CS 563 Advanced Topics in Computer Graphics

by Emmanuel Agu

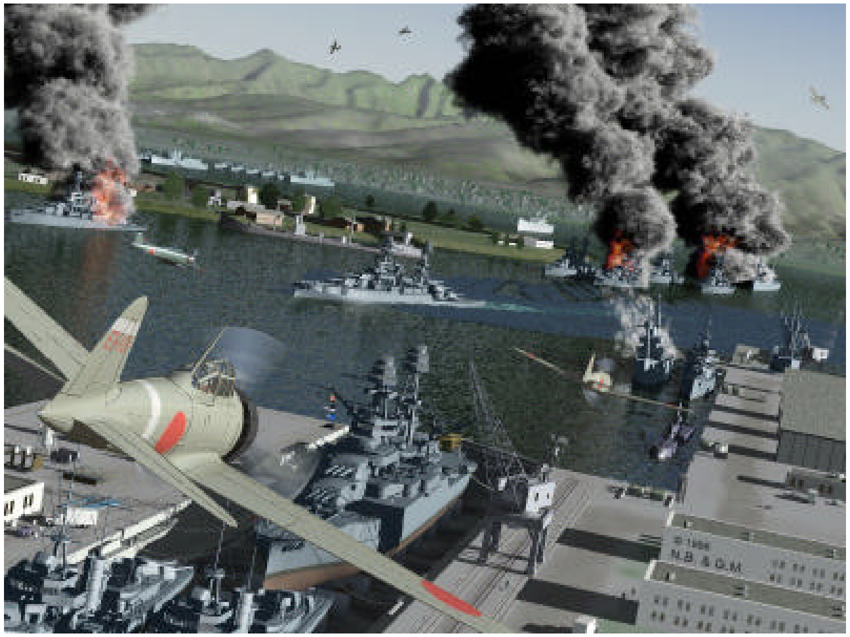
What is Rendering?

- Create a 2D picture of a 3D world
- Photorealistic: Indistinguishable from photo

The Kitchen - Jaime Vives Piqueres - POVCOMP 2004

Applications

- Movies
- Interactive entertainment
- Industrial design
- Architecture
- Demo products
- Virtual reality (games)



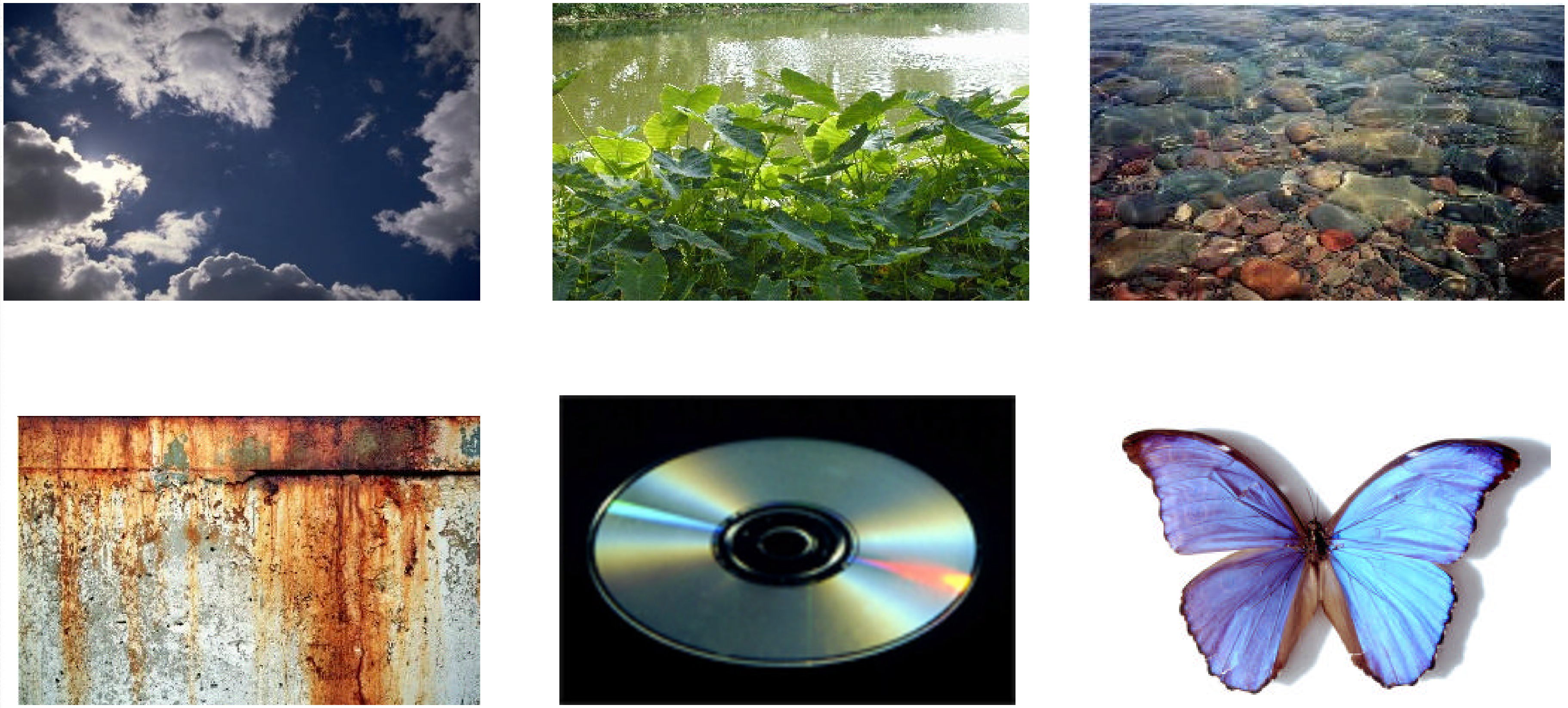
High Quality Rendering

- Ingredients: Require good models for
  - Light source (sky, light bulb, flourescent)
  - Volume through which light travels (smoke, fog, mist, water)
  - Reflection at object surfaces (velvet, wood, polished, rough, smooth)
- Old approach: Fudge it! (Phong's shading)
- New approach:
  - study light physics
  - derive models
  - Use physically-based models for rendering

Physically-based rendering

uses physics to simulate the interaction between matter and light, realism is primary goal

What can we model?



Physically-based Appearance Models

- Why does the sky **appear** blue?
- Why does wet sand **appear** darker than dry sand?
- Why do iridescent surfaces (CD-ROM) **appear** to have different colors when viewed in different directions ?
- Why do old and weathered surfaces **appear** different from new ones?
- Why do rusted surfaces **appear** different from un-rusted ones?
- **Physically-based appearance models** in computer graphics try to use laws of physics to answer these questions

Physically-Based Appearance Modeling

- Using physics-based appearance models to render:
  - Humans (face, skin)
  - Nature (water, trees, seashells)
  - Animals (feathers, butterflies)

Modeling & Simulating Appearance

- Models
    - Light and color
    - Light sources
    - Shapes
    - Materials
        - Interfaces: Reflection and texture models
        - Medium: Atmospheric scattering models
    - Cameras
        - Lens and film
- Simulation
    - Illumination

History: Geometric Aspects First

- Transformation/clipping and the graphics pipeline
  - Evans and Sutherland
- Hidden line and surface algorithms
  - Sutherland, Sproull, Shumacker

History: Simple Shading

- Simple shading and texturing
  - Gouraud ⇒ interpolating colors
  - Phong ⇒ interpolating normals
  - Blinn, Catmull, Williams ⇒ texturing

History: Optical Aspects Second

- Reflection and texture models
  - Cook and Torrance ⇒ BRDF
  - Perlin ⇒ Procedural textures
  - Cook, Perlin ⇒ Shading languages
- Illumination algorithms
  - Whitted ⇒ Ray tracing
  - Cohen, Goral, Wallace, Greenberg, Torrance Nishita, Nakamae ⇒ Radiosity
  - Kajiya ⇒ Rendering equation

Lighting

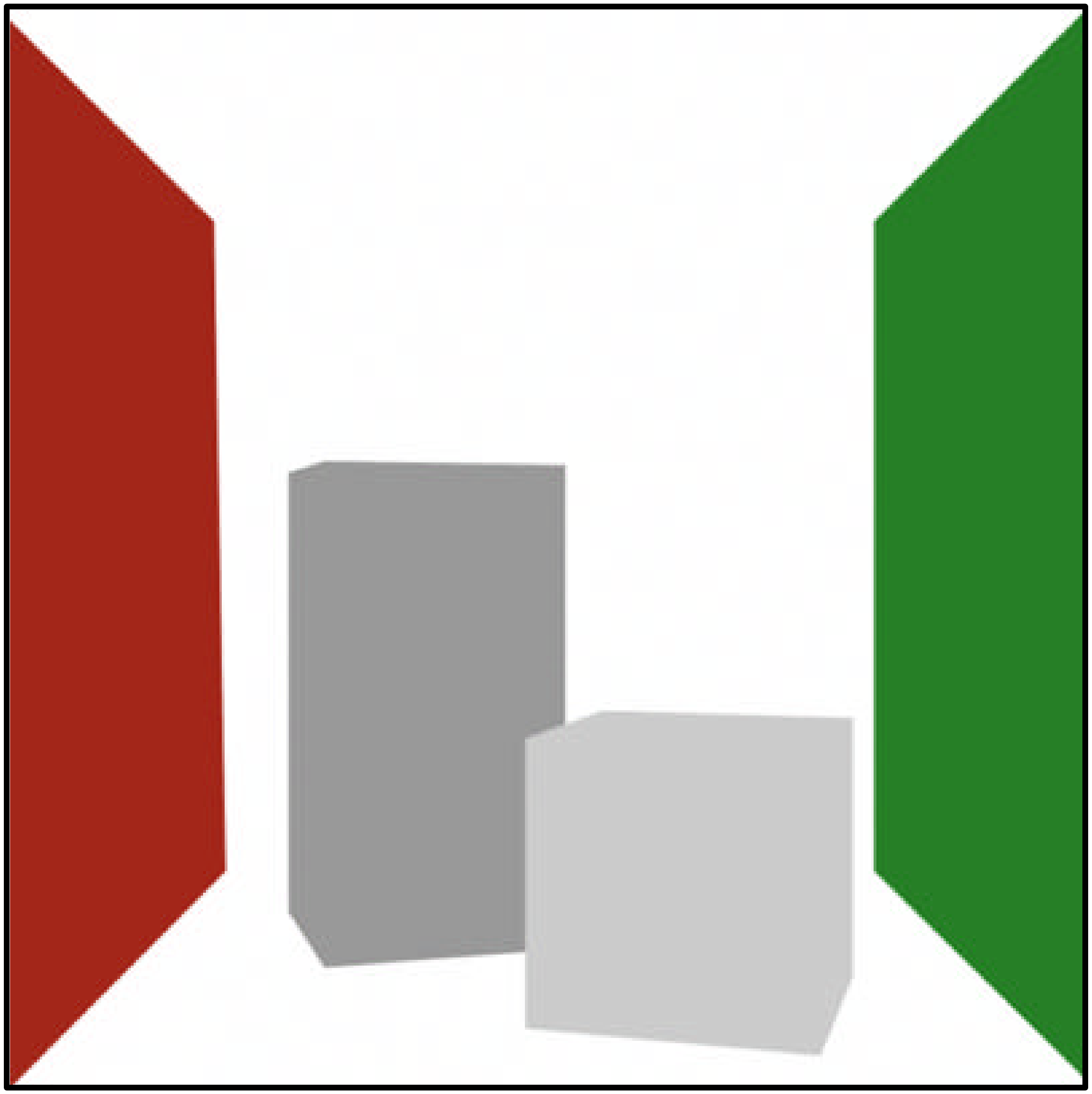
Lighting Simulation

- The Rendering Equation

  Given a scene consisting of geometric primitives with material properties and a set of light sources, compute the illumination at each point on each surface

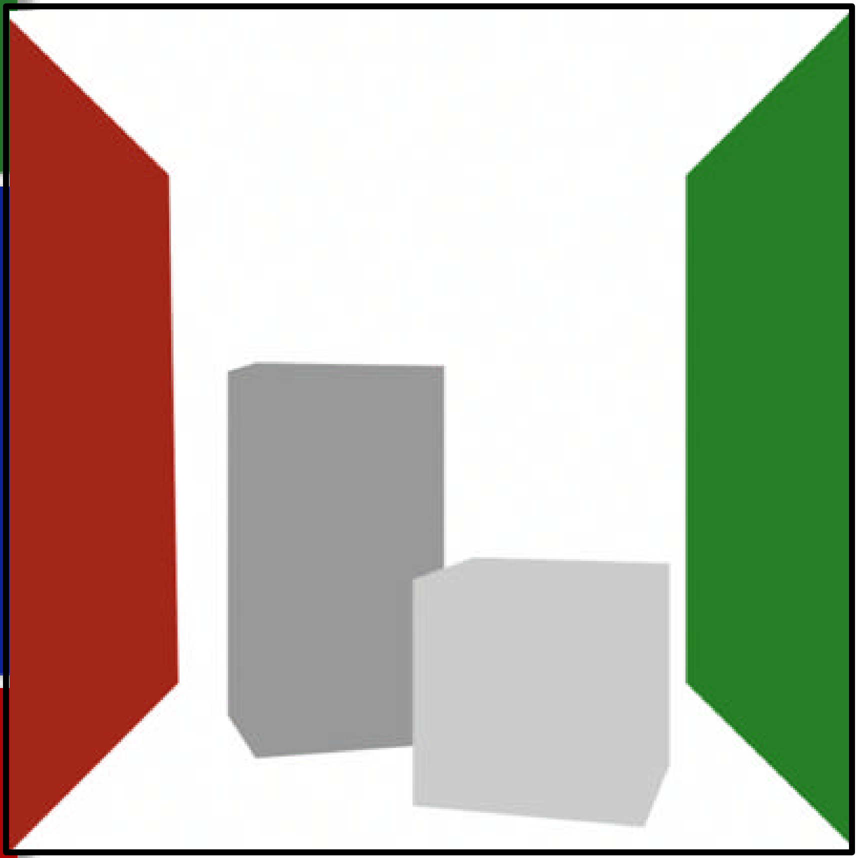
- Challenges
  - Primitives complex: lights, materials, shapes
  - Infinite number of light paths
- How to solve it?
  - Radiosity ⇨ Finite element
  - Ray tracing ⇨ Monte Carlo

