COMPUTER GRAPHICS

INTRO TO RAY TRACING

TOPICS

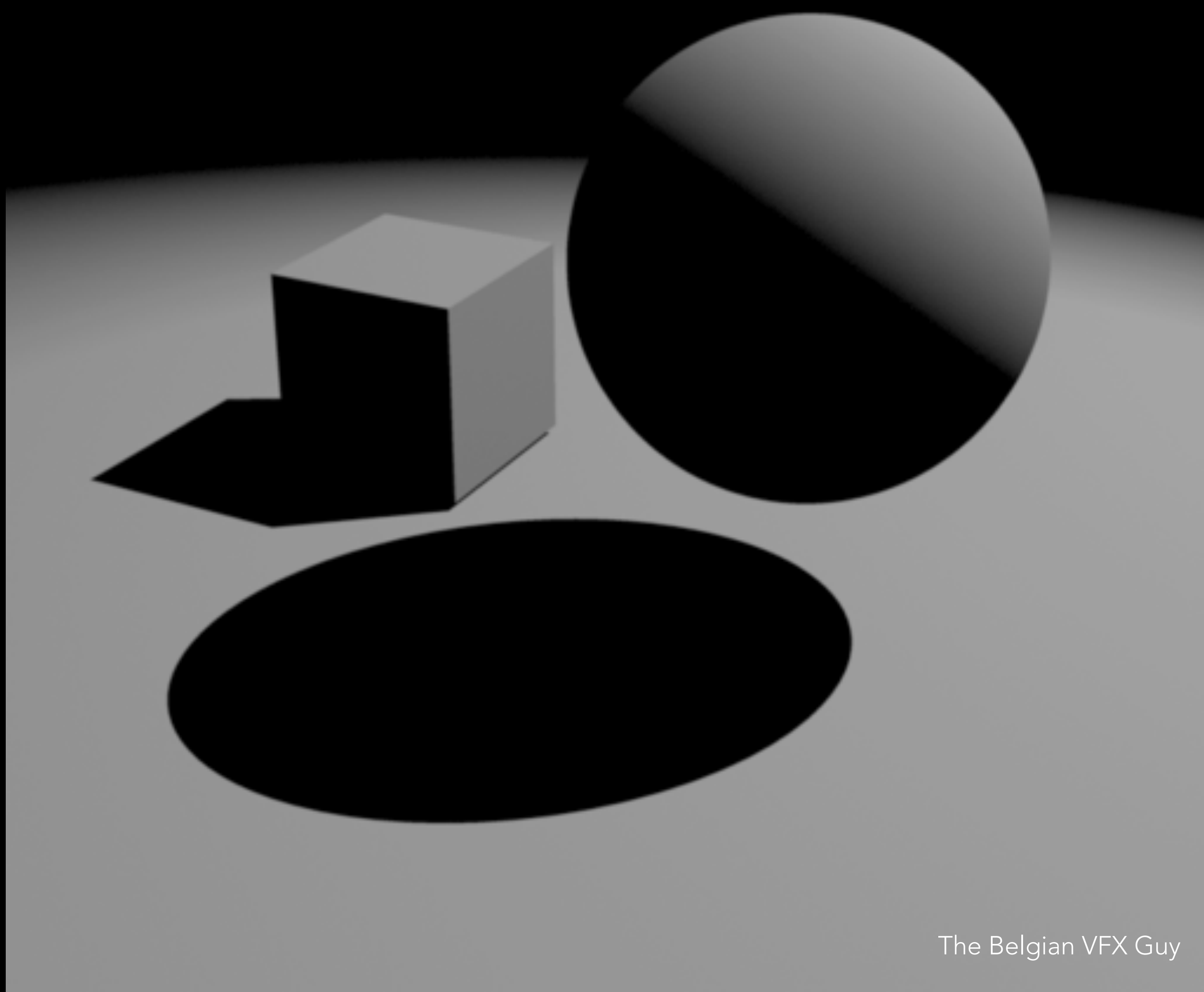
- What is ray tracing?
- The life of a ray
- The rendering equation
- Let's write a path tracer!
- Moving forward

WHAT IS RAY TRACING?

