INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2019

Lecture 5: “Graphics Fundamentals”

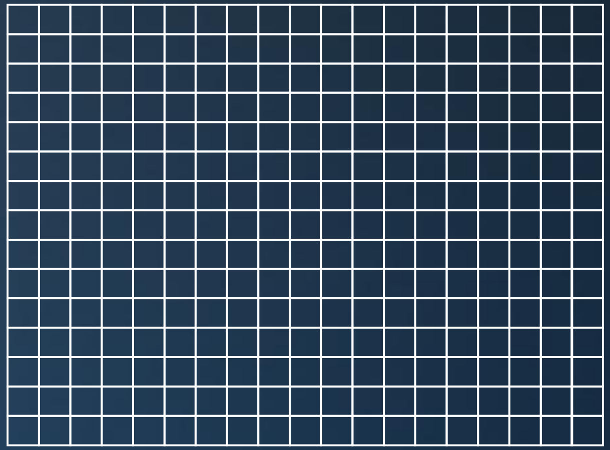
Welcome!

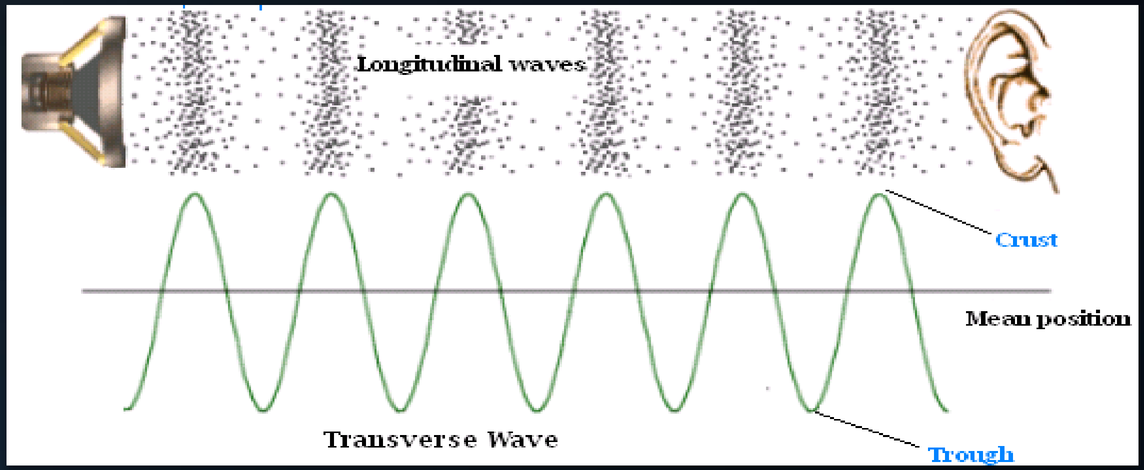
Today's Agenda:

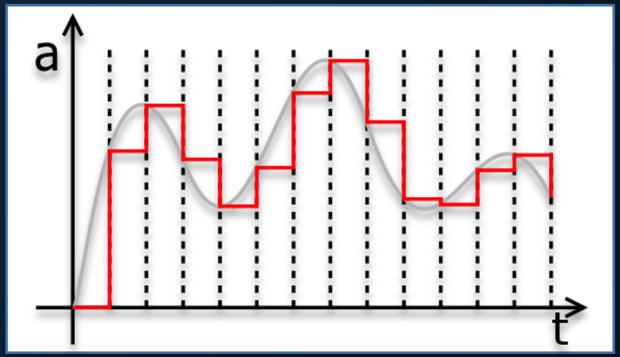
- Rasters
- Colors
- Ray Tracing
- Assignment P2

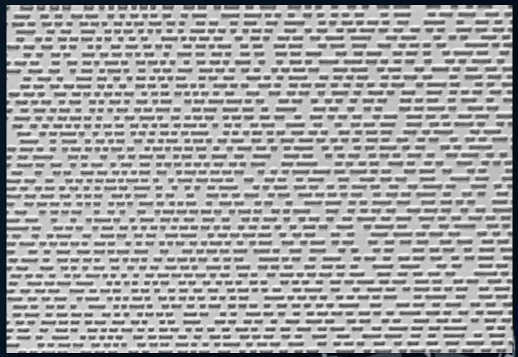
Raster Displays

Discretization









Raster Displays

Discretization

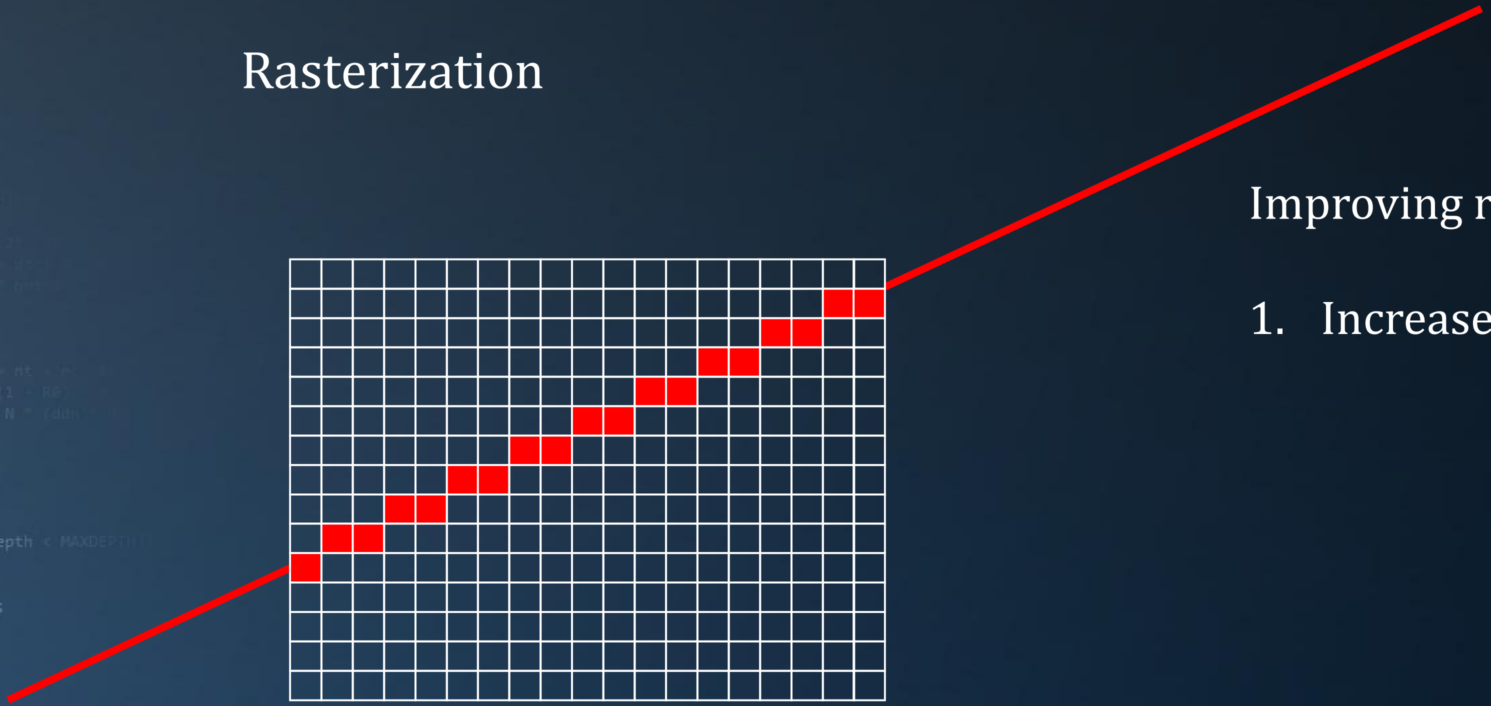
Rasterization:

“Converting a vector image into a raster image for output on a video display or printer or storage in a bitmap file format.”

(Wikipedia)

Raster Displays

Rasterization

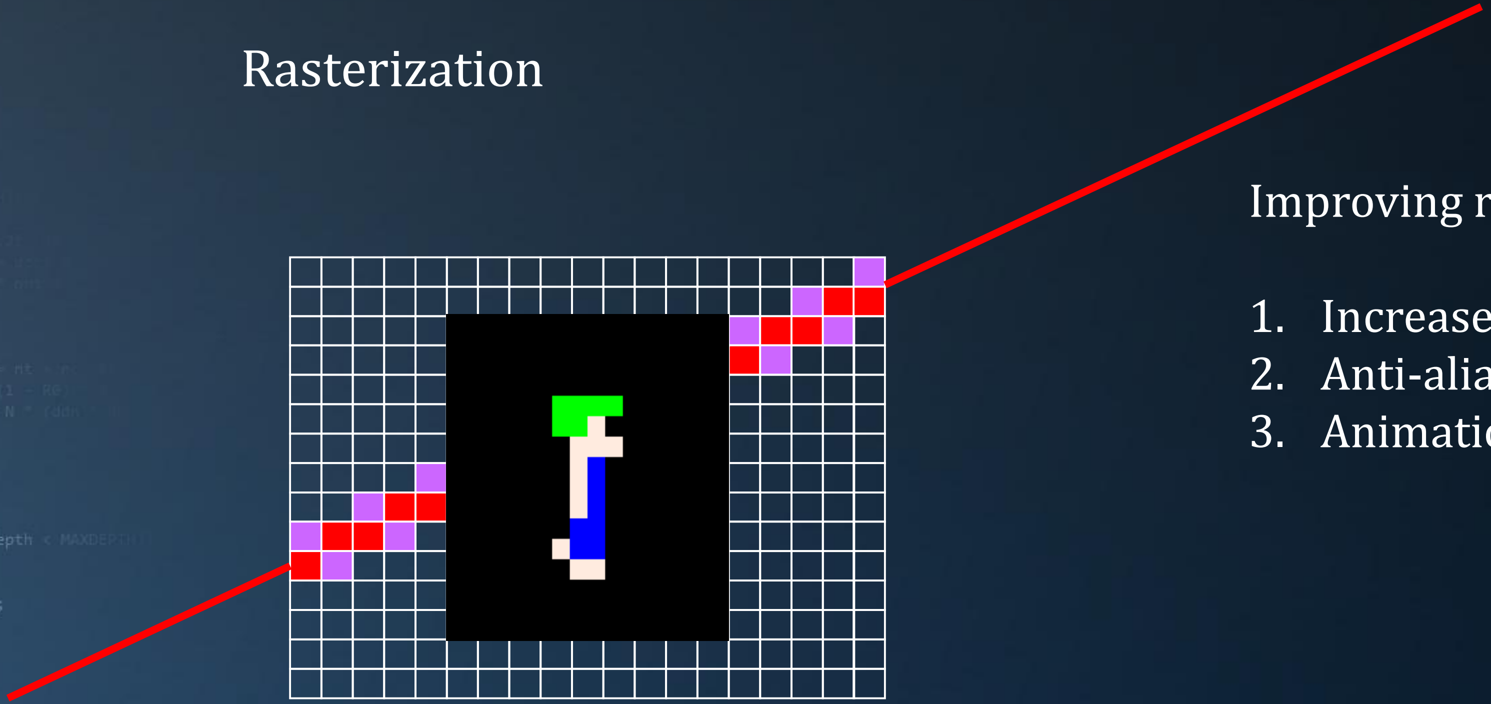


Improving rasterization:

