CS 410 Lecture 01: Introduction

August 22, 2017

CS 410:
Computer Graphics

Professor:
Ross Beveridge
Room 348 CSB
(970) 491-5877
Office Hours:

- TBD
- *or by appointment*

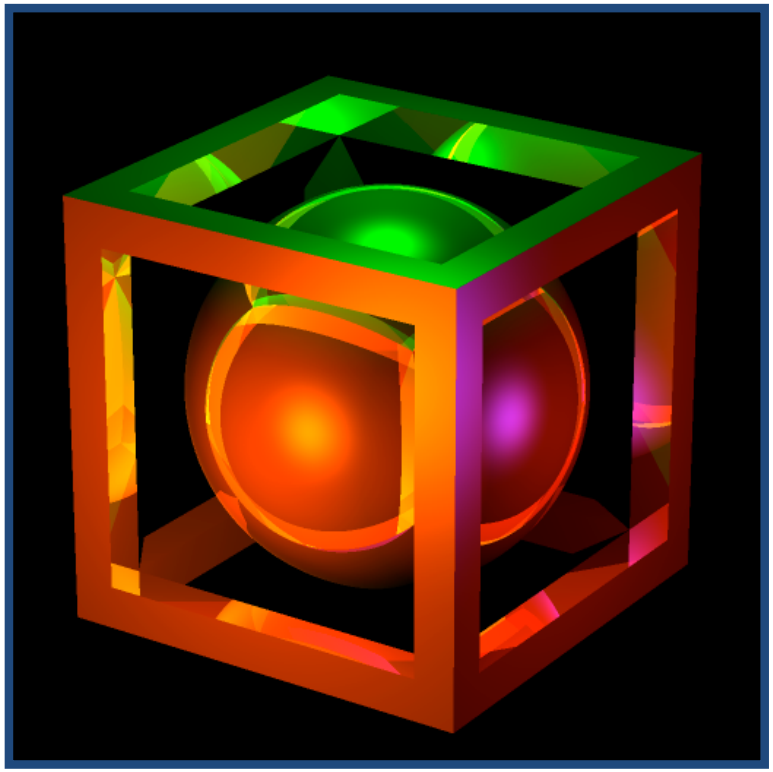
Ross.Beveridge@colostate.edu

Teaching Assistant :
Gururaj Mulay
Office Hours: TBD
cs410@cs.colostate.edu

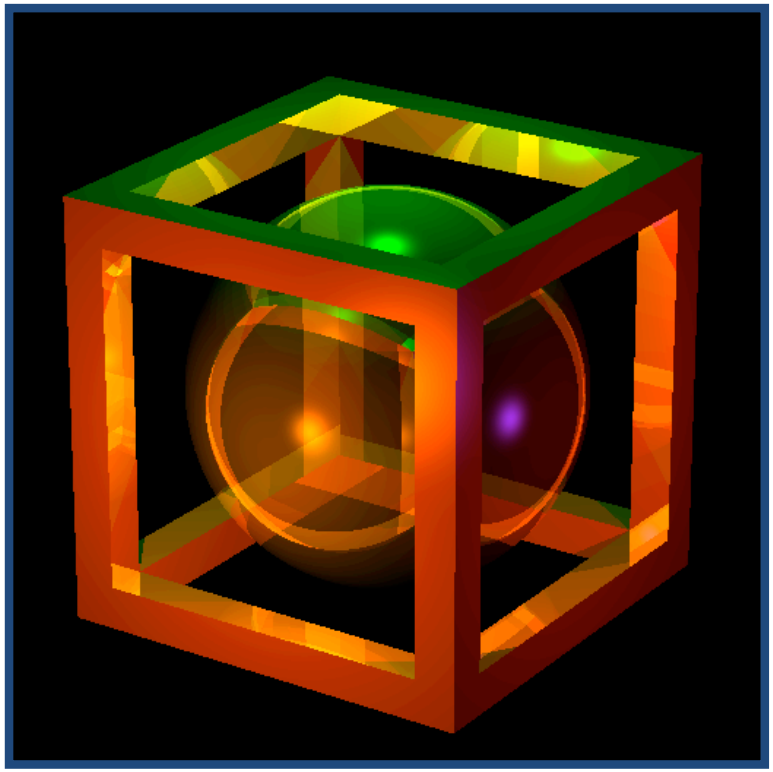
Web Site: http://www.cs.colostate.edu/~cs410

