Linear Algebra for Graphics

Lecturer: Carol O'Sullivan

carol.osullivan@tcd.ie

Overview

- Vector addition, subtraction, multiplication
- Normalising vectors
- Dot Product
- Cross Product & Polygon normals
- Changing Basis

Extra Reading

- **Chapter 3: Geometric Objects and Transformations**
- Interactive Computer Graphics: A Top Down Approach with OpenGL, 6^{th} Edition (or other) Angel
- **Chapter 4: Math for 3D Graphics**
- OpenGL Superbible, 6^{th} Edition
- **Elementary Linear Algebra**, Anton

Linear Algebra

- Linear algebra is the cornerstone of computer graphics.
- Fundamentally, we need to be able to manipulate *points* and *vectors*.
  - these form the basis of all geometric objects & operations
- Geometric operations (*scale*, *rotate*, *translate*, *perspective projection*) are defined using matrix transformations.
- Optical effects (*reflect*, *refract*) defined using vector algebra.

Conventions

- Vector quantities denoted as **v** or $\vec{v}$
- Each vector is defined with respect to a set of *basis vectors* (which define a co-ordinate system).
- We will use *column format* vectors:

$$\mathbf{v} = \boxed{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}} \neq \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \quad \left(= \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T \right)$$

Row vs. Column Formats

- Both formats, though appearing equivalent, are in fact fundamentally different:
  - be wary of different formats used in textbooks

column format

row format

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} u & v & w \end{bmatrix} \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$$

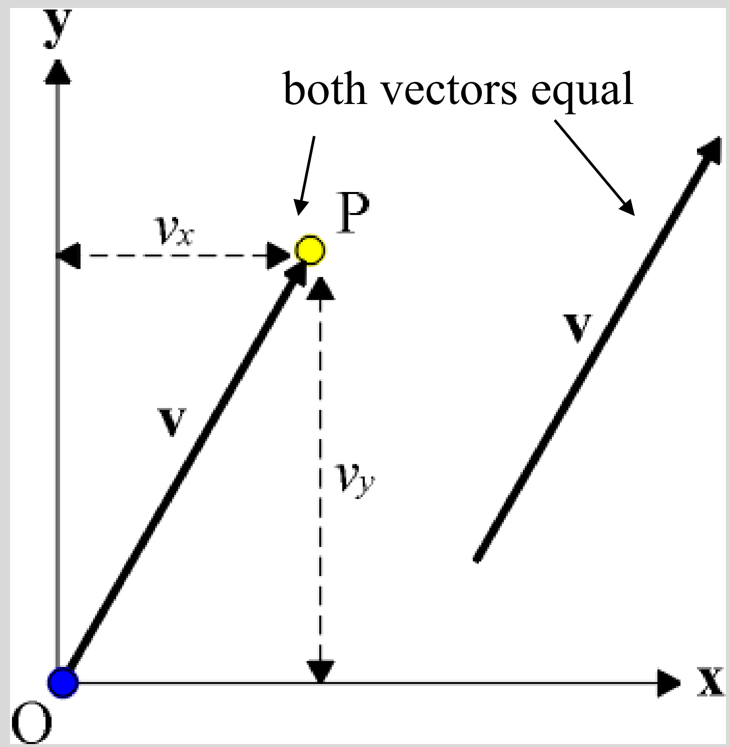
$$\mathbf{M}\mathbf{v} = \mathbf{v}^T \mathbf{M}^T$$

transposed

Vectors & Points

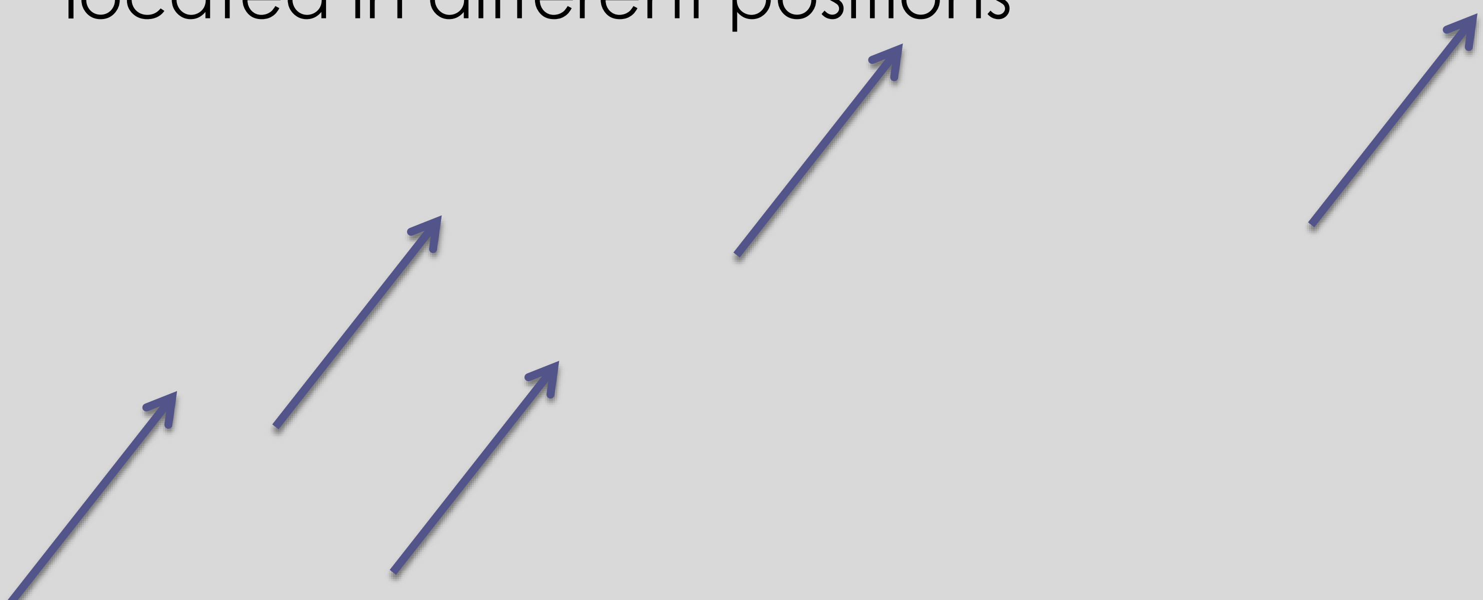
- Although *vectors* and *points* are often used inter-changeably in graphics texts, it is important to distinguish between them.
  - vectors represent directions
  - points represent positions
- Both are meaningless without reference to a *coordinate system*
  - vectors require a set of *basis vectors*
  - points require an *origin* and a *vector space*

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} \quad \mathrm{P} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$



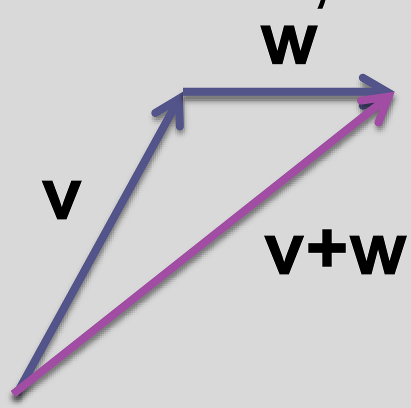
Equivalent Vectors

- Vectors with the same length and same direction are called equivalent. Since we want a vector to be determined solely by its length and direction, equivalent vectors are regarded as equal, even if located in different positions



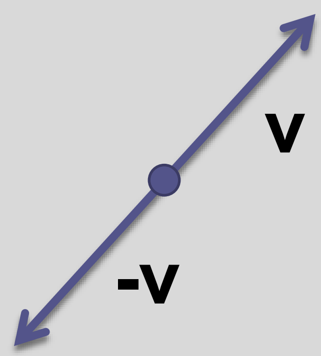
Vector Addition

- If **v** and **w** are any two vectors then their sum is the vector determined as follows:
  - Position the vector **w** so that its initial point coincides with the terminal point of **v**
  - The vector **v+w** is represented by the arrow from **v** to **w** (head-to-tail rule)



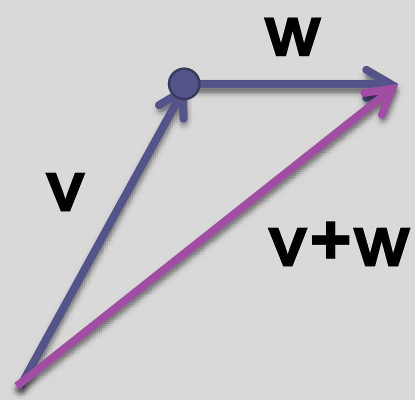
Negative Vectors

- If v is any nonzero vector, then **–v**, the negative of v, is defined to be the vector having the same magnitude as **v**, but oppositely directed



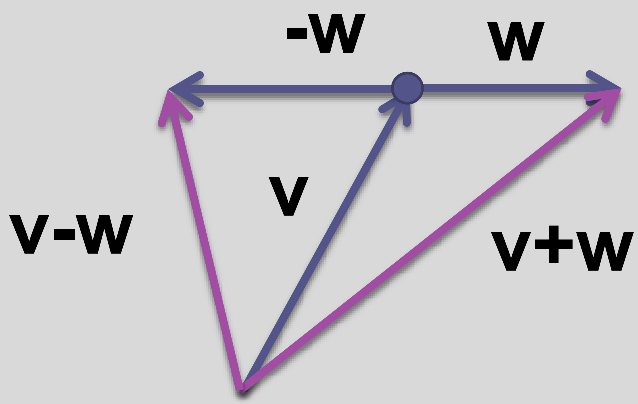
Vector Subtraction

- If **v** and **w** are any two vectors, then difference of w from v is defined by:
  - **v** - **w** = **v** + (-**w**)



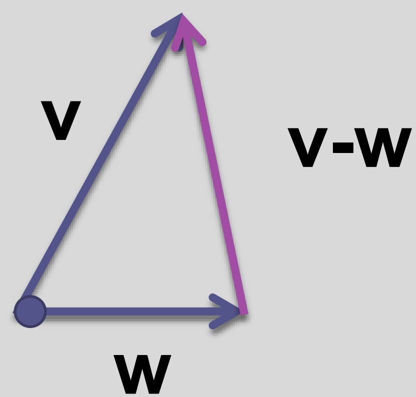
Vector Subtraction

- If **v** and **w** are any two vectors, then difference of w from v is defined by:
  - **v** - **w** = **v** + (-**w**)



Vector Subtraction

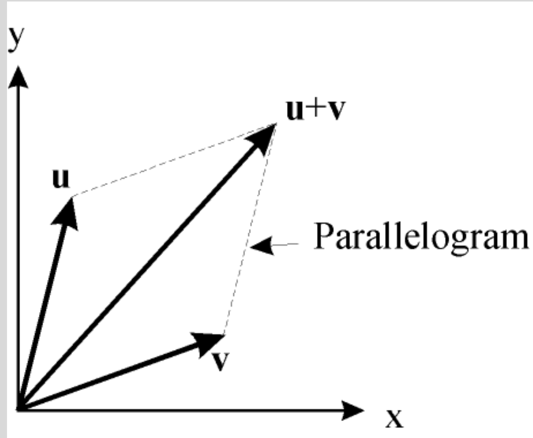
- Position v and w so their initial points coincide
  - The vector from the terminal point of **w** to the terminal point of **v** is then **v-w**

