School of Electrical Engineering and Computing
Faculty of Engineering and Built Environment
***The* University *of* Newcastle**

Comp 3320/6370 Computer Graphics

Lecture 09: Hermite, Bezier, and Spline Curves

Note: These lecture slides use concepts, statements and figures

1.) From chapter 17 of the book: Akenine-Möller, T. , Haines, E., Hoffman, N., Pesce, A., Iwanicki, M. and Hillaire, S.: Real-Time Rendering, 4th edition, A K Peters, 2018.

2.) Lecture slides of Prof. Peter Eades Comp3320 course.

3.) Chapter 15 of P. Shirley: Fundamentals of Computer Graphics, 3rd edition, 2009.

Objective

Past lectures: We started with lines and planes and intersections of them. These are linear objects in a vector space. We had different representations. One of them was the parameterised representation of a line. We also saw a parameterised version of a circle.

Today: With splines we are looking at non-linear objects. Splines are a special simplified version. They approximate general non-linear objects and are suitable for a large variety of computational tasks.

Soon: In one of the next lectures we will look at the basics of general non-linear parameterised curves, surfaces and summarise all this under the concept of a manifold.

Overview

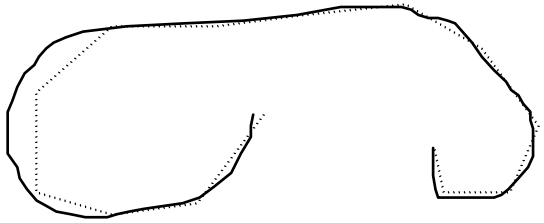
- Parametric curves and cubic polynomials
- Continuity
- Hermite Curves
- Bezier Curves
- Splines
- Parametric Bicubic Surfaces

Note:

In most of the following slides we use standard mathematical notation for clarity (if not said otherwise).

It is straight forward to translate the formalisms into homogeneous notation if required.

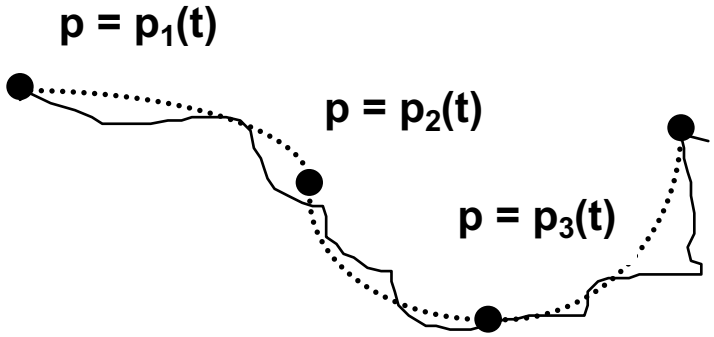
Parametric Curves



We can use poly-lines to approximate curves and polygonal meshes to approximate curved surfaces.

But:

- large numbers of line segments may be needed for accurate representations; this is both a storage and a processing problem.
- interactive manipulation of the curve/surface may be tedious.



The most convenient way to represent curves is to approximate them using parametric representations:

- **p = p(t)**,
- or **x = x(t), y = y(t), z = z(t),**
- or **(x,y,z) = (x(t), y(t), z(t)).**

In many circumstances we can choose the functions **x(t), y(t), z(t)** to be cubic in **t.**

This makes a **parametric cubic curve.**

We can join successive parametric cubic curves to approximate a curve.

