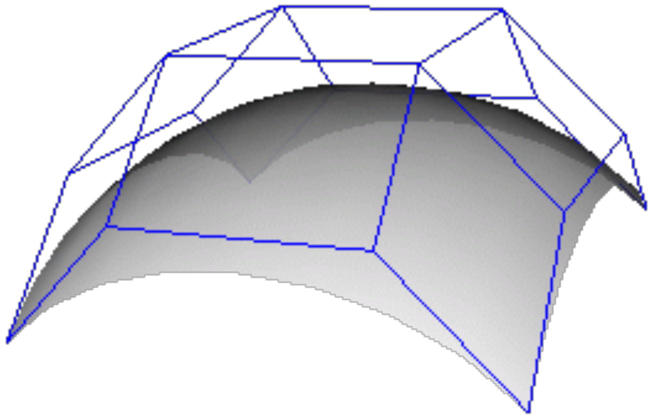
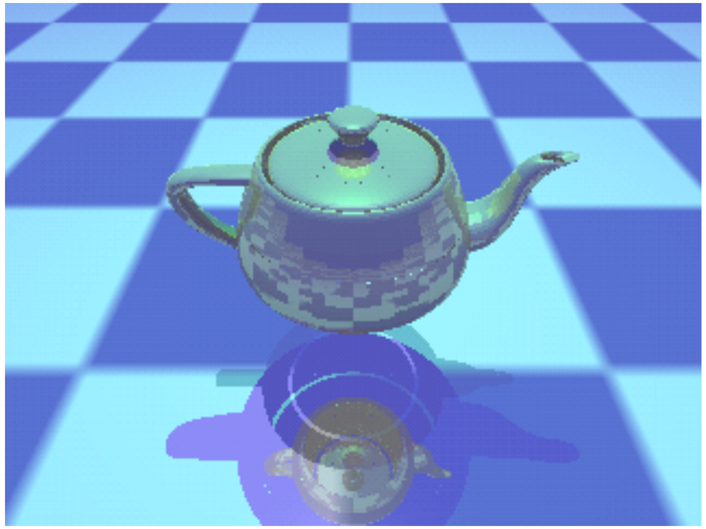
Surfaces



http://www.ibiblio.org/e-notes/Splines/fig/surf2.gif

Bezier Surfaces Introduction

- Constructing a surface relies very much on the ideas behind constructing curves
- Surfaces can be thought of as 'Bezier curves in all directions' across the surface
- *Tensor products* of Bezier curves
- Teapot most famous example
  - produced entirely by Bezier surfaces



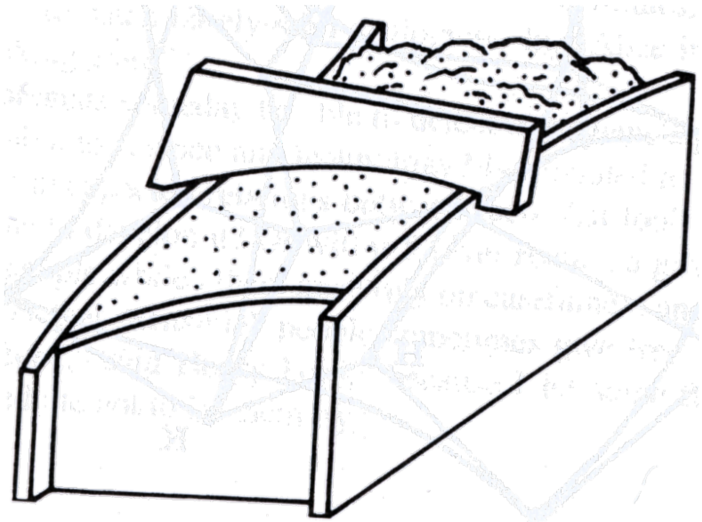
http://cg.cs.ucl.ac.uk/playgroundGL.html

Tensor Product

- Of two vectors:

$$\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \otimes \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix} = \begin{bmatrix} a_1 b_1 & a_2 b_1 & a_3 b_1 \\ a_1 b_2 & a_2 b_2 & a_3 b_2 \\ a_1 b_3 & a_2 b_3 & a_3 b_3 \\ a_1 b_4 & a_2 b_4 & a_3 b_4 \end{bmatrix}$$

- Similarly, we can define a surface as the tensor product of two curves....



