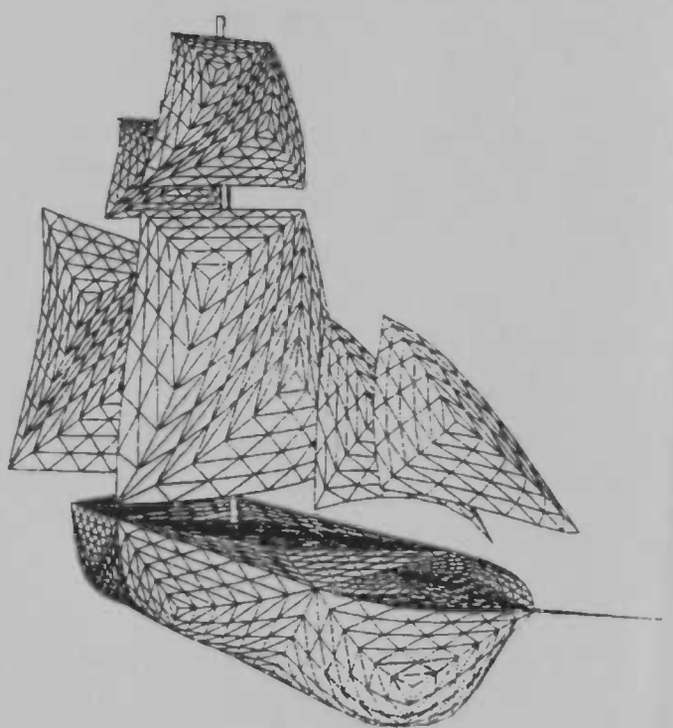
ngen Faculteit der Wiskunde en Natuurwetenschappen

**Vakgroep
Informatica**

Interpolation of Curves and Surfaces



W.J.D. Lamberts

begeleider: Dr. G. Vegter

11 februari 1997

Contents

1 Introduction

In this paper we will present two interpolation schemes. The first interpolation scheme is a 2-D interpolation scheme. This scheme was proposed by *G. Farin* in [1, §8.2]. The 2-D scheme will interpolate points on a cubic curve which is defined by two data points and tangential data in these points. The second scheme is a 3-D interpolation scheme.
This scheme, which was proposed by *Bruce R. Piper* in [3], interpolates points on a surface, which is defined by three data points forming a triangle $\mathcal{P}$, tangential data in these points and, if present, information of adjacent triangles. The information of adjacent triangles (triangles that share a common edge with triangle $\mathcal{P}$) is used to produce a surface over these triangles that is C^1-continuous.
Both interpolation schemes will use the *de Casteljau* algorithm to interpolate points. This algorithm was developed by P. de Casteljau who worked for Citroën and by P. Bézier who worked for Renault. Although de Casteljau developed this method earlier than Bézier did, the whole theory of polynomial curve and surface interpolation bears Béziers name, because the work of de Casteljau was never published. First only rectangular patches were used for interpolation, but soon people realized the need for triangular patches. An advantage of triangular patches was that it allowed scientists to describe complex surfaces, like those used in the interior of cars. The method is also used for fitting scattered data on surfaces, and for interactive design of curves and surfaces. Some examples of surface interpolation are depicted below.

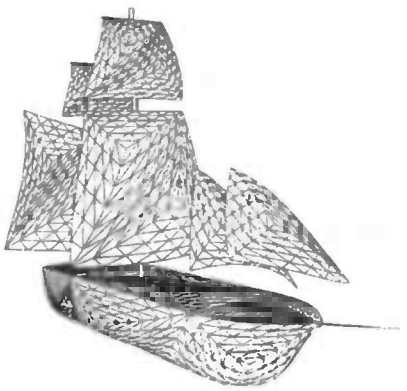


Figure 1: A control net of a simple ship from [7].

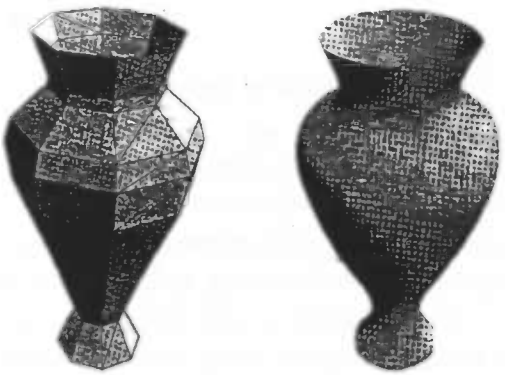


Figure 2: A control net of a vase and its interpolated surface from [8].

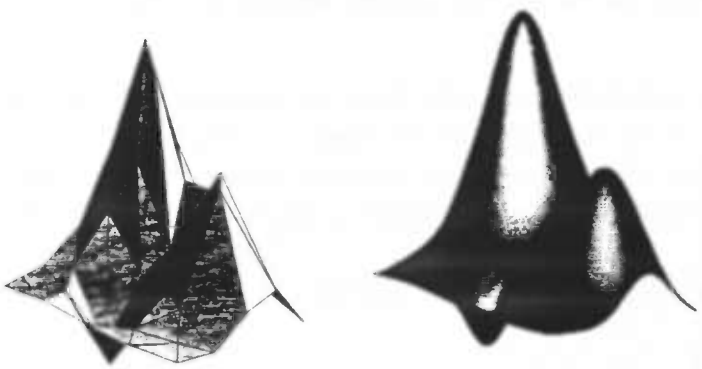


Figure 3: A mathematical surface and its control net from [6].

However, the described interpolation algorithms take as input the data points with the associated tangential data, and, in addition to these, a collection of *control points*. The freedom in the choice of these control points is exploited in order to ensure C^1-continuity of adjacent curve or surface patches. Together with the data points, the control points will uniquely define the curve or surface.

2 2-D interpolation

In this section the following problem is solved.
Suppose we have data points $\mathbf{p}_0, \ldots, \mathbf{p}_n$ in the plane and tangent vectors in these points $\mathbf{v}_0, \ldots, \mathbf{v}_n$. These tangent vectors are of unit length, i.e. they have length one. The data points are ordered in such a way that $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ are adjacent points on a C^1 piecewise cubic polynomial.
We now wish to find a curve that passes through the given data points and is tangent to the given tangential data in these points.
In this article we assume that the given tangent vectors are all of length 1.
We will now give a definition of interpolation as we will use it.

> Given $n+1$ distinct points $\mathbf{x}_0, \ldots, \mathbf{x}_n$ with $\mathbf{x}_i \in \mathbb{R}^2$; $i = 0, \ldots, n$, find a polynomial $p(t) \in \mathbf{R}^2$ with $t \in \mathbb{R}$ so that $p_0(t_0) = \mathbf{x}_0, \ldots, p(t_n) = \mathbf{x}_n$. This polynomial $p(t)$ is unique among the set of all polynomials of degree at most n.

To solve this problem two methods are used which are described by G. Farin in [1, pages 113–116, §8.2] and [1, pages 29–30, §3.2]. In §8.2 a method is described for finding Bézier points between each pair of data points. These points are used for calculating a single point on the curve, using the *de Casteljau* Algorithm as described in §3.2 .
In order to apply these methods on the data points, the following constraints apply to the data points:

1. Two consecutive data points are not allowed to coincide.
2. The tangent vector must be a non-zero vector.

Between each pair of successive data points $(\mathbf{p}_i, \mathbf{p}_{i+1})$ two *Bézier* points are calculated. The following formula is used to calculate these points:

$$\begin{aligned} \mathbf{t}_1 &= \mathbf{p}_i + \alpha \mathbf{v}_i \\ \mathbf{t}_2 &= \mathbf{p}_{i+1} - \beta \mathbf{v}_{i+1} \end{aligned} \tag{1}$$

where $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$ are the tangent vectors in respectively $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$.
Since $\mathbf{v}_i$, $\mathbf{v}_{i+1}$, $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ are given, we need to find values for α and β to get the two *Bézier* points. Farin describes several different ways for finding α and β. We will use the following:

$$\alpha = \frac{1}{3cos\Theta}\|\Delta \mathbf{p}_i\|$$

$$\beta = \frac{1}{3cos\Psi}\|\Delta \mathbf{p}_i\| \quad \text{with } \Delta \mathbf{p}_i = \mathbf{p}_{i+1} - \mathbf{p}_i$$

In this formula Θ equals the angle between $\Delta \mathbf{p}_i$ and $\mathbf{v}_i$. Ψ equals the angle between $\Delta \mathbf{p}_i$ and $\mathbf{v}_{i+1}$. We can interpret the curve as the graph of a function of t (see Figure 4), where the parameter t varies along the straight line through $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. The x-axis is the line through $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ and the y-axis is the line perpendicular to $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$.
If we look at figure 4 we see that we want the projections of the points t_1 and t_2 on the line through $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ to divide the line through $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ in three equal segments. This means that the cosine of Θ is equal to

$$\cos\Theta = \frac{\frac{1}{3}\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}{\|\mathbf{t}_1 - \mathbf{p}_i\|}$$

Since we want to determine t_1, we will rewrite the above formula. This gives us

$$\|\mathbf{t}_1 - \mathbf{p}_i\| = \frac{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}{3\cos\Theta} \tag{2}$$

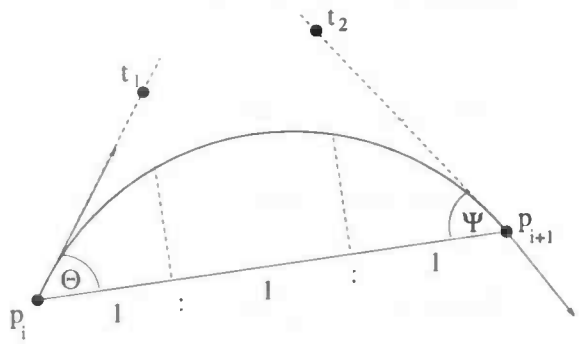


Figure 4: A 2-D example for calculating Θ and Ψ

However, if one of the angles Θ and Ψ is 90°, this formula cannot be used, because expression (2) is undefined. Farin makes a case distinction for angles smaller than 60° and angles larger than 60°.

$$\alpha = \begin{cases} \frac{1}{3\cos\Theta}\|\Delta\mathbf{p}_i\| & \text{if } |\Theta| \leq 60° \\ \frac{2}{3}\|\Delta\mathbf{p}_i\| & \text{otherwise} \end{cases}$$

$$\beta = \begin{cases} \frac{1}{3\cos\Psi}\|\Delta\mathbf{p}_i\| & \text{if } |\Psi| \leq 60° \\ \frac{2}{3}\|\Delta\mathbf{p}_i\| & \text{otherwise} \end{cases}$$

We can rewrite this formulas for α and β by using the cosine rule:

$$\cos\Theta = \frac{\Delta\mathbf{p}_i \cdot \mathbf{v}_i}{\|\Delta\mathbf{p}_i\|\|\mathbf{v}_i\|}$$

$$\cos\Psi = \frac{\Delta\mathbf{p}_i \cdot \mathbf{v}_{i+1}}{\|\Delta\mathbf{p}_i\|\|\mathbf{v}_{i+1}\|}$$

Substituting the cosine rule in the expressions we have for α and β we get (remember that the tangent vectors are of unit length):

$$\alpha = \begin{cases} \frac{\|\Delta\mathbf{p}_i\|^2}{3\mathbf{v}_i \cdot \Delta\mathbf{p}_i} & \text{if } |\Theta| \leq 60° \\ \frac{2}{3}\|\Delta\mathbf{p}_i\| & \text{otherwise} \end{cases} \tag{3}$$

$$\beta = \begin{cases} \frac{\|\Delta\mathbf{p}_i\|^2}{3\mathbf{v}_{i+1} \cdot \Delta\mathbf{p}_i} & \text{if } |\Psi| \leq 60° \\ \frac{2}{3}\|\Delta\mathbf{p}_i\| & \text{otherwise} \end{cases} \tag{4}$$

Now we have found the two Bézier points, $\mathbf{t}_1$ and $\mathbf{t}_2$ we can apply the *de Casteljau* algorithm. This algorithm is used to interpolate points on a m^{th} degree curve. The *de Casteljau* algorithm in [1, §3.2] is described as follows:

Given: $\mathbf{b}_0, \mathbf{b}_1, \cdots \mathbf{b}_m \in \mathbb{R}^2$ and $t \in \mathbb{R}$
Find: point $\mathbf{b}_0^m$, which is a point on a m^{th} degree curve between points $\mathbf{b}_0$ and $\mathbf{b}_m$, by using the following algorithm:

$$\mathbf{b}_{i_r}^r(t) = (1-t)\mathbf{b}_{i_r}^{r-1}(t) + t\mathbf{b}_{i_r+1}^{r-1}(t) \qquad \begin{cases} r &= 1, \ldots, m \\ i_r &= 0, \ldots, m-r \\ t &\in [0,1] \end{cases} \tag{5}$$

and $\mathbf{b}_{i_r}^0(t) = \mathbf{b}_{i_r}$.
An example of the cubic case ($m = 3$) of this algorithm, where $\mathbf{b}_0^3$ is the interpolated point on the cubic curve, is:

$$\begin{array}{llll} \mathbf{b}_0 & & & \\ \mathbf{b}_1 & \mathbf{b}_0^1 & & \\ \mathbf{b}_2 & \mathbf{b}_1^1 & \mathbf{b}_0^2 & \\ \mathbf{b}_3 & \mathbf{b}_2^1 & \mathbf{b}_1^2 & \mathbf{b}_0^3 \end{array}$$

If we look at the rightmost column of the above array we see the point we wish to interpolate. The leftmost column are the Bézier points. The points in the two center columns are the intermediate points of the de Casteljau algorithm.
A picture of this example could be:

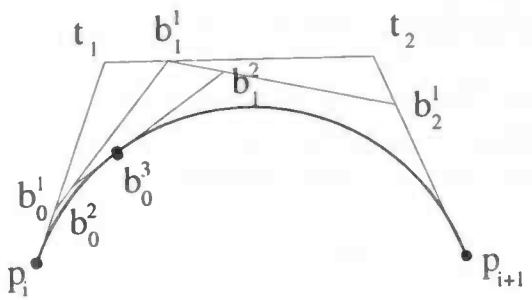


Figure 5: A 2-D example of the *de Casteljau* algorithm, with t = $\frac{1}{4}$

If we use the the data points and the Bézier points $\mathbf{t}_1$ and $\mathbf{t}_2$ the array will look as follows:

$$
\begin{array}{llll}
\mathbf{p}_i & & & \\
\mathbf{t}_1 & \mathbf{b}_0^1 & & \\
\mathbf{t}_2 & \mathbf{b}_1^1 & \mathbf{b}_0^2 & \\
\mathbf{p}_{i+1} & \mathbf{b}_2^1 & \mathbf{b}_1^2 & \mathbf{b}_0^3
\end{array}
$$

The point $\mathbf{b}_0^3$ is a point on a cubic curve between data points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. The curve that is drawn through the data is also calculated with the method mentioned above, this is done by letting the parameter t vary over the interval $[0, 1]$.
We now have a method to define a C^1 curve between a pair of data points. If we want to add extra points to our data set, by interpolating a single points between each pair of data points, we want the new set of data points to define the same C^1 curve as the original set of data points did.

