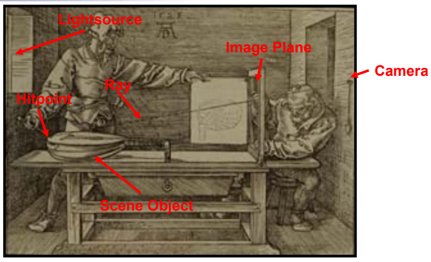
Ray Tracing

Quicker than a ray of light…

Elmar Eisemann
TU Delft

What are the basic principles of ray tracing?

„Man drawing a lute“,
woodcut by Albrecht Dürer (1525)

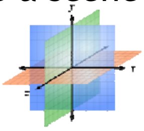


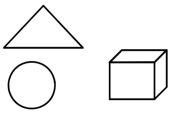
Basic principles of ray tracing

Send straight rays from the camera through the image plane and evaluate the lighting at the first hitpoint with the scene.

How to describe a scene

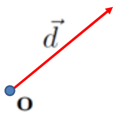
- World coordinate system
- Camera
- Lights
- Materials
- Objects
  - Triangle
    - 3 vertices (points)
  - Sphere
    - midpoint and radius
  - Box
    - min, max vertex
  - …





Ray Representation

- Ray in space defined by
  - Origin $\mathbf{o}$
  - Direction $\vec{d}$



- Any point along the ray defined by

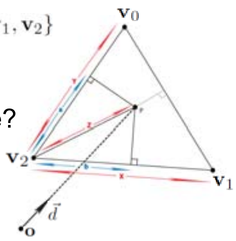
$$\mathbf{r}(t) = \mathbf{o} + t \cdot \vec{d}$$

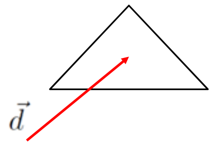
- t greater 0: point is in front of origin
- t smaller 0: point is behind origin

- Most important task in ray tracing:
  Intersect ray with scene

Ray / Triangle Intersection

- Given a triangle $\mathbf{V} = \{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2\}$ and ray $\mathbf{r}(t) = \mathbf{o} + t \cdot \vec{d}$
- Do they intersect? Where?
- Many techniques exist
- Here comes the most „intuitive" one...

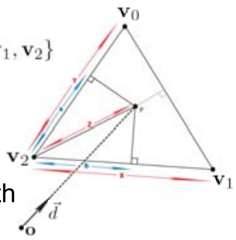




Ray / Triangle Intersection

- Given a triangle $\mathbf{V} = \{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2\}$ and ray $\mathbf{r}(t) = \mathbf{o} + t \cdot \vec{d}$

1. Compute intersection with triangle plane
2. Check if intersection is inside triangle



Different Intersection Techniques

- Ray vs. Plane
- Ray vs. Triangle
- Ray vs. Axis-aligned Box
- Ray vs. Sphere

Different Intersection Techniques

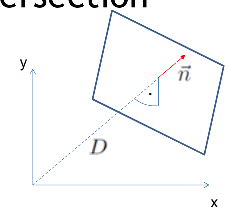
- Ray vs. Plane
- Ray vs. Triangle
- Ray vs. Axis-aligned Box
- Ray vs. Sphere

Ray / Plane Intersection

- Plane equation

$$\mathbf{p} \circ \vec{n} - D = 0 \qquad |\vec{n}| = 1$$

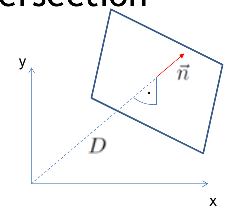
  - Implicit representation (Hesse Form)
  - Normal vector: $\vec{n}$
  - Normal distance from origin (0,0,0): D



Ray / Plane Intersection

- Plane equation

$$\mathbf{p} \circ \vec{n} - D = 0 \qquad |\vec{n}| = 1$$



  - Implicit representation (Hesse Form)
  - Normal vector: $\vec{n}$
  - Normal distance from origin (0,0,0): D

Let's assume a ray: $\mathbf{r}(t) = \mathbf{o} + t \cdot \vec{d}$

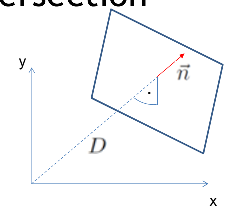
Place in plane equation: $(\mathbf{o} + t \cdot \vec{d}) \circ \vec{n} - D \;=\; 0$

$$\boxed{t \;=\; \frac{D - \mathbf{o} \circ \vec{n}}{\vec{d} \circ \vec{n}}}$$

Ray / Plane Intersection

- Plane equation

$$\mathbf{p} \circ \vec{n} - D = 0 \qquad |\vec{n}| = 1$$



How do we find $\vec{n}$ and D ?

