Advanced Modeling 2

Katja Bühler, Andrej Varchola, Eduard Gröller

April 4, 2011

1 Parametric Representations

A parametric curve in $\mathbb{E}^3$ is given by

$$c : c(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}; t \in I = [a, b] \subset \mathbb{R}$$

where $x(t)$, $y(t)$ and $z(t)$ are differentiable functions in t. The vector $\dot{c}$ is called *tangent vector*. A curve point $c(t_0)$, $t_0 \in I$ is called regular if $\dot{c}(t_0) \neq o$.

A parametrization is called regular, if $\dot{c}(t) \neq o$ for all $t \in I$. Any differentable change of the parameter $\tau = \tau(t)$ does not change the curve. Moreover, if $\dot{\tau} \neq 0$ in I then $d(t) = c(\tau(t))$ is also a regular parametrization. A curve which admits a regular parametrization is called regular.

A parametric surface is given by

$$s : s(u, v) = \begin{pmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{pmatrix}, (u, v) \in [a, b] \times [c, d] = I \times J \subset \mathbb{R}^2$$

where $x(u, v)$, $y(u, v)$ and $z(u, v)$ are differentiable functions of the parameters u and v.

The lines $s(u, v_0)$ with $v_0 \in J$ fixed and $s(u_0, v)$ with $u_0 \in I$ fixed are called *isoparametric lines* of the surface (see Figure 1). The tangent plane of a surface point is defined by the two tangent vectors s_u and s_v, the surface normal vector at this point is $n = s_u \times s_v$. A surface point is called regular if $n \neq 0$.

For a good introduction to differential geometry see e.g. the book by Aumann and Spitzmueller [1]. A detailed discussion of the following topics can be found in the literature listed at the end of this summary. Almost all pictures are generated with the help of a collection of **CAGD-Java applets** written by the Geometric Design Group at the University of Karlsruhe:

http ://i33www.ira.uka.de/applets/mocca/html/noplugin/inhalt.html

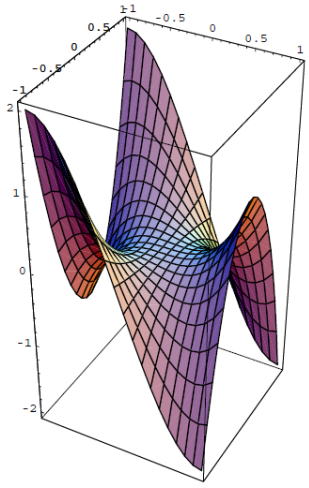

Figure 1: Surface with Isolines

2 Bézier Curves

2.1 The Constructive Approach: The de Casteljau Algorithm

First described by de Casteljau it is probably the most fundamental algorithm in the field of curve and surface design, not least because it is so easy to understand. "Its main attraction is the beautiful interplay between geometry and algebra: a very intuitive geometric construction leads to a powerful theory"[2].

The Algorithm

Given $n+1$ points $b_0, ..., b_n \in \mathbb{E}^3$ and an arbitrary $t \in \mathbb{R}$. Then

$$b_i^r = (1-t)b_i^{r-1} + tb_{i+1}^{r-1}, \quad i = r, ..., n \tag{1}$$

and $b_i^0 = b_i$, defines the point b_0^n with parameter value t on the so called *Bézier curve* $b(t)$. The points b_i are called *Bézier points*.

Equation 1 describes a repeated linear interpolation whose intermediate coefficients can be written into the triangular de Casteljau scheme (see Figure 2):

$$
\begin{array}{ccccccc}
b_0^0 & & & & & & \\
 & b_0^1 & & & & & \\
b_1^0 & & b_0^2 & & & & \\
 & b_1^1 & & b_0^3 & & & \\
b_2^0 & & b_1^2 & & & & \\
 & b_2^1 & & & & & \\
b_3^0 & & & & & &
\end{array}
$$

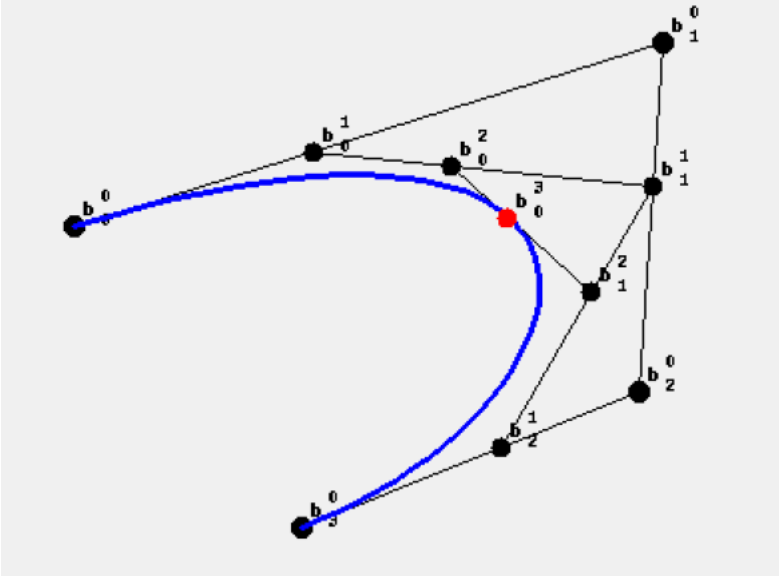

Figure 2: De Casteljau scheme

2.2 The Analytical Approach: Bézier Curves and Bernstein Polynomials

The de Casteljau algorithm gives a recursive definition of Bézier curves in terms of an algorithm. For further theoretical development it is also necessary to have an explicit parametric representation for them. Based on the de

