Characters and Objects

• Important for composing the scene
• Need to design and model them in the first place
Curves / curved surfaces

Can produce smooth surfaces with less parameters

- Easier to design
- Can efficiently preserve complex structures
Today

• Parametric curves
  – Introduction
  – Hermite curves
  – Bezier curves
  – Uniform cubic B-splines
  – Catmull-Rom spline

• Bicubic patches

• Tessellation
  – Adaptive tessellation
Types of Curves and Surfaces

- Explicit:
  \[ y = mx + b \quad \text{and} \quad r = Arx + Br^2y + C_r \]

- Implicit:
  \[ Ax + By + C = 0 \quad \text{and} \quad (x-x_0)^2 + (y-y_0)^2 - r^2 = 0 \]

- Parametric:
  \[ x = x_0 + (x_1 - x_0)t \quad \text{and} \quad x = x_0 + r\cos\theta \]
  \[ y = y_0 + (y_1 - y_0)t \quad \text{and} \quad y = y_0 + r\sin\theta \]
Why parametric?

- Simple and flexible
- The function of each coordinates can be defined independently.
  
  \[(x(t), y(t)) : 1D \text{ curve in } 2D \text{ space} \]
  \[(x(t), y(t), z(t)) : 1D \text{ curve in } 3D \text{ space} \]
  \[(x(s,t), y(s,t), z(s,t)) : 2D \text{ surface in } 3D \text{ space} \]
- Polynomial are suitable for creating smooth surfaces with less computation

\[x(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0\]
Today

• Parametric curves
  – Introduction
  – Hermite curves
  – Bezier curves
  – Uniform cubic B-splines
  – Catmull-Rom spline
• Bicubic patches
• Tessellation
  – Adaptive tessellation
Hermite curves

- A cubic polynomial
  \[ x(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0 \]
- \( t \) ranging from 0 to 1
- Polynomial can be specified by the position of, and gradient at, each endpoint of curve.
Family of Hermite curves.

Note:
Start points on left.

http://www.inf.ed.ac.uk/teaching/courses/cg/d3/hermite.html
http://www.rose-hulman.edu/~finn/CCLI/Applets/CubicHermiteApplet.html
Finding Hermite coefficients

Can solve them by using the boundary conditions

\[ X(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0, \quad X'(t) = 3a_3 t^2 + 2a_2 t + a_1 \]

Substituting for \( t \) at each endpoint:

\[ x_0 = X(0) = a_0 \]
\[ x_0' = X'(0) = a_1 \]
\[ x_1 = X(1) = a_3 + a_2 + a_1 + a_0 \]
\[ x_1' = X'(1) = 3a_3 + 2a_2 + a_1 \]

And the solution is:

\[ a_0 = x_0 \]
\[ a_1 = x_0' \]
\[ a_2 = -3x_0 - 2x_0' + 3x_1 - x_1' \]
\[ a_3 = 2x_0 + x_0' - 2x_1 + x_1' \]

\[ X(t) = \left(2x_0 + x_0' - 2x_1 + x_1'\right) t^3 + \left(-3x_0 - 2x_0' + 3x_1 - x_1'\right) t^2 + (x_0') t + x_0 \]
Finding Hermite coefficients

Can solve them by using the boundary conditions

\[ X(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0, \quad X'(t) = 3a_3 t^2 + 2a_2 t + a_1 \]

Substituting for \( t \) at each endpoint:

\[
\begin{align*}
  x_0 &= X(0) = a_0 \\
  x_0' &= X'(0) = a_1 \\
  x_1 &= X(1) = a_3 + a_2 + a_1 + a_0 \\
  x_1' &= X'(1) = 3a_3 + 2a_2 + a_1
\end{align*}
\]

And the solution is:

\[
\begin{align*}
  a_0 &= x_0 \\
  a_1 &= x_0' \\
  a_2 &= -3x_0 - 2x_0' + 3x_1 - x_1' \\
  a_3 &= 2x_0 + x_0' - 2x_1 + x_1' \\
  X(t) &= (2x_0 + x_0' - 2x_1 + x_1') t^3 + (-3x_0 - 2x_0' + 3x_1 - x_1') t^2 + (x_0') t + x_0
\end{align*}
\]
Finding Hermite coefficients

Can solve them by using the boundary conditions

\[ X(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0, \quad X'(t) = 3a_3 t^2 + 2a_2 t + a_1 \]

Substituting for \( t \) at each endpoint:

\[ x_0 = X(0) = a_0 \quad x_0' = X'(0) = a_1 \]
\[ x_1 = X(1) = a_3 + a_2 + a_1 + a_0 \quad x_1' = X'(1) = 3a_3 + 2a_2 + a_1 \]

