Vector, Matrix, Trigonometric Functions, and Intersections

Game Mathematics

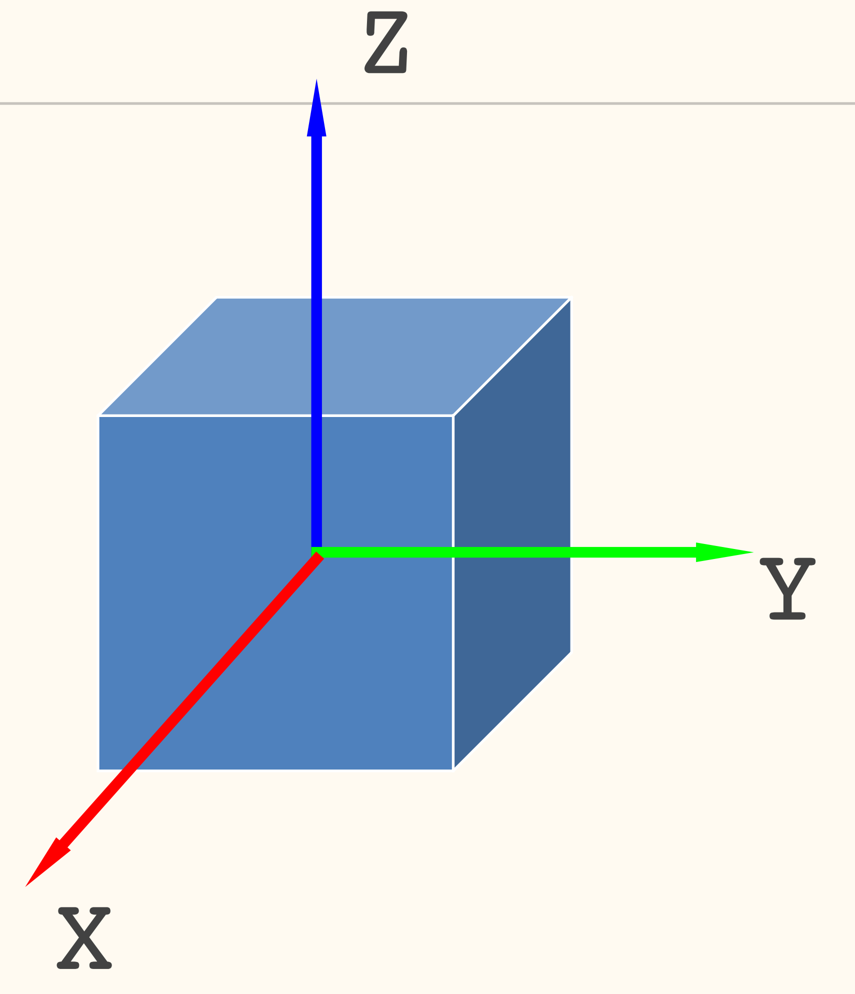
王銓彰
墨匠科技 BlackSmith CEO
cwang001@mac.com
cwang001@blacksmis.com

Content

- Coordinate Systems (座標系統)
- Trigonometric Functions (三角函數)
- Vectors (向量)
- Matrix and Transformations (矩陣與空間轉換)
- Intersections (解交點)
- Math about Rotations (旋轉)

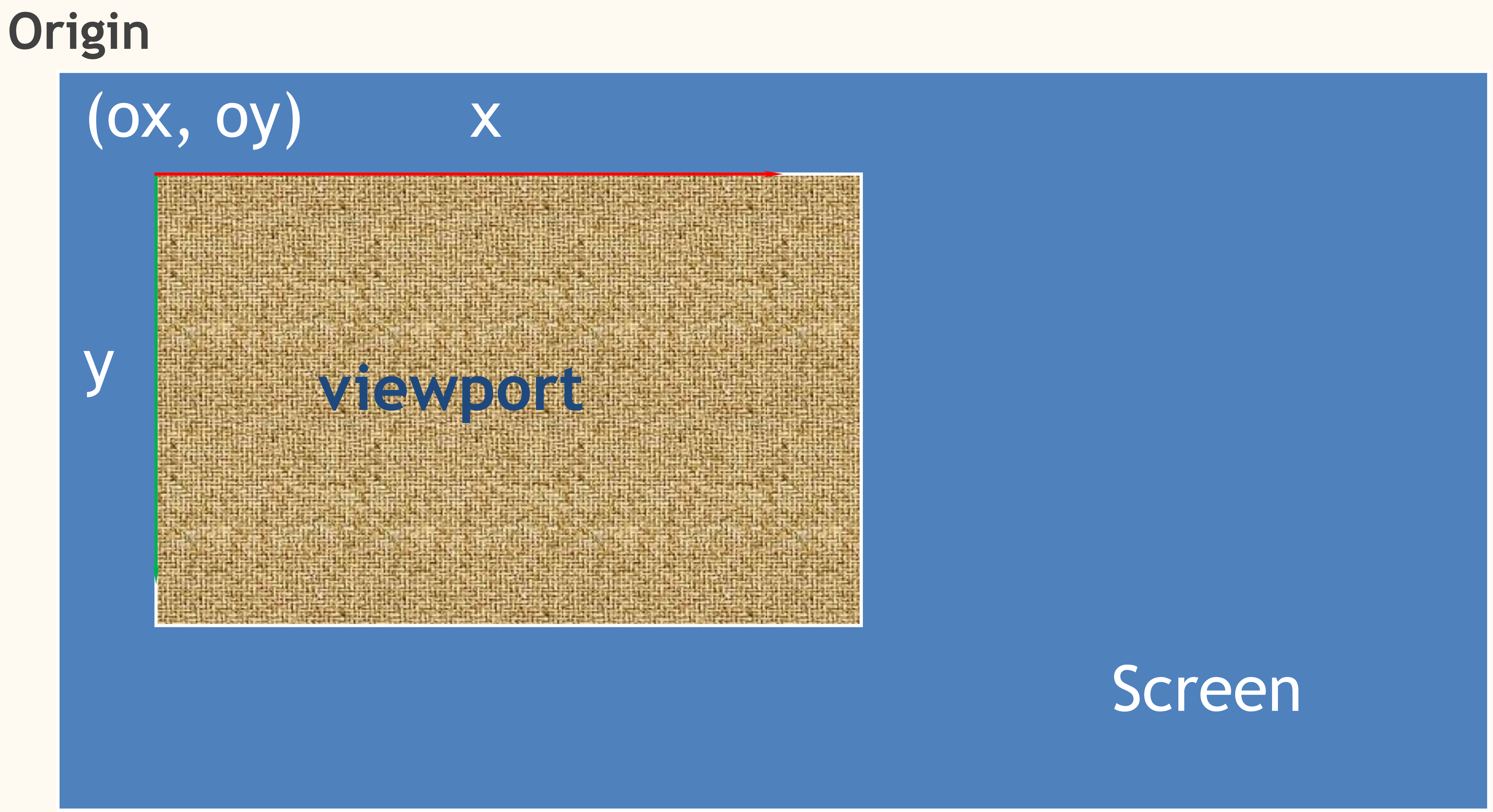
3D座標系統

- 直角坐標系
  - 笛卡兒座標系統(Cartesian coordinate system)
  - 通常使用右手定則，但非絕對
    - Unity使用左手系統
  - Z-up or Y-up
    - 以往學校或3D軟體習慣以Z軸為朝上的座標軸，近來流行使用Y軸
    - Unity uses Y-up.
    - 3ds Max uses Z-up.