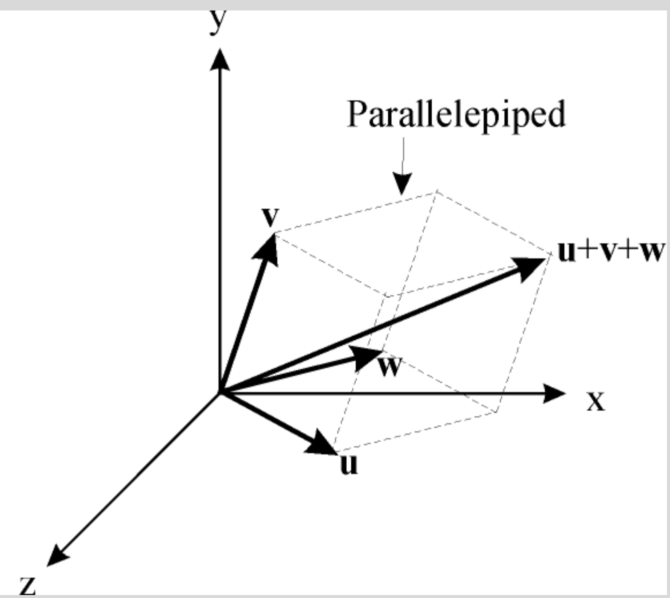


Vector Addition & Subtraction

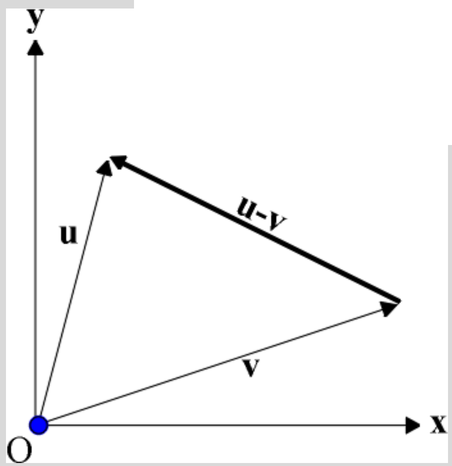
- Addition of vectors follows the *parallelogram* law in 2D and the *parallelepiped* law in higher dimensions:

$$\underset{\mathbf{u}}{\begin{bmatrix}1\\2\\3\end{bmatrix}}+\underset{\mathbf{v}}{\begin{bmatrix}4\\5\\6\end{bmatrix}}=\underset{\mathbf{u+v}}{\begin{bmatrix}5\\7\\9\end{bmatrix}}$$



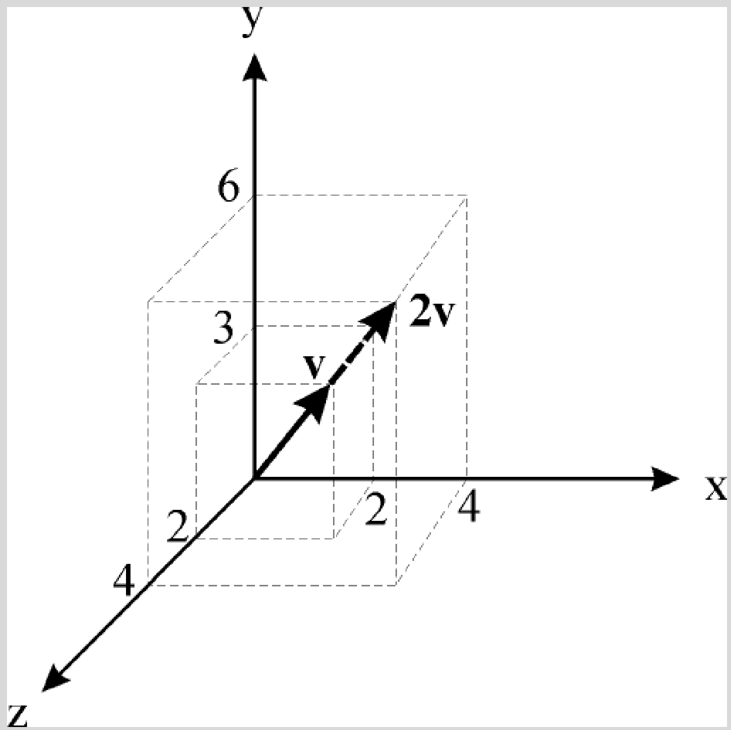


- Subtraction:



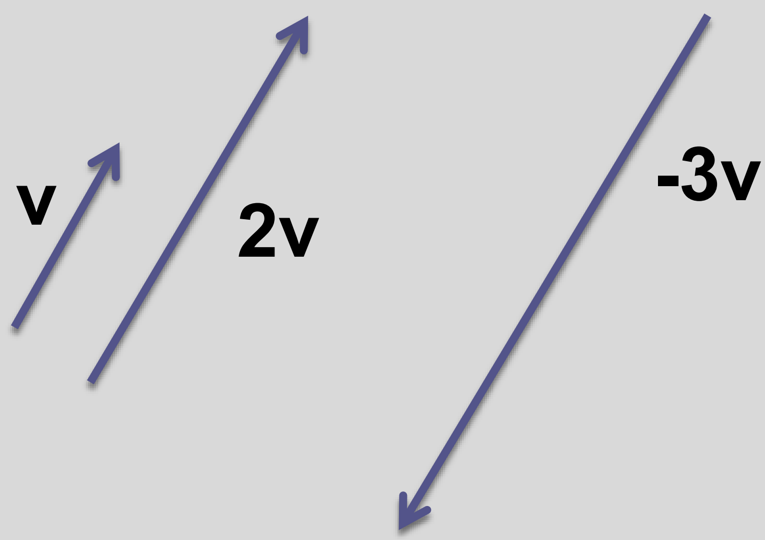
Vector Multiplication by a Scalar

- Each vector has an associated length
- Multiplication by a scalar scales the vectors length appropriately (but does not affect direction):



Vector Multiplication by a Scalar

- Vectors that are scalar multiples of each other are parallel



Exercise

- If $v = \begin{bmatrix} 1 \\ -3 \\ 2 \end{bmatrix}$ and $w = \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix}$ find:
  - v+w =
  - 2v =
  - -w =
  - v-w =

Answer

- If $v = \begin{bmatrix} 1 \\ -3 \\ 2 \end{bmatrix}$ and $w = \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix}$ find:
  - v+w = $\begin{bmatrix} 5 \\ -1 \\ 3 \end{bmatrix}$
  - 2v = $\begin{bmatrix} 2 \\ -6 \\ 4 \end{bmatrix}$
  - -w = $\begin{bmatrix} -4 \\ -2 \\ -1 \end{bmatrix}$
  - v-w = $\begin{bmatrix} -3 \\ -5 \\ 1 \end{bmatrix}$

Linear Combinations

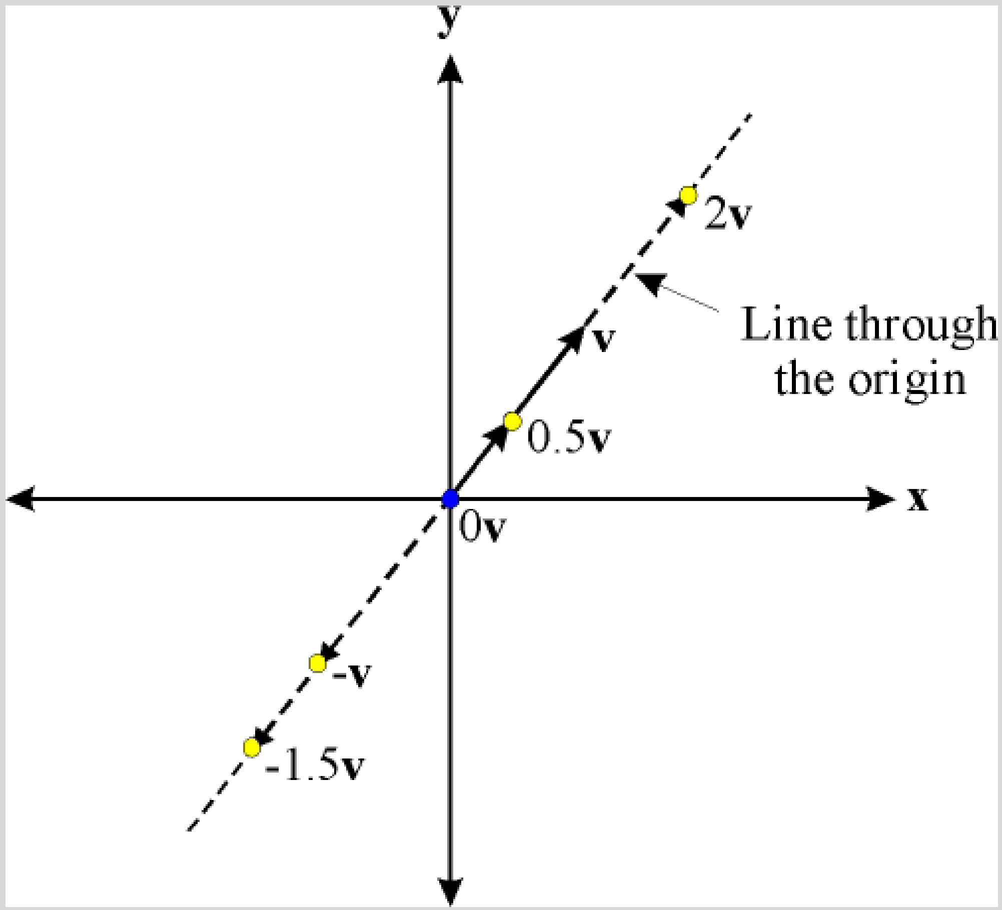
- The *linear combination* of a set of vectors is the sum of scalar multiples of those vectors:

$$\mathbf{u} = a_1\mathbf{v_1} + a_2\mathbf{v_2} + \cdots + a_n\mathbf{v_n}$$

- Fixing vectors $\mathbf{v_i}$ yields an infinite number of $\mathbf{u}$ depending on the scalars $\boldsymbol{a_i}$.
- The set $\mathbf{u}$ is called the *span* of the vectors $\mathbf{v_i}$
- The vectors $\mathbf{v_i}$ are termed *basis vectors* for the space.
- If none of the $\mathbf{v_i}$ can be created as a linear combination of the others, the vectors $\mathbf{v_i}$ are said to be *linearly independent*.
- All linear combinations contain the *zero vector*.

