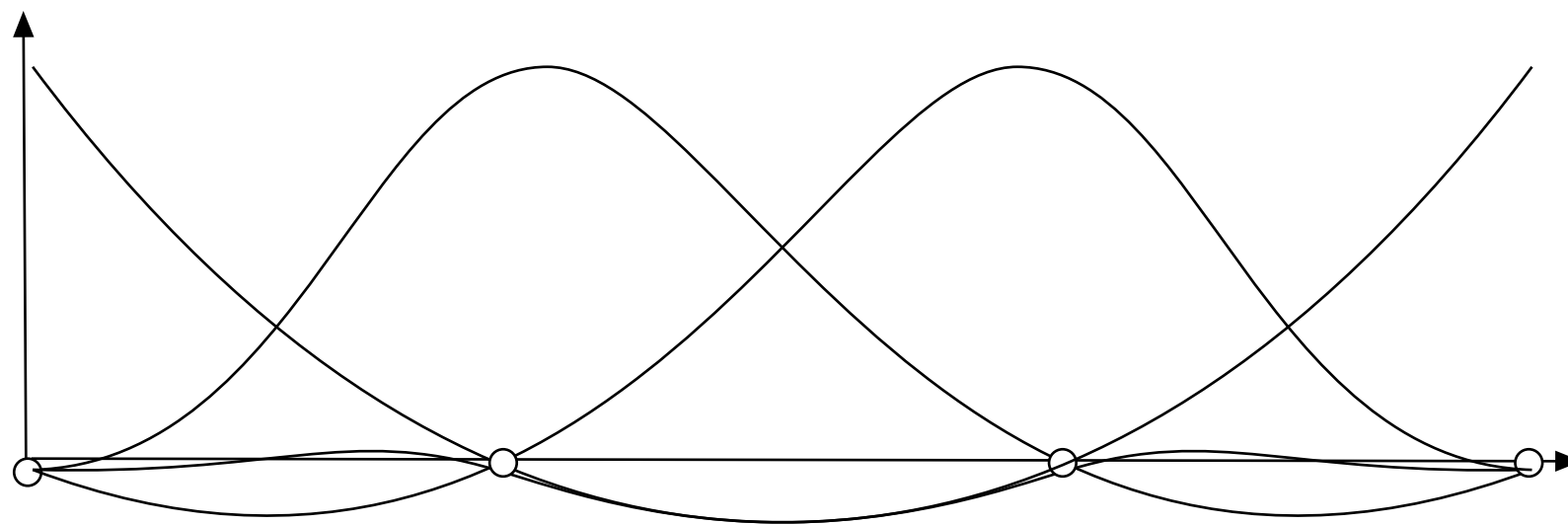




Blending functions for interpolation spline

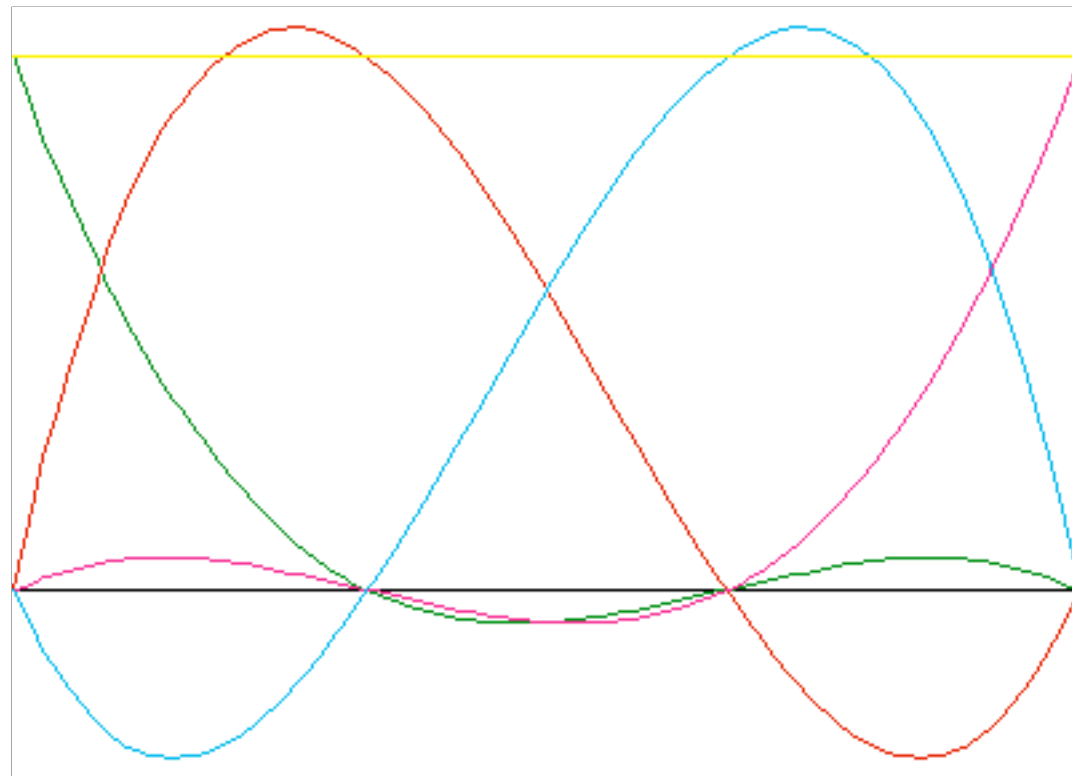
The points are *blended* together using blending functions