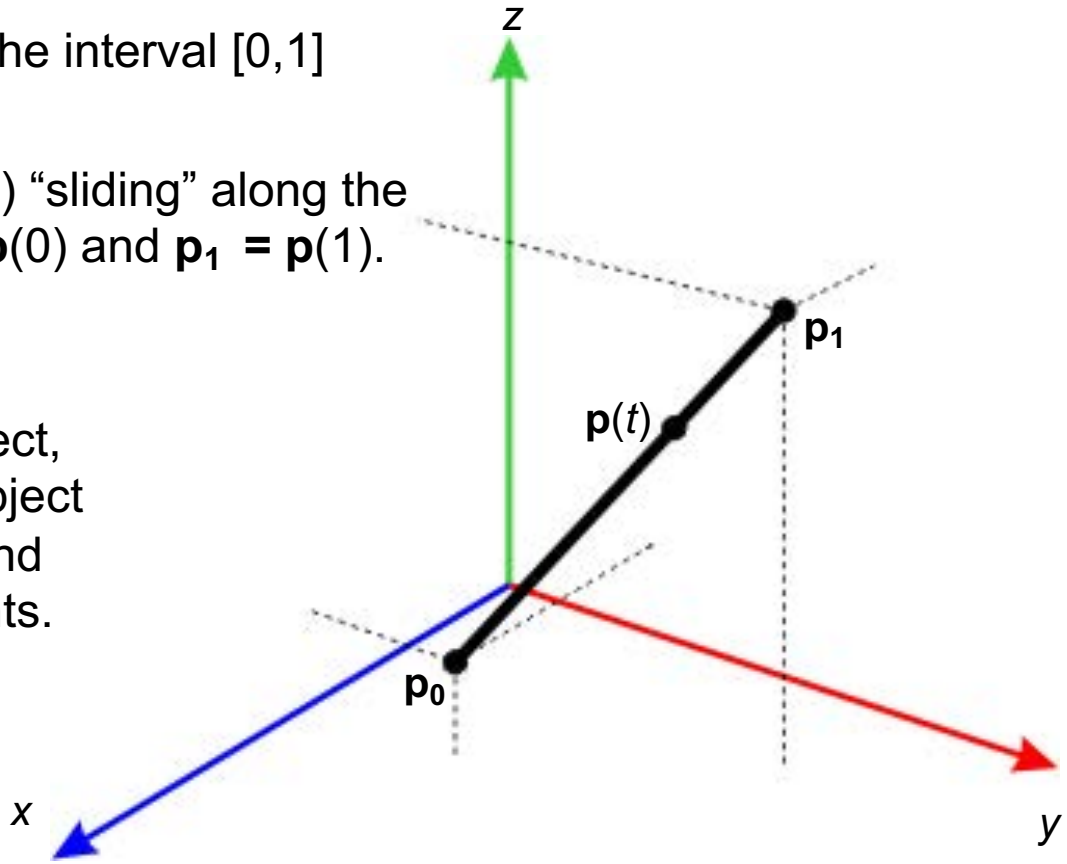
Parametric Curves

You are already familiar with the simplest parametric curve, a straight line segment:

$$\mathbf{p}(t) = \mathbf{p_0} + t(\mathbf{p_1}-\mathbf{p_0}), \text{ with } t \text{ from the interval } [0,1]$$

Moving t between 0 and 1 results in p(t) “sliding” along the segment between the endpoints $\mathbf{p_0} = \mathbf{p}(0)$ and $\mathbf{p_1} = \mathbf{p}(1)$.

If we want to create a “path” for an object, rather than storing each position the object may visit, store only the “way points” and interpolate the position from these points.



Basics of cubic parametric curves

To define a cubic segment

$$\mathbf{Q}(t) = (\mathbf{x}(t), \mathbf{y}(t), \mathbf{z}(t))^{\mathbf{T}}$$

we have 12 coefficients:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$

We limit the value of t to give a finite length curve, i.e., we use $0 <= t <= 1$.

If $\mathbf{T} = (t^3, t^2, t, 1)^{\mathbf{T}}$ and we define a matrix $\mathbf{C}$ as the *coefficient matrix*, then we can write the equation of the curve segment as:

$$\mathbf{Q}(t) = \mathbf{C}\,\mathbf{T}$$

$$C = \begin{pmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \end{pmatrix}$$

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

Why taking cubic and not quadratic or higher order polynomials ?

A cubic polynomial in t is of the form

$$p(t) = at^3 + bt^2 + ct + d$$

It has four coefficients a, b, c, d.

A curve segment of this form allows to interpolate two specified endpoints with specified derivatives at each endpoint.

Lower-degree polynomials give too little flexibility in controlling the shape of the curve, and higher-degree polynomials can introduce unwanted wiggles and require more computation.

Overview

- Parametric curves and cubic polynomials
- **Continuity**
- Hermite Curves
- Bezier Curves
- Splines
- Parametric Bicubic Surfaces

Geometric and parametric continuity of a curve at the joint point between two curve segments

$C^0 = G^0$ geometric continuity: Two curve segments join together.

G^1 geometric continuity: The curves join together and their tangent vectors at the joint point have the same direction (but the curves might have different speeds at the connecting point).

C^1 parametric continuity: The curves join together and their tangent vectors at the joint point are identical (direction, length, orientation).

C^n parametric continuity: The curves join together and the derivatives up to their *n*-th derivative at the joint point are identical.

The tangent of a cubic parametric curve

When two curves join we must ensure that the join "looks smooth"; this can be done by requiring G^1 geometric continuity, that is, that the direction of the tangent at the finish of one curve segment is the same as the direction of the tangent at the start of the next curve segment.

The tangent to the curve **Q(*t*) = C T** is given by the derivative

$x'(t) = 3a_x t^2 + 2b_x t + c_x$

$y'(t) = 3a_y t^2 + 2b_y t + c_y$

$z'(t) = 3a_z t^2 + 2b_z t + c_z$

$$\frac{dQ(t)}{dt} = \begin{pmatrix} x'(t) \\ y'(t) \\ z'(t) \end{pmatrix} = \begin{pmatrix} 0 & 3a_x & 2b_x & c_x \\ 0 & 3a_y & 2b_y & c_y \\ 0 & 3a_z & 2b_z & c_z \end{pmatrix} \begin{pmatrix} t^3 \\ t^2 \\ t \\ 1 \end{pmatrix}$$

That is:

Q'(*t*) = C' T

where **C'** is a new coefficient matrix.

