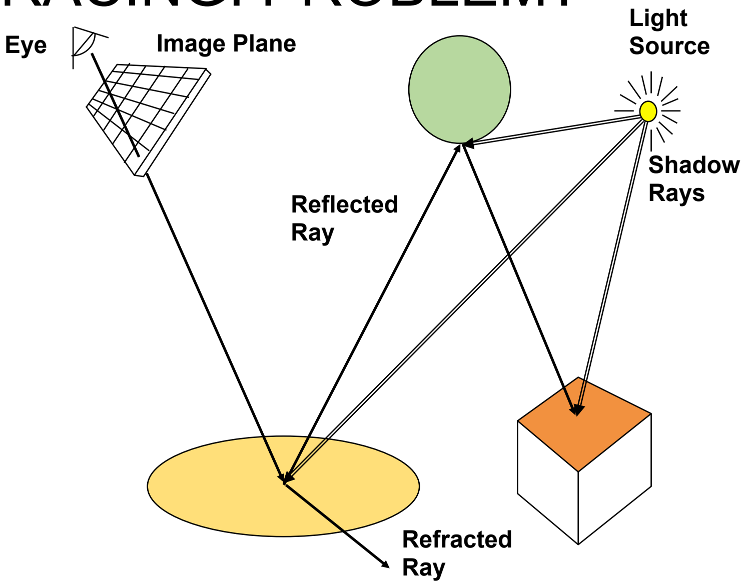
CPSC 314
GLOBAL ILLUMINATION
PATH TRACING.

UGRAD.CS.UBC.CA/~CS314

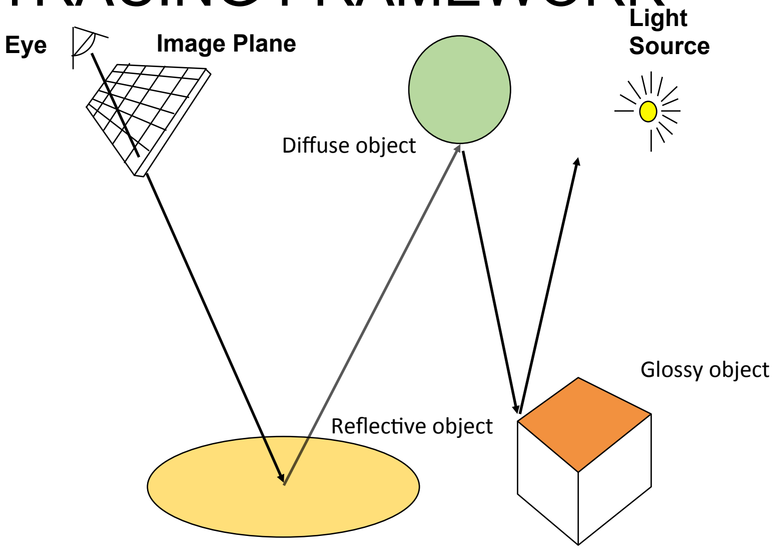
RAY TRACING: PROBLEM?

