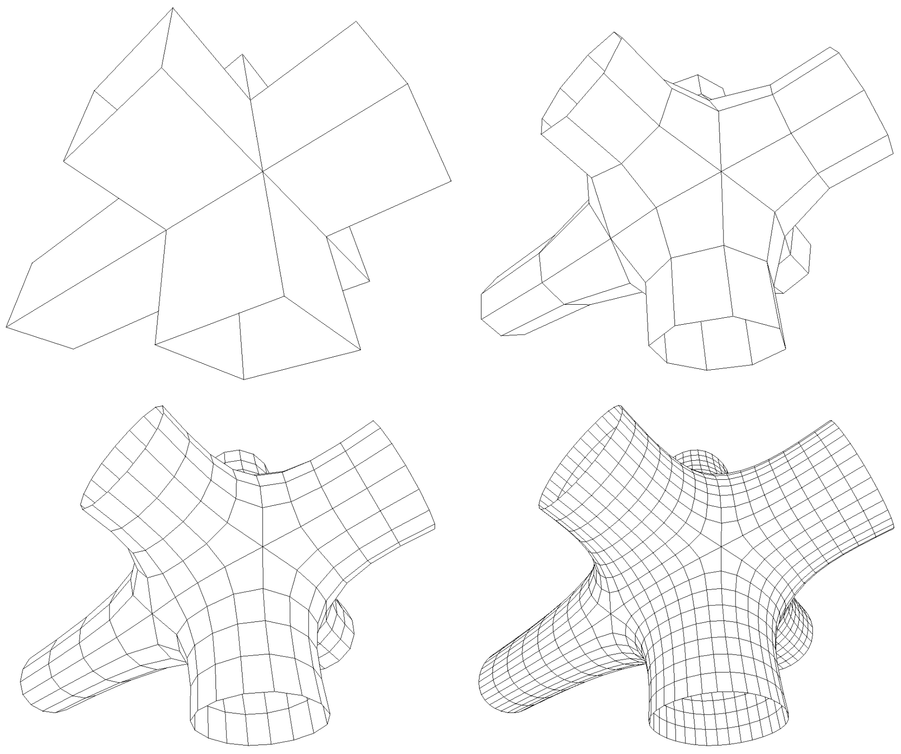


Figure 13.32: The top left image shows the control mesh, i.e., that original mesh, which is the only geometrical data that describes the resulting subdivision surface. The following images are subdivided one, two, and three times. As can be seen, more and more polygons are generated and the surface gets smoother and smoother. The scheme used here is the Catmull-Clark scheme, described in Section 13.5.4.

The topic of subdivision curves has only been touched upon, but it is sufficient for the presentation of subdivision surfaces that follows in the next section. See the Further Reading and Resources section at the end of this chapter for more references and information.

13.5 Subdivision Surfaces

Subdivision surfaces are a powerful paradigm in defining smooth, continuous, crackless surfaces from meshes with arbitrary topology. As with all other surfaces in this chapter, subdivision surfaces also provide infinite level of detail. That is, you can generate as many triangles or polygons as you wish, and the original surface representation is compact. An example of a surface being subdivided is shown in Figure 13.32. Another advantage is that subdivision rules are simple and easily implemented. A disadvantage is that the analysis of surface

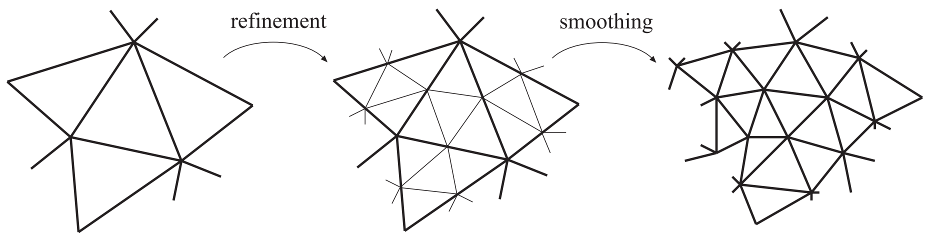


Figure 13.33: Subdivision as refinement and smoothing. The refinement phase creates new vertices and reconnects to create new triangles, and the smoothing phase computes new positions for the vertices.

continuity often is very mathematically involved. However, this sort of analysis is often only of interest to those who wish to create new subdivision schemes, and is out of the scope of this book—consult Warren and Weimer's book [958] and the SIGGRAPH course on subdivision [1024].

