# Johns Hopkins
# Engineering for Professionals

## 605.767 Applied Computer Graphics

Brian Russin

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

# Module 6G
# Rendering Parametric Curves and Surfaces

# Rendering Parametric Curves and Surfaces

- Primary methods for rendering parametric curves and surfaces
  - Iterative evaluation
    - Forward differencing
  - Recursive subdivision
- Tessellation is process of creating triangles on the surface of a parametric surface
- Iterative evaluation of a Bezier surface patch
  - Solve p(u,v) for incrementally spaced values of u and v
    - x(u,v), y(u,v), z(u,v)
    - Uniform sampling of u,v
    - Example: create 10 evenly spaced points
      - $(u_k, v_l)=(0.1k, 0.1l)$
  - Create 2 triangles for each set of 4 surface points
    - $p(u_k, v_l)$, $p(u_{k+1}, v_l)$, $p(u_k, v_{l+1})$, $p(u_{k+1}, v_{l+1})$
  - Straightforward, but not very efficient

# Rendering Parametric Curves and Surfaces

- Bezier curve can be written as $f(t) = p(t) = \sum_{i=0}^{n} t^i c_i$
- **Forward differences** can be used to rapidly evaluate such polynomials in uniformly spaced intervals

  $$f_k = f(sk)$$

    - Evaluate at 0, s, 2s, 3s, 4s, etc…
  - Compute initial values for the curve and delta values
    - Delta values which are also simplified using forward differences
  - Cubic polynomials require 3 forward differences
    - Text has example for a quadratic polynomial

$\delta_k^1 = f_{k+1} - f_k$ 1st forward difference

$\delta_k^2 = \delta_{k+1}^1 - \delta_k^1$ 2nd forward difference

$\delta_k^3 = \delta_{k+1}^2 - \delta_k^2$ 3rd forward difference

```
Compute initial f, d1, d2, d3
for (i = 1 to l)
   f   += d1
   d1 += d2
   d2 += d3
```

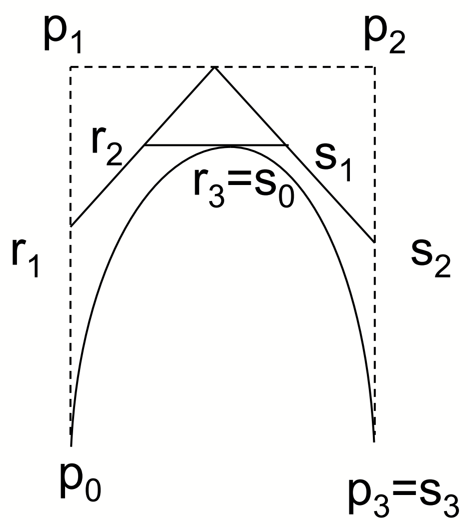
Pseudocode for Forward Differences

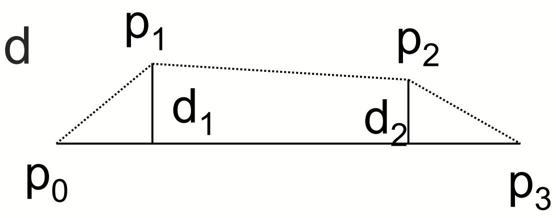
# Recursive Subdivision Techniques

- Recursively subdividing spline curves and surfaces is a common rendering method
  - Can also be used to produce more control points to allow better shaping of a curve
- Subdivision technique
  - Repeatedly divide a curve section in half
    - Creating additional control points
  - Until the control points are sufficiently close to the desired curve
- Recursive subdivision is most easily done with Bezier representations
  - Curve always goes through first and last points
  - Parameter t is between 0 and 1
  - Easy to determine when the control points are close enough to the curve

# Bezier Recursive Subdivision

- Bezier curves are subdivided by subdividing the control points
  - Creates two new sets of control points: $r_i$, $s_i$
  - $r_0 = p_0$,     $r1 = (p_0 + p_1) / 2$,
  - $r_2 = r_1/2 + (p_1 + p_2) / 4$
  - $r_3 = s_0 = (r_2 + S_1) / 2$
  - $s_1 = (p_1 + p_2) / 4 + s_2 / 2$
  - $s_2 =(p_2 + p_3) / 2$,   $s_3 = p_3$
- Each successive subdivision will be closer to the curve
- Can control the depth of the subdivision with a linearity criteria
  - Can test the distance from the middle two control points to the end point joining line
  - Stop subdividing when the distances are below a tolerance



# Subdivision of Bezier Surfaces – Patch Splitting

- **Patch splitting** extends recursive subdivision of curves to surfaces
  - Creates nearly planar quadrilaterals
  - Split the surface along one parameter and then split each of the two resulting surfaces along the other parameter
    - Apply curve splitting method to each set of 4 control points in the direction of the parameter along which the split is made
  - Simplest if Bezier representation is used

# Subdivision of Bezier Surfaces (cont.)

- **Uniform patch splitting**
  - Subdivide to a specified level (# of subdivisions) over entire surface
  - Fast and flexible
    - Can vary the depth of the subdivision to affect speed
  - Some areas become unnecessarily subdivided
    - May be nearly flat already
  - Simpler, but less elegant than an adaptive subdivision
  - Can become a preprocessing stage in the rendering engine
    - Convert parametric surfaces to triangles/quadrilaterals

# Adaptive Patch Splitting

- **Recursive** or **adaptive subdivision**
  - Subdivide individual patches until **flatness test** is passed
    - Flatness is tested against a plane through 3 of the 4 corner points
      - Find distances of the other 13 control points to the plane
      - Test against tolerance
  - Areas become subdivided in relation to their degree of local curvature
    - Flat areas - less subdivision into fewer, larger polygons
    - High curvature - more subdivision into more, smaller polygon

# Tears in Patch Splitting

- Adaptive subdivision can lead to 'tears' or 'cracks'
    - Different levels of subdivision between two patches sharing a boundary
    - Will appear as a hole or gap in the rendered image Eliminating tears
  - Tears can be avoided by uniform subdivision or making the flatness criteria very small
    - Both cause needless subdivisions
- Can modify the approximating quadrilaterals to eliminate the tear