Now Playing:

Still Alive
Jonathan Coulton
from *Portal: The Game*
Released October 9, 2007

Real Cameras and Light Transport

Rick Skarbez, Instructor
COMP 575
October 23, 2007

Announcements

- Programming Assignment 2 (3D graphics in OpenGL) is out
  - Due THIS Thursday, October 25 by 11:59pm
- Programming Assignment 3 (Rasterization) is out
  - Due NEXT Thursday, November 1 by 11:59pm

Last Time

- Finished our discussions of mapping
  - Texture Mapping
  - Bump Maps
  - Displacement Maps
- Talked about programmable graphics hardware
- Discussed the class project

Today

- Further discussion of real lights and real cameras
- Easing into ray-tracing

Recap of Rasterization-Based Rendering

- Very fast
  - Computer games get 30+ frames per second on commodity hardware
- Not very good (comparatively speaking)
  - Need lots of hacks/workarounds to get effects like shadows, global illumination, reflection, etc.

Recap of Rasterization-Based Rendering

- Everything in the world is a colored polygon
- Projects these polygons onto the "screen" to generate the image seen by the user
- All processed independently
  - One polygon's color doesn't depend on another's

Hallmarks of Rasterization

- Very high rendering speed
- Lots of parallelism
- Independence of vertices/polygons/pixels
- Runs in "assembly line" fashion
- Not very realistic
  - We have other options, though...
  - Let's revisit the rendering equation

The Rendering Equation

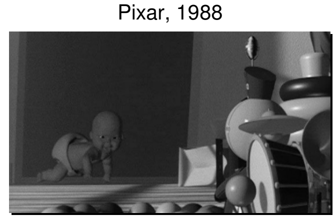
Jim Kajiya, 1986

$$L_o(x,\vec{w}) = L_e(x,\vec{w}) + \int_\Omega f_r(x,\vec{w}',\vec{w}) L_i(x,\vec{w}') (\vec{w}' \cdot \vec{n}) d\vec{w}'$$

- In short, the light out from a point in a specific direction depends on:
  - The light it emits in that direction
  - The light incident on that point from every direction, affected by
    - How reflective the material is for that pair of directions
    - How close the incoming direction is to the surface normal

Movie Break! Tin Toy

Pixar, 1988



Available online: http://v.youku.com/v_show/id_ch00XMTE1OTIw.html

Lights, Cameras, and Surfaces

- Remember way back at the beginning of the semester we talked about these?
  - Now we're ready to talk about them again

Lights, Cameras, Surfaces

- How are these connected in the real world?
  - Light!
  - More precisely, photons
    - Light sources emit photons
    - Surfaces reflect & absorb photons
    - Cameras measure photons

Photons

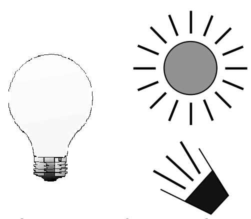
- The quanta of electromagnetic energy
  - “Particles of light”
- Appear to be particles and waves simultaneously
  - We ignore wave behavior (i.e. diffraction)
  - We simulate particle behavior

Photons as Rays

- We model photons as “rays” of light
  - Rays in the mathematical sense
    - Defined by a starting point and a vector

Lights

- There are several kinds of lights (or light sources)
  - Light bulbs
  - The sun
  - Spot Lights
  - Ceiling Lights
- These are different because they emit photons differently

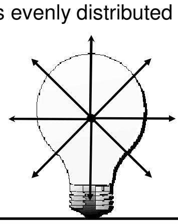


Lights

- We’ve already talked an awful lot about lights
- There are several types:
  - Point lights (*i.e* light bulbs)
  - Directional lights (*i.e.* the Sun)
  - Spot lights
  - Area lights
    - In the real world, just about EVERY light is an area light

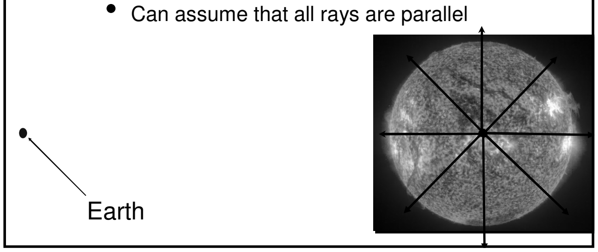
Point Lights (i.e. Light Bulbs)

- Emit light evenly in all directions
  - That is, photons (rays) from a point light all originate from the same point, and have directions evenly distributed over the sphere



