Lecture 32 of 41

Lab 6: Ray Tracing with ACM SIGGRAPH Demo & POV-Ray

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course pages: http://bit.ly/hGvXlH / http://bit.ly/eVizrE
Public mirror web site: http://www.kddresearch.org/Courses/CIS636
Instructor home page: http://www.cis.ksu.edu/~bhsu

Readings:

Last class: Chapter 14, Eberly *2e* – see http://bit.ly/ieUq45
Today: **Ray Tracing Handout**
Next class: Chapter 15, **Ray Tracing Handout**

Lecture Outline

- Reading for Last Class: Chapter 14, Eberly *2e*
- Reading for Today: Ray Tracing Handout
- Reading for Next Class: Chapter 15, Eberly *2e*; Ray Tracing Handout
- Last Time: Ray Tracing (RT), Part 1 of 2
  - Vectors: Light (L) & shadow, Reflected (R), Transmitted & refraction
  - Basic recursive ray tracing & ray trees
  - Phong illumination model, texture mapping revisited
  - Distributed RT: survey, supersampling illustrated
  - Things you get "for free": clipping, VSD (backface/occlusion culling)
- Today: Ray Tracing Lab
  - ACM SIGGRAPH demo: http://bit.ly/cllgx2
  - POV-Ray: http://www.povray.org
- Next Class: Ray Tracing 2 of 2
  - Hybridizing RT with radiosity (photon maps)
  - Progressive refinement

Where We Are

21	**Lab 4a: Animation Basics**	**Flash animation handout**
22	Animation 2: Rotations; Dynamics, Kinematics	Chapter 17, esp. §17.1 – 17.2
23	Demos 4: Modeling & Simulation; Rotations	Chapter 10[1], 13[2], §17.3 – 17.5
24	**Collisions 1: axes, OBBs, Lab 4b**	**§2.4.3, 8.1, GL handout**
25	Spatial Sorting: Binary Space Partitioning	Chapter 6, esp. §6.1
26	**Demos 5: More CGA; Picking; HW/Exam**	**Chapter 7[2]; § 8.4**
27	**Lab 5a: Interaction Handling**	**§ 8.3 – 8.4; 4.2, 5.0, 5.6, 9.1**
28	Collisions 2: Dynamic, Particle Systems	§ 9.1, particle system handout
	Exam 2 review; Hour Exam 2 (evening)	**Chapters 5 – 6, 7[2] – 8, 12, 17**
29	**Lab 5b: Particle Systems**	**Particle system handout**
30	**Animation 3: Control & IK**	**§ 5.3, CGA handout**
31	Ray Tracing 1: intersections, ray trees	Chapter 14
32	Lab 6a: Ray Tracing Basics with POV-Ray	RT handout
33	Ray Tracing 2: advanced topic survey	Chapter 15, **RT handout**
34	Visualization 1: Data (Quantities & Evidence)	Tufte handout (1)
35	**Lab 6b: More Ray Tracing**	**RT handout**
36	Visualization 2: Objects	Tufte handout (2 & 4)
37	Color Basics; Term Project Prep	**Color handout**
38	Lab 7: Fractals & Terrain Generation	**Fractals/Terrain handout**
39	**Visualization 3: Processes; Final Review 1**	Tufte handout (3)
40	**Project presentations 1; Final Review 2**	**–**
41	Project presentations 2	–
	Final Exam	**Ch. 1 – 8, 10 – 15, 17, 20**

Lightly-shaded entries denote the due date of a written problem set; heavily-shaded entries, that of a machine problem (programming assignment); blue-shaded entries, that of a paper review; and the green-shaded entry, that of the term project.

Green, **blue** and **red** letters denote **exam review**, **exam**, and **exam solution review** dates.

Acknowledgements: Ray Tracing

Dave Shreiner & Brad Grantham

Adjunct Professor & Adjunct Lecturer,
Santa Clara University
ARM Holdings, plc
http://www.plunk.org/~shreiner/
http://www.plunk.org/~grantham/

ARM The Architecture for the Digital World®

SANTA CLARA UNIVERSITY SCHOOL OF ENGINEERING

David K. Buck, Aaron Collins, *et al.*

Developers
Persistence of Vision Raytracer (POV-Ray)
http://www.povray.org

G. Scott Owen & Yan Liu

Professor Emeritus / ACM SIGGRAPH President &
Graduate Research Assistant
Hypermedia and Visualization Laboratory
Department of Computer Science
Georgia State University / ACM
http://www.cs.gsu.edu/gsowen/

