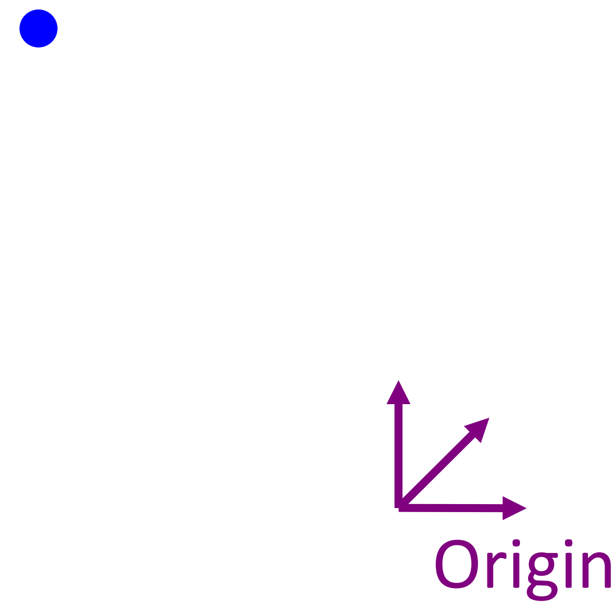
Physically Based Rendering (600.657)

Geometry and Transformations

3D Point

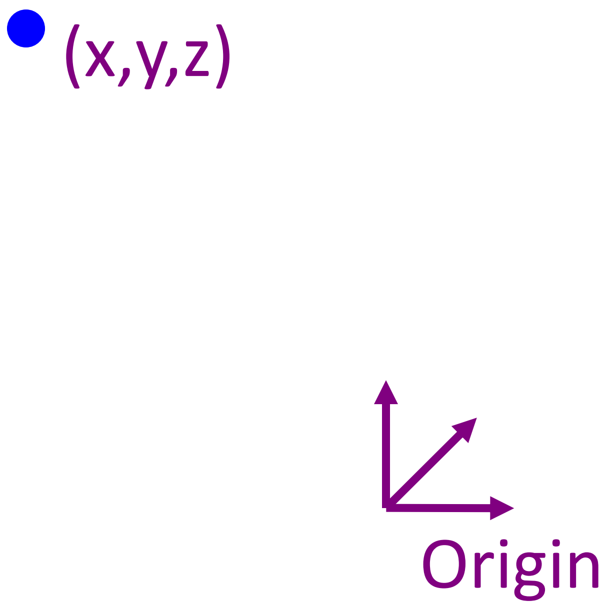
- Specifies a location



3D Point

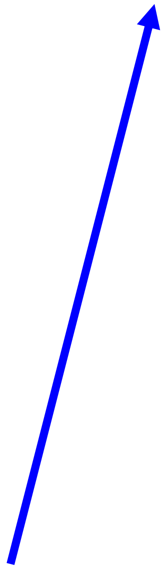
- Specifies a location
  – Represented by three coordinates
  – Infinitely small

```
class Point3D
{
public:
    Coordinate x;
    Coordinate y;
    Coordinate z;
};
```



3D Vector

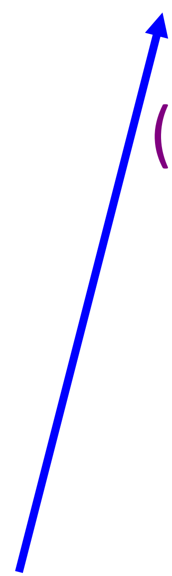
- Specifies a direction and a magnitude



3D Vector

- Specifies a direction and a magnitude
  - Represented by three coordinates
  - Magnitude ||V|| = sqrt(dx dx + dy dy + dz dz)
  - Has no location

```
class Vector3D
{
public:
    Coordinate dx;
    Coordinate dy;
    Coordinate dz;
};
```

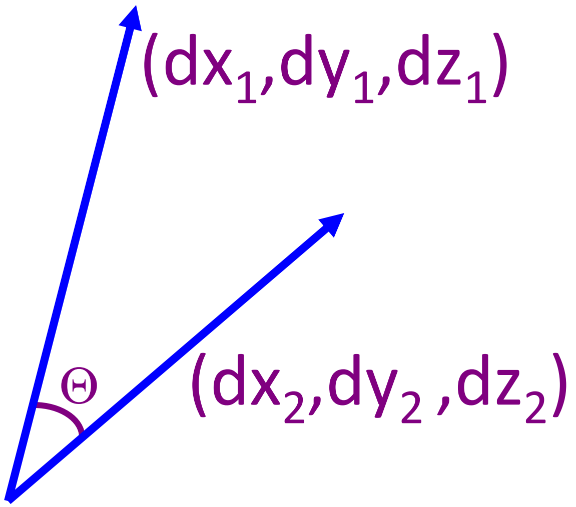


(dx,dy,dz)

3D Vector

- Specifies a direction and a magnitude
  - Represented by three coordinates
  - Magnitude $||V|| = \mathrm{sqrt}(dx\ dx + dy\ dy + dz\ dz)$
  - Has no location

```
class Vector3D
{
public:
    Coordinate dx;
    Coordinate dy;
    Coordinate dz;
};
```

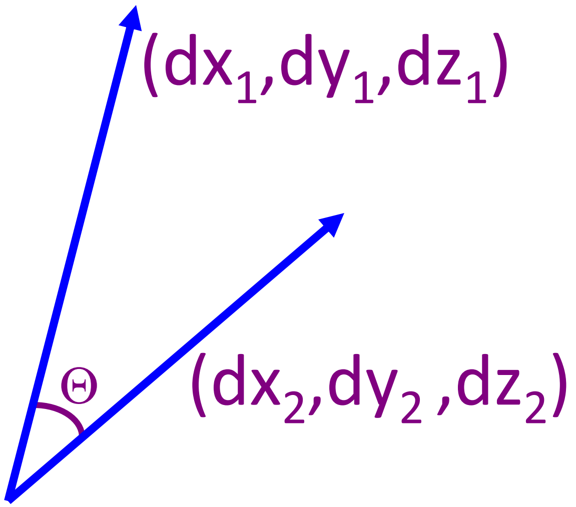


- Dot product of two 3D vectors
  - $V_1 \cdot V_2 = dx_1 dx_2 + dy_1 dy_2 + dz_1 dz_2$
  - $V_1 \cdot V_2 = ||V_1||\ ||V_2||\cos(\Theta)$

