# NURBS (Redbook ch12)

29 March 2007
CMPT370
Dr. Sean Ho
Trinity Western University

*IBiblio e-notes*

*Cambridge notes*

TRINITY WESTERN UNIVERSITY

# Review last time

- Polynomial curves and surfaces
- Cubic polynomial curves:
  - Interpolating
  - Hermite
  - Bezier
  - Using Bezier evaluators in OpenGL

# Bezier evaluators in OpenGL

- Specify array (1D or 2D) of control points:
  - GLfloat ctrlpoints[4][3] = { {-4.0, -4.0, 0.0}, ...
- Create a Bezier evaluator: *(type=GL_MAP1_VERTEX_3)*
  - glMap1f( *type*, $u_{min}$, $u_{max}$, *stride*, *order*, *points* );
- Enable the evaluator:
  - glEnable( *type* );
- Evaluate the Bezier at a particular u/v:
  - glEvalCoord1f( (GLfloat) u );
    - Use this instead of glVertex(), e.g., within glBegin( GL_LINE_STRIP )

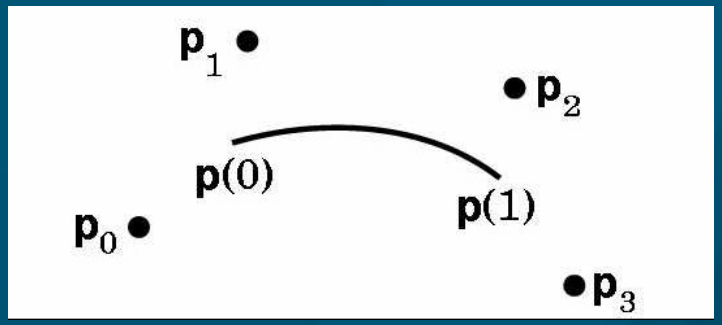
# How OpenGL computes Beziers

- de Casteljau's algorithm:
- 4 control points:
  - Plot a point u of the way from $p_0$ to $p_1$
  - Similarly between $(p_1, p_2)$, and $(p_2, p_3)$
  - Get 3 points $(q_0, q_1, q_2)$
- Plot points u of the way between $(q_0, q_1)$, $(q_1, q_2)$
  - Get 2 points $(r_0, r_1)$
- Plot a point u of the way between $(r_0, r_1)$
  - This is our point on the Bezier curve

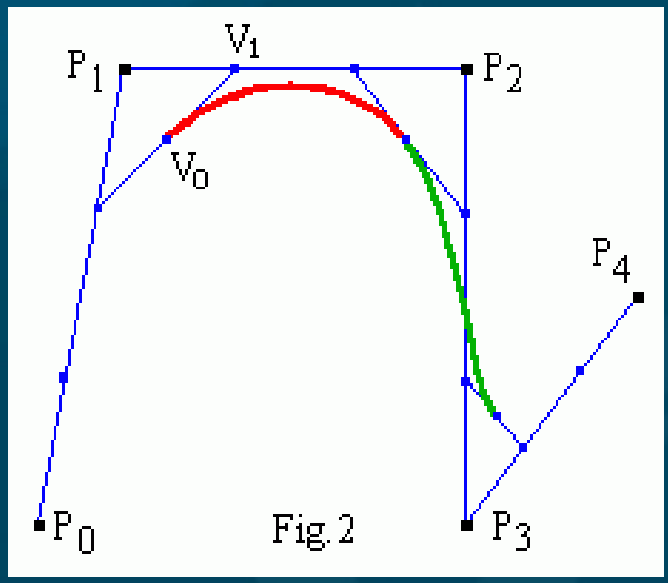
# Splines

- Draftsman's tool for drawing smooth curves:
  - Flexible wood/plastic strip
  - Bent to pass through knots (control points)
- A spline is any sort of smooth curve that has a series of control points
  - Interpolating splines
    - Interpolating cubic spline
    - Interpolating Catmull-Rom spline
  - Cubic Bezier is a spline
  - B-splines

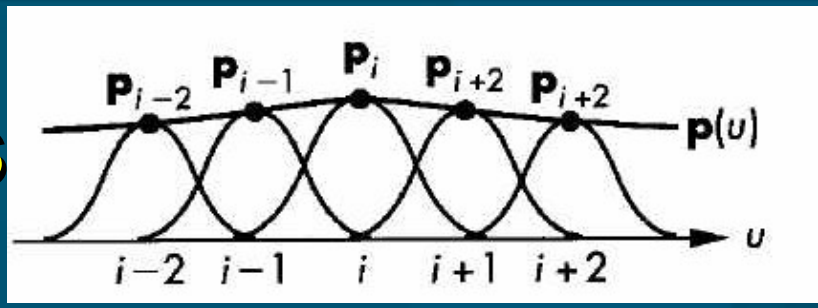
# Cubic B-splines




- n+1 deBoor control points $p_0$, ..., $p_n$.
- Make n-2 Bezier curve segments
  - Want $C^2$ at the joins; sacrifice interpolation
  - Derive Bezier control points ($v_0$, $v_1$, $v_2$, $v_3$) from deBoor points ($p_0$, $p_1$, $p_2$, $p_3$):
  - $v_1$ is (1/3)-way between $p_1$ and $p_2$
  - $v_0$ is halfway between $v_1$ and $(1/3)p_0 + (2/3)p_1$
- Cubic B-spline: order k=4



# B-spline basis functions




- Basis functions are simple polynomials
- Region of influence for each deBoor control point is 4 Bezier segments
- Each point on the curve is affected by 4 deBoor control points
- Knots ($u_0$, ..., $u_{n+4}$) specify where the joins are in parameter space: e.g., {0.0, 0.25, 0.50, 0.75, 1.0}
- Uniform B-spline: uniform spacing of knots
- Open-spacing: duplicate end knots to get interpolation:
  - {0.0, 0.0, 0.0, 0.0, 0.25, 0.50, 0.75, 1.0, 1.0, 1.0, 1.0}

# NURBS

- Spline:
  - Smoothish curve defined by control points
- B-spline:
  - Joined Bezier curves with $C^2$ continuity
- Non-uniform B-spline:
  - Non-uniform spacing of knots (e.g., can use multiplicity to get interpolation of endpoints)
- Rational B-spline:
  - Add weights to each control point
  - Takes advantage of perspective division hardware

# Properties of NURBS

- More computationally expensive than Bezier curves/patches
- $C^2$ continuity makes shading look much better
- Local control: moving a control point only affects 4 Bezier segments
- Convex hull property: each point on the spline is within the convex hull of the four control points it's affected by
- Affine-invariant (including perspective):
  - Transforming control points = transforming curve