Today
Shadows for interactive rendering
• Introduction
• Shadow mapping
• Shadow volumes

Why are shadows important?
• Cues on scene lighting

Why are shadows important?
• Contact points
• Depth cues

Why are shadows important?
• Realism

Terminology
• Umbra: fully shadowed region
• Penumbra: partially shadowed region

Without self-shadowing
Without self-shadowing
Hard and soft shadows
• Point and directional lights lead to hard shadows, no penumbra
• Area light sources lead to soft shadows, with penumbra

Shadows for interactive rendering
• Focus on hard shadows
• Two main techniques
  - Shadow mapping
  - Shadow volumes
• Many variations, subtleties
• Active research area

Shadow mapping
Main idea
• Scene point is lit by light source if it is visible from light source
• Determine visibility from light source by placing camera at light source position and rendering scene

Two pass algorithm
First pass
• Render scene by placing camera at light source position
• Store depth image (shadow map)
Two pass algorithm

Second pass
- Render scene from camera position
- At each pixel, compare distance to light source with value in shadow map
  - If distance is larger, we are in shadow
  - If distance is smaller or equal, pixel is lit

Issues
- Limited field of view of shadow map
- Z-fighting
- Sampling problems

Limited field of view
- What if a scene point is outside the field of view of the shadow map?

Limited field of view
- What if a scene point is outside the field of view of the shadow map?
  - Use six shadow maps, arranged in a cube
  - Requires rendering pass for each shadow map!

z-fighting
- Depth values for points visible from light source are equal in both rendering passes
- Because of limited resolution, depth of pixel visible from light could be larger than shadow map value
- Need to add bias in first pass to make sure pixels are lit

Solution
- Add bias when rendering shadow map
  - Move geometry away from light by small amount
- Finding correct amount of bias is tricky

Correct bias Not enough bias Too much bias
Bias

Not enough

Too much

Correct

Sampling problems

• Shadow map pixel may project to many image pixels
• Ugly stair-stepping artifacts

Solutions

• Increase resolution of shadow map
  - Not always sufficient
• Split shadow map into several slices
• Tweak projection for shadow map rendering
  - Light space perspective shadow maps (LiSPSM)
    http://www.cg.tuwien.ac.at/research/vr/lispsm/
  - With GLSL source code!
• Combination of splitting and LiSPSM
  - Basis for most serious implementations

LiSPSM

Basic shadow map

Light space perspective shadow map

Percentage closer filtering

• Instead of looking up one shadow map pixel, look up several pixels
• Perform depth test for each shadow map pixel
• Compute percentage of pixels that are lit

Percentage closer filtering

• Supported in hardware for small filters (2x2 shadow map pixels)
• Can use larger filters with additional rendering passes
• Fake soft shadows
**Shadow mapping with GLSL**

**First pass**
- Render scene by placing camera at light source position
- Compute light view (look-at) matrix
  - Similar to computing camera matrix from look-at, up vector
  - Compute its inverse to get world-to-light transform
- Determine view frustum such that scene is completely enclosed
  - Use several view frustums/shadow maps if necessary

**First pass**
- Each vertex point is transformed by
  \[ P_{\text{light}} V_{\text{light}} M \]
  - Object-to-world (modeling) matrix \( M \)
  - World-to-light space matrix \( V_{\text{light}} \)
  - Light frustum (projection) matrix \( P_{\text{light}} \)
- Remember: points within frustum are transformed to unit cube \([-1,1]^3\)

**Second pass**
- Render scene from camera
- At each pixel, look up corresponding location in shadow map
- Compare depths with respect to light source

**Looking up shadow map**
- Need to transform each point from object space to shadow map
- Shadow map texture coordinates are in \([0,1]^2\)
- Transformation from object to shadow map coordinates
  \[ T' = \begin{bmatrix} T_{xx} & T_{xy} & T_{xz} \\ T_{yx} & T_{yy} & T_{yz} \\ T_{zx} & T_{zy} & T_{zz} \end{bmatrix} \]
  - \( T \) is called texture matrix
- After perspective projection we have shadow map coordinates

**Looking up shadow map**
- Transform each vertex to normalized frustum of light
  \[
  \begin{bmatrix} s \\ t \\ r \\ q \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} T \\
  \]
  - Pass \( s,t,r,q \) as texture coordinates to rasterizer
  - Rasterizer interpolates \( s,t,r,q \) to each pixel
  - Use projective texturing to look up shadow map
  - This means, the texturing unit automatically computes \( s/q,t/q,r/q,1 \)
  - \( s/q,t/q \) are shadow map coordinates in \([0,1]^2\)
  - \( r/q \) is depth in light space
- Shadow depth test: compare shadow map at \((s/q,t/q)\) to \(r/q\)
**GLSL specifics**

*In application*
- Store matrix $T$ in OpenGL texture matrix
- Set using `glMatrixMode(GL_TEXTURE)`

*In vertex shader*
- Access texture matrix through predefined uniform `gl_TextureMatrix`

*In fragment shader*
- Declare shadow map as `sampler2DShadow`
- Look up shadow map using projective texturing with `vec4 texture2DProj(sampler2D, vec4)`

