Geometric Objects - Spaces and Matrix

Fall 2023
10/5/2023

Kyoung Shin Park
Computer Engineering
Dankook University

Spaces

- Vector space
  - The vector space has scalars and vectors.
  - Scalars: α, β, δ
  - Vectors: u, v, w
- Affine space
  - The affine space has point in addition to the vector space.
  - Points: P, Q, R
- Euclidean space
  - In Euclidean space, the concept of distance is added.

Scalars, Points, Vectors

- 3 basic types needed to describe the geometric objects and their relations
- Scalars: α, β, δ
- Points: P, Q, R
- Vectors: u, v, w
- Vector space
  - scalars & vectors
- Affine space
  - Extension of the vector space that includes a point

Scalars

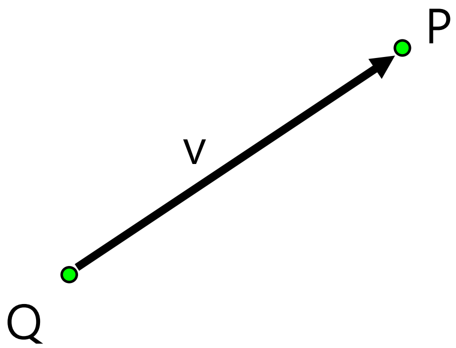
- Commutative, associative, and distribution laws are established for addition and multiplication
  - $\alpha + \beta = \beta + \alpha$
  - $\alpha \cdot \beta = \beta \cdot \alpha$
  - $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$
  - $\alpha \cdot (\beta \cdot \gamma) = (\alpha \cdot \beta) \cdot \gamma$
  - $\alpha \cdot (\beta + \gamma) = (\alpha \cdot \beta) + (\alpha \cdot \gamma)$
- Addition identity is 0 and multiplication identity is 1.
  - $\alpha + 0 = 0 + \alpha = \alpha$
  - $\alpha \cdot 1 = 1 \cdot \alpha = \alpha$
- Inverse of addition and inverse of multiplication
  - $\alpha + (-\alpha) = 0$
  - $\alpha \cdot \alpha^{-1} = 1$

Vectors

- Vectors have magnitude (or length) and direction.
- Physical quantities, such as velocity or force, are vectors.
- Directed line segments used in computer graphics are vectors.
- Vectors do not have a fixed position in space.

Points

- Points have a position in space.
- Operations with points and vectors:
  - Point-point subtraction creates a vector.
  - Point-vector addition creates points.

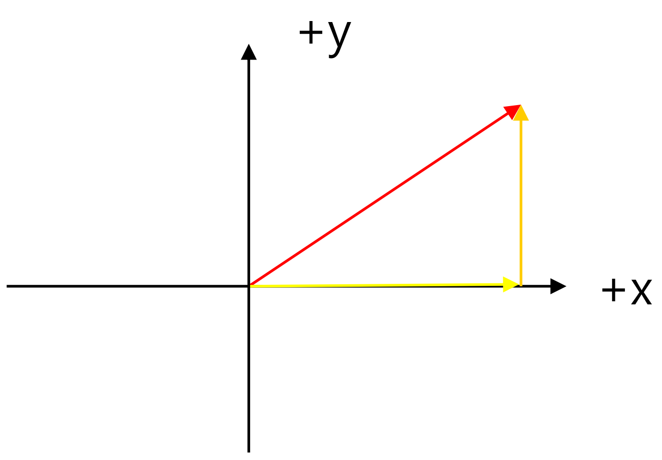


$$v = P - Q$$

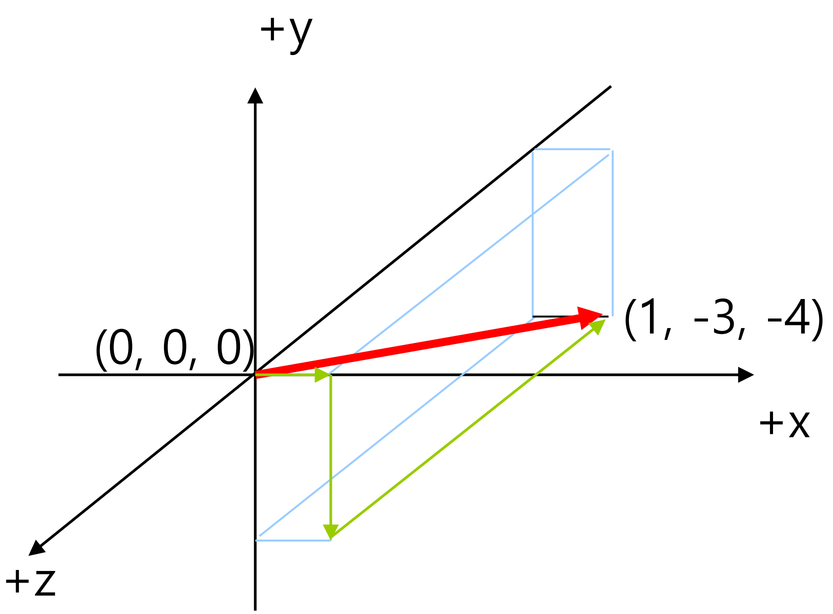
$$P = Q + v$$

Specifying Vectors

- 2D Vector: (x, y)
- 3D Vector: (x, y, z)



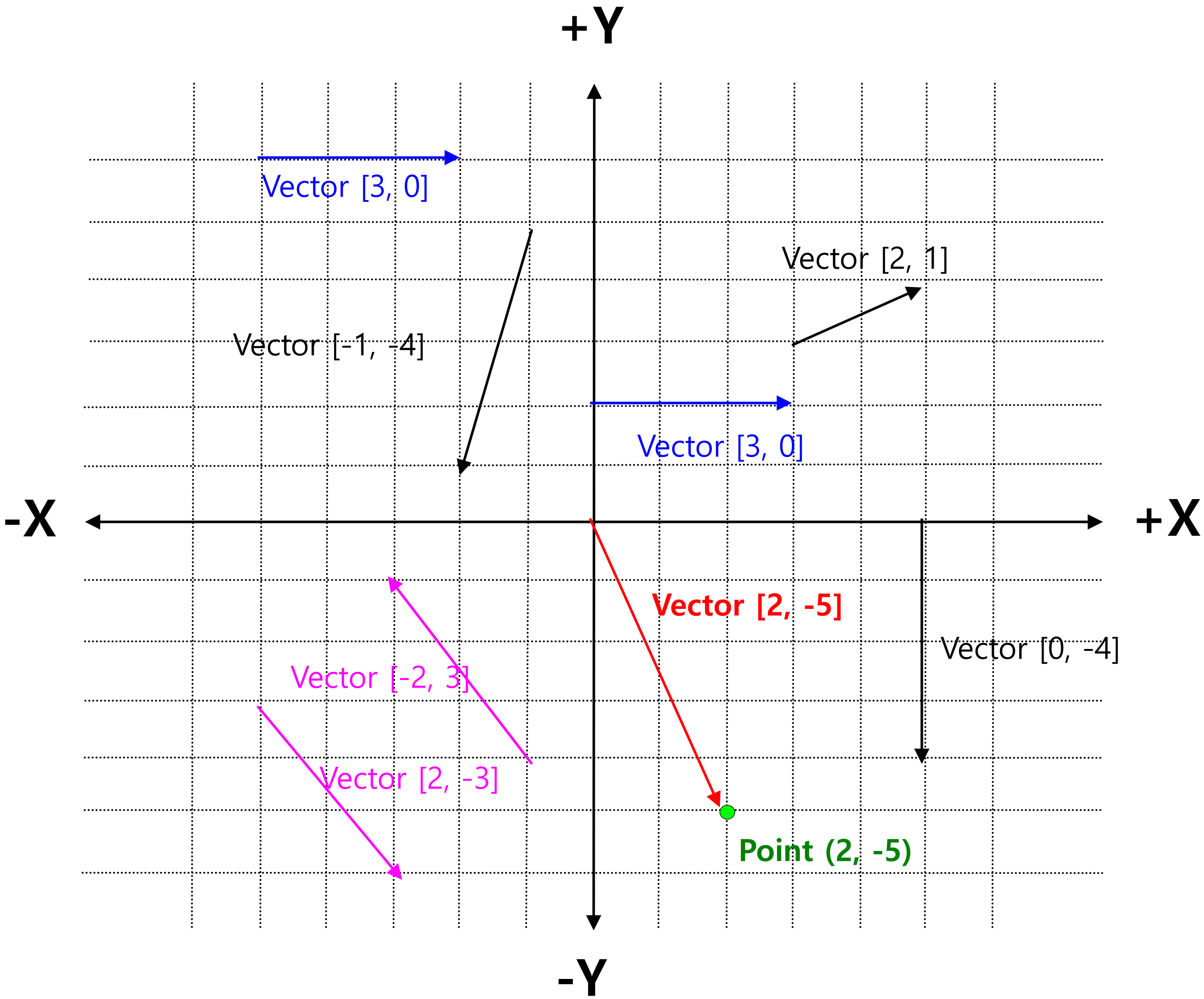
2D Vector



3D Vector

Vector from the origin O(0, 0, 0) to the point P(1, -3, -4)

Examples of 2D vectors

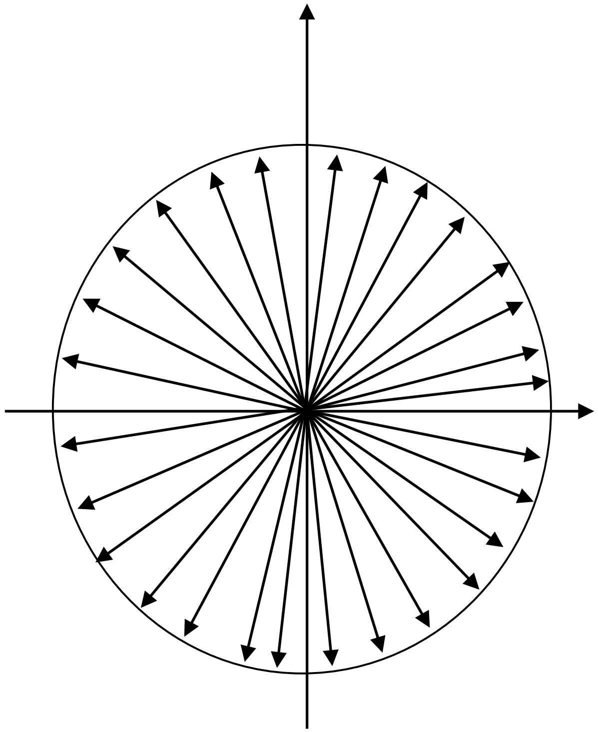


Vector Operations

- zero vector
- vector negation
- vector/scalar multiply
- add & subtract two vectors
- vector magnitude (length)
- normalized vector
- distance formula
- vector product
  - dot product
  - cross product

The Zero Vector

- The three-dimensional zero vector is (0, 0, 0).
- The zero vector has zero magnitude.
- The zero vector has no direction.



