Cosc342: Basic Maths

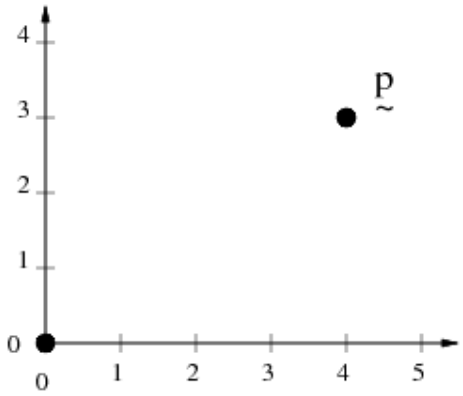
Vectors and Matrices

Here we provide an overview of:

- how we can represent **points** in space
- discuss **vectors**
- how we can represent **lines** in space
- discuss **matrices**

Points

Points are fundimental to the work that is done in graphics,



To specify the point shown we would say that, from the origin, we take 4 steps in the x direction, and 3 steps in the y direction.

We typically say that this point is at $x = 4$, and $y = 3$.

We can capture the meaning of "taking steps" by writing the point as

$$\underset{\sim}{\mathbf{p}} = x \underset{\sim}{\mathbf{i}} + y \underset{\sim}{\mathbf{j}}$$

where we use $\underset{\sim}{\mathbf{i}}$ to represent a step in the x-direction, and $\underset{\sim}{\mathbf{j}}$ to represent a step in the y-direction.

About the notation

The fact that $\underset{\sim}{\mathbf{p}}$, $\underset{\sim}{\mathbf{i}}$, and $\underset{\sim}{\mathbf{j}}$ are not simple numbers is indicated in printed material (books) by printing them in **bold**.

When people write manuscripts to be printed, they signify that something is to be printed in bold by drawing a *tilde* underneath it.

Since we cannot write in bold on paper or the whiteboard, it is normal to just use the tilde.

In this write up, we shall show these as bold, (since this is printed material), **and** we shall show the tilde underneath.

`Note: There are other notations used, such as` $\vec{p}$`.`
`We will not be using these.`

Higher dimensions

We will want to work in three dimensions (3D), which means we will need to be able to take steps in the z direction. Doing this means we can write a 3D point as:

$$\underset{\sim}{\mathbf{p}} = x\ \underset{\sim}{\mathbf{i}} + y\ \underset{\sim}{\mathbf{j}} + z\ \underset{\sim}{\mathbf{k}}$$

More convenient notation

While this way of writing points is meaningful, it is also rather cumbersome, so instead we write the components of the point as a **tuple**. (A tuple is just a structured collection of information, for example we could have a tuple that contained (name, height) – in programming it is the same as a C struct).

For points we write:

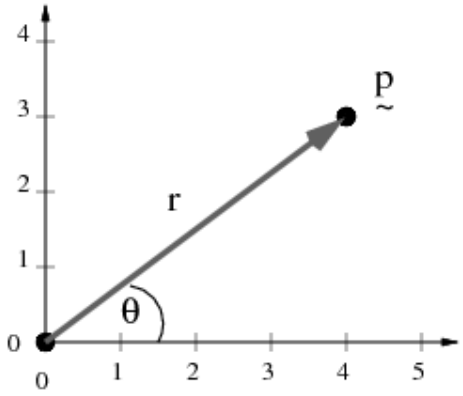
$$\underset{\sim}{\mathbf{p}} = (x, y, z)$$

where the first element in the tuple is the x coordinate, the second element is the y coordinate, and the third element is the z coordinate.

Although we are writing the point as a tuple, you should remember that it has a geometric meaning which relates the three components – the same is not true of most tuples (e.g., the tuple (name, age, address)).

Alternative representation

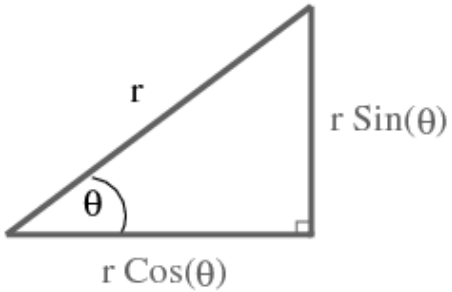
We use the descriptions above for representing a point – i.e., we record the "x coordinate" and the "y coordinate". Other representations are possible, and another popular way to represent a 2D point is to use **polar coordinates**, which is where we record the distance the point is from the origin, and the angle that the line from the origin to the point makes with the positive x axis.



For the point shown, the distance from the origin to the point, r, is 5, and the angle θ is 36.8699^o (or 0.6435 radians, since we use radians to measure angles - remember, a full circle has 360^o, and 2π rad).
We would write this in *polar form* as the tuple $(r, \theta) = (5, 0.6435)$.

We can move from 2D polar coordinates to 3D by: using a distance to measure the z coordinate – this gives *cylindrical coordinates*; or by taking the angle the point is from the xy-plane – this gives *spherical coordinates*.

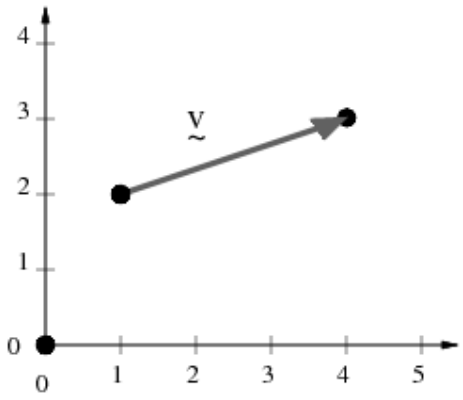
We will not be using polar coordinates, but it is useful to know that there are other ways of representing points, and that we write points as a *tuple* for convenience – this convenience can hide the underlying structure of what a point is, for example, if we rotate a point about the origin, then a tuple that is storing a point in polar coordinates will simply update the second, angular, coordinate, while a tuple storing a point using the x and y coordinates will need to update both coordinates.
It is also an opportunity to remind you of some trigonometry:



Recall: $x = r\ Cos(\theta)$ and $y = r\ Sin(\theta)$.