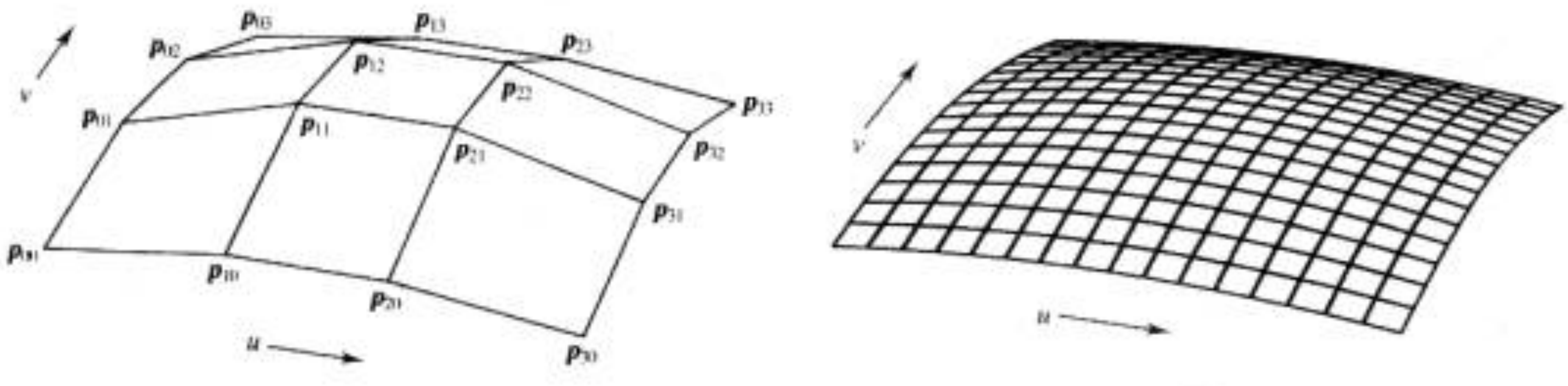
Farin, Curves and Surfaces for Computer Aided Geometric Design

Bicubic Bezier Patch

Notation: $\mathbf{CB}(P_1, P_2, P_3, P_4, \alpha)$ is Bézier curve with control points P_i evaluated at α

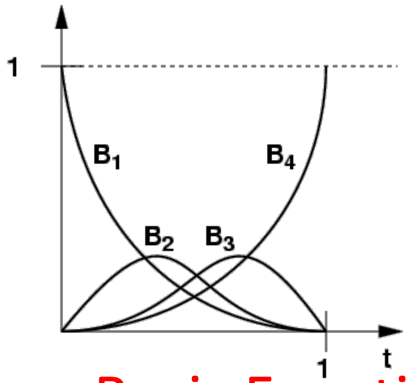
Define "Tensor-product" Bézier surface

$$
\begin{aligned}
Q(s,t) = \mathbf{CB}(\quad & \mathbf{CB}(P_{00}, P_{01}, P_{02}, P_{03}, t), \\
& \mathbf{CB}(P_{10}, P_{11}, P_{12}, P_{13}, t), \\
& \mathbf{CB}(P_{20}, P_{21}, P_{22}, P_{23}, t), \\
& \mathbf{CB}(P_{30}, P_{31}, P_{32}, P_{33}, t), \\
& s)
\end{aligned}
$$

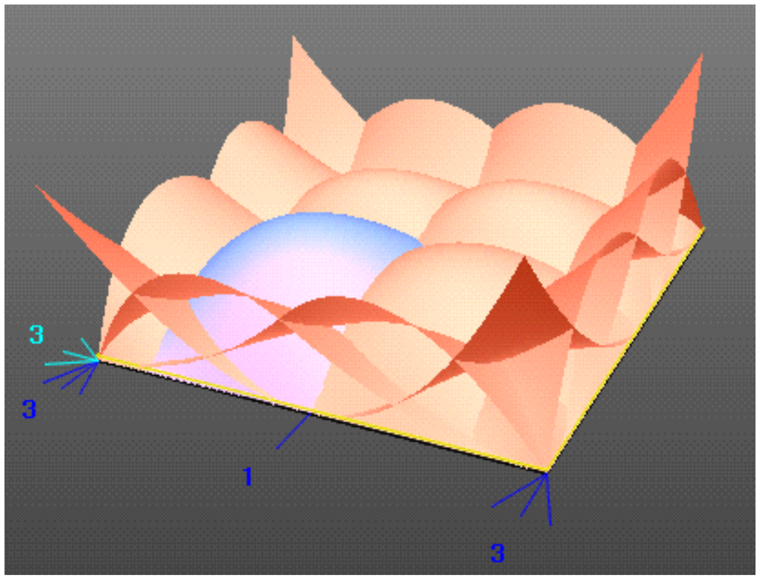


(a) (b)

