transnationale Universiteit Limburg
School voor Informatietechnologie

"Nieuwe technieken voor de modellering van subdivisie-oppervlakken"

"New approaches in the modeling of subdivision surfaces"

Proefschrift voorgelegd tot het behalen van de graad van

doctor in de wetenschappen, richting informatica

aan de transnationale Universiteit Limburg te verdedigen door

Johan CLAES

Promotor : Prof. dr. E. Flerackers

Co-promotor : Prof. dr. F. Van Reeth

2001

Preface

This dissertation would not have been possible without the help of many.

First of all, I wish to express my gratitude to my thesis supervisors, Eddy Flerackers and Frank Van Reeth, for their continuing support, guidance, encouragement and friendship during the preparation of this work. Frank's introduction to the exciting world of subdivision surfaces allowed me to combine my interests for computational geometry with the wonderful world of computer animation.

Koen Beets, Fabian Di Fiore, Jan Van den Bergh and Marc Ramaekers successfully implemented many of the ideas presented in this dissertation into working prototypes. I enjoyed discussing new ideas with them and the search for solutions to tricky implementation problems.

Without the help of the graphical designers, especially Joan Cabot, but also Paul Akkermans, Bart "Pokke" Van Bael and Luis Gutierrez, our publications would have been much less lively. They also pointed out important user interface issues and discussed potential extensions and improvements.

Special words of thank also have to go to Peter Vandoren for kindly helping with many layout issues and to Mike Staples for carefully checking my use of the English language.

Certainly, I need to thank the institutions helping to fund our research, especially the European TMR (Training and Mobility of Researchers) project PAVR (Platform for Animation and Virtual Reality), the EFRD (European Fund for Regional Development) and the Flemish Government. This funding further allowed me to travel to conferences and personally meet other researchers, which led to motivating international contacts and warm friendships.

Last, but not least, I have to thank my family and close friends for their warm support during this time of hard work. I have been especially fortunate in having the care and affection of my girlfriend Maribel to help me through it – *nunca hay demasiado amor*.

Abstract

Recursive subdivision surfaces allow considerable freedom in designing surfaces of arbitrary topology. Tools to manipulate them, however, are still not as powerful as existing tools for longer established modeling paradigms, such as B-spline surfaces. Furthermore, the most well-behaved and most widely used schemes are approximating schemes, which do not interpolate their initial control points.

This dissertation describes a new modeling paradigm, providing the possibility of locally choosing an interpolating variant of the conventionally approximating subdivision scheme. Our approach combines the advantages of approximating schemes with the precise control of interpolating schemes. Unlike other solutions that mostly focus on locally changing the weighting factors of the subdivision scheme, we keep the underlying uniform scheme intact. Our method is based upon introducing additional control points on well-chosen locations, with optional interactive user control over the tangent plane (or surface normal) and the tension of the surface near the interpolating control points.

The same techniques used for surface modeling and editing are also adapted to implement a versatile free-form deformation tool, especially designed for 2D textured objects. Based on subdivision surfaces applied in 2D, our method successfully combines the following features: fluid good-looking movement, both general global and precise local control and explicit discontinuities.

As a different item of interest, we noticed a lack in the current range of subdivision surface schemes. Quadrilateral schemes are organized logically as primal and dual schemes, but for triangular subdivision, only primal schemes are described in the literature. This is a pity, as recently research papers have been published showing that primal and dual schemes can be successfully combined to create surfaces with an arbitrarily high degree of continuity. Therefore, we introduce a new hexagonal scheme, opening a fascinating range of possibilities.

Table of contents

1 Introduction

In this dissertation, new techniques are developed for the modeling of subdivision surfaces. Although the theories behind recursive subdivision surface schemes have been around for more than 20 years, only recently have they begun to get full attention. An important factor in the increase of their popularity was Pixar's very successful experience in 1998 with subdivision surfaces in the creation of the character of their short animations, Geri's Game [DeRose98] and later Toy Story 2 [Porter00].

For modeling surfaces such as the ones used in character animation, there exist many good reasons to employ the subdivision paradigm. Subdivision schemes use simple rules to generate high-quality surfaces from coarse polygonal models. Unlike most competing methods for generating surfaces, they allow surfaces of arbitrary topology to be created using one single consistent paradigm. There is no need to stitch together different surface parts. This makes animating these surfaces much easier, as there is no fear of breaking the borders where patches are stitched together.

The most important implication of allowing arbitrary topologies is not the creation of surfaces containing holes, but the ability to vary the density of control points over the surface. This permits the creation of small details and bodily limbs without the obligation to add numerous control points.

Also important for subdivision surfaces is its extensive mathematical background, with important links to wavelet theory and multi-resolution analysis, which have proven their usefulness in many scientific fields. The divide-and-conquer approach, furthermore, allows for many applications in the field of simulating physical processes.

Besides their specific usefulness for modeling surfaces used in animation, as well as in engineering applications, subdivision surfaces can easily be

extended to allow for creases and sharp and semi-sharp edges. Due to these extensions, subdivision surfaces are starting to become commonplace in professional animation packages, such as Alias|Wavefront's Maya [Maya01].

The main contributions of our research in the field of recursive subdivision schemes are the following:

- a local interpolation tool for subdivision curves of any degree; this tool, furthermore, allows for control of the tangent line and a useful tension parameter;
- an extension of this tool for subdivision surfaces, with specific algorithms for its optimal use in the Catmull-Clark and Loop subdivision schemes [Catmu78, Loop87];
- an application of these techniques outside the world of surface modeling, where subdivision surfaces are used in 2D as a base for free-form deformations to fluently manipulate 2D animation objects;
- and the introduction of a new subdivision surface scheme, based on hexagonal meshes.

In chapter 2, we start by explaining the paradigm of subdivision curves. Algorithms for curves are easier to understand and analyze, so they form a good introduction for studying their generalization to subdivision surfaces. Properties of interpolatory and approximate schemes are discussed and compared.

The concept of subdivision surface schemes is introduced in chapter 3, where we explain the usual taxonomy of subdivision surface schemes, which is still in use at prominent conferences [Zorin00a, Hubeli01]. Then we show that this taxonomy lacks, in our opinion, an important class of schemes, namely hexagonal ones. Hexagonal schemes are important, as they form the dual of triangular schemes. The duality between subdivision surface schemes has been proven very useful in recent papers [Zorin01a, Stam01], which show how to combine dual quadrilateral schemes to create surfaces with an arbitrarily high degree of continuity. Chapter 3 also contains an extensive list of existing subdivision schemes, each with its specific properties. This forms the base for studying the properties of our own new hexagonal scheme, which is postponed to chapter 5.

An extensive list of properties and practical applications of subdivision surfaces form the main ingredients of chapter 4, while chapter 5 introduces and analyzes our new hexagonal subdivision scheme.

In chapter 6, we describe our extension to allow for local interpolation on otherwise approximate subdivision curves of any degree. This extension, furthermore, allows for a handy tension parameter and control of the curve normal.

Chapter 7 forms an introduction to the subsequent chapters, outlining the need for local interpolation for subdivision surfaces and investigating related work.

In chapter 8, we show how this local interpolation technique can be implemented for the quadrilateral Catmull-Clark subdivision scheme. Also, different methods for setting up and arranging the newly introduced ghost points are discussed. It is further shown that the new techniques can be combined successfully with recent extensions like semi-sharp edges.

A similar technique for the triangular Loop scheme is explained in chapter 9, where we further employ the results of chapter 6 to create locally interpolating boundaries. This leads to a very versatile modeling tool, allowing for an intuitive control over surface normal and a convenient tension parameter.

The techniques of chapter 9 can be applied in a totally different field, which is elaborated in chapter 10. A 2D representation of locally interpolating subdivision surfaces is used to create a flexible free-form deformation tool.

In chapter 11, we discuss potential directions to extend our research, which is an ongoing process. This is followed by our general conclusions in chapter 12. This dissertation ends with a bibliography, lists of figures and tables, and two appendices containing the proofs behind the techniques described in chapter 6.

2 Subdivision curves

2.1 Introduction

In this chapter, we explain the concept of subdivision curves. Studying their properties usually is much simpler than for surfaces, making it easier to gain insights that later can be generalized to subdivision surfaces. Moreover, many subdivision surfaces schemes are directly or indirectly based on subdivision curve schemes. Therefore, this chapter is not only an introduction to chapter 6, where we explain a local interpolation technique for subdivision curves, but also for all the other chapters concerning subdivision surfaces.

In the context of this chapter (and also chapter 6), recursive subdivision is the process of repeatedly refining an initial control polygon P^0 in order to produce a sequence of increasingly more refined polygons P^0, P^1, P^2, P^3, … hence approaching a limit polygon, actually a curve:

$$P = \lim_{j \to \infty} P^j$$

In [Stoll96], it is elucidated that the subdivision process can be viewed as a two-step process of *splitting* and *averaging*. Given a control polygon P^j at level j in the subdivision process, the splitting step generates an intermediate control polygon $\bar{P}^{j+1}$ that contains all the control points of P^j, as well as additional control points inserted at the midpoints of all the edges constituting P^j. This narrows down to:

$$\bar{c}_{2i}^{j+1} = c_i^j \text{ (vertex split point)}$$

$$\text{and} \qquad \bar{c}_{2i+1}^{j+1} = \tfrac{1}{2}(c_i^j + c_{i+1}^j) \text{ (edge split point)} \tag{2-1}$$

In order to get the final positions of the control points c_i^{j+1} in P^{j+1}, the intermediate control points $\widehat{c}_i^{\,j+1}$ in $\widehat{P}^{\,j+1}$ are averaged using a so-called *averaging mask* $r = (r_k)_{-m \le k \le m}$ (the exact meaning/size of m is not important right now, it will be given in section 6.3):

$$c_i^{j+1} = \sum_{k=-m}^{m} r_k \widehat{c}_{i+k}^{\,j+1} \tag{2-2}$$

In Chaikin's algorithm [Chaik74] the averaging mask is r = ½ (0, 1, 1). In cases where the averaging mask remains the same along the curve, the scheme is called a uniform subdivision scheme. It is called a stationary scheme in cases where the same mask is used in each subdivision level. In this dissertation, the focus is on uniform and stationary subdivision schemes.

When reconstructing the history of subdivision schemes for curves and surfaces, it can be noted that most researchers refer back to Chaikin's 1974 paper. However, it turns out that he was not the first one to publish about this topic. Already in 1956, G. de Rham, a French mathematician, published about recursively corner cutting a piecewise linear approximation to obtain a smooth curve [DeRham56, Dubuc98]. [Sabin01] describes some interesting thoughts about de Rham's algorithm.

In section 2.2, we show how this formulation is extended to B-splines of any degree, and in section 2.3 we discuss an example of an interpolatory scheme. In section 2.4, we compare approximating and interpolatory schemes. Finally, in section 2.5, we demonstrate how eigenanalysis supports the study of the limit behavior of subdivision curves.

2.2 Extension to higher degree B-splines

Riesenfeld [Riese75] was able to show that the curves generated by Chaikin's algorithm are uniform quadratic B-splines. It is proven by Lane and Riesenfeld

[Lane80] that Chaikin's algorithm can be generalized to generate uniform B-splines of degree *n+1* by using an averaging mask with entries

$$\frac{1}{2^n}\left(\binom{n}{0},\binom{n}{1},\ldots,\binom{n}{n-1},\binom{n}{n}\right). \tag{2-3}$$

Thus, in case $n = 2$, the averaging mask $r = \frac{1}{4}(r_{-1}, r_0, r_1) = \frac{1}{4}(1, 2, 1)$ results in a cubic B-spline as limit curve.

This is a so-called approximating subdivision scheme, since the limit curve generated by these averaging masks in general does not interpolate the vertices c_i^0 of the initial control polygon P^0. The first smooth interpolating subdivision curve scheme was presented by Dyn, Levin and Gregory in 1987 [Dyn87]. This scheme is described in more detail in section 2.3.

A very interesting observation made by Lane and Riesenfeld, was that in order to calculate a refinement step for a degree *n+1* B-spline, it is not necessary to collect all *n+1* control points from the previous subdivision level. The same result can be obtained by *n* subsequent average steps of just two control vertices. For subdivision curves, this does not seem to be too useful, but when the scheme is used as a base for a subdivision surface, this very local behavior proves to be very powerful. For subdivision surfaces it is far from clear how to collect further away neighbor points, as the mesh for the surface can contain multiple extraordinary vertices, where the regular spline pattern is lost. Both Jos Stam [Stam01] and Zorin and Schröder [Zorin01a] came to the conclusion that subdivision surface schemes that were first developed to generalize lower degree curves, could be extended to higher degree surfaces, by applying this repeated averaging technique.

In figures 2-1 through 2-4, an example of the generation of a cubic B-Spline via subdivision is shown. Starting from the eight vertices of the initial control polygon of figure 2-1, the first subdivision step adds a new vertex in the center of each edge (figure 2-2). As the averaging mask is $\frac{1}{4}(1, 2, 1)$ the newly added vertices stay on their original position, while the old vertices are relaxed towards the mean of the two new surrounding vertices. Figure 2-3 shows one subdivision step further, while figure 2-4 shows the resulting limit curve.

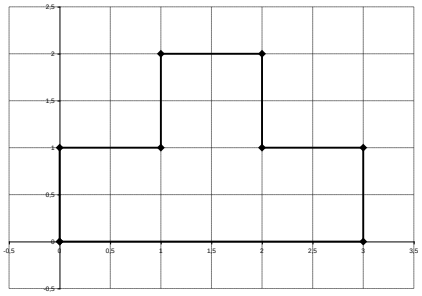


Fig. 2-1. The original control polygon.

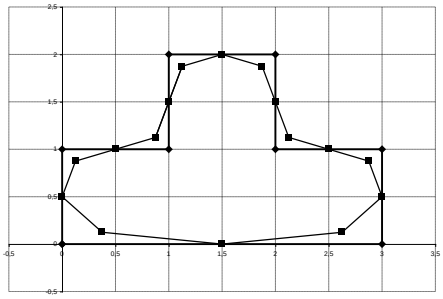


Fig. 2-2. One subdivision step for the cubic B-spline scheme.

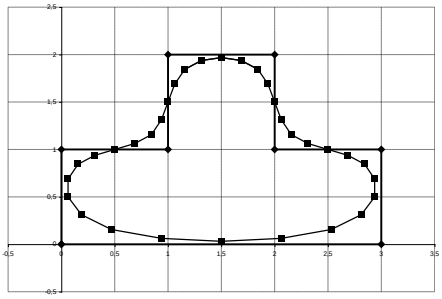


Fig. 2-3. Two subdivision steps for the cubic B-spline scheme.

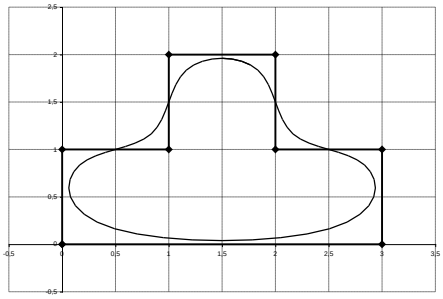


Fig. 2-4. The limit curve for the cubic B-spline scheme.

Another observation from Lane and Riesenfeld was that it is not necessary that the subdivision process is restricted to divisions by a factor of two. In fact, the same principles can be used to generate B-splines by any integer number of splitting steps. For example, for a ternary refinement of a quadratic curve, [Sabin01] derived the coefficients of equation 2-4. The first formula relaxes the position of an old control point, while the second and third define the position of the newly inserted vertices.

$$p_0' = \frac{1}{9}\left(p_{-1} + 7p_0 + p_1\right)$$

$$p_1' = \frac{1}{9}\left(6p_0 + 3p_1\right) \qquad (2\text{-}4)$$

$$p_2' = \frac{1}{9}\left(3p_0 + 6p_1\right)$$

In figure 2-5 an example of this ternary scheme is shown, while figure 2-6 shows the resulting quadratic B-spline. In [Kobbe00] a ternary subdivision scheme for cubic curves is used to serve as a border for his Sqrt(3) subdivision surface scheme. In our research about a hexagonal subdivision surface scheme, we used a ternary scheme for a quadratic curve. We refer to chapter 5 for more details about this new scheme for subdivision surfaces.

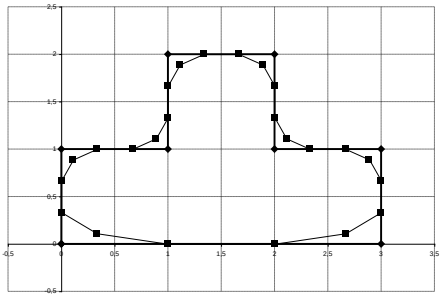


Fig. 2-5. The first subdivision step of the quadratic B-spline scheme, each time dividing the original edges by a factor of three.

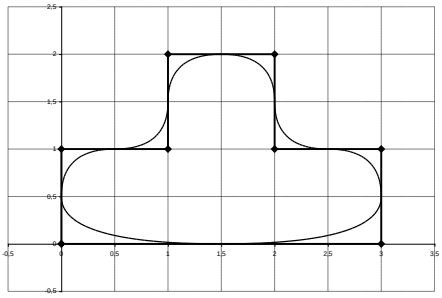


Fig. 2-6. The limit curve for the quadratic B-spline scheme (obtained by ternary subdivision).

2.3 Interpolating subdivision curves – the four-point scheme

Dyn, Gregory and Levin [Dyn87] suggested a way of creating a subdivision curve that interpolates the points of a given input polygon. Between each two successive points, a new point is inserted depending on the positions of the surrounding control points. If only two nearby points are used, without any additional information, the optimal position would be to add the new point in the middle between the two existing points. This process leads to a piecewise linear interpolation of the original control points, thus a C^0 curve.

As only looking at two surrounding points turns out to only lead to a piecewise linear approximation, Dyn et al. considered the four point neighborhood: two points on either side of the new point. Four points can be used to construct a third degree interpolating polynomial. The parameter value in the middle of that curve defines the location of a new point.

If the old points are taken at equidistant parameter values, the new point can be calculated using fixed weights. The new point p is inserted between the existing points p_1 and p_2, which have p_0 and p_3 as immediate neighbors:

$$p = \frac{1}{16}\left(-p_{-1} + 9p_0 + 9p_1 - p_2\right) \tag{2-5}$$

This process can be executed for all points of the polygon, and repeated recursively to generate a sequence of each time denser polygons. In the limit, a smooth curve will be obtained. The limit curve turns out to be C^1, but not C^2. In general the limit curve will not be a cubic polynomial. Only when the initial points all lie on the same cubic polynomial will the resulting limit curve be that same cubic polynomial. However, the curve is generally much smoother than a general C^1 curve, which leads to the statement that the curve is "almost C^2".

The four-point scheme is illustrated in figures 2-7 through 2-10. Starting from the initial control polygon of figure 2-7, new points are added between the existing ones. Figure 2-8 shows the first subdivision step where the position of the new points is defined as the center of the unique cubic curve that interpolates the four surrounding points. Figure 2-9 shows the second step, and after an infinite refinement results in the curve of figure 2-10.

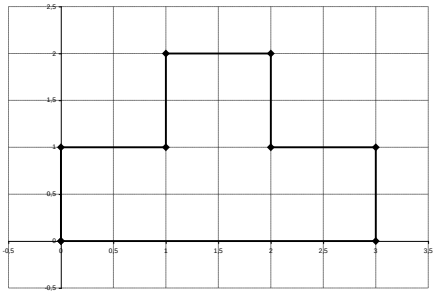


Fig. 2-7. Original control polygon.

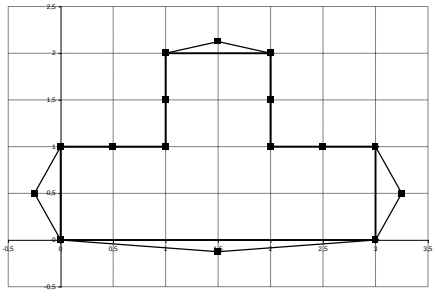


Fig. 2-8. One subdivision step of the four-point scheme.

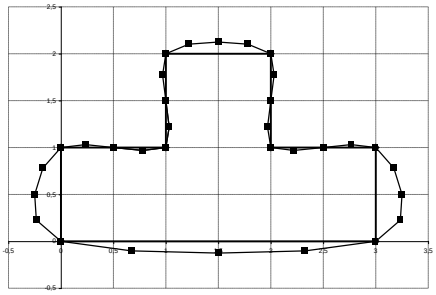


Fig. 2-9. Two subdivision steps of the four-point scheme.

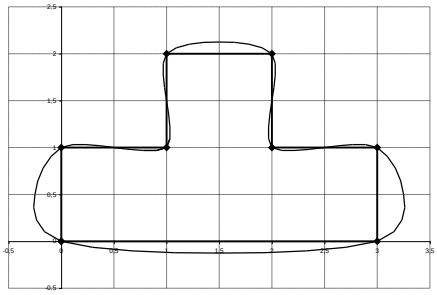


Fig. 2-10. Limit curve step of the four-point scheme.

By considering larger neighborhoods, the idea behind the four-point scheme can be extended to create higher-degree curves. For example, a six-point neighborhood creates new points using a fifth-degree polynomial. If applied using equidistant parameter values, also this scheme can be expressed with fixed weights. In [Sabin01], the following weights are calculated for the new point between p_0 and p_1 in the six-point scheme:

$$p = \frac{1}{256}\left(3p_{-2} - 25p_{-1} + 150p_0 + 150p_1 - 25p_2 + 3p_3\right) \qquad (2\text{-}7)$$

The four-point subdivision scheme has been used a lot as an example for further investigation. [Dyn98] derived conditions to maintain geometric constraints, such as preserving the convexity of the initial data, by allowing the weights of the scheme to vary. A method to analyze the continuity of this kind of schemes can be found in [Kobbe98].

The four-point subdivision scheme inspired Dyn, Levin and Gregory to create a variant of Loop's scheme that interpolates the points of its initial control mesh. Kobbelt used a similar approach to create an interpolatory quadrilateral scheme [Kobbe96]. Later, also Labsik used the same curve scheme to create another interpolatory triangular scheme [Labsik00a]. These schemes are described in sections 3.9, 3.10 and 3.11 of this dissertation.

2.4 Comparing approximating and interpolatory curves

The approach used to create the four-point scheme and examples such as figure 2-10 suggest that the scheme would always produce nice-looking

interpolating curves. In reality, it turns out that these interpolatory curves have some annoying peculiarities.

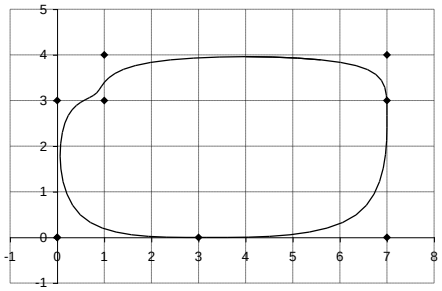


Fig. 2-11. The approximating scheme for cubic B-Splines using a simple looking control polygon.

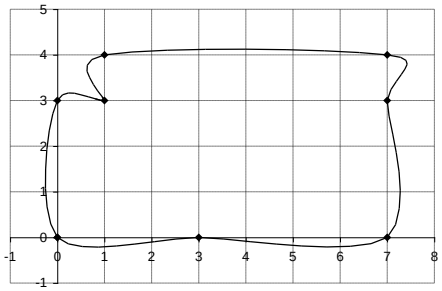


Fig. 2-12. The interpolating four-point scheme using the same control polygon.

Figure 2-11 shows an example of a cubic approximating scheme for a quite innocent-looking control polygon. Locally, four control points influence the form of the curve, which approximates their form, but normally does not interpolate them.

The same control polygon is used showing the behavior of the interpolatory four-point scheme (figure 2-12). Again, four surrounding points control the local behavior of the curve. Although the curve is smooth, it is not as “fair” as one would like. The popping up of bumps is difficult to control, due to the interpolatory conditions put on the curve. One partial solution to the problem would be to allow the four-point scheme to be non-uniform: instead of taking all parameter values equidistant, allow them to be irregularly spaced, which actually creates many more degrees of freedom. Kobbelt and Schröder [Kobbe97] also suggested a variational approach to further improve the quality of the curve: at each subdivision step, an energy function is evaluated as well in order to minimize the bending energy.

As reverting to non-uniformly sampled B-Splines, and even more so for the variational approach, results in curves that are more difficult to evaluate and analyze mathematically, for many applications the more intuitively behaving approximating schemes are used. When subdivision schemes are applied to create surfaces, the interpolating schemes produces bulges which are even more difficult to manipulate, implying that the approximating schemes are seen much more often in practical situations.

2.5 Eigenanalysis of subdivision curve schemes

Eigenanalysis is a handy tool to study the limit behavior of a subdivision curve scheme. A single step can be described in matrix form. In order to cope with end-point conditions, the matrix formulation has the problem that the matrix should double in size after every subdivision step. Therefore, usually a matrix of infinite dimensions is used. Such a matrix can either represent the subdivision of an infinite chain of points, or a closed curve. However, for many practical investigations, also a very limited matrix can also be used. In that case, the matrix represents a local environment, that shrinks with every subdivision step [Halst93, Joy96].

As an example, let us consider the subdivision scheme for cubic B-splines [Lane80].

In the local environment of three consecutive control points p_{-1}, p_0 and p_1 the refinement process can be written as:

$$\begin{bmatrix} p_0^{\,1} \\ p_{-1}^{\,1} \\ p_1^{\,1} \end{bmatrix} = S \begin{bmatrix} p_0 \\ p_{-1} \\ p_1 \end{bmatrix} \tag{2-8}$$

For the cubic scheme, the refinement matrix S is equal to:

$$S = \frac{1}{8}\begin{bmatrix} 6 & 1 & 1 \\ 4 & 4 & 0 \\ 4 & 0 & 4 \end{bmatrix} \tag{2-9}$$

Repeated refinement can be seen as a matrix multiplication. The n-th refinement step can be expressed by applying S k times:

$$\begin{bmatrix} p_0^{\,n} \\ p_{-1}^{\,n} \\ p_1^{\,n} \end{bmatrix} = S \begin{bmatrix} p_0^{\,n-1} \\ p_{-1}^{\,n-1} \\ p_1^{\,n-1} \end{bmatrix} = \cdots = S^n \begin{bmatrix} p_0 \\ p_{-1} \\ p_1 \end{bmatrix} \tag{2-10}$$

The limit position p_0^{∞} can be calculated after decomposing S into three separate matrices:

- a matrix, R, whose rows are the right eigenvectors of S,
- a matrix, L, whose columns are the left eigenvectors,

– and the matrix Λ with a diagonal of eigenvalues.

Using the fact that $R \cdot L = L \cdot R$ is equal to the identity matrix, applying the subdivision matrix n times to the local environment can be expressed as:

$$S^n = (R\Lambda L)^n = R\Lambda^n L \tag{2-11}$$

Therefore, R and L control the behavior of the local environment as n approaches infinity:

$$\begin{bmatrix} p_0^{\,n} \\ p_{-1}^{\,n} \\ p_1^{\,n} \end{bmatrix} = S^n \begin{bmatrix} p_0 \\ p_{-1} \\ p_1 \end{bmatrix} = R\Lambda^n L \begin{bmatrix} p_0 \\ p_{-1} \\ p_1 \end{bmatrix} = R \begin{bmatrix} 1^n & 0 & 0 \\ 0 & \left(\frac{1}{2}\right)^n & 0 \\ 0 & 0 & \left(\frac{1}{2}\right)^n \end{bmatrix} L \begin{bmatrix} p_0 \\ p_{-1} \\ p_1 \end{bmatrix} \tag{2-12}$$

In the limit, with n approaching infinity this equation narrows down to:

$$\begin{bmatrix} p_0^{\,\infty} \\ p_{-1}^{\,\infty} \\ p_1^{\,\infty} \end{bmatrix} = R \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} L \begin{bmatrix} p_0 \\ p_{-1} \\ p_1 \end{bmatrix} \tag{2-13}$$

Using the concrete values of R and L for the refinement matrix of Chaikin's scheme, this equation can be expanded as:

$$\begin{bmatrix} p_0^{\,\infty} \\ p_{-1}^{\,\infty} \\ p_1^{\,\infty} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \frac{1}{6} \begin{bmatrix} 4 & 1 & 1 \\ 0 & 3 & 3 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ p_{-1} \\ p_1 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 4 & 1 & 1 \\ 4 & 1 & 1 \\ 4 & 1 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ p_{-1} \\ p_1 \end{bmatrix} \tag{2-14}$$

Therefore, in the limit this local neighborhood ends in one point:

$$p_0^{\,\infty} = \frac{2}{3} p_0 + \frac{1}{6} p_{-1} + \frac{1}{6} p_1 \tag{2-15}$$

The tangent vectors can also be calculated similarly. To study such properties as curvature, an extended matrix using a larger local environment must be employed. Halstead et al. describe an extension of eigenanalysis to subdivision surfaces in order to deduce much useful information about the

limit surface [Halst93]. In chapter 5, eigenanalysis is used to study our new hexagonal subdivision surface scheme.

3 Subdivision surfaces

3.1 Introduction

Subdivision surfaces have been known to the computer graphics community since 1978, when Ed Catmull and Jim Clark published their tensor product extension of cubic uniform B-splines to surfaces [Catmu78]. They were followed at the end of the same year by Donald Doo and Malcolm Sabin who applied a similar technique to generalize quadratic uniform B-splines [Doo78]. Both schemes were named for their inventors and both operate on quadrilateral polygonal meshes of arbitrary topology. Some years later, in 1987, Charles Loop developed the first subdivision surface scheme based on a triangular control mesh, extending the formulas for a symmetric quartic box-spline to cope with extraordinary vertices [Loop87].

Since then, the research in these recursive subdivision surface schemes has led a rather neglected existence, until recently the legendary animation company Pixar focussed attention on them again. In 1998 subdivision surfaces were used as the main modeling tool for their short Geri's Game [DeRose98]. Developments picked up during 2000 and 2001 by the introduction of several new subdivision schemes and many new features, applications areas, modeling tools and profound mathematical analysis.

Subdivision surfaces are defined as the limit surface obtained by repeated refinement of a 3D control point mesh. In general, this limit surface does not have an exact algebraic representation. Many subdivision surface schemes are generalizations of tensor product B-spline surfaces or box-splines, so they are equal to these underlying schemes in the regions where the mesh exhibits a regular structure.

The 3D control point mesh that is used as input for subdivision surface generation is organized as a non-manifold polygonal mesh. The word "polygonal" is deceiving, however, as the polygons do not need to be planar. Even if they were planar, the subdivision process usually creates new non-planar polygons. In the rest of the text, the word "face" will often be used to refer to these polygons, as in general computer graphics applications a "face" is a 3D polygon with an established orientation. The orientation decides which side should be considered "outside" and it also defines a natural order around vertices, which is used during the creation of new faces.

The *valence* of vertex is also often referred to. The valence is the number of neighboring edges of a vertex. For example, a cube contains six faces and eight vertices. Each of the eight vertices has three neighbors, so their valence is three. The valence is very important in a subdivision surface scheme. The scheme behaves best for vertices with a regular valence. In triangular meshes this regular valence is six, in quadrilateral meshes it is four and in hexagonal meshes this valence is three. Vertices with a valence that is preferred by the scheme are called ordinary vertices. In general, the surface will be the smoothest around these vertices. Vertices with any other valence are called extraordinary, and usually the surface will have a lower degree of continuity in such regions. Luckily, all the new vertices produced by a subdivision scheme are all ordinary. (Section 3.3 explains in more detail the differences between primal and dual schemes.) During the recursive subdivision process, more and more ordinary vertices are created, which leads to an isolation of the extraordinary vertices.

Important for subdivision surfaces compared to other surface representations is that they allow the surface to have an arbitrary topology. Subdivision surfaces can be used to model a large class of surfaces, which can combine smooth and non-smooth features like creases and semi-sharp edges. The polygonal mesh that represents the limit surface allows for a simple and intuitive interaction with the models.

Subdivision surface schemes are defined as a multi-stage recursive process. During the first stage, the input mesh is upsampled by adding new vertices, in a way depending on the particular subdivision scheme. For example, the Loop scheme, which is only defined on triangles, adds new vertices at the center of the existing edges. Some subdivision schemes keep their old vertices, while in other schemes they are removed. The initial upsampling stage is followed by one or more averaging stages. The positions of the new vertices are averaged with their neighbors. The rules again depend on the particularities of the subdivision scheme, and on whether the vertex is an

existing one, or where exactly it is inserted relative to the original polygon mesh. Furthermore, these rules depend on the valence of the vertex. These stages are repeated recursively, usually leading to a smooth limit surface. In practice, the recursive process is stopped after some determined number of repetitions. In order to minimize the number of polygons generated to represent a smooth enough representation of the surface, adaptive subdivision techniques are introduced, which subdivide further where the surface is highly curved and less in more planar regions.

Although the limit surface can never actually be reached, mathematical tools like eigenanalysis make it possible to obtain exact information about the limit surface, such as the limit position of arbitrary points on the surface, as well as their normal and even their curvature. This is again useful for physical simulations and for applications like ray-tracing the surface.

A good introduction to subdivision schemes can be found at [Joy96]. For a more in-depth overview of the state of the current research in subdivision surfaces, we refer to [Zorin00a] and [Hubeli01]. An overview of more general surface representations can be found in [Hubeli00]. Although this work is mainly directed at geologic applications, its 142 pages give an extensive taxonomy of surface representations.

Subdivision surface schemes have many attractive features, which are elaborated in chapter 4. First, the following sections provide a global overview of existing schemes known in the subdivision literature.

3.2 Classification of subdivision surface schemes

In this section, we give a classification of subdivision schemes. For more details about the mentioned schemes, we refer to the subsequent sections. This classification is not only a useful introduction to the growing set of subdivision schemes, it also explains our motivation for proposing a new subdivision scheme, which is presented in chapter 5.

	Primal (face-split)		Dual (vertex-split)
	Triangles	Quadrilaterals	Quadrilaterals
Approximating (not interpolating original vertices)	Loop	Catmull-Clark	Doo-Sabin Midedge
Interpolating the original vertices	Butterfly Modif.Butterfly	Kobbelt	

Table 3-1. Usual classification of subdivision surface schemes. The mentioned schemes are explained in more detail in the later sections of this chapter.

Table 3-1 represents the usual classification for the different subdivision surface schemes that exist today: schemes are classified as either approximating or interpolating, and further as either primal face-splitting algorithms or dual vertex-split algorithms.

Interpolating schemes, such as the Butterfly scheme ([Dyn90], see section 3.11), interpolate all vertices of their initial control mesh. Approximating schemes only approximate their initial control mesh, leading to surfaces that have less unwanted bulges and behave more predictable during interactive manipulations.

A face-split algorithm splits the existing faces by dividing their edges (usually into two) and creates new faces. In both the triangular and the quadrilateral schemes, the faces are typically split into four. In the quadrilateral case, this requires also a new vertex in the center. More details about the properties of both type of schemes can be found in section 3.3.

A vertex-split algorithm replaces all existing vertices by a new face. As the existing faces survive the subdivision process, but with their old vertices cut away, these schemes are also called corner-cutting schemes. In order to get the mesh closed again, usually the existing edges also need to be replaced by new faces. These schemes are called dual, because their effect on the mesh connectivity can be seen as a face-split operation, followed by interchanging the roles of vertices and of faces. Old vertices become the center of new faces and old faces are reduced to new vertices. The classical example of such a vertex-split algorithm is the Doo-Sabin scheme.

The classification of table 3-1 is found for example in the subdivision course notes of Siggraph 2000 [Zorin00a], that are generally regarded as representing the current state-of-the-art. Similar classifications can be found in the even more recently course notes of IEEE Visualization 2001 [Hubeli01] or the paper by Zorin and Schröder [Zorin01a].

Recently, however, new schemes have been presented that do not directly fit into this classification. Kobbelt's Sqrt(3) scheme [Kobbe00], splits triangles into three, but needs to flip the existing edges to create new faces. Two subsequent subdivisions, however, flip the edges back to their original orientation and result in splitting the faces into nine subfaces. A scheme with the same effect on the connections of the mesh but with different subdivision rules has been presented at a Dagstuhl conference and is mentioned in [Sabin01]. The double step of both schemes fits in the approximating face-split classification.

Based on the connectivity rules of the Sqrt(3) scheme, Labsik and Greiner [Labsik00a] introduced an interpolatory variant, employing the interpolation ideas of the Butterfly scheme [Dyn90] and Kobbelt's interpolating scheme for quadrilateral meshes [Kobbe96].

Velho and Zorin came up with a new idea based on dividing quadrilaterals diagonally into two triangles [Velho01a]. The scheme behaves like a face-splitting scheme, but surprisingly it splits quadrilaterals by creating triangles. The originators called this the 4-8 scheme, as it works with vertices which alternate valences four and eight.

As these recent schemes do not strictly fit into the original classification of table 3-2, we added them, *in italics*, at the most appropriate position.