Vectors

A **vector** can be thought as an operation that will take us from one point to another point



In this example, we have a vector that takes us from the point $(1, 2)$ to the point $(4, 3)$.

This means we will take 3 steps in the x direction, and 1 step in the y direction, which gives:

$$\underset{\sim}{\mathbf{v}} = 3\ \underset{\sim}{\mathbf{i}} + 1\ \underset{\sim}{\mathbf{j}}$$

or in our compact tuple notation, the vector is $(3, 1)$.

Is it a point or a vector?

You will notice that we are writing the vectors exactly the same way as we are writing the points – which is not difficult to understand, because if you read how we defined the point, we said ... " from the origin ".

We normally call the tuple a vector, and leave its interpretation to the context in which we are using it.
If the vector is taken from the origin, then it represents a **point**, or a position in space, otherwise it represents a **direction** in space.

Vector arithmetic

Scaling vectors

We can multiply a vector, $\underset{\sim}{\mathbf{v}} = x\ \underset{\sim}{\mathbf{i}} + y\ \underset{\sim}{\mathbf{j}}$, by a scalar (number) k:

$$k\ \underset{\sim}{\mathbf{v}} \quad = \quad k\ (\ x\ \underset{\sim}{\mathbf{i}} + y\ \underset{\sim}{\mathbf{j}}) \quad = \quad kx\ \underset{\sim}{\mathbf{i}} + ky\ \underset{\sim}{\mathbf{j}}$$

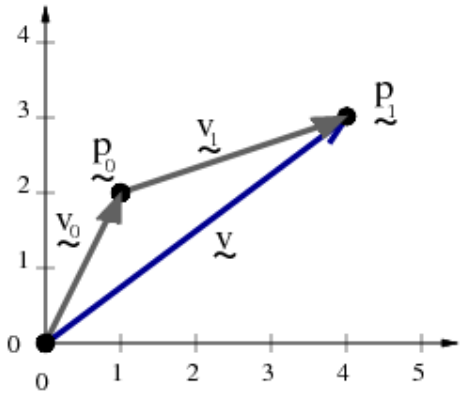
which we can write as:

$$k\ \underset{\sim}{\mathbf{v}} \quad = \quad k\ (x, y) \quad = \quad (kx, ky)$$

Multiplying a vector by a scalar changes the length of the vector, and if the scalar is negative, then the vector result will point in the opposite direction.

Vector addition

Since vectors are geometric objects, we can add them together to create new vectors:



Here we see we are adding the vectors $\underset{\sim}{\mathbf{v_0}}$ and $\underset{\sim}{\mathbf{v_1}}$ to get a new vector $\underset{\sim}{\mathbf{v}}$, where:

- $\underset{\sim}{\mathbf{v_0}}$ is a direction vector $(1, 2)$ which takes us from the origin $(0, 0)$ to the point $\underset{\sim}{\mathbf{p_0}}$ at $(1, 2)$;
- $\underset{\sim}{\mathbf{v_1}}$ is a direction vector $(3, 1)$ which takes us from the point $\underset{\sim}{\mathbf{p_0}}$ at $(1, 2)$ to the point $\underset{\sim}{\mathbf{p_1}}$ at $(4, 3)$;
- $\underset{\sim}{\mathbf{v}}$ is a direction vector from the origin $(0, 0)$ to the point $\underset{\sim}{\mathbf{p_1}}$ at $(4, 3)$.

If we have $\underset{\sim}{\mathbf{v}} = \underset{\sim}{\mathbf{v_0}} + \underset{\sim}{\mathbf{v_1}}$ where

$\underset{\sim}{\mathbf{v_0}} = x_0\ \underset{\sim}{\mathbf{i}} + y_0\ \underset{\sim}{\mathbf{j}}$

and

$\underset{\sim}{\mathbf{v_1}} = x_1\ \underset{\sim}{\mathbf{i}} + y_1\ \underset{\sim}{\mathbf{j}}$

then

$\underset{\sim}{\mathbf{v}} = (\ x_0\ \underset{\sim}{\mathbf{i}} + y_0\ \underset{\sim}{\mathbf{j}}) + (\ x_1\ \underset{\sim}{\mathbf{i}} + y_1\ \underset{\sim}{\mathbf{j}}) = (x_0 + x_1)\ \underset{\sim}{\mathbf{i}} + (y_0 + y_1)\ \underset{\sim}{\mathbf{j}}$

i.e., we add the corresponding components together.

Writing these as tuples gives:

$(x, y) = (x_0, y_0) + (x_1, y_1) = (x_0 + x_1, y_0 + y_1)$

So, from the earlier example:

$\underset{\sim}{\mathbf{v}} = \underset{\sim}{\mathbf{v_0}} + \underset{\sim}{\mathbf{v_1}} = (1, 2) + (3, 1) = (4, 3).$

Note that a direction vector added to a point will give us a new point, while adding two direction vectors gives us a new direction vector.

Vector subtraction

Just as we can add two vectors, we can also subtract two vectors. We do this by adding the **negative vector**, where the negative vector is same vector pointing in the other direction.

If $\underset{\sim}{\mathbf{v}} = (x, y)$ then $-\underset{\sim}{\mathbf{v}} = (-x, -y)$

So, from the earlier example:

$$\underset{\sim}{\mathbf{v}} - \underset{\sim}{\mathbf{v_1}} \quad = \quad \underset{\sim}{\mathbf{v}} + -\underset{\sim}{\mathbf{v_1}} \quad = \quad (4,3) + (-3,-1) \quad = \quad (1,2).$$

Writing these as tuples gives:

$$(x, y) = (x_0, y_0) - (x_1, y_1) = (x_0 - x_1, y_0 - y_1)$$

Note: The difference between two points results in a direction vector!

