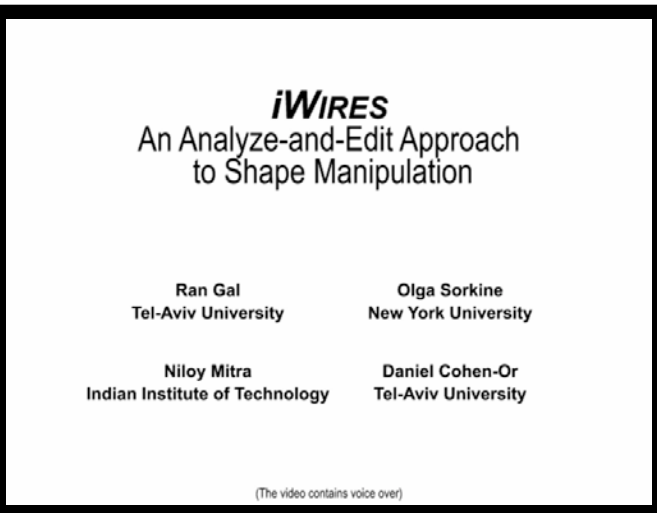


CS-184: Computer Graphics

Lecture12: Curves and Surfaces

Maneesh Agrawala
University of California, Berkeley

Slides based on those of James O'Brien

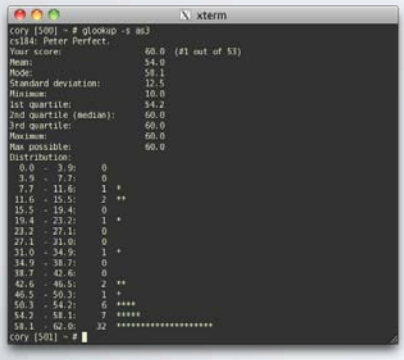
Announcements

Assignment 5: due Fri Nov 5 by 11pm

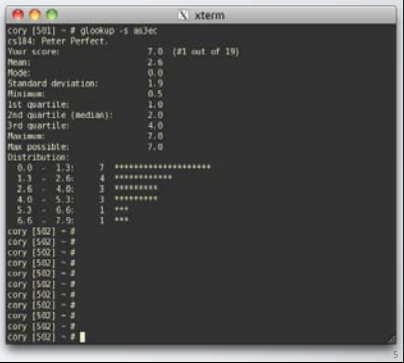
Announcements

Assignment 3 graded

- glookup -s as3, glookup -s as3ec



```
cory [500] ~ # glookup -s as3
cs184: Peter Perfect.
Your score:                 60.0  (#1 out of 53)
Mean:                       54.0
Mode:                       58.1
Standard deviation:         12.5
Minimum:                    10.0
1st quartile:               54.2
2nd quartile (median):      60.0
3rd quartile:               60.0
Maximum:                    60.0
Max possible:               60.0
Distribution:
  0.0  -  3.9:     0
  3.9  -  7.7:     0
  7.7  - 11.6:     1  *
 11.6  - 15.5:     2  **
 15.5  - 19.4:     0
 19.4  - 23.2:     1  *
 23.2  - 27.1:     0
 27.1  - 31.0:     0
 31.0  - 34.9:     1  *
 34.9  - 38.7:     0
 38.7  - 42.6:     0
 42.6  - 46.5:     2  **
 46.5  - 50.3:     1  *
 50.3  - 54.2:     6  ****
 54.2  - 58.1:     7  *****
 58.1  - 62.0:    32  ********************
cory [501] ~ #
```

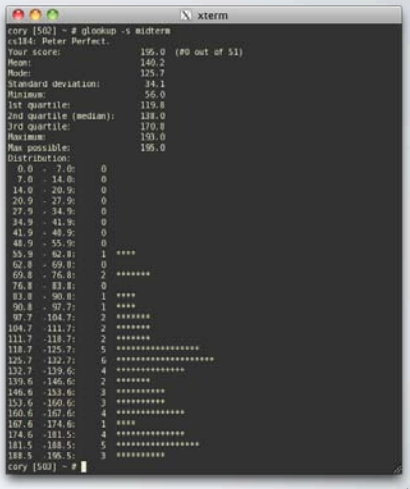


```
cory [501] ~ # glookup -s as3ec
cs184: Peter Perfect.
Your score:                  7.0  (#1 out of 19)
Mean:                        2.6
Mode:                        0.0
Standard deviation:          1.9
Minimum:                     0.5
1st quartile:                1.0
2nd quartile (median):       2.0
3rd quartile:                4.0
Maximum:                     7.0
Max possible:                7.0
Distribution:
  0.0  -  1.3:     7  ********************
  1.3  -  2.6:     4  ************
  2.6  -  4.0:     3  *********
  4.0  -  5.3:     3  *********
  5.3  -  6.6:     1  ***
  6.6  -  7.9:     1  ***
cory [502] ~ #
cory [502] ~ #
cory [502] ~ #
cory [502] ~ #
cory [502] ~ #
cory [502] ~ #
cory [502] ~ #
cory [502] ~ #
cory [502] ~ #
cory [502] ~ #
cory [502] ~ #
```


Announcements

Midterm graded

- glookup -s midterm, glookup -s midtermec



```
cory [502] ~ # glookup -s midterm
cs184: Peter Perfect.
Your score:                195.0  (#0 out of 51)
Mean:                      140.2
Mode:                      125.7
Standard deviation:         34.1
Minimum:                    56.0
1st quartile:              119.8
2nd quartile (median):     138.0
3rd quartile:              170.8
Maximum:                   193.0
Max possible:              195.0
Distribution:
  0.0  -  7.0:     0
  7.0  - 14.0:     0
 14.0  - 20.9:     0
 20.9  - 27.9:     0
 27.9  - 34.9:     0
 34.9  - 41.9:     0
 41.9  - 48.9:     0
 48.9  - 55.9:     0
 55.9  - 62.8:     1  ****
 62.8  - 69.8:     0
 69.8  - 76.8:     2  *******
 76.8  - 83.8:     0
 83.8  - 90.8:     1  ****
 90.8  - 97.7:     1  ****
 97.7  - 104.7:    2  *******
104.7  - 111.7:    2  *******
111.7  - 118.7:    2  *******
118.7  - 125.7:    5  *****************
125.7  - 132.7:    6  ********************
132.7  - 139.6:    4  **************
139.6  - 146.6:    2  *******
146.6  - 153.6:    3  **********
153.6  - 160.6:    3  **********
160.6  - 167.6:    4  **************
167.6  - 174.6:    1  ****
174.6  - 181.5:    4  **************
181.5  - 188.5:    5  *****************
188.5  - 195.5:    3  **********
cory [502] ~ #
```

Today

General curve and surface representations

Splines and other polynomial bases

Geometry Representations

Constructive Solid Geometry (CSG)

