EUROGRAPHICS 2001 / A. Chalmers and T.-M. Rhyne
(Guest Editors)

Volume 20 (2001), Number 3

A Unified Subdivision Scheme for Polygonal Modeling

Jérôme Maillot
Alias|Wavefront
210 King St. E.
Toronto, Ontario, Canada M5A 1J7
jmaillot@aw.sgi.com

Jos Stam
Alias|Wavefront
1218 Third Ave, Suite 800
Seattle, WA 98101
jstam@aw.sgi.com

Abstract

Subdivision rules have traditionally been designed to generate smooth surfaces from polygonal meshes. In this paper we propose to employ subdivision rules as a polygonal modeling tool, specifically to add additional level of detail to meshes. However, existing subdivision schemes have several undesirable properties making them ill suited for polygonal modeling. In this paper we propose a general set of subdivision rules which provides users with more control over the subdivision process. Most existing subdivision schemes are special cases. In particular, we provide subdivision rules which blend approximating spline based schemes with interpolatory ones. Also, we generalize subdivision to allow any number of refinements to be performed in a single step.

1. Introduction

Subdivision surfaces have recently emerged as the most popular modeling tool in computer graphics. This is not surprising since these surfaces combine the benefits of both polygonal and spline NURBS modeling. Subdivision surfaces, like NURBS, allow users to model smooth surfaces by manipulating a small set of control vertices. Unlike NURBS, however, there is no constraint on the connectivity of the control vertices.

The first subdivision schemes were proposed in the late seventies [1, 3], while later research has focused generally on the properties of the limit surface, such as smoothness [14, 19] and evaluation [16]. Properties of subdivision surfaces are now well understood, making them attractive in design applications. But, subdivision is rarely used as a polygonal modeling tool. Very little attention has been devoted to the influence of the first couple of subdivision steps on the final shape of the surface. However, in practice it turns out that the initial subdivision steps greatly influence the shape of the surface. In this paper we address this problem by adding more parameters to the subdivision rules thus providing more user control. This is in contrast with previous research where the shape of the surface was improved by changing the initial control mesh, not the subdivision rules themselves [6].

An important application of our new subdivision rules is the generation of level of detail on arbitrary meshes. We will demonstrate that our schemes can efficiently compress such models for transfers over small bandwidth networks.

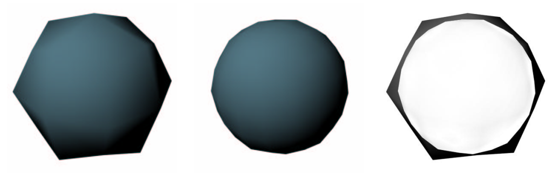


Fig. 1-a Fig. 1-b Fig. 1-c

Figure 1: *Each refinement step of the Catmull-Clark algorithm shrinks the convex areas. The difference between two levels is strongest along the silhouette of the models.*

2. Motivation

Current subdivision rules applied to the problem of generating level of detail suffer from two major limitations. Firstly, the most popular spline-based schemes [1, 3, 13] produce surfaces that approximate the base mesh. The limit surface, especially in locally convex areas, is smaller than the base mesh. This is because the refined control meshes progressively shrink towards the limit surface. Consequently, noticeable "popping effects" occur when switching between meshes at different levels of resolution.

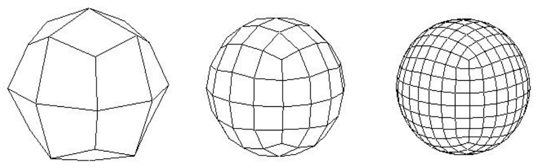


Figure 2: *Each Catmull-Clark refinement increases the number of faces by four, thus making few levels of detail available in practice.*

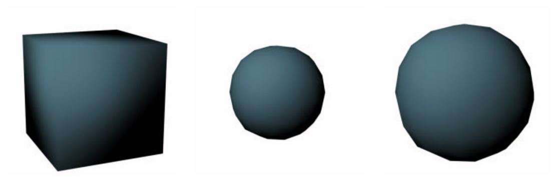
Figure 1 shows one (1-a) and two (1-b) Catmull-Clark refinements applied to the same cube. Figure 1-c shows the difference between these two meshes. Dark areas correspond to larger differences and are mainly concentrated near the silhouette of the mesh. The difference is even larger if we compare the initial cube with the first subdivision level (3-d). Although the differences are less important between finer levels of subdivision, they remain substantial for the first few levels.

Interpolatory schemes have also been proposed [5, 9, 11], but they suffer from the opposite problem. Limit surfaces tend to bulge out of the polygonal control mesh, and successive subdivision steps converge to a surface that is too big. They also introduce undesirable undulations in the refined meshes.

Another problem with current subdivision schemes is that the number of refined faces grows exponentially with the number of subdivisions. In many applications, we want to increase the number of faces by an arbitrary number instead of some power of four (Catmull-Clark) as shown in Figure 2. This problem was partially addressed with the introduction of $\sqrt{3}$-subdivision schemes [10, 11], where the number of triangles increases only by a factor of three at each subdivision step. However, in these schemes the number of triangles still increases exponentially by a power of three and they do not seem to extend to quadrilaterals or schemes of higher degree. Finally, these schemes do not preserve the initial triangle boundaries, which leads to a sudden change in color or texture interpolation between two consecutive subdivision levels. In practice, this means that even a simple object can only be refined three or four times before the level of detail database becomes too large. It turns out that this constraint on the number of refinements is generally too small to efficiently control the geometry's resolution in a real time rendering application.