We use this process to generate direction vectors, e.g., $\underset{\sim}{\mathbf{v_1}} = \underset{\sim}{\mathbf{p_1}} - \underset{\sim}{\mathbf{p_0}}$

The results for vector addition and vector subtraction continue to work as expected in other dimensions (e.g., in 3D) so:

$(x_0, y_0, z_0) + (x_1, y_1, z_1) = (x_0 + x_1, y_0 + y_1, z_0 + z_1)$, and
$(x_0, y_0, z_0) - (x_1, y_1, z_1) = (x_0 - x_1, y_0 - y_1, z_0 - z_1)$.

Vector multiplication

We might guess that vector multiplication will be the same as addition and subtraction – i.e., we just get the result by multiplying the similar tuple elements together – but this is not the case.

Using $*$ to indicate an as yet undefined 'multiplication', we find that multiplying the following two 3D vectors:

$$\underset{\sim}{\mathbf{v_0}} = x_0\ \underset{\sim}{\mathbf{i}} + y_0\ \underset{\sim}{\mathbf{j}} + z_0\ \underset{\sim}{\mathbf{k}}, \quad \text{and} \quad \underset{\sim}{\mathbf{v_1}} = x_1\ \underset{\sim}{\mathbf{i}} + y_1\ \underset{\sim}{\mathbf{j}} + z_1\ \underset{\sim}{\mathbf{k}}$$

term by term, results in the rather complicated looking:

$$\begin{aligned}
\underset{\sim}{\mathbf{v_0}} * \underset{\sim}{\mathbf{v_1}} = \quad & x_0\ x_1\ \underset{\sim}{\mathbf{i}} * \underset{\sim}{\mathbf{i}} + x_0\ y_1\ \underset{\sim}{\mathbf{i}} * \underset{\sim}{\mathbf{j}} + x_0\ z_1\ \underset{\sim}{\mathbf{i}} * \underset{\sim}{\mathbf{k}} \\
& + y_0\ x_1\ \underset{\sim}{\mathbf{j}} * \underset{\sim}{\mathbf{i}} + y_0\ y_1\ \underset{\sim}{\mathbf{j}} * \underset{\sim}{\mathbf{j}} + y_0\ z_1\ \underset{\sim}{\mathbf{j}} * \underset{\sim}{\mathbf{k}} \\
& + z_0\ x_1\ \underset{\sim}{\mathbf{k}} * \underset{\sim}{\mathbf{i}} + z_0\ y_1\ \underset{\sim}{\mathbf{k}} * \underset{\sim}{\mathbf{j}} + z_0\ z_1\ \underset{\sim}{\mathbf{k}} * \underset{\sim}{\mathbf{k}}
\end{aligned}$$

The x, y, and z values are just numbers (scalars), so we know how to multiply them – but how do we "multiply" the $\underset{\sim}{\mathbf{i}}$, $\underset{\sim}{\mathbf{j}}$, and $\underset{\sim}{\mathbf{k}}$'s together?

What does this actually "mean"?

It turns out that there are two useful definitions for vector multiplication, and both of these methods produce a number of interesting and useful results. These are the **dot product** and the **cross product**.

The dot product

We denote that we are taking the **dot product** of two vectors, $\underset{\sim}{\mathbf{v_0}}$ and $\underset{\sim}{\mathbf{v_1}}$, by writing $\underset{\sim}{\mathbf{v_0}} \cdot \underset{\sim}{\mathbf{v_1}}$

where the dot product provides a **projection** of one vector on another. This statement means $\underset{\sim}{\mathbf{v_0}} \cdot \underset{\sim}{\mathbf{i}} = x_0$, $\underset{\sim}{\mathbf{v_0}} \cdot \underset{\sim}{\mathbf{j}} = y_0$, $\underset{\sim}{\mathbf{v_0}} \cdot \underset{\sim}{\mathbf{k}} = z_0$.

In other words, if we take the dot product of a vector with one of the "step in one direction" vectors, then we get the component of the vector that is in that direction.

This idea, that we are taking the projection of one vector on another, gives us:

$$\underset{\sim}{\mathbf{i}} \cdot \underset{\sim}{\mathbf{i}} = 1 \quad \underset{\sim}{\mathbf{j}} \cdot \underset{\sim}{\mathbf{j}} = 1 \quad \underset{\sim}{\mathbf{k}} \cdot \underset{\sim}{\mathbf{k}} = 1$$

because each of these vectors projects totally onto itself.
Otherwise, the dot product will be zero.

Replacing "*" by $\cdot$ in the equation for vector multiplication gives:

$$\begin{aligned}
\underset{\sim}{\mathbf{v_0}} \cdot \underset{\sim}{\mathbf{v_1}} = \quad & x_0\ x_1\ \underset{\sim}{\mathbf{i}} \cdot \underset{\sim}{\mathbf{i}} + x_0\ y_1\ \underset{\sim}{\mathbf{i}} \cdot \underset{\sim}{\mathbf{j}} + x_0\ z_1\ \underset{\sim}{\mathbf{i}} \cdot \underset{\sim}{\mathbf{k}} \\
& + y_0\ x_1\ \underset{\sim}{\mathbf{j}} \cdot \underset{\sim}{\mathbf{i}} + y_0\ y_1\ \underset{\sim}{\mathbf{j}} \cdot \underset{\sim}{\mathbf{j}} + y_0\ z_1\ \underset{\sim}{\mathbf{j}} \cdot \underset{\sim}{\mathbf{k}} \\
& + z_0\ x_1\ \underset{\sim}{\mathbf{k}} \cdot \underset{\sim}{\mathbf{i}} + z_0\ y_1\ \underset{\sim}{\mathbf{k}} \cdot \underset{\sim}{\mathbf{j}} + z_0\ z_1\ \underset{\sim}{\mathbf{k}} \cdot \underset{\sim}{\mathbf{k}} \\
= \quad & x_0\ x_1\ 1 + x_0\ y_1\ 0 + x_0\ z_1\ 0 \\
& + y_0\ x_1\ 0 + y_0\ y_1\ 1 + y_0\ z_1\ 0 \\
& + z_0\ x_1\ 0 + z_0\ y_1\ 0 + z_0\ z_1\ 1 \\
= \quad & x_0\ x_1 + y_0\ y_1 + z_0\ z_1
\end{aligned}$$

