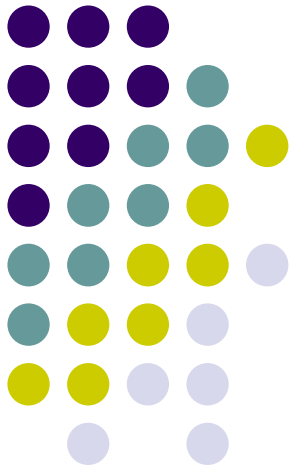
Advanced Computer Graphics
CS 563: Screen Space GI Techniques: Real-Time

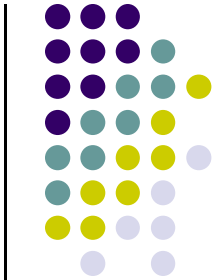
William DiSanto

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

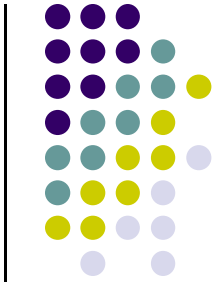


Overview



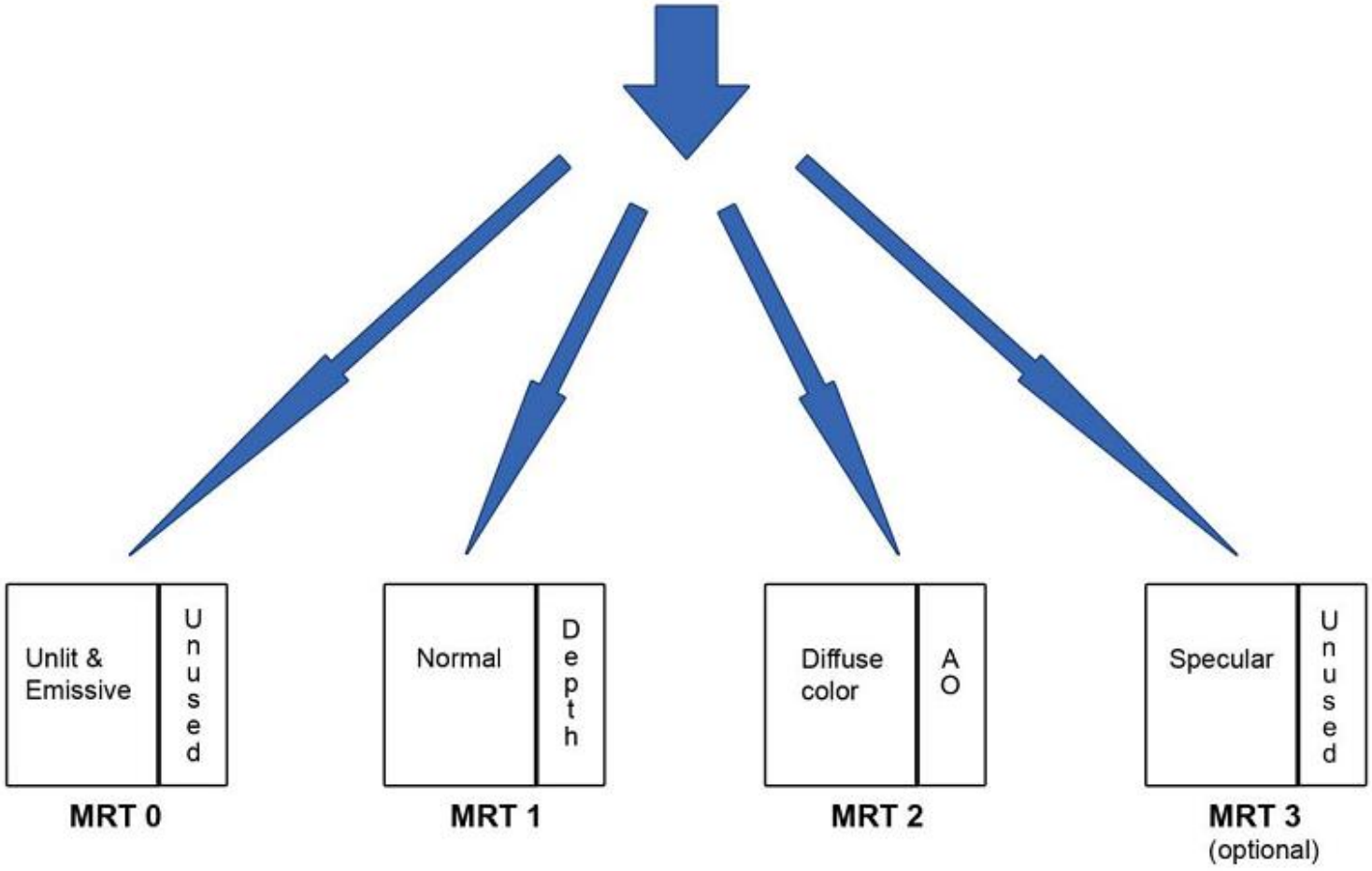
- Deferred Shading
- Ambient Occlusion
- Screen Space Ambient Occlusion
- Horizon Occlusion
- Directional Occlusion
- Single Bounce Indirect Lighting
- Reflective Shadow Maps
  - Gathering / Shooting
- Multi-resolution Splatting

Deferred Shading

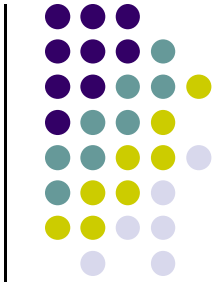


- Provides a framework for many screen space techniques
- Geometry is rendered first
- Shading is completed in another pass
- Render the depth (position relative to camera), normal, and material onto a set of textures
- Results in higher memory usage and bandwidth
- More sampling required in later stages
- Reduces cost of operations by implementing effects dependent only on screen size not scene complexity

Deferred Shading: Multiple Targets

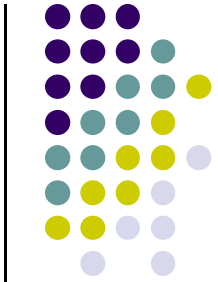


Screen Space Techniques



- Generally less dependent on scene complexity.
- Obtaining and manipulating data in a way that is GPU friendly
- Requires knowledge of the hardware capabilities of the typical graphics card
- Offer set of parameters:
  - Physically accuracy
  - Artistic control

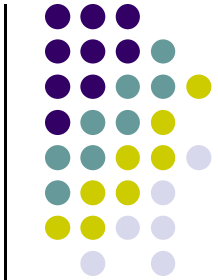
Ambient Occlusion



$$A = 1 - \frac{1}{2\pi}\int_{\Omega} V(\vec{\omega})W(\vec{\omega})d\omega$$

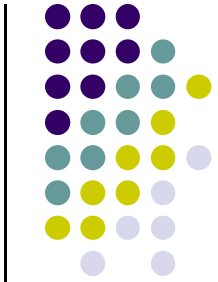
- V function: 0 for visible, 1 for blockage
- W function: attenuation based on some condition
- Ray Cast with some randomization to compute
- Generally low frequency result
  - This opens the door to some approximate representations
  - Can use maps but this restricts animations, dynamic lighting

Screen Space Ambient Occlusion



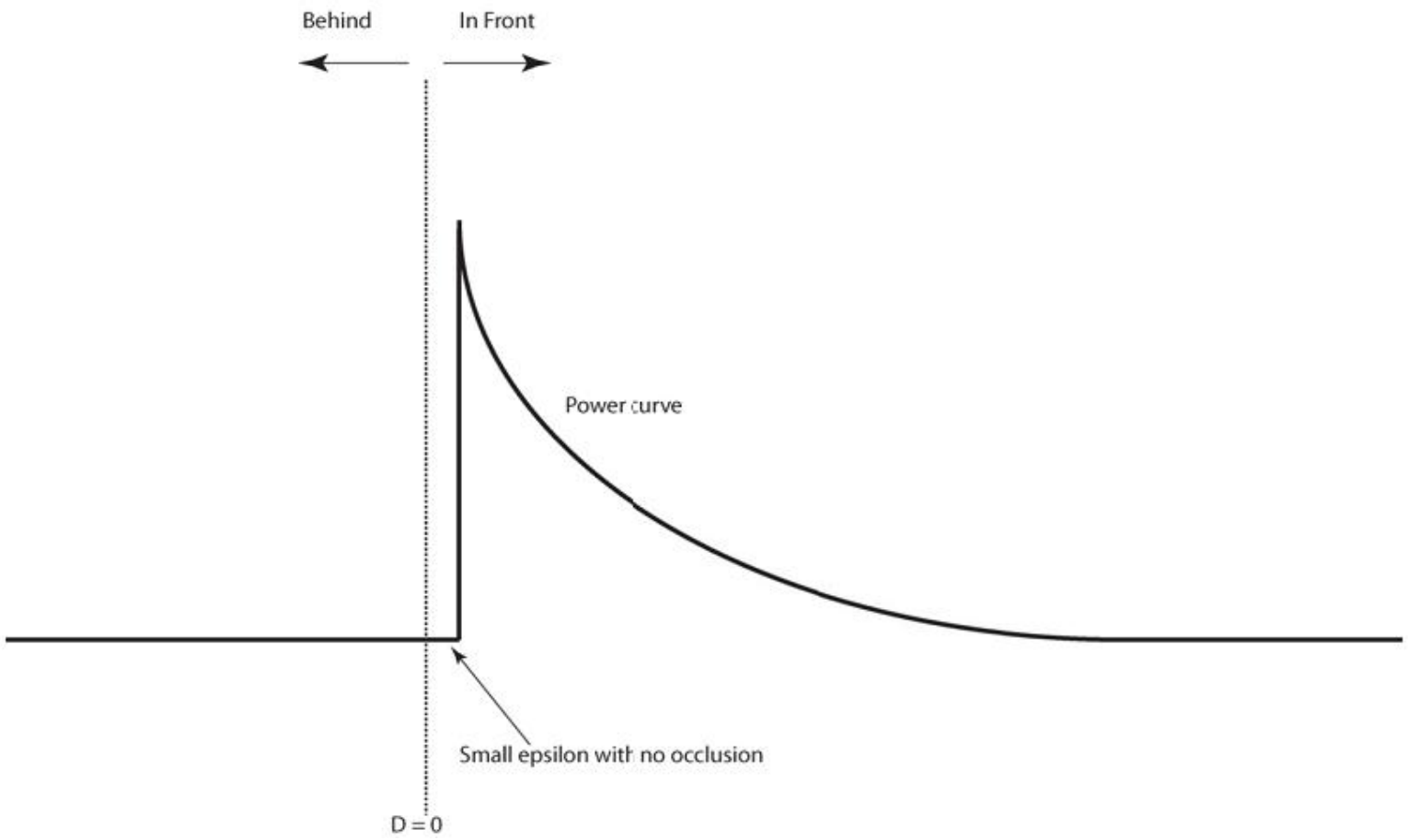
- Z-buffer data might already be available in a texture
- Estimate of ambient occlusion taken from neighboring pixels
  - Term is used to attenuate incoming light
- Looks best when applied to the ambient term only
- Is not very accurate but provides a convincing effect

