GDC 09

learn
network
inspire

www.GDConf.com

Game Developers Conference®

March 23-27, 2009 | Moscone Center, San Francisco

DirectX 10 / 11 Visual Effects

Simon Green, NVIDIA

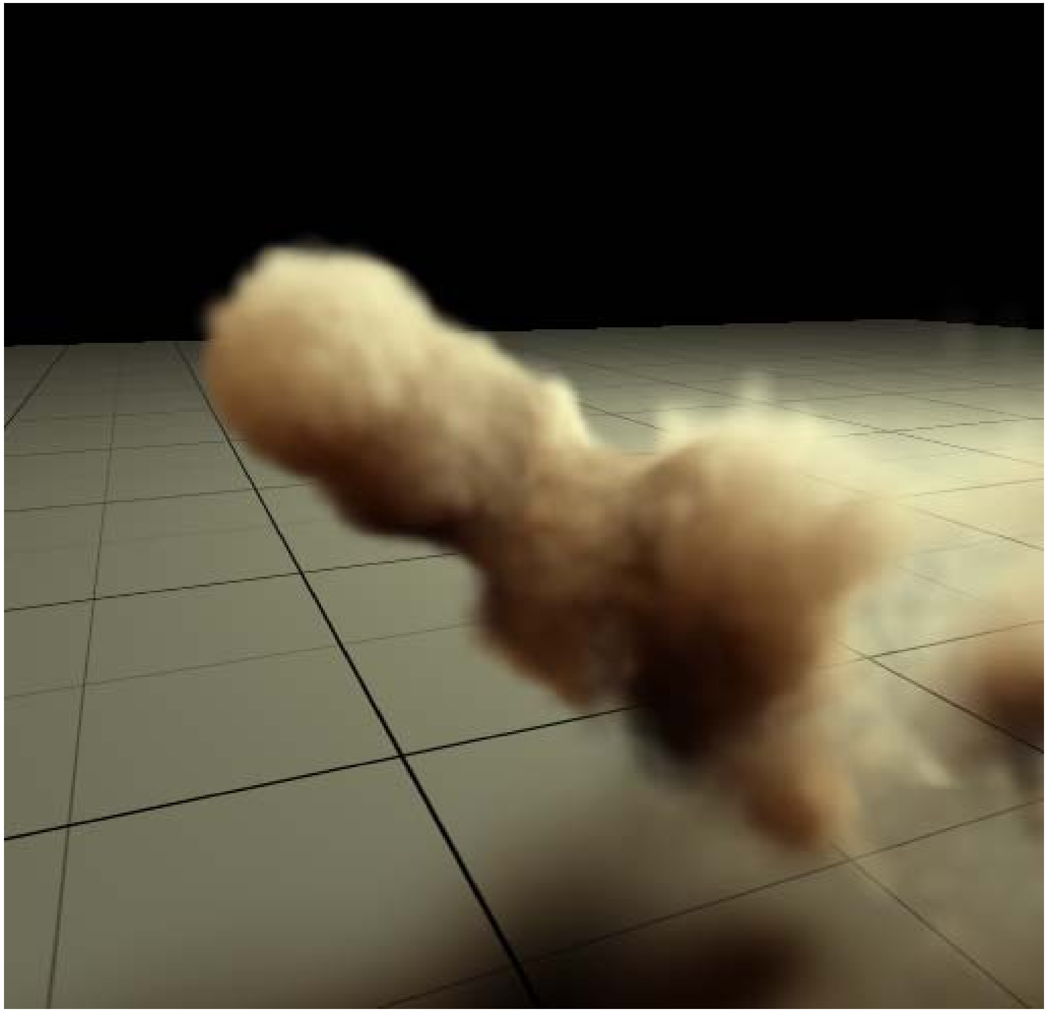
Introduction

- Graphics hardware feature set is starting to stabilize and mature
- New general-purpose compute functionality (DX11 CS)
  - enables new graphical effects
  - and allows more of game computation to move to the GPU
  - Physics, AI, image processing
- Fast hardware graphics combined with compute is a powerful combination!
- Next generation consoles will likely follow this path

Overview

- Volumetric Particle Shadowing
- Horizon Based Ambient Occlusion (HBAO)
- DirectX 11 Compute Shader Effects

Volumetric Particle Shadowing



Particle Systems in Today's Games

- Commonly used for smoke, explosions, spark effects
- Typically use relatively small number of large particles (10,000s)
- Rendered using point sprites with artist painted textures
  - Use animation / movies to hide large particles
- Sometimes include some lighting effects
  - normal mapping
- Don't interact much with scene

Particle Systems in Today's Games

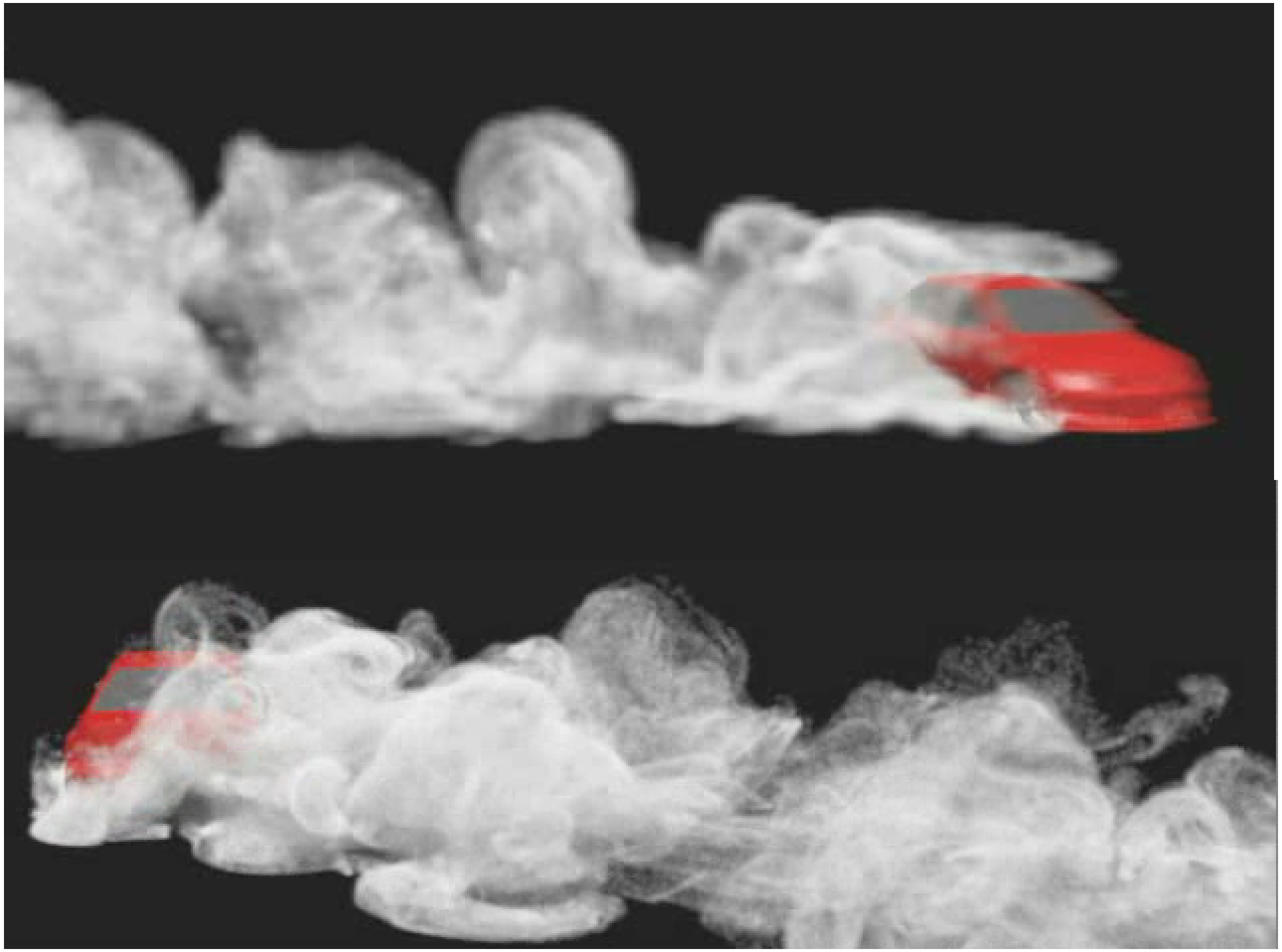
- Can get some great effects with current technology

- Game screen shot here (pending approval)
- World in Conflict?

Tomorrow's Particle Systems

- Will likely be more similar to particle effects used in film
- Millions of particles
- Physically simulated
  - With artist control
- Interaction (collisions) with scene and characters
- Simulation using custom compute shaders or physics middleware
- High quality shading and shadowing

Tomorrow's Particle Systems - Example



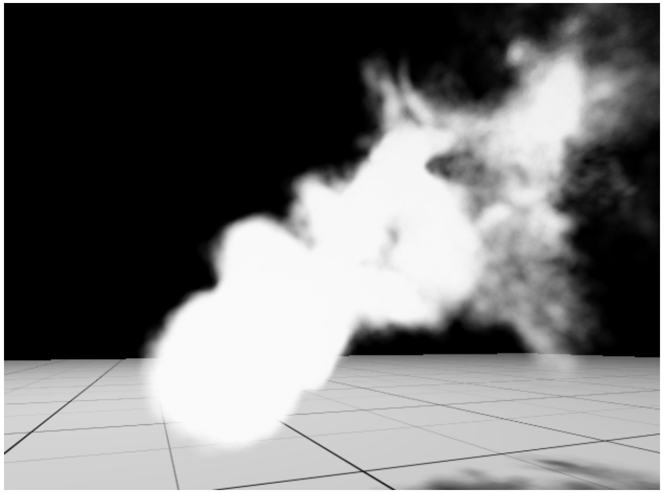
Low Viscosity Flow Simulations for Animation, Molemaker et al., 2008