1-slide Overview

- This is what you will learn to make



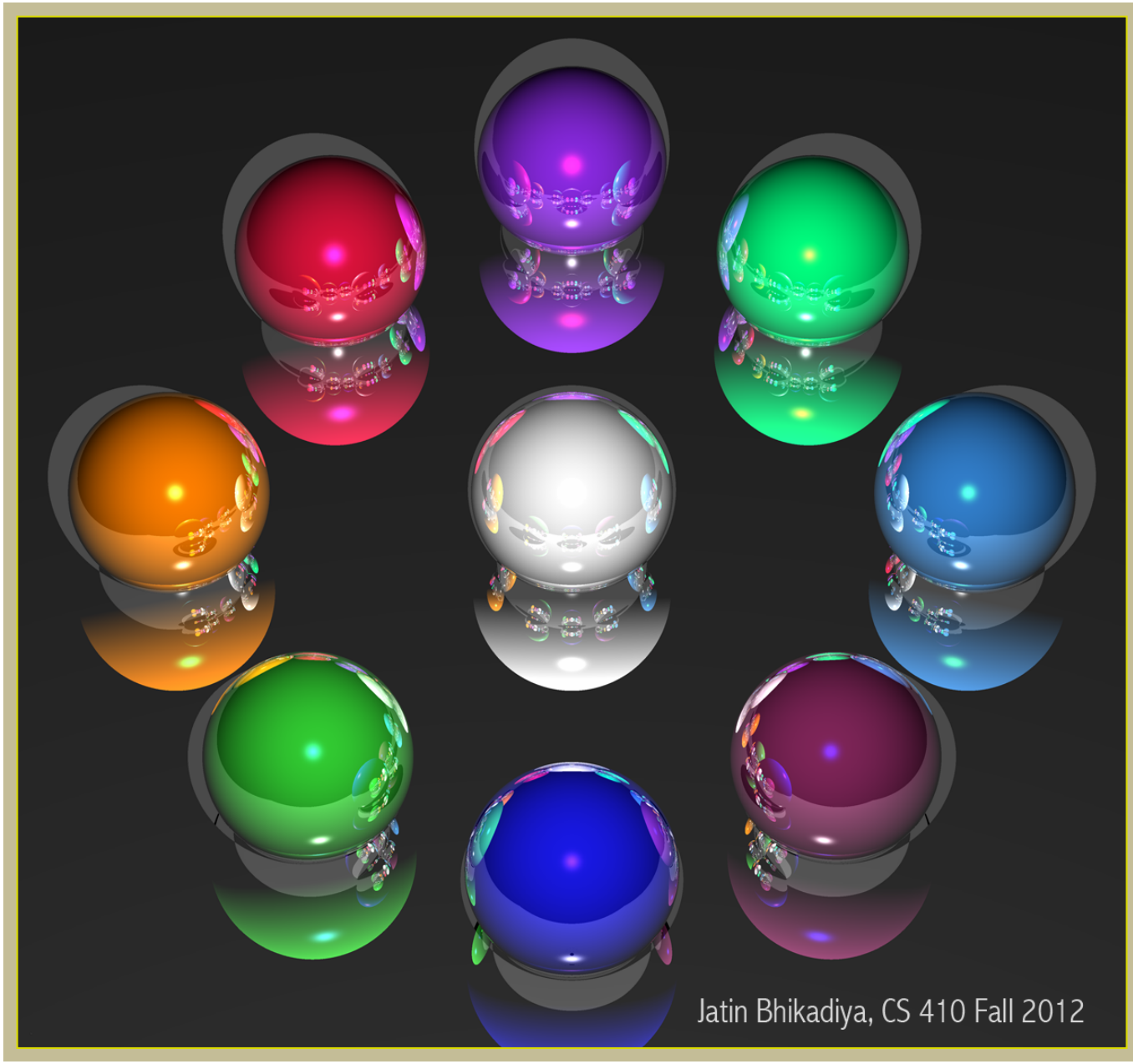
Charumathi Chandrasekaran



Ben Randall

- But you will have to work hard to get there

Another Example – Fall 2012



What can you say about this image?

In other words, what processes are at work – simulated – to create what you are seeing?

In This Class

You will build a ‘ray tracer’.

- Input
  - Scene model (objects, surfaces & materials)
  - Lighting models
  - Camera models
- Output
  - An image showing what the camera would see.

What we will do….

- Program a ray tracer from scratch
- General flow of programming assignments
  - Simple ray casting with global geometry
  - Geometric generality – relative placement
  - Illumination and surface reflectance
  - Multi-bounce tracing with reflection/refraction
- Each assignment builds on the one before
  - 20 – 30 hours each
    - Following good software practices

In the Process …

- Study computational geometry
  - Dot (i.e. inner) products (of vectors)
  - Cross products (of vectors)
  - Homogeneous coordinates
  - Affine & perspective transformations
  - Matrix multiplication
  - Orthonormal matrices
- Expertise of value beyond Graphics !

… 3-D geometric intuition …

- Expertise requires three things.
  - Linear algebraic objects and operations
  - Visual intuition – “seeing” what your specifying
  - Quick and easy shifting back and forth
- Most obvious examples.
  - Move the camera back a bit left
  - Place object A on top of object B
  - ….

… and of course also …

- Learn about light
  - Color spaces
  - Reflection & refraction
- Surface properties
  - Lambertian (Matte) reflection
  - Specular reflection
  - Hybrid reflection models

Resources (1)

- Lectures – very important
  - General concepts
  - Illustrative Examples

- Optional Textbook – a good reference
  - Details generally presented in a clear exposition.

Resources – Public Website

www.cs.colostate.edu/~cs410/yr2017fa/index.php

Introduction to Computer Graphics

Fall 2017

Computer Science Department

CS 410 : Homepage

Colorado State University

Home | Syllabus | Progress | Assignments | Resources | Canvas

CS 410 teaches students how to program a computer to generate photo-realistic images. The general idea is that given a scene model, a sensor model and a viewpoint, one should be able to create the same image that a camera would for that scene and viewpoint. Scene models include 3D object models and light sources. Objects models are composed of surfaces, and include both geometry (where is the surface?) and material properties (what is it made of? What color is it?).

This course will emphasize geometric objects and transformations, perspective projections, lighting and reflectance models, shading models, and 3D curves and surfaces. Students will design and implement a ray tracing program from scratch, thereby becoming intimately familiar with the sensor, lighting and object models described above. Perhaps most importantly, students may come to more fully appreciate the power of linear algebra.

Instructor:
Ross Beveridge
Office: 348 CS Building
Office Hours: TBD
Email: Ross.Beveridge@colostate.edu
GTA:
Gururaj Mulay
Office: CSB 120
Office Hours: TBD
Email: gururaj.mulay@gmail.com
Lecture Time and Place:
3:30-4:45, Tue, Thur, ECE 100

News:
This site is up-to-date for the start of Fall 2017. CANVAS is not (yet). (8/18/17 Ross)
Do not miss the eclipse Monday. See you all Tuesday for lecture. (08/17/17 Ross)

