Modern Rendering Techniques

CS425: Computer Graphics I

Fabio Miranda

https://fmiranda.me

Overview

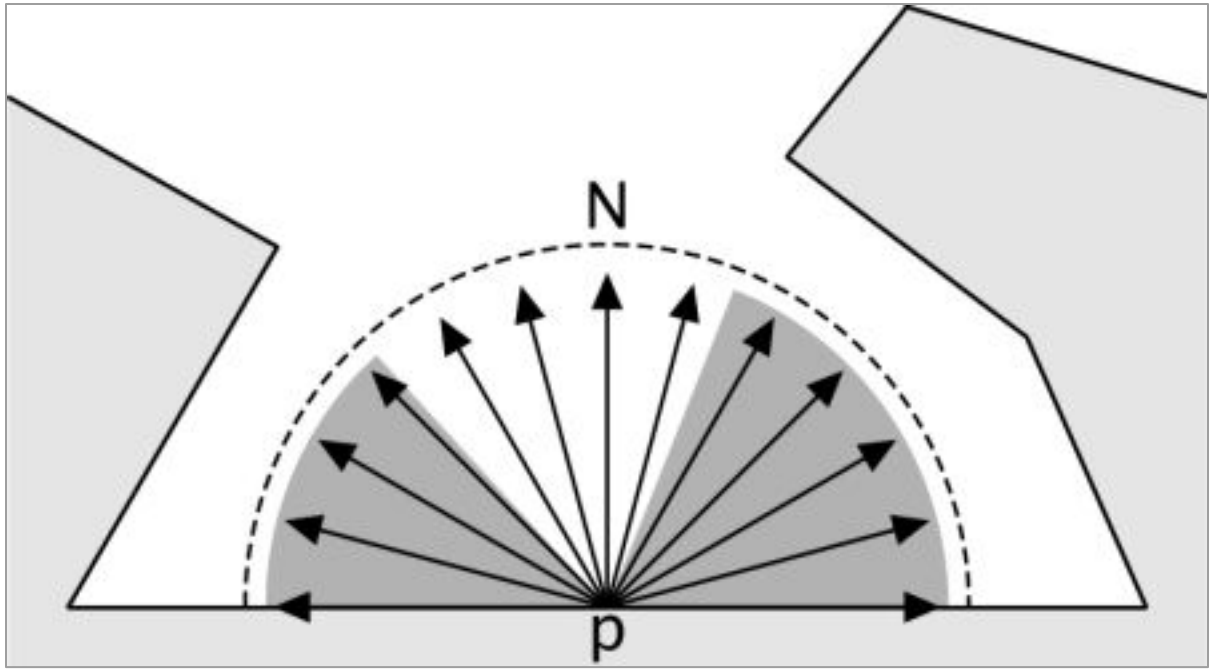
- Ambient occlusion
  - Screen-space ambient occlusion
  - Screen-space directional occlusion
- Deferred shading

Ambient occlusion

- Shading that computes how each point is exposed to ambient lighting.
- Ambient light illuminates evenly from ***all*** directions, casting soft shadows.
- Ambient occlusion: shadowing of ambient light.

Ambient occlusion

- Compute fraction of hemisphere that is occluded within distance *d* from a point.
- When shading, attenuate ambient lighting by this amount.

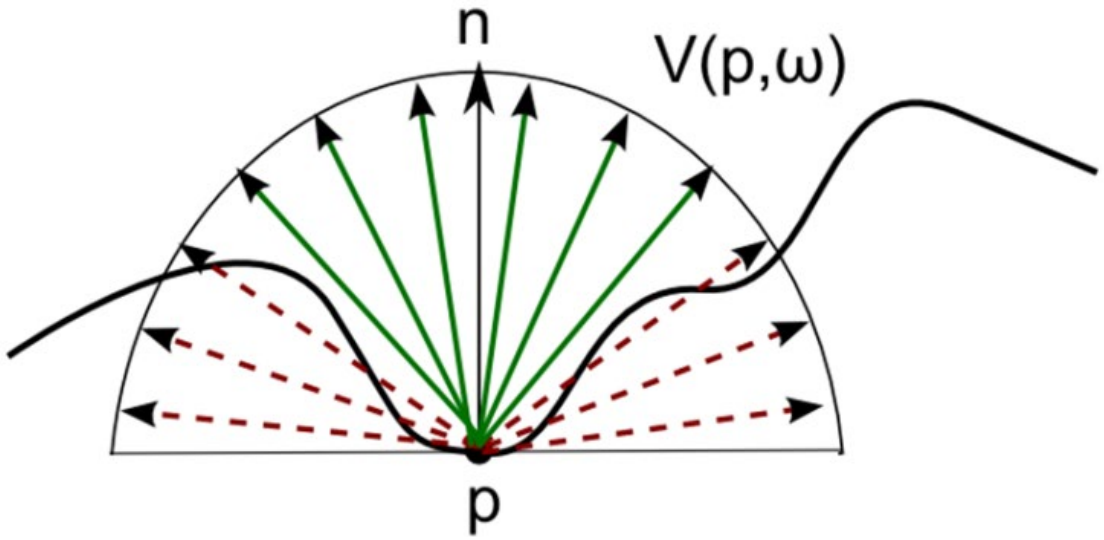


$$Ambient\ Color = k_{ambient}\mathbf{L}_{ambient}(1.0 - A_{occlusion})$$

Ambient occlusion

- How to compute the occluded fraction of the hemisphere?
- Ambient occlusion approximates the rendering equation by introducing some assumptions:

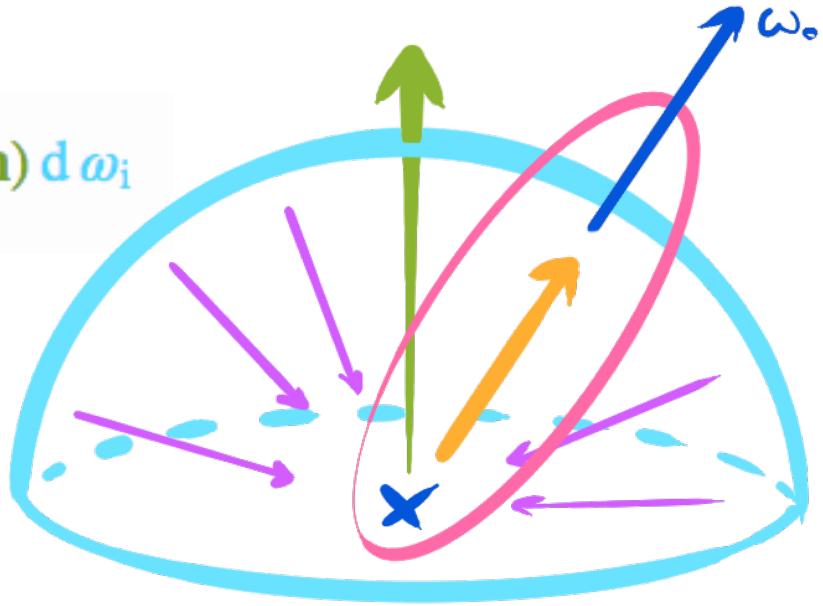
1) All surfaces around $\mathbf{p}$ are only absorbing.
2) Surface at $\mathbf{p}$ is a diffuse surface – scatters ambient light equally in all directions.
3) All light comes from an infinite uniformly white environment light (which can be occluded by geometry around $\mathbf{p}$).



Ambient occlusion

- Radiance $L(\mathbf{x}, \boldsymbol{w}_0)$ from a point $\mathbf{x}$ with normal $\mathbf{n}$ towards a direction $\boldsymbol{w}_0$:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i) (\omega_i \cdot \mathbf{n}) \, d\omega_i$$

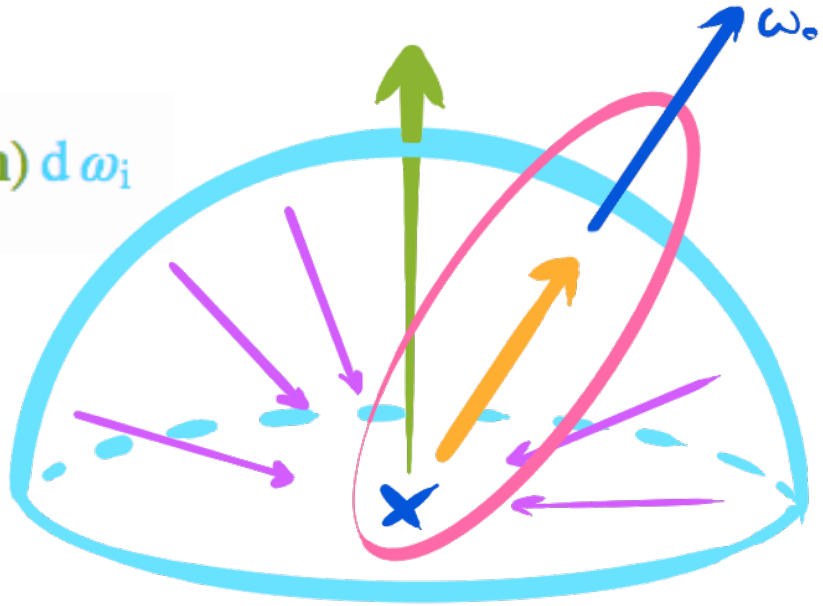


To find the light towards the viewer from a specific point, we sum the light emitted from such point plus the integral within the unit hemisphere of the light coming from a any given direction multiplied by the chances of such light rays bouncing towards the viewer (BRDF) and also by the irradiance factor over the normal at the point.

Ambient occlusion

- Radiance $L(\mathbf{x}, \boldsymbol{w}_0)$ from a point $\mathbf{x}$ with normal $\mathbf{n}$ towards a direction $\boldsymbol{w}_0$:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i) (\omega_i \cdot \mathbf{n}) \, \mathrm{d}\omega_i$$



After accounting for three assumptions, we have:

$$L_0(x, w_0) = f_r L_i \int_\Omega (\mathbf{n} \cdot \boldsymbol{\omega}) d\omega$$

$$AO(\mathbf{p}, \mathbf{n}) = \frac{1}{\pi} \int_\Omega V(\mathbf{p}, \boldsymbol{\omega}) \cdot (\mathbf{n} \cdot \boldsymbol{\omega}) d\omega$$

And why $\frac{1}{\pi}$? $\int_0^{2\pi} \int_0^{\pi/2} cos\theta sin\theta d\theta d\emptyset = \pi$ (surface area of hemisphere)

Ambient occlusion

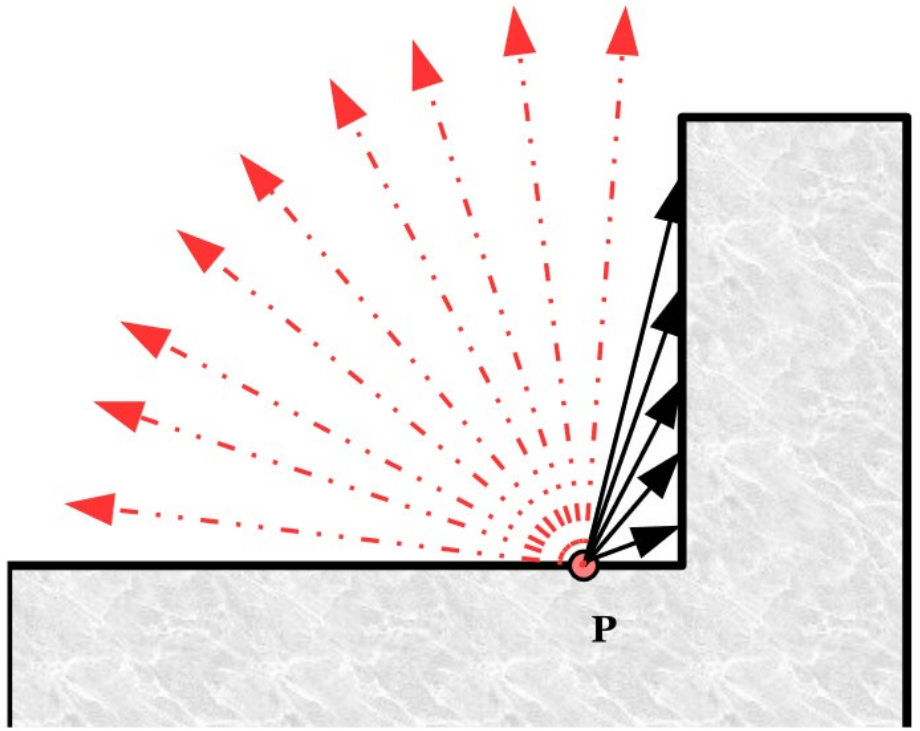
- Occlusion at a point $\mathbf{p}$:

$$AO(\mathbf{p}, \mathbf{n}) = \frac{1}{\pi} \int_{\Omega} V(\mathbf{p}, \boldsymbol{\omega}) \cdot (\mathbf{n} \cdot \boldsymbol{\omega}) d\omega$$

$V(\mathbf{p}, \boldsymbol{\omega})$: visibility function (0 or 1)

Ω: hemisphere at point $\mathbf{p}$

- Visible fraction: $1 - AO$
- The ambient component will be modulated by $1 - AO$, approximating effects that are otherwise only achieved by full global illumination.

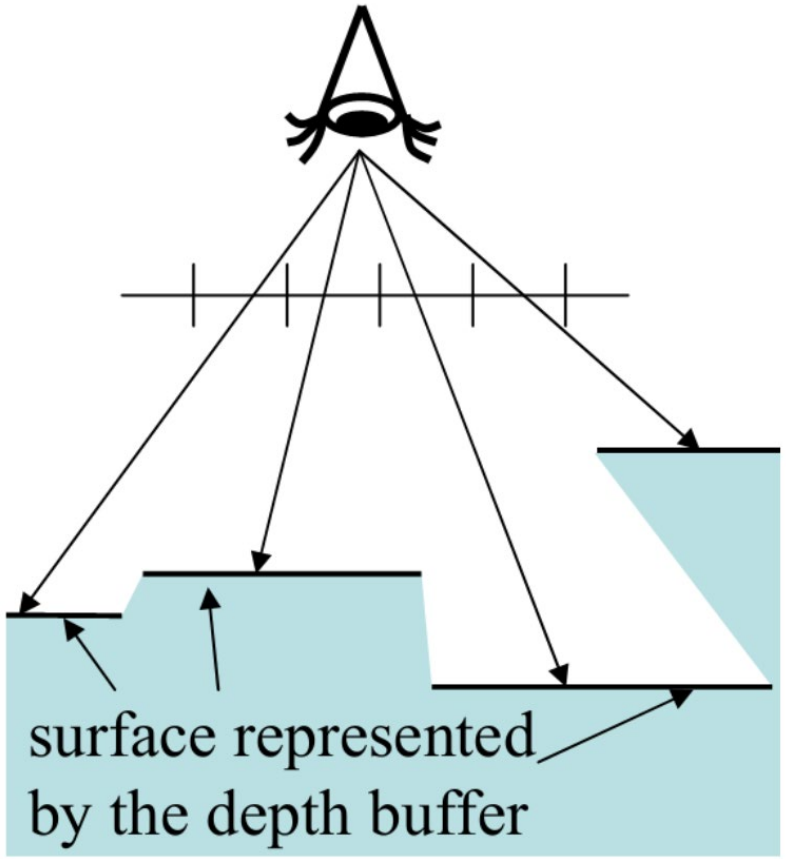


Computing ambient occlusion

- Different approaches:
  - Ray tracing
  - Screen-space ambient occlusion
  - Screen-space directional occlusion
  - Voxel accelerated ambient occlusion
  - …

Screen-space ambient occlusion

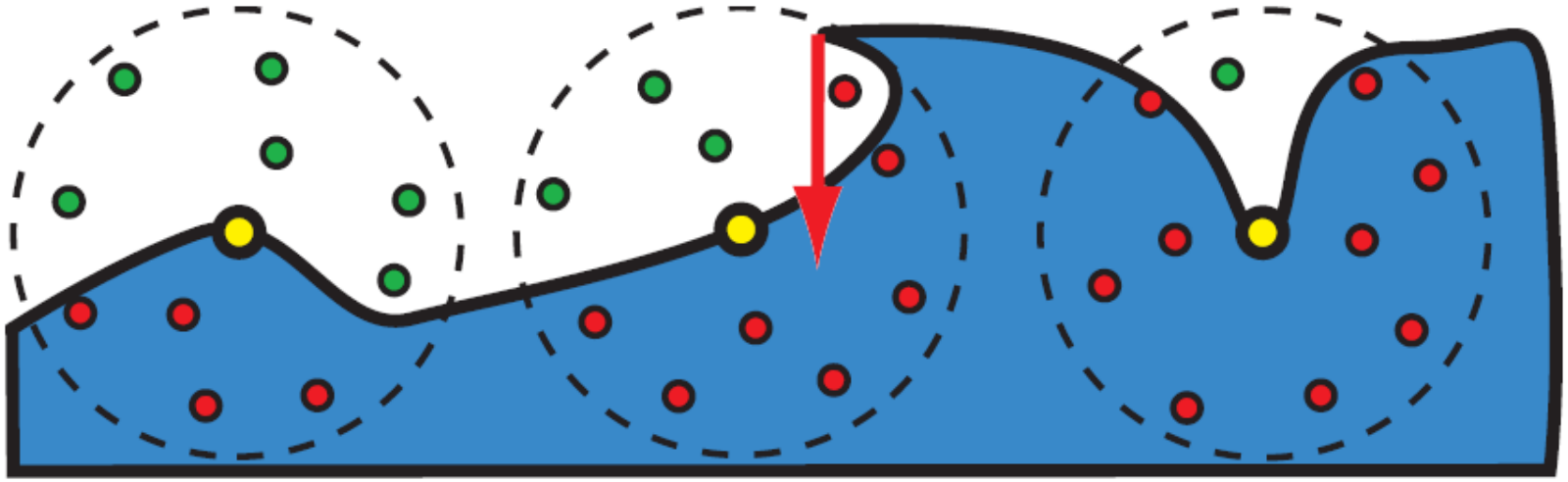
- Simply tracing rays for each point is expensive.
  - Dependent on scene complexity.
- How to reduce the amount of work?
  - Screen-space methods: independent of scene complexity and can use information such as screen-space depth and normal.



