Cg Example – Vertex Shader

- Vertex Shader: animated teapot

```cpp
void main(
    // input
    float4 position : POSITION, // position in object coordinates
    float4 normal : NORMAL, // normal
    float2 texCoord0 : TEXCOORD0, // texturing coordinates
    float2 texCoord1 : TEXCOORD1, // texturing coordinates
    float2 texCoord2 : TEXCOORD2, // texturing coordinates

    // output
    out float4 outPosition: POSITION, // position in clip space
    out float4 outNormal: NORMAL, // normal in model coordinates
    out float4 outColor : COLOR ) // out color
{

    // compute the ambient term
    outColor.rgb = materialAmbient * lightAmbient
        * attenuationFactor * multiply2f(modelViewMatrixIT, position Object)
        * multiply2f(projectionMatrix, outColor.rgb);

    // compute the diffuse term
    outColor.rgb += materialDiffuse * lightDiffuse
        * attenuationFactor // inverse of 1 + materialEmission
        * multiply2f(modelViewMatrixIT, normalize(normalObject.xyz))
        * multiply2f(projectionMatrix, outColor.rgb);

    // compute the specular term
    outColor.rgb += materialSpecular * lightSpecular
        * pow(max(dot(normalObject, lightPositionWorld), 0.0), specularFactor)
        * attenuationFactor
        * multiply2f(modelViewMatrixIT, normalize(normalObject.xyz - lightPositionWorld))
        * multiply2f(projectionMatrix, outColor.rgb);

    // compute the emission term
    float4 emissiveColor = color1 + color2;
    outColor.rgb += emissiveColor;
}
```

Modern Hardware II, Curves

- Modern Hardware
  - finish up nice slides by Gordon Wetzstein
  - lecture 23 from
    - slides, downloadable demos
GPGPU
- general purpose computation on the GPU
- in the past: access via shading languages and rendering pipeline
- now: access via cuda interface in C environment

GPGPU Applications

Splines
- a spline is a parametric curve defined by control points
- term "spline" dates from engineering drawing, where a spline was a piece of flexible wood used to draw smooth curves
- control points are adjusted by the user to control shape of curve

Bézier Blending Functions
- every point on curve is linear combination of control points
- weights of combination are all positive
- sum of weights is 1
- therefore, curve is a convex combination of the control points

Sample Hermite Curves

Bézier Curves
- similar to Hermite, but more intuitive definition of endpoint derivatives
- four control points, two of which are knots

Curves

Splines - History
- draftsman used 'ducks' and strips of wood (splines) to draw curves
- wood splines have second-order continuity, pass through the control points
- a duck (weight) ducks trace out curve

Reading
- FCG Chap 15 Curves
- Ch 13 2nd edition

Bézier Curves
- curve will always remain within convex hull (bounding region) defined by control points

Bézier Curves
- interpolate between first, last control points
- 1st point’s tangent along line joining 1st, 2nd pts
- 4th point’s tangent along line joining 3rd, 4th pts

Parametric Curves
- parametric form for a line:
  
  \[ x = x_0 + (1-t)x_1 \]
  
  \[ y = y_0 + (1-t)y_1 \]
  
  \[ z = z_0 + (1-t)z_1 \]

  - x, y and z are each given by an equation that involves:
    - parameter t
    - some user specified control points, \( x_0 \) and \( x_1 \)
  - this is an example of a parametric curve

Basis Functions
- a point on a Hermite curve is obtained by multiplying each control point by some function and summing
- functions are called basis functions

Bézier Curves
- derivative values of Bezier curve at knots dependent on adjacent points
  
  \[ V_p = 3(p_{i+1} - 2p_i + p_{i-1}) \]
  
  \[ V_{p_2} = 3(p_3 - p_1) \]
  
  \[ V_{p_3} = 3(p_4 - p_2) \]

Splines
- a spline is a parametric curve defined by control points
- term "spline" dates from engineering drawing, where a spline was a piece of flexible wood used to draw smooth curves
- control points are adjusted by the user to control shape of curve

Bézier Curves
- Hermite spline is curve for which user provides:
  - endpoints of curve
  - parametric derivatives of curve at endpoints
    - parametric derivatives are \( dx/dt, dy/dt, dz/dt \)
  - more derivatives would be required for higher order curves

Hermite Spline

Splines
- parametric form for a line:
  
  \[ x = x_0 + (1-t)x_1 \]
  
  \[ y = y_0 + (1-t)y_1 \]
  
  \[ z = z_0 + (1-t)z_1 \]

  - x, y and z are each given by an equation that involves:
    - parameter t
    - some user specified control points, \( x_0 \) and \( x_1 \)
  - this is an example of a parametric curve

Basis Functions
- a point on a Hermite curve is obtained by multiplying each control point by some function and summing
- functions are called basis functions

Bézier Blending Functions
- family of polynomials called order-3 Bernstein polynomials
  - \( C(3, k) \); \( 0 \leq k \leq 3 \)
  - all positive in interval \([0,1]\)
  - sum is equal to 1
Comparing Hermite and Bézier

- Hermite
- Bézier

Geometric Continuity

- Geometric version of C² is G², based on curves having the same radius of curvature across the knot

Sub-Dividing Bezier Curves

- Step 2: find the midpoints of the lines joining \( M_{012} \) and \( M_{123} \). Call them \( M_{0123} \).

