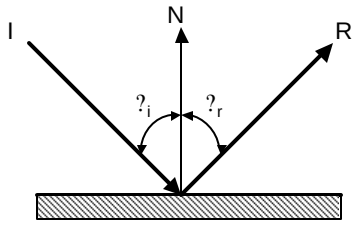
Ray Tracing II

Lecture 18

Ray Reflections



$\vec{R} = ?\vec{I} + ?\vec{N}$

$?_I = ?_r$

Ray Tracing II

Lecture 18

Ray Reflections

$\vec{R} = ?\vec{I} + ?\vec{N}$

$?_I = ?_r$

$\text{Cos}(?_I) = \text{Cos}(?_r)$

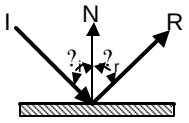
$-\vec{I} ? \vec{N} = \vec{N} ? \vec{R}$

$-\vec{I} ? \vec{N} = \vec{N} ? ?\vec{I} + ?\vec{N}$

$-\vec{I} ? \vec{N} = ? ?\vec{N} ? \vec{I}) + ??\vec{N} ? \vec{N})$

$-\vec{I} ? \vec{N} = ? ?\vec{N} ? \vec{I}) + ?$

$-2(\vec{I} ? \vec{N}) = ?$



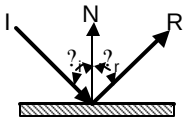
Ray Tracing II

Lecture 18

Ray Reflections

$-2(\vec{I} \cdot \vec{N}) = \cdot$

$\vec{R} = \cdot\vec{I} + \cdot\vec{N}$



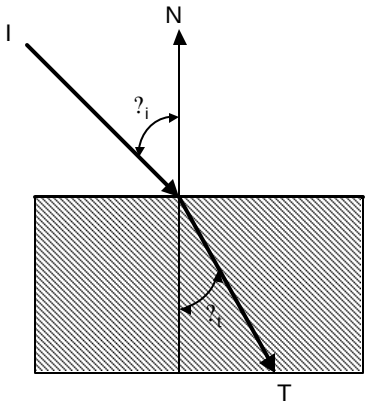
$$\vec{R} = \vec{I} - 2(\vec{N} \cdot \vec{I})\vec{N}$$

3

Ray Tracing II

Lecture 18

Ray Refraction



Snell's Law

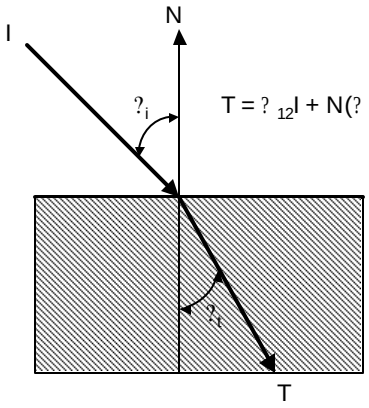
$$\frac{\sin(\cdot_i)}{\sin(\cdot_t)} = \cdot = \cdot_{21} = \frac{\cdot_2}{\cdot_1}$$

4

Ray Tracing II

Lecture 18

Ray Refraction



$$T = ?_{12} I + N(?_{12} \mathrm{Cos}(?_i) - \mathrm{sqrt}(1 + ?^2_{12}(\mathrm{Cos}^2(?_i))))$$

5

Ray Tracing II

Lecture 18

Shadows

To get shadows in ray tracing, at every intersection we shoot a test ray toward each light source.

If the ray gets to the light source then this light is used in the lighting calculation.

If the ray hits something then the light source is not used in the lighting calculation.

The ray direction used to test the light source can be used in the lighting calculation

6

Ray Tracing II

Lecture 18

Anti - Aliasing

- Sample More Rays
- Instead of Vectors use Truncated Cones, Rods, Pyramids
- Stochastic Sampling

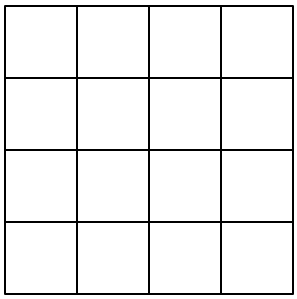
7

Ray Tracing II

Lecture 18

Anti - Aliasing

- Sample More Rays



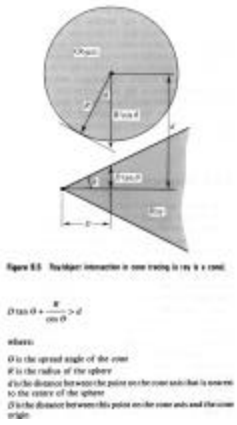
8

Ray Tracing II

Lecture 18

Anti - Aliasing

- Instead of Vectors use Truncated Cones, Rods, Pyramids



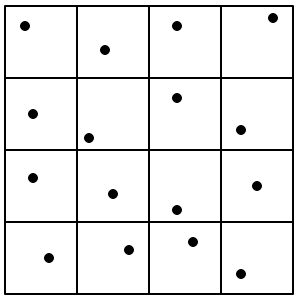
9

Ray Tracing II

Lecture 18

Anti - Aliasing

- Stochastic Sampling



10

Ray Tracing II

Lecture 18

Anti - Aliasing

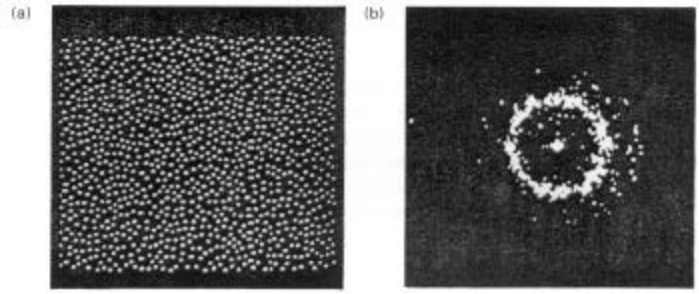


Fig. 3. (a) Monkey eye photoreceptor distribution. (b) Optical transform of monkey eye.