Parametric

- Polygons
- Subdivision surfaces

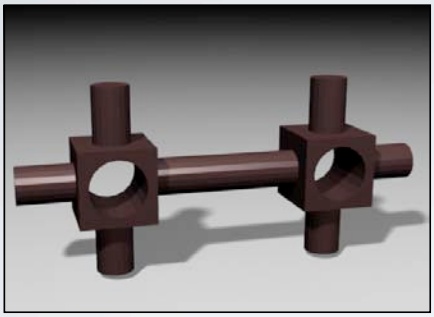
Implicit Surfaces

Point-based Surface

Not always clear distinctions

- *i.e.* CSG done with implicits

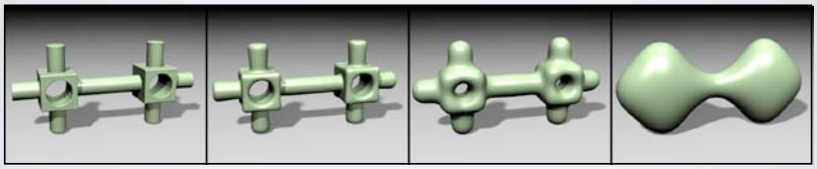
Geometry Representations



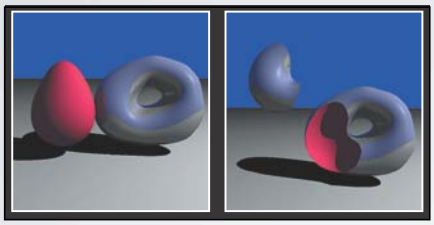
Object made by CSG
Converted to polygons

Geometry Representations

Object made by CSG
Converted to polygons
Converted to implicit surface

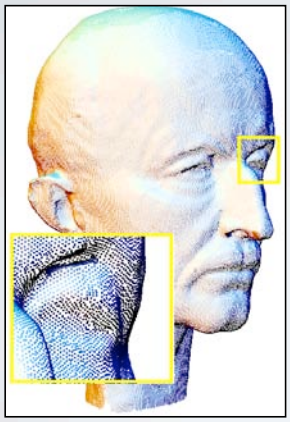


Geometry Representations



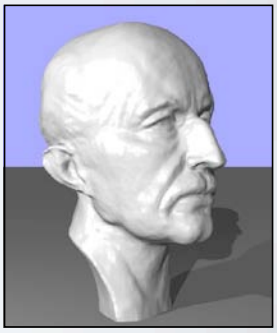
CSG on implicit surfaces

Geometry Representations



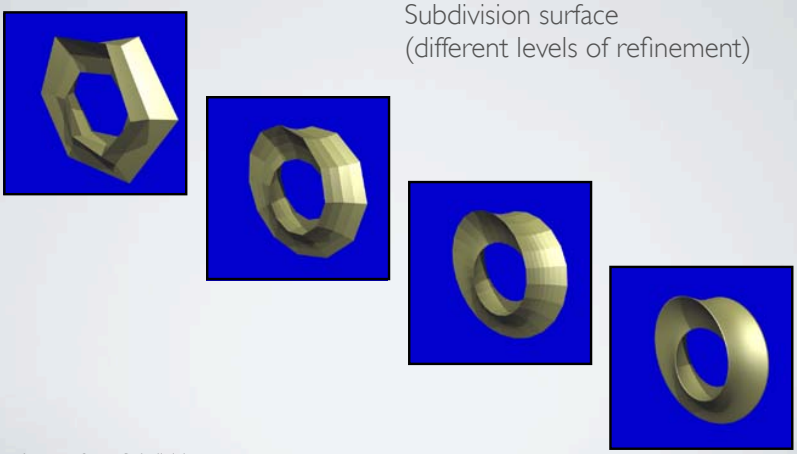
Ohtake, ***et al.***, SIGGRAPH 2003

Point-based surface descriptions



Geometry Representations

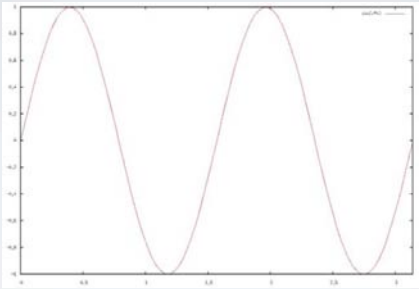
Subdivision surface
(different levels of refinement)



Images from Subdivision.org

Curve Drawing

Drawing Curves



$$y = f(x)$$

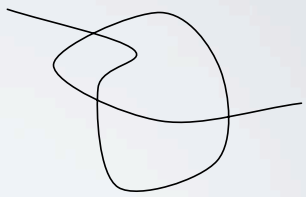
Only one value of y for each value of x...

Drawing Curves

Parametric curves

- Both x and y are a function of some third parameter

$$x = f(u)$$
$$y = f(u)$$



$$\mathbf{x} = \mathbf{f}(u)$$

$$u \in [u_0 \ldots u_1]$$

Drawing Curves

$$\mathbf{x} = \mathbf{f}(u) \qquad u \in [u_0 \ldots u_1]$$

Drawing Curves

Draw curves by drawing line segments

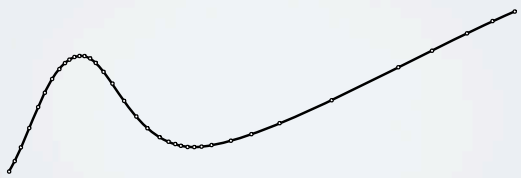
- Must take care in computing end points for lines
- How long should each line segment be?

$$\mathbf{x} = \mathbf{f}(u) \qquad u \in [u_0 \ldots u_1]$$

Drawing Curves

Draw curves by drawing line segments

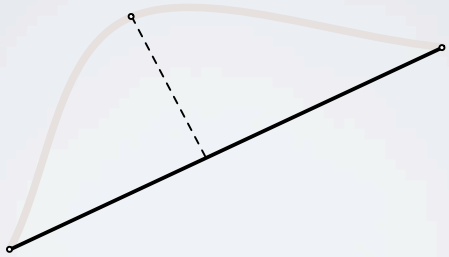
- Must take care in computing end points for lines
- How long should each line segment be?
- Variable spaced points



$$\mathbf{x} = \mathbf{f}(u) \qquad u \in [u_0 \ldots u_1]$$

Drawing Curves

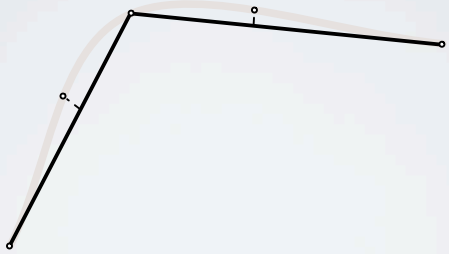
Midpoint-test subdivision



$$|\mathbf{f}(u_{mid}) - \mathbf{l}(0.5)|$$

Drawing Curves

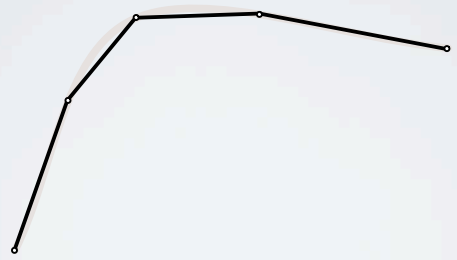
Midpoint-test subdivision



$$|\mathbf{f}(u_{mid}) - \mathbf{l}(0.5)|$$

Drawing Curves

Midpoint-test subdivision

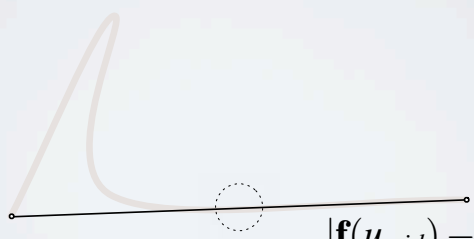


$$|\mathbf{f}(u_{mid}) - \mathbf{l}(0.5)|$$

Drawing Curves

Midpoint-test subdivision

- Not perfect
- We need more information for a guarantee...