Negating a Vector

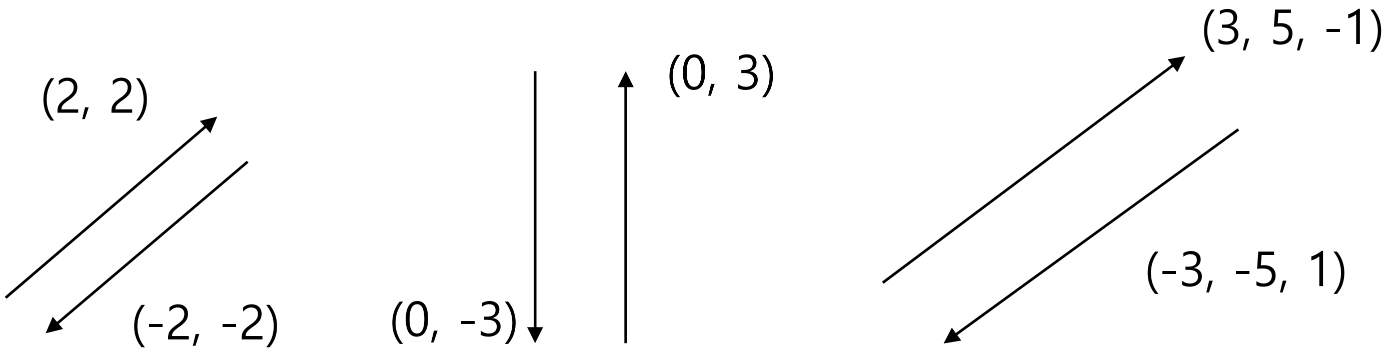
- Every vector **v** has a negative vector **–v**: **v + (-v) = 0**
- Negative vector

  $-(a_1, a_2, a_3, \ldots, a_n) = (-a_1, -a_2, -a_3, \ldots, -a_n)$
- 2D, 3D, 4D vector negation

  $-(x, y) = (-x, -y)$

  $-(x, y, z) = (-x, -y, -z)$

  $-(x, y, z, w) = (-x, -y, -z, -w)$



Vector-Scalar Multiplication

- Vector scalar multiplication

  $\alpha * (x, y, z) = (\alpha x, \alpha y, \alpha z)$

- Vector scale division

  $1/\alpha * (x, y, z) = (x/\alpha, y/\alpha, z/\alpha)$

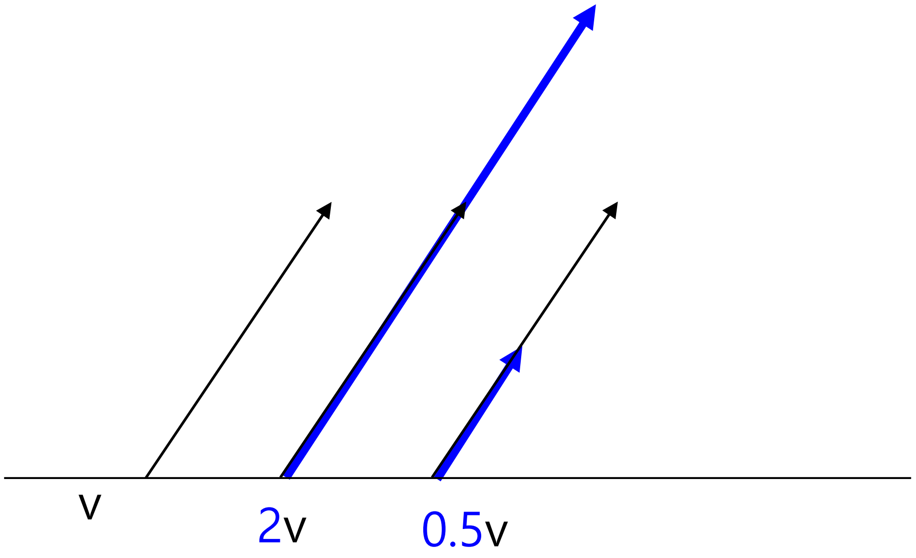
- Example:

  2 * (4, 5, 6) = (8, 10, 12)

  ½ * (4, 5, 6) = (2, 2.5, 3)

  -3 * (-5, 0, 0.4) = (15, 0, -1.2)

  3**u** + **v** = (3**u**) + **v**



Vector Addition and Subtraction

- Vector Addition
  - Defined as a head-to-tail axiom

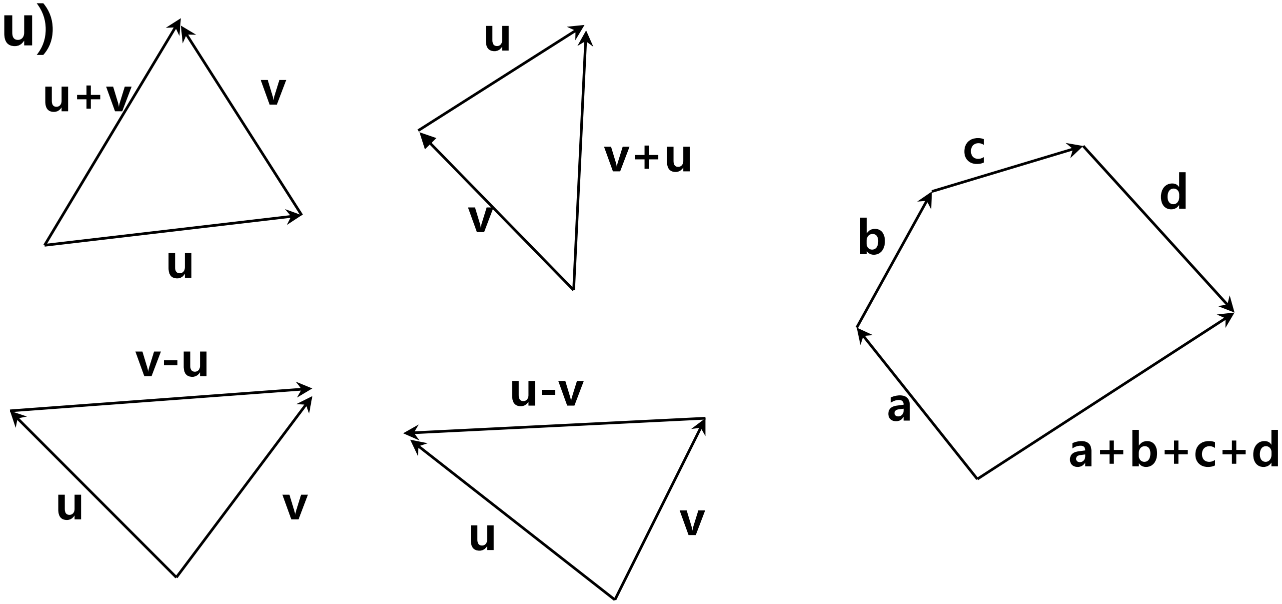
$(x_1, y_1, z_1) + (x_2, y_2, z_2) = (x_1+x_2, y_1+y_2, z_1+z_2)$

$\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$

- Vector Subtraction

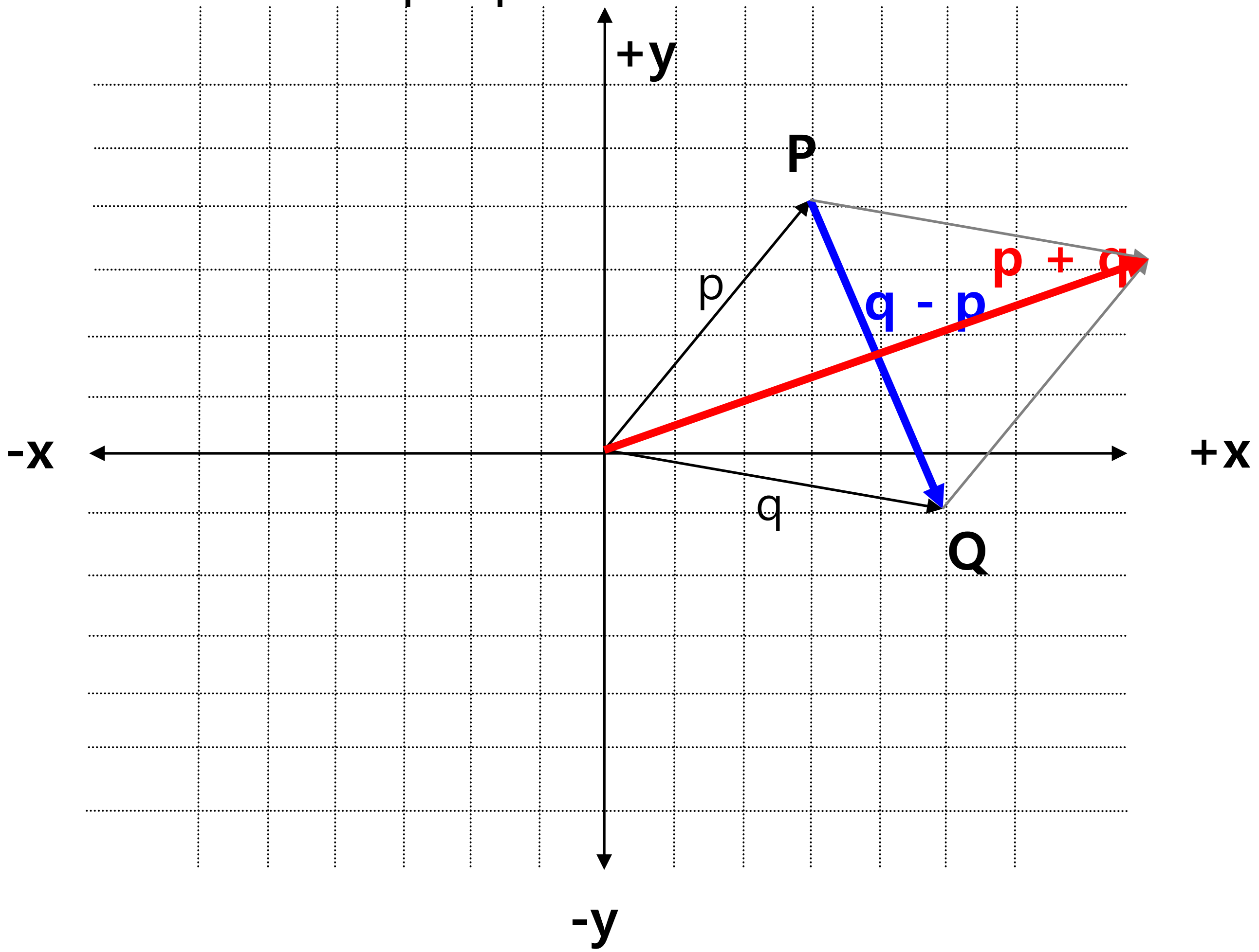
$(x_1, y_1, z_1) - (x_2, y_2, z_2) = (x_1-x_2, y_1-y_2, z_1-z_2)$