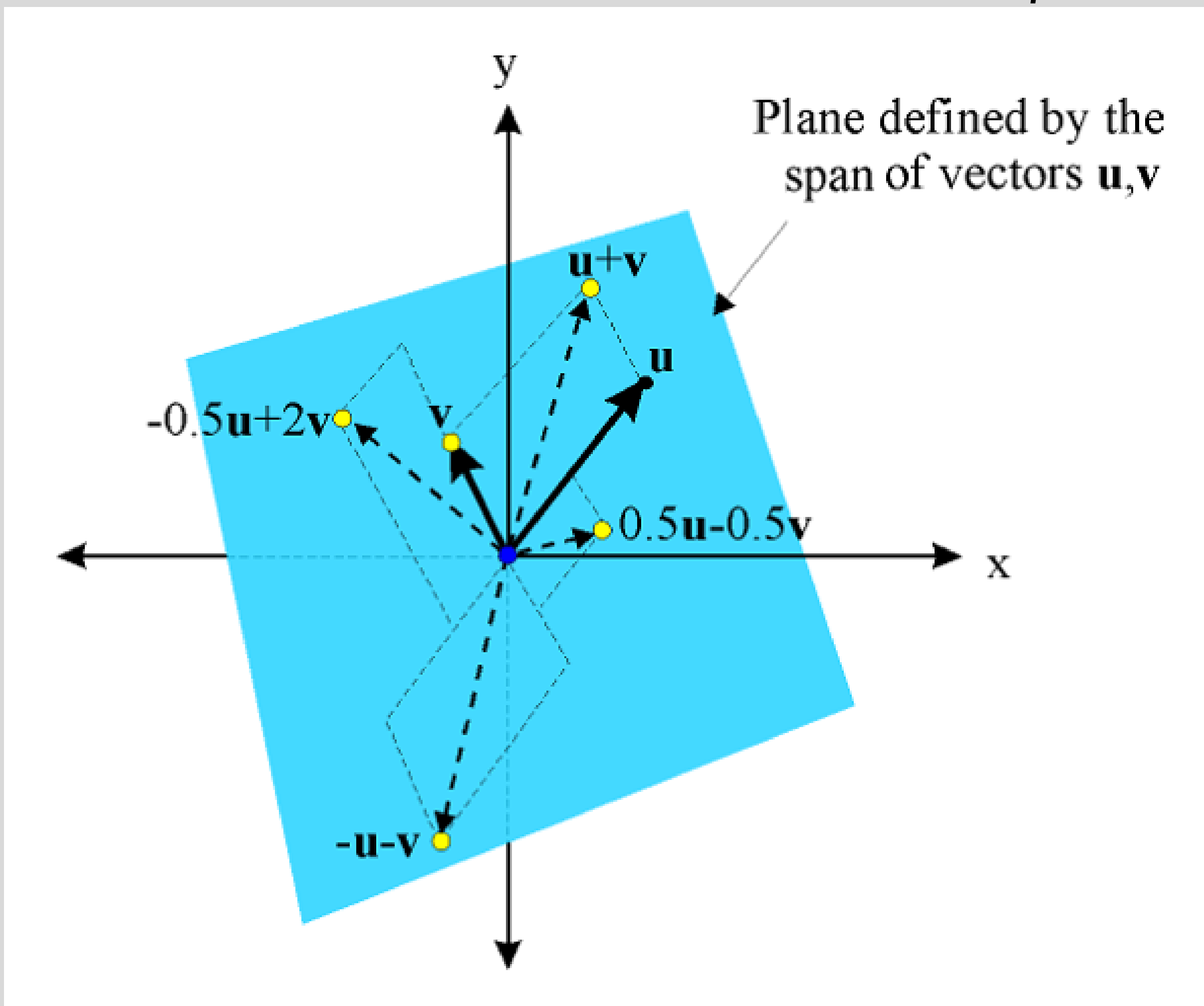
Linear Combinations

- Linear combinations of 1 vector = an *infinite line*:



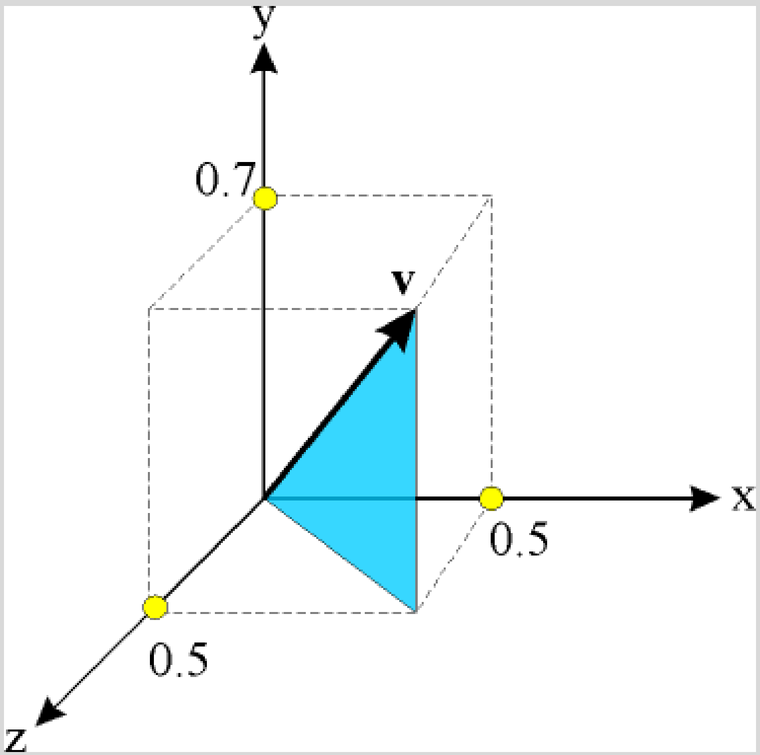
Linear Combinations

- Linear combinations of 2 vectors = a *plane*



Linear Combinations

- The linear combination of 3 vectors = a *3D volume*.
- The 3D Cartesian coordinate system employs the well-known 3D co-ordinate basis: **x**, **y** and **z**



$$\mathbf{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{z} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The vector **v** here is a *linear combination* of the basis vectors **x**, **y** and **z**:

$$\mathbf{v} = \begin{bmatrix} 0.5 \\ 0.7 \\ 0.5 \end{bmatrix} = 0.5 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + 0.7 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + 0.5 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Vector Magnitude

- The *magnitude* or *norm* of a vector of dimension *n* is given by the standard *Euclidean distance metric*:

- For example:

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

$$\left\| \begin{matrix} 1 \\ 3 \\ 1 \end{matrix} \right\| = \sqrt{1^2 + 3^2 + 1^2} = \sqrt{11}$$

- Vectors of length 1 (unit vectors) are often termed *normal or normalised vectors*.

Normalised Vectors

- When we wish to describe direction we use *normalised* vectors.
- We normalise a vector by dividing by its magnitude:

$$\mathbf{v}' = \frac{\mathbf{v}}{\|\mathbf{v}\|} = \frac{1}{\sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}}\,\mathbf{v}$$

Exercise

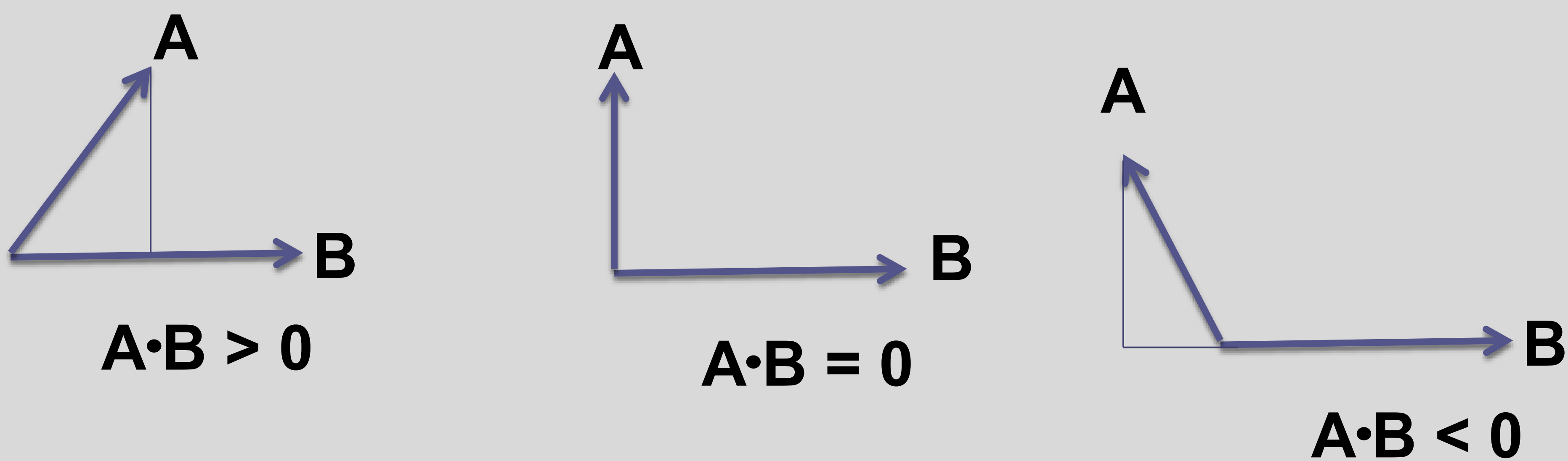
- Let **u** = (2,-2,3), **v**=(1,-3,4), **w**=(3,6,-4)
  - $\|\mathbf{u} + \mathbf{v}\| =$
  - $\|\mathbf{u}\| + \|\mathbf{v}\| =$
  - $\|-2\mathbf{u}\| + 2\|\mathbf{u}\| =$

Answer

- Let **u** = (2,-2,3), **v**=(1,-3,4), **w**=(3,6,-4)
  - $\|\mathrm{u}+\mathrm{v}\| = \sqrt{83}$
  - $\|\mathrm{u}\| + \|\mathrm{v}\| = \sqrt{17} + \sqrt{26}$
  - $\|-2\mathrm{u}\| + 2\|\mathrm{u}\| = 4\sqrt{17}$

Dot Product

- A dot product of two vectors gives a *scalar*. It calculates angles.
- *The length of the projection of **A** onto **B***



Dot Product

- Dot product (*inner product*) is defined as:

$$\mathbf{u} \cdot \mathbf{v} = \sum_i u_i v_i$$

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T \mathbf{v} = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = u_1 v_1 + u_2 v_2 + u_3 v_3$$

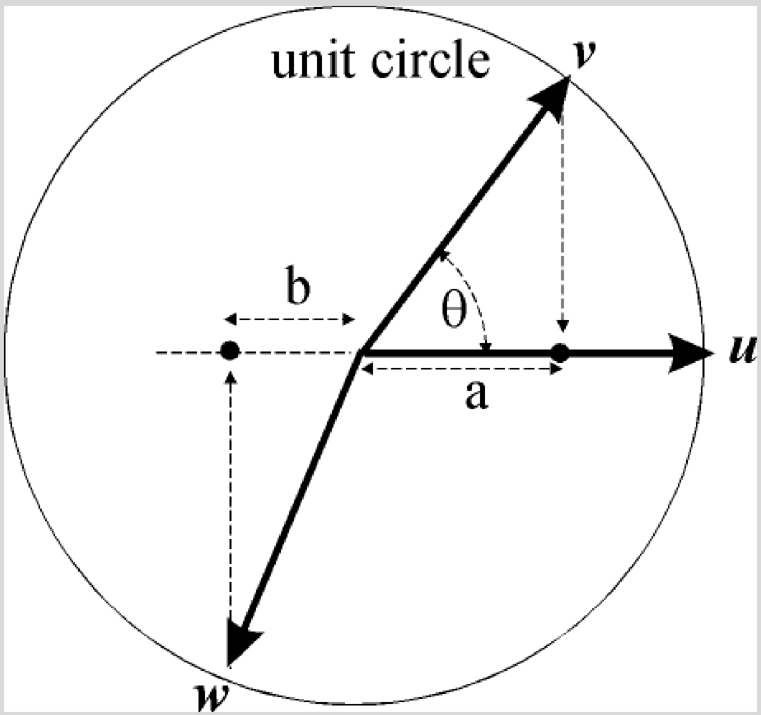
- <u>Note</u>: $\mathbf{u} \cdot \mathbf{u} = u_1^2 + u_2^2 + u_3^2 = \|\mathbf{u}\|^2$

- Therefore we can also define magnitude in terms of the dot-product operator: $\|\mathbf{u}\| = \sqrt{\mathbf{u} \cdot \mathbf{u}}$

- Dot product operator is *commutative*.

