

Mathematical Fundamentals of Computer Graphics

(Review)

It's Boring, but...

- Basic Linear Algebra
- Coordinate System
- 2D Transformation
- 3D Transformation
- Composition of transformations



Basic Linear Algebra Review

- **Basic object types**
 - Scalar
 - Vector
 - Point
- **Spaces**
 - Vector space
 - Affine space
 - Euclidean space
- **Matrix operations**

Scalar

- **A real number**
- **No geometric properties**
- **Usually used as units of measurement**
Ex: length of line, angle of rotation, ...
- **Basic operations:**
 - Scalar addition
 - Scalar multiplication

Basic Scalar Operations

Two scalars operation still forms a scalar

$$\alpha + \beta = \beta + \alpha$$

$$\alpha + (\beta + \lambda) = (\alpha + \beta) + \lambda$$

$$\alpha * \beta = \beta * \alpha$$

$$\alpha * (\beta * \lambda) = (\alpha * \beta) * \lambda$$

$$\alpha * (\beta + \lambda) = (\alpha * \beta) + (\alpha * \lambda)$$

$$\alpha + 0 = 0 + \alpha = \alpha$$

$$\alpha * 1 = 1 * \alpha = \alpha$$

$$\alpha + (-\alpha) = 0$$

$$\alpha * \alpha^{-1} = 1$$

Vector

- **A quantity with direction and magnitude**
Ex: directed line segment, velocity, ...
- **Have no position (but we often reference them from the origin)**
- **Basic operations:**
 - Addition
 - Multiplication

Basic Vector Operations

Vector operations still form a vector

- scalar-vector multiplication \Rightarrow vector

$$\alpha * \mathbf{v}$$

- vector-vector addition \Rightarrow vector

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

$$\alpha * (\mathbf{u} + \mathbf{v}) = \alpha * \mathbf{u} + \alpha * \mathbf{v}$$

$$(\alpha + \beta) * \mathbf{u} = \alpha * \mathbf{u} + \beta * \mathbf{u}$$

- there must exist a zero vector ($\mathbf{0}$)

$$\mathbf{u} + \mathbf{0} = \mathbf{u}$$

$$\mathbf{u} + (-\mathbf{u}) = \mathbf{0}$$

Geometric View of Vector



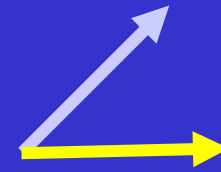
Vector has magnitude
and direction

A



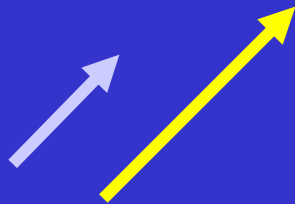
Same directions
same magnitudes

$A = B$



Same magnitudes
different directions

$A \neq B$



Same directions
different magnitudes

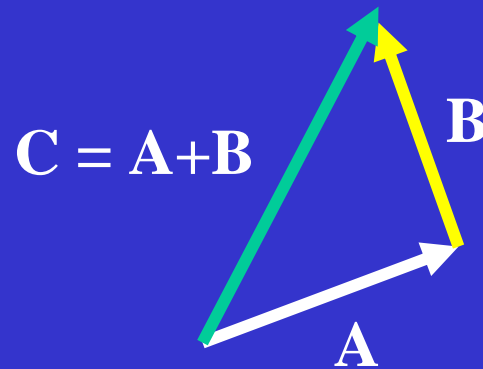
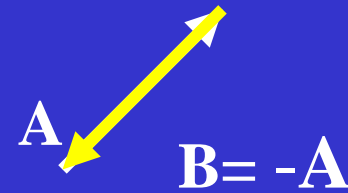
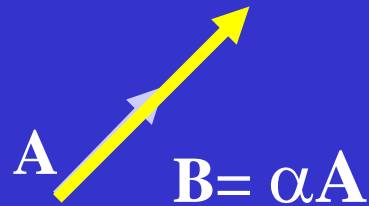
$A = 2B$



Same magnitudes
opposite directions

$A = -B$

Geometric View of Vector



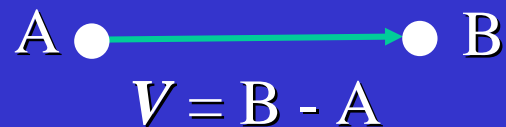
Head-to-tail rule

Point

- **A location in the space**
- **Only geometric property is its location**
- **Have no mathematical size (?)**
- **Basic operations:**
 - Subtraction (addition)

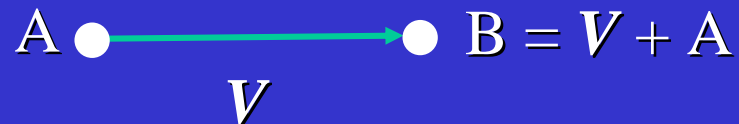
Basic Point Operations

- point-point subtraction \Rightarrow vector



A \bullet $\xrightarrow{\text{green}}$ \bullet B
 $V = B - A$

- vector-point addition \Rightarrow point



A \bullet $\xrightarrow{\text{green}}$ \bullet B = $V + A$
 V

Spaces

Space

- Contains a set of objects
- Define a set of mathematic operations

- Vector space
- Affine space
- Euclidean space

Vector Space

- Contains two types of objects: *scalar* and *vector*
- Defines mathematic operations (vector and scalar) on them

Ex:

scalar addition and multiplication \Rightarrow scalar

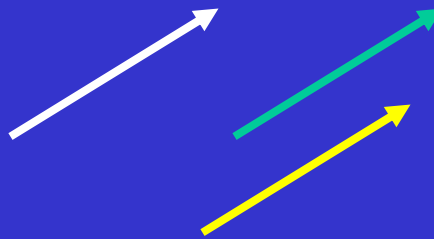
scalar-vector multiplication \Rightarrow vector

vector-vector addition \Rightarrow vector

- Must exist a zero vector (0)

Vector Space

Vector Space can describe the direction and magnitude of a quantity, but lacks any geometric concepts, such as location, distance...



We need additional constraint(s)

Affine Space

- Contains three types of objects: *scalar*, *vector*, and *point*
- Defines mathematic operations (vector, point, and scalar) on them

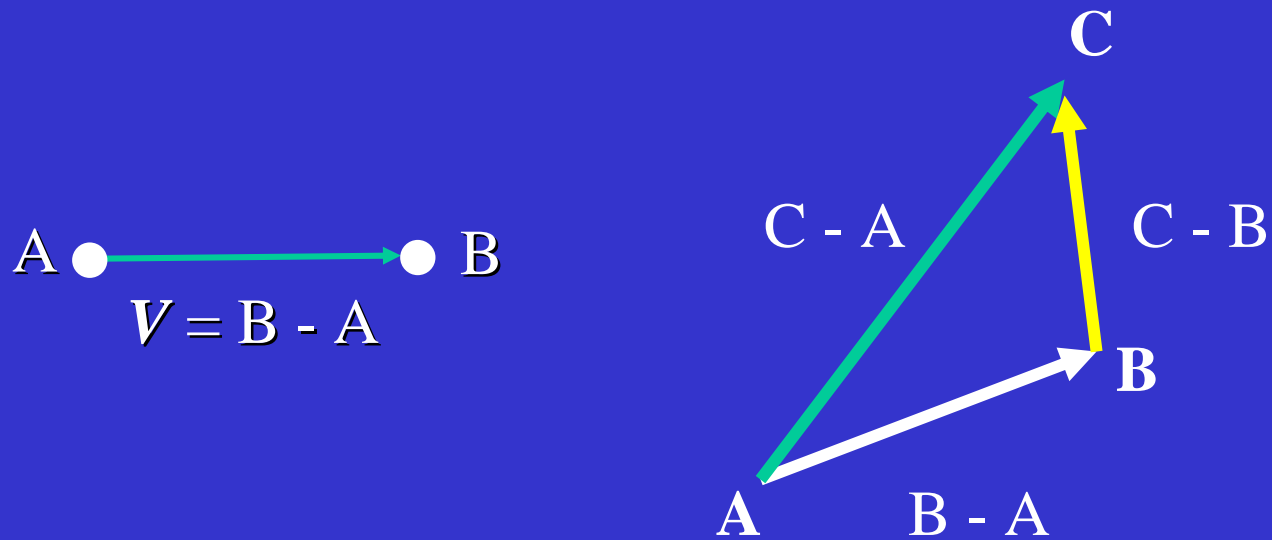
Ex:

point-point subtraction \Rightarrow vector

vector-point addition \Rightarrow point

Affine Space (vector)

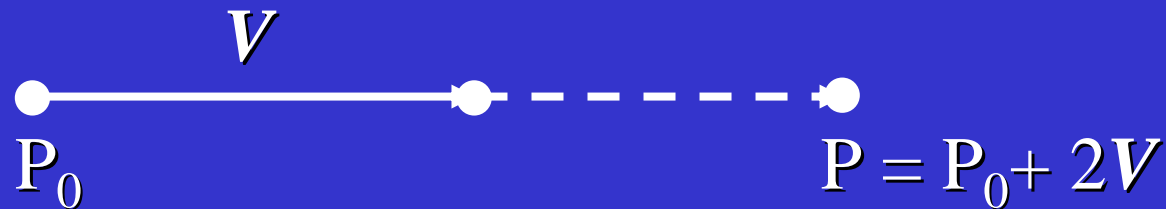
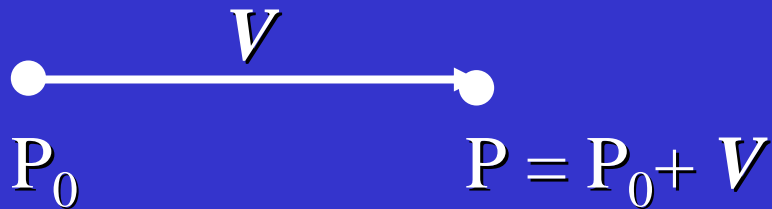
- point-point subtraction \Rightarrow vector



Head-to-tail rule

Affine Space (line)

- vector-point addition \Rightarrow point (line)



.....

Parametrized line

Let: P_0 is a point, V is a arbitrary vector, and α is a arbitrary scalar

Then: $P(\alpha) = P_0 + \alpha V$, is a point for any value of α . All these points lie on a line that defined by $P(\alpha)$

This form is called the **parametric form of line**

Parametrized line

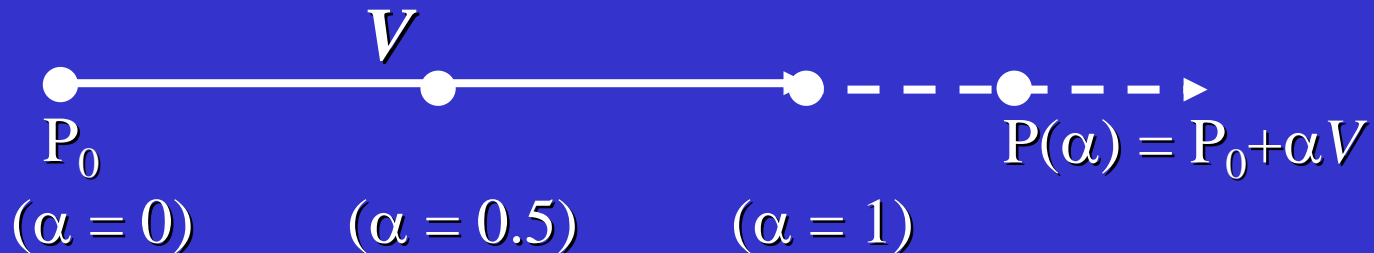
We can generate points on the line by varying the parameter α .

Ex:

$$\alpha = 0, \quad P(\alpha) = P_0$$

$$\alpha = 1, \quad P(\alpha) = P_0 + V$$

$$\alpha = 0.5, \quad P(\alpha) = P_0 + 0.5V$$



Affine Space (Basis, Frame)

- Let vectors $v_0 v_1 v_2 \dots v_{N-1}$. If the only set of scalars $a_0 a_1 \dots a_{N-1}$, such as

$$a_0 v_0 + a_1 v_1 + a_2 v_2 + \dots a_{N-1} v_{N-1} = 0$$

is

$$a_0 = a_1 = a_2 = \dots a_{N-1} = 0$$

Then, the vectors $v_0 v_1 v_2 \dots v_{N-1}$ are *linearly independent*

- The greatest number of linearly independent vectors that we can find in the space gives the *dimension* of the space.
- If the dimension is N , any set of N linearly independent vectors form a *basis* of the space
- Let P_0 is a point, and $v_0 v_1 v_2 \dots v_{N-1}$ are basis vectors. Then $\{P_0, v_0 v_1 v_2 \dots v_{N-1}\}$ defines a *frame* of the affine space

Linear Combination

Let P_0 is a point, and $v_0 v_1 v_2 \dots v_{N-1}$ are vectors that are linearly independent (basis) in affine space

Then

1. A arbitrary vector can be written uniquely as

$$v = a_0 v_0 + a_1 v_1 + a_2 v_2 + \dots a_{N-1} v_{N-1}$$

2. An arbitrary point can be written uniquely as

$$P = P_0 + a_0 v_0 + a_1 v_1 + a_2 v_2 + \dots a_{N-1} v_{N-1}$$

Euclidean Space

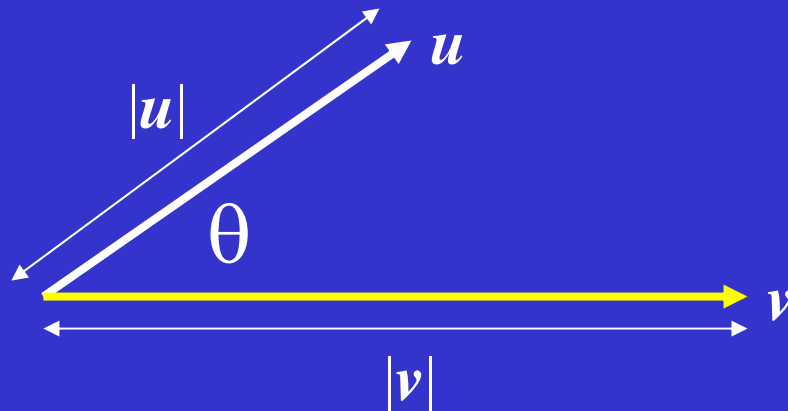
- Affine Space contains the necessary elements for geometric concepts, but lacks the **length** concept, such as how far apart two points, what the length of a vector...
- Defines new mathematic operations on them Ex:

Dot (Inner) Product, Cross Product

Dot Product

Let \mathbf{u} , \mathbf{v} are two vectors, then

$$\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}| |\mathbf{v}| \cos\theta$$



The dot product of two vectors is a scalar

Dot Product Properties

Vector Dot product must satisfy the following properties:

- $\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{u}$
- $(\alpha \mathbf{u} + \beta \mathbf{v}) \cdot \mathbf{w} = \alpha \mathbf{u} \cdot \mathbf{w} + \beta \mathbf{v} \cdot \mathbf{w}$
- $\mathbf{v} \cdot \mathbf{v} > \mathbf{0}$ ($\mathbf{v} \neq \mathbf{0}$)
- $\mathbf{0} \cdot \mathbf{0} = \mathbf{0}$
- $|\mathbf{u}|^2 = \mathbf{u} \cdot \mathbf{u}$ ($|\mathbf{u}| = \sqrt{\mathbf{u} \cdot \mathbf{u}}$)