$$|\mathbf{f}(u_{mid}) - \mathbf{l}(0.5)|$$

Curve Representations

Geometry Representations

Various strengths and weaknesses

- Ease of use for design
- Ease/speed for rendering
- Simplicity
- Smoothness
- Collision detection
- Flexibility (in more than one sense)
- Suitability for simulation
- *many others...*

Parametric Representations

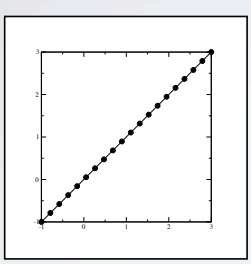
Curves:	$\boldsymbol{x} = \boldsymbol{x}(u)$	$\boldsymbol{x} \in \Re^n$	$u \in \Re$
Surfaces:	$\boldsymbol{x} = \boldsymbol{x}(u, v)$	$\boldsymbol{x} \in \Re^n$	$u, v \in \Re$
	$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{u})$		$\boldsymbol{u} \in \Re^2$
Volumes:	$\boldsymbol{x} = \boldsymbol{x}(u, v, w)$	$\boldsymbol{x} \in \Re^n$	$u, v, w \in \Re$
	$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{u})$		$\boldsymbol{u} \in \Re^3$

and so on...

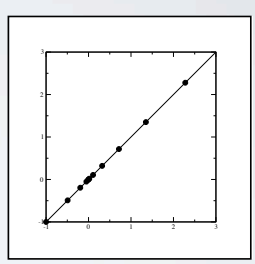
Note: a vector function is really $\boldsymbol{n}$ scalar functions

Parametric Rep. Non-unique

Same curve/surface may have multiple formulae



$\boldsymbol{x}(u) = [u, u]$



$\boldsymbol{x}(u) = [u^3, u^3]$

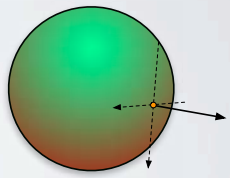
Simple Differential Geometry

Tangent to curve

$$\boldsymbol{t}(u) = \left.\frac{\partial \boldsymbol{x}}{\partial u}\right|_u$$

Tangents to surface

$$\boldsymbol{t}_u(u,v) = \left.\frac{\partial \boldsymbol{x}}{\partial u}\right|_{u,v} \qquad \boldsymbol{t}_v(u,v) = \left.\frac{\partial \boldsymbol{x}}{\partial v}\right|_{u,v}$$



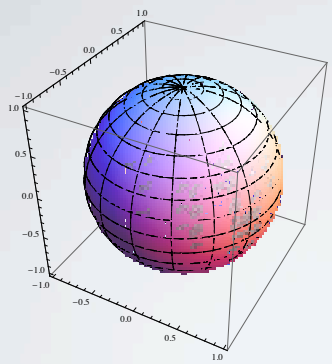
Normal of surface

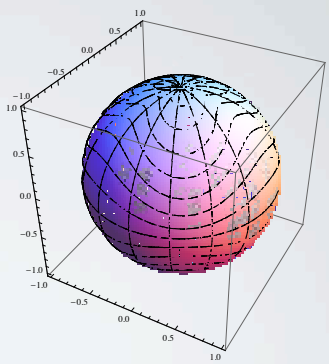
$$\hat{\boldsymbol{n}} = \frac{\boldsymbol{t}_u \times \boldsymbol{t}_v}{||\boldsymbol{t}_u \times \boldsymbol{t}_v||}$$

Also: curvature, curve normals, curve bi-normal, ***others...***

Degeneracies: $\partial \boldsymbol{x}/\partial u = 0$ or $\boldsymbol{t}_u \times \boldsymbol{t}_v = 0$

Non Orthogonal Tangents



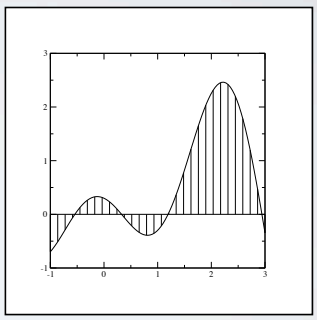


$$\begin{bmatrix} \cos(\theta 2\pi)\cos(\phi pi/2) \\ \sin(\theta 2\pi)\cos(\phi\pi/2) \\ \sin(\phi\pi/2) \end{bmatrix} \qquad \begin{bmatrix} \cos(2\pi\theta)\cos\left(\frac{1}{2}\pi\left(\frac{1}{2}(1-|\phi|)\cos(6\pi\theta)\phi+\phi\right)\right) \\ \cos\left(\frac{1}{2}\pi\left(\frac{1}{2}(1-|\phi|)\cos(6\pi\theta)\phi+\phi\right)\right)\sin(2\pi\theta) \\ \sin\left(\frac{1}{2}\pi\left(\frac{1}{2}(1-|\phi|)\cos(6\pi\theta)\phi+\phi\right)\right) \end{bmatrix}$$

$\theta \in [0..1] \quad \phi \in [-1..1]$

Discretization

Arbitrary curves have an uncountable number of parameters



i.e. specify function value at all points on real number line

Discretization

Arbitrary curves have an uncountable number of parameters

Pick ***complete*** set of basis functions

$$x(u) = \sum_{i=0}^{\infty} c_i \phi_i(u)$$

- Polynomials, Fourier series, ***etc.***

Truncate set at some reasonable point

$$x(u) = \sum_{i=0}^{3} c_i \phi_i(u) = \sum_{i=0}^{3} c_i \, u^i$$

Function represented by the vector (list) of c_i

The c_i may themselves be vectors

$$\boldsymbol{x}(u) = \sum_{i=0}^{3} \boldsymbol{c}_i \phi_i(u)$$

Polynomial Basis

Power Basis

$$x(u) = \sum_{i=0}^{d} c_i \, u^i$$

$$x(u) = \boldsymbol{C} \cdot \boldsymbol{\mathcal{P}}^d$$

$$\boldsymbol{C} = [c_0, c_1, c_2, \ldots, c_d]$$

$$\boldsymbol{\mathcal{P}}^d = [1, u, u^2, \ldots, u^d]$$

The elements of $\boldsymbol{\mathcal{P}}^d$ are ***linearly independent***

i.e. no good approximation

$$u^k \not\approx \sum_{i \neq k} c_i \, u^i$$

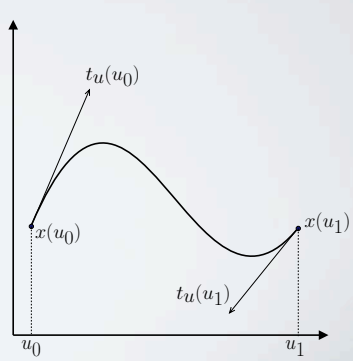
Skipping something would lead to bad results... odd stiffness

Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

For now, assume

$$u_0 = 0 \quad u_1 = 1 \quad x(u) = \sum_{i=0}^{d} c_i \, u^i$$



Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

For now, assume

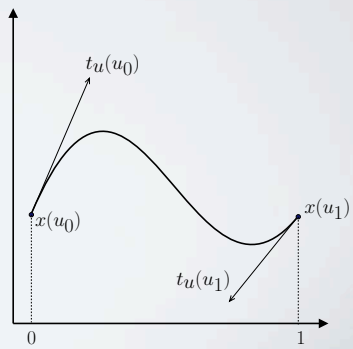
$$u_0 = 0 \quad u_1 = 1 \quad x(u) = \sum_{i=0}^{d} c_i\, u^i$$

$$x(0) = c_0 \quad = x_0$$

$$x(1) = \Sigma\, c_i \quad = x_1$$

$$x'(0) = c_1 \quad = x'_0$$

$$x'(1) = \Sigma\, i\, c_i = x'_1$$



Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

For now, assume

$$u_0 = 0 \quad u_1 = 1 \quad x(u) = \sum_{i=0}^{d} c_i\, u^i$$

$$x(0) = c_0 \quad = x_0$$

$$x(1) = \Sigma\, c_i \quad = x_1$$

$$x'(0) = c_1 \quad = x'_0$$

$$x'(1) = \Sigma\, i\, c_i = x'_1$$

$$\begin{bmatrix} x_0 \\ x_1 \\ x'_0 \\ x'_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$\mathbf{p} = \mathbf{B} \cdot \mathbf{c}$$

Specifying a Curve

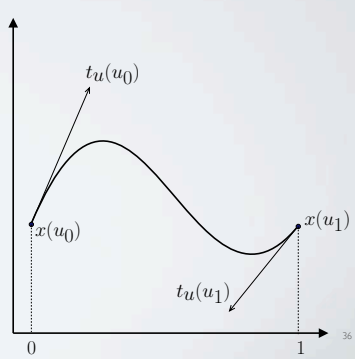
Given desired values (constraints) how do we determine the coefficients for cubic power basis?

$$\mathbf{p} = \mathbf{B} \cdot \mathbf{c}$$

$$\mathbf{c} = \mathbf{B}^{-1} \cdot \mathbf{p}$$

$$\mathbf{c} = \beta_{\mathrm{H}} \cdot \mathbf{p}$$

$$\beta_{\mathbf{H}} = \mathbf{B}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$



Specifying a Curve

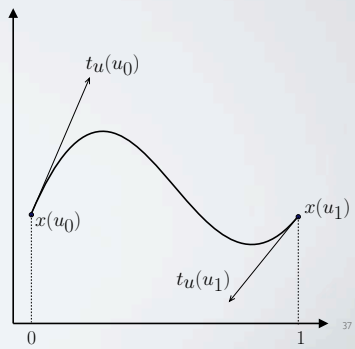
Given desired values (constraints) how do we determine the coefficients for cubic power basis?

$$\mathbf{c} = \beta_{\text{H}} \cdot \mathbf{p}$$

$$x(u) = \mathcal{P}^3 \cdot \mathbf{c} = \boxed{\mathcal{P}^3 \boldsymbol{\beta}_{\text{H}}} \mathbf{p}$$

$$\beta_{\mathbf{H}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 + 0u - 3u^2 + 2u^3 \\ 0 + 0u + 3u^2 - 2u^3 \\ 0 + 1u - 2u^2 + 1u^3 \\ 0 + 0u - 1u^2 + 1u^3 \end{bmatrix} \mathbf{p}$$

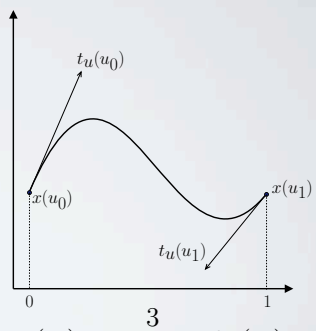


Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

$$\mathbf{c} = \beta_{\text{H}} \cdot \mathbf{p}$$

$$x(u) = \begin{bmatrix} 1 + 0u - 3u^2 + 2u^3 \\ 0 + 0u + 3u^2 - 2u^3 \\ 0 + 1u - 2u^2 + 1u^3 \\ 0 + 0u - 1u^2 + 1u^3 \end{bmatrix} \mathbf{p}$$



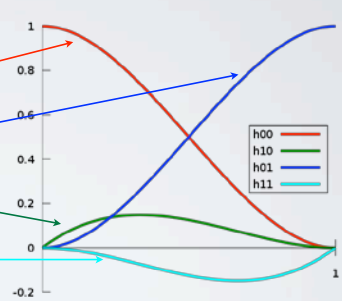
$$x(u) = \sum_{i=0}^{3} p_i b_i(u)$$

Hermite basis functions

Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

$$x(u) = \begin{bmatrix} 1 + 0u - 3u^2 + 2u^3 \\ 0 + 0u + 3u^2 - 2u^3 \\ 0 + 1u - 2u^2 + 1u^3 \\ 0 + 0u - 1u^2 + 1u^3 \end{bmatrix} \mathbf{p}$$



$$x(u) = \sum_{i=0}^{3} p_i b_i(u)$$

Hermite basis functions

Hermite Basis

Specify curve by

- Endpoint values
- Endpoint tangents (derivatives)

Parameter interval is arbitrary (most times)

- Don't need to recompute basis functions

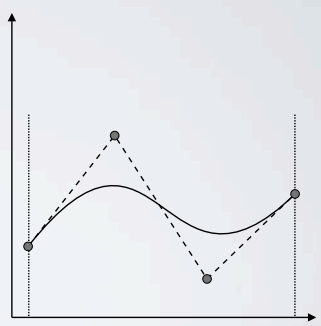
These are ***cubic*** Hermite

- Could do construction for any odd degree
- $(d-1)/2$ derivatives at end points + constant constraints

Cubic Bézier

Similar to Hermite, but specify tangents indirectly

$$
\begin{aligned}
x_0 &= p_0 \\
x_1 &= p_3 \\
x_0' &= 3(p_1 - p_0) \\
x_1' &= 3(p_3 - p_2)
\end{aligned}
$$