So, we can calculate the dot product of two vectors by taking the sum of the products of each of the elements in the tuple

$$(x_0, y_0, z_0) \cdot (x_1, y_1, z_1) = x_0\ x_1 + y_0\ y_1 + z_0\ z_1 \tag{1}$$

It is important to notice that the result of the dot product is a scalar (i.e., just a number).
If we write $|\underset{\sim}{\mathbf{v}}|$ to denote the **length** of the vector $\underset{\sim}{\mathbf{v}}$, then an important result for the dot product is

$$\underset{\sim}{\mathbf{v_0}} \cdot \underset{\sim}{\mathbf{v_1}} = |\underset{\sim}{\mathbf{v_0}}||\underset{\sim}{\mathbf{v_1}}|\ \ Cos(\theta) \tag{2}$$

where θ is the angle between the two vectors.

If the two vectors are **orthogonal** (i.e., at right angles, or perpendicular), then the angle between them is 90^o and the cosine will be zero – which means the dot product of two vectors that are orthogonal is zero (as none of the vectors will be projected onto the other vector).
For Example: $(1, 2, 3) \cdot (3, -3, 1) = 0$

Unit vectors

Since the angle between a vector and itself is zero

$\underset{\sim}{\mathbf{v}} \cdot \underset{\sim}{\mathbf{v}} = |\underset{\sim}{\mathbf{v}}||\underset{\sim}{\mathbf{v}}| \; Cos(0) = |\underset{\sim}{\mathbf{v}}|^2$

which provides us with a way to calulate the length of a vector from the dot product!

$$|\underset{\sim}{\mathbf{v}}| = \sqrt{\underset{\sim}{\mathbf{v}} \cdot \underset{\sim}{\mathbf{v}}} \tag{3}$$

If the length of a vector $\underset{\sim}{\mathbf{v}}$ is equal to 1, then we call that vector a **unit vector**, and denote that it is a unit vector by writing a circumflex, $\widehat{\ }$ above the vector: $\underset{\sim}{\widehat{\mathbf{v}}}$.

We can easily turn any (non-zero) vector into a unit vector by dividing the vector by its length

$$\underset{\sim}{\widehat{\mathbf{v}}} = \frac{\underset{\sim}{\mathbf{v}}}{|\underset{\sim}{\mathbf{v}}|}$$

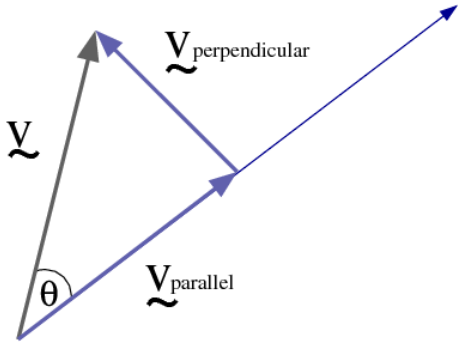
If we use unit vectors, then the equation for the dot product simplifies to

$$\underset{\sim}{\widehat{\mathbf{v_0}}} \cdot \underset{\sim}{\widehat{\mathbf{v_1}}} = Cos(\theta)$$

Component vectors

Given a vector, we can express it in terms of another vector. We are sometimes interested in knowing how much of a vector is parallel to another vector (this is called the **projection** of the first vector on the other vector), and how much is perpendicular to the other vector.

$$\underset{\sim}{\mathbf{v}} = \underset{\sim_{\parallel}}{\mathbf{v}} + \underset{\sim_{\perp}}{\mathbf{v}}$$



If we recall the results for a right angled triangle, then $|\underset{\sim_{\parallel}}{\mathbf{v}}| = |\underset{\sim}{\mathbf{v}}| \; Cos(\theta) \quad \ldots$ which means we can use the dot product of the two vectors to get the component of the vector that is parallel. As a result, we do not have to explicitly calculate the angle, θ.

Since $\underset{\sim}{\mathbf{v}} = \underset{\sim_{\parallel}}{\mathbf{v}} + \underset{\sim_{\perp}}{\mathbf{v}}$ we can use vector subtraction to obtain $\underset{\sim_{\perp}}{\mathbf{v}}$ from $\underset{\sim_{\parallel}}{\mathbf{v}}$.

The cross product

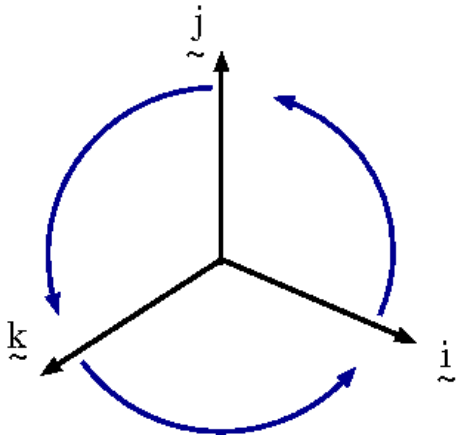
Before we can start talking about the "cross product", we need to clarify what we mean by a three dimensional coordinate system. It is natural to think of having our x and y axes as lying on the plane surface – but in which direction does the z axis lie?