Blending functions for interpolation spline

All blending functions are zero or 1 at the control points!



Actual blending functions for interpolated spline of 4 control points (similar to Bézier)



Cardinal splines Catmull-Rom splines

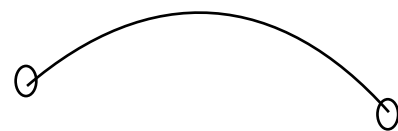
Interpolation spline

Specified *only* by control points

Calculated from 4 control points,
define between the middle two!

A tension parameter t can adjust
the shape

$t = 0 \Rightarrow$ Catmull-Rom



0

0



Catmull-Rom splines, Matrix form

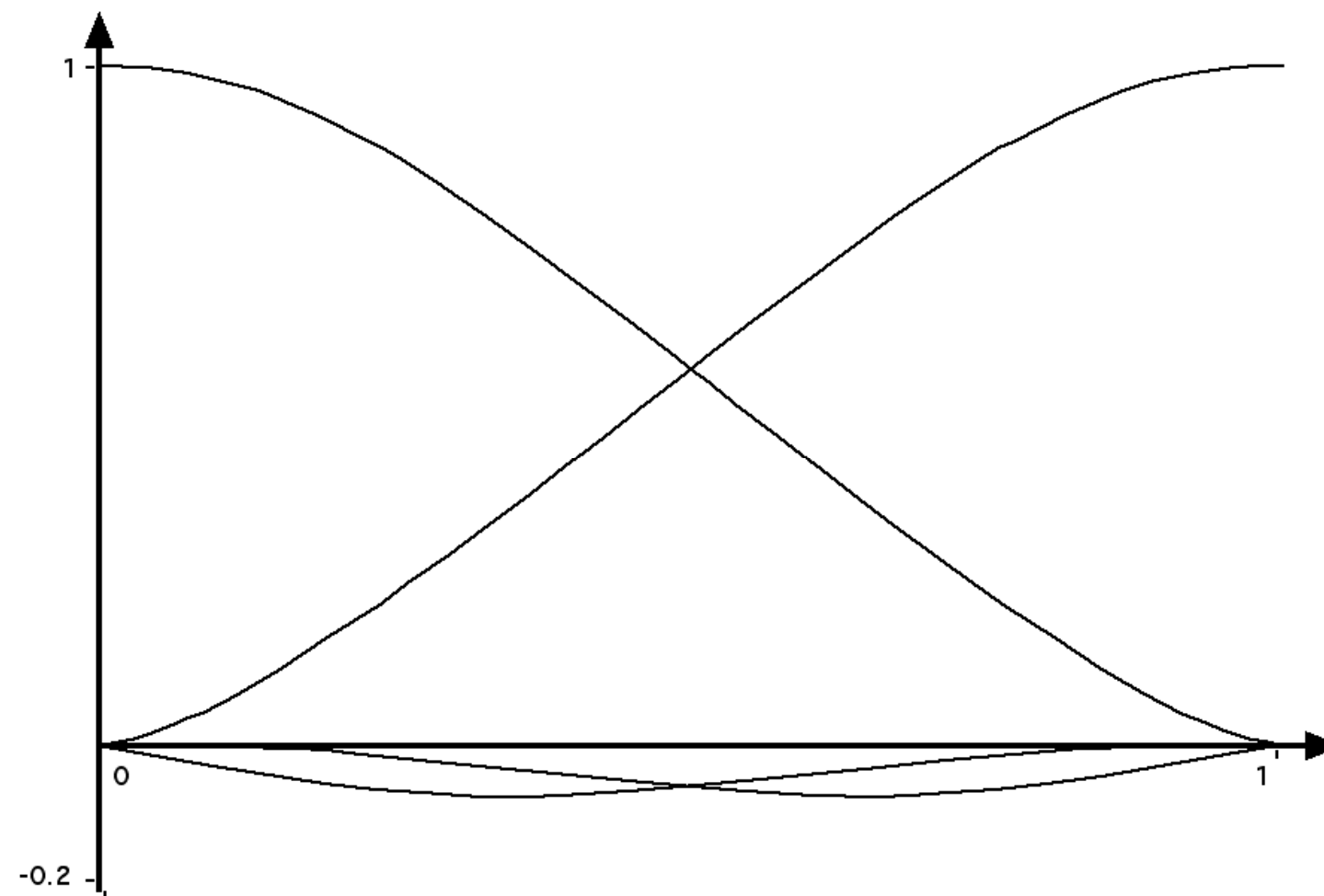
$$P(u) = [u^3 \ u^2 \ u \ 1] \begin{bmatrix} -1/2 & 3/2 & -3/2 & 1/2 \\ 1 & -5/2 & 2 & -1/2 \\ -1/2 & 0 & 1/2 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{k-1} \\ p_k \\ p_{k+1} \\ p_{k+2} \end{bmatrix}$$