Casteljau recursion it can be shown by mathematical induction, that a Bézier curve $b(t)$ with respect to the Bézier points $b_i,\ i = 0, ..., n$ is given by

$$b(t) = \sum_{i=1}^{n} b_i B_i^n(t)$$

where $B_i^n(t) = \binom{n}{i}(1-t)^{n-1}t^i$ are the *Bernstein Polynomials* of n-th degree with the following important properties

- partition of unity: $\sum_{i=0}^{n} B_i^n(t) \equiv 1$
- positivity: $B_i^n(t) > 0$
- recursion: $B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i+1}^{n-1}(t)$

2.3 Some Properties of Bézier Curves

- Affine invariance: An affine transformation of the Bézier points is equivalent to an affine transformation of the whole curve.
- Convex hull property: A Bézier curve lies inside the convex hull of its Bézier points.
- Endpoint interpolation: The first and last Bézier points lie on the curve.
- Linear precision: If all Bézier points lie on a straight line, the corresponding Bézier curve is identical to this line.
- Variation diminishing property: A Bézier curve has no more intersections with a plane than its Bézier polygon.

Disadvantage of Bézier curves:

- Pseudo local control: changing one of the control points changes the shape of the whole curve, although, if for instance b_i is changed, it is mostly affected around the point corresponding to the parameter value i/n, if n denotes the algebraic degree of the parametrization.
- The degree of a Bézier curve depends directly on the number of control points. Thus, higher flexibility through more control points is equivalent to a higher degree of the curve, which means higher computational cost and less control on the behavior of the curve.

2.4 Derivatives

It can be easily shown, that

- the lines b_0b_1 and $b_{n-1}b_n$ are the tangent lines at the curve points b_0 and b_n
- the intermediate points b_0^{n-1} and b_1^{n-1} of the de Casteljau algorithm determine the tangent line at the position $b_0^n(t)$.

2.5 Important Algorithms

- **Degree elevation:** Adding new control points to increase flexibility without changing the shape of the curve (see Figure 3).

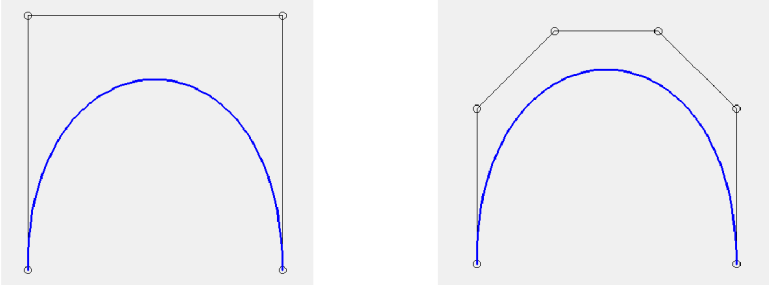

Figure 3: A curve of degree 3 displayed as curve of degree 5

- **Subdivision:** Subdivision of a Bézier curve increases its flexibility without increasing its degree and repeated subdivision converges very fast toward the curve. The subdivision algorithm is a byproduct of the de Casteljau algorithm: The two new generated sides of the triangle contain the Bézier points of the two parts of the subdivided curve (see Figure 4). For a detailed description of the algorithms see e.g. book by Hoshek and Lasser [3].

2.6 Algorithms to Evaluate Bézier Curves

Following the same idea like the Horner scheme, a Bézier curve can be written in the nested form

$$b(t) = (...((\binom{n}{0}sb_0 + \binom{n}{1}tb_1)s + \binom{n}{0}t^2b_2)s + ...)s + \binom{n}{n}t^nb_n$$

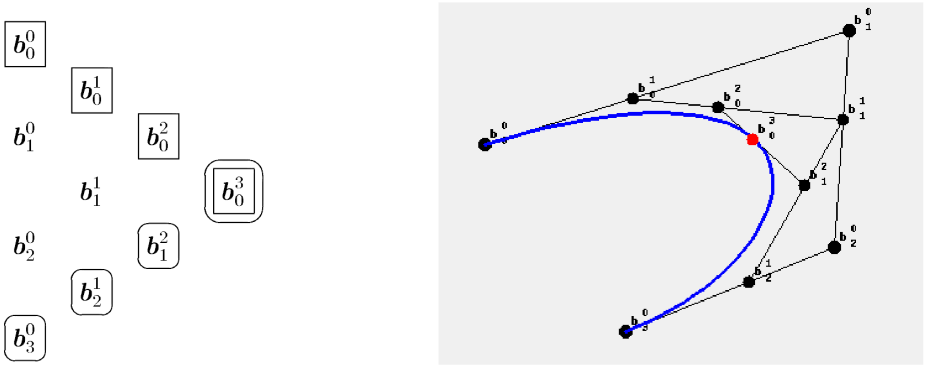

Figure 4: The subdivision algorithm with the de Casteljau scheme

with $s = 1 - t$, which leads to a more efficient algorithm.