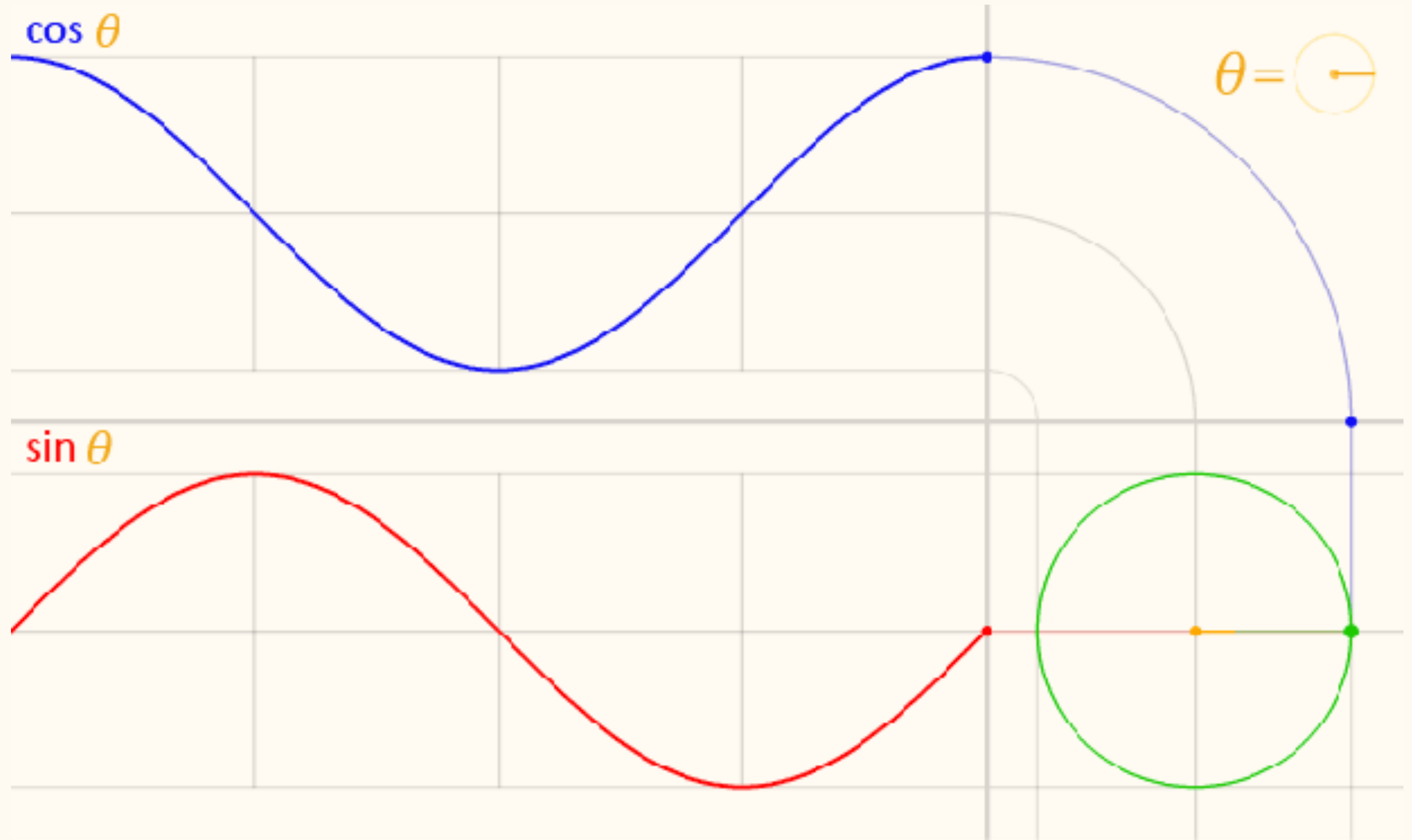
2D座標系統

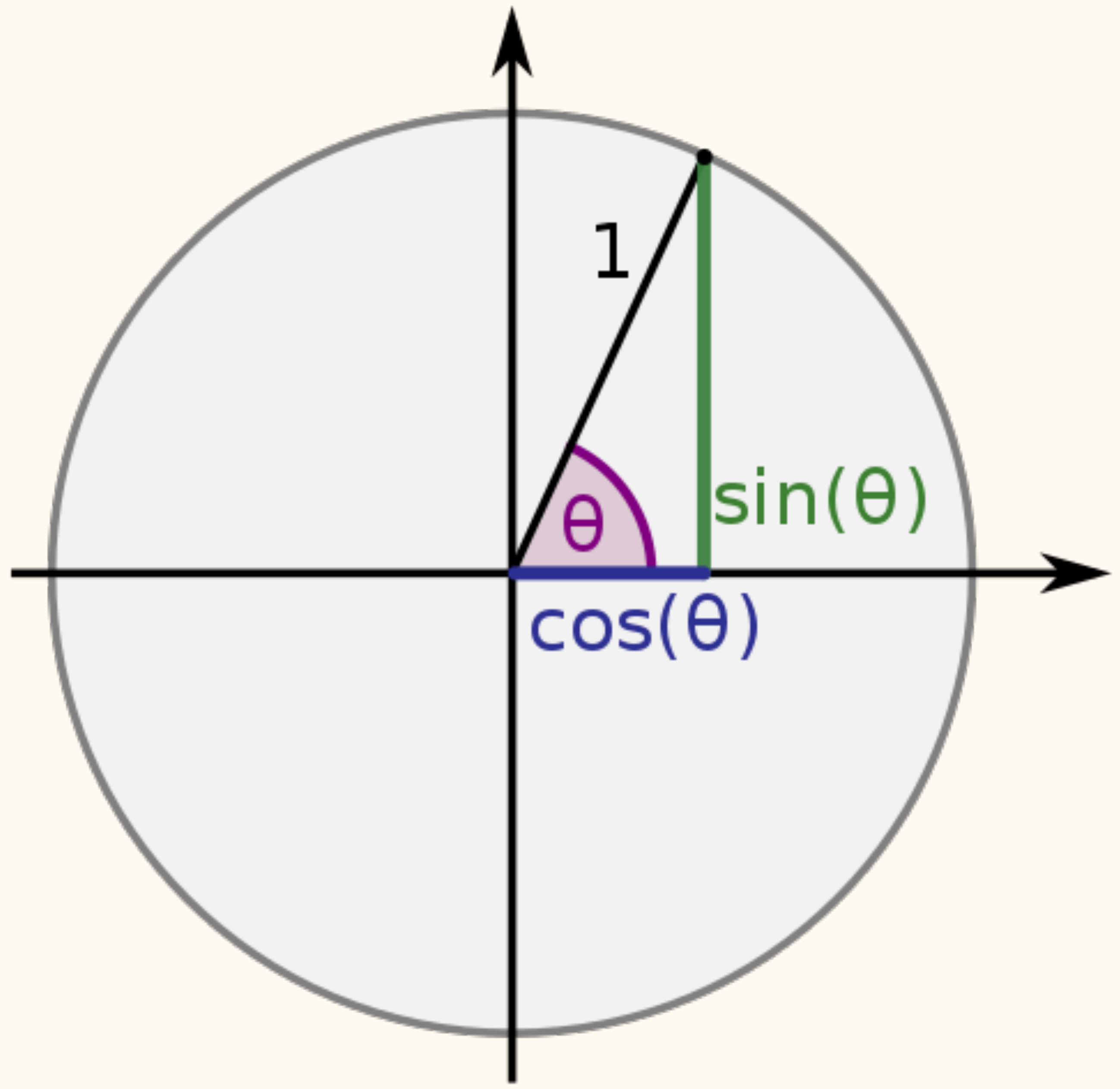
- For 2D, 我們習慣使用左手系統
  - Windows Desktop (Screen)
  - Viewport for rendering



Trigonometric Functions (三角函數)

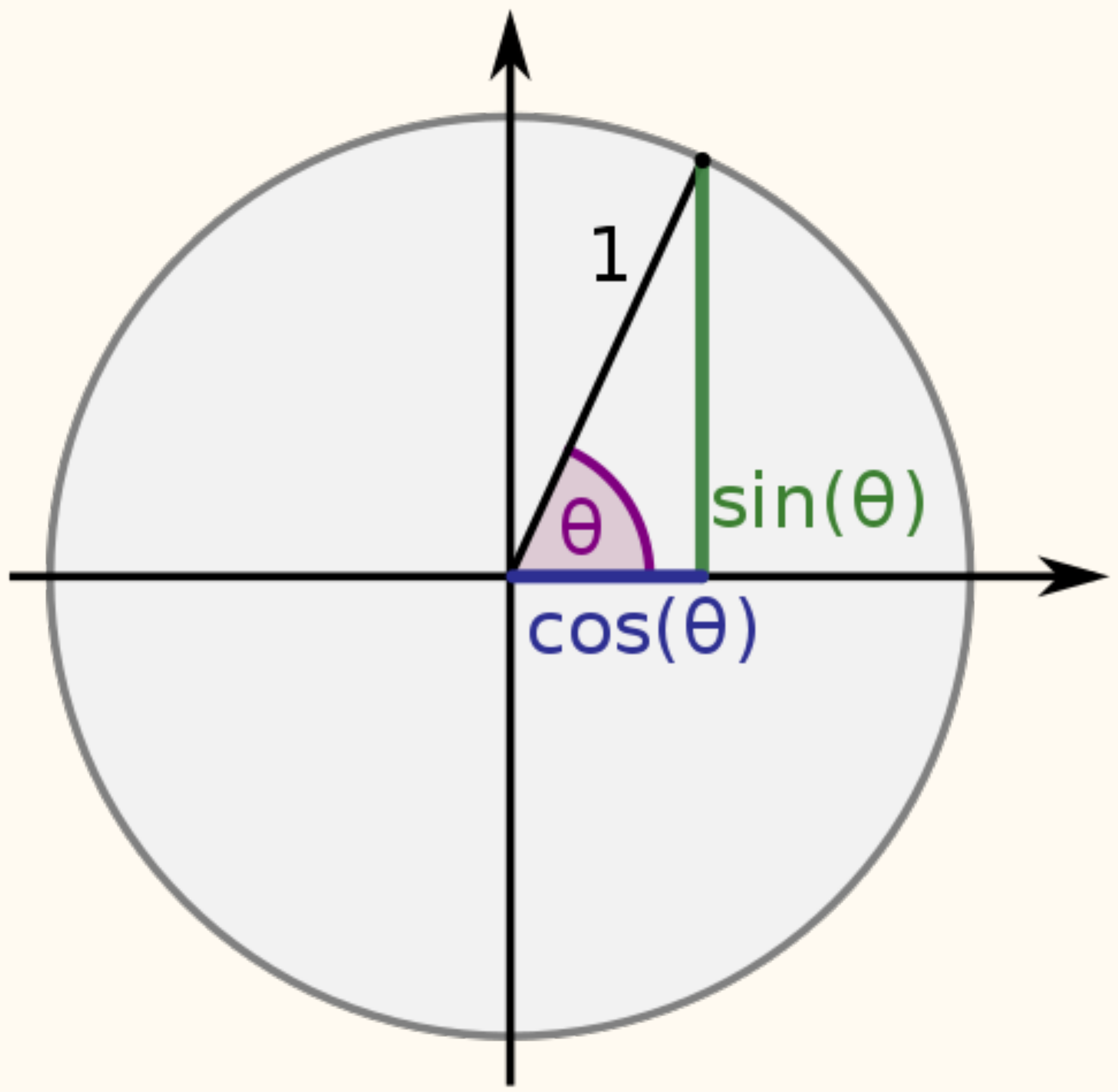
- Definition:
  - A standard unit circle (a circle with radius 1 unit)
  - A triangle is formed by a ray starting at the origin and making some angle with the x-axis
  - sin(θ) is the y-component of the triangle
  - cos(θ) is the x-component of the triangle





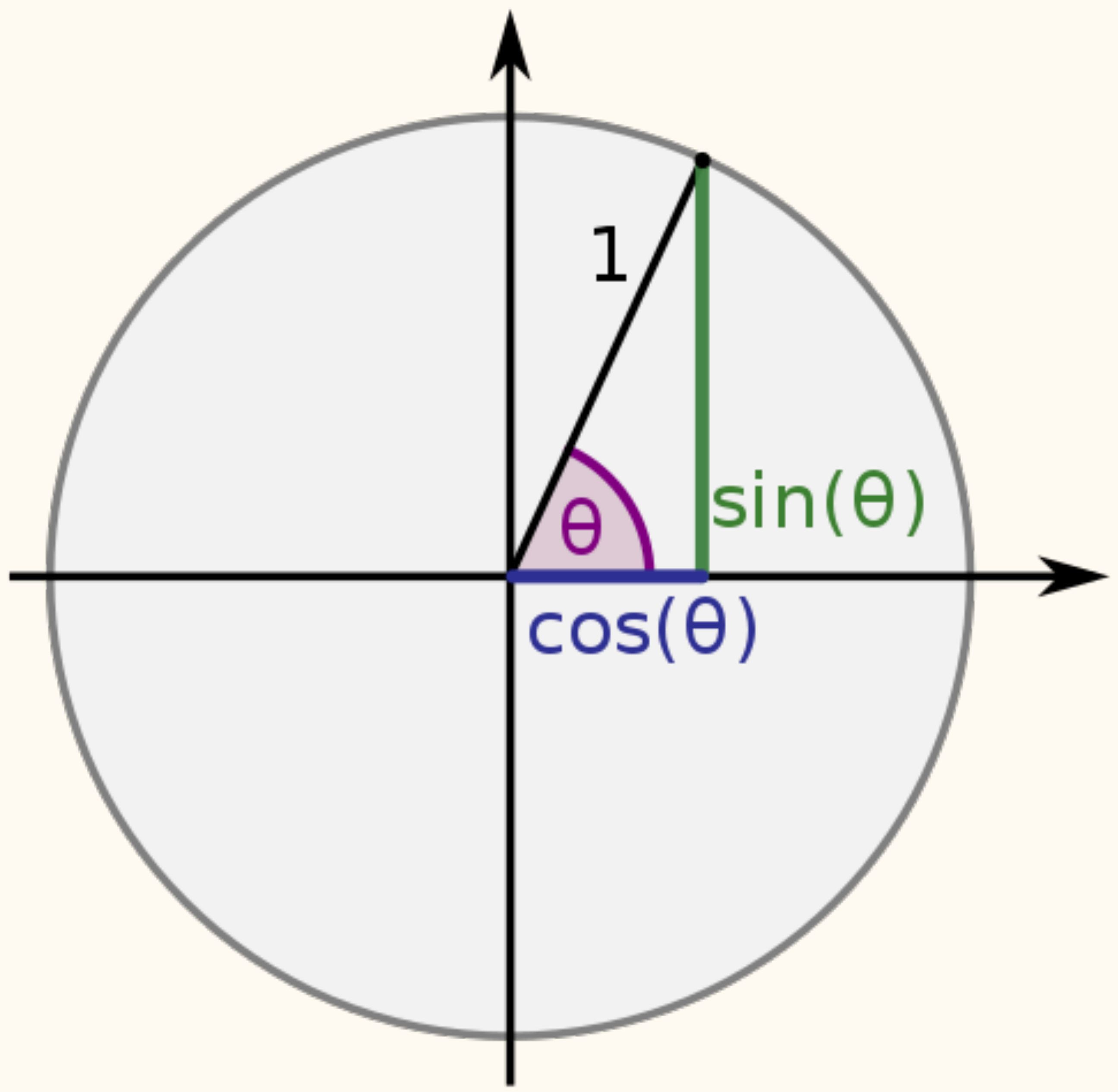
Trigonometric Functions (三角函數)

- Six Trigonometric functions
  - Sine function, $\sin\theta$
  - Cosine function, $\cos\theta$
  - Tangent function, $\tan\theta = \sin\theta / \cos\theta$
    - Slope of the triangle
  - Cotangent function, $\cot\theta = 1/\tan\theta$
  - Secant function, $\sec\theta = 1/\cos\theta$
  - Cosecant function, $\csc\theta = 1/\sin\theta$
- Only one function you need to learn in detail : $\sin\theta$
  - $\cos\theta = \sin(\theta + \pi/2)$