$\mathbf{u} - \mathbf{v} = -(\mathbf{v} - \mathbf{u})$



Vector Addition and Subtraction

- The displacement vector from the point P to the point Q is calculated as q – p.



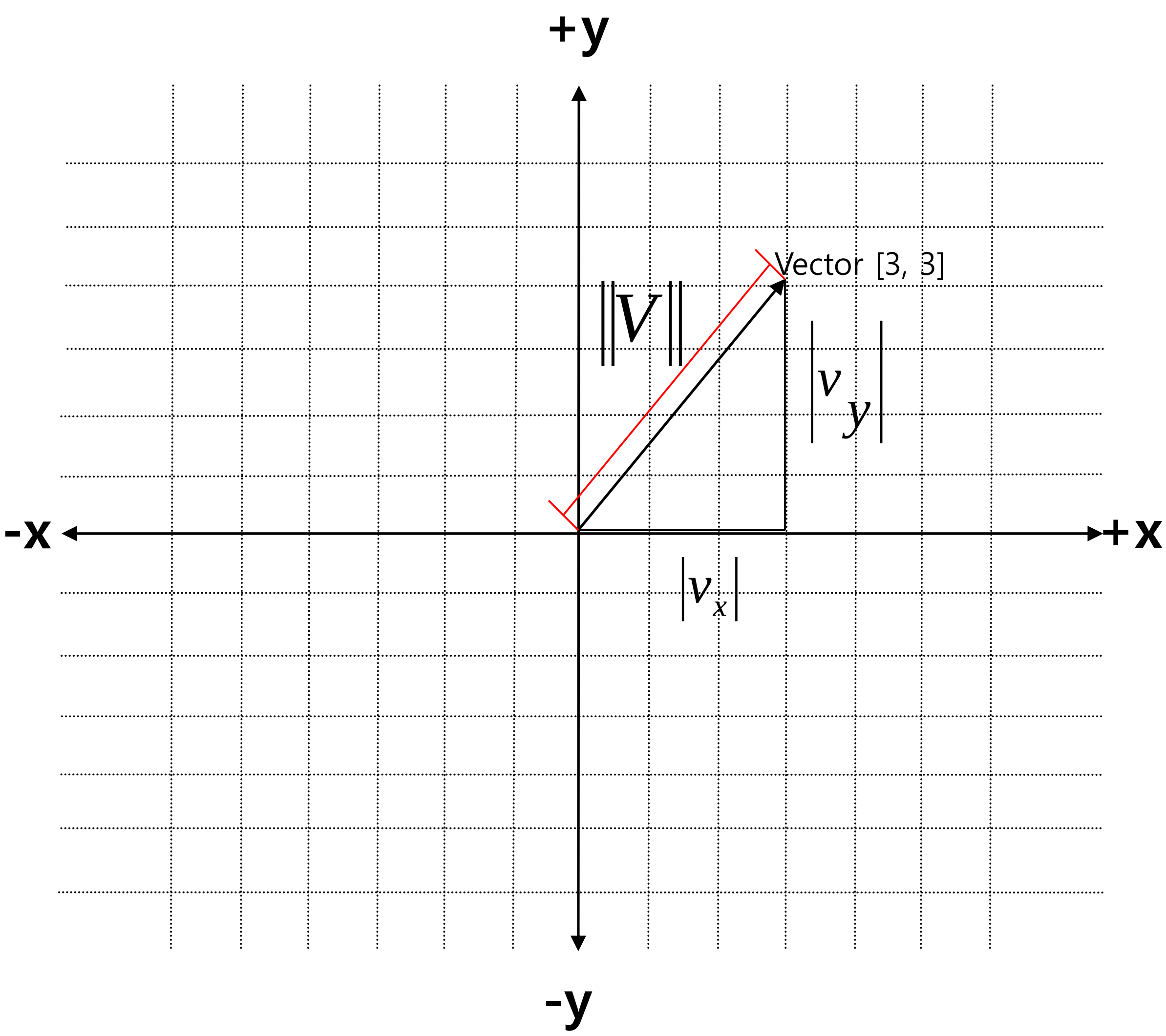
Vector Magnitude (Length)

- Vector magnitude (or length):

Examples: $$\|v\| = \sqrt{v_1^2 + v_2^2 + \ldots + v_{n-1}^2 + v_n^2}$$

$$\begin{aligned}\|(5,-4,7)\| &= \sqrt{5^2 + (-4)^2 + 7^2} \\ &= \sqrt{25+16+49} \\ &= \sqrt{90} \\ &= 3\sqrt{10} \\ &\approx 9.4868\end{aligned}$$

Vector Magnitude



$$\|v\|^2 = |v_x|^2 + |v_y|^2$$

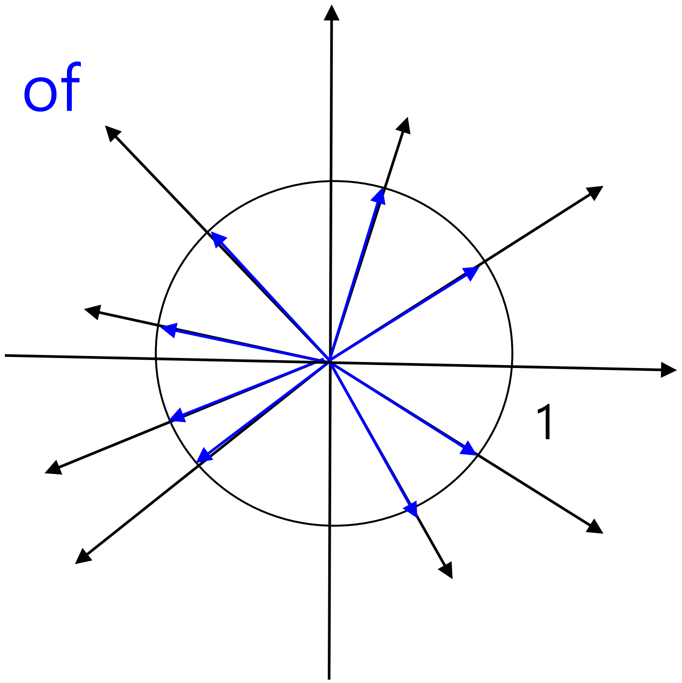
$$\sqrt{\|v\|^2} = \sqrt{v_x^{\ 2} + v_y^{\ 2}}$$

$$\|v\| = \sqrt{v_x^{\ 2} + v_y^{\ 2}}$$

Normalized Vectors

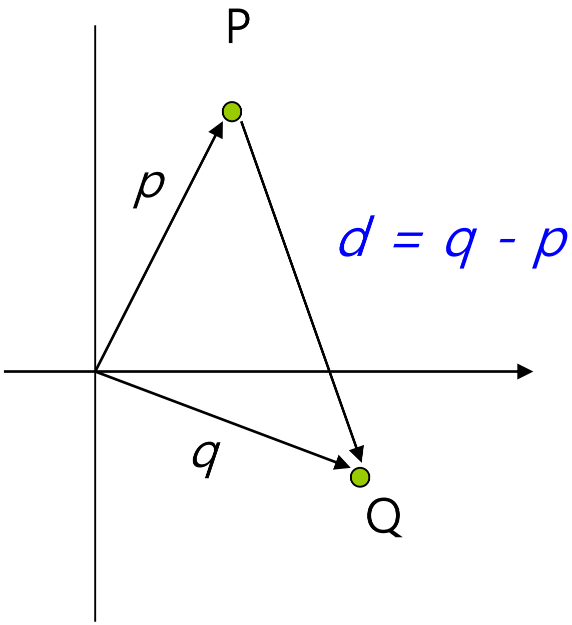
- There is case where you only need the direction of the vector, regardless of the vector length.
- The unit vector has a magnitude of 1.
- The unit vector is also called as *normalized vectors or normal*.
- "Normalizing" a vector:

$$v_{norm} = \frac{v}{\|v\|}, v \neq 0$$



Distance

- The distance between two points P and Q is calculated as follows.
  - Vector *p*
  - Vector *q*
  - Displacement vector $d = q - p$
  - Find the length of the vector d.
  - distance(P, Q) = $\| d \| = \| q - p \|$



Vector Dot Product

- Dot product between two vectors: **u • v**

  $(u_1, u_2, u_3, \ldots, u_n) \cdot (v_1, v_2, v_3, \ldots, v_n) = u_1v_1 + u_2v_2 + \ldots + u_{n-1}v_{n-1} + u_nv_n$

  or

  $$u \cdot v = \sum_{i=1}^{n} u_i v_i$$

  $$u \cdot u = \|u\|^2$$

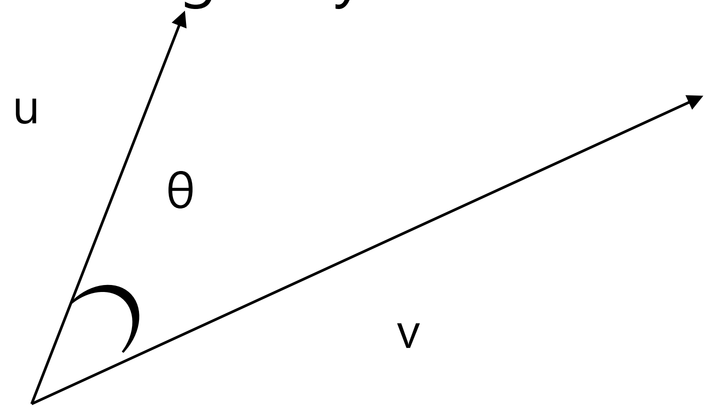
- Example:

  $(4, 6) \cdot (-3, 7) = 4*-3 + 6*7 = 30$

  $(3, -2, 7) \cdot (0, 4, -1) = 3*0 + -2*4 + 7*-1 = -15$

Vector Dot Product

- The dot product of the two vectors is the cosine of the angle between two vectors (assuming they are normalized).



