11. Screen-space Object Manipulation

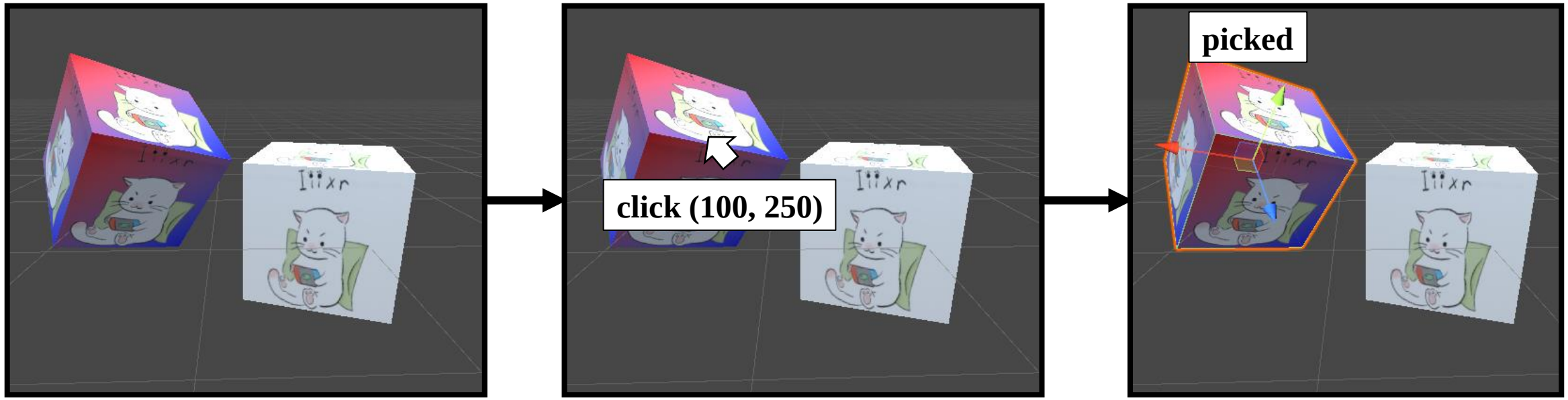
Prof. HyeongYeop Kang
siamiz@khu.ac.kr
YouTube: HKang IIIXR LAB
IIIXR LAB

Object Picking

Picking?

- On touchscreen, an object can be picked by tapping it with a finger. In PC screen, mouse clicking is popularly used for the same purpose.

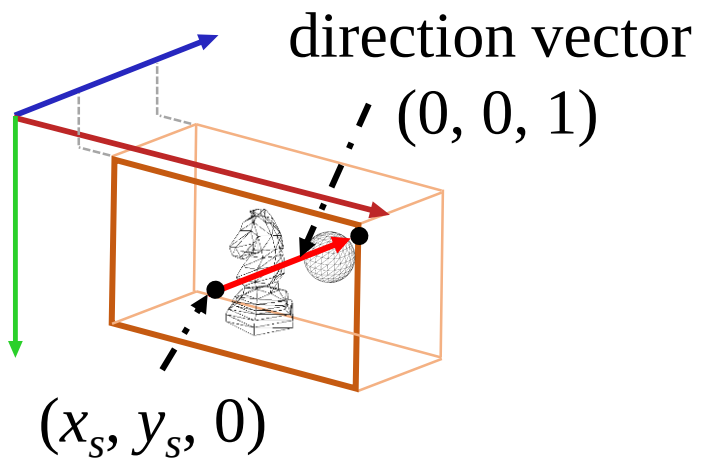
- A screen touch or mouse click simply returns the 2D pixel coordinates (x_s, y_s), and we need a procedure to identify which object is selected.



Object Picking

Ray-object intersection

- Given (x_s, y_s), we can consider a ray described by the start point $(x_s, y_s, 0)$ and the direction vector (0, 0, 1).



- We need ray- object intersection tests to find the object first hit by the ray, which is the piece in the example. However, it is not good to compute the intersections in the screen space because the screen-space information available to the user is not about the objects.

- Therefore, we need to transform the screen-space ray back to the object spaces and do the intersection tests in the object spaces.

Camera-space Ray

Ray in the camera-space

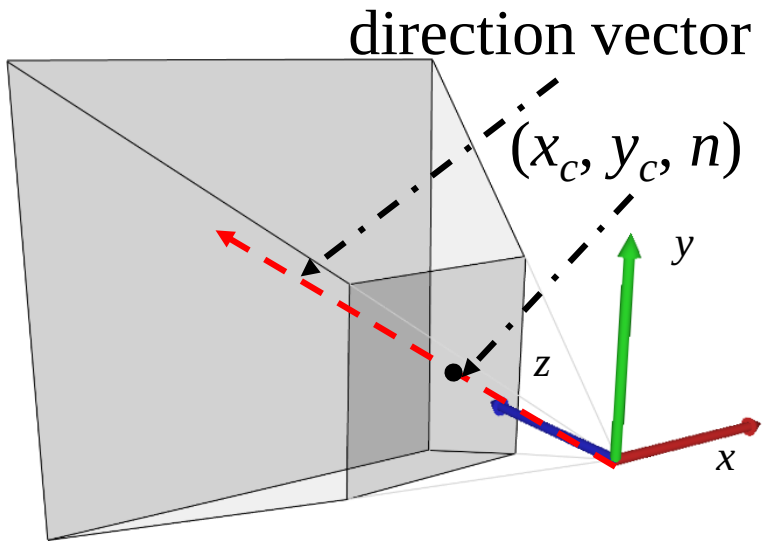
- The camera-space ray's start point is represented by (x_c, y_c, n). Note that the ray's start point is on the near plane, n.

- The projection matrix (we discussed before) can be applied to (x_c, y_c, n):

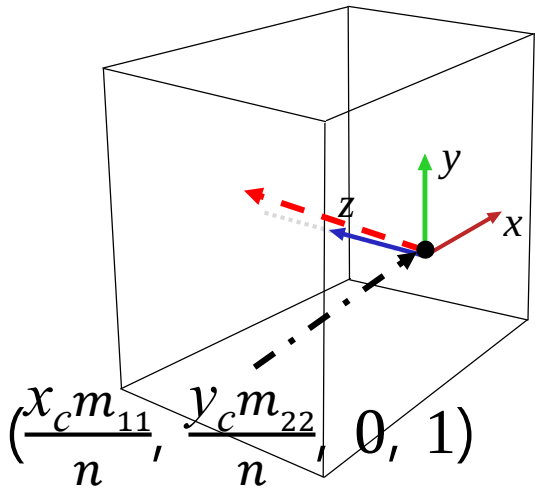
$$(m_{11} = \frac{cot\frac{fovy}{2}}{aspect}, m_{22} = \cot\frac{fovy}{2})$$

$$(x_c, y_c, n, 1)\begin{pmatrix} m_{11} & 0 & 0 & 0 \\ 0 & m_{22} & 0 & 0 \\ 0 & 0 & \frac{f}{f-n} & 1 \\ 0 & 0 & \frac{-fn}{f-n} & 0 \end{pmatrix}$$

$$= (x_c m_{11}, y_c m_{22}, 0, n) = (\frac{x_c m_{11}}{n}, \frac{y_c m_{22}}{n}, 0, 1)$$