Volume Shadowing

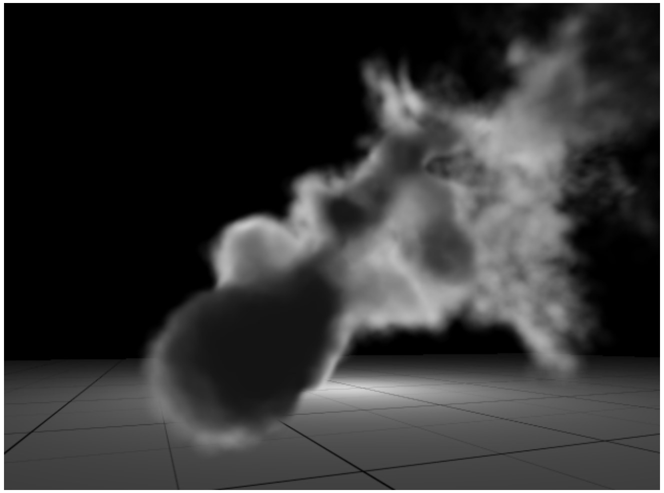
- Shadows are very important for diffuse volumes like smoke
  - show density and shape
- Not much diffuse reflection from a cloud of smoke
  - traditional lighting doesn't help much
- Usually achieved in off-line rendering using deep shadow maps
  - still too expensive for real time

Volume Shadowing

Before



After

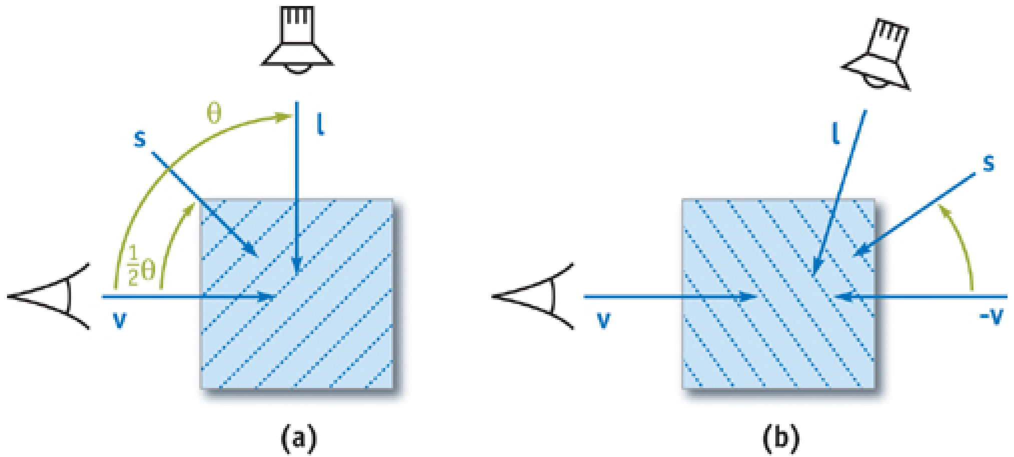


Half-Angle Slice Rendering

- Very simple idea
- Based on old volume rendering technique by Joe Kniss et. Al [1]
- Only requires sorting particles along a given axis
  - you're probably already doing this
- Plus a single 2D shadow texture
  - no 3D textures required
- Works well with simulation and sorting done on GPU (compute)

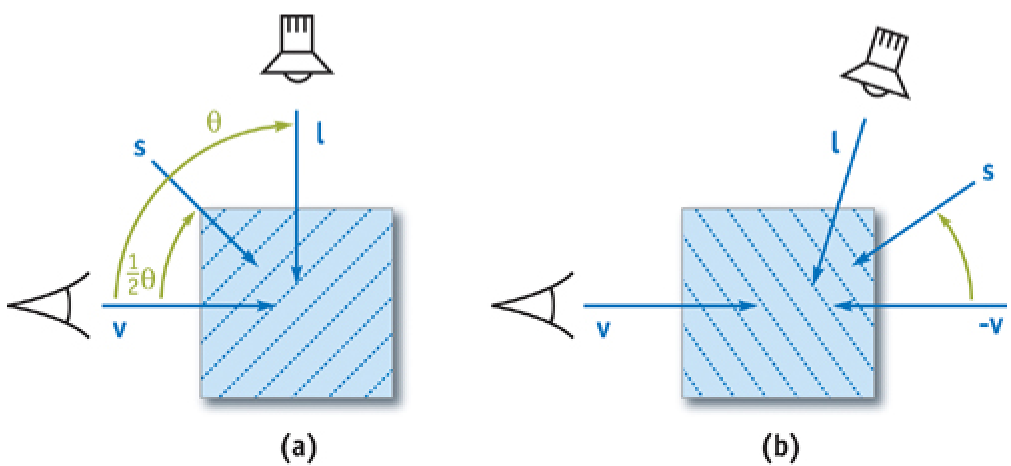
Half-Angle Slice Rendering

- Calculate vector half way between light and view direction
- Render particles in slices perpendicular to this half-angle vector



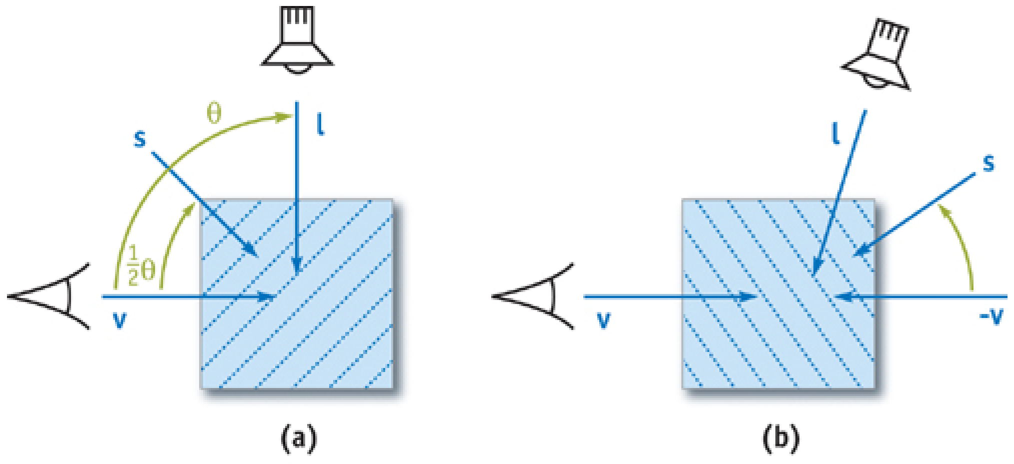
Half-Angle Slice Rendering

- » Same slices are visible to both camera and light
- » Lets us accumulate shadowing to shadow buffer *at the same time* as we are rendering to the screen



Half-Angle Slice Rendering

- Need to change rendering direction (and blend mode) based on dot(l, v)
- if (dot(l, v) > 0) - render front-to-back
- if (dot(l,v) < 0) – render back-to-front
- Always render from front-to-back w.r.t. light



Half-Angle Slice Rendering

» Sort particles along half-angle axis
  - based on dot(p, h)
  - can be done very quickly using compute shader
» Choose a number of slices
  - more slices improves quality
  - but causes more draw calls and render target switches
» batchSize = numParticles / numSlices
» Render slices as batches of particles starting at i*batchSize
» Render particles as billboards using GS

Pseudo-Code

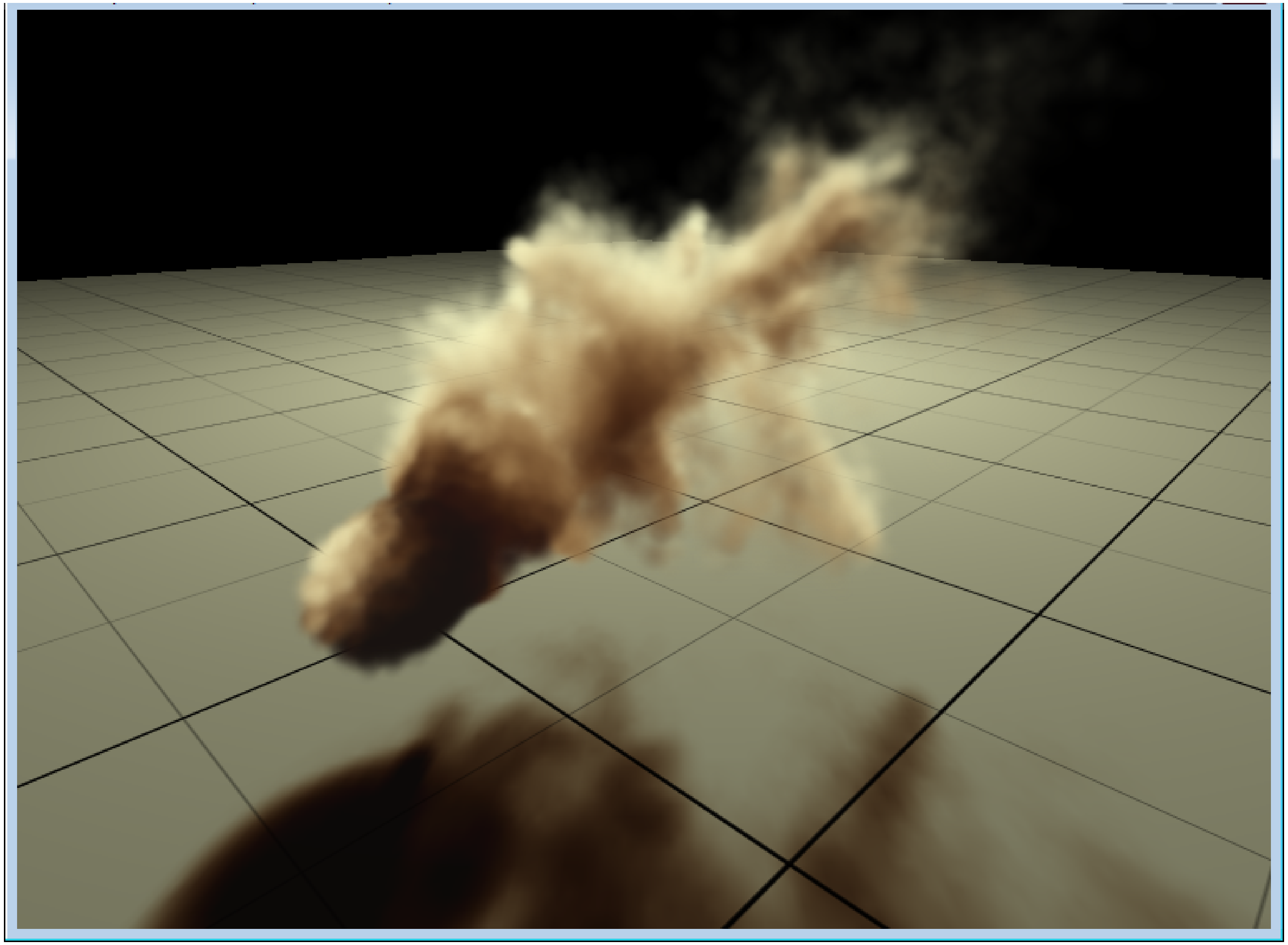
```
If (dot(v, l) > 0) {
   h = normalize(v + l)
   draw front-to-back
} else {
   h = normalize(-v + l)
   draw back-to-front
}
sort particles along h
batchSize = numParticles / numSlices
for(i=0; i<numSlices; i++) {
   draw particles to screen
        looking up in shadow buffer

   draw particles to shadow buffer
}
```

