INFOGR – Computer Graphics

J. Bikker - April-July 2015 - Lecture 8: “Ray Tracing”

Welcome!

Today’s Agenda:

- Introduction to Ray Tracing
- Whitted-style Ray Tracing
- Intersections
- Shadows, Reflections & Refractions

Practicum

Exam

- Answers will be available today
- Can be discussed in tutorial (Thursday)
- Still questions?
- Make an appointment.

P2

- Will be graded as soon as possible
- Partner dropped? Let me know.

Final challenge is coming up…

Practicum

Deadline: June 30

P3 available today!

Easy Assignments (choose four):

E.1 Multiple Light Sources
E.2 Spotlight
E.3 Cell Shading
E.4 Frustum Culling
E.5 Color Filter
E.6 Gaussian Blur

Medium Assignments (choose up to two, or up to one hard one):

M.1 Textured Light
M.2 Shadow Mapping
M.3 Reflection
M.4 Cube Mapping
M.5 Transparency
M.6 Bloom
M.7 God Rays (Volumetric Lighting)

Hard Assignments (choose up to one, or two medium ones):

H.1 HDR Lighting and Tone Mapping
H.2 Deferred Shading
H.3 Parallax Mapping
H.4 Screen-space Ambient Occlusion
H.5 Image-based lighting (see M.4 for description; H.5 is the “hard” variant of M.4)
H.6 Ray Tracing

Today’s Agenda:

- Introduction to Ray Tracing
- Whitted-style Ray Tracing
- Intersections
- Shadows, Reflections & Refractions

Introduction

Rendering – “The God Algorithm”

Image synthesis in nature:

Large amounts of particles,
emitted by light sources,
bouncing around a scene,
until they reach a sensor.

Pro:
Simple.

Con:
No way we can simulate that many particles.

Introduction

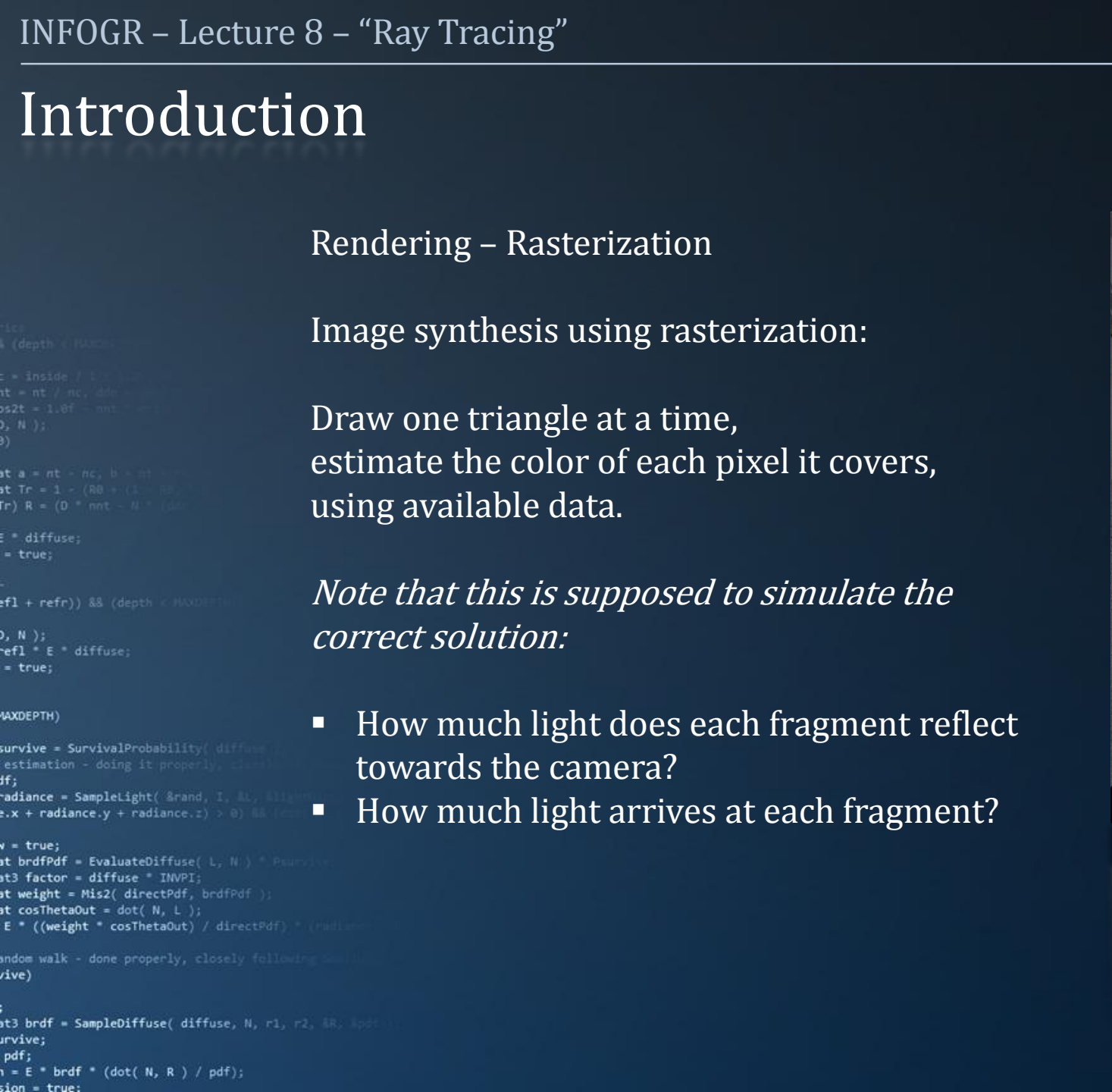
Rendering – Rasterization

Image synthesis using rasterization:

Draw one triangle at a time,
estimate the color of each pixel it covers,
using available data.

Note that this is supposed to simulate the correct solution:

- How much light does each fragment reflect towards the camera?
- How much light arrives at each fragment?



Introduction

Rasterization - Reflection

- How much light does each fragment reflect towards the camera?
- How much light arrives at each fragment?

We can answer these questions accurately for a limited scene:

- One or more point lights
- A single convex object

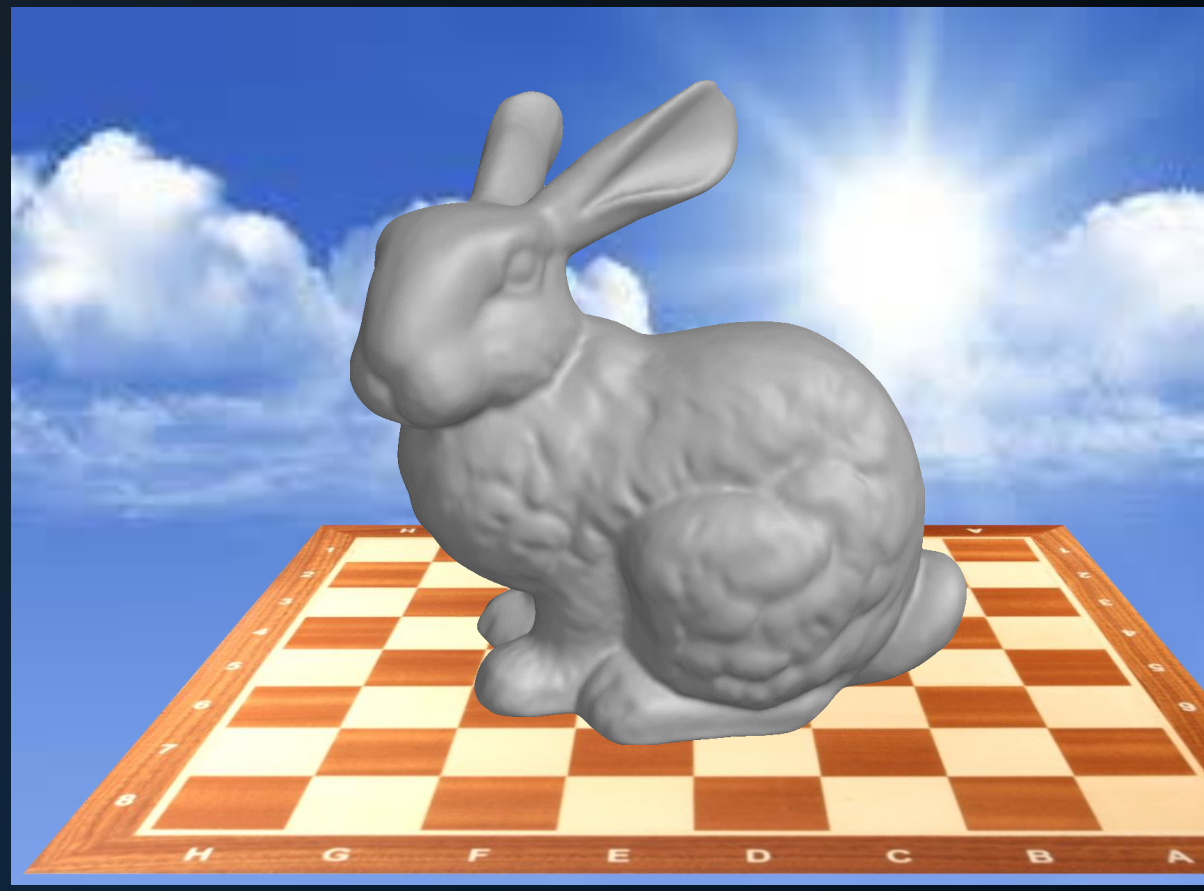
Introduction

Rasterization - Reflection

For a more elaborate scene, things break:

- Occluded incoming light (shadows)
- Indirect light

It gets worse if we don’t limit ourselves to point lights.



Introduction

Rasterization - Reflection

We can simulate certain light transport paths, such as reflections:

Here, the light arriving at the bunny is stored in an ‘environment map’.

This is only accurate when reflecting an infinitely far environment.

Introduction

Rasterization - Shadows

We can simulate occlusion using shadow maps:

A z-buffer from the viewpoint of the light source allows us to see, per fragment, if we are closer to the light than an occluder.

This works only for point lights.

The shadow map is a raster, which is often visible.

