Raytracing
IV – Distribution Raytracing

Stefan BORNHOFEN

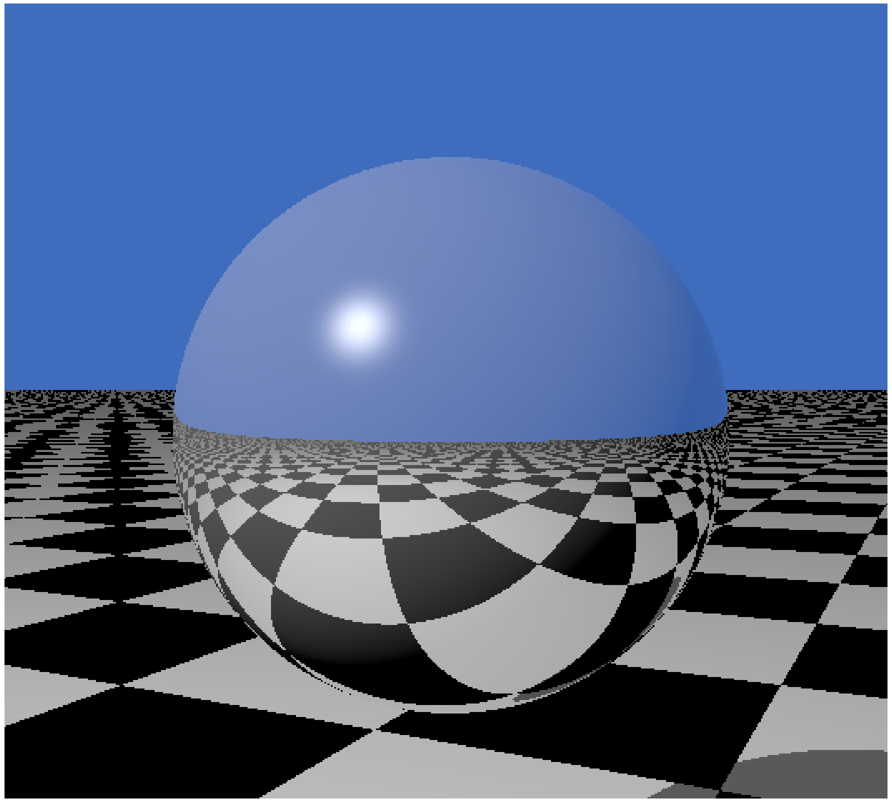
Why does raytracing look so obviously computer generated?

The images are too crisp:

- Edges are jagged
- Surfaces are shiny
- Glass is crystal clear
- Everything is in focus
- Everything is completely still
- Shadows have marked silhouettes

“Distributed Ray Tracing” was introduced by Robert L. Cook in 1984

- Allows for the rendering of "soft" phenomena
- Later renamed to **distribution raytracing** so as not to confuse with parallel computing.



Distribution Raytracing

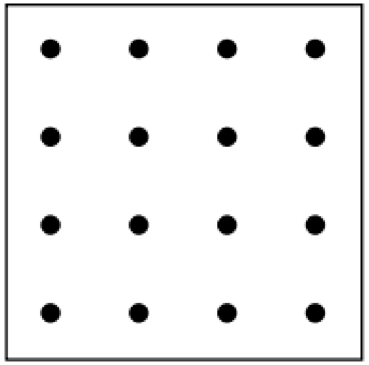
Replace a single ray with a distribution of multiple rays and average the results together.

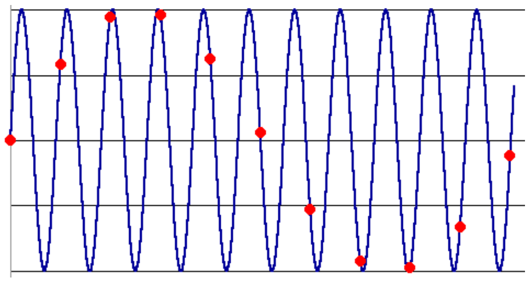
- **Multiple primary rays:**
  - Antialiasing: distribute rays over pixel area
  - Motion blur: distribute rays over time interval
  - Depth of field: distribute rays over lens aperture
- **Multiple secondary rays:**
  - Soft shadows: distribute shadow rays over light area
  - Glossy surfaces: distribute rays over reflection cone
  - Blurred transparency: distribute rays over refraction cone

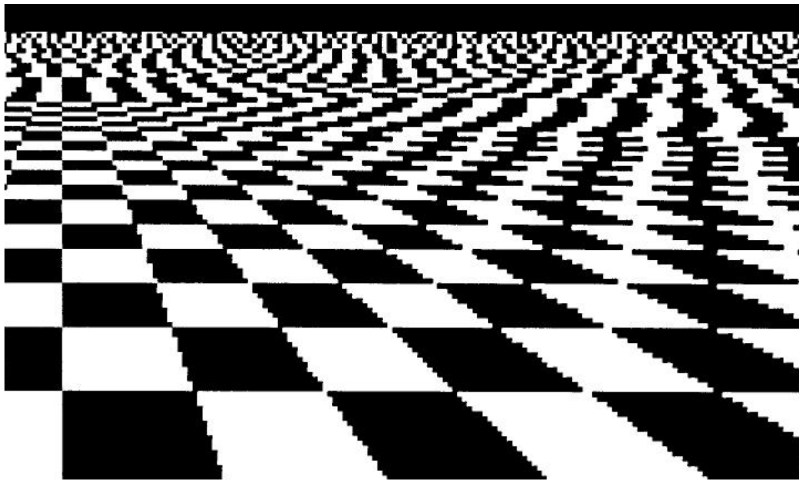
The distributed rays sample values over the integration domain. Different sampling techniques are used to attempt to approximate the integral as precisely and as fast as possible. Good sample generation is art in itself.

Regular Sampling

- Subdivide the domain into a regular grid with a fixed number of samples
- A major problem with regular samples is that aliasing artifacts such as **Moiré patterns** can arise on surfaces with periodic motifs.
- The **Nyquist frequency** sets the limit that determines how fine structures can be captured by a given sampling process.

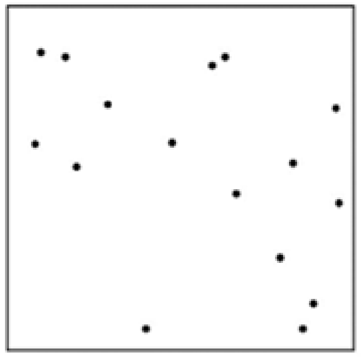






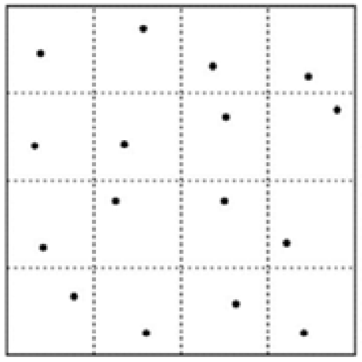
Random Sampling

- Numerical integration methods based on random samples are called **Monte-Carlo Integration**
- Moiré artifacts are turned into noise, which is often considered as the lesser of two evils.
- However, basic random sampling has a chance of producing bad distributions such as many sample points clumped together
- “White noise”


Random

Stratified Sampling (Jittering)

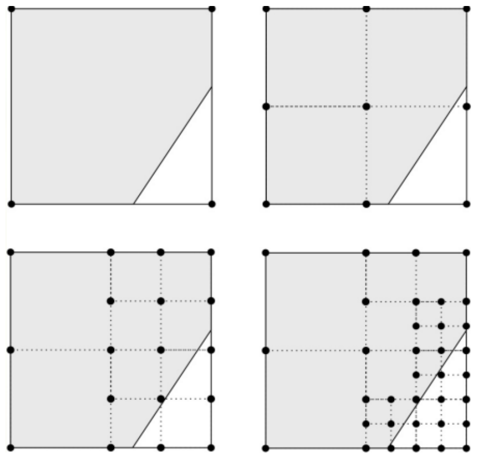
- A widely used sampling strategy reducing noise variance
- Sample points are stochastic yet evenly dispersed in the spatial domain
- Subdivide your domain into equally-sized patches called “strata”, and take one random sample per stratum
- Samples can only clump near the boundaries.
- “Blue Noise”


Stratified

More Sampling Strategies

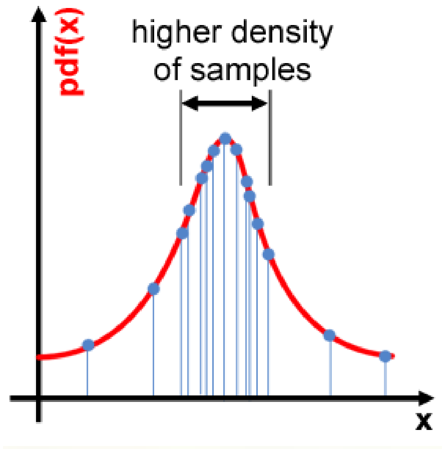
Adaptive Sampling

- Noise can be eliminated by increasing the number of samples, but casting many rays can be unnecessarily costly.
- Adaptive sampling tries to avoid the problem of using a fixed high number of samples, by concentrating the samples in the more difficult parts of the image.



Importance Sampling

- Use heuristics to guess where the sampled function will be most significant
- Choose a probability distribution function (pdf) that produces more samples in important regions.

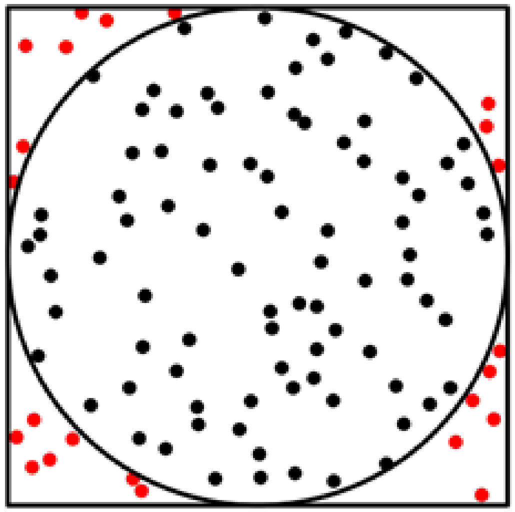


2D Disc Sampling

Rejection method

A simple approach consists in sampling the unit square and rejecting any sample falling outside the disk. It is quite efficient for 2 dimensions as the acceptance rate is high and the calculations are fast. However, the rejection method carries a bad reputation as it get more and more inefficient for higher dimensions.

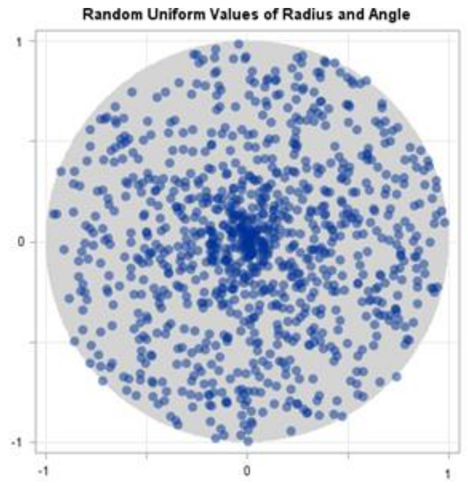
```
1) y = random(-1,1), y = random(-1,1)
2) if ($x^2+y^2>1$) then go to step 1
3) return (x,y)
```



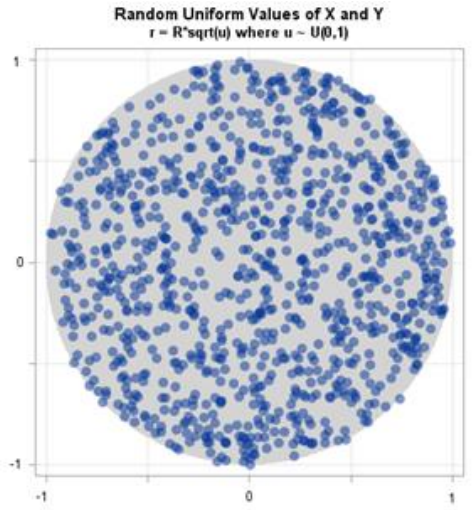
2D Disc Sampling

Polar sampling

- Draw a radius within [0,1[and an angle within [0, 2π[.
- If both the radius and angle are uniformly sampled, this leads to a density singularity at the origin.
- We want the density to be uniformly proportional to area rather than distance to the origin. Since area grows according to r^2, generate uniform random values and take their square roots.



```
theta = 2*PI*rand(0,1)
r = rand(0,1) // THIS IS WRONG
x = r*cos(theta)
y = r*sin(theta)
return (x,y)
```



```
theta = 2*PI*rand(0,1);
r = SQRT(rand(0,1));
x = r*cos(theta)
y = r*sin(theta)
return (x,y)
```

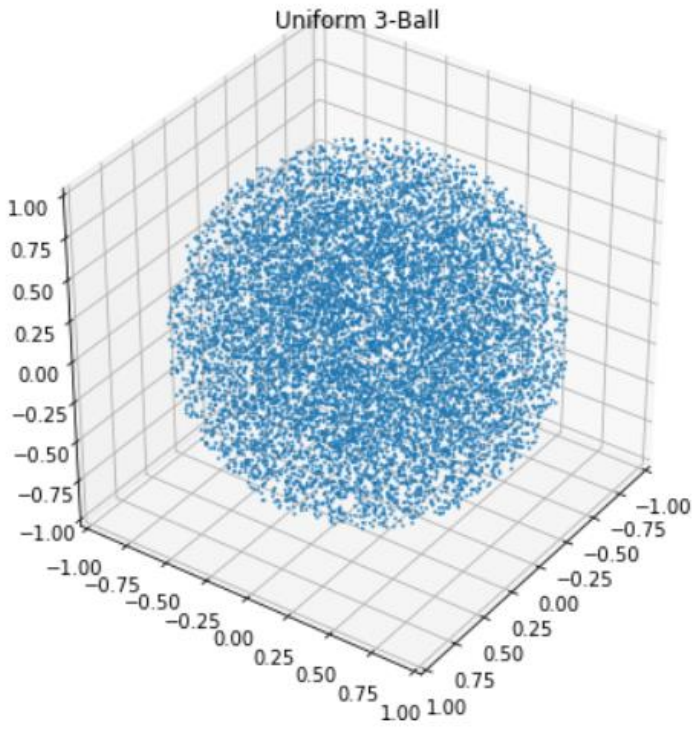
3D Sphere Sampling

Rejection method

```
1) x = random(-1,1), y = random(-1,1), z = random(-1,1)
2) if ($x^2$+$y^2$+$z^2$>1) then go to step 1
3) return (x,y,z)
```

Polar sampling

```
u = random(-1,1)
phi = 2 * PI * random(0,1)
r = random(0,1)$^{1/3}$
x = r*cos(phi) * sqrt(1-$u^2$)
y = r*sin(phi) * sqrt(1-$u^2$)
z = r*u
```



More about disc and sphere sampling methods:
https://extremelearning.com.au/how-to-generate-uniformly-random-points-on-n-spheres-and-n-balls

Random in GLSL

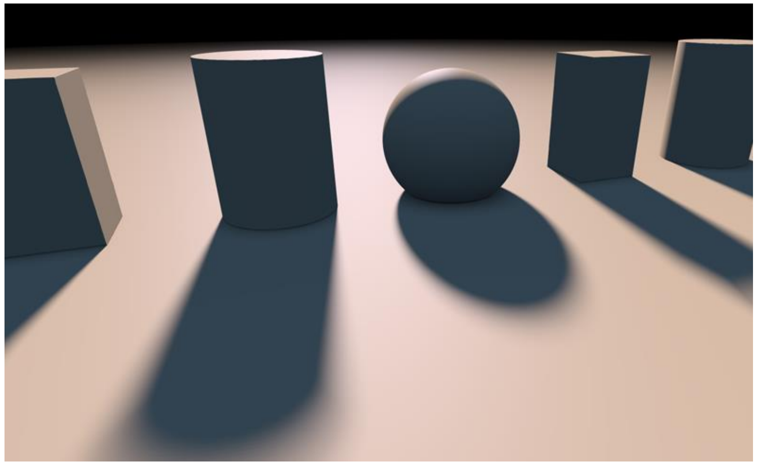
GLSL has no built-in random number generator, so we need to write our own one.
The following algorithm is good enough for us.

