



Ambient Occlusion

CS535

Daniel G. Aliaga

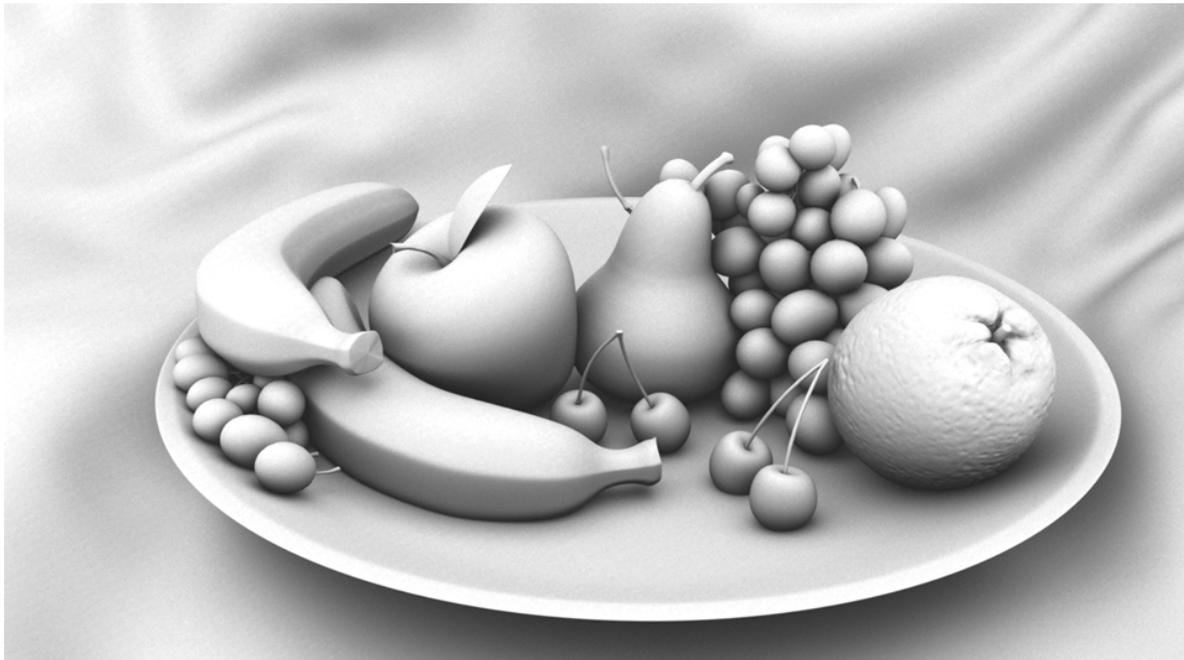
Department of Computer Science

Purdue University



Ambient Occlusion

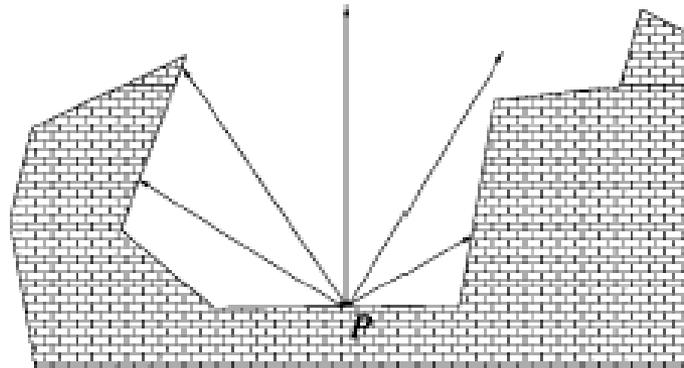
- It is a lighting technique to increase the realism of a 3D scene by a “cheap” imitation of global illumination of global illumination