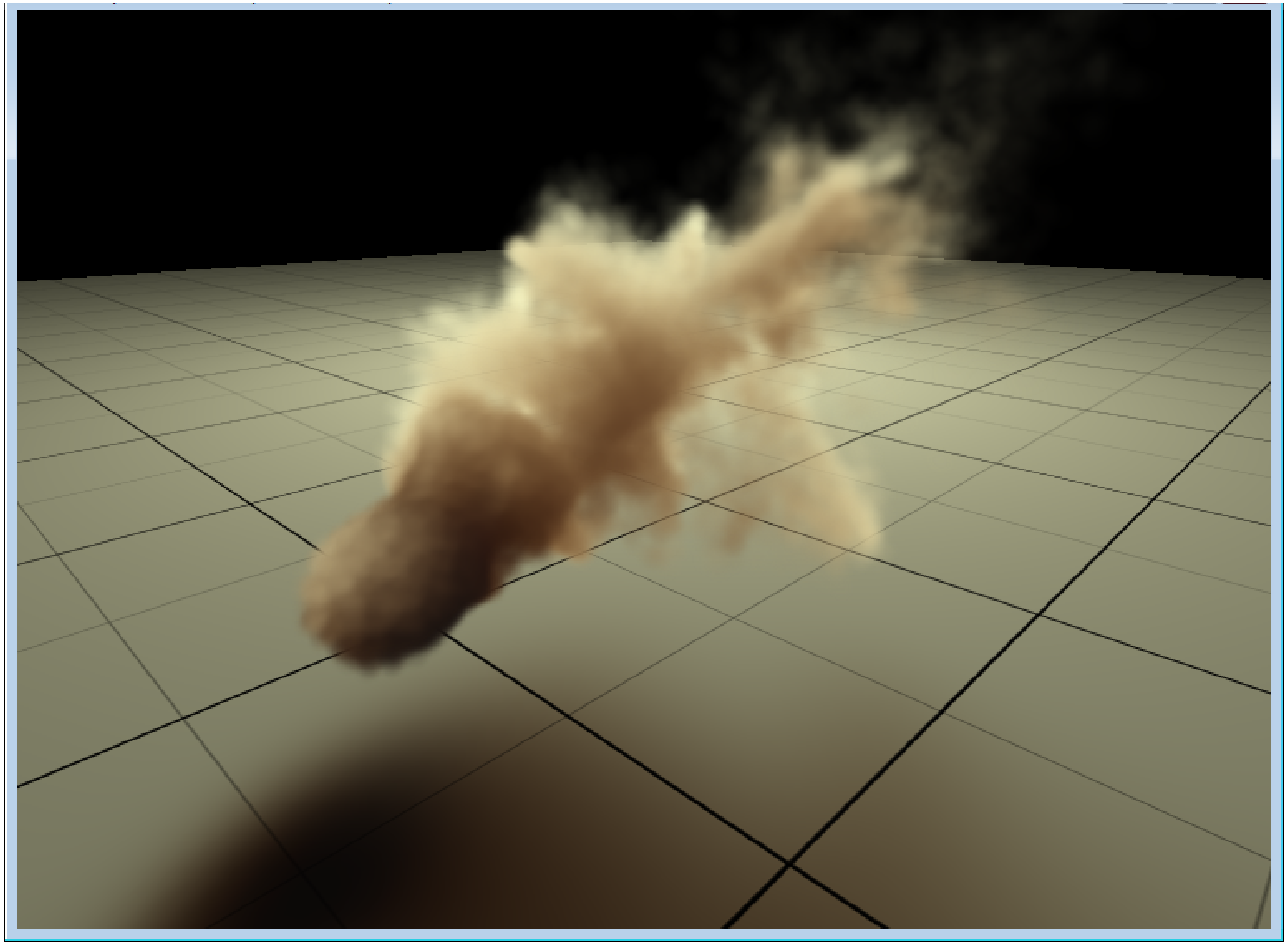
Tricks

- Shadow buffer can be quite low resolution (e.g. 256 x 256)
- Can also use final shadow buffer to shadow scene
- Screen image can also be rendered at reduced resolution (2 or 4x) to reduce fill rate requirements
- Can blur shadow buffer at each iteration to simulate scattering:

Without Scattering



With Scattering

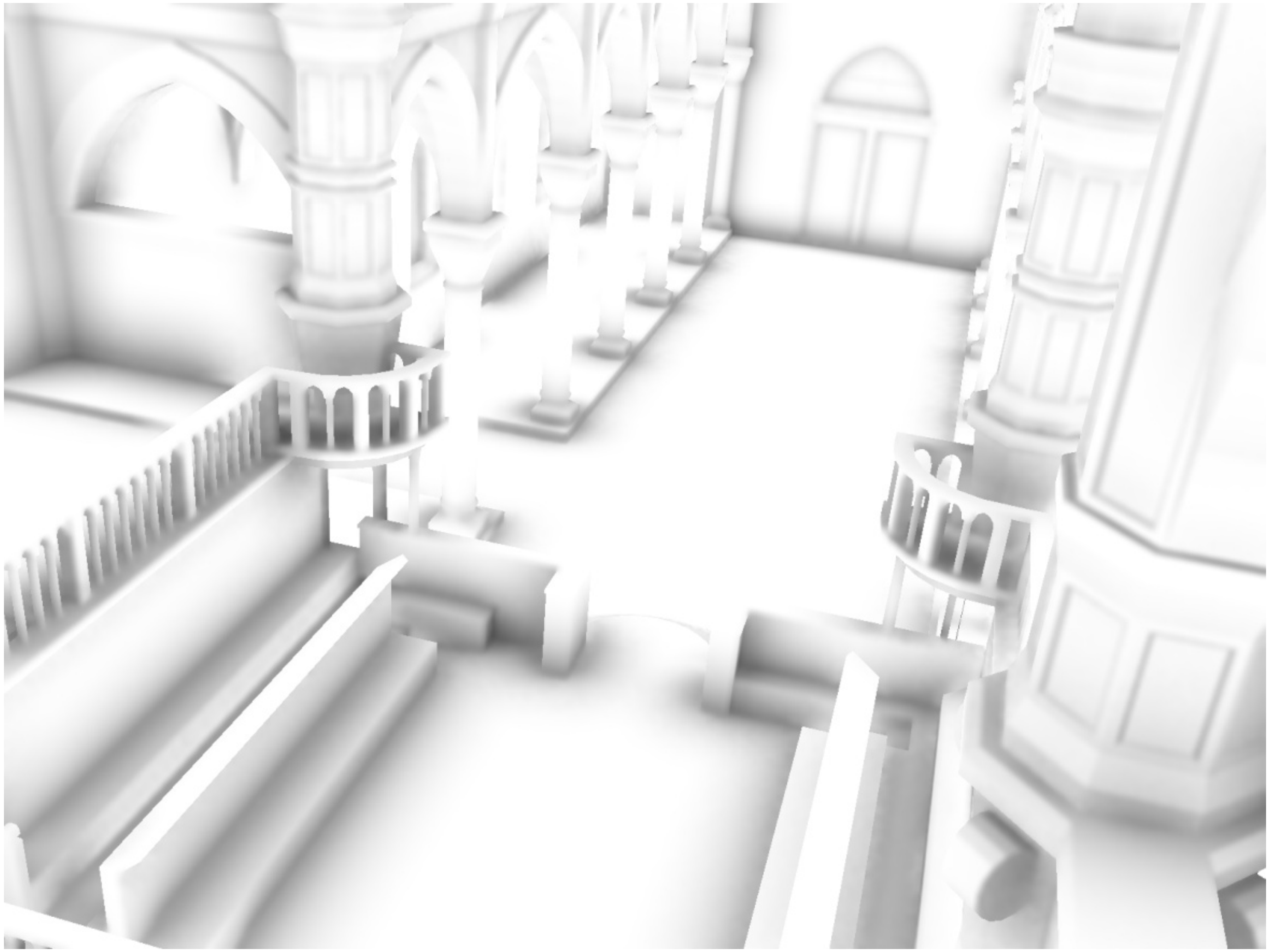


Demo

Volume Shadowing - Conclusion

- Very simple to add to existing particle system renderer
- Only requires depth-sorting along a different axis
  - Can be done using CPU radix sort or Compute
- Plus a single shadow map
- Simulating particle systems on the GPU can enable millions of particles in real-time

