Lecture Computer Graphics

Raytracing - Intersections

Prof. Dr. David Bommes
Computer Graphics Group

Part I: Raytracing

Image Source

Part I: Raytracing

- Goal: How to generate realistic digital images of synthetic 3D scenes?

Image Source

Part I: Raytracing

- Goal: How to generate realistic digital images of synthetic 3D scenes?

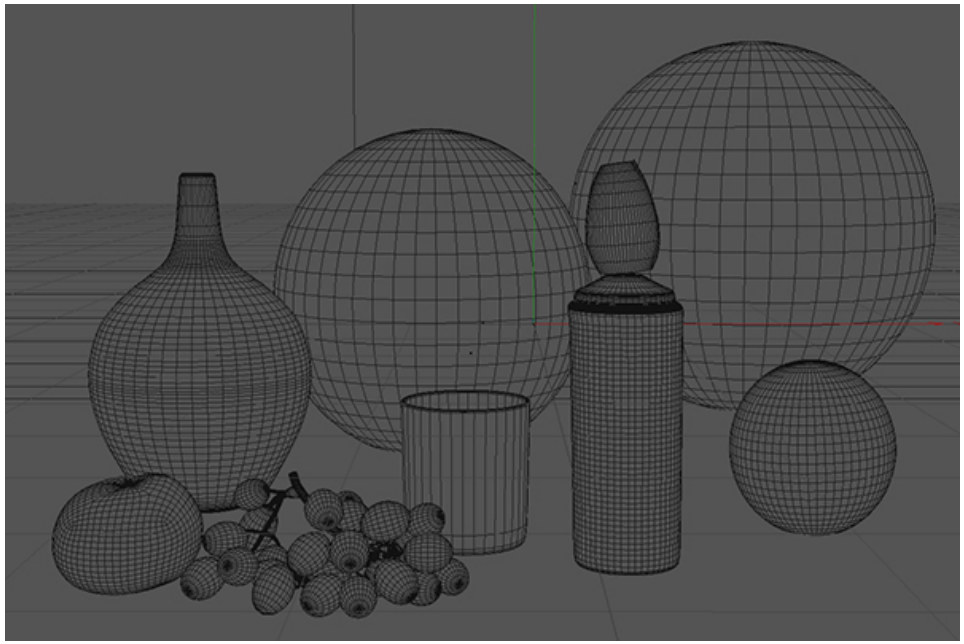


Image Source

Part I: Raytracing

- Goal: How to generate realistic digital images of synthetic 3D scenes?

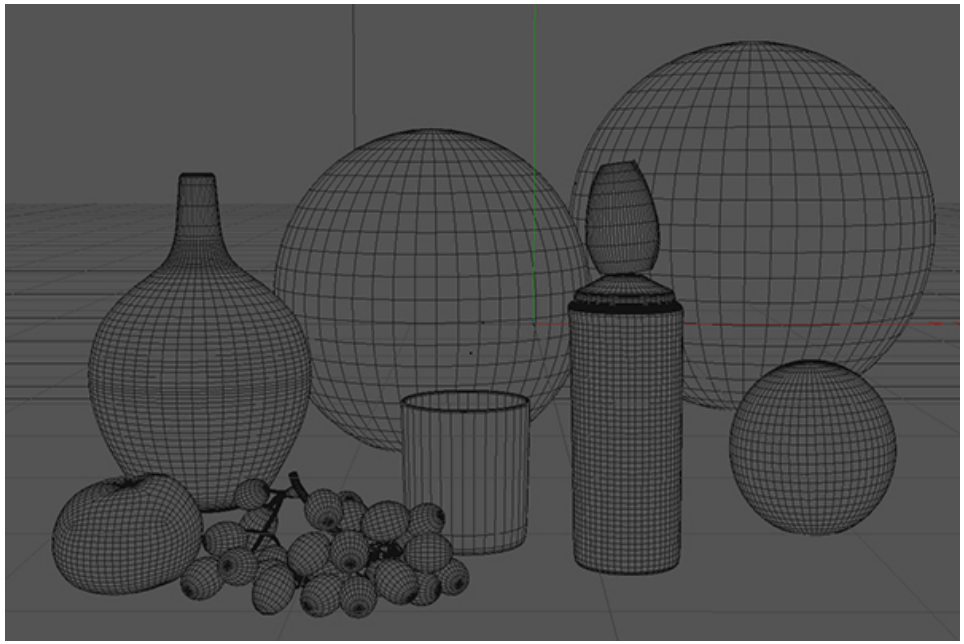
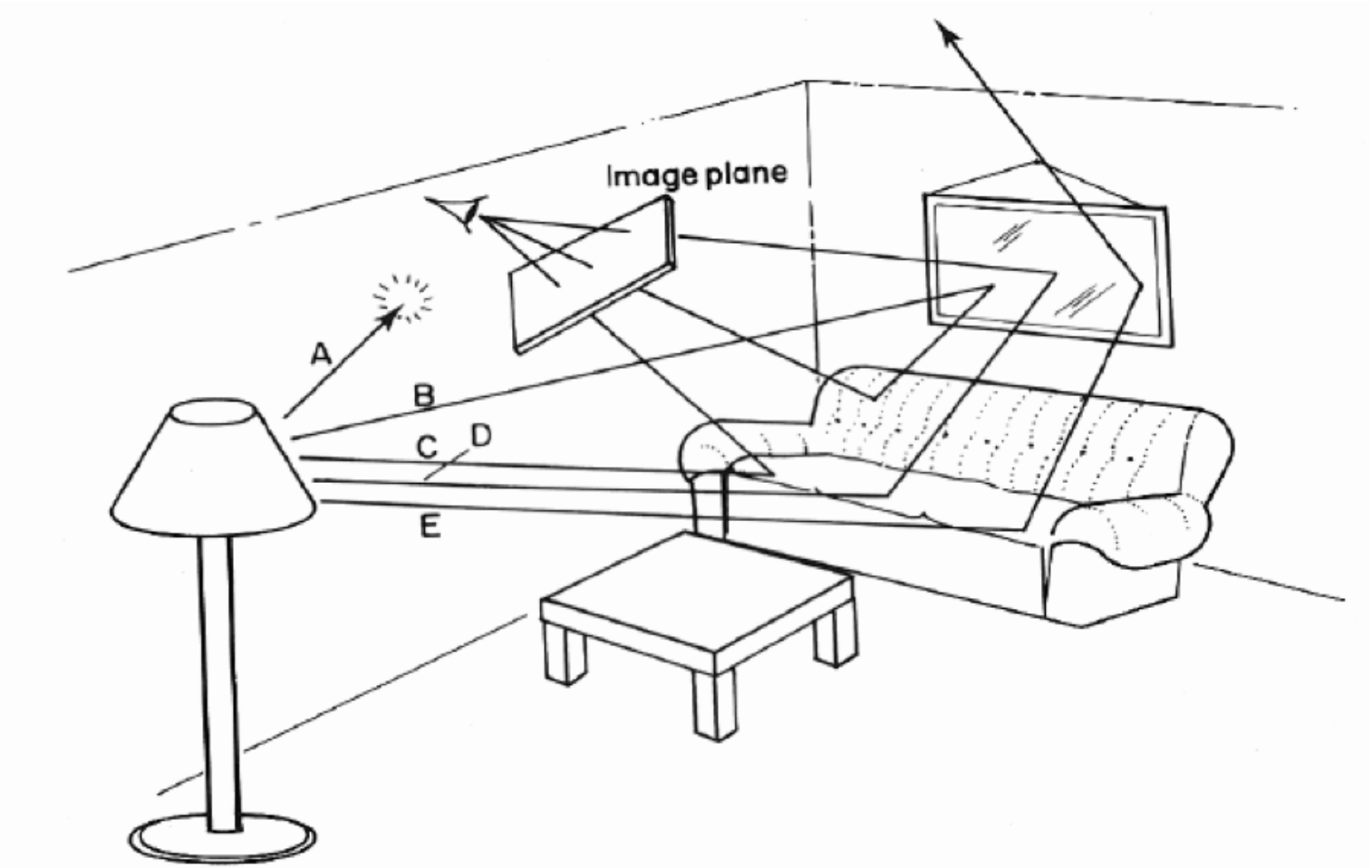


Image Source

Part I: Raytracing

- Goal: How to generate realistic digital images of synthetic 3D scenes?
- Several questions arise:
  - What is light?
  - How is light propagated?
  - How does light interact with objects?
  - How does a camera work?

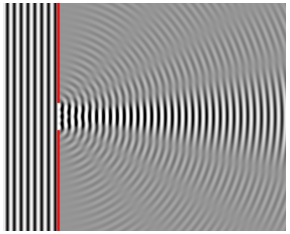
Light Transport



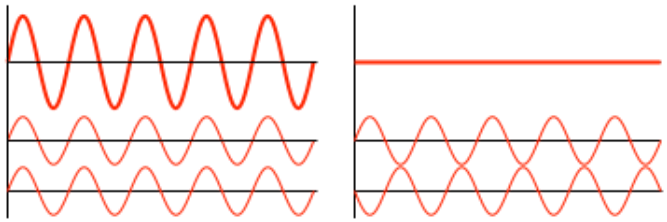
© *Andrew Glassner*

Light Transport Assumptions

- **Geometric optics**
  Light propagates along rays, not waves
  ⇒ no diffraction, no interference, no polarization



diffraction



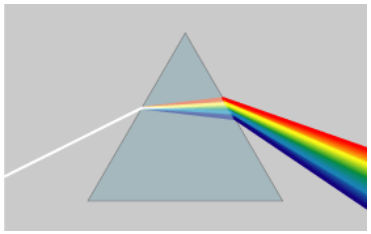
interference

polarization

Images from Wikipedia

Light Transport Assumptions

- **Geometric optics**
  Light propagates along rays, not waves
  ⇒ no diffraction, no interference, no polarization
- **RGB approximation**
  Approximate light spectrum by discrete RGB wavelengths
  ⇒ no dispersion, no fluorescence



dispersion

fluorescence

Images from Wikipedia

Light Transport Assumptions

- **Geometric optics**
  Light propagates along rays, not waves
  ⇒ no diffraction, no interference, no polarization
- **RGB approximation**
  Approximate light spectrum by discrete RGB wavelengths
  ⇒ no dispersion, no fluorescence
- **No participating media**
  Light travels along straight rays through vacuum
  ⇒ no atmospheric scattering, no gravity effects
- **Superposition**
  Simply add multiple light contributions
  ⇒ No nonlinearly reflecting materials

Quiz Setup

http://cgg.unibe.ch:8080

Rendering or Photograph?

cgtrader.com

A: Rendering

A: Photo

Rendering or Photograph?

cgtrader.com

A: Rendering

A: Photo

Rendering or Photograph?

cgtrader.com

A: Rendering

A: Photo

Rendering or Photograph?

cgtrader.com

A: Rendering

A: Photo

Rendering or Photograph?

Autodesk

A: Rendering

A: Photo

Rendering or Photograph?

Autodesk

A: Rendering

A: Photo

Rendering or Photograph?

