Overview

- More on Bézier and B-splines
  - de Casteljau’s algorithm
  - General form of B-splines
  - de Boor’s algorithm
  - Knot insertion
- NURBS
- Subdivision surfaces
de Casteljau’s algorithm

- A method to evaluate (sample points in) or draw a Bézier curve
- Works with Bézier curves of any degree
- A precise method to evaluate the curve
de Casteljau’s algorithm

- Given the control points $P_1, \ldots, P_n$ and the parameter value $0 \leq t \leq 1$
- Repeat the following procedure:
  \[ P_i^r(t) = (1 - t)P_i^{r-1}(t) + tP_{i+1}^{r-1}(t) \]
  \[ P_i^0(t) = P_i \]
- Then $P_0^n(t)$ is the point with parameter value $t$ on the Bézier curve
Why does this work?

• In the quadratic Bézier curve case with 3 control points, \( P_0, P_1, P_2 \)

\[
\begin{align*}
P_0^1(t) &= (1 - t)P_0 + tP_1 \\
P_1^1(t) &= (1 - t)P_1 + tP_2 \\
P_0^2(t) &= (1 - t)P_0^1 + tP_1^1
\end{align*}
\]

• By inserting the first two equations into the third we obtain

\[
P_0^2(t) = (1 - t)^2 P_0 + 2t(1 - t)P_1 + t^2 P_2
\]

• Doing this for 4 control points will give the cubic formula we saw last week
Why do we need this?

- The explicit representation can result in some instability
- Control points are randomly changed by 0.001
- The curve from de Casteljau’s algorithm stays almost the same
- The curve from the polynomial basis form can deviate from the original curve if the degree is high
Connecting Bézier patches

- The same thing applies to patches
- The degree of the surface can easily become high since it is the multiplication of two curves, e.g. Bicubic is of degree 6
- The error of 16 control points is accumulated
Overview

- More on Bézier and B-splines
  - de Casteljau’s algorithm
- General form of B-splines
  - de Boor’s algorithm
  - Knot insertion
- NURBS
- Subdivision surfaces
a B-spline of order \( k \) (polynomial of degree \( k-1 \)) is a parametric curve composed of a linear combination of basis B-splines \( B_{i,k} \): 

\[ P_i(i = 0, \ldots, m) \] are the control points 

\[ p(t) = \sum_{i=0}^{m} P_i B_{i,k}(t) \]

Knots: \( t_0 \leq t_1 \leq \ldots \leq t_{k+m} \) - the knots subdivide the domain of the B-spline curve into a set of knot spans \([t_i, t_{i+1})\)

The B-splines can be defined by

\[ B_{i,1}(t) = \begin{cases} 1, & t_i \leq t < t_{i+1} \\ 0, & \text{otherwise} \end{cases} \]

\[ B_{i,k}(t) = \frac{t-t_i}{t_{i+k-1}-t_i} B_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k-1}-t_i} B_{i+1,k-1}(t) \]
B-spline basis
B-spline basis

\[ B_{i,3}(t) \]

- Quadratic

\[ B_{i,4}(t) \]

- Cubic
Producing curves using B-splines

- The basis functions are multiplied by the control points to define arbitrary curves
**Knots**

- The knots produce a vector that defines the domain of the curve.
- The knots must be in increasing order.
- Not necessarily uniform spacing.
- If uniformly sampled and degree is 3 we have a uniform cubic B-spline.
Knots

- Non-uniform knots:

![Non-uniform knots](http://i33www.ira.uka.de/applets/mocca/html/noplugin/BSplineBasis/AppBSplineBasis/index.html)
B-spline terms

- Order $k$: the number of control points affecting the sampled value
- Degree $k-1$: the degree of the basis function polynomial
- Control points $P_i \quad i = (0, \ldots, m)$
- Knots $t_j \quad (j = 0, \ldots, n)$
- An important rule: $n - m = k$
- The domain of the curve is $t_{k-1} \leq t \leq t_{m+1}$
- Below, $k=4$, $m=9$, domain is $t_3 \leq t \leq t_{10}$
Clamped B-splines