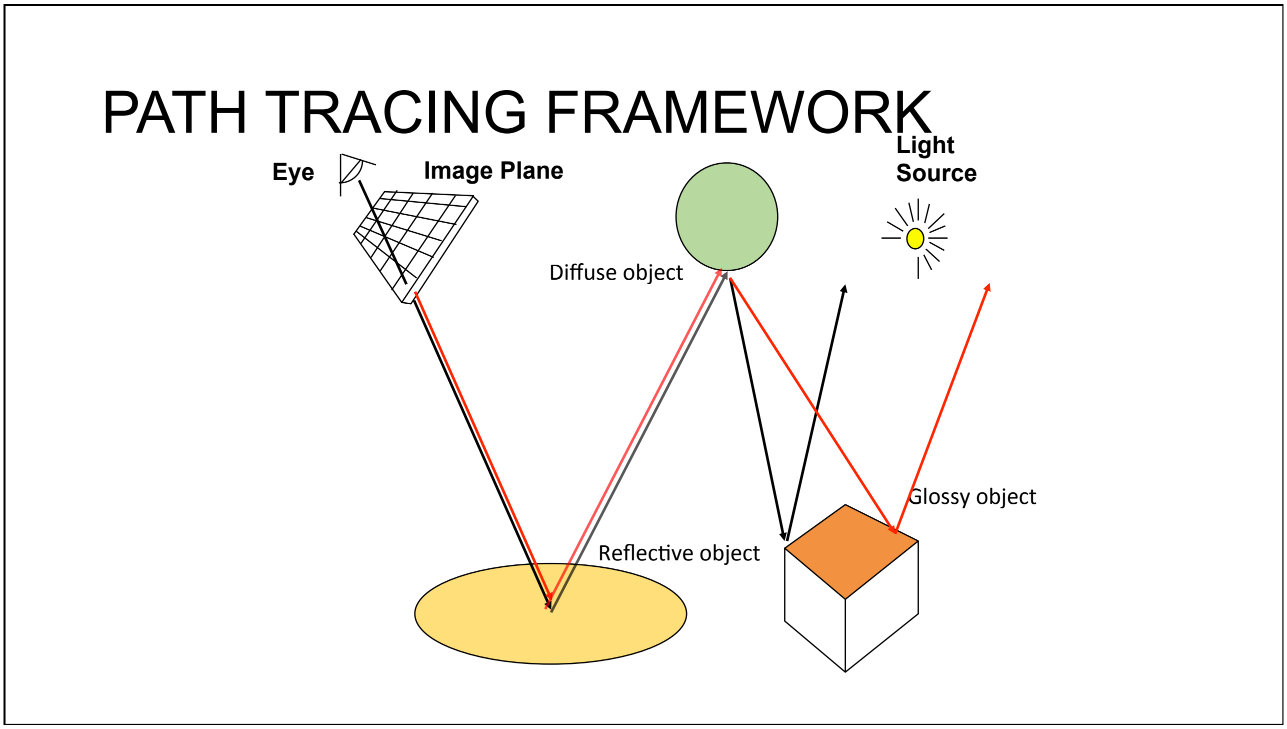


RAY TRACING LIMITATION

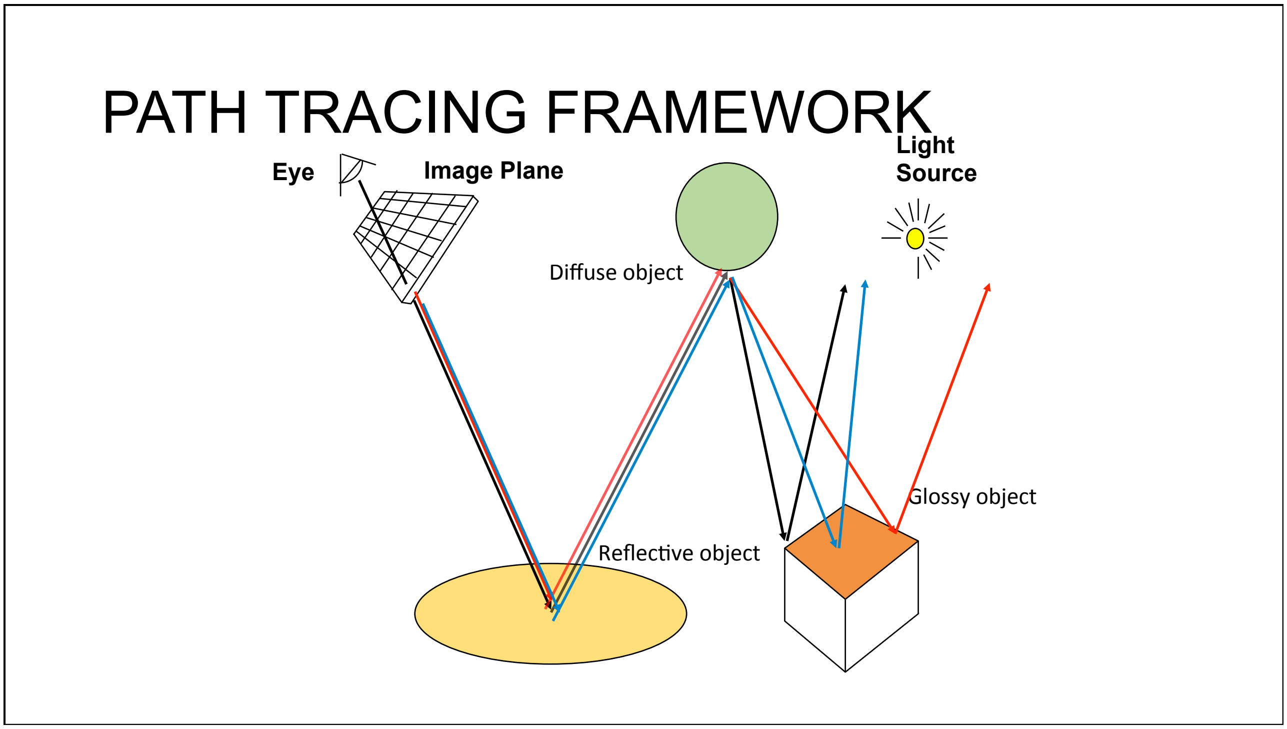
- Only specular reflections consider other objects
  - Well okay, refractions too
- Diffuse and glossy surfaces will only reflect light source!
- We're using Lambert and Phong models of **direct illumination**
- How can we model diffuse/glossy models in a similar way?