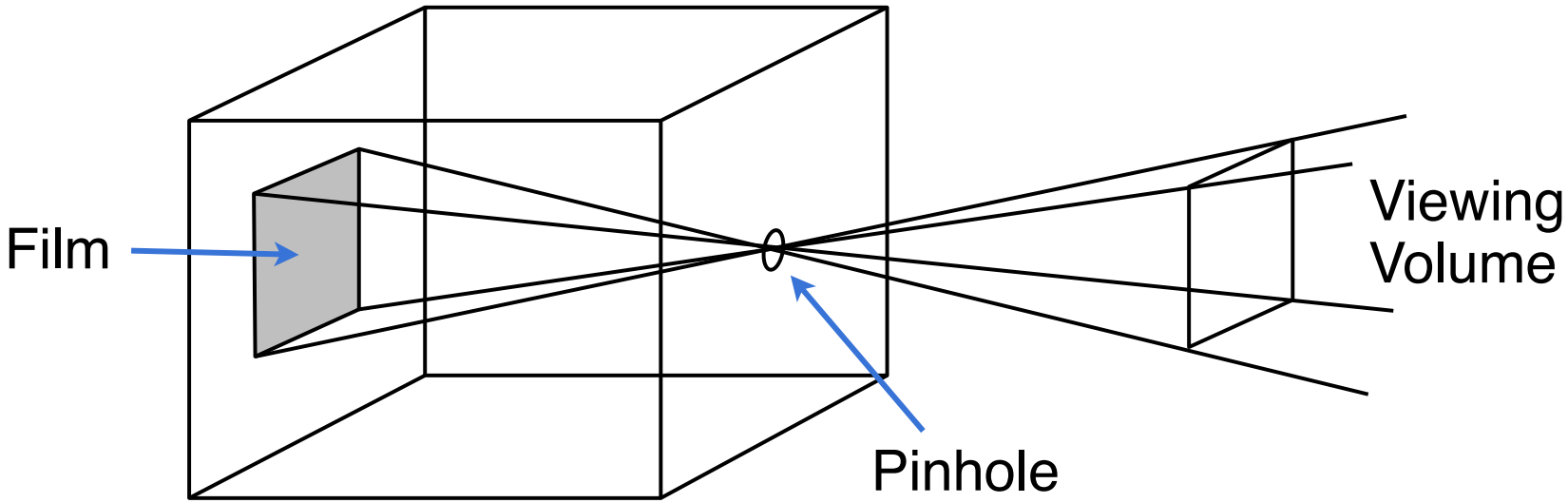
Autodesk

A: Rendering

A: Photo

Pinhole Camera

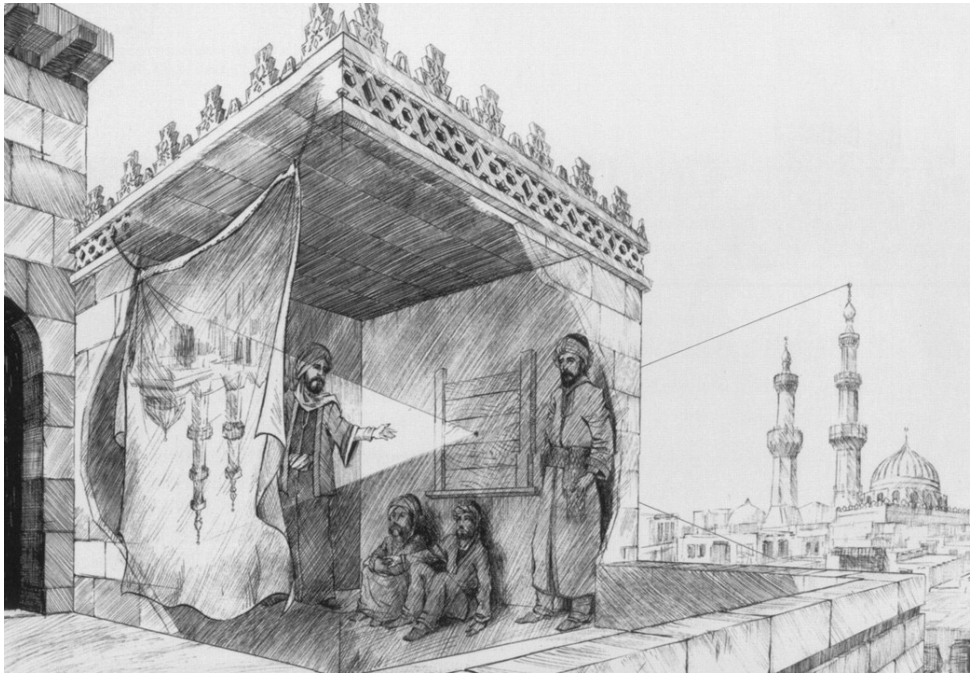
- We assume a classical pinhole camera



Pinhole Camera

- Studied extensively throughout history
- Application: Camera Obscura

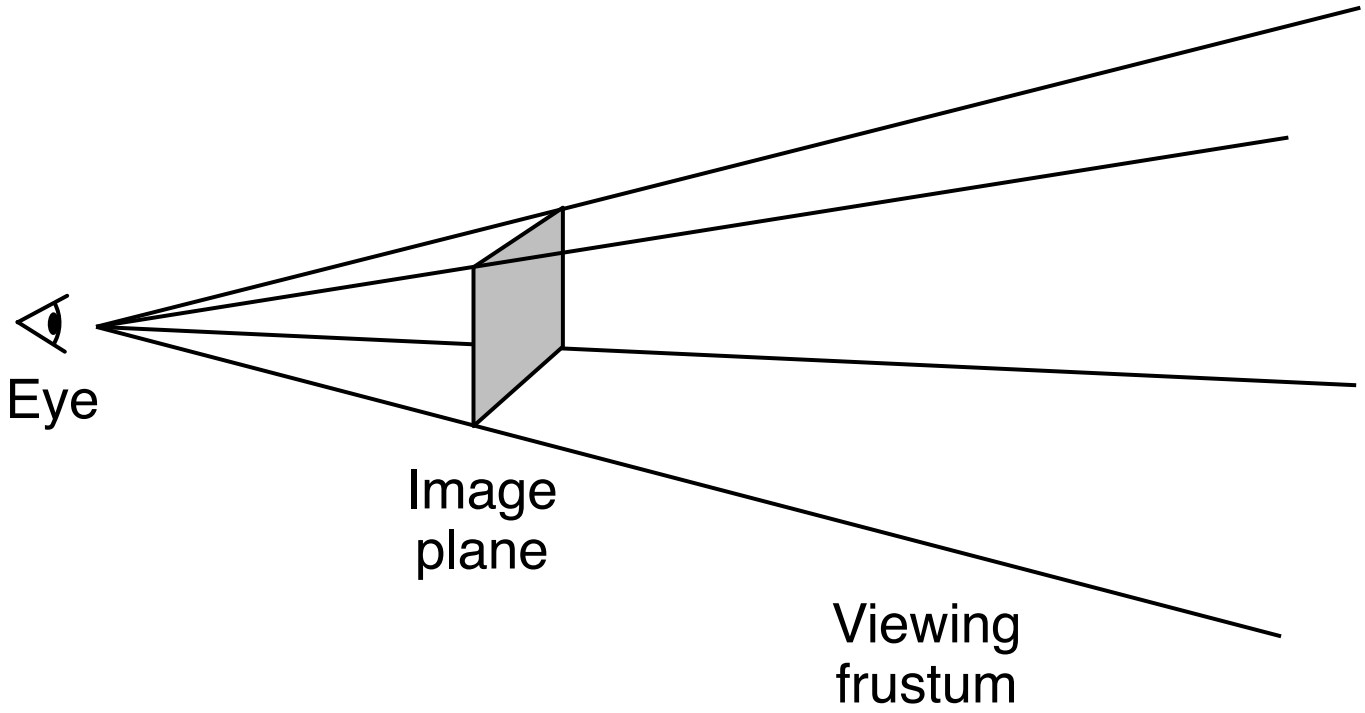
  - Mo Tzu (c. 470–c. 390 BC)
  - Aristotle (384–322 BC)
  - **Ibn al-Haytham (965–1040)**
  - Shen Kuo (1031–1095)
  - Roger Bacon (c. 1214–1294)
  - Johannes Kepler (1571–1630)



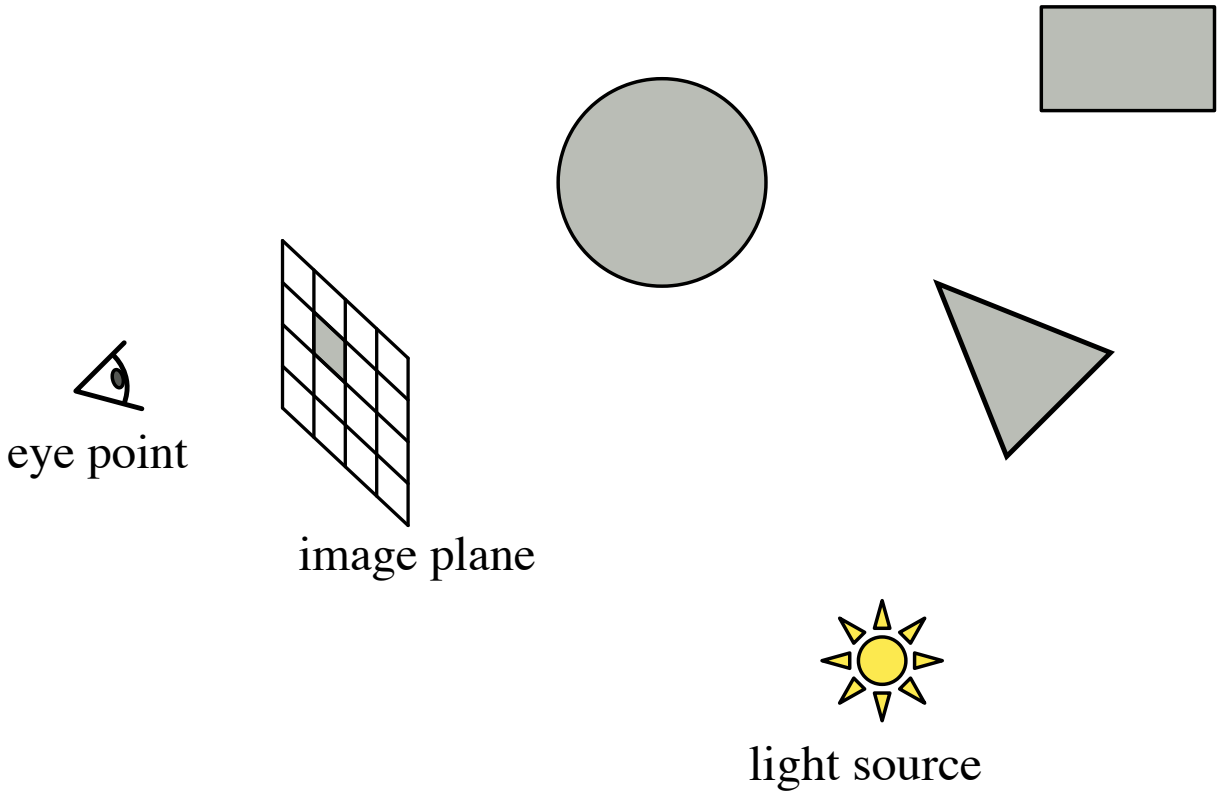
Source: A. H. Zewail,Phil. Trans. R. Soc. A 2010;368:1191-1204

Pinhole Camera

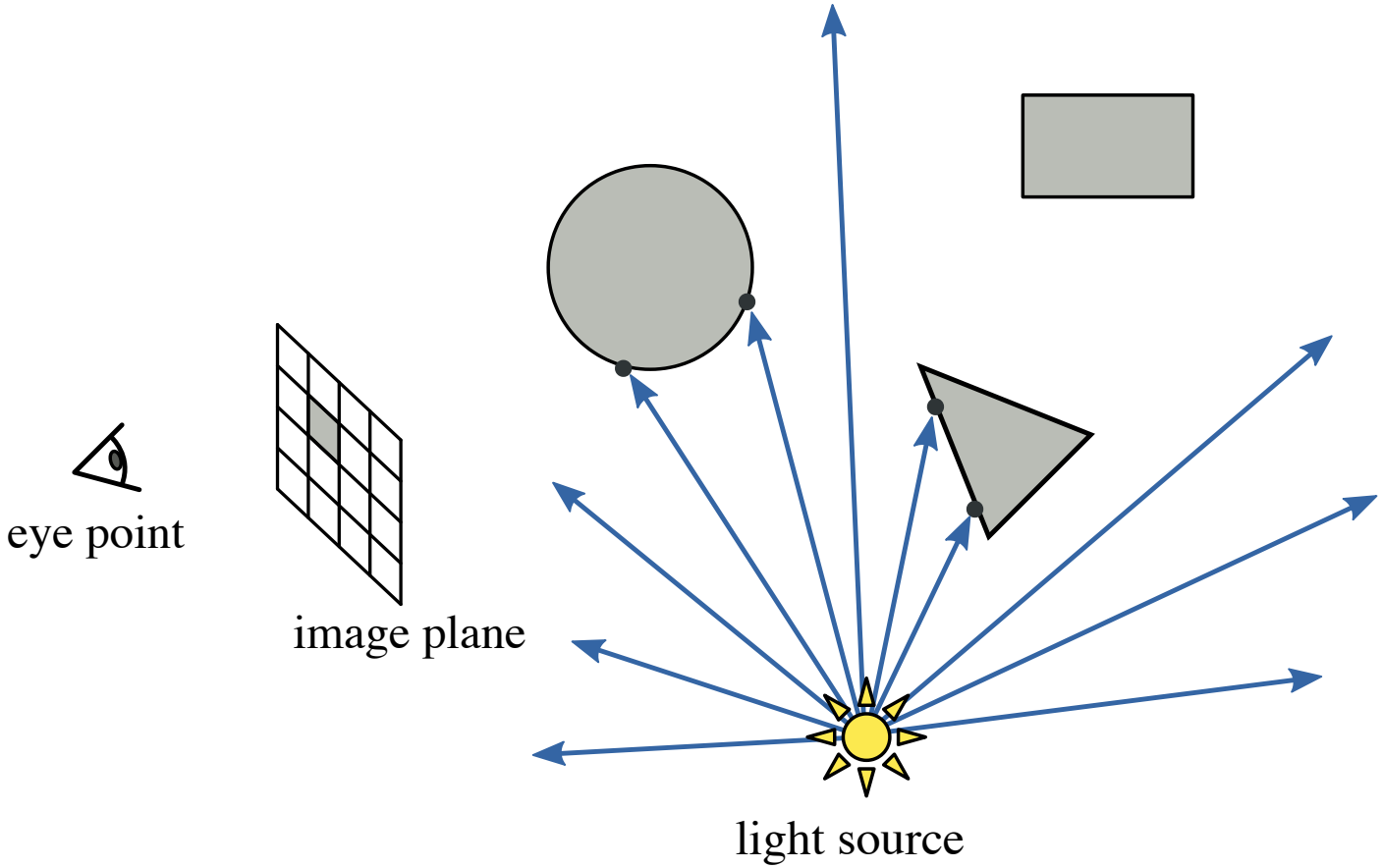
- We represent the camera using eye point (pinhole) and image plane (mirrored film)



