CS 430
Computer Graphics

NURBS Drawing

Week 5, Lecture 10

David Breen, William Regli and Maxim Peysakhov
Department of Computer Science
Drexel University

1

1

Outline

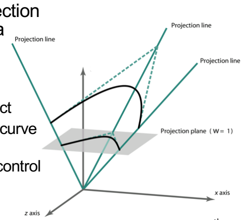
- Conic Sections via NURBS
- Knot insertion algorithm
- The de Boor's algorithm
  - for B-Splines
  - for NURBS
- Oslo Algorithm
- Barycentric Coordinates
- Discussion of homework #3

2

2

Conic Sections via NURBS

- Obtained via projection of the 3D parabola onto a plane
- Note:
  - 3D Case: rational curve is a 4D object
  - 2D Case: rational curve is a 3D object
  - assign *w* to each control point

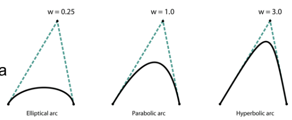


From http://devworld.apple.com/dev/techsupport/develop/issue25/schneider.html

3

Conic Sections via NURBS

- Define the curve with three control points
- Weights of first/last control points are 1
- For center control point
  - w<1 gives an ellipse
  - w>1 gives a hyperbola
  - w=1 gives a parabola
  - Knot vector is {0.0, 0.0, 0.0, 1.0, 1.0, 1.0}



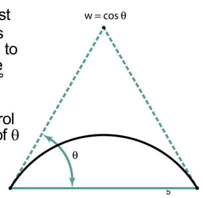
4

From http://devworld.apple.com/dev/techsupport/develop/issue25/schneider.html

4

Conic Sections via NURBS: A Circular Arc

- The two sides of the control polygon are of equal length
- The chord connecting the first and last control points meets each leg at an angle θ equal to half the angular extent of the desired arc (for instance, 30° for a 60° arc)
- The weight of the inner control point is equal to the cosine of θ
- Knot vector is {0.0, 0.0, 0.0, 1.0, 1.0, 1.0}



5

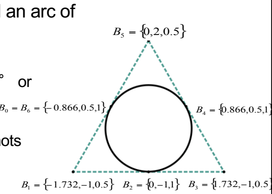
From http://devworld.apple.com/dev/techsupport/develop/issue25/schneider.html

5

Conic Sections via NURBS: A Circle

- What if we need an arc of >180° ?
- Idea:
  - Use multiple 90° or 120° arcs
  - stitch them together with knots

Example:
3 arcs of 120°



knots = { 0,0,0,1,1,2,2,3,3,3 }

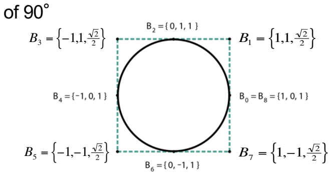
6

From http://devworld.apple.com/dev/techsupport/develop/issue25/schneider.html

6

Conic Sections via NURBS

Example:
4 arcs of 90°



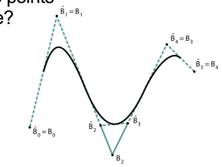
knots $= \{0, 0, 0, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{3}{4}, \frac{3}{4}, 1, 1, 1\}$

From http://devworld.apple.com/dev/techsupport/develop/issue25/schneider.html

7

Knot Insertion

- Issue: More control points mean more control
- How do we add more points and keep same curve?

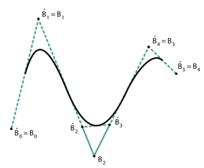


From http://devworld.apple.com/dev/techsupport/develop/issue25/schneider.html

8

Knot Insertion

- Basic Approach
  - Decide where we'd like to tweak the curve
  - Add a new knot
  - Find affected *d-1* control points
  - Replace it with *d* new control points



Example:
New knot at u=2.6

From http://devworld.apple.com/dev/techsupport/develop/issue25/schneider.html

9

Knot Insertion

- Given: *n+1* control points $(P_0, P_1, \ldots, P_n)$, a knot vector of *m+1* knots $U = \{u_0, u_1, \ldots, u_m\}$ and a degree *d* B-spline curve *C(u)*.
- Insert a new knot *t* into the knot vector without changing the shape of the curve.
- If *t* lies in knot span $[u_k, u_{k+1})$, only the basis functions for $(P_k, \ldots P_{k-d})$ are non-zero.
- Find *d* new control points Q_k on edge $P_{k-1}P_k$, Q_{k-1} on edge $P_{k-2}P_{k-1}$, ..., and Q_{k-d+1} on edge $P_{k-d}P_{k-d+1}$
- All other control points are unchanged.
- Note that *d-1* control points of the original control polyline are removed and replaced with *d* new control points.

