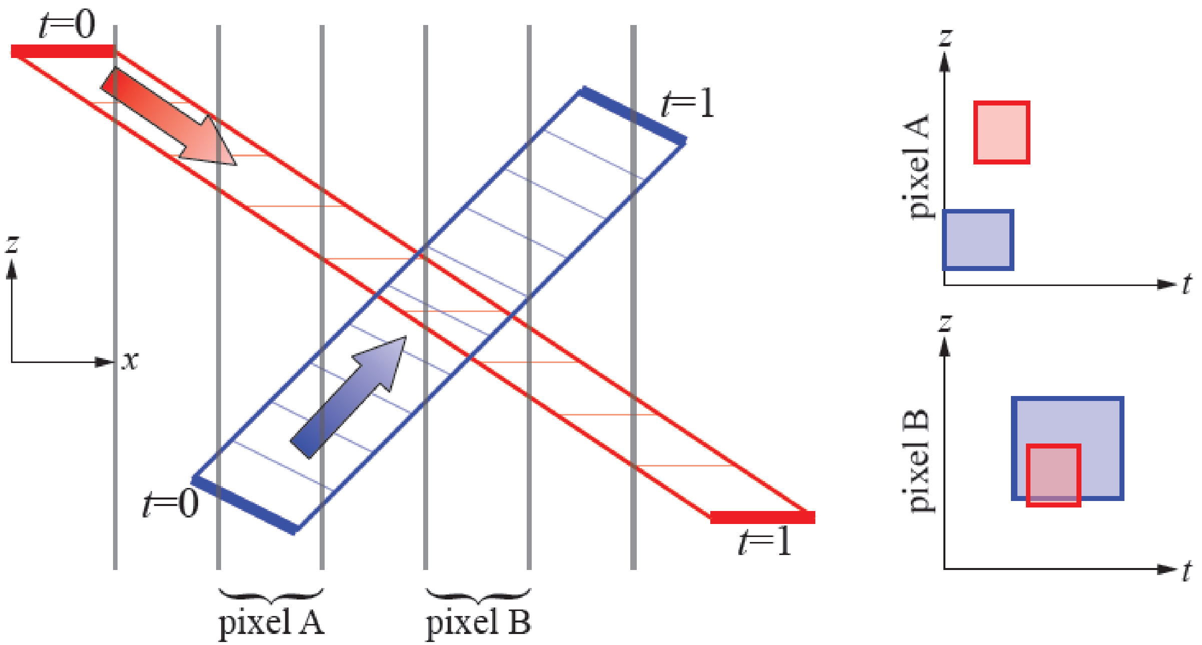
Five-Dimensional Occlusion Queries

- *Sample-based*: use (many) samples, count #fragments passing depth test
  - Generalization of 2D occlusion query

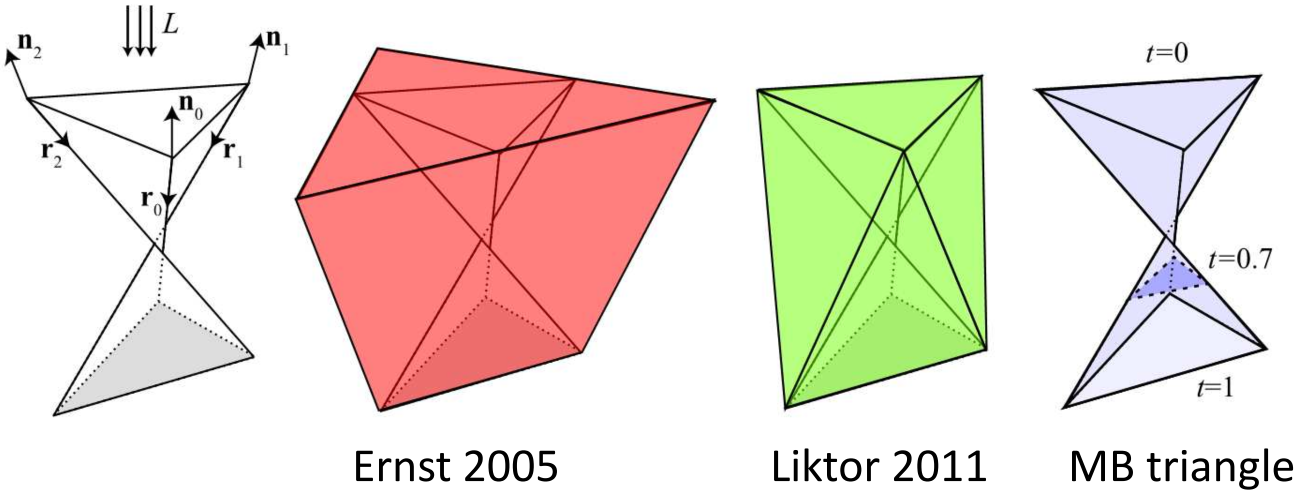
- *Interval-based*: hierarchy of u,v,t,z intervals to cull regions in time/lens
  - Compare z interval of triangle to $z_{min/max}$ for *uvt*-box
  - No sample-inside-triangle test
  - No per-sample depth computation
  - Conservative: no false positives