3. From Approximation to Interpolation

In order to address the silhouette problem, we propose a single parameterized scheme which is a blend of an approximating and an interpolating scheme. We will show that these *intermediate* rules do a better job at preserving the silhouette and the volume of the meshes.



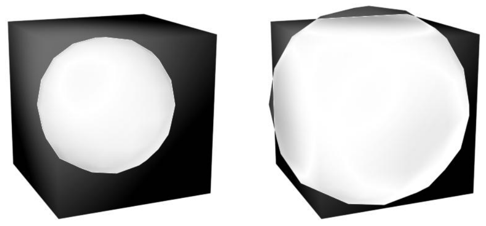


Figure 3: *Silhouette improvement using our new scheme. Figure 3-b was built using Catmull-Clark algorithm, while 3-b uses our method. The bottom row show the image difference is both cases.*

Our work is based on subdivision rules built as a succession of a linear subdivide operator followed by a number of *smoothing* steps [17]. This framework is general enough to work for both triangular and quadrilateral meshes, and produces uniform B-splines of any odd degree p on the regular part of the mesh. We introduce an extra vertex moving step on the subdivided geometry to handle interpolation.

3.1. The Curve Case

As is usually the case in subdivision papers, we will first describe our schemes in a curve setting and then generalize them to surfaces.

3.1.1. Subdivision Into any Number of Pieces

Stam [17] showed that B-splines of odd degree p could be subdivided by first linearly subdividing the control mesh and then performing $m = \frac{p-1}{2}$ smoothing steps. Each step involves averaging a vertex with its immediate neighbors using the $[\frac{1}{4}\,\frac{1}{2}\,\frac{1}{4}]$ weights. The novelty over the well known Lane-Riesenfeld algorithm [12] is that this scheme performs two averaging steps at once and therefore leaves the control vertices "in place."

Stam's result relies on the fact that the subdivision masks are related to the binomial coefficients [2]. The binomial coefficients are easily computed using Pascal's triangle:

$$\begin{array}{ccccccccccc} \frac{1}{1} & \times & & & & 1 & & 1 & & & \\ \frac{1}{2} & \times & & & 1 & & 2 & & 1 & & \\ \frac{1}{4} & \times & & 1 & & 3 & & 3 & & 1 & \\ \frac{1}{8} & \times & 1 & & 4 & & 6 & & 4 & & 1 \end{array}$$

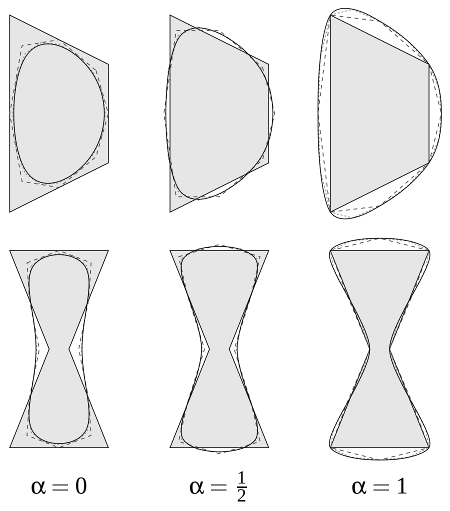


Figure 4: *Refinement steps 1, 2, and limit curve of our scheme, using degree three interpolation and different α values. A division value $d = 2$ was used for each step.*

For example, from the last line we obtain the subdivision masks for B-spline curves of degree three. Every old vertex is updated using the $[\frac{1}{8}\frac{6}{8}\frac{1}{8}]$ weights, while the new vertices are inserted between the old vertices using the $[\frac{4}{8}\frac{4}{8}]$ masks. The crucial observation is that these two masks are obtained by simply applying the $[\frac{1}{4}\frac{1}{2}\frac{1}{4}]$ mask to the second row, which corresponds to the subdivision rules of linear subdivision. In fact, this construction is easily generalized to any number of subdivisions d. In this case we generalize the Pascal triangle to take the average of the d elements in the row directly above it:

$$\begin{array}{ccc} 1 & 1 & 1 \\ 1\ 1 & 1\ 1\ 1 & 1 \ldots 1 \\ 1\ 2\ 1 & 1\ 2\ 3\ 2\ 1 & 1\ 2 \ldots d \ldots 2\ 1 \end{array}$$

This gives us a recipe similar to the one found in Stam [17] to compute the corresponding masks for these subdivision schemes. First linearly subdivide each segment into d pieces, then smooth each vertex using the mask $\frac{1}{d^2}[1, 2, \ldots, d, \ldots, 2, 1]$. In the limit this process generates B-spline curves of degree $2m+1$ if the smoothing is applied m times. See Chui's monograph [2] for a rigorous proof.

