Path Tracing Workshop
Part 1: Ray Tracing

Christoph Peters
Intel Graphics Research Organization

Production rendering is complicated

Wrangling huge scenes

Giant code bases in multiple languages

Heavily optimized

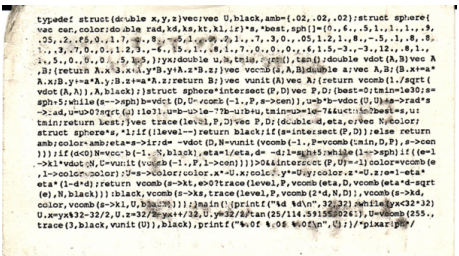
Lots of math

So much research

Cosmos Laundromat, (c) Blender Foundation

A tiny ray tracer

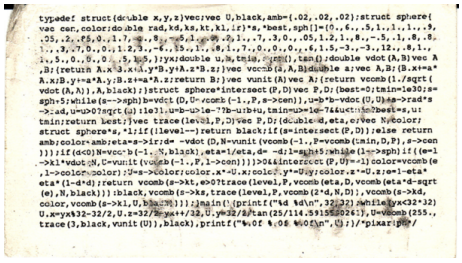
Paul Heckbert's business card (1987)



Source: https://www.realtimerendering.com/blog/back-of-the-business-card-ray-tracers/

A tiny ray tracer

Paul Heckbert's business card (1987)



typedef struct{double x,y,z}vec;vec U,black,amb={.02,.02,.02};struct sphere{
vec cen,color;double rad,kd,ks,kt,kl,ir}*s,*best,sph[]={0,.6,.5,1.,1.,1.,.9,
.05,.2,.5,0.,1.7,-1.,.8,.3,.05,1.2,1.,.8,.,-.5,.1,.8,.8
1.,.3,.7,0.,0.,1.2,3.,-.6,.15,1.,.8,1.,.7,.0,.0,.,.6,1.5,.3,-.3,.12,.,.8,1.,
1.,5.,0.,0.,.5,1.5};yx;double u,b,tmin,sqrt(),tan();double vdot(A,B)vec A
,B;{return A.x*B.x+A.y*B.y+A.z*B.z;}vec vcomb(A,A,B)double a;vec A,B;{B.x+=a*
A.x;B.y+=a*A.y;B.z+=a*A.z;return B;}vec vunit(A)vec A;{return vcomb(1./sqrt(
vdot(A,A)),A,black);}struct sphere*intersect(P,D)vec P,D;{best=0;tmin=1e30;s=
sph+5;while(s-->sph)b=vdot(D,U=vcomb(-1.,P,s->cen)),u=b*b-vdot(U,U)+s->rad*s
->rad,u=u>0?sqrt(u):1e31,u=b-u>1e-7?b-u:b+u,tmin=u>1e-7&&u<tmin?best=s,u:
tmin;return best;}vec trace(level,P,D)vec P,D;{double d,eta,e;vec N,color;
struct sphere*s,*l;if(!level--)return black;if(s=intersect(P,D));else return
amb;color=amb;eta=s->ir;d=-vdot(D,N=vunit(vcomb(-1.,P=vcomb(tmin,D,P),s->cen
)));if(d<0)N=vcomb(-1.,N,black),eta=1/eta,d=-d;l=sph+5;while(l-->sph)if((e=l
->kl*vdot(N,U=vunit(vcomb(-1.,P,l->cen)))>0&&intersect(P,U)==l)color=vcomb(e
,l->color,color);U=s->color;color.x*=U.x;color.y*=U.y;color.z*=U.z;e=1-eta*
eta*(1-d*d);return vcomb(s->kt,e>0?trace(level,P,vcomb(eta,D,vcomb(eta*d-sqrt
(e),N,black))):black,vcomb(s->ks,trace(level,P,vcomb(2*d,N,D)),vcomb(s->kd,
color,vcomb(s->kl,U,b>0?U:black))));}main(){printf("%d %d\n",32,32);while(yx<32*32)
U.x=yx%32-32/2,U.z=32/2-yx++/32,U.y=32/2/tan(25/114.5915590261),U=vcomb(255.,
trace(3,black,vunit(U)),black),printf("%0f %0f %0f\n",U.);}/*pixarific*/