Types of cubic parametric curves: Basis matrix

Generally: To specify a parametric cubic curve we need to find 12 coefficients (matrix **C**). This can be done by solving linear equations; the linear equations are defined in different ways, giving different types of curves.

We will study three main types of parametric cubic curves:

1. Hermite curves: each segment is a parametric cubic curve defined by specifying two endpoints plus specifying the tangents at the two endpoints.

2. Bezier curves: defined by specifying two endpoints plus specifying two "control points", which control the tangents at the two endpoints.

3. Several kinds of **splines**. Each kind is specified by four control points.

For convenience we split up the (3X4) matrix **C** in the equation

$$\mathbf{Q(t) = C\ T}$$

into two matrices:

- The 4X4 **basis matrix M**: This differs from one *type* of curve to another. For example, all Hermite curves have the same basis matrix, but a Hermite curve has a different basis matrix from a Bezier curve.
- The 3X4 **geometry matrix G**: This differs from one curve to another.

The equation is then:

$$\mathbf{Q(t) = G\ M\ T,}$$

i.e. **C = G M**.

Overview

- Parametric curves and cubic polynomials
- Continuity
- **Hermite Curves**
- Bezier Curves
- Splines
- Parametric Bicubic Surfaces

Hermite curves

We use four vectors (2 points, 2 tangent vectors) to specify a Hermite curve segment:

P_1: the start point of the curve segment (**t=0**);

P_4: the finish point of the curve segment (**t=1**);

R_1: the tangent vector at the start of the curve segment (the **t=0** end);

R_4: the tangent vector at the finish of the curve segment (the **t=1** end).

Say:

$P_1 = (P_{1x}, P_{1y}, P_{1z})$

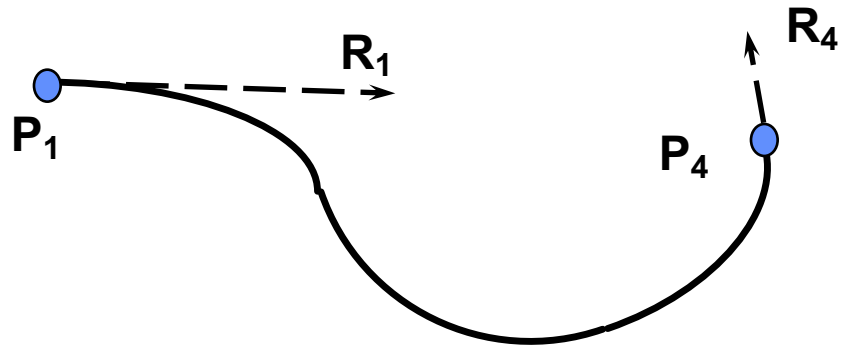
$P_4 = (P_{4x}, P_{4y}, P_{4z})$

$R_1 = (R_{1x}, R_{1y}, R_{1z})$

$R_4 = (R_{4x}, R_{4y}, R_{4z})$

The equation for the curve is **Q(t) = G M T**.

We need to compute a basis matrix **M** and a geometry matrix **G.**



$$G = \begin{pmatrix} P_{1x} & P_{4x} & R_{1x} & R_{4x} \\ P_{1y} & P_{4y} & R_{1y} & R_{4y} \\ P_{1z} & P_{4z} & R_{1z} & R_{4z} \end{pmatrix}$$

$$M = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

Blending functions for Hermite curves

The blending functions are the four components of

M T

for Hermite curves; they are defined as

$$B_1(t) = 2t^3 - 3t^2 + 1$$

$$B_2(t) = -2t^3 + 3t^2$$

$$B_3(t) = t^3 - 2t^2 + t$$

$$B_4(t) = t^3 - t^2$$

These functions are "blended" to give the equation of the curve. The equation of the curve segment can be written as a combination of blending functions, that is:

$$Q(t) = G\,B$$

Where

$$B = (B_1(t), B_2(t), B_3(t), B_4(t))^T.$$

In other words,

$$Q(t) = B_1(t)\,P_1 + B_2(t)\,P_4 + B_3(t)\,R_1 + B_4(t)\,R_4.$$

The blending functions are really just another way of coding the basis matrix.

The basis matrix **M** and the blending functions are constant for all Hermite curves; only the geometry matrix **G** differs from one Hermite curve to another. This helps for efficient implementation.

Note: Hermite curves can be rotated and translated by transforming the geometry matrix **G**, that is, by matrix multiplication.