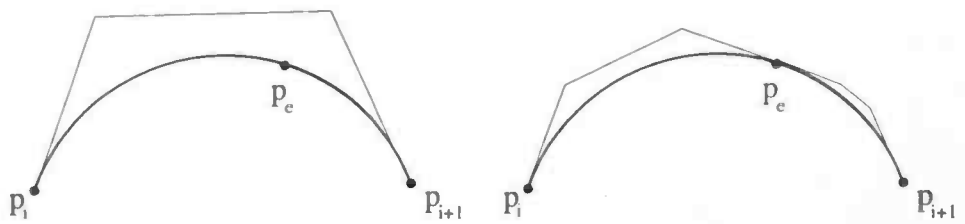


Figure 6: An example of what was proven by Schwartz.

This means that the two sub-curves must join in a C^1 way at point $\mathbf{p}_e$ (see Figure 6). A prove of the C^1-continuity in point $\mathbf{p}_e$ is given by *Schwartz* in [4, §6]. He reparameterizes the original curve $x(t)$ via $t = a + bu$:

$$y(u) = x(a + bu)$$

From this we conclude:

$$\{y(u) \mid 0 \leq u \leq 1\} = \{x(t) \mid a \leq t \leq a + b\}$$

This reparameterization means that $y(u)$ is a segment of the curve $x(t)$ that starts at $x(a)$ and ends at $x(a+b)$. Since we wish to divide the original curve in two sub-curves (see figure 6), we will reparameterize the original curve two times. Once via $t = bu$, this gives us the curve between points $\mathbf{p}_i$ and $\mathbf{p}_e$

in figure 6, and once via $t = b + u(1 - b)$, which gives us the curve between points $\mathbf{p}_e$ and $\mathbf{p}_{i+1}$.
Since the sub-curves coincide with the original curve $x(t)$ and together form the original curve, the connection between the two sub-curves (see Figure 6) will join in a C^1 way.

Application.

The method of finding and calculating points on a curve is also implemented in a program. For this purpose we embedded the described methods in an existing drawing program. This program was written by *Douglas A. Young*; see [5, chapter 13].

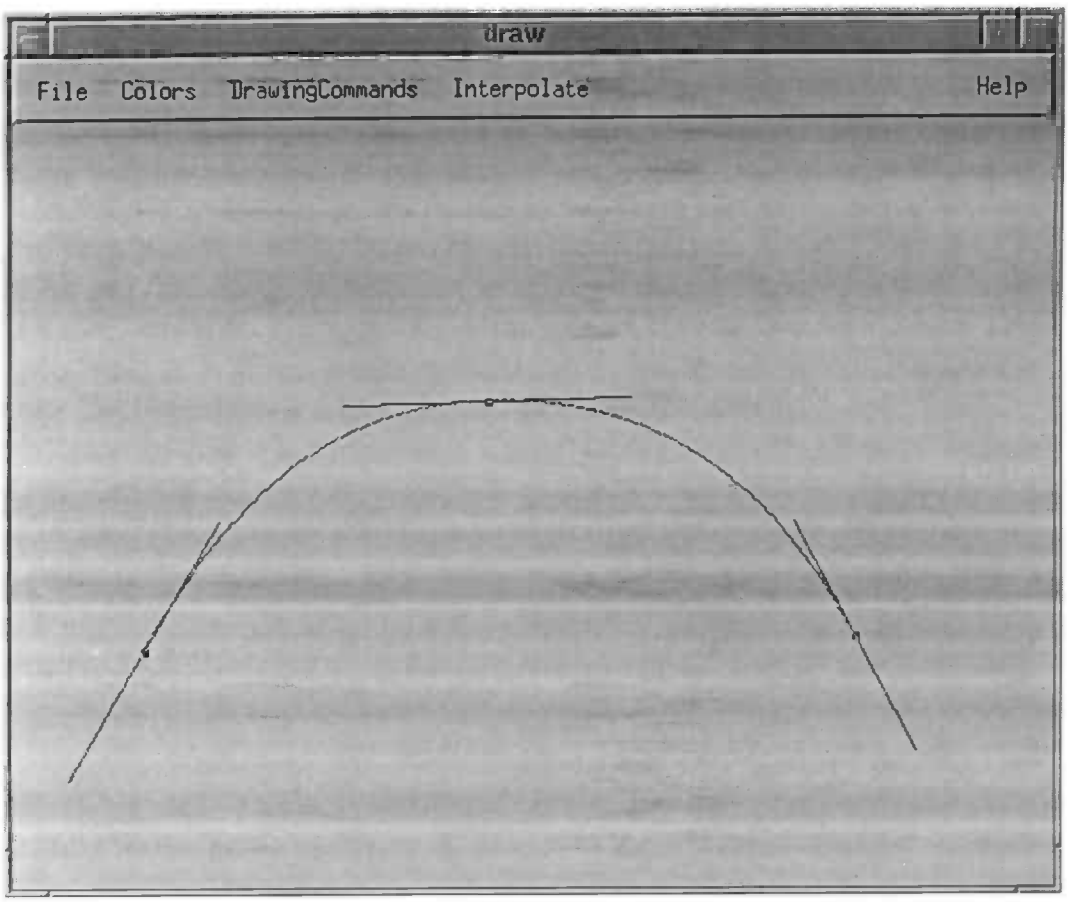


Figure 7: The application of 2-D interpolation.

To be able to interpolate a curve we first need to select the right drawing function. We can do so by pressing the button "DrawingCommands". We then select "Curve" from the pop-up menu. Now we can insert the data points by pressing the left mouse button. The last data point of the curve

is selected by pressing the right button. The mouse cursor will now jump back to the first data point. By moving the mouse we can specify the tangent vector in that data point. If the right tangent vector is specified, we can set it by pressing the right mouse button. The mouse cursor will now jump to the next data point. After determining all tangent vectors, we can interpolate either a curve or a point between two data points. This is done by pressing the button marked "Interpolate". We can now choose for interpolating a curve or interpolating a point between each pair of data points, by pressing either the button marked "Add Points" or by pressing the button marked "Draw Curve". Interpolating the points is done by setting the parameter t in equation (5) equal to $\frac{1}{2}$. This gives us a point that lies on the center of the curve between each pair of consecutive data points. The curve is drawn by calculating a number of points on the curve. For a part of the curve between two data points, we let the parameter t in equation (5) vary from 0 to 1, by steps of δ. Since we only use integer coordinates in our application we can set δ equal to $\frac{1}{\|\mathbf{p}_{i+1}-\mathbf{p}_i\|}$. By using integer coordinates we are sure that $\|\mathbf{p}_{i+1}-\mathbf{p}_i\| \geq 1$. If this length equals 1, we do not have to interpolate a curve since the two points are two adjacent pixels on the screen.

3 3-D interpolation

3.1 Introduction

In this section we present an interpolation scheme proposed by *Bruce R. Piper* in [3, §5].
The interpolation scheme assumes we have the following data: a set of edges and vertices in $\mathbb{R}^3$ which form a triangulation. A triangulation is a set of triangles. Each edge of this triangulation occurs in at most two triangles. Each vertex of the triangulation can be part of several edges, but is part of at most two edges which only occur in one triangle in the triangulation. Tangential data is given for each vertex in the triangulation.
The interpolation scheme solves the following problem:
interpolate for each triangle points on a C^1 continuous surface over all triangles within the triangulation, which are determined by the vertices of the triangle and the tangential data in these vertices.

We will first give a global description of the interpolation scheme, and later on it will be given in detail.
As interpolation method we wish to use the *de Casteljau* algorithm. The parameters of this algorithm are the barycentric coordinates, with respect to a given triangle, of the point we wish to interpolate. In order to be able to use the de Casteljau algorithm we need to define a set of control points called the control net (see figure 9). The scheme proposed by *Bruce R. Piper* produces a control net. A control net is a set of control points that defines a surface of a certain degree. Points on this surface can be calculated by using the *de Casteljau* algorithm. The control points are, as we already saw in the previous section, derived from the input data. Instead of $n+1$ control points for a curve of degree n, we now need $\frac{1}{2}(m+1)(m+2)$ control points for a surface of degree m (see [1, §8.2]). The control net is produced in five steps, to be described in detail in section 3.8:

Step 1. Create for each triangle in the input data an initial net which describes a surface of degree 3.

Step 2. Subdivide the net into three sub-nets each describing a surface of degree three. Together they describe same surface as the initial net.

Step 3. Adjust the center control point of each of the sub-nets.

Step 4. Elevate the degree of the surfaces described by the sub-nets, by adjusting the sub-nets, so that the sub-net now describe a surface of degree 4 (for details on degree elevation: see section 3.6).

Step 5. Adjust the control points of the sub-nets.

The control points calculated in Step 1 are called candidate control points because in order to get a C^1 surface we need to adjust these points. In steps 2, 3 and 5 we will adjust these points in order to get a C^1 surface. We start by defining a control net of degree 3 because these control points can be created relatively easy. A cubic control net consists of ten control points. Three of these control points are given input data. The seven other points are derived from these points and the tangential data in these points. Of these seven points, six points lie on the edge of the control net. One point is in the center of the control net, this point is called the center control point.
In Step 2 we will subdivide the triangle into three sub-triangles. The reason for the subdivision is that, although the surface defined on each triangle looks smooth, two adjacent triangles [1] fit together, without subdivision and further adjustments that are described in Step 3 and 5, at sharp angles.
In Step 3 we adjust the center control point of the sub-triangle. This is also done to prevent sharp angles between adjacent triangles. The points are adjusted so that the center control point of two adjacent sub-triangles (sub-triangles of two triangles that share a common edge) and two control points on the common edge are coplanar.
In Step 4 we raise the degree of the surface described by the control net. These control points are still candidate control points. To ensure tangent plane continuity between two adjacent triangles we still need to adjust some of the control points. We need to raise the degree of the surface to 4 because in general it is impossible to determine the center control points of a control net that defines a surface of degree 3, so that they satisfy a necessary condition for tangent plane continuity (see *Bruce R. Piper* in [3, §3])
In Step 5 we adjust some of the control points. This is done to ensure the tangent plane continuity between two triangles that share a common edge. This finally gives us the control net we need to interpolate a smooth surface defined by the given data.
We will now first explain some methods and concepts mentioned above in more detail. Finally, a detailed description of the interpolation scheme is

[1] Adjacent triangles are triangles that have one edge in common.

given.

3.2 Barycentric coordinates

Since we wish to use the de Casteljau algorithm for our interpolation scheme we will explain the concept of barycentric coordinates because they are used as parameters by the de Casteljau algorithm. In this algorithm the barycentric coordinates determine the point we wish to interpolate with this algorithm.
Any point in the plane can be expressed in terms of *barycentric coordinates* with respect to any non-degenerate triangle in that plane. A triangle is non-degenerate if the vertices of the triangle are not on one straight line. Suppose a triangle $\mathcal{T}$ has vertices $\mathbf{T}_1$,$\mathbf{T}_2$ and $\mathbf{T}_3$.
The triple of numbers (τ_1, τ_2, τ_3), called *barycentric coordinates* of $\mathbf{P}$ with respect to $\mathbf{T}_1$,$\mathbf{T}_2$ and $\mathbf{T}_3$, is uniquely determined by

$$\mathbf{P} = \sum_{i=1}^{3} \tau_i \mathbf{T}_i \quad \text{and} \quad \sum_{i=1}^{3} \tau_i = 1 \tag{6}$$

The barycenter of a triangle is the center of gravity of this triangle. An interpretation of the barycentric coordinates is that if we hang weights τ_i on the vertices $\mathbf{T}_i$ the center of gravity of triangle $\mathcal{T}$ is $\mathbf{P}$.