A more important observation about the classification of table 3-1 is that for the primal schemes two types of meshes can be used: triangular and quadrilateral. For the dual schemes, however, the classification only considers quadrilateral meshes. It is strange that the table does not provide a dual scheme for triangular meshes. The dual of a regular triangular mesh is a hexagonal mesh, and it is not too difficult to define schemes that operate on these meshes, as we show in chapter 5 of this dissertation. Regarding the connectivity of the subdivided mesh, two types of schemes can be defined. The first one divides hexagons into four new hexagons, of which three need neighboring hexagons to get closed. Such a scheme is described in a paper under preparation by Simoens, Dyn and Levin [Simoe01]. As this scheme does not yet have a fixed name, we'll refer to it as the hexagon-by-four scheme. The scheme is a dual scheme, but is not really a vertex split scheme as the

original vertices are not removed (see section 5.4). The second type of scheme replaces existing hexagons with three new hexagons and effectively replaces every old vertex with a new one. We call this scheme the hexagon-by-three scheme. It will be explained in full detail in chapter 5 of this dissertation.

All these considerations convert the classification of table 3-1 to the extended classification of table 3-2, shown below.

<table>
<tr><th></th><th colspan="2">Primal (face split)</th><th colspan="2">Dual (vertex split)</th></tr>
<tr><td></td><td>Triangles</td><td>Quadrilaterals</td><td>Quadril.</td><td>Hexagons</td></tr>
<tr><td rowspan="2">Approximating
(not interpolating original vertices)</td><td>Loop
Sqrt(3)
Dagstuhl</td><td>Catmull-Clark</td><td rowspan="2">Doo-Sabin
Midedge</td><td rowspan="2">Hexagon-by-3
Hexagon-by-4</td></tr>
<tr><td colspan="2">4-8 Scheme (Velho)</td></tr>
<tr><td>Interpolating
the original vertices</td><td>Butterfly
Modif.Butterfly
Interp. Sqrt(3)</td><td>Kobbelt</td><td></td><td></td></tr>
</table>

Table 3-2. Extended classification of subdivision surface schemes.

Each of the schemes depicted in table 3-2 will be detailed in the sections starting with section 3.4, explaining their definition and their most important properties. But first, section 3.3 will elaborate the important difference between primal and dual schemes.

3.3 Primal versus dual subdivision surface schemes

In [Zorin01a] and [Stam01], the duality between subdivision surface schemes is an important concept. The dual of a polygonal mesh is formed by exchanging the role of points and faces. For example, in the case of the Platonic solids (see figure 3-1), the cube and the octahedron are each other's dual: the cube has faces with four edges and vertices with valence three, while the octahedron has faces with three edges and vertices with valence four. In the same way, the dodecahedron with faces with five edges and

vertices with valence three is the dual of the icosahedron. The tetrahedron is special because its dual is again a tetrahedron.

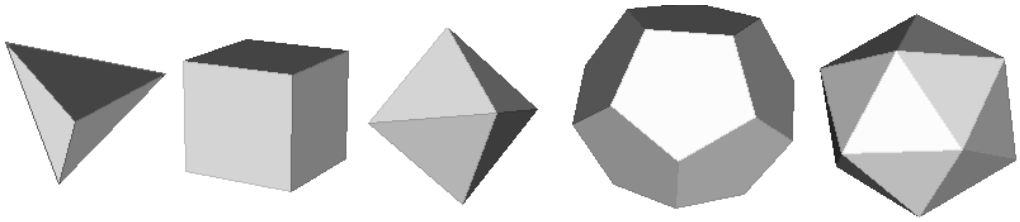


Fig. 3-1. The five Platonic solids: The tetrahedron, the cube, the octahedron, the dodecahedron and the icosahedron. The cube and the octahedron are each other's dual, just as the dodecahedron and the icosahedron. The tetrahedron is its own dual.

Subdivision surface schemes are considered primal if the meshes they generate are obtained by splitting the faces, usually into four smaller faces [Catmu78, Loop87]. These four parts together again represent their original face. For a dual scheme, this is no longer true: the newly created faces can either match vertices, edges or faces of the original faces [Doo78, Peters97].

The preferred number of vertices for the faces of a scheme depends on the regular grid on which it is based. This regular grid can either be triangular [Loop87, Dyn90, Kobbe00], quadrilateral [Catmu78, Doo78, Kobbe96, Peters97] or hexagonal [Simoe01, Claes02]. The preferred valence also depends on this regular grid, and is four for quadrilateral schemes, six for triangular schemes and three for hexagonal schemes. Every vertex that has the preferred valence is called ordinary; all the others are extraordinary. The 4-8 scheme [Velho01a] is a special case, as it is based on regular grids with mixed valences.

Primal schemes generate new polygons that all have the scheme's preferred number of edges. Existing extraordinary vertices are left untouched, while newly generated vertices again have the scheme's preferred valence.

For dual schemes, also here the roles of vertices and faces are exchanged. Dual schemes shrink existing irregular faces, leaving their number of edges intact. Extraordinary vertices, however, are converted to faces with an irregular number of edges. For quadrilateral schemes, the new faces have the same number of edges as the valence of the vertex from which they originate. All newly created vertices are all ordinary.

Therefore, after the first subdivision step, either only faces with the preferred number of edges (for the primal schemes) or only ordinary vertices (for the dual schemes) remain. The dual schemes can create extraordinary faces during their first subdivision step, but from then on, the number of extraordinary faces stays constant, while the number of regular faces grows exponentially during each subsequent subdivision step.

In practice, dual subdivision schemes are used much more than primal schemes. The most important reason for this is that primal schemes keep their existing edges, which are subdivided in a very similar way as if they were subdivision curves. In fact, by arranging the points of the faces in a symmetrical way, it is possible to oblige the edges to interpolate a subdivision curve. This property of dual schemes turns out very handy for creating borders and sharp edges, and even semi-sharp edges [Hoppe94, Schwe96, DeRose98]. Dual schemes cut off their existing edges, which makes implementing borders and sharp edges more complicated [Nasri87].

For primal schemes, the silhouette edges keep looking good, even when the mesh is only subdivided a few times. This is an important benefit for primal schemes, as silhouettes are an important visual clue to how a surface looks. In dual schemes, the edges are constantly cut away, considerably changing the look of the silhouettes at each subsequent step. After a few subdivision steps, however, the surface is smoothed a lot, and silhouettes only change in a subtle way thereafter.

3.4 Midedge subdivision

Jörg Peters and Ulrich Reif proposed a very simple way to subdivide a polygonal mesh [Peters97]. They create new edges in a polygonal mesh by interconnecting the midpoints of each edge to the midpoints of all edges that have both a face and a vertex in common. By joining these edges a new mesh is created to form two types of faces:

- Each existing vertex in the old mesh is replaced by a new face, effectively cutting the vertex away.
- Each existing face of the old mesh is replaced by a smaller face consisting of the midpoints of its old edges.

Remark that in general these cuts are not planar. The subdivision rules are very simple: there is only one type of new point (the midpoint of the edge), whose position is calculated using only two old vertices.

It turns out that, when limited to regular quadrilateral meshes, two steps of this subdivision algorithm are equal to a factored box-spline subdivision. The underlying four-directional symmetrical box-spline is known as the Zwart-Powell element.

As these simple rules lead to too slow convergence speeds for polygons with a large number of vertices, Peters and Reif adapted their scheme to make sure that all polygons shrink at the same rate. Therefore, they need to include all vertices of the polygon into the equation, which is furthermore only defined when combining two steps. This makes their modified scheme very similar to the Doo-Sabin scheme, with some slightly different rules for extraordinary vertices. Unfortunately, no adapted rules exist for the single step algorithm.

Six subsequent steps of the Midedge subdivision are shown in figure 3-2. Note that each step simply creates new faces by connecting the centers of the edges. In this simple version of the Midedge scheme, the triangles that are formed at the corners of the original cubes shrink more quickly then the quadrilaterals.

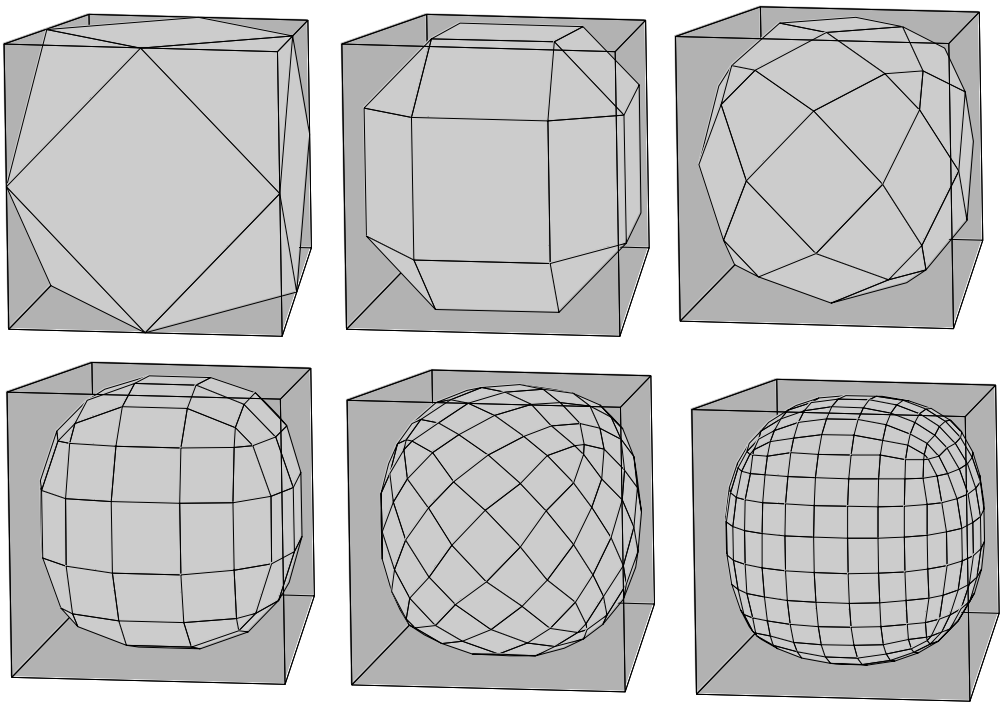


Fig. 3-2. Six steps of the Midedge subdivision of a cube.

3.5 Catmull-Clark

In 1978, Ed Catmull and Jim Clark introduced the first subdivision scheme to generate surfaces with an arbitrary topology [Catmu78]. They generalized the subdivision scheme for cubic B-splines to a tensor-product definition (see section 2.2 of chapter 2). This approach is nicely defined on regular quadrilateral meshes and can be executed in two separate passes. First, the curve scheme is executed in one direction, followed by a second pass in the orthogonal direction. Catmull and Clark's most important innovation was the extension of the scheme allowing it to cope with non-regular meshes. In the regular setting, the mesh consists solely of quadrilaterals and all vertices have a valence of four. Catmull and Clark observed that they could split faces that are not quadrilaterals in a similar way as the faces are split in the regular case. Just add a point in the center of the face and connect it to the center of every edge. This ensures that starting from the first subdivision step, all generated

faces are quadrilaterals. Also newly generated points at the centers of the edges nicely get a valence of four. Only the centers of input faces that were not quadrilaterals lead to the creation of an extraordinary vertex. This implies that the number of extraordinary vertices stays constant, namely one for each extraordinary vertex in the input mesh and one for each face that was not a quadrilateral.

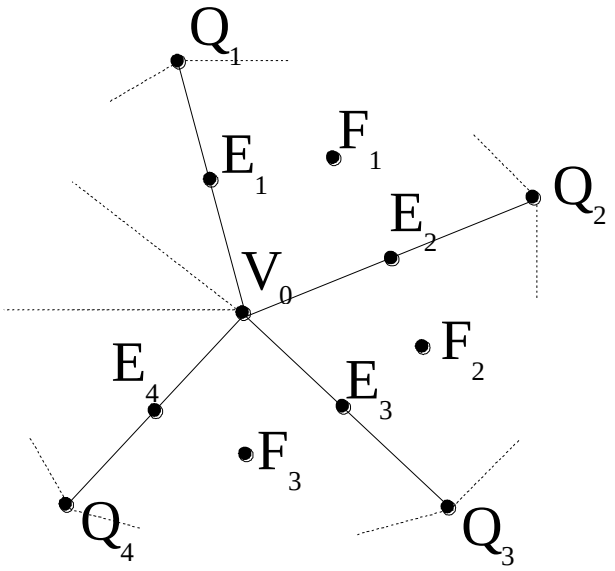


Fig. 3-3. Subdivision around a central vertex V_0, showing surrounding control points (Q_i), edge points (E_i) and face points (F_i).

Starting from figure 3-3, we'll describe the rules used by the Catmull-Clark scheme. Catmull and Clark designed their rules to match the tensor product in the regular case and to try to mimic this as much as possible for irregular cases. Figure 3-3 depicts the situation around an existing vertex. An initial vertex V_0 is surrounded by n edges, leading to n neighbor vertices Q_i. A first step in the subdivision process is to insert so-called face points F_i at the centers of the faces. Then, for every edge a so-called edge point E_i is calculated as the mean between the two vertices and the two face points of the faces that make up the edge. Finally, the positions of the existing vertices are relaxed by averaging them with their neighbors in the following way. V_1 is the position of V_0 after the first subdivision step and is calculated as:

$$V_1 = \frac{n-2}{n} V_0 + \frac{1}{n^2} \sum Q_i + \frac{1}{n^2} \sum F_i \qquad (3\text{-}1)$$

After using these rules for face points, edge points and vertex points, new faces are formed. First the existing edges are split using the edge points and the new vertex points, and then new faces are formed by connecting the edge points with the face points.

In the literature, different weighting factors have been used for the calculation of the new vertex point. But, as long as these weights stay within certain limits and are applied in a uniform and stationary way, most features of the original scheme stay valid [Zorin00a].

Catmull-Clark's scheme is not only the first scheme, it is also the most used in modeling applications, as it lends itself to model objects with rectangular symmetries and typical cylindrical features, such as arms and legs. Furthermore, it can be extended quite easily to support sharp and semi-sharp edges [DeRose98] (see also section 4.7). Halstead et al. used eigenanalysis on Catmull-Clark's scheme to obtain explicit formulas for the limit position of the vertices and the surface normals [Halst93].

3.6 Doo-Sabin

During the same year that Catmull and Clark published their scheme, Donald Doo and Malcolm Sabin introduced their scheme, also based on a tensor product for subdivision curves. Instead of cubic curves, they used quadratic curves. This leads to quite simple rules. Only one type of new point is introduced, at the center of a quadrilateral formed by an existing vertex, two edge points and the center of the face. This effectively shrinks the existing faces to half their original size. In order to close the mesh again, also new faces are put around the old vertices and edges. This has the visual effect of cutting away the corners of the polygonal mesh. The mesh obtained is the dual of the mesh from the Catmull-Clark scheme, interchanging the roles of points and faces.

In figure 3-4 the Doo-Sabin subdivision process is illustrated for a polygon with five vertices.

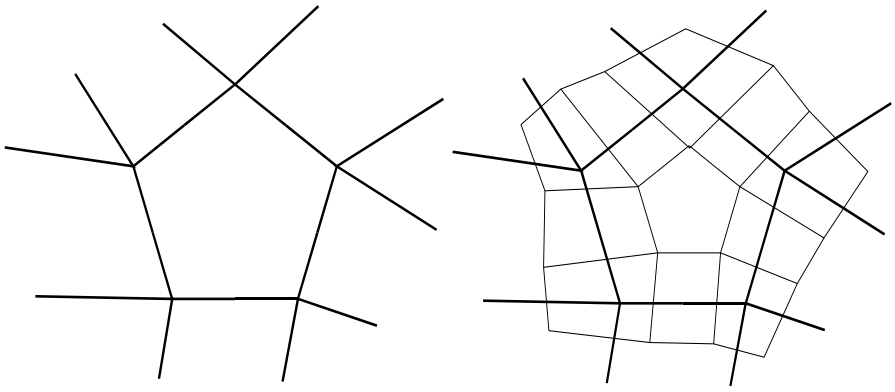


Fig. 3-4. Left: An input polygon with surrounding edges. Right: The new faces created by one subdivision step of the Doo-Sabin algorithm.

3.7 Higher degree B-Spline surfaces

In 2001, two research papers were published containing the idea to use repeated averaging to obtain surfaces with a higher degree of continuity. Zorin and Schröder [Zorin01a] as well as Jos Stam [Stam01] started from the idea of factorizing Lane and Riesenfeld's formula for subdivision curves with any degree of continuity [Lane80] (see section 2.2). For example, a subdivision scheme for a B-spline curve of degree six normally needs a support of seven neighboring vertices. The interesting observation made by [Zorin01a] and [Stam01] is that the same scheme can be factorized in six subsequent averaging steps, each time using only two neighboring vertices. For curves, this factorization does not seem to help a lot, but for subdivision surface schemes based on these curves, it brings essential new possibilities. For surface meshes, a neighborhood of seven vertices in every direction would require a region of 49 vertices in a regular quadrilateral mesh. A key issue for subdivision surface schemes, however, is that they also should be defined for irregular meshes, making it practically impossible to define how the surrounding region should be sampled for every irregular combination. Subdivision surface schemes already have enough problems defining how to adequately sample a neighborhood of one or maximally two surrounding vertices. The factorization of the subdivision rules, however, permits this higher degree of continuity to be obtained by repeatedly averaging a limited region.

It turns out that, for the repeated averaging, it is not possible to work with one simple mesh. In the case of curves, every averaging step of the vertices of a control polygon creates a new control polygon with vertices in the middle of the existing ones. The total number of vertices stays the same, however. The new mesh can be considered as the dual of the old one: just interchange the roles of vertices and the edges connecting them. To imitate this behavior on the surface schemes that are defined as tensor products of these curves, also a dual mesh is needed. For the polygonal meshes, this dual mesh consists of interchanging the roles of polygons and vertices. The meshes generated by Catmull-Clark's and Doo-Sabin's scheme possess such a duality and were employed by Stam, Zorin and Schröder to create surfaces with a high degree of continuity.

3.8 Loop's scheme

In 1987, Charles Loop generalized the subdivision rules of a symmetric quartic box-spline over a regular triangulation to include rules to be applied in the vicinity of extraordinary points [Loop87]. The limit surface is C^2 continuous everywhere except at the extraordinary points, where it is only C^1. Loop showed that at the extraordinary points the surface exhibits a continuous tangent plane, as long as the weighting factors stay between certain limits. Although part of Loop's motivations were based on intuition, it turned out that the rules that he considered as optimal still survive today as being the most suited for stationary triangular subdivision.

In a document that has only been published as a draft on the Internet, but nevertheless is often referenced, Joe Warren [Warren95] proposed some alternative rules. For most practical applications, the visual difference between the two sets of rules is minimal, but Warren's rules have the advantage that they are simpler for mathematical analysis.

Loop's scheme has been studied extensively. Prautzsch, for example, showed how the eigenvalues of the subdivision matrix can be changed to optimize the continuity of the limit surfaces [Praut99, Praut00]. And Bischoff et al. even studied the possibilities of a hardware implementation of this scheme [Bisch00].

Loop's scheme is used in chapters 9 and 10 of this dissertation, where more mathematical details about the scheme can be found.

In figure 3-5, four consecutive steps of the Loop subdivision scheme of a triangle is shown. Each time all existing triangles are divided into four smaller triangles.

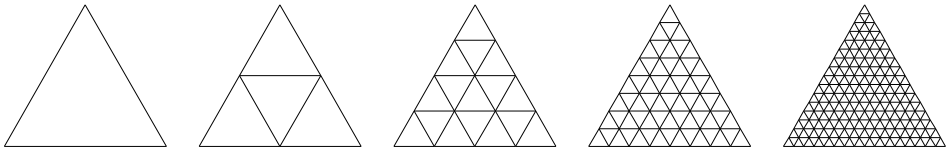


Fig. 3-5. Four steps in the subdivision of a triangle.

3.9 Sqrt(3) subdivision

At Siggraph 2000, Leif Kobbelt presented a new scheme for triangular meshes. While Loop divided the existing triangles by cutting every edge into two pieces, Kobbelt adds a new point in its center. In order to keep the newly generated triangles from getting skinnier and skinnier, at each subdivision step, the connecting edge between neighboring triangles is flipped (see figure 3-6). After two successive subdivisions, the triangles revert to their original orientation. It turns out that two steps divide the original triangle into nine smaller ones and split the edges into three. Therefore, Kobbelt suggested that one subdivision step cuts the edge length by a factor of sqrt(3). This remark inspired the name for his new subdivision scheme.

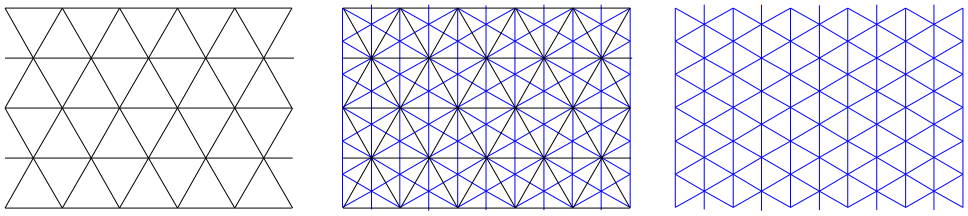


Fig. 3-6. The Sqrt(3) subdivision scheme on a regular triangular grid. Left: The original triangles. Center: New points are added in the center and new triangles are created by flipping the existing edges. Right: The result after one subdivision step.

Kobbelt suggested the following rules for his Sqrt(3) scheme. A new point Q in the center of a triangle $P_1P_2P_3$ is simply calculated as:

$$Q = \frac{1}{3}\left(P_1 + P_2 + P_3\right) \tag{3-2}$$

Furthermore, the position of the existing points are relaxed by averaging them with their immediate neighbors. The new position P' of an existing point P, surrounded by n original points $P_1P_2 \ldots P_n$ is calculated as:

$$P' = \left(1 - \alpha_n\right)P + \alpha_n \frac{1}{n}\sum P_i \tag{3-3}$$

Equation 3-3 uses a weighting factor α_n that depends on the valence of P. Using eigenanalysis, Kobbelt concluded that the following value for α_n would be optimal:

$$\alpha_n = \frac{4 - 2\cos\left(2\pi / n\right)}{9} \tag{3-4}$$

The Sqrt(3) scheme does not seem to be based on a particular box-spline. Therefore, evaluating the surface and analyzing the behavior is more complicated than with the more conventional subdivision surface schemes.

An advantage of the Sqrt(3) scheme compared with schemes like Loop's and Catmull-Clark's, is that the number of generated polygons only grows with a factor of three by each subdivision step, compared to the factor of four for the conventional schemes. This makes it more probable that a certain level of detail is reached using fewer polygons. The difference between three and four becomes clear when one looks at multiplication factors by which new polygons are generated. For the Sqrt(3) scheme, these factors are growing like 3, 9, 27, 81, 243, … for subsequent subdivision steps, while for the Loop scheme they grow like 4, 16, 64, 256, … . After four steps, the Loop scheme already created more than three times the number of faces compared to the Sqrt(s) scheme and that difference grows exponentially with the number of subdivision steps.

3.10 Interpolatory Sqrt(3) subdivision

Inspired by the new connectivity of the Sqrt(3) scheme, Labsik and Greiner [Labsik00a] created an interpolatory variant. They employed the interpolation ideas of the Butterfly scheme [Dyn90] and Kobbelt's interpolating scheme for quadrilateral meshes [Kobbe96]. In order to get a small support region they also employed the four-point scheme for curves to derive a rule for inserting

the new points. As it is an interpolating scheme, there are no rules for the existing points: they simply stay in their original position. Only a rule for the new point in the center of each triangle is needed. When vertices with a valence different to six are encountered in the neighborhood, the rules are a bit complicated and are not written down explicitly. They are calculated from the results of an eigenanalysis of the subdivision matrix. As in the ordinary Sqrt(3) scheme, they need to combine two steps to get an analyzable situation.

As the connectivity of the mesh is the same as for the Sqrt(3) scheme, the generated meshes have the same structure as figure 3-6.

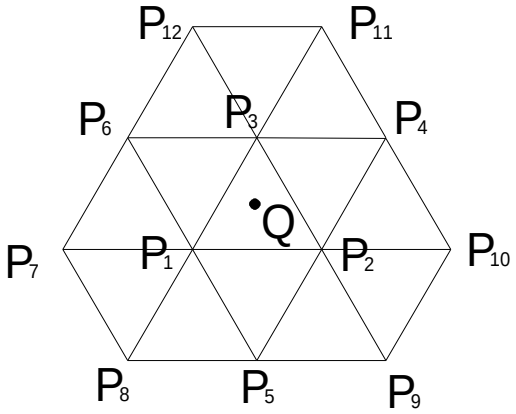


Fig. 3-7. Situation around a new point Q for the interpolating Sqrt(3) scheme.

For the configuration shown in figure 3-7, Labsik and Greiner derived the following formula to calculate the new position of the point Q. As it is an interpolating scheme, no new positions are needed for the already existing points.

$$Q = \frac{32}{81}\left(P_1 + P_2 + P_3\right) - \frac{1}{81}\left(P_4 + P_5 + P_6\right) - \frac{2}{81}\left(P_7 + P_8 + P_9 + P_{10} + P_{11} + P_{12}\right) \quad (3\text{-}5)$$

Around extraordinary points, more complicated formulas are needed, which were only derived for a double subdivision step.

3.11 The Butterfly scheme

The structure of the meshes created by the Butterfly algorithm is very similar to the meshes created by Loop's scheme [Dyn90, Dyn93]. It also creates new points by splitting the edges into two, followed by a relaxing step. The averaging masks used are quite different, however. The vertex-points always stay in their original position, which causes this scheme to be an interpolating one. The averaging mask for the newly inserted edge-points is depicted in figure 3-8 The form of this mask resembles a butterfly, hence the name of the scheme.

The limit surface is differentiable everywhere except at extraordinary points of valence n = 3 and n >= 8 [Praut00]. Although the surface is tangent plane continuous at extraordinary points of valence n >= 8, the surface is not regular as it has self-intersections. To overcome this shortcoming Zorin, Schröder and Sweldens [Zorin96] extended the algorithm creating G^1-surfaces for every type of extraordinary point.

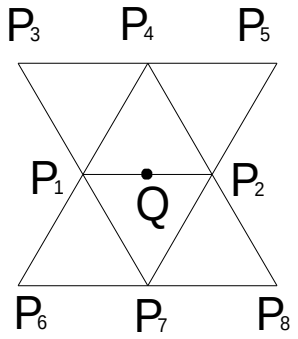


Fig. 3-8. Situation around a newly inserted edge point for the interpolatory Butterfly scheme.

Dyn et al. derived the following formula to calculate the position of the newly inserted point Q depicted in figure 3-8:

$$Q = \frac{1}{2}\left(P_1 + P_2\right) + 2w\left(P_3 + P_4\right) - w\left(P_5 + P_6 + P_7 + P_8\right) \qquad (3\text{-}7)$$

This equation uses a tension parameter that can be freely chosen between some bounds. Usually it is set to 1/16.

Junkins et al. based a real-time game application on the Butterfly subdivision scheme [Junki00]. The subdivision approach allowed them to work with objects in different levels of detail, generate efficient bounding boxes and implement a lazy evaluation approach.

3.12 Interpolatory subdivision for quadrilateral meshes

Leif Kobbelt further generalized the four-point interpolatory subdivision scheme for curves to a tensor product subdivision scheme for surfaces [Kobbe96]. So, where the Butterfly scheme is defined on triangular meshes, Kobbelt's scheme operates on quadrilateral meshes. The connectivity of how new points are introduced into the mesh, is the same as for Catmull-Clark's scheme, but the rules for calculating their positions is different. In his Siggraph'96 paper, Kobbelt describes specific rules for the computation of these positions, which are too elaborated to copy here. Kobbelt also introduces methods to create boundaries, sharp features and adaptive refinement. He further proves the C^1 continuity of his scheme.

3.13 Velho and Zorin's 4-8 scheme

Recently, in 2001, Luiz Velho and Denis Zorin [Velho01a] introduced the first subdivision surface scheme that is not based on meshes of regular polygons. Until then, the standard meshes were either based on quadrilaterals with regular vertices having valence four or on triangles with regular vertices having valence six. Instead, their new scheme has regular vertices that alternate valences four and eight, with the basic polygon type being a right triangle. These right triangles are obtained by cutting quadrilaterals over their diagonal into two pieces.

Usually the starting mesh of their scheme needs some preprocessing. If the given mesh is mainly quadrangular, it suffices to add diagonals in alternating directions. For triangular starting meshes, they described a more involved geometric construction.

Figure 3-9 shows a regular 4-8 tiling and two refinement steps. Meshes with irregular vertices are subdivided in a similar way.

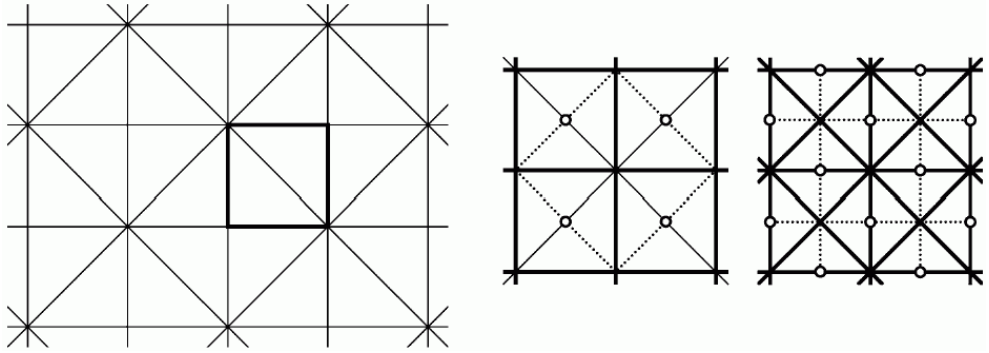


Fig. 3-9. Left: A regular 4-8 tiling, with one basic tile highlighted. Center: A new subdivision step first introduces new vertices in the center of the diagonals (marked with small circles) and adds new diagonals that are rotated 45° compared with the previous step (thick lines). Right: The subsequent subdivision step.

Due to how Velho and Zorin designed their subdivision rules, the scheme is more closely related to quadrilateral than to triangular schemes. Hence, two successive subdivision steps mimic a quadrilateral subdivision like Catmull-Clark's. An important difference with Catmull-Clark's scheme is that the 4-8 scheme generates surfaces that are C^4 everywhere, except in the neighborhood of a few isolated extraordinary points, where the surface in only C^1. A drawback, however, is that the 4-8 scheme needs a larger support area, making the required masks for calculating tangent vectors significantly larger.

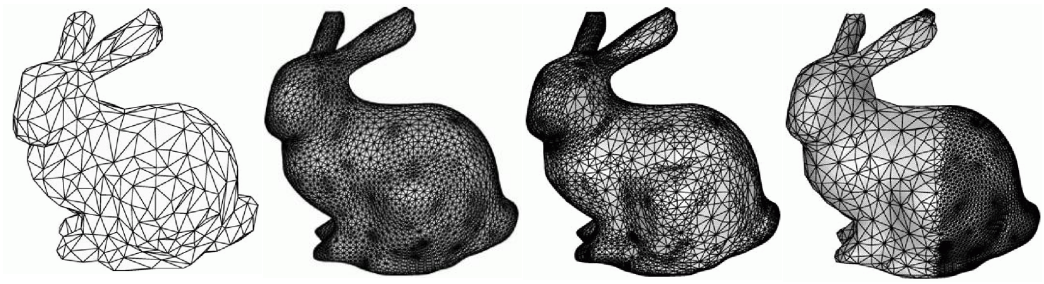


Fig. 3-10. A: The initial control mesh. B: Uniform subdivision. C: An adaptively refined mesh applying geometric stopping criteria. D: Adaptive subdivision with a spatial threshold, illustrating how rapidly the polygon density can change.

The most important feature of the 4-8 is its flexibility in which it allows adaptive refinement. As the number of faces only multiplies with a factor of two with each subdivision step, there can be an enormous difference in number of polygons between nearby areas. Figure 3-10 (courtesy of [Velho01a]) shows such adaptive subdivisions of a simplified mesh of the Stanford bunny.

3.14 The Dagstuhl scheme

This scheme is only mentioned in [Sabin01] and seems to be proposed at a Dagstuhl conference. Its way of generating new meshes, is similar to the Sqrt(3) scheme, but it uses a little larger support.

Sabin only describes the rules for the regular setting, which are derived from a particular box-spline surface. For a new point Q generated in the center of a triangle $P_1P_2P_3$, and with $P_4P_5P_6$ as the extreme corner points of the three direct neighboring triangles the formula is (see figure 3-11, left):

$$Q = \frac{1}{9}\left(2P_1 + 2P_2 + 2P_3 + P_4 + P_5 + P_6\right) \tag{3-8}$$

And the position of an existing point P_0 surrounded by six direct neighbors ($P_1P_2P_3P_4P_5P_6$) is relaxed using equation 3-9 (see figure 3-11, right):

$$P_0' = \frac{1}{9}\left(3P_0 + P_1 + P_2 + P_3 + P_4 + P_5 + P_6\right) \tag{3-9}$$

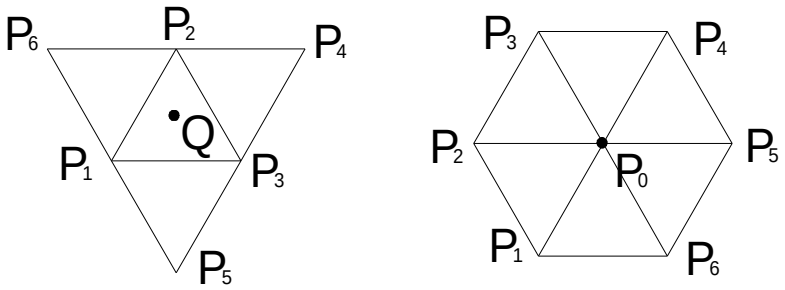


Fig. 3-11. The regular masks for the Dagstuhl scheme.

4 Properties of subdivision surfaces

4.1 Arbitrary topology

The most important advantage of subdivision surfaces compared to alternative techniques is that subdivision surfaces allow surfaces of arbitrary topology to be created. Not only can the mesh have any number of holes, it is also possible to seamlessly combine rough regions having a small polygon count with regions containing much more detail.

Before subdivision surfaces were in widespread use, surfaces were stitched together from rectangular or triangular patches, which needed to be trimmed to fit together. As the trimming process cannot always guarantee a precise border, small gaps and discontinuities are difficult to prevent.

The topology does not need to be restricted to manifold meshes. In a manifold mesh, each local environment of a point is topologically equivalent to a disk. In many applications, it can be handy to work with non-manifold meshes, where several surfaces can share common boundaries. Examples are a fish with fins or a plane with wings. Ying and Zorin show how Loop's subdivision scheme can be extended to operate on these non-manifold meshes [Ying01].

4.2 Level of detail

Depending on the requirements of the task at hand, subdivision surfaces can be generated with different levels of detail. When speed is important, a quick preview can be obtained using only one or two subdivisions. For a final high-quality rendering, many more subdivision steps will be computed [Pulli96].

Furthermore, techniques have been developed to adaptively approximate a surface depending on error bounds, like flatness or size of the final polygons. A subdivision surface scheme having an extreme capacity of combining both coarse and highly subdivided regions is the 4-8 subdivision introduced by Velho and Zorin [Velho01a].

Many practical tips are described in [Junki00] to incorporate the use of subdivision surfaces in a real-time game. With the help of lazy evaluation, parts of a subdivision surface that would be clipped away during the rendering, can be skipped, effectively minimizing the consumed computation time.

4.3 Numerical quality

The meshes produced by subdivision possess many of the nice properties finite element solvers require [Reif00]. Commonly applied subdivision surfaces schemes guarantee at least C^1 continuity and are suitable for many numerical simulation tasks, so they can be used in animation systems or engineering calculations.

Furthermore, the subdivision process makes use of simple linear combinations to calculate new points and new positions, so there is no need to multiply or divide the input variables with each other or to evaluate them in higher degree polynomials. All this leads to less susceptibility to rounding errors, ensuring a numerically stabile environment.

4.4 Convex hull property

When the weights in the subdivision mask are all positive and sum to 1, the limit surface is guaranteed to lie inside the convex hull of a limited set of neighboring control points. Moreover, this property is propagated by the recursive subdivision process, where each subsequent step puts the limit surface in a smaller convex hull [Pulli96].

This property is very useful, for example, when displaying the surface; as soon as all the control points of a part of the surface lie outside the view frustum, this part can be clipped away. A similar approach can be used for collision detection algorithms.

While almost all approximating subdivision schemes are restricted to use positive masks, interpolating schemes necessarily need negative weights for

generating a smooth interpolating surface [Dyn90]. Consequently, the interpolating schemes don't exhibit an easy convex hull property, which is one of the reasons why interpolating schemes are used less in practical applications.