Compile and run

Our goals

Learn rendering basics

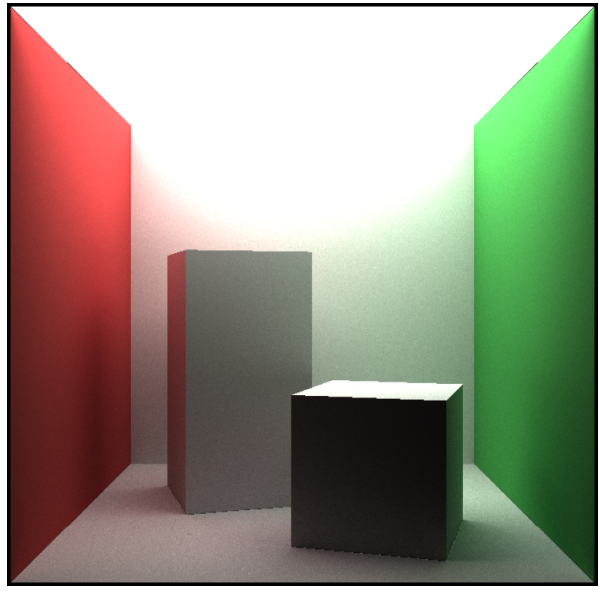
Write a path tracer

In GLSL on ShaderToy

Have fun

Part 1: Ray tracing

Part 2: Path tracing



Our goals

Learn rendering basics

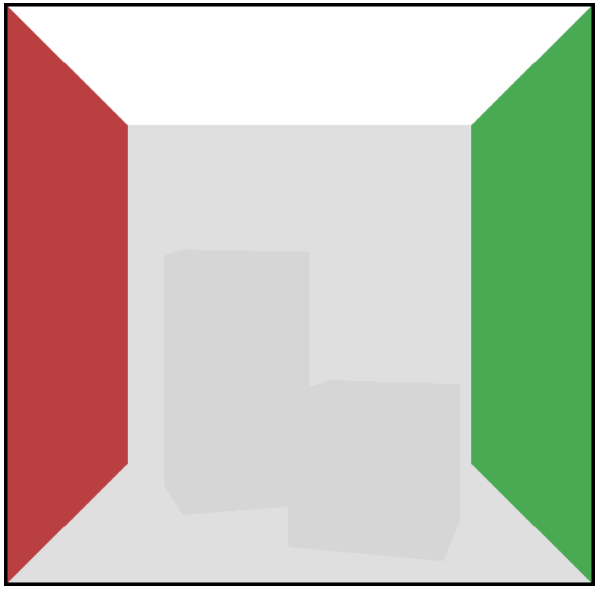
Write a path tracer

In GLSL on ShaderToy

Have fun

Part 1: Ray tracing

Part 2: Path tracing



Our goals

Learn rendering basics

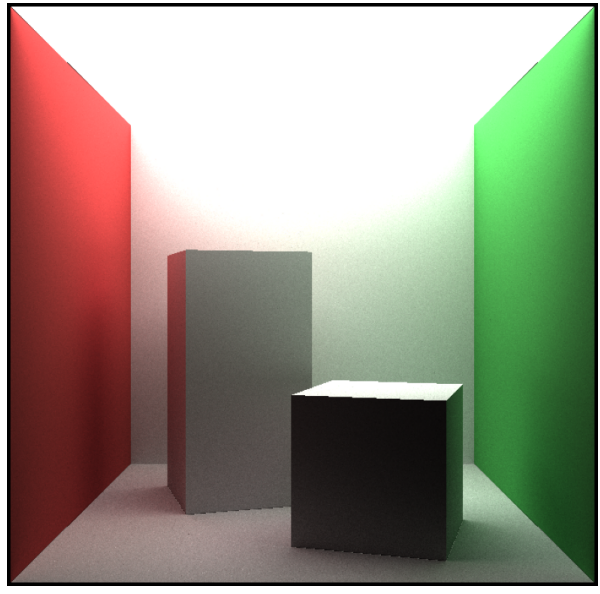
Write a path tracer

In GLSL on ShaderToy

Have fun

Part 1: Ray tracing

Part 2: Path tracing



Not our goals

Framework provided (~60 lines)

Scene provided

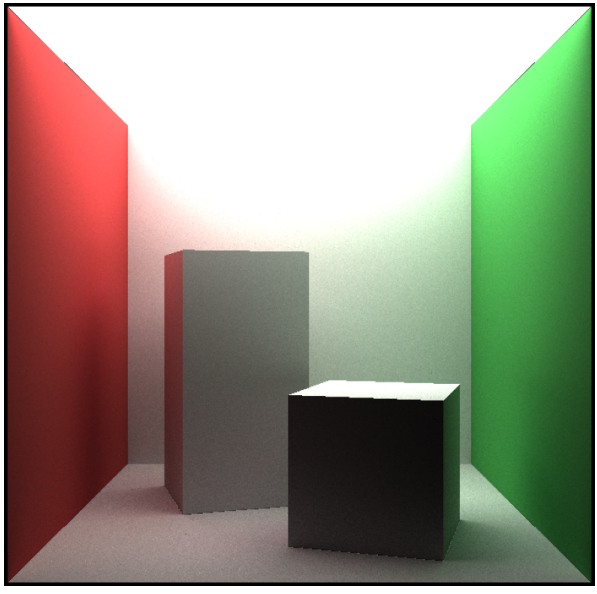
Only diffuse triangles

No textures

No acceleration structures

No clever sampling/denoising

GPUs but no graphics APIs



Not our goals

Framework provided (~60 lines)

Scene provided

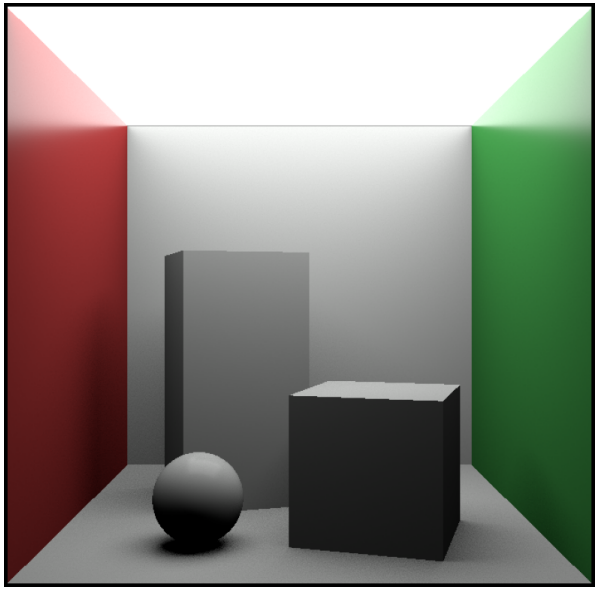
Only diffuse triangles

No textures

No acceleration structures

No clever sampling/denoising

GPUs but no graphics APIs



Not our goals

Framework provided (~60 lines)

Scene provided

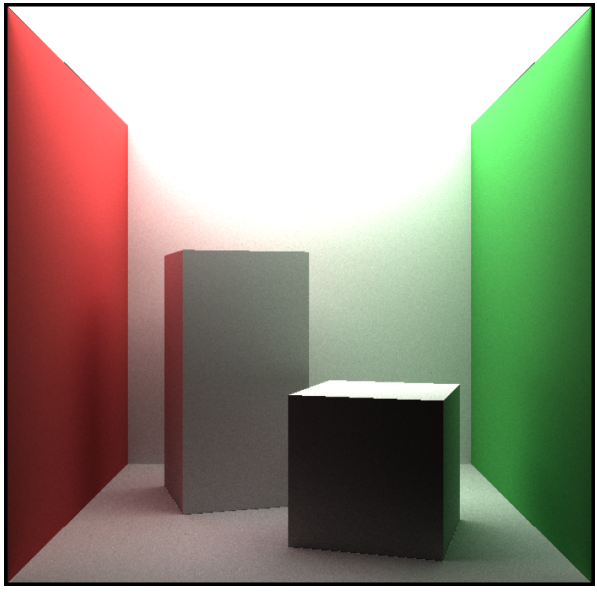
Only diffuse triangles

No textures

No acceleration structures

No clever sampling/denoising

GPUs but no graphics APIs



Format

Two sessions, each with:

- ~35 minutes of presentation
- 4 exercises for you to solve in between

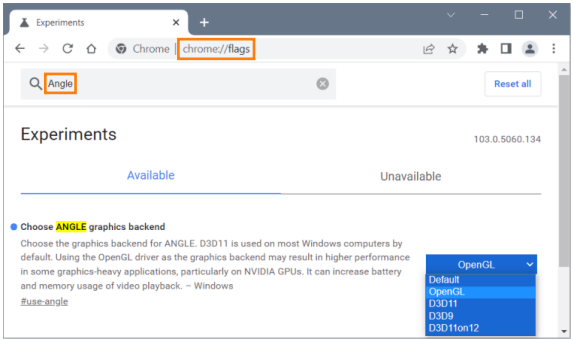
Videos and PDF slides available

Prerequisites:

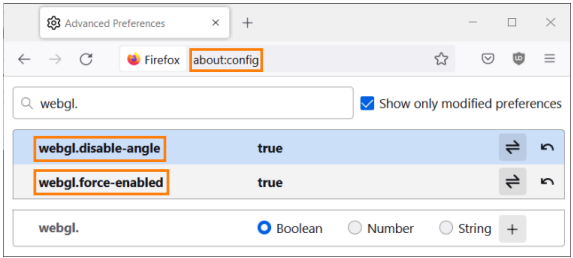
- Browser
- Experience with C-like languages
- Basic vector math (e.g. dot products, linear equations)