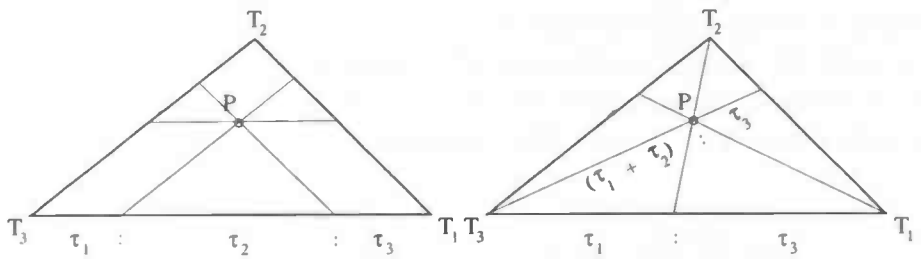


Figure 8: An example of Barycentric coordinates

Equation (6) forms a 3×3 linear system which has the unique solution which is obtained by applying *Cramer's rule*:

$$\tau_1 = \frac{area(\mathbf{P}, \mathbf{T}_2, \mathbf{T}_3)}{area(\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3)}, \tau_2 = \frac{area(\mathbf{T}_1, \mathbf{P}, \mathbf{T}_2)}{area(\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3)}, \tau_3 = \frac{area(\mathbf{T}_1, \mathbf{T}_2, \mathbf{P})}{area(\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3)}.$$

In order for the above formula to be well defined, we require
$area(\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3) \neq 0$, which means that $\mathbf{T}_1$,$\mathbf{T}_2$ and $\mathbf{T}_3$ must not lie on a straight line. But since $\mathbf{T}_1$, $\mathbf{T}_2$ and $\mathbf{T}_3$ form a nondegenerate triangle, this situation will never occur.
Because *Cramer's* rule makes use of determinants, the area used in the above equation can be defined as:

$$area(\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3) = \frac{1}{2} \begin{vmatrix} \mathbf{T}_1.x & \mathbf{T}_2.x & \mathbf{T}_3.x \\ \mathbf{T}_1.y & \mathbf{T}_2.y & \mathbf{T}_3.y \\ 1 & 1 & 1 \end{vmatrix}$$

In this equation $\mathbf{T}_i.x$ stands for the x-coordinate of point $\mathbf{T}_i$ and $\mathbf{T}_i.y$ stands for the y-coordinate of $\mathbf{T}_i$ $(i = 1, 2, 3)$.
An important property of barycentric coordinates is that they are invariant under affine transformations. This means that the barycentric coordinates of a point do not change if the point and the triangle are transformed by an affine map (see [2, §1.1]).
Barycentric coordinates can also be used for linear interpolation. For any point with $\sum \tau_i = 1$ and $\tau_i \geq 0$ we find a point inside the triangle, but for other values of τ_i we find points in the face spanned by the triangle.
Later on we will use functions that are defined over triangles. We can write these functions as $f(\tau)$ with $\tau = (\tau_1, \tau_2, \tau_3)$. This notation presents us a problem: how to handle differentiation. The term $\frac{\partial f}{\partial \tau_i}$ does not have a geometric interpretation since barycentric coordinates with respect to a triangle $\mathcal{T}$ define a plane and not a space. To solve this problem we have to use *directional derivatives*. Suppose ν is a vector expressed in barycentric vector coordinates so that ν defines a direction with respect to which a directional derivative can be taken:

$$D_\nu f(\tau) = \lim_{h \to 0} \frac{1}{h}(f(\tau + h\nu) - f(\tau)) = \sum_{i=1}^{3} \frac{\partial f}{\partial \tau_i}(\tau)\nu_i$$

Since τ is a triple of numbers, the functions f is a function with domain $\mathbf{R}^3$. The domain of f is restricted to $\Lambda = \{(\tau_1, \tau_2, \tau_3) \mid \sum \tau_i = 1\}$. A property of barycentric vector-coordinates is that they sum to zero. This property also ensure that $\tau + \nu \in \Lambda$ for all $\tau \in \Lambda$. This implies that the expression $(f(\tau + h\nu) - f(\tau))$, as well as $D_\nu f(\tau)$, is well defined.

3.3 Bernstein Polynomials

In this section we will briefly discuss *Bernstein* polynomials because we will use them for computing (cross-boundary) derivatives. In this section we will first give a definition of the Bernstein polynomial and we also show that Berstein polynomials can be used for describing a surface over a triangle.

Definition Bernstein polynomials of degree n over a triangle are defined by

$$B_{\mathbf{i}}^n(\tau) = \frac{n!}{i_1!i_2!i_3!}\tau_1^{i_1}\tau_2^{i_2}\tau_3^{i_3} \tag{7}$$

with $B_{\mathbf{i}}^n = 0$ if one of the components of $\mathbf{i}$ is negative or greater than n, $\mathbf{i} = (i_1, i_2, i_3)$ is a multi-index with $|\,\mathbf{i}\,| = n$ and $\tau = (\tau_1, \tau_2, \tau_3)$ is the barycentric coordinate of a point in the triangle.

Two properties of Bernstein polynomials are that they have only one maximum over the triangle, namely $B_{\mathbf{i}}^n$ assumes its maximum at $\tau = \frac{\mathbf{i}}{n}$ (see [2, §1.3]) and that $\sum_{|\mathbf{i}|=n} B_{\mathbf{i}}^n(\tau) \equiv 1$ (also in [2, §1.3]), and $B_{\mathbf{j}}^n(\tau) \geq 0$ when $\tau_i \geq 0$ $(i = 1, 2, 3)$.
We can also use the Bernstein polynomials to describe a surface over a triangle. This surface is called the *Bernstein-Bézier surface* and it is described by making use of a *control net*. This control net consists of several control points. These control points form a triangular structure (see figure 9). The control points $\mathbf{b}_{\mathbf{i}}$ are defined by a multi-index $\mathbf{i} = (i_1, i_2, i_3)$. We will use the notation $|\,\mathbf{i}\,| = i_1 + i_2 + i_3$. The multi-index of the control points of a control net that defines a surface of degree n (see Figure 9) all sum to n $(|\,\mathbf{i}\,| = n)$. Two points of the control net $\mathbf{b}_{\mathbf{i}}$ and $\mathbf{b}_{\mathbf{j}}$ are connected by an edge if $|\,\mathbf{i} - \mathbf{j}\,| = 2$. We have now explained the notation that is used in the definition of a Berstein-Bézier surface so that we can now give the definition of it.

Definition The *Bernstein-Bézier* surface $\mathbf{b}^n$ with control points $\mathbf{b}_{\mathbf{j}}$, $|\,\mathbf{j}\,| = n$ is defined by

$$\mathbf{b}^n(\tau) = \sum_{|\mathbf{j}|=n} B_{\mathbf{j}}^n(\tau)\mathbf{b}_{\mathbf{j}}$$

The number of control points needed for the interpolation depends on the degree of the surface. If we want to interpolate a point on a surface of degree n we need $\frac{1}{2}(n+1)(n+2)$ control points. The numbers $\frac{1}{2}(n+1)(n+2)$ are called the *triangle numbers* (see [1, §18.2]). The triangle numbers are equal to the dimension of the space of polynomials of degree n in three variables. Since the surface is determined by three polynomials of this type, it is also the minimum number of control points .The representation of $\mathbf{b}^n$ is unique, this means that $\{B_{\mathbf{j}}^n \mid |\,\mathbf{j}\,| = n$ and i_1, i_2, i_3 are all non-negative $\}$ form a basis for all polynomials of degree n that are defined over a triangle $\mathcal{T}$.

3.4 The de Casteljau Algorithm

The *de Casteljau* algorithm is used for interpolating a point on a surface defined by a set of control points, called a control net (see also Figure 9). The points of the control net are given.
If we want to interpolate a point $\mathbf{b}_{\mathbf{0}}^n$ on a surface of degree n we will use the *barycentric coordinate* τ of this point as a parameter for the *de Casteljau* algorithm. The sub-index $\mathbf{0}$ $(= (0,0,0))$ is used as a starting value for the algorithm. The super index n denotes the degree of the surface we wish to interpolate the point upon.

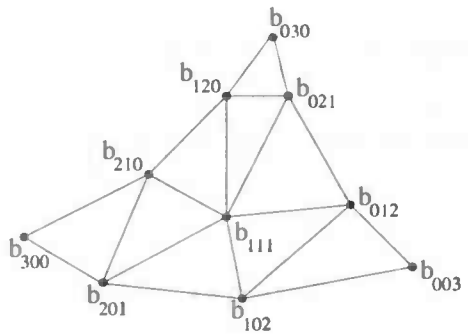


Figure 9: An example of a cubic control net.

In the algorithm we will use the following notation: $|\tau| = \sum \tau_i$. We will also use the *multi-indices* $\mathbf{e_1}$, $\mathbf{e_2}$ and $\mathbf{e_3}$, where $\mathbf{e}_1 = (1,0,0)$, $\mathbf{e}_2 = (0,1,0)$ and $\mathbf{e}_3 = (0,0,1)$.
The *de Casteljau* algorithm is defined as follows

Given: a control net of degree n, with control points $\mathbf{b_i} \in \mathbb{R}^3$, interpolate a point with barycentric coordinates $\tau = (\tau_1, \tau_2, \tau_3)$.

Set:

$$\mathbf{b}_{\mathbf{i}}^{r}(\tau) = \tau_1 \mathbf{b}_{\mathbf{i}+\mathbf{e}_1}^{r-1}(\tau) + \tau_2 \mathbf{b}_{\mathbf{i}+\mathbf{e}_2}^{r-1}(\tau) + \tau_3 \mathbf{b}_{\mathbf{i}+\mathbf{e}_3}^{r-1}(\tau) \tag{8}$$

with $r = 1, \ldots, n$ and $|\,\mathbf{i}\,| = n - r$.

Where $\mathbf{b}_{\mathbf{m}}^{0}(\tau)$, this is a point of the control net, or $\mathbf{b}_{\mathbf{m}}^{0}(\tau) = \mathbf{b_m}$.
The point interpolated by this algorithm is $\mathbf{b}_{0}^{n}$.
We will now present a proof that we can express the values computed by the de Casteljau algorithm in term of Bernstein polynomials.

$$b_{\mathbf{i}}^{r}(\tau) = \sum_{|\mathbf{j}|=r} b_{\mathbf{i}+\mathbf{j}} B_{\mathbf{j}}^{r}(\tau); \qquad |\mathbf{i}| = n - r \tag{9}$$

We can prove, by the principle of mathematical induction, that the surface described by equation (9) is equal to the surface described by the *de Casteljau* algorithm (equation (8)). First we define the predicate $\mathcal{P}(n)$, for integer $n \geq 0$:

$$\mathcal{P}(n) : \forall \text{ multi} - \text{indices } \mathbf{i} \text{ with } |\,\mathbf{i}\,| \leq n : b_{\mathbf{i}}^{n}(\tau) = \sum_{|\mathbf{j}|=r} b_{\mathbf{i}+\mathbf{j}} B_{\mathbf{j}}^{n}(\tau)$$

Since $B_{\mathbf{j}}^{n}(\tau) = 1$ for $|\,\mathbf{j}\,| = n = 0$ and $\mathbf{b}_{\mathbf{i}}^{0}(\tau) = \mathbf{b_i}$ we see that $\mathcal{P}(0)$ holds.
Assume $\mathcal{P}(n)$, now prove $\mathcal{P}(n+1)$:

$$\begin{aligned}
& \mathbf{b}_{\mathbf{i}}^{n+1}(\tau) \\
= \;& \{\text{de Casteljau}\} \\
& \tau_1 \mathbf{b}_{\mathbf{i}+\mathbf{e}_1}^{n}(\tau) + \tau_2 \mathbf{b}_{\mathbf{i}+\mathbf{e}_2}^{n}(\tau) + \tau_3 \mathbf{b}_{\mathbf{i}+\mathbf{e}_3}^{n}(\tau) \\
= \;& \{\text{ Induction hypothesis } \mathcal{P}(n)\} \\
& \tau_1 \sum_{|\mathbf{j}|=n} \mathbf{b}_{\mathbf{i}+\mathbf{e}_1+\mathbf{j}} B_{\mathbf{j}}^{n}(\tau) + \tau_2 \sum_{|\mathbf{j}|=n} \mathbf{b}_{\mathbf{i}+\mathbf{e}_2+\mathbf{j}} B_{\mathbf{j}}^{n}(\tau) + \tau_3 \sum_{|\mathbf{j}|=n} \mathbf{b}_{\mathbf{i}+\mathbf{e}_3+\mathbf{j}} B_{\mathbf{j}}^{n}(\tau) \\
= \;& \tau_1 \sum_{|\mathbf{k}|=n+1} \mathbf{b}_{\mathbf{k}+\mathbf{i}} B_{\mathbf{k}-\mathbf{e}_1}^{n}(\tau) + \tau_2 \sum_{|\mathbf{k}|=n+1} \mathbf{b}_{\mathbf{k}+\mathbf{i}} B_{\mathbf{k}-\mathbf{e}_2}^{n}(\tau) + \tau_3 \sum_{|\mathbf{k}|=n+1} \mathbf{b}_{\mathbf{k}+\mathbf{i}} B_{\mathbf{k}-\mathbf{e}_3}^{n}(\tau) \\
= \;& \{\mathbf{k} = (k_1, k_2, k_3), \tau_1 B_{\mathbf{k}-\mathbf{e}_1}^{n}(\tau) = \frac{k_1}{n+1} B_{\mathbf{k}}^{n}(\tau), \tau_2 B_{\mathbf{k}-\mathbf{e}_2}^{n}(\tau) = \frac{k_2}{n+1} B_{\mathbf{k}}^{n}(\tau),
\end{aligned}$$

$$\tau_3 B^n_{\mathbf{k}-\mathbf{e}_3}(\tau) = \frac{k_3}{n+1} B^n_{\mathbf{k}}(\tau) \text{ and } \tau_1 + \tau_2 + \tau_3 = 1\}$$

$$\sum_{|\mathbf{k}|=n+1} \left(\frac{k_1}{n+1}\mathbf{b}_{\mathbf{k}+\mathbf{i}} B^n_{\mathbf{k}}(\tau) + \frac{k_2}{n+1}\mathbf{b}_{\mathbf{k}+\mathbf{i}} B^n_{\mathbf{k}}(\tau) + \frac{k_3}{n+1}\mathbf{b}_{\mathbf{k}+\mathbf{i}} B^n_{\mathbf{k}}(\tau)\right)$$

$$= \sum_{|\mathbf{k}|=n+1} \mathbf{b}_{\mathbf{i}+\mathbf{k}} B^{n+1}_{\mathbf{k}}(\tau)$$

We now have proven that equation (9) represents the *de Casteljau* algorithm.

