Lecture 1 Supplementary Material: Linear Algebra

Computer Animation

Vector Arithmetic

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

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

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

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

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

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

Vector Magnitude

- The magnitude (length) of a vector is:

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

- Unit vector (magnitude=1.0)

$$\frac{\mathbf{v}}{|\mathbf{v}|}$$

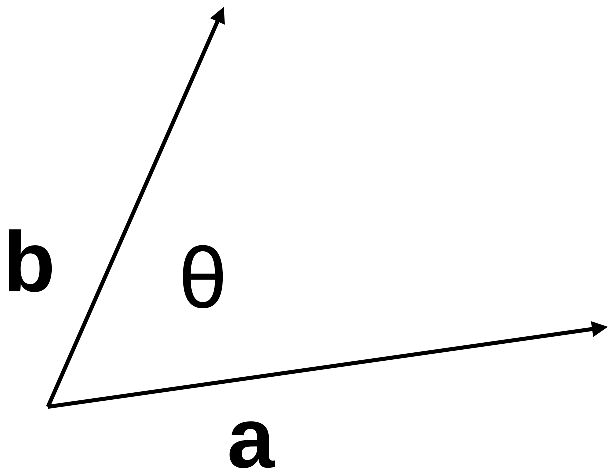
Dot Product

$$\mathbf{a} \cdot \mathbf{b} = a_x b_x + a_y b_y + a_z b_z = \sum a_i b_i$$

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

Example: Angle Between Vectors

- How do you find the angle θ between vectors **a** and **b**?

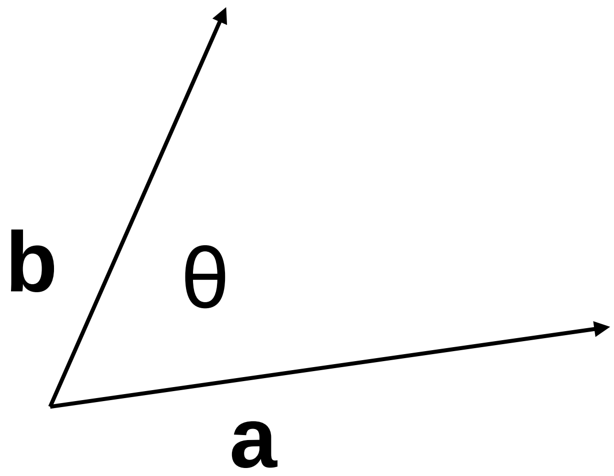


Example: Angle Between Vectors

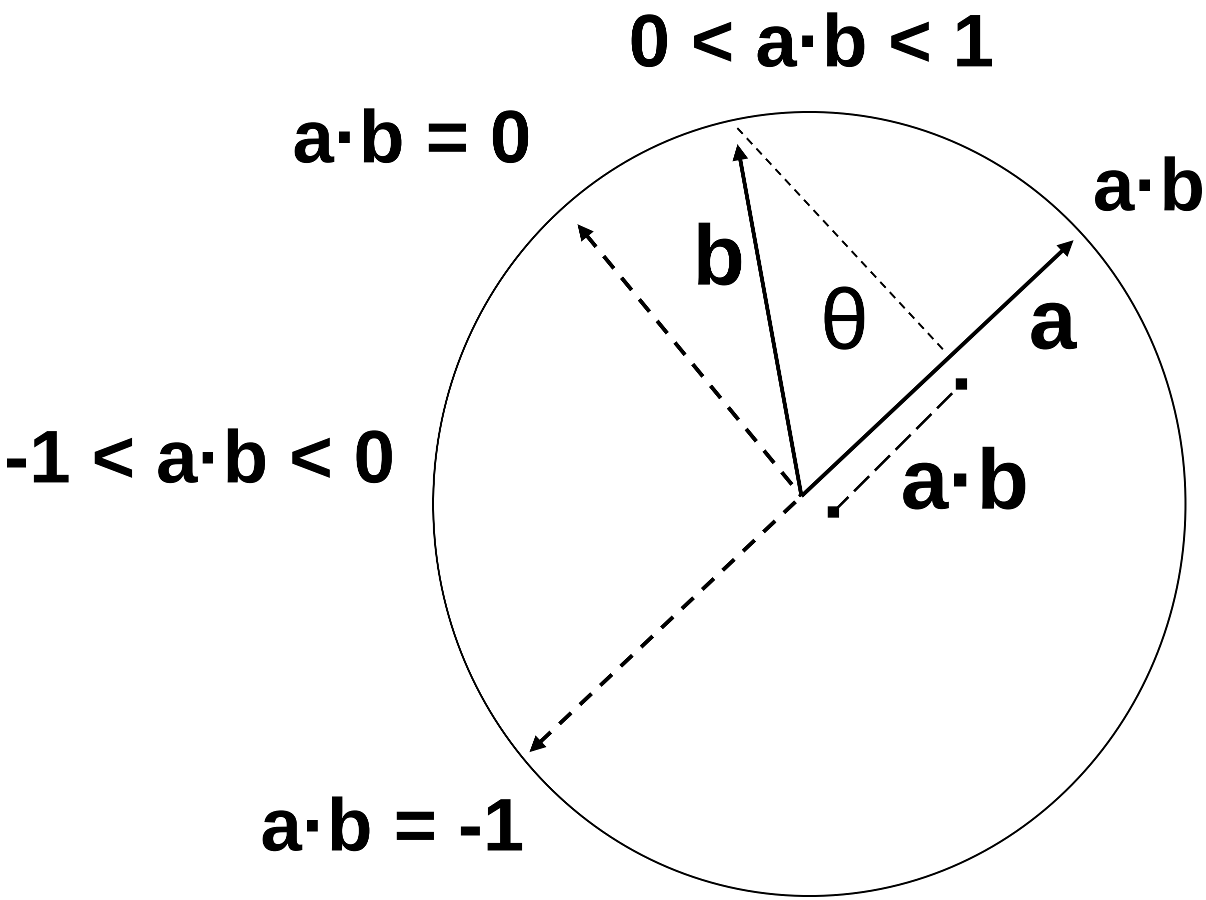
$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos\theta$$