Proper WebGL config

By default, ANGLE makes big WebGL shaders run slowly on Windows



Then restart Chrome



Then reload ShaderToy tabs

GLSL and ShaderToy

GLSL

C-like shader language for OpenGL, Vulkan and WebGL

Unusual restrictions, especially with WebGL

for-loops of fixed max. length

No while-loops

Fixed-size arrays, no pointers

Indices must be constants

```
bool find(out int out_index, int a[6], int v) {
    for (int i = 0; i != 6; ++i) {
        if (a[i] == v) {
            out_index = i;
            return true;
        }
    }
    return false;
}
```

Parameters marked `in`, `out` or `inout` (copy values in, out or both)

GLSL

Built-in scalar, vector and matrix types:

```
int   float   vec2   vec3   vec4   mat2   mat3
```

Constructor-like functions:

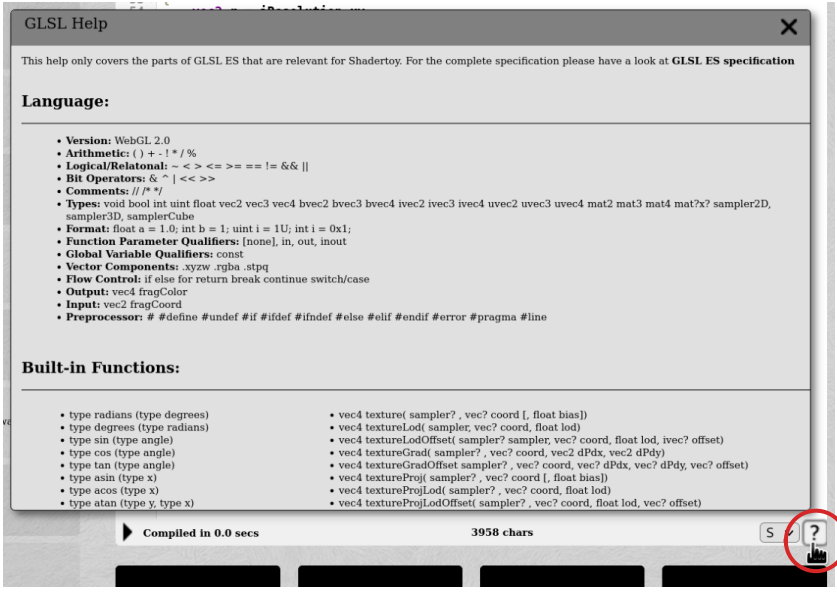
```
vec3 v = vec3(x, y, z);   mat3 A = mat3(col_0, col_1, col_2);
```

Standard functions/operators for geometric operations:

```
dot(v, w)   inverse(A)   v + w   A * v   2.0 * v
```

Swizzles and array operators:

```
vec2(v[1], v[2]) == v.yz   v.xyz == v.rgb   float col_1_row_2 = A[1][2];
```



Full docs: https://www.khronos.org/registry/OpenGL/specs/es/3.0/GLSL_ES_Specification_3.00.pdf

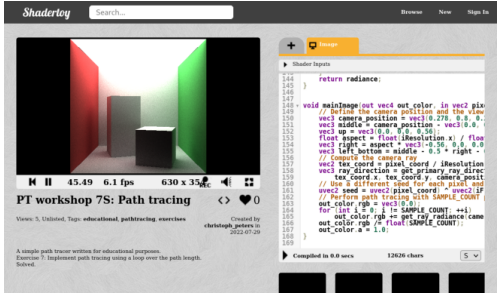
ShaderToy

Runs a full-viewport fragment shader in WebGL

Program that runs once per pixel to compute its color

In each exercise you complete 1 function in a ShaderToy (`// TODO`)

Exercise N+1 has a reference solution for exercise N (no peeking)



ShaderToy

Runs a full-viewport fragment shader in WebGL

Program that runs once per pixel to compute its color

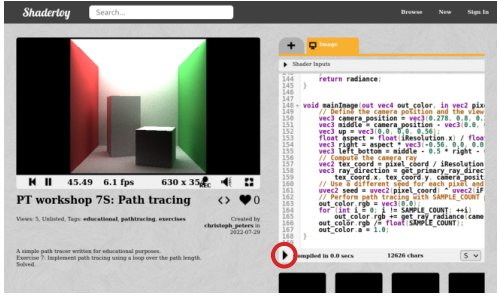
In each exercise you complete 1 function in a ShaderToy (`// TODO`)

Exercise N+1 has a reference solution for exercise N (no peeking)

To change the code, just type

To recompile/run, click play

To save, copy your code to a text file



ShaderToy

Runs a full-viewport fragment shader in WebGL

Program that runs once per pixel to compute its color

In each exercise you complete 1 function in a ShaderToy (`// TODO`)

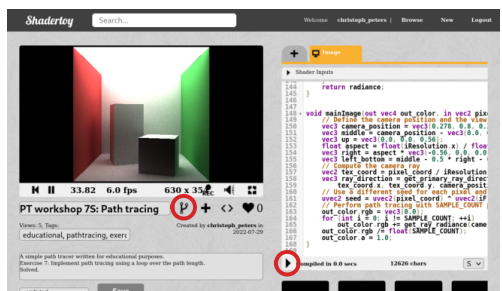
Exercise N+1 has a reference solution for exercise N (no peeking)

To change the code, just type

To recompile/run, click play

To save, copy your code to a text file

Or create an account and a fork



Shadertoy | Search... | Browse | New | Sign In

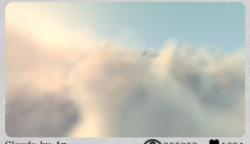
Sort: Popular | Newest | Love | Hot

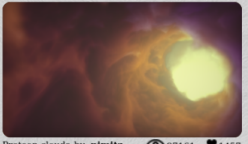
Filter: Multipass | GPU Sound | VR | Microphone | Soundcloud | Webcam