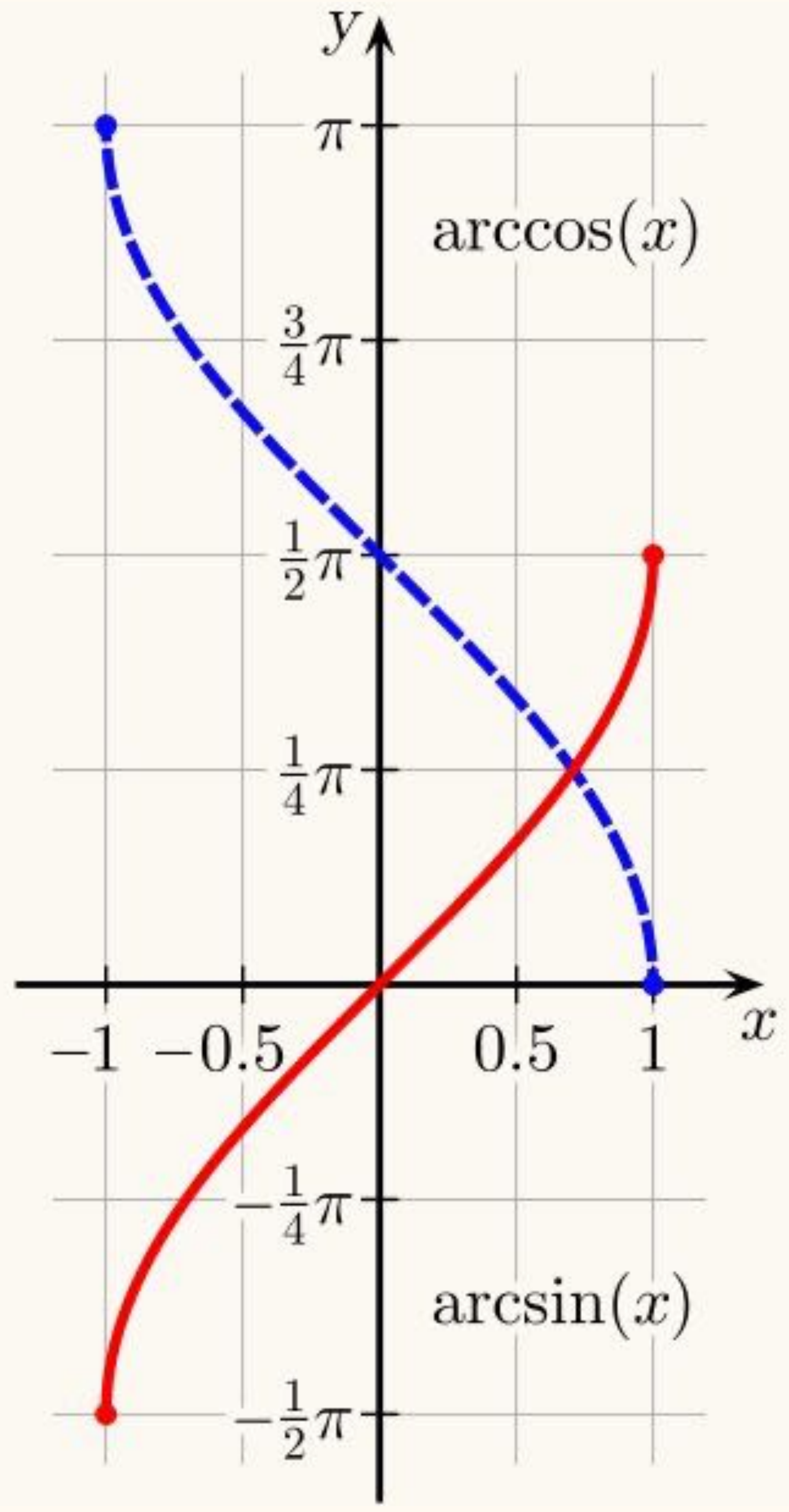
Trigonometric Functions (三角函數)

- Useful formula :
  - $\sin^2\theta + \cos^2\theta = 1$
  - $\sin 2\theta = 2\sin\theta\cos\theta$
  - $\cos 2\theta = \cos^2\theta - \sin^2\theta$
  - $\tan 2\theta = 2\tan\theta/(1 - \tan^2\theta)$



Inverse Trigonometry (反三角函數)

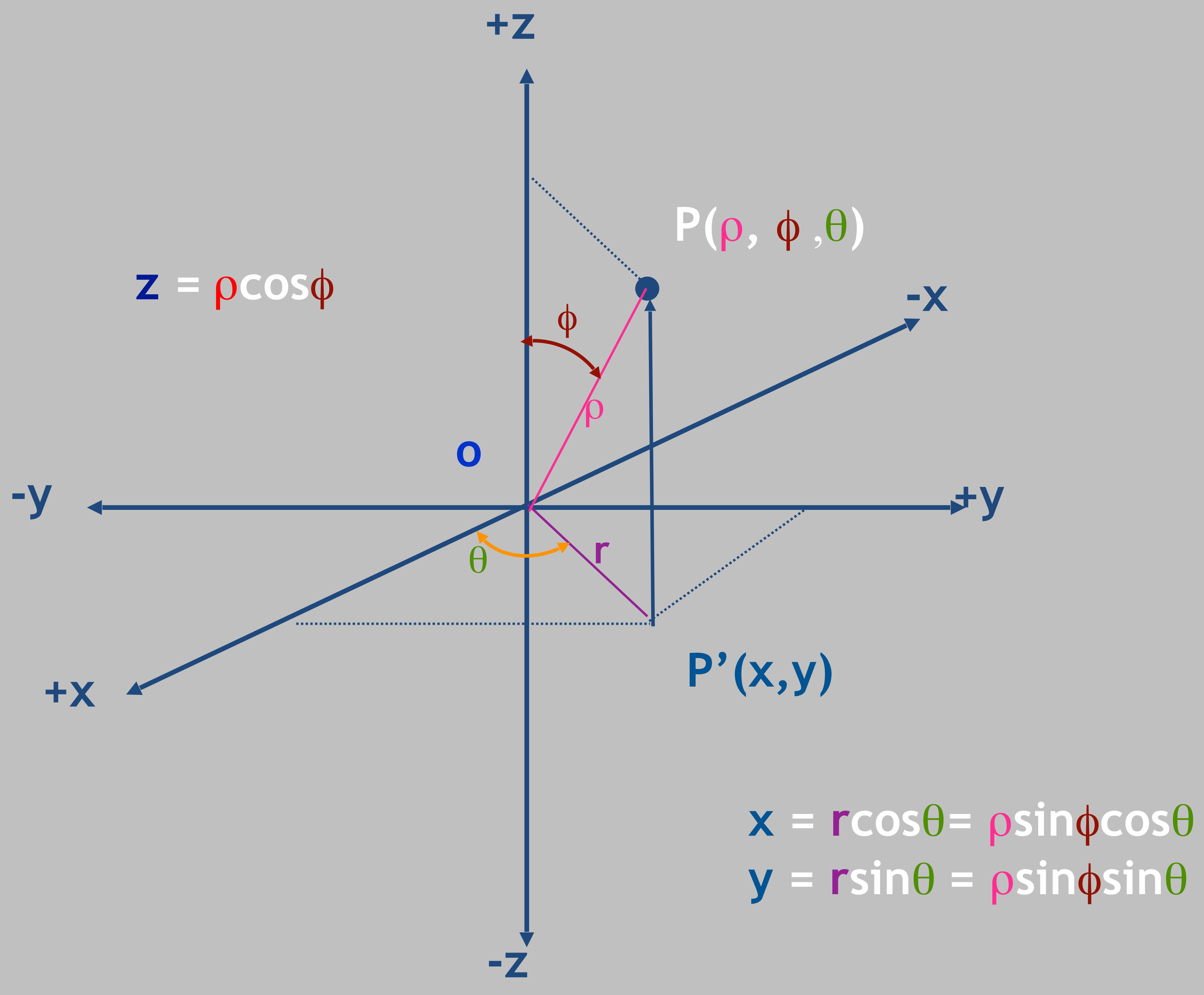
- $\sin^{-1}(\theta)$
  - arcsine function
- $\cos^{-1}(\theta)$
- $\tan^{-1}(\theta)$
- $\cot^{-1}(\theta)$
- $\sec^{-1}(\theta)$
- $\csc^{-1}(\theta)$



- Be careful to use inverse trigonometric functions :
  - Numerical error in floating-point computing

Spherical Coordinate System

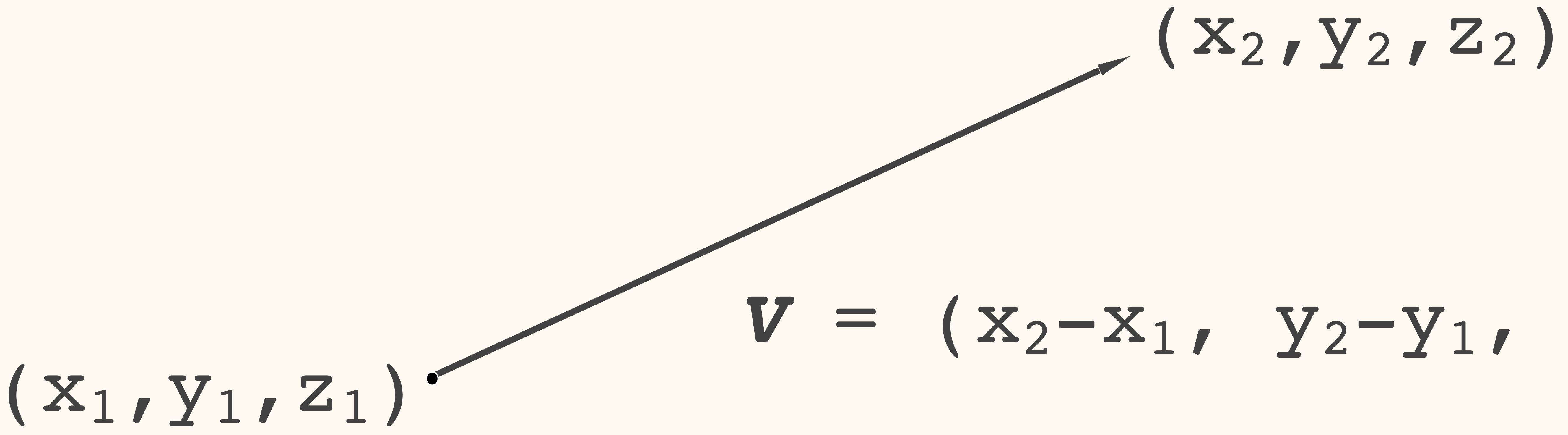
- 球座標系
  - $p(\rho,\phi,\theta)$
  - ρ : the distance to origin
  - ϕ : the angle from Z-axis
  - θ : the angle from X-axis to the projection of the P on XY-plane



3D Spherical Coordinate System

Vectors

- A vector is an entity that possesses *magnitude* and *direction*.
- A 3D vector is a triple :
  - $\boldsymbol{V} = (v_1, v_2, v_3)$, where each component v_i is a scalar.
- A ray (directed line segment), that possesses *position*, *magnitude* and *direction*.



$$\boldsymbol{V} = (x_2 - x_1,\ y_2 - y_1,\ z_2 - z_1)$$

Vectors

- Length of a vector

$$|\boldsymbol{V}| = (v_1^2 + v_2^2 + v_3^2)^{1/2}$$

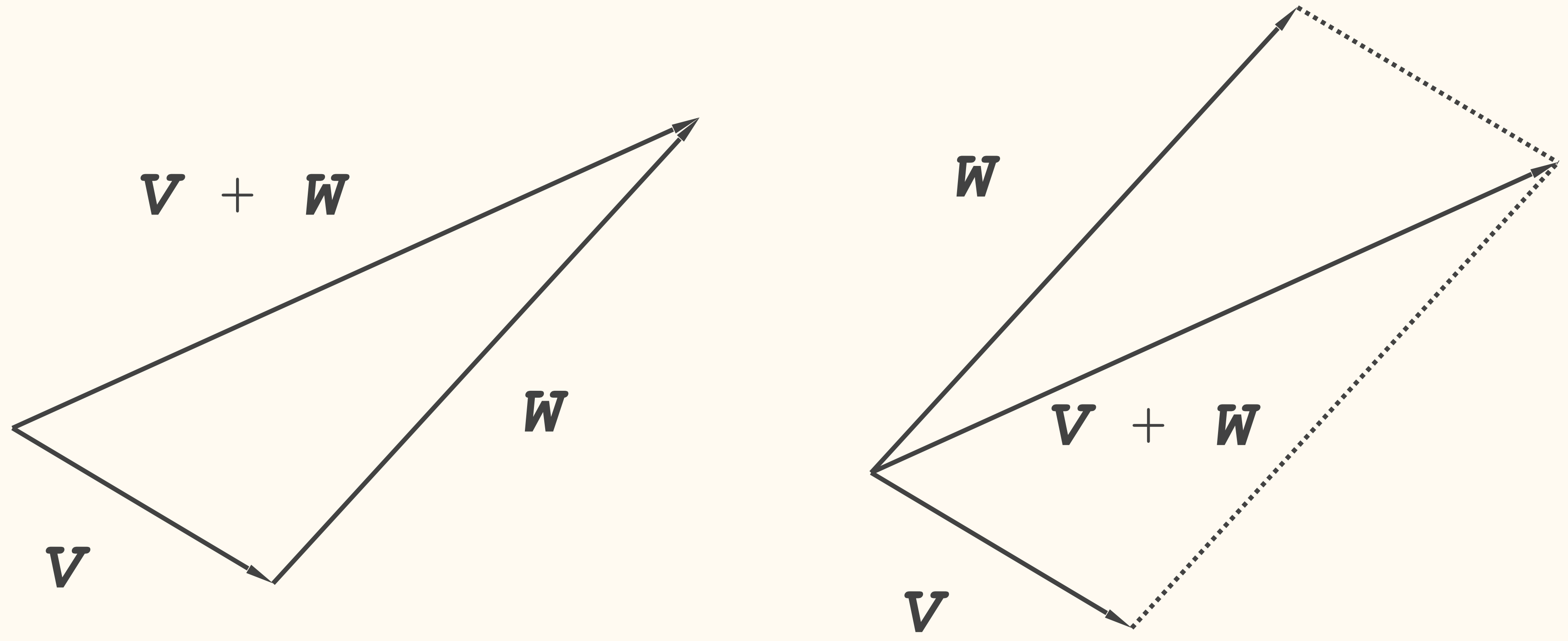
- Unit vector
  - 單位向量
  - Length = 1.0

$$\boldsymbol{U} = \boldsymbol{V} / |\boldsymbol{V}|$$

Vectors

- Addition of vectors

$$\boldsymbol{X} = \boldsymbol{V} + \boldsymbol{W}$$
$$= (x_1, y_1, z_1)$$
$$= (v_1 + w_1, v_2 + w_2, v_3 + w_3)$$



