The Maths That Makes Movies: Matrices

By: Arnav Ajay

Have you ever wondered how SpongeBob fries Krabby Patties? How Donald Duck waddles? How Keanu Reeves managed to dodge bullets in the Matrix? What a cartoon chef, an animated duck, and a Hollywood action star have in common is, frankly, more interesting than any of them. Ironically, the answer is my favourite part of linear algebra: matrices.

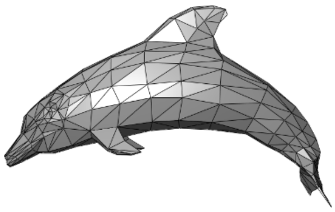
1 Pointless? Far From It

Not to point out the obvious, but before we can move a continuous shape, we must first convert it into form that a computer can process. Real-world objects consist of infinitely many points which cannot be represented or manipulated by a digital system.

Getting straight to the point, we approximate through the process known as discretisation, where a finite number of points are selected to represent the surface of an object. These points are then joined together to form a mesh, which typically consists of triangular faces to approximate the original shape.

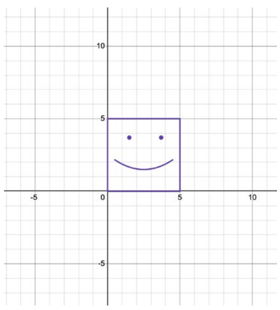
For example, the curved dolphin can be reduced to a collection of points, joined together to form a triangular mesh. Each of these points is then rewritten as a vector.

The real point of this all is to allow us to conduct matrix multiplication (if you're not sure what this is, I recommend you search it up) on every individual point. This is the process that creates artificial movement. There are three main movements to consider: translation (moving it in a direction), enlargement (making it larger or smaller), and rotation.

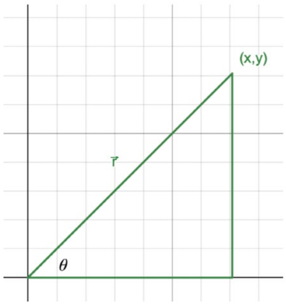


2 A Sine of the Times

The rotational matrix is my favourite simply because of its derivation. Consider this crudely drawn smiley face, which needs to be rotated. But how?

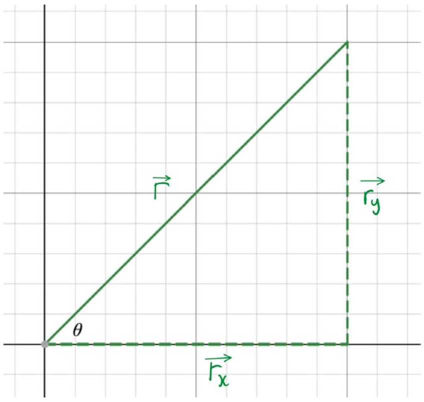


First consider this diagram:



Consider vector $\boldsymbol{r}$ where $\boldsymbol{r} = \binom{x}{y}$

This vector can be split into its x-component, where $\boldsymbol{r_x} = \binom{x}{0}$, and its y-component, where $\boldsymbol{r_y} = \binom{0}{y}$. We can find the magnitude (length) of each of these vectors by taking the square root of the sum of the squares of its components. This is denoted through the absolute value.

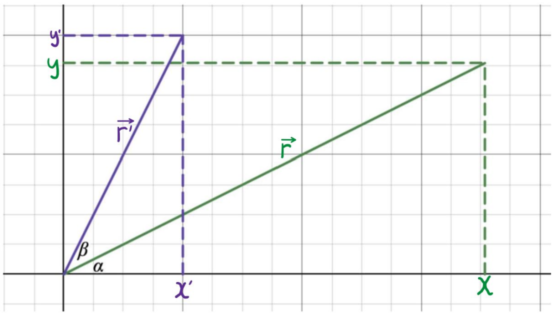


Using the trigonometric rules, we are all well versed in (SOH CAH TOA), we know that:

$\cos\theta = \frac{adjacent}{hypotenuse} = \frac{|\boldsymbol{r_x}|}{|\boldsymbol{r}|}$, which can be rearranged: $|\boldsymbol{r_x}| = \cos\theta \cdot |\boldsymbol{r}|$

$\sin\theta = \frac{opposite}{hypotenuse} = \frac{|\boldsymbol{r_y}|}{|\boldsymbol{r}|}$, which can be rearranged: $|\boldsymbol{r_y}| = \sin\theta \cdot \ |\boldsymbol{r}|$

Now let's consider this diagram:



Here we rotate the vector $\boldsymbol{r}$ by β to get vector $\boldsymbol{r}'$(which has the same magnitude as $\boldsymbol{r}$).