Cross Product

$$\mathbf{c} = \mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \times \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix}$$

$$|\mathbf{c}| = |\mathbf{a} \times \mathbf{b}| = |\mathbf{a}| \cdot |\mathbf{b}| \cdot \sin\alpha$$

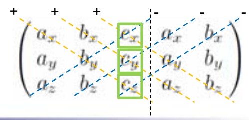
- c is a vector which is orthogonal to a and b!

Cross Product

$$\mathbf{c} = \mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \times \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix}$$

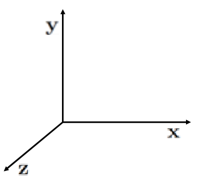
$$|\mathbf{c}| = |\mathbf{a}| \cdot |\mathbf{b}| \cdot \sin\alpha$$

- c is a vector which is orthogonal to a and b!
- How to compute? Sarrus:



Example

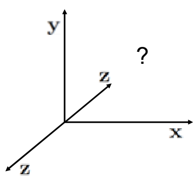
$$\mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathbf{z} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$



$$\mathbf{x} \times \mathbf{y} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \mathbf{z}$$

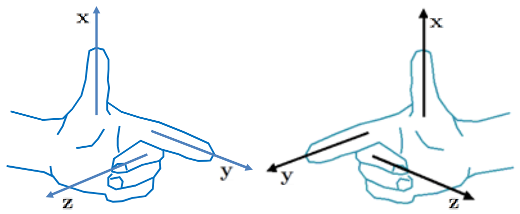
Example

$$\mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \mathbf{z} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$



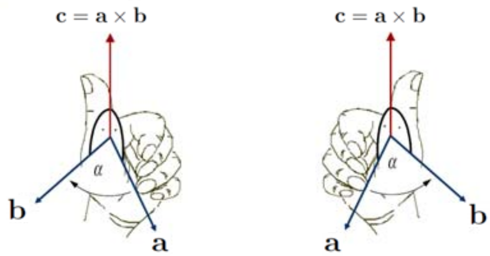
$$\mathbf{x} \times \mathbf{y} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \mathbf{z}$$

3D Coordinate System



- Left-handed coordinate system
- Right-handed coordinate system

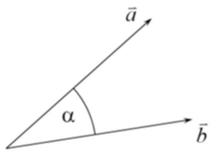
3D Coordinate System



- Left-handed coordinate system
- Right-handed coordinate system

Dot Product

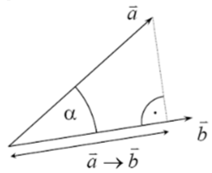
$$\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} \quad \vec{b} = \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix}$$



- Dot product

$$\vec{a} \circ \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot \cos\alpha = a_x b_x + a_y b_y + a_z b_z$$

Projection



- Orthogonal projection

$$\vec{a} \to \vec{b} = |\vec{a}| \cdot \cos\alpha = \frac{\vec{a} \circ \vec{b}}{|\vec{b}|}$$

- The projected vector is

$$(\vec{a} \to \vec{b}) \cdot \frac{\vec{b}}{|\vec{b}|}$$

- If the vector we project onto is normalized $|\vec{b}| = 1$

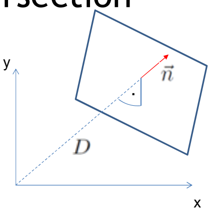
Then $\vec{a} \to \vec{b} = \vec{a} \circ \vec{b}$

The dot product is the orthogonal projection onto a normalized vector!

Ray / Plane Intersection

- Plane equation

$$\mathbf{p} \circ \vec{n} - D = 0 \qquad |\vec{n}| = 1$$

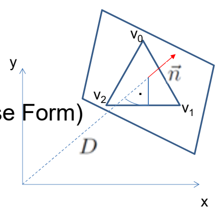


How do we find $\vec{n}$ and D ?

Ray / Plane Intersection

- Plane equation

$$\mathbf{p} \circ \vec{n} - D = 0 \qquad |\vec{n}| = 1$$



  - Implicit representation (Hesse Form)
  - Normal vector: $\vec{n}$
  - Normal distance from origin (0,0,0): D

Computation of $\vec{n}$: $\vec{n} = \frac{((v_0 - v_2) \times (v_1 - v_2))}{|(v_0 - v_2) \times (v_1 - v_2)|}$

If $\vec{n} \circ vi < 0$, then switch direction $\vec{n} = -\vec{n}$

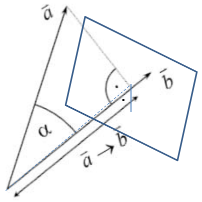
Computation of D?