Note: All 4 control points are points in space, no tangents
Edges of control polygon tangent to curve at endpts

Cubic Bézier

Similar to Hermite, but specify tangents indirectly

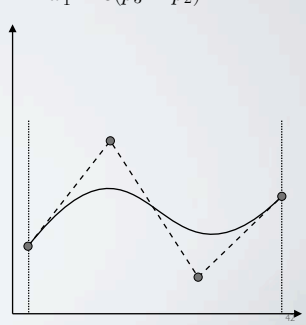
$$
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \mathbf{c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \mathbf{p}
$$

$$
\begin{aligned}
x_0 &= p_0 \\
x_1 &= p_3 \\
x_0' &= 3(p_1 - p_0) \\
x_1' &= 3(p_3 - p_2)
\end{aligned}
$$

$$
\mathbf{c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \mathbf{p}
$$

$$
\mathbf{c} = \boxed{\begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}} \mathbf{p}
$$

$$
\mathbf{c} = \boldsymbol{\beta}_{\mathrm{Z}}\, \mathbf{p}
$$



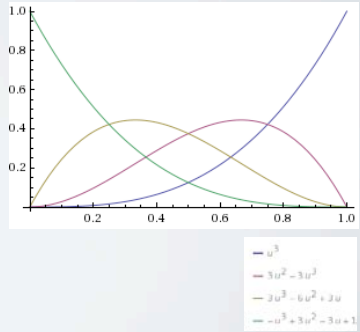
Cubic Bézier

Bézier basis functions

$$\mathbf{c} = \boldsymbol{\beta}_{\mathrm{Z}}\,\mathbf{p} \qquad \mathbf{c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \mathbf{p}$$

$$x(u) = \mathcal{P}^3 \cdot \mathbf{c}$$

$$x(u) = \begin{bmatrix} 1 - 3u + 3u^2 - 1u^3 \\ 0 + 3u - 6u^2 + 3u^3 \\ 0 + 0u + 3u^2 - 3u^3 \\ 0 + 0u + 0u^2 + 1u^3 \end{bmatrix} \mathbf{p}$$



Changing Bases

Power basis, Hermite, and Bézier all are still just cubic polynomials

- The three basis sets all span the same space
- Like different axes in $\cancel{\Re^3}$ $\Re^4$

Changing basis

$$\mathbf{c} = \boldsymbol{\beta}_{\mathrm{Z}}\,\mathbf{p}_{\mathrm{Z}}$$
$$\mathbf{c} = \boldsymbol{\beta}_{\mathrm{H}}\,\mathbf{p}_{\mathrm{H}}$$

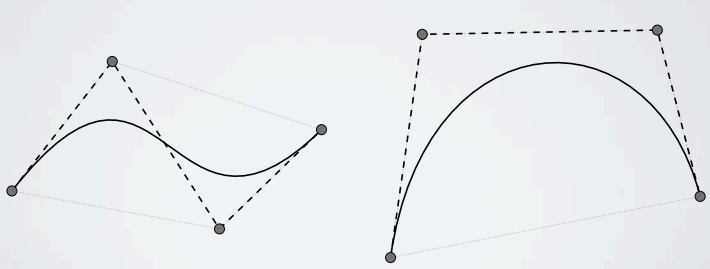
$$\mathbf{p}_{\mathrm{Z}} = \boldsymbol{\beta}_{\mathrm{Z}}^{-1}\,\boldsymbol{\beta}_{\mathrm{H}}\,\mathbf{p}_{\mathrm{H}}$$

Useful Properties of Bézier Basis

Convex Hull

- All points on curve inside convex hull of control points
  - Bézier basis has convex hull property

$$\sum_i b_i(u) = 1 \qquad b_i(u) \geq 0 \qquad \forall u \in \Omega$$



Useful Properties of a Basis

Invariance under class of transforms

- Transforming curve is same as transforming control points
  - Bézier basis invariant for affine transforms

$$\boldsymbol{x}(u) = \sum_i \boldsymbol{p}_i b_i(u) \Leftrightarrow \mathcal{T}\boldsymbol{x}(u) = \sum_i (\mathcal{T}\boldsymbol{p}_i) b_i(u)$$

  - Bézier basis NOT invariant for perspective transforms
    - NURBS are though...

Why is such invariance useful?

Useful Properties of a Basis

Local support

- Changing one control point has limited impact on entire curve

Nice subdivision rules

Orthogonality ($\int_\Omega b_i(u) b_j(u) \mathrm{d}u = \delta_{ij}$)

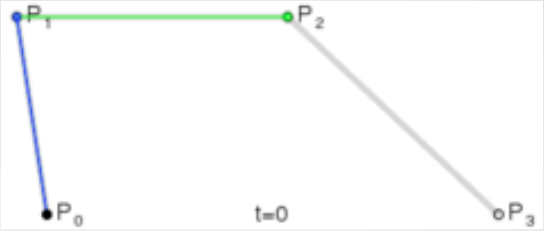
Fast evaluation scheme

Interpolation -vs- approximation

DeCasteljau Evaluation

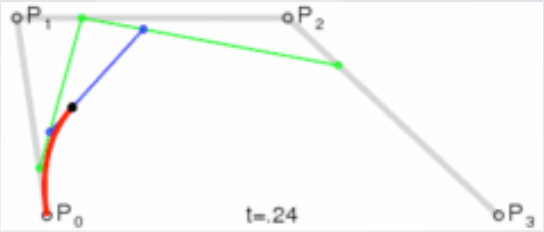
DeCasteljau Evaluation

A geometric evaluation scheme for Bézier



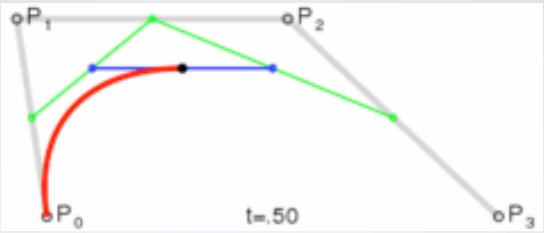
DeCasteljau Evaluation

A geometric evaluation scheme for Bézier



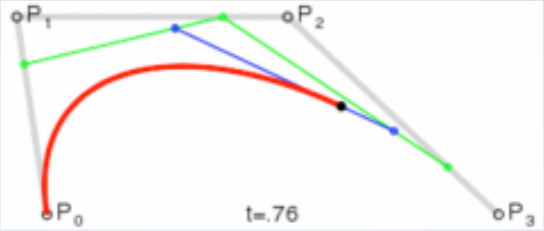
DeCasteljau Evaluation

A geometric evaluation scheme for Bézier



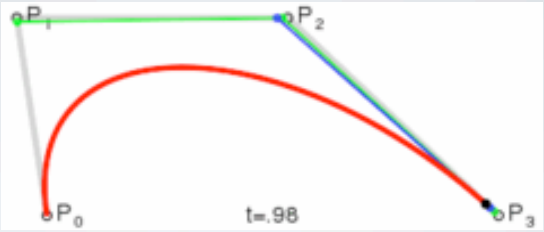
DeCasteljau Evaluation

A geometric evaluation scheme for Bézier



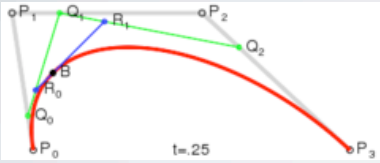
DeCasteljau Evaluation

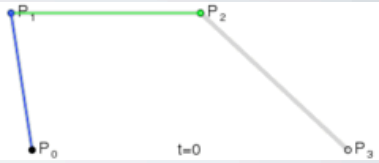
A geometric evaluation scheme for Bézier



DeCasteljau Evaluation

A geometric evaluation scheme for Bézier





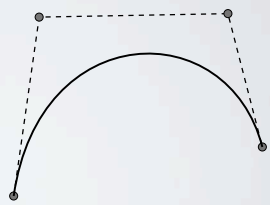
Notice: Blue line always tangent to curve

Curve Drawing: Adaptive Subdiv.