PATH TRACING FRAMEWORK



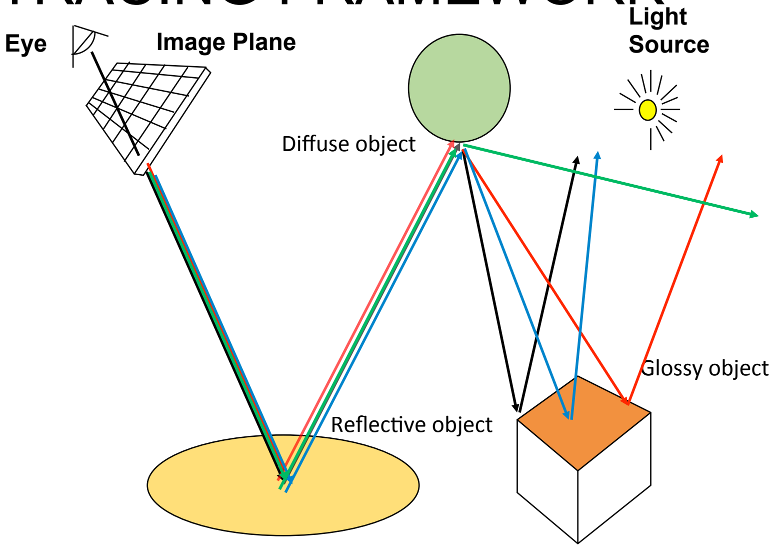


PATH TRACING FRAMEWORK



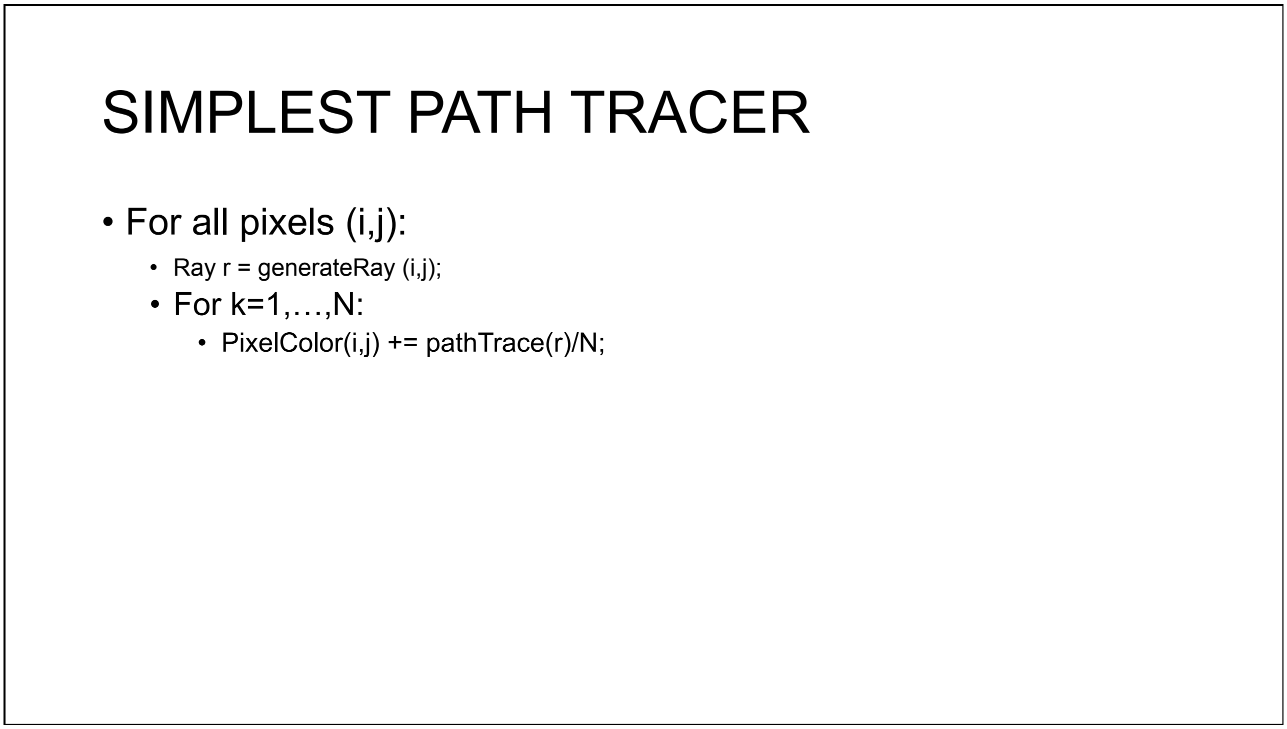
PATH TRACING FRAMEWORK

PATH TRACING FRAMEWORK



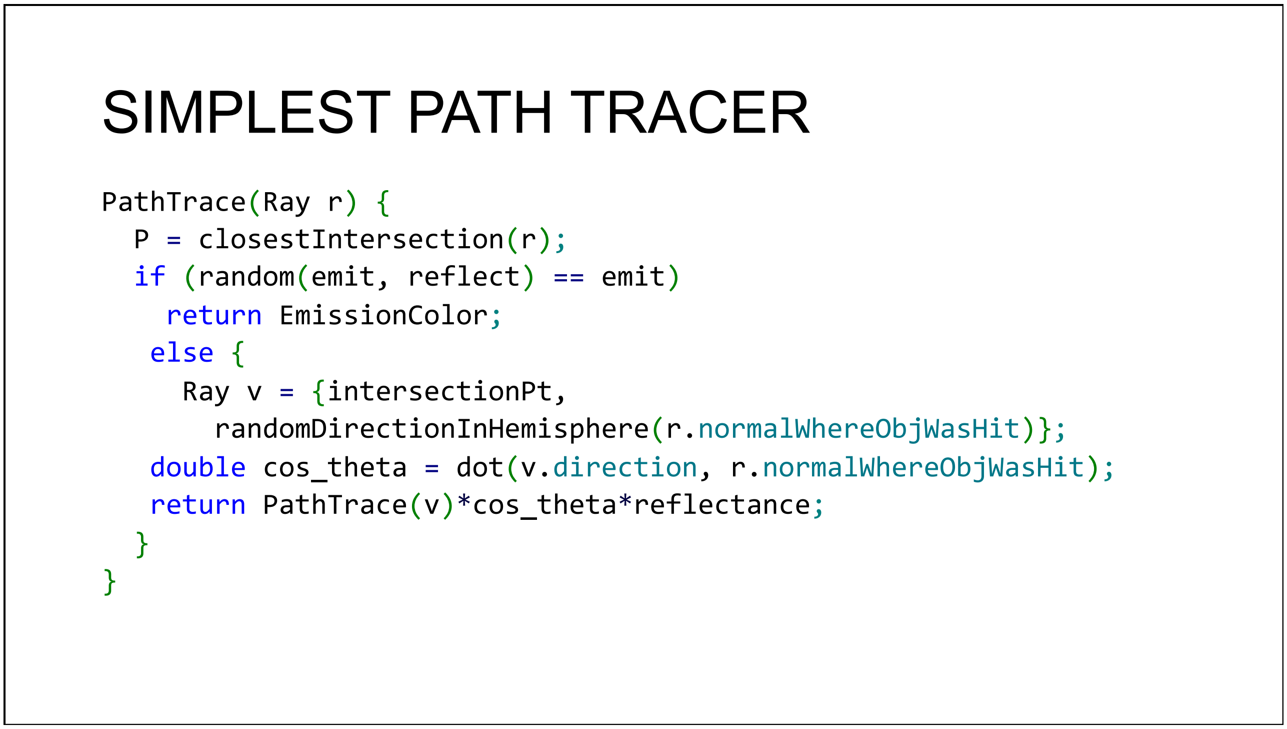
PATH TRACING ALGORITHM

1. Shoot a ray though pixel (i,j). Set *attenuation* to 1.0.
2. Find the closest intersection of the ray with an object
3. Randomly choose between "*emission*" and "*reflection*"
    a. If "emission", return **emissionColor;**
    b. If "reflection",
        Reflect a ray in a random direction
        rayWeight *= reflectance;
        Go to 2.



SIMPLEST PATH TRACER

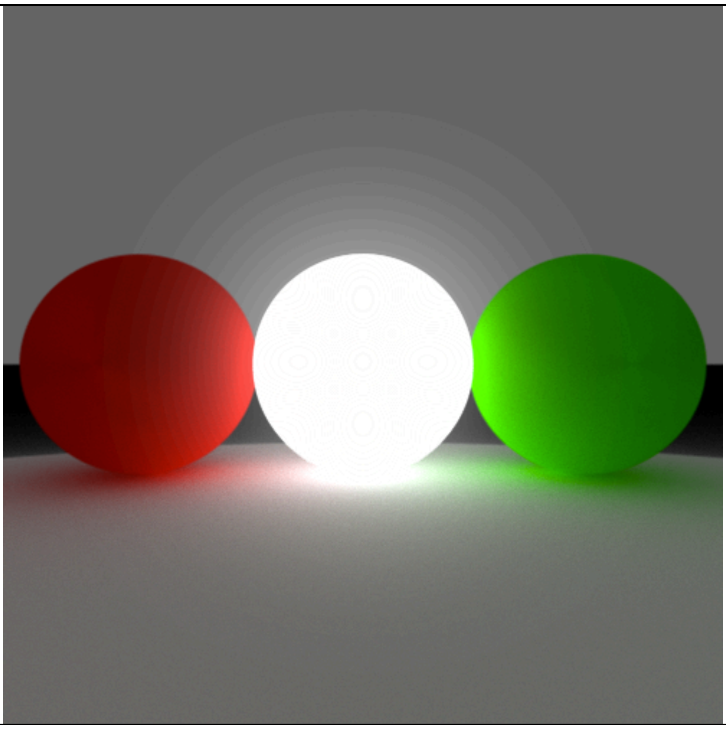
- For all pixels (i,j):
  - Ray r = generateRay (i,j);
  - For k=1,…,N:
    - PixelColor(i,j) += pathTrace(r)/N;



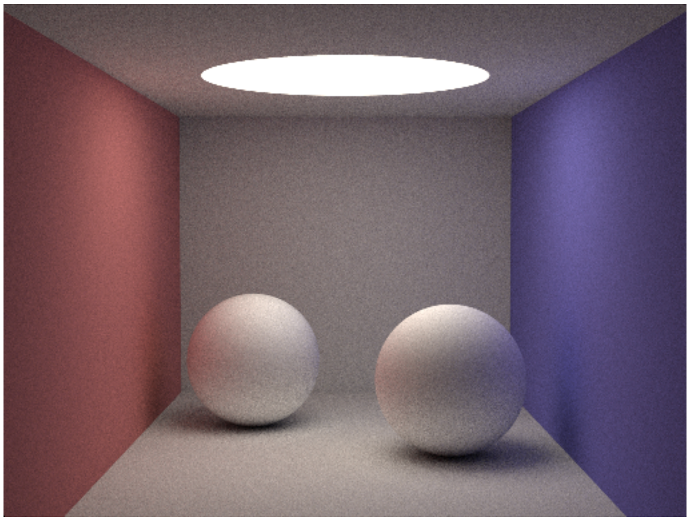
SIMPLEST PATH TRACER

```
PathTrace(Ray r) {
  P = closestIntersection(r);
  if (random(emit, reflect) == emit)
    return EmissionColor;
   else {
     Ray v = {intersectionPt,
       randomDirectionInHemisphere(r.normalWhereObjWasHit)};
   double cos_theta = dot(v.direction, r.normalWhereObjWasHit);
   return PathTrace(v)*cos_theta*reflectance;
  }
}
```