Continuous Collision Detection

- Use 3D time-dependent occlusion query to avoid ”tunneling” effects in static CD
- Build **Potentially Colliding Set** [Govindaraju et al.]; cull non-colliding objects from detailed triangle-triangle intersection tests

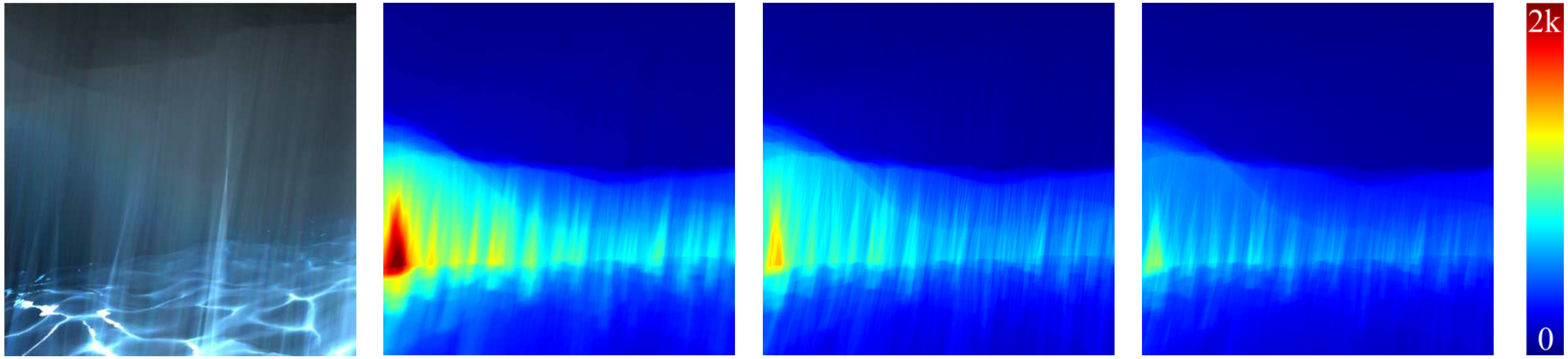


Caustics Rendering



- Insight: warped caustic volume = moving triangle
- Ernst et al. used prisms to bound bilinear side patches
- We use 3D rasterization HW and conservative test per 1x1 pixel