Vectors

- Cross product of two vectors

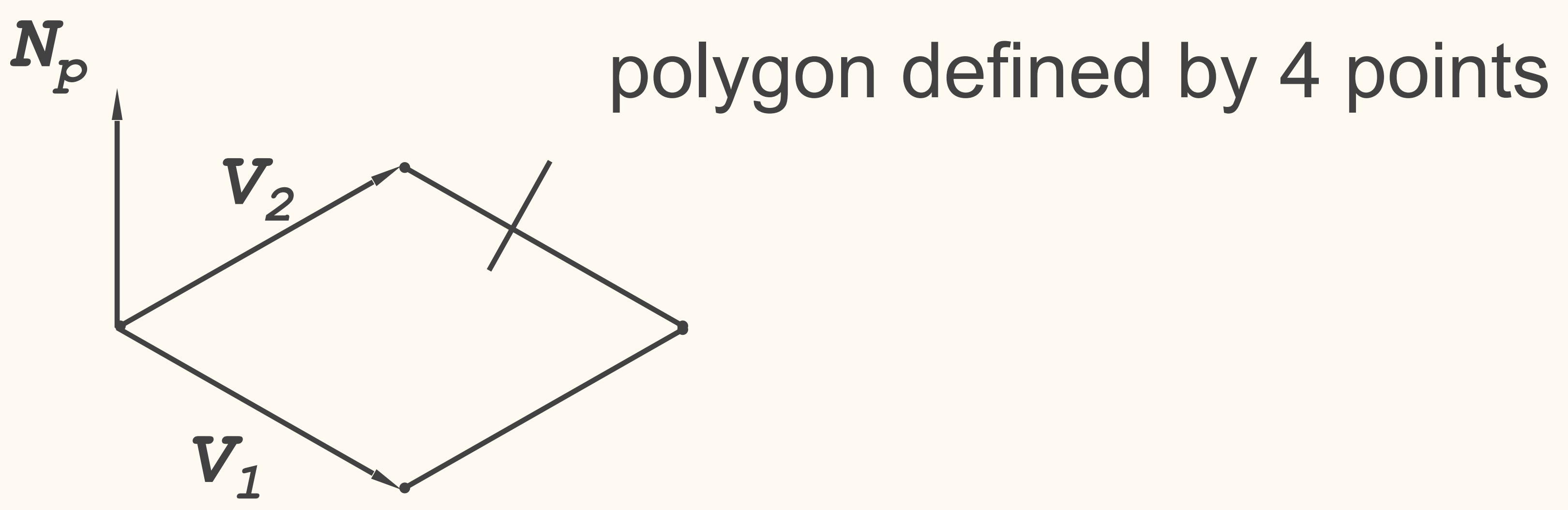
$$\boldsymbol{X} = \boldsymbol{V} \times \boldsymbol{W}$$
$$= (v_2w_3 - v_3w_2)\boldsymbol{i} + (v_3w_1 - v_1w_3)\boldsymbol{j} + (v_1w_2 - v_2w_1)\boldsymbol{k}$$

where $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$ are standard unit vectors :

$$\boldsymbol{i} = (1, 0, 0), \ \boldsymbol{j} = (0, 1, 0), \ \boldsymbol{k} = (0, 0, 1)$$

- Application : A normal vector to a polygon is calculated from 3 (non-collinear) vertices of the polygon.

$$\boldsymbol{N_p} = \boldsymbol{V_1} \times \boldsymbol{V_2}$$



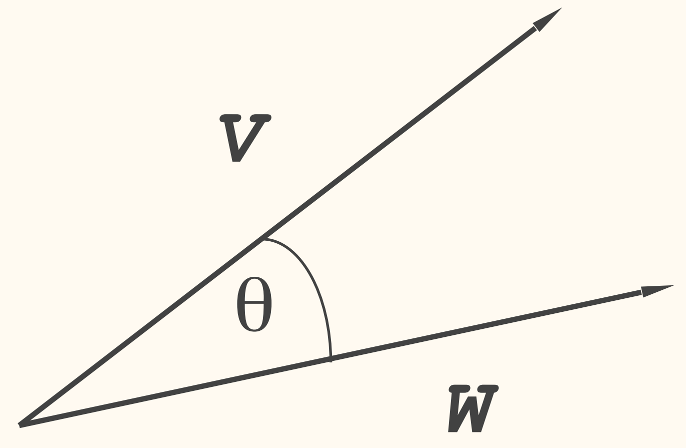
Vectors

- Dot product of two vectors

$$\begin{aligned} x &= \boldsymbol{V} \cdot \boldsymbol{W} \\ &= v_1w_1 + v_2w_2 + v_3w_3 \end{aligned}$$

- Application : A dot product of two unit vectors = the cosine value of the angle between these two vectors

$$\cos\theta = \frac{\boldsymbol{V} \cdot \boldsymbol{W}}{|\boldsymbol{V}|\,|\boldsymbol{W}|}$$



Matrix

- Basics :
  - Definition
    - $\boldsymbol{A}$ is a *mxn* matrix :

$$\boldsymbol{A} = (a_{ij}) = \begin{bmatrix} a_{11} & .. & a_{1n} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ a_{m1} & .. & a_{mn} \end{bmatrix}$$

  - Transpose

$$\boldsymbol{C} = \boldsymbol{A}^{T} = (a_{ji})$$

  - Addition

$$\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B} \qquad c_{ij} = a_{ij} + b_{ij}$$

Matrix

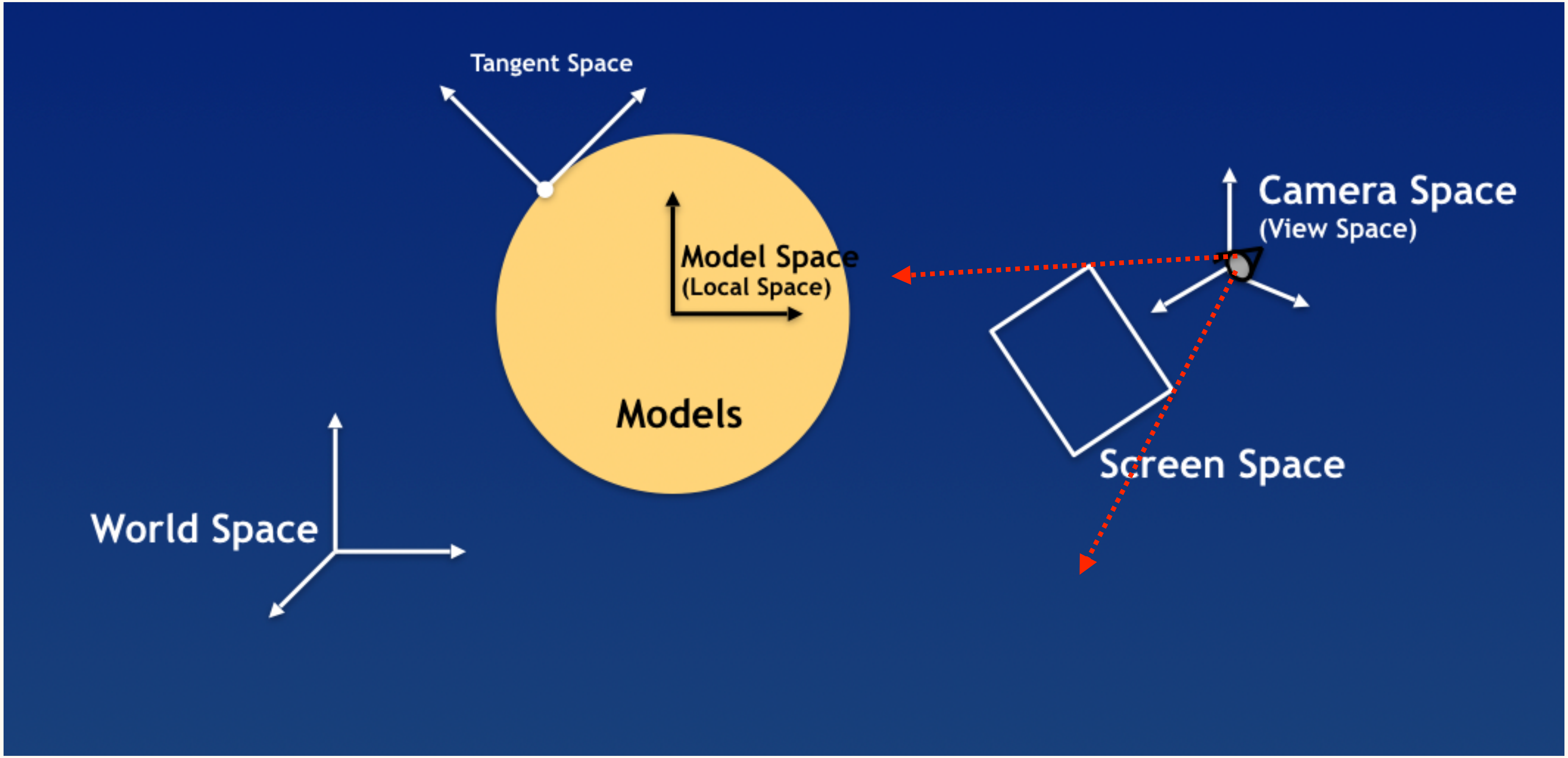
- Scalar-matrix multiplication

$$\boldsymbol{C} = \alpha\boldsymbol{A} \qquad c_{ij} = \alpha a_{ij}$$

- Matrix-matrix multiplication

$$\boldsymbol{C} = \boldsymbol{A}\,\boldsymbol{B} \qquad c_{ij} = \sum_{k=1}^{r} a_{ik}b_{kj}$$

Coordinate System in 3D (Space)



Linear Transformations

- Linear transformations are combinations of …
  - Scale, rotation, shear, and mirror
- Properties of linear transformations :
  - Origin maps to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved

Affine Transformations

- Affine transformations are combinations of …
  - Linear transformations, and translation
- Properties of affine transformations :
  - Origin does not necessary map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved

Transformations in Matrix Form

- A point is a column matrix $\mathbf{V}^T = [x\ y\ z]$

- Using matrix notation, a point $\mathbf{V}$ is transformed under translation, scaling and rotation as :

$$\mathbf{V}' = \mathbf{V} + \mathbf{D}$$
$$\mathbf{V}' = \mathbf{S}\mathbf{V}$$
$$\mathbf{V}' = \mathbf{R}\mathbf{V}$$

where $\mathbf{D}$ is a translation vector and $\mathbf{S}$ and $\mathbf{R}$ are scaling and rotation matrices

Translation

- Translation tranformation :

$$\mathbf{V}' = \mathbf{V} + \mathbf{D}$$

$\mathbf{D}$ is the translation vector (T_x, T_y, Tz)

$$x' = x + T_x$$
$$y' = y + T_y$$
$$z' = z + T_z$$

Rotations

- Rotations are the major transformation tool used in 3D
  - In the formats :
    - Euler angles :
      - Rotation with X, Y, and Z axes : $(\theta_x, \theta_y, \theta_z)$
    - Rotation with an arbitrary axis (x, y, z, θ) : a 4D vector
    - Quaternion : (w, x, y, z)
- Here we only discuss one of the Euler angles :
  - Rotation with Z-axis : $\mathbf{R_Z}$

$$\mathbf{R_Z} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling

- Scaling Transformation :

$$x' = xS_x$$
$$y' = yS_y$$
$$z' = zS_z$$

- In matrix form : **S**

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix}$$

A Series of Transformations

- We can net a series of transformation together

$$V' = \mathbf{M_1}V$$
$$V'' = \mathbf{M_2}V'$$

then the transformation matrices can be concatenated

$$\mathbf{M_3} = \mathbf{M_2 M_1}$$

$$V'' = \mathbf{M_3}V$$

Homogeneous Coordinate System

- To make the translation can be in matrix multiplication, we introduce the homogeneous coordinate system

$$\mathbf{v}^T = (x,\ y,\ z,\ w),\ \text{where } w \text{ is } 1$$

- Translation can be represented as :

$$\mathbf{V}' = \mathbf{TV}$$

$$= \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Homogeneous Coordinate System