[Szirmay-Kalos et al., 2010]

Screen-space ambient occlusion

- Crytek's screen-space dynamic ambient occlusion:
  - Ambient occlusion factor k_a of each pixel is estimated by z-testing a set of points distributed in a sphere around the pixel's location.
    - $k_a = 1$ if more than half of samples pass z-test.

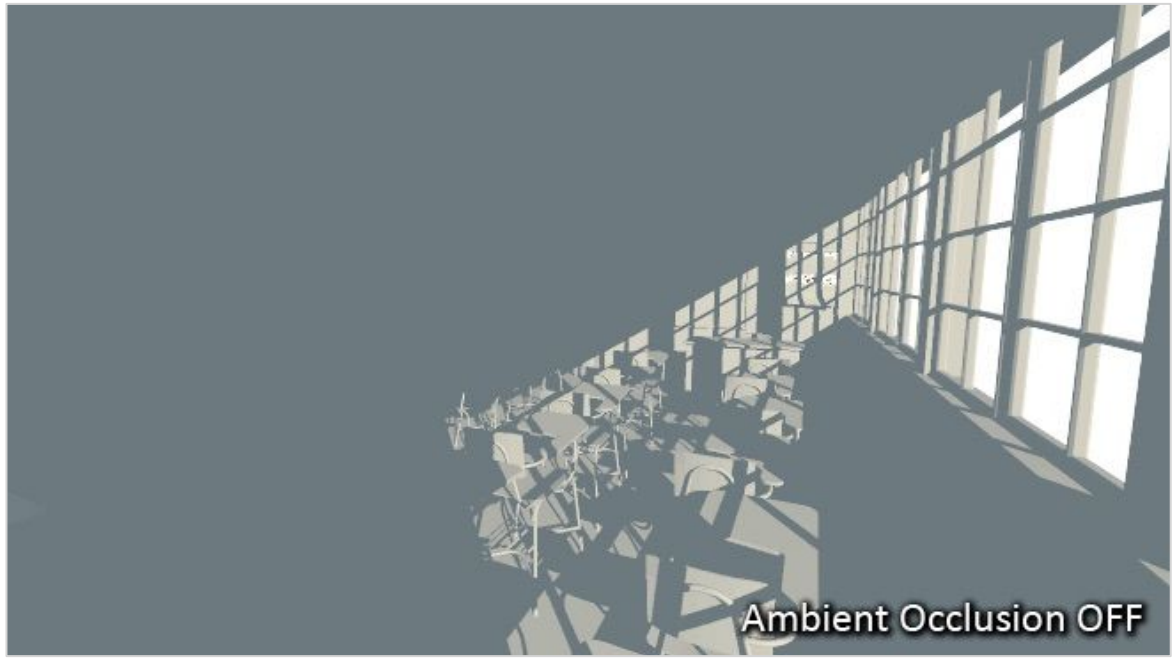


Screen-space ambient occlusion

- For every fragment processed by the fragment shader:
  - Sample depth values around the current pixel, computing amount of occlusion from each sampled point.
- We might need up to 200 samples to get good occlusion results. How to reduce that?
  - Random kernel rotations.

```
vec3 pos = texture(posmap, texcoord).xyz;
float ao = 0.0;

for (int i = 0 ; i < MAX_KERNEL_SIZE ; i++) {
    vec3 samplePos = pos + gKernel[i];
    vec4 offset = vec4(samplePos, 1.0);
    offset = gProj * offset;
    offset.xy /= offset.w;
    offset.xy = offset.xy * 0.5 + vec2(0.5);

    float depth = texture(posmap, offset.xy).b;

    if (abs(pos.z - depth) < gSampleRad) {
        ao += step(depth,samplePos.z);
    }
}

outColor = vec4(pow(1.0 - ao/128.0, 2.0));
```

