

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)

```
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,lr}*s,*best,sph[]={0.,6.,.5,1.,1.,1.,.9,
.05,.2,.05,0.,1.7,.0.,.8,.5,1.,.0.,.2,1.,.7,.3,0.,.05,1.2,1.,.8.,-.5.,1.,.8,.8,
1.,.3,.7,0.,0.;1.2,3.,-6.,.5,1.,.8,1.,7.,0.,0.,.6,1.5,-3.,-3.,12.,.8,1.,
1.,5.,0.,0.,0.,.5,.5,);}x;double u,v,tmin,u=1.,v=1.,t=1.;double vdot(A,B)vec A
,B;{return A.x*B.x+A.y*B.y+A.z*B.z;}vec vcomb(s,A,B)double s;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;while(s-->0){sph}b=vdot(D,U-vcomb(-1.,P,s->cen)),u=b*b-vdot(U,U)/s->rad*s
->ad,u=u>0?sqrt(u):1e31;u=b-u*1e-7;b=ur+u,tmin=u>1e-7&&u<tmin?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->lr;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;while(l-->0){sph}if((e=1
->kl*vdot(N,C=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;eta=1-eta*
eta*(1-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(1-d,N,D)),vcomb(s->kd,
color,vcomb(s->kl,U,black)));}main(){printf("kd kd\n",.32);while(yx<32*32)
U.x=yx*32-32/2,U.z=32/yx+32,U.y=32/x*atan(25/114.59155226);U=vcomb(255.,
trace(3,black,vunit(U)),black),printf("%0f %0f %0f\n",U);/*pixar!*/
```

A tiny ray tracer

Paul Heckbert's business card (1987)

```
typedef struct {double x,y,z;vec v;vec U,black,amb={.02,.02,.02};}struct sphere{
vec cen,color;double rad,kd,ks,kt,kl,lr}*s,*best,sph[]={0.,6.,.5,1.,1.,.9,
.05,.2,.05,0.,1.7,0.,8.,.5,1.,.0,0.,2,1.,.7,.3,0.,.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.,0.,.6,1.5,-3.,-3.,12.,.8,1.,
2.,5,0.,0.,0.,.5,1.5,},yx;double u,v,tmin,v;double vdot(A,B){vec A,B;
return A.x*B.x+A.y*B.y+A.z*B.z;}vec vcomb(s,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 vnunit(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;while(s-->sph)b=vdot(D,U=vcomb(-1.,P,s->cen)),u=b*b-vdot(U,U)/s->rad*s
->ad,u>0?sqrt(u):1e31;u=b-u*le-7;b=ur+u,tmin=u>le-7&&u<tmin?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->lr;d=-vdot(D,N=vunit(vcomb(-1.,P=vcomb(tmin,D,P),s->cen
)));if(d<0)N=vcc-b(-1.,N,black),eta=1/eta,d=-d;l=s+5;while(l-->sph)if((e=l
->kl*vdot(N,C=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;eta=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(1*d,M,D)),vcomb(s->kd,
color,vcomb(s->kl,U,black))););main(){printf("4d 4d\n",32,32);while(yx<32*32)
U.x=yx*32-32/2,U.z=32/yx+1/2,U.y=32/atan(25/114.591552026);U=vcomb(255,
trace(3,black,vunit(U)),black),printf("%0f %0f %0f\n",U);}/*pixar/pt/
```



