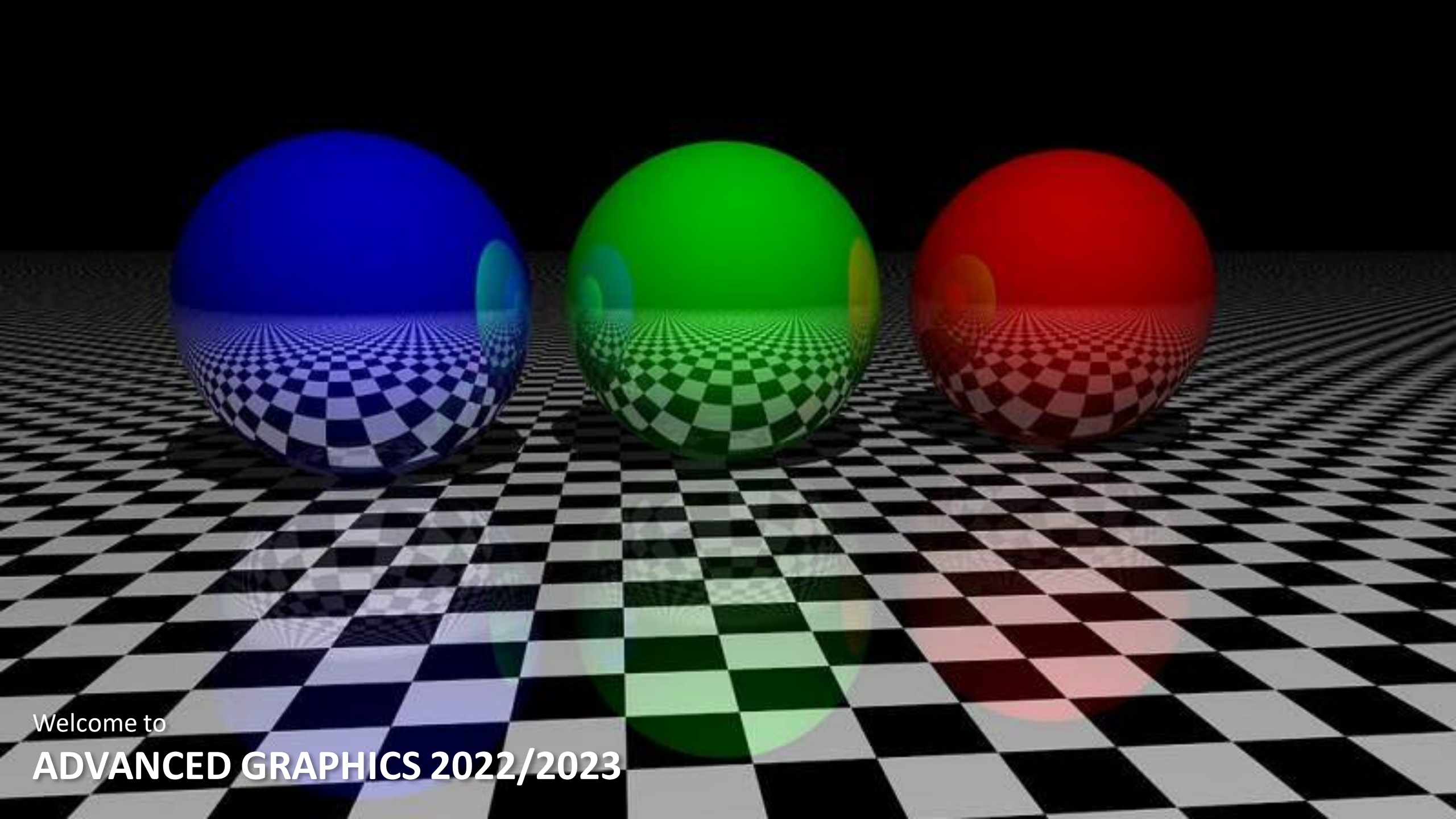
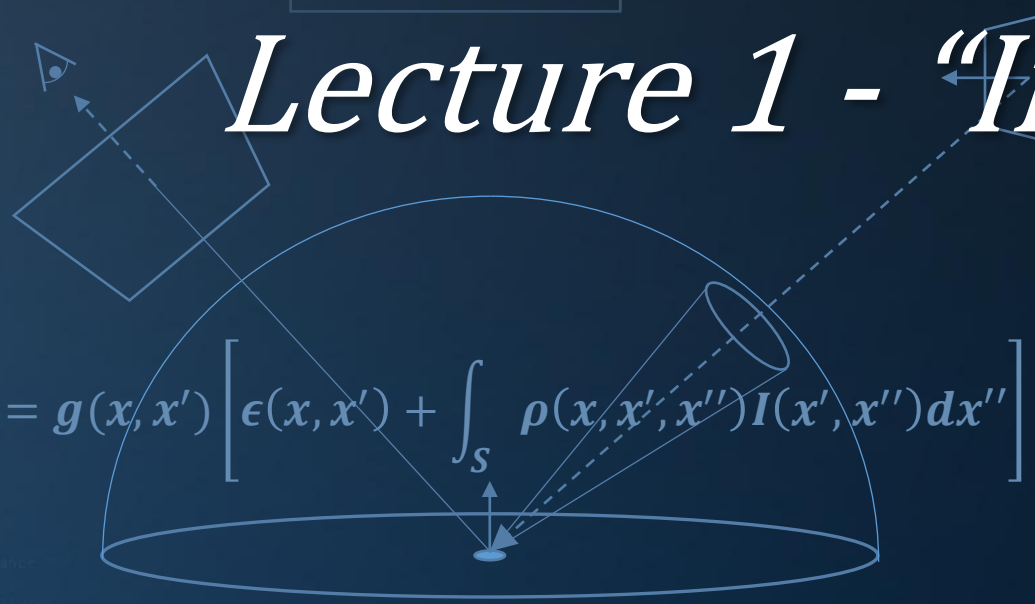

Welcome to
ADVANCED GRAPHICS 2022/2023

INFOMAGR – Advanced Graphics

Jacco Bikker - November 2022 - February 2023

Lecture 1 - "Introduction"



Welcome!

Today’s Agenda:

- Advanced Graphics
- Recap: Ray Tracing
- Assignment 1

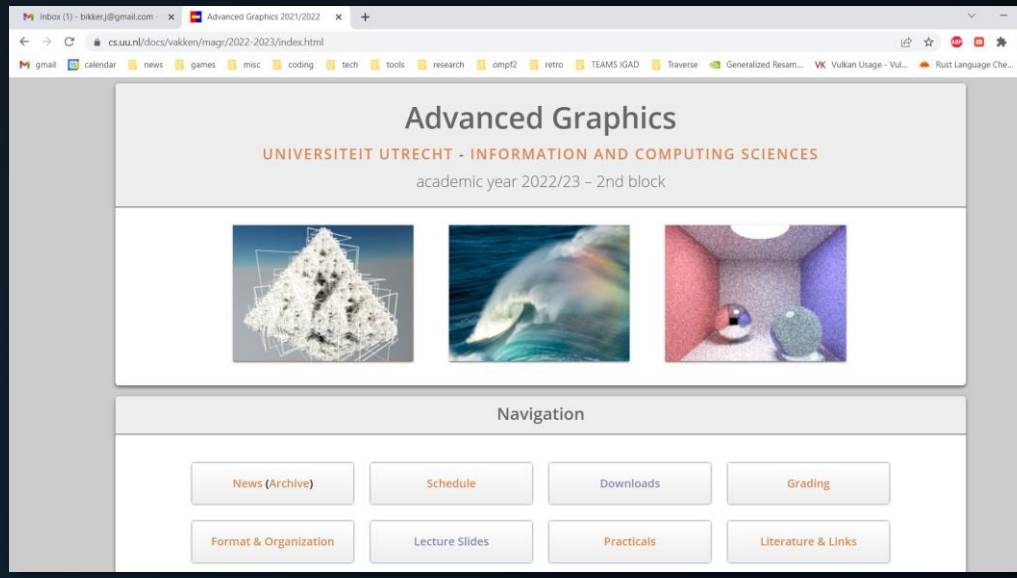
INFOMAGR

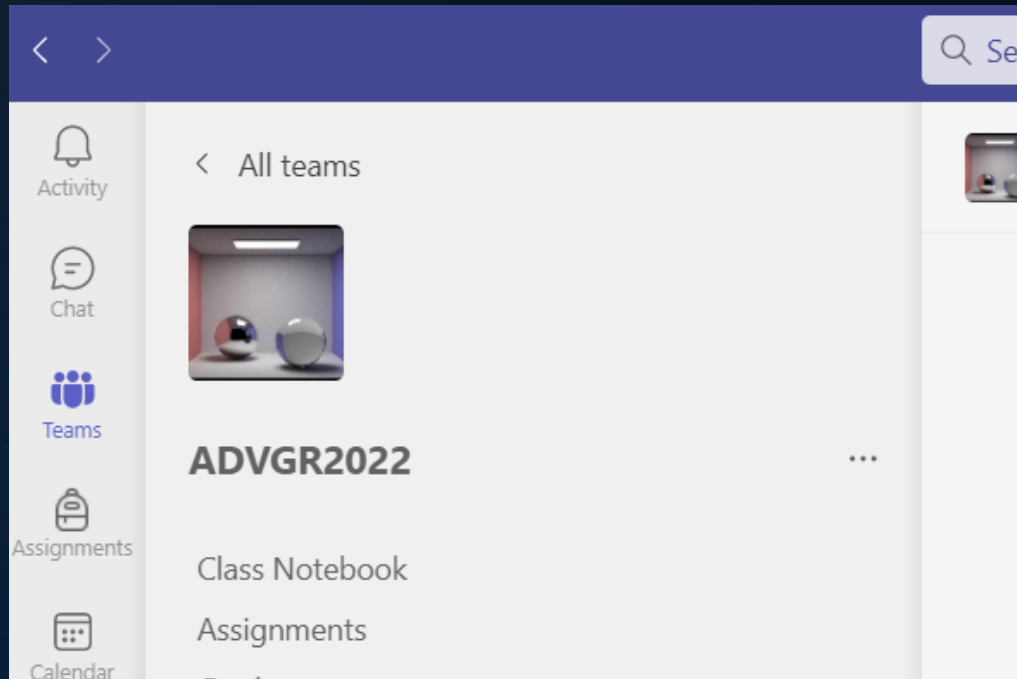
Website

http://www.cs.uu.nl/docs/vakken/magr

- Downloads, news, slides, deadlines, links.
- Main source of information!
- Includes complex weekly room allocation.

Please check Teams for operational comms.





INFOMAGR

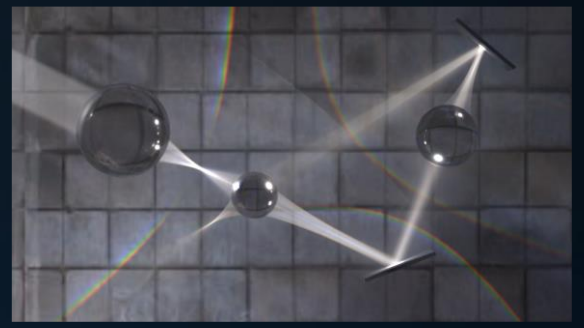
Abstract

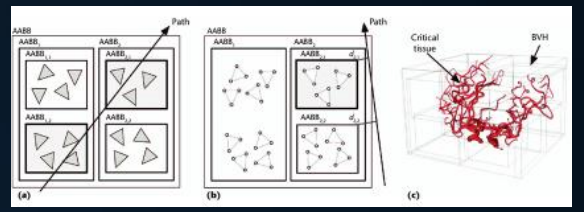
In this course, we explore *Physically Based Rendering* (PBR), with a focus on *interactivity*.

At the end of this course, you will have a solid theoretical understanding of efficient physically based light transport using ray tracing and stochastic evaluation of the Rendering Equation (so: *no rasterization, sorry*).

You will also have a good understanding of acceleration structures for fast ray/scene intersection for static and dynamic scenes.

You will have hands-on experience with algorithms for efficient realistic rendering of static and dynamic scenes using ray tracing on CPU and GPU.





INFOMAGR

Abstract

In this course, we explore *physically based rendering*, with a focus on interactivity.

At the end of this course, you will have a solid theoretical understanding of efficient physically based light transport using ray tracing and stochastic evaluation of the Rendering Equation (so: *no rasterization, sorry*).

You will also have a good understanding of acceleration structures for fast ray/scene intersection for static and dynamic scenes.

You will have hands-on experience with algorithms for efficient realistic rendering of static and dynamic scenes using ray tracing on CPU and GPU.

Concrete / informal:

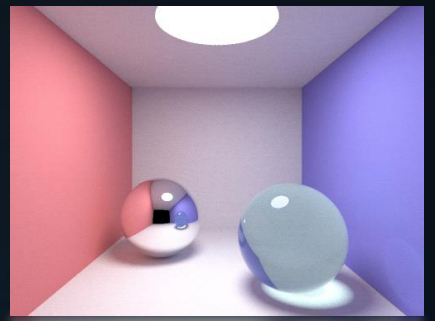
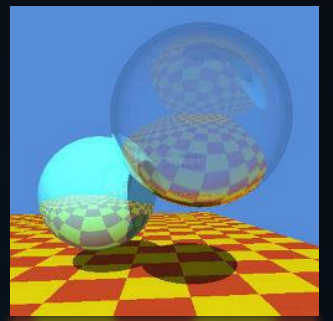
1. You'll know how a photo-realistic image is produced
2. You know how to do this quickly / efficient
3. You have built such a renderer
4. You have built an interactive ray tracer
5. You know how to do this on the GPU
6. You got a great score
7. You had fun

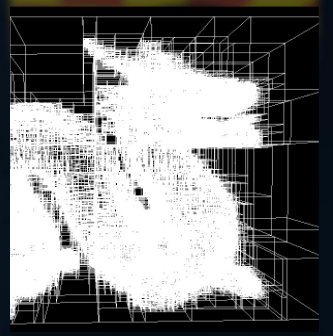
INFOMAGR

Topics

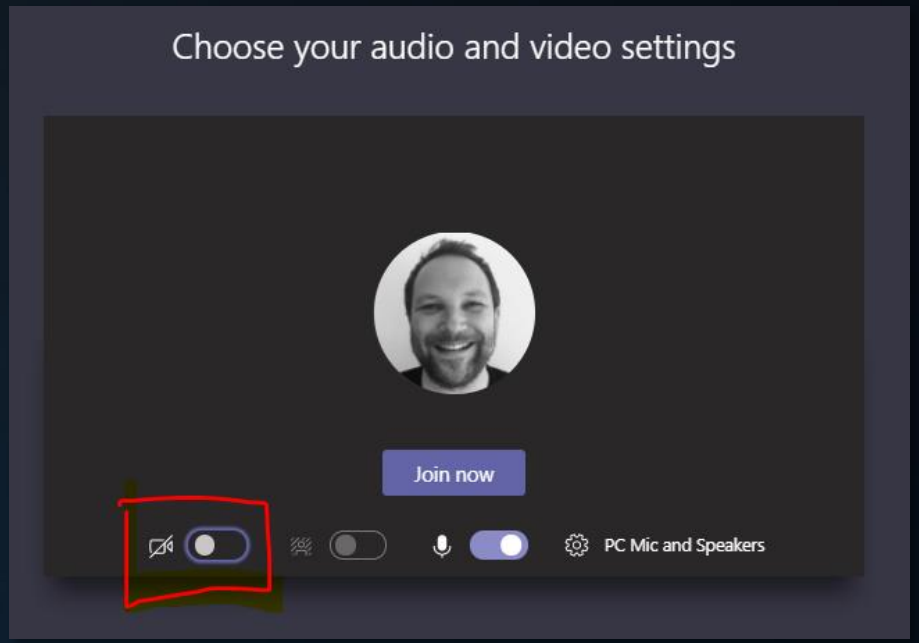
We will cover the following topics:

- Ray tracing fundamentals;
- Whitted-style ray tracing;
- Acceleration structure construction;
- Acceleration structure traversal;
- Data structures and algorithms for animation;
- Stochastic approaches to AA, DOF, soft shadows, … ;
- Path tracing;
- Variance reduction in path tracing algorithms;
- Filtering techniques;
- RTX / DXR (hardware options);
- State-of-the-art in ray tracing for games;
- Various forms of parallelism in ray tracing.