3D Vector

- Specifies a direction and a magnitude
  - Represented by three coordinates
  - Magnitude $||V|| = \text{sqrt}(dx\ dx + dy\ dy + dz\ dz)$
  - Has no location

```
class Vector3D
{
public:
     Coordinate dx;
     Coordinate dy;
     Coordinate dz;
};
```



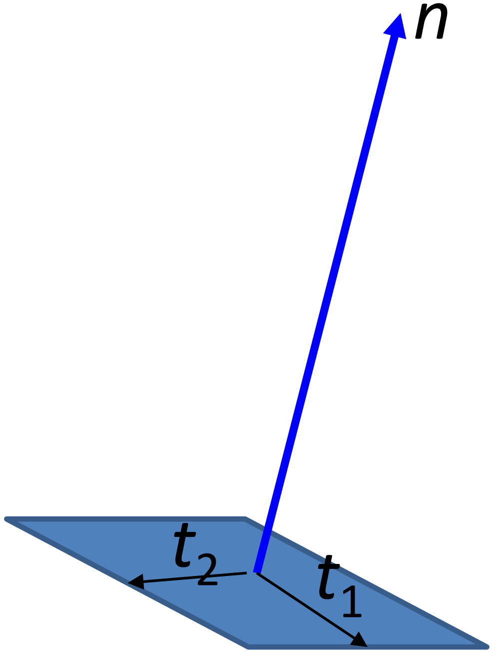
- Cross product of two 3D vectors
  - $V_1 \times V_2$ = Vector normal to plane V_1, V_2
  - $|| V_1 \times V_2 || = ||V_1|| \; ||V_2|| \sin(\Theta)$

Cross Product: Review

- Let $U = V \times W$:
  - $U_x = V_y W_z - V_z W_y$
  - $U_y = V_z W_x - V_x W_z$
  - $U_z = V_x W_y - V_y W_x$
- $V \times W = - W \times V$ (remember "right-hand" rule)
- We can do similar derivations to show:
  - $V_1 \times V_2 = \|V_1\| \, \|V_2\| \sin(\Theta) n$, where n is unit vector normal to V_1 and V_2
  - $\| V_1 \times V_1 \| = 0$

3D Normal

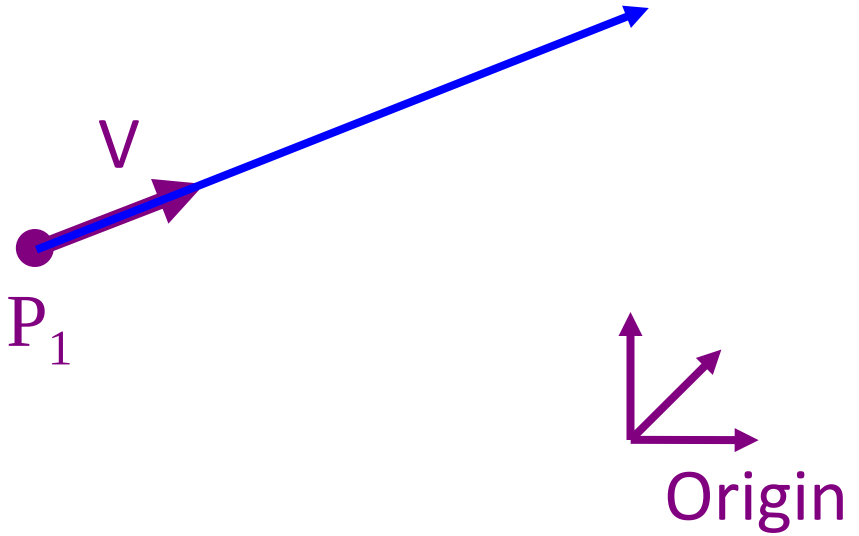
- Specifies a differential patch by the area and perpendicular direction.



3D Ray

- Line segment with one endpoint at infinity
  - Parametric representation:
    - $P = P_1 + t\,V, \quad (0 <= t < \infty)$

```
class Ray3D
{
public:
    Point3D P1;
    Vector3D V;
};
```