10

See http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/single-insertion.html

10

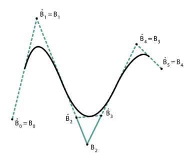
Knot Insertion Algorithm

- Create new control point

$$Q_j = (1-\alpha_j)P_{j-1} + \alpha_j P_j$$

- Where α is defined as

$$\alpha_j = \frac{t - u_j}{u_{j+d} - u_j}$$



See http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/single-insertion.html

11

11

Properties of Knot Insertion

- Increasing the multiplicity of a knot decreases the number of non-zero basis functions at this knot
- At a knot of multiplicity *d*, there will be only one non-zero basis function
- Corresponding point on the curve **p**(*u*) is affected by *exactly one* control point $\mathbf{p}_i$
- In fact **p**(*u*) is $\mathbf{p}_i$*!*

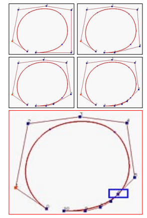
12

Compiled from Lecture notes of **Dr. Ching-Kuang Shene** @ Michigan Technological University

12

The de Boor Algorithm

- Generalization of de Casteljau's algorithm
- It provides a fast and numerically stable way for finding a point on a B-spline curve
- Observation: if a knot *u* is inserted *d* times to a B-spline, then p(u) is the point on the curve.
- Idea: We just simply insert *u* d times and the *last* point is **p**(*u*)!



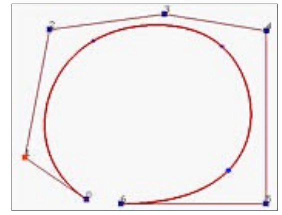
See http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/de-Boor.html

13

Compiled from Lecture notes of **Dr. Ching-Kuang Shene** @ Michigan Technological University

13

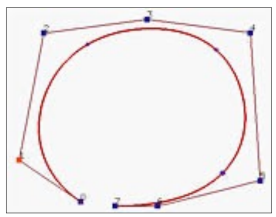
The de Boor Algorithm



14

14

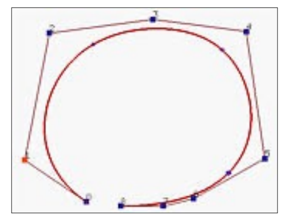
The de Boor Algorithm



15

15

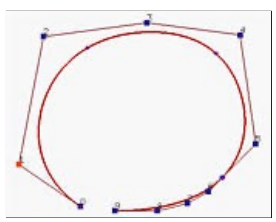
The de Boor Algorithm



16

16

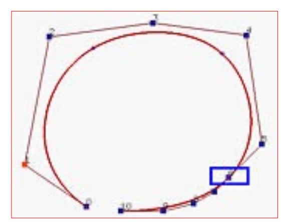
The de Boor Algorithm



17

17

The de Boor Algorithm



18

18

De Boor's Algorithm

If u lies in $[u_k, u_{k+1})$ and $u \neq u_k$, let $h = d$
If $u = u_k$ and u_k is a knot of multiplicity s, let $h = d - s$
Copy the affected control points $\mathbf{p}_{k-s}, \mathbf{p}_{k-s-1}, \ldots, \mathbf{p}_{k-d+1}, \mathbf{p}_{k-d}$
to a new array and rename them as $\mathbf{p}_{k-s,0}, \mathbf{p}_{k-s-1,0}, \ldots, \mathbf{p}_{k-d+1,0}$

```
for r := 1 to h do
    for i := k-d+r to k-s do
    {
        Let a_{i,r} = (u - u_i) / ( u_{i+d-r+1} - u_i )
        Let p_{i,r} = (1 - a_{i,r}) p_{i-1,r-1} + a_{i,r} p_{i,r-1}
    }
```

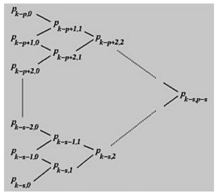
$\mathbf{p}_{k-s,d-s}$ is the point $\mathbf{p}(u)$.