GeorgiaStateUniversity COMPUTER SCIENCE

ACMSIGGRAPH

Review [1]: Reasons for Using Ray Tracing

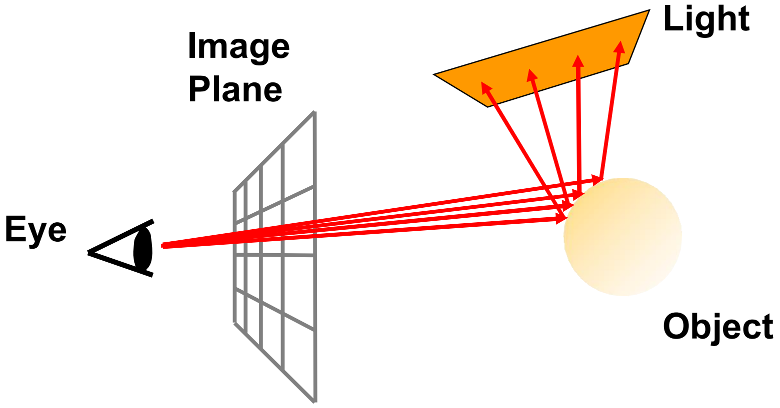
- Simulate rays of light
- Produces natural lighting effects
    - Reflection
    - Refraction
    - Soft Shadows
    - Depth of Field
    - Motion Blur
    - Caustics
- Hard to simulate effects with rasterization techniques (OpenGL)
- Rasterizers require many passes
- Ray-tracing easier to implement

Adapted from slides ♥ 2001 D. Shreiner & B. Grantham, SCU
COEN 290: Computer Graphics I, Winter 2001 – http://bit.ly/hz1kfU

SANTA CLARA UNIVERSITY
SCHOOL OF ENGINEERING

Review [2]: How Ray Tracing Works

- Trace rays from eye instead
- Do work where it matters

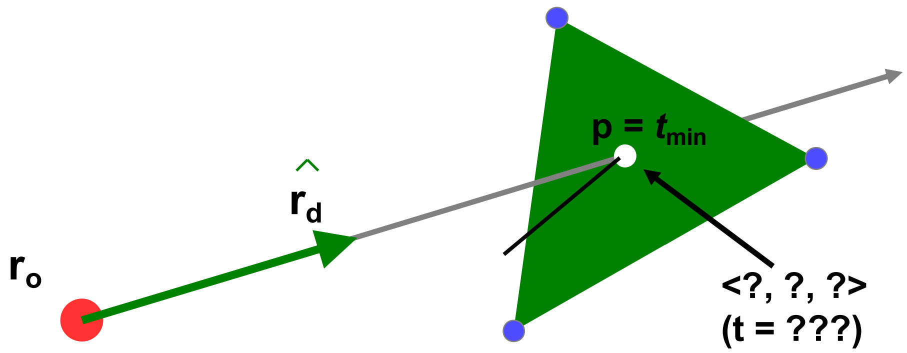


This is what most people mean by "ray tracing".

Adapted from slides ♥ 2001 D. Shreiner & B. Grantham, SCU
COEN 290: Computer Graphics I, Winter 2001 – http://bit.ly/hz1kfU

Review [3]: Ray/Triangle Intersection

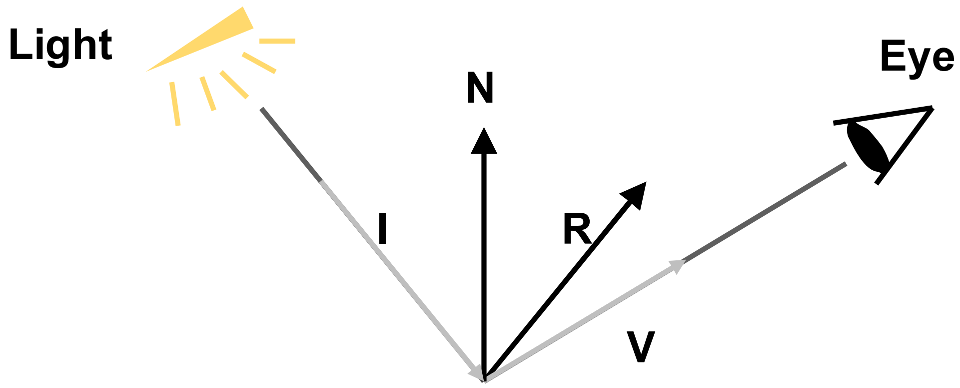
- Want to know: at what *point* p does ray intersect triangle?
- Compute lighting, reflected rays, shadowing *from that point*