Project a point of the plane onto the normal vector

Projection

- Orthogonal projection

$$\vec{a} \rightarrow \vec{b} = |\vec{a}| \cdot \cos\alpha = \frac{\vec{a} \circ \vec{b}}{|\vec{b}|}$$



- The projected vector is

$$(\vec{a} \rightarrow \vec{b}) \cdot \frac{\vec{b}}{|\vec{b}|}$$

- If the vector we project onto is normalized $|\vec{b}| = 1$

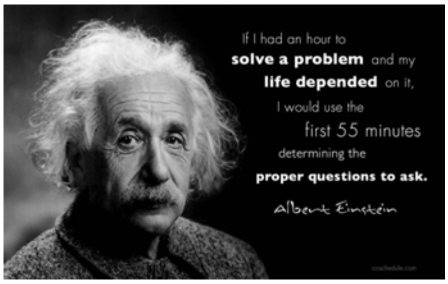
Then $\vec{a} \rightarrow \vec{b} = \vec{a} \circ \vec{b}$

Hence, we can compute plane distance

The dot product is the orthogonal projection onto a normalized vector!

25

Questions

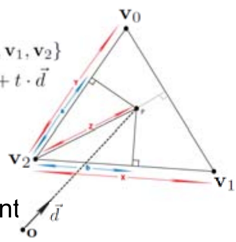


Different Intersection Techniques

- Ray vs. Plane
- Ray vs. Triangle
- Ray vs. Axis-aligned Box
- Ray vs. Sphere

Ray / Triangle Intersection

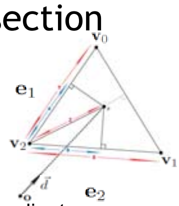
- Given a triangle $\mathbf{V} = \{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2\}$
  and ray $\mathbf{r}(t) = \mathbf{o} + t \cdot \vec{d}$
  and hit parameter t_{hit}



- Compute intersection point
  $\mathbf{p} = \mathbf{r}(t_{hit}) = \mathbf{o} + t_{hit} \cdot \vec{d}$
- Test if **p** is inside triangle

28

Ray / Triangle Intersection



- Test if **p** is inside triangle
  - Express **p** as a linear combination of the edge vectors:

$$\mathbf{p} = \mathbf{v}_2 + a \cdot \mathbf{e}_1 + b \cdot \mathbf{e}_2$$
$$\mathbf{p} = \mathbf{v}_2 + a \cdot (\mathbf{v}_0 - \mathbf{v}_2) + b \cdot (\mathbf{v}_1 - \mathbf{v}_2)$$

  - Rearranging yields Barycentric Coordinates:

$$\mathbf{p} = a \cdot \mathbf{v}_0 + b \cdot \mathbf{v}_1 + (1 - a - b) \cdot \mathbf{v}_2$$

  - Use Barycentric Coordinates to determine hit
    - if ((a < 0) || (a>1)): no hit
    - if (b < 0): no hit (due to the third statement below a check for (b>1) is not necessary)
    - if (a+b > 1): no hit
    - else: hit

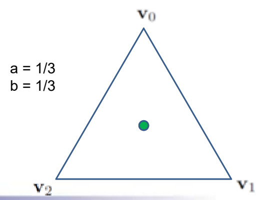
29

Barycentric Coordinates

- Parameters a, b and (1-a-b) are called Barycentric Coordinates

$$\mathbf{p} = a \cdot \mathbf{v}_0 + b \cdot \mathbf{v}_1 + (1 - a - b) \cdot \mathbf{v}_2$$

- Changing these parameters moves a point inside the triangle



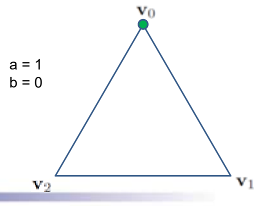
30

Barycentric Coordinates

- Parameters a, b and (1-a-b) are called Barycentric Coordinates

$$\mathbf{p} = a \cdot \mathbf{v}_0 + b \cdot \mathbf{v}_1 + (1 - a - b) \cdot \mathbf{v}_2$$

- Changing these parameters moves a point inside the triangle



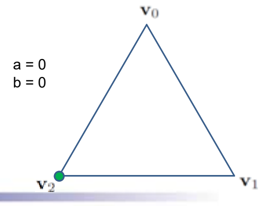
31

Barycentric Coordinates

- Parameters a, b and (1-a-b) are called Barycentric Coordinates

$$\mathbf{p} = a \cdot \mathbf{v}_0 + b \cdot \mathbf{v}_1 + (1 - a - b) \cdot \mathbf{v}_2$$

- Changing these parameters moves a point inside the triangle



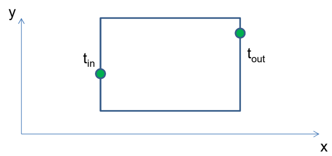
32

Different Intersection Techniques

- Ray vs. Plane
- Ray vs. Triangle
- Ray vs. Axis-aligned Box
- Ray vs. Sphere

Ray / Box Intersection

- How to intersect a ray with an axis-aligned box? How to find the intersection parameters t_{in} and t_{out}?



