In cinema and photography, shadows are important for setting mood and directing attention.

Shadows indicate spatial relationships between objects e.g. contact with floor.

Without shadow

With shadow

Two types of shadow:
- "attached" \( n \cdot l < 0 \)
- "cast"

Blinn-Phong Model with Shadows

Let the function \( S(x) = 1 \) when the light source is visible from \( x \), and let \( S(x) = 0 \) when it not (i.e. shadow).

\[
I(x) = S(x) \left\{ I_l(x) k_d(x) \mathbf{n}(x) \cdot \mathbf{I}(x) + I_{\text{specular}}(x) \right\} + I_{\text{amb}} k_e(x)
\]

Shadow Mapping (basic idea)

Instead of asking:
"for each point in the scene, which lights are visible?"
we ask
"what is seen from the light source's viewpoint?"

"Shadow map" [Haines and Greenberg 1986] = a depth map as seen from the light source

This term is potentially confusing. A surface is seen from the light source when it is NOT in shadow.
Let \((x_{\text{light}}, y_{\text{light}}, z_{\text{light}})\) be \textit{continuous} light source coordinates.

Let \((x_{\text{shadow}}, y_{\text{shadow}}, z_{\text{shadow}})\) be \textit{discrete} shadow map coordinates, with \(z_{\text{shadow}}\) having 16 bits per pixel.

In both cases, assume the points have been projectively transformed, and coordinates are normalized to \([0, 1] \times [0, 1] \times [0, 1]\).

Notation:

**Shadow Mapping algorithm (sketch)**

for each camera image pixel \((x_p, y_p)\) {
  find depth \(z_p\) of closest visible surface // using whatever method
transform \((x_p, y_p, z_p)\) to light coordinates \((x_{\text{light}}, y_{\text{light}}, z_{\text{light}})\)
  compare \(z_{\text{light}}\) to \(z_{\text{shadow}}(x_{\text{shadow}}, y_{\text{shadow}})\) to decide if
    3D point is in shadow
  compute RGB
}

What causes the aliasing shown on the previous slide? Consider the 2D xz example below.

\((x_{\text{light}}, z_{\text{light}})\) is on a \textit{continuous} visible surface (blue curve).

\((x_{\text{shadow}}, z_{\text{shadow}})\) is in the \textit{discretized} shadow map (black points).

The shadow condition, \(z_{\text{shadow}}(x_{\text{shadow}}, y_{\text{shadow}}) < z_{\text{light}}\), is supposed to be false because all points on the surface are visible. However, because of discretization, the condition is often true and the algorithm mistakenly concludes that some points are shadowed.

To conclude that \((x_{\text{light}}, z_{\text{light}})\) is in shadow, in the presence of discretization, we require that a stronger condition is met:

\[ z_{\text{shadow}}(x_{\text{shadow}}) < z_{\text{light}} - \epsilon. \]

However, as we show on next slide, this can lead us to conclude that a point is not in shadow when in fact it is in shadow.

In fact, there is no gap between ground and vertical wall.

Yet algorithm allows light to leak under the wall.
It fails to detect this shadow.
Real Time Rendering

How are shadows computed in the OpenGL pipeline?

Pass 1: make shadow map and store as a texture
Pass 2: make RGB image using shadow map

How to handle 'diffuse lighting'?
- outdoors on an overcast day
- uniformly illuminated indoor scene
  e.g. classroom, factory, office, retail store

Visibility of the "sky"

Solution 1: (cheap) use attached shadows only.
I skipped this in the lecture because I thought I was running out of time. See Exercises.
Assume:
- the light source is a uniform distant hemisphere
- the fraction of the source is determined only by the surface normal.

What is the differences of this model and "sunny day"?
Solution 2: Ray tracing (expensive)

for each pixel in the image {
    cast a ray through the scene point to find the nearest surface (x,y,z)
    shoot out rays from (x, y, z) into the hemisphere and check which of them reach the "sky"
    i.e. infinity or some finite distance
    add up environment contribution of rays that reach the "sky"
    // you could have a non-uniform sky
}

Solution 3: Ambient Occlusion [Zhukov,1998]

// precompute
for each vertex x {
    shoot out rays into the hemisphere and calculate the fraction of rays, S(x) in [0, 1], that reach the "sky"
    // S(x) is an attribute of x, along with n and material
    // If you are willing to use more memory, then store the a boolean map S(x, l), where l is direction of light
    // See Exercises.
}

// We say that S(x) is "baked" into the surface.

I_{\text{diffuse}}(x) = n(x) \cdot l

This example has no shadows, point source at upper right, and uniform reflectance

I_{\text{diffuse}}(x) = S(x)

Ambient occlusion can replace n(x).l term in more general Blinn-Phong model instance.

This allows for real-time rendering (moving the camera).

Examples of ambient occlusion

Q: How to do ambient occlusion with indoor scenes?
A: For each vertex, compute S(x) in [0, 1] by considering only surfaces within some distance of x.

Ambient Occlusion in My Own Research

No, I will not ask about this (or the next two slides) on the final exam.

Ph.D. thesis

- "Shape from shading on a cloudy day" (Langer & Zucker, 1994)
I independently discovered the principle of ambient occlusion.
I used it to introduce a new version of a classic computer vision problem (shape from shading).

Post-doctoral Research [Langer & Buelthoff, 2000]

- I carried out the first shape perception experiments that compared images rendered with vs. without ambient occlusion.
Local intensity maxima occur in valleys where surface normal turns to face the visible part of the light source.

One of our experiments looked at whether people (and computer vision algorithms) misinterpret these as local "hills".

---

Interreflections (sketch only)

Computing interreflections is a linear algebra problem:

\[ I(x) = I_{\text{direct}}(x) + I_{\text{indirect}}(x) \]

But \( I_{\text{indirect}}(x) \) is the sum of \( I(y) \) for points \( y \) that are visible from \( x \). Thus,

\[ I(x) = I_{\text{direct}}(x) + \sum y F(x,y) I(y) \]

where \( F(x,y) \) depends on:
- distance between \( x \) and \( y \)
- \( n(x) \) and \( n(y) \)
- whether \( x \) and \( y \) see each other

These are huge vectors and matrices. Researchers have developed many clever numerical methods for solving this.

---

Shadows
- ray tracing
- shadow maps
- ambient occlusion

Interreflections

---

Cornell Box (1984) [http://www.graphics.cornell.edu/online/box/]

Surfaces are illuminated directly by light sources, but also indirectly by other surfaces. This leads to color bleeding. (Red and green walls bleed color onto cubes.)

These interreflection methods ("global illumination") can now produce photorealistic images of complex scenes. Solutions are still expensive though...