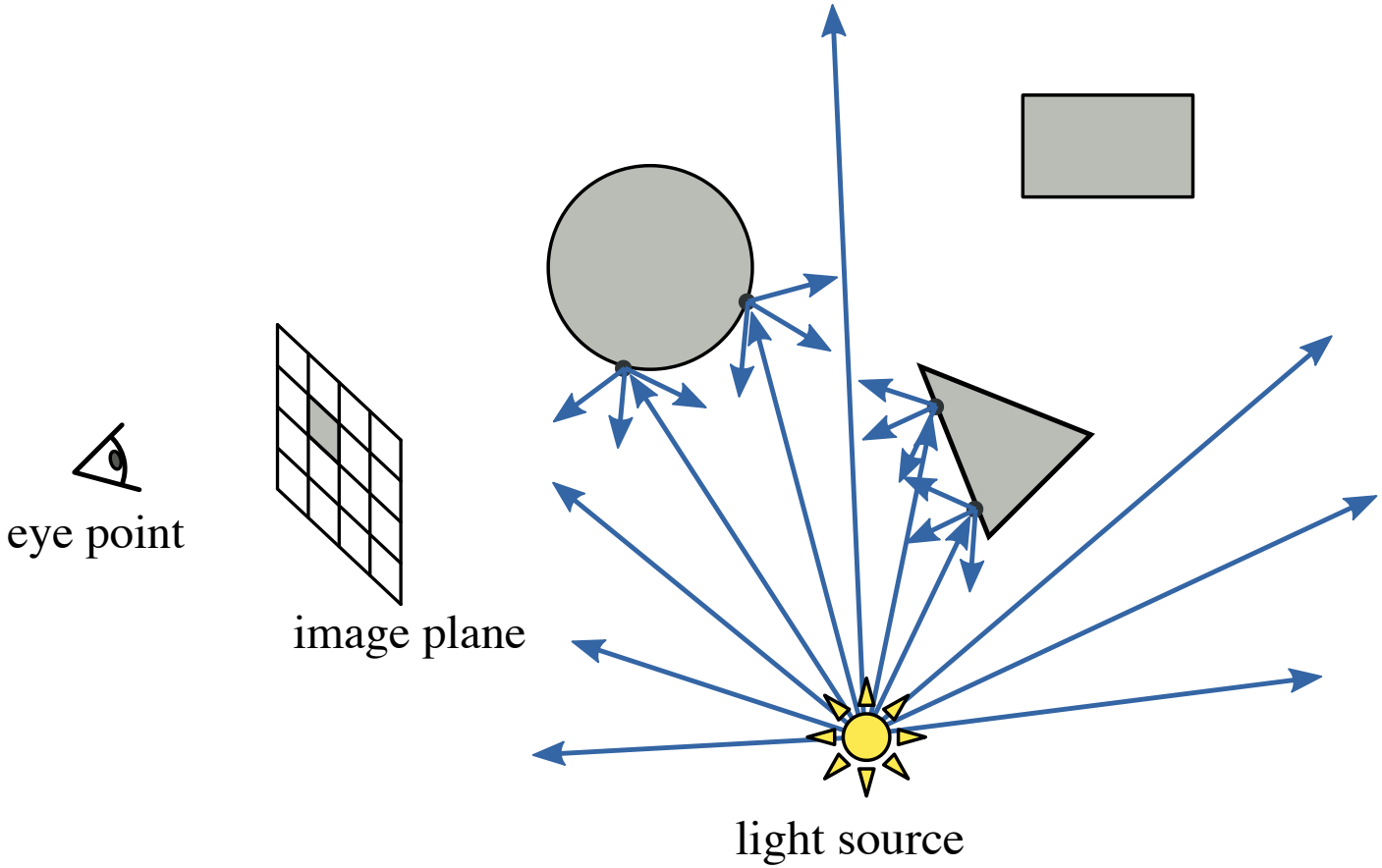
Forward Ray Tracing



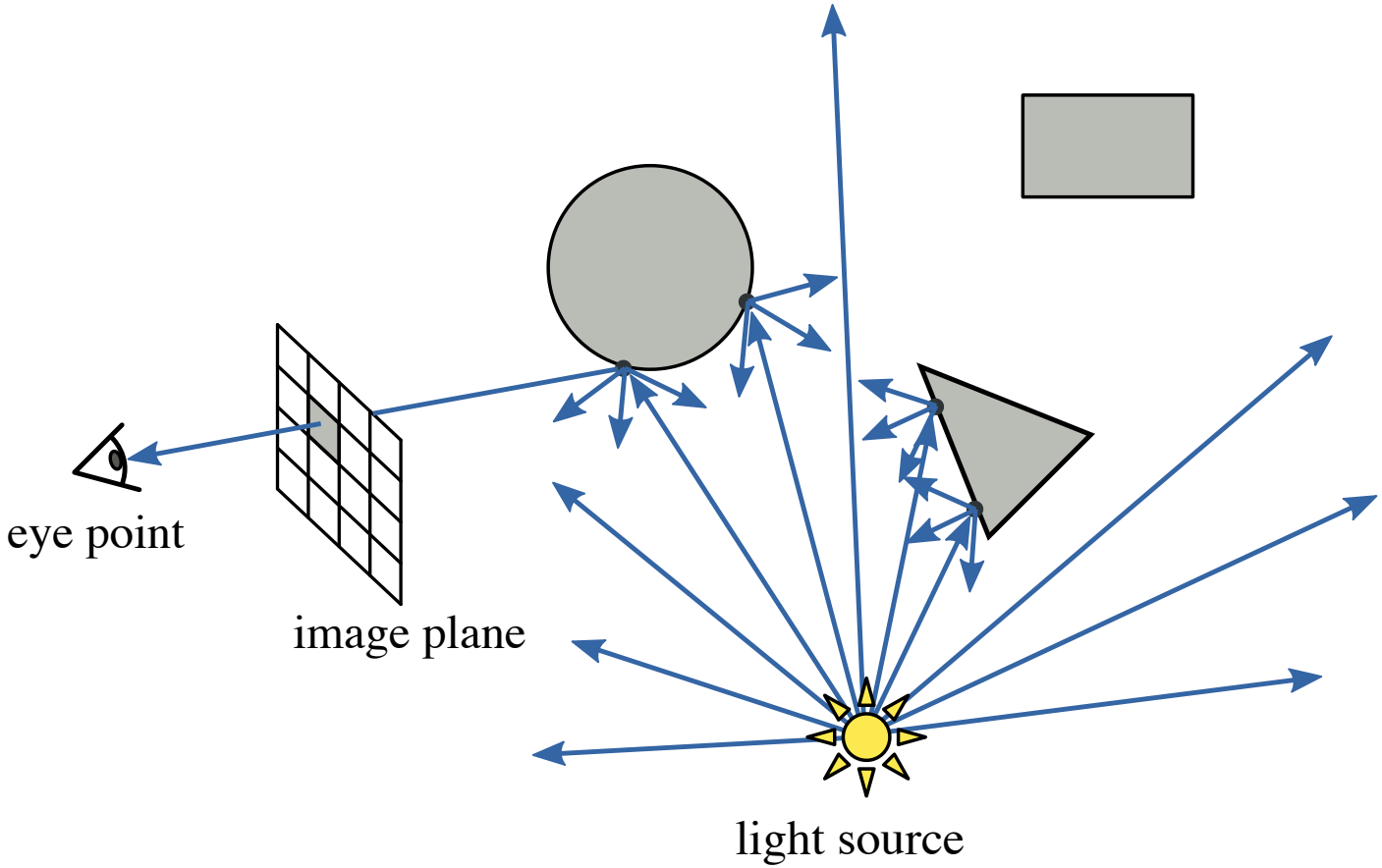
Forward Ray Tracing



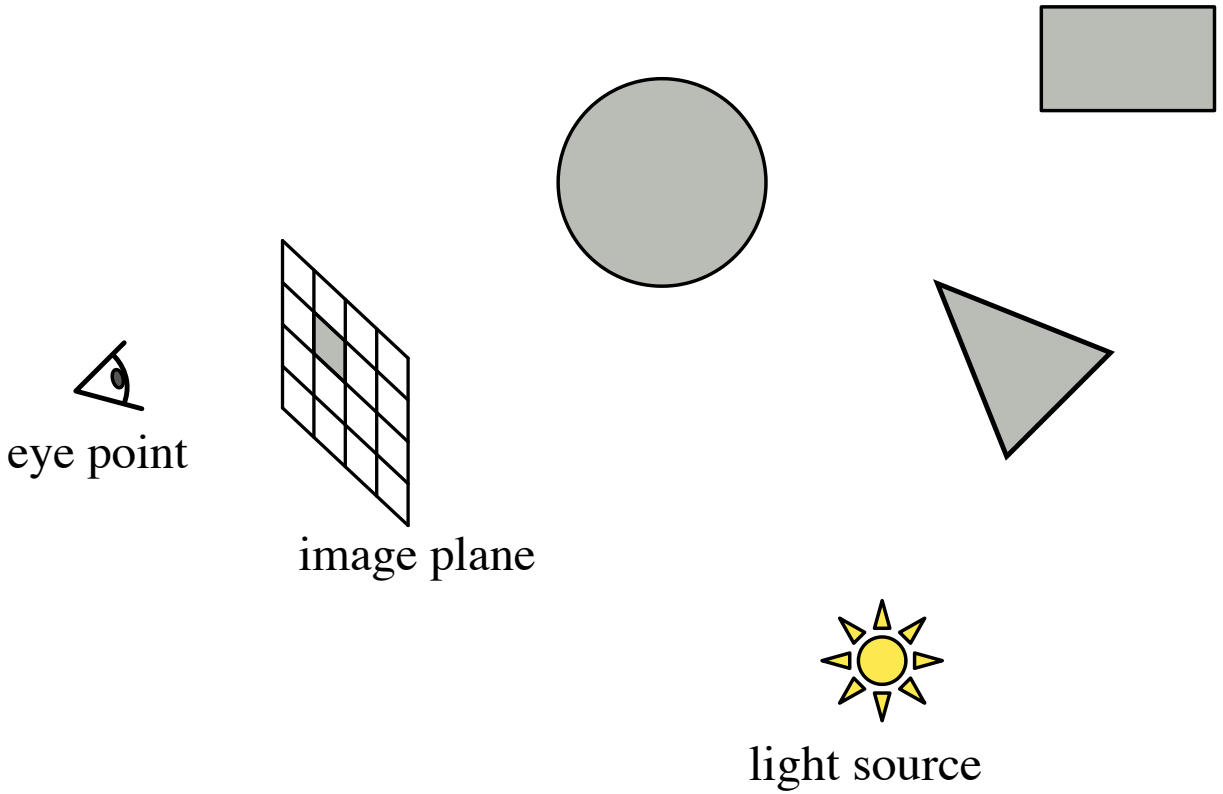
Forward Ray Tracing



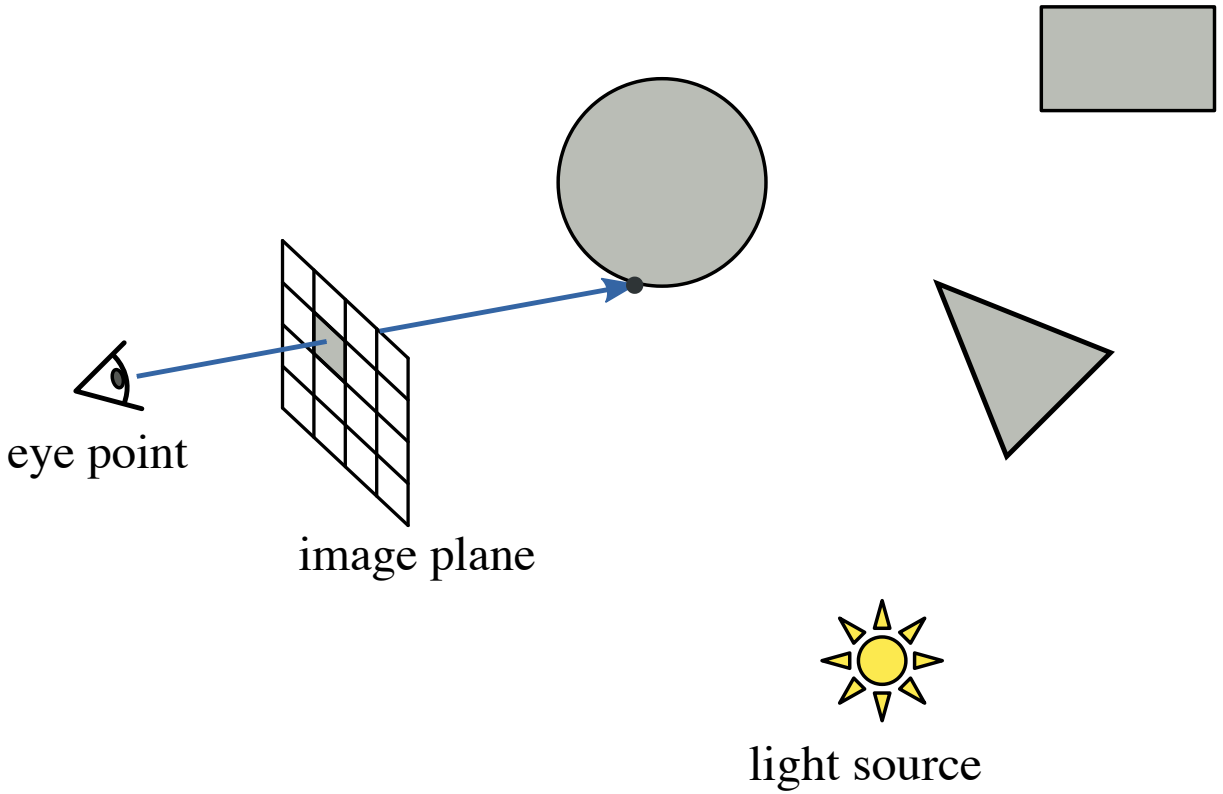
Forward Ray Tracing



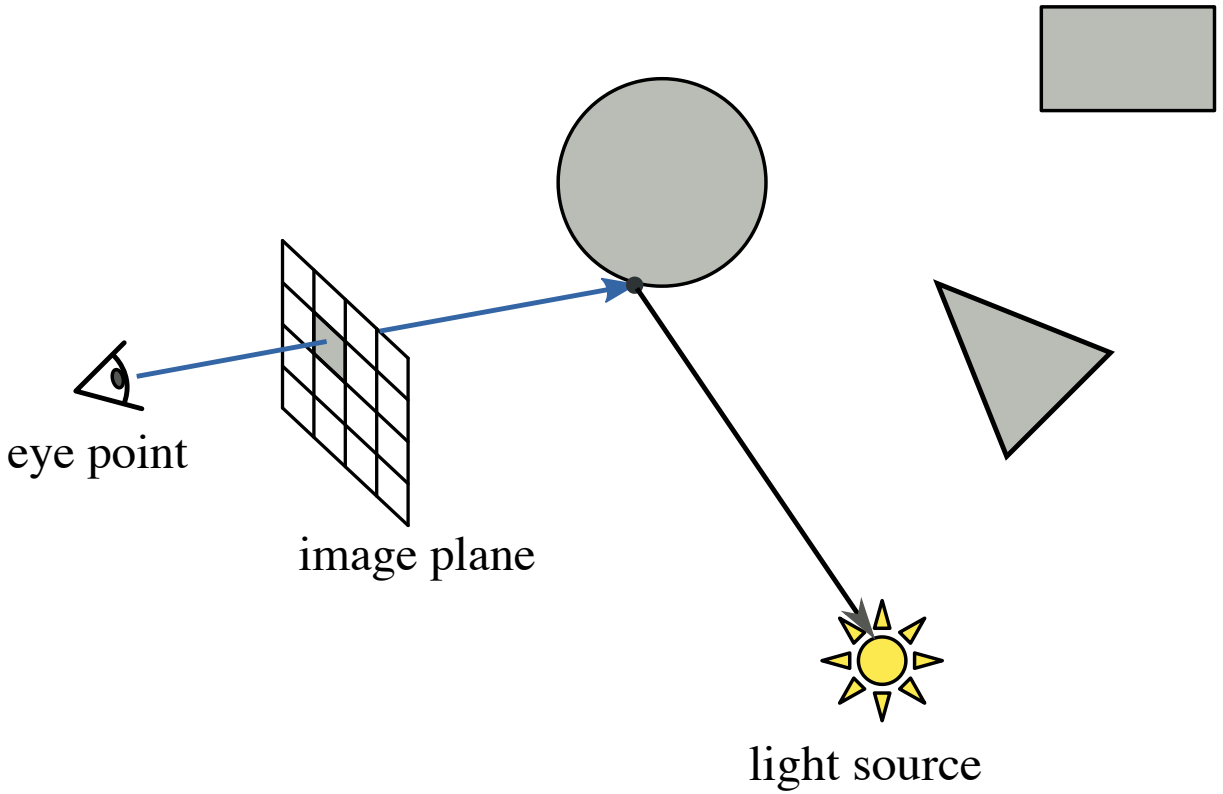
Backward Ray Tracing



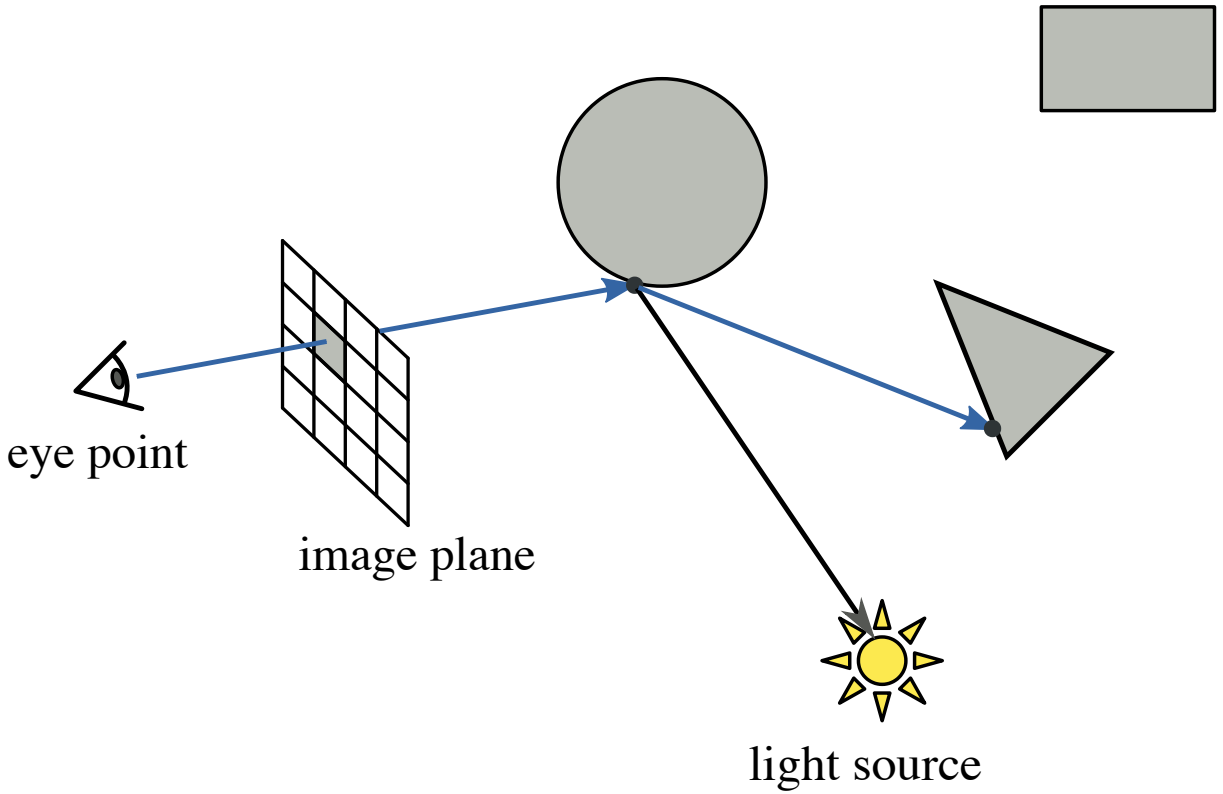
Backward Ray Tracing



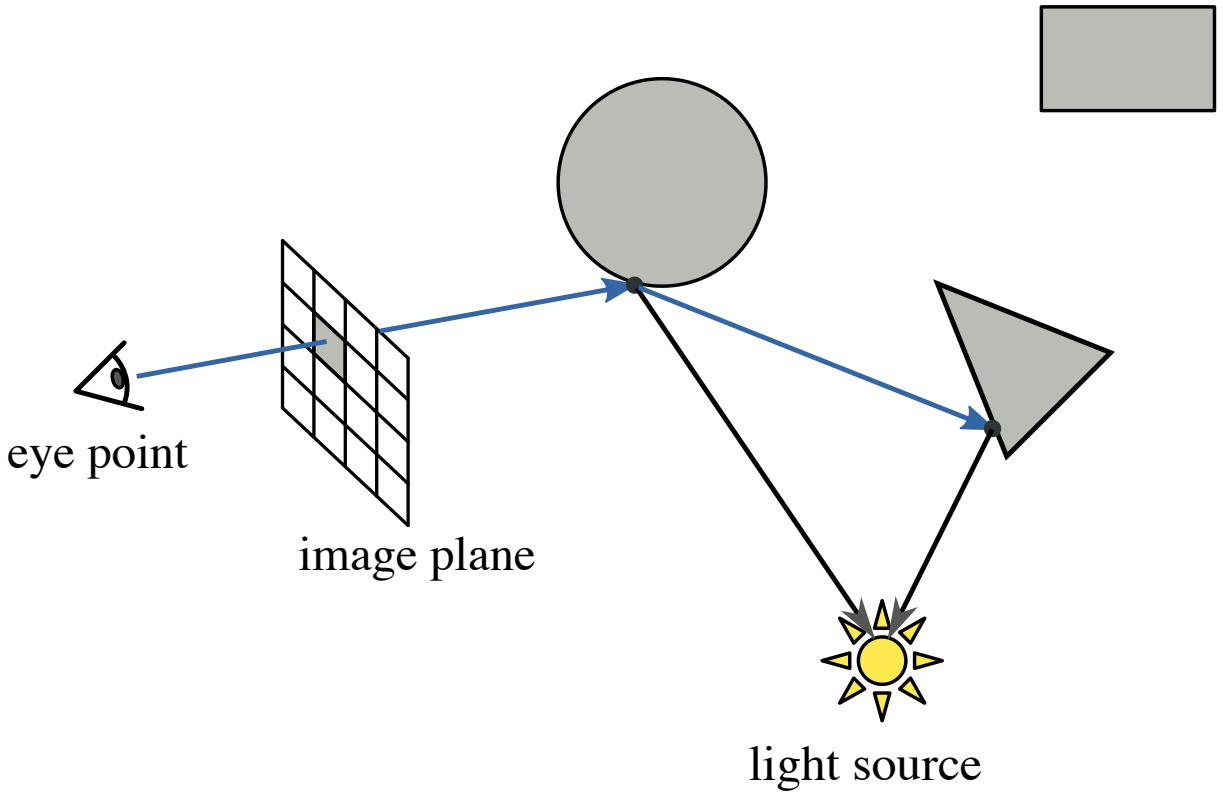
Backward Ray Tracing



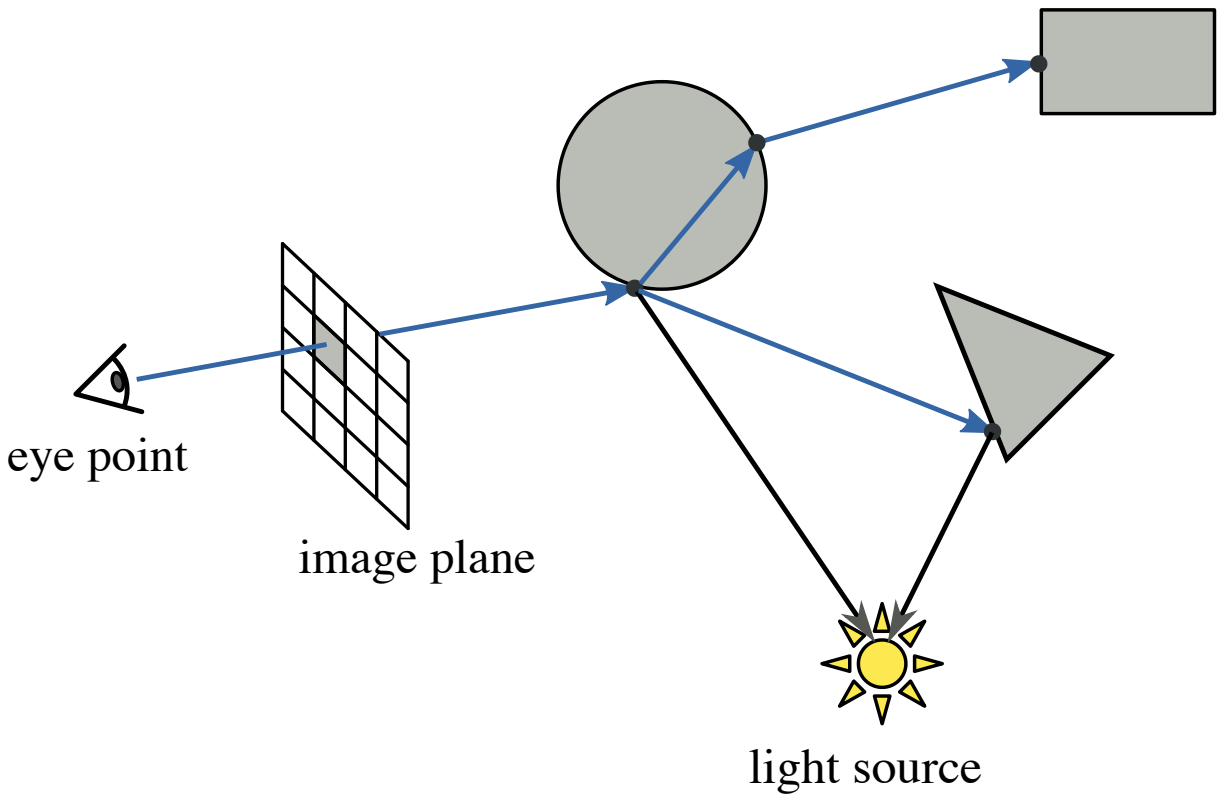
Backward Ray Tracing



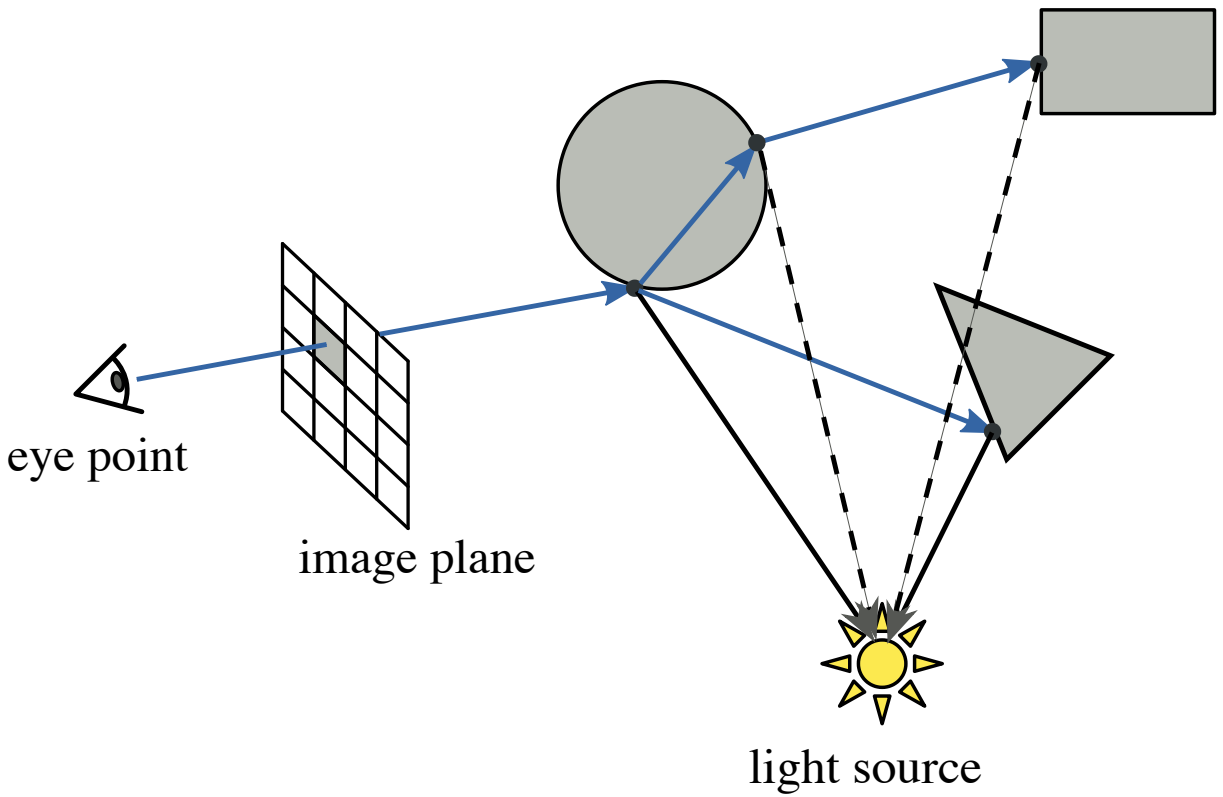
Backward Ray Tracing



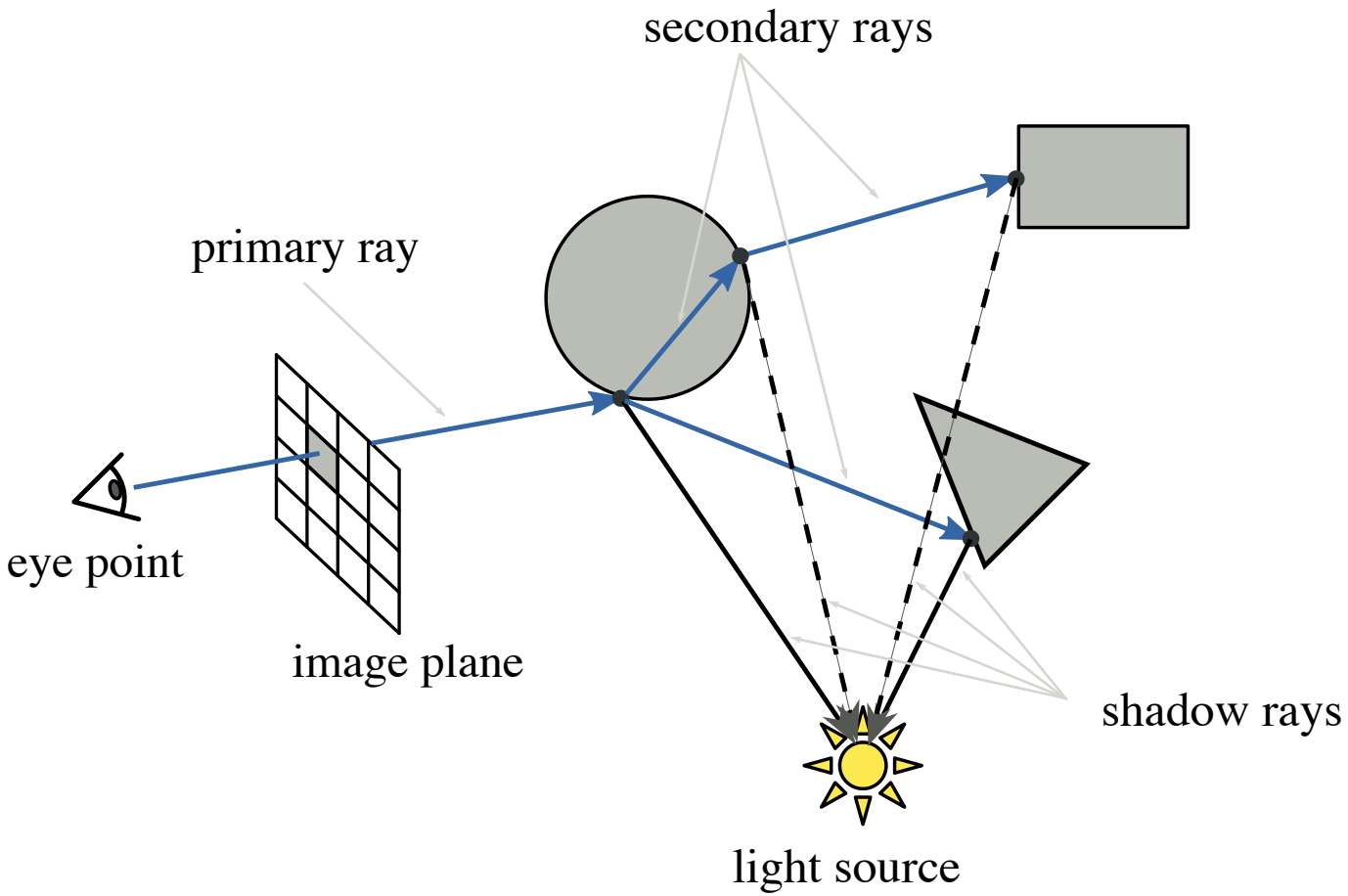
Backward Ray Tracing



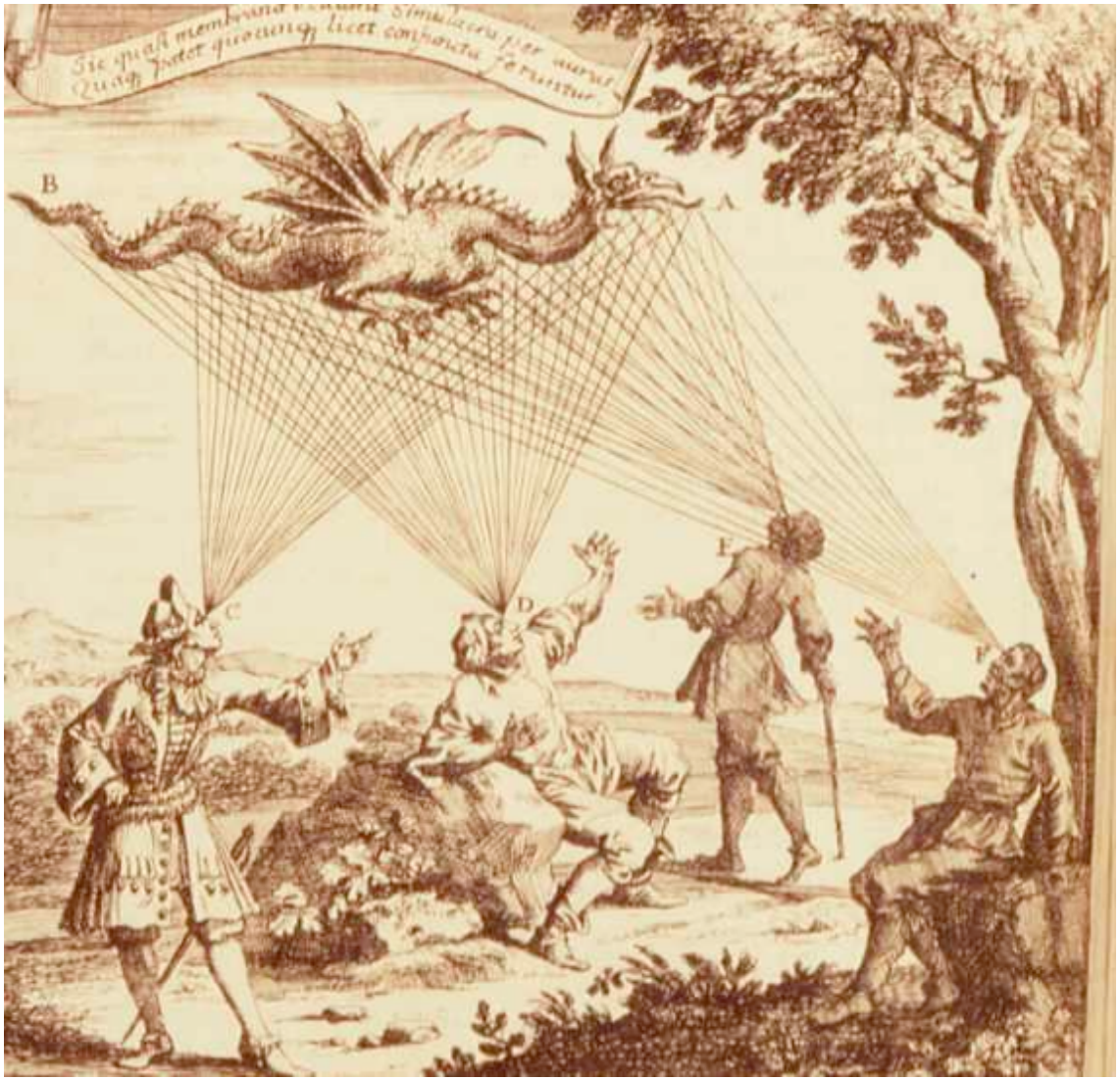
Backward Ray Tracing



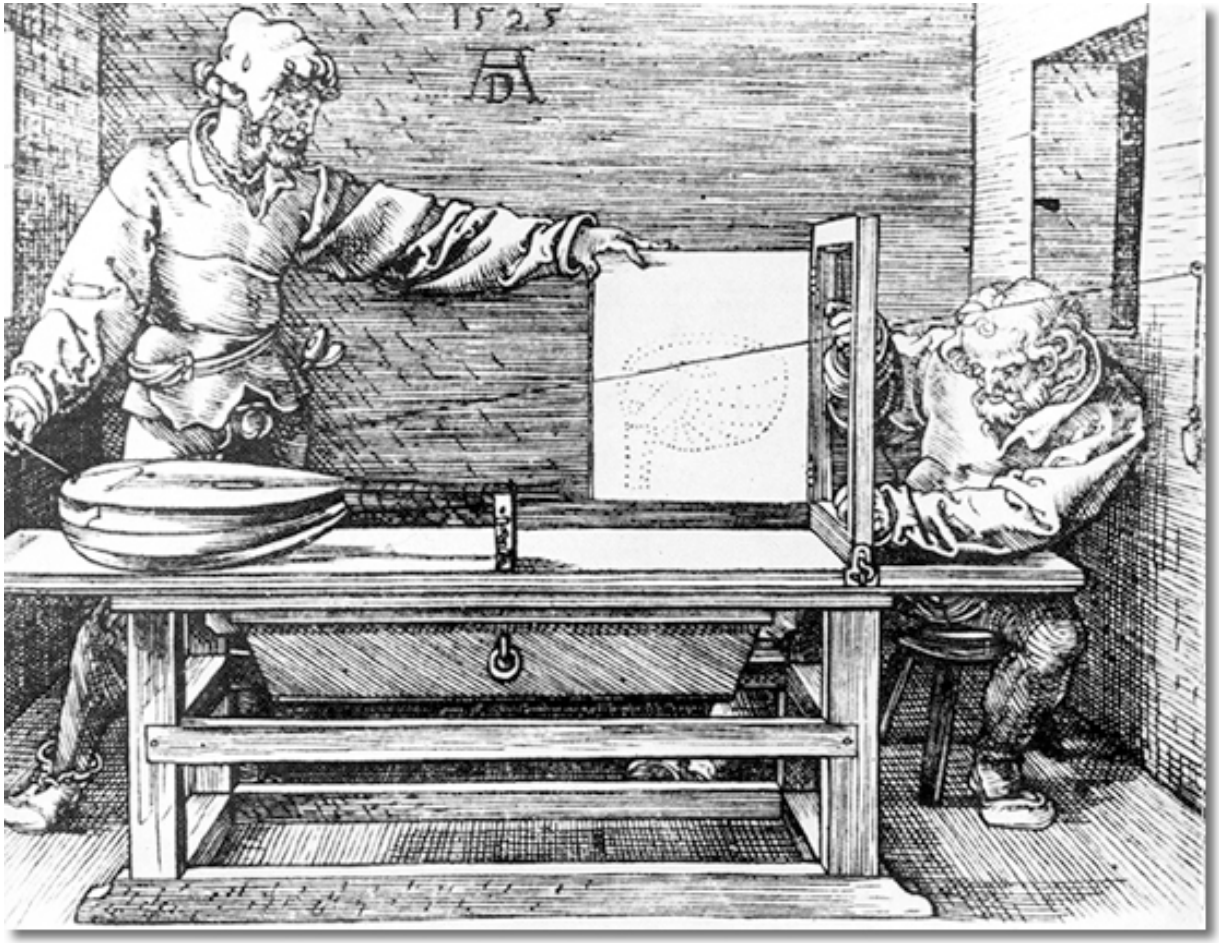
Backward Ray Tracing



Ancient Greeks - Emission Theory