Adapted from slides ♥ 2001 D. Shreiner & B. Grantham, SCU
COEN 290: Computer Graphics I, Winter 2001 – http://bit.ly/hz1kfU

Review [4]: General Notation Review

- We'll use triangles for lights
- Can build complex shapes from triangles
- Some lighting terms



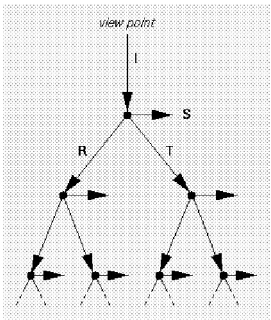
Adapted from slides ♥ 2001 D. Shreiner & B. Grantham, SCU
COEN 290: Computer Graphics I, Winter 2001 – http://bit.ly/hz1kfU

Review [5]: Recursive Calculation & Ray Tree

- Recursive ray evaluation

```
rayTrace(ray) {
    hitObject(ray, p, n, triangle);
    color = object color;
    if(object is light)
        return(color);
    else
        return(lighting(p, n, color));
}
```

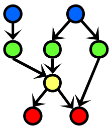
- Generates ray tree shown at right



Ray tree
© 2000 N. Patrikalakis, MIT
http://bit.ly/fjcGGk

I = Incident ray
S = light Source vector (*aka* L)
R = reflected ray
T = transmitted ray

Adapted from slides ♥ 2001 D. Shreiner & B. Grantham, SCU
COEN 290: Computer Graphics I, Winter 2001 – http://bit.ly/hz1kfU



Review [6]: Putting It All Together

- **Calculating surface color**

```
lighting(point) {
    color = ambient color;
    for each light
        if(hitObject(shadow ray))
            color += lightcolor *
                  dot(shadow ray, n);
    color += rayTrace(reflection) *
        pow(dot(reflection, ray), shininess);
    return(color);
}
```

Adapted from slides ♥ 2001 D. Shreiner & B. Grantham, SCU
COEN 290: Computer Graphics I, Winter 2001 – http://bit.ly/hz1kfU

Review [7]: More Quality, More Speed

- Better Lighting + Forward Tracing
- Texture Mapping
- Modeling Techniques
- Distributed Ray Tracing: Techniques
  - Motion Blur
  - Depth of Field
  - Blurry Reflection/Refraction
  - Wikipedia, *Distributed Ray Tracing*: http://bit.ly/ihyVUs
- Improving Image Quality
- Acceleration Techniques

Adapted from slides ♥ 2001 D. Shreiner & B. Grantham, SCU
COEN 290: Computer Graphics I, Winter 2001 – http://bit.ly/hz1kfU

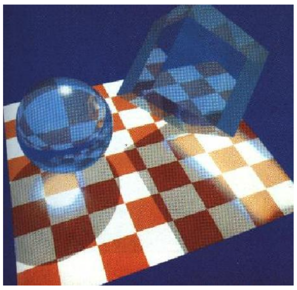
SANTA CLARA UNIVERSITY
SCHOOL OF ENGINEERING

Review [8]: Distributed Ray Tracing

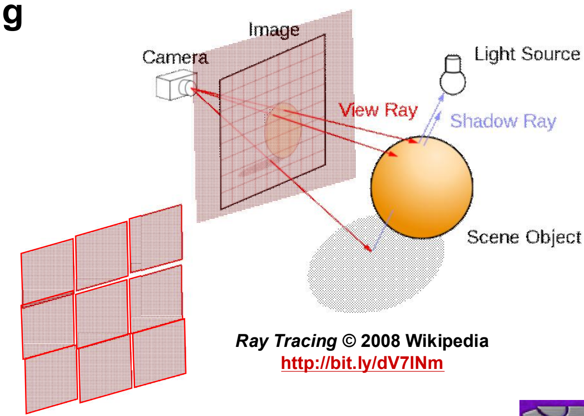
Adapted from slides ♥ 2001 D. Shreiner & B. Grantham, SCU
COEN 290: Computer Graphics I, Winter 2001 – http://bit.ly/hz1kfU

Review [9]: Supersampling, "Forward" RT

- One ray is not enough (jaggies)
- Can use multiple rays per pixel - *supersampling*
- Can use a few samples, continue if they're very different - *adaptive supersampling*
- Texture interpolation & filtering



