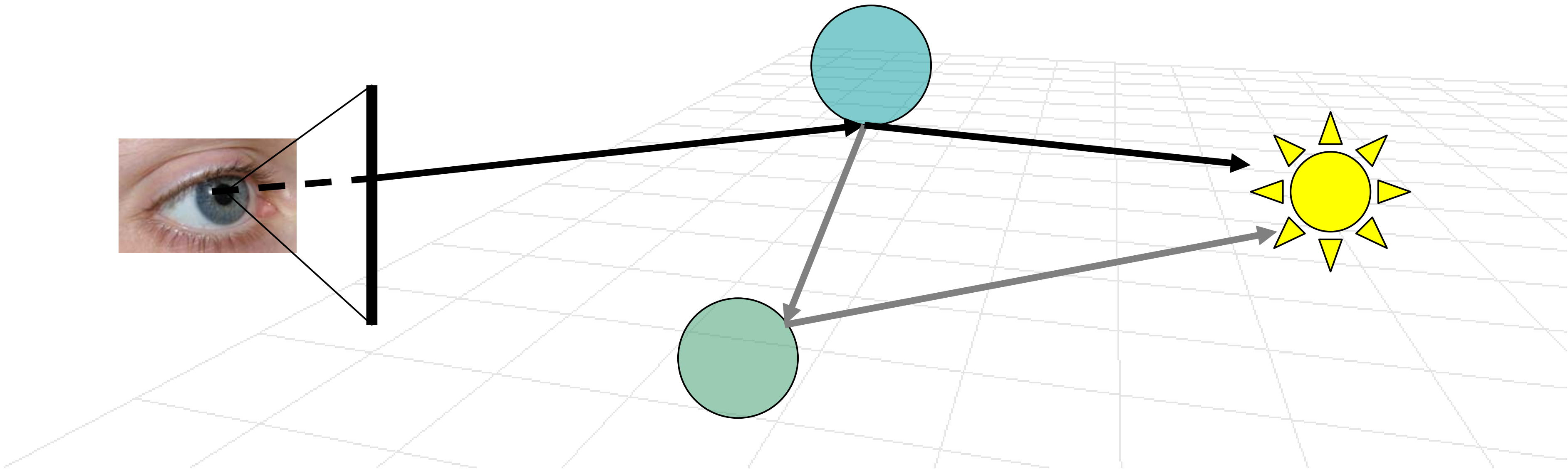
Announcements

- **Assignment 2**
  - Theory (late submissions in effect till today)
  - Programming **due next Friday**
- **Midterms**
  - **Mean: ~74%**
  - If you did poorly on the midterm, don’t worry, you can still get an “A” in the course (50% on the final means you can still get 92.5% in the course)
- **Assignment 1**
  - Programming **back**

Ray Tracing

- For each pixel
  - Form a ray (a.k.a. ray casting)
  - Find intersection of this ray with objects in the scene
  - Find closest object intersection (there could be multiple object intersections for any given ray)
  - Find normal at the closest intersection point (a.k.a hit point)
  - Evaluate reflectance model at the hit point (global + local)



Finding Intersections (for triangle)

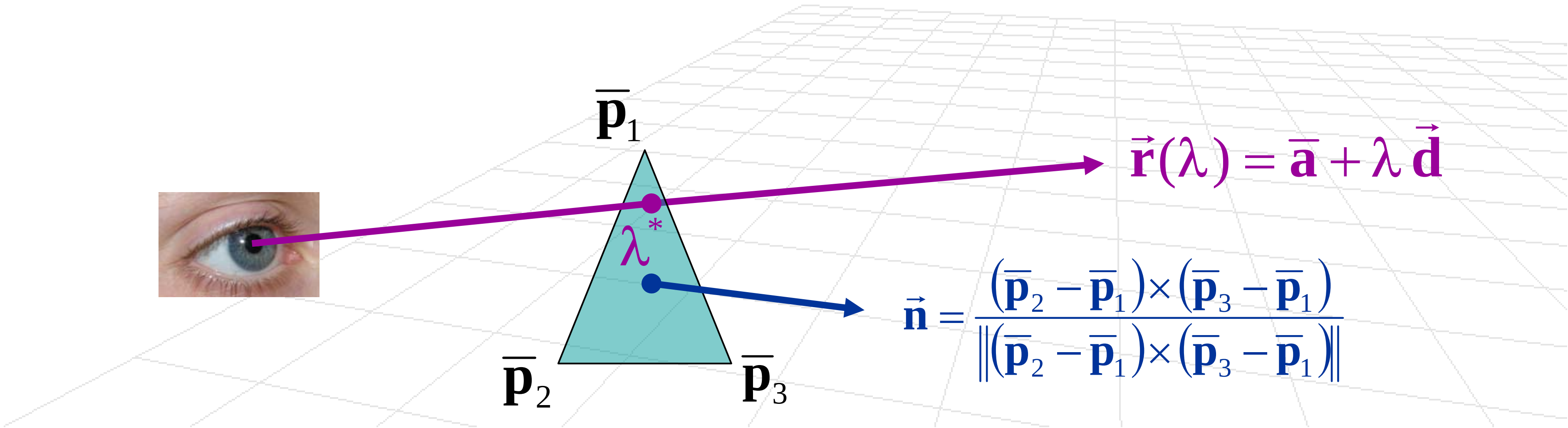
Algorithm 1

- Intersect the ray with the plane, $(\overline{\mathbf{p}} - \overline{\mathbf{p}}_1) \cdot \vec{\mathbf{n}} = 0$, in which the triangle lies

$$(\overline{\mathbf{a}} + \lambda^* \vec{\mathbf{d}} - \overline{\mathbf{p}}_1) \cdot \vec{\mathbf{n}} = 0$$

$$\lambda^* = \frac{(\overline{\mathbf{p}}_1 - \overline{\mathbf{a}}) \cdot \vec{\mathbf{n}}}{\vec{\mathbf{d}} \cdot \vec{\mathbf{n}}}$$

- Test if the intersection is within the triangle bounds (using half-plane tests)



$\overline{\mathbf{p}}_1$
$\vec{\mathbf{r}}(\lambda) = \overline{\mathbf{a}} + \lambda\,\vec{\mathbf{d}}$
λ^*
$\vec{\mathbf{n}} = \frac{(\overline{\mathbf{p}}_2 - \overline{\mathbf{p}}_1) \times (\overline{\mathbf{p}}_3 - \overline{\mathbf{p}}_1)}{\|(\overline{\mathbf{p}}_2 - \overline{\mathbf{p}}_1) \times (\overline{\mathbf{p}}_3 - \overline{\mathbf{p}}_1)\|}$
$\overline{\mathbf{p}}_2$
$\overline{\mathbf{p}}_3$

Finding Intersections (for triangle)

Algorithm 2

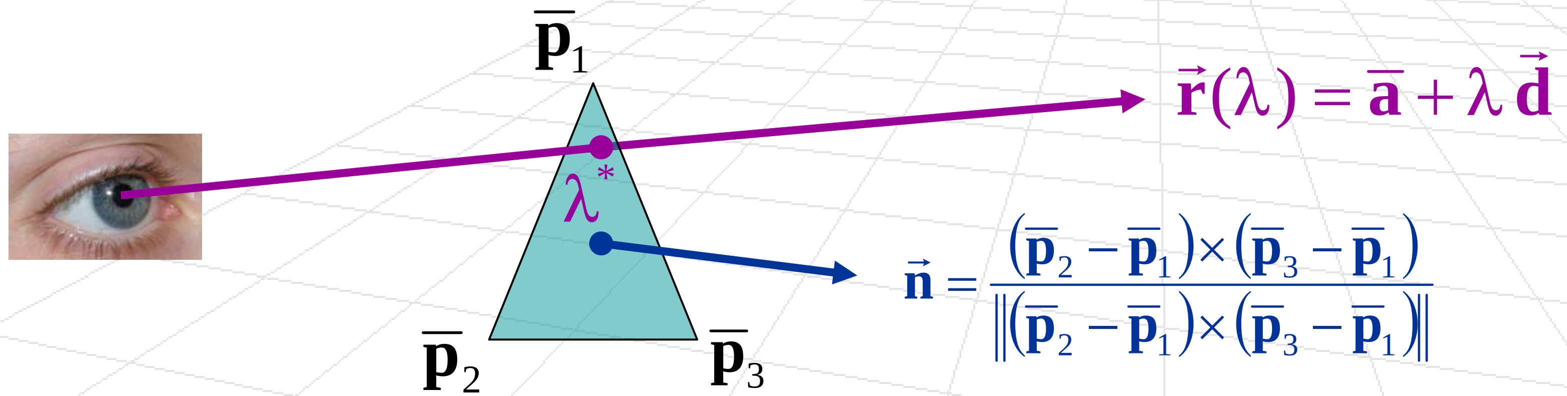
- Use an alternative parameterization for the plane

$$\mathbf{p}(\alpha,\beta) = \overline{\mathbf{p}}_1 + \alpha(\overline{\mathbf{p}}_2 - \overline{\mathbf{p}}_1) + \beta(\overline{\mathbf{p}}_3 - \overline{\mathbf{p}}_1)$$

  and solve for the parameters α,β , which leads 3x3 system

$$\begin{bmatrix} \uparrow & \uparrow & \uparrow \\ -(\overline{\mathbf{p}}_2 - \overline{\mathbf{p}}_1) & -(\overline{\mathbf{p}}_3 - \overline{\mathbf{p}}_1) & \vec{\mathbf{d}} \\ \downarrow & \downarrow & \downarrow \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \lambda^* \end{bmatrix} = (\overline{\mathbf{p}}_1 - \overline{\mathbf{a}})$$

- If intersection is in the triangle then the all following conditions will hold: $\alpha \geq 0, \quad \beta \geq 0, \quad \alpha + \beta \leq 1$

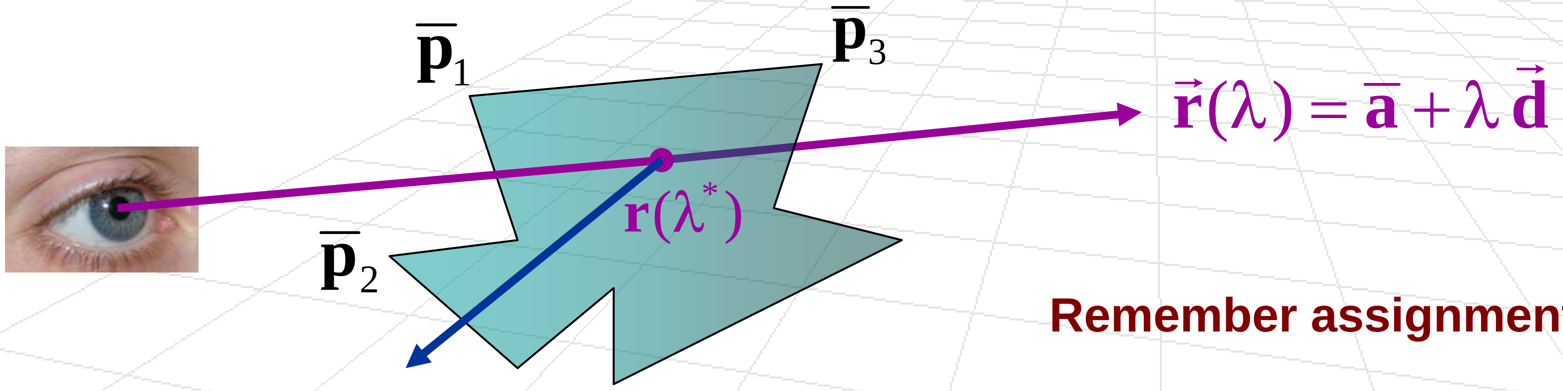

$\overline{\mathbf{p}}_1$
$\vec{\mathbf{r}}(\lambda) = \overline{\mathbf{a}} + \lambda\,\vec{\mathbf{d}}$
λ^*
$\vec{\mathbf{n}} = \frac{(\overline{\mathbf{p}}_2 - \overline{\mathbf{p}}_1)\times(\overline{\mathbf{p}}_3 - \overline{\mathbf{p}}_1)}{\|(\overline{\mathbf{p}}_2 - \overline{\mathbf{p}}_1)\times(\overline{\mathbf{p}}_3 - \overline{\mathbf{p}}_1)\|}$
$\overline{\mathbf{p}}_2$
$\overline{\mathbf{p}}_3$