Albrecht Dürer (1525)

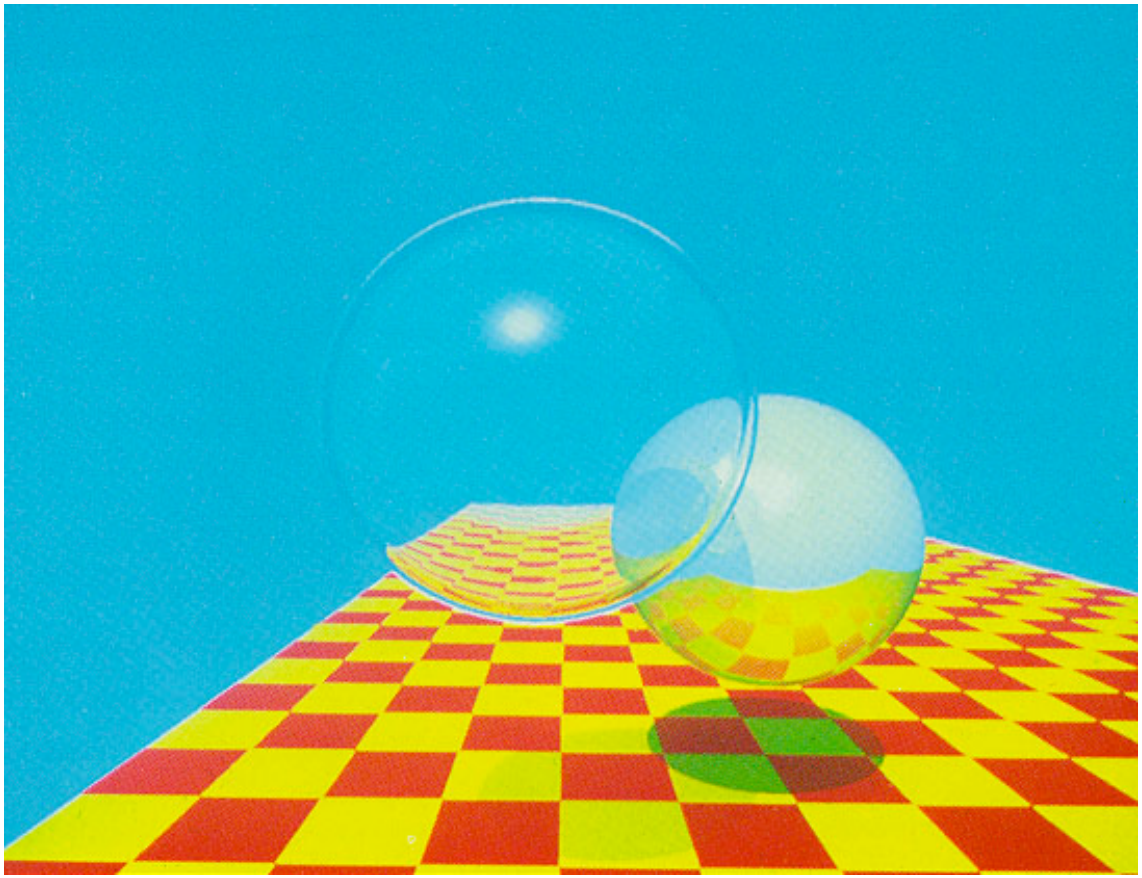


Raycasting Machine
Images from Wikipedia

Albrecht Dürer (1525)

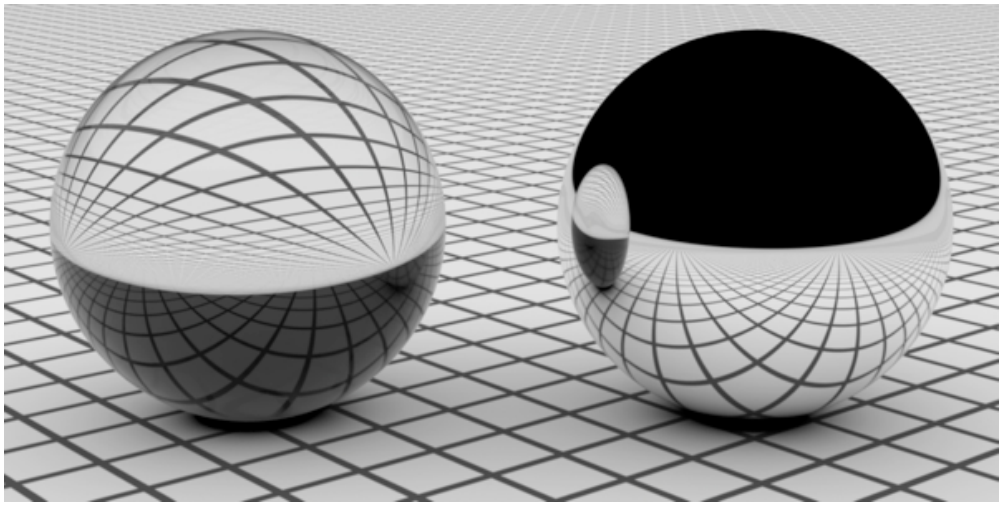
Rasterization
Images from Wikipedia

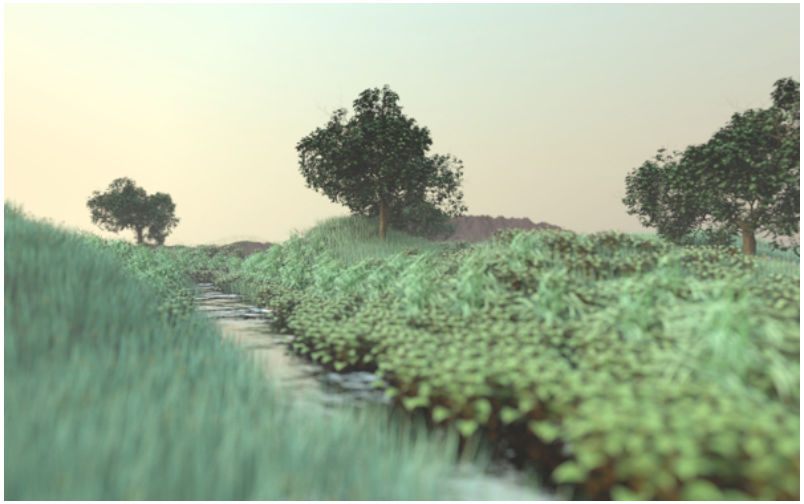
Turner Whitted (1979)



ACM SIGGRAPH 1979 - An improved illumination model for shaded display

Examples from PBRT

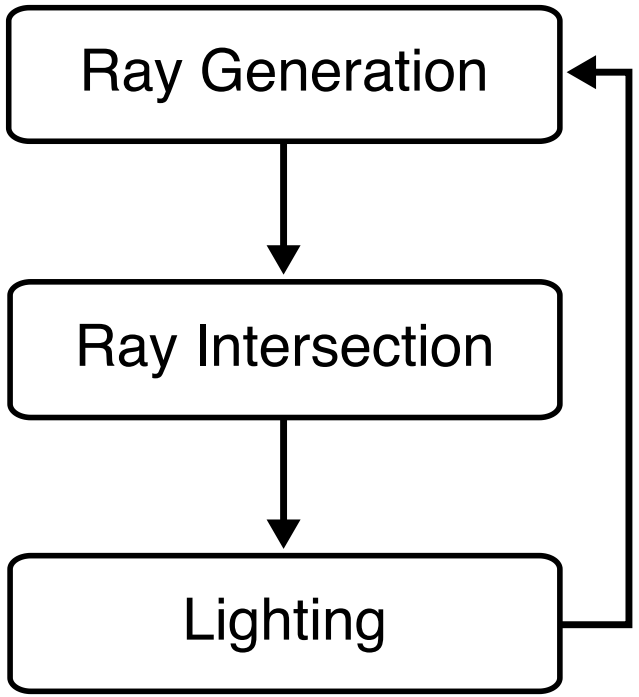


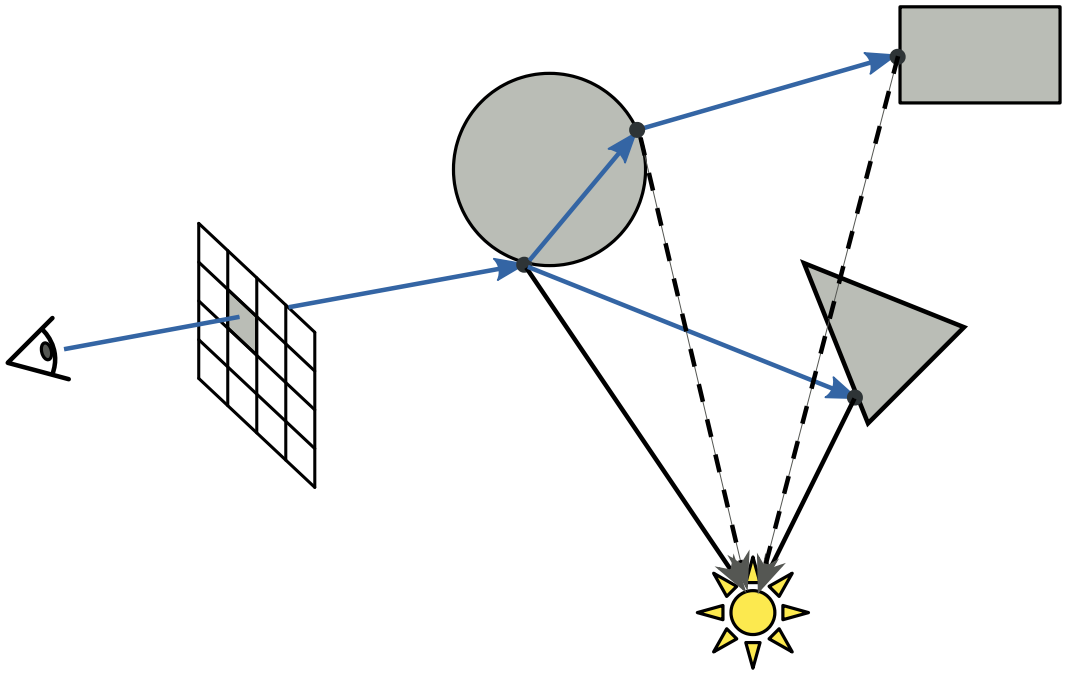


http://www.pbrt.org/gallery.html

Basic Ray Tracing Pipeline

Implementing a ray tracer basically comes down to three operators that we have to understand and implement.





Ray Tracing Pseudo Code

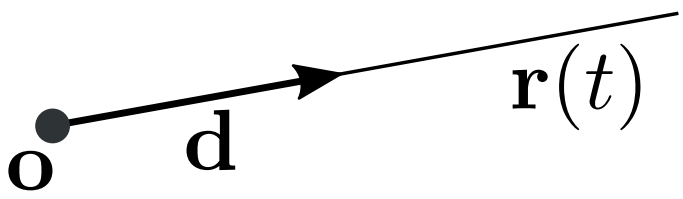
```
void raytrace()
{
    for (int x=0; x<width; ++x)
    {
        for (int y=0; y<height; ++y)
        {
            // generate primary ray through pixel (x,y)
            ray = primary_ray(x,y);

            // determine color through (recursive!) trace function
            color[x,y] = trace(ray);
        }
    }
}
```

Ray Generation

Ray Equation

- Explicit equation for ray $\mathbf{r}(t)$, starting at origin $\mathbf{o}$ and going in (normalized) direction $\mathbf{d}$:

$$\mathbf{r}(t) = \mathbf{o} + t\,\mathbf{d}$$



Vector Space $\mathbb{R}^n$

- Scalars $\alpha, \beta, \gamma \in \mathbb{R}$ are written in italic (mostly Greek) font.
- Vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ are written in bold font, their components are denoted as $\mathbf{x} = (x_1, \ldots, x_n)^\mathsf{T}$.
- Linear combinations $\alpha\,\mathbf{x} + \beta\,\mathbf{y}$ stay in vector space and are performed component-wise.

