3D Rendering and Ray Casting

Michael Kazhdan

(601.457/657)

HB Ch. 13.7, 14.6
FvDFH 15.5, 15.10
Rendering

• Generate an image from geometric primitives

Geometric Primitives (3D)
Rendering

- Generate an image from geometric primitives
What issues must be addressed by a 3D rendering system?
Overview

- 3D scene representation
- 3D viewer representation
- What do we see?
- How does it look?

How is the 3D scene described in a computer?
3D Scene Representation

- Scene is usually approximated by 3D primitives
  - Point
  - Line segment
  - Triangles
  - Polygon
  - Curved surface
  - Solid object
  - etc.
3D Point

- Specifies a location
3D Point

- Specifies a location
  - Represented by three coordinates
  - Infinitely small

```cpp
template< unsigned int Dim >
struct Point
{
    float c[Dim];
};
```

$(x, y, z)$

Origin
3D Vector

- Specifies a direction and a magnitude
3D Vector

- Specifies a direction and a magnitude
  - Represented by three coordinates
  - Magnitude $\|\vec{v}\| = \sqrt{dx^2 + dy^2 + dz^2}$
  - Has no location

```c
template<unsigned int Dim>
struct Vector
{
  float d[Dim];
  Vector &operator += ( const Vector );
  ...
  float norm( void ) const;
  ...
};
```

$\vec{v} = (dx, dy, dz)$
3D Vector

- Specifies a direction and a magnitude
  - Represented by three coordinates
  - Magnitude \( \|\vec{v}\| = \sqrt{dx^2 + dy^2 + dz^2} \)
  - Has no location

- Dot product of two vectors
  - \( \langle \vec{v}_1, \vec{v}_2 \rangle = dx_1 \cdot dx_2 + dy_1 \cdot dy_2 + dz_1 \cdot dz_2 \)
  - \( \langle \vec{v}_1, \vec{v}_2 \rangle = \|\vec{v}_1\| \cdot \|\vec{v}_2\| \cdot \cos \theta \)

- Cross product of two 3D vectors
  - \( \vec{v}_1 \times \vec{v}_2 = \text{Vector normal to } \nu_1 \text{ and } \nu_2 \)
  - \( \|\vec{v}_1 \times \vec{v}_2\| = \|\vec{v}_1\| \cdot \|\vec{v}_2\| \cdot \sin \theta \)
  - Aligned with the right-hand-rule
Cross Product: Review

- Let
  \[ \vec{v}_i = (dx_i, dy_i, dz_i) \quad \text{with} \quad i \in \{1,2,3\} \]

Then \( \vec{v}_1 = \vec{v}_2 \times \vec{v}_3 \) is expressed as:
  - \( dx_1 = dy_2 \cdot dz_3 - dz_2 \cdot dy_3 \)
  - \( dy_1 = dz_2 \cdot dx_3 - dx_2 \cdot dz_3 \)
  - \( dz_1 = dx_2 \cdot dy_3 - dy_2 \cdot dx_3 \)

- Anti-symmetric: \( \vec{v} \times \vec{w} = -\vec{w} \times \vec{v} \)

Can similarly define the cross-product of \((d - 1)\) vectors in \(d\) dimensional space.
Cross Product: Review

\[ \| \vec{v} \times \vec{w} \| = \| \vec{v} \| \cdot \| \vec{w} \| \cdot \sin \theta \]

Geometrically speaking, we can consider the parallelogram defined by \( \vec{v} \) and \( \vec{w} \).

The area of the parallelogram is the product of the base and the height.

- \textit{base} = \( \| \vec{v} \| \)
- \textit{height} = \( \sin(\theta) \cdot \| \vec{w} \| \)

\[ \Rightarrow \text{Area}(\vec{v}, \vec{w}) = \| \vec{v} \| \cdot \| \vec{w} \| \cdot \sin \theta = \| \vec{v} \times \vec{w} \| \]
3D Line Segment

- Linear path between two points
3D Line Segment

- Linear path between two points
  - Parametric representation:
    » \( p(t) = p_1 + t \cdot (p_2 - p_1), \quad (0 \leq t \leq 1) \)

```cpp
template< unsigned int Dim >
struct Segment
{
    Point< Dim > p1, p2;
};
```