Implementing Hermite curves

We can evaluate a cubic function in 9 multiplies and 10 additions per 3D point:

$$\mathbf{f(t) = at^3 + bt^2 + ct + d = ((at + b) t + c) t + d}.$$

A naive algorithm to compute the Hermite curve is as here.

1. Input the geometry matrix **G**
2. Compute the coefficient matrix **C = GM**
3. $\mathbf{p_0 = Q(0) = C\ T(0)}$
4. For **t=0** to **t=1** in small steps of Δ:{

    $\mathbf{p_1 = Q(t) = C\ T(t);}$

    draw a line segment from $\mathbf{p_0}$ to $\mathbf{p_1}$;

    $\mathbf{p_0 = p_1}$;

    }

Overview

- Parametric curves and cubic polynomials
- Continuity
- Hermite Curves
- **Bezier Curves**
- Splines
- Parametric Bicubic Surfaces

Bezier Curves

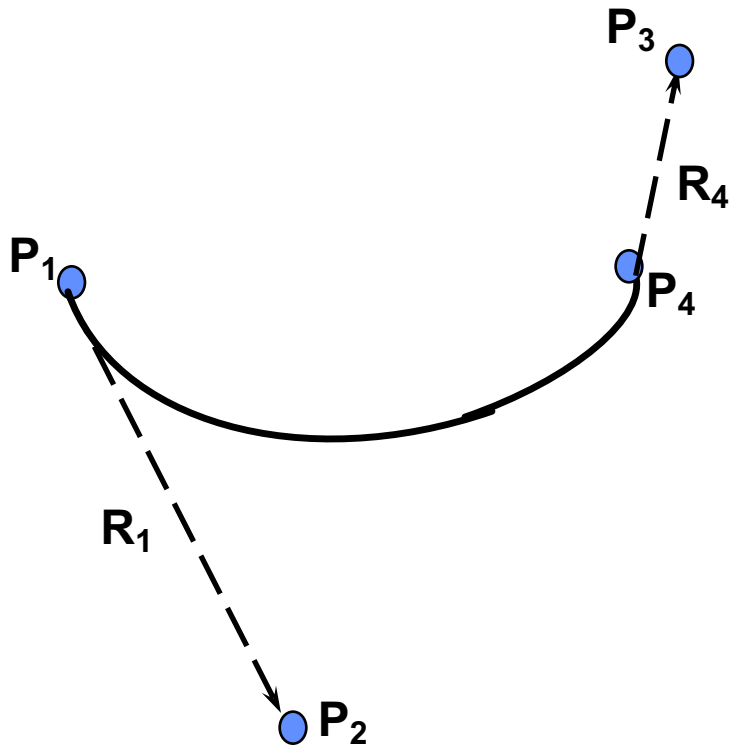
For Bezier curves we specify **four control points**:

- the two endpoints $\mathbf{P_1}$ and $\mathbf{P_4}$.
- two points $\mathbf{P_2}$ and $\mathbf{P_3}$ which are not on the curve but give tangent vectors $\mathbf{R_1}$ and $\mathbf{R_4}$ at the endpoints, by the following equations:

  $\mathbf{R_1 = 3(P_2 - P_1)}$,

  $\mathbf{R_4 = 3(P_4 - P_3)}$.

There are some advantages with specifying the curve in this way (e.g. when joining curve segments; clipping).



Bezier curves

The Bezier geometry matrix is

$G = (P_1, P_2, P_3, P_4)$,

where P_1, P_2, P_3, P_4 are the control points as column vectors.

The Bezier basis matrix is defined as **M**.

$$M = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

The resulting blending functions for Bezier curves are (**B = MT**):