Introduction

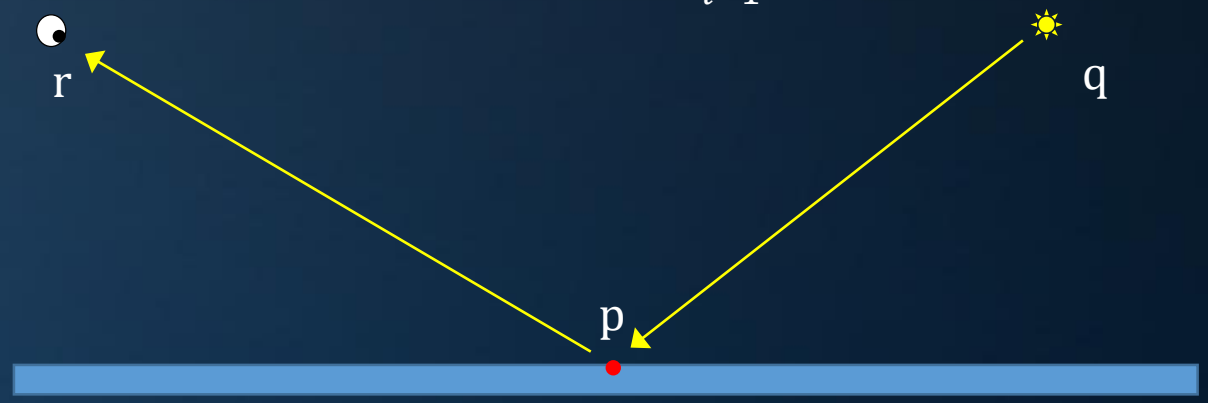
Rasterization – Fundamental limitations

When drawing triangles one by one, what we lack is *global data.*

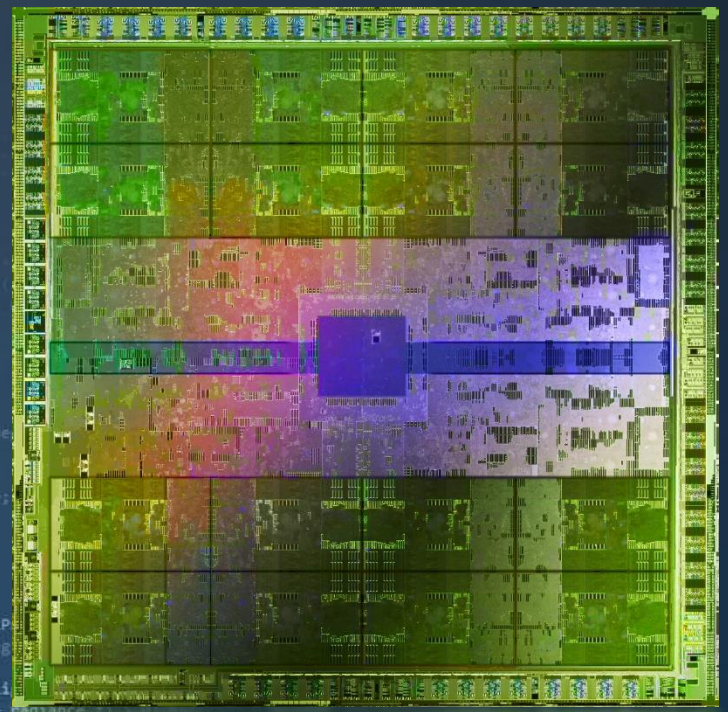
Environment maps and shadow maps capture some data about the environment of the triangle, but this data is a (coarse) approximation.

The core algorithm supports direct light, without occlusion:

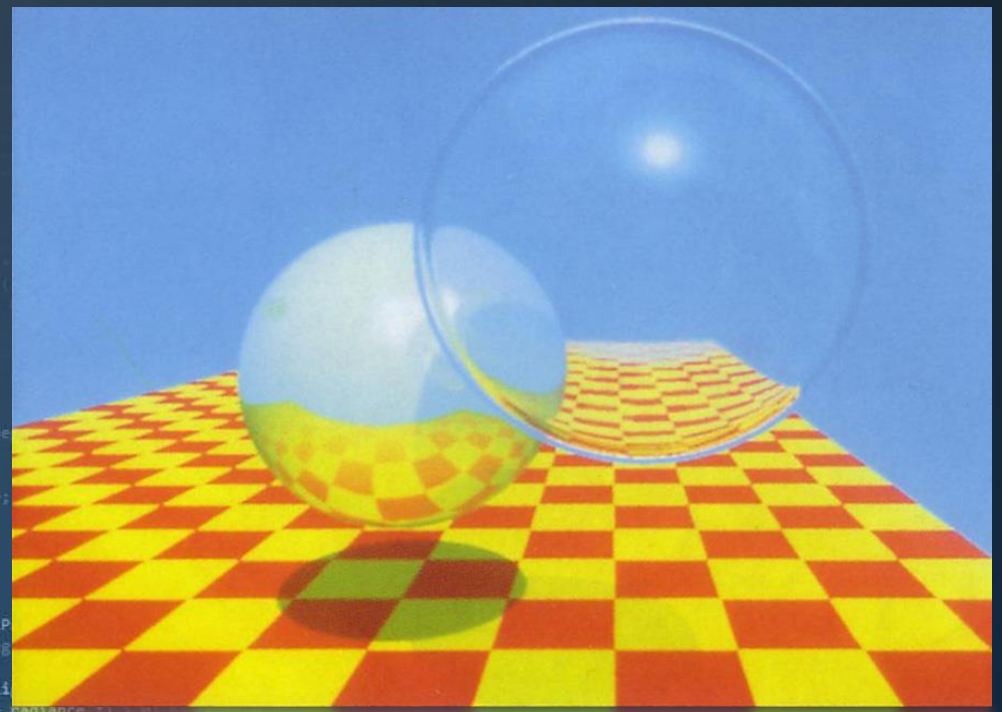
$$L(p \rightarrow r) = L_e(p \rightarrow r) + \sum_{i=1}^{N_L} L(q_i \rightarrow p)\, f_r\,(q_i \rightarrow p \rightarrow r)\, G(q_i \leftrightarrow p)$$



Introduction



Introduction




QUAKE WARS
RAYTRACED

Introduction

Ray Tracing

This lecture: basic algorithm
Next lecture: ‘ground truth’

After that, we converge with rasterization again.

Today’s Agenda:

- Introduction to Ray Tracing
- Whitted-style Ray Tracing
- Intersections
- Shadows, Reflections & Refractions

Ray Tracing

Ray Tracing:

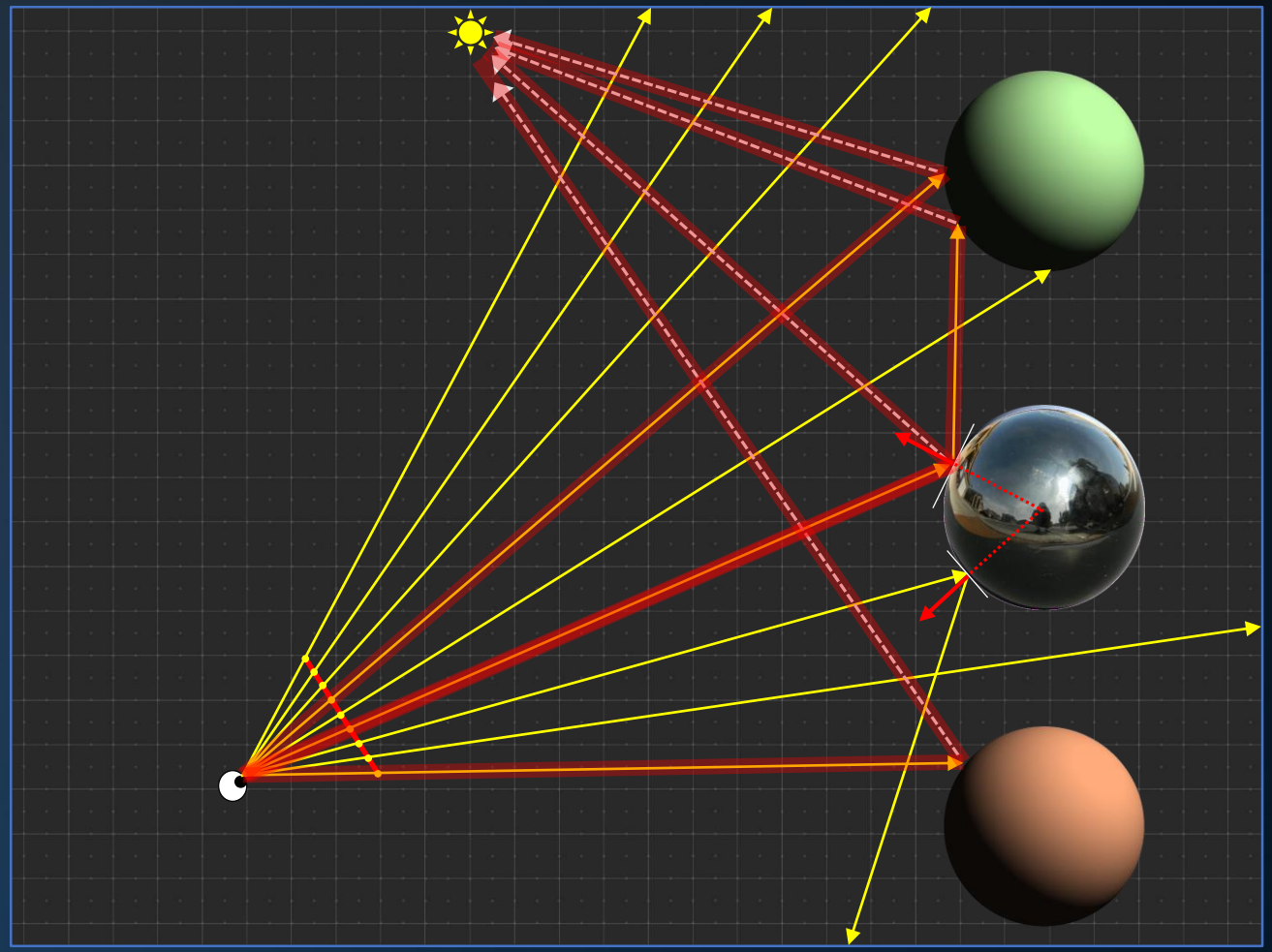
World space

- Geometry
- Eye
- Screen plane
- Screen pixels
- Primary rays
- Intersections
- Point light
- Shadow rays

Light transport

- Extension rays

Light transport



Ray Tracing

Ray Tracing:

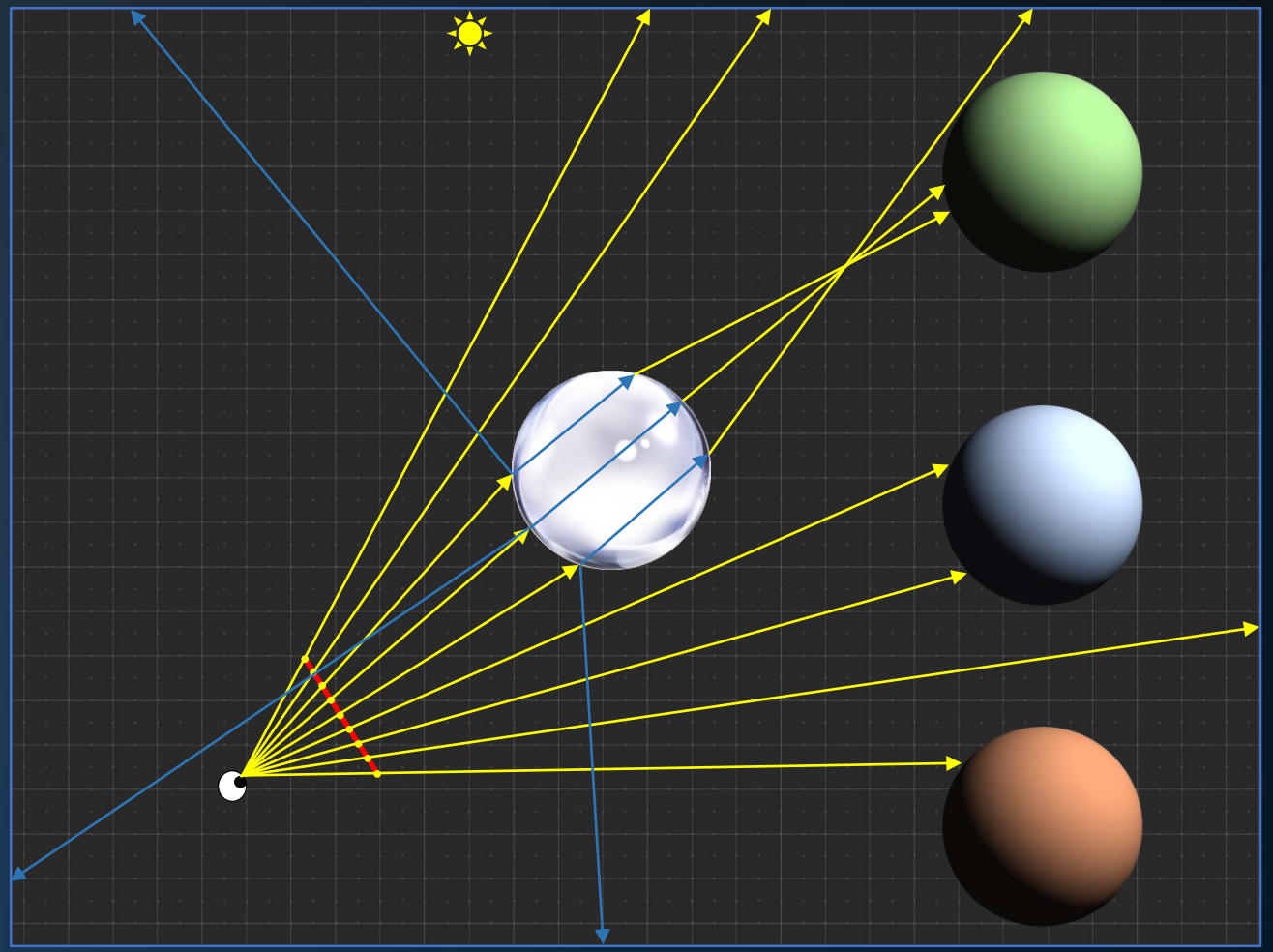
World space

- Geometry
- Eye
- Screen plane
- Screen pixels
- Primary rays
- Intersections
- Point light
- Shadow rays

Light transport

- Extension rays

Light transport



Ray Tracing

Ray Tracing:

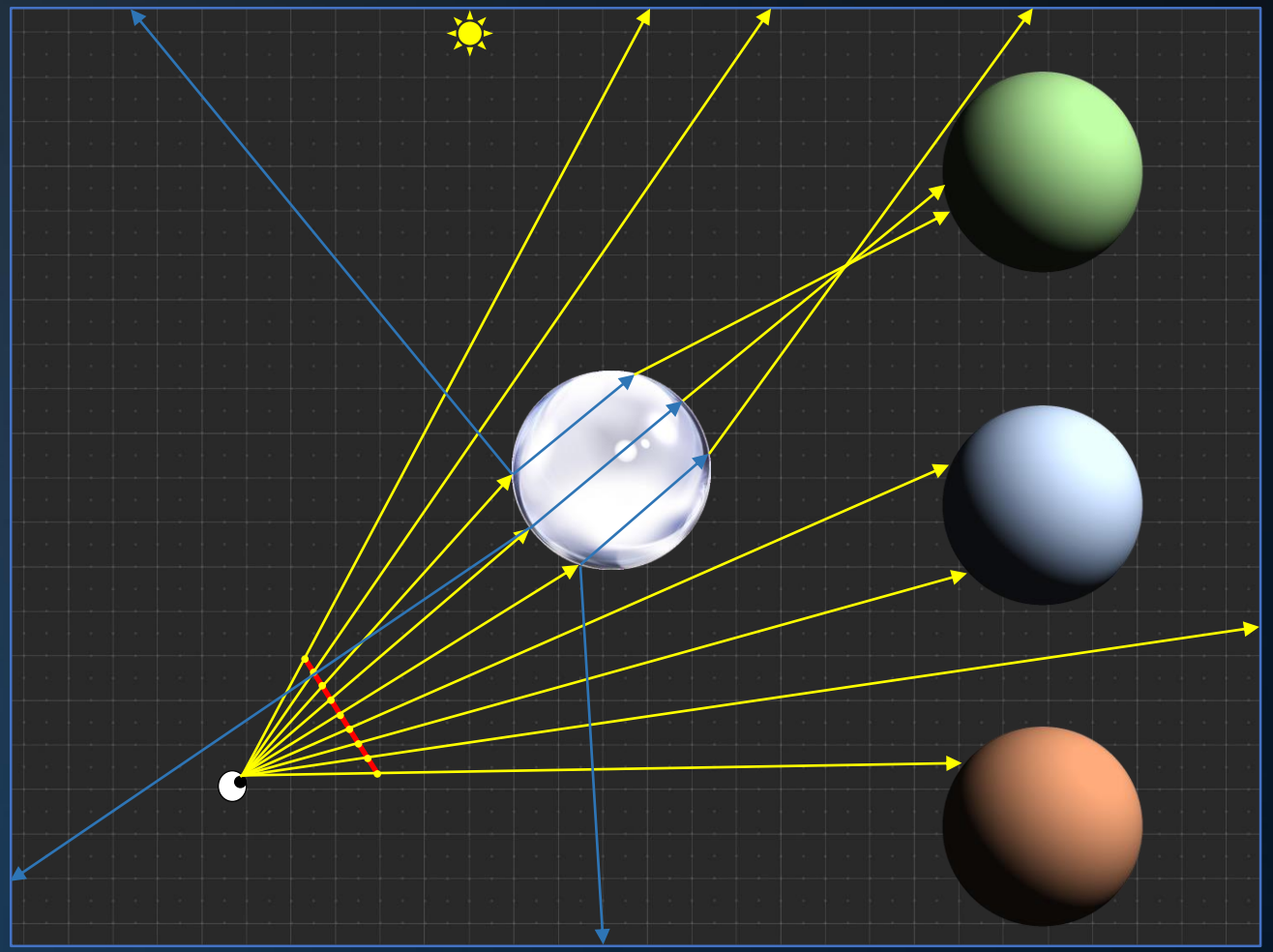
World space

- Geometry
- Eye
- Screen plane
- Screen pixels
- Primary rays
- Intersections
- Point light
- Shadow rays