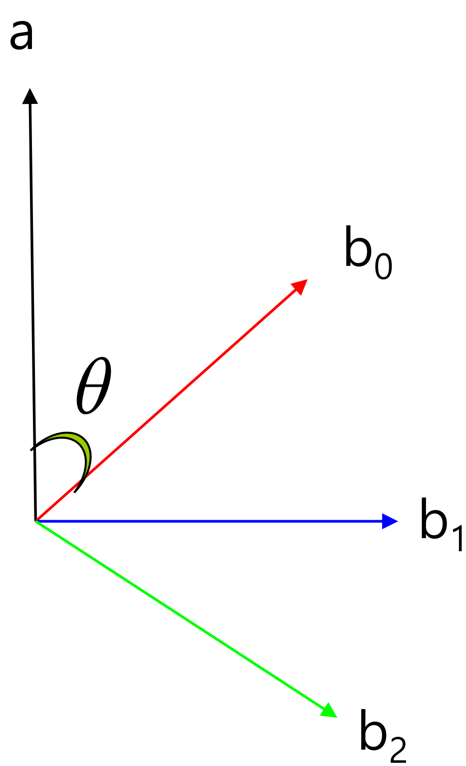
$$u \cdot v = \|u\| \|v\| \cos\theta$$

$$\theta = \mathrm{acos}\left(\frac{u \cdot v}{\|u\| \|v\|}\right)$$

$$\theta = \mathrm{acos}(u \cdot v), \textit{ where } u, v \textit{ are unit vectors}$$

Dot Product as Measurement of Angle

- The following is the characteristics of the dot product.



$$a \cdot b_0 > 0 \ \ when \ \ 0^\circ \le \theta < 90^\circ$$

$$a \cdot b_1 = 0 \ \ when \ \theta = 90^\circ$$

$$a \cdot b_2 < 0 \ \ when \ \ 90^\circ < \theta \le 180^\circ$$

Projecting One Vector onto Another

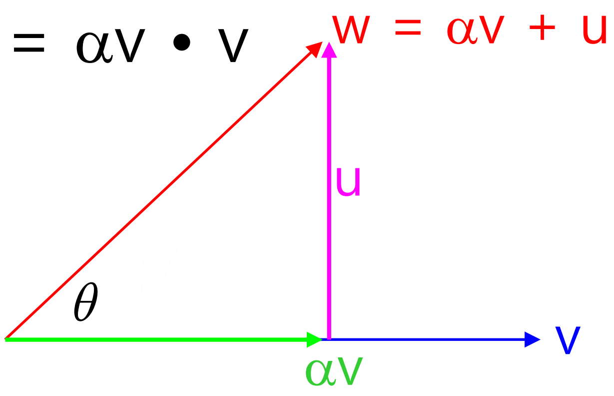
- Given two vectors, w and v, one vector w can be divided into parallel and orthogonal to the other vector v.

$w = w_{par} + w_{per}$

$w = \alpha v + u$

u must be orthogonal to v, $u \bullet v = 0$

$w \bullet v = (\alpha v + u) \bullet v = \alpha v \bullet v + u \bullet v = \alpha v \bullet v$



$$\alpha = \frac{w \cdot v}{v \cdot v}$$

$$u = w - \alpha v = w - \frac{w \cdot v}{v \cdot v} v = w - \frac{w \cdot v}{\|v\|^2} v$$

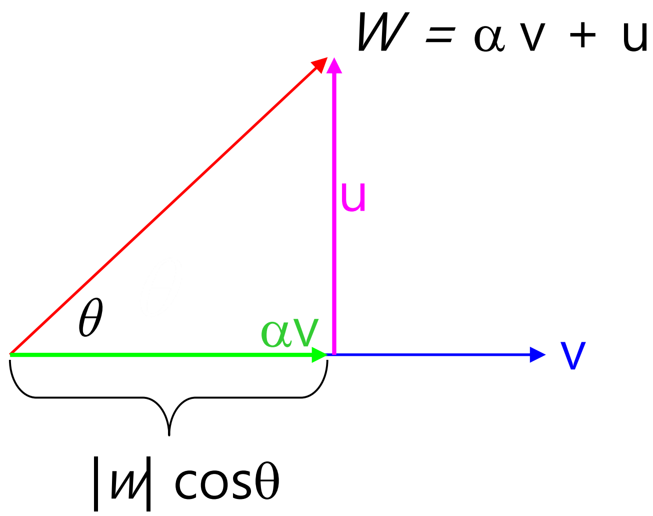
$$\alpha v = w - u = w - w + \frac{w \cdot v}{v \cdot v} v = \frac{w \cdot v}{v \cdot v} v = \frac{w \cdot v}{\|v\|^2} v$$

Projecting One Vector onto Another

If v is a unit vector,
then $\|v\| = 1$

$$w_{per} = u = w - (w \cdot v)v$$

$$w_{par} = \alpha v = (w \cdot v)v$$



$$\cos\theta = \frac{\|\alpha v\|}{\|w\|} \Rightarrow \|\alpha v\| = \|w\| \cos\theta$$

$$\sin\theta = \frac{\|u\|}{\|w\|} \Rightarrow \|u\| = \|w\| \sin\theta$$

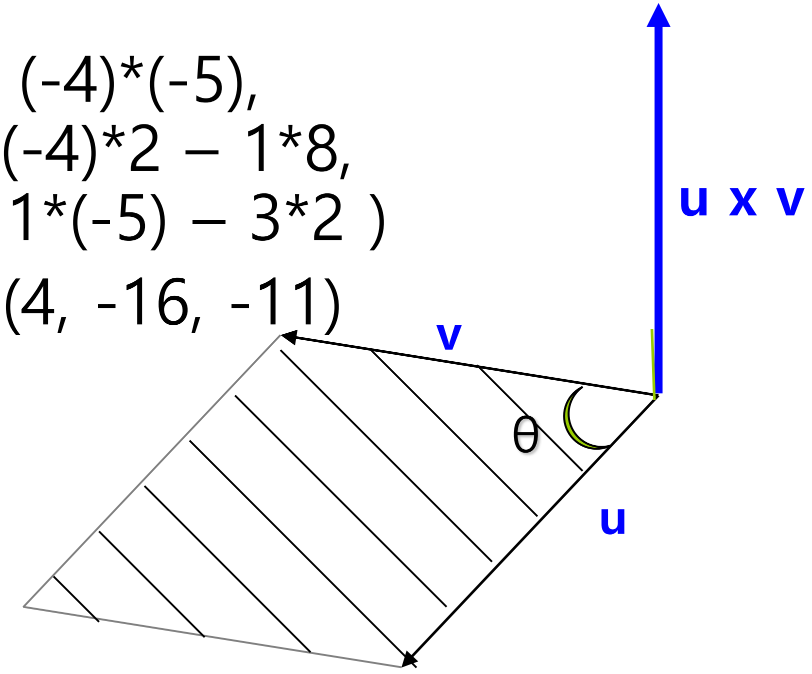
Vector Cross Product

- Cross product: **u x v**

  (x_1, y_1, z_1) **x** $(x_2, y_2, z_2) = (\ y_1 z_2 - z_1 y_2,$
  $z_1 x_2 - x_1 z_2,$
  $x_1 y_2 - y_1 x_2\ )$

- Example:

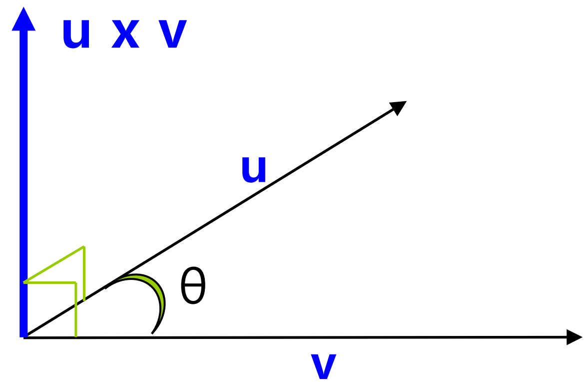
  (1, 3, -4) **x** (2, -5, 8) = (3*8 – (-4)*(-5),
  (-4)*2 – 1*8,
  1*(-5) – 3*2)
  = (4, -16, -11)



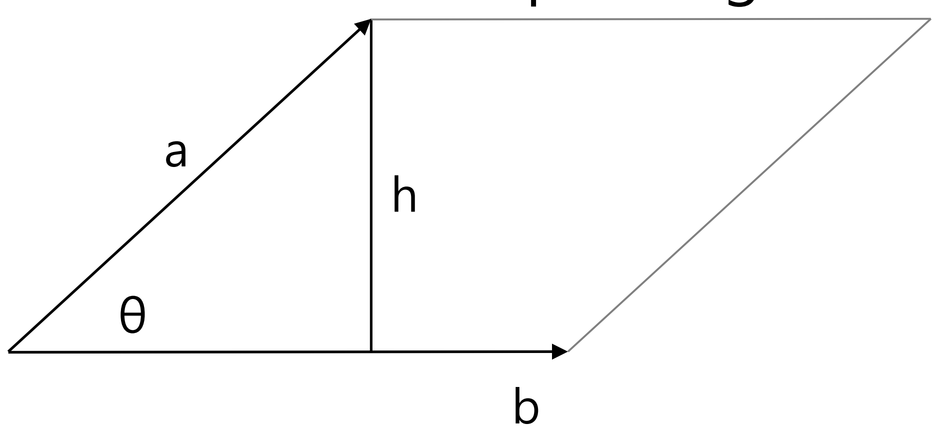
Vector Cross Product

- The magnitude of the cross product between two vectors, |(**u x v**)|, is the product of the magnitude of each other and the sine of the angle between the two vectors.

$$\|u \times v\| = \|u\|\|v\| \sin \theta$$



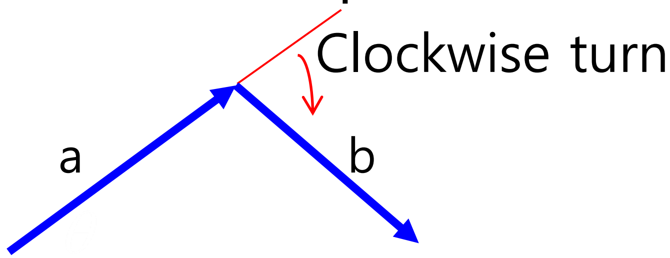
- The area of the parallogram is calculated as *bh*.