Origin
3D Ray

- Line segment with one endpoint at infinity
  - Parametric representation:
    \[ p(t) = p_1 + t \cdot \vec{v}, \quad (0 \leq t < \infty) \]

```cpp
template< unsigned int Dim >
struct Ray
{
    Point< Dim > p1;
    Vector< Dim > v;
};
```
3D Line

- Line segment with both endpoints at infinity
  - **Parametric representation:**
    - \( p(t) = p_1 + t \cdot \vec{v}, \quad (-\infty < t < \infty) \)

```cpp
template< unsigned int Dim >
struct Line
{
    Point< Dim > p1;
    Vector< Dim > v;
};
```
Surfaces in 3D

So far, we have represented geometry parametrically – defining a function which takes in a parameter and returns a position on the geometry.

(2D) surfaces in 3D can also be represented by an implicit function -- a function $\Phi: \mathbb{R}^3 \rightarrow \mathbb{R}$ which:

- Equals zero on the surface
- Is positive outside the surface
- Is negative inside the surface

This makes it easy to evaluate if a point is on the surface.
3D Plane

• A linear combination of three points
3D Plane

- A linear combination of three points
  - Implicit representation:
    » $\Phi(p) = ap_x + bp_y + cp_z - d = 0$
    » $\Phi(p) = \langle p, \vec{n} \rangle - d = 0$

  Template< unsigned int Dim >
  struct Plane
  {
    Vector n;
    float d;
  };

  - $\vec{n}$ is the plane normal
    » (May be) unit-length vector
    » Perpendicular to plane
  - $d$ is the signed (weighted) distance of the plane from the origin.
3D Polygon

- Area “inside” a sequence of coplanar points
  - Triangle
  - Quadrilateral
  - Convex
  - Star-shaped
  - Concave
  - Self-intersecting

Template\(<\text{unsigned int}\text{ Dim }\rangle\
\text{struct Polygon}\n\{\
  \text{Point}<\text{ Dim }\rangle *\text{points};
  \text{size}_\text{t} \text{ size};
\};\

Points are in counter-clockwise order
- Holes (use > 1 polygon struct)
3D Polygon

- Area “inside” a sequence of coplanar points
  - Triangle
  - Quadrilateral
  - Convex
  - Star-shaped
  - Concave
  - Self-intersecting

Note: If a 3D polygon has more than three points, the points don’t have to be coplanar, so the “interior” may not be well-defined.
3D Sphere

• All points at distance $r$ from center point $c = (c_x, c_y, c_z)$
  
  ◦ Implicit representation:
    
    » $\Phi(p) = \|p - c\|^2 - r^2 = 0$
  
  ◦ Parametric representation:
    
    » $x(\phi, \theta) = r \cdot \cos \phi \cdot \sin \theta + c_x$
    » $y(\phi, \theta) = r \cdot \cos \phi \cdot \sin \theta + c_y$
    » $z(\theta, \phi) = r \cdot \sin \phi + c_z$

```c
template< unsigned int Dim >
struct Sphere
{
    Point< Dim > center;
    float radius;
};
```
Other 3D primitives

- Cone
- Cylinder
- Ellipsoid
- Box
- Etc.
3D Geometric Primitives

• More detail on 3D modeling later in course
  ◦ Point
  ◦ Line segment
  ◦ Triangle
  ◦ Polygon
  ◦ Curved surface
  ◦ Solid object
  ◦ etc.
Overview

• 3D scene representation
• 3D viewer representation
• What do we see?
• How does it look?

How is the viewing device described in a computer?
Camera Models

- The most common model is pin-hole camera
  - All captured light rays arrive along paths toward focal point without lens distortion (everything is in focus)
Camera Parameters

- What are the parameters of a camera?
Camera Parameters

• Position
  ◦ Eye position: Point< 3 > eye

• Orientation
  ◦ View direction: Vector< 3 > view
  ◦ Up direction: Vector< 3 > up

• Aperture
  ◦ Field of view angle: float xFov , yFov
  ◦ Resolution of film plane: int width , height
Other Models: Depth of Field

Close Focused

Distance Focused

P. Haeberli
Other Models: Motion Blur

- Mimics effect of open camera shutter
- Gives perceptual effect of high-speed motion
- Generally involves temporal super-sampling
Traditional Pinhole Camera

- The film sits behind the pinhole of the camera.
- Rays come in from the outside, pass through the pinhole, and hit the film plane.
Traditional Pinhole Camera

- The film sits behind the pinhole of the camera.
- Rays come in from the outside, pass through the pinhole, and hit the film plane.