Finding Intersections (for polygons)

- Find the intersection with the general planar polygon
  - Same as **Algorithm 2**
- Construct a ray in the plane originating at $\mathbf{r}(\lambda^*)$ and count the number of intersections with polygon sides:

# of intersections is odd => point inside the polygon

# of intersections is even => point outside the polygon


$\overline{\mathbf{p}}_1$
$\overline{\mathbf{p}}_3$
$\overline{\mathbf{p}}_2$
$\vec{\mathbf{r}}(\lambda) = \overline{\mathbf{a}} + \lambda\,\vec{\mathbf{d}}$
$\mathbf{r}(\lambda^*)$

Remember assignment 1

Ray Tracing
Part 2: Continuation

Computer Graphics, CSCD18

Fall 2008

Instructor: Leonid Sigal

Finding Intersections (for spheres)

- Let's consider simple case, unit length sphere at a given point $\bar{\mathbf{p}}$, $||\bar{\mathbf{p}}||^2=1$, by first substituting:

$$(\bar{\mathbf{a}} + \lambda^* \vec{\mathbf{d}}) \cdot (\bar{\mathbf{a}} + \lambda^* \vec{\mathbf{d}}) - 1 = 0$$

then expanding:

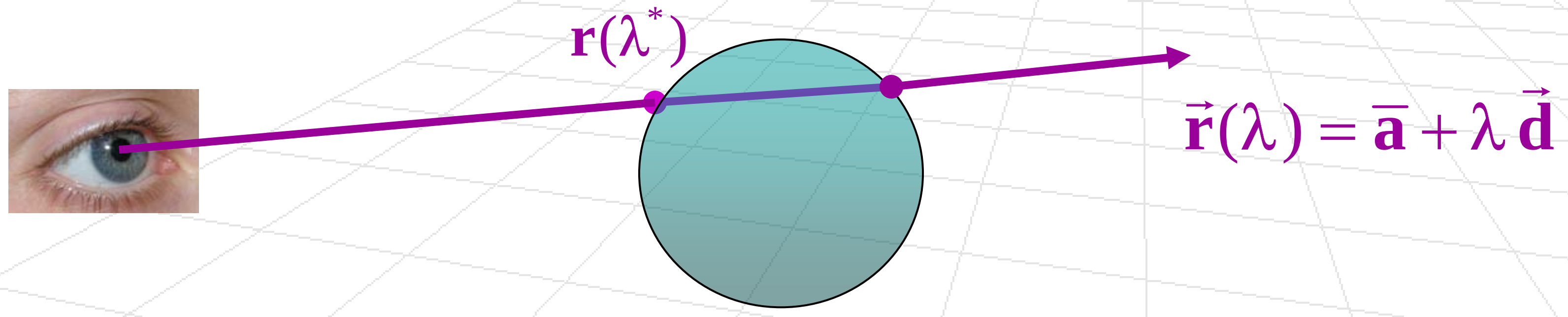
$$\mathbf{A}(\lambda^*)^2 + 2\mathbf{B}\lambda^* + \mathbf{C} = 0$$ **, where**

$$\mathbf{A} = \vec{\mathbf{d}} \cdot \vec{\mathbf{d}} \qquad \mathbf{C} = \bar{\mathbf{a}} \cdot \bar{\mathbf{a}} - 1$$

$$\mathbf{B} = \bar{\mathbf{a}} \cdot \vec{\mathbf{d}} \qquad \mathbf{D} = \mathbf{B}^2 - \mathbf{AC}$$

Producing **solution**:

$$\lambda^* = \frac{-2\mathbf{B} \pm \sqrt{4\mathbf{B}^2 - 4\mathbf{AC}}}{2\mathbf{A}} = -\frac{\mathbf{B}}{\mathbf{A}} \pm \frac{\sqrt{\mathbf{D}}}{\mathbf{A}}$$


$\mathbf{r}(\lambda^*)$
$\vec{\mathbf{r}}(\lambda) = \bar{\mathbf{a}} + \lambda \vec{\mathbf{d}}$

Finding Intersections (for spheres)

$$\lambda^* = \frac{-2\mathbf{B} \pm \sqrt{4\mathbf{B}^2 - 4\mathbf{AC}}}{2\mathbf{A}} = -\frac{\mathbf{B}}{\mathbf{A}} \pm \frac{\sqrt{\mathbf{D}}}{\mathbf{A}}$$

$$\mathbf{A} = \vec{\mathbf{d}} \cdot \vec{\mathbf{d}} \qquad \mathbf{C} = \overline{\mathbf{a}} \cdot \overline{\mathbf{a}} - 1$$

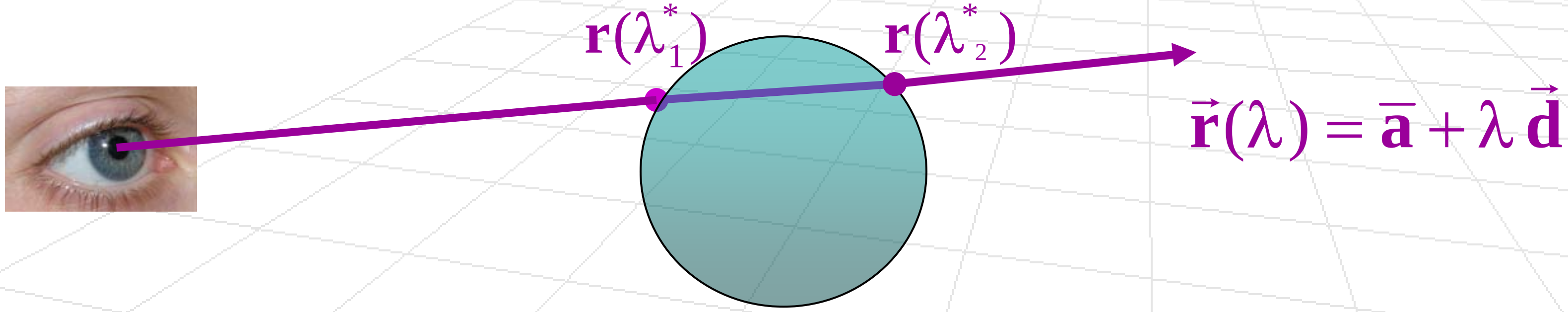
$$\mathbf{B} = \overline{\mathbf{a}} \cdot \vec{\mathbf{d}} \qquad \mathbf{D} = \mathbf{B}^2 - \mathbf{AC}$$

- Cases
  - D < 0 no intersection
  - D = 0 1 hit (ray grazes the sphere)
  - D > 0 2 hits with at $\mathbf{r}(\lambda_1^*), \mathbf{r}(\lambda_2^*)$

$\lambda_1^* < 0, \lambda_2^* < 0$ - Both hits are behind viewplane (not visible)

$\lambda_1^* > 0, \lambda_2^* < 0$ - Eye inside sphere, first hit is valid

$\lambda_1^* > \lambda_2^* > 0$ - Both hits are valid (in front of viewplane), 2nd closer


$\mathbf{r}(\lambda_1^*)$
$\mathbf{r}(\lambda_2^*)$
$\vec{\mathbf{r}}(\lambda) = \overline{\mathbf{a}} + \lambda\, \vec{\mathbf{d}}$

Finding Intersections (for spheres)

$$\lambda^* = \frac{-2\mathbf{B} \pm \sqrt{4\mathbf{B}^2 - 4\mathbf{AC}}}{2\mathbf{A}} = -\frac{\mathbf{B}}{\mathbf{A}} \pm \frac{\sqrt{\mathbf{D}}}{\mathbf{A}}$$

$$\mathbf{A} = \vec{\mathbf{d}} \cdot \vec{\mathbf{d}} \quad \mathbf{C} = \overline{\mathbf{a}} \cdot \overline{\mathbf{a}} - 1$$

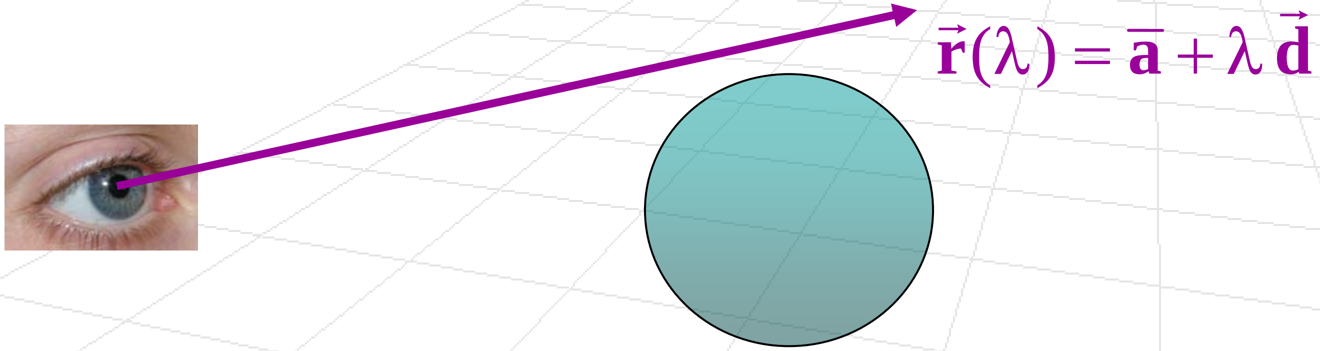
$$\mathbf{B} = \overline{\mathbf{a}} \cdot \vec{\mathbf{d}} \quad \mathbf{D} = \mathbf{B}^2 - \mathbf{AC}$$

- Cases
  - **D < 0 no intersection**
  - D = 0 1 hit (ray grazes the sphere)
  - D > 0 2 hits with at $\mathbf{r}(\lambda_1^*), \mathbf{r}(\lambda_2^*)$

$\lambda_1^* < 0, \lambda_2^* < 0$ - Both hits are behind viewplane (not visible)

$\lambda_1^* > 0, \lambda_2^* < 0$ - Eye inside sphere, first hit is valid

$\lambda_1^* > \lambda_2^* > 0$ - Both hits are valid (in front of viewplane), 2nd closer


$\vec{\mathbf{r}}(\lambda) = \overline{\mathbf{a}} + \lambda\,\vec{\mathbf{d}}$

Finding Intersections (for spheres)

$$\lambda^* = \frac{-2\mathbf{B} \pm \sqrt{4\mathbf{B}^2 - 4\mathbf{AC}}}{2\mathbf{A}} = -\frac{\mathbf{B}}{\mathbf{A}} \pm \frac{\sqrt{\mathbf{D}}}{\mathbf{A}}$$

$$\mathbf{A} = \vec{\mathbf{d}} \cdot \vec{\mathbf{d}} \qquad \mathbf{C} = \bar{\mathbf{a}} \cdot \bar{\mathbf{a}} - 1$$