INFOMAGR



Lectures

16 lectures:
Tuesday **10:00 – 11:45**, Thursday 13:15 – 15:00

Working colleges:
Tuesdays 09:00 – 10:00 *(1 hour, before the lecture)*
Thursdays 15:15 – 17:00 *(2 hours, after the lecture)*

All lectures are ON CAMPUS and will be recorded.

Slides will be made available, along with recordings.

Attendance is not mandatory, but of course highly recommended.
We move fast; missing a key lecture may be a serious problem.

INFOMAGR

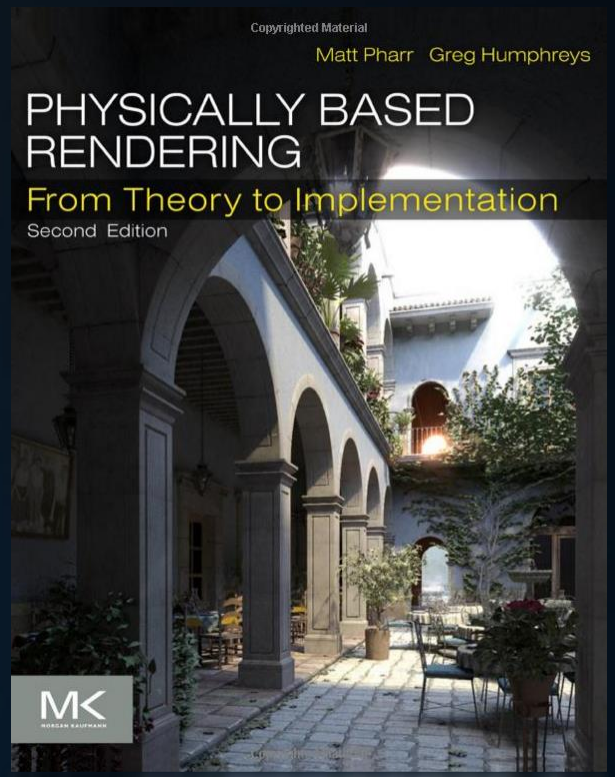
Literature

Papers and online resources will be supplied during the course.

Slides will be made available after each lecture.

Recommended literature:

Physically Based Rendering – From Theory to Implementation, Pharr & Humphreys. ISBN-10: 9780128006450.



The 3rd edition is available for free: www.pbr-book.org

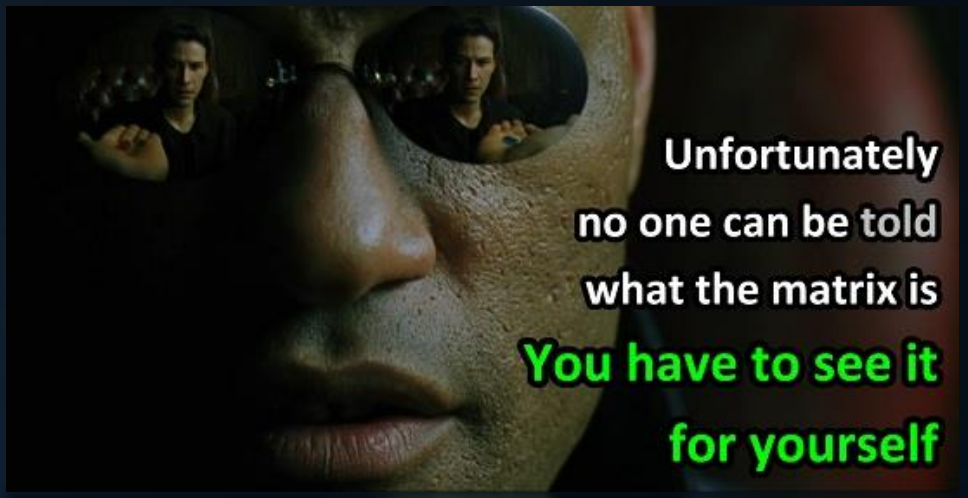
INFOMAGR

Dependencies

It is assumed that you have basic knowledge of rendering (INFOGR) and associated mathematics.

You also should be a decent programmer; this is explicitly not a purely theoretical course. You are expected to verify the theory and experience the good and the bad.

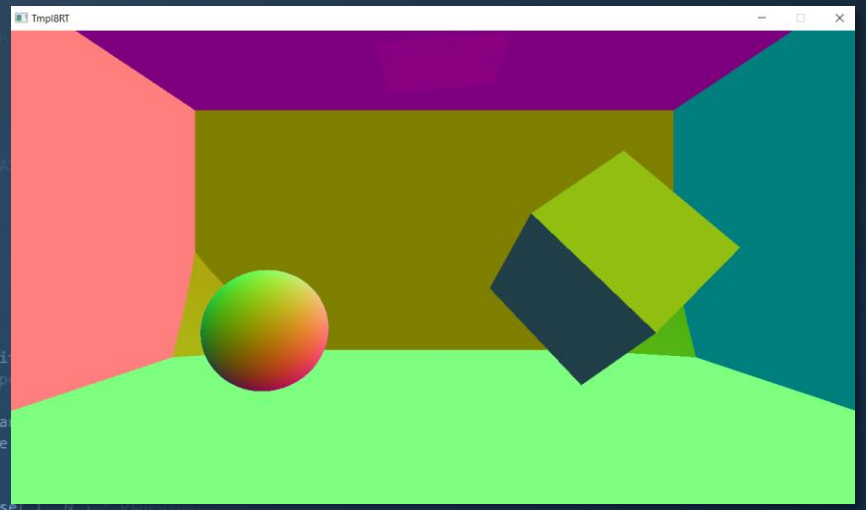
You can code in C/C++ or C# or Rust or basically any other Turing-complete language.



INFOMAGR

Resources

You will develop a ray tracing testbed for assignment 1.
As a starting point, a ‘template’ is available.



However: feel free to use your own framework.

INFOMAGR

Assignments

1. (weight: 1):
   Light transport framework

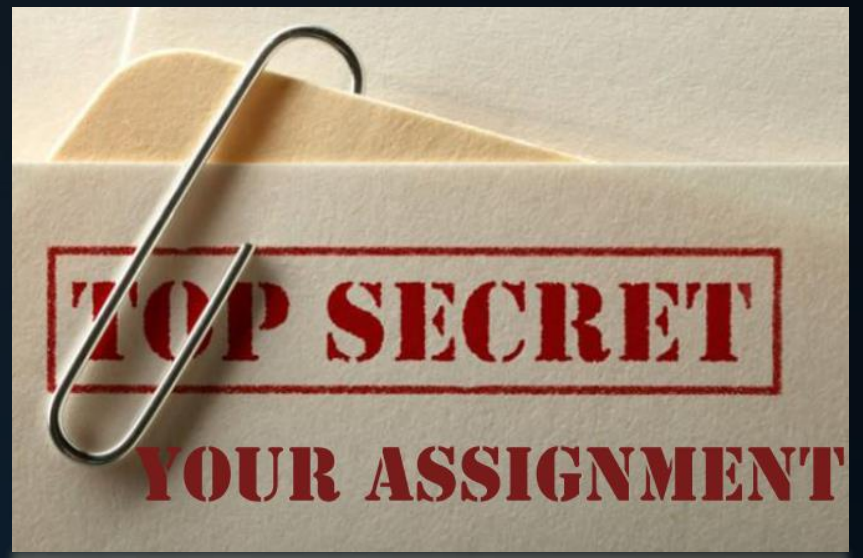
   For this assignment, you prepare a testbed for subsequent assignments.

