I3D 2006

Symposium on Interactive 3D Graphics and Games

Sponsored by ACM SIGGRAPH

Dynamic Parallax Occlusion Mapping with Approximate Soft Shadows

Natalya Tatarchuk

ATI

3D Application Research Group
ATI Research, Inc.

Outline

- Problem definition
- Related work review
- Parallax occlusion mapping algorithm
- Results discussion
- Conclusions

Objective

- We want to render very detailed surfaces
- Don’t want to pay the price of millions of triangles
  - Vertex transform cost
  - Memory footprint
- Want to render those detailed surfaces accurately
  - Preserve depth at all angles
  - Dynamic lighting
  - Self occlusion resulting in correct shadowing

Parallax Occlusion Mapping

- Per-pixel ray tracing of a height field in tangent space
- Correctly handles complicated viewing phenomena and surface details
  - Displays motion parallax
  - Renders complex geometric surfaces such as displaced text / sharp objects
- Calculates occlusion and filters visibility samples for soft self-shadowing
  - Uses flexible lighting model
- Adaptive LOD system to maximize quality and performance

Parallax Occlusion Mapping versus Normal Mapping

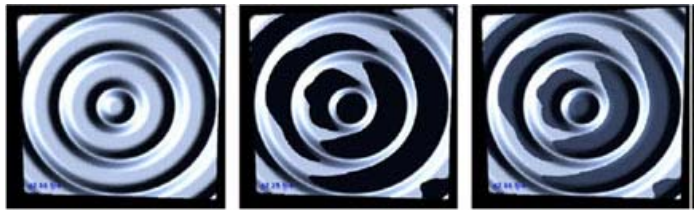
Approximating Surface Details

- First there was bump mapping… [Blinn78]
  - Rendering detailed and uneven surfaces where normals are perturbed in some pre-determined manner
  - Popularized as *normal mapping* – as a *per-pixel* technique
  - No self-shadowing of the surface
  - Coarse silhouettes expose the actual geometry being drawn
- Doesn't take into account geometric surface depth
  - Does not exhibit **parallax**

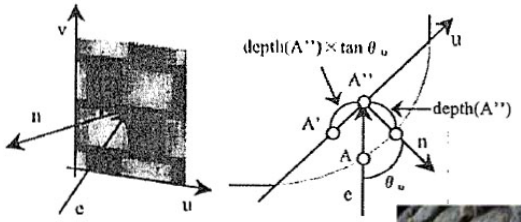
apparent displacement of the object due to viewpoint change

Selected Related Work

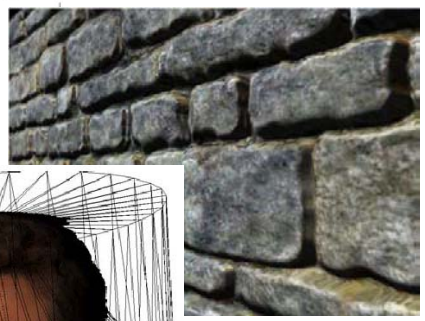
- Horizon mapping [Max88]
- Interactive horizon mapping [Sloan00]



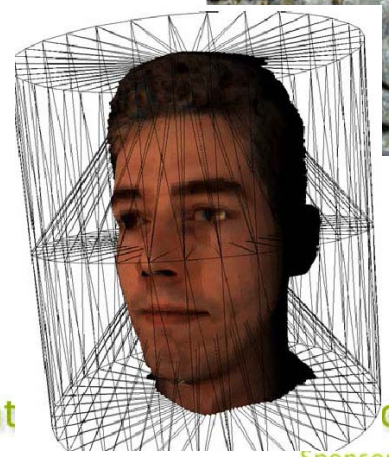
- Parallax mapping [Kaneko01]



- Parallax mapping with offset limiting [Welsh03]



- Hardware Accelerated Per-Pixel Displacement Mapping [Hirche04]



Real-Time Relief Mapping [Policarpo05]

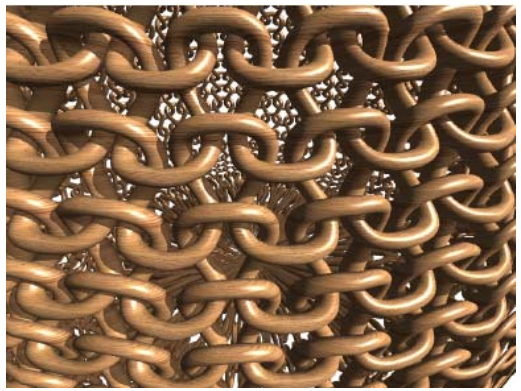
- Similar idea to one presented here
  - Per-pixel ray tracing to arrive at displaced point on the extruded surface
- Different implementation
  - A combination of a static linear search and a binary search to determine an approximation for ray - height field intersection
  - Linear search finds a point below the extruded surface along the ray
  - Binary search is used to arrive at approximate displaced point on the surface
  - Does not compute the ray-surface intersection, just samples the height field
- Hard shadows computed for self-occlusion based shading

Binary Search for Surface-Ray Intersection

- Binary search refers to repeatedly halving the search distance to determine the displaced point
  - The height field is not sorted a priori
  - Requires dependent texture fetches for computation
    - Incurs latency cost for each successive depth level
    - Uses 5 or more levels of dependent texture fetches (therefore only SM 3.0 GPUs), written as SM 2.a

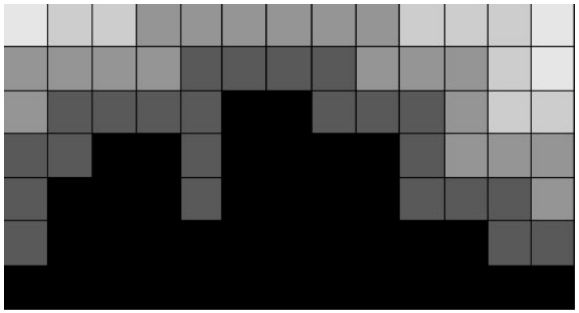
Non-Height-Field Surface Details

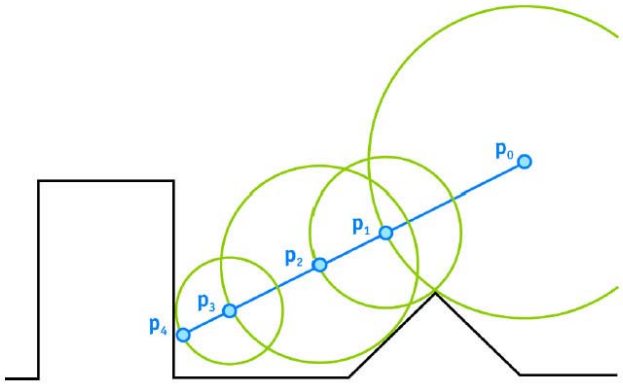
- F. Policarpo, M. M. Oliveira. 2006. *"Relief Mapping of Non-Height-Field Surface Details", I3D 2006*
  - Earlier presentation
- Allows representing non-height-field mesostructure details for rendering complex surfaces



Per-Pixel Displacement Mapping with Distance Functions [Donnely05]

- Also a real-time technique for rendering per-pixel displacement mapped surfaces on the GPU
  - Stores a ‘slab’ of distances to the height field in a volumetric texture
- To arrive at the displaced point, walks the volume texture in the direction of the ray
  - Instead of performing a ray-height field intersection
  - Uses dependent texture fetches, amount varies





Per-Pixel Displacement Mapping with Distance Functions [Donnely05]

- Visible aliasing
  - Not just at grazing angles
- Only supports precomputed height fields
  - Requires preprocessing to compute volumetric distance map
  - Volumetric texture size is prohibitive
- The idea of using a distance map to arrive at the extruded surface is very useful