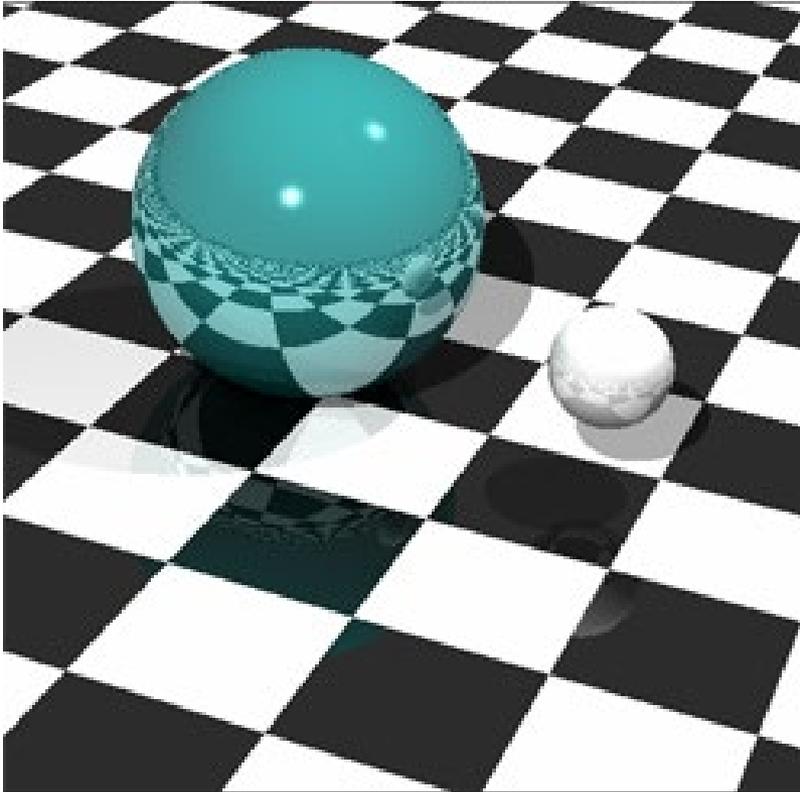
History

- In 1998, Zhukov introduced *obscurances* in the paper “An Ambient Light Illumination Model.”
- The effect of obscurances : we just need to evaluate the *hiddenness* or occlusion of the point by considering the objects around it.





