Interpolating rotations

A common task in animation is interpolating between *key frames*. Film animations typically run at the rate of 24 frames per second. Preparing a scene for each one of these 24 frames is too tedious a task, so animators will typically set up only a few of those 24 frames at key points in the time sequence. To fill in the frames between those key frames, software will typically interpolate the positions of all of the scene elements between those key frames.

Some interpolation tasks are mathematically trivial. For example, if an animator wants an object to appear at location $\tilde{p}$ at time $t = 0$ and at location $\tilde{q}$ at time $t = 1$ and move in a straight line between those points, the intermediate locations can be computed via the straightforward interpolation formula

$$\tilde{s}(t) = \tilde{p}\,(1 - t) + \tilde{q}\,t$$

Alternatively, we can achieve the same effect by doing an interpolation of transformation matrices. If T_1 is the transformation matrix that places an object at point $\tilde{p}$ and T_2 is a transformation matrix that places an object at point $\tilde{q}$, we can construct an interpolating transformation

$$T(t) = T_1\,(1 - t) + T_2\,t$$

Other interpolation computations are less trivial. One example of a more challenging interpolation has to do with rotations. If we want to animate a rotation in which an object is rotated through an angle θ_1 at time $t = 0$ and through an angle θ_2 at time $t = 1$ we can not interpolate the rotations in a straightforward way by interpolating the corresponding transformation matrices. An even more challenging example is an interpolation where the starting position is expressed as a rotation through an angle θ about some axis $\vec{k}$ and the ending position is expressed as a rotation through an angle φ about some other axis $\vec{l}$.

We could potentially interpolate rotations by doing something like the following. If R is the matrix for a rotation and Q is the matrix for a second rotation, we can interpolate between them by doing something like

$$S = \left(Q\,R^{-1}\right)^{\alpha} R$$

This is correct, but computationally difficult. The basic problem here is that although it is possible to define a matrix exponentiation operation (take a course in linear algebra to see the details), computationally that operation is not very straightforward.

Representing rotations via complex numbers

If we want to interpolate rotations in a more straightforward way, we will have to shift to an alternative representation for the rotations.

I am going to show one example of how this can be done by looking at rotations in 2d space.

The alternative representation system I am going to use is to map a point (x,y) in the standard 2d coordinate space to a point $x + i\,y$ in the complex plane.

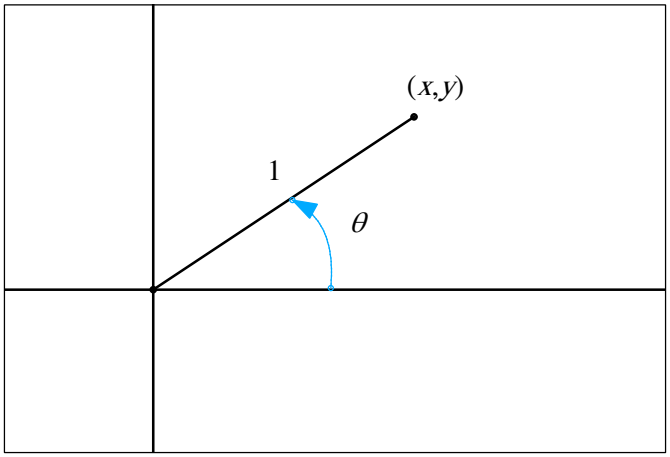
Consider a point (x,y) in the 2d plane. We can map this point to a point $z = x + i\,y$ in the complex plane. Once in the complex plane we can switch to an alternative representation of that point by expressing the point in polar coordinates:

$$z = r\,e^{i\,\theta}$$

$$r = \sqrt{x^2 + y^2}$$

$$\theta = \tan^{-1}\left(\frac{y}{x}\right)$$

The analogy that we are going to set up to help us handle rotations in the 2d plane is that a rotation is analogous to a point in the plane with distance 1 from the origin.



The analogy is that

$$(x,y) \Rightarrow x + i\,y \Rightarrow e^{i\,\theta}$$

The advantage here is that we can now compose rotations by multiplying. If (x,y) is the point that rotating through an angle θ maps us to, and (s,t) is the point that rotating through an angle φ takes us to, then the point we arrive at after rotating through an angle $\theta + \varphi$ is

$$(p,q) \Leftarrow e^{i\,\theta}\,e^{i\,\varphi} = e^{i(\theta+\varphi)}$$

In other words, we can compose rotations in the 2d plane by instead multiplying numbers in the complex plane.