Horizon Based Ambient Occlusion

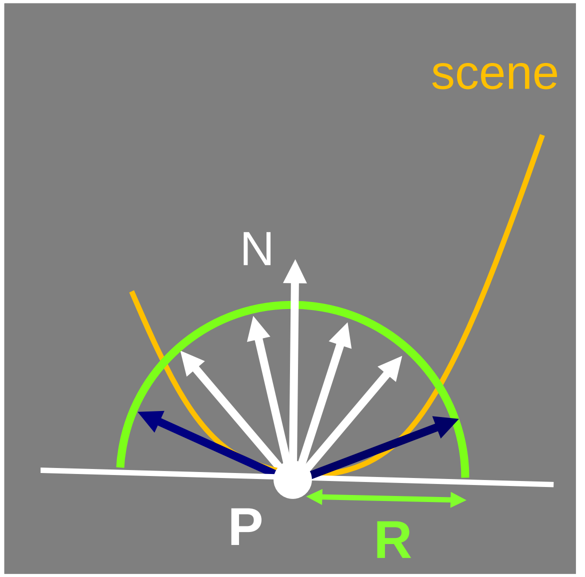


Ambient Occlusion

» Simulates lighting from hemi-spherical sky light

» Occlusion amount is % of rays that hit something within a given radius R

» Usually solved offline using ray-tracing



Ambient Occlusion

» Gives perceptual clues to depth, curvature and spatial proximity

Without AO

With AO

Screen Space Ambient Occlusion

» Approach introduced by

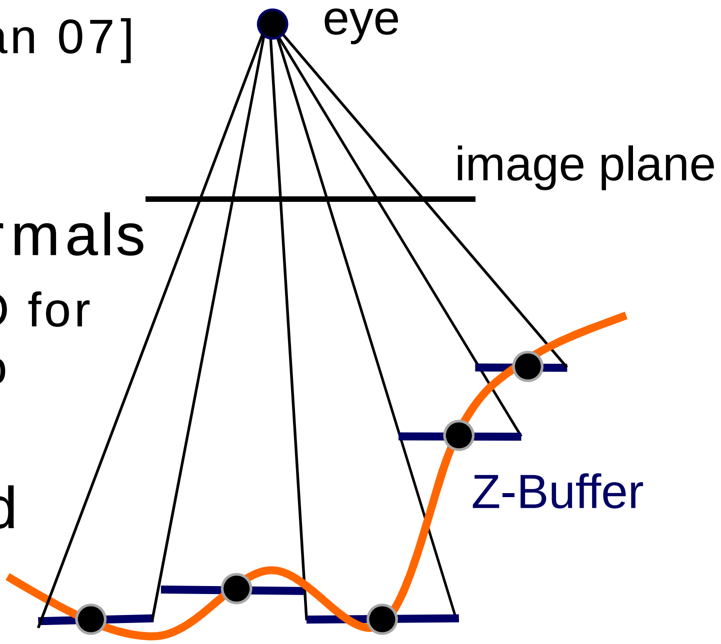
[Shanmugam and Orikan 07]
[Mittring 07]
[Fox and Compton 08]

» Input - Z-Buffer + normals

Render approximate AO for dynamic scenes with no precomputations

» Z-Buffer = Height field

$z = f(x,y)$



Horizon Based Ambient Occlusion

- Screen Space Ambient Occlusion (SSAO) technique presented at SIGGRAPH'08 and in ShaderX7 [2]

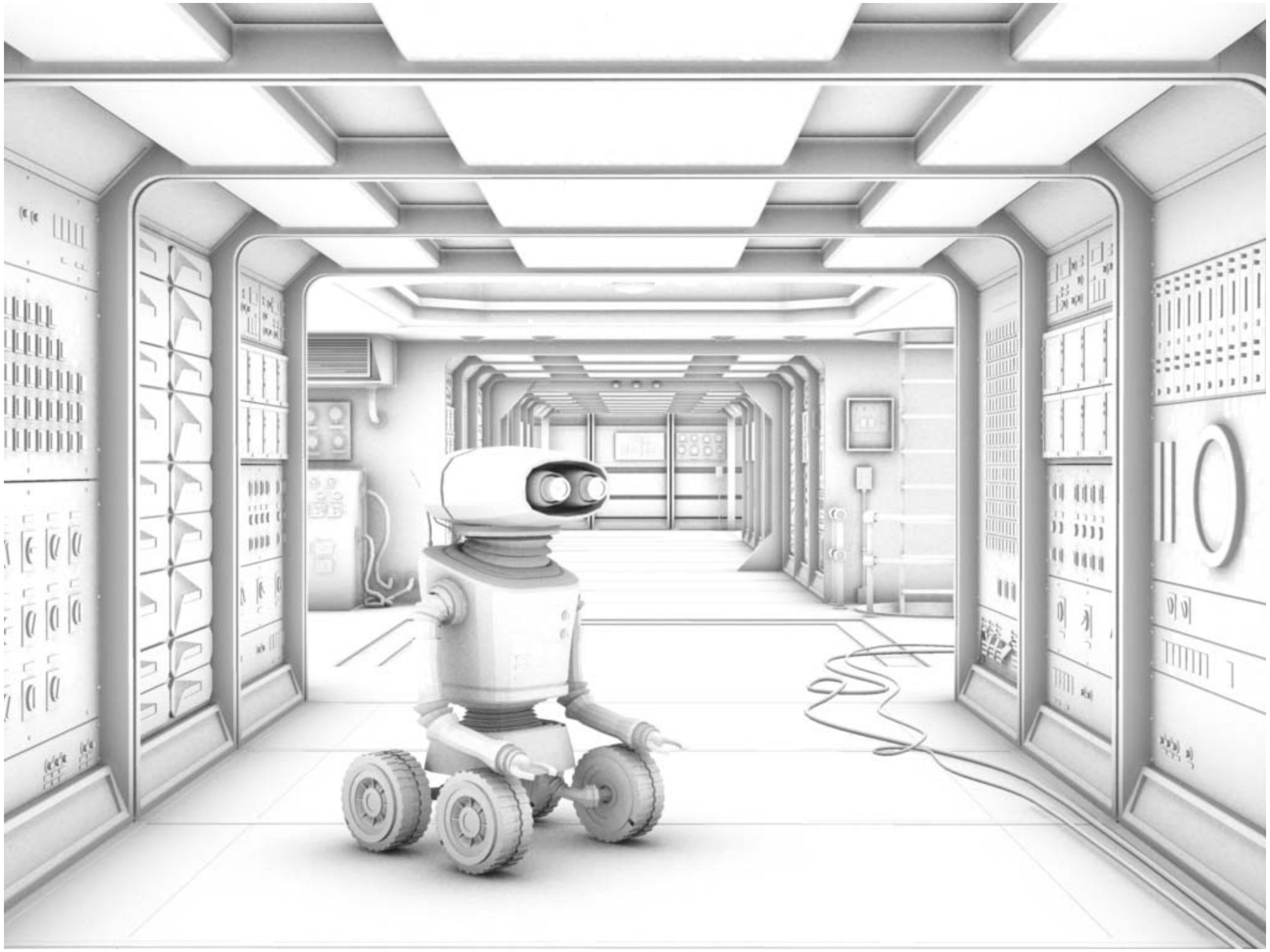
- HBAO Approach
  - Goal = approximate the result of ray tracing the depth buffer in 2.5D
  - Based on ideas from horizon mapping [Max 1986]

Integration in Games

» Implemented in DirectX 9 and DirectX 10

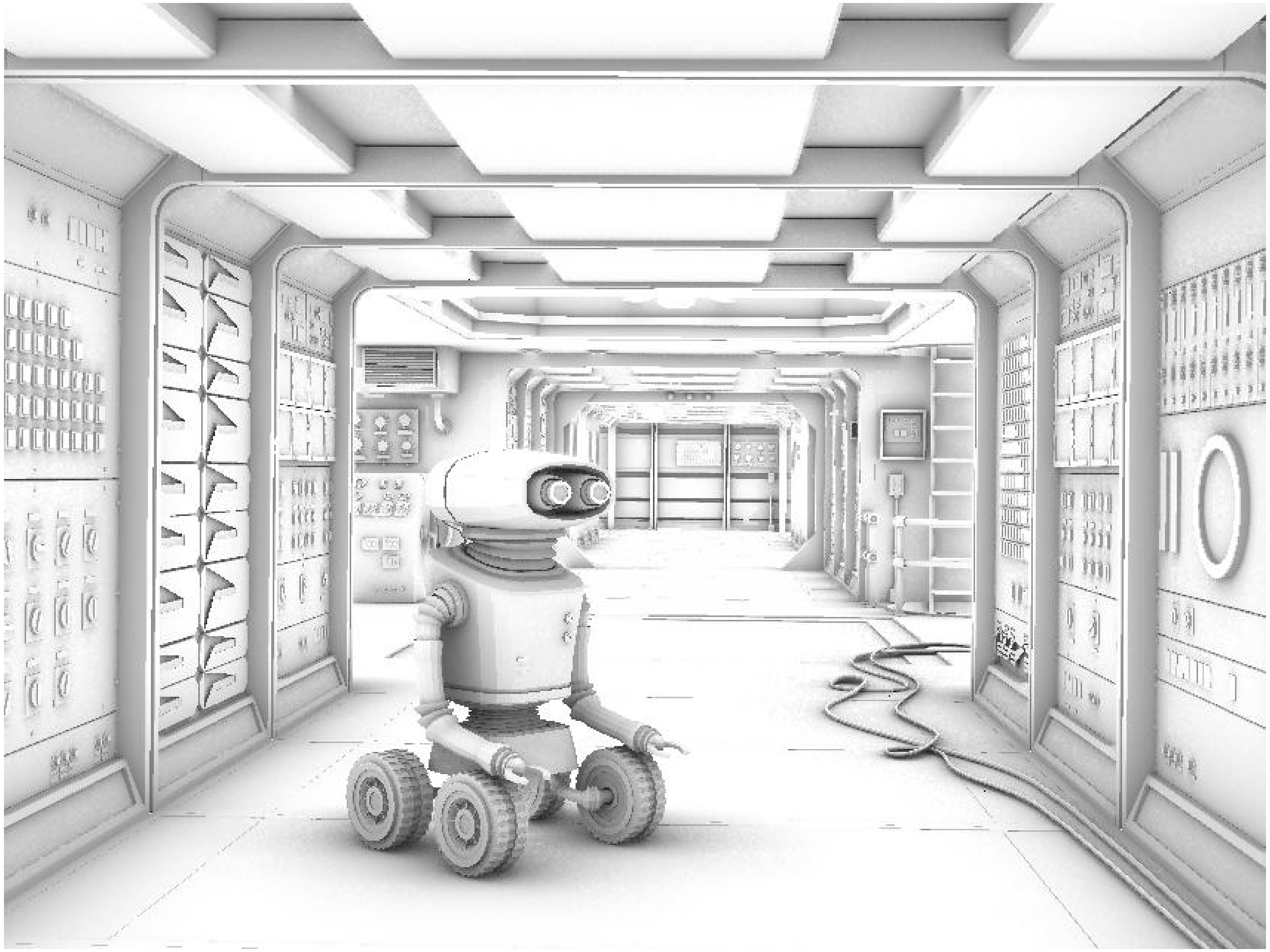
» Has been used successfully in several shipping games