View: Slideshow

Results (62249): 1 | 2 | 3 | ... | 5188

Seascape by **TDM** 572647 2277


Clouds by **iq** 255953 1684


Protean clouds by **nimitz** 87161 1457

Raymarching - Primitives by **I** 624851 1260

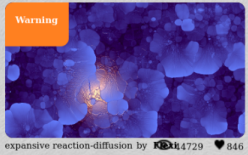

Rainforest by **iq** 216604 996


Flame by **XT95** 142290 982


Star Nest by **Kali** 58215 966

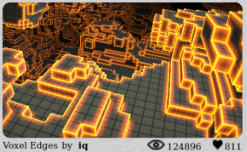

Creation by Silexars by **Danguafer** 137162 945


Fractal Land by **Kali** 85468 894



expansive reaction-diffusion by **K** 4729 846


Elevated by **iq** 189183 840


Voxel Edges by **iq** 124896 811

intel

https://www.shadertoy.com/results?sort=love

Exercise 0: Hello ShaderToy

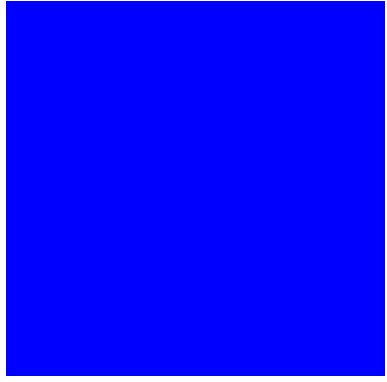
Open the ShaderToy linked below

Change background color from red to blue

Click the play button

See your result

Use `vec3()`



Correct result

shadertoy.com/view/ssyfWK

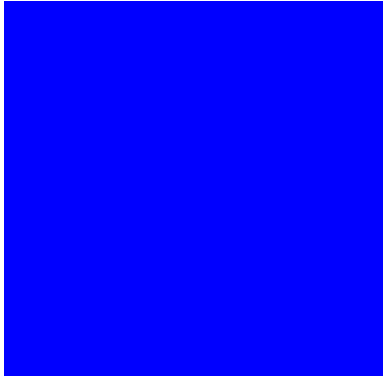
Exercise 0: Hello ShaderToy

Open the ShaderToy linked below

Change background color from red to blue

Click the play button

See your result



Correct result

Use `vec3()`

shadertoy.com/view/ssyfWK

Scene representation

Camera

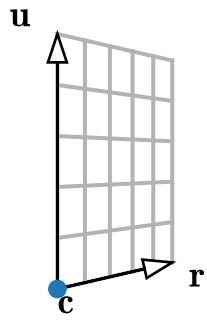
Defined by position $\mathbf{o} \in \mathbb{R}^3$

Camera

Defined by position $\mathbf{o} \in \mathbb{R}^3$

And an image plane, with:

- Left bottom corner $\mathbf{c} \in \mathbb{R}^3$
- Right and up vectors $\mathbf{r}, \mathbf{u} \in \mathbb{R}^3$



Camera

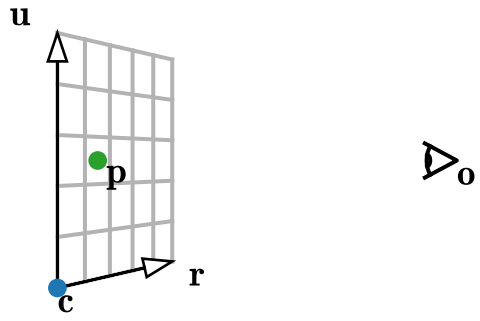
Defined by position $\mathbf{o} \in \mathbb{R}^3$

And an image plane, with:

- Left bottom corner $\mathbf{c} \in \mathbb{R}^3$
- Right and up vectors $\mathbf{r}, \mathbf{u} \in \mathbb{R}^3$

Point for image coords $x, y \in [0, 1]$:

$$\mathbf{p} = \mathbf{c} + x\,\mathbf{r} + y\,\mathbf{u}$$



Camera

Defined by position $\mathbf{o} \in \mathbb{R}^3$

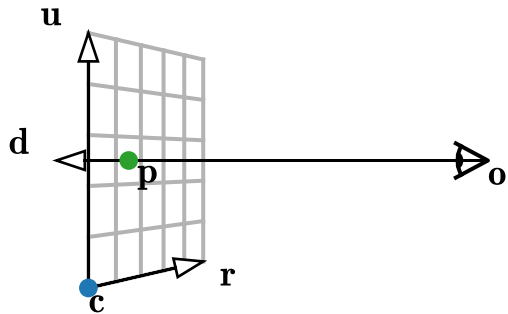
And an image plane, with:

- Left bottom corner $\mathbf{c} \in \mathbb{R}^3$
- Right and up vectors $\mathbf{r}, \mathbf{u} \in \mathbb{R}^3$

Point for image coords $x, y \in [0, 1]$:

$$\mathbf{p} = \mathbf{c} + x\,\mathbf{r} + y\,\mathbf{u}$$

Ray direction: $\mathbf{d} = \frac{\mathbf{p} - \mathbf{o}}{||\mathbf{p} - \mathbf{o}||}$

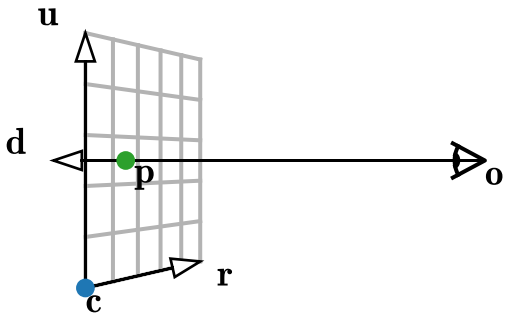


Camera

Defined by position $\mathbf{o} \in \mathbb{R}^3$

And an image plane, with:

- Left bottom corner $\mathbf{c} \in \mathbb{R}^3$
- Right and up vectors $\mathbf{r}, \mathbf{u} \in \mathbb{R}^3$



Point for image coords $x, y \in [0, 1]$:

$$\mathbf{p} = \mathbf{c} + x\,\mathbf{r} + y\,\mathbf{u}$$

Ray direction: $\mathbf{d} = \frac{\mathbf{p} - \mathbf{o}}{||\mathbf{p} - \mathbf{o}||} = \text{normalize}(\mathbf{p} - \mathbf{o})$

Exercise 1: Primary rays (a.k.a. camera rays)

Complete `get_primary_ray_direction()`

Inputs: $x, y, \mathbf{o}, \mathbf{c}, \mathbf{r}, \mathbf{u}$

Output: ray direction $\mathbf{d}$

The framework displays $\mathbf{d}$ as color

Use formulas from the previous slide

Use `+, -, *, normalize()`

Correct result

shadertoy.com/view/7dGBDK

Exercise 1: Primary rays (a.k.a. camera rays)

Complete `get_primary_ray_direction()`

Inputs: $x, y, \mathbf{o}, \mathbf{c}, \mathbf{r}, \mathbf{u}$

Output: ray direction $\mathbf{d}$

The framework displays $\mathbf{d}$ as color

Use formulas from the previous slide

Use `+, -, *, normalize()`

Correct result

shadertoy.com/view/7dGBDK

Triangle mesh

Our scene geometry is a triangle mesh

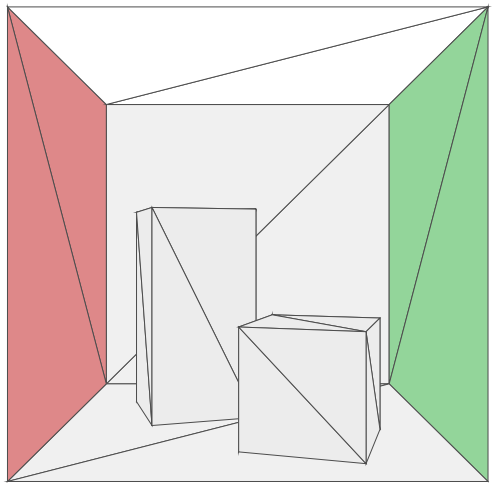
Each triangle has:

3 vertex positions $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^3$

A normal vector $\mathbf{n} := \frac{(\mathbf{v}_1 - \mathbf{v}_0) \times (\mathbf{v}_2 - \mathbf{v}_0)}{||(\mathbf{v}_1 - \mathbf{v}_0) \times (\mathbf{v}_2 - \mathbf{v}_0)||}$

An RGB color

A color for light emission (usually 0)



Mesh representation

```
// A triangle along with some shading parameters
struct triangle_t {
    // The positions of the three vertices (v_0, v_1, v_2)
    vec3 positions[3];
    // A vector of length 1, orthogonal to the triangle (n)
    vec3 normal;
    // The albedo of the triangle (i.e. the fraction of
    // red/green/blue light that gets reflected) (a)
    vec3 color;
    // The radiance emitted by the triangle (for light sources) (L_e)
    vec3 emission;
};
// ...
#define TRIANGLE_COUNT 30
triangle_t tris[TRIANGLE_COUNT];
tris[0].positions[0] = vec3(0.000000133, -0.559199989, 0.548799932); tris[0].pos
tris[0].normal = vec3(0.0, 1.0, 0.0); tris[1].normal = vec3(0.301707575, -0.9534
tris[0].color = vec3(0.874000013, 0.874000013, 0.875000000); tris[1].color = vec
```

Mesh representation

```
8799932); tris[29].positions[2] = vec3(0.555999935, -0.559199989, 0.548799932);
```

Ray tracing

Ray-mesh intersection test

What do we see along a ray?

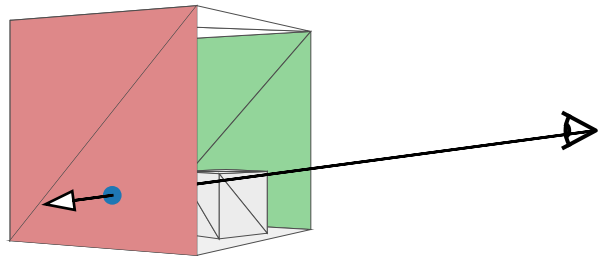
The closest intersected triangle!

Ray tracing finds this closest hit

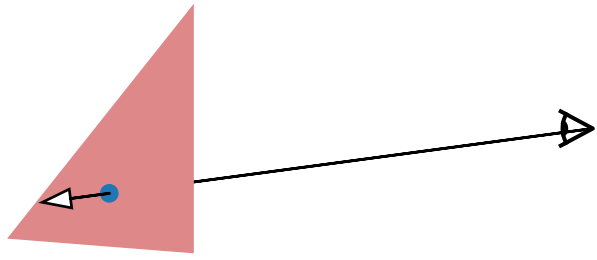
Foundation of path tracing

Implemented in hardware

But we do it in software



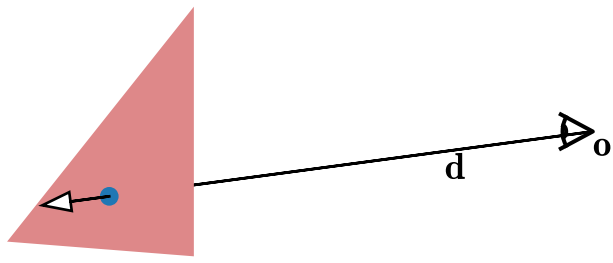
Ray-triangle intersection test



Ray-triangle intersection test

Ray has origin $\mathbf{o}$ and direction $\mathbf{d}$

$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with $t \geq 0$

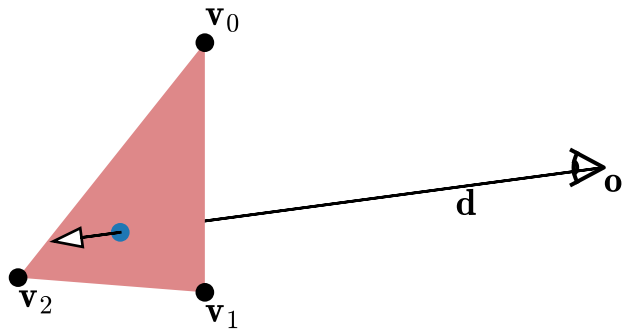


Ray-triangle intersection test

Ray has origin $\mathbf{o}$ and direction $\mathbf{d}$

$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with $t \geq 0$

Triangle has vertices $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$



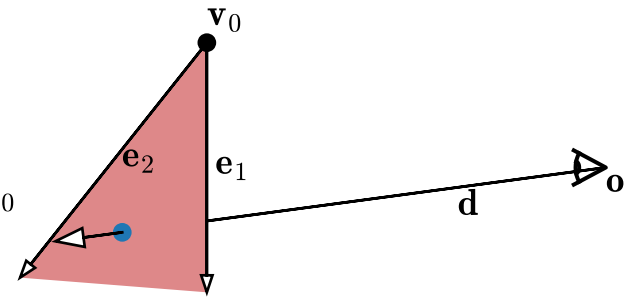
Ray-triangle intersection test

Ray has origin $\mathbf{o}$ and direction $\mathbf{d}$

$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with $t \geq 0$

Triangle has vertices $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$

Edges $\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0,\ \mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0$



Ray-triangle intersection test

Ray has origin $\mathbf{o}$ and direction $\mathbf{d}$

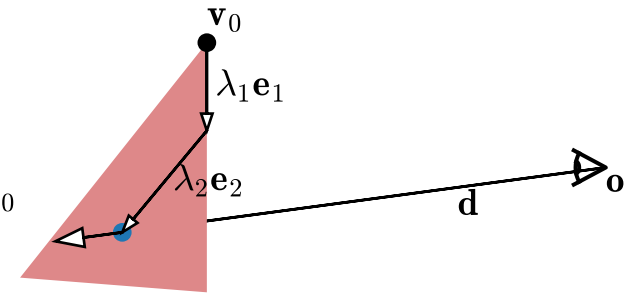
$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with $t \geq 0$

Triangle has vertices $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$

Edges $\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0,\ \mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0$