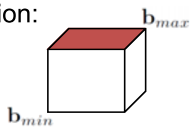
- Hints:
  - How to represent an axis-aligned box?
  - Simplify the box!
  - Try to find a solution in 2D first

34

Ray / Box Intersection

- Axis-aligned box representation:

$\mathbf{B} = [\mathbf{b}_{min}, \mathbf{b}_{max}]$



- Plane equations:

$$\boxed{x - x_{min} = 0}$$
$$x - x_{max} = 0$$
$$y - y_{min} = 0$$
$$\boxed{y - y_{max} = 0}$$
$$z - z_{min} = 0$$
$$z - z_{max} = 0$$

35

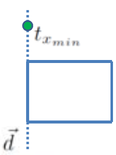
Ray / Box Intersection

- Substitution of ray into plane equations reveals intersection parameter, e.g. for left plane

$$o_x + t \cdot d_x - x_{min} = 0 \quad \Rightarrow \quad t = \frac{x_{min} - o_x}{d_x}$$

- Repeat for all six sides

$$t_{x_{min}},\ t_{x_{max}}$$
$$t_{y_{min}},\ t_{y_{max}}$$
$$t_{z_{min}},\ t_{z_{max}}$$



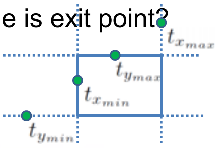
$\mathbf{r}(t) = \mathbf{o} + t \cdot \vec{d}$

36

Ray / Box Intersection

- Given: $t_{x_{min}}, t_{x_{max}}$
  $t_{y_{min}}, t_{y_{max}}$
  $t_{z_{min}}, t_{z_{max}}$
- Which one is entry, which one is exit point?
  – e.g.for x

$$t_{in,x} = \min(t_{x_{min}}, t_{x_{max}})$$
$$t_{out,x} = \max(t_{x_{min}}, t_{x_{max}})$$



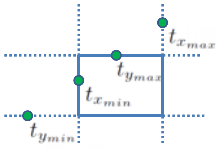
Ray / Box Intersection

- Which one is entry, which one is exit point?
  – e.g.for x

$$t_{in,x} = \min(t_{x_{min}}, t_{x_{max}})$$
$$t_{out,x} = \max(t_{x_{min}}, t_{x_{max}})$$

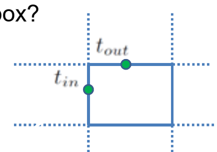
  – repeat for y and z axis
- Find global entry and exit

$$t_{in} = \max(t_{in,x}, t_{in,y}, t_{in,z})$$
$$t_{out} = \min(t_{out,x}, t_{out,y}, t_{out,z})$$



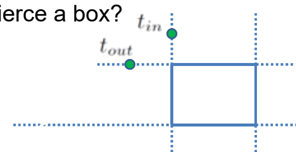
Ray / Box Intersection

- We know the global entry and exit points t_{in} and t_{out}
- Does that mean we are done?
  – We always have an entry and exit point
  – Does a ray always pierce a box?



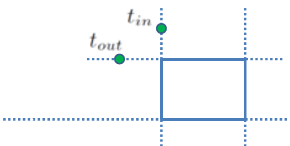
Ray / Box Intersection

- We know the global entry and exit points t_{in} and t_{out}
- Does that mean we are done?
  – We always have an entry and exit point
  – Does a ray always pierce a box?
- What if it misses?



Ray / Box Intersection

- Check for valid t_{in} and t_{out}
  – if ($t_{in} > t_{out}$) or ($t_{out} < 0$)
  – then ray misses



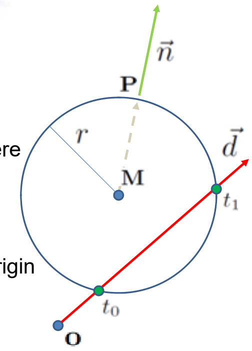
Different Intersection Techniques

- Ray vs. Plane
- Ray vs. Triangle
- Ray vs. Axis-aligned Box
- Ray vs. Sphere

Ray / Sphere Intersection

- Sphere defined by
  - Centre $\mathbf{M}$
  - Radius r
  - Normal $\vec{n} = |\mathbf{P} - \mathbf{M}|$

  For any point $\mathbf{P}$ on the sphere
- Simplifying idea:
  - Move sphere AND ray by
    - $\mathbf{M}$ to place sphere at origin
  - Intersection parameters t_0 and t_1 do not change



43

Sidenote

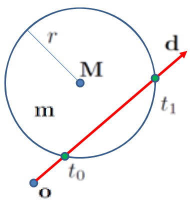
- Therefore, we introduce a marking to formulae that you don't need to learn by heart. The marking is a yellow border with an encircled „i" in it, like this:

  ⓘ or ⓘ

- Please note that we might use the same color just to mark something. So please pay attention to the „i" symbol.
- Also: Though you might not have to learn the marked formulae you should still be able to grasp the idea behind it!