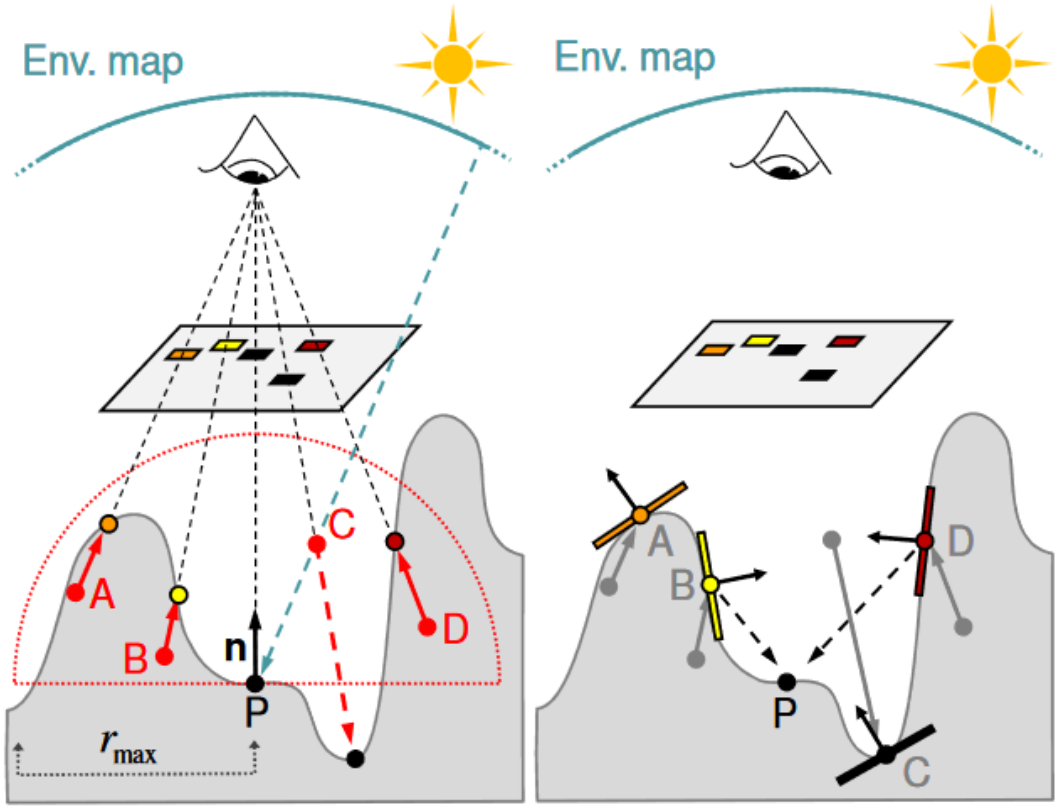
Screen-space ambient occlusion



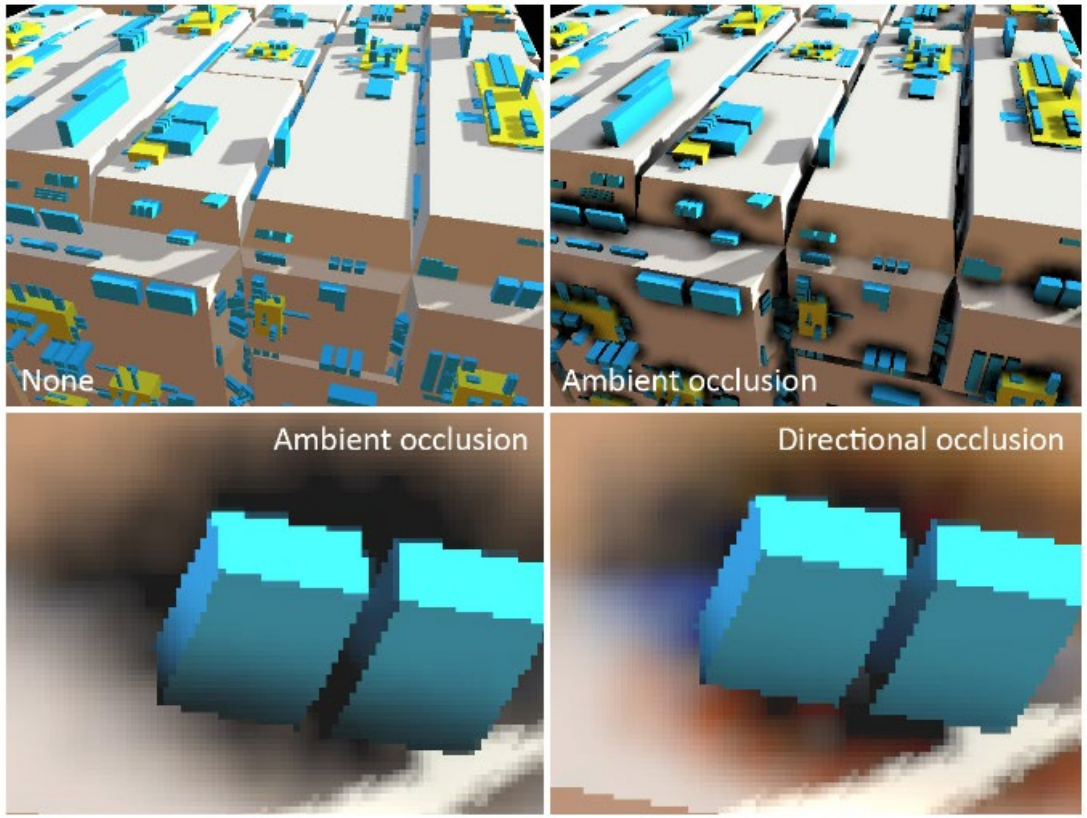
[aaa-studio.eu]

Screen-space directional occlusion

- Improves SSAO taking direction into account when sampling ambient light.
- Combines calculation of irradiance and occlusion at the same time.
- Steps:
  - Sample depth values around the current pixel, computing amount of occlusion from each sampled point.
  - Samples that are above occluded surface allow radiance.



Screen-space directional occlusion



[Ritschel et al., 2009]

Deferred shading

- Main idea: decouple geometry and illumination processing.
- Geometry processing:
  - Result stored in **geometry buffers**.
  - Proportional to scene complexity (number of primitives).
- Illumination processing (shading):
  - Render simple quad.
  - Proportional to image resolution

Deferred shading

- Blinn-Phong model:

$$f_{Phong}(\mathbf{L}_{light}, \mathbf{l}, \mathbf{v}, \mathbf{n}) = k_{ambient}\mathbf{L}_{ambient} + \sum_{m \in lights} k_{diffuse}(\hat{\mathbf{l}}_{\boldsymbol{m}} \cdot \hat{\mathbf{n}})\mathbf{L}_{m,diffuse} + k_{specular}(\hat{\boldsymbol{h}}_{\boldsymbol{m}} \cdot \hat{\boldsymbol{n}})^{\alpha}\mathbf{L}_{m,specular}$$

- What information do we need to store in the geometry buffer?
  - Geometry
    - Position
    - Normal
  - Material:
    - Ambient, diffuse, specular components
  - Uniforms:
    - Light sources
  - Viewer:
    - Position