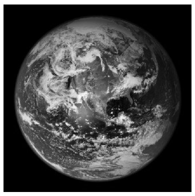
Directional Lights (i.e. the Sun)

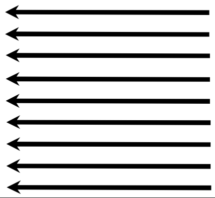
- Point light sources at an infinite (or near infinite) distance
  - Can assume that all rays are parallel



Directional Lights (i.e. the Sun)

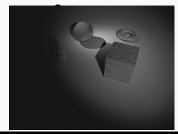
- Point light sources at an infinite (or near infinite) distance
  - Can assume that all rays are parallel

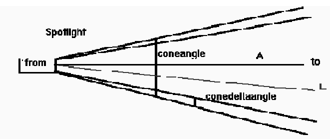




Spot Lights (i.e., well, Spot Lights)

- Similar to point lights, but intensity of emitted light varies by direction
  - Can think of it as only emitting rays over a patch on the sphere





Area Lights (i.e. Ceiling Lights)

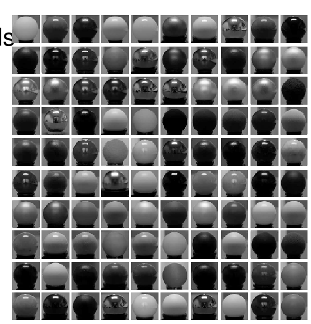
- Emits light in every direction from a surface
  - Can think of it as a set of point lights, or a patch on which every point is a point light

Surfaces

- We've talked a little bit about surfaces
  - In terms of OpenGL lighting
- We're not going to talk about them in a lot more detail here
  - Just remember that in OpenGL, surfaces can't be reflective/refractive
  - In the real world, they can be

Surfaces

- Surface materials can be:
  - Dull
  - Shiny
  - Bumpy
  - Smooth
  - Transparent
  - Translucent...



Surfaces

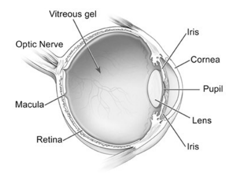
- So what do surfaces do?
  - They selectively react to incoming photons
    - Absorb
    - Reflect
    - Refract
    - Fluoresce
    - etc.

Surfaces

- Surfaces make up the “interesting” things in a scene, but from a light transport point of view, they are just intermediaries

Cameras

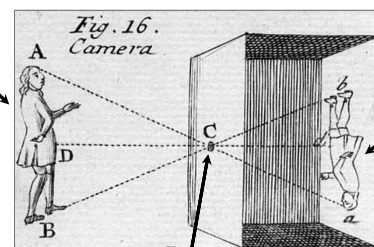
- Almost all cameras work in a similar fashion
  - Use a lens to map rays from the scene (i.e. the world) onto an image plane



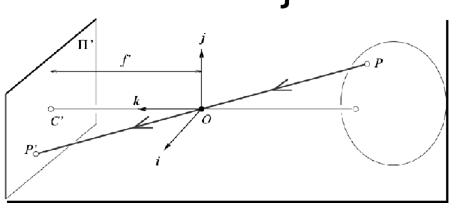
Cameras

- Last time we talked about cameras, we only considered “pinhole” cameras
  - Where all light passes through a single point
    - *i.e.* camera obscura
- These do exist in the real world
- But almost all real cameras have lenses

Camera Obscura



Pinhole Projection



$$\begin{cases} x' = f'\dfrac{x}{z} \\ y' = f'\dfrac{y}{z} \end{cases}$$

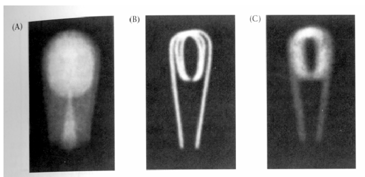
This is essentially what we’ve seen in OpenGL

So why do we need lenses?

- To gather more light
  - In an ideal pinhole camera, exactly one ray would reach each point in the image plane
- To focus the light
  - In an ideal pinhole camera, this is not a problem
  - But in a real camera (even a pinhole one), objects will appear unfocused

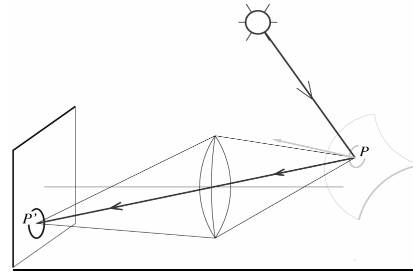
Pinhole Focus Problems

Pinhole Size



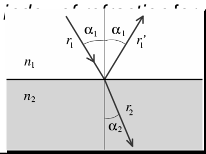
2.18 DIFFRACTION LIMITS THE QUALITY OF PINHOLE OPTICS. These three images of a bulb filament were made using pinholes with decreasing size. (A) When the pinhole is relatively large, the image rays are not properly converged, and the image is blurred. (B) Reducing the size of the pinhole improves the focus. (C) Reducing the size of the pinhole further worsens the focus, due to diffraction. From Ruechardt, 1958.