$$P(u) = p_{k-1} (-u^3/2 + u^2 - u/2) + \\ p_k (3u^3/2 - 5u^2/2 + 1) + \\ p_{k+1} (-3u^3/2 + 2u^2 + u/2) + \\ p_{k+2} (u^3/2 - u^2/2)$$

$$= p_{k-1} * CAR_0(u) + p_k * CAR_1(u) + \\ p_{k+1} * CAR_2(u) + p_{k+2} * CAR_3(u)$$



Catmull-Rom splines, Blending functions





Application of Catmull-Rom splines:

Animation paths!

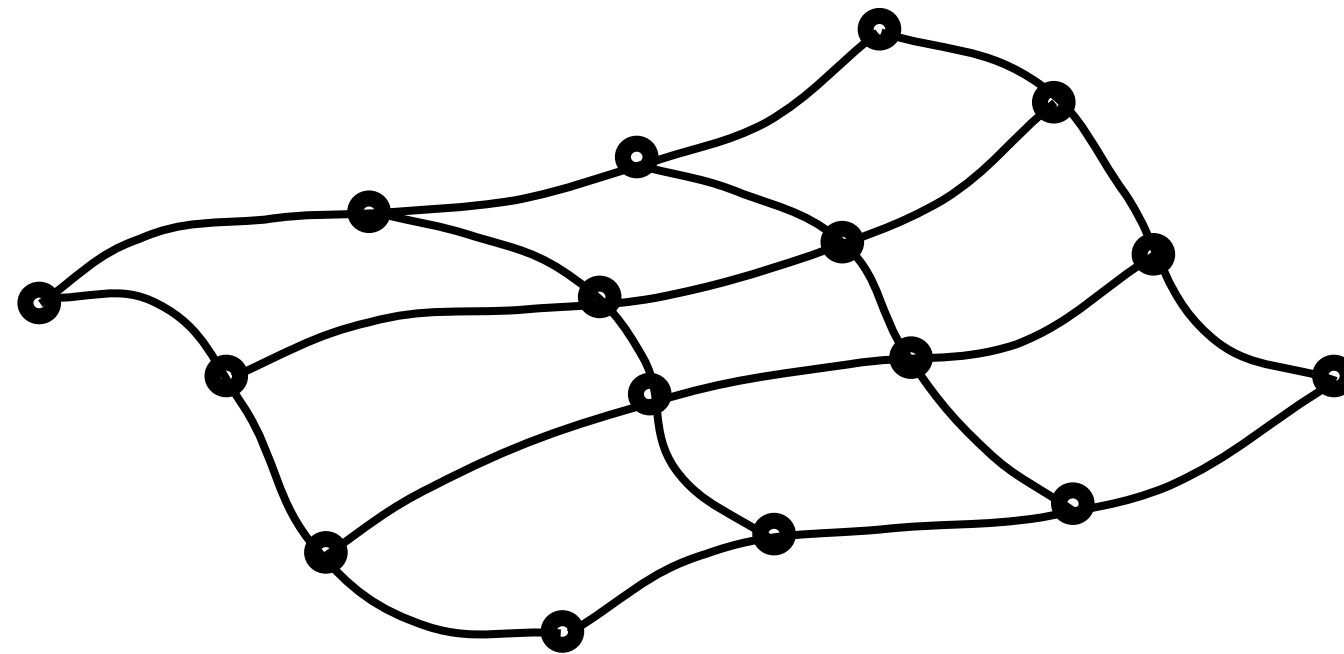
- **Defined by a sequence of points on the curve**
- **Always G_1/C_1 continuous**



