



Information Coding / Computer Graphics, ISY, LiTH

# **Splines and surfaces in OpenGL**

**Pre-generated shapes on CPU**

**Generate by multi-pass GPU processing**

**Old OpenGL: Evaluators (glMap)**

**3.2: Geometry shaders**

**4: Tessellation shaders**



Information Coding / Computer Graphics, ISY, LiTH

# **Pre-generating splines**

**Evaluation of polynomials many times**

**Optimizations will pay!**



## Evaluating polynomials

Important problem for efficient spline calculations.

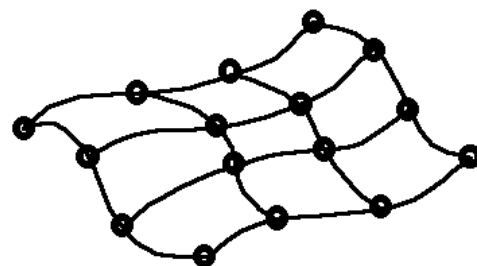
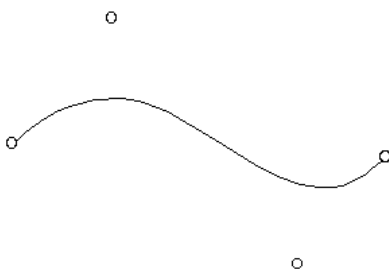
- 1) Horner's Rule
- 2) Forward-difference calculations



## Evaluators

Old built-in Bezier evaluator. Easy to use but no longer recommended.

Supported both curves and surfaces





# Evaluators

**Very straight-forward but not very flexible.**

```
glMap1f(GL_MAP1_VERTEX_3, u0, u1, 3, 4, &data2[0][0]);
glEnable(GL_MAP1_VERTEX_3);
glBegin(GL_LINE_STRIP);
for (int i = 0; i <= 20; i++)
    glEvalCoord1f(u0 + i*(u1-u0)/20);
glEnd();
```

Control points

Evaluation, specifies  
vertices



# Geometry shaders

OpenGL 3 (extension in GL 2)

**Shader between vertex and fragment, converts geometry, can add new vertices**

**Modest hardware demand: G80 or better (2007+)**

**Core functionality since GL3 - but will it last? Some irritating limitations. Criticised for low performance.**

**GL4 adds tessellation shaders, more like Evaluators.**



## Applications:

- Splines/surfaces
  - Edge extraction, silhouettes
- Polygon-level effects (shrinking triangles)
  - Adaptive subdivision
- Visualizing normal vectors etc



## Geometry shader examples

Input: Single triangle (as our first example)

Load geometry shader together with vertex and fragment.

Input to shader: triangle, lines or point

Output: triangle strip, line strip



## Pass-through geometry shader

```
#version 150

layout(triangles) in;
layout(triangle_strip, max_vertices = 3)
out;

void main()
{
    for(int i = 0; i < gl_in.length(); i++)
    {
        gl_Position = gl_in[i].gl_Position;
        EmitVertex();
    }
}
```

**Just pass on the same vertices**

**EmitVertex tells that we have a finished vertex ready**



## Bezier curve geometry shader

```
// Simple 3-point spline geometry shader
#version 150

layout(triangles) in;
layout(line_strip, max_vertices = 50) out;

// quadratic bezier 0..1
vec4 bez3(vec3 a, vec3 b, vec3 c, float u)
{
    float aw = (1-u)*(1-u);
    float bw = 2*(1-u)*u;
    float cw = u*u;
    return vec4(a*aw+b*bw+c*cw, 1.0);
}

void main()
{
    for (int i = 0; i <= 20; i++)
    {
        gl_Position = bez3(
            vec3(gl_in[0].gl_Position),
            vec3(gl_in[1].gl_Position),
            vec3(gl_in[2].gl_Position),
            float(i)/20.0);
        EmitVertex();
    }
}
```