Recall problem: Midpoint test subdivision

- Simple solution if curve basis has ***convex hull*** property

If curve inside convex hull and the convex hull is nearly flat: curve is nearly flat and can be drawn as straight line



Better: draw convex hull

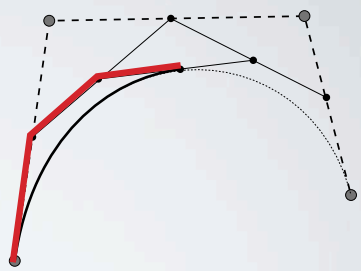
Works for Bézier because the ends are interpolated

Bézier Subdivision

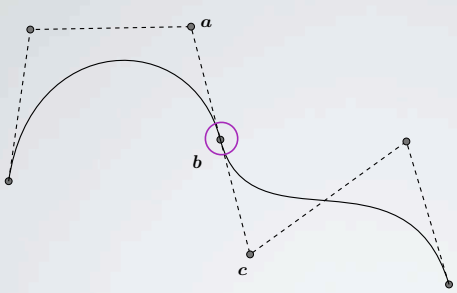
Form control polygon for half of curve by evaluating at $u = 0.5$

Repeated subdivision makes smaller/flatter segments

Also works for surfaces...



Joining



$$\mathcal{C}^0 \Leftrightarrow \boldsymbol{b} = \boldsymbol{b}$$

$$\mathcal{C}^1 \Leftrightarrow \boldsymbol{b} - \boldsymbol{a} = \boldsymbol{c} - \boldsymbol{b}$$

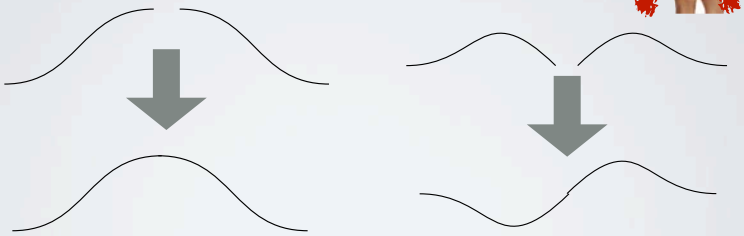
$$\mathcal{G}^1 \Leftrightarrow \frac{\boldsymbol{b} - \boldsymbol{a}}{||\boldsymbol{b} - \boldsymbol{a}||} = \frac{\boldsymbol{c} - \boldsymbol{b}}{||\boldsymbol{c} - \boldsymbol{b}||}$$

If you change $\boldsymbol{a}$, $\boldsymbol{b}$, or $\boldsymbol{c}$ you must change the others

But if you change $\boldsymbol{a}$, $\boldsymbol{b}$, or $\boldsymbol{c}$ you do not have to change beyond those three. *LOCAL SUPPORT*

"Hump" Functions

Constraints at joining can be built in to make new basis



Demo in Illustrator

Extension to Surfaces

Tensor-Product Surfaces

Surface is a curve swept through space

Replace control points of curve with other curves

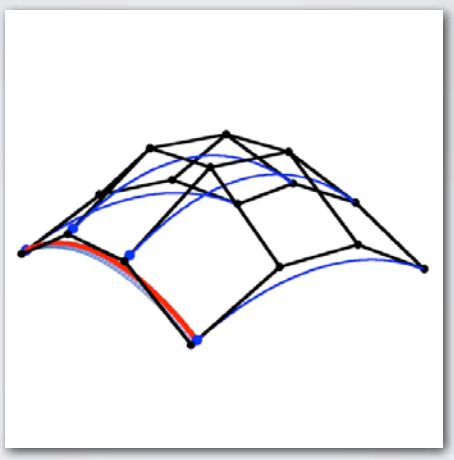
$$x(u,v) = \Sigma_i \, p_i \, b_i(u)$$

$$\Sigma_i \, q_i(v) \, b_i(u) \qquad q_i(v) = \Sigma_j \, p_{ji} \, b_j(v)$$

$$x(u,v) = \sum_{ij} p_{ij} b_i(u) b_j(v) \qquad b_{ij}(u,v) = b_i(u) b_j(v)$$

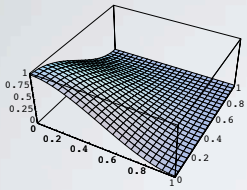
$$x(u,v) = \sum_{ij} p_{ij} b_{ij}(u,v)$$

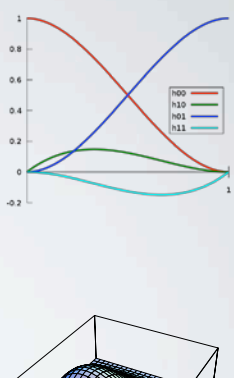
Tensor-Product Surfaces

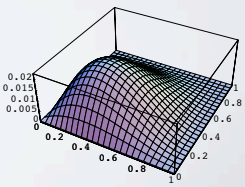


Hermite Surface Bases

Domain [0,1] for u and v



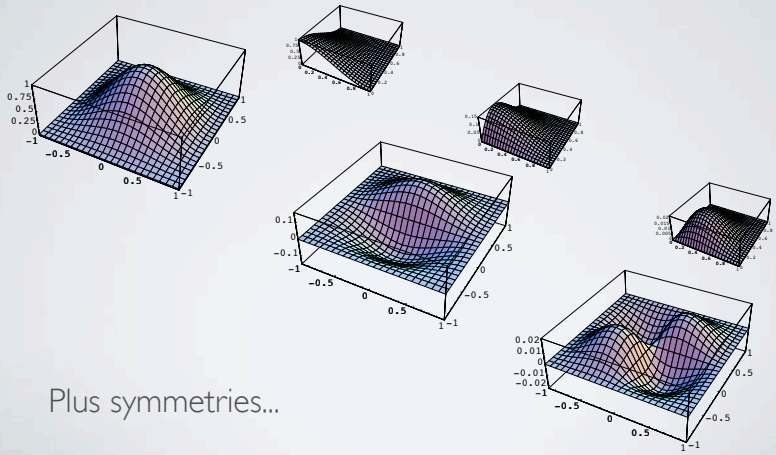




Plus symmetries...