Bézier surfaces

A surface is built from a set of Bézier patches

A Bézier patch consists of 16 control points in a 4x4 grid

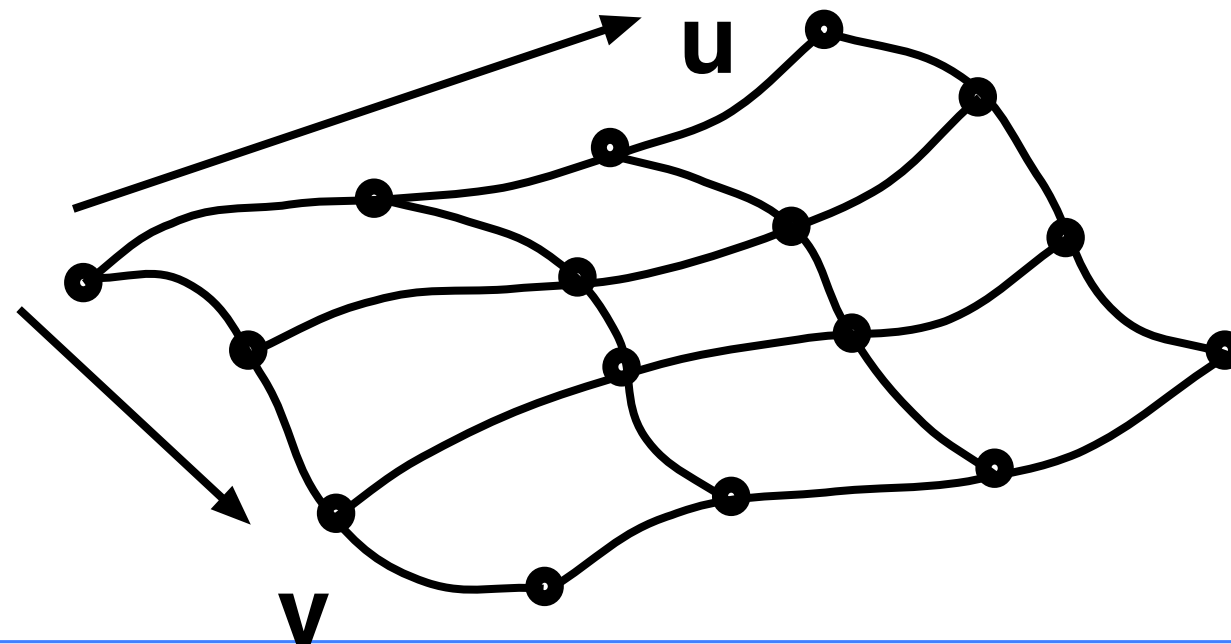




Bézier surfaces

Blending of the 16 control points as a 2-dimensional sum

