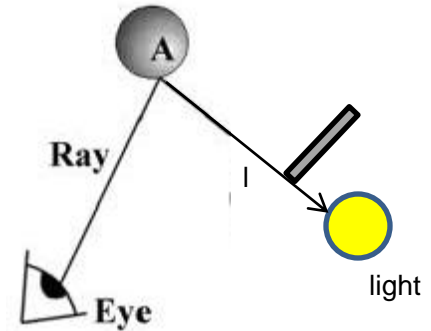# Ray Tracing (Part 2)

CS334 Spring 2025

Daniel G. Aliaga
Department of Computer Science
Purdue University

(Hard) Shadows

- If ray l is unoccluded from surface point to light source, then surface point is illuminated (i.e., <u>not</u> in shadow)

- Use same recursive rayTrace function but cast ray from surface point to light
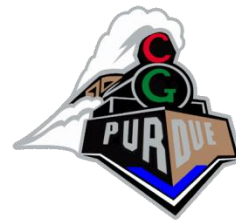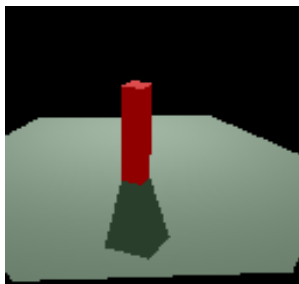
Soft Shadows

- Lights are actually areas, so use area light sources

- Using areas enables both umbra and penumbra to appear

light source

shadow object

umbra   penumbra
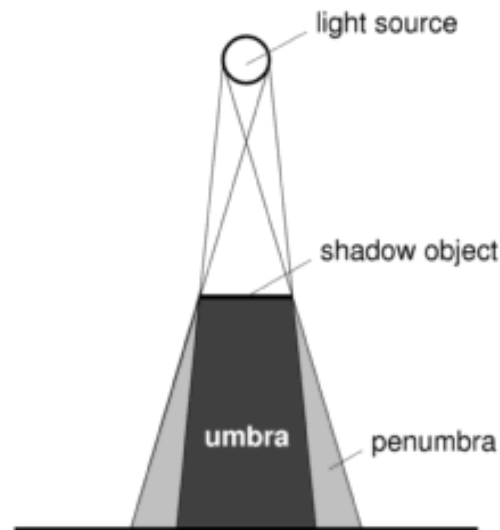
Soft Shadows

- Lights are actually areas, so use use area light sources

- *Distributed Ray Tracing*
  - Replace light with a collection of point light sources (e.g., up to 50 rays jittered samples)
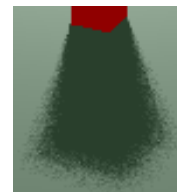

Example Scene

1 light ray
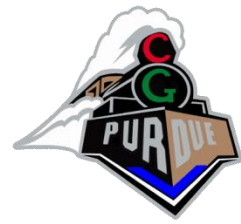
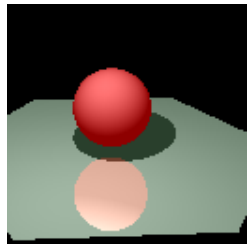Uniformly sampled light rays

10 light rays

20 light rays

50 light rays

Distributed Ray Tracing

- Soft shadows (previous slide)
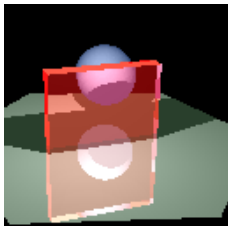
- Glossy Reflections

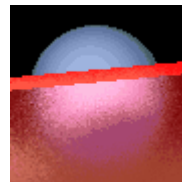1 ray   10 rays   20 rays   50 rays
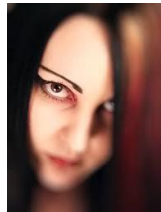
- Fuzzy Transluceny