Vector Angle Measurement

Let u, v are two vectors, then

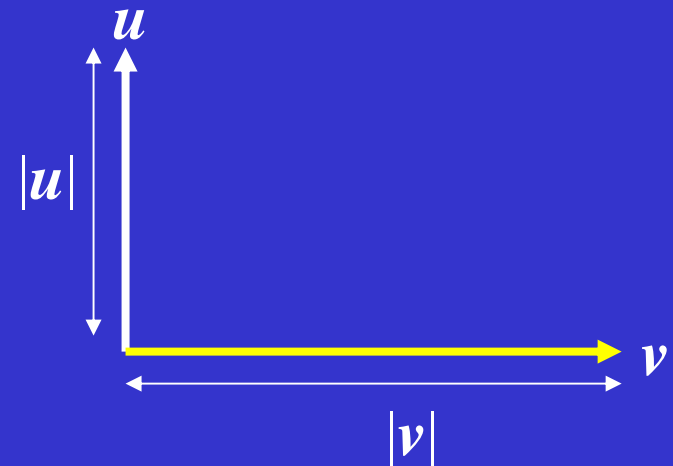
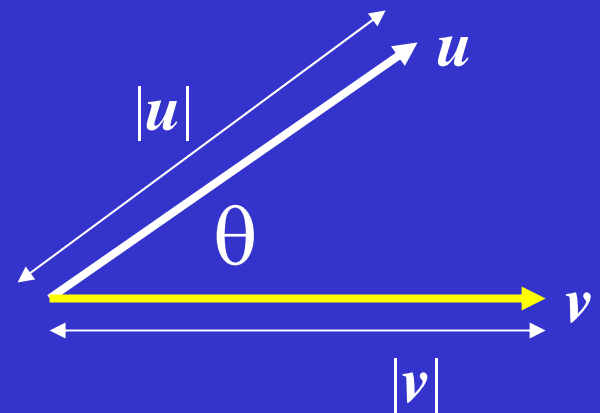
$$\cos\theta = \frac{u \cdot v}{|u| |v|}$$

measures the angle between the vectors

If $u \cdot v = 0$, then

$$\theta = 90^\circ$$

the vectors are **orthogonal**



Vector Magnitude Measurement

- **Vector magnitude**

$$|u|^2 = u \cdot u \quad (|u| = \sqrt{u \cdot u})$$

- **Vector normalization**

Given a vector u , we want to create a **unit vector** that has a magnitude of **1** and has the same direction as u , e.g

$$u_{\text{normalized}} = \frac{u}{|u|}$$

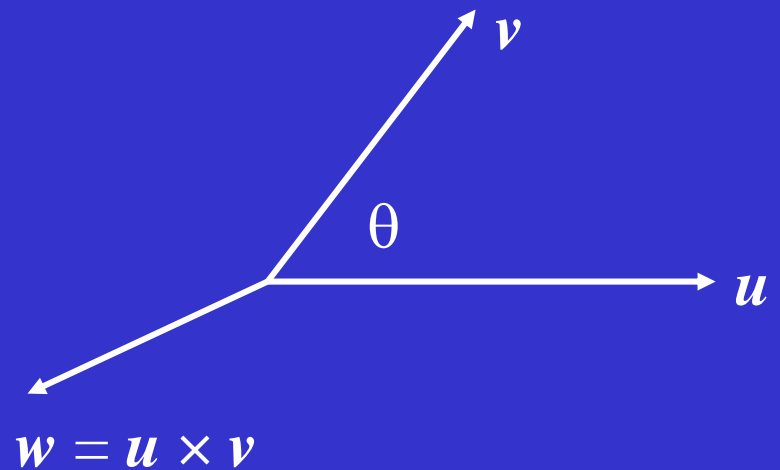
Cross Product

Let \mathbf{u} , \mathbf{v} are two non-parallel vectors, then the *cross product* gives a third vector, \mathbf{w} , that is orthogonal to both \mathbf{u} , \mathbf{v} , e.g

$$\mathbf{w} = \mathbf{u} \times \mathbf{v}$$

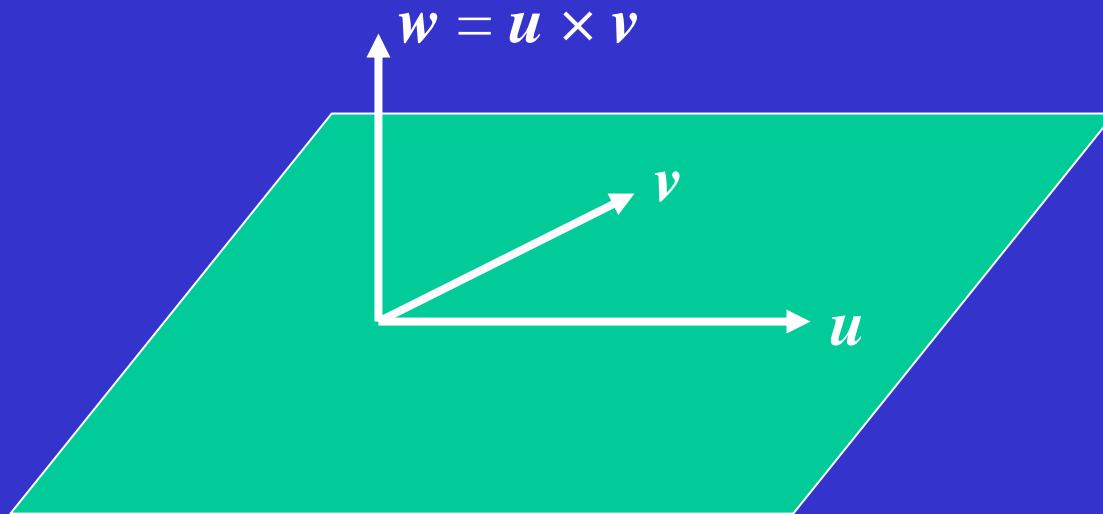
$$\mathbf{w} \cdot \mathbf{u} = \mathbf{w} \cdot \mathbf{v} = 0$$

$$|\sin\theta| = \frac{|\mathbf{u} \times \mathbf{v}|}{|\mathbf{u}| |\mathbf{v}|}$$



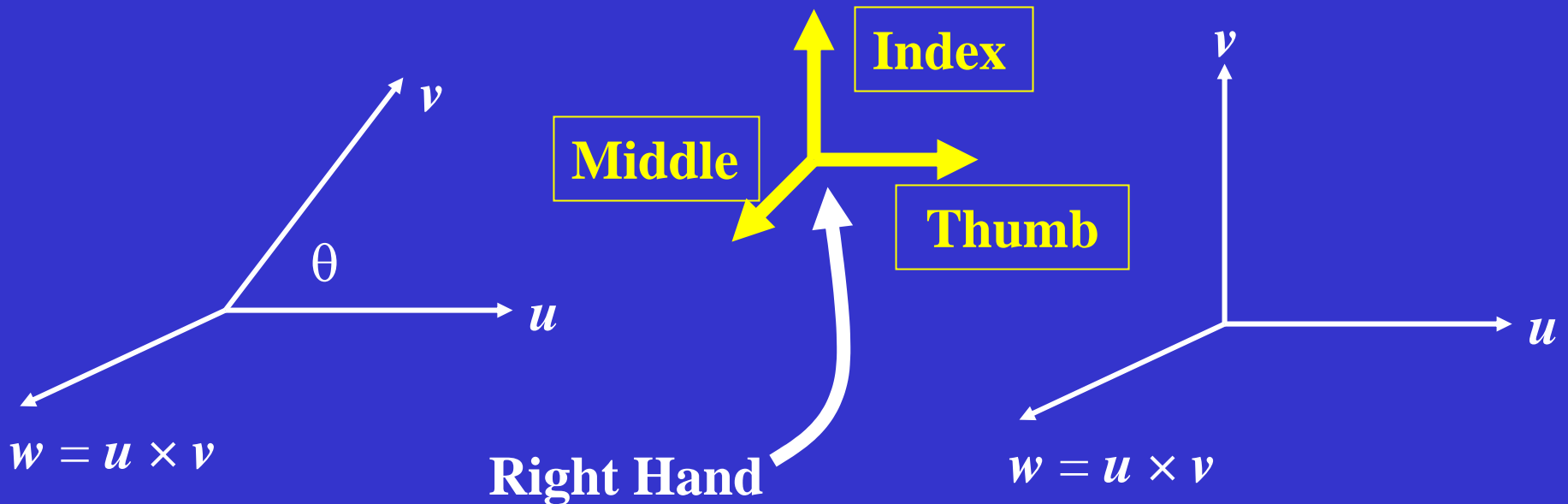
Cross Product

- The cross product of two vectors is a vector and gives a **normal** to the plane defined by those two vectors



Right-handed Coordinate

Note: Cross product defines a **Right-handed** coordinate system



Matrix

- A matrix is an $N \times M$ array of scalars, arranged conceptually as N rows and M columns

$$\mathbf{A} = [a_{ij}], \quad i = 1, 2, \dots, N; \quad j = 1, 2, \dots, M$$

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

- Square matrix

$$N = M$$

Matrix

- **Identity matrix**

a square matrix with 1s on the diagonal and 0s elsewhere

$$\mathbf{I} = [a_{ij}], \quad a_{ij} = 1 \text{ if } i = j; \quad a_{ij} = 0 \text{ otherwise}$$

- **Transpose matrix**

$$\mathbf{A}^T = [a_{ji}], \quad i = 1, 2, \dots, N; \quad j = 1, 2, \dots, M$$

- **Inverse matrix**

$$\mathbf{A}^{-1} \mathbf{A} = \mathbf{I}$$

Matrix Operations

- Matrix addition
- Matrix multiplication
- Matrix transpose
- Matrix Inverse

Matrix + Matrix

If matrix A and B have the same dimensions $N \times M$, then they are **conformal for addition**, and the dimension of $A+B$ is also $N \times M$

$$C_{ij} = A_{ij} + B_{ij}$$

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad B = \begin{bmatrix} x & y \\ z & w \end{bmatrix}$$

Does $A + B = B + A$?

$$A+B = \begin{bmatrix} a + x & b + y \\ c + z & d + w \end{bmatrix}$$

Matrix * Matrix

If the no. of columns of A equals the no. of rows of B , then A and B are **conformal for multiplication**

$$C_{ij} = \sum_{r=1}^k A_{ir} B_{rj}$$

If A is $N \times K$ and B is $K \times M$, then $A*B$ is $N \times M$

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad B = \begin{bmatrix} x & y \\ z & w \end{bmatrix}$$

Does $A * B = B * A$?

$$A * B = \begin{bmatrix} ax + bz & ay + bw \\ cx + dz & cy + dw \end{bmatrix}$$

What does the identity do?

Matrix * Vector (Matrix)

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

$$x' = ax + by$$

$$y' = cx + dy$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$x' = ax + by + cz$$

$$y' = dx + ey + fz$$

$$z' = gx + hy + iz$$

Matrix Transpose

The **transpose** of a matrix is A^T

$$A^T = [a_{ij}]^T = [a_{ji}]$$

- $(\alpha A)^T = \alpha A^T$
- $(A + B)^T = A^T + B^T$
- $(A^T)^T = A$

Matrix Inverse

The **inverse** of a matrix is A^{-1}

if $A\mathbf{x} = \mathbf{b}$, then $\mathbf{x} = A^{-1}\mathbf{b}$

- $A^{-1}A = AA^{-1} = I$
- $(AB)^{-1} = B^{-1}A^{-1}$

Matrix Inverse

These statements are equivalent:

1. A^{-1} does not exist
2. A is *singular*
3. There exists a non-zero vector \mathbf{x} such that

$$\mathbf{Ax} = \mathbf{0}$$

4. $\det(\mathbf{A}) = 0$

Matrix Properties

- $A + B = B + A$
- $A + (B + C) = (A + B) + C$
- $A * (B * C) = (A * B) * C$
- $A * B \neq B * A$
- $A * I = I * A = A$

Again: Dot Product

Let vectors

$\mathbf{u} = (u_1, u_2, u_3, \dots, u_N)$ and $\mathbf{v} = (v_1, v_2, v_3, \dots, v_N)$, then the dot product is

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

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + u_3 v_3$$

$$\mathbf{u} = (u_1, u_2, u_3) \text{ and } \mathbf{v} = (v_1, v_2, v_3)$$

Again: Cross Product

Let vectors

$\mathbf{u} = (u_1, u_2, u_3)$ and $\mathbf{v} = (v_1, v_2, v_3)$, then
the **cross product** is

$$\mathbf{w} = \mathbf{u} \times \mathbf{v} = \begin{pmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{pmatrix}$$

Again: Cross Product

We can also represent the cross product as

$$\mathbf{w} = \mathbf{u} \times \mathbf{v} = \begin{pmatrix} \mathbf{0} & -u_3 & u_2 \\ u_3 & \mathbf{0} & -u_1 \\ -u_2 & u_1 & \mathbf{0} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

skew symmetric matrix

Summary (Basic Object Types)

- **Scalar**
 - A real number
 - Has no geometric properties
- **Vector**
 - A quantity with direction and magnitude
 - Has no position
- **Point**
 - A location in the space
 - Has no direction (size)

Summary (Space)

- **Vector space**

- Contains two types of objects: *scalars* and *vectors*
- Can describe the direction and magnitude of a quantity, but lacks any geometric concepts, such as location...

- **Affine space**

- Contains three types of objects: *scalars*, *vectors*, and *points*
- Has the necessary elements for geometric concepts, but lacks the length concept, such as how the length of a vector...

- **Euclidean space**

- Contains three types of objects: *scalars*, *vectors*, and *points* (strictly speaking, a Euclidean space needs to contain only *scalar* and *vectors*)
- Defines new operations. Ex: dot product, cross product...

Coordinate System

- Vector/point coordinate representation
- Homogeneous Coordinates

Linear Combination

Let P_0 is a point, and $v_0 v_1 v_2 \dots v_{N-1}$ are vectors that are linearly independent (basis) in affine space

Then

1. A arbitrary vector can be written uniquely as

$$v = a_0 v_0 + a_1 v_1 + a_2 v_2 + \dots a_{N-1} v_{N-1}$$

2. An arbitrary point can be written uniquely as

$$P = P_0 + a_0 v_0 + a_1 v_1 + a_2 v_2 + \dots a_{N-1} v_{N-1}$$

Vector Representation

Within a given frame (basis)

- Every vector can be written uniquely as

$$v = a_0 v_0 + a_1 v_1 + a_2 v_2 + \dots + a_{N-1} v_{N-1}$$

- We say that vector v has the representation $(a_0 a_1 \dots a_{N-1})$ with respect to the basis $(v_0 v_1 v_2 \dots v_{N-1})$. And the scalars $a_0 a_1 \dots a_{N-1}$ are the *components* of v with respect to the basis.
- We can write the vector as

$$v = a^T \begin{pmatrix} v_0 \\ v_1 \\ \dots \\ v_{N-1} \end{pmatrix} \quad \text{where } a = \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_{N-1} \end{pmatrix} \quad \text{column matrix representation}$$

Point Representation

Within a given frame

- Every point can be written uniquely as

$$P = P_0 + a_0 v_0 + a_1 v_1 + a_2 v_2 + \dots a_{N-1} v_{N-1}$$

- We say that point P has the representation $(a_0 a_1 \dots a_{N-1})$ with respect to the frame $(P_0 v_0 v_1 v_2 \dots v_{N-1})$. And the scalars $a_0 a_1 \dots a_{N-1}$ are the *components* of P with respect to the frame.
- We can write the point as

$$P = a^T \begin{pmatrix} v_0 \\ v_1 \\ \dots \\ v_{N-1} \\ P_0 \end{pmatrix} \quad \text{where } a = \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_{N-1} \\ 1 \end{pmatrix} \quad \text{column matrix representation}$$

Coordinate System

- A coordinate system consists of a specific point P_0 , called the *origin*, and basis vectors $v_0 v_1 v_2 \dots v_{N-1}$
- Normally, the basis vectors that span the coordinate system are *unit vectors*

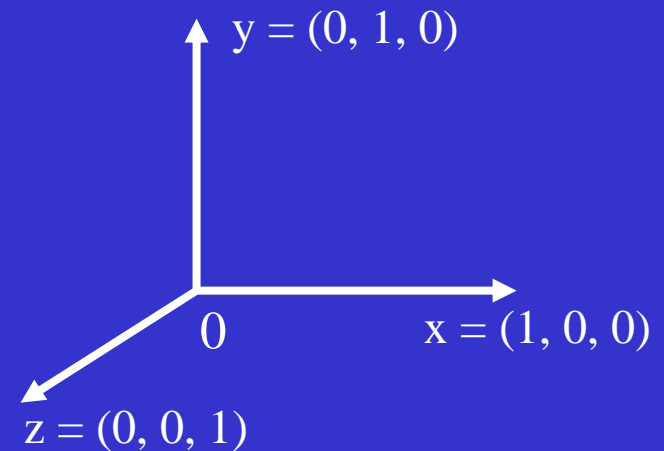
Ex:

$$v_0 = (1, 0, 0, \dots, 0),$$

$$v_1 = (0, 1, 0, \dots, 0),$$

.....

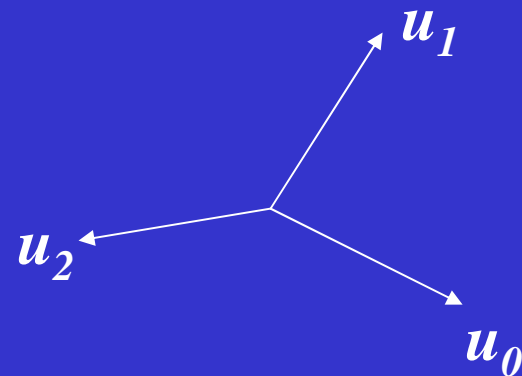
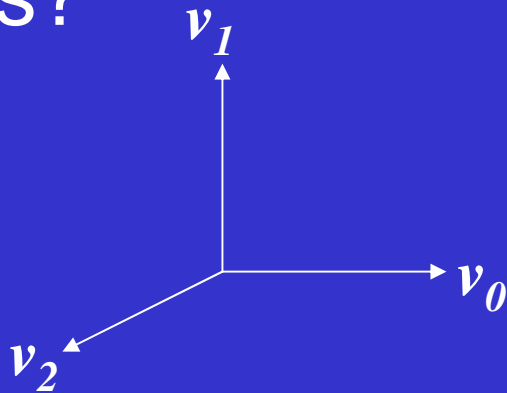
$$v_{N-1} = (0, 0, 0, \dots, 1)$$



A Coordinate Transformation

Let $\{v_0 v_1 v_2\}$ and $\{u_0 u_1 u_2\}$ are two basis vectors

Find how the representation of a vector changes when we change the basis vectors?