SSAO: Calculation

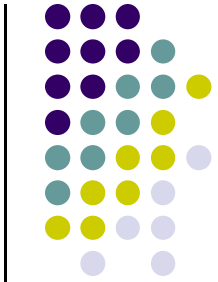


- Use random samples inside a sphere centered a surface point
  - Prevents banding
- Occlusion function used to relate sample depth delta and distance from the central point
  - Negative depth deltas give zero occlusion
  - Small positive depth delta produces high occlusion term
  - Large positive depth delta tend to zero
    - Because this calculation happens in screen space
  - Simple exponentials or look-ups used

SSAO: Calculation



SSAO: Randomization

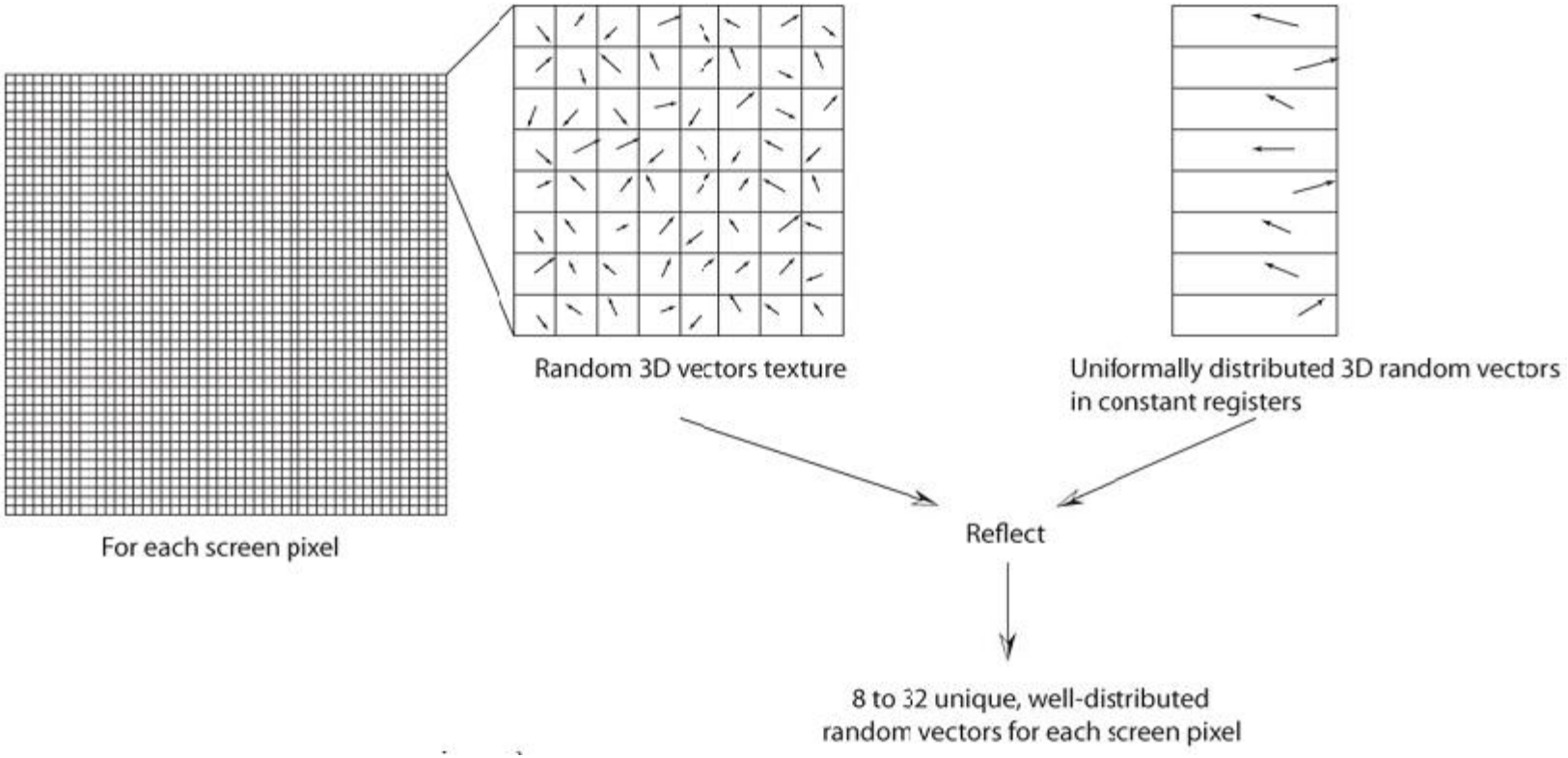


- Generate some number of random normal vectors per pixel
  - (8-32) Starcraft II, 16 Cryengine II
- Reflect random vectors off another set of varying length vectors with uniform distribution in solid sphere
  - Range of length is scaled by some artistic parameter
- Samples then passed through the occlusion function

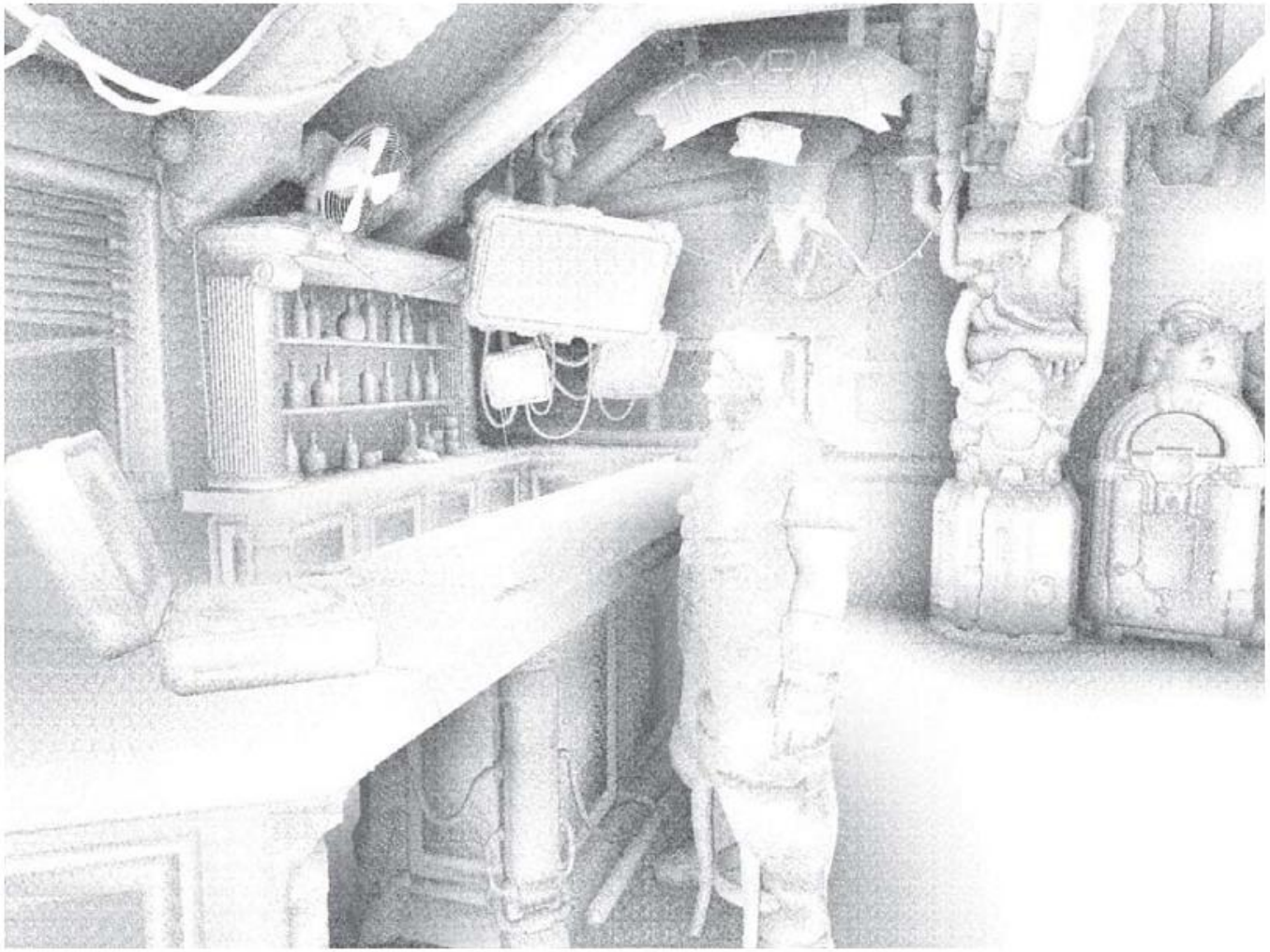
n: the normalized random per pixel vector from the texture
i: one of the 3D sample positions in a sphere

```
float3 reflect( float3 i,  float3 n ) { return i - 2 * dot(i, n) * n;
```