1 ray   10 rays   20 rays

- Defocus

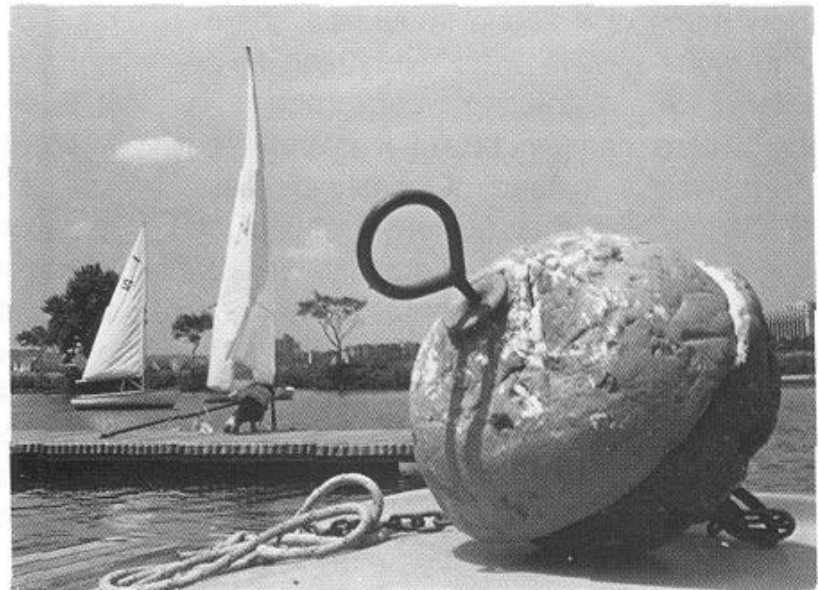(examples using per-pixel and ray tracing logic)

Depth of Field

- The area in front of your camera where everything looks sharp and in focus.
  - objects falling within that area will be acceptably-sharp and in focus;
  - objects falling outside the area will be soft and out of focus.
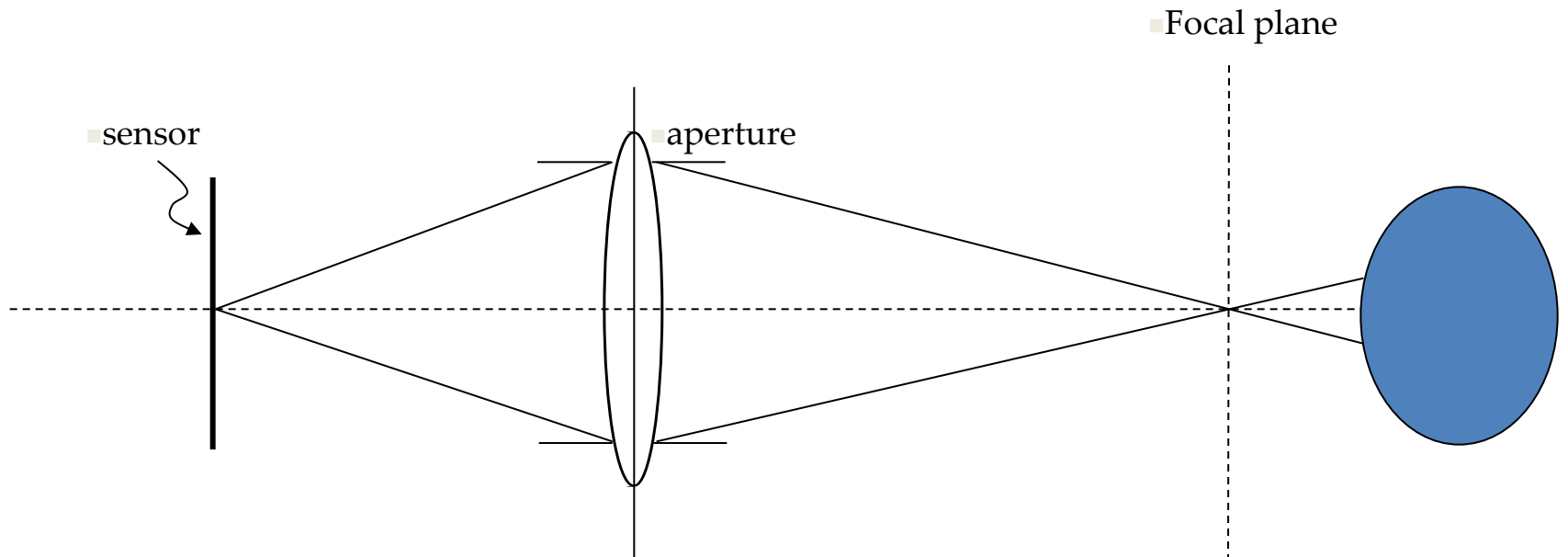
less depth of field

