

Today I will give an introduction to the topic of smooth curves and surfaces in \mathfrak{R}^3 . My intention here is give you a taste of a very powerful, vast, and commonly used set of techniques. (For example, you may have heard of Bezier curves, B-splines, NURBS, etc.) The material today is a first step toward eventually understanding these techniques.

Cubic Curves

So far, we have consider simple shapes only: lines, polygons, quadrics. In the next few lectures, we will increase the complexity of our models. We begin today by considering smooth cubic curves, which are parametric curves defined by third order polynomials. Such curves are useful for two reasons. First, they can be used to define the paths of points in an animation. These paths can be the trajectory of an object, or they can be the paths of the camera. Second, smooth curves can be used to define smooth (bicubic) surfaces, which we will examine later in the lecture.

Suppose we want to define a 3D curve that passes through three points $\mathbf{p}(0), \mathbf{p}(1), \mathbf{p}(2) \in \mathfrak{R}^3$. This is relatively easy. We can always fit a circle to any three points, as long as they are not collinear. A circle is a quadratic curve, which can be defined by a second order polynomial in x, y, z , namely apply a 3D rotation to a planar circle.

To allow ourselves more complexity, suppose we want to define a 3D parametric curve

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

that passes through four points. As we argue below, we can easily do this if each of the functions $x(t), y(t), z(t)$ is a third order polynomial in parameter t , for example,

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$

and similarly for $y(t)$ and $z(t)$. Since each polynomial has four coefficients, we need to specify 12 coefficients in total. Since we have 12 degrees of freedom, we might guess that we can specify a curve by choosing four points in \mathfrak{R}^3 (since $4 \times 3 = 12$). This is correct, as we show next.

We define a curve $\mathbf{p}(t)$ by choosing four points in \mathfrak{R}^3 and defining these points to be *on the curve* and at values $t = 0, 1, 2, 3$, that is, the points are $\mathbf{p}(0), \mathbf{p}(1), \mathbf{p}(2), \mathbf{p}(3)$. How do we do it? Let's define $\mathbf{p}(t)$ to be a column vector:

$$\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \quad (1)$$

Solving for the twelve a, b, c, d coefficients turns out to be easy. Just make a 3×4 matrix from the four points and substitute the four values $t = 0, 1, 2, 3$:

$$\begin{bmatrix} \mathbf{p}(0) & \mathbf{p}(1) & \mathbf{p}(2) & \mathbf{p}(3) \end{bmatrix} = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \end{bmatrix} \begin{bmatrix} 0 & 1 & 8 & 27 \\ 0 & 1 & 4 & 9 \\ 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

We then solve for the coefficients by right-multiplying by a matrix

$$\mathbf{B} = \begin{bmatrix} 0 & 1 & 8 & 27 \\ 0 & 1 & 4 & 9 \\ 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1}$$

i.e.

$$\begin{bmatrix} \mathbf{p}(0) & \mathbf{p}(1) & \mathbf{p}(2) & \mathbf{p}(3) \end{bmatrix} \begin{bmatrix} 0 & 1 & 8 & 27 \\ 0 & 1 & 4 & 9 \\ 0 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \end{bmatrix}$$

We refer to the 3×4 matrix with the $\mathbf{p}()$ values as the *geometry matrix* \mathbf{G} , or the matrix of *control points*. We rewrite Eq. (1) as:

$$\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \mathbf{GB} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}.$$

The product

$$\mathbf{B} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

is a 4×1 column vector whose rows (single elements) are *blending functions*. These are four (third order) polynomials in t which define the contribution of each of the four control points $\mathbf{p}(0), \mathbf{p}(1), \mathbf{p}(2), \mathbf{p}(3)$ to the curve point $\mathbf{p}(t)$.

Finally, we can approximate the curve $\mathbf{p}(t)$ using a set of lines. This can be done by dividing the interval $t \in [0, 4]$ into pieces (possibly of equal length, possibly not) and substituting these t values into the equation for $\mathbf{p}(t)$.

Hermite Curves

A similar approach is to take two points $\mathbf{p}(0)$ and $\mathbf{p}(1)$ and to choose the tangent vector at each of these two points. By *tangent vector*, I mean the derivative with respect to the parameter t :

$$\mathbf{p}'(t) \equiv \frac{d\mathbf{p}(t)}{dt} \equiv \lim_{\delta t \rightarrow 0} \frac{\mathbf{p}(t + \delta t) - \mathbf{p}(t)}{\delta t}$$

Intuitively, if t were interpreted as time, then the tangent vector would be the 3D velocity of the curve. This interpretation is relevant, for example, in animations if we are defining the path of a camera or the path of a vertex.