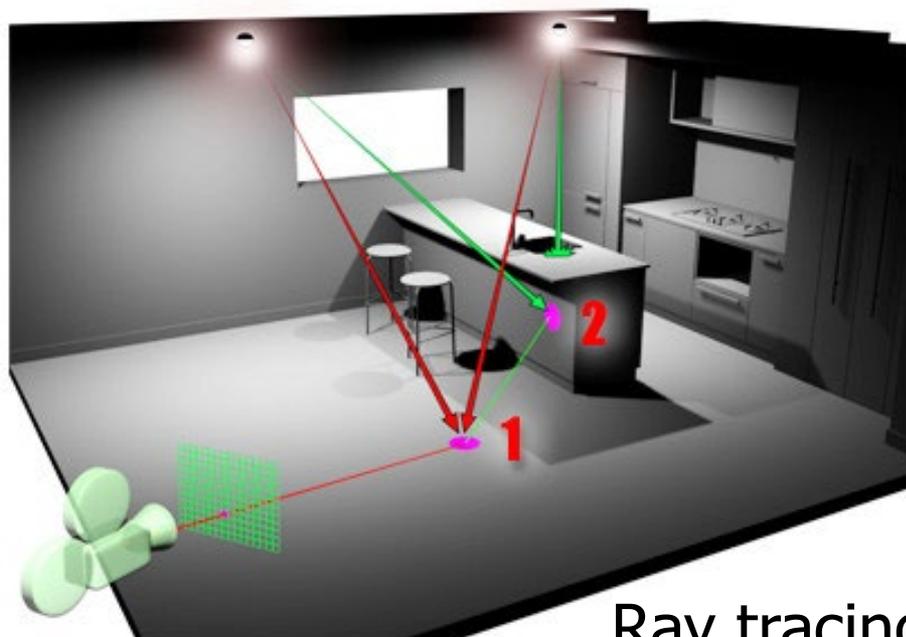
Global Illumination



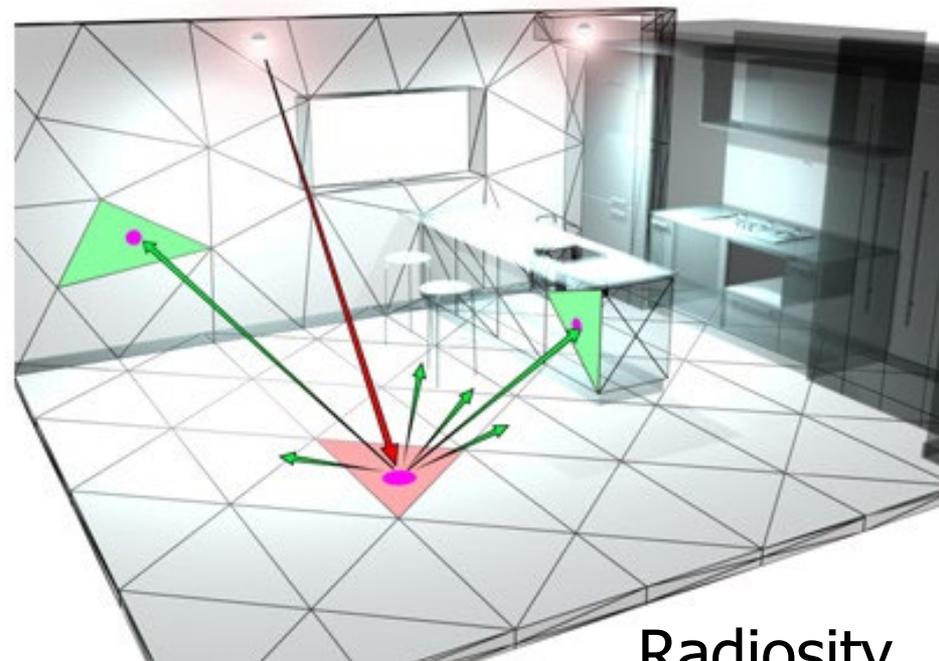
Ray tracing



Radiosity



Ray tracing



Radiosity



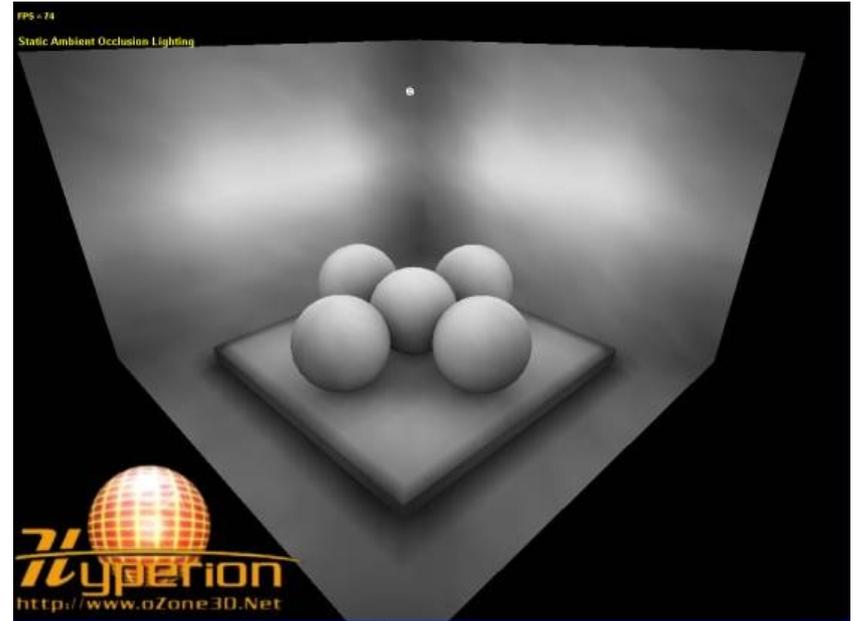
Phong Illumination Model

$$I = I_a + I_d + I_s$$

$$I_a = IA \cdot occ(v)$$



Constant ambient intensity rendering

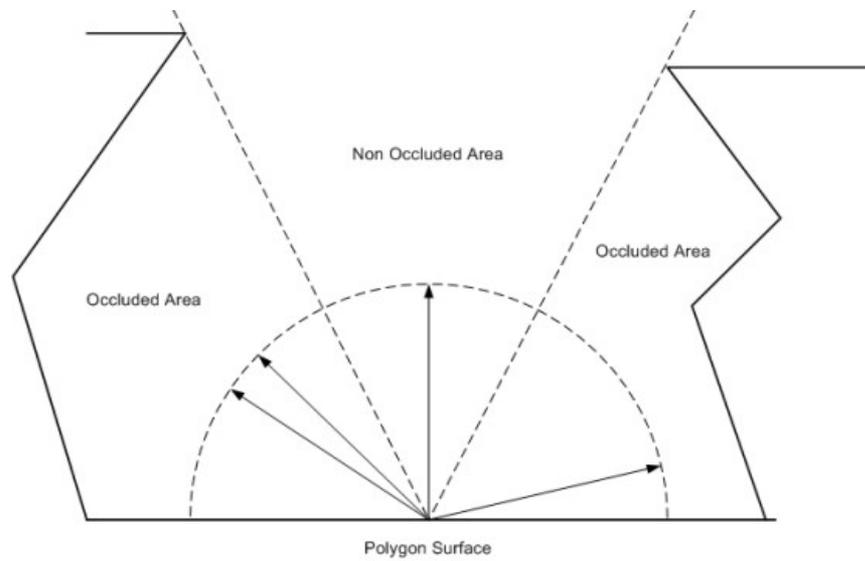


Modulate the intensity by an occlusion factor



Occlusion Factor/Map

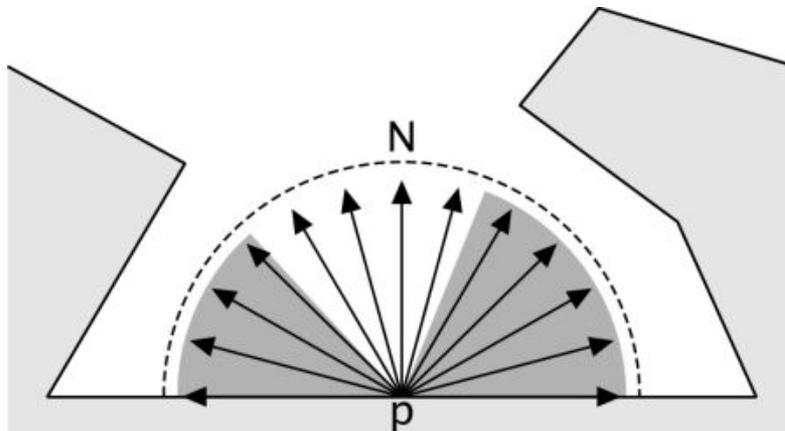
- Shooting rays outwards
- Determine the occlusion factor as a percentage



Inside-Looking-Out Approach: Ray Casting



- Cast rays from p in uniform pattern across the hemisphere.