Our goals

Learn rendering basics

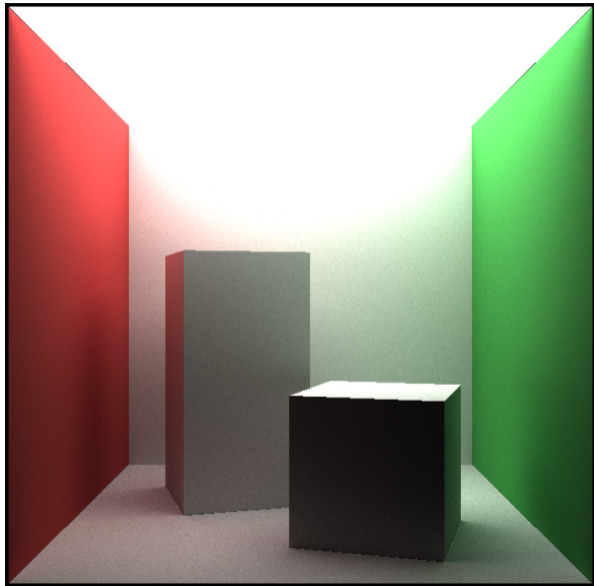
Write a path tracer

In GLSL on ShaderToy

Have fun

Part 1: Ray tracing

Part 2: Path tracing



Our goals

Learn rendering basics

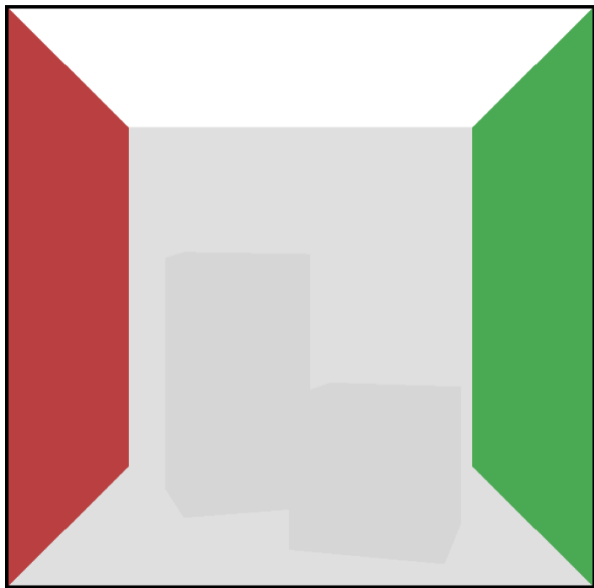
Write a path tracer

In GLSL on ShaderToy

Have fun

Part 1: Ray tracing

Part 2: Path tracing



Our goals

Learn rendering basics

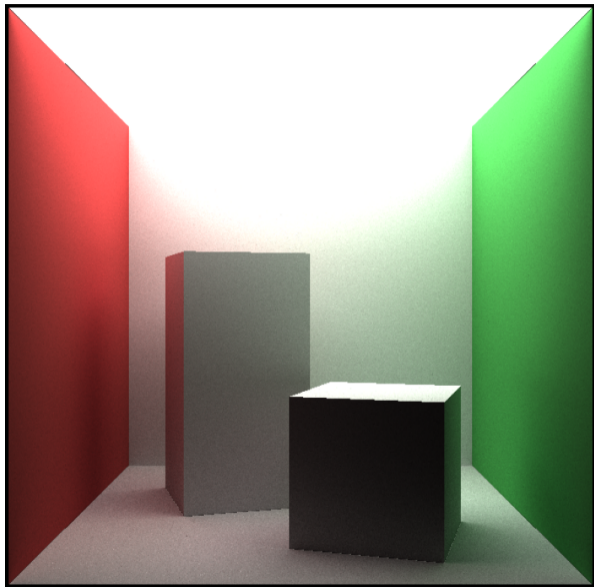
Write a path tracer

In GLSL on ShaderToy

Have fun

Part 1: Ray tracing

Part 2: Path tracing



Not our goals

Framework provided (~60 lines)

Scene provided

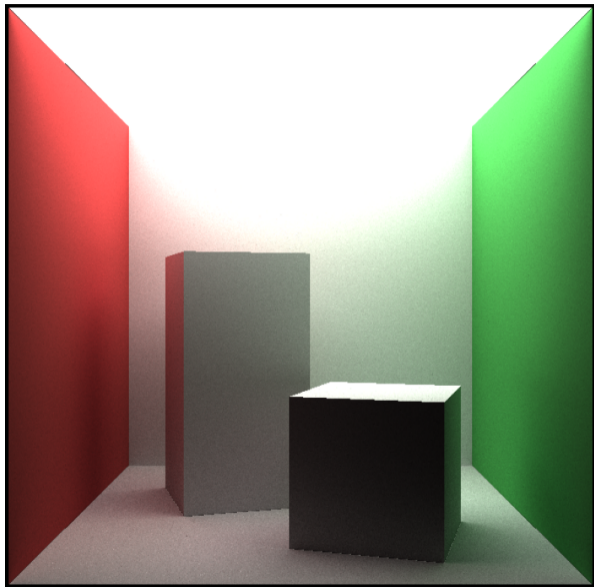
Only diffuse triangles

No textures

No acceleration structures

No clever sampling/denoising

GPUs but no graphics APIs



Not our goals

Framework provided (~60 lines)

Scene provided

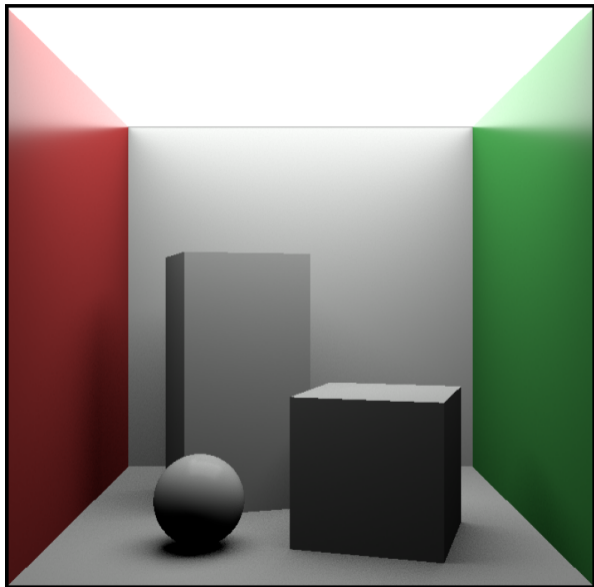
Only diffuse triangles

No textures

No acceleration structures

No clever sampling/denoising

GPUs but no graphics APIs



Not our goals

Framework provided (~60 lines)