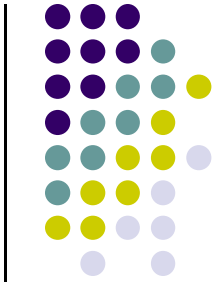
SSAO: Randomization



SSAO: Noise

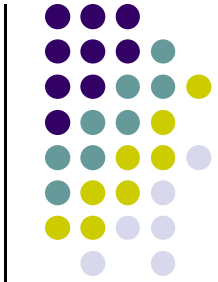


SSAO: Noise Reduction

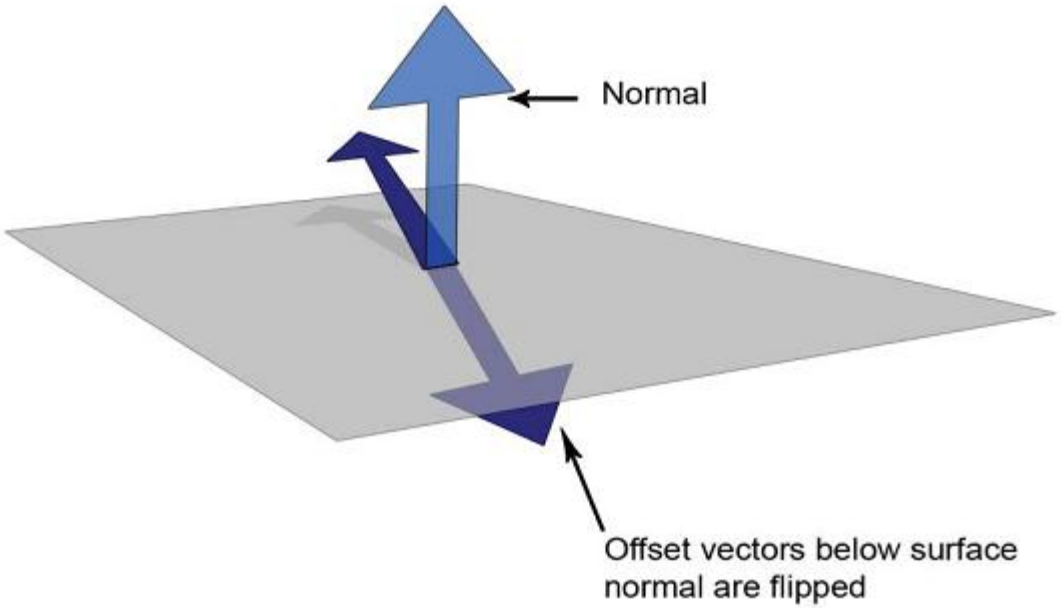


- Noise is then reduced with smart Gaussian blur
- Consider the depth and normal buffer information determines if the blur is reasonable for a sample
  - If the difference between normals or depths from point at Gaussian center to the sample is too great:
    - Sample is tossed
    - Result of operation is renormalized
- Several passes may be required to eliminate grain

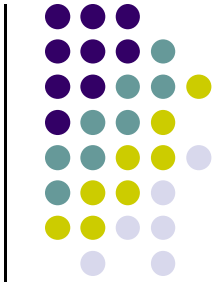
SSAO: Self Occlusion



- Sample may occlude itself, recalculate all vector to top hemisphere of sample point normal
  - Otherwise every object will always be partially occluded
- More accurate and expensive techniques exist

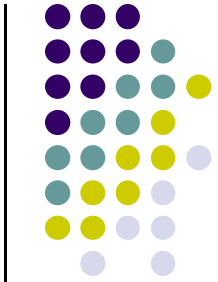


SSAO: Edge Case



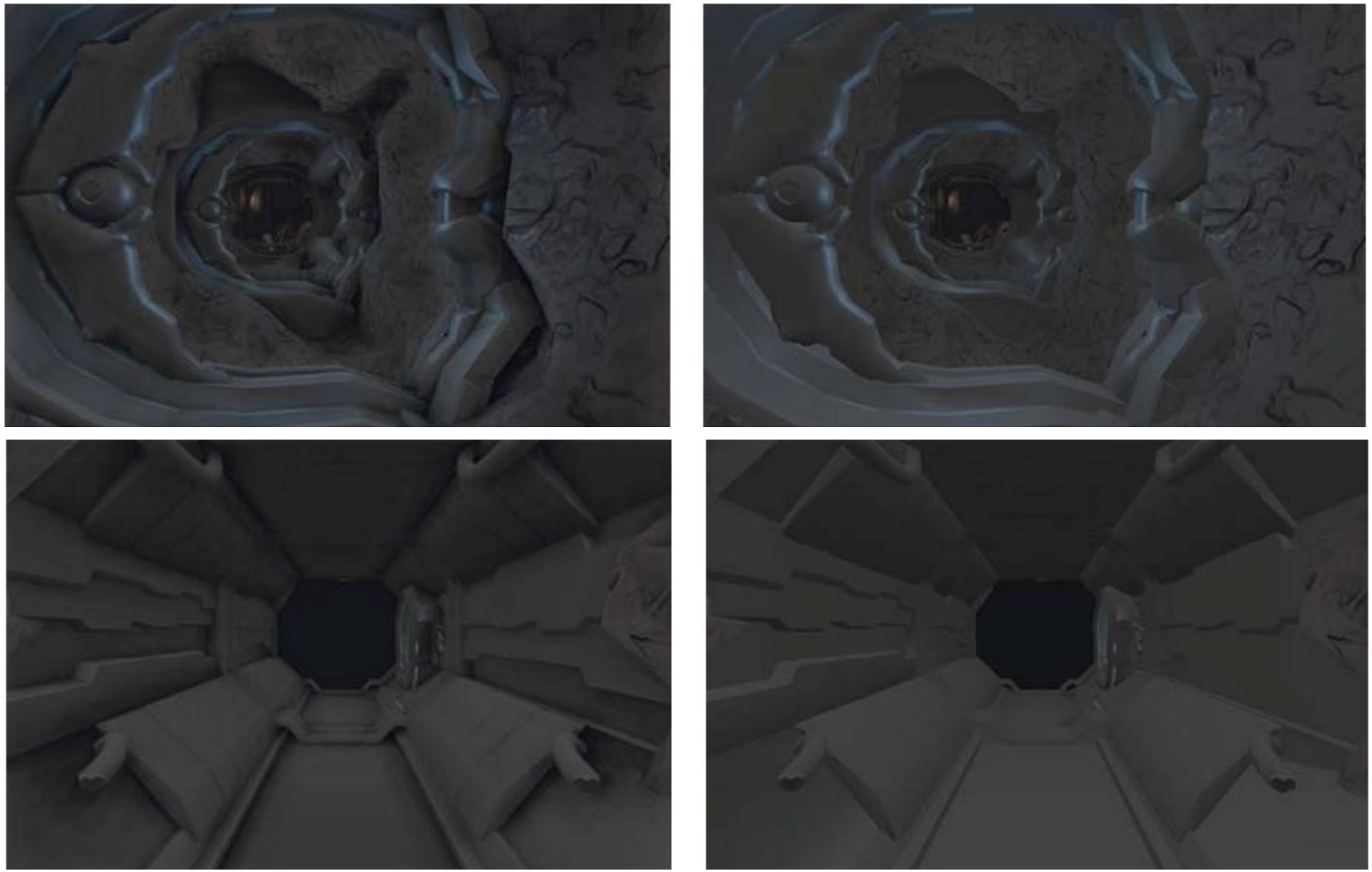
- SSAO has no sample outside of the render target
  - Throw a boundary color around the texture
- When the camera moves close to an object noise will become more noticeable
  - Increasing number of samples based on view proximity would bring the performance of the algorithm closer to the world space AO calculation
  - Constrain the area in which samples are taken

SSAO: Performance

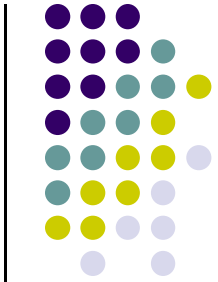


- Stable performance from any camera view
- Bottleneck at the random sampling
- Tends to over illuminate solid object edges
- Found low screen resolution depth buffer sufficient
  - ¼ size of original depth render
- Multiple SSAO functions can be used together (along with different sampling constraints) to model different AO effects.
  - Greatest occlusion of all occlusion averages taken