$B_1(t) = (1 - t)^3$

$B_2(t) = 3t\ (1 - t)^2$

$B_3(t) = 3t^2\ (1 - t)$

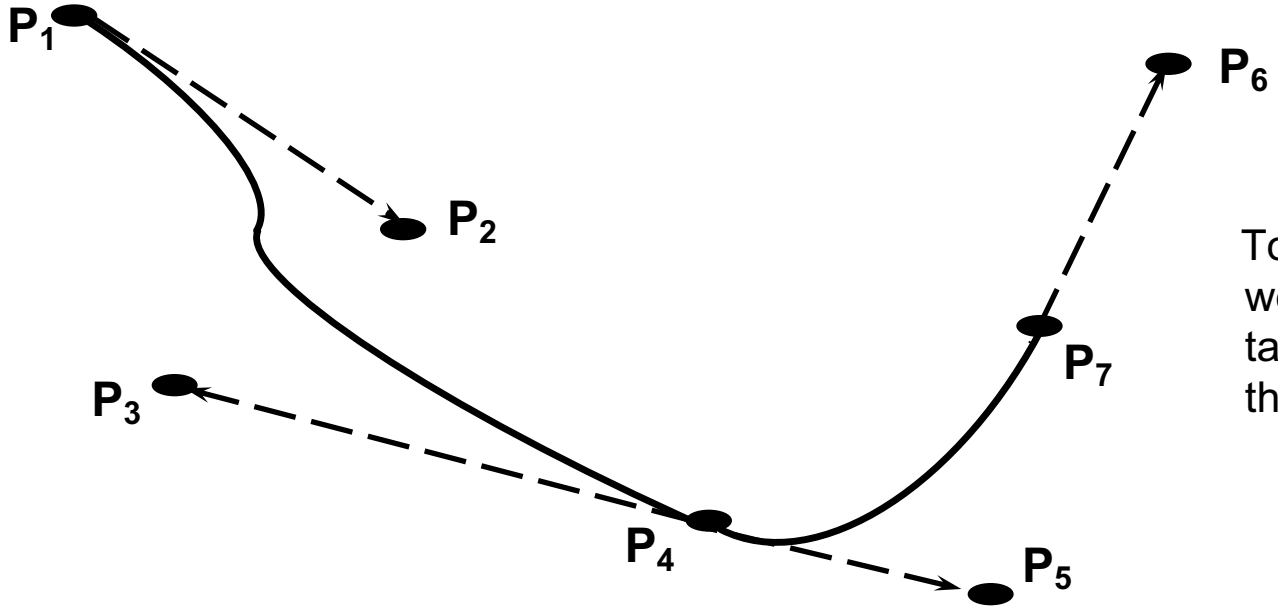
$B_4(t) = t^3$

I.e. we can compute **Q(t) = GB** by the equation

$$Q(t) = B_1(t)\ P_1 + B_2(t)\ P_2 + B_3(t)\ P_3 + B_4(t)\ P_4$$

which is relatively simple.

Joining Bezier curve segments

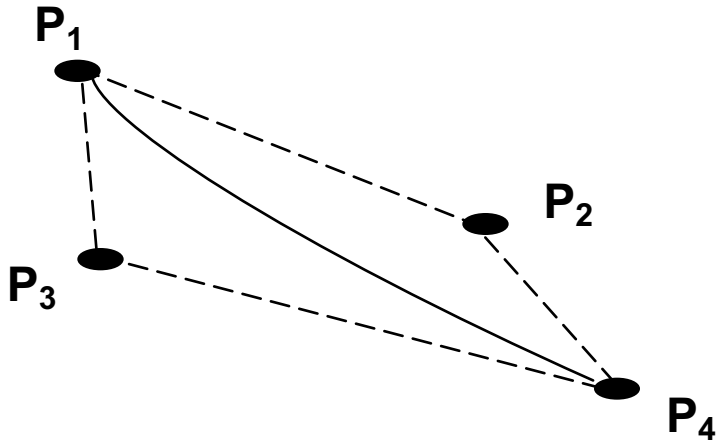


To join two Bezier curves we must ensure that the tangent on one matches the tangent on the next.

Here $\mathbf{P_1}$, $\mathbf{P_2}$, $\mathbf{P_3}$, $\mathbf{P_4}$ are the control points for one Bezier curve segment, and $\mathbf{P_4}$, $\mathbf{P_5}$, $\mathbf{P_6}$, $\mathbf{P_7}$ are the control points for the next.

To ensure G^1 geometric continuity, we must ensure that $\mathbf{P_3P_4P_5}$ is a straight line.

Convex Hulls and Bezier Curves



A Bezier curve always lies within the convex hull of the points which define it; this is because the blending functions

$$B_1(t) = (1-t)^3$$

$$B_2(t) = 3t\,(1-t)^2$$

$$B_3(t) = 3t^2\,(1-t)$$

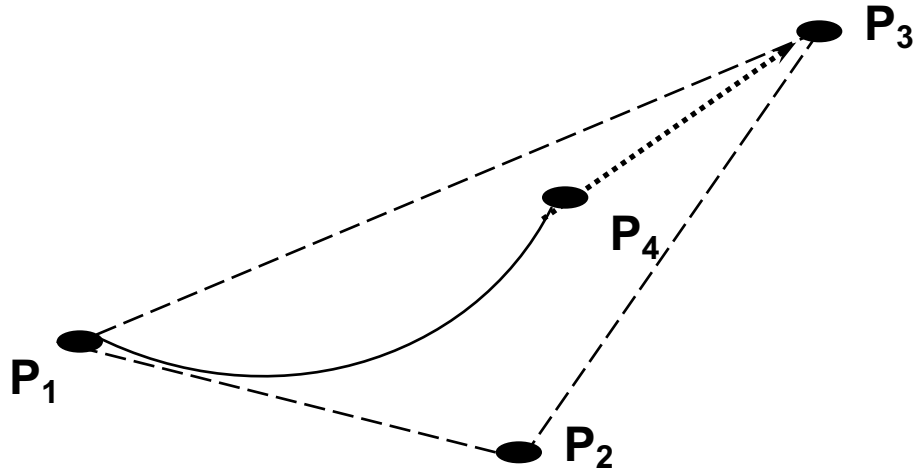
$$B_4(t) = t^3$$

add up to one (check!).

