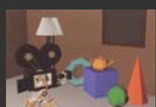
Advanced Computer Graphics (Fall 2009)

CS 294-13, Lecture 1: Introduction and History

Ravi Ramamoorthi

http://inst.eecs.berkeley.edu/~cs294-13/fa09



Some slides courtesy Thomas Funkhouser and Pat Hanrahan

Demo

- Precomputed relighting: Vase
- Real-Time complex shading

Overview

- CS 294-13, Advanced Computer Graphics
  - Prerequisite: Done well in CS 184 or equivalent elsewhere
  - Strong interest in computer graphics
- Advanced topics in rendering/geometry/animation
  - Background for modern topics
  - Areas of current research interest
- Goal is background and up to research frontier
  - Aimed at beginning PhD students and advanced ugrads
- Regular lecture class but less rigid than CS 184
- Encourage you to take other CS 28x, 29x in graphics

Administrivia

- Website http://inst.eecs.berkeley.edu/~cs294-13/fa09
- Co-Instructors James O'Brien and Ravi Ramamoorthi
  - First half of class mostly on rendering (Prof. Ramamoorthi)
  - Second half of class geometry/animation (Prof. O'Brien)
- Lectures MW 1-2:30pm in Soda 310
- E-mail instructors directly for questions, meetings …
  - ravir@cs.berkeley.edu job@cs.berkeley.edu
  - Talk to us after class re issues, getting off waitlist etc.
- **TODO: E-mail us picture (small 120x160), name, e-mail, scribing prefs (at least 3) by tomorrow**

Scribing

- No books. Lectures online, reading/refs as needed
- We request each student scribe 1 or 2 lectures as notes, and for future reference
- Your e-mail should include 3 scribing prefs
  - We will assign scribes by this week and let you know

Course Logistics

- Graded on basis of 4 mostly programming homeworks
- Can be done in groups of two
- Turned in by creating website, sending e-mail
  - Do not modify site after deadline
  - May schedule demo sessions
- *Can substitute research or implementation project for one or more of assignments (encouraged to do so)*
  - With instructor approval of specific plan
  - Allows you to focus on topics of interest and research
- See website for more details

Rendering and Appearance (1st half)

- Core area in computer graphics
- Efficiently and easily create visual appearance
- Long history (1960s to current time): Variety of old and new topics
- From basic visibility and shading, to global illumination, to image-based rendering, to data-driven appearance and light fields
- Many links to physics, math, computer science

Rendering: 1960s (visibility)

- Roberts (1963), Appel (1967) - hidden-line algorithms
- Warnock (1969), Watkins (1970) - hidden-surface
- Sutherland (1974) - visibility = sorting

Images from FvDFH, Pixar's Shutterbug
Slide ideas for history of Rendering courtesy Marc Levoy

Rendering: 1970s (lighting)

1970s - raster graphics

- Gouraud (1971) - diffuse lighting, Phong (1974) - specular lighting
- Blinn (1974) - curved surfaces, texture
- Catmull (1974) - Z-buffer hidden-surface algorithm

Rendering (1980s, 90s: Global Illumination)

early 1980s - global illumination

- Whitted (1980) - ray tracing
- Goral, Torrance et al. (1984) radiosity
- Kajiya (1986) - the rendering equation

Overview of Course

- Weeks 1-2: Basic ray, path tracing and Monte Carlo global illumination rendering
- Weeks 3-7: Topics of current research interest
- Offline Rendering (efficient sampling): Week 3
- Image-Based Rendering: Week 4
- Real-Time Rendering: Weeks 4, 5
- Data-Driven Appearance Acquisition: Week 6
- Other Topics (Light Fields, Sparse Reconstruction)

First Assignment

- In groups of two (find partners)
- Monte Carlo Path Tracer
- If no previous ray tracing experience, ray tracer first.
- See how far you go. Many extra credit items possible, fast multi-dim. rendering, imp. sampling…
- Second assignment: Choice of real-time, precomputation-based and image-based rendering
- Or a research/implementation project of your choice

Outline

- *Basic Ray Tracing*
- Global Illumination
- Image-Based Rendering
- Real-Time Rendering