Does it lie pointing 'out' of the screen?

Does it lie pointing 'into' the screen?

Because either choice is possible, both possibilities are used in different fields (e.g., engineering, computer graphics), and sometimes within the same area of study.

Unless otherwise stated, we shall be using the first option – which is called a **right handed coordinate system**, because if you use your right hand, and line up your thumb with the x axis, your index finger with the y axis, then bending your middle finger will point along the z axis.
Alternatively: point your fingers in the direction of the x axis, with the palm of your hand pointing in the direction of the y axis, then curling your fingers towards y will show the thumb pointing in the direction of the z axis.
If you do the same thing with your left hand, you will get a left handed coordinate system.



We can take our three basis vectors, $\underset{\sim}{\mathbf{i}}$, $\underset{\sim}{\mathbf{j}}$, and $\underset{\sim}{\mathbf{k}}$ as shown, and by going in an anticlockwise direction around the axes, we can wrap around the axes to create the following rules for our 'new multiplication', $\times$:

$$\underset{\sim}{\mathbf{i}} \times \underset{\sim}{\mathbf{j}} = \underset{\sim}{\mathbf{k}}, \quad \underset{\sim}{\mathbf{j}} \times \underset{\sim}{\mathbf{k}} = \underset{\sim}{\mathbf{i}}, \quad \underset{\sim}{\mathbf{k}} \times \underset{\sim}{\mathbf{i}} = \underset{\sim}{\mathbf{j}}.$$

If we go clockwise around the axes, we negate the result:

$$\underset{\sim}{\mathbf{j}} \times \underset{\sim}{\mathbf{i}} = -\underset{\sim}{\mathbf{k}}, \quad \underset{\sim}{\mathbf{k}} \times \underset{\sim}{\mathbf{j}} = -\underset{\sim}{\mathbf{i}}, \quad \underset{\sim}{\mathbf{i}} \times \underset{\sim}{\mathbf{k}} = -\underset{\sim}{\mathbf{j}}.$$

Otherwise the result will be zero, i.e.,

$$\underset{\sim}{\mathbf{i}} \times \underset{\sim}{\mathbf{i}} = 0, \quad \underset{\sim}{\mathbf{j}} \times \underset{\sim}{\mathbf{j}} = 0, \quad \underset{\sim}{\mathbf{k}} \times \underset{\sim}{\mathbf{k}} = 0.$$

Replacing "*" by $\times$ in the equation for vector multiplication gives:

$$\begin{aligned}
\underset{\sim}{\mathbf{v_0}} \times \underset{\sim}{\mathbf{v_1}} = \quad & x_0\, x_1\, \underset{\sim}{\mathbf{i}} \times \underset{\sim}{\mathbf{i}} + x_0\, y_1\, \underset{\sim}{\mathbf{i}} \times \underset{\sim}{\mathbf{j}} + x_0\, z_1\, \underset{\sim}{\mathbf{i}} \times \underset{\sim}{\mathbf{k}} \\
& + y_0\, x_1\, \underset{\sim}{\mathbf{j}} \times \underset{\sim}{\mathbf{i}} + y_0\, y_1\, \underset{\sim}{\mathbf{j}} \times \underset{\sim}{\mathbf{j}} + y_0\, z_1\, \underset{\sim}{\mathbf{j}} \times \underset{\sim}{\mathbf{k}} \\
& + z_0\, x_1\, \underset{\sim}{\mathbf{k}} \times \underset{\sim}{\mathbf{i}} + z_0\, y_1\, \underset{\sim}{\mathbf{k}} \times \underset{\sim}{\mathbf{j}} + z_0\, z_1\, \underset{\sim}{\mathbf{k}} \times \underset{\sim}{\mathbf{k}} \\
= \quad & x_0\, x_1\, 0 + x_0\, y_1\, \underset{\sim}{\mathbf{k}} + x_0\, z_1\, (-\underset{\sim}{\mathbf{j}}) \\
& + y_0\, x_1\, (-\underset{\sim}{\mathbf{k}}) + y_0\, y_1\, 0 + y_0\, z_1\, \underset{\sim}{\mathbf{i}} \\
& + z_0\, x_1\, \underset{\sim}{\mathbf{j}} + z_0\, y_1\, (-\underset{\sim}{\mathbf{i}}) + z_0\, z_1\, 0 \\
= \quad & (y_0\, z_1 - z_0\, y_1)\, \underset{\sim}{\mathbf{i}} + (z_0\, x_1 - x_0\, z_1)\, \underset{\sim}{\mathbf{j}} + (x_0\, y_1 - y_0\, x_1)\, \underset{\sim}{\mathbf{k}}
\end{aligned}$$

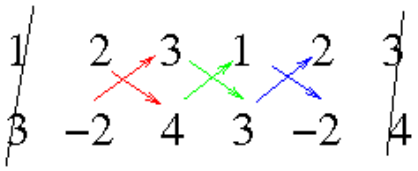
So, we can calculate the cross product of two vectors as follows:

$$(x_0, y_0, z_0) \times (x_1, y_1, z_1) \;=\; (y_0\, z_1 - z_0\, y_1,\; z_0\, x_1 - x_0\, z_1,\; x_0\, y_1 - y_0\, x_1) \quad (4)$$

Probably the easiest way to 'remember' how to do this is:

- write down the values in the first vector twice
- underneath write out the values in the second vector, also twice.
- cross out the first column, and the last column
- step though the remainder doing the following:
  multiply the top left value with next value diagonally on the bottom;
  and subtract the bottom left value multiplied by the next value which is diagonally on the top.