4.5 Exact evaluations of points and normals

Besides the ability to construct a mesh using recursive refinement, some applications need a more exact evaluation of the limit points and limit normals of a surface. At the places where the mesh is regular, this information can be calculated straightforwardly using the underlying tensor product or box-spline. For the extraordinary points, Halstead et al. [Halst93] showed how eigenanalysis leads to exact formulas for their limit positions. The area around the extraordinary points, however, is more complicated, as it is formed by a recursive cascade of concentric bands. Each band viewed separately is regular, but bears an infinite series of smaller similar bands inside, ultimately converging to the limit position of the extraordinary point. With the help of eigenanalysis and pre-computed matrices, however, Jos Stam succeeded in calculating limit positions and normals for arbitrary points of the surface [Stam98, Stam99]. Later Zorin and Kristjansson extended this research to work in even more general conditions [Zorin00b].

4.6 Editing subdivision surfaces

The edit interface is simple and without the many limitations imposed by patched surfaces. Users can edit the arbitrary meshes freely and do not need to worry about the conditions to keep the surfaces without undesired discontinuities.

Special purpose editors can try to optimally exploit the possibilities of subdivision surfaces, minimizing the number of required polygons and trying to avoid as much as possible the introduction of extra-ordinary points. An example is the editor for cartoon faces by Skaria, Akleman and Parke [Skaria01].

4.7 Sharp and semi-sharp features

Subdivision surfaces are not limited to completely smooth surfaces. For example, in the Catmull-Clark scheme (see section 3.5), sharp edges can be

generated by keeping the points of the sharp edges in their original place instead of relaxing them by the normal subdivision rules [DeRose98]. The surrounding points keep following the standard rules of the scheme. Earlier Hugues Hoppe [Hoppe94] described similar approaches for Loop's scheme, for which he also added corners and cusp and conical points. These extensions were further analyzed and formalized by the work of Jean Schweitzer [Schwe96].

Fig. 4-1. An image taken from Geri's Game. DeRose et al. used subdivision surfaces with sharp and semi-sharp edges to model Geri's head and jacket (©Pixar).

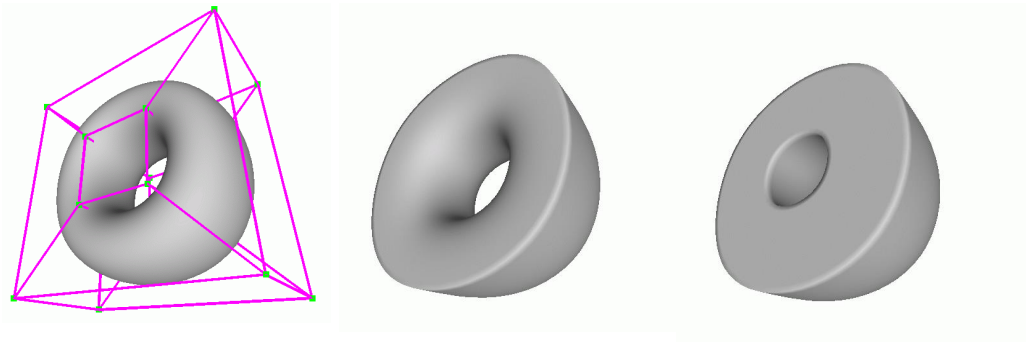
Instead of completely sharp edges, also semi-sharp edges are a desired feature, both for modeling artists and for industrial designers. These semi-sharp edges can be generated in a way similar to the sharp ones. The first few subdivision steps keep the involved points in their original place, but after a user-defined number of steps, also these points revert to the standard rules. To make things even more adapted to the user's needs, this number of steps does not need to be restricted to an integer. When it is set to a fractional number, such as for example 3.45, the first three steps are performed without moving the point. For the fourth step, the fixed position and the position obtained by standard rule are interpolated using the fractional part (0.45 in the example given). And for every subsequent subdivision step, again the

standard rule is used. Figure 4-1 shows an example of sharp and semi-sharp features in Pixar's short animation, Geri's Game [DeRose98].

Sharp edges are most easily implemented on schemes where the original points and edges do not get replaced by multiple new points, but instead are only moved to relax the scheme. Fully interpolating schemes are also difficult to adapt to allow sharp edges, as the existing points are already kept in their place and the surrounding new points would have to fulfill too many constraints.

Therefore, except for the Catmull-Clark scheme, also the Loop scheme (see section 3.8) and the 4-8 scheme (see section 3.10) lend themselves naturally to being adapted to cope with sharp edges. The Sqrt(3) scheme (see section 3.9) is a little more complicated to adapt. This scheme also moves its existing points, but at the same time restructures the connections of the edges, so that an edge is only at its same topological position every two steps.

On the other hand, corner-cutting schemes – like the Doo-Sabin scheme (see section 3.6), the midedge scheme (see section 3.4) and the hexagonal scheme described in chapter 5 – are not so easy to adapt for sharp edges. In these schemes every subdivision step replaces existing points and edges by completely new polygons, making it difficult to indicate an edge that has to be sharp. Tricks like adding an imaginary border and making the surface touch itself can establish a sharp C^0 border. Semi-sharp edges would need an algorithm that first performs some subdivision steps, taking the imaginary border into account, and after a user defined number of steps, ignores that border again, using the standard rules.



ig. 4-2. A control mesh for a smooth Catmull-Clark surface (left); the resulting surface afte
dicating four edges as sharp (center) and after indicating four more edges as sharp (right

Figure 4-2 gives an example of our implementation of semi-sharp edges in the Catmull-Clark scheme [Claes01b]. At the left, a control mesh for a torus, together with the resulting smooth surface is shown. The image in the center is the result of indicating four of the outermost edges as interpolating. The image at the right also has four of the innermost edges indicated as sharp.

4.8 Boundaries

Although the basic formulas for subdivision surfaces are defined for closed meshes, they do not need to stay restricted to closed surfaces. Special rules for borders can simply mimic the behavior of a particular subdivision surface, where newly generated points on the boundary only depend on other boundary points. This simple setup works for face-splitting subdivision schemes with a small support, such as the Catmull-Clark [Catmu78] and the Loop scheme [Loop87]. Levin remarked that this approach sometimes leads to self-overlapping surfaces near the border, and proposed some modification to the subdivision rules for the point near the borders [Levin00].

Making sure that rules on the boundary do not depend on internal points is useful to create creases and sharp edges. The only requirement to get a neatly closed crease is to put the control points for the boundary on the same position for different surfaces.

Doo defined boundaries to the Doo-Sabin scheme by simply doubling the boundary edges [Nasri87]. In his setup, at each boundary edge he placed an artificial rectangle replicating the points of the edge. Nasri described more complex ways to define boundaries [Nasri87].

4.9 Parameterization (texture mapping coordinates)

Rectangular patches that are tensor products, for example of two splines, have a very straightforward texture parameterization. Each point of the surface is the sum of two points, one on each of the spanning curves, letting the point inherit its u and v coordinate from the parameterization of the curves. Similarly, triangular patches have a u and v coordinate limited to a triangular domain.

As subdivision surfaces are defined over meshes exhibiting an arbitrary topology, they lack a simple parameterization. In their paper discussing the subdivision techniques used to create the Geri's Game short animation, Tony

DeRose and coworkers [DeRose98] propose the following approach. Texture coordinates can be attached to the points of the defining mesh, using techniques like planar, spherical and cylindrical projections or letting a user assign them in an interactive way. Afterwards, these coordinates need to be distributed properly over the newly generated points of the little polygons making up the subdivision surface. It turns out that this distribution can be accomplished in exactly the same way as the distribution of the X, Y and Z values of the points of the initial mesh. So, whenever a new point is created (or an existing one is moved), not only is the 3D coordinate calculated, but also the u and the v coordinates are averaged using the same scheme.

This approach does not need to be restricted to texture coordinates. Tony DeRose also introduced scalar values that were distributed in a similar way. In the construction of the Geri's Game animation, darker regions like cavities and bright spots on the surface were marked by assigning scalar values to the control points of the coarse mesh. The subdivision process then distributed these values to every vertex of the subdivided mesh. During the rendering stage, these values were used as part of the procedural shading calculations. This method turned out very effective in allowing the artist to fine-tune the optimal shading he had in mind. Figure 4-3 illustrates this with an actual image taken from Geri's Game.

Fig. 4-3. Geri, from the Geri's Game animation short. DeRose et al. distributed a scalar parameter value over the subdivision surface that represents his jacket. This scalar value was used in the procedural shading to accentuate stiches and folds (© Pixar).

4.10 Multi-resolution editing

The recursive subdivision process is fundamentally linked to multi-resolution analysis and wavelet theory, as described in the book by Stollnitz, DeRose and Salesin [Stoll96]. In a multi-resolution description of an object, a very coarse first approximation is used. Gradually more levels, each one attaching more details, are added. This allows for powerful editing at one level without losing the details added by the lower levels. A multi-resolution description strongly resembles the subdivision process. The main difference is that the subsequent representations of a subdivision surface are all approximations of the same limit surface, while in a multi-resolution representation more information can be added at each level. Eck et al. show how to convert arbitrary polygon meshes into multi-resolution meshes [Eck95].

A multi-resolution representation is also very useful when sending geometric information over the Internet. After receiving the coarsest mesh, the target

computer can already start displaying it, gradually updating the finer details as they get progressively transmitted [Hoppe96, Certa96, Labsik00b].

Khodakovsky et al. show how multi-resolution analysis and wavelet based methods can be used to progressively compress a polygonal mesh [Khoda00]. An underlying subdivision surface scheme allows for meshes with arbitrary topology. An example from their results is shown in figure 4-4.

Fig. 4-4. Using multi-resolution methods to obtain progressive meshes [Khoda00].

An interesting variation on multi-resolution representations is a so-called displaced subdivision surface [Lee00]. Here the fine detail is represented as a scalar-valued displacement over a smooth domain surface. Similar to a multi-resolution representation, these meshes can be used to strongly compress detailed geometric information, but also allow an easier way of editing and animating the surface.

Another example of the power of the subdivision paradigm for multi-resolution meshes is the integration of engraving, embossing and trimming [Bierm01, Litke01]. In a similar context, [Velho01b] shows how a multi-resolution representation can be combined successfully with synthetic procedural texture generators. This is a powerful and general shape modeling paradigm as these textures can be applied at different levels and can be fused together seamlessly. The underlying surface can either be captured or modeled by an artist. As these surface representations can rely on lazy evaluation, the finest detail only needs to be generated when viewed from a close distance.

4.11 Wavelets

Wavelets play a very important role in many different scientific research fields, such as image analysis and signal processing. As wavelets can also be applied to tackle differential equations, the theory behind it has been applied successfully in many studied physical phenomena [Stoll96].

While a Fourier transform converts a signal to a frequency domain, wavelets simultaneously approach the frequency and the time domain. In wavelet analysis, a window is shifted along the signal and for every position the spectrum is calculated. This process is repeated many times with a scaled down version of the window.

Image-processing applications use wavelets to highly compress images in a top-down way or to fill in gaps of missing information in images [Adels87, Lundm01]. In medical image processing, wavelets are used to find correlations between different parts of the image [Laine93].

4.12 Interpolating point sets

Subdivision surfaces have also been used successfully to approximate or interpolate point sets, such as the ones obtained by optical scanners [Hoppe93, Hoppe94, Schwe96]. Nasri [Nasri87] describes an approach to constructing a control mesh for a Doo-Sabin scheme such that this mesh interpolates an initial set of points. Similarly, Halstead et al. [Halst93] moved the control points for a Catmull-Clark subdivision such that the scheme would interpolate its original mesh. As the resulting mesh had unwanted bumps and irregularities, they applied an energy-minimizing process, moving the points of the first and the second subdivision, in order to get the surface as fair as possible. An interesting approach is described by Zhang et al., who use a strategy based on subdivision surfaces and minimizing strain energy to move bad data-points [Zhang01]. These points could be obtained from a laser scanner, or they could be badly located points that appeared during the interactive design of a model.

Another more direct approach comes from Dyn, Levin and Gregory, when they presented their Butterfly scheme (see section 3.10), directly interpolating the initial points of the control mesh. Similar methods have been used by Kobbelt [Kobbe96] and Labsik and Greiner [Labsik00a] for other types of meshes.

4.13 Free-form deformations

Subdivision surfaces and their extension to volumes have been used as a base for free-form animation tools. By moving only the control vertices, the subdivision process smoothly transforms the space between them. MacCracken and Joy used a particular extension of the Catmull-Clark scheme to volumes [MacCr96] and in our work we applied the Loop scheme with local interpolation [Claes00]. In chapter 10 we explain more details about the techniques we used.

4.14 Simulating physical processes

Subdivision schemes combined with multi-resolution methods have been proven successful in studying physical processes like the motion of fluid flow [Weimer99]. In this context, the subdivision schemes serve as a framework to solve differential equations. The same authors, Weimer and Warren, also proved how thin plate splines, as an approximation of bending energy, are a good way to tackle the modeling of "fair" surfaces [Weimer98].

A specific advantage of both subdivision surfaces and subdivision volumes for studying physical processes, are the arbitrary topology, allowing the solution space to be planar, hyperbolic, or free-form. Another advantage is the possibility of keeping the influence between nodes local, highly speeding up the calculations.

Cirak, Ortiz and Schröder show how Loop's subdivision surface scheme can be incorporated in thin-shell finite-element calculations [Cirak00]. Their new paradigm turned out to be highly accurate with an optimal convergence. Especially the fact that Loop surfaces are guaranteed to have a finite bending energy plays a key role in their analysis methods. Moreover, subdivision allows the geometric modeling and the finite-element analysis to use an identical representation, making the implementation more robust. Additional advantages are the integration of non-smooth elements such as boundary conditions and creases (see sections 4.7 and 4.8).

Subdivision is also used in engineering applications, in combination with the finite element method. If not only the outer surface, but also the inner structure of an object is important, subdivision volumes are used [MacCr96, Bajaj01, Mandal99].

McDonnells and Qin used finite elements methods in combination with physics based modeling to simulate the behavior of clay [McDon00]. Qin also

studied howto use subdivision methods to tackle variational problems [Qin98]. Recently Chris Raymaekers and Koen Beets published a paper about how to use subdivision techniques in haptic rendering of a surface [Rayma01].

5 A hexagonal subdivision surface scheme

5.1 Introduction

In this chapter we introduce a new hexagonal scheme for subdivision surfaces. The main idea was born after reading a recent paper by Zorin and Schröder about how the duality between subdivision surface schemes leads to higher-degree continuity [Zorin01a]. They only consider quadrilateral subdivision schemes, as the dual of a quadrilateral scheme is again a quadrilateral scheme, while the dual of a triangular scheme would be a hexagonal scheme. In this chapter we propose such a hexagonal scheme, which can be considered the dual of Kobbelt's Sqrt(3) scheme for triangular meshes [Kobbe00]. Figure 5-1 already gives a first impression of how such a hexagonal scheme would operate on a regular dodecahedron as input mesh. Each time, all corners of the previous mesh are cut off, replacing them with a hexagon.

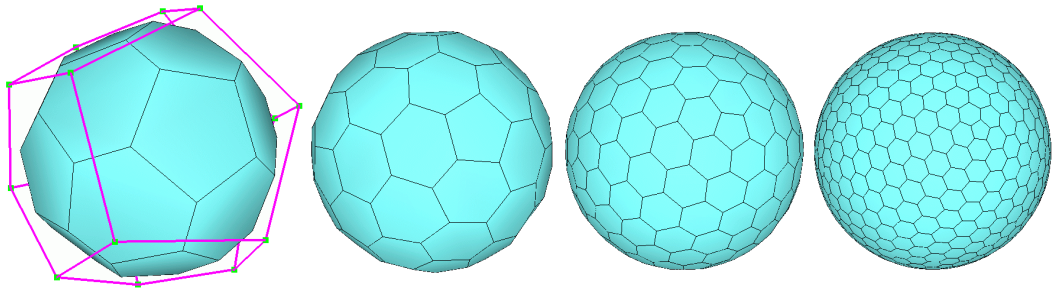


Fig. 5-1. Four consecutive steps of a hexagonal corner-cutting scheme applied to a dodecahedron. Each of the generated polyhedra consists of 12 pentagons and a number of hexagons that triples at each subdivision step. The polyhedron at the right contains 360 hexagons.

We also introduce recursive subdivision rules for meshes with arbitrary topology. These rules have a simplicity comparable to the Doo-Sabin scheme: only new vertices of one type are introduced and every subdivision step removes the vertices of the previous steps. Furthermore, we show the relationship between the scheme and a quadratic subdivision curve, which can be used as a border.

The rest of this chapter is organized as follows. We start with section 5.2 introducing the duality between subdivision surface schemes. In section 5.3 we give an overview of the use of hexagonal meshes in other research fields, followed by a discussion in section 5.4 about the possible ways of recursively subdividing a regular hexagonal mesh. Then, in section 5.5, we propose stationary subdivision rules for a new hexagonal subdivision scheme. Section 5.6 is dedicated to the continuity analysis of the surfaces generated by this new scheme. As many common meshes found in computer graphics environments are triangular, section 5.7 proposes two different constructions to convert these meshes to hexagonal ones. Section 5.8 explains a relation with curves and an idea for creating borders for the hexagonal scheme, while section 5.9 gives some preliminary ideas about adaptive subdivision. In section 5.10 we shortly introduce a method to interpolate the initial control points. Finally, in section 5.11 we show some results, followed by a concluding discussion in section 5.12.

5.2 Subdivision surface schemes and duality

For a recursive subdivision surface scheme to be based on a mesh of one type of polygons, either triangles, quadrilaterals or hexagons have to be considered. These are the only types of polygons that allow a regular tiling of an infinite 2D plane.

A mentioned in chapter 3, in 1978 the first two subdivision schemes were published, both based on quadrilaterals. Ed Catmull and Jim Clark's scheme is a so-called primal scheme, splitting each input quadrilateral on one level into four smaller quadrilaterals on the next subdivision level, after which the new positions are averaged out [Catmu78]. Donald Doo and Malcolm Sabin's scheme operates on the dual mesh of Catmull-Clark's scheme: the roles of points and polygons are interchanged, just as a Voronoi diagram is the dual of a Delaunay triangulation [O'Rour94]. Doo-Sabin's scheme is called a corner-cutting scheme, because it has the visual appearance of recursively cutting away corners of the input polyhedron [Doo78].

In 2001, two research papers were published containing the idea of using repeated averaging to obtain surfaces with a higher degree of continuity [Zorin01a, Stam01]. The starting idea comes from factorizing Lane and Riesenfeld's formula for subdivision curves with any degree of continuity [Lane80]. For example, a subdivision scheme for a B-spline curve of degree six normally needs a support of seven neighboring vertices. The interesting observation made by [Zorin01a] and [Stam01] is that the same scheme can be factorized in six subsequent averaging steps, each time using only two neighboring vertices. For curves, this factorization does not seem to help a lot, but for subdivision surface schemes based on these curves, it brings essential new possibilities. For surface meshes, a neighborhood of seven vertices in every direction would require a region of 49 vertices in a regular quadrilateral mesh. A key issue for subdivision surface schemes however, is that they should also be defined for irregular meshes, making it practically impossible to define how the surrounding region should be sampled for every irregular combination. Subdivision surface schemes already have enough problems defining how to adequately sample a neighborhood of one or maximally two surrounding vertices. The factorization of the subdivision rules, however, allows this higher degree of continuity to be obtained by repeatedly averaging a limited region.

5.3 Hexagonal meshes

Although the ancient Greeks were already convinced of the honeycomb conjecture, it has only recently been proven mathematically: a hexagonal tiling is the least-perimeter way to enclose infinitely many unit areas in the plane [Hales99].

This is just one of the features that makes hexagonal lattices interesting. In the field of wavelet analysis, it has been argued that hexagonal sampling is the optimal sampling strategy for signals that are band-limited over a circular region in the frequency domain [He97]. This is similar to what human eyes are believed to do [Adels87, Watson87, Aznar00]. Their three axes of symmetry (0°, 60° and 120°) give more possibilities compared to the two axes (0° and 90°) of quadrangular lattices.

Lundmark et al. [Lundm99, Lundm01] stated that a recursive subdivision using nearly hexagonal fractal tiles forms a good basis for variable resolution image coding. They used the same strategy for distributing the cells of a cell-phone system with a mesh resolution adapted to the different densities needed in different areas.

Tiles with a fractal boundary that can be replicated to fill an infinite plane have been studied by Vince, who also provides numerous pointers to related work about self-replicating tilings [Vince99]. Recursive regular tilings have also been studied by Cannon et al. [Cannon99, Cannon01].

Sahr and White apply a multi-resolution hexagonal grid to create a spatial database, for example, to cover the earth's surface. They noted that in a hexagonal grid, all adjacent cells of whose corners touch always have an edge in common [Sahr98]. They called this uniform adjacency, as all six neighbors have their center at the same distance. In triangular and quadrilateral meshes, different types of neighbors exist, with some only touching by a corner. In a related type of application, Ferhatosmanoglu et al. [Ferha01] showed how hexagonal partitioning has an optimal I/O cost for querying a spatial database.

Hexagonal meshes can also break the annoying straight-line patterns as seen, for example, in fractal terrain modelers using quadrilateral or triangular grids. These patterns also pop up in subdivision surfaces. Although these patterns can be desirable for artificial objects, for many natural-looking surfaces these patterns are to be avoided.

These promising experiences in other fields of research form a good basis to assume that studying a hexagonal scheme for subdivision surfaces would open a world of new exciting ideas. In the following sections, we'll describe the possibilities to create hexagonal schemes and propose a set of stationary subdivision rules.

5.4 Hexagonal subdivision

Similar to other recursive subdivision schemes, we would like to repeatedly subdivide an initial hexagonal mesh into smaller hexagonal meshes. In order to get simple symmetrical subdivision rules this subdivision should treat all input polygons in an equal way, independent from their position in the mesh.

After some subdivision steps, subdivision schemes locally look like a regular grid. Therefore, we'll first have a look at how to refine a regular hexagonal grid in the plane. It turns out there exist many ways in which a hexagonal grid can be recursively refined. A simple construction to create such a subdivision relation between an original hexagonal grid G_0 and a refined grid G_1, is to choose two arbitrary cells, *a* and *b*, in G_1, as shown in figure 5-2. Then align the center of *a* with a cell *p* in the coarse grid G_0 and scale and rotate the G_1 until the center of cell *b* is exactly aligned to a direct neighbor of *p* in G_0. In figure 5-3 two cells of grid G_0 are drawn in thicker lines, while in figure 5-4 the centers of *a* and *b* are aligned to *p* and its neighboring cell in G_0, shown in figure 5-3.

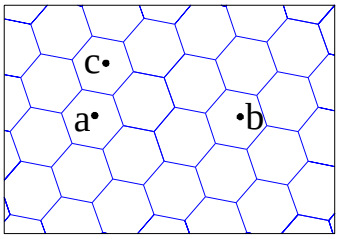


Fig. 5-2. Choosing two arbitrary cells, *a* and *b* in the fine grid G_1. Cell *c* is a direct neighbor of *a*.

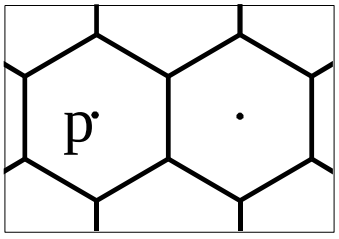


Fig. 5-3. Two cells, *p* and a direct neighbor, in the coarse grid G_0.

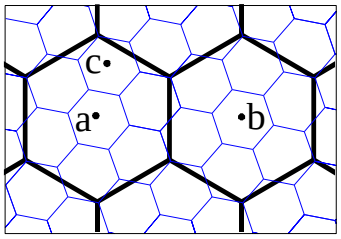


Fig. 5-4. The centers of *a* and *b* get aligned to the centers of *p* and its neighbor.

In this construction, the distance between *a* and *b*, divided by the cell distance in G_1, determines how many times the grid is scaled down by the

subdivision. In the example of figure 5-2, where *c* is such a direct neighbor of *a*, this factor is:

$$\frac{dist(a,b)}{dist(a,c)} = \sqrt{7} \tag{5-1}$$

While the grid is scaled down by this number, the area of each cell is scaled down by the square of this number. Therefore, the square of the scale factor, is the multiplication factor for the cells. In this example, the area is scaled by a factor of seven, allowing the linking of exactly seven cells of G_1 to each cell of G_0.

In figure 5-5 we show an overview of the smallest of the possible multiplication factors. These factors can grow to arbitrarily large numbers, but for a subdivision scheme the small numbers are especially interesting. The cell at the top, marked with "0", represents cell *a* of figure 5-2. For each possible choice for cell *b*, the multiplication factor is written down inside the cell. As all around cell *a* these factors are repeated in a regular pattern, we only show one sextant of the grid.

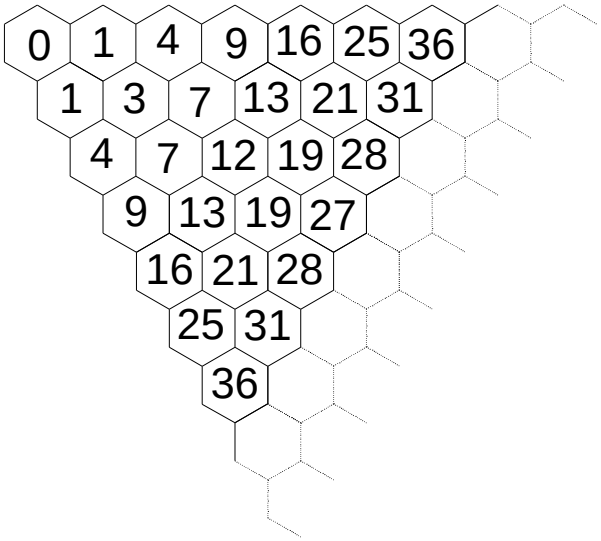


Fig. 5-5. Possible multiplication factors for hexagonal subdivision, depending on the distance where the finer grid is aligned to the coarse one. The actual scaling factor is the square root of the displayed number.

In figure 5-6, we show the first three non-trivial of the possible subdivisions that can be obtained via the above procedure. For the factors three and four, most cells of the finer grid are at the border between two cells of the coarse grid.

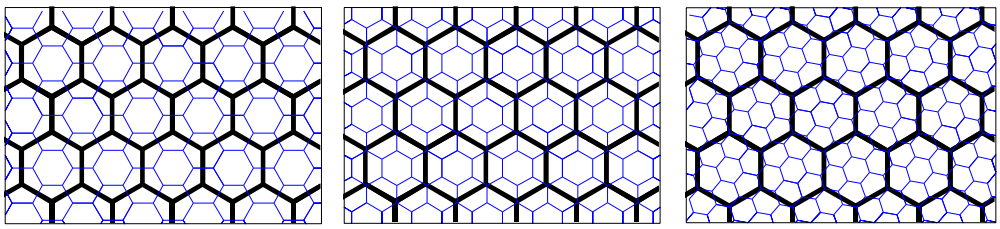


Fig. 5-6. A subdivision of a regular mesh with a factor of three, four and seven. Each time the center of the cell of the finer grid is aligned to the center of the coarse grid.

It turns out that, apart from aligning the center of the finer grid to the centers of the coarse grid, another possible construction is to align the vertices of the finer grid to the centers of the coarse grid [Lundm01]. This would have the

advantage that more of the hexagons of the finer grid can be linked to each single cell of the coarse grid, as is demonstrated in the right image of figure 5-6. When we consider this type of arrangement for our subdivision application, however, we notice that its dual arrangement would have two different types of subdivision for the triangles.

For hierarchical subsampling of images, Lundmark, Wadströmer and Li suggest recursively dividing hexagons into seven smaller hexagons [Lundm99, Lundm01], as in the right image of figure 5-6. In their apparently independent research concerning geodesic mapping, Sahr and White choose a similar approach, after first mapping the earth's surface to a regular icosaheder [Sahr98]. Mathematical properties of this special type of subdivision and its fractal boundary have also been investigated by Vince [Vince99].

From figure 5-5, it is clear that the smallest non-trivial scaling factors are three and four. The scaling factor seven is also appealing, because by partitioning a hexagon into seven, the new hexagons occupy more or less the same area as the original one. With a partition into three or four, some of the new hexagons have necessarily to be shared with the neighbors. A very annoying problem with the scaling factor of seven is that the new mesh is rotated by about 19° (to be precise, asin(sqrt(3/28)), which implies that the repeated subdivision steps lose their alignment with the original mesh. This makes analyses for its use as a subdivision surface scheme quite complicated. Furthermore, this rotation prevents the scheme from being applied in a symmetrical way. Unfortunately, all the scaling factors that are not combinations of three and four are subject to this kind of irregular rotations.

A scaling factor of four is investigated in the paper by Simoens, Dyn and Levin [Simoe01]. A scaled-down version of the input hexagon is dropped in its center, after which the old vertices are reconnected to the nearest new vertices, replacing each old edge by a new hexagon (such as in the center image of figure 5-6). This can be viewed as an edge-cutting operation. This setup introduces one type of new vertex at a similar position as in the Doo-Sabin scheme. However, contrary to Doo-Sabin's scheme, this hexagonal scheme needs to keep the existing vertices. So two types of rules are needed: one for the new vertices, and possibly one for relaxing the position of the old ones. Possible rules for calculating the new vertices and relaxing the old ones can be found in [Simoe01], which also proposes rules to cope with extraordinary vertices. It should be noted that Simoens et al. mainly look at combining hexagonal and triangular schemes, implying that their hexagonal scheme is not treated as an explicit separate scheme. For regular meshes, this

subdivision by a factor of four is also addressed in [Praut01] as an example of the subdivision of half-box-splines. Such a scheme with a scaling factor of four would be the dual of triangular schemes that split triangles into four, such as Charles Loop's scheme [Loop87]. As this scheme for hexagonal subdivision (multiplying the number of hexagons by four in each subdivision step) does not have a fixed name yet, we propose to call it the hexagon-by-four scheme.

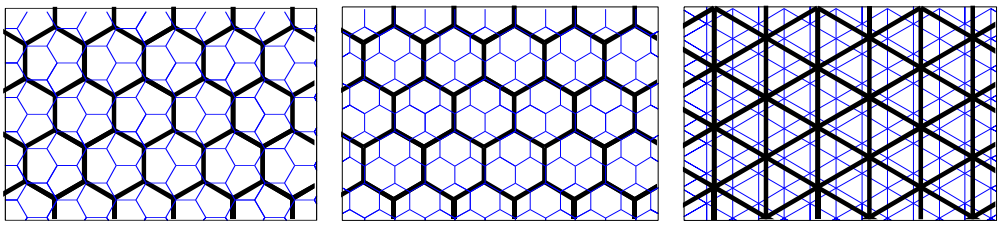


Fig. 5-7. A subdivision of a regular mesh with a factor of three and four (left and center). This time the vertices of the cell of the finer grid are aligned to the center of the coarse grid. At the right, the triangular dual of this subdivision by four is shown. Unfortunately, the triangles are not treated evenly is this dual subdivision.

An observation is that to get a symmetrical subdivision, it is not necessary to align the centers of the new grid with the centers of the old one. An alternative possibility is to align the vertices of the new grid with the centers of the old one, as shown in figure 5-7. This alignment is less interesting however, as rules for more types of new vertices need to be addressed. Furthermore, the dual of such a scheme would be a triangular scheme that no longer treats all of the triangles equally.

Our attention is focussed on a hexagonal subdivision that multiplies the number of hexagons by a factor of three (such as in the left image of figure 5-6). At the same time the average surface area of the hexagons is divided by three, and the average edge length by a factor of sqrt(3). This last observation inspired Kobbelt to call his scheme the Sqrt(3) scheme [Kobbe00]. We propose to simply call the new scheme the hexagon-by-three scheme.

An interesting feature that a hexagon-by-three subdivision shares with the well-known Doo-Sabin scheme [Doo78] is that only one type of new vertex is introduced. In our scheme, the vertex is created in the center of the triangle formed by the two neighboring old vertices and the center of a hexagon. For the quadrilateral Doo-Sabin scheme, the new vertex is created in the center of the quadrilateral formed by the old vertex, the center of the polygon and the

two midpoints of the neighboring edges. Also mimicking the Doo-Sabin scheme, each subdivision step removes all old vertices. A difference with the Doo-Sabin scheme is that our new scheme, except for keeping smaller copies of the existing faces, creates only one type of new face, while the Doo-Sabin scheme creates two types of new faces.

In the next section, we make a proposal for stationary subdivision rules for this new hexagonal scheme.

5.5 Proposed stationary subdivision rules

5.5.1 Subdivision rules in the regular case

In order to search for the most interesting values for the subdivision weights, we make the following considerations. First, we would like the support area to be small such that every control point exhibits only a local influence. Therefore, we look for solutions that are restricted to using the points of the polygon in which we are creating the new points. Furthermore, just as in existing schemes, it makes sense to look for a symmetrical scheme, invariant to the order in which the points are considered and letting all points play an equal role. For the standard mesh (hexagons where every point has valence 3), these considerations lead to the existence of three different weights (see figure 5-8):

- one for the two points closest to the new point (a),
- another weight for the two points in the middle (b),
- and a third weight for the two furthest points (c).

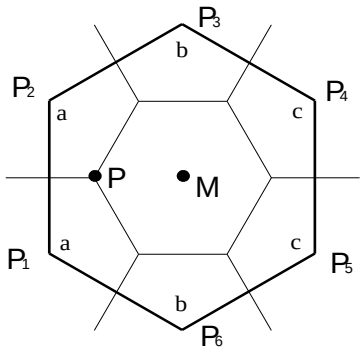


Fig. 5-8. The position of the new point P is a weighted average of the points of the surrounding hexagon.

Expressed using these factors, the equation for the new point is then written as:

$$\mathrm{P} = aP_1 + aP_2 + bP_3 + bP_4 + cP_5 + cP_6 \tag{5-2}$$

Another consideration is that for an input configuration of all equal regular hexagons, the new points should be located such that all newly created small hexagons are again exactly equally sized. As for each input hexagon, three new hexagons are created, and the area of the new hexagons has to be equal to one third of the area of the original ones. To obtain this, the sides of the hexagons have to be divided by a factor of *1/sqrt(3)*. Due to the symmetry of the regular hexagons, this is only possible if the weight $c = a - 1/3$.

For the scheme to be invariant under affine transformations, the sum of these weights should be equal to one: $2a + 2b + 2c = 1$. So, together with the condition on *c*, this condition leads to $b = 5/6 - 2a$.

Putting all this in a matrix, we get the subdivision matrix with only one variable left (*a*).

$$S' = \begin{bmatrix} a & a & 5/6 - 2a & a - 1/3 & a - 1/3 & 5/6 - 2a \\ 5/6 - 2a & a & a & 5/6 - 2a & a - 1/3 & a - 1/3 \\ a - 1/3 & 5/6 - 2a & a & a & 5/6 - 2a & a - 1/3 \\ a - 1/3 & a - 1/3 & 5/6 - 2a & a & a & 5/6 - 2a \\ 5/6 - 2a & a - 1/3 & a - 1/3 & 5/6 - 2a & a & a \\ a & 5/6 - 2a & a - 1/3 & a - 1/3 & 5/6 - 2a & a \end{bmatrix} \quad (5\text{-}3)$$

As the new points are rotating the input points by 30°, this matrix has only complex eigenvalues, which is not suited for our goal to analyze the scheme. Therefore, as in [Kobbe00] and [Labsik00a], we consider a combination of two subdivision steps, after which we rotate the configuration back by 60° to match the original orientation.

This double matrix looks like equation (5-4):

$$S = \begin{bmatrix} 12a^2 - 8a + \frac{29}{18} & -6a^2 + 4a - \frac{4}{9} & -6a^2 + 4a - \frac{5}{9} & 12a^2 - 8a + \frac{25}{18} & -6a^2 + 4a - \frac{5}{9} & -6a^2 + 4a - \frac{4}{9} \\ -6a^2 + 4a - \frac{4}{9} & 12a^2 - 8a + \frac{29}{18} & -6a^2 + 4a - \frac{4}{9} & -6a^2 + 4a - \frac{5}{9} & 12a^2 - 8a + \frac{25}{18} & -6a^2 + 4a - \frac{5}{9} \\ -6a^2 + 4a - \frac{5}{9} & -6a^2 + 4a - \frac{4}{9} & 12a^2 - 8a + \frac{29}{18} & -6a^2 + 4a - \frac{4}{9} & -6a^2 + 4a - \frac{5}{9} & 12a^2 - 8a + \frac{25}{18} \\ 12a^2 - 8a + \frac{25}{18} & -6a^2 + 4a - \frac{5}{9} & -6a^2 + 4a - \frac{4}{9} & 12a^2 - 8a + \frac{29}{18} & -6a^2 + 4a - \frac{4}{9} & -6a^2 + 4a - \frac{5}{9} \\ -6a^2 + 4a - \frac{5}{9} & 12a^2 - 8a + \frac{25}{18} & -6a^2 + 4a - \frac{5}{9} & -6a^2 + 4a - \frac{4}{9} & 12a^2 - 8a + \frac{29}{18} & -6a^2 + 4a - \frac{4}{9} \\ -6a^2 + 4a - \frac{4}{9} & -6a^2 + 4a - \frac{5}{9} & 12a^2 - 8a + \frac{25}{18} & -6a^2 + 4a - \frac{5}{9} & -6a^2 + 4a - \frac{4}{9} & 12a^2 - 8a + \frac{29}{18} \end{bmatrix}$$

This matrix has the following eigenvalues:

$$\lambda_0 = 1$$

$$\lambda_1, \lambda_2 = 1/3$$

$$\lambda_3, \lambda_4 = 36a^2 - 24a + 4 \quad (5\text{-}5)$$

$$\lambda_5 = 0$$

The symmetry of the scheme forces the matrix to be of rank 5, implying that zero is one of the eigenvalues. In the literature [Doo78, Ball86, Ball88, Reif95, Kobbe00, Praut97, Zorin00a, Umlauf00, Sabin01], the following is reported about the nature of the eigenvalues of the subdivision matrix:

- The largest (or dominant) eigenvalue should be 1. This is necessary for the scheme to be affine invariant.
- The second and third largest eigenvalues (sorted by their absolute values) should be equal, and they should be strictly larger than the next eigenvalues for the scheme to be C^1 [Umlauf00]. These are the so-called subdominant eigenvalues. They are the factor by which the configuration shrinks at each subdivision step and control the first derivative. The left eigenvalues belonging to these subdominant eigenvalues determine the tangent plane of the limit surface.
- The fourth and fifth eigenvalues should optimally be the square of the subdominant eigenvalues. These eigenvalues are related to the out-of-the-plane behavior of the configuration.