Like mentioned above, repeated subdivision gives a good approximation for the curve.

3 Splines

The use of B-Splines to define curves and surface for CAGD was first proposed by Gordon and Riesenfeld in the early 70's, but its theory goes back to the early 19th century. B-Splines are a very common tool in CAD-Systems for the design of curves and surfaces and have two advantages over Bézier techniques:

- The degree of a B-spline polynomial can be set independently of the number of control points (with certain limitations).
- B-splines allow local control over the shape of a spline curve or surface.

3.1 B-Spline Curves

Given a set of $n+1$ control points, called *de Boor points*, $d_i, i = 0, ..., n$ and a knot vector $U = (u_0, ..., u_{n+k})$. The corresponding B-spline curve $s(u)$ is a piecewise polynomial curve of order k (of degree $k-1$) with $1 \leq k \leq n+1$ of the form

$$s(u) = \sum_{i=0}^{n} d_i N_i^k(u)$$

The $N_i^k(u)$ are the normalized B-spline basis functions of order k with respect to U with the following recursive definition (see Figure 5):

$$N_i^0(u) = \begin{cases} 1 & \text{if } u \in [u_i, u_{i+1}), \\ 0 & \text{else.} \end{cases}$$

$$N_i^r(u) = \frac{u - u_i}{u_{i+r-1} - u_i} N_i^{r-1}(u) + \frac{u_{i+r} - u}{u_{i+r} - u_{i+1}} N_{i+1}^{r-1}(u); \quad r = 1, ..., k$$

Properties of the basis functions:

- partition of unity: $\sum_{i=0}^{n} N_i^k(u) \equiv 1$
- positivity: $N_i^k(u) > 0$
- local support $N_i^k(u) = 0$ if $u \notin [u_i, u_{i+k})$

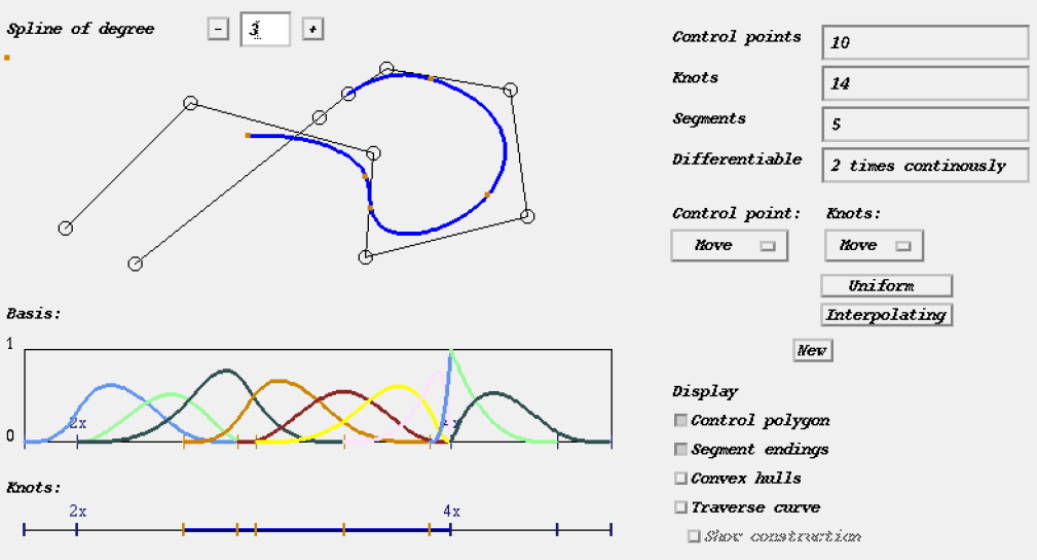

Figure 5: A B-spline curve with all related information

3.2 Some Properties of B-Spline Curves

- Affine invariance

- Strong convex hull property: The curve segment corresponding to the parameter values $[u_i; u_{i+1})$ lies inside the convex hull of the control points $d_{i-k}, ..., d_i$.
- Variation diminishing property.
- Local support: Moving d_i changes $s(u)$ only in the parameter interval $[u_i; u_{i+1})$.
- Multiple knot points $u_i = ... = u_{i+s}$ are possible. If $s \geq k$ the curve goes through the control point d_i. Furthermore if $n = k - 1$ and $U = (u_0, ..., u_0, u_1, ..., u_1)$ then $s(u)$ is a Bézier curve.
- Differentiability: $s(u)$ is $k - l - 1$ times differentiable at a knot u_i, if u_i is of multiplicity $l \geq 1$.

3.3 Some Special Types of B-Spline Curves

The form of the knot vector determines three special cases of B-spline curves (see Figure 6):

- open: $u_0 = ... = u_{k-1}$ and $u_{n+1} = ... = u_{n+k}$
- closed: $d_{n+1} = d_0$, $d_{n+2} = d_1$, ...
- uniform: If the spacing between knot values is constant, the resulting curve is called a uniform B-spline curve. Uniform B-spline curves have periodic basis functions and therefore many algorithms have a simpler and more effective implementation.