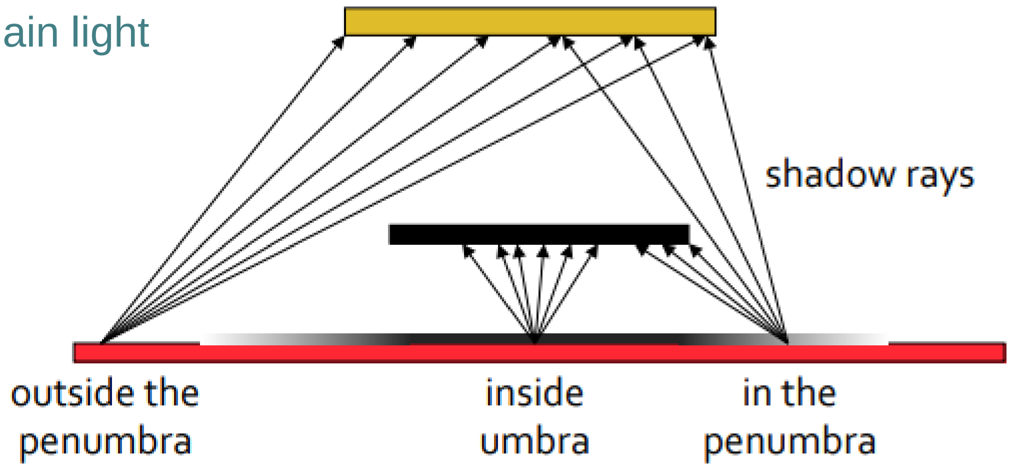
(source: https://www.shadertoy.com/view/tsf3Dn)

```
int seed;

int xorshift(int value) {
  value ^= value << 13;
  value ^= value >> 17;
  value ^= value << 5;
  return value;
}

float random() { // generates a pseudo random number in [0,1[
  seed = xorshift(seed);
  return abs(fract(float(seed) / 3141.592653));
}

void main()
{
   // initial random seed, unique per pixel
   seed = int(gl_FragCoord.x) + SCREEN_X*int(gl_FragCoord.y);
   ...
}
```

Soft Shadows

- Point lights have sharp shadows while shadows of area lights are soft.
- Cast multiple shadow rays, each ray to a different point on the surface of the light source.
- Count the unoccluded shadow rays to find the strength of the shadow.
  - No shadow ray gets through: umbra
  - Some shadow rays get through: penumbra
  - all shadow rays get through: plain light

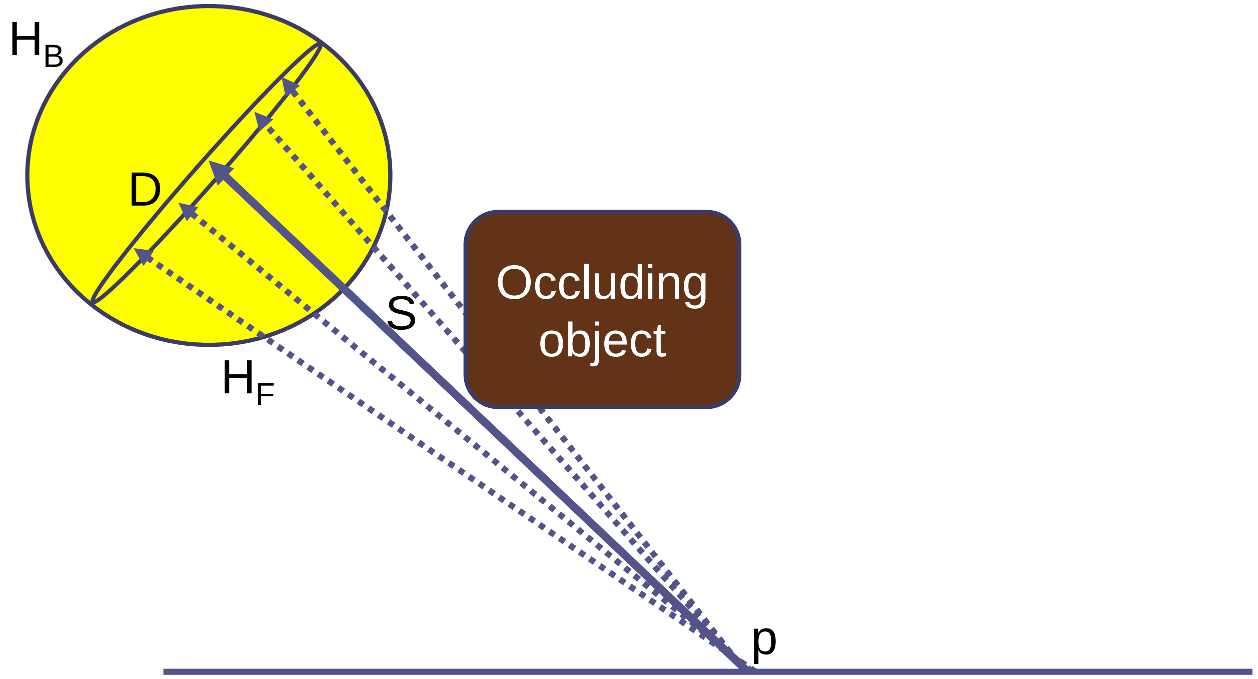




Soft Shadows

To compute soft shadows for a spherical light source, we sample shadow rays toward the disc D that is orthogonal to the principal shadow ray S targeting the sphere center.

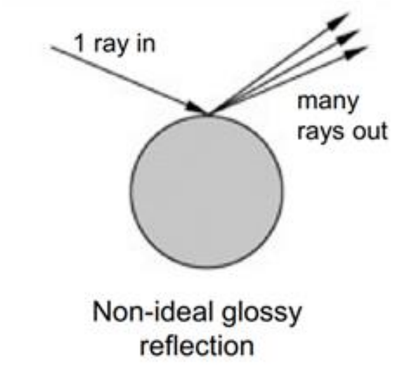
Uniformly distributed sampling points on D lead to an uneven density of the corresponding sampling points on H_F and model the uneven light contribution of the hemisphere.