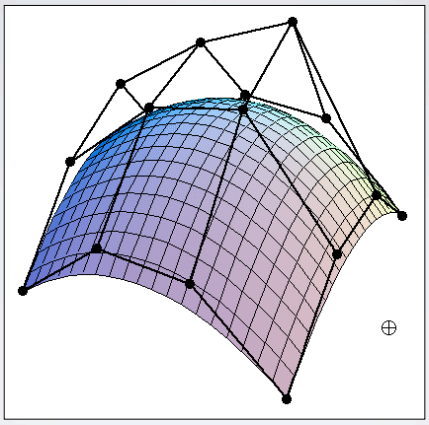
Hermite Surface Hump Functions

Domain [-1,1] for u and v



Plus symmetries...

Bézier Surface Patch



Bezier surface and 4 x 4 array of control points

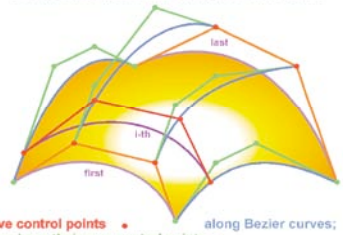
Bezier Surfaces. Smooth Operators.

```
# given the control points of a bezier curve
# and a parametric value, return the curve
# point and derivative
bezcurveinterp(curve, u)
    # first, split each of the three segments
    # to form two new ones AB and BC
    A = curve[0] * (1.0-u) + curve[1] * u
    B = curve[1] * (1.0-u) + curve[2] * u
    C = curve[2] * (1.0-u) + curve[3] * u

    # now, split AB and BC to form a new segment DE
    D = A * (1.0-u) + B * u
    E = B * (1.0-u) + C * u

    # finally, pick the right point on DE,
    # this is the point on the curve
    p = D * (1.0-u) + E * u

    # compute derivative also
    dPdu = 3 * (E - D)

    return p, dPdu
```

```
# given a control patch and (u,v) values, find
# the surface point and normal
bezpatchinterp(patch, u, v)
    # build control points for a Bezier curve in v
    vcurve[0] = bezcurveinterp(patch[0][0:3], u)
    vcurve[1] = bezcurveinterp(patch[1][0:3], u)
    vcurve[2] = bezcurveinterp(patch[2][0:3], u)
    vcurve[3] = bezcurveinterp(patch[3][0:3], u)

    # build control points for a Bezier curve in u
    ucurve[0] = bezcurveinterp(patch[0:3][0], v)
    ucurve[1] = bezcurveinterp(patch[0:3][1], v)
    ucurve[2] = bezcurveinterp(patch[0:3][2], v)
    ucurve[3] = bezcurveinterp(patch[0:3][3], v)

    # evaluate surface and derivative for u and v
    p, dPdv = bezcurveinterp(vcurve, v)
    p, dPdu = bezcurveinterp(ucurve, u)

    # take cross product of partials to find normal
    n = cross(dPdu, dPdv)
    n = n / length(n)

    return p, n
```

Bicubic Bezier Patch

Continuously Moved and Deformed Bezier Curve

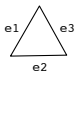


Move control points along Bezier curves;
these have their own control points;
leading to a total of 16 control points for the cubic case.

```
# given a patch, perform uniform subdivision
subdividepatch(patch, step)
    # compute how many subdivisions there
    # are for this step size
    numdiv = ((1 + epsilon) / step)

    # for each parametric value of u
    for (iu = 0 to numdiv)
        u = iu * step

        # for each parametric value of v
        for (iv = 0 to numdiv)
            v = iv * step

            # evaluate surface
            p, n = bezpatchinterp(patch, u, v)
            savesurfacepointandnormal(p,n)
```

Split?

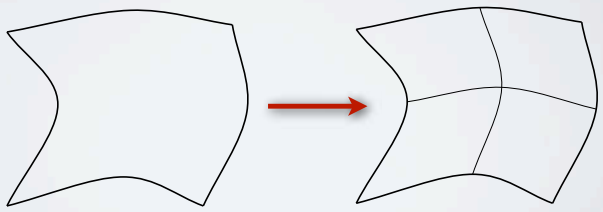
e3	e2	e1	
0	0	0	output as is
0	0	1	
0	1	0	
1	0	0	
0	1	1	
1	1	0	
1	0	1	
1	1	1	



Adaptive Tessellation

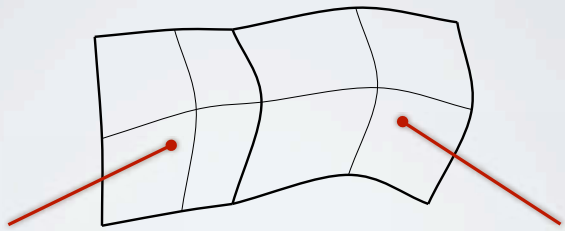
Given surface patch

- If close to flat: draw it
- Else subdivide 4 ways



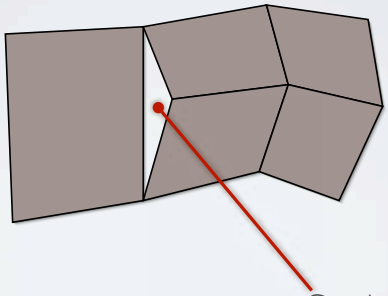
Adaptive Tessellation

Avoid cracking



Adaptive Tessellation

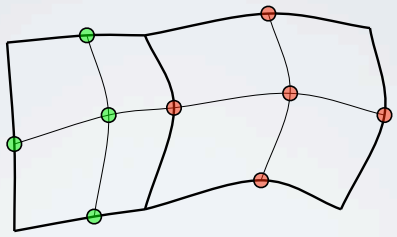
Avoid cracking



Cracks may be okay in some contexts...

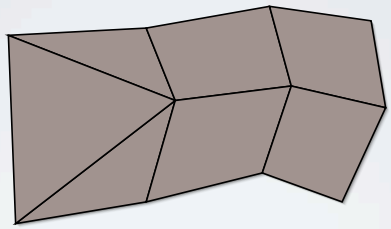
Adaptive Tessellation

Avoid cracking



Adaptive Tessellation

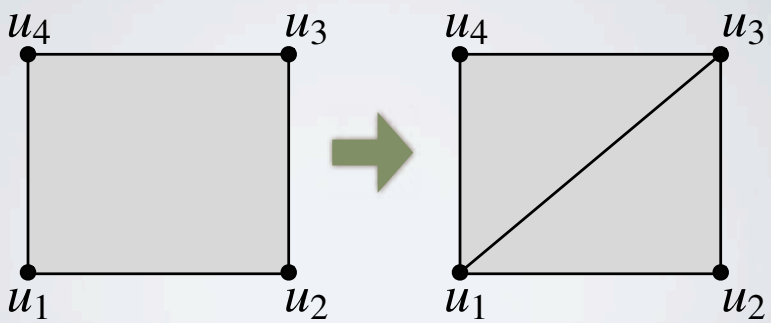
Avoid cracking



Test interior and boundary of patch
Split boundary based on boundary test
Table of polygon patterns
May wish to avoid "slivers"