Photograph is upside down
Virtual Camera

• The film sits in front of the pinhole of the camera.
• Rays come in from the outside, pass through the film plane, and hit the pinhole.
Virtual Camera

- The film sits in front of the pinhole of the camera.
- Rays come in from the outside, pass through the film plane, and hit the pinhole.

Photograph is right side up
Overview

• 3D scene representation
• 3D viewer representation

• Ray Casting
  ◦ Where are we looking?
  ◦ What do we see?
  ◦ How does it look?
Ray Casting

• For each sample …
  ○ **Where**: Construct ray from eye through view plane
  ○ **What**: Find **first** surface intersected by ray through pixel
  ○ **How**: Compute color sample based on surface radiance
Ray Casting

• Simple implementation:

```c
Image RayCast( Camera camera , Scene scene , int width , int height )
{
    Image image( width , height );
    for( int j=0 ; j<height ; j++ ) for( int i=0 ; i<width ; i++ )
    {
        Ray< 3 > ray = ConstructRayThroughPixel( camera , i , j );
        Intersection hit = FindIntersection( ray , scene );
        image[i][j] = GetColor( hit );
    }
    return image;
}
```
Ray Casting

Where?

Image RayCast( Camera camera , Scene scene , int width , int height)
{
    Image image( width , height );
    for( int j=0 ; j<height ; j++ ) for( int i=0 ; i<width ; i++ )
    {
        Ray< 3 > ray = ConstructRayThroughPixel( camera , i , j );
        Intersection hit = FindIntersection( ray , scene );
        image[i][j] = GetColor( hit );
    }
    return image;
}
Constructing a Ray Through a Pixel

- **Up direction**
- **Towards**
- **View Plane**
- **Pixel** $p_0$
- **Pixel position** $p[i][j]$
Constructing a Ray Through a Pixel

The ray originates at $p_0$ (the eye position of the camera). So the equation for the ray is:

$$\text{Ray}(t) = p_0 + t \cdot \vec{v}$$
If the ray passes through the point $p[i][j]$, then the solution for (unit) $\mathbf{v}$ is:

$$\mathbf{v} = \frac{p[i][j] - p_0}{\|p[i][j] - p_0\|}$$
If $p[i][j]$ represents the $(i,j)$-th pixel of the image, what is its position?
Constructing Ray Through a Pixel

- 2D Example: Side view of camera at $p_0$
  - Where is the $i$-th pixel, $p[i]$, with $i \in [0, \text{height})$?

$\theta = \text{field of view angle (given)}$

$d = \text{distance to view plane (arbitrary = you pick)}$

\[
p_1 = p_0 + d \cdot \text{towards} - d \cdot \tan\frac{\theta}{2} \cdot \text{up}
\]

\[
p_2 = p_0 + d \cdot \text{towards} + d \cdot \tan\frac{\theta}{2} \cdot \text{up}
\]

\[
p[i] = p_1 + \left(\frac{i + 0.5}{\text{height}}\right) \cdot (p_2 - p_1)
\]
Constructing Ray Through a Pixel

Figuring out how to do this in 3D is assignment 2
Constructing Ray Through a Pixel

Figuring out how to do this in 3D is assignment 2

Note:
- Given the vertical field of view angle, $\theta_v$
Constructing Ray Through a Pixel

Figuring out how to do this in 3D is assignment 2

Note:

- Given the vertical field of view angle, $\theta_v$
- And the aspect ratio, $ar = \frac{\text{height}}{\text{width}}$
Constructing Ray Through a Pixel

Figuring out how to do this in 3D is assignment 2

Note:

- Given the vertical field of view angle, $\theta_v$
- And the aspect ratio, $ar = \frac{height}{width}$

The horizontal field of view angle, $\theta_h$, satisfies:

$$\frac{\tan(\theta_v/2)}{\tan(\theta_h/2)} = ar$$
Ray Casting

What?