- The first and last knot values are repeated with multiplicity equal to the order (degree + 1)
- The end points pass the control point
- For cubic bsplines, the multiplicity of the first / last knots must be 4 (repeated four times)
Controlling the shape of B-splines

- Moving the control points is the most obvious way to control bspline curves

- Changing the position of control point $P_i$ only affects the interval $[t_i, t_{i+k})$, where $k$ is the order of a B-spline curve
  - Editing the shape through the knot vector is not very intuitive
Overview

- More on Bézier and B-splines
  - de Casteljau’s algorithm
  - General form of B-splines
  - de Boor’s algorithm
  - Knot insertion
- NURBS
- Subdivision surfaces
de Boor’s algorithm

- B-spline version of de Casteljau’s algorithm
- A precise method to evaluate the curve
- Starting from control points and parameter value $t$, recursively solve:

$$ P_i^r = (1 - a_{i,r}) P_{i-1}^{r-1} + a_{i,r} P_i^{r-1} $$

$$ a_{i,r} = \frac{t - t_i}{t_{i+k-1-r} - t_i} $$
Example

- Assume we have a cubic B-spline with knot vector:
  \[ [0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1] \]
- Computing the point at \( t = 0.4 \)
  
  Then \( t_4 \leq t \leq t_5 \) and the control points that affect the final position are \( P_4, P_3, P_2, P_1 \)
Example

Assume we have a cubic B-spline whose knot vector is \{0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1\}.

Let's compute a point at \( t = 0.4 \).

Then, \( t_4 < t < t_5 \), and the control points that affect the final position are \( P_4, P_3, P_2, P_1 \).

\[ a_{4,1} = (t - t_4)/(t_{4+3} - t_4) = 0.2 \]
\[ a_{3,1} = (t - t_3)/(t_{3+3} - t_3) = 0.53 \]
\[ a_{2,1} = (t - t_2)/(t_{2+3} - t_2) = 0.8 \]
\[ P_{4,1} = (1 - a_{4,1})P_{3,0} + a_{4,1}P_{4,0} \]
\[ P_{3,1} = (1 - a_{3,1})P_{2,0} + a_{3,1}P_{3,0} \]
\[ P_{2,1} = (1 - a_{4,1})P_{1,0} + a_{2,1}P_{2,0} \]
Example

Assume we have a cubic B-spline whose knot vector is \{0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1\}.

Let's compute a point at \( t = 0.4 \).

Then, \( t_4 < t < t_5 \), and the control points that affect the final position are \( P_4, P_3, P_2, P_1 \).

\[
a_{4,2} = \frac{(t - t_4)}{(t_4 + 3 - 1 - t_4)} = 0.3
\]
\[
a_{3,2} = \frac{(t - t_3)}{(t_3 + 3 - 1 - t_3)} = 0.8
\]

\[
P_{4,2} = (1 - a_{4,2}) P_{3,1} + a_{4,2} P_{4,1}
\]
\[
P_{3,2} = (1 - a_{3,2}) P_{2,1} + a_{3,2} P_{3,1}
\]
Example

Assume we have a cubic B-spline whose knot vector is \{0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1\}

Let's compute a point at \( t = 0.4 \)

Then, \( t_4 < t < t_5 \), and the control points that affect the final position are \( P_4, P_3, P_2, P_1 \)

\[
\alpha_{4,3} = (u - u_4)/(u_{4+3-2} - u_4) = 0.6
\]

\[
P_{4,3} = (1 - \alpha_{4,3})P_{3,2} + \alpha_{4,2}P_{4,2}
\]

\[
= 0.4P_{3,2} + 0.6P_{4,2}
\]
What if you want to edit some details?