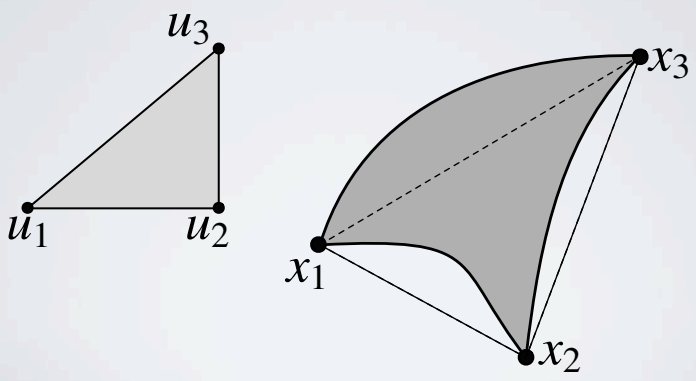
Adaptive Tessellation

Triangle Based Method (no cracks)



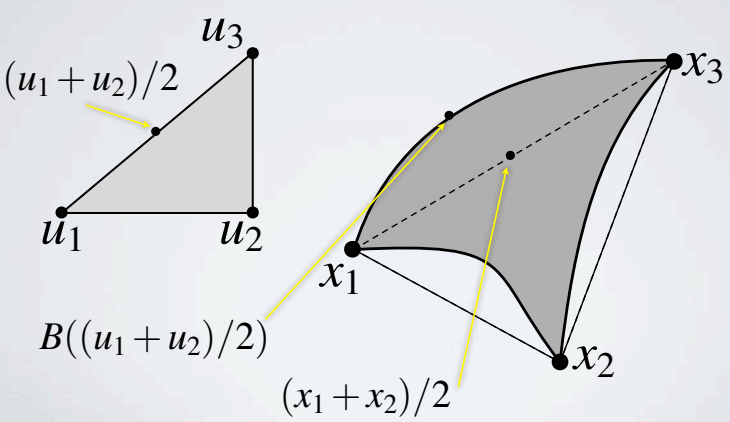
Adaptive Tessellation

Triangle Based Method (no cracks)



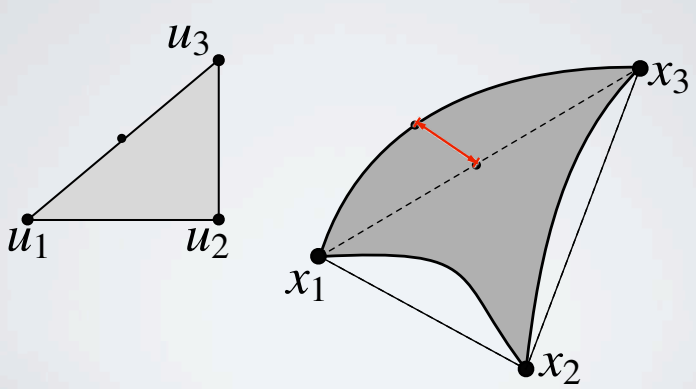
Adaptive Tessellation

Triangle Based Method (no cracks)



Adaptive Tessellation

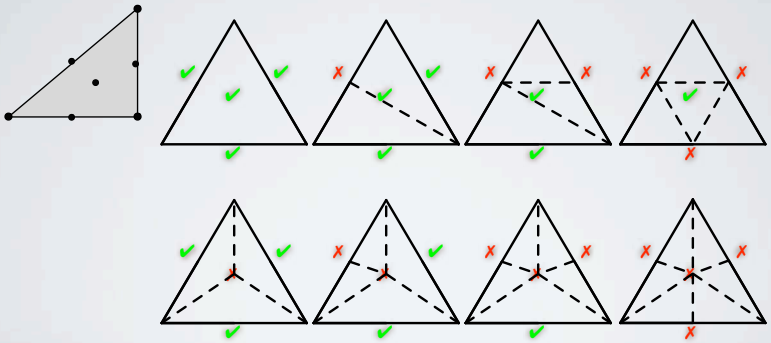
Triangle Based Method (no cracks)



$||B((u_1+u_2)/2)-(x_1+x_2)/2|| < \tau$?

Adaptive Tessellation

Triangle Based Method (no cracks)



Center test tends to generate slivers.
Often better to leave it out.

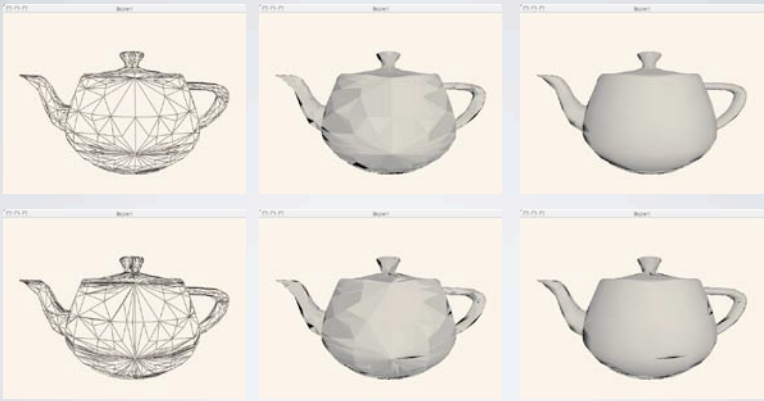
Adaptive Tessellation

Without center test

With center test

Yiding Jia, CS184 S08

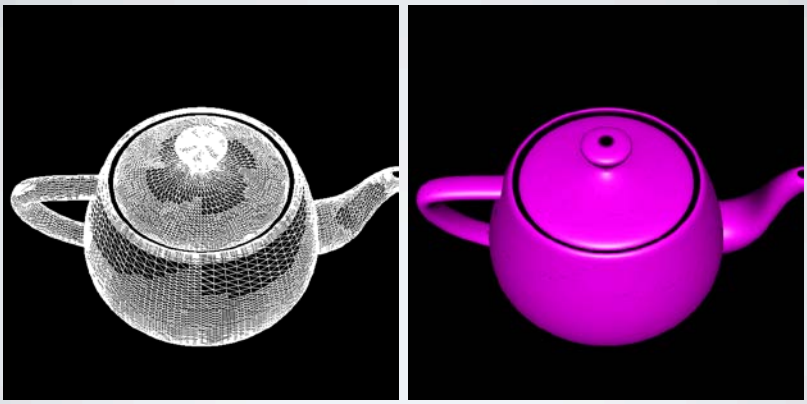
Adaptive Tessellation



Second row shows typical error of swapping tests.

Yiding Jia, CS184 S08 -- I broke his code to make this example.

Adaptive Tessellation



Visible artifacts from cracks.