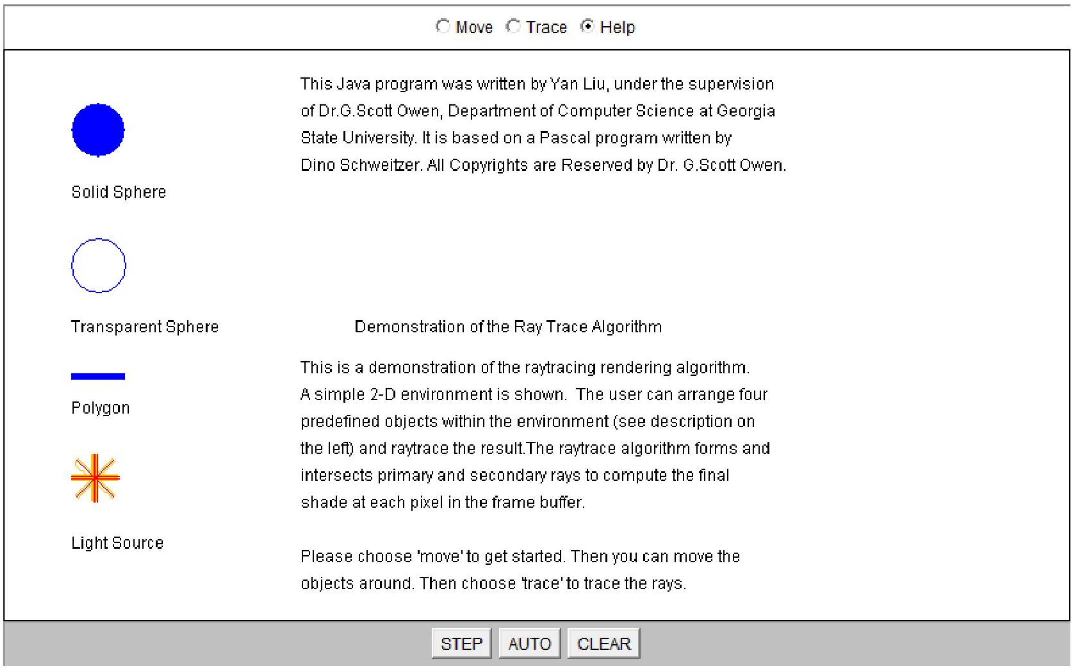
"Forward" RT for Caustics



Ray Tracing **© 2008 Wikipedia**
http://bit.ly/dV7lNm

Adapted from slides ♥ 2001 D. Shreiner & B. Grantham, SCU
COEN 290: Computer Graphics I, Winter 2001 – http://bit.ly/hz1kfU

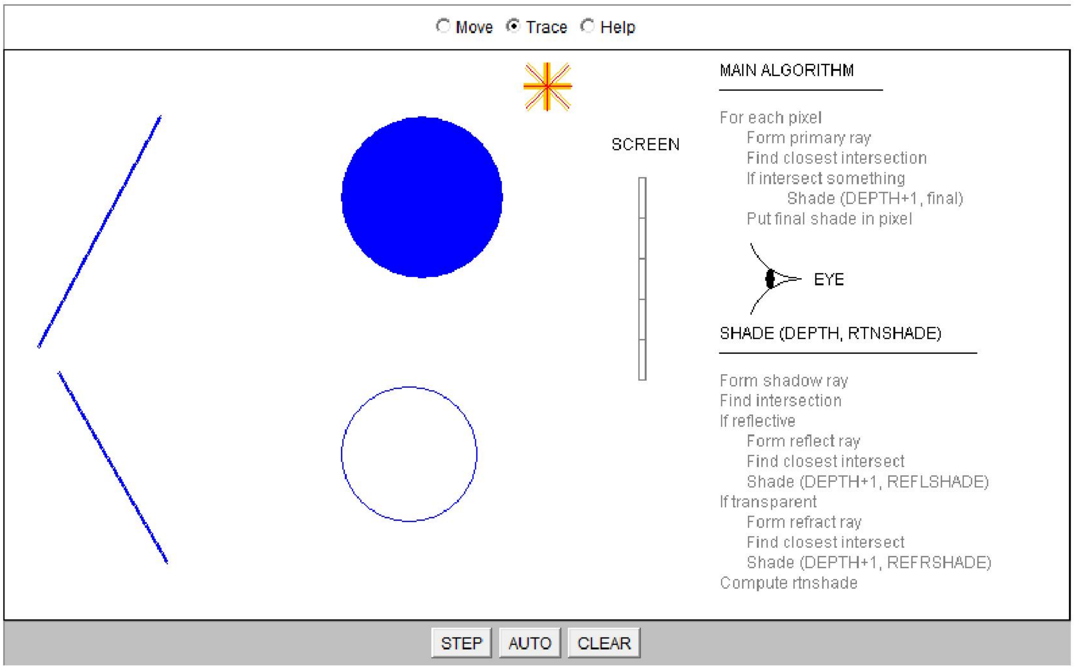
Lab 6a [1]: ACM SIGGRAPH 2-D RT Program Help