A Coordinate Transformation

- Represent basis vector $\{u_0 u_1 u_2\}$ in term of the basis $\{v_0 v_1 v_2\}$ as

$$u_0 = m_{00} v_0 + m_{01} v_1 + m_{02} v_2$$

$$u_1 = m_{10} v_0 + m_{11} v_1 + m_{12} v_2$$

$$u_2 = m_{20} v_0 + m_{21} v_1 + m_{22} v_2$$

or
$$\begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} = \mathbf{M}_R \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix}$$

where
$$\mathbf{M}_R = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix}$$

Transformation matrix

$$\{v_0 v_1 v_2\} \Rightarrow \{u_0 u_1 u_2\}$$

A Coordinate Transformation

- Let vector w . Represent it in term of the basis vector $\{v_0 v_1 v_2\}$ as

$$w = a_0 v_0 + a_1 v_1 + a_2 v_2$$

or $w = a^T \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix}$

where $a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$

A Coordinate Transformation

- Similarly, represent vector w in term of the basis vector $\{u_0 u_1 u_2\}$ as

$$w = b_0 u_0 + b_1 u_1 + b_2 u_2$$

or $w = \mathbf{b}^T \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix}$

where $\mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$

A Coordinate Transformation

- Combine above, we have

$$\mathbf{w} = \mathbf{b}^T \begin{bmatrix} u_0 \\ u_1 \\ u_2 \end{bmatrix} = \mathbf{b}^T \mathbf{M}_R \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix} = \mathbf{a}^T \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix}$$

Thus,

$$\mathbf{a} = \mathbf{M}_R^T \mathbf{b}$$

or

$$\mathbf{b} = (\mathbf{M}_R^T)^{-1} \mathbf{a}$$
$$\mathbf{M}_R = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix}$$

A Coordinate Transformation

- Conclusion

If we have transformation matrix \mathbf{M}_R

$$\mathbf{M}_R = \begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{pmatrix}$$

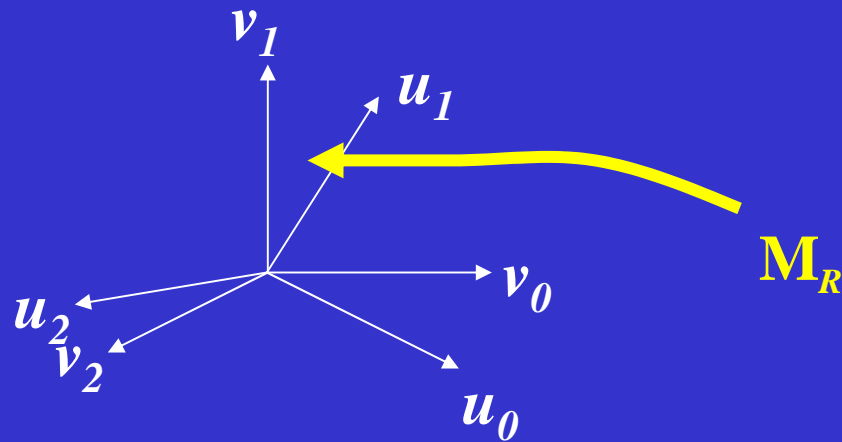
such that

$$\begin{pmatrix} u_0 \\ u_1 \\ u_2 \end{pmatrix} = \mathbf{M}_R \begin{pmatrix} v_0 \\ v_1 \\ v_2 \end{pmatrix} \quad \{v_0 v_1 v_2\} \Rightarrow \{u_0 u_1 u_2\}$$

then

$$\mathbf{a} = \mathbf{M}_R^T \mathbf{b} \quad \text{or} \quad \mathbf{b} = (\mathbf{M}_R^T)^{-1} \mathbf{a}$$

A Coordinate Transformation



Note: Origin unchanged!

Rotation or Scaling transformation

Check Vector/Point Representation

Within a given frame

- The representation of a particular *vector* in a frame requires 3 scalars

$$\mathbf{v} = \mathbf{a}^T \begin{pmatrix} v_0 \\ v_1 \\ v_2 \end{pmatrix}$$

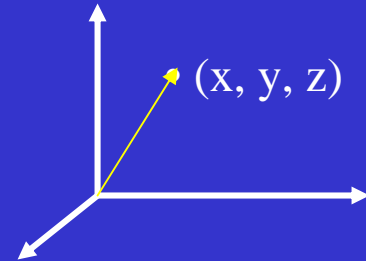
- The representation of a particular *point* in a frame requires 3 scalars and a reference point (the knowledge of where the origin is located)

$$\mathbf{P} = \mathbf{a}^T \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ P_0 \end{pmatrix}$$

Check Vector/Point Representation

- What the difference between a vector and point?

$$\mathbf{v} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$



- Different dimensions

$$\mathbf{v} = \mathbf{a}^T \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix}_{3 \times 3} \quad \mathbf{P} = \mathbf{a}^T \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ P_0 \end{bmatrix}_{4 \times 4}$$

Can we make them same?

Make them Same

- Assume the multiplication of a point by the scalar 0 and 1 as

$$\mathbf{0} * \mathbf{P} = \mathbf{0}, \quad \mathbf{1} * \mathbf{P} = \mathbf{P}$$

- The representation of a particular *point* in a frame requires 3 scalars and a reference point

$$\mathbf{P} = [a_0 \ a_0 \ a_0 \ \mathbf{1}] \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \mathbf{P}_0 \end{pmatrix}$$

- The representation of a particular *vector* in a frame requires 3 scalars

$$\mathbf{v} = [a_0 \ a_0 \ a_0 \ \mathbf{0}] \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \mathbf{P}_0 \end{pmatrix}$$

Homogeneous Coordinates

In the *homogeneous coordinate system*, both the vector and point can be represented as 4 dimension column matrix

$$\text{Point} \quad P = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ 1 \end{pmatrix}$$

$$\text{Vector} \quad v = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ 0 \end{pmatrix}$$

Ordinary/Homogeneous Coordinates

- To go from **ordinary** to **homogeneous** coordinates

- If the object is a point, append a “1”
- If the object is a vector, append a “0”

- To go from **homogeneous** to **ordinary** coordinates

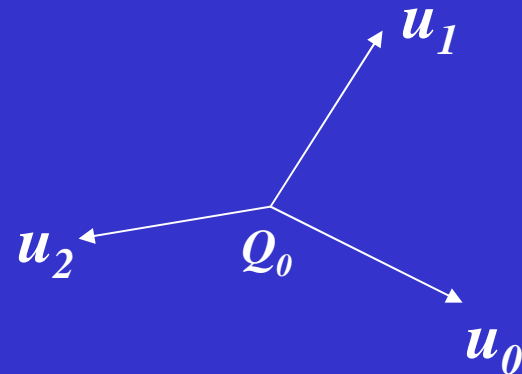
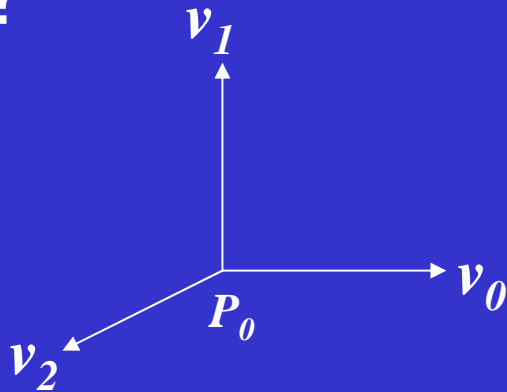
- If the object is a point, its final coordinate is 1, delete the “1”
- If the object is a vector, its final coordinate is 0, delete the “0”

$$P = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \mathbf{1} \end{pmatrix} \quad v = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \mathbf{0} \end{pmatrix}$$

Homogeneous Coordinate Transformation

Let $\{v_0 v_1 v_2 P_0\}$ and $\{u_0 u_1 u_2 Q_0\}$ are two frames

Find how the representation of a vector (point) changes when we change the frame?



Homogeneous Coordinate Transformation

- Represent frame $\{u_0 u_1 u_2 Q_0\}$ in term of the frame $\{v_0 v_1 v_2 P_0\}$ as

$$u_0 = m_{00}v_0 + m_{01}v_1 + m_{02}v_2$$

$$u_1 = m_{10}v_0 + m_{11}v_1 + m_{12}v_2$$

$$u_2 = m_{20}v_0 + m_{21}v_1 + m_{22}v_2$$

$$Q_0 = m_{30}v_0 + m_{31}v_1 + m_{32}v_2 + P_0$$

or

$$\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ Q_0 \end{pmatrix} = \mathbf{M}_H \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ P_0 \end{pmatrix}$$

where $\mathbf{M}_H =$

$$\begin{pmatrix} m_{00} & m_{01} & m_{02} & 0 \\ m_{10} & m_{11} & m_{12} & 0 \\ m_{20} & m_{21} & m_{22} & 0 \\ m_{30} & m_{31} & m_{32} & 1 \end{pmatrix}$$

Transformation matrix

$$\{v_0 v_1 v_2 P_0\} \Rightarrow \{u_0 u_1 u_2 Q_0\}$$

Homogeneous Coordinate Transformation

If we have two vectors or points a and b that are represented as homogeneous coordinates, we can use transformation matrix M_H to compute their representations directly

$$b^T \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ Q_0 \end{pmatrix} = b^T M_H \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ P_0 \end{pmatrix} = a^T \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ P_0 \end{pmatrix}$$

$$a = M_H^T b$$
$$b = (M_H^T)^{-1} a$$
$$M_H = \begin{pmatrix} m_{00} & m_{01} & m_{02} & 0 \\ m_{10} & m_{11} & m_{12} & 0 \\ m_{20} & m_{21} & m_{22} & 0 \\ m_{30} & m_{31} & m_{32} & 1 \end{pmatrix}$$

Homogeneous Coordinate Transformation

Let's check M_R and M_H

$$M_R = \begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{pmatrix}$$

3 x 3 matrix

$$M_H = \begin{pmatrix} m_{00} & m_{01} & m_{02} & 0 \\ m_{10} & m_{11} & m_{12} & 0 \\ m_{20} & m_{21} & m_{22} & 0 \\ m_{30} & m_{31} & m_{32} & 1 \end{pmatrix}$$

4 x 4 matrix

What does the last row (m_{30} m_{31} m_{32} 1) of M_H mean ?

Homogeneous Coordinate Transformation

- M_R

$$u_0 = m_{00}v_0 + m_{01}v_1 + m_{02}v_2$$

$$u_1 = m_{10}v_0 + m_{11}v_1 + m_{12}v_2$$

$$u_2 = m_{20}v_0 + m_{21}v_1 + m_{22}v_2$$

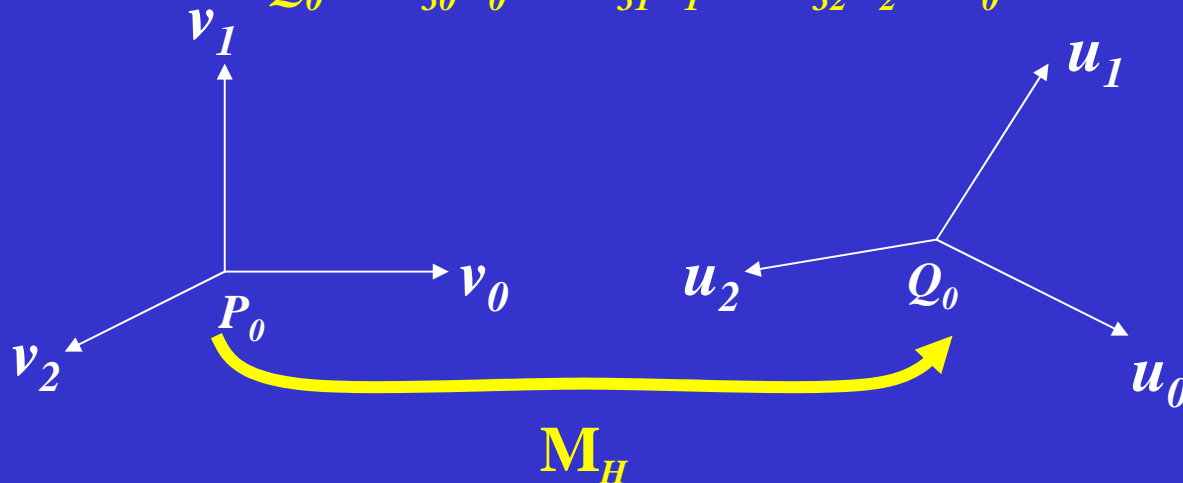
- M_H

$$u_0 = m_{00}v_0 + m_{01}v_1 + m_{02}v_2$$

$$u_1 = m_{10}v_0 + m_{11}v_1 + m_{12}v_2$$

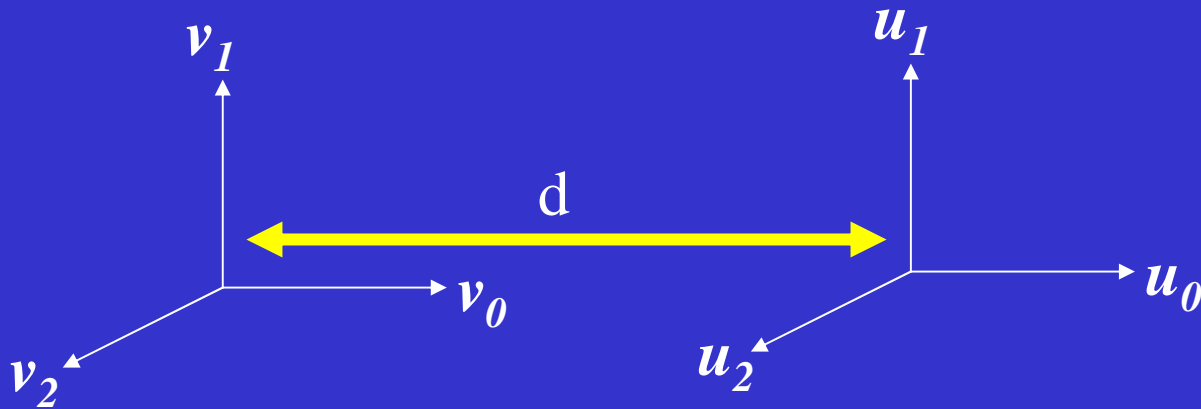
$$u_2 = m_{20}v_0 + m_{21}v_1 + m_{22}v_2$$

$$Q_0 = m_{30}v_0 + m_{31}v_1 + m_{32}v_2 + P_0$$



Example

$$\mathbf{M}_H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -d & 0 & 0 & 1 \end{pmatrix}$$



Frames in OpenGL

- OpenGL uses homogeneous coordinates for all its vertices
 - If the input to OpenGL is in the form (x, y, z) , OpenGL converts it immediately to $(x, y, z, 1)$
 - If the input to OpenGL is a 2D point (x, y) , OpenGL first appends a “0” for the z-component and then a “1”, to form $(x, y, 0, 1)$.
- All computations are done within OpenGL in 4D homogeneous coordinates

Frames in OpenGL

- In OpenGL, we can use **glLoadMatrix** function to load transformation matrix (4x4)
- Note: In OpenGL, the 4x4 transformation matrix is in the form

$$\mathbf{M} = \begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

If you want to send a matrix using `glLoadMatrix` with C array, be sure the order is Correct!

2D Transformation

- Affine transformations
- 2D transformation
 - Translate
 - Rotate
 - Scale
- Transformation in homogeneous coordinates

Transformations

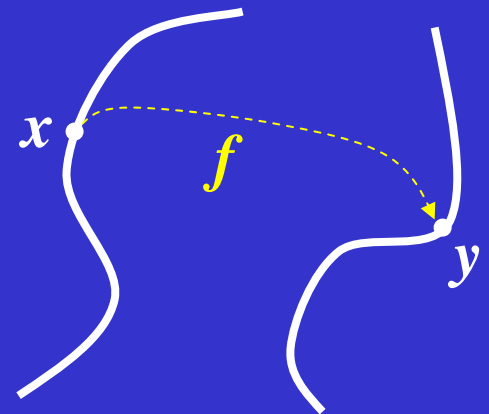
A Transformation is a function that takes variable(s) in a domain and maps that into another domain

$$y = f(x) \quad f: x \Rightarrow y$$

Ex:

$$y = \sin(x)$$

$$y = a_0x + a_1x^2 + \dots + a_{N-1}x^{N-1}$$



Transformations in Graphics

- Transformations make possible the projection of 3D objects onto 2D screen
- The graphics transformation process is analogous to take a photograph with a camera
- Every transformation can be thought as changing the representation of a vertex from one coordinate system to another

What is f (...)