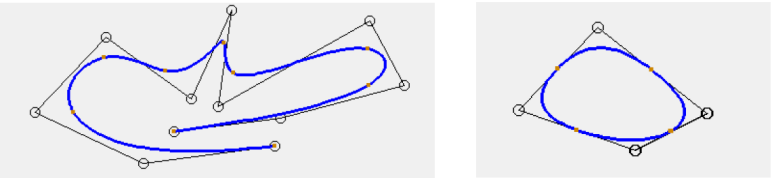

Figure 6: An open and a closed B-spline curve.

3.4 Evaluating B-Spline Curves

3.4.1 The de Boor algorithm

The de Boor algorithm is a generalized de Casteljau algorithm and works following the same principles as linear interpolation. To evaluate $s(u)$ at $u = \bar{u}, \bar{u} \in [u_l; u_{l+1})$ the following recursion has to be done:

$$d_i^r = (1 - \alpha_i^r) d_{i-1}^{r-1} + \alpha_i^r d_i^{r-1}, \quad i = l - k + 1, ..., l; \quad r = 0, ..., k - 1$$

with

$$\alpha_i^r = \frac{\bar{u} - u_i}{u_{i+k-r} - u_i}$$

where $d_i^0 = d_i$ and $s(\bar{u}) = d_l^{k-1}$. The de Boor scheme has the same form like the de Casteljau scheme. The de Boor algorithm allows the evaluation of the curve, without any knowledge of the basis functions and proposes an effective method.

3.4.2 The direct evaluation

A second possibility to evaluate a B-Spline function for a parameter value $\bar{u}$ is given by the following algorithm.

1. Find the knot span $[u_i, u_{i+1})$ in which $\bar{u}$ lies.
2. Compute the non zero basis functions.
3. Multiply the values of the nonzero basis functions with the corresponding control points.

3.5 Algorithms

Knot insertion and knot refinement

The *knot insertion* algorithm inserts a knot one or multiple times into the knot vector (see Figure 7). Knot insertion does not change the shape of the curve, but refines its segmentation. This algorithm is used to increase the flexibility of a curve, to compute derivatives, to split curves and to evaluate a curve for a certain parameter value: The de Boor algorithm is a repeated knot insertion algorithm that inserts this parameter value $k + 1$ times into the knot vector. There exist special algorithms that insert several different knots simultaneously *(knot refinement)*.

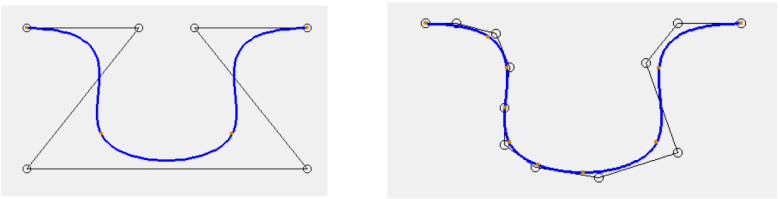

Figure 7: An example for knot refinement.

Degree elevation:

Adapts curve degrees without changing the shape to build combined structures, like tensor product surfaces or to connect curves and surfaces. For a detailed description of the algorithms see e.g. book by Piegel and Tiller[4].

4 Rational Curves

Although polynomials offer a lot of advantages, there exist a number of important curve and surface types which cannot be represented precisely using polynomials, conic sections and quadrics that have a rationally parametrization. In general a rational parametrized curve has the form:

$$c(u) = \begin{pmatrix} \frac{x(u)}{w(u)} \\ \frac{y(u)}{w(u)} \\ \frac{z(u)}{w(u)} \end{pmatrix}$$

A more elegant and very useful representation is the one using homogeneous coordinates: the curve c is represented as a polynomial curve in $\mathbb{E}^4$.

$$c(u) = \begin{pmatrix} w(u) \\ x(u) \\ y(u) \\ z(u) \end{pmatrix}$$

The original curve has to be interpreted as a projection of this curve onto the hyperplane $w(u) = 1$ of $\mathbb{E}^4$. A homogeneous representation $p = (w, x, y, z)^T$ of a point can be converted back to the euclidean representation in the following way: $p = (x/w; y/w; z/w)^T$.

4.1 Rational Bézier Curves

A rational Bézier curve is defined as

$$b(u) = \frac{\sum_{i=0}^{n} w_i b_i B_i^n(u)}{\sum_{i=0}^{n} w_i B_i^n(u)}$$

The $w_i, i = 0, ..., n$ are called weights and are assumed to be positive. If all $w_i = 1$, $b(u)$ denotes a polynomial Bézier curve. Writing $b(u)$ in terms of homogeneous coordinates yields the following representation:

$$b(u) = \sum_{i=0}^{n} b_i B_i^n(u)$$

with the *homogeneous Bézier points* $b_i = (w_i, w_i b_i^T)^T$.

Properties Rational Bézier curves have the same properties as non-rational ones, but they

- are even projective invariant,
- do not lie inside the control polygon if negative weights are allowed, and
- have the weights as additional design parameter: increasing the weight w_i causes an attraction of the curve towards the Bézier point b_i.