Image RayCast( Camera camera, Scene scene, int width, int height)
{
    Image image( width, height);
    for( int j=0; j<height; j++ ) for( int i=0; i<width; i++ )
    {
        Ray< 3 > ray = ConstructRayThroughPixel( camera, i, j );
        Intersection hit = FindIntersection( ray, scene );
        image[i][j] = GetColor( hit );
    }
    return image;
}
Ray-Scene Intersection

- Intersections with geometric primitives
  - Sphere
  - Triangle
Ray-Sphere Intersection

Ray: \( p(t) = p_0 + t \cdot \vec{v}, \quad (0 \leq t < \infty) \)

Sphere: \( \Phi(p) = \|p - c\|^2 - r^2 = 0 \)
Ray-Sphere Intersection

Ray: $p(t) = p_0 + t \cdot \vec{v}$, \hspace{1em} (0 ≤ t < ∞)
Sphere: $\Phi(p) = ||p - c||^2 - r^2 = 0$

Substituting for $p$, we get:
$\Phi(t) = ||p_0 + t \cdot \vec{v} - c||^2 - r^2 = 0$

Solve quadratic equation:
$a \cdot t^2 + b \cdot t + c = 0$

where:
$a = 1$
$b = 2\langle \vec{v}, p_0 - c \rangle$
$c = ||p_0 - c||^2 - r^2$
Ray-Sphere Intersection

Ray: \( p(t) = p_0 + t \cdot \vec{v}, \ (0 \leq t < \infty) \)
Sphere: \( \Phi(p) = \|p - c\|^2 - r^2 = 0 \)

Substituting for \( p \), we get:
\[
\Phi(t) = \|p_0 + t \cdot \vec{v} - c\|^2 - r^2 = 0
\]

Solve quadratic equation:
\[
a \cdot t^2 + b \cdot t + c = 0
\]
where:

Generally, there are two solutions to the quadratic equation, giving two points of intersection, \( p \) and \( p' \). Want to return the first positive hit.
Ray-Sphere Intersection

- Need normal vector at intersection for lighting calculations:

\[ \vec{n} = \frac{\vec{p} - \vec{c}}{||\vec{p} - \vec{c}||} \]
Ray-Sphere Intersection

• More generally, if the shape is given as the set of points \( p \) satisfying:

\[
\Phi(p) = 0
\]

for some function \( \Phi: \mathbb{R}^3 \to \mathbb{R} \), then the normal of the surface will be parallel to the gradient.
Ray-Scene Intersection

• Intersections with geometric primitives
  ◦ Sphere
  » Triangle
Ray-Triangle Intersection

1. Intersect ray with plane
2. Check if the point is inside the triangle
Ray-Plane Intersection

Ray: \( p(t) = p_0 + t \cdot \vec{v}, \quad (0 \leq t < \infty) \)
Plane: \( \Phi(p) = \langle p, \vec{n} \rangle - d = 0 \)

Substituting for \( p \), we get:
\[ \Phi(t) = \langle p_0 + t \cdot \vec{v}, \vec{n} \rangle - d = 0 \]

Solution:
\[ t = -\frac{\langle p_0, \vec{n} \rangle - d}{\langle \vec{v}, \vec{n} \rangle} \]

What are the implications of \( \langle \vec{v}, \vec{n} \rangle = 0 \)?

Algebraic Method
Ray-Triangle Intersection

- Check for point-triangle intersection parametrically

In general, given \( p \in \mathbb{R}^3 \) and given three points \( \{v_1, v_2, v_3\} \subset \mathbb{R}^3 \) (in general position) we can solve for \( \alpha, \beta, \gamma \in \mathbb{R} \) such that:

\[
p = \alpha v_1 + \beta v_2 + \gamma v_3
\]

\( p \) is in the plane spanned by \( \{v_1, v_2, v_3\} \) if and only if (iff.):

\[
\alpha + \beta + \gamma = 1
\]

\( p \) is inside the triangle with vertices \( \{v_1, v_2, v_3\} \) iff.:

\[
\alpha, \beta, \gamma \geq 0
\]
Ray-Triangle Intersection

• Check for point-triangle intersection parametrically

In general, given $p \in \mathbb{R}^3$ and given three points $\{v_1, v_2, v_3\} \subset \mathbb{R}^3$ (in general position) we can solve for $\alpha, \beta, \gamma \in \mathbb{R}$ such that:

$$p = \alpha v_1 + \beta v_2 + \gamma v_3$$

Naively, to get $\alpha, \beta, \gamma$, we could try to solve the system:

$$
\begin{pmatrix}
    v_1^x & v_2^x & v_3^x \\
    v_1^y & v_2^y & v_3^y \\
    v_1^z & v_2^z & v_3^z
\end{pmatrix}
\begin{pmatrix}
    \alpha \\
    \beta \\
    \gamma
\end{pmatrix} =
\begin{pmatrix}
    p^x \\
    p^y \\
    p^z
\end{pmatrix}
$$
Ray-Triangle Intersection

