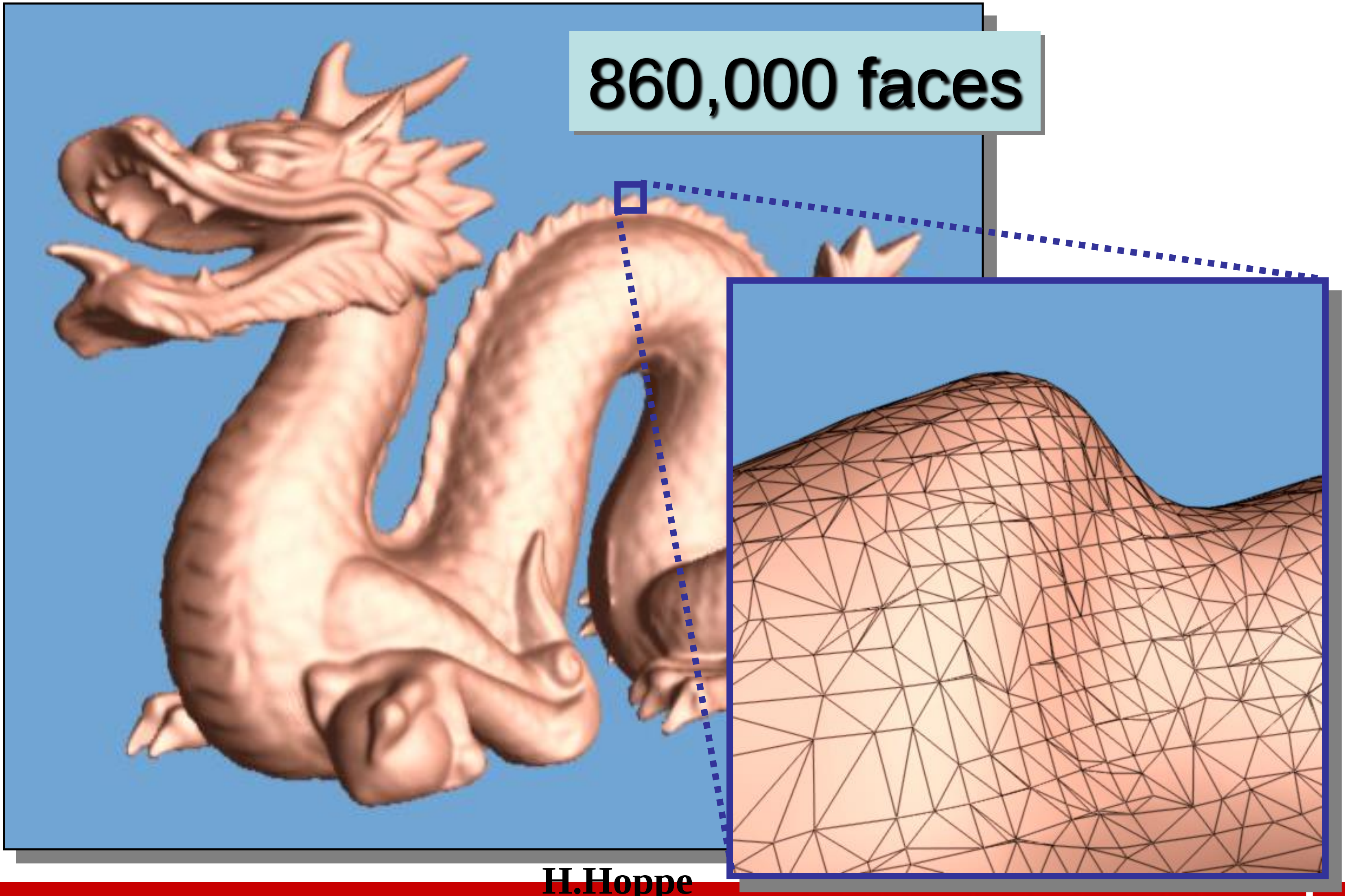
Geometry for Computer Graphics

Fondamenti di Computer Graphics 2018/2019
Serena Morigi
serena.morigi@unibo.it
http://www.dm.unibo.it/~morigi/

https://www.youtube.com/watch?v=vgnZ4PcqV98

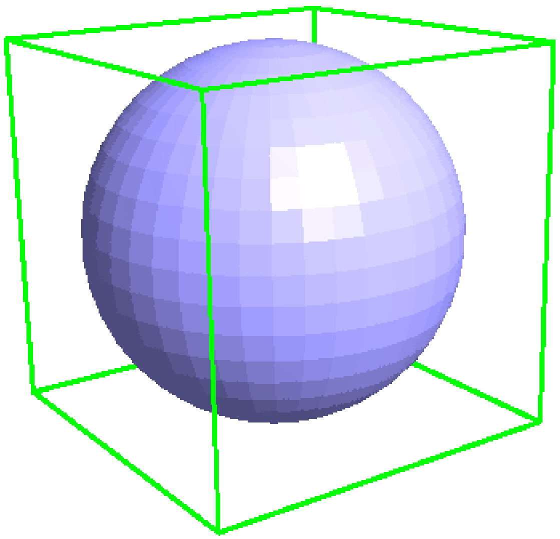


H.Hoppe

Geometric Objects and transformations

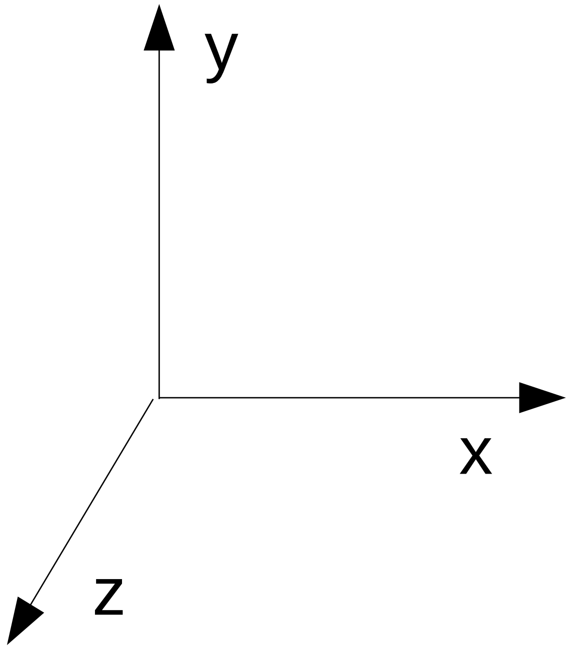
- We represent objects using mainly linear primitives:
  - points
  - lines, segments
  - planes, polygons

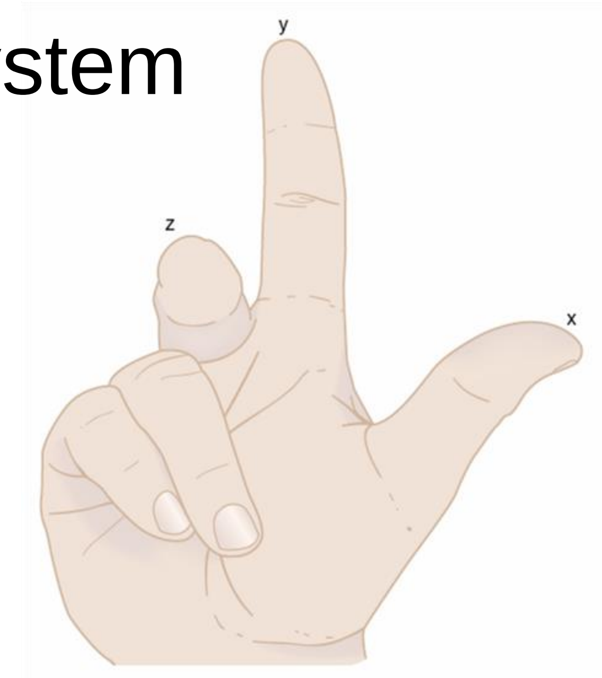
- Need to know how to transform objects, compute distances, change coordinate systems…



Coordinate Systems

- Right handed coordinate system

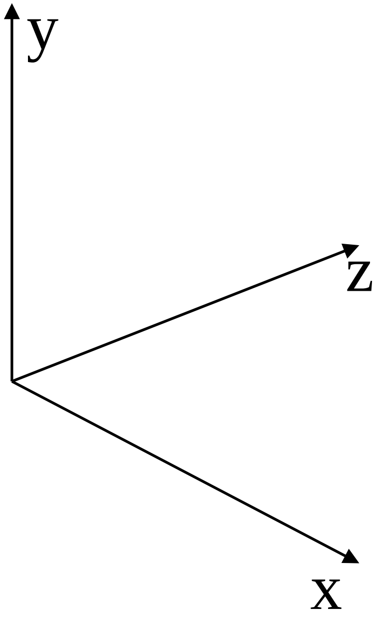




In CG we use the rule:
z axis perpendicular to the screen

Coordinate Systems

- Left handed coordinate system



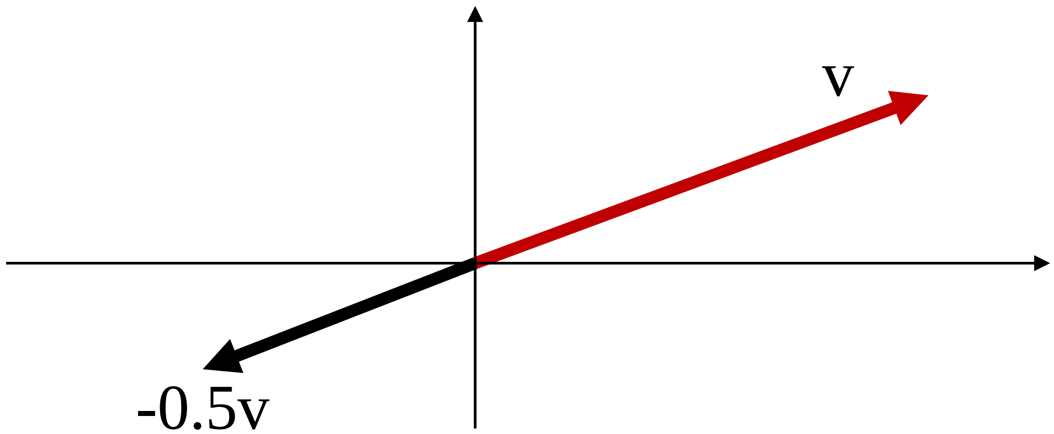
Scalars, Points and Vectors

- **Scalar** specifies ***magnitude*** (quantità)
- **Point** specifies ***location*** in space (or in the plane)
- **Vector** is a quantity with two attributes: ***magnitude*** and ***direction*** (direzione+verso). No location in space.

$$\boxed{\text{Points} \neq \text{Vectors}}$$

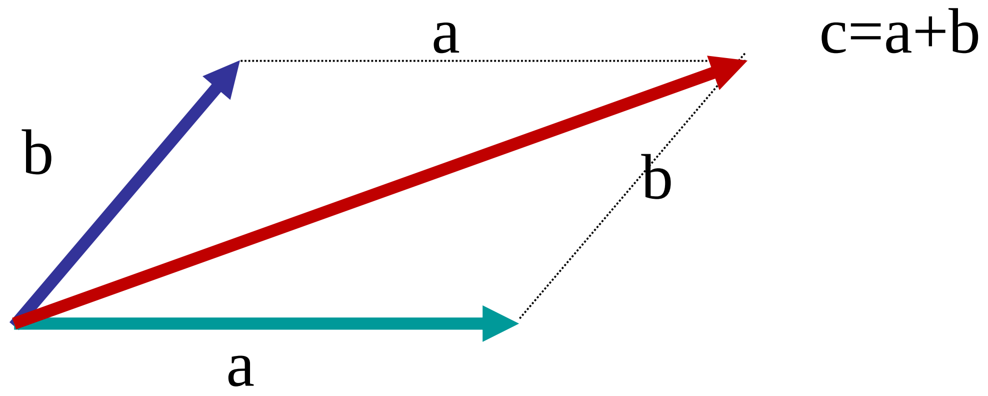
Linear/Vector Space

- Entities: VECTORS and SCALARS
- Operations:
  - Multiplication   scalar vector
  - Vector sum



vector + vector = vector

Parallelogram rule



$$\mathbf{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} b_x & b_y & b_z \end{bmatrix}$$

$$\mathbf{a} + \mathbf{b} = \begin{bmatrix} a_x + b_x & a_y + b_y & a_z + b_z \end{bmatrix}$$

$$\mathbf{a} - \mathbf{b} = \begin{bmatrix} a_x - b_x & a_y - b_y & a_z - b_z \end{bmatrix}$$

$$-\mathbf{a} = \begin{bmatrix} -a_x & -a_y & -a_z \end{bmatrix}$$

$$s\mathbf{a} = \begin{bmatrix} sa_x & sa_y & sa_z \end{bmatrix}$$

Space Dimension

- In a vector space V, the maximum number of linearly independent vectors is fixed and is called the *dimension* of the space
- In an *n*-dimensional space, any set of n linearly independent vectors form a *basis* for the space
- Given a **basis** $a_1, a_2, \ldots, a_n$, any vector v in V can be written as the *linear combination*

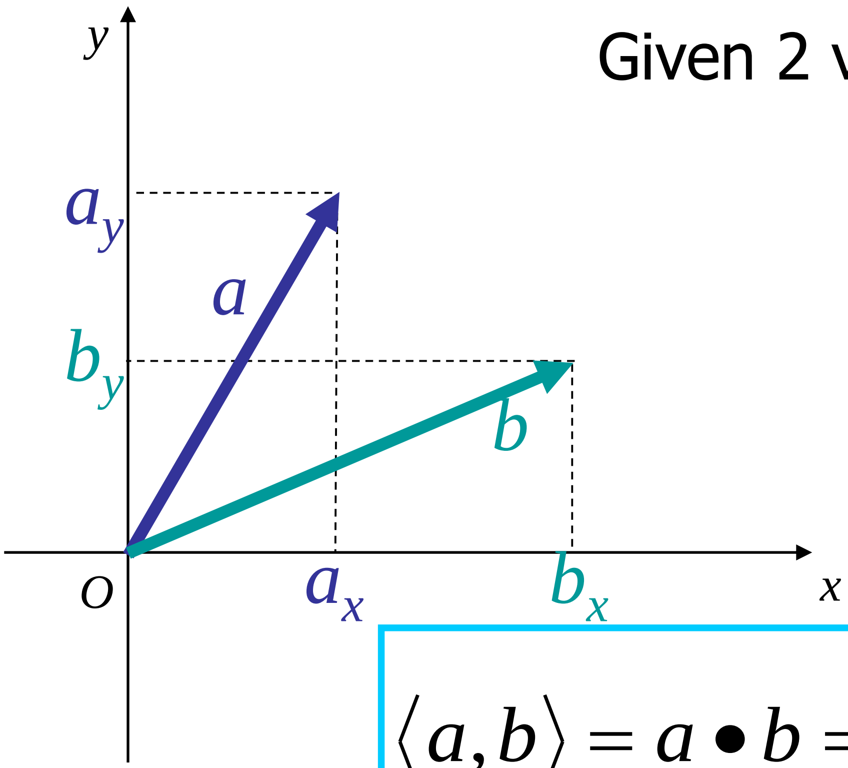
$$v = \alpha_1 a_1 + \alpha_2 a_2 + \ldots + \alpha_n a_n$$

where the $\{\alpha_i\}$ are unique

In $V=R^3$, the Euclidean vectors a_1, a_2, a_3 define a ***coordinate system***

Linear Space with **Dot Product** (or Inner Product)

Dot product in coordinates x,y:



Given 2 vectors get a scalar value

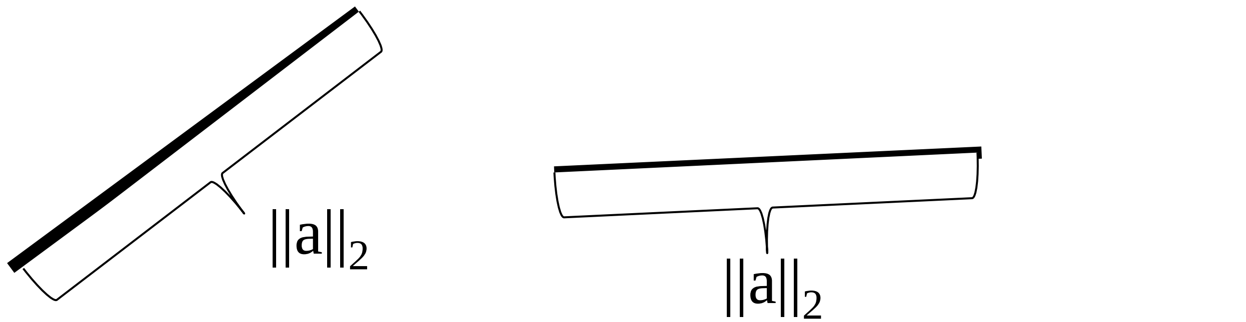
$$a = (a_x, a_y)$$
$$b = (b_x, b_y)$$
$$< a, b > = a_x b_x + a_y b_y$$

$$\langle a, b \rangle = a \bullet b = a^T b = \sum_{i=1}^{n} a_i b_i \quad in \quad \Re^n$$

2-norm or Euclidean norm

$$\|a\|_2 = \sqrt{\langle a, a \rangle} = \sqrt{a^T a} = \sqrt{\sum_{i=1}^{n} (a_i)^2}$$

Euclidean norm is a notion of length preserved by rotations/translations/reflections of space.