more depth of field

wider aperture

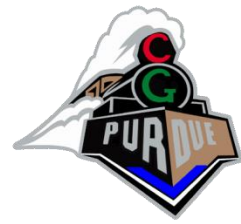smaller aperture

Depth of Field

sensor

aperture

Focal plane

Depth of Field

Focal plane

sensor

aperture
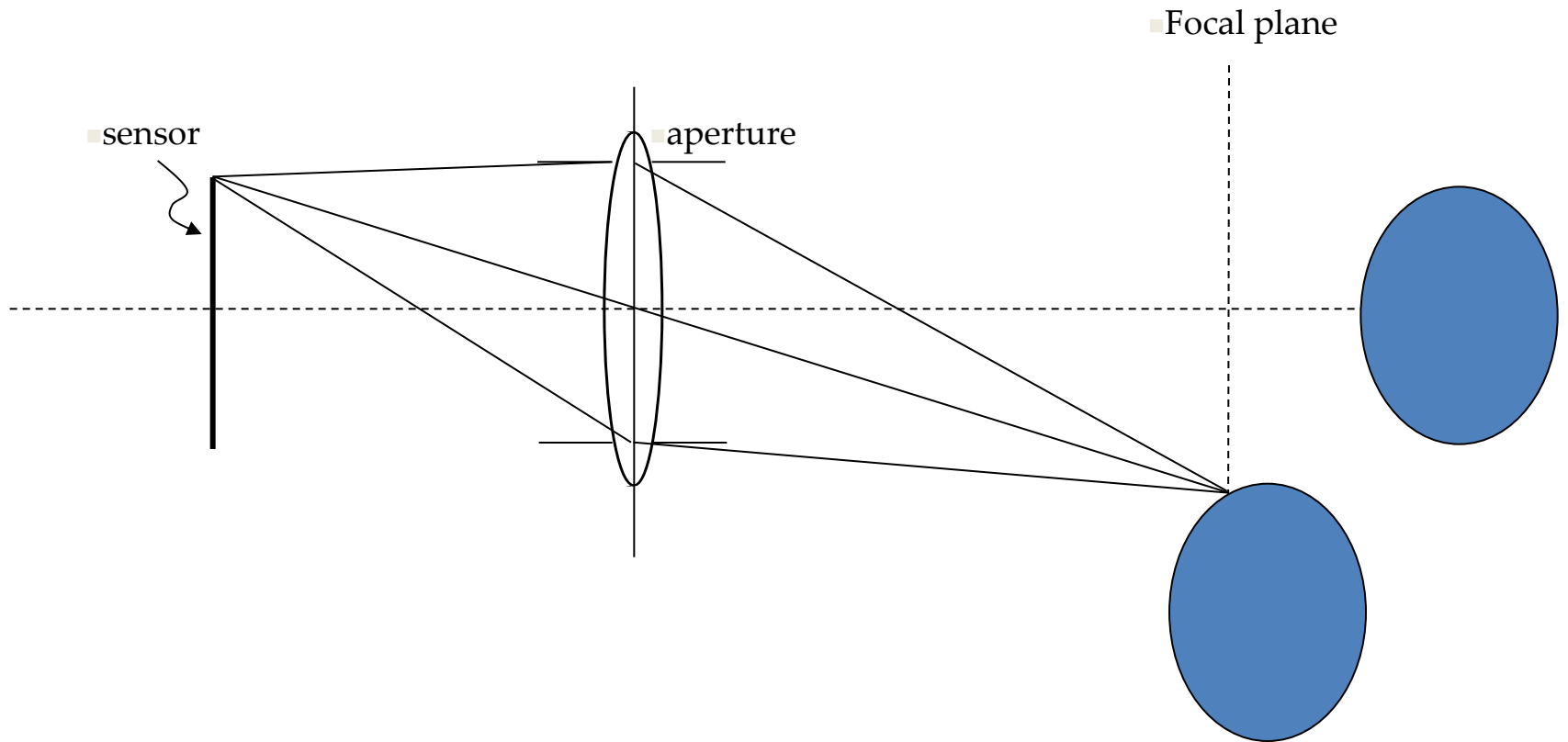
Computer Graphics Camera Models