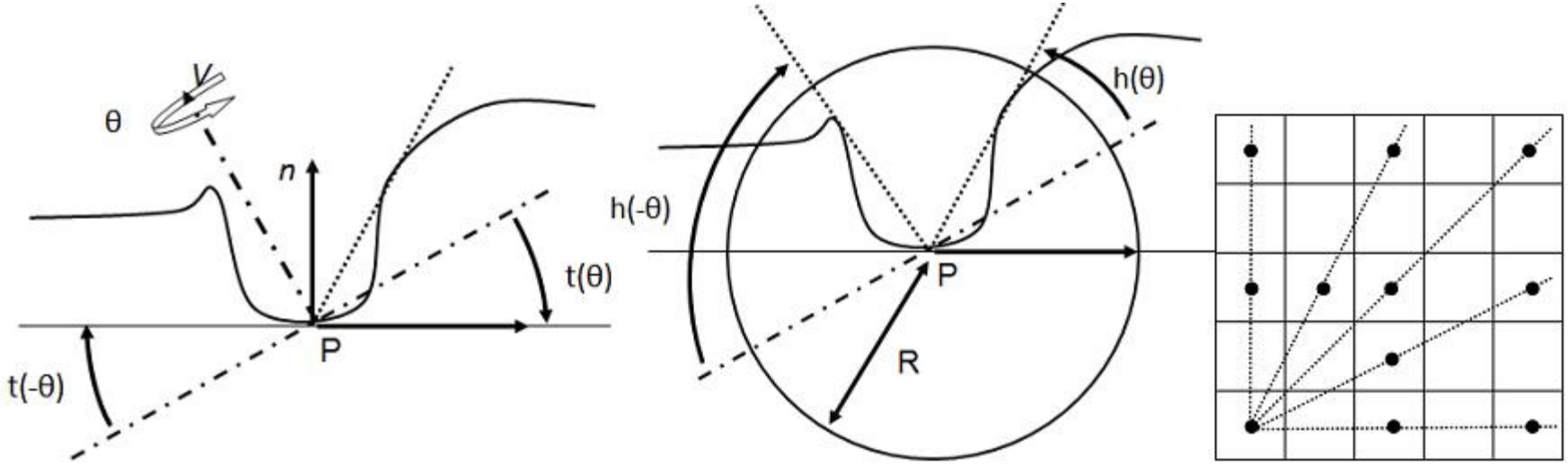
SSAO: Results



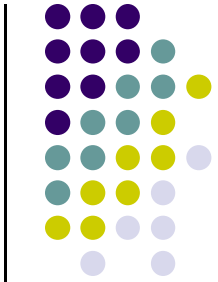
SSAO: Horizon Based Results



- For some radius around sample point:
  - Step through depth buffer in some number of randomized directions for a fixed number of samples
  - Find highest altitudes from center within radius (horizon)
  - Average the weighted samples over the directions
  - Intergrade radiance over through visible angle



SSAO: Horizon Based Results

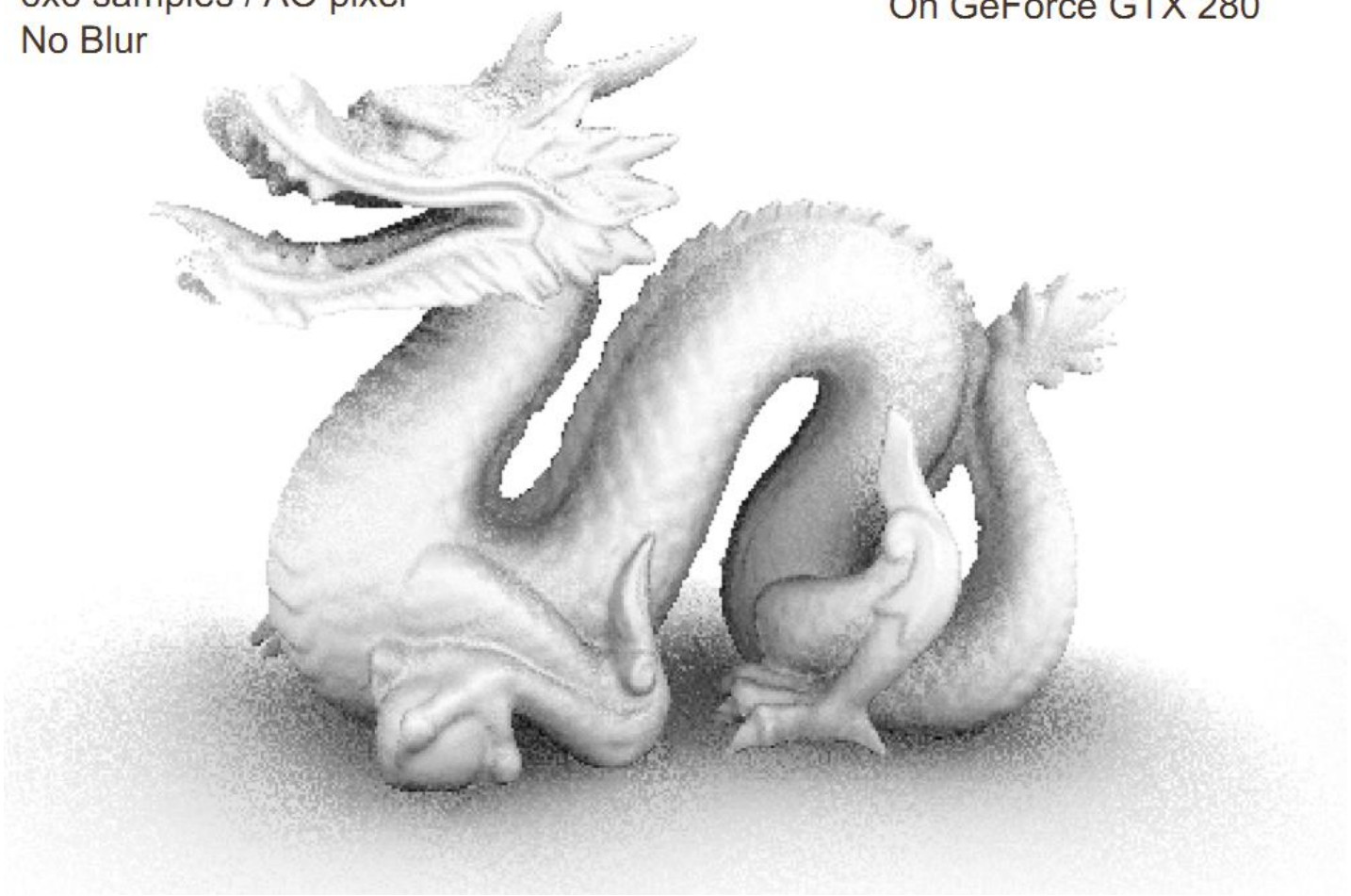


- Interactive frame rates (2008)
- Relies on blurring techniques
- Might require some biasing in horizon angle
- Per sample falloff (radial falloff function)
- Jitter samples
- More accurately simulates ray casting AO

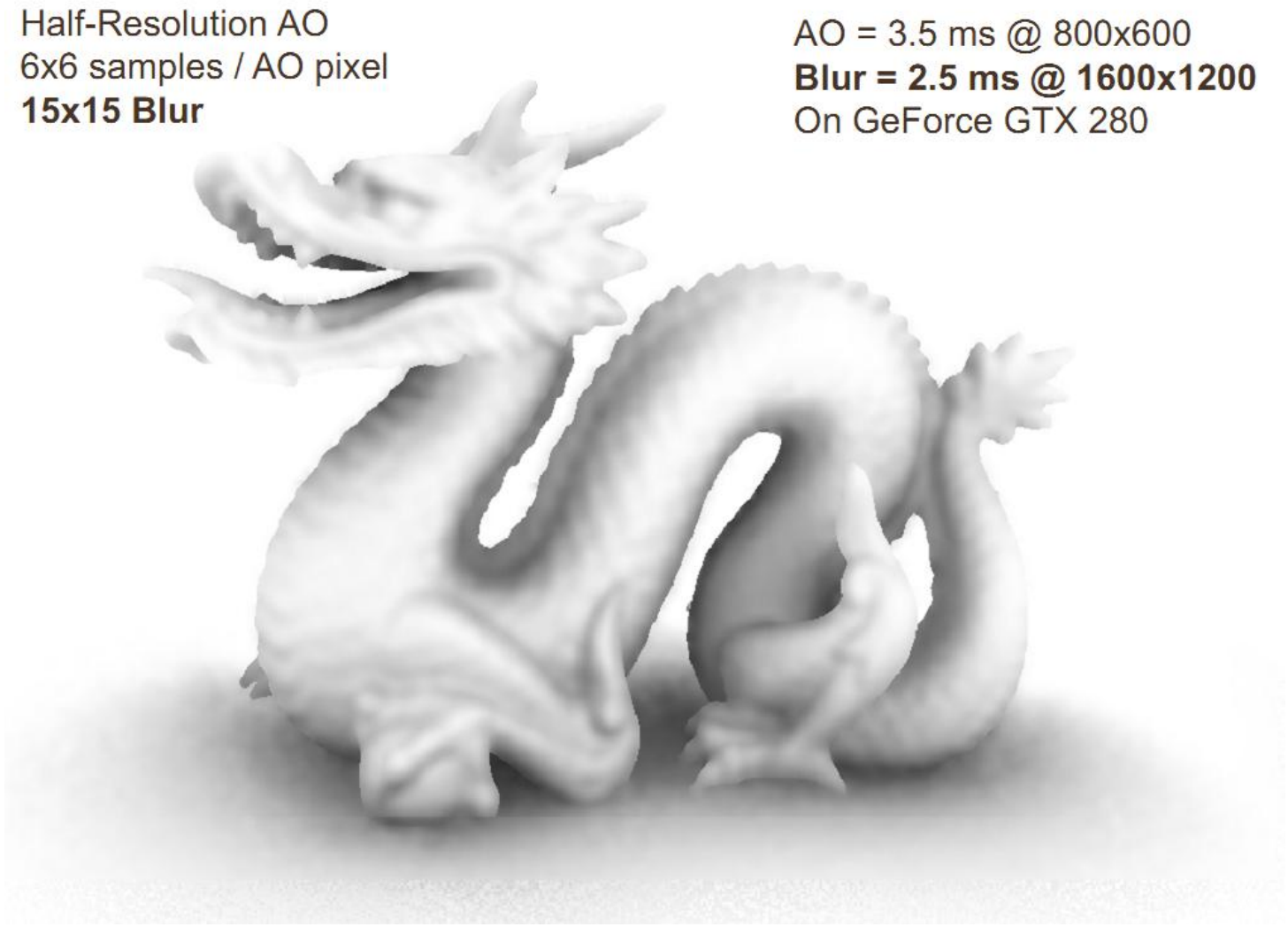
SSAO: Horizon Based Results

Half-Resolution AO
6x6 samples / AO pixel
No Blur

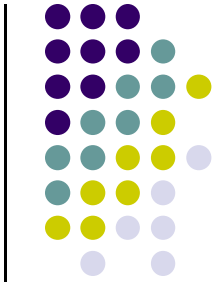
AO = 3.5 ms @ 800x600
On GeForce GTX 280