Vector Magnitude

- The magnitude (length) of a vector a in R^3 is:

$$\|a\|_2 = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

- A vector with length=1.0 is called a *unit vector*
- We can also *normalize* a vector to make it a unit vector:

$$\frac{a}{\|a\|_2}$$

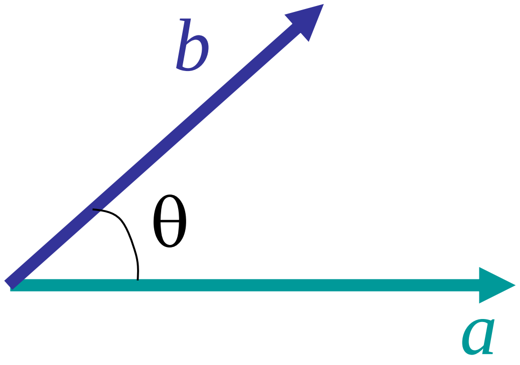
Inner (dot) product

Angle between 2 vectors

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos\theta$$

$$\cos\theta = \left(\frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} \right)$$

$$\theta = \cos^{-1}\left(\frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} \right)$$



Properties

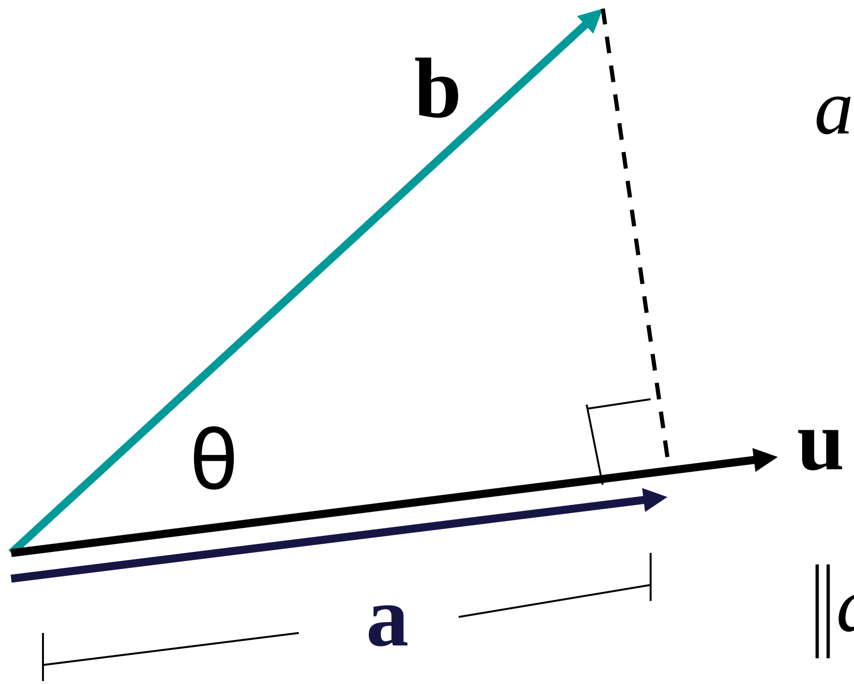
Commutative
<a,b>=<b,a>

Distributive
<a,b+c>=<a,b> + <a,c>

Bilinear, r scalar
<a,*r* b+c>=*r*<a,b>+<a,c>

Vector Projection (orthogonal projection)

- If $||\mathbf{u}||\neq 1.0$ then **||a||** is the length of the vector **a** which is the *projection* of **b** onto **u**



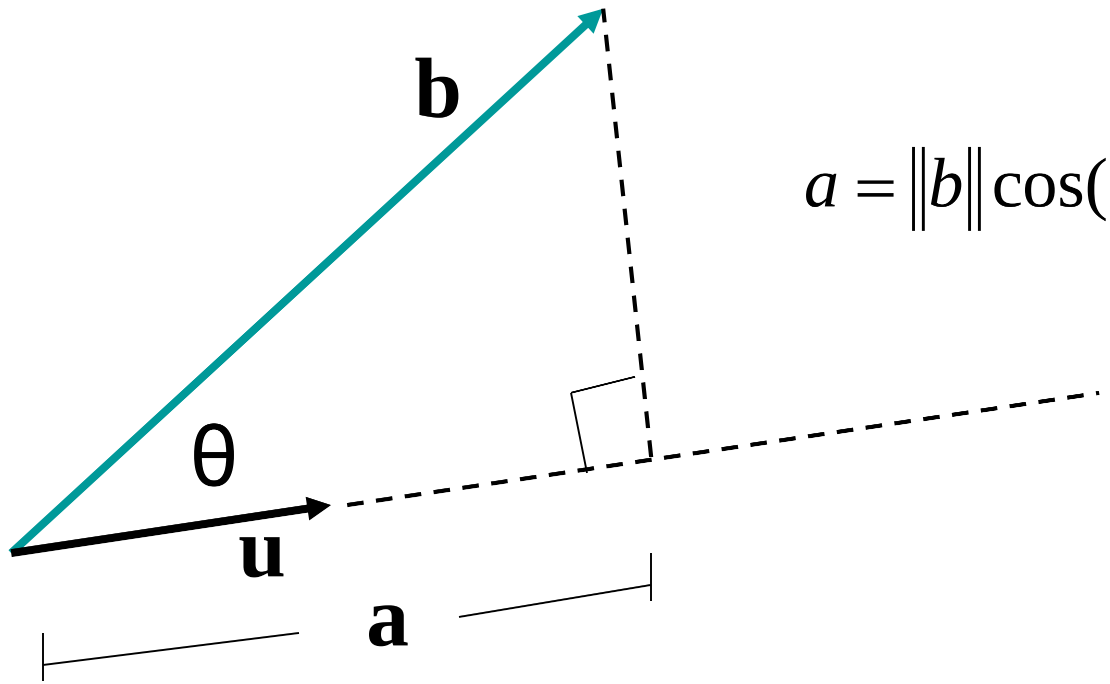
$$a = \|a\| \frac{u}{\|u\|} = \|b\| \cos(\theta) \frac{u}{\|u\|}$$

$$= \frac{<b,u>}{\|u\|} \frac{u}{\|u\|}$$

$$\|a\| = \frac{<b,u>}{\|u\|}$$

Vector Projection (orthogonal projection)

- If $||\mathbf{u}||=1.0$ then $<\mathbf{b},\mathbf{u}>$ is the length of the vector **a** which is the *projection* of **b** onto **u**



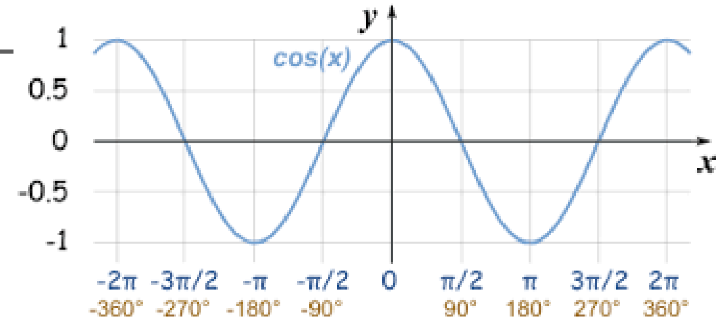
$$a = \|b\| \cos(\theta) \frac{u}{\|u\|} =< b,u > u$$

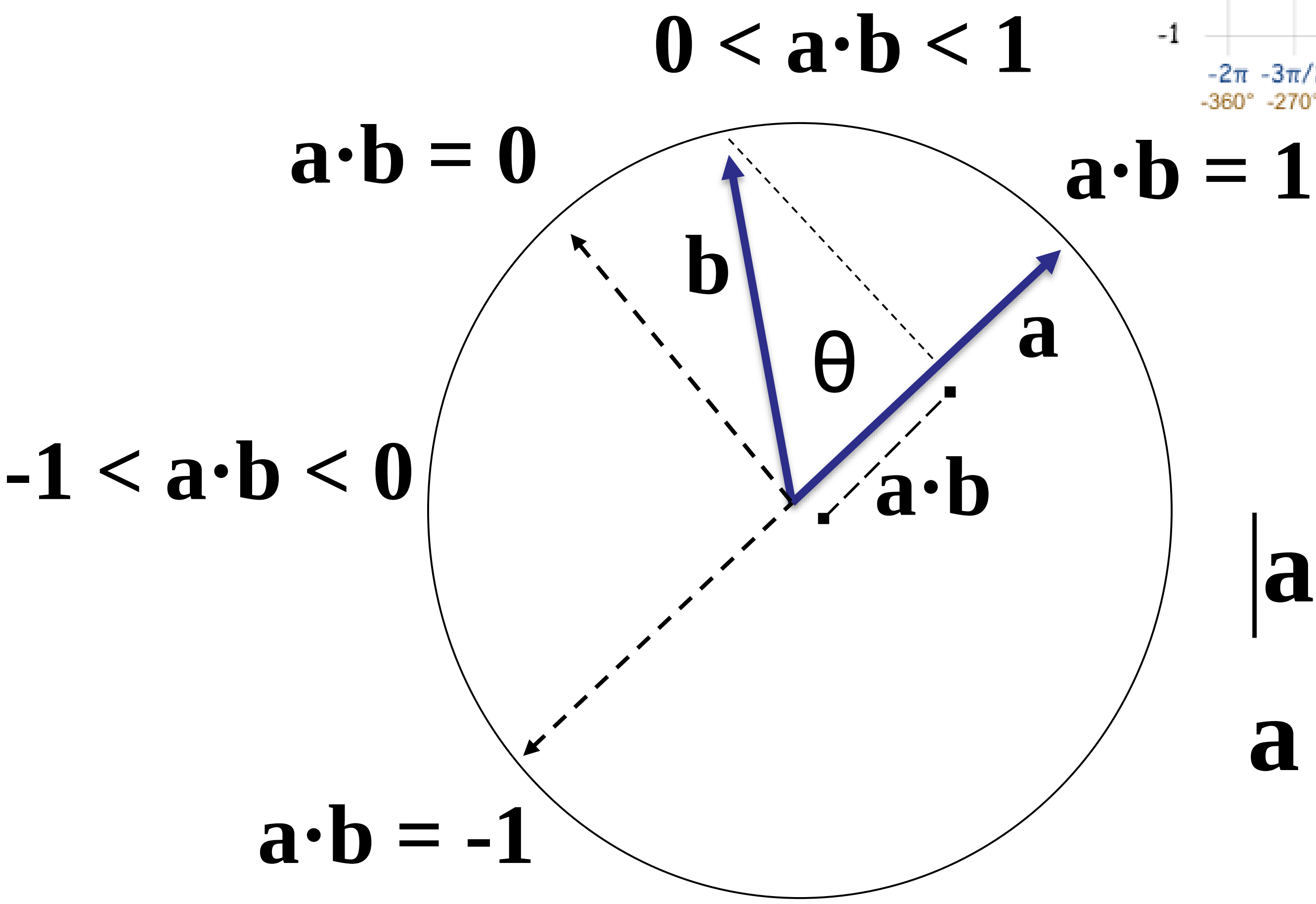
Dot Products with General Vectors

- The dot product is a scalar value that tells us something about the relationship between two vectors

  - If $\mathbf{a} \cdot \mathbf{b} > 0$ then $\theta < 90^\circ$
  - If $\mathbf{a} \cdot \mathbf{b} < 0$ then $\theta > 90^\circ$
  - If $\mathbf{a} \cdot \mathbf{b} = 0$ then $\theta = 90^\circ$ (or one or more of the vectors is degenerate (0,0,0))

Dot Products with unit Vectors



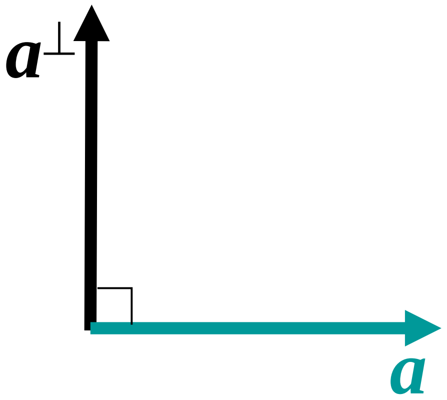


$$|\mathbf{a}| = |\mathbf{b}| = 1.0$$

$$\mathbf{a} \cdot \mathbf{b} = \cos(\theta)$$

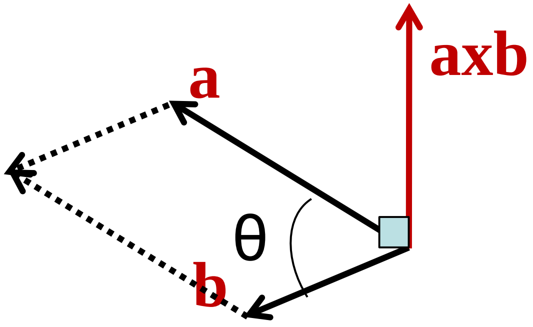
Perpendicular vectors

$$a = (a_x, a_y) \quad \Rightarrow \quad a^{\perp} = \pm(-a_y, a_x)$$



Cross Product : Properties

$\mathbf{a}\times\mathbf{b}$ is a *vector* perpendicular to both **a** and **b**, in the direction defined by the right hand rule



$$\left\|\mathbf{a}\times\mathbf{b}\right\| = \left\|\mathbf{a}\right\|\left\|\mathbf{b}\right\|\sin\theta$$

$\left\|\mathbf{a}\times\mathbf{b}\right\| =$ Area of the parallelogram **ab**

$\left\|\mathbf{a}\times\mathbf{b}\right\| = 0$ if **a** and **b** are parallel

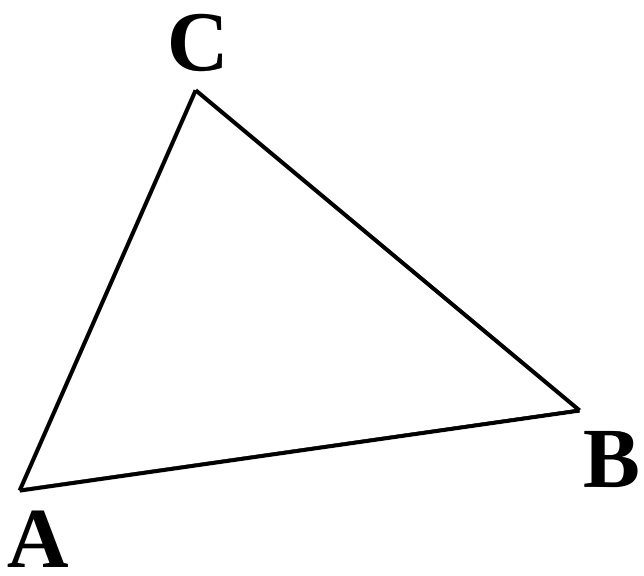
Cross Product

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} i & j & k \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_y b_z - a_z b_y & a_z b_x - a_x b_z & a_x b_y - a_y b_x \end{bmatrix}$$

Example: Normal of a Triangle

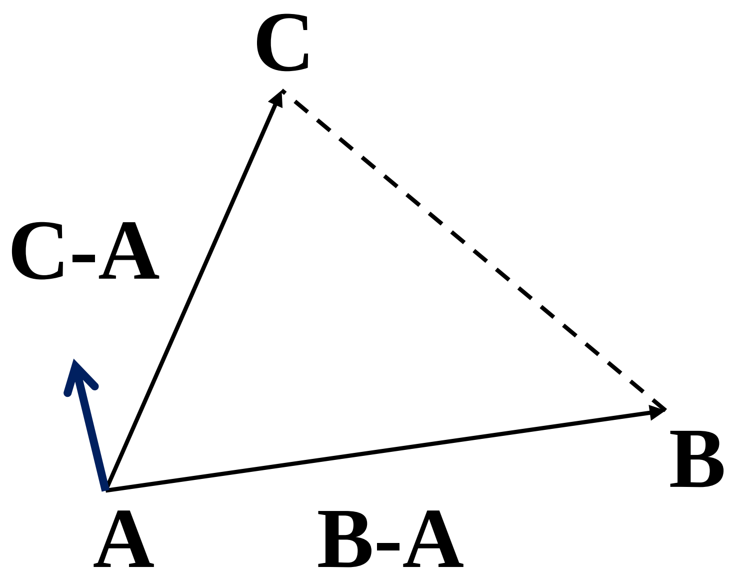
- Find the unit length normal of the triangle defined by 3D points **A**, **B**, and **C**