Lighting Example: Cornell Box

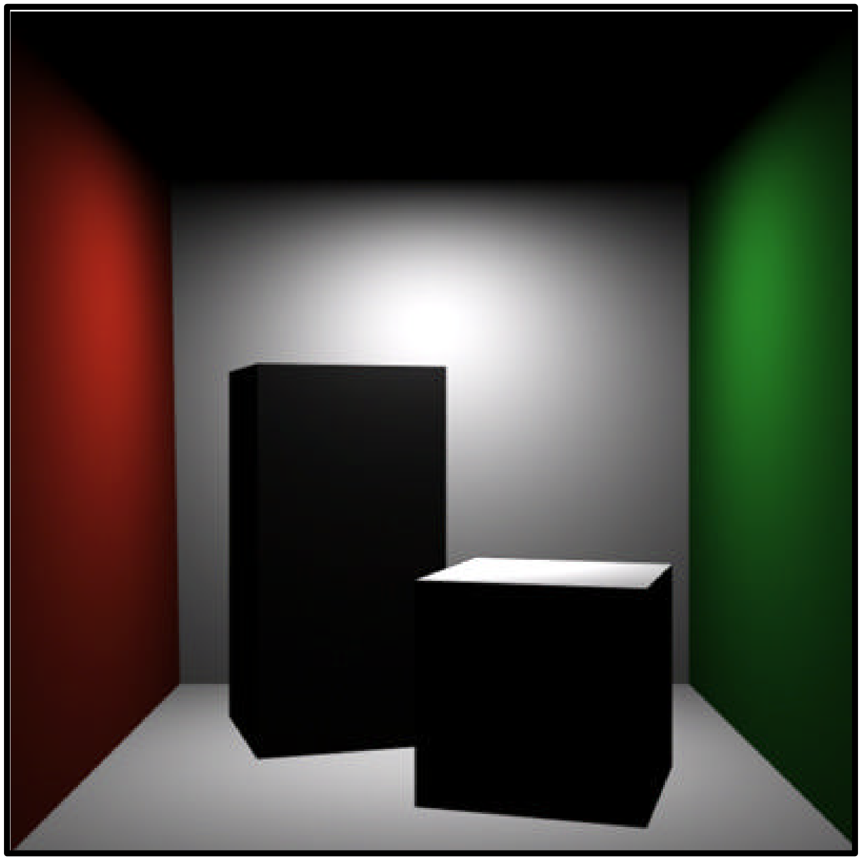


Surface Color

Lighting Example: Diffuse Reflection

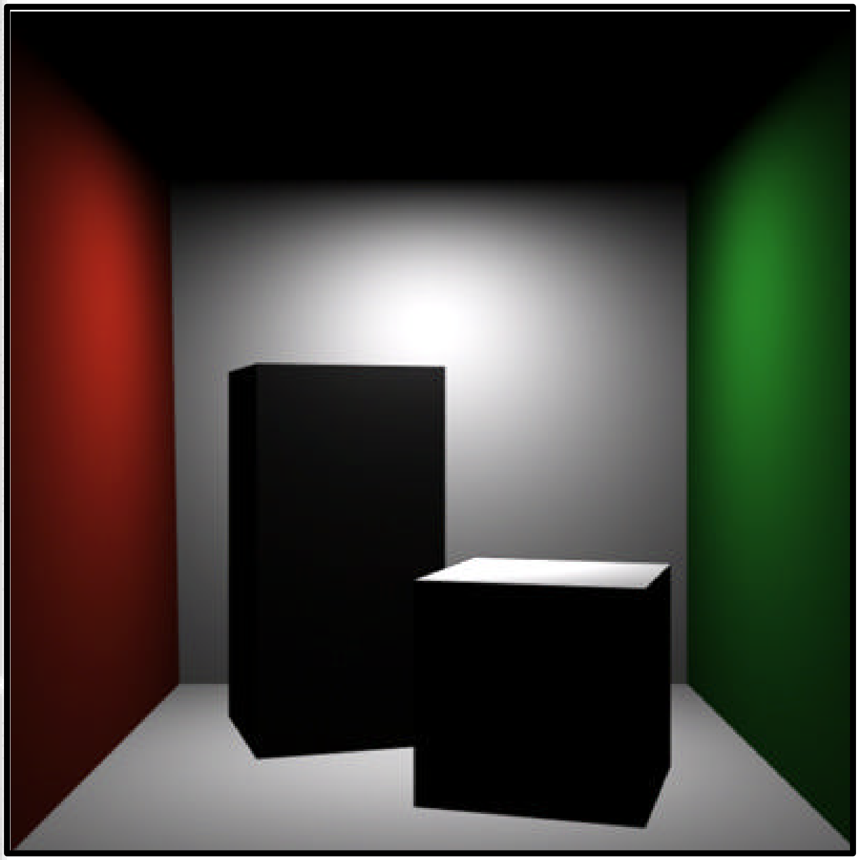


Surface Color

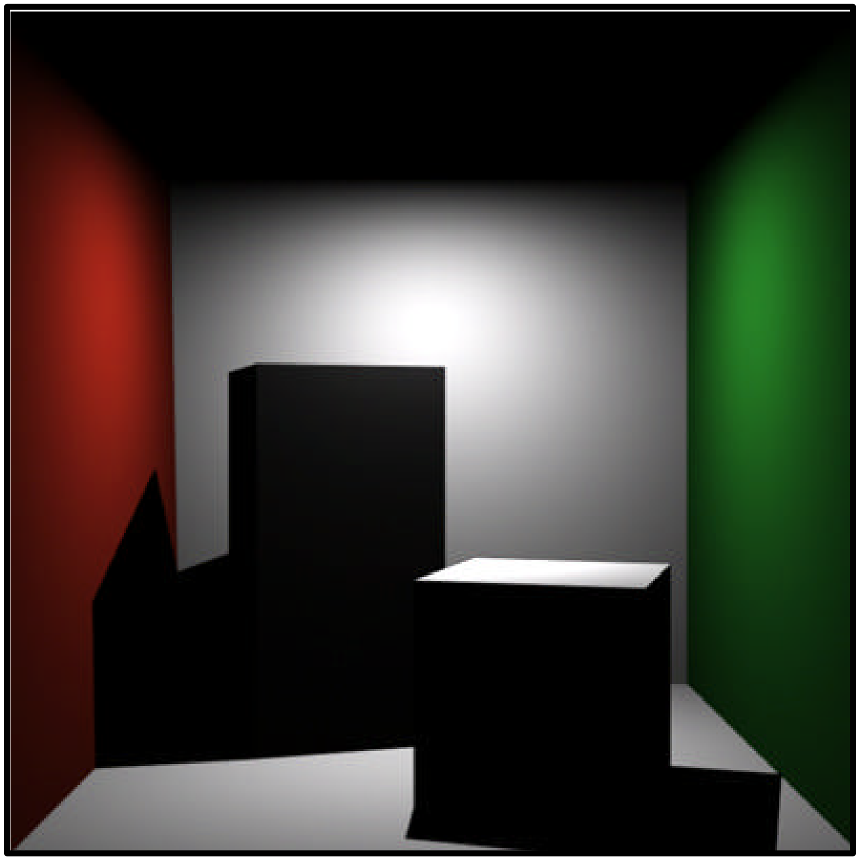


Diffuse Shading

Lighting Example: Shadows

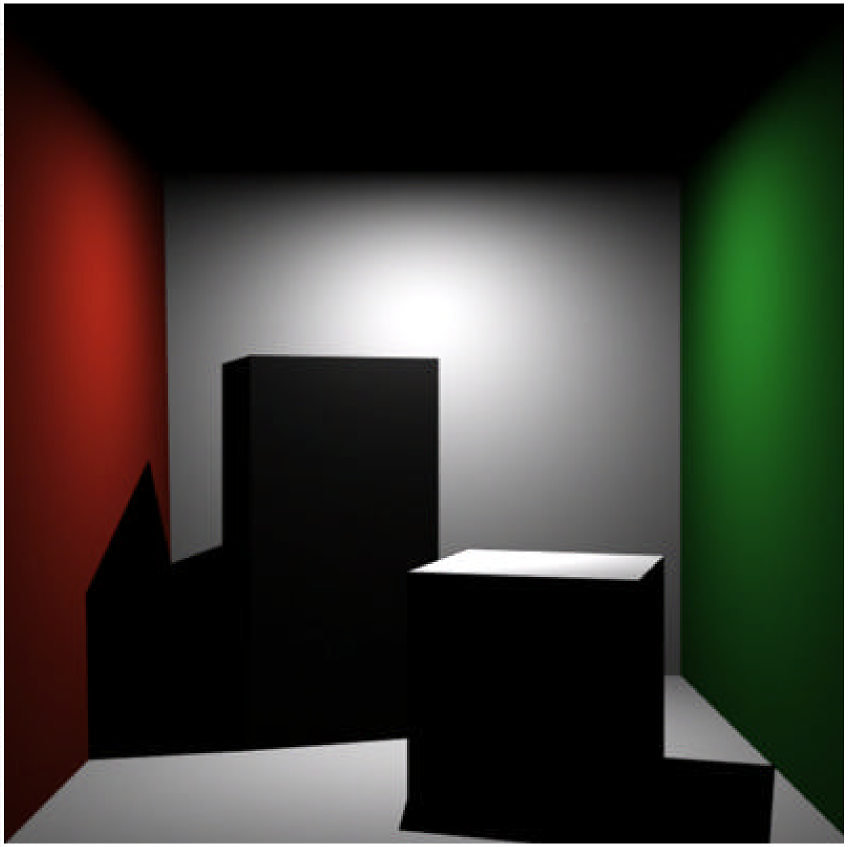


No Shadows

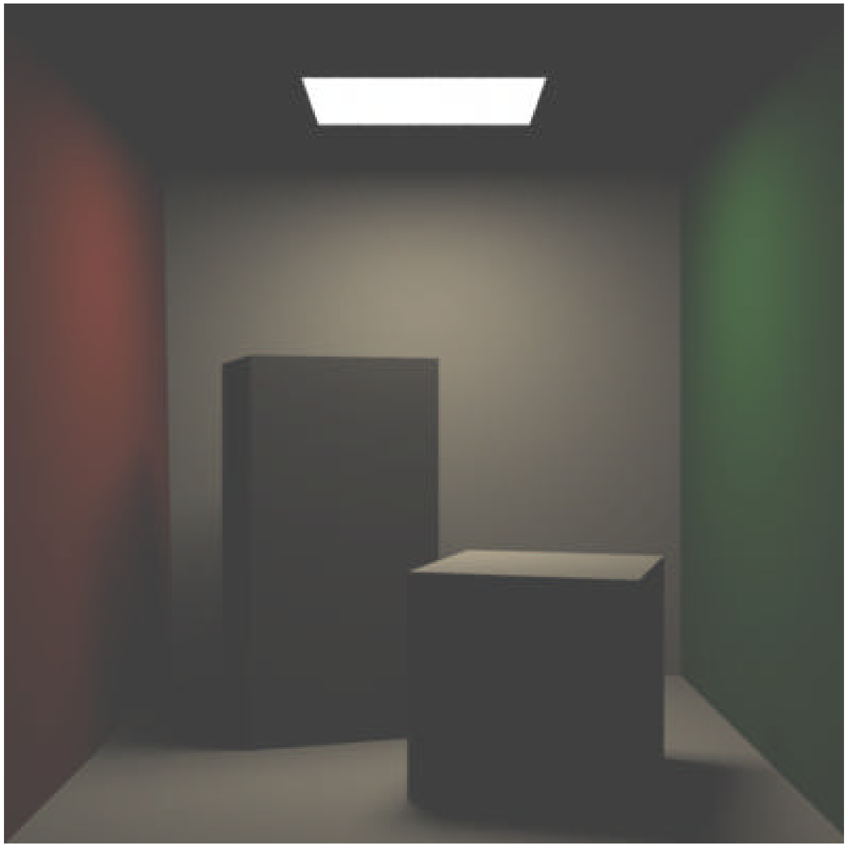


Shadows

Lighting Example: Soft Shadows



Hard Shadows
Point Light Source



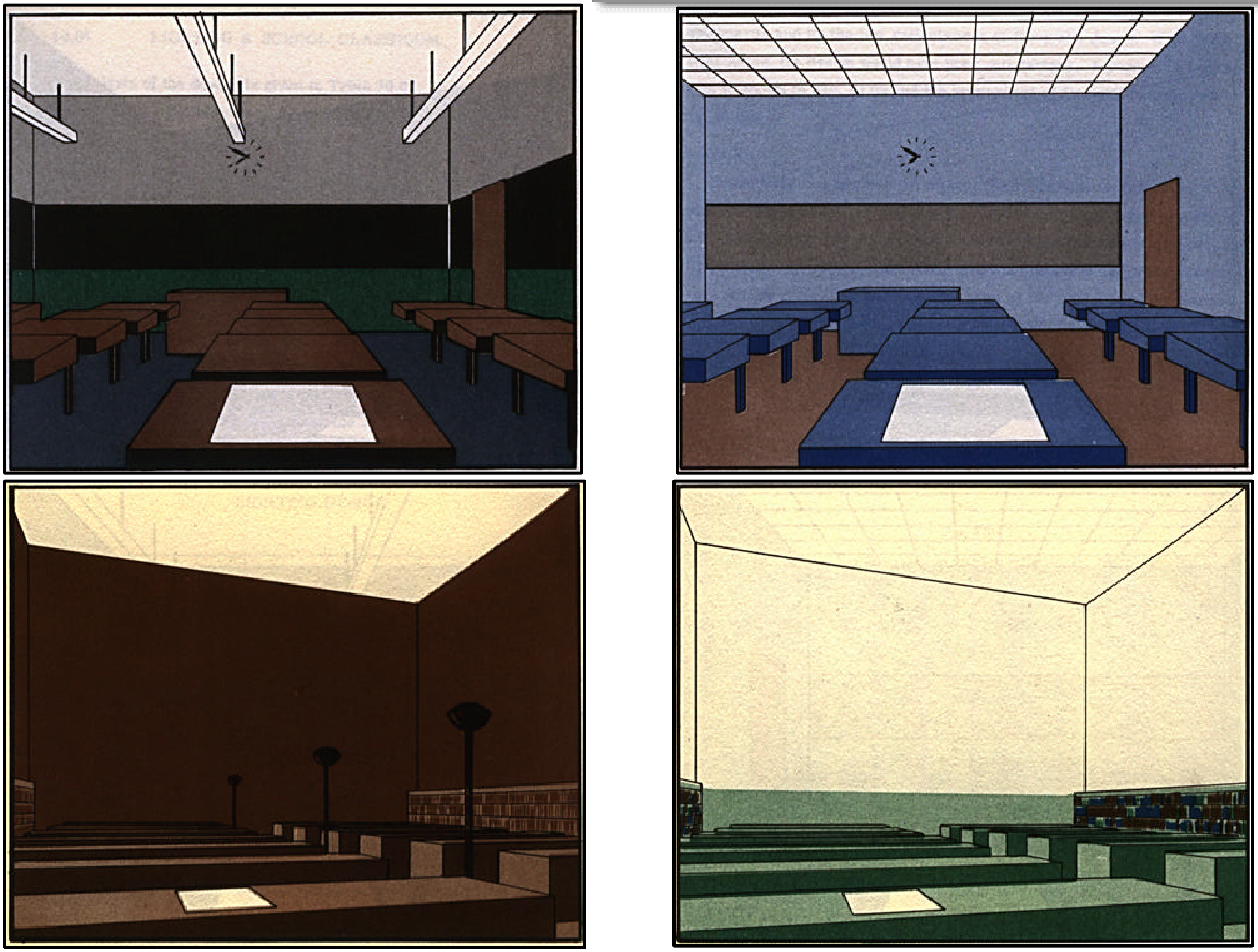
Soft Shadows
Area Light Source

Radiosity: Indirect Illumination

Program of Computer Graphics
Cornell University