3.1.2. Blending Interpolating and Approximating Schemes

In order to limit the amount of shrinking characteristic of approximating subdivision schemes, we propose to add an extra step which updates the position of the vertices after smoothing. We call this a *push-back* step: each original vertex is moved back towards its original position by an amount controlled by the user. Newly introduced vertices are also adjusted by linear interpolation of the adjusted original vertices. We denote by P_j the new vertices obtained by subdividing the original vertices P'_i. In the rest of the paper, ":=" denotes assignment, while "=" denotes a true equality of two quantities. In these notations, the first step is:

$$P_{di} := P'_i \quad (1)$$

$$P_{di+k} := \frac{d-k}{d}P'_i + \frac{k}{d}P'_{i+1}, \quad 0 < k < d \quad (2)$$

This step is followed by a smoothing step that modifies the vertices P_i:

$$P_i := \frac{1}{d^2}\left(dP_i + \sum_{k=1}^{k<d}(d-k)(P_{i-k}+P_{i+k})\right) \quad (3)$$

Finally, the smoothing step is followed by a push-back of these new vertices:

$$\Delta_i := \alpha(P'_i - P_{di}) \quad (4)$$

$$P_{di} := P_{di} + \Delta_i \quad (5)$$

$$P_{di+k} := P_{di+k} + \frac{d-k}{d}\Delta_i + \frac{k}{d}\Delta_{i+1} \quad 0 < k < d \quad (6)$$

All evaluations are done in parallel in a "Jacobi manner" to avoid any side effects. In practice this requires the use of an intermediate array to store the vertices' positions. The *volume parameter* α controls the transition from approximation to interpolation. When $\alpha = 0$ there is no push-back step and the subdivision scheme produces uniform B-splines in the limit. On the other hand, when $\alpha = 1$ our schemes are interpolatory. Figure 4 shows the influence of the parameter α on the subdivided control vertices after several refinements.

When the degree p is greater than three, the smoothing and push-back steps are repeated $\frac{p-1}{2}$ times. In particular, when $p = 3$ and $d = 2$, only one smoothing and one push-back step are performed. In this case we can explicitly write down the subdivision matrix applied to five consecutive control vertices:

$$M = \begin{pmatrix} 2-2\alpha & 4(3+\alpha) & 2-2\alpha & 0 & 0 \\ -\alpha & 8+\alpha & 8+\alpha & -\alpha & 0 \\ 0 & 2-2\alpha & 4(3+\alpha) & 2-2\alpha & 0 \\ 0 & -\alpha & 8+\alpha & 8+\alpha & -\alpha \\ 0 & 0 & 2-2\alpha & 4(3+\alpha) & 2-2\alpha \end{pmatrix}$$

$$\begin{pmatrix} P_{2i-2} \\ P_{2i-1} \\ P_{2i} \\ P_{2i+1} \\ P_{2i+2} \end{pmatrix} = \frac{1}{16}M. \begin{pmatrix} P'_{i-2} \\ P'_{i-1} \\ P'_i \\ P'_{i+1} \\ P'_{i+2} \end{pmatrix} \quad (7)$$

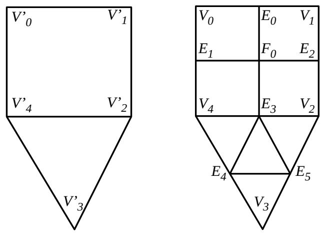


Fig. 5-a Fig. 5-b

Figure 5: *Subdivision of quadrilaterals and triangles. Each subdivision produces V, E, and F vertex types. Faces with 5 and more vertices use the quadrilateral subdivision rule.*

In particular, when $\alpha = 1$, the P_{2i} are moved back exactly to their original position P_i', and we obtain the well known four point interpolation scheme, with $[\frac{-1}{16}\,\frac{9}{16}\,\frac{9}{16}\,\frac{-1}{16}]$ weights [4].

3.2. Generalization to Surfaces

The surface case is similar to the curve one: we perform one bilinear subdivision step followed by a smoothing step.

3.2.1. Binary Subdivisions

We first define our rules for binary subdivision schemes when $d = 2$. We introduce the following notations. The number of elements in a set A is denoted by $A^\sharp$. The vertices of the mesh before a subdivision step are denoted by V_i'. During a subdivision step these vertices are transformed into new vertices V_i. At the same time new vertices E_i are introduced by splitting each edge, and new vertices F_i are introduced for each face as in Figure 5. Let P be a vertex of the mesh, then $\mathcal{E}(P)$ is the set containing all the vertices sharing an edge with P. The set $\mathcal{C}(P)$ contains the "corner vertices:" the vertices sharing a face with P not in $\mathcal{E}(P)$. To illustrate these definitions refer to Figure 5 where $\mathcal{E}(V_2) = \{E_2, E_3, E_5\}$, and $\mathcal{C}(E_3) = \{E_1, E_2\}$.

In the rest of this paper we will focus entirely on quadrilateral schemes. However, triangular schemes can be treated in a similar way, with the exception that there are no *face* vertices F_i, and $\mathcal{C}(V)$ is always empty. In fact, our images were produced from meshes containing simultaneously triangles and quadrilaterals. Only the subdivision step must distinguish between those two types of faces.

Stam provides different smoothing rules for the vertices that result in uniform B-spline surfaces in the limit on the regular part of the mesh [17]. The simplest smoothing algorithm which corresponds to "repeated averaging" [20] replaces each vertex by a weighted average of its direct neighbors:

$$N_i \;=\; \mathcal{E}(V_i)^\sharp \tag{8}$$

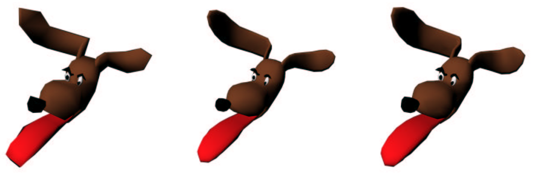


Fig. 6-a Fig. 6-b. $\alpha = 0$ Fig. 6-c. $\alpha = .7$

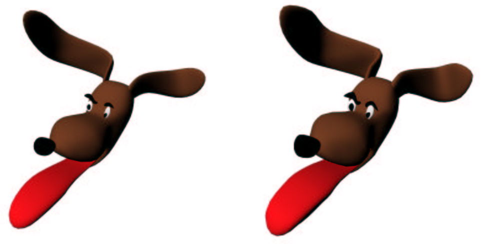


Fig. 6-d. $\alpha = 0$ Fig. 6-e. $\alpha = .7$

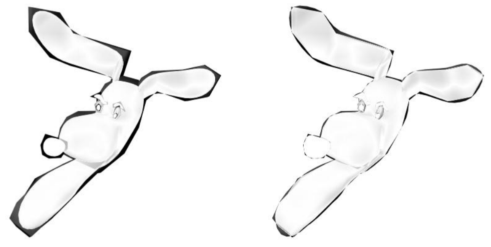


Fig. 6-f Fig. 6-g

Figure 6: *Influence of parameter α on silhouette changes.*

$$V_i \;:=\; \frac{1}{4}V_i + \frac{1}{2N_i}\sum_{P\in\mathcal{E}(V_i)} P + \frac{1}{4N_i}\sum_{P\in\mathcal{C}(V_i)} P \tag{9}$$

Catmull-Clark surfaces are obtained with a different choice for the weights:

$$V_i \;:=\; \frac{N_i-3}{N_i}V_i + \frac{2}{N_i^2}\sum_{P\in\mathcal{E}(V_i)} P + \frac{1}{N_i^2}\sum_{P\in\mathcal{C}(V_i)} P \tag{10}$$

We observe that Formulae 9 and 10 are identical when $N_i = 4$. This comes as no surprise since both of these schemes produce uniform B-spline surfaces on regular meshes ($N_i = 4$ everywhere). We further observe that, when $N_i \neq 4$, the Catmull-Clark rule can be obtained by following Formula 9 with an adjustment of all the extraordinary vertices:

$$\delta_i \;=\; \left(\frac{4}{N_i} - 1\right)V_i + \frac{N_i-4}{N_i}V_i' \tag{11}$$

$$\;=\; \frac{N_i-4}{N_i}(V_i' - V_i) \tag{12}$$

$$V_i \;:=\; V_i + \gamma\delta_i \tag{13}$$

The parameter γ allows us to interpolate between the two

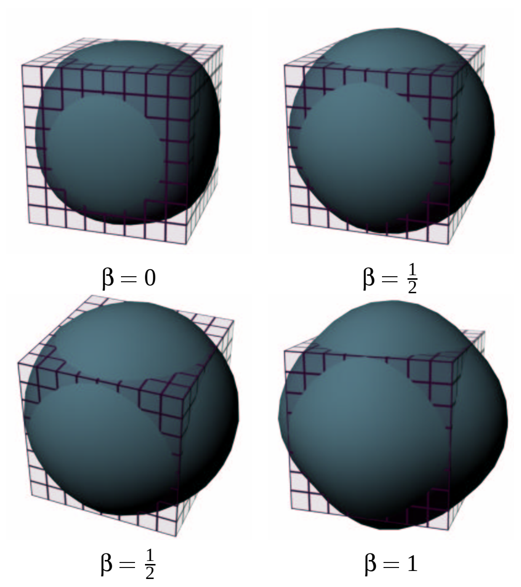


Figure 7: *Influence of β parameter. The original cube is displayed as wireframe.*

schemes. Not only does this adjustment unify these two schemes, but it simplifies the implementation of the Catmull-Clark subdivision: a simple smoothing followed by a vertex update step.

Following the curve case, the simplest push-back step is to compute the differences Δ_i between V_i' and V_i, followed by a (bi-)linear interpolation of these differences for the new E_i and F_i vertices.

$$\Delta(V_i) = \alpha(V_i' - V_i) \tag{14}$$

$$\Delta(E_i) = \frac{1}{2}(\Delta(V_0) + \Delta(V_1)) \tag{15}$$

$$\Delta(F_i) = \frac{1}{\mathcal{E}(F_i)^\sharp} \sum_{E_k \in \mathcal{E}(F_i)} \Delta(E_k) \tag{16}$$

$$= \frac{1}{\mathcal{C}(F_i)^\sharp} \sum_{V_k \in \mathcal{C}(F_i)} \Delta(V_k) \tag{17}$$

In Figure 6 we demonstrate how the α parameter improves the silhouette difference between a polygonal object and its refinements. Figure 6-a is the original model, while Figures 6-b through 6-e illustrate the first and second refinements for $\alpha = 0$ (Catmull-Clark) and $\alpha = .7$. Figure 6-f (resp. 6-g) is the difference between 6-a and 6-d (resp. 6-e). During an animation, the popping effect can be substantially reduced by choosing an appropriate α value.

However, close to very sharp corners our scheme tends to create flat areas around the face centers. The reason is that a bilinear interpolation of vectors of the same length with

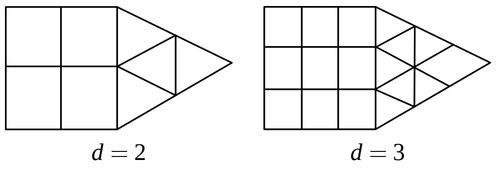


Figure 8: *Each face is divided in $d \times d$ smaller one.*

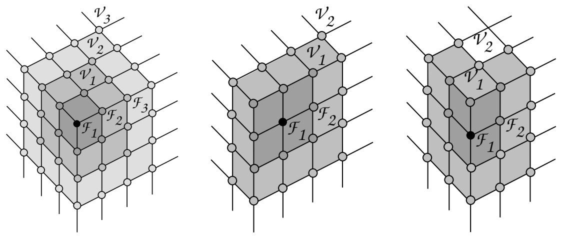


Figure 9: *Vertex [face] neighborhoods $\mathcal{V}_k$ [$\mathcal{F}_k$] are defined by adding one more ring to the previous set.*

different angles produces smaller vectors at the center of the faces. This is a well know artifact of certain renderers, which do not renormalize vertex normals after interpolation, and consequently produce darker areas in the face centers.