Ray Traced AO



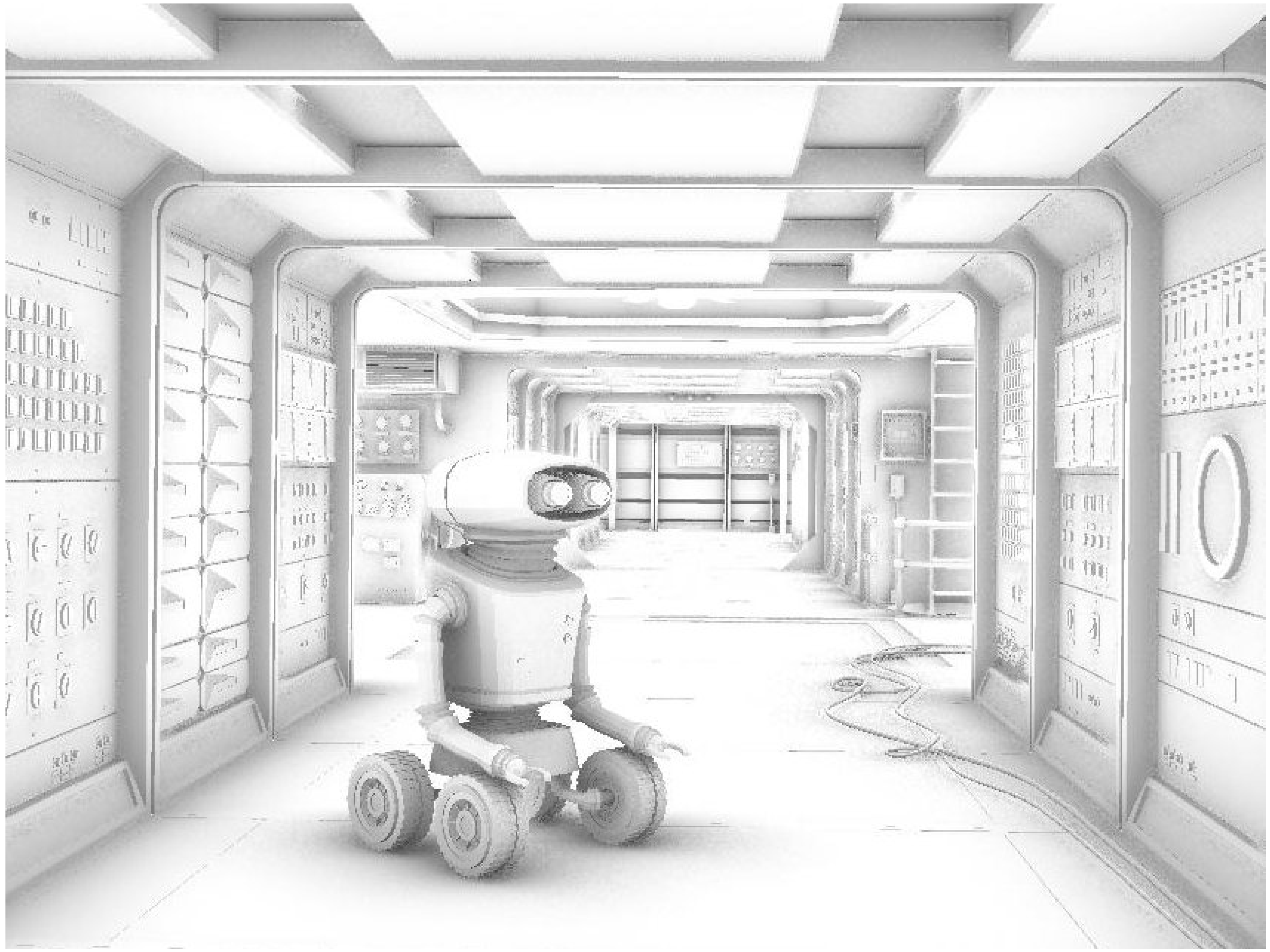
Several minutes with Gelato and 64 rays per pixel