Example: Normal of a Triangle

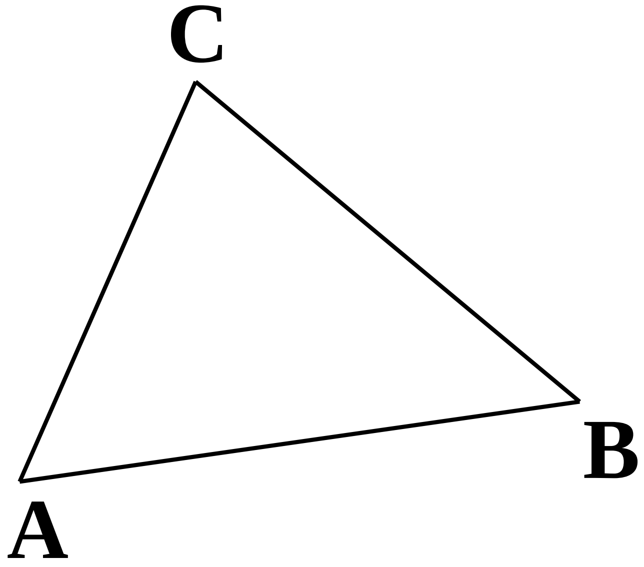
$$\mathbf{n}^* = (B - A) \times (C - A)$$

$$\mathbf{n} = \frac{\mathbf{n}^*}{\left\| \mathbf{n}^* \right\|}$$



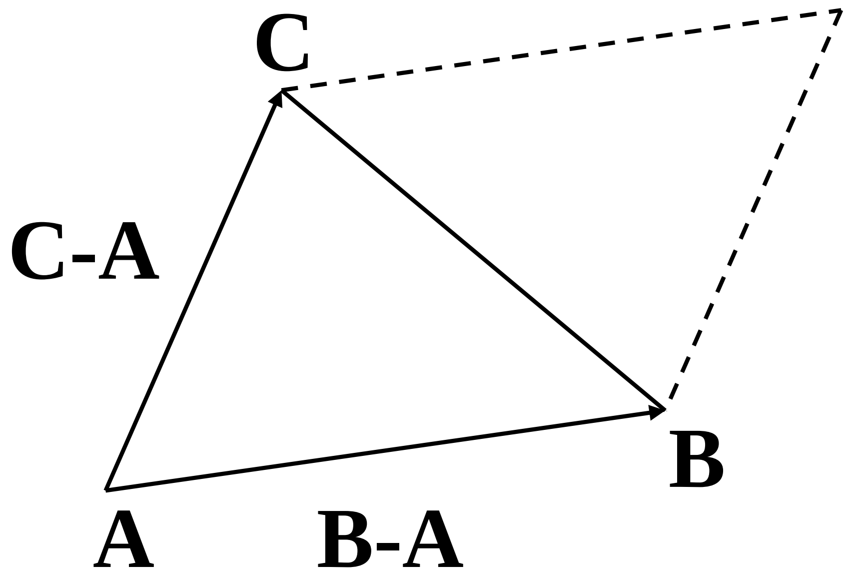
Example: Area of a Triangle

- Find the area of the triangle defined by 3D points **A**, **B**, and **C**



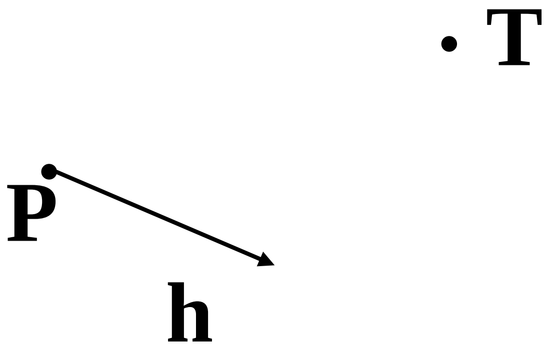
Example: Area of a Triangle

$$area = \frac{1}{2}\left\|\left(B - A\right)\times\left(C - A\right)\right\|$$



Example: Alignment to Target

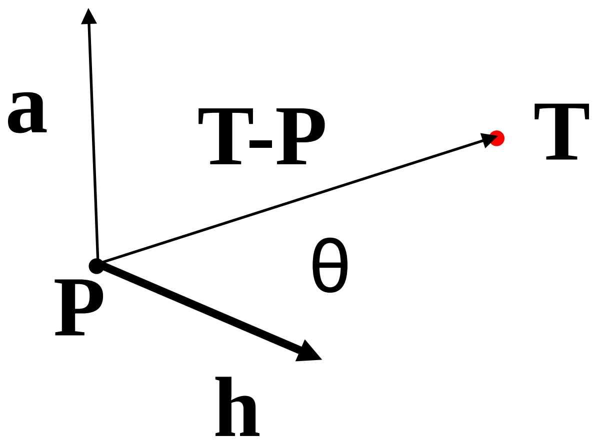
- An object is at position **P** with a unit length heading of **h**. We want to rotate it so that the heading is facing some target **T**.
- Find a unit axis **a** and an angle θ to rotate around.



Example: Alignment to Target

$$\mathbf{a} = \frac{\mathbf{h} \times (T - P)}{\left\| \mathbf{h} \times (T - P) \right\|}$$

$$\theta = \cos^{-1}\left(\frac{\mathbf{h} \cdot (T - P)}{\left\| (T - P) \right\|} \right)$$



Vector Class (C++)

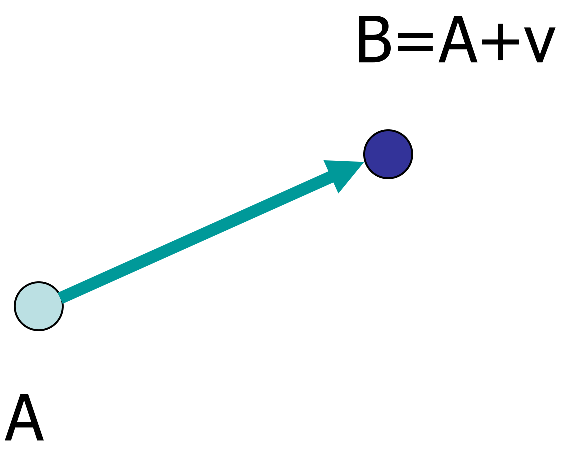
```
class Vector3 {
public:
    Vector3()                                  {x=0.0f; y=0.0f; z=0.0f;}
    Vector3(float x0,float y0,float z0)        {x=x0; y=y0; z=z0;}
    void Set(float x0,float y0,float z0)       {x=x0; y=y0; z=z0;}
    void Add(Vector3 &a)                       {x+=a.x; y+=a.y; z+=a.z;}
    void Add(Vector3 &a,Vector3 &b)            {x=a.x+b.x; y=a.y+b.y; z=a.z+b.z;}
    void Subtract(Vector3 &a)                  {x-=a.x; y-=a.y; z-=a.z;}
    void Subtract(Vector3 &a,Vector3 &b)       {x=a.x-b.x; y=a.y-b.y; z=a.z-b.z;}
    void Negate()                              {x=-x; y=-y; z=-z;}
    void Negate(Vector3 &a)                    {x=-a.x; y=-a.y; z=-a.z;}
    void Scale(float s)                        {x*=s; y*=s; z*=s;}
    void Scale(float s,Vector3 &a)             {x=s*a.x; y=s*a.y; z=s*a.z;}
    float Dot(Vector3 &a)                      {return x*a.x+y*a.y+z*a.z;}
    void Cross(Vector3 &a,Vector3 &b)
            {x=a.y*b.z-a.z*b.y; y=a.z*b.x-a.x*b.z; z=a.x*b.y-a.y*b.x;}
    float Magnitude()                          {return sqrtf(x*x+y*y+z*z);}
    void Normalize()                           {Scale(1.0f/Magnitude());}

    float x,y,z;
};
```

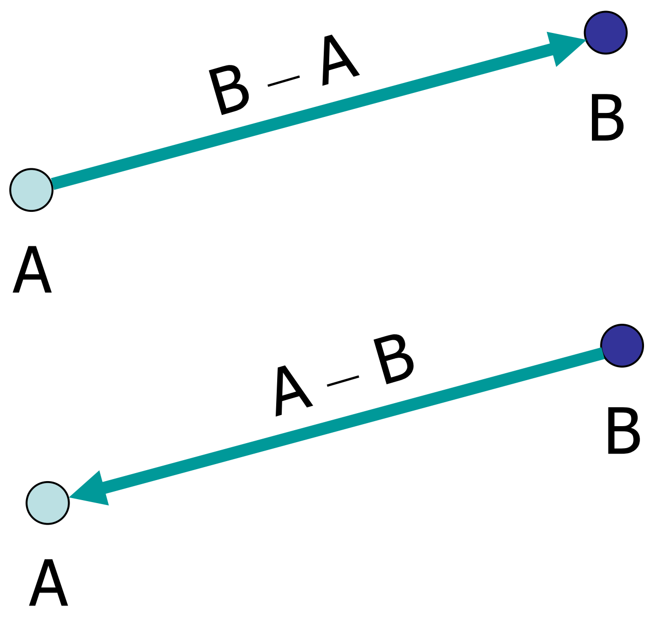
Affine Space

- In a linear space the concept of location is missing
- An affine space is an extension of a linear space which includes the **Point**
- New operations:
  - sum *point + vector*: defines a new point
  - difference *point-point*: defines a vector

Point + vector = point



point - point = vector



point + point: not defined!!

Map points to vectors

- If we have a coordinate system with origin at point O
- We can define correspondence between points and vectors:

$$\boldsymbol{P} \quad \rightarrow \quad \boldsymbol{v} = \boldsymbol{P} - \boldsymbol{O}$$
$$\boldsymbol{v} \quad \rightarrow \quad \boldsymbol{P} = \boldsymbol{O} + \boldsymbol{v}$$

Affine Combination

Affine Combination is a linear combination of points with coefficients that sum up to 1

$$P = a_1 P_1 + a_2 P_2 + \dots\dots + a_n P_n$$

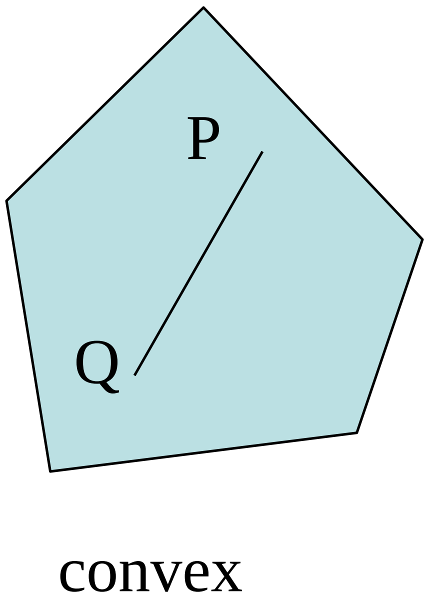
$$for \quad some \quad a_1 + a_2 + \dots\dots + a_n = 1$$

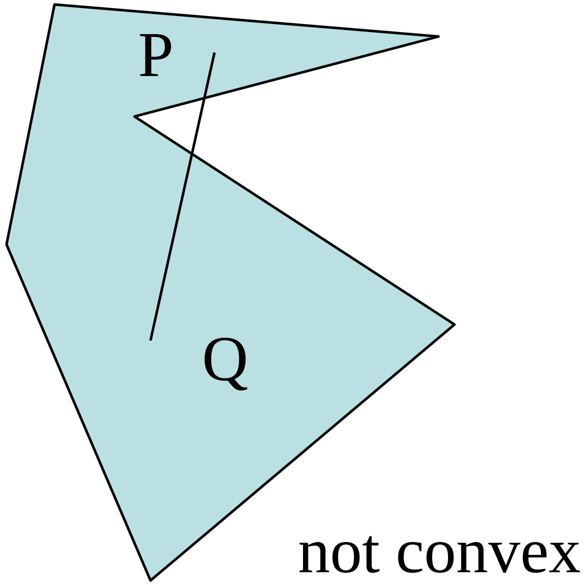
The coefficients $(a_1, a_2, \dots, a_n)$ are defined as **barycentric coordinates** of P in the affine space

Affine Combinations with coefficients $(a_1, a_2, \dots, a_n)$ in [0,1] are called **convex combinations**

Convexity

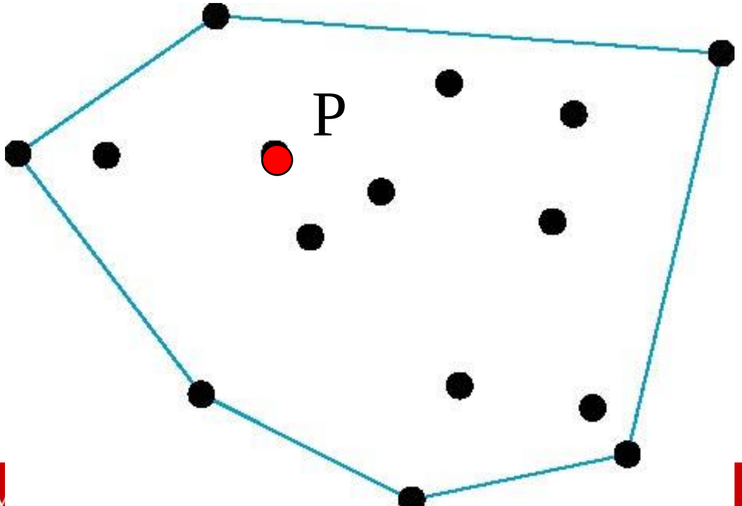
- An object is *convex* iff for any two points in the object all points on the line segment between these points are also in the object





Convex hull

- Given a set of points $$P_1, P_2, \ldots\ldots, P_n$$
- The set of all the points **P** which can be represented as a convex combinations is called convex hull (guscio convesso) of the set
- Smallest convex object containing $P_1, P_2, \ldots\ldots P_n$



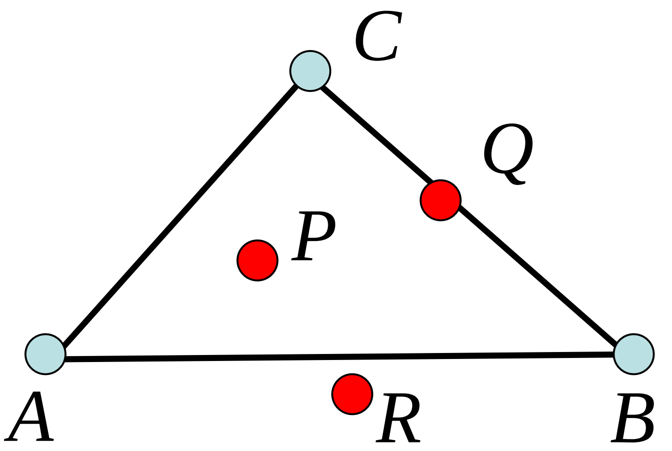
Barycentric coordinates (2D)

- Define a point's position relatively to some fixed points A,B,C

$$P = \alpha A + \beta B + \gamma C,$$

where A, B, C are not on one line, and $\alpha, \beta, \gamma \in \mathrm{R}$.