Thus every point on the curve

$$Q(t) = B_1(t)P_1+B_2(t)P_2+B_3(t)P_3+B_4(t)P_4$$

is inside the convex hull of P_1, P_2, P_3, P_4.



This gives a good clipping algorithm for Bezier curves: we can clip to the convex hull of the control points as a pre-processing approximation.

Overview

- Parametric curves and cubic polynomials
- Continuity
- Hermite Curves
- Bezier Curves
- **Splines**
- Parametric Bicubic Surfaces

Splines

B-Splines: Local Control means that polynomial coefficients of curve segments depend only on a few control points.

Historically: Spline = long flexible strips of metal to lay out the surfaces of airplanes, cars and ships.

Mathematically: A spline is a $\mathbf{C^2}$-continuous cubic polynomial that passes through (interpolates) the control points.

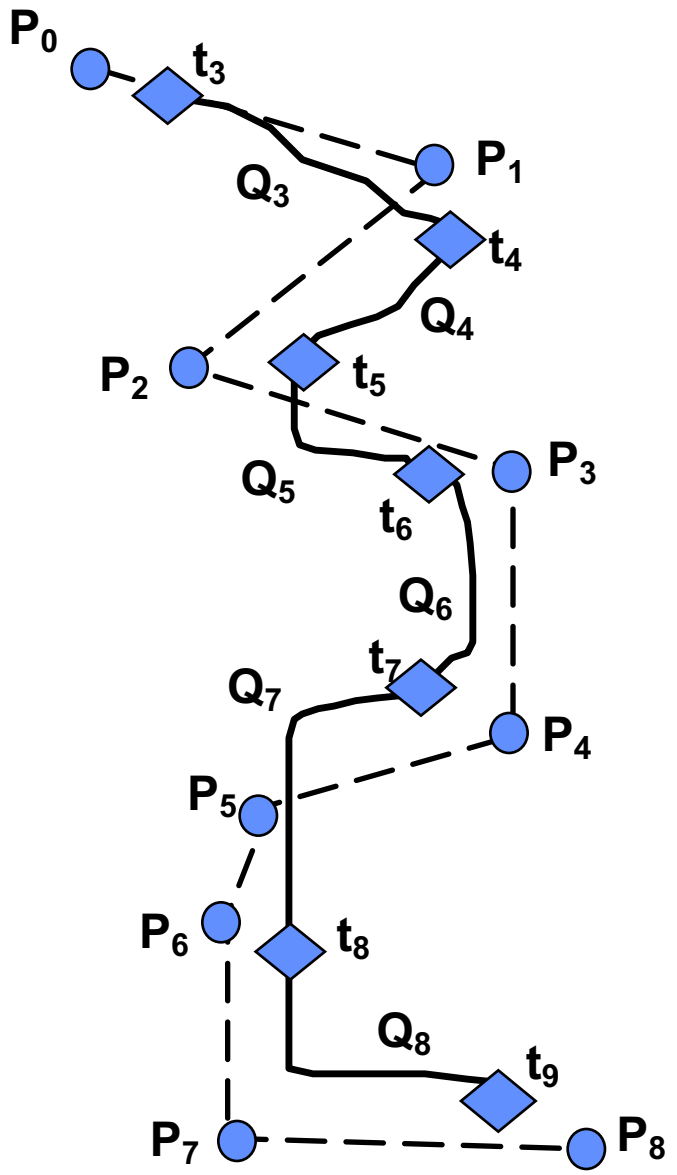
B-splines

B-splines have slightly better continuity properties than Bezier curves.

The control points are usually *not* on the curve.

The curve is made up of cubic segments $\mathbf{Q_3}$, $\mathbf{Q_4}$, …, $\mathbf{Q_m}$, but we discuss the whole curve rather than a single segment:

- **m+1** control points $\mathbf{P_0}$, $\mathbf{P_1}$, $\mathbf{P_2}$, …, $\mathbf{P_m}$.
- **m-2** curve segments $\mathbf{Q_3}$, $\mathbf{Q_4}$, …, $\mathbf{Q_m}$ where $\mathbf{Q_i}$ is defined for $\mathbf{t_i <= t < t_{i+1}}$.
- For each **i >3** there is a ***knot*** between $\mathbf{Q_{i-1}}$ and $\mathbf{Q_i}$ at the parameter value $\mathbf{t_i}$, called the **knot value**.
- The initial and final points at $\mathbf{t_3}$ and $\mathbf{t_{m+1}}$ are also called knots, so there are **m-1** knots in total.