1. Increase resolution;

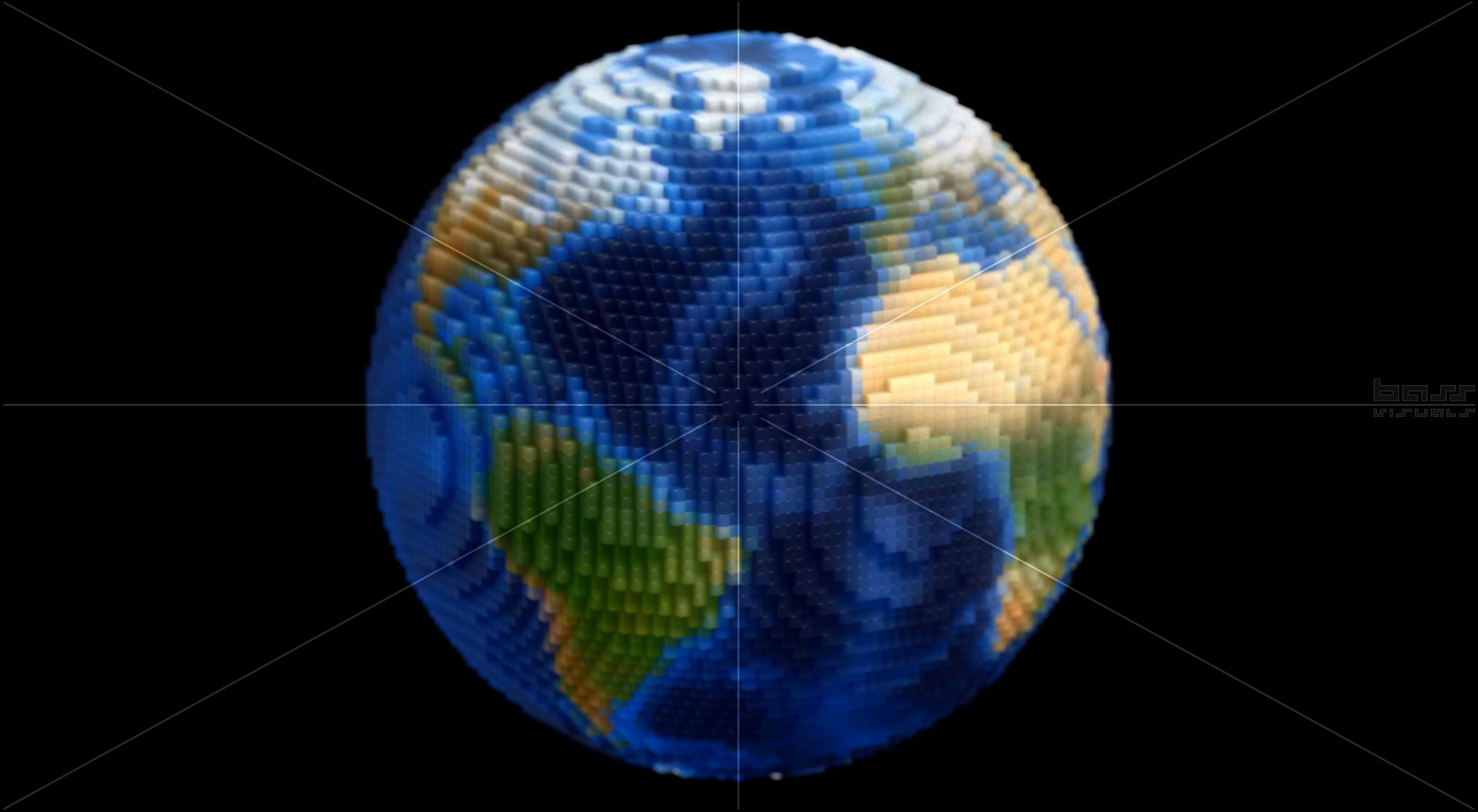
Raster Displays

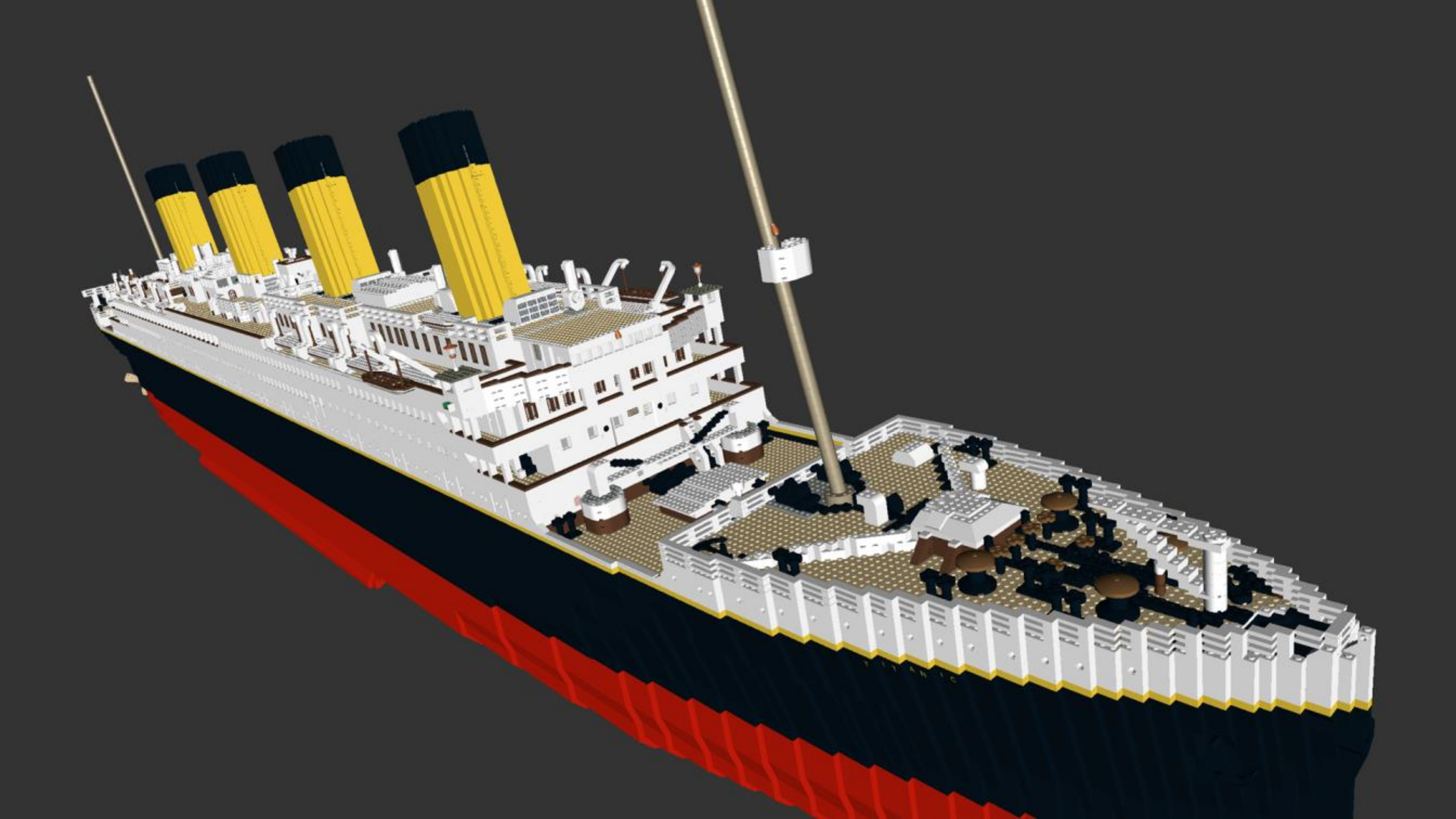
Rasterization



Improving rasterization:

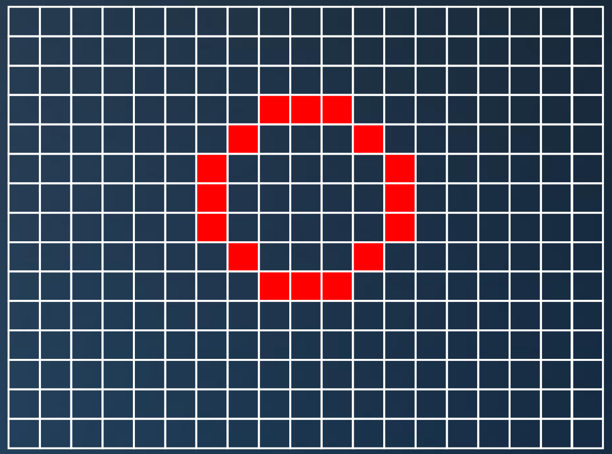
1. Increase resolution;
2. Anti-aliasing;
3. Animation.





Raster Displays

Discretization



$$\pi = 4$$

$$\sout{a^2 + b^2 = c^2}$$

$$a^1 + b^1 = c^1$$

Raster Displays

CRT – Cathode Ray Tube

Physical implementation – origins

Electron beam zig-zagging over a fluorescent screen.

Raster Displays

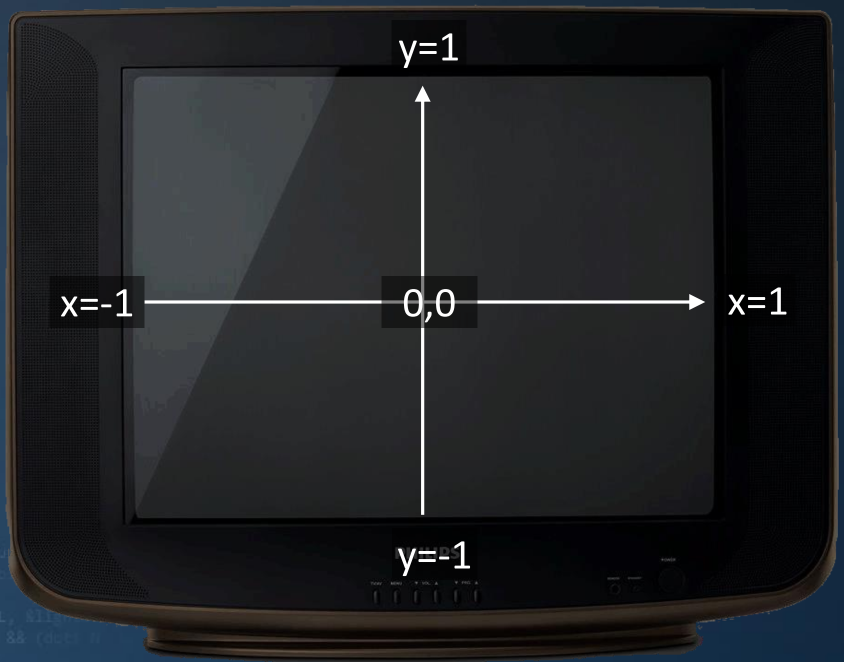
CRT – Cathode Ray Tube

Physical implementation – consequences

- Origin in the top-left corner of the screen
- Axis system directly related to pixel count