We can arrange the cubic Bernstein polynomials in a triangular scheme (following the triangular structure of a cubic control net (see Figure 9)):

$$\begin{array}{cccc} & & \tau_2^3 & \\ & & 3\tau_1\tau_2^2 \quad 3\tau_2^2\tau_3 & \\ & & 3\tau_1^2\tau_2 \quad 6\tau_1\tau_2\tau_3 \quad 3\tau_2\tau_3^2 & \\ \tau_1^3 & \quad 3\tau_1^2\tau_3 & \quad 3\tau_1\tau_3^2 & \quad \tau_3^3 \end{array}$$

It now is easy to see that interpolating a point with barycentric coordinates τ on the edge of the triangle (that is a point with one of the $\tau_i = 0$) that only the control points on the same edge are used to interpolate the point.
We can use this triangular scheme for calculating $\mathbf{b}^n_0(\tau)$ by multiplying the control points $\mathbf{b}_\mathbf{i}$ with $B^3_\mathbf{i}$, we get an explicit formula for this point on the surface which is found by using the cubic de Casteljau algorithm:

$$\begin{aligned} \mathbf{b}^3_0(\tau) \;=\; & \tau_2^3\mathbf{b}_{030} + \\ & 3\tau_1\tau_2^2\mathbf{b}_{120} + 3\tau_2^2\tau_3\mathbf{b}_{021} + \\ & 3\tau_1^2\tau_2\mathbf{b}_{210} + 6\tau_1\tau_2\tau_3\mathbf{b}_{111} + 3\tau_2\tau_3^2\mathbf{b}_{012} + \\ & \tau_1^3\mathbf{b}_{300} + 3\tau_1^2\tau_3\mathbf{b}_{201} + 3\tau_1\tau_3^3\mathbf{b}_{102} + \tau_3^3\mathbf{b}_{003} \end{aligned}$$

Note that if we want to interpolate a point with one of the $\tau_j = 0$, which we will do in our implementation, the algorithm will only use the points of the edge of the sub-net that has index $i_j = 0$. Suppose we want to interpolate a point P with barycentric coordinates $(0, \tau_2, \tau_3)$. If we look at equation (8) we see that every time the index is raised with $\mathbf{e}_1$, it is multiplied by zero,

so that this point has no effect on the interpolation of P. This means that P is a combination of the control points $\mathbf{b}_{0ij}$. When all of the τ_i are not equal to zero, all of the control points of the control net are used to interpolate a point.

3.5 Derivatives

The derivatives will be used in our interpolation scheme to achieve tangent plane continuity. If we want to calculate the r-th derivative, $r > 1$, of an arbitrary function f, defined on Λ (see section 3.2), with respect to the direction $\mathbf{d} = (d_1, d_2, d_3)$ which satisfies $d_1 + d_2 + d_3 = 0$, we can use the following definition:

$$D^r_{\mathbf{d}} f(\tau) = \frac{\partial^r}{\partial t^r}(f(\tau + t\mathbf{d}))\mid_{t=0}$$

We can rewrite this to (see [1, §18.4]):

$$D^r_{\mathbf{d}}\mathbf{b}^n(\tau) = \frac{n!}{(n-r)!}\sum_{|\mathbf{j}|=r}\mathbf{b}^{n-r}_{\mathbf{j}}(\tau)B^r_{\mathbf{j}}(\mathbf{d}) \qquad (10)$$

We can find the first order derivative by substituting $r = 1$ in equation (10). We can reduce this to :

$$D_{\mathbf{d}}\mathbf{b}^n(\tau) = n(d_1\mathbf{b}^{n-1}_{\mathbf{e}_1}(\tau) + d_2\mathbf{b}^{n-1}_{\mathbf{e}_2}(\tau) + d_3\mathbf{b}^{n-1}_{\mathbf{e}_3}(\tau))$$

where $\mathbf{d} = (d_1, d_2, d_3)$.
Since this holds for all directions $\mathbf{d}$ with $d_1 + d_2 + d_3 = 0$, it follows that $\mathbf{b}^{n-1}_{\mathbf{e}_1}(\tau)$,$\mathbf{b}^{n-1}_{\mathbf{e}_2}(\tau)$ and $\mathbf{b}^{n-1}_{\mathbf{e}_3}(\tau)$ define the slope of tangent plane at $\mathbf{b}^n(\tau)$ in the direction $\mathbf{d}$.
Suppose we want to calculate the directional derivative along one of the edges with a direction not parallel to the edge, we need a *cross-boundary-derivative*, which is a special case of the directional derivative. Without loss of generality we can choose the cross-boundary-derivative over edge $\tau_1 = 0$. The cross-boundary-derivative which is derived from a dual of equation (10), see [1, §18.4], is as follows:

$$D^r_{\mathbf{d}}\mathbf{b}^n(\tau^0) = \frac{n!}{(n-r)!}\sum_{|\mathbf{i}^0|=n-r}\mathbf{b}^r_{\mathbf{i}^0}(\mathbf{d})B^{n-r}_{\mathbf{i}^0}(\tau^0) \qquad (11)$$

where $\tau_0 = (0, \tau_2, \tau_3)$, $\mathbf{i}^0 = (0, i_2, i_3)$ and $\mathbf{d}$ is a direction not parallel to this edge.
A condition for C^s continuity, $0 \leq s \leq n$ between two adjacent triangles $\mathcal{T}$, with control points $\mathbf{b_i}$, and $\hat{\mathcal{T}}$, with control points $\hat{\mathbf{b}}_\mathbf{j}$ is (see [1, §18.6, equation 18.20]):

$$\hat{\mathbf{b}}_{(r,i_2,i_3)} = \mathbf{b}^r_{\mathbf{i}^0}(\mathbf{d}); \qquad r = 0, \ldots, s. \tag{12}$$

where $\mathbf{d} = (d_1, d_2, d_3)$ is a direction not parallel to the common edge, and $\mathbf{i}^0 = (0, i_2, i_3)$. In our interpolation scheme we want C^1 continuity between two adjacent triangles. If we use $s = 1$ in equation (12) and (8) we get:

$$\hat{\mathbf{b}}_{(1,i_2,i_3)} = d_1 \mathbf{b}_{(1,i_2,i_3)} + d_2 \mathbf{b}_{(0,i_2+1,i_3)} + d_3 \mathbf{b}_{(0,i_2,i_3+1)}.$$

Since $d_1 + d_2 + d_3 = 0$, the four points $\hat{\mathbf{b}}_{1,i_2,i_3}$, $\mathbf{b}_{1,i_2,i_3}$, $\mathbf{b}_{0,i_2+1,i_3}$ and $\mathbf{b}_{0,i_2,i_3+1}$ are affinely dependent, and hence coplanar.

Corollary When two triangles $\mathcal{T}$, with control points $\hat{\mathbf{b}}_\mathbf{j}$, and $\hat{\mathcal{T}}$, with control points $\hat{\mathbf{b}}_\mathbf{j}$, join is C^1 continuous, then the control points $\hat{\mathbf{b}}_{\mathbf{i}^0+\mathbf{e}_1}$, $\mathbf{b}_{\mathbf{i}^0+\mathbf{e}_1}$, $\mathbf{b}_{\mathbf{i}^0+\mathbf{e}_2}$ and $\mathbf{b}_{\mathbf{i}^0+\mathbf{e}_3}$ along the common edge, with multi-index $\mathbf{i}^0 = (0, i_2, i_3)$, are coplanar.

We will use the above corollary in our interpolation scheme to ensure tangent plane continuity between two adjacent triangles.

3.6 Degree Elevation

As the name already suggests degree elevation is a method used to raise the degree of a the Berstein-Bézier basis. In the interpolation scheme we will use this method.
This method is applied to the control net (or points in case of a curve) of a surface with a Berstein-Bézier basis of degree n, and provides us with a new control net that describes the surface with a Bernstein-Bézier basis of degree $n + 1$. The method of degree elevation presented here is described by Farin in [2, §1.4].
If we want to write a surface $\mathbf{b}^n(\tau)$ as a Bernstein-Bézier basis of degree $n+1$ we get:

$$\sum_{|\mathbf{i}|=n} b_\mathbf{i} B^n_\mathbf{i}(\tau) = \sum_{|\mathbf{i}|=n+1} b^{(1)}_\mathbf{i} B^{n+1}_\mathbf{i}(\tau)$$

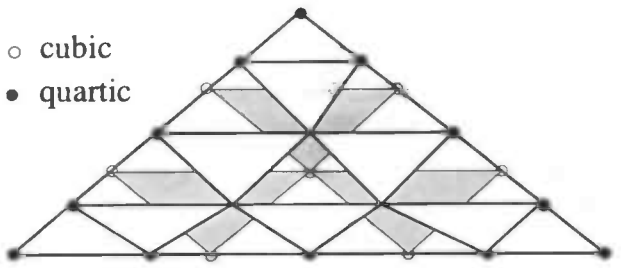
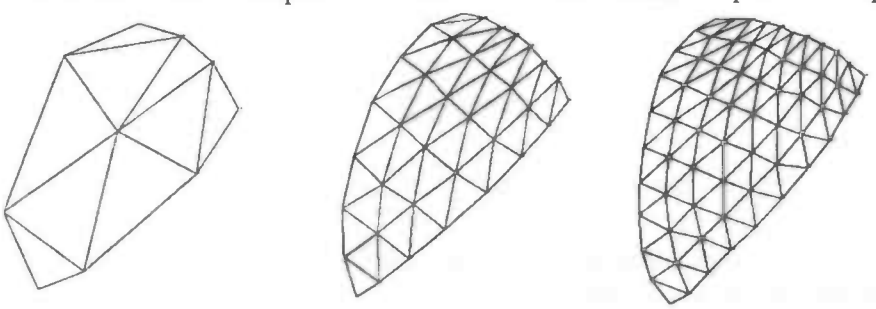


Figure 10: Raising a cubic control net to a quartic control net

So we get a new control net that describes the same surface as the original control net did.
From the previous expression one derives:

$$\mathbf{b}_{\mathbf{i}}^{(1)} = \frac{1}{n+1}(i_1\mathbf{b}_{\mathbf{i}-\mathbf{e}_1} + i_2\mathbf{b}_{\mathbf{i}-\mathbf{e}_2} + i_3\mathbf{b}_{\mathbf{i}-\mathbf{e}_3})$$

where $\mathbf{i} = (i_1, i_2, i_3)$.
If we repeat the degree elevation often enough, the control net will approach the surface defined by the control net. A proof of this can be found Farin [2, §1.4, Theorem 1.3].
We will now show an example of a control net that defines a part of a sphere.



The left control net is a control net that defines a part of a sphere by a surface of degree 3. It is constructed by the method we will explain later on in section 3.8. The center control net defines a surface of degree 7. This net is constructed by elevating the degree of the surface described by the left-most net. The right-most net describes a surface of degree 11. This net is also constructed by elevating the degree of the leftmost surface. The degree raising algorithm is applied eight times to the control net of degree 3, which gives us degree 11, with $\frac{1}{2}(11+1)(12+1) = 78$ control points.

3.7 Subdivision

Subdivision will later on be used in our interpolation scheme to ensure the tangent plane continuity between two adjacent triangles by preventing sharp edges between two adjacent patches. The subdivision algorithm will divide a triangle into three sub-triangles which together describe the same surface as the original triangle did.

Subdividing a triangle is derived from domain transformation. Suppose we have two triangles $\mathcal{T}$ and $\hat{\mathcal{T}}$ (see Figure 11) with common edge $\mathbf{T_2T_3}$, and we want to transform the domain of $\mathcal{T}$ to that of $\hat{\mathcal{T}}$. With domain transformation we can describe the polynomial surface $\mathbf{b}^n(\tau)$ with barycentric coordinates of $\mathcal{T}$ as a polynomial surface over $\hat{\mathcal{T}}$ (see Figure 11).

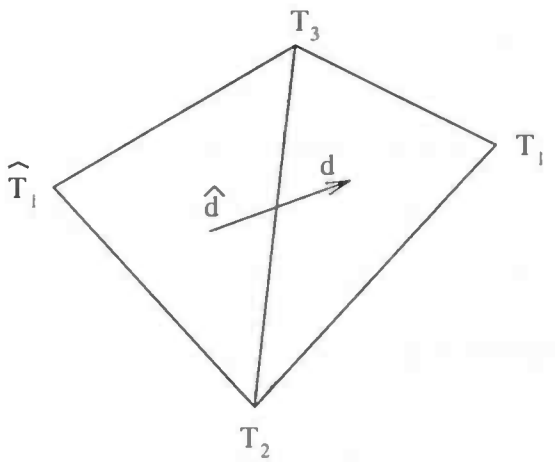


Figure 11: An example of domain transformation

Let $\mathcal{T}$ have barycentric coordinates τ and let $\hat{\mathcal{T}}$ have barycentric coordinates $\hat{\tau}$, the common edge $\mathbf{T_2T_3}$ will correspond to the barycentric coordinates $\tau^0 = (0, \tau_2, \tau_3)$ and to $\hat{\tau}^0 = (0, \hat{\tau}_2, \hat{\tau}_3)$. Let $\hat{T}_1$ have barycentric coordinates σ with respect to $\mathcal{T}$.