Consequently, we try to set the fourth and fifth eigenvalue to the square of the subdominant eigenvalue:

$$\lambda_3 = \lambda_1^{\,2} \tag{5-6}$$

which is rewritten as:

$$36a^2 - 24a + 4 = \left(\frac{1}{3}\right)^2 \tag{5-7}$$

This equation leads to the following values for a, b and c (in equation 5-8, see figure 5-8):

$$a = \frac{7}{18} \quad b = \frac{1}{18} \quad c = \frac{1}{18} \tag{5-8}$$

or:

$$a = \frac{5}{18} \quad b = \frac{5}{18} \quad c = \frac{-1}{18} \tag{5-9}$$

The first solution consists entirely of positive weights, while the second solution uses a negative weight. As negative weights result in the loss of the convex hull property for the scheme, we opt for the first solution [Reif95, Melkm97, Kobbe00].

For that first solution, the single- and double-step subdivision matrices around a hexagon would be

$$S' = \frac{1}{18}\begin{bmatrix} 7 & 7 & 1 & 1 & 1 & 1 \\ 1 & 7 & 7 & 1 & 1 & 1 \\ 1 & 1 & 7 & 7 & 1 & 1 \\ 1 & 1 & 1 & 7 & 7 & 1 \\ 1 & 1 & 1 & 1 & 7 & 7 \\ 7 & 1 & 1 & 1 & 1 & 7 \end{bmatrix} \tag{5-10}$$

and

$$S = \frac{1}{54}\begin{bmatrix} 17 & 11 & 5 & 5 & 5 & 11 \\ 11 & 17 & 11 & 5 & 5 & 5 \\ 5 & 11 & 17 & 11 & 5 & 5 \\ 5 & 5 & 11 & 17 & 11 & 5 \\ 5 & 5 & 5 & 11 & 17 & 11 \\ 11 & 5 & 5 & 5 & 11 & 17 \end{bmatrix} \tag{5-11}$$

5.5.2 Rules for the extraordinary case

The rule for the case that the original polygon is a hexagon can be viewed in another way. Instead of formulating it using the six points of the hexagon, we could see it as taking the uniformly weighted average of the two nearby points with the center of the hexagon. This evokes the formulation of the Doo-Sabin scheme [Doo78], where the new point is the average between the nearby original point, the center and the two edge points.

A simple rule for the polygons of the old mesh that are not hexagons is to use the same approach as for hexagons: just take the average of the center and the two nearby points. As in the Midedge scheme [Peters97], however, such weights would mean that polygons with more than six points would shrink more slowly than the hexagons, leading to flat spots on the surface. And polygons with fewer points would shrink too fast, leading to annoying pointy features.

Therefore, as Peters and Reif did for their Midedge scheme, the weights should be adapted to get a uniform shrinking between all types of polygons. If we stay with the idea that the new point should be an average (though not uniformly weighted) between the two nearby points and the center, we would get the following formula (see figure 5-9):

$$\mathrm{P} = w\frac{P_1 + P_2}{2} + (1-w)M \tag{5-12}$$

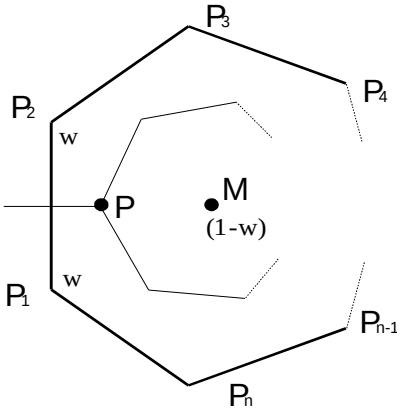


Fig. 5-9. Situation of a polygon in the extraordinary case.

Here *M* is the center of the polygon, calculated as the weighted average of its *n* points. The only way to get the shrinking factor equal to the shrinking by one-third of the hexagon is for *w* to equal:

$$w = \frac{1}{\sqrt{3}\cos(\pi/n)} \tag{5-13}$$

For use in matrix calculations, we should write the formula with *M* being replaced by its original components:

$$\mathrm{P} = \frac{(n-2)w+2}{2n}P_1 + \frac{(n-2)w+2}{2n}P_2 + \frac{1-w}{n}P_3 + \frac{1-w}{n}P_4 + \ldots + \frac{1-w}{n}P_n \tag{5-14}$$

5.5.3 Isolation of singularities

As in the Doo-Sabin scheme, two different kinds of irregularities can be present in the original mesh. First, some of the original vertices can have a valence different from three, which is the standard valence for regular hexagonal meshes. Second, the original mesh can contain polygons that are different from hexagons. The first kind of irregularity disappears after one subdivision step. Each vertex that doesn't have valence three will be converted to a polygon with twice the number of vertices as the valence of the original vertex. If the mesh contains polygons that are not hexagons, the

subdivision process will scale them down and surround them with hexagons. Each subdivision step isolates these non-regular polygons further, in the limit contracting each of them to a point: a so-called extraordinary point. This isolation process is illustrated by figure 5-10.

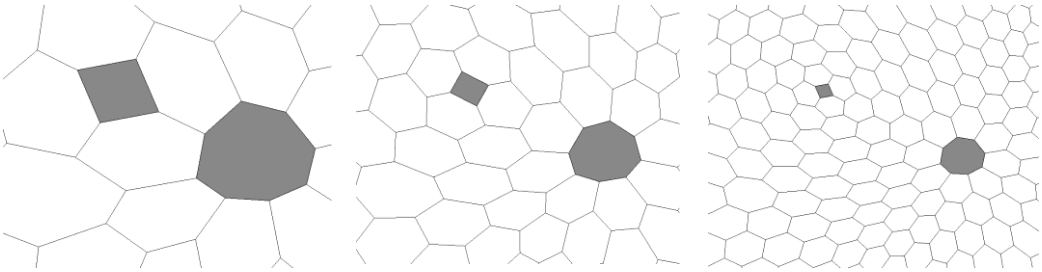


Fig. 5-10. The quadrilateral and the octagon are extraordinary polygons in the original mostly hexagonal mesh (left). Each subsequent step of the subdivision process isolates them further apart, increasingly filling the rest of the space with hexagons (center and right).

5.6 Surface continuity

A standard way to study the properties of a subdivision scheme is to look at the underlying generating function [Catmu78, Doo78, Loop87]. Most existing subdivision schemes are either based on box-splines or are a tensor product of B-splines. Therefore, in the regular case, they are equal to these underlying functions, inheriting among others their degree of continuity. For our new scheme, however, such an underlying function is not known, similar to the situation with the Sqrt(3) scheme [Kobbe00].

We had a closer look at box-splines, as Prautzsch and Böhm [Praut01] already described some kind of regular hexagonal subdivision. In their study of box-splines, they developed a theory of half-box-splines. Box-splines are a generalization of B-splines to higher dimensions and have been used to develop subdivision surface schemes, such as Loop's scheme. Tensor product B-splines, such as the one used for the Catmull-Clark subdivision scheme, are just one of the special cases of box-splines. Prautzsch and Böhm observed that by combining two half-box-splines, the theory of box-splines can be generalized to regular hexagonal grids. We refer to [Praut01] for the details about these half-box-splines and to the book by De Boor, Höllig and Riemenschneider [DeBoor93] for a more comprehensive treatment of box-

splines. Another interesting work in this context is by Mueller, who describes ways to adopt box-splines to model for free-form surface modeling [Muell96].

Unfortunately, the symmetric half-box-splines with small support lead to schemes that are not compatible with our hexagon-by-three scheme. For example, if a ternary subdivision were applied to the symmetric cubic C^1 scheme described in section 4.3 of [Praut01], the resulting mesh would have the same type of connectivity as the double step of our scheme. Its weights are not compatible, however. Using Prautzsch's formulas, we derived a matrix for a ternary subdivision of a hexagonal mesh. This matrix is depicted in equation 5-15 and has rank six, while the symmetry conditions of our scheme oblige dependencies between the rows, yielding a matrix of rank five.

$$S_{boxspline} = \frac{1}{27}\begin{bmatrix} 10 & 4 & 4 & 1 & 4 & 4 \\ 4 & 10 & 4 & 4 & 1 & 4 \\ 4 & 4 & 10 & 4 & 4 & 1 \\ 1 & 4 & 4 & 10 & 4 & 4 \\ 4 & 1 & 4 & 4 & 10 & 4 \\ 4 & 4 & 1 & 4 & 4 & 10 \end{bmatrix} \tag{5-15}$$

So, although the half-box-splines are closely related to our scheme, they don't seem to give direct answers to its nature. The next step in our analysis is to decompose the subdivision matrix into a product of three matrices: a matrix with left eigenvectors, a diagonal matrix of eigenvectors and a matrix of right eigenvectors. Following the analysis of Halstead et al. [Halst93], this decomposition is an essential issue in deriving the limit behavior of the scheme. A condition posed by [Halst93] is that the subdivision matrix may not be defective: the matrix of eigenvectors should be inversible. We demonstrate that our hexagonal scheme accomplishes these conditions.

Equation 5-16 shows this decomposition of the double subdivision matrix *S*. The right eigenvectors are the columns of matrix *R*, while the left eigenvectors form the rows of matrix *L*. As $R \cdot L = L \cdot R$ is equal to the identity matrix, applying the subdivision matrix *n* times to a given configuration of control points can be expressed as $S^n = (R \cdot \Lambda \cdot L)^n = R \cdot \Lambda^n \cdot L$. Therefore, *R* and *L* control the behavior of the configuration as n approaches infinity.

$$S = R \cdot \Lambda \cdot L = \begin{bmatrix} 1 & -1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & 1 \\ 1 & 1 & 0 & 1 & 0 & -1 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \cdot \Lambda \cdot \frac{1}{6} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -2 & -1 & 1 & 2 & 1 \\ 1 & -1 & -2 & -1 & 1 & 2 \\ -1 & 2 & -1 & -1 & 2 & -1 \\ -1 & -1 & 2 & -1 & -1 & 2 \\ -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix}$$

with Λ the matrix of eigenvalues: $\Lambda = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ (5-16)

Following the conclusions of [Halst93], the normalized left eigenvector belonging to the dominant eigenvalue 1 determines the position of the limit point. Here "normalized" should be interpreted as summing to 1. In our case, this is a vector with all components equal to one-sixth. Hence, in the limit of the recursive subdivision process, all vertices of the constantly shrinking hexagon end up in its center.

Further, the normal in the limit position is the plane spanned by $c_2 \cdot P$ and $c_3 \cdot P$. Here c_2 and c_3 are the left eigenvectors belonging to the second and third largest eigenvalue. P is the column matrix representing the neighborhood of the extraordinary point that is being investigated. Therefore, these two vectors spanning the limit tangent plane are:

$$\begin{aligned} c_2 \cdot P &= \frac{1}{6}(-P_1 - 2P_2 - P_3 + P_4 + 2P_5 + P_6) \\ c_3 \cdot P &= \frac{1}{6}(P_1 - 1P_2 - 2P_3 - P_4 + 1P_5 + 2P_6) \end{aligned} \quad (5\text{-}17)$$

Eigenanalysis shows that the generated surface of our hexagonal scheme is C^1 and probably C^2 in the regions where the mesh is regular. In the environment of extraordinary vertices the surface is probably C^1. These hypotheses are further endorsed by studies like [Peters00] revealing that

almost all subdivision schemes are C^1 continuous, especially when they are purely based on positive weights. Further investigations are needed to understand more about the properties of the kind of surfaces generated by our scheme. As can be verified by [Zorin00b], rigorous analysis of this kind of subdivision surface schemes can be very mathematically involved.

Reif noted that another condition for a subdivision scheme to behave properly is that the characteristic map needs underlying analyzable basis functions and should be regular and injective (without local self-intersections) [Reif95]. The characteristic map defines a local parameterization based on the subdominant eigenvectors.

As rigorously checking the characteristic map can be rather complicated, many researchers [Schwe96, Kobbe00, Simoe01] suggest a visual check on the characteristic map to verify that the scheme behaves well around extraordinary vertices. In figure 5-11 we show a visualization of the characteristic map for polygons with three to ten edges. These characteristic maps show a very uniform and smooth behavior.

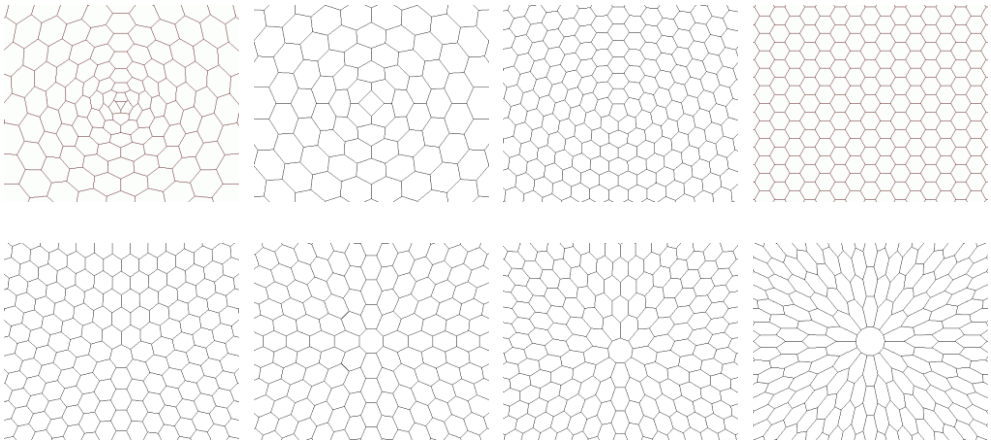


Fig. 5-11. Characteristic map around polygons with three to ten vertices. The closer the number of edges is to the preferred number of six, the more regular the characteristic map.

5.7 Converting triangular to hexagonal meshes

In general, polygonal meshes found in computer graphics environments mainly consist of triangles and quadrilaterals. Meshes that mostly contain

hexagons are quite rare. The meshes obtained from 3D laser scanners for example are typically arranged in a triangular way, exhibiting an arbitrary topology. In this section, we'll describe ways of converting triangular meshes to mainly hexagonal ones. Meshes containing polygons with more than three vertices can easily be triangulated using standard techniques [O'Rour94].

Our hexagonal scheme is not the only one that requires a preprocessing step to make general meshes more suitable. Also Velho and Zorin [Velho01a], who recently introduced the promising 4-8 scheme, require adapting meshes that were not especially designed for their scheme.

It turns out that two suitable methods to convert a triangular to a hexagonal mesh can be constructed: namely, by cutting the corners of the triangles or by replacing the mesh by its dual. These methods are described in the following subsections. At this stage, we suppose the input is a closed manifold triangular mesh, exhibiting an arbitrary topology. The handling of borders will be discussed in section 5.8.

5.7.1 Replacing the triangular mesh by its dual

A first way to convert a triangular to a hexagonal mesh is to replace the mesh by its dual. All triangles of the original mesh will be converted to vertices, and all vertices of the original mesh will be converted to polygons. For each vertex, the centers of the surrounding triangles are connected to form a polygon. The number of sides is determined by the valence of the original vertex. In figure 5-12, the vertices with the regular valence of six in the triangular mesh are converted to hexagons, while valence five vertices are converted to pentagons. While the polygons created by this process can have an extraordinary number of vertices, the newly created vertices themselves all have a valence of three, as would be needed for a regular hexagonal mesh. Luckily, for all subsequent subdivisions, the only newly created polygons will be hexagons.

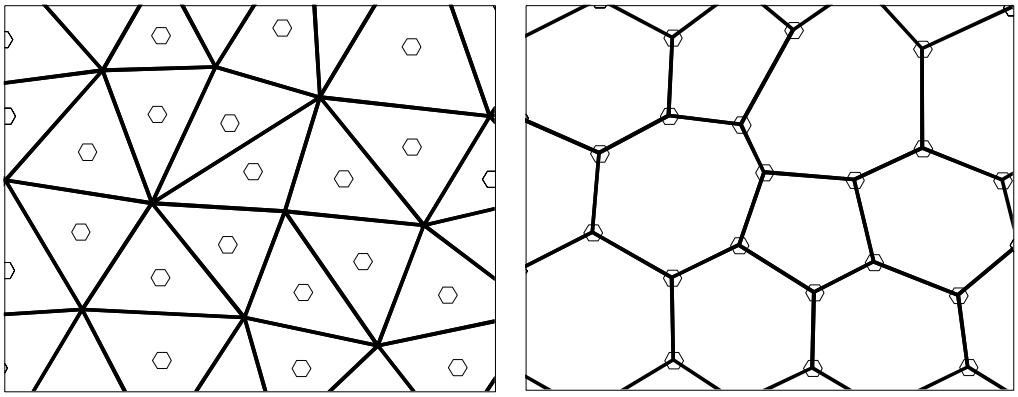


Fig. 5-12. Left: In a triangular mesh the centers of the triangles are marked. Right: These centers are used to construct the dual hexagonal mesh.

5.7.2 Cutting the corners of the triangles

Another method for converting a triangular mesh to a hexagonal is to cut the edges of the triangles at one-third and at two-thirds. Suppose we have an edge between points P_1 and P_2. The position of a new point P, at one-third of the edge and next to P_1, would then be:

$$P = \frac{2}{3}P_1 + \frac{1}{3}P_2 \tag{5-18}$$

Inside each existing triangle a new hexagon is formed by joining the shortened edges and cutting away the corners of the triangle. Figure 5-13 illustrates this process. The neighborhood around the old vertices is filled in again by replacing the vertex with a polygon, with a number of sides equal to the valence of the old vertex. As in the previous section, only if the old vertex has a regular valence of six will a hexagon be created, and thus polygons with any number of edges are possible. Luckily, the subdivision process quickly isolates these extraordinary cases.

In the newly generated hexagonal mesh, all vertices have a valence of three, as all of them are at the border of two old triangles converted to hexagons and one new polygon replacing the vertex.

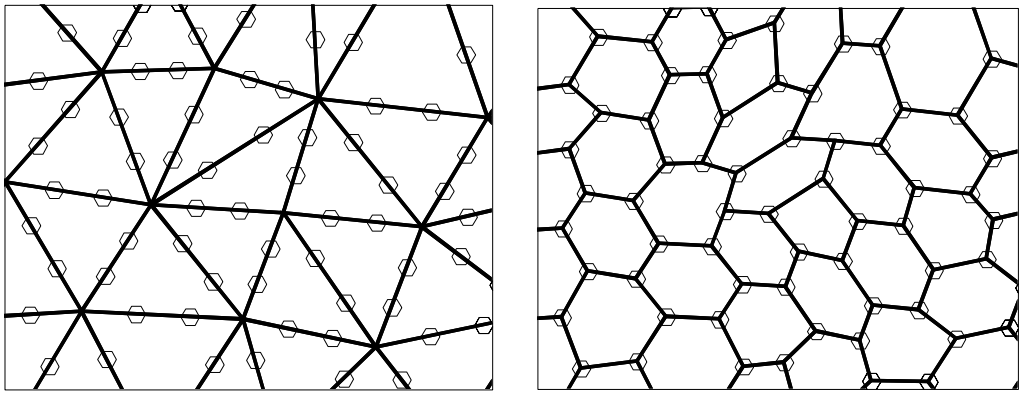


Fig. 5-13. Left: In the triangular mesh of figure 5-12, points now mark the division of the edges at one-third and two-thirds. Right: Using the marked points to convert the triangular mesh to a hexagonal one.

A first impression of figure 5-13 is that the mesh looks rather irregular: although the triangular mesh looks quite homogeneously distributed, the generated polygons are not as convex as one would expect. This is because a stencil of only two nearby points is used to calculate the position of the newly introduced points. The next possible stencil would also include the two extreme points of the two triangles sharing the edge. If P_3 and P_4 are the points that together with P_1 and P_2 form the triangles sharing the edge for the new point, a good formula for the position of the new point would be:

$$P = \frac{5}{9}P_1 + \frac{2}{9}P_2 + \frac{1}{9}P_3 + \frac{1}{9}P_4 \tag{5-19}$$

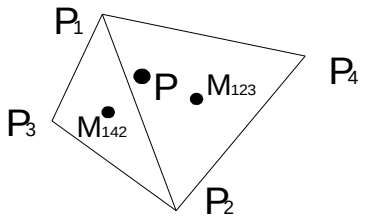


Fig. 5-14 Two triangles sharing the edge near which a new point will be inserted.

Other weights would also be possible, as long as the new point would be positioned at one-third of the edge in the case of both triangles being equilateral. The suggested weights seem to be natural, as they uniformly average the two centers of the triangles with P_1 (M_{123} and M_{142} in figure 5-14). The result is shown at the left of figure 5-15.

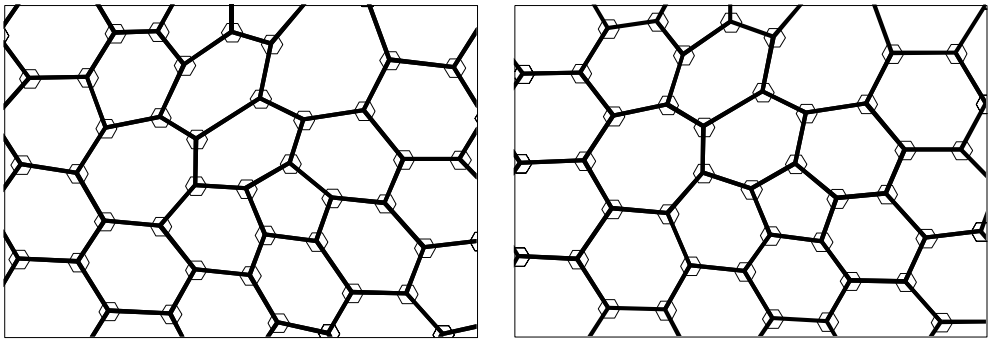


Fig. 5-15. Left: The mesh obtained by corner-cutting the triangles, with additional averaging by direct neighbors. Right: The mesh obtained by subdividing the dual mesh once, resulting in a more regular appearance.

5.7.3 Which method is better?

It turns out that when the mesh obtained via the dual method of section 5.7.1 is subdivided once, it is topologically equivalent to the mesh obtained by the splitting method of section 5.7.2. The meshes have the same number of polygons and vertices arranged in exactly the same way. The positions of the vertices, however, are slightly different. With the dual-mesh method, the positions will be averaged with a support area of $n+1$ in the original triangular mesh, where n is the valence of the nearest point. On the contrary, the method of section 5.7.1 only uses four surrounding points.

Which method to choose depends on whether the application at hand prefers a smoother surface versus a surface that more strictly follows the original edges. For meshes resulting from an optical scanner for example, the dual-conversion method will be preferred. Usually those meshes are dense and demonstrate small accuracy errors, which will be nicely smoothed by the conversion to hexagons. For an interactive modeling application, the one-third - two-thirds method looks more appealing, as the user will have a higher control over the local form of the surface.

5.8 Curves and borders

By putting hexagons symmetrically around the control points of a quadratic subdivision curve, the resulting subdivision surface will interpolate that curve. Under these conditions, the subdivision rules mimic a ternary division of the cubic curve (see section 2.2). This is similar to how the four-point subdivision scheme for curves is embedded in the Butterfly scheme [Dyn90].

For a ternary refinement of a quadratic curve, [Sabin01] derived the coefficients of equation 5-20. The first formula relaxes the position of an old control point, while the second and third define the position of the newly inserted vertices.

$$p_0' = \frac{1}{9}\left(p_{-1} + 7p_0 + p_1\right)$$

$$p_1' = \frac{1}{9}\left(6p_0 + 3p_1\right) \qquad \text{(5-20)}$$

$$p_2' = \frac{1}{9}\left(3p_0 + 6p_1\right)$$

While on triangular schemes, the related curve is defined by the control vertices, for our dual-hexagonal scheme it is defined by the centers of the edges, as with the Doo-Sabin scheme [Nasri01b]. Nasri also showed how to create a border by virtually mirroring the mesh polygons near the border for the Doo-Sabin scheme. A similar approach could be worked out for our hexagonal scheme.

The simpler approach of letting vertices fall together near the border also gives a reasonable result for the Doo-Sabin scheme. The border of a hexagonal mesh is more complicated and does not allow for such an approach.

5.9 Adaptive subdivision

Although our hexagonal scheme has only a multiplication factor of three, as in any subdivision surface scheme, the number of polygons generated grows exponentially. Therefore, most practical implementations of subdivision surface schemes build in an adaptive subdivision strategy. Depending on user-controlled stopping criteria the mesh will be subdivided less into regions that are relatively flat, and more into heavily curved regions.

With the hexagonal scheme, as in most other adaptive subdivision approaches, adaptive subdivision is possible if the subdivision of neighboring polygons never differs more than one level. Effectively implementing such an adaptive subdivision is a topic of future research.

5.10 An interpolating variant

Analogous to the approach of Halstead et al. [Halst93], instead of just approximating the original vertices, we can adapt the scheme to interpolate these vertices. By the scheme's construction, it is clear that each polygon constantly shrinks towards its center, in the limit being contracted completely to that center.

In Halstead's constructions for the Catmull-Clark scheme, the vertex only interpolated the desired control point in its limit position. Every finite number of subdivision steps only approximates the chosen control point. With our hexagonal scheme, however, the center of the polygons will be interpolated during every subdivision step, allowing an application to interpolate a desired set of points even after executing just a few subdivision steps.

Therefore, we implemented a variant to Halstead's approach, where we move the vertices of the hexagonal mesh in an iterative process, to interpolate the vertices of the original triangular mesh. We choose an iterative approach instead of solving a set of equations. Particularly, the equations leave some degree of freedom, which permits our approach to simultaneously optimize other constraints, such as maximizing the hexagons' convexity.

5.11 Results

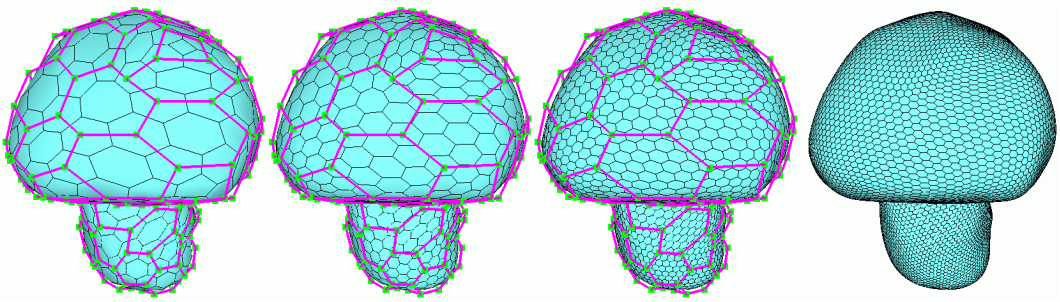


Fig. 5-16. Four steps of the recursive hexagonal subdivision on a mushroom mesh.

Figure 5-16 shows four subsequent steps of the subdivision process for an irregularly modeled mushroom.

Figures 5-17 and 5-18 show the result of different subdivision schemes for a mesh with an extraordinary vertex of valence 20. After three subdivision steps, the Loop subdivision of figure 5-17 multiplies the initial number of triangles by 64 and is comparable to the Sqrt(3) subdivision which multiplies the number of triangles by 81 after four steps. Figure 5-18 uses the dual of the triangular mesh. The hexagonal subdivision multiplies the number of polygons by about 81. The optimized subdivision rules result in the 20-gon shrinking at a pace that is more comparable to that of the hexagons.

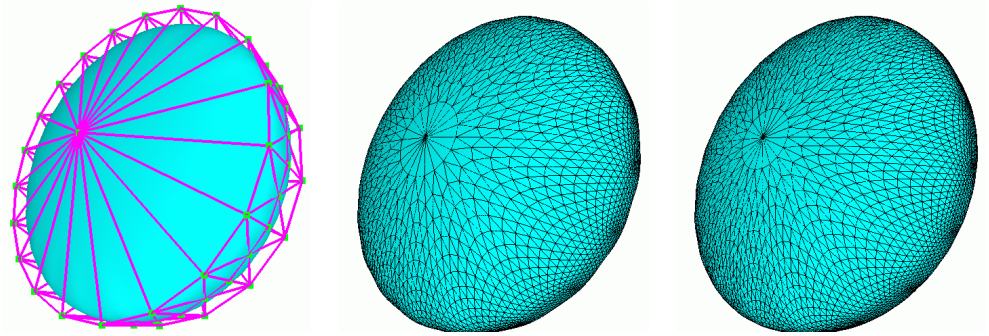


Fig. 5-17. Left: A triangular mesh with a vertex of valence 20 (160 triangles). Center: A Loop subdivision after three steps (10,240 triangles). Right: A Sqrt(3) subdivision after four steps (12,960 triangles).

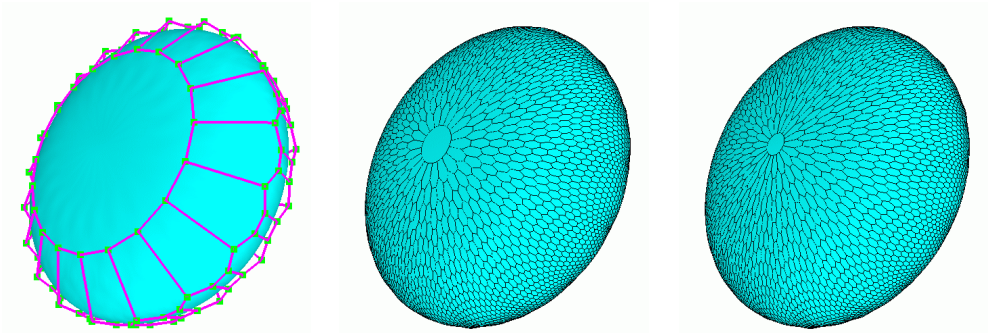


Fig. 5-18. Left: The dual hexagonal mesh from the mesh of the previous figure (82 polygons). Center: A hexagon-by-three subdivision after three steps (6,486 polygons) and using the simple rules. Right: The same scheme using the optimized rules (also 6,486 polygons).

Figure 5-19 shows three subsequent steps in the hexagonal subdivision of a cat model, and in figure 5-20 a Phong-rendered image of the mesh generated by the third subdivision level of figure 5-19 is shown.

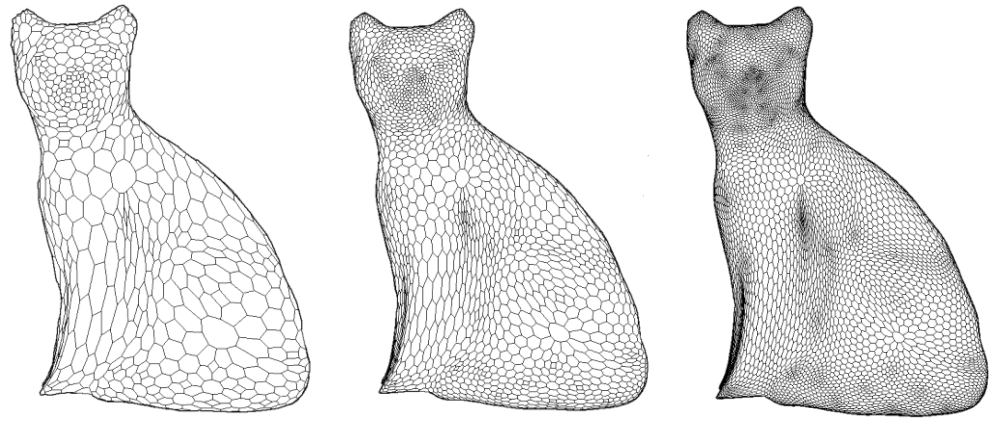


Fig. 5-19. Three consecutive steps of the Hexagon-by-three scheme.

Fig. 5-20. A Phong-rendered image of the third subdivision level in figure 5-19.

In figure 5-21, three different approximating schemes operating on triangular meshes are compared. All three schemes lead to similar results, showing that the new hexagonal scheme is an equal competitor with the existing subdivision schemes.

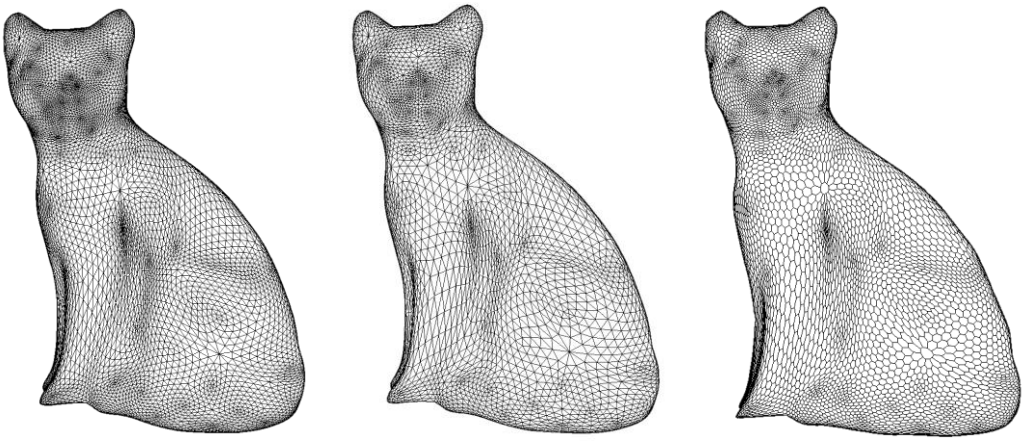


Fig. 5-21. A cat model, comparing different subdivision schemes: subdivided three times using the Sqrt(3) scheme (left), two times using Loop's scheme (center) and three times using the Hexagon-by-three scheme (right).

Finally, in figures 5-22, 5-23 and 5-24, we compare the different corner-cutting schemes. In figure 5-22, four subdivision steps for a cube with the Midedge scheme are shown [Peters97], while figure 5-23 shows four steps of our new hexagonal scheme using the same cube. And in figure 5-24, the Doo-Sabin subdivision is shown [Doo78]. Most notable, is the difference in convergence speed. The Midedge scheme only doubles the number of polygons in each step, our hexagonal triples this number, and the Doo-Sabin scheme quadruples it.

All three schemes shrink the original squares by a factor that is the square root of the multiplication factor of the scheme. For the Doo-Sabin scheme, this is the integer factor of 2, for the Hexagon-by-three scheme this is sqrt(3) and for the Midegde scheme this is sqrt(2). Both the hexagonal and the Midedge scheme constantly rotate these original squares, while the Doo-Sabin scheme maintains their original orientation.

A cube is a very interesting object for comparing quadrilateral and hexagonal schemes. The faces of the cubes have the preferred number of vertices for the quadrilateral schemes, while the vertices have the preferred valence for hexagonal schemes.

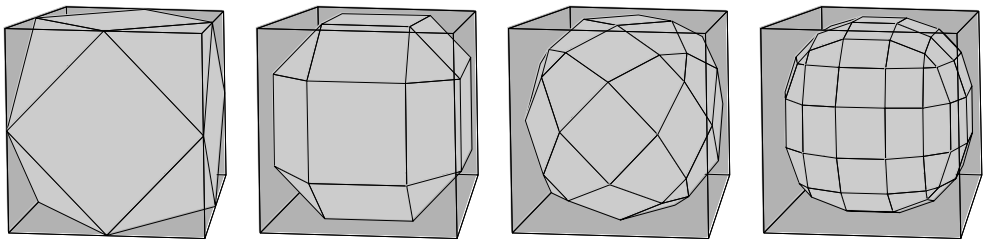


Fig. 5-22. Four steps of the Midedge scheme subdividing a cube.