Dot Product

- If both vectors are normalised, the dot product defines the cosine of the angle between the vectors:



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

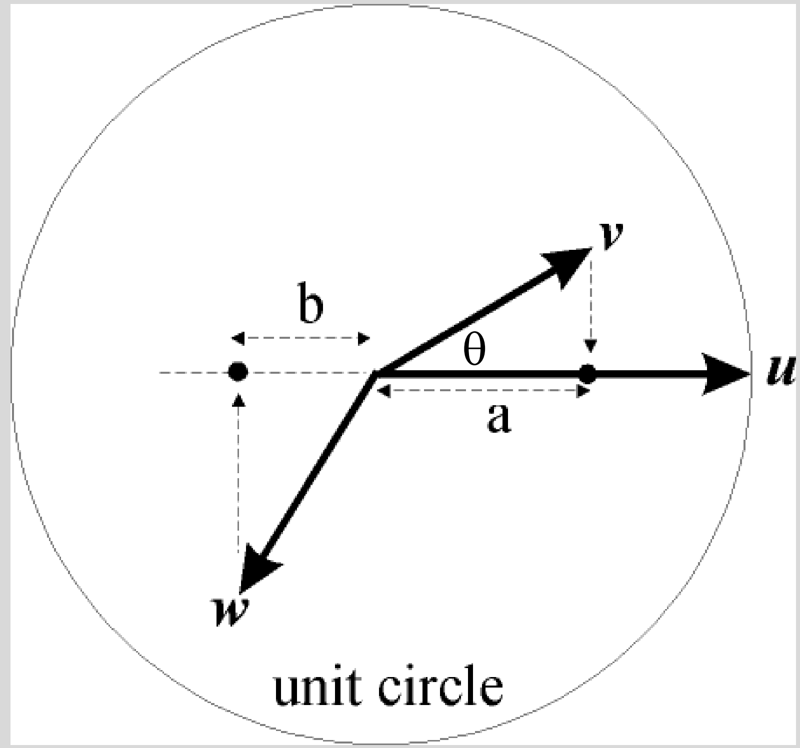
In general:

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

$$\Rightarrow \theta = \cos^{-1}\left[\frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}\right]$$

Dot Product

- If one of the vectors is normalised, the dot product defines the *projection* of the other onto it (perpendicularly)
- In this example, *a* is positive and *b* is negative.
- Note that if both vectors are pointing in *same direction*, the dot-product is positive.



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

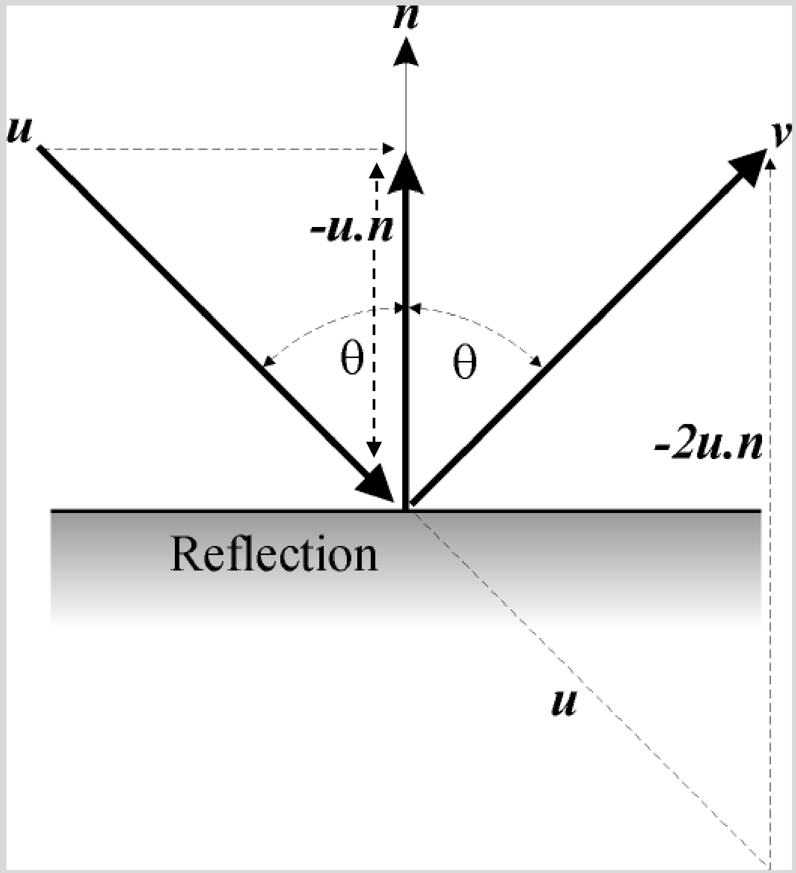
$$\Rightarrow a = \|\mathbf{v}\|\cos\theta$$

$$\therefore \cos\theta = \frac{a}{\|\mathbf{v}\|}$$

$$a = \mathbf{u} \cdot \mathbf{v} \qquad b = \mathbf{u} \cdot \mathbf{w}$$

Dot Product

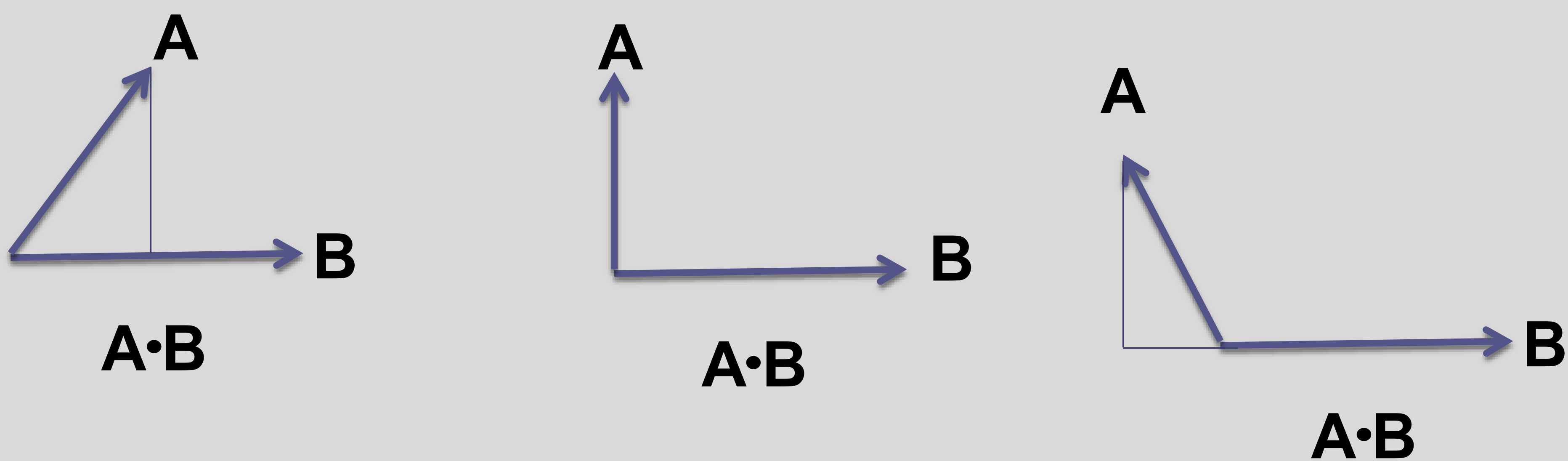
- Note that if $\theta = 90$ then the dot product = 0, i.e. the projection of one onto the other has zero length $\Rightarrow$ vectors are *orthogonal*.
- Also, if $\theta > 90$ then the dot product is negative.
- Example:



$$\mathbf{v} = \mathbf{u} - 2\mathbf{n}(\mathbf{u} \cdot \mathbf{n})$$

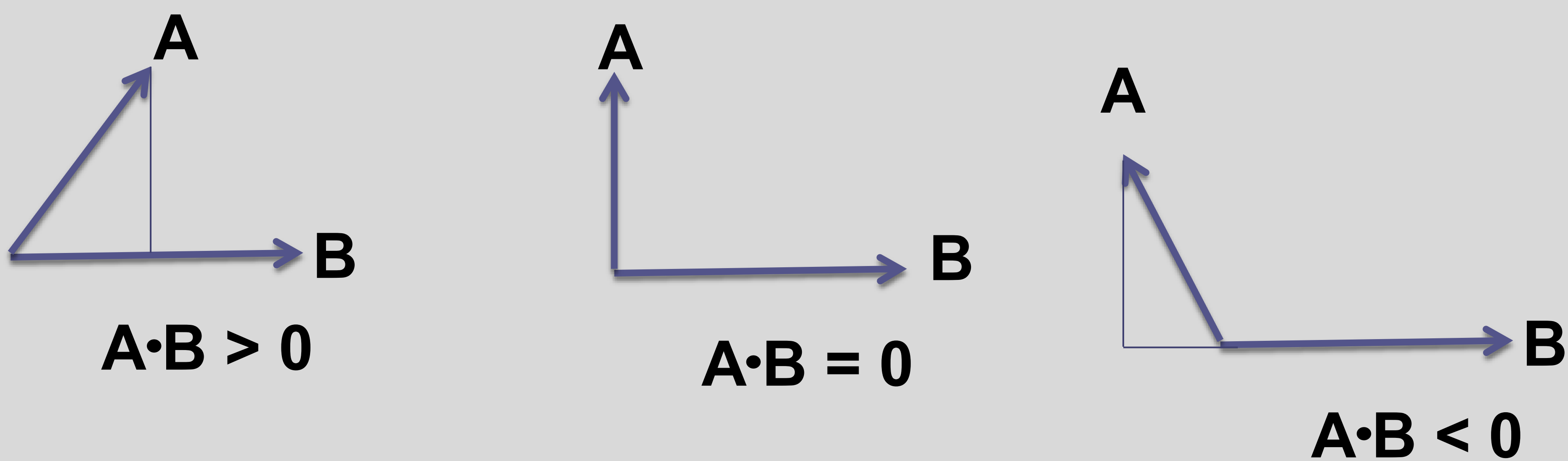
Dot Product

- A dot product of two vectors gives a *scalar*. It calculates angles.
- *The length of the projection of **A** onto **B***



Dot Product

- A dot product of two vectors gives a *scalar*. It calculates angles.
- *The length of the projection of* ***A*** *onto* ***B***



Exercise

- Consider the vectors
  - u = (2,-1,1) and v=(1,1,2)
  - Find u dot v and determine the angle between them

Exercise

- Consider the vectors
  - u = (2,-1,1) and v=(1,1,2)
  - Find u dot v and determine the angle between them
- u dot v = u1v1 + u2v2 + u3v3 =3
- Angle between = 60
  - Arccos (u dot v over magnitude of u by magnitude of v)

Cross Product

- The cross product of two vectors gives a *vector*. It calculates direction.
- Graphically, the cross product returns a vector that is orthogonal to the plane formed by the two input vectors.
- A x B is not equal to B x A

Cross Product

- Used for defining *orientation* and constructing *co-ordinate axes*.
- Cross product defined as:

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

- The result is a *vector (w)*, perpendicular to the plane defined by **u** and **v:**

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

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

Cross Product Example

- Find **u** x **v** where **u** = (1,2,-2) and **v** = (3,0,1)

$$\mathbf{u}\times\mathbf{v}=\begin{bmatrix}u_1\\u_2\\u_3\end{bmatrix}\times\begin{bmatrix}v_1\\v_2\\v_3\end{bmatrix}=\begin{bmatrix}u_2v_3-u_3v_2\\u_3v_1-u_1v_3\\u_1v_2-u_2v_1\end{bmatrix}$$

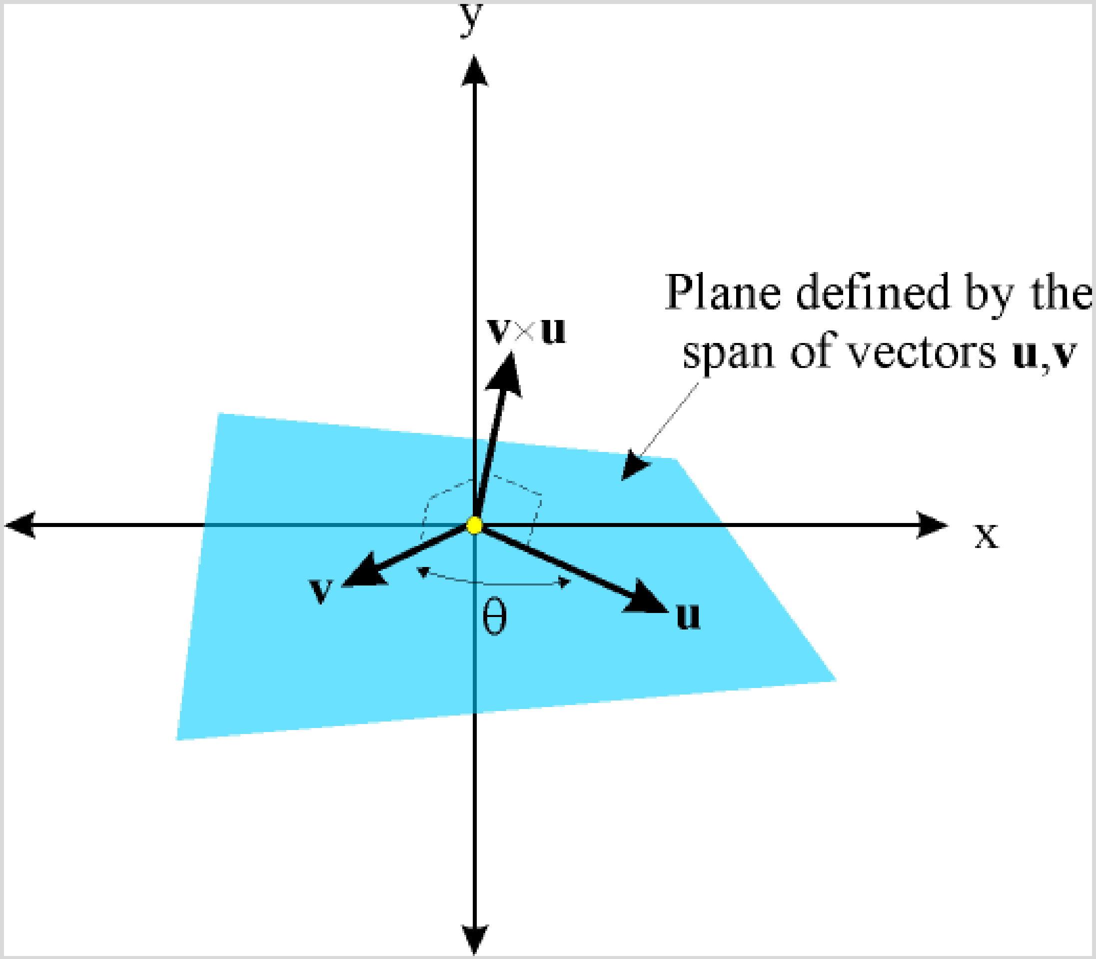
$$\mathbf{u}\times\mathbf{v}=\begin{bmatrix}1\\2\\-2\end{bmatrix}\times\begin{bmatrix}3\\0\\1\end{bmatrix}=\begin{bmatrix}2-0\\-6-1\\0-6\end{bmatrix}$$

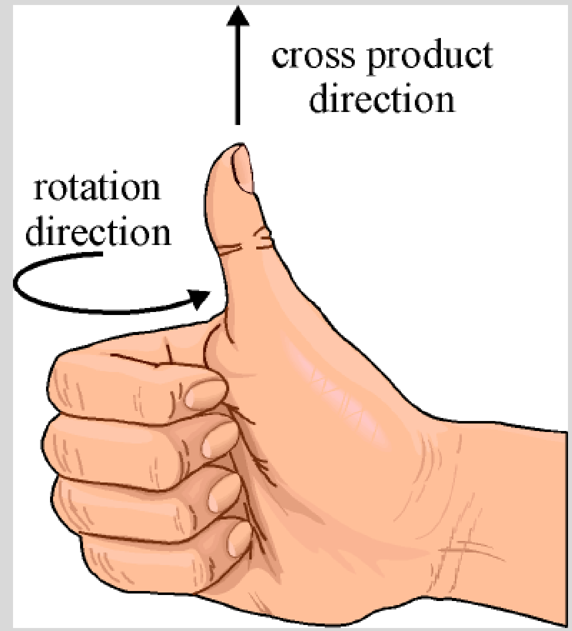
Cross Product Example

- Find **u** x **v** where **u** = (1,2,-2) and **v** = (3,0,1)

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

Cross Product





Right Handed Coordinate System

Cross Product

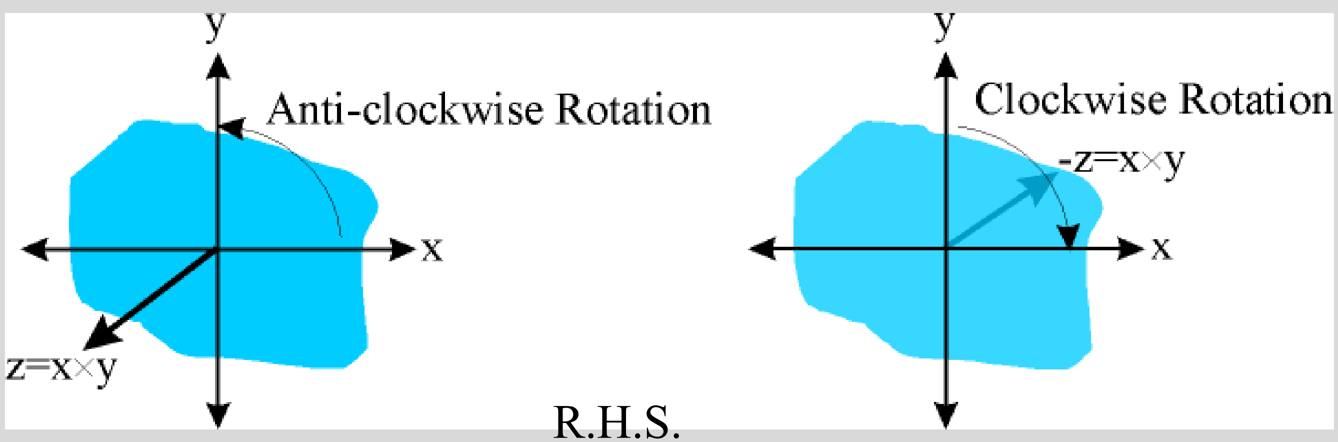
- Cross product is *anti-commutative*:

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

- It is **not** *associative*:

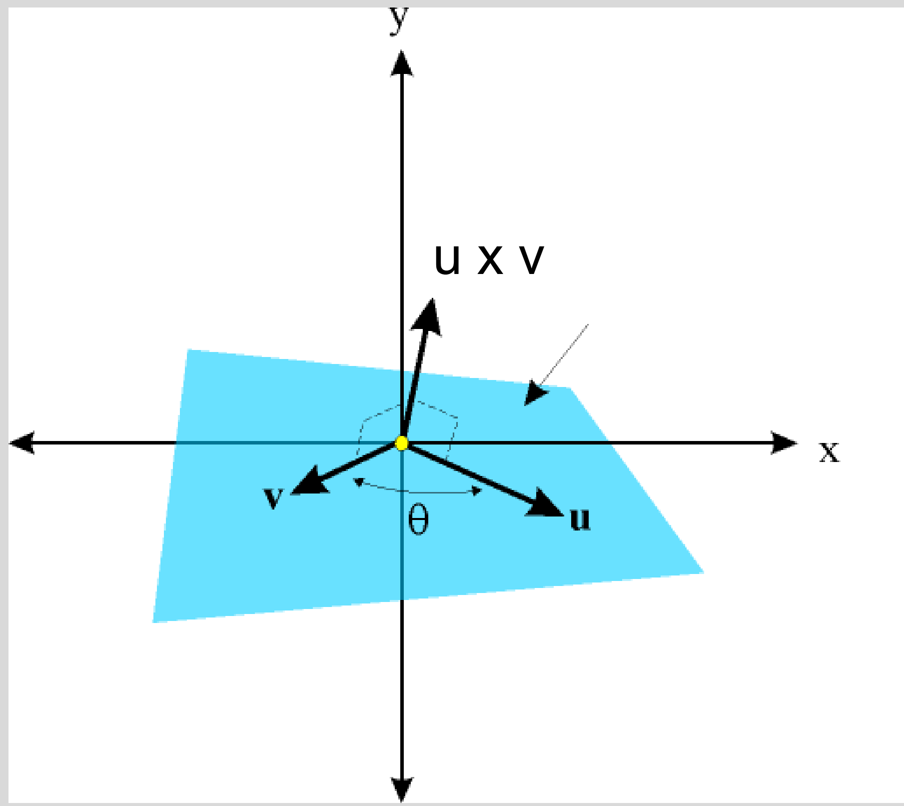
$$\mathbf{u} \times (\mathbf{v} \times \mathbf{w}) \neq (\mathbf{u} \times \mathbf{v}) \times \mathbf{w}$$

- Direction of resulting vector defined by operand order:



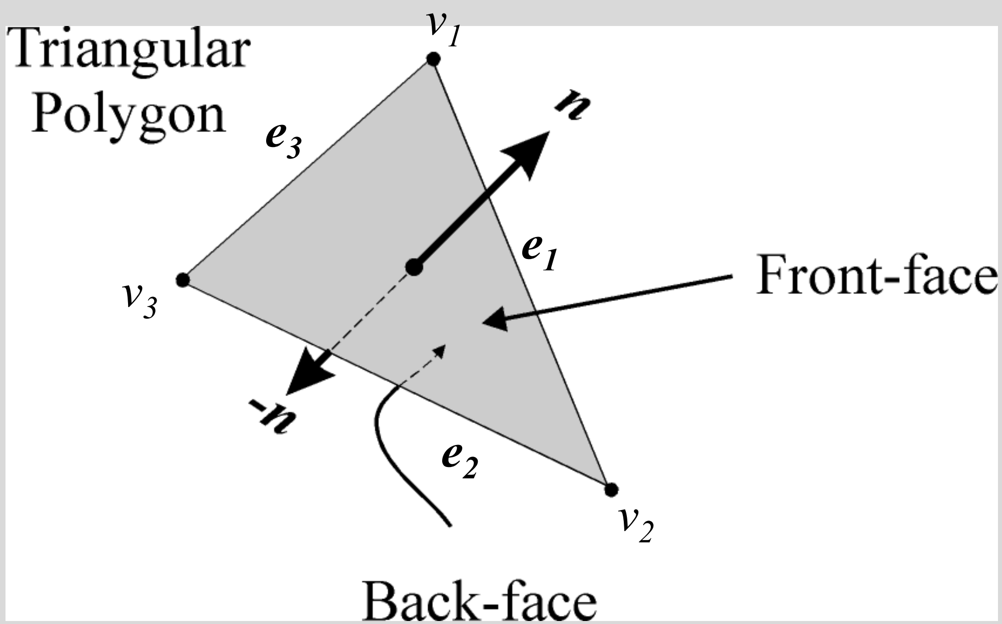
Exercise

- LHS
- is u x v correct in the diagram?