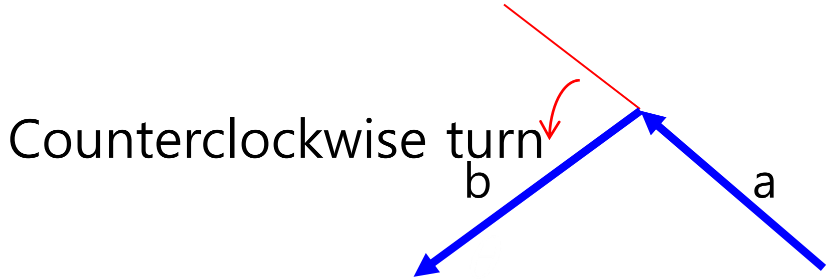
$$\begin{aligned} A &= bh \\ &= b(a \sin \theta) \\ &= \|a\|\|b\| \sin \theta \\ &= \|a \times b\| \end{aligned}$$

Vector Cross Product

- In the left-handed coordinate system, when the vectors u and v move in a clockwise turn, u x v points in the direction toward us, and when moving in a counter-clockwise turn, u x v points in the direction away from us.
- In the right-handed coordinate system, when the vectors u and v move in a counter-clockwise turn, u x v points in the direction toward us, and when moving in a clockwise turn, u x v points in the direction away from us.



Left-handed Coordinates



Right-handed Coordinates

Linear Algebra Identities

Identity	Comments
u + v = v + u	벡터 덧셈 교환법칙
u – v = u + (-v)	벡터 뺄셈
(u+v)+w = u+(v+w)	벡터 덧셈 결합법칙
$\alpha(\beta u)=(\alpha\beta)u$	스칼라-벡터 곱셈 결합법칙
$\alpha(u + v) = \alpha u + \alpha v$ $(\alpha + \beta)u = \alpha u + \beta u$	스칼라-벡터 분배법칙
$\|\alpha v\| = \|\alpha\|\|v\|$	스칼라의 곱
$\|v\| \geq 0$	벡터의 크기는 양수 (nonnegative)
$\|u\|^2 + \|v\|^2 = \|u+v\|^2$	피타고리안 법칙 (Pythagorean theorem)
$\|u\| + \|v\| \geq \|u+v\|$	벡터 덧셈 삼각법칙 (Triangle rule)
u · v = v · u	내적(dot product) 교환법칙
$\|v\| = \sqrt{v \cdot v}$	내적(dot product)을 이용한 벡터의 크기 정의

Linear Algebra Identities

Identity	Comments
$\alpha(u \cdot v) = (\alpha u) \cdot v = u \cdot (\alpha v)$	벡터의 내적과 스칼라 곱 결합법칙
$u \cdot (v + w) = u \cdot v + u \cdot w$	벡터 덧셈/뺄셈과 내적 분배법칙
$u \times u = 0$	벡터 자신의 외적 (cross product)은 0
$u \times v = -(v \times u)$	벡터의 외적은 반교환법칙 (anti-commutative)
$u \times v = (-u) \times (-v)$	벡터의 외적은 각 벡터의 역에 외적과 같다
$\alpha(u \times v) = (\alpha u) \times v = u \times (\alpha v)$	벡터의 외적과 스칼라 곱 결합법칙
$u \times (v+w) = (u \times v) + (u \times w)$	두 벡터의 덧셈과 다른 벡터와의 외적은 분배법칙을 성립
$u \cdot (u \times v) = 0$	Dot product of any vector with cross product of that vector & another vector is 0

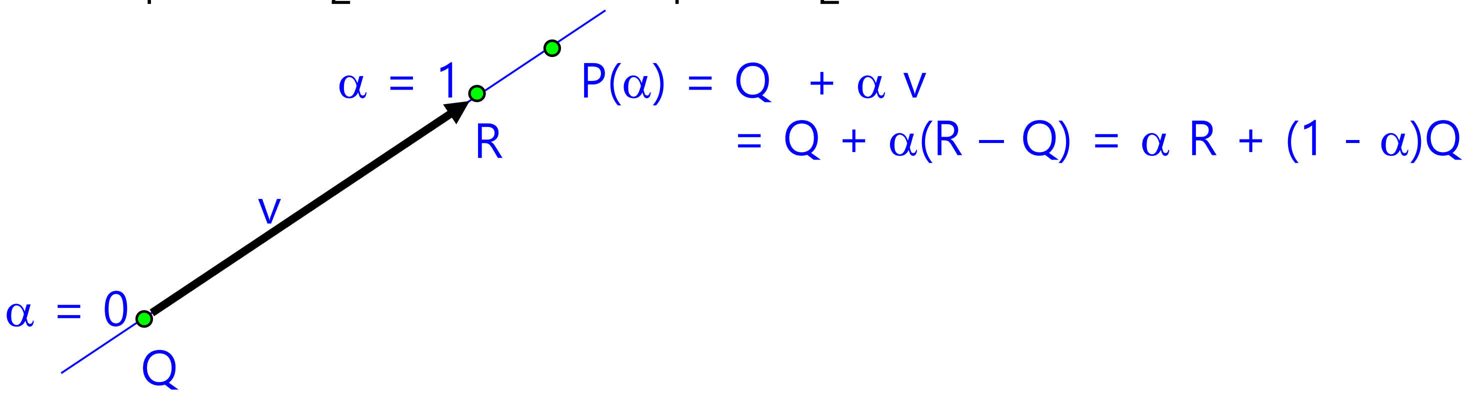
Geometric Objects

- Line
  - 2 points
- Plane
  - 3 points
- 3D objects
  - Defined by a set of triangles
  - Simple convex flat polygons
  - hollow

Lines

- Line is point-vector addition (or subtraction of two points).
- Line parametric form: $P(\alpha) = P_0 + \alpha v$
  - P_0 is arbitrary point, and v is arbitrary vector
  - Points are created on a straight line by changing the parameter.
- $v = R - Q$

  $P = Q + \alpha v = Q + \alpha(R - Q) = \alpha R + (1 - \alpha)Q$
- $P = \alpha_1 R + \alpha_2 Q$ where $\alpha_1 + \alpha_2 = 1$



Lines, Rays, Line Segments

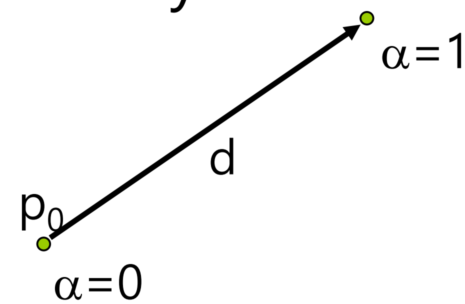
- The line is infinitely long in both directions.
- A line segment is a piece of line between two endpoints. $0 <= \alpha <= 1$
- A ray has one end point and continues infinitely in one direction. $\alpha >= 0$
- Line:

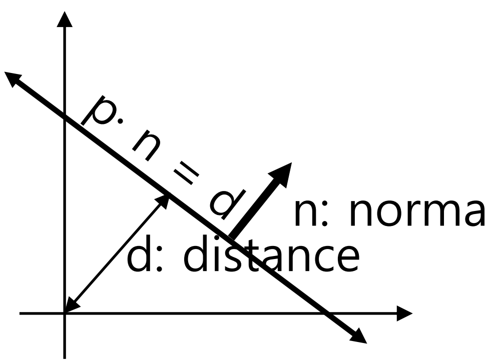
$p(\alpha) = p_0 + \alpha d$ (parametric)

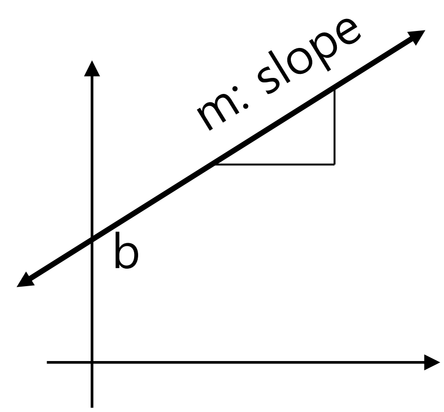
$y = mx + b$ (explicit)

$ax + by = d$ (implicit)

$p \bullet n = d$

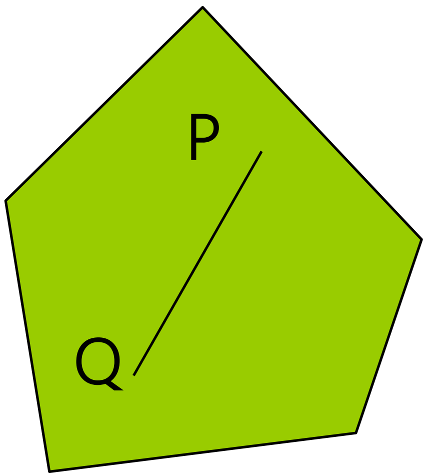




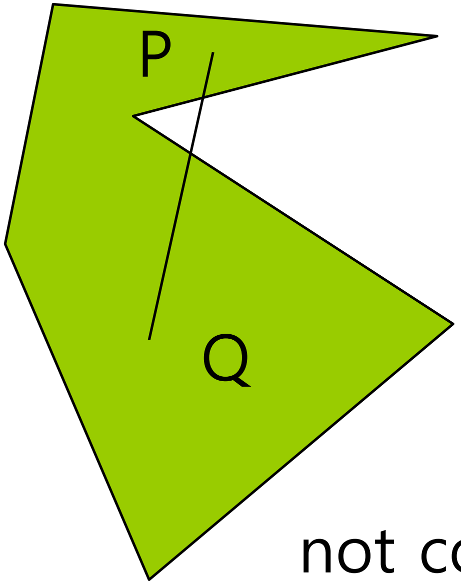


Convexity

- An object is *convex* if only if for any two points in the object all points on the line segment between these points are also in the object.