We can fix this problem by introducing a renormalization step for the interpolation of the Δ vectors. This is achieved by interpolating the length and direction of the Δ vectors separately. To smooth the transition between these new rules and the ones without the normalization, we introduce a *rounding factor* parameter β.

$$l(E_i) = \beta \frac{||\Delta(V_0)|| + ||\Delta(V_1)||}{2||\Delta(E_i)||} + 1 - \beta \tag{18}$$

$$l(F_i) = \beta \frac{\sum_{V_k \in \mathcal{E}(F_i)} ||\Delta(V_k)||}{\mathcal{E}(F_i)^\sharp ||\Delta(F_i)||} + 1 - \beta \tag{19}$$

$$\Delta(E_i) := l(E_i)\Delta(E_i) \tag{20}$$

$$\Delta(F_i) := l(F_i)\Delta(F_i) \tag{21}$$

When $\beta = 0$ there is no renormalization, while when $\beta = 1$ the lengths of the Δ are exactly interpolated. In the case $\beta \neq 0$ our subdivision rules do not reproduce uniform B-splines on the regular part of the mesh in the limit. This doesn't matter since we do not use our rules to generate limit surfaces. Figure 7 illustrates how our meshes are deformed when β is increased. For a cube, $\beta = \frac{1}{2}$ produces the most "rounded" meshes. Note that in this example we have $\gamma = 0$ (no Catmull-Clark) to emphasize the flattening problem.

3.2.2. Subdivision Into any Number of Pieces

For regular meshes the corresponding limit surfaces $\Sigma(s,t)$ are equal to a tensor product of uniform B-spline curves. Therefore, the subdivision scheme for these surfaces is simply a linear subdivision step followed by a smoothing step with a mask equal to the tensor product of the mask $\frac{1}{d^2}(1,2,...,d,...,2,1)$ derived in Section 3.1.1.

These rules are naturally extended to irregular regions. In practice, it turns out that it is easier to decompose the smoothing step into two simple averaging steps. The averaging step is different depending on whether d is odd or even. In the odd case we replace each vertex by a simple average of its k-ring neighborhood, where $k = \frac{d-1}{2}$. When d is even, each averaging step replaces each face with a vertex that is the average of the k-ring of vertices surrounding it, where $k = \frac{d}{2}$. The new vertices after this step form the dual of the initial mesh. In practice, however, the dual is never explicitly computed since the averaging step is always performed twice (an even amount in general since we consider only odd degrees p in this paper). Indeed, after two dualizations the vertices are again "in place."

More formally, let $\mathcal{V}_k(V_i)$ be the set of all vertices which can be reached from V_i by traversing at most k faces and let $\mathcal{F}_k(V_i)$ denote the corresponding set of the faces traversed. See figure 9 for some examples. We also define a set of face neighborhoods by $\mathcal{V}_k(F_i) = \cup_{V \in F_i} \mathcal{V}_k(V)$.

Using these definitions we can explicitly state the smoothing steps. When d is odd, we apply the following rule $p-1$ times, where $k = \frac{d-1}{2}$:

$$V_i \;:=\; \frac{1}{\mathcal{V}_k(V_i)^\sharp} \sum_{V_j \in \mathcal{V}_k(V_i)} V_j \tag{22}$$

When d is even the procedure only works for odd degrees p. We set the neighborhood to $k = \frac{d}{2}$, and we apply the rule (23) $\frac{p-1}{2}$ times followed by (24):

$$F_i \;:=\; \frac{1}{\mathcal{V}_k(F_i)^\sharp} \sum_{V_j \in \mathcal{V}_k(F_i)} V_j \tag{23}$$

$$V_i \;:=\; \frac{1}{\mathcal{F}_k(V_i)^\sharp} \sum_{F_j \in \mathcal{F}_k(V_i)} F_j \tag{24}$$

In practice, we limited our application to odd degrees only so that no constraint is necessary on the number of subdivisions d.

3.3. Catmull-Clark Correction

The Catmull-Clark correction step defined by Formula 11 was introduced for the case $d = 2$ and is only applied to the extraordinary vertices of the mesh. For arbitrary divisions d we observe that this correction only influences a small neighborhood around each extraordinary vertex. More precisely, this correction never propagates further than two rings of faces around the extraordinary vertex as shown in Figure 10.

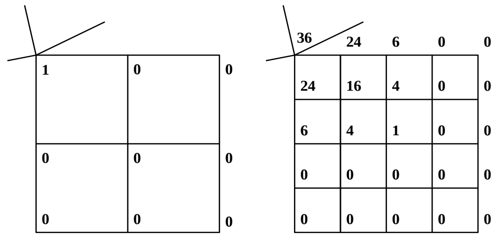


Figure 10: *Catmull-Clark correction weights of the first two steps, along the $\frac{N_i-4}{N_i}(V_i' - V_i)$ vector. Weights in the right image are meant to be divided by 64.*

In addition, the correction is only noticeable in the first couple of subdivision steps. The first subdivision step produces the most visible change which from Formula 11 is equal to $C_i = \frac{N_i-4}{N_i}(V_i' - V_i)$. Subsequent subdivisions produce changes, $w_i C_i$, which are proportional to the first one by a weight w_i. It is possible to compute these weights exactly for the first couple of subdivision steps. These sampled weights then define a piecewise bilinear function on the unit square that can be used to compute the corresponding weight values when d is a power of two. For more general d values the weights can be interpolated from this function.