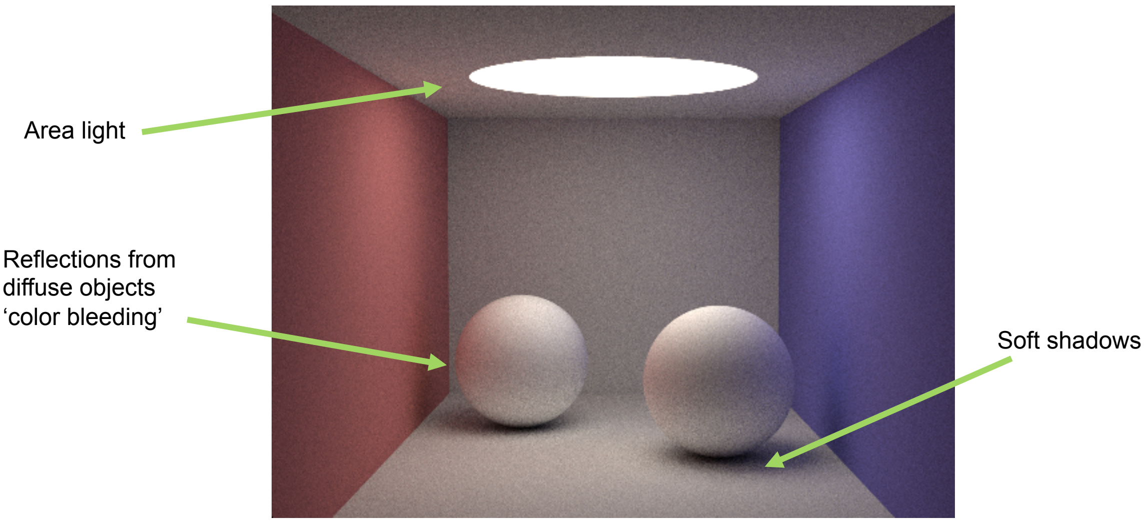
EXAMPLE



WHAT YOU GET



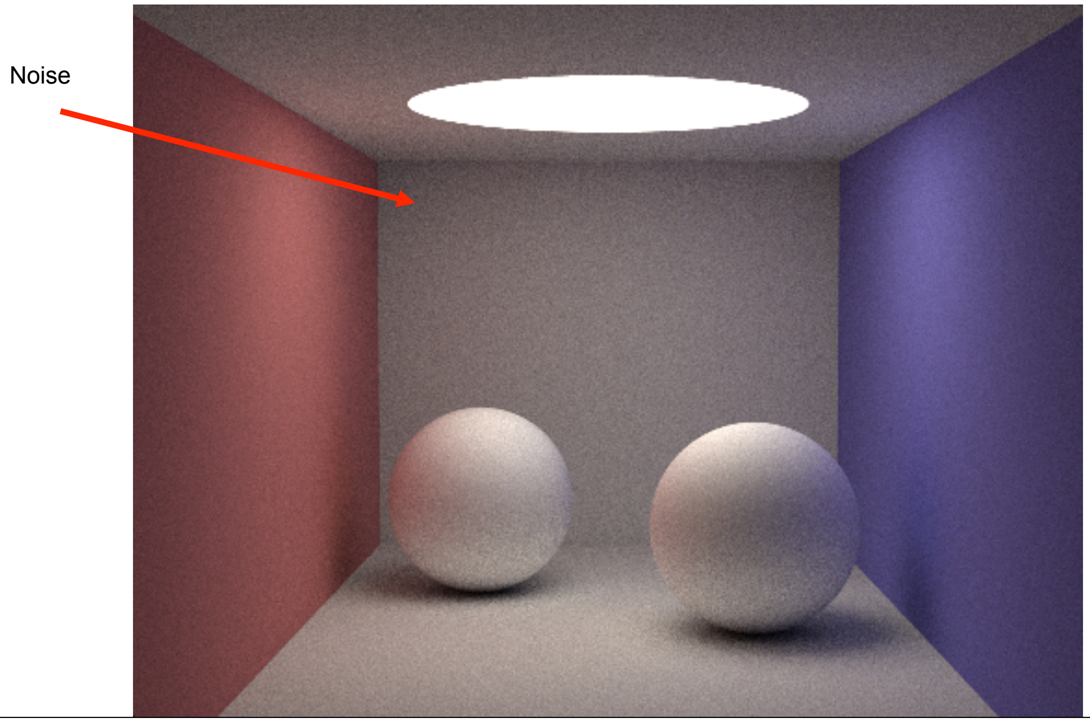
WHAT YOU GET



Area light

Reflections from diffuse objects 'color bleeding'