SSAO: Horizon Based Results



SS: Directional Occlusion

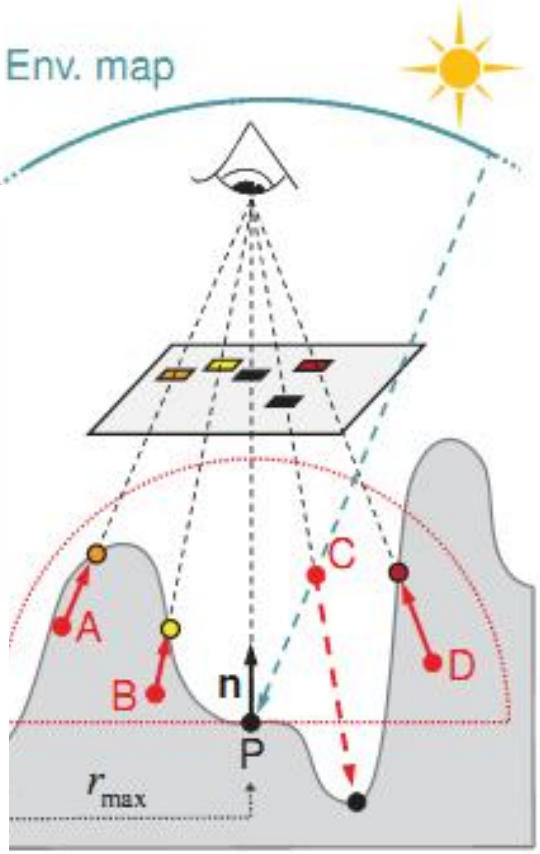


- Combines calculation of irradiance and occlusion at the same time
- Calculation:
  - Sample points around hemisphere centered at position in the depth buffer
  - Project points into z-buffer surface
  - Samples that are below the surface occlude, those that are above allow radiance through

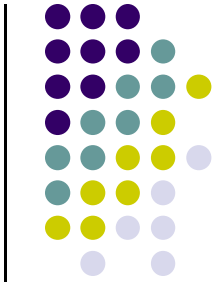
$$L_{\mathrm{dir}}(\mathbf{P}) = \sum_{i=1}^{N} \frac{\rho}{\pi} L_{\mathrm{in}}(\omega_i) V(\omega_i) \cos\theta_i \Delta\omega$$

SSDO:

- Allows for multiple colored soft shadows
- Cost grows with number of light sources



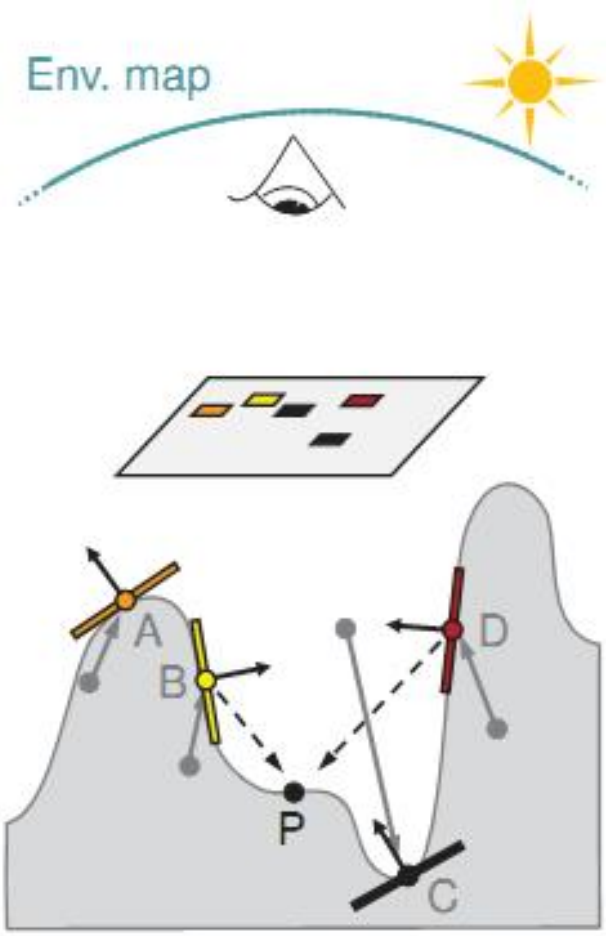
SS: Single Indirect Bounce

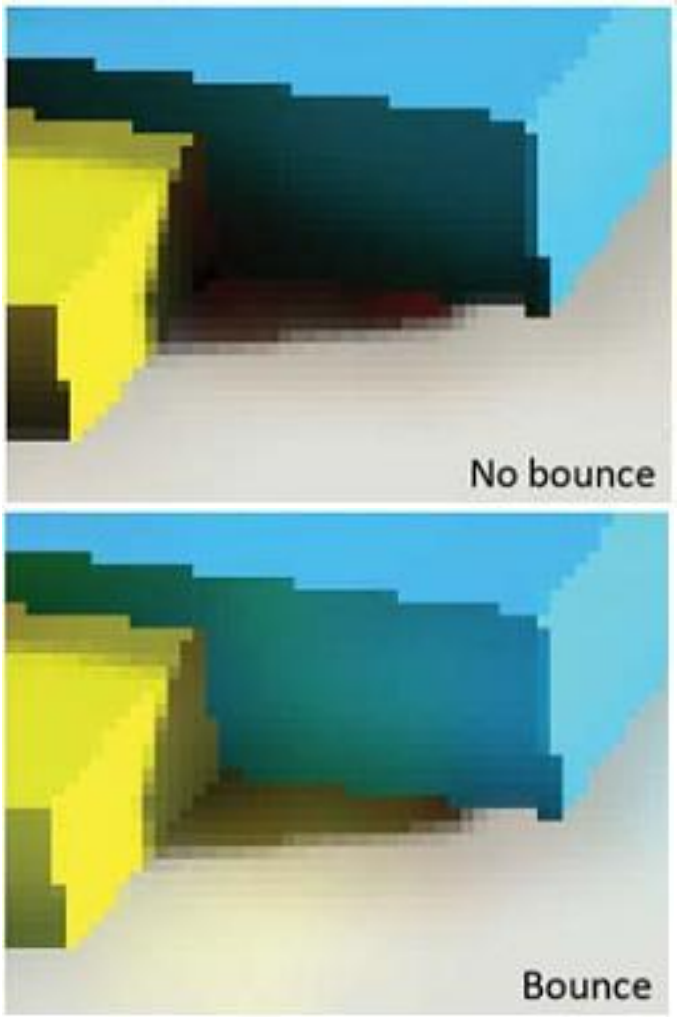


- Same sample surface points from previous stage used
- View points as small surfaces
- Project light onto point p attenuated by:
  - Distance to sample point
  - Area of sample
  - Orientation of point normal and of surface normal relative to the direction of incoming indirect light

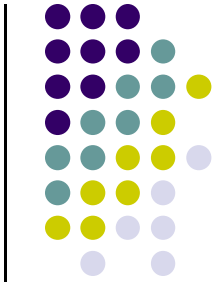
$$L_{\text{ind}}(\mathbf{P}) = \sum_{i=1}^{N} \frac{\rho}{\pi} L_{\text{pixel}}(1 - V(\omega_i)) \frac{A_s \cos\theta_{s_i} \cos\theta_{r_i}}{d_i^2}$$

SS: Single Indirect Bounce



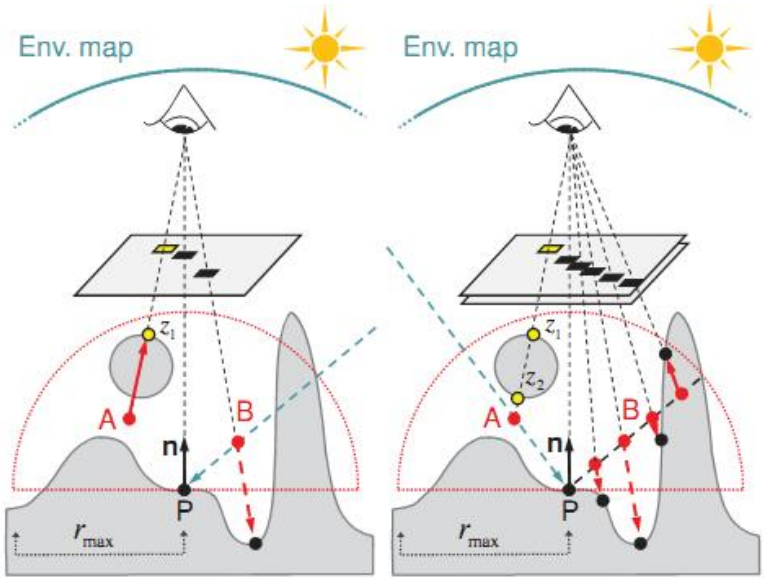


Single Indirect Bounce: Some Issues

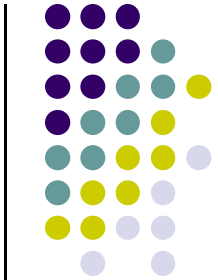


- Reflections are highly dependent on the view
  - Some reflection information not rendered (occluded or back facing)

- Incorrect occludes and visibility
- Only models local effects
- Some scenes close to PBRT
- Corrections are expensive
  - Multiple Cameras (+160%)
  - Depth Peel (+30%)