In practice, however, we found that a similar behavior can be achieved using the push-back step described in the next section. The effect of the Catmull-Clark correction can be emulated by using a higher α value and by adjusting the β parameter. This is apparent in Figure 7, where a value $\beta = \frac{1}{2}$ produces a "rounded" spherical shape despite the fact that $\gamma = 0$.

3.4. Push-back Step

The push-back is similar to the $d = 2$ case described above: we first compute the Δ values for the original vertices and then update the newly introduced vertices using bilinear interpolation. In a similar fashion we can use the normalized interpolation of the Δ values to keep the lengths equal.

For even d, the push-back step can only be applied after Rules (23) and (24) have been applied. This is because it doesn't make any sense to apply the push-back to the "intermediate" vertices F_i which are only used temporarily to compute the new vertex positions. To make our algorithm consistent for every number of divisions d, we restrict our algorithm to only perform the push-back for odd d when Rule (22) is applied twice.

We first intended to use a smoother interpolation of the Δ values, but after some experimentation with higher order interpolation schemes we concluded that the differences were too small to justify a more expensive interpolant.

4. Application to Level of Detail

The motivation behind our work was to provide users with a simple smoothing tool for polygonal meshes. The smoothing operation allows users to create refined versions of their models. Crucial to the success of such a model is that the transitions between the different resolutions of the meshes are almost imperceptible.

In practice, we found that our new subdivision scheme worked best when we used a push-back step with $\alpha = \frac{1}{2}$, $\beta = \frac{1}{2}$ and $\gamma = 0$. Indeed, these are the default values of our smoothing tool. Of course, we expose these parameters to the artist who can freely explore the effect of varying the parameters to meet her particular needs. Although this might be tricky, it is a huge improvement over current practice, where artists sometimes have to adjust individual vertices at each level of refinement. With our model, on the other hand, artists only have to worry about a few parameters at each level.

Sending the different levels of detail of the mesh to a remote viewer is also much more efficient with our representation. Instead of sending the coarse mesh and the updates for each level of resolution as in the progressive meshes compression scheme [8], we only require a few numerical parameters to be sent for each level. Consequently, remote viewers can almost instantly view any level of the mesh refinement. Of course our scheme is not as general as the progressive meshes compression scheme, but we found that in practice it applies to many interesting shapes.

Even when the higher resolution meshes are supplemented with a set of detail offsets for each vertex we believe that our approach offers a better starting point for the base meshes and results in substantial savings in both memory and speed.

5. Results

The images in Figure 11 show meshes of a game character at different levels of detail. The first image shows the base mesh. Subsequent images are created using our new subdivision scheme with different resolutions, ranging from $d = 2$ to $d = 7$. In each case we set our parameters to $\alpha = .5, \beta = 0, \gamma = 0, p = 3$. Notice that there is very little difference in the silhouette between the levels. Applying standard approximating subdivision schemes to the base mesh would have resulted in more shrinking and restricted us to only three different levels due to the exponential increase in the number of vertices with each subdivision.

We also performed some animation tests where we interactively switch between different levels of detail depending on the distance of the model to the camera. The dog images in the color plate were extracted from one such test. Compare the sequence computed using the standard Catmull-Clark (left column) to the ones computed using our new scheme (right column).