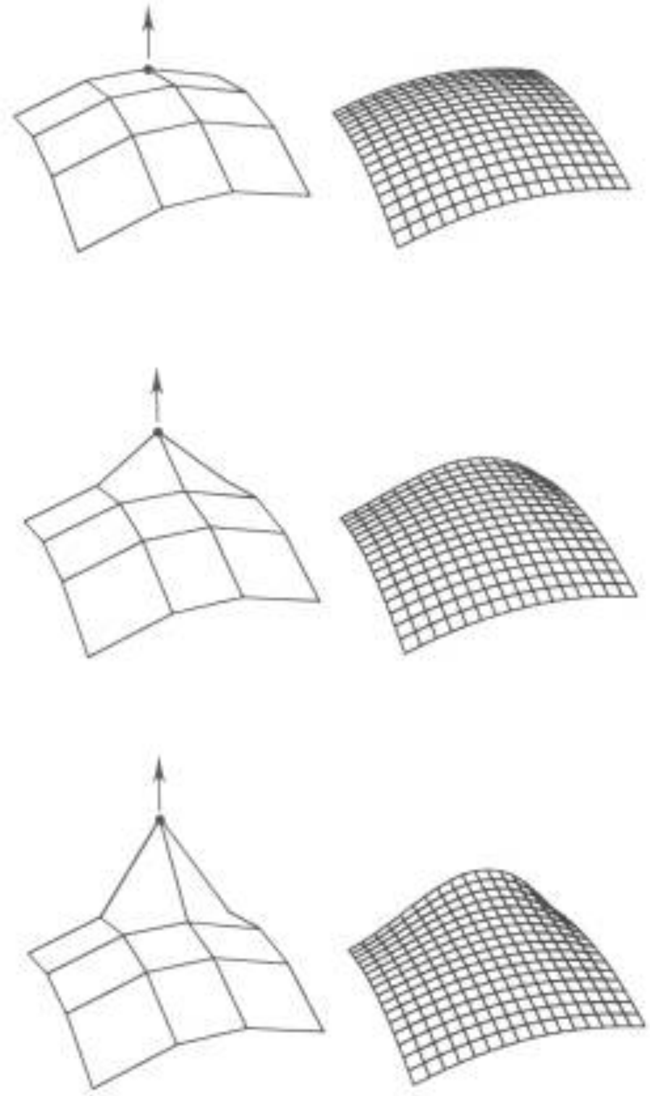
Editing Bicubic Bezier Patches



Curve Basis Functions



Surface Basis Functions



Control Points

- Consider the (m+1)*(n+1) array of 3D control points
- This array can be used to define a Bezier of surface of degree m and n.
- If m=n=3 this is called 'bi-cubic'.
- The same relation between surface and control points holds as in curves
  - If the points are on a plane the surface is a plane
  - If the edges are straight the Bezier surface edges are straight
  - The entire surface lies inside the convex hull of the control points.

$$\begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0n} \\ p_{10} & p_{11} & \cdots & p_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m0} & p_{m1} & \cdots & p_{mn} \end{bmatrix}$$

Subdivision 4*4 Cubic Case

pppp
pppp
pppp
pppp

split along rows

pppmqqq
pppmqqq
pppmqqq
pppmqqq

split down columns

pppmqqq
pppmqqq
pppmqqq
pppmqqq
pppmqqq
pppmqqq
pppmqqq

This gives 4 sets of 4*4 arrays of control points. In each case the middle values are shared by the two adjacent sets.

