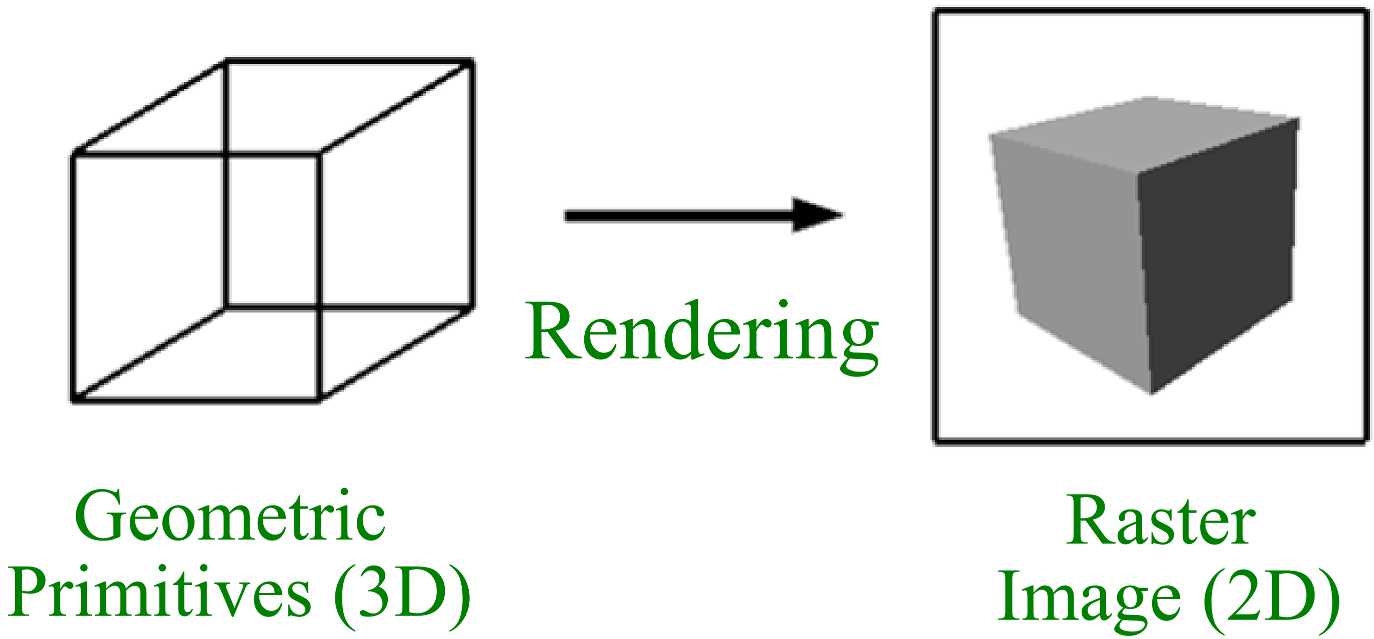
3D Rendering and Ray Casting

Michael Kazhdan

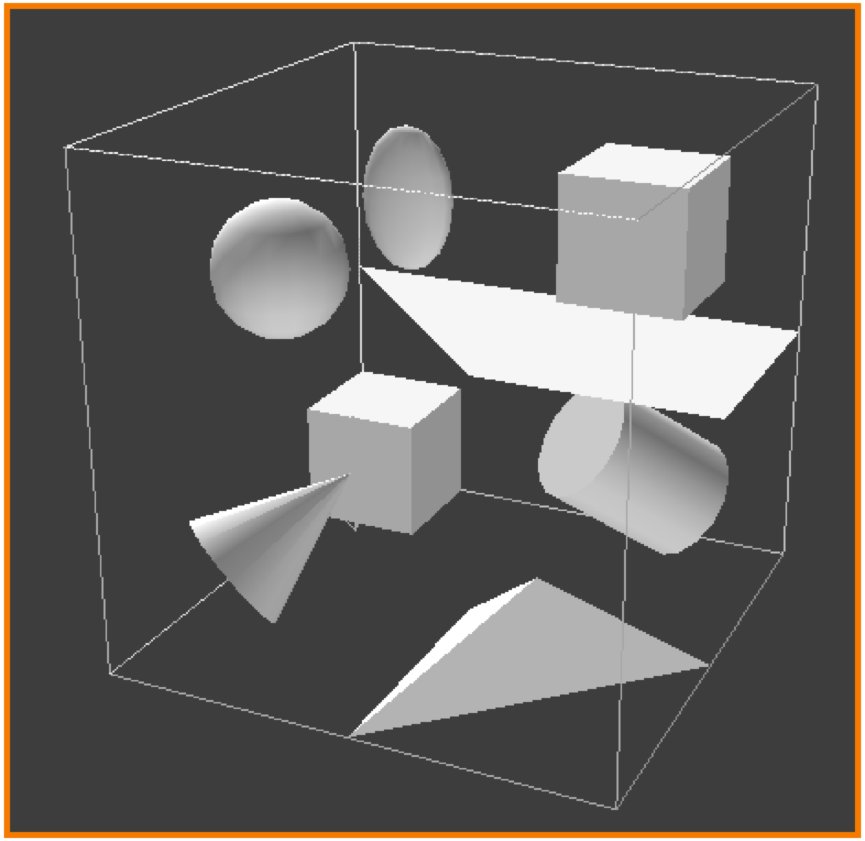
(601.457/657)

Rendering

- Generate an image from geometric primitives



3D Rendering Example

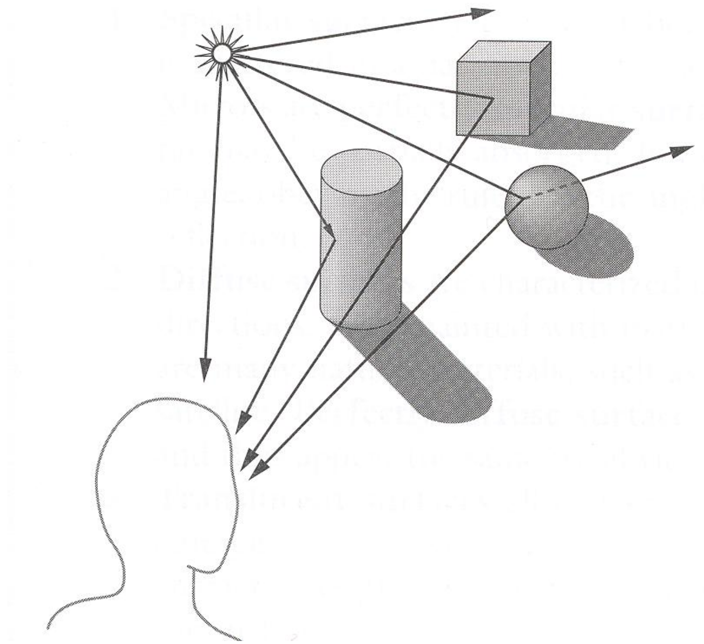


What issues must be addressed by a
3D rendering system?

Overview

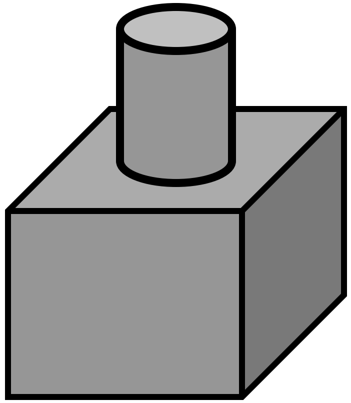
- 3D scene representation
- 3D viewer representation
- What do we see?
- How does it look?

How is the 3D scene described in a computer?



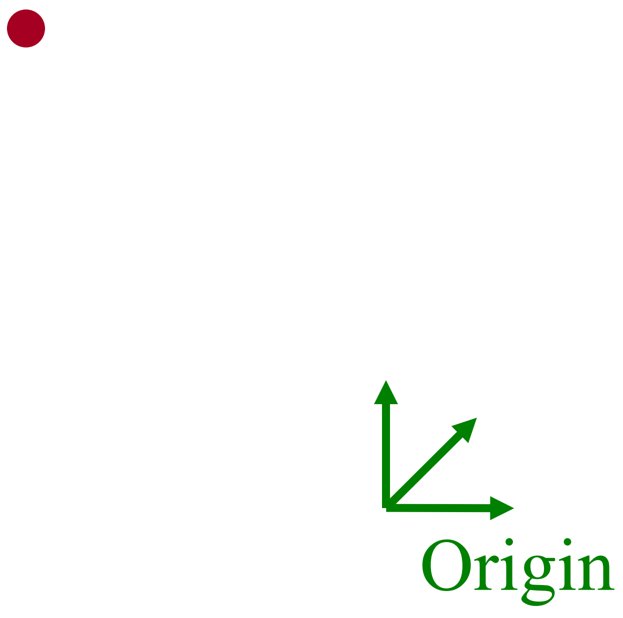
3D Scene Representation

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



3D Position

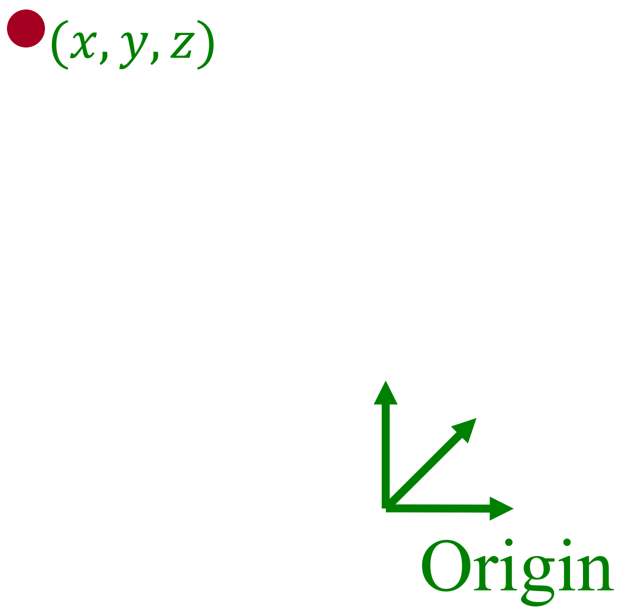
- Specifies a location



3D Position

- Specifies a location in space
  - Represented by three coordinates
  - No notion of length

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



3D Vector

- Specifies a direction (and magnitude)

3D Vector

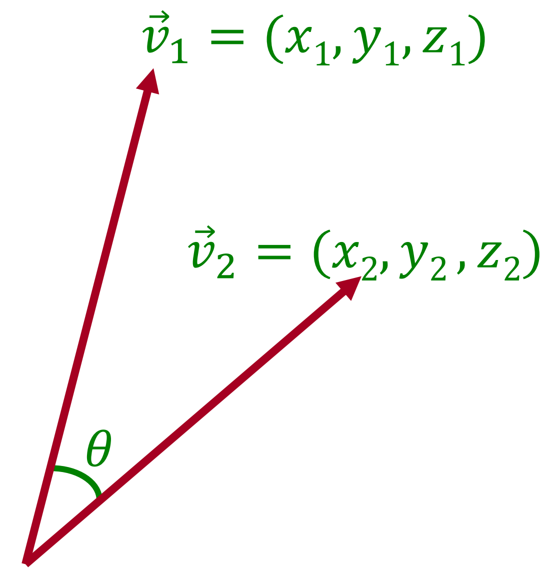
- Specifies a direction (and magnitude)
  - Represented by three coordinates
  - Magnitude $||\vec{v}|| = \sqrt{x^2 + y^2 + z^2}$
  - Has no location

```
template< unsigned int Dim >
struct Direction
{
    float d[Dim];
};
```

$\vec{v} = (x, y, z)$

3D Vector

- Specifies a direction (and magnitude)
  - Represented by three coordinates
  - Magnitude $\|\vec{v}\| = \sqrt{x^2 + y^2 + z^2}$
  - Has no location
- Dot product of two vectors
  - $\langle \vec{v}_1, \vec{v}_2 \rangle = x_1 \cdot x_2 + y_1 \cdot y_2 + z_1 \cdot z_2$
  - $\langle \vec{v}_1, \vec{v}_2 \rangle = \|\vec{v}_1\| \cdot \|\vec{v}_2\| \cdot \cos\theta$
- Cross product of two 3D vectors
  - $\vec{v}_1 \times \vec{v}_2$ = Vector normal to v_1 and v_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
$$\vec{v}_i = (x_i, y_i, z_i) \quad \text{with } i \in \{1,2,3\}$$
Then $\vec{v}_1 = \vec{v}_2 \times \vec{v}_3$ is expressed as:
  - $x_1 = y_2 \cdot z_3 - z_2 \cdot y_3$
  - $y_1 = z_2 \cdot x_3 - x_2 \cdot z_3$
  - $z_1 = x_2 \cdot y_3 - y_2 \cdot x_3$
- 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

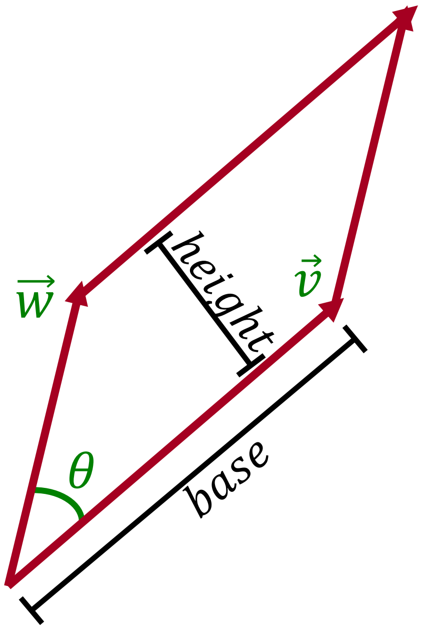
$$\|\vec{v} \times \vec{w}\| = \|\vec{v}\| \cdot \|\vec{w}\| \cdot \sin\theta$$

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.