- A technique of generating an image by tracing paths of light
- Capable of simulating variety of optical phenomenon
  - Reflection, refraction, caustics, scattering
- High computational cost
- Comes in variety of forms

LIFE OF A RAY

1. Spawn from camera
2. Intersect with primitive
3. Get surface colour
4. Dot product with light
5. Shadow path
6. Repeat…



1. SPAWN RAY

ORTHONORMAL BASIS

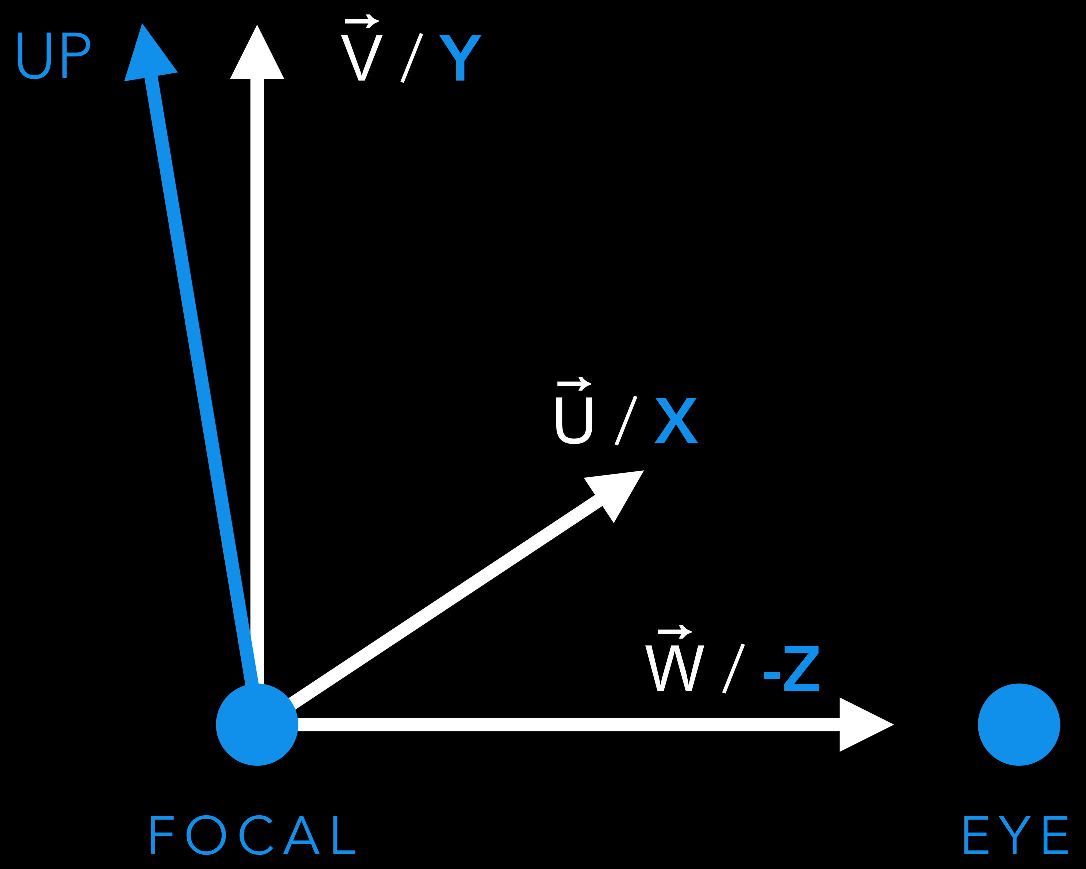
$$\hat{w} = \frac{\text{eye} - \text{focal}}{|\text{eye} - \text{focal}|}$$

$$\hat{u} = \frac{\text{up} \times \hat{w}}{|\text{up} \times \hat{w}|}$$

$$\hat{v} = \hat{w} \times \hat{u}$$

LINEAR TRANSFORMATION

$$\vec{d} = \hat{u}x + \hat{v}y - \hat{w}\text{z}$$



**Right handed coordinate system*

2. RAY-SPHERE INTERSECTION

RAY:

$$\vec{p} = \vec{o} + t\vec{d}$$

SPHERE:

$$r^2 = x^2 + y^2 + z^2$$

$$r^2 = (x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2$$

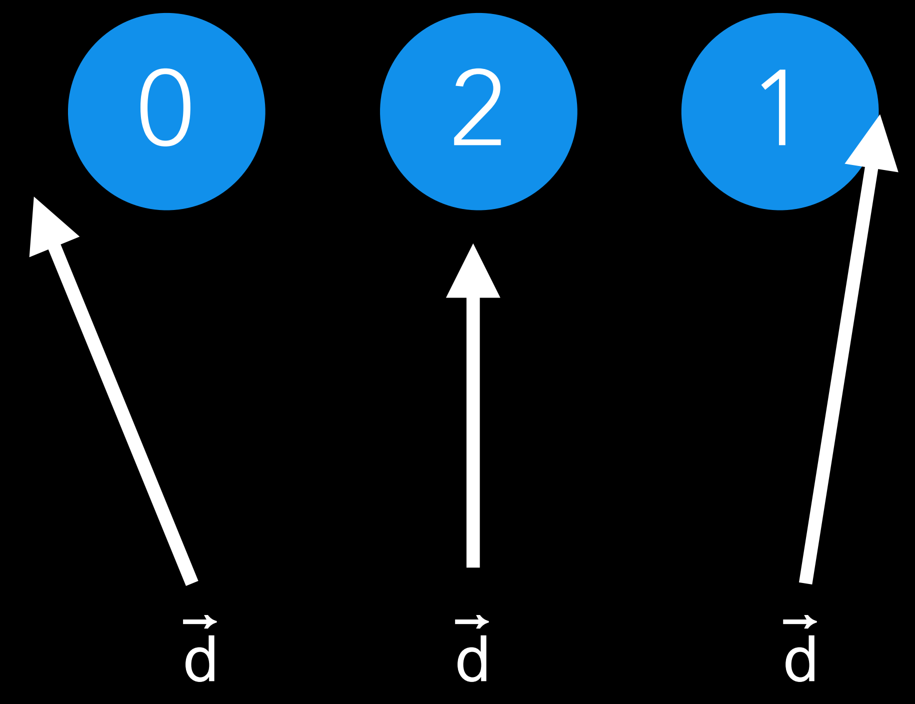
$$0 = (\vec{p} - \vec{c})^2 - r^2$$

SOLVE FOR t:

$$0 = (\vec{o} + t\vec{d} - \vec{c})^2 - r^2$$

NORMAL:

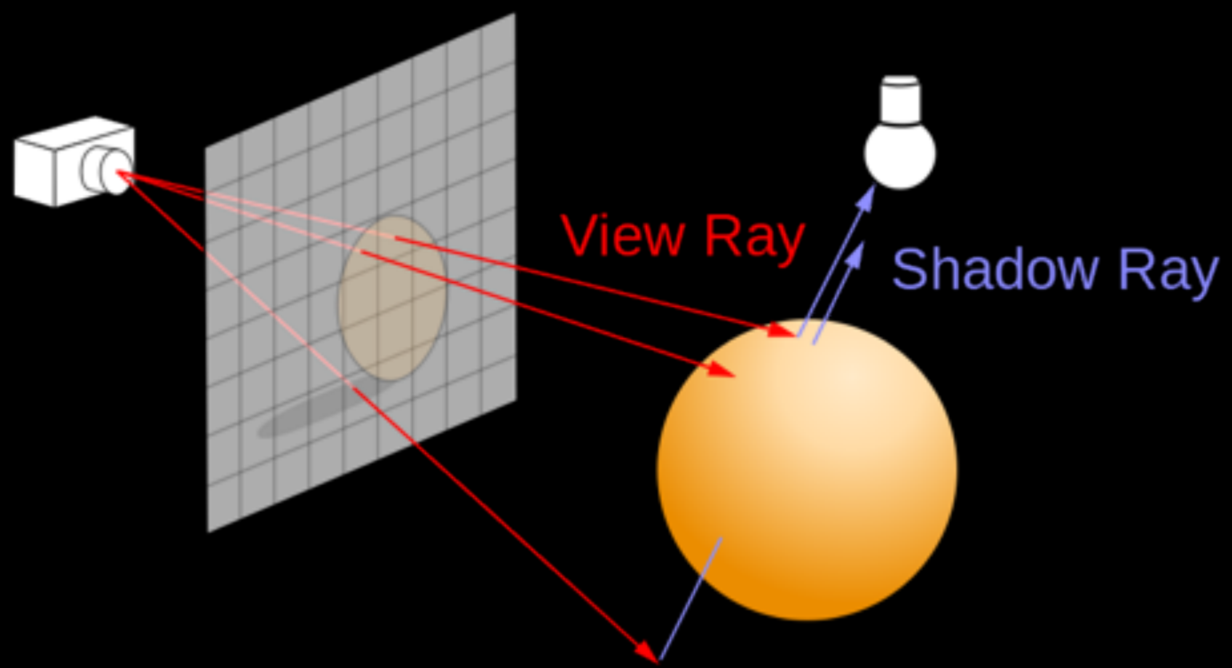
$$\hat{n} = \frac{(\vec{p} - \vec{c})/r}{|(\vec{p} - \vec{c})/r|}$$