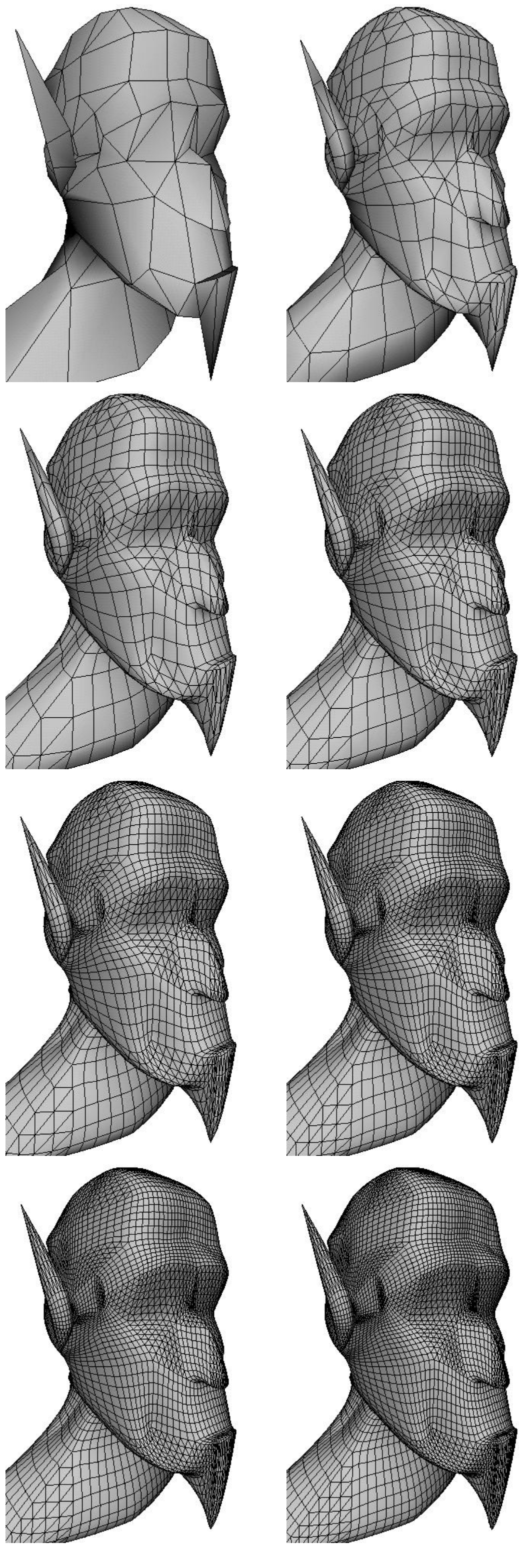


Figure 11: *Eight different resolutions of the same model*

The bottom two images in the Color Plate demonstrate that texture coordinates can be interpolated in the same manner. These models were used in an actual game where low resolution textures are crucial to keep the texture memory consumption as low as possible.

6. Conclusion

In this paper we have presented a novel set of subdivision rules specifically designed for the problem of creating different levels of detail for meshes. Our models depend on a few parameters that an artist can adjust to achieve various effects. Existing approximating and interpolating subdivision schemes are special cases of our new scheme. The advantages of our new scheme over existing ones is that (1) we provide user control over the amount of shrinking and (2) we allow arbitrary refinements which do not grow exponentially. These two features make subdivision schemes more attractive in the realm of polygonal modeling.

Our new rules are quite easy to implement. The entire subdivision system was coded as a MAYA plug-in in only roughly 500 lines of C++ code. We were able to interact in real time with our models on a relatively modest desktop machine (O^2, R5000). Of course, this was true only when our division number d was approximately between 2 and 6.

Our subdivision schemes allow both quadrilaterals and triangles to co-exist in a mesh and its refined versions. This is in contrast to existing subdivision schemes which generate meshes which are either all triangles or all quadrilaterals. Artists often want to keep both triangles and quadrilaterals in their models. Also it is well known that triangles in a base mesh create artifacts in the refined meshes when a quad-based subdivision scheme is used. The reason that we can incorporate both triangles and quadrilaterals simultaneously in our subdivision scheme is that our scheme is based on a decomposition of the rules into a linear step followed by smoothing.

7. Future work

A straightforward extension of our model is to allow our parameters to vary on the mesh. These values could be painted onto the mesh by the artist for additional control, for example. However, we found that our current scheme provided enough flexibility with the α, β and γ parameters fixed for each level. On the other hand, allowing the parameters to vary in very localized regions could still be helpful in certain applications.

In this paper we have ignored the properties of the limit surfaces generated by our schemes. The reason for this omission is that we are interested solely in the shape of a finite number of intermediate meshes, not in the limit surface. On the other hand our new schemes might be useful in surface design as well. In this case it is important to study the smoothness of the limit surface. At this point it is not clear how the smoothness depends on our parameters. We leave this as an open problem for future research.

Also, we did not pay much attention to creases and object boundaries. Boundaries and creases can be dealt with by applying our curve rules to the corresponding polylines. Vertex creases are more difficult to handle. This is because creased vertices are not updated during our smoothing step. Consequently the push-back vertex is always zero, resulting in undesirable wrinkles in the meshes. More work is needed in this case to improve the behavior of the refined shape.

Finally, we found that the optimal set of parameters, especially α, depends on the camera position. This is particularly visible for a simple object like a cube, where larger values of α are required when viewed from the side. One solution would be to allow the user to specify different values of α for different views of an object. Our system would then automatically interpolate α values for intermediate viewing positions as the camera moves.

Acknowledgments

The help and comments of A|W artists have been very useful in order to understand the level of detail problems. We would like to thank particularly C. Chaix, J. Bell, and P. Pattel who built the objects we used.

References

1. **E. Catmull, J. Clark.** *Recursively Generated B-Spline Surfaces On Arbitrary Topological Meshes.* Computer Aided Design, 10(6):350–355, 1978.
2. **C.K. Chui.** *Multivariate Splines.* CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, Philadelphia, PA, 1988.
3. **D. Doo, M. A. Sabin.** *Behaviour Of Recursive Subdivision Surfaces Near Extraordinary Points..* Computer Aided Design, 10(6):356–360, 1978.
4. **N. Dyn, D. Levin, J. Gregory.** *A 4-point interpolatory subdivision scheme for curve design.* CAGD 4, 257–268, 1987.
5. **N. Dyn, D. Levin, J. Gregory.** *A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control.* ACM Trans. Graph. 9, 160–169, 1990.
6. **M. Halstead, M. Kass, T. DeRose.** *Efficient, Fair Interpolation Using Catmull-Clark Surfaces.* Computer Graphics Proceedings, Annual Conference Series, 35–44, August, 1993.
7. **H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, W. Stuetzle.** *Piecewise smooth surface reconstruction.* Computer Graphics Proceedings, Annual Conference Series, 295–302, July, 1994.

8. **H. Hoppe.** *Progressive Meshes.* Computer Graphics Proceedings, Annual Conference Series, 99-108, August, 1996.

