Representing Curved Objects

- So far we’ve seen
  - Polygonal objects (triangles) and Spheres
- Now, polynomial curves
  - Hermite curves
  - Bezier curves
  - B-Splines
  - NURBS
- Bivariate polynomial surface patches
Beyond linear approximation

Instead of approximating everything by zillions of lines and planes, it is possible to approximate shapes using higher-order curves. Advantages:

• More compact

• Reduces “artifacts”

Use of sphere in ray tracer is an example of an implicit curve.
The Pen Metaphor

• Think of putting a pen to paper
• Pen position described by time $t$

$x(t), y(t)$

Seeing the action of drawing is the key, so this static drawing only partly captures the point of this slide.
Design Criteria

• Local control of shape
• Smoothness and continuity
• Ability to evaluate derivatives
• Stability
• Ease of rendering
Review Forms - Explicit

• Explicit representation: $y = f(x)$

\[ y = ax^3 + bx^2 + cx + d \]

• Drawbacks:
  – Multiple values of $y$ for a single $x$ impossible.
  – Not rotationally invariant.
Review Forms - Implicit

• Implicit representation: \( f(x, y, z) = 0 \)

\[
\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0
\]

• Advantages:
  – On curve & relative distance to curve tests.

• Drawbacks:
  – Enumerating points on the curve is hard.
  – Extra constraints needed – half a circle?
  – Difficult to express and test tangents.
Parametric Representations

We will represent 3D curves using a parametric representation, introducing a new variable $t$:

$$Q(t) = \begin{vmatrix} x(t) & y(t) & z(t) \end{vmatrix}$$

Note that $x$, $y$ and $z$ are dependent on $t$ alone, making it clear that there is only one free variable.

Think of $t$ as time associated with movement along the curve.
Third Order Curves

• Third-order functions are the standard:

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

• Why 3?
  – Lower-order curves cannot be smoothly joined.
  – Higher-order curves introduce “wiggles”.

• Without loss of generality: 0 \leq t \leq 1.
Cubic Examples

\[ x^3 + 40x^2 + 10x + 2 \]

\[ -x^3 + 40x^2 - 10x + 2 \]
Notation

\[ T = \begin{bmatrix} t^3, t^2, t, 1 \end{bmatrix} \]

\[ Q(t) = T \cdot C \]

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

Alternatively:

\[ Q(t)^T = C^T \cdot T^T \]
The derivative of $Q(t)$ is its tangent:

$$\frac{d}{dt} Q(t) = \left[ \frac{d}{dt} x(t), \frac{d}{dt} y(t), \frac{d}{dt} z(t) \right]$$

$$\frac{d}{dt} Q(t)_x = 3a_xt^2 + 2b_xt + c_x$$

$$\frac{d}{dt} Q(t) = \left[ 3t^2, 2t, 1, 0 \right] C$$

The same matrix as on previous slide

Again the time metaphor is useful, the tangent indicates instantaneous direction and speed.
We want curves that fit together smoothly. To accomplish this, we would like to specify a curve by providing:

- The endpoints
- The 1st derivatives at the endpoints

The result is called a *Hermite Curve*. 
Since $Q(t) = TC$, we factor $C$ into two matrices:
- $G$ (a 3x4 geometry matrix)
- $M$ (a 4x4 basis matrix)

such that $C = G \cdot M$.

Note: $G$ will hold our geometric constraints (endpoints and derivatives), while $M$ will be constant across all Hermite curves.

This step is a big deal. It makes thinking about curve geometry tractable.
Let us concentrate on the $x$ component:

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

Remember that its derivative is:

$$\frac{d}{dt} P(t)_x = 3a_x t^2 + 2b_x t + c_x$$

Therefore

$$\begin{bmatrix}
P(0)_x \\
P(1)_x \\
d/dt P(0)_x \\
d/dt P(1)_x \\
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 \\
3 & 2 & 1 & 0 \\
\end{bmatrix} \begin{bmatrix}
a_x \\
b_x \\
c_x \\
d_x \\
\end{bmatrix}$$
Therefore:

\[
\begin{bmatrix}
    a_x \\
    b_x \\
    c_x \\
    d_x \\
\end{bmatrix}
= \begin{bmatrix}
    0 & 0 & 0 & 1 \\
    1 & 1 & 1 & 1 \\
    0 & 0 & 1 & 0 \\
    3 & 2 & 1 & 0 \\
\end{bmatrix}
\begin{bmatrix}
    -1 \\
    P(0)_x \\
    P(1)_x \\
    d/dt P(0)_x \\
    d/dt P(1)_x \\
\end{bmatrix}
\]

And taking the inverse:

\[
\begin{bmatrix}
    a_x \\
    b_x \\
    c_x \\
    d_x \\
\end{bmatrix}
= \begin{bmatrix}
    2 & -2 & 1 & 1 \\
    -3 & 3 & -2 & -1 \\
    0 & 0 & 1 & 0 \\
    1 & 0 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
    P(0)_x \\
    P(1)_x \\
    d/dt P(0)_x \\
    d/dt P(1)_x \\
\end{bmatrix}
\]
The Hermite Matrix

OK, that was the x dimension. How about the others?