For example: $(1, 2, 3) \times (3, -2, 4)$



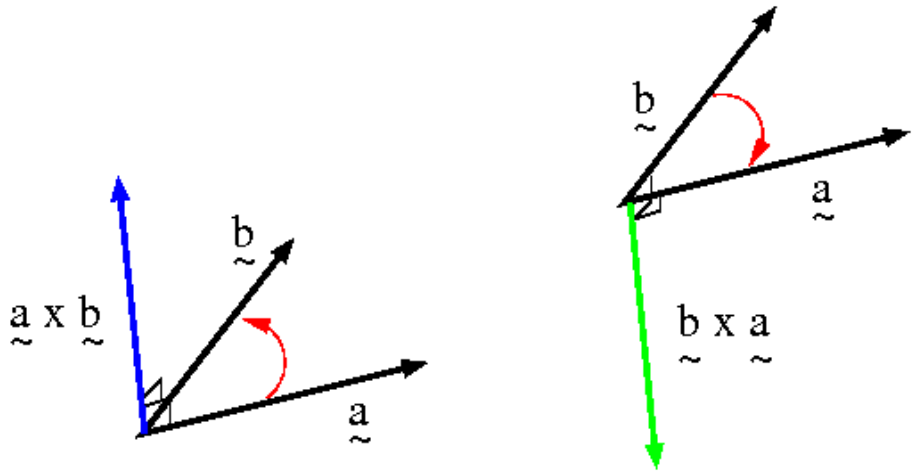
gives the result $(2*4-(-2)*3,\; 3*3-4*1,\; 1*(-2)-3*2) \quad = \quad (14,\; 5,\; -8)$

You will see that, while the dot product gives us a scalar result, the cross product gives us a new vector. It turns out, this new vector is at right angles to both of the original vectors! In other words,

$$\underset{\sim}{\mathbf{a}} \times \underset{\sim}{\mathbf{b}} \,.\, \underset{\sim}{\mathbf{a}} \quad = \quad 0 \quad = \quad \underset{\sim}{\mathbf{a}} \times \underset{\sim}{\mathbf{b}} \,.\, \underset{\sim}{\mathbf{b}}$$

It is also important to note that the direction of the result depends on the order in which you take the cross product, as

$$\underset{\sim}{\mathbf{a}} \times \underset{\sim}{\mathbf{b}} \quad = \quad -\underset{\sim}{\mathbf{b}} \times \underset{\sim}{\mathbf{a}}$$



Here we see that the direction of the result depends on the order of the vectors in the cross product – you should see that the direction of the result depends on the right hand rule.

There is an interesting result for the length of a cross product

$$|\underset{\sim}{\mathbf{a}} \times \underset{\sim}{\mathbf{b}}| \quad = \quad |\underset{\sim}{\mathbf{a}}||\underset{\sim}{\mathbf{b}}|Sin(\theta)$$

where θ is the angle between the two vectors – but we will not be using this result in the course.

Lines

Loosely speaking, a line is a collection of points that satisfy the condition that all of the points lie on the line. In other words, we may think of a line as

$$\{\underset{\sim}{\mathbf{p}} \quad | \quad \underset{\sim}{\mathbf{p}} \text{ lies on the line.}\}$$

With the understanding that we are talking about a collection of points, we give the following methods for representing a line in space

Explicit line

This is the 'standard' way lines are described:

$$y = mx + c$$

where m is the *slope* of the line, and c is where the line crosses the y axis (when $x = 0$).

Given any value for x we can easily calculate the y value, and the line is the collection of (x, y) points that satisfy the equation of the line.

Implicit line

Rather than having an equation from which we can explicitly calculate y for a given x value, we have a relationship that the points x and y must satisfy:

$$ay + bx = c$$

It is easy to rearrange the implicit equation to represent the line explicitly.

Parametric line

In the previous two representations, we can arbitrarily select values of x and y to add to our collection of points to make up the line. Of course, in practice, we will select a reasonable sequence of x values from which to construct the line.

We can do this by adding a parameter to x, so, in the simplest case:

$$x(t) = t \quad \text{where} \quad \mathrm{t} \in \mathbb{R}$$

Here you can think of t as representing "time", so when $t = 0$ we are at one point on the line, and when $t = 1$ we are at another point on the line.

In general, we can specify a parametric line as:

$$x(t) = a + bt \quad \text{where} \quad \mathrm{t} \in \mathbb{R}$$
$$y(t) = c + dt \quad \text{where} \quad \mathrm{t} \in \mathbb{R}$$

which allows us to write that points will lie on the line if they satisfy:

$$(x(t),\ y(t)) = (a + bt,\ c + dt)$$

or

$$\underset{\sim}{\mathbf{p}}(t) \quad = \quad \underset{\sim}{\mathbf{u}} + \underset{\sim}{\mathbf{v}}t \quad \text{where} \quad \underset{\sim}{\mathbf{u}} = (a, c) \quad \text{and} \quad \underset{\sim}{\mathbf{v}} = (b, d) \tag{5}$$

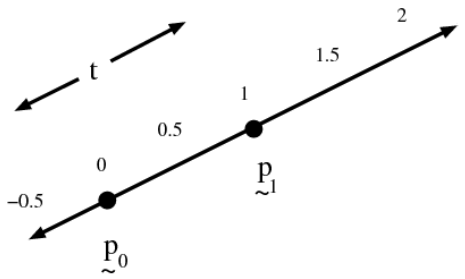
Parametric line between two points

A common problem involving lines is to find a line that will go through two points, $\underset{\sim}{\mathbf{p_0}}$ and $\underset{\sim}{\mathbf{p_1}}$. We can easily do this using the parametric equation for a line, by noting that a line is a *linear interpolation* of the two points:

$$\underset{\sim}{\mathbf{p}}(t) \quad = \quad (1 \; - \; t) \; \underset{\sim}{\mathbf{p_0}} \quad + \quad t \; \underset{\sim}{\mathbf{p_1}} \tag{6}$$