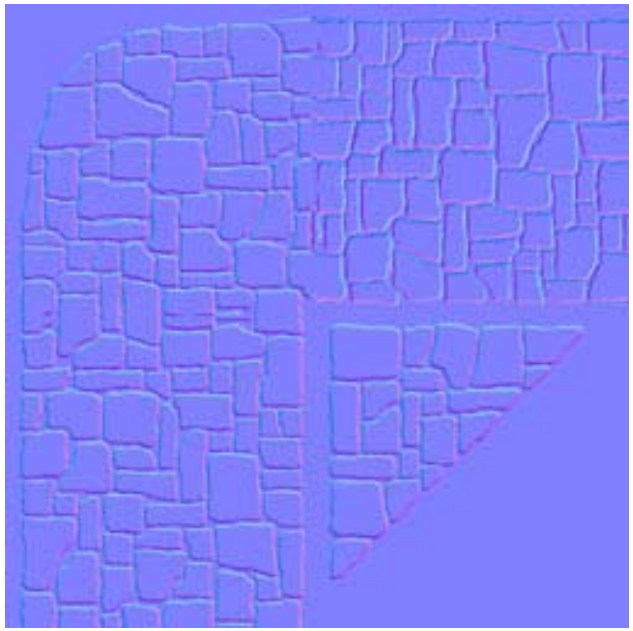
Our Contributions

- Increased precision of height field – ray intersections
- Dynamic real-time lighting of surfaces with soft shadows due to self-occlusion under varying light conditions
- Directable level-of-detail control system with smooth transitions between levels
- Motion parallax simulation with perspective-correct depth

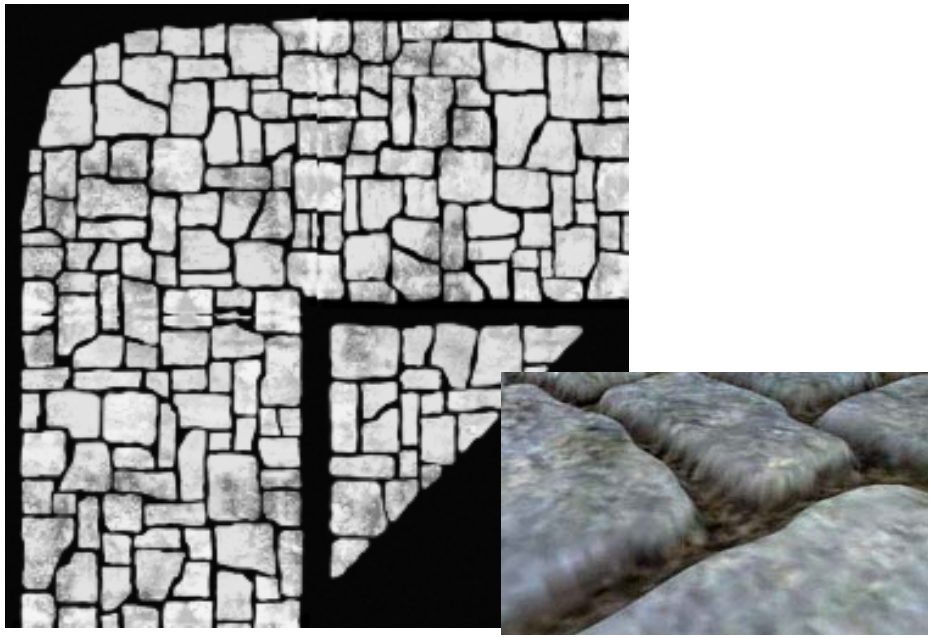
Parallax Occlusion Mapping

- Introduced in [Browley04] “Self-Shadowing, Perspective-Correct Bump Mapping Using Reverse Height Map Tracing”
- Efficiently utilizes programmable GPU pipeline for interactive rendering rates
- Current algorithm has several significant improvements over the earlier technique

Encoding Displacement Information



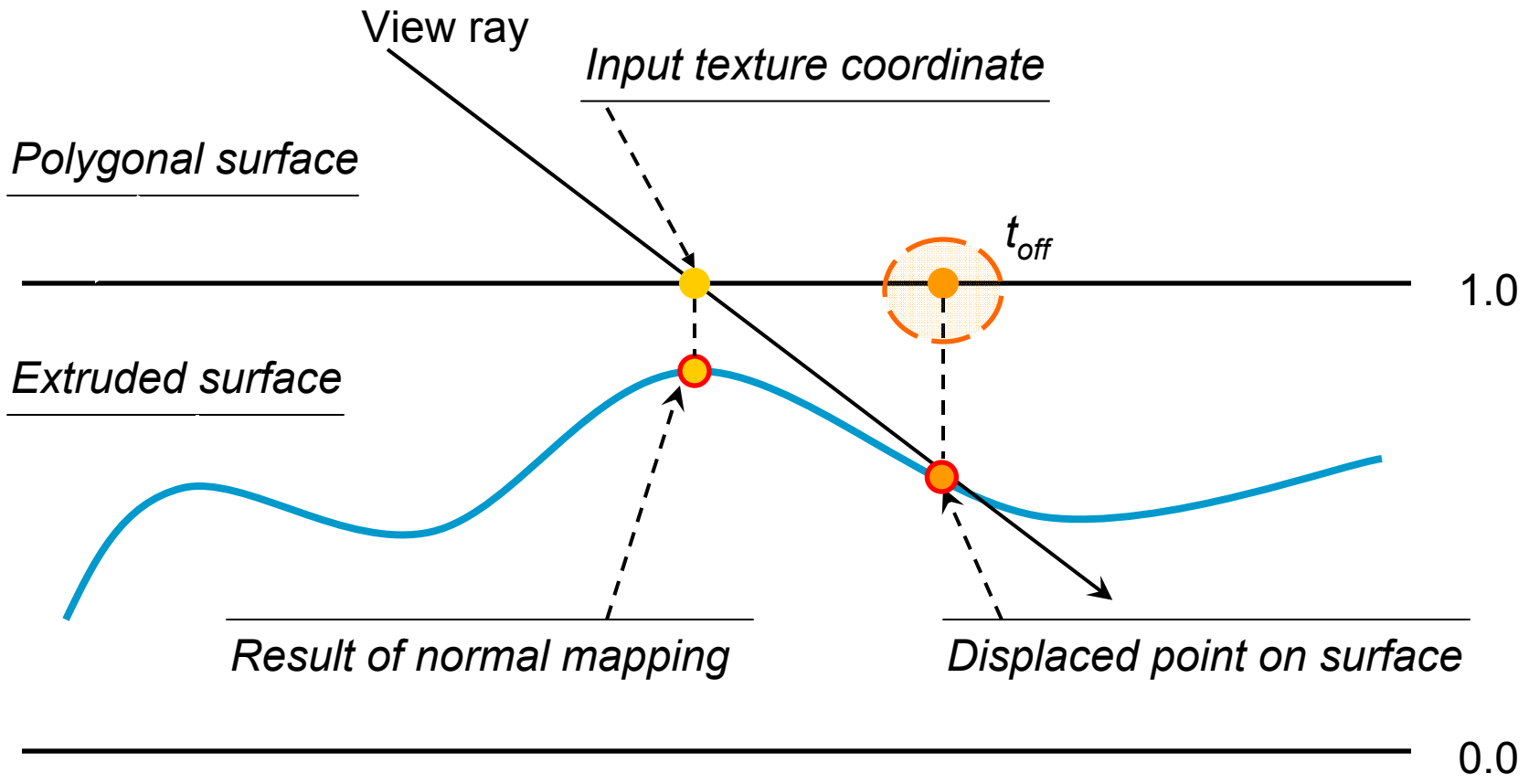
Tangent-space normal map



Height map (displacement values)

All computations are done in tangent space, and thus can be applied to arbitrary surfaces