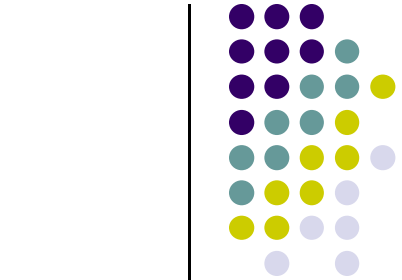


SS: Reflective Shadow Maps

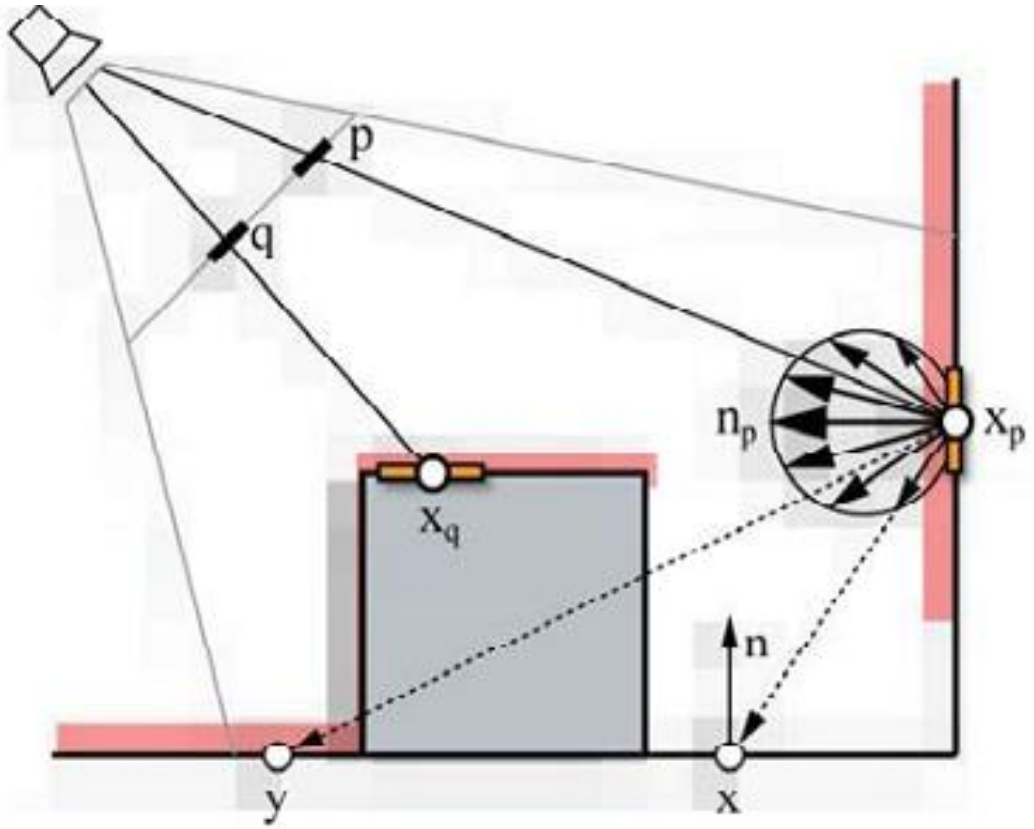


- Render scene from the view of the light source
- Record the following:
  - Depth (position), normal, and flux
- These are the only surfaces from which a first bounce of reflected light can originate
- It is argued that one bounce is sufficient for some applications

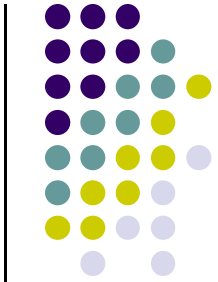
RSM: Pixel Light Sources



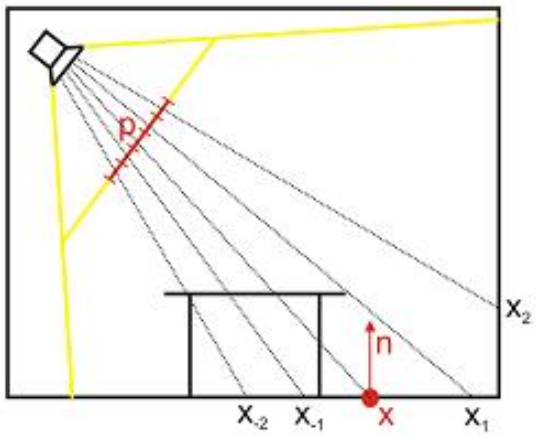
- View each pixel in the shadow map as a light source
- Will not look correct without AO calculation
- Will introduce error

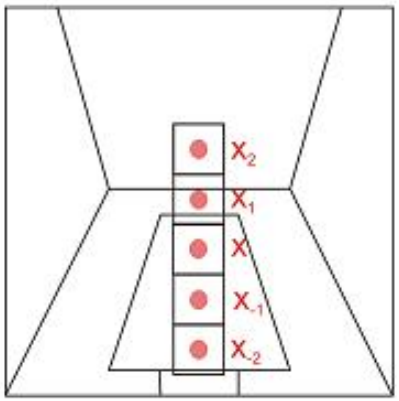


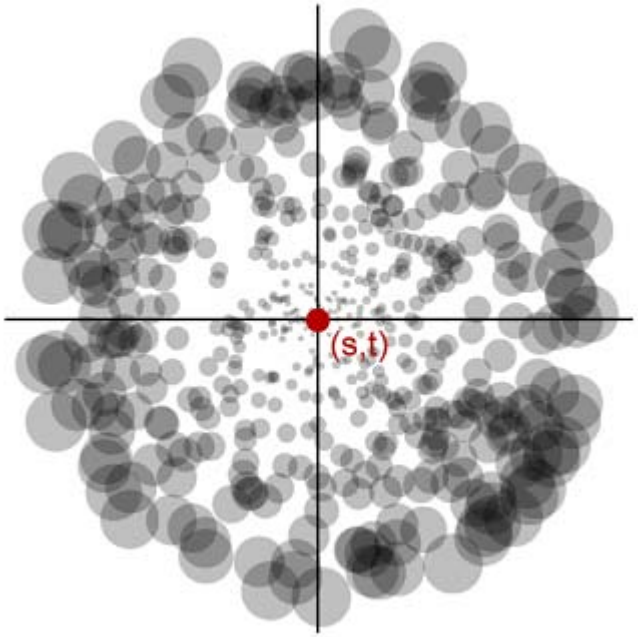
RSM: Gathering



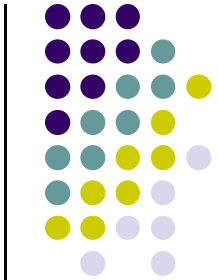
- Choose a small number of well chosen pixel lights
- More pixel lights in areas of higher flux.
  - A few hundred seems to work well (400)
- Center sampling around surface point to illuminate



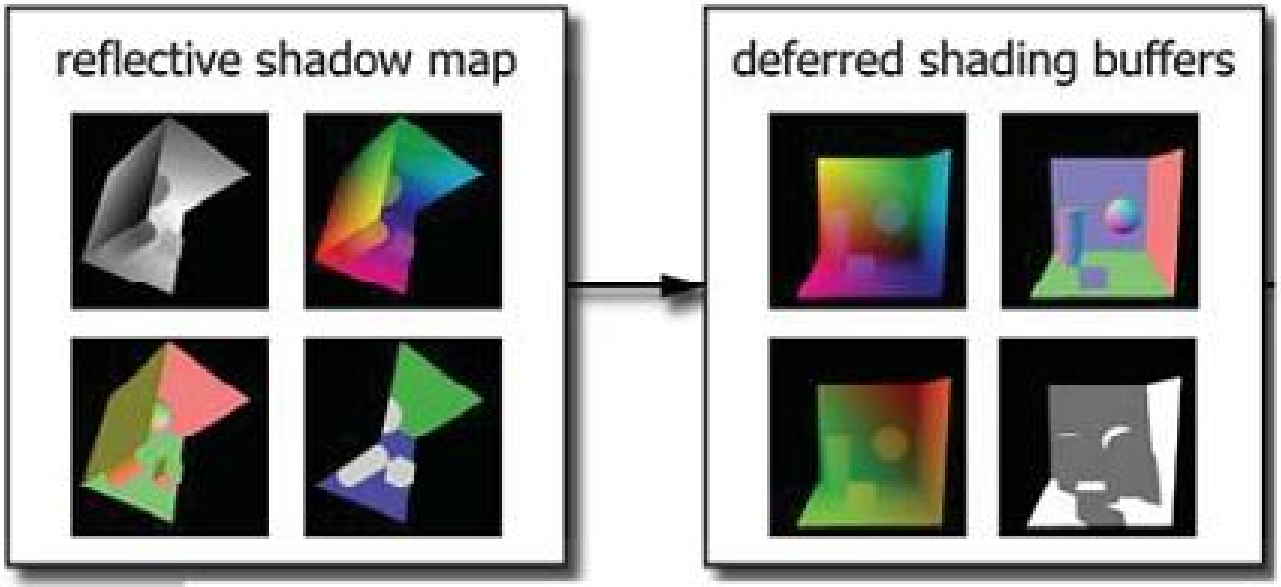




RSM: Render



- Render RSM and Deferred Shading buffers
- Gather indirect illumination (on low resolution image)
- Interpolation of indirect illumination where possible
- Compute illumination directly where interpolation fails