$\mathbf{x}(\lambda_1, \lambda_2) = \mathbf{v}_0 + \lambda_1\mathbf{e}_1 + \lambda_2\mathbf{e}_2$

with $\lambda_1, \lambda_2 \geq 0$ and $\lambda_1 + \lambda_2 \leq 1$



Ray-triangle intersection test

Ray has origin $\mathbf{o}$ and direction $\mathbf{d}$

$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with $t \geq 0$

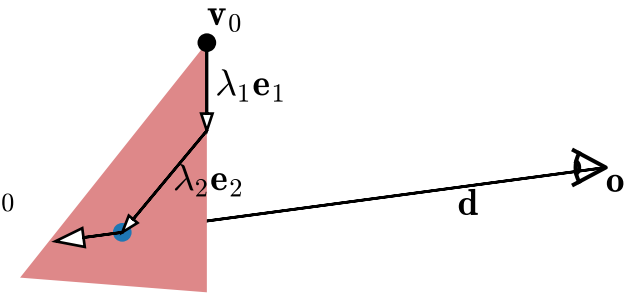
Triangle has vertices $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$

Edges $\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0,\ \mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0$

$\mathbf{x}(\lambda_1, \lambda_2) = \mathbf{v}_0 + \lambda_1\mathbf{e}_1 + \lambda_2\mathbf{e}_2$

with $\lambda_1, \lambda_2 \geq 0$ and $\lambda_1 + \lambda_2 \leq 1$

Intersection expressed as $\mathbf{x}(\lambda_1, \lambda_2) = \mathbf{r}(t)$



Ray-triangle intersection test

$$\mathbf{x}(\lambda_1, \lambda_2) = \mathbf{r}(t)$$

Ray-triangle intersection test

$$\mathbf{x}(\lambda_1, \lambda_2) = \mathbf{r}(t)$$

$$\Leftrightarrow \ \mathbf{v}_0 + \lambda_1 \mathbf{e}_1 + \lambda_2 \mathbf{e}_2 = \mathbf{o} + t\mathbf{d}$$

Ray-triangle intersection test

$$\mathbf{x}(\lambda_1, \lambda_2) = \mathbf{r}(t)$$

$$\Leftrightarrow \ \mathbf{v}_0 + \lambda_1 \mathbf{e}_1 + \lambda_2 \mathbf{e}_2 = \mathbf{o} + t\mathbf{d}$$

$$\Leftrightarrow \ -t\mathbf{d} + \lambda_1 \mathbf{e}_1 + \lambda_2 \mathbf{e}_2 = \mathbf{o} - \mathbf{v}_0$$

Ray-triangle intersection test

$$\mathbf{x}(\lambda_1, \lambda_2) = \mathbf{r}(t)$$

$$\Leftrightarrow \ \mathbf{v}_0 + \lambda_1 \mathbf{e}_1 + \lambda_2 \mathbf{e}_2 = \mathbf{o} + t\mathbf{d}$$

$$\Leftrightarrow \ -t\mathbf{d} + \lambda_1 \mathbf{e}_1 + \lambda_2 \mathbf{e}_2 = \mathbf{o} - \mathbf{v}_0$$

$$\Leftrightarrow \ \begin{pmatrix} | & | & | \\ -\mathbf{d} & \mathbf{e}_1 & \mathbf{e}_2 \\ | & | & | \end{pmatrix} \begin{pmatrix} t \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \mathbf{o} - \mathbf{v}_0$$

3×3 matrix with columns $-\mathbf{d}, \mathbf{e}_1, \mathbf{e}_2$

Ray-triangle intersection test

$$\mathbf{x}(\lambda_1, \lambda_2) = \mathbf{r}(t)$$

$$\Leftrightarrow \ \mathbf{v}_0 + \lambda_1 \mathbf{e}_1 + \lambda_2 \mathbf{e}_2 = \mathbf{o} + t\mathbf{d}$$

$$\Leftrightarrow \ -t\mathbf{d} + \lambda_1 \mathbf{e}_1 + \lambda_2 \mathbf{e}_2 = \mathbf{o} - \mathbf{v}_0$$

$$\Leftrightarrow \ \begin{pmatrix} | & | & | \\ -\mathbf{d} & \mathbf{e}_1 & \mathbf{e}_2 \\ | & | & | \end{pmatrix} \begin{pmatrix} t \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \mathbf{o} - \mathbf{v}_0$$

$$\Leftrightarrow \ \begin{pmatrix} t \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} | & | & | \\ -\mathbf{d} & \mathbf{e}_1 & \mathbf{e}_2 \\ | & | & | \end{pmatrix}^{-1} (\mathbf{o} - \mathbf{v}_0)$$

3×3 matrix with columns $-\mathbf{d}, \mathbf{e}_1, \mathbf{e}_2$

Ray-triangle intersection test

$$\mathbf{x}(\lambda_1, \lambda_2) = \mathbf{r}(t)$$

$$\Leftrightarrow \ \mathbf{v}_0 + \lambda_1 \mathbf{e}_1 + \lambda_2 \mathbf{e}_2 = \mathbf{o} + t\mathbf{d}$$

$$\Leftrightarrow \ -t\mathbf{d} + \lambda_1 \mathbf{e}_1 + \lambda_2 \mathbf{e}_2 = \mathbf{o} - \mathbf{v}_0$$

$$\Leftrightarrow \ \begin{pmatrix} | & | & | \\ -\mathbf{d} & \mathbf{e}_1 & \mathbf{e}_2 \\ | & | & | \end{pmatrix} \begin{pmatrix} t \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \mathbf{o} - \mathbf{v}_0$$

3×3 matrix with columns $-\mathbf{d}, \mathbf{e}_1, \mathbf{e}_2$

$$\Leftrightarrow \ \begin{pmatrix} t \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} | & | & | \\ -\mathbf{d} & \mathbf{e}_1 & \mathbf{e}_2 \\ | & | & | \end{pmatrix}^{-1} (\mathbf{o} - \mathbf{v}_0)$$

Intersection if $t, \lambda_1, \lambda_2 \geq 0$ and $\lambda_1 + \lambda_2 \leq 1$

Ray-triangle intersection test

$$\mathbf{x}(\lambda_1, \lambda_2) = \mathbf{r}(t)$$

$$\Leftrightarrow \quad \mathbf{v}_0 + \lambda_1 \mathbf{e}_1 + \lambda_2 \mathbf{e}_2 = \mathbf{o} + t\mathbf{d}$$

$$\Leftrightarrow \quad -t\mathbf{d} + \lambda_1 \mathbf{e}_1 + \lambda_2 \mathbf{e}_2 = \mathbf{o} - \mathbf{v}_0$$

$$\Leftrightarrow \quad \begin{pmatrix} | & | & | \\ -\mathbf{d} & \mathbf{e}_1 & \mathbf{e}_2 \\ | & | & | \end{pmatrix} \begin{pmatrix} t \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \mathbf{o} - \mathbf{v}_0$$

3×3 matrix with columns $-\mathbf{d}, \mathbf{e}_1, \mathbf{e}_2$

$$\Leftrightarrow \quad \begin{pmatrix} t \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} | & | & | \\ -\mathbf{d} & \mathbf{e}_1 & \mathbf{e}_2 \\ | & | & | \end{pmatrix}^{-1} (\mathbf{o} - \mathbf{v}_0)$$