Screenshots from Java program ♥ 2001 G. S. Owen & Y. Liu, GSU
ACM SIGGRAPH Ray Trace Java Demo – http://bit.ly/cllgx2

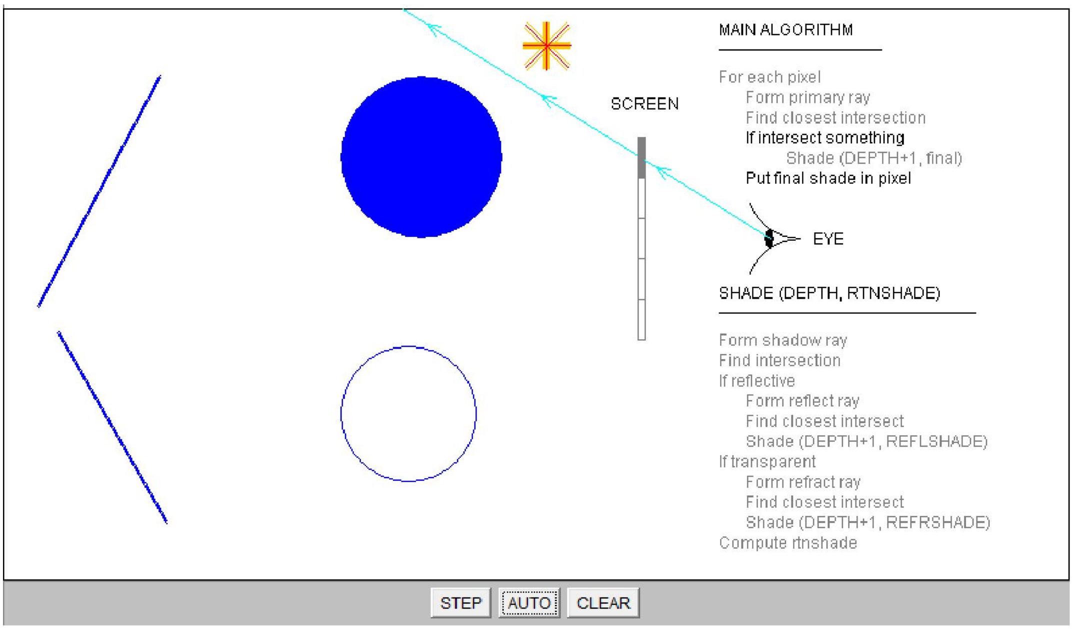
ACMSIGGRAPH

Lab 6a [2]: Trace Screen



Screenshots from Java program ♥ 2001 G. S. Owen & Y. Liu, GSU
ACM SIGGRAPH Ray Trace Java Demo – http://bit.ly/cllgx2

