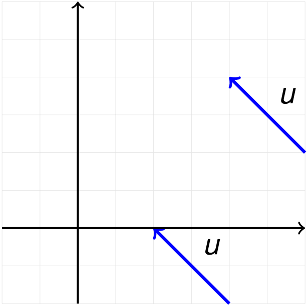
Vectors Review

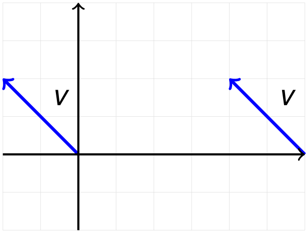
- Vectors have **no** position; they are defined by their *length* and their *direction*; more precisely, they are defined as the **difference** between two points



- That said, at times it is often handy to place them at the origin; if we do this then a point, $p = (x, y)$, can be represented by the vector v from the origin to p, $v = (x - 0, y - 0) = (x, y)$

Vectors Review

- Vectors have **no** position; they are defined by their *length* and their *direction*; more precisely, they are defined as the **difference** between two points
- That said, at times it is often handy to place them at the origin; if we do this then a point, $p = (x, y)$, can be represented by the vector v from the origin to p, $v = (x - 0, y - 0) = (x, y)$



Vectors Review (contd.)

Vector Bases

- Just as words are made up from the basic unit of language, letters, likewise vectors are composed of basic vectors
- We call this set of basic vectors a **basis**
- The vectors in a basis do not have to be orthogonal but they usually are
- The vectors in a basis do not have to be of unit length but they usually are
- We will encounter several co-ordinate systems during the semester and each will actually be a vector basis; that is, any point in the co-ordinate system will be representable by some combination of the basis vectors

Vectors Review (contd.)

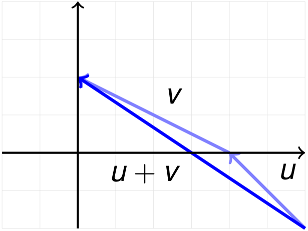
Vector Bases (contd.)

- Some maths textbooks use the notation that the vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are the three unit-length, orthogonal vectors in the basis; other books use the notation $\mathbf{i}, \mathbf{j}, \mathbf{k}$
- So if a vector, v, is drawn pointing from $(2, 0)$ to $(0, 3)$
  - in one notation it is $v = -2\mathbf{x} + 3\mathbf{y}$;
  - in another notation (book) it could be $v = -2\mathbf{i} + 3\mathbf{j}$;
  - and it could also be the vector $v = (-2, 3)$, where it is implicit that we start from the origin $(0, 0)$

Vectors Review (contd.)

Arithmetic

- vector addition is “head to tail”:

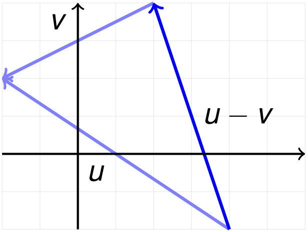


- vector subtraction is “head to head”:
- scalar multiplication is a “fraction” of a vector and is a vector, itself; if $u = (u_1, u_2, \ldots, u_n)^T$ then when r is a real no. (a scalar), $ru = (ru_1, ru_2, \ldots, ru_n)^T$:

Vectors Review (contd.)

Arithmetic

- vector addition is "head to tail":
- vector subtraction is "head to head":

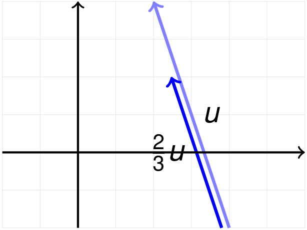


- scalar multiplication is a "fraction" of a vector and is a vector, itself; if $u = (u_1, u_2, \ldots, u_n)^T$ then when r is a real no. (a scalar), $ru = (ru_1, ru_2, \ldots, ru_n)^T$:

Vectors Review (contd.)

Arithmetic

- vector addition is “head to tail”:
- vector subtraction is “head to head”:
- scalar multiplication is a “fraction” of a vector and is a vector, itself; if $u = (u_1, u_2, \ldots, u_n)^T$ then when r is a real no. (a scalar), $ru = (ru_1, ru_2, \ldots, ru_n)^T$:



Vectors Review (contd.)

