Interactive Computer Graphics

Lecture 14

Introduction to Surface Construction

Non Parametric Surface

Surfaces can be constructed from Cartesian equations directly, and this is acceptable for specific applications.

As before the polynomial surface can be used.

Non Parametric Polynomial Surface

$$\begin{pmatrix} x & y & z & 1 \end{pmatrix} \begin{pmatrix} a & b & c & d \\ b & e & f & g \\ c & f & h & i \\ d & g & i & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = 0$$

The matrix formulation is a good way of expressing polynomial equations.

Because of the symmetry it is really just upper triangular (there are 9 unknowns)

Multiplying out

$ax^2+ey^2+hz^2+2bxy+2cxz+2fyz+2dx+2gy+2iz+1 = 0$

We have 9 unknowns, so we need to be able to supply 9 points from which we obtain 9 equations and can solve for coefficients [a..i]

As Before

This formulation suffers the same problems as the non-parametric spline curve. It is a fixed surface for a given set of knots.

It has no flexibility for design of surfaces, though it could be used for particular problems.

Simple Parametric surfaces

We can extend the formulation to simple parametric surfaces

$$P(\mu,\nu) = \begin{pmatrix} \mu & \nu & 1 \end{pmatrix} \begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \\ \mathbf{b} & \mathbf{d} & \mathbf{e} \\ \mathbf{c} & \mathbf{e} & \mathbf{f} \end{pmatrix} \begin{pmatrix} \mu \\ \nu \\ 1 \end{pmatrix}$$

Or if multiplied out

$\mathbf{P}(\mu,\nu) = \mathbf{a}\mu^2 + 2\mathbf{b}\mu\nu + 2\mathbf{c}\mu + \mathbf{d}\nu^2 + 2\mathbf{e}\nu + \mathbf{f}$

We have six vector unknowns in this equation, so we need six points to create a surface.

This time we have two parameters, and as before we will restrict them to the range [0..1]

Surface Edges

$\mathbf{P}(\mu,\nu) = \mathbf{a}\mu^2 + 2\mathbf{b}\mu\nu + 2\mathbf{c}\mu + \mathbf{d}\nu^2 + 2\mathbf{e}\nu + \mathbf{f}$

The boundary of the surface is given by the four curves where:

$\mu=0 \quad \mathbf{P}(0,\nu) = \mathbf{d}\nu^2 + 2\mathbf{e}\nu + \mathbf{f}$

$\mu=1 \quad \mathbf{P}(1,\nu) = \mathbf{d}\nu^2 + 2(\mathbf{e}+\mathbf{b})\nu + \mathbf{f} + \mathbf{a} + 2\mathbf{c}$

$\nu=0 \quad \mathbf{P}(\mu,0) = \mathbf{a}\mu^2 + 2\mathbf{c}\mu + \mathbf{f}$

$\nu=1 \quad \mathbf{P}(\mu,1) = \mathbf{a}\mu^2 + 2(\mathbf{b}+\mathbf{c})\mu + \mathbf{d} + 2\mathbf{e} + \mathbf{f}$

Associating points and parameters

We can solve for the unknowns by substituting in six points at known values of ν and μ. We might have an arrangement such as:

	μ	ν
$\mathbf{P_0}$	0	0
$\mathbf{P_1}$	0	1
$\mathbf{P_2}$	1	0
$\mathbf{P_3}$	1	1
$\mathbf{P_4}$	1/2	0
$\mathbf{P_5}$	1/2	1

Surface equations

This data then gives us six equations in the unknowns which we can solve using standard methods.

$\mathbf{P_0} = \mathbf{f}$

$\mathbf{P_1} = \mathbf{d} + 2\mathbf{e} + \mathbf{f}$

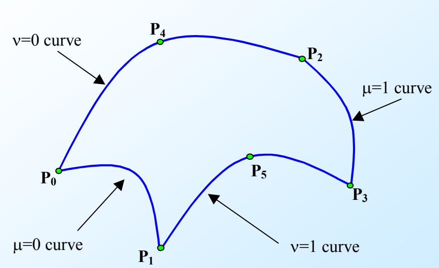
$\mathbf{P_2} = \mathbf{a} + 2\mathbf{c} + \mathbf{f}$

$\mathbf{P_3} = \mathbf{a} + 2\mathbf{b} + 2\mathbf{c} + \mathbf{d} + 2\mathbf{e} + \mathbf{f}$

$\mathbf{P_4} = \mathbf{a}/4 + \mathbf{c} + \mathbf{f}$

$\mathbf{P_5} = \mathbf{a}/4 + \mathbf{b} + \mathbf{c} + \mathbf{d} + 2\mathbf{e} + \mathbf{f}$

The resulting surface



The boundary equations

The boundaries are all second order curves and so will be nice and smooth.

There is quite a lot of flexibility in this formulation, but it is still only suitable for simple surfaces.

A higher order tensor product

$$\mathbf{P}(\nu,\mu) = \begin{pmatrix} \mu^3 & \mu^2 & \mu & 1 \end{pmatrix} \begin{pmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} \\ \mathbf{b} & \mathbf{e} & \mathbf{f} & \mathbf{g} \\ \mathbf{c} & \mathbf{f} & \mathbf{h} & \mathbf{i} \\ \mathbf{d} & \mathbf{g} & \mathbf{i} & \mathbf{j} \end{pmatrix} \begin{pmatrix} \nu^3 \\ \nu^2 \\ \nu \\ 1 \end{pmatrix}$$

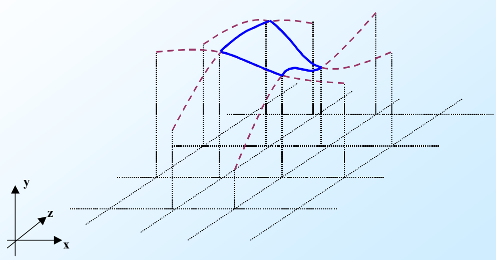
Using higher orders gives more variety in shape and better control, but the method is hard to apply and generalise

Cubic Spline Patches

The patch method is generally effective in creating more complex surfaces.

The idea is, as in the case of the curves, to create a surface by joining a lot of simple surfaces continuously.

Cartesian formulation

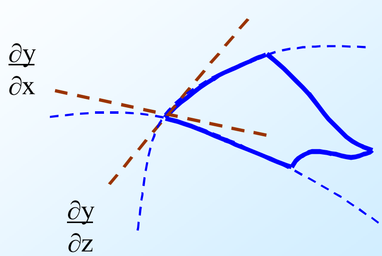


Points and Gradients

At each corner of the patch we need to interpolate the points and set the gradients to match the adjacent patch.

There are two gradients

Patch gradients



Parametric patches

In practice we use parametric patches with two parameters μ and ν.

We need to match three values at each corner:

$$\mathbf{P}(\mu,\nu) \qquad \frac{\partial \mathbf{P}(\mu,\nu)}{\partial \mu} \qquad \frac{\partial \mathbf{P}(\mu,\nu)}{\partial \nu}$$

Corners

As usual we adopt the convention that the corners are at parameter values (0,0) (0,1) (1,0) and (1,1)

We need to ensure that the patch joins its neighbours exactly at the edges.

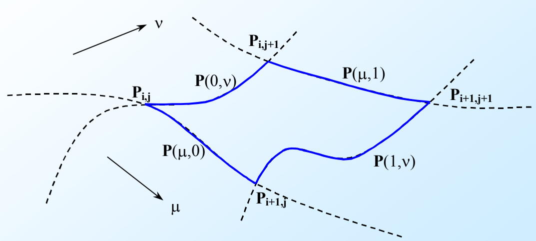
Hence we ensure that the edge contours are the same on adjacent patches

Edges

We do this by designing the edge curves in an identical manner to the cubic spline curve patch.

$\mathbf{P}(0,\nu)$ joining $\mathbf{P}_{i,j}$ to $\mathbf{P}_{i,j+1}$
$\mathbf{P}(1,\nu)$ joining $\mathbf{P}_{i+1,j}$ to $\mathbf{P}_{i+1,j+1}$
$\mathbf{P}(\mu,0)$ joining $\mathbf{P}_{i,j}$ to $\mathbf{P}_{i+1,j}$
$\mathbf{P}(\mu,1)$ joining $\mathbf{P}_{i,j+1}$ to $\mathbf{P}_{i+1,j+1}$

A spline patch



The Coon's patch

To define the internal points we linearly interpolate the edge curves:

$$\mathbf{P}(\mu,\nu) =$$
$$\mathbf{P}(\mu,0)(1-\nu) + \mathbf{P}(\mu,1)\nu + \mathbf{P}(0,\nu)(1-\mu) + \mathbf{P}(1,\nu)\mu - \mathbf{P}(0,0)(1-\nu)(1-\mu) - \mathbf{P}(0,1)\nu(1-\mu) - \mathbf{P}(1,0)(1-\nu)\mu - \mathbf{P}(1,1)\nu\mu$$

Polygonisation

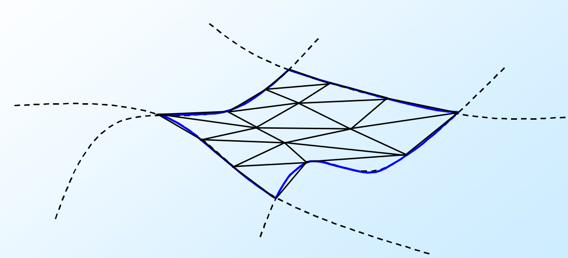
To draw a spline patch we can simply transform it into polygons.

We select a grid of points, eg:

$\mu = 0.0, 0.1, 0.2, 0.3, \ldots\ 1.0$
$\nu = 0.0, 0.1, 0.2, 0.3, \ldots\ 1.0$

and triangulate to that grid

Polygonisation of a patch



Choosing the polygon size

For speed we can use large polygons with Gouraud or Phong shading.

For accuracy we use small polygons, chosen to match the pixel size.

Lofting

Surfaces can also be drawn by a technique called lofting.