- **Need restrictions**

- Linear functions

$$f(\alpha p + \beta q) = \alpha f(p) + \beta f(q)$$

- Geometry preserving

f : line (polygon, quadric) \Rightarrow a line (polygon, quadric)

Affine Transformations

- Affine transformation is a special class of transformation that is very important for graphical applications
- Affine transformation will *not alter* the type of object. A transformed line (polygon, quadric) is still a line (polygon, quadric)
- Any composition of affine transformations is still affine

Affine Transformations

- **Affine = line preserving**

Let: Parametrized line

$$\mathbf{p}(\alpha) = \mathbf{p}_0 + \alpha \mathbf{v}$$

where, \mathbf{p}_0 is a point, \mathbf{v} is an arbitrary vector, which both are represented in homogeneous coordinates, and α is an arbitrary scalar

Apply any affine transformation \mathbf{M} to both sides of the Parametrized line equation

$$\mathbf{M}\mathbf{p}(\alpha) = \mathbf{M}\mathbf{p}_0 + \alpha\mathbf{M}\mathbf{v}$$

Affine Transformations

- **Conclusions**

- Affine transformation is line preserving
 - Rotation, translation, scaling
- Any composition (concatenation) of affine transformations is still affine

Ex: A rotation followed by a translation followed by a projection preserves the lines and polygons

- Transformed line can be completely determined by transforming its endpoints (vertices)

2D Transformations

- A special case of 3D ($z = 0$)
- Represented points (vertices) as: (x,y) , or

- Matrix form:

$$(x, y) = \begin{bmatrix} x \\ y \end{bmatrix} \quad \text{or}$$

- Homogeneous coordinates

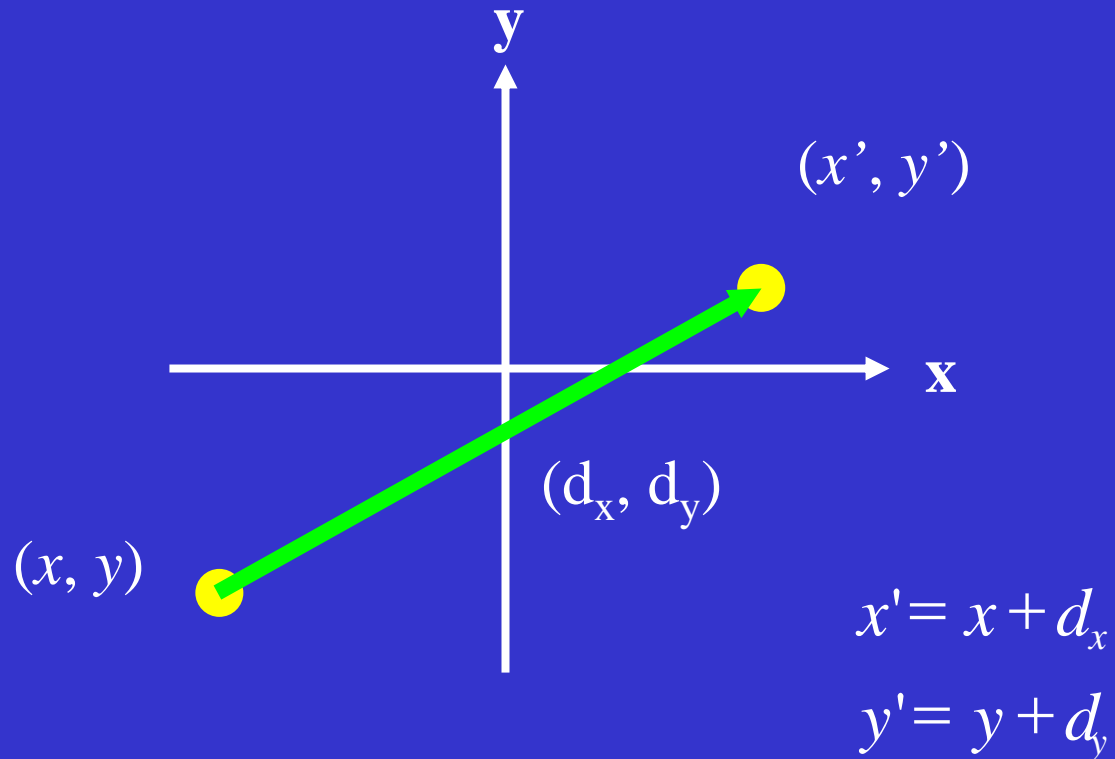
$$(x, y) = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translation

- **Translation**

an operation that re-positions points along a given straight-line path (the **translation direction**) from one coordinate location to another

Translation



Translation

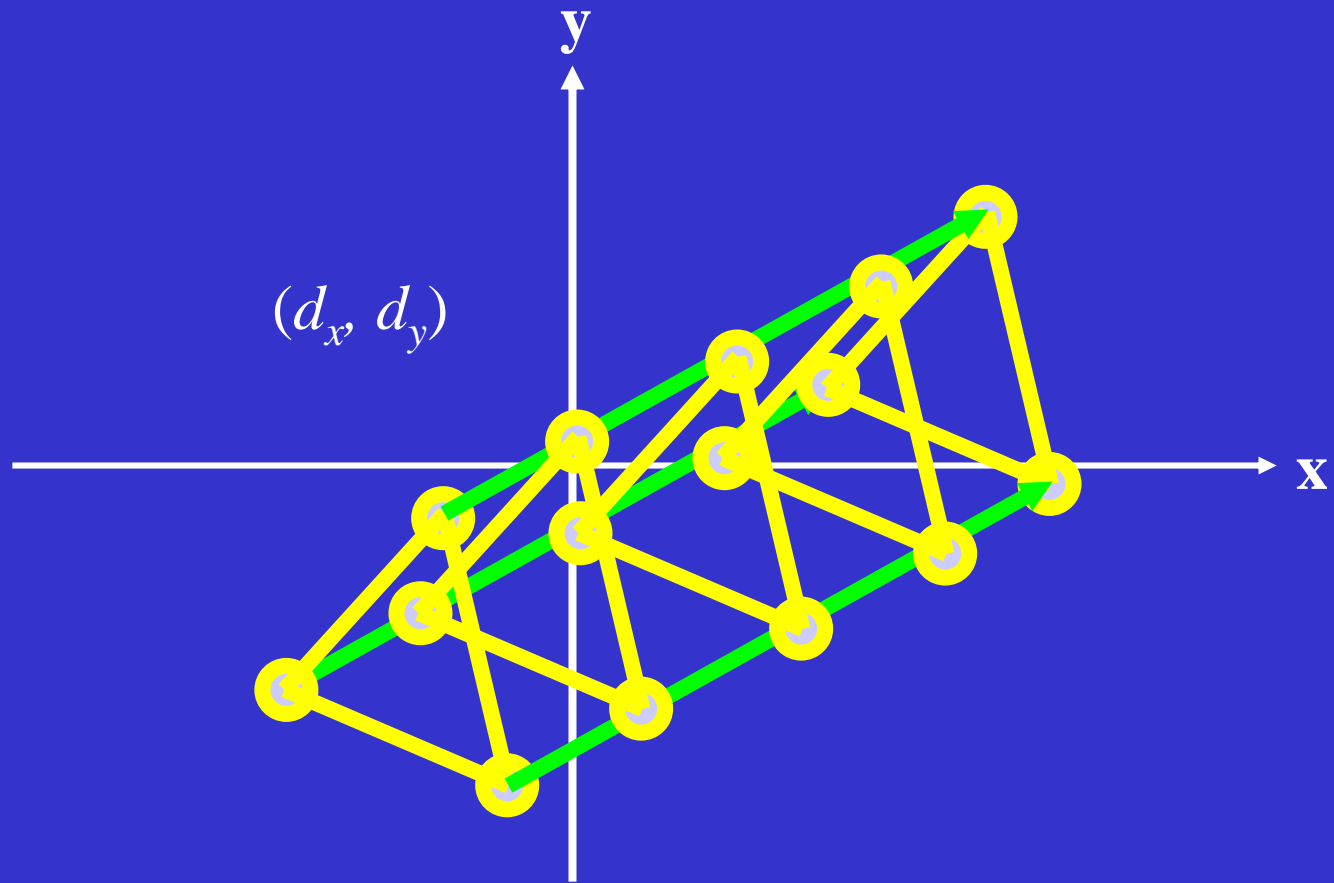
- Known: $P = (x, y)$
 $d = (d_x, d_y)$

- Want: $x' = x + d_x$
 $y' = y + d_y$

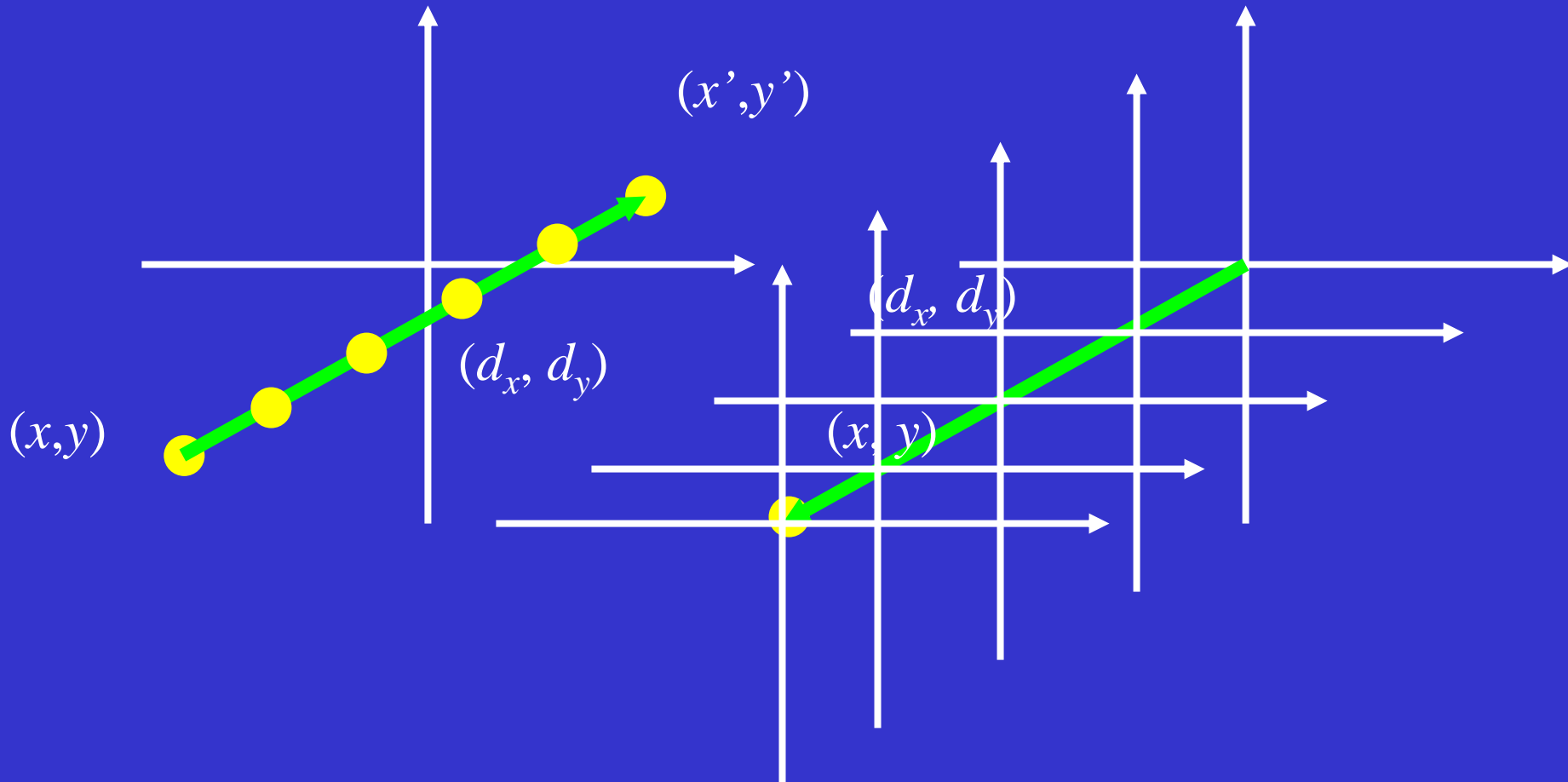
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

$$P' = P + d$$

Applying to Objects



See it From Different View



Translation

Remember

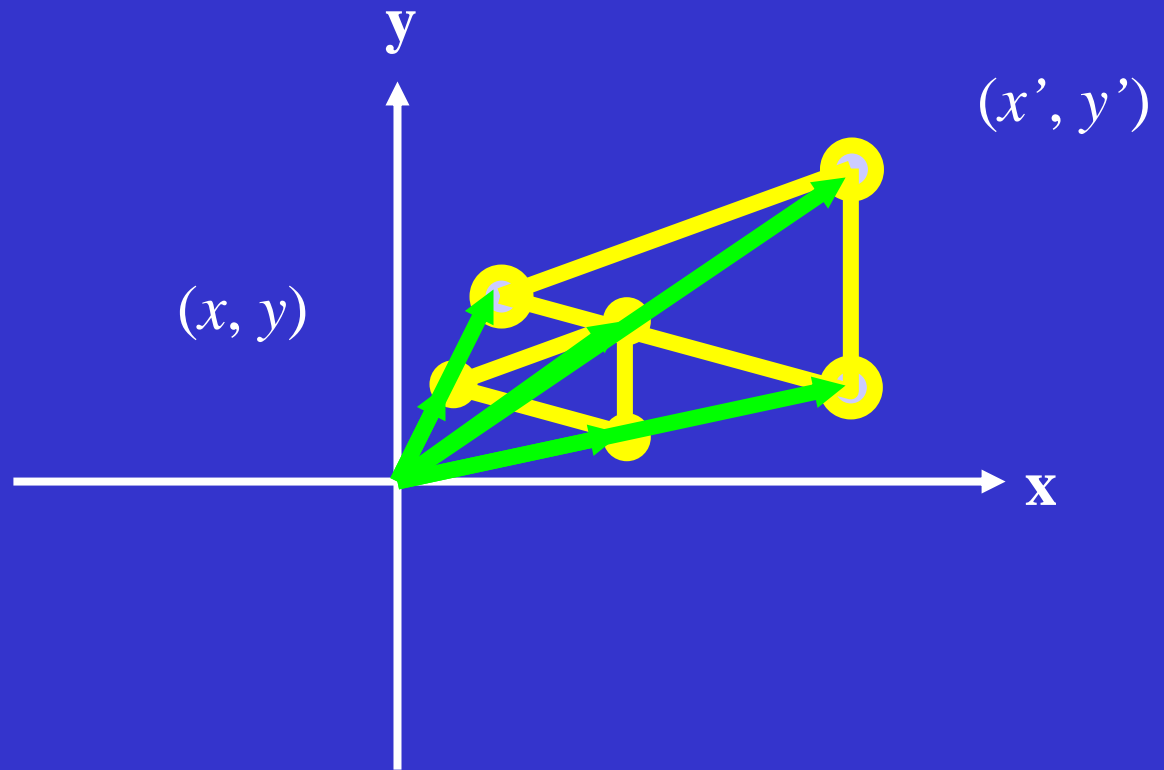
- Translation direction
- Translation distance
- 3 degrees of freedom (DoF)
- Translation is **rigid-body transformation** (a transformation that does not change the shape of objects)

Scaling

- **Scaling**

an operation that alters the size of an object about a **fixed point**

Scaling



Scaling

- Known: $P = (x, y)$
 $S = (s_x, s_y)$

- Want: $x' = s_x x$
 $y' = s_y y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$P' = S \cdot P$$

Scaling

- **Scale factors**

- if $s_x, s_y > 1$, the objects are stretched
- if $0 < s_x, s_y < 1$, the objects are shrunken
- if $s_x, s_y < 0$, the objects are flipped (reflection)

- **Uniform/differential scaling**

- if $s_x = s_y$, the scaling is **uniform**
- if $s_x \neq s_y$, the scaling is **differential** (non-uniform)

Scaling

Remember

- Scaling factors
- Fixed point
- Scaling is an affine **non-rigid-body transformation**

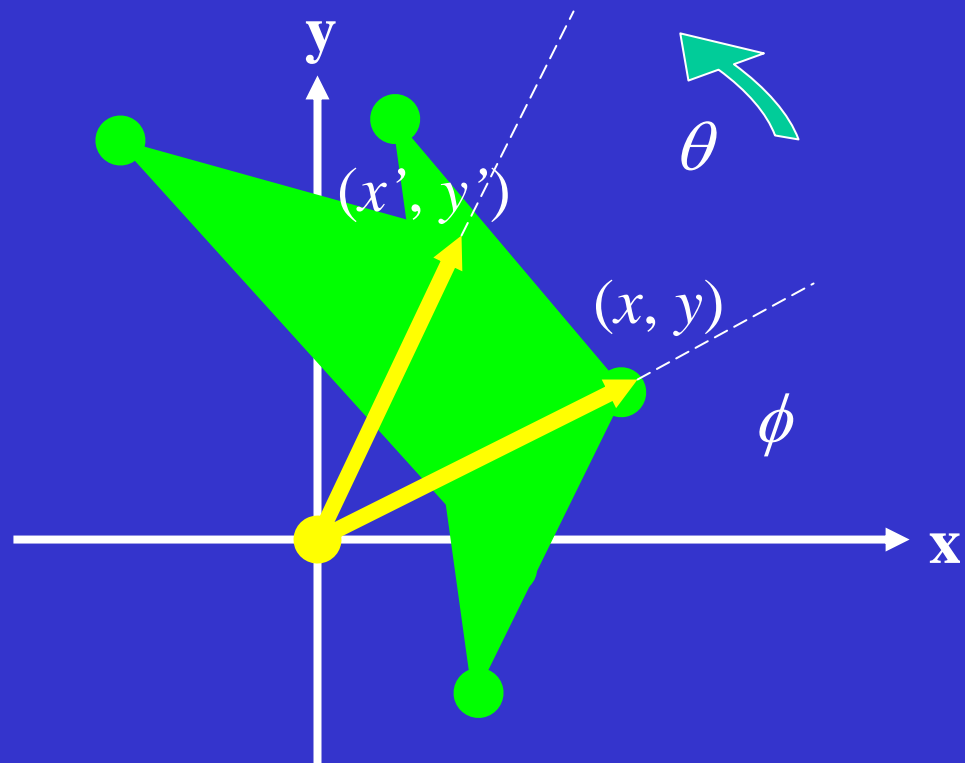
Rotation

- **Rotation**

an operation that repositions points along a given circular path (the **rotation direction**) from one coordinate location to another