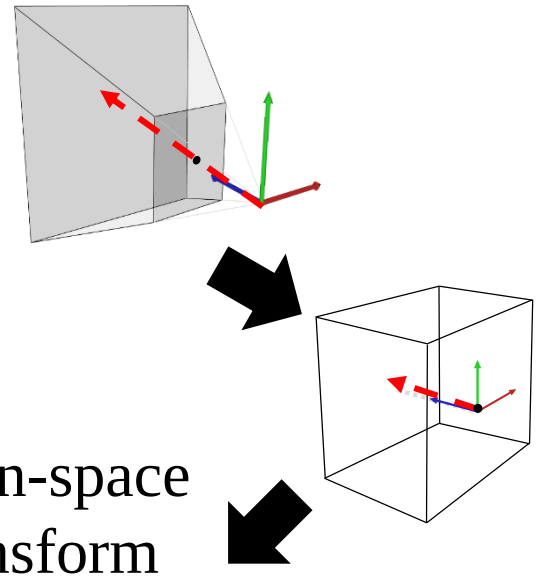
projection transform



Camera-space Ray

Ray in the camera-space

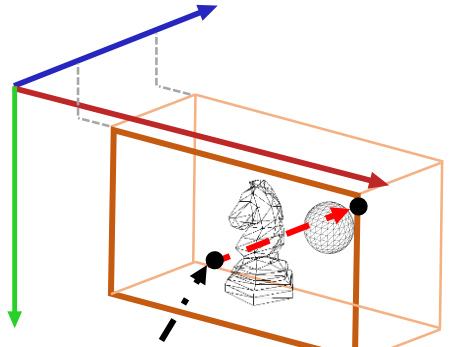
- The point is then transformed to the screen space by the viewport matrix:



screen-space transform

$$\left(\frac{x_c m_{11}}{n},\ \frac{y_c m_{22}}{n},\ 0,\ 1\right)\begin{pmatrix} \frac{w}{2} & 0 & 0 & 0 \\ 0 & -\frac{h}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ \frac{w}{2} & \frac{h}{2} & 0 & 1 \end{pmatrix}$$

$$= \left(\frac{w}{2}\left(\frac{x_c m_{11}}{n}+1\right),\ -\frac{h}{2}\left(\frac{y_c m_{22}}{n}-1\right),\ 0,\ 1\right)$$



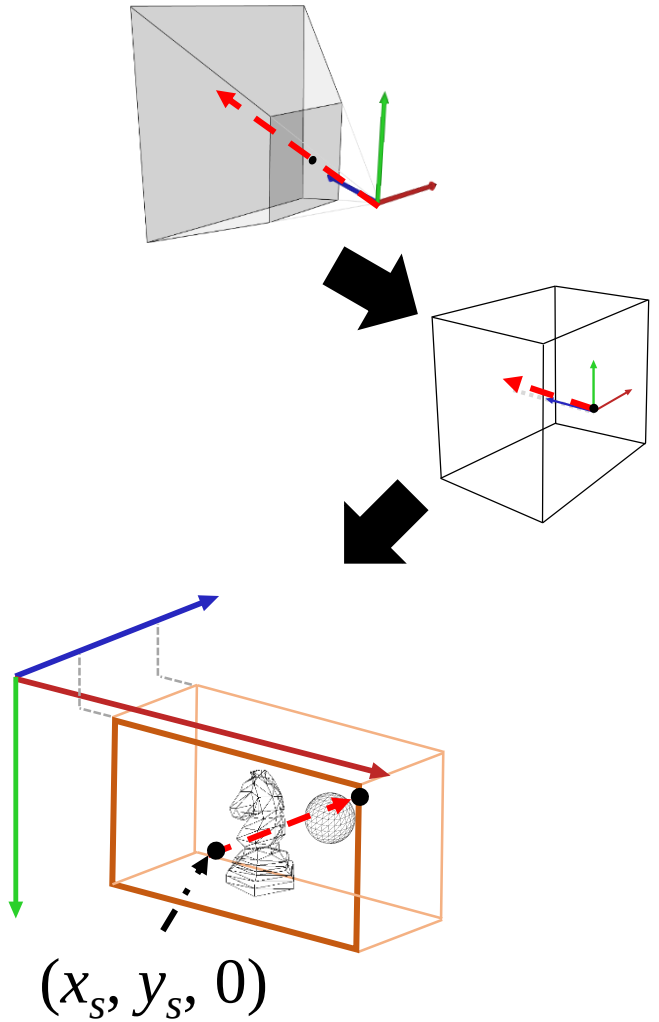
$(x_s, y_s, 0)$

Camera-space Ray

Ray in the camera-space

- The x- and y- coordinates of the screen-space ray should be identical to x_s and y_s, respectively.
- Therefore, we can obtain the start point of the camera-space ray.

$$\left(\frac{w}{2}\left(\frac{x_c m_{11}}{n}+1\right), -\frac{h}{2}\left(\frac{y_c m_{22}}{n}-1\right), 0, 1\right) = (x_s, y_s, 0, 1)$$



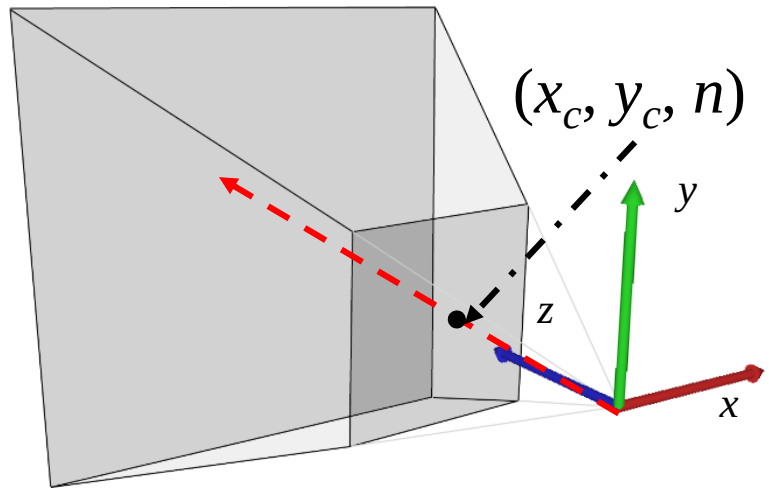
Camera-space Ray

Ray in the camera-space

- $(\frac{w}{2}(\frac{x_c m_{11}}{n}+1), -\frac{h}{2}(\frac{y_c m_{22}}{n}-1), 0, 1) = (x_s, y_s, 0, 1)$

  ➡ $x_c = \frac{n}{m_{11}}(\frac{2x_s}{w}-1), y_c = \frac{n}{m_{22}}(\frac{2y_s}{w}1)$