The surface (described by these two triangles) can then be written in two ways:

$$\sum_{|\mathbf{i}|=n} \mathbf{b}_\mathbf{i} B^n_\mathbf{i}(\tau) = \sum_{|\mathbf{i}|=n} \hat{\mathbf{b}}_\mathbf{i} B^n_\mathbf{i}(\hat{\tau}) \tag{13}$$

We now want to find the Bézier points of the surface that are defined over

$\hat{\mathcal{T}}$. Since the two triangles share a common edge, we immediately find (from equation (13)) $\mathbf{b}_{\mathbf{i}^0} = \hat{\mathbf{b}}_{\mathbf{i}^0}$ with $|\,\mathbf{i}\,| = n$, $\mathbf{i} = (i_1, i_2, i_3)$ and $\mathbf{i}^0 = (0, i_2, i_3)$. We can determine the other Bézier points by using the directional derivative of a direction $\mathbf{d}$ that is not parallel to the common edge $\mathbf{T_2T_3}$. $\hat{\mathbf{d}}$ is the same direction expressed in barycentric vector coordinates of $\hat{\mathcal{T}}$. If we substitute this in equation (11) we get:

$$\sum_{|\mathbf{i}^0|=n-r} \mathbf{b}^r_{\mathbf{i}^0}(\mathbf{d}) B^{n-r}_{\mathbf{i}^0}(\tau^0) = \sum_{|\mathbf{i}^0|=n-r} \hat{\mathbf{b}}^r_{\mathbf{i}^0}(\hat{\mathbf{d}}) B^{n-r}_{\mathbf{i}^0}(\hat{\tau}^0); \qquad r = 0, \ldots, n$$

If we now can show that the above equation holds for $r = n$, we can conclude that the surfaces described by $\mathcal{T}$ and $\hat{\mathcal{T}}$ coincide.
Since $\tau^0 = \hat{\tau}^0$ (substituting this in the previous equation) we find:

$$\sum_{|\mathbf{i}^0|=n-r} \mathbf{b}^r_{\mathbf{i}^0}(\mathbf{d}) = \sum_{|\mathbf{i}^0|=n-r} \hat{\mathbf{b}}^r_{\mathbf{i}^0}(\hat{\mathbf{d}}); \qquad r = 0, \ldots, n$$

Because the edge is a common edge we have $\mathbf{b}_{\mathbf{i}^0} = \hat{\mathbf{b}}_{\mathbf{i}^0}$. This also means that

$$\mathbf{b}^r_{\mathbf{i}^0}(\mathbf{d}) = \hat{\mathbf{b}}^r_{\mathbf{i}^0}(\hat{\mathbf{d}}); \qquad r = 0, \ldots, n$$

because the points $\mathbf{b}^r_{\mathbf{i}^0}$ and $\hat{\mathbf{b}}^r_{\mathbf{i}^0}$ completely depend on the control points on the common edge.
Hence the two polynomials $\mathbf{b}^r_{\mathbf{i}^0}$ and $\hat{\mathbf{b}}^r_{\mathbf{i}^0}$ agree in all derivatives in all directions $\mathbf{d}$ up to order n:

$$D^r_{\mathbf{d}} \mathbf{b}_{\mathbf{i}^0}{}^r(\tau) = D^r_{\hat{\mathbf{d}}} \mathbf{b}_{\mathbf{i}^0}{}^r(\hat{\tau})$$

and so they must be equal:

$$\mathbf{b}^r_{\mathbf{i}^0}(\tau) = \hat{\mathbf{b}}^r_{\mathbf{i}^0}(\hat{\tau}); \qquad r = 0, \ldots, n \tag{14}$$

We now have Theorem 2.8 from [2] which states:

Theorem Let $\mathbf{b}^n$ (with control points $\mathbf{b}_\mathbf{i}$) be defined over $\mathcal{T} = \{T_1, T_2, T_3\}$ and let $\hat{\mathbf{b}}^n$ (with control points $\hat{\mathbf{b}}_\mathbf{i}$) be defined over $\hat{\mathcal{T}} = \{\hat{T}_1, T_2, T_3\}$. The two polynomials $\mathbf{b}^n$ and $\hat{\mathbf{b}}^n$ are identical if and only if

$$\hat{\mathbf{b}}_{\mathbf{i}^r} = \mathbf{b}_{\mathbf{i}^0}(\sigma); \quad 0 \le r \le n, \tag{15}$$

where $\mathbf{i}^r = (r, i_2, i_3)$; $|\,\mathbf{i}^r\,| = n$, and σ are the barycentric coordinates of $\hat{T}_1$ with respect to $\mathcal{T}$.

Using (15) we can find the Bézier points $\hat{b}_i$ from the given b_i (see [2, §2.3]). However, equation (15) doesn't provide a stable method for points that are outside the triangle, because it then is a formula that uses repeated *extrapolation* and has the same numerical problems as other extrapolation schemes. The difference between domain transformation and subdivision is that $\hat{T}_1$ is inside $\mathcal{T}$ (see Figure 12), so we don't extrapolate but interpolate. We can use equation (15) (see [2, §3.2]) for subdivision without worrying about the method not being stable. With $\hat{T}_1$ inside of $\mathcal{T}$ the subdivision formula gives us three sub-control nets. The intermediate points of the de Casteljau algorithm $b_{\mathbf{i}}^r(\tau)$, $|\ \mathbf{i}\ | = n - r$, are the points of the control net of the surfaces defined over the triangles $\{\hat{T}_1, T_3, T_1\}$, $\{\hat{T}_1, T_1, T_2\}$ and $\{\hat{T}_1, T_2, T_3\}$, this gives us:

Corollary The intermediate points $b_{\mathbf{i}}^r$ with $|\ \mathbf{i}\ | = n - r$ of the de Casteljau algorithm, are the control points of the three sub-triangles $\{\hat{T}_1, T_3, T_1\}$, $\{\hat{T}_1, T_1, T_2\}$ and $\{\hat{T}_1, T_2, T_3\}$.

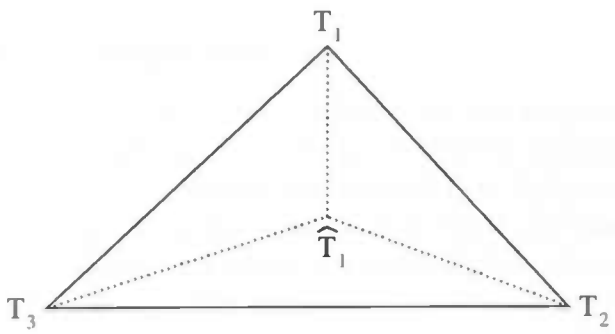


Figure 12: An example of a subdivision

We will now present an example of a subdivision. Suppose we want to subdivide a triangle at its centroid (i.e. the point with barycentric coordinates $\sigma = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$). We now wish to redefine the control net that used to describe the surface over the whole triangle to one that describes the surface over a part of the triangle. We will call our original control net b_i and our sub-triangle is $\hat{b}_j$. Suppose we want to know the position of $\hat{b}_{210}$. If we apply equation (15) we get:

$$
\begin{aligned}
\hat{\mathbf{b}}_{210} = \mathbf{b}^1_{010}(\sigma) &= \sum_{|j|=2} \mathbf{b}_{010+j} B^2_j(\sigma) \\
&= \mathbf{b}_{210} B^2_{200}(\sigma) + \mathbf{b}_{120} B^2_{110}(\sigma) + \mathbf{b}_{111} B^2_{101}(\sigma) + \\
&\quad \mathbf{b}_{021} B^2_{011}(\sigma) + \mathbf{b}_{030} B^2_{020}(\sigma) + \mathbf{b}_{012} B^2_{002}(\sigma) \\
&= (\frac{1}{3})^2 \mathbf{b}_{210} + 2(\frac{1}{3})^2 \mathbf{b}_{120} + 2(\frac{1}{3})^2 \mathbf{b}_{111} + \\
&\quad 2(\frac{1}{3})^2 \mathbf{b}_{021} + (\frac{1}{3})^2 \mathbf{b}_{030} + (\frac{1}{3})^2 \mathbf{b}_{012}
\end{aligned}
$$

We now have a closed expression for $\hat{\mathbf{b}}_{210}$, expressed in the control points of the original surface. If we want to get three sub-triangles out of one triangle we have to apply this method to each of the edges of the original triangle. This provides us with three sub-nets that describe the same surface as the original control net.

3.8 Interpolation scheme

We will now present the interpolation scheme, as was proposed by *Bruce R. Piper* in [3, §5]. We will use this scheme instead of the scheme presented by Farin in [2, §4.2] because Farin uses control nets that describe a surface of degree 3, while Piper presents a proof that there are cases of piecewise polynomial surfaces of degree 3 where a control net that describes an overall C^1 surface connot be found. The scheme proposed by Piper can be used for interpolating surfaces, given some points on the surface and the tangent plane in these points. The problem is described as follows:
We have given a triangulation in $\mathbf{R}^3$. On this triangulation the following constraints must hold: each edge of this triangulation occurs in at most 2 triangles. Each vertex occurs in at least 2 triangles. Each triangle is connected with at least one other triangle. For each vertex tangential data is given by the normal vector of the tangent plane in this vertex.
We now wish to interpolate for each triangle a point on a C^1 continuous surface which is determined by the vertices of the triangle and the tangential data in these vertices.

As we explained in Section 3.1, this interpolation scheme consists of five steps to create a control net.

In step 1 we create an candidate control net. This net will be adjusted for reasons of tangent plane continuity in steps 2, 3 and 5. In step 2 we will subdivide the candidate control net, that was created in step 1, into three sub-nets. These sub-nets will be further adjusted in steps 3 and 5. In step 3 we will adjust the center control points of the sub-nets that were created in step 2. They are adjusted in such a way that each control point becomes coplanar with the common edge and the center point of an adjacent sub-control net of an adjacent triangle. In step 4 we will adjust the degree of surface that each sub-control net, created in step 3, describes. The reason for this is explained in detail in the following description of step 4. In step 5 we will adjust some control points of the sub-control nets created in step 4, to achieve the tangent plane continuity between adjacent triangles. We also will present a proof that we need at least a control net that described a surface of degree 4 to be able to get tangent plane continuity between two adjacent triangles.
We will now present a detailed description of each step.

Step 1. Create an initial control net.

In this step we will create a candidate set of control points. These points will be used to create the final control net. The candidate set is deduced from the input data. For each triangle in the triangulation we will calculate a candidate control net. This candidate control net will be adjusted later on to avoid sharp angles between two adjacent triangles.

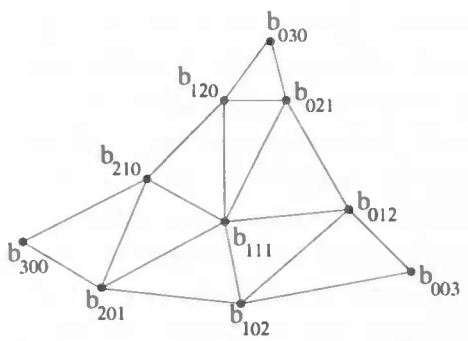


Figure 13: An example of a candidate control net.

From each triangle we need the vertices and the normal vector of the tangent plane of these vertices. The control points b_{300}, b_{030} and b_{003} are equal

to one of the vertices of the input triangle. The other boundary control points (control points except $\mathbf{b}_{300}$, $\mathbf{b}_{030}$, $\mathbf{b}_{003}$ and $\mathbf{b}_{111}$ (see also figure 13)) are constructed by projecting one vertex on the tangent plane of both other vertices of the triangle (see figure 14). This gives us two new candidate control points and it is done for all three vertices so that we get six new candidate control points.

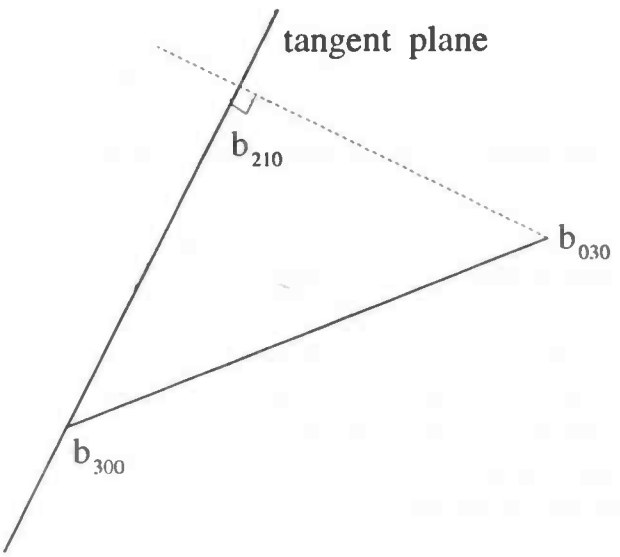


Figure 14: A 2 D representation of the projection.

However, only projecting a vertex of the triangle on the tangent plane of an other vertex, is not always sufficient. It is possible that the control points get "crossed" (see figure 15), this means e.g. the control points along the edge $(\mathbf{b}_{030}, \mathbf{b}_{300})$ are $\mathbf{b}_{030}$, $\mathbf{b}_{120}$, $\mathbf{b}_{210}$ and $\mathbf{b}_{030}$ instead of $\mathbf{b}_{300}$, $\mathbf{b}_{210}$, $\mathbf{b}_{120}$ and $\mathbf{b}_{030}$. This "crossing" results in cusps in the surface. A solution to this problem is scaling. By scaling we mean multiplying the vector from a vertex to the projected point on the tangent plane of this vertex by a certain factor. Piper suggests (in [3, §5]) a factor that depends on the distance between the two vertices (e.g. $\mathbf{b}_{300}$, $\mathbf{b}_{030}$), namely one-third plus one-nineth the distance from the vertex to the projected point $\mathbf{q}$ so that

$$\mathbf{b}_{210} = \mathbf{b}_{300} + (\frac{1}{3} + \frac{1}{9}\|\mathbf{q} - \mathbf{b}_{300}\|)(\mathbf{q} - \mathbf{b}_{300})$$

All the other control points on the edge of the triangle are calculated similarly.

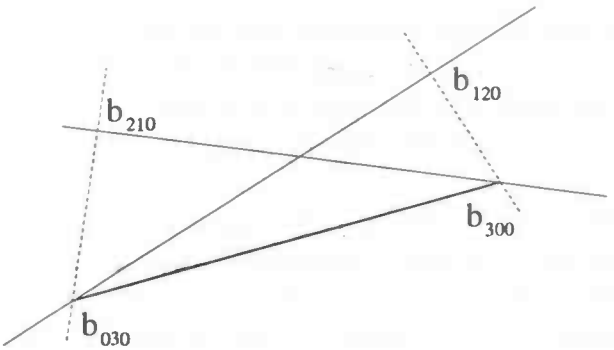


Figure 15: Problems which can occur if scaling isn't used.

However this scaling factor depends completely on the given data and must therefore be determined by trying several values until a result is found that looks fine.
We now have all the control points on the edges, but we still need one point in the center of the triangle (b_{111} in Figure 9). The exact position of this point is not important since it will be repositioned later-on when we consider the surface continuity. We will use the method that Farin suggests in [2, §4.1]. He suggests to use a C^0 interpolant which depends on the control points on the edges of the triangle:

$$b_{111} = \frac{1}{4}(b_{201} + b_{102} + b_{012} + b_{021} + b_{120} + b_{210}) - \frac{1}{6}(b_{300} + b_{030} + b_{003})$$

We now have an candidate position of point b_{111}. We need this position to be able to subdivide the triangle. The exact position of is determined later on in step 3 and 5. This candidate control net now describes a Bézier surface that goes through the points b_{300},b_{030} and b_{003} and is tangent to the given tangent plane in these points.
We now have the complete candidate control net, so we can continue with step 2.