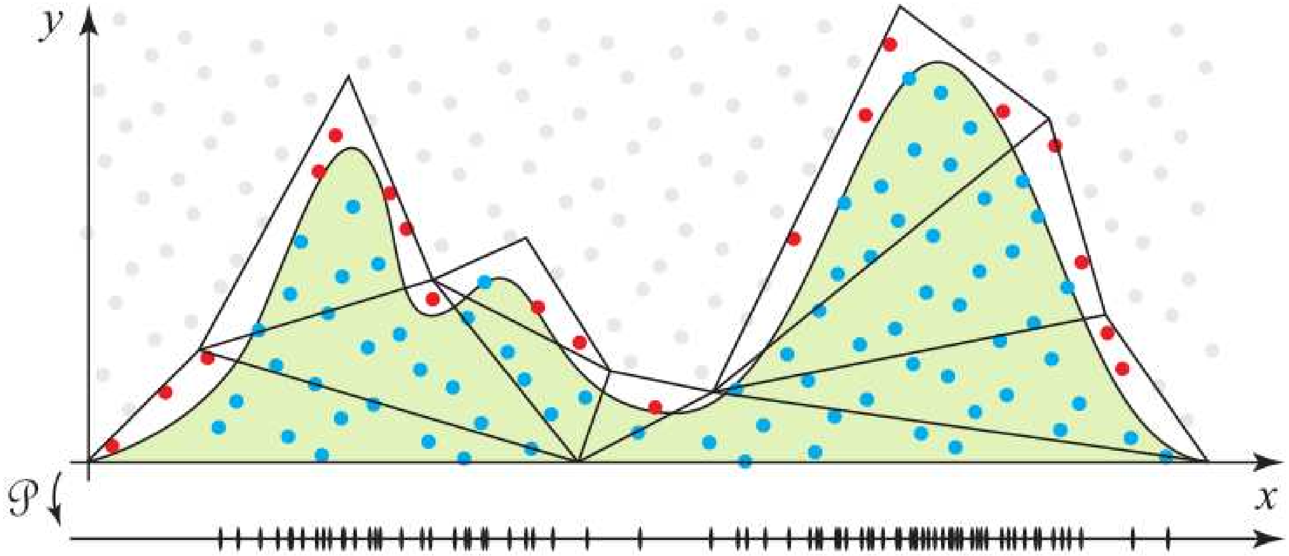
Caustics (contd.)



Ernst 2005 (100%) Liktor 2011 (79%) Ours (65%)

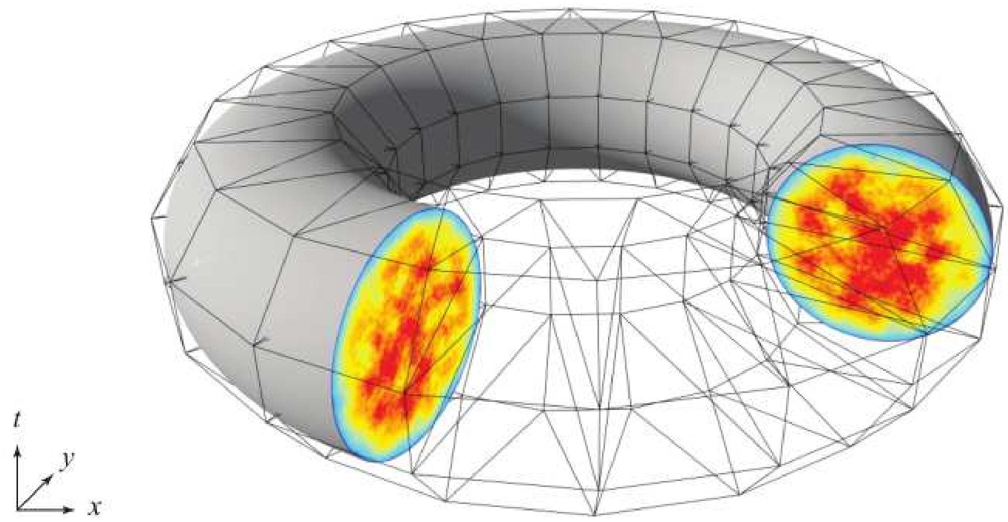
- Heat map showing number of pixel shader executions
- More efficient in terms of traversed pixels, Z_{min}/Z_{max} culling, and pixel shader complexity
  - Use bounds for light attenuation function