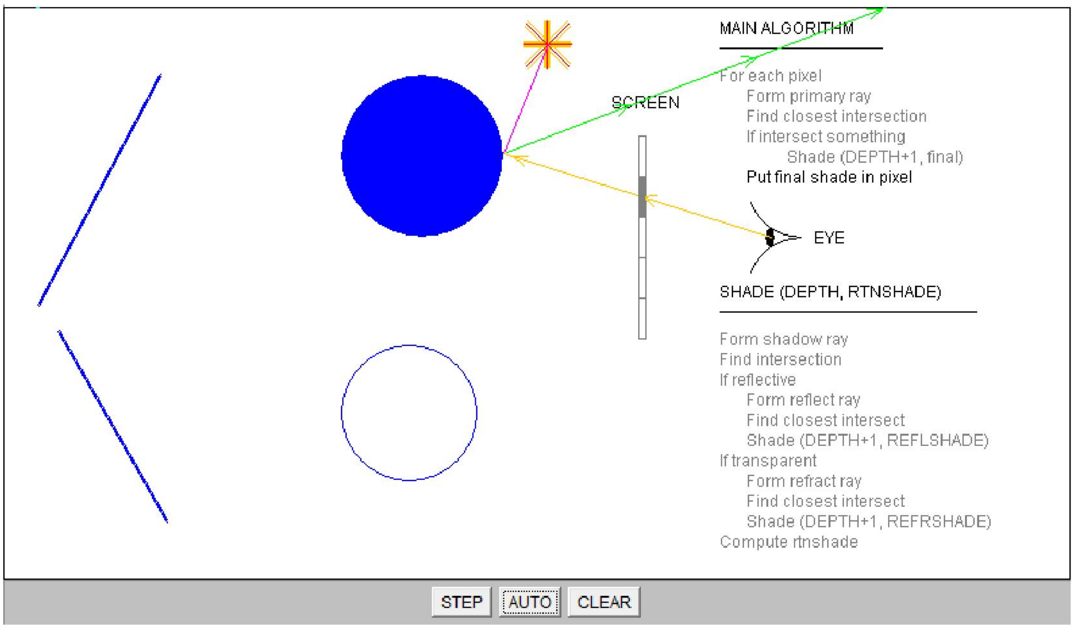
Lab 6a [3]:
First Ray (Click "Clear" & "Auto")



Screenshots from Java program ♥ 2001 G. S. Owen & Y. Liu, GSU
ACM SIGGRAPH Ray Trace Java Demo – http://bit.ly/cllgx2

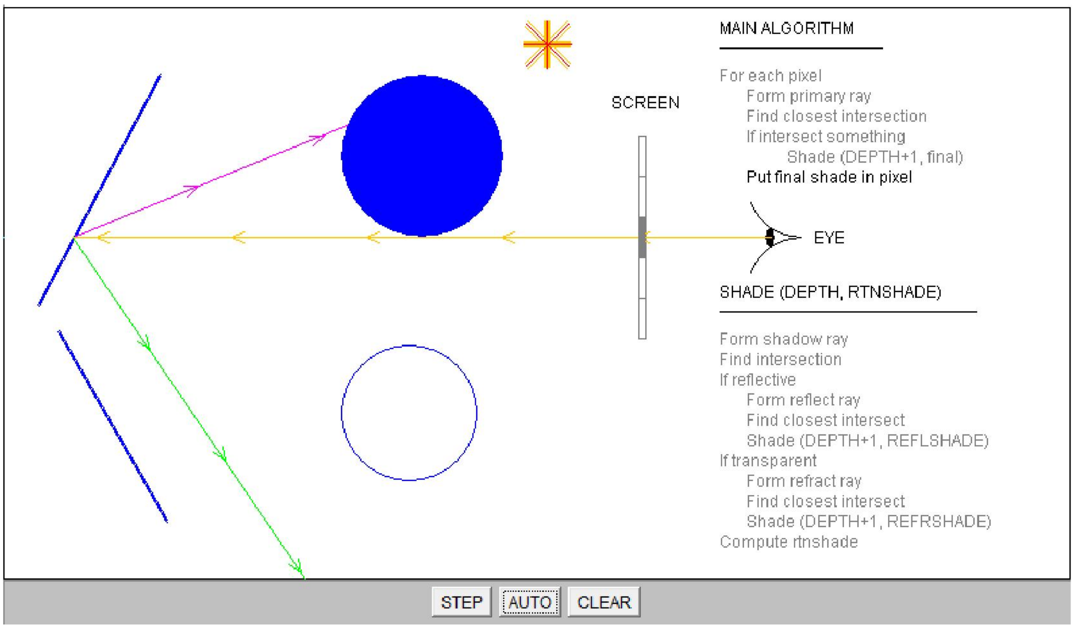
ACMSIGGRAPH

Lab 6a [4]:
Second Ray (Click "Auto" to Advance)



Screenshots from Java program ♥ 2001 G. S. Owen & Y. Liu, GSU
ACM SIGGRAPH Ray Trace Java Demo – http://bit.ly/cllgx2