- Scaling can be represented as :

$$\mathbf{V'} = \mathbf{SV}$$

$$= \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

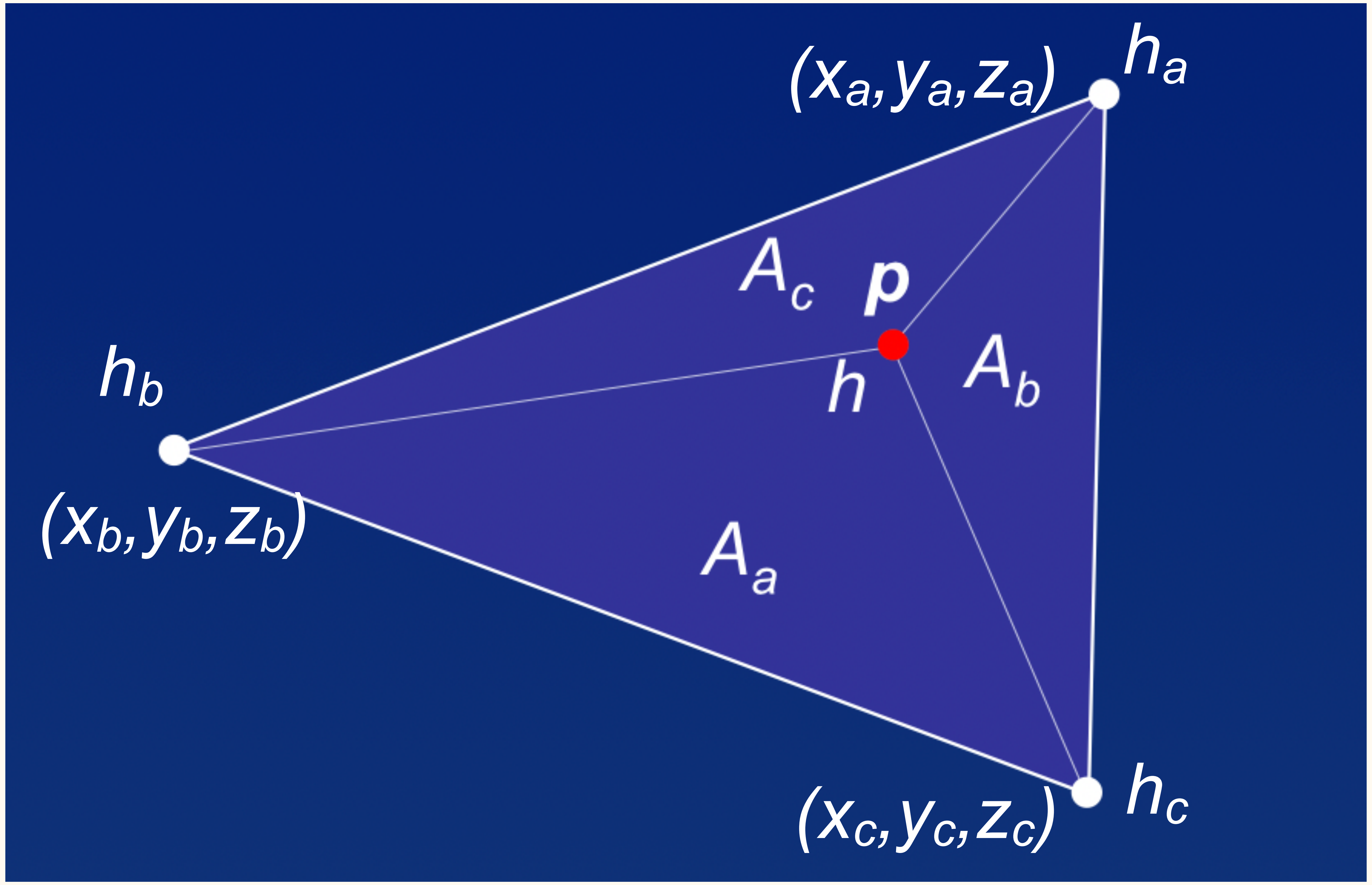
Homogeneous Coordinate System

- Rotation to Z-axis can be represented as :

$$\mathbf{V'} = \mathbf{R_z V}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Barycentric Coordinate System



$$h = \frac{A_a}{A} h_a + \frac{A_b}{A} h_b + \frac{A_c}{A} h_c$$

where $A = A_a + A_b + A_c$

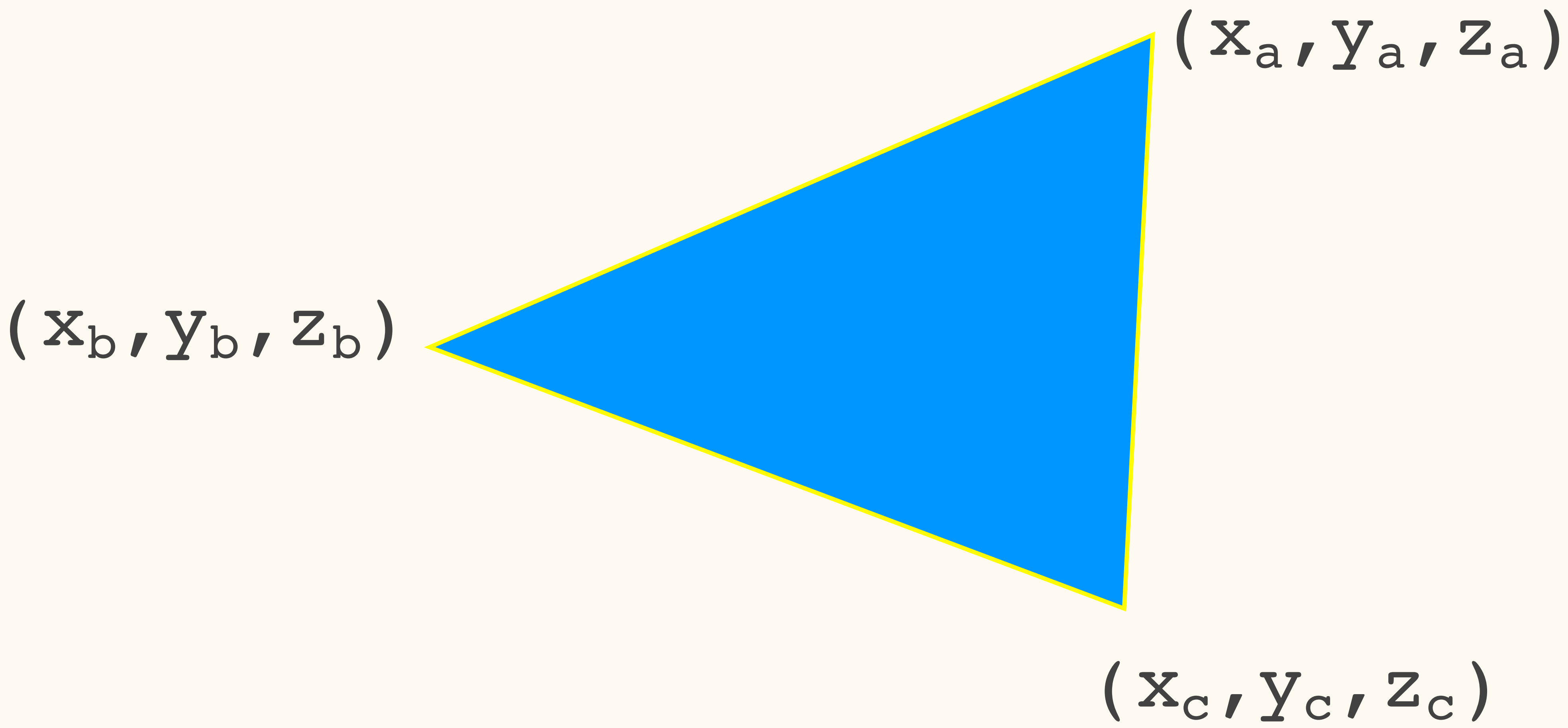
If $(A_a < 0 \,||\, A_b < 0 \,||\, A_c < 0)$ then the point is outside the triangle

<u>"Triangular Coordinate System"</u>
<u>"Barycentric Coordinate System"</u>

Triangle Area – 2D Solution

- If we only consider the 2D area :

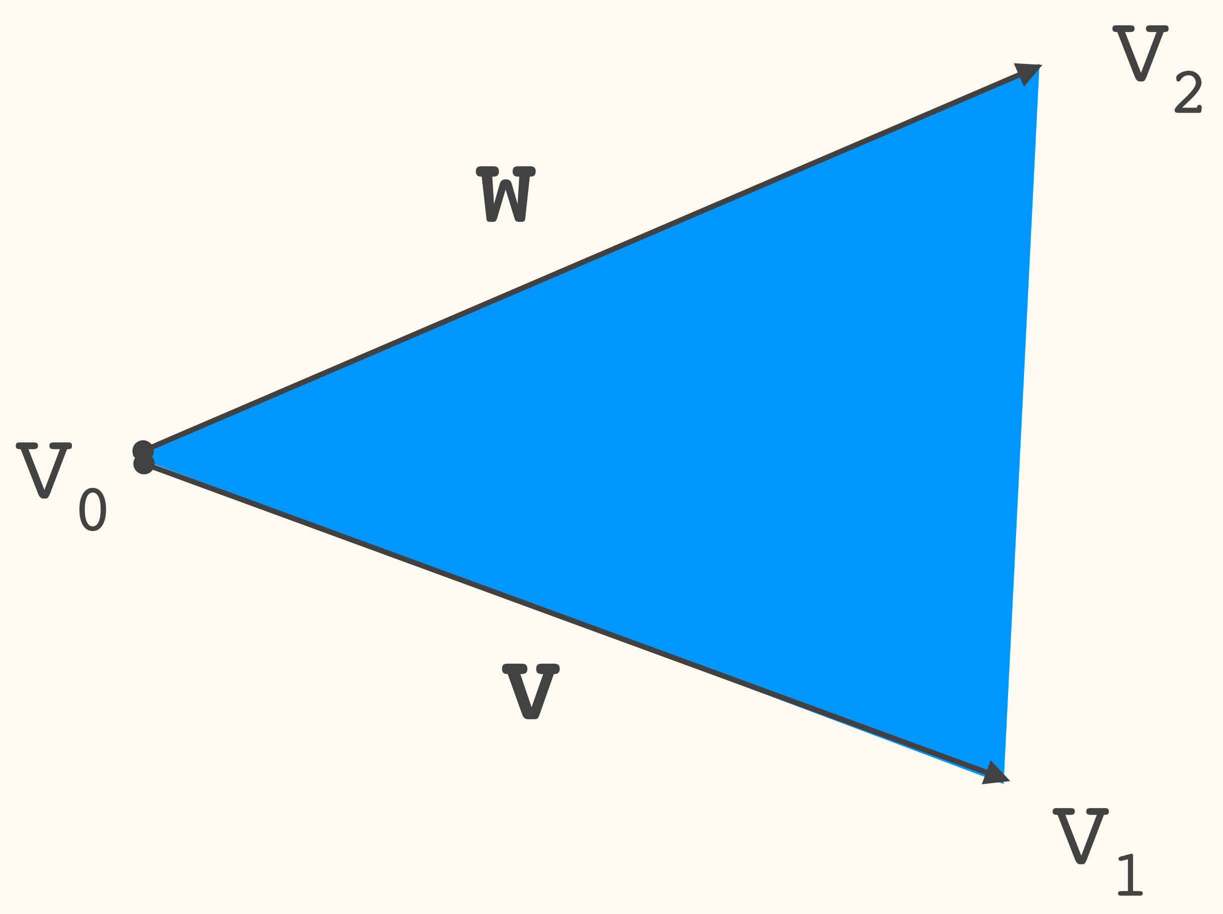
$$A = \tfrac{1}{2}\begin{vmatrix} x_a & y_a \\ x_b & y_b \\ x_c & y_c \\ x_a & y_a \end{vmatrix}$$



$$= \tfrac{1}{2}(x_a * y_b + x_b * y_c + x_c * y_a - x_b * y_a - x_c * y_b - x_a * y_c)$$

Triangle Area – 3D Solution

$$A(\Delta) = \tfrac{1}{2}\left|\mathbf{v} \times \mathbf{w}\right|$$
$$= \tfrac{1}{2}\left|\left(V_1 - V_0\right) \times \left(V_2 - V_0\right)\right|$$