Parallax Displacement



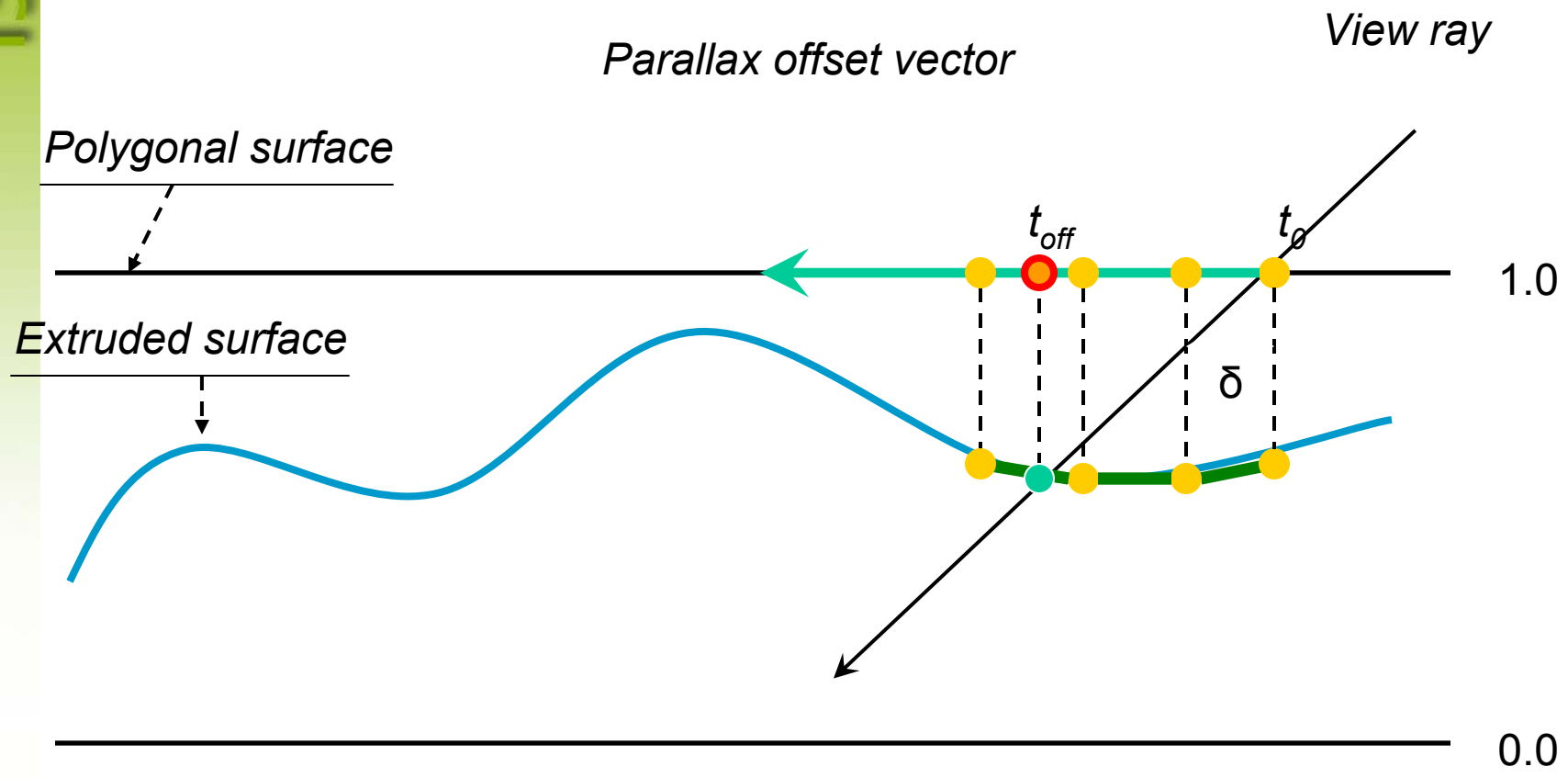
Implementation: Per-Vertex

- Compute the viewing direction, the light direction in tangent space
- Can compute the parallax offset vector (as an optimization)
  - Interpolated by the rasterizer

Implementation: Per-Pixel

- Ray-cast the view ray along the parallax offset vector
- Ray – height field profile intersection as a texture offset
  - Yields the correct displaced point visible from the given view angle
- Light ray – height profile intersection for occlusion computation to determine the visibility coefficient
- Shading
  - Using any attributes
  - Any lighting model