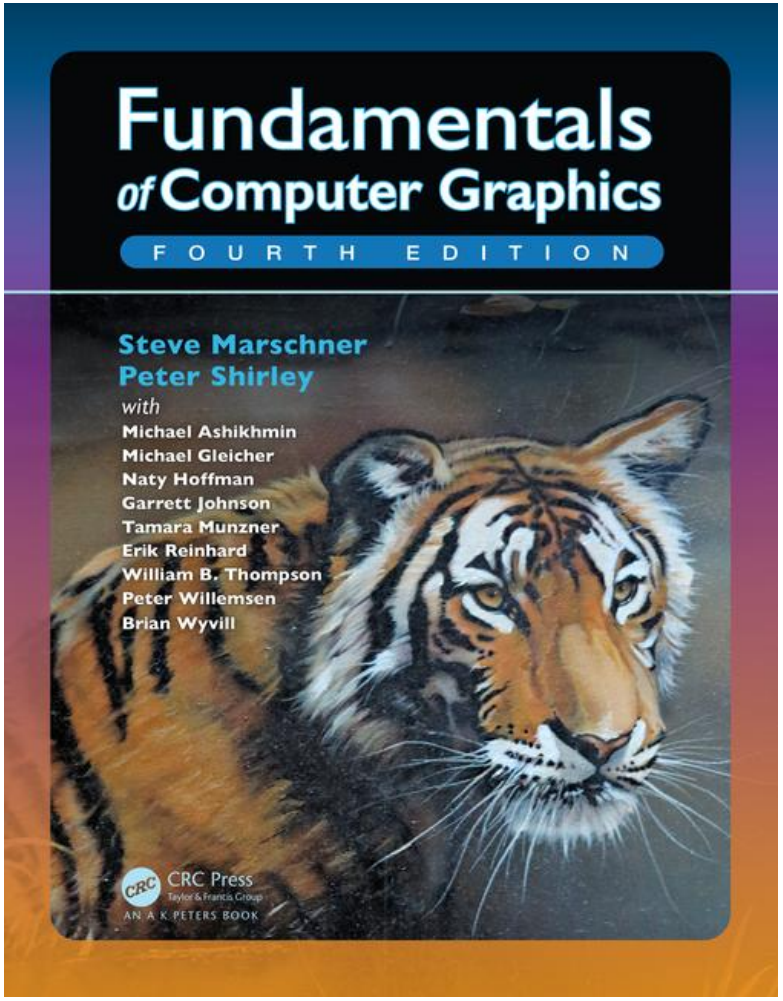
The Caged Sphere logo originally ray traced by Charlie Ross.

Resources (3)

- Private Website – Canvas
  - Used for Quizzes, Assignments, Grading …
- Office hours
  - Mine: TBD (good starting point is email)
  - Gururaj Mulay's : TBD (not his fault)
  - Don't be a stranger!

Optional Textbook

- Adopted in 2011 at CSU
- Virtues
  - Up-to-date
  - Focuses on Ray Tracing
  - Consistent terminology
  - Well staged mathematics
  - E-format available
- Either 3rd or 4th edition



Course Rules

- Be on time
  - Class starts at 3:30
- Ask & Answer questions
  - Always give your name in the process
- Professional behavior at all times
  - No non-course related chatter
  - Cell phones on silent
    - Leave the room to answer them
  - Be polite & respect others

Major Activities / Grading

- Quizes
  - 10% of your grade – there to help you study
- Programming Projects (~5)
  - 50% of your grade total
  - 10% each
- Midterms (2)
  - 20 % of your grade (total)
  - 10 % of your grade (each)
- Final exam
  - 20% of your grade

Policies

- Assigned work is done alone
  - No joint projects
  - No open note exams
  - No taking code from the internet
  - Follow the department academic integrity policy
- All work done on time
  - No late period – deadlines are deadlines
    - Multiple submissions OK – last is graded
  - When given two (or three) weeks, start right away!
- Exceptions: unforeseeable circumstances

Systems and Tools

- You may be surprised …
- You will not learn a complex and/or arcane API to a giant graph package
- Instead, you will build your ray tracer from scratch in C++ or Java (no Python)
- Applying programming techniques from CS253 and CS314.
- A taste of web programming along the way.

This course is NOT…

- A course in OpenGL
  - OpenGL shaders are complex & detailed
  - Ray Tracing will become the dominant paradigm
    - Thanks to GPUs and parallel architectures
- A course in using any other Graphics API

This is a course about the mathematics and algorithms underlying Ray Tracing. It should make you a better programmer, improve your linear algebra, and prepare you to study computer vision (which is where the jobs are…)

Systems - Requirements

- Ray Tracer Mechanics
  - Must run on CS Linux Machines
  - Tools for viewing models will be provided
  - Tools for viewing images will be provided
- On More Thing
  - Expect one assignment using JavaScript

The Pelican

The person with the pelican speaks next.

Why?

- One-way lectures suffer many flaws.
- Instructor can become detached.
- Students less engaged.
- Practice speaking is valuable.
- Why the Pelican?
- Same people always speaking.
- Knowing when your up next.

Save 2D for Last

- Semester is front loaded for 3D Ray Tracing
- At the end to pickup some missing pieces
- HTML5 and the Drawing Canvas
  - Bad news …
    - You will be asked to learn about JavaScript.
  - Good news …
    - You will be asked to learn about JavaScript.
  - Best news …
    - **Simple universal event driven 2D graphics.**

Example of 2-D Animation

Developer Tools - https://www.airtightinteractive.com/demos/processing_js/noisefield08.html

Elements Console Sources Network Timeline Profiles Application Security Audits

Sources Content scripts Snippets

- top
  - www.airtightinteractive.com
    - demos/processing_js
      - noisefield08.html
      - processing-0.9.6.min.js
      - style.css
      - black.png

noisefield08.html processing-0.9.6.min.js processing-0.9....n.js:formatted

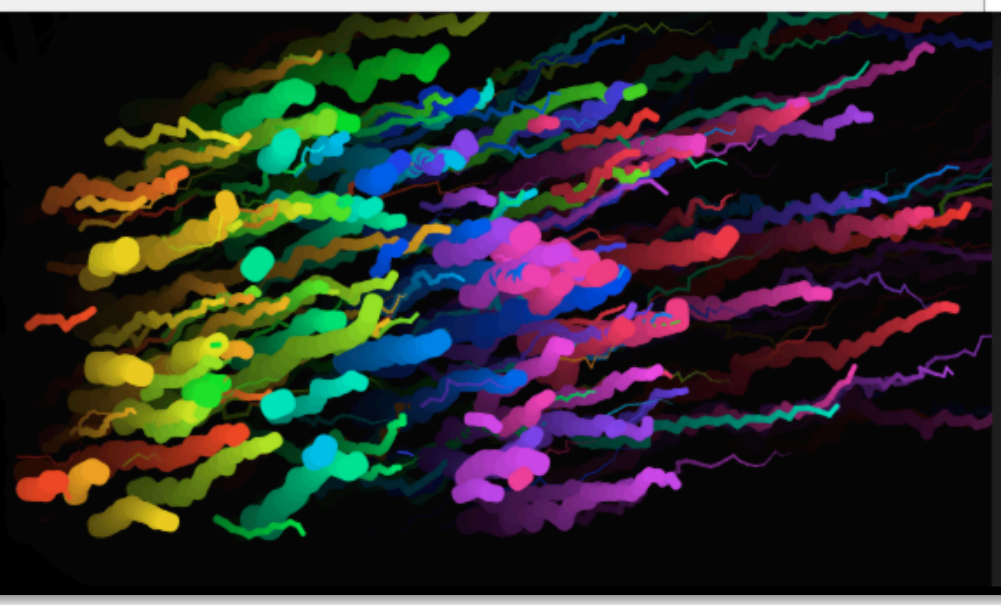
```
(function() {
    var undef;
    var ajax = function ajax(url) {
        var xhr = new XMLHttpRequest();
        xhr.open("GET", url, false);
        xhr.setRequestHeader("If-Modified-Since", "Fri, 1 Jan 1960 00:00:00 GMT");
        xhr.send(null );
        if (xhr.status !== 200 && xhr.status !== 0) {
            throw ("XMLHttpRequest failed, status code " + xhr.status)
        }
        return xhr.responseText
    }
    ;
    var PConstants = {
        X: 0,
        Y: 1,
        Z: 2,
```

Line 1, Column 1

Watch
Call Stack
Not Paused
Scope
Not Paused
Breakpoints
No Breakpoints
DOM Breakpoints
XHR Breakpoints
Event Listener Break
Event Listeners

Noise Field

https://w



In CS 410

- You will NOT be learning AJAX, etc.
- You will create moving 2D graphics.

The Main Act – Create an Image

Let us walk quickly through some of the key concepts of this course as a means of setting up a context for what is to come.

What is an image?

What is an image? (cont)

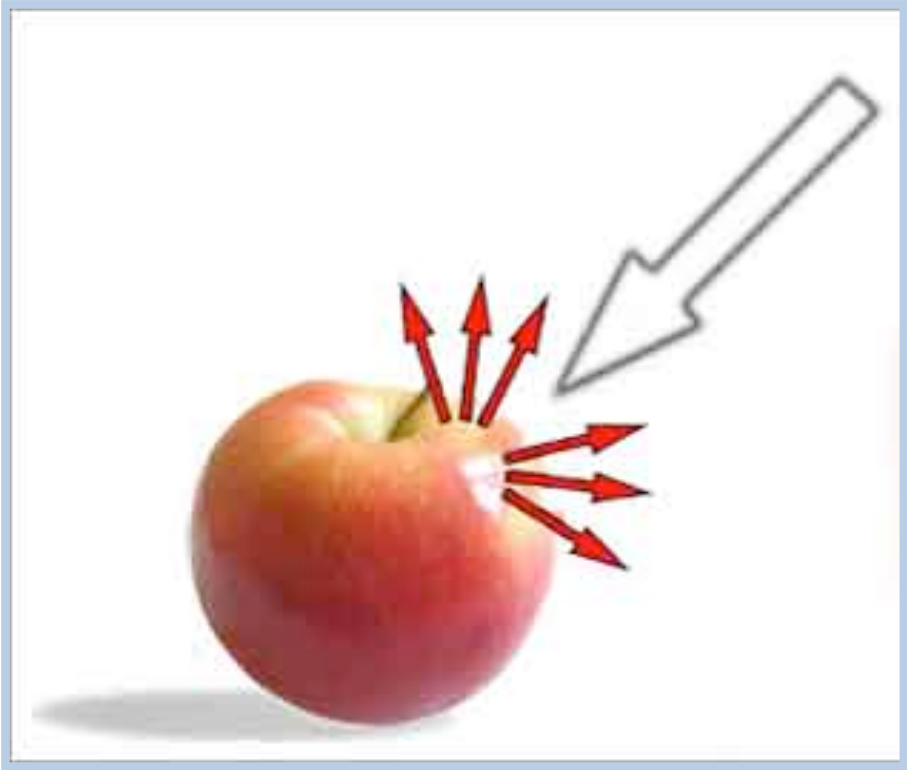
- An array of values
  - Intensities (if gray scale)
  - Triples of red, green & blue (color)
- This image is a part of the previous image
  - Where?

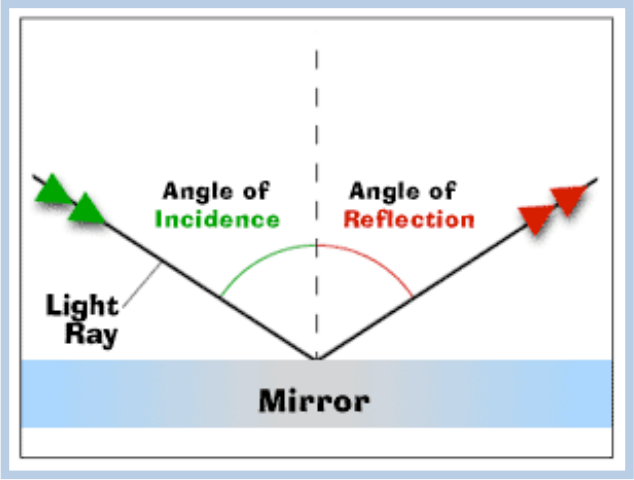
How is an image *formed*?

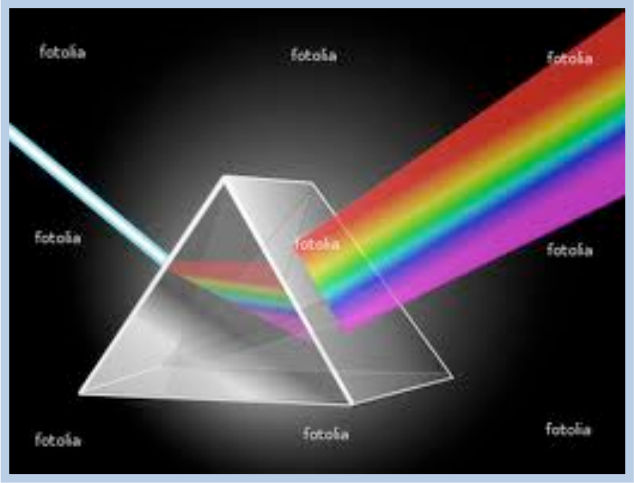
Step 1: Light Source

- Lights Sources have
  - Intensities
  - Color
  - Positions
- Bonus: atmospheric effects

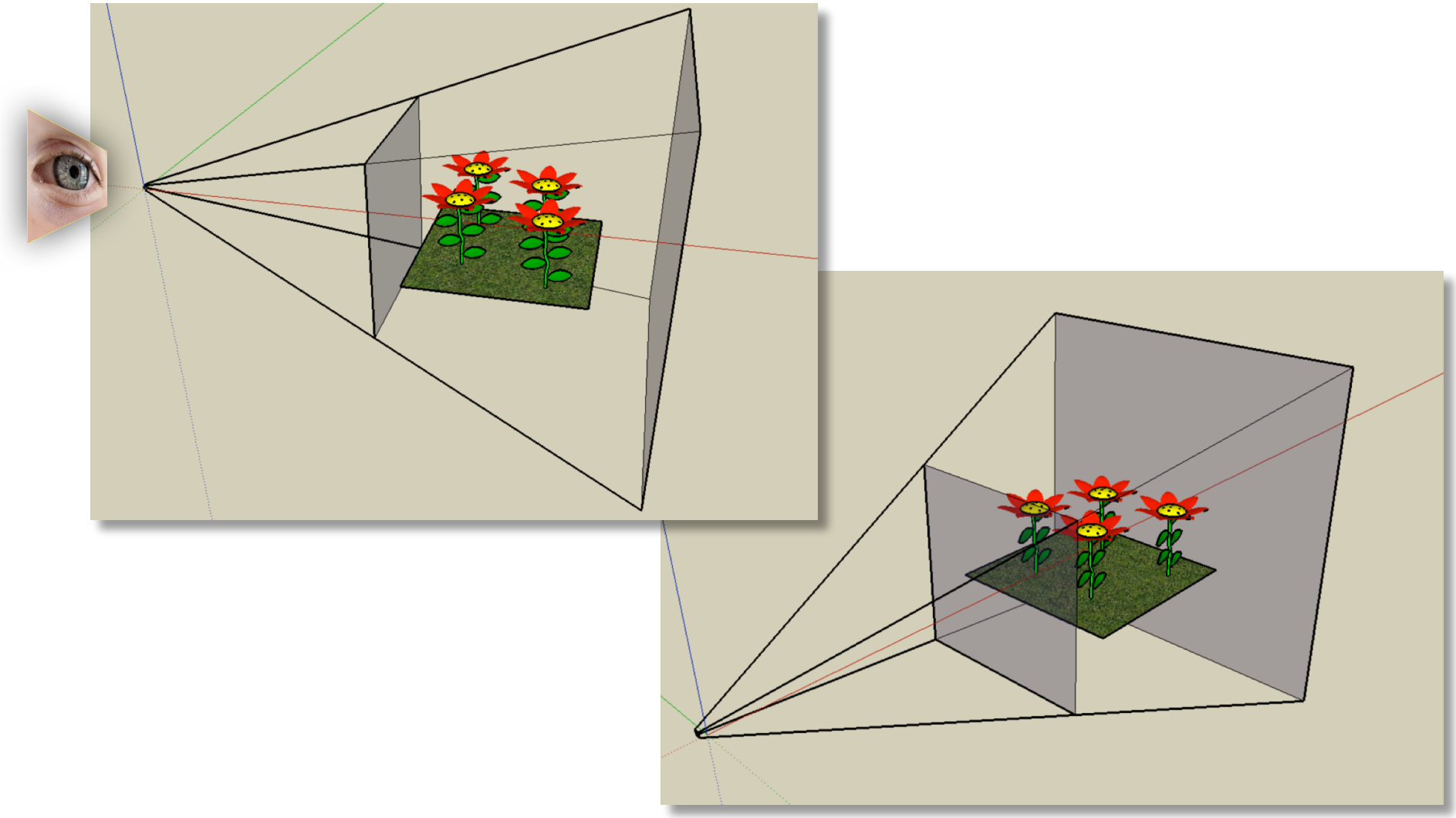
Step 2: Reflection



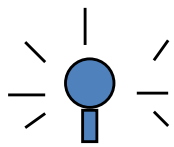




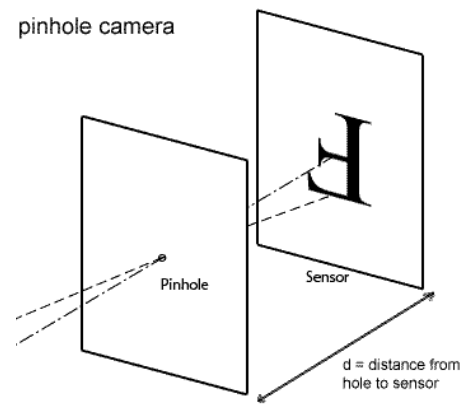
Step 3: Projection



How (artificial) images are formed

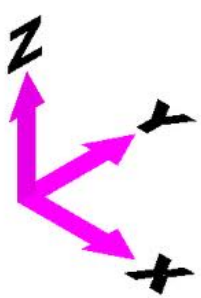


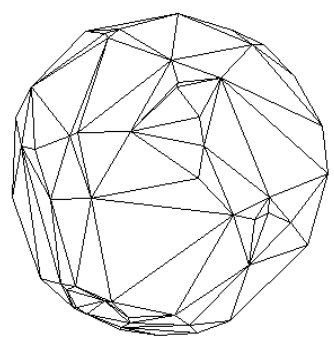
Light Source



Sensor

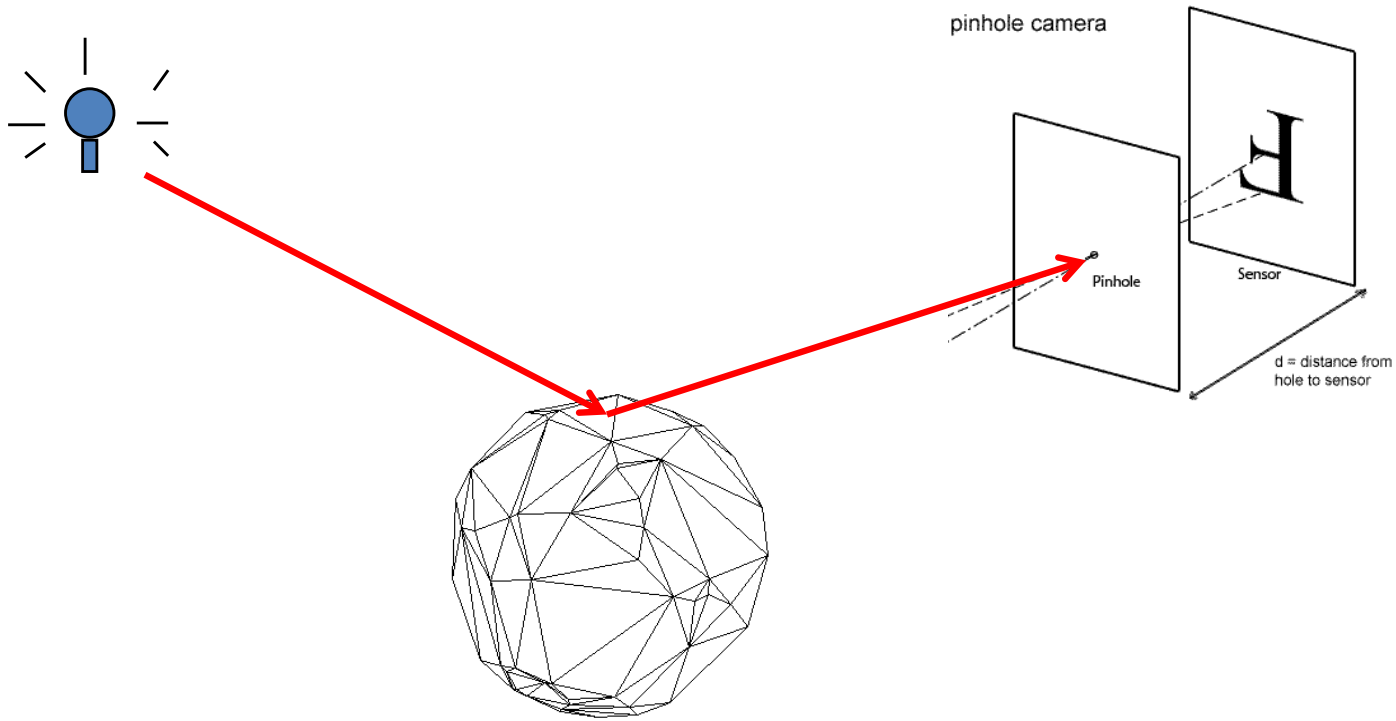
Coordinate System





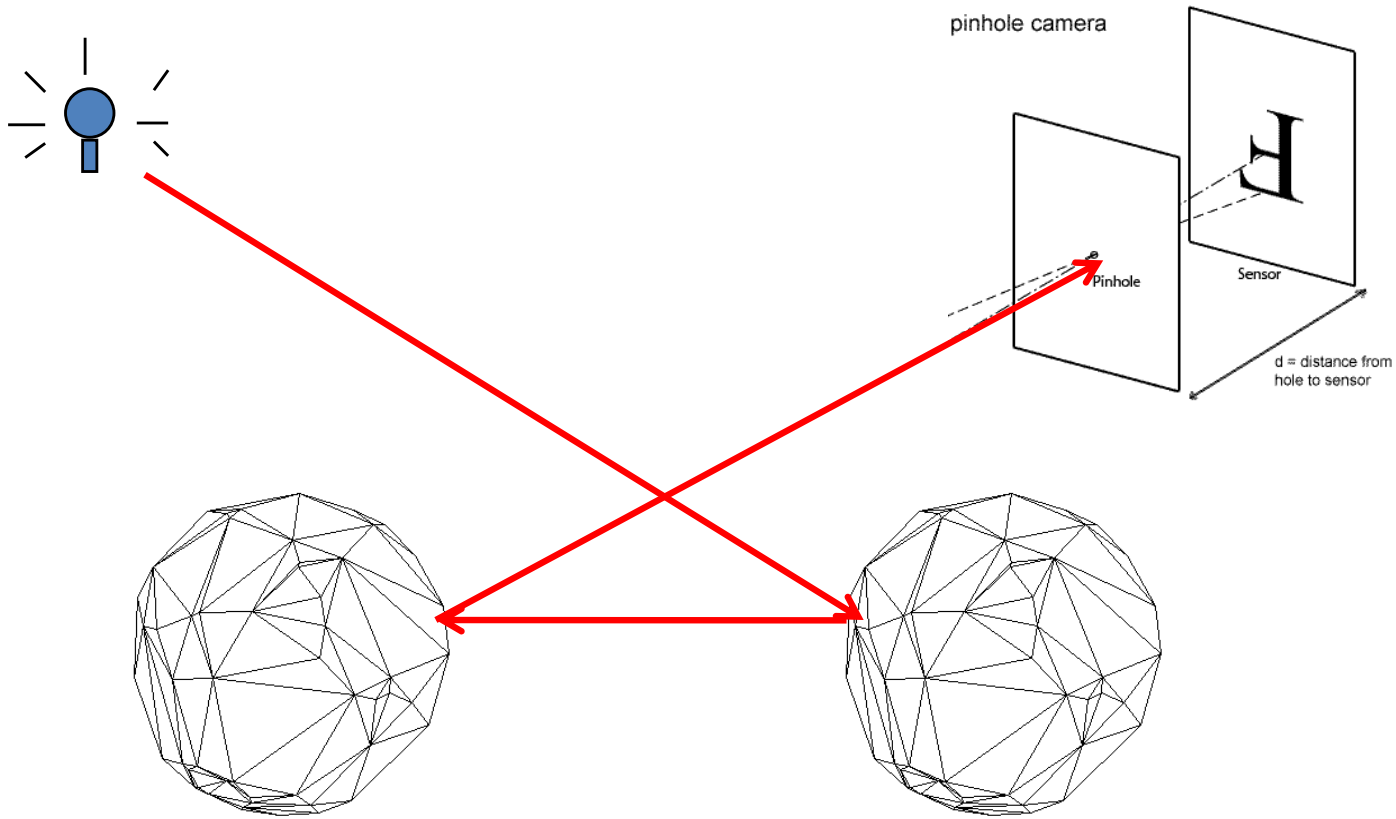
Geometric Objects

Now mimic light…



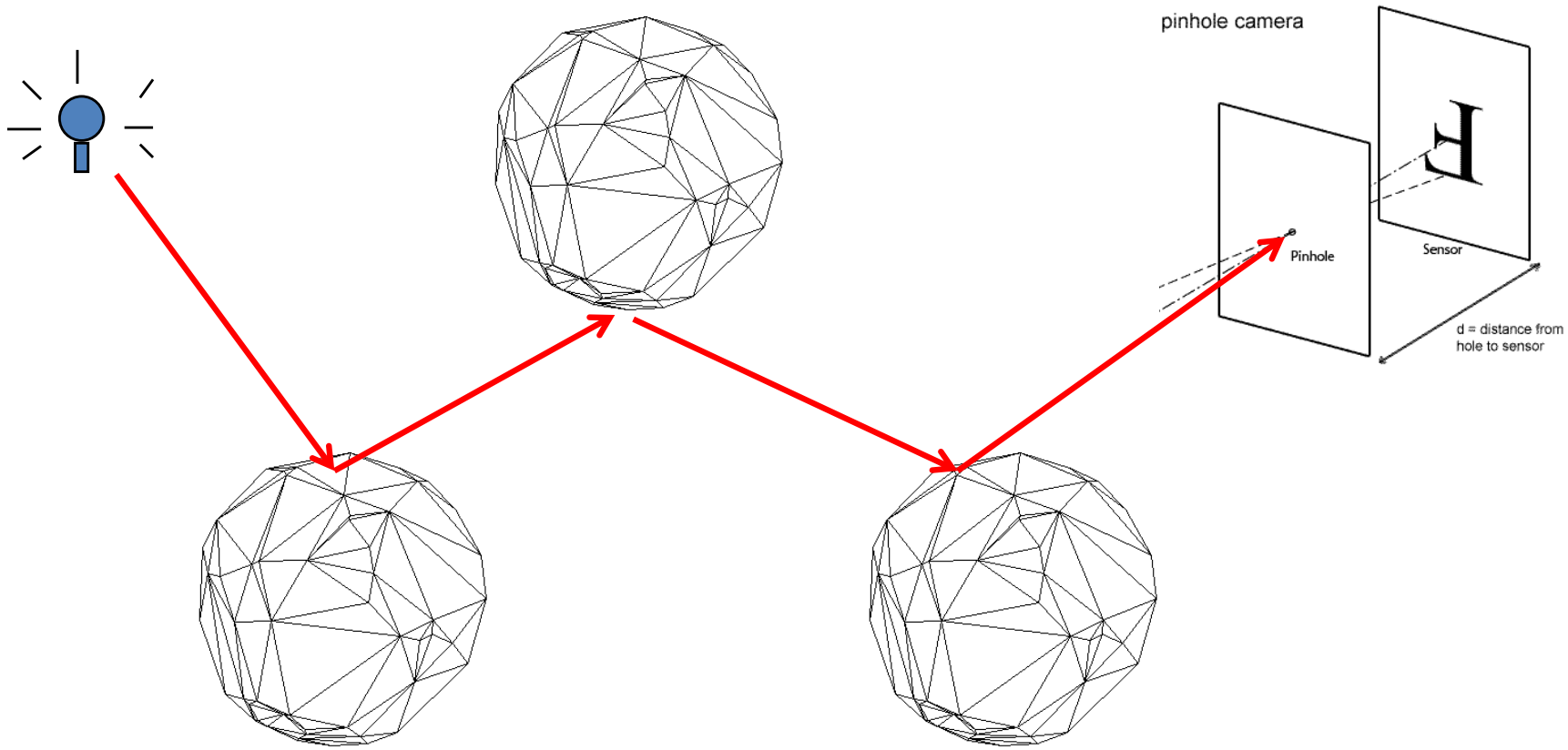
From source to object to sensor

… or …



…from source to object to object to sensor

… or …



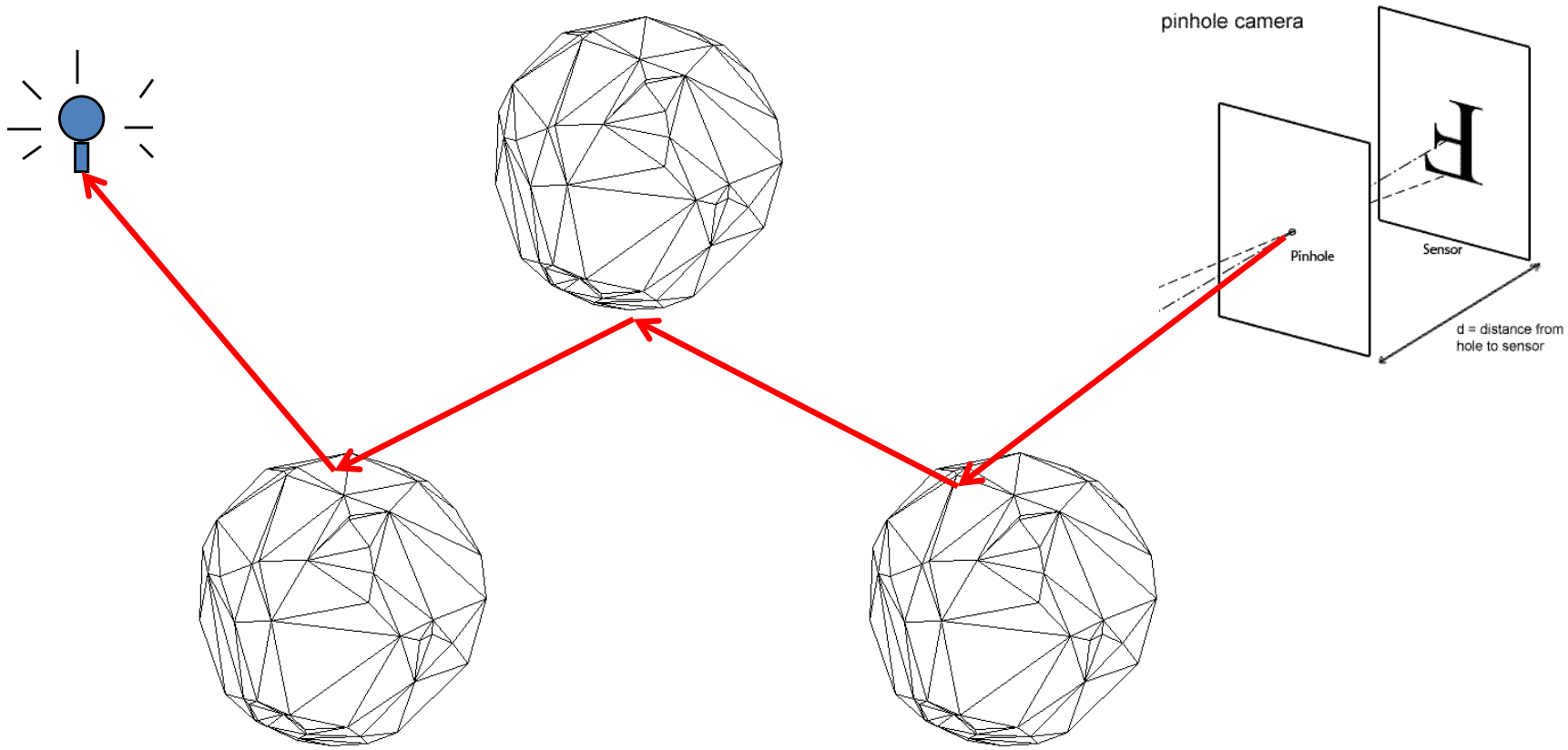
…you get the point.