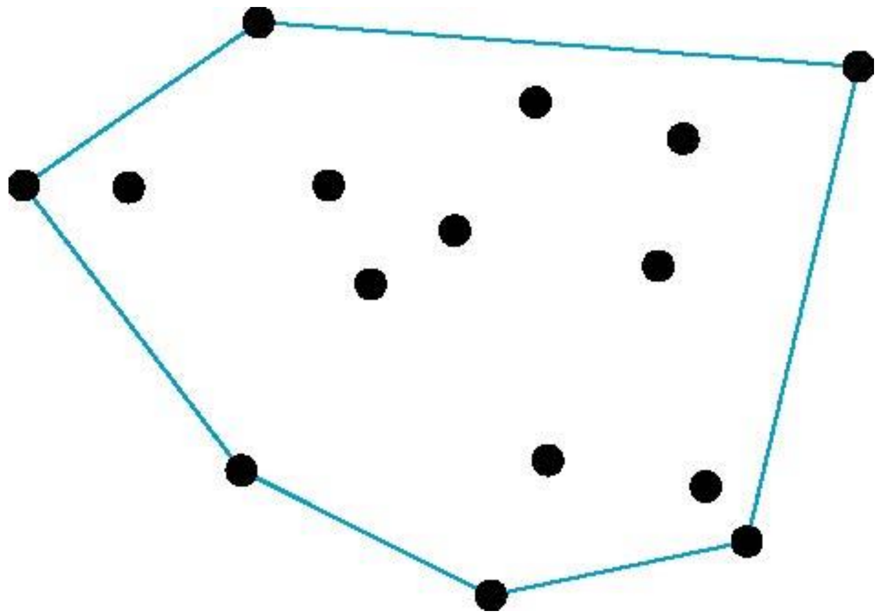
convex



not convex

Convex Hull

- **Smallest convex object** containing $P_1, P_2, \ldots\ldots P_n$
- Formed by "**shrink wrapping**" points



Affine Sums

- The affine sum of the points defined by $P_1, P_2, \ldots P_n$ is

  $$P = \alpha_1 P_1 + \alpha_2 P_2 + \ldots + \alpha_n P_n$$

  Can show by induction that this sum makes sense iff

  $$\alpha_1 + \alpha_2 + \ldots \alpha_n = 1$$

- If, in addition, $\alpha_i >= 0$, i=1,2, ..,n, we have the **convex hull** of $P_1, P_2, \ldots P_n$.
- Convex hull $\{P_1, P_2, \ldots P_n\}$, you can see that it includes all the line segments connecting the pairs of points.

Linear/Affine Combination of Vectors

- Linear combination of m vectors
  - Vector v_1, v_2, .. v_m
  - $w = \alpha_1 v_1 + \alpha_2 v_2 + \ldots \alpha_m v_m$ where α_1, α_2, .. α_m are scalars
- If the sum of the scalar values, α_1, α_2, .. α_m is 1, it becomes an affine combination.
  - $\alpha_1 + \alpha_2 + .. + \alpha_m = 1$

Convex Combination

- If, in addition, $\alpha_i >= 0$, i=1,2, ..,n, we have the **convex hull** of $P_1, P_2, \ldots P_n$.
- Therefore, the linear combination of vectors satisfying the following condition is a convex.

  $\alpha_1 + \alpha_2 + .. + \alpha_m = 1$

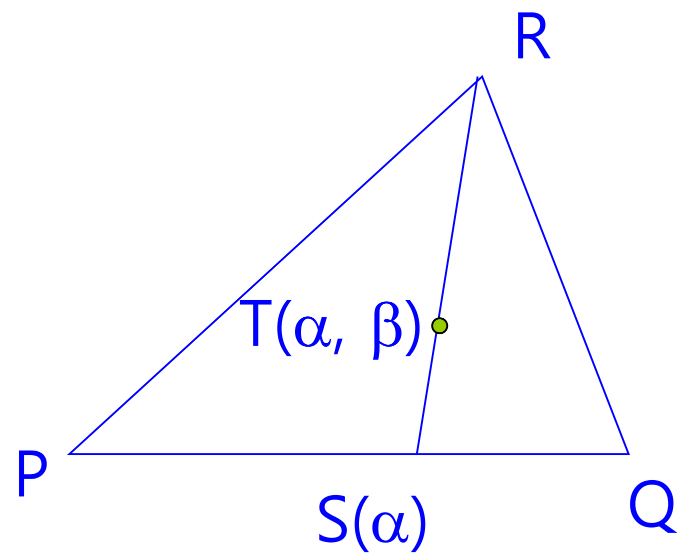
  and

  $\alpha_i \geq 0$ for i=1,2, .. m

  α_i is between 0 and 1
- Convexity
  - Convex hull

Plane

- A plane can be defined by a point and two vectors or by three points.
- Suppose 3 points, P, Q, R
- Line segment PQ
  - $S(\alpha) = \alpha P + (1 - \alpha)Q$
- Line segment SR
  - $T(\beta) = \beta S + (1 - \beta)R$
- Plane defined by P, Q, R
  - $T(\alpha, \beta) = \beta(\alpha P + (1 - \alpha)Q) + (1 - \beta)R$
    $= P + \beta(1 - \alpha)(Q - P) + (1 - \beta)(R - P)$
  - For $0 \le \alpha, \beta \le 1$, we get all points in triangle, $T(\alpha, \beta)$.



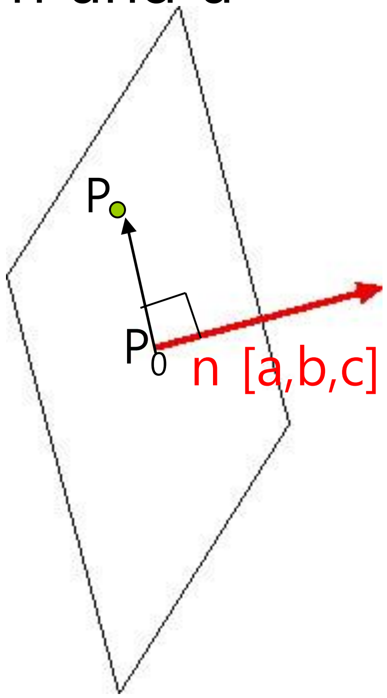
Plane

- Plane equation defined by a point P_0 and two non parallel vectors, u, v
  - $T(\alpha, \beta) = P_0 + \alpha u + \beta v$
  - $P - P_0 = \alpha u + \beta v$ (P is a point on the plane)
- Using n (the cross product of u, v), the plane equation is as follows
  - $n \bullet (P - P_0) = 0$ (where $n = u \times v$ and n is a normal vector)

Plane

- The plane is represented by a normal vector n and a point P_0 on the plane.
  - Plane (n, d) where n (a, b, c)
  - ax + by + cz + d = 0
  - n•p + d = 0

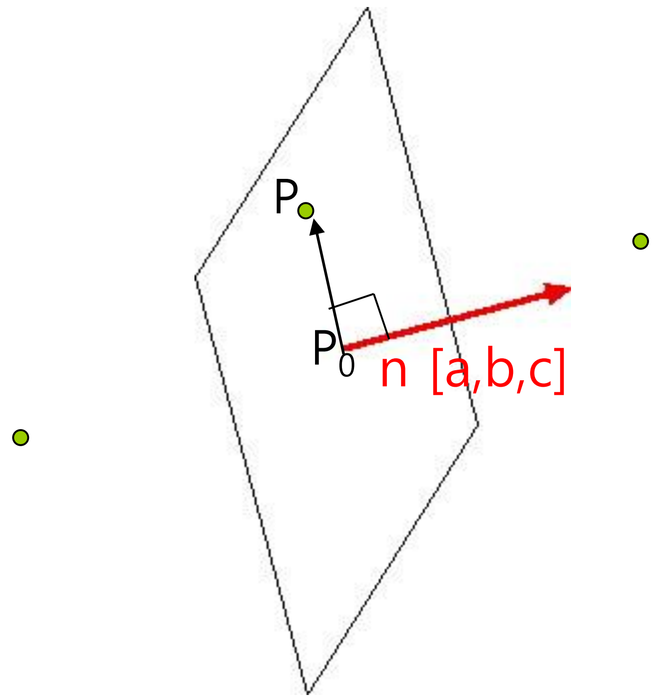
    d = -n•p



- For point p on the plane, n•(p - p_0) =0
- If the plane normal n is a unit vector, then n•p + d gives the shortest signed distance from the plane to point p: d = -n•p

Relationship between Point and Plane

- Relationship between point p and plane (n, d)
  - If n•p + d = 0, then p is in the plane.
  - If n•p + d > 0, then p is outside the plane.
  - If n•p + d < 0, then p is inside the plane.