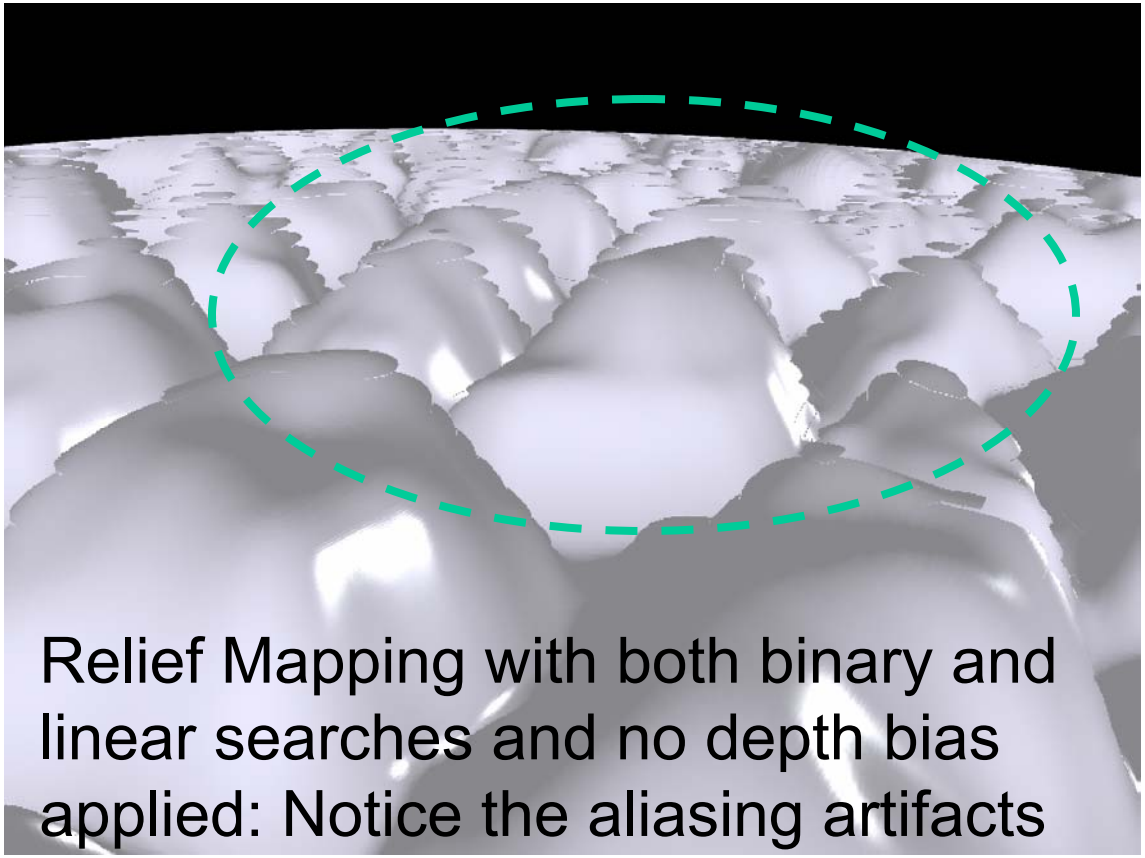
Height Field Profile Tracing



Linear Search for Surface-Ray Intersection

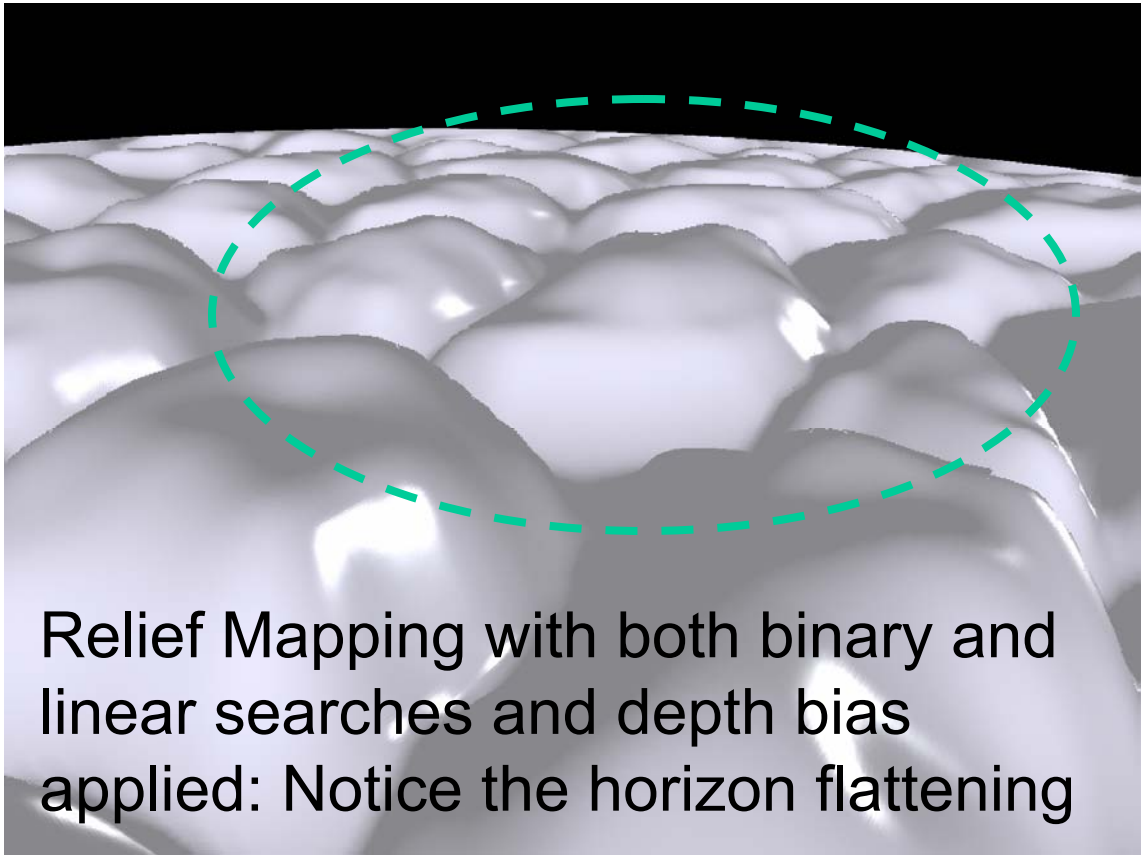
- We use just the linear search which requires only regular texture fetches
  - Fast performance
  - Using dynamic flow control, can break out of execution once the intersection is found
- Linear search alone does not yield good rendering results
  - Requires high precision calculations for surface-ray intersections
  - Otherwise produces visible aliasing artifacts

Comparison of Intersection Search Types and Depth Bias Application



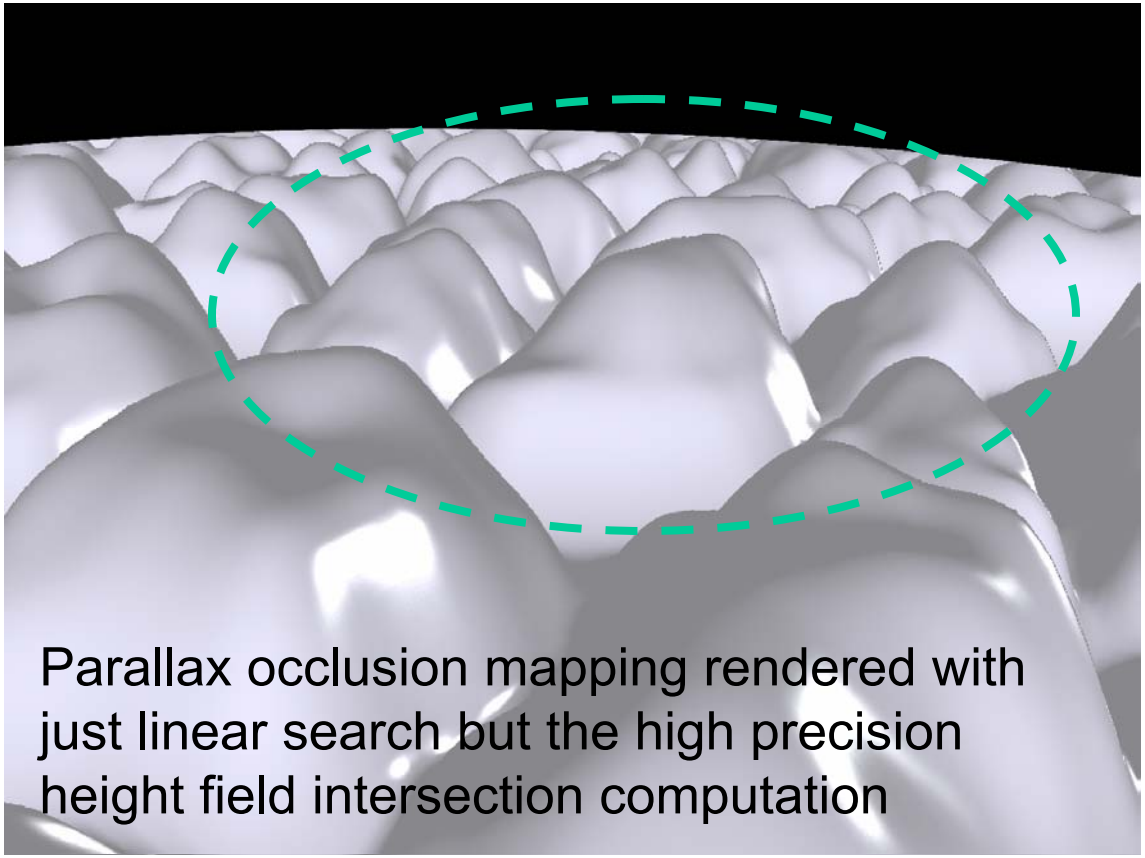
Relief Mapping with both binary and linear searches and no depth bias applied: Notice the aliasing artifacts

Comparison of Intersection Search Types and Depth Bias Application



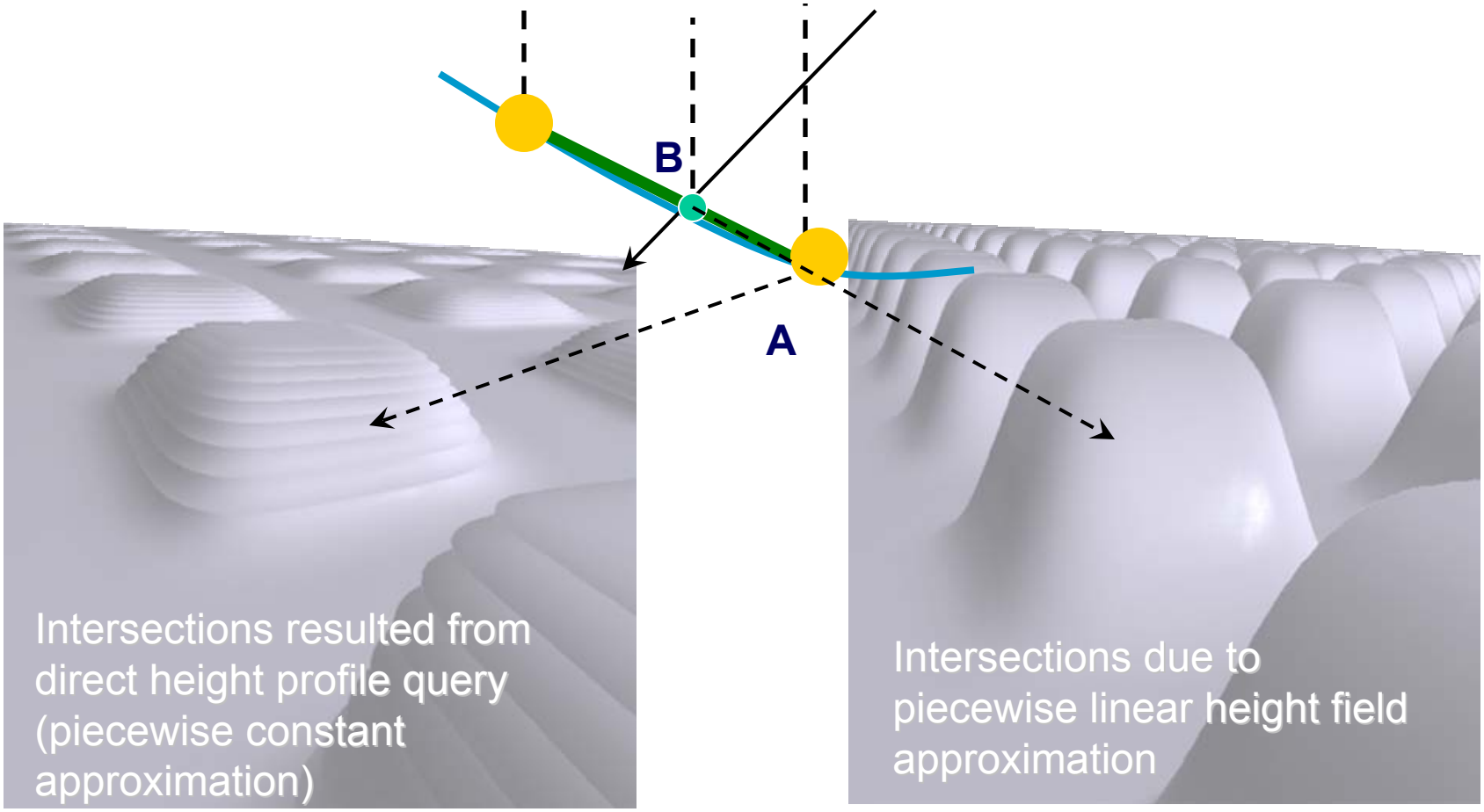
Relief Mapping with both binary and linear searches and depth bias applied: Notice the horizon flattening

