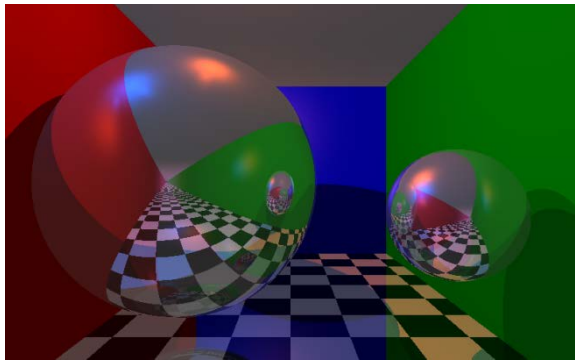
CSI 4105 Computer Graphics

Spring 2017

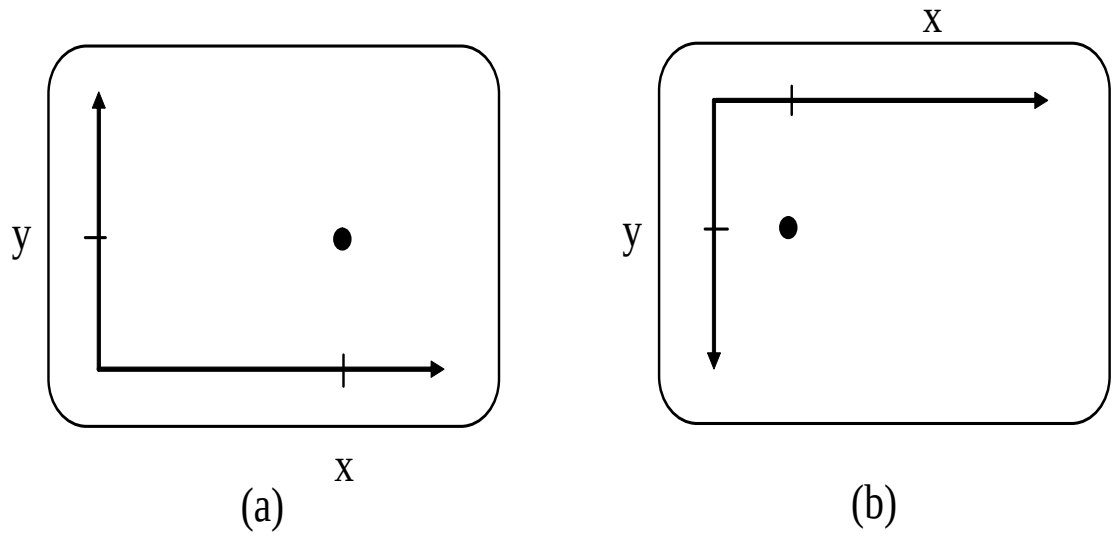
Lecture 1:
Mathematical Preliminaries

Seon Joo Kim
Yonsei University



Cartesian Reference Frame

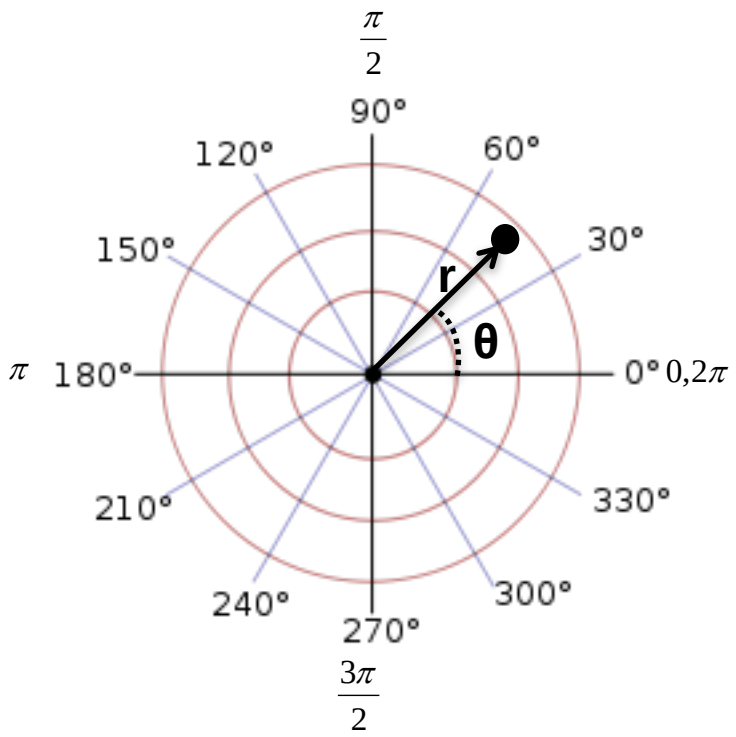
Cartesian coordinates represents points in a rectangular frames (in 2D or 3D). This is perfect for computer graphics, especially when working with the display or framebuffer, since they are designed to be rectangular *Note that, the configuration of the frame is not fixed.*