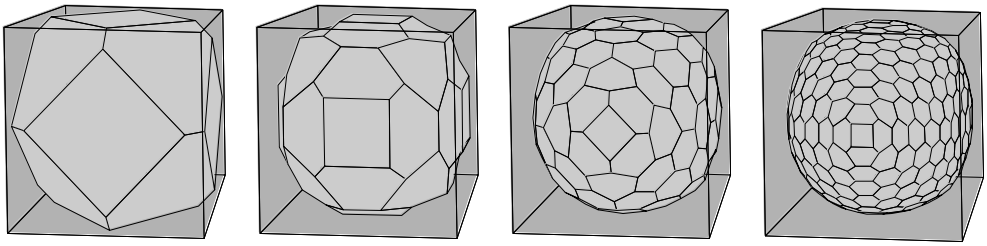


Fig. 5-23. Four steps of the Hexagon-by-three scheme on a cube.

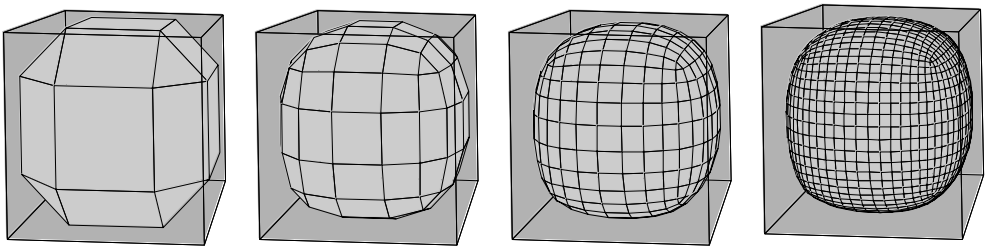


Fig. 5-24. Four steps of the Doo-Sabin scheme subdividing a cube.

5.12 Discussion

In this chapter we proposed a new hexagonal subdivision surface scheme. When operating on a triangular input mesh, the mesh is first transformed to its dual by putting new faces in the place of old points, and new points at the center of the old faces. The scheme uses a minimum of existing vertices to generate smooth surfaces from meshes of arbitrary topology. As the scheme

consequently cuts corners of polygonal meshes, only one type of new points is generated; consequently, the scheme has a simplicity comparable to the well-known Doo-Sabin scheme.

Considering the recent work of [Zorin01a] and [Stam01] about using dual-quadrilateral schemes to create surfaces with a high continuity, the new scheme would be an ideal candidate to be used together with Kobbelt's Sqrt(3) scheme for a similar setup for triangular and hexagonal schemes.

Furthermore, a close relation of the new hexagonal scheme with a ternary subdivision for quadratic B-spline curves is shown. These curves could be used to form the border of the generated surfaces and to create sharp edges.

As in the Sqrt(3) scheme, each subdivision step multiplies the number of polygons by a factor of three. Fewer polygons are created by each step compared to the standard algorithms that have a multiplication factor of four. Therefore, the mesh that meets the requirements for a close enough approximation of the limiting surface will in general be smaller.

6 Local interpolation for subdivision curves

6.1 Introduction

As mentioned in chapter 2, fully interpolating schemes are not very well suited for interactive curve design. Approximation schemes are much easier to manipulate, but sometimes it is desired that the curve locally interpolates one or more of its control points. In this chapter, we show how we can get the best of both worlds, introducing a new method for turning approximating recursive subdivision techniques into an interpolating modeling tool. The approach is based on generating local invariances for the subdivision process around the control points to be interpolated, and allows normal interpolation as well as tension control. The underlying methodology is explained and implementation results and applications are elucidated. We also dedicated a research paper to this topic [VanRe01].

The main objective of the work reported upon in this chapter is to introduce a method for transforming these approximating uniform subdivision schemes into an interpolating subdivision curve design and manipulation tool, at the same time achieving additional advantages. B-splines of any degree as limit curves can be generated, with optional normal interpolation as well as tension control around the vertices c_i^0 of the initial control polygon P^0. Our method mainly involves introducing suitable ghost points around the vertices c_i^0. In the next section (section 6.2) we develop the technique in the context of uniform cubic B-splines. In section 6.3 we generalize the method to B-spines of degree n+1, and in section 6.4 we give some examples illustrating the method. We finish with conclusions and views on further work (section 6.5).

6.2 Control point interpolation, normal interpolation and tension control: the cubic case

Recall the *splitting* and *averaging* process described in chapter 2. Given a control polygon P^j at level j in the subdivision process, the splitting step generates an intermediate control polygon $\widehat{P}^{j+1}$ that contains all the control points of P^j, as well as additional control points inserted at the midpoints of all the edges constituting P^j. This narrows down to:

$$\widehat{c}_{2i}^{\,j+1} = c_i^j \text{ (vertex split point)} \tag{6-1}$$

and

$$\widehat{c}_{2i+1}^{\,j+1} = \tfrac{1}{2}(c_i^j + c_{i+1}^j) \text{ (edge split point)} \tag{6-2}$$

In order to get the final positions of the control points c_i^{j+1} in P^{j+1}, the intermediate control points $\widehat{c}_i^{\,j+1}$ in $\widehat{P}^{j+1}$ are averaged using a so-called *averaging mask* $r = (r_k)_{-m \le k \le m}$ (the exact meaning of m will be given in section 6.3):

$$c_i^{j+1} = \sum_{k=-m}^{m} r_k \widehat{c}_{i+k}^{\,j+1} \tag{6-3}$$

In Chaikin's algorithm [Chaik74] the averaging mask is r = ½ (0, 1, 1). Riesenfeld [Riese75] was able to show that the curves generated by Chaikin's algorithm are uniform quadratic B-splines. It is proven by Lane and Riesenfeld [Lane80] that Chaikin's algorithm can be generalized to generate uniform B-splines of degree $n+1$ by using an averaging mask with entries

$$\frac{1}{2^n}\left(\binom{n}{0}, \binom{n}{1}, \ldots, \binom{n}{n-1}, \binom{n}{n}\right). \tag{6-4}$$

The key insight of the proposed method exploits the fact that, under certain conditions, a generally approximating subdivision scheme can yield a locally interpolating result. Consider the case in which the averaging mask $r = \frac{1}{4}(r_{-1}, r_0, r_1) = \frac{1}{4}(1, 2, 1)$ is used. As mentioned in section 2.2, applying this averaging mask in the subdivision scheme of an initial control polygon P^0 generates an approximating cubic B-spline as the limit curve. An important case is that when:

i. three successive control points c_{i-1}^j, c_i^j and c_{i+1}^j are collinear, and

ii. c_i^j is the midpoint between c_{i-1}^j and c_{i+1}^j

After the splitting step, the intermediate control points $(\widehat{c}_{2i+k}^{j+1})_{-2\le k\le 2}$ are generated around c_i^j according to equation 6-1. This situation is schematically depicted in figure 6-1.

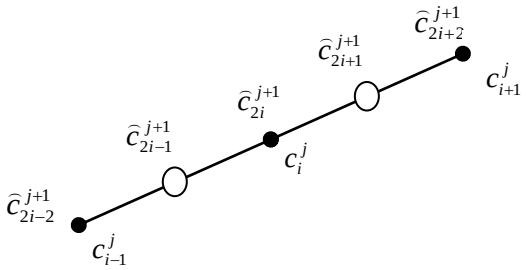


Fig. 6-1. Specific conditions on successive control points (cubic case).

It is easy to verify (cfr. Appendix 1, at the end of this dissertation) that the averaging step will not change the position of $\widehat{c}_{2i-1}^{j+1}$, $\widehat{c}_{2i}^{j+1}$ and $\widehat{c}_{2i+1}^{j+1}$, and that we have a situation in which:

(i) three successive control points c_{2i-1}^{j+1}, c_{2i}^{j+1} and c_{2i+1}^{j+1} are collinear,

(ii) c_{2i}^{j+1} is the midpoint between c_{2i-1}^{j+1} and c_{2i+1}^{j+1}, and,

(iii) $c_{2i}^{j+1} = c_i^j$

Here the subdivision rules generate a so-called local invariance around c_i^j with respect to the averaging rules, and the approximating scheme becomes a locally interpolating scheme around c_i^j. Further, where the straight-line situation shown in figure 6-1 is present in the initial control polygon P^0 (i.e. when $j=0$), the above scheme will interpolate c_i^j in each iteration of the subdivision process. This property is fundamental in the proposed curve design method.

Indeed, around each control point c_i^0 of P^0 (cfr. figure 6-2, in which the dashed line represents a part of the initial control polygon P^0) through which

an interpolatory condition is required (possibly all initial control points), additional points – so-called ghost points $g_{i,-1}^0$ and $g_{i,+1}^0$ – are introduced in such a way that a local invariance is introduced: $g_{i,-1}^0$, c_i^0 and $g_{i,+1}^0$ are collinear, and c_i^0 is the midpoint between $g_{i,-1}^0$ and $g_{i,+1}^0$. The new configuration around c_i^0 is depicted in solid lines in figure 6-2.

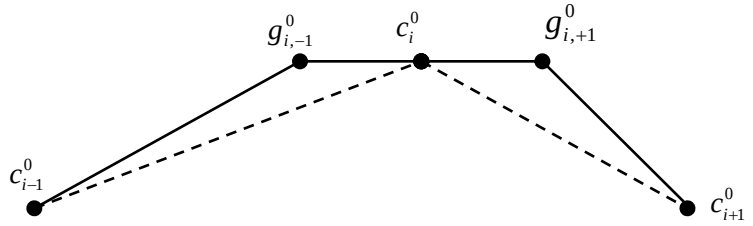


Fig. 6-2. Introduction of additional ghost points.

Extending the initial control polygon P^0 with all the additionally created ghost points and subsequently applying the uniform recursive subdivision rules with the averaging mask $r = ¼\ (r_{-1},\ r_0,\ r_1) = ¼\ (1,\ 2,\ 1)$ will hence lead to an interpolatory subdivision curve with a cubic B-spline as limit curve. The normal to the curve in an interpolatory c_i^0 is perpendicular to the line passing through the ghost points at issue (i.e. $g_{i,-1}^0$ and $g_{i,+1}^0$). Setting the orientation of this line perpendicular to the desired normal in c_i^0 implies normal interpolation. Changing the (equal) distances from the ghost points to c_i^0 will affect the tension of the resulting curve around c_i^0.

6.3 The general cases

We now show that it is possible to generate an interpolatory uniform subdivision curve with normal interpolation and tension control, which has a B-spline of degree $n+1$ as limit curve. Consider the case in which the averaging mask of equation 6-4 is used for a given n. Since the results are slightly different depending on n being odd or even, we first treat the case *n even*.

Case n is even:

Here we have:

(i) 2m + 1 successive collinear control points $(c_{i+k}^j)_{-m\le k\le m}$, and

(ii) each of the control points c_{i-k}^j and c_{i+k}^j $(0 < k \le m)$ are equidistant to c_i^j.

After the splitting step, the intermediate control points $(\widehat{c}_{2i+k}^{j+1})_{-2m\le k\le 2m}$ are generated around c_i^j according to equation 6-1 and 6-2.

The situation we have after the splitting step is schematically depicted in figures 6-3a (for *m* odd) and 6-3b (for *m* even).

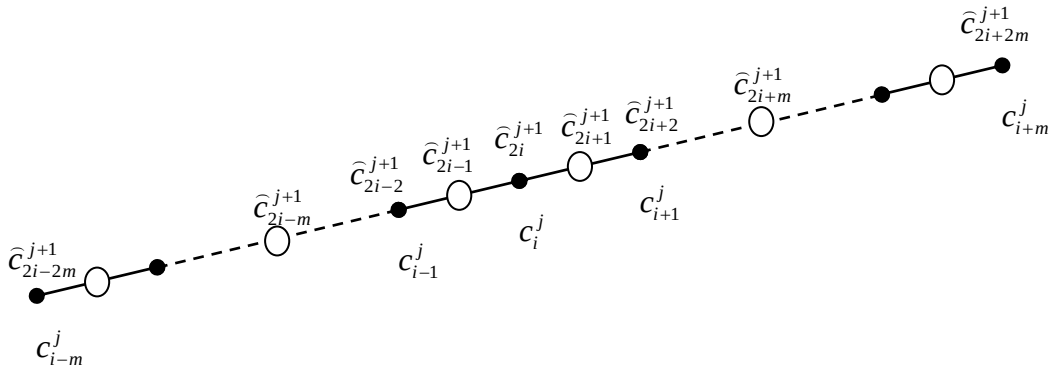


Fig. 6-3a. Specific conditions on successive control points in the general case, for *m* odd.

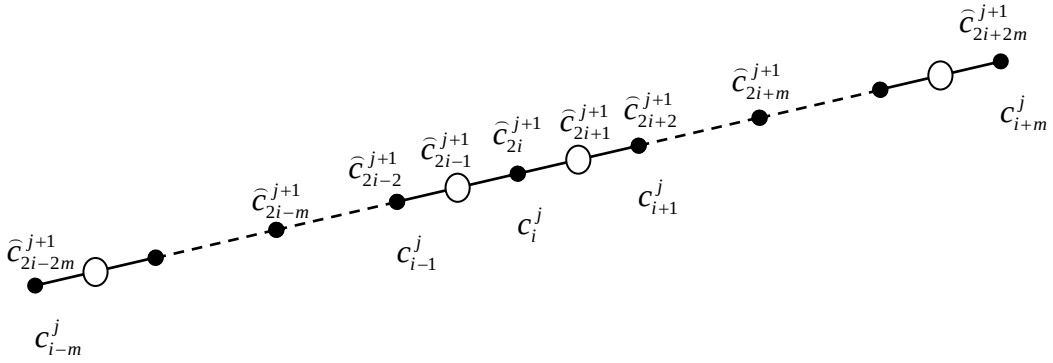


Fig. 6-3b. Specific conditions on successive control points in the general case, for *m* even.

We now have an odd number (*n+1*) of normalized binomial coefficients in the averaging mask and *m = n/2*. (We note that, if m is even, then $\widehat{c}_{2i-m}^{\,j+1}$ and $\widehat{c}_{2i+m}^{\,j+1}$ will be vertex split points; otherwise, they will be edge-split points; hence, the reason for the difference of the points at issue in figures 6-3a and 6-3b.) Because the *2m+1* successive control points $(c_{i+k}^{j})_{-m\le k\le m}$ are collinear and equidistant, the intermediate control points $(\widehat{c}_{i+k}^{\,j+1})_{-2m\le k\le 2m}$ are collinear and equidistant as well. Furthermore, in Appendix 2 (at the end of this dissertation) it is shown that after the averaging step, we have a situation in which the control points $(c_{i+k}^{j+1})_{-m\le k\le m}$ around c_i^{j+1} are on the same position as the intermediate control points $(\widehat{c}_{i+k}^{\,j+1})_{-m\le k\le m}$. Since this invariance is independent of *j*, on each level we have an interpolating subdivision scheme around c_i^{j}.

We now follow the approach as set up in section 2.2 to come to an interpolating subdivision curve design tool. We introduce ghost points $(g_{i,-k}^{0})_{0<k\le m}$ and $(g_{i,+k}^{0})_{0<k\le m}$ at each control point c_i^0 of P^0 we wish to interpolate. These points are introduced so that $(g_{i,-k}^{0})_{0<k\le m}$, c_i^0 and $(g_{i,+k}^{0})_{0<k\le m}$ are collinear and the ghost points are equidistant from c_i^0, as in figure 6-4 where the original control polygon is shown dashed. Here the new configuration around c_i^0 is depicted with solid lines.

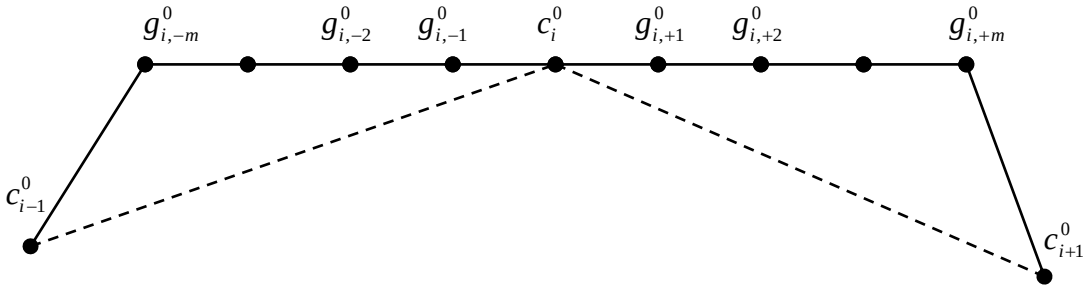


Fig. 6-4. Introduction of additional ghost points.

The generalization of the cubic case to B-splines of degree *n+1* follows straightforwardly if we use the rule in equation 6-4 to generate our averaging mask. Here the normal to the curve in an interpolatory c_i^0 is perpendicular to the line passing through the ghost points $(g_{i,-k}^0)_{0<k\le m}$ and $(g_{i,+k}^0)_{0<k\le m}$ at issue. Setting the orientation of this line perpendicular to the desired normal in c_i^0 implies normal interpolation. Changing the equidistances from the ghost points to c_i^0 will affect the tension of the resulting curve around c_i^0.

Case n is odd:

In the general case, we have to deal with the possibility of *n* being odd. In this case, *n+1* is even and we will have an even number of normalized binomial coefficients in the averaging mask. Accordingly, the averaging step will move the intermediate control points $(\bar{c}_{2i+k}^{j+1})_{-m\le k\le m}$ away from the position of an initial control point c_i^0. Here each intermediate control point $(\bar{c}_{2i+k}^{j+1})_{-m\le k\le m}$ will become the midpoint to the two surrounding control points in the averaging step (proof in Appendix 2, at the end of this dissertation). This implies that on each iteration of the subdivision c_i^0 will be interpolated, but, contrary to the case where *n* is even, we no longer have a control point at c_i^0. After each subdivision step, the midpoint of the two control points surrounding c_i^0 will interpolate it.

Some closing remarks:

(1) It is not necessary for the control points c_{i+k}^{j} to be distributed equidistantly among themselves. As long as the corresponding points c_{i-k}^{j} and c_{i+k}^{j} are equidistant to the interpolatory point c_{i}^{j} (and equal corresponding weights), we maintain interpolation. If they are not distributed equidistantly among themselves, then averaging the corresponding intermediate points $\widehat{c}_{i-k}^{j+1}$ and $\widehat{c}_{i+k}^{j+1}$ will move them over an equal distance with respect to c_{i}^{j} (the proof becomes a bit more complicated as well). We set them equidistantly in our implementation for reasons of symmetry.

(2) It is not necessary for the averaging mask entries to be equal to the normalized binomial coefficients in order to have an interpolatory situation. It is sufficient for them to sum to one and to form a palindrome (i.e. the corresponding split points $\widehat{c}_{i-k}^{j+1}$ and $\widehat{c}_{i+k}^{j+1}$ get the same weighting factor). As mentioned in section 6.1, in [Lane80] it is proven that the limit curve is a B-spline of degree *n+1* if they do equal the normalized binomial coefficients. Since this is a desirable property to have, we set our entries to these values.

6.4 Results

The techniques presented in this chapter have been implemented in C++, using the OpenGL graphic interface [Fosne96]. The basic approximating subdivision scheme is depicted in figure 6-5, where four control vertices define a smooth cubic B-spline. Figure 6-6 shows the same scheme with ghost points added to obtain interpolation in one of the original control vertices. The position of the ghost points further influences the direction of the normal and a tension around the interpolating vertex.

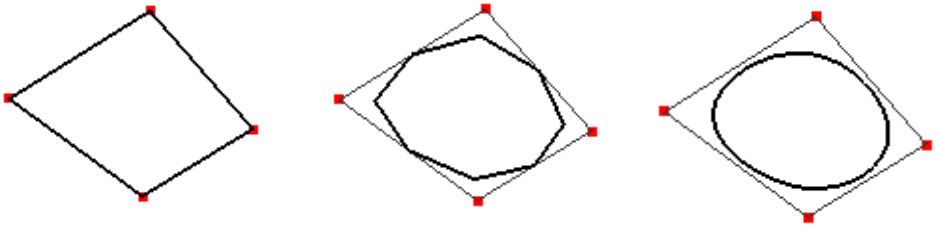


Fig. 6-5. The standard approximating cubic subdivision process: starting from a set of 4 original control vertices (left), a finer mesh (center) is created, in the limit converging to a smooth curve (right).

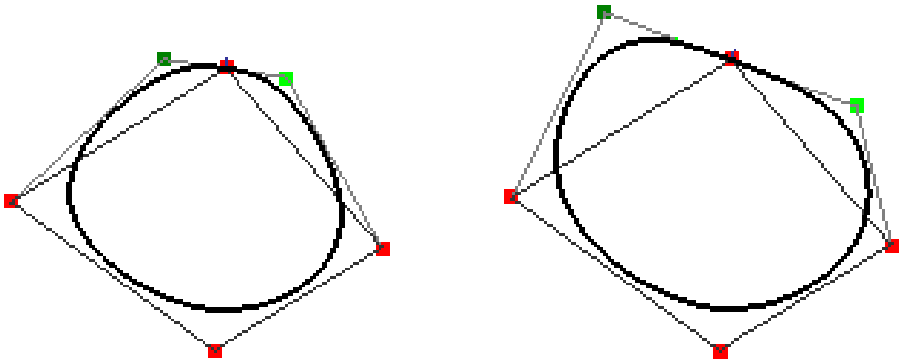


Fig. 6-6. Adding two ghost points around one vertex of the mesh of figure 6-5 makes sure the curve smoothly interpolates that vertex (left). The normal in that vertex and a tension parameter can be controlled by interactively moving the position of the ghost points (right).

The same technique as used at the interior points of the curve, can also be applied to the ends of an open curve (see figure 6-7, left and center), resulting in precise control at these ends. Connecting two open curves creates the possibility of a sharp corner at the joint point (see figure 6-7, right).

Fig. 6-7. The same technique can be applied to control normal and tension at the ends of an open curve. Left: The control vertices and ghost points. Center: The resulting open curve. Right: Connecting the ends of an open curve allows the creation of a closed curve with a sharp corner.

Applying a binomial averaging mask with a larger support generates B-spline curves of a higher degree. Adding the appropriate number of ghost points will also make these curves locally interpolating (see figure 6-8).

In figures 6-9, 6-10 and 6-11 we show some characters that are created using our tool. Local interpolation, normal and tension control and sharp corners were used to create and manipulate these drawings in a straightforward way. The curve editing tools can be used as a base of an animation tool, which is illustrated by the eight frames of a running animation of figure 6-12, inspired by Preston Blair's book about traditional animation [Blair94].

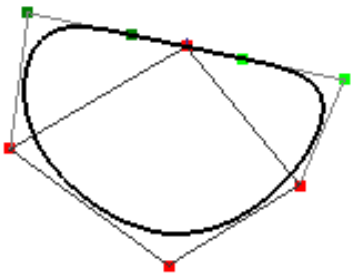


Fig. 6-8. A 4th degree curve with four ghost points added.

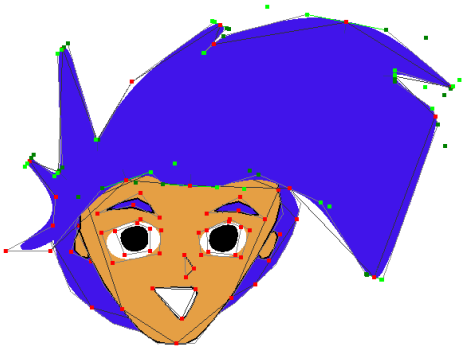


Fig. 6-9. The control vertices determining the contours of a 2D face.

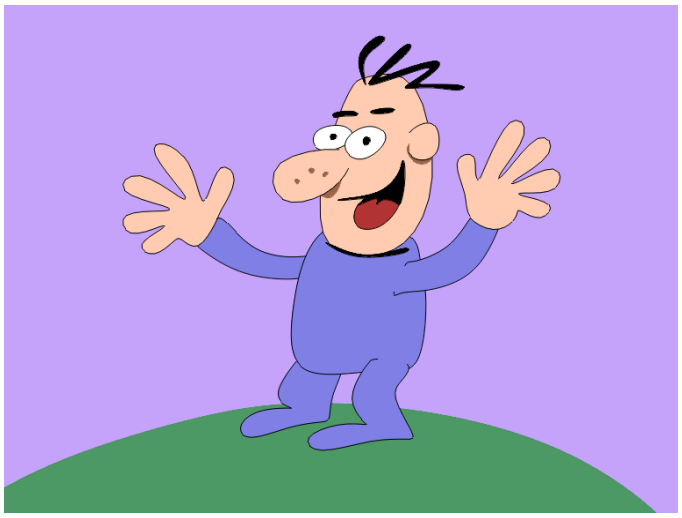


Fig. 6-10. An example of an animation character created via our curve tool, combining local interpolation, normal and tension control.

Fig. 6-11. Another example, where a varying line thickness is also applied.

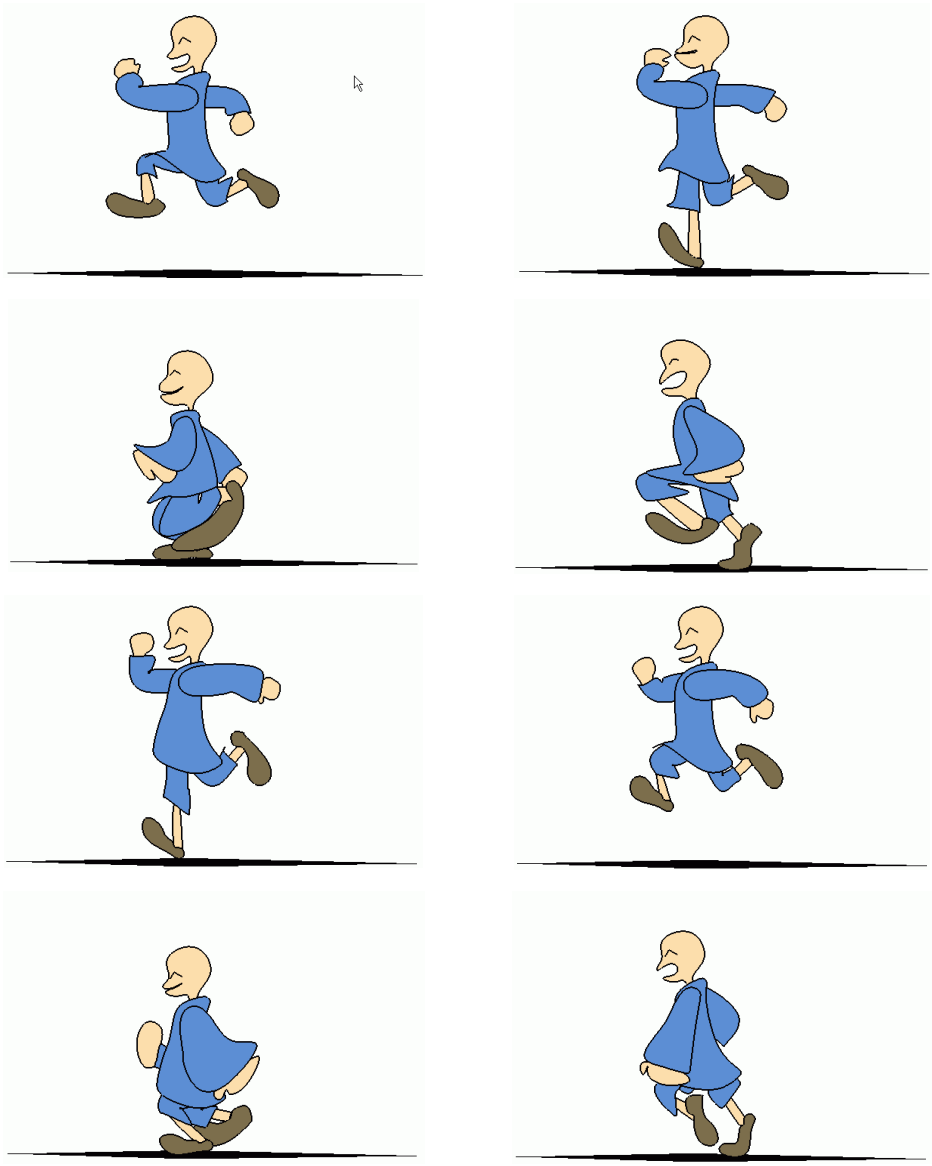


Fig. 6-12. Some frames from an animation sequence created using the locally interpolating curves described in this chapter.

6.5 Discussion

In this chapter, we described a new technique for generating and manipulating interpolatory subdivision curves. The central idea for obtaining local interpolation is based on generating a local invariance with respect to the subdivision process by accurately placing additional control points (ghost points) in the initial control polygon. The proposed method interpolates the initial control points in each iteration of the subdivision process and supports normal manipulation as well as tension control. It should be noted that the idea of generating local invariance could be applied to make other approximating subdivision schemes locally interpolatory as well. Applications of the proposed scheme go beyond curve-editing and manipulation; for example, it can be applied to free-form deformation (see chapter 10) and morphing. Moreover, we have been working on extending the ideas to 3D modeling and manipulation, as described in chapters 5, 6 and 7. Future work can also involve applying similar methods in non-uniform subdivision schemes; e.g. to address bias control around the interpolatory points.

7 Locally interpolating subdivision surfaces

7.1 Introduction

This chapter forms an introduction to the three following ones, explaining specific local interpolation techniques for different subdivision surface schemes.

As explained in chapter 3, the subdivision surface schemes that are most applied in practical applications are Catmull-Clark's [Catmu78] and Loop's schemes [Loop87]. Catmull-Clark's quadrilateral scheme is preferred when the objects to be modeled have 90° and 180° symmetries or many rectangular parts, such as animation characters. Loop's triangular scheme is chosen for free-form surfaces without this kind of symmetry [Zorin00a].

Both schemes are non-interpolating: in general, their limit surface only approximates the original control points, without actually interpolating them. Besides these approximating schemes, interpolating schemes have also been investigated. The most well-known is the extended Butterfly subdivision scheme [Dyn90, Zorin96], creating C^1 continuous surfaces interpolating every single point of the original control mesh. Unfortunately, schemes that rely purely on interpolation have some intrinsic problems: their appearance is harder to control and they produce more bulges and unwanted folds. This makes them much less attractive for use in animation, limiting their main application to situations where a given set of measured or calculated points has to be interpolated by a smooth surface.

In section 7.2 we explain why we want to create local interpolation for approximating schemes, instead of reverting to other solutions such as fully interpolatory schemes. In section 7.3 we explain our requirements in more

detail, and finally section 7.4 gives an overview of the related work in this area.

7.2 Advantages of the most widespread schemes

Both the Catmull-Clark and Loop schemes have been studied extensively, and many features have been developed. Most features explained in chapter 4 are applied most easily on these two schemes. Sharp and semi-sharp edges, for example, have not really been investigated yet for other schemes [Hoppe94, Schwe96, DeRose98].

Another advantage of approximating schemes compared to interpolating schemes is their convex hull property: all newly generated points remain inside the box defined by a small set of spanning control points. Due to the stationary nature of the scheme, this property can be applied recursively, helping, for example, in deciding which part of a mesh to display, further gaining in performance as non-displayed subsets need not be subdivided [Pulli96]. Approximating schemes also have a much narrower support as compared to interpolating schemes, ensuring that editing of the control mesh has a purely local effect.

Specifically interesting for Catmull-Clark surfaces is that they can be rendered directly using off-the-shelf software, such as Alias|Wavefront's Maya [Maya01]. This means that the models created by our tools can be employed directly in this high-end animation software.

7.3 Requirements

The most studied subdivision schemes are both uniform and stationary. In a uniform scheme, the same subdivision rules are used for every point of the mesh. If the same scheme is applied unaltered for every recursive subdivision step, the scheme is said to be stationary. Both characteristics facilitate the writing of applications and studying of mathematical properties. Exceptions to a uniform or stationary approach are normally only employed to cope with special situations; for example, at a boundary or to create creases. Due to the importance of both uniform and stationary rules, we try to avoid affecting these properties with our modifications.

In our modeling application, we want to combine the benefits of the approximating schemes with the possibility of editing locally in an

interpolating way. We noticed that the users of our modeling tools sometimes needed the surface they were designing to interpolate a given point. As the subdivision meshes we generate are displayed by standard rendering software, we require the underlying subdivision scheme to stay intact.

7.4 Related work

An interesting view on interpolating a given set of points comes from Halstead et al. [Halst93]. They describe a way of displacing all control points of an existing approximating mesh and so obtaining a new mesh whose limit surface interpolates the original mesh points. They continue to use the original subdivision rules of Catmull and Clark, but applied to a larger mesh. As the mesh they obtained from displacing the points of the approximating mesh turned out to be too bumpy, they applied a second step, moving the points of the first and second subdivisions in order to optimize some local and global fairness constraints.

As they describe global displacements of an already subdivided mesh, their approach is not very appropriate if one's goals include interactive modifications. Furthermore, their global optimization technique imposes local changes with a possible global effect. This is not a problem for their primary goal – interpolation of a given set of points – but is undesirable for an interactive modeling tool.

More recently, Levin and Biermann [Levin00, Bierm00] describe a clever mathematical framework modifying existing subdivision rules to globally use the normal Catmull-Clark scheme, except at a chosen set of points, where an interpolating variant is used. A non-uniform scheme replaces the original uniform scheme, offering new functionality. Making the scheme non-uniform, however, especially when allowing negative weight factors, dramatically changes the characteristics of the subdivision scheme. The convex hull property is lost, hindering the quick decisions about which parts of an object to display, or about collision detection. Also, schemes for describing the surface in an easily computable way, as in [Stam98, Zorin01b], get much more complicated. Furthermore, extensions such as the sharp and semi-sharp edges introduced by DeRose et al. [DeRose98] are not easily incorporated with the changed subdivision rules. Or, to put it more generally, many studied features, tools [Pulli96] and extensions [Reif00] depend heavily on the original rules set out by Catmull and Clark.

In [Nasri87] some editing and modeling tools especially directed to Doo-Sabin subdivision surfaces are described. Although the Doo-Sabin subdivision scheme is not much used in practice, the techniques suggested in [Nasri87] help a lot in defining which tools can be used to improve the modeling capabilities for subdivision surfaces. Among the editing tools, Nasri describes a geometric construction to interpolate some or all of the vertices of a subdivision mesh. As all points to be interpolated need to be indicated in one step, and a potentially global optimization process is involved, this approach is less fit for the interactive control we have in mind. Furthermore, his approach does not provide tension control. Later [Nasri99, Nasri00, Nasri01b] he extended his set of tools with methods to construct Doo-Sabin surfaces which interpolate some specific type of curves.

Considering all this, we decided to try to achieve interpolation on selected control points without modifying the original scheme. By introducing additional control points – which we call ghost points – and calculating their positions carefully, we not only achieve our goal, but also give the user optional control of the tangent plane and the tension around the interpolating point.

In chapter 8 we describe how such a local interpolation tool can be designed for the Catmull-Clark scheme, while in chapter 9 we describe the intricacies of a similar tool for Loop surfaces. Chapter 10 discusses a free-form deformation tool based on these extensions. Our approaches are also discussed in three of our own publications [Claes00, Claes01a, Claes01b].

8 Locally interpolating Catmull-Clark surfaces

8.1 Introduction

Motivated by the discussion of chapter 7, we now describe a new method to locally interpolate a Catmull-Clark surface. The method not only allows for locally interpolating any number of indicated points, it also provides a control over the tangent plane and a tension parameter in these points.

The rest of this chapter is organized as follows. In section 8.2, we investigate the conditions for getting local interpolation. Section 8.3 discusses the geometric possibilities for arranging the ghost points and an algorithm for constructing an optimal configuration. Finally, section 8.4 shows some images illustrating the techniques discussed in this chapter.

8.2 Geometric conditions for interpolation

We refer to section 3.5 in chapter3 for a more elaborated description of Catmull-Clark's subdivision surface scheme.

Let's assume we have an original control point V_0, which we want to interpolate with a Catmull-Clark subdivision surface. In order to do so, without touching the uniformness of the subdivision scheme, we add a number of ghost points. The purpose of these ghost points is to create around V_0 a carefully constructed submesh that automatically induces interpolation at V_0. Two decisions need to be taken:

- determine where these ghost points need to be inserted in the mesh, and
- determine what the positions of these ghost points should be.

Therefore, it is useful to have a look at the formulas to calculate the limit position V_∞ of V_0. In [Halst93] the following formula is derived and analyzed (see figure 8-1):

$$V_\infty = \frac{n^2 V_1 + \sum (4E_i + F_i)}{n(n+5)} \tag{8-1}$$

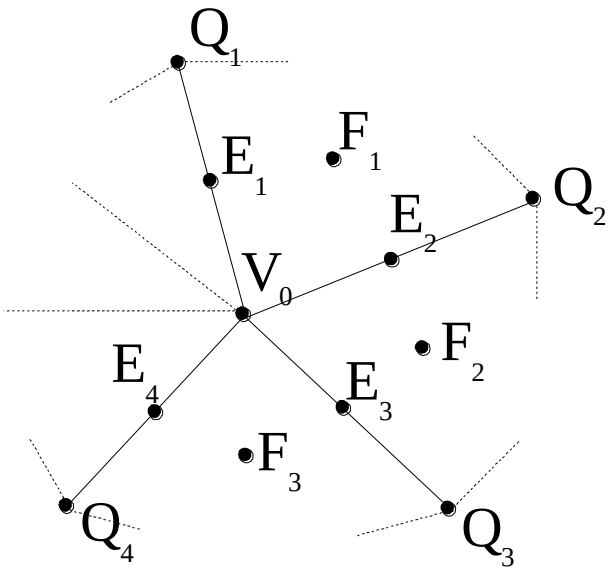


Fig. 8-1. Subdivision around a central vertex V_0, showing surrounding control points (Q_i), edge points (E_i) and face points (F_i).

This formula uses the points from the first subdivision. V_0 is surrounded by n edges, leading to n ghost vertices Q_i, inserted into the original mesh. The n points F_i are called the face points, where each F_i is the mean of all vertices making up one of the polygons surrounding V_0. The n edge points E_i are the mean between the two vertices and the two face points of the polygons that make up the edge. V_1 is the position of V_0 after the first subdivision step of Catmull and Clark's scheme, and is given as:

$$V_1 = \frac{n-2}{n} V_0 + \frac{1}{n^2} \sum Q_i + \frac{1}{n^2} \sum F_i \tag{8-2}$$

In the literature, different weighting factors have been used between these terms. But, as long as these weights stay within certain limits, and are applied in a uniform and stationary way, most features of the original scheme stay valid. Changing these weighting factors mostly affects the curvature, which contributes to a better continuity in all points. In equation 8-2 we used the original weights suggested by Catmull and Clark, but our construction would be exactly the same if we employed other weights.

Substituting equation 8-2 into equation 8-1 results in expressing the limit position in terms of the original points:

$$V_\infty = \frac{n^2\left(\frac{n-2}{n}V_0 + \frac{1}{n^2}\sum Q_i + \frac{1}{n^2}\sum F_i\right) + \sum\left(4E_i + F_i\right)}{n(n+5)}$$

or:

$$V_\infty = \frac{n(n-2)V_0 + \sum Q_i + 4\sum E_i + 2\sum F_i}{n(n+5)} \tag{8-3}$$

In this equation, we can express the E_i in terms of their assembling parts, leading to the following formula:

$$V_\infty = \frac{n(n-2)V_0 + \sum Q_i + 4\sum \frac{1}{4}\left(V_0 + Q_i + F_i + F_{(i+1) \bmod n}\right) + 2\sum F_i}{n(n+5)}$$

or after regrouping:

$$V_\infty = \frac{n(n-1)V_0 + 2\sum Q_i + 4\sum F_i}{n(n+5)} \tag{8-4}$$

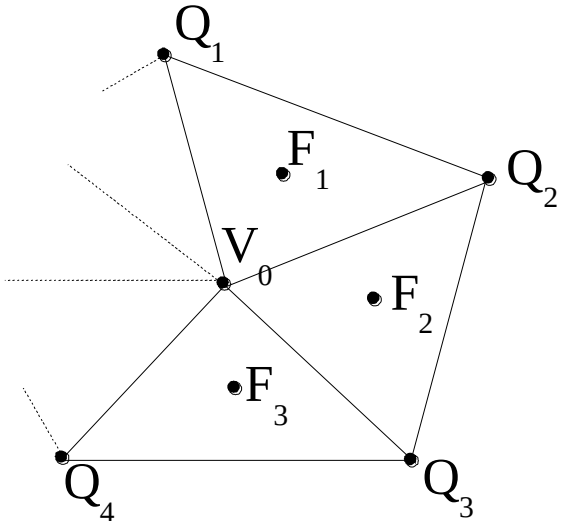


Fig. 8-2. Situation around V_0 when the ghost points are arranged in triangles.

If all F_i have exactly three vertices (see figure 8-2), this equation can be further simplified:

$$V_\infty = \frac{n(n-1)V_0 + 2\sum Q_i + 4\sum \frac{1}{3}\left(V_0 + Q_i + Q_{(i+1) \bmod n}\right)}{n(n+5)}$$

or after regrouping:

$$V_\infty = \frac{n\left(n+\frac{1}{3}\right)V_0 + \frac{14}{3}\sum Q_i}{n(n+5)} \qquad (8\text{-}5)$$

Now, if we want the limit position to be equal to our original input point, all we need to do is to make sure we can get $V_{\infty\infty}$ equal to V_0. In that case, equation 8-5 leads to a simple relationship between the position of the vertex to be interpolated, and the surrounding ghost points Q_i:

$$\frac{1}{n}\sum Q_i = V_0 \qquad (8\text{-}6)$$

Thus, to get our desired interpolation, it is both a necessary and sufficient condition that the ghost points be arranged in triangles and moved so that their average is at the same position as the point we want to interpolate. Moreover, not only will the limit position stay in this same position, but so will all the points generated during the subsequent subdivision steps (this can be verified by substituting the formula for the face points into equation 8-2). This characteristic is useful because it guarantees that also a slightly subdivided mesh will be interpolating at the chosen vertex. For use in a real-time environment, only a limited number of polygons can be processed, which is translated in executing only a few subdivision steps.

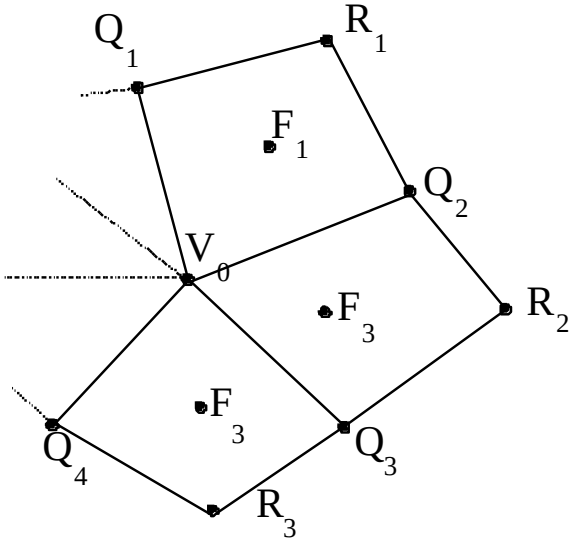


Fig. 8-3. Situation around V_0 when the ghost points are arranged in quadrilaterals.

In equation 8-4 we could also choose all surrounding polygons to have exactly four vertices (see figure 8-3). Such a polygon (around the face point F_i) will consist of the central point V_0, the neighboring ghost vertices Q_i and $Q_{(i+1) \bmod n}$ and a fourth vertex, R_i. Equation 8-4 will then be converted to:

$$V_\infty = \frac{n(n-1)V_0 + 2\sum Q_i + 4\sum \frac{1}{4}\left(V_0 + Q_i + Q_{(i+1) \bmod n} + R_i\right)}{n(n+5)}$$

or after regrouping:

$$V_\infty = \frac{n^2 V_0 + 4\sum Q_i + \sum R_i}{n(n+5)} \tag{8-7}$$

So, with ghost polygons consisting of four vertices, getting the limit position the same as the original point needs the following relation to be true:

$$V_0 = \frac{4\sum Q_i + \sum R_i}{5n}$$

One simple way to make this happen is to make both

$$\frac{1}{n}\sum Q_i = V_0 \text{ and } \frac{1}{n}\sum R_i = V_0 \tag{8-8}$$

Equation 8-8 has the same advantage as equation 8-6: the vertex at issue is interpolated in all iterations of the subdivision scheme (and not only in the theoretical limit). Again, this property can be verified by substituting the formulas for the face points of this specific configuration into the formulas of the subdivision steps (equation 8-2).

It can be noted that except for all triangles or all quadrilaterals, other arrangements of ghost points can also be found that would lead to simple equations like equations 8-6 and 8-8. As Catmull-Clark's scheme prefers polygons with four vertices, we opt to compose multiple quadrilaterals together whenever polygons with more than four vertices are desired.

8.3 Methods for setting up ghost points

In the previous section, it was shown that, when arranging the ghost points in triangles, it suffices to make sure that their mean is equal to V_0. This can be achieved in multiple ways. First, we observe that it would be desirable to have all ghost points in one plane throughout V_0. Indeed, when they are not in the same plane and in order to keep equation 8-6 (or equation 8-8) fulfilled, some points need to be "higher" than others. In that case, the subdivision scheme will generate extra wiggles, which we absolutely want to avoid. We call this plane the ghost plane, as it contains all our ghost points.

8.3.1 Calculating the ghost plane

The orientation of the ghost plane described in the previous section, can be calculated in one of the following ways:

- using Newell's method [Foley91] to calculate the ghost plane parallel to the average plane throughout all the neighboring vertices of V_0;
- employing a similar approach as the normal calculation for Phong shading [Foley91], in particularly calculating the average normal of the planes of the polygons touching V_0;
- using the formulas of [Halst93] for calculating the exact limit normal in V_0.

All of these methods give rise to more or less the same plane equation. Only in some special cases, one method looks more intuitive to the user. Therefore, we let users choose between one of the three suggested methods, but also let them interactively rotate the plane to model a specific feature.

We will be calculating the positions of the ghost points starting from the positions of the edges that neighbor V_0 in the original mesh. As a first step in our algorithm, these points will be projected to the ghost plane. These projections will reside too tightly to the original neighboring vertices, so we scale them down by a user-tunable tension factor. This factor is default set to 0.5. But, depending on the user's modeling needs, it can interactively be put to any desired value, hence providing a flexible tension control. Making the tension factor smaller results in a narrower bump, while making it larger creates a bigger bump around the interpolation point (see figure 8-5).

In the following sections, different approaches for distributing the ghost points are studied.

8.3.2 Even distribution on a circle

Distributing the ghost points evenly on a circle with V_0 as center is an easy way to make sure their average coincides with V_0. A disadvantage is that this will not necessarily be compatible with the original distribution of points around V_0. If there is a big difference between the angles of the different edges, distributing the points evenly on a circle will lead to unwanted twists in the final subdivided surface.

8.3.3 Just moving all points with a vector

Instead of arranging the ghost points on a circle, we could also translate them from their temporary position (obtained from projecting the original

neighboring points to the ghost plane and scaling by the tension factor). From equation 8-6, we can calculate :

$$D := V_0 - \frac{1}{n}\sum Q_i \tag{8-9}$$

being the error between the temporary position and the position needed to get the subdivision surface to interpolate V_0. Just adding D to all the Qi will make sure that their new mean position will be equal to V_0.

8.3.4 Keeping the points near their original location

The method proposed in the previous section is still not the optimal one, as the translation by D moves the points further away from their original edges, which could cause undesired twists.

Therefore we worked out an algorithm that tries to satisfy both the condition from equation 8-6 and to keep the points as close as possible to their original edges. The following pseudo-code describes the algorithm:

Input: A vertex V_0 surrounded by n vertices P_i, all on the original subdivision mesh.

Output: Adding a set of ghost vertices to the mesh such that the surface obtained by the Catmull-Clark subdivision of the resulting mesh interpolates V_0.

Algorithm:

1. Calculate T, the tangent plane in V_0 (see section 8.3.1). This plane is optionally rotated by the user.
2. Project all P_i to plane T, obtaining points P'_i. Then scale these points with the tension factor towards V_0, obtaining the temporary position for our ghost points, the Q_i. So $Q_i := V_0 + \text{tensionFactor} * (P'_i - V_0)$.
3. Calculate M being the mean of the distances between the Q_i and V_0.
4. Calculate D := 1/n * sum (Q_i). This is the mean of the ghost points, that ultimately should coincide with V_0.

5. Translate all Q_i with the vector V_0 - D, so $Q_i := Q_i + (V_0 - D)$ (this will put the new mean of the Q_i at V_0).

6. Project all Q_i back on the original line ($P'_i - V_0$) where they were at step 2. This will move the mean of the Q_i a little bit away from V_0.

7. Calculate M', the new mean of the distances between the Q_i and V_0. Because step 6 also pushes all the Q_i closer to V_0, in order to prevent the ghost points collapsing towards V_0, they should be moved again to their original distance $Q_i := V_0 + M / M' * (Q_i - V_0)$.

8. Repeat steps 4 to 7 until D and V_0 get close enough together. As in some configurations this will lead to an infinite loop, we will stop the loop after 100 iterations.

9. The ghost points obtained in this way, will be connected into triangles between the original surrounding points (the P_i) and the point V_0 that should be interpolated.

8.3.5 Using the first subdivision points as ghost points

Instead of adding completely new points, we could also run a similar algorithm with the points obtained from the first subdivision. But this has as its main drawback that when many neighboring control points are indicated as interpolating, the algorithm gets into trouble positioning the subdivision points in between these control points. Furthermore, this would lead to non-local effects when indicating points as interpolating, which we want to avoid.

8.4 Results

In this section we show some pictures illustrating our editing tool. Figure 8-4 shows a torus where we constructed a submesh so the surface interpolates a given vertex. In figure 8-5 we show the effect of interactively manipulating the tension parameter. Figure 8-6 makes clear that our method can be combined with other methods, such as the sharp edges introduced by [DeRose98].

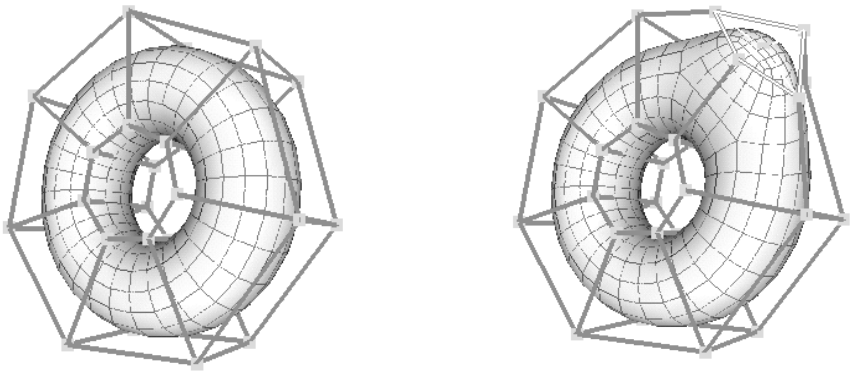


Fig. 8-4. A torus: at the left with the original mesh, at the right with the modified mesh, making one vertex interpolating.

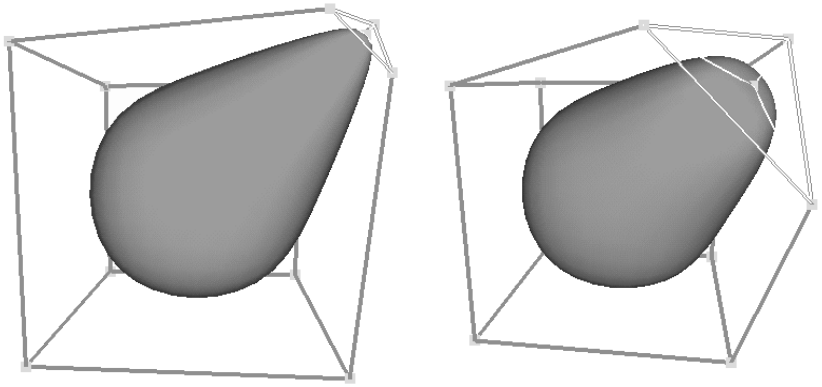


Fig. 8-5. Setting the tension parameter to a small value (at the left) or a large one (at the right, from a slightly different viewpoint) influences the form of the bump.

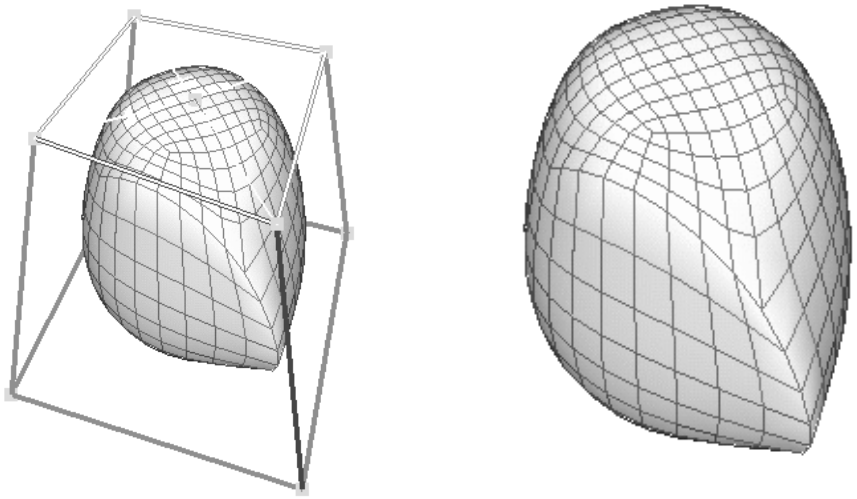


Fig. 8-6. Combining our method with sharp edges. The original input mesh is a pyramid, where we marked the top as interpolating. Two edges were marked as being sharp.

9 Locally interpolating Loop surfaces

9.1 Introduction

In this chapter, we extend the existing editing tools and manipulation possibilities of the approximating Loop scheme for triangular meshes [Loop87]. When interactively creating a new surface, local interpolation is often desired, preferably without having to cope with the difficulties of the fully interpolating schemes. It turns out that we can achieve this goal by extending the original control mesh with a particular geometric construction. The points introduced by this construction will be called ghost points. In the previous chapter, we described a similar approach for the interior of Catmull-Clark surfaces. Here we concentrate on which constructions and algorithms are needed for the Loop scheme, and we investigate how this approach can be extended to the borders of the surface.

The ghost points used for local interpolation also give rise to even more attractive editing tools. By rotating the plane containing these ghost points, the user can easily give any direction to the tangent plane (or surface normal) in the interpolated point. Furthermore, the distance between the interpolating point and the ghost points can be scaled, providing an intuitive tension control.

All this can be achieved while keeping the underlying Loop subdivision scheme intact, enabling the resulting control meshes to be incorporated directly by existing rendering and modeling software.

The rest of this chapter is organized as follows. In section 9.2 we analyze the geometric constructions that achieve local interpolation in points indicated by a user. These points can be situated either at the interior of the surface or at its border. Section 9.3 discuses some implementation issues and a practical

algorithm, whereas section 9.4 illustrates the techniques described in this chapter. Finally, section 9.5 concludes this chapter discussing the benefits of these techniques and comparing them with alternative approaches.

9.2 Geometric discussion

9.2.1 Loop's subdivision scheme

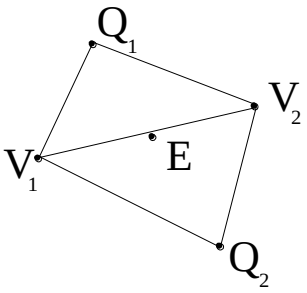


Fig. 9-1. Situation around an interior edge.

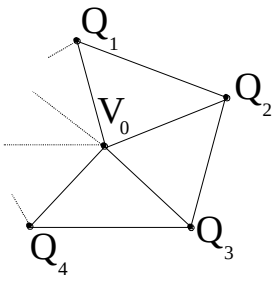


Fig. 9-2. Situation around an interior vertex.

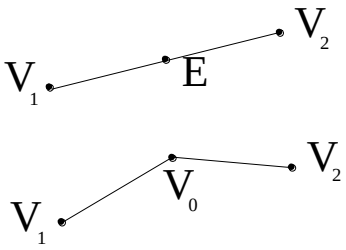


Fig. 9-3. Situation around a border edge and a border vertex.

We refer to section 3.8 of this dissertation for a more general introduction to Loop's scheme. Here, we only describe the essential formulas needed to explain our approach to local interpolation.

Loop's subdivision scheme consists of two alternating stages executed in a recursive way. During the first stage, new control vertices are introduced in the middle of each edge and both old and new vertices are all reconnected to form four new smaller triangles. In the second smoothing stage, all vertices are averaged with their surrounding vertices. Executing this process recursively to a usually coarse initial polygon mesh obtains a fine subdivided mesh of small triangles, in the limit forming a smooth surface. The new points at the middle of an edge are called edge points, while the points of the existing mesh are called vertex points. The following formula controls the averaging of a new edge point E (on the edge between V_1 and V_2 and with Q_1 and Q_2 as immediate neighbors) in the interior of the mesh (see figure 9-1):

$$E = \frac{3}{8}(V_1 + V_2) + \frac{1}{8}(Q_1 + Q_2) \qquad (9\text{-}1)$$

And the formula for averaging an interior point V_0 (surrounded by k vertices $Q_1 \ldots Q_k$) is the following (see figure 9-2):

$$V_0' = \sum \beta Q_i + (1 - k\beta)V_0$$

$$\text{with } \beta = \frac{1}{k}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{k}\right)^2\right) \qquad (9\text{-}2)$$

For border-vertices there are only "surrounding" vertices on one side. If one would employ the same rules as at the interior, the averaging process pulls everything too far to the interior. Therefore, the formulas for new edge and vertex points at the border do not take the interior points into account. In practical situations, the formulas are often simplified to the ones used for subdivision curves (see figure 9-3):

$$V_0' = \frac{3}{4}V_0 + \frac{1}{8}(V_1 + V_2) \text{ and } E = \frac{1}{2}(V_1 + V_2) \qquad (9\text{-}3)$$

9.2.2 Interpolation at the border

As the formulas at the border of the Loop surfaces are the same as for subdivision curves, it looks appealing to investigate whether we can incorporate the technique described in chapter 6 dealing with locally interpolating subdivision curves. In chapter 6, local interpolation is accomplished by extending the control polygon of the curve with ghost points on a line throughout the point to be interpolated. The orientation of this line controls the tangent (thus the normal) at the interpolated point, while the distance between the ghost points affects the tension. Hence, besides local interpolation, the described technique also provides normal and tension control without having to revert to a non-uniform or a non-static scheme. See figure 9-4 for an example of a curve interpolating one of its points. Extra details and a mathematical discussion about this type of local interpolation can be found in chapter 6.

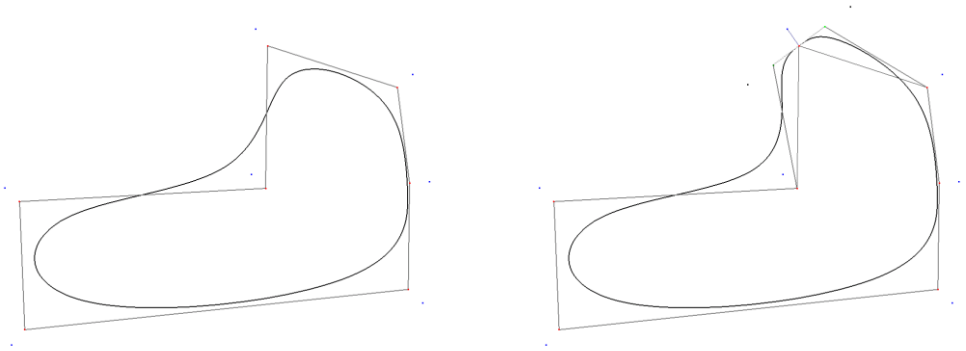


Fig. 9-4. A control polygon and the resulting curve without interpolation (left) and with ghost points added to obtain interpolation in one of the points (right).

When extending this approach for curves to get interpolation at the border of a subdivision surface, it is not sufficient just to add some ghost points. As Loop's control mesh may only consist of triangles, these ghost points have to be connected to the interior points as well. The insertion of the two ghost points will change the two neighboring polygons from triangles to quadrilaterals (or into pentagons, when two neighboring border vertices should be interpolating). A simple solution would be to divide each quadrilateral into two triangles using one of the diagonals of the quadrilateral. However, in general this creates rather narrow triangles, which can cause undesired wrinkles in the resulting subdivision surface. That's why we opt to add a new point in the center of each quadrilateral and form triangles by connecting each vertex to the center. This creates an extra degree of freedom, which we can use to optimize the fairness of the surface.

The geometric construction needed for the local interpolation at the border, can also be used to change the tangent (and thus the normal) in the interpolated point. To achieve this, it suffices to rotate the line that the ghost points are put on. Furthermore, by changing the distance between the ghost points and the interpolated vertex, a handy tension parameter can easily manipulate the form of the curve forming the border of the subdivision surface.

9.2.3 Interpolation at interior vertices

While at the border the formulas only take neighboring border vertices into account, getting interpolation at interior vertices is more complicated. At the interior, there is a ring of surrounding vertices that jointly influence the subsequent positions of a point in the subdivision mesh.

In order to derive conditions to get interpolation at a point V_0 surrounded by k ghost vertices Q_i we start by rewriting equation 9-2, as:

$$V_0' = k\beta \frac{1}{k} \sum Q_i + (1 - k\beta) V_0$$

Here we see that if the mean of the surrounding ghost points Q_i is equal to V_0, then the next iteration of V_0 will stay on the same place. This condition is formulated as:

$$\frac{1}{k} \sum Q_i = V_0 \tag{9-4}$$

Continuing our derivation, we use equation 9-1, which defines how the neighboring edge points will be moved. In the recursive subdivision process these edge points will namely be the surrounding vertices for the next subdivision iteration. Equation 9-2 will be able to calculate V_0'' - the position of V_0 in the second iteration - expressed in function of the position V_0' at the first iteration:

$$V_0'' = \sum \beta E_i + (1 - k\beta) V_0'$$

If we substitute equation 9-1, adapted to the specific edges, we get:

$$V_0'' = \sum \beta \left(\frac{3}{8} (V_0' + Q_i) + \frac{1}{8} \left(Q_{(i-1) \bmod k} + Q_{(i+1) \bmod k} \right) \right) + (1 - k\beta) V_0'$$

or:

$$V_0'' = \sum \beta \left(\frac{3}{8} Q_i \right) + \sum \beta \left(\frac{1}{8} Q_{(i-1) \bmod k} \right) + \sum \beta \left(\frac{1}{8} Q_{(i+1) \bmod k} \right) + (1 - \frac{5}{8} k\beta) V_0'$$

or:

$$V_0'' = \frac{5}{8} k\beta \frac{1}{k} \sum Q_i + (1 - \frac{5}{8} k\beta) V_0' \tag{9-5}$$

Equation 9-5 shows that if equation 9-4 is true for the first iteration, it will hold again for the next iteration. Therefore, applying this line of thought by induction, equation 9-4 is a sufficient condition to keep every subsequent mesh interpolating in V_0 during every single subdivision step and thus also in the final limit.

We will be adding a ring of ghost points around V_0 to satisfy equation 9-4. In theory these ghost points don't need to lie in one plane. But then equation 9-4 will oblige some points to be above the tangent plane, and some under the tangent plane. This will introduce some folding that we want to avoid.

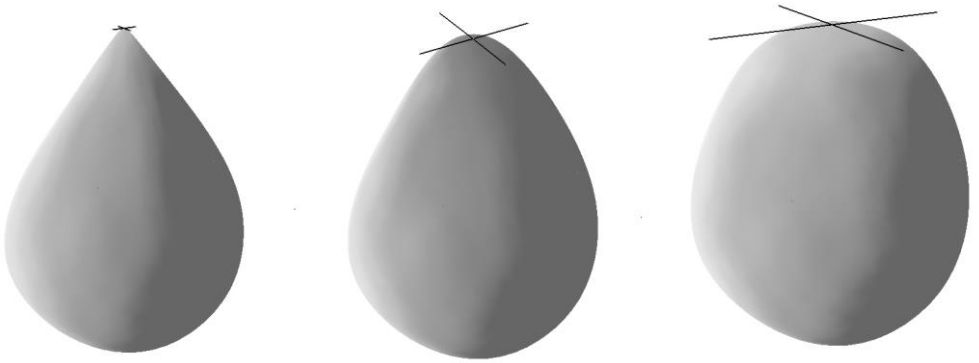


Fig. 9-5. Showing the difference between a very small (left), a normal (center) and a large tension (right).

Just as we can manipulate the tangent line at the selected border vertices, we can also rotate the tangent plane for the ghost points of the interior points. This can be used as a modeling tool to give the surface any desired tangent plane in an interpolating interior vertex. Furthermore, equation 9-4 will stay invariable when the ghost points Q_i are scaled with respect to V_0. So scaling these ghost points gives the possibility of manipulating a tension parameter. Figure 9-5 shows an example of changing the tension parameter.

9.3 Implementation

The geometric construction of the previous section, led to a practical algorithm containing the following steps:

1. We start with a polygonal mesh and mark some of its points as interpolating. The polygonal mesh can have an arbitrary topology and optionally have a border. An example of such an original mesh is shown in figure 9-6.

2. Ghost points are added into the topology of the mesh. For every interpolating vertex there will be added a ghost point on every surrounding edge. The resulting mesh will contain polygons of more than three vertices. Depending on the number of points that have to be made interpolating, these polygons can have four, five or six vertices.

3. Just triangulating the polygons with too many vertices will usually result in some of the polygons being quite narrow, causing unwanted folds. Therefore, a center point of the polygon is calculated, and all of its vertices are connected to this center point.

4. A tangent plane to the surface is calculated, which is optionally rotated by the user. All ghost points introduced in step 2 are projected onto this plane.

5. Then these ghost points are moved such that their mean is equal to the point we want to interpolate, satisfying equation 4. Optionally, these projected ghost points are scaled by a user-supplied tension parameter.

6. Finally, the center points introduced in step 3 are moved to get the surface as fair as possible. The final construction together with the resulting surface is shown in figure 9-7.

To get equation 9-4 satisfied, we tried several approaches. Our first implementation distributed the points evenly over a circle, which resulted in a very fair surface near our interpolating point, but in many cases got some torsion effects. A next approach was to just calculate the current mean of the ghost points and move all of them with the vector between this mean and the desired mean V_0. Usually this gives acceptable results, but sometimes it resulted in a torsion effect. Therefore, we extended this last approach with an iterative loop, moving the points to get their desired mean and reprojecting them towards their original edge.

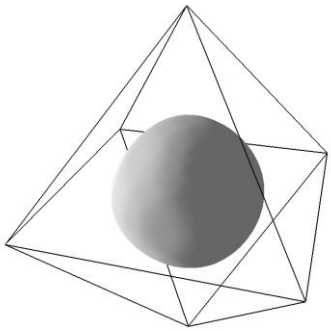


Fig. 9-6. A standard mesh. The resulting Loop surface does not interpolate its control points.

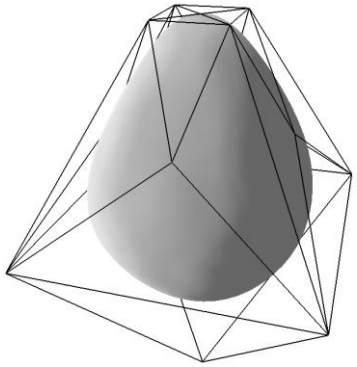


Fig. 9-7. The mesh is extended with a geometric construction to make the limit surface interpolate the topmost point.

9.4 Results

Figure 9-8 shows an object modeled with Loop subdivision surfaces, making use of the tools introduced in this chapter. In figure 9-9, a close-up of the beak shows the difference between a standard modeling technique and the application of our extensions.

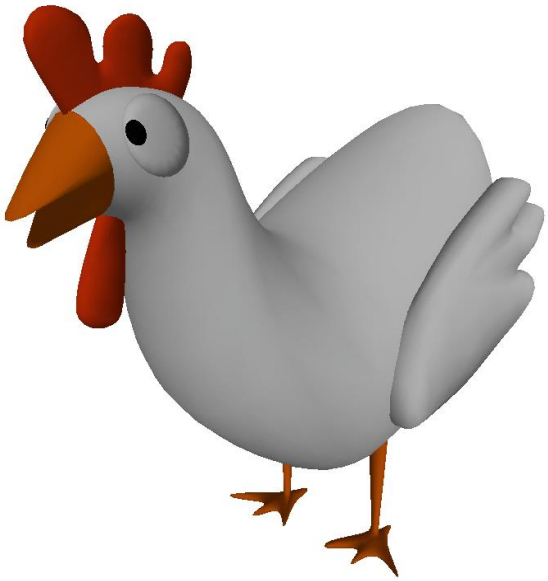


Fig. 9-8. A chicken modeled with Loop surfaces, making use of interpolatory points, tension and normal control.

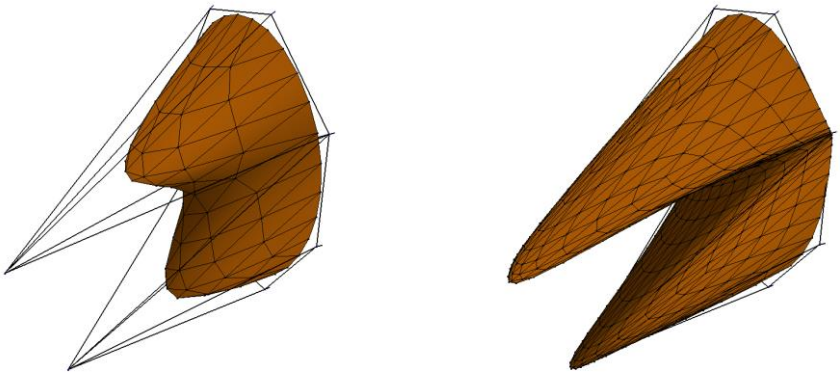


Fig. 9-9. Zoomed in on the beak of the chicken. At the left no interpolation is used, while in the image at the right some points are made interpolating and adequate tension parameters are set.

9.5 Discussion

In this chapter, we described a modeling tool that has the ability to interpolate selected points for surfaces constructed by the traditionally approximating Loop scheme. The main advantage of our solution compared with other approaches is that we keep the underlying uniform and stationary scheme completely intact, only adding carefully located ghost points to the original mesh. Hence, we keep all the existing features of the Loop scheme, only adding new editing tools. This way existing rendering engines, for example, can be employed directly without any change. Our new tools can also extend an existing modeling toolbox, to be used either separately or in combination.

Furthermore, we provide the user with the possibility to edit the interpolatory normal direction and control the surface tension at the interpolating point.

It turned out that the described framework is not only useful for modeling 3D objects, but is also very useful for creating an interactive deformation tool, which we elaborate in the next chapter.

10 An application: a free-form deformation tool

10.1 Introduction

This chapter describes a free-form deformation scheme dealing with 2D animated objects. As animations are mostly shown as moving 2D images, it often suffices to only decide about the movements in two dimensions to create convincing animations. This does not work out properly when physically correct movements are needed, but is very suitable to informally deliver creative ideas to a viewer.

The following requirements showed up for free-form deformations suited to accomplish this goal:

- There should be fluid movement, not only at the border, but also at the interior of the animated object, and the texture parameterization of the surface should be deformed in a smooth, natural-looking way.
- Both global control, needing limited user interaction, and fine local control near specified joints should be integrated into one consistent interface paradigm.
- Specific discontinuities should be allowed, such as a hole inside the animated object or limbs sticking out of it. For example, although a character's two feet can be situated close together, usually they should be animated and deformed independently and they can even overlap.

In order to cope with all this, we closely examined existing free-form deformation schemes, but, unfortunately, none of them combined all desired requirements. Therefore, we opted to investigate the application of the extensions of the subdivision scheme described in chapter 9.

The rest of this chapter is organized as follows. Section 10.2 describes free-form deformations, gives some pointers to related work and explains how

they will be used in our application. Section 10.3 deals with subdivision surfaces with local interpolation and explains how they can be used for free-form deformations. In section 10.4, the details of our implementation are elucidated, while the next section formulates a conclusion and explains our ongoing future work.

10.2 Free-form deformation (FFD) in 2D

10.2.1 Existing FFD schemes

Sederberg [Seder86] and Barr [Barr84] were among the first to point out possibilities, advantages and implementation schemes of deformations and, more particularly, of free-form deformations (FFDs). Many followed this trail, improving and extending their usability for different tasks and requirements.

Sederberg put a 3D B-spline lattice around a selected object, then modified the positions of the vertices of the control lattice, and finally applied that deformation to the object. Coquillart combined Sederberg's lattices to allow more complicated deformations [Coqui90]. In a follow-up paper, she also decoupled the lattice from the object to allow animating the lattice separately or to move the object through a deformed space [Coqui91].

Different representations of the deformation tool were investigated:
- a surrounding control lattice [Seder86],
- combining multiple lattices [Coqui90],
- a lattice build up from subdivision volumes [MacCr96],
- some controlling curves or based on an axis [Barr84],
- control surfaces [Feng96] or
- a scattered set of points [Mocco97].

The type of tool used for the deformation strongly determines what kinds of deformations are feasible and how easily the user can control them. Each tool can be adequate in its own right, depending on the needs in the specific application.

Most work in FFDs concentrates on 3D deformations, considering 2D deformations as a simplification: just leave out one dimension. This ignores that when you restrict yourself to 2D deformations, additional goals can be achieved, as explained in the introduction (see section 10.1). One of the people specifically tackling 2D deformations was Sederberg in his Siggraph'93 paper [Seder93], where he describes a method to interpolate

between two deformed 2D objects. Each object is represented by a polygon. The paper restricts itself to the behavior of the border, giving no clue about how the interior of the polygons should be deformed.

10.2.2 Deforming parameterization and local control

In [Inter97] arguments are given to show the significance of texture mapping for conveying 3D shape, even for non-deformable objects. Moreover, when we only dispose of a flat 2D deformable object that pretends to represent a 3D shape, precise control of the texture mapping becomes extremely important in order to deform in a convincing way.