Comparison of Intersection Search Types and Depth Bias Application



Parallax occlusion mapping rendered with just linear search but the high precision height field intersection computation

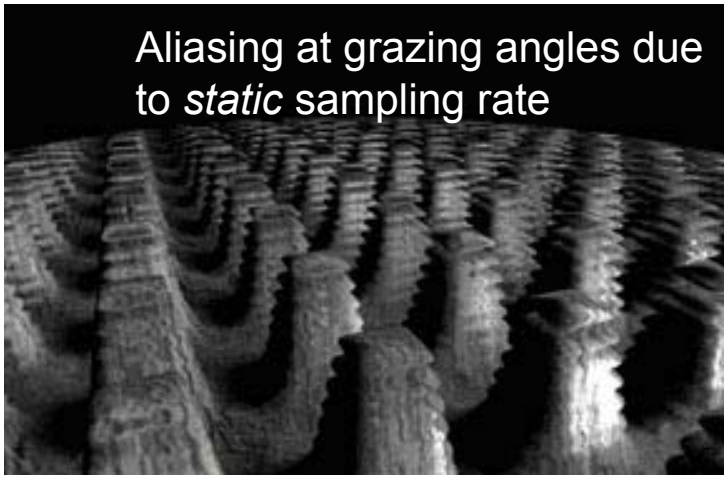
Height Field Profile – Ray Intersection

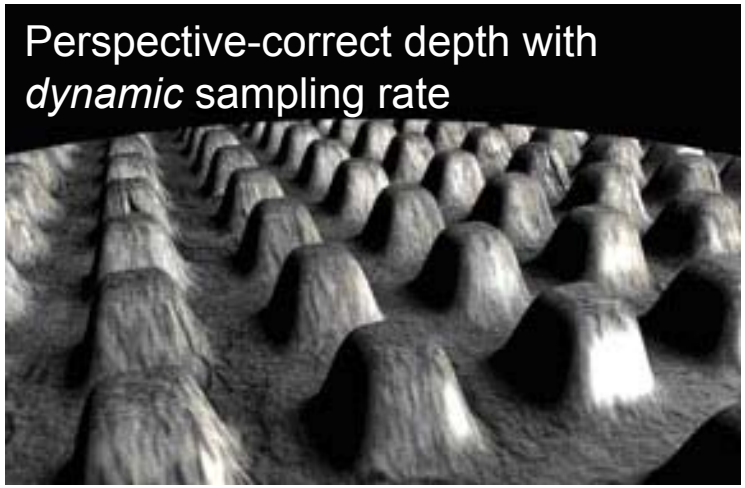


Higher Quality With Dynamic Sampling Rate

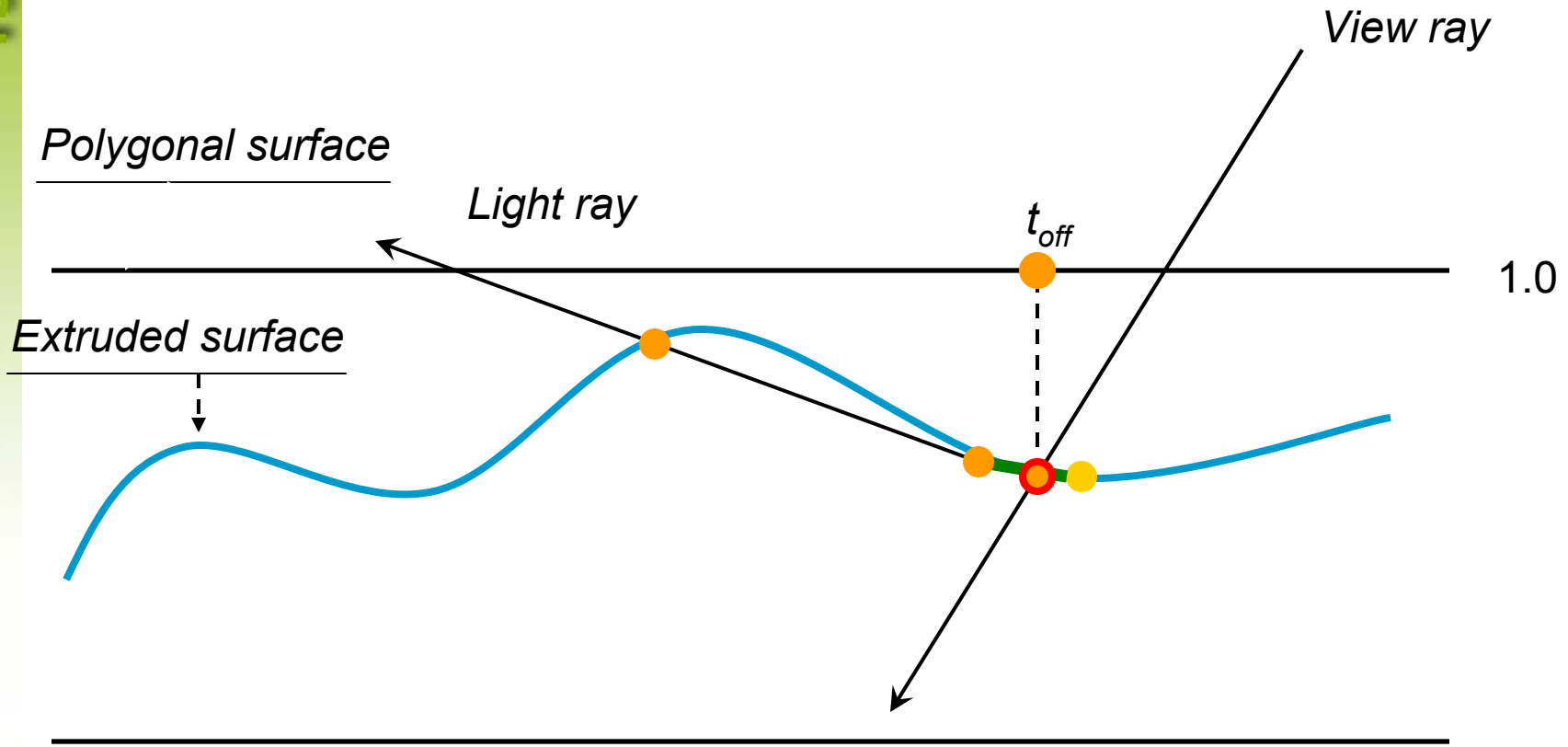
- Sampling-based algorithms are prone to aliasing
- Solution: *Dynamically* adjust the sampling rate for ray tracing as a linear function of angle between the geometric normal and the view direction ray