3,4. COLOUR AND SHADING

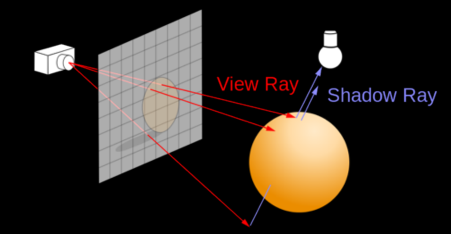
LAMBERTIAN SHADING:

$$I_D = L \cdot NCI_L$$

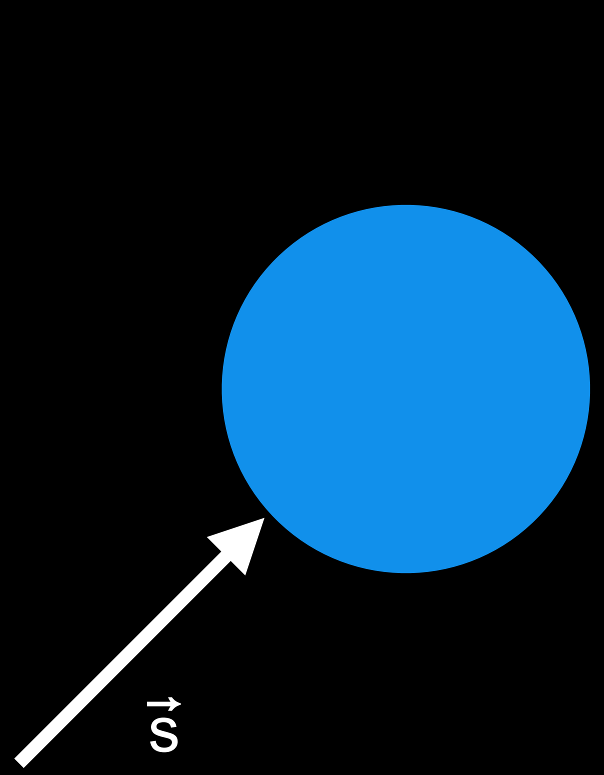


I_D = Intensity of diffuse light
L = Light-direction vector
N = Surface normal
C = Surface colour
I_L = Light intensity