Raster Displays

CRT – Cathode Ray Tube



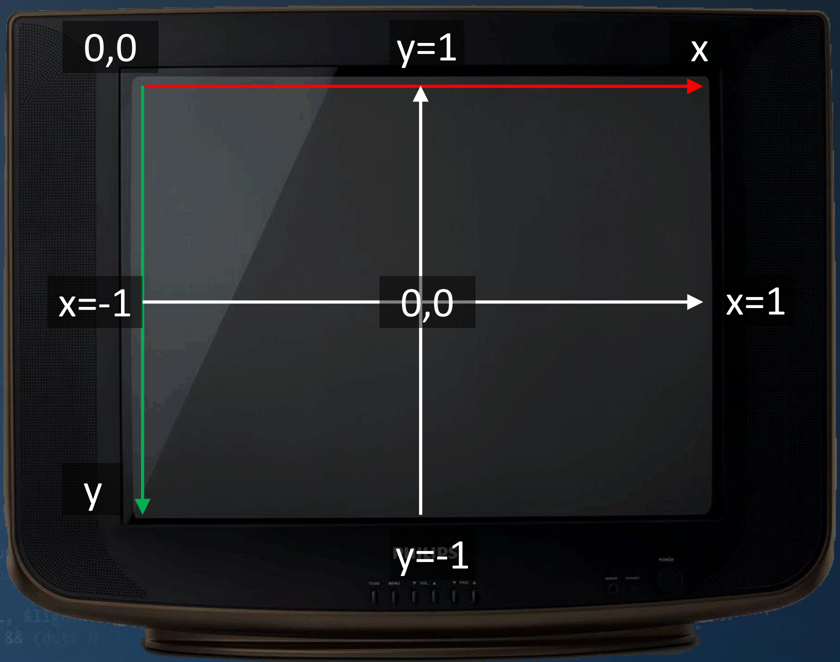
Physical implementation – consequences

- Origin in the top-left corner of the screen
- Axis system directly related to pixel count

Not the coordinate system we expected…

Raster Displays

CRT – Cathode Ray Tube



Proper screen coordinates

- Pixel coordinates are only relevant for the final step: plotting pixels
- Decouple the 2D screen coordinates in your game / app from the physical mapping.

Raster Displays

Frame rate

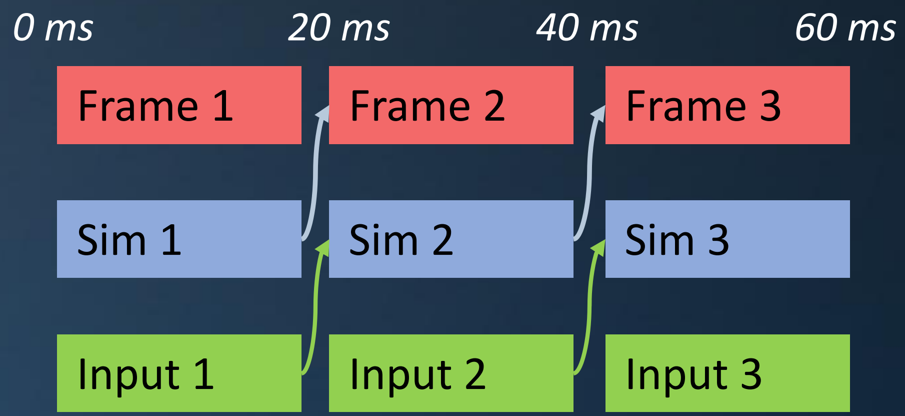
PAL: 25fps
NTSC: 30fps (actually: 29.97)
Typical laptop screen: 60Hz
High-end monitors: 120-240Hz
Cartoons: 12-15fps

Human eye:
‘Frame-less’
Not a raster.

How many fps / megapixels is ‘enough’?

Raster Displays

Frame rate



Even 100 frames per second may result in a noticeable delay of 30ms.

A very high frame rate minimizes the response time of the simulation.

Raster Displays

Generating images

Rendering:

on a raster

"The process of generating an image from a 2D or 3D model by means of a computer program."
(Wikipedia)

Two main methods:

1. Ray tracing: for each pixel: what color do we assign to it?
2. Rasterization: for each triangle, which pixels does it affect?

Today's Agenda:

- Rasters
- Colors
- Ray Tracing
- Assignment P2

Colors

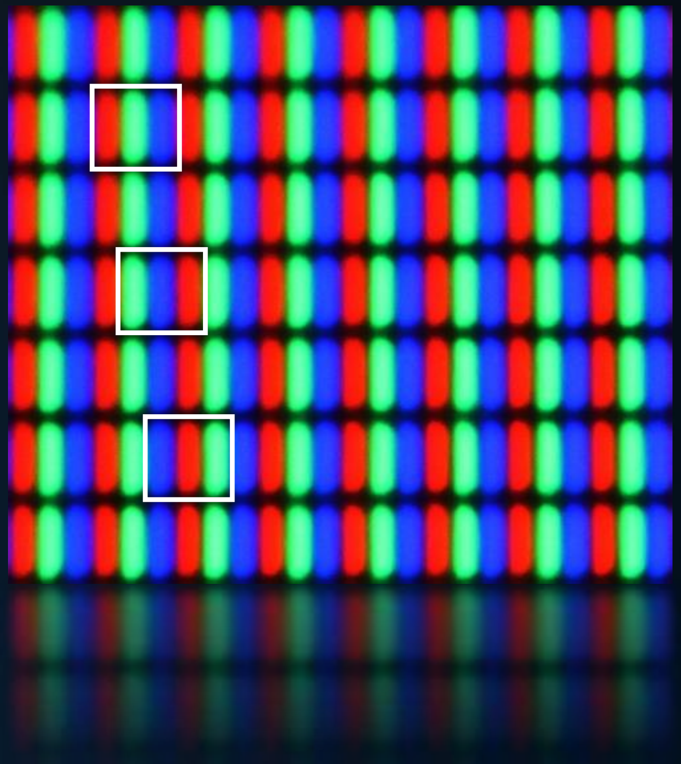
Color representation

Computer screens emit light in three colors: red, green and blue.

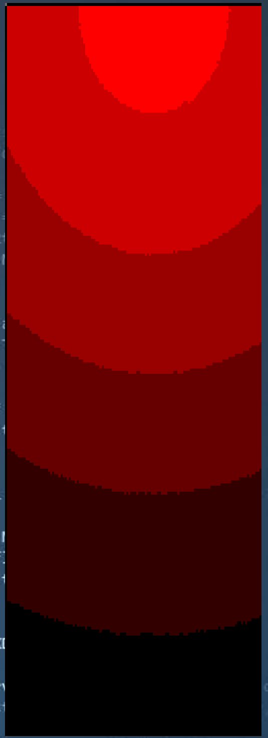
By additively mixing these, we can produce most colors: from black (red, green and blue turned off) to white (red, green and blue at full brightness).

In computer graphics, colors are stored in discrete form. This has implications for:

- Color resolution (i.e., number of unique values per component);
- Maximum brightness (i.e., range of component values).



Colors

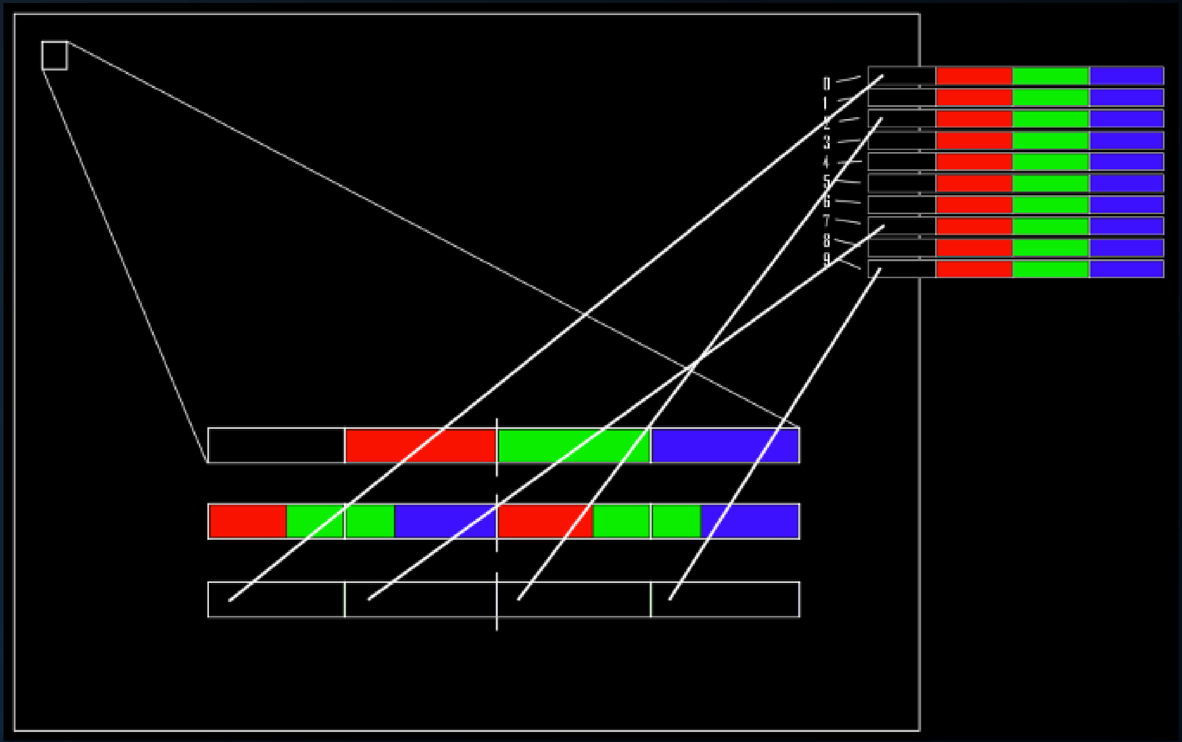


Color representation

The most common color representation is 32-bit ARGB, which stores red, green and blue as 8 bit values (0..255).

Alternatively, we can use 16 bit for one pixel (RGB 565),

or a color palette. In that case, one byte is used per pixel, but only 256 unique colors can be used for the image.