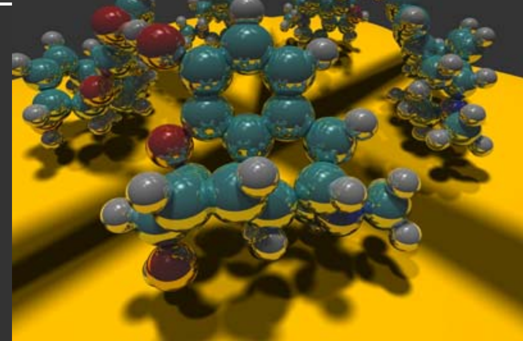


Image courtesy Paul Heckbert 1983

Ray Tracing Basics

Classic Ray Tracing

Greeks: Do light rays proceed from the eye to the light, or from the light to the eye?

Three ideas about light

1. Light rays travel in straight lines (mostly)
2. Light rays do not interfere with each other if they cross (light is invisible!)
3. Light rays travel from the light sources to the eye (but the physics is invariant under path reversal - reciprocity).

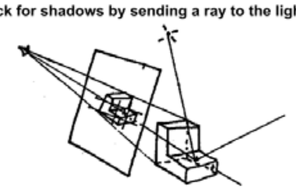
CS348B Lecture 2 Pat Hanrahan, Spring 2009

Ray Tracing History

Ray Tracing in Computer Graphics

Appel 1968 - Ray casting

1. Generate an image by sending one ray per pixel
2. Check for shadows by sending a ray to the light



CS348B Lecture 2 Pat Hanrahan, Spring 2009

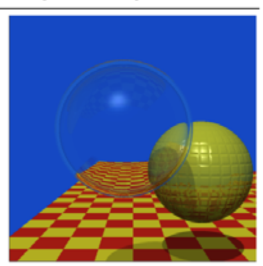
Ray Tracing History

Ray Tracing in Computer Graphics

"An improved Illumination model for shaded display," T. Whitted, CACM 1980

Resolution:
512 x 512

Time:
VAX 11/780 (1979)
74 min.
PC (2006)
6 sec.



Spheres and Checkerboard, T. Whitted, 1979

CS348B Lecture 2 Pat Hanrahan, Spring 2009

Heckbert's Business Card Ray Tracer

```
typedef struct{double x,y,z}vec;vec U,black,amb={.02,.02,.02};struct sphere{ vec cen,color;
double rad,kd,ks,kt,kl,ir}*s,*best,sph[]={0.,6.,.5,1.,1.,1.,.9, .05,.2,.85,0.,1.7,-1.,8.,-.5,1.,.5,.2,1.,
.7,.3,0.,.05,1.2,1.,8.,-.5,.1,.8,.8, 1.,.3,.7,0.,0.,1.2,3.,-6.,15.,1.,.8,1.,7.,0.,0.,0.,.6,1.5,-3.,-3.,12.,
.8,1., 1.,5.,0.,0.,0.,.5,1.5,};yx;double u,b,tmin,sqrt(),tan();double vdot(A,B)vec A ,B;{return A.x
*B.x+A.y*B.y+A.z*B.z;}vec vcomb(a,A,B)double a;vec A,B;{B.x+=a* A.x;B.y+=a*A.y;B.z+=a*A.z;
return B;}vec vunit(A)vec A;{return vcomb(1./sqrt( vdot(A,A)),A,black);}struct sphere*intersect
(P,D)vec P,D;{best=0;tmin=1e30;s= sph+5;while(s-->sph)b=vdot(D,U=vcomb(-1.,P,s->cen)),
u=b*b-vdot(U,U)+s->rad*s ->rad,u=u>0?sqrt(u):1e31,u=b-u>1e-7?b-u:b+u,tmin=u>=1e-7&&
u<tmin?best=s,u: tmin;return best;}vec trace(level,P,D)vec P,D;{double d,eta,e;vec N,color;
struct sphere*s,*l;if(!level--)return black;if(s=intersect(P,D));else return amb;color=amb;eta=
s->ir;d= -vdot(D,N=vunit(vcomb(-1.,P=vcomb(tmin,D,P),s->cen )));if(d<0)N=vcomb(-1.,N,black),
eta=1/eta,d= -d;l=sph+5;while(l-->sph)if((e=l ->kl*vdot(N,U=vunit(vcomb(-1.,P,l->cen))))>0&&
intersect(P,U)==l)color=vcomb(e ,l->color,color);U=s->color;color.x*=U.x;color.y*=U.y;color.z
*=U.z;e=1-eta* eta*(1-d*d);return vcomb(s->kt,e>0?trace(level,P,vcomb(eta,D,vcomb(eta*d-
sqrt (e),N,black))):black,vcomb(s->ks,trace(level,P,vcomb(2*d,N,D)),vcomb(s->kd, color,vcomb
(s->kl,U,black))));}main(){printf("%d %d\n",32,32);while(yx<32*32) U.x=yx%32-32/2,U.z=32/2-
yx++/32,U.y=32/2/tan(25/114.5915590261),U=vcomb(255., trace(3,black,vunit(U)),black),printf
("%.0f %.0f %.0f\n",U);}/*minray!*/
```