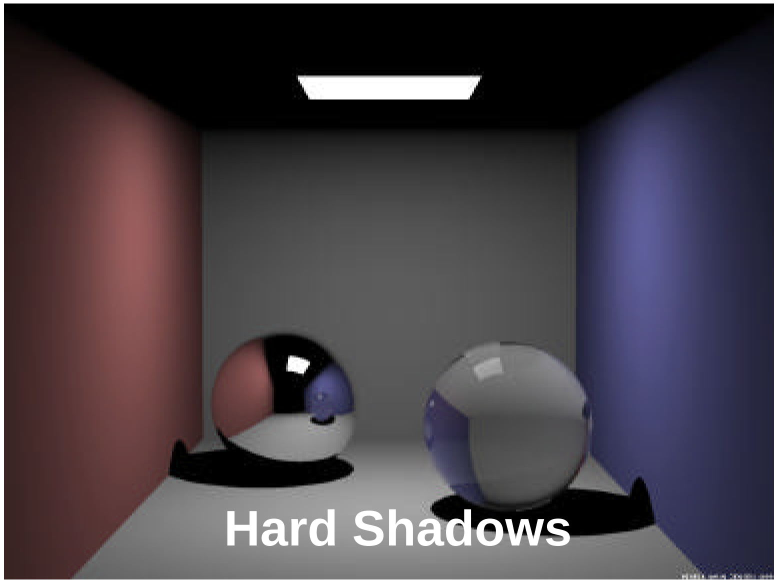
Early Radiosity

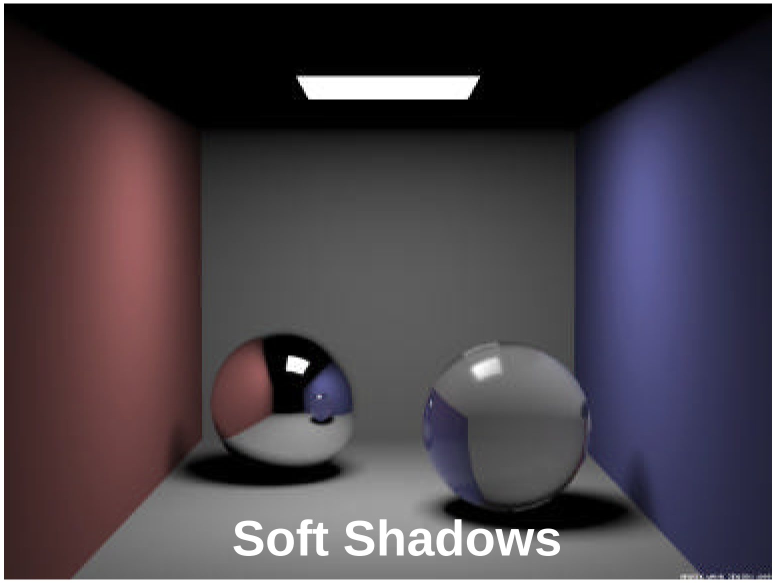
Early, Early Radiosity

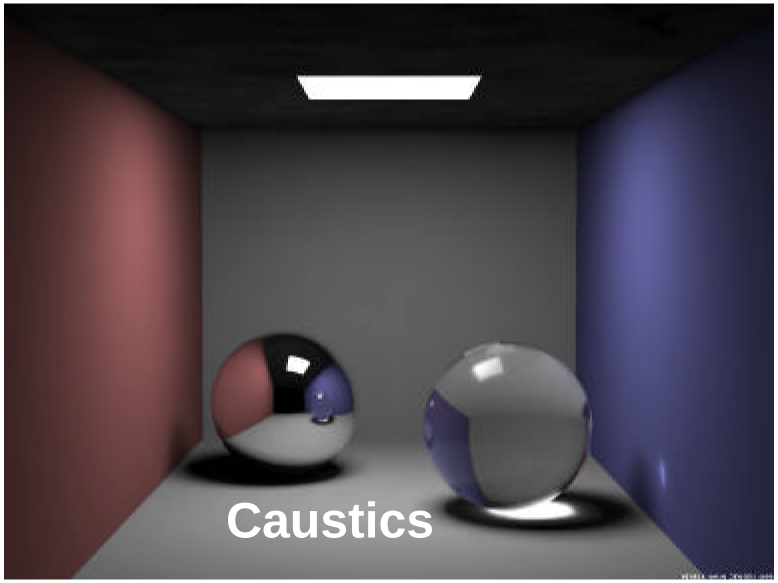


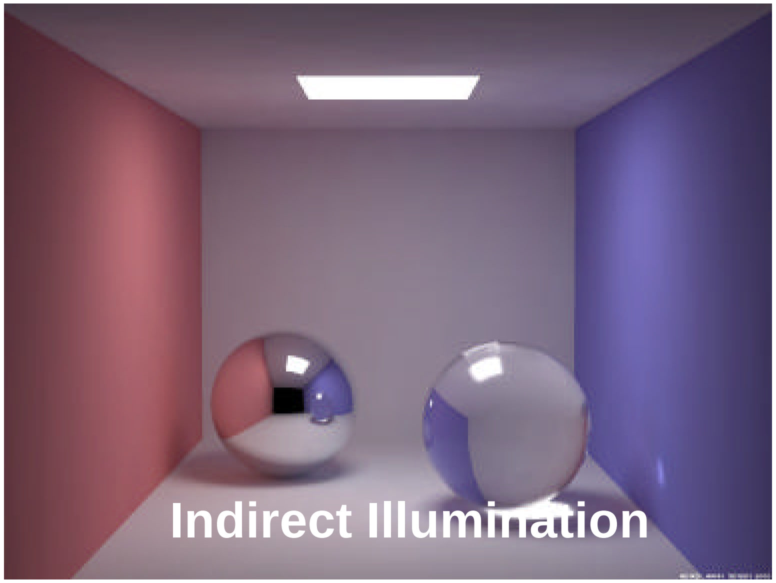
Parry Moon and Domina Spencer (MIT), Lighting Design, 1948

Lighting Effects: Glossy Materials

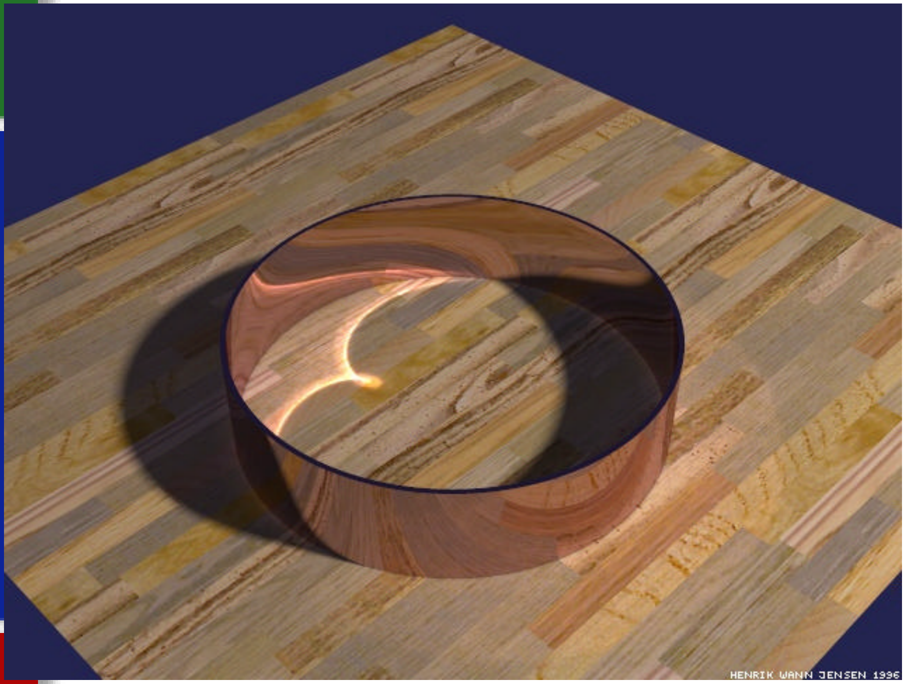

Hard Shadows


Soft Shadows


Caustics


Indirect Illumination

Caustics



Jensen 1995

Complex lighting

Complex Indirect Illumination

Mies Courtyard House with Curved Elements

Modeling: Stephen Duck; Rendering: Henrik Wann Jensen

Radiosity: “Turing Test”