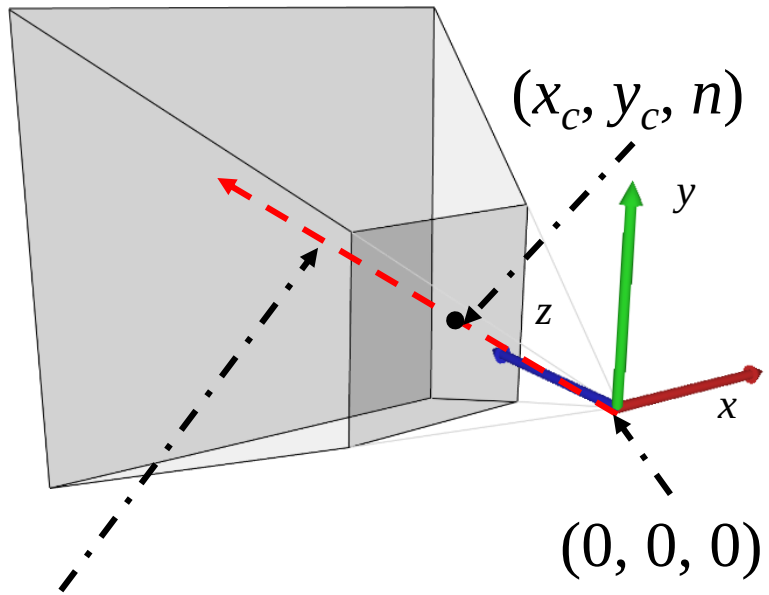
  ➡ $(x_c, y_c, n) = (\frac{n}{m_{11}}(\frac{2x_s}{w}-1), \frac{n}{m_{22}}(\frac{2y_s}{w}1), n, 1)$



Camera-space Ray

Ray in the camera-space

- In the camera-space, the direction vector can be obtained by connecting the origin and the start point (x_c, y_c):
  - $(\frac{n}{m_{11}}(\frac{2x_s}{w}-1), \frac{n}{m_{22}}(\frac{2y_s}{w}1), n, 1) - (0, 0, 0, 1)$
    $= (\frac{n}{m_{11}}(\frac{2x_s}{w}-1), \frac{n}{m_{22}}(\frac{2y_s}{w}1), n, 0)$
    $= (\frac{1}{m_{11}}(\frac{2x_s}{w}-1), \frac{1}{m_{22}}(\frac{2y_s}{w}1), 1, 0)$



direction vector (picking ray)

$(\frac{1}{m_{11}}(\frac{2x_s}{w}-1), \frac{1}{m_{22}}(\frac{2y_s}{w}1), 1, 0)$

World-space Ray

Camera-space ray to the object-space ray.

- The back-transform from the camera space to the world space is done by the inverse of the view transform.
  - The view transform M_{view} is a translation followed by a rotation, which we denote by TR. Then, its inverse, M_{view}^{-1}, is $R^{-1}T^{-1}$. T is a translation by –**EYE**, and therefore T^{-1} is a translation by **EYE**.
  - The inverse of a rotation is simply its transpose.

$$M_{view}^{-1} = R^{-1}T^{-1}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \mathbf{EYE}_x & \mathbf{EYE}_y & \mathbf{EYE}_z & 1 \end{pmatrix} \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} u_x & u_y & u_z & u \cdot \mathbf{EYE} \\ v_x & v_y & v_z & v \cdot \mathbf{EYE} \\ n_x & n_y & n_z & n \cdot \mathbf{EYE} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

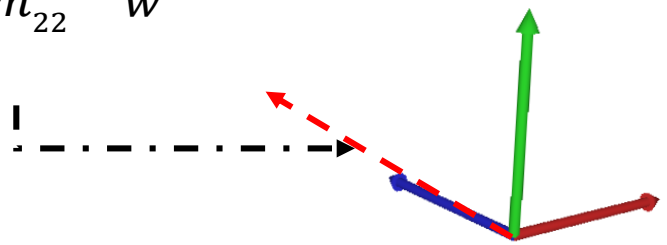
World-space Ray

Camera-space ray to the object-space ray.

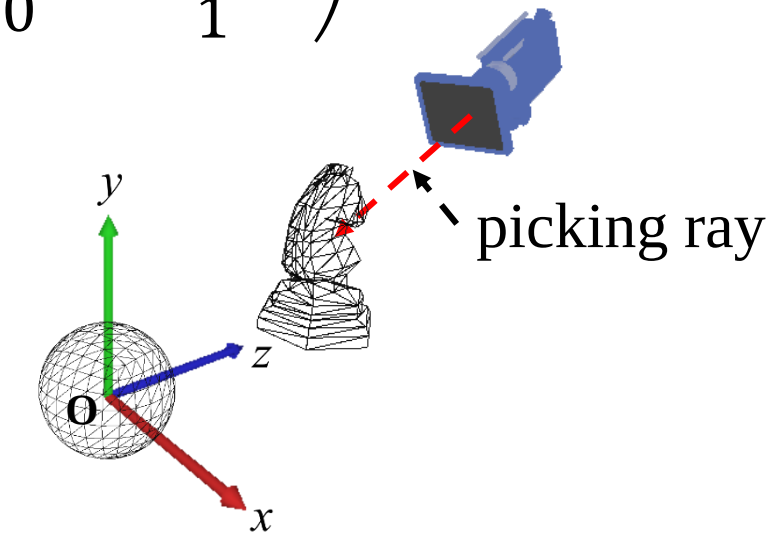
$$M_{view}^{-1} = \begin{pmatrix} u_x & u_y & u_z & u \cdot \mathbf{EYE} \\ v_x & v_y & v_z & v \cdot \mathbf{EYE} \\ n_x & n_y & n_z & n \cdot \mathbf{EYE} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

picking ray

$$\left(\frac{1}{m_{11}}\left(\frac{2x_s}{w} - 1\right), \frac{1}{m_{22}}\left(\frac{2y_s}{w} 1\right), 1, 0\right)$$



[Camera Space]