Screen Cartesian reference systems : (a) coordinate origin at the lower-left screen corner and (b) coordinate origin in the upper-left corner. In OpenGL, the bottom of the screen is (0,0).

Thinking in other coordinates (polar)

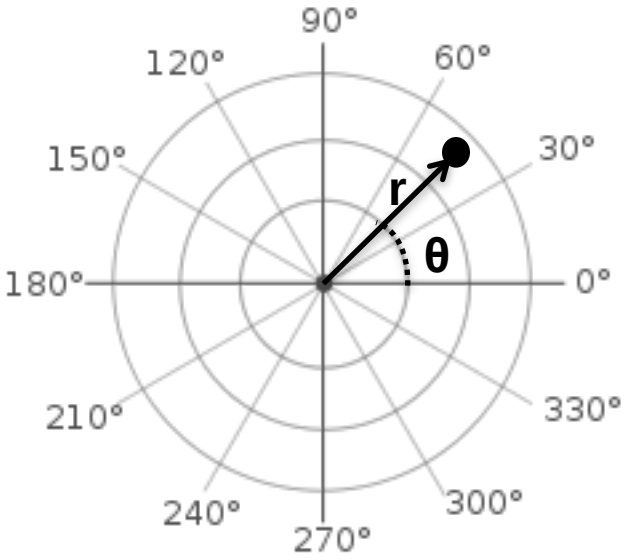
In polar coordinates, a position is specified with a radial distance, ***r,*** from the coordinate origin, and an angular displacement, ***θ***, from the horizontal.

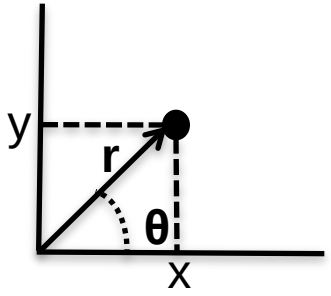


A polar coordinate reference frame, formed with concentric circles and radial lines.

A point in polar coordinates is expressed as:
P(r, θ)

Converting between Polar and Cartesian





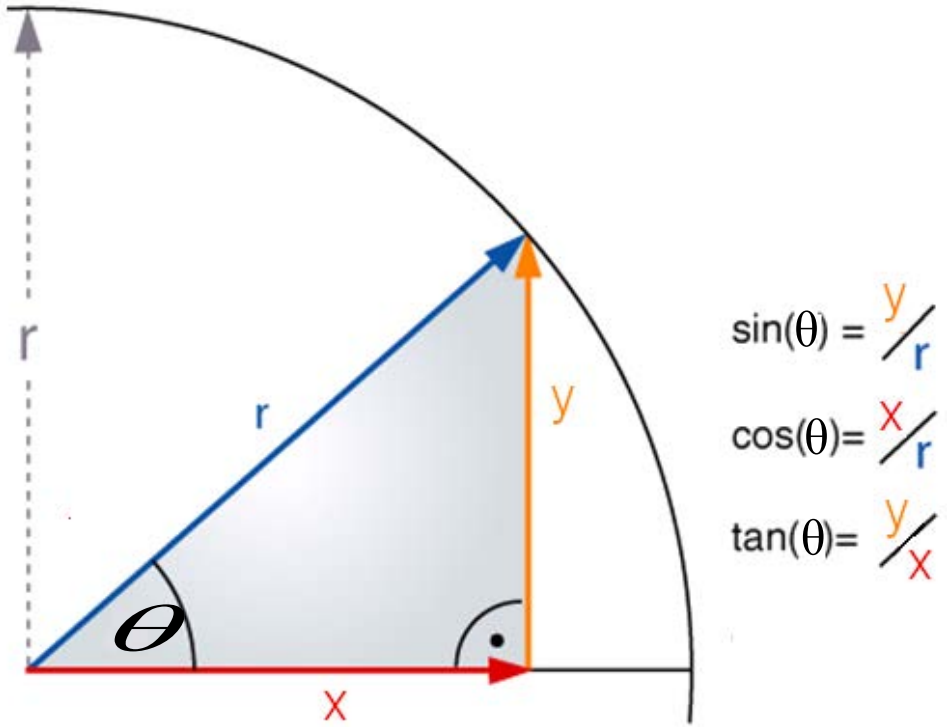
$$r = \sqrt{x^2 + y^2} \quad , \quad \theta = \tan^{-1}\left(\frac{y}{x}\right)$$