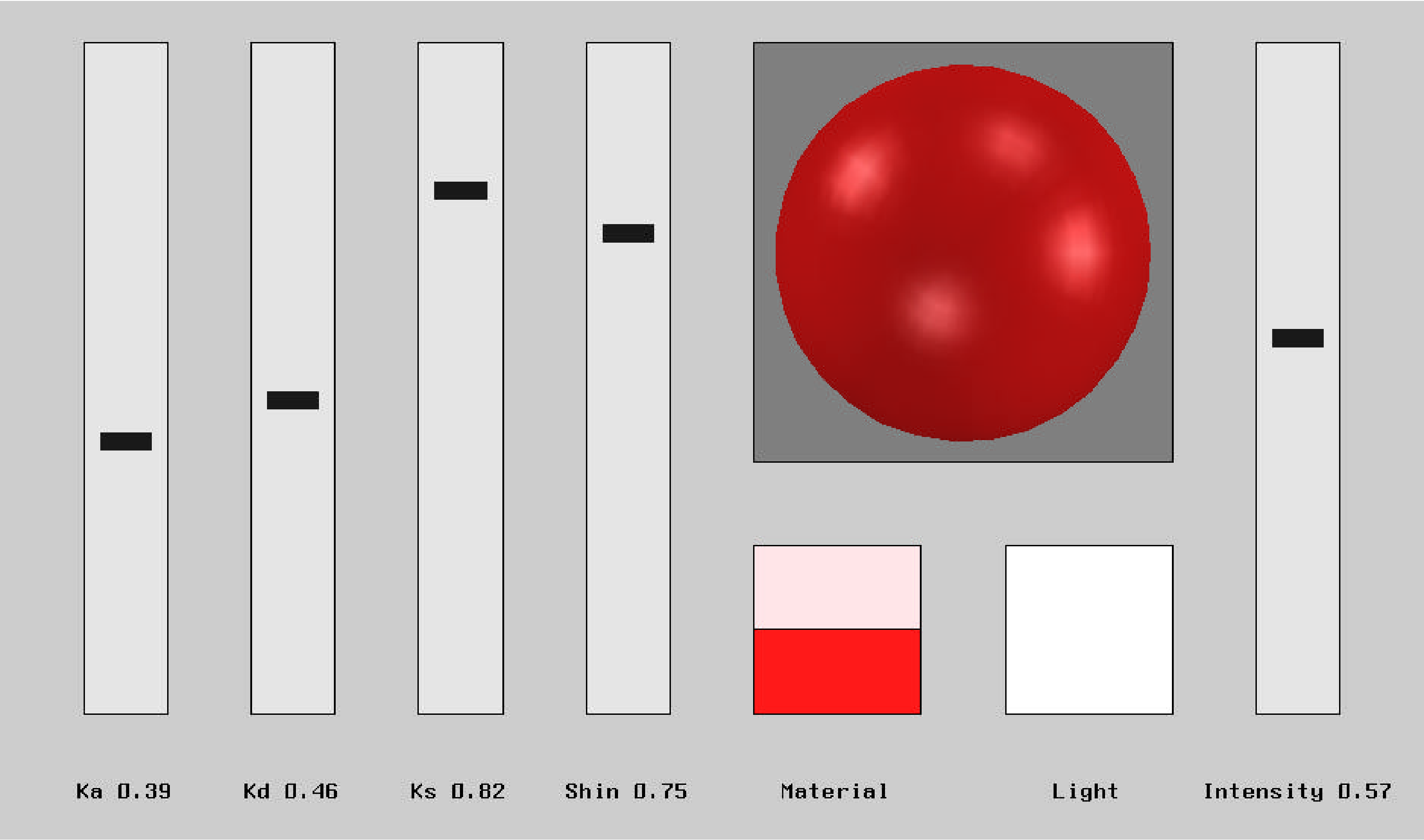
Measured

Simulated

Program of Computer Graphics
Cornell University

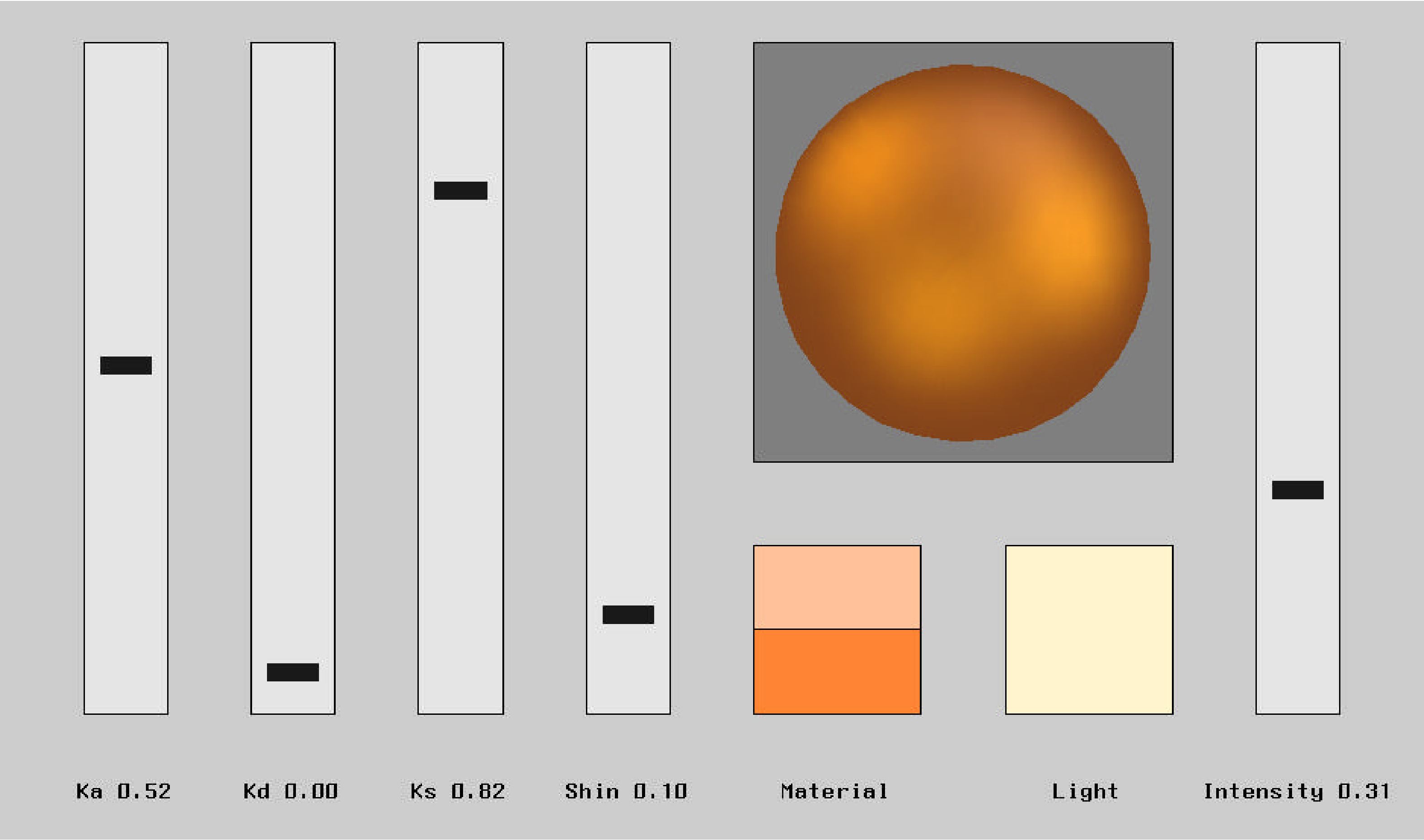
Materials

Classic Computer Graphics Model



Plastic

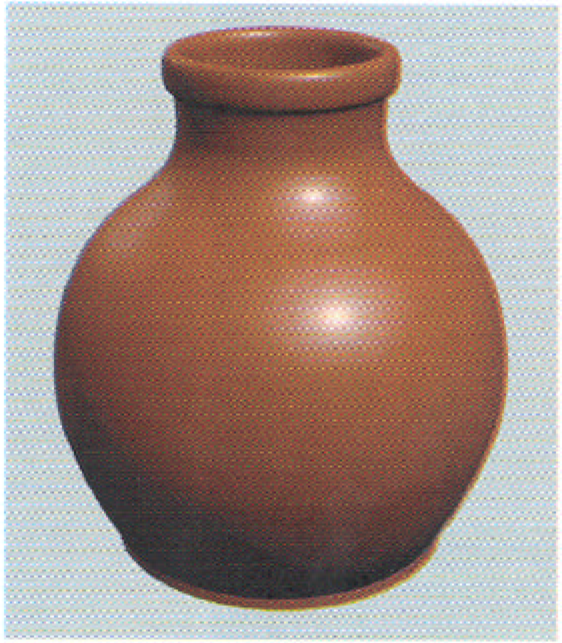
Classic Computer Graphics Model



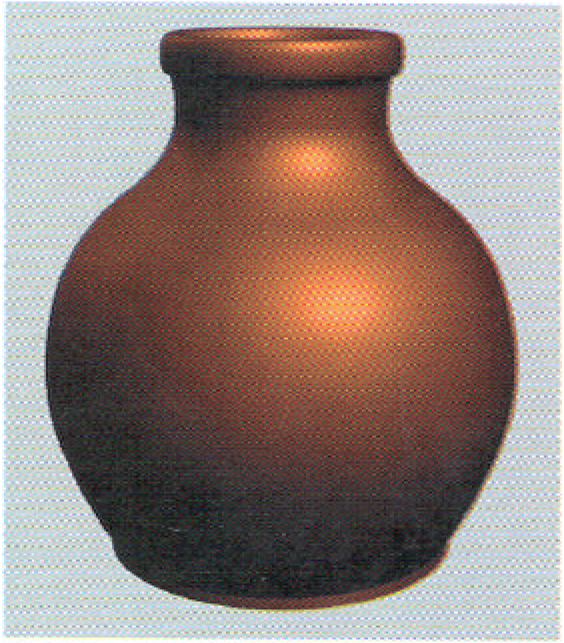
Brushed Copper

Material Taxonomy

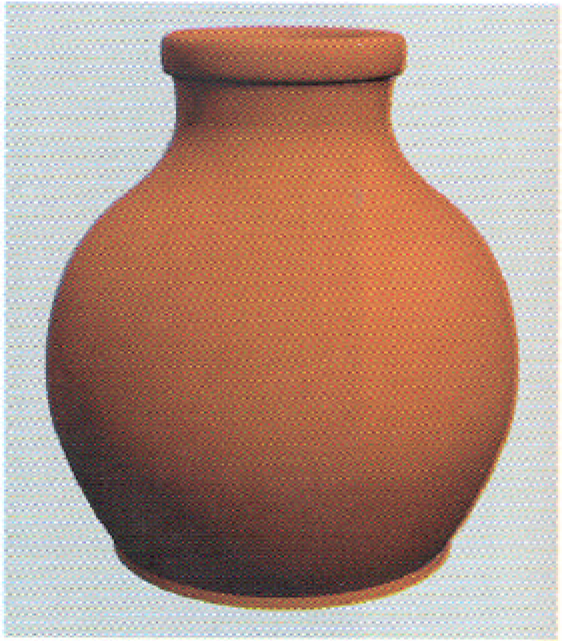
RenderMan



Plastic
Shiny Plastic



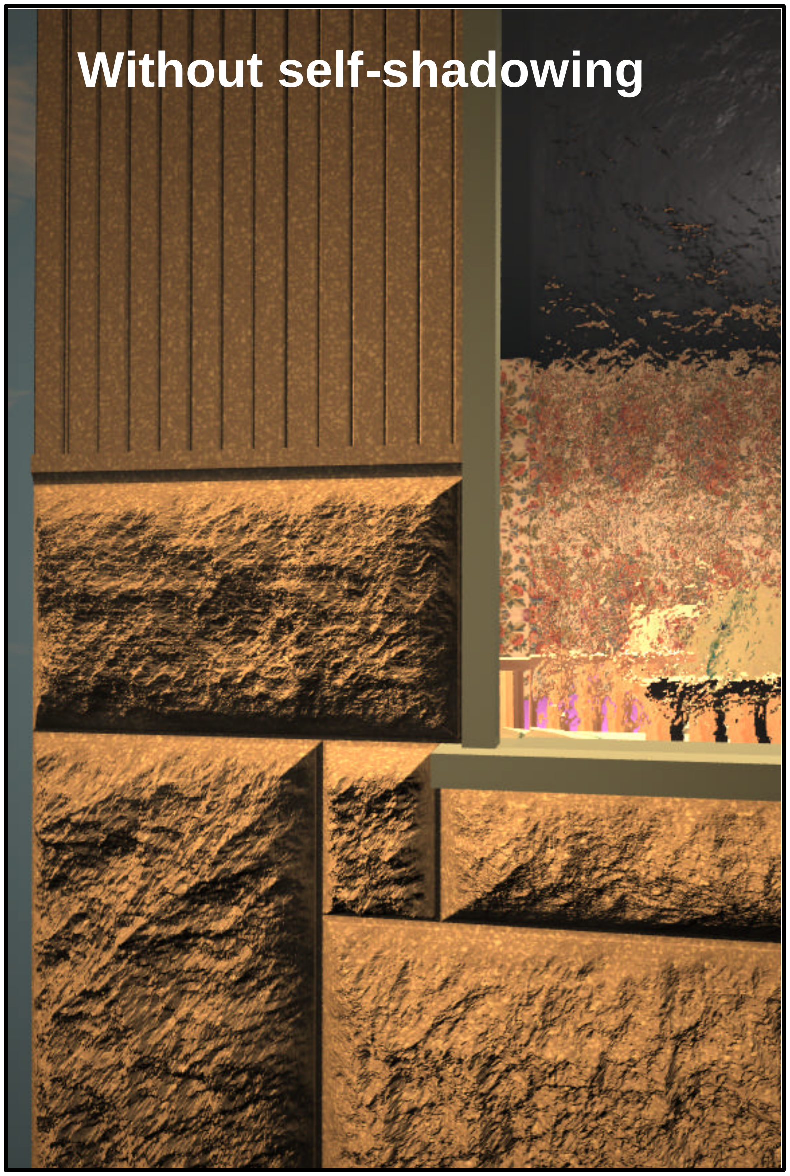
Rough Metal
Shiny Metal

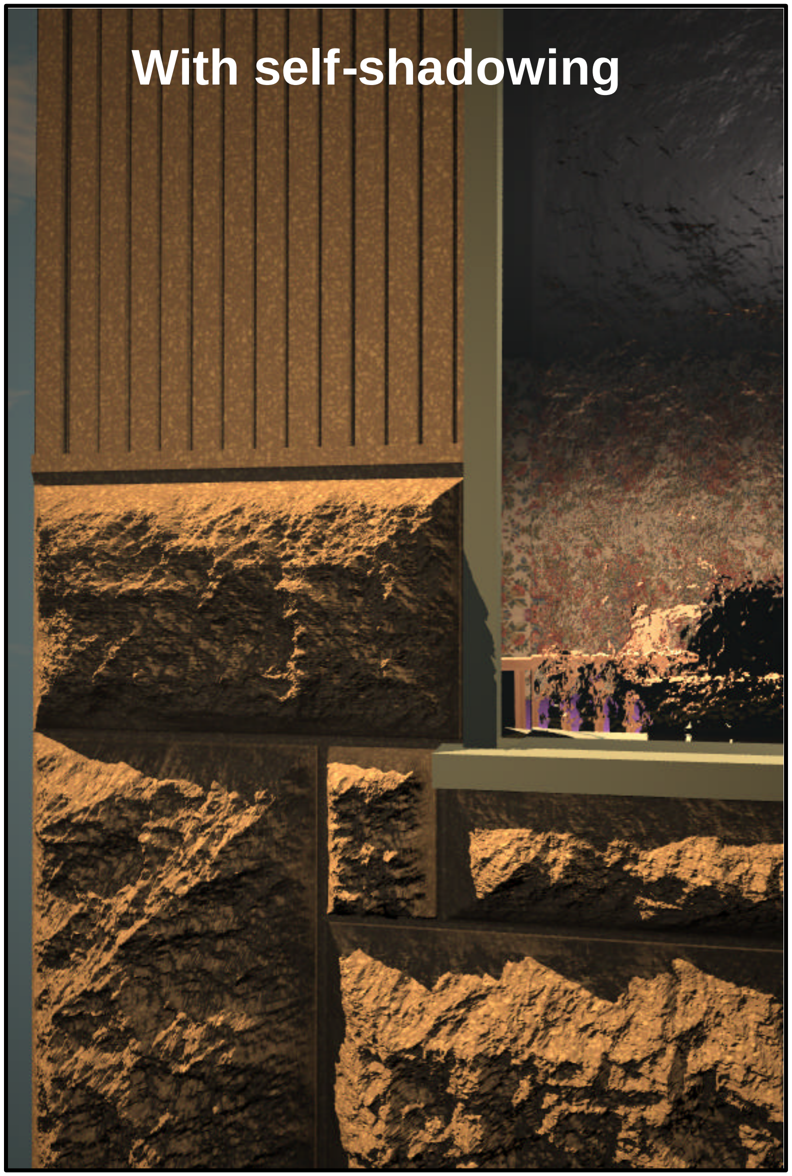


Matte

From Apodaca and Gritz, *Advanced RenderMan*

Shadows on Rough Surfaces





Translucency

Surface Reflection

Subsurface Reflection

Translucent objects

Water Flows on the Venus

Patinas

A Sense of TIme

Virtual Actors: Faces

Square USA

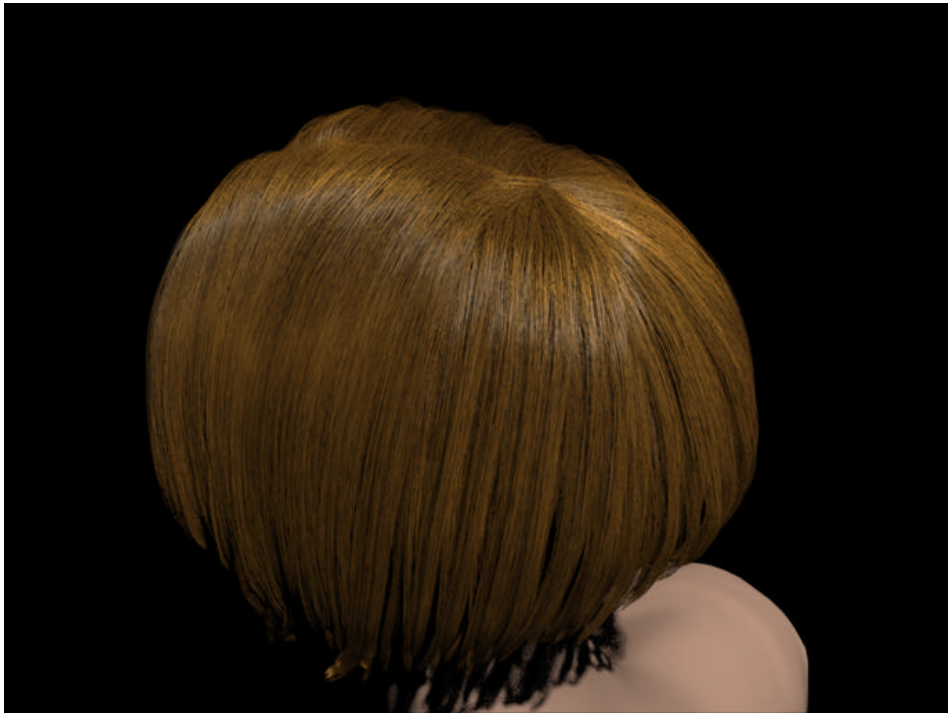
The digital heroine of the Final Fantasy film.