We can write the x and y components of $\boldsymbol{r}$ through our working out above, so:

$x = \cos\alpha \cdot |\boldsymbol{r}|$

$y = \sin\alpha \cdot |\boldsymbol{r}|$

We can also find the x and y components of $\boldsymbol{r}'$:

$x' = \cos\,(\alpha + \beta) \cdot |\boldsymbol{r}'|$

$y' = \sin\,(\alpha + \beta) \cdot |\boldsymbol{r}'|$

Now we need find an expression for (x', y') as a function of our initial (x, y).

To do this we use the trigonometric identities:

$\cos(\alpha + \beta) \equiv \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$

$\sin(\alpha + \beta) \equiv \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta)$

Since $\boldsymbol{r}'$ has the same magnitude as $\boldsymbol{r}$, we know:

$x' = \cos\,(\alpha + \beta) \cdot |\boldsymbol{r}|$

$x' = |\boldsymbol{r}|\,(\cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta))$

$x' = |\boldsymbol{r}| \cdot \cos(\alpha)\cos(\beta) - |\boldsymbol{r}| \cdot \sin(\alpha)\sin(\beta))$

And:

$y' = \sin\,(\alpha + \beta) \cdot |\boldsymbol{r}|$

$y' = |\boldsymbol{r}|\,(\sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta))$

$y' = |\boldsymbol{r}| \cdot \sin(\alpha)\cos(\beta) + |\boldsymbol{r}| \cdot \cos(\alpha)\sin(\beta))$

We can now substitute our values for x and y from earlier, to get:

$x' = x \cdot \cos(\beta) - y \cdot \sin(\beta))$

$y' = \mathrm{y} \cdot \cos(\beta) + x \cdot \sin(\beta)$, so $y' = x \cdot \sin(\beta) + \mathrm{y} \cdot \cos(\beta)$

We can write this in matrix form:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

And thus, we have our matrix to rotate by angle β (anticlockwise)!

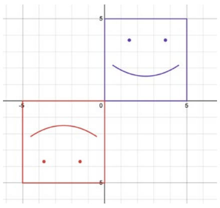
So, we can rotate out previous smiley face by any angle β we want. Take for example a rotation by 180°. We apply the rotational matrix to all the points on the object being rotated:

So, for the smiley face, we do:

$\begin{pmatrix}\cos 180 & -\sin 180\\ \sin 180 & \cos 180\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}$, for all the points.

$$\begin{pmatrix}x'\\ y'\end{pmatrix} = \begin{pmatrix}\cos 180 & -\sin 180\\ \sin 180 & \cos 180\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} = \begin{pmatrix}-1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix} = \begin{pmatrix}-x\\ -y\end{pmatrix}$$

This gives us:



And now, he is mathematically upside down and, presumably, unhappy about it.

Because apparently two dimensions wasn't complicated enough, this can be extended to three dimensions, as is used in CGI for example. However, in 3D space rotations occur about an axis rather than a single point.

For example, when rotating in the z-axis, the x and y coordinates transform in exactly the same way as in the two-dimensional case, while the z-coordinate remains unchanged.

So, we can construct a three-dimensional rotation matrix by embedding the two-dimensional rotation within a larger matrix. For example, rotation around the z-axis is:

$$\begin{pmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{pmatrix}$$

3 Go Big or Go Home

Enlargement is how movies make objects smaller or bigger on screen. Take:

$$\boldsymbol{r} = \begin{pmatrix}x\\ y\end{pmatrix}$$

If we want to enlarge by factor k,

The new x is kx, the new y is ky. So:

$$\boldsymbol{r}' = \begin{pmatrix}kx\\ ky\end{pmatrix}$$

So, consider, what matrix allows us to go from $\begin{pmatrix}x\\ y\end{pmatrix}$ to $\begin{pmatrix}kx\\ ky\end{pmatrix}$. From that, we get:

$$\boldsymbol{r}' = \begin{pmatrix}k & 0\\ 0 & k\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}$$

We can actually use this matrix to stretch the object if we have two different values of k:

For example, we can enlarge the x-axis and keep the y-axis unchanged to stretch the x-axis with the matrix:

$$\begin{pmatrix} k & 0 \\ 0 & 1 \end{pmatrix}$$

Similarly, we can enlarge the y-axis and keep the x-axis unchanged to stretch the y-axis with the matrix:

$$\begin{pmatrix} 1 & 0 \\ 0 & k \end{pmatrix}$$

4 Lost in Translation

A translation is just moving a point to either side, up or down or both. Again consider:

$$\boldsymbol{r} = \begin{pmatrix} x \\ y \end{pmatrix}$$

And we want to move it a units right, and b units up. Our new x coordinate is $x + a$, and our new y coordinate is $y + b$. So, we can write:

$$\boldsymbol{r}' = \begin{pmatrix} x + a \\ y + b \end{pmatrix}$$

This is just a form of addition and so we can split this to:

$$\boldsymbol{r}' = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix}$$

By the same logic, for $\boldsymbol{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$, $\boldsymbol{r}' = \begin{pmatrix} x + a \\ y + b \\ z + c \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ in 3D coordinates.

Although this works, most films don't use this method as this is not matrix multiplication. Computers prefer one consistent system, and this allows transformations to be combined easily. Suppose we have some 2x2 matrix where this matrix multiplied by $\boldsymbol{r}$ = $\boldsymbol{r}'$.

Let this matrix = $\begin{pmatrix} p & q \\ r & s \end{pmatrix}$

So $r' = \begin{pmatrix} p & q \\ r & s \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} px + qy \\ rx + sy \end{pmatrix}$

However, these are all multiples of x or y so we can't get a $x + a$ or $y + b$ because no matter the value of p, q, r, or s, we can't get a number (a or b) by itself.

We fix this by using 'homogenous coordinates'. Basically, we sneak in a 1 to get: $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

Now consider a 3x3 matrix:

$$\boldsymbol{r}' = \begin{pmatrix} p & q & r \\ s & t & u \\ v & w & z \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} px + qy + r \\ sx + ty + u \\ vx + wy + z \end{pmatrix}$$

Now since we want out top column to be $x + a$:

$px + qy + r = x + a$

So, $p = 1, q = 0, r = a$

$sx + ty + u = y + b$

So, $s = 0, t = 1, u = b$

And in the third row, $vx + wy + z = 1$

So, $v = 0, w = 0, z = 1$

This gives us our final matrix that allows us to do linear transformations:

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix}.$$

The same can be done to extend this for 3D coordinates with the matrix:

$$\begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5 Let there be Light!

One of the most interesting parts of CGI and animation is how they create artificial lighting. To do this, they ask the question: how bright is a surface? After years of research, scientists observed the following shocking discoveries about light:

1. Light hitting a surface head-on leads to a very bright surface
2. Light hitting a surface at an angle leads to a dimmer surface
3. Light parallel to or behind a surface leads to darkness.

This is where the first section on discretisation becomes key. By splitting a curved/continuous surface into a triangular mesh, we have created a large (but finite) number of flat surfaces. You may be wondering why triangles are used and the answer is that any three points in 3D space (so long as they are not in a straight line) will always lie on a single flat plane. Comparatively, other shapes - like quadrilaterals - can warp or become non-planar if one corner is moved out of alignment, so it can be ambiguous to render.

Instead of attempting to light a curved surface, they now light each triangle. Each triangle gets a ‘brightness’ value that we can calculate. The accuracy of lighting depends on the level of discretisation, as increasing the number of triangles improves the approximation of curved surfaces.

To find this 'brightness' value, we introduce two vectors: the light direction (L) and the surface normal (N).

The normal vector is perpendicular to the surface and represents which way the surface is facing. But how do we find this vector? Before trying to find this, let's take a look at two incredibly important and useful vector products.