Projection through a Lens



Refraction

- Lenses refract light passing through them
  - That is, change the direction of rays
- Refraction is governed by Snell's law:
  - $n_1 \sin(\alpha_1) = n_2 \sin(\alpha_2)$
  - n is the ... a material



Paraxial (or First-Order) Optics

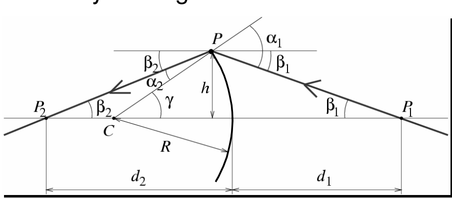
- Assumes that angles are small (*i.e.* the lens is small and/or any objects are far away)

➡Can approximate $\sin(\alpha_1)$ with α_1

- Assumes all lenses are spherical
- Assumes all lenses are symmetric about their optical axes

Paraxial Optics

- Scary-looking math slide



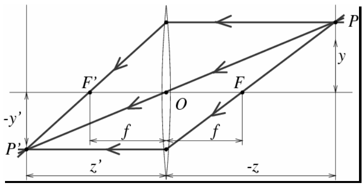
Snell's Law — Small Angles — Paraxial Refraction

$n_1 \sin \langle_1 = n_2 \sin \langle_2$ ➡ $n_1 \langle_1 \; H \; n_2 \langle_2$ ➡ $\frac{n_1}{d_1} + \frac{n_2}{d_2} = \frac{n_2 - n_1}{R}$

Thin Lenses

- Don't need to know all the math from the last slide
  - Just showing it to you
- But we can use that to talk about actual lenses
  - Esp. *thin lenses*, where a ray can be assumed to refract at one boundary, and immediately refract again at the other boundary

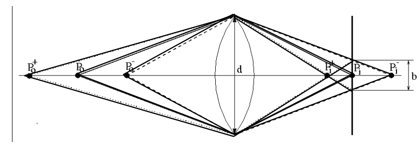
Thin Lenses



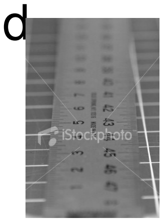
$$\begin{cases} x' = z'\dfrac{x}{z} \\ y' = z'\dfrac{y}{z} \end{cases} \quad \text{where} \quad \frac{1}{z'} - \frac{1}{z} = \frac{1}{f} \quad \text{and} \quad f = \frac{R}{2(n-1)}$$

Depth-of-Field

- Real lenses have what is termed *depth-of-field*
  - The range over which objects in the scene appear in focus

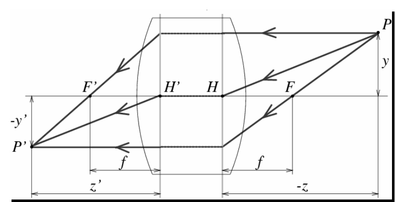


Depth-of-Field Examples



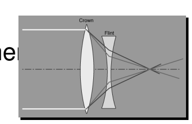
Thick Lenses

- Lenses actually have some thickness
  - We're not going to do the math here



Compound Lenses

- In real cameras, there is usually not just one lens, but a series of lenses
  - The entire set is called a *compound lens*
  - This is done to reduce aberrations introduced by lenses
- We won't discuss this further

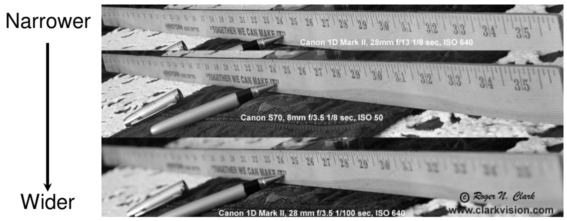


Properties of Cameras

- There are 2 primary characteristics of a camera that affect how much light gets to the film:
  - Aperture
    - How wide the lens is
  - Shutter speed
    - How long the film is exposed
- There are trade-offs between the two