Another related trick is to construct a fractional rotation by using powers. For example, if (x,y) is the point we map to after rotating through an angle θ, then a fractional rotation through an angle $\alpha\,\theta$ where $0 \le \alpha \le 1$ is given by

$$(p,q) \Leftarrow \left(e^{i\,\theta}\right)^{\alpha} = e^{i\,\alpha\,\theta}$$

This now gives us a convenient way to interpolate between rotations in the plane. If (x,y) is the point that rotating through an angle θ maps us to, and (s,t) is the point that rotating through an angle φ takes us to, then the point we

arrive at after rotating through an angle $\theta + \alpha(\varphi\text{-}\theta)$ is

$$(p,q) \Leftarrow \left(\frac{e^{i\varphi}}{e^{i\theta}}\right)^{\alpha} e^{i\theta} \Leftarrow \left(e^{i(\varphi\text{-}\theta)}\right)^{\alpha} e^{i\theta} \Leftarrow e^{i\theta} e^{i\alpha(\varphi\text{-}\theta)} = e^{i(\theta + \alpha(\varphi\text{-}\theta))}$$

The upshot of all of this is that we have figured out how to interpolate rotations by transforming to the complex plane, doing divisions, multiplications, and powers in the complex plane, and then mapping back from the complex plane to the real plane.

Shifting the analogy to three dimensions

Now that we have seen the trick for interpolating rotations in the 2d plane, we will want to do the same in 3d. The only problem we are going to encounter is that although the field $\mathbb{C}$ of complex numbers makes an obvious analog for the plane $\mathbb{R}^2$, there does not appear to be an obvious analog for $\mathbb{R}^3$. This is in fact the wrong way to think about what we are doing: remember, the main thing we are trying to do here is set up an analogy in which *rotations* map to a point in some field. The actual analogy we used in the section above was

$$\text{rotate counter-clockwise through an angle } \theta \Rightarrow$$

$$(x,y) = (\cos\theta, \sin\theta) \Rightarrow$$

$$z = x + i\,y \Rightarrow$$

$$z = e^{i\theta}$$

In our actual analogy, a rotation in the plane maps to a point on the unit circle in the field $\mathbb{C}$.

To extend this analogy to 3d, we first need to think about rotations in 3d. To fully describe a rotation in 3d we need to specify an axis of rotation, $\vec{k}$, and an angle θ to rotate through. Between them, these two things have four variables that can be varied independently. That means that any field we map the rotation to will need to have a coordinate system based on 4 variables. A field with the desired characteristics is the field of *quaternions*. In our analogy we will map a rotation in $\mathbb{R}^3$ to a point on the unit sphere in the field of quaternions.

Following our analogy from $\mathbb{C}$, if R and Q are two rotations in $\mathbb{R}^3$ that map to points r and q in the quaternion space, the rotation S that interpolates between these two rotations is

$$S \Leftarrow \left(\frac{q}{r}\right)^{\alpha} r$$

where the operations of multiplication, division, and exponentiation used on the right are the appropriate operations in the field of quaternions.

More about quaternions

To describe the field of quaternions in a little more detail, I need to establish some basic properties of quaternions.

A quaternion is a point in a four-dimensional space. The *basis* for the space of quaternions is the set of elements $(1,i,j,k)$ that satisfy the following properties:

$$i^2 = j^2 = k^2 = -1$$

$$ijk = -1$$

A typical point in quaternion space can be represented in a coordinate representation as (a,b,c,d). In terms of the basis elements, this point is

$$a + b\,i + c\,j + d\,k$$

Using the properties of the basis elements and assuming that the usual distributive laws of arithmetic apply we can work out what the product of two quaternions looks like:

$$(a,b,c,d)\cdot(e,f,g,h) =$$

$$(a + b\,i + c\,j + d\,k)(e + f\,i + g\,j + h\,k)$$

$$= (a\,e - b\,f - c\,g - d\,h) + (a\,f + b\,e + c\,h - d\,g)\,i + (a\,g - b\,h + c\,e + d\,f)\,j + (a\,h + b\,g - c\,f + d\,e)\,k$$

Another notation system used with quaternions groups the last three coordinates together to make a vector.

$$(a,\ b,\ c,\ d) \Rightarrow \begin{bmatrix} a \\ \vec{v} \end{bmatrix} \text{ where } \vec{v} = \begin{bmatrix} b \\ c \\ d \end{bmatrix}$$

In this notation system the multiplication rule becomes

$$\begin{bmatrix} a \\ \vec{v} \end{bmatrix}\begin{bmatrix} e \\ \vec{w} \end{bmatrix} = \begin{bmatrix} a\,e - \vec{v}\cdot\vec{w} \\ a\,\vec{w} + e\,\vec{v} + \vec{v}\times\vec{w} \end{bmatrix}$$

One complication with quaternion multiplication is that it is not commutative. This means that we will have to exercise a little caution below when constructing products.