• Check for point-triangle intersection parametrically

In general, given \( p \in \mathbb{R}^3 \) and given three points \( \{v_1, v_2, v_3\} \subset \mathbb{R}^3 \) (in general position) we can solve for \( \alpha, \beta, \gamma \in \mathbb{R} \) such that:

\[ p = \alpha v_1 + \beta v_2 + \gamma v_3 \]

Naively, to get \( \alpha, \beta, \gamma \), we could try to solve the system:

\[
\begin{pmatrix}
  v_1^x & v_2^x & v_3^x \\
  v_1^y & v_2^y & v_3^y \\
  v_1^z & v_2^z & v_3^z
\end{pmatrix}
\begin{pmatrix}
  \alpha \\
  \beta \\
  \gamma
\end{pmatrix} =
\begin{pmatrix}
  p^x \\
  p^y \\
  p^z
\end{pmatrix}
\]

\[ \Leftrightarrow \]

\[
\begin{pmatrix}
  \alpha \\
  \beta \\
  \gamma
\end{pmatrix} =
\begin{pmatrix}
  v_1^x & v_2^x & v_3^x \\
  v_1^y & v_2^y & v_3^y \\
  v_1^z & v_2^z & v_3^z
\end{pmatrix}^{-1}
\begin{pmatrix}
  p^x \\
  p^y \\
  p^z
\end{pmatrix}
\]
Ray-Triangle Intersection

- Check for point-triangle intersection parametrically

In general, given \( p \in \mathbb{R}^3 \) and given three points \( \{v_1, v_2, v_3\} \subset \mathbb{R}^3 \) (in general position) we can solve for \( \alpha, \beta, \gamma \in \mathbb{R} \) such that:

\[
p = \alpha v_1 + \beta v_2 + \gamma v_3
\]

Naively, to get \( \alpha, \beta, \gamma \), we could try to solve the system:

\[
\begin{pmatrix}
  v_1^x & v_2^x & v_3^x \\
v_1^y & v_2^y & v_3^y \\
v_1^z & v_2^z & v_3^z
\end{pmatrix}
\begin{pmatrix}
  \alpha \\
  \beta \\
  \gamma
\end{pmatrix}
= 
\begin{pmatrix}
  p^x \\
p^y \\
p^z
\end{pmatrix}
\]

\[
\iff
\begin{pmatrix}
  \alpha \\
  \beta \\
  \gamma
\end{pmatrix}
= 
\begin{pmatrix}
  v_1^x & v_2^x & v_3^x \\
v_1^y & v_2^y & v_3^y \\
v_1^z & v_2^z & v_3^z
\end{pmatrix}^{-1}
\begin{pmatrix}
  p^x \\
p^y \\
p^z
\end{pmatrix}
\]

This will fail if the vertices \( \{v_1, v_2, v_3\} \) lie in a plane through the origin.
Ray-Triangle Intersection

Intuitively:

The weights $\alpha, \beta, \gamma \in \mathbb{R}$ describe how close $p$ is to $v_1, v_2, v_3 \in \mathbb{R}^3$.

If we consider the triangle opposite vertex $v_k$, $\{p, v_{k+1}, v_{k+2}\}$, the area of the triangle:

- Tends to zero as $p$ moves away from $v_k$
- Tends to the area of triangle $\{v_1, v_2, v_3\}$ as $p$ moves towards $v_k$. 
Ray-Triangle Intersection

Intuitively:

The weights $\alpha, \beta, \gamma \in \mathbb{R}$ describe how close $p$ is to $v_1, v_2, v_3 \in \mathbb{R}^3$.