11

Ray Tracing II

Lecture 18

Temporal Anti - Aliasing

Motion Blur

Sub-pixel samples are stochastically sampled over time and space

12

Ray Tracing II

Lecture 18

Temporal Anti - Aliasing

13

Ray Tracing II

Lecture 18

Temporal Anti - Aliasing

14

Ray Tracing II

Lecture 18

Forward Ray Tracing

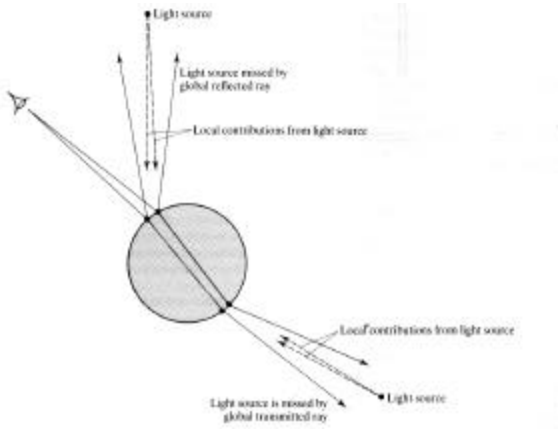
This means tracing rays from the light source into the scene.

- This can be very expensive
- But adds effect that can only be found if this is done.
- One of the ways to get water caustics

Ray Tracing II

Lecture 18

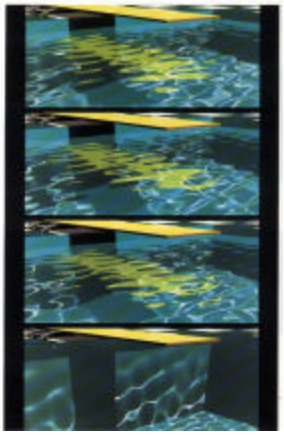
Forward Ray Tracing



Ray Tracing II

Lecture 18

Forward Ray Tracing

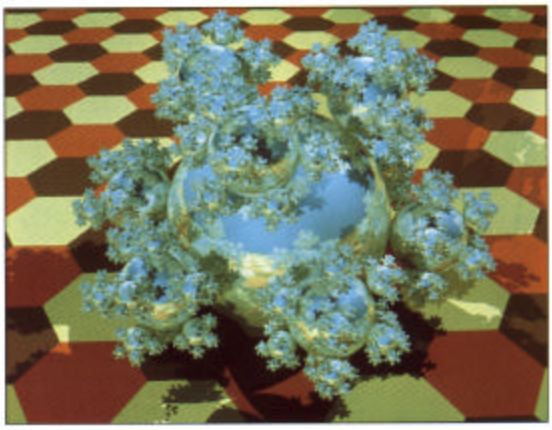


17

Ray Tracing II

Lecture 18

Sphereflake



18

Ray Tracing II

Lecture 18

MinRay

```
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!*/
```


Ray Tracing II

Lecture 18

MinRay

Those who entered the contest also learned some repulsive C coding tricks:

- color = point: `struct {float r, g, b;}` ⇒ `struct {float x, y, z;}`
- pass structures (not just pointers to structs) to allow vector expressions
- use global variables for return values
- no optimizations: trace specular and transmitted rays even if coeff=0!
- reuse variables!
- `for (i=0; i<n; i++)` ⇒ `i = n; while (i--)`
- merge x and y loops into one
- assume statics are initialized to 0
- `&sph[i]` ⇒ `sph+i`
- choose just the right set of utility routines
- creative use of commas, e.g.: `if (a) {b;c;)` ⇒ `if (a) b,c;`
- move assignments into expressions: `b = vdot(D, U = vcomb(-1., P, s->cen))`

Cheats (non-portable C):

- eliminate semicolons in struct def: `struct {int a;}` ⇒ `struct {int a}`
- assume right-to-left argument evaluation order

Winner of the most shocking cheat award, a little gem by Joe Cychosz:

- `printf("%f %f %f\n", pt.x, pt.y, pt.z);` ⇒ `printf("%f %f %f\n", pt);`

Since Joe Cychosz said this was only his second C program (!), I asked him how he managed to do so well. He responded:

> "My primary program language is FORTRAN and COMPASS (CDC assembly) for the CYBER 205 and the CYBER 800s.... This is an interesting way to learn C. I spent the weekend looking through Kernighan & Ritchie trying to determine if what I wanted to do was legal C. Finally, Kirk Smith found 22 tokens while we were sitting in a bar."

But if you were expecting a useful ray tracing program out of this, a warning: as one would expect of any "minimal" program, the winning program is cryptic, inefficient, unmaintainable, and nearly useless except as a source of C coding tricks. If you want a good ray tracer, look elsewhere!

Ray Tracing II

Lecture 18

MinRay

```
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,.1,.2,3,-.6,.15,.1,.1,.8,1,.7,.0,.0,.0,1.5,-3,.5,12,.8,1,.
1,.5,.8,.0,.0,.5,1.5,};vec 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>=0?sqrt(u):1e31,u=b-u>1e-7?b-u:b+u,tmin>u&&u>1e-7&&(tmin=u)?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(y<32*32)
U.x=y%32-32/2,U.z=32/2/tan(25/114.591559026),U.y=32/2-y++/32,U=vcomb(255.,
trace(3,black,vunit(U)),black),printf("%.0f %.0f %.0f\n",U);}/*minray!*/
```