In general, the subdivision of surfaces (and curves) can be thought of as a two-phase process [505]. Starting with a polygonal mesh, called the *control mesh*, the first phase, called the *refinement phase*, creates new vertices and reconnects to create new, smaller triangles. The second, called the *smoothing phase*, typically computes new positions for some or all vertices in the mesh. This is illustrated in Figure 13.33. It is the details of these two phases that characterize a subdivision scheme. In the first phase, a polygon can be split in different ways, and in the second phase, the choice of subdivision rules give different characteristics such as the level of continuity, and whether the surface is approximating or interpolating.

A subdivision scheme can be characterized by whether it is *stationary*, whether it is *uniform*, and whether it is *triangle-based* or *polygon-based*. A stationary scheme uses the same subdivision rules at every subdivision step, while a nonstationary may change the rules depending on which step currently is being processed. The schemes treated below are all stationary. A uniform scheme uses the same rules for every vertex or edge, while a nonuniform scheme may use different rules for different vertices or edges. As an example, a different set of rules is often used for edges that are on the boundaries of a surface. A triangle-based scheme only operates on triangles, and thus only generates triangles, while a polygon-based scheme operates on arbitrary polygons. We will mostly present triangle-based schemes here because that is what graphics hardware is targeted for, but we will also briefly cover some well-known polygon-based schemes.

Several different subdivision schemes are presented next. Following these, two techniques are presented that extend the use of subdivision surfaces, along with methods for subdividing normals, texture coordinates, and colors. Finally, some practical algorithms for subdivision and rendering are presented.

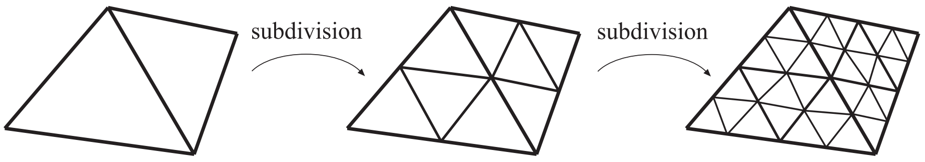


Figure 13.34: The connectivity of two subdivision steps for schemes such as Loop's and the modified butterfly scheme (see Section 13.5.2). Each triangle generates four new triangles.

13.5.1 Loop Subdivision

Loop's subdivision scheme [589][5] was the first subdivision scheme for triangles. It is similar to the last scheme in Section 13.4 in that it is approximating, and that it updates each existing vertex and creates a new vertex for each edge. The connectivity for this scheme is shown in Figure 13.34. As can be seen, each triangle is subdivided into four new triangles, so after n subdivision steps, a triangle has been subdivided into 4^n triangles.

First, let us focus on an existing vertex $\mathbf{p}^k$, where k is the number of subdivision steps. This means that $\mathbf{p}^0$ is the vertex of the control mesh. After one subdivision step, $\mathbf{p}^0$ turns into $\mathbf{p}^1$. In general, $\mathbf{p}^0 \rightarrow \mathbf{p}^1 \rightarrow \mathbf{p}^2 \rightarrow \cdots \rightarrow \mathbf{p}^\infty$, where $\mathbf{p}^\infty$ is the limit point. If the *valence* of $\mathbf{p}^k$ is n, then $\mathbf{p}^k$ has n neighboring vertices, $\mathbf{p}_i^k$, $i \in \{0, 1, \ldots, n-1\}$. See Figure 13.35 for the notation described above. Also, a vertex that has valence 6 is called *regular* or *ordinary*; otherwise it is called *irregular* or *extraordinary*.

Below, the subdivision rules for Loop's scheme are given, where the first formula is the rule for updating an existing vertex $\mathbf{p}^k$ into $\mathbf{p}^{k+1}$, and the second formula is for creating a new vertex, $\mathbf{p}_i^{k+1}$, between $\mathbf{p}^k$ and each of the $\mathbf{p}_i^k$. Again, n is the valence of $\mathbf{p}^k$:

$$\begin{aligned} \mathbf{p}^{k+1} &= (1-n\beta)\mathbf{p}^k + \beta(\mathbf{p}_0^k + \cdots + \mathbf{p}_{n-1}^k), \\ \mathbf{p}_i^{k+1} &= \frac{3\mathbf{p}^k + 3\mathbf{p}_i^k + \mathbf{p}_{i-1}^k + \mathbf{p}_{i+1}^k}{8}, \; i = 0 \ldots n-1. \end{aligned} \tag{13.47}$$