Zonenschein et al. [Zonen98] studied the texturing of deformable implicit surfaces, indicating texture artifacts ("ghosting") when the objects are deformed. In their implementation, they needed to blend colors and transformations to get a plausible result. We opt for a more exact control of the texture, so we try to avoid blending.

The FFD schemes mentioned in section 10.2.1 do not specifically take care of the parameterization (texturing) of the surface; they only concentrate on the general shape. Furthermore, with most of these FFD schemes, local control is not so easy. Local control implicates a denser mesh, but usually this is only possible if the complete mesh is subdivided, which obliges the user to control a huge set of points. Only [Mocco97] and [MacCr96] allow local control, so their approaches needed to be studied closer in view of our application.

We considered the approach of [Mocco97], who organizes scattered control points into a Delaunay triangulation. Their mesh is not explicitly visible to the user, which has the advantage that the user doesn't need to spend time to create the connections, but has the disadvantage that the user cannot make different connections when needed, for example to mimic certain physical connections. As the main goal in [Mocco97] is deforming hands represented by many control points that are positioned relatively close together, a Delaunay triangulation forms the most adequate connectivity. When attempting to apply this approach for 2D animation purposes, however, with only a limited number of control points, the possibility to create own connections, including explicit discontinuities, turned out to be a necessity. Nevertheless, [Mocco97]'s idea to start out with a Delaunay triangulation is also useful in our approach, where we extend the idea with the possibility of re-editing the generated mesh. Unfortunately, their scheme for calculating the coordinates in the mesh is no longer applicable, as it strongly depends on

the Voronoi diagrams defined by these triangulations; furthermore, the convex hull property prohibits having the type of discontinuities we need.

MacCracken and Joy's solution to FFDs [MacCr96] is based on subdivision volumes created by 3D lattices of arbitrary topology. We liked their idea to use subdivision, as it is the only FFD approach facilitating arbitrary topologies. Nevertheless, although in theory there is considerable freedom in manipulating deformations, their setup is rather hard to establish and control by a user. Also, their way of subdividing space makes calculating the coordinates of a point referring to the deforming mesh less straightforward. In our approach, instead of their 3D subdivision volumes, we apply subdivision surfaces, augmented with adequate control tools.

The system we propose has specific advantages and features when compared to the previously described techniques. None of the techniques combines all of these features into one concise interface. The main differences are:

- We allow both general global local control in areas of less interest and simultaneously precise local control where needed. This combined type of control is also possible in [MacCr96], but their 3D lattices are hard to handle and to position precisely, and furthermore they don't allow for local interpolations. [Mocco97] also allows some combination of local and global control, but does not provide discontinuities.
- None of the FFD techniques described in section 10.2.1 explicitly cares what happens to the object outside of the border. Objects are just embedded in a larger space. Everything that could happen to the FFD transformation outside of the border is simply ignored. In our approach, however, we want to allow for discontinuities. If the transformation extends too far outside the border, the effect of an FFD applied to one part will result in an overlap with neighboring parts of the object. This overlapping complicates making sure the animation of one part does not influence a neighboring part, as, for example, in the case of two legs. Therefore, we provide very precise border control.
- Most FFD approaches can easily deform an object as a whole, but have problems handling the interior just as easy. The interior is deformed as to minimize distortions, but this cannot be guided as fluently as desired by an animator. We solve this by allowing for interpolating points, not only at the border but also at the interior.

10.3 Locally interpolating subdivision surfaces

10.3.1 Recursive subdivision schemes

Recursive subdivision schemes have been used to define curves (in 2D or in 3D), surfaces (usually in 3D) and volumes (in 3D) [MacCr96]. Such a scheme starts with a set of control points, and in each subsequent subdivision step, in-between points are introduced and simultaneously averaged by their neighbors. Depending on how adequate the averaging scheme is, this process will eventually converge to a smooth curve, surface or volume. It will result in a curve if the points are connected in one linked list (like a polygon), in a surface if the points are connected like a polyhedron and in a volume with points connected in a lattice. Recursive subdivision schemes are explained in more detail in chapters 3 and 4 of this dissertation.

10.3.2 Using subdivision surfaces for FFD

We based our FFD scheme on subdivision surfaces, as such a surface can both represent the border and the interior of a 2D object. A subdivision scheme is said to be uniform if the same scheme is applied unchanged to every point. The scheme is stationary if the same rules are used for all subsequent subdivisions. As interesting mathematical and practical properties depend on the scheme being both stationary and uniform, people only avoid them if they want to achieve exceptional goals. One of these goals can be coping with boundary conditions, because the ordinary rules for the interior do not work at the border. As we want to describe a 2D surface that does not cover the entire plane, we necessarily need to have surfaces with a border. Luckily, the standard rules for borders keep the properties of the otherwise fully uniform Loop scheme mostly intact [Hoppe94, Schwe96].

Chapter 7 discussed the need for local interpolation of subdivision surfaces, and chapters 8 and 9 presented actual methods to achieve local interpolation for the most widespread schemes, Catmull-Clark's and Loop's. For reasons of easy and smooth editing control and the requirements of our free-form deformation application, we chose to work with an approximating scheme based on triangles: Loop's subdivision surfaces. We preferred Loop's triangular scheme, because triangles are easier to parameterize unambiguously. Furthermore, in a 2D environment, triangles can more freely be adapted to specific configurations, while quadrilaterals as in Catmull-Clark's scheme are more suited for rectangular symmetries.

10.3.3 Local interpolation, normal and tension control

As the surface is employed in 2D, it necessarily has to deal with a border. From the formulas of chapter 9, it is clear that the border of the Loop surfaces is just a subdivision curve, with no interior point taken into account. This makes the technique described in chapter 6, dealing with locally interpolating subdivision curves, very valuable for our FFD implementation. Local interpolation is accomplished by extending the control polygon of the curve with ghost points on a line throughout the point to be interpolated. The orientation of this line controls the tangent (and thus the normal) at the interpolated point, while the distance between the ghost points affects the tension. Hence, besides local interpolation, the described techniques also provide normal and tension control, without having to revert to a non-uniform scheme. In figure 10-1, we show an example of the behavior of the mesh. For a more detailed description of the interpolation at the border, we refer to chapter 6.

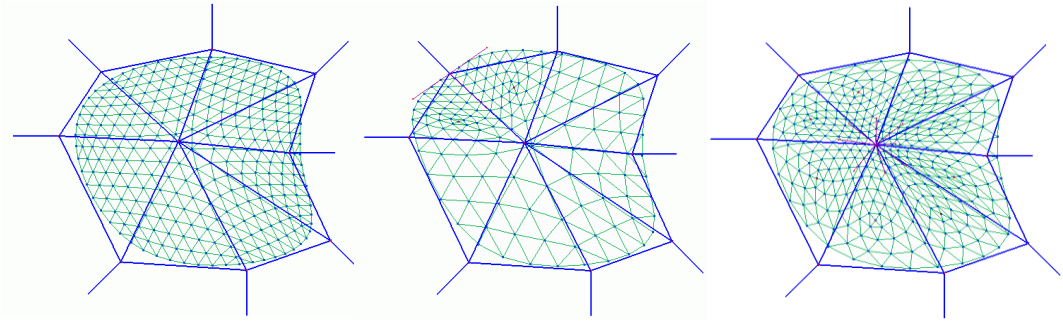


Fig. 10-1. A subdivision surface mesh in 2D, at the left without interpolation, at the center interpolating a border vertex, and at the right interpolating an interior point (note that for clarity also the normals at the border edges are shown).

The technique for local interpolation can furthermore be extended to the interior of the Loop surfaces, as described in chapter 9. Figure 10-1 is an example of a mesh with the two sorts of interpolation.

10.4 Implementation

In our basic approach, we start out with a 2D object to be deformed. The object is represented by a 2D image and can have an arbitrary topology, like having holes or limbs sticking out. On this image, the user draws a net of

control points. The control points are put both at the interior and near the border. At most places, just an approximating control mesh suffices, but wherever more control is needed, the user can choose to insert an interpolating point, as demonstrated in figure 10-2. An example of a mesh for the head of an animation character is shown in figure 10-3. At the right of figure 10-3, this mesh is subdivided once. The user can also create some kind of skeleton using the mesh, but this is not a necessity.

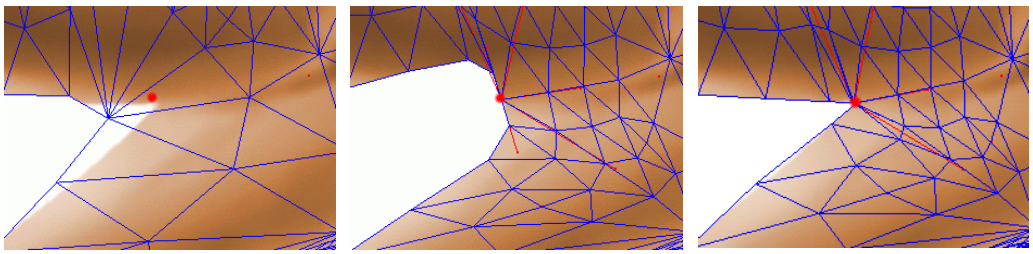


Fig. 10-2. Using the local interpolation and normal control of the border to fit the surface to the object. Left: The situation without local interpolation. Center: Interpolation, but with a bad tension. Right: Fitting the tension at the border.

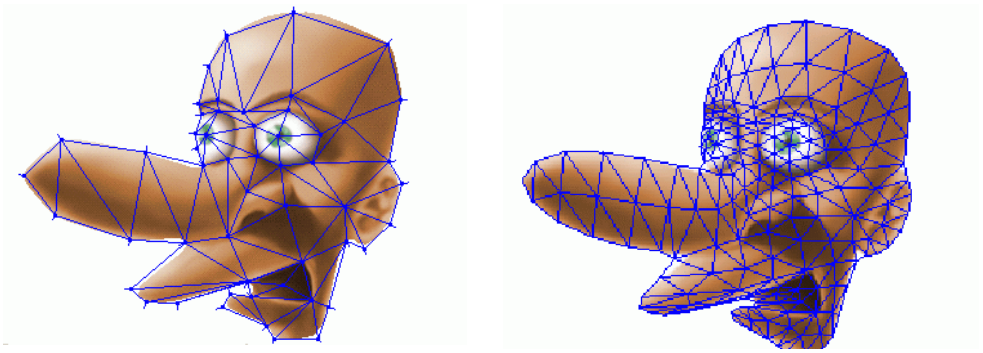


Fig. 10-3. The original and the subdivided control mesh for an animation character.

Furthermore, special care is taken on the border near places where separate parts that stick out come close together, as, for example, between the upper lip and the nose. In more traditional FFD approaches, at those places the control mesh would be interconnected, forming something similar to a convex hull. We will draw the discontinuities explicitly, by continuing the border between them.

When the user finishes setting up the initial mesh, this mesh is frozen to the object, analogous to other FFD approaches. Internally in our program, the mesh is converted into the triangles belonging to some level of subdivision. At the corner of each triangle, texture coordinates will be generated, mapping the undeformed initial 2D image to this geometry.

In the next stage – also a typical step in FFD – the user can start moving points of the control mesh or even animate them. In the program, the mesh will be subdivided again, the texture coordinates belonging to the initial position will be applied and everything will be redrawn, resulting in a deformed object. Typical for our approach, is that apart from moving control points, the user can also manipulate the tension and the normal, giving rise to appealing effects that are hard to establish with other methods. Figure 10-4 shows a typical example.

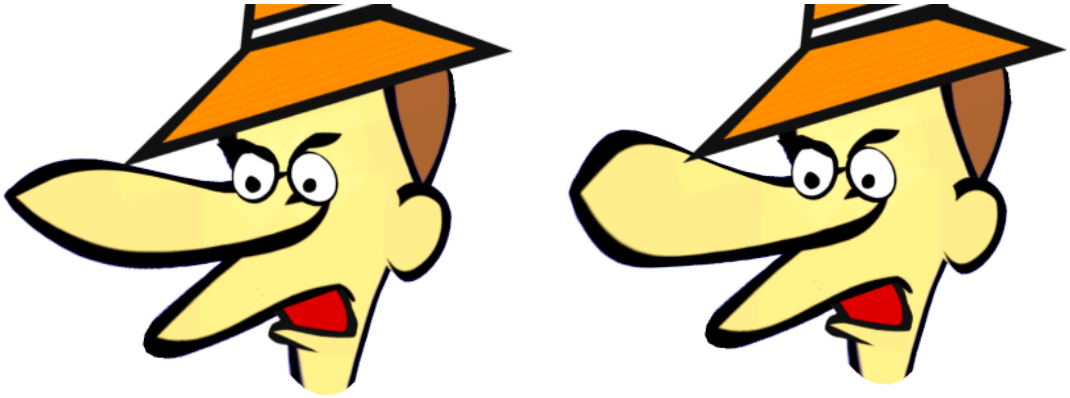


Fig 10-4. An example of changing only the tension in the border point at the tip of the nose.

An additional advantage of working with the approximating Loop subdivision scheme is that we can set up a tree of convex hulls. In that tree, each subsequent subdivision level is contained into a convex hull defined by the control points of that level. This enables a quick search for where a point resides in the generated mesh.

Figure 10-5 refers to a very expressive animation that was created with a small amount of user input. The animation gives a lot of 3D feeling, while all manipulations are kept strictly 2D.

Fig. 10-5. Some frames from an animation created by our system. More control points and interpolation are used around the eyes, to provide better local control. Between the upper lip and the nose there is an explicit discontinuity to prevent lip movements from having undesired effects on the nose.

10.5 Discussion

In this chapter we described a method for deforming 2D images, based on locally interpolating subdivision surfaces with normal and tension control. Our method enables a very smooth movement, explicit discontinuities and both global and local control. None of the other FFD approaches, described in section 10.2.1 is able to combine all these features in one uniform concept.

In our ongoing future work, we are investigating ways to incorporate higher-level editing of the mesh, such as multi-resolution editing. Furthermore, we want to have a closer look at combining our methods with physically-based modeling techniques and constraint-based systems.

In addition, we are thinking about extending our approach to 3D, keeping in mind the requirements that are also important for 2D deformations. Another track is instead of deforming objects, deforming the space through which the object moves, similar to the ideas presented in [Coqui91].

11 Directions for future research

11.1 Further extending subdivision surface editing

Skaria et al. [Skaria01] described an interesting implementation of a tool to create faces for cartoon characters. With a combination of easy-to-use 2D interfaces, they create subdivision surfaces with a minimal number of control vertices, especially taking care to strongly restrict the number of extraordinary points. In a similar way, powerful editors should be built for many more applications, helping the designer of a surface to obtain a good-looking model with low polygon count. A very interesting feature of subdivision surface modeling is that the designer can work with an easy-to-understand-and-manipulate polygonal model. Nevertheless, specific tools for high-level control are still a long way from full maturity.

We are also convinced that a surface modeler that directly operates on the hexagonal meshes for the new subdivision surface scheme described in chapter 5, could have the same control and flexibility as existing modelers for Catmull-Clark and Loop surfaces. Maybe it would even be possible to specify a hybrid subdivision scheme, combining the advantages of the schemes for each of the three types of meshes, depending on local criteria.

11.2 Other applications benefiting local interpolation

As morphing between objects is quite related to free-form deformations, it makes sense to investigate how the locally interpolating subdivision can be put to use for that kind of application. Moreover, by combining free-form deformations and morphing into an animation tool, it would become

possible to create stunning animations needing little user effort. Physics-based methods could further facilitate intuitive control.

At the Expertise Centre for Digital Media, a new 2.5D rendering and automatic inbetweening tool is currently being investigated [DiFio01]. Subdivision curves and surfaces could also play a role there - for example, to control the texture mapping of these 2.5 D objects. The higher level of control described in this dissertation would also bring additional advantages and possibilities to this kind of applications.

11.3 Further investigation of hexagonal subdivision

The full mathematical analysis of our hexagonal subdivision scheme still has to be done, as is the case for subdivision surface schemes in general. For example, most of these schemes are tangent plane continuous, but not curvature continuous at extraordinary points. Mathematically, it is stated these surfaces are C^1, but not C^2. However, there is a strong feeling that they are "more" continuous than just C^1. To take an example of subdivision curves, the interpolating four-point scheme is C^1 but not C^2. Yet, between many possible ways of defining a curve that locally only uniformly depend on four surrounding control vertices, they provide a very smooth solution.

Another topic of further research is to investigate how the hexagonal scheme can be effectively combined with a triangular scheme to obtain surfaces with a higher order of continuity. For this, also a further investigation of the relation with box-splines and half-box-splines [Praut01], which are defined on purely regular meshes, would be very helpful.

Furthermore, just as Nasri's continuing research has improved possibilities for the quadrilateral corner-cutting Doo-Sabin scheme [Nasri87, Nasri99, Nasri00, Nasri01a, Nasri01b], also for the hexagonal scheme improved methods for borders, curve interpolation, sharp and semi-sharp edges and adaptive subdivision would be very welcome.

In other research areas, this new hexagonal scheme can also give new input to finite element methods and to wavelet analysis.

11.4 Putting functions into the weights

A very useful property of subdivision surfaces, is the ability to have rather high differences in density of the control points in different zones. This

permits the creation of fine detail in one zone, without the need to deal with too many control points all over the surface.

In many practical situations, however, the surface designer needs to be careful to arrange control points in the regions where dense and coarse zones meet each other. This problem is mainly caused by the uniform subdivision rules, assigning the same weights to nearby and more distant control points.

Therefore, an interesting area of research is the study of non-uniform subdivision schemes. To simplify, let us consider the subdivision scheme for a cubic B-spline. From the masks from section 2.2, the following formulas are derived. New vertices are inserted in the center of the existing ones:

$$p_{0,1}' = \frac{1}{2}\left(p_0 + p_1\right) \tag{11-1}$$

And the existing vertices are relaxed with their neighbors using:

$$p_0' = \frac{1}{4}\left(p_{-1} + 2p_0 + p_1\right) \tag{11-2}$$

As only two vertices are used in equation 11-1, it is better to leave it untouched so as not to incorporate vertices that are further away. Equation 11-2, on the other hand, can be adapted to give higher influence to points that are nearer. Therefore, we suggest calculating the new point as:

$$p_0' = \frac{1}{2}\left(w_{-1}p_{-1} + p_0 + w_1 p_1\right) \tag{11-3}$$

The weights depend on the distances between the points (we use Euclidean distances):

$$w_{-1} = \frac{dist\left(p_0, p_1\right)}{dist\left(p_{-1}, p_0\right) + dist\left(p_0, p_1\right)} \tag{11-4}$$

$$w_1 = \frac{dist\left(p_{-1}, p_0\right)}{dist\left(p_{-1}, p_0\right) + dist\left(p_0, p_1\right)} \tag{11-5}$$

This approach generates a subdivision curve, where nearby points more strongly influence the final form of the curve. The curve is not a cubic B-spline anymore, but still very smooth and more adapted to the form of the initial control polygon. A similar technique could be applied to subdivision surfaces, making their appearance more similar to the control mesh. This kind of

modification would make the scheme non-uniform, urging the development of new tools to analyze their properties.

One example of non-uniform subdivision to generate surfaces can be found in the work by Ivrissimtzis et al. [Ivris01]. They employ trigonometric calculations, projecting the configuration on a sphere or hyperbole to derive coefficients for approximating or interpolating schemes. By allowing the subdivision rules to be non-stationary and letting the weights depend on the dihedral angles between the faces, they created a non-stationary scheme which when used on a regular Platonic solid as input will result in a sphere.

12 Conclusions

In this dissertation, we investigated new techniques for recursive subdivision, both for curves and for surfaces. The main contributions are the construction of a new hexagonal scheme for subdivision surfaces and a new editing paradigm.

We introduced a new subdivision surface scheme based on hexagonal meshes. In recent classifications, the possible existence of these schemes seems to be ignored. Such hexagonal schemes are important, however, as they are the dual of existing triangular schemes. For quadrilateral schemes, is has been proven that using repeated averaging, alternating between the primal and the dual scheme, surfaces of higher continuity can be created. The development of a concrete hexagonal scheme is a first step in a similar setup for triangular schemes. In order to cope with already existing triangular meshes, methods to convert them to hexagonal meshes are included. It turns out that the hexagonal scheme can be applied in very similar ways as already existing subdivision schemes, resulting in high-quality surfaces. Furthermore, the new scheme has important properties such as simple construction rules and a small local support. The scheme a dual to the Sqrt(3) scheme, recently presented by Leif Kobbelt at Siggraph 2000 [Kobbe00].

The new editing paradigm is first worked out for subdivision curves enabling local interpolation in points that the user can indicate interactively. Moreover, the tool allows for easy control over the normal direction and includes a handy tension parameter.

Later, this tool was extended to also permit local interpolation for subdivision surfaces. Again, local interpolation and normal and tension control are provided as direct manipulation tools. Ghost points are introduced, whose position is carefully calculated. Both for the quadrilateral Catmull-Clark scheme and the triangular Loop scheme, adequate algorithms are designed.

Finally, to show that this approach is not only useful for interactive surface design, a new free-form deformation paradigm for 2D animated objects is also constructed. The subdivision techniques with our extensions turn out to be very powerful in combining precise local control with rapid global manipulations. Furthermore, the underlying subdivision enables explicit discontinuities.

Bibliography

[Adels87] E. H. Adelson, E. Simoncelli, R. Hingorani, "Orthogonal Pyramid Transforms for Image Coding", in *Proceedings SPIE, Visual Communication and Image Processing II, Cambridge, MA, Vol. 845*, pp. 50-58, October 1987.

[Aznar00] J.A. Aznar, M. Moreno, G.s Cristobal, "La Transformada de Malla Log-hexagonal: Muestreo Retiniano y Empaquetamiento de la Imagen", *available at http://revc.uab.es/revista/01/0102-abs.htm*, Revista Electrónica de Visión por Ordenador, REVC, Vol. 1,2, January 2000.

[Bajaj01] C. Bajaj, J. Warren, G. Xu, "A Smooth Subdivision Scheme for Hexahedral Meshes", *submitted to the special issue of The Visual Computer on subdivision*, available at http://www.cs.rice.edu/~jwarren/, April 2001.

[Ball86] A. Ball, D. Storry, "A Matrix Approach to the Analysis of Recursively Generated B-Spline Surfaces", in *Computer Aided Design*, 18(8), pp. 437-447, October 1986.

[Ball88] A. Ball, D. Storry, "Conditions for Tangent Plane Continuity over Recursively Generated B-Spline Surfaces", in *ACM Transactions on Graphics*, 7(2), pp. 83-102, April 1988.

[Barr84] A. Barr, "Global and Local Deformation of Solid Primitives", *Computer Graphics*, Vol.18, No.3 (Proc. Siggraph'84), pp. 21-30, 1984.

[Bierm00] H. Biermann, A. Levin, D. Zorin, "Piecewise Smooth Subdivision Surfaces with Normal Control", *Computer Graphics Proceedings (SIGGRAPH)*, Annual Conference Series, 2000, pp. 113-120.

[Bierm01] H. Biermann, I. M. Martin, D. Zorin, F. Bernardini, "Sharp Features on Multiresolution Subdivision Surfaces", to appear in *Proceedings of Pacific Graphics 2001*, Tokyo, Japan, October 2001.

[Bisch00] S. Bischoff, L. Kobbelt, H.-P. Seidel, "Towards Hardware Implementation of Loop Subdivision", in *Proceedings of Eurographics/SIGGRAPH Graphics Hardware Workshop 2000*, pp. 41–50, July 2000.

[Blair94] P. Blair, "Cartoon Animation", *Walter Foster Publishing Inc.*, ISBN : 1-56010-084-2, 1994.

[Cannon99] J. W. Cannon, W. J. Floyd, W. R. Parry, "Finite Subdivision Rules", *available at http://www.math.vt.edu/people/floyd/research/papers/fsr.html*, 1999.

[Cannon01] J. W. Cannon, W. J. Floyd, R. Kenyon, W. R. Parry, "Constructing Rational Maps from Subdivision Rules", preprint *available at http://www.math.vt.edu/people/floyd/research/papers/ratsub.html*

[Catmu78] E. Catmull, J. Clark, "Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes", *Computer-Aided Design*, 10 (Sept. 1978), pp.350–355.

[Certa96] A. Certain, J. Popovic, T. Duchamp, D. Salesin, W. Stuetzle, T. DeRose, "Interactive Multi-resolution Surface Viewing", *Computer Graphics (SIGGRAPH'96 Proceedings)*, pp.91-98, August 1996.

[Chaik74] G.M. Chaikin, "An Algorithm for High Speed Curve Generation", *Computer Graphics and Image Processing*, 1974, 3(4), pp. 346-349.

[Cirak00] F. Cirak, M. Ortiz, P. Schröder, "Subdivision Surfaces: A New Paradigm For Thin-Shell Finite-Element Analysis", *International Journal for Numerical Methods in Engineering*, 47:(12), pp. 2039-2072, April 2000.

[Claes00] J. Claes, F. Van Reeth, M. Ramaekers, "Locally Interpolating Subdivision Surfaces Supporting Free-Form 2D Deformation", *Proceedings of Deform2000*, Geneva. Switzerland, pp. 51-59, November 2000.

[Claes01a] J. Claes, K. Beets, F. Van Reeth, A. Iones and A. Krupkin, "Turning the Approximating Catmull-Clark Subdivision Scheme into a Locally Interpolating Surface Modeling Tool", in *Proceedings of Shape Modeling and Applications (SMI'01)*, pp.42-48, Genoa, Italy, May 2001.

[Claes01b] J. Claes, M. Ramaekers and F. Van Reeth, "Providing local interpolation, tension and normal control in the manipulation of Loop subdivision surfaces", in *Proceedings of CGI'01*, Hong-Kong, July 3 - 6, 2001, pp.299-305.

[Claes02] J. Claes, K. Beets, F. Van Reeth, "A Corner-Cutting Scheme for Hexagonal Subdivision Surfaces", submitted to *Shape Modeling and Applications (SMI'02)*, Calgary, Canada, May 2002.

[Coqui90] S. Coquillart, "Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling", in *Siggraph '90*, pp. 187-196, August 1990.

[Coqui91] S. Coquillart, "Animated Free-Form Deformation: An Interactive Animation Technique", in *Siggraph '91*, pp. 23-26, July 1991.

[DeBoor93] C. De Boor, K. Höllig, D. Riemenschneider, "Box Splines", Springer-Verlag Berlin, 1993.

[DeRham56] G. de Rham, "Sur une Courbe Plane", J. de Math. Pures & Appl. 35, pp. 25-42, 1956.

[DeRose98] T. DeRose, M. Kass, T. Truong, "Subdivision Surfaces in Character Animation", in *SIGGRAPH 98 Conference Proceedings*, pp.85–94, July 1998.

[DiFio01] F. Di Fiore, P. Schaeken, K. Elens, F. Van Reeth, "Automatic Inbetweening in Computer Assisted Animation by Exploiting 2.5D Modelling Techniques", *in Computer Animation 2001 Conference Proceedings*, pp. 192-200, November 2001.

[Doo78] D. Doo, M. Sabin, "Behaviour of Recursive Division Surfaces near Extraordinary Points", in *Computer-Aided Design,* 10, pp. 356–360, September 1978.

[Dubuc98] S. Dubuc, J.L. Merrien, P. Sablonnière , "The Length of the de Rham Curve", in the *Journal of Mathematical Analysis and Application 1998, Vol 223, Iss 1*, pp. 182-195, 1998.

[Dyn87] N. Dyn, D. Levin, J. Gregory, "A Four-point Interpolatory Subdivision Scheme for Curve Design", *Computer Aided Geometric Design,* 4, pp. 257-268, 1987.

[Dyn90] N. Dyn, J. A. Gregory, D. Levin, "A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control", *ACM Transactions on Graphics*, Vol. 9, No. 2, pp. 160–169, April 1990.

[Dyn93] N. Dyn, S. Hed, D. Levin, "Subdivision Schemes for Surface Interpolation", Workshop in Computational Geometry (1993), *World Scientific*, pp. 97-118.

[Dyn98] N. Dyn, F. Kuijt, D. Levin, R.M.J. Van Damme, "Convexity Preservation of the Four-Point Interpolatory Subdivision Scheme", University of Twente (Enschede, The Netherlands), *memorando 1457*, August 1998.

[Eck95] M. Eck, T. DeRose, T. Duchamp, "Multiresolution Analysis of Arbitrary Meshes", in *Proceeding of SIGGRAPH '95*, pp.173-182, August 1995.

[Feng96] J. Feng, L. Ma, Q. Peng, "A New Free-Form Deformation Through the Control of Parametric Surfaces", in *Computers & Graphics*, Vol. 20, No. 4, pp. 531-539, 1996.

[Ferha01] H. Ferhatosmanoglu, D. Agrawal, A. El Abbadi, "Optimal Partitioning for Efficient I/O in Spatial Databases", in the *Proceedings of the European Conference on Parallel Computing (Euro-Par), Parallel I/O and Storage Technology*, Manchester, United Kingdom, August 2001.

[Foley91] J. Foley, A. Van Dam, S. Feiner, J. Hughes, "Computer Graphics – Principles and Practice", *Addison-Wesley Publishing Company*, 1991.

[Fosne96] R. Fosner, "OpenGL Programming for Windows 95 and Windows NT", *Addison-Wesley developers Press*; ISBN: 0201407094, 1996.

[Hales99] T. C. Hales, "The Honeycomb Conjecture", *available at http://xxx.lanl.gov/abs/math.MG/9906042*, June 1999.

[Halst93] M. Halstead, M. Kass, T. DeRose, "Efficient, Fair Interpolation Using Catmull-Clark Surfaces", in the *Proceedings of SIGGRAPH '93*, pp. 35-44, July 1993.

[He97] W. He, M.-J. Lai, "On the Digital Filter Associated with Bivariate Box Spline Wavelets", in *Journal of Electronic Imaging*, 6(1997), pp. 453-466, 1997.

[Hoppe93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonalds, W. Stuetzle, "Surface Reconstruction from Unorganized Points", in *Computer Graphics (Siggraph '93)*, 26(2), pp.71-78, July 1993.

[Hoppe94] H. Hoppe, "Surface Reconstruction from Unorganized Points", Ph.D. thesis, *Department of Computer Science and Engineering, University of Washington, TR-94-06-01*, June 1994.

[Hoppe96] H. Hoppe, "Progressive Meshes", in C*omputer Graphics (SIGGRAPH 1996 Proceedings)*, pp. 99-108, August 1996.

[Hubeli00] A. Hubeli, M. Gross, "A Survey of Surface Representations", ETH Zürich, CS Technical Report #335, *Institute of Scientific Computing*, February 28, 2000.

[Hubeli01] A. Hubeli, "Subdivision Surfaces – IEEE Visualization '01 Tutorial", *available at http://www.cg.inf.ethz.ch/~hubeli/Vis2001/Tutorial.html*, 2001.

[Inter97] V. Interrante, H. Fuchs, S. Pizer, "Conveying the 3D Shape of Smoothly Curving Transparent Surfaces via Texture", *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 98-117, April-June 1997.

[Ivris01] I. Ivrissimtzis, N. Dogdson, M. Hassan, "On the Geometry of Recursive Subdivision", preprint *available at http://www.cl.cam.ac.uk/~ipi20/recentresearch.html* , 2001.

[Joy96] K. Joy, "On-Line Geometric Modeling Notes", webpages available at *http://muldoon.cipic.ucdavis.edu/CAGDNotes/,* 1996.

[Junki00] S. Junkins, A. Hux, "Subdivision Reality – Employing Subdivision Surfaces for Real-time Scalable 3D", in *Proceedings of the Game Developers Conference 2000*, available at http://www.gdconf.com/archives/proceedings/2000/prog_papers.html

[Kobbe96] L. Kobbelt, "Interpolatory Subdivision on Open Quadrilateral Nets with Arbitrary Topology", in *Computer Graphics Forum,* 15 (Eurographics'96 conference proceedings), pp. 409-420, 1996.

[Kobbe97] L. Kobbelt, P. Schröder, "Constructing Variationally Optimal Curves through Subdivision", *Technical Report 97-05*, California Institute of Technology, Department of Computer Science, 1997.

[Kobbe98] L. Kobbelt, "Using the Discrete Fourier-Transform to Analyze the Convergence of Subdivision Schemes", *available at http://www9.informatik.uni-erlangen.de/~Kobbelt/papers*, 1998.

[Kobbe00] L. Kobbelt, "Sqrt(3) Subdivision", in *Proceedings of the Conference on Computer Graphics (SIGGRAPH 2000)*, pp. 103-112, July 2000.

[Khoda00] A. Khodakovsky, P. Schröder, W. Sweldens. "Progressive Geometry Compression", in Siggraph'2000 Conference Proceedings, pp. 271-278, July 2000.

[Labsik00a] U. Labsik, G. Greiner, "Interpolatory Sqrt(3)-Subdivision", in *Computer Grahics Forum (Eurographics 2000 Proceedings)*, pp. 131-138, 2000.

[Labsik00b] U. Labsik, L. Kobbelt, R. Schneider, H.-P. Seidel, "Progressive Transmission of Subdivision Surfaces", in *Computational Geometry*, pp. 25-39, 2000.

[Laine93] A. F. Laine, S. Schuler, W. Huda, J. C. Honeyman, B. Steinbach, "Hexagonal Wavelet Processing of Digital Mammography", in Medical Imaging 1993: Image Processing, *Proceedings of SPIE*, Newport Beach, CA, vol. 1898, pp. 559-573, February 1993.

[Lane80] J. Lane, R. Riesenfield, "A Theoretical Development for the Computer Generation and Display Piecewise Polynomial Surfaces", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1), pp. 35-46, 1980.

[Lee00] A. Lee, H. Moreton, H. Hoppe, "Displaced Subdivision Surfaces" in *Computer Graphics (SIGGRAPH 2000 Proceedings)*, pages 85-94, July 2000.

[Levin00] A. Levin, "Surface Design Using Locally Interpolating Subdivision Schemes", in *Journal of Approximation Theory*, Vol. 104, No. 1, pp. 98-120, May 2000.

[Litke01] N. Litke, A. Levin, P. Schröder, "Fitting Subdivision Surfaces", in *Proceedings of Scientific Visualization 2001*.

[Loop87] C. Loop, "Smooth Subdivision Surfaces Based on Triangles", *Master's thesis, University of Utah*, Department of Mathematics, 1987.

[Lundm99] A. Lundmark, N. Wadströmer, H. Li, "Recursive Subdivision of the Plane Yielding Nearly Hexagonal Regions", *RadioVetenskap och Komunikation*, June 1999.

[Lundm01] A. Lundmark, N. Wadströmer, H. Li, "Hierarchical Subsampling Giving Fractal Regions", *IEEE Transactions on Image Processing*, vol. 10, no. 1, pp. 167-173, January 2001.

[MacCr96] R. MacCracken, K. Joy, "Free-Form Deformations with Lattices of Arbitrary Topology", *Siggraph'96*, pp.181-188, August 1996.

[Mandal99] C. Mandal, Hong Qin , Baba C. Vemuri, "A Novel FEM-based Dynamic Framework for Subdivision Surfaces", in *Proceedings of the fifth symposium on Solid modeling and applications*, pp.191-202, June 1999.

[Maya01] http://www.aliaswavefront.com

[McDon00] K. T. McDonnell, H. Qin. "Dynamic Sculpting and Animation of Free-Form Subdivision Solids", in *Proceedings of IEEE Computer Animation 2000*, pp. 126-133, May 2000.

[Melkm97] A. A. Melkman, "Subdivision Schemes with Non-negative Masks Converge Always - Unless They Obviously Cannot?", in *The heritage of P. L. Chebyshev: a Festschrift in honor of the 70th birthday of T. J. Rivlin., Ann. Numer. Math. 4*, pp. 451-460, 1997.

[Mocco97] L. Moccozet, N. Magnenat Thalmann, "Dirichlet Free-Form Deformations and their Applications to Hand Simulation", *Proceedings of Computer Animation*, IEEE Computer Society, pp. 93-102, 1997.

[Muell96] T. Mueller, "Geometric Modelling with Multivariate B-Splines", *Ph.D. dissertation, University of Utah*, Dept. of Computer Science, 1996.

[Nasri87] A. Nasri, "Polyhedral Subdivision Methods for Free-Form Surfaces", *Communications of the ACM*, Transactions on Graphics, Vol. 6, No. 1, pp. 29-73, January 1987.

[Nasri99] A. Nasri, "Free-Form Curve Generation by Recursive Subdivision of Polygonal Strip Complexes", Sketch, *ACM Siggraph 99*, Los Angeles, pp. 8-13, August 1999.

[Nasri00] A. Nasri, "Interpolating Meshes of Boundary Intersecting Curves by Subdivision Surfaces", in *The Visual Computer Journal*, Vol. 16, No. 1, pp. 3-14, 2000.

[Nasri01a] A. Nasri, T. Kim, K. Lee, "Fairing Recursive Subdivision Surfaces with Curve Interpolation Constraints", in *Proceedings of Shape Modeling and Applications (SMI'01)*, pp. 49-59.

[Nasri01b] A. Nasri, "Constructing Polygonal Complexes with Shape Handles for Curve Interpolation by Subdivision Surfaces", in *Computer Aided Design,* 33, pp.753-765, September 2001.