- Pinhole – ideal camera

- All rays go through single point

- Everything in focus -- unrealistic
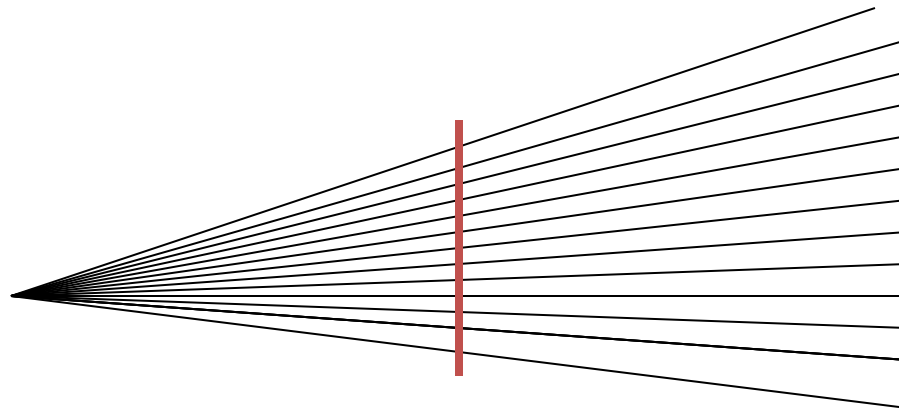
More Realistic Model