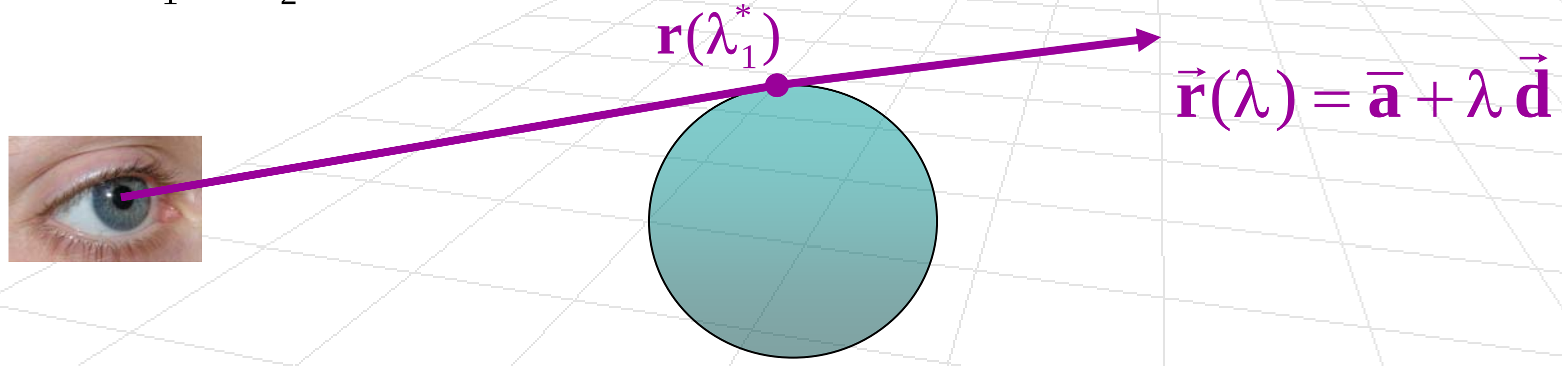
$$\mathbf{B} = \bar{\mathbf{a}} \cdot \vec{\mathbf{d}} \qquad \mathbf{D} = \mathbf{B}^2 - \mathbf{AC}$$

- Cases
  - D < 0 no intersection
  - **D = 0 1 hit (ray grazes the sphere)**
  - D > 0 2 hits with at $\mathbf{r}(\lambda_1^*), \mathbf{r}(\lambda_2^*)$

    $\lambda_1^* < 0, \lambda_2^* < 0$ - Both hits are behind viewplane (not visible)

    $\lambda_1^* > 0, \lambda_2^* < 0$ - Eye inside sphere, first hit is valid

    $\lambda_1^* > \lambda_2^* > 0$ - Both hits are valid (in front of viewplane), 2nd closer


$\mathbf{r}(\lambda_1^*)$
$\vec{\mathbf{r}}(\lambda) = \bar{\mathbf{a}} + \lambda\,\vec{\mathbf{d}}$

Finding Intersections (for spheres)

$$\lambda^* = \frac{-2\mathbf{B} \pm \sqrt{4\mathbf{B}^2 - 4\mathbf{A}\mathbf{C}}}{2\mathbf{A}} = -\frac{\mathbf{B}}{\mathbf{A}} \pm \frac{\sqrt{\mathbf{D}}}{\mathbf{A}}$$

$$\mathbf{A} = \vec{\mathbf{d}} \cdot \vec{\mathbf{d}} \qquad \mathbf{C} = \bar{\mathbf{a}} \cdot \bar{\mathbf{a}} - 1$$

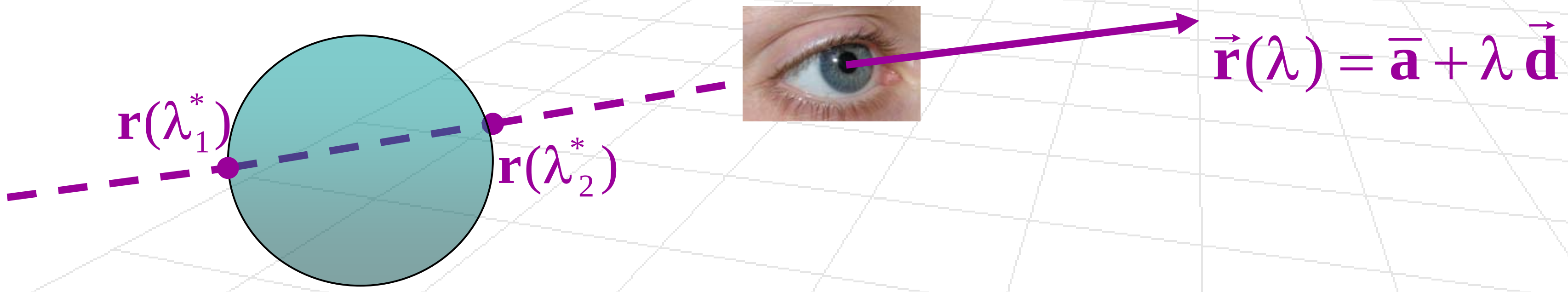
$$\mathbf{B} = \bar{\mathbf{a}} \cdot \vec{\mathbf{d}} \qquad \mathbf{D} = \mathbf{B}^2 - \mathbf{AC}$$

- Cases
  - D < 0 no intersection
  - D = 0 1 hit (ray grazes the sphere)
  - D > 0 2 hits with at $\mathbf{r}(\lambda_1^*), \mathbf{r}(\lambda_2^*)$

$\lambda_1^* < 0, \lambda_2^* < 0$ **- Both hits are behind viewplane (not visible)**

$\lambda_1^* > 0, \lambda_2^* < 0$ - Eye inside sphere, first hit is valid

$\lambda_1^* > \lambda_2^* > 0$ - Both hits are valid (in front of viewplane), 2nd closer


$\mathbf{r}(\lambda_1^*)$
$\mathbf{r}(\lambda_2^*)$
$\vec{\mathbf{r}}(\lambda) = \bar{\mathbf{a}} + \lambda\,\vec{\mathbf{d}}$

Finding Intersections (for spheres)

$$\lambda^* = \frac{-2\mathbf{B} \pm \sqrt{4\mathbf{B}^2 - 4\mathbf{AC}}}{2\mathbf{A}} = -\frac{\mathbf{B}}{\mathbf{A}} \pm \frac{\sqrt{\mathbf{D}}}{\mathbf{A}}$$

$$\mathbf{A} = \vec{\mathbf{d}} \cdot \vec{\mathbf{d}} \qquad \mathbf{C} = \bar{\mathbf{a}} \cdot \bar{\mathbf{a}} - 1$$

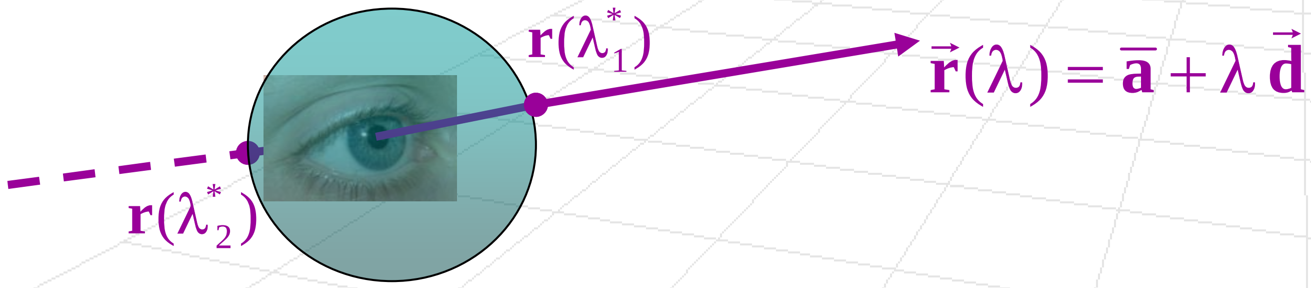
$$\mathbf{B} = \bar{\mathbf{a}} \cdot \vec{\mathbf{d}} \qquad \mathbf{D} = \mathbf{B}^2 - \mathbf{AC}$$

- Cases
  - D < 0 no intersection
  - D = 0 1 hit (ray grazes the sphere)
  - D > 0 2 hits with at $\mathbf{r}(\lambda_1^*), \mathbf{r}(\lambda_2^*)$

$\lambda_1^* < 0, \lambda_2^* < 0$ - Both hits are behind viewplane (not visible)

$\lambda_1^* > 0, \lambda_2^* < 0$ - Eye inside sphere, first hit is valid

$\lambda_1^* > \lambda_2^* > 0$ - Both hits are valid (in front of viewplane), 2nd closer


$\mathbf{r}(\lambda_1^*)$
$\vec{\mathbf{r}}(\lambda) = \bar{\mathbf{a}} + \lambda\,\vec{\mathbf{d}}$
$\mathbf{r}(\lambda_2^*)$

Finding Intersections (for spheres)

$$\lambda^* = \frac{-2\mathbf{B} \pm \sqrt{4\mathbf{B}^2 - 4\mathbf{AC}}}{2\mathbf{A}} = -\frac{\mathbf{B}}{\mathbf{A}} \pm \frac{\sqrt{\mathbf{D}}}{\mathbf{A}}$$

$$\mathbf{A} = \vec{\mathbf{d}} \cdot \vec{\mathbf{d}} \qquad \mathbf{C} = \bar{\mathbf{a}} \cdot \bar{\mathbf{a}} - 1$$

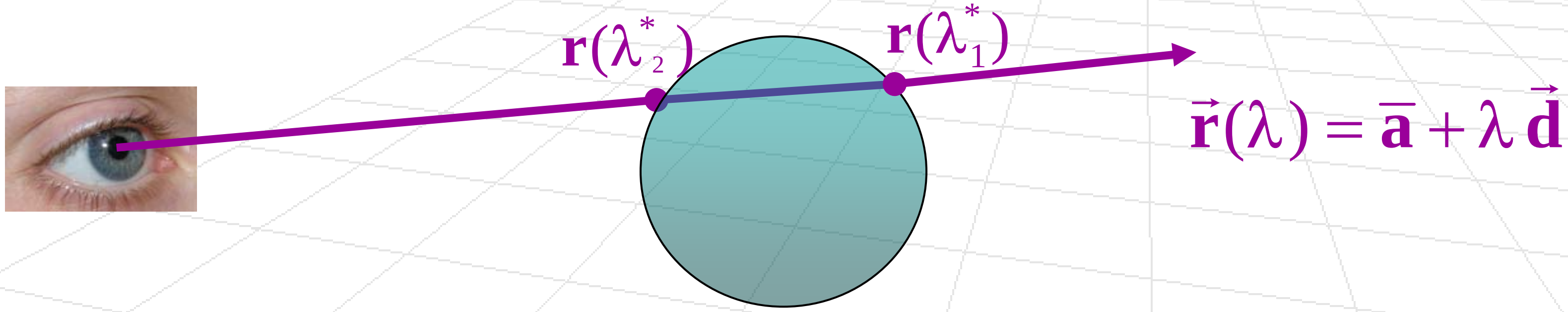
$$\mathbf{B} = \bar{\mathbf{a}} \cdot \vec{\mathbf{d}} \qquad \mathbf{D} = \mathbf{B}^2 - \mathbf{AC}$$

- Cases
  - D < 0 no intersection
  - D = 0 1 hit (ray grazes the sphere)
  - D > 0 2 hits with at $\mathbf{r}(\lambda^*_1), \mathbf{r}(\lambda^*_2)$

$\lambda^*_1 < 0, \lambda^*_2 < 0$ - Both hits are behind viewplane (not visible)

$\lambda^*_1 > 0, \lambda^*_2 < 0$ - Eye inside sphere, first hit is valid

$\lambda^*_1 > \lambda^*_2 > 0$ **- Both hits are valid (in front of viewplane), 2nd closer**


$\mathbf{r}(\lambda^*_2)$
$\mathbf{r}(\lambda^*_1)$
$\vec{\mathbf{r}}(\lambda) = \bar{\mathbf{a}} + \lambda\,\vec{\mathbf{d}}$

What if we want to intersect with a general sphere?