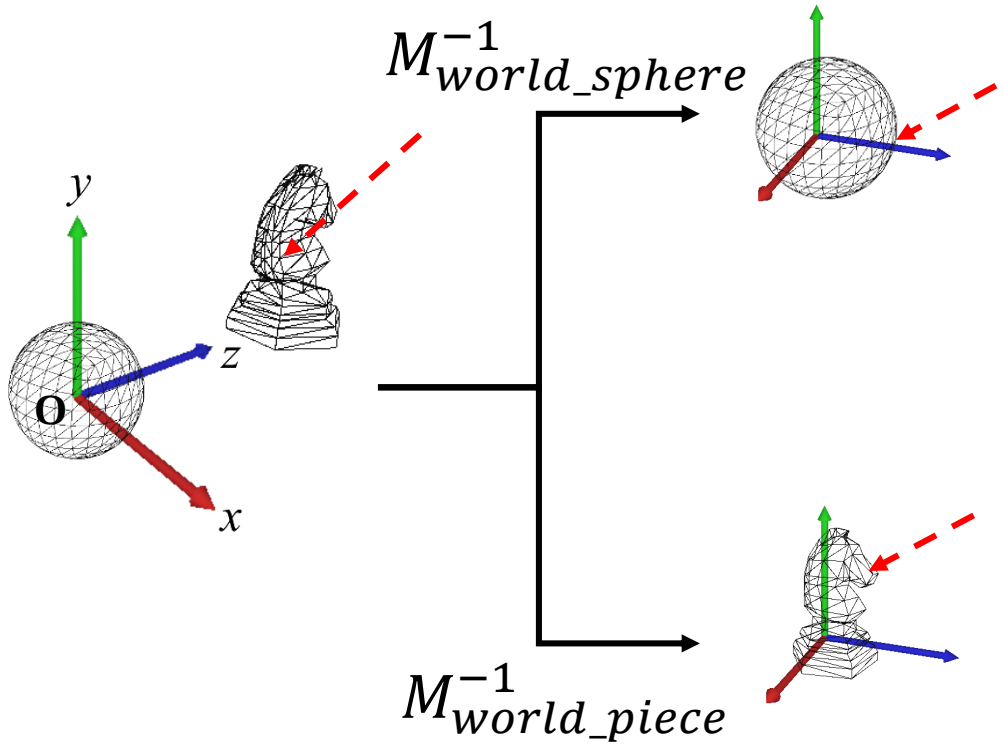
[World Space]

Object-space Ray

Intersection per object

- The piece and sphere have their own world transforms.

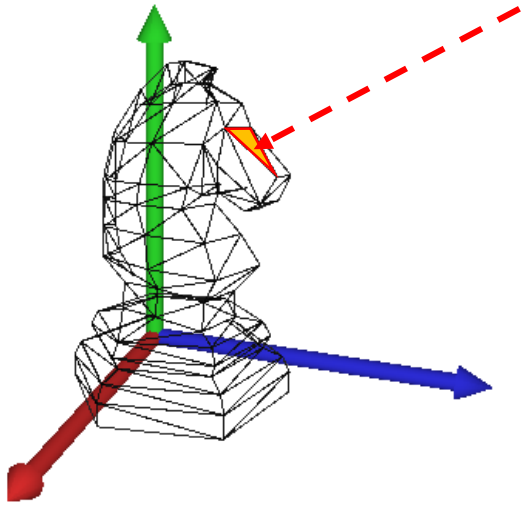
- Therefore, the coordinates of the start point (x, y) in the teapot's object space are different from those in the sphere's.



Object-space Ray

Intersection per object

- In order to determine if a ray hits an object represented in a polygon mesh, we have to perform a ray-triangle intersection test for every triangle of the object.

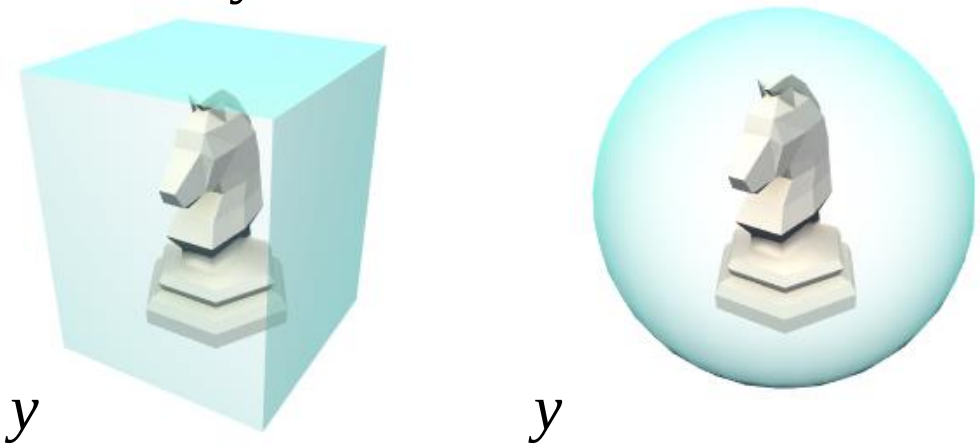


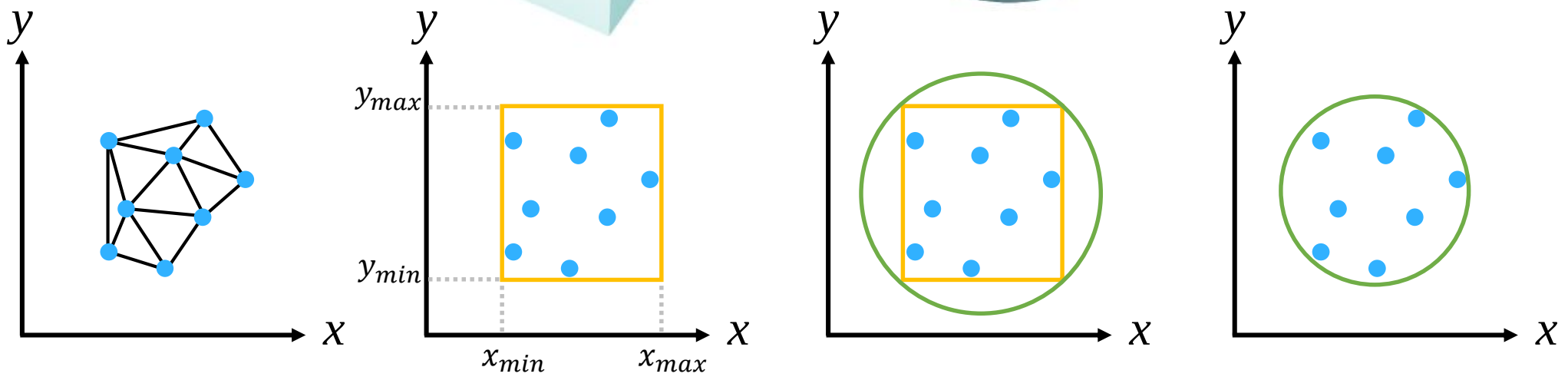
- Unfortunately, processing every triangle is costly. A faster but less accurate method is to approximate a polygon mesh with a *bounding volume* (BV).

Bounding Volumes

Bounding volume covers the polygon mesh

- There are two popular BVs: axis-aligned bounding box (AABB) and bounding sphere.
- An AABB is represented by the extents along the principal axes, (x, y, z).
- A bounding sphere is represented by its center and radius.

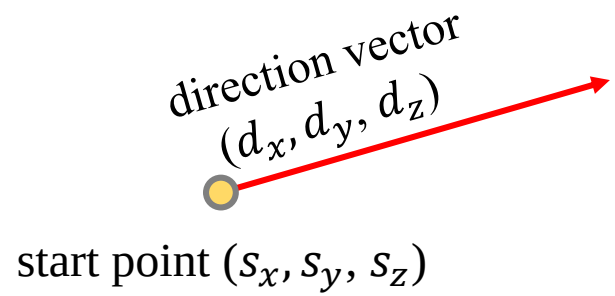




Ray-BV Intersection

Ray equation

- For the ray-sphere intersection test, we can represent the ray in three parametric equations.