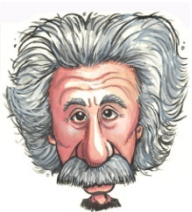
$$x = r\cos\theta$$
$$y = r\sin\theta$$

Converting between polar coordinates (r, θ) and Cartesian (x,y) is a powerful tool in Graphics. Remember it!!!

Basic Trigonometry

- It is all about the right triangle (Trig)
  - Look carefully, this is polar coorinate!





Remember these basic formulas! They are also very important. Matter of fact, if you can remember these, you can easily derive the polar transform.

Some comments about trig functions

- In C, C++, and Java
- Be careful about your trig functions in C++/Java accepts or returns angles in radians or degrees
- Remember, there are 360deg and 2pi rads in a circle, so:

$$radian = (\deg \times 2\pi) / 360$$

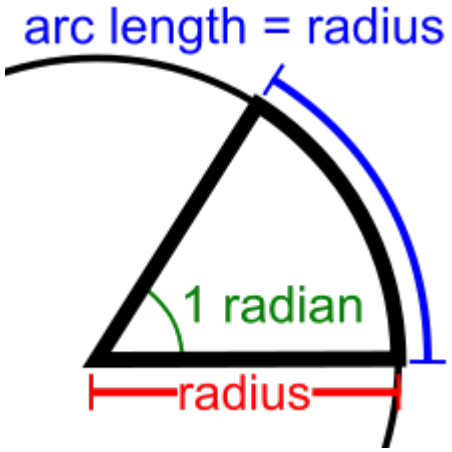
$$\deg = 360 \times (radian / 2\pi)$$

- Also, remember that inverses are only defined over certain ranges. For example, $\tan^{-1}$ or arctan is defined over: $-\frac{\pi}{2} < y < \frac{\pi}{2}, \qquad y = \arctan x$

Ever wonder what a radian is?

One radian measures an angle whose sweeps an arc length “s” that is the same size of the radius.

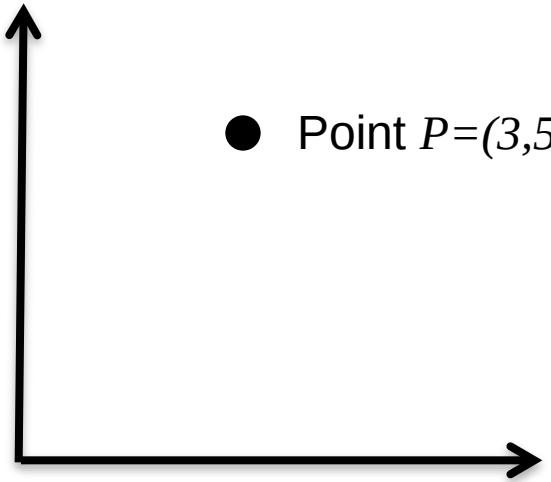
$$\theta = \frac{s}{r}$$



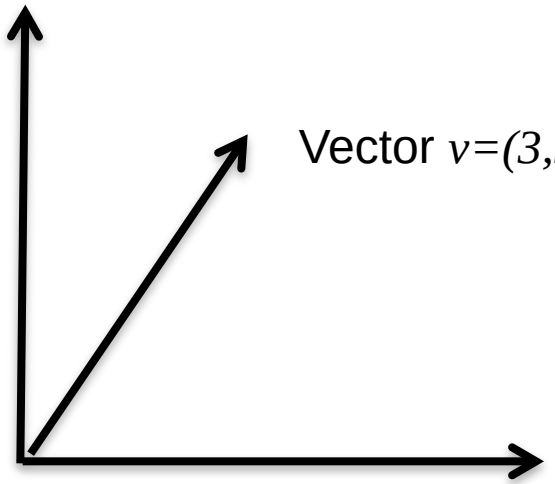
An angle θ subtended by a circular arc of length *s* and *r*. This is 1 rad.

Since the circumference of a circle is given as $2\pi r$, there are 2π radians in a circle.

Vectors and Points



A simple point.



A vector .. See the difference? No.
You can't see it, you need to "think" it.

Points and **vectors** look similar in terms of representation, but are conceptually different.

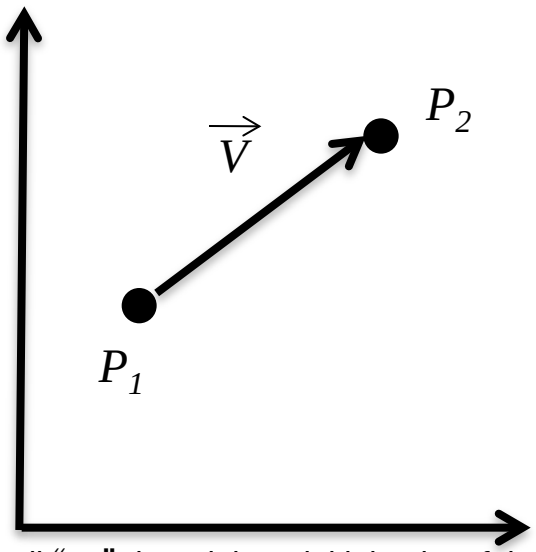
A **point** is a location in space.
A **vector** is a direction and magnitude in space.

Notation convention
Points will be in uppercase Roman Letters (e.g. P, Q, R)
Vectors will be in lowercase Roman Letters (e.g. u, v, w) sometimes with arrows over them.

Vectors

- Vectors are defined by two points:

  P_1 and P_2.



We call "P_1" the origin or initial point of the vector.
We call "P_2" the terminal or ending point of the vector.

Vector v is expressed as:

$$\vec{v} = P_2 - P_1$$
$$= (x_2 - x_1 , y_2 - y_1) = (v_x , v_y)$$

We can say that vector V is in the direction:

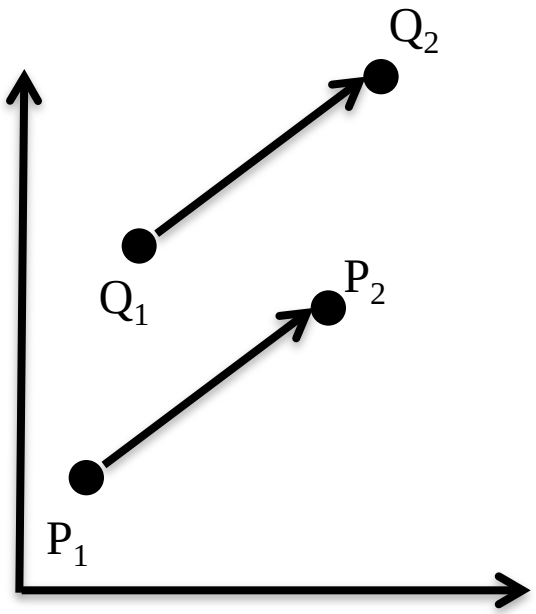
$$(v_x , v_y)$$

We can its magnitude (or length), $|v|$, is:

$$\left|\vec{v}\right| = \sqrt{v_x^2 + v_y^2}$$

Vectors

- Different pairs of points result in the same vector.



In this case, the vector $\overrightarrow{Q_1Q_2}$

is the same as the vector $\overrightarrow{P_1P_2}$

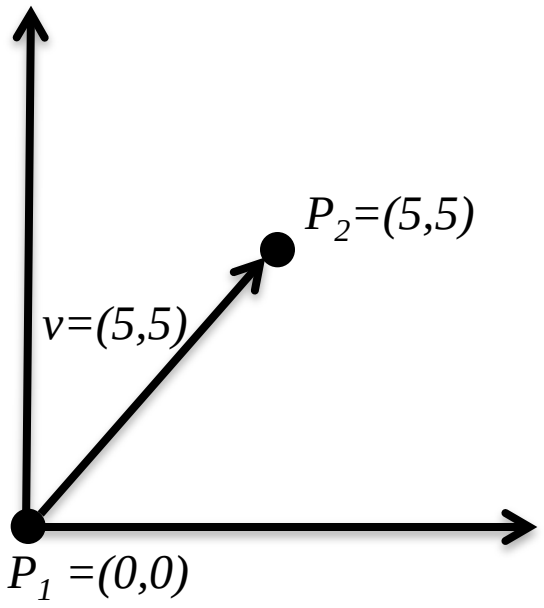
Conceptually we **cannot say** the same think about the points. P_1 and Q_1 – they are not the same locations in space.

Are you starting to understand the difference? You need to "think it", not see it.

Its very hard to draw the "$\rightarrow$" over the vector variables in PPT, so I will often omit it.

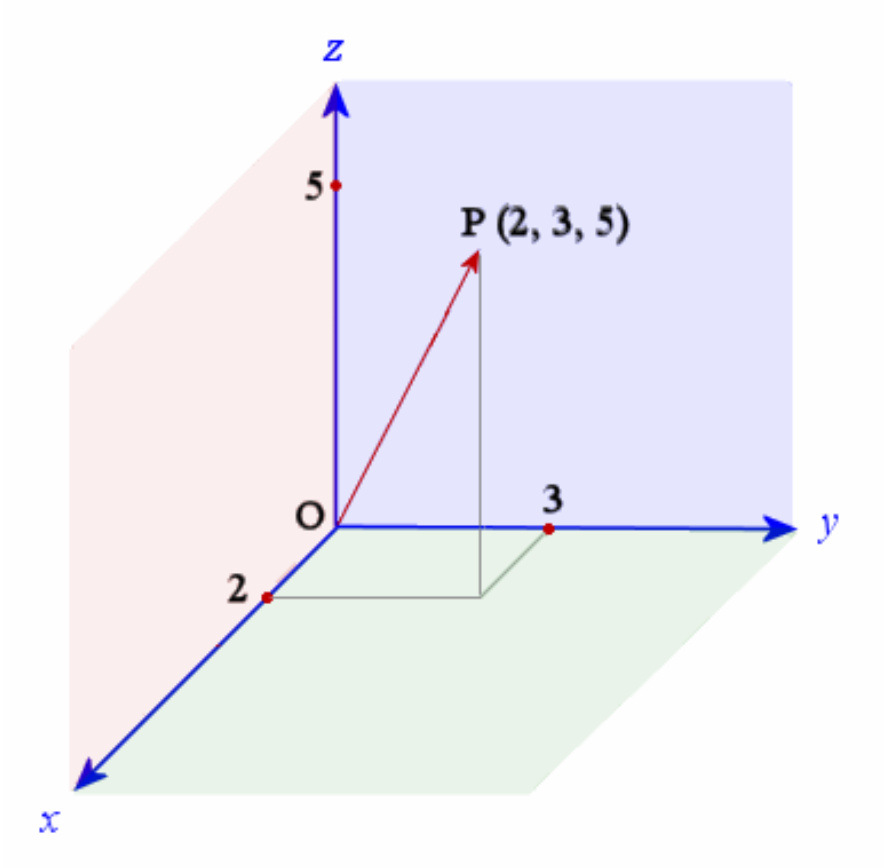
Vectors

- Generally, once we have the vector, we don't care about the points anymore. So, often you'll just say I have a vector $v=(5,5)$.
- But, technically a vector is defined by two points. So if you are given a vector, like $v=(5,5)$, we can always assume it is formed from a point pair $P_2=(5,5)$ and $P_1=(0,0)$.



3D (and N-D) Points &Vectors

- Vectors and points can be of any dimension
- For us in CG, the most common will be 2D and 3D
  - Its hard to draw vectors that have more than 3D dimensions!

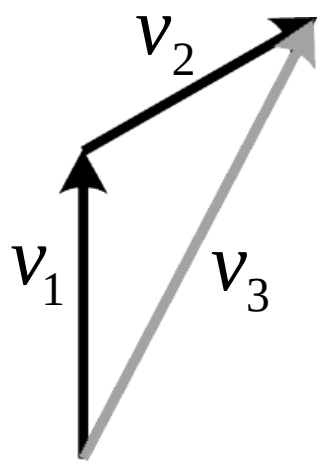


For many of the concepts in CG we will draw 2D vectors and points, even though we are talking about 3D. It's just easier that way.

Vector Math

Vector Addition

$$v_1 + v_2 = \left(v_{1x} + v_{2x}\ ,\ v_{1y} + v_{2y}\ ,\ v_{1z} + v_{2z}\right)$$

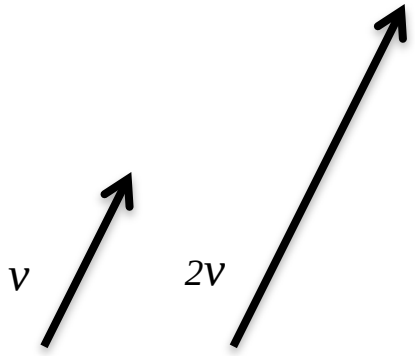


We can add vectors together to get new vectors.
Of course, we can also apply subtraction.

Scalar Multiplication

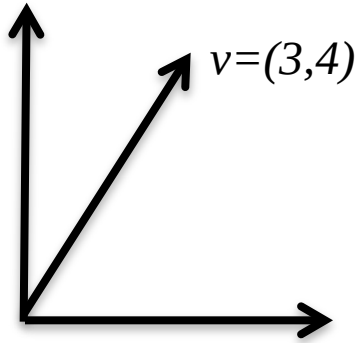
$$\alpha v = (\alpha v_x\ ,\ \alpha v_y\ ,\ \alpha v_z)$$

We can uniformly scale vector by multiplying by a scalar value. Note that scalar by a positive value doesn't change the direction (a negative flips it).



Vector Normalization

Sometimes we want our vector to have size “1”. Often this means we are more interested in the direction of the vector and not its magnitude.



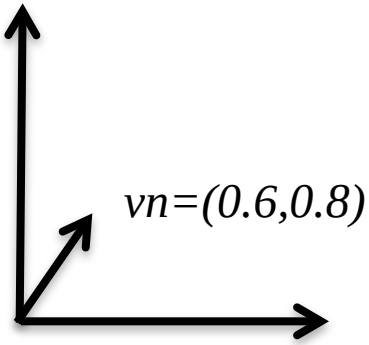
To normalize take the vector and divide by its magnitude.

$$v = (3,4)$$

$$|v| = \sqrt{3^2 + 4^2} = 5$$

$$vn = (\frac{3}{5}, \frac{4}{5}) = (0.6,0.8)$$

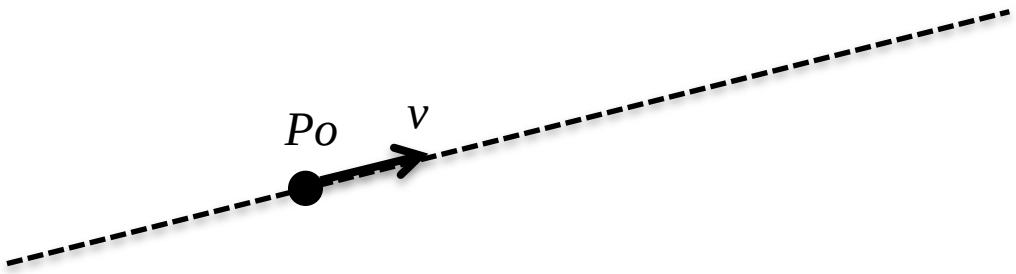
We call, *vn*, a **unit vector**.



Unit vector *vn* has same direction as v, but length is 1.

Vectors + Parametric Line Eq

- You will see there are many uses for vectors in Graphics
- But one for solving problems is the parametric line eq



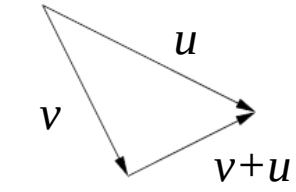
A point and a vector define any point on the line: $l = Po + t * v$

$$lx = Px + t\,vx$$
$$ly = Py + t\,vy$$

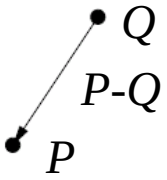
Vector and Point Math

- We consider point-vector addition, $R+v$, as a translation of point R by vector v (the result is a new point)
- As already mentioned, point subtraction results in a vector
- Certain operation, like point + point are not defined*.
- The following is table of allowed operations and the results.

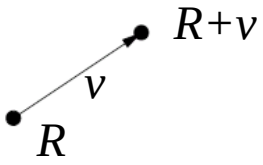
$vector \leftarrow scalar \cdot vector$, $vector \leftarrow vector/scalar$	scalar-vector multiplication
$vector \leftarrow vector + vector$, $vector \leftarrow vector - vector$	vector-vector addition
$vector \leftarrow point - point$	point-point difference
$point \leftarrow point + vector$, $point \leftarrow point - vector$	point-vector addition



vector addition



point subtraction

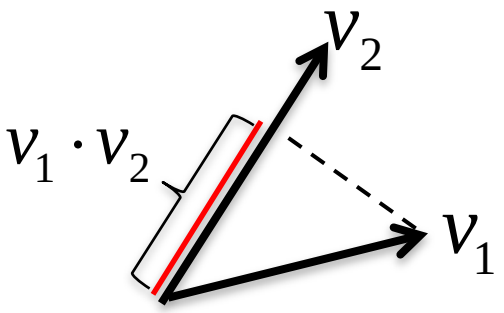


point−vector addition

* Point+point is not technically defined, but it does occur, e.g. computing the mid-point between two points, or the geometric centroid of a set of points.

Vector inner product (or dot product)

- The dot product



$$v_1 \cdot v_2 = \left(v_{1x} v_{2x} + v_{1y} v_{2y} + v_{1z} v_{2z} \right)$$

Also written as: $v_1^T v_2$

The result is a scalar value (that is, its only a single number).

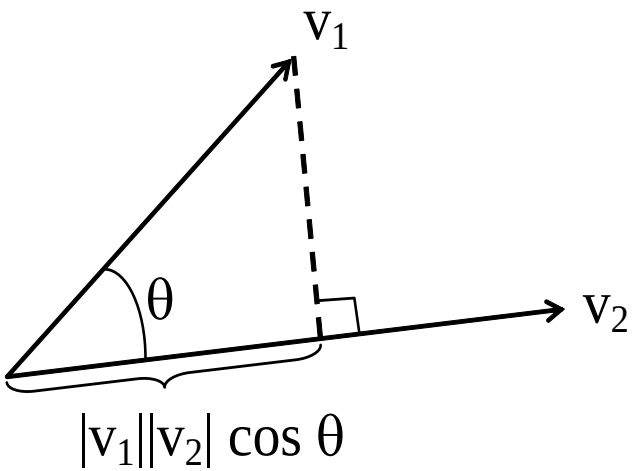
When *v2* is normalized the dot product represents the amount of *v1* that is in the direction of *v2*.

Sometimes we also call this "the projection of *v1* onto *v2*".

Nice applet: http://www.falstad.com/dotproduct/

Dot Product

- Dot product can also be computed using the cosθ.

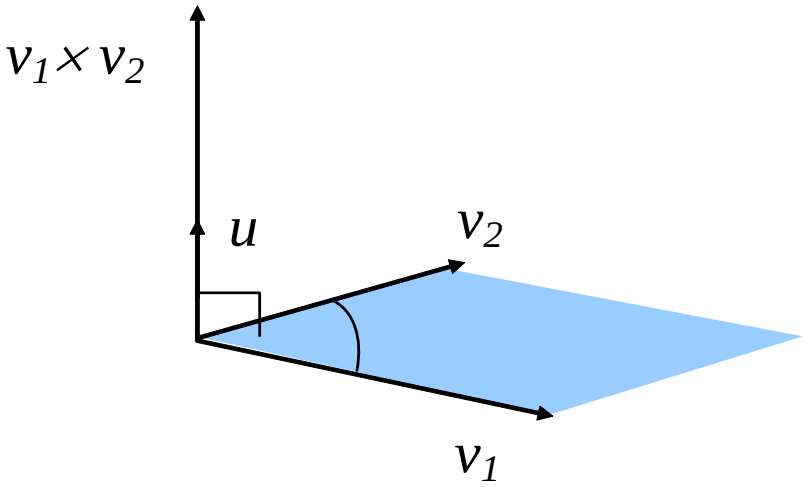


- If both vectors are normalized, we have:

$$\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 = \cos\theta$$

Vector Product (cross product)

Multiplication of two vectors to produce another *vector*.