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

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



Dot Products with Unit Vectors



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

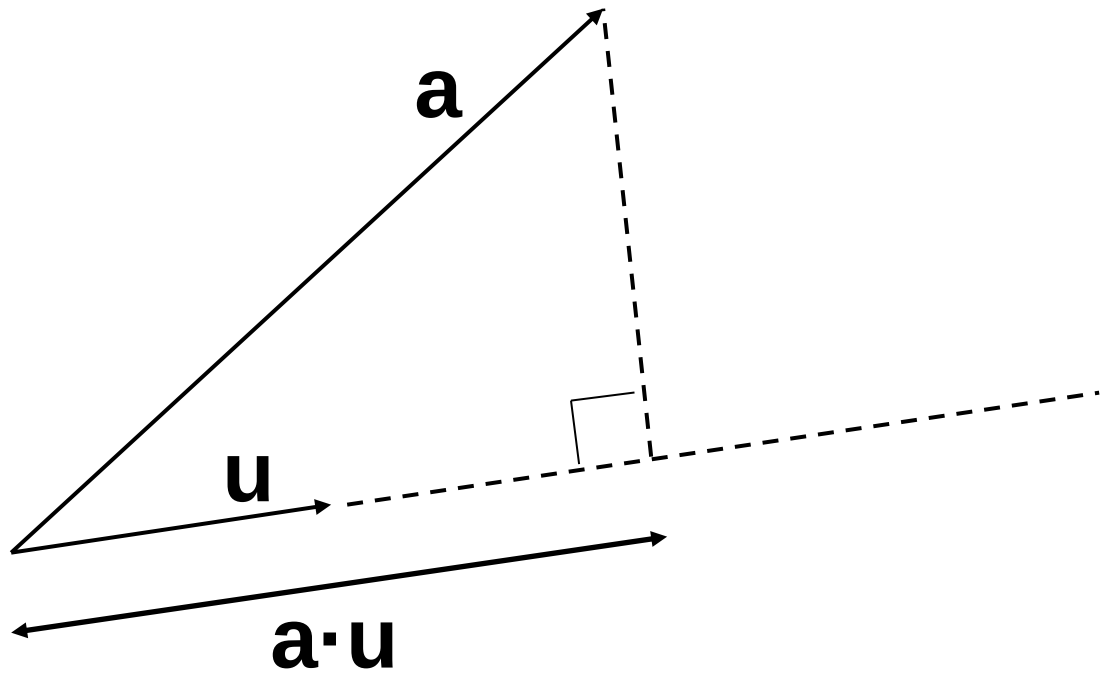
$$\mathbf{a} \cdot \mathbf{b} = \cos(\theta)$$

Dot Products with Non-Unit Vectors

- If **a** and **b** are arbitrary (non-unit) vectors, then the following are still true:
  - If $\theta < 90°$ then $\mathbf{a}\cdot\mathbf{b} > 0$
  - If $\theta = 90°$ then $\mathbf{a}\cdot\mathbf{b} = 0$
  - If $\theta > 90°$ then $\mathbf{a}\cdot\mathbf{b} < 0$

Dot Products with One Unit Vector

- If $|\mathbf{u}|=1.0$ then $\mathbf{a}\cdot\mathbf{u}$ is the length of the *projection* of **a** onto **u**



Cross Product

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

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

Properties of the Cross Product

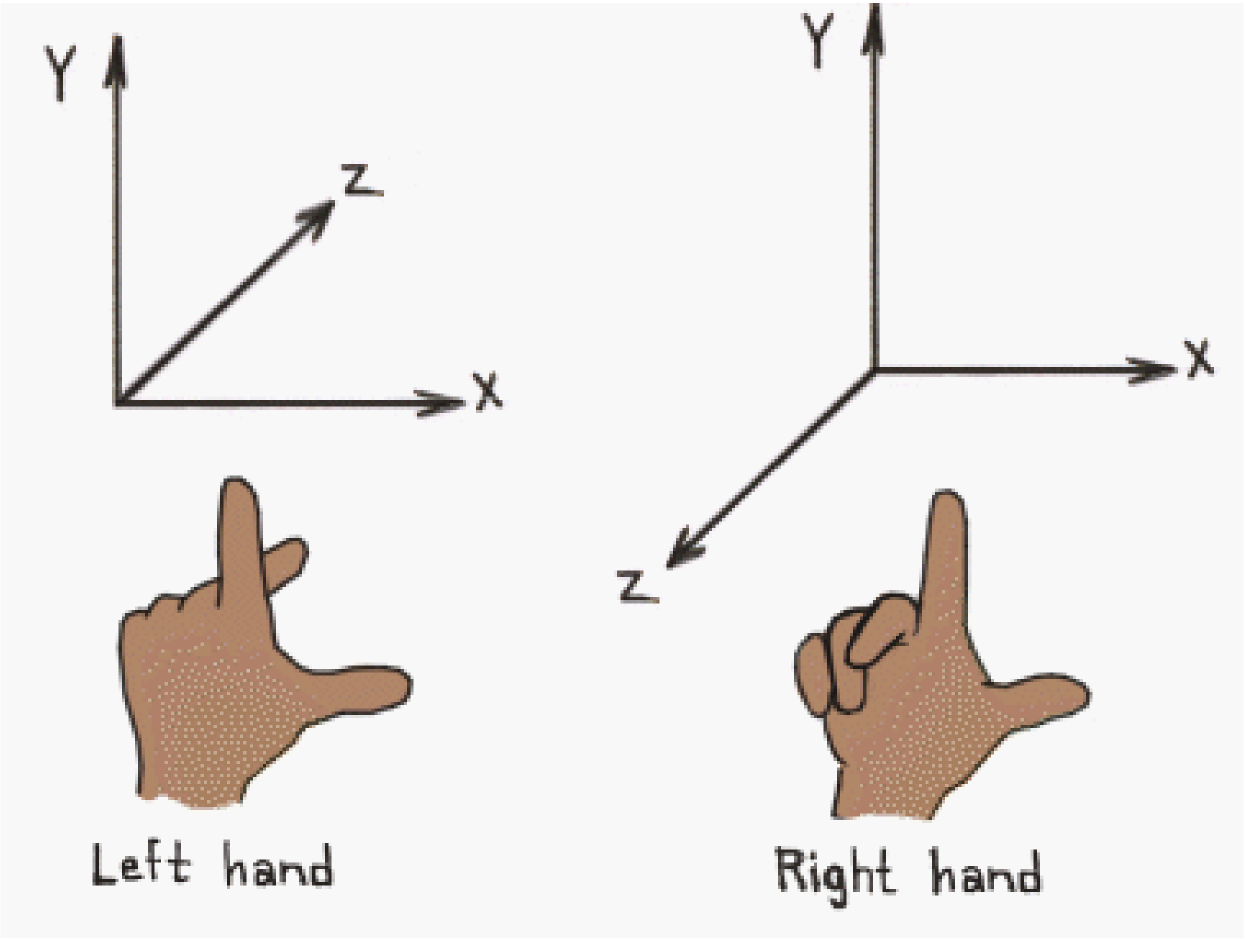
$$|\mathbf{a}\times\mathbf{b}| = |\mathbf{a}||\mathbf{b}|\sin\theta$$