Note that we assume that the indices are computed modulo n, so that if $i = n-1$, then for $i+1$, we use index 0, and likewise when $i = 0$, then for $i-1$, we use index $n-1$. These subdivision rules can easily be visualized as masks, also called stencils; see Figure 13.36. The major use of these is that they communicate almost an entire subdivision scheme using only a simple illustration. Note that the weights sum to one for both masks. This is a characteristic that is true for all subdivision schemes, and the rationale for this is that a new point should lie in the neighborhood of the weighted points. In Equation 13.47, the constant β is actually a function of n, and is given by:

$$\beta(n) = \frac{1}{n}\left(\frac{5}{8} - \frac{(3 + 2\cos(2\pi/n))^2}{64}\right). \tag{13.48}$$

[5]A brief overview of Loop's subdivision scheme is also presented by Hoppe et al. [418].

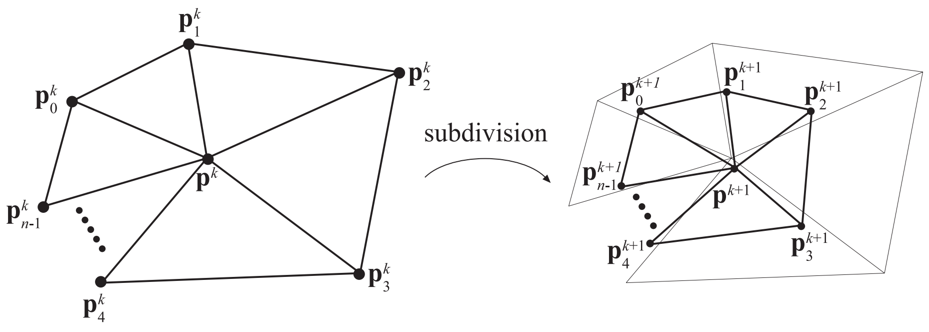


Figure 13.35: The notation used for Loop's subdivision scheme. The left neighborhood is subdivided into the neighborhood to the right. The center point $\mathbf{p}^k$ is updated and replaced by $\mathbf{p}^{k+1}$, and for each edge between $\mathbf{p}^k$ and $\mathbf{p}_i^k$, a new point is created ($\mathbf{p}_i^{k+1}$, $i \in 1, \ldots, n$).

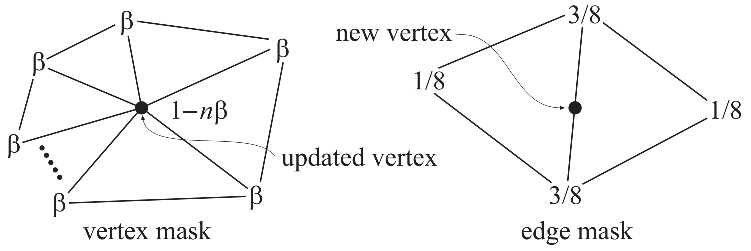


Figure 13.36: The masks for Loop's subdivision scheme (black circles indicate which vertex is updated/generated). A mask shows the weights for each involved vertex. For example, when updating an existing vertex, the weight $1 - n\beta$ is used for the existing vertex, and the weight β is used for all the neighboring vertices, called the 1-ring.

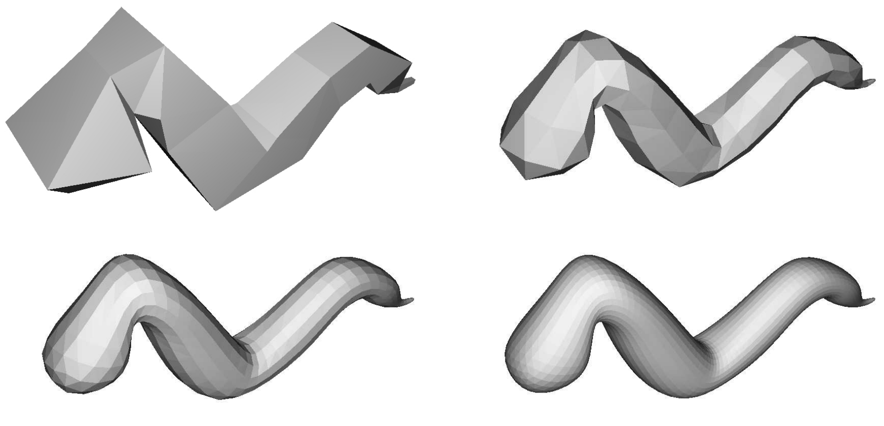


Figure 13.37: A worm subdivided three times with Loop's subdivision scheme.