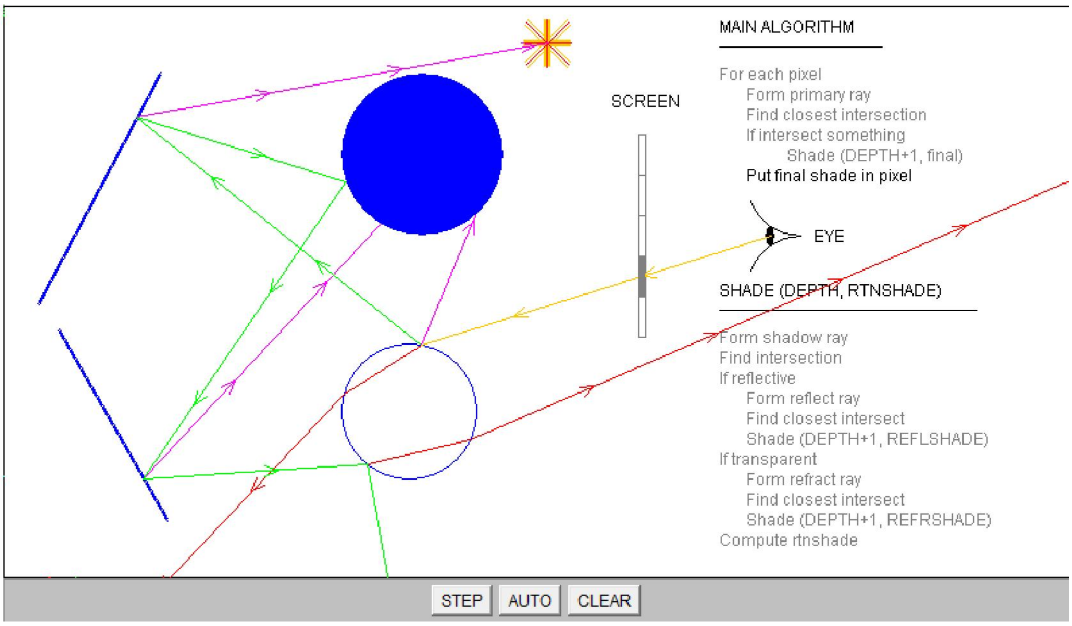
Lab 6a [5]: Third Ray



Screenshots from Java program ♥ 2001 G. S. Owen & Y. Liu, GSU
ACM SIGGRAPH Ray Trace Java Demo – http://bit.ly/cllgx2

ACMSIGGRAPH

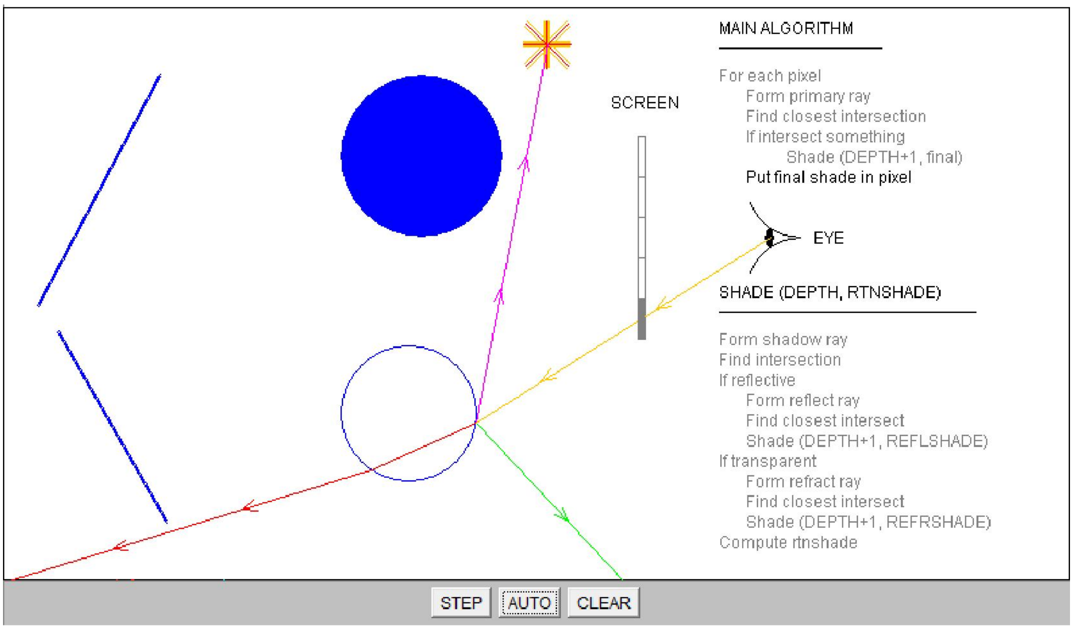
Lab 6a [6]: Fourth Ray



Screenshots from Java program ♥ 2001 G. S. Owen & Y. Liu, GSU
ACM SIGGRAPH Ray Trace Java Demo – http://bit.ly/cllgx2