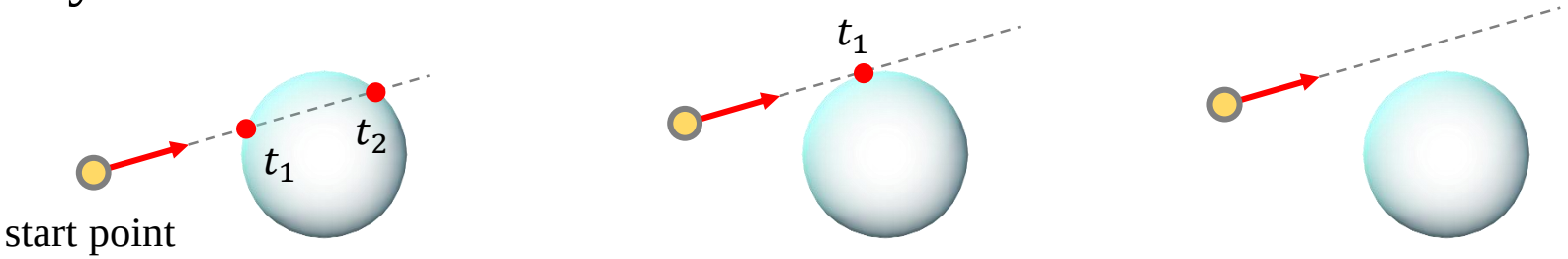


$$x(t) = s_x + td_x$$
$$y(t) = s_y + td_y$$
$$z(t) = s_z + td_z$$

- Insert $x(t)$, $y(t)$, and $z(t)$ into the bounding sphere equation.

$$(x - C_x)^2 + (y - C_y)^2 + (z - C_z)^2 = r^2 \Rightarrow at^2 + bt + c = 0 \Rightarrow t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

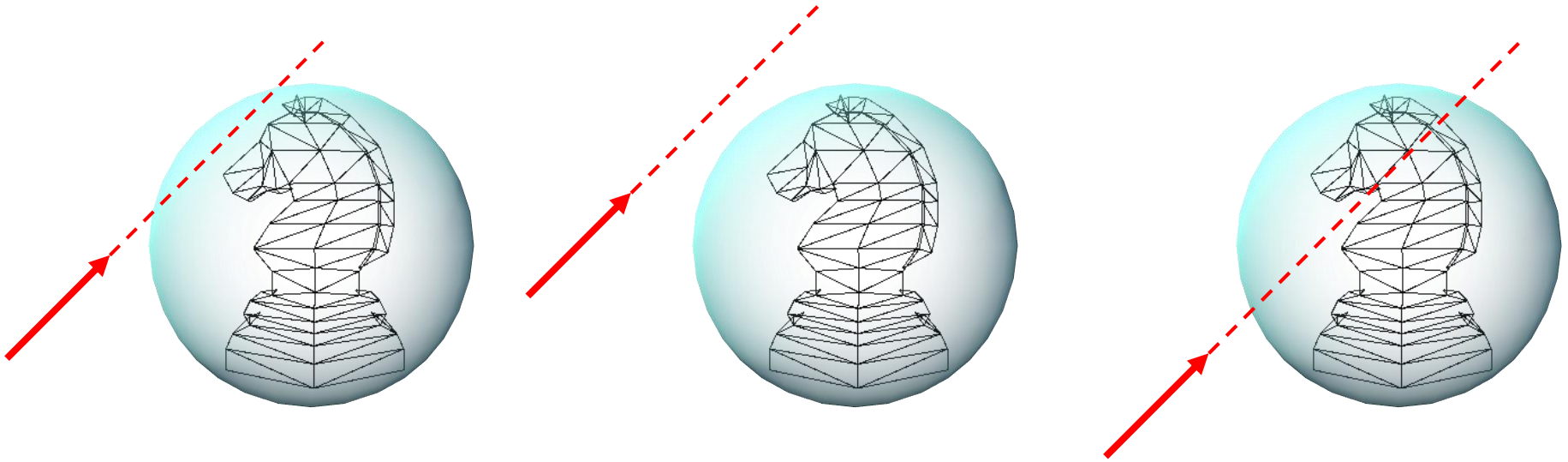
- The expression underneath the square root sign is the discriminant. Three cases determined by the discriminant.



Ray-BV Intersection

Ray equation

- Ray-sphere intersection test is often performed at the preprocessing step, and discards the polygon mesh that is guaranteed not to intersect the ray.

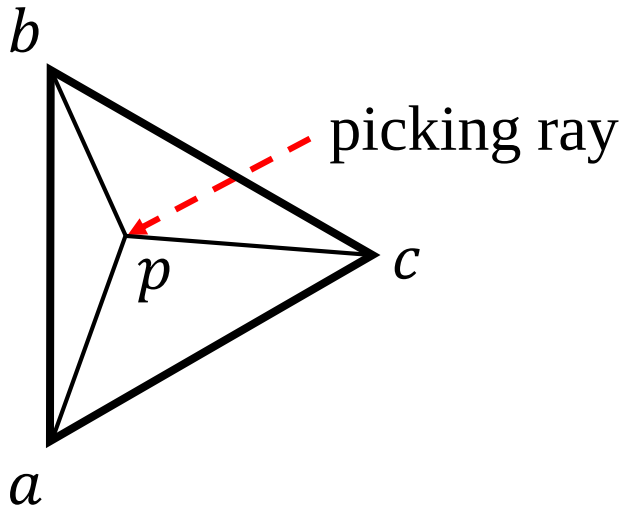


- If accurate results are required, we may resort to the ray-triangle intersection test or multiple bounding volume test.

Ray-Triangle Intersection

Intersection test with barycentric coordinates.

- When a triangle <*a*, *b*, *c*> is hit by a ray at *p*, the triangle is divided by *p* into three sub-triangles.
- Let *u* denote the ratio of the area of <*p*, *b*, *c*> to that of <*a*, *b*, *c*>. It roughly describes how close *p* is to *a*. The closer, the larger. It can be taken as the weight of *a* on *p*.

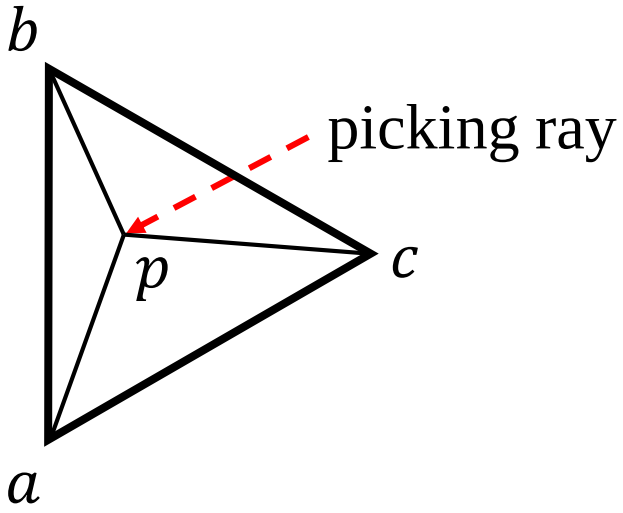


$$u = \frac{area(p, b, c)}{area(a, b, c)}$$

Ray-Triangle Intersection

Intersection test with barycentric coordinates.