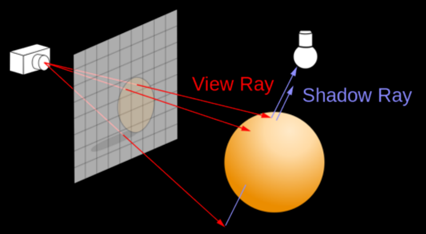
5. SHADOW PATH



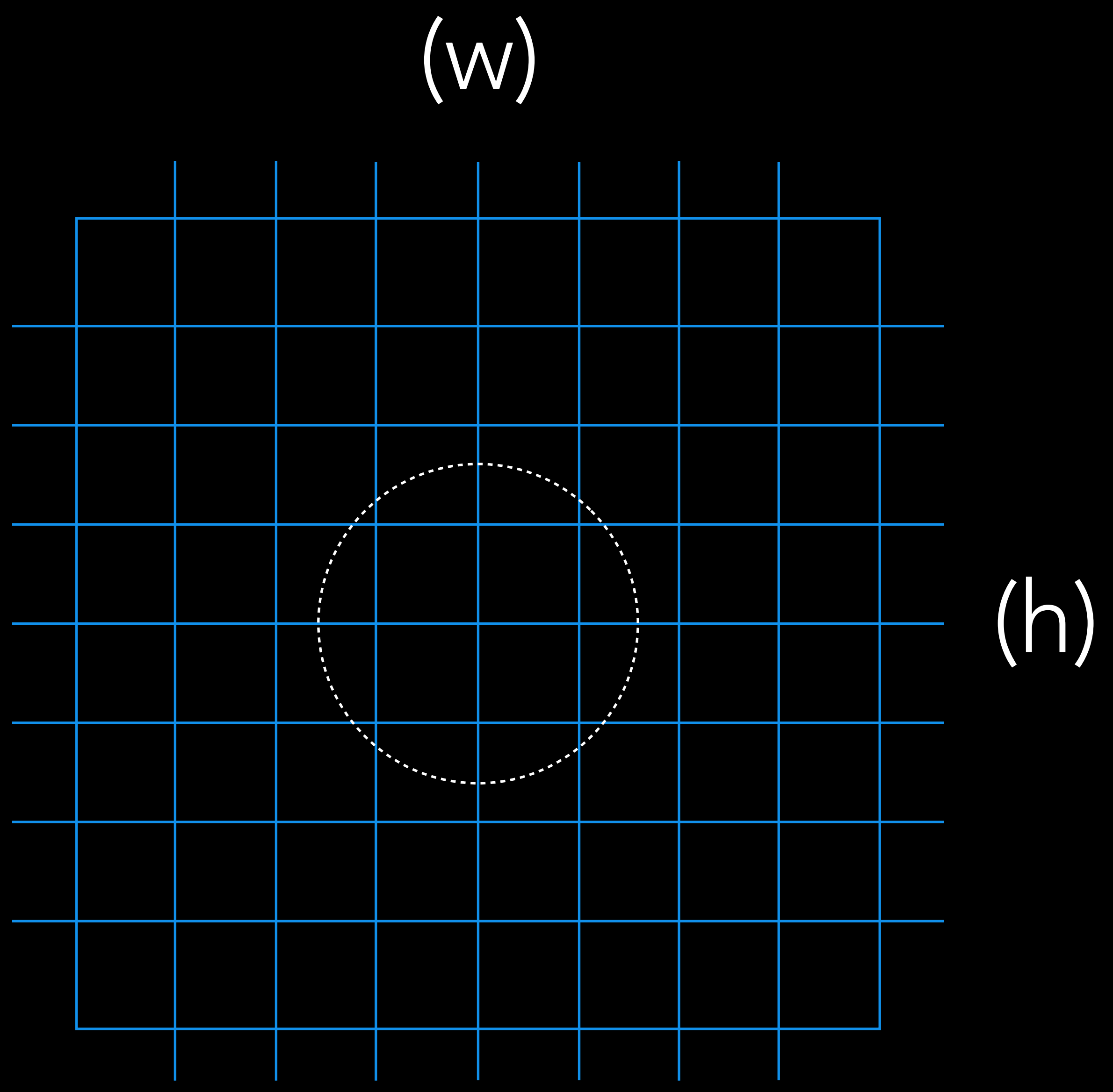
- Trace path from hit point to light
- Account for self-intersection
- Add ambient term