Geometry buffer

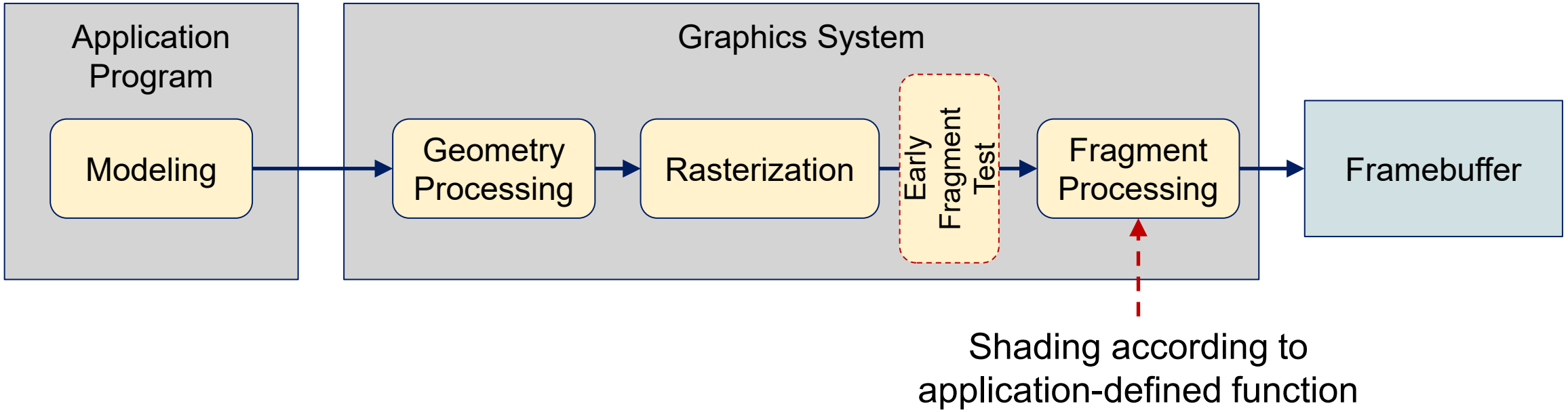
- Result of the geometry processing step.
- Render to four RGBA output buffers + depth:
  - Geometry:
    - Normal (x,y,z)
    - Depth
  - Material:
    - Albedo (reflectance) (rgb), diffuse (rgb), specular (rgb) with shininess factor
  - Position:
    - Light sources
    - Viewer

Deferred shading

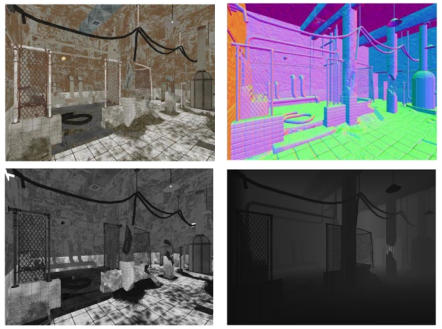
- Why deferred shading?

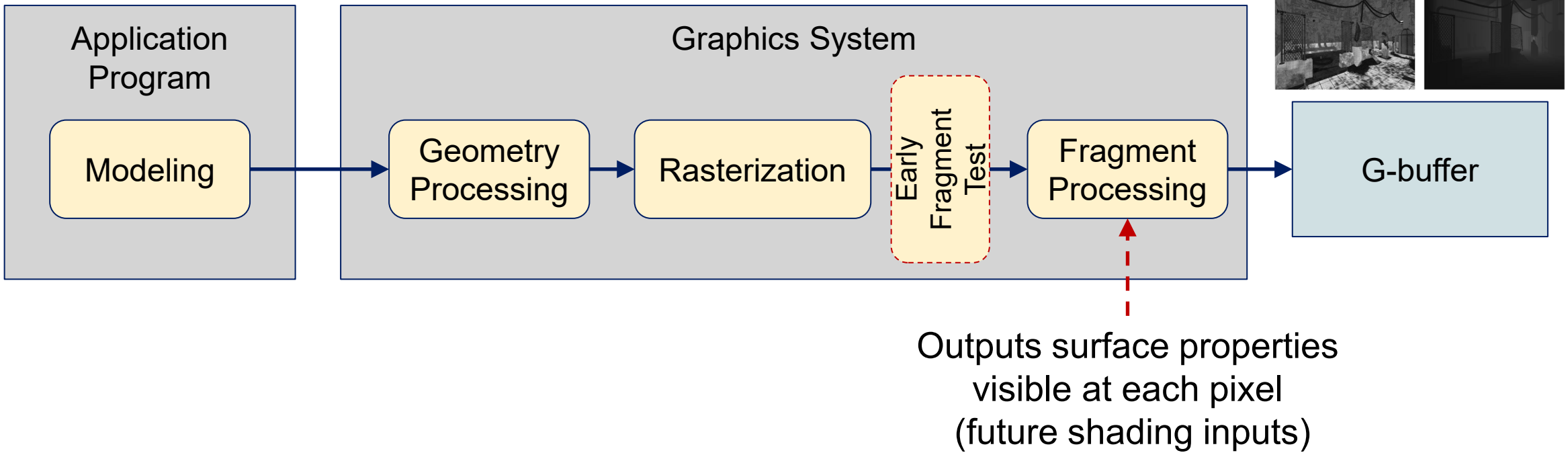
1. Shading is expensive: deferred shading ensures that only visible fragments are shaded.
   - Regardless of the number of triangles, exactly one shade per screen output.
2. Rendering shades small triangles inefficiently.

Graphics pipeline: forward rendering



Graphics pipeline: deferred shading





Two-pass deferred shading

```
for object in render
    for light in scene
        render(object, light)
```

Single pass:

- Process and shade all objects to framebuffer

```
for object in scene
    renderGBuffer(object)

for light in scene
    renderShading(light)
```

Two pass:

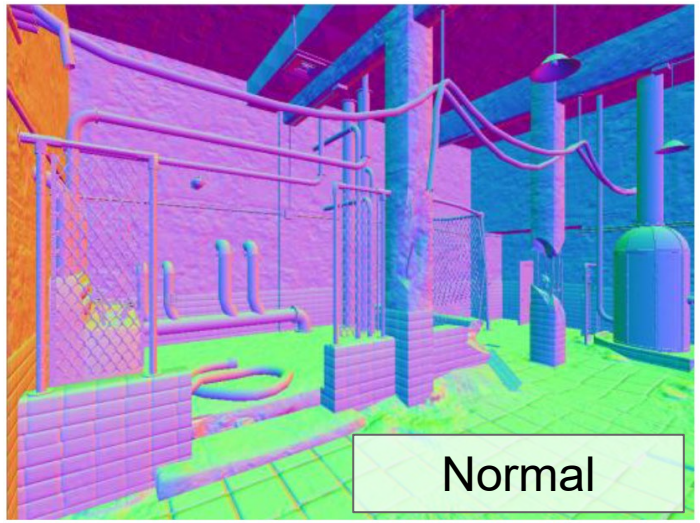
- Create g-buffer
- Shade all samples in the g-buffer

Two-pass deferred shading

- Pass 1: generate g-buffer
  - Render scene using traditional graphics pipeline.
  - Visible geometry goes to g-buffer.
- Pass 2: shading pass
  - For each fragment:
    - Restore g-buffer data for current sample.
    - Accumulate contribution from all lights.
    - Output final surface color for sample.

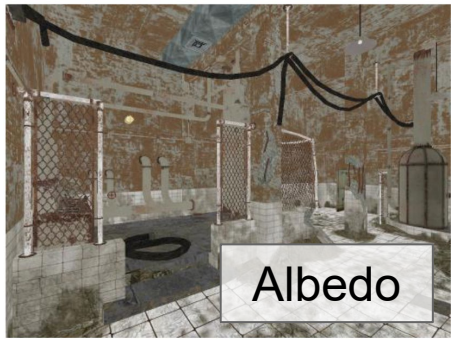
Shading is “deferred” until all geometry is complete

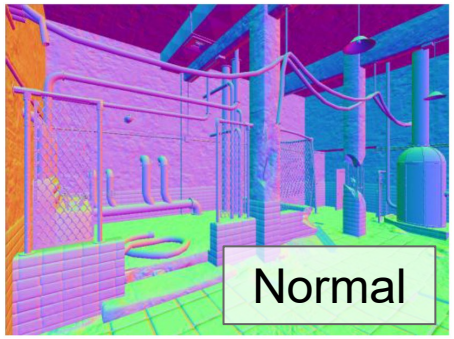
Geometry buffer

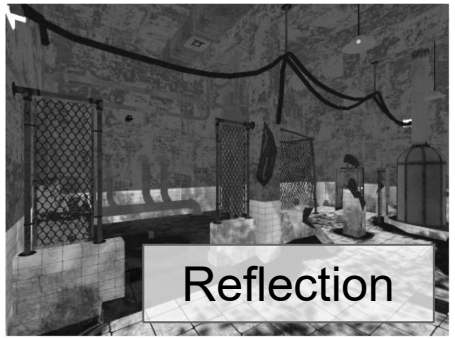


[Josh Klint, 2008]

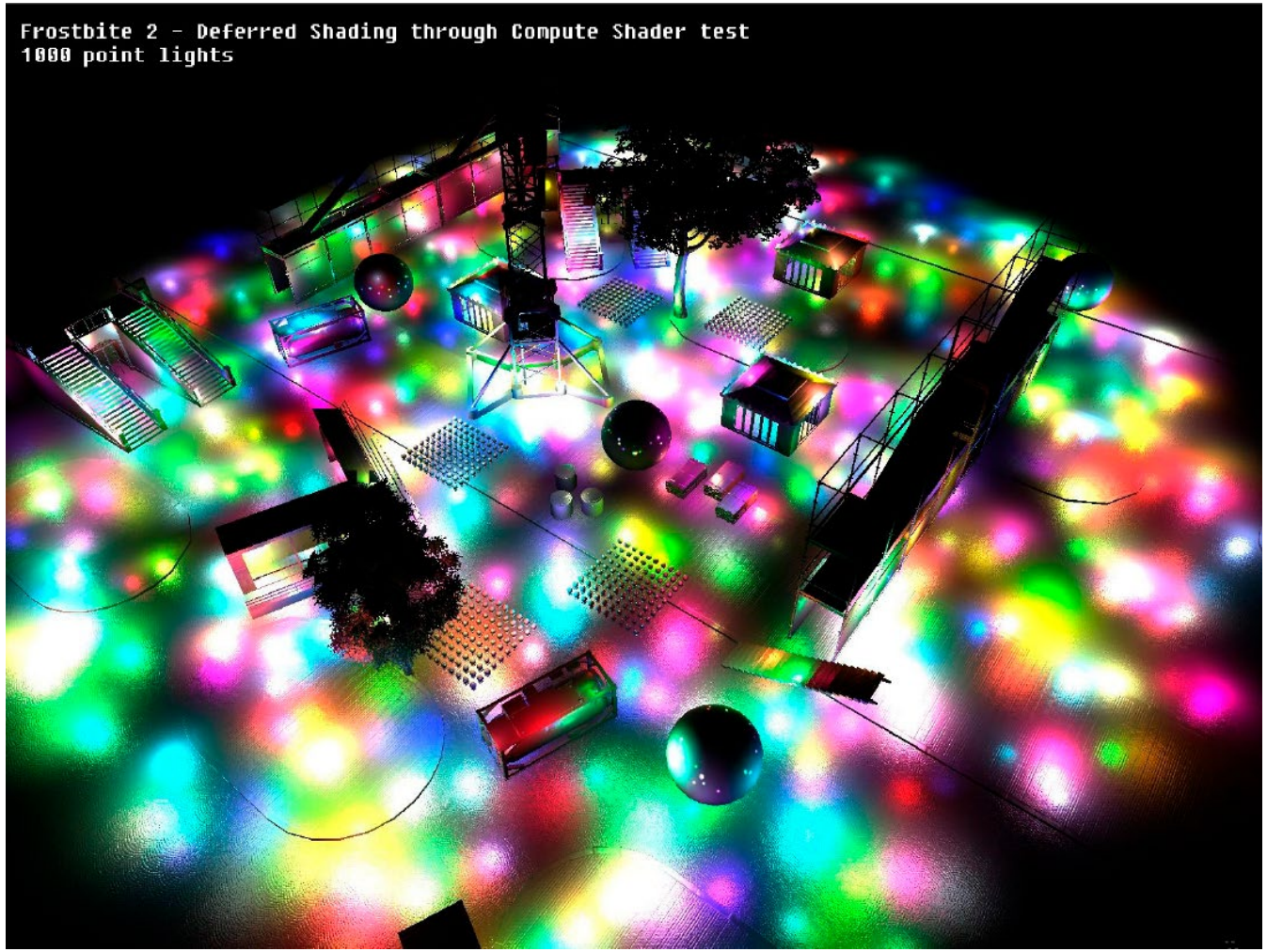
Two-pass deferred shading







Two-pass deferred shading



Limitations

- Supersampling in deferred shading is challenging.
- Large amount of information that must be stored per pixel:
  - 3840 x 216 (4K display) must store a g-buffer with almost 700MB.
- Since we are storing geometry information on a g-buffer, transparency is also a challenge.
  - Each pixel has the information for one single object (and transparency is about combining two or more objects to a single object).

Deferred sharing

- Popular technique in modern games.
- Creates a g-buffer, not a final image.
- Convenience and simplicity separating geometry processing and shading.
- High performance under complex conditions:
  - Large number of light sources.
  - Complex scenes.

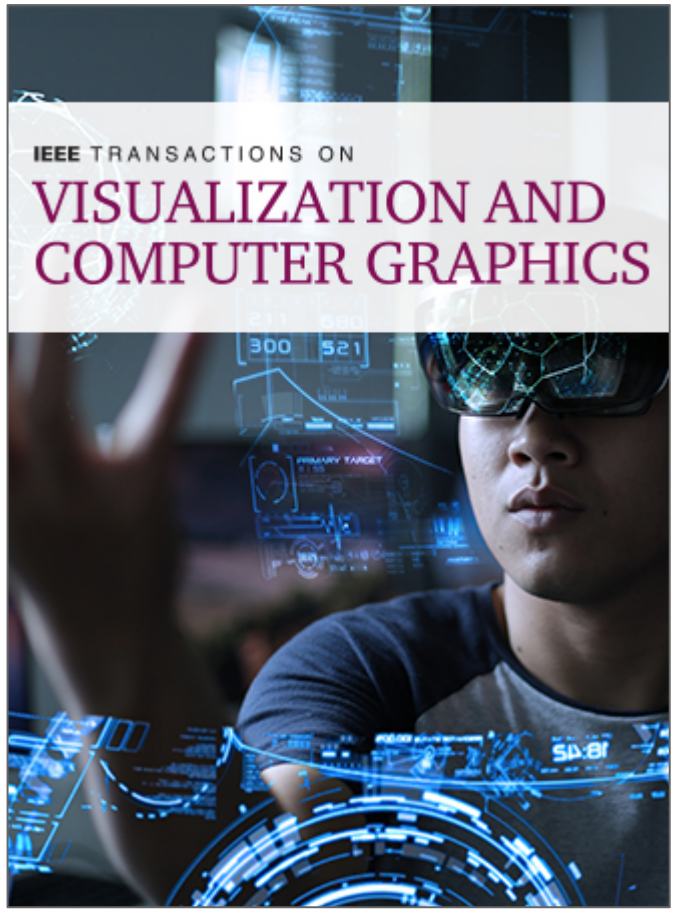
What's next?

- Get involved in research!
- IEEE Transactions on Visualization and Computer Graphics:

https://www.computer.org/csdl/journal/tg

- ACM Transactions on Graphics:

https://dl.acm.org/journal/tog



How to get involved?

- E-mail professors ask them about independent study.
- VIS/VR/HCI/Graphics: https://evl.uic.edu/
- Why research (or independent study), according to Kayvon Fatahalian, Stanford:
  - You will learn more about a topic than in class.
  - It's more fun to be on the cutting edge. Industry might not even know about what you are working on.
  - It widens your mind as to what is possible.

Grad school

- Great resource:

http://www.cs.cmu.edu/~harchol/gradschooltalk.pdf

Which letter do you think is stronger? It turns out that Letter 2 is very strong. Letter 1 actually counts as 0. At CMU we mark all letters like letter 1 with the acronym **D.W.I.C.**. This stands for "Did Well In Class" which counts for 0, since we already know from the student's transcript that he did well in class. By contrast, student Y's letter gives us a lot of information. It explains that the reason student Y didn't do better in class was that he was busy doing research. It also tells us that student Y started doing research on his own initiative, and that he is quite good at doing research. The professor was impressed enough with student Y's ideas that he took him on as a student researcher despite student Y not having high grades.

Applying to Ph.D. Programs in Computer Science

Mor Harchol-Balter
Computer Science Department
Carnegie Mellon University

Last updated 2014

1 Introduction

This document is intended for people applying to Ph.D. programs in computer science or related areas. The document is informal in nature and is meant to express only the opinions of the author. The author is a professor of computer science at CMU, and has been involved in the Ph.D. admissions process at CMU, U.C. Berkeley, and MIT.

Please direct any further questions you have after reading this document to our Admissions Coordinator (applyweb@cs.cmu.edu). Do not send email to the author of this document.

Contents

1