True – But Backwards!

- Light sources emit a LOT of light rays
  - Most of them never strike your camera
- And you might have multiple light sources
  - In fact, you usually do, and the light intermingles
- So following every light ray is expensive

So instead, work backwards: start at the surface of the lens, and figure out where the light came from … all the way back to the light source.

In other words



What can this approach do?

A photorealistic rendered image created by using POV-Ray 3.6
Image from Wikipedia entry on rendering.

CS 410 Project done by Kyle Olson in Fall of 2014.

This particular project went a bit beyond what is required - that is why it is framed on my office wall.

Still, this is what you can do if you put your mind to it.

What skills will we need?

- Lots of Math
  - Linear Algebra (Matrices, Vectors, Dot Products)
  - Discrete representations (Images)
- A little Physics
  - Lighting/Energy Transfer
  - Color reflections/refraction
- A little Mechanical Engineering
  - CAD/CAM, Solid Models
- A whole bunch of Computer Science

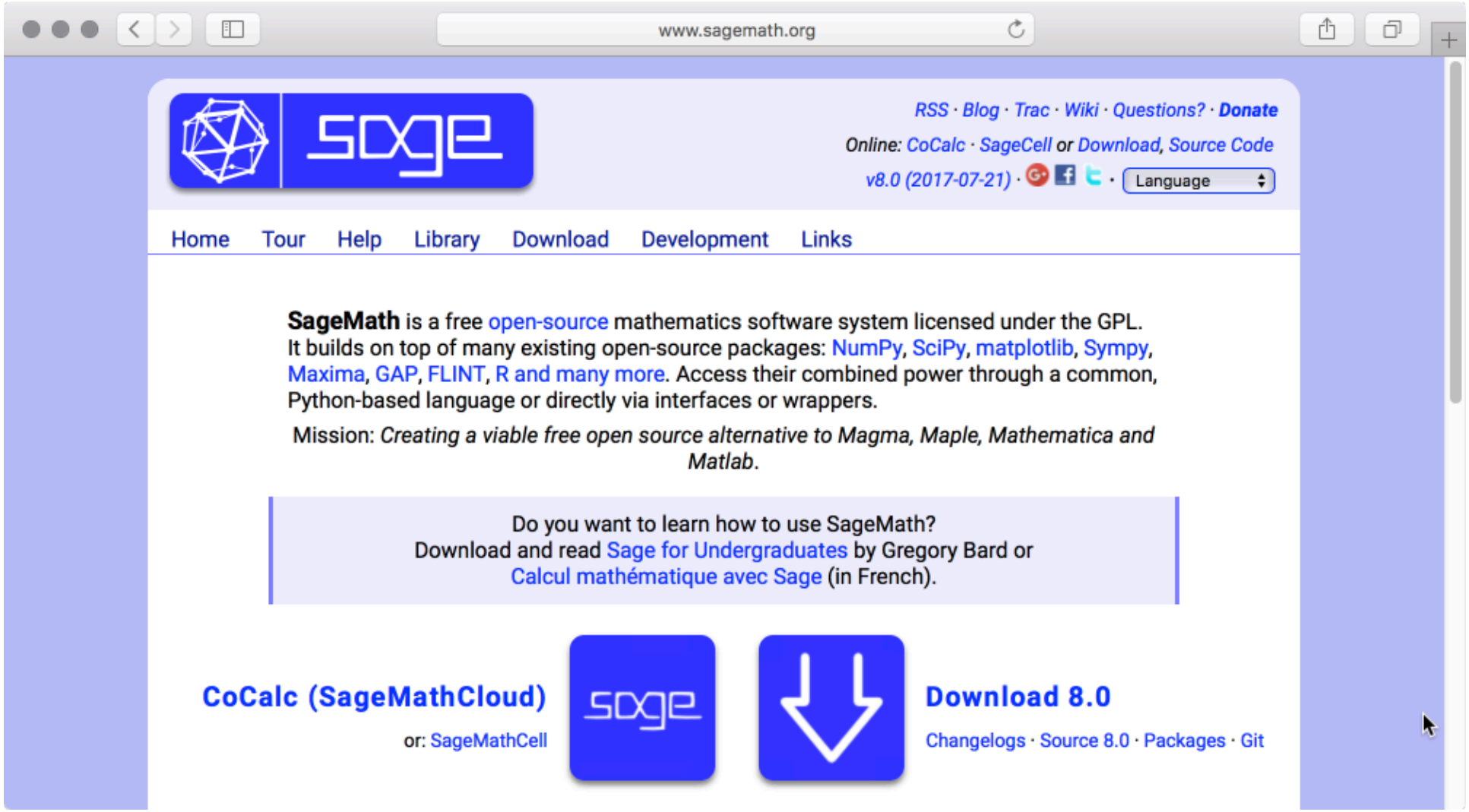
What do I expect?

- Familiarity with vectors and matrices
  - Mechanics, e.g. multiplication
  - Basic ideas (perhaps rusty)
- Programming ability
  - You know how to design, implement & test thousand line plus programs
    - Test-first development
    - Design patterns
    - Object-oriented designs
- Willingness to work (hard)

Quick Who Does What?

- I will describe graphics principles
  - Mathematics (e.g. projection)
  - Physics (e.g. reflections)
- I will describe some algorithms
  - E.g. ray/polygon intersection
- We will discuss/practice concepts.
- You will convert ideas into code.

One More Thing



The End