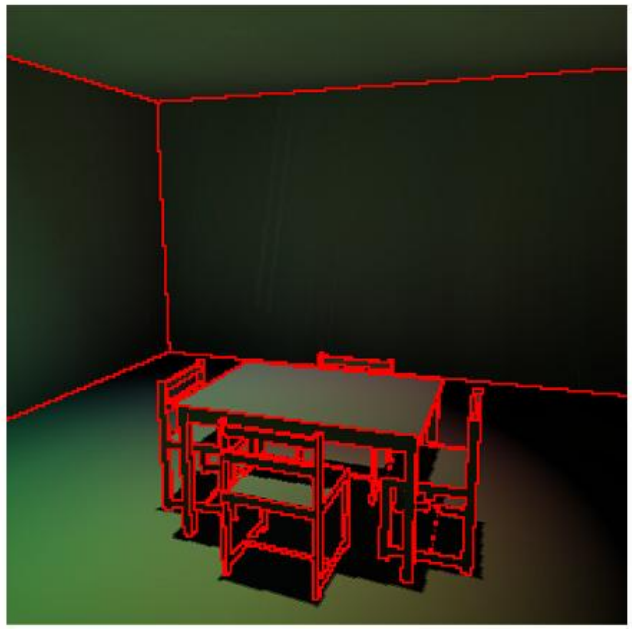


RSM: Result

- Does not support many reflections or light types
- Good deal of banding
  - Same sampling distribution use throughout renders

$$(s + r_{\max}\xi_1 \sin(2\pi\xi_2), t + r_{\max}\xi_1 \cos(2\pi\xi_2))$$

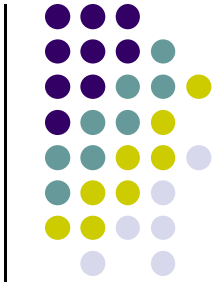
  - Different distributions between renders will lead to temporal incoherence
- Interpolation works well
- Areas with varying normal can not be reliably interpolated
- Real time frame rates



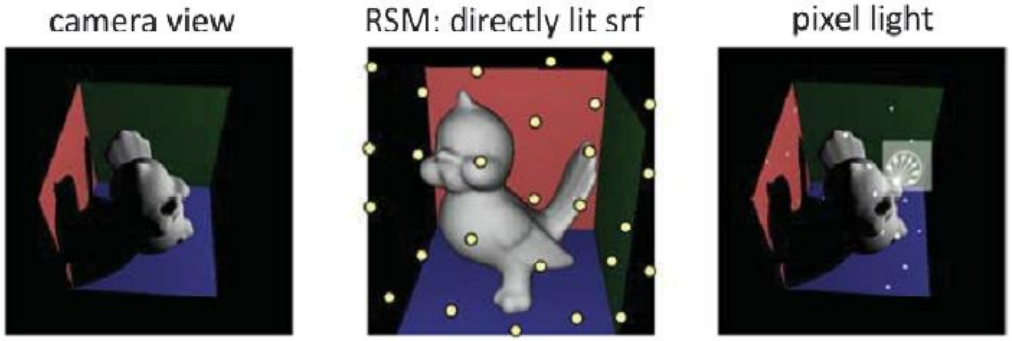
RSM: Result

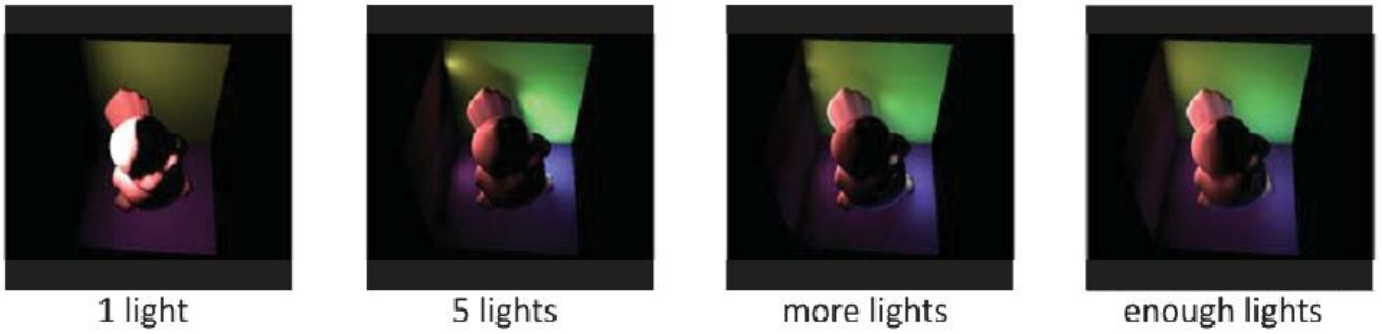
Rendered with AO and RSM

RSM: Shooting

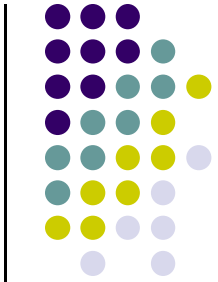


- Choose a set of Pixel lights for the entire scene
  - Use samples as Virtual Point Lights (hemispherical)
- Use them to illuminate the scene by splatting quads
- Accumulated in screen space, quads mapped to RSM



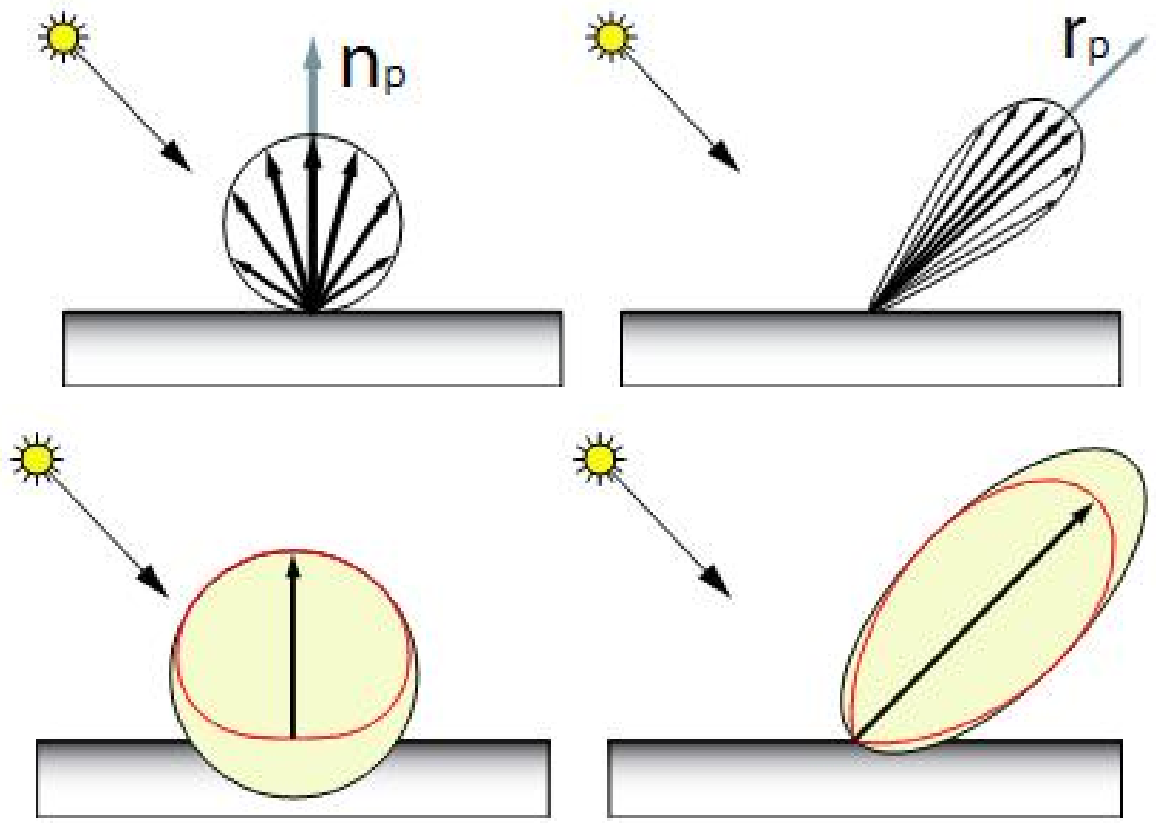


Shooting : GPU Optimizations



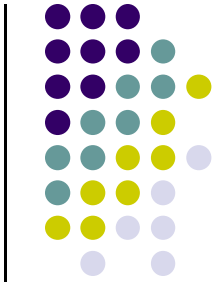
- Limit lights by distance/energy
- More VPLs on glossy of surface
- Bound regions for VPLs tighter than quads

Shooting : Bounding Regions

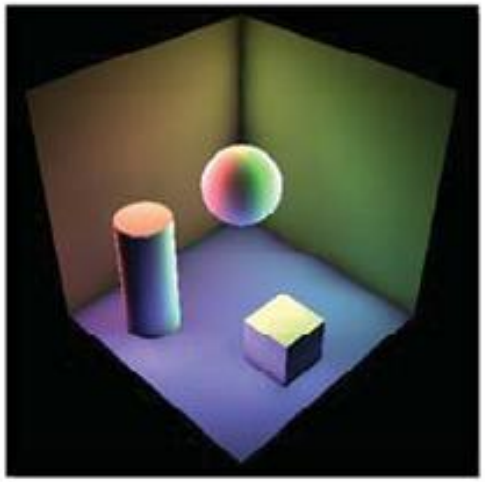


Computational load shifted to vertex shader, more efficient over all.