- **Property:** Given an intersection method for an object, it is easy to intersect rays with affinely deformed versions of the object

- Assuming we have an invertible affine transformation $\mathbf{f}(\bar{\mathbf{x}}) = \mathbf{A}\bar{\mathbf{x}}+\vec{\mathbf{t}}$ that is applied to the object, it is relatively easy to show that intersection of the deformed object is the same as the intersection of the original object with inversely transformed ray:

$$\boxed{\vec{\mathbf{r}}'(\lambda) = \bar{\mathbf{a}}' + \lambda \vec{\mathbf{d}}'}$$

$$\bar{\mathbf{a}}' = \mathbf{A}^{-1}(\bar{\mathbf{a}} - \vec{\mathbf{t}})$$

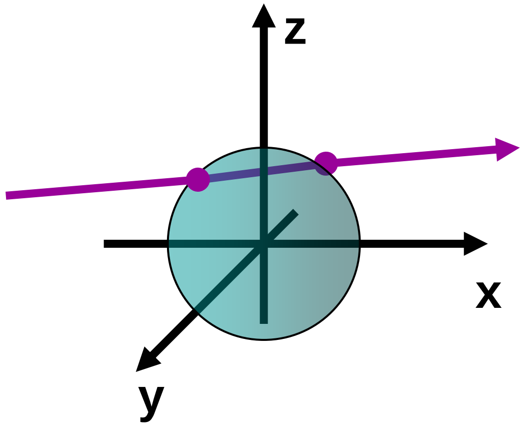
$$\vec{\mathbf{d}}' = \mathbf{A}^{-1}\vec{\mathbf{d}}$$

Full proof is in the lecture notes

Short Proof

Given intersection method for

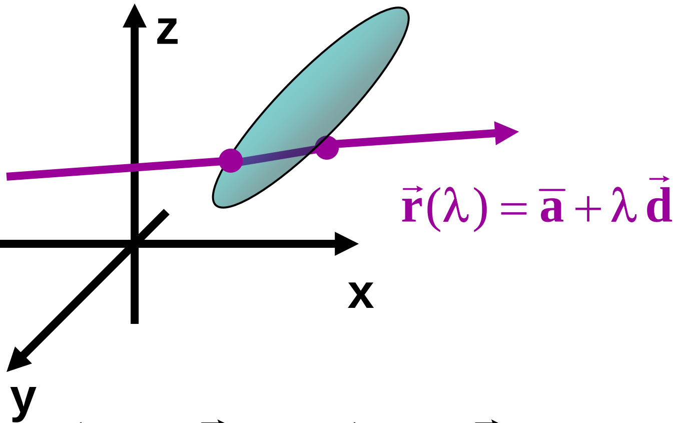
$$\mathbf{f}(\bar{\mathbf{x}})$$



$$\mathbf{f}(\bar{\mathbf{x}}) = \bar{\mathbf{x}} \cdot \bar{\mathbf{x}} - 1 = 0$$

Find intersection with affine version

$$\mathbf{F}(\bar{\mathbf{y}}) = \mathbf{f}\left(\mathbf{A}^{-1}(\bar{\mathbf{y}} - \vec{\mathbf{t}})\right)$$


z
x
y
$\vec{\mathbf{r}}(\lambda) = \bar{\mathbf{a}} + \lambda\,\vec{\mathbf{d}}$

$$\mathbf{F}(\bar{\mathbf{y}}) = \mathbf{A}^{-1}(\bar{\mathbf{y}} - \vec{\mathbf{t}}) \cdot \mathbf{A}^{-1}(\bar{\mathbf{y}} - \vec{\mathbf{t}}) - 1 = 0$$

We substitute equation of the ray into implicit function of the affinely transformed surface

$$\begin{aligned}
\mathbf{F}(\bar{\mathbf{r}}(\lambda)) &= \mathbf{f}\left(\mathbf{A}^{-1}\left(\bar{\mathbf{r}}(\lambda) - \vec{\mathbf{t}}\right)\right) \\
&= \mathbf{f}\left(\mathbf{A}^{-1}\left(\bar{\mathbf{a}} + \lambda\,\vec{\mathbf{d}} - \vec{\mathbf{t}}\right)\right) \\
&= \mathbf{f}\left(\mathbf{A}^{-1}\left(\bar{\mathbf{a}} - \vec{\mathbf{t}}\right) + \lambda\,\mathbf{A}^{-1}\vec{\mathbf{d}}\right) \\
&= \mathbf{f}\left(\vec{\mathbf{r}}'(\lambda)\right)
\end{aligned}$$

where

$$\vec{\mathbf{r}}'(\lambda) = \bar{\mathbf{a}}' + \lambda\,\vec{\mathbf{d}}'$$

$$\bar{\mathbf{a}}' = \mathbf{A}^{-1}\left(\bar{\mathbf{a}} + \vec{\mathbf{t}}\right)$$

$$\vec{\mathbf{d}}' = \mathbf{A}^{-1}\vec{\mathbf{d}}$$

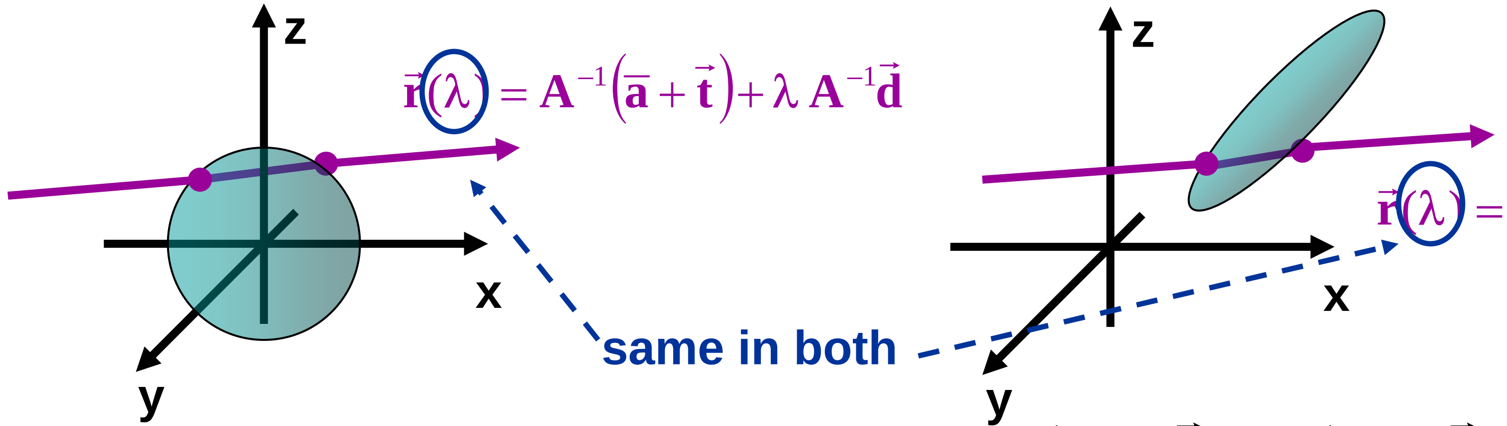
Short Proof

Given intersection method for

$$\mathbf{f}(\bar{\mathbf{x}})$$

Find intersection with affine version

$$\mathbf{F}(\bar{\mathbf{y}})=\mathbf{f}\left(\mathbf{A}^{-1}(\bar{\mathbf{y}}-\vec{\mathbf{t}})\right)$$



$$\vec{\mathbf{r}}(\lambda)=\mathbf{A}^{-1}\left(\bar{\mathbf{a}}+\vec{\mathbf{t}}\right)+\lambda\,\mathbf{A}^{-1}\vec{\mathbf{d}}$$

$$\vec{\mathbf{r}}(\lambda)=\bar{\mathbf{a}}+\lambda\,\vec{\mathbf{d}}$$

$$\mathbf{f}(\bar{\mathbf{x}})=\bar{\mathbf{x}}\cdot\bar{\mathbf{x}}-1=0$$

$$\mathbf{F}(\bar{\mathbf{y}})=\mathbf{A}^{-1}(\bar{\mathbf{y}}-\vec{\mathbf{t}})\cdot\mathbf{A}^{-1}(\bar{\mathbf{y}}-\vec{\mathbf{t}})-1=0$$

We substitute equation of the ray into implicit function of the affinely transformed surface

$$\begin{aligned}\mathbf{F}(\bar{\mathbf{r}}(\lambda))&=\mathbf{f}\left(\mathbf{A}^{-1}\left(\bar{\mathbf{r}}(\lambda)-\vec{\mathbf{t}}\right)\right)\\&=\mathbf{f}\left(\mathbf{A}^{-1}\left(\bar{\mathbf{a}}+\lambda\,\vec{\mathbf{d}}-\vec{\mathbf{t}}\right)\right)\\&=\mathbf{f}\left(\mathbf{A}^{-1}\left(\bar{\mathbf{a}}-\vec{\mathbf{t}}\right)+\lambda\,\mathbf{A}^{-1}\vec{\mathbf{d}}\right)\\&=\mathbf{f}\left(\vec{\mathbf{r}}'(\lambda)\right)\end{aligned}$$

where

$$\vec{\mathbf{r}}'(\lambda)=\bar{\mathbf{a}}'+\lambda\,\vec{\mathbf{d}}'$$

$$\bar{\mathbf{a}}'=\mathbf{A}^{-1}\left(\bar{\mathbf{a}}+\vec{\mathbf{t}}\right)$$

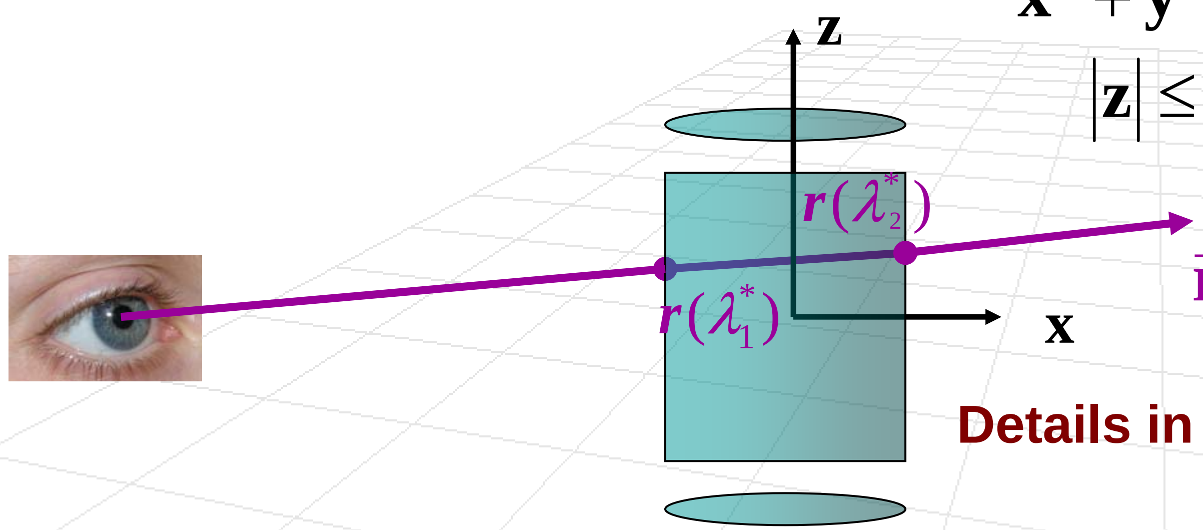
$$\vec{\mathbf{d}}'=\mathbf{A}^{-1}\vec{\mathbf{d}}$$

Intersection with Cylinders and Cones

- Find intersection with "quadratic walls", ignoring constraints on $\mathbf{z}$
- Test the resulting $\mathbf{z}$ component of $\overline{\mathbf{p}}(\lambda^*)$ against the constraints on $\mathbf{z}$
- Intersect the ray with plains constraining end caps
- Test to ensure they satisfy interior circular constraint
- If there are multiple intersections take the intersection with smallest positive λ^*

$$\mathbf{x}^2 + \mathbf{y}^2 = 1$$

$$|\mathbf{z}| \leq 1$$



$$\vec{\mathbf{r}}(\lambda) = \overline{\mathbf{a}} + \lambda\,\vec{\mathbf{d}}$$

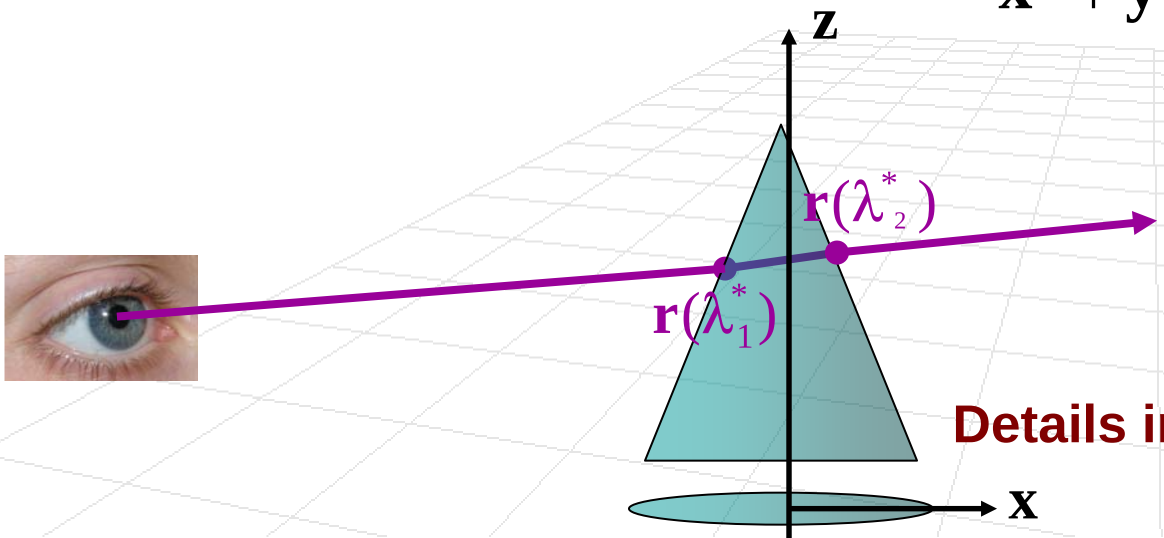
Details in the lecture notes

Intersection with Cylinders and Cones

- Find intersection with “quadratic walls”, ignoring constraints on $\mathbf{z}$
- Test the resulting $\mathbf{z}$ component of $\overline{\mathbf{p}}(\lambda^*)$ against the constraints on $\mathbf{z}$
- Intersect the ray with plains constraining end caps
- Test to ensure they satisfy interior circular constraint
- If there are multiple intersections take the intersection with smallest positive λ^*

$$\mathbf{x}^2 + \mathbf{y}^2 - \frac{1}{4}\left(1 - \mathbf{z}^2\right) = 0$$

$$0 \le \mathbf{z} \le 1$$



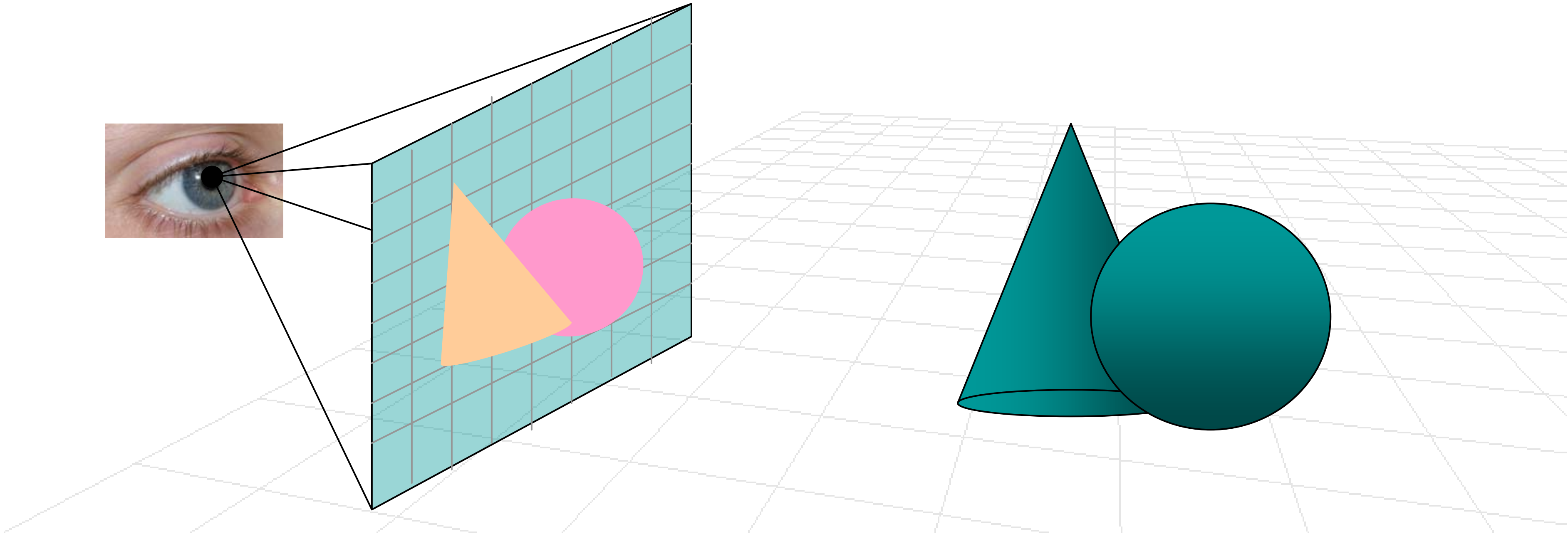
$$\vec{\mathbf{r}}(\lambda) = \overline{\mathbf{a}} + \lambda\,\vec{\mathbf{d}}$$

Details in the lecture notes

Scene Signature

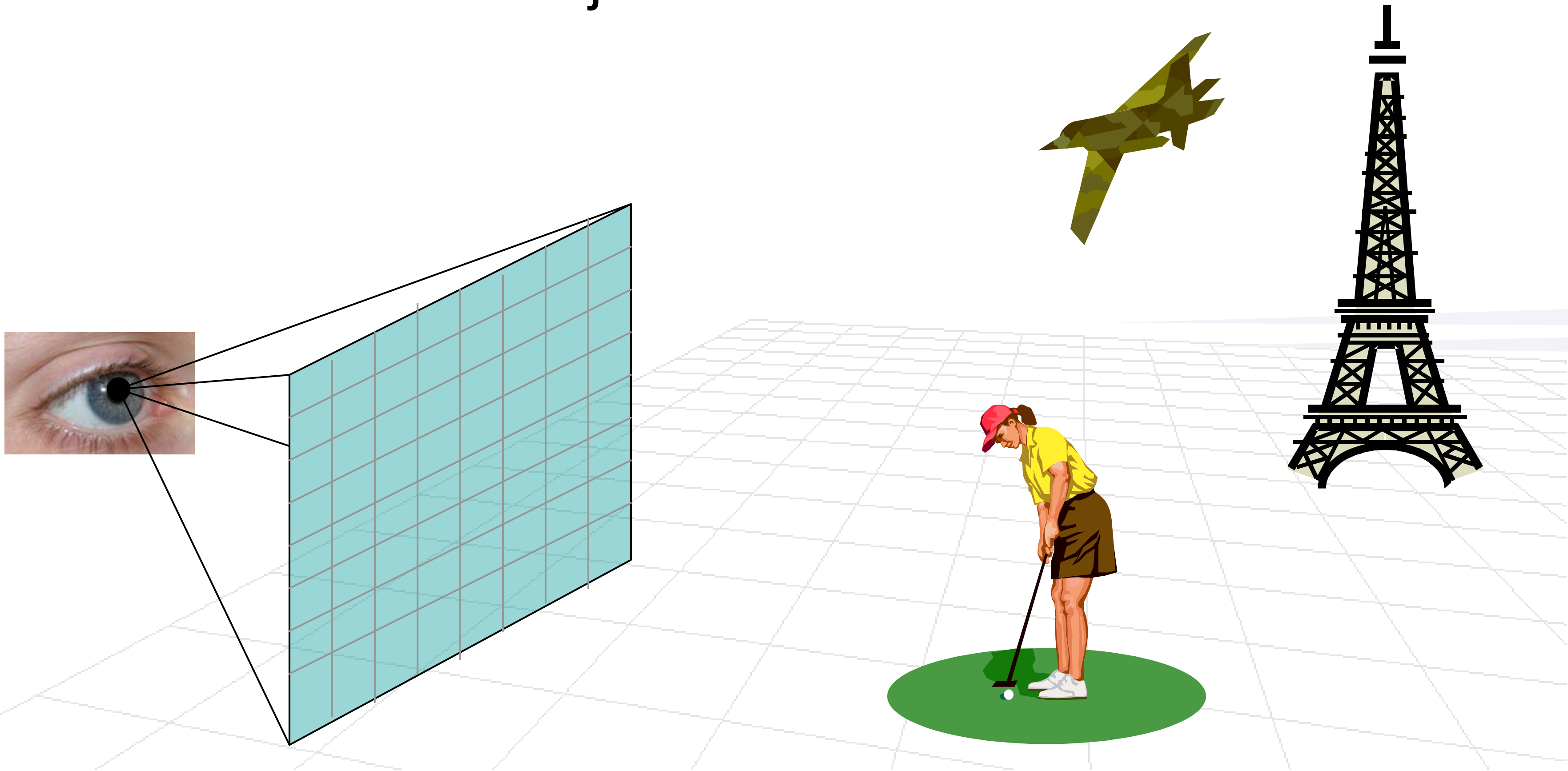
- Simple way to test geometry and intersection methods (great tool for debugging)

- **Idea:** Create image in which pixel (**i**,**j**) has intensity **k** if object **k** is the first object hit by the ray through (**i**,**j**)
  - Each object gets one unique color