Euclidean Vector Space $\mathbb{R}^n$

Euclidean Vector Space $\mathbb{R}^n$

- Inner product (or scalar product, or dot product)

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$$

  is symmetric and linear in both arguments

Euclidean Vector Space $\mathbb{R}^n$

- Inner product (or scalar product, or dot product)

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$$

  is symmetric and linear in both arguments
- The induced metric measures geometric length

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}} = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}$$

Euclidean Vector Space $\mathbb{R}^n$

- Inner product (or scalar product, or dot product)

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i$$

  is symmetric and linear in both arguments
- The induced metric measures geometric length

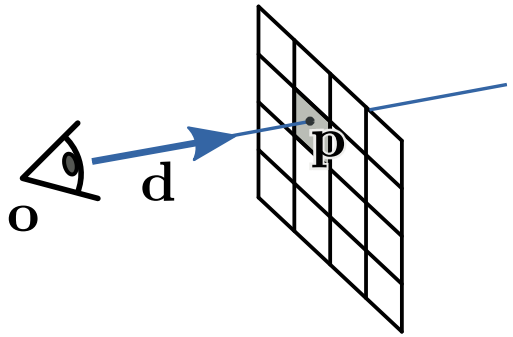
$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}} = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}$$

- Scalar product can measure angle θ between two vectors

Primary Rays

Shoot a primary ray through each pixel (x, y) of the image plane:

1. use eye point as origin $\mathbf{o}$
2. compute 3D position $\mathbf{p}$ of pixel (x, y)
3. compute direction $\mathbf{d} = \frac{\mathbf{p}-\mathbf{o}}{\|\mathbf{p}-\mathbf{o}\|}$



Ray Intersection

Ray-Object Intersections

Ray-Object Intersections

- Which surface types or surface primitives do we want to support?

Ray-Object Intersections

- Which surface types or surface primitives do we want to support?
- Which properties should surface primitives have?

Ray-Object Intersections

- Which surface types or surface primitives do we want to support?
- Which properties should surface primitives have?
- Spheres, cylinders, planes, boxes, triangles…

Ray-Object Intersections

- Which surface types or surface primitives do we want to support?
- Which properties should surface primitives have?
- Spheres, cylinders, planes, boxes, triangles…

We will focus on spheres and planes first.

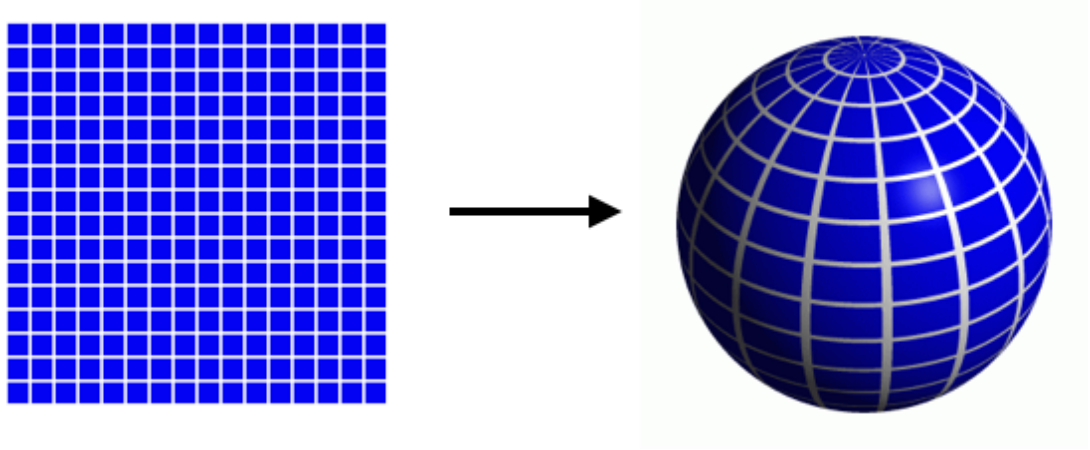
Ray-Sphere Intersection

- How to represent a sphere?
- How to intersect a ray and a sphere?

Unit Sphere

- Unit sphere parameterized by polar coordinates:

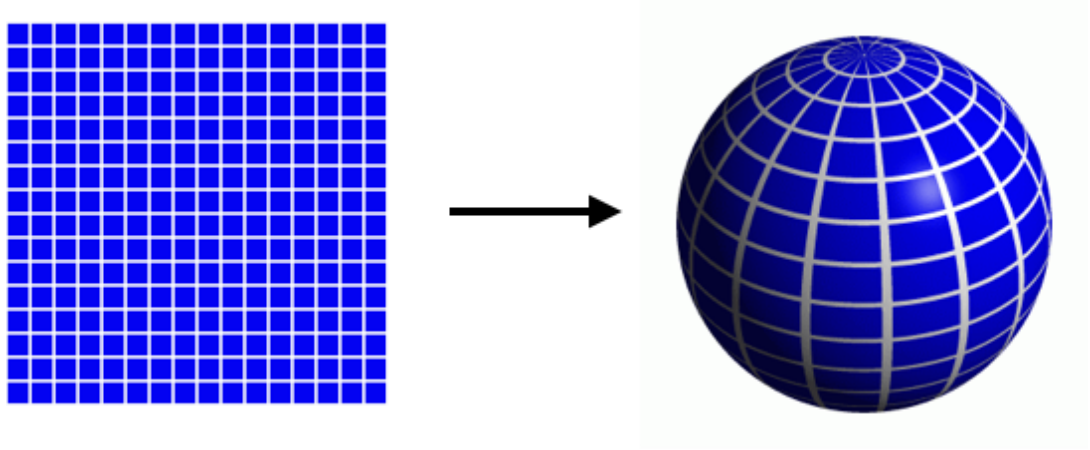
$$\begin{pmatrix} \phi \\ \theta \end{pmatrix} \mapsto \begin{pmatrix} \cos\phi\ \cos\theta \\ \sin\phi\ \cos\theta \\ \sin\theta \end{pmatrix}$$



Arbitrary Sphere

- Sphere with center $\mathbf{c}$ and radius r:

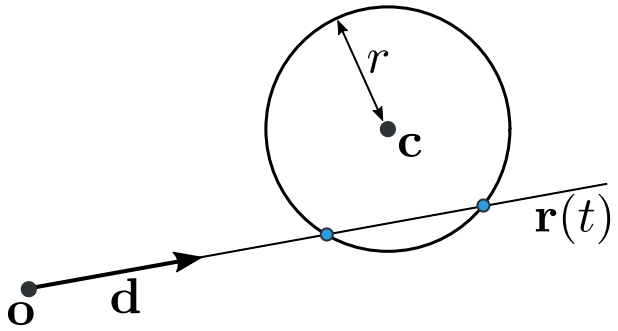
$$\begin{pmatrix} \phi \\ \theta \end{pmatrix} \mapsto \mathbf{c} + r \begin{pmatrix} \cos\phi\ \cos\theta \\ \sin\phi\ \cos\theta \\ \sin\theta \end{pmatrix}$$



Ray-Sphere Intersection

- Ray and sphere have to coincide, leading to this equation

$$\mathbf{o} + t\,\mathbf{d} = \mathbf{c} + r \begin{pmatrix} \cos\phi\ \cos\theta \\ \sin\phi\ \cos\theta \\ \sin\theta \end{pmatrix}$$

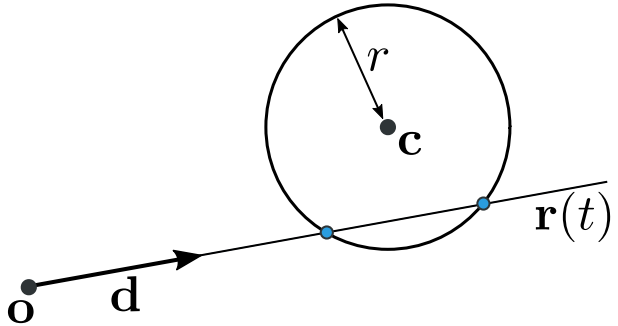


which we have to solve for t, ϕ, and θ.

Ray-Sphere Intersection

- Ray and sphere have to coincide, leading to this equation

$$\mathbf{o} + \mathrm{t}\,\mathbf{d} = \mathbf{c} + \mathrm{r} \begin{pmatrix} \cos\varphi\cos\theta \\ \sin\varphi\cos\theta \\ \sin\theta \end{pmatrix}$$



which we have to solve for t, φ and θ.

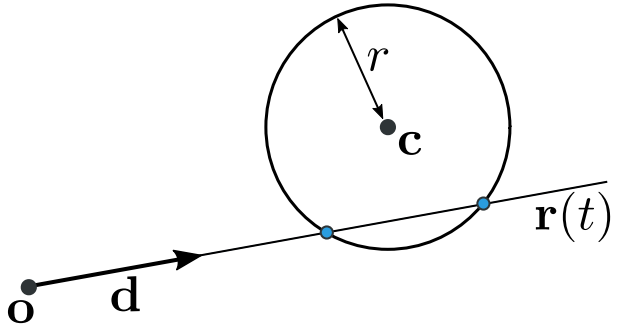
Trigonometric functions make this equation nonlinear and complicated!

Ray-Sphere Intersection, 2nd try

- Let us use the *implicit* representation of a sphere with center $\mathbf{c}$ and radius r:

$$\|\mathbf{x} - \mathbf{c}\| - r = 0$$

This equation is satisfied by all points $\mathbf{x} \in \mathbb{R}^3$ that lie on the sphere.

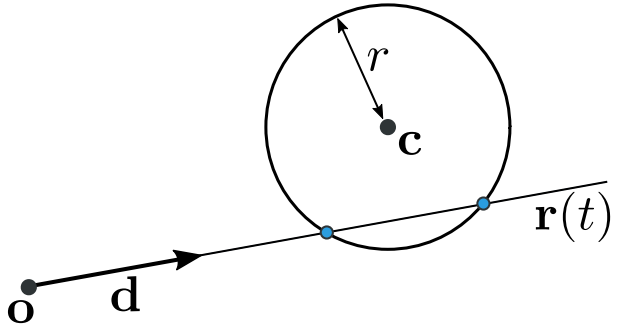


Ray-Sphere Intersection, 2nd try

- Let us use the *implicit* representation of a sphere with center $\mathbf{c}$ and radius r:

$$\|\mathbf{x} - \mathbf{c}\| - \mathrm{r} = 0$$

This equation is satisfied by all points $\mathbf{x} \in \mathbb{R}^3$ that lie on the sphere.



$$\|o + td - c\| = r$$

$$\Leftrightarrow \|o + td - c\|^2 = r^2$$

$$\Leftrightarrow \|td + (o-c)\|^2 = r^2$$

$$\Leftrightarrow (td + (o-c))^T (td + (o-c)) = r^2$$

$$\Leftrightarrow \underbrace{t^2 d^T d}_{a} + \underbrace{2t\, d^T (o-c)}_{t\,2\,d^T(o-c)} + \underbrace{(o-c)^T (o-c) = r^2}_{(o-c)^T(o-c) - r^2}$$

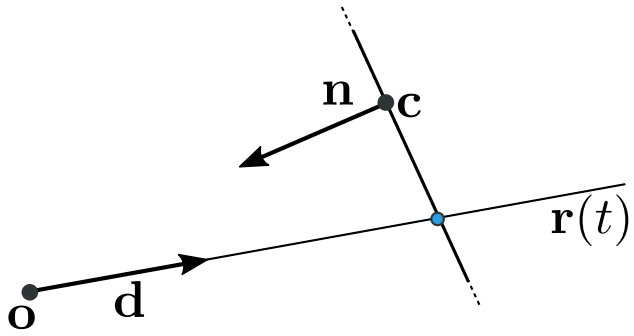
$$\Leftrightarrow at^2 + bt + c = 0$$

Ray-Plane Intersection

- Implicit equation for a plane with normal $\mathbf{n}$ centered at point $\mathbf{c}$ (or with distance d from $(0,0,0)$):

$$\mathbf{n}^\mathsf{T}(\mathbf{x}-\mathbf{c}) = 0$$

$$\mathbf{n}^\mathsf{T}\mathbf{x} - d = 0$$

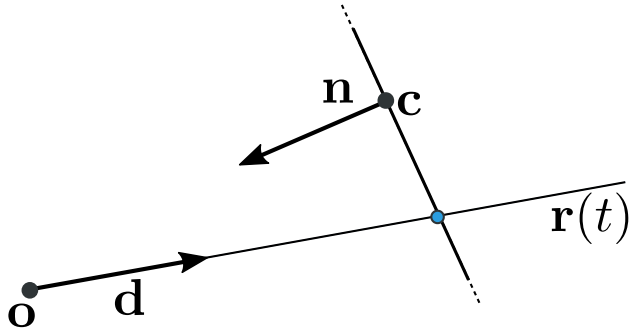


Ray-Plane Intersection

- Implicit equation for a plane with normal $\mathbf{n}$ centered at point $\mathbf{c}$ (or with distance d from $(0, 0, 0)$):

$$\mathbf{n}^{\mathrm{T}}(\mathbf{x} - \mathbf{c}) = 0$$

$$\mathbf{n}^{\mathrm{T}}\mathbf{x} - \mathrm{d} = 0$$



- Insert *explicit* ray equation into *implicit* plane equation

$$\mathbf{n}^{\mathrm{T}}(\mathbf{o} + \mathrm{t}\,\mathbf{d}) - \mathrm{d} = 0$$

and solve for t.

Ray-Triangle Intersection

Ray-Triangle Intersection

- How can we represent a triangle?

Ray-Triangle Intersection

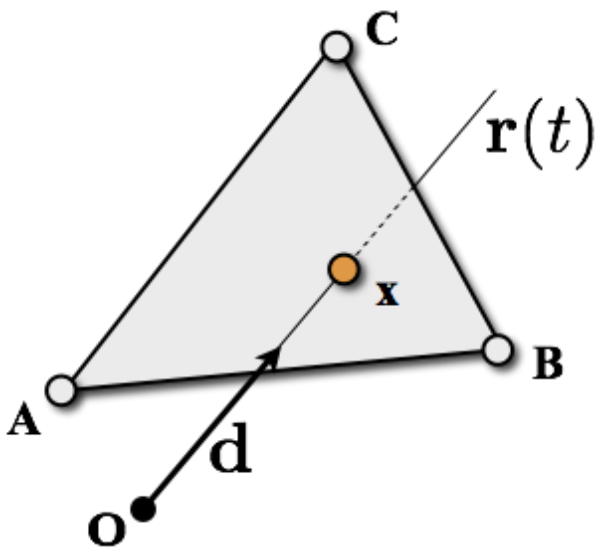
- How can we represent a triangle?
- Should we rather use an *implicit* or *explicit* formulation?

Ray-Triangle Intersection

- How can we represent a triangle?
- Should we rather use an *implicit* or *explicit* formulation?
- Which one can be intersected with a ray more efficiently?