[O'Rour94] J. O'Rourke, "Computational Geometry in C", *Cambridge University Press*, ISBN: 0521445922, 1994.

[Peters97] J. Peters, U. Reif, "The Simplest Subdivision Scheme for Smoothing Polyhedra", in *ACM Transactions on Graphics*, vol. 16, no. 4, pp. 420-431, October 1997.

[Peters00] J. Peters, G. Umlauf, "Gaussian and Mean Curvature of Subdivision Surfaces", in R. Cipolla and R. Martin, editors, *The Mathematics of Surfaces IX*, pp. 59-69, Springer, 2000.

[Porter00] T. Porter, G. Susman, "Creating Lifelike Characters in Pixar Movies", *Communications of the ACM*, Vol.43, no.1, pp. 25-29, January 2000.

[Praut97] H. Prautzsch, U. Reif, "Necessary Conditions for Subdivision Surfaces", *Technical report*, Sonderforschungsbereich 404, Universitat Stuttgart, Bericht 97/04, 1997.

[Praut99] H. Prautzch, G. Umlauf, "Triangular G^2-Splines", in: P.-L. Laurent, P. Sablonniere, L.L. Schumaker (eds.), Curve and Surface Design, *Vanderbilt University Press*, pp.335-342, 1999.

[Praut00] H. Prautzch, G. Umlauf, "A G^1 and G^2 Subdivision Scheme for Triangular Nets", in *International Journal for Shape Modelling*, 6(1), pp. 21-35, 2000.

[Praut01] H. Prautzsch, W. Böhm, "Box Splines", in: *The Handbook of Computer Aided Geometric Design*, Farin, Hoschek, Kim (eds.) to appear (Elsevier), 2001/2.

[Pulli96] K. Pulli, M. Segal, "Fast Rendering of Subdivision Surfaces", *Proceedings of 7th Eurographics Workshop on Rendering*, June 1996.

[Qin98] H. Qin, B. Vemuri, "Dynamic Catmull-Clark Subdivision Surfaces", *Siggraph 98*, pp. 215-229, July 1998.

[Rayma01] C. Raymaekers, K. Beets, F. Van Reeth, "Fast Haptic Rendering of Complex Objects Using Subdivision Surfaces", in the *Proceedings of the Sixth PHANToM Users Group Workshop*, Aspen, USA, October 2001.

[Reif95] U. Reif, "A Unified Approach to Subdivision Schemes near Extraordinary Vertices", in *Computer Aided Geometric Design*, 12(2), pp. 153-174, 1995.

[Reif00] U. Reif, P. Schröder, "Curvature Smoothness of Subdivision Surfaces", TR-00-03, Caltech, *Department of Computer Science*, 2000.

[Riese75] R. Riesenfield, "On Chaikin's Algorithm", *Computer Graphics and Image Processing*, 4(3), pp. 304-310, 1975

[Sabin01] M. Sabin, "Subdivision: Tutorial Notes", tutorial of Shape Modeling International (SMI'01), Genoa (Italy), May 2001.

[Sahr98] K. Sahr, D. White, "Discrete Global Grid Systems", in Proceedings of the 30th Symposium on the Interface, *Computing Science and Statistics, 30*, pp. 269-278, 1998.

[Schwe96] J. Schweitzer, "Analysis and Application of Subdivision Surfaces", PhD dissertation, Department of Computer Science and Engineering, University of Washington, *Technical Report UW-CSE-96-08-02*, August 1996.

[Seder86] T. Sederberg, S. Parry, "Free-Form Deformation of Solid Geometric Models", *Siggraph '86*, pp. 151-160, August 1986.

[Seder93] T. Sederberg, P. Gao, G. Wang, H. Mu, "2-D Shape Blending: An Intrinsic Solution to the Vertex Path Problem", in *Proceedings of Siggraph '93*, pp. 15-18, July 1993.

[Simoe01] J. Simoens, N. Dyn, D. Levin, "Face Value Subdivision Schemes on Triangulations by Repeated Averaging", in preparation, 2001.

[Skaria01] S. Skaria. E. Akleman, F. I. Parke, "Modeling Subdivision Control Meshes for Creating Cartoon Faces", in *Proceedings of Shape and Modeling International 2001*, pp.216-227, May 2001.

[Stam98] J. Stam, "Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values", *(Siggraph98) Computer Graphics Proceedings*, ACM SIGGRAPH, 1998, pp.395-404, July 1998.

[Stam99] J. Stam, "Evaluation of Loop Subdivision Surfaces", *SIGGRAPH'99 Course Notes*, August 1999.

[Stam01] J. Stam, "On Subdivision Schemes Generalizing Uniform B-Spline Surfaces of Arbitrary Degree", in *Computer Aided Geometric Design*, Special Edition on Subdivision Surfaces, Volume 18, pp. 383-396, 2001.

[Stoll96] E. J. Stollnitz, T. D. DeRose, D. H. Salesin, "Wavelets for Computer Graphics", *Morgan Kaufmann Publishers*, San Francisco, 1996.

[Umlauf00] G. Umlauf, "Analyzing the Characteristic Map of Triangular Subdivision Schemes", in *Constructive Approximation,* 16 (1), pp. 145-155, 2000.

[VanRe01] F. Van Reeth, J. Claes, "Interpolatory Uniform Subdivision Curves with Normal Interpolation and Tension Control, Generating B-splines of Any Degree", *submitted to The Visual Computer*.

[Velho01a] L. Velho, D. Zorin, "4-8 Subdivision", in *Computer-Aided Geometric Design*, 18(5), pp. 397-427, Special Issue on Subdivision Techniques, 2001.

[Velho01b] L. Velho, K. Perlin, L. Ying, H. Biermann, "Procedural Shape Synthesis on Subdivision Surfaces", in *Proceedings of the Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, 2001.

[Vince99] A. Vince, "Self-replicating Tiles and Their Boundary", in *Discrete and Computational Geometry*, 21, pp. 463-476, 1999.

[Warren95] J. Warren, "Subdivision Methods for Geometric Design", unpublished manuscript, Department of Computer Science, Rice University, Preprint *available at http://www.cs.rice.edu/jwarren/papers/book.ps.gz*, November 1995.

[Watson87] A. B. Watson, A. J. Ahumada Jr, "An Orthogonal Oriented Quadrature Hexagonal Image Pyramid", *NASA Technical Memorandum NASA TM-100054, NASA,* December 1987.

[Weimer98] H. Weimer, J. Warren, "Subdivision Schemes for Thin Plate Splines", in *Proceedings of Eurographics '98, Computer Graphics Forum*, Vol. 17, No. 3, pp. 303-313 & 392, 1998.

[Weimer99] H. Weimer, J. Warren, "Subdivision Schemes for Fluid Flow", *SIGGRAPH 99 conference proceedings*, pp. 111-120, August 1999.

[Ying01] L. Ying, D. Zorin, "Non-manifold Subdivision", *Proceedings of IEEE Visualization 2001*, to appear (also available at http://mrl.nyu.edu/publications/), 2001.

[Zhang01] C. Zhang, P. Zhang, F. Cheng, "Fairing Spline Curves and Surfaces by Minimizing Energy", *Computer-Aided Design*, 33, pp. 913-923, 2001.

[Zonen98] R. Zonenschein, J. Gomes, L. Velho, L. H. de Figueredo, M. Tigges, B. Wyvill, "Texturing Composite Deformable Implicit Objects", *Proceedings of the XI International Symposium on Computer Graphics, Image Processing and Vision*, pp. 346-353, Rio de Janeiro, October 1998.

[Zorin96] D. Zorin, P. Schröder, W. Sweldens, "Interpolating Subdivision for Meshes with Arbitrary Topology", in *Proceedings of SIGGRAPH 1996*, pp. 189-192, August 1996.

[Zorin00a] D. Zorin, P. Schröder, A. Levin, L. Kobbelt, W. Sweldens, T. DeRose, "Subdivision for Modeling and Animation", course SIGGRAPH, July 2000.

[Zorin00b] D. Zorin, "A Method for Analysis of C^1-continuity of Subdivision Surfaces", *SIAM Journal of Numerical Analysis*, vol. 37, no. 5, 2000, pp. 1677-1708.

[Zorin01a] D. Zorin, P. Schröder, "A Unified Framework for Primal/Dual Quadrilateral Subdivision Schemes" to appear in CAGD, 2001 (also *available at http://mrl.nyu.edu/publications/*).

[Zorin01b] D. Zorin, D. Kristjansson, "Evaluation of Piecewise Smooth Subdivision Surfaces", to appear in The Visual Computer (also *available at http://mrl.nyu.edu/publications/*), 2001.

List of figures

List of tables

Appendix 1: Invariance conditions for cubic curves

In this appendix, we give a proof of the invariance conditions from section 6.2 of chapter 6 applying to locally interpolating cubic curves. That section also contains the meaning of the variables together with the necessary illustrations.

$$
\begin{aligned}
c_{2i}^{j+1} &= \tfrac{1}{4}\widehat{c}_{2i-1}^{j+1} + \tfrac{1}{2}\widehat{c}_{2i}^{j+1} + \tfrac{1}{4}\widehat{c}_{2i+1}^{j+1} \\
&= \tfrac{1}{4}\tfrac{1}{2}(c_{i-1}^{j} + c_{i}^{j}) + \tfrac{1}{2}c_{i}^{j} + \tfrac{1}{4}\tfrac{1}{2}(c_{i}^{j} + c_{i+1}^{j}) \\
&= \tfrac{1}{8}c_{i-1}^{j} + \tfrac{6}{8}c_{i}^{j} + \tfrac{1}{8}c_{i+1}^{j} \\
&= \tfrac{1}{8}c_{i-1}^{j} + \tfrac{6}{8}\tfrac{1}{2}(c_{i-1}^{j} + c_{i+1}^{j}) + \tfrac{1}{8}c_{i+1}^{j} \\
&= \tfrac{1}{2}c_{i-1}^{j} + \tfrac{1}{2}c_{i+1}^{j} \\
&= c_{i}^{j} \\
c_{2i-1}^{j+1} &= \tfrac{1}{4}\widehat{c}_{2i-2}^{j+1} + \tfrac{1}{2}\widehat{c}_{2i-1}^{j+1} + \tfrac{1}{4}\widehat{c}_{2i}^{j+1} \\
&= \tfrac{1}{4}c_{i-1}^{j} + \tfrac{1}{2}\widehat{c}_{2i-1}^{j+1} + \tfrac{1}{4}c_{i}^{j} \\
&= \tfrac{1}{2}\tfrac{1}{2}(c_{i-1}^{j} + c_{i}^{j}) + \tfrac{1}{2}\widehat{c}_{2i-1}^{j+1} \\
&= \widehat{c}_{2i-1}^{j+1}
\end{aligned}
$$

$$
\begin{aligned}
c_{2i+1}^{j+1} &= \tfrac{1}{4}\hat{c}_{2i}^{j+1} + \tfrac{1}{2}\hat{c}_{2i+1}^{j+1} + \tfrac{1}{4}\hat{c}_{2i+2}^{j+1} \\
&= \tfrac{1}{4}c_i^j + \tfrac{1}{2}\hat{c}_{2i+1}^{j+1} + \tfrac{1}{4}c_{i+1}^j \\
&= \tfrac{1}{2}\tfrac{1}{2}(c_i^j + c_{i+1}^j) + \tfrac{1}{2}\hat{c}_{2i+1}^{j+1} \\
&= \hat{c}_{2i+1}^{j+1}
\end{aligned}
$$

$\hat{c}_{2i-1}^{j+1}$ and $\hat{c}_{2i+1}^{j+1}$, being the midpoints of two equidistant point to c_i^j, are equidistant to c_i^j. It follows immediately from the above equations that we have a local invariance on level *j+1* around c_{2i}^{j+1}.

Appendix 2: Invariance conditions for curves of any degree

In this appendix, we give a proof of the invariance conditions from section 6.3 of chapter 6 applying to locally interpolating curves of any degree.

The approach is slightly different depending on *n* (in equation 6-3 of chapter 6) being even or odd:

n (in equation 3) even:

Theorem: The intermediate control points $(\widehat{c}_{2i+k}^{\,j+1})_{-m\le k\le m}$ will not move due to the averaging step.

Proof for situation around $\widehat{c}_{2i}^{\,j+1}(=c_i^j)$. The cases for $(\widehat{c}_{2i+k}^{\,j+1})_{-m\le k\le m}$ give the same result due to the collinearity of the $(\widehat{c}_{2i+k}^{\,j+1})_{-2m\le k\le 2m}$ and the equidistance of the $(\widehat{c}_{2i+k}^{\,j+1})_{-2m\le k\le 2m}$ among themselves.

$$\sum_{k=-n/2}^{n/2} {}^{1}\!/_{2^n} \binom{n}{\frac{n}{2}+k} \widehat{c}_{2i+k}^{\,j+1}$$ (the averaging operation of $\widehat{c}_{2i}^{\,j+1}$)

= (bring constant in front and split summation in three parts)

$${}^{1}\!/_{2^n} \left(\sum_{k=-n/2}^{-1} \binom{n}{\frac{n}{2}+k} \widehat{c}_{2i+k}^{\,j+1} + \binom{n}{\frac{n}{2}} \widehat{c}_{2i}^{\,j+1} + \sum_{k=1}^{n/2} \binom{n}{\frac{n}{2}+k} \widehat{c}_{2i+k}^{\,j+1} \right)$$

= (add and subtract terms)

$$\frac{1}{2^n}\Big(\sum_{k=-n/2}^{-1}\binom{n}{\frac{n}{2}+k}\widehat{c}_{2i+k}^{\,j+1}-\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}\widehat{c}_{2i}^{\,j+1}-\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}\widehat{c}_{2i}^{\,j+1}+$$

$$2\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}\widehat{c}_{2i}^{\,j+1}+\binom{n}{\frac{n}{2}}\widehat{c}_{2i}^{\,j+1}+\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}\widehat{c}_{2i+k}^{\,j+1}\Big)$$

= (rewrite term 1 and move term 3)

$$\frac{1}{2^n}\Big(\sum_{k=n/2}^{1}\binom{n}{\frac{n}{2}-k}\widehat{c}_{2i-k}^{\,j+1}-\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}\widehat{c}_{2i}^{\,j+1}+2\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}\widehat{c}_{2i}^{\,j+1}+$$

$$\binom{n}{\frac{n}{2}}\widehat{c}_{2i}^{\,j+1}+\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}\widehat{c}_{2i+k}^{\,j+1}-\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}\widehat{c}_{2i}^{\,j+1}\Big)$$

= ($\binom{n}{\frac{n}{2}-k}=\binom{n}{\frac{n}{2}+k}$ and regroup some terms)

$$\frac{1}{2^n}\Big(\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}(\widehat{c}_{2i-k}^{\,j+1}-\widehat{c}_{2i}^{\,j+1})+2\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}\widehat{c}_{2i}^{\,j+1}+\binom{n}{\frac{n}{2}}\widehat{c}_{2i}^{\,j+1}+$$

$$\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}(\widehat{c}_{2i+k}^{\,j+1}-\widehat{c}_{2i}^{\,j+1})\Big)$$

= (the equidistance of corresponding points implies $(\widehat{c}_{2i-k}^{\,j+1}-\widehat{c}_{2i}^{\,j+1})=-(\widehat{c}_{2i+k}^{\,j+1}-\widehat{c}_{2i}^{\,j+1})$)

$$\frac{1}{2^n}\left(2\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}\hat{c}_{2i}^{j+1}+\binom{n}{\frac{n}{2}}\hat{c}_{2i}^{j+1}\right)$$

=

$$\frac{1}{2^n}\left(\left(2\sum_{k=1}^{n/2}\binom{n}{\frac{n}{2}+k}+\binom{n}{\frac{n}{2}}\right)\hat{c}_{2i}^{j+1}\right)$$

= (the entries in a row of the Pascal triangle sum to 2^n).

$$\hat{c}_{2i}^{j+1}$$

n (in equation 3) odd:

Theorem: Each intermediate control point $(\hat{c}_{2i+k}^{j+1})_{-m\le k\le m}$ will become midpoint of the two surrounding control points due to the averaging step.

Proof for situation around $\hat{c}_{2i}^{j+1}$ (the point to be interpolated). The cases for $(\hat{c}_{2i+k}^{j+1})_{-m\le k\le m}$ give the same result due to the collinearity of the $(\hat{c}_{2i+k}^{j+1})_{-2m\le k\le 2m}$ and the equidistance of the $(\hat{c}_{2i+k}^{j+1})_{-2m\le k\le 2m}$ among themselves.

Midpoint of the two surrounding control points:

$$\frac{1}{2}\left(\sum_{k=-\frac{n-1}{2}-1}^{(n-1)/2}\frac{1}{2^n}\binom{n}{\frac{n-1}{2}+1+k}\hat{c}_{2i+k}^{j+1}+\sum_{k=-(n-1)/2}^{\frac{(n-1)}{2}+1}\frac{1}{2^n}\binom{n}{\frac{n-1}{2}+k}\hat{c}_{2i+k}^{j+1}\right)$$

= (bring common factor in front and redefine k in term 2)

$$\frac{1}{2^{n+1}}\left(\sum_{k=-\frac{n-1}{2}-1}^{(n-1)/2}\binom{n}{\frac{n-1}{2}+1+k}\hat{c}_{2i+k}^{j+1}+\sum_{k=-\frac{n-1}{2}-1}^{(n-1)/2}\binom{n}{\frac{n-1}{2}-k}\hat{c}_{2i-k}^{j+1}\right)$$

$=\left(\begin{array}{c} n \\ \frac{n-1}{2}+1+k \end{array}\right)=\left(\begin{array}{c} n \\ \frac{n-1}{2}-k \end{array}\right)$, add and delete $2\sum_{k=-\frac{n-1}{2}-1}^{(n-1)/2}\left(\begin{array}{c} n \\ \frac{n-1}{2}-k \end{array}\right)\widehat{c}_{2i}^{\,j+1}$ and regroup some terms)

$$\frac{1}{2^{n+1}}\Big(\sum_{k=-\frac{n-1}{2}-1}^{(n-1)/2}\left(\begin{array}{c} n \\ \frac{n-1}{2}-k \end{array}\right)(\widehat{c}_{2i+k}^{\,j+1}-\widehat{c}_{2i}^{\,j+1})+$$

$$\sum_{k=-\frac{n-1}{2}-1}^{(n-1)/2}\left(\begin{array}{c} n \\ \frac{n-1}{2}-k \end{array}\right)(\widehat{c}_{2i-k}^{\,j+1}-\widehat{c}_{2i}^{\,j+1})+2\sum_{k=-\frac{n-1}{2}-1}^{(n-1)/2}\left(\begin{array}{c} n \\ \frac{n-1}{2}-k \end{array}\right)\widehat{c}_{2i}^{\,j+1}\Big)$$

= (the equidistance of corresponding points implies $(\widehat{c}_{2i-k}^{\,j+1}-\widehat{c}_{2i}^{\,j+1})=-(\widehat{c}_{2i+k}^{\,j+1}-\widehat{c}_{2i}^{\,j+1})$)

$$\frac{1}{2^{n+1}}\Big(2\sum_{k=-\frac{n-1}{2}-1}^{(n-1)/2}\left(\begin{array}{c} n \\ \frac{n-1}{2}-k \end{array}\right)\widehat{c}_{2i}^{\,j+1}\Big)$$

= (redefine k)

$$\frac{1}{2^{n+1}}\Big(2\sum_{k=0}^{n}\binom{n}{k}\widehat{c}_{2i}^{\,j+1}\Big)$$

= (the entries in a row of the Pascal triangle sum to 2^n).

$\widehat{c}_{2i}^{\,j+1}$

Samenvatting

Alhoewel subdivisie-oppervlakken reeds meer dan 20 jaar gekend zijn in de academische wereld, werden ze pas zeer onlangs een populaire representatie voor willekeurige oppervlakken, zowel in de wereld van de computeranimatie als voor industriële ontwerpen. Een belangrijke rol in deze populariteit werd gespeeld door hun succesvolle toepassing in een kortfilm van het beroemde productiehuis Pixar [DeRose98]. Voor hun animatiefilm, "Geri's Game", pasten ze voor het eerst subdivisie-oppervlakken toe als basisconcept voor alle geanimeerde oppervlakken, zoals het hoofd en het jasje van "Geri", het sympathieke hoofdpersonage. Later perfectioneerde Pixar deze technieken voor de productie van "Toy Story II" [Porter00].

Subdivisie-oppervlakken vormen een compacte representatie van gladde oppervlakken en bieden een aantal voordelen ten opzichte van andere bestaande representaties. Als alternatief voor subdivisie-oppervlakken wordt vaak gebruik gemaakt van parametrische patches. Dit soort patches, zoals de gekende B-spline-patches, zijn echter slechts gedefinieerd op een eenvoudig rechthoekig of driehoekig domein. Om oppervlakken van een willekeurige topologie (met gaten en uitsteeksels) te kunnen representeren, moeten verscheidene patches aan elkaar vastgehecht worden. De scheidingsranden zijn echter niet gedefinieerd in dezelfde representatie, wat dikwijls aanleiding geeft tot oneffenheden en scheurtjes op deze randen, vooral wanneer de oppervlakken geanimeerd worden. Een echte nachtmerrie, die de artistieke vrijheid van de animators sterk beknot.

Met subdivisie-oppervlakken kunnen dit soort oppervlakken echter wel in hun geheel via een eenduidig paradigma gedefinieerd worden. Subdivisie-oppervlakken vertrekken van een polygonale mesh die een ruwe benadering van het oppervlak weergeeft. Zo een mesh bestaat uit een reeks 3D punten, die via polygonen met elkaar verbonden zijn. In een eerste stap wordt de ruwe mesh vervangen door een iets fijnere mesh, waarbij de punten uitgemiddeld worden. Dit proces wordt recursief herhaald en resulteert in de limiet in een glad oppervlak.

De basis van de huidige kennis rond subdivisie-oppervlakken, werd in 1974 door Chaikin gelegd, die opmerkte dat door het repetitief afsnijden van de hoeken van een polygoon, uiteindelijk een gladde curve bekomen wordt [Chaik74]. Later bewezen Lane en Reisenfeld dat Chaikin's curve een kwadratische B-spline definieert, en ze breidden het concept verder uit naar B-splines van willekeurige orde [Lane80]. In de beginjaren werd dit subdivisie-principe vooral gebruikt om zoekalgoritmes te versnellen, bv. voor het vinden van de snijpunten tussen twee curves. Eigenlijk was Chaikin niet de eerste in het publiceren van subdivisie-curves. Later werd duidelijk dat een Frans wiskundige, G. de Rham, reeds in de jaren 50 een gelijkaardig schema beschreven had, dat later echter in de vergetelheid raakte [DeRham56].

Het eerste schema voor subdivisie-oppervlakken, het Catmull-Clark schema, werd in 1978 beschreven door Ed Catmull en Jim Clark, die zich baseerden op het repetitief opsplitsen van vierhoeken in vier kleinere vierhoeken [Catmu78]. De nieuwe punten werden berekend als een uitmiddeling van de omliggende bestaande punten, waarna ook de bestaande punten uitgemiddeld werden. Voor een regulier rooster past deze methode eigenlijk gewoon tweemaal een derdegraads subdivisie toe, een tensorproduct in twee loodrecht op elkaar staande richtingen. Catmull en Clarks belangrijkste inzicht was echter de uitbreiding van deze principes naar roosters die niet uitsluitend uit vierhoeken bestaan. Met deze roosters kunnen dan gesloten oppervlakken gedefinieerd worden, waardoor eensklaps oppervlakken van een willekeurige topologie in naadloos hetzelfde concept opgenomen worden.

Gedurende hetzelfde jaar, 1978, toonden Donald Doo en Malcolm Sabin een variant van subdivisie-oppervlakken gebaseerd op het tensorproduct van twee kwadratische curves [Doo78]. Visueel komt deze methode overeen met het wegsnijden van de hoeken en randen van een polygonale mesh. Hun schema werd bekend onder de naam Doo-Sabin-subdivisie en heeft als een interessante eigenschap dat de gegenereerde limietoppervlakken de middelpunten van de originele polygonen interpoleren. Het Doo-Sabin schema genereert meshes die het duaal zijn van de meshes gegenereerd via het Catmull-Clark schema: overal waar het Catmull-Clark schema een nieuwe polygoon definieert, definieert het Doo-Sabin schema een nieuw punt in het midden van die polygoon. Deze punten worden dan weer verbonden tot polygonen, die op hun beurt rond de punten van het Catmull-Clark schema heen lopen.

In 1987 presenteerde Charles Loop een subdivisie-techniek die volledig gebaseerd is op een uit driehoeken bestaande mesh [Loop87]. Hij breidde

eigenlijk een bestaand spline-type, met name een vierdegraads box-spline, dat reeds gedefinieerd was voor reguliere driehoekige roosters, uit naar roosters die ook niet-reguliere punten bevatten.

De schema's van Catmull-Clark, Doo-Sabin en Loop zijn allen benaderende schema's. De gegenereerde oppervlakken benaderen slechts hun controlepunten, zonder ze effectief te interpoleren. In 1990 stelden Dyn, Levin en Gregory het eerste schema voor dat zijn controlepunten allemaal interpoleert [Doo90]. Doordat de regels voor het schema beschreven worden uitgaande van een configuratie van driehoeken die op een vlinder lijkt, werd het schema bekend onder de naam "Butterfly-schema". Later breidden Zorin, Schröder en Sweldens dit schema uit zodat onder alle omstandigheden een C^1-oppervlak bekomen wordt [Zorin96].

Het kunnen representeren van oppervlakken met een willekeurige topologie is vooral interessant, omdat het ook toelaat dat een oppervlak op verschillende plaatsen met een verschillende dichtheid van controlepunten gedefinieerd kan worden. Er kan daarom locaal met fijn detail gewerkt worden zonder de noodzaak om ook elders een grote massa punten te introduceren.

Ondanks hun recente populariteit, zijn de eigenschappen van subdivisie-oppervlakken grondig wiskundig geanalyseerd. Dit hebben ze o.a. te danken aan hun sterke relatie met wavelet- en multi-resolutie-analyse [Stoll96], met tal van toepassingen in vele wetenschappelijke en technische domeinen.

In vele praktische toepassingen blijkt dat volledig gladde oppervlakken niet voldoende zijn. Een belangrijke uitbreiding vormt dan ook de mogelijkheid om scherpe en halfscherpe randen te creëren. Gelukkig past dit concept volledig in het paradigma van subdivisie-oppervlakken. Het werd o.a. uitgebreid toegepast in "Geri's Game".

Doordat subdivisie-oppervlakken gedefinieerd zijn op een polygonaal model, zijn ze ook vrij intuïtief te editeren. Polygonen kunnen makkelijk verplaatst en geroteerd worden om het oppervlak te vervormen en punten kunnen op een eenvoudige manier ingevoegd en verwijderd worden. Gelukkig is het voor subdivisie-oppervlakken geen vereiste dat de punten van de polygonen in eenzelfde plat vlak liggen.

De voornaamste bijdragen van deze doctoraatsverhandeling situeren zich in twee domeinen. Enerzijds ontwikkelden we een techniek die het toelaat dat een benaderend subdivisie-schema toch een aantal van zijn controlepunten

interpoleert. Anderzijds werkten we een volledig nieuw subdivisie-schema uit, gebaseerd op zeshoekige meshes.

Volledig interpolerende schema's voor subdivisie-oppervlakken, hebben wel als voordeel dat ze een oppervlak kunnen construeren dat precies doorheen een verzameling punten gaat, maar ze hebben ook een aantal inherente nadelen. Het voornaamste probleem is dat ze moeilijk te controleren bulten en plooien vormen. Een kleine verschuiving van een controlepunt geeft plotseling een heel ander oppervlak. Dat maakt dit soort schema's ongeschikt voor modelerings- en animatietoepassingen. Ook hebben volledig interpolerende schema's minder interessante wiskundige eigenschappen. Zo bestaan er bv. geen wiskundige basisfuncties die het oppervlak kunnen beschrijven en beïnvloeden de controlepunten een grotere zone. Ook blijft bij een approximatief schema het oppervlak dat door enkele controlepunten gedefinieerd wordt, volledig in de convexe omhullende van die punten, een eigenschap die interpolerende schema's moeten missen.

Aangezien enerzijds de volledige interpolerende schema's niet interessant zijn voor modeleringstoepassingen, maar er anderzijds toch dikwijls behoefte is om een bepaald controlepunt te interpoleren, ontwikkelden wij een nieuwe techniek. Deze techniek laat toe om, met behulp van nauwkeurig berekende hulppunten ("ghost points"), toch een bepaald controlepunt te interpoleren. Onze techniek heeft als bijkomend voordeel dat ook de oppervlaktenormale en een handige tensie-parameter eenvoudig mee gecontroleerd kunnen worden in de modeleringstoepassing.

We ontwikkelden deze methode eerst voor subdivisie-curves, en breidden dat daarna uit voor de twee meest gebruikte schema's voor subdivisie-oppervlakken, namelijk Catmull-Clark en Loop. We pasten deze technieken bovendien toe buiten het domein van modelering, namelijk in dat van het vervormen van tweedimensionale objecten ("free-form deformation"). We ontwierpen een applicatie die toelaat om op een zeer intuïtieve manier tweedimensionale animaties te creëren met een tot dan toe onbestaande vrijheid.

Een beetje losstaand van de technieken rond locale interpolatie, werkten we ook een volledig nieuw schema voor subdivisie-oppervlakken uit, ditmaal gebaseerd op zeshoekige meshes. Aangezien zeshoekige meshes in de praktijk niet zo frequent voorkomen in grafische computertoepassingen, werkten we ook methodes uit om driehoekige meshes te converteren naar zeshoekige.

Onze interesse voor zeshoekige schema's werd vooral gewekt door twee recente wetenschappelijke publicaties. Zowel een publicatie van Jos Stam [Stam01] als van Zorin en Schröder [Zorin01a] beschreven een techniek waardoor ze oppervlakken van willekeurig hoge graad van continuïteit konden bekomen, door repetitief uit te middelen tussen twee duale subdivisie-schema's. Beide publicaties vermeldden enkel het bestaan van duale schema's voor vierhoekige meshes. Een duaal schema voor een op driehoekige meshes gebaseerd schema, zou met zeshoekige meshes moeten werken, en die waren totnogtoe niet beschreven.

Ons onderzoek omtrent het nieuwe schema voor zeshoekige meshes toont aan dat het een volwaardig alternatief is voor de bestaande schema's en opent een fascinerende wereld van nieuwe mogelijkheden en toepassingen.

De verschillende hoofdstukken van deze doctoraatsverhandeling worden hierna kort ingeleid.

Hoofdstuk 1 is een algemene introductie van de verschillende onderwerpen in de rest van de verhandeling.

In hoofdstuk 2 beschrijven we het principe van subdivisie-curves. We starten met Chaikin's wegsnijden van hoeken voor kwadratische curves en breiden dat uit naar hogere-graadscurves. Naast deze curves die hun controlepunten benaderen, behandelen we ook een interpolerend schema, en we vergelijken hun eigenschappen. We tonen ook aan hoe eigenanalyse gebruikt kan worden om meer te weten te komen over het limietgedrag van de curve. Bovendien tonen we dat behalve via tweedeling, curves ook via driedeling gedefinieerd kunnen worden. Dit schema komt ons later van pas bij de bestudering van ons nieuwe schema voor zeshoekige meshes.

Hoofdstuk 3 begint met een algemene inleiding omtrent subdivisie-oppervlakken en toont eerst de traditionele classificatie van de verschillende schema's. We merken dat deze veelgebruikte classificatie enkele leemtes vertoont. Enerzijds zijn er recent nieuwe schema's omschreven die niet zomaar in een bestaand hokje te duwen zijn. Belangrijker is echter dat schema's voor vierhoekige meshes ingedeeld zijn in primaire en duale schema's, terwijl er voor driehoekige schema's enkel primaire schema's beschreven zijn. Een schema duaal aan een driehoekig is noodzakelijk gebaseerd op zeshoekige meshes, wat het onderwerp wordt van hoofdstuk 5. Eerst gaan we in hoofdstuk 3 dieper in op het verschil tussen duale en primaire subdivisie-schema's. We sluiten het hoofdstuk af met een uitgebreid overzicht van de verschillende subdivisie-schema's en hun eigenschappen, om deze in hoofstuk 5 te vergelijken met ons nieuwe schema.

Een uitgebreid overzicht van interessante eigenschappen en toepassingen van subdivisie-oppervlakken komt in hoofdstuk 4 aan bod.

Hoofdstuk 5 is volledig aan ons nieuwe schema voor zeshoekige meshes gewijd. We gaan eerst na hoe subdivisie voor een reguliere zeshoekige mesh gedefinieerd zou kunnen worden en ontdekken dat er oneindig veel configuraties denkbaar zijn. Er blijken echter slechts twee configuraties te zijn die praktisch interessant lijken voor subdivisie-oppervlakken. Het verschil tussen de methodes wordt vooral bepaald door de factor waarmee het aantal polygonen per stap gemultipliceerd wordt. Een factor zeven lijkt interessant in andere toepassingsdomeinen, maar mist de symmetrie die nodig is voor subdivisie-oppervlakken. De kleinste factoren zijn drie en vier en het blijkt dat multiplicatiefactoren die geen combinatie zijn van deze twee basisfactoren, allemaal dit gebrek aan symmetrie vertonen. Een schema dat voor driehoekige meshes gedefinieerd werd, blijkt indirect op zeshoekige meshes te werken, en wordt op dit ogenblik bestudeerd door een collega-onderzoeker uit Leuven [Simoe01]. We besluiten om een schema te ontwikkelen gebaseerd op een multiplicatiefactor van drie. Enerzijds is dat de kleinst mogelijke niet-triviale multiplicatiefactor. Anderzijds geeft die aanleiding tot de eenvoudigst mogelijke subdivisie-regels. Bovendien blijkt het schema duaal te zijn aan een onlangs gepubliceerd nieuw schema voor driehoekige meshes, Kobbelt's Sqrt(3) schema [Kobbe00]. Via eigenanalyse worden nieuwe stationaire subdivisie-regels afgeleid en bestuderen we de eigenschappen van het nieuwe schema. Verder tonen we een interessant verband aan met een subdivisie-curve uit hoofdstuk 2.

In hoofdstuk 6 bespreken we een andere belangrijke bijdrage van deze doctoraatsverhandeling, met name een techniek om benaderende subdivisie-curves toch een aantal interactief door de ontwerper aangeduide punten te kunnen interpoleren. Dit doel blijkt bereikbaar via het introduceren van goedgekozen hulppunten. Ook voor hogeregraadscurves kan interpolatie bereikt worden, mits het introduceren van meer hulppunten. Deze hulppunten bieden bovendien de mogelijkheid om de raaklijn aan de curve te manipuleren. Ook kan een tensie-parameter gradueel ingesteld worden die de vorm in de omgeving van het interpolerende punt beïnvloedt. De correctheid van de gebruikte technieken wordt bewezen in twee appendices op het einde van de verhandeling.

Hoofdstuk 7 is een algemene inleiding op de drie daaropvolgende hoofdstukken. De noodzaak voor locale interpolatie voor subdivisie-oppervlakken wordt uit de doeken gedaan, en de bestaande literatuur wordt doorgenomen.

In hoofdstuk 8 is het Catmull-Clark schema aan de beurt. Er wordt nagegaan hoe de hulppunten het best geconfigureerd worden om bestaande punten te kunnen interpoleren. Analoog als in hoofdstuk 6, kan ook de oppervlaktenormale vrij gemanipuleerd worden. Ook is er een nieuwe tensie-parameter, die de vorm van het oppervlak in de omgeving van het interpolerende punt beïnvloedt. Hoofdstuk 8 biedt ook een praktisch algoritme voor het berekenen van de meest geschikte posities van de hulppunten.

Daarna worden, in hoofdstuk 9, de specifieke mogelijkheden voor het op driehoeken gebaseerde Loop-schema bestudeerd. Dit geeft aanleiding tot een aangepast algoritme dat de hulppunten zorgvuldig positioneert. Verder wordt nagegaan hoe met behulp van de technieken uit hoofdstuk 6 ook de punten op de rand van het oppervlak gemanipuleerd kunnen worden.

In hoofdstuk 10 wordt een nieuwe vervormingstechniek ("free-form deformation") voor tweedimensionale geanimeerde objecten uitgewerkt. Het blijkt dat door de techniek te baseren op subdivisie-oppervlakken in twee dimensies, er een interessante vervormingstechniek ontstaat, die specifieke nieuwe mogelijkheden biedt. Het grote voordeel van subdivisie is dat willekeurige topologieën en expliciete discontinuïteiten deel uitmaken van het basisconcept.

Hoofdstuk 11 legt uit hoe dit onderzoek in de toekomst verder uitgebreid kan worden. Een aantal van de ideeën wordt nu reeds door onze onderzoeksgroep dieper bestudeerd.

Om af te sluiten, formuleert hoofdstuk 12 de algemene conclusies van deze verhandeling.