And the solution is:

\[
\begin{align*}
    a_0 &= x_0 & a_1 &= x_0' \\
    a_2 &= -3x_0 - 2x_0' + 3x_1 - x_1' & a_3 &= 2x_0 + x_0' - 2x_1 + x_1' \\
    X(t) &= (2x_0 + x_0' - 2x_1 + x_1') t^3 + (-3x_0 - 2x_0' + 3x_1 - x_1') t^2 + (x_0') t + x_0
\end{align*}
\]
The Hermite matrix: $M_H$

The resultant polynomial can be expressed in matrix form:

$$X(t) = t^T M_H q$$

( $q$ is the control vector)

$$X(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & -2 & 1 \\ -3 & -2 & 3 & -1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_0' \\ x_1 \\ x_1' \end{bmatrix}$$

We can now define a parametric polynomial for each coordinate required independently, ie. $X(t)$, $Y(t)$ and $Z(t)$
Hermite Basis (Blending) Functions

\[ X(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & -2 & 1 \\ -3 & -2 & 3 & -1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_0' \\ x_1 \\ x_1' \end{bmatrix} \]

\[ = (2t^3 - 3t^2 + 1)x_0 + (t^3 - 2t^2 + t)x_0' + (-2t^3 + 3t^2)x_1 + (t^3 - t^2)x_1' \]
Hermite Basis (Blending) Functions

\[ X(t) = (2t^3 - 3t^2 + 1)x_0 + (t^3 - 2t^2 + t)x'_0 + (-2t^3 + 3t^2)x_1 + (t^3 - t^2)x'_1 \]

The graph shows the shape of the four basis functions – often called blending functions.

They are labelled with the elements of the control vector that they weight.

Note that at each end only position is non-zero, so the curve must touch the endpoints.
Bézier Curves

- Hermite cubic curves are difficult to model – need to specify point and gradient.
- Paul de Casteljau who was working for Citroën, invented another way to compute the curves.
- Publicised by Pierre Bézier from Renault.
- By only giving points instead of the derivatives.
Bézier Curves (2)

Can define a curve by specifying 2 endpoints and 2 additional control points

The two middle points are used to specify the gradient at the endpoints

Fit within the convex hull by the control points

http://www.inf.ed.ac.uk/teaching/courses/cg/d3/bezier.html

http://www.rose-hulman.edu/~finn/CCLI/Applets/BezierBernsteinApplet.html
Bézier Matrix

• The cubic form is the most popular
  \[ X(t) = t^T M_B q \quad (M_B \text{ is the Bézier matrix}) \]

• With \( n=4 \) and \( r=0,1,2,3 \) we get:

\[
X(t) = \begin{bmatrix}
    t^3 & t^2 & t & 1
\end{bmatrix}
\begin{bmatrix}
    -1 & 3 & -3 & 1 \\
    3 & -6 & 3 & 0 \\
    -3 & 3 & 0 & 0 \\
    1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
    q_0 \\
    q_1 \\
    q_2 \\
    q_3
\end{bmatrix}
\]

\[
X(t) = (-t^3 + 3t^2 - 3t + 1)q_0 + (3t^3 - 6t^2 + 3t)q_1 + (-3t^3 + 3t^2)q_2 + (t^3)q_3
\]
Bézier blending functions

This is how the polynomials for each coefficient looks like

The functions sum to 1 at any point along the curve.

Endpoints have full weight

The weights of each function is clear and the labels show the control
How to produce complex, long curves?

- We could only use 4 control points to design curves.
- What if we want to produce long curves with complex shapes.
- How do can we do that?
Drawing Complex Long Curves

- Using higher order curves
  - costly
  - Need many multiplications
- Pierce together low order curves
  - Need to make sure the connection points are smooth
Continuity between curve segments

- If the direction and magnitude of $\frac{d}{dt} [X(t)]$ are equal at the join point, the curve is called $C^n$ continuous.
- i.e. if two curve segments are simply connected, the curve is Co continuous.
- If the tangent vectors of two cubic curve segments are equal at the join point, the curve is $C_1$ continuous.
Continuity between curve segments

- If the directions (but not necessarily the magnitudes) of two segments’ tangent vectors are equal at the join point, the curve has $G^i$ continuity.
Continuity with Hermite and Bezier Curves

- How to achieve C0, C1, G1 continuity?
Joining Bezier Curves

- $G^1$ continuity is provided at the endpoint when $P_3 - P_4 = k (P_4 - P_5)$
- if $k=1$, $C^1$ continuity is obtained

http://www.inf.ed.ac.uk/teaching/courses/cg/d3/bezierJoin.html
Uniform Cubic B-Splines

• Another popular form of curves
• The curve does not necessarily pass through the control points
• Can produce a longer continuous curve without worrying about the boundaries
• Has $C^2$ continuity at the boundaries

http://www.rose-hulman.edu/~finn/CCLI/Applets/BSplineApplet.html
Uniform Cubic B-Splines (2)

• The matrix form and the basis functions

• The knots specify the range of the curve

\[ X(t) = t^T M Q^i \quad \text{for} \quad t_i \leq t \leq t_{i+1} \]

where \[ Q^i = (x_{i-3}, \ldots, x_i) \]

\[ M = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \]

\[ t^T = ((t-t_i)^3, (t-t_i)^2, t-t_i, 1) \]

\[ t_i : \text{knots, } 3 \leq i \]
Uniform Cubic B-Splines (3)

- This is how the basis look like over the domain
- The initial part is defined after passing the fourth knot
Another usage of uniform cubic B-splines

- Representing the joint angle trajectories of characters and robots
- Need more control points to represent a longer continuous movement
- Need C² continuity to make the acceleration smooth
- And not changing the torques suddenly
Catmull-Rom Spline

• A curve that interpolates control points
• The tangent vectors at the endpoints of a Hermite curve is set such that they are decided by the two surrounding control points
Catmull-Rom Spline

- $C_1$ continuity

$$P^i(t) = T \cdot M_{CR} \cdot G_B$$

$$= \frac{1}{2} T \cdot \begin{bmatrix}
-1 & 3 & -3 & 1 \\
2 & -5 & 4 & -1 \\
-1 & 0 & 1 & 0 \\
0 & 2 & 0 & 0
\end{bmatrix} \begin{bmatrix}
P_{i-3} \\
P_{i-2} \\
P_{i-1} \\
P_i
\end{bmatrix}$$
Today

• Parametric curves
  – Introduction
  – Hermite curves
  – Bezier curves
  – Uniform cubic B-splines
  – Catmull-Rom spline

• Bicubic patches

• Tessellation
  – Adaptive tessellation
Bicubic patches

• The concept of parametric curves can be extended to surfaces

• The cubic parametric curve is in the form of $Q(t) = t^T M \mathbf{q}$ where $\mathbf{q} = (q_1, q_2, q_3, q_4)$ : $q_i$ control points, $M$ is the basis matrix (Hermite or Bezier,...), $t^T = (t^3, t^2, t, 1)$
• Now we assume $q_i$ to vary along a parameter $s$,
• $Q_i(s,t) = t^T M \ [q_1(s), q_2(s), q_3(s), q_4(s)]$
• $q_i(s)$ are themselves cubic curves, we can write them in the form ...
Bicubic patches

\[ Q(s,t) = t^T \cdot M \cdot (s^T \cdot M \cdot [q_{11}, q_{12}, q_{13}, q_{14}], \ldots, s^T \cdot M \cdot [q_{41}, q_{42}, q_{43}, q_{44}]) \]

\[ = t^T \cdot M \cdot q \cdot M^T \cdot s \]

where \( q \) is a 4x4 matrix

Each column contains the control points of \( q_1(s), \ldots, q_4(s) \)

\( x, y, z \) computed by

\[ x(s,t) = t^T \cdot M \cdot q_x \cdot M^T \cdot s \]

\[ y(s,t) = t^T \cdot M \cdot q_y \cdot M^T \cdot s \]

\[ z(s,t) = t^T \cdot M \cdot q_z \cdot M^T \cdot s \]
Bézier example

• We compute \((x,y,z)\) by

\[
x(s,t) = t^T \cdot M_B \cdot q_x \cdot M_B^T \cdot s
\]

\(q_x\) is \(4 \times 4\) array of \(x\) coords

\[
y(s,t) = t^T \cdot M_B \cdot q_y \cdot M_B^T \cdot s
\]

\(q_y\) is \(4 \times 4\) array of \(y\) coords

\[
z(s,t) = t^T \cdot M_B \cdot q_z \cdot M_B^T \cdot s
\]

\(q_z\) is \(4 \times 4\) array of \(z\) coords

http://www.inf.ed.ac.uk/teaching/courses/cg/d3/bezierPatch.html
http://www.math.psu.edu/dlittle/java/parametricEquations/bezierSurfaces/index.html
Today

- Parametric curves
  - Introduction
  - Hermite curves
  - Bezier curves
  - Uniform cubic B-splines
  - Catmull-Rom spline
- Bicubic patches
- **Tessellation**
  - Adaptive tessellation
Displaying Bicubic patches.

- Directly rasterizing bicubic patches is not so easy
- Need to convert the bicubic patches into a polygon mesh
  - tessellation
- Need to compute the normals
  - vector cross product of the 2 tangent vectors.
Tessellation

• As computers are optimized for rendering triangles, the easiest way to display parametric surfaces is to convert them into triangle meshes.