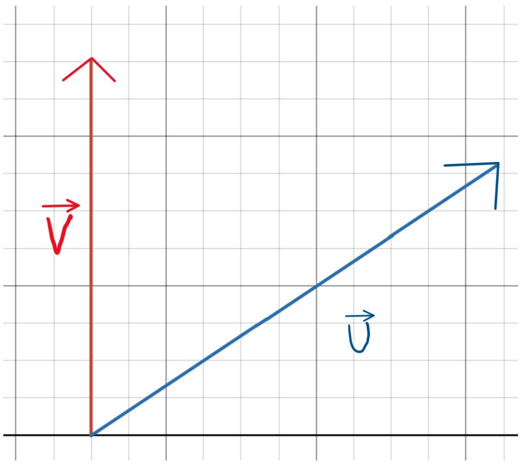
The first is the dot product. This is used to measure how aligned two vectors are (their 'sameness') and is defined as the sum of the products of their corresponding components. For example, given:

$\boldsymbol{a} = (a_1, a_2, a_3)$ and $\boldsymbol{b} = (b_1, b_2, b_3)$

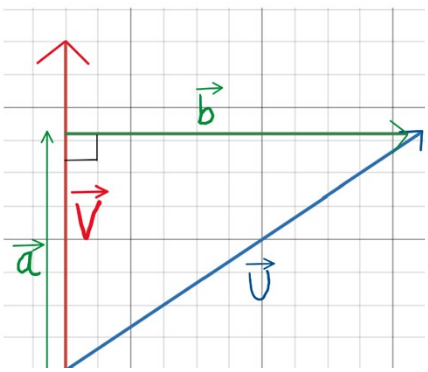
$\boldsymbol{a} \cdot \boldsymbol{b} = a_1 b_1 + a_2 b_2 + a_3 b_3$

This can also be computed geometrically, as $\boldsymbol{a} \cdot \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}| \cos\theta$. But why?

Let's consider two vectors (which both have the same length of 1):



You can always write that $\boldsymbol{u}$ is equal to some part in the same direction as $\boldsymbol{v}$ and some part that is perpendicular to $\boldsymbol{v}$, by considering the following triangle:



$\boldsymbol{u} = \boldsymbol{a} + \boldsymbol{b}$ where $\boldsymbol{a}$ is in the same direction as $\boldsymbol{v}$ and $\boldsymbol{b}$ is perpendicular to $\boldsymbol{v}$.

We can rewrite this as $\boldsymbol{u} = \mu\boldsymbol{v} + \boldsymbol{b}$, where μ basically quantifies how much the $\boldsymbol{u}$ direction has of the $\boldsymbol{v}$ direction.

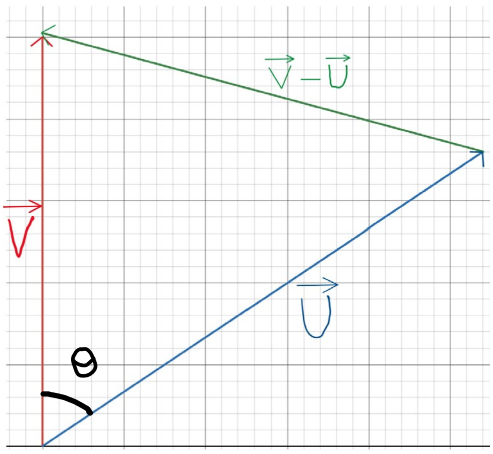
This value of μ decreases as $\boldsymbol{u}$ rotates away from $\boldsymbol{v}$: when they are parallel, $\mu = 1$, when they are perpendicular, $\mu = 0$, and when they are pointing in opposite directions, $\mu = -1$. This idea of μ represents the dot product quite clearly, but we still haven't derived the geometric definition.

By using its algebraic definition as the sum of the products of corresponding component, we can see that the dot product of a vector with itself is equal to the sum of the squares of its components, which is the square of its magnitude. Basically:

$$\boldsymbol{a} \cdot \boldsymbol{a} = a_1a_1 + a_2a_2 \ldots = a_1^2 + a_2^2 \ldots = \sqrt{a_1^2 + a_2^2 \ldots}^{\,2} = |\boldsymbol{a}|^2$$

$$\boldsymbol{a} \cdot \boldsymbol{a} = |\boldsymbol{a}|^2$$

We can use this to derive its geometric definition using the cosine rule:



$$a^2 = b^2 + c^2 - 2bc\cos A$$

$$|\boldsymbol{v} - \boldsymbol{u}|^2 = |\boldsymbol{v}|^2 + |\boldsymbol{u}|^2 - 2|\boldsymbol{v}||\boldsymbol{u}|\cos\theta$$