Final Fantasy
SquareUSA

Jensen,
Marschner,
Levoy,
Hanrahan

Virtual Actors: Hair

Black

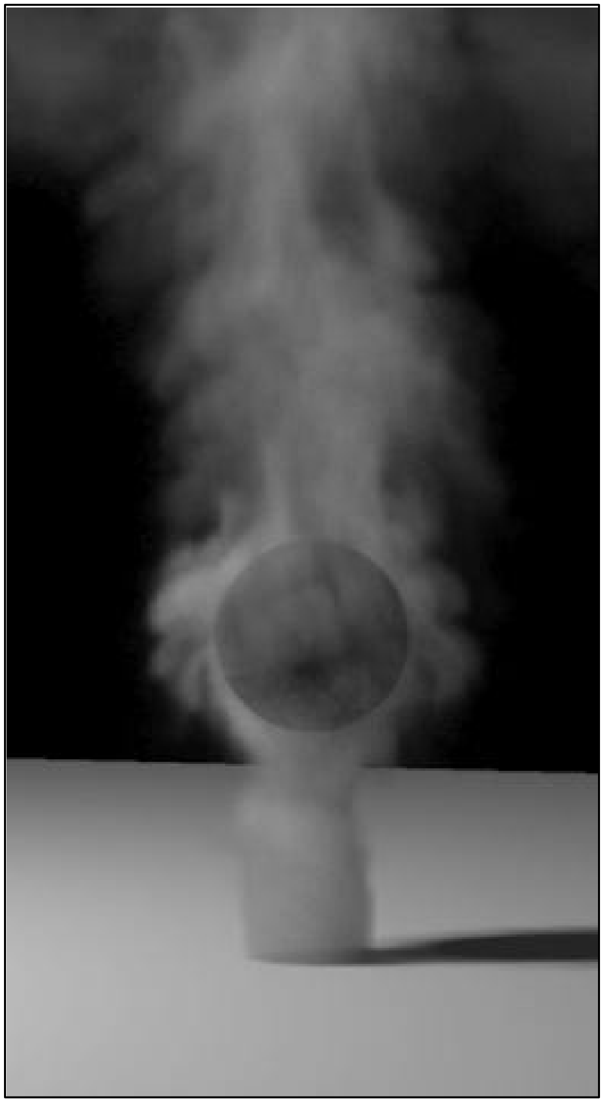


Brown

Refraction / dispersion

- Iridescent: Wavelength-dependent phenomena

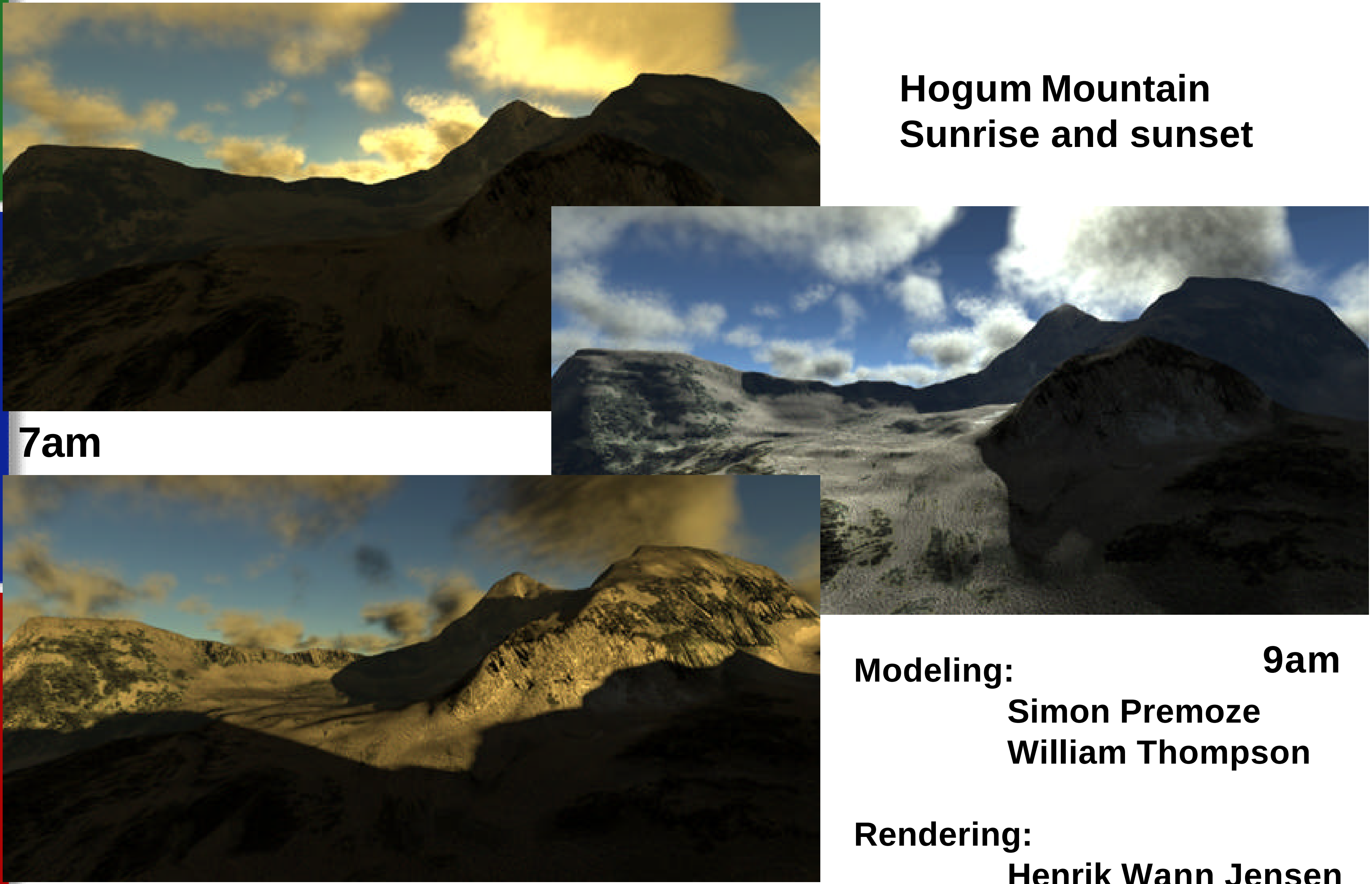
Coupling Modeling & Rendering



Fedkiw, Stam, Jensen 2001

Clouds and Atmospheric Phenomena

Hogum Mountain
Sunrise and sunset



7am

9am

6:30pm

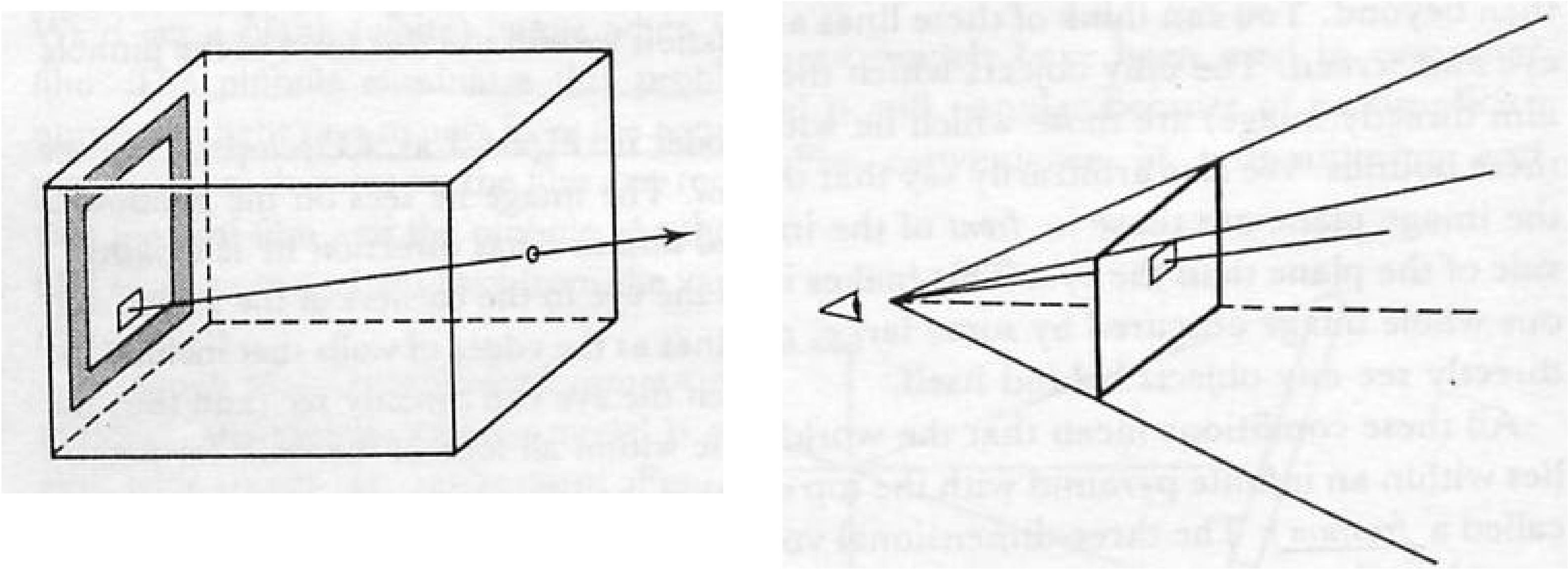
Modeling:
Simon Premoze
William Thompson