HBAO with large radius



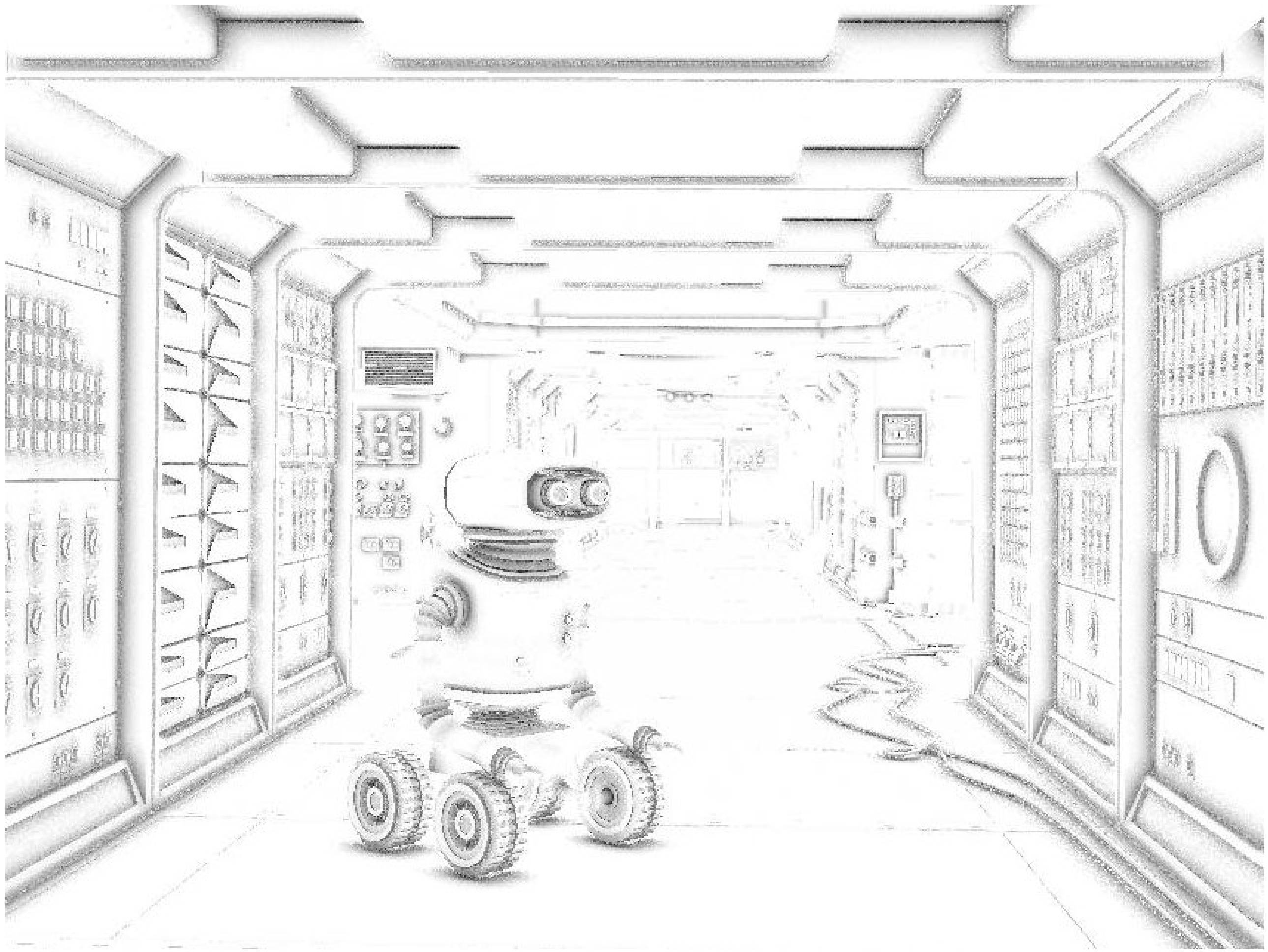
HBAO with 16x64 depth samples per pixel

HBAO with large radius



HBAO with 16x16 depth samples per pixel

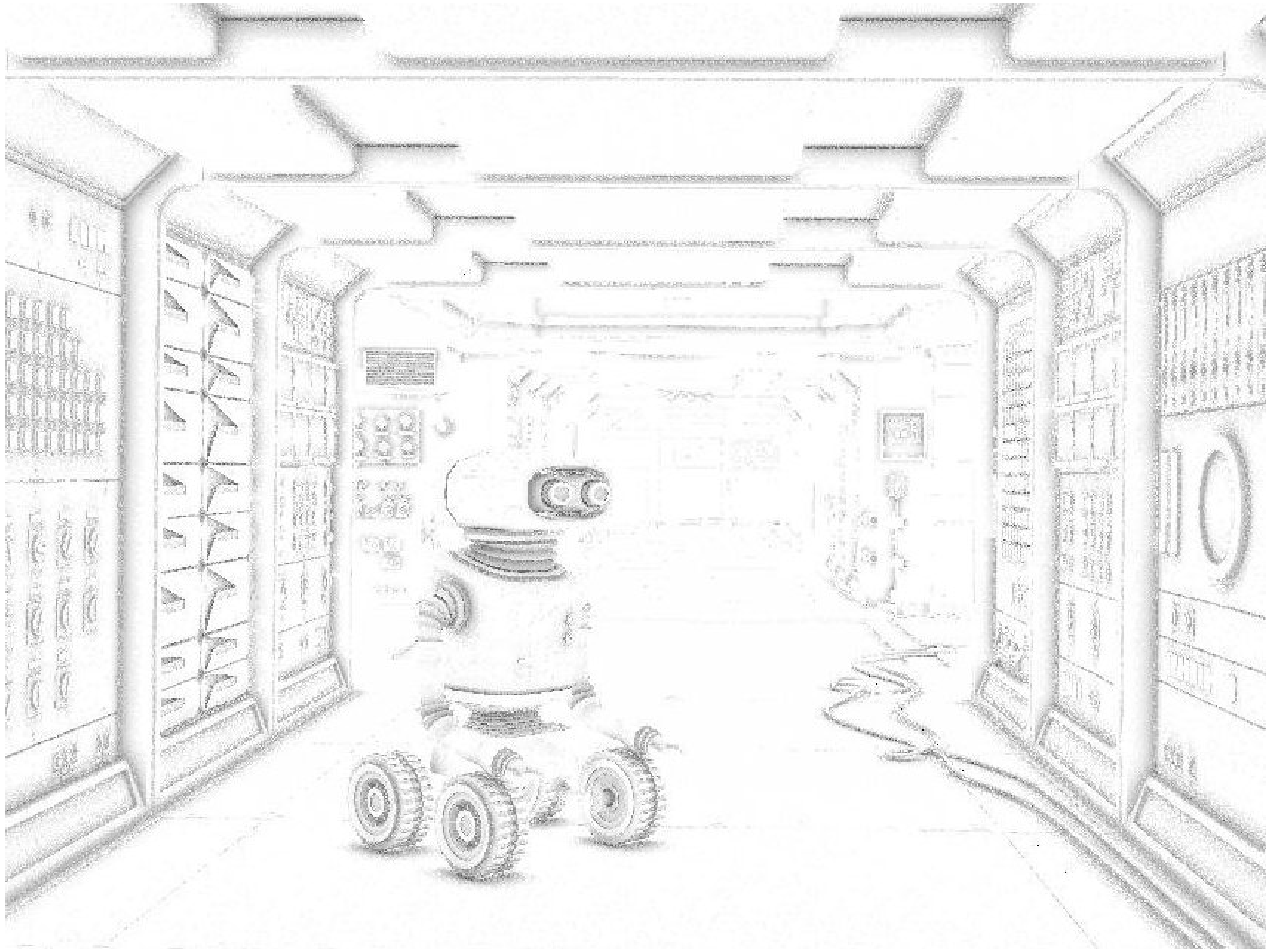
HBAO with small radius



"Crease shading" look

with 6x6 depth samples per pixel

HBAO with small radius



"Crease shading" look

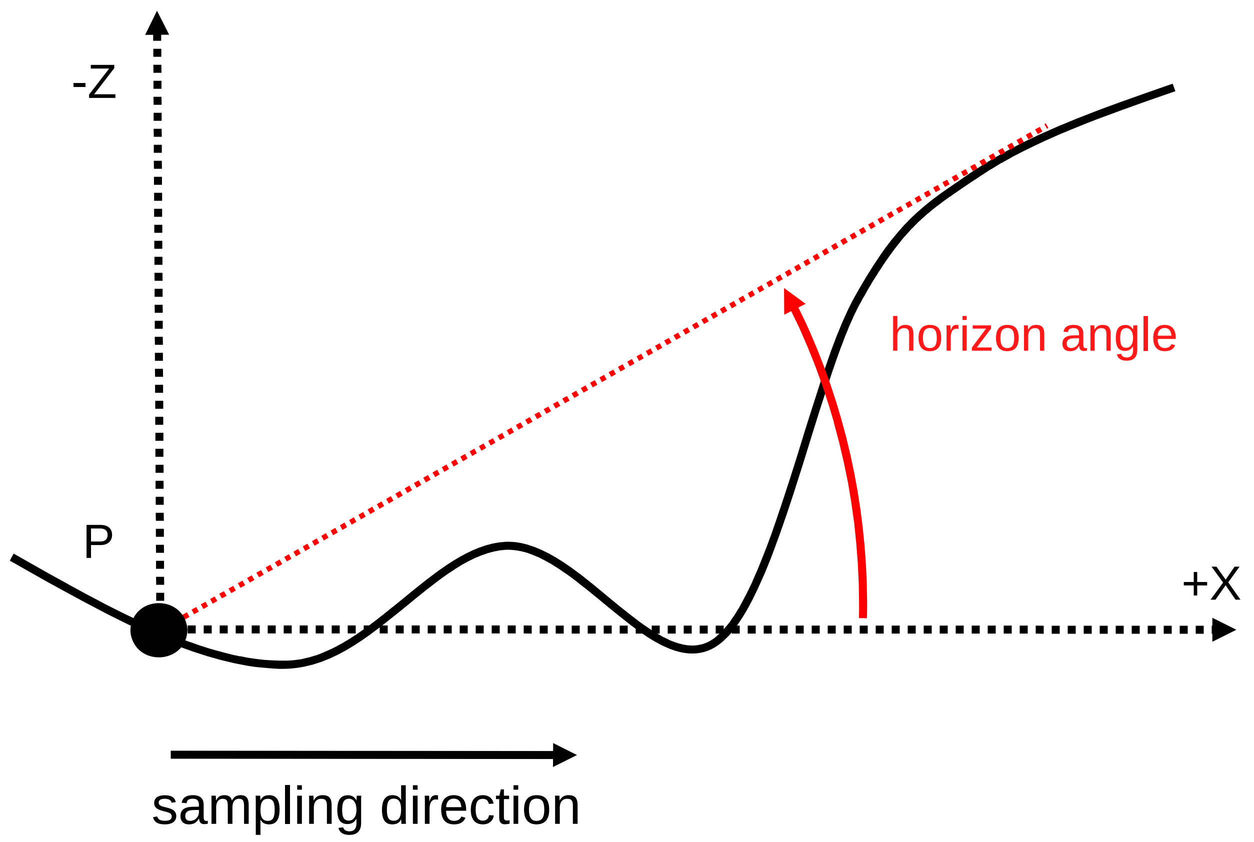
with 4x8 depth samples per pixel

HBAO Game Screenshots

» Screenshots pending approval

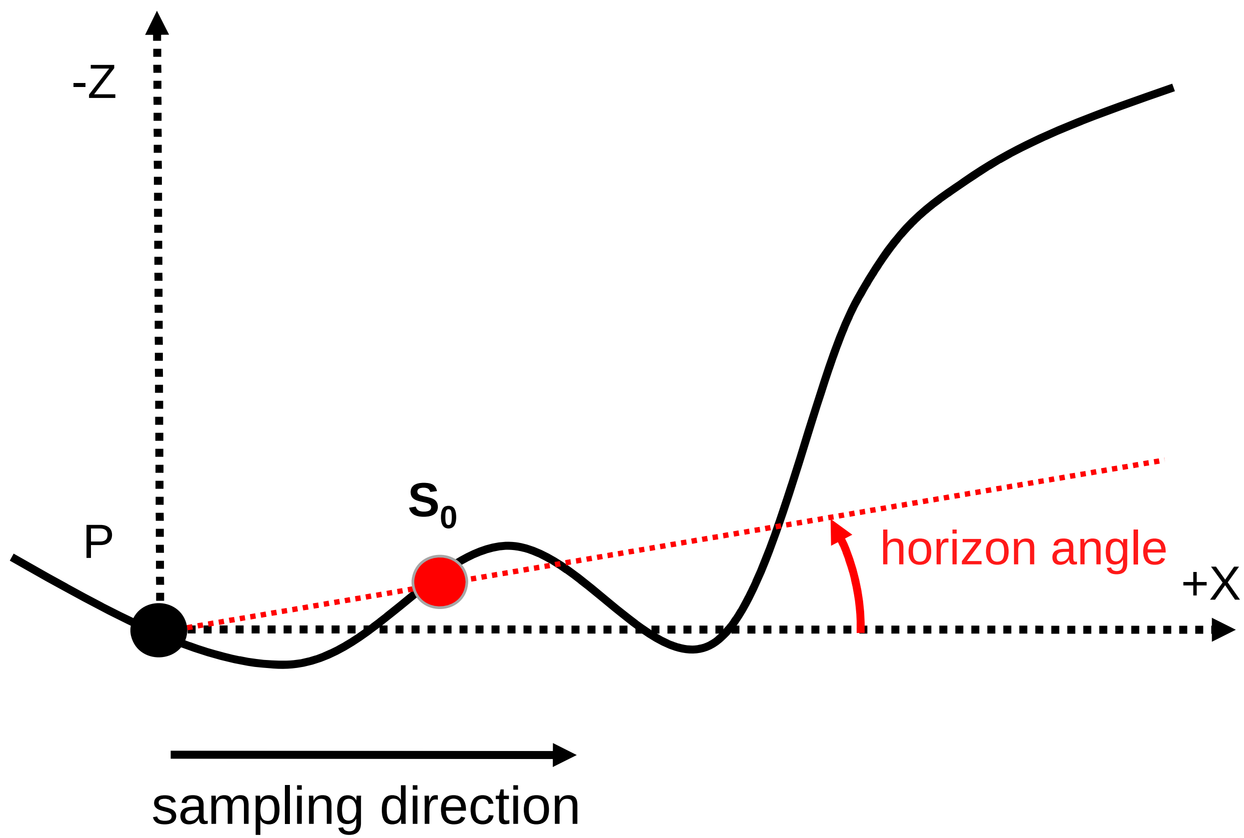
Horizon Mapping

- Given a 1D height field