They are, of course, the same:

\[
\begin{bmatrix}
  a_x & a_y & a_z \\
  b_x & b_y & b_z \\
  c_x & c_y & c_z \\
  d_x & d_y & d_z \\
\end{bmatrix} =
\begin{pmatrix}
  2 & -2 & 1 & 1 \\
  -3 & 3 & -2 & -1 \\
  0 & 0 & 1 & 0 \\
  1 & 0 & 0 & 0 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
  P(0) \\
  P(1) \\
  \frac{dP(0)}{dt} \\
  \frac{dP(1)}{dt} \\
\end{pmatrix}
\]
Punchline

\[ Q(t) = \begin{vmatrix} x(t) & y(t) & z(t) \end{vmatrix} = T \cdot C = T \cdot M_H \cdot \begin{bmatrix} P(0) \\ P(1) \\ \frac{d}{dt} P(0) \\ \frac{d}{dt} P(1) \end{bmatrix} \]

Since \( M_H \) and \( T \) are known, you can write down a cubic polynomial curve by inspection ending at points \( P(0) \) and \( P(1) \) with tangents \( \frac{d}{dt} P(0) \) and \( \frac{d}{dt} P(1) \).
Recall Parametric Equation: \( Q(t) = T \ M_H \ G \)

Where

\[
Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ dx1 & dy1 & dz1 \\ dx2 & dy2 & dz2 \end{bmatrix}
\]

Fully Expanded (note transpose)

\[
Q(t)^T = \begin{bmatrix} (2 \ t^3 - 3 \ t^2 + 1) \ x1 + (3 \ t^2 - 2 \ t^3) \ x2 + (-2 \ t^2 + t^3 + t) \ dx1 + (-t^2 + t^3) \ dx2 \\ (2 \ t^3 - 3 \ t^2 + 1) \ y1 + (3 \ t^2 - 2 \ t^3) \ y2 + (-2 \ t^2 + t^3 + t) \ dy1 + (-t^2 + t^3) \ dy2 \\ (2 \ t^3 - 3 \ t^2 + 1) \ z1 + (3 \ t^2 - 2 \ t^3) \ z2 + (-2 \ t^2 + t^3 + t) \ dz1 + (-t^2 + t^3) \ dz2 \end{bmatrix}
\]
If you Prefer

• There are two equivalent setups
• The difference is solely transposition

\[ Q(t) = G M T \]

\[
Q(t) = \begin{bmatrix}
  x1 & x2 & dx1 & dx2 \\
  y1 & y2 & dy1 & dy2 \\
  z1 & z2 & dz1 & dz2
\end{bmatrix}
\begin{bmatrix}
  2 & -3 & 0 & 1 & t^3 \\
  -2 & 3 & 0 & 0 & t^2 \\
  1 & -2 & 1 & 0 & t \\
  1 & -1 & 0 & 0 & 1
\end{bmatrix}
\]

\[
Q(t) = \begin{bmatrix}
  (2 t^3 - 3 t^2 + 1) x1 + (3 t^2 - 2 t^3) x2 + (-2 t^2 + t^3 + t) dx1 + (-t^2 + t^3) dx2 \\
  (2 t^3 - 3 t^2 + 1) y1 + (3 t^2 - 2 t^3) y2 + (-2 t^2 + t^3 + t) dy1 + (-t^2 + t^3) dy2 \\
  (2 t^3 - 3 t^2 + 1) z1 + (3 t^2 - 2 t^3) z2 + (-2 t^2 + t^3 + t) dz1 + (-t^2 + t^3) dz2
\end{bmatrix}
\]
Examples

\[ G = \begin{pmatrix} 100 & 100 & 0 \\ 200 & 200 & 0 \\ 10 & 10 & 0 \end{pmatrix} \]

\[ G = \begin{pmatrix} 100 & 100 & 0 \\ 200 & 200 & 0 \\ 0 & 200 & 0 \\ 200 & 0 & 0 \end{pmatrix} \]
More Examples

\[
G = \begin{pmatrix}
100 & 100 & 0 \\
200 & 200 & 0 \\
0 & 1000 & 0 \\
1000 & 0 & 0
\end{pmatrix}
\]

\[
G = \begin{pmatrix}
100 & 100 & 0 \\
200 & 200 & 0 \\
100 & 2000 & 0 \\
-500 & -200 & 0
\end{pmatrix}
\]
Hermite Blending Functions

