CSE168
Computer Graphics II, Rendering

Spring 2006
Matthias Zwicker
Today

- Course overview
- Organization
- Introduction to ray tracing
Course overview

• Algorithms for creating photo-realistic images
• Having fun improving your C++ programming skills
Images

Ray tracing

Soft shadows

Caustics

Global illum.
Images
Images
Images
Images

[Image of a modern interior design with glass walls and minimal furniture]

[Jensen]
Images
Images
What you should know

- Basic 3D graphics (CSE167)
- Basic knowledge of C++
- Shirley, Chapter 1-6: Linear algebra, vectors, matrices
Syllabus

Part 1: Implementing a basic ray tracer

• Ray tracing algorithm
• Camera model
• Ray-geometry intersection techniques
• Shading
• Acceleration structures
• Textures, environments maps, bump maps
Syllabus

Part 2: The physics of light transport and how to simulate it

- The rendering equation
- Monte carlo ray tracing
- Photon mapping
- Radiosity
Part 3: Advanced topics

- Sampling and anti-aliasing
- Realistic camera models
- High dynamic range imaging
- Advanced geometry
- Subsurface scattering and participating media
Assignments

1. Basic ray tracer
2. Acceleration structures
3. Texturing
4. Rendering a realistic image (a.k.a. “Rendering competition”)
Previous images

Sameer, 2003
Previous images
Previous images

Craig, 2003
Previous images

Cyrus, 2003
Previous images

Arash, 2003
Previous images

Wojciech, 2004
Previous images

Siddharta, 2004
Previous images

Alex, 2005
Organization

Grading

• Assignment 1: 20%
• Assignment 2: 20%
• Assignment 3: 15%
• Final project: 30%
• Midterm: 15%
Organization

- Course webpage
  [http://graphics.ucsd.edu/courses/cse168_s06/](http://graphics.ucsd.edu/courses/cse168_s06/)
- Mailing list
  [cse168@graphics.ucsd.edu](mailto:cse168@graphics.ucsd.edu)
- TA
  Wojciech Jarosz, [cse168-ta@graphics.ucsd.edu](mailto:cse168-ta@graphics.ucsd.edu)
- Lab
  EBU3B 240-260
Organization

- Lectures
  Mondays, Wednesdays, 5 00pm - 6 20pm, WLH 2204
- Discussion section
  TBD
- Lab hours
  TBD
- Office hours
  Fridays 2 30 -3 30pm
Questions?
Introduction to ray tracing

Image generation (rendering)
  • Z-buffering
  • Ray tracing
Z-buffering

Rendering pipeline

Command
→ Geometry
→ Rasterization
→ Texture
→ Display

Object order algorithm
Z-buffering

- Implemented in graphics hardware (ATI, NVidia GPUs)
- Standardized APIs (OpenGL, Direct3D)
- Interactive rendering, games
- Limited photo-realism
Ray tracing

- Imager
  - Primary rays
    - Find first intersection
      - Shading
        - Scene representation
- Output image
- Image order algorithm
Ray tracing

- Usually implemented in software
- Photo-realistic images
- Slow, batch rendering
- No dominant standard for scene description (RenderMan format, POV ray, PBRT, ...)
Ray tracing pseudocode

```
ray_trace() {
    construct scene representation

    for each pixel
        trace( primary ray)
}
```
Ray tracing pseudocode

trace( ray ) {
    find first intersection
    shade( hit )
}

Data structures

class object {
    bool intersect( ray )
}

class light {
    void illuminate(...)
}

class ray {
    vector origin
    vector direction
}
Computing primary rays

- Define camera coordinate system and viewing frustum
- Given an image pixel
  - Determine ray in camera coordinates
  - Transform ray to canonic coordinates
Camera coordinate system
Camera coordinate system
Camera coordinate system

- Given from, to, up

\[ w = \frac{\text{from} - \text{to}}{||\text{from} - \text{to}||} \]

\[ u = \frac{\text{up} \times w}{||\text{up} \times w||} \]

\[ v = w \times u \]

\[ e = \text{from} \]
Viewing frustum

- Vertical field-of-view $\theta$
- Aspect ratio width/height \textit{aspect}
- Image plane at $w=-1$
Viewing frustum
Computing primary rays

\[ p_{uvw}(t) = e + t(s - e) \]
Computing primary rays

- Vertical field-of-view $\theta$
- Aspect ratio width/height $aspect$

\[
\begin{align*}
t &= -b = \tan(\frac{\theta}{2}) \\
r &= -l = aspect \cdot t
\end{align*}
\]
Computing primary rays

- Image resolution $m \times n$ pixels

$$\begin{align*}
u(i, j) &= l + (r - l) \frac{i + 0.5}{m} \\
v(i, j) &= b + (t - b) \frac{j + 0.5}{n}
\end{align*}$$
Computing primary rays

- Primary ray in camera coordinates

\[ p_{uvw}(t) = e + t(s - e) \]

\[ s = \begin{bmatrix} u(i, j) \\ v(i, j) \\ -1 \\ 1 \end{bmatrix} \]
Computing primary rays

- Transformation to canonic coordinates
- Column vectors $u, v, w, e$
- Transformation matrix
  \[
  M = \begin{bmatrix}
  u & v & w & e
  \end{bmatrix}
  \]
- Multiply matrices from the left
  \[
  p_{xyz}(t) = Mp_{uvw}(t)
  = Me + t(Ms - Me)
  \]
Transforming Normal Vectors

- Normal is perpendicular to tangent
  \[ n^T \cdot t = 0 \]
- Under transformation \( M t \), find normal transformation \( X_n \) that satisfies normal constraint
  \[ (X_n)^T \cdot M t = 0 \]
  \[ n^T X^T M t = 0 \]
  \[ \Rightarrow X^T M = I \Rightarrow X = M^{-1^T} \]