44

Ray / Sphere Intersection



Implicit representation:

$$x^2 + y^2 + z^2 - r^2 = 0$$

Substitute ray into equation:

$$(o_x + td_x)^2 + (o_y + td_y)^2 + (o_z + td_z)^2 - r^2 = 0$$

General quadratic equation in t:

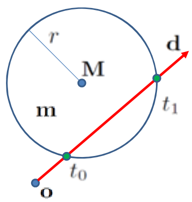
$$At^2 + Bt + C = 0$$

with:

$$\begin{aligned} A &= d_x^2 + d_y^2 + d_z^2 \\ B &= 2(d_x o_x + d_y o_y + d_z o_z) \\ C &= o_x^2 + o_y^2 + o_z^2 - r^2 \end{aligned}$$ ⓘ

45

Ray / Sphere Intersection



Two possible solutions:

$$t_{0,1} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$ ⓘ

$B^2 - 4AC$ is called the discriminant

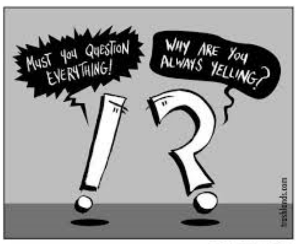
If discriminant < 0: No real solution

If discriminant = 0: One solution
Ray is tangent on sphere

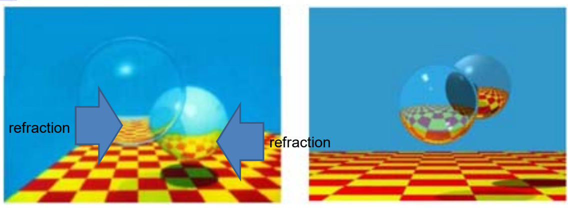
If discriminant > 0: Two solutions

46

Questions?



more awesome pictures at THEMETAPICTURE.COM



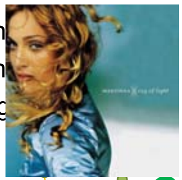
Turner Whitted 1980

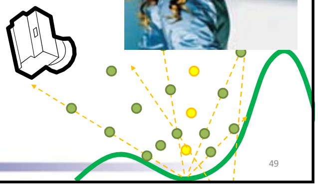
Developed the first recursive ray tracer to display reflections and refractions

48

Theory

- Simulation of light as photons bouncing in the scene
- Physically valid theoretical foun
- Too few photons reach the cam
- Hence the backward ray tracing





49

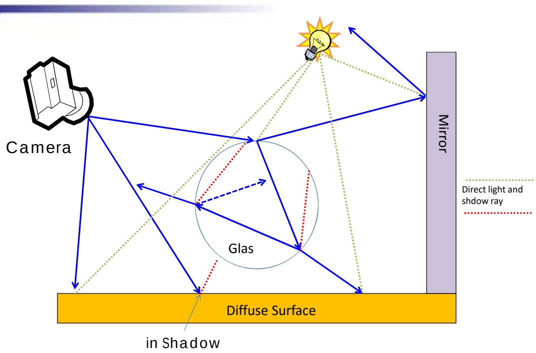
Recursive Ray Tracing

aka. Backward Ray Tracing or Whitted-Style Ray Tracing [Whitted1980]

Algorithm:
- For all pixels
  - Compute ray through pixel
  - Compute first hitpoint
  - Compute direct lighting and evaluate shadows
  - Compute reflected and refracted ray
  - Recursively continue for both rays

51

Ray Tracing Principle



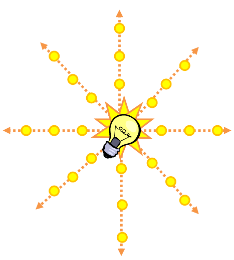
52

Light Sources

53

Point Light Source

- Emits light equally in all directions
- Infinitely small light source
- Defined by
  - Position
  - Color (Spectrum)

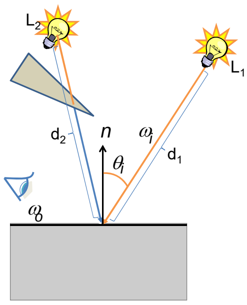


$$I(\theta) = I_0$$

54

Direct Light and Shadows

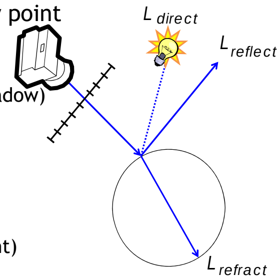
- Given
  - Hitpoint
  - Normal n at hitpoint
  - BRDF / Material
  - Point Light Source
- Wanted
  - Radiance resulting from direct light
- Shadow test
  - Test if any object lies between hitpoint and light source
  - anyIntersect() with max. distance d
  - If in shadow skip further computation for this light source