Step 2. Subdividing the surface.

In this step we subdivide the triangle into three sub-triangles. The reason for the subdivision is that, although the surface, defined by the candidate control net, that was produced in step 1, over each triangle looks smooth, two adjacent triangles often fit together at sharp angles. The result of the

subdivision does not provide us with adjacent triangles that join in a smooth way, because our choice of the points $\mathbf{b}_{111}$ is arbitrary, but it is a necessary step to eventually get a smooth join between the adjacent triangles. The adjustments to make a smooth join between two adjacent triangles are made in step 3 and 5.
We will apply the subdivision algorithm, as described in section 3.7, on the control net we have so far. We divide the triangle at the centroid (see also Figure 12). The centroid of a triangle has barycentric coordinates $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. We will use formula (15) where the parameter is the barycentric coordinates of the centroid.
This formula gives us one of the three sub-control nets, namely the control net over $\mathbf{b}_{030}$, $\mathbf{b}_{003}$ and the centroid of the triangle. Note that for point $\hat{\mathbf{b}}_{300}$, the right hand side of equation (15) is determined by *de Casteljau* algorithm for interpolating a point with barycentric coordinates $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. The other points calculated by the subdivision method are the intermediate point of the de Casteljau algorithm.
In order to get another sub-control net we have to change the indices of the formula. For the sub-control net over $\mathbf{b}_{300}$, $\mathbf{b}_{030}$ and the point corresponding to the centroid, the index $\mathbf{i}^r$ becomes (i_1, r, i_3) and $\mathbf{i}^0 = (i_1, 0, i_3)$. The last sub-control net is found by setting $\mathbf{i}^r = (i_1, i_2, r)$ and $\mathbf{i}^0 = (i_1, i_2, 0)$.
We now have three sub-control nets that describe a surface of degree 3. Together they describe the same surface as the original control net.

Step 3. Adjusting the center points.

In this step we adjust the center points of the sub-triangles. The result of this step provides us with a set of triangles that join smoothly but the interior of the triangles is no longer smooth. Making the interior of the triangles smooth again is done in step 5, but this may require increasing the number of degrees of freedom by raising the degree of the surface described by the control net, which is done in step 4. See [3, §3] for an example showing the necessity of degree at least 4. In this step the points are adjusted so that the center points (like $\mathbf{b}_{111}$, see Figure 13) of two adjacent sub-triangles (sub-triangles of two triangles that share a common edge) and two control points on the common edge are coplanar.
If the points are not already coplanar the center points will be adjusted in the following way:
first we project the center point $\mathbf{b}_{111}$ on the plane through the common edge e

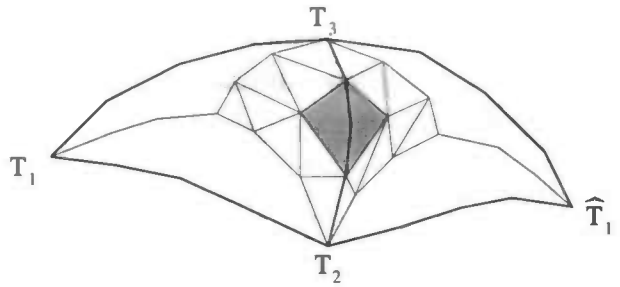


Figure 16: Adjusting the center control points of two adjacent patches

and the center point d_{111}, the center point of the adjacent triangle (see figure 17). The projected points are not coplanar, we still need to scale one of the line segments (the segment from a center point to the projected point) with a factor f and the other one with a factor $1 - f$ $(0 < f < 1)$.

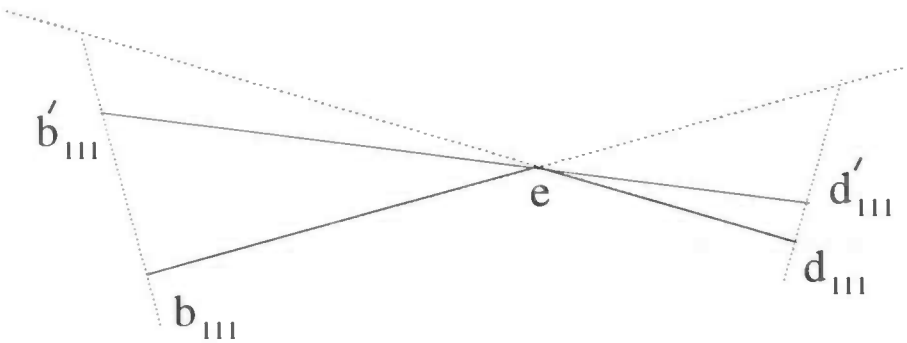


Figure 17: A projection of two adjacent triangles with common edge e

The factor f is a weighted average of the heights of the triangle with base being the common edge e of the two adjacent triangle. The factor f is chosen in such a way that it favors the triangle with the smallest height. This triangle is favored because otherwise this would result in relatively big changes to the sub-control net of small triangles. The effect of the same change to a bigger triangle is relatively small. Suppose we have two triangles with heights h_1 and h_2, and h_1 is larger than h_2. We set the scaling factor f equal to $\frac{h_1}{h_1+h_2}$. We will then multiply the vector, from the center point to the projection of this center point, of the triangle with the largest height with f. The vector of the triangle with the smallest height will be multiplied with $1 - f$.

Step 4. Raising the degree.

In this step we raise the degree of the surface described by the control net. These control points are still candidate control points because the adjustment we made to the control points in step 3 disturbed the C^1 continuity of the surface of a triangle. To ensure tangent plane continuity over the triangle and between two adjacent triangles we still need to adjust some of the control points. This last adjustment is done in step 5. We need to raise the degree of the surface to 4 because it is, in some cases, impossible to determine the center control points of a control net that defines a surface of degree 3, so that they satisfy a necessary condition for tangent plane continuity, see *Bruce R. Piper* in [3, §3].

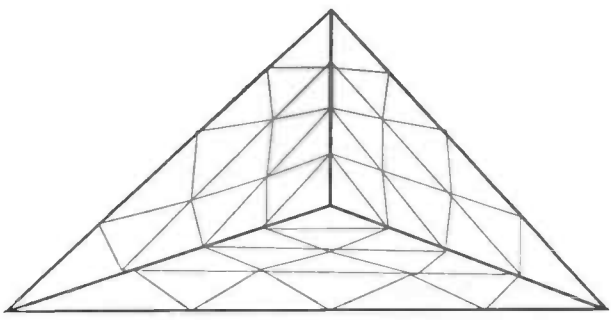


Figure 18: The triangle after subdivision and degree elevation

In this section Piper proves that a control net that describes a surface of degree 3 does not always give us a solution for a smooth surface over and between triangles. We will now present this proof. The proof is given by presenting a counter example of a cubic interpolation problem. Before we present this counter example we will first explain some of the notation used in that example.

Suppose we have two triangles, $\mathcal{P}$ and $\mathcal{Q}$, which share a common edge with barycentric coordinates $(\tau_1, \tau_2, 0)$. A necessary requirement for tangent plane continuity is that the triangles $\mathcal{P}$, with surface $\mathbf{p}^n$ and control points $\mathbf{p}_i$, and $\mathcal{Q}$, with surface $\mathbf{q}^n$ and control points $\mathbf{q}_j$, join continuously over the common edge. The construction of the control net, as was explained in step 1, ensure that the triangles $\mathcal{P}$ and $\mathcal{Q}$ join continuously over the common edge. We will now present a formula with which we can find the tangent vector at any point on the surface described by the control net of $\mathcal{P}$, corresponding to the vector

d (see equation (10) with $r = 1$):

$$D_{\mathbf{d}}\mathbf{p}^n(\tau) = n \sum_{|\mathbf{i}|=n-1} B_{\mathbf{i}}^{n-1}(d_1\mathbf{p}_{\mathbf{i}+\mathbf{e}_1} + d_2\mathbf{p}_{\mathbf{i}+\mathbf{e}_2} + d_3\mathbf{p}_{\mathbf{i}+\mathbf{e}_3}) \qquad (16)$$

The tangent plane of the surface of $\mathcal{P}$ of a point on the common edge of $\mathcal{P}$ and $\mathcal{Q}$, is spanned by two different directions. We can choose these directions, without loss of generality, as $\mathbf{d_1} = (-1, 1, 0)$, a direction parallel to the common edge, and $\mathbf{d}_2 = (-1, 0, 1)$, a direction not parallel to the common edge. Since we have continuity over the common edge, the tangent vectors $D_{\mathbf{d}_1}\mathbf{b}^n$ and $D_{\mathbf{d}_1}\mathbf{q}^n$ are identical. To achieve tangent plane continuity between $\mathcal{P}$ and $\mathcal{Q}$ the tangent vectors of $D_{\mathbf{d}_1}\mathbf{p}^n$, $D_{\mathbf{d}_2}\mathbf{p}^n$ and $D_{\mathbf{d}_1}\mathbf{q}^n$ must all lie in the same plane (see section 3.7). Using formula (16) we get

$$\begin{aligned} D_{\mathbf{d}_1}\mathbf{p}^n(\tau_1, \tau_2, 0) &= D_{\mathbf{d}_1}\mathbf{q}^n(\tau_1, \tau_2, 0) = n(\mathbf{J} - \mathbf{I}) \\ D_{\mathbf{d}_2}\mathbf{p}^n(\tau_1, \tau_2, 0) &= n(\mathbf{K} - \mathbf{I}) \text{ and } D_{\mathbf{d}_2}\mathbf{q}^n\tau_1, \tau_2, 0) = n(\mathbf{L} - \mathbf{I}) \end{aligned}$$

In the following notation $n = 3$, $u + v = 1$, $\mathbf{p}_\mathbf{i}$ are the control points of the control net of triangle $\mathcal{P}$, and $\mathbf{q}_\mathbf{i}$ are the control points of an adjacent triangle $\mathcal{Q}$.

$$\begin{aligned} \mathbf{I} &= \sum_{i+j=n-1} B^{n-1}_{(i,j,0)}(\tau_1, \tau_2, 0)\mathbf{p}_{i+1,j,0} \\ \mathbf{J} &= \sum_{i+j=n-1} B^{n-1}_{(i,j,0)}(\tau_1, \tau_2, 0)\mathbf{p}_{i,j+1,0} \\ \mathbf{K} &= \sum_{i+j=n-1} B^{n-1}_{(i,j,0)}(\tau_1, \tau_2, 0)\mathbf{p}_{i,j,1} \\ \mathbf{L} &= \sum_{i+j=n-1} B^{n-1}_{(i,j,0)}(\tau_1, \tau_2, 0)\mathbf{q}_{i,j,1} \end{aligned}$$

Note that **I**, see figure 19, is a point in the triangle spanned by $\mathbf{p}_{300}$, $\mathbf{p}_{210}$ and $\mathbf{p}_{120}$. To be more precise, **I** is a point on a quadratic bézier curve (see section 2). The same observation can be made about the points **J**, **K** and **L**.
A necessary condition for the tangent plane continuity between $\mathcal{P}$ and $\mathcal{Q}$ is that:

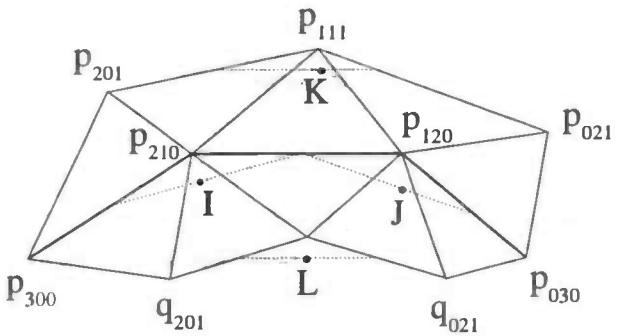


Figure 19: The points **I**, **J**, **K** and **L** together with a part of the control-nets of $\mathcal{P}$ and $\mathcal{Q}$.

$$\det \begin{bmatrix} 1 & \mathbf{I} \\ 1 & \mathbf{J} \\ 1 & \mathbf{K} \\ 1 & \mathbf{L} \end{bmatrix} = 0 \tag{17}$$

We will now show a cubic case, for which it is impossible to find the points $\mathbf{b}_{111}$ and $\mathbf{q}_{111}$ satisfying equation (17). The input data for this case are the following control points:

$$\begin{aligned} \mathbf{p}_{300} &= \mathbf{q}_{300} = (0,0,0) & \mathbf{p}_{210} &= \mathbf{q}_{210} = (1,0,0) \\ \mathbf{p}_{120} &= \mathbf{q}_{120} = (2,0,1) & \mathbf{p}_{030} &= \mathbf{q}_{030} = (4,0,1) \\ \mathbf{p}_{201} &= (1,1,0) & \mathbf{q}_{201} &= (1,-1,0) \\ \mathbf{p}_{021} &= (3,1,1) & \mathbf{q}_{021} &= (3,-1,0) \end{aligned}$$

We now need to determine the center control points $\mathbf{b}_{111}$ and $\mathbf{q}_{111}$ in such a way that equation (17) still holds. Let $\mathbf{b}_{111} = (x_p, y_p, z_p)$ and $\mathbf{q}_{111} = (x_q, y_q, z_q)$. If we now calculate the determinant of equation (17) with the unknown u, v, $\mathbf{b}_{111}$ and $\mathbf{q}_{111}$, we get the expression:

$$2uv(d_0u^4 + 4d_1u^3v + 6d_2u^2v^2 + 4d_3uv^3 + d_4v^4) \tag{18}$$

where

$$\begin{aligned} d_0 &= (z_p + z_q) - 2, \\ d_1 &= ((z_q y_p - z_p y_q) - (y_p - y_q) - (x_p + x_q) + (z_p + z_q) + 2)/2 \\ d_2 &= (4(y_q x_p - y_p x_q) + 4(z_q y_p - z_p y_q) + 4(y_p - y_q) + 3(z_p + z_q) - 4)/6 \end{aligned}$$

$$
\begin{aligned}
d_3 &= (2(z_q y_p - z_p y_q) - (y_p - y_q) - (x_p + x_q) + (z_p + z_q) + 2)/2 \\
d_4 &= 2(z_p + z_q) - 2
\end{aligned}
$$

Since uv is not equal to zero and the one-dimensional Bernstein polynomials are independent, equation (18) is equal to zero if and only if all the d_i are equal to zero. However, it is impossible to choose z_p and z_q so that d_0 and d_4 are both equal to zero. This means that it is impossible to choose the points $\mathbf{p}_{111}$ and $\mathbf{q}_{111}$ in such a way that equation (17) holds.
We can now conclude that, since a cubic surface satisfying equation (17) is not always realizable, it is necessary to use surfaces of degree at least 4. In the next step we will show that degree 4 always provides us with an overall smooth surface.