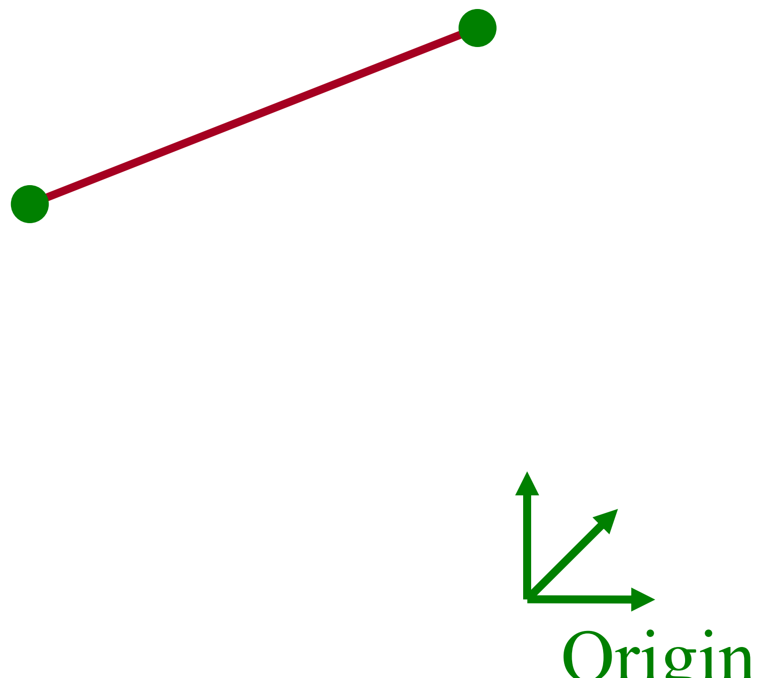
- $base = \|\vec{v}\|$
- $height = \sin(\theta) \cdot \|\vec{w}\|$

$$\Rightarrow \text{Area}(\vec{v}, \vec{w}) = \|\vec{v}\| \cdot \|\vec{w}\| \cdot \sin\theta$$
$$= \|\vec{v} \times \vec{w}\|$$



3D Line Segment

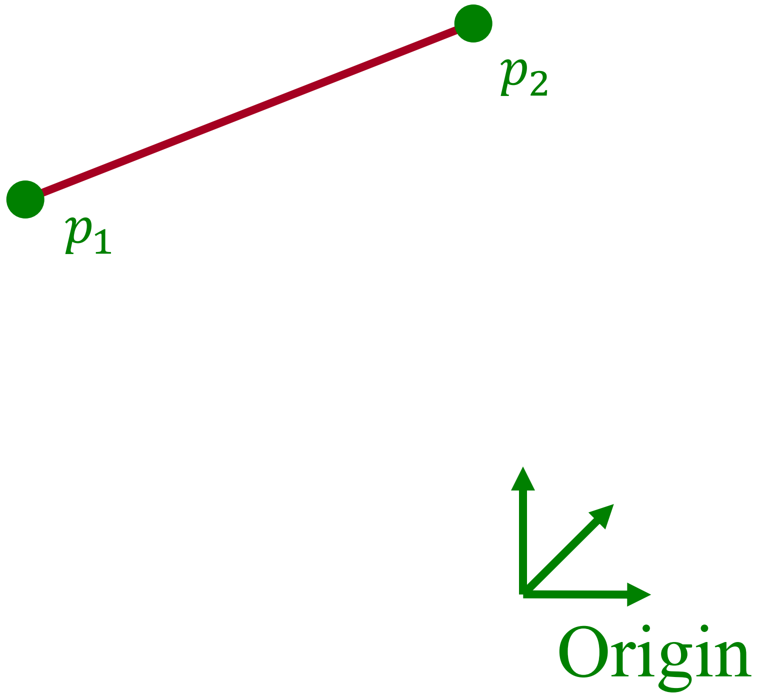
- Linear path between two points



3D Line Segment

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

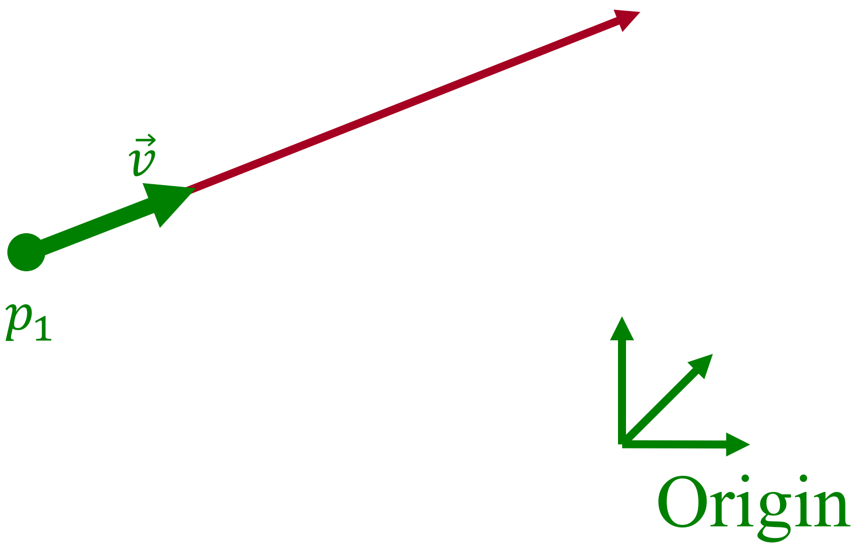
```
template< unsigned int Dim >
struct Segment
{
    Position< Dim > p1 , p2;
};
```



3D Ray

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

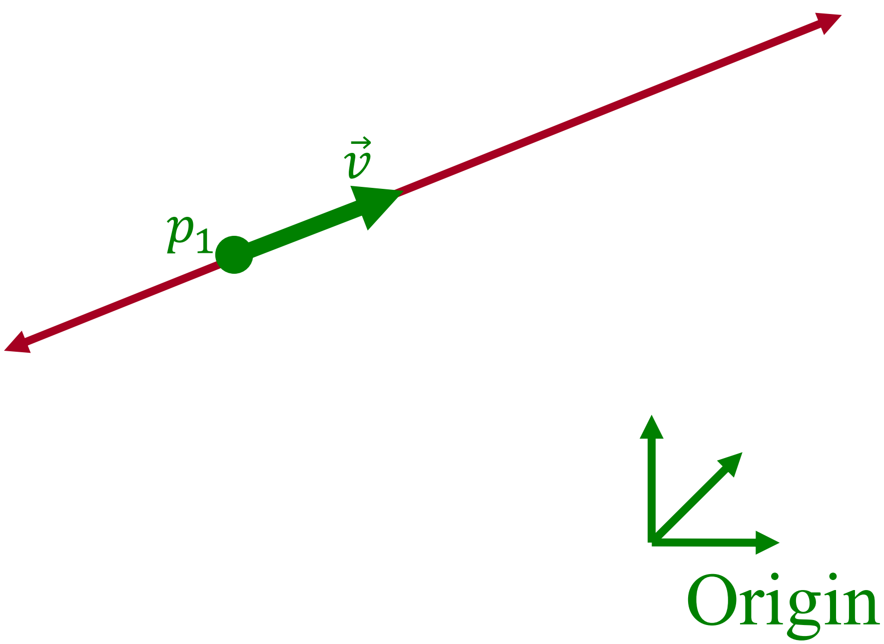
```
template< unsigned int Dim >
struct Ray
{
    Position< Dim > p1;
    Direction< Dim > v;
};
```



3D Line

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

```
template< unsigned int Dim >
struct Line
{
    Position< Dim > p1;
    Direction< Dim > v;
};
```



Geometry in 3D

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

2D geometry in 3D can also be represented by an **implicit function** – a function $\Phi: \mathbb{R}^3 \to \mathbb{R}$ which:

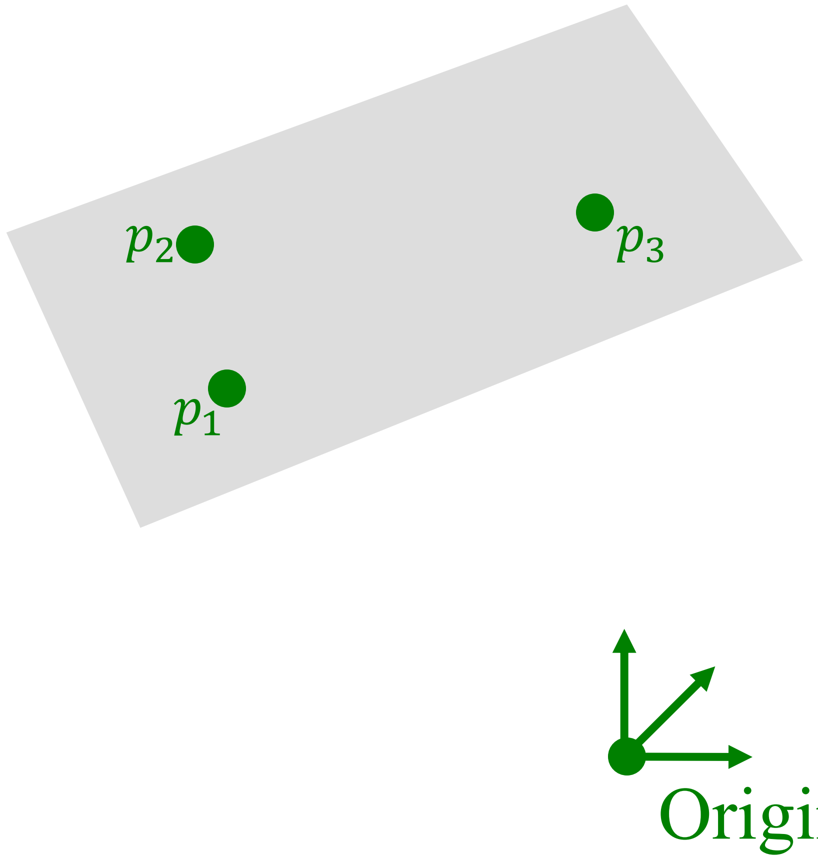
- Equals zero on the geometry
- Is positive outside the geometry
- Is negative inside the geometry

We can also represent 1D geometry using a function $\Phi: \mathbb{R}^3 \to \mathbb{R}^2$, with both coordinates of the output equal to zero on the geometry.

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

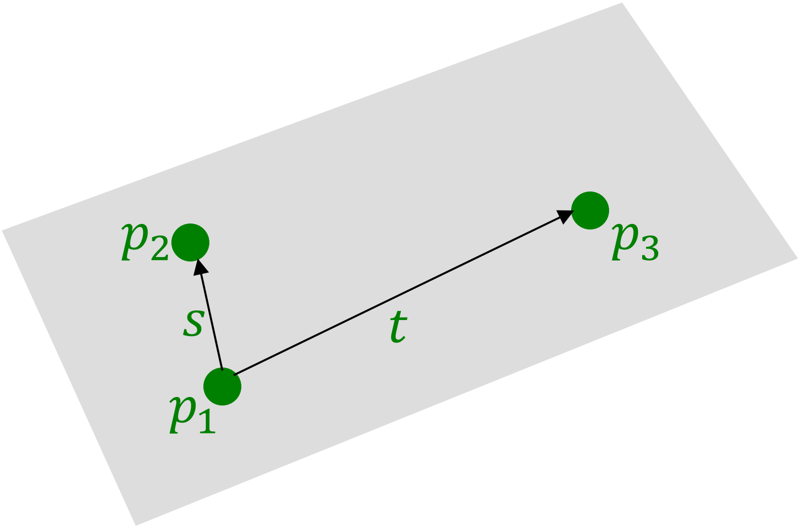
3D Plane

- A linear combination of three points



3D Plane

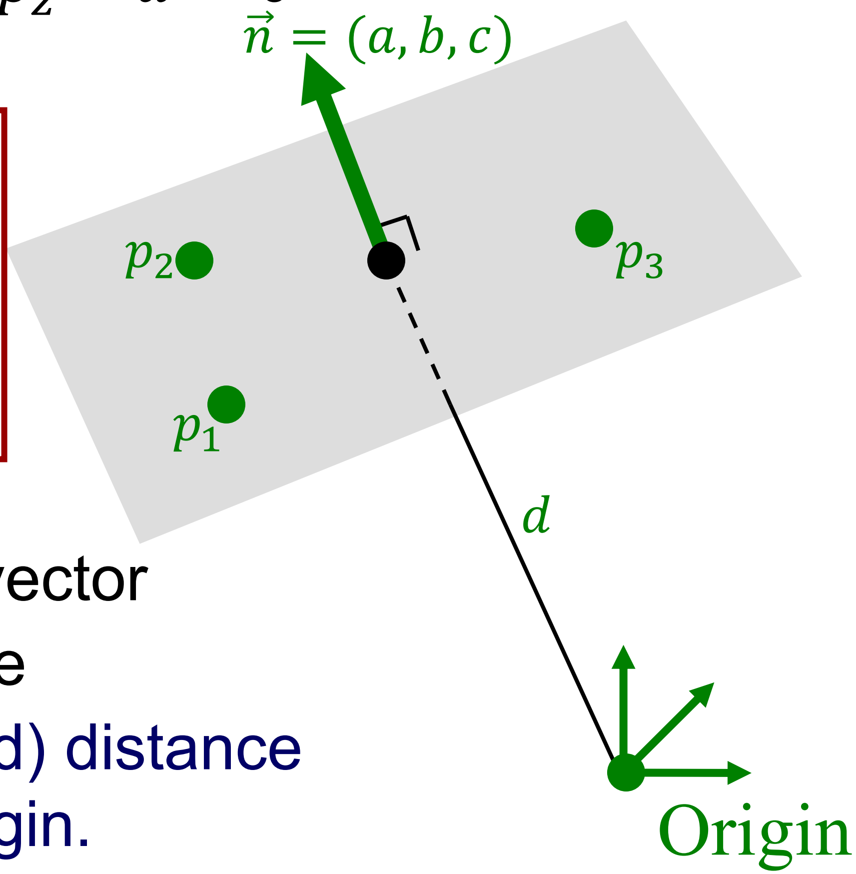
- A linear combination of three points
  - Explicit representation:
    - » $p(s,t) = p_1 + s \cdot (p_2 - p_1) + t \cdot (p_3 - p_1)$