Barycentric Coordinate System Applications

- Terrain following
  - Interpolating the height of arbitrary point within the triangle
- Hit test
  - Intersection of a ray from camera to a screen position with a triangle
- Ray cast
  - Intersection of a ray with a triangle
- Collision detection
  - Intersection

Intersections

- Ray Casting
- Containment Test
- Separating Axis

Ray Casting – The Ray Equation

- Cast a ray to calculate the intersection of the ray with models
- Use parametric equation for a ray

$$
\begin{aligned}
x &= x_0 + (x_1 - x_0)\, t \\
y &= y_0 + (y_1 - y_0)\, t, \qquad t = 0,\ 8 \\
z &= z_0 + (z_1 - z_0)\, t
\end{aligned}
$$

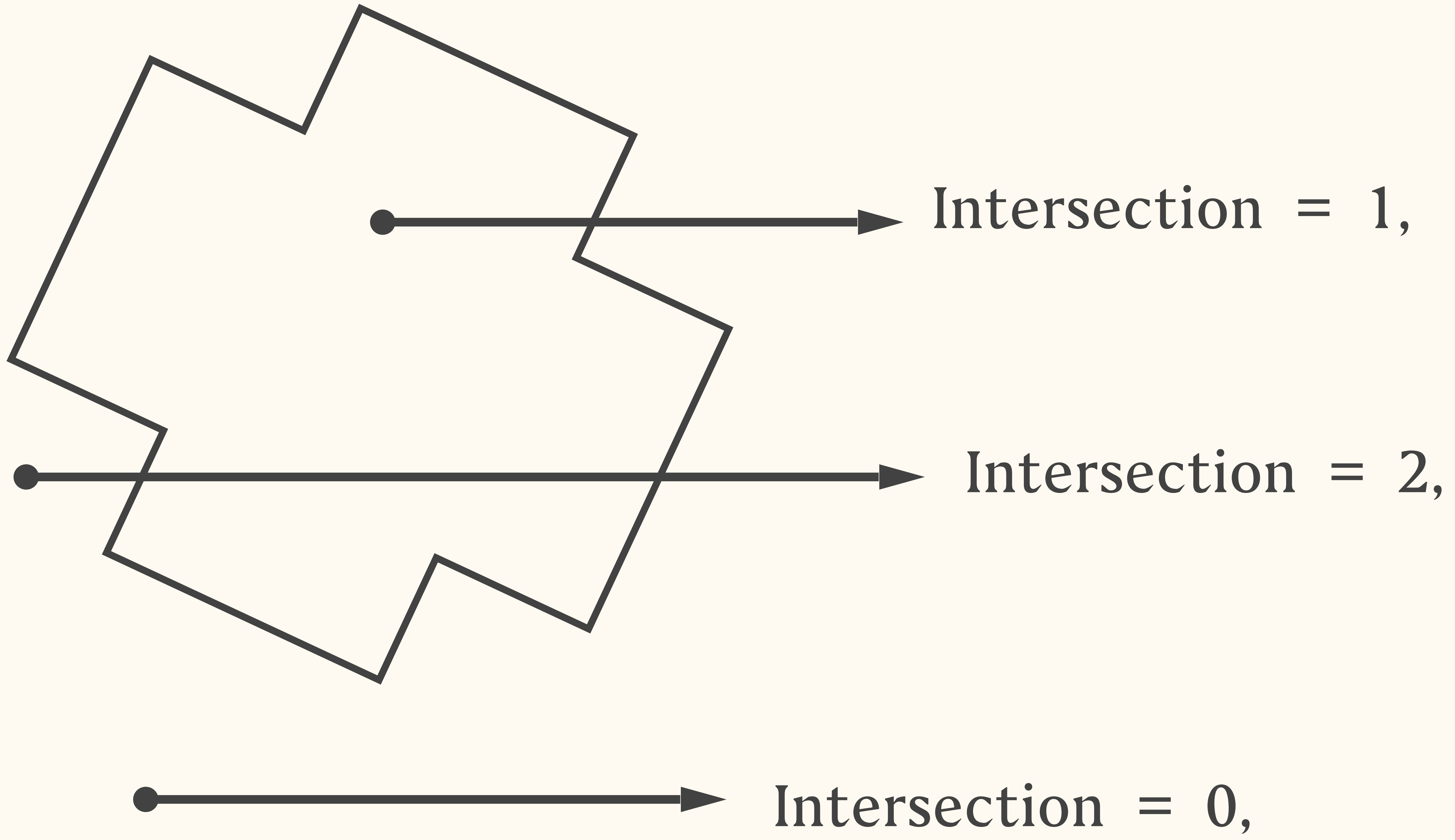
- When $t = 0$, the ray is on the start point (x_0, y_0, z_0)
- Only the $t \geq 0$ is the answer candidate
- The smallest positive t is the answer

Ray Casting – The Plane Equation

- Each triangle in the 3D models has its plane equation.
- Use `ax + by + cz + d = 0` as the plane equation.
- `(a, b, c)` is the plane normal vector.
- `|d|` is the distance of the plane to origin.
- Substitute the ray equation into the plane.
- Solve the t to find the intersect point.
- Check the intersect point within the triangle or not by using “Triangle Area Test”

Containment Test

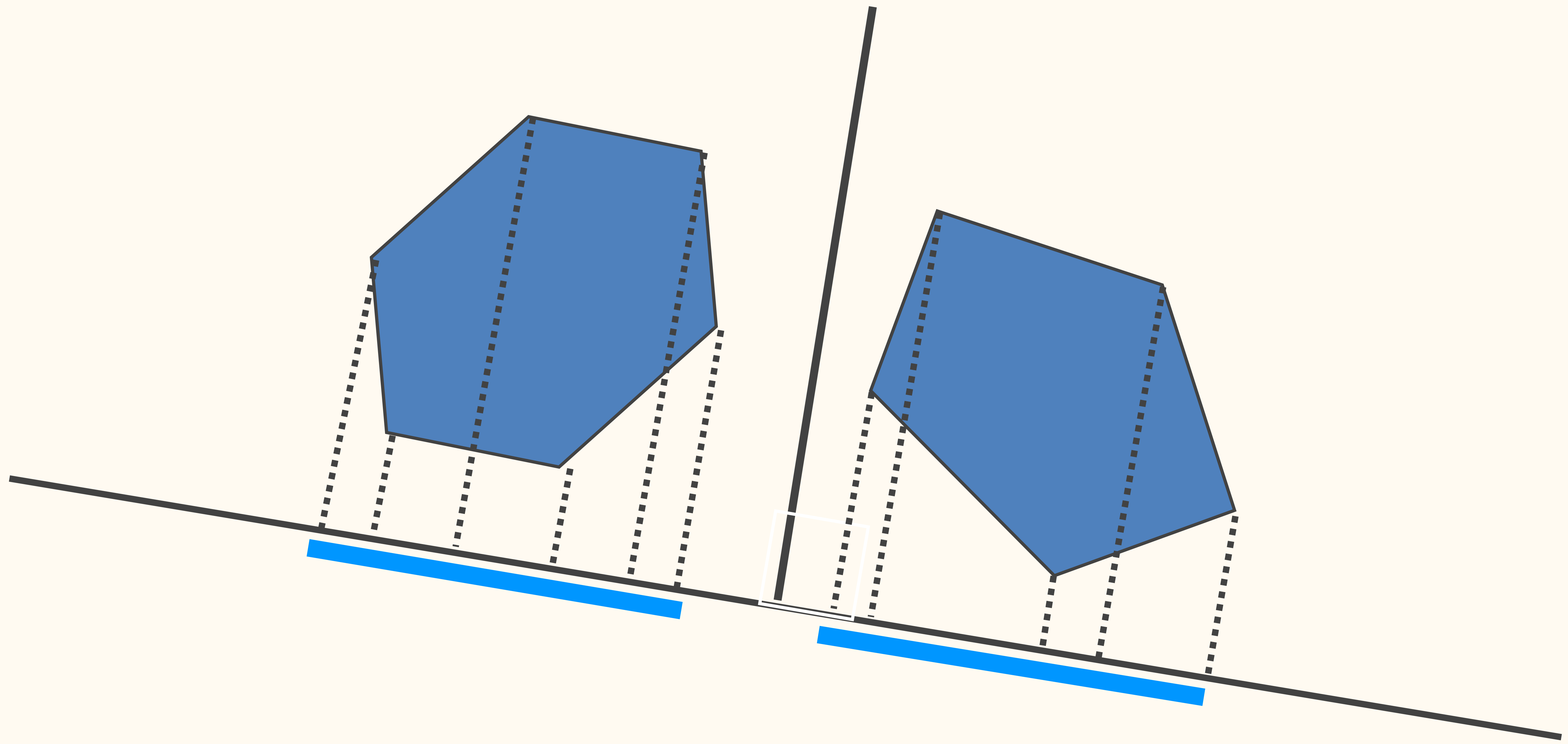
- 2D version



"If no. of intersection is odd, the point is inside, otherwise, it's outside"

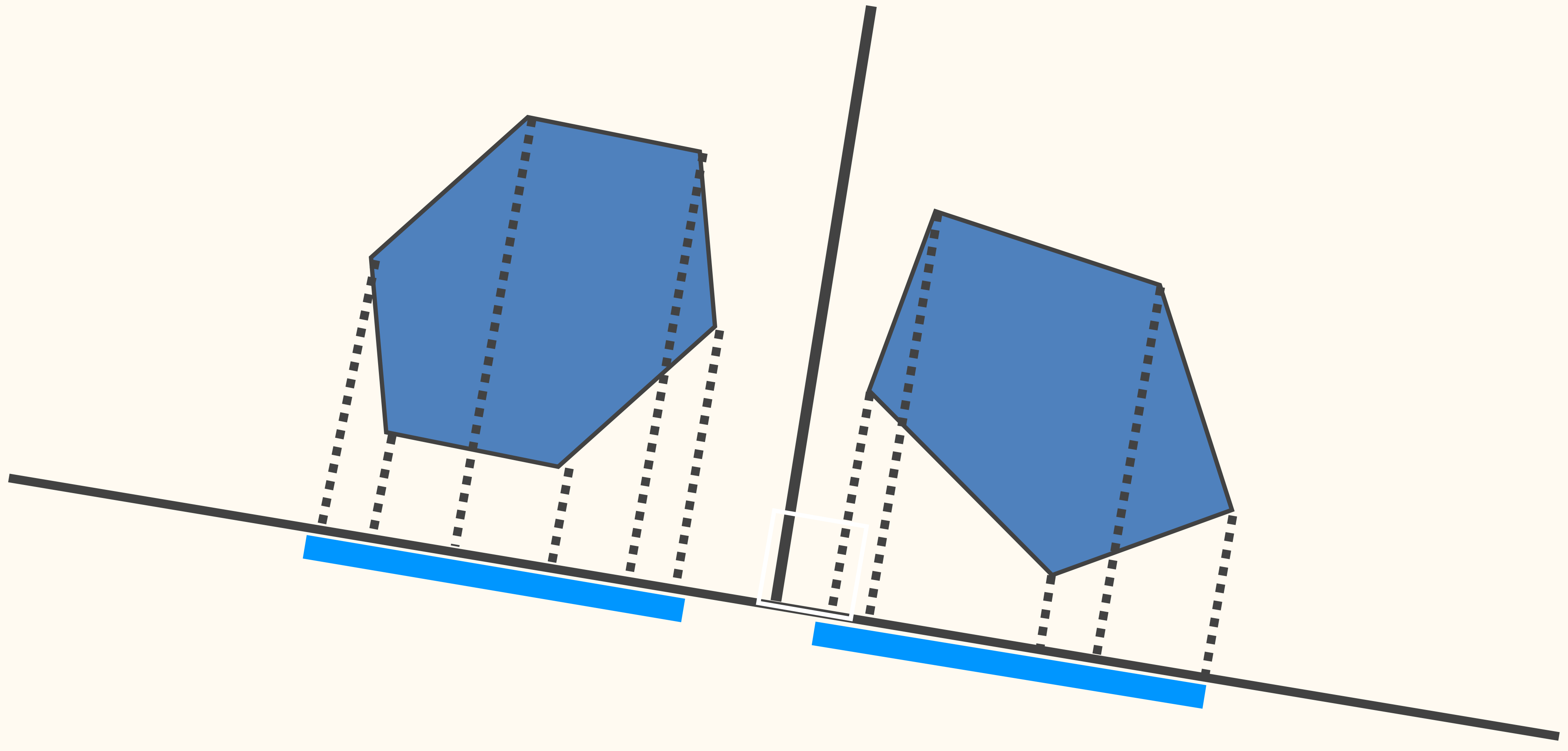
Separating Axis

- For convex objects only
- If there is existing an axis (2D) or a plane (3D) to separate two convex objects, these two objects are not intersected.