Intersection if $t, \lambda_1, \lambda_2 \geq 0$ and $\lambda_1 + \lambda_2 \leq 1$

Implement this

Exercise 2: Ray-triangle intersection test

Complete `ray_triangle_intersection()`

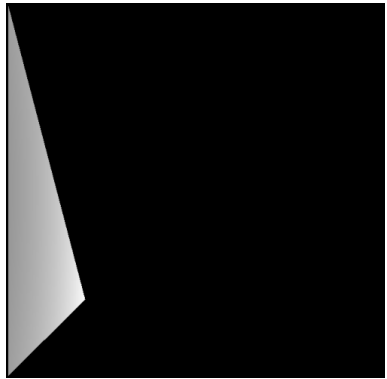
Inputs: $\mathbf{o}$, $\mathbf{d}$, triangle

Outputs: boolean and ray parameter t

Framework displays t for one triangle

Use end result from the previous slide

Use `mat3(col_0, col_1, col_2), inverse(), *, &&`



Correct result
(cropped)

shadertoy.com/view/sdyBDK

Exercise 2: Ray-triangle intersection test

Complete `ray_triangle_intersection()`

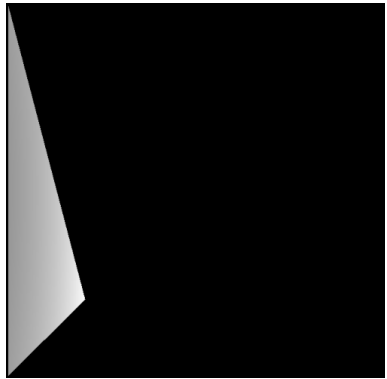
Inputs: $\mathbf{o}$, $\mathbf{d}$, triangle

Outputs: boolean and ray parameter t

Framework displays t for one triangle

Use end result from the previous slide

Use `mat3(col_0, col_1, col_2), inverse(), *, &&`



Correct result (cropped)

shadertoy.com/view/sdyBDK

Ray-mesh intersection test

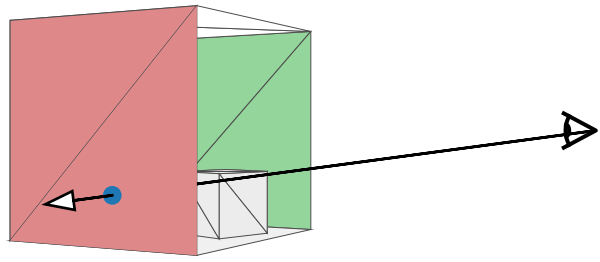
Initialize $t_{\min} = \infty$ (or 1.0e38)

Test each triangle in a for-loop

If there is a hit with $t < t_{\min}$:

Set $t_{\min} = t$

Store the triangle



Output closest hit triangle and $t_{\min}$ or no hit if still $t_{\min} = \infty$

Does not scale but we do it anyway

Exercise 3: Ray-mesh intersection test

Complete `ray_mesh_intersection()`

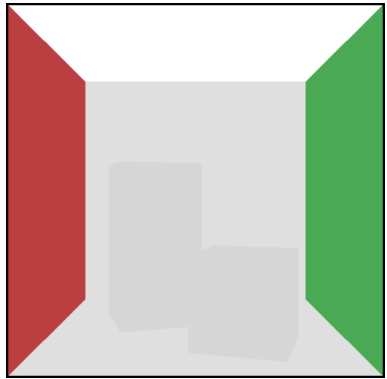
Inputs: $\mathbf{o}$, $\mathbf{d}$, mesh triangles (defined inline)

Outputs: boolean, t and intersected triangle

Framework draws scene with ray tracing

Use the approach on the previous slide

Use `for, if, =, <, &&`



Correct result

shadertoy.com/view/7dKBD3

Exercise 3: Ray-mesh intersection test

Complete `ray_mesh_intersection()`

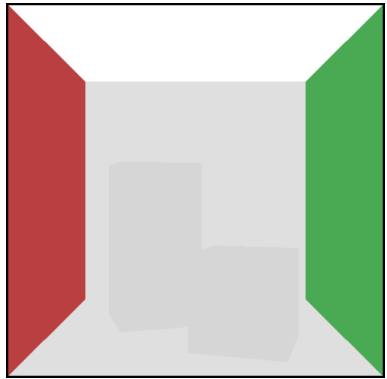
Inputs: $\mathbf{o}$, $\mathbf{d}$, mesh triangles (defined inline)