Rotation requires a rotation center (*pivot point*)

Rotation



Rotation

- Known: $P = (x, y)$ $x = r \cos \phi$
 $R = (\theta)$ $y = r \sin \phi$

- Want: $x' = r \cos(\phi + \theta)$
 $y' = r \sin(\phi + \theta)$

Rotation

$$x' = r \cos \phi \cos \theta - r \sin \phi \sin \theta$$

$$y' = r \cos \phi \sin \theta + r \sin \phi \cos \theta$$

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

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

• Matrix form
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$P' = R \cdot P$$

Rotation

Remember

- Rotation direction
- Rotation angle
- Rotation center
- Rotation is **rigid-body transformation**

Recall the Homogeneous Coordinates

In the *homogeneous coordinate system*, both the vector and point can be represented as 4 dimension column matrix

$$\text{Point} \quad P = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \mathbf{1} \end{pmatrix}$$

$$\text{Vector} \quad v = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \mathbf{0} \end{pmatrix}$$

Recall the Homogeneous Coordinates

If we have two vectors or points a and b that are represented as homogeneous coordinates, we can use transformation matrix M to compute their representations directly

$$a = M b$$

$$b = M^{-1} a$$

$$M = \begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Transformation in Homogeneous Coordinates

- Translation

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \mathbf{p}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad \mathbf{d} = \begin{bmatrix} d_x \\ d_y \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

Transformation in Homogeneous Coordinates

- Translation

$$\mathbf{p}' = \mathbf{T}\mathbf{p}$$

$$\mathbf{T}(d_x, d_y) = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}$$

Translation matrix

$$\mathbf{T}^{-1}(d_x, d_y) = \begin{bmatrix} 1 & 0 & -d_x \\ 0 & 1 & -d_y \\ 0 & 0 & 1 \end{bmatrix}$$

Transformation in Homogeneous Coordinates

- Scaling

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \mathbf{p}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad \mathbf{S} = (s_x \ s_y)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Transformation in Homogeneous Coordinates

- Scaling

$$\mathbf{p}' = \mathbf{S}\mathbf{p}$$

$$\mathbf{S}(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling matrix

$$\mathbf{S}^{-1}(s_x, s_y) = \begin{bmatrix} 1/s_x & 0 & 0 \\ 0 & 1/s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformation in Homogeneous Coordinates

- Rotation

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \mathbf{p}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad \mathbf{r} = (\theta)$$

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

Transformation in Homogeneous Coordinates

- Rotation

$$\mathbf{p}' = \mathbf{R}\mathbf{p}$$

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

Rotation matrix

$$\mathbf{R}^{-1}(\theta) = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transformation in Homogeneous Coordinates

- More about rotation matrix

$$\mathbf{R}(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

vectors $\mathbf{v}_1 = [\cos q, -\sin q]$ and $\mathbf{v}_2 = [\cos q, \sin q]$:

- Each is a unit vector, e.g. $|\mathbf{v}_1| = 1, |\mathbf{v}_2| = 1$
- Each is perpendicular to the other, $\mathbf{v}_1 \cdot \mathbf{v}_2 = 0$

Transformation in Homogeneous Coordinates

- Rigid-body transformation

A transformation matrix of form

$$\begin{bmatrix} r_{11} & r_{12} & d_x \\ r_{21} & r_{22} & d_y \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

where the upper 2 x 2 sub-matrix is orthogonal, preserves angles and lengths. Such transformation is called **rigid-body transformation** (e.g. the body/object being transformed is not distorted in any way)

Summary

- Translation, Scaling, and Rotation all are *affine* transformation
- Translation and Rotation are *rigid-body* transformation
- Scaling is *non-rigid-body* transformation

Summary

- We can use 2x2 matrix or 3x3 homogeneous form

Translation: $T(d_x, d_y) = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}$

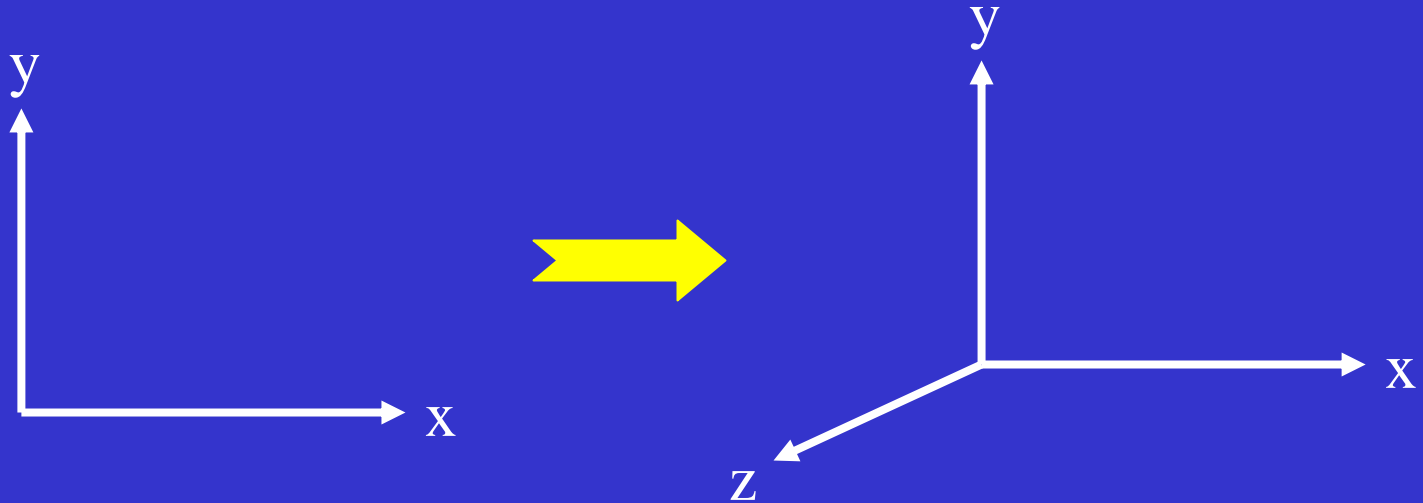
Scaling: $S(s_x, s_y) = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Rotation: $R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

3D Transformation

- Extend to 3D
 - Translate
 - Rotate
 - Scale

Extend to 3D

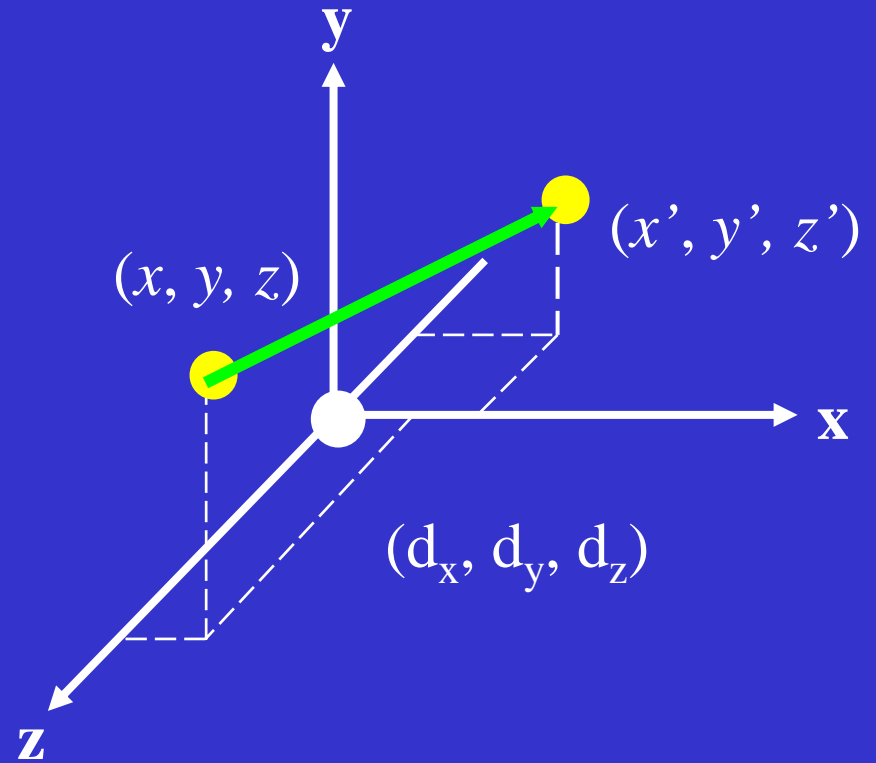
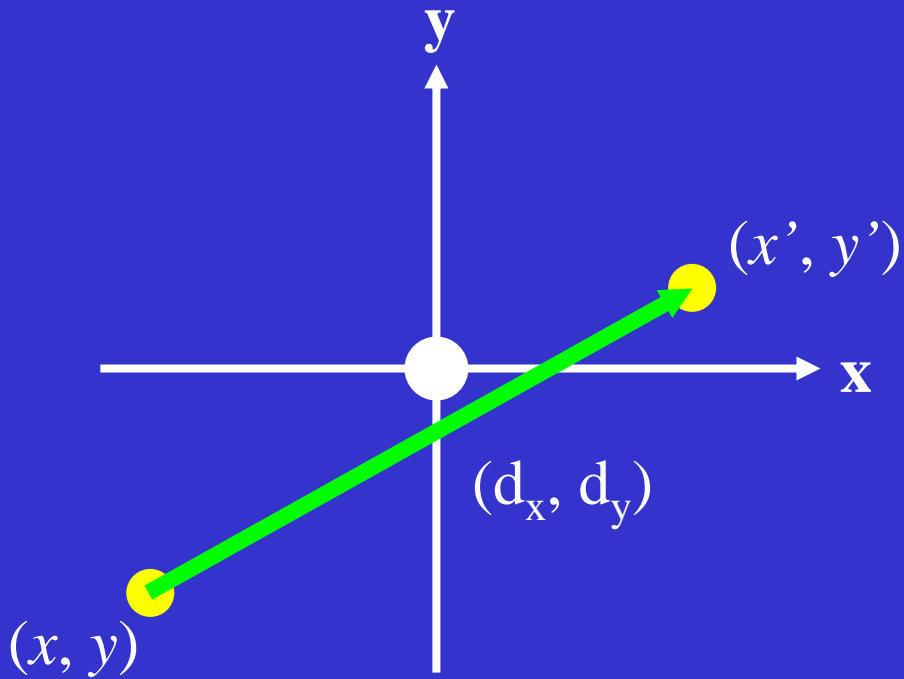


$$\begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$



$$\begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{pmatrix}$$

Translation



Translation Matrix

$$\mathbf{p}' = \mathbf{T}\mathbf{p}$$

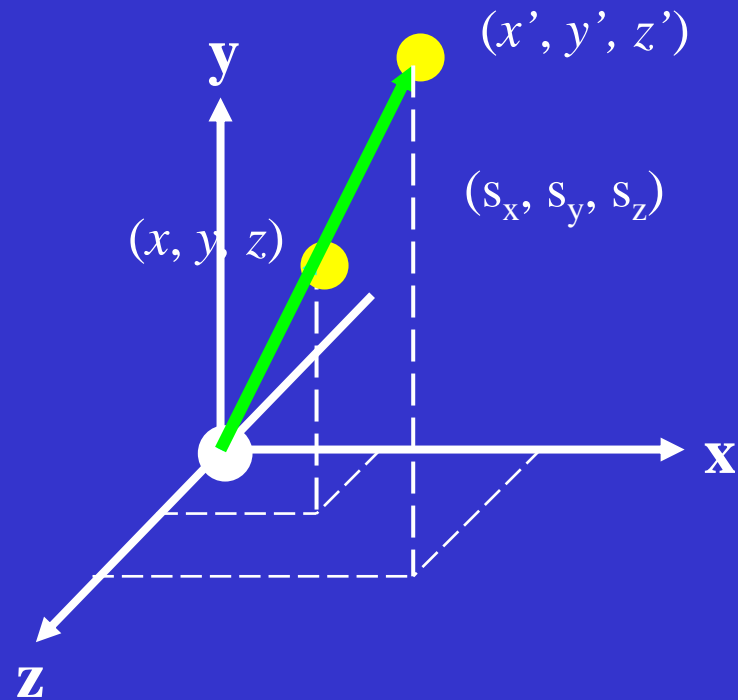
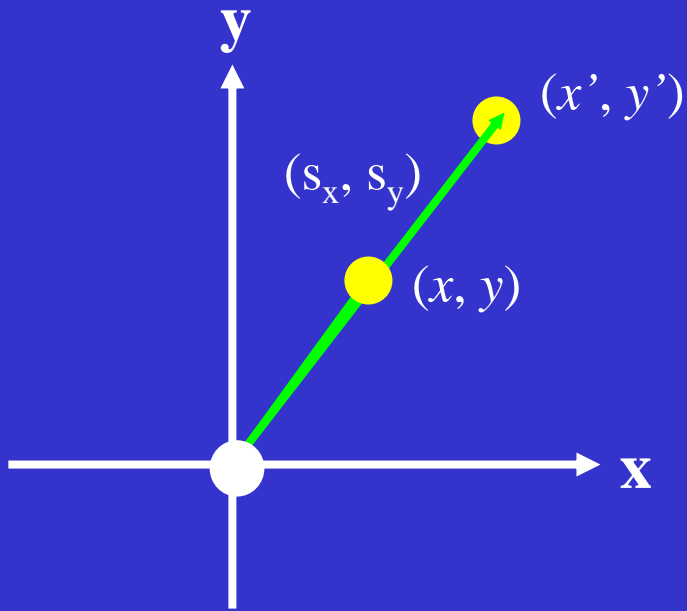
2D:

$$T(d_x, d_y) = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix}$$

3D:

$$T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling



Scaling Matrix

$$\mathbf{p}' = \mathbf{S}\mathbf{p}$$

2D:

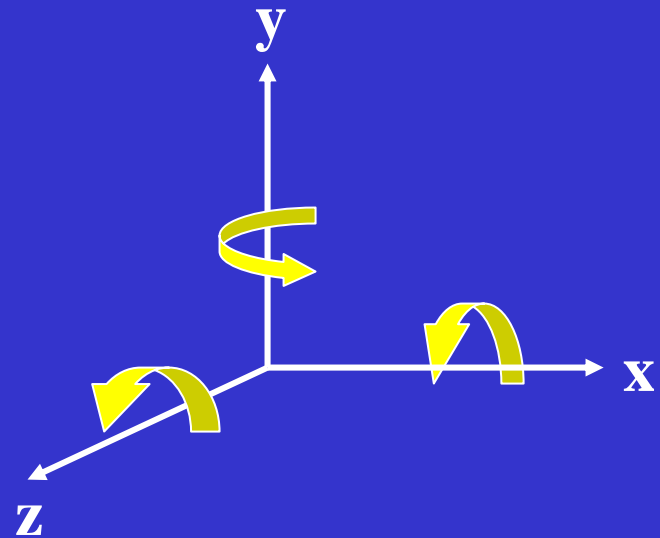
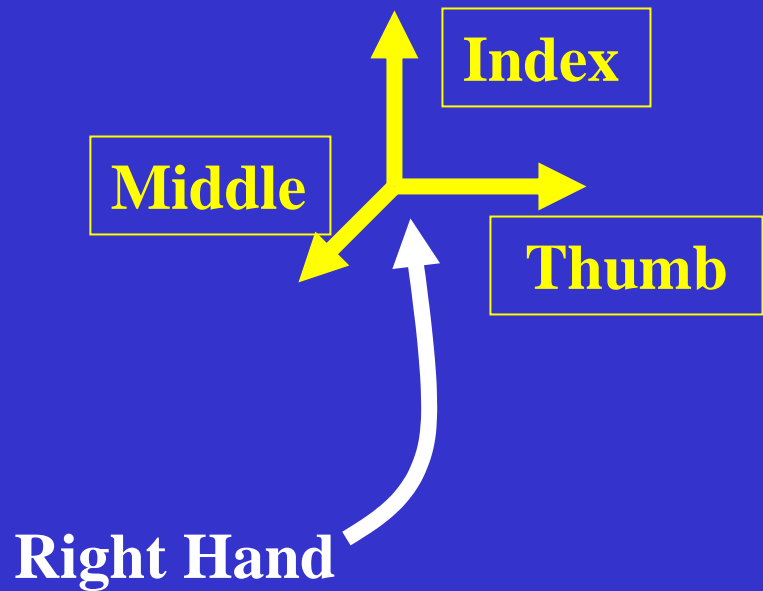
$$\mathbf{S}(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3D:

$$\mathbf{S}(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Right-handed Coordinate