- Each surface point is shaded by a ratio of ray intersections to number of original samples.
- Subtracting this ratio from 1 gives us dark areas in the occluded portions of the surface.

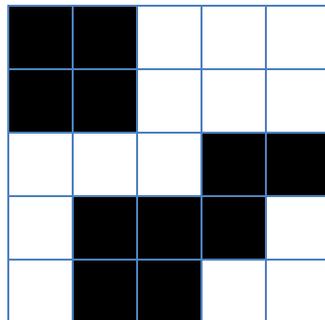


e.g.: Cast 13 rays
9 intersections
⇒ Color * 4/13

Inside-Looking-Out Approach: Hardware Rendering



- Render the view from p toward the normal N
- Rasterize black geometry against a white background.
- Take the (cosine-weighted) average of rasterized fragments.



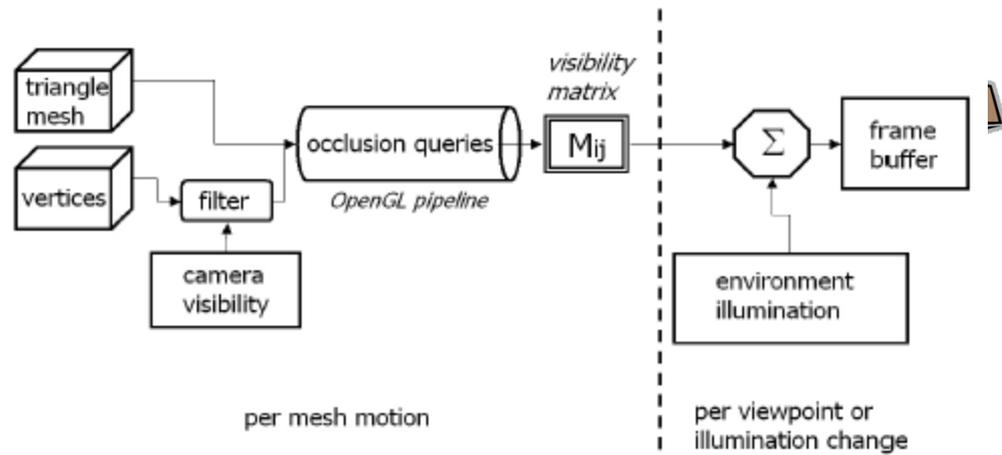
11 black fragments
⇒ Color * 14/25



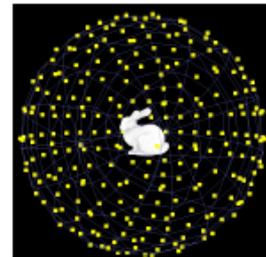
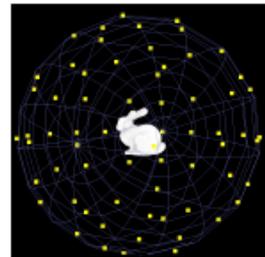
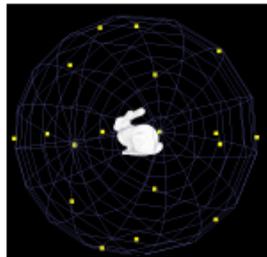
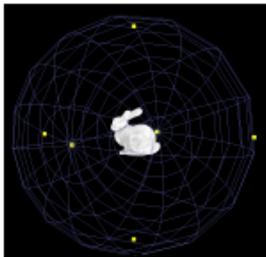
Comments