- When v and w are similarly defined, p is defined as a weighted sum of the vertices: $p = ua + vb + wc$.
- The weights (u, v, w) are called the barycentric coordinates of p.
- Obviously, $u + v + w = 1$, and therefore w can be replaced by $(1 - u - v)$, i.e., $p = ua + vb + (1 - u - v)c$.

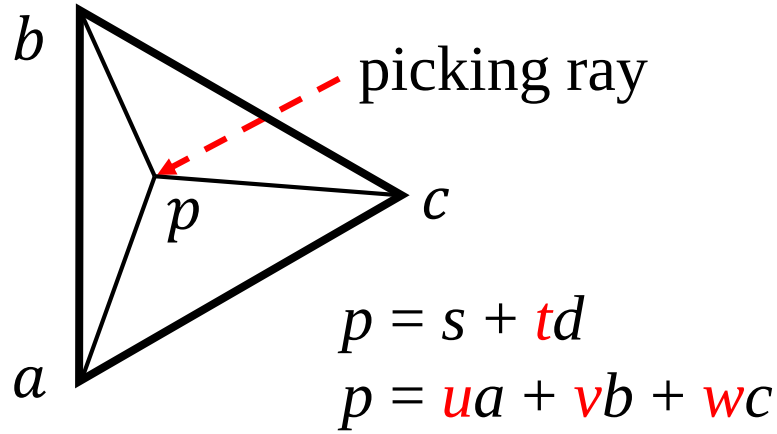


$$u = \frac{area(p,b,c)}{area(a,b,c)}, v = \frac{area(p,c,a)}{area(a,b,c)}, w = \frac{area(p,a,b)}{area(a,b,c)}$$

Ray-Triangle Intersection

Intersection test with barycentric coordinates.

- Compute the intersection between a ray, $s + td$, and a triangle, $\langle a, b, c \rangle$. Note that d, A, B and S all have 3D coordinates.
  - $s + td = ua + vb + wc$
  - $td + u(c-a) + v(c-b) = c - s$
  - $td + uA + vB = S$



- The solution to this linear system is obtained using Cramer's rule:

$$t = \frac{\begin{vmatrix} S_x & A_x & B_x \\ S_y & A_y & B_y \\ S_z & A_z & B_z \end{vmatrix}}{\begin{vmatrix} d_x & A_x & B_x \\ d_y & A_y & B_y \\ d_z & A_z & B_z \end{vmatrix}}, u = \frac{\begin{vmatrix} d_x & S_x & B_x \\ d_y & S_y & B_y \\ d_z & S_z & B_z \end{vmatrix}}{\begin{vmatrix} d_x & A_x & B_x \\ d_y & A_y & B_y \\ d_z & A_z & B_z \end{vmatrix}}, v = \frac{\begin{vmatrix} d_x & A_x & S_x \\ d_y & A_y & S_y \\ d_z & A_z & S_z \end{vmatrix}}{\begin{vmatrix} d_x & A_x & B_x \\ d_y & A_y & B_y \\ d_z & A_z & B_z \end{vmatrix}}$$

Cramer's rule

Cramer's rule: $Ax = B \rightarrow x = A^{-1}B$, $(A^{-1} = 1/\det(A) \times \mathrm{adj}(A))$

$$\begin{bmatrix} d_x & A_x & B_x \\ d_y & A_y & B_y \\ d_z & A_z & B_z \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = \begin{bmatrix} S_x \\ S_y \\ S_z \end{bmatrix}$$

$$t = \frac{\begin{vmatrix} S_x & A_x & B_x \\ S_y & A_y & B_y \\ S_z & A_z & B_z \end{vmatrix}}{\begin{vmatrix} d_x & A_x & B_x \\ d_y & A_y & B_y \\ d_z & A_z & B_z \end{vmatrix}}, u = \frac{\begin{vmatrix} d_x & S_x & B_x \\ d_y & S_y & B_y \\ d_z & S_z & B_z \end{vmatrix}}{\begin{vmatrix} d_x & A_x & B_x \\ d_y & A_y & B_y \\ d_z & A_z & B_z \end{vmatrix}}, v = \frac{\begin{vmatrix} d_x & A_x & S_x \\ d_y & A_y & S_y \\ d_z & A_z & S_z \end{vmatrix}}{\begin{vmatrix} d_x & A_x & B_x \\ d_y & A_y & B_y \\ d_z & A_z & B_z \end{vmatrix}}$$

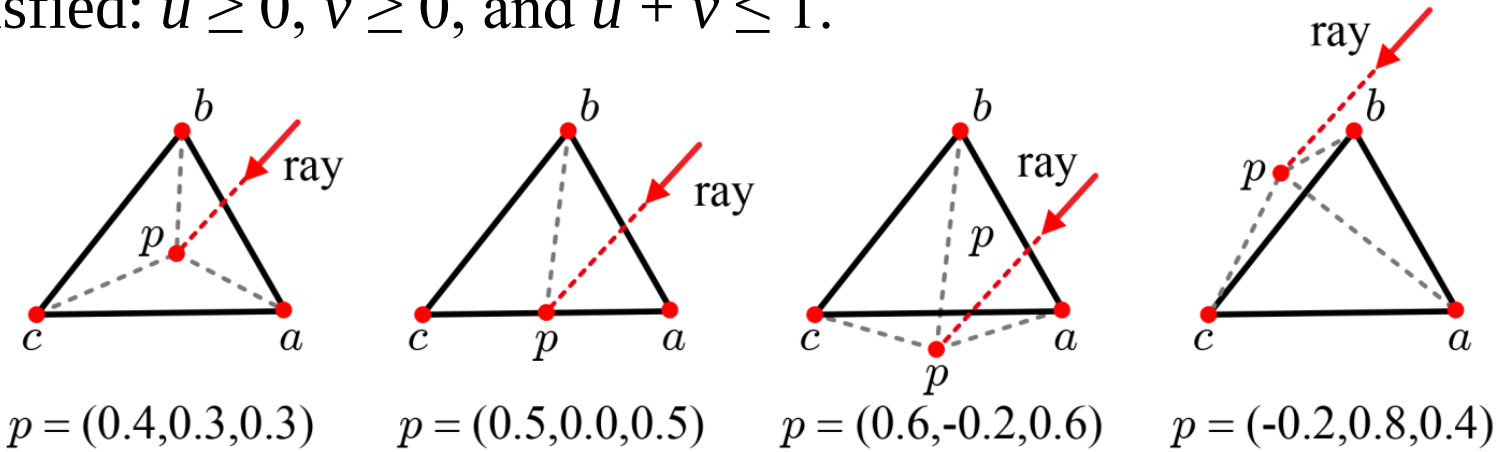
$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

$$\mathrm{adj}(\mathbf{A}) = \mathbf{C}^{\mathsf{T}} = \begin{pmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} \\ +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{pmatrix}.$$