We use **Right-handed** coordinate to define 3D rotation

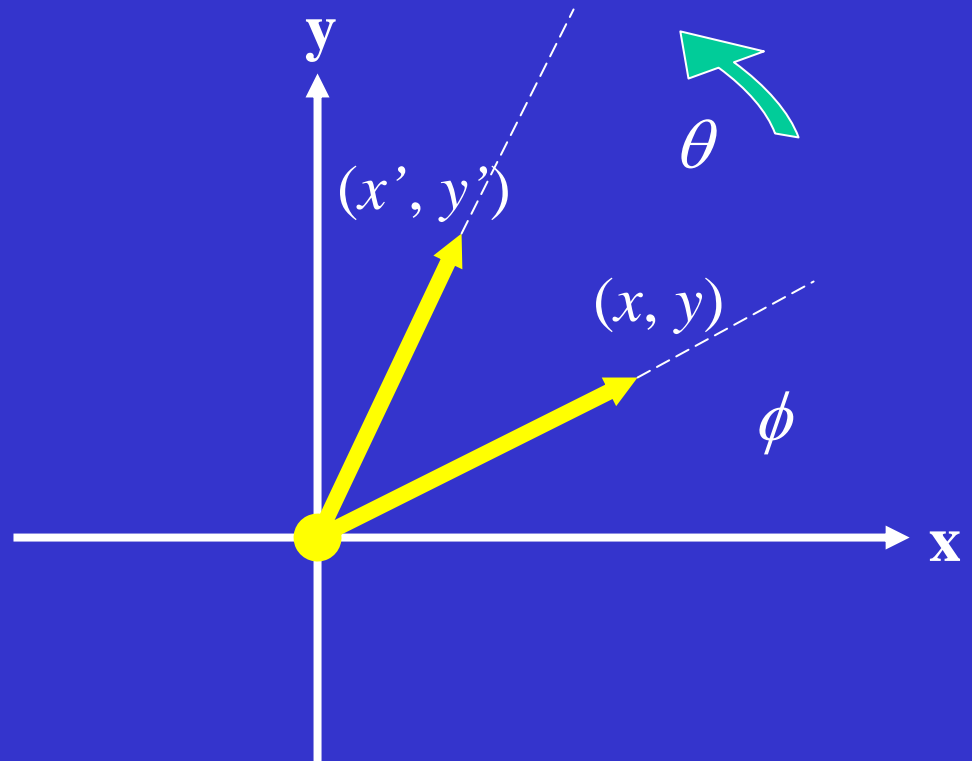


Rotation

- To specify a 2D rotation, we need an rotation angle and a *pivot point*

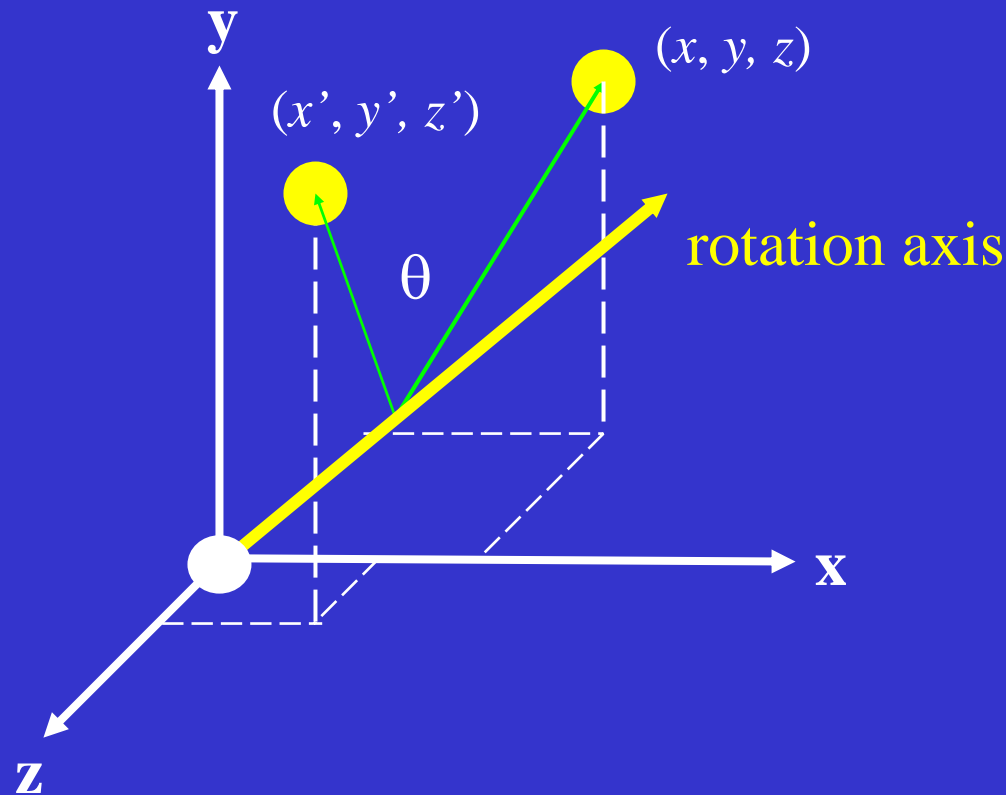
How about for 3D rotation?

- 2D is a special case of 3D ($z = 0$). What does mean in the rotation case?



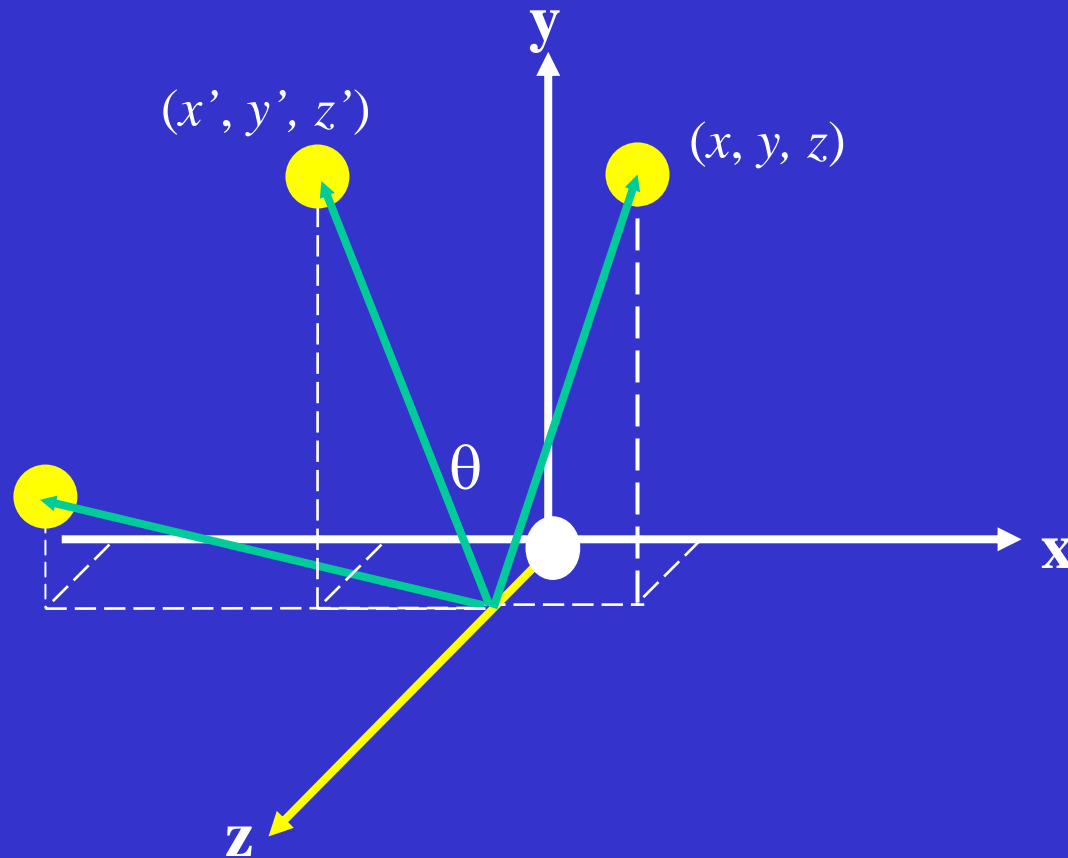
3D Rotation

- To specify a 3D rotation, we requires an rotation angle and a *vector* (rotation axis)



3D Rotation About Z axis

- 2D rotation is just a 3D rotation about the z axis



3D Rotation About Z axis

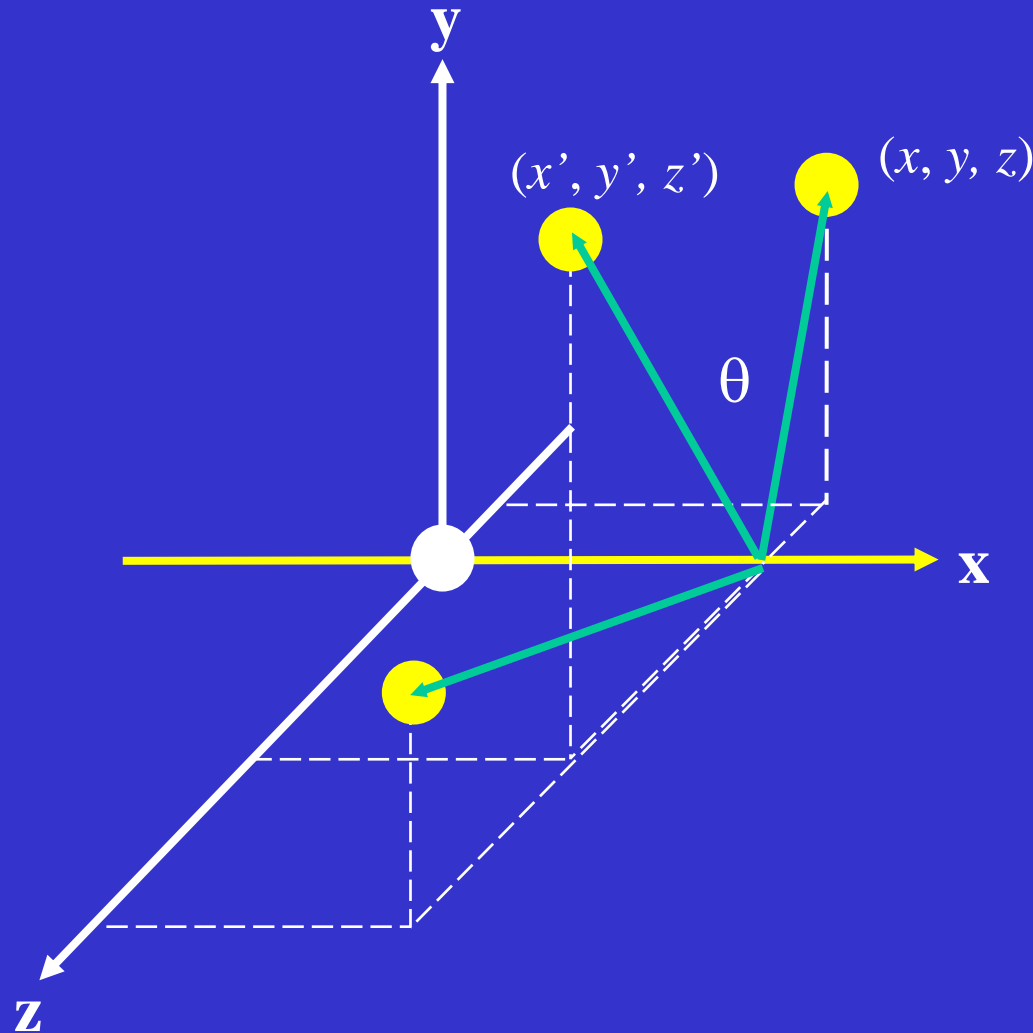
2D:

$$R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3D about z axis:

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Rotation About X axis

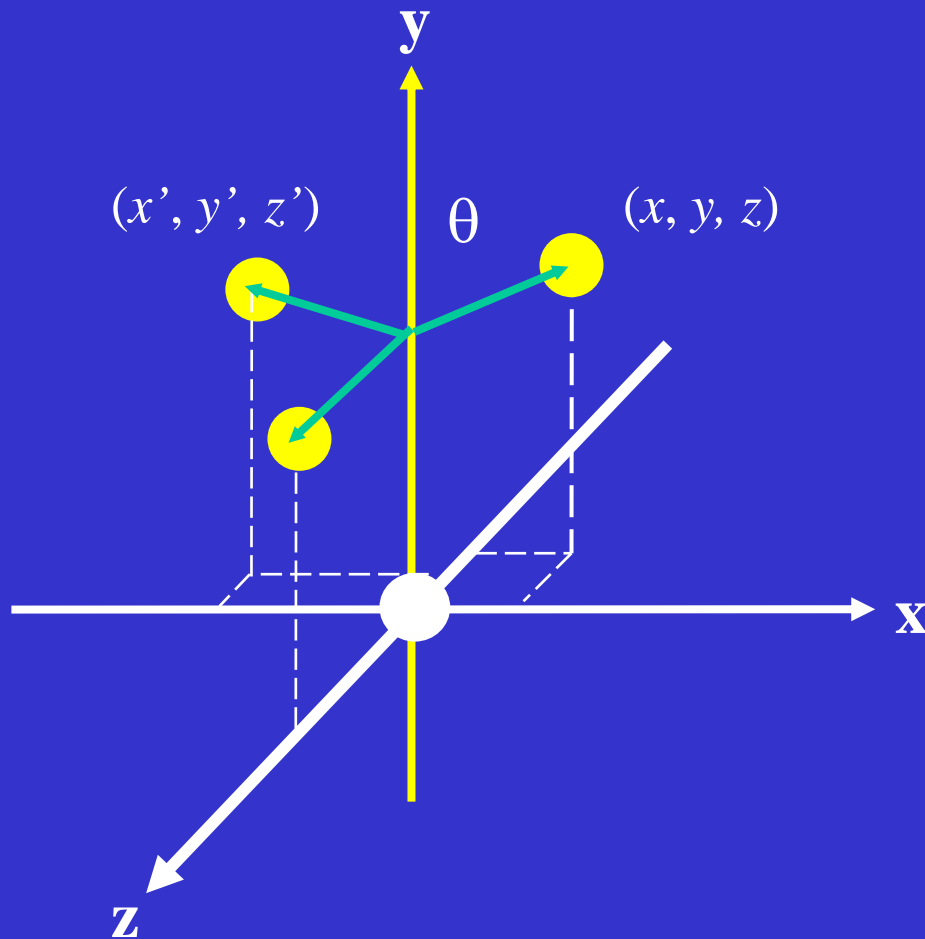


3D Rotation About X axis

Rotation matrix $R_x(\theta)$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Rotation About Y axis

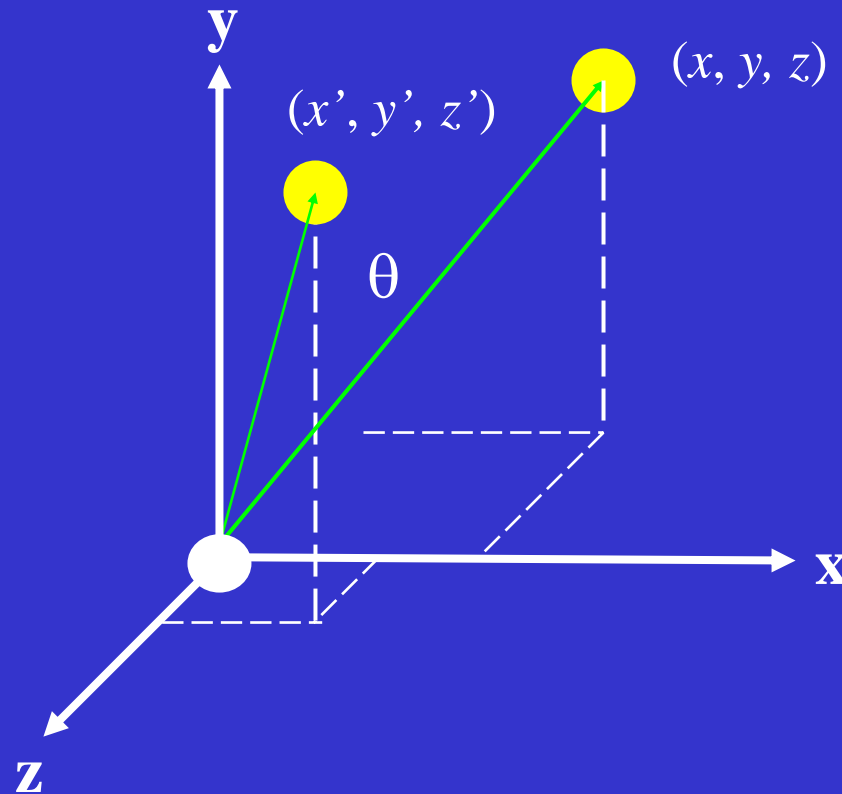


3D Rotation About Y axis

Rotation matrix $R_y(\theta)$

$$R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Rotation About Origin



3D Rotation About Origin

- An arbitrary rotation about the origin can be composed of three successive rotations about three axes

$$\mathbf{R}(\theta) = \mathbf{R}_z(\theta_z) \mathbf{R}_y(\theta_y) \mathbf{R}_x(\theta_x)$$

- The order of the composed rotations is not unique

Ex:

$$\mathbf{R}_x(\theta_x) \rightarrow \mathbf{R}_y(\theta_y) \rightarrow \mathbf{R}_z(\theta_z)$$

- The three angles q_x, q_y, q_z are often called **Euler Angles** or called **Yaw** (q_y), **Pitch** (q_x), and **Roll** (q_z)

Summary

- Three elementary 3D affine transformations

Translation: $T(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Scaling: $S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Summary

Rotation $R_x(\theta)$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

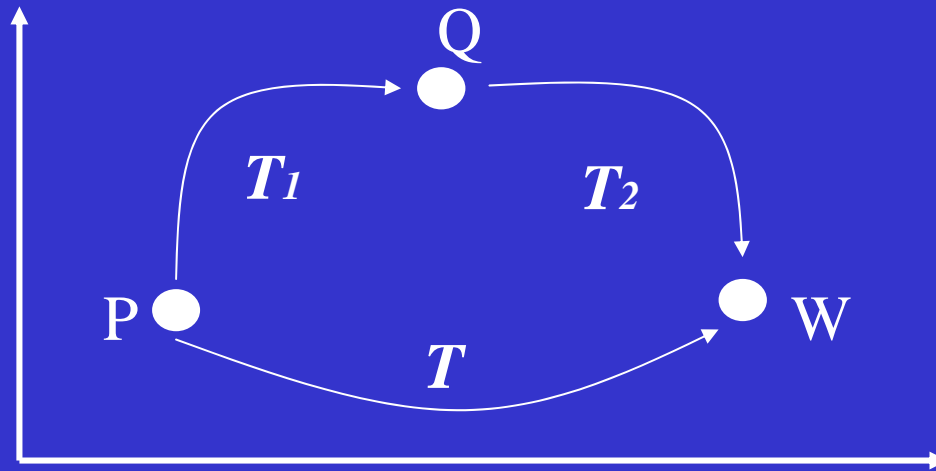
Rotation $R_y(\theta)$

$$R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation $R_z(\theta)$

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Composition of Transformations



$T_1: P \rightarrow Q$

$T_2: Q \rightarrow W$

$T: P \rightarrow W$

What is T ?