Scene provided

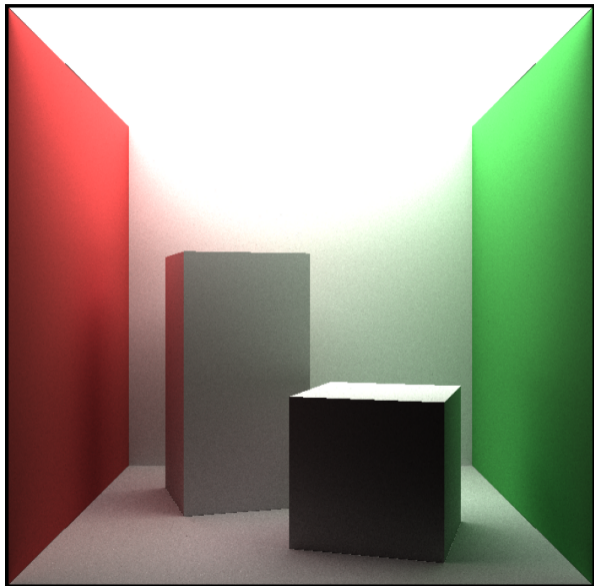
Only diffuse triangles

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:

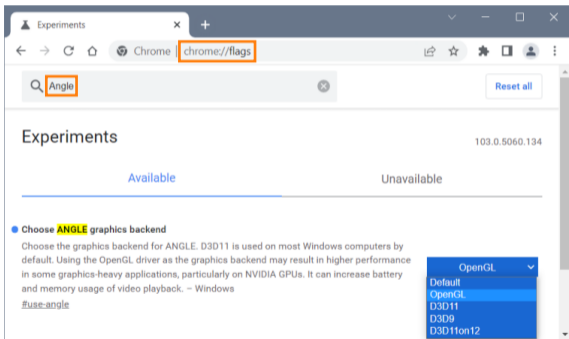
- Browser

- Experience with C-like languages

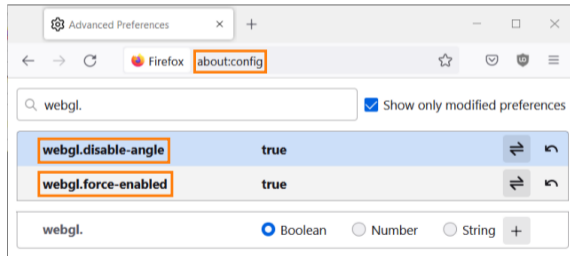
- Basic vector math (e.g. dot products, linear equations)

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

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

```
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;
}
```