2. (weight: 1):
   Acceleration structures

   In this assignment, you expand your testbed with efficient acceleration structure construction and traversal. This enables you to run ray tracing in real-time (well…)

3. (weight: 2):
   Final assignment

   In this assignment, you either implement an interactive path tracer, or a rendering algorithm you chose, using CPU and/or GPU rendering.



INFOMAGR

Exam

One final exam at the end of the block.

Materials to study:

- Slides
- Notes taken during the lectures
- Provided literature
- Assignments

INFOMAGR

Grading & Retake

Final grade for assignments $P = (P1 + P2 + 2 * P3) / 4$
Final grade for INFOMAGR $G = (2P + E) / 3$

Passing criteria:

- $P \geq 4.50$
- $E \geq 4.50$
- $G \geq 5.50$

Repairing your grade using the retake exam or retake assignment:

- only if $5.50 > G \geq 4.00$
- you redo P1 or P2 or P3 or E
- this replaces the original P1, P2, P3 or E grade.

Today’s Agenda:

- Advanced Graphics
- Recap: Ray Tracing
- Assignment 1

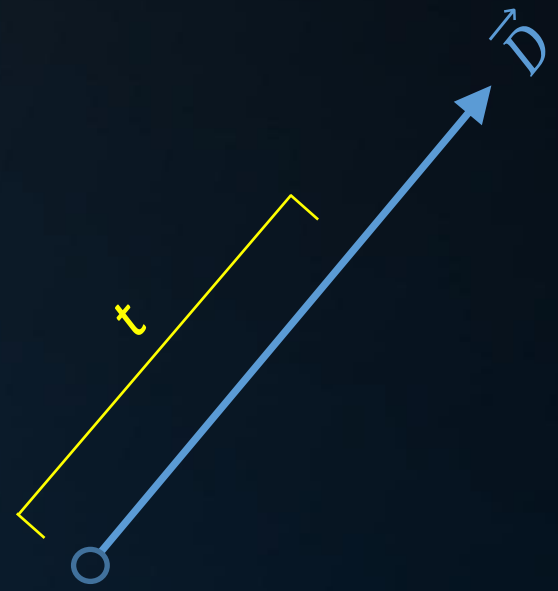
Recap

Ray

A ray is an infinite line with a start point:

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

The ray direction $\vec{D}$ is usually normalized:
this way, t becomes a distance along the ray.



Recap

Scene

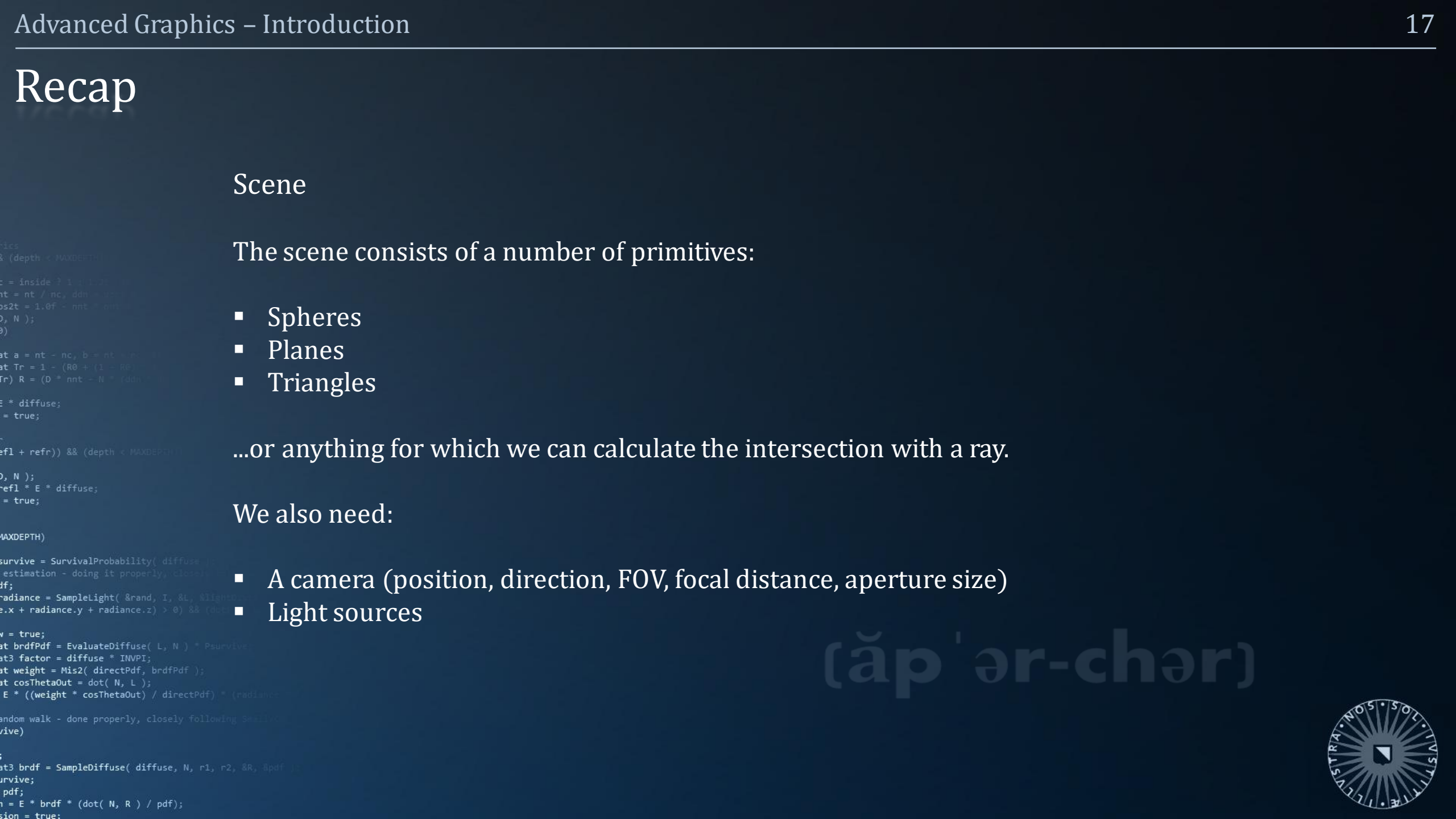
The scene consists of a number of primitives:

- Spheres
- Planes
- Triangles

...or anything for which we can calculate the intersection with a ray.

We also need:

- A camera (position, direction, FOV, focal distance, aperture size)
- Light sources