Light transport

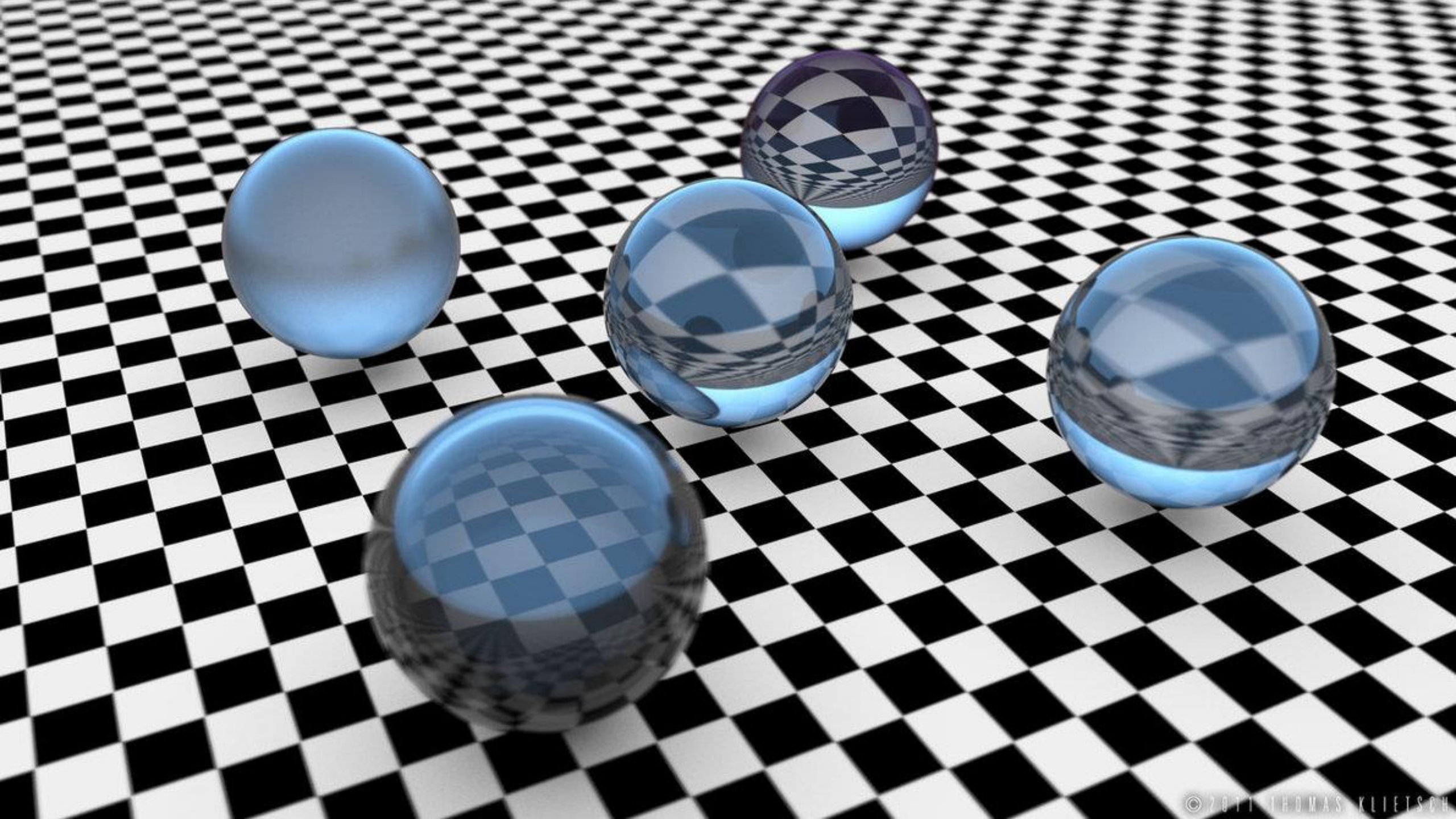
- Extension rays

Light transport

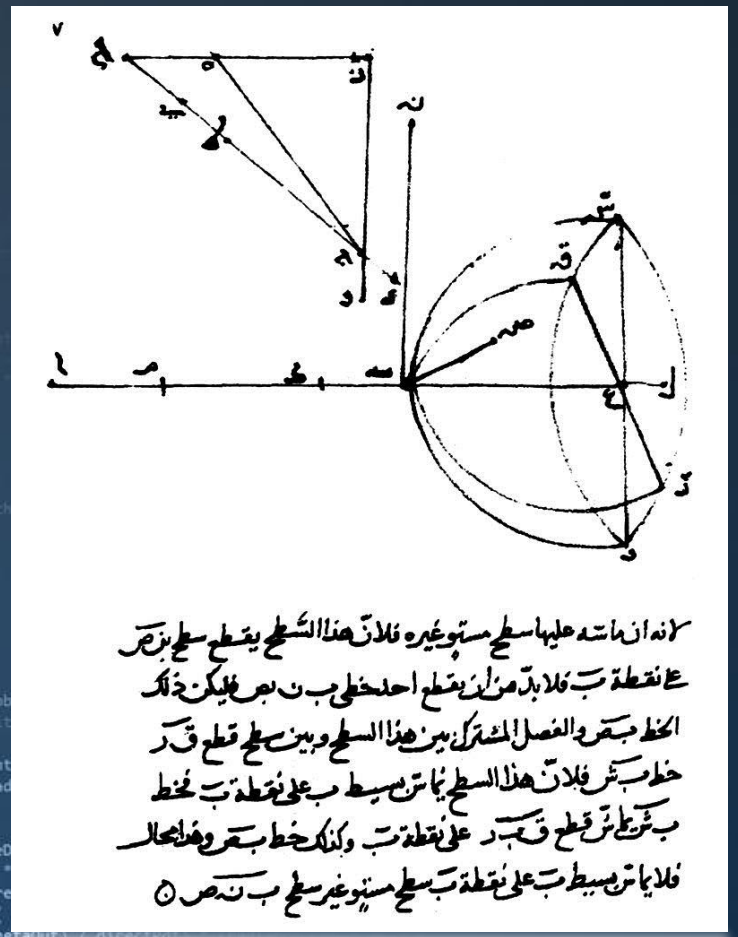


Note:

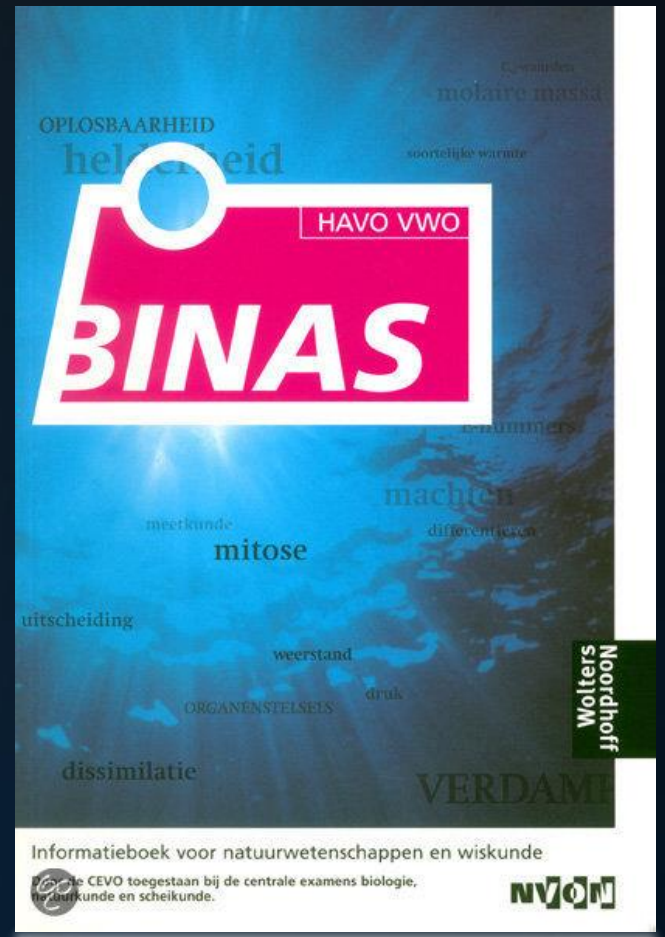
We are calculating light transport *backwards.*


© 2011 THOMAS KLIETSCH

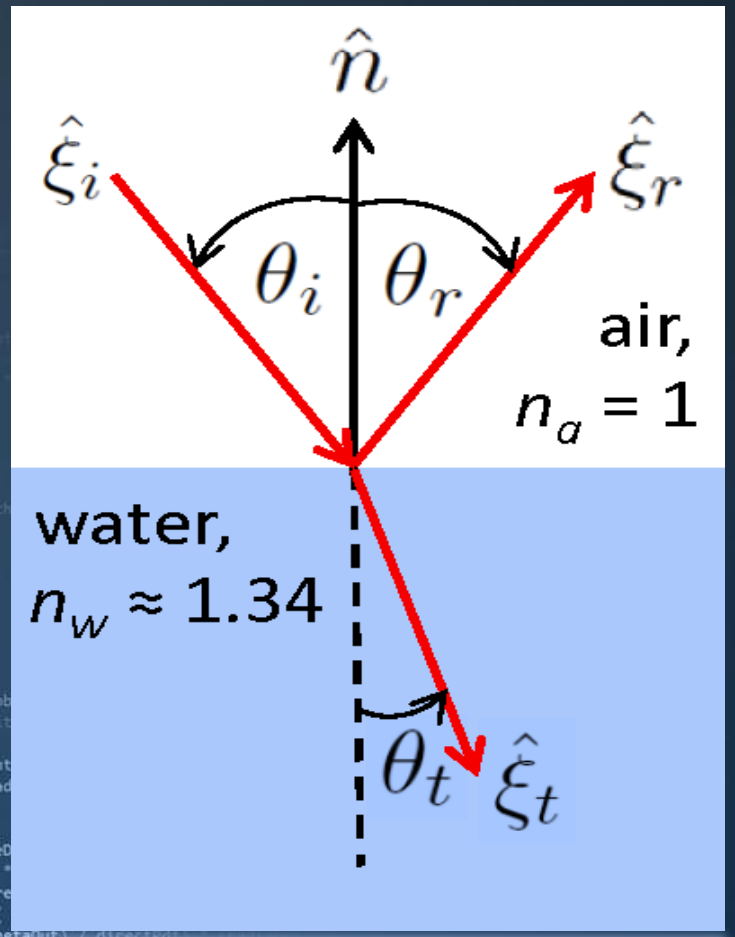
Ray Tracing

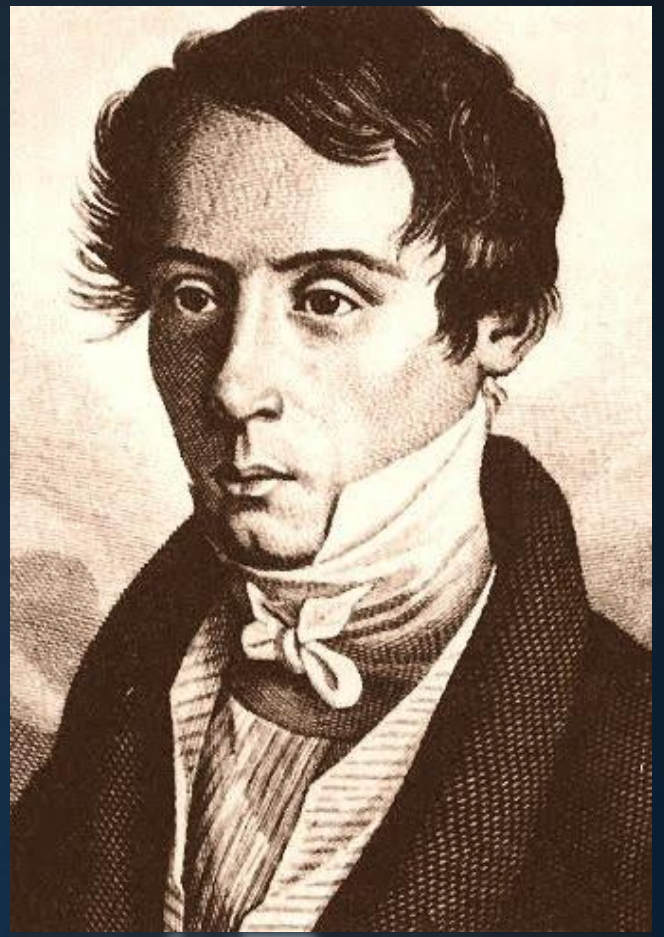


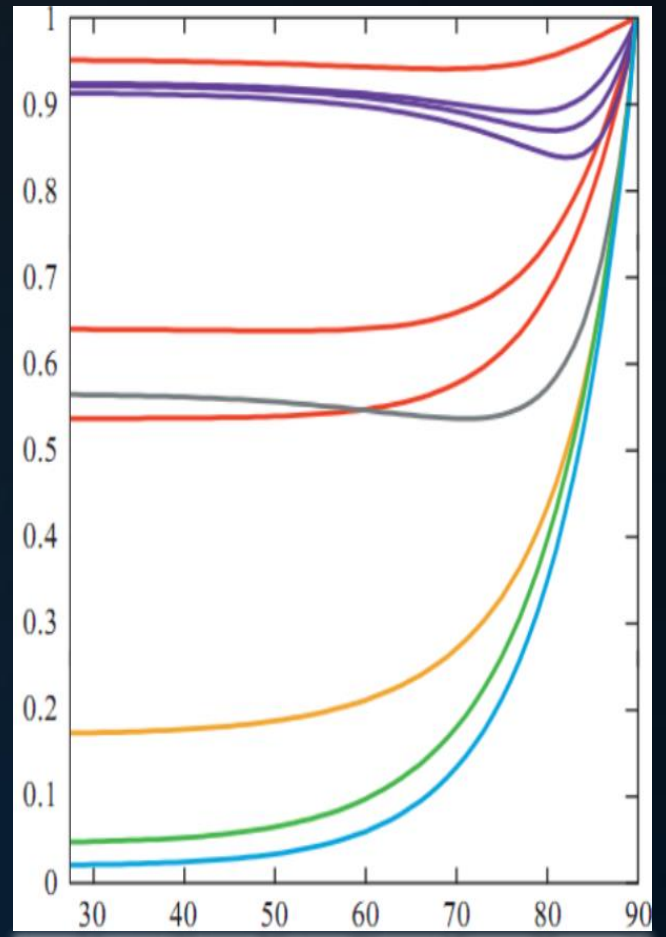

WILLEBRORDUS SNELLIUS.
Mathescos Professor.