9. **L. Kobbelt.** *Interpolatory subdivision on open quadrilateral nets with arbitrary topology.* Computer Graphics Forum 15(3), 409–420, September, 1996.

10. **L. Kobbelt.** $\sqrt{3}$*-Subdivisions.* Computer Graphics Proceedings, Annual Conference Series, pages 103–112, July, 2000.

11. **U. Labsik, G. Greiner.** *Interpolatory* $\sqrt{3}$*-Subdivision.* Proceedings of Eurographics 2000, Computer Graphics Forum, 19(3):131–138, September, 2000.

12. **J.M. Lane, R. F. Riesenfeld.** *A Theoretical Development For the Computer Generation and Display of Piecewise Polynomial Surfaces.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2(1):35–46, January, 1980.

13. **C.T. Loop.** *Smooth Subdivision Surfaces Based on Triangles.* M.S. Thesis, Department of Mathematics, University of Utah, August, 1987.

14. **U. Reif.** *A Unified Approach To Subdivision Algorithms Near Extraordinary Vertices.* Computer Aided Geometric Design, 12:153–174, 1995.

15. **U. Reif.** *A degree estimate for subdivision surfaces of higher regularity.* Proceedings of the American Mathematical Society, 124:2167–2174, 1996.

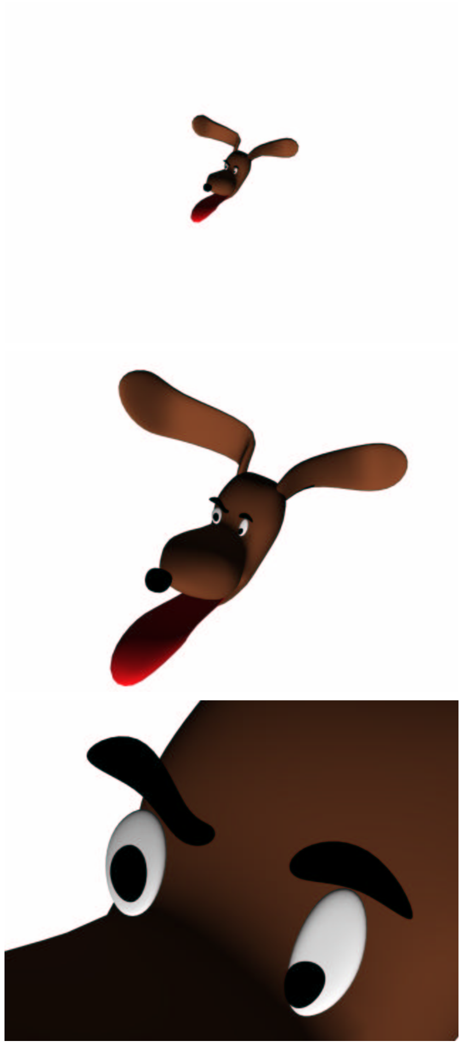
16. **J. Stam.** *Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values.* In Computer Graphics Proceedings, Annual Conference Series, pages 395–404, July, 1998.

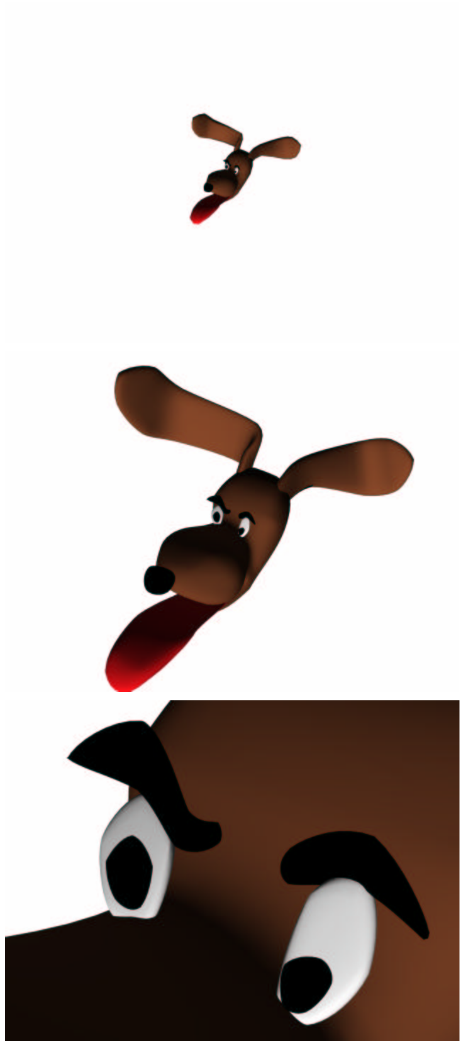
17. **J. Stam.** *On Subdivision Schemes Generalizing Uniform B-Spline Surfaces of Arbitrary Degree.* Computer Aided Geometric Design 18, Special issue on Subdivision Surfaces (to appear), 2001.

18. **H. Weimer, J. Warren.** *Subdivision Schemes for Thin Plate Splines.* Proceedings of Eurographics 1998, Computer Graphics Forum, 17(3):303–313, 1998.

19. **D. Zorin.** *Subdivision and Multiresolution Surface Representations.* PhD thesis, Caltech, Pasadena, California, 1997.

20. **D. Zorin, P. Schröder.** *A Unified Framework for Primal/Dual Quadrilateral Subdivision Schemes.* Computer Aided Geometric Design 18, Special issue on Subdivision Surfaces (to appear), 2001.





Catmull-Clark $\alpha = .5$

coarse mesh $\alpha = .7, d = 3$