$T = T_2T_1$ or $T = T_1T_2$?

Composition of Transformations

- **Composition:** the process of applying several transformations in succession to form one overall transformation
- Any composition of affine transformations is still affine
- When homogeneous coordinates are used, affine transformations are composed by simple matrix multiplication

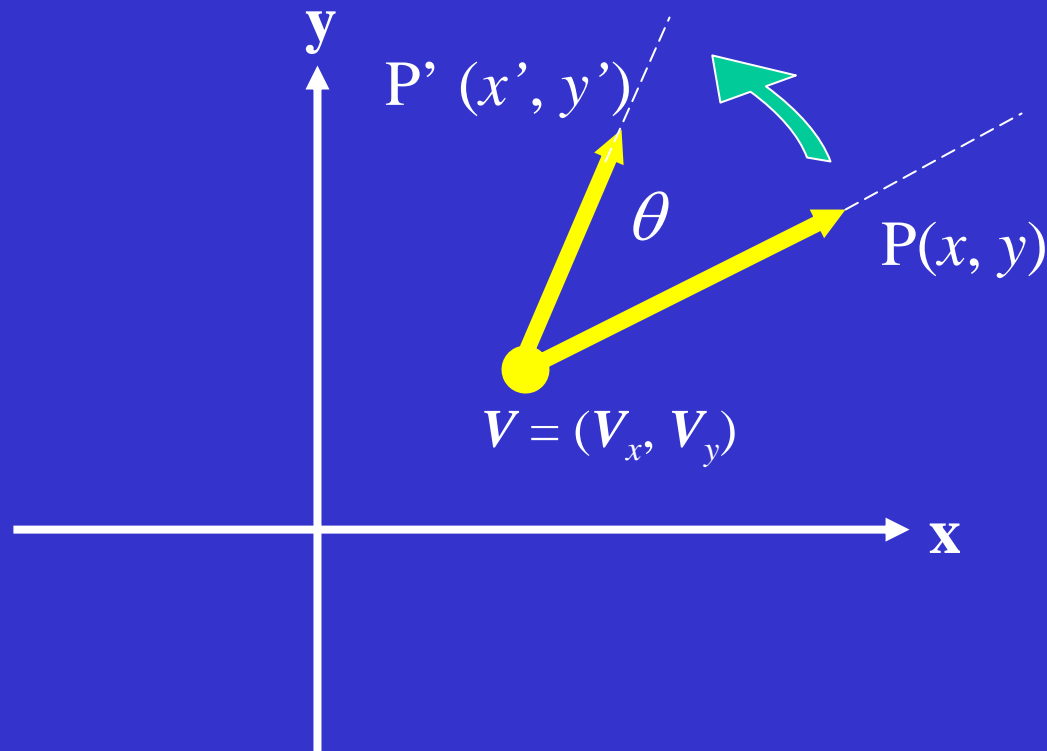
Ex:
$$T = T_4 T_3 T_2 T_1$$

- Note: the matrices appear in *reverse* order to that in which the transformations are applied

Composition of Transformations

Let's examine a simple example:

- 2D Rotation about an arbitrary point



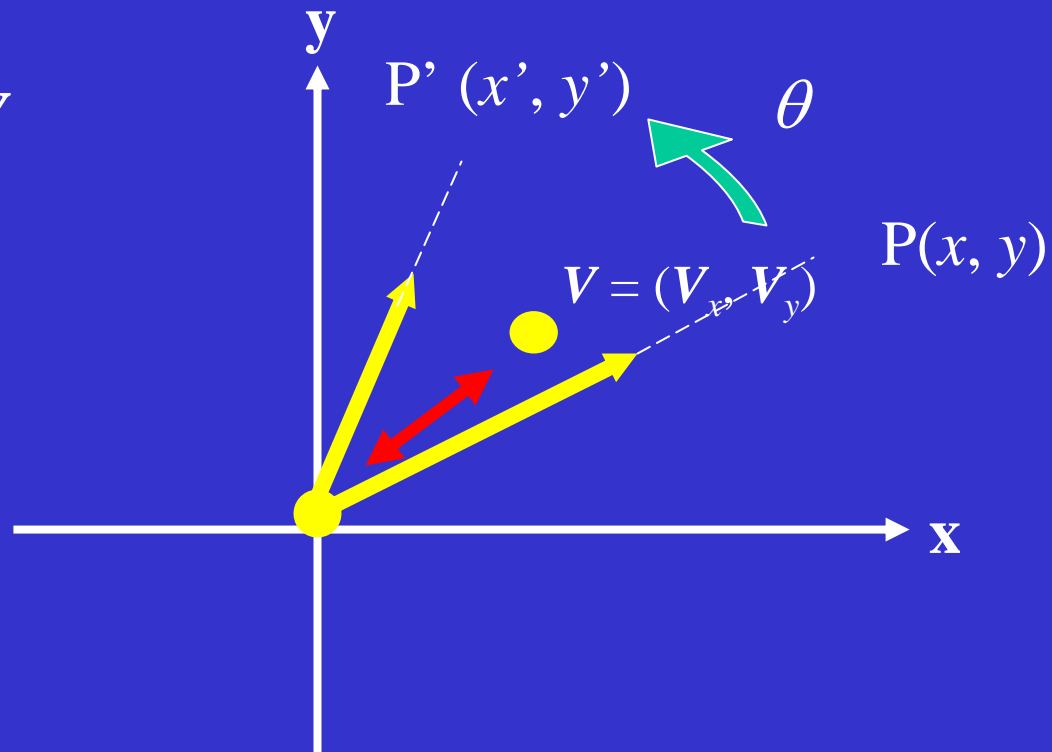
Rotation About an Arbitrary Point

Since we have known rotation about origin, we can use that and construct successive transformations to compose the rotation about an arbitrary pivot point

Three Steps...

Rotation About an Arbitrary Point

1. Translate point P through vector $-V$
2. Rotate about the origin through angle θ
3. Translate P back through vector V



Rotation About an Arbitrary Point

Find rotation matrix

$$\mathbf{M} = \mathbf{T}(V) \mathbf{R}(\theta) \mathbf{T}(-V)$$

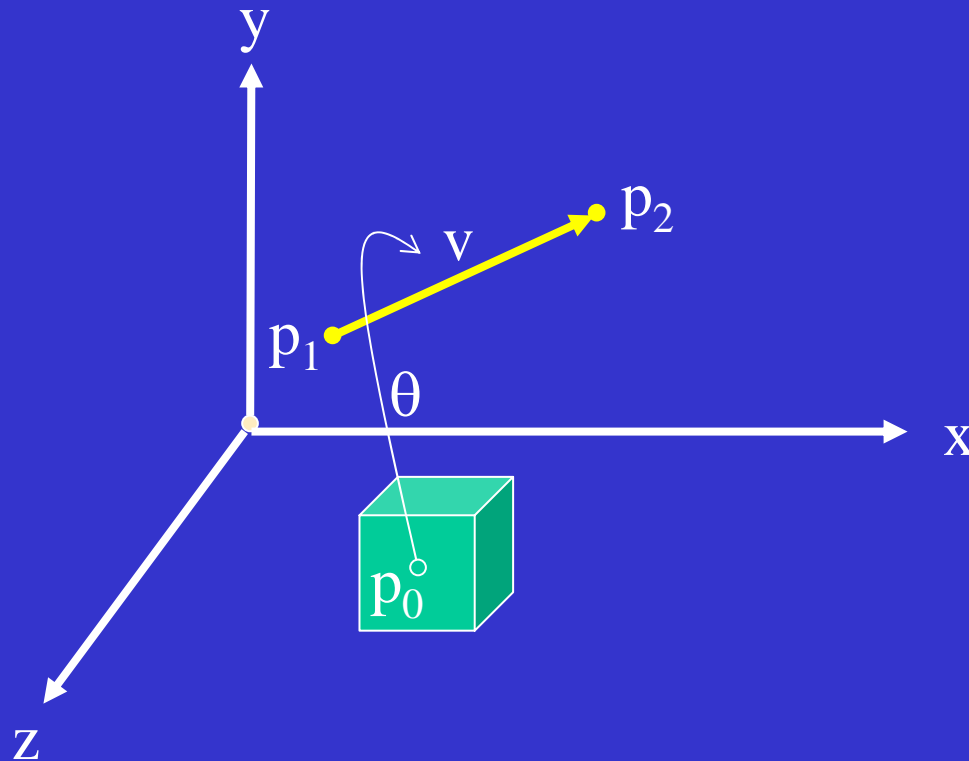
$$\mathbf{M} = \begin{bmatrix} 1 & 0 & V_x \\ 0 & 1 & V_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -V_x \\ 0 & 1 & -V_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos \theta & -\sin \theta & V_x \cos \theta - V_y \sin \theta \\ \sin \theta & \cos \theta & V_x \sin \theta + V_y \cos \theta \\ 0 & 0 & 1 \end{bmatrix}$$

Composition of Transformations

More complicated example:

- Rotation about an arbitrary axis



Rotation About an Arbitrary Axis

Idea

- We can always rotate about the origin, if we can translate everything first (It's a very useful technique)

$$M = T(p_0) \boxed{?} T(-p_0)$$

- We can compose rotations about the axes to create a general rotation

Rotation About an Arbitrary Axis

Find 

- We can compose general rotation from the three rotations about individual axes, but we do not know what angles to use for the individual rotations!
- Strategy
 - Use two rotations to align the axis of rotation \mathbf{v} , with the \mathbf{Z} axis
 - Rotate by θ about the \mathbf{Z} axis
 - Undo the two rotations that did the aligning

Rotation About an Arbitrary Axis

We can find

$$\boxed{?} = R_x(-\theta_x) R_y(-\theta_y) R_z(\theta) R_y(\theta_y) R_x(\theta_x)$$

Finally, the rotation matrix is

$$M = T(\mathbf{p}_0) R_x(-\theta_x) R_y(-\theta_y) R_z(\theta) R_y(\theta_y) R_x(\theta_x) T(-\mathbf{p}_0)$$

We can always use this form to find a rotation around an arbitrary axis

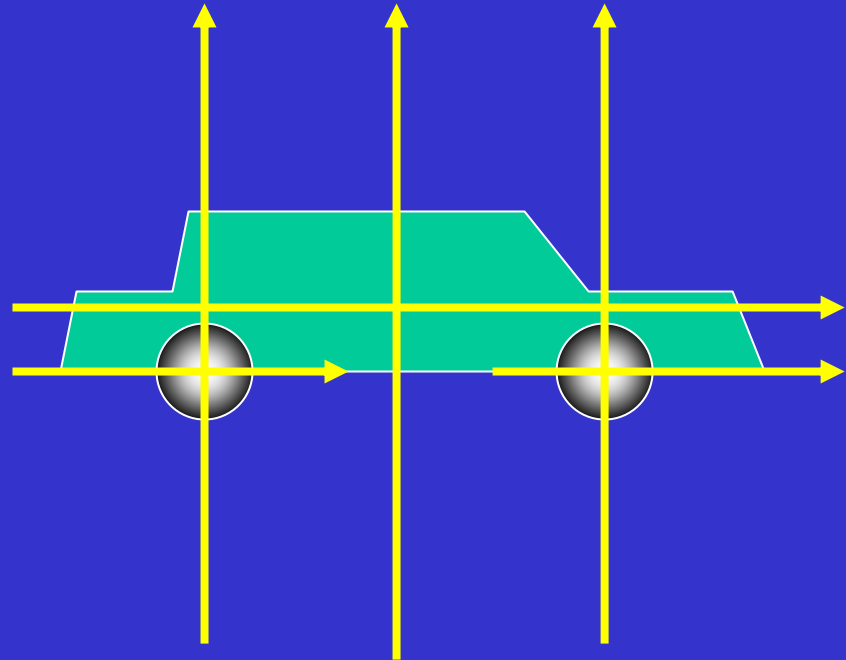
Summary

- We can compose an overall transformation by applying several transformations in succession
- Any composition of affine transformations is still affine
- When homogeneous coordinates are used, affine transformations are composed by simple matrix multiplication
- We can always rotate about the origin, if we can translate everything first

Very Useful Technique

1. Draw the car body
2. Translate to right front and draw wheel
3. Return back to car body
4. Translate to left front and draw wheel
5. Return back to car body

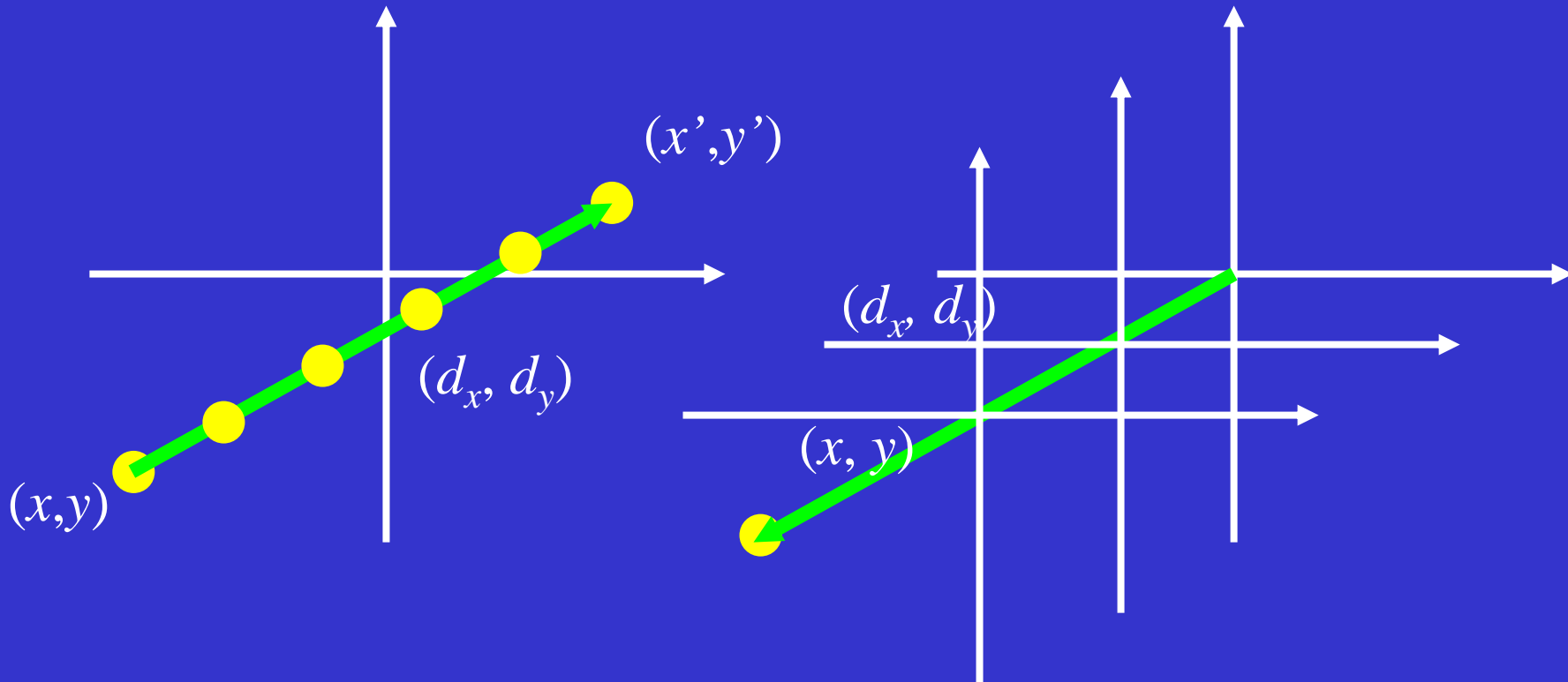
.....