Rendering:
Henrik Wann Jensen

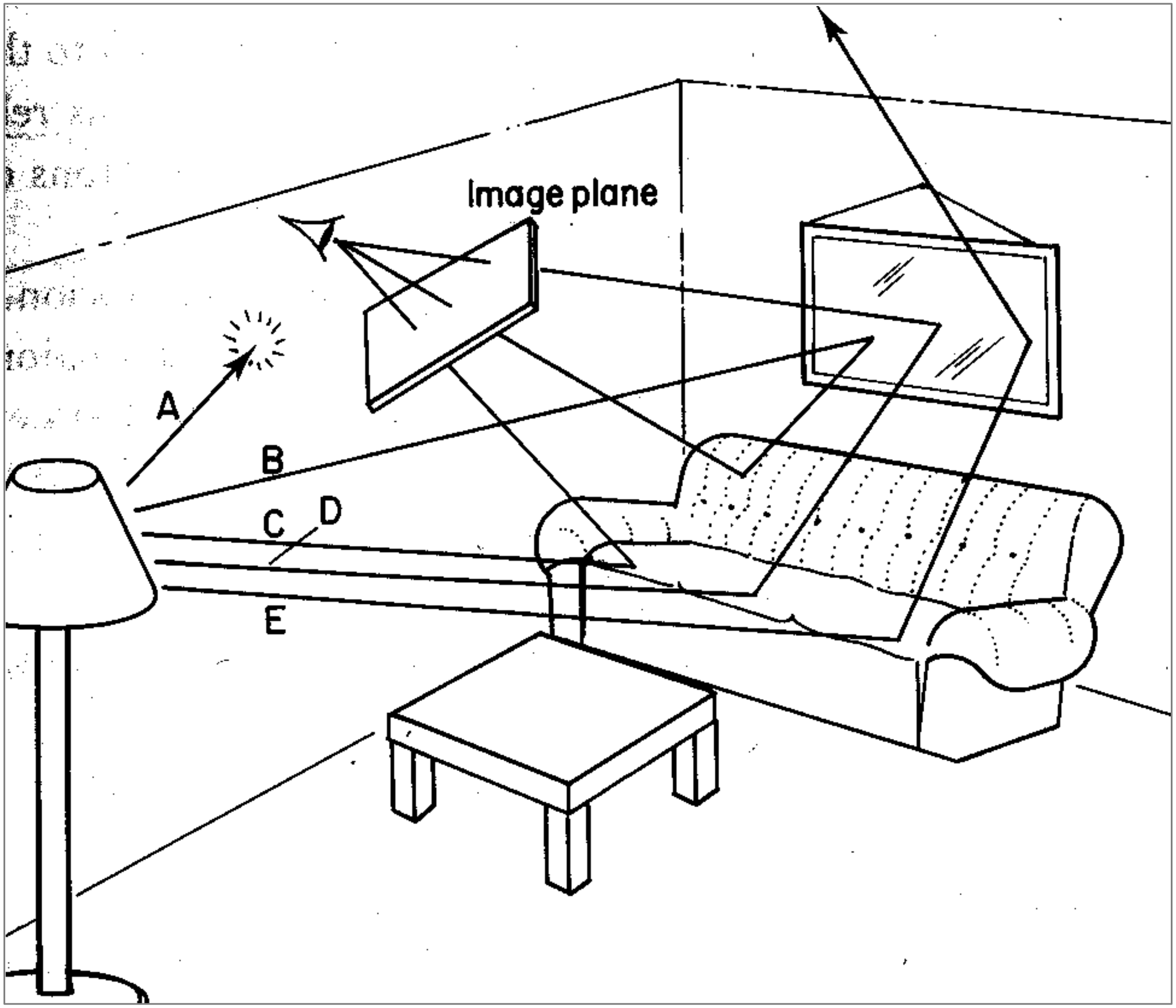
Vegetation

Texture and complex materials

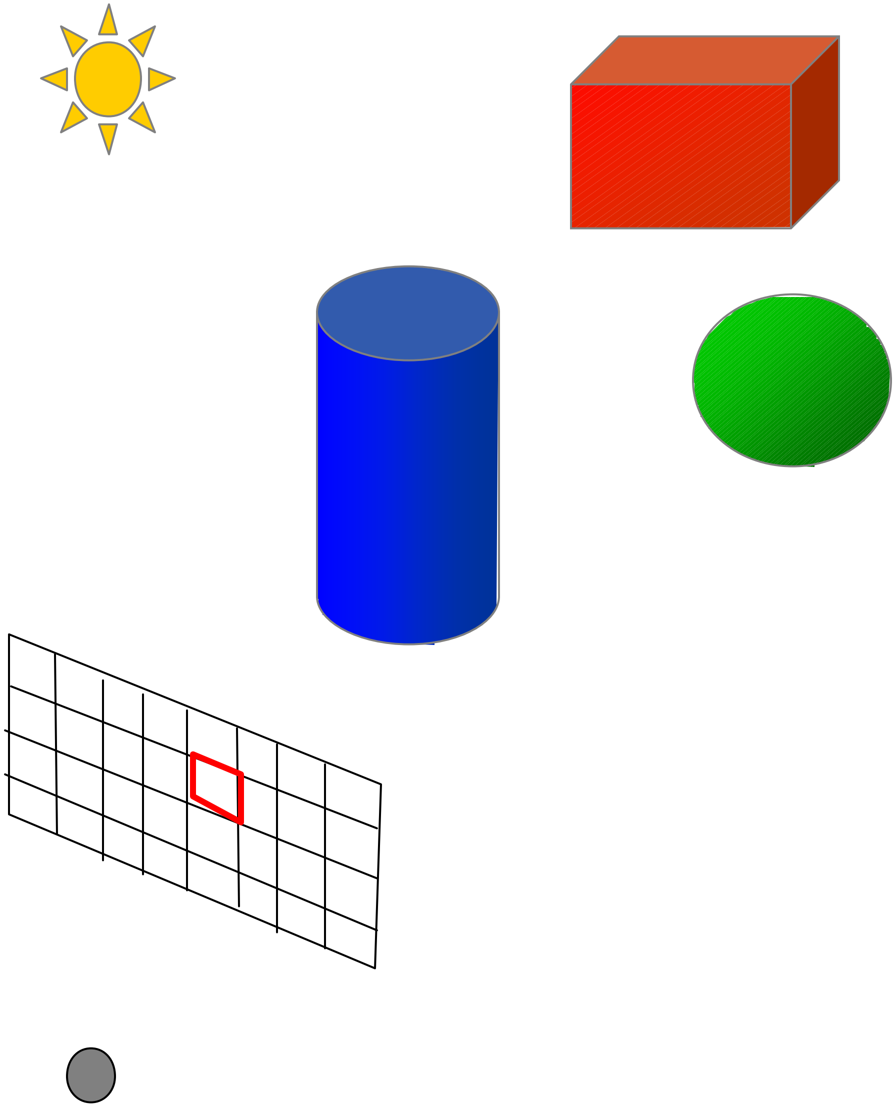
Pinhole camera



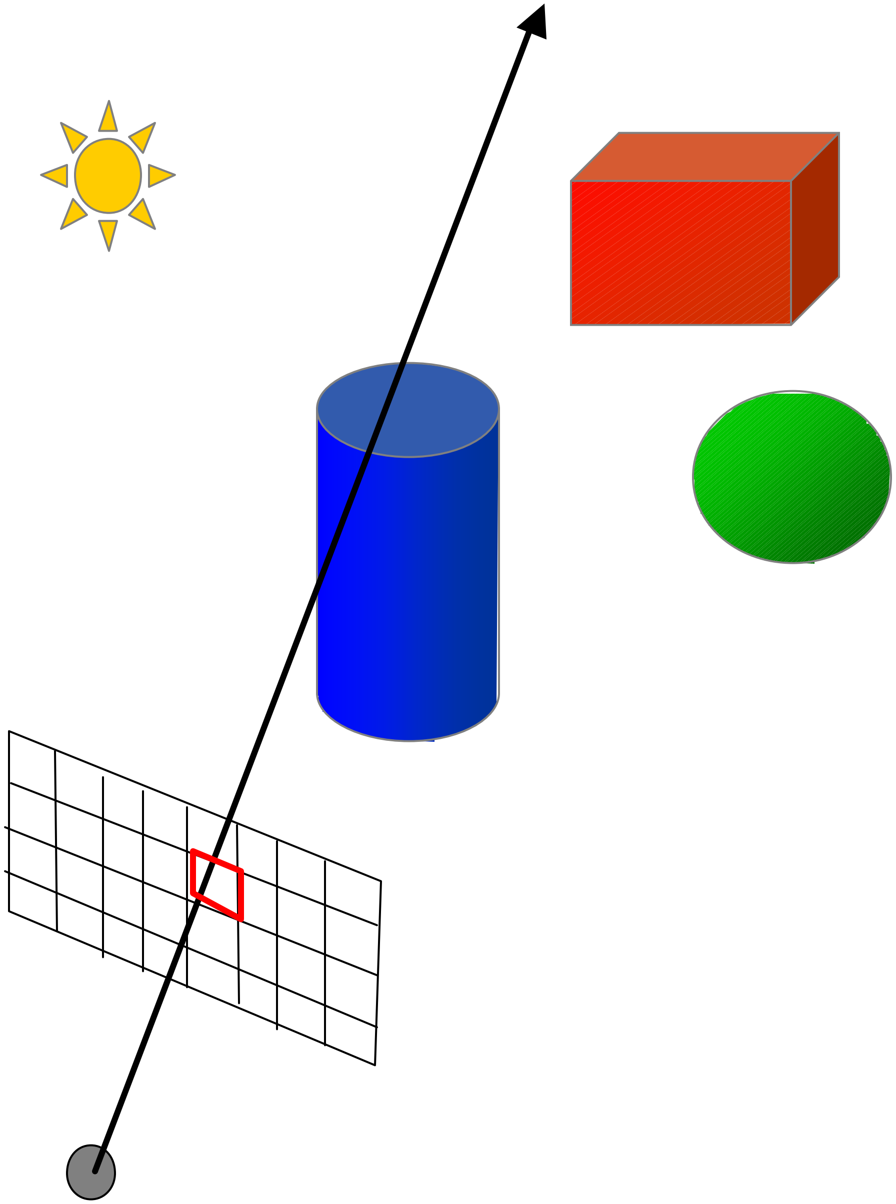
Introduction to ray tracing



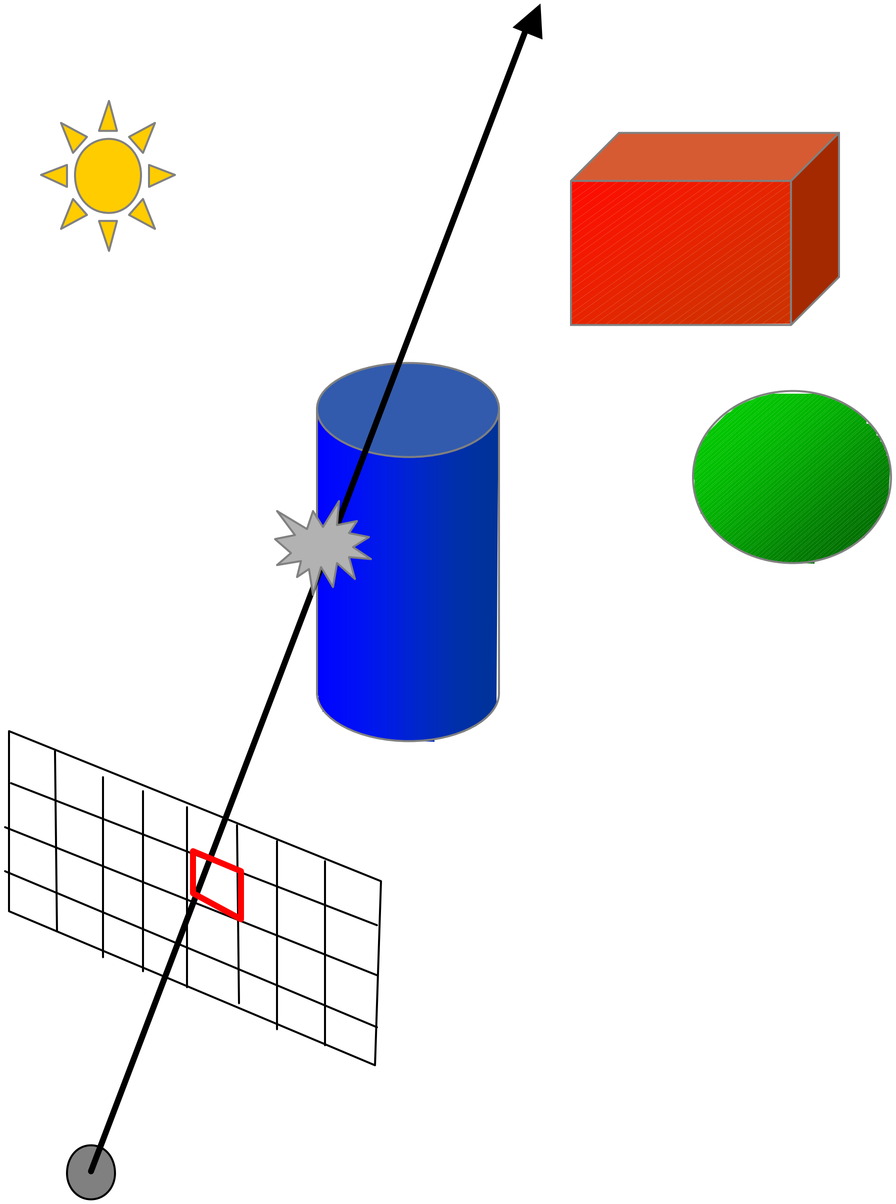
Ray Casting (Appel, 1968)



Ray Casting (Appel, 1968)

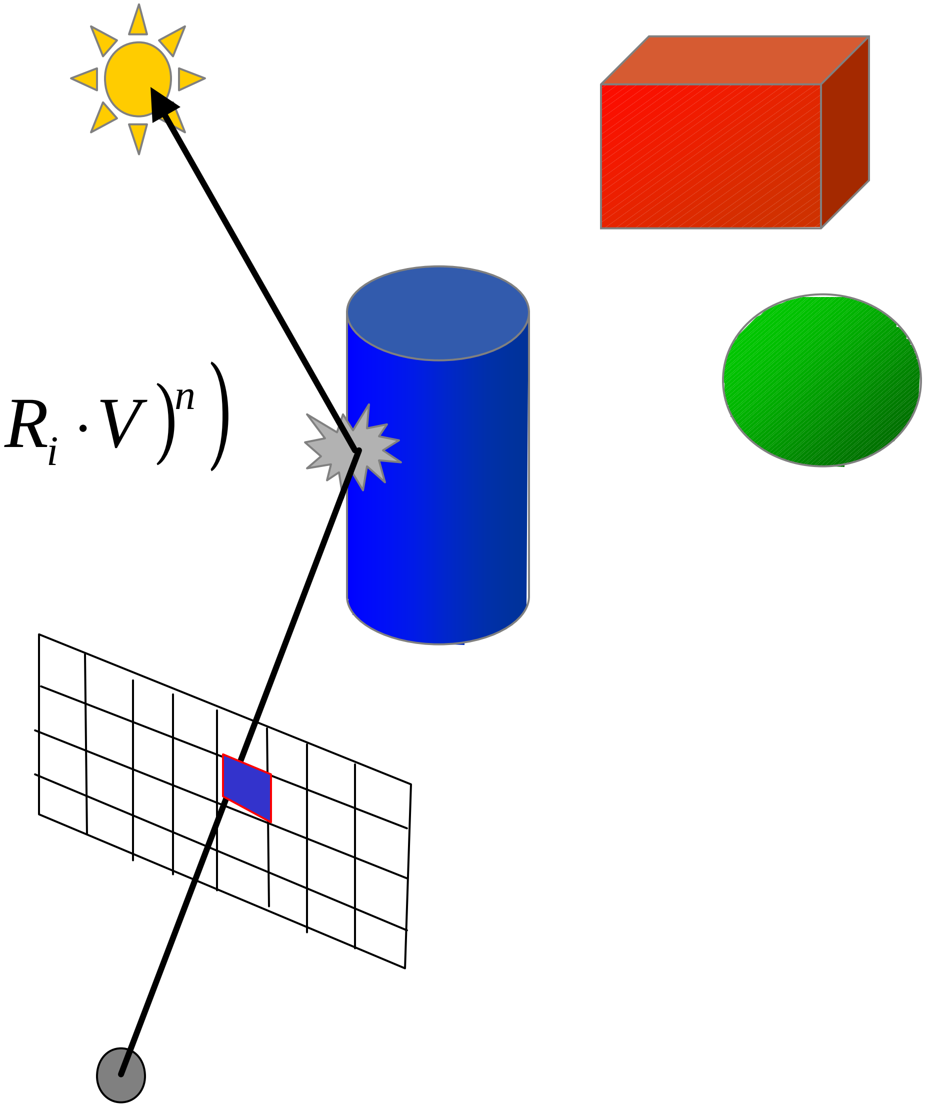


Ray Casting (Appel, 1968)

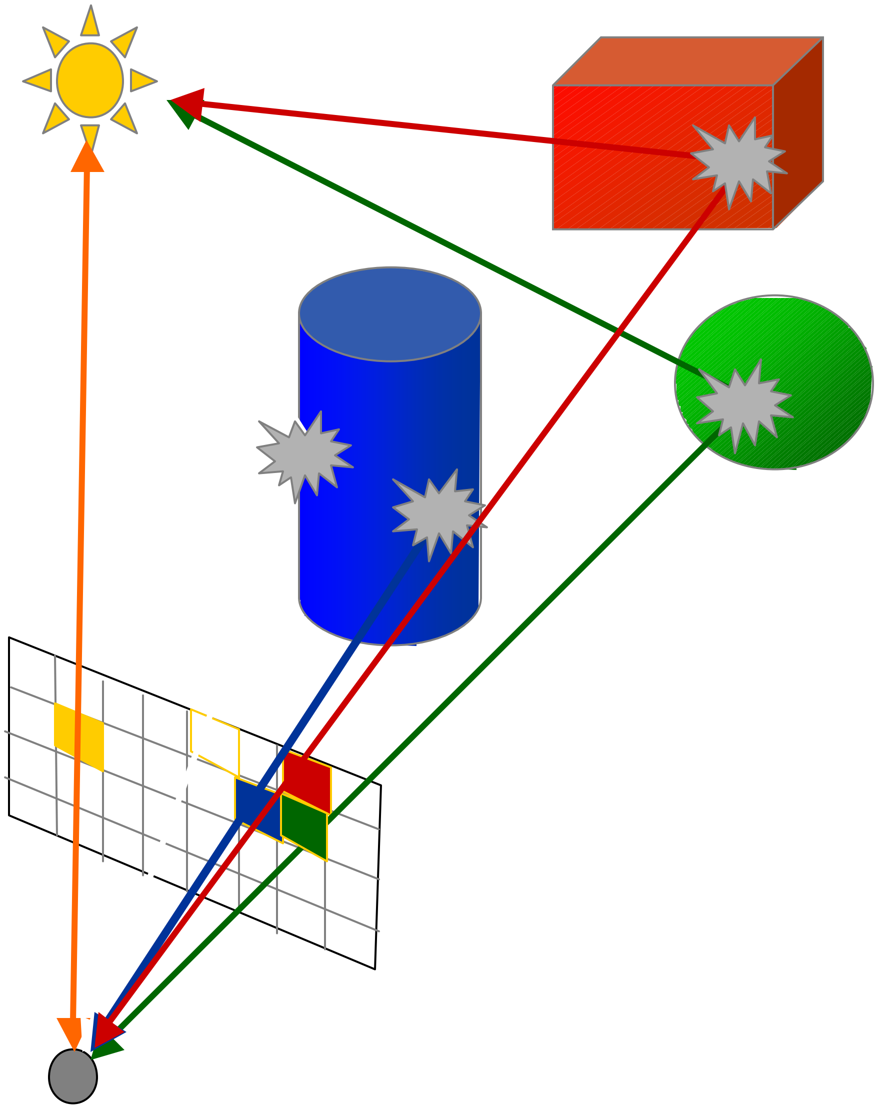


Ray Casting (Appel, 1968)

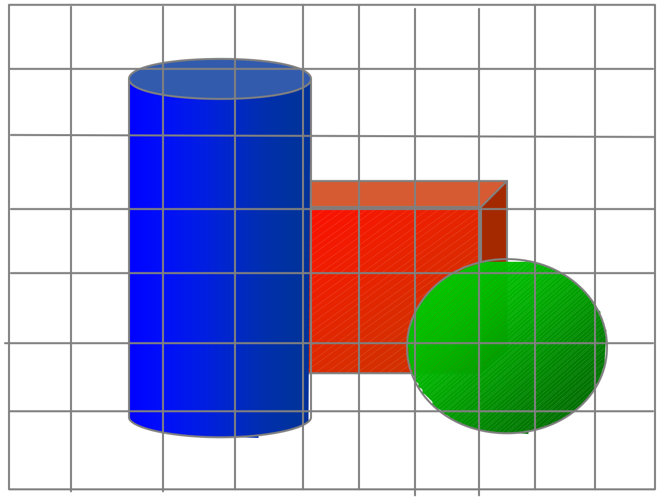
$$k_a I_a + \sum_{i=1}^{nls} I_i \left(k_d \left(L_i \cdot N \right) + k_s \left(R_i \cdot V \right)^n \right)$$