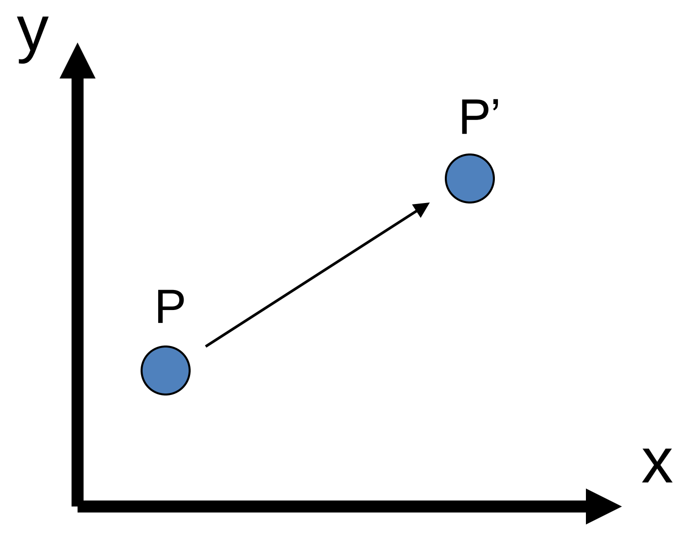


Simple 2D Transformations

Translation

$$p' = T + p$$

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

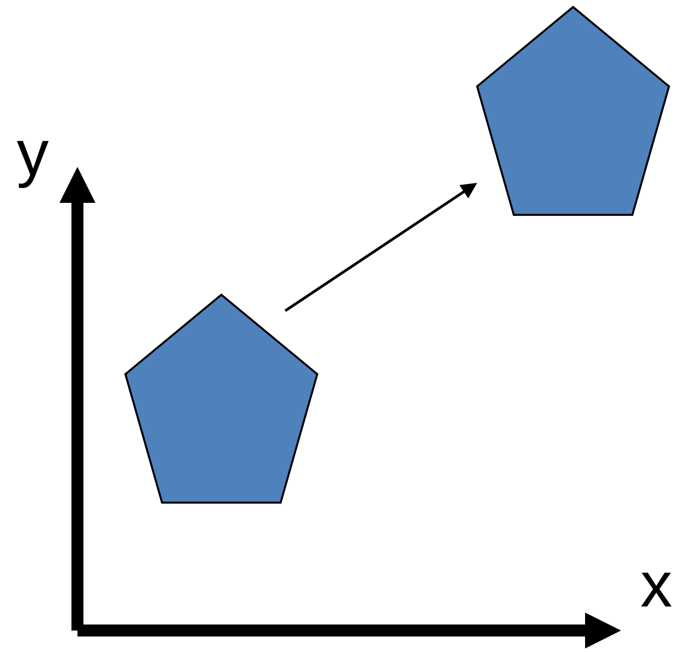


Simple 2D Transformations

Translation

$$p' = T + p$$

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

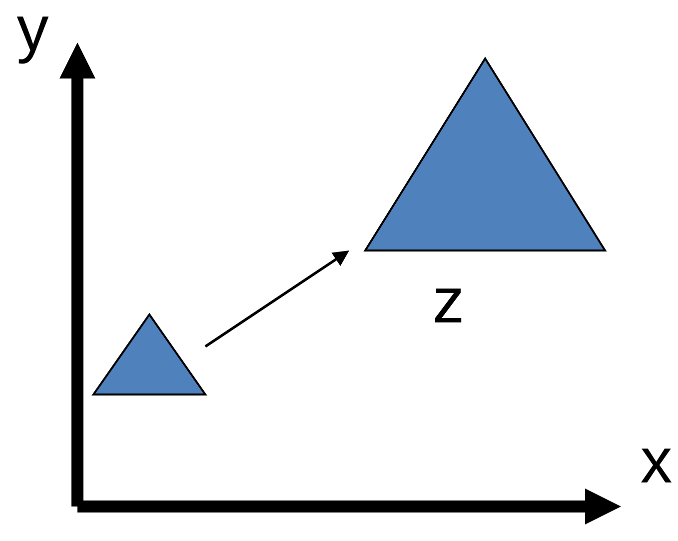


Simple 2D Transformations

Scale

$$p' = S \bullet p$$

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

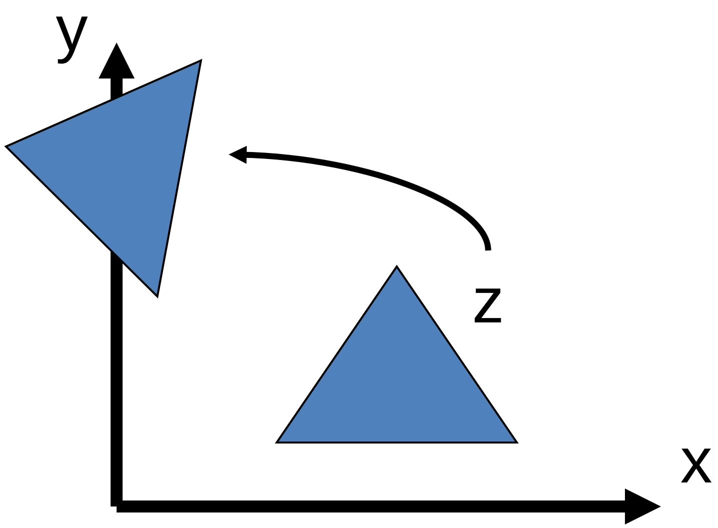


Simple 2D Transformation

Rotation

$$p' = R \bullet p$$

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



Matrix Representation

- Represent 2D transformation by a matrix

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- Multiply matrix by column vector
  $\Leftrightarrow$ apply transformation to point

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

$$x' = ax + by$$

$$y' = cx + dy$$

Matrix Representation

- Transformations combined by multiplication

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

Matrices are a convenient and efficient way to represent a sequence of transformations!

2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

Only linear 2D transformations can be represented with a 2x2 matrix

Linear Transformations

- Linear transformations are combinations of …
  - Scale, and
  - Rotation

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

- Properties of linear transformations:
  - Satisfies: $T(s_1\mathbf{p}_1 + s_2\mathbf{p}_2) = s_1T(\mathbf{p}_1) + s_2T(\mathbf{p}_2)$
  - Origin maps to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Closed under composition

Linear Transformations

- Linear transformations are combinations of …
  - Scale, and
  - Rotation

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

- Properties of linear transformations:
  - Satisfies: $\boxed{T(s_1\mathbf{p}_1 + s_2\mathbf{p}_2) = s_1 T(\mathbf{p}_1) + s_2 T(\mathbf{p}_2)}$
  - Origin maps to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Closed under composition