$|\mathbf{a}\times\mathbf{b}| =$ area of parallelogram **ab**

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

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

Right and left handed coordinate system



Dot vs Cross product

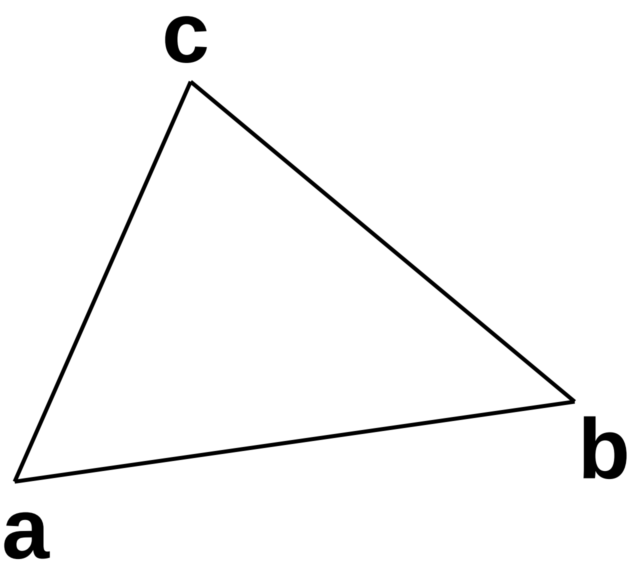
- **Dot product** produces a **scalar**
- **Cross product** of two vectors is a **vector**

- **Dot product** applies to **n dimensional** vectors
- **Cross product** applies to **3 dimensional** vectors

- Intuition:
  - **Dot product** shows how much part of the vector **a** is in the same direction as vector **b**
  - **Cross product** is how much part of the vector **a** is perpendicular to the vector **b**

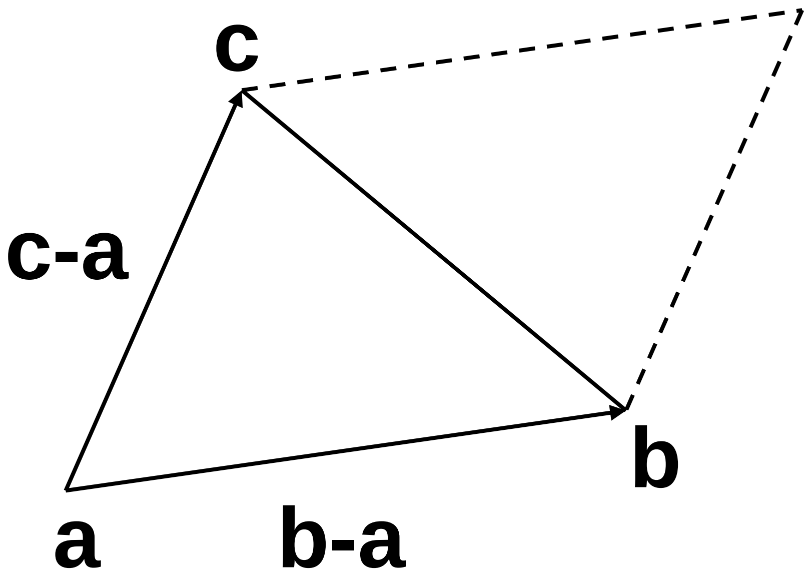
Example: Area of a Triangle

- Find the area of the triangle defined by 3D points **a**, **b**, and **c**



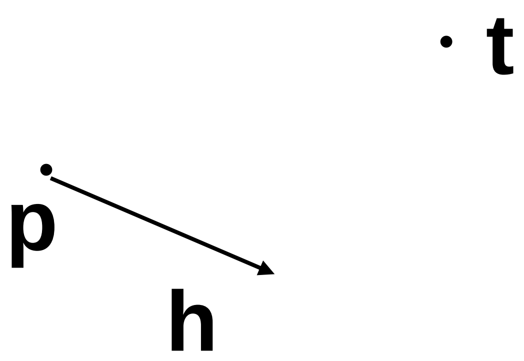
Example: Area of a Triangle

$$area = \frac{1}{2}\left|\left(\mathbf{b}-\mathbf{a}\right)\times\left(\mathbf{c}-\mathbf{a}\right)\right|$$



Example: Alignment to Target

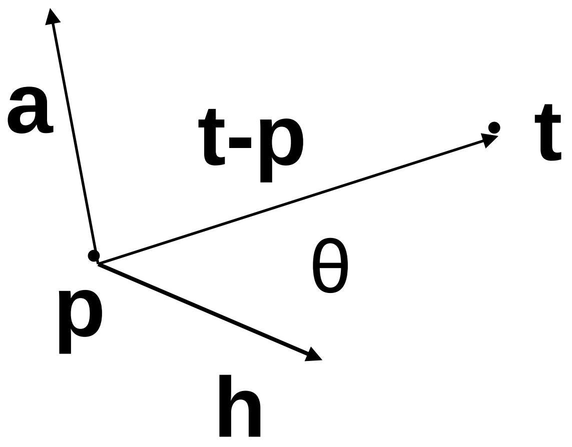
- An object is at position **p** with a unit length heading of **h**. We want to rotate it so that the heading is facing some target **t**. Find a unit axis **a** and an angle θ to rotate around.



Example: Alignment to Target

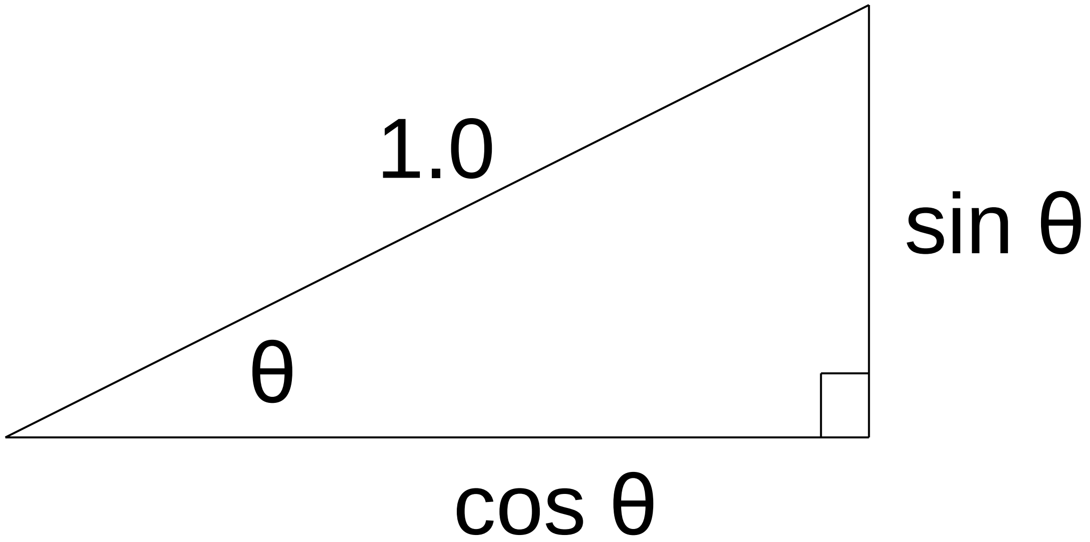
$$\mathbf{a} = \frac{\mathbf{h} \times (\mathbf{t} - \mathbf{p})}{|\mathbf{h} \times (\mathbf{t} - \mathbf{p})|}$$

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



Trigonometry

$\cos^2\theta + \sin^2\theta = 1$



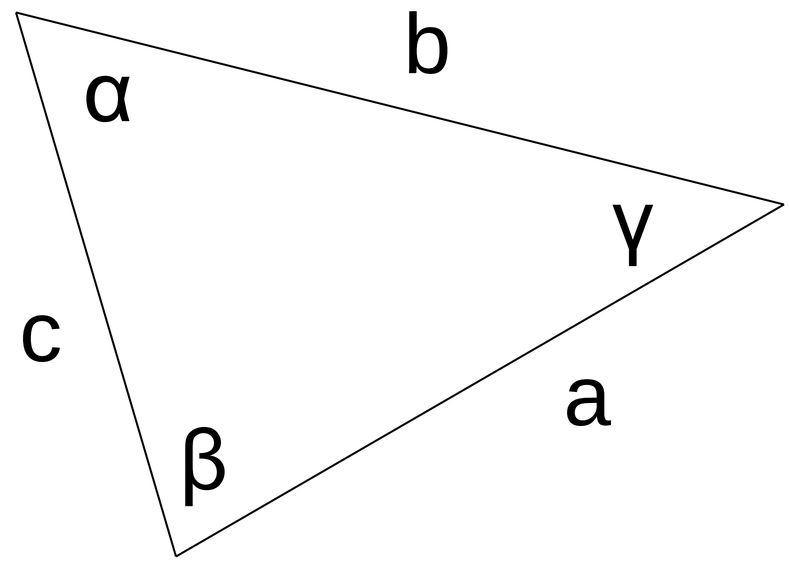
Laws of Sines and Cosines

- Law of Sines:

$$\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma}$$

- Law of Cosines:

$$c^2 = a^2 + b^2 - 2ab\cos\gamma$$



Determinant and Inverse of 2x2 matrix

- A= $\begin{bmatrix} 5 & 3 \\ -1 & 4 \end{bmatrix}$
- Det $A = \begin{vmatrix} 5 & 3 \\ -1 & 4 \end{vmatrix}$= 5.4 – (-1.3) = 23
- Inv $A = \frac{1}{\det(A)}$. adj A = $\frac{1}{23}$. $\begin{bmatrix} 4 & -3 \\ 1 & 5 \end{bmatrix}$
- Adj A = $\begin{bmatrix} 4 & -3 \\ 1 & 5 \end{bmatrix}$

Vector Dot Vector

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

$$\mathbf{a} \cdot \mathbf{b} = a_x b_x + a_y b_y + a_z b_z$$

Matrix Dot Matrix

$$\mathbf{L} = \mathbf{M} \cdot \mathbf{N}$$

$$\begin{bmatrix} l_{11} & l_{12} & l_{13} \\ l_{21} & l_{22} & l_{23} \\ l_{31} & l_{32} & l_{33} \end{bmatrix} = \begin{bmatrix} \cancel{m_{11}} & \cancel{m_{12}} & \cancel{m_{13}} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \cdot \begin{bmatrix} n_{11} & \cancel{n_{12}} & n_{13} \\ n_{21} & \cancel{n_{22}} & n_{23} \\ n_{31} & \cancel{n_{32}} & n_{33} \end{bmatrix}$$

$$l_{12} = m_{11}n_{12} + m_{12}n_{22} + m_{13}n_{32}$$

Translation

- Let's say we have a 3D model that has an array of position vectors describing its shape
  - We store all position vectors: v_n where $0 \le n \le NumVerts-1$
- If we want to move the object (translate)
  - $v_n' = v_n + d$ (relative offset)

$$\begin{bmatrix} v_x' & v_y' & v_z' \end{bmatrix} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix} + \begin{bmatrix} d_x & d_y & d_z \end{bmatrix}$$

  - $v_x' = v_x + d_x$
  - $v_y' = v_y + d_y$
  - $v_z' = v_z + d_z$

Identity matrix and translation

$$\begin{bmatrix} v_x' & v_y' & v_z' \end{bmatrix} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} d_x & d_y & d_z \end{bmatrix}$$

$$v_x' = v_x 1 + v_y 0 + v_z 0 + d_x$$
$$v_y' = v_x 0 + v_y 1 + v_z 0 + d_y$$
$$v_z' = v_x 0 + v_y 0 + v_z 1 + d_z$$

Rotation

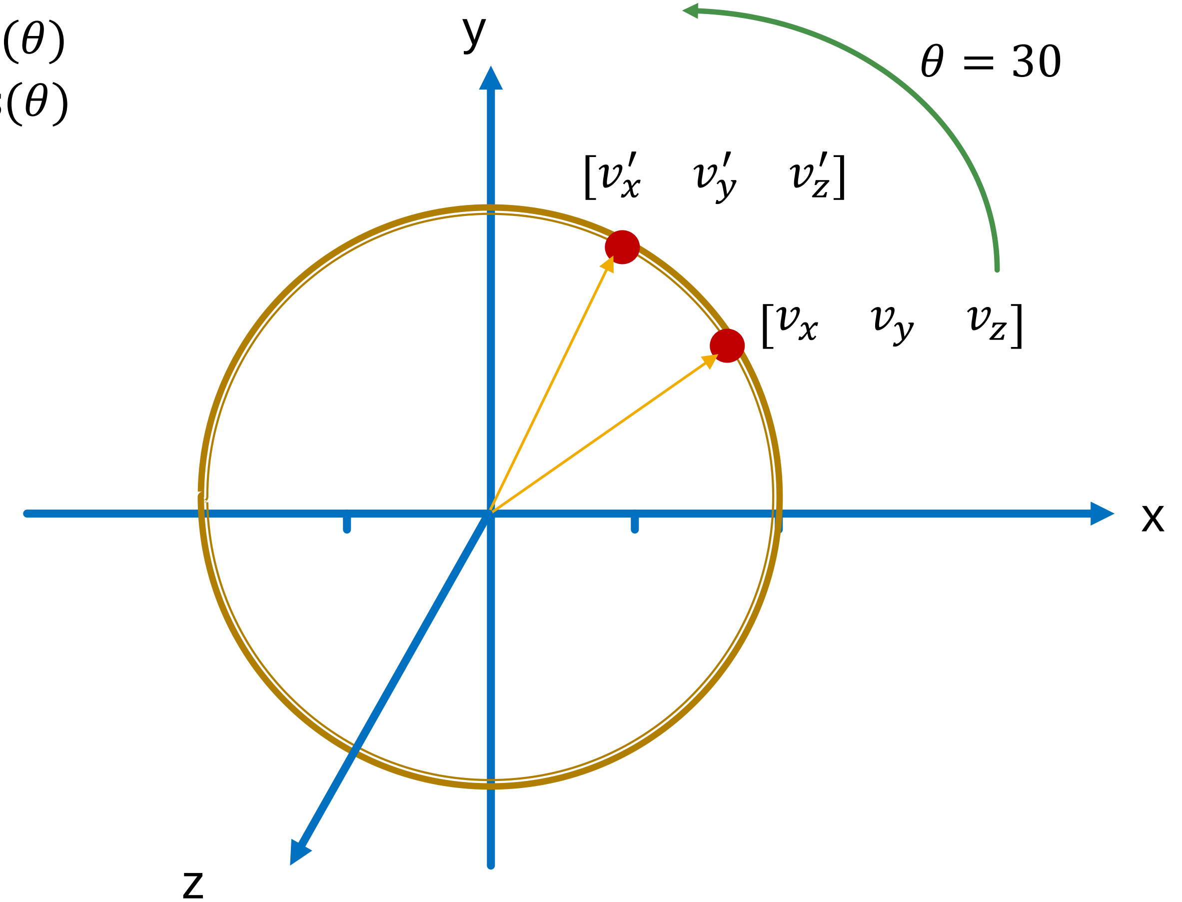
- Now, let's rotate the object in the xy plane by an angle θ, as if we were spinning it around the z axis

- $v_x' = v_x \cos(\theta) - v_y \sin(\theta)$
- $v_y' = v_x \, sin(\theta) + v_y \cos(\theta)$
- $v_z' = v_z$

- Note: a positive rotation will rotate the object counterclockwise when the rotation axis (z) is pointing towards the observer

Example

$$v'_x = v_x \cos(\theta) - v_y \sin(\theta)$$
$$v'_y = v_x\, sin(\theta) + v_y \cos(\theta)$$
$$v'_z = v_z$$



Rotation

- $v'_x = v_x \cos(\theta) - v_y \sin(\theta) + 0v_z$
- $v'_y = v_x\, sin(\theta) + v_y \cos(\theta) + 0v_z$
- $v'_z = 0v_x + \ 0v_y + 1v_z$

- $[v'_x \quad v'_y \quad v'_z] = [v_x \quad v_y \quad v_z] . \begin{bmatrix} \cos(\theta) & \sin\theta & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- $v' = v.M$

Rotation

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

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

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

Multiple rotations

- If we have a vector v, and an x-axis rotation:

$$\mathbf{v}' = v \, . \, R_x(\theta)$$

- If we then want to rotate it around y-axis:

$$\mathbf{v}'' = \mathbf{v}' . \, R_y(\theta)$$

$$\mathbf{v}'' = \mathbf{v} . \, (M_1 M_2 M_3 M_4)$$

$$\mathbf{v}'' = \mathbf{v} . \, R_x(\theta) R_y(\theta)$$

$$\mathbf{v}'' = \mathbf{v} . \, M_{total}$$

Rotation as linear equation

$$\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{bmatrix}$$

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

$$v'_x = v_x a_x + v_y b_x + v_z c_x$$
$$v'_y = v_x a_y + v_y b_y + v_z c_y$$
$$v'_z = v_x a_z + v_y b_z + v_z c_z$$

$$\mathbf{v}' = v_x \mathbf{a} + v_y \mathbf{b} + v_z \mathbf{c}$$

Rotation and translation as linear equation

$$v'_x = v_x a_x + v_y b_x + v_z c_x + d_x$$
$$v'_y = v_x a_y + v_y b_y + v_z c_y + d_y$$
$$v'_z = v_x a_z + v_y b_z + v_z c_z + d_z$$

a, b, c, d are all constants (12 in total)

Rotation and Translation in a Matrix

- How do we represent rotation and orientation together in a matrix?

$$v'_x = v_x a_x + v_y b_x + v_z c_x + d_x$$

$$v'_y = v_x a_y + v_y b_y + v_z c_y + d_y$$

$$v'_z = v_x a_z + v_y b_z + v_z c_z + d_z$$

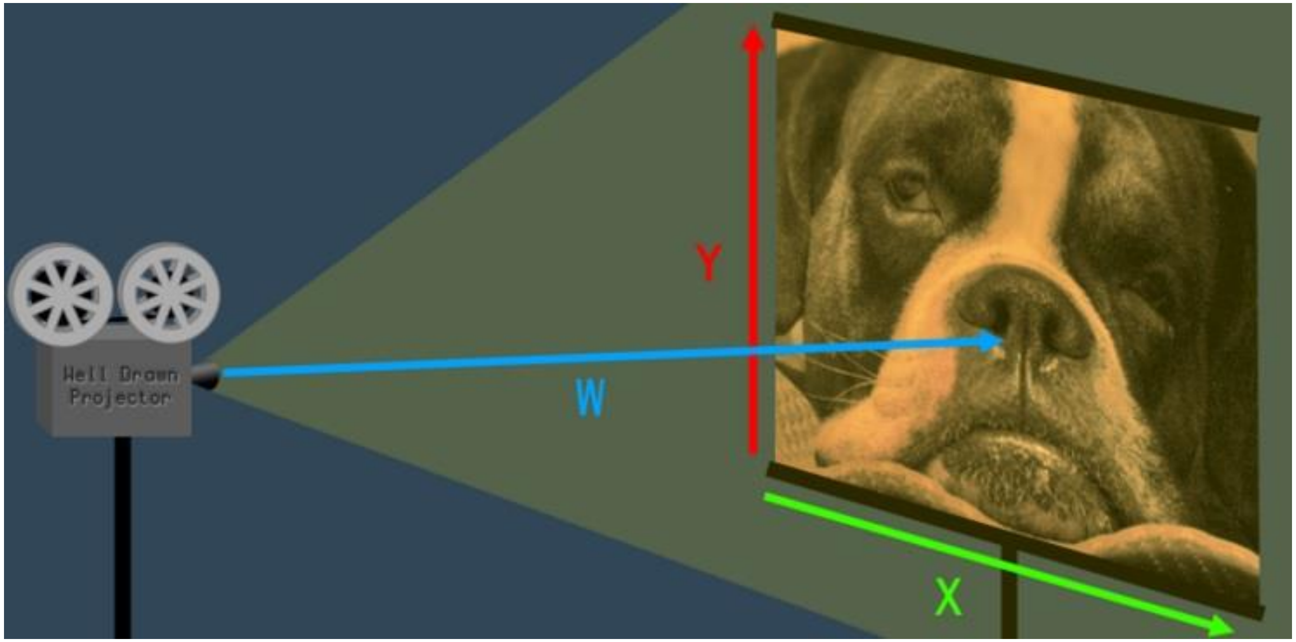
$$\begin{bmatrix} v'_x & v'_y & v'_z & 1 \end{bmatrix} = \begin{bmatrix} v_x & v_y & v_z & 1 \end{bmatrix} \begin{bmatrix} a_x & a_y & a_z & 0 \\ b_x & b_y & b_z & 0 \\ c_x & c_y & c_z & 0 \\ d_x & d_y & d_z & 1 \end{bmatrix}$$

Homogeneous Transformations

- 3x3 rotation matrix and 3x1 translation vector combined in a 4x4 matrix (with [0 0 0 1] at the right)
- 3D position vector v is changed to $[v_x\ v_y\ v_z\ 1]$
- The line at the right is not used here but it is necessary when rendering objects as a 2D image

Homogeneous Transformations

- First, let's look at how projective geometry works in 2D, before we move on to 3D.
- Imagine a projector that is projecting a 2D image onto a screen.

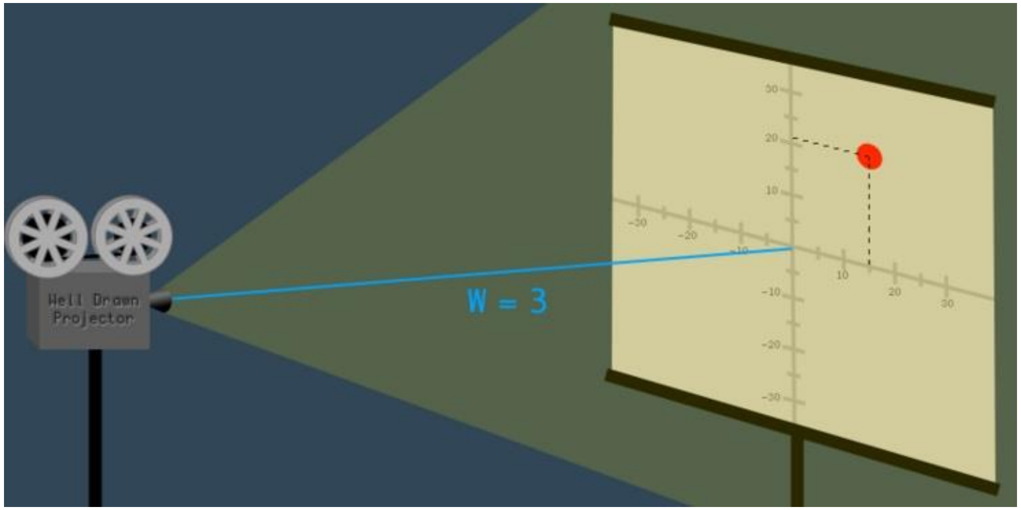


Homogeneous Coordinates

- What happens when the projector goes closer to the screen?
- What is the role of W?

Homogeneous Coordinates

- Applying it to 3D
  - When W increases the coordinate scales up and when W decreases it scales down.
- Coordinates are said to be correct in 3D, only when W = 1. (convention)
  - W < 1 everything would look too big
  - W > 1 everything would look too small
  - W = 0 division by zero error
  - W < 0 everything would flip upside down and back-to-front
- (15,21,3) => (15/3, 21/3, 3/3) => (5, 7, 1)
- 1/5 (10, 20, 30, 5) = (2, 4, 6, 1)



Perspective Transformation

- Perspective is the phenomenon where an object appears smaller the further away it is from the camera. (because it is scaled down)
- A far-away mountain can appear to be smaller than a cat, if the cat is close enough to the camera.