Sampling



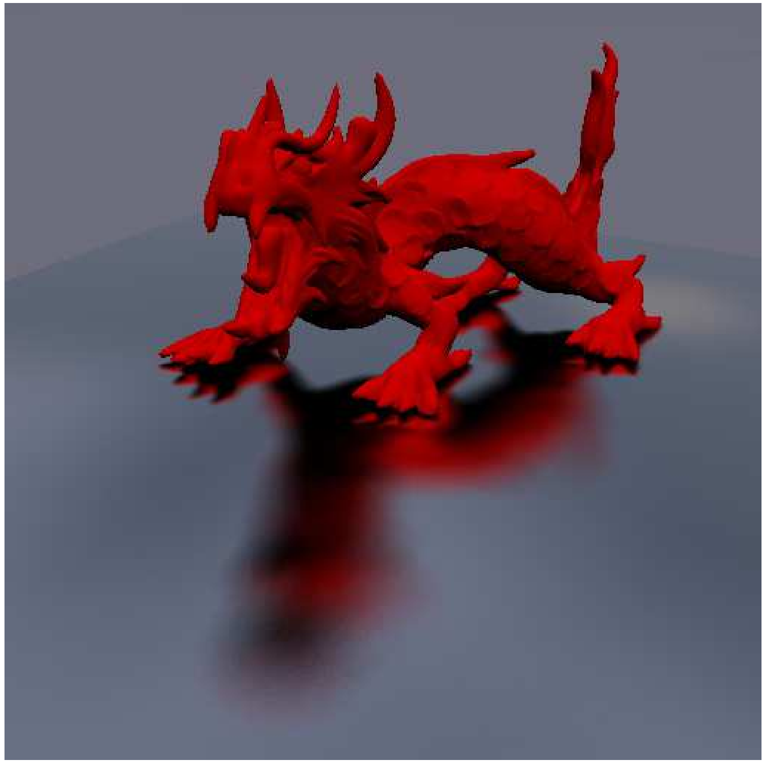
- Example: non-uniform sample generation (white noise)
  - Create samples in (x, y)
  - Samples outside triangles quickly rejected
  - Limit by density function to create height field *(x, p(x)) in* $\mathbf{R}^{N+1}$
  - Project to $\mathbf{R}^N$

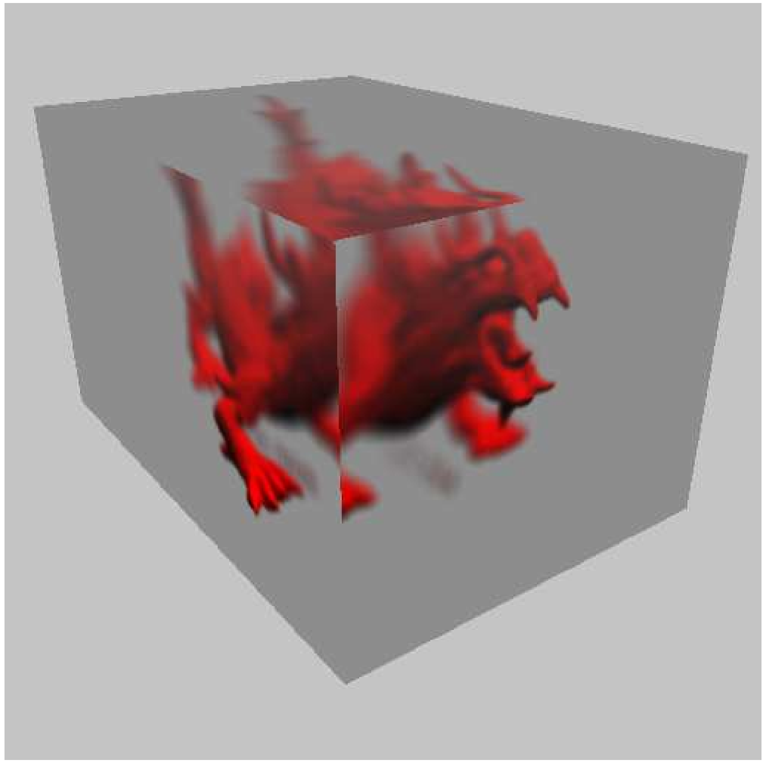
Sampling (contd.)