- You might want to add some high resolution details in a particular area whilst leaving the rest of the curve unchanged.
Knot insertion

- We can do this by knot Insertion

- New knots can be added without changing the shape of the curve

- Because of the basic rule $n-m = k$ ($n+1$: number of knots, $m+1$: the number control points, $k$: order) the number of control points will also increase
Knot insertion

- For a curve of degree f we remove f-1 points and add f points
- i.e. for a cubic B-spline, remove 2 points and add 3 points
Knot insertion

- If the new knot $t$ is inserted into the span $[t_j, t_{j+1})$, the new control points can be computed by

$$Q_i = (1 - a_i)P_{i-1} + a_i P_i$$

where $Q_i$ is the new control point and $a_i$ is computed by

$$a_i = \frac{t - t_i}{t_{i+k-1} - t_i} \text{ for } j-k+2 \leq i \leq j$$

$P_{j-k+1}, P_{j-k+2}, ..., P_{j-1}, P_j$ is replaced with $P_{j-k+1}, Q_{j-k+2}, ..., Q_{j-1}, Q_j, P_j$. 

\[t_i \quad \quad \quad \quad t \quad \quad \quad \quad t_{i+k-1}\]

\[a_i \quad 1-a_i\]
Example

- A bspline curve of degree 3 ($k=4$) having the following knots
- $t=0.5$ inserted
Example

- A bspline curve of degree 3 (k=4) having the following knots
- \( t=0.5 \) inserted

<table>
<thead>
<tr>
<th></th>
<th>( t_0 ) to ( t_3 )</th>
<th>( t_4 )</th>
<th>( t_5 )</th>
<th>( t_6 )</th>
<th>( t_7 )</th>
<th>( t_8 ) to ( t_{11} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0.2</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>( t_0 ) to ( t_3 )</th>
<th>( t_4 )</th>
<th>( t_5 )</th>
<th>( t_6 )</th>
<th>( t_7 )</th>
<th>( t_8 )</th>
<th>( t_9 ) to ( t_{12} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0.2</td>
<td>0.4</td>
<td><strong>0.5</strong></td>
<td>0.6</td>
<td>0.8</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\alpha_3 = \frac{t - t_3}{t_6 - t_3} = \frac{0.5 - 0}{0.6 - 0} = \frac{5}{6}
\]

\[
\alpha_4 = \frac{t - t_4}{t_7 - t_4} = \frac{0.5 - 0.2}{0.8 - 0.2} = \frac{1}{2}
\]

\[
\alpha_5 = \frac{t - t_5}{t_8 - t_5} = \frac{0.5 - 0.4}{1 - 0.4} = \frac{1}{6}
\]
Example

• A bspline curve of degree 3 ($k=4$) having the following knots

<table>
<thead>
<tr>
<th>$t_0$ to $t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
<th>$t_6$</th>
<th>$t_7$</th>
<th>$t_8$ to $t_{11}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$t_0$ to $t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
<th>$t_6$</th>
<th>$t_7$</th>
<th>$t_8$</th>
<th>$t_9$ to $t_{12}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
<td>0.4</td>
<td>0.5</td>
<td>0.6</td>
<td>0.8</td>
<td>1</td>
</tr>
</tbody>
</table>

$a_5 = \frac{t - t_5}{t_8 - t_5} = \frac{0.5 - 0.4}{1 - 0.4} = \frac{1}{6}$

$a_4 = \frac{t - t_4}{t_7 - t_4} = \frac{0.5 - 0.2}{0.8 - 0.2} = \frac{1}{2}$

$a_3 = \frac{t - t_3}{t_6 - t_3} = \frac{0.5 - 0}{0.6 - 0} = \frac{5}{6}$

$Q_5 = \left(1 - \frac{1}{6}P_4\right) + \frac{1}{6}P_5$

$Q_4 = \left(1 - \frac{1}{2}P_3\right) + \frac{1}{2}P_4$

$Q_3 = \left(1 - \frac{5}{6}P_2\right) + \frac{5}{6}P_3$

Summary of B-splines

- Knot vector defines the domain
- Evaluation by de Boor’s algorithm
- Controlling the shape by the control points
- Clamping the points by increasing the multiplicity of the knots at the end points
- Increase the resolution by knot insertion
Overview

- More on Bézier and B-splines
  - de Casteljau’s algorithm
  - General form of B-splines
  - de Boor’s algorithm
  - Knot insertion

- NURBS

- Subdivision surfaces
NURBS (Non-uniform rational B-spline)