Colors

Color representation

Colors

Color representation

Colors

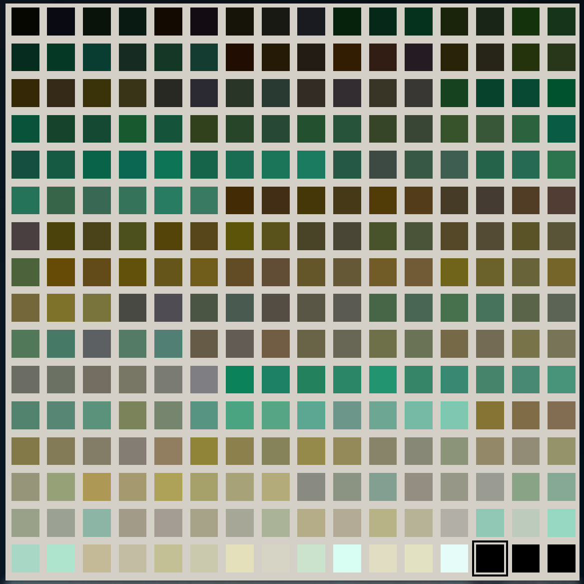
Color representation

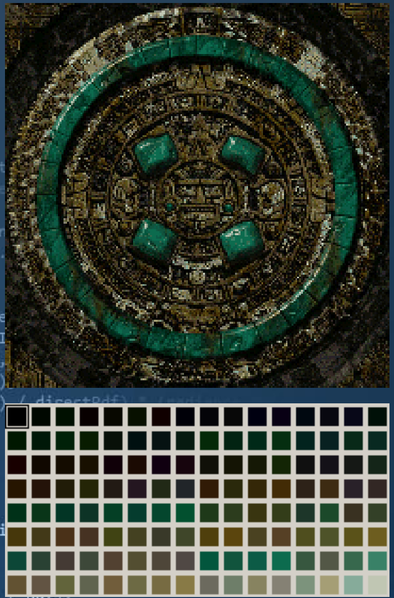
Colors

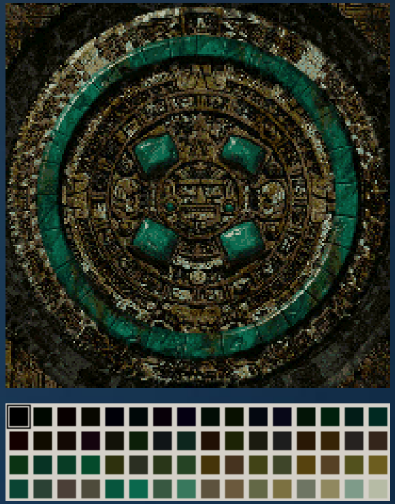
Color representation

Textures can typically safely be stored as palletized images.

Using a smaller palette will result in smaller compressed files.







Colors

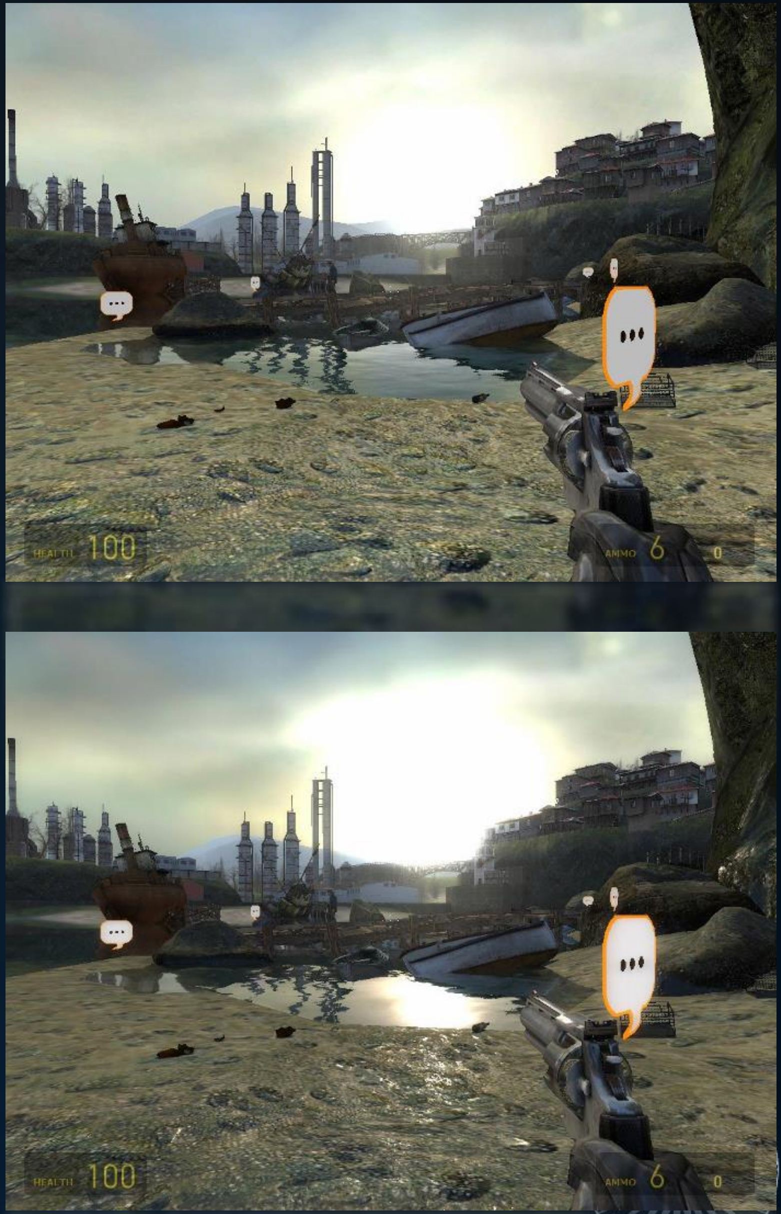
Color representation

Using a fixed range (0:0:0 ... 255:255:255) places a cap on the maximum brightness that can be represented:

- A white sheet of paper: (255,255,255)
- A bright sky: (255,255,255)

The difference becomes apparent when we look at the sky and the sheet of paper through sunglasses.

(or, when the sky is reflected in murky water)



Colors

Color representation

For realistic rendering, it is important to use an internal color representation with a much greater range than 0..255 per color component.

HDR: High Dynamic Range;

We store one float value per color component.

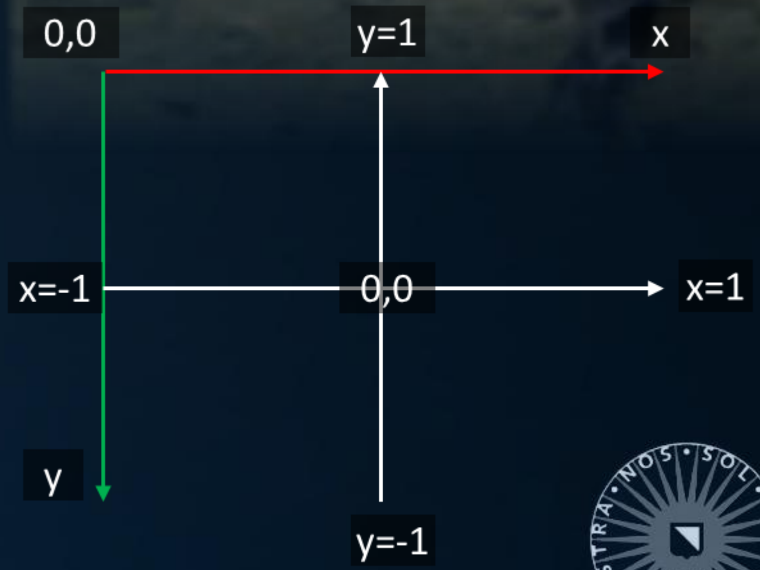
Including alpha, this requires 128bit per pixel.

Colors

Color representation

Like pixel coordinates, pixel colors on the physical screen are only useful for final pixel plotting:

Do not use integer colors clamped to [0..255] internally, unless you have a good reason for this.



Today's Agenda:

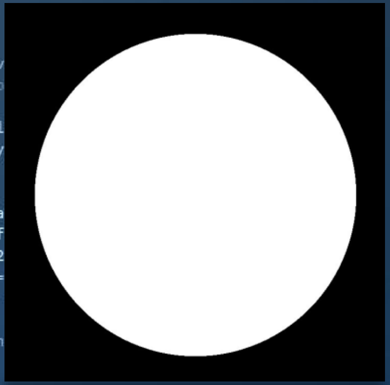
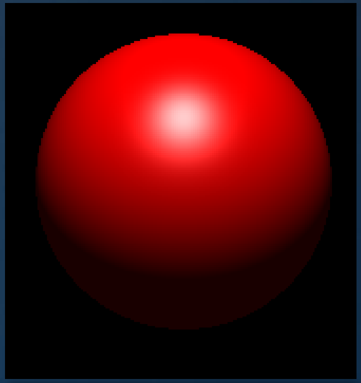
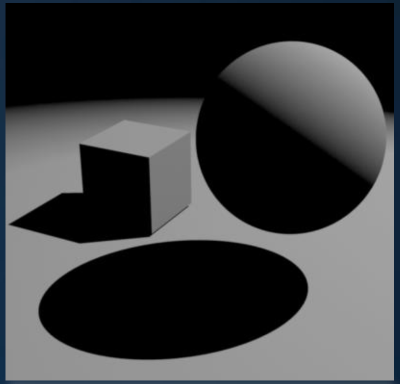
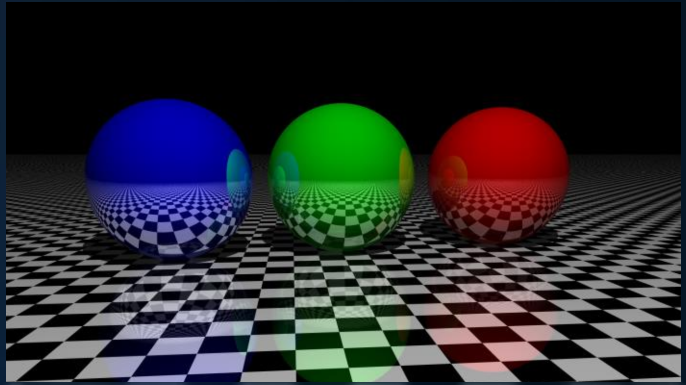
- Rasters
- Colors
- Ray Tracing
- Assignment P2

Ray Tracing

PART 1: Introduction & shading (today, Thursday)

PART 2: Reflections, refraction, absorption (next week)

PART 3: Path Tracing (later)

Ray Tracing

Ray Tracing:

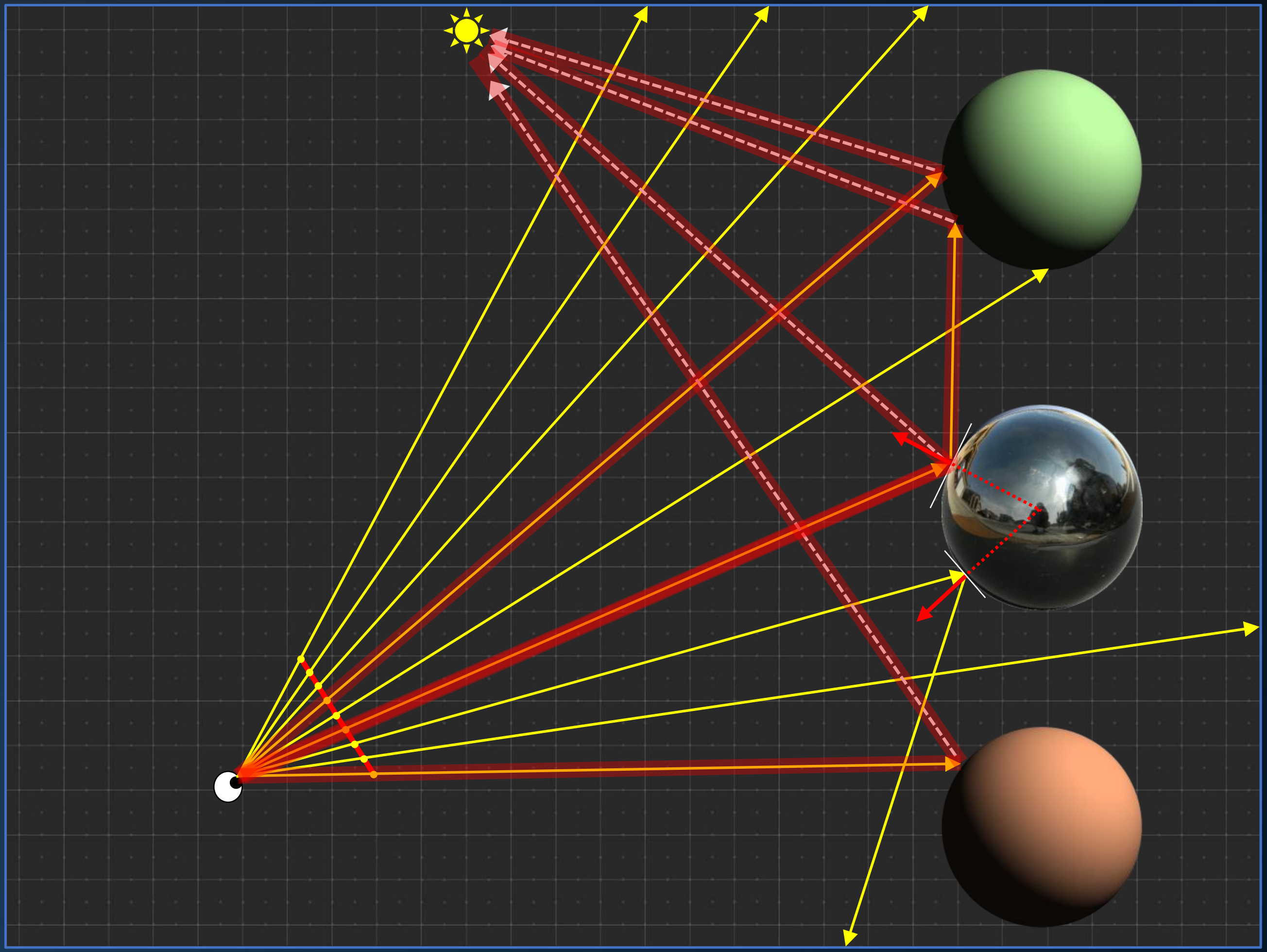
World space

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

Light transport

- Extension rays

Light transport



Ray Tracing

Ray Tracing:

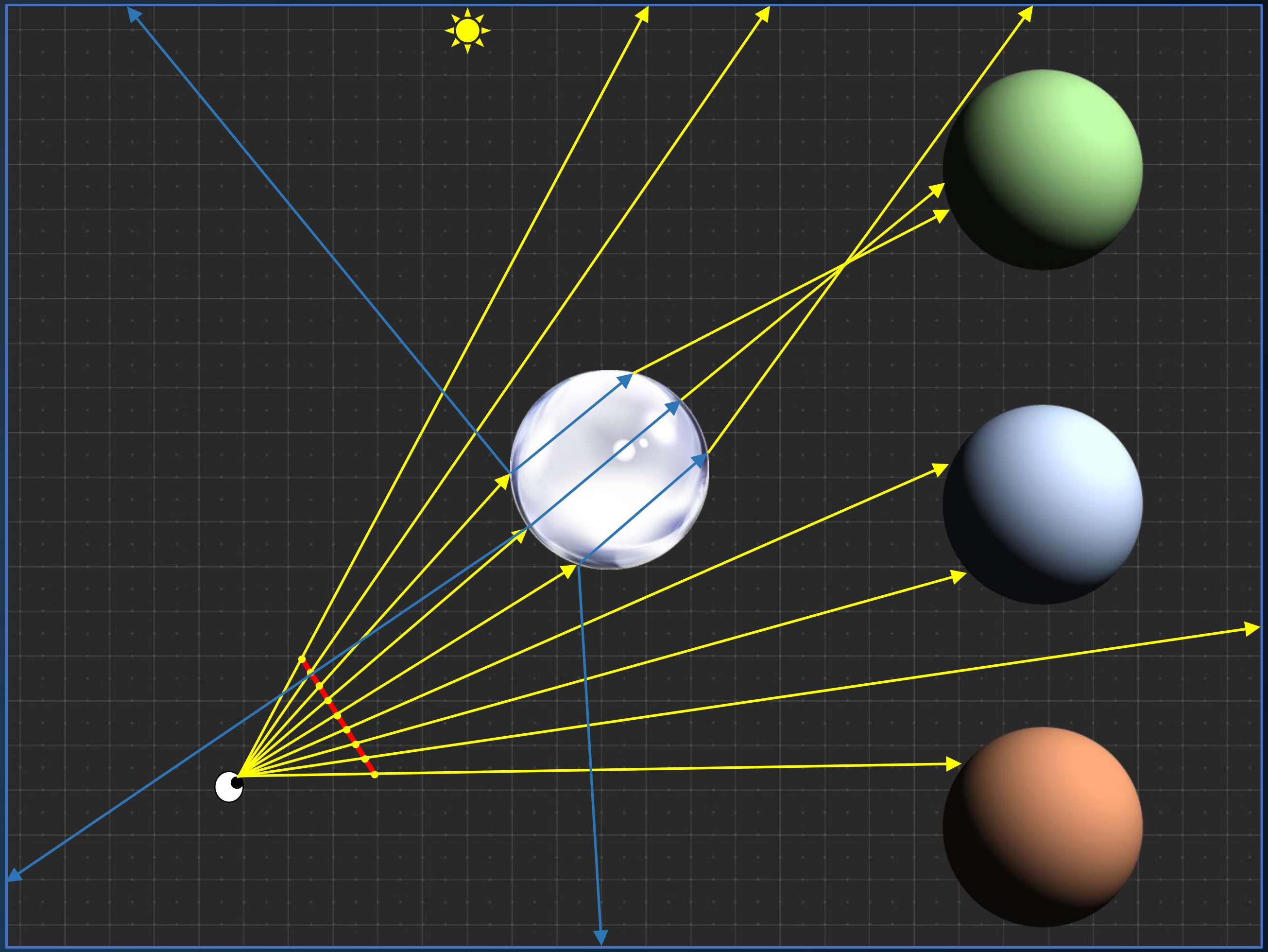
World space

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

Light transport

- Extension rays

Light transport



Ray Tracing

Ray Tracing:

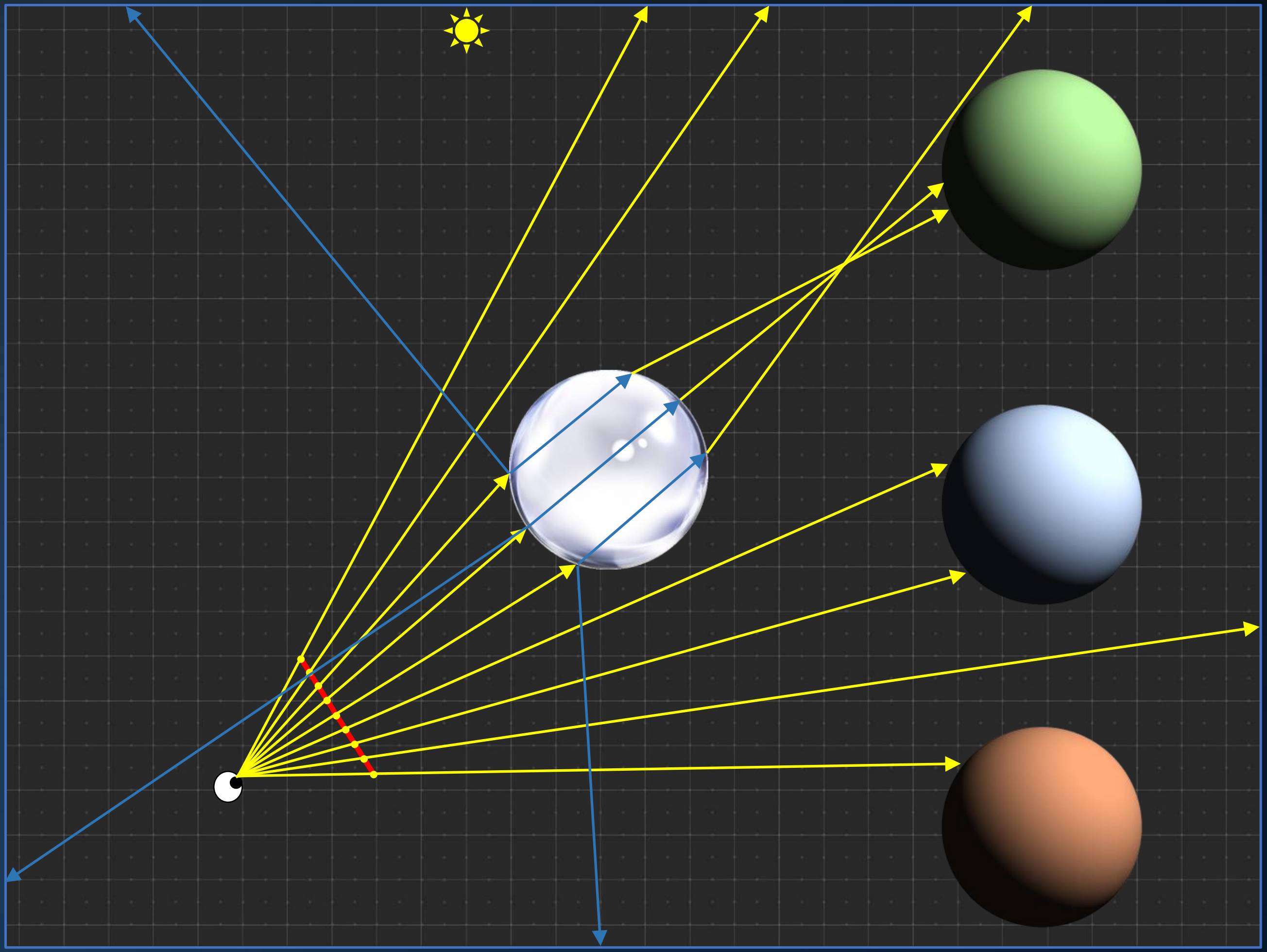
World space

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

Light transport

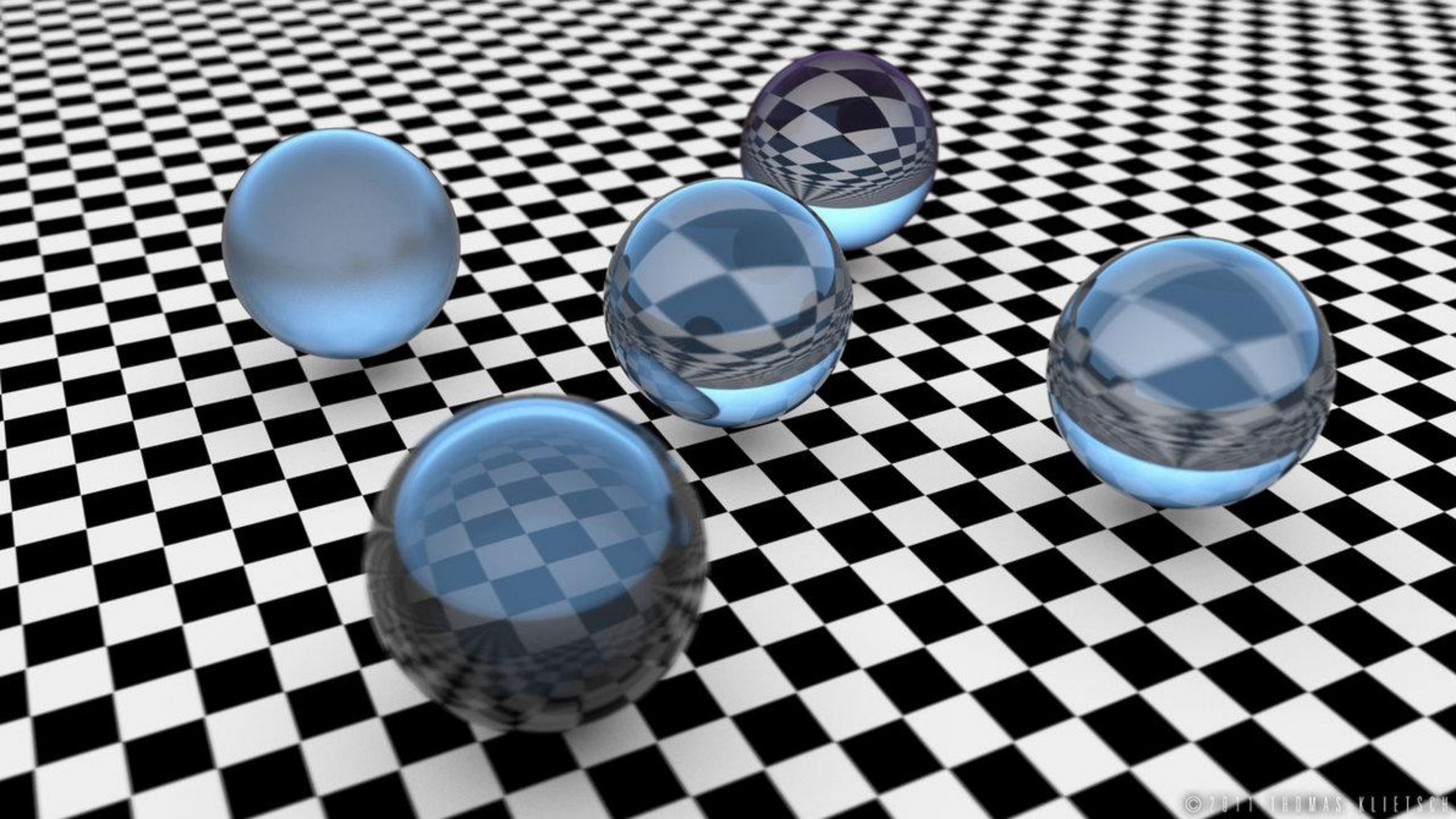
- Extension rays

Light transport

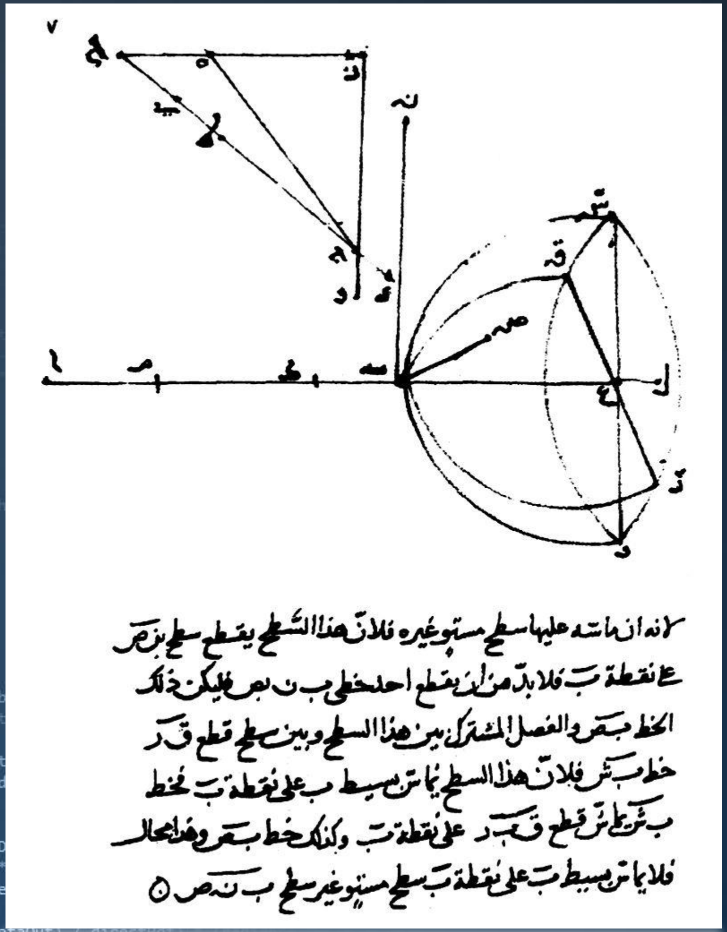


Note:

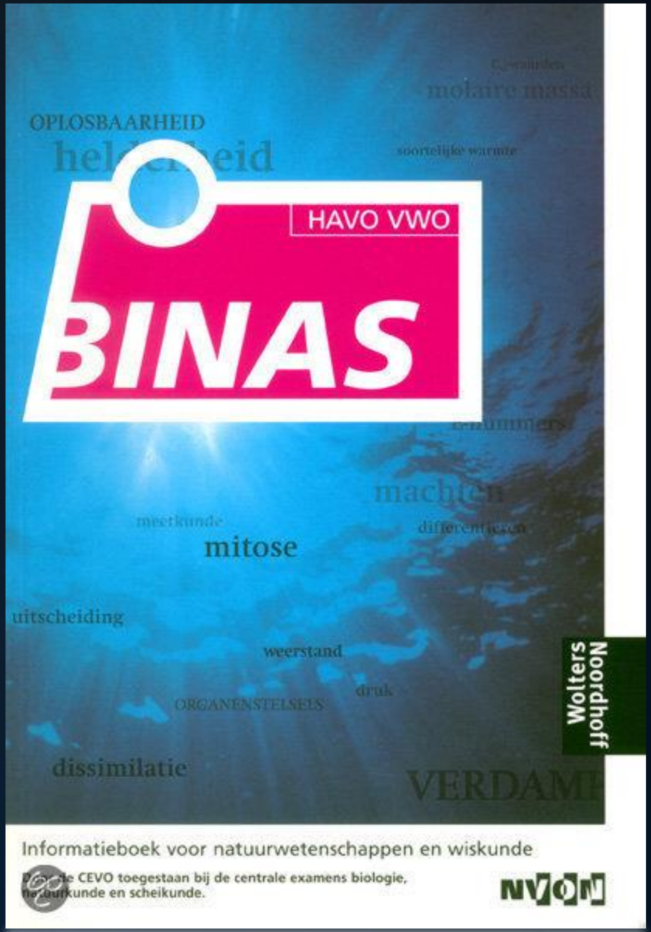
We are calculating light transport *backwards.*