Matrices

- The right hand column can cause a projection, which we won't use in character animation, so we leave it as 0,0,0,1
- Some books store their matrices in a transposed form. This is fine as long as you remember that: $(A \cdot B)^T = B^T \cdot A^T$

Perspective Transformation

- Perspective in 3D graphics is implemented by using a transformation matrix that changes the W element of each vertex.
- Z represent the distance, the larger the Z is, the more it needs to be scaled down.
- W effects the scale and it is related with Z
- Perspective projection matrix applied to a homogeneous coordinate

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 4 \end{bmatrix}$$

Perspective division

- After the perspective projection matrix is applied, each vertex goes under ***perspective division***.
- Converting a homogeneous coordinate back to W = 1.
- ¼ (2,3,4,4) = (0.5, 0.75, 1, 1)
- After the perspective division, W is discarded.
  - Correct 3D coordinate that has been scaled according to a 3D perspective projection

Homogeneous Vectors

- Technically, homogeneous vectors are 4D vectors that get projected into the 3D w=1 space

$$\begin{bmatrix} v_x & v_y & v_z & v_w \end{bmatrix} \Rightarrow \begin{bmatrix} \dfrac{v_x}{v_w} & \dfrac{v_y}{v_w} & \dfrac{v_z}{v_w} \end{bmatrix}$$

Homogeneous Vectors

- Vectors representing a position in 3D space can just be written as:

$$\begin{bmatrix} v_x & v_y & v_z & 1 \end{bmatrix}$$

- Vectors representing direction are written:

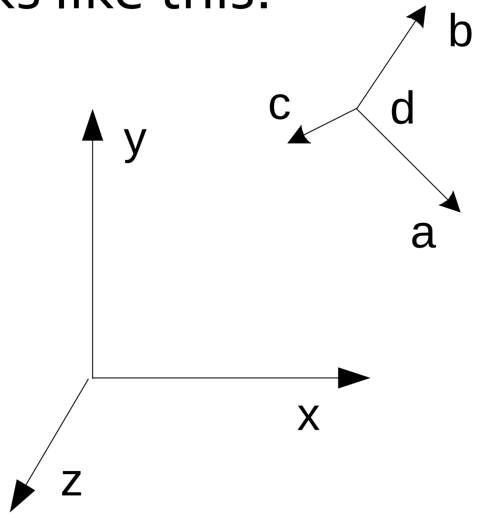
$$\begin{bmatrix} v_x & v_y & v_z & 0 \end{bmatrix}$$

- The only time the w coordinate will be something other than 0 or 1 is in the projection phase of rendering, which is not our problem

Matrices

- Computer graphics apps commonly use 4x4 homogeneous matrices
- A *rigid* 4x4 matrix transformation looks like this:

$$\mathbf{M} = \begin{bmatrix} a_x & a_y & a_z & 0 \\ b_x & b_y & b_z & 0 \\ c_x & c_y & c_z & 0 \\ d_x & d_y & d_z & 1 \end{bmatrix}$$



- Where **a**, **b**, & **c** are orthogonal unit length vectors representing orientation, and **d** is a vector representing position

Object Space

- The space that an object is defined in is called **object space or local space**
- The object is located at or near the origin and is aligned with the xyz axes
- The units in this space can be whatever we choose (i.e. meters, etc)
- A 3D object would be stored on disk and in memory in this coordinate system
- When we draw the object, we want to transform it into another space

World Space

- We will define a new space called **world space or global space**
- This space represents a 3D world or scene and may contain several objects in various locations
- Every object in the world needs a matrix that transforms its vertices from its own object space into the world space
- We call this **object's world matrix**
- For example, if we have 100 chairs in a room, we only need to store the object space data for one chair once.
- We can use 100 different matrices to transform the chair model into 100 locations in the world.

Meaning of abcd

- The 9 constants make up 3 vectors a, b and c
- If we think of the matrix as a transformation from object to world space
  - the **a** vector is essentially the object's x-axis rotated in world space
  - **b** is its y-axis in world space
  - and **c** is its z-axis in world space.
- d is the position in world space

Identity

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Take one more look at the identity matrix
- Its **a** axis lines up with x, **b** lines up with y, and **c** lines up with z
- Position **d** is at the origin
- Therefore, it represents a transformation with no rotation or translation

Rotation

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

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

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

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

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

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

Rotation and translation

- For example, a translation by vector r followed by a z-axis rotation is:

$$\begin{bmatrix} v_x^{'} \\ v_y^{'} \\ v_z^{'} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & r_x \\ 0 & 1 & 0 & r_y \\ 0 & 0 & 1 & r_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \\ 1 \end{bmatrix}$$

Rigid Matrices

- If the upper 3x3 portion is orthonormal, we say that 4x4 matrix is **rigid**
  - only translated and rotated (it will not have any scale or shears which distort the object)

Orthonormality

- If all row vectors and all column vectors of a matrix are unit length, that matrix is said to be orthonormal

- This also implies that all vectors are perpendicular to each other

- Orthonormal matrices have some useful mathematical properties, such as:
  - $M^{-1} = M^T$

Orthonormality

- If a 4x4 matrix represents a rigid transformation, then the upper 3x3 portion will be orthonormal

$$|\mathbf{a}| = |\mathbf{b}| = |\mathbf{c}| = 1$$

$$\mathbf{a} = \mathbf{b} \times \mathbf{c}$$

$$\mathbf{b} = \mathbf{c} \times \mathbf{a}$$

$$\mathbf{c} = \mathbf{a} \times \mathbf{b}$$

Determinants

- The determinant is a scalar value that represents the volume change that the transformation will cause
- An orthonormal matrix will have a determinant of 1, but non-orthonormal volume preserving matrices will have a determinant of 1 also
- A degenerate matrix has a determinant of 0
- A matrix that has been mirrored will have a negative determinant

Position Vector Dot Matrix

$$\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z & 1 \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} a_x & a_y & a_z & 0 \\ b_x & b_y & b_z & 0 \\ c_x & c_y & c_z & 0 \\ d_x & d_y & d_z & 1 \end{bmatrix}$$

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

$$v'_x = v_x a_x + v_y b_x + v_z c_x + d_x$$

$$v'_y = v_x a_y + v_y b_y + v_z c_y + d_y$$

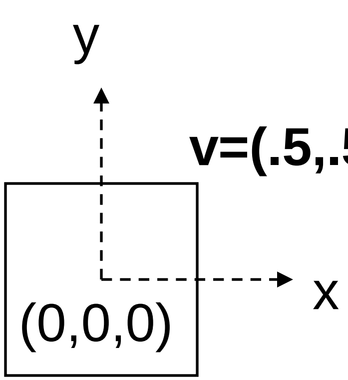
$$v'_z = v_x a_z + v_y b_z + v_z c_z + d_z$$

$$v'_w = 1$$

$$\mathbf{v}' = v_x \mathbf{a} + v_y \mathbf{b} + v_z \mathbf{c} + \mathbf{d}$$

Position Vector Dot Matrix

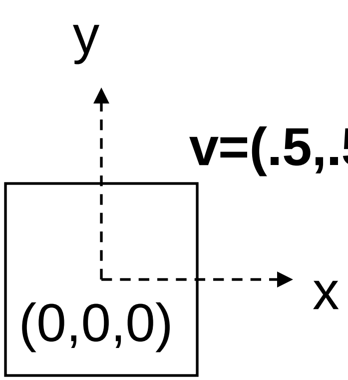
$$\mathbf{v}' = v_x\mathbf{a} + v_y\mathbf{b} + v_z\mathbf{c} + \mathbf{d}$$