- (α, β, γ) are called Barycentric coordinates of P with respect to A, B, C (unique!)
- If P is inside the triangle, $\alpha, \beta, \gamma \in [0, 1]$, $\alpha+\beta+\gamma = 1$

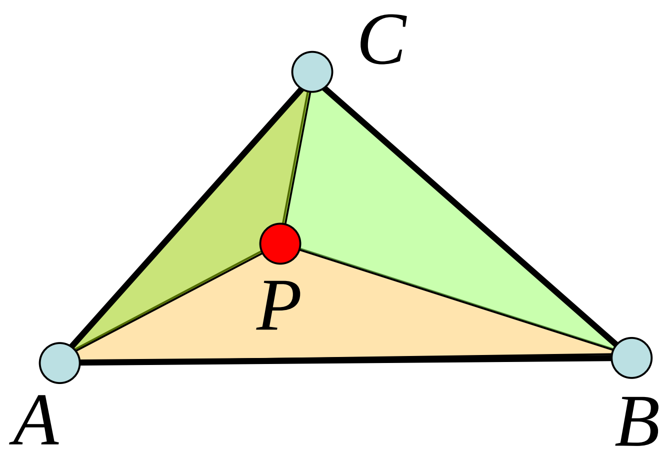


$Q, R??$

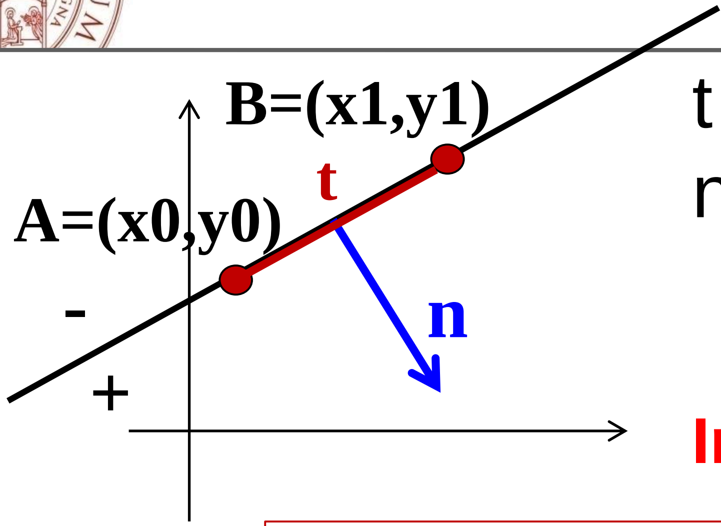
Barycentric coordinates (2D)

$$P = \frac{\langle P, B, C \rangle}{\langle A, B, C \rangle} A + \frac{\langle P, C, A \rangle}{\langle A, B, C \rangle} B + \frac{\langle P, A, B \rangle}{\langle A, B, C \rangle} C$$

$\langle \cdot, \cdot, \cdot \rangle$ denotes the area of the triangle



Compute BC



$t = (x_1\text{-}x_0, y_1\text{-}y_0)$

$n = \mathrm{Perp}(t) = (y_1\text{-}y_0, -(x_1\text{-}x_0))$

Normalize if desired

Implicit line equation E_{AB} :

$(P - A) \bullet n = 0$ for all points P on the line

$$\begin{pmatrix} x - x_0 & y - y_0 \end{pmatrix} \begin{pmatrix} y_1 - y_0 \\ -(x_1 - x_0) \end{pmatrix} \begin{matrix} = 0 & on \\ < 0 & left \\ > 0 & right \end{matrix}$$

$$E_{AB}(x, y) = (x - x_0)(y_1 - y_0) - (y - y_0)(x_1 - x_0)$$

Compute BC

$$area_{ABC} = \frac{1}{2}\left\|(B-A)\times(C-A)\right\|$$

$$= \frac{1}{2}((x_1 - x_0)(y_2 - y_0) - (y_1 - y_0)(x_2 - x_0))$$

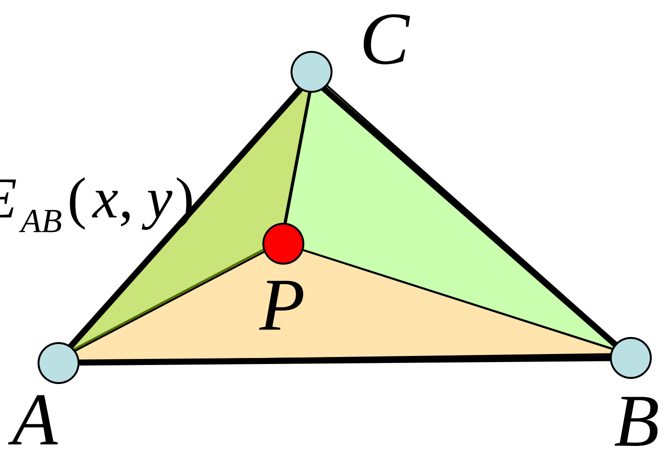
$$area_{ABC} = area_{APB} + area_{BPC} + area_{CPA}$$

$$area_{APB} = \frac{1}{2}\left\|(P-A)\times(B-A)\right\| =$$

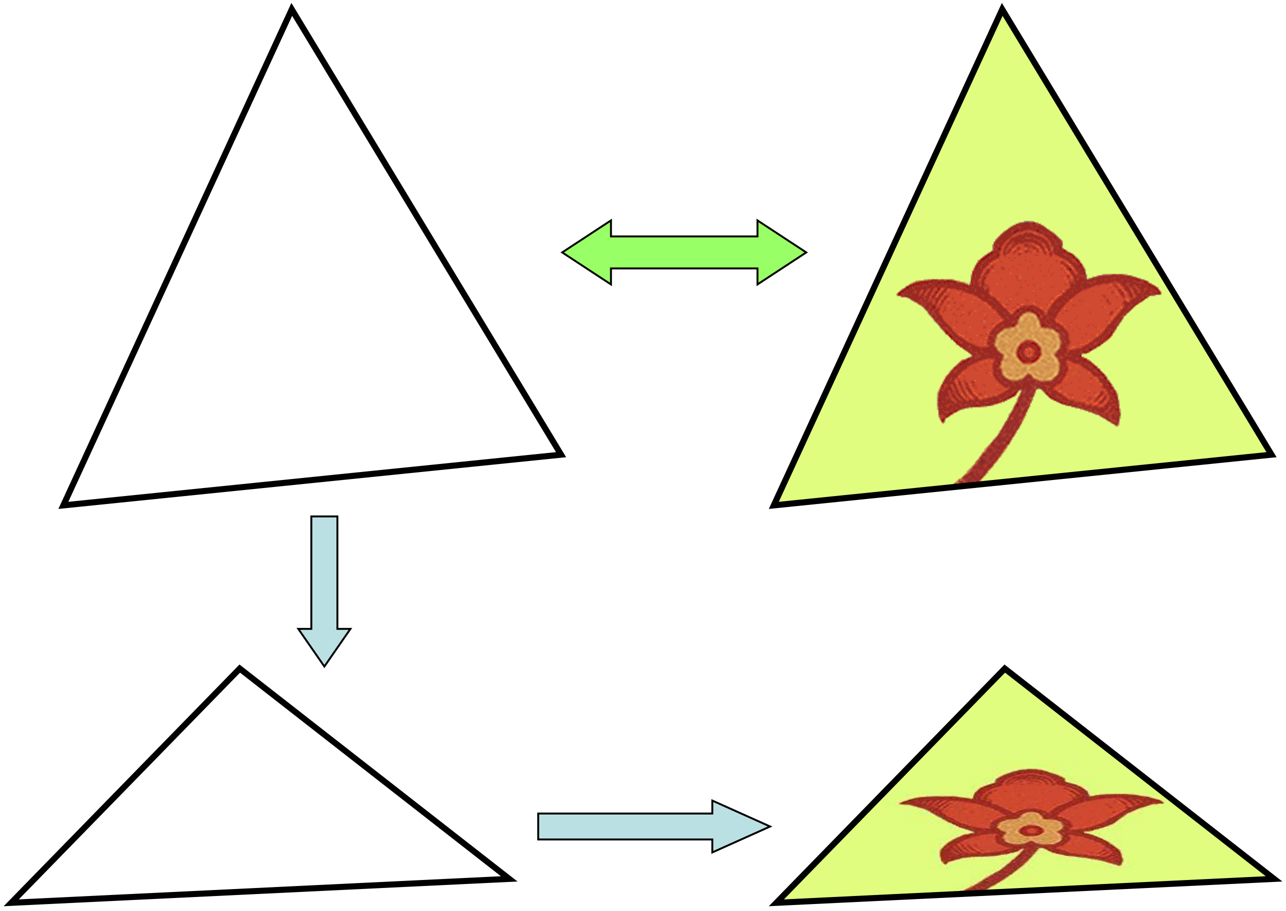
$$= \frac{1}{2}((x - x_0)(y_1 - y_0) - (y - y_0)(x_1 - x_0)) = \frac{1}{2}E_{AB}(x,y)$$

$$area_{BPC} = \frac{1}{2}E_{BC}(x,y)$$

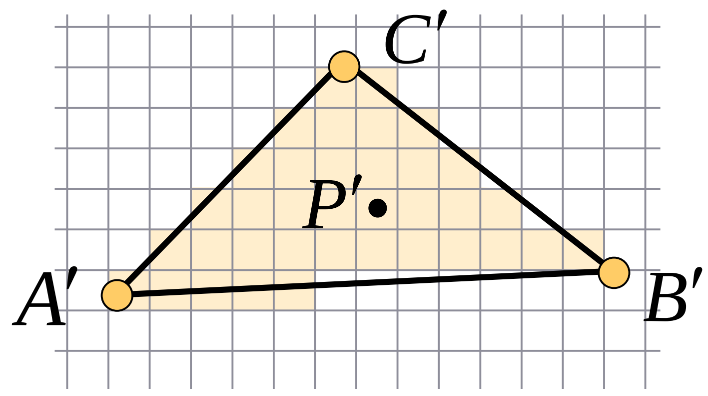
$$area_{CPA} = \frac{1}{2}E_{CA}(x,y)$$



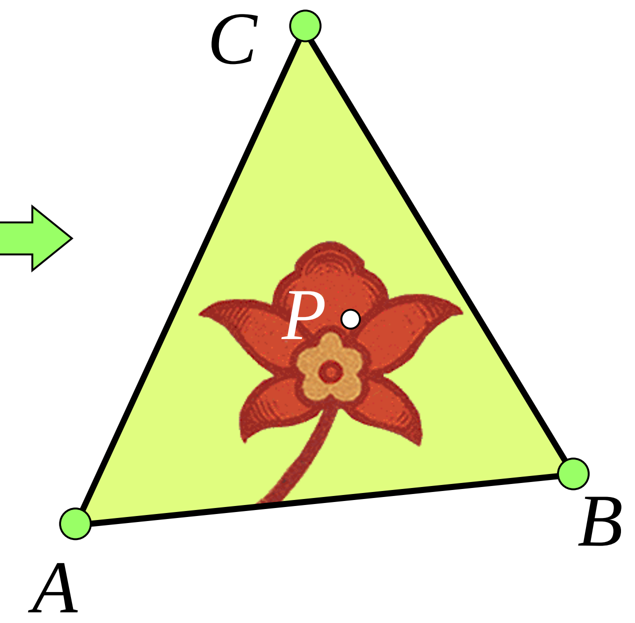
Example of usage: warping

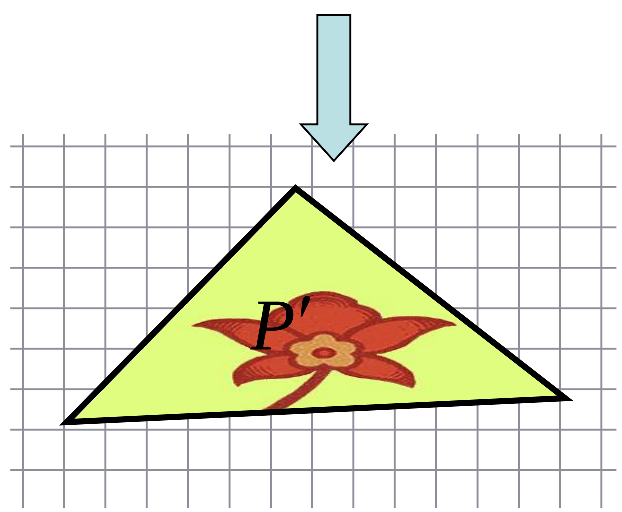


Example of usage: warping



TARGET





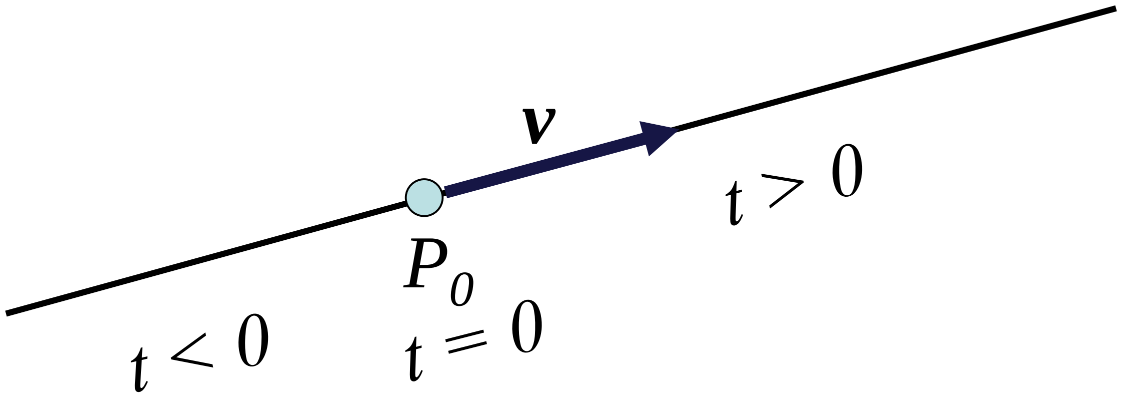
We take the barycentric coordinates α, β, γ of *P’* with respect to *A’*, *B’*, *C’*

Color(*P’*) = Color(α *A* + β *B* + γ *C*)

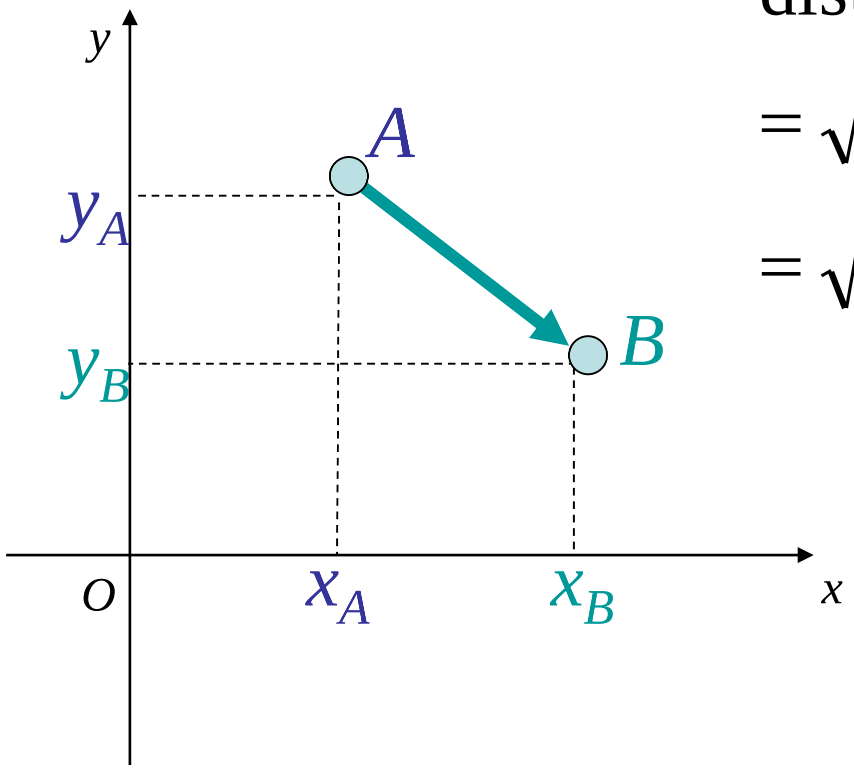
Parametric equation of a line

- Set of all points that pass through P_0 in the direction of the vector $\mathbf{v}$

$$\ell(t) = P_0 + t\mathbf{v}, \quad t \in (-\infty, \infty)$$



Distance between two points



$$\mathrm{dist}(A,B) = \| B - A \| =$$

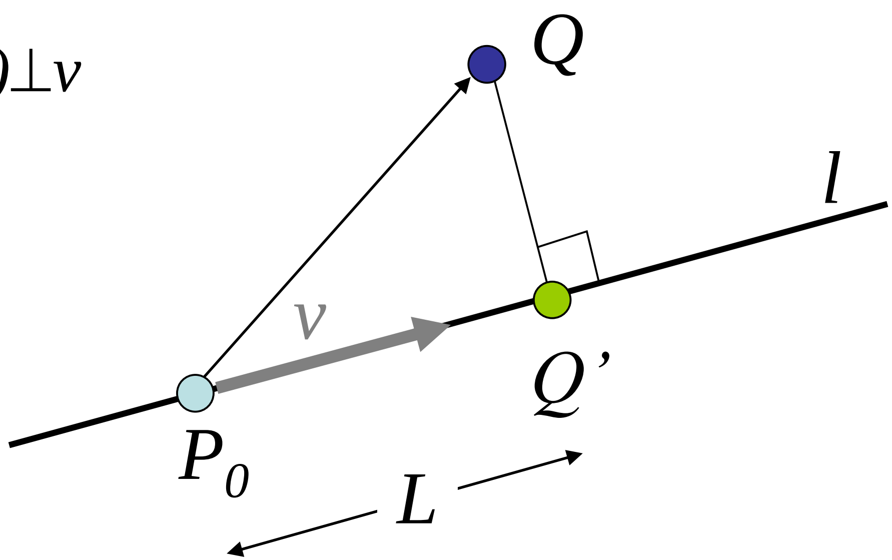
$$= \sqrt{< B - A, B - A >} =$$

$$= \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

Distance between point and line

Find a point Q' such that $(Q - Q') \perp v$

dist(Q, l) = $\|Q - Q'\|$



Pitagora :