Outputs: boolean, t and intersected triangle

Framework draws scene with ray tracing

Use the approach on the previous slide

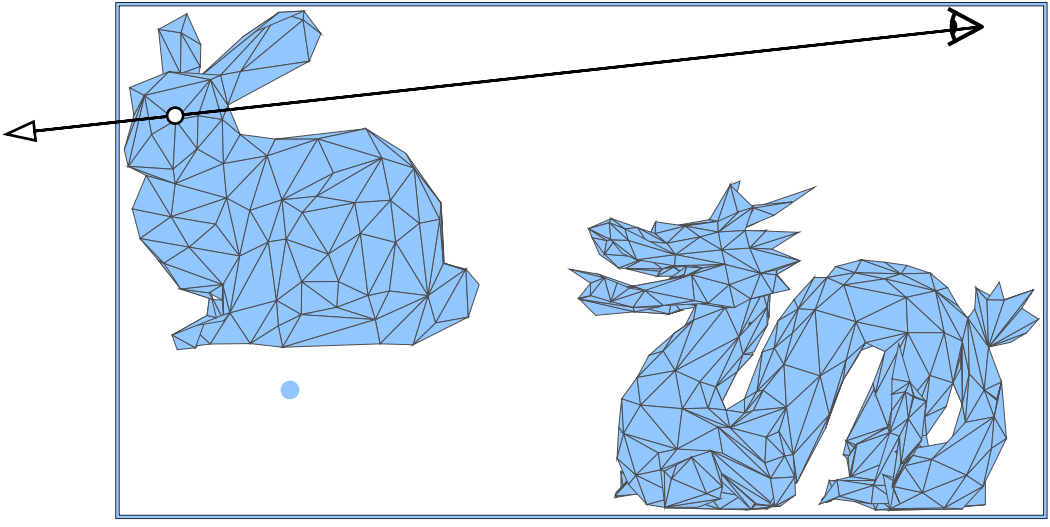
Use `for, if, =, <, &&`



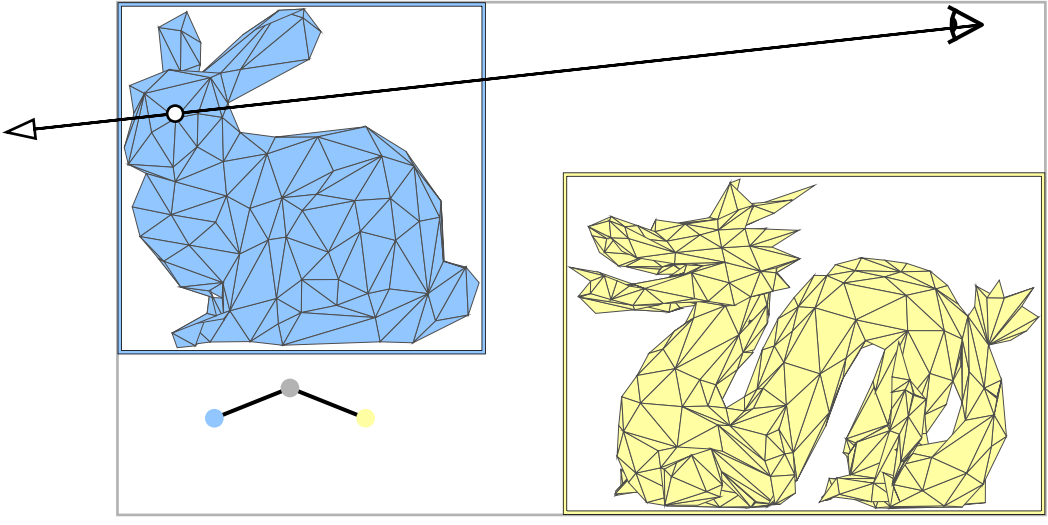
Correct result

shadertoy.com/view/7dKBD3

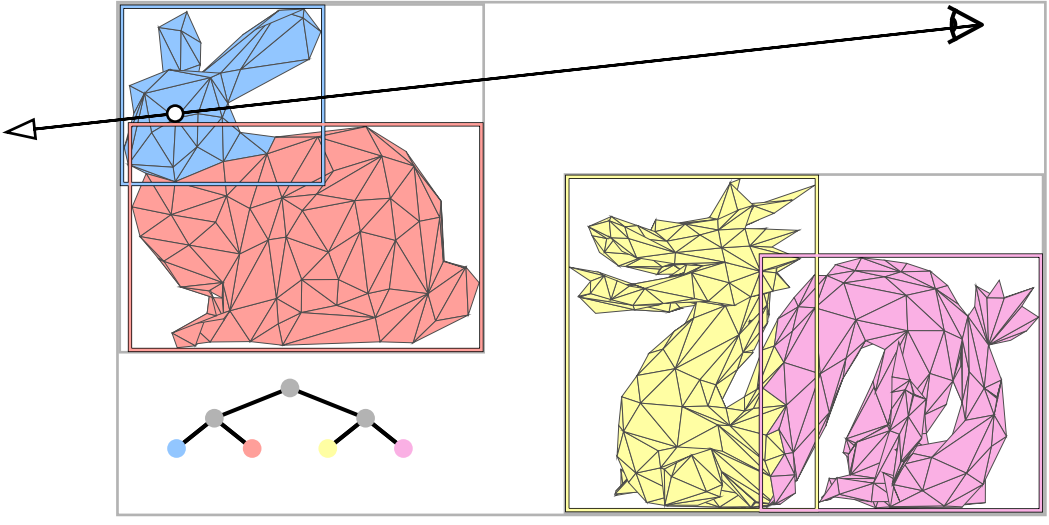
Bounding volume hierarchies (BVH)



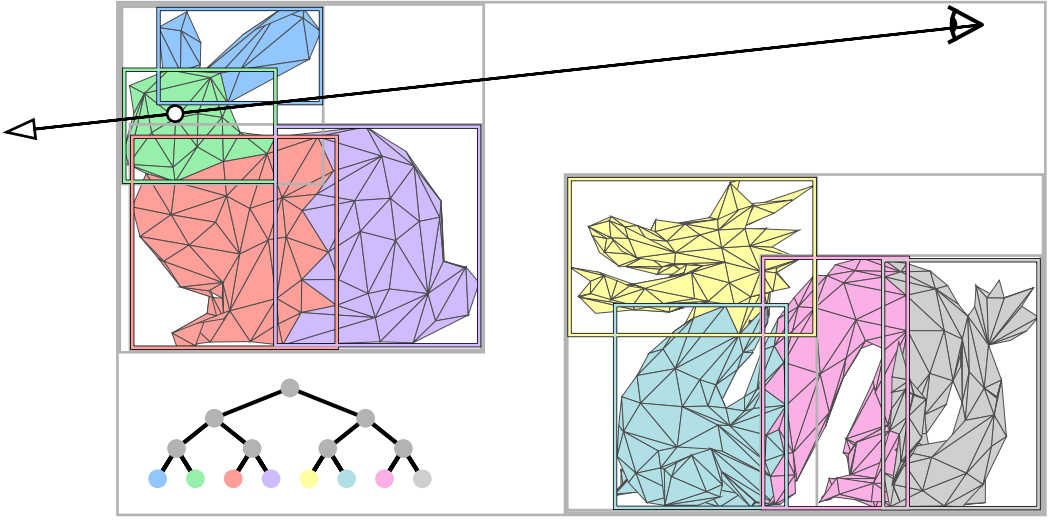
Bounding volume hierarchies (BVH)



Bounding volume hierarchies (BVH)



Bounding volume hierarchies (BVH)



Our goals

Learn rendering basics

Write a path tracer

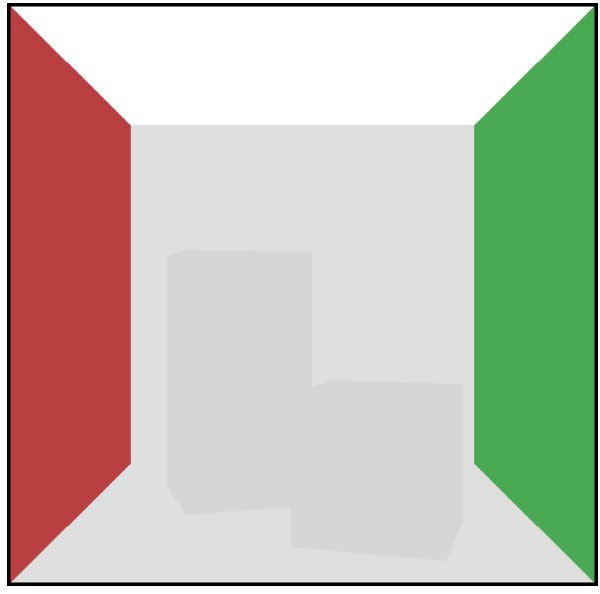
In GLSL on ShaderToy

Have fun

Part 1: Ray tracing

Part 2: Path tracing

See you in part 2!



Our goals

Learn rendering basics

Write a path tracer

In GLSL on ShaderToy

Have fun

Part 1: Ray tracing

Part 2: Path tracing

See you in part 2!