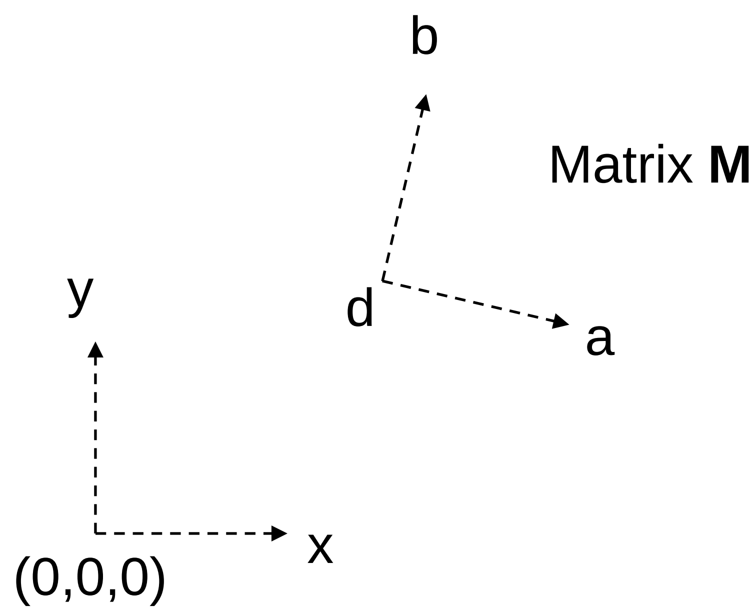
Local Space

Position Vector Dot Matrix

$$\mathbf{v}' = v_x\mathbf{a} + v_y\mathbf{b} + v_z\mathbf{c} + \mathbf{d}$$



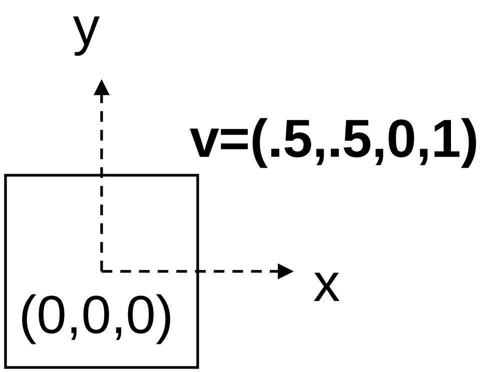
Local Space



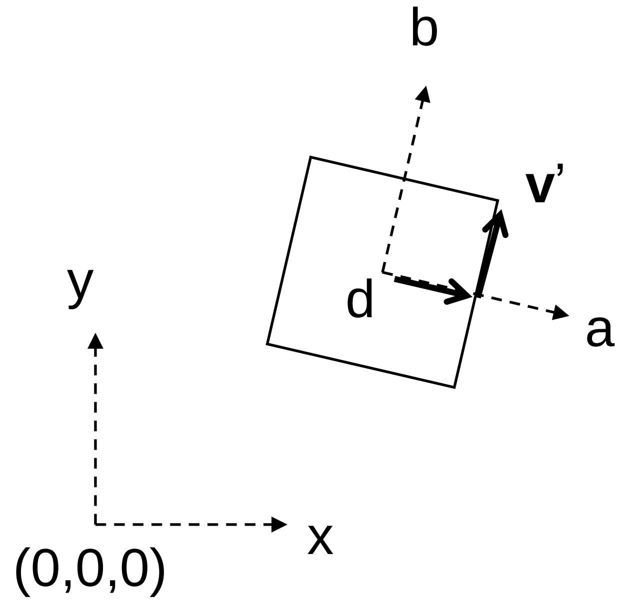
World Space

Position Vector Dot Matrix

$$\mathbf{v}' = v_x\mathbf{a} + v_y\mathbf{b} + v_z\mathbf{c} + \mathbf{d}$$



Local Space



World Space

Direction Vector Dot Matrix

$$\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z & 0 \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} a_x & a_y & a_z & 0 \\ b_x & b_y & b_z & 0 \\ c_x & c_y & c_z & 0 \\ d_x & d_y & d_z & 1 \end{bmatrix}$$

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

$$v'_x = v_x a_x + v_y b_x + v_z c_x$$

$$v'_y = v_x a_y + v_y b_y + v_z c_y$$

$$v'_z = v_x a_z + v_y b_z + v_z c_z$$

$$v'_w = 0$$

$$\mathbf{v}' = v_x \mathbf{a} + v_y \mathbf{b} + v_z \mathbf{c}$$

Matrix Dot Matrix (4x4)

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

$$\mathbf{M} = \begin{bmatrix} a_x & a_y & a_z & 0 \\ b_x & b_y & b_z & 0 \\ c_x & c_y & c_z & 0 \\ d_x & d_y & d_z & 1 \end{bmatrix}$$

- The row vectors of **M′** are the row vectors of **M** transformed by matrix **N**
- Notice that **a**, **b**, and **c** transform as direction vectors and **d** transforms as a position

Matrix Dot Matrix

$$\mathbf{L} = \mathbf{M} \cdot \mathbf{N}$$

$$\begin{bmatrix} l_{11} & l_{12} & l_{13} \\ l_{21} & l_{22} & l_{23} \\ l_{31} & l_{32} & l_{33} \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \cdot \begin{bmatrix} n_{11} & n_{12} & n_{13} \\ n_{21} & n_{22} & n_{23} \\ n_{31} & n_{32} & n_{33} \end{bmatrix}$$

$$l_{12} = m_{11}n_{12} + m_{12}n_{22} + m_{13}n_{32}$$

$$\mathbf{a}_L = \mathbf{a}_M \cdot \mathbf{N}$$

$$\mathbf{b}_L = \mathbf{b}_M \cdot \mathbf{N}$$

$$\mathbf{c}_L = \mathbf{c}_M \cdot \mathbf{N}$$

Supplementary Material and References

- Computer Animation, Rick Parent, Chapter 2 and Appendix
- Khan Academy online courses
  - Linear Algebra and Calculus
  - https://www.khanacademy.org/
- Some of the slides of this lecture are based on the Computer Animation course at the University of California San Diego.