- Lenses with spherical surfaces
- Depth of field control

Depth of Field

focal plane

image plane

V_D

D

V_P aperture P
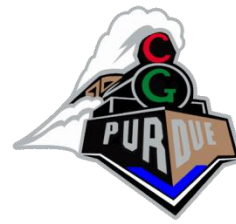
A

C

R
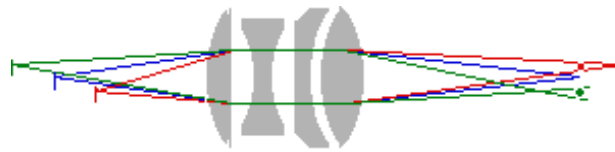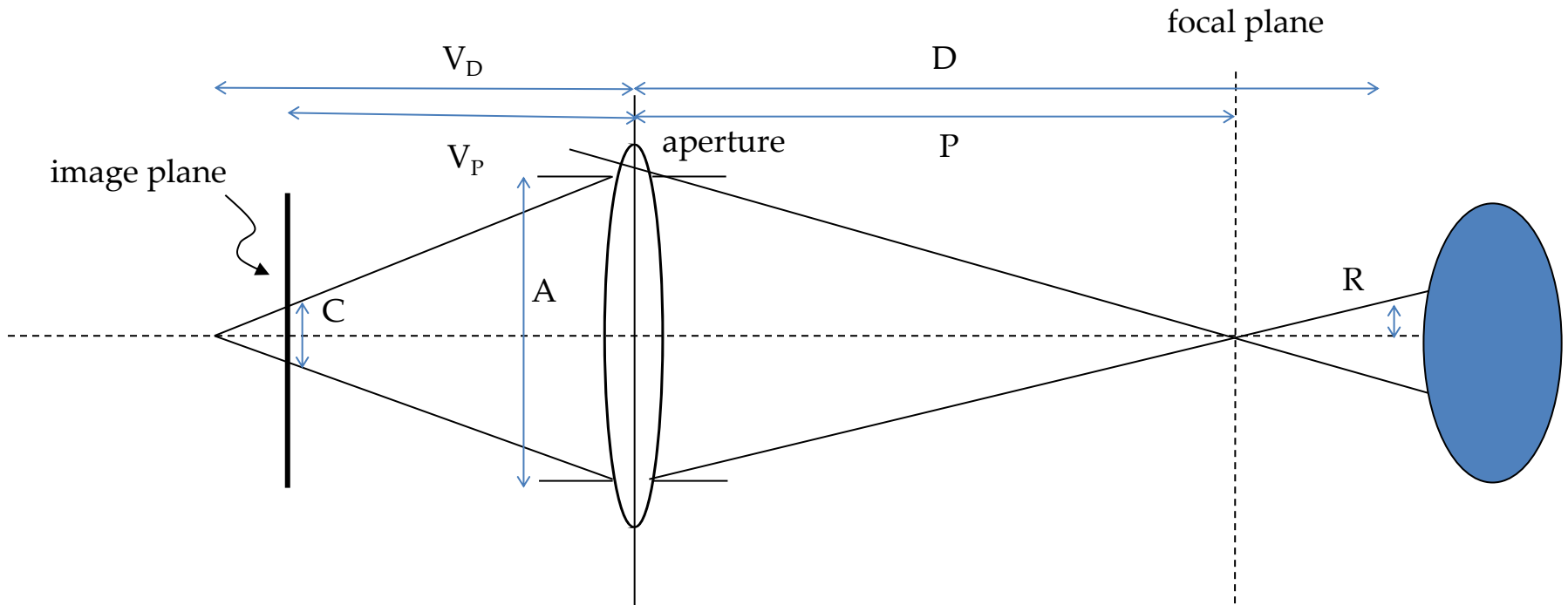
$V_P = AP/(P-A)$ for P>A $C = |V_D - V_P| \, (A/V_P)$ "=circle of confusion"

$V_D = AD/(D-A)$ for D>A $R = 0.5 \, A \, (\, |D-P| \, / \, P \,)$
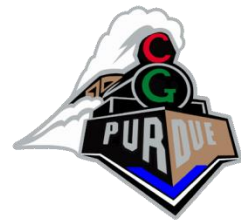
Depth of Field