Soft shadows



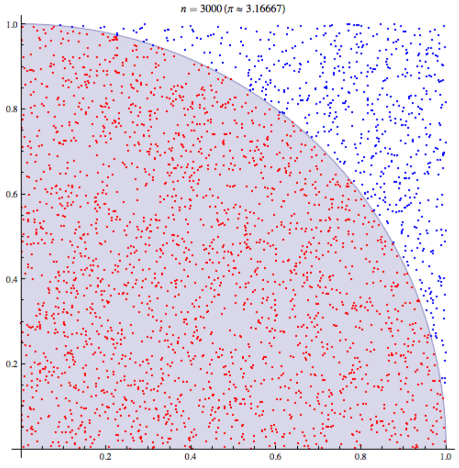
Noise

MONTE CARLO METHODS

- General idea: compute something using random sampling
- Used for computing integrals of complex functions
- E.g. areas or volumes
  - If it’s hard to compute analytically
  - But easy to test if a point is inside
- If we throw enough random samples, by the law of large numbers, mean ~ empirical mean

MONTE CARLO METHODS

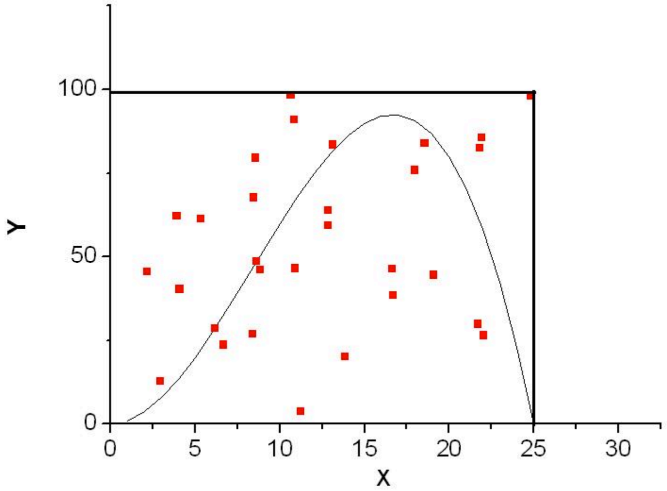
- Example: approximating π:



"Pi 30K" by CaitlinJo - Own work. This mathematical image was created with Mathematica. Licensed under CC BY 3.0 via Commons - https://commons.wikimedia.org/wiki/File:Pi_30K.gif#/media/File:Pi_30K.gif

MONTE CARLO METHODS

- Example: computing a weird integral



"Pi 30K" by CaitlinJo - Owr Ↄommons - https:// commons.wikimedia.org/wiki/File:Pi_30K.gif#/media/File:Pi_30K.gif

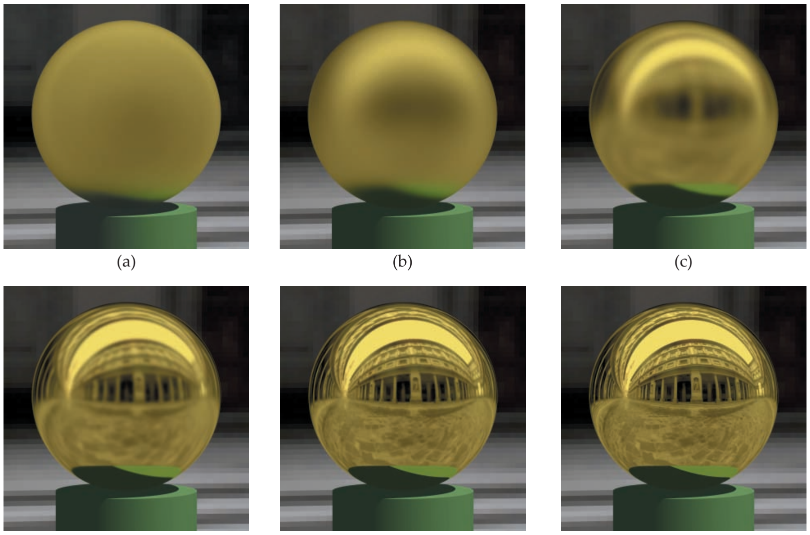
MONTE-CARLO: RAY TRACING

- Now in RAY tracing, we can generate rays randomly from an area light source
- Instead of shooting a single shadow ray,
  - Generate many randomly towards a light source
    - Generate a point on the light source
    - Shoot a ray towards that point
  - Average their contribution
- Soft shadows in RAY tracing!