Aperture Artifacts

- A wider aperture produces a narrower depth-of-field



Shutter Speed Artifacts

- A slower shutter speed means the shutter is open for more time
  - This can result in motion blur

Recap

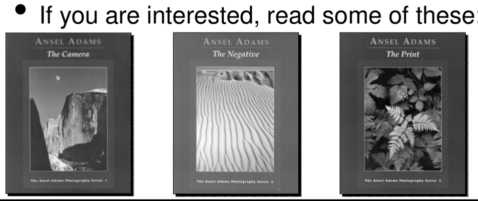
- Images are formed by lights, surfaces, and cameras
  - Lights emit rays
  - Surfaces change rays
  - Cameras capture rays
- Real cameras have lenses
  - These introduce various effects to the final image

Image Plane

- Real cameras have an image plane behind the lens
  - Film cameras project onto film
  - Digital cameras project onto a chip
    - CCD
    - CMOS
- “Virtual cameras”, like in computer graphics, can have the image plane in front of the lens

Photographic Film

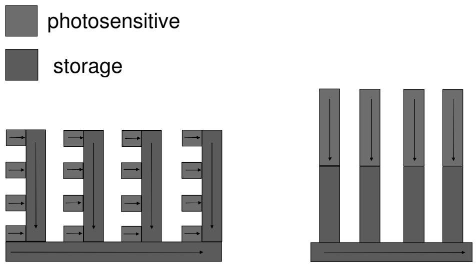
- Too much detail to cover here
- In short, film is covered with a chemical that undergoes a reaction when hit with a photon
- If you are interested, read some of these:



Charge-Coupled Devices (CCDs)

- Have discrete detectors (photometric sensors) at pixel (or subpixel) locations
- When you take a picture:
  - Each pixel is pre-charged
  - The shutter is opened
  - Photons “knock off” electrons from the pixels
  - The shutter is closed
  - Pixels are read and stored

CCDs



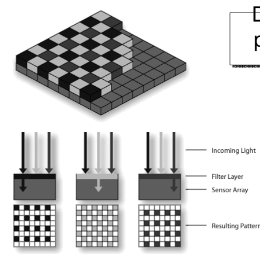
CMOS

- Same sensor elements as CCD
- But read-out is different
  - Uses standard CMOS technology
- Won't go into details
  - Can find a bit more info online if you're interested
    - http://www.image-designer.com/digital-camera-ccd.html

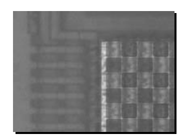
Color in Digital Cameras

- Each sensor only detects, essentially, the number of electrons that hit it
  - Generates a luminance (black & white) image
- 2 ways to get color
  - 3 separate sensor arrays
    - 1 each for red, green, blue
  - Bayer Pattern

Bayer Patterns



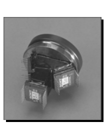
Do you see a problem with this?

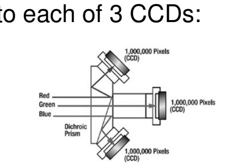


The process of reconstructing the "real" color at each pixel is called *demosaicing*

3 CCD Color

- Uses a beam-splitter to redirect incoming light to each of 3 CCDs:



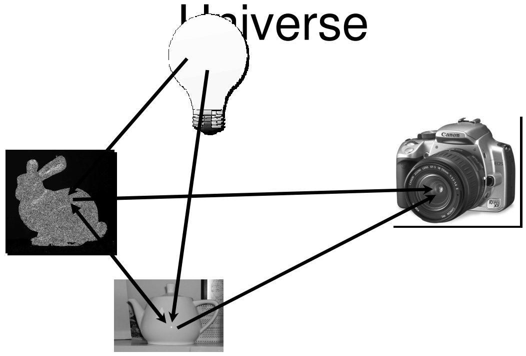


What's better about this?
What's worse?

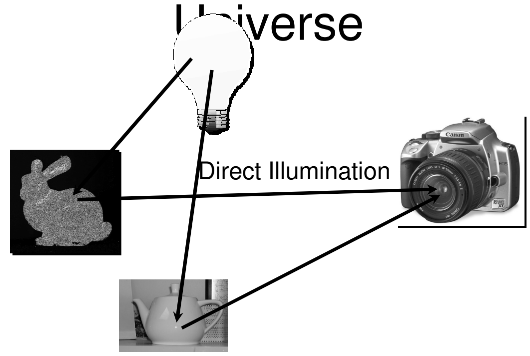
Recap

- Real cameras also need a sensing image plane
  - Photographic film in traditional cameras
  - CCD or CMOS chips in digital cameras
- Digital cameras need to do something extra to get color images
  - Multiple sensors
  - Bayer patterns