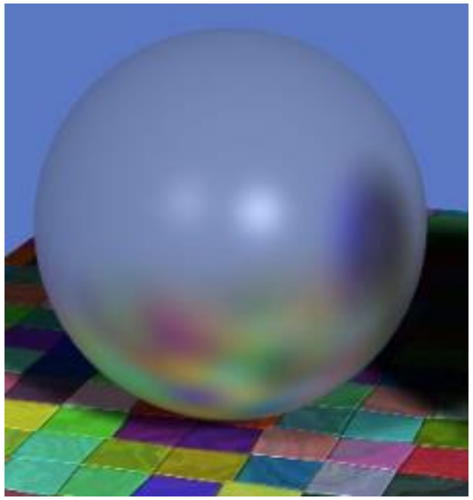


Glossy Reflections

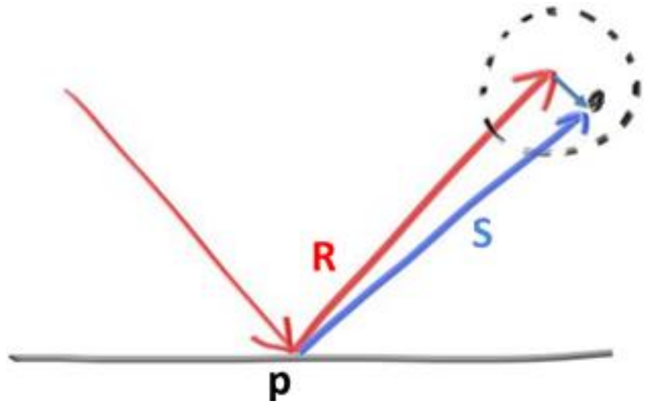
Most surfaces are imperfect reflectors.

- Surface microfacets perturb the direction of reflected rays
- Reflect rays in a cone around the perfect specular direction
- Phong specular lighting tries to fake this for the light source



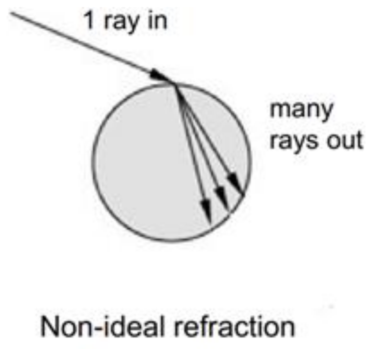


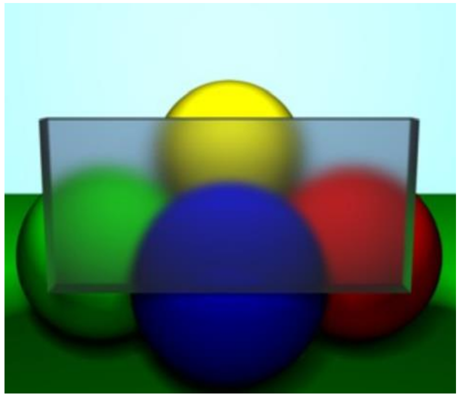
Randomize the reflected direction by using a small sphere and choosing a new endpoint for the ray. The bigger the sphere, the fuzzier the reflections will be.



```
S = normalize (R + glossiness
* samplePointInUnitSphere())
```

Blurred transparency
Instead of distributing rays around the reflected ray, distribute them around the refracted ray:



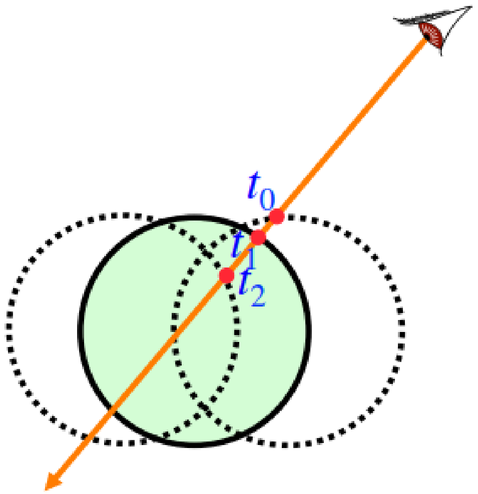


Motion Blur

In a real camera, the shutter remains open for a short time interval, during which the camera and the objects may move.

To reproduce such a camera shot, we seek an average of what the camera senses while its shutter is open to the world.

- Add a velocity vector to the objects in the scene.
- Assuming a start at time = 0 and ending at time = 1, modify the raytracer such that you can launch primary rays at random times in [0,1].
- Average out the results.



Virtual and Real Cameras

Image Plane vs. View Plane

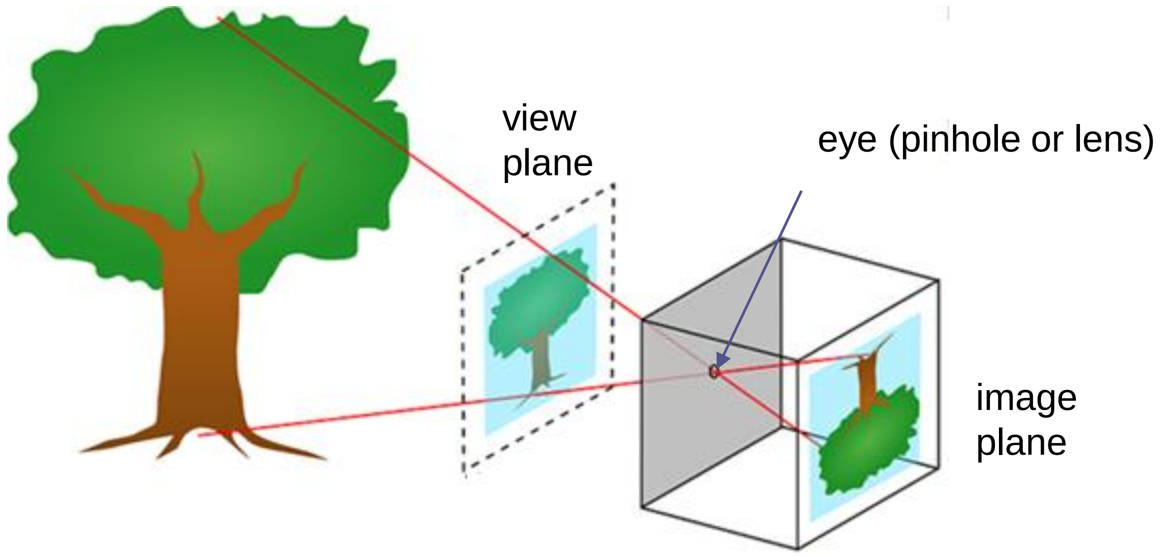
- In a real camera the image plane is present inside the camera behind the eye. To get the actual view, the image plane must be inverted.
- For a virtual camera we can define a view plane between the eye and the objects.