Normals & Polygons

- Polygons are (usually) planar regions bounded by *n* edges connecting *n* points or *vertices*.
- For lighting and viewing calculations we need to define the normal to a polygon:



- The normal distinguishes the *front-face* from the *back-face* of the polygon.

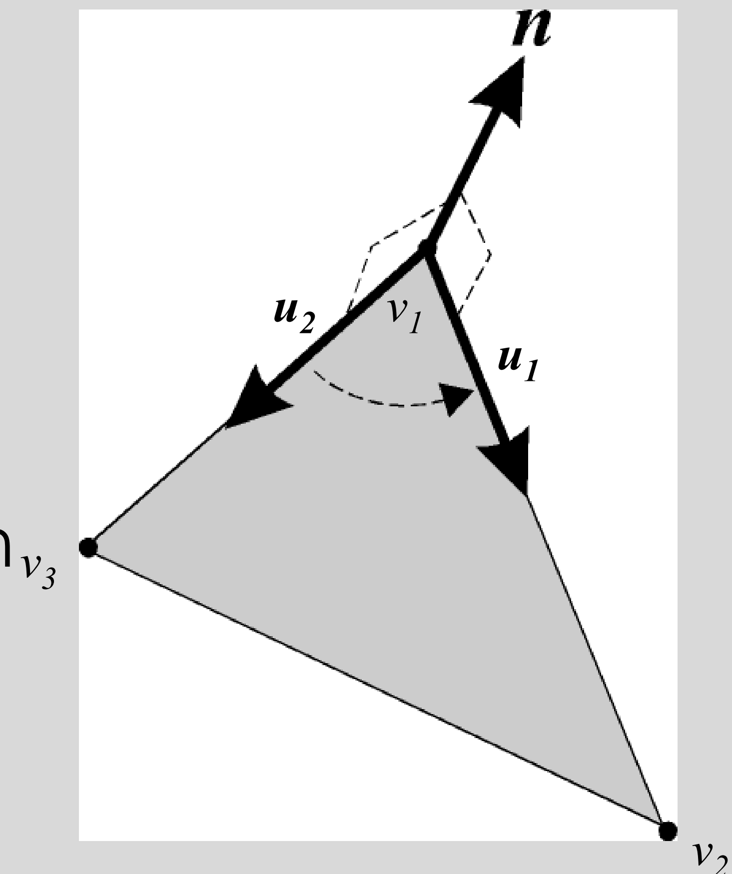
Normals & Polygons

- First determine the 2 *edge vectors* from the vertices:

$$\mathbf{u}_1 = \frac{v_2 - v_1}{\|v_2 - v_1\|} \qquad \mathbf{u}_2 = \frac{v_3 - v_1}{\|v_3 - v_1\|}$$

- The polygon normal is given by:

$$\mathbf{n} = \frac{\mathbf{u}_2 \times \mathbf{u}_1}{\|\mathbf{u}_2 \times \mathbf{u}_1\|}$$

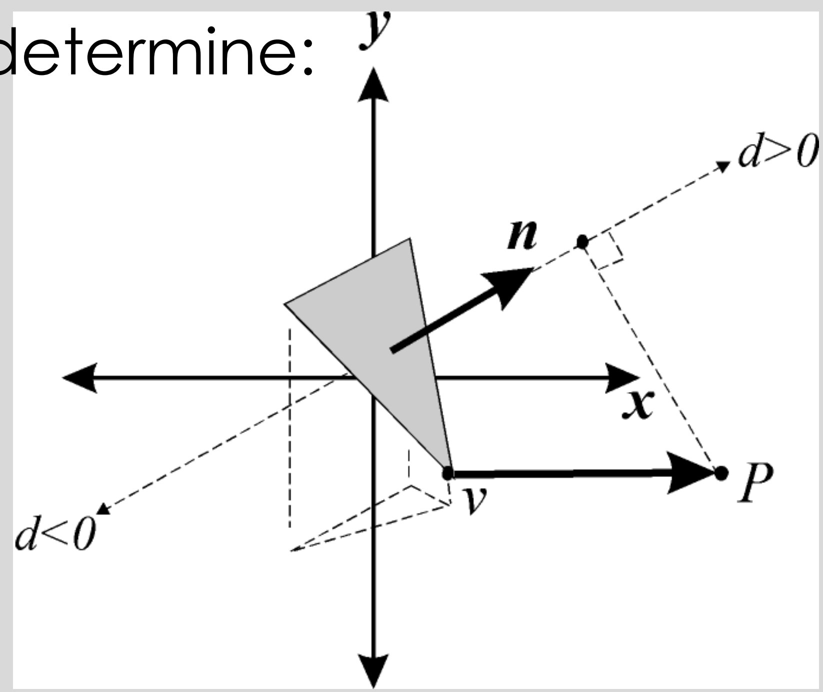


Normals & Polygons

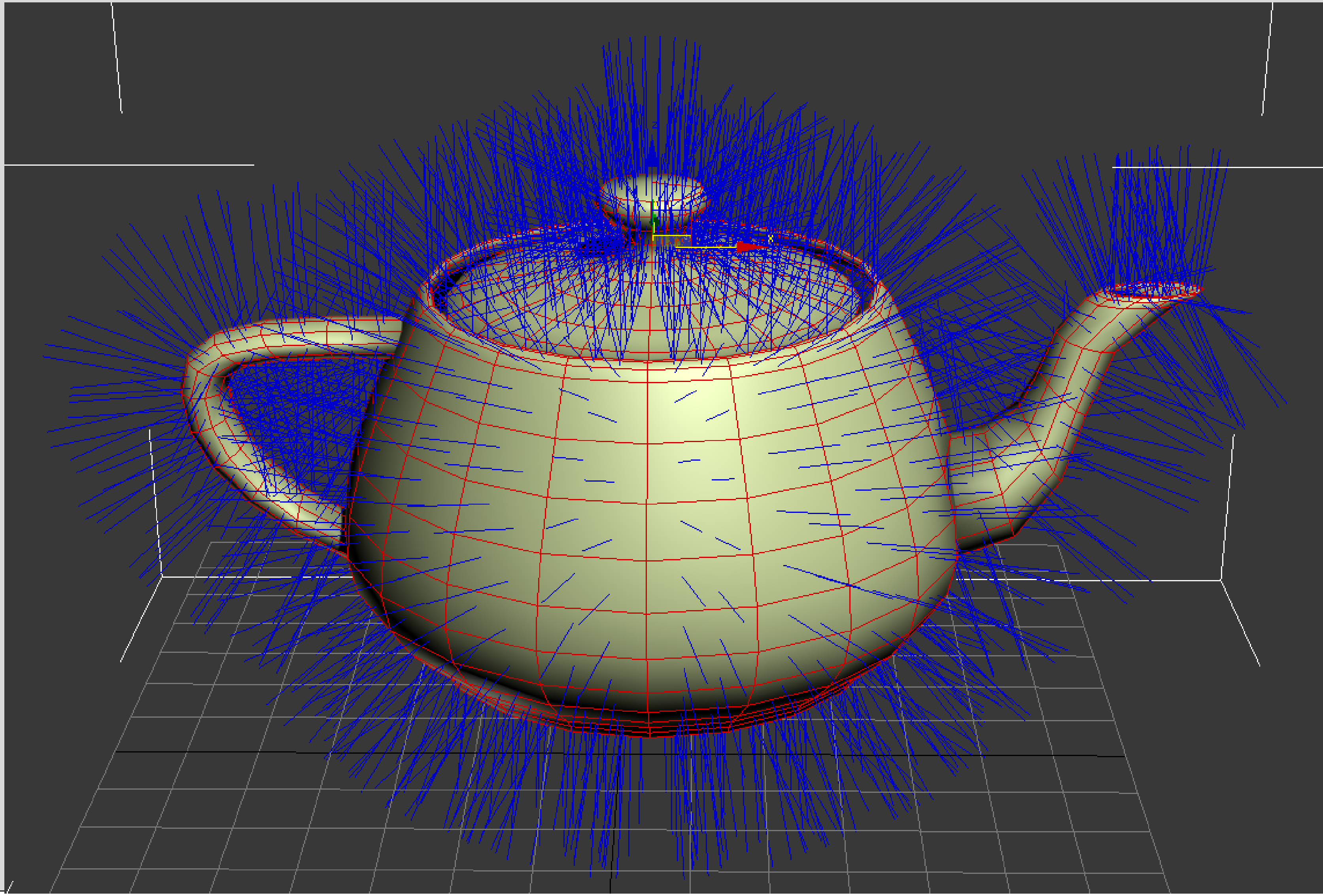
- The plane of the polygon divides 3D space into 2 *half-spaces*
- All points *P* are either in front of or behind the polygon.
- To determine the side determine:

$$d = \mathbf{n} \cdot (P - v_i)$$

  - $d < 0 \Rightarrow P$ behind
  - $d = 0 \Rightarrow P$ on polygon
  - $d > 0 \Rightarrow P$ in front



Polygon Normals



Cross Product in Computer Graphics

- The classic use of the cross product is figuring out the normal vector of a polygon
- The normal vector is fundamental to calculating which polygons are facing the camera
  - Which polygons are drawn and which can be ignored