This means drawing contours of constant μ and of constant ν

Algorithms for eliminating the hidden parts have been devised

Ray tracing a Patch

The patch equation is fourth order

$$\mathbf{P}(\mu,\nu) = \mathbf{P}(\mu,0)(1-\nu) + \mathbf{P}(\mu,1)\nu + \mathbf{P}(0,\nu)(1-\mu) + \mathbf{P}(1,\nu)\mu - \mathbf{P}(0,0)(1-\nu)(1-\mu) - \mathbf{P}(0,1)\nu(1-\mu) - \mathbf{P}(1,0)(1-\nu)\mu - \mathbf{P}(1,1)\nu\mu$$

Hence no closed form solution exists for a ray patch intersection.

There are various numeric algorithms.

Numerical Ray-Patch algorithm

1. Polygonise the patch at a low resolution (say 4*4)

2. Calculate the ray intersection with the 32 triangles and find the nearest intersection.

3. Polygonise the immediate area of the insection and calculate a better estimate of the intersection

4. Continue until the best estimate is found

Multiple Roots

This algorithm will find a root, but it is not guaranteed to find the nearest root.

However, it the object is relatively smooth it should work well in all cases.

Note that it will be necessary to do a ray intersection with each patch of the object to find the nearest intersection.

Example of Using a Coon's Patch

Part of a terrain map defined on a regular x,y grid is as follows:

x,μ \ y,ν	3	4	5	6
8		10	9	
9	14	12	11	10
10	15	13	14	10
11		10	11	

Find the Coons patch on the centre four points

Corners

The corners are defined directly in the question:

P(0,0) = (9,4,12)
P(0,1) = (9,5,11)
P(1,0) = (10,4,13)
P(1,1) = (10,5,14)

Gradients in the x (μ) direction

∂P/dμ (at P(0,0)) = ((10,4,13) - (8,4,10))/2 = (1,0,1.5)

∂P/dμ (at P(1,0)) = ((11,4,10) - (9,4,12))/2 = (1,0,-1)

∂P/dμ (at P(0,1)) = ((10,5,14) - (8,5,9))/2 = (1,0,2.5)

∂P/dμ (at P(1,1)) = ((11,5,11) - (9,5,11))/2 = (1,0,0)

Gradients in the y (ν) direction

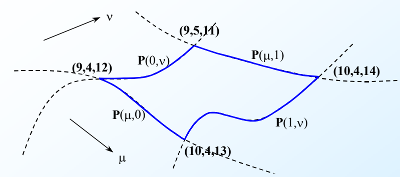
∂P/dν (at P(0,0)) = ((9,5,11) - (9,3,14))/2 = (0,1,-1.5)

∂P/dν (at P(1,0)) = ((9,6,10) - (9,4,12))/2 = (0,1,-1)

∂P/dν (at P(0,1)) = ((10,5,14) - (10,3,15))/2 = (0,1,-0.5)

∂P/dν (at P(1,1)) = ((10,6,10) - (10,4,13))/2 = (0,1,-1.5)

Finding the boundary curves



$$P(\mu,0) = \mathbf{a_3}\mu^3 + \mathbf{a_2}\mu^2 + \mathbf{a_1}\mu + \mathbf{a_0}$$

$$\begin{bmatrix} \mathbf{a_0} \\ \mathbf{a_1} \\ \mathbf{a_2} \\ \mathbf{a_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix} \begin{pmatrix} (9,4,12) \\ (1,0,1.5) \\ (10,4,13) \\ (1,0,-1) \end{pmatrix}$$

Solving for the constants $\mathbf{a_0}$ - $\mathbf{a_3}$

$\mathbf{a_0}$ = (9,4,12)
$\mathbf{a_1}$ = (1,0,1.5)
$\mathbf{a_2}$ = -3*(9,4,12) - 2*(1,0,1.5) + 3*(10,4,13) - (1,0,1)
= (0,0,1)
$\mathbf{a_3}$ = 2*(9,4,12) + (1,0,1.5) - 2*(10,4,13) + (1,0,1)
= (0,0,0.5)

Solving for the curves P(μ,1), P(0,ν) and P(1,ν)

These curves are found identically to P(μ,0).

We now have all the individual bits:

- P(μ,0) cubic polynomial in μ
- P(μ,1) cubic polynomial in μ
- P(0,ν) cubic polynomial in ν
- P(1,ν) cubic polynomial in ν
- P(0,0), P(0,1), P(1,0), P(1,1)

Given μ and ν we can evaluate each of these eight points

So, for any given value for μ and ν

we can evaluate the coordinate on the Coon's patch:

$$\mathbf{P}(\mu,\nu) =$$
$$\mathbf{P}(\mu,0)(1-\nu) + \mathbf{P}(\mu,1)\nu + \mathbf{P}(0,\nu)(1-\mu) + \mathbf{P}(1,\nu)\mu - \mathbf{P}(0,0)(1-\nu)(1-\mu) - \mathbf{P}(0,1)\nu(1-\mu) - \mathbf{P}(1,0)(1-\nu)\mu - \mathbf{P}(1,1)\nu\mu$$