- Step 3: find the midpoint of the line joining \( M_{0123} \) and \( M_{123} \). Call it \( M_{0123} \).

de Casteljau's Algorithm

- Can find the point on a Bezier curve for any parameter value \( t \) with similar algorithm

- For \( t=0.25 \), instead of taking midpoints take points 0.25 of the way

Rendering Bezier Curves: Simple

- Evaluate curve at fixed set of parameter values, join points with straight lines
- Advantage: very simple
- Disadvantages:
  - Expensive to evaluate the curve at many points
  - No easy way of knowing how fine to sample points, and maybe sampling rate must be different along curve
  - No easy way to adapt: hard to measure deviation of line segment from exact curve

Rendering Bezier Curves: Subdivision

- A cubic Bezier curve can be broken into two shorter cubic Bezier curves that exactly cover original curve

- Suggests a rendering algorithm:
  - Keep breaking curve into sub-curves
  - Stop when control points of each sub-curve are nearly collinear
  - Draw the control polygon: polygon formed by control points

B-Spline

- By far the most popular spline used

- \( C_0 \), \( C_1 \), and \( C_2 \) continuous

- \( C_0 \), \( C_1 \), \( C_2 \), \( C_3 \) continuity

Piecewise Bezier: Continuity Problems

- When two curves joined, typically want some degree of continuity across knot boundary

- \( C_0 \), \( C_1 \), point-wise continuous, curves share same point where they join

- \( C_1 \), \( C_2 \), continuous derivatives

- \( C_2 \), \( C_3 \), continuous second derivatives

Continuity

- \( P_0, P_1, P_2, P_3 \) a single cubic Bezier or Hermite curve can only capture a small class of curves

- At most 2 inflection points

- One solution is to raise the degree

- Adopts more control, at the expense of more control points and higher degree polynomials

- Control is not local, one control point influences entire curve

- Better solution is to join pieces of cubic curve together into piecewise cubic curves

- Total curve can be broken into piece, each of which is cubic

- Local control: each control point only influences a limited part of the curve

- Interaction and design is much easier

Sub-Dividing Bezier Curves

- Continue process to create smooth curve

- \( P_0, P_1, P_2, P_3 \) a cubic Bezier curve can be broken into two shorter cubic Bezier curves that exactly cover original curve

- When control points of each sub-curve are nearly collinear

- Draw the control polygon: polygon formed by control points

- \( C_0 \), \( C_1 \), \( C_2 \), \( C_3 \) continuity

- \( C_0 \), \( C_1 \), \( C_2 \), \( C_3 \) continuity

- \( C_0 \), \( C_1 \), \( C_2 \), \( C_3 \) continuity

- \( C_0 \), \( C_1 \), \( C_2 \), \( C_3 \) continuity

Geometric Continuity

- Geometric continuity is important for animation

- If object moves along curve with constant parametric speed, should be no sudden jump at knots

- For other applications, tangent continuity suffices

- Requires that the tangents point in the same direction

- Referred to as \( G^1 \) geometric continuity

- Curves could be made \( C^1 \) with a re-parameterization

- Geometric version of \( C^1 \) is \( G^1 \), based on curves having the same radius of curvature across the knot

Achieving Continuity

- Hermite curves

- User specifies derivatives, so \( C^1 \) by sharing points and derivatives across knot

- Bezier curves

- They interpolate endpoints, so \( C^0 \) by sharing control pts

- Introduce additional constraints to get \( C^1 \)

- Parametric derivative is a constant multiple of vector joining first/last 2 control points

- \( C^1 \) achieved by setting \( p_{10} = p_{11} \), and making \( p_{11} \) and \( J \) and \( P_{11} \) collinear, with \( J = p_{11} \) and \( P_{11} \)

- \( C^2 \) comes from further constraints on \( P_{11} \) and \( P_{12} \)

- Leads to...
B-Spline

- locality of points

Figure 10.16
Local modification of a B-spline curve. Changing one of the control points in the previous curve fit, which is modified only in the neighborhood of the altered control point.