Step 5. Adjusting the control points

In this last step we will adjust some of the control points of the sub-control nets which each describe a surface of degree 4. This is done to ensure the tangent plane continuity between two triangles that share a common edge. This finally gives us the control net we need to interpolate a smooth surface defined by the given data.
We will present a proof, as was given by Piper in [3, §4], that we can always find a control net that realizes tangent plane continuity between two adjacent triangles. These control points may still not establish tangent plane continuity over the triangle itself, and therefore we will adjust them.

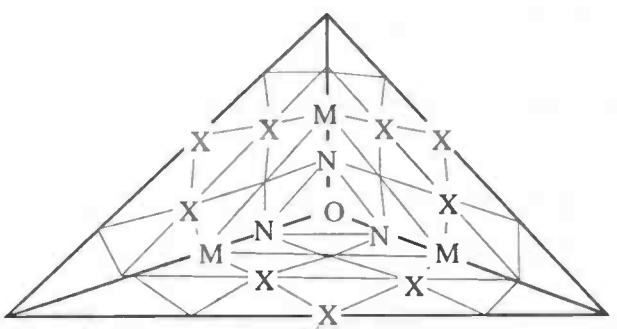


Figure 20: The triangle after subdivision and degree elevation

We will use the same formula for the points **I**, **J**, **K** and **L** as in step 4:

$$
\begin{aligned}
\mathbf{I} &= \sum_{i+j=n-1} B^{n-1}_{(i,j,0)}(u,v,0)\mathbf{p}_{i+1,j,0} \\
\mathbf{J} &= \sum_{i+j=n-1} B^{n-1}_{(i,j,0)}(u,v,0)\mathbf{p}_{i,j+1,0} \\
\mathbf{K} &= \sum_{i+j=n-1} B^{n-1}_{(i,j,0)}(u,v,0)\mathbf{p}_{i,j,1} \\
\mathbf{L} &= \sum_{i+j=n-1} B^{n-1}_{(i,j,0)}(u,v,0)\mathbf{q}_{i,j,1}
\end{aligned}
$$

In this proof we will use an equivalent condition of equation (17). This is done because an expansion of this formula with the new points **I**, **J**, **K** and **L** will be quit difficult to read. The condition we will use here is that for each u,v $(u+v=1)$ there exists scalars $E(u,v)$, $F(u,v)$, $G(u,v)$ and $H(u,v)$, not all zero, such that:

$$E(u,v)\mathbf{I} + F(u,v)\mathbf{J} + G(u,v)\mathbf{K} + H(u,v)\mathbf{L} = 0 \qquad (19)$$

The scalars are chosen to be linear functions of u and v:

$$
\begin{aligned}
E(u,v) &= e_1 u + e_2 v, \quad & F(u,v) &= f_1 u + f_2 v, \\
G(u,v) &= g_1 u + g_2 v, \quad & H(u,v) &= h_1 u + h_2 v.
\end{aligned}
$$

Here e_1, f_1, g_1, h_1, e_2, f_2, g_2 and h_2 are real constants, to be determined in such a way that the patches corresponding to triangles $\mathcal{P}$ and $\mathcal{Q}$ are C^1 continuous along the edge shared by $\mathcal{P}$ and $\mathcal{Q}$, i.e. such that the expansion (19) of (17) is satisfied.
We will use the following notation to make the proof more readable:

$$
\begin{aligned}
\mathbf{R}_i = \mathbf{p}_{3-i,i,1}, \quad \mathbf{T}_i = \mathbf{q}_{3-i,i,1} \quad & i = 0,1,2,3 \\
\mathbf{S}_i = \mathbf{p}_{4-i,i,0} = \mathbf{q}_{4-i,i,0} \quad & i = 0,1,2,3,4
\end{aligned}
$$

The points $\mathbf{S}_i$ in this notation are the Bézier points on a common edge between two adjacent triangles $\mathcal{P}$ and $\mathcal{Q}$, $\mathbf{R}_i$ are the control points of the control net of $\mathcal{P}$ which are parallel to this edge and $\mathbf{T}_i$ are the control points of the control net $\mathcal{Q}$ which are also parallel to this common edge (see Figure 21).
If we expand equation (19) we get:

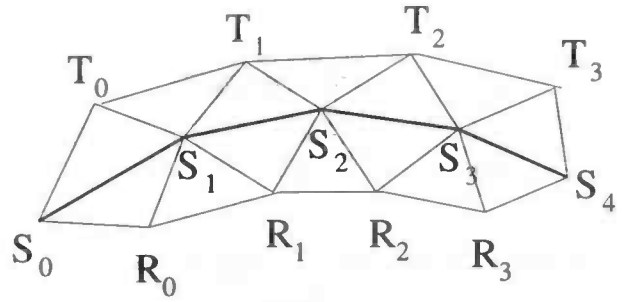


Figure 21: Bézier control points that are important for the surface continuity

$$u^4\mathbf{C}_0 + 4u^3v\mathbf{C}_1 + 6u^2v^2\mathbf{C}_2 + 4uv^3\mathbf{C}_3 + v^4\mathbf{C}_4 = 0 \qquad (20)$$

where

$$\begin{aligned}
\mathbf{C}_0 &= e_1\mathbf{S}_0 + f_1\mathbf{S}_1 + g_1\mathbf{R}_0 + h_1\mathbf{T}_0 \\
\mathbf{C}_1 &= [3(e_1\mathbf{S}_1 + f_1\mathbf{S}_2 + g_1\mathbf{R}_1 + h_1\mathbf{T}_1) + (e_2\mathbf{S}_0 + f_2\mathbf{S}_1 + g_2\mathbf{R}_0 + h_2\mathbf{T}_0)]/4 \\
\mathbf{C}_2 &= [2_e\mathbf{S}_1 + f_2\mathbf{S}_2 + g_2\mathbf{R}_1 + h_2\mathbf{T}_1 + e_1\mathbf{S}_2 + f_1\mathbf{S}_3 + g_1\mathbf{R}_2 + h_1\mathbf{T}_2]/2 \\
\mathbf{C}_3 &= [3(e_2\mathbf{S}_2 + f_2\mathbf{S}_3 + g_2\mathbf{R}_2 + h_2\mathbf{T}_2) + (e_1\mathbf{S}_3 + f_1\mathbf{S}_4 + g_1\mathbf{R}_3 + h_1\mathbf{T}_3)]/4 \\
\mathbf{C}_4 &= e_2\mathbf{S}_3 + f_2\mathbf{S}_4 + g_2\mathbf{R}_3 + h_2\mathbf{T}_3
\end{aligned}$$

Since u and v are both not equal to zero, the only solution to equation (20) is that all the $\mathbf{C}_i$ are equal to zero. Since the points in the expression $\mathbf{C}_0$ are all coplanar, because they are all on, or projected on the tangent plane through $\mathbf{S}_0$, we can choose e_1, f_1, g_1 and h_1 such that $\mathbf{C}_0$ will be equal to zero, while not all of e_1, f_1, g_1 and h_1 are zero, but so that $e_1 + f_1 + g_1 + h_1 = 0$. Since $\mathbf{R}_0$ and $\mathbf{T}_0$ are coplanar with $\mathbf{S}_0$ and $\mathbf{S}_1$, and are on opposite sides of the line through $\mathbf{S}_0$ and $\mathbf{S}_1$ we can choose g_1 and h_1 so that $g_1 + h_1 = 1$. Since the four points in $\mathbf{C}_0$ are coplanar we can't solve $\mathbf{C}_0 = 0$ by making a linear system by using all three coordinates of the points in $\mathbf{C}_0$. So we leave out one of the coordinates and use the equations $e_1 + f_1 + g_1 + h_1 = 0$ and $g_1 + h_1 = 1$ to get a solvable four-by-four system. We can solve the scalars e_2, f_2, g_2 and h_2 in the same way, by setting $g_2 + h_2 = 1$, $e_2 + f_2 + g_2 + h_2 = 0$ and by using $\mathbf{C}_4$.

In order to solve unknown control points $\mathbf{R}_1$, $\mathbf{R}_2$, $\mathbf{T}_1$, $\mathbf{T}_2$ and $\mathbf{S}_2$. We can simplify this by setting $4\mathbf{C}_1 = \mathbf{0}$, $6\mathbf{C}_2 = \mathbf{0}$ and $4\mathbf{C}_3 = \mathbf{0}$. We can rewrite this

as a system: $A\vec{x} = \vec{b}$ with

$$A = \begin{bmatrix} 3g_1 & 3h_1 & 0 & 0 & 3f_1 \\ 3g_2 & 3h_2 & 3g_1 & 3h_1 & 3(f_2 + e_1) \\ 0 & 0 & 3g_2 & 3h_2 & 3e_2 \end{bmatrix}$$

$$\vec{b} = -\begin{bmatrix} e_2\mathbf{S}_0 + f_2\mathbf{S}_1 + g_2\mathbf{R}_0 + h_2\mathbf{T}_0 + 3e_1\mathbf{S}_1 \\ 3(e_2\mathbf{S}_1 + f_1\mathbf{S}_3) \\ e_1\mathbf{S}_3 + f_1\mathbf{S}_4 + g_1\mathbf{R}_3 + h_1\mathbf{T}_3 + 3f_2\mathbf{S}_3 \end{bmatrix}$$

$$\vec{x} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{T}_1 \\ \mathbf{R}_2 \\ \mathbf{T}_2 \\ \mathbf{S}_2 \end{bmatrix}$$

This system can be solved when the rows of the matrix A are independent, if the rows are dependent we need an other method to solve this problem. We will now show how to find the points in $\hat{x}$ when the rows are dependent. These rows are dependent only if $e_1 = e_2$, $f_1 = f_2$, $g_1 = g_2$ and $h_1 = h_2$. If the rows are dependent, matrix A becomes:

$$A' = \begin{bmatrix} 3g_1 & 3h_1 & 0 & 0 & 3f_1 \\ 3g_1 & 3h_1 & 3g_1 & 3h_1 & 3(f_1 + e_1) \\ 0 & 0 & 3g_1 & 3h_1 & 3e_1 \end{bmatrix}$$

We can now see that if we add row 1 and 3 of matrix A' we get row 2 of matrix A'. If we now want to find a solution for $\vec{x}$ we get:

$$A' = \begin{bmatrix} 3g_1 & 3h_1 & 0 & 0 & 3f_1 \\ 0 & 0 & 3g_1 & 3h_1 & 3e_1 \end{bmatrix}$$

$$\vec{b} = -\begin{bmatrix} e_1\mathbf{S}_0 + f_1\mathbf{S}_1 + g_1\mathbf{R}_0 + h_1\mathbf{T}_0 + 3e_1\mathbf{S}_1 \\ e_1\mathbf{S}_3 + f_1\mathbf{S}_4 + g_1\mathbf{R}_3 + h_1\mathbf{T}_3 + 3f_1\mathbf{S}_3 \end{bmatrix}$$

which is a solvable two-by-five system. Hence, we can always find the points $\mathbf{R}_1$, $\mathbf{R}_2$, $\mathbf{T}_1$, $\mathbf{T}_2$ and $\mathbf{S}_2$ so that they satisfy tangent plane continuity between two adjacent triangles.

We now have enough information to adjust the control points for making a visually smooth join between two adjacent patches. After determining the matrix A, by solving $\mathbf{C_0}$ and $\mathbf{C_4}$ as described above, and $\vec{b}$, we adjust the points marked X in Figure 20. This can be done by solving the following equation:

$$A\vec{e} = \vec{b} - A\vec{y}$$

where $\vec{y}$ are the control points we have so far. $\vec{e}$ is the correction for the control points so that our new control points will be : $\vec{e} + \vec{y}$.
The adjustment of these points causes the join between two triangles to be smooth, but this adjustment and the adjustment of the center points in Step 3 can cause the join between two adjacent subtriangles of one triangle to loose its C^1 continuity. We will have to adjust some points inside the triangle to restore this smoothness. This is achieved by adjusting the points labeled M, N and O in Figure 20. These points will become the centroid of the three points surrounding them. By making this point the centroid of the surrounding control points, we ensure C^1 continuity between the subtriangles, because this adjustment makes the adjusted point and the three surrounding points coplanar. We will have to start with the points marked M since these points are used to modify the points marked N, which are used to modify the point marked O.
According to Piper, the described method gives us a smooth surface with the given control points.