Loop's suggestion [589] for the β-function gives a surface of C^2 continuity at every regular vertex, and C^1 elsewhere [1022], that is, at all irregular vertices. As only regular vertices are created during subdivision, the surface is only C^1 at the places where we had irregular vertices in the control mesh. See Figure 13.37 for an example of a mesh subdivided with Loop's scheme. A variant of Equation 13.48, which avoids trigonometric functions, is given by Warren and Weimer [958]:

$$\beta(n) = \frac{3}{n(n+2)}. \tag{13.49}$$

For regular valences, this gives a C^2 surface, and C^1 elsewhere. The resulting surface is hard to distinguish from a regular Loop surface. For a mesh that is not closed, we cannot use the presented subdivision rules. Instead, special rules have to be used for such boundaries. For Loop's scheme, the reflection rules of Equation 13.46 can be used. This is also treated in Section 13.5.5.

The surface after infinitely many subdivision steps is called the limit surface. Limit surface points and limit tangents can be computed using closed form expressions. The limit position of a vertex is computed [418, 1024] using the formula on the first row in Equation 13.47, by replacing $\beta(n)$ with:

$$\gamma(n) = \frac{1}{n + \frac{3}{8\beta(n)}}. \tag{13.50}$$

Two limit tangents for a vertex $\mathbf{p}^k$ can be computed by weighting the immediate neighboring vertices, called the *1-ring* or *1-neighborhood*, as shown below [418, 589]:

$$\mathbf{t}_u = \sum_{i=0}^{n-1} \cos(2\pi i/n)\mathbf{p}_i^k, \quad \mathbf{t}_v = \sum_{i=0}^{n-1} \sin(2\pi i/n)\mathbf{p}_i^k. \tag{13.51}$$

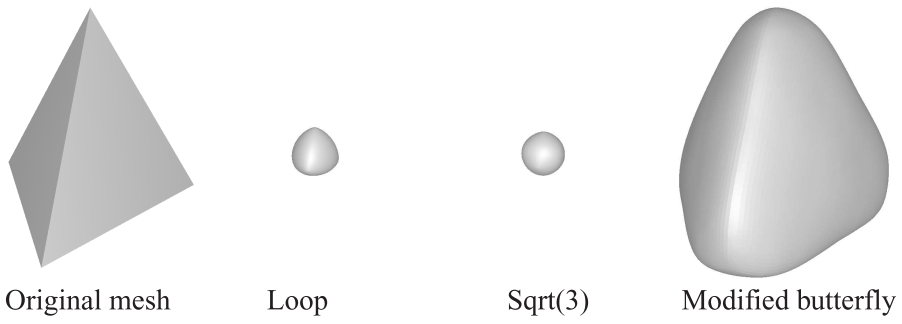


Figure 13.38: A tetrahedron is subdivided five times with Loop's, the $\sqrt{3}$, and the *Modified Butterfly* (MB) scheme. Loop's and the $\sqrt{3}$-scheme are both approximating, while MB is interpolating. Approximating schemes are acting as lowpass filters, which means that shrinking occurs.

The normal is then $\mathbf{n} = \mathbf{t}_u \times \mathbf{t}_v$. Note that this often is less expensive [1024] than the methods described in Section 12.3, which need to compute the normals of the neighboring triangles. More importantly, this gives the exact normal at the point.

A major advantage of approximating subdivision schemes is that the resulting surface tends to get very fair. *Fairness* is, loosely speaking, related to how smoothly a curve or surface bends [674]. A higher degree of fairness implies a smoother curve or surface. Another advantage is that approximating schemes converge faster than interpolating schemes. However, this comes at the cost of the shape being kind of low-pass filtered, which in turn means that the shapes often shrink. This is most notable for small meshes, such as the tetrahedron shown in Figure 13.38. One way to decrease this effect is to use more vertices in the control mesh. i.e., care must be taken while modeling the control mesh. Maillot and Stam present a framework for combining subdivision schemes so that the shrinking can be controlled [607]. A characteristic that can be used to great advantage at times is that a Loop surface is contained inside the convex hull of the original control points [1022].