Algorithms All algorithms for polynomial Bézier curves can be applied in the same way to the homogeneous representation of a rational Bézier curve.

4.2 Non Uniform Rational B-Spline Curves (NURBS)

NURBS are the most important and flexible design elements provided in CAD systems. Polynomial and rational Bézier curves and B-spline curves are subsets of NURBS. A NURBS with respect to the control points $d_0, .., d_n$ and the knot vector $U = (u_0, ..., u_{n+k})$ is defined as

$$n(u) = \frac{\sum_{i=0}^{n} w_i d_i N_i^n(u)}{\sum_{i=0}^{n} w_i N_i^n(u)}$$

The $w_i, i = 0, ..., n$ are called weights and are assumed to be positive. If all $w_i = 1$, $n(u)$ denotes a polynomial B-Spline curve. Writing $n(u)$ in terms of

homogeneous coordinates yields the following representation:

$$n(u) = \sum_{i=0}^{n} d_i N_i^k(u)$$

with the *homogeneous control points* $d_i = (w_i, w_i n_i^T)^T$.

Properties: NURBS have the same properties as polynomial B-spline curves, but they

- are even projective invariant
- do not lie inside the control polygon if negative weights are allowed and
- have the weights as additional design parameter: changing the weight w_i affects only the curve in the interval $[u_i, u_{i+k})$

Algorithms All algorithms for polynomial Bézier curves can be applied without any change to the homogeneous representation of a NURBS curve.

5 Surfaces

5.1 Tensor Product Surfaces

A surface is the locus of a curve that is moving through space and thereby changing its shape. Let

$$f(u) = \sum_{i=0}^{n} c_i F_i(u)$$

be a curve in $\mathbb{E}^3$ with $F_i(u), i = 0, ..., n$ as basis functions (e.g., Bernstein polynomials or B-spline basis functions). Moving f through space while deforming it, is equivalent to continuously changing the control points c_i which can be described by

$$c_i(v) = \sum_{j=0}^{m} a_{ij} G_j(v)$$

where the $G_j(v), j = 0, ..., m$ are basis functions too. Combining both equations yields the definition of a tensor product surface:

$$s(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} a_{ij} F_i(u) G_j(v).$$

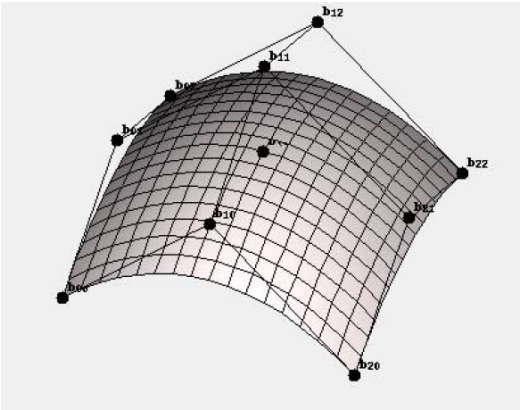

Figure 8: A Bézier tensor product surface

Bézier Surfaces

A tensor product Bézier surface (see Figure 8) is given by

$$b(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{ij} B_i^m(u) B_j^n(v).$$

The Bézier points b_{ij} form the *control net* of the surface. Tensor product Bézier surfaces have properties analogue to that of Bézier curves. All algorithms for Bézier curves can be applied in two steps to the surface $b(u,v)$:

1. Apply algorithm on the curves $b_i(v) = \sum_{j=0}^{n} b_{ij} B_j^n(v)$.
2. Apply algorithm on the curves $b(u,v) = \sum_{i=0}^{m} b_i(v) B_i^m(u)$.

B-Spline Surfaces

A tensor product B-spline surface with respect to the control points d_{ij} and the knot vectors $U = (u_0, ..., u_{m+k})$ and $V = (v_0, ..., v_{n+l})$ is given by

$$s(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} d_{ij} N_i^k(u) B_j^l(v).$$

Tensor product B-spline surfaces have properties analogue to that of Bézier curves. All algorithms for B-spline curves can be applied in the same two steps to the surface $s(u,v)$ like the algorithms for Bézier curves on tensor product Bézier surfaces.

5.2 Bézier Triangles

A triangular Bézier patch is defined by

$$b_p(u,v) = \sum_{i+j+k=n} b_{i,j,k} B^n_{i,j,k}(u,v,w)$$

where $i,j,k \geq 0$ and u,v,w are barycentric coordinates of the triangular parameter domain. The $B^n_{i,j,k}$ are generalized Bernstein polynomials of degree n:

$$B^n_{i,j,k}(u,v,w) = \frac{n!}{i!j!k!} u^i v^j w^k$$

The Bézier net of the surface is formed by the $1/2(n+1)(n+2)$ Bézier points $b_{i,j,k}$ (see Figure 9). The triangular Bézier patch inherits many properties from the univariate Bézier curve.

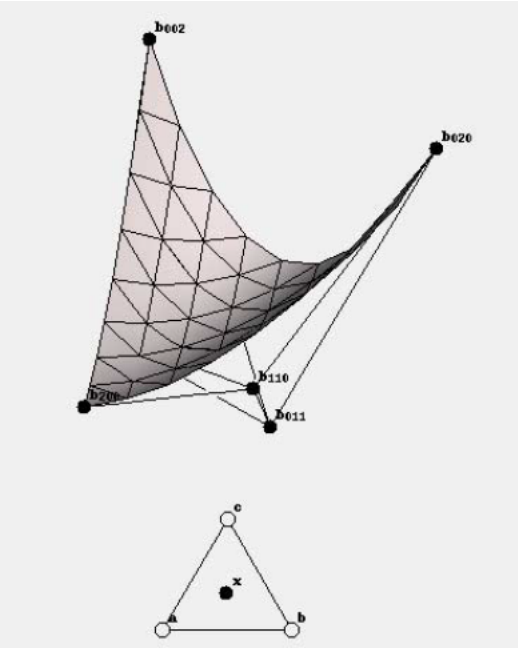

Figure 9: An elliptic paraboloid as Bézier triangle

5.3 Algorithms

- The de Casteljau algorithm for triangular patches produces a tetrahedral scheme. The recursion formula for the computation is:

$$b^l_{ijk} = ub^{l-1}_{i-1jk} + vb^{l-1}_{ij-1k} + wb^{l-1}_{ijk-1}$$

  where $i + j + k = n - l, (i, j, k \geq 0)$ and $b^0_{ijk} = b_{ijk}$. It is easy to show that $b_p(u, v, w) = b^n_{000}$.

- Degree elevation

- Subdivision: the subdivision of the patch into three subpatches can be derived from the de Casteljau algorithm, like in the univariate case.

6 Subdivision Surfaces

Subdivision surfaces are polygon-mesh surfaces generated from a base mesh through an iterative process that smooths the mesh while increasing its density (see Figure 10). It is convenient to represent subdivision surfaces as functions defined on some parametric domain with values in $\mathbb{R}^3$. A simple

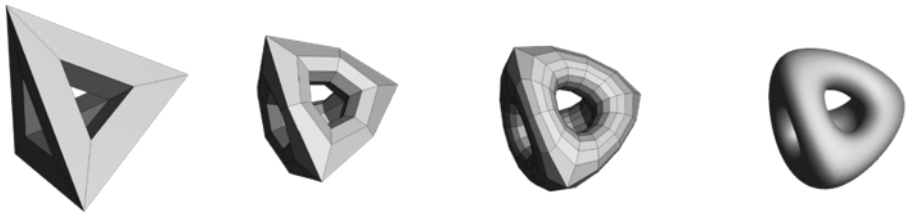

Figure 10: Illustration of the progressive smoothing of the mesh by a subdivision scheme [5].

construction allows to use the *initial control mesh*, as the domain for the surface. Complex smooth surfaces can be derived in a predictable way from simple initial control meshes. The subdivision surfaces are defined recursively. The process starts with a given initial control mesh. In each iteration, the

subdivision process produces a increasing number of polygons [6]. The mesh vertices converge to a limit surface through iteration step. Every subdivision algorithm has rules to calculate the locations of new vertices. Subdivision representations are suitable for many numerical simulation tasks which are of importance in engineering and computer animation.

A *vertex* v is a 3D position which describes the corners or intersections of geometric shapes. For example a triangle has three vertices. An *edge* is a one-dimensional line segment connecting two adjacent zero-dimensional vertices in a polygon. A planar closed sequence of edges forms a *polygon* (*face*). A *polyhedron* is a geometric solid in three dimensions with flat faces and straight edges. The *valence* of a vertex is the number of edges at the vertex. Subdivision rules are often specified by providing a *mask*. The mask is a visual scheme showing which vertices and weights are used to compute a new vertex.

6.1 The Loop Scheme

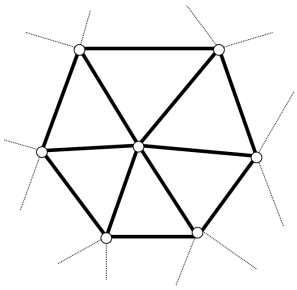

Figure 11: Initial triangle mesh before the Loop subdivision scheme [5].

One of the simplest subdivision schemes is the one, invented by Charles Loop. The Loop scheme defines the subdivision process for triangle meshes only (see Figure 11). At each step of the scheme, each triangle is split into four smaller triangles.

First, an edge vertex v_e is constructed on each edge (see Figure 12). This edge vertex is three eighths of the sum of the two vertices of the edge (v_1,v_2) plus one eighth of the sum of the two other vertices (v_3,v_4) that form the two triangles that share the edge in question:

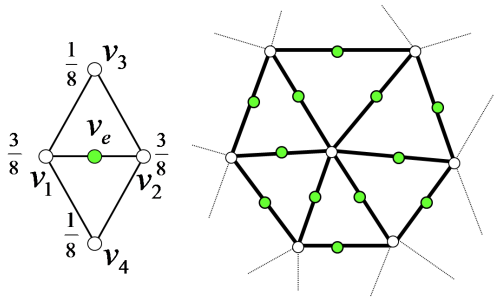

Figure 12: Left: mask for the construction of edge vertices with the Loop subdivision. Right: edge vertices constructed on each edge of polygons from the initial control mesh [5].