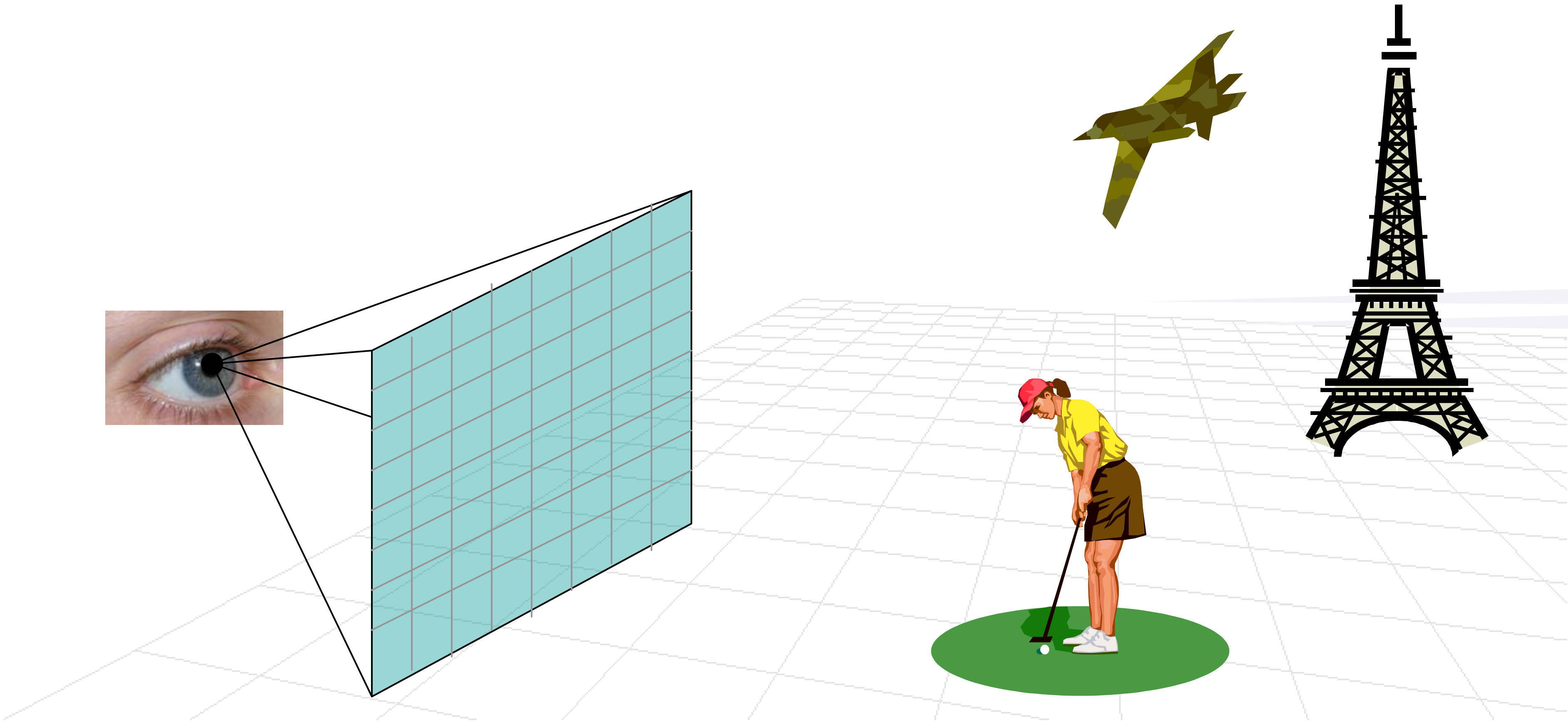
Efficiency

- Intersection tests are expensive, especially for complex geometries.
- Data structures are typically used to avoid testing intersections with objects that are not hit.



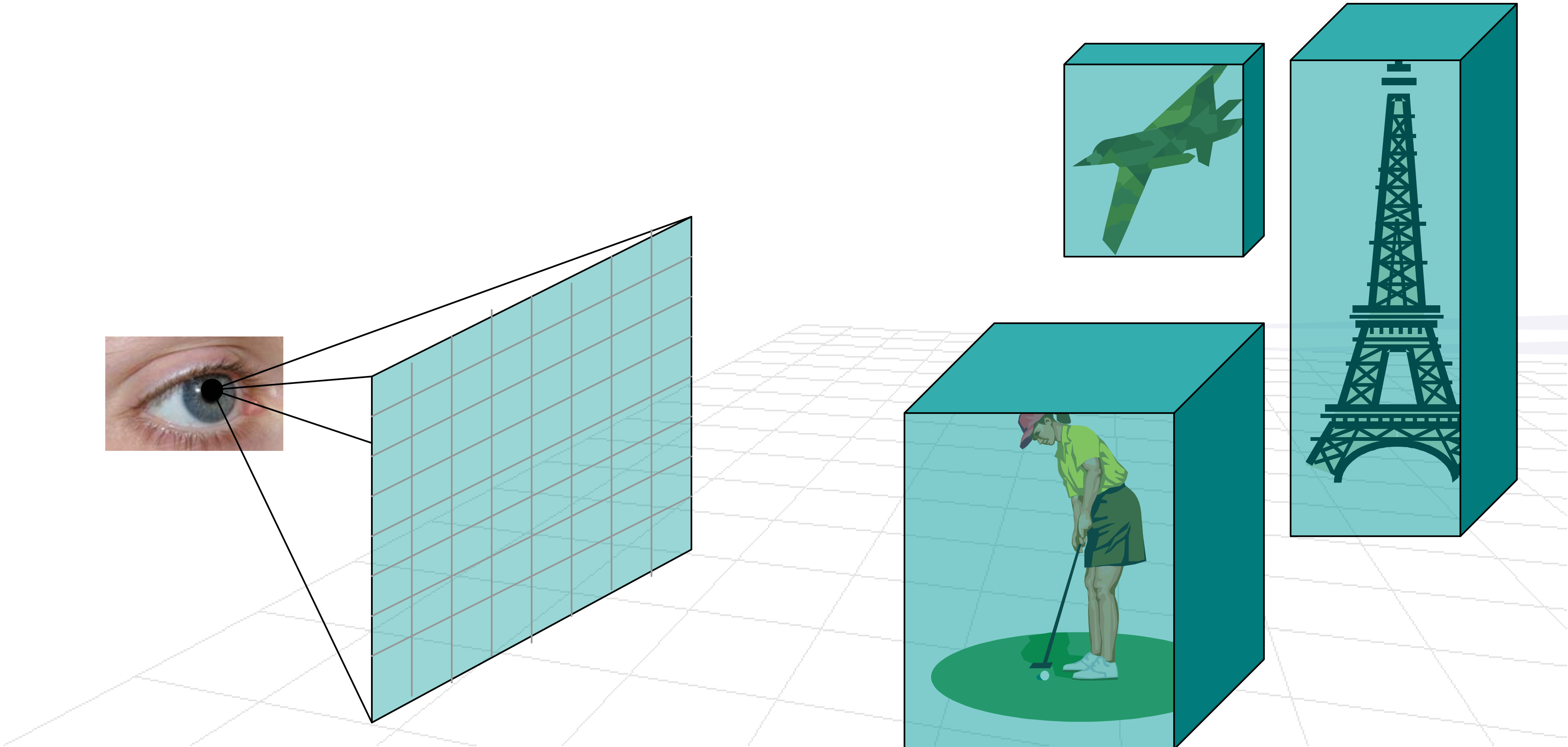
Efficiency

- **Idea:** Bound 3D objects (meshes) with single simple bounding volume (e.g. sphere or a cube).
  - Only test intersections with the object if there exists positive intersection with bounding volume



Efficiency

- **Idea:** Bound 3D objects (meshes) with single simple bounding volume (e.g. sphere or a cube).
  - Only test intersections with the object if there exists positive intersection with bounding volume



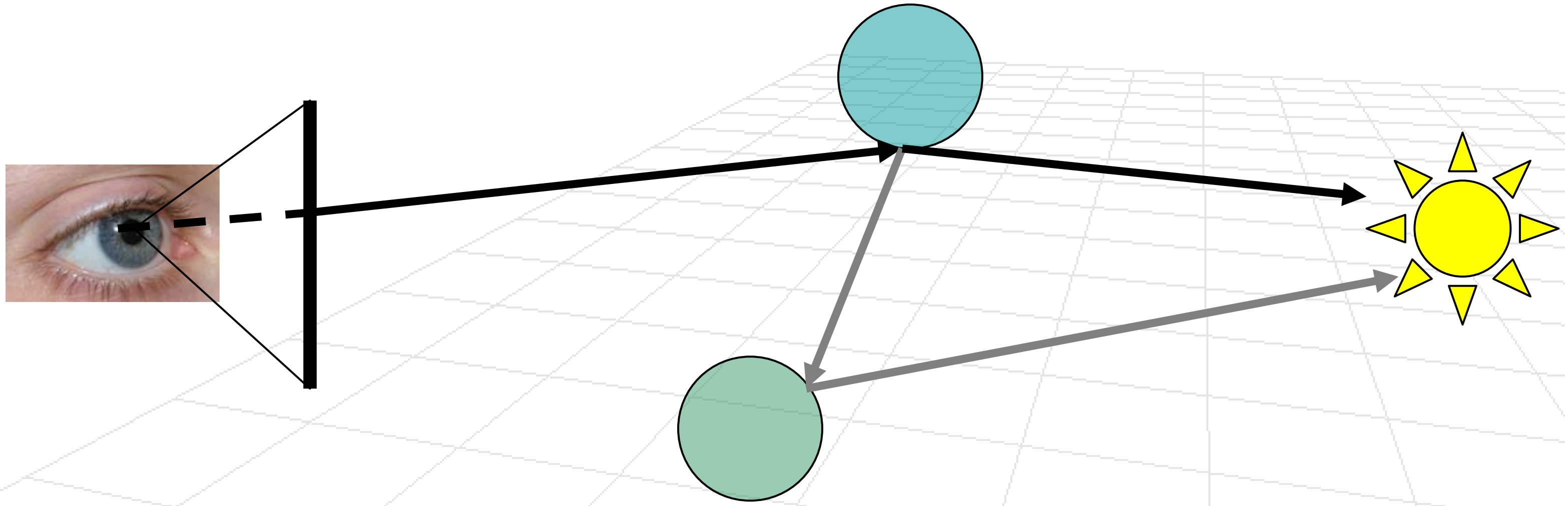
Efficiency

- **Idea:** Bound 3D objects (meshes) with single simple bounding volume (e.g. sphere or a cube).
  - Only test intersections with the object if there exists positive intersection with bounding volume

(This can also be done hierarchically)

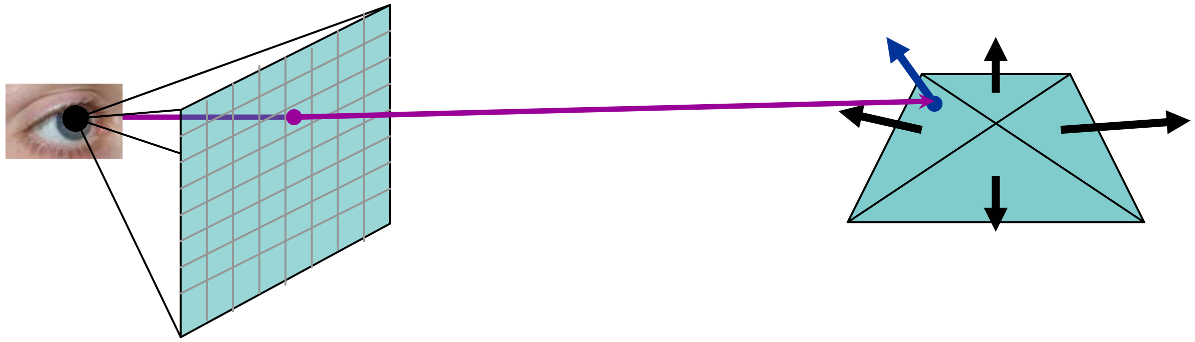
Computational Issues to Consider

- Form rays (a.k.a. ray casting)
- Find intersection of rays with objects
- Find closest object intersection (there could be multiple object intersections for any given ray)
- Find normal at the closest intersection point (a.k.a hit point)
- Evaluate reflectance model at the hit point



Finding surface normal at the hit point

- For mesh surface one might smoothly interpolate normal at the hit point from the nearby face normals (same as for the vertex before)



- Given parametric shape, we can compute normal $\vec{\mathbf{n}}_j$ for the hit point $\overline{\mathbf{p}}_j$ explicitly
  - **Implicit form** $\vec{\mathbf{n}}_j(\overline{\mathbf{p}}_j) = \nabla \mathbf{f}(\overline{\mathbf{p}}_j)$
  - **Explicit form** $\vec{\mathbf{n}}_j(\overline{\mathbf{p}}_j) = \left.\frac{\partial \mathbf{s}(\alpha,\beta)}{\partial \alpha}\right|_{\alpha_0,\beta_0} \times \left.\frac{\partial \mathbf{s}(\alpha,\beta)}{\partial \beta}\right|_{\alpha_0,\beta_0}$

Normals for affinely-deformed surfaces

- Let $\mathbf{f}(\overline{\mathbf{p}}) = 0$ be an implicit surface, and let $\mathbf{Q}(\overline{\mathbf{p}}) = \mathbf{A}\overline{\mathbf{p}} + \vec{\mathbf{t}}$ be an affine transformation.