Recap

Ray Tracing

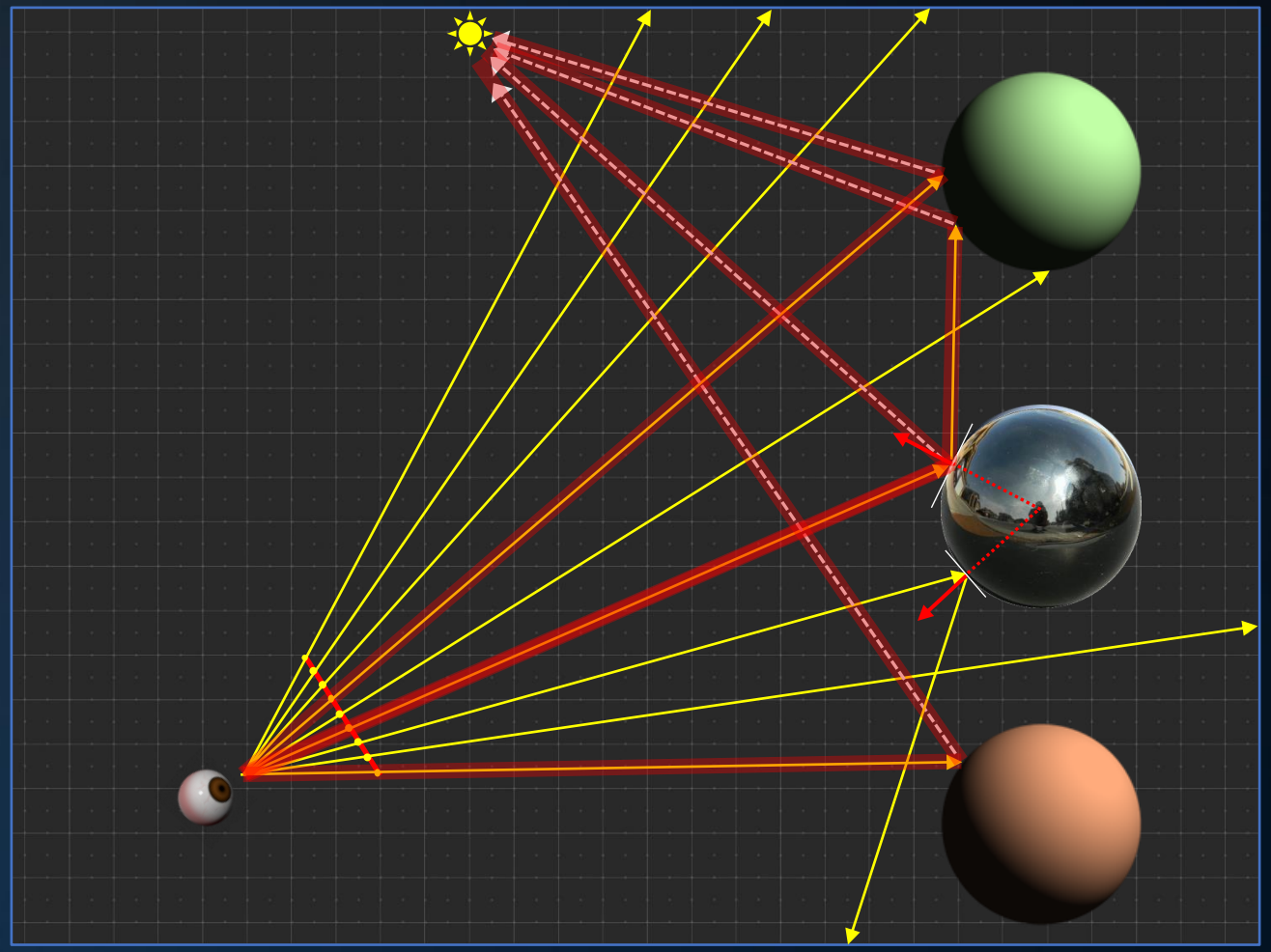
World space

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

Light transport

- Extension rays

Light transport



Recap

Ray setup

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

Camera position: $E = (0,0,0)$
View direction: $\vec{V} = (0,0,1)$
Screen center: $C = E + d\vec{V}$
Screen corners: $P_0 = C + (-1,-1,0), \quad P_1 = C + (1,-1,0), \quad P_2 = C + (-1,1,0)$

From here:

- Change FOV by altering d;
- Transform camera by multiplying E, P_0, P_1, P_2 with a camera matrix.

Recap

Ray setup

Point on the screen:

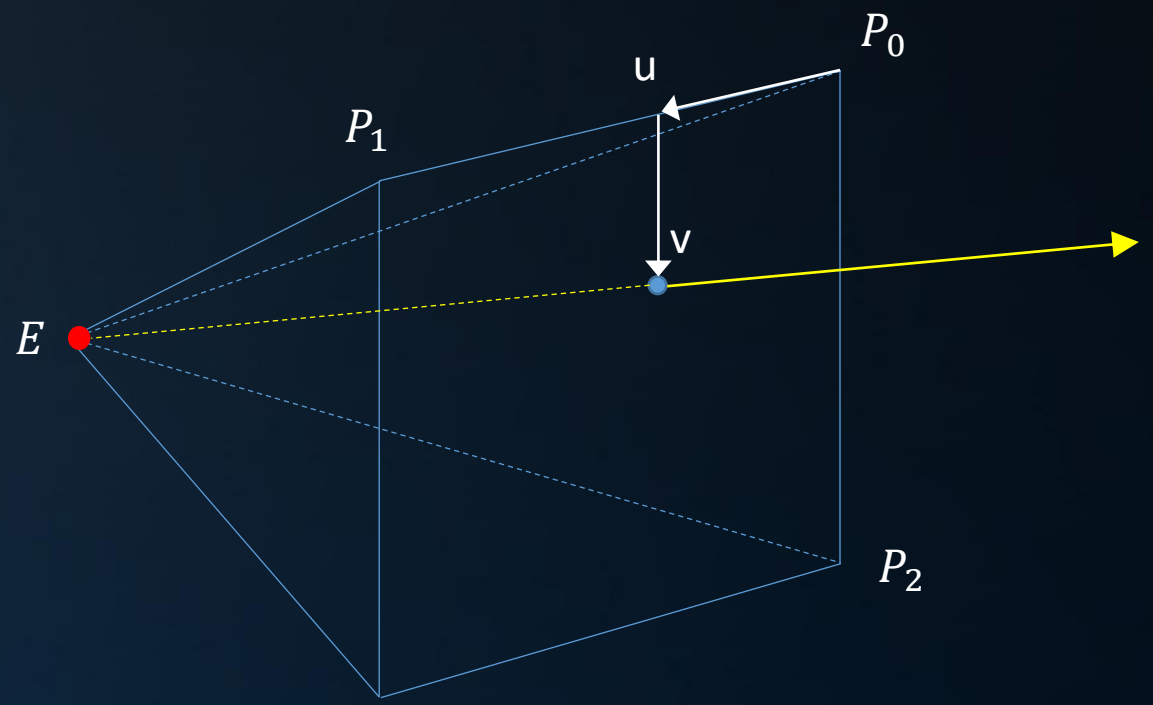
$$P(u,v) = P_0 + u(P_1 - P_0) + v(P_2 - P_0)$$
$$u, v \in [0,1]$$

Ray direction (normalized):

$$\vec{D} = \frac{P(u,v) - E}{\| P(u,v) - E \|}$$

Ray origin:

$$O = E$$

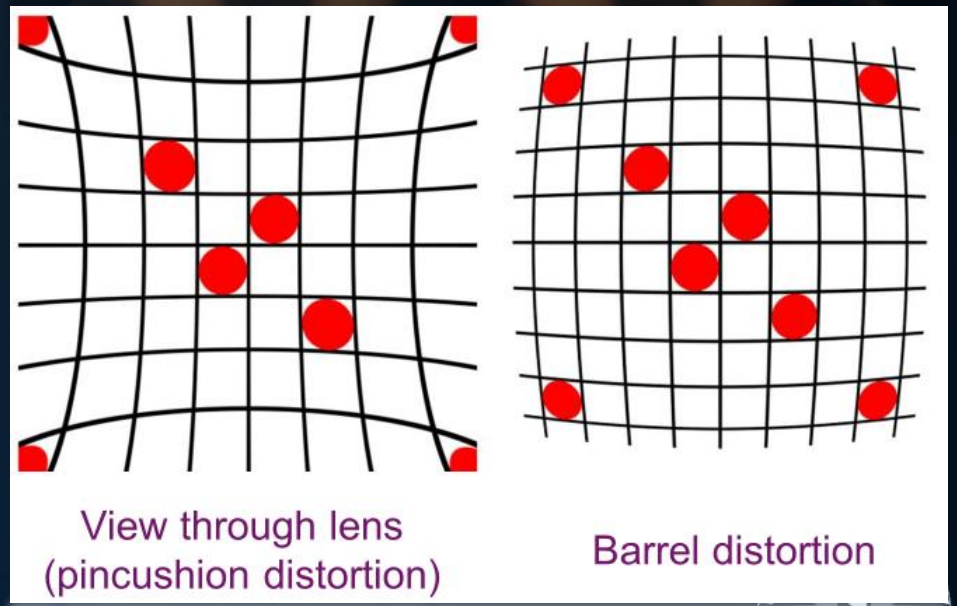


Recap

Ray setup

Alternatives:

- Taking into account HMD lens distortion



Recap

Ray setup

Alternatives:

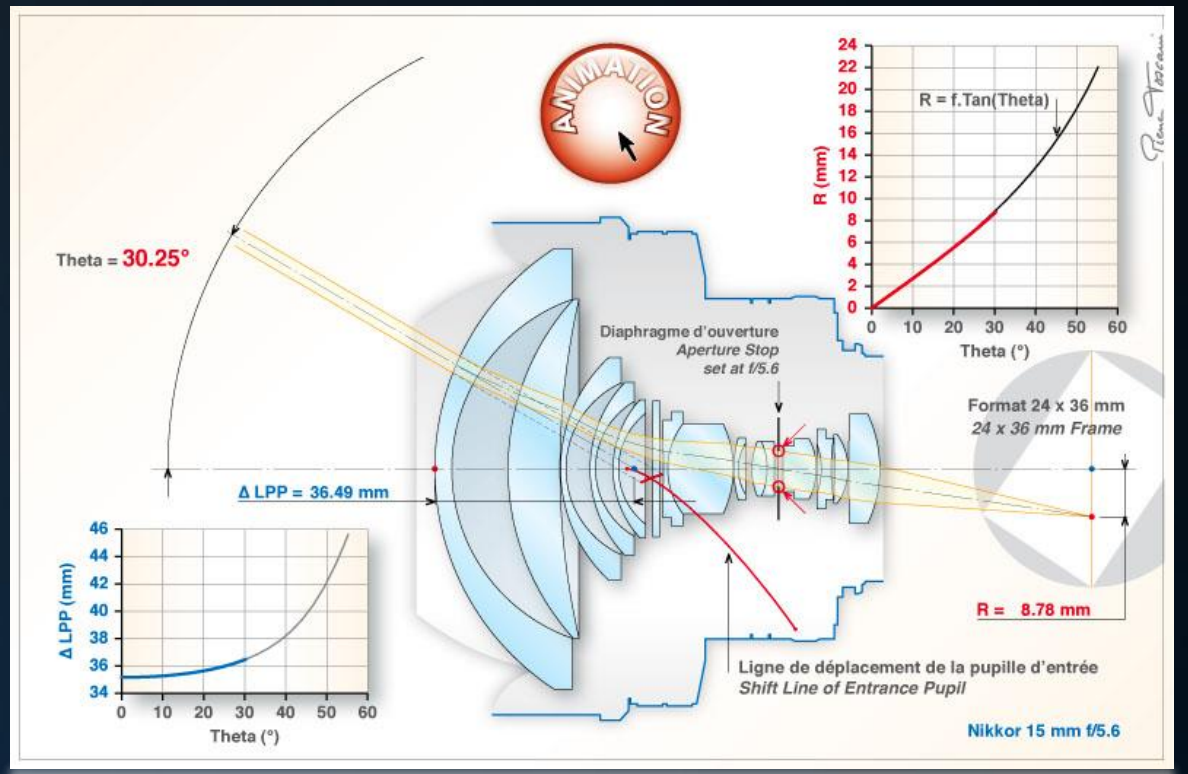
- Taking into account HMD lens distortion
- Fisheye lens

Recap

Ray setup

Alternatives:

- Taking into account HMD lens distortion
- Fisheye lens
- Complex lens system



Recap

Ray Intersection

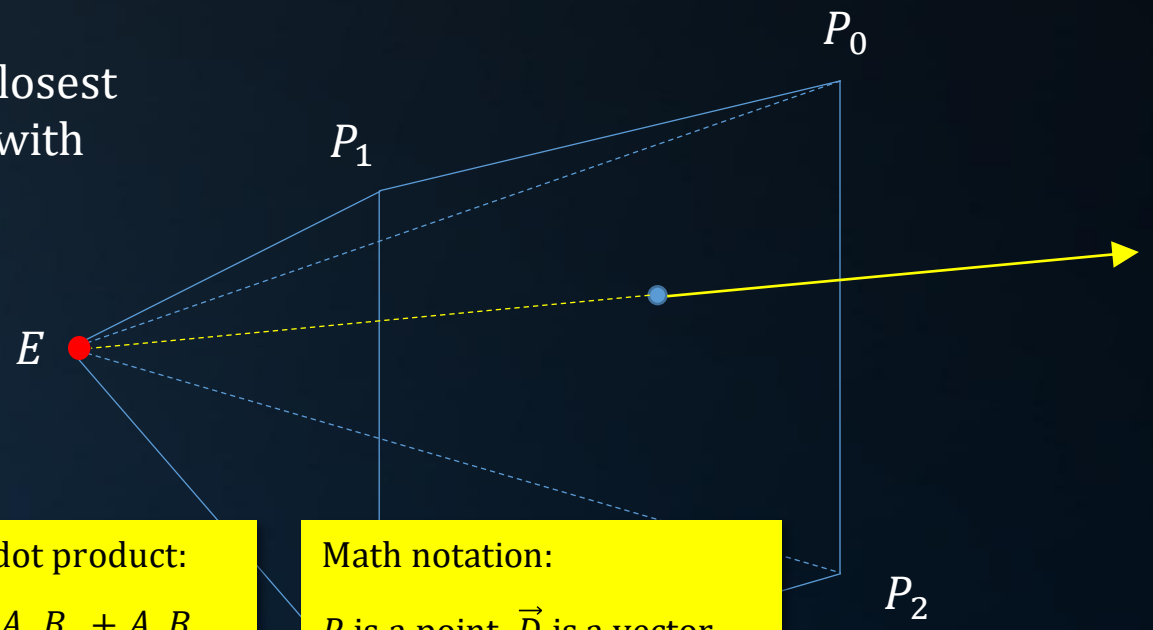
Given a ray $P(t) = O + t\vec{D}$, we determine the closest intersection distance t by intersecting the ray with each of the primitives in the scene.

Ray / plane intersection:

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

Substituting for $P(t)$, we get

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



Math reminder, dot product:

$A \cdot B = A_x B_x + A_y B_y + A_z B_z$
$A \cdot B = \cos\theta$
$A \cdot B$ is: the *length* of the *projection* of A on B.
$A \cdot B$ is a *scalar.*

Math notation:

P is a point, $\vec{D}$ is a vector
t is a scalar.

Recap

Ray Intersection

Ray / sphere intersection:

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

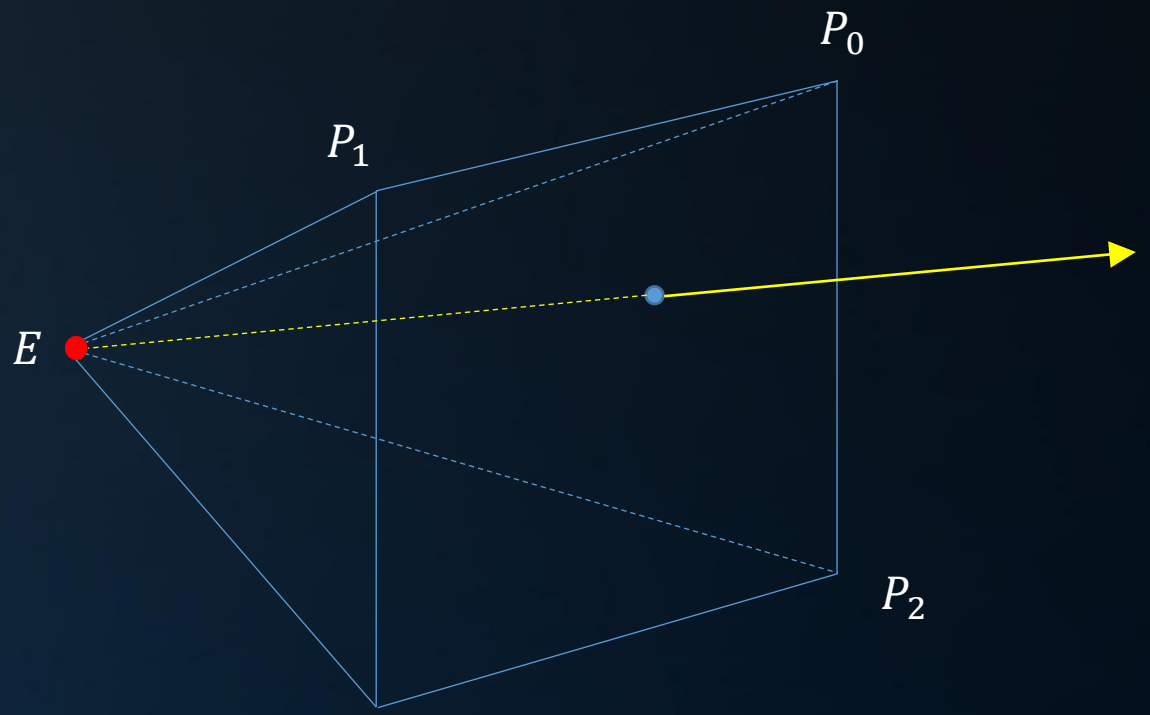
Substituting for $P(t)$, we get

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

$$at^2 + bt + c = 0 \rightarrow t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

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

Negative:
no intersections



Recap

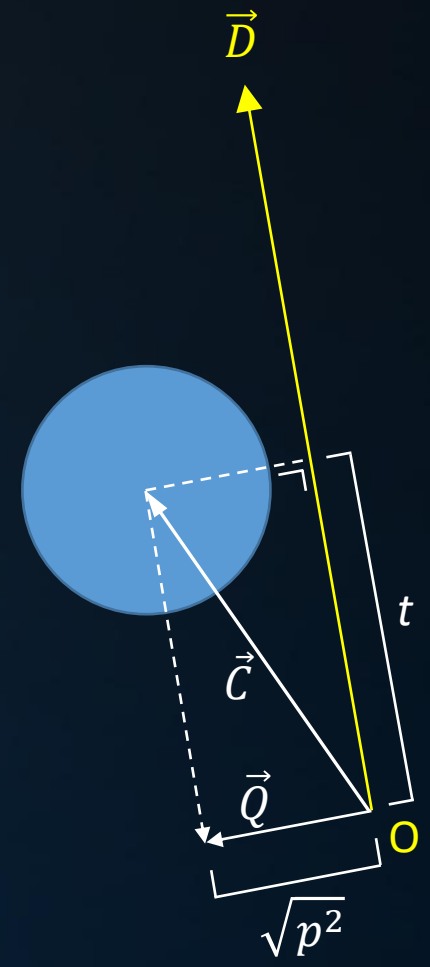
Ray Intersection

Efficient ray / sphere intersection:

```
void Sphere::IntersectSphere( Ray ray )
{
    vec3 C = this.pos - ray.O;
    float t = dot( C, ray.D );
    vec3 Q = C - t * ray.D;
    float p2 = dot( Q, Q );
    if (p2 > sphere.r2) return; // r2 = r * r
    t -= sqrt( sphere.r2 - p2 );
    if ((t < ray.t) && (t > 0)) ray.t = t;
}
```

Note:

This only works for rays that start outside the sphere.



Recap

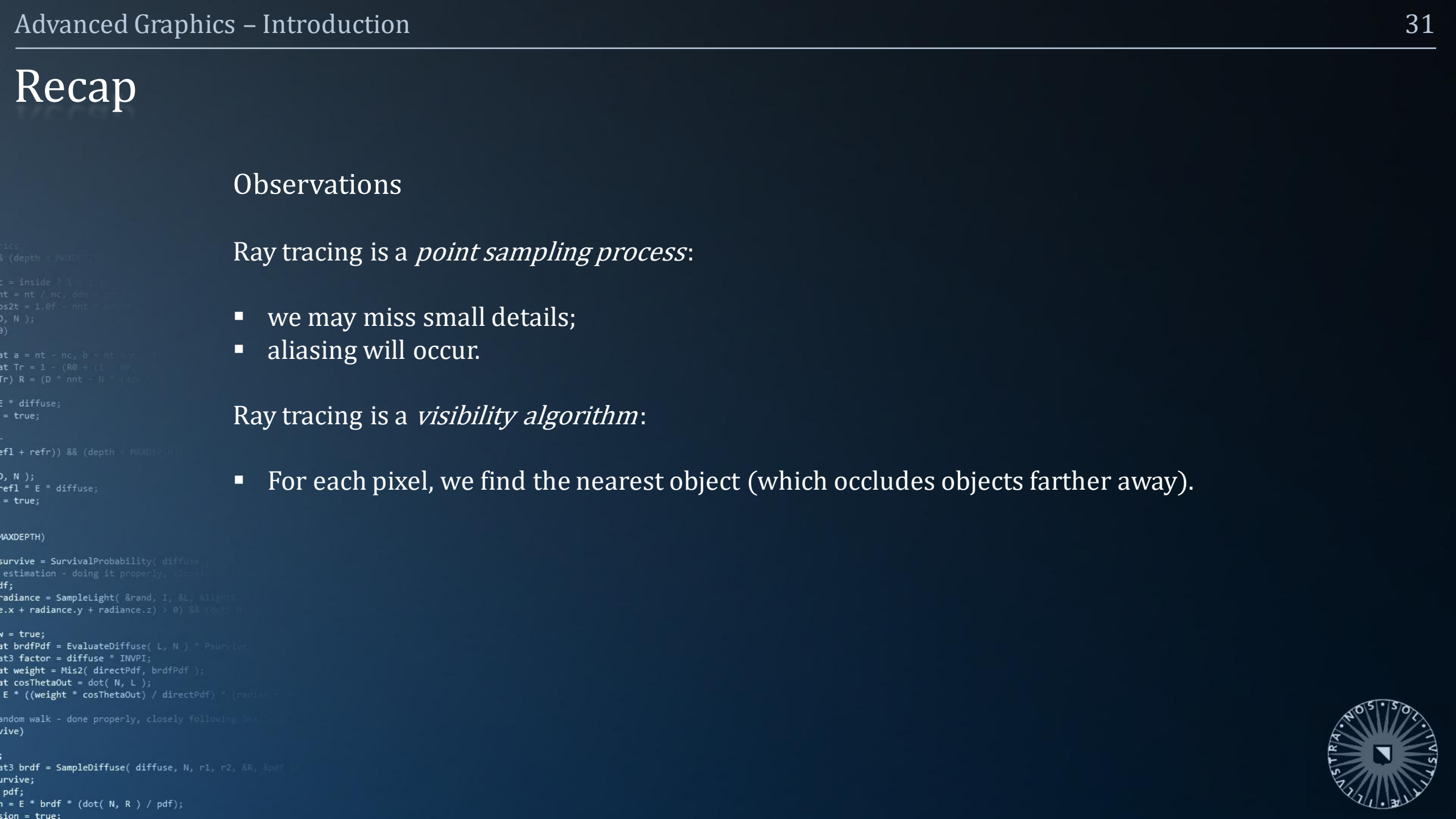
Observations

Ray tracing is a *point sampling process*:

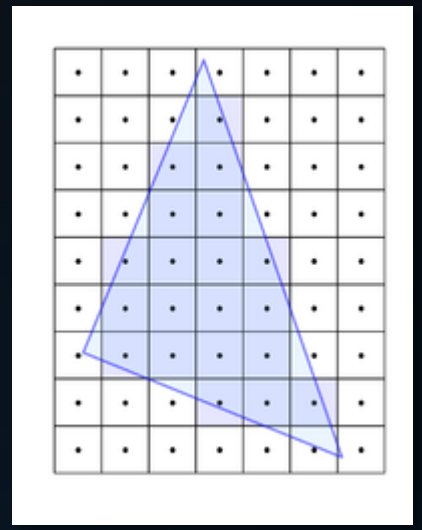
- we may miss small details;
- aliasing will occur.