Plane Normalization

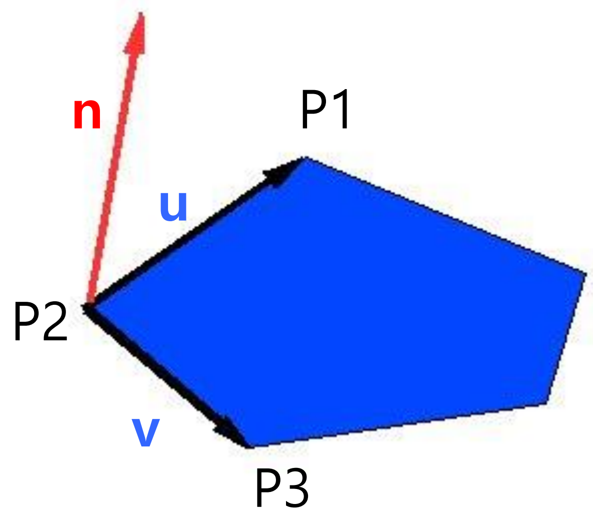
- Plane normalization
  - Normalize the plane normal vector
  - Since the length of the normal vector affects the constant d, d is also normalized.

$$\frac{1}{\|\mathbf{n}\|}(\mathbf{n}, \mathbf{d}) = \left(\frac{n}{\|n\|}, \frac{d}{\|n\|} \right)$$

Computing a Normal from 3 Points in Plane

- Find the normal from the polygon's vertices.
  - The polygon's normal computes two non-collinear edges. (assuming that no two adjacent edges will be collinear)
  - Then, normalize it after the cross product.

```
void computeNormal(vector P1, vector P2, vector P3) {
        vector u, v, n, y(0, 1, 0);
        u = P1 – P2;
        v = P3 – P2;
        n = cross(u, v);
        if (n.length()==0)
                return y;
        else
                return n.normalize();
}
```



Computing a Distance from Point to Plane

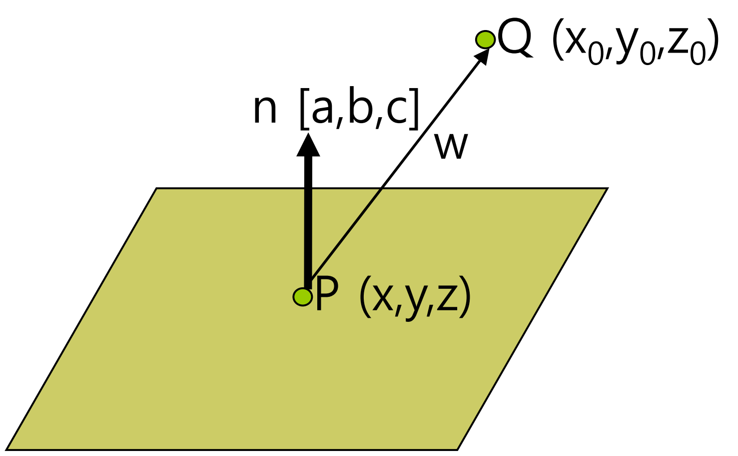
- Find the closest distance to a plane (n, d) in space and a point Q out of the plane.
  - The plane's normal is n, and *D* is the distance between a point P and a point Q on the plane.

$$w = Q - P = [x_0 - x, y_0 - y, z_0 - z]$$

$$D = \frac{|n \bullet w|}{\|n\|}$$

$$= \frac{|a(x_0 - x) + b(y_0 - y) + c(z_0 - z)|}{\sqrt{a^2 + b^2 + c^2}}$$

$$= \frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2}}$$



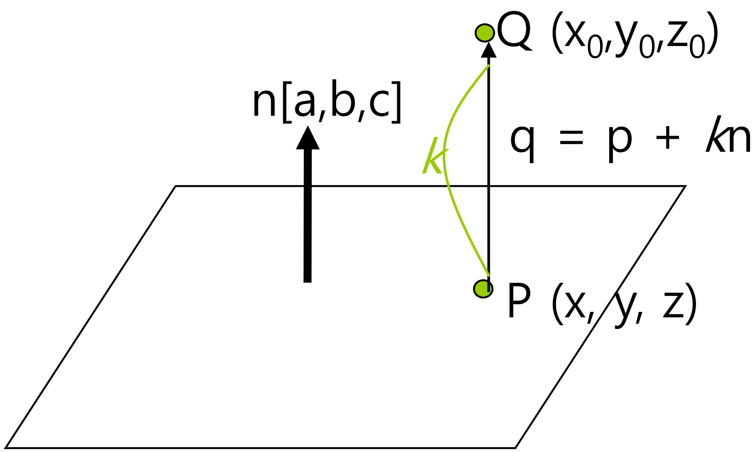
Projecting w onto n : $w_{\|} = n\frac{w \cdot n}{\|n\|^2}$ & $\|w_{\|}\| = \frac{|w \cdot n|}{\|n\|}$

Closest Point on the Plane

- Find a point P on the plane (n, d) closest to one point Q in space.
  - p = q – *k*n (*k* is the shortest signed distance from point Q to the plane)
  - If n is a unit vector,

    k = n•q + d

    p = q – (n•q + d)n



$$\textbf{Distance(q,plane)} = \frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2}}$$

$$where\ q(x_0, y_0, z_0)\ \ and\ \ Plane\ ax + by + cz + d = 0$$

$$\textbf{Distance(q,plane)} = n \cdot q + d\ \ (n\ is\ a\ unit\ vector)$$

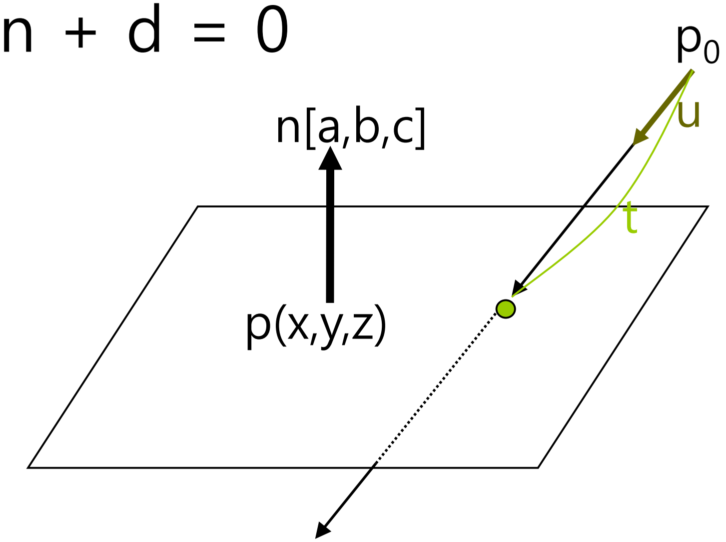
Intersection of Ray and Plane

- Ray **p**(t) = $\mathbf{p}_0$ + t**u** & plane p•n + d = 0
- Ray/Plane intersection:

$$(\mathrm{p_o} + t\mathrm{u}) \cdot \mathrm{n} + d = 0$$

$$t\mathrm{u} \cdot \mathrm{n} = -d - \mathrm{p_o} \cdot \mathrm{n}$$

$$t = \frac{-(\mathrm{p_o} \cdot \mathrm{n} + d)}{\mathrm{u} \cdot \mathrm{n}}$$



- If the ray is parallel to the plane, the denominator **u•n**=0. Thus, the ray does not intersect the plane.
- If the value of t is not in the range [0, ∞), the ray does not intersect the plane.
- $$\mathrm{p}\left(\frac{-(\mathrm{p_o} \cdot \mathrm{n} + d)}{\mathrm{u} \cdot \mathrm{n}}\right) = \mathrm{p_o} + \frac{-(\mathrm{p_o} \cdot \mathrm{n} + d)}{\mathrm{u} \cdot \mathrm{n}}\mathrm{u}$$

Matrix

- Matrix M (***r*** x ***c*** matrix)
  - **Row** of horizontally arranged matrix elements
  - **Column** of vertically arranged matrix elements
  - Mij is the **element** in row i and column j

$$\boldsymbol{M} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$$

r(2) rows

c(2) columns

Matrix

2x5 matrix

2	-4	7	7⁄8	8
-3	4	3⁄8	0	1

$m_{12} = -4$

4x3 matrix

4	0	12
-5	4	3
12	3⁄8	-1
1/2	18	0

$m_{42} = 18$

Mij is the **element** in row i and column j

Square Matrix

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{33} & m_{33} \end{pmatrix}$$

Nondiagonal elements

Diagonal elements

- The n x n matrix is called an n-th square matrix. e.g. 2x2, 3x3, 4x4
- Diagonal elements vs. Non-diagonal elements

Identity Matrix

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- The identity matrix is expressed as I.
- All of the diagonals are 1, the remaining elements are 0 in ***n*** x ***n*** square matrix.
- M I = I M = M

Vectors as Matrices

- The n-dimension vector is expressed as a 1x***n*** matrix or an ***n***x1 matrix.
  - 1xn matrix is a row vector (also called a row matrix)
  - nx1 matrix is a column vector (also called a column matrix)

$$\mathbf{A} = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix} \qquad \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \end{bmatrix}$$

Transpose Matrix

- Transpose of M (rxc matrix) is denoted by M^T and is converted to cxr matrix.
  - $M^T_{ij} = M_{ji}$
  - $(M^T)^T = M$
  - $D^T = D$ for any diagonal matrix D.

$$\begin{pmatrix} a & m & c \\ d & e & f \\ g & h & i \end{pmatrix}^T = \begin{pmatrix} a & d & g \\ m & e & h \\ c & f & i \end{pmatrix}$$

Transposing Matrix

$$\begin{pmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{pmatrix}^{T} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}^{T} = \begin{bmatrix} x & y & z \end{bmatrix}$$

Matrix Scalar Multiplication

- Multiplying a matrix **M** with a scalar α = α **M**

$$\alpha \mathbf{M} = \alpha \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{33} & m_{33} \end{pmatrix} = \begin{pmatrix} \alpha m_{11} & \alpha m_{12} & \alpha m_{13} \\ \alpha m_{21} & \alpha m_{22} & \alpha m_{23} \\ \alpha m_{31} & \alpha m_{33} & \alpha m_{33} \end{pmatrix}$$

Two Matrices Addition

- Matrix C is the addition of A (r x c matrix) and B (r x c matrix), which is a r x c matrix.
- Each element $\mathbf{c_{ij}}$ is the sum of the ijth element of A and the ijth element of B.
- $c_{ij} = a_{ij} + b_{ij}$

1+3

$$\begin{pmatrix} 1 & 3 & 6 \\ 10 & 0 & -5 \\ 4 & 7 & 2 \end{pmatrix} + \begin{pmatrix} 3 & 7 & 1 \\ 6 & 4 & 9 \\ 8 & -9 & 4 \end{pmatrix} = \begin{pmatrix} 4 & 10 & 7 \\ 16 & 4 & 4 \\ 12 & -2 & 6 \end{pmatrix}$$

r* x *c ***r* x *c*** ***r* x *c***

Two Matrices Multiplication

- Matrix C(rxc matrix) is the product of A (rxn matrix) and B (nxc matrix).
- Each element $\mathbf{c}_{ij}$ is the vector dot product of the i^{th} row of A and the j^{th} column of B.
- $$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