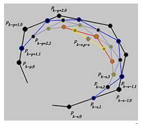
19

19

De Boor's Algorithm (*cont*)

```
for u := 0 to u_max do
{
  …
  for r := 1 to h do
    for i := k-p+r to k-s do
    {
      Let a_{i,r} = (u - u_i) / ( u_{i+p-r+1} - u_i )
      Let p_{i,r} = (1 - a_{i,r}) p_{i-1,r-1} + a_{i,r} p_{i,r-1}
    }
  p_{k-s,p-s} is the point p(u).
}
```





21

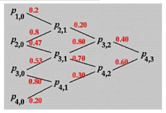
21

Example of de Boor's Algorithm

Degree 3 B-spline curve (*i.e.*, $d = 3$)
Defined by seven control points $\mathbf{p}_0, \ldots, \mathbf{p}_6$
And knot vector:
$u = 0.4$

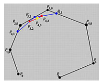
$u_0 = u_1 = u_2 = u_3$	u_4	u_5	u_6	$u_7 = u_8 = u_9 = u_{10}$
0	0.25	0.5	0.75	1

$a_{4,1} = (u - u_4) / (u_{4+3} - u_4) = 0.2$
$a_{3,1} = (u - u_3) / (u_{3+3} - u_3) = 8/15 = 0.53$
$a_{2,1} = (u - u_2) / (u_{2+3} - u_2) = 0.8$
$\mathbf{p}_{4,1} = (1 - a_{4,1})\mathbf{p}_{3,0} + a_{4,1}\mathbf{p}_{4,0} = 0.8\mathbf{p}_{3,0} + 0.2\mathbf{p}_{4,0}$
$\mathbf{p}_{3,1} = (1 - a_{3,1})\mathbf{p}_{2,0} + a_{3,1}\mathbf{p}_{3,0} = 0.47\mathbf{p}_{2,0} + 0.53\mathbf{p}_{3,0}$
$\mathbf{p}_{2,1} = (1 - a_{2,1})\mathbf{p}_{1,0} + a_{2,1}\mathbf{p}_{2,0} = 0.2\mathbf{p}_{1,0} + 0.8\mathbf{p}_{2,0}$



$a_{4,2} = (u - u_4) / (u_{4+3-1} - u_4) = 0.3$
$a_{3,2} = (u - u_3) / (u_{3+3-1} - u_3) = 0.8$
$\mathbf{p}_{4,2} = (1 - a_{4,2})\,\mathbf{p}_{3,1} + a_{4,2}\mathbf{p}_{4,1} = 0.7\mathbf{p}_{3,1} + 0.3\mathbf{p}_{4,1}$
$\mathbf{p}_{3,2} = (1 - a_{3,2})\,\mathbf{p}_{2,1} + a_{3,2}\mathbf{p}_{3,1} = 0.2\mathbf{p}_{2,1} + 0.8\mathbf{p}_{3,1}$

$a_{4,3} = (u - u_4) / (u_{4+3-2} - u_4) = 0.6$
$\mathbf{p}_{4,3} = (1 - a_{4,3})\mathbf{p}_{3,2} + a_{4,3}\mathbf{p}_{4,2} = 0.4\mathbf{p}_{3,2} + 0.6\mathbf{p}_{4,2}$



22

22

Similar but Different

De Casteljau's:

- Dividing points are computed with a pair of numbers *(1 - u)* and *u* that never change
- Can be used for curve subdivision
- Uses *all* control points

De Boor's

- These pairs of numbers are different and depend on the column number and control point number
- Intermediate control points are not sufficient
- *d-1* affected control points are involved in the computation

23

23

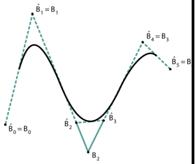
De Boor's: Curves

25

25

Oslo Algorithm

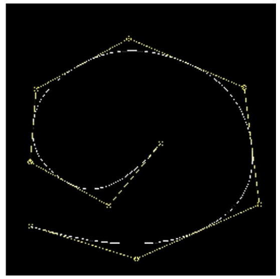
- A subdivision algorithm for B-splines, the basic idea:
- Take the curve with $m+1$ control points P_0 to P_m
- Insert a knot in any point (0.5 maybe?)
- As a result you will have 2 new points P_k' and P_k''
- Take curves with $m+1$ control points $P_0 \ldots P_k', P_k'' \ldots P_{m-1}$ and $P_1 \ldots P_k', P_k'' \ldots P_m$
- Apply procedure recursively on each part



26

26

Oslo Algorithm

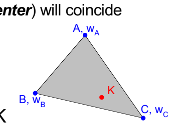


27

27

Barycentric Coordinates

- By Ceva's Theorem:
  - For any point K inside the triangle ABC
  - Consider the existence of *masses* w_A, w_B, and w_C, placed at the vertices of the triangle
  - Their center of gravity (***barycenter***) will coincide with the point K.
- August Ferdinand Moebius (1790-1868) defined (1827) w_A, w_B, and w_C as the ***barycentric*** coordinates of K
- $K = w_A A + w_B B + w_C C$



28

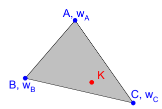
28

Properties of Barycentric Coordinates

- Not unique
- Can be generalized to negative masses
- Can be made unique by setting

  $$w_A + w_B + w_C = 1$$

  $$w_C = 1 - w_A - w_B$$
- $w_A = 0$ for points on BC
  $w_B = 0$ for points on AC
  $w_C = 0$ on AB



29

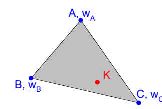
29

Properties of Barycentric Coordinates

- Vertices' Barycentric Coordinates
  - A: (1,0,0)
  - B: (0,1,0)
  - C: (0,0,1)
- Points inside a triangle

  $$0 <= w_A,\ w_B,\ w_C =< 1$$

- $K = w_A A + w_B B + w_C C$



30

30

Calculating the Weights

- Given vertices A, B, C and Centroid K
- What are the weights, w_A, w_B, w_C?

  $$x_K = w_A x_A + w_B x_B + w_C x_C$$

  $$y_K = w_A y_A + w_B y_B + w_C y_C$$
- Substitute $w_C = 1 - w_A - w_B$

  $$x_K = w_A x_A + w_B x_B + (1 - w_A - w_B) x_C$$

  $$y_K = w_A y_A + w_B y_B + (1 - w_A - w_B) y_C$$

33

33

Calculating Weights (cont.)

- Solve for w_A and w_B

$$w_A = \frac{(x_B - x_C)(y_C - y_K) - (x_C - x_K)(y_B - y_C)}{(x_A - x_C)(y_B - y_C) - (x_B - x_C)(y_A - y_C)}$$

$$w_B = \frac{(x_A - x_C)(y_C - y_K) - (x_C - x_K)(y_A - y_C)}{(x_B - x_C)(y_A - y_C) - (x_A - x_C)(y_B - y_C)}$$

- $w_C = 1 - w_A - w_B$

34

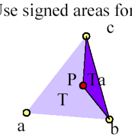
34

Given P, how can we compute weights?

- Compute the areas of the opposite subtriangle
  - Ratio with complete area

$w_A = A_a/A$, $w_B = A_b/A$ $w_C = A_c/A$

Use signed areas for points outside the triangle



Area Ta:
|(b-P) x (c-P)|/2

Onto…

- Bézier Surfaces
- B-spline Surfaces
- NURBS Surfaces
- Faceting, Subdivision, Tessellation
- 3D Objects

Programming assignment 3

- Input PostScript-like file containing polygons
- Output B/W XPM
- Implement viewports
- Use Sutherland-Hodgman intersection for polygon clipping
- Implement scanline polygon filling. (*You can not use flood filling algorithms*)