Rendering – de Casteljau

- Use de Casteljau to subdivide each row.
- Then use de Casteljau to subdivide each of the 7 resulting columns.
- This will result in 4 sets of (m+1)*(n+1) array with one common row and one common column.
- If all points are on a plane and the edges are straight line then we get a polygon with 4 vertices.
- So the recursive algorithm is as follows:

Rendering 3D de Casteljau

```
typedef struct{
        float x,y,z;
}Point3D;

typedef Point3D ControlPointArray[4][4];

void Bezier3D(ControlPointArray p) {
        ControlPointArray q,r,s,t;
        if(Coplanar(p)) RenderPolygon(p[0][0],p[3][0],p[3][3],p[0][3]);
        else{
                /*split p into q,r,s,t*/
                Split3D(p,q,r,s,t);
                Bezier3D(q);
                Bezier3D(r);
                Bezier3D(s);
                Bezier3D(t);
        }
}
```

Testing Colinearity

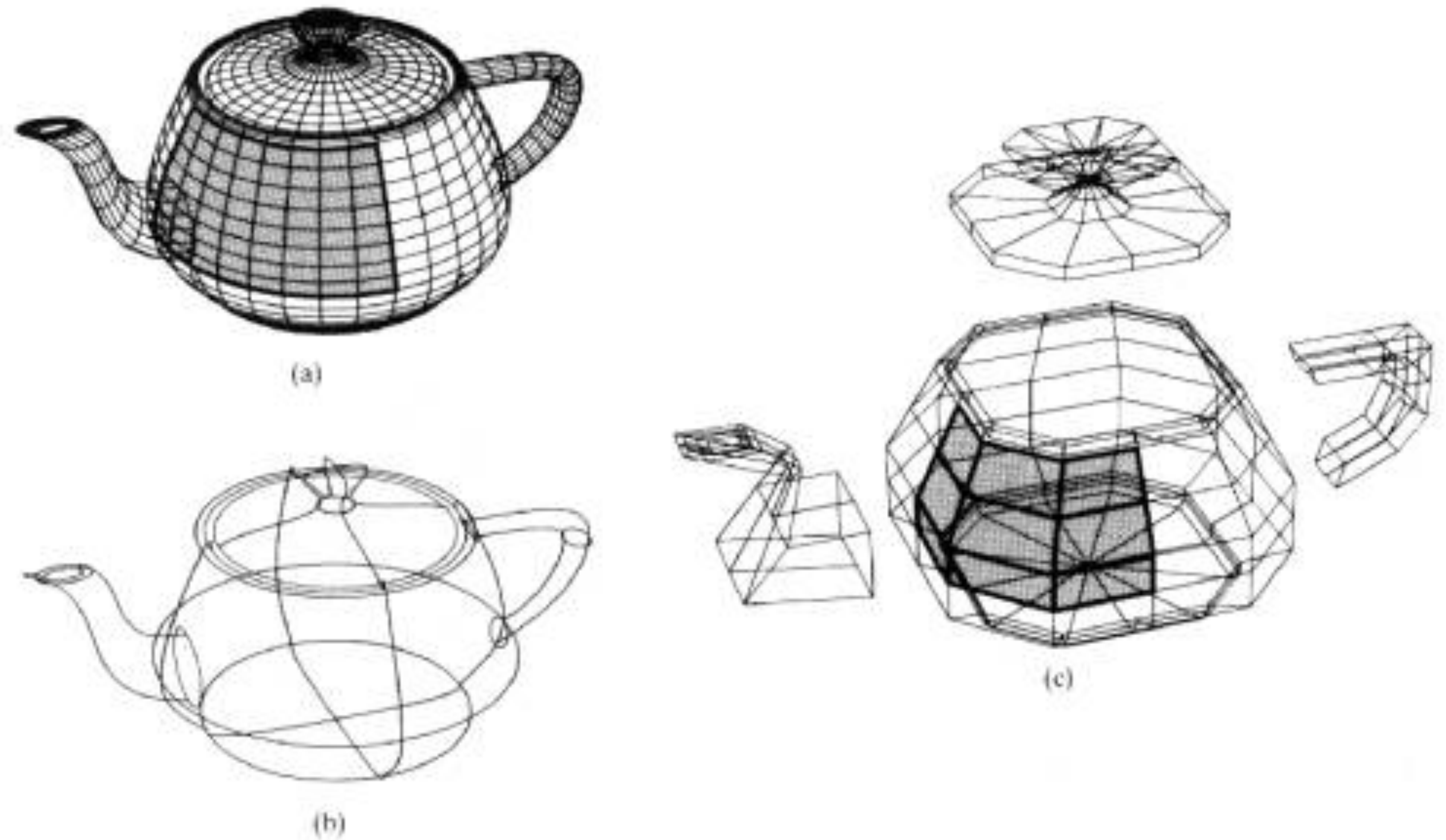
- This is where the computational work of the algorithm is located.
- If the equation of the plane is ax+by+cz=d and (x,y,z) is a point then its distance from the plane is:

$$D^2 = \frac{(ax + by + cz - d)^2}{a^2 + b^2 + c^2}$$

- So we have an analogy to the curve case, except also we should check that the edges are straight, and that adjacent regions have been split to the same level.
- A simple approach is to just run the recursion to the same level irrespective of testing whether the final pieces are really flat.