$\Rightarrow$ Define $\alpha, \beta, \gamma \in \mathbb{R}$ in terms of the signed triangle area ratios:

$$
\alpha = \frac{\text{SignedArea}\{(p, v_2, v_3)\}}{\text{SignedArea}\{(v_1, v_2, v_3)\}}
$$

$$
\beta = \frac{\text{SignedArea}\{(p, v_3, v_1)\}}{\text{SignedArea}\{(v_1, v_2, v_3)\}}
$$

$$
\gamma = \frac{\text{SignedArea}\{(p, v_1, v_2)\}}{\text{SignedArea}\{(v_1, v_2, v_3)\}}
$$
Ray-Triangle Intersection

Recall:

Given vectors $\vec{w}_1, \vec{w}_2 \in \mathbb{R}^3$, the (unsigned) area of the parallelogram spanned by $\vec{w}_1$ and $\vec{w}_2$ is:

$$\text{ParallelogramArea}(\vec{w}_1, \vec{w}_2) = |\vec{w}_1 \times \vec{w}_2|$$

Assuming that we are given a unit vector $\vec{n} \in \mathbb{R}^3$ perpendicular to both $\vec{w}_1$ and $\vec{w}_2$:

$$\langle \vec{n}, \vec{w}_1 \rangle = \langle \vec{n}, \vec{w}_2 \rangle = 0$$

we can obtain the signed area (relative to $\vec{n}$) by taking the dot-product:

$$\text{SignedParallelogramArea}(\vec{w}_1, \vec{w}_2) = \langle \vec{w}_1 \times \vec{w}_2, \vec{n} \rangle$$
Ray-Triangle Intersection

Recall:

Given vectors $\vec{w}_1, \vec{w}_2 \in \mathbb{R}^3$, the (unsigned) area of the parallelogram spanned by $\vec{w}_1$ and $\vec{w}_2$ is:

$$\text{ParallelogramArea}(\vec{w}_1, \vec{w}_2) = |\vec{w}_1 \times \vec{w}_2|$$

Assuming that we are given a unit vector $\vec{n} \in \mathbb{R}^3$ perpendicular to both $\vec{w}_1$ and $\vec{w}_2$:

$$\langle \vec{n}, \vec{w}_1 \rangle = \langle \vec{n}, \vec{w}_2 \rangle = 0$$

we can obtain the signed area (relative to $\vec{n}$) by taking the dot-product:

$$\text{SignedParallelogramArea}(\vec{w}_1, \vec{w}_2) = \langle \vec{w}_1 \times \vec{w}_2, \vec{n} \rangle$$

$$\text{SignedTriangleArea}(\vec{w}_1, \vec{w}_2) = \langle \vec{w}_1 \times \vec{w}_2, \vec{n} \rangle / 2$$
Ray-Triangle Intersection

To compute the ratio of signed areas, set:

\[ \vec{w}_1 = \vec{v}_2 - \vec{v}_1 \]
\[ \vec{w}_2 = \vec{v}_3 - \vec{v}_1 \]
\[ \vec{n} = \frac{\vec{w}_1 \times \vec{w}_2}{|\vec{w}_1 \times \vec{w}_2|} \]

Then we get:

\[ \alpha = \frac{\langle (\vec{v}_2 - \vec{p}) \times (\vec{v}_3 - \vec{p}), \vec{n} \rangle}{\langle (\vec{v}_2 - \vec{v}_1) \times (\vec{v}_3 - \vec{v}_1), \vec{n} \rangle} / 2 \]
Ray-Triangle Intersection

To compute the ratio of signed areas, set:

\[ \mathbf{w}_1 = \mathbf{v}_2 - \mathbf{v}_1 \]
\[ \mathbf{w}_2 = \mathbf{v}_3 - \mathbf{v}_1 \]
\[ \mathbf{n} = \frac{\mathbf{w}_1 \times \mathbf{w}_2}{|\mathbf{w}_1 \times \mathbf{w}_2|} \]

Then we get:

\[ \alpha = \frac{\langle (\mathbf{v}_2 - \mathbf{p}) \times (\mathbf{v}_3 - \mathbf{p}), \mathbf{n} \rangle}{\langle (\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1), \mathbf{n} \rangle} / 2 \]

\[ \vdots \]

Note: If we flip the sign of \( \mathbf{n} \), the minus signs in the numerator/denominator cancel and we get the same weights.
Other Ray-Primitive Intersections

- Cone, cylinder, ellipsoid:
  - Similar to sphere

- Box
  - Intersect 3 front-facing planes, return closest

- Convex (planar) polygon
  - Find the intersection of the ray with the plane
  - Slightly more complex point-in-polygon test

- Concave (planar) polygon
  - Find the intersection of the ray with the plane
  - Markedly more complex point-in-polygon test