Separating Axis

- How?
  - Project the vertices of each object on the axis/plane that is perpendicular to axis/plane we are going to find.
  - Get the extreme of the projection area of each object.
  - If the projection are of these two object are not overlapped, the two objects are not intersected.

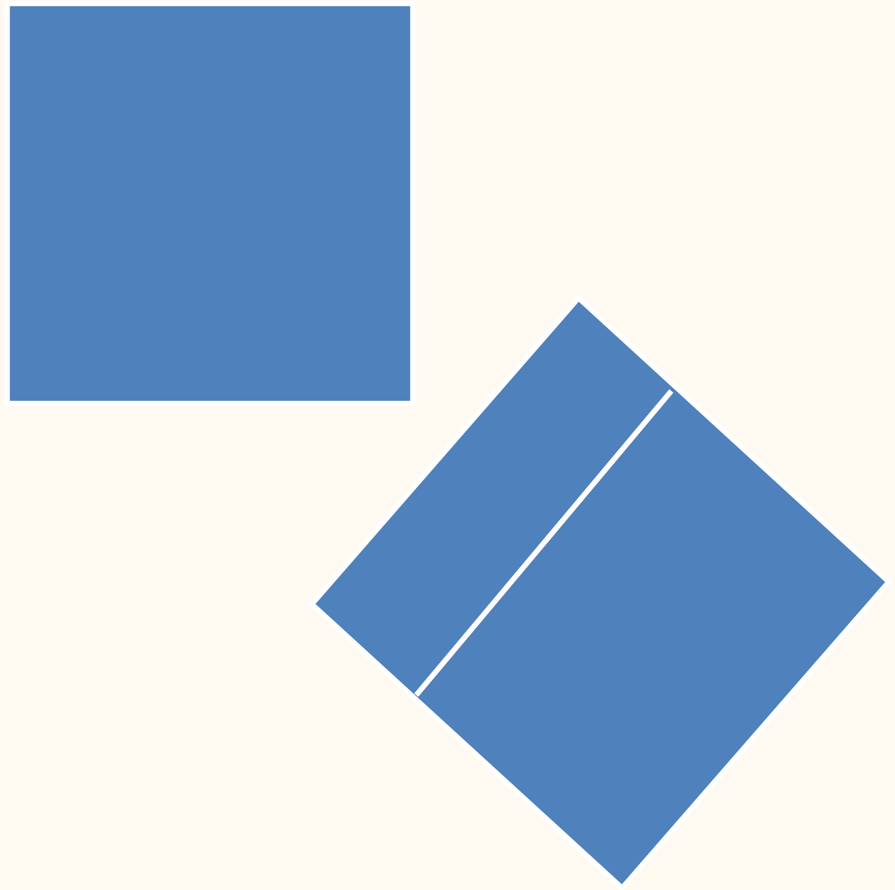


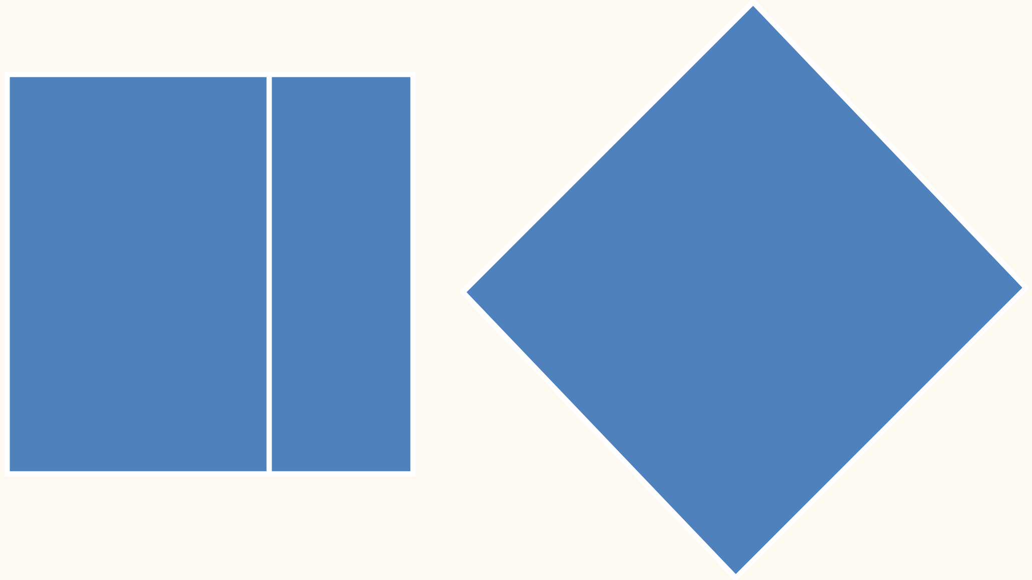
Separating Axis Algorithm

```
Bool TestIntersect(ConvexPolyhedron C0, ConvexPolyhedron C1)
{
   // test faces of C0 for separation
   for (i = 0; i < C0.GetFaceCount(); i++) {
      D = C0.GetNormal(i);
      ComputeInterval(C0, D, min0, max0);
      ComputeInterval(C1, D, min1, max1);
      if (max1 < min0 || max0 < min1) return false;
   }

   // test faces of C1 for separation
   for (i = 0; i < C1.GetFaceCount(); i++) {
      D = C1.GetNormal(i);
      ComputeInterval(C0, D, min0, max0);
      ComputeInterval(C1, D, min1, max1);
      if (max1 < min0 || max0 < min1) return false;
   }
```

Separating Axis Algorithm

```
    // test cross products of pairs of edges
    for (i = 0; i < C0.GetEdgeCount(); i++) {
        for (j = 0; j < C1.GetEdgeCount(); j++) {
            D = Cross(C0.GetEdge(i), C1.GetEdge(j));
            ComputeInterval(C0, D, min0, max0);
            ComputeInterval(C1, D, min1, max1);
            if (max1 < min0 || max0 < min1) return false;
        }
    }

    return true;
}
```





Separating Axis Algorithm

```
void ComputeInterval(ConvexPolyhedron C, Vector D,
                                  double &min, double &max)
{
   min = Dot(D, C.GetVertex(0));
   max = min;
   for (i = 1; i < C.GetVertexCount(); i++) {
      value = Dot(D, C.GetVertex(i));
      if (value < min) min = value;
      else if (value > max) max = value;
   }
}
```

Rotations

- Euler Angles
- Rotation with an Arbitrary Axis
- Quaternion

Euler Angles

- A rotation is described as a sequence of rotations about three mutually orthogonal coordinates axes fixed in space
  - X-roll, Y-roll, Z-roll

$\boldsymbol{R}(\theta_1, \theta_2, \theta_3)$ represents an x-roll, followed by y-roll, followed by z-roll

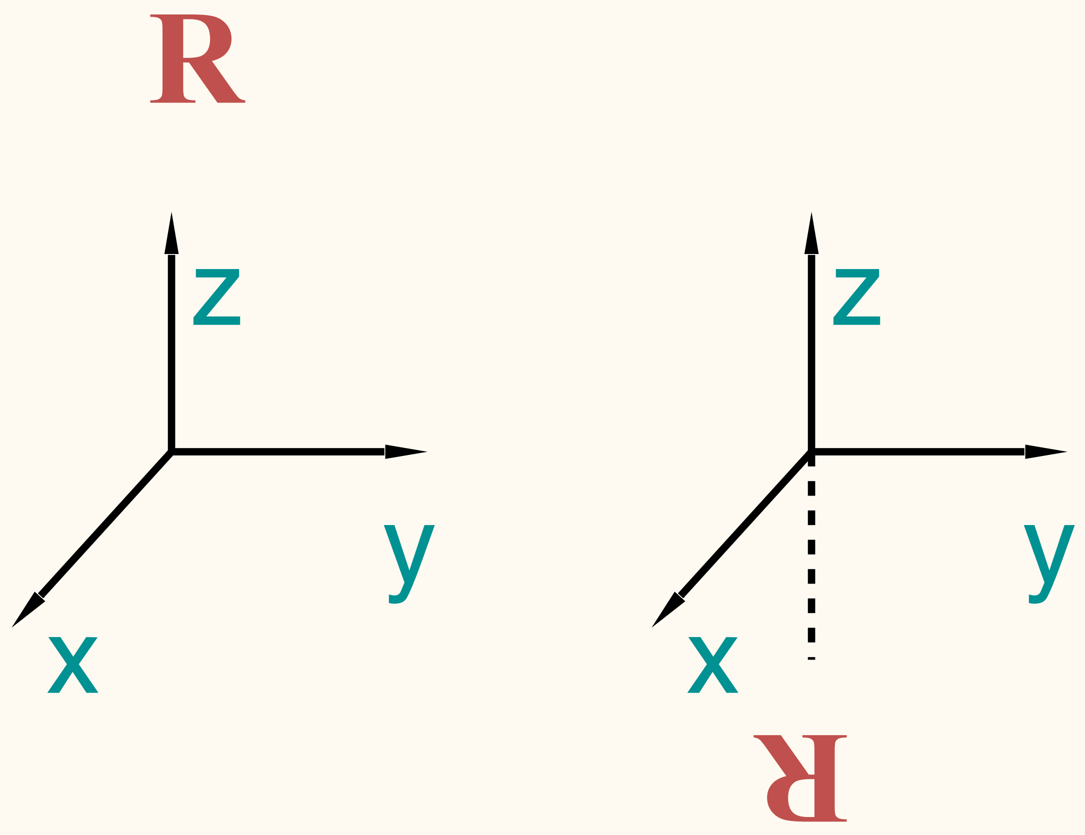
$$\boldsymbol{R}(\theta_1, \theta_2, \theta_3) = \begin{bmatrix} c_2c_3 & c_2s_3 & -s_2 & 0 \\ s_1s_2c_3-c_1s_3 & s_1s_2s_3+c_1c_3 & s_1c_2 & 0 \\ c_1s_2c_3+s_1s_3 & c_1s_2s_3-s_1c_3 & c_1c_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $s_i = \sin\theta_i$ and $c_i = \cos\theta_i$

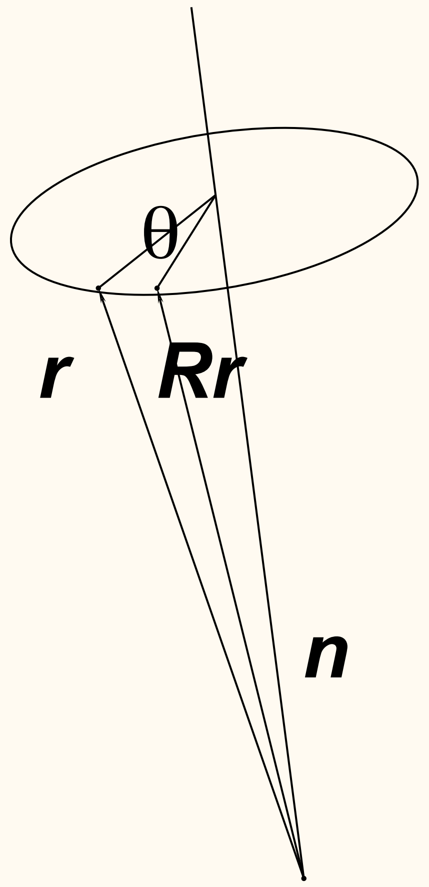
- 3! possibilities

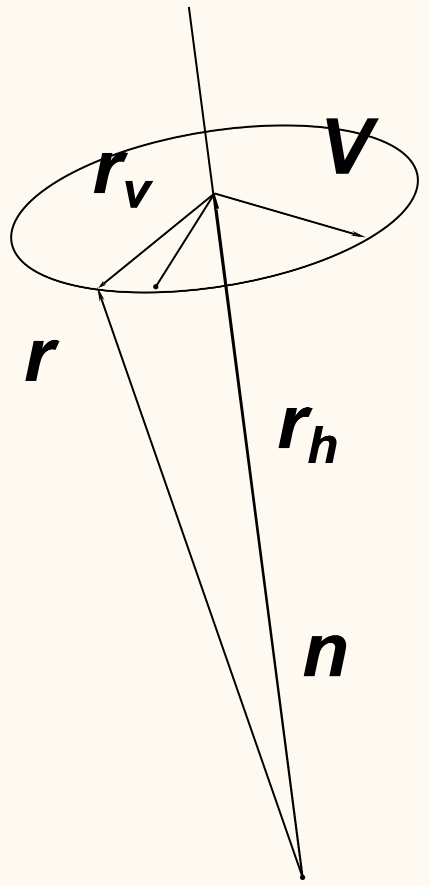
Euler Angles & Interpolation

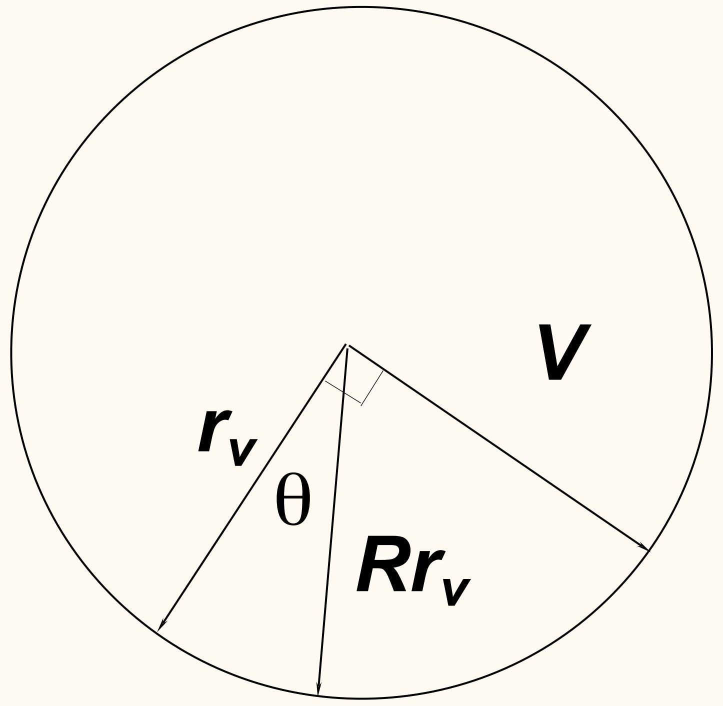
- Interpolation happening on each angle
- Multiple routes for interpolation
- More keys for constrains