- The new affinely-deformed surface can then be written as follows: $\mathbf{F}(\overline{\mathbf{q}}) = \mathbf{f}\left(\mathbf{A}^{-1}(\overline{\mathbf{p}} - \vec{\mathbf{t}})\right) = 0$

- The normal of $\mathbf{F}$ at a point $\overline{\mathbf{q}}$ is given by

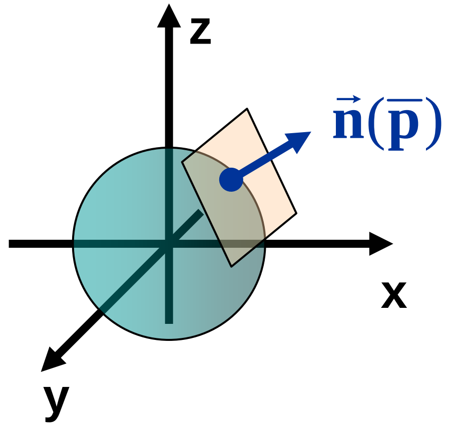
$$\boxed{\frac{\left(\mathbf{A}^{-1}\right)^{\mathbf{T}}\vec{\mathbf{n}}}{\left\|\left(\mathbf{A}^{-1}\right)^{\mathbf{T}}\vec{\mathbf{n}}\right\|}}$$

Full proof in given in the lecture notes

Short Proof

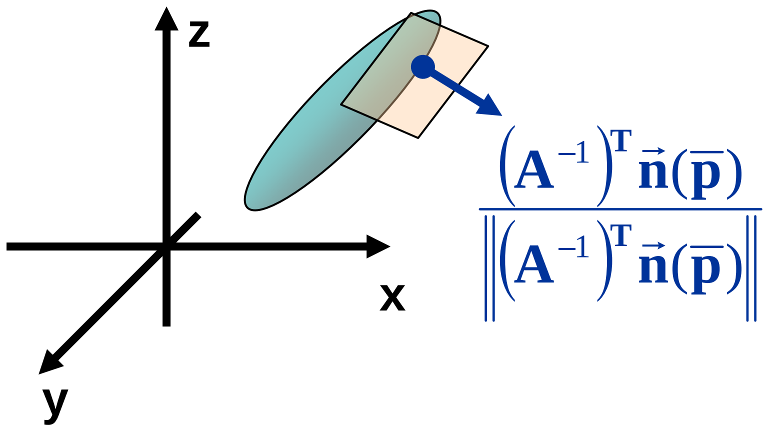
Given a normal

$$\vec{n}(\bar{p}) = \nabla f(\bar{p}) / \|\nabla f(\bar{p})\|$$



Find a normal for affine version

$$F(\bar{q}) = f\left(A^{-1}(\bar{q} - \vec{t})\right)$$



$$\frac{\left(A^{-1}\right)^T \vec{n}(\bar{p})}{\left\|\left(A^{-1}\right)^T \vec{n}(\bar{p})\right\|}$$

Assume that the tangent plane at hit point, $\bar{r}(\lambda^*)$, on the generic surface is given by $\left(\bar{p} - \bar{r}(\lambda^*)\right) \cdot \vec{n} = 0$ or $\bar{p}^T \vec{n} = D$, where $D = \bar{r}(\lambda^*) \cdot \vec{n}$

Under affine transformation, planarity is preserved so tangent plane on deformed surface is given by

$$\left(A^{-1}(\bar{q} - \vec{t})\right)^T \vec{n} = D$$

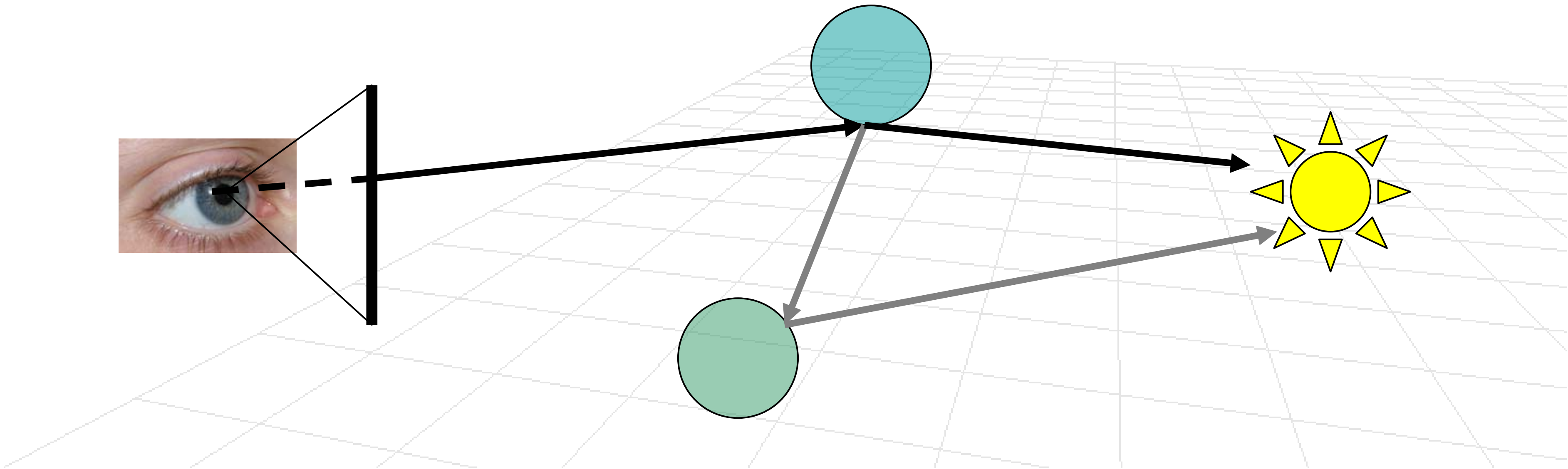
$$\left(A^{-1}\bar{q} - A^{-1}\vec{t}\right)^T \vec{n} = D$$

$$\left(A^{-1}\bar{q}\right)^T \vec{n} = D + \left(A^{-1}\vec{t}\right)^T$$

$$\bar{q}^T \left(A^{-1}\right)^T \vec{n} = D + \left(A^{-1}\vec{t}\right)^T$$

Computational Issues to Consider

- Form rays (a.k.a. ray casting)
- Find intersection of rays with objects
- Find closest object intersection (there could be multiple object intersections for any given ray)
- Find normal at the closest intersection point (a.k.a hit point)
- Evaluate reflectance model at the hit point



Conventional (Whitted) Ray Tracing

- **Local** model (e.g. Phong) to account for diffuse and specular highlights due to the direct lighting
- Use ambient term to approximate **global** diffuse lighting
- Cast rays to estimate ideal "mirror" reflections of other objects
  - Feasible since for perfect specular reflection there is a unique direction of preference
  - In general, however, we need a Bidirectional Reflectance Distribution Function (BRDF)
- So radiance at a hit point becomes

$\mathbf{r}_g$ can depend on distance

$$\mathbf{E}_j = \underbrace{\mathbf{r}_d \mathbf{I}_d \max(0, \vec{\mathbf{s}}_j \cdot \vec{\mathbf{n}}_j) + \mathbf{r}_a \mathbf{I}_a + \mathbf{r}_s \mathbf{I}_s \max(0, \vec{\mathbf{r}}_j \cdot \vec{\mathbf{c}}_j)^{\alpha}}_{\text{Local Phong Model}} + \underbrace{\mathbf{r}_g \mathbf{I}_{\mathbf{spec}}}_{\text{Global Specular Model}}$$

Texture

- Texture can be used to modulate diffuse and ambient reflection coefficients, as with Gouraud or Phong shading

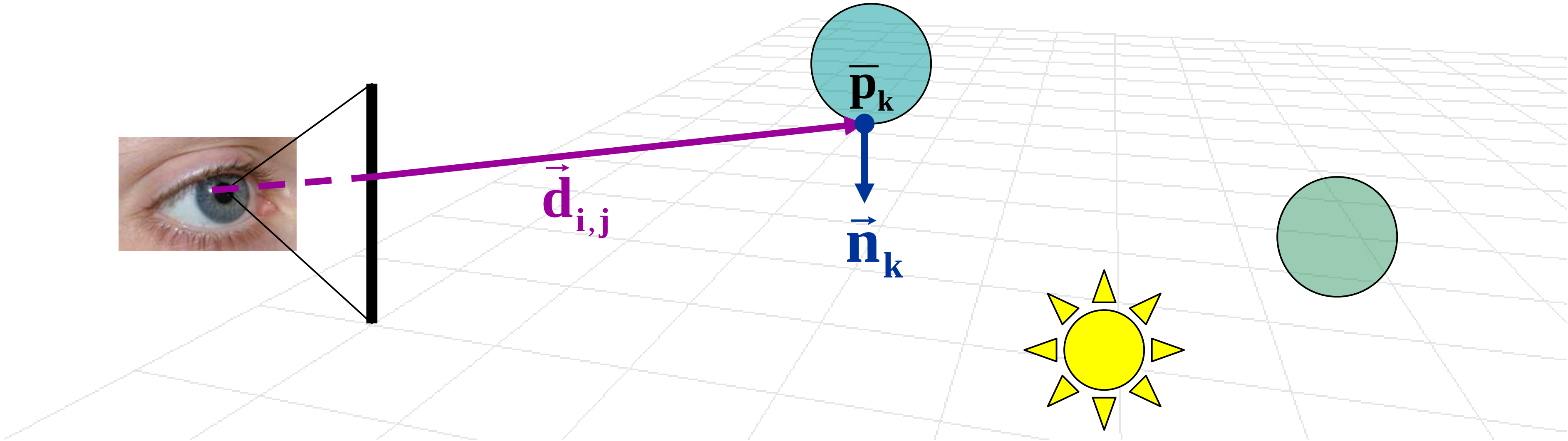
- All we need, is a way of mapping a point on the surface (hit point) to a point in the texture space

- Unlike with Gouraud or Phong shading models we don't need to interpolate texture coordinates over polygons

- Anti-aliasing and super-sampling we will cover later (next week)

