Final projects

- Presentations next week
- Tentative schedule (web form soon)
  - Wednesday 10-12 and 2-4
  - Thursday 10-12 and 1-2
  - Friday 10-12 and 2-4
- 20 minutes presentation + 5 minutes Q&A
- Final report due on Friday Dec 6

Special topics

- Today: Shadows
- Tuesday Dec 3: Graphics hardware
- Thursday Dec. 5: Image-based modeling & rendering

Shadows

- Realism
- Depth cue

Shadows as depth cue
Shadows and art

• Only in Western pictures (here Caravaggio)

Duality shadow-view

• A point is lit if it is visible from the light source
• Shadow computation very similar to view computation

Shadow ray

• Ray from visible point to light source
• If blocked, discard light contribution
• One shadow ray per light
• Optimization?
  – Stop after first intersection (don’t worry about tmin)
  – Test latest obstacle first

Ray-casting shadows

Overview

• Shadow map
  – Image-precision, texture mapping
• Shadow volume
  – Object space
• Soft shadows and Monte-Carlo ray tracing
Questions?

Fake shadows using textures
- Separate obstacle and receiver
- Compute b/w image of obstacle from light
- Use projective textures

Image from light source  BW image of obstacle  Final image

Fake shadows using textures
- Limitations?

Limitations?

Shadow maps
- Use texture mapping but using depth
- 2 passes (at least)
  - Compute shadow map from light source
    - Store depth buffer (shadow map)
  - Compute final image
    - Look up the shadow map to know if points are in shadow

Shadow map look up
- We have a 3D point $x,y,z$
- How do we look up the shadow map?

Shadow map look up
- We have a 3D point $x,y,z$
- How do we look up the shadow map?
  - Use the 4x4 camera matrix from the light source
  - We get $(x',y',z')$
  - Test: $\text{ShadowMap}(x',y') < z'$
Shadow maps

- In Renderman
  - (High-end production software)

- Can be done in hardware
- Using hardware texture mapping
  - Texture coordinates u,v,w generated using 4x4 matrix
  - Modern hardware permits tests on texture values

Problems with shadow maps?

- Field of view
- Bias
- Aliasing

Field of view problem

- What if point to shadow is outside field of view of shadow map?
- Use cubical shadow map
- Use only spot lights!

The bias nightmare

- For a point visible from the light source
  \( \text{ShadowMap}(x',y') \approx z' \)
- Avoid erroneous self shadowing
- Remember the ray-tracing shadows in assignment 6

The bias nightmare

- Remember shadow ray casting
  - We started the ray at \( \text{hit+light}*\epsilon \)
  - We added bias to avoid degeneracy
  - Yet another instance of geometric robustness
Bias for shadow maps

\[ \text{ShadowMap}(x', y') + \text{bias} < z' \]

Choosing the good bias value can be very tricky.

<table>
<thead>
<tr>
<th>Correct image</th>
<th>Not enough bias</th>
<th>Too much bias bias</th>
</tr>
</thead>
</table>

Shadow map aliasing

- Undersampling of shadow map
- Reprojection aliasing

Shadow map filtering

- Does not work!
- Filtering depth is not meaningful

Percentage closer filtering

- Filter the result of the test
- But makes the bias issue more tricky

Percentage closer filtering

- 5x5 samples
- Nice antialiased shadow
- Using a bigger filter produces fake soft shadows
- But makes the bias issue more tricky

Shadows in production

- Often use shadow maps
- Ray casting as fallback in case of robustness issues
Questions?

Overview

- Shadow map
  - Image-precision, texture mapping
- Shadow volume
  - Object space
- Soft shadows and Monte-Carlo ray tracing

Shadow volumes

- Explicitly represent the volume of space in shadow
- For each polygon
  - Pyramid with point light as apex
  - Include polygon to cap
- Shadow test similar to clipping

Shadow volumes naïve rendering

- Pick your favorite rendering algorithm
- For each visible point
  - For each light
    - For each shadow volume
      - If point inside volume
        - Point is in shadow

Great but costly
(#poly * #light tests for each visible point)

Shadow volumes smarter rendering

- New way to define inside/outside
- Consider ray from eye to visible point
- Increment or decrement a counter each time we intersect a shadow volume polygon
- Points lit: counter = 0

Hardware shadow volumes

- Add counter buffer
- Draw scene with no shading
- Turn off buffer update
- Draw front-facing shadow polygons
  - If z-pass
    - Increment counter
- Draw back-facing shadow polygons
  - If z-pass
  - Decrement counter
- Compute shading
  - Lit Points have counter=0
Optimizing shadow volumes

- Use silhouette edges only

Problem if eye inside shadow

- Then a counter of 0 does not necessarily mean lit
- Three solutions
  - Explicitly test eye point wrt all volumes
  - Clip the shadow volumes and include these new polygons
  - Z-fail shadow volumes

Problem if eye inside shadow

- Clip the shadow volumes and include these new polygons

Z-fail shadow volume

- Count from infinity
- Draw scene with no shading
- Turn off buffer update
- Draw front-facing shadow polygons
  - If z-fail
    - Decrement counter
- Draw back-facing shadow polygons
  - If z-fail
    - Increment counter
- Compute shading
  - Lit points have counter=0

Questions?
Overview

- Shadow map
  - Image-precision, texture mapping

- Shadow volume
  - Object space

- Soft shadows and Monte-Carlo ray tracing

Soft shadows

- Caused by extended light sources
- Umbra
  - Source completely occluded
- Penumbra
  - Source partially occluded
- Fully lit

Soft shadows

- Radiosity
- Shadow map filtering
  - Hack
  - Not accurate
  - Bias issue
- Sample the extended light source

Soft shadows using ray-tracing

- Send multiple rays to sample the light source
- Use the fraction of occluded rays

Traditional Ray Tracing

Ray Tracing+soft shadows
Monte-Carlo ray-tracing

- Probabilistic sampling approach
- Solve the complete light transport equation
- Probabilistic alternative to radiosity

Monte-Carlo computation of $\pi$

- Take a square
- Take a random point $(x, y)$ in the square
- Test if it is inside the $\frac{1}{4}$ disc ($x^2 + y^2 < 1$)
- The probability is $\pi / 4$

- The probability is $\pi / 4$
- Count the inside ratio $n = \text{# inside} / \text{total # trials}$
- $\pi \approx n \times 4$
- The error depends on the number of trials

Monte-Carlo

- Cast a ray from the eye through each pixel
- Cast random rays from the visible point

- Cast a ray from the eye through each pixel
- Cast random rays from the visible point
- Recurse
Monte-Carlo

- Cast a ray from the eye through each pixel
- Cast random rays from the visible point
- Recurse

Monte-Carlo

- Systematically sample primary light

Monte-Carlo

- Take BRD (e.g. Phong model) into account
  - Multiply incoming light
  - Sampling density

Radiance rendering system (Ward)

Radiance rendering system (Ward)
Variation: Photon mapping

- Animation by Henrik Wann Jensen

Questions?

Next week

- Graphics hardware

- Image-based modeling & rendering