**Implementation specifics**

- When you do a projective texture look up on a `sampler2DShadow`, the depth test is performed automatically
  - Return value is $(1,1,1,1)$ if lit
  - Return value is $(0,0,0,1)$ if shadowed
- Simply multiply result of shading with current light source with this value

**Demo**

- Cg tutorial examples shadowMapping

**Today**

Shadows for interactive rendering
- Introduction
- Shadow mapping
  - Shadow volumes

**In shadow or not**

- Use shadow volume to perform such a test
  1. Allocate a counter per pixel
  2. Cast a ray into the scene, starting from eye
  3. Increment the counter when the ray enters the shadow volume
  4. Decrement the counter when the ray leaves the shadow volume
  5. When we hit the object, check the counter.
     - If counter > 0, in shadow
     - Otherwise, not in shadow
### In shadow or not

<table>
<thead>
<tr>
<th>Light source</th>
<th>Occluder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eye position</td>
<td></td>
</tr>
</tbody>
</table>

In shadow or not

### Shadow volume construction

- Need to generate shadow polygons to bound shadow volume
- Extrude silhouette edges from light source

### Shadow volume construction

- Needs to be done on the CPU
- Silhouette edge detection
  - An edge is a silhouette if one adjacent triangle is front facing, the other back facing with respect to the light
- Extrude polygons from silhouette edges

### Shadow volumes with OpenGL

**Using the stencil buffer**

- A framebuffer channel (like RGB colors, depth) that contains a per-pixel counter
- Stencil test
  - Similar to depth test (z-buffering)
  - Control whether a fragment is discarded or not
  - **Stencil function** to decide whether to discard a fragment
  - **Stencil operation** to decide how the stencil buffer is updated as the result of the test

### Stencil test

- Comparison test between reference and stencil value
  - GL NEVER always fails
  - GL ALWAYS always passes
  - GL LESS passes if reference value is less than stencil buffer
  - GL EQUAL passes if reference value is equal to stencil buffer
  - Etc.
- **If the stencil test fails**, the fragment is discarded and the stencil operations associated with the stencil test failing is applied to the stencil value
- **If the stencil test passes**, the depth test is applied
  - **If the depth test passes**, the fragment continue through the graphics pipeline, and the stencil operation for stencil and depth test passing is applied
  - **If the depth test fails**, the stencil operation for stencil passing but depth failing is applied

### Stencil operation

- Can set different operations for
  - Stencil fails
  - Stencil passes, depth fails
  - Stencil passes, depth passes
- **Options**
  - GL KEEP stencil value unchanged
  - GL ZERO stencil value set to zero
  - GL REPLACE stencil value replaced by stencil reference value
  - GL INCR stencil value incremented
  - GL DECR stencil value decremented GL INVERT
    stencil value bitwise inverted
Shadow volume algorithm
Z-pass approach
- Render scene with only ambient light
  - Update depth buffer
- Turn off depth and color write, turn on stencil, keep the depth test on
- Init stencil buffer to 0
- Draw shadow volume twice using face culling
  - 1st pass: render front faces and increment stencil buffer when depth test passes
  - 2nd pass: render back faces and decrement when depth test passes
- At each pixel
  - Stencil ! = 0, in shadow
  - Stencil = 0, lit
- Render the scene again with diffuse and specular lighting
  - Write to framebuffer only pixels with stencil = 0

Issues
- Z-pass fails if
  - Eye is in shadow
  - Shadow polygon clipped by near clip plane

Shadow volume algorithm
Z-fail approach
- Count number of invisible (occluded) front-facing and back-facing shadow volume polygons
  - Update stencil buffer when depth test fails
- If equal, pixel is not in shadow

Issues
- Z-fail fails if
  - Shadow polygon clipped by far plane

Demo
- Cg toolkit, browser, infinite shadow volumes

Summary
- Two main techniques for shadows in interactive rendering
  - Shadow mapping
  - Shadow volumes
Shadow volumes

**Pros**
- Does not require hardware support for shadow mapping
- Pixel accurate shadows, no sampling issues

**Cons**
- More CPU intensive
- Fill-rate intensive
- Expensive for complex geometry
- Tricky to handle all cases correctly
- Hard to extend to soft shadows

Shadow maps

**Pros:**
- Little CPU overhead
- No need to construct geometry to represent shadows
- Hardware support
- Can fake soft shadows easily

**Cons:**
- Sampling issues
- Depth bias is not completely foolproof
- Shadow mapping has become more popular with better hardware support

Recommendation

- If you like to do shadows for the final project, implement shadow mapping

**Links**
- Overview, lots of links
  - [Overview](http://www.realtimerendering.com/)
- Basic shadow maps
  - [Overview](http://en.wikipedia.org/wiki/Shadow_mapping)
- Avoiding sampling problems in shadow maps
  - [Overview](http://www.cs.technion.ac.il/~tams/num/num.pdf)
- Faking soft shadows with shadow maps
  - [Overview](http://people.csail.mit.edu/ericchan/papers/smoothie/)
- Shadow volumes
  - [Overview](http://en.wikipedia.org/wiki/Shadow_volume)

Next time

- Procedural modeling