Ray-Triangle Intersection (*implicit version*)

1. Construct triangle's supporting plane

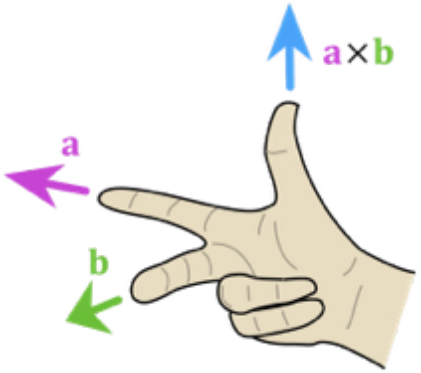


3D Cross Product

Images from Wikipedia

3D Cross Product

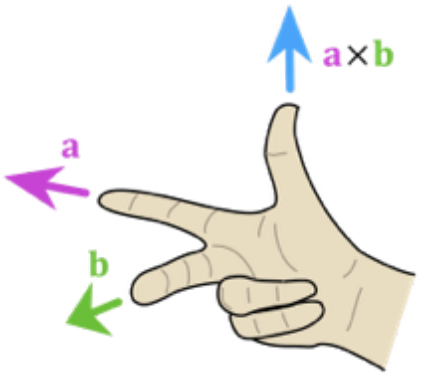
- Given two 3D vectors $\mathbf{a}$ and $\mathbf{b}$, finds a vector $\mathbf{c}$ that is orthogonal to them



$$\mathbf{c} = \mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix}$$

3D Cross Product

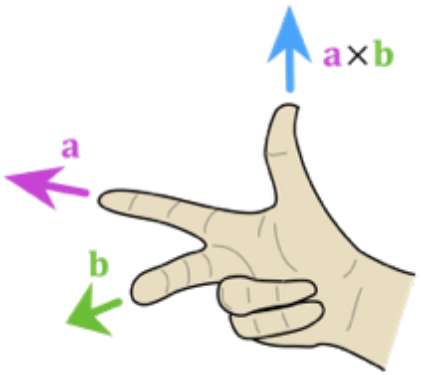
- Given two 3D vectors $\mathbf{a}$ and $\mathbf{b}$, finds a vector $\mathbf{c}$ that is orthogonal to them



$$\mathbf{c} = \mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix}$$

3D Cross Product

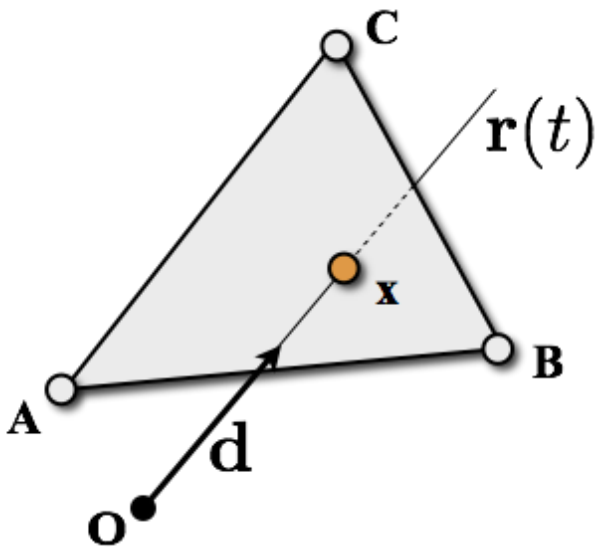
- Given two 3D vectors $\mathbf{a}$ and $\mathbf{b}$, finds a vector $\mathbf{c}$ that is orthogonal to them



$$\mathbf{c} = \mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

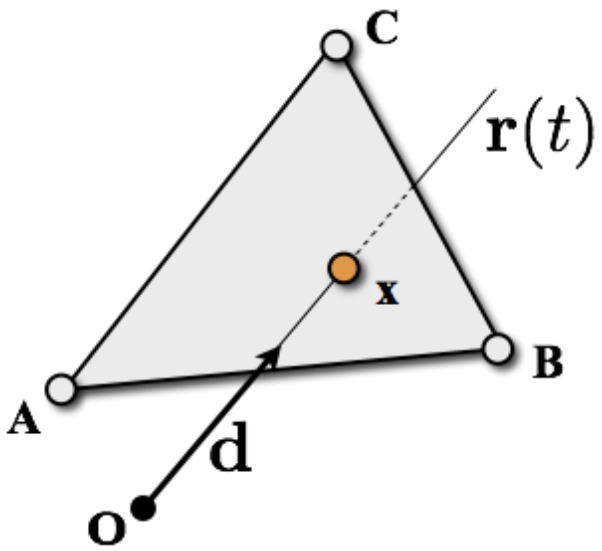
Ray-Triangle Intersection (*implicit version*)

1. Construct triangle's supporting plane
   - Normal $\mathbf{n} = \frac{(\mathbf{B}-\mathbf{A})\times(\mathbf{C}-\mathbf{A})}{\|(\mathbf{B}-\mathbf{A})\times(\mathbf{C}-\mathbf{A})\|}$
   - Distance from origin $d = \mathbf{n}^\mathsf{T}\mathbf{A} = \mathbf{n}^\mathsf{T}\mathbf{B} = \mathbf{n}^\mathsf{T}\mathbf{C}$
2. Intersect ray with triangle's plane
   - Solve $(\mathbf{o} + t\mathbf{d})^\mathsf{T}\mathbf{n} - d = 0$
3. Is intersection point inside/outside of triangle?
   - Project to 2D coordinates
   - Point-in-polygon test



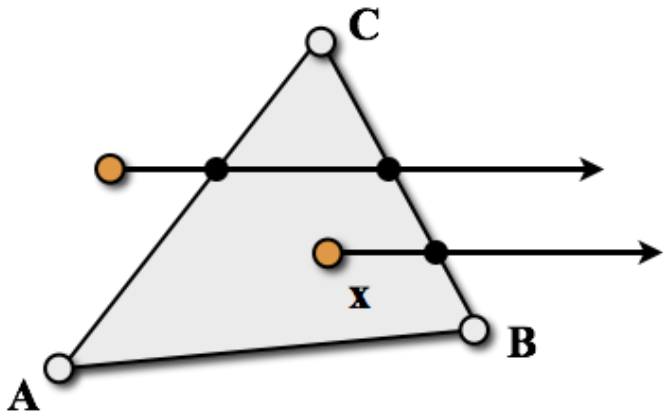
Ray-Triangle Intersection (*implicit version*)

1. Construct triangle's supporting plane
   - Normal $\mathbf{n} = \frac{(\mathbf{B}-\mathbf{A}) \times (\mathbf{C}-\mathbf{A})}{\| (\mathbf{B}-\mathbf{A}) \times (\mathbf{C}-\mathbf{A}) \|}$
   - Distance from origin
     $d = \mathbf{n}^T\mathbf{A} = \mathbf{n}^T\mathbf{B} = \mathbf{n}^T\mathbf{C}$



2. Intersect ray with triangle's plane
   - Solve $(\mathbf{o} + t\mathbf{d})^T\mathbf{n} - d = 0$

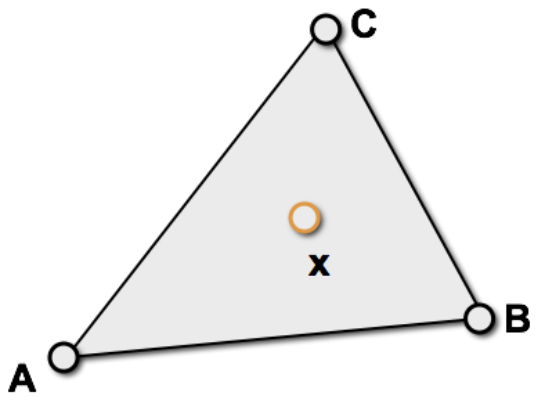
3. Is intersection point inside/outside of triangle?
   - Project to 2D coordinates
   - Point-in-polygon test



Barycentric Coordinates

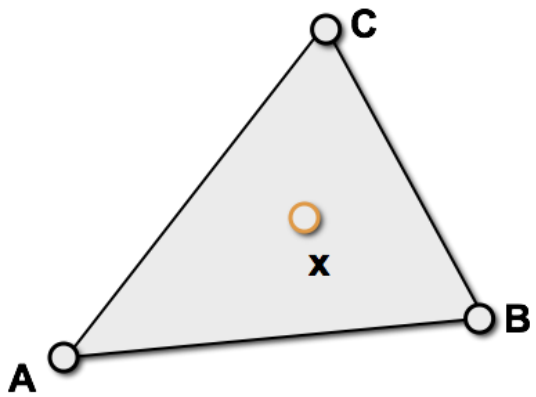
Barycentric Coordinates

- Explicit triangle parameterization (first guess)
  - Origin $\mathbf{A}$, edges $\mathbf{U} = \mathbf{B} - \mathbf{A}$ and $\mathbf{V} = \mathbf{C} - \mathbf{A}$
  - Triangle points $\mathbf{x} = \mathbf{A} + u\,\mathbf{U} + v\,\mathbf{V}$ with $u, v \geq 0$ and $u + v \leq 1$

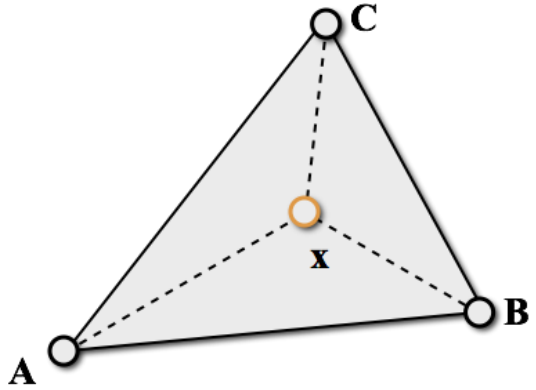


Barycentric Coordinates

- Explicit triangle parameterization (first guess)
  - Origin $\mathbf{A}$, edges $\mathbf{U} = \mathbf{B} - \mathbf{A}$ and $\mathbf{V} = \mathbf{C} - \mathbf{A}$
  - Triangle points $\mathbf{x} = \mathbf{A} + u\,\mathbf{U} + v\,\mathbf{V}$ with $u, v \geq 0$ and $u + v \leq 1$



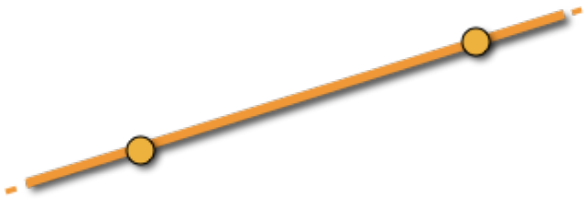
- *Barycentric coordinates* (more useful later)
  - $\mathbf{x} = \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$ with $\alpha + \beta + \gamma = 1$ and $\alpha, \beta, \gamma \geq 0$.
  - Equivalent to above (why?)



Barycentric Coordinates

- *Affine combination* of two points

$$\mathbf{x} = \alpha\mathbf{A} + \beta\mathbf{B}$$



with $\alpha + \beta = 1$.

Barycentric Coordinates

- *Affine combination* of two points

$$\mathbf{x} = \alpha\mathbf{A} + \beta\mathbf{B}$$

with $\alpha + \beta = 1$.

- *Convex combination* of two points

$$\mathbf{x} = \alpha\mathbf{A} + \beta\mathbf{B}$$

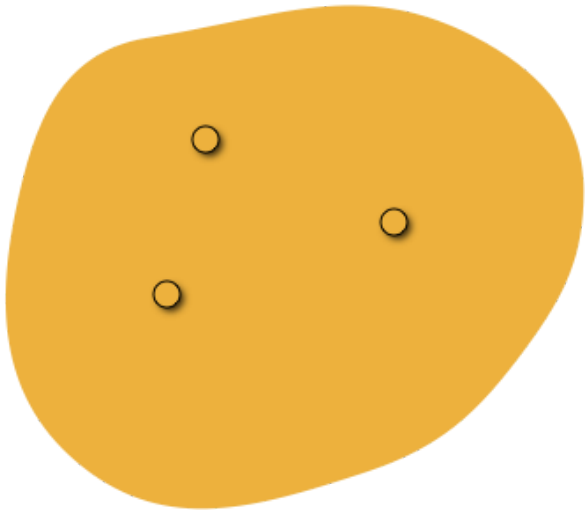
with $\alpha + \beta = 1$ **and** $\alpha, \beta \geq 0$.

Barycentric Coordinates

- *Affine combination* of three points

$$\mathbf{x} = \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$$

with $\alpha + \beta + \gamma = 1$.

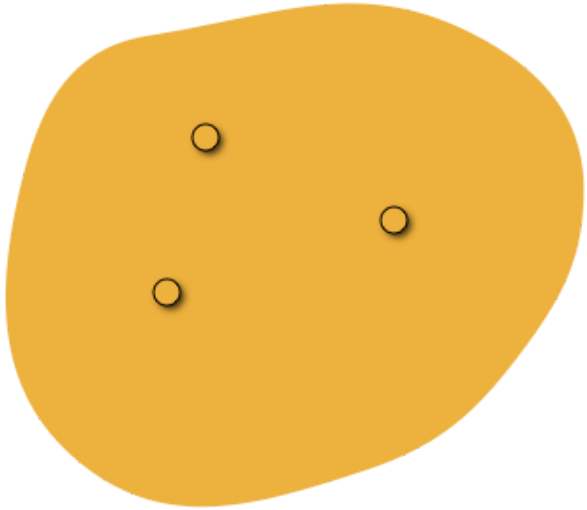


Barycentric Coordinates

- *Affine combination* of three points

$$\mathbf{x} = \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$$

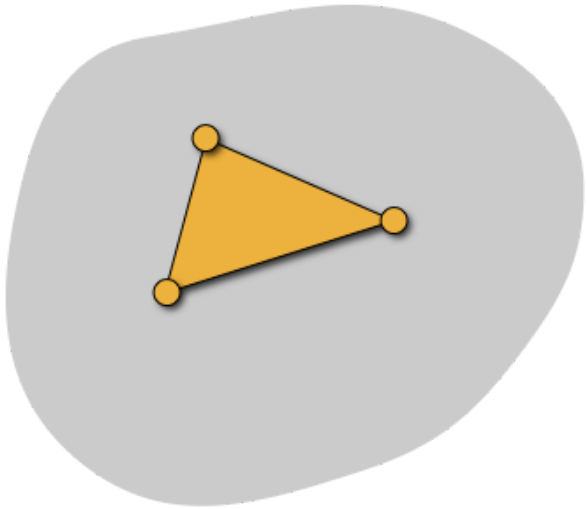
with $\alpha + \beta + \gamma = 1$.



- *Convex combination* of three points

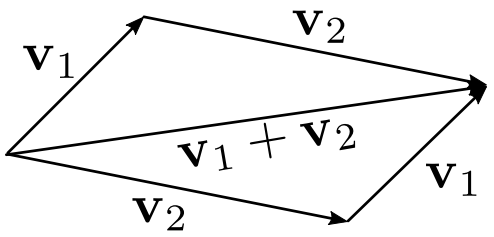
$$\mathbf{x} = \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$$

with $\alpha + \beta + \gamma = 1$ **and** $\alpha, \beta, \gamma \geq 0$.

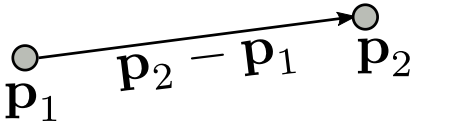


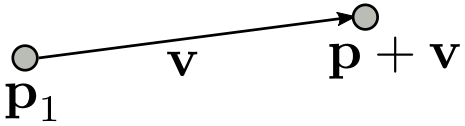
Points vs. Vectors

- Subtle distinction
  - Points denote positions in $\mathbb{R}^3$
  - Vectors denote differences of points



- Meaningful operations
  - vector + vector = vector
  - point - point = vector
  - point + vector = point
  - point + point = ???