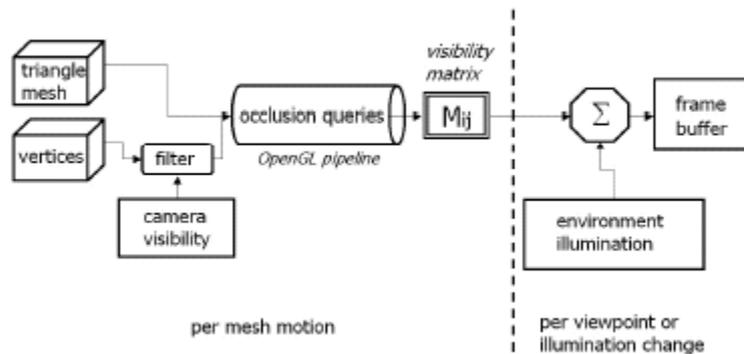
- Huge pre-computation time per scene (20min)
- Store occlusion factor as vertex attributes
- Variations on sampling method
- “Inside-out” algorithm
- “outside-in” alternative

Outside-In Alternative [Sattler et. al 2004]



$$c_i = \sum_{j=1}^k M_{ij} I_j$$





```
enable orthographic projection
disable framebuffer
for all light directions  $j$  do
  set camera at light direction  $l_j$ 
  render object into depth buffer with polygon offset
  for all vertices  $i$  do
    begin query  $i$ 
    render vertex  $i$ 
    end query  $i$ 
  end for
  for all vertices  $i$  do
    retrieve result from query  $i$ 
    if result is "visible" then
       $M_{ij} = \mathbf{n}_i \cdot \mathbf{l}_j$ 
    end if
  end for
end for
```

$$M_{ij} = \begin{cases} \mathbf{n}_i \cdot \mathbf{l}_j & : \text{ vertex visible} \\ 0 & : \text{ vertex invisible} \end{cases}$$

$$c_i = \sum_{j=1}^k M_{ij} I_j$$



Sattler et al.

- For each light on the light sphere
- Take the depth map (for occlusion query)
- Use occlusion query to determine the visibility matrix



Image-Based AO

- SHANMUGAM, P., AND ARIKAN, O. 2007. Hardware Accelerated Ambient Occlusion Techniques on GPUs. In Proceedings of ACM Symposium in Interactive 3D Graphics and Games, ACM.

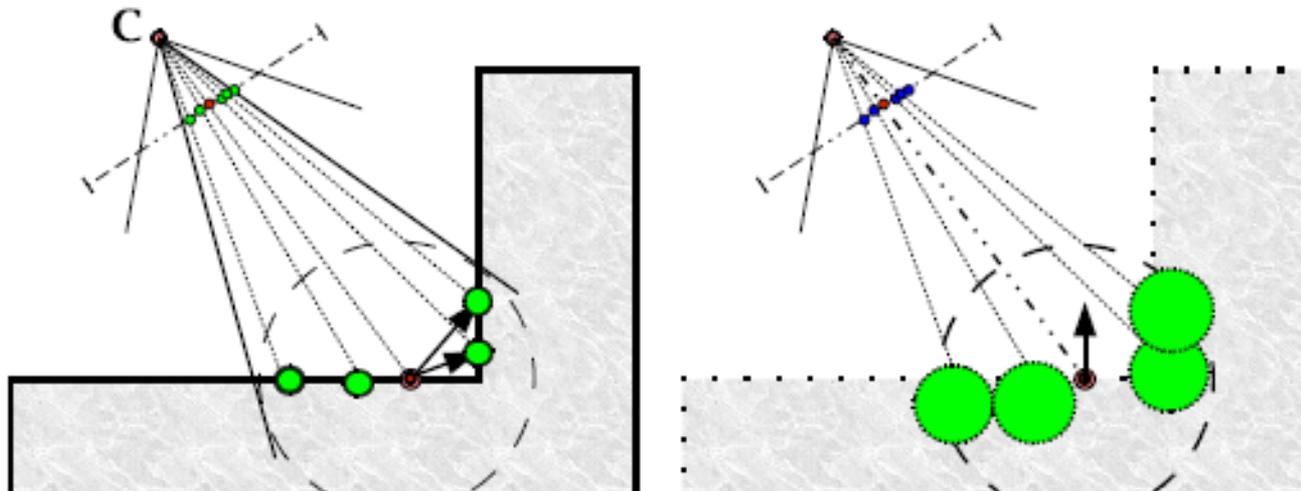




Image-Based AO