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];
```

GLSL Help ✕


This help only covers the parts of GLSL ES that are relevant for Shadertoy. For the complete specification please have a look at **GLSL ES specification**

Language:

- **Version:** WebGL 2.0
- **Arithmetic:** () + - ! * / %
- **Logical/Relational:** ~ < > <= >= == != && ||
- **Bit Operators:** & ^ | << >>
- **Comments:** // /* */
- **Types:** void bool int uint float vec2 vec3 vec4 bvec2 bvec3 bvec4 ivec2 ivec3 ivec4 uvec2 uvec3 uvec4 mat2 mat3 mat4 mat?x? sampler2D, sampler3D, samplerCube
- **Format:** float a = 1.0; int b = 1; uint i = 1U; int i = 0x1;
- **Function Parameter Qualifiers:** [none], in, out, inout
- **Global Variable Qualifiers:** const
- **Vector Components:** .xyzw .rgba .stpq
- **Flow Control:** if else for return break continue switch/case
- **Output:** vec4 fragColor
- **Input:** vec2 fragCoord
- **Preprocessor:** # #define #undef #if #ifdef #ifndef #else #elif #endif #error #pragma #line

Built-in Functions:

<ul style="list-style-type: none">• type radians (type degrees)• type degrees (type radians)• type sin (type angle)• type cos (type angle)• type tan (type angle)• type asin (type x)• type acos (type x)• type atan (type y, type x)	<ul style="list-style-type: none">• vec4 texture(sampler?, vec? coord [, float bias])• vec4 textureLod(sampler, vec? coord, float lod)• vec4 textureLodOffset(sampler? sampler, vec? coord, float lod, ivec? offset)• vec4 textureGrad(sampler?, vec? coord, vec2 dPdx, vec2 dPdy)• vec4 textureGradOffset sampler? , vec? coord, vec? dPdx, vec? dPdy, vec? offset)• vec4 textureProj(sampler?, vec? coord [, float bias])• vec4 textureProjLod(sampler?, vec? coord, float lod)• vec4 textureProjLodOffset(sampler?, vec? coord, float lod, vec? offset)
--	--

▶ Compiled in 0.0 secs 3958 chars S ↵ ? 

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

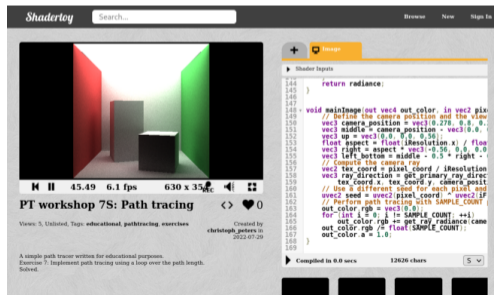