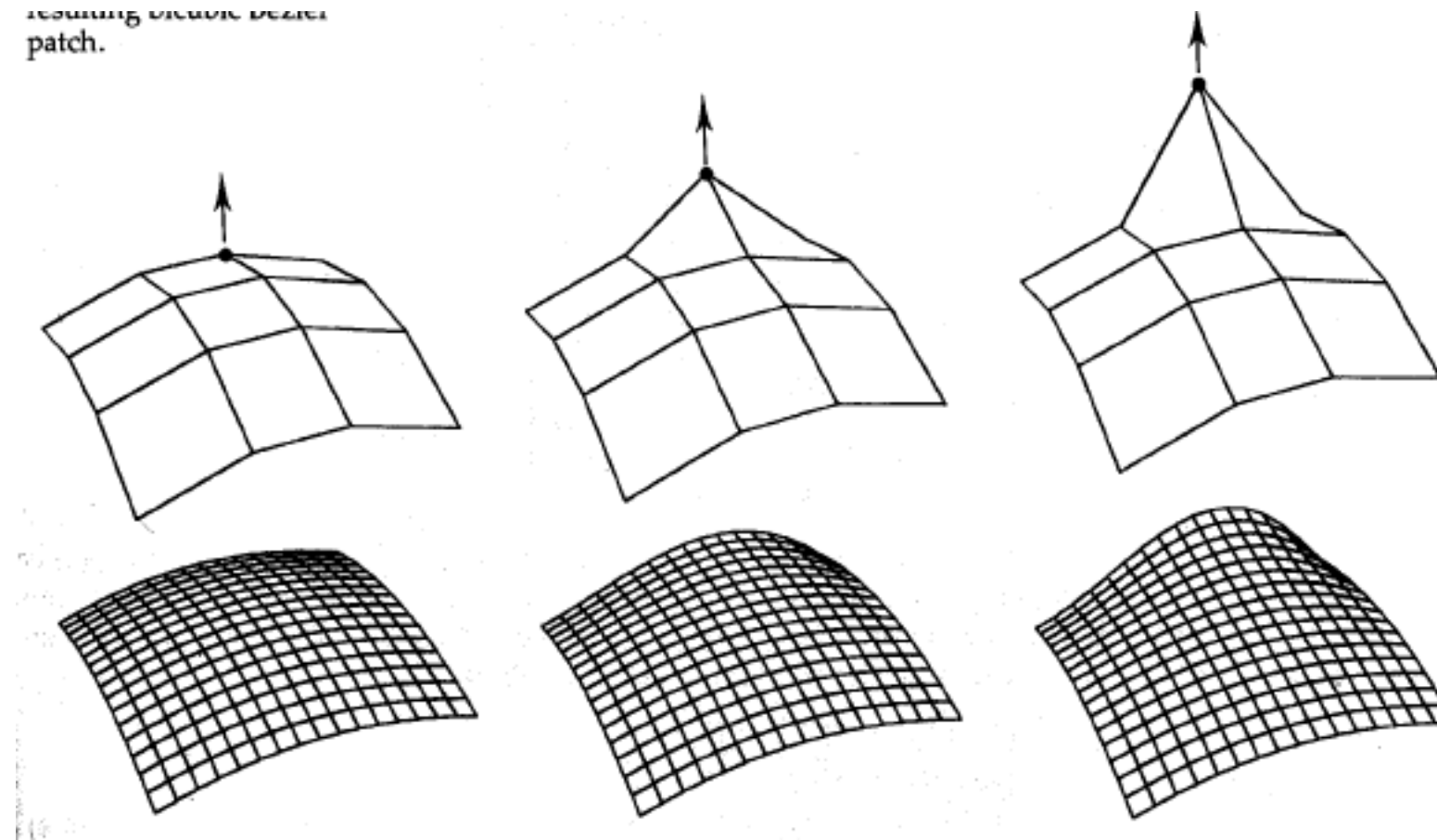
$$P(u,v) = \sum_{j=0}^3 \sum_{k=0}^3 p_{j,k} \text{BEZ}_{j,3}(v) \text{BEZ}_{k,3}(u)$$





Bézier surface example

resulting bicubic Bézier patch.