55

Local View

- Radiance (Color) at any point is the sum of
  - Radiance from direct light (if not in shadow)
  - Plus radiance from reflected light (if surface is glossy)
  - Plus radiance from refracted light (if surface transmits light)



$$L_{\text{pixel}} = L_{\text{direct}} + L_{\text{reflect}} + L_{\text{refract}}$$

56

Pseudocode Ray Tracing

A ray tracer has two main functions:
- trace(): computes hitpoint of ray with scene
- shade(): computes color (radiance) for a given point in the scene

These functions recursively call each other

```
RayTrace ( view )
   {

   for( y=0; y<view.yres; ++y ){
   for( x=0; x<view.xres; ++x){
        ComputeRay( x, y, view, &ray );

        Trace( 0, ray, &color );

        PutPixel( x, y, color );
   }}}

Trace ( level, ray, &color )
   {
   if( Intersect( level, ray, max, &hit) )
        Shade( level, hit, &color );
   else
        color = BackgroundColor;
   }
```

57

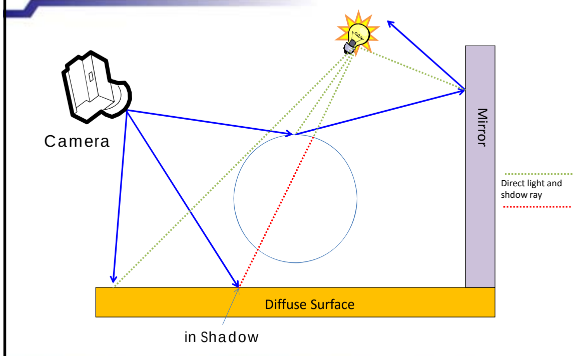
Pseudocode Ray Tracing

```
Shade ( level, hit, &color ){

    ComputeDirectLight ( hit, &directColor );

    ComputeReflectedRay( hit, &reflectedRay );

    Trace( level+1, reflectedRay, &reflectedColor );

    ComputeRefractedRay( hit, &refractedRay );

    Trace( level+1, refractedRay, &refractedColor );

    color = directColor + reflection * reflectedColor + transmission * refractedColor;
    }
```

- Stopping criterion:
  - maximum recursion depth: `level < maxLevel`
    - up to 2^{maxLevel} rays (!)
  - otherwise possible infinite loop
    - recursion stops only at diffuse surfaces or background

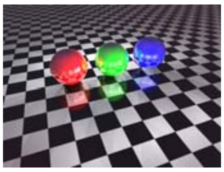
58

Ray Tracing Principle - Reflection

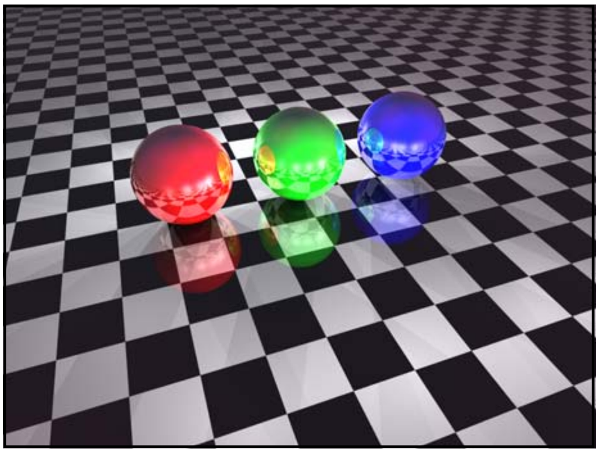


63

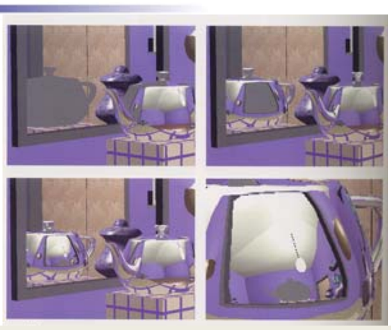
Examples



64

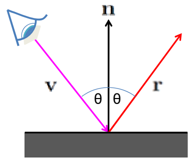


Recursion Depth



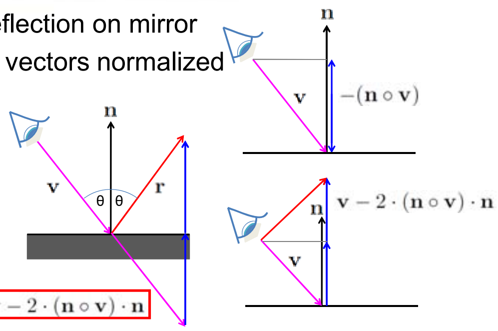
Reflection

- Reflection on mirror
- All vectors normalized



Reflection

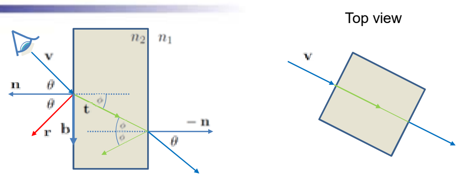
- Reflection on mirror
- All vectors normalized



$$\boxed{\mathbf{r} = \mathbf{v} - 2 \cdot (\mathbf{n} \circ \mathbf{v}) \cdot \mathbf{n}}$$

Questions?