3D Plane

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

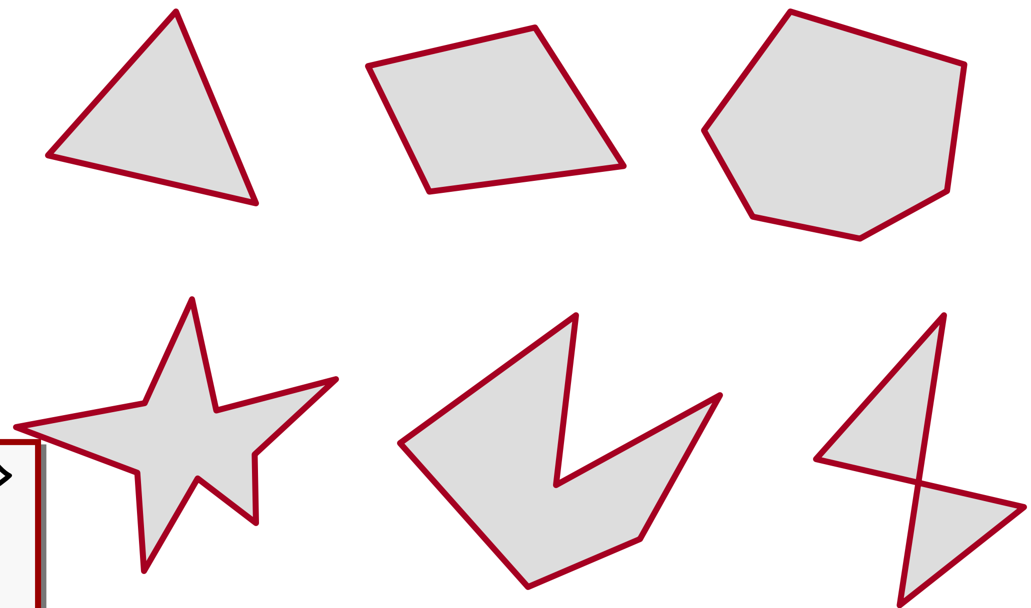
```
Template< unsigned int Dim >
struct Plane
{
    Direction< Dim > n;
    float d;
};
```



  - $\vec{n}$ 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

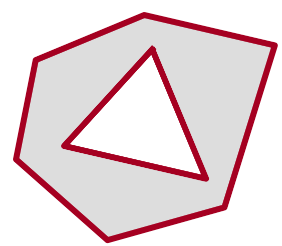
- Area “inside” a sequence of coplanar points
  - Triangle
  - Quadrilateral
  - Convex
  - Star-shaped
  - Concave
  - Self-intersecting



```
Template< unsigned int Dim >
struct Polygon
{
    Position< Dim > *points;
    size_t size;
};
```

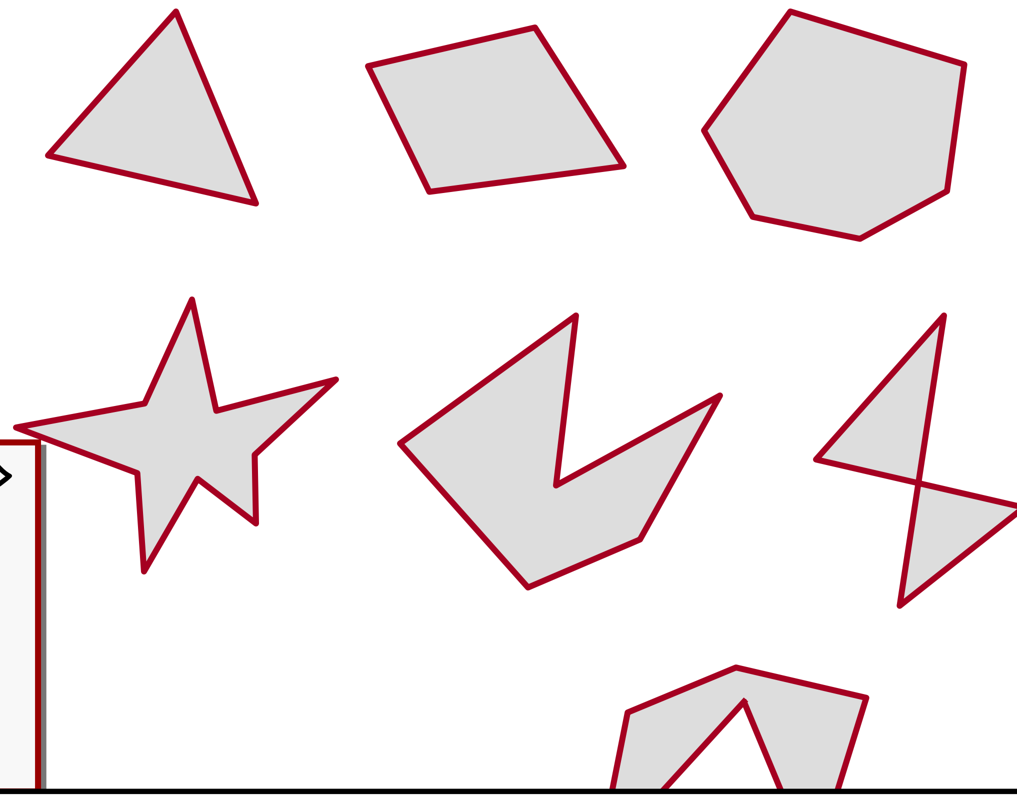
Points are in counter-clockwise order

  - Holes (use > 1 polygon struct)



3D Polygon

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

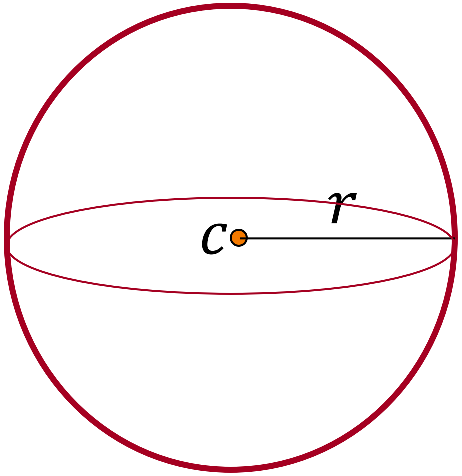


```
Template< unsigned int Dim >
struct Polygon
{
    Position< Dim > *points;
    size_t size;
};
```

[WARNING] If the polygon has more than three points, the points may not be coplanar, so “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:
    - » $x(\phi, \theta) = r \cdot \cos\phi \cdot \sin\theta + c_x$
    - » $y(\phi, \theta) = r \cdot \cos\phi \cdot \sin\theta + c_y$
    - » $z(\theta, \phi) = r \cdot \sin\phi + c_z$




```
template< unsigned int Dim >
struct Sphere
{
    Position< Dim > center;
    float radius;
};
```

Other 3D primitives

- Cone
- Cylinder
- Ellipsoid
- Box
- Etc.

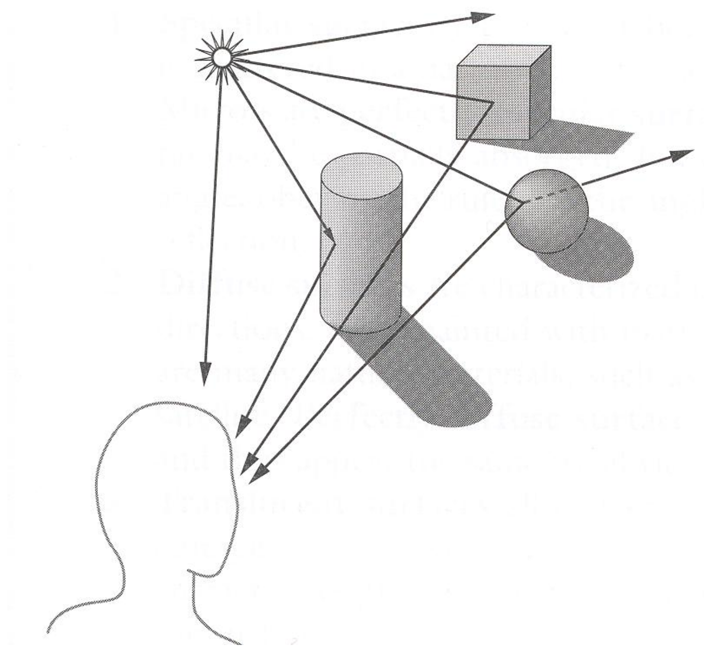
3D Geometric Primitives

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

Overview

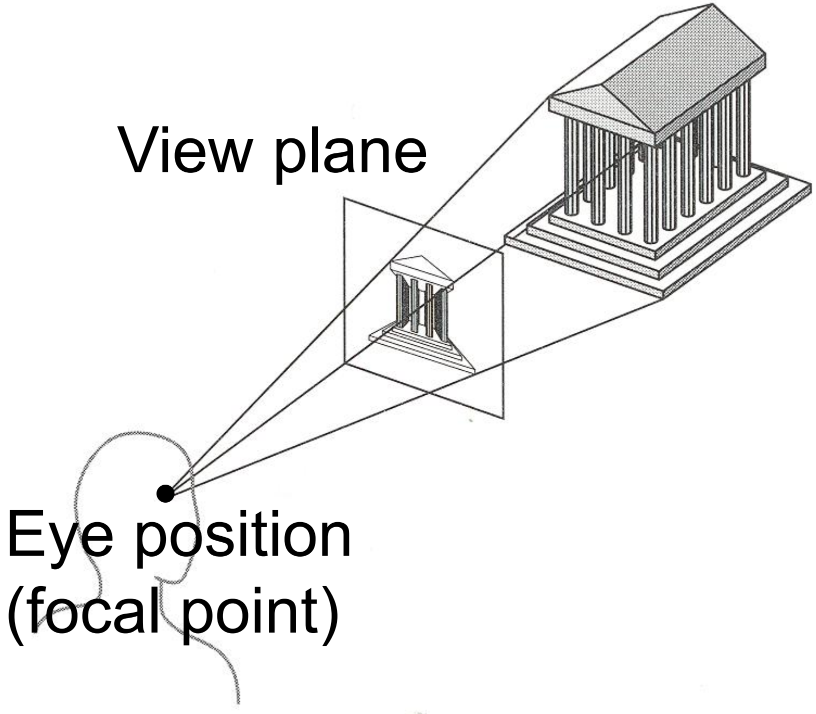
- 3D scene representation
- 3D viewer representation
- What do we see?
- How does it look?

How is the viewing device described in a computer?



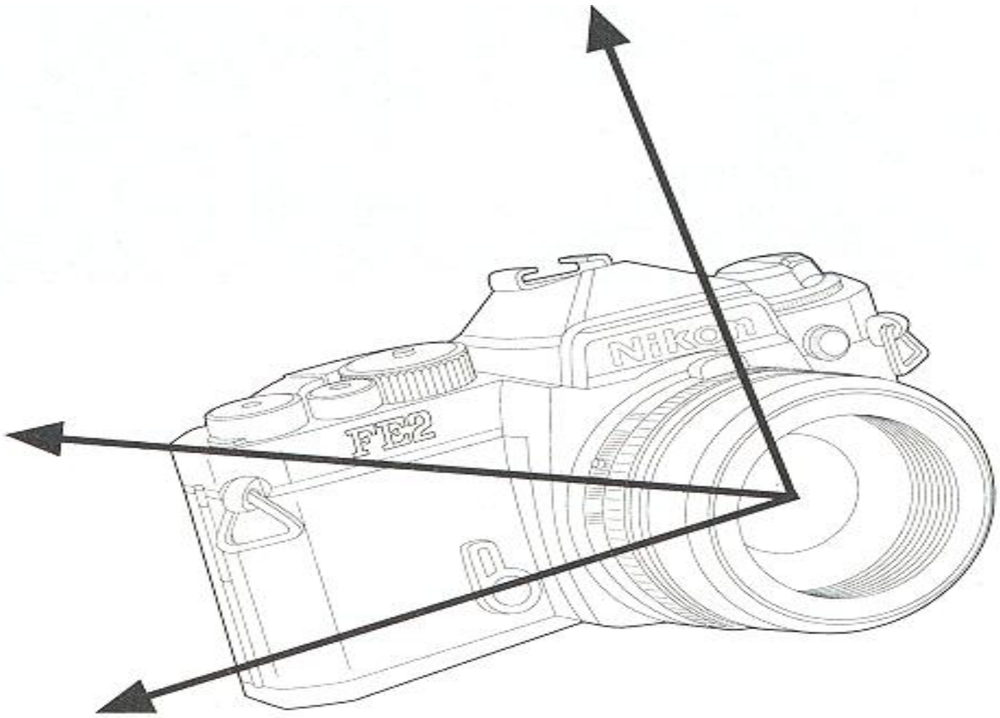
Camera Models

- The most common model is pin-hole camera
  - All captured light rays arrive along paths toward the **focal point** (w/o lens distortion, so that everything is in focus)



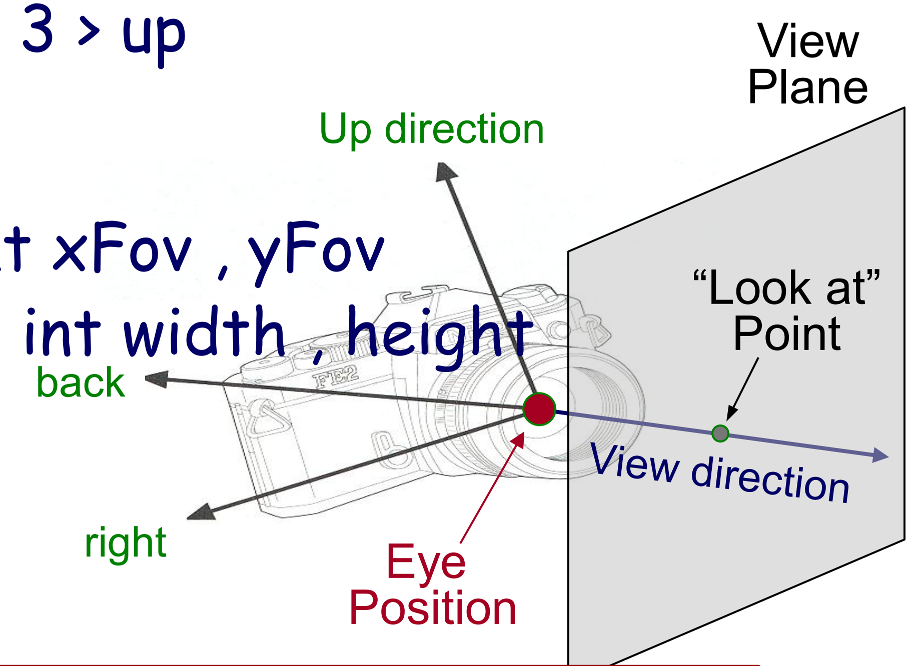
Camera Parameters

- What are the parameters of a camera?