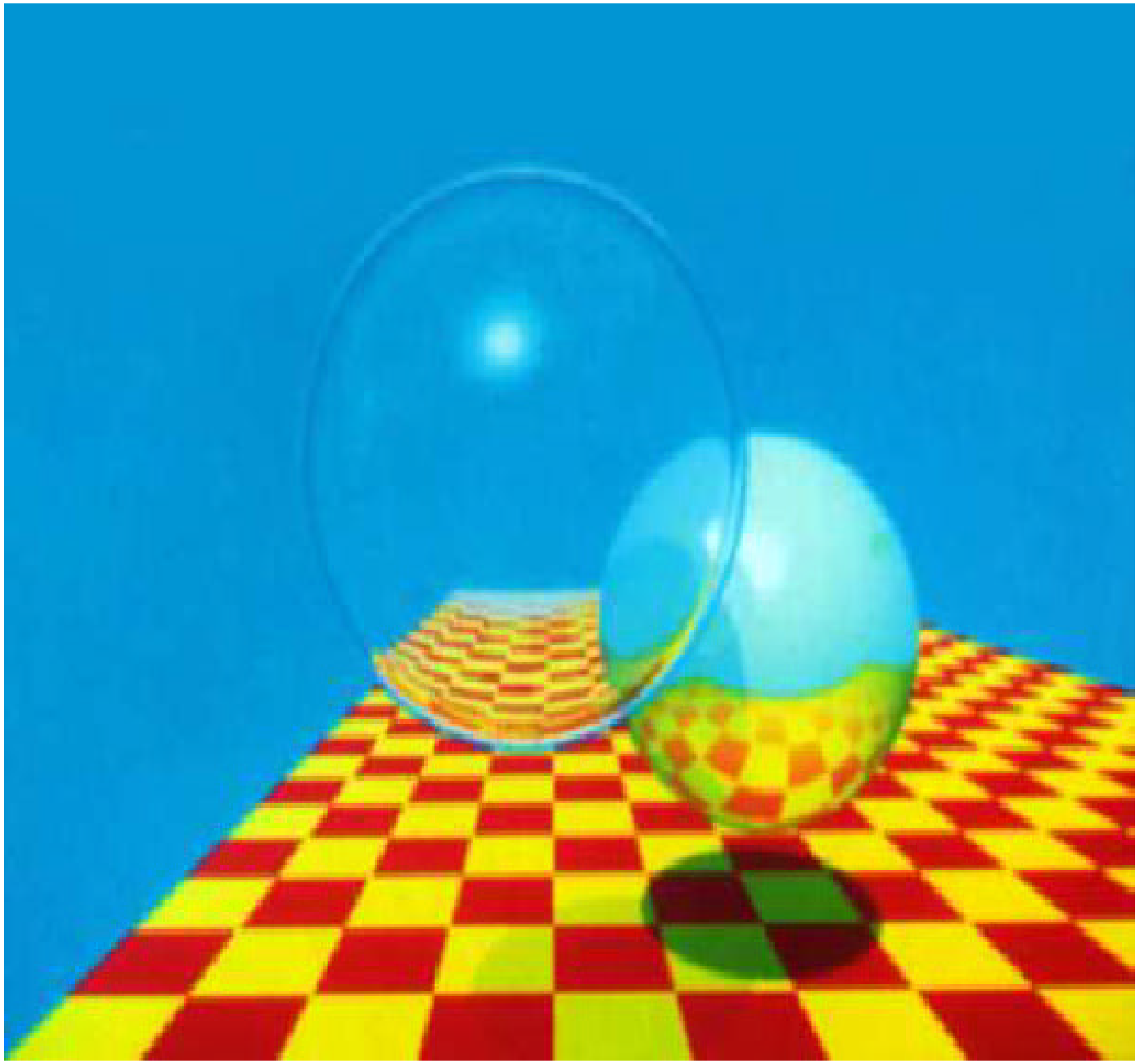
Ray Casting (Appel, 1968)



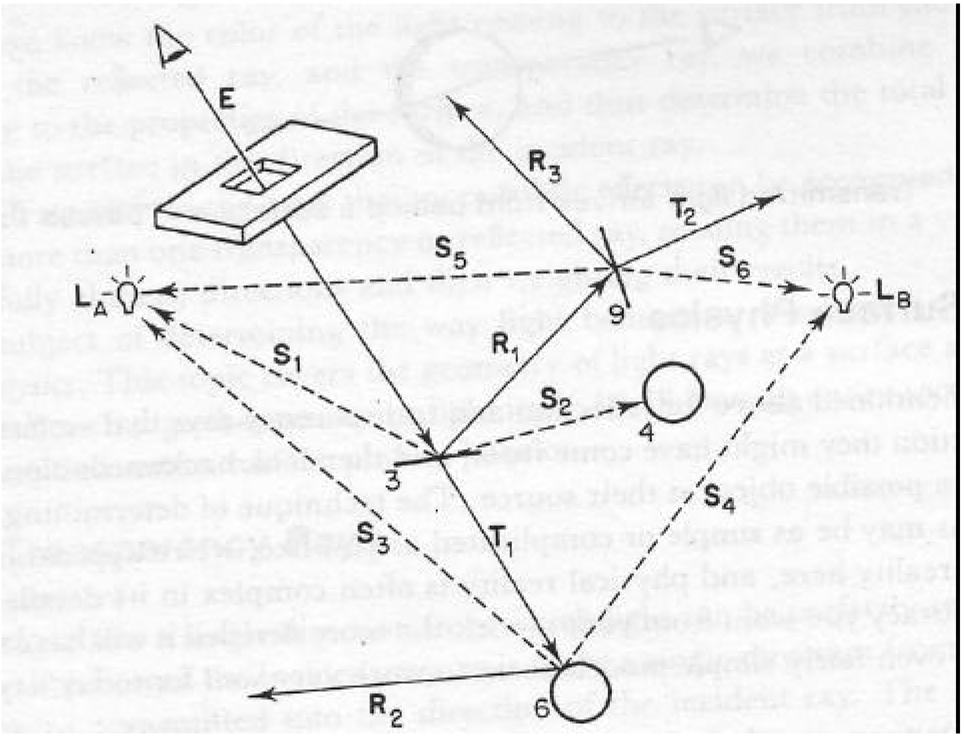
direct illumination

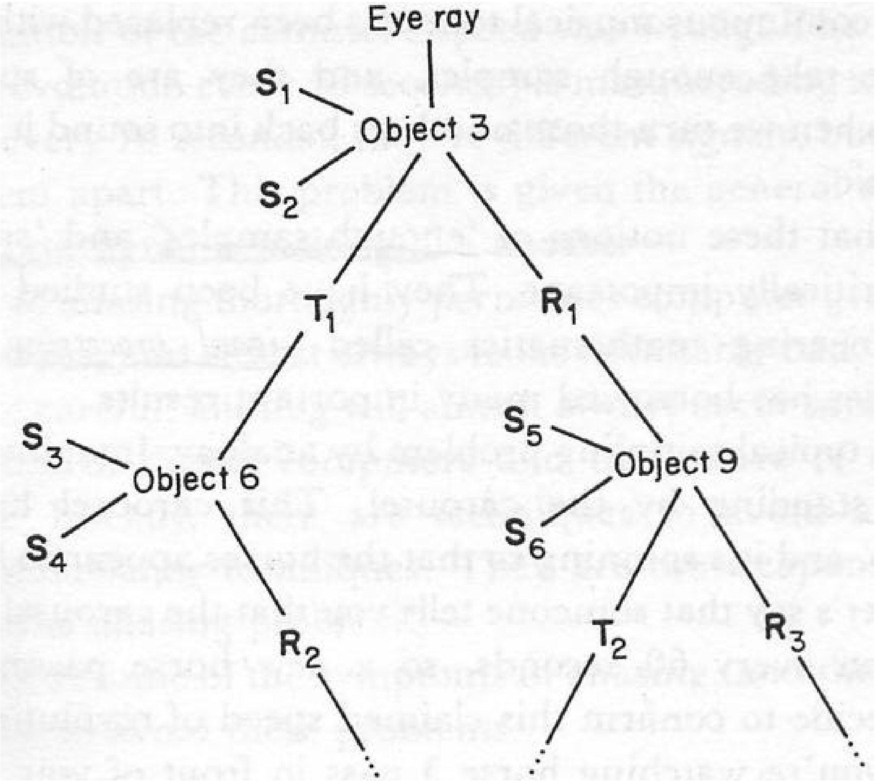


Recursive ray tracing (Whitted, 1980)



- Recursive ray tracing creates tree of rays





Ray tracer components

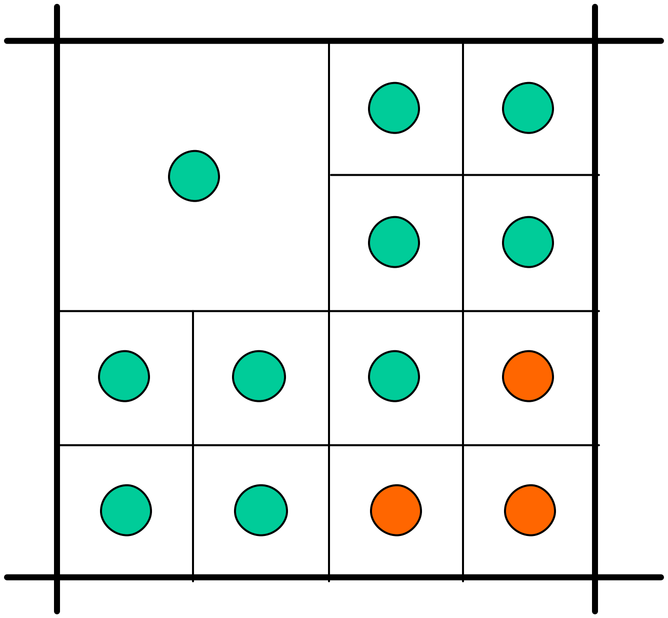
- Cameras
- Films
- Lights
- Ray-object intersection
- Visibility
- Surface scattering
- Recursive ray tracing

Why Ray Tracing Looks Fake / Effects

- Jagged edges
- Hard shadows
- Everything in focus
- Objects completely still
- Surfaces perfectly shiny
- Glass perfectly clear

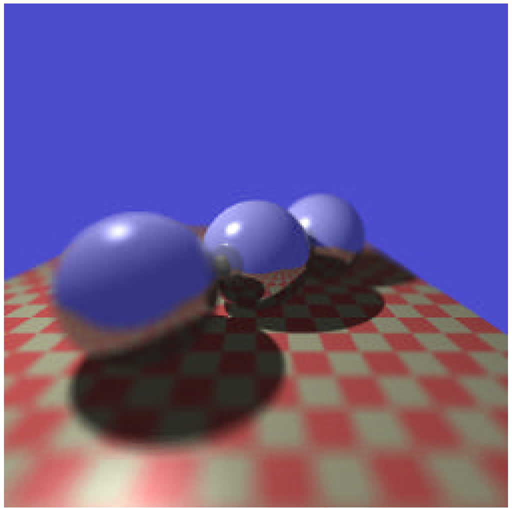
Why Ray Tracing Looks Fake

- Distributed Ray Tracing
  - Rob Cook, SIGGRAPH 84
  - Replace single ray with distribution of rays
  - Not just fat ray through pixel, but fat rays everywhere
  - Cast Multiple
    - Eye rays
    - Shadow rays
    - Reflection rays
    - Refraction rays
- Supersampling
  - Cast multiple rays from eye through different parts of same pixel



Why Ray Tracing Looks Fake

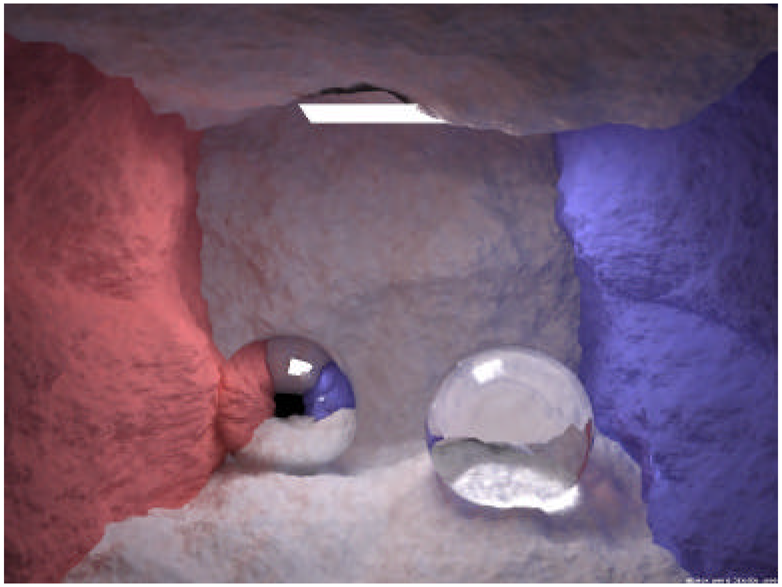
- Motion blur
  - Cast multiple rays from eye through same point in each pixel
  - Each of these rays intersects the scene at a different time
  - Reconstruction filter controls shutter speed, length



- Depth of Field
  - Better simulation of camera model
    - f-stop
    - focus
- Others (soft shadow, glossy, etc)

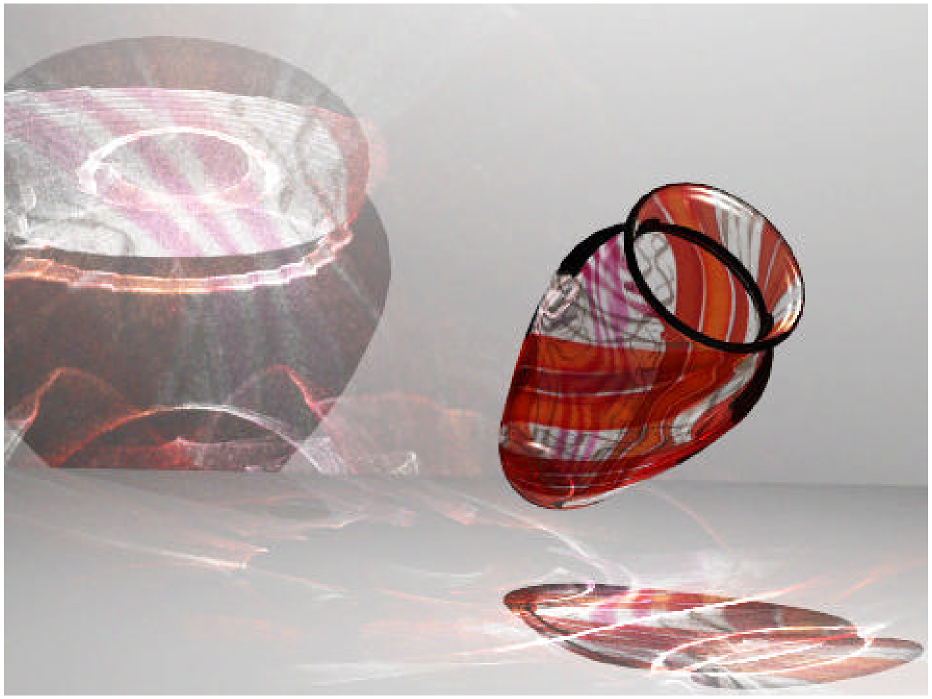
Photon Mapping

- Jensen EGRW 95, 96
- Simulates the transport of individual photons
- Two parts. First
  - Photons emitted from source
  - Photons deposited on surfaces
- Secondly:
  - Photons reflected from surfaces to other surfaces
  - Photons collected by rendering
- Good for:
  - Light through water
  - Cloud illumination
  - Marble



Rendering Techniques

- Photon mapping examples



Images: courtesy of Stanford rendering contest

Professor Background

- Dr. Emmanuel Agu (professor, “Emmanuel”)
- Research areas
    - Computer Graphics (appearance modeling, etc)
    - Mobile Computing (mobile graphics), wireless networks
- Research opportunities
    - MQP
    - MS theses
    - PhD theses

Course Prerequisites

- No official prerequisite
- However, will assume you
  - Can program in C++
  - Have basic knowledge of data structures and algorithms
  - Have taken at least one graphics class (4731, 543)
  - can understand text, graphics papers (book gives good coverage + Discussions in class)
  - Can fill in gaps (extra work) if required
  - Linear algebra, probability, compilers
  - Can learn and use rendering package (Maya, Studio Max)
- Questions? See me

Syllabus

- http://www.cs.wpi.edu/~emmanuel/courses/cs563/
- Office hours:
  - Monday: 3:00-4:00 Thursday: 3:00-4:00
  - Note: Please use office hours or book appointments first
- Important: All questions on myWPI
- Email to make appointment or ask questions specific to you

Textbook

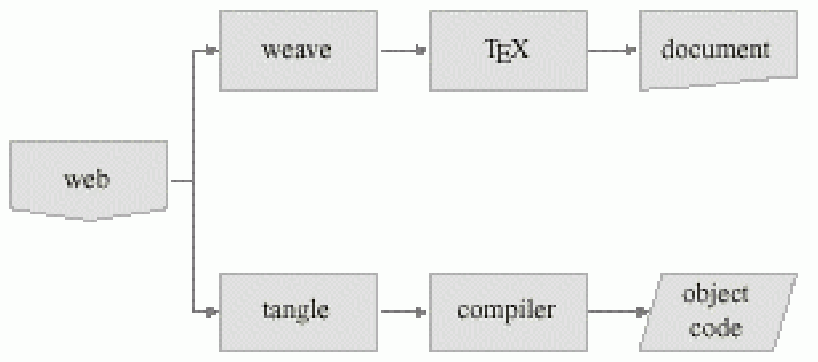
Physically Based Rendering from Theory to Implementation,

by Matt Pharr and Greg Humphreys

- Authors have experience in ray tracing
- Text Condenses lots of state-of-the art theory + code + explanation of code
- Complete code, more concrete
- Plug-in architecture

Literate programming

- A programming paradigm proposed by Knuth when he was developing Tex.
- Programs should be written more for people’s consumption than for computers’ consumption.
- Entire book is a long literate program. When you read book, you also read a complete program.