- Standard curves/surface representation in computer aided design

\[ C(t) = \frac{\sum_{i=0}^{n} B_{i,k}(t)w_i P_i}{\sum_{i=0}^{n} B_{i,k}(t)w_i} \]

- \( P_i \): control points
- \( B_{i,k} \): Bspline basis of order \( k \)
- \( w_i \): weights
Benefits of using NURBS

• More degrees of freedom to control the curve (can control the weights)

• Invariant under perspective transformation
  – Can project the control points onto the screen and interpolate on the screen
  – Don’t need to apply the perspective transformation to all the points on the curve

• Can model conic sections such as circles, ellipses and hyperbolas
Example of changing weights

- Increasing the weight will bring the curve closer to the corresponding control point
B-spline Surfaces

- Given the following information:
- a set of $m+1$ rows and $n+1$ control points $P_{i,j}$ where $0 \leq i \leq m$ and $0 \leq j \leq n$
- Corresponding knot vectors in the $u$ and $v$ direction,

\[
p(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_{i,p}(u)B_{j,q}(v)P_{i,j} : \text{non-rational B-spline}
\]

\[
p(u,v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j}B_{i,p}(u)B_{j,q}(v)P_{i,j}}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j}B_{i,p}(u)B_{j,q}(v)} : \text{NURBS}
\]
Clamped, Closed and Open B-spline Surfaces

• Since a B-spline curve can be clamped, closed or open, a B-spline surface can also have three types in each direction.

• That is, we could ask to have a B-spline surface clamped in the $u$-direction and closed in the $v$-direction.

• If a B-spline is clamped in both directions, then this surface passes though control points $p_{0,0}$, $p_{m,0}$, $p_{0,n}$ and $p_{m,n}$

• If a B-spline surface is closed in one direction, then the surface becomes a tube.

• Closed in two direction: torus
  - Problems handling objects of arbitrary topology, such as a ball, double torus
Overview

• More on Bézier and B-splines
  • de Casteljau’s algorithm
  • General form of B-splines
  • de Boor’s algorithm
  • Knot insertion

• NURBS

• Subdivision surfaces
Subdivision Surfaces

- A method to model smooth surfaces
3D subdivision surface

- Start with a rough shape first and subdivide it recursively
- Stop when the shape is smooth enough
- Used for modelling smooth surfaces
Motivation

- Shape modeling
  - Topological restrictions of NURBS surfaces
    - Plane, Cylinder, and Torus
    - It is difficult to maintain smoothness at seams of patchwork.
      - Example: hiding seams in Woody (*Toy Story*), [DeRose98]
  - NURBS also require the control nets consist of a regular rectangular grid of control points

- LOD in a scene
  - A coarse shape when far away, a smooth dense surface when closer to the camera
Subdivision surface

- Can handle arbitrary topology

Different Schemes
- Doo-Sabin ‘78
- Catmull-Clark ‘78
- Etc (Loop, Butterfly, and many others)
A Primer: Chaiken’s Algorithm

Apply Iterated Function System

\[ Q_{2i} = \frac{3}{4} P_i + \frac{1}{4} P_{i+1} \]

\[ Q_{2i+1} = \frac{1}{4} P_i + \frac{3}{4} P_{i+1} \]

Limit Curve Surface

\[ Q_0 = \frac{3}{4} P_0 + \frac{1}{4} P_1 \]

\[ Q_1 = \frac{1}{4} P_0 + \frac{3}{4} P_1 \]

\[ Q_2 = \frac{3}{4} P_1 + \frac{1}{4} P_2 \]

\[ Q_3 = \frac{1}{4} P_1 + \frac{3}{4} P_2 \]

\[ Q_4 = \frac{3}{4} P_2 + \frac{1}{4} P_3 \]

\[ Q_5 = \frac{1}{4} P_2 + \frac{3}{4} P_3 \]

http://www.multires.caltech.edu/teaching/demos/java/chaikin.htm
Doo-Sabin Subdivision

• An *edge point* is formed from the midpoint of each edge.
• A *face point* is formed as the centroid of each polygon of the mesh.
• Finally, each vertex in the new mesh is formed as the average of
  - a vertex in the old mesh,
  - a face point for a polygon that touches that old vertex, and
  - the edge points for the two edges that belong to that polygon and touch that old vertex.
Doo-Sabin Subdivision