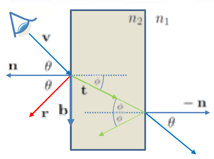
Refraction



- Refraction Index:
  - Air: n = 1.0
  - Water: n = 1.33
  - Glas: n = 1.5 - 1.7
  - Diamond: n = 2.4
- Snell's Law
  $$n_1 \sin \theta = n_2 \sin \phi$$
- The transmitted ray always lies in plane spanned by viewing vector and normal

How to compute **t**? And how to compute it fast?

Refraction



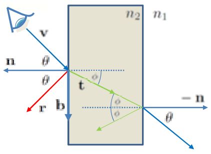
- Snell's Law
  $$n_1 \sin \theta = n_2 \sin \phi$$

- How to compute **t**? Fast!

1. Derive a refraction relationship for cosine Φ
2. Describe **t** in the orthonormal basis spanned by **n** and **b**
   (note: we don't know **b** yet)

For simplicity:
Assume all vectors are normalized, i.e. length = 1

Refraction



- Snell‘s Law
  $n_1 \sin\theta = n_2 \sin\phi$
- Trigonometric identity
  $\boxed{\sin^2\alpha + \cos^2\alpha = 1}$
  derived from Pythagorean theorem

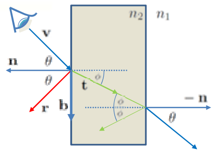
- How to compute **t**?
1. Derive a refraction relationship for cosine Φ
   (looks complicated, is simple)

$$
\begin{aligned}
n_1 \sin\theta &= n_2 \sin\phi \\
n_1^2 \sin^2\theta &= n_2^2 \boxed{\sin^2\phi} \\
n_1^2 \sin^2\theta &= n_2^2 \boxed{1-\cos^2\phi} \\
\frac{n_1^2 \sin^2\theta}{n_2^2} &= 1-\cos^2\phi \\
1-\frac{n_1^2 \sin^2\theta}{n_2^2} &= \cos^2\phi
\end{aligned}
$$

$$\boxed{1-\frac{n_1^2(1-\cos^2\theta)}{n_2^2} = \cos^2\phi}$$???

73

Refraction



- Snell‘s Law
  $n_1 \sin\theta = n_2 \sin\phi$
- Cosine relations
  $\boxed{1-\frac{n_1^2(1-\cos^2\theta)}{n_2^2} = \cos^2\phi}$

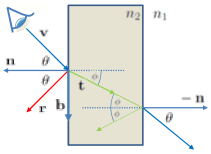
Assume all vectors are normalized, i.e. length = 1

2. Describe **t** in the orthonormal basis spanned by **n** and **b**
   – Given: **n,v**

- Describe **v** in this basis
  $\mathbf{v} = \sin\theta\mathbf{b} - \cos\theta\mathbf{n}$
- Derive **b**
  $\mathbf{b} = \frac{\mathbf{v}+\cos\theta\mathbf{n}}{\sin\theta}$
- Describe **t** in same basis
  $\mathbf{t} = \sin\phi\mathbf{b} - \cos\phi\mathbf{n}$
  But Φ is unknown!

74

Refraction



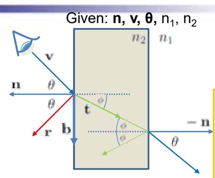
$\boxed{\mathbf{b} = \frac{\mathbf{v}+\cos\theta\mathbf{n}}{\sin\theta}}$

- Describe **t** in same basis

- Snell‘s Law
  $\boxed{n_1 \sin\theta = n_2 \sin\phi}$
- Cosine relations
  $\boxed{1-\frac{n_1^2(1-\cos^2\theta)}{n_2^2} = \cos^2\phi}$
- Dot product
  $\boxed{\mathbf{a}\circ\mathbf{b} = |\mathbf{a}|\cdot|\mathbf{b}|\cdot\cos\alpha}$

Refraction

Given: **n, v, θ,** n_1, n_2



- Snell‘s Law
  $\boxed{n_1 \sin\theta = n_2 \sin\phi}$
- Cosine relations
  $\boxed{1-\frac{n_1^2(1-\cos^2\theta)}{n_2^2} = \cos^2\phi}$
- Dot product
  $\boxed{\mathbf{a}\circ\mathbf{b} = |\mathbf{a}|\cdot|\mathbf{b}|\cdot\cos\alpha}$

$\boxed{\mathbf{b} = \frac{\mathbf{v}+\cos\theta\mathbf{n}}{\sin\theta}}$

- Describe **t** in same basis

$$
\begin{aligned}
\mathbf{t} &= \sin\phi\boxed{\mathbf{b}} - \cos\phi\mathbf{n} \\
&= \boxed{\sin\phi}\boxed{\frac{\mathbf{v}+\cos\theta\cdot\mathbf{n}}{\sin\theta}} - \cos\phi\cdot\mathbf{n} \\
&= \boxed{\frac{n_1\sin\theta}{n_2}}\cdot\frac{\mathbf{v}+\cos\theta\cdot\mathbf{n}}{\sin\theta} - \cos\phi\cdot\mathbf{n} \\
&= \frac{n_1}{n_2}\cdot(\mathbf{v}+\cos\theta\cdot\mathbf{n}) - \boxed{\cos\phi}\,\mathbf{n} \\
&= \frac{n_1}{n_2}(\mathbf{v}+\boxed{\cos\theta}\cdot\mathbf{n}) - \mathbf{n}\cdot\boxed{\sqrt{1-\frac{n_1^2(1-\boxed{\cos^2\theta})}{n_2^2}}} \\
&= \frac{n_1}{n_2}(\mathbf{v}+\boxed{(-\mathbf{v}\circ\mathbf{n})}\cdot\mathbf{n}) - \mathbf{n}\cdot\sqrt{1-\frac{n_1^2(1-\boxed{(\mathbf{v}\circ\mathbf{n})^2})}{n_2^2}} \\
&= \frac{n_1}{n_2}(\mathbf{v}-(\mathbf{v}\circ\mathbf{n})\cdot\mathbf{n}) - \mathbf{n}\cdot\sqrt{1-\frac{n_1^2(1-(\mathbf{v}\circ\mathbf{n})^2)}{n_2^2}}
\end{aligned}
$$

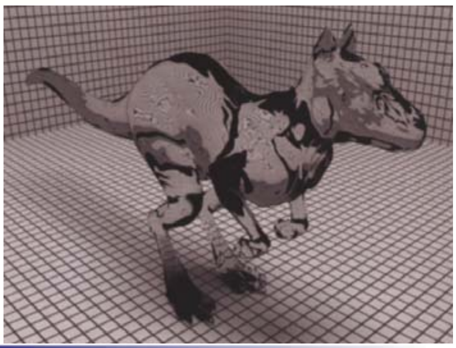
76

Conclusion on Refraction

$$\mathbf{t} = \sin\phi\mathbf{b} - \cos\phi\mathbf{n} = \frac{n_1}{n_2}(\mathbf{v}-(\mathbf{v}\circ\mathbf{n})\cdot\mathbf{n}) - \mathbf{n}\cdot\sqrt{1-\frac{n_1^2(1-(\mathbf{v}\circ\mathbf{n})^2)}{n_2^2}}$$

„I have to apologize for the formulae here. But these are not mine, so don't blame me.“
- Liang Wang

Questions



78

Recursive Ray Tracing

```
RayTrace ( view )
{
  for( y=0; y<view.yres; ++y ){
    for( x=0; x<view.xres; ++x){
      ComputeRay( x, y, view, &ray );
      Trace( 0, ray, &color );
      PutPixel( x, y, color );
    }
  }
}

Trace ( level, ray, &color )
{
  if( Intersect( level, ray, max, &hit) )
    Shade( level, hit, &color );
  else
    color = BackgroundColor;
}
```

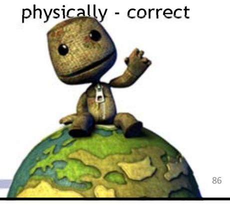
```
Shade ( level, hit, &color )
{
  for each light source
    ComputeDirectLight ( hit, &directColor );

  if material reflects && (level < maxLevel)
    ComputeReflectedRay( hit, &reflectedRay );
    Trace( level+1, reflectedRay,
   &reflectedColor );

  if material refracts && (level < maxLevel)
    ComputeRefractedRay( hit, &refractedRay );
    Trace( level+1, refractedRay,
   &refractedColor );

  color = directColor + reflection *
   reflectedColor + transmission *
   refractedColor;
}
```

85

Summary

- Linear Algebra Primer
- Intersection Methods
- Recursive (Whitted-style) Ray Tracing
- Basics
- Shadows
- Reflection
- Refraction

- Next time:
  - How to make RT fast?
  - How to make RT beautiful?
  - How to make RT physically - correct



86