- **Always remember where you are!**
- **Matrix multiplication is not commutative, the transformation order is very important**

Coordinate Transformations

- We can think about, alternatively, a transformation is as a change of coordinate systems



Coordinate Transformations

- **Transforming points**

To apply a sequence of transformations T_1, T_2, T_3 (in that order) to a point, form matrix

$$T = T_3 T_2 T_1$$

Then, P is transformed to TP

- **Transforming the coordinate system**

To apply a sequence of transformations T_1, T_2, T_3 (in that order) to the coordinate system, form matrix

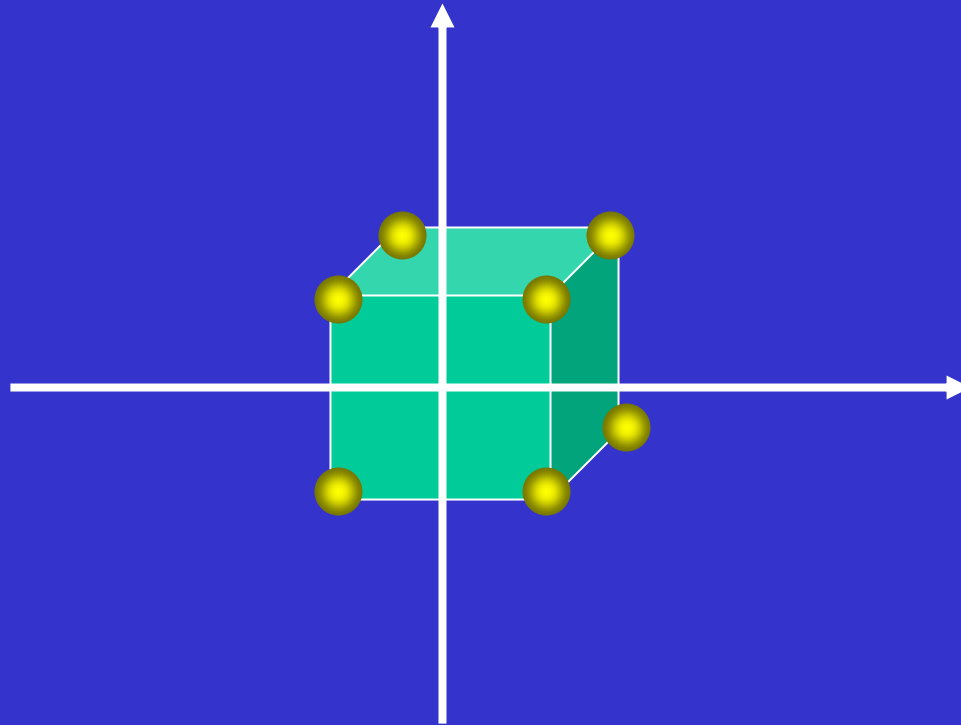
$$T = T_1 T_2 T_3$$

Then, a point P is expressed in the transformed system has coordinates MP in the original system.

Graphics Transformations

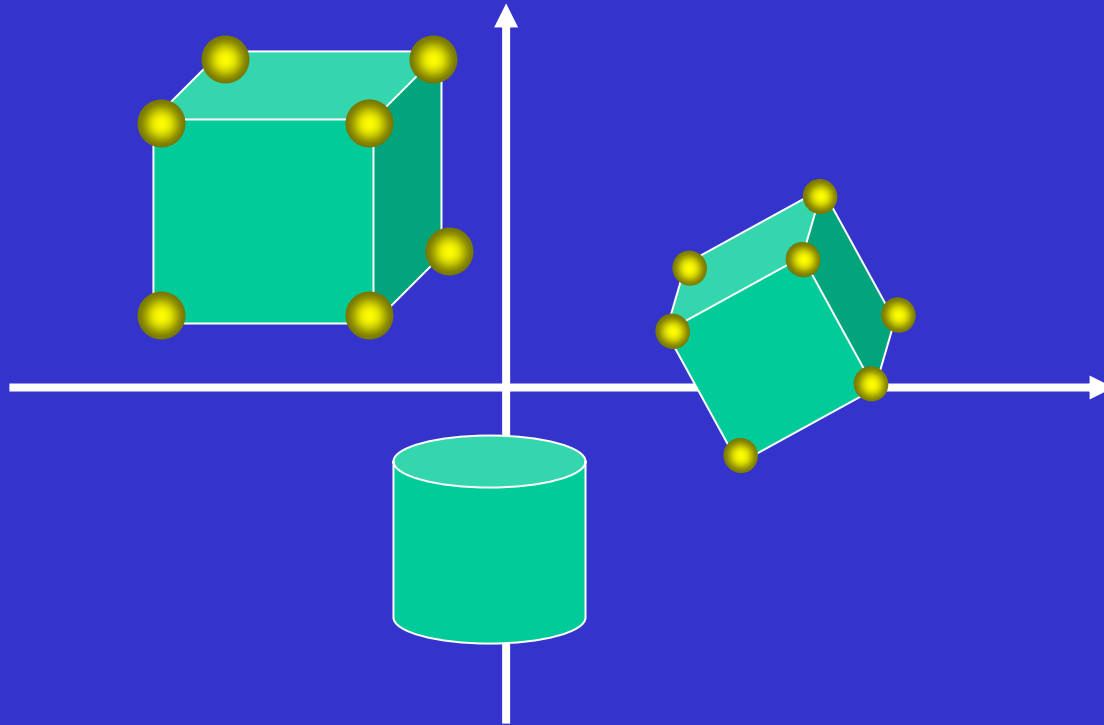
- Objects are usually defined in their own local coordinate systems
- We wish to express these objects' coordinates in a single, global system
 - Object Coordinates
 - World Coordinates
 - Screen Coordinates

Object Coordinates



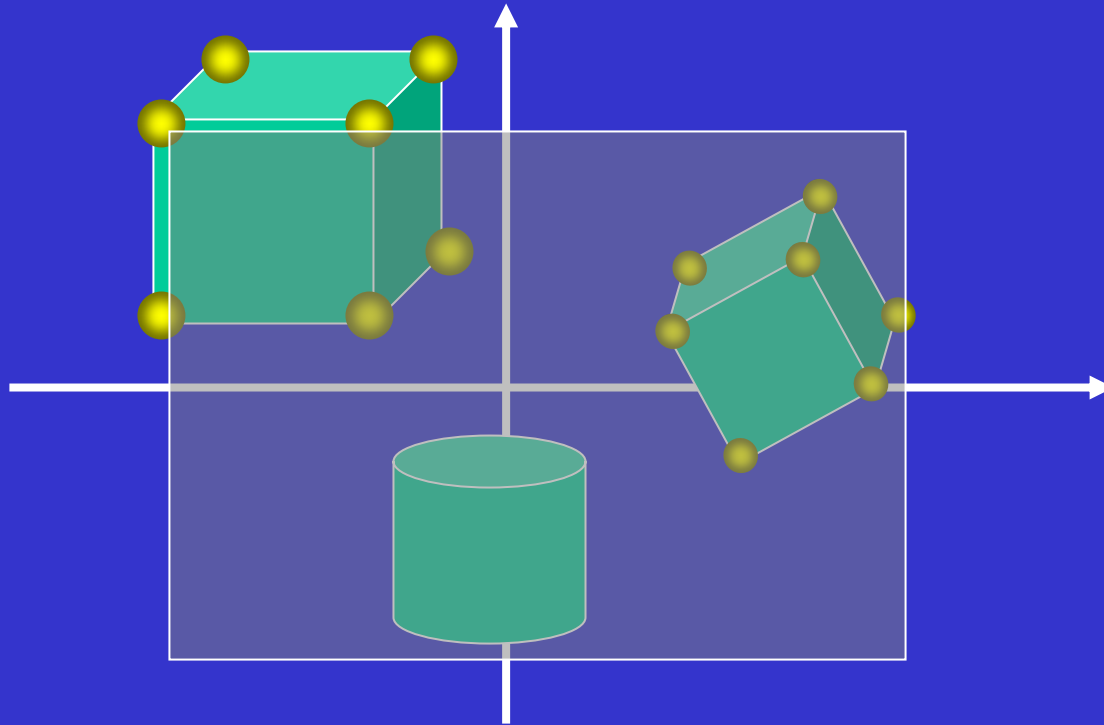
Objects are usually defined in their own local coordinate systems

World Coordinates



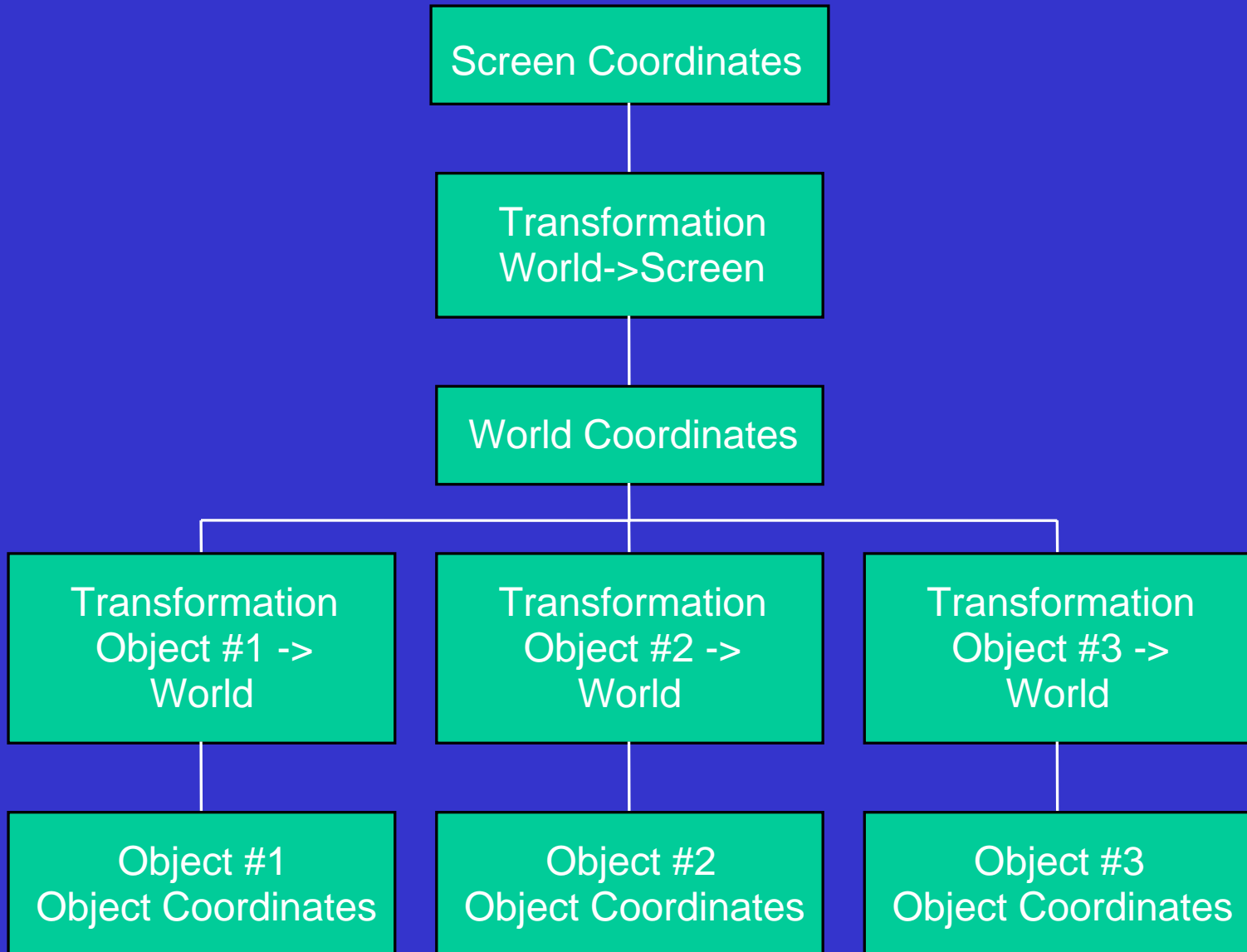
Represent these objects' coordinates in a single, global coordinates system

Screen Coordinates

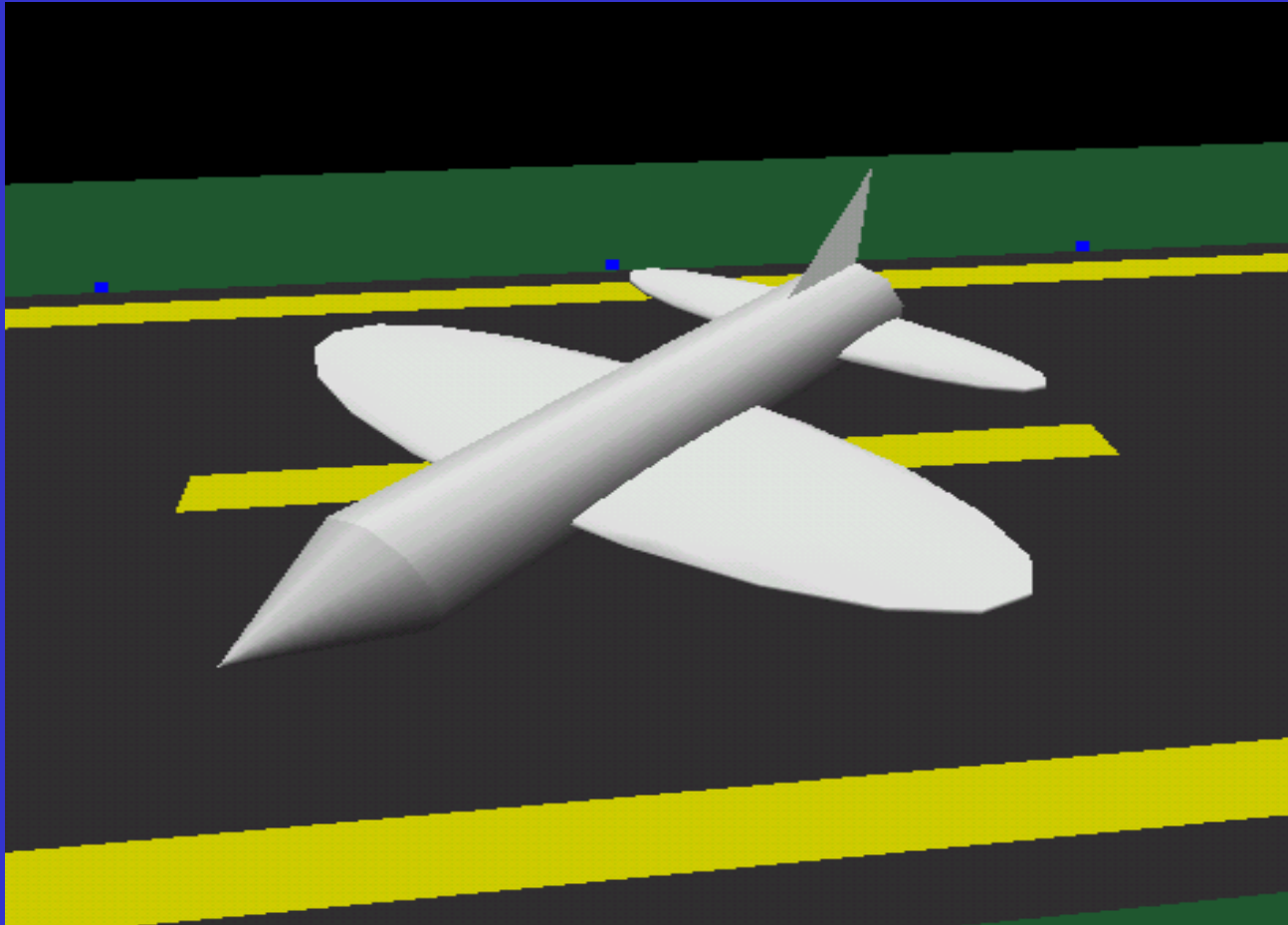


Finally, we want to project these objects onto the screen

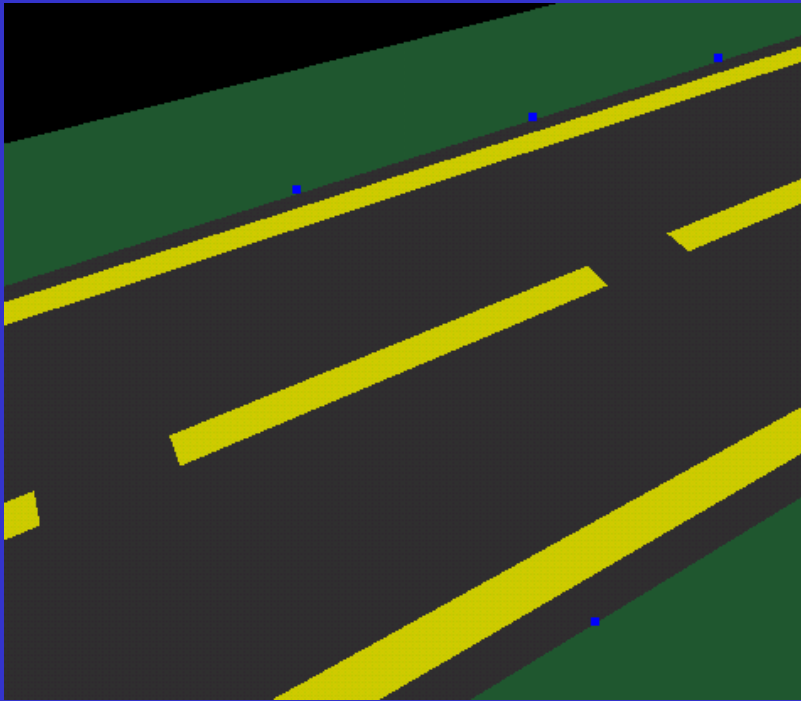
Coordinate Hierarchy



Examples



Object Hierarchy



Object Hierarchy