B-splines

Local Control:

Note that each $\mathbf{Q_i}$ is defined by only four control points $\mathbf{P_{i-3}, P_{i-2}, P_{i-1}, P_i}$:

- $\mathbf{Q_3}$ is defined by $\mathbf{P_0, P_1, P_2, P_3,}$
- $\mathbf{Q_4}$ is defined by $\mathbf{P_1, P_2, P_3, P_4,}$
- $\mathbf{Q_5}$ is defined by $\mathbf{P_2, P_3, P_4, P_5,}$
- ...

Each control point $\mathbf{P_i}$ influences four curve segments $\mathbf{Q_i, Q_{i+1}, Q_{i+2}, Q_{i+3}}$. If we move a control point, the curve moves locally.

Uniform:

B-splines have the knots placed at equal **t** intervals; we can assume that $\mathbf{t_3 = 0}$ and $\mathbf{t_{i+1} - t_i = 1}$ for $\mathbf{i > 3}$.

Rational:

B-splines use rational cubic functions for **x(t), y(t) and z(t);** we study non-rational B-splines first.

"B":

"B" stands for basis, because B-splines can be represented as weighted sums of polynomial basis functions (natural splines cannot).

Uniform non-rational B-splines (cont.)

Define

$\mathbf{T_i = (\ (t-t_i)^3\ ,\ (t-t_i)^2\ ,\ (t-t_i)\ ,\ 1\ )^T}$

Then the equation for segment $\mathbf{Q_i}$ is

$$\mathbf{Q_i(t) = G\ M\ T_i}$$

where **G** is the geometry matrix and **M** is the basis matrix.

$$G = (P_{i-3}, P_{i-2}, P_{i-1}, P),$$

$$M = \frac{1}{6}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}$$

$$B_1(t-t_i) = \frac{1}{6}\left[-(t-t_i)^3 + 3(t-t_i)^2 - 3(t-t_i) + 1\right]$$

$$B_2(t-t_i) = \frac{1}{6}\left[3(t-t_i)^3 - 6(t-t_i)^2 + 3(t-t_i)\right]$$

$$B_3(t-t_i) = \frac{1}{6}\left[-3(t-t_i)^3 + 3(t-t_i)\right]$$

$$B_4(t-t_i) = \frac{1}{6}\left[(t-t_i)^3 - 4(t-t_i)^2 + (t-t_i)\right]$$

The blending functions **B** for each $\mathbf{Q_i}$ are given by $\mathbf{MT_i}$, thus or in terms of functions of a local value *t* for each curve segment ($0 < t < 1$),

$$B_1(t) = \frac{1}{6}\left[-t^3 + 3t^2 - 3t + 1\right]$$

$$B_2(t) = \frac{1}{6}\left[3t^3 - 6t^2 + 3t\right]$$

$$B_3(t) = \frac{1}{6}\left[-3t^3 + 3t\right]$$

$$B_4(t) = \frac{1}{6}\left[t^3 - 4t^2 + t\right]$$

Uniform non-rational B-splines (cont.)

Uniform nonrational B-splines:

- The joins or knots between the uniform nonrational B-splines are *"C^2 continuous",* that is, the second derivative is continuous.
- The curve is a little more difficult to control than Bezier curves because the control points are not on the curve.
- However, the curve still lies within the convex hull of the control points.

Nonuniform nonrational B-splines:

- The knots are not necessarily equally spaced.
- Continuity can be varied: it can be $\mathbf{C^2}$, can be $\mathbf{C^1}$, can even have no continuity.
- Interesting effects can be induced by using multiple knots at the same point; for instance, we can make parts of the curve into straight line segments, and control the continuity.

Other B-splines

Non-uniform rational cubic polynomial curve segments

- For these curves we use ratios of polynomials:

  $\mathbf{x(t) = X(t)/W(t),}$

  $\mathbf{y(t) = Y(t)/W(t),}$

  $\mathbf{z(t) = Z(t)/W(t).}$

- We can think of this as a curve in homogeneous coordinates, i.e., in 4 dimensions:

  $\mathbf{Q(t) = (X(t) , Y(t) , Z(t) , W(t))^T.}$

- The polynomials **X(t) , Y(t) , Z(t) , W(t)** can be Hermite curves, Bezier curves, or B-splines.
- If they are B-splines then they are called **NURBS**: non-uniform rational B-splines.

These curves are invariant under rotations, scaling, translation and even perspective transformation of the control points. Thus to transform the curves, we only need to transform their control points.

NURBS also can define conic sections (circles, ellipses, parabolas, etc) precisely.

This is useful in any solid modelling system.