Camera Parameters

- Position
  - Eye position: Position< 3 > eye
- Orientation
  - Forward/view direction: Direction< 3 > view
  - Up direction: Direction< 3 > up
- Aperture
  - Field of view angle: float xFov , yFov
  - Resolution of film plane: int width , height



In some domains, a "look at" point is prescribed instead of a view direction.

Other Models: Depth of Field

Close Focused

Distance Focused

P. Haeberli

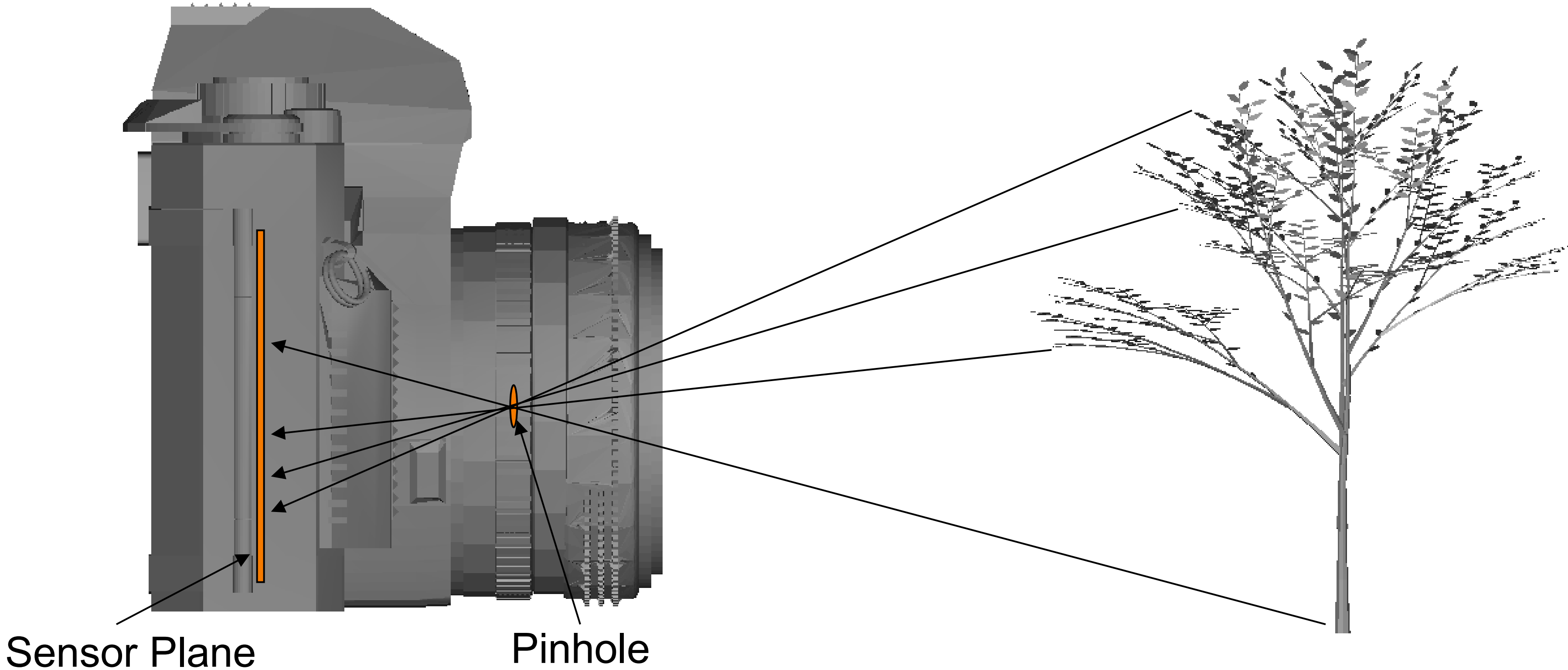
Other Models: Motion Blur

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

Brostow & Essa

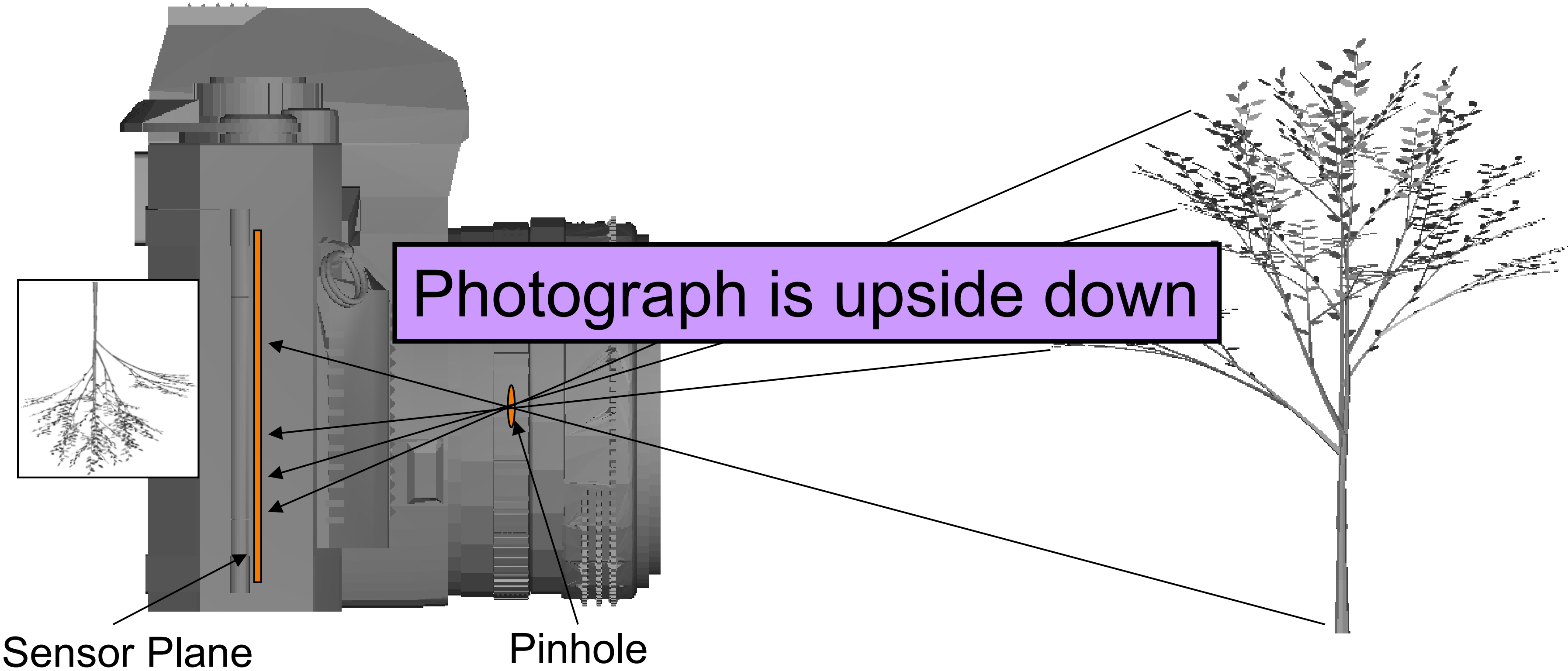
Traditional Pinhole Camera

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



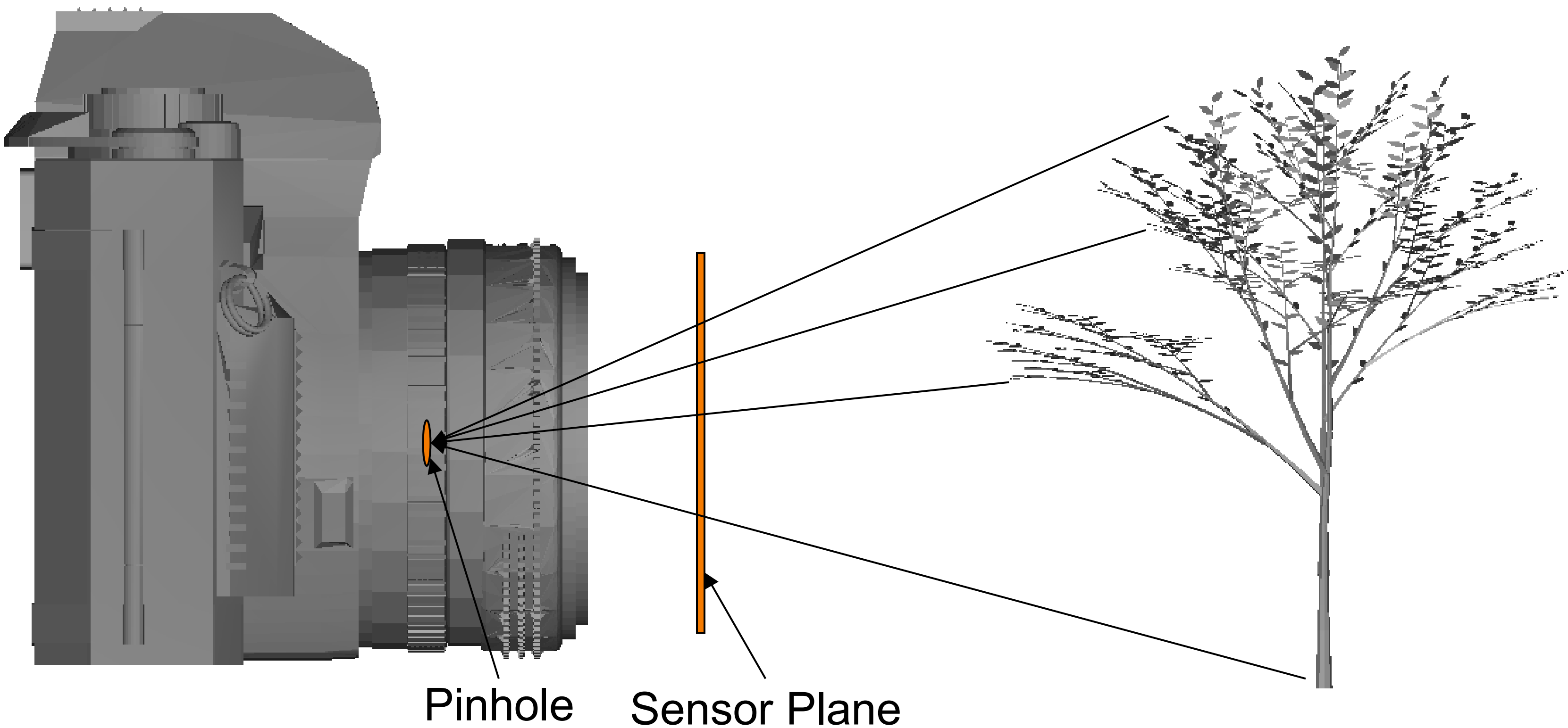
Traditional Pinhole Camera

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