The new mesh, therefore, will
- create quadrilaterals for each edge in the old mesh,
- create a smaller n-sided polygon for each n-sided polygon in the old mesh, and
- create an n-sided polygon for each n-valence vertex (Valence being the number of edges that touch the vertex).
Catmull-Clark Subdivision

• A face with $n$ edges are subdivided into $n$ quadrilaterals

• Quads are better than triangles at capturing the symmetries of natural and man-made objects. Tube like surfaces (arms, legs, fingers) are easier to model.
Catmull-Clark Subdivision

\[ f = \frac{1}{n} \sum_{i=1}^{n} v_i \]

\[ e = \frac{v_1 + v_2 + f_1 + f_2}{4} \]

\[ v_{i+1} = \frac{n-2}{n} v_i + \frac{1}{n^2} \sum_j e_j + \frac{1}{n^2} \sum_j f_j \]
Modelling with Catmull-Clark

- Subdivision produces smooth continuous surfaces.
- How can “sharpness” and creases be controlled in a modeling environment?

**ANSWER:** Define new subdivision rules for “creased” edges and vertices.

1. Tag Edges sharp edges.
2. If an edge is sharp, apply new sharp subdivision rules.
3. Otherwise subdivide with normal rules.
Sharp Edges...

- Tag Edges as "sharp" or "not-sharp"
  - $n = 0$ – "not sharp"
  - $n > 0$ – sharp

During Subdivision,
- if an edge is "sharp", use sharp subdivision rules. Newly created edges, are assigned a sharpness of $n-1$.
- If an edge is "not-sharp", use normal smooth subdivision rules.

IDEA: Edges with a sharpness of “$n$” do not get subdivided smoothly for “$n$” iterations of the algorithm.

- In the picture on the right, the control mesh is a unit cube
- Different sharpness applied
Sharp Rules

- FACE (unchanged)

- EDGE

\[ f = \frac{1}{n} \sum_{i}^{n} v_i \]

- VERTEX

# adj. Sharp edges

corner >2
crease 2

\[ v_{i+1} = v_i \]

\[ v_{i+1} = \frac{e_1 + 6v_i + e_2}{8} \]
Another example of creases
Non-Integer Sharpness

- Density of newly generated mesh increases rapidly.
- In practice, 2 or 3 iterations of subdivision is sufficient.
- Need better “control”.

IDEA: Interpolate between smooth and sharp rules for non-integer sharpness values of $n$. 
Subdivision Surfaces in character animation [DeRose98]

- Used for first time in Geri’s game to overcome topological restriction of NURBS
- Modelled Geri’s head, hands, jacket, trousers, shirt, tie, and shoes
- Developed cloth simulation methods
Demo movie [Geri’s Game]

- Academy Award winning movie by Pixar

Demo of Catmull-Clark subdivision surface

- http://www.youtube.com/watch?v=lU8f0hnorU8&feature=related
Adaptive Subdivision

• Not all regions of a model need to be subdivided.

• Idea: Use some criteria and adaptively subdivide mesh where needed.
  – Curvature
  – Screen size ( make triangles < size of pixel )
  – View dependence
    • Distance from viewer
    • Silhouettes
    • In view frustum
  – Careful! Must ensure that “cracks” aren’t made

View-dependent refinement of progressive meshes
Hugues Hoppe.
(SIGGRAPH ’97)
Subdivision Surface Summary

• Advantages
  – Simple method for describing complex surfaces
  – Relatively easy to implement
  – Arbitrary topology
  – Local support
  – Guaranteed continuity
  – Multi-resolution

• Difficulties
  – Intuitive specification
  – Parameterization
  – Intersections
• A very good website for parametric curves / surfaces http://www.cs.mtu.edu/~shene/COURSES/cs3621/


Links

- http://www.ibiblio.org/e-notes/Splines/basis.html
- http://www.multires.caltech.edu/teaching/demos/java/chaikin.htm
- http://www.rose-hulman.edu/~finn/CCLI/Applets/DooSabinApplet.html