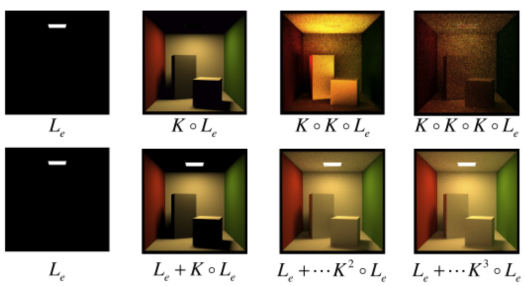
Outline

- Basic Ray Tracing
- *Global Illumination*
- Image-Based Rendering
- Real-Time Rendering

Global Illumination

Radiosity

Successive Approximation

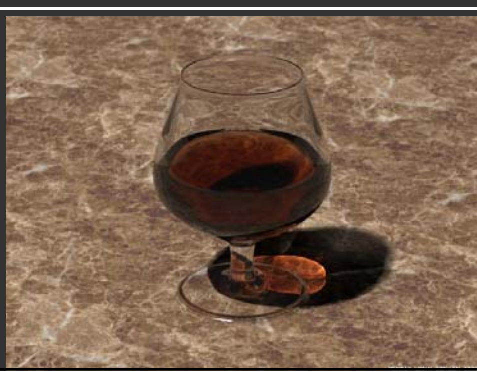


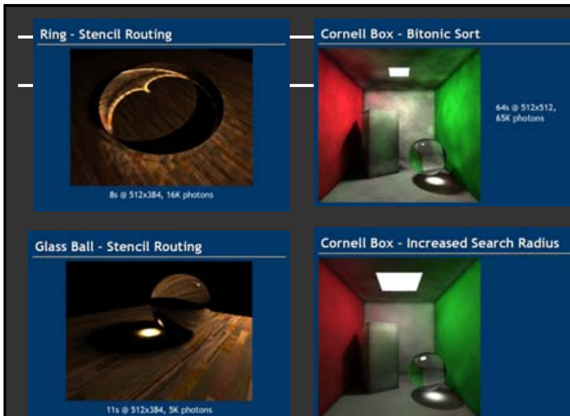
CS348B Lecture 13 Pat Hanrahan, Spring 2009

Rendering Equation (Kajiya 86)

Figure 6. A sample image. All objects are neutral grey. Color on the objects is due to caustics from the green glass balls and color bleeding from the base polygon.