6. REPEAT PER PIXEL

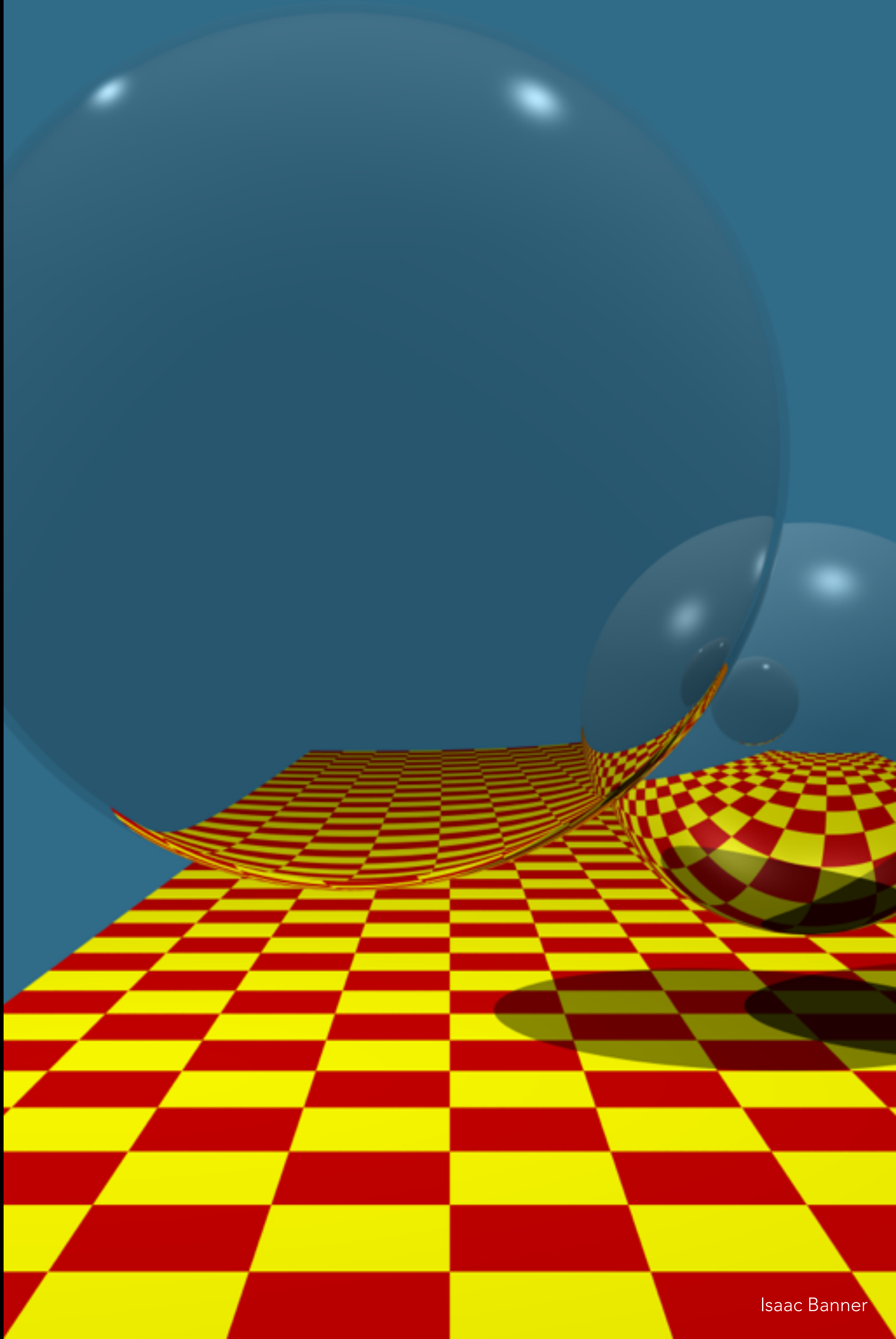



```
pixels = [height, width]
for y in 0..(height - 1)
  for x in 0..(width - 1)
    ray = camera.spawn_ray(x, y)
    pixel[y, x] = trace(ray)
  end
end
save_image(pixels)
```



IMPROVEMENTS

- Anti-aliasing
- Shadows
- Ambient occlusion, ambient term
- Whitted ray tracing
  - Reflection, refraction, shadow
- Distributed (stochastic) ray tracing
  - Soft phenomena
  - Glossy surfaces
- Specular Reflection
- Refraction