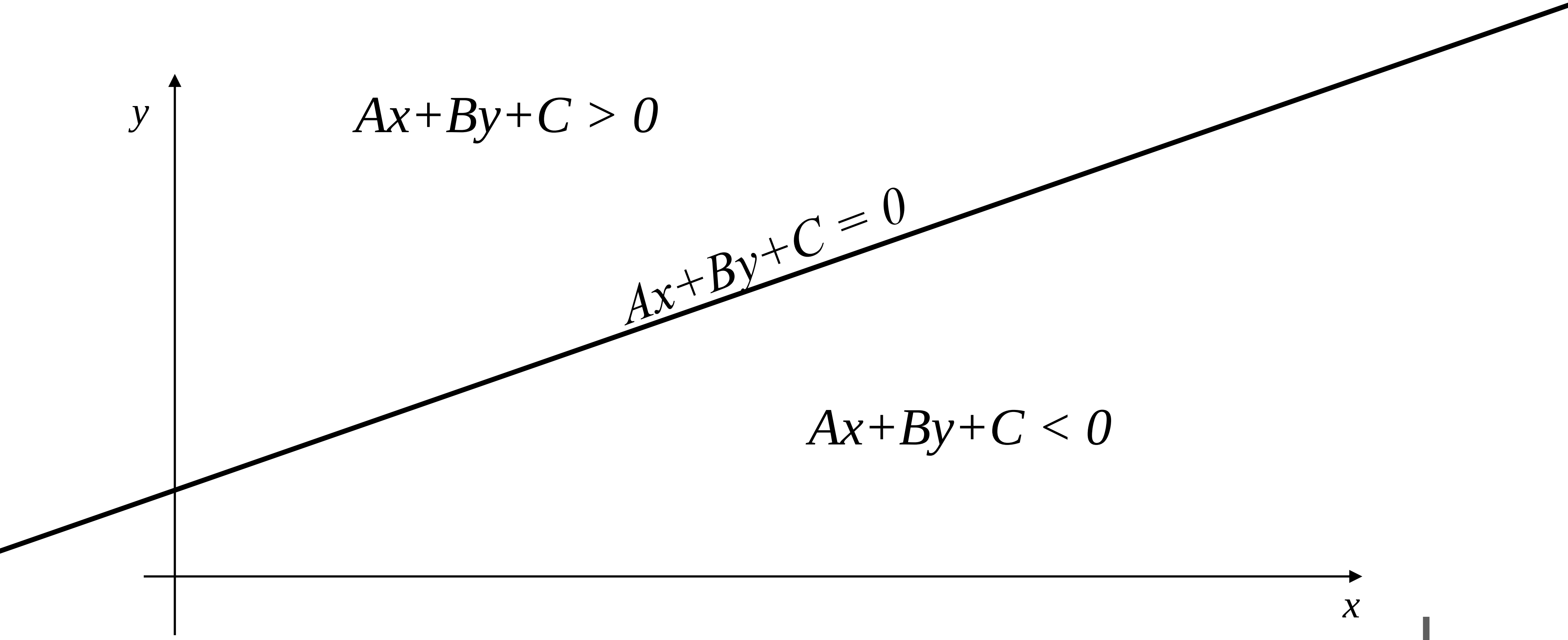
$$L^2 + \mathrm{dist}(Q, Q')^2 = \| Q - P_0 \|^2$$

$$L = \frac{< Q - P_0, v >}{\| v \|}$$

$$\Rightarrow \quad \mathrm{dist}(Q, Q')^2 = \| Q - P_0 \|^2 - L^2 = \| Q - P_0 \|^2 - \frac{< Q - P_0, v >^2}{\| v \|^2}.$$

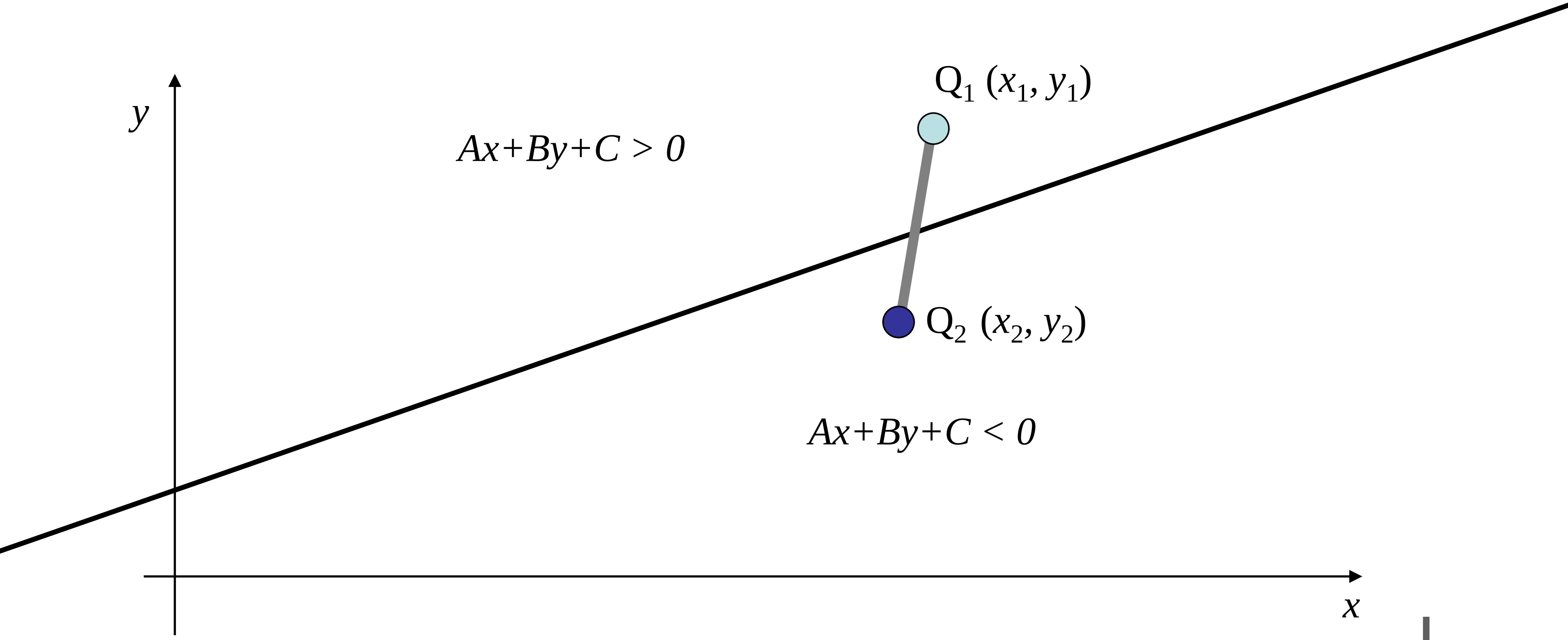
Implicit equation of a line in 2D

$$\boxed{Ax + By + C = 0, \quad A, B, C \in \mathbb{R},\ AB \neq 0}$$



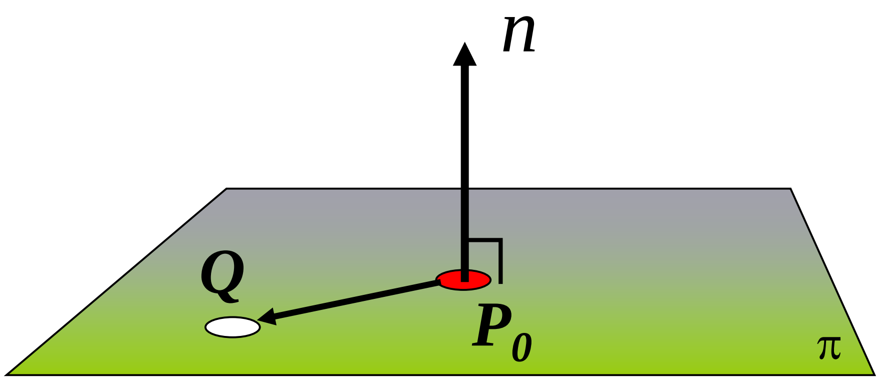
Line-segment intersection

$$\text{The segment } Q_1Q_2 \text{ intersects the line} \Leftrightarrow$$
$$\Leftrightarrow (Ax_1 + By_1 + C)(Ax_2 + By_2 + C) \le 0$$



Representation of a plane in 3D space

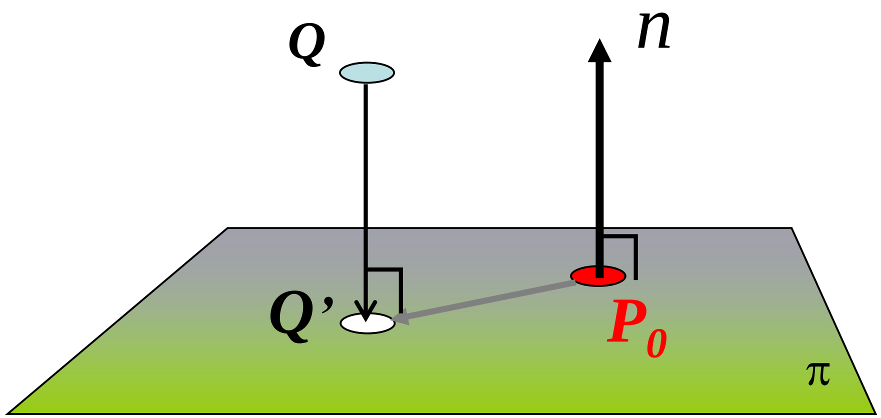
- The plane π is defined by a normal n and one point in the plane (P_0).
- A point Q belongs to the plane $\Leftrightarrow$ $\boxed{<Q - P_0 , n> = 0}$
- The normal n is perpendicular to all vectors in the plane



Distance between point and plane

- Project the point onto the plane in the direction of the normal:

$$\mathrm{dist}(Q, \pi) = ||Q' - Q||$$



Distance between point and plane

$$(Q' - Q) \parallel n \quad \Rightarrow \quad Q' - Q = sn, \quad s \in \mathrm{R} \quad \Rightarrow \quad Q' = Q + sn$$

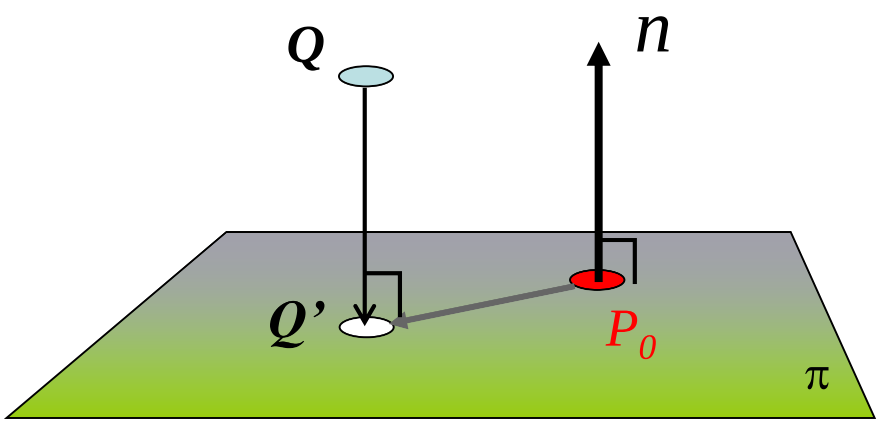
$$< Q' - P_0, n > = 0$$

$$< Q - P_0 + sn, n > = 0$$

$$< Q - P_0, n > + s < n, n > = 0$$

$$\boxed{s = -\frac{< Q - P_0, n >}{\| n \|^2}}$$

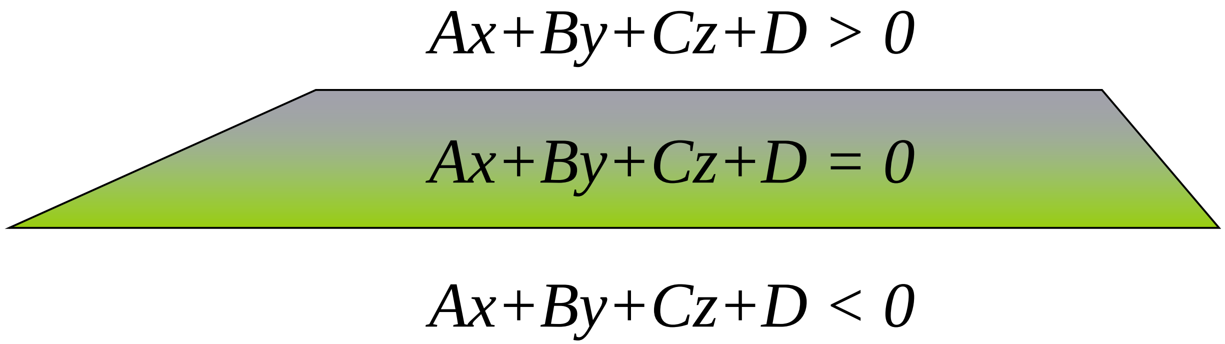
$$\mathrm{dist}^2(Q, \pi) = \| Q' - Q \|^2 = s^2 \| n \|^2 = \frac{< Q - P_0, n >^2}{\| n \|^2}$$



Implicit representation of planes in 3D

- (x, y, z) are coordinates of a point on the plane
- (A, B, C) are the coordinates of a normal vector to the plane

$$\boxed{Ax + By + Cz + D = 0, \quad A, B, C, D \in \mathrm{R},\ ABC \neq 0}$$



Coordinate Frame
(Sistemi di riferimento)

- A frame is defined by the quadruple

  F=(Po,v1,v2,v3)

  where
  - Po is a point (origin)
  - [v1,v2,v3] is a vector basis (orthonormal)

- In a frame F the coordinates (a_1, a_2, a_3) uniquely describe the point:

$$P = P_0 + a_1 v_1 + a_2 v_2 + a_3 v_3$$

Homogeneous Coordinates

- Representing both vectors and points using three scalar values is ambiguous..
- We consider a coordinate system which allows for a unique representation for points and vectors
- A **vector** is represented as

$$w = a_1 v_1 + a_2 v_2 + a_3 v_3$$

- A **point** is represented as

$$P = P_0 + a_1 v_1 + a_2 v_2 + a_3 v_3$$

Homogeneous Coordinates

- Assume

  1*P=P   and   0*P=0   (zero vector)

- A **vector** is then given by

$$w = a_1 v_1 + a_2 v_2 + a_3 v_3 + 0 \cdot P_0$$

- A **point** is then given by

$$P = a_1 v_1 + a_2 v_2 + a_3 v_3 + 1 \cdot P_0$$

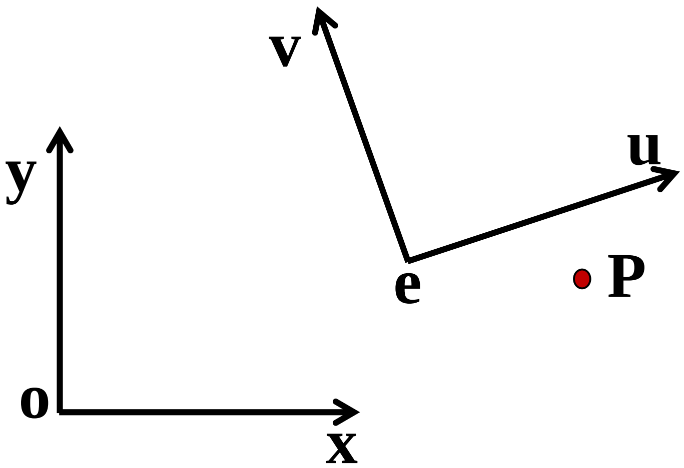
Homogeneous Coordinates

- Add an extra dimension, each point has an extra value, *0 or 1*
- Every point/vector is defined by 4 coordinates. In vector form:

Point coordinates: $\begin{bmatrix} a_1 & a_2 & a_3 & 1 \end{bmatrix}^T$

Vector coordinates: $\begin{bmatrix} a_1 & a_2 & a_3 & 0 \end{bmatrix}^T$

Change of reference systems (frames) in 2D



Change the coordinates of a vector/point from one system to another

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \begin{bmatrix} x_u & x_v & x_e \\ y_u & y_v & y_e \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}$$

$$P_{xy} = \begin{bmatrix} u & v & e \\ 0 & 0 & 1 \end{bmatrix} P_{uv}$$

Matrix of the coordinate change

Express P in the frame (o,x,y)

$$\mathbf{P} = (x_p, y_p) = \mathbf{o} + x_p\,\mathbf{x} + y_p\,\mathbf{y}$$

$$(x_p, y_p) = (2.5, 0.9)$$

Express P in the frame (e,u,v)

$$\mathbf{P} = (u_p, v_p) = \mathbf{e} + u_p\,\mathbf{u} + v_p\,\mathbf{v}$$

$$(u_p, v_p) = (0.5, -0.7)$$

Change of reference systems (frames) in 3D

Given the reference systems (frames) (o,x,y,z) and (e,u,v,w) represent P:

Coords of u
wrt (o,x,y,z)

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} x_u & x_v & x_w & x_e \\ y_u & y_v & y_w & y_e \\ z_u & z_v & z_w & z_e \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ w_p \\ 1 \end{bmatrix}$$

$$P_{xyz} = \begin{bmatrix} u & v & w & e \\ 0 & 0 & 0 & 1 \end{bmatrix} P_{uvw}$$

$$P_{xyz} = M \; P_{uvw}$$

$$\begin{bmatrix} u_p \\ v_p \\ w_p \\ 1 \end{bmatrix} = \begin{bmatrix} x_u & x_v & x_w & x_e \\ y_u & y_v & y_w & y_e \\ z_u & z_v & z_w & z_e \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

$$P_{uvw} = \begin{bmatrix} u & v & w & e \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} P_{xyz}$$

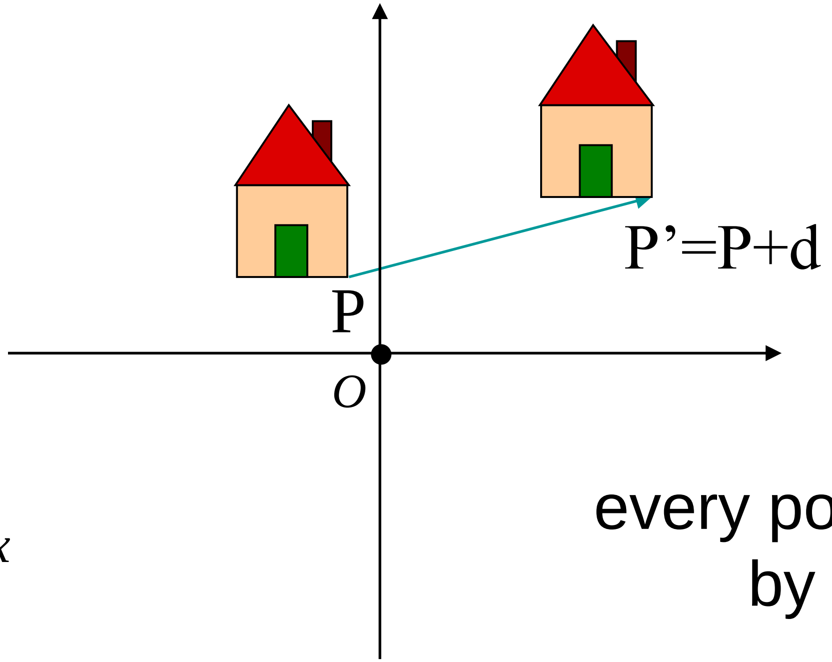
$$P_{uvw} = M^{-1} P_{xyz}$$

Geometric Transformations

- Transform the object coords in order to obtain a similar object which differs for **position, orientation and size**
- Modify the geometry but not the topology of the objects
- Transformations are used:
  - Position objects in a scene (modeling)
  - Change the shape of objects
  - Create multiple copies of objects
  - Projection for virtual cameras
  - Animations

2D Translation

- Move (translate, displace) a point P to a new location
- Displacement determined by a vector $d=(d_x,d_y)$

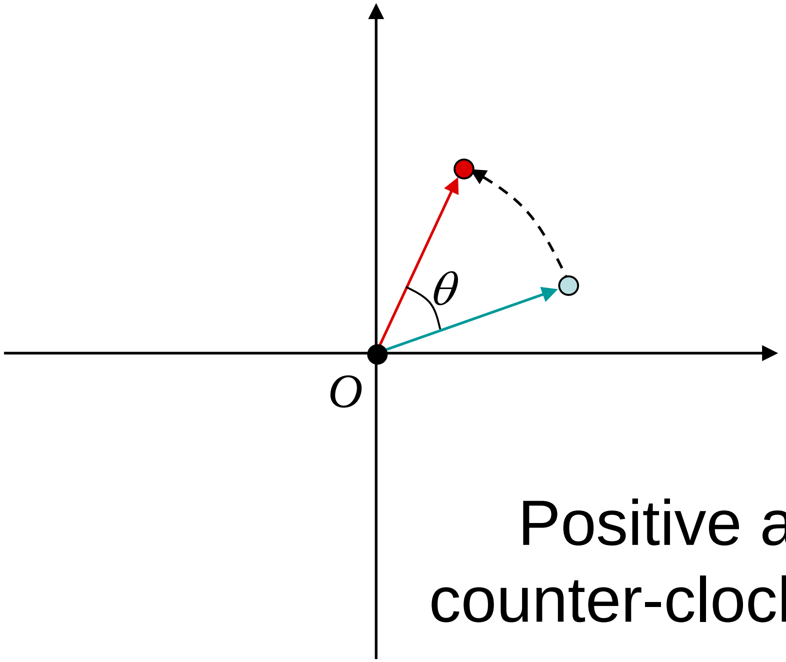


$$x' = x + d_x$$
$$y' = y + d_y$$

every point displaced by same vector

2D Rotation

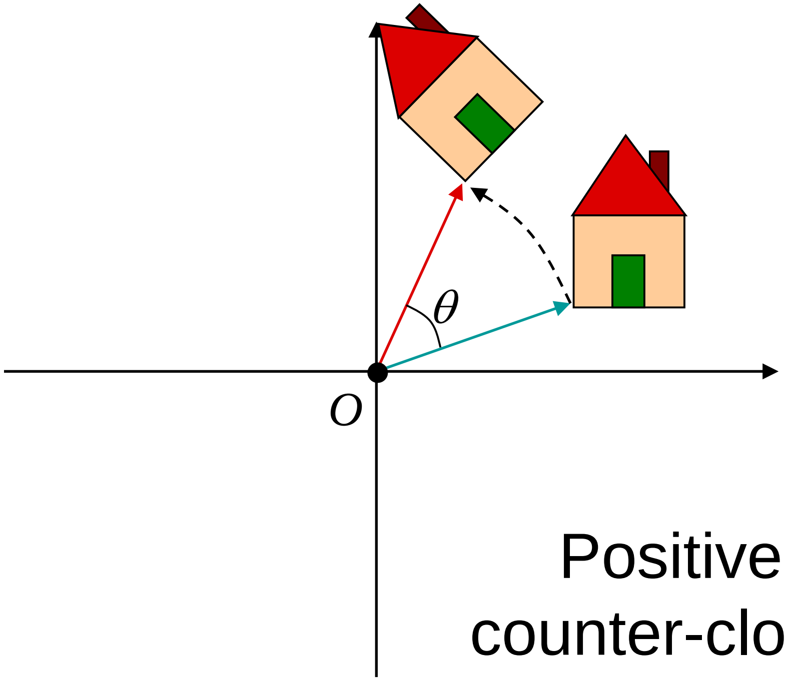
- Rotation about the origin by angle θ



Positive angle means counter-clockwise direction.

2D Rotation

- Rotation about the origin by angle θ



Positive angle means counter-clockwise direction.

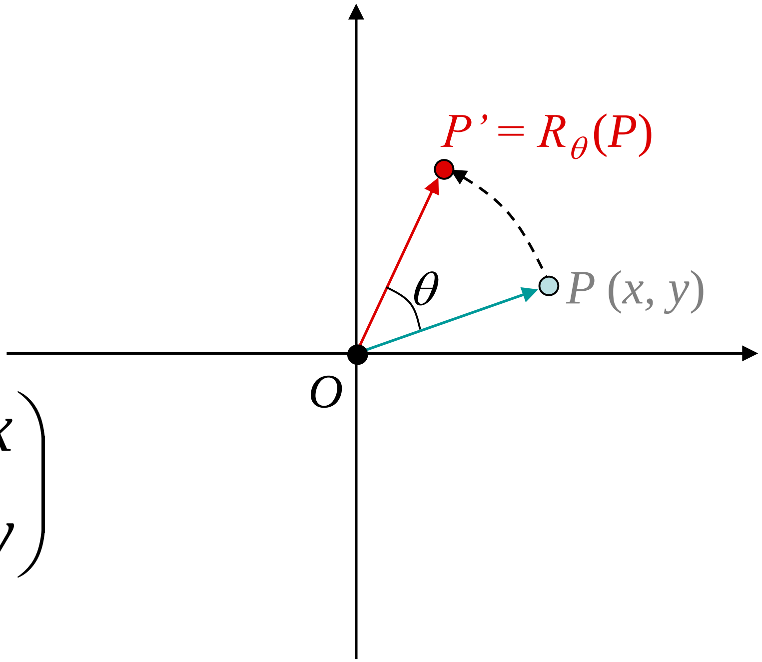
Rotation in 2D – matrix representation

- Multiply P=(x, y) by the rotation matrix:

$$P' = R_\theta(P)$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$x' = x\cos\theta \; - \; y\sin\theta$$
$$y' = x\sin\theta \; + \; y\cos\theta$$



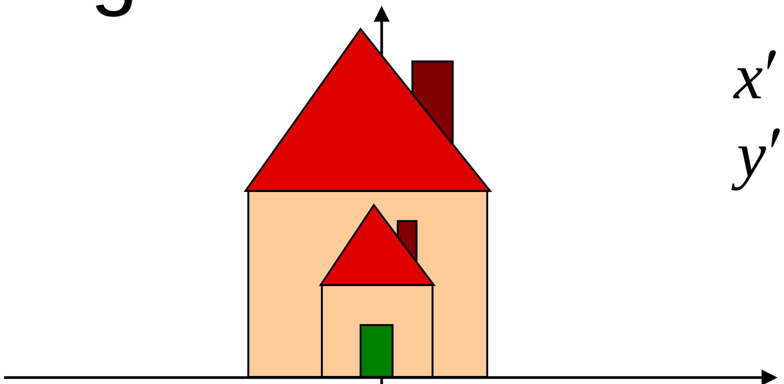
2D Scale

- Uniform sx=sy
- Non uniform sx≠sy
- About the origin

$$P' = S(P)$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} sx & 0 \\ 0 & sy \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$x' = sx * x$$
$$y' = sy * y$$



$0 \leq sx, sy < 1$ • the object will shrink

$sx, sy > 1$ • the object will grow by a factor of s in each dimension

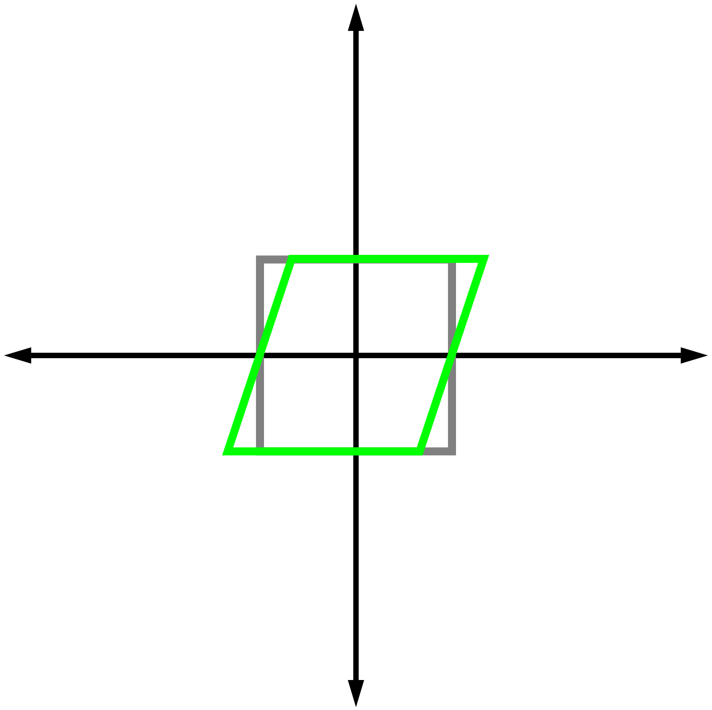
$sx < 0 \; o \; sy < 0$ • the object will be *reflected* across all two dimensions, leading to an object that is 'inside out'

2-D Shear (Horizontal)

Shear factor **s** (positive for the figure below)

$$P' = H_{xy}(s)P$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



$$x' = x + sy$$

$$y' = y$$

Horizontal displacement proportional to vertical position

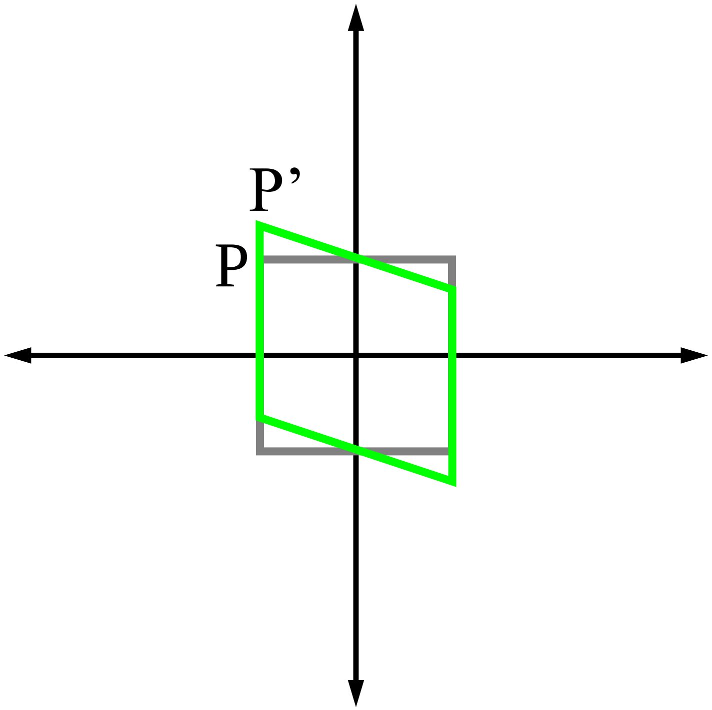
H_{ij} i coord. will change, j coord. will deform

2-D Shear (Vertical)

Shear factor **s** (negative for the figure below)

$$P' = H_{yx}(s)P$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



$$x' = x$$

$$y' = y + sx$$

How are Invertible Linear Transformations Represented?

$$x' = ax + by$$

$$y' = dx + ey$$

An invertible linear transformation is represented by a non-singular matrix M

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ d & e \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$p' = M p$$

Each non-singular matrix defines a linear transformation

$$\mathbf{L(p + q) = L(p) + L(q)}$$

$$\mathbf{L(ap) = a\, L(p)}$$

How are Affine Transformations Represented?

$$x' = ax + by + c$$

$$y' = dx + ey + f$$

Compose linear transformation and translation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$

$$p' = M p + t$$

Affine Transformations in Homogeneous Coordinates

Translations can be encoded in the matrix

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$p' = M p$$

M affine matrix

$$x' = ax + by + c$$

$$y' = dx + ey + f$$

Affine Transformations in Homogeneous Coordinates

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Inverses

If **M** transforms **P** into **P'**, then **M**$^{-1}$ transforms **P'** back to **P**

$$MM^{-1} = I$$

$$P' = MP \quad -> \quad P = M^{-1}P'$$

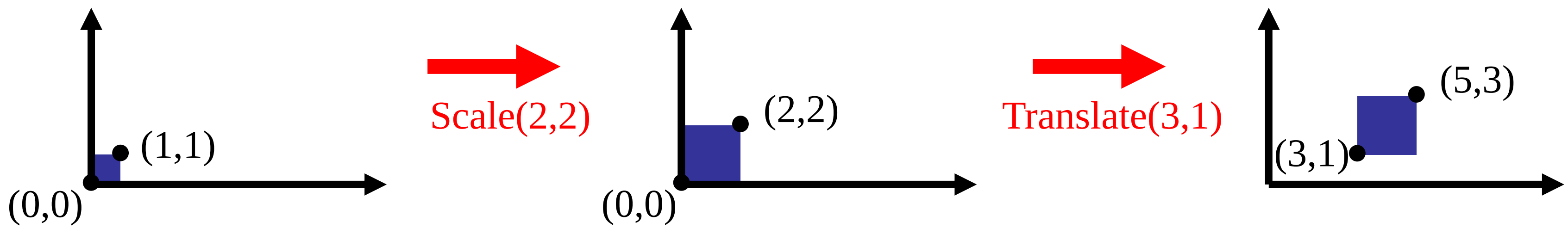
$$T^{-1}(d) = T(-d)$$

$$S^{-1}(s) = S(1/s_x, 1/s_y)$$

$$R^{-1}(\vartheta) = R^T(\vartheta) = R(-\vartheta)$$

How are transforms combined?

Scale then Translate



Use matrix multiplication:

p’ = S(p) p’’=T(p’)=T (S p) = TS p

$$TS = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Multiple Transformations

- v is transformed in v' by means of sequence of transformations:

$$\mathbf{v}' = \mathbf{M}_4 \cdot \left(\mathbf{M}_3 \cdot \left(\mathbf{M}_2 \cdot \left(\mathbf{M}_1 \cdot \mathbf{v}\right)\right)\right)$$

- Because matrix algebra obeys the *associative* law, we can regroup this as:

$$\mathbf{v}' = \left(\mathbf{M}_4 \cdot \mathbf{M}_3 \cdot \mathbf{M}_2 \cdot \mathbf{M}_1\right) \cdot \mathbf{v}$$

- This allows us to *concatenate* them into a single matrix:

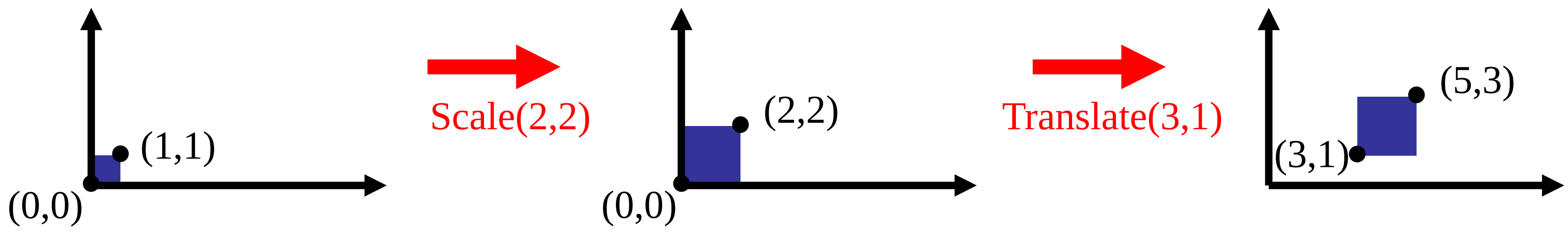
$$\mathbf{M}_{total} = \mathbf{M}_4 \cdot \mathbf{M}_3 \cdot \mathbf{M}_2 \cdot \mathbf{M}_1$$

$$\mathbf{v}' = \mathbf{M}_{total} \cdot \mathbf{v}$$

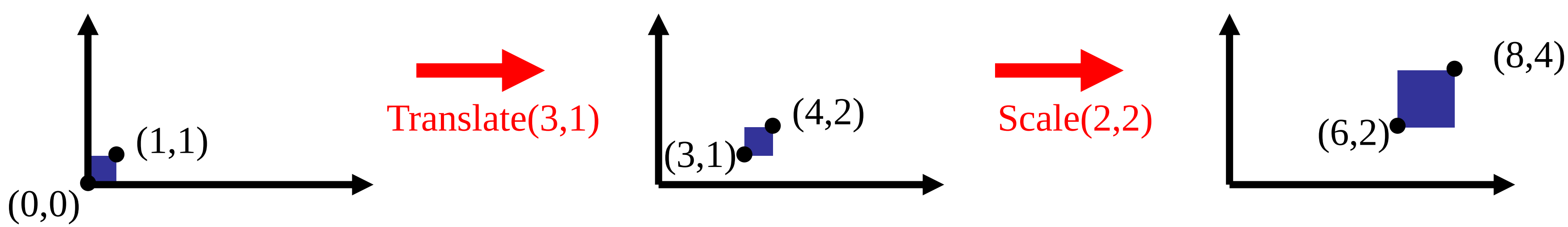
- Caution: matrix multiplication is NOT commutative! So the order of multiplications is important!

Non-commutative Composition

Scale then Translate: $p' = T(S\,p) = TS\,p$



Translate then Scale: $p' = S(T\,p) = ST\,p$



Non-commutative Composition

Scale then Translate: $p' = T(S\,p) = TS\,p$

$$TS = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Translate then Scale: $p' = S(T\,p) = ST\,p$

$$ST = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 6 \\ 0 & 2 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

Pivot Transformations

scaling around a point (dx,dy) that is not the origin

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate to the origin, **scale**, translate to the initial position

rotation around a point (dx,dy) that is not the origin

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate to the origin, **rotate**, translate to the initial position

3D Translate

$$x' = x + d_x$$

$$y' = y + d_y$$

$$z' = z + d_z$$

$$P' = P + \mathbf{d} \quad \Rightarrow \quad \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$$

Homogeneous Transformations

- Affine 3D Transformations in homogeneous coordinates. In a general matrix form:

$$x' = a_1 x + b_1 y + c_1 z + d_1$$

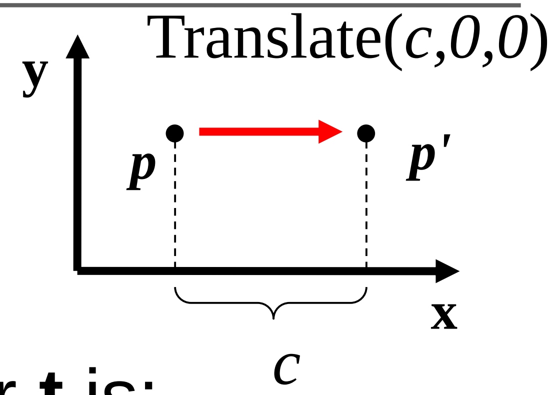
$$y' = a_2 x + b_2 y + c_2 z + d_2$$

$$z' = a_3 x + b_3 y + c_3 z + d_3$$

$$1 = 0x + 0y + 0z + 1$$

$$P' = \mathbf{M} \cdot P \Rightarrow \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

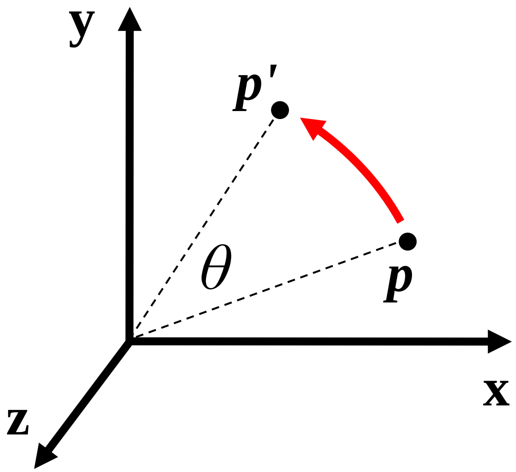
3D Translate (*tx, ty, tz*) in homogeneous coordinates



- Now translations
  can be encoded in the matrix!
- A 4x4 translation matrix that
  translates an object by the vector **t** is:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3D Rotations



- About z axis

$$\mathbf{v}' = R_z(\vartheta) \cdot \mathbf{v}$$

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

Rotation about z axis in three dimensions leaves all points with the same z

3D Rotations

- About x axis:

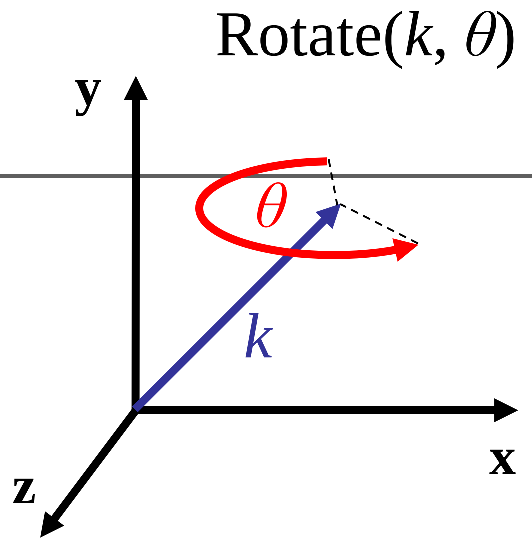
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

- About y axis:

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

Multiple Rotations about an axis are commutative

Arbitrary Axis Rotation

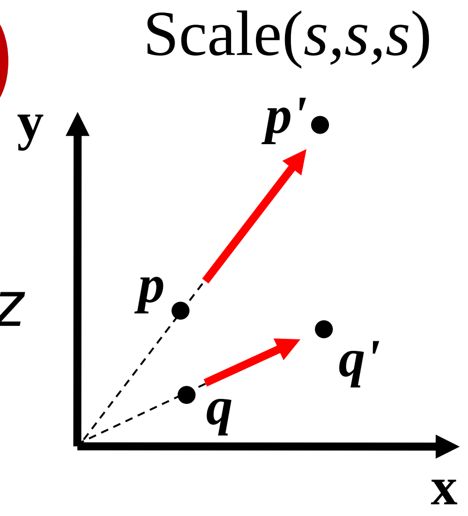


- About (k_x, k_y, k_z), a unit vector on an arbitrary axis (Rodrigues Formula)

$$
\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} =
\begin{pmatrix}
k_x k_x(1-c)+c & k_z k_x(1-c)-k_z s & k_x k_z(1-c)+k_y s & 0 \\
k_y k_x(1-c)+k_z s & k_z k_x(1-c)+c & k_y k_z(1-c)-k_x s & 0 \\
k_z k_x(1-c)-k_y s & k_z k_x(1-c)-k_x s & k_z k_z(1-c)+c & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}
$$

where $c = \cos\theta$ & $s = \sin\theta$

Scale (s_x, s_y, s_z)



- The uniform scaling matrix scales an entire object by scale factor $s = s_x = s_y = s_z$
- The non-uniform scaling matrix scales independently along the x, y, and z axes

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$x' = s_x x$$

$$y' = s_y y$$

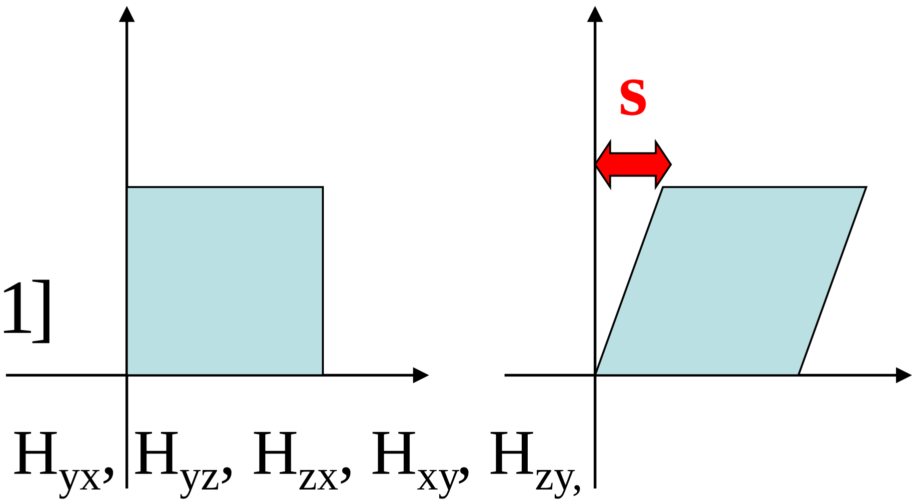
$$z' = s_z z$$

Shear Transformations

- Modify 2 or 3 vectors coords proportionally to the value of the other coords;
- H_{ij} i coord. will change, j coord. will deform

$$H_{xz}(s) = \begin{bmatrix} 1 & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$v' = [v_x + s v_z \quad v_y \quad v_z \quad 1]$$

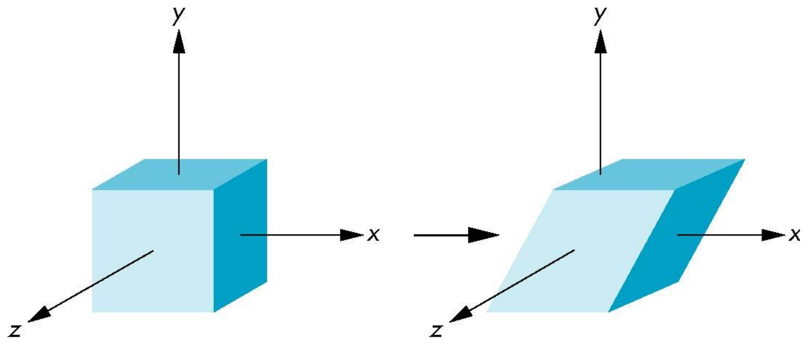


Analogously for H_{xy}, H_{yx}, H_{yz}, H_{zx}, H_{xy}, H_{zy},
$H(s)^{-1} = H(-s)$

Shear Transformations

- A *shear transformation* matrix looks something like this:

$$H(z_1 \ldots z_6) = \begin{bmatrix} 1 & z_1 & z_2 & 0 \\ z_3 & 1 & z_4 & 0 \\ z_5 & z_6 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



- With pure shears, only one of the constants is non-zero

- A shear can also be interpreted as a non-uniform scale along a rotated set of axes
- Shears are sometimes used in computer graphics for simple deformations or cartoon-like effects

Generalized 4 x 4 transformation matrix in homogeneous coordinates

$$P' = \mathbf{M} \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

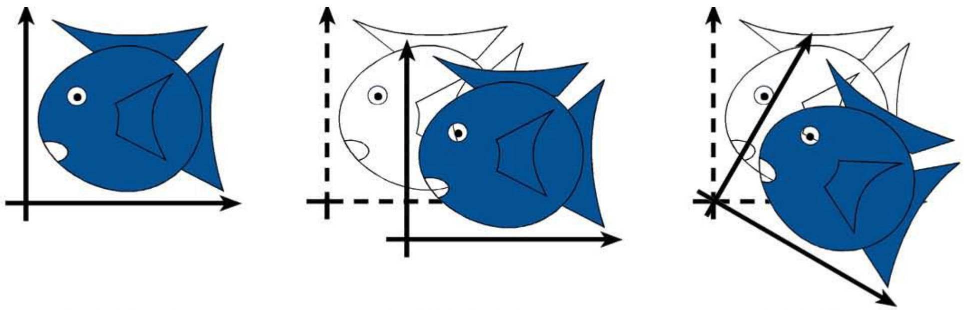
Linear Transformations

Translations

Perspective Projection

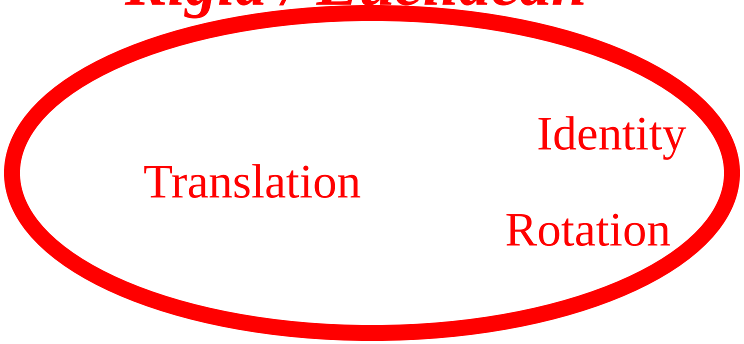
Rigid-Body / Euclidean Transformations

- Move the objects leaving shape and dimension unchanged
- Preserves distances
- Preserves angles