Pinhole vs. Lens

A pinhole camera is a simple camera without a lens. The smaller the hole, the bigger the sharpness.

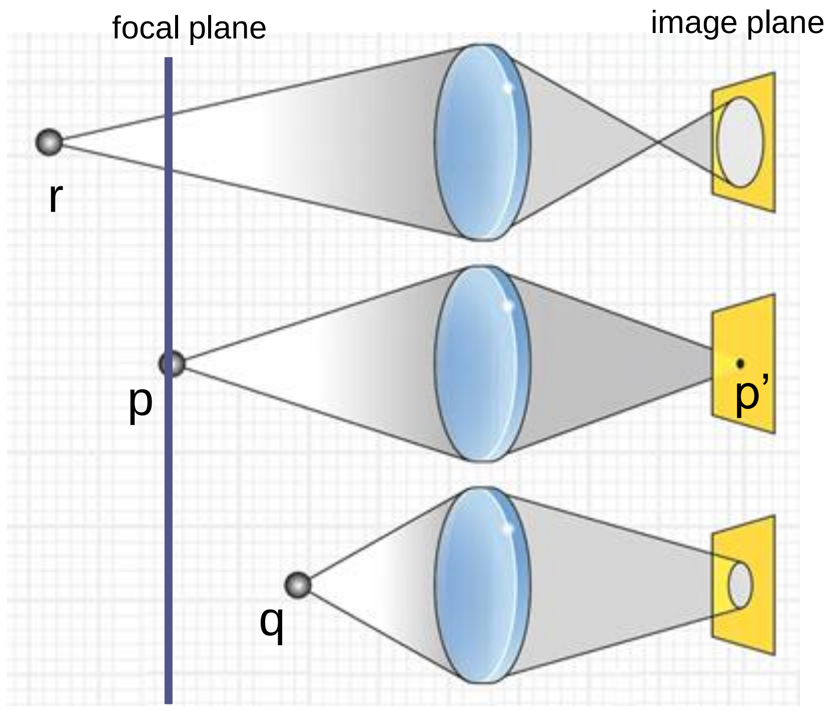
- For a virtual camera we can define an ideal pinhole camera with an infinitely small hole. Everything appears in focus.
- In the real world too little light enters a pinhole. A lens admits more light but needs to be focused.

To produce more realistic images with out-of-focus effects need to model a lens.



Thin Lens Simulation

- We consider an ideal **thin lens**, i.e. a lens whose thickness has no optical effects
- The **focal plane** is defined as the plane perpendicular to the optical axis which contains the focal points.
- Every point p on the focal plane has a corresponding image point p' on the image plane.
- Rays starting at points q or r outside the focal plane go through the image plane at different locations, with the result that they will be recorded as a circle (**circle of confusion**) and therefore appear out of focus.
- **Depth of field** is the range of distances parallel to the lens axis in which the scene is in focus

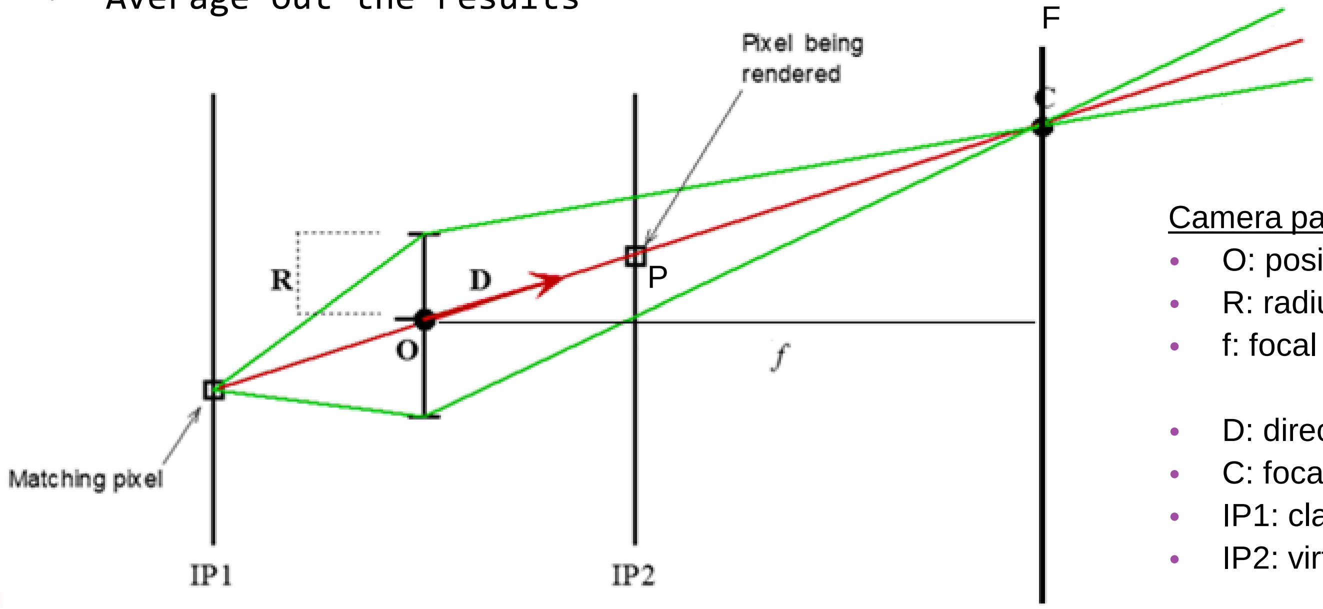


In raytracing, the image appears in perfect focus over the range of distances where the circle of confusion is smaller than a pixel.