Using our previous rule:

$$|\boldsymbol{v} - \boldsymbol{u}|^2 = (\boldsymbol{v} - \boldsymbol{u}) \cdot (\boldsymbol{v} - \boldsymbol{u}) = \boldsymbol{v} \cdot \boldsymbol{v} + \boldsymbol{u} \cdot \boldsymbol{u} - 2(\boldsymbol{v} \cdot \boldsymbol{u})$$

$$|\boldsymbol{v}|^2 + |\boldsymbol{u}|^2 - 2|\boldsymbol{v}||\boldsymbol{u}|\cos\theta = \boldsymbol{v} \cdot \boldsymbol{v} + \boldsymbol{u} \cdot \boldsymbol{u} - 2|\boldsymbol{v}||\boldsymbol{u}|\cos\theta$$

Which gives us:

$$\boldsymbol{v} \cdot \boldsymbol{v} + \boldsymbol{u} \cdot \boldsymbol{u} - 2(\boldsymbol{v} \cdot \boldsymbol{u}) = \boldsymbol{v} \cdot \boldsymbol{v} + \boldsymbol{u} \cdot \boldsymbol{u} - 2|\boldsymbol{v}||\boldsymbol{u}|\cos\theta$$

$$-2(\boldsymbol{v} \cdot \boldsymbol{u}) = -2|\boldsymbol{v}||\boldsymbol{u}|\cos\theta$$

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

This is the basis of how lighting is conducted in CGI and animation, but let's first look at the cross product. The cross product of two vectors is an operation that results in a third vector that is perpendicular to the plane containing the original two. How?

Let's have our two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, and we want the vector perpendicular to both, call it $\boldsymbol{m}$.

Let: $\boldsymbol{m} = (x, y, z), \boldsymbol{u} = (u_1, u_2, u_3), \boldsymbol{v} = (v_1, v_2, v_3)$

If it is perpendicular to both, according to the geometric definition of the dot product, we know that:

$\boldsymbol{m} \cdot \boldsymbol{u} = |\boldsymbol{m}||\boldsymbol{u}| \cos 90 = |\boldsymbol{m}||\boldsymbol{u}|0 = 0$

$\boldsymbol{m} \cdot \boldsymbol{v} = |\boldsymbol{m}||\boldsymbol{v}| \cos 90 = |\boldsymbol{m}||\boldsymbol{v}| \cos 90 = 0$

Using the algebraic definition, we therefore know that:

$\boldsymbol{m} \cdot \boldsymbol{u} = u_1 x + u_2 y + u_3 z = 0$

$\boldsymbol{m} \cdot \boldsymbol{v} = v_1 x + v_2 y + v_3 z = 0$

We now need a non-zero solution to these equations, which gives:

$x = u_2 v_3 - u_3 v_2$

$y = u_3 v_1 - u_1 v_3$

$z = u_1 v_2 - u_2 v_1$

So, $\boldsymbol{m} = (u_2 v_3 - u_3 v_2 \,, u_3 v_1 - u_1 v_3 \,, u_1 v_2 - u_2 v_1)$

This vector satisfies both equations and so is perpendicular to both $\boldsymbol{u}$ and $\boldsymbol{v}$.

We define this vector as the cross product $\boldsymbol{u} \times \boldsymbol{v}$.

Back to the idea of lighting, we use the cross product to find the normal (N) to each triangle on the surface of our object - though there's nothing normal about needing it. This tells us the direction the surface is 'facing', though we must ensure that both vectors are normalised before they are used.

Now we use our dot product:

$\boldsymbol{L} \cdot \boldsymbol{N} = |\boldsymbol{L}||\boldsymbol{N}| \cos \theta$

$\boldsymbol{L} \cdot \boldsymbol{N} = \cos \theta$, as the magnitude of both is 1.

Where θ is the angle between light and the surface, and $\cos \theta$ tells us about the alignment between them.

We define brightness $\propto \cos \theta$.

So:

When $\theta = 0°$, $\cos\theta = 1$ which results in maximum brightness acting on the triangle.

For $0 < \theta < 90°$, we get varying brightness

When $\theta > 90°$, $\cos\theta < 0$ so the light is behind the surface and appears dark.

This is known as Lambert's cosine law and is how computers determine the brightness acting on each triangle on the mesh of the object.

6 That's a Wrap

In conclusion, the seemingly complex and lifelike visuals seen in modern films are built upon matrices, and more fundamentally, vectors. All these complex and confusing processes used really make you reflect upon the great minds that contribute to parts of our lives we take for granted. I hope next time you watch Spiderman's escapades in space or one of your childhood favourite cartoons you remember the role that maths has played to bring the story to life. The magic of cinema is, at its core, an elegant application of mathematics.