SIMPLEST PATH TRACER

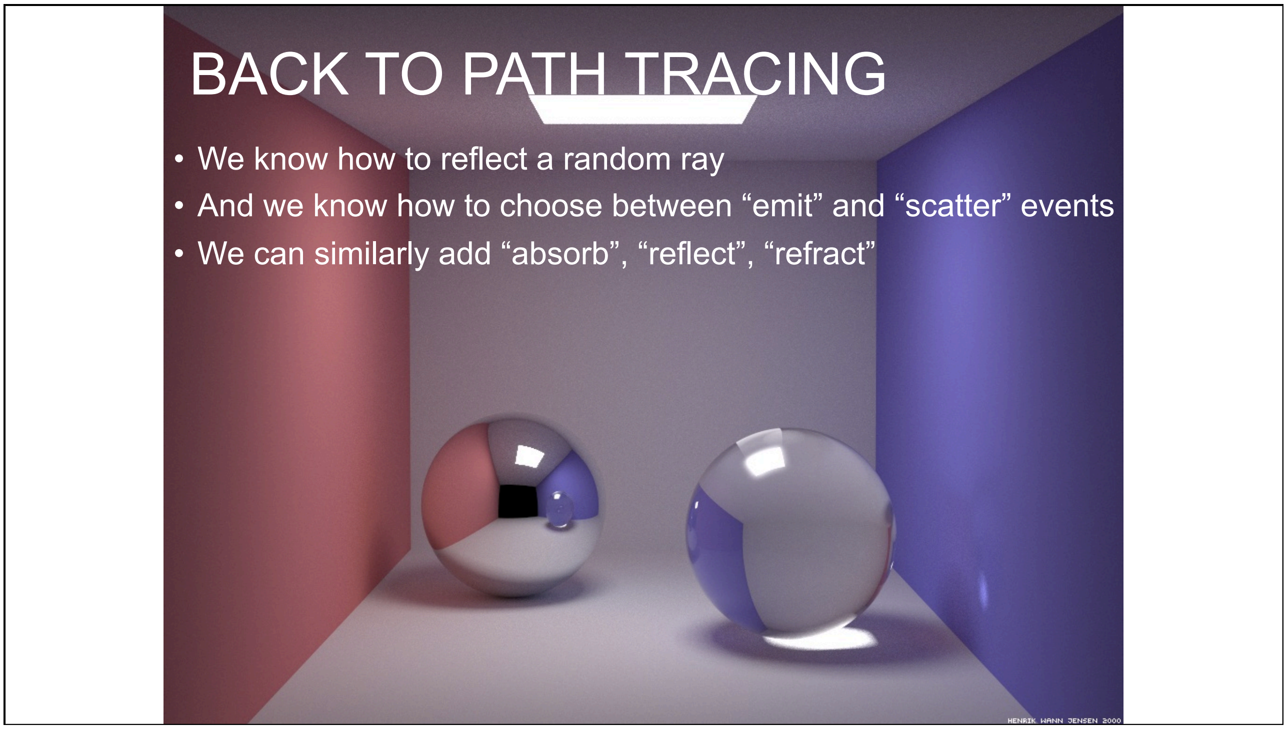
```
PathTrace(Ray r) {
  P = closestIntersection(r);
  if (random(emit, reflect) == emit)
    return EmissionColor;
  else {
    Ray v = {intersectionPt,
        randomDirectionInHemisphere(r.normalWhereObjWasHit)};
    double cos_theta = dot(v.direction, r.normalWhereObjWasHit);
    return PathTrace(v)*cos_theta*reflectance;
  }
}
```

HOW TO GENERATE REFLECTION FOR A GLOSSY SURFACE



(a) (b) (c)

(d) (e) (f)



BACK TO PATH TRACING

- We know how to reflect a random ray
- And we know how to choose between “emit” and “scatter” events
- We can similarly add “absorb”, “reflect”, “refract”

RAY TRACING VS PATH TRACING

- Global illumination algorithms
- Rays emitted FROM camera

- Ray Tracing
  - Single ray per pixel
  - Supports indirect lighting only from specular surfaces
    - No color bleeding
  - Shoots shadow rays to compute direct illumination
    - Soft shadows are harder to get

- Path Tracing (*may produce renders indistinguishable from photos)*
  - Many rays per pixel, their color averaged
  - At each interaction, ray direction changes randomly with some distribution
  - No difference between light sources and objects
    - Soft shadows, complex materials, etc.
    - Supports all sorts of indirect lighting