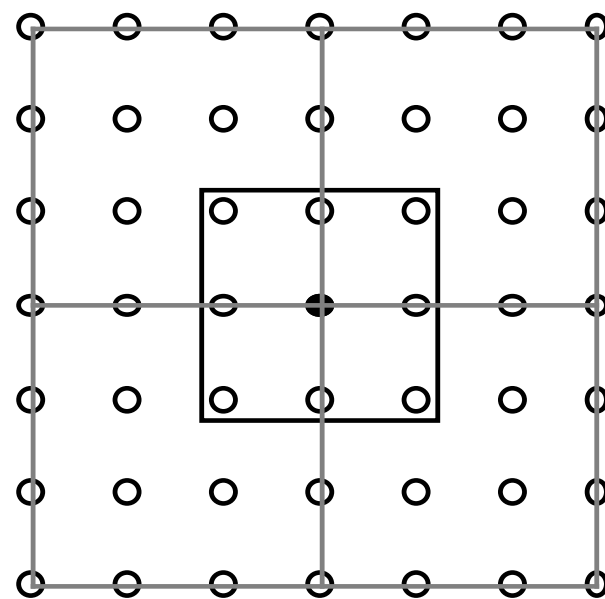




Fitting together patches

Fit in both u and v direction

Make a 3x3 “joystick” at each corner





Splines and surfaces in OpenGL

Pre-generated shapes on CPU

Generate by multi-pass GPU processing

Old OpenGL: Evaluators (glMap)

3.2: Geometry shaders

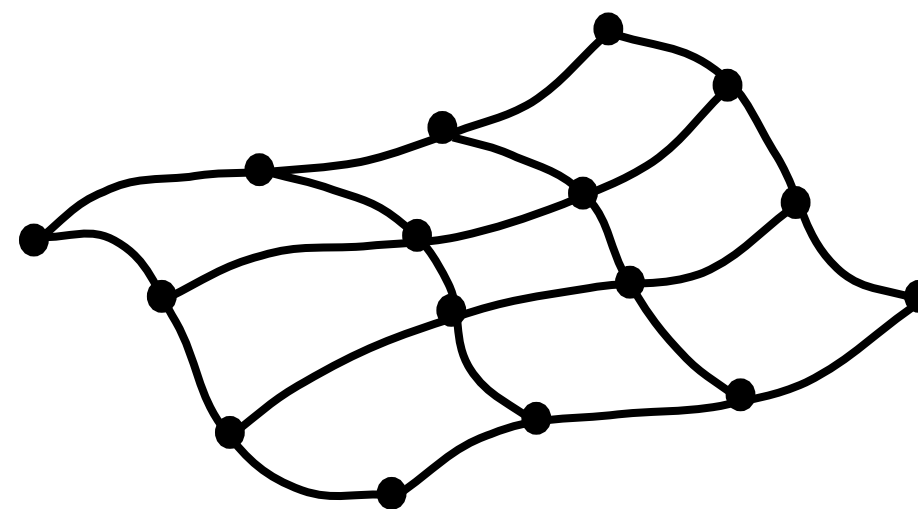
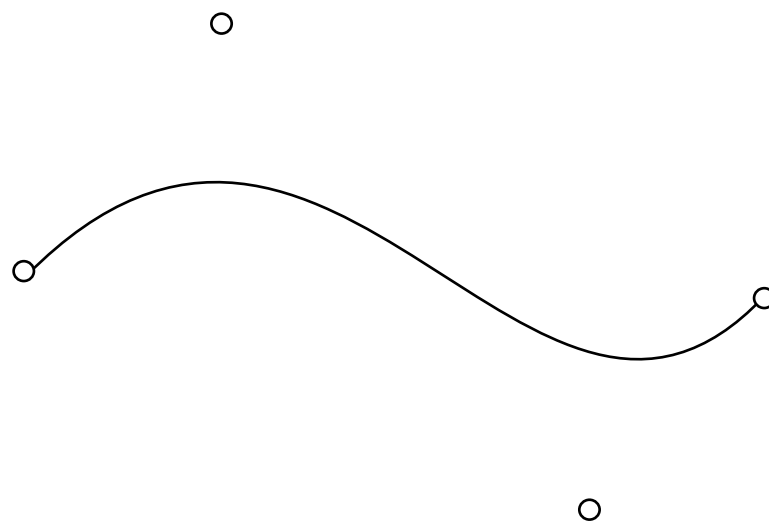
4: Tessellation shaders



Evaluators

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

Supported both curves and surfaces





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+)



Information Coding / Computer Graphics, ISY, LiTH

Tessellation shaders

OpenGL 4

Shader between vertex and fragment (before geometry shader), focused on subdivision

Higher hardware demand! (Southfork is OK)



Applications:

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



Information Coding / Computer Graphics, ISY, LiTH

More object representation soon...

Fractals and procedural generation

but first

let us continue on with the animation subject!