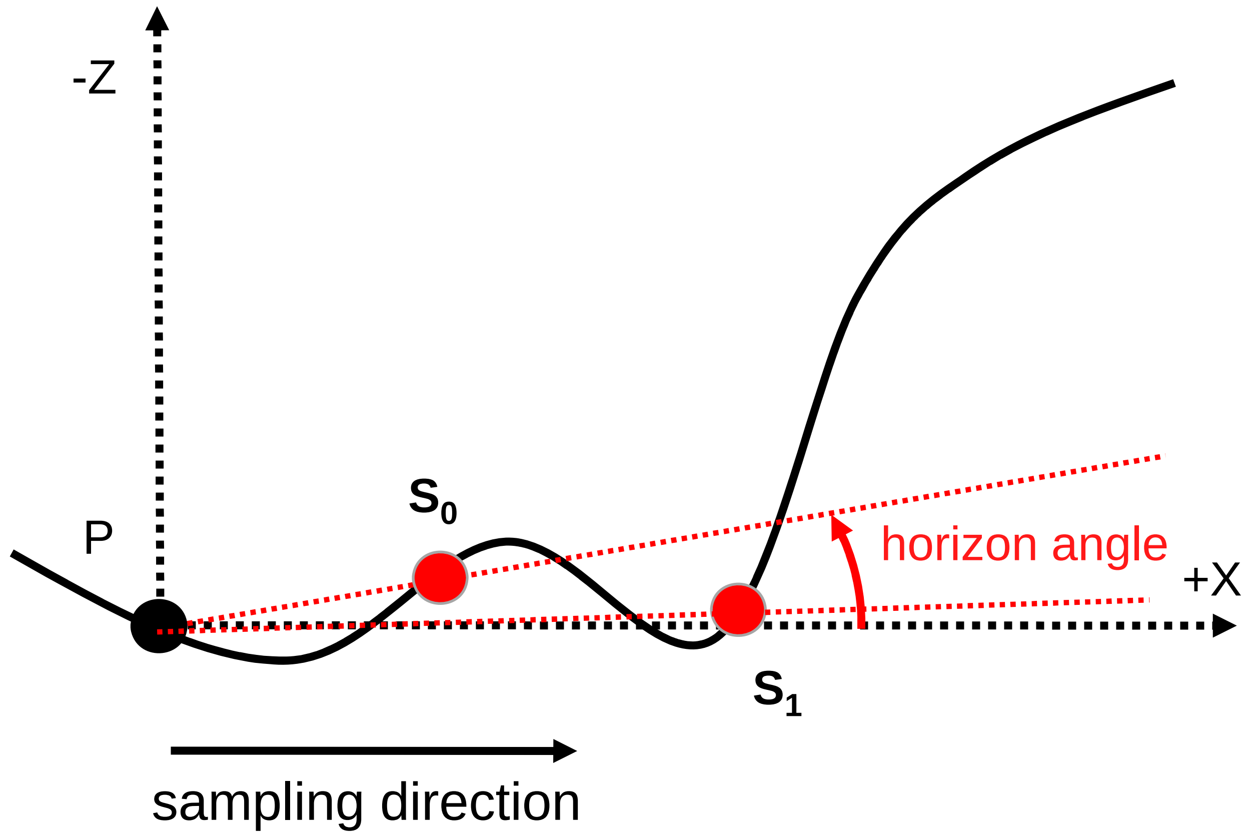
Finding the Horizon

» March along the height field

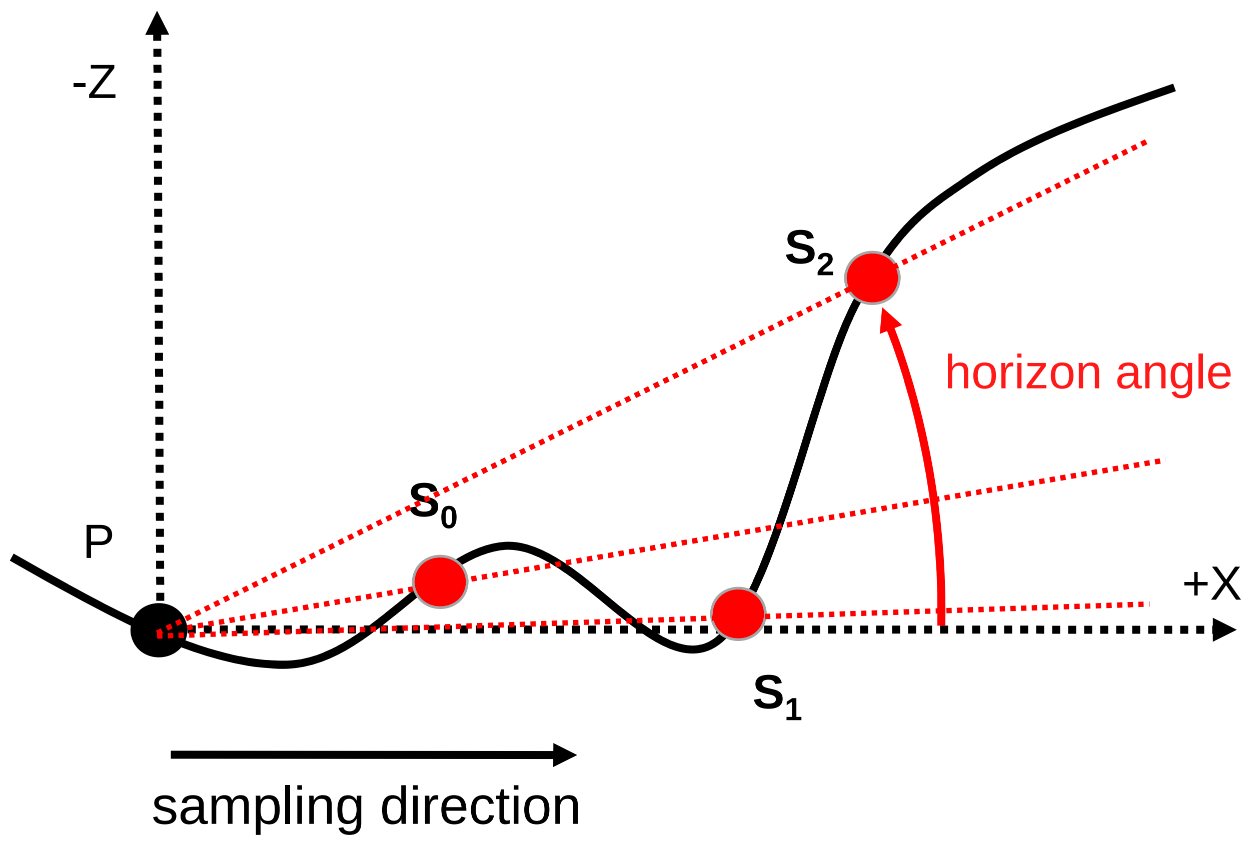


Finding the Horizon

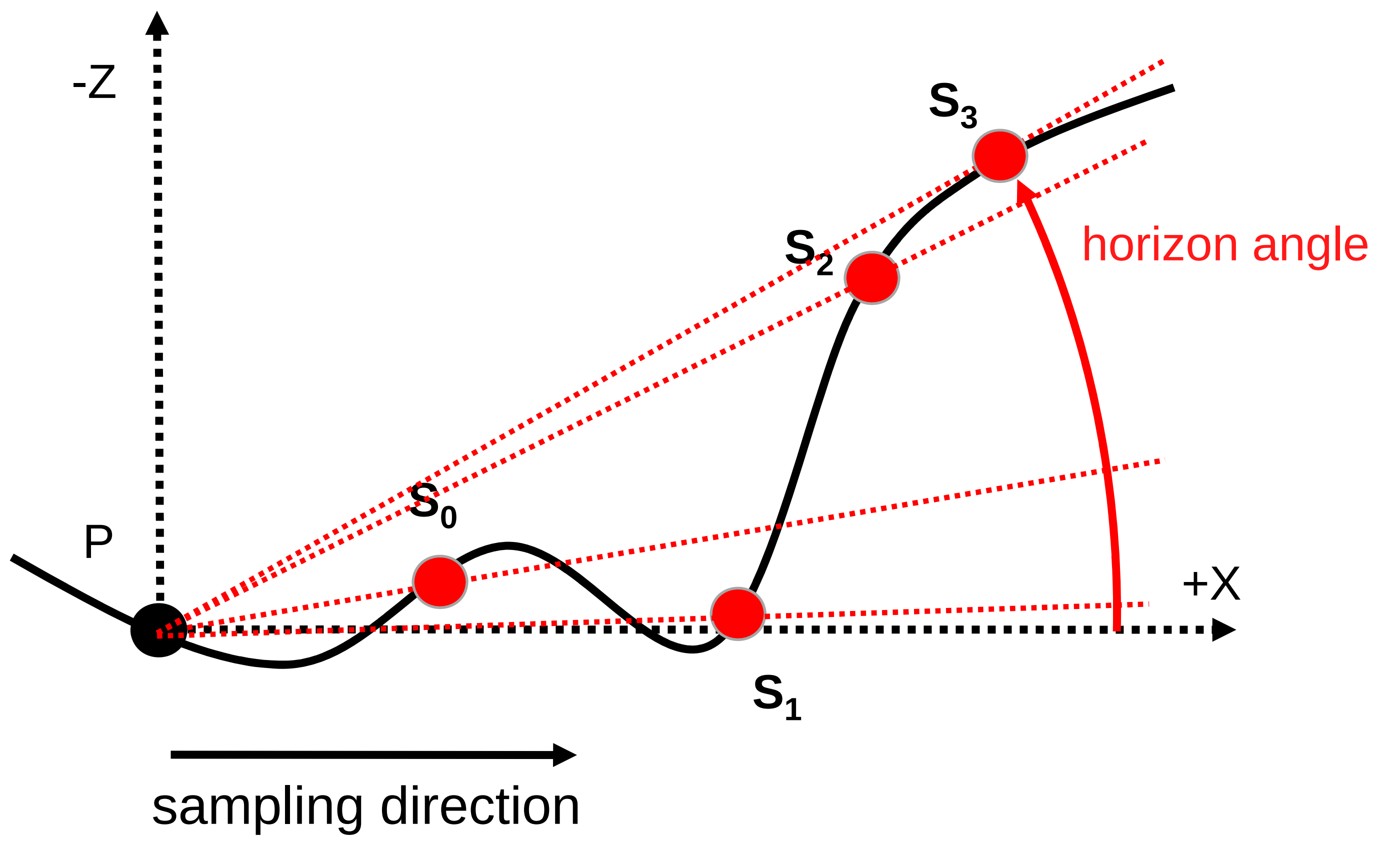
» Keeping track of maximum angle



Finding the Horizon

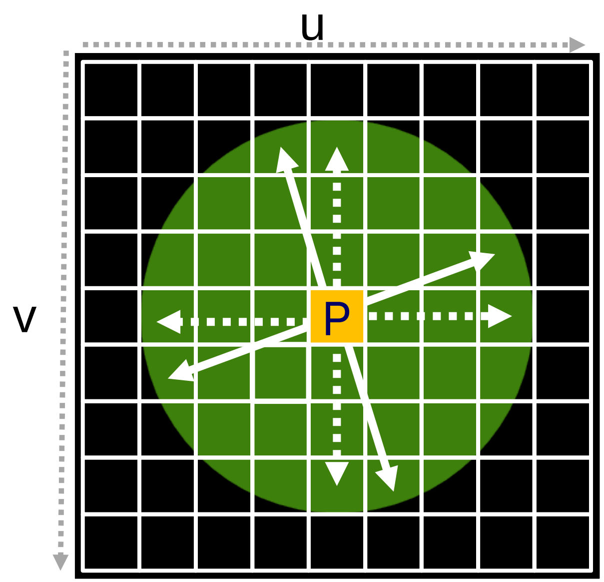


Finding the Horizon



Sampling the Depth Image

- Estimate occlusion by sampling depth image
- Use uniform distribution of directions per pixel
  - Fixed number of samples / dir
- Per-pixel randomization
  - Rotate directions by random per-pixel angle
  - Jitter samples along ray by a random offset