ShaderToy

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)



The screenshot shows the ShaderToy interface. On the left, a 3D rendered scene titled "PT workshop 7S: Path tracing" is displayed. The scene features a red wall on the left, a green wall on the right, and a grey floor with a small grey cube in the center. The interface includes a search bar, a "Browser" button, and a "Sign In" button. Below the render, there are playback controls (play, pause, stop) and performance metrics: "45.49 6.1 fps 630 x 350". The title "PT workshop 7S: Path tracing" is followed by a share icon and a heart icon. Below the title, it says "Views: 5, Unlisted, Tags: educational, pathtracing, exercises" and "Created by christoph_peters in 2022-07-29". A small description at the bottom left reads: "A simple path tracer written for educational purposes. Exercise 7. Implement path tracing using a loop over the path length. Solved." On the right side of the interface, there is a code editor showing the shader code. The code is in GLSL and implements a path tracer. It includes comments and a `return radiance;` statement. The code is compiled in 0.0 secs and is 12626 chars long.

ShaderToy

Runs a full-viewport fragment shader in WebGL

Program that runs once per pixel to compute its color

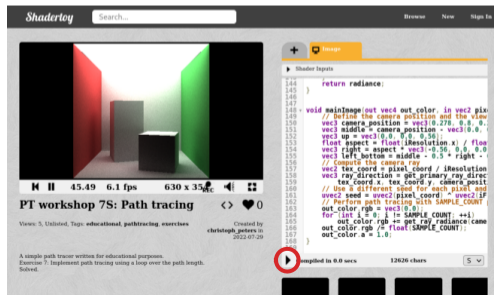
In each exercise you complete 1 function in a ShaderToy (`// TODO`)

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`)

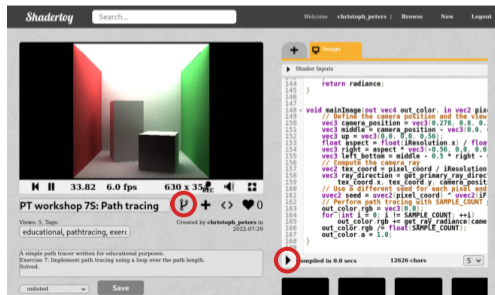
Exercise N+1 has a reference solution for exercise N (no peeking)

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

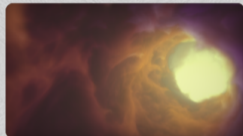
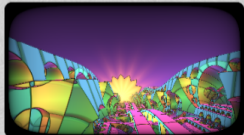
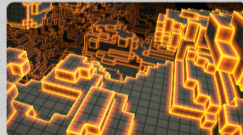


Sort: **Popular** Newest Love Hot

Filter: Multipass GPU Sound VR Microphone Soundcloud Webcam

View: Slideshow

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