Shooting: Clamping

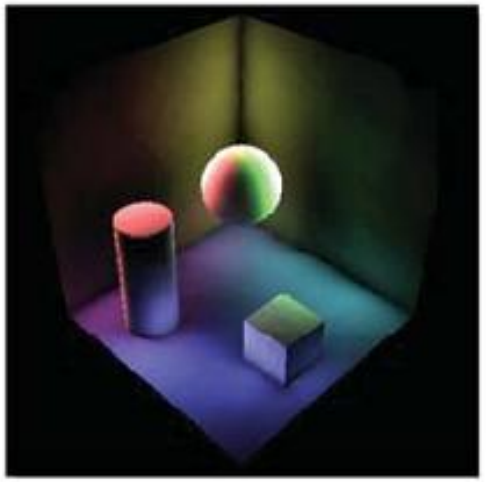


- Clamping range for VPLs
- Increased frame rate
- Decreased accuracy of long distance indirect illumination



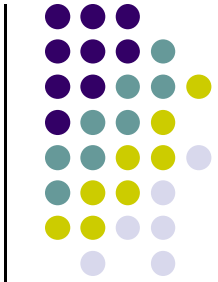
$I=0.01I_0$

$I=0.05I_0$



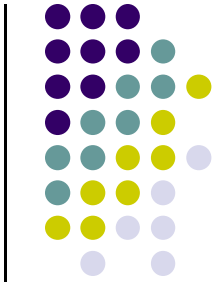
$I=0.07I_0$

Limitations of Gather and Shoot

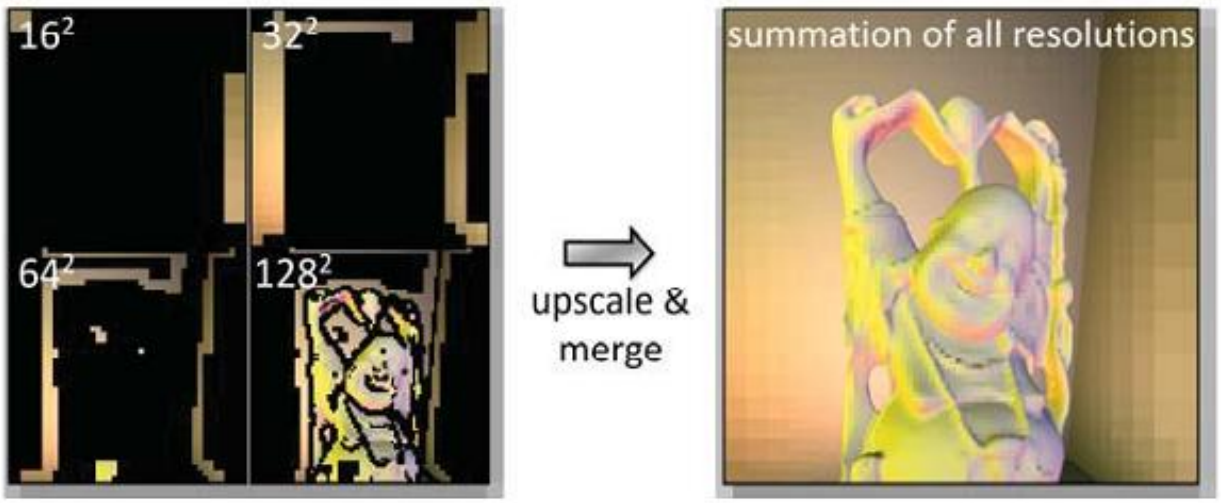


- Gathering: many texture lookups
- Shooting: overdraw since many splat geometries overlap in screen space
- Need to take full advantage of low frequency nature of the indirect lighting

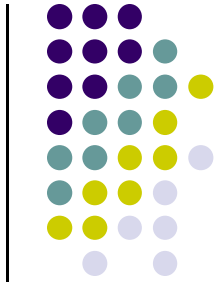
Hierarchical Approach



- Create RSM and deferred render
- Generate VPLs as before
- Create initial subsplats in an efficient way
- Subdivide splat according to the complexity of the scene
- Avoid overdraw and high number of texture reads

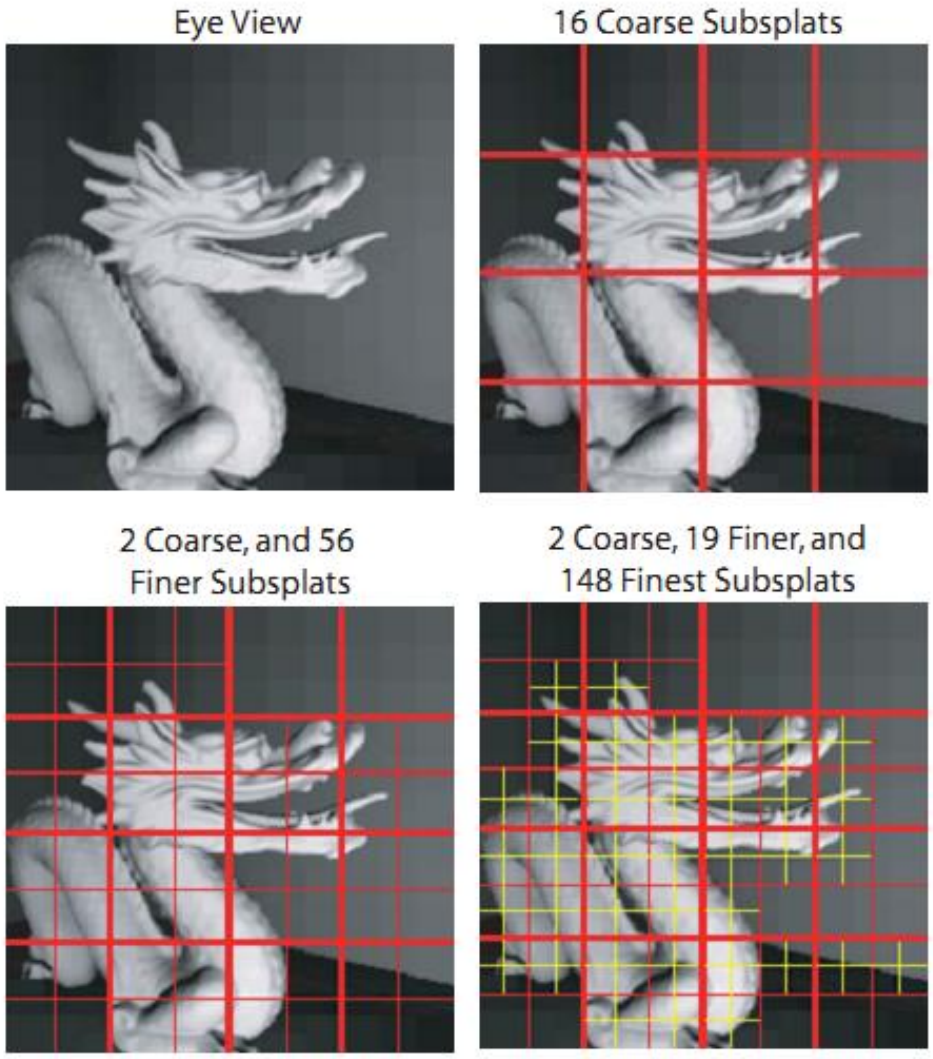


Adaptive Refinement

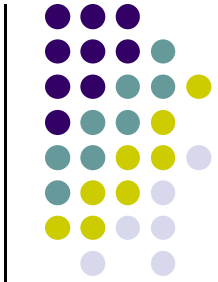


- Create min-max mipmap from view
  - Detect discontinuities in normal
    - X, Y, Z directions separated into channels
  - Detect discontinuities in depth
- Distribute splats at lowest resolution of mipmap
- Splats which contain discontinuities are subdivided up to the maximum resolution of the mipmap
- All VPLs contribute to the full scene

Adaptive Refinement

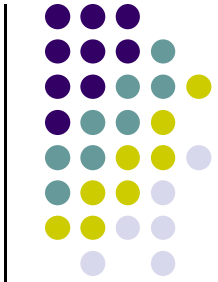


Hierarchical Approach: Conclusion



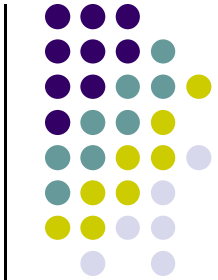
- Methods avoid pre-computation allowing for highly dynamic scenes
- Less texture reads, reads at different resolutions
- Will not provide constant frame rates since subdivision of splats depends on screen complexity
- Could require geometry shader which is not available on all graphics cards

General: Conclusion



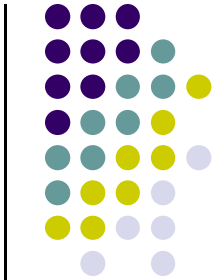
- Methods avoid pre-computation allowing for highly dynamic scenes
- Different techniques concentrate on different aspects of the complete render equation to differing degrees of accuracy
- Many of the techniques might cause some color bleeding through occludes, though the effect is negligible

References



- Martin Mittring, Finding next gen: CryEngine 2, ACM SIGGRAPH 2007 courses, August 05-09, 2007, San Diego, California
- Carsten Dachsbacher , Marc Stamminger, Splatting indirect illumination, Proceedings of the 2006 symposium on Interactive 3D graphics and games, March 14-17, 2006, Redwood City, California
- Carsten Dachsbacher , Jan Kautz, Real-time global illumination for dynamic scenes, ACM SIGGRAPH 2009 Courses, p.1-217, August 03-07, 2009, New Orleans, Louisiana
- Louis Bavoil , Miguel Sainz , Rouslan Dimitrov, Image-space horizon-based ambient occlusion, ACM SIGGRAPH 2008 talks, August 11-15, 2008, Los Angeles, California

References



- Dominic Filion , Rob McNaughton, Effects & techniques, ACM SIGGRAPH 2008 classes, August 11-15, 2008, Los Angeles, California
- Carsten Dachsbacher , Marc Stamminger, Reflective shadow maps, Proceedings of the 2005 symposium on Interactive 3D graphics and games, April 03-06, 2005, Washington, District of Columbia
- Tobias Ritschel , Thorsten Grosch , Hans-Peter Seidel, Approximating dynamic global illumination in image space, Proceedings of the 2009 symposium on Interactive 3D graphics and games, February 27-March 01, 2009, Boston, Massachusetts
- Greg Nichols , Chris Wyman, Multiresolution splatting for indirect illumination, Proceedings of the 2009 symposium on Interactive 3D graphics and games, February 27-March 01, 2009, Boston, Massachusetts