Conventional (Whitted) Ray Tracing

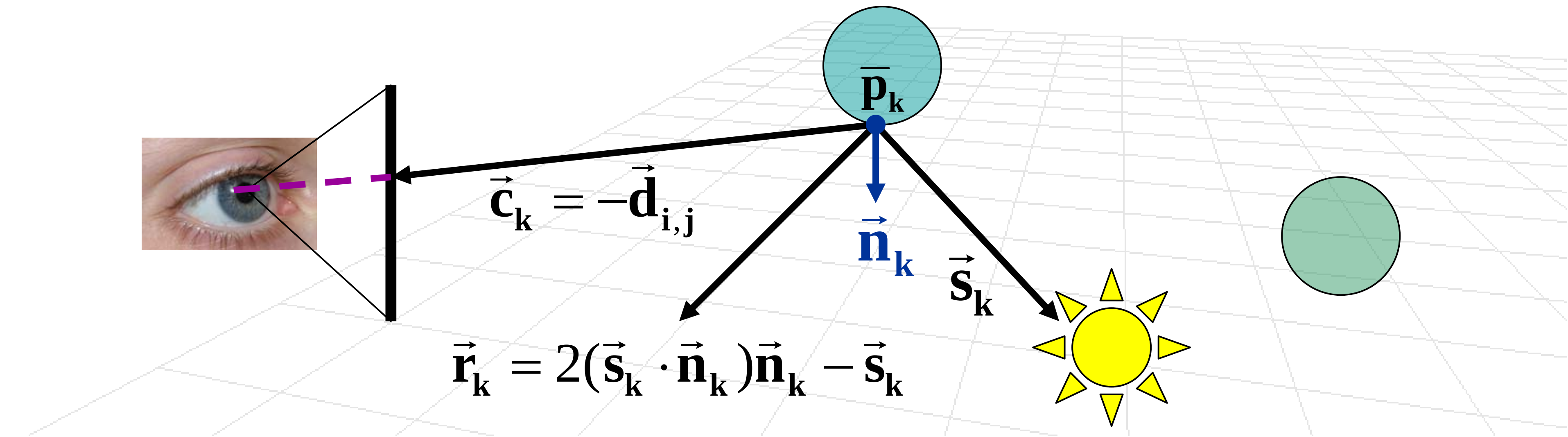
- Cast a ray and find
  - “hit point” and normal at the “hit point”



Conventional (Whitted) Ray Tracing

- Cast a ray and find
  - “hit point” and normal at the “hit point”
- Compute **local lighting** at the “hit point” (Phong)

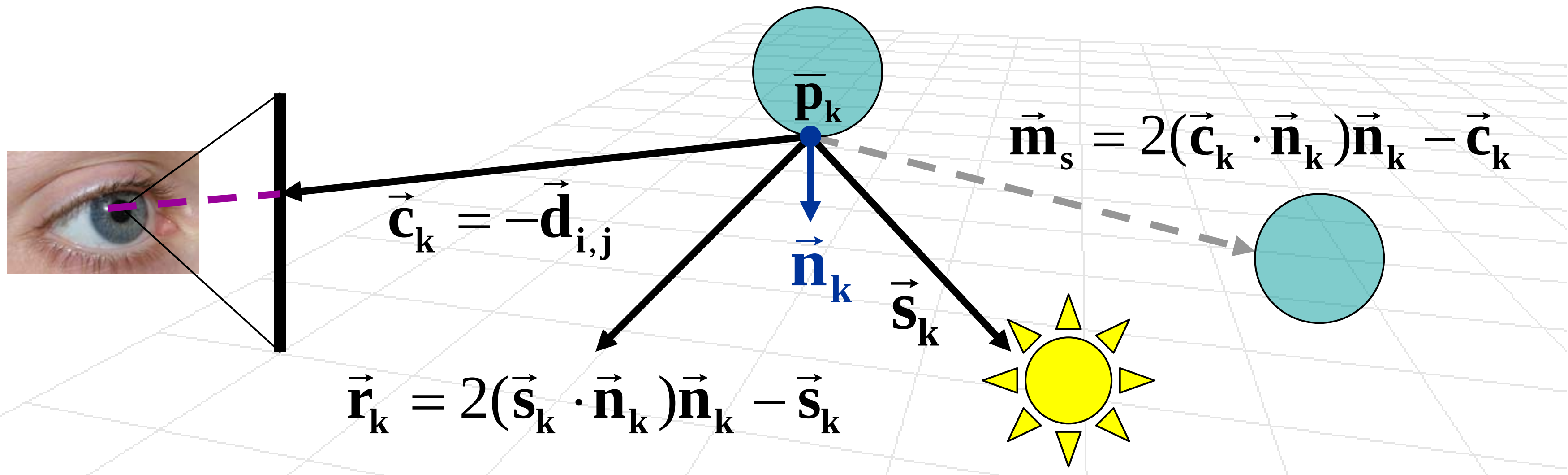
$$\mathbf{E_k} = \mathbf{r_d I_d} \max(0, \vec{\mathbf{s}}_\mathbf{k} \cdot \vec{\mathbf{n}}_\mathbf{k}) + \mathbf{r_a I_a} + \mathbf{r_s I_s} \max(0, \vec{\mathbf{r}}_\mathbf{k} \cdot \vec{\mathbf{c}}_\mathbf{k})^\alpha$$


$\bar{\mathbf{p}}_\mathbf{k}$
$\vec{\mathbf{c}}_\mathbf{k} = -\vec{\mathbf{d}}_\mathbf{i,j}$
$\vec{\mathbf{n}}_\mathbf{k}$
$\vec{\mathbf{s}}_\mathbf{k}$
$\vec{\mathbf{r}}_\mathbf{k} = 2(\vec{\mathbf{s}}_\mathbf{k} \cdot \vec{\mathbf{n}}_\mathbf{k})\vec{\mathbf{n}}_\mathbf{k} - \vec{\mathbf{s}}_\mathbf{k}$

Conventional (Whitted) Ray Tracing

- Cast a ray and find
  - "hit point" and normal at the "hit point"
- Compute **local lighting** at the "hit point" (Phong)
- Compute **global specular lighting**, assuming perfect mirror

$$\mathbf{E_k} = \mathbf{r_d I_d} \max(0, \vec{\mathbf{s}}_\mathbf{k} \cdot \vec{\mathbf{n}}_\mathbf{k}) + \mathbf{r_a I_a} + \mathbf{r_s I_s} \max(0, \vec{\mathbf{r}}_\mathbf{k} \cdot \vec{\mathbf{c}}_\mathbf{k})^\alpha + \boxed{\mathbf{r_g I_{spec}}}$$

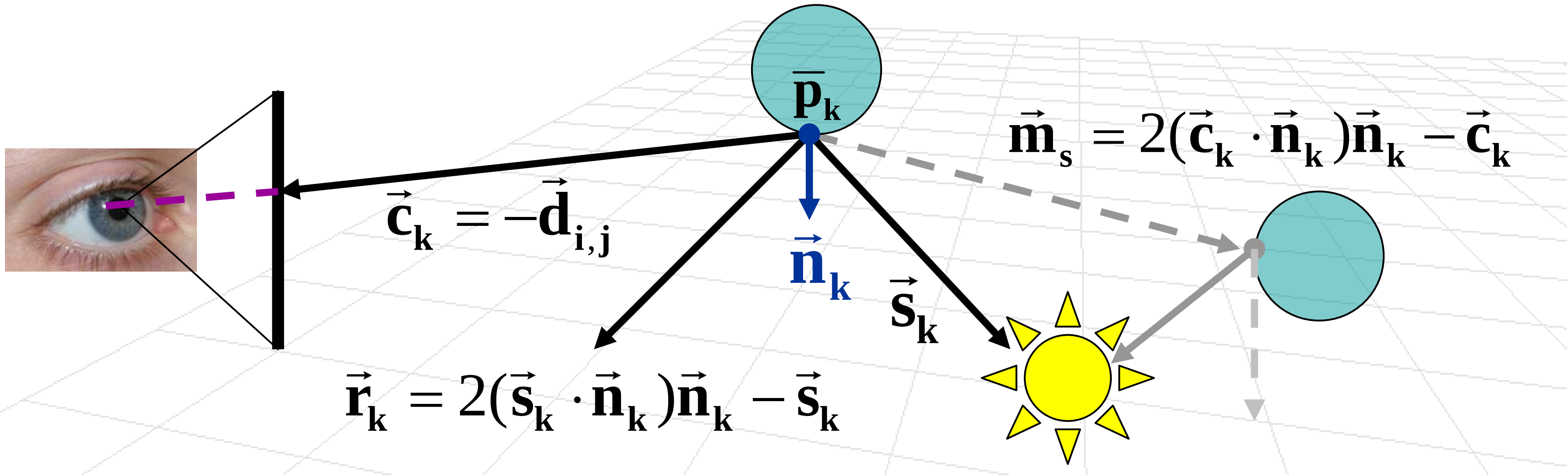


$\bar{\mathbf{p}}_\mathbf{k}$
$\vec{\mathbf{m}}_\mathbf{s} = 2(\vec{\mathbf{c}}_\mathbf{k} \cdot \vec{\mathbf{n}}_\mathbf{k})\vec{\mathbf{n}}_\mathbf{k} - \vec{\mathbf{c}}_\mathbf{k}$
$\vec{\mathbf{c}}_\mathbf{k} = -\vec{\mathbf{d}}_\mathbf{i,j}$
$\vec{\mathbf{n}}_\mathbf{k}$
$\vec{\mathbf{s}}_\mathbf{k}$
$\vec{\mathbf{r}}_\mathbf{k} = 2(\vec{\mathbf{s}}_\mathbf{k} \cdot \vec{\mathbf{n}}_\mathbf{k})\vec{\mathbf{n}}_\mathbf{k} - \vec{\mathbf{s}}_\mathbf{k}$

Conventional (Whitted) Ray Tracing

- Cast a ray and find
  - “hit point” and normal at the “hit point”
- Compute **local lighting** at the “hit point” (Phong)
- Compute **global specular lighting**, assuming perfect mirror

$$\mathbf{E_k} = \mathbf{r_d I_d} \max(0, \vec{\mathbf{s}}_\mathbf{k} \cdot \vec{\mathbf{n}}_\mathbf{k}) + \mathbf{r_a I_a} + \mathbf{r_s I_s} \max(0, \vec{\mathbf{r}}_\mathbf{k} \cdot \vec{\mathbf{c}}_\mathbf{k})^\alpha + \boxed{\mathbf{r_g I_{spec}}}$$



$\bar{\mathbf{p}}_\mathbf{k}$
$\vec{\mathbf{m}}_\mathbf{s} = 2(\vec{\mathbf{c}}_\mathbf{k} \cdot \vec{\mathbf{n}}_\mathbf{k})\vec{\mathbf{n}}_\mathbf{k} - \vec{\mathbf{c}}_\mathbf{k}$
$\vec{\mathbf{c}}_\mathbf{k} = -\vec{\mathbf{d}}_\mathbf{i,j}$
$\vec{\mathbf{n}}_\mathbf{k}$
$\vec{\mathbf{s}}_\mathbf{k}$
$\vec{\mathbf{r}}_\mathbf{k} = 2(\vec{\mathbf{s}}_\mathbf{k} \cdot \vec{\mathbf{n}}_\mathbf{k})\vec{\mathbf{n}}_\mathbf{k} - \vec{\mathbf{s}}_\mathbf{k}$

Texture

- Texture can be used to modulate diffuse and ambient reflection coefficients, as with Gouraud or Phong shading
- All we need, is a way of mapping a point on the surface (hit point) to a point in the texture space
  - e.g. given a hit point of parametric surface, we can convert the 3D point coordinates to surface parameters, and use them to get texture coordinates (as with standard texture mapping)
- Unlike with Gouraud or Phong shading models we don’t need to interpolate texture coordinates over polygons
- Anti-aliasing and super-sampling we will cover later (next week)