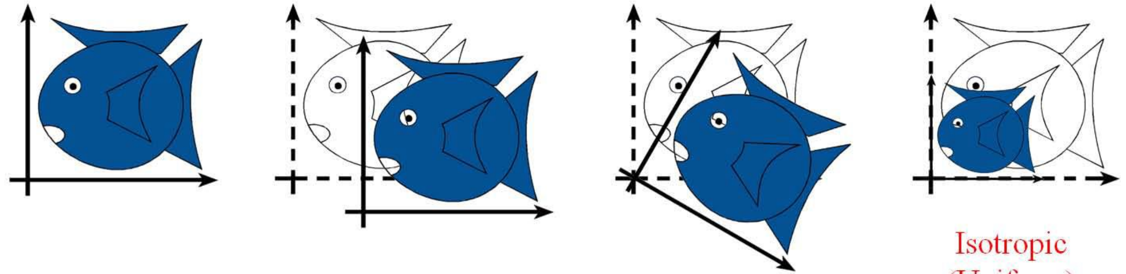
Identity Translation Rotation

Rigid / Euclidean

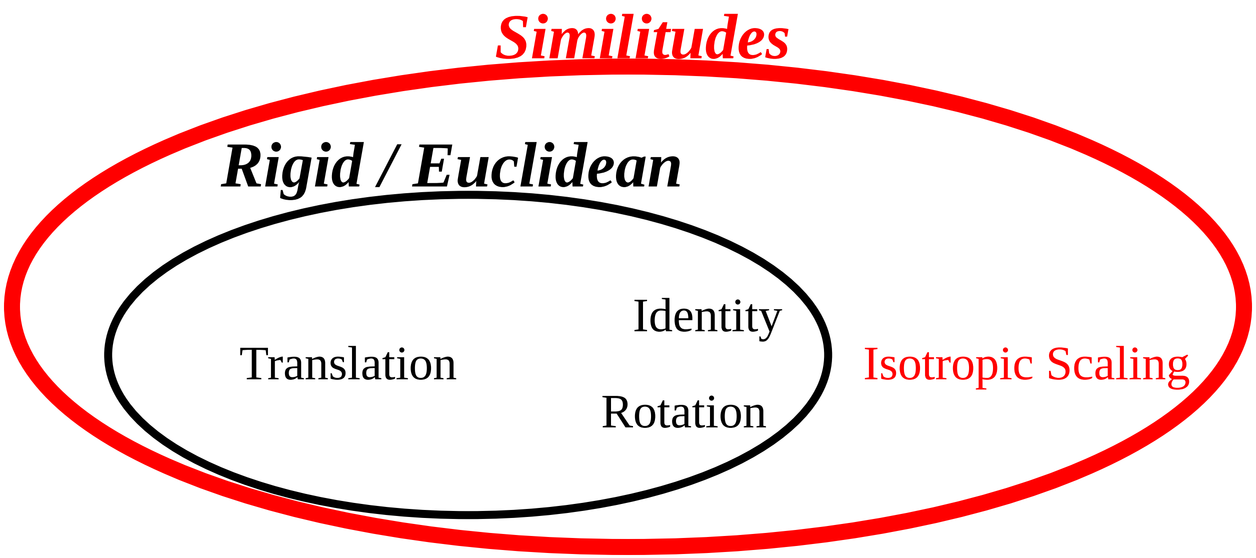


Similitudes / Similarity Transforms

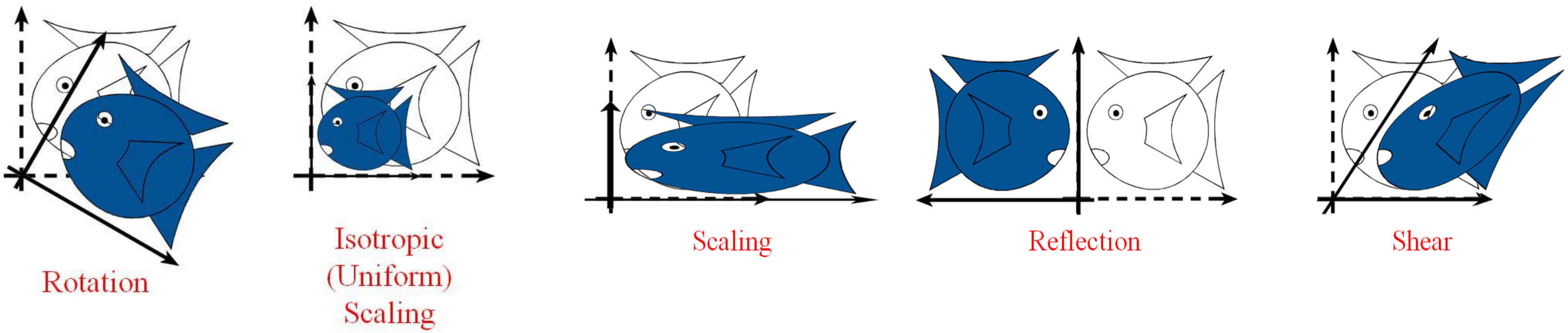
- Preserves angles

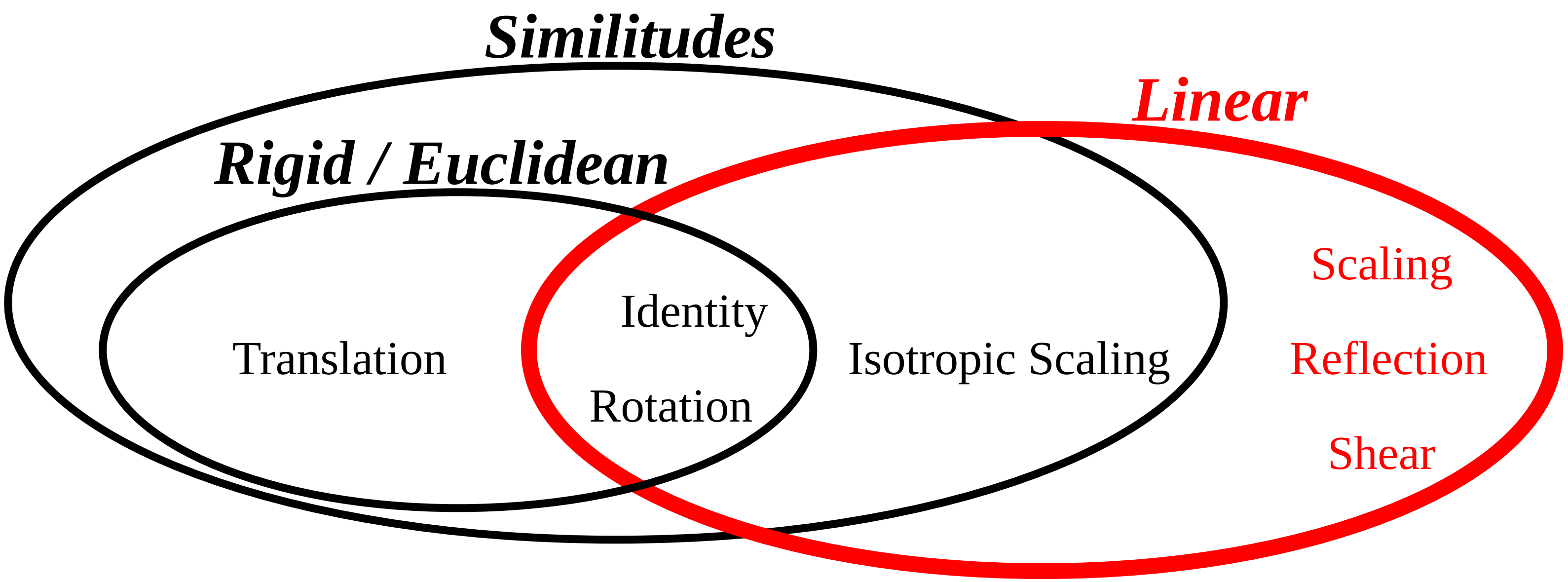


Identity — Translation — Rotation — Isotropic (Uniform) Scaling



Linear Transformations

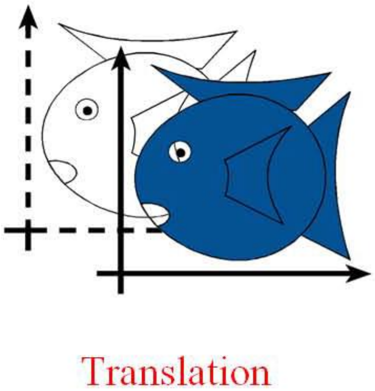


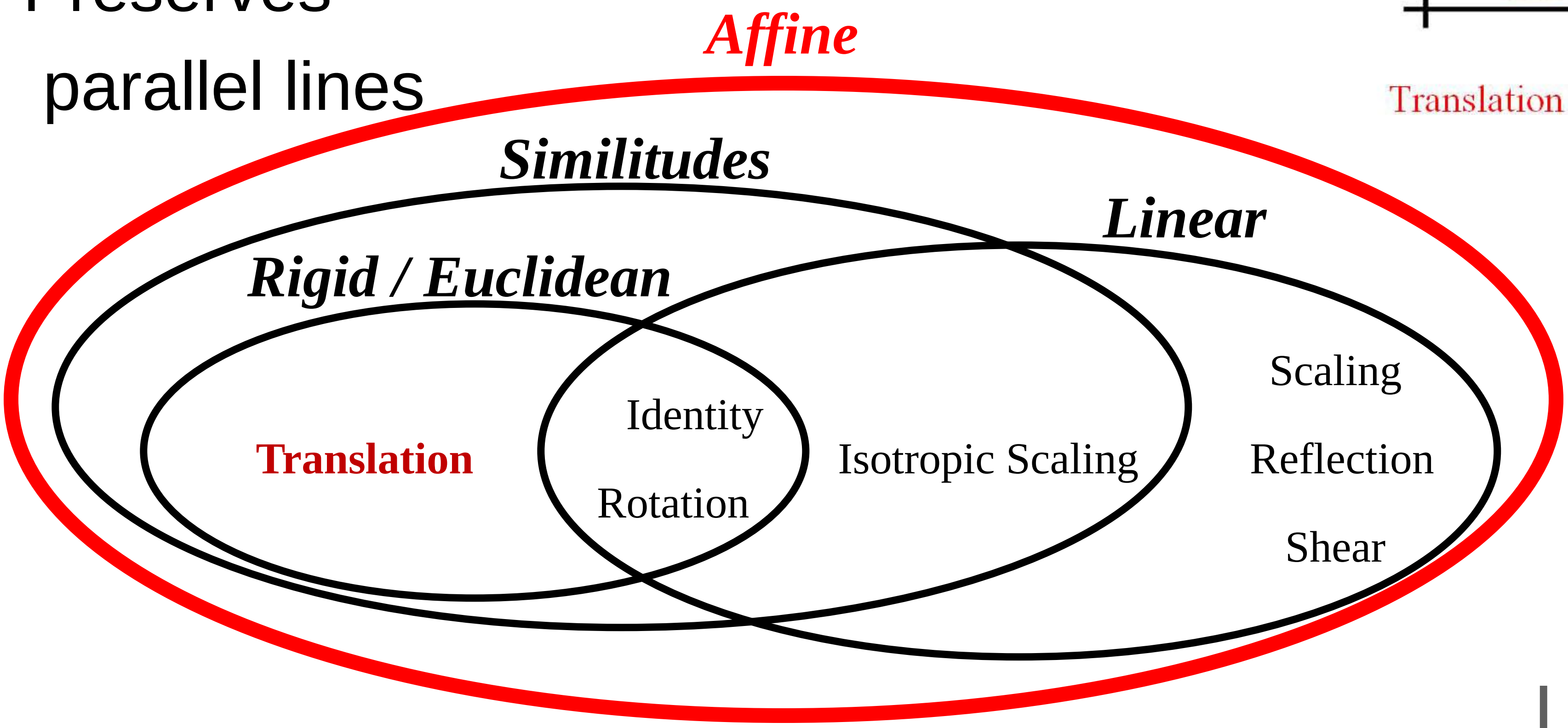


Courtesy of Prof. Fredo Durand

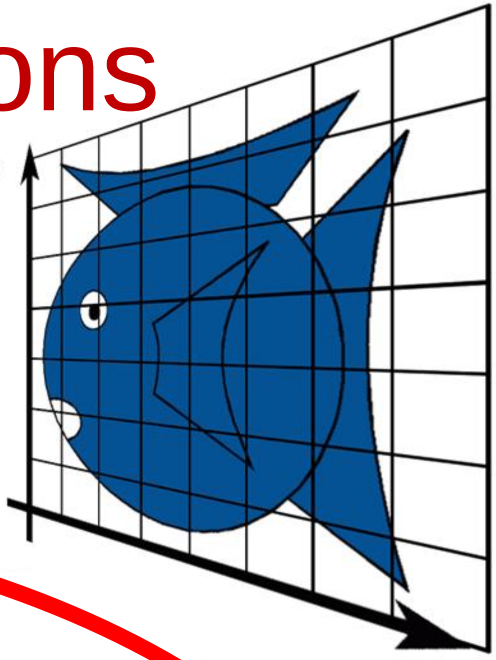
Affine Transformations

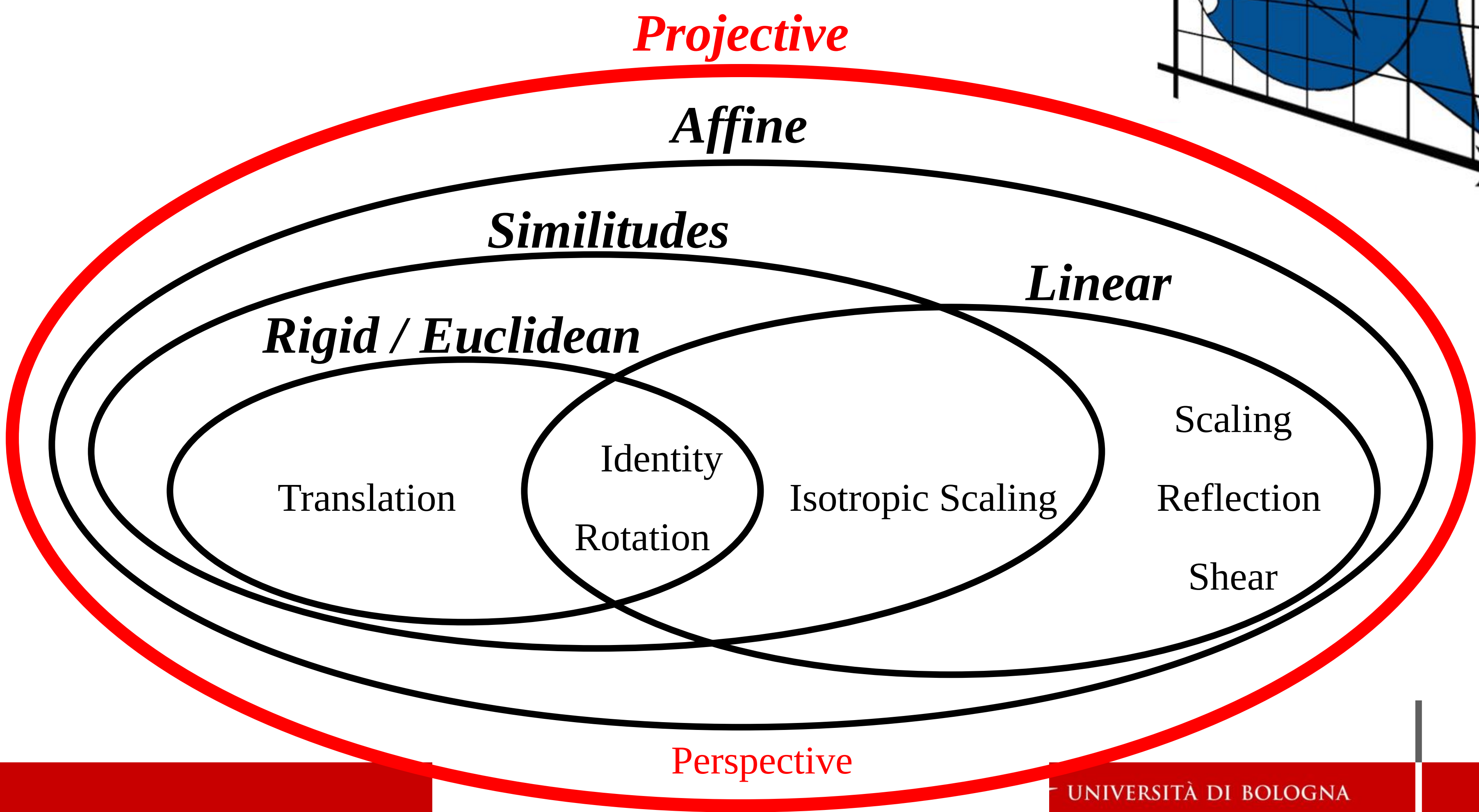
- Include all linear transformations plus translation
- Preserves parallel lines





Projective Transformations





Properties of Transformations

Type / Preserves	Rigid Body: Rotation & translation	Linear: General 3x3 matrix	Affine: Linear + translation	Projective: 4x4 matrix with last row $\neq(0,0,0,1)$
Lengths	Yes	No	No	No
Angles	Yes	No	No	No
Parallelness	Yes	Yes	Yes	No
Straight lines	Yes	Yes	Yes	Yes

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Serena Morigi

Dipartimento di Matematica

serena.morigi@unibo.it

http://www.dm.unibo.it/~morigi