Another standard operation in the space of quaternions is the operation of conjugation:

$$q^* = -\frac{1}{2}\left(q + iqi + jqj + kqk\right)$$

Conjugation leads to a norm operation:

$$\|q\| = \sqrt{q\,q^*}$$

This in turn allows us to define a multiplicative inverse:

$$q^{-1} = \frac{1}{\|q\|^2}\,q^*$$

Quaternions can also be expressed in a polar notation. The key to this is defining an exponential function:

$$e^q = \sum_{n=0}^{\infty} \frac{1}{n!} q^n$$

Using this exponential notation we can write any quaternion in polar form as

$$q = \|q\| \, e^{\hat{n}\,\theta}$$

Here

$$q = a + b\,i + c\,j + d\,k = a + \mathbf{v}$$

$$\mathbf{v} = \hat{n}\,\|\mathbf{v}\|$$

$$a = \|q\| \cos\,\theta$$

This polar notation allows us to define fractional powers:

$$q^\alpha = \|q\|^\alpha \, e^{\hat{n}\,\alpha\,\theta}$$

Outline of the mapping

Here now is an outline of the procedure we are going to use to compose and interpolation rotations.

1. Given a rotation R through an angle θ about a unit vector $\vec{k}$ in $\mathbb{R}^3$, we somehow use $\vec{k}$ and θ to construct a quaternion with unit norm.

$$R \Rightarrow r = e^{\hat{n}\,\varphi}$$

2. We map a second rotation Q in a similar way.

$$Q \Rightarrow q = e^{\hat{m}\,\psi}$$

3. To interpolate between these two quaternions we do

$$\left(q\,r^{-1}\right)^\alpha r$$

4. Finally, we map this interpolated quaternion back to a result rotation.

Further details

The key to getting the outline above to work correctly is to specify the mapping that maps a rotation to a quaternion on the unit sphere in quaternion space. Given a rotation R through an angle θ about a unit vector $\vec{k}$ in $\mathbb{R}^3$, we will map to the quaternion

$$r = \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right) k_1, \sin\left(\frac{\theta}{2}\right) k_2, \sin\left(\frac{\theta}{2}\right) k_3\right)$$

It is easy to see what the multiplicative inverse r^{-1} should be, since the inverse of the original rotation is a rotation about the same axis through and angle of $-\theta$.

$$r^{-1} = \left(\cos\left(\frac{\theta}{2}\right), -\sin\left(\frac{\theta}{2}\right) k_1, -\sin\left(\frac{\theta}{2}\right) k_2, -\sin\left(\frac{\theta}{2}\right) k_3\right)$$

For a quaternion on the unit sphere, the exponentiation operation works out to be similarly simple.

$$r^{\alpha} = \left(\cos\left(\frac{\alpha\,\theta}{2}\right), \sin\left(\frac{\alpha\,\theta}{2}\right) k_1, \sin\left(\frac{\alpha\,\theta}{2}\right) k_2, \sin\left(\frac{\alpha\,\theta}{2}\right) k_3\right)$$

Now that we know that all the operations we want to carry out are simple to do in practice, we can put this all together to compute the *spherical linear interpolation* (or *slerp*) of two rotations. If R is a rotation through an angle θ about a unit vector $\vec{k}$ in $\mathbb{R}^3$ and Q is a rotation through an angle φ about a unit vector $\vec{l}$ in $\mathbb{R}^3$ we do the following to interpolate between the two rotations smoothly:

$$r = \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right) k_1, \sin\left(\frac{\theta}{2}\right) k_2, \sin\left(\frac{\theta}{2}\right) k_3\right)$$

$$q = \left(\cos\left(\frac{\varphi}{2}\right), \sin\left(\frac{\varphi}{2}\right) l_1, \sin\left(\frac{\varphi}{2}\right) l_2, \sin\left(\frac{\varphi}{2}\right) l_3\right)$$

$$s = \left(q\, r^{-1}\right)^{\alpha} r$$

Note that the multiplications in the last expression above are all quaternion multiplications, and will have to use the more complex rules for quaternion multiplication.

The only remaining work is the mapping back to a rotation in $\mathbb{R}^3$:

$$\omega = 2 \arccos(s_1)$$

$$m_1 = s_2/\sin(\omega/2)$$

$$m_2 = s_3/\sin(\omega/2)$$

$$m_3 = s_4/\sin(\omega/2)$$