Lecture 32 of 41

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

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

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

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

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
  - POV-Ray: http://www.povray.org
- Next Class: Ray Tracing 2 of 2
  - Hybridizing RT with radiosity (photon maps)
  - Progressive refinement
Where We Are

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Title</th>
<th>Lab Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Lab 4: Animation Basics</td>
<td>Flash animation handout</td>
</tr>
<tr>
<td>22</td>
<td>Animation 2: Rotations, Dynamics, Kinematics</td>
<td>Chapter 17, esp. §17.1–17.2</td>
</tr>
<tr>
<td>23</td>
<td>Demos 4: Modeling &amp; Simulation: Rotations</td>
<td>Chapter 10, p. 19, §17.3–17.5</td>
</tr>
<tr>
<td>24</td>
<td>Collisions 1: axes, OBs, Lab 4b</td>
<td>§24.3, 8.1, 9.1, QL handout</td>
</tr>
<tr>
<td>25</td>
<td>Spatial Sorting: Binary Space Partitioning</td>
<td>Chapter 6, esp. 66.1</td>
</tr>
<tr>
<td>26</td>
<td>Demos 5: More CGA: Picking, HW/Exam</td>
<td>Chapter 7, §6.4</td>
</tr>
<tr>
<td>27</td>
<td>Lab 5a: Interaction Handling</td>
<td>§8.3 – 8.4, 4.2, 5.0, 5.6, 9.1</td>
</tr>
<tr>
<td>28</td>
<td>Collisions 2: Dynamic, Particle Systems</td>
<td>§9.1, particle system handout</td>
</tr>
<tr>
<td>29</td>
<td>Exam 2: Hour Exam 2 (evening)</td>
<td>Chapters 5 – 6, 7 – 8, 12, 17</td>
</tr>
<tr>
<td>30</td>
<td>Lab 5b: Particle Systems</td>
<td>Particle system handout</td>
</tr>
<tr>
<td>31</td>
<td>Ray Tracing 1: Intersections, ray tracing</td>
<td>Chapter 14</td>
</tr>
<tr>
<td>32</td>
<td>Lab 6a: Ray Tracing Basics with POV-Ray</td>
<td>Full handout</td>
</tr>
<tr>
<td>33</td>
<td>Ray Tracing 2: Advanced topics survey</td>
<td>Chapter 15, RT handout</td>
</tr>
<tr>
<td>34</td>
<td>Visualization 1: Data (Quantities &amp; Evidence)</td>
<td>Full handout (1)</td>
</tr>
<tr>
<td>35</td>
<td>Lab 6b: More Ray Tracing</td>
<td>RT handout</td>
</tr>
<tr>
<td>36</td>
<td>Visualization 2: Objects</td>
<td>Full handout (2 &amp; 4)</td>
</tr>
<tr>
<td>37</td>
<td>Color Basics, Term Project Prep</td>
<td>Color handout</td>
</tr>
<tr>
<td>38</td>
<td>Lab 7: Fractals &amp; Terrain Generation</td>
<td>Fractals/Terrain handout</td>
</tr>
<tr>
<td>39</td>
<td>Visualization 3: Processes; Final Review 1</td>
<td>Full handout (3)</td>
</tr>
<tr>
<td>40</td>
<td>Project presentations 1: Final Review 2</td>
<td>–</td>
</tr>
<tr>
<td>41</td>
<td>Project presentations 2</td>
<td>–</td>
</tr>
<tr>
<td>42</td>
<td>Final Exam</td>
<td>Ch. 1 – 8, 10 – 15, 17, 20</td>
</tr>
</tbody>
</table>

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:

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

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/
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

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”*. 
Review [3]: Ray/Triangle Intersection

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

\[
p = t_{\text{min}}
\]

\( r_0 \)
\( r_d \)
\( <?, ?, ?> \)
\( (t = ???) \)

Review [4]: General Notation Review

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

```c
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

Calculating surface color

```c
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);
}
```
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
- Improving Image Quality
- Acceleration Techniques

Adapted from slides 2001 D. Shreiner & B. Grantham, SCU

Review [8]:
Distributed Ray Tracing

Adapted from slides 2001 D. Shreiner & B. Grantham, SCU
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

Adapted from slides © 2001 D. Shreiner & B. Grantham, SCU


Review [9]:
Supersampling, “Forward” RT

“Forward” RT for Caustics

Adapted from slides © 2001 D. Shreiner & B. Grantham, SCU


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

- This Java program was written by Nan Liu, under the supervision of Dr. G. S. Owen, Department of Computer Science at Georgia State University. It is based on a Pascal program written by G. S. Owen. All Copyrights are Reserved by Dr. G. S. Owen.

Screenshots from Java program © 2001 G. S. Owen & Y. Liu, GSU

Lab 6a [2]: Trace Screen

Screenshots from Java program © 2001 G. S. Owen & Y. Liu, GSU

Lab 6a [3]: First Ray (Click “Clear” & “Auto”)
Lab 6a [4]:
Second Ray (Click “Auto” to Advance)

Lab 6a [5]:
Third Ray
Lab 6b [1]:
POV-Ray

“Office” © 2004 Jaime Vives Piqueres
http://bit.ly/gBn0H

“My First CGSphere” © 2008 Robert McGregor

Lab 6b [2]:
POV-Ray

“Dissolution” © 2005 Newt
http://bit.ly/6Q4g5d

“The Wet Bird” © 2001 Gilles Tran

“Thanks for all the fish” © 2008 Robert McGregor
Summary

- 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: I (incident ray), L, R, T
  - Basic recursive ray tracing & ray trees
  - Distributed RT: survey, supersampling illustrated
- Today: Ray Tracing Lab
    - 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 \(t_{\text{min}}(u, v)\) of rays through each \((u, v)\) with scene
  - Vectors emanating from world-space coordinate of \(t_{\text{min}}\)
    - 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
  - 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