- Example: numerical integration
  - Rasterizer as volumetric sampler
  - Bounding volume of torus (motion blurred triangles)
  - Each primitive rasterized, samples in *xyt*-space
  - Inside-test and heat function integration in pixel shader

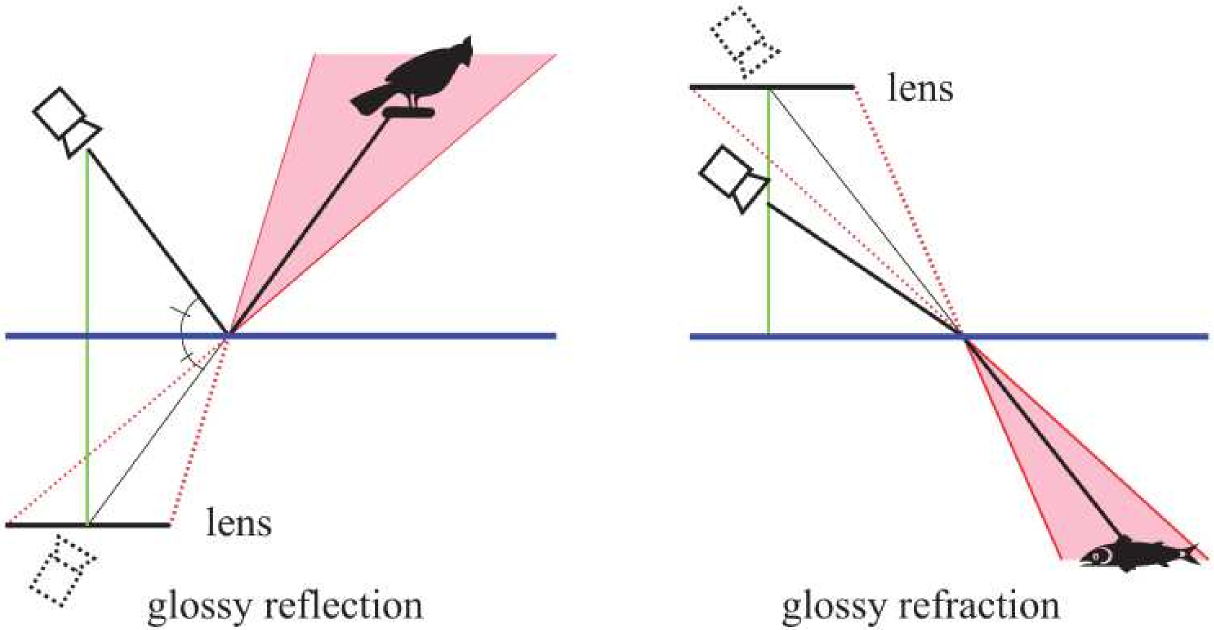
Planar Glossy Reflections & Refractions





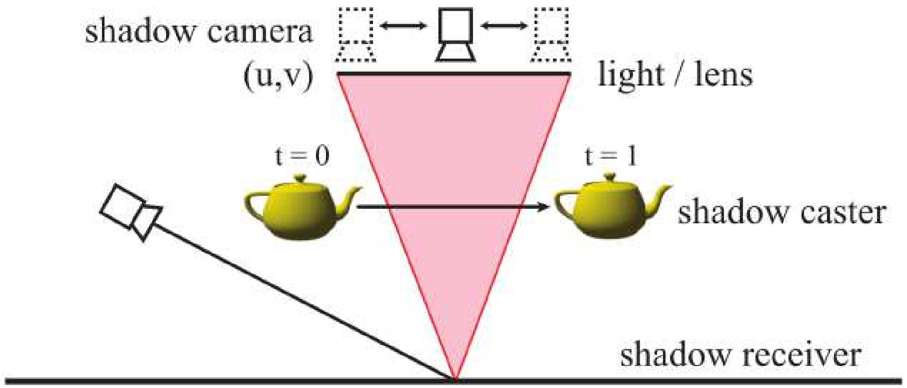
Glossy (contd.)

- Idea: Render entire frustum of reflected rays for each pixel on surface using DOF mechanism
- Approximate amount of refraction
  - But plausible appearance