Modeling with Bicubic Bezier Patches

- Original Teapot specified with Bezier Patches



(a) (b) (c)

Alternative Splines Surfaces

- You can make surfaces from B-Splines in a similar way
- A particular types of B-Spline generalisation, Non-uniform rational Basis spline (NURBS) surfaces are particularly common

Catmull-Clark Subdivision Surfaces

- A *constructive* way to get a smooth surface from a cage
  - Strongly related to interpolation/subdivision schemes used in recursive curve drawing (c.g. De Castlejau construction)

- In a recursive scheme each face is replaced by a new set of faces defined by interpolating the points/faces

Catmull-Clark Subdivision Surfaces

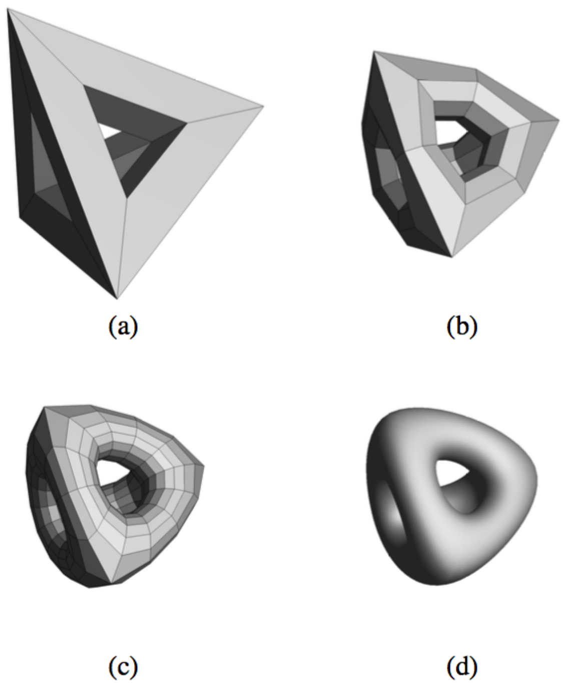
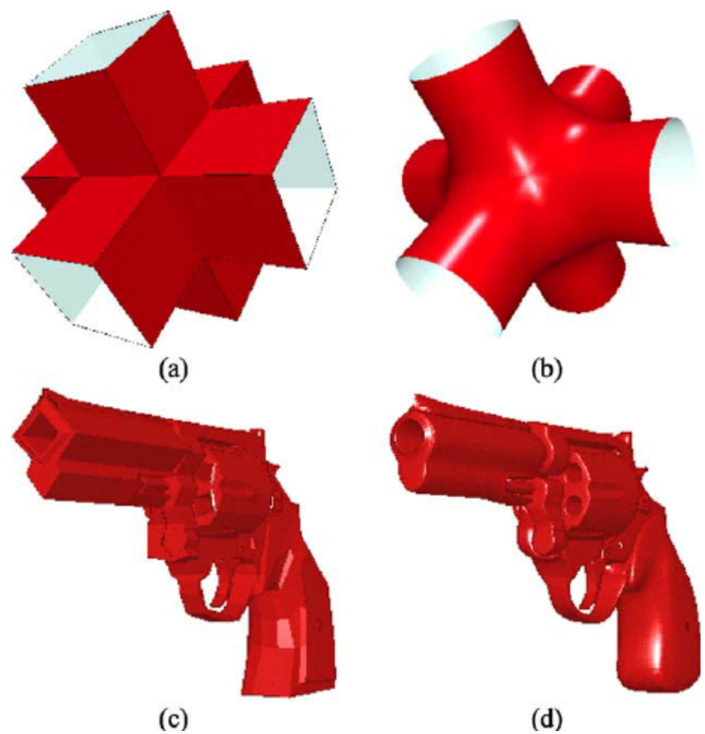


Figure 3: Recursive subdivision of a topologically complicated mesh: (a) the control mesh; (b) after one subdivision step; (c) after two subdivision steps; (d) the limit surface.



Conclusions

- Surfaces are a simple extension to curves
- Really just a tensor-product between two curves
  - One curve gets extruded along the other
- Subdivision surfaces are another way of generating curves
  - Particularly amenable to GPU implementation!