A Model of the Universe



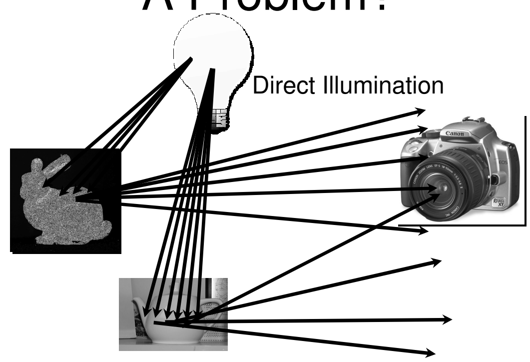
A Model of the Universe



A Problem?

- It is perfectly reasonable to try to implement an algorithm based on the model discussed so far
- However, it would be VERY inefficient
- Why?
  - Many (probably most) light rays in a scene would never hit the image plane

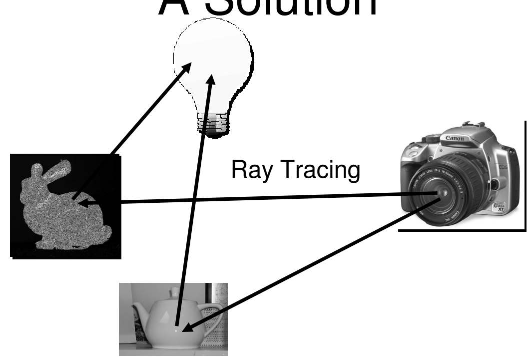
A Problem?



A Solution

- Instead of shooting rays from the light, shoot rays from the camera
  - Shoot one (or, realistically, many) rays per pixel
  - Can stop when you hit a light and/or a non-reflective surface
- This is the basis of the ray tracing algorithm

A Solution



Ray-Tracing Algorithm

- for each pixel / subpixel
  - shoot a ray into the scene
  - find nearest object the ray intersects
  - if surface is (nonreflecting OR light)
    - color the pixel
  - else
    - calculate new ray direction
    - recurse

Ray-Tracing Algorithm

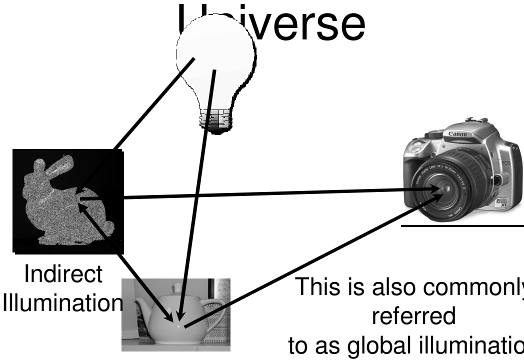
- for each pixel / subpixel
  - shoot a ray into the scene
  - find nearest object the ray intersects
  - if surface is (nonreflecting OR light)
    - color the pixel
  - else
    - calculate new ray direction
    - recurse

Ray Casting

Ray-Tracing Components

1. Generate the rays that are seen by the eye
   - One (or more) for each pixel
   - Need to determine ray origin / directions
2. Figure out what (if anything) those rays hit
   - Compute the nearest intersection
   - Ray-object intersections
3. Determine the color of the pixel

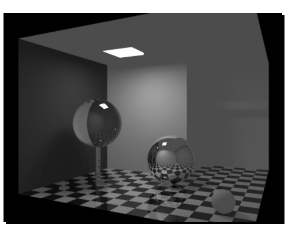
A Model of the Universe



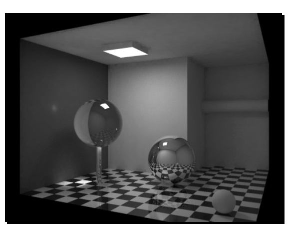
Indirect Illumination

This is also commonly referred
to as global illumination

Direct Illumination vs. Global Illumination



Direct Illumination Only



With Global Illumination

Class Schedule

- This week and next
  - Casting Rays
- Weeks 12 & 13
  - Ray Tracing
  - Radiosity
- Week 14
  - Photon Mapping

Class Schedule

- Week 15
  - Special Topics (tentative)
    - High Dynamic Range (HDR) Rendering
    - Image-Based Rendering
    - Suggestions?
- Week 16
  - Course / Final Exam Review

Next Time

- Figuring out how to cast rays
  - Turning the concepts from today into concrete math
- Programming Assignment 2 is due
  - 11:59pm Thursday