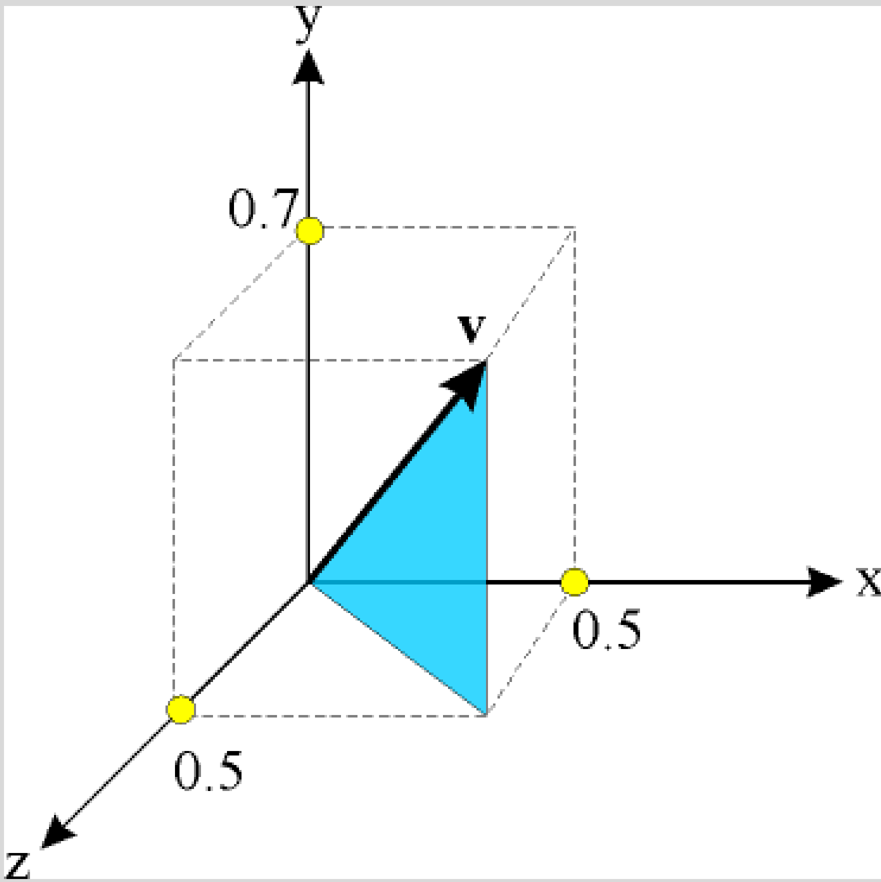
Cross vs. Dot Product

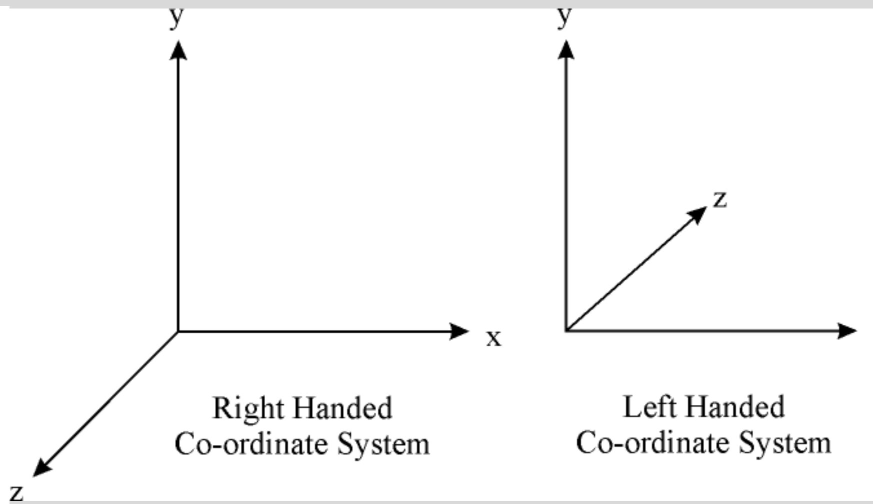
- A dot product of two vectors gives a *scalar*. It calculates angles.
- The cross product of two vectors gives a *vector*. It calculates direction.
- A dot B = B dot A
- A cross B /= B cross A

Co-ordinate Systems

- By convention we usually employ a *Cartesian basis*:
  - basis vectors are *mutually orthogonal* and *unit length*
  - basis vectors named **x**, **y** and **z**
- We need to define the relationship between the 3 vectors: there are 2 possibilities:
  - *right handed systems*: **z** comes out of page
  - *left handed systems*: **z** goes into page
  - (note: OpenGL uses a right handed system)
- This affects direction of rotations and specification of *normal vectors*

Cartesian co-ordinate System



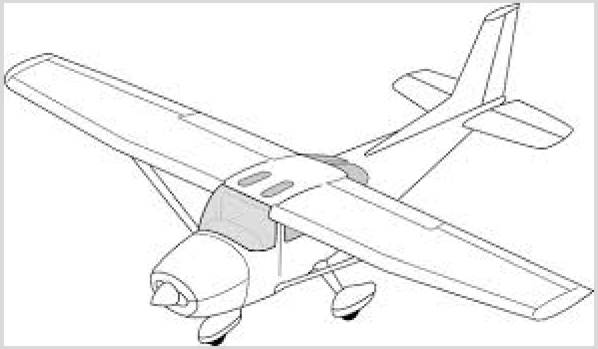


$$\mathbf{v} = \begin{bmatrix} 0.5 \\ 0.7 \\ 0.5 \end{bmatrix} =$$

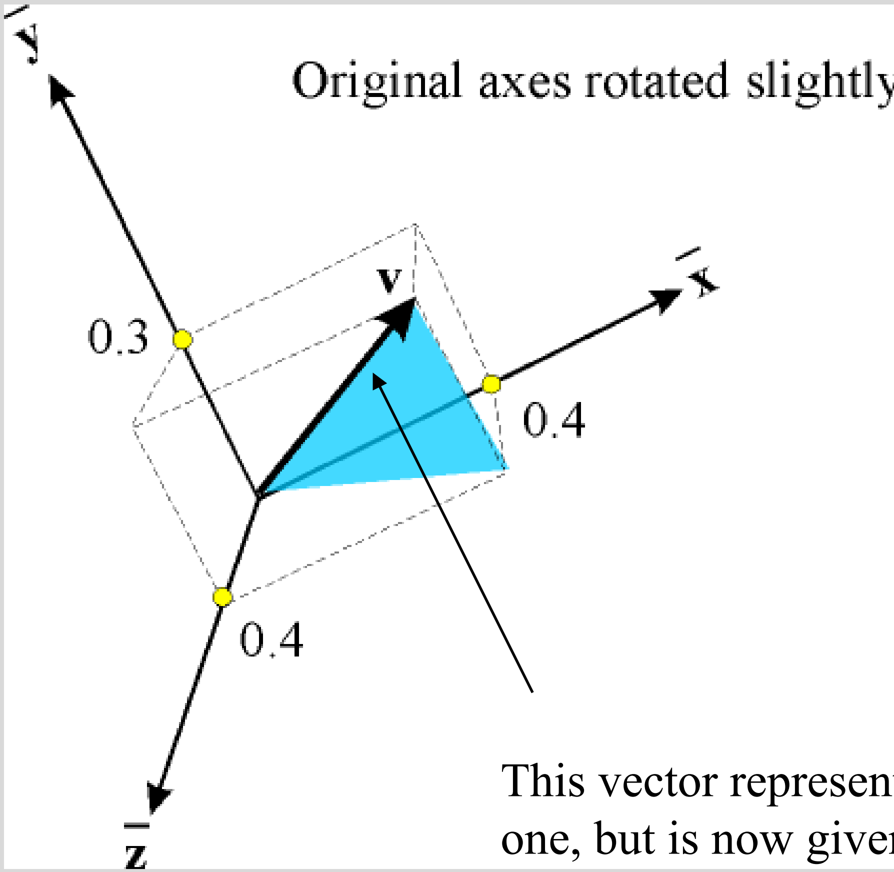
RHS LHS

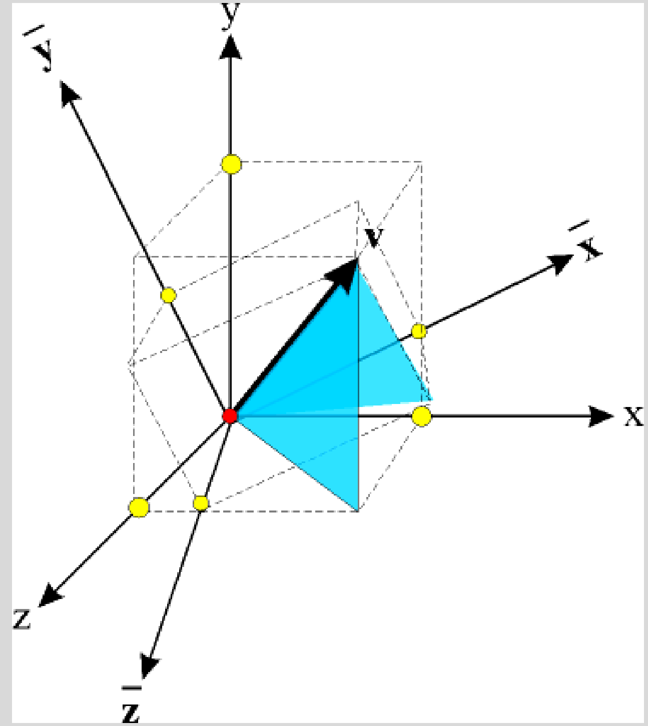
Cartesian co-ordinate System

- One of infinitely many possible orthonormal basis
- Global coordinate system in graphics is the canonical coordinate system
- Special because **x**, **y**, **z**, and origin are never explicitly stored
- However, if we want to use another coordinate system with origin **p** and orthonormal basis vectors **u**, **v**, **w**, the we *do* store those vectors explicitly – flight example
- The coordinate system associated with the plane is the *local coordinate system*



… same vector in a new co-ordinate system





This vector represents the same direction as the previous one, but is now given with respect to a new basis and therefore its value changes accordingly.

Change of Basis

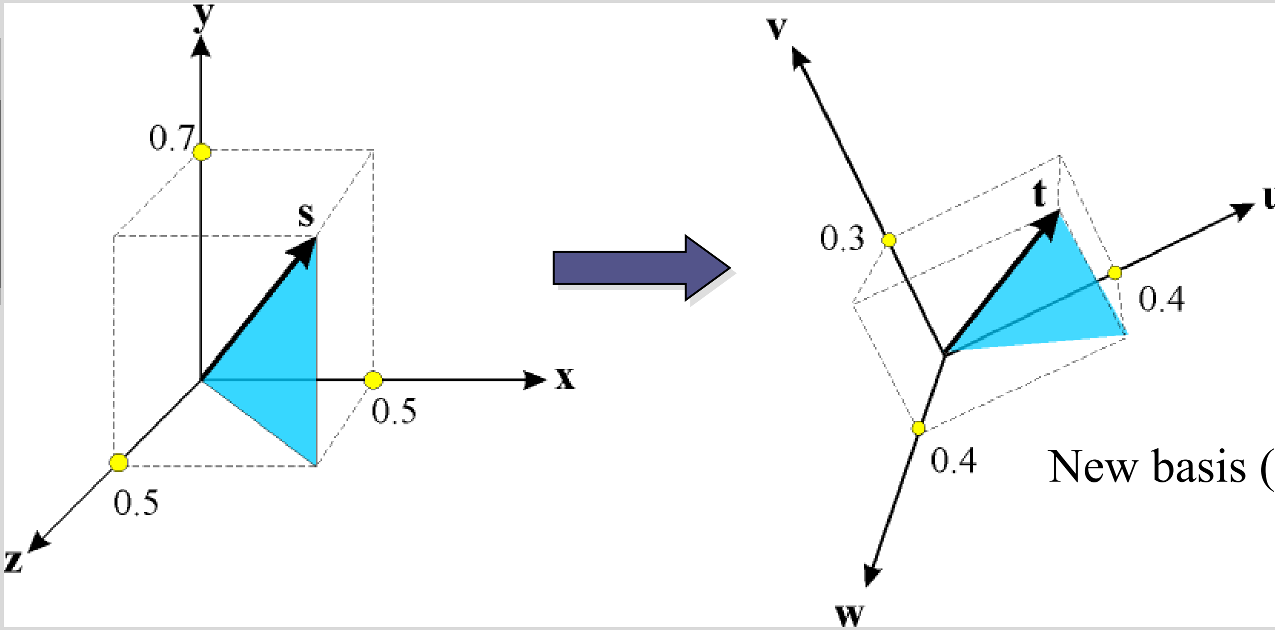
- If we know **s** defined w.r.t. basis **xyz** we can determine **t** which is the same vector defined w.r.t. basis **uvw**.
  - t_u is the projected distance of **s** onto **u**
  - t_v is the projected distance of **s** onto **v**
  - t_w is the projected distance of **s** onto **w**

$$\mathbf{t} = \begin{bmatrix} \mathbf{s}\cdot\mathbf{u} \\ \mathbf{s}\cdot\mathbf{v} \\ \mathbf{s}\cdot\mathbf{w} \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{bmatrix} \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} = \mathbf{Ms} \begin{cases} t_u = u_x s_x + u_y s_y + u_z s_z = \mathbf{u}\cdot\mathbf{s} \\ t_v = v_x s_x + v_y s_y + v_z s_z = \mathbf{v}\cdot\mathbf{s} \\ t_w = w_x s_x + w_y s_y + w_z s_z = \mathbf{w}\cdot\mathbf{s} \end{cases}$$

- Matrix **M** allows us to transform a vector from one basis to another $\Rightarrow$ **M** is a *transformation matrix*.
- Many common geometric operations can be expressed as a transformation matrix.