©2011 THOMAS KLIETSCH

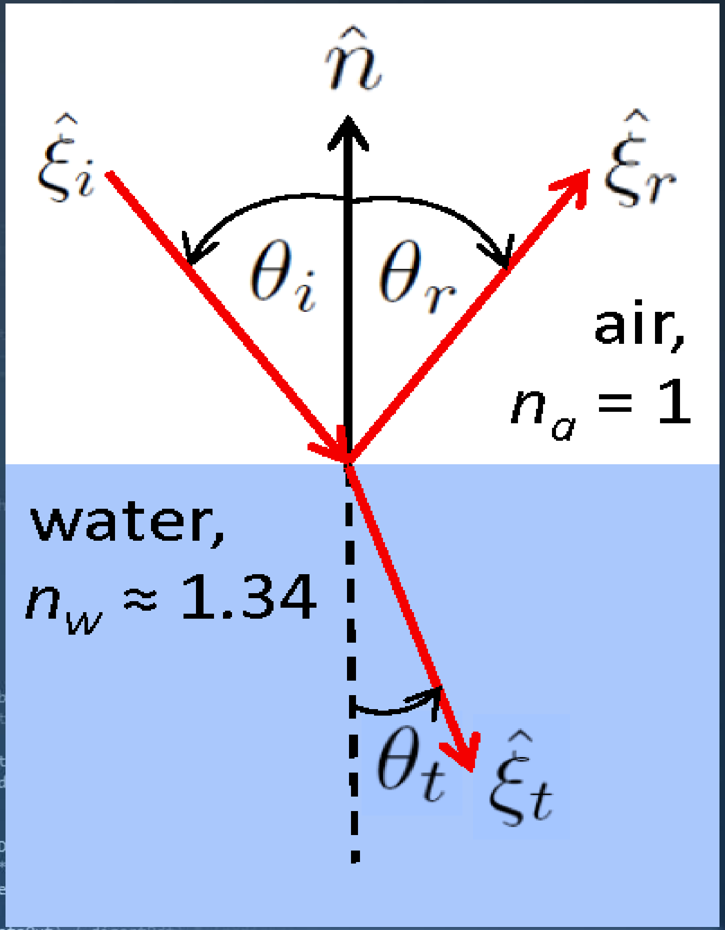
Ray Tracing

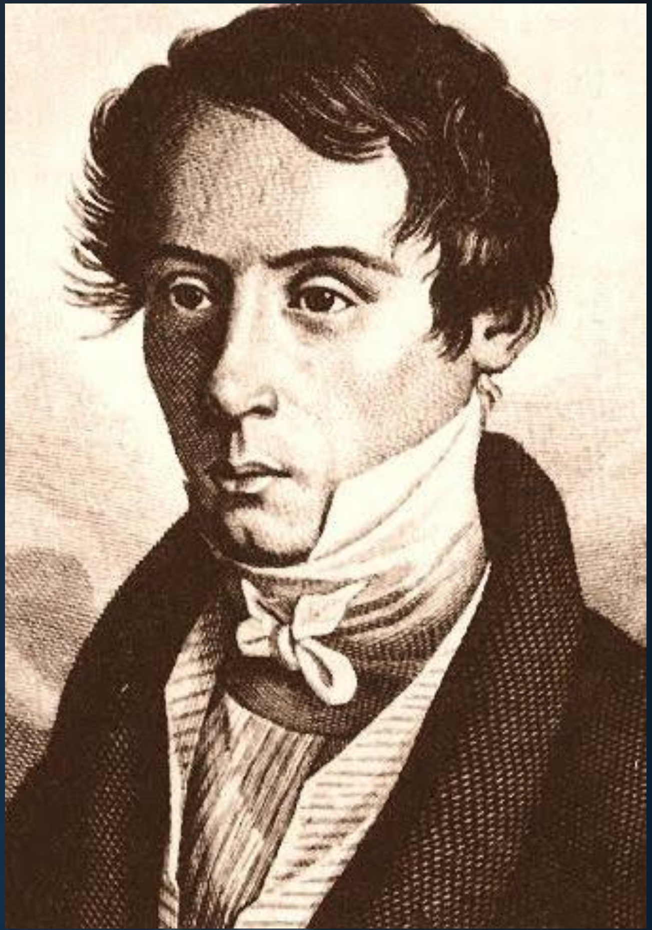


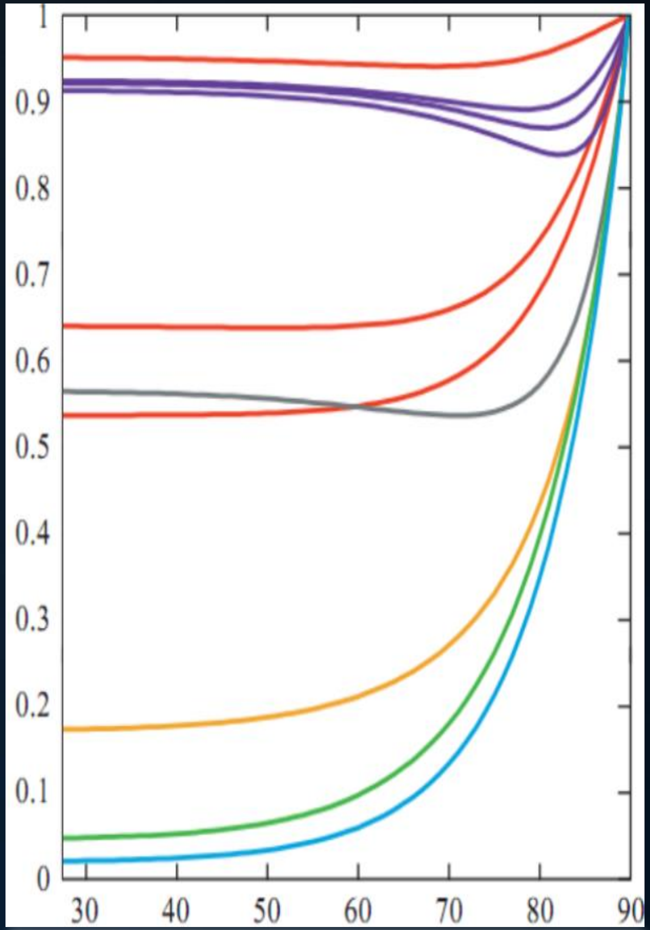

WILLEBRORDUS SNELLIUS,
Matheseos Professor.


OPLOSBAARHEID
HAVO VWO
BINAS
mitose
uitscheiding
weerstand
dissimilatie
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

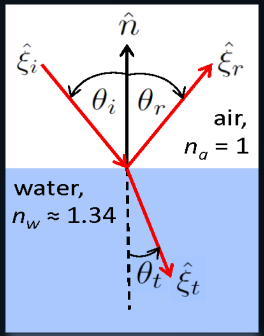
Ray tracing uses *ray optics* to simulate the behavior of light in a virtual environment.

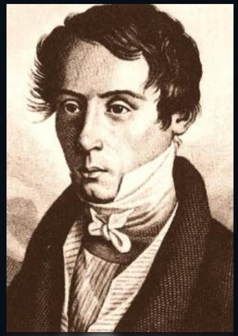
It does so by finding light transport paths:

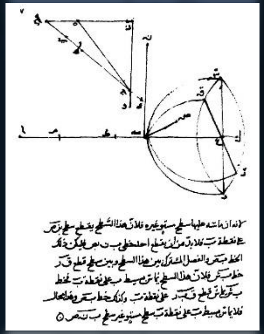
- From the ‘eye’
- Through a pixel
- Via scene surfaces
- To one or more light sources.

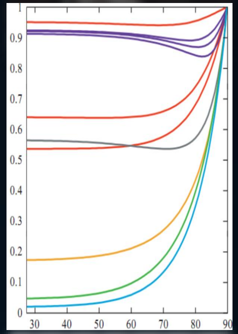
At each surface, the light is modulated. The final value is deposited at the pixel (simulating reception by a sensor).

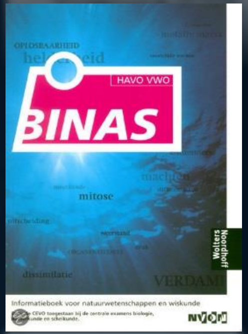
T. Whitted, “An Improved Illumination Model for Shaded Display”, ACM, 1980.











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*.

Intersect

Ray setup

Only if $\vec{V} = (0,0,1)$ *of course.*

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 E, p_0, p_1, p_2 with the camera matrix.

Intersect

Ray setup

Point on the screen:

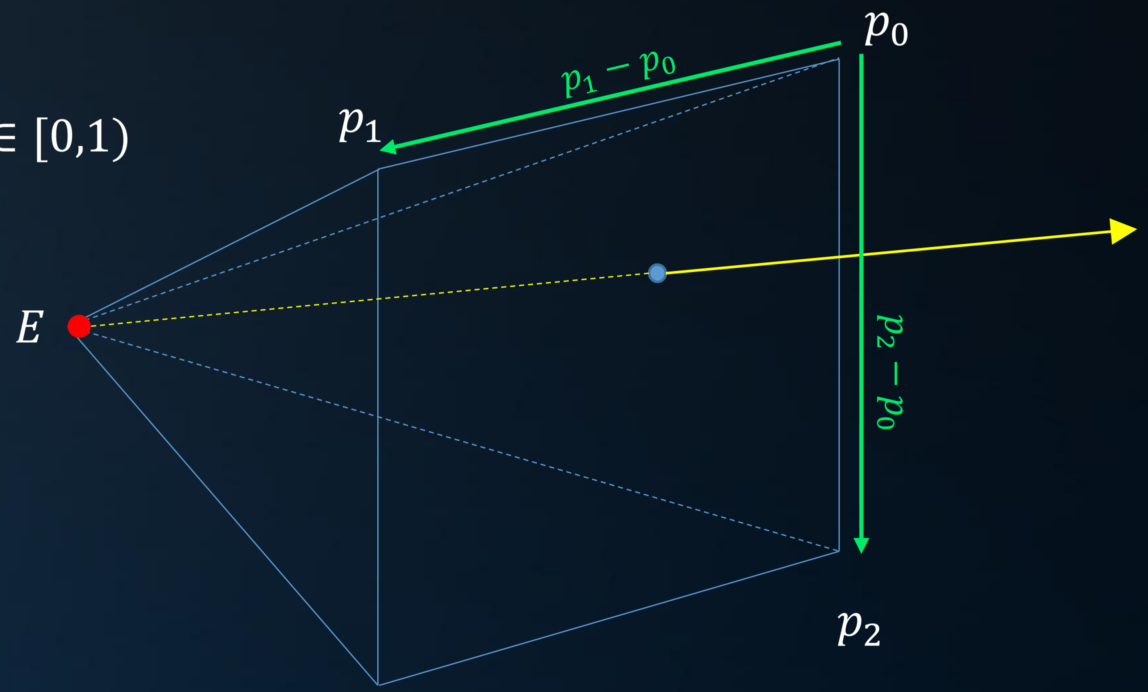
$$p(u,v) = p_0 + u(p_1 - p_0) + v(p_2 - p_0), \quad u,v \in [0,1)$$

Ray direction (before normalization):

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

Ray origin:

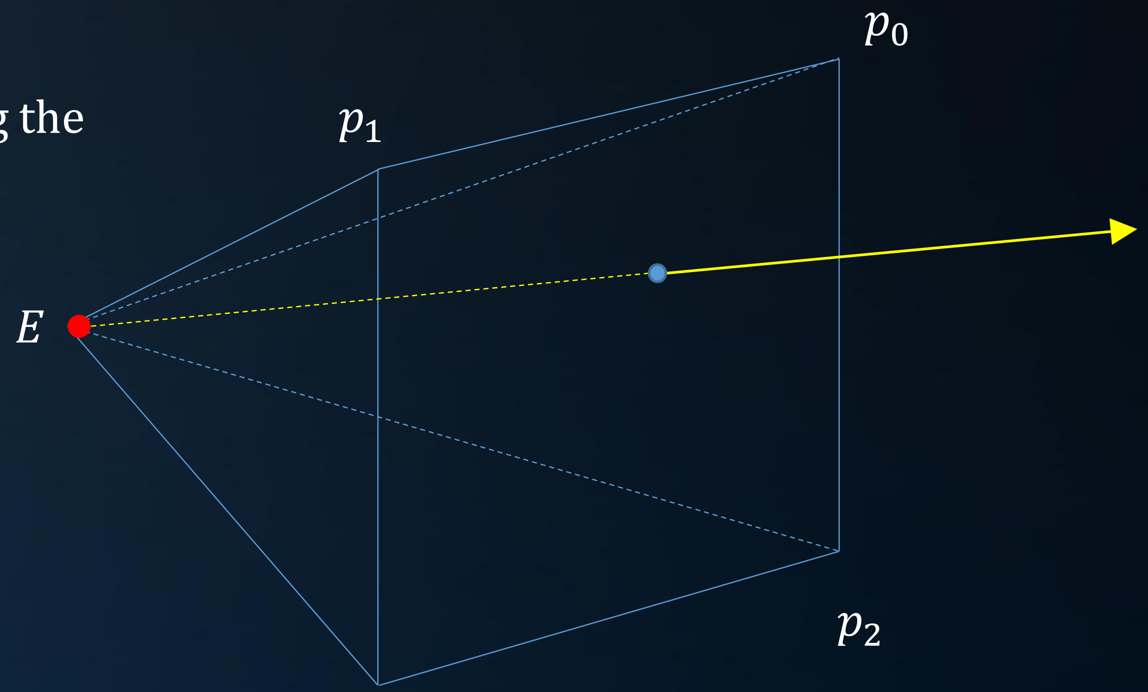
$$O = E$$



Intersect

Ray intersection

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



Ray / plane intersection:

Plane: $\text{p} \cdot \vec{N} + d = 0$

Ray: $p(t) = O + t\vec{D}$

Substituting for $p(t)$, we get

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

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

$$P = O + t\vec{D}$$

Intersect

Ray intersection

Ray / sphere intersection:

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

Ray: $p(t) = O + t\vec{D}$

Substituting for $p(t)$, we get

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

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

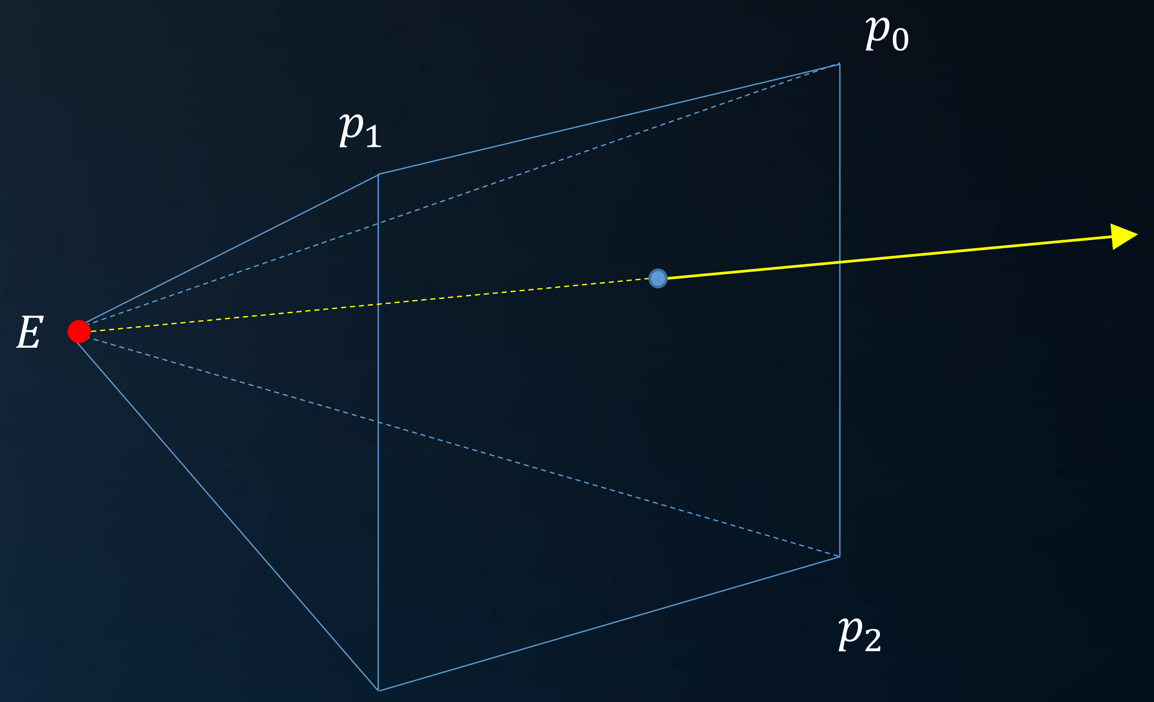
$$ax^2 + bx + c = 0 \ \rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Negative:
no intersections

$$a = \vec{D} \cdot \vec{D}$$

$$b = 2\vec{D} \cdot (O - C)$$

$$c = (O - C) \cdot (O - C) - r^2$$

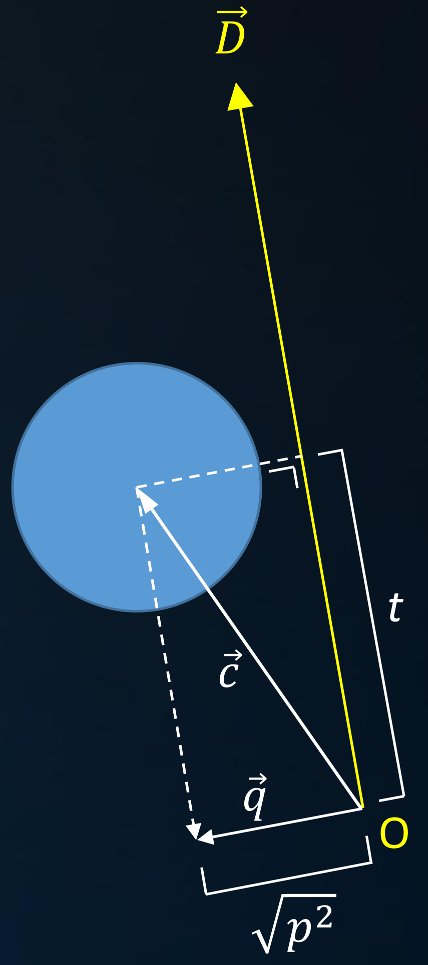


Intersect

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;
   t -= sqrt( sphere.r² – p² );
   if ((t < ray.t) && (t > 0)) ray.t = t;
   // or: ray.t = min( ray.t, max( 0, t ) );
}
```



Note:

This only works for rays that start outside the sphere.

Today's Agenda:

- Rasters
- Colors
- Ray Tracing
- Assignment P2

Checkpoint

Math Basics

- vector != point
- planes, normals
- spheres
- dot product, cross product

Assignment P1

- template
- rgb colors in 32-bit
- coordinate systems in practice
- OpenGL in C#
- vertex buffers & shaders

Checkpoint 1: MIDTERM on May 16nd
Checkpoint 2: P2 on May 28th

Assignment 2

Assignment 2: *”Write a 2D ray tracer.”*

- Using the template
- Floating point pixels
- Pretty easy minimal specs
- Tons of extra challenges

https://ncase.me/sight-and-light

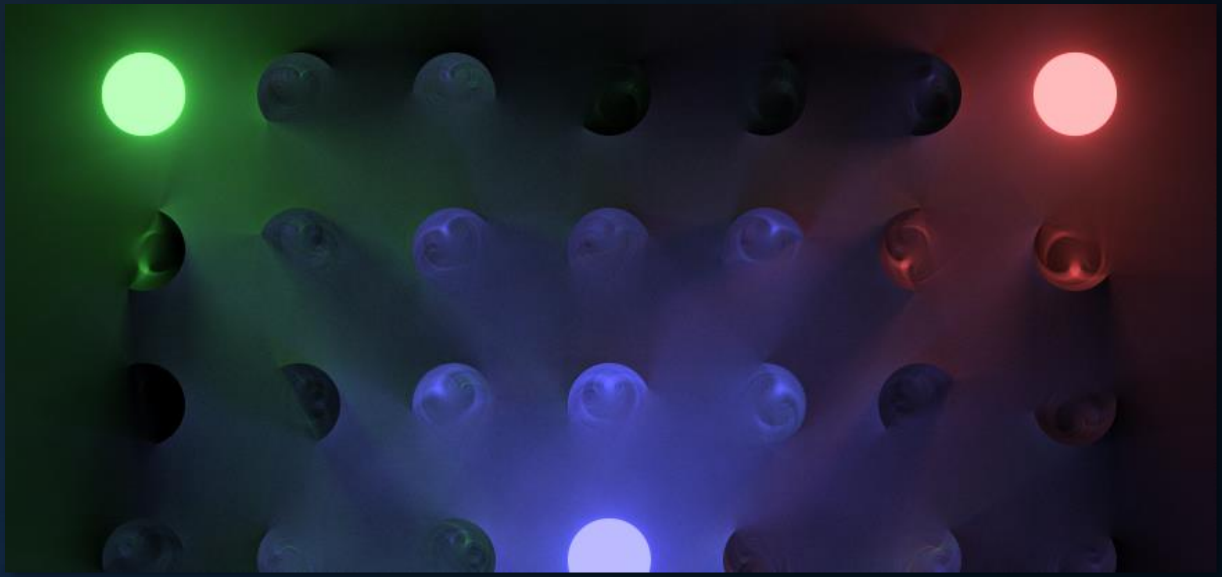
Assignment 2

Assignment 2: *"Write a 2D ray tracer."*

- Work in C#, C++, Java, ...
- Work alone or in pairs

- Special challenges

- Fastest ray tracer
- Smallest ray tracer



Assignment 2

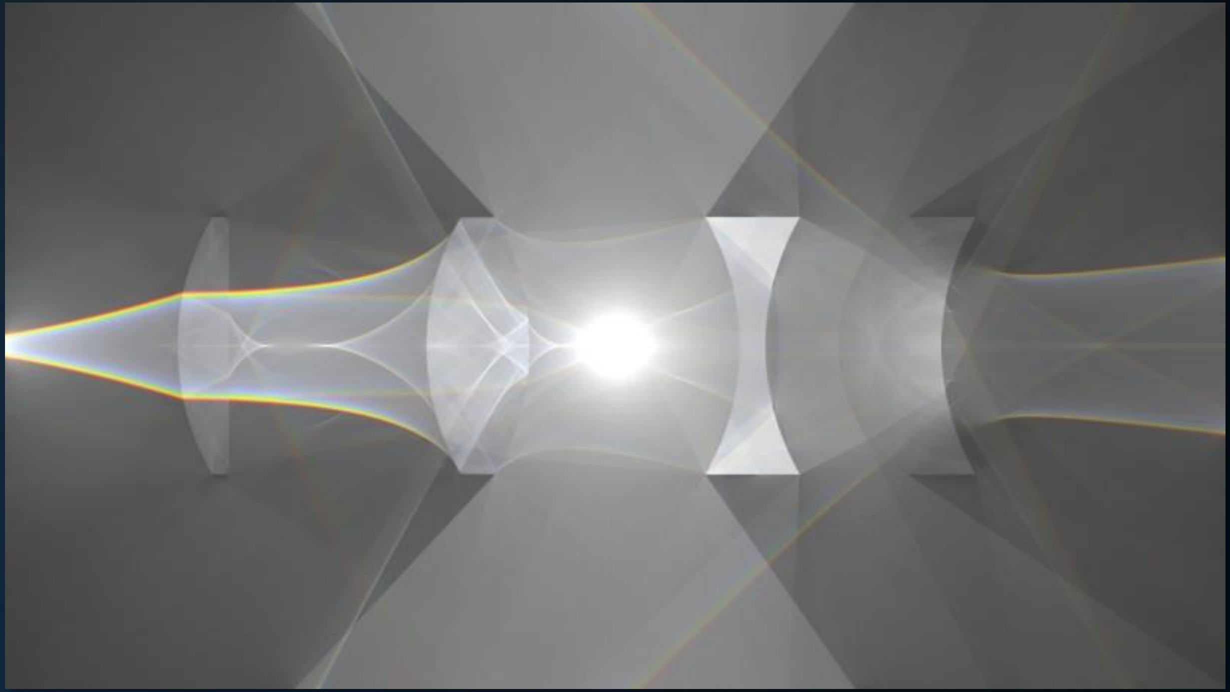
www.youtube.com/watch?time_continue=37&v=iIlu7kPCN-8

medium.com/@jerry.ylilammi/making-of-newton-protocol-e9ccde41af30

Assignment 2

Assignment 2: *”Write a 2D ray tracer.”*

- Assignment now online
- Deadline: May 28.



Today's Agenda:

- Rasters
- Colors
- Ray Tracing
- Assignment P2

INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2019

END OF lecture 5: “Graphics Fundamentals”

Next lecture: “More on Ray Tracing”