Translations do not map the origin to the origin

2D Translation

- 2D translation represented by a 3x3 matrix
  - Point represented with *homogeneous coordinates*

$$x' = x + tx * w$$

$$y' = y + ty * w$$

$$w' = w$$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

2D Translation

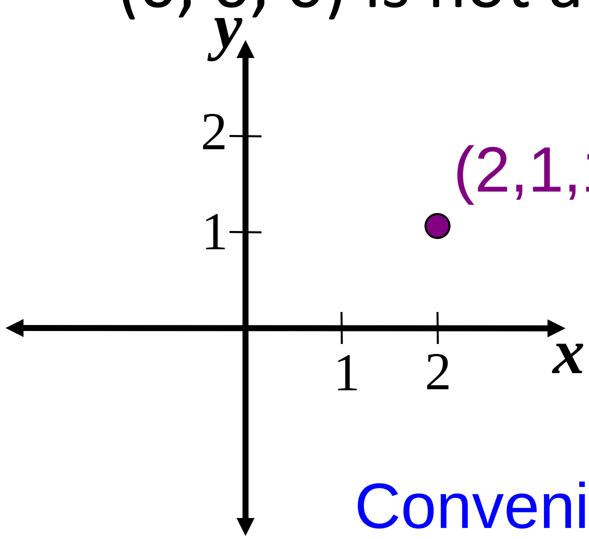
- 2D translation represented by a 3x3 matrix
  - Point represented with *homogeneous coordinates*

$$x' = x + tx$$
$$y' = y + ty$$

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Homogeneous Coordinates

- Add a 3rd coordinate to every 2D point
  - (x, y, w) represents a point at location (x/w, y/w)
  - (x, y, 0) represents a point at infinity
  - (0, 0, 0) is not allowed



(2,1,1) or (4,2,2) or (6,3,3)

Convenient coordinate system to
represent many useful transformations

Basic 2D Transformations

- Basic 2D transformations as 3x3 matrices

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

Translate

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

Scale

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

Rotate

Affine Transformations

- Affine transformations are combinations of …
  - Linear transformations, and
  - Translations

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

- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Closed under composition

Projective Transformations

- Projective transformations …
  - Affine transformations, and
  - Projective warps

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

- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Closed under composition

Matrix Composition

- Matrices are a convenient and efficient way to represent a sequence of transformations
  - General purpose representation
  - Hardware matrix multiply
  - Efficiency with pre-multiplication
    - Matrix multiplication is associative

$$\mathbf{p}' = (T * (R * (S*\mathbf{p})\ )\ )$$
$$\mathbf{p}' = (T*R*S) * \mathbf{p}$$

3D Transformations

- Same idea as 2D transformations
  - Homogeneous coordinates: (x,y,z,w)
  - 4x4 transformation matrices

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Basic 3D Transformations

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

Identity

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

Scale

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Translation

Basic 3D Transformations

Pitch-Roll-Yaw Convention:

- Any rotation can be expressed as the combination of a rotation about the *x*-, the *y*-, and the *z*-axis.

Rotate around Z axis: $$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 & 0 \\ \sin\Theta & \cos\Theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Rotate around Y axis: $$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} \cos\Theta & 0 & \sin\Theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Theta & 0 & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Rotate around X axis: $$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\Theta & -\sin\Theta & 0 \\ 0 & \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Homogenous Coordinates

In 3D we can represent points by the 4-tuple:

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

under the relationship that the position is unchanged by scale:

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

for all non-zero values α.

Homogenous Coordinates

<u>Two cases:</u>

1. $w \neq 0$: The 4-tuple represents a point

$$(x,y,z,w)=(x/w,y/w,z/w,1)$$

2. $w=0$: The 4-tuple represents a (unit) vector

$$(x,y,z,w)=(x,y,z,0)/\sqrt{x^2+y^2+z^2}$$

Applying a Transformation

- Position
- Direction
- Normal

$$\underset{M}{\overset{\text{Affine}}{\begin{bmatrix} a & b & c & tx \\ d & e & f & ty \\ g & h & i & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Linear}}{\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}}$$

Applying a Transformation

- Position
  - Apply the full affine transformation:

$$p' = M(p) = (M_T \times M_L)(p) = M(p_x, p_y, p_z, 1)$$

$$\underset{M}{\overset{\text{Affine}}{\begin{bmatrix} a & b & c & tx \\ d & e & f & ty \\ g & h & i & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Linear}}{\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}}$$

Applying a Transformation

- Direction
  - Apply the linear component of the transformation:

$$p' = M_L(p) = M(p_x, p_y, p_z, 0)$$

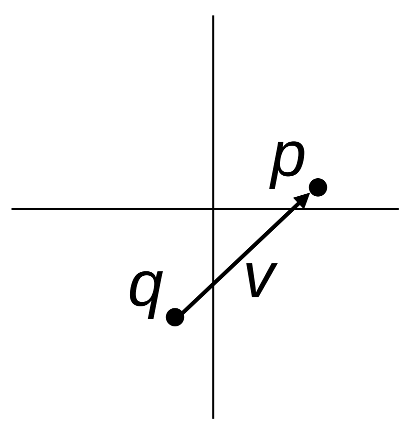
$$\underset{M}{\overset{\text{Affine}}{\begin{bmatrix} a & b & c & tx \\ d & e & f & ty \\ g & h & i & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Linear}}{\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}}$$

Applying a Transformation

- Direction
  - Apply the linear component of the transformation:

$$p' = M_L(p) = M(p_x, p_y, p_z, 0)$$

A direction vector v is defined as the difference between two positional vectors p and q: $\boldsymbol{v=p-q}$.