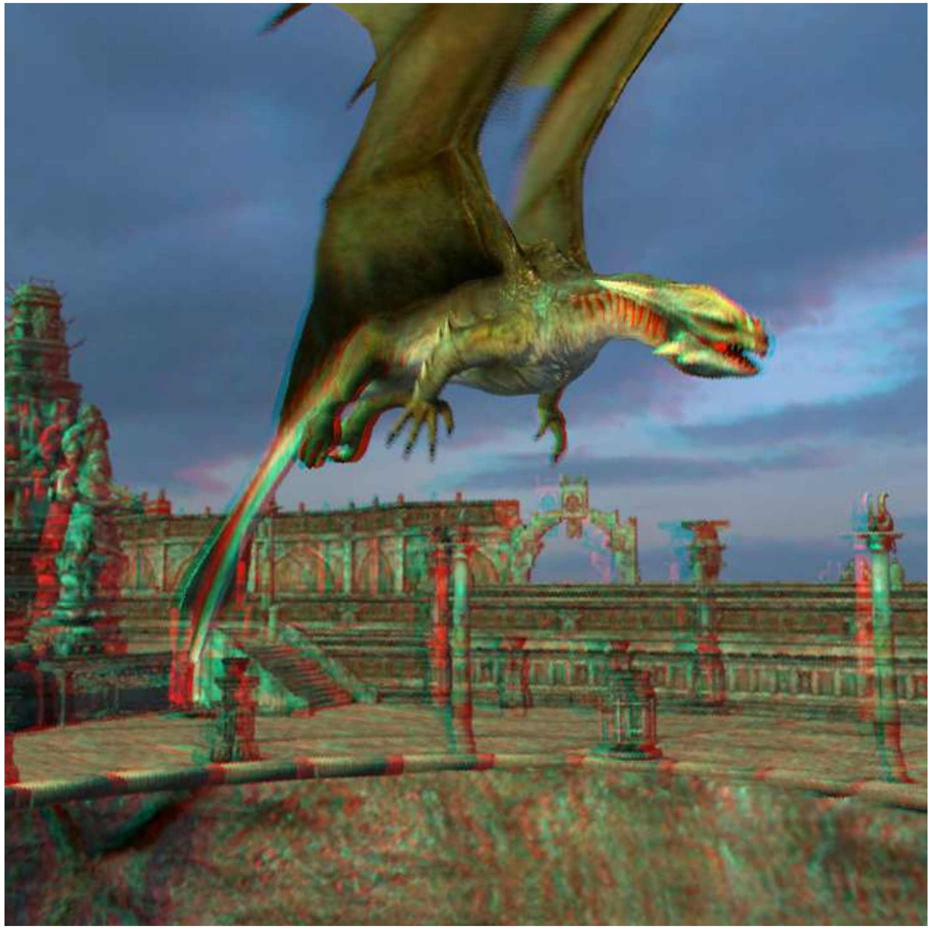
Motion Blurred Soft Shadow Mapping

- Idea: Use 4D rasterization to render a shadow map from an area light source in one pass
  - When rendering lighting, search through the shadow map to find N rays ”closest” to the point to shade



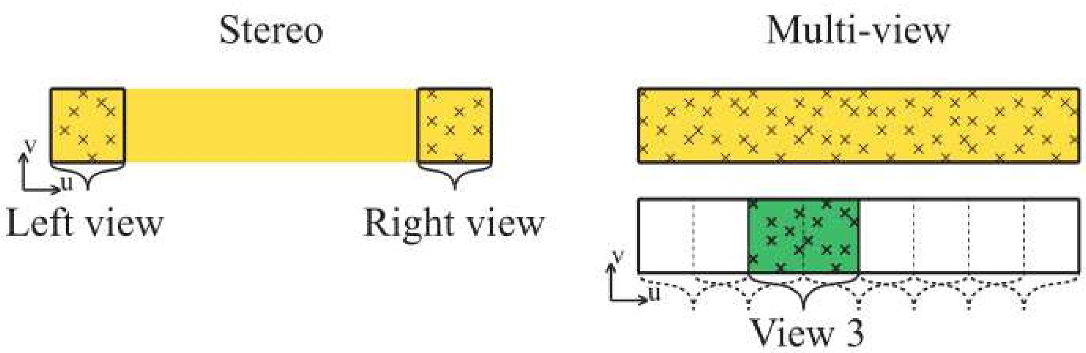
- Lookup is potentially very expensive
  - Especially if the shading point is out of focus.

Stereo and Multi-View Rendering



Idea: Use 5D rasterization to directly render for stereo-/multi-view screens

Multi-View (contd.)