Caustics

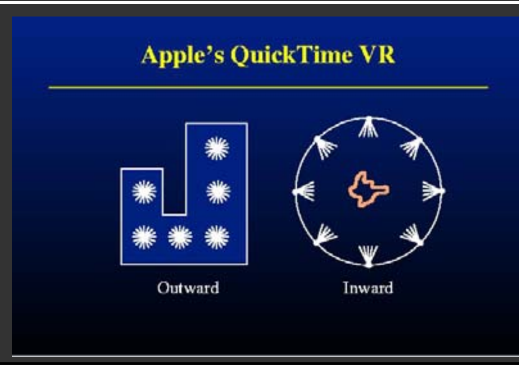


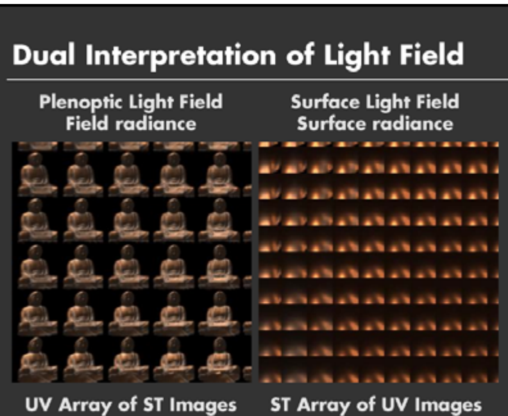


Outline

- Basic Ray Tracing
- Global Illumination
- *Image-Based Rendering*
- Real-Time Rendering

Image-Based Rendering

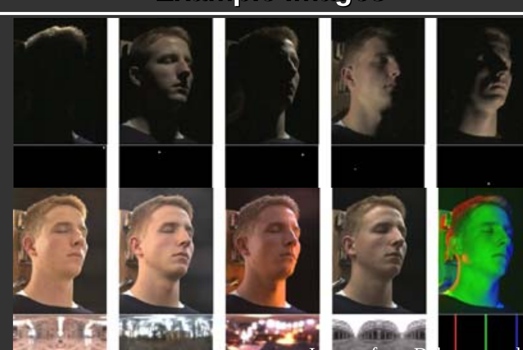




Acquiring Reflectance Field of Human Face [Debevec et al. SIGGRAPH 00]

Illuminate subject from many incident directions

Example Images



Images from Debevec et al. 00

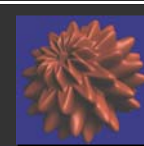
Outline

- Basic Ray Tracing
- Global Illumination
- Image-Based Rendering
- *Real-Time Rendering*

Precomputed Radiance Transfer

- Better light integration and transport
  - dynamic, area lights
  - self-shadowing
  - interreflections
- For diffuse and glossy surfaces
- At real-time rates
- Sloan et al. 02

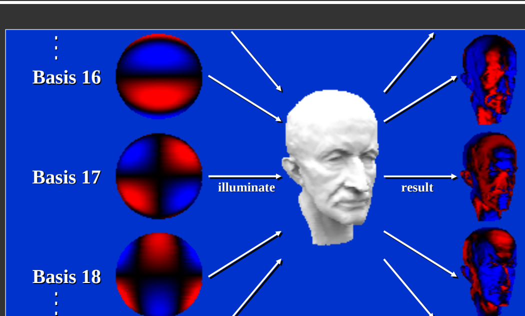

point light


area light


area lighting, no shadows


area lighting, shadows

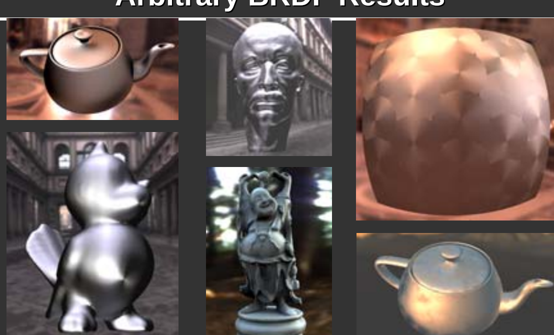
Precomputation: Spherical Harmonics



Diffuse Transfer Results


No Shadows/Inter Shadows Shadows+Inter

Arbitrary BRDF Results


Anisotropic BRDFs Other BRDFs Spatially Varying

Relighting as a Matrix-Vector Multiply

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_N \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1M} \\ T_{21} & T_{22} & \cdots & T_{2M} \\ T_{31} & T_{32} & \cdots & T_{3M} \\ \vdots & \vdots & \ddots & \vdots \\ T_{N1} & T_{N2} & \cdots & T_{NM} \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_M \end{bmatrix}$$