We will now show that local changes do only have a local effect. This is important, because when you are designing an object and wish to change a part of the surface, you do not always want this change to affect the entire surface. Suppose we change a point $\mathbf{p}$ or the tangent plane in point $\mathbf{p}$ (see figure 22) where $\mathbf{p}$ is one of the data points.
It is clear that all the triangles that have $\mathbf{p}$ as a vertex, will change, and so will the associated surface patches. We will now show that this change in data will only have an effect on the triangles containing $\mathbf{p}$ as a vertex or sharing a common edge with the triangles that have $\mathbf{p}$ as a vertex. If we change point $\mathbf{p}$ in triangle $\triangle\mathbf{pqr}$ (see figure 22), the candidate sub-control nets and the centroid of this triangle will change. The candidate sub-control nets of the adjacent triangle $\triangle\mathbf{qrs}$ do not change, since the control nets of a

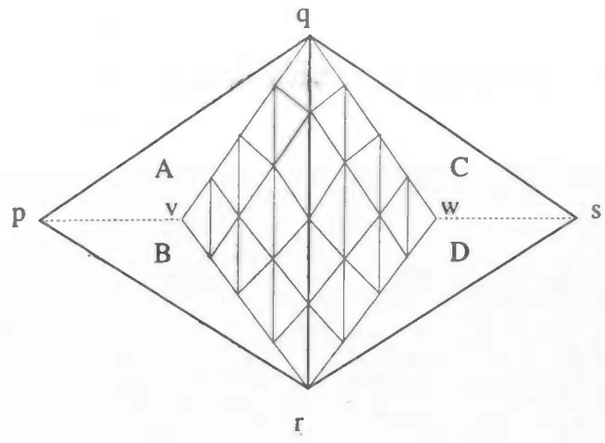


Figure 22: Local changes have only a local effect.

triangle depend only on the vertices and the tangent planes in these vertices. Of the triangle $\triangle$qrs the final control net of sub-triangle $\triangle$qwr will change most. The points that are determined for the tangent plane continuity (the points marked X in figure 20) will change. The control points on the edges qw and wr will also change because they are set to be be the centroid of the surrounding points. The other control points in the triangles $\triangle$qws and $\triangle$wrs do not change. Since the other triangles adjacent to triangle $\triangle$qrs, i.e. triangles that do not contain data point p, do not use the control points of triangle $\triangle$qrs, they will not be affected in any way by the changes made to data point p. So we can conclude that local changes will only have a local effect.

3.9 Application

We implemented the described method in an application. The application's interface looks as follows:

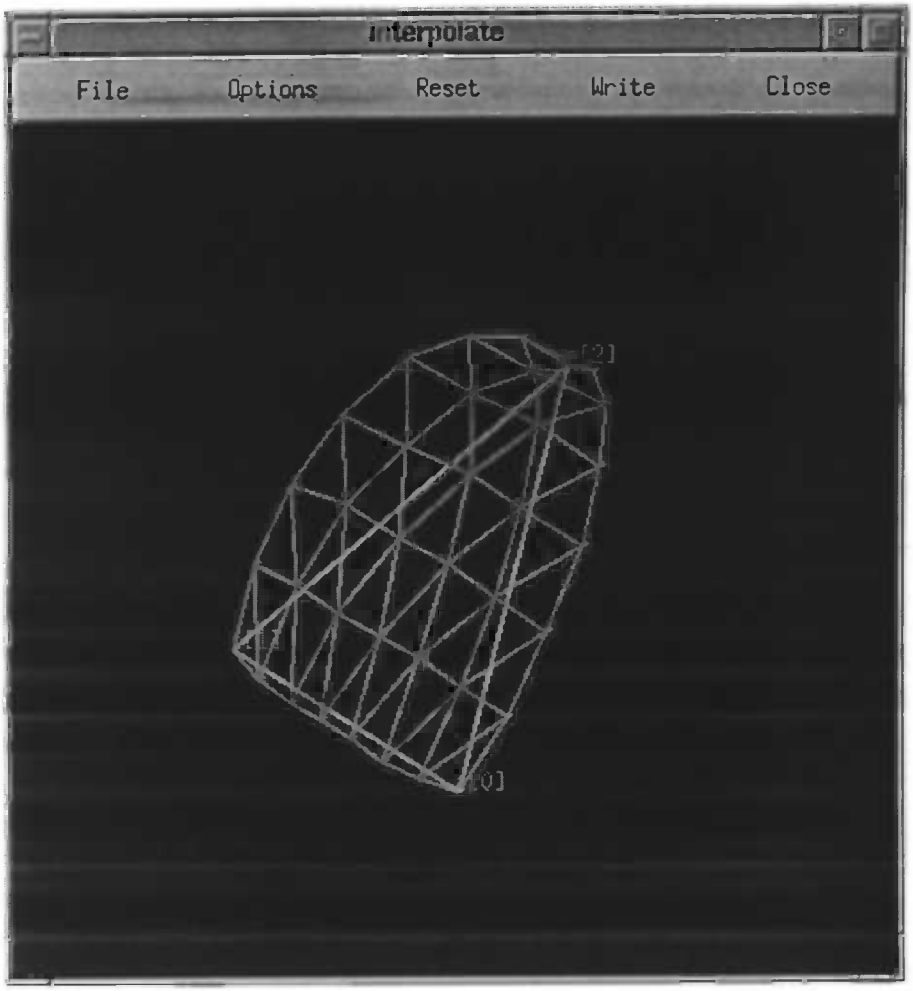


Figure 23: The interface of the application of 3-D interpolation.

This application can be used to view data and to interpolate new data points as was described in section 3.8. The application takes input from a file. This input is then displayed. We can now interpolate new data points. This interpolation can be done in one step. One can also choose to do the interpolation

step by step. These steps correspond with step 1 through 5, which were explained in section 3.8. Every time one step is executed, the result of this step will be displayed. It is also possible to write the result to a file. If we want to use the output again as an input file, only the data points will be stored in this file. If one chooses to use the output as an input for Mathematica ®, one gets the same data as the one that is displayed by the application. We have chosen to save the entire control net because one can use Mathematica ® to create postscript files of the displayed object.
The view point of the object displayed by the program can be modified by using the mouse. By pressing the left mouse button down and moving the mouse, the figure will rotate. The center mouse button is used for zooming in or out. Pressing this button and moving the mouse down will result in zooming out on the figure, move the mouse up and it will result in zooming in. The right mouse button can be used for moving the figure around. Pressing the button and moving the mouse will also move the figure.
We will now explain the buttons in the top part of the window.

The "File" button
This button is used for selecting files. Pressing this button will present a tear-off menu. This means that by clicking on the dotted line the menu will become a window which can be placed at a desired position. This menu consists of three buttons, a button labeled "Input File", a button labeled "Math File" and a button labeled "Output File". The button "Input File" will select a file which can be used as an input file of this application. The line of the file is equal to a face in the figure. A line can look as follows:

$$(((17,2,3),(3,2,1)),((13,3,4),(4,3,2)),((43,4,5),(5,4,3)))$$

The first three-tuple is, as are the third and fifth, a vertex of the face this line describes. The second three-tuple, as are the fourth and sixth, is the normal vector of the tangent plane in this vertex. The number of braces are of importance.
The button labeled "Math File" is used for selecting a file, to which output in Mathematica ® style can be written. This file can be used in Mathematica ® for examining the figure closer or making a postscript file of this figure.
The button labeled "Output File" is used for selecting a file wherein the data points, which are shown in the figure, can be written. This file can be used as an input file for this application.

The "Options" Button

This button is used for the interpolation. Pressing this button will present a menu with the following options.

Initial Net. Pressing this button will make an initial control net for each triangle, as was described in the previous section in *Step 1.*

Subdivide. Pressing this button will subdivide the initial control net into three sub control nets. If the initial control net was not made yet, it will be made first. This operation is equal to *Step 2*, as was described in the previous section. Instead of adjusting the indices for each sub-triangle, we rotate the control net so that the indices for the subdivision method stay the same (see equation (15)).

Adjust Center Points. Pressing this button will result in the adjustments of the center points of the control nets. If the previous two steps were not executed first, they will be executed first. This operation is equal to the one described in *Step 3* in the previous section.

Elevate Degree. Pressing this button will raise the degree of the surfaces described by the sub-nets by adjusting the sub-nets. The operation is equal to the operation described in *Step 4* in the previous section. If the previous operations were not executed first, the application will execute them first.

Adjust Control Points. Pressing this button will adjust the control points. This operation is equal to the one described in *Step 5* in the previous section. If the previous steps were not executed first, the application will execute them first.

Interpolate. Pressing this button will cause the application to interpolate points on the surface. The above five steps will be executed first.

By rotation in the "Subdivide" option we mean the following:
If we represent a control net as a 2-dimensional array A, with $\mathbf{b}_{030}$ at position $[0,0]$, $\mathbf{b}_{300}$ at position $[3,0]$ and $\mathbf{b}_{003}$ at position $[3,3]$, we can rotate this array by using the following formula:

$$A.[i,j] = A.[3-(i-j), 3-i]$$

The "Interpolation" button will interpolate three points by using the *de Casteljau* algorithm. These points are in the center of each edge of the triangle. The result of this action is that each triangle is split in four triangles (see figure 24).

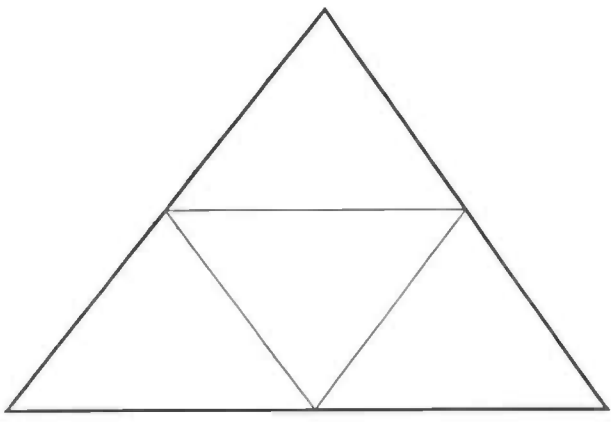


Figure 24: An example of splitting up a triangle.

The tangential data in these points are calculated by using (cross-boundary-) derivatives. If we take the derivative in two directions, we get two tangent vectors. We can calculate the normal vector of the tangent plane in these points by taking the cross product of the two tangent vectors.

The "Reset" Button
This button can be used for resetting the figure. Pressing it will give you a menu with two options. The first option, "Reset", will reset the figure to its initial state. This means that all zooming, rotating and moving is undone. The second option, "Reload" will even go one step further. This option will read the input file, and build the figure according to the data in this file.

The "Write" Button
Pressing this button will present you a tear of menu. This menu has two options, namely "Math File" and "As Input File". Pressing on of these options will cause the application to write the current figure to a file. Pressing the button labeled "Math File" will write the current figure to the Mathematica ® file that was selected in the "File" menu. Selecting the "As Input File", will cause the application to write to the file that was selected in the "File" menu

as "Output File". This file can later on be used as an input file for this application.

The "Close" Button
This button is pressed when one wishes to leave the application.

Data Structure

In our application we used the following data structure. We stored each data, data point and tangent vector of the tangent plane in this point, only once. This implies that we need lists of pointers to get a data structure we can use to interpolate surfaces. The data structure consists of three pointer lists. For each list we also administrate the length of this list.
First we have a list of points. This list contains pointers to the data. This data consists of a point and the tangent vector of the surface in this point. There are also pointers to the list of edges.

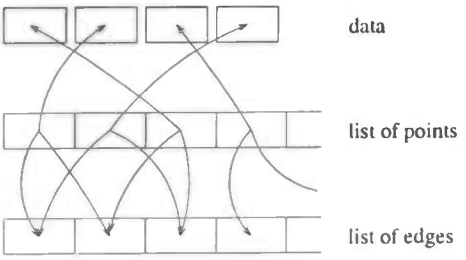


Figure 25: An example of a list of points.

These references are made so that we can find all the edges which contain this point. One point can have references to several edges.
The list of edges contains references to the list of points,

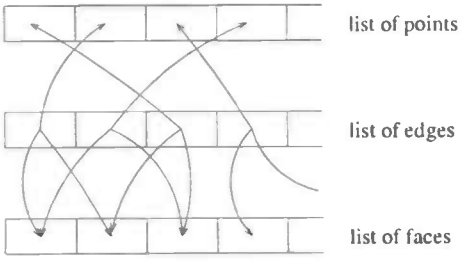


Figure 26: An example of a list of edges.

so that we know of which vertices the edge consists. The edge list has also references to the faces, or triangles, it is part of. An edge can occur in at most two faces.
The face list has references to the edges it consists of but it

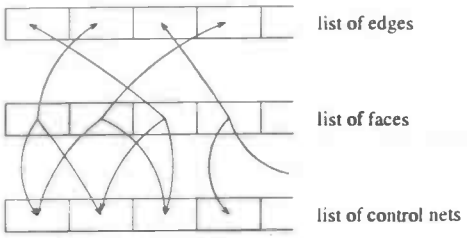


Figure 27: An example of a list of faces.

also contains references to the control net of this face. Each face has a list of three control nets. We need three control nets for each face because the subdivision algorithm we will use in our interpolation scheme, gives us three sub-nets for each triangle.

References

[1] Gerald E. Farin
Curves and Surfaces for Computer Aided Geometric Design
Second Edition, 1990, Academic Press.

[2] Gerald E. Farin
Triangular Bernstein-Bézier patches
Computer Aided Geometric Design, 3(2): pages 83–128, 1986.

[3] Bruce R. Piper
Visually Smooth Interpolation with Triangular Bézier Patches
In G. Farin, editor, *Geometric Modeling - Algorithms and New Trends*, pages 221–233, SIAM, Philadelphia, 1987.

[4] Art J. Schwartz
Subdividing Bézier Curves and Surfaces
In G. Farin, editor, *Geometric Modeling - Algorithms and New Trends*, pages 55–66, SIAM, Philadelphia, 1987.

[5] Douglas A. Young
OSF Motif® Edition, The X Window System ™
Programming and applications
SecondEdition, 1994, Prentice Hall.

[6] Xhiaochun Liu, Yun Zhu
A characterization of certain C^2 discrete triangular interpolants
Computer Aided Geometric Design, 12: pages 329–348, 1995.

[7] Thomas Jensen
Assembling Triangular and Rectangular Patches and Multivariate Splines.
In G. Farin, editor, *Geometric Modeling - Algorithms and New Trends*, pages 203–220, SIAM, Philadelphia, 1987.

[8] M. Neamtu, P.R. Pfluger
Degenerate ploynomials patches of degree 4 and 5 used for geometrically smooth interpolation in $\mathbb{R}^3$
Computer Aided Geometric Design, 11: pages 451–474, 1994.