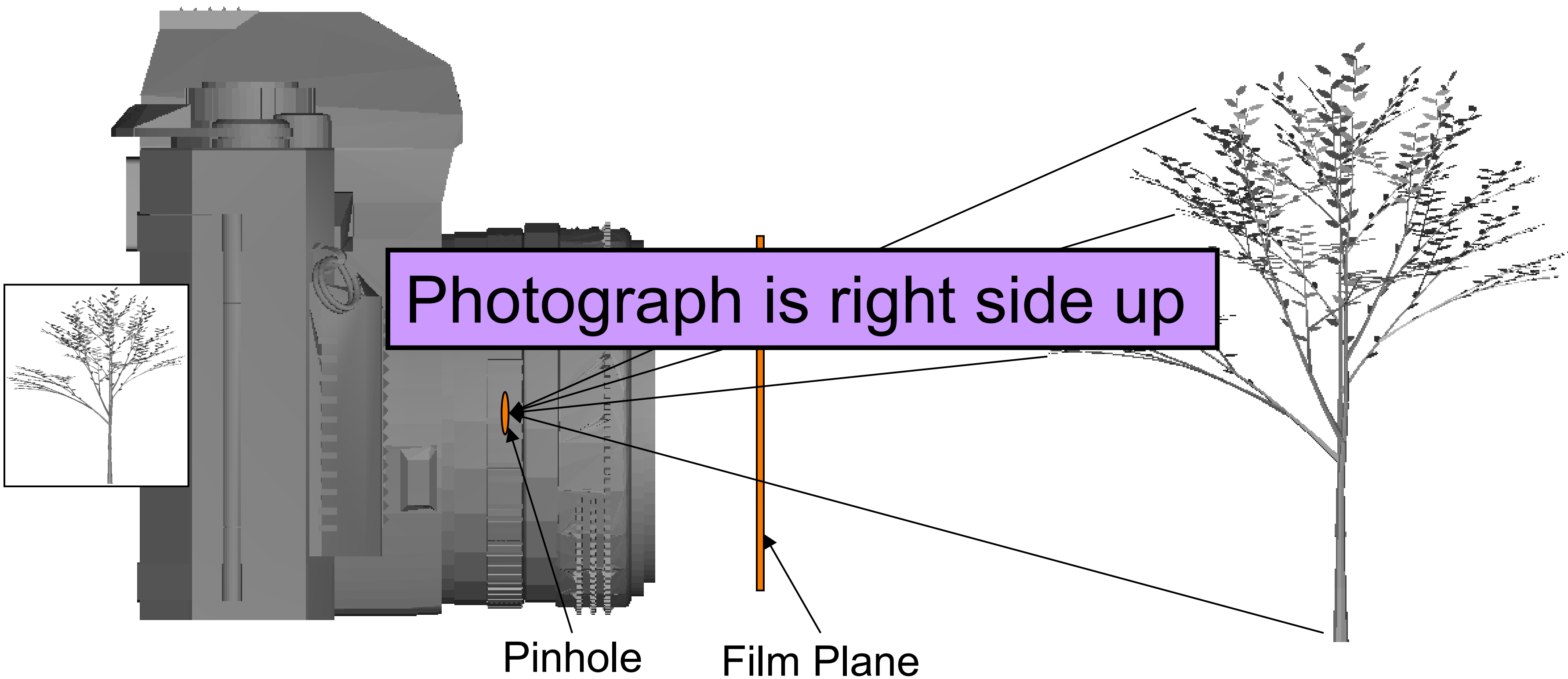
Virtual Camera

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



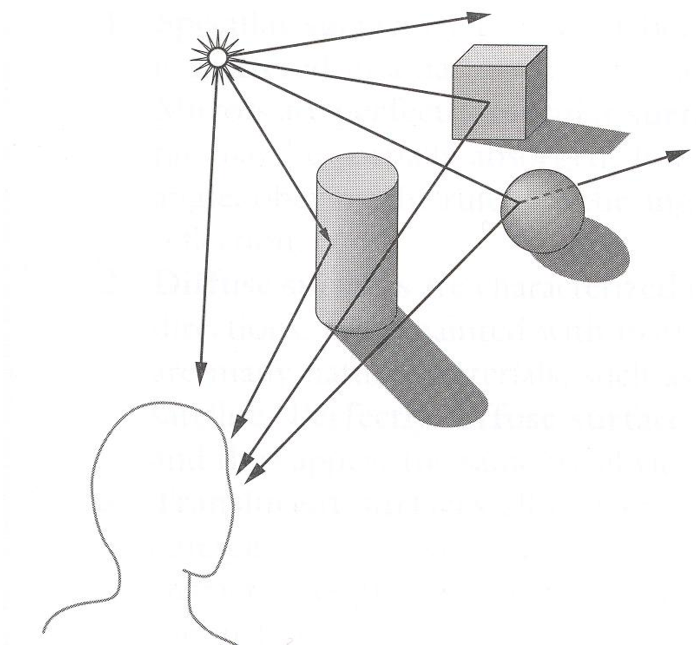
Virtual Camera

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



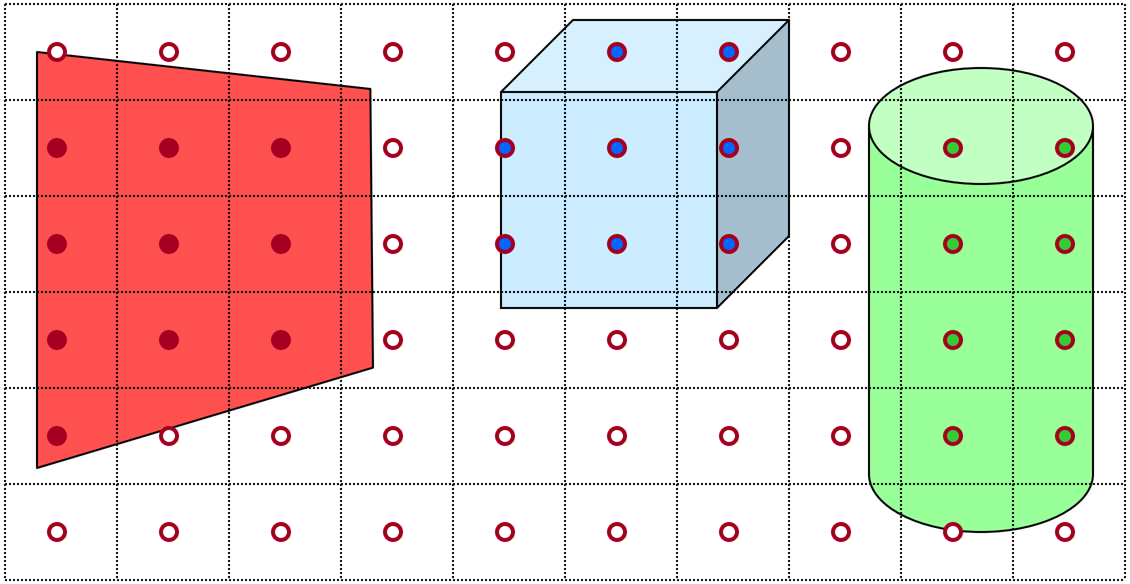
Overview

- 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

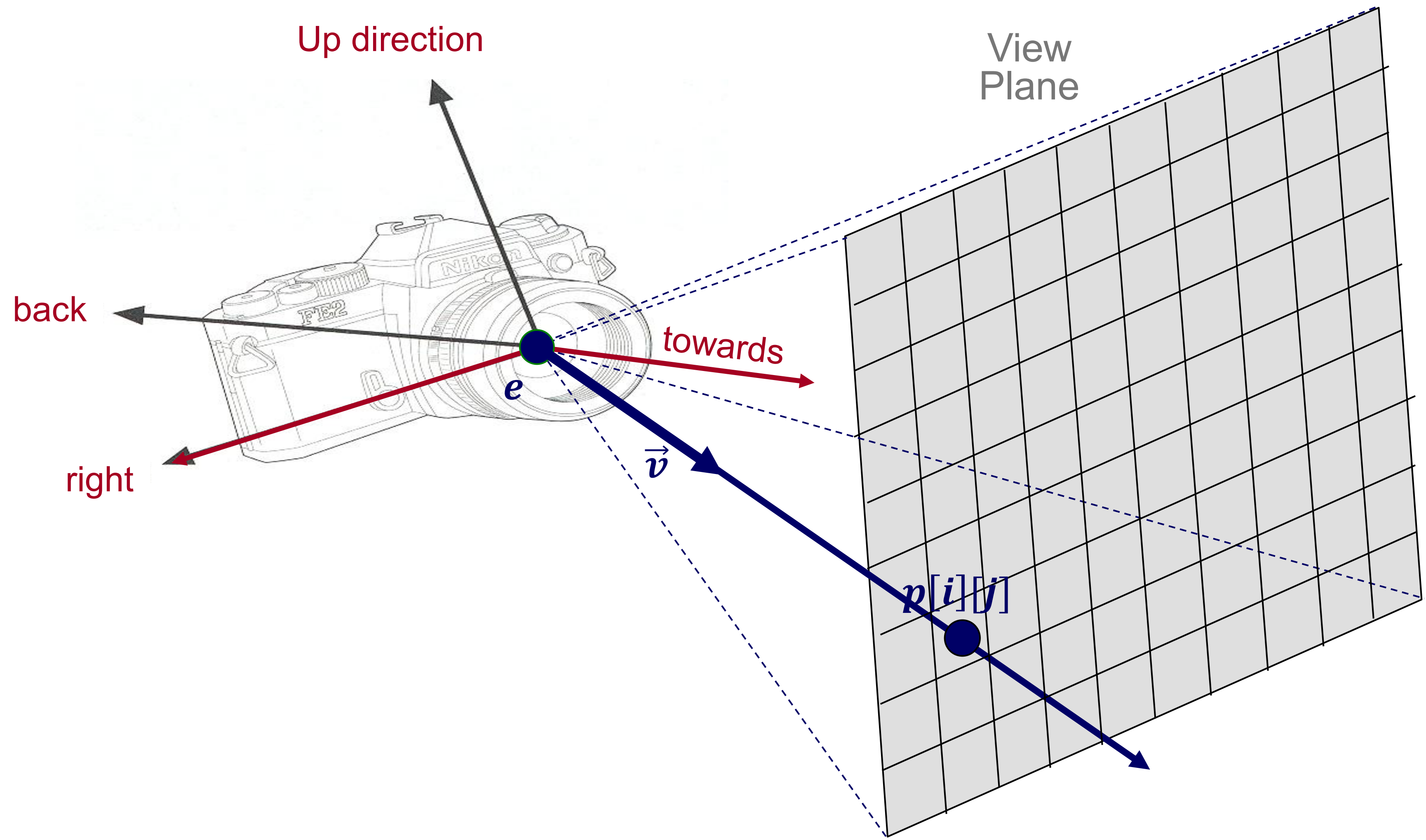
```
Image RayCast( Camera camera , Scene scene , int width , int height)
{
    Image image( width , height );
    for( int j=0 ; j<height ; j++ ) for( int i=0 ; i<width ; i++ )
    {
        Ray< 3 > ray = ConstructRayThroughPixel( camera , i , j );
        Intersection hit = FindIntersection( ray ,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

Ray Casting

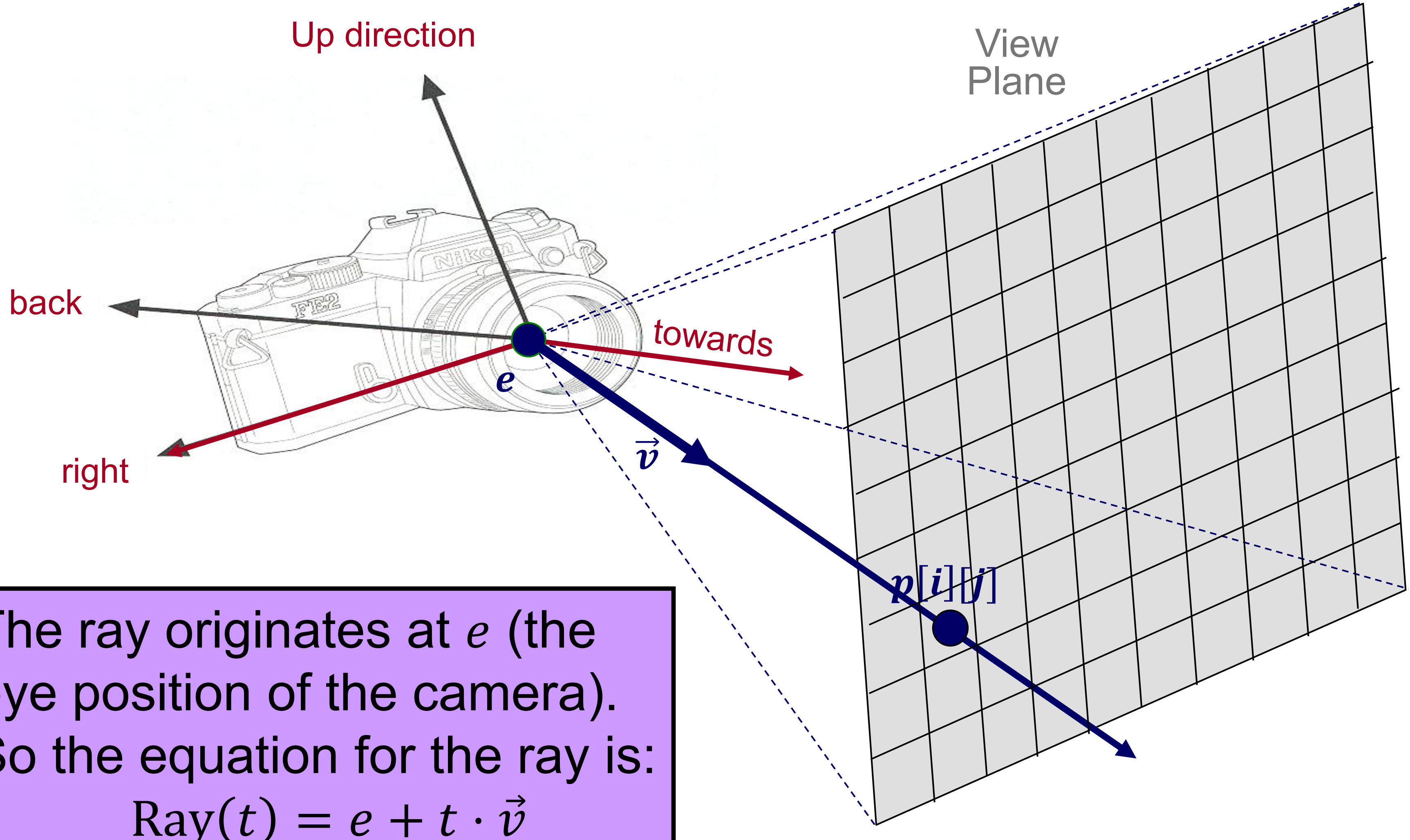
Where?

```
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



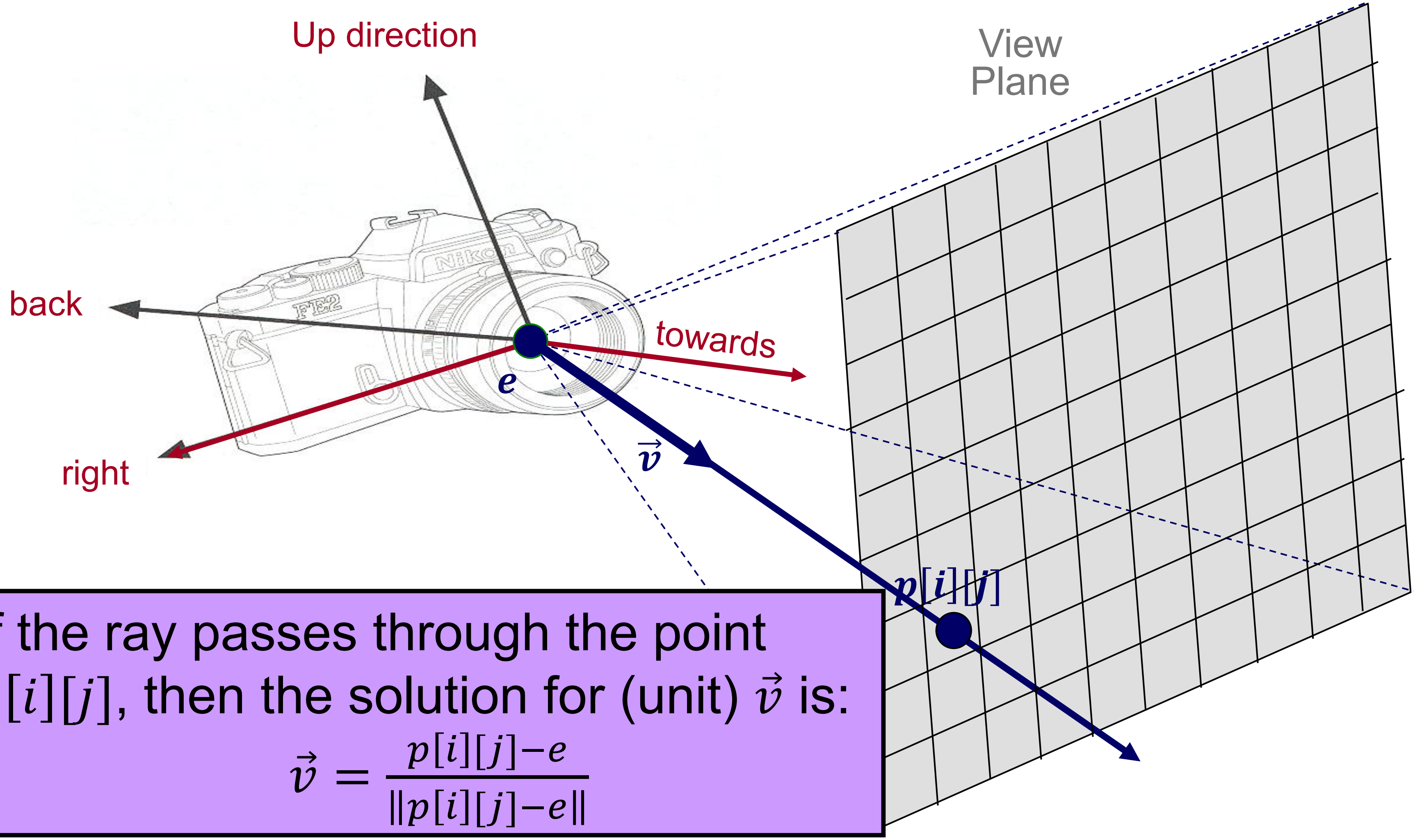
Constructing a Ray Through a Pixel



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

$$\text{Ray}(t) = e + t \cdot \vec{v}$$

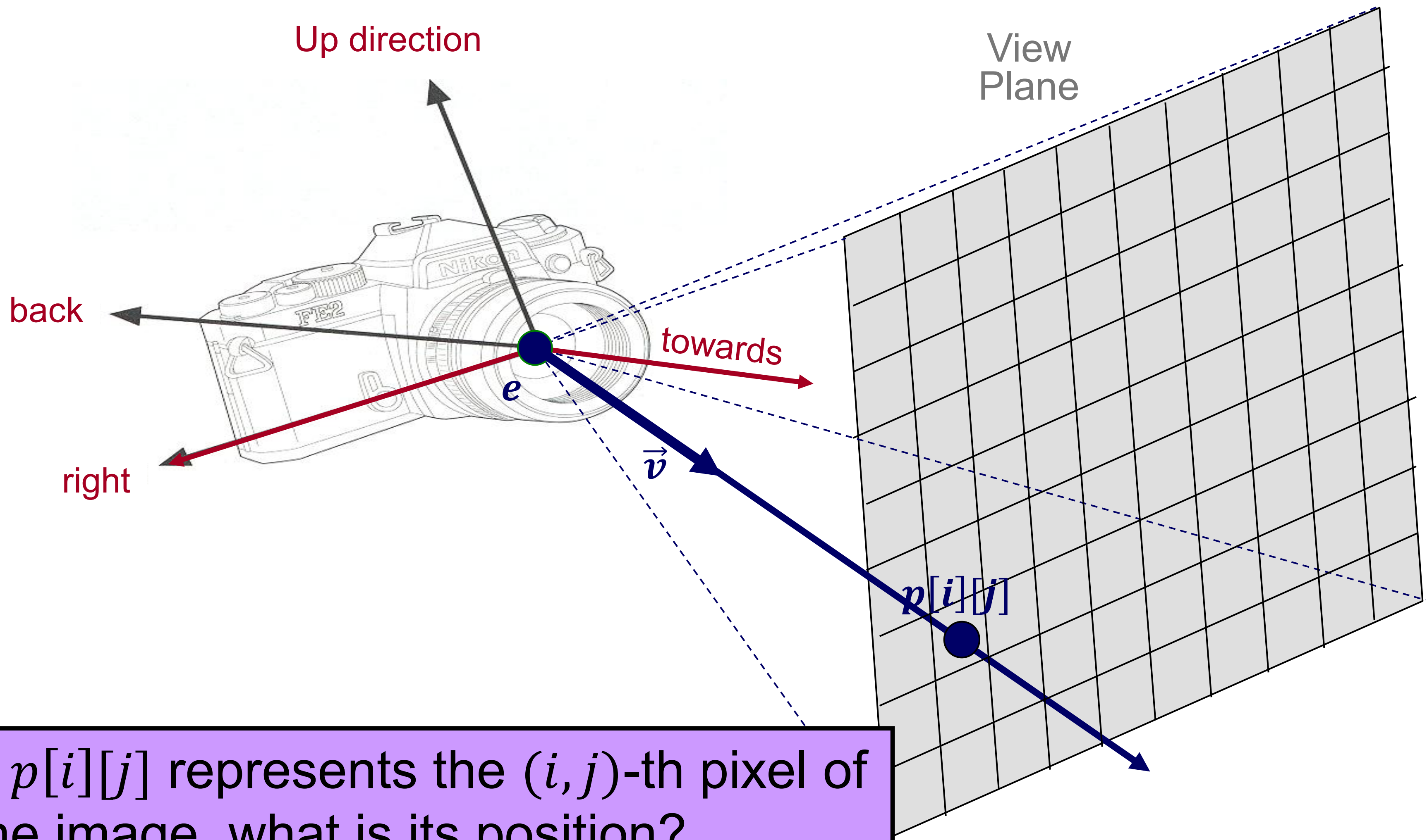
Constructing a Ray Through a Pixel



If the ray passes through the point $p[i][j]$, then the solution for (unit) $\vec{v}$ is:

$$\vec{v} = \frac{p[i][j]-e}{\|p[i][j]-e\|}$$

Constructing a Ray Through a Pixel



If $p[i][j]$ represents the (i,j)-th pixel of the image, what is its position?

Constructing Ray Through a Pixel

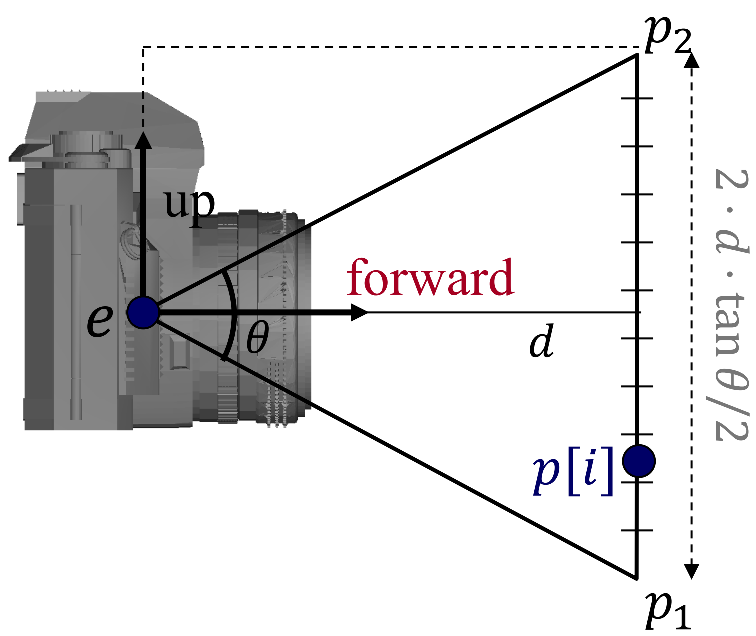
- 2D Example: Side view of camera
  - Where is the i-th pixel, $p[i]$, with $i \in [0, \text{height})$?

θ = field of view angle (given)
d = distance to view plane (arbitrary)

$$p_1 = e + d \cdot \text{forward} - d \cdot \tan\frac{\theta}{2} \cdot \text{up}$$

$$p_2 = e + d \cdot \text{forward} + d \cdot \tan\frac{\theta}{2} \cdot \text{up}$$

$$p[i] = p_1 + \left(\frac{i \boxed{+\ 0.5}}{\text{height}}\right) \cdot (p_2 - p_1)$$



[NOTE]
The offset by 0.5 places the pixel in the center of the cell.

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.

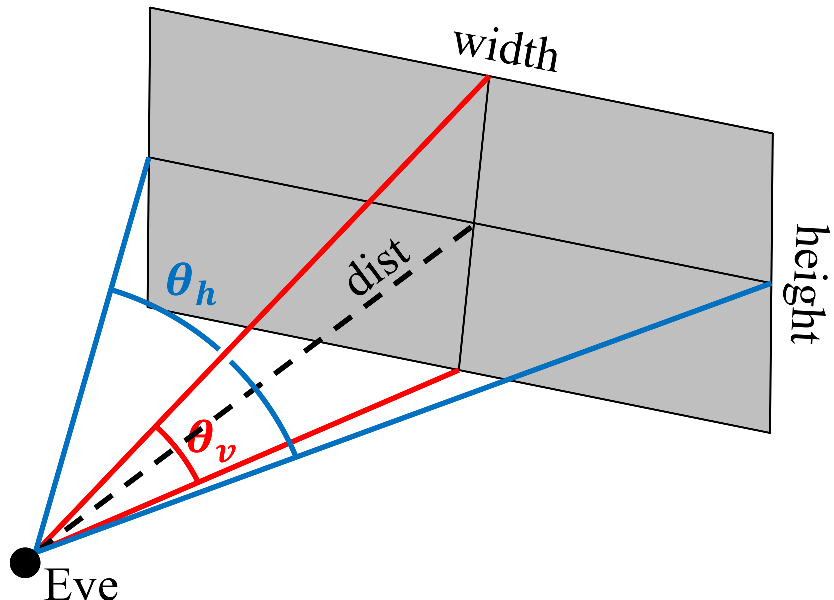
Relating field of view angles and aspect ratio, (assuming square pixels):

- Aspect ratio, $ar = \frac{height}{width}$
- Distance to center, $dist$
- Vertical/horizontal field of view angles, θ_v and θ_h

⇒ Tangents of half the field of view angles are:

$$\tan(\theta_v/2) = \frac{height/2}{dist}$$

$$\tan(\theta_h/2) = \frac{width/2}{dist}$$



Constructing Ray Through a Pixel

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

Relating field of view angles and aspect ratio, (assuming square pixels):

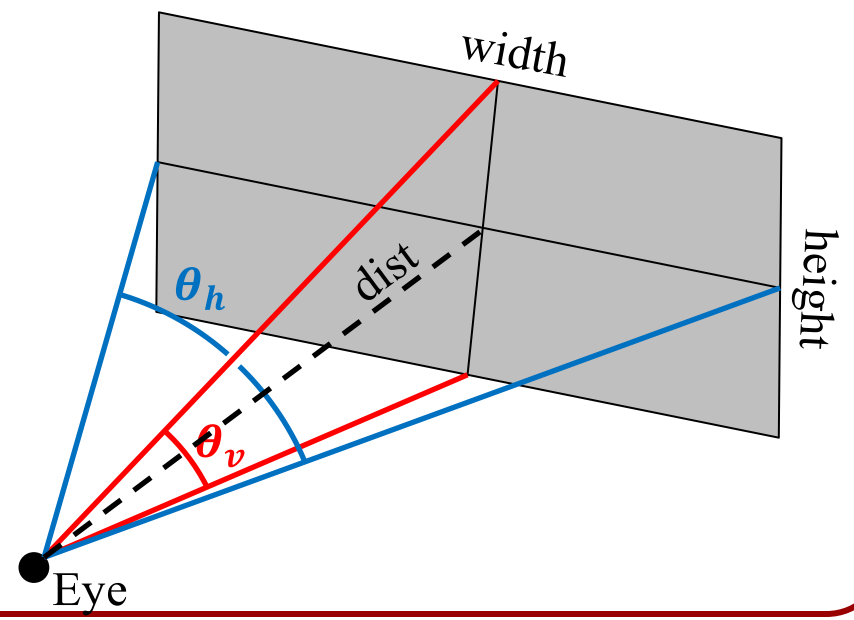
- Aspect ratio, $ar = \frac{height}{width}$
- Distance to center, $dist$
- Vertical/horizontal field of view angles, θ_v and θ_h

$\Rightarrow$ Tangents of half the field of view angles are:

$$\tan(\theta_v/2) = \frac{height/2}{dist}$$

$$\tan(\theta_h/2) = \frac{width/2}{dist}$$

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



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: $p(t) = e + t \cdot \vec{v}, \quad (0 \le t < \infty)$
Sphere: $\Phi(p) = \|p - c\|^2 - r^2 = 0$

Substituting for p, we get:

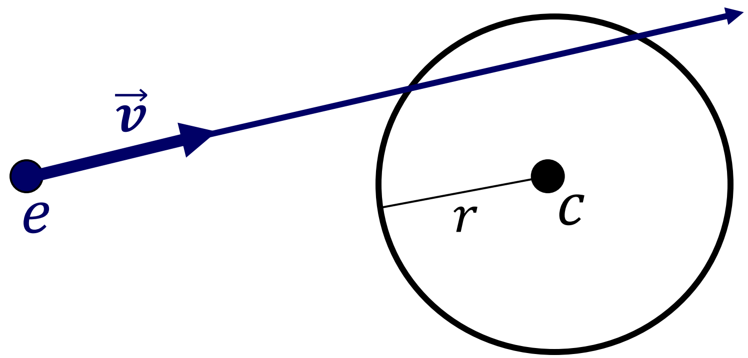
$$\Phi(t) \equiv \Phi\big(p(t)\big) = \|e + t \cdot \vec{v} - c\|^2 - r^2 = 0$$

Solve quadratic equation:

$$a \cdot t^2 + b \cdot t + c = 0$$

where:

$$a = 1$$
$$b = 2\langle \vec{v}, e - c \rangle$$
$$c = \|e - c\|^2 - r^2$$



Ray-Sphere Intersection

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

Substituting for p, we get:

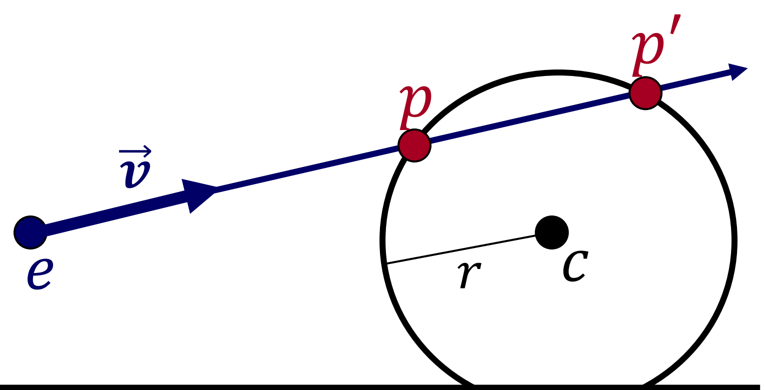
$$\Phi(t) \equiv \Phi\big(p(t)\big) = \|e + t \cdot \vec{v} - c\|^2 - r^2 = 0$$

Solve quadratic equation:

$$a \cdot t^2 + b \cdot t + c = 0$$

where:

$$a = 1$$
$$b = 2\langle \vec{v}, e - c \rangle$$

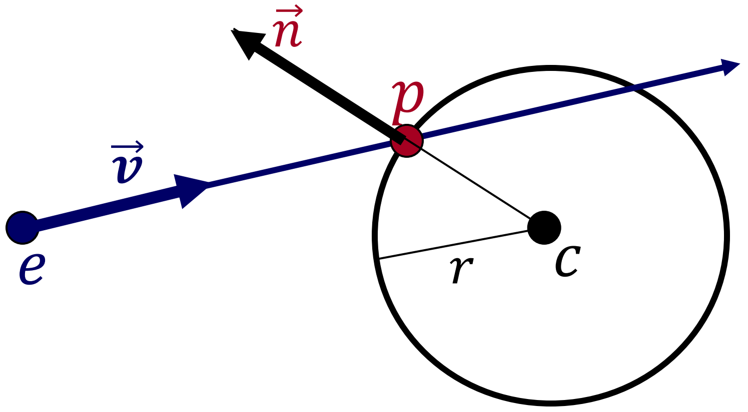


There can be two solutions to the quadratic equation, giving two points of intersection, p and p'. Want to return the first <u>positive</u> hit.

Ray-Sphere Intersection

- Need normal vector at intersection for lighting calculations:

$$\vec{n} = \frac{p - c}{\|p - c\|}$$



Ray-Sphere Intersection

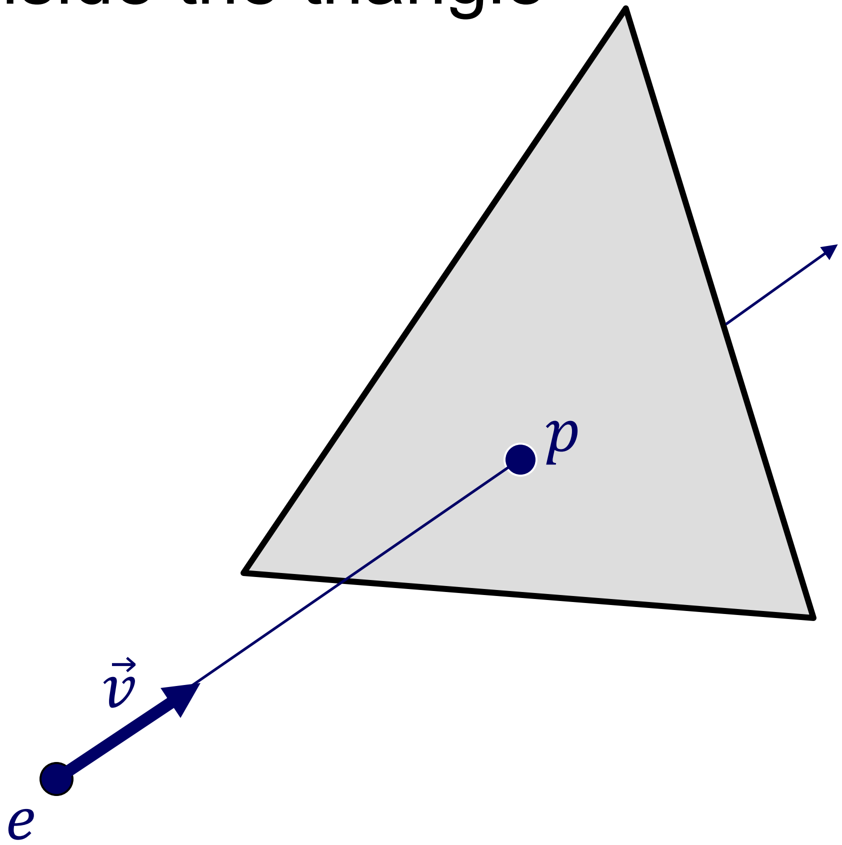
- More generally, if the shape is given as the set of points p satisfying:
$$\Phi(p) = 0$$
for some function $\Phi: \mathbb{R}^3 \to \mathbb{R}$, then the normal of the surface will be parallel to the gradient.

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

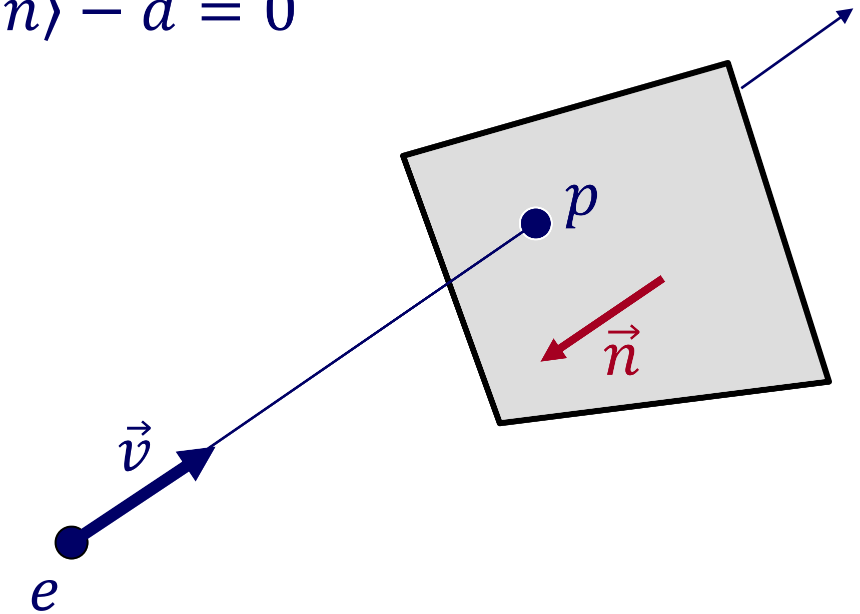
Ray: $p(t) = e + t \cdot \vec{v}, \quad (0 \le t < \infty)$
Plane: $\Phi(p) = \langle p, \vec{n} \rangle - d = 0$

Substituting for p we get:

$$\Phi(t) \equiv \Phi\big(p(t)\big) = \langle e + t \cdot \vec{v}, \vec{n} \rangle - d = 0$$

Solving gives:

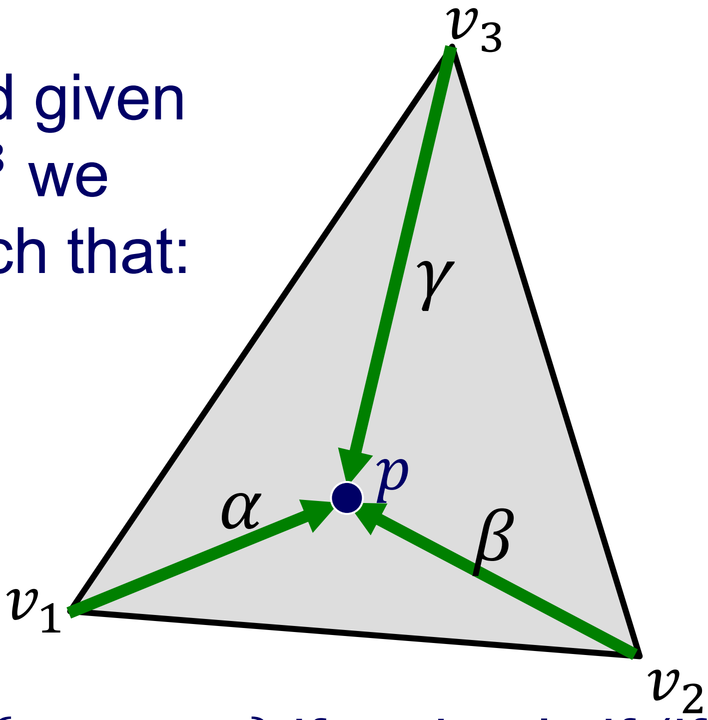
$$t = -\frac{\langle e, \vec{n} \rangle - d}{\langle \vec{v}, \vec{n} \rangle}$$



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

Ray-Triangle Intersection

- Check for point-triangle intersection parametrically
  - In general, given $p \in \mathbb{R}^3$ and given three points $\{v_1, v_2, v_3\} \subset \mathbb{R}^3$ we can solve for $\alpha, \beta, \gamma \in \mathbb{R}$ such that:
    $$p = \alpha v_1 + \beta v_2 + \gamma v_3$$

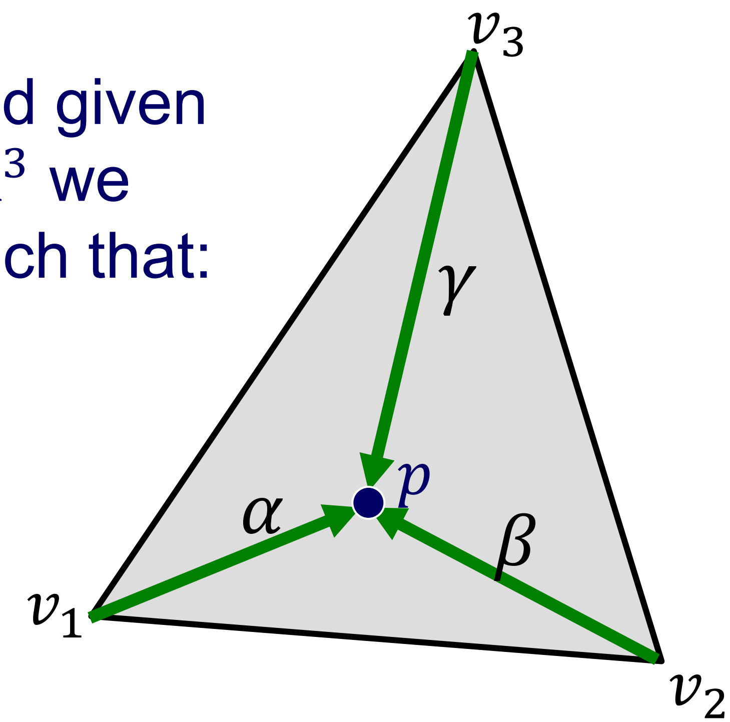


  - p is in the plane containing $\{v_1, v_2, v_3\}$ if and only if (iff.):
    $$\alpha + \beta + \gamma = 1$$
  - p is inside the triangle with vertices $\{v_1, v_2, v_3\}$ iff.:
    $$\alpha, \beta, \gamma \geq 0$$

Ray-Triangle Intersection

- Check for point-triangle intersection parametrically
  - In general, given $p \in \mathbb{R}^3$ and given three points $\{v_1, v_2, v_3\} \subset \mathbb{R}^3$ we can solve for $\alpha, \beta, \gamma \in \mathbb{R}$ such that:

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



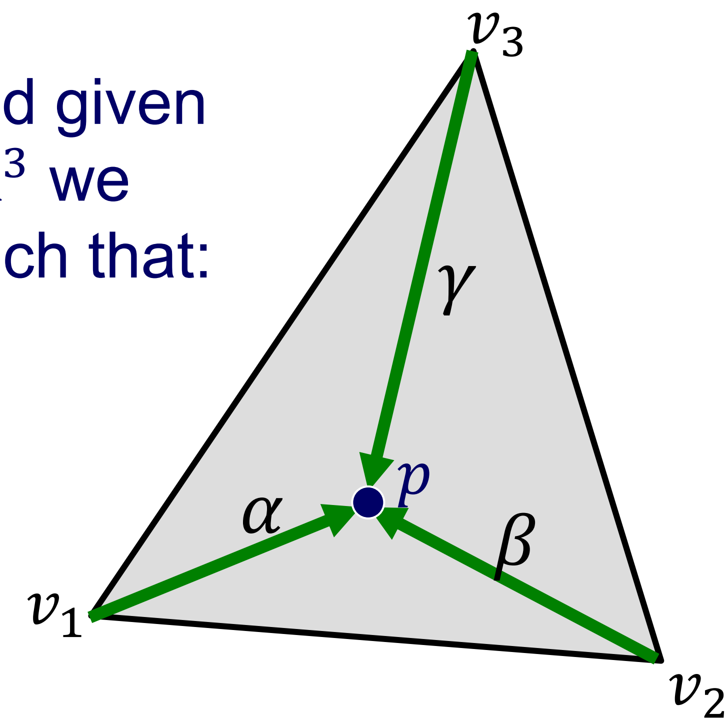
To get α, β, γ, we could try to solve the system:

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

Ray-Triangle Intersection

- Check for point-triangle intersection parametrically
  - In general, given $p \in \mathbb{R}^3$ and given three points $\{v_1, v_2, v_3\} \subset \mathbb{R}^3$ we can solve for $\alpha, \beta, \gamma \in \mathbb{R}$ such that:

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



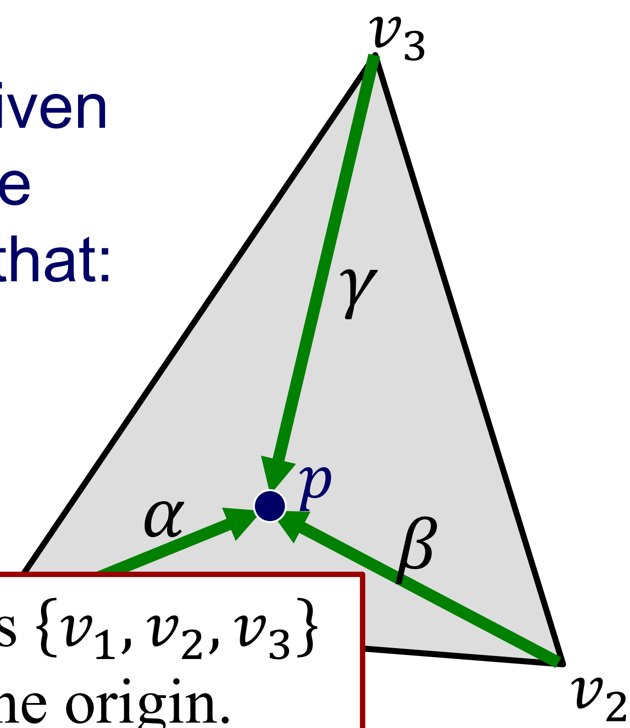
To get α, β, γ, we could try to solve the system:

$$\begin{pmatrix} v_1^x & v_2^x & v_3^x \\ v_1^y & v_2^y & v_3^y \\ v_1^z & v_2^z & v_3^z \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} p^x \\ p^y \\ p^z \end{pmatrix} \quad \Leftrightarrow \quad \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} v_1^x & v_2^x & v_3^x \\ v_1^y & v_2^y & v_3^y \\ v_1^z & v_2^z & v_3^1 \end{pmatrix}^{-1} \begin{pmatrix} p^x \\ p^y \\ p^z \end{pmatrix}$$

Ray-Triangle Intersection

- Check for point-triangle intersection parametrically
  - In general, given $p \in \mathbb{R}^3$ and given three points $\{v_1, v_2, v_3\} \subset \mathbb{R}^3$ we can solve for $\alpha, \beta, \gamma \in \mathbb{R}$ such that:

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



This will fail if the vertices $\{v_1, v_2, v_3\}$ lie in a plane through the origin.

To get α, β, γ, we could try to solve the system:

$$\begin{pmatrix} v_1^x & v_2^x & v_3^x \\ v_1^y & v_2^y & v_3^y \\ v_1^z & v_2^z & v_3^z \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} p^x \\ p^y \\ p^z \end{pmatrix} \quad \Leftrightarrow \quad \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} v_1^x & v_2^x & v_3^x \\ v_1^y & v_2^y & v_3^y \\ v_1^z & v_2^z & v_3^1 \end{pmatrix}^{-1} \begin{pmatrix} p^x \\ p^y \\ p^z \end{pmatrix}$$

Ray-Triangle Intersection

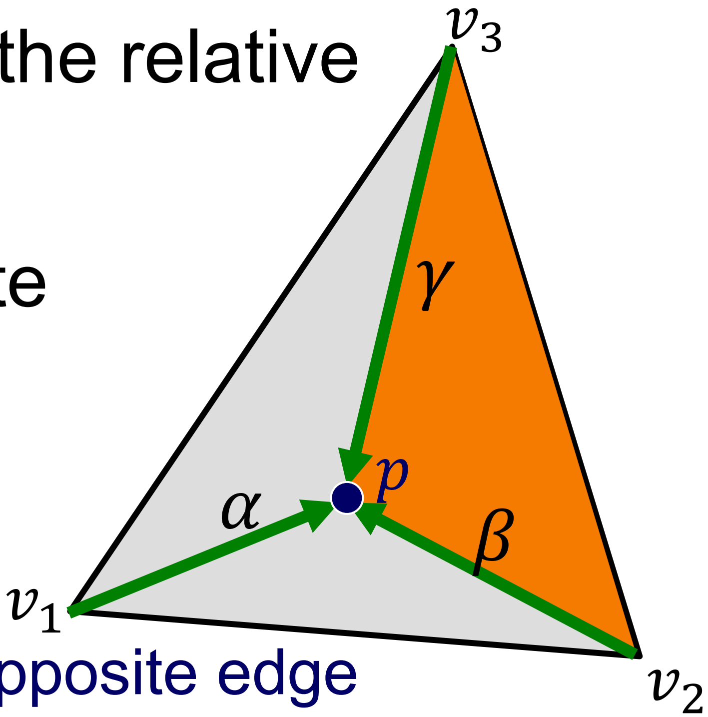
Intuitively:

The weights α, β, γ describe the relative proximity of p to v_1, v_2, v_3.

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 towards the opposite edge
- Tends to the area of triangle $\{v_1, v_2, v_3\}$ as p moves towards v_k.



Ray-Triangle Intersection

<u>Intuitively</u>:

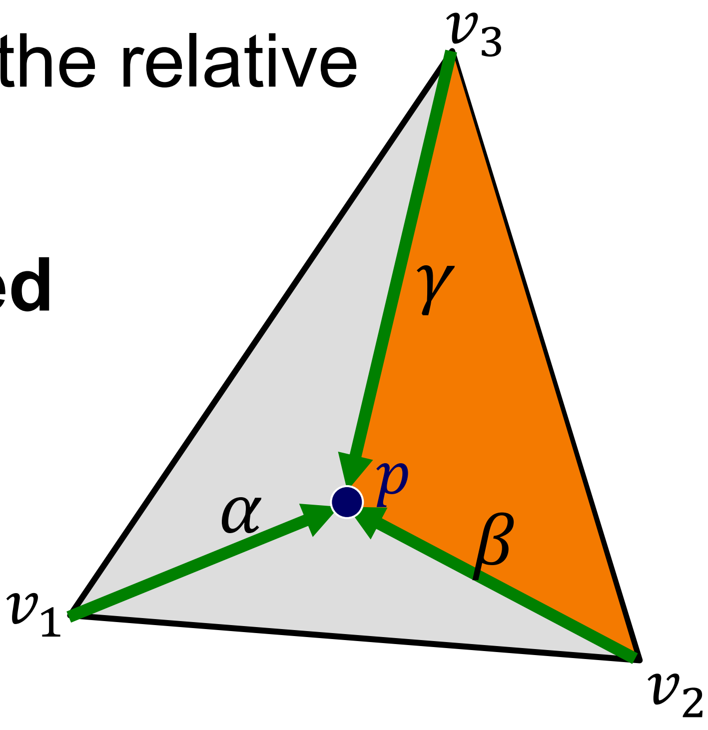
The weights α, β, γ describe the relative proximity of p to v_1, v_2, v_3.

$\Rightarrow$ Define α, β, γ as 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:

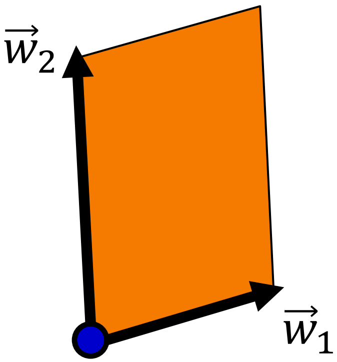
Given vectors $\vec{w}_1, \vec{w}_2 \in \mathbb{R}^3$, the **unsigned** area of the parallelogram spanned by $\vec{w}_1$ and $\vec{w}_2$ is:

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

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

<u>Recall</u>:

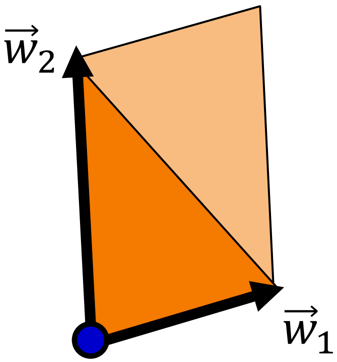
Given vectors $\vec{w}_1, \vec{w}_2 \in \mathbb{R}^3$, the **unsigned** area of the parallelogram spanned by $\vec{w}_1$ and $\vec{w}_2$ is:

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

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:

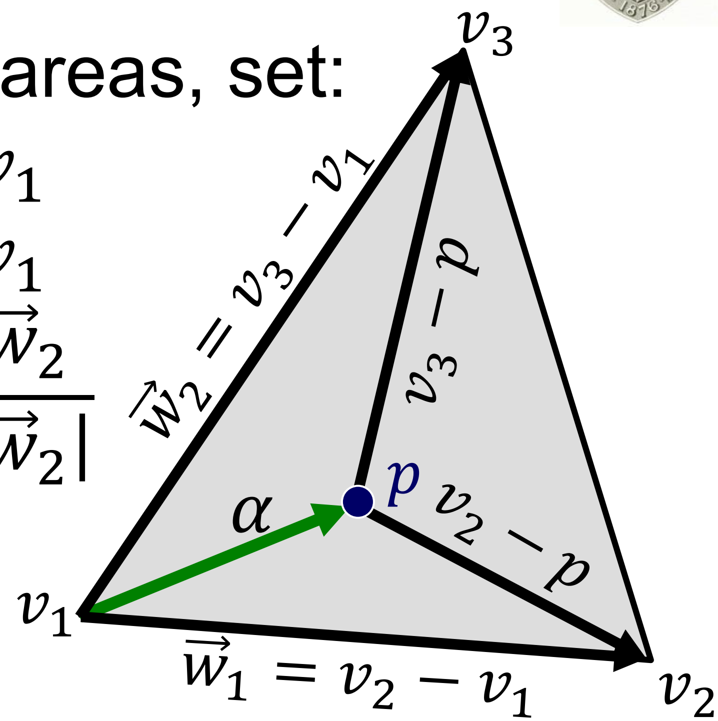


$$\text{SignedParallelogramArea}(\vec{w}_1, \vec{w}_2) = \langle \vec{w}_1 \times \vec{w}_2, \vec{n} \rangle$$
$$\text{SignedTriangleArea}(\vec{w}_1, \vec{w}_2) = \langle \vec{w}_1 \times \vec{w}_2, \vec{n} \rangle / 2$$

Ray-Triangle Intersection

To compute the ratio of signed areas, set:

$$\vec{w}_1 = v_2 - v_1$$
$$\vec{w}_2 = v_3 - v_1$$
$$\vec{n} = \frac{\vec{w}_1 \times \vec{w}_2}{|\vec{w}_1 \times \vec{w}_2|}$$



Then we get:

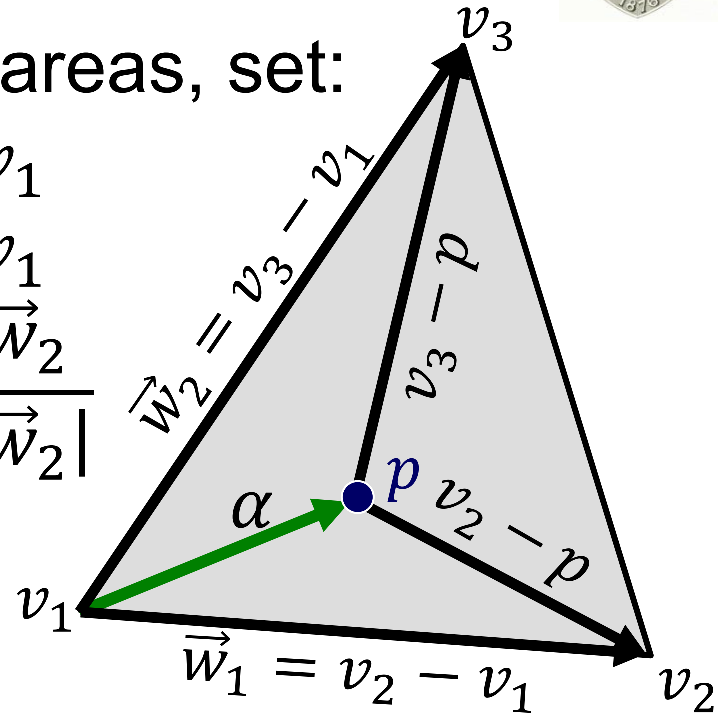
$$\alpha = \frac{\langle (v_2 - p) \times (v_3 - p), \vec{n} \rangle / 2}{\langle (v_2 - v_1) \times (v_3 - v_1), \vec{n} \rangle / 2}$$

$$\vdots$$

Ray-Triangle Intersection

To compute the ratio of signed areas, set:

$$\vec{w}_1 = v_2 - v_1$$
$$\vec{w}_2 = v_3 - v_1$$
$$\vec{n} = \frac{\vec{w}_1 \times \vec{w}_2}{|\vec{w}_1 \times \vec{w}_2|}$$



Then we get:

$$\alpha = \frac{\langle (v_2 - p) \times (v_3 - p), \vec{n} \rangle / 2}{\langle (v_2 - v_1) \times (v_3 - v_1), \vec{n} \rangle / 2}$$

<u>Note</u>: If we flip the sign of $\vec{n}$, 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