- Conceptual Realignment
  - Curves are weighted averages of points/vectors.
  - Blending functions specify the weighting.

\[
Q = \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} (2t^3 - 3t^2 + 1) + \begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix} (-2t^3 + 3t^2) + \begin{bmatrix} dx1 \\ dy1 \\ dz1 \end{bmatrix} (t^3 - 2t^2 + t) + \begin{bmatrix} dx2 \\ dy2 \\ dz2 \end{bmatrix} (t^3 - t^2)
\]

- \( Bh_1 = 2t^3 - 3t^2 + 1 \)
- \( Bh_2 = -2t^3 + 3t^2 \)
- \( Bh_3 = t^3 - 2t^2 + t \)
- \( Bh_4 = t^3 - t^2 \)
From Hermite to Bezier

What’s wrong with Hermite curves?

Nothing, unless you are using a point-and-click interface

Bezier curves are like Hermite curves, except that the user specifies four points \((p_1, p_2, p_3, p_4)\). The curve goes through \(p_1 \) & \(p_4\). Points \(p_2 \) & \(p_3\) specify the tangents at the endpoints.
More Precisely....

\[ R_1 = \frac{d}{dt} P_1 = 3(P_2 - P_1) \quad R_4 = \frac{d}{dt} P_4 = 3(P_4 - P_3) \]

Tangents at start and end are now defined by intermediate points.

Q: Why ‘3’? Why not \( R_1 = (P_2 - P_1) \)?

A: Think of 4 evenly spaced points in a line.
The Hermite geometry matrix is related to the Bezier geometry matrix by:

\[ G_H = \begin{pmatrix} P1 \\ P4 \\ R1 \\ R4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{pmatrix} \begin{pmatrix} P1 \\ P2 \\ P3 \\ P4 \end{pmatrix} = M_{HB}G_B \]
Hermite → Bezier

For Hermite curves, \( Q(t) = T M_H G_H \), where \( G_H = [P_1, P_4, R_1, R_4]^T \), \( T = [t^3, t^2, t, 1] \)

\[
M_H = \begin{vmatrix}
2 & -2 & 1 & 1 \\
-3 & 3 & -2 & -1 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 \\
\end{vmatrix}
\]

So, \( Q(t) = T M_H M_{HB} G_B \)
The Bezier Basis Matrix

\[ Q(t) = T(M_H M_{HB})G_B \]

\[ M_B = M_H M_{HB} = \begin{vmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{vmatrix} \]

\[ Q(t) = TM_B G_B \]

See page 364 to connect with description in our textbook.
The Bezier Blending Functions

$$Q(t) = \begin{bmatrix} (-t^3 - 3t^2 + 3t^2 + 1)x_1 + (3t^2 - 6t^2 + 3t^3)x_2 + (3t^2 - 3t^3)x_3 + t^3x_4 \\ (-t^3 - 3t^2 + 3t^2 + 1)y_1 + (3t^2 - 6t^2 + 3t^3)y_2 + (3t^2 - 3t^3)y_3 + t^3y_4 \\ (-t^3 - 3t^2 + 3t^2 + 1)z_1 + (3t^2 - 6t^2 + 3t^3)z_2 + (3t^2 - 3t^3)z_3 + t^3z_4 \end{bmatrix}$$

$$Q(t) = P_1(-t^3 + 3t^2 - 3t + 1) + P_2(3t^3 - 6t^2 + 3t) + P_3(-3t^3 + 3t^2) + P_4(t^3)$$
Add them up.

\[
\begin{align*}
( -t^3 &+ 3t^2 - 3t + 1) \\
+ (3t^3 &- 6t^2 + 3t) \\
+ (-3t^3 &+ 3t^2) \\
+ (t^3) & \\
\hline
0t^3 + 0t^2 + 0t + 1
\end{align*}
\]
If you graph the four Bezier blending functions for $t=0$ to $t=1$, you find that they are always positive and always sum to one.