$$n = n_{\min} + \hat{N} \bullet \hat{V}_{ts}(n_{\max} - n_{\min})$$



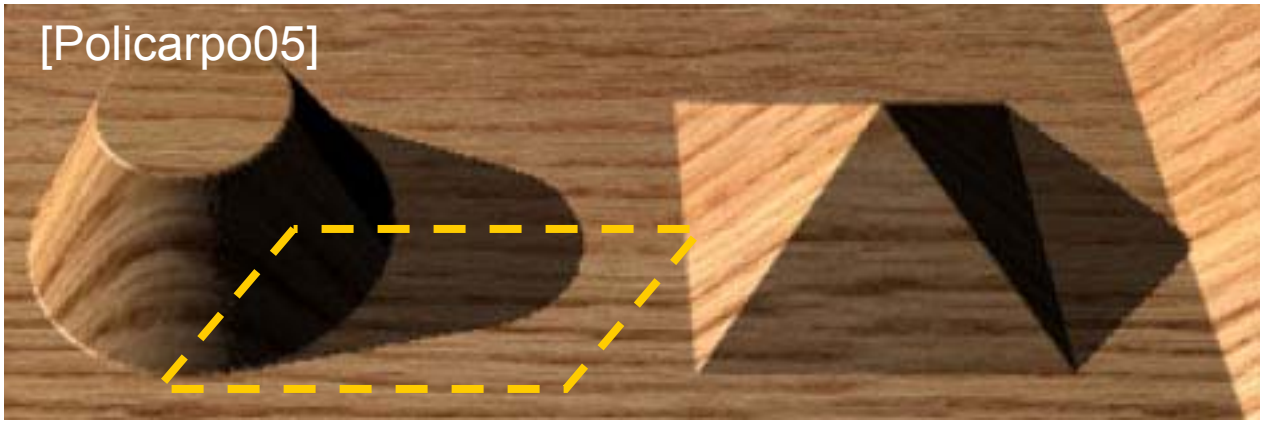


Self-Occlusion Shadows



Hard Shadows Computation

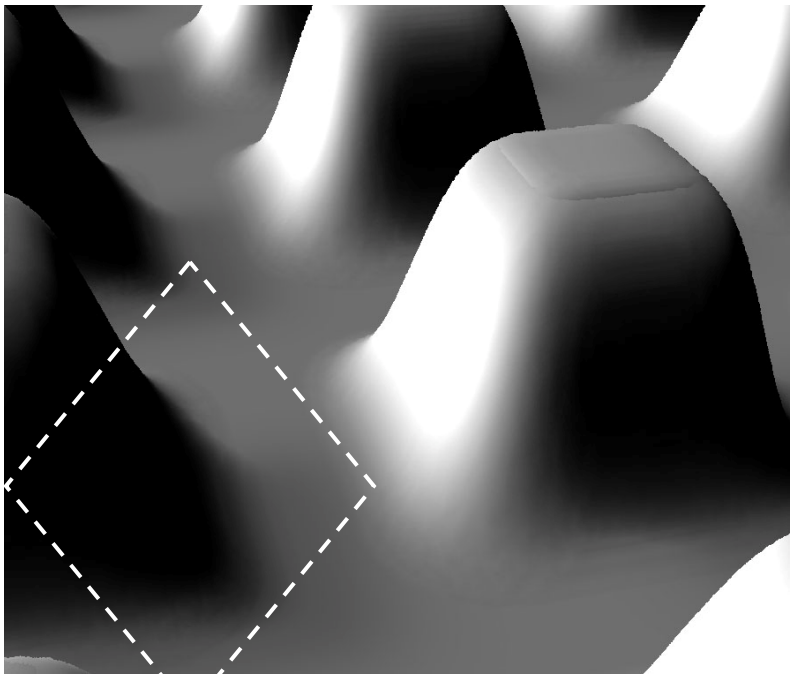
- Simply determining whether the current feature is occluded yields hard shadows



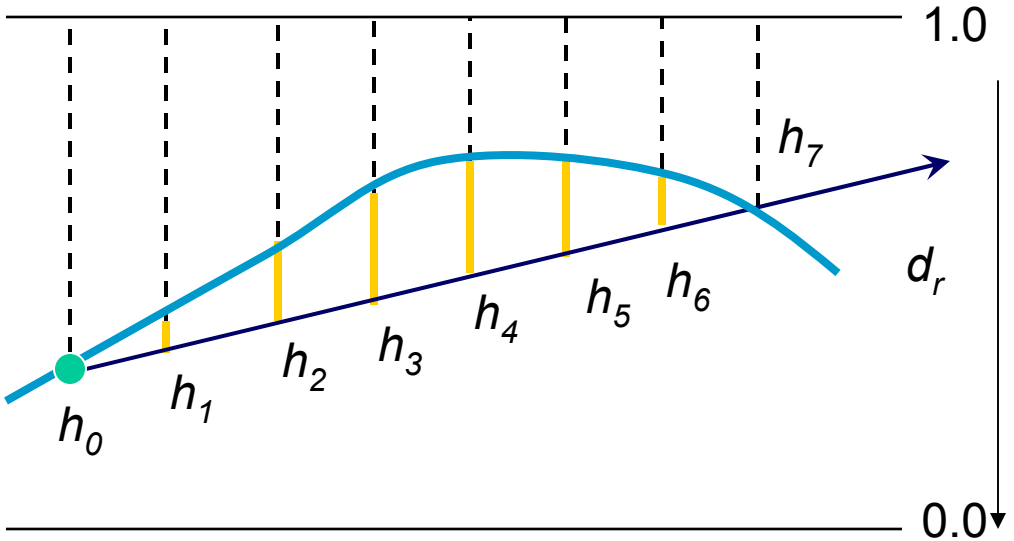
Soft Shadows Computation

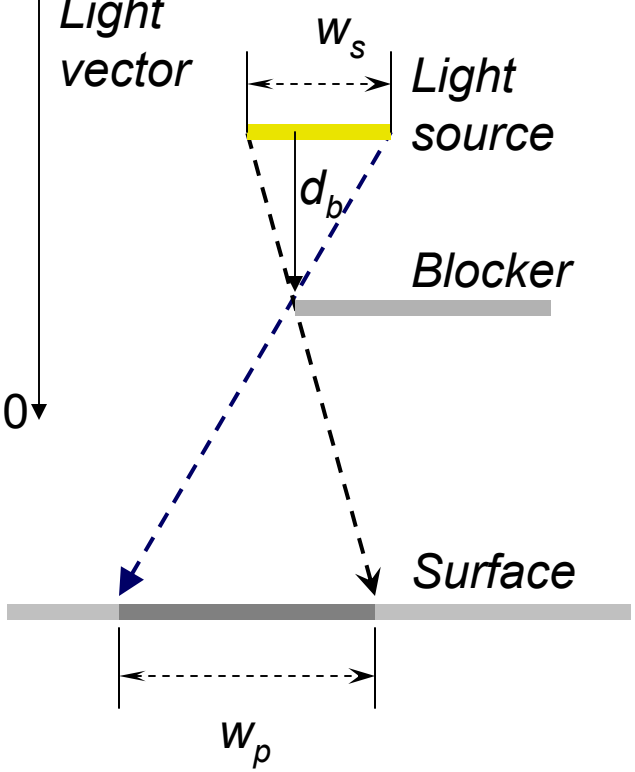
- We can compute soft shadows by filtering the visibility samples during the occlusion computation
- Don't compute shadows for objects not facing the light source:

$$N \bullet L > 0$$



Penumbral Size Approximation

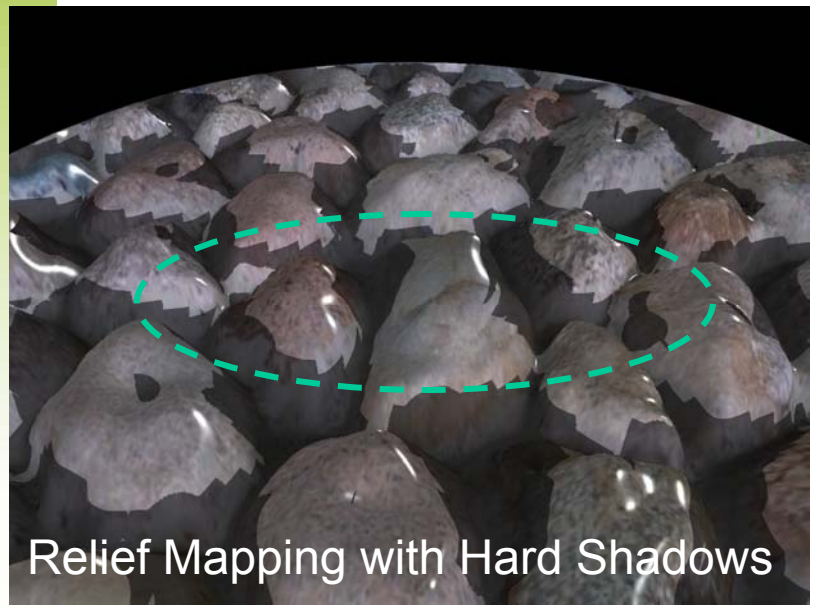


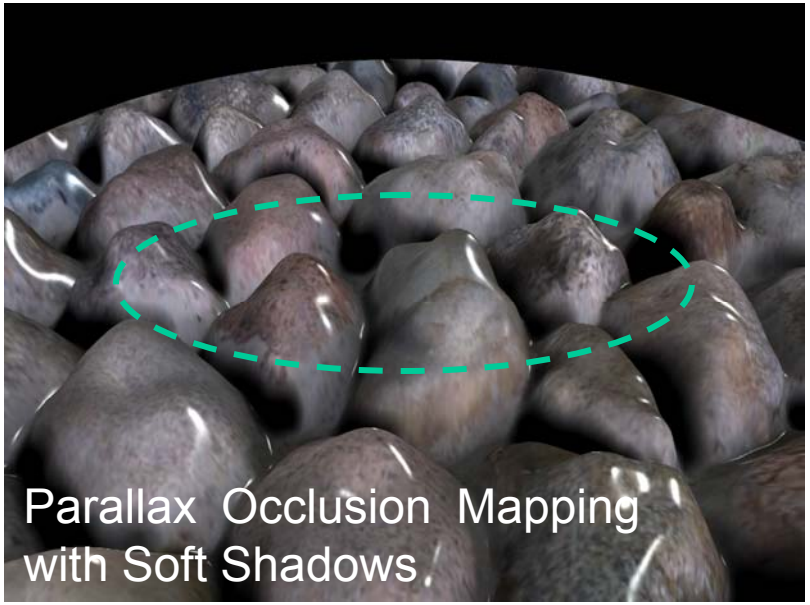


The blocker heights h_i allow us to compute the *blocker-to-receiver* ratio

$$w_p = w_s \, (d_r - d_b) \, / \, d_b$$

Shadows Comparison Example





Illuminating the Surface

- Use the computed texture coordinate offset to sample desired maps (albedo, normal, detail, etc.)
- Given those parameters and the visibility information, we can apply any lighting model as desired
  - Phong
  - Compute reflection / refraction
  - Very flexible

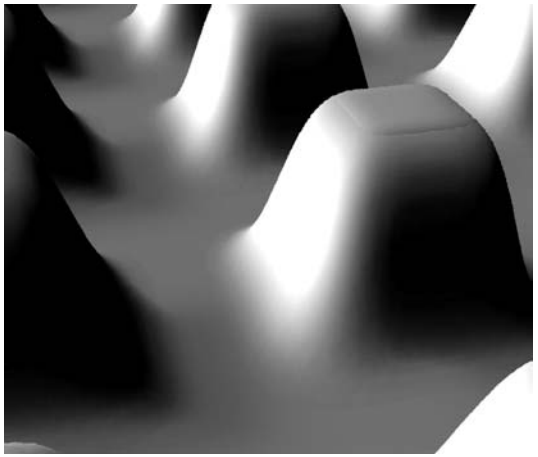
Adaptive Level-of-Detail System

- Compute the current mip map level
- For furthest LOD levels, render using normal mapping (threshold level)
- As the surface approaches the viewer, increase the sampling rate as a function of the current mip map level
- In transition region between the threshold LOD level, blend between the normal mapping and the full parallax occlusion mapping

Results

- Implemented using DirectX 9.0c shaders (separate implementations in SM 2.0, 2.b and 3.0)

RGBα texture: 1024 x 1024, non-contiguous *uv*s



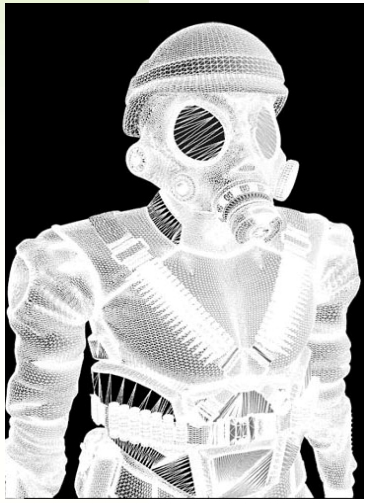
RGBα texture: tiled 128 x 128

Parallax Occlusion Mapping vs. Actual Geometry

-1100 polygons with parallax occlusion mapping (8 to 50 samples used)
- **Memory**: 79K vertex buffer
  6K index buffer
  13Mb texture (3Dc)
  (2048 x 2048 maps)

Total: **< 14 Mb**

Frame Rate:
- ***255 fps*** on ATI Radeon hardware
- ***235 fps*** with skinning



- 1,500,000 polygons with normal mapping
- **Memory**: 31Mb vertex buffer
  14Mb index buffer

Total: **45 Mb**

Frame Rate:
- ***32 fps*** on ATI Radeon hardware


Demo:
ToyShop

Incorporate Dynamic Height Field Rendering with POM

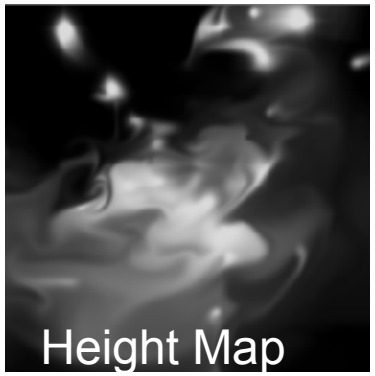
- Easily supports *dynamically rendered* height fields
  - Generate height field
  - Compute normals for this height field
  - Apply inverse displacement mapping w/ POM algorithm to that height field
  - Shade using computed normals
- Examples of dynamic HF generation:
  - Water waves / procedurally generated objects / noise
  - Explosions in objects
  - Bullet holes
- Approaches that rely on precomputation do not support dynamic height field rendering in real-time
  - Displacement mapping with distance maps
  - Encoding additional vertex data such as curvature

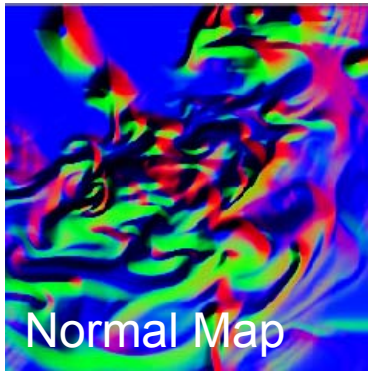
Combine Fluid Dynamics with POM

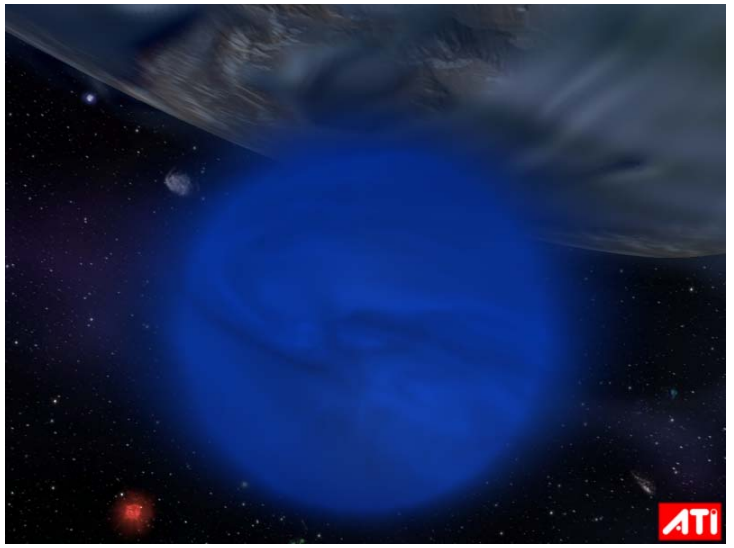
- Compute Navier-Stokes simulation for fluid dynamics for a height field
  - Example: Fluid flow in mysterious galaxies from "Screen Space" ATI X1900 screen saver
- Fluid dynamics algorithm can be executed entirely on the GPU
  - See ATI technical report on "Explicit Early-Z Culling for Efficient Fluid Flow Simulation and Rendering" by P. Sander, N. Tatarchuk and J.L. Mitchell for details