ACMSIGGRAPH

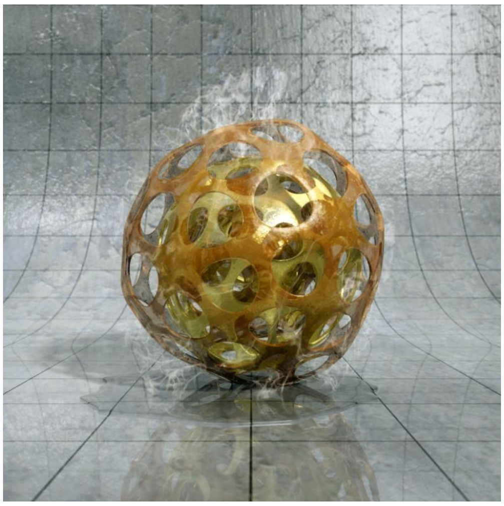
Lab 6a [7]: Fifth Ray



Screenshots from Java program ♥ 2001 G. S. Owen & Y. Liu, GSU
ACM SIGGRAPH Ray Trace Java Demo – http://bit.ly/cllgx2

Lab 6b [1]: POV-Ray

“Office“ © 2004 Jaime Vives Piqueres
http://bit.ly/g9dnGH

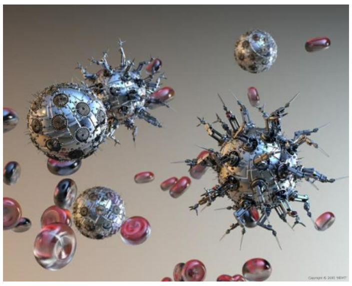


“My First CGSphere“ © 2008 Robert McGregor
http://bit.ly/fGb6Pj

Images ♥ respective authors, generated using *POV-Ray*
© 1991 – 2011 D. K. Buck *et al.* – http://www.povray.org

Lab 6b [2]: POV-Ray

"The Wet Bird" © 2001 Gilles Tran
http://bit.ly/gMBuGt



"Dissolution" © 2005 Newt
http://bit.ly/fVqj5d

"Thanks for all the fish" © 2008 Robert McGregor
http://bit.ly/fE04gm

Images ♥ respective authors, generated using *POV-Ray*
© 1991 – 2011 D. K. Buck *et al.* – http://www.povray.org

Summary

- Reading for Last Class: Chapter 14, Eberly *2*^{e}
- Reading for Today: Ray Tracing Handout
- Reading for Next Class: Chapter 15, Eberly *2*^{e}; Ray Tracing Handout
- Last Time: Ray Tracing (RT), Part 1 of 2
  - Vectors: I (incident ray), L, R, T
  - Basic recursive ray tracing & ray trees
  - Distributed RT: survey, supersampling illustrated
- Today: Ray Tracing Lab
  - ACM SIGGRAPH demo: http://bit.ly/cllgx2
    - 2-D "screen"
    - Moveable objects: transparent, opaque (both reflective)
  - POV-Ray (http://www.povray.org) Example Renderings
- Next Class: Ray Tracing 2 of 2
  - Progressive refinement radiosity (photon maps) introduced
  - Using RT/radiosity together and with shading

Terminology

- Ray Tracing *aka* Ray Casting
  - Given: screen with pixels (*u*, *v*)
  - Find intersection $t_{min}(u, v)$ of rays through each (*u*, *v*) with scene
  - Vectors emanating from world-space coordinate of t_{min}
    - Light (L) *aka* Source (S): to point light sources (or shadows)
    - Reflected (R): from object surface
    - Transmitted or Transparency (T): through transparent object
  - Recursive RT: call raytracer for each intersection, get ray tree
  - Incident vector (I): incoming from eye
- Caustic: Envelope of Light Rays Reflected/Refracted by Curved Object
  - Wikipedia: http://bit.ly/etlXld
  - Example: Slide 13 (today's lecture)
- "Backward" RT: Eye-to-Scene, Scene-to-Light (Typical Order)
- "Forward" RT: Light-to-Scene, Scene-to-Eye (Only for Caustics)
- Screen: Parallel to View "Plane", Rays Shot Through It