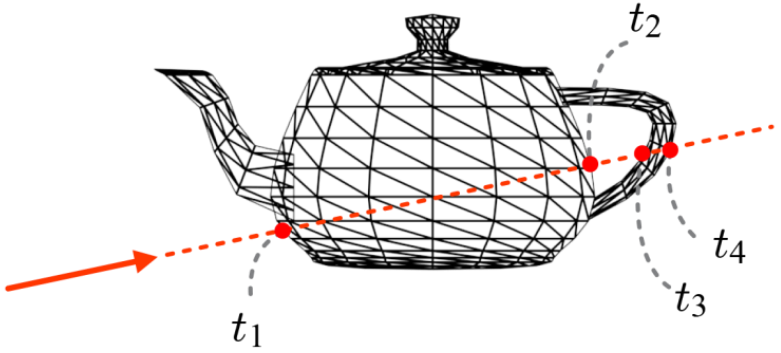
Ray-Triangle Intersection

Constraints

- In order for the intersection point to be confined to the triangle, the following condition should be satisfied: $u \geq 0$, $v \geq 0$, and $u + v \leq 1$.



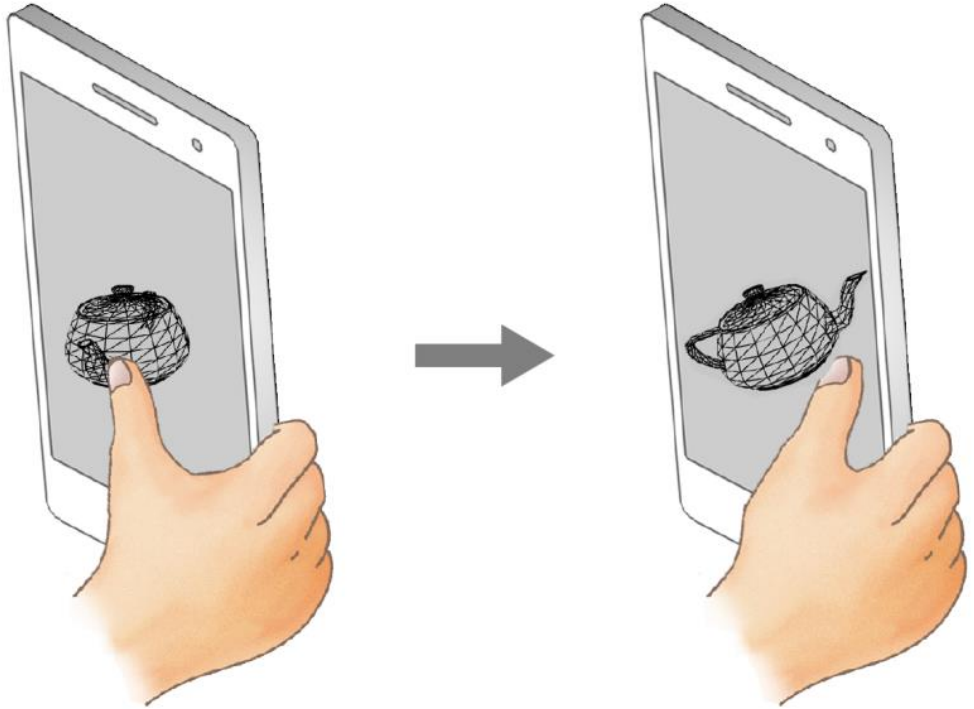
- When every triangle of a mesh is tested for intersection with the ray, multiple intersections can be found. Then, choose the point with the smallest positive t.

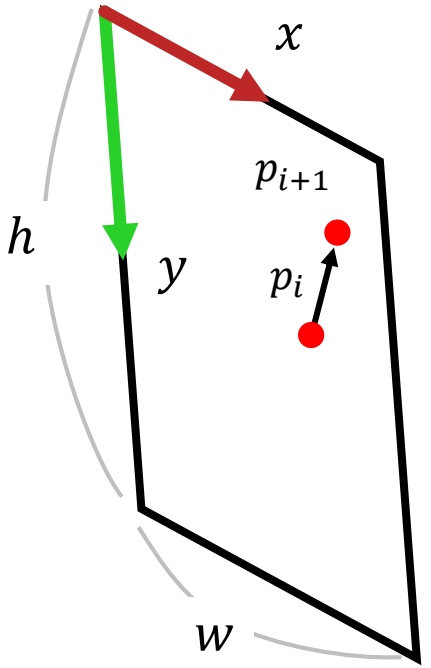


Rotating an object

Picking an abject and rotating it.

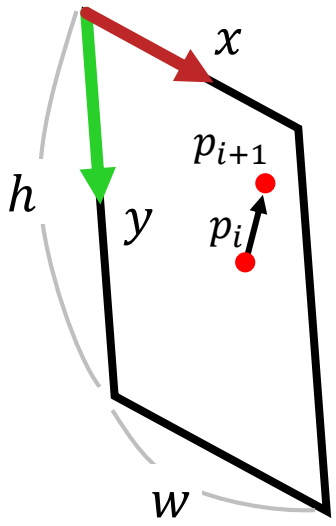
- On the 2D screen, the sliding finger's positions, $\{p_1, p_2, \ldots, p_n\}$, are tracked.
- For each pair of successive finger positions, p_i and p_{i+1}, a 3D rotation is computed and applied to the object.





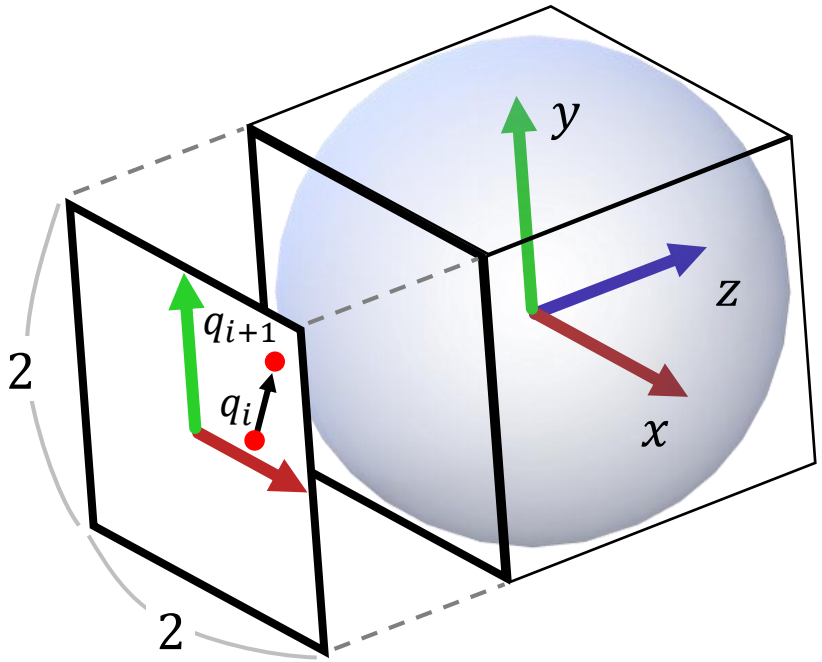
Arcball

- The arcball is a virtual ball located behind the screen and encloses the object to rotate.
  - The sliding finger rotates the arcball and the same rotation applies to the object.
  - For efficient implementation, the 2D screen with dimensions $w \times h$ is normalized to the 2×2 square, and the 2D coordinates, (x, y), of the finger's position are accordingly normalized, i.e., p_i and p_{i+1} are converted to q_i and q_{i+1}, respectively, in the square.