Ray tracing is a *visibility algorithm*:

- For each pixel, we find the nearest object (which occludes objects farther away).



Recap



Observations

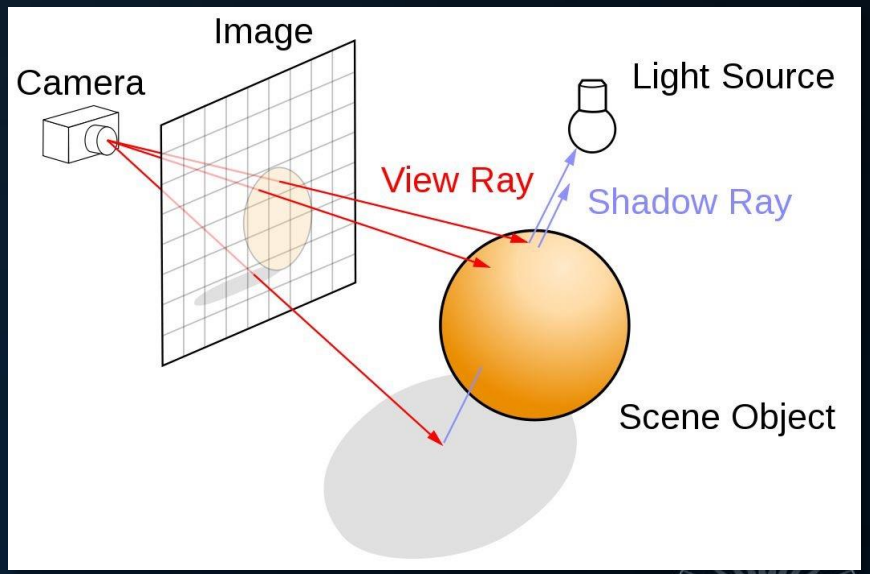
Note: rasterization (Painter's or z-buffer) is also a visibility algorithm.

Rasterization:

- loop over objects / primitives;
- per primitive: loop over pixels.

Ray tracing:

- loop over pixels;
- per pixel: loop over objects / primitives.



Today’s Agenda:

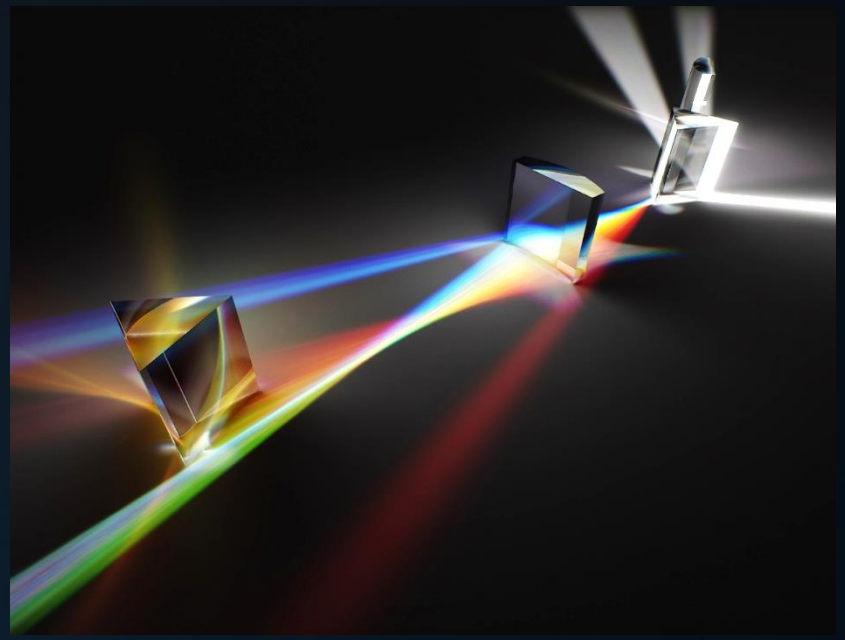
- Advanced Graphics
- Recap: Ray Tracing
- Assignment 1

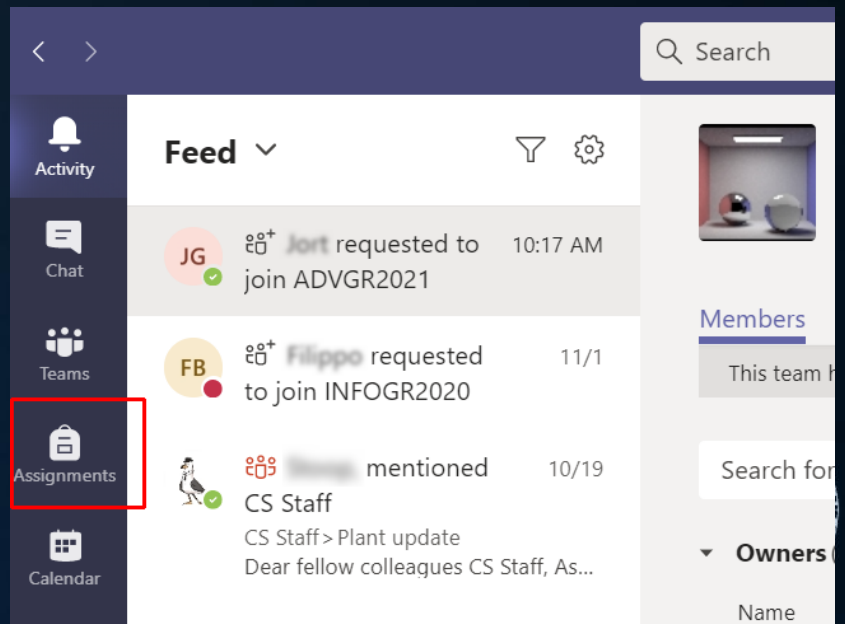
Assignment 1

Ray Tracing Testbed

Basic functionality of a ray tracer:

1. Calculate a color for each pixel on the screen
2. Do so using rays:
   - Start a ray at the camera location
   - Figure out where the pixel is in world space
   - Extend the (normalized) ray through the pixel
   - Find the nearest intersection (try them all)
   - Do fancy things at the nearest intersection.





Assignment 1

Ray Tracing Testbed

Implement an experimentation framework for ray tracing. Ingredients:

Scene

- Primitives: spheres, planes, triangles
- I/O (e.g., obj loader)
- Intersection
- Materials: diffuse color, diffuse / specular / dielectric, absorption

Camera

- Position, target, FOV
- Ray generation

Ray

Renderer

- Whitted-style

User interface

- Input handling
- Presentation

Assignment 1

Ray Tracing Testbed

Regarding the file loading requirement:

- You may want to start with the included hardcoded scene
- Don't build your own OBJ loader, that's a waste of time
- Use assimp or tinyobjloader (C++) or find a lib if you're using C#

  http://www.stefangordon.com/parsing-wavefront-obj-files-in-c/
  http://www.rexcardan.com/2014/10/read-obj-file-in-c-in-just-10-lines-of-code/
  https://github.com/ChrisJansson/ObjLoader

- Start with small scenes; minimize your development cycle.

Assignment 1

Ray Tracing Testbed

Intersecting triangles:

An easy to implement and quite efficient algorithm is:

Fast, Minimum Storage Ray/Triangle Intersection, Möller & Trumbore, 1997.

…which is explained in elaborate detail by scratchapixel.com:

http://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-rendering-a-triangle/moller-trumbore-ray-triangle-intersection

Today’s Agenda:

- Advanced Graphics
- Recap: Ray Tracing
- Assignment 1

INFOMAGR – Advanced Graphics

Jacco Bikker - November 2022 – February 2023

END of “Introduction”

next lecture: “Whitted-style Ray Tracing”