Rotation with Arbitrary Axis







- R(θ , n), n is the rotation axis, θ is the angle.

$$\boldsymbol{r_h} = (\boldsymbol{n}\cdot\boldsymbol{r})\boldsymbol{n}$$

$\boldsymbol{r_v} = \boldsymbol{r} - (\boldsymbol{n}\cdot\boldsymbol{r})\boldsymbol{n}$, rotate into position $\boldsymbol{Rr_v}$

$$\boldsymbol{V} = \boldsymbol{n}\times\boldsymbol{r_v} = \boldsymbol{n}\times\boldsymbol{r}$$

$$\boldsymbol{Rr_v} = (\cos\theta)\boldsymbol{r_v} + (\sin\theta)\boldsymbol{V}$$

->

$$\begin{aligned}\boldsymbol{Rr} &= \boldsymbol{Rr_h} + \boldsymbol{Rr_v} \\ &= \boldsymbol{r_h} + (\cos\theta)\boldsymbol{r_v} + (\sin\theta)\boldsymbol{V} \\ &= (\boldsymbol{n}\cdot\boldsymbol{r})\boldsymbol{n} + (\cos\theta)\,(\boldsymbol{r} - (\boldsymbol{n}\cdot\boldsymbol{r})\boldsymbol{n}) + (\sin\theta)\,\boldsymbol{n}\times\boldsymbol{r} \\ &= (\cos\theta)\boldsymbol{r} + (1-\cos\theta)\,\boldsymbol{n}\,(\boldsymbol{n}\cdot\boldsymbol{r}) + (\sin\theta)\,\boldsymbol{n}\times\boldsymbol{r}\end{aligned}$$

Quaternions

- By Sir William Hamilton (1843)
- From Complex numbers ($a + \boldsymbol{i}b$), $\boldsymbol{i}^2 = -1$
- 16,October, 1843, Broome Bridge in Dublin
- 1 real + 3 imaginary = 1 quaternion
- $\boldsymbol{q} = a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$
- $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = -1$
- $\boldsymbol{ij} = \boldsymbol{k}$ & $\boldsymbol{ji} = -\boldsymbol{k}$, cyclic permutation $\boldsymbol{i}$-$\boldsymbol{j}$-$\boldsymbol{k}$-$\boldsymbol{i}$
- $\boldsymbol{q} = (s, \boldsymbol{v})$, where $(s, \boldsymbol{v}) = s + v_x\boldsymbol{i} + v_y\boldsymbol{j} + v_z\boldsymbol{k}$

Quaternion Algebra

$\boldsymbol{q_1} = (s_1, \boldsymbol{v_1})$ *and* $\boldsymbol{q_2} = (s_2, \boldsymbol{v_2})$

$\boldsymbol{q_3} = \boldsymbol{q_1 q_2} = (s_1 s_2 - \boldsymbol{v_1} \cdot \boldsymbol{v_2}, s_1 \boldsymbol{v_2} + s_2 \boldsymbol{v_1} + \boldsymbol{v_1} x \boldsymbol{v_2})$

Conjugate of $\boldsymbol{q} = (s, \boldsymbol{v})$, $\quad \boldsymbol{q} = (s, -\boldsymbol{v})$

$\boldsymbol{qq} = s^2 + |\boldsymbol{v}|^2 = |\boldsymbol{q}|^2$

A unit quaternion $\boldsymbol{q} = (s, \boldsymbol{v})$*, where* $\boldsymbol{qq} = 1$

A pure quaternion $\boldsymbol{p} = (0, \boldsymbol{v})$

Quaternion As Rotations

Take a pure quaternion $\mathbf{p} = (0, \mathbf{r})$
and a unit quaternion $\mathbf{q} = (s, \mathbf{v})$ *where* $\mathbf{qq} = 1$
and define $\mathbf{R_q}(\mathbf{p}) = \mathbf{qpq}^{-1}$ *where* $\mathbf{q}^{-1} = \mathbf{q}$ *for a unit quaternion*

$$\mathbf{R_q}(\mathbf{p}) = (0, (s^2 - \mathbf{v}\cdot\mathbf{v})\mathbf{r} + 2\mathbf{v}(\mathbf{v}\cdot\mathbf{r}) + 2s\mathbf{v}x\mathbf{r})$$

Let $\mathbf{q} = (\cos\theta, \sin\theta\ \mathbf{n}), \quad |\mathbf{n}| = 1$

$$\mathbf{R_q}(\mathbf{p}) = (0, (\cos^2\theta - \sin^2\theta)\mathbf{r} + 2\sin^2\theta\ \mathbf{n}(\mathbf{n}\cdot\mathbf{r}) + 2\cos\theta\sin\theta\ \mathbf{n}x\mathbf{r})$$
$$= (0, \cos 2\theta\mathbf{r} + (1 - \cos 2\theta)\mathbf{n}(\mathbf{n}\cdot\mathbf{r}) + \sin 2\theta\ \mathbf{n}x\mathbf{r})$$

Conclusion :
The act of rotating a vector r by an angular displacement $(\theta, \mathbf{n})$ *is the same as taking this displacement, 'lifting' it into quaternion space, by using a unit quaternion* $(\cos(\theta/2), \sin(\theta/2)\mathbf{n})$

Quaternion To Rotation Matrix

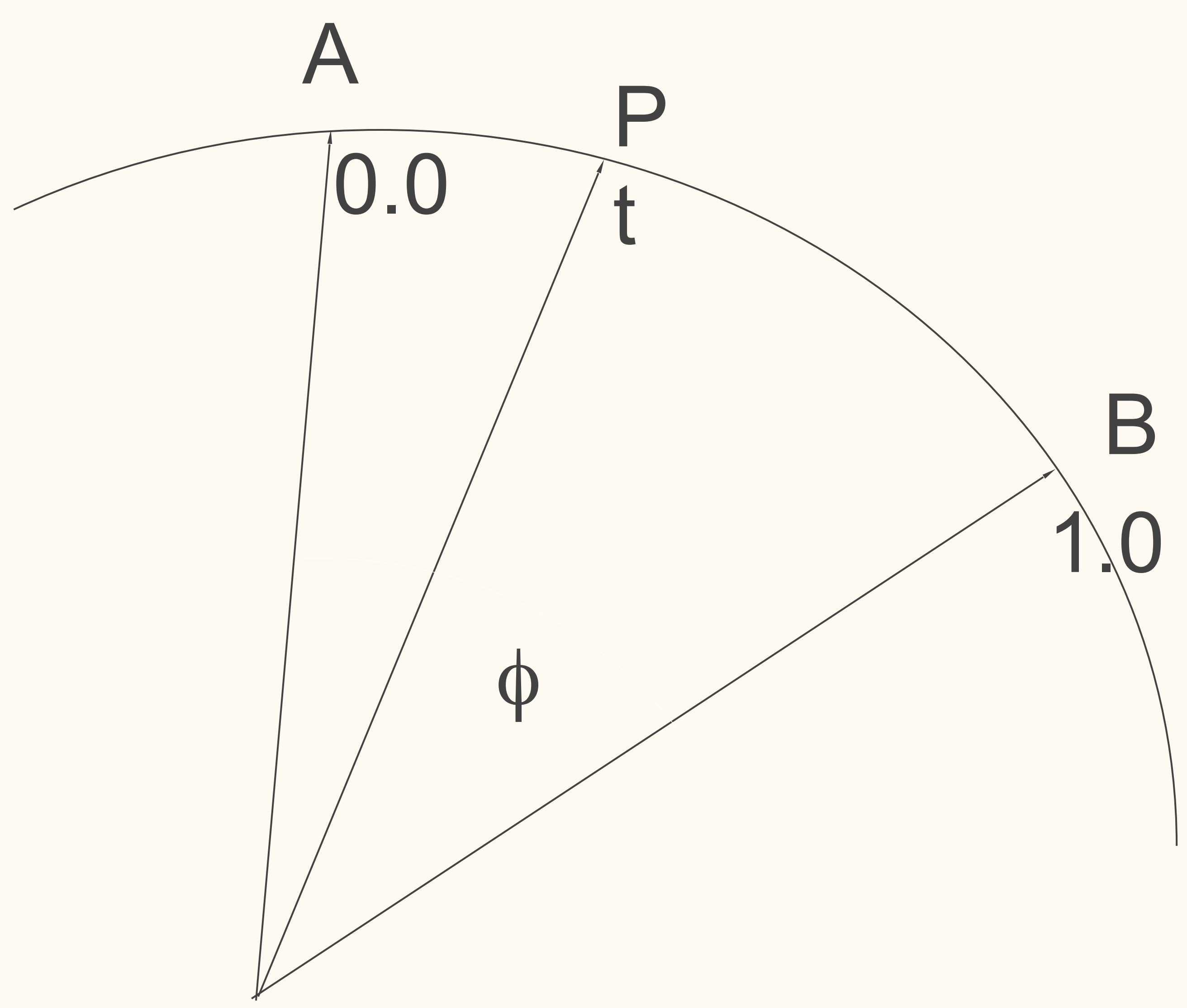
$$\boldsymbol{q} = (w,x,y,z) \quad \longrightarrow \quad \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2wz & 2xz+2wy & 0 \\ 2xy+2wz & 1-2x^2-2z^2 & 2yz-2wx & 0 \\ 2xz-2wy & 2yz+2wx & 1-2x^2-2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation Matrix To Quaternion

$$\begin{bmatrix} M_0 & M_1 & M_2 & 0 \\ M_3 & M_4 & M_5 & 0 \\ M_6 & M_7 & M_8 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \longrightarrow$$

```
float tr, s;

  tr = m[0] + m[4] + m[8];
  if (tr > 0.0f) {
    s = (float) sqrt(tr + 1.0f);
    q->w = s/2.0f;
    s = 0.5f/s;

    q->x = (m[7] - m[5])*s;
    q->y = (m[2] - m[6])*s;
    q->z = (m[3] - m[1])*s;
  }
  else {
    float qq[4];
    int i, j, k;
    int nxt[3] = {1, 2, 0};

    i = 0;
    if (m[4] > m[0]) i = 1;
    if (m[8] > m[i*3+i]) i = 2;
    j = nxt[i]; k = nxt[j];

    s = (float) sqrt((m[i*3+i] - (m[j*3+j] + m[k*3+k]))
          + 1.0f);
    qq[i] = s*0.5f;
    if (s != 0.0f) s = 0.5f/s;

    qq[3] = (m[j+k*3] - m[k+j*3])*s;
    qq[j] = (m[i+j*3] + m[j+i*3])*s;
    qq[k] = (m[i+k*3] + m[k+i*3])*s;

    q->w = qq[3];
    q->x = qq[0];
    q->y = qq[1];
    q->z = qq[2];
  }
```

Quaternion Interpolation



$$\mathbf{slerp}(q_1, q_2, t) = q_1 \frac{\sin((1-t)\phi)}{\sin\phi} + q_2 \frac{\sin(t\phi)}{\sin\phi}$$