(Vector Notation)

$$v_1 \times v_2 = \begin{pmatrix} v_{1y}v_{2z} - v_{1z}v_{2y} \\ v_{1z}v_{2x} - v_{1x}v_{2z} \\ v_{1x}v_{2y} - v_{1y}V_{2x} \end{pmatrix}$$

Where u is a unit vector that is perpendicular to both v_1 and v_2

The cross product gives a vector in a direction perpendicular to the two original vectors (u) and with a magnitude equal to the area of the shaded parallelogram

Nice applet demo at:
http://physics.syr.edu/courses/java-suite/crosspro.html

Properties of Vector Cross Product

Cross product of parallel vectors

$$v_1 \times v_2 = v_2 \times v_1 = \vec{0} \qquad \text{iff } v_1 \text{ parallel to } v_2$$

Anti-commutative

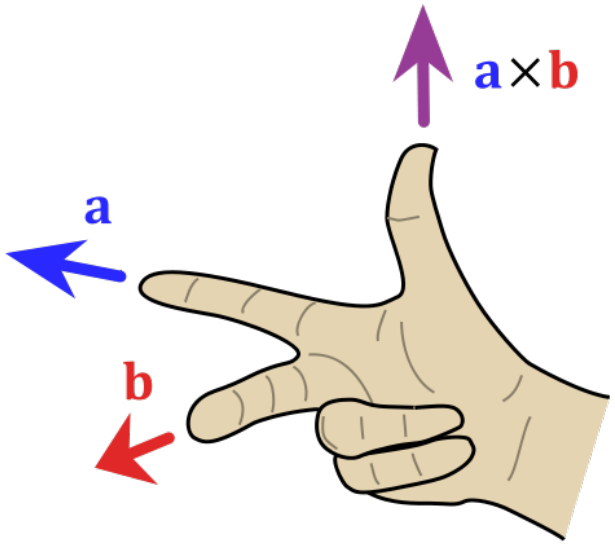
$$v_1 \times v_2 = -\left(v_2 \times v_1\right)$$

Not associative

$$v_1 \times \left(v_2 \times v_3\right) \neq \left(v_1 \times v_2\right) \times v_3$$

Right Hand/Left Hand Rule

- How do you know which way the cross-product vector is pointing?
- We define our coordinate system using either the "right-hand rule" or "left-hand rule"
  - In CG, we almost always used right-hand rule
    - (sorry you lefties!)



For a x b, use your right hand. Have your index finger be the "a", and the 2nd finger be the "b". Your thumb is the direction of a x b.

Another way is to curl your hand from a to b.

Try this with your left hand and you'll see you'll need to hold your hand upside down. This means with left hand rule the result will be –(a x b) [which is the same as (b x a) in right hand rule!]

Vector (the term)

- You are going to see the term "vector" used in many different contexts.
  - Don't confuse that with the mathematical concept of vector.
- Examples:

In programming

int a[10]; (arrray or vector)

Java and C++ both have "vector class". Typically acts like a 1D array (collection of data)

http://java.sun.com/j2se/1.3/docs/api/java/util/Vector.html

In Math

Vector notation is simply a 1-column (multiple rows) matrix. It doesn't necessarily mean it is conceptually a "vector", it could be a point.

$$a = \begin{bmatrix} 1 \\ 4 \\ 5 \\ 0 \end{bmatrix}$$

Matrices and Vectors

- Matrix and vector math will be very useful in computer graphics:
- Almost all of our operations can be expressed by matrix and vector operations
- Recall basic matrix math:

$$x = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} * \begin{pmatrix} 9 \\ 8 \\ 7 \end{pmatrix}$$

$$1*9+2*8+3*7=46$$

$$4*9+5*8+6*7=118$$

$$x = \begin{pmatrix} 46 \\ 118 \end{pmatrix}$$

See: http://en.wikipedia.org/wiki/Matrix_(mathematics)

Only need to care about basic operations + multiplication.

Summary: Basics

- Coordinate Frames
- Polar Coordinates & Trig
  - Please memorize basic trig formula and polar coordinate to Cartesian transform
- Points & Vectors, Vector Normalization
- Vector operators
  - Dot product and cross product
- Matrix math

* We will review basic calculus when we discuss curves and particle systems, it's a bit too far off now.

Up Next …

- Programming with OpenGL