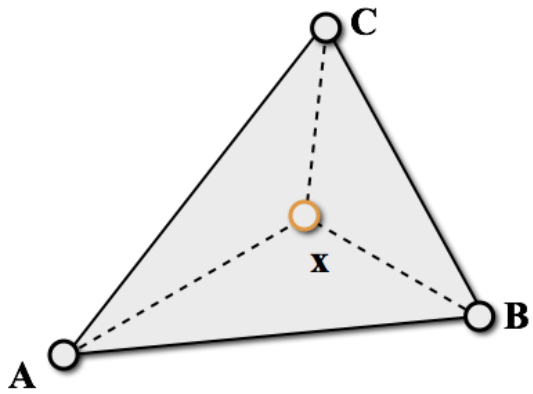


Barycentric Coordinates

Barycentric Coordinates

- Ratio of (signed!) areas (volumes in 3D)

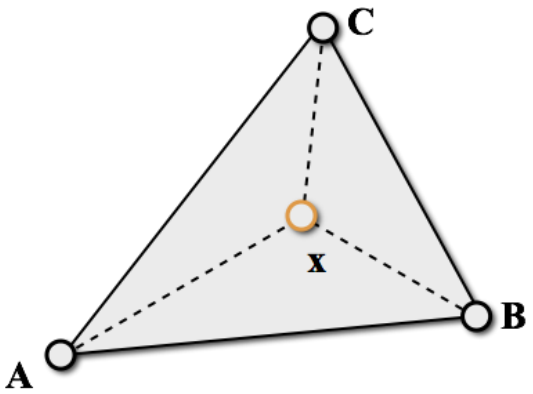
$$\alpha(\mathbf{x}) = \text{area}(\mathbf{x}, \mathbf{B}, \mathbf{C}) / \text{area}(\mathbf{A}, \mathbf{B}, \mathbf{C})$$
$$\beta(\mathbf{x}) = \text{area}(\mathbf{A}, \mathbf{x}, \mathbf{C}) / \text{area}(\mathbf{A}, \mathbf{B}, \mathbf{C})$$
$$\gamma(\mathbf{x}) = \text{area}(\mathbf{A}, \mathbf{B}, \mathbf{x}) / \text{area}(\mathbf{A}, \mathbf{B}, \mathbf{C})$$



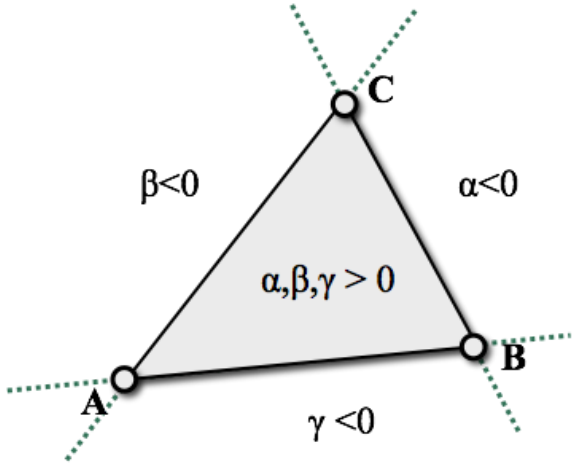
Barycentric Coordinates

- Ratio of (signed!) areas (volumes in 3D)

$$\alpha(\mathbf{x}) = \text{area}(\mathbf{x}, \mathbf{B}, \mathbf{C}) / \text{area}(\mathbf{A}, \mathbf{B}, \mathbf{C})$$
$$\beta(\mathbf{x}) = \text{area}(\mathbf{A}, \mathbf{x}, \mathbf{C}) / \text{area}(\mathbf{A}, \mathbf{B}, \mathbf{C})$$
$$\gamma(\mathbf{x}) = \text{area}(\mathbf{A}, \mathbf{B}, \mathbf{x}) / \text{area}(\mathbf{A}, \mathbf{B}, \mathbf{C})$$



- Gives the required inside/outside information!

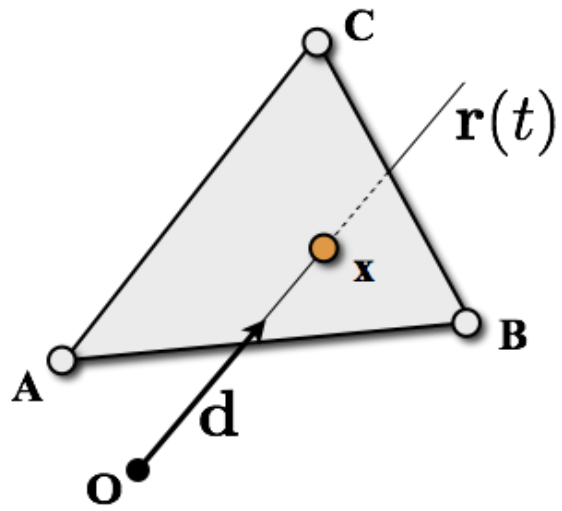


Ray-Triangle Intersection (explicit)

- Ray and triangle have to coincide

$$\mathbf{o} + t\,\mathbf{d} \;=\; \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$$

Four unknowns, but only three equations!

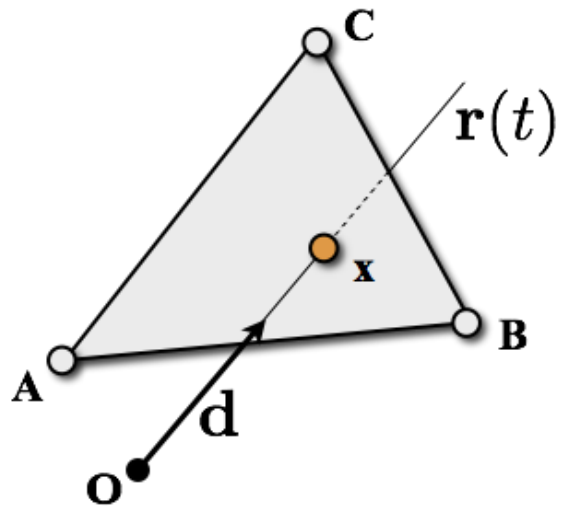


Ray-Triangle Intersection (explicit)

- Ray and triangle have to coincide

$$\mathbf{o} + t\,\mathbf{d} \;=\; \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$$

Four unknowns, but only three equations!



- Exploit condition $\alpha + \beta + \gamma = 1$ to eleminate α

$$\mathbf{o} + t\,\mathbf{d} \;=\; (1 - \beta - \gamma)\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$$

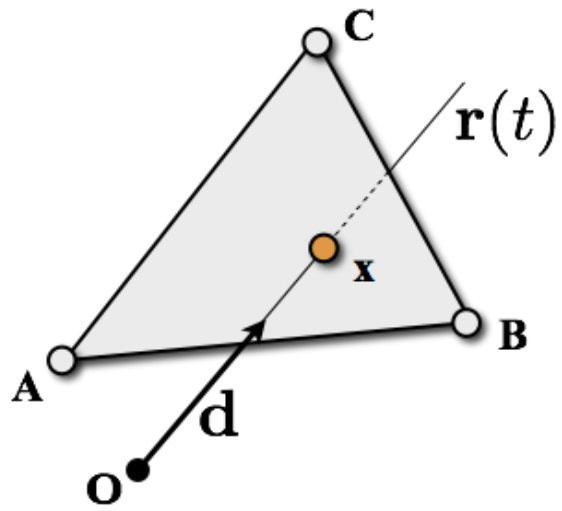
Solve 3×3 linear system (Cramer's rule)

Ray-Triangle Intersection (explicit)

- Ray and triangle have to coincide

$$\mathbf{o} + t\,\mathbf{d} \;=\; \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$$

Four unknowns, but only three equations!



- Exploit condition $\alpha + \beta + \gamma = 1$ to eleminate α

$$\mathbf{o} + t\,\mathbf{d} \;=\; (1 - \beta - \gamma)\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$$

Solve 3×3 linear system (Cramer's rule)

Other Primitives

Other Primitives

- Cylinder (exercise)

Other Primitives

- Cylinder (exercise)
- Cone, paraboloid, hyperboloid

Other Primitives

- Cylinder (exercise)
- Cone, paraboloid, hyperboloid
- Meshes (in two weeks)

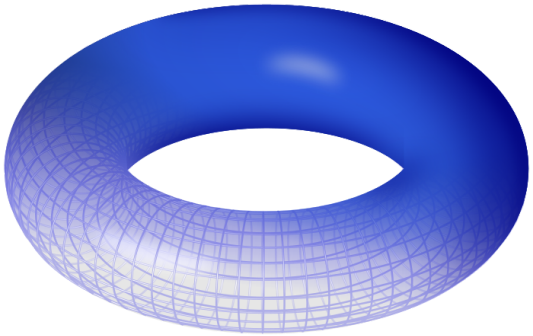
Other Primitives

- Cylinder (exercise)
- Cone, paraboloid, hyperboloid
- Meshes (in two weeks)
- Torus

Other Primitives

- Cylinder (exercise)
- Cone, paraboloid, hyperboloid
- Meshes (in two weeks)
- Torus

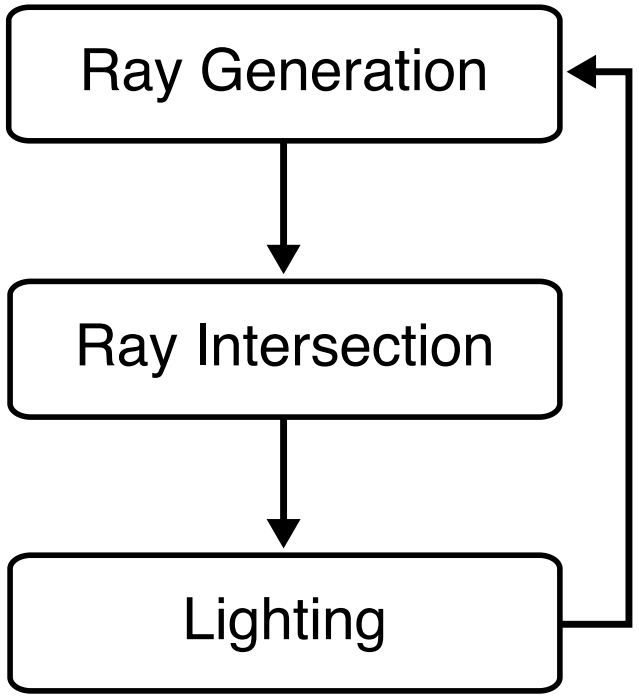
$$(\mathrm{R} - \sqrt{\mathbf{x}^2 + \mathbf{y}^2})^2 + \mathbf{z}^2 = \mathrm{r}^2$$

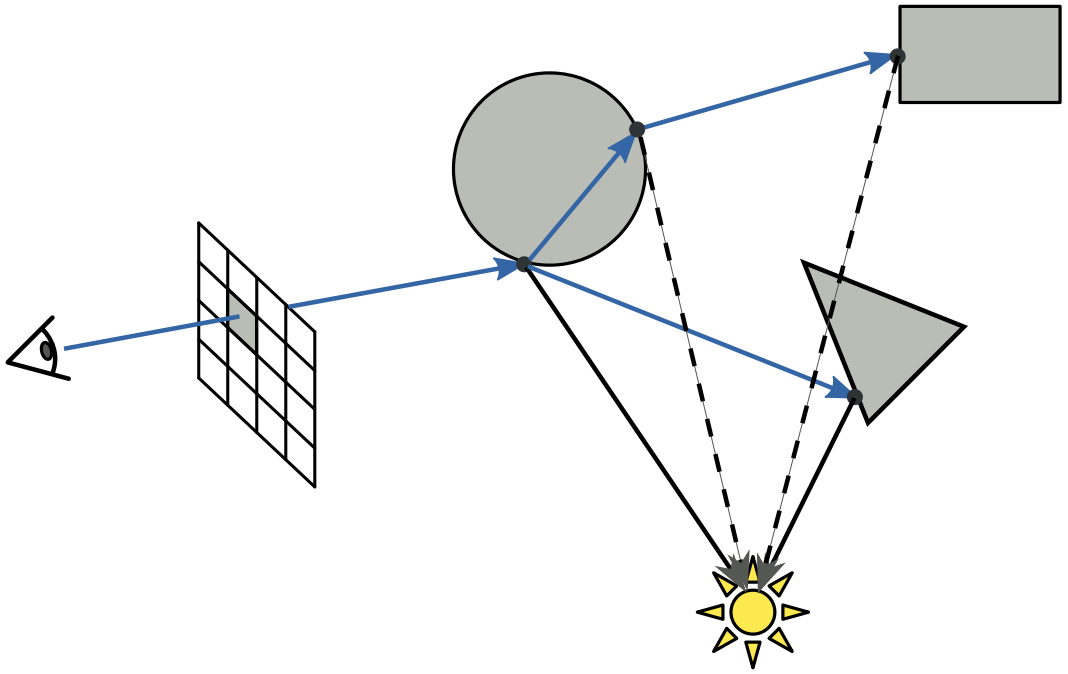


Wikipedia

Ray Tracing Pipeline

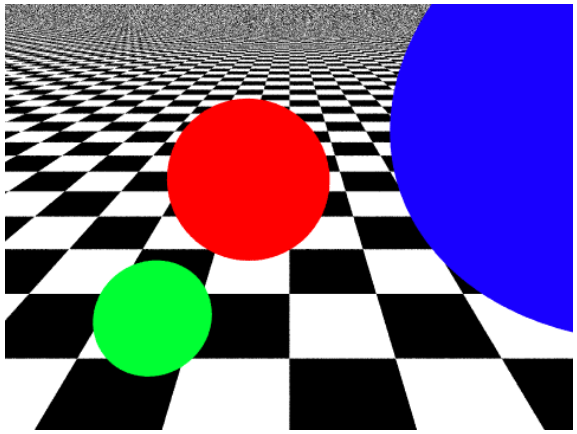
- We understood **ray generation** and **ray intersections**. Next week we'll talk about **lighting**.



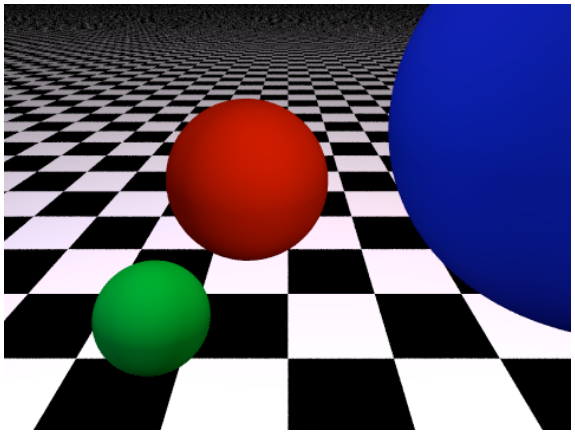


Ray Tracing Pipeline

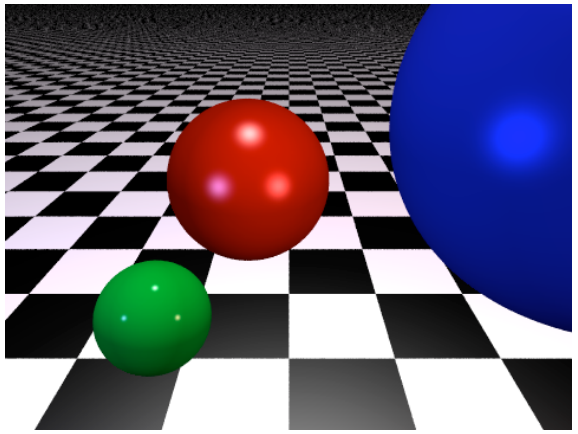
- We understood **ray generation** and **ray intersections**. Next week we'll talk about **lighting**.



color only



+diffuse

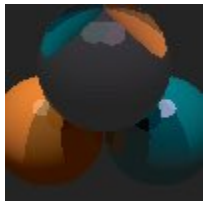


+specular

Is it difficult to code?

Is it difficult to code?

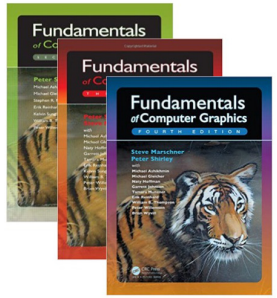
- Paul Heckbert's Minimal Ray Tracer
  (was printed on the back of his business card)



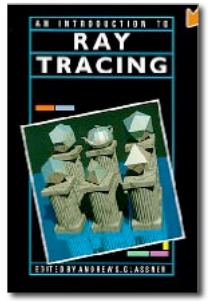
```
typedef struct{double x,y,z}vec;vec U,black,amb={.02,.02,.02};struct sphere{
vec cen,color;double rad,kd,ks,kt,kl,ir}*s,*best,sph[]={0.,6.,.5,1.,1.,1.,.9,
.05,.2,.85,0.,1.7,-1.,8.,-.5,1.,.5,.2,1.,.7,.3,0.,.05,1.2,1.,8.,-.5,.1,.8,.8,
1.,.3,.7,0.,0.,1.2,3.,-6.,15.,1.,.8,1.,7.,0.,0.,0.,.6,1.5,-3.,-3.,12.,.8,1.,
1.,5.,0.,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,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);}/*minray!*/
```

Literature

- Shirley et al.: *Fundamentals of Computer Graphics*, 3rd Edition, AK Peters, 2009.
  - Chapter 4



- Glassner: *An Introduction to Ray Tracing*, Academic Press, 1989.
  - Chapters 2 & 4



- Pharr, Humphreys: *Physically Based Rendering*, Morgan Kaufmann, 2004.
  - Chapters 1-3