Example
Results
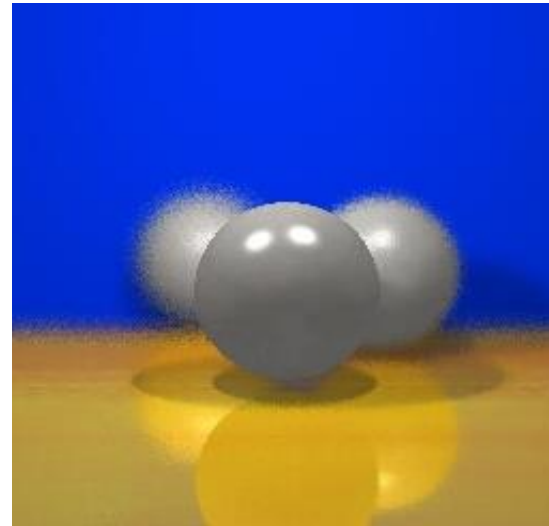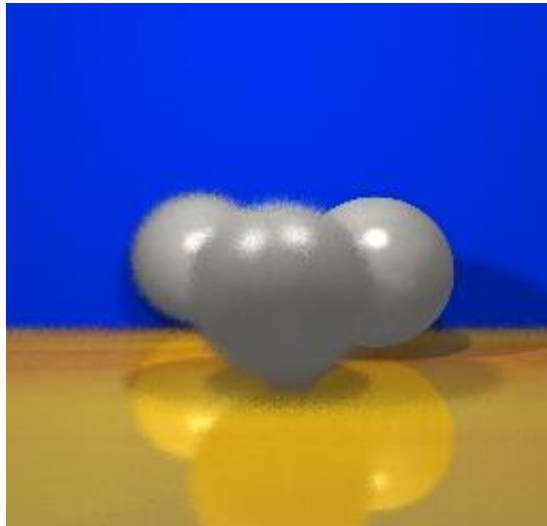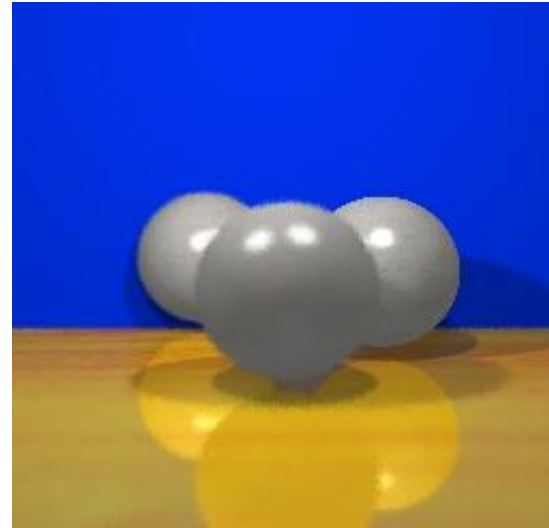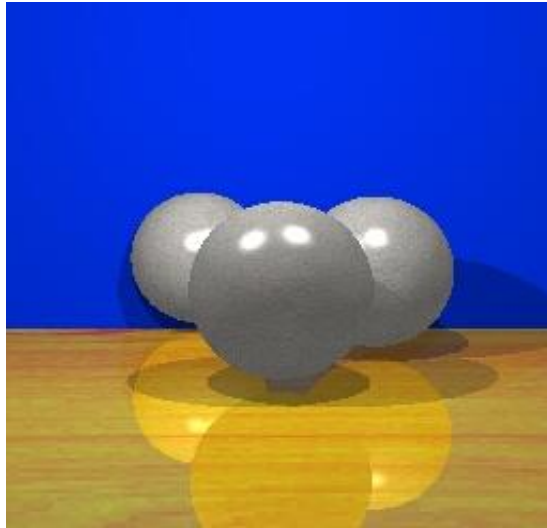
Depth of Field: Out of Focus Blur

- How to approximate without actually creating an entire physically based rendering system?

- Basic idea:
  - You want something at distance "P" to be have its rays converge
  - So think backwards: how can you distort multiple rays per pixel so that they converge at distance P but not otherwise?

Example Real-Time Ray Tracer
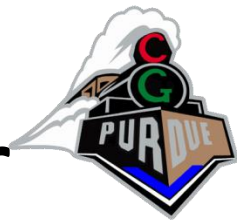
- Large Scale Voxel Renderer using Ray Tracing