Handedness of co-ordinate systems

- The **handedness** of a 3D co-ordinate system is crucial for keeping consistency between the three (mutually perpendicular) dimensions: given the positive directions of 2 of the axes, the handedness determines which way the positive third axis points

- In an RHCS where the normal ordering is x, y, z **if** the y axis points “up” and the x axis points right then the z axis would point out of the page
- OpenGL is a **right** handed co-ordinate system; Direct3D is a **left** handed co-ordinate system

Vectors Review (contd.)

Handedness of co-ordinate systems

- The **handedness** of a 3D co-ordinate system is crucial for keeping consistency between the three (mutually perpendicular) dimensions: given the positive directions of 2 of the axes, the handedness determines which way the positive third axis points

Right-Hand Curl (Screw) Rule

Curl your **right** hand around the third axis grasping from positive x to y, and your thumb will point in the positive z direction; this assumes that normal ordering of axes is x, y, z.

-

- In an RHCS where the normal ordering is x, y, z if the y

Vectors Review (contd.)

Handedness of co-ordinate systems

- The **handedness** of a 3D co-ordinate system is crucial for keeping consistency between the three (mutually perpendicular) dimensions: given the positive directions of 2 of the axes, the handedness determines which way the positive third axis points

Important:

When we talk about **cross products** later remember that if the third axis is to point **towards** you then the ordering has got to be **anti-clockwise**

Vectors Review (contd.)

Handedness of co-ordinate systems

- The **handedness** of a 3D co-ordinate system is crucial for keeping consistency between the three (mutually perpendicular) dimensions: given the positive directions of 2 of the axes, the handedness determines which way the positive third axis points
- In an RHCS where the normal ordering is x, y, z **if** the y axis points "up" and the x axis points right then the z axis would point out of the page
- OpenGL is a **right** handed co-ordinate system; Direct3D is a **left** handed co-ordinate system

Vectors Review (contd.)

Handedness of co-ordinate systems

- The **handedness** of a 3D co-ordinate system is crucial for keeping consistency between the three (mutually perpendicular) dimensions: given the positive directions of 2 of the axes, the handedness determines which way the positive third axis points

- In an RHCS where the normal ordering is x, y, z **if** the y axis points "up" and the x axis points right then the z axis would point out of the page
- OpenGL is a **right** handed co-ordinate system; Direct3D is a **left** handed co-ordinate system

Vectors Review (contd.)

Dot Product of two Vectors, $u \cdot v$

- For two vectors $u = (u_1, u_2, \ldots, u_n)^T$ and $v = (v_1, v_2, \ldots, v_n)^T$ the **dot product**, $u \cdot v$ is defined as

$$u \cdot v = \sum_{i=1}^{n} u_i v_i = u_1 v_1 + \cdots + u_n v_n = v \cdot u$$

- The importance of the dot product is that geometrically it is the orthogonal projection of one vector on to the other, expressed as a multiple of the other vector's length
- That is, it gives a measure of how much of one vector points in the other's direction
- The dot product returns a number, or **scalar**

Vectors Review (contd.)

Dot Product of two Vectors, $u \cdot v$

- For two vectors $u = (u_1, u_2, \ldots, u_n)^T$ and $v = (v_1, v_2, \ldots, v_n)^T$ the **dot product**, $u \cdot v$ is defined as

$$u \cdot v = \sum_{i=1}^{n} u_i v_i = u_1 v_1 + \cdots + u_n v_n = v \cdot u$$

- The importance of the dot product is that geometrically it is the orthogonal projection of one vector on to the other, expressed as a multiple of the other vector's length
- That is, it gives a measure of how much of one vector points in the other's direction
- The dot product returns a number, or **scalar**

Vectors Review (contd.)