Noise

» Per-pixel randomization generates visible noise

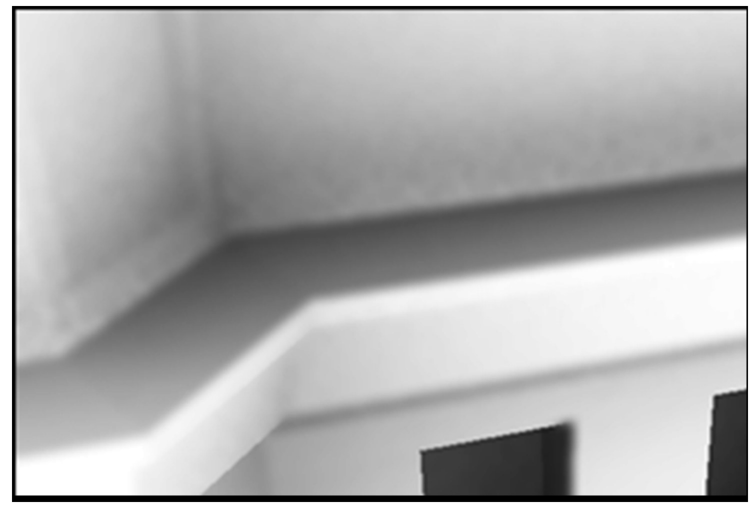
AO with 6 directions x 6 steps/dir

Cross Bilateral Filter

- Blur the ambient occlusion to remove noise
- Depth-dependent Gaussian blur
  - [Petschnigg et al. 04]
    [Eisemann and Durand 04]
  - \- Reduces blurring across edges
- Although it is a non-separable filter, we apply it separately in the X and Y directions
  - No significant artifacts visible

Cross Bilateral Filter

Without Blur



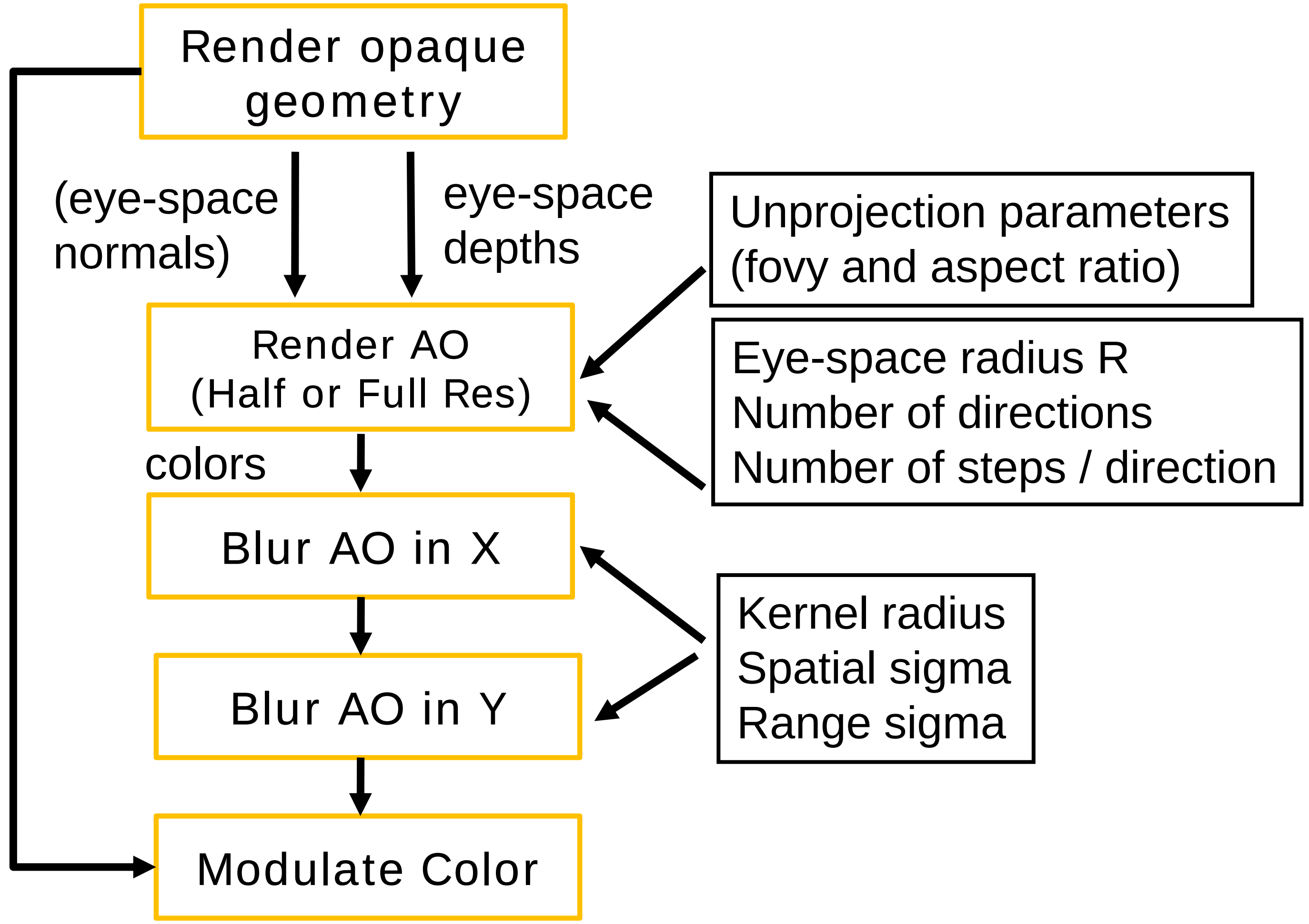
With 15x15 Blur

Half-Resolution AO

» AO is mostly low frequency
   - Can render the AO in half resolution
   - Source half-resolution depth image

» Still do the blur passes in full resolution
   - To avoid bleeding across edges
   - Source full resolution eye-space depths [Kopf et al. 07]

Rendering Pipeline



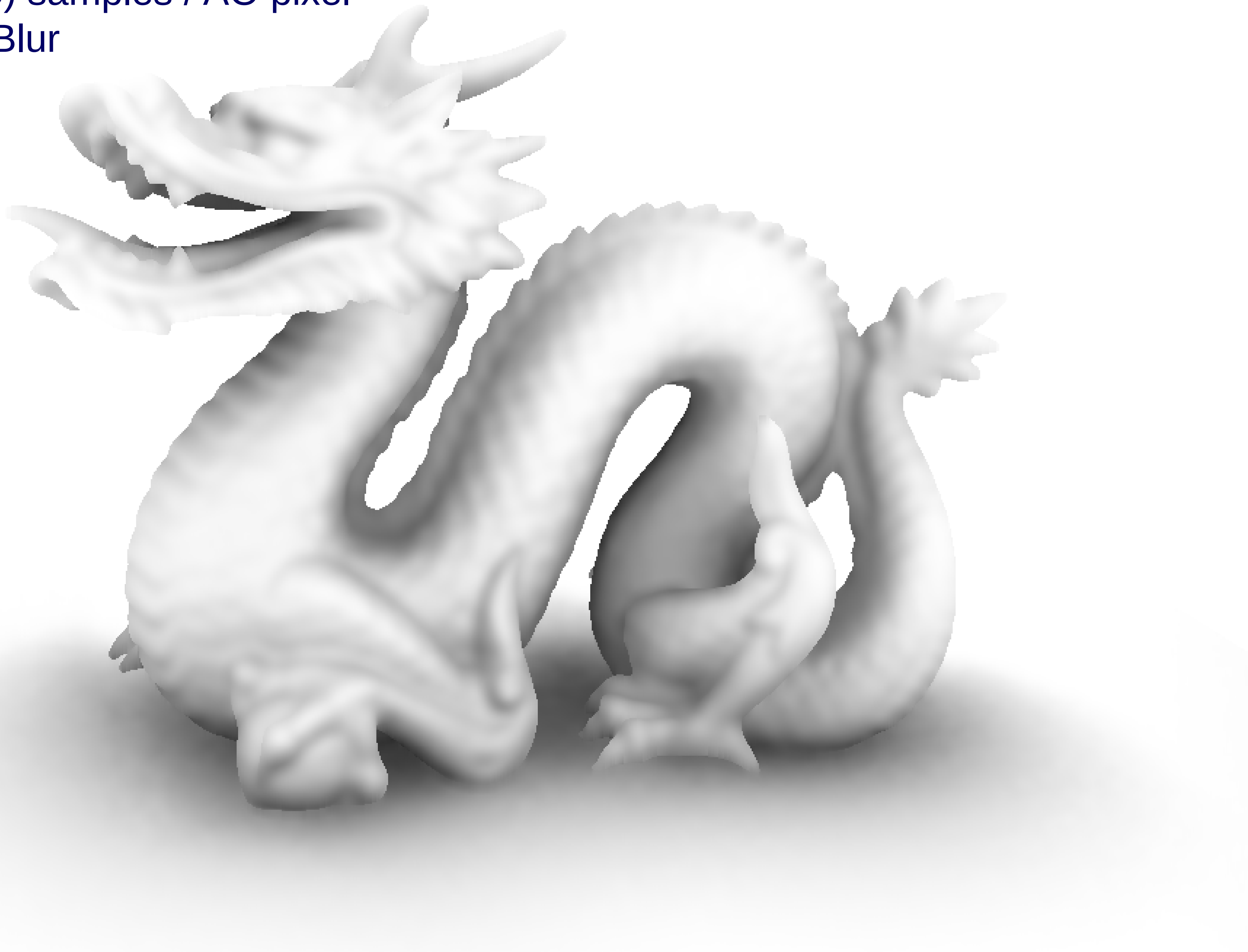
Half-Resolution AO
6x6 (36) samples / AO pixel
No Blur

Half-Resolution AO
6x6 (36) samples / AO pixel
15x15 Blur

Full-Resolution AO

6x6 (36) samples / AO pixel

15x15 Blur



Full-Resolution AO

16x16 (256) samples / pixel

No Blur

Full-Resolution AO

16x32 (512) samples / pixel

No Blur

Demo

HBAO - Conclusion

- DirectX10 SDK sample
  - Now available on developer.nvidia.com
  - Including video and whitepaper
- DirectX9 and OpenGL samples to be released soon
- Easy to integrate into a game engine
  - Rendered as a post-processing pass
  - Only requires eye-space depths (normals can be derived from depth)
- More details in ShaderX^7 (to appear)

Acknowledgments

NVIDIA

- Miguel Sainz, Louis Bavoil, Rouslan Dimitrov, Samuel Gateau, Jon Jansen

Models

- Dragon - Stanford 3D Scanning Repository
- Science-Fiction scene - Juan Carlos Silva http://www.3drender.com/challenges/index.htm
- Sibenik Cathedral - Marko Dabrovic

References

1. Volume Rendering Techniques, Milan Ikits, Joe Kniss, Aaron Lefohn, Charles Hansen. Chapter 39, section 39.5.1, *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics* (2004).
2. BAVOIL, L., AND SAINZ, M. 2009. Image-space horizon-based ambient occlusion. In ShaderX7 - Advanced Rendering Techniques.