  "Efficient Sparse Voxel Octrees", Samuli and Karras, ACM I3D, 2010

  Paper = http://www.tml.tkk.fi/~samuli/publications/laine2010i3d_paper.pdf

  Video = http://www.tml.tkk.fi/~samuli/publications/laine2010i3d_video.avi

Minimalistic Ray Tracer
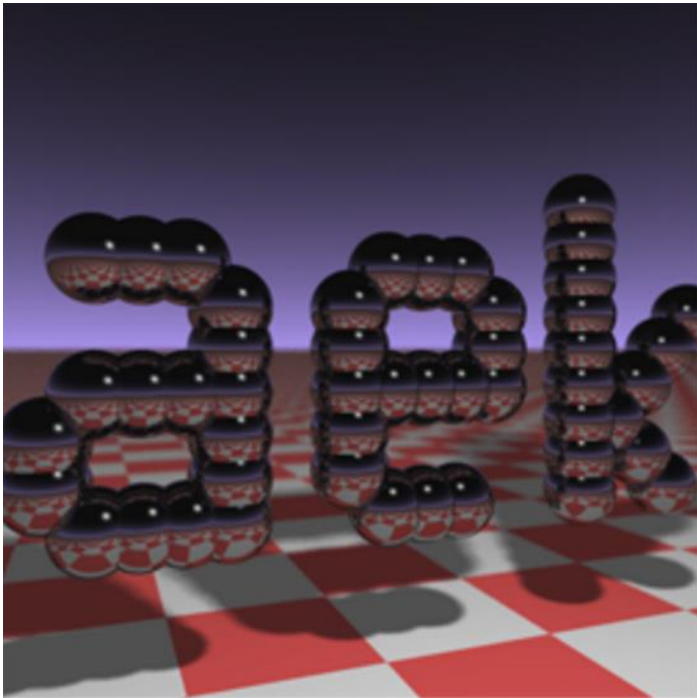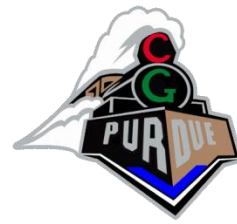
```
#include <stdlib.h> // card > aek.ppm #include <stdio.h>
#include <math.h> typedef int i;typedef float f;struct v{f x,y,z;v
operator+(v r){return v(x+r.x,y+r.y,z+r.z);}v operator*(f r){return
v(x*r,y*r,z*r);}f operator%(v r){return x*r.x+y*r.y+z*r.z;}v(){}v
operator^(v r){return v(y*r.z-z*r.y,z*r.x-x*r.z,x*r.y-y*r.x);}v(f a,f
b,f c){x=a;y=b;z=c;}v
operator!(){return*this*(1/sqrt(*this%*this));}};i
G[]={247570,280596,280600,249748,18578,18577,231184,16,1
6};f R(){return(f)rand()/RAND_MAX;}i T(v o,v d,f&t,v&n){t=1e9;i
m=0;f p=-o.z/d.z;if(.01<p)t=p,n=v(0,0,1),m=1;for(i k=19;k--;)for(i
j=9;j--;)if(G[j]&1<<k){v p=o+v(-k,0,-j-4);f b=p%d,c=p%p-1,q=b*b-
c;if(q>0){f s=-b-
sqrt(q);if(s<t&&s>.01)t=s,n=!(p+d*t),m=2;}}return m;}v S(v o,v
d){f t;v n;i m=T(o,d,t,n);if(!m)return v(.7,.6,1)*pow(1-d.z,4);v
h=o+d*t,l=!(v(9+R(),9+R(),16)+h*-1),r=d+n*(n%d*-2);f
b=l%n;if(b<0||T(h,l,t,n))b=0;f
p=pow(l%r*(b>0),99);if(m&1){h=h*.2;return((i)(ceil(h.x)+ceil(h.
y))&1?v(3,1,1):v(3,3,3))*(b*.2+.1);}return v(p,p,p)+S(h,r)*.5;}i
main(){printf("P6 512 512 255 ");v g=!v(-6,-
16,0),a=!(v(0,0,1)^g)*.002,b=!(g^a)*.002,c=(a+b)*-256+g;for(i
y=512;y--;)for(i x=512;x--;){v p(13,13,13);for(i r=64;r--;){v
t=a*(R()-.5)*99+b*(R()-.5)*99;p=S(v(17,16,8)+t,!(t*-
1+(a*(R()+x)+b*(y+R())+c)*16))*3.5+p;}printf("%c%c%c",(i)p.x,(i
)p.y,(i)p.z);}}
```

Evan's Demos

- PathTracer (not quite a ray tracer but almost, and its pretty cool):
  - http://madebyevan.com/webgl-path-tracing/

- Water (pretty cool preview of what next):
  - http://madebyevan.com/webgl-water/

Ray Tracing Explained

- https://www.youtube.com/watch?v=gBPNO6ruevk

- By Eric Haines, author of "Real-time Rendering" and current NVIDIA researcher

- (9 minutes)