• The simplest way is to do uniform tessellation, which samples points uniformly in the parameter space.
Uniform Tessellation

- Sampling points uniformly with the parameters
- What are the problems with uniform tessellation?
- Which area needs more tessellation?
- Which area does not need much tessellation?
Adaptive Tessellation

- Adaptive tessellation – adapt the size of triangles to the shape of the surface

On the other hand, for flat areas we do not need many triangles.
Adaptive Tessellation

- For every triangle edges, check if each edge is tessellated enough (curveTessEnough())
- If all edges are tessellated enough, check if the whole triangle is tessellated enough as a whole (triTessEnough())
- If one or more of the edges or the triangle’s interior is not tessellated enough, then further tessellation is needed
Adaptive Tessellation

• When an edge is not tessellated enough, a point is created halfway between the edge points’ uv-values
• New triangles are created and the tessellator is once again called with the new triangles as input

Four cases of further tessellation
Adaptive Tessellation

AdaptiveTessellate(p,q,r)

• tessPQ=not curveTessEnough(p,q)
• tessQR=not curveTessEnough(q,r)
• tessRP=not curveTessEnough(r,p)
• If (tessPQ and tessQR and tessRP)
  – AdaptiveTessellate(p,(p+q)/2,(p+r)/2);
  – AdaptiveTessellate(q,(q+r)/2,(q+p)/2);
  – AdaptiveTessellate(r,(r+p)/2,(r+q)/2);
  – AdaptiveTessellate((p+q)/2,(q+r)/2,(r+p)/2);
• else if (tessPQ and tessQR)
  – AdaptiveTessellate(p,(p+q)/2,r);
  – AdaptiveTessellate((p+q)/2,(q+r)/2,r);
  – AdaptiveTessellate((p+q)/2,q,(q+r)/2);
• else if (tessPQ)
  – AdaptiveTessellate(p,(p+q)/2,r);
  – AdaptiveTessellate(q,r,(p+q)/2);
• Else if (not triTessEnough(p,q,r))
  AdaptiveTessellate((p+q+r)/3,p,q);
  AdaptiveTessellate((p+q+r)/3,q,r);
  AdaptiveTessellate((p+q+r)/3,r,p);
end;
**curveTessEnough**

- Say you are to judge whether \( \mathbf{ab} \) needs tessellation
- You can compute the midpoint \( c \), and compute the curve’s distance \( l \) from \( d \), the midpoint of \( \mathbf{ab} \)
- Check if \( l/||\mathbf{a-b}|| \) is under a threshold
- Can do something similar for triTessEnough
  - Sample at the mass center and calculate its distance from the triangle
Normal Vectors

\[
\frac{\partial}{\partial s} Q(s, t) = \frac{\partial}{\partial s} (t^T M q M^T s) = t^T M q M^T \frac{\partial}{\partial s} (s)
= t^T M q_x M^T [3s^2, 2s, 1, 0]^T
\]

\[
\frac{\partial}{\partial t} Q(s, t) = \frac{\partial}{\partial t} (t^T M q M^T s) = \frac{\partial}{\partial t} (t^T) M q M^T s
= [3t^2, 2t, 1, 0]^T M q M^T s
\]

\[
\frac{\partial}{\partial s} Q(s, t) \times \frac{\partial}{\partial t} Q(s, t) = (y_s z_t - y_t z_s, z_s x_t - z_t x_s, x_s y_t - x_t y_s)
\]

Tangent vectors can be computed by computing the partial derivatives
Then computing the cross product of the two partial derivative vectors
On-the-fly tessellation

• In many cases, it is preferred to tessellate on-the-fly
• The size of the data can be kept small
• Tessellation is a highly parallel process
  – Can make use of the GPU
• The shape may deform in real-time
On-the-fly tessellation

- So, say in a dynamic environment, what are the factors that we need to take into account when doing the tessellation?
  - in addition to curvature?
Other factors?
Other factors?
Other factors?
Other factors to evaluate

- Inside the view frustum
- Front facing
- Occupying a large area in screen space
- Close to the silhouette of the object
- Illuminated by a significant amount of specular lighting
Summary

- Hermite, Bezier, B-Spline curves
- Bicubic patches
- Tessellation
  - Triangulation of parametric surfaces
  - On-the-fly tessellation
Reading for this lecture

• Shirley Chapter 15 (Curves)
• Foley et al. Chapter 11, section 11.2 up to and including 11.2.3
• Introductory text Chapter 9, section 9.2 up to and including section 9.2.4
• Foley et al., Chapter 11, sections 11.2.3, 11.2.4, 11.2.9, 11.2.10, 11.3 and 11.5.
• Introductory text, Chapter 9, sections 9.2.4, 9.2.5, 9.2.7, 9.2.8 and 9.3.
• Real-time Rendering 2nd Edition Chapter 12.1-3