Seascape by **TDM** 572647 2277Clouds by **iq** 255953 1684Protean clouds by **nimitz** 87161 1457Raymarching - Primitives by **iq** 624851 1260Rainforest by **iq** 216604 996Flame by **XT95** 142290 982Star Nest by **Kali** 58215 966Creation by Silexars by **Dangu** 437162 945Fractal Land by **Kali** 85468 894expansive reaction-diffusion by **Kali** 4729 846Elevated by **iq** 189183 840Voxel Edges by **iq** 124896 811

Exercise 0: Hello ShaderToy

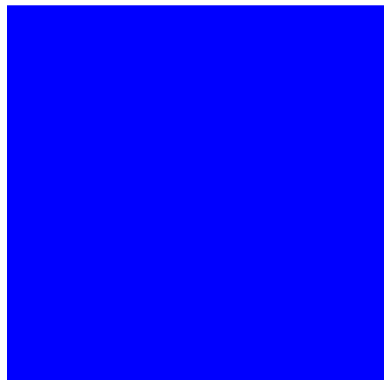
Open the ShaderToy linked below

Change background color from red to blue

Click the play button

See your result

Use `vec3()`



Correct result

Exercise 0: Hello ShaderToy

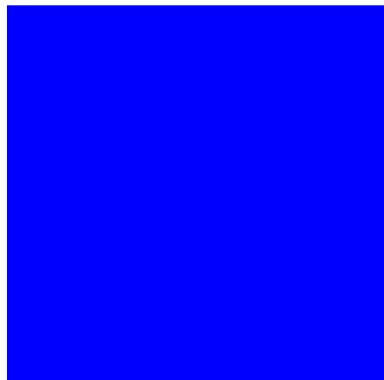
Open the ShaderToy linked below

Change background color from red to blue

Click the play button

See your result

Use `vec3()`



Correct result



Scene representation

Camera

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



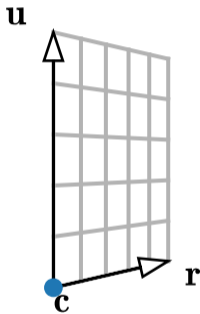
Camera

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 $\mathbf{o} \in \mathbb{R}^3$

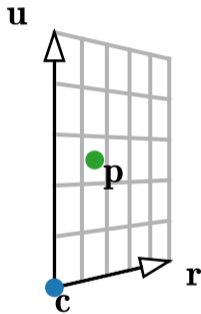
And an image plane, with:

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:

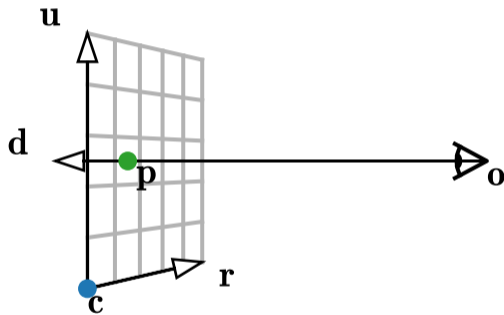
Left bottom corner $\mathbf{c} \in \mathbb{R}^3$

Right and up vectors $\mathbf{r}, \mathbf{u} \in \mathbb{R}^3$

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

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

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



Camera

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

And an image plane, with:

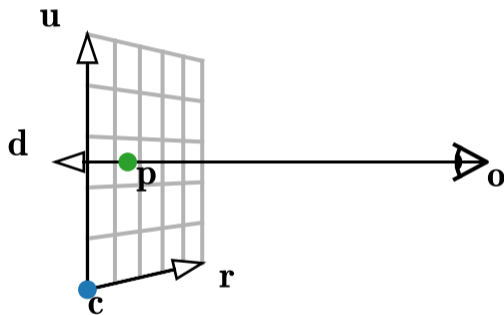
Left bottom corner $\mathbf{c} \in \mathbb{R}^3$

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

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



Triangle mesh

Our scene geometry is a triangle mesh

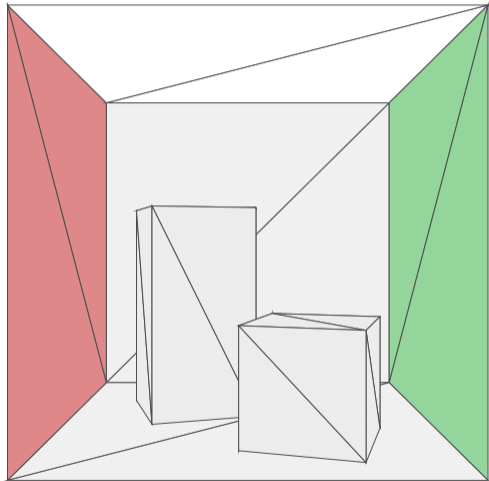