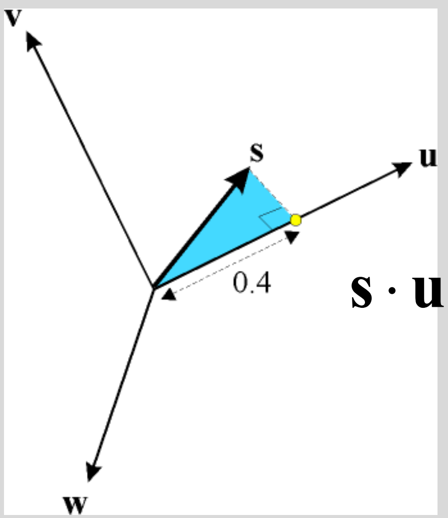
Change of Basis

$$\mathbf{s} = \begin{bmatrix} 0\cdot 5 \\ 0\cdot 7 \\ 0\cdot 5 \end{bmatrix}$$

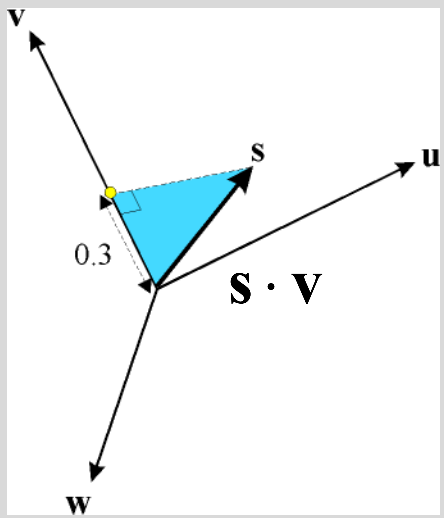


$$\mathbf{t} = \begin{bmatrix} 0\cdot 4 \\ 0\cdot 3 \\ 0\cdot 4 \end{bmatrix}$$

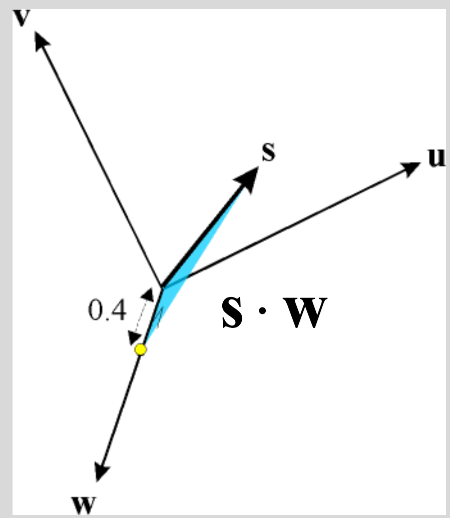
New basis (co-ordinate system)



$\mathbf{s}\cdot\mathbf{u}$



$\mathbf{s}\cdot\mathbf{v}$



$\mathbf{s}\cdot\mathbf{w}$

Change of Basis

- Normally the vectors forming the basis of a coordinate system are *unit length* and *mutually orthogonal*
  - basis is said to be *orthonormal*
- This leads to a useful property of the coordinate matrix: $\mathbf{M}^{-1} = \mathbf{M}^{\mathrm{T}}$
  - a property shared by all rotation matrices
  - not true for scaling transformation
- Therefore if we have a vector **t** defined w.r.t. basis **uvw** then the vector w.r.t. basis **xyz** is given by:

$$\mathbf{s} = t_u \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} + t_v \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} + t_w \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \begin{bmatrix} t_u \\ t_v \\ t_w \end{bmatrix} = \mathbf{M}^{-1}\mathbf{t} = \mathbf{M}^{\mathrm{T}}\mathbf{t}$$

Exercise

$$u = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad v = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad w = \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix}$$

- **a** in uvw

$$a = \begin{bmatrix} 7 \\ 4 \\ 4 \end{bmatrix}$$

- **a** in **xyz?**

Exercise

$$u = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad v = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad w = \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix}$$

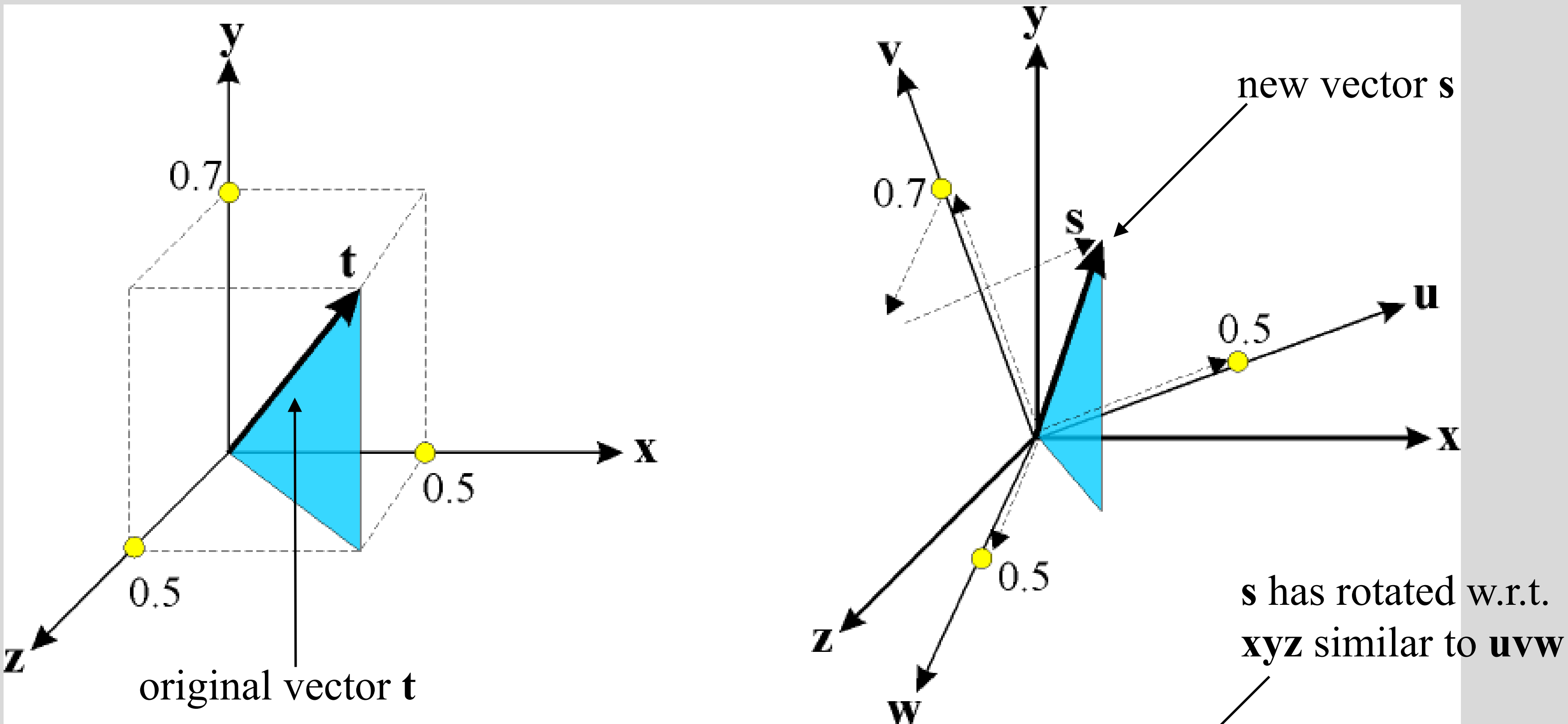
- **a** in uvw

$$a = \begin{bmatrix} 7 \\ 4 \\ 4 \end{bmatrix}$$

- **a** in **xyz?**
  - **Change of basis matrix?** $\begin{bmatrix} 1 & 1 & 2 \\ 2 & 0 & 3 \\ 3 & 1 & 3 \end{bmatrix}\begin{bmatrix} 7 \\ 4 \\ 4 \end{bmatrix}$

Change of Basis = Transformation

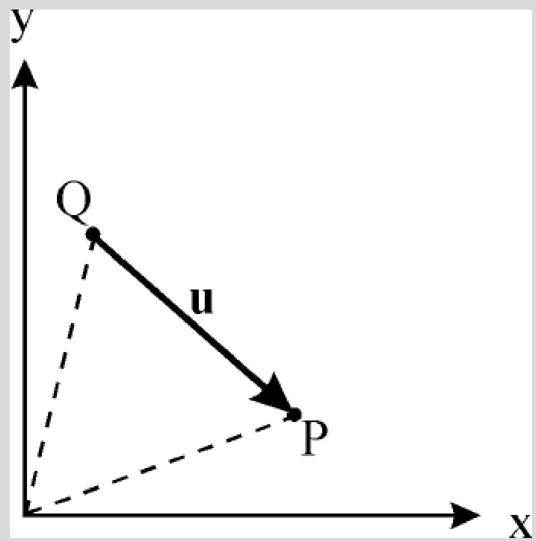
Changing basis is geometrically equivalent to transformation:



$$\mathbf{t} \text{ w.r.t. } \mathbf{xyz} = \begin{bmatrix} 0\cdot5 \\ 0\cdot7 \\ 0\cdot5 \end{bmatrix} \qquad \mathbf{s} \text{ w.r.t. } \mathbf{uvw} = \begin{bmatrix} 0\cdot5 \\ 0\cdot7 \\ 0\cdot5 \end{bmatrix} \qquad \mathbf{s} \text{ w.r.t. } \mathbf{xyz} = \begin{bmatrix} 0\cdot2 \\ 0\cdot8 \\ 0\cdot3 \end{bmatrix}$$

Affine Spaces

- Vectors define direction and magnitude only.
- To encode position we need to fix the *origin*.
- The origin is a *point*.
- *Affine space* = a set of points with an associated vector space with the operations difference and translate.
- Points are related by vectors: **u** = *P* - *Q* or *Q* + **u** = *P*



Linear Algebra

- Vector addition, subtraction, multiplication
- Normalising vectors
- Dot Product
- Cross Product & Polygon normals
- Changing Basis
- Next:
  - Geometric Transformations!