Processing a WEB

Literate Programming Features

- Mix prose with source: description of the code is as important as the code itself
- Allow presenting the code to the reader in a different order than to the compiler
- Easy to make index
- Traditional text comments usually not enough, especially for graphics
- This decomposition lets us present code a few lines at a time, making it easier to understand.
- It looks more like pseudo code.

Literate Programming Example

- Consider function

```
void InitGlobals(void){
    num_marbles = 25.7;
    shoe_size = 13;
    dielectric = true;
    my_senator = REPUBLICAN;
}
```

- Problem? Are these types double, int, etc.
- May be defined elsewhere. Unsuitable for human

Literate Programming Example

- Solution: define function in fragments

- `<Function Definitions>=`

```
   void InitGlobals( ){
         < Initialize Global Variables 3>
```

Insert explanation here

- `<Initialize Global Variables>=`

```
   shoe_size = 13;
```

Insert explanation here

- `<Initialize Global Variables>+=`

```
     dielectric = true;
```

pbrt

- Plug-in architecture
- Core code performs the main flow and defines the interfaces to plug-ins. Necessary modules are loaded at run time as DLLs, so that it is easy to extend the system.
- **`main()`** in renderer/pbrt.cpp

pbrt plug-ins

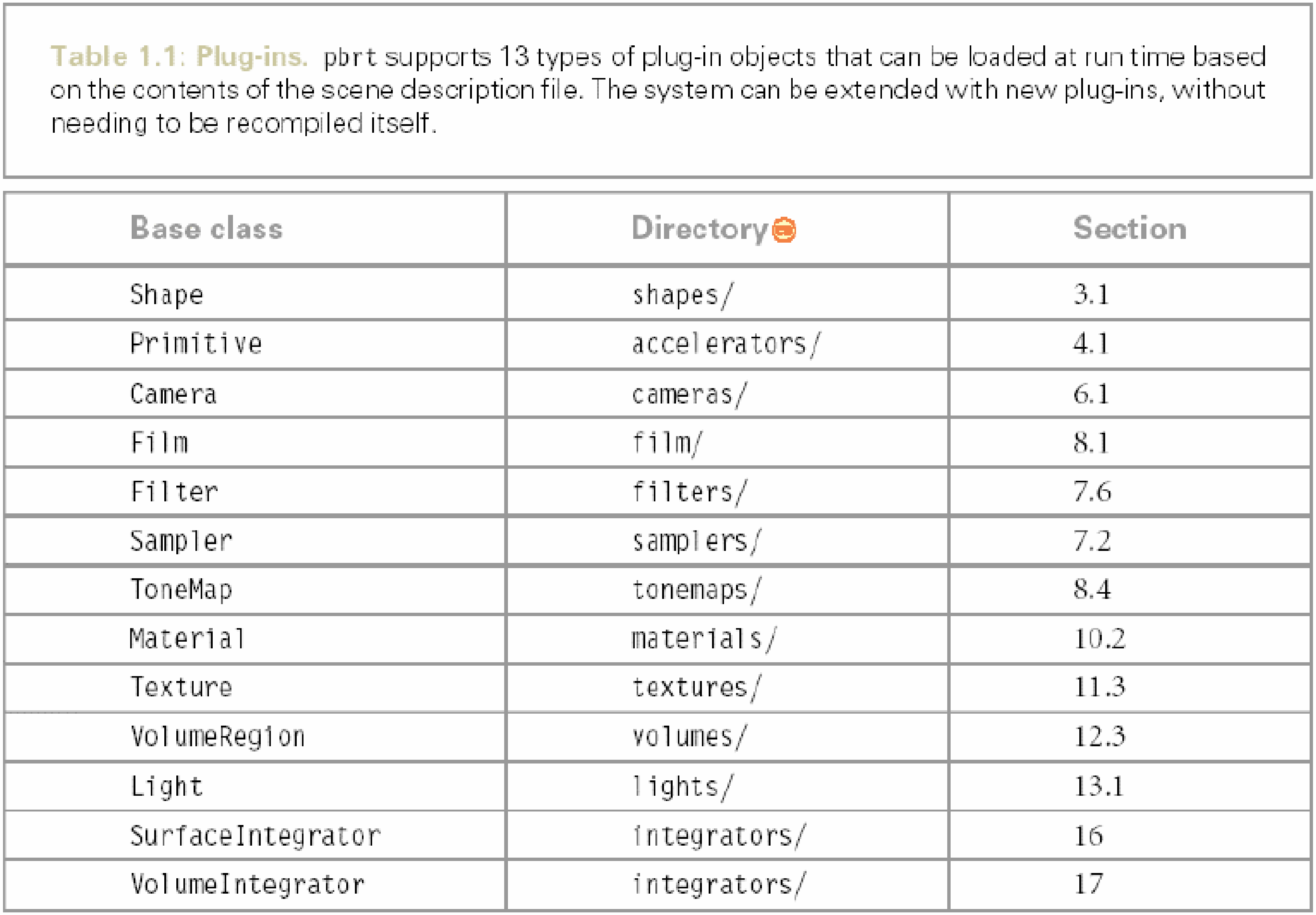
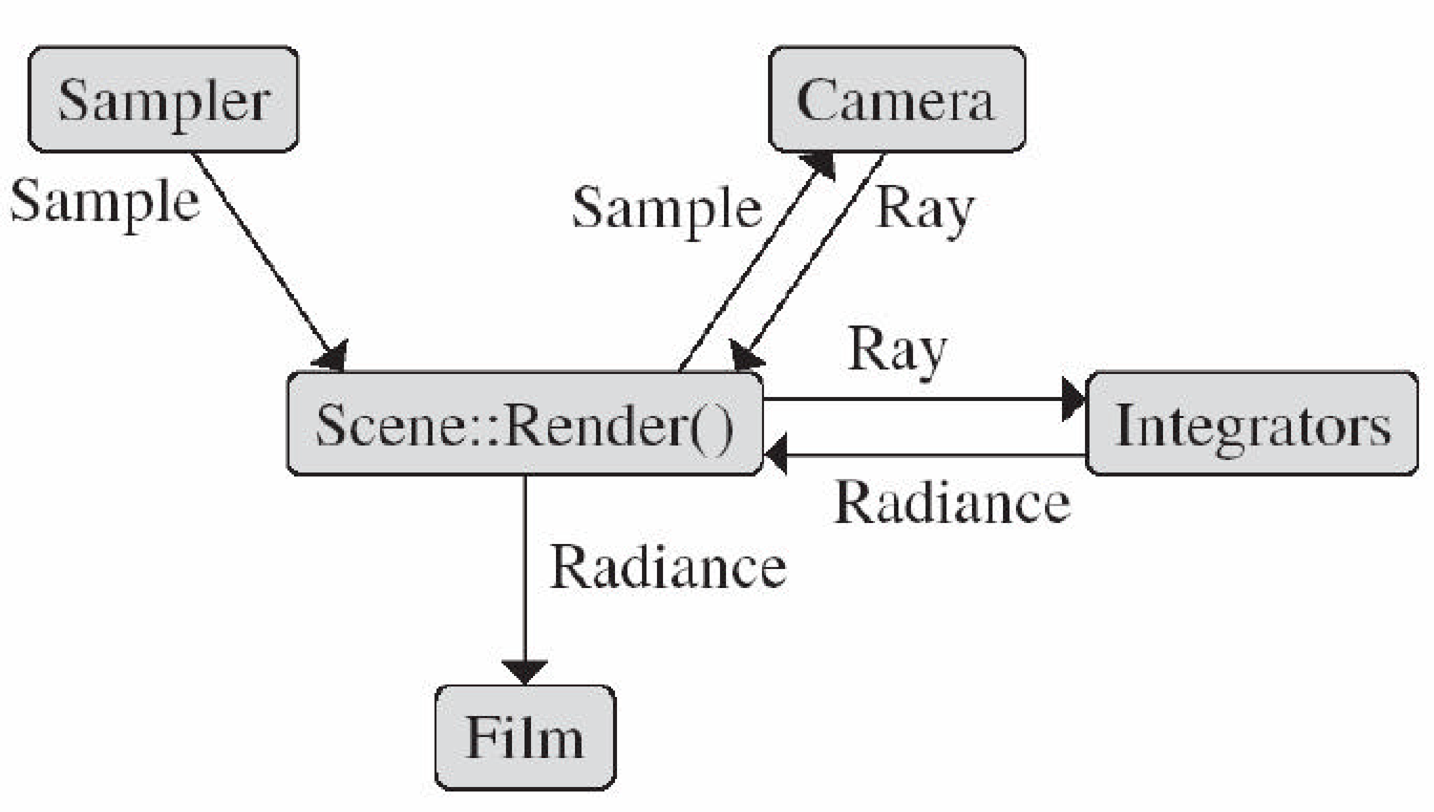


Table 1.1: Plug-ins. `pbrt` supports 13 types of plug-in objects that can be loaded at run time based on the contents of the scene description file. The system can be extended with new plug-ins, without needing to be recompiled itself.

Base class	Directory	Section
`Shape`	`shapes/`	3.1
`Primitive`	`accelerators/`	4.1
`Camera`	`cameras/`	6.1
`Film`	`film/`	8.1
`Filter`	`filters/`	7.6
`Sampler`	`samplers/`	7.2
`ToneMap`	`tonemaps/`	8.4
`Material`	`materials/`	10.2
`Texture`	`textures/`	11.3
`VolumeRegion`	`volumes/`	12.3
`Light`	`lights/`	13.1
`SurfaceIntegrator`	`integrators/`	16
`VolumeIntegrator`	`integrators/`	17

PBRT Flow

- Parsing: uses lex and yacc: core/pbrtlex.l and core/pbrtparse.y
- After parsing, a **scene** object is created (core/scene.*)
- Rendering: **`Scene::Render()`** is invoked.



Course Objectives

- Understand state-of-the-art techniques and literature for photorealistic rendering
- Learn from working code
- Hands-on exploration of one of the models/techniques encountered.
- Work with cutting edge ray tracer
- Possibly extend one of ray tracer (write plug in) to handle new effect/feature

Sample Course Topics

- High Dynamic Range Lighting
- Reflection/refraction
- Texture Mapping
- Motion Blur, Depth of Field
  - *Distributed Ray-Tracing*
- Ray tracing acceleration Techniques (kd-trees, BVH, uniform grid)
- Sub-surface scattering (skin, milk, marble)
- Monte Carlo ray tracing
- Sampling and reconstruction

Computer Skills to learn?

- Literate programming
- Lex and yacc?
- Object-oriented design
- C++ programming
- Code optimization tricks
- Modeling Techniques

Why This Class?

- WPI graduate course requirements
  - Masters, PhD, grad course requirements
- WPI research requirements
  - Want to do research in graphics (MS, PhD theses)
- Work in graphics
  - Rendering
  - Animation, etc.
- Hobbyist
  - Want to build cooler stuff
  - Understand more how visual effects, etc happen

Course Structure

- Grading
  - Presentations (2) (40%)
  - Class participation (10%)
  - Projects (50%)
    - Assigned projects +
    - Final project: Rendering contest

- Class Time:
- 2 halves with 10 minutes break
- Each half
  - 45 minute presentation
  - 30 minute discussion of topic(s) and questions

About This Course

- Previous versions of class
  - Students chose any topics/papers they liked
  - Students tend to pick what's easy
  - Sometimes big picture lost
- This version..
  - Learn how state-of-the-art physically-based rendering techniques
  - Focus on coverage in text
  - Book provides full-blown physically-based ray tracer (PBRT), description, concrete implementation
  - Projects will focus on using and modifying PBRT

Presentations

- Goal is to teach you how to present effectively
- I will be strict with time (Good practice!!)
- Try to teach concepts carefully, don’t just recite
- Communicate basic ideas to fellow students
- Offer a ‘roadmap’ for studying assigned section
- This week: Skim text
    - Next week: pick sections you want to present
- **Note:** can use any resources to build your talk. Must give credit, references. If not.. **Cheating**!!!

Presentations

- Common mistakes:
  - Avoid: putting too much on a slide (talk!!)
  - Too many slides for alloted time (2-3 mins/slide)

- First two student presentations in two weeks:

To do

- Before next class
  - Read chapters 1 –2
  - Many concepts familiar to CS 543 students
  - If you did not take CS 543 with me, skim
    - Ray tracing chapter: F.S Hill, “Computer Graphics Using OpenGL”, 2nd edition, Prentice Hall, 2000
- Homework 0
  - Download and install pbrt
  - Run several examples

Final Project

- Use some of techniques discussed to render photorealistic image
- You propose what you want to do
- Use high end package
  - Maya
  - Renderman
  - Blender
  - PovRay, etc
- Must submit proposal by March 31st, 2007
- Ideas?? See Stanford rendering competition
- *http://graphics.stanford.edu/courses/cs348b-competition/*

References/Shamelessly stolen

- Pat Hanrahan, CS 348B, Spring 2005 class slides
- Yung-Yu Chuang, Image Synthesis, class slides, National Taiwan University, Fall 2005
- Kutulakos K, CSC 2530H: Visual Modeling, course slides
- UIUC CS 319, Advanced Computer Graphics Course slides