From Eq. (1), we observe that

$$\mathbf{p}'(t) = \begin{bmatrix} x'(t) \\ y'(t) \\ z'(t) \end{bmatrix} = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \end{bmatrix} \begin{bmatrix} 3t^2 \\ 2t \\ 1 \\ 0 \end{bmatrix}$$

Substituting for $t = 0$ and $t = 1$ for $\mathbf{p}(t)$ and $\mathbf{p}'(t)$ gives us a 3×4 matrix:

$$\begin{bmatrix} \mathbf{p}(0) & \mathbf{p}(1) & \mathbf{p}'(0) & \mathbf{p}'(1) \end{bmatrix} = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}.$$

Again, we can compute the coefficient matrix by right multiplying by the inverse of the 4×4 matrix at the right end of the above equation. Define this inverse to be

$$\mathbf{B}_{Hermite} \equiv \begin{bmatrix} 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}^{-1}$$

and define the 3×4 *geometry matrix* (or *control points*)

$$\mathbf{G}_{Hermite} \equiv \begin{bmatrix} \mathbf{p}(0) & \mathbf{p}(1) & \mathbf{p}'(0) & \mathbf{p}'(1) \end{bmatrix}$$

then we can write

$$\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \mathbf{G}_{Hermite} \mathbf{B}_{Hermite} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}. \quad (2)$$

The 4×1 column vector

$$\mathbf{B}_{Hermite} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

contains four *Hermite blending functions*. These four functions are each third order polynomials in t , and define the contribution of the four control points $\mathbf{p}(0), \mathbf{p}(1), \mathbf{p}'(0), \mathbf{p}'(1)$, respectively, to the curve $\mathbf{p}(t)$. Note that these blending functions themselves are independent of the chosen control points.

In practice, we often want to draw a complicated curve in which we have n points $\mathbf{p}(0)$ to $\mathbf{p}(n-1)$ and we have tangents $\mathbf{p}'(0)$ to $\mathbf{p}'(n-1)$ defined at these points as well. We would like to fit a curve (say Hermite) between each pair of points. We can be sure that the curve passes through the points and that the tangents are continuous, simply by making sure that the endpoint and tangent of one curve fragment (say, from $t = k-1$ to $t = k$) is the same as the beginning point and tangent of the next curve fragment (from $t = k$ to $t = k+1$).

Bicubic Surfaces

We next turn from curves to surfaces. We wish to define a two parameter surface:

$$\mathbf{p}(s, t) = (x(s, t), y(s, t), z(s, t))$$

which maps

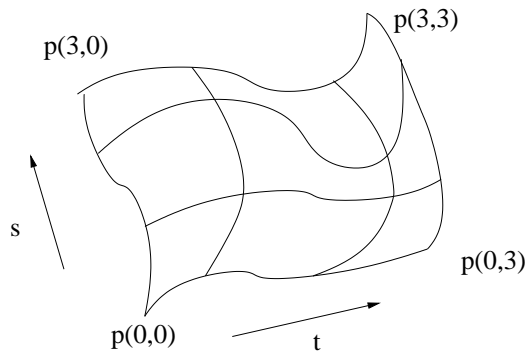
$$\mathbf{p} : \mathbb{R}^2 \rightarrow \mathbb{R}^3.$$

such that each x, y, z is a polynomial in s, t of (maximum) degree 3, and $\mathbf{p}(s, t)$ passes through a given set of 3D data points (see below). The surface $\mathbf{p}(s, t)$ is a *parametric bicubic* surface with parameters s and t , in the following sense. For any fixed s , the curve $\mathbf{p}(s, t)$ is a bicubic function of t and, for any fixed t , the curve $\mathbf{p}(s, t)$ is a bicubic function of s .

The idea for constructing this surface follows from what saw earlier this lecture. Any four distinct points in \mathbb{R}^3 , define a cubic curve that passes through these four points. To define a surface, we use four sets of four points, giving us four cubic curves. We then take corresponding points on the four cubic curves and for each of them we define a cubic curve that passes through these four corresponding points. The math behind this is as follows.