Example: Gas Planet Scene

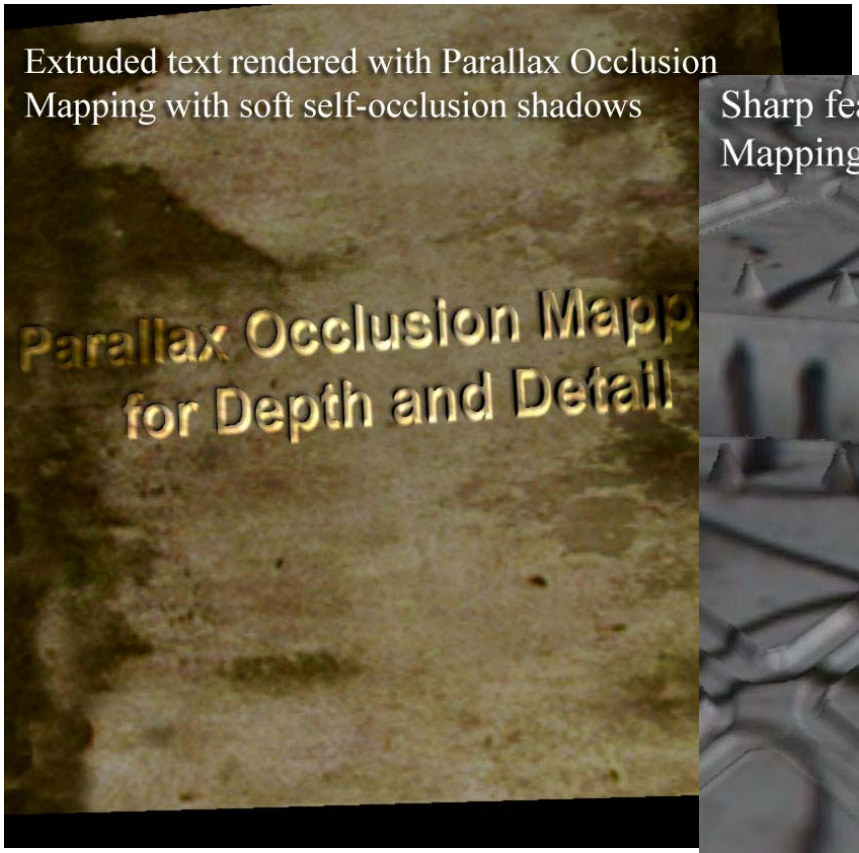
- Random particles in texture space emit flow density and velocity
- Flow used to compute height field for parallax occlusion mapping
- Compute dynamic normals for the flow height field
- Parallax occlusion mapping used to simulate cloud layer on large planet

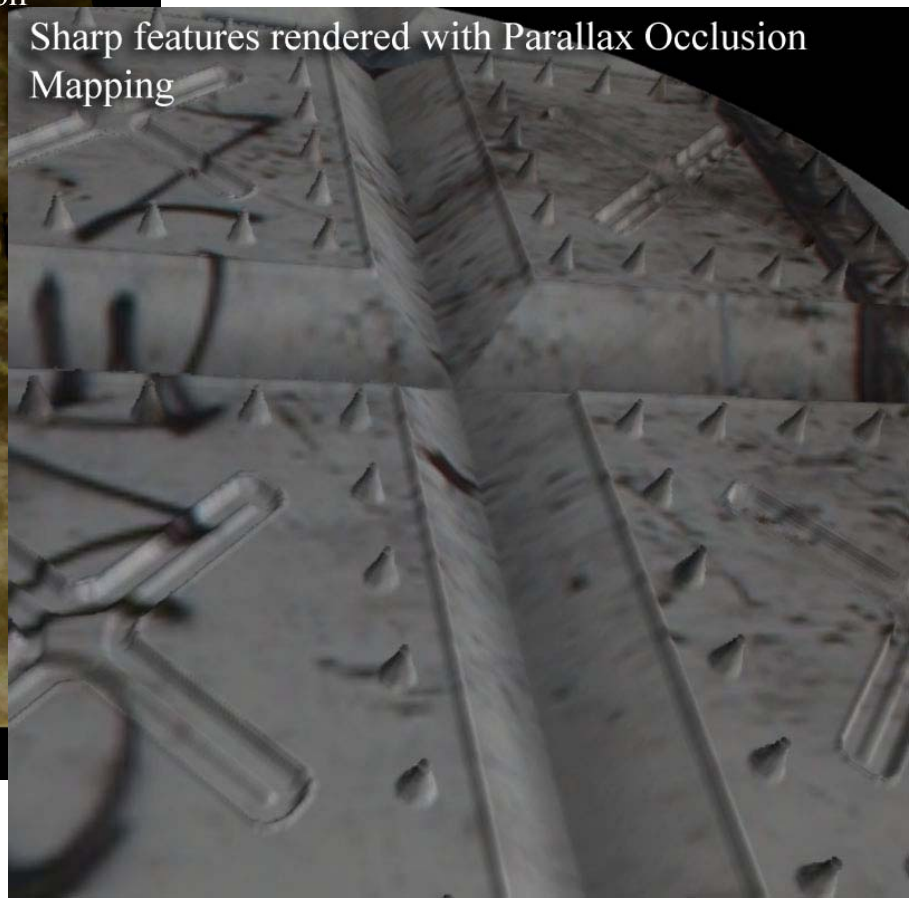




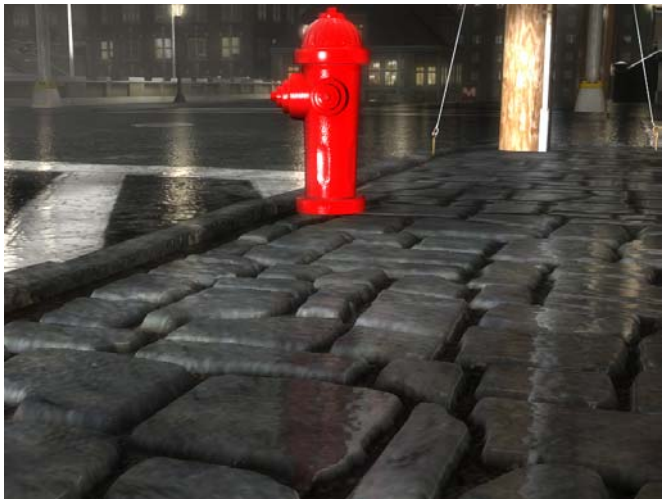


Able to Handle Difficult Cases





Conclusions



- Powerful technique for rendering complex surface details in real time
  - Higher precision height field – ray intersection computation
  - Self-shadowing for self-occlusion in real-time
  - LOD rendering technique for textured scenes
- Produces excellent lighting results
- Has modest texture memory footprint
  - Comparable to normal mapping
- Efficiently uses existing pixel pipelines for highly interactive rendering
- Supports dynamic rendering of height fields and animated objects

The ToyShop Team

Lead Artist

Dan Roeger

Lead Programmer

Natalya Tatarchuk

David Gosselin

Artists

Daniel Szecket, Eli Turner, and Abe Wiley

Engine / Shader Programming

John Isidoro, Dan Ginsburg, Thorsten Scheuermann and Chris Oat

Producer

Lisa Close

Manager

Callan McInally

Reference Material

- www.ati.com/developer
  - Demos, GDC presentations, papers and technical reports, and related materials
- N. Tatarchuk. 2006. “Dynamic Parallax Occlusion Mapping with Approximate Soft Shadows”, *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*
- P. Sander, N. Tatarchuk, J. L. Mitchell. 2004. “Explicit Early-Z Culling for Efficient Flow Simulation and Rendering”, *ATI Research Technical Report,* August 2004.
- ATI ToyShop demo:
  http://www.ati.com/developer/demos/rx1800.html
  ATI ScreenSpace screen saver:
  http://www.ati.com/designpartners/media/screensavers/RadeonX1k.html