$$x' = \frac{2x}{w} - 1$$

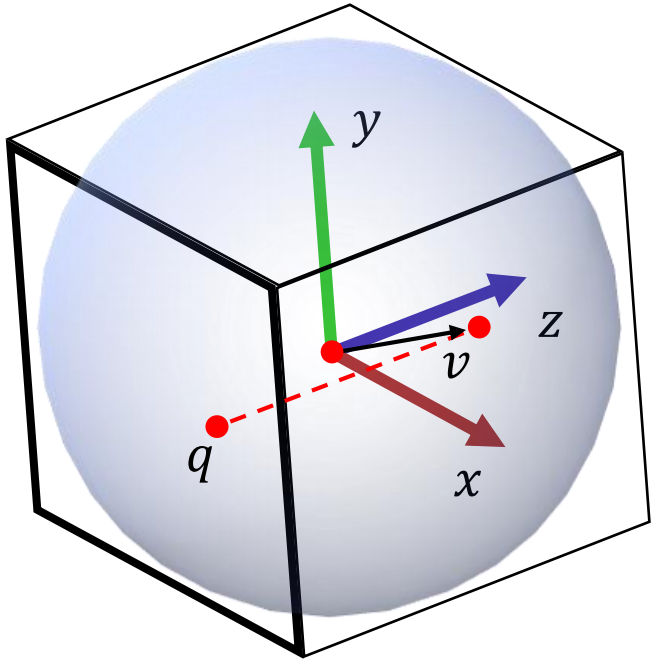
$$y' = -\frac{2y}{h} + 1$$

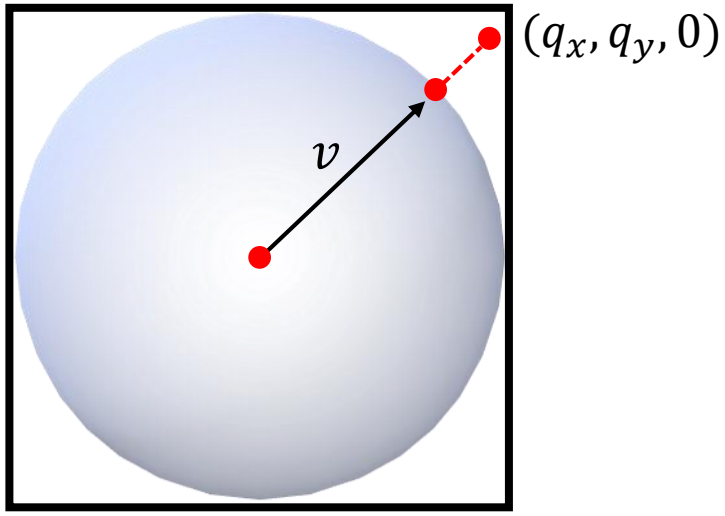


- Note that the arcball is a unit sphere, the radius of which is one.

Arcball

- Then, q is orthographically projected along z onto the surface of the arcball.
  - Let v denote the vector connecting the arcball's center and the projected point.
  - Note that $v_x = q_x$, $v_y = q_y$, and $v_z = \sqrt{1 - v_x^2 - v_y^2}$, since arcball is a unit sphere.
- If the projection is out of the arcball, i.e., if $q_x^2 + q_y^2 > 1$, v is normalized.

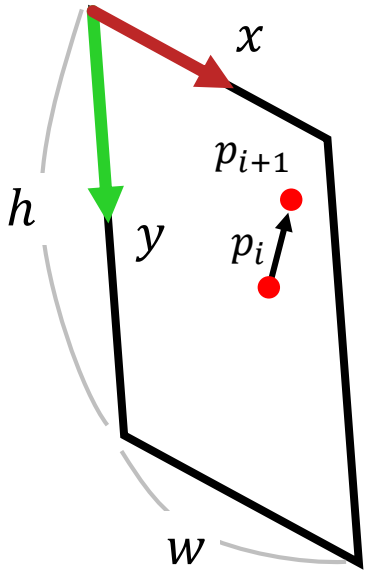




$q_x^2 + q_y^2 > 1$

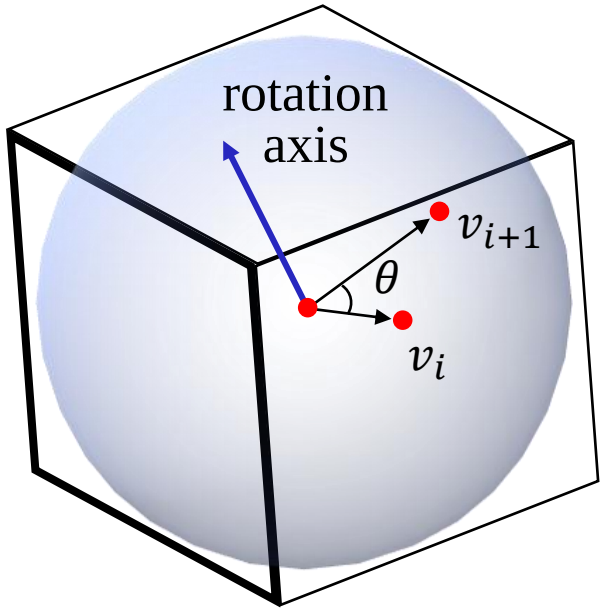
Arcball

- Given v_i and v_{i+1}, we can compute the rotation axis and angle.
  - The rotation axis is obtained by taking the cross product of v_i and v_{i+1}.
  - The rotation angle is obtained from the dot product of v_i and v_{i+1}.



$$x' = \frac{2x}{w} - 1$$

$$y' = -\frac{2y}{h} + 1$$



$$v_i \cdot v_{i+1} = \|v_i\| \, \|v_{i+1}\| cos\theta = cos\theta$$

$$\rightarrow \ \theta = \arccos(v_i \cdot v_{i+1})$$

Object-space Rotation Axis

- The rotation axis should be redefined in the object space so that the rotation is made prior to the original world transform.
  - We computed the rotation axis in the arcball's space.
  - It is a temporarily devised coordinate system, which we have not encountered in the rendering pipeline.
- Our strategy is then to take the rotation axis as is into the camera space of the rendering pipeline.
  - The rotation axis, as a vector, can be thought of as passing through the object in the camera space, and such a configuration is preserved in the object space.