Depth of Field

How to sample the pixel color of pixel P with DOF?

```
For a given pixel P
•  Compute the ray O + t*D going through P
•  Find the corresponding convergence point C on the focal plane F
•  Select random points on the lens disk: Q_i = O + R*randomPointOnUnitDisk_i
•  Sample rays Q_i + t*normalize(C-Q_i)
•  Average out the results
```



Camera parameters

- O: position, center of the lens.
- R: radius of the lens disk, or aperture
- f: focal distance

- D: direction of the primary ray
- C: focal point
- IP1: classical image plane behind the lens.
- IP2: virtual view plane in front of the lens.

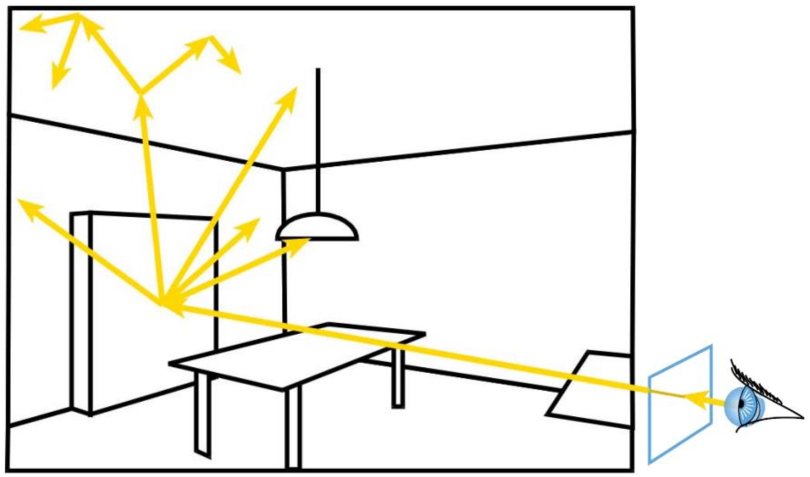
The smaller the aperture, the larger the depth of field becomes. In the limit case with an aperture size equal to zero, we are back to a pinhole camera where everything is in focus.

From Raytracing to Path Tracing

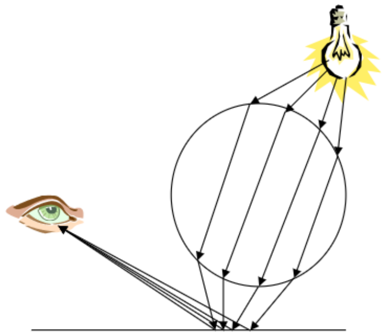
Distribution Raytracing

- Cast a ray from the eye through each pixel
- Cast random rays from the hit point to evaluate hemispherical integral using random sampling
- Recurse
- Systematically sample light sources at each hit (don't just wait the rays will hit it by chance)

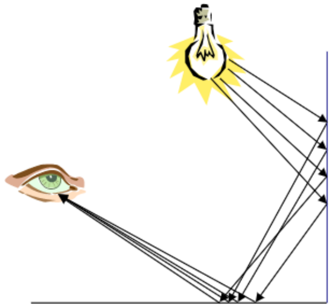
Distribution raytracing is not exactly photorealistic because the rendering equation is not fully evaluated. In particular, indirect diffuse reflections are ignored otherwise the number of rays explodes.



Light effects not seen with distribution raytracing:



Caustics
Light focuses through a specular surface or due to refraction onto a diffuse surface.



Color Bleeding
The color of a diffuse surface is reflected in another diffuse surface.

From Raytracing to Path Tracing

Path Tracing

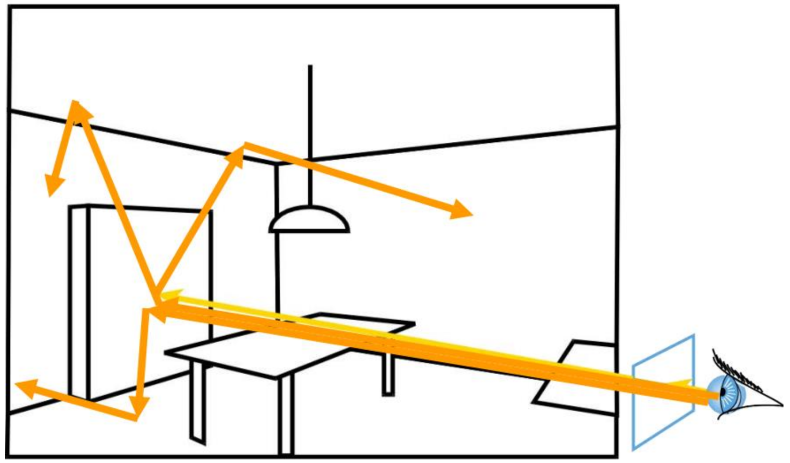
Path tracing was developed as a solution to the complete rendering equation. It combines all integration domains into a single, high-dimensional domain and samples it in a unified way.