3+18+48

$$\begin{pmatrix} 1 & 3 & 6 \\ 10 & 0 & -5 \\ 4 & 7 & 2 \end{pmatrix} * \begin{pmatrix} 3 & 7 & 1 \\ 6 & 4 & 9 \\ 8 & -9 & 4 \end{pmatrix} = \begin{pmatrix} 69 & -35 & 52 \\ -10 & 115 & -10 \\ 70 & 38 & 75 \end{pmatrix}$$

r* x *n ***n* x *c*** ***r* x *c***

must match

columns in result

rows in result

Multiplying Two Matrices

$$\begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{pmatrix} \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} \end{pmatrix}$$

$$c_{24} = a_{21}m_{14} + a_{22}m_{24}$$

Matrix Operation

- MI = IM = M (I is identity matrix)
- A + B = B + A : matrix addition commutative law
- A + (B + C) = (A + B) + C : matrix addition associative law
- AB $\neq$BA : Not hold matrix product commutative law
- (AB)C = A(BC) : matrix product associative law
- ABCDEF = ((((AB)C)D)E)F = A((((BC)D)E)F) = (AB)(CD)(EF)
- α(AB) = (αA)B = A(αB) : Scalar-matrix product
- $\alpha(\beta A) = (\alpha\beta)A$
- (vA)B = v (AB)
- $(AB)^T = B^T A^T$
- $(M_1M_2M_3 \ldots M_{n-1}M_n)^T = M_n^TM_{n-1}^T \ldots M_3^TM_2^TM_1^T$

Matrix Determinant

- The determinant of a square matrix M is denoted by **|M|** or "**det M**".
- The determinant of non-square matrix is not defined.

$$|M| = \begin{vmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{vmatrix} = m_{11}m_{22} - m_{12}m_{21}$$

$$|M| = \begin{vmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{vmatrix} = \begin{aligned} & m_{11}(m_{22}m_{33} - m_{23}m_{32}) + \\ & m_{12}(m_{23}m_{31} - m_{21}m_{33}) + \\ & m_{13}(m_{21}m_{32} - m_{22}m_{31}) \end{aligned}$$

Inverse Matrix

- Inverse of M (square matrix) is denoted by M^{-1}.
- $$M^{-1} = \frac{adjM}{|M|}$$
- $(M^{-1})^{-1} = M$
- $M(M^{-1}) = M^{-1}M = I$
- The determinant of a non-singular matrix (i.e, invertible) is nonzero.
- The *adjoint* of M, denoted **"adj M"** is **the transpose of the matrix of cofactors**.

$$\text{adjM} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}^T$$

Cofactor of a Square Matrix & Computing Determinant using Cofactor

- *Cofactor* of a square matrix *M* at a given row and column is the signed determinant of the corresponding *Minor* of M.
- $C_{ij} = (-1)^{ij} \mid M^{\{ij\}} \mid$
- Calculation of n x n determinant using cofactor:

$$|M| = \sum_{j=1}^{n} m_{ij} c_{ij} = \sum_{j=1}^{n} m_{ij} (-1)^{i+j} \left| M^{\{ij\}} \right|$$

$$|M| = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix} = m_{11} \begin{pmatrix} m_{22} & m_{23} & m_{24} \\ m_{32} & m_{33} & m_{34} \\ m_{42} & m_{43} & m_{44} \end{pmatrix} - m_{12} \, |M^{\{12\}}| + m_{13} \, |M^{\{13\}}| - m_{14} \, |M^{\{14\}}|$$

Minor of a Matrix

- The submatrix $M^{\{ij\}}$ is known as a minor of M, obtained by deleting row *i* and column *j* from M.

$$M = \begin{pmatrix} -4 & -3 & 3 \\ 0 & 2 & -2 \\ 1 & 4 & -1 \end{pmatrix} \quad M^{\{12\}} = \begin{pmatrix} 0 & -2 \\ 1 & -1 \end{pmatrix}$$

Determinant, Cofactor, Inverse Matrix

$$M = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}$$

$$\det M = m_{11}m_{22} - m_{12}m_{21}$$

$$C = \begin{pmatrix} m_{22} & -m_{21} \\ -m_{12} & m_{11} \end{pmatrix}$$

$$adj\, M = \begin{pmatrix} m_{22} & -m_{12} \\ -m_{21} & m_{11} \end{pmatrix}$$

$$M^{-1} = \frac{1}{\det M} \begin{pmatrix} m_{22} & -m_{12} \\ -m_{21} & m_{11} \end{pmatrix}$$

Determinant, Cofactor, Inverse Matrix

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}$$

$$\begin{aligned}\det M = m_{11}(m_{22}m_{33} - m_{23}m_{32}) \\ - m_{12}(m_{21}m_{33} - m_{23}m_{31}) \\ + m_{13}(m_{21}m_{32} - m_{22}m_{31})\end{aligned}$$

$$C = \begin{pmatrix} (m_{22}m_{33} - m_{23}m_{32}) & -(m_{21}m_{33} - m_{23}m_{31}) & (m_{21}m_{32} - m_{22}m_{31}) \\ -(m_{12}m_{33} - m_{13}m_{32}) & (m_{11}m_{33} - m_{13}m_{31}) & -(m_{11}m_{32} - m_{21}m_{31}) \\ (m_{12}m_{23} - m_{22}m_{13}) & -(m_{11}m_{23} - m_{13}m_{21}) & (m_{11}m_{22} - m_{12}m_{21}) \end{pmatrix}$$

$$adj M = \begin{pmatrix} (m_{22}m_{33} - m_{23}m_{32}) & -(m_{12}m_{33} - m_{13}m_{32}) & (m_{12}m_{23} - m_{22}m_{13}) \\ -(m_{21}m_{33} - m_{23}m_{31}) & (m_{11}m_{33} - m_{13}m_{31}) & -(m_{11}m_{23} - m_{13}m_{21}) \\ (m_{21}m_{32} - m_{22}m_{31}) & -(m_{11}m_{32} - m_{21}m_{31}) & (m_{11}m_{22} - m_{12}m_{21}) \end{pmatrix}$$

$$M^{-1} = \frac{adj M}{\det M}$$

Multiplying a Vector and a Matrix

$$\begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} p_x & p_y & p_z \\ q_x & q_y & q_z \\ r_x & r_y & r_z \end{pmatrix}$$

$$= \begin{pmatrix} x\mathbf{p}_x + y\mathbf{q}_x + z\mathbf{r}_x & x\mathbf{p}_y + y\mathbf{q}_y + z\mathbf{r}_y & x\mathbf{p}_z + y\mathbf{q}_z + z\mathbf{r}_z \end{pmatrix}$$

$$= x\mathbf{p} + y\mathbf{q} + z\mathbf{r}$$

- A coordinate space transformation can be expressed using a vector-matrix product.

  uM = v // matrix M converts vector u to vector v

Multiplying a Vector and a Matrix

- Vector-matrix multiplication in Unity (Column-Major Order)

  v = M * u // matrix M converts vector u to vector v

$$\mathbf{v} = \mathbf{M} * \mathbf{u}$$

$$\begin{pmatrix} x m_{11} + y m_{12} + z m_{13} \\ x m_{21} + y m_{22} + z m_{23} \\ x m_{31} + y m_{32} + z m_{33} \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Mathf Class

- Mathf
  - Unity's Mathf class provides a collection of common math functions, including trigonometric, logarithmic, etc.
  - Trigonometric (work in **radians**)
    - Sin, Cos, Tan, Asin, Acos, Atan, Atan2
  - Powers and Square Roots
    - Pow, Sqrt, Exp, ClosestPowerOfTwo, NextPowerOfTwo, IsPowerOfTwo
  - Interpolation
    - Lerp, LerpAngle, LerpUnclamped, InverseLerp, MoveTowards, MoveTowardsAngle, SmoothDamp, SmoothDampAngle, SmoothStep
  - Limiting and repeating values
    - Max, Min, Repeat, PingPong, Clamp, Clamp01, Ceil, Floor
  - Logarithmic
    - Log

Vector3 Struct

- Vector3
  - Representation of 3D vectors and points.
  - This **structure** is used throughout Unity to pass **3D positions and directions** around. It also contains functions for doing common vector operations.
  - The Quaternion and the Matrix4x4 classes are useful for rotating or transforming vectors and points.

C# struct is the value type (allocated on the stack)

Matrix4x4 Struct

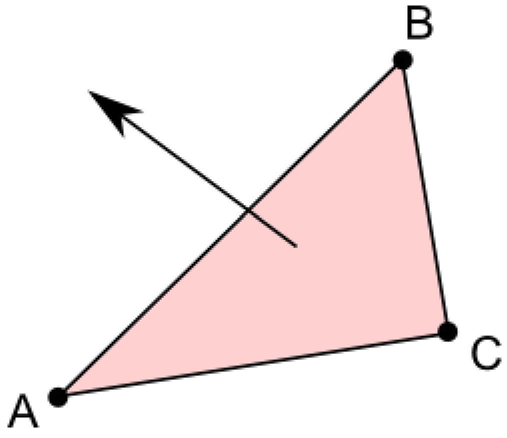
Matrix4x4

- A standard 4x4 transformation matrix. Matrix4x4 is struct
- A transformation matrix can perform arbitrary linear 3D transformations (i.e. translation, rotation, scale, shear etc.) and perspective transformations using homogenous coordinates.
- *You rarely use matrices in scripts*; most often using Vector3, Quaternions, and functionality of Transform class is more straightforward.
- In Unity, Matrix4x4 is used by several Transform, Camera, Material and GL functions.
- Matrices in unity are **column major**.

C# struct is the value type (allocated on the stack)

Plane Struct

- Plane
  - Representation of a plane in 3D space.
  - A plane can also be defined by the three corner points of a triangle that lies within the plane. In this case, the normal vector points toward you if the corner points go around **clockwise** as you look at the triangle face-on.



C# struct is the value type (allocated on the stack)

Quaternion Struct

- Quaternion
  - Quaternions are used to represent rotations.
  - The Quaternion functions that you use 99% of the time are:
    - Quaternion.LookRotation
    - Quaternion.Angle
    - Quaternion.Euler
    - Quaternion.Slerp
    - Quaternion.FromToRotation
    - Quaternion.identity

C# struct is the value type (allocated on the stack)