Overview

- Parametric curves and cubic polynomials
- Continuity
- Hermite Curves
- Bezier Curves
- Splines
- **Parametric Bicubic Surfaces**

Parametric Bicubic Surfaces

Surfaces are a simple generalisations of curves.

$$\mathbf{Q(s,t) = T^T M^T G M S}$$

where

Q(s,t) is a 4X1 vector, with each component a function of **s** and **t** that is cubic in both **s** and **t**.

T is the vector $\mathbf{(t^3, t^2, t, 1)^T}$

S is the vector $\mathbf{(s^3, s^2, s, 1)^T}$

M is a 4x4 basis matrix: this varies depending on whether the surface is specified as a Hermite surface, a Bezier surface, or a B-spline surface.

G is the geometry vector, defined by control points.

Note that for a fixed value of $\mathbf{t = t_0}$, we have

$$\mathbf{Q(s, t_0) = G(t_0) M S,}$$

where $\mathbf{G(t_0) = T^T(t_0) M^T G}$, that is, $\mathbf{G(t_0) = (G^T M T (t_0))^T}$

Thus as we vary **t**, we get a parametric cubic curve. The parametric bicubic surface is formed by dragging this curve along another parametric cubic curve.

We can define:

– Hermite surfaces

– Bezier surfaces

– B-spline surfaces

in this way.

Normals (= normal vectors) to surfaces

It is important to compute normals to surfaces for use in

- interference detection in robotics
- calculating offsets in numerically controlled machining
- hidden surface elimination
- shading surfaces by several means

The equations for the derivatives in both directions are here.

This gives tangent vectors in two directions, that is, a plane tangent to the surface. Thus the normal to the plane is the cross product of these two vectors.

The normal is a biquintic (two variable, fifth degree) polynomial and is expensive to compute. It is possible to find cheaper approximations.

$$\frac{\partial}{\partial s}Q(s,t) = \frac{\partial}{\partial s}\left(T^T M^T GMS\right)$$

$$= T^T M^T GM\frac{\partial}{\partial s}(S)$$

$$= T^T M^T GM\left(3s^2,2s,1,0\right)^T$$

$$\frac{\partial}{\partial t}Q(s,t) = \frac{\partial}{\partial t}\left(T^T M^T GMS\right)$$

$$= \frac{\partial}{\partial t}\left(T^T\right)M^T GMS$$

$$= \left(3t^2,2t,1,0\right)^T M^T GMS$$

Displaying bicubic surfaces

There are many ways to display a surface. Here we will just describe a method which draws curves in both the **s** and **t** direction:

/* First run through values of **s** between **0** and **1** in small increments Δ **= 1/(k-1)** */

For **i = 0 to k-1**

draw the curve

$\mathbf{Q(i\Delta, t) = T^T M^T G M S(i\Delta);}$

/* Next run through values of **t** between **0** and **1** in small increments Δ **= 1/(k-1)** */

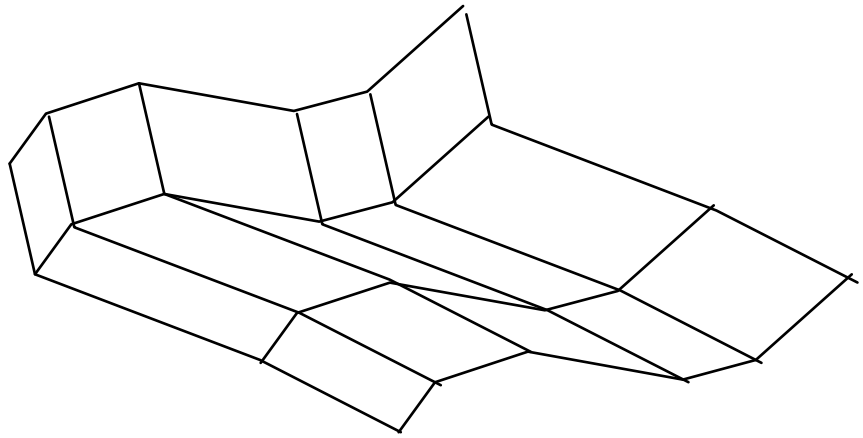
For **i = 0 to k-1**

draw the curve

$\mathbf{Q(s, i\Delta) = T^T(i\Delta) M^T G M S;}$

The curves are drawn in the same way as we drew the parametric bicubic curves.

This technique is fairly inefficient and we must use clever ways to update rather than recompute the functions to achieve reasonable performance.



Literature

This topic is covered frequently in literature.

Two possible sources are:

Peter Shirley and Steve Marschner: Fundamentals of computer graphics. AK Peters, 2009. Chapter 15 Curves.

F.S. Hill and S.M Kelley: Computer graphics using OpenGL. Prentice-Hall, 3rd edition, 2007. Chapter 10.

Videos

B-Splines

https://www.youtube.com/watch?v=X0uurzdHzHI

NURBS

https://www.youtube.com/watch?v=Lm1G5jJ6JC8