OPLOSBAARHEID
molaire massa
soortelijke warmte
HAVO VWO
BINAS
machten
meetkunde
differentiëren
mitose
uitscheiding
weerstand
druk
ORGANENSTELSELS
dissimilatie
VERDAMPING
Wolters Noordhoff
Informatieboek voor natuurwetenschappen en wiskunde
NVON

Ray Tracing


$\hat{n}$
$\hat{\xi}_i$
$\hat{\xi}_r$
θ_i
θ_r
air,
$n_a = 1$
water,
$n_w \approx 1.34$
θ_t
$\hat{\xi}_t$




1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
30
40
50
60
70
80
90

Ray Tracing

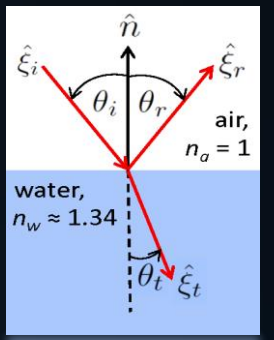
Physical basis:

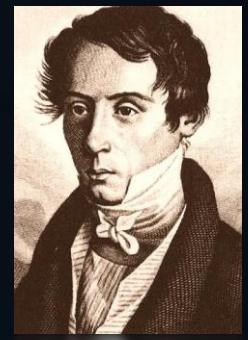
Ray tracing uses *ray optics* to simulate the behavior of light in a virtual environment.

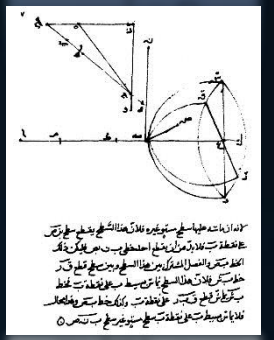
It does so by finding light transport paths:

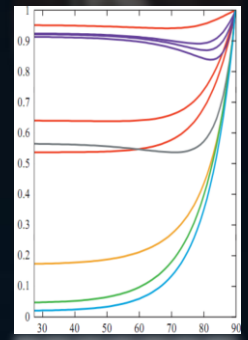
- From the ‘eye’
- Through a pixel
- Via scene surfaces
- To one or more light sources.

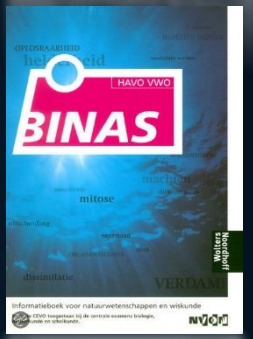
At each surface, the light is modulated.
The final value is deposited at the pixel
(simulating reception by a sensor).

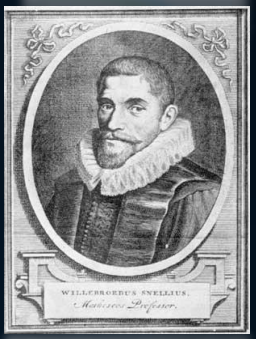












Today’s Agenda:

- Introduction to Ray Tracing
- Whitted-style Ray Tracing
- Intersections
- Shadows, Reflections & Refractions

Intersections

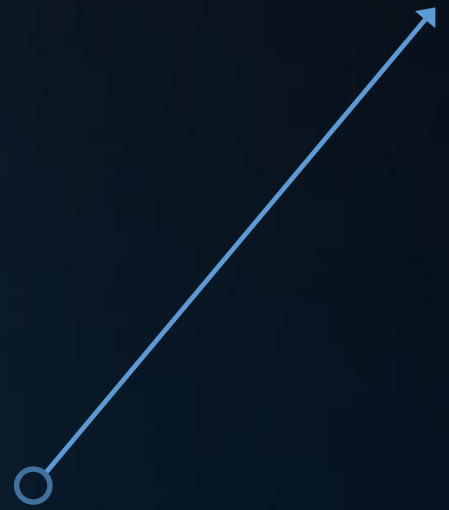
Ray definition

A ray is an infinite line with a start point:

$p(t) = O + t\vec{D}$, where $t > 0$.

```
struct Ray
{
    float3 O;    // ray origin
    float3 D;    // ray direction
    float t;     // distance
};
```

The ray direction is generally normalized.



Intersections

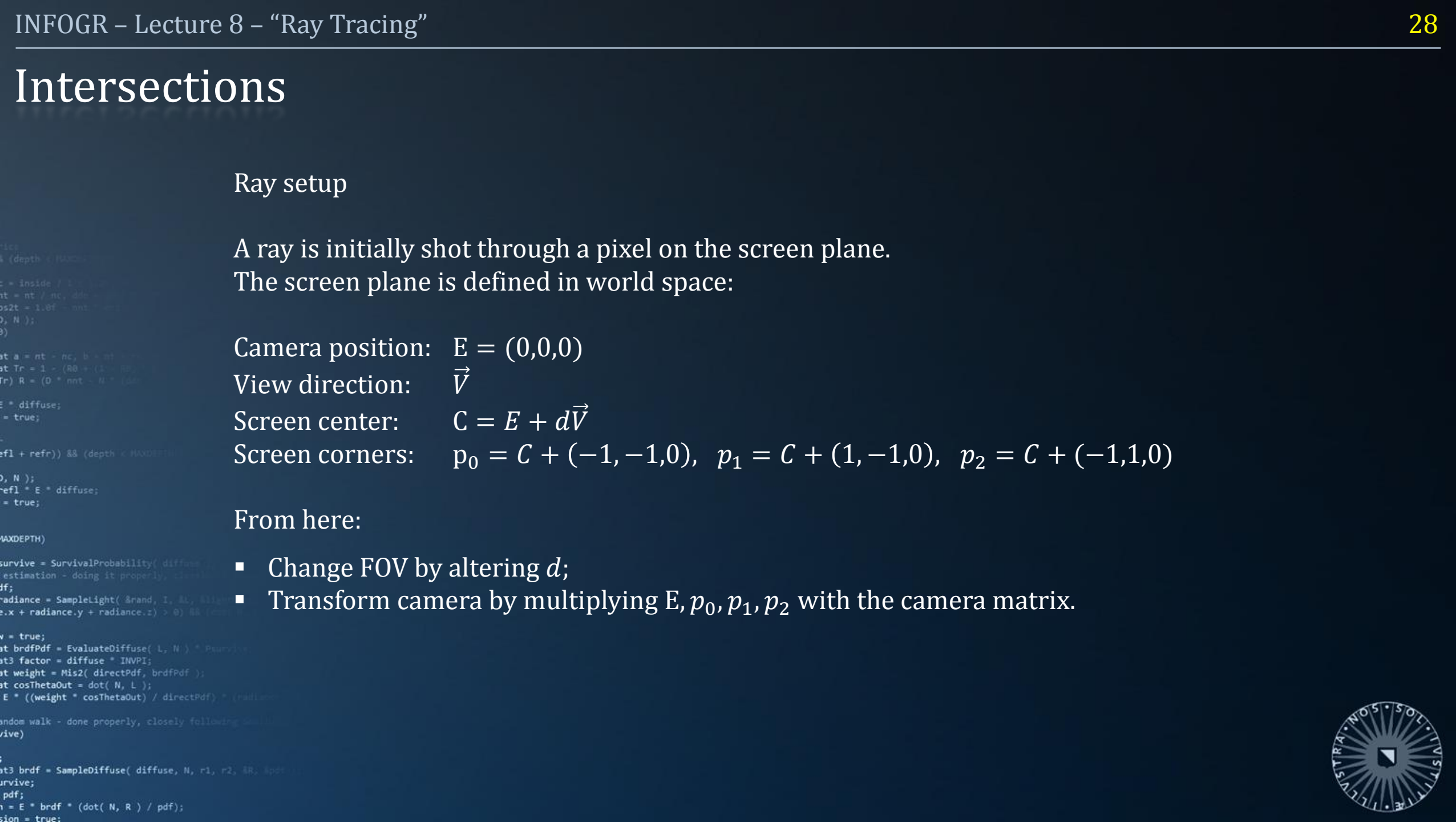
Ray setup

A ray is initially shot through a pixel on the screen plane.
The screen plane is defined in world space:

Camera position: $\mathrm{E} = (0,0,0)$
View direction: $\vec{V}$
Screen center: $\mathrm{C} = E + d\vec{V}$
Screen corners: $\mathrm{p}_0 = C + (-1,-1,0),\ \ p_1 = C + (1,-1,0),\ \ p_2 = C + (-1,1,0)$

From here:

- Change FOV by altering d;
- Transform camera by multiplying $\mathrm{E}, p_0, p_1, p_2$ with the camera matrix.



Intersections

Ray setup

Point on the screen:

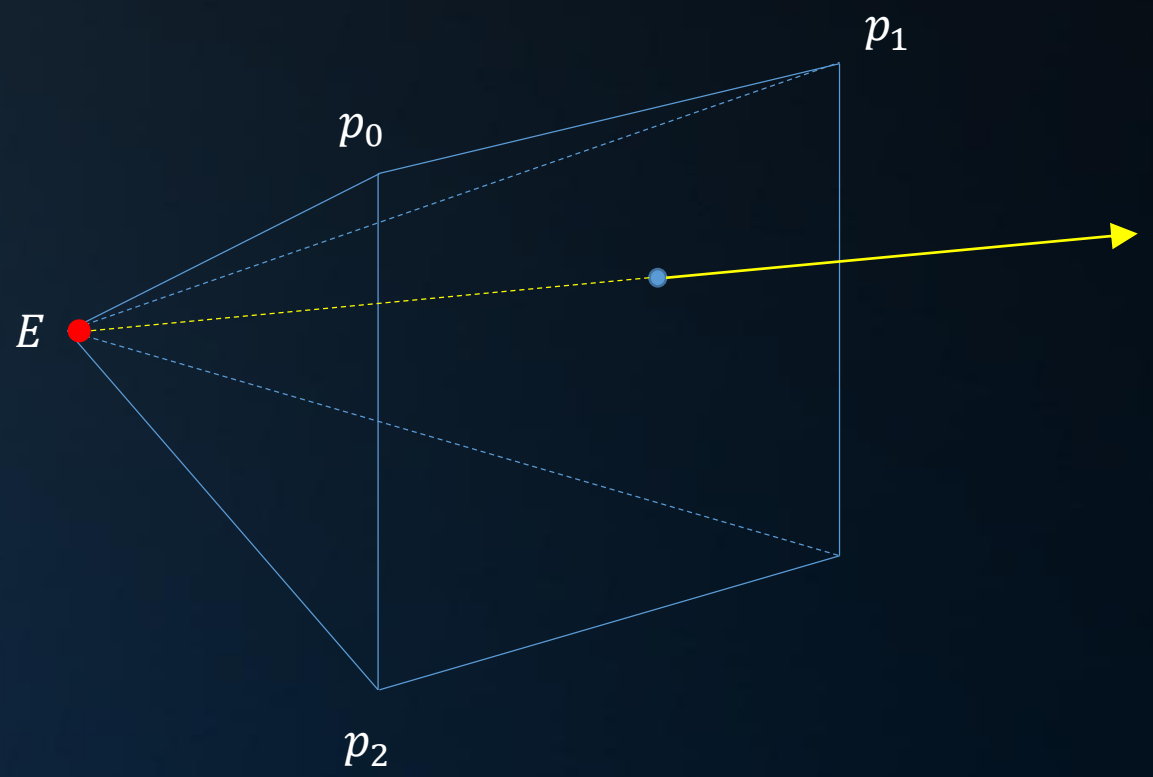
$$p(u,v) = p_0 + u(p_1 - p_0) + v(p_2 - p_0)$$