Each triangle has:

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?

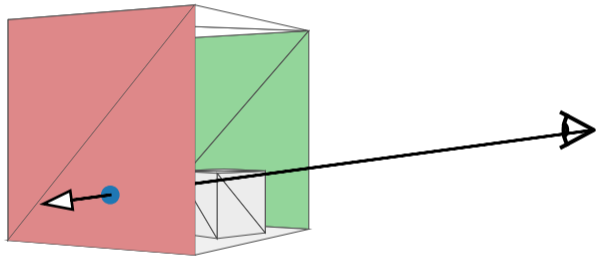
The closest intersected triangle!

Ray tracing finds this closest hit

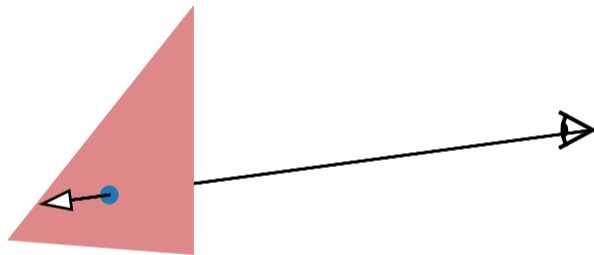
Foundation of path tracing

Implemented in hardware

But we do it in software



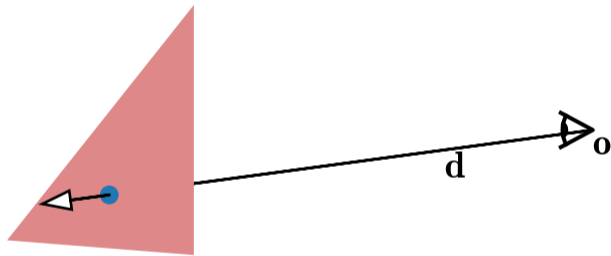
Ray-triangle intersection test



Ray-triangle intersection test

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

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

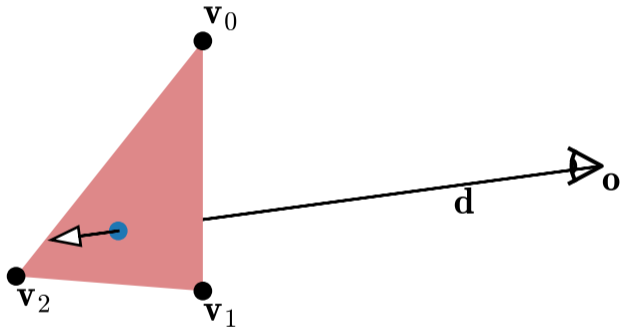


Ray-triangle intersection test

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

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

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



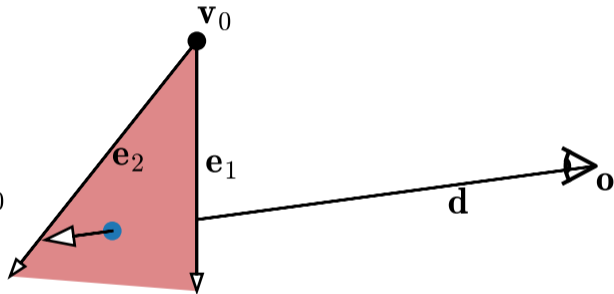
Ray-triangle intersection test

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

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

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}

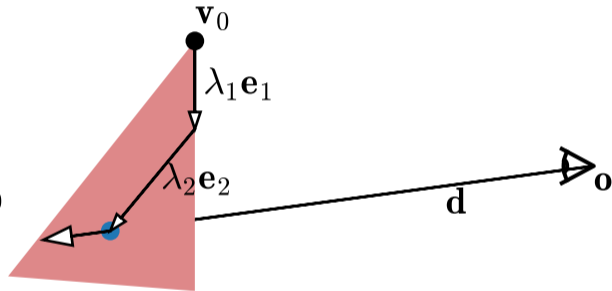
$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d} \text{ with } t \geq 0$$

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}

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

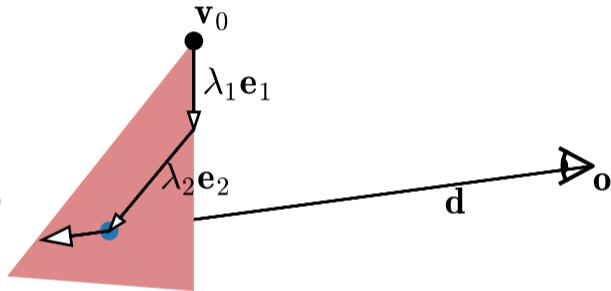
Triangle has vertices $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$

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

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

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

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

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

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

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

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

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

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