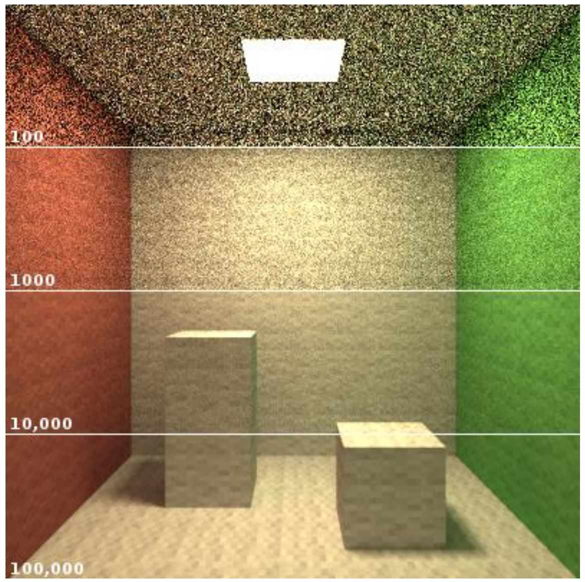
We randomly sample the space of all possible light paths between the source and the camera:

- Send many primary rays per pixel
- Trace only one secondary ray per recursion
- Some of these paths reach light and contribute to the finished scene, while others do not.

Global illumination is taken into account!

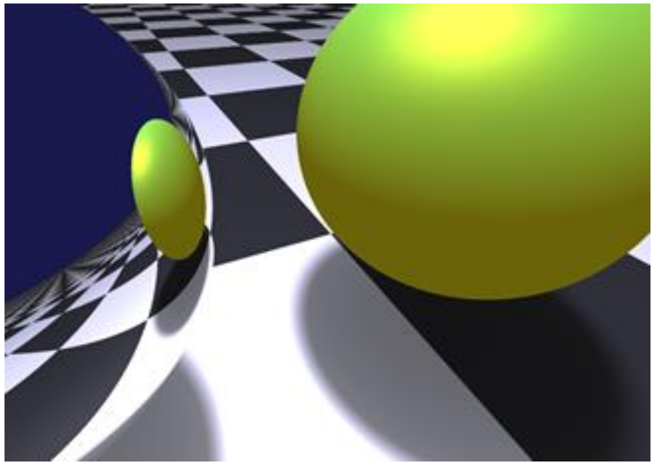
Path Tracing is very costly because thousands of rays have to be sampled to eliminate the noise. Many optimization techniques have been designed to accelerate the convergence of the sampling process (Bidirectional Path Tracing, Metropolis Light Transport, Photon Mapping, …)

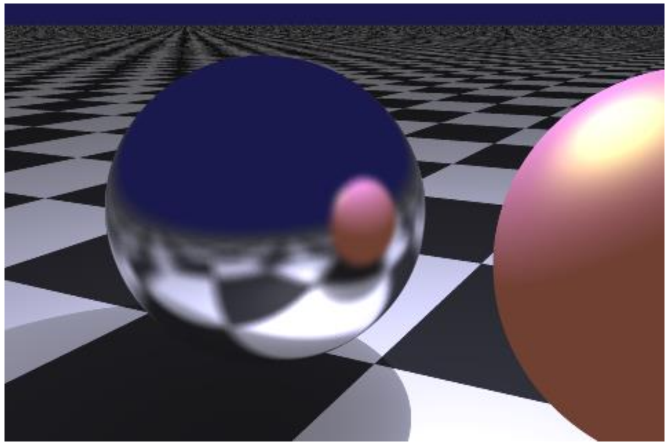


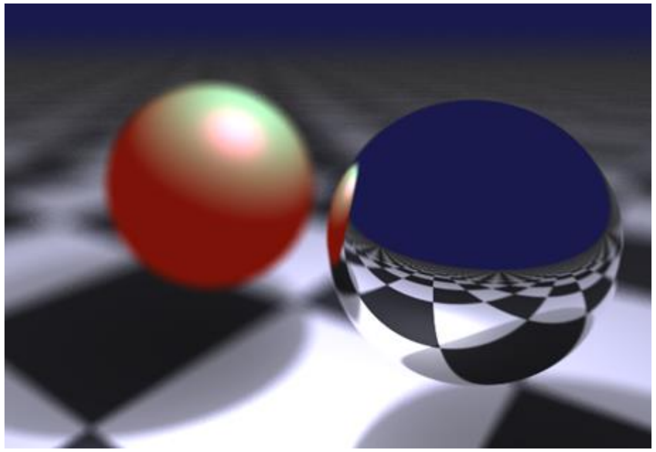


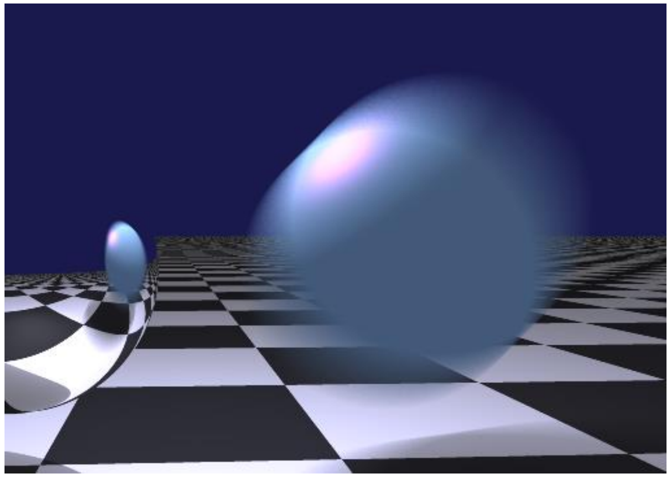
Your work

Implement the following distribution raytracing algorithms using random sampling.


Soft Shadows (spherical light source)


Glossy surface


Depth of Field


Motion Blur