Moving at Constant Speed

David Eberly
Magic Software, Inc.
http://www.magic-software.com

Created: January 10, 2001

1 Introduction

A frequently asked question in comp.graphics.algorithms is how to move an object (typically the eyepoint of a camera model) along a parameterized curve with constant speed. The method to do this requires the curve to be *reparameterized by arc length*. The following construction applies in 2D or 3D.

Let the curve be parameterized as $\vec{X}(t)$ for $t \in [a, b]$. Imagine a particle moving along the curve, its *position* being $\vec{X}(t)$ at the time t. The derivative is denoted $d\vec{X}(t)/dt$ and is the *velocity* vector for the particle. This vector is tangent to the curve at position $\vec{X}(t)$. The *speed* of the particle is the length of the velocity vector, $|d\vec{X}/dt|$. If $|d\vec{X}(t)| = c$, a constant for all t, then the particle travels with *constant speed* along the curve. If $c = 1$, then the particle travels with *unit speed* along the curve.

For example, consider $\vec{X}(s) = (\cos(s), \sin(s))$ for $s \in [0, 2\pi]$. The curve is a circle in the plane and is centered at $(0, 0)$ and has radius 1. The derivative is $d\vec{X}/ds = (-\sin(s), \cos(s))$ and has length $|d\vec{X}/ds| = 1$. A particle traveling on this curve moves with constant speed. A different parameterization of the circle is $\vec{Y}(t) = (r\cos(t^2), r\sin(t^2))$ for $t \in [0, \sqrt{2\pi}]$. The derivative is $d\vec{Y}/dt = (-2rt\sin(t^2), 2rt\cos(t^2))$ and has length $|d\vec{Y}/dt| = 2rt$. The speed increases with time, so the particle does not move with constant speed.

Suppose that you were given $\vec{Y}(t)$, the parameterization of the circle for which the particle does not travel with constant speed. And suppose you want to change the parameterization so that the particle does travel with constant speed. That is, you want to relate the parameter t to a new parameter s, say by a function $t = f(s)$, so that $\vec{X}(s) = \vec{Y}(t)$ and the particle moves with constant speed with respect to the parameter s. In the contrived example, $t = \sqrt{s}$, but how do you actually find this relationship between t and s generally?

2 Reparameterization by Arc Length

If $|d\vec{X}/ds| = 1$ for all s, the parameter s represents length measured while traveling along the curve, called *arc length*. For each "small" change ds in arc length, the length of the difference in positions is the same "small" change $|d\vec{X}| = |ds|$. Given any other curve parameter t (representing time), the speed at which a particle travels along the curve is ds/dt. Observe that this is "change in distance" (ds) per "change in time" (dt). Applying the chain rule from Calculus,

$$\frac{d\vec{Y}(t)}{dt} = \frac{d\vec{X}(s)}{dt} = \frac{d\vec{X}(s)}{ds}\frac{ds}{dt}.$$

Computing lengths yields

$$\left|\frac{d\vec{Y}(t)}{dt}\right| = \left|\frac{d\vec{X}(s)}{ds}\frac{ds}{dt}\right| = \left|\frac{ds}{dt}\right|$$

since $|d\vec{X}/ds| = 1$. If we also assume the particle traveling along the curve can never backtrack (reverse direction and visit positions already encountered earlier), then $ds/dt \geq 0$ and

$$\frac{ds}{dt} = \left|\frac{d\vec{Y}}{dt}\right|.$$

As expected, the speed of traversal along the curve is the length of the velocity vector. This equation tells you how ds/dt depends on time t. To obtain a relationship between s and t, you have to integrate this to obtain s as a function $g(t)$ of time:

$$s = g(t) = \int_a^t \left|\frac{d\vec{Y}(\tau)}{dt}\right| d\tau.$$

When $t = a$, $s = g(a) = 0$. This makes sense since the particle's initial position is at $t = a$ and has not yet moved; distance traveled is zero. When $t = b$, $s = g(b) = L$ is the total distance L traveled along the curve.

Given the time t, we can determine the corresponding arc length s from the integration. However, what is needed is the inverse problem. Given an arc length s, we want to know the time t at which this arc length occurs. The idea in the original application is that you decide the distance that the object should move first, then figure out what position $\vec{Y}(t)$ corresponds to it. Abstractly, this means inverting the function g to obtain

$$t = g^{-1}(s).$$

Mathematically, the chances of inverting g in terms of elementary functions are slim to none. You have to rely on numerical methods to compute $t \in [a, b]$ given a value of $s \in [0, L]$.

3 Numerical Solution

Define $F(t) = g(t) - s$. Given a value s, the problem is now to find a value t so that $F(t) = 0$. This is a root–finding problem that can be solved using Newton's method. If $t_0 \in [a, b]$ is an initial guess for t, then Newton's method produces a sequence

$$t_{i+1} = t_i - \frac{F(t_i)}{F'(t_i)}, \quad i \geq 0$$

where

$$F'(t) = \frac{dF}{dt} = \frac{dg}{dt} = \left|\frac{d\vec{Y}(t)}{dt}\right|.$$

The iterate t_1 is determined by t_0, $F(t_0)$, and $F'(t_0)$. Evaluation of $F'(t_0)$ is straightforward since you already have a formula for $\vec{Y}(t)$ and can compute $d\vec{Y}(t)/dt$ from it. Evaluation of $F(t_0)$ requires computing $g(t_0)$, an integration that can be done using standard numerical integrators.

A reasonable choice for initial iterate is $t_0 = a + (s/L)(b - a)$. The value s/L is the fraction of arc length at which the particle should be located. The initial iterate applys that same fraction to the parameter

interval $[a, b]$. The subsequent iterates are computed until either $F(t_i)$ is sufficiently close to zero (sufficiency determined by the application) or until a maximum number of iterates has been computed (maximum determined by the application). Pseudocode is

```
Point Y (float t);   // The parameterization of the curve, a <= t <= b.
Point DY (float t);  // The derivative of Y(t), a <= t <= b.
float Speed (float t) { return Length(DY(t)); }
float ArcLength (float t) { return Integral(a,t,Length(DY()); }

float GetT (float s)  // 0 <= s <= total_arc_length
{
    float L = ArcLength(b);  // precompute L if you frequently call GetT
    float t = a + s*(b-a)/L;  // precompute (b-a)/L if you frequently call GetT

    for (int i = 0; i < max; i++)  // 'max' is application-specified
    {
        float F = ArcLength(t) - s;
        if ( Abs(F) < epsilon )  // 'epsilon' is application-specified
            return t;

        float DF = Speed(t);
        t -= F/DF;  // warning:  DF = 0 might be a problem
    }

    // maximum iterations exceeded, you might want to trap this
    return t;
}
```

The function `Integral(a,t,f())` is any numerical integrator that computes the integral of $f(\tau)$ over the interval $[a, t]$.