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

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

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 $\lambda_1 + \lambda_2 \leq 1$

Exercise 2: Ray-triangle intersection test

Complete `ray_triangle_intersection()`

Inputs: `o`, `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)

Exercise 2: Ray-triangle intersection test

Complete `ray_triangle_intersection()`

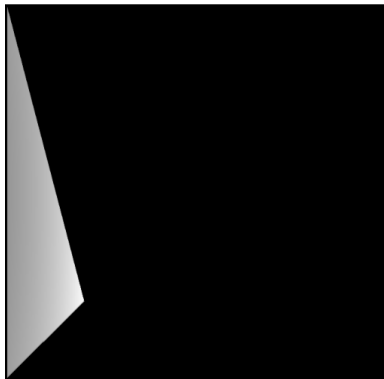
Inputs: `o`, `d`, triangle

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)



Ray-mesh intersection test

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

Test each triangle in a for-loop

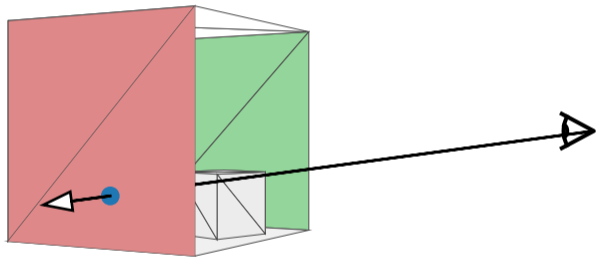
If there is a hit with $t < t_{\min}$:

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()`

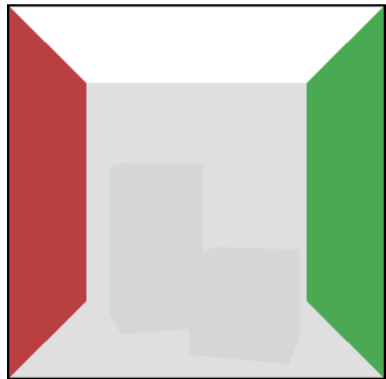
Inputs: `o`, `d`, mesh triangles (defined inline)

Outputs: boolean, t and intersected triangle

Framework draws scene with ray tracing

Use the approach on the previous slide

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



Correct result

Exercise 3: Ray-mesh intersection test

Complete `ray_mesh_intersection()`

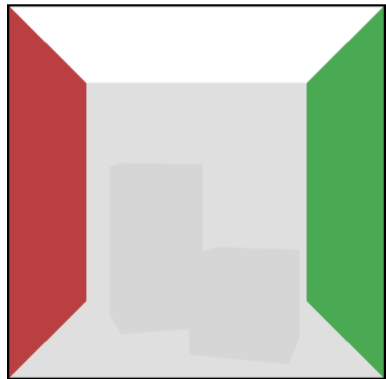
Inputs: `o`, `d`, mesh triangles (defined inline)

Outputs: boolean, t and intersected triangle

Framework draws scene with ray tracing

Use the approach on the previous slide

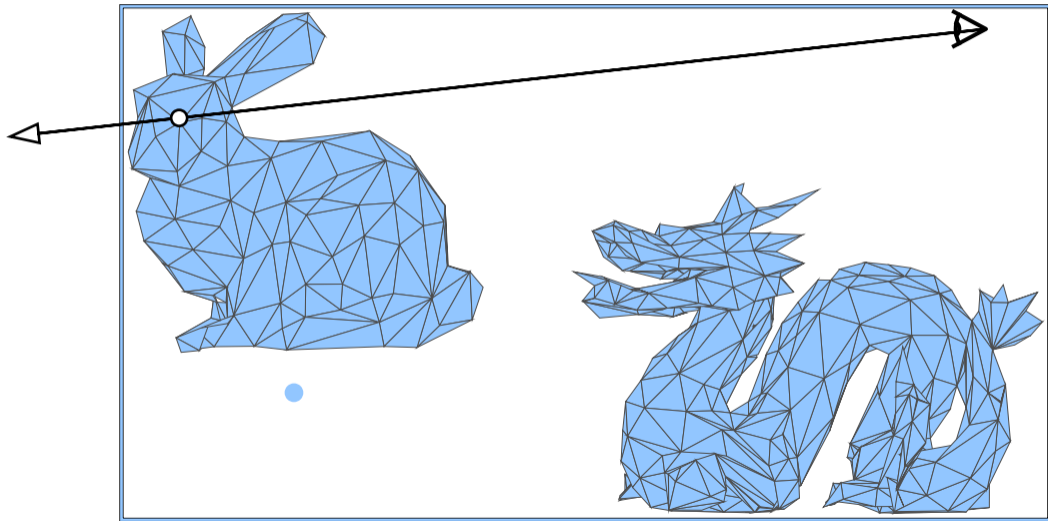
Use `for`, `if`, `=`, `<`, `&&`



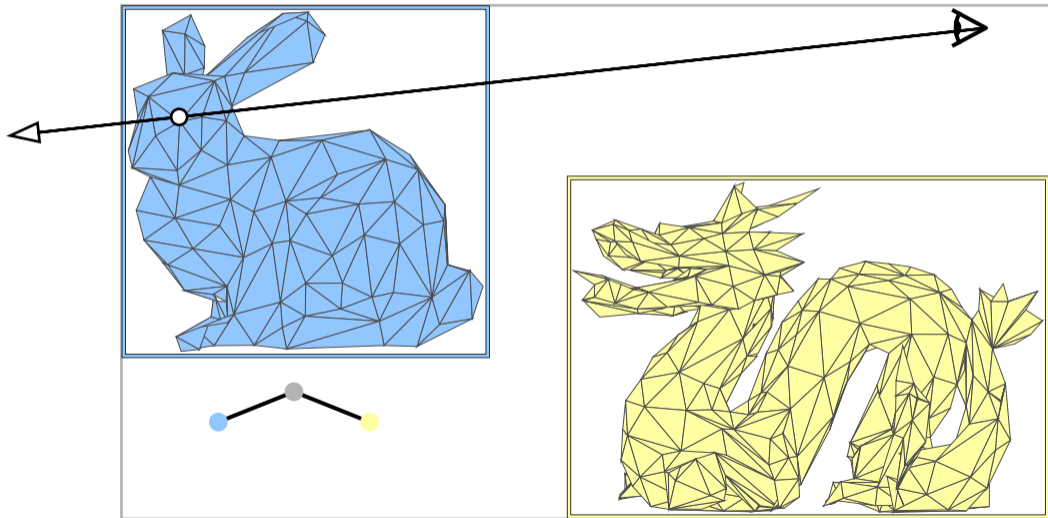
Correct result



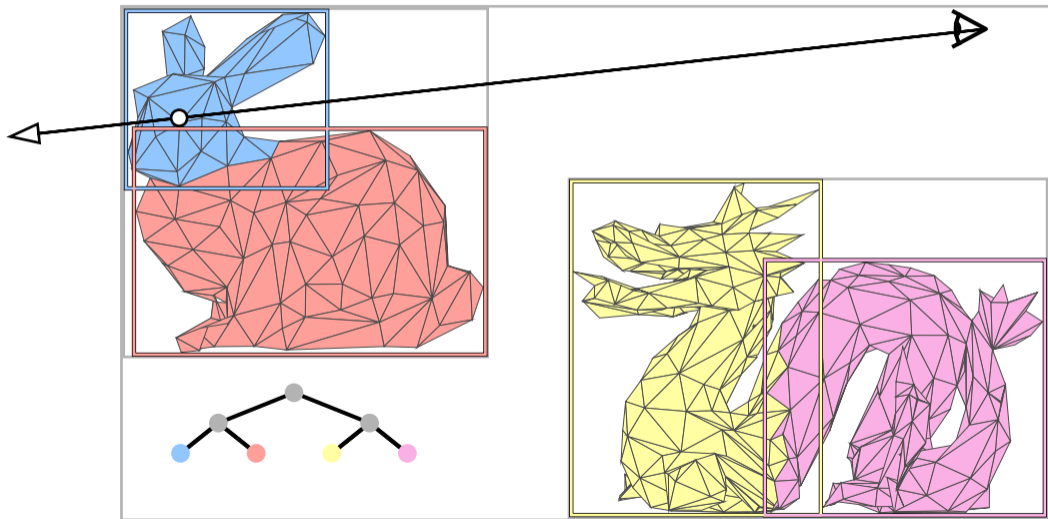
Bounding volume hierarchies (BVH)



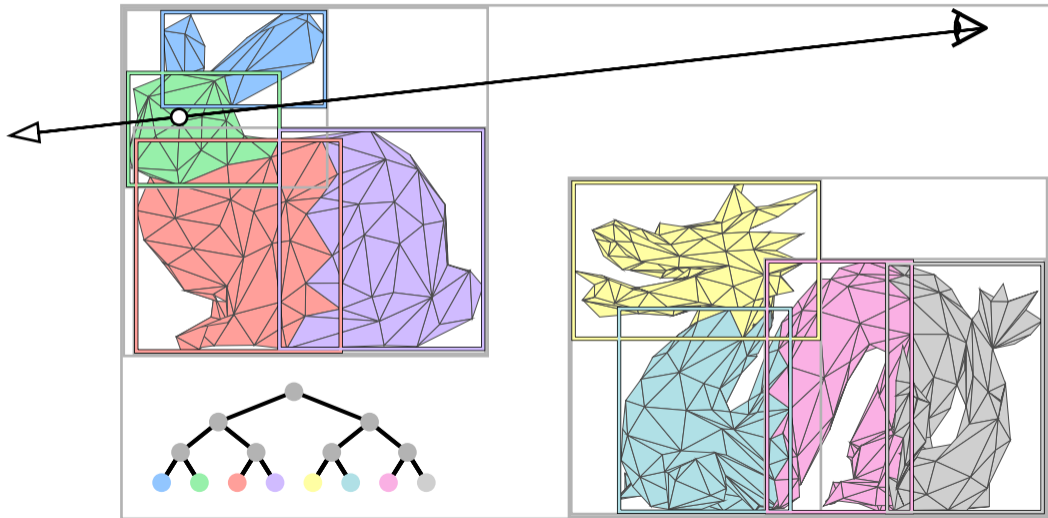
Bounding volume hierarchies (BVH)



Bounding volume hierarchies (BVH)



Bounding volume hierarchies (BVH)



Our goals

Learn rendering basics

Write a path tracer

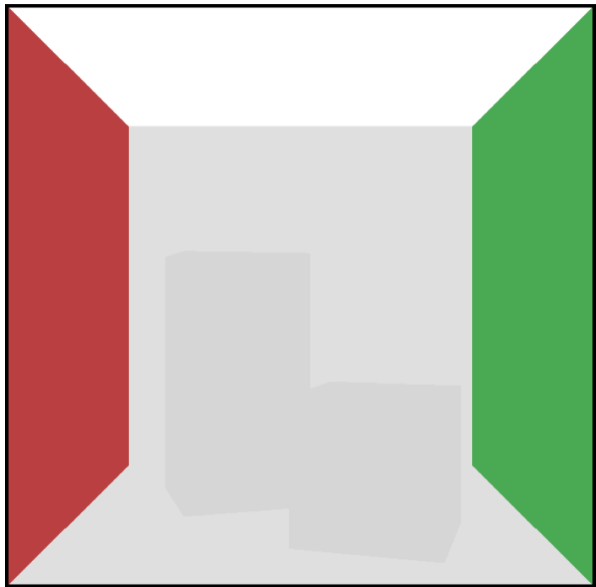
In GLSL on ShaderToy

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!