Ray direction (before normalization):

$$\vec{D} = p(u,v) - E$$

Ray origin:

$$O = E$$



Intersections

Ray intersection

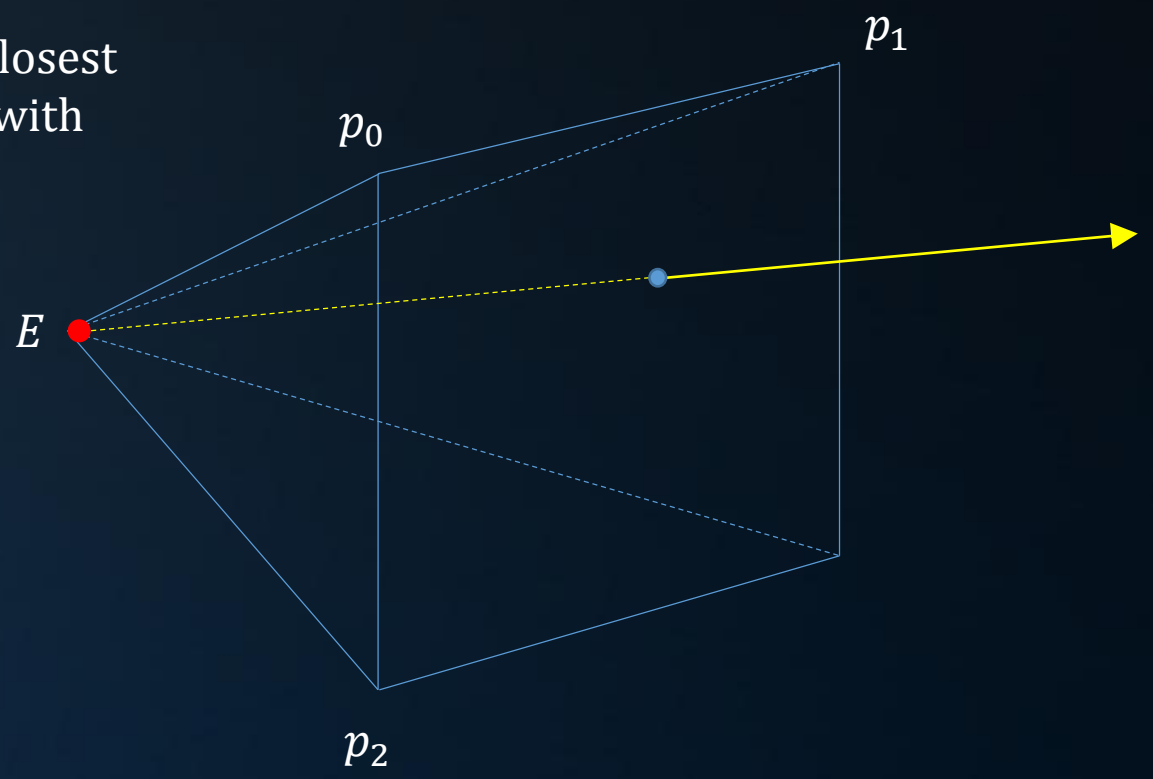
Given a ray $p(t) = O + t\vec{D}$, we determine the closest intersection distance t by intersecting the ray with each of the primitives in the scene.

Ray / plane intersection:

Plane: $\mathrm{p} \cdot \vec{N} + d = 0$
Ray: $p(t) = O + t\vec{D}$

Substituting for $p(t)$, we get

$$(O + t\vec{D}) \cdot \vec{N} + d = 0$$
$$t = -(O \cdot \vec{N} + d)/(\vec{D} \cdot \vec{N})$$
$$P = O + t\vec{D}$$



Intersections

Ray intersection

Ray / sphere intersection:

Sphere: $(p - C) \cdot (p - C) - r^2 = 0$

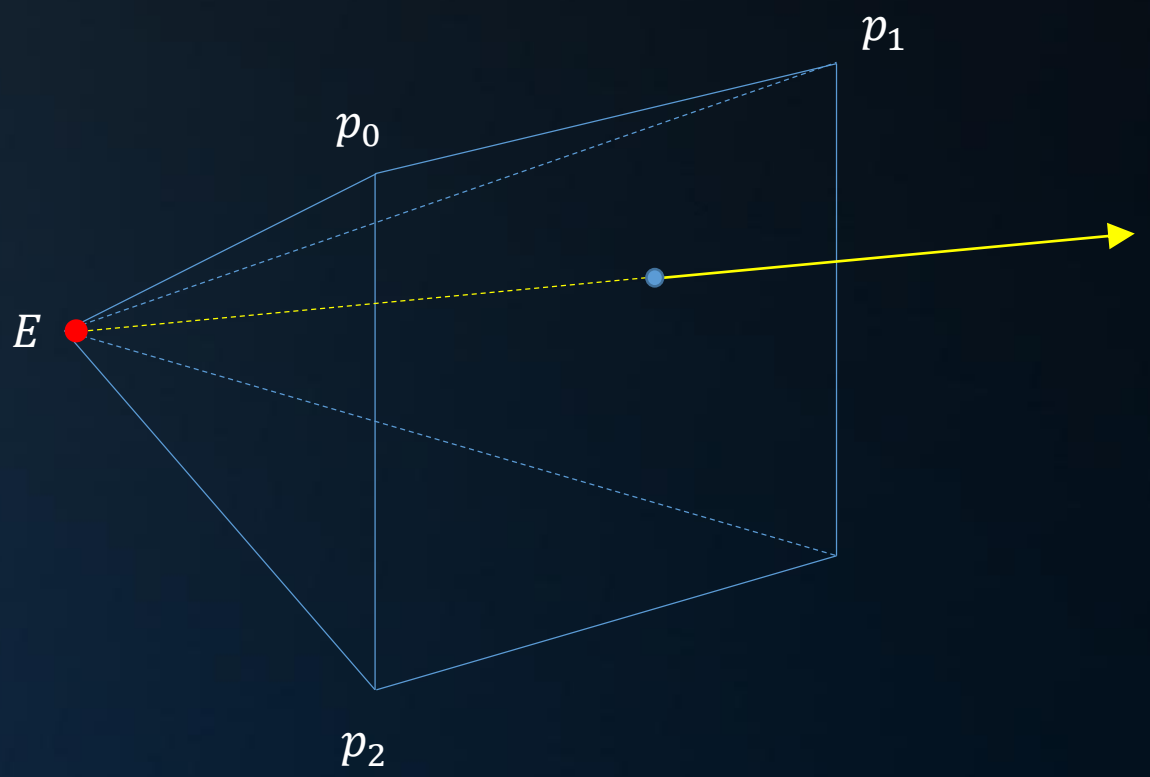
Substituting for $p(t)$, we get

$$(O + t\vec{D} - C) \cdot (O + t\vec{D} - C) - r^2 = 0$$
$$\vec{D} \cdot \vec{D}\ t^2 + 2\vec{D} \cdot (O - C)\, t + (O - C)^2 - r^2 = 0$$

$$ax^2 + bx + c = 0 \rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$a = \vec{D} \cdot \vec{D}$$
$$b = 2\vec{D} \cdot (O - C)$$
$$c = (O - C) \cdot (O - C) - r^2$$

Negative:
no intersections



Today’s Agenda:

- Introduction to Ray Tracing
- Whitted-style Ray Tracing
- Intersections
- Shadows, Reflections & Refractions

Whitted

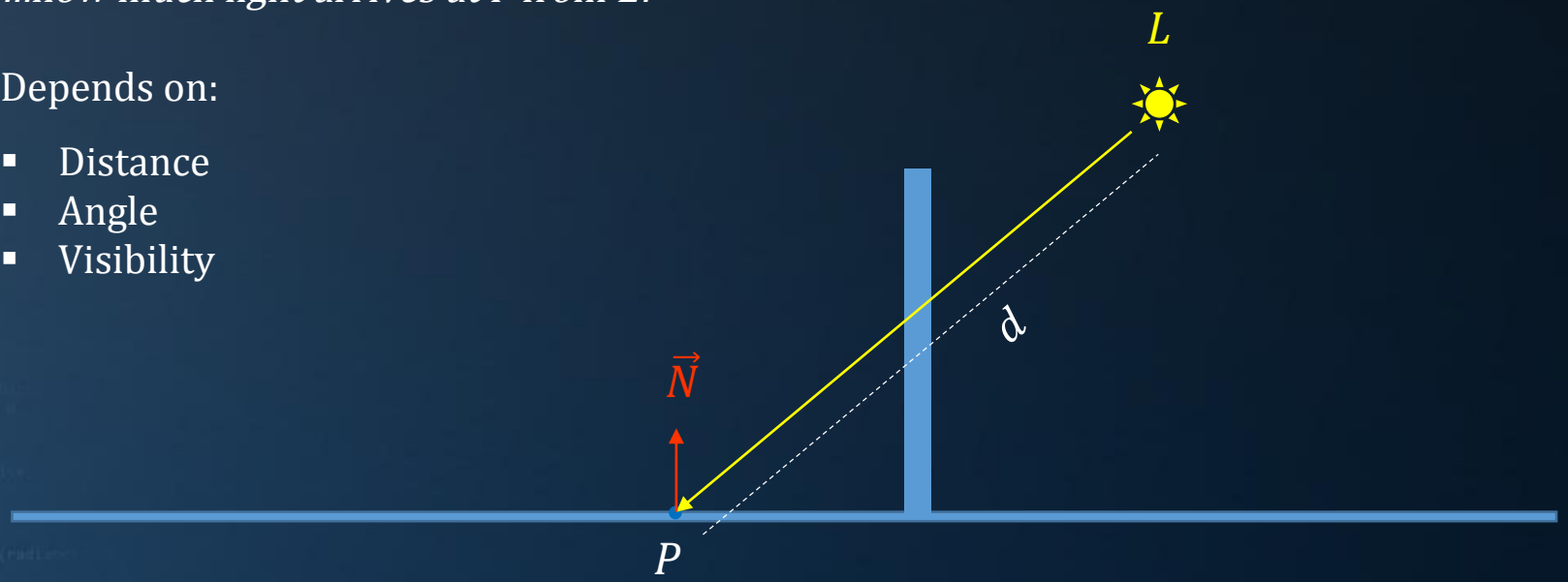
Shading basics

Given a point light L,
at distance d from a surface point P
with normal $\vec{N}$…

…how much light arrives at P from L?