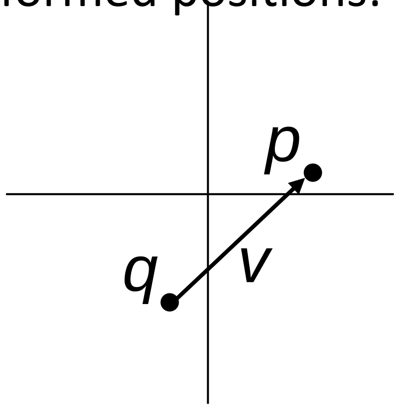
Applying a Transformation

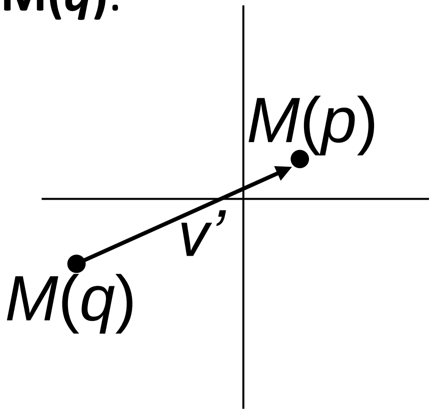
- Direction
  - Apply the linear component of the transformation:

$$p' = M_L(p) = M(p_x, p_y, p_z, 0)$$

A direction vector *v* is defined as the difference between two positional vectors *p* and *q*: ***v=p-q***.

Applying the transformation M, we compute the transformed direction as the distance between the transformed positions: ***v'*=M(*p*)-M(*q*)**.





Applying a Transformation

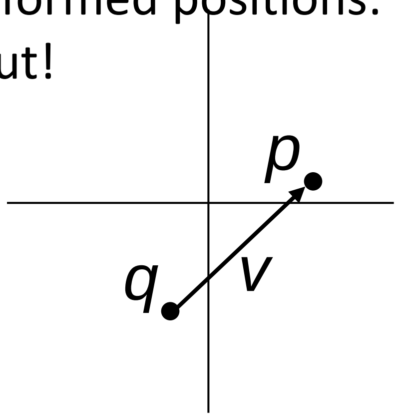
- Direction
  - Apply the linear component of the transformation:

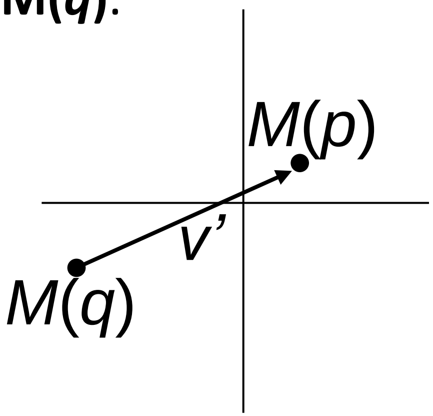
$$p' = M_L(p) = M(p_x, p_y, p_z, 0)$$

A direction vector *v* is defined as the difference between two positional vectors *p* and *q*: ***v=p-q***.

Applying the transformation M, we compute the transformed direction as the distance between the transformed positions: ***v'*=M(*p*)-M(*q*)**.

The translation terms cancel out!





Applying a Transformation

- Normal

$p' = ?$

$$\begin{array}{ccccc} \text{Affine} & & \text{Translate} & & \text{Linear} \\ \begin{bmatrix} a & b & c & tx \\ d & e & f & ty \\ g & h & i & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} & = & \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} & \times & \begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ M & & M_T & & M_L \end{array}$$

Normal Transformation

2D Example:

$$\underset{M}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Scale}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}}$$

Normal Transformation

2D Example:

$$\underset{M}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Scale}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}}$$

If *v* is a direction in 2D, and *n* is a vector perpendicular to *v*, we want the transformed *n* to be perpendicular to the transformed *v*:

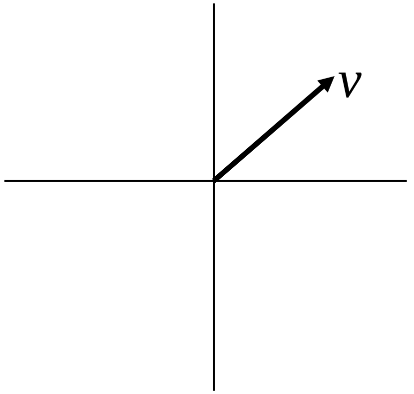
$$\langle v, n \rangle = 0 \quad \Longrightarrow \quad \langle M_L(v), n' \rangle = 0$$

Normal Transformation

2D Example:

$$\underset{M}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Scale}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}}$$

Say $v = (2,2)$...

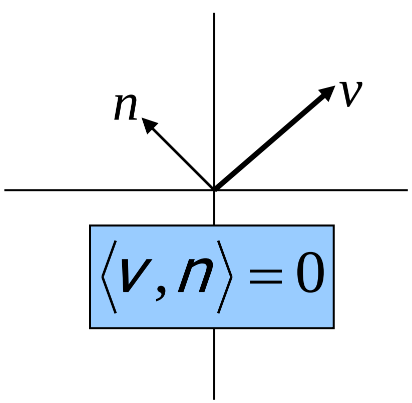


Normal Transformation

2D Example:

$$\underset{M}{\begin{bmatrix}1 & 0 & 1\\ 0 & 2 & 1\\ 0 & 0 & 1\end{bmatrix}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix}1 & 0 & 1\\ 0 & 1 & 1\\ 0 & 0 & 1\end{bmatrix}}} \times \underset{M_L}{\overset{\text{Scale}}{\begin{bmatrix}1 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 1\end{bmatrix}}}$$