$$v_e = \frac{3}{8}v_1 + \frac{3}{8}v_2 + \frac{1}{8}v_3 + \frac{1}{8}v_4$$

Second, a new vertex v_{new} is constructed for each old vertex v_{old} (see Figure 13). If a given vertex has n neighbor vertices v_j, the new vertex point is calculated as:

$$v_{new} = (1 - n\beta)v_{old} + \beta \sum_{j=1}^{n} v_j,$$

where β is a scaling factor. Smoothness considerations lead to specific values of β. For $n = 3$, β is 3/16. For $n > 3$:

$$\beta = \frac{1}{n}(\frac{5}{8} - (\frac{3}{8} + \frac{2}{8}\cos\frac{2\pi}{n})^2).$$

Finally, each old triangle has three edge vertices, one for each edge, and three new vertices, one for each old vertex. To form the new triangles these points are then connected, creating four triangles (see Figure 14).

The cube in Figure 15 has been tessellated into triangles with a vertex in the center of each original cube face. Four triangles after each subdivision step correspond to each triangle in the prior level of subdivision.

6.2 The Catmull-Clark Scheme

The rules of the Catmull-Clark scheme are defined for meshes with quadrilateral faces (see Figure 16).First, a face vertex v_f is created for each old polygon

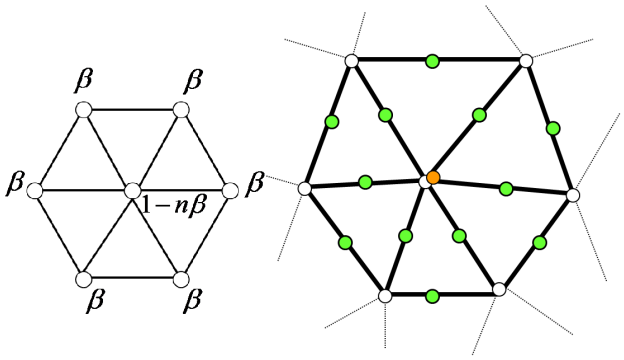

Figure 13: Left: mask for the construction of a new vertex with the Loop subdivision. Right: a new vertex is constructed from each old vertex and the weighted sum of neighboring vertices [5].

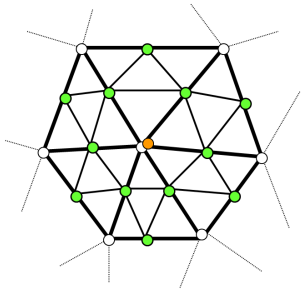

Figure 14: Final reconnection of the new triangles in the Loop subdivision [5].

(see Figure 17), defined as the average of every vertex in the polygon:

$$v_f = \frac{1}{4}\sum_{i=1}^{4} v_i$$

Second, an edge vertex v_e is created for each old edge (see Figure 18), defined as the average of the edge vertices and the face vertices for the polygons that adjoin the old edge:

$$v_e = \frac{v_1 + v_2 + v_{f1} + v_{f2}}{4}$$

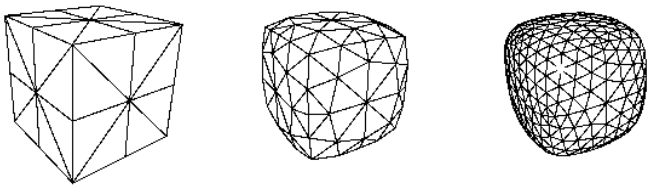

Figure 15: The Loop subdivision of the tesselated cube with two successive steps [7].

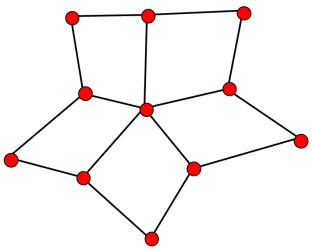

Figure 16: Initial quadrilateral mesh before the Catmull-Clark subdivision scheme [5].

Finally, new vertices are defined (see Figure 19). For each old vertex v_{old}, there are n polygons sharing it. The new vertex v_{new} is given as:

$$v_{new} = \frac{Q}{n} + \frac{2R}{n} + \frac{v_{old}(n-3)}{n}.$$

Q is the average of the adjacent face vertices. R is the average of the adjacent edge vertices. The new vertices are then connected to produce a finer grid of quadrilaterals (see Figure 19). Each face of the cube in the Figure 20 has been divided into four quadrilaterals.

6.3 Classification of Subdivision Schemes

Subdivision schemes can be classified based on:

- the type of refinement rule
  - vertex insertion (Loop, Catmull-Clark, Modified Butterfly, Kobbelt)

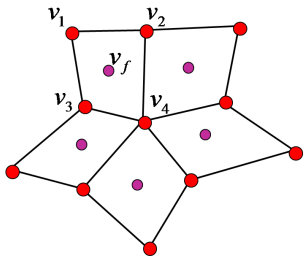

Figure 17: Face vertices are created for each polygon in the first step of the Catmull-Clark scheme [5].

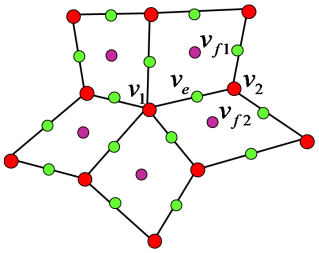

Figure 18: Edge vertices are created for each old edge in the second step of the Catmull-Clark scheme [5].

  - corner-cutting (Doo-Sabin, Midedge)

- the type of generated mesh
  - triangular (Loop, Modified Butterfly)
  - quadrilateral (Catmull-Clark, Kobbelt)
- whether the scheme is:
  - approximating (Loop, Catmull-Clark)
  - interpolating (Modified Butterfly, Kobbelt)

There are two main approaches that are used to generate a refined mesh: one is vertex insertion and the other is corner cutting. For vertex insertion, each edge of a triangular or a quadrilateral mesh is split into two old vertices of the mesh are retained, and new vertices inserted on edges are connected. For quadrilaterals, an additional vertex is inserted for each face. For corner

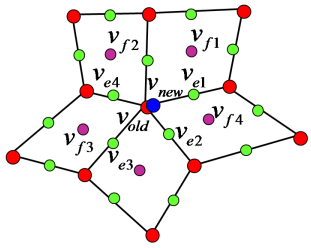

Figure 19: New vertex (blue point) is created close to, but usually not precisely at, the old vertex [5].

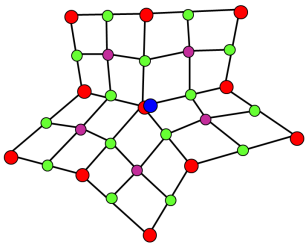

Figure 20: In the last step of the Catmull-Clark scheme, the new vertices are connected [5].

cutting, for each old face, a new similar face is created inside of it and the newly created faces are connected. As a result, we get four new vertices for each old edge, a new face for each edge and each vertex. The old vertices are discarded.

Subdivision schemes can be classified based on the type of polygon they operate on. Some schemes work for quadrilaterals (quads), while others operate on triangles.

Interpolating schemes are required to match the original position of vertices in the original mesh. Approximating schemes do not match the original positions and they can and will adjust these positions as needed. In general, approximating schemes have greater smoothness. Examples with different subdivision schemes for simple initial meshes are shown on the Figure 21.

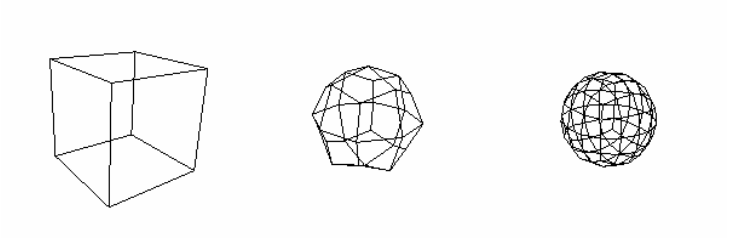

Figure 21: The Catmull-Clark subdivision of the cube with two successive steps [7].

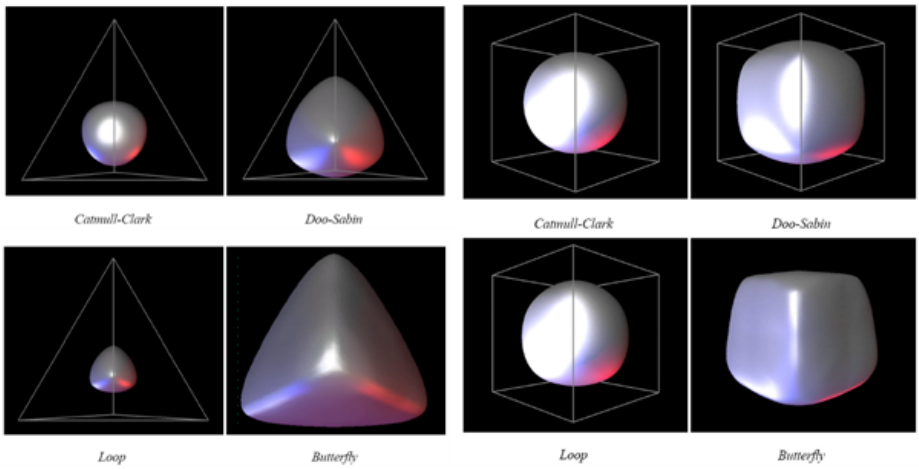

Figure 22: Examples with different subdivision schemes [5].

References

[1] G. Aumann and K. Spitzmüller. *Computerorientierte Geometrie.* BI-Wissenschaftsverlag, Mannheim, 1993.

[2] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Academic Press, 1993.

[3] J. Hoschek and D. Lasser. *Grundlagen der geometrischen Datenverarbeitung.* B.G. Teubner Verlag, 1997.

[4] L. Piegl and W. Tiller. *The NURBS book (2nd ed.).* Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[5] O. Weber. Subdivision Surfaces. http://www.cs.technion.ac.il/~cs236716/, 2011. [Online; accessed 04-March-2011].

[6] D. Zorin, P. Schröder, T. Derose, L. Kobbelt, A. Levin, and W. Sweldens. Subdivision for Modeling and Animation. In *SIGGRAPH Course Notes*, New York, 2000. ACM.

[7] R. Holmes. A Quick Introduction to Subdivision Surfaces. http://www.holmes3d.net/graphics/subdivision/, 2011. [Online; accessed 04-March-2011].