Dot Product of two Vectors, $u \cdot v$

- For two vectors $u = (u_1, u_2, \ldots, u_n)^T$ and $v = (v_1, v_2, \ldots, v_n)^T$ the **dot product**, $u \cdot v$ is defined as

$$u \cdot v = \sum_{i=1}^{n} u_i v_i = u_1 v_1 + \cdots + u_n v_n = v \cdot u$$

- The importance of the dot product is that geometrically it is the orthogonal projection of one vector on to the other, expressed as a multiple of the other vector's length
- That is, it gives a measure of how much of one vector points in the other's direction
- The dot product returns a number, or **scalar**

Vectors Review (contd.)

Dot Product of two Vectors, $u \cdot v$

- For two vectors $u = (u_1, u_2, \ldots, u_n)^T$ and $v = (v_1, v_2, \ldots, v_n)^T$ the **dot product**, $u \cdot v$ is defined as

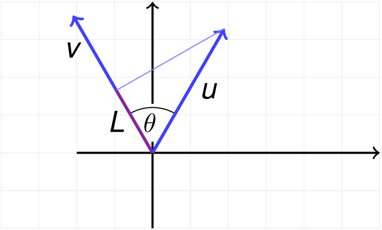
$$u \cdot v = \sum_{i=1}^{n} u_i v_i = u_1 v_1 + \cdots + u_n v_n = v \cdot u$$

- The importance of the dot product is that geometrically it is the orthogonal projection of one vector on to the other, expressed as a multiple of the other vector's length
- That is, it gives a measure of how much of one vector points in the other's direction
- The dot product returns a number, or **scalar**

Vectors Review (contd.)

Dot Product of two Vectors, $u \cdot v$

- Two vectors u and v, graphically



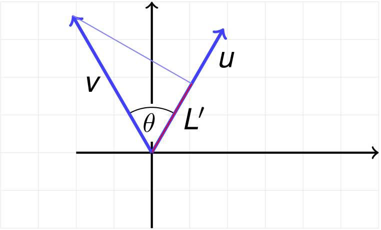
$$L = |u| \cos\theta \ , \ u \cdot v = L|v| = |u||v| \cos\theta$$

- ... which is also

Vectors Review (contd.)

Dot Product of two Vectors, $u \cdot v$

- Two vectors u and v, graphically
- ... which is also

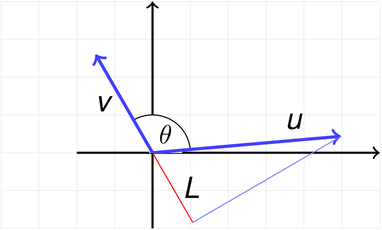


$L' = |v| \cos\theta$, $u \cdot v = L'|u| = |u||v| \cos\theta$
Note, $L' \neq L$

Vectors Review (contd.)

Dot Product of two Vectors, $u \cdot v$

- Two more vectors u and v, graphically, $u \cdot v$ is



$L = |u| \cos \theta$ and $u \cdot v = L|v| = |u||v| \cos \theta$
Since $\theta > 90^\circ$, $\cos \theta < 0$ and then $u \cdot v < 0$

Vectors Review (contd.)

Dot Product of two Vectors, $u \cdot v$

Two important properties of dot product

- if u and v are orthogonal then $u \cdot v = 0$ ($\cos\theta = 0$)
- if u is of length 1 then $u \cdot u = 1$ ($\cos\theta = 1$)

The connection between the *geometric* interpretation of the dot product and the algebraic formula can be found here

Forward note:

When we describe vectors in terms of matrices

$$u \cdot v = u^T v = v^T u \neq v u^T \neq u v^T$$

Vectors Review (contd.)

The Normal

- In 2-D (3-d, respectively) the **normal** is a vector that is **perpendicular** to a given line (resp. plane)
- If you are given a line $ax + by + c = 0$ then its normal is the vector (a, b) and is at **right angles** to the line
- In graphics when we represent objects (cars, animals, etc.) by “wire meshes” of smaller polygons the normal will play a crucial role because it will allow us determine if our camera is pointing towards a polygon or if the polygon faces the other way (the polygon visibility problem)