Therefore, *the Bezier curve stays within the convex hull defined by $P_1$, $P_2$, $P_3$ & $P_4$.***
Examples 1.

\[ G^T = \begin{bmatrix} 100 & 100 & 0 \\ 103 & 103 & 0 \\ 197 & 197 & 0 \\ 200 & 200 & 0 \end{bmatrix} \]

\[ G^T = \begin{bmatrix} 100 & 100 & 0 \\ 100 & 150 & 0 \\ 150 & 200 & 0 \\ 200 & 200 & 0 \end{bmatrix} \]
Examples 2.

\[ G^T = \begin{bmatrix} 100 & 100 & 0 \\ 100 & 300 & 0 \\ -100 & 200 & 0 \\ 200 & 200 & 0 \end{bmatrix} \]

\[ G^T = \begin{bmatrix} 100 & 100 & 0 \\ 0 & 200 & 0 \\ 201 & 200 & 0 \\ 101 & 100 & 0 \end{bmatrix} \]
Example 3D

\[ P1 = (0, 0, 0) \]
\[ P2 = (100, 0, 0) \]
\[ P3 = (0, 100, 100) \]
\[ P4 = (0, 0, 100) \]
Bezier Curves are Common

• You have probably already used them.
• For example, in PowerPoint
  – Build a shape
  – Then select edit points
  – Notice the control ‘wings’
• Enhancements
  – Ways to introduce constraints
    • Smooth Point
    • Straight Point
    • Corner Point
Stepping Back

What should you be learning? Should you memorize $M_H$ and $M_B$? No! That’s what reference books are for.

You should know what $G_H$ and $G_B$ are. You should know how to derive $M_H$ from the parametric form of the cubic equations. You should know how to derive $M_B$ from $M_H$. If you understand these concepts, you can look up or rederive the matrices as necessary.
Now for something completely different.

There is another way to motivate curves.

Let's say that I have four control points.
To find the midpoint of the curve corresponding to those control points:

Connect the point between $P_1$ and $P_2$ where $t = 0.5$ with the point between $P_3$ & $P_2$ where $t = 0.5$; Do the same with the $P_2$-$P_3$ & $P_3$-$P_4$ lines
Now, connect these two lines at their $t = 0.5$ points

The $t = 0.5$ point on the resulting segment is the midpoint ($t = 0.5$ point) of some type of curve which is made up of weighted averages of the control points (we’ll soon see what kind of curve)
We can extend this idea to any $t$ value.

To compute the $t = 0.25$ point, connect the 0.25 points of the original lines...
Now, connect these lines at their $t = 0.25$ locations, and find the point where $t = 0.25$ of the resulting line.

In this way, you can compute the 3rd-order curve for any value of $t$. 

$P_1$, $P_2$, $P_3$, $P_4$
Algebraic Definition

• The equations of the three original lines are:
  \[ A_1(t) = (1 - t) P_1 + t P_2 \]
  \[ A_2(t) = (1 - t) P_2 + t P_3 \]
  \[ A_3(t) = (1 - t) P_3 + t P_4 \]

• The equations of the next two joining lines are:
  \[ B_1(t) = (1 - t) A_1 + t A_2 \]
  \[ B_2(t) = (1 - t) A_2 + t A_3 \]

• Finally, the line between the two joining lines is:
  \[ C_1(t) = (1 - t) B_1 + t B_2 \]
Begin Substitutions

• Substitute equations for $A_1$ and $A_2$ into $B_1$

\[
B_1(t) = (1 - t) \left( ((1 - t) P_1 + t P_2) + t ((1 - t) P_2 + t P_3) \right)
\]
\[
= P_1 - 2 t P_1 + t^2 P_1 + 2 t P_2 - 2 t^2 P_2 + t^2 P_3
\]
\[
= (t^2 + 1 - 2 t) P_1 + (-2 t^2 + 2 t) P_2 + t^2 P_3
\]

*Factoring the result*

\[
B_1(t) = (t - 1)^2 P_1 - 2 t (t - 1) P_2 + t^2 P_3
\]

• Likewise, substitute $A_2$ and $A_3$ into $B_2$

\[
B_2(t) = (t - 1)^2 P_2 - 2 t (t - 1) P_3 + t^2 P_4
\]
Resulting Third Order Curve

Substitute equations for $B_1$ and $B_2$ into $C_1$

$$C_1(t) = (1 - t) \left((t - 1)^2 P_1 - 2t (t - 1) P_2 + t^2 P_3 \right)$$

$$+ t \left((t - 1)^2 P_2 - 2t (t - 1) P_3 + t^2 P_4 \right)$$

$$= (-3t - t^3 + 3t^2 + 1) P_1$$

$$+ (3t^3 - 6t^2 + 3t) P_2$$

$$+ (-3t^3 + 3t^2) P_3 + t^3 P_4$$

And After Factoring

$$C_1(t) = (1 - t)^3 P_1 + 3t (t - 1)^2 P_2 - 3t^2 (t - 1) P_3 + t^3 P_4$$
Bezier = de Casteljau

But those last four functions are exactly the Bezier blending functions!

The recursive line intersection algorithm can therefore be used to gain intuition about the behavior of Bezier functions

Not something completely different afterall.