Say $v = (2,2)$... then $n = \left(-\sqrt{.5}, \sqrt{.5}\right)$



$$\boxed{\langle v, n\rangle = 0}$$

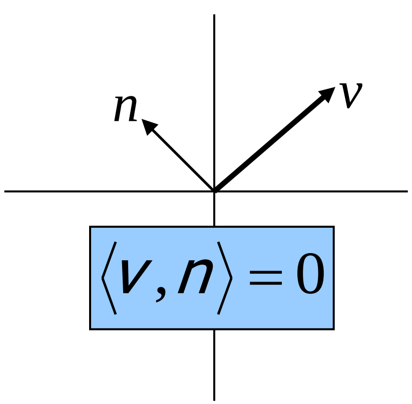
Normal Transformation

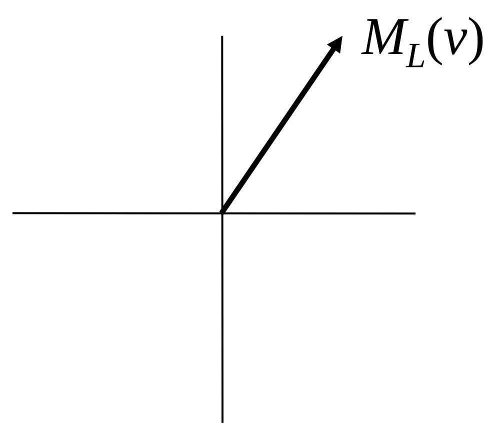
2D Example:

$$\underset{M}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Scale}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}}$$

Say $v = (2,2)$... then $n = \left(-\sqrt{.5}, \sqrt{.5}\right)$

Transforming, $M_L(v) = (2,4)$...





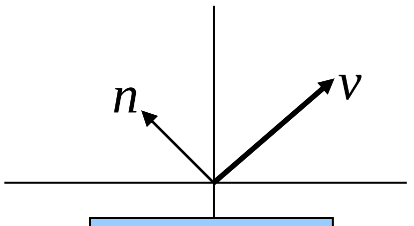
Normal Transformation

2D Example:

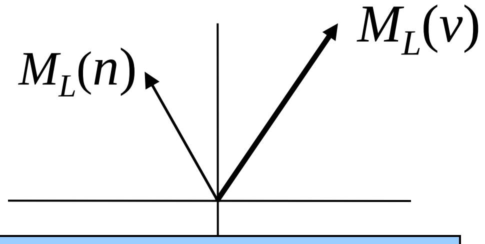
$$\underset{M}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}} \times \underset{M_L}{\overset{\text{Scale}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}}}$$

Say $v = (2,2)$… then $n = \left(-\sqrt{.5}, \sqrt{.5}\right)$

Transforming, $M_L(v) = (2,4)$… and $M_L(n) = \left(-\sqrt{.5}, \sqrt{2}\right)$



$\langle v, n \rangle = 0$



$\langle M_L(v), M_L(n) \rangle \neq 0$

Normal Transformation

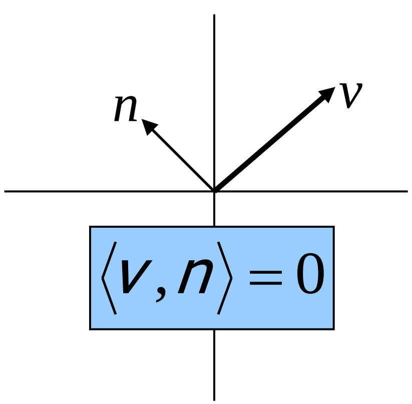
2D Example:

$$\underset{M}{\begin{bmatrix}1&0&1\\0&2&1\\0&0&1\end{bmatrix}} = \underset{M_T}{\overset{\text{Translate}}{\begin{bmatrix}1&0&1\\0&1&1\\0&0&1\end{bmatrix}}} \times \underset{M_L}{\overset{\text{Scale}}{\begin{bmatrix}1&0&0\\0&2&0\\0&0&1\end{bmatrix}}}$$

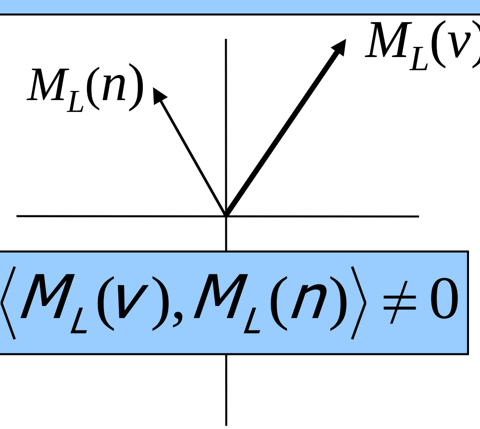
Say $v =$

Transf ... $.5,\sqrt{2}\big)$

Simply applying the directional part of the transformation to *n* does not result in a vector that is perpendicular to the transformed *v*.



$\langle v, n\rangle = 0$



$\langle M_L(v), M_L(n)\rangle \neq 0$

Recall

Transposes:

- The transpose of a matrix M is the matrix M^t whose (i,j)-th coeff. is the (j,i)-th coeff. of M:

$$M = \begin{bmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \end{bmatrix} \qquad M^t = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

Recall

Transposes:

- The transpose of a matrix M is the matrix M^t whose (i,j)-th coeff. is the (j,i)-th coeff. of M:

$$M = \begin{bmatrix} m_{11} & m_{21} & m_{31} \\ m_{12} & m_{22} & m_{32} \\ m_{13} & m_{23} & m_{33} \end{bmatrix} \qquad M^t = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

- If M and N are two matrices, then the transpose of the product is the inverted product of the transposes:

$$(MN)^t = N^t M^t$$

Recall

Dot-Products:

- The dot product of two vectors $v=(v_x, v_y, v_z)$ and $w=(w_x, w_y, w_z)$ is obtained by summing the product of the coefficients:

$$\langle v, w \rangle = v_x w_x + v_y w_y + v_z w_z$$

Recall

Dot-Products:

- The dot product of two vectors $v=(v_x,v_y,v_z)$ and $w=(w_x,w_y,w_z)$ is obtained by summing the product of the coefficients:

$$\langle v,w\rangle = v_x w_x + v_y w_y + v_z w_z$$

- We can also express this as a matrix product:

$$\langle v,w\rangle = v^t w = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix}$$

Recall

Transposes and Dot-Products:

- If *M* is a matrix, the dot product of *v* with *M* applied to *w* is the dot product of the transpose of *M* applied to *v* with *w*:

Recall

Transposes and Dot-Products:

- If *M* is a matrix, the dot product of *v* with *M* applied to *w* is the dot product of the transpose of *M* applied to *v* with *w*:

$$\langle v, Mw\rangle = v^t(Mw)$$

Recall

Transposes and Dot-Products:

- If M is a matrix, the dot product of v with M applied to w is the dot product of the transpose of M applied to v with w:

$$\begin{aligned}\langle v, Mw \rangle &= v^t (Mw) \\ &= (v^t M) w\end{aligned}$$

Recall

Transposes and Dot-Products:

- If M is a matrix, the dot product of v with M applied to w is the dot product of the transpose of M applied to v with w:

$$\begin{aligned}\langle v, Mw\rangle &= v^t(Mw)\\ &= (v^t M)w\\ &= (M^t v)^t w\end{aligned}$$

Recall

Transposes and Dot-Products:

- If M is a matrix, the dot product of v with M applied to w is the dot product of the transpose of M applied to v with w:

$$
\begin{aligned}
\langle v, Mw \rangle &= v^t (Mw) \\
&= (v^t M) w \\
&= (M^t v)^t w \\
\langle v, Mw \rangle &= \langle M^t v, w \rangle
\end{aligned}
$$

Applying a Transformation

- If we apply the transformation M to 3D space, how does it act on normals?

Applying a Transformation

- If we apply the transformation *M* to 3D space, how does it act on normals?
- A normal *n* is defined by being perpendicular to some vector(s) *v*. The transformed normal *n′* should be perpendicular to *M*(*v*):

$$\langle n, v \rangle = \langle n', Mv \rangle$$

Applying a Transformation

- If we apply the transformation *M* to 3D space, how does it act on normals?
- A normal *n* is defined by being perpendicular to some vector(s) *v*. The transformed normal *n′* should be perpendicular to *M*(*v*):

$$\begin{aligned}\langle n, v\rangle &= \langle n', Mv\rangle \\ &= \langle M^t n', v\rangle\end{aligned}$$

Applying a Transformation

- If we apply the transformation *M* to 3D space, how does it act on normals?
- A normal *n* is defined by being perpendicular to some vector(s) *v.* The transformed normal *n'* should be perpendicular to *M*(*v*):

$$\begin{aligned}\langle n, v\rangle &= \langle n', Mv\rangle \\ &= \langle M^t n', v\rangle\end{aligned}$$

$$\Updownarrow$$

$$n = M^t n'$$

Applying a Transformation

- If we apply the transformation *M* to 3D space, how does it act on normals?
- A normal *n* is defined by being perpendicular to some vector(s) *v*. The transformed normal *n'* should be perpendicular to *M*(*v*):

$$\langle n, v \rangle = \langle n', Mv \rangle$$

$$= \langle M^t n', v \rangle$$

$$\Updownarrow$$

$$n = M^t n'$$

$$\Updownarrow$$

$$n' = \left(M^t\right)^{-1} n$$

Quaternions

Quaternions are extensions of complex numbers, with 3 imaginary values instead of 1:

$$a + ib + jc + kd$$

Quaternions

Quaternions are extensions of complex numbers, with 3 imaginary values instead of 1:

$$a + ib + jc + kd$$

Like the complex numbers, we can add quaternions together by summing the individual components:

$$\begin{array}{r} (a_1 + ib_1 + jc_1 + kd_1) \\ +\, (a_2 + ib_2 + jc_2 + kd_2) \\ \hline = (a_1 + a_2) + i(b_1 + b_2) + j(c_1 + c_2) + k(d_1 + d_2) \end{array}$$

Quaternions

Quaternions are extensions of complex numbers, with 3 imaginary values instead of 1:

$$a + ib + jc + kd$$

Like the imaginary component of complex numbers, squaring the components gives:

$$i^2 = j^2 = k^2 = -1$$

Quaternions

Quaternions are extensions of complex numbers, with 3 imaginary values instead of 1:

$$a + ib + jc + kd$$

Like the imaginary component of complex numbers, squaring the components gives:

$$i^2 = j^2 = k^2 = -1$$

However, the multiplication rules are more complex: $ij = k \qquad ik = -j \qquad jk = i$

$$ji = -k \qquad ki = j \qquad kj = -i$$

Quaternions

Quaternions are extensions of complex numbers, with 3 imaginary values instead of 1:

$$a + ib + jc + kd$$

Like the … x

number … :

> Note that multiplication of quaternions is not commutative:
> The result of the multiplication depends on the order in which it was done

However, the multiplication rules are more complex:

$$ij = k \qquad ik = -j \qquad jk = i$$

$$ji = -k \qquad ki = j \qquad kj = -i$$

Quaternions

More generally, the product of two quaternions is:

$$
\begin{array}{r}
\left(a_1 + ib_1 + jc_1 + kd_1\right) \\
\times \left(a_2 + ib_2 + jc_2 + kd_2\right) \\
\hline
\end{array}
$$

$$
\begin{aligned}
= &\left(a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2\right) \\
+ i&\left(a_1b_2 + a_2b_1 + c_1d_2 - c_2d_1\right) \\
+ j&\left(a_1c_2 + a_2c_1 - b_1d_2 + b_2d_1\right) \\
+ k&\left(a_1d_2 + a_2d_1 + b_1c_2 - b_2c_1\right)
\end{aligned}
$$

$$i^2 = j^2 = k^2 = -1$$

$$ij = k \qquad ik = -j \qquad jk = i$$

$$ji = -k \qquad ki = j \qquad kj = -i$$

Quaternions

As with complex numbers, we define the <u>conjugate</u> of a quaternion $q=a+ib+jc+kd$ as:

$$\bar{q} = a - ib - jc - kd$$

Quaternions

As with complex numbers, we define the <u>conjugate</u> of a quaternion $q=a+ib+jc+kd$ as:

$$\bar{q} = a - ib - jc - kd$$

As with complex numbers, we define the <u>norm</u> of a quaternion $q=a+ib+jc+kd$ as:

$$\|q\| = \sqrt{a^2 + b^2 + c^2 + d^2} = \sqrt{q\bar{q}}$$

Quaternions

As with complex numbers, we define the <u>conjugate</u> of a quaternion $q=a+ib+jc+kd$ as:

$$\bar{q} = a - ib - jc - kd$$

As with complex numbers, we define the <u>norm</u> of a quaternion $q=a+ib+jc+kd$ as:

$$\|q\| = \sqrt{a^2 + b^2 + c^2 + d^2} = \sqrt{q\bar{q}}$$

As with complex numbers, the <u>reciprocal</u> is defined by dividing the conjugate by the square norm:

$$\frac{1}{q} = \frac{\bar{q}}{\|q\|^2}$$

Quaternions

One way to express a quaternion is as a pair consisting of the real value and the 3D vector consisting of the imaginary components:

$$q = (\alpha, w) \quad \text{with} \quad \alpha = a,\ w = (b, c, d)$$

Quaternions

One way to express a quaternion is as a pair consisting of the real value and the 3D vector consisting of the imaginary components:

$$q = (\alpha, w) \quad \text{with} \quad \alpha = a,\ w = (b, c, d)$$

The advantage of this representation is that it is easier to express quaternion multiplication:

$$\begin{aligned} q_1 q_2 &= (\alpha_1, w_1)(\alpha_2, w_2) \\ &= \left(\alpha_1\alpha_2 - \langle w_1, w_2 \rangle, \alpha_1 w_2 + \alpha_2 w_1 + w_1 \times w_2\right) \end{aligned}$$

$$
\begin{aligned}
q_1q_2 = \; & \left(a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2\right) \\
& + i\left(a_1b_2 + a_2b_1 + c_1d_2 - c_2d_1\right) \\
& + j\left(a_1c_2 + a_2c_1 - b_1d_2 + b_2d_1\right) \\
& + k\left(a_1d_2 + a_2d_1 + b_1c_2 - b_2c_1\right)
\end{aligned}
$$

One way … a pair consisting … vector consisting of the imaginary components:

$$q = (\alpha, w) \quad \text{with} \quad \alpha = a,\; w = (b, c, d)$$

The advantage of this representation is that it is easier to express quaternion multiplication:

$$
\begin{aligned}
q_1q_2 &= (\alpha_1, w_1)(\alpha_2, w_2) \\
&= \left(\alpha_1\alpha_2 - \langle w_1, w_2\rangle, \alpha_1 w_2 + \alpha_2 w_1 + w_1 \times w_2\right)
\end{aligned}
$$

Quaternions

Given a 3D vector (x,y,z), we can think of the vector as an imaginary quaternion:

$$v=ix+jy+kz$$

Quaternions

Given a 3D vector (x,y,z), we can think of the vector as an imaginary quaternion:

$$v=ix+jy+kz$$

Then multiplying v by a quaternion q on the left and its conjugate on the right gives:

$$q \cdot v \cdot \overline{q} = \left(0, w\right)$$

Quaternions

Given a 3D vector (*x*,*y*,*z*), we can think of the vector as an imaginary quaternion:

$$v = ix + jy + kz$$

Then multiplying *v* by a quaternion *q* on the left and its co

This means that unit quaternions correspond to rotations in 3D.

$$q \cdot v \cdot q = (0, w)$$

The mapping is linear in *v*, and preserves lengths when *q* is a unit quaternion.

Quaternions and Rotations

If $q=a+ib+jc+kd$ is a unit quaternion ($||q||=1$), q corresponds to a rotation:

$$R(q)=\begin{pmatrix} 1-2c^2-2d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & 1-2b^2-2d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & 1-2b^2-2c^2 \end{pmatrix}$$

Note that because all of the terms are quadratic, the rotation associated with q is the same as the rotation associated with $-q$.

Quaternions and Rotations

If $q=a+ib+jc+kd$ is a unit quaternion ($||q||=1$), q corresponds to a rotation:

$$R(q)=\begin{pmatrix} 1-2c^2-2d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & 1-2b^2-2d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & 1-2b^2-2c^2 \end{pmatrix}$$

Since q is a unit quaternion, we can write q as:

$$q=\left(\cos(\theta/2),\sin(\theta/2)w\right) \qquad \|w\|=1$$

Quaternions and Rotations

If $q=a+ib+jc+kd$ is a unit quaternion ($||q||=1$), q corresponds to a rotation:

$$R(q)=\begin{pmatrix} 1-2c^2-2d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & 1-2b^2-2d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & 1-2b^2-2c^2 \end{pmatrix}$$

Since q is a unit quaternion, we can write q as:

$$q=\left(\cos(\theta/2),\sin(\theta/2)w\right) \qquad \|w\|=1$$

It turns out that q corresponds to the rotation:

- Whose axis of rotation is w, and
- Whose angle of rotation is θ.

Quaternions and Rotations

$$R(q) = \begin{pmatrix} 1-2c^2-2d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & 1-2b^2-2d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & 1-2b^2-2c^2 \end{pmatrix}$$

Since $R(q)$ preserves distances, shortest paths between rotations can be obtained by computing shortest paths between unit quaternions.

Quaternions and Rotations

$$R(q) = \begin{pmatrix} 1-2c^2-2d^2 & 2bc-2ad & 2bd+2ac \\ 2bc+2ad & 1-2b^2-2d^2 & 2cd-2ab \\ 2bd-2ac & 2cd+2ab & 1-2b^2-2c^2 \end{pmatrix}$$

Since $R(q)$ preserves distances, shortest paths between rotations can be obtained by computing shortest paths between unit quaternions.

But shortest paths between two points on a sphere are just a great arcs (SLERP).