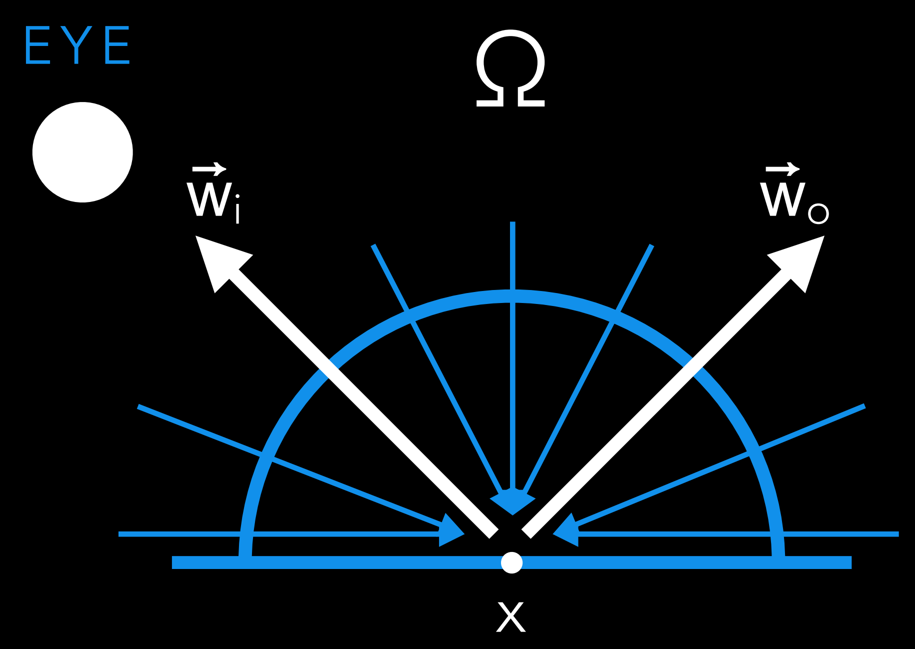
REALISTIC RENDERING

THE RENDERING EQUATION

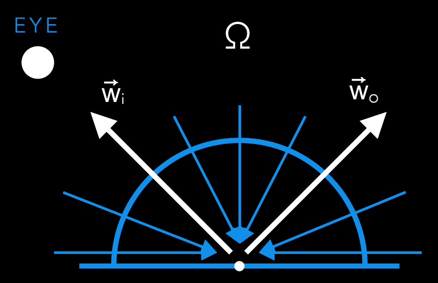
Vlad "SATtva" Miller

CALCULATING REFLECTED LIGHT

- Emitted light
- BRDF of surface
  - Reciprocity principal
  - Conservation of energy
- Incoming light
- Geometric relationship
- Occluded?



CALCULATING REFLECTED LIGHT



$$L(x, \vec{w_o}) = L_e(x, \vec{w_o}) + \int_\Omega f_r(x, \vec{w_i} \rightarrow \vec{w_o}) L(x', \vec{w_i}) G(x, x') V(x, x') dw_i$$

$L(x, \vec{w_o})$ = intensity reflected from position x in direction $\vec{w_o}$

$f_r(x, \vec{w_i} \rightarrow \vec{w_o})$ = BRDF of surface at point x

$L_e(x, \vec{w_o})$ = light emitted from x by object

$L(x', \vec{w_i})$ = light from x' by another object along $\vec{w_i}$

$G(x, x')$ = geometric relationship between x and x'

$V(x, x')$ = visibility test, 1 if x can see x', 0 otherwise

$\int_\Omega \ldots \, dw_i$ = integral over the unit hemisphere

PRACTICAL EXAMPLE

BUILDING A PATH TRACER

freejack

FEATURES

- Ruby (feature?) ~ 200 lines
- Monte Carlo method - repeated random sampling
- Specular and diffuse BRDFs
- Global illumination
- Anti-aliasing
- Ray-sphere intersection
- Soft-shadows
- Modified Cornell box

IMPROVEMENTS

- Meshes
- Acceleration structures
- Explicit light sampling
- Bidirectional path tracing
- Multiple importance sampling
  - Sampling BRDF vs luminaries
- Making it faster
  - GPGPU
  - SIMD - data level parallelism