This gives us:

$$\begin{aligned} \underset{\sim}{\mathbf{p}}(0) &= \underset{\sim}{\mathbf{p_0}} \\ \underset{\sim}{\mathbf{p}}(\tfrac{1}{2}) &= \tfrac{1}{2}(\underset{\sim}{\mathbf{p_0}} + \underset{\sim}{\mathbf{p_1}}) \\ \underset{\sim}{\mathbf{p}}(1) &= \underset{\sim}{\mathbf{p_1}} \end{aligned}$$



So, we can imagine that at "time" $t = 0$ we are at the point $\underset{\sim}{\mathbf{p_0}}$, and as "time" increases we move along the line so that when $t = 1$ we are at the point $\underset{\sim}{\mathbf{p_1}}$. As t keeps increasing, we travel along line and we talk about the number of *steps* we have taken from $\underset{\sim}{\mathbf{p_0}}$, for example when $t = 2$, we have taken 2 steps along the line.

We can rewrite this equation as

$$\underset{\sim}{\mathbf{p}}(t) \quad = \quad \underset{\sim}{\mathbf{u}} + \underset{\sim}{\mathbf{v}}t$$

where

$$\underset{\sim}{\mathbf{u}} \quad = \quad \underset{\sim}{\mathbf{p_0}} \quad \text{and} \quad \underset{\sim}{\mathbf{v}} \quad = \quad \underset{\sim}{\mathbf{p_1}} \; - \; \underset{\sim}{\mathbf{p_0}}$$

So we see

$\underset{\sim}{\mathbf{u}}$ represents a *point* – a vector (u_x, u_y, u_z) for a position on the line,

and $\underset{\sim}{\mathbf{v}}$ represents a *direction* – a vector (v_x, v_y, v_z) for the direction of the line.

We can place limits on the values of t to define:

a ray is a line where $t \geq t_{min}$, typically $t_{min} = 0$
A ray is a line that travels to infinity in only one direction.

a line segment is a line where $t_{min} \leq t \leq t_{max}$, typically $0 \leq t \leq 1$.
A line segment is the part of the line that is between the two points.

Matrices

Suppose we had two lines, and we wanted to know where they intersect

$$\begin{aligned} 3x - 4y &= 1 \\ 2x + 2y &= 3 \end{aligned}$$

This gives us a set of equations where we want to find value(s) for x and y that satisfy the equations at the same time, *simultaneous equations*.

There are several techniques for solving simultaneous equations, and one way is to rewrite the equations as a **matrix** equation. Just as we are representing a vector as a tuple, we can represent these equations in terms of matrices:

$$\begin{bmatrix} 3 & -4 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Vector representation

We have had some different ways to represent vectors, and we notice here that we have another important representation – the vector tuple (x, y) appears in the above equation as a **column vector**, as does the vector tuple $(1, 3)$.

We will freely swap notation between tuples and column vectors during the course.

Transpose matrices

Just as there are two different 3D coordinate systems (the right handed coordinate system, and the left handed coordinate system), there are two ways we could have represented the equations above. The other way is:

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 3 & 2 \\ -4 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 3 \end{bmatrix}$$

You will note that we have swapped the rows and columns in the matrix, A. This process of swapping the rows and columns of A is called forming the **transpose** of a matrix, A^T.

The transpose of a product of matrices is the product of transposes *taken in reverse order*:

$$(A\ B\ C)^T = C^T\ B^T\ A^T$$

So, you will notice, the alternative way for writing the matrix equation is simply the transpose of the way that we shall write them.

We mention this here, because there are several resources that use these transposed equations and matrices – they will talk about representing a vector as a row matrix, and will do matrix multiplication on right of the vector, rather than on the left.
(This transposed view of matrices is actually implemented in OpenGL !)

Matrix arithmetic

There are many interesting things that we can use matrices for, however, for graphics we will only be interested in a few results – and our work is simplified because we shall only be dealing with square matrices, and column vectors.

Matrix addition, subtraction, and scaling

When we add/subtract two matrices, we add/subtract each element in the same corresponding position in each matrix (using a 2×2 matrix as a simple example):

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} x & y \\ z & w \end{bmatrix} = \begin{bmatrix} a+x & b+y \\ c+z & d+w \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} x & y \\ z & w \end{bmatrix} = \begin{bmatrix} a-x & b-y \\ c-z & d-w \end{bmatrix}$$

We can also *scale* a matrix by multiplying all the elements by a scalar:

$$k \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} ka & kb \\ kc & kd \end{bmatrix}$$

Matrix multiplication

When we introduced matrices we used the multiplication of a matrix and a column vector, but did not describe what we were doing. When you multiply two matrices you multiply the elements in the rows of the first matrix by the elements in the column of the second matrix, and sum those values to create a new element in the solution. For example:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x & y \\ z & w \end{bmatrix} = \begin{bmatrix} ax+bz & ay+bw \\ cx+dz & cy+dw \end{bmatrix}$$

The result of multiplying a matrix of size $n\times m$ by a matrix of size $p\times q$ requires that $m = p$ (i.e., there are the same number of columns in the first matrix, as there are rows in the second matrix). The resultant matrix will be of size $n\times q$. *Remember: For matrix multiplication, it's always **rows** times **columns***

So, the result of multiplying two square matrices will be another square matrix, and multiplying a square matrix by a column vector will give another column vector.

There is a connection between matrix multiplication and the vector dot product:

$$\begin{aligned} \underset{\sim}{\mathbf{a}} \cdot \underset{\sim}{\mathbf{b}} &= (a_x, a_y, a_z) \cdot (b_x, b_y, b_z) \\ &= a_x b_x + a_y b_y + a_z b_z \\ &= \begin{bmatrix} a_x & a_y & a_z \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \\ &= \underset{\sim}{\mathbf{a}}^T \underset{\sim}{\mathbf{b}} \end{aligned}$$

You will notice this result mixes two different representations of a vector.