Depends on:

- Distance
- Angle
- Visibility



Whitted

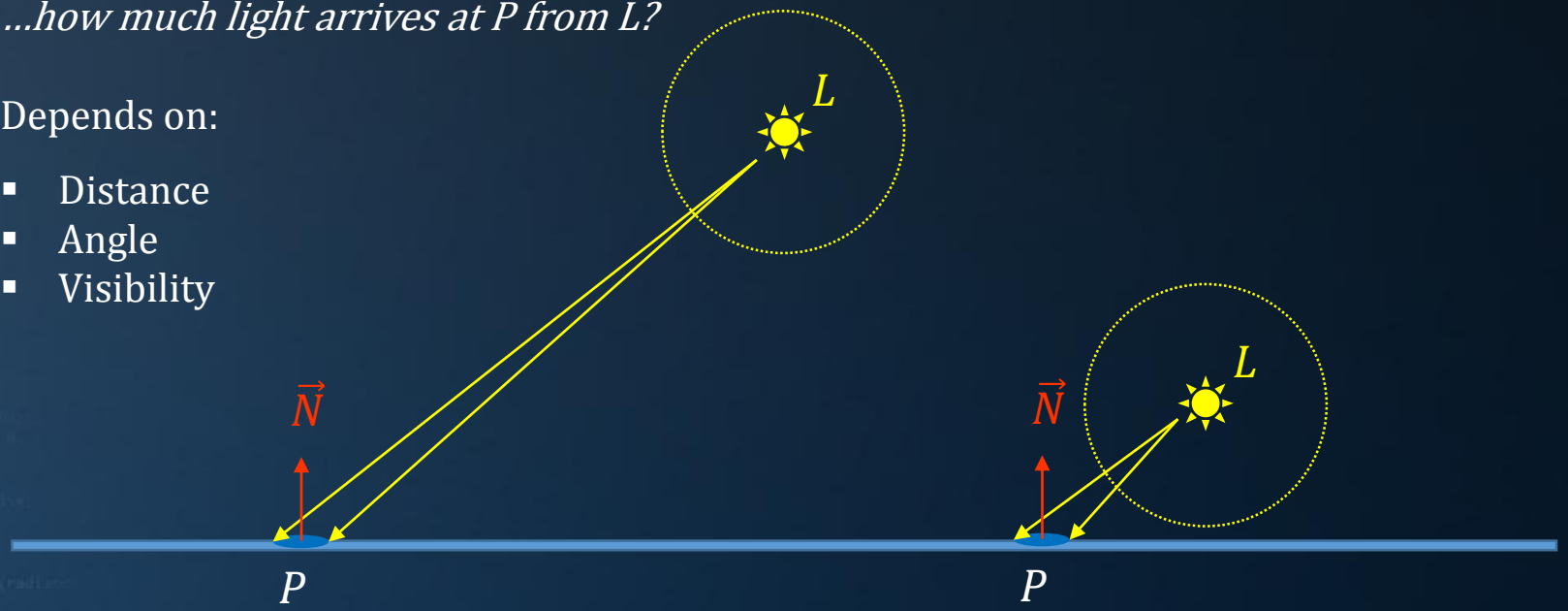
Shading basics

Given a point light L,
at distance d from a surface point P
with normal $\vec{N}$…

…how much light arrives at P from L?

Depends on:

- Distance
- Angle
- Visibility



Relation:

$$\frac{1}{d^2}$$

Distance attenuation

Whitted

Shading basics

Given a point light L,
at distance d from a surface point P
with normal $\vec{N}$…

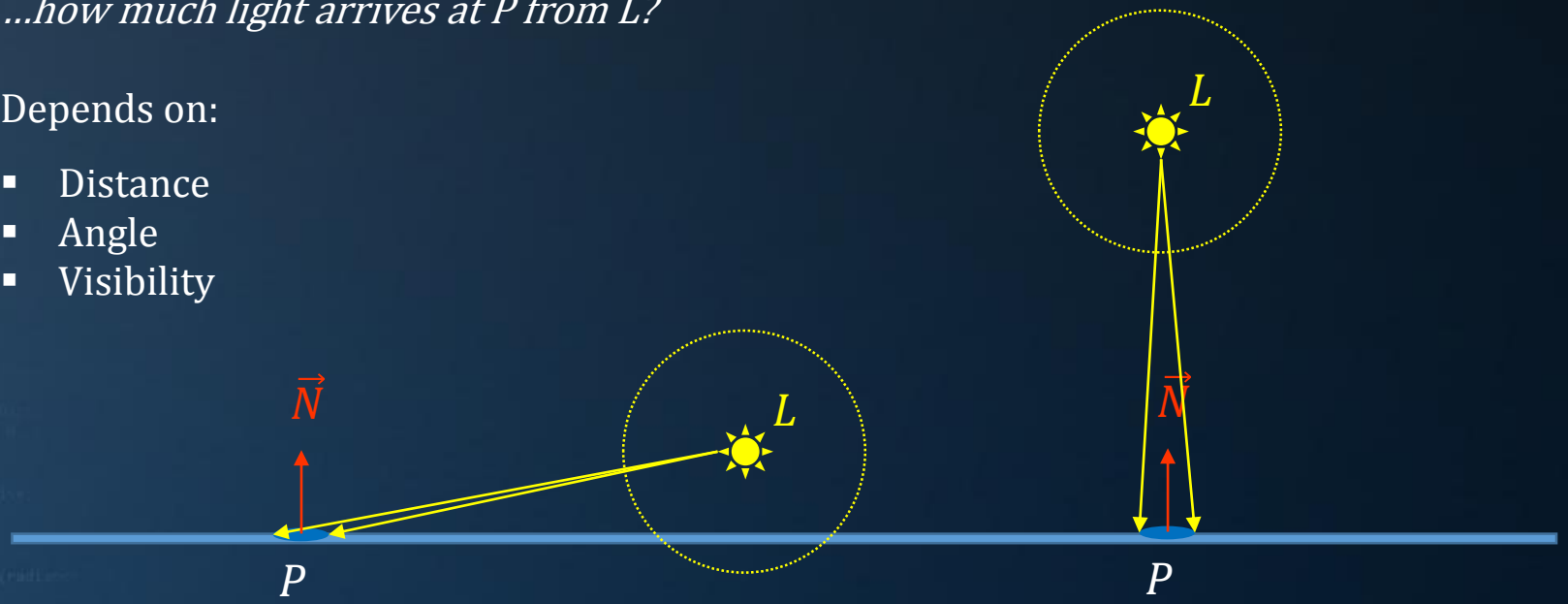
…how much light arrives at P from L?

Depends on:

- Distance
- Angle
- Visibility

Relation:

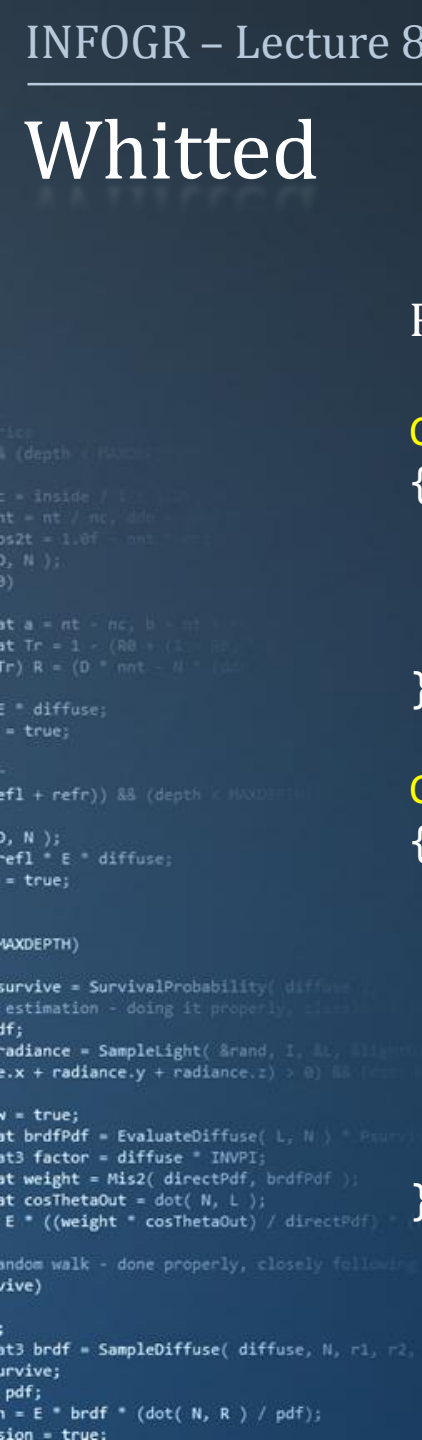
$$\vec{N} \cdot \frac{L - P}{|L - P|}$$



Whitted

Ray intersection

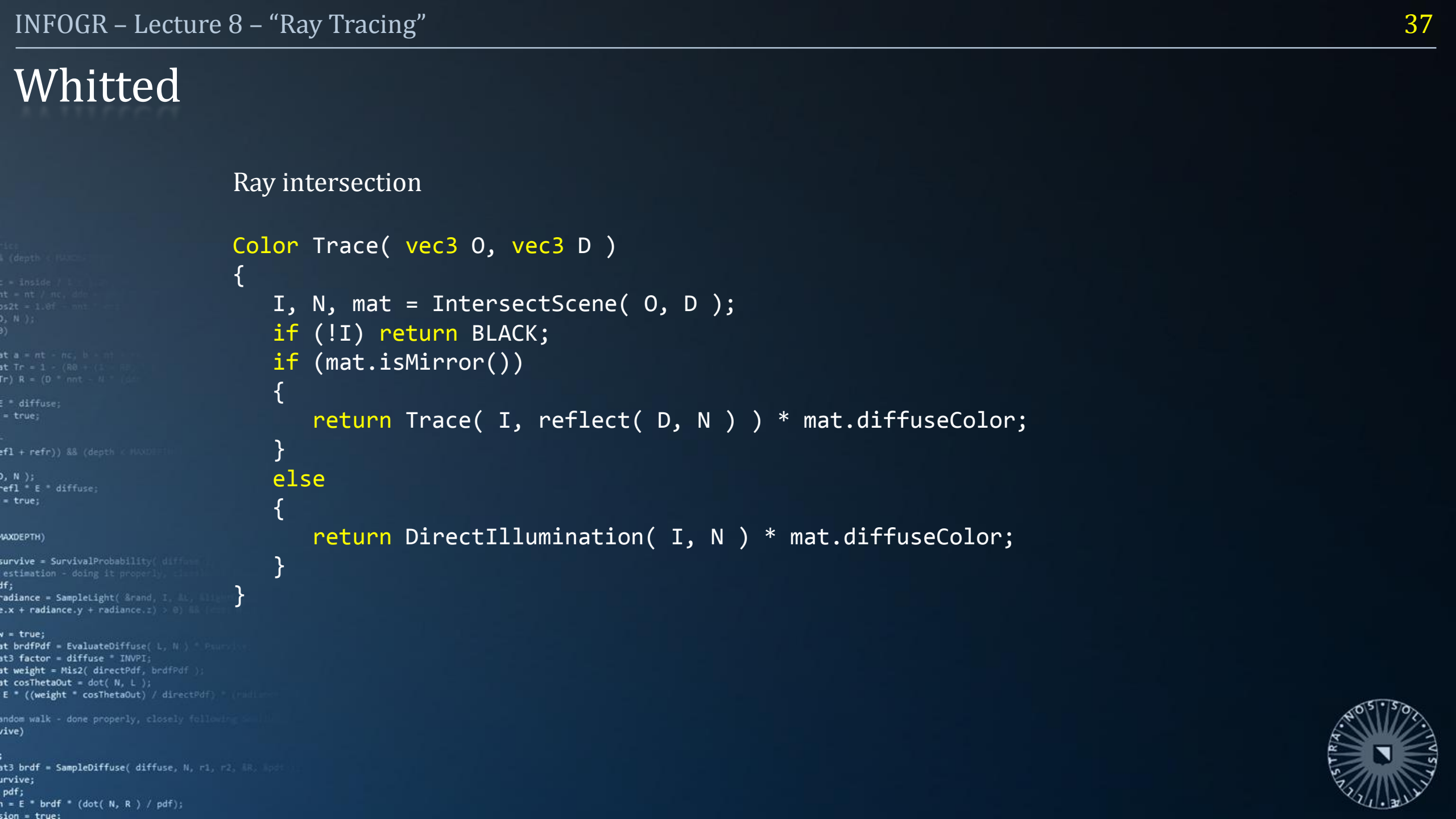
```
Color Trace( vec3 O, vec3 D )
{
   I, N, mat = IntersectScene( O, D );
   if (!I) return BLACK;
   return DirectIllumination( I, N ) * mat.diffuseColor;
}

Color DirectIllumination( vec3 I, vec3 N )
{
   vec3 L = lightPos – I;
   float dist = length( L );
   L /= dist;
   if (!IsVisibile( I, L, dist )) return BLACK;
   float attenuation = 1 / (dist * dist);
   return lightColor * dot( N, L ) * attenuation;
}
```



Whitted

Ray intersection

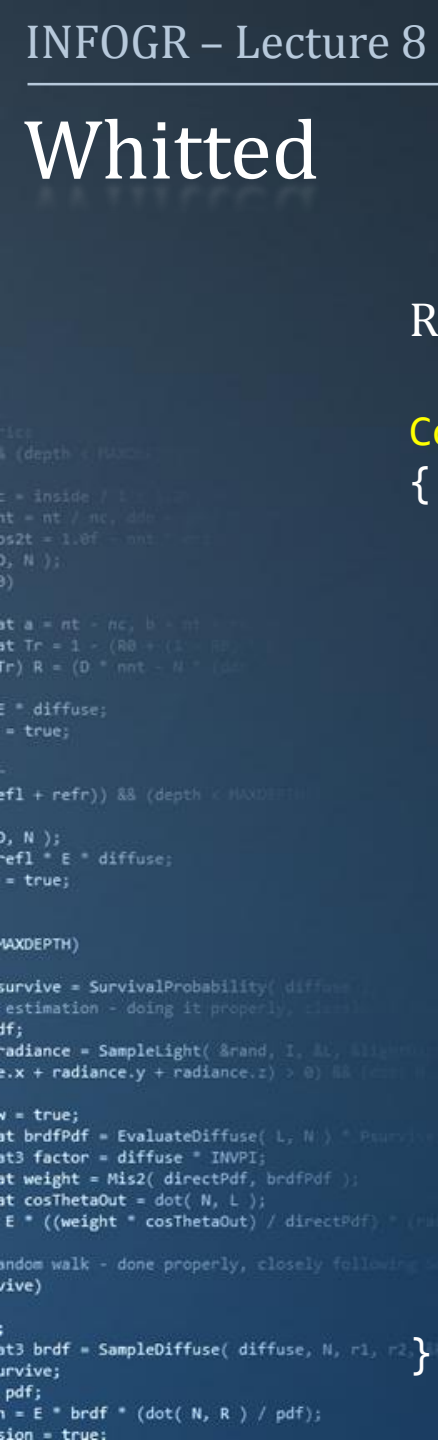
```
Color Trace( vec3 O, vec3 D )
{
   I, N, mat = IntersectScene( O, D );
   if (!I) return BLACK;
   if (mat.isMirror())
   {
      return Trace( I, reflect( D, N ) ) * mat.diffuseColor;
   }
   else
   {
      return DirectIllumination( I, N ) * mat.diffuseColor;
   }
}
```



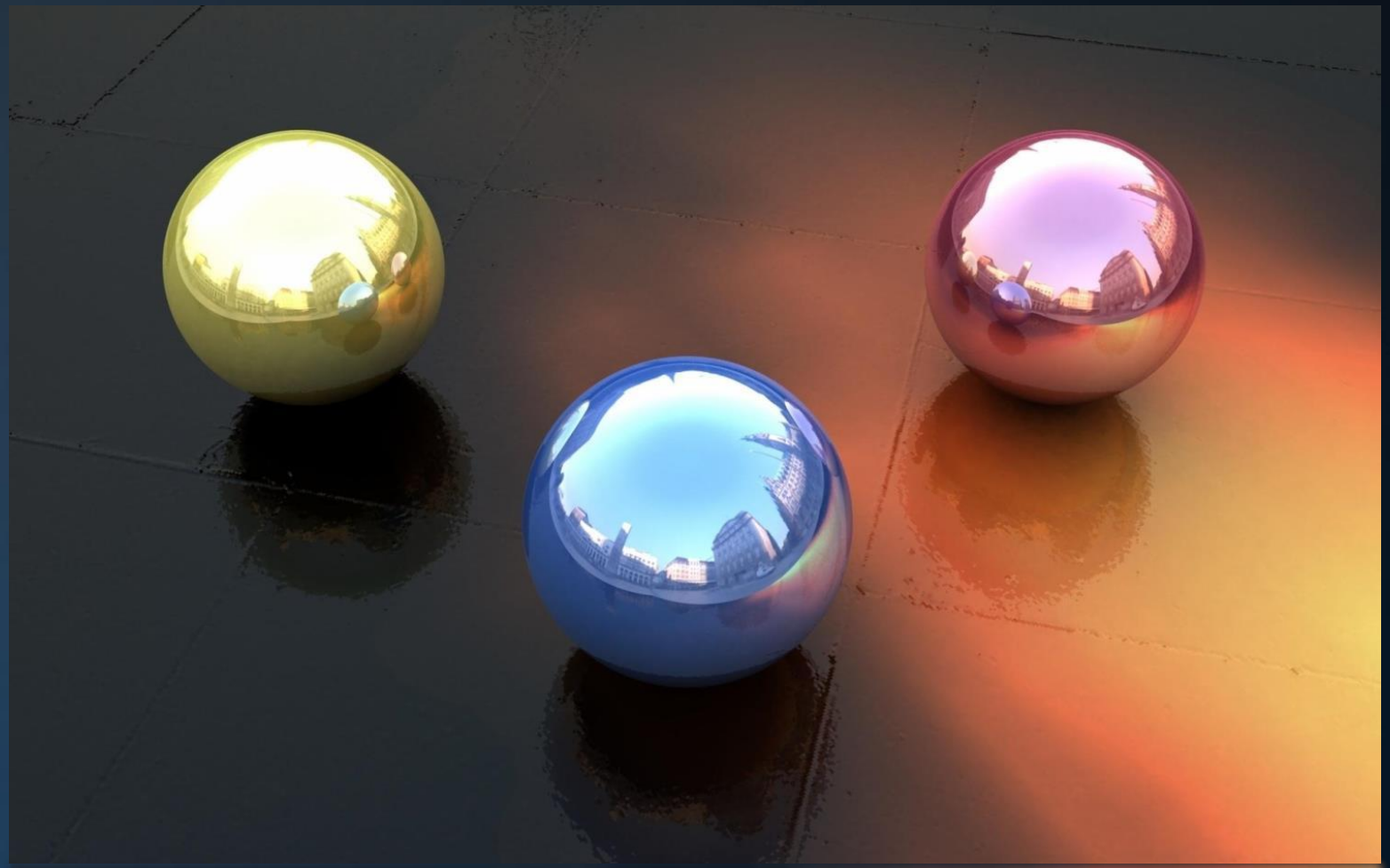
Whitted

Ray intersection

```
Color Trace( vec3 O, vec3 D )
{
   I, N, mat = IntersectScene( O, D );
   if (!I) return BLACK;
   if (mat.isMirror())
   {
      return Trace( I, reflect( D, N ) ) * mat.diffuseColor;
   }
   else if (mat.IsDielectric())
   {
      f = Fresnel( … );
      return (f * Trace( I, reflect( D, N ) )
        + (1-f) * Trace( I, refract( D, N, … ) ) ) * mat.DiffuseColor;
   }
   else
   {
      return DirectIllumination( I, N ) * mat.diffuseColor;
   }
}
```



Whitted



Today’s Agenda:

- Introduction to Ray Tracing
- Whitted-style Ray Tracing
- Intersections
- Shadows, Reflections & Refractions

INFOGR – Computer Graphics

dr. ing. J. Bikker - April-July 2015 - Lecture 8: “Ray Tracing”

END of “Ray Tracing”

next lecture: “Shading Models”