We fit a *bicubic surface* such that it passes through a grid of 4×4 points. These sixteen points are the intersections of the eight curves shown in the following sketch. The eight curves correspond to four values (0, 1, 2, 3) for each of the s, t variables that define the surface. Only the four corner points of the surface patch are labelled in the figure.

Earlier we solved the problem simultaneously for the x, y, z functions. But if you re-examine how we did this, you can see that solutions $x(t), y(t), z(t)$ were independent. So let's just look at one of these functions, namely $x(s, t)$.



Fix the parameter s (we will eventually substitute $s = 0, 1, 2, 3$) and define a cubic curve $x(s, t)$ for $t = 0, 1, 2, 3$. Applying the solution from earlier, we get

$$x(s, t) = [x(s, 0) \quad x(s, 1) \quad x(s, 2) \quad x(s, 3)] \mathbf{B} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \tag{3}$$

Next, suppose we define four curves by choosing $s = 0, 1, 2, 3$ and stacking them. The x component of these four curves would be:

$$\begin{bmatrix} x(0, t) \\ x(1, t) \\ x(2, t) \\ x(3, t) \end{bmatrix} = \begin{bmatrix} x(0, 0) & x(0, 1) & x(0, 2) & x(0, 3) \\ x(1, 0) & x(1, 1) & x(1, 2) & x(1, 3) \\ x(2, 0) & x(2, 1) & x(2, 2) & x(2, 3) \\ x(3, 0) & x(3, 1) & x(3, 2) & x(3, 3) \end{bmatrix} \mathbf{B} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \tag{4}$$

The left hand side is interesting. For any t , it is a four-tuple. To fit a cubic to this four-tuple (for fixed t), we apply the solution as before, but now the curve parameter is s .

The only minor subtlety is that the four-tuple is written a column vector whereas previously we wrote it as row vector. So to use the same form as Eq. 3 (but using s as the parameter instead of t), we need to take the transpose, i.e. we consider

$$x(s, t) = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix} \mathbf{B}^T \begin{bmatrix} x(0, t) \\ x(1, t) \\ x(2, t) \\ x(3, t) \end{bmatrix} \quad (5)$$

Substituting Eq. (4) into Eq. (5), we get:

$$x(s, t) = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix}^T \mathbf{B}^T \begin{bmatrix} x(0, 0) & x(0, 1) & x(0, 2) & x(0, 3) \\ x(1, 0) & x(1, 1) & x(1, 2) & x(1, 3) \\ x(2, 0) & x(2, 1) & x(2, 2) & x(2, 3) \\ x(3, 0) & x(3, 1) & x(3, 2) & x(3, 3) \end{bmatrix} \mathbf{B} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}.$$

which is a parametric bicubic function. The 4×4 $x(*, *)$ matrix is a geometry matrix which we can call \mathbf{G}_x . We do the same thing to define $y(s, t)$ and $z(s, t)$ via geometry matrices \mathbf{G}_y and \mathbf{G}_z .

Surface Normals

As we will see later in the course, it is often very useful to know the surface normal for any (s, t) . How could we define this? Once we have the geometry matrices $\mathbf{G}_x, \mathbf{G}_y, \mathbf{G}_z$, we can compute tangent vectors to the surfaces for any (s, t) , namely

$$\frac{\partial}{\partial t} x(s, t) = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix} \mathbf{B}^T \mathbf{G}_x \mathbf{B} \begin{bmatrix} 3t^2 \\ 2t \\ 1 \\ 0 \end{bmatrix}.$$

and

$$\frac{\partial}{\partial s} x(s, t) = \begin{bmatrix} 3s^2 & 2s & 1 & 0 \end{bmatrix} \mathbf{B}^T \mathbf{G}_x \mathbf{B} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

The same form is used for the partial derivatives of $y(s, t)$ and $z(s, t)$.

Since these two partial derivatives are both tangent to the surface (and are not identical), the surface normal must be perpendicular to both of these tangent vectors. Hence, the surface normal is parallel to the cross product of these two partial derivatives.

$$\frac{\partial}{\partial s} \mathbf{p}(s, t) \times \frac{\partial}{\partial t} \mathbf{p}(s, t)$$

We will use this surface normal in later lectures e.g. when we discuss bump mapping.