If we have two numbers, a and b, then we know that we can add and multiply them in any order (the *commutative property*):

$a + b = b + a \qquad a * b = b * a$

and while vector and matrix addition is commutative

$\underset{\sim}{\mathbf{a}} + \underset{\sim}{\mathbf{b}} = \underset{\sim}{\mathbf{b}} + \underset{\sim}{\mathbf{a}}$ and $A + B = B + A$

we have already seen that vector multiplication is more complicated, where the dot product is commutative

$\underset{\sim}{\mathbf{a}} \cdot \underset{\sim}{\mathbf{b}} = \underset{\sim}{\mathbf{b}} \cdot \underset{\sim}{\mathbf{a}}$

but the cross product is **not** commutative, since changing the order that we multiply the two vectors changes the direction of the resulting vector

$\underset{\sim}{\mathbf{a}} \times \underset{\sim}{\mathbf{b}} = - \underset{\sim}{\mathbf{b}} \times \underset{\sim}{\mathbf{a}}$

Matrix multiplication, for any given A and B, is **not commutative**, so:

$$A\ B \quad \neq \quad B\ A \tag{7}$$

Matrix multiplication is **associative**, which means we can multiply a sequence of matrices together by working on any subsequence. This is an important result which we will make use of later in the course.

$$(A\ B)\ C \quad = \quad A\ (B\ C) \quad = \quad A\ B\ C \tag{8}$$

The inverse matrix

Because we will be using relatively nice square matrices, we will be able to use the **inverse** of a matrix, A. The inverse, A^{-1}, has the property that

$$A\ A^{-1} \quad = \quad I \quad = \quad A^{-1}\ A$$

where I is the **identity matrix**, which is a matrix with all elements being 0, except for the diagonal values which are 1.

While it is easy to find the inverse of a 2×2 matrix, it is not as easy to find the inverse of a 3×3 matrix, and as the size of the matrices increases the problem becomes exponentially more difficult. For this course, we will want to find the inverse of 4×4 matrices (true!), and while there are algorithms for finding the inverse of an arbitrary 4×4 matrix, we will be using matrices that are not truely arbitrary as they will be constructed from a number of simpler matrices – each of which has its own simple inverse.

This means we can use the following result for inverses:

if $M = D\ C\ B\ A$ then $M^{-1} = A^{-1}\ B^{-1}\ C^{-1}\ D^{-1}$

You can test this by multiplying M and M^{-1} together, and seeing that the simple inverses are all cancelled out.

Summary of vector results

We can represent a vector, $\underset{\sim}{\mathbf{v}}$ with components x, y, and z, as:

- $x\underset{\sim}{\mathbf{i}} + y\underset{\sim}{\mathbf{j}} + z\underset{\sim}{\mathbf{k}}$
- (x, y, z)
- $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$

Given a scalar, k, and vectors $\underset{\sim}{\mathbf{u}} = (a, b, c)$ and $\underset{\sim}{\mathbf{v}} = (x, y, z)$, we have:

- $k\underset{\sim}{\mathbf{u}} = k(a, b, c) = (ka, kb, kc)$
- $\underset{\sim}{\mathbf{u}} + \underset{\sim}{\mathbf{v}} = (a, b, c) + (x, y, z) = (a + x, b + y, c + z)$
- $\underset{\sim}{\mathbf{u}} - \underset{\sim}{\mathbf{v}} = (a, b, c) - (x, y, z) = (a - x, b - y, c - z)$
- $\underset{\sim}{\mathbf{u}} \cdot \underset{\sim}{\mathbf{v}} = (a, b, c) \cdot (x, y, z) = ax + by + cz$
- $\underset{\sim}{\mathbf{u}} \times \underset{\sim}{\mathbf{v}} = (a, b, c) \times (x, y, z) = (bz - cy,\ cx - az,\ ay - bx)$

The dot product has many important results:

- $\underset{\sim}{\mathbf{u}} \cdot \underset{\sim}{\mathbf{v}} = \underset{\sim}{\mathbf{v}} \cdot \underset{\sim}{\mathbf{u}}$
- $\underset{\sim}{\mathbf{u}} \cdot \underset{\sim}{\mathbf{v}} = |\underset{\sim}{\mathbf{u}}||\underset{\sim}{\mathbf{v}}|Cos(\theta)$ relates to projections of vectors
- $|\underset{\sim}{\mathbf{u}}| = \sqrt{\underset{\sim}{\mathbf{u}} \cdot \underset{\sim}{\mathbf{u}}}$ the length of a vector
- $\underset{\sim}{\widehat{\mathbf{u}}} = \frac{\underset{\sim}{\mathbf{u}}}{|\underset{\sim}{\mathbf{u}}|}$ a unit vector (length of 1)

The cross product of two vectors gives us a new vector which is *orthogonal* to those two vectors. We determine the direction of the resulting vector by using the right hand rule. Note:

$$\underset{\sim}{\mathbf{u}} \times \underset{\sim}{\mathbf{v}} = -\underset{\sim}{\mathbf{v}} \times \underset{\sim}{\mathbf{u}}$$

It is convenient to represent a line by using the vector parametic equation:

$$\underset{\sim}{\mathbf{p}}(t) = \underset{\sim}{\mathbf{u}} + \underset{\sim}{\mathbf{v}}t$$

where the vector $\underset{\sim}{\mathbf{u}}$ is the point on the line where $t = 0$, and $\underset{\sim}{\mathbf{v}}$ is the direction in which the line is going.

If we multiply a vector, $\underset{\sim}{\mathbf{v}}$, by a matrix, M, then the result is a new vector, $\underset{\sim}{\mathbf{v}'}$:

$$\underset{\sim}{\mathbf{v}'} = M\ \underset{\sim}{\mathbf{v}}$$