- Filter shaded 5D samples on u and/or v, and write to different render targets
  - Reduces vertex shader execution
  - Can reduce texture bandwidth[1]
  - Reduce workload for developers (stereo just works)
  - Can render images with DOF and MB and stereo
- Shader caching works well
  - 10 % extra shading for stereo, 15 % for seven-view

[1] *An Efficient Multi-View Rasterization Architecture*, Hasselgren and Akenine-Möller

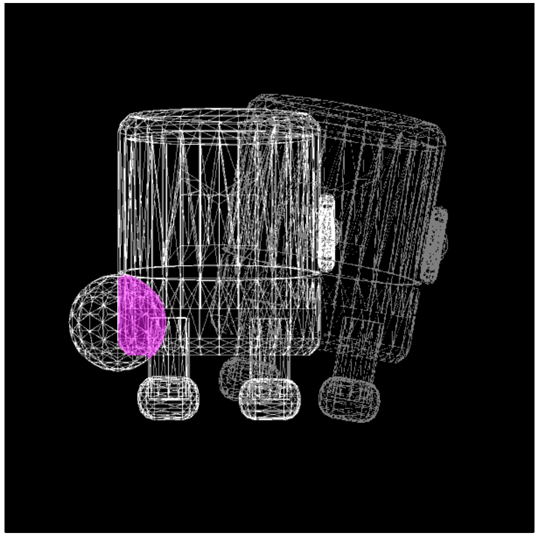
Conclusions

- Still many open questions on how to build a 5D rasterizer, BUT…
- A number of future-looking use cases
  - Time & lens bounds for culling
  - Beam interpretation of time-continuous triangles
  - Efficient sampling of arbitrary volumes
  - Single-pass algorithms, shader reuse
- Practical design aspects and motivation

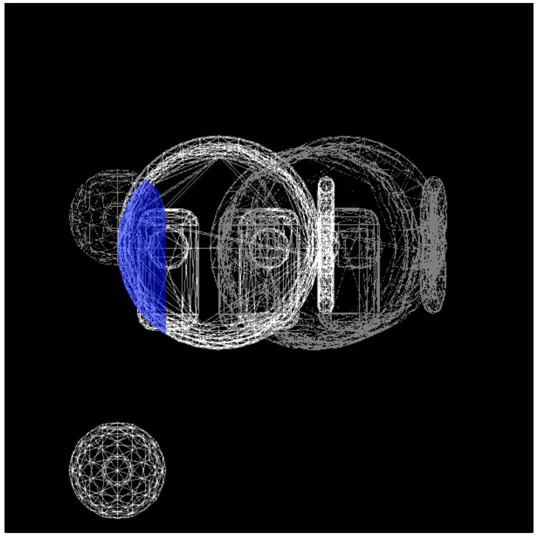
Thank you!

And thanks to Aaron Coday, Tom Piazza, Charles Lingle, and Aaron Lefohn for supporting this research; to Nir Benty and Uzi Sarel for supplying simulation infrastructure; and to Mafred Ernst for the caustics scenes!

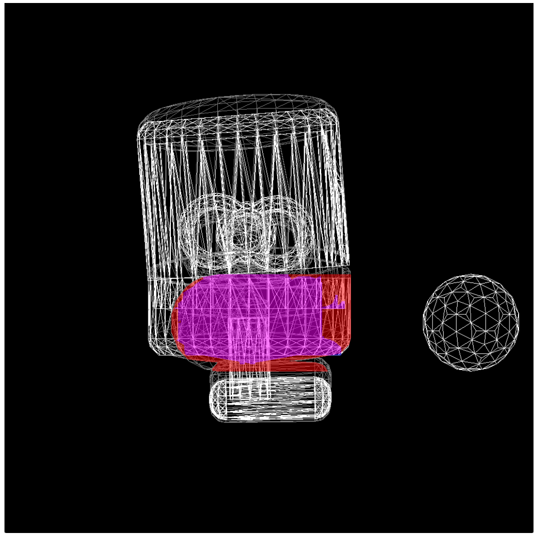
CCD (contd.)



XY



XZ



YZ

- Example: overlap in XY and YZ, but not in XZ