The Loop subdivision scheme generates a generalized three-directional quartic box spline.[6] So, for a mesh consisting only of regular vertices, we could actually describe the surface as a type of spline surface. However, this description is not possible for irregular settings. Being able to generate smooth surfaces from any mesh of vertices is one of the great strengths of subdivision schemes. See also Sections 13.5.5 and 13.5.6 for different extensions to subdivision surfaces that use Loop's scheme.

[6]These spline surfaces are out of the scope of this book. Consult Warren's book [958], the SIGGRAPH course [1024], or Loop's thesis [589].

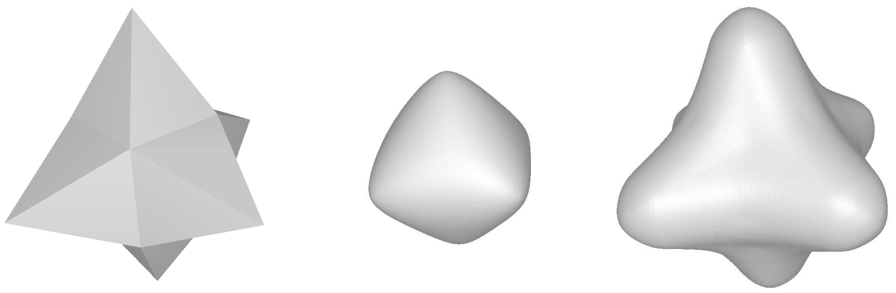


Figure 13.39: To the left is a simple three-dimensional star mesh. The middle image shows the resulting surface using Loop's subdivision scheme, which is approximating. The right image shows the result using the modified butterfly scheme, which is interpolating. An advantage of using interpolating schemes is that they often resemble the control mesh more than approximating schemes do. However, for detailed meshes the difference is not as distinct as shown here.

13.5.2 Modified Butterfly Subdivision

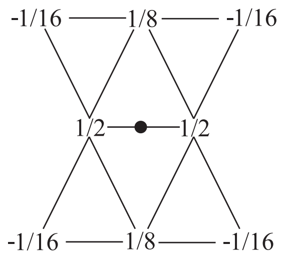
Here we will present the subdivision scheme by Zorin et al. [1019, 1022], which is a modification of the butterfly scheme by Dyn et al. [225], and therefore often referred to as the *Modified Butterfly* (MB) scheme. This scheme is nonuniform, both because it uses different rules at the boundaries, and because different rules are used depending on the valence of the vertices. The main difference from Loop subdivision, however, is that it is interpolating, rather than approximating. An interpolating scheme means that once a vertex exists in the mesh, its location cannot change. Therefore, this scheme never modifies, but only generates new vertices for the edges. See Figure 13.39 for an example between interpolating and approximating schemes. The connectivity is the same as for Loop's scheme, shown in Figure 13.34.

The MB scheme uses four different subdivision rules for creating new vertices between two existing vertices. These are all described below, and the corresponding masks are shown in Figure 13.40.

1. Regular setting: Assume that we want to generate a new vertex between two existing vertices, $\mathbf{v}$ and $\mathbf{w}$, that each has valence 6. These vertices are called regular or ordinary, and we call the situation a *regular setting*. The mask for this situation is shown to the left in Figure 13.40.

2. Semiregular setting: A *semiregular* setting occurs when one vertex is regular ($n = 6$), and another is irregular ($n \neq 6$), also called extraordinary, and we want to generate a new vertex between these vertices. The following formula computes the new vertex, where n is the valence of the irregular vertex:

$$\begin{aligned} n = 3: &\quad w_0 = 5/12, \quad w_1 = -1/12, \quad w_2 = -1/12, \\ n = 4: &\quad w_0 = 3/8, \quad w_1 = 0, \quad w_2 = -1/8, \quad w_3 = 0, \\ n \geq 5: &\quad w_j = \frac{0.25 + \cos(2\pi j/n) + 0.5\cos(4\pi j/n)}{n}. \end{aligned} \tag{13.52}$$



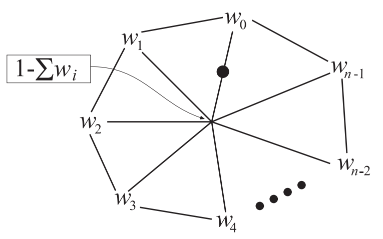


Figure 13.40: The mask to the left is the "butterfly" mask, which is used when generating a new vertex between two regular vertices (those with weights $1/2$). The mask to the right shows the weights when one vertex is irregular (the one with weight $1-\sum w_i$), and one vertex is regular (with weight w_0). Note that black circles indicate which vertex is generated. *(Illustration after Zorin et al. [1024].)*

Note that we used only the immediate neighborhood of the irregular vertex to compute the new vertex, as shown in the mask in Figure 13.40.

3. Irregular setting: When an edge connects two vertices, where both vertices are irregular ($n \neq 6$), we temporarily compute a new vertex for each of these two vertices using the formula for the semiregular setting (2). The average of these two vertices is used as the new vertex. This can happen at only the first subdivision step, because after that there will be only regular and semiregular settings in the mesh. Therefore the continuity of this choice does not affect the limit surface. Zorin et al. [1019] note that this rule generates shapes with better fairness.

4. Boundaries: At boundaries, where an edge in the triangle mesh has only one triangle connected to it, the interpolating scheme [224] with $w = 1/16$, described in Section 13.4, is used. This means that the weights, whose masks are shown in Figure 13.40, are:

$$w_{-1} = -1/16, \quad w_0 = 9/16, \quad w_1 = 9/16, \quad w_2 = -1/16. \tag{13.53}$$

More types of boundary cases exist—consult the SIGGRAPH course for more about this [1024]. Implementation details are discussed by Sharp [841]. Since this scheme is interpolating, limit positions of the vertices are the vertices themselves. Limit tangents are more complex to compute. For extraordinary vertices ($n \neq 6$), the tangents can be calculated using Equation 13.51, that is, the same formulae as for Loop [1022]. For ordinary vertices ($n = 6$), the *2-ring* (also called the *2-neighborhood*) is used. The 1-ring and the 2-ring of an ordinary vertex, $\mathbf{p}$, is shown in Figure 13.41. The tangents vectors, $\mathbf{t}_u$ and $\mathbf{t}_v$, are then computed as:

$$\begin{aligned} \mathbf{t}_u &= \mathbf{u} \cdot \mathbf{r}, \\ \mathbf{t}_v &= \mathbf{v} \cdot \mathbf{r}, \end{aligned} \tag{13.54}$$

where $\mathbf{r}$ is a vector of the difference vectors $\mathbf{p}_i - \mathbf{p}$ of the entire 2-ring, and $\mathbf{u}$

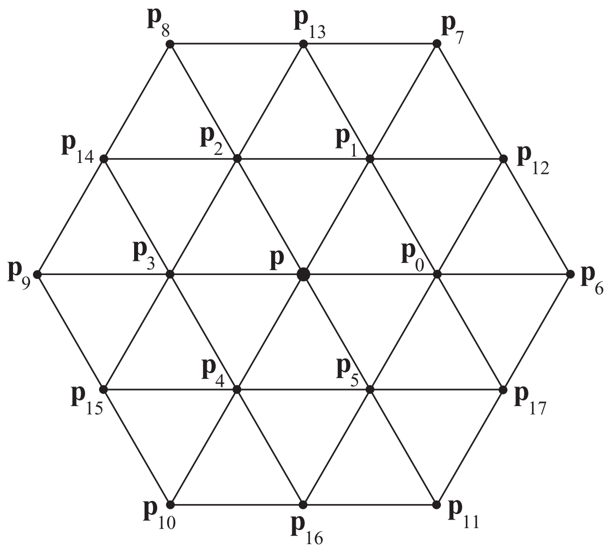


Figure 13.41: An ordinary vertex, $\mathbf{p}$, with its 1-ring $(\mathbf{p}_0, \dots, \mathbf{p}_5)$, and its 2-ring $(\mathbf{p}_6, \dots, \mathbf{p}_{17})$.

and $\mathbf{v}$ are vectors of scalars:

$$\begin{aligned}
\mathbf{r} &= (\mathbf{p}_0 - \mathbf{p}, \mathbf{p}_1 - \mathbf{p}, \mathbf{p}_2 - \mathbf{p}, \dots, \mathbf{p}_{16} - \mathbf{p}, \mathbf{p}_{17} - \mathbf{p}), \\
\mathbf{u} &= (16, -8, -8, 16, -8, -8, -\frac{8}{\sqrt{3}}, \frac{4}{\sqrt{3}}, \frac{4}{\sqrt{3}}, -\frac{8}{\sqrt{3}}, \frac{4}{\sqrt{3}}, \frac{4}{\sqrt{3}}, \\
&\quad 1, -\frac{1}{2}, -\frac{1}{2}, 1, -\frac{1}{2}, -\frac{1}{2}), \\
\mathbf{v} &= (0, 8, -8, 0, 8, -8, 0, -\frac{4}{\sqrt{3}}, \frac{4}{\sqrt{3}}, 0, -\frac{4}{\sqrt{3}}, \frac{4}{\sqrt{3}}, \\
&\quad 0, \frac{1}{2}, -\frac{1}{2}, 0, \frac{1}{2}, -\frac{1}{2}).
\end{aligned} \tag{13.55}$$

This means that, for example, $\mathbf{t}_u$ is computed as:

$$\mathbf{t}_u = 16(\mathbf{p}_0 - \mathbf{p}) - 8(\mathbf{p}_1 - \mathbf{p}) - \cdots - 0.5(\mathbf{p}_{17} - \mathbf{p}). \tag{13.56}$$

After computing both $\mathbf{t}_u$ and $\mathbf{t}_v$, the normal is $\mathbf{n} = \mathbf{t}_u \times \mathbf{t}_v$.

When an interpolating scheme is desired, the MB scheme is a good choice. For example, say you have modeled a human, and decide that you want to subdivide it to get better shading. An interpolating scheme will generate a surface that is more like the control mesh. This is most notable when using meshes with few triangles. For larger meshes, the differences disappear. However, the interpolating characteristics come at the cost that the scheme may generate weird shapes with "unnatural" undulations, and thus, less fairness. This is common for all interpolating schemes. See Figure 13.42 for a nasty example. Another

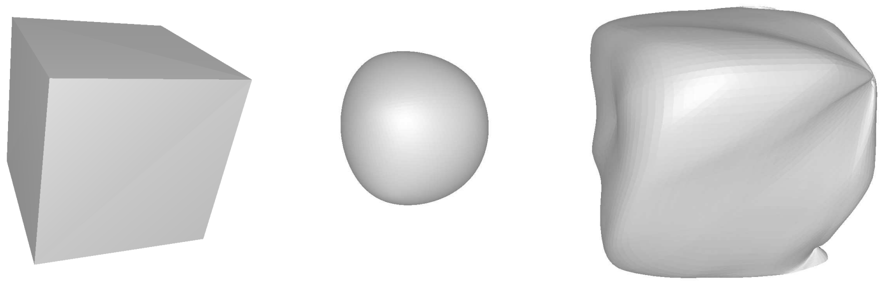


Figure 13.42: The cube on the left is subdivided using Loop's scheme (middle), and the modified butterfly scheme (right). Each face on the cube consists of two triangles. Note the "unnatural" undulations on the right surface. This is because it is much harder to interpolate a given set of vertices.

disadvantage is that the masks are bigger than those used for Loop's scheme and the $\sqrt{3}$-scheme presented in Section 13.5.3, and thus it is more expensive to evaluate.

Despite these disadvantages, interpolating schemes such as MB can be well-suited for real-time rendering work. Meshes for real-time work are normally not finely tessellated, so an interpolated surface is usually more intuitive, as it more closely matches the location of the control mesh. The tradeoff is that fairness problems can occur, but in many cases, minor adjustments to the underlying mesh can smooth out rippling [840]. The MB scheme is C^1-continuous all over the surface, even at irregular vertices [1022]. See Figure 13.38 on page 403, and Figure 13.43 for two examples. More about this scheme can be found in Zorin's Ph.D. thesis [1022] and in Sharp's articles [840, 841].

13.5.3 $\sqrt{3}$-Subdivision

Both Loop's and the MB schemes split each triangle into four new ones, and so create triangles at a rate of $4^n m$, where m is the number of triangles in the control mesh, and n is the number of subdivision steps. A feature of Kobbelt's $\sqrt{3}$-scheme [505] is that it creates only three new triangles per subdivision step.[7] The trick is to create a new vertex (here called *mid-vertex*) in the middle of each triangle, instead of one new vertex per edge. This is shown in Figure 13.44. To get more uniformly shaped triangles, each old edge is flipped so that it connects two neighboring midvertices. In the subsequent subdivision step (and in every second subdivision step thereafter), the shapes of the triangles more resemble the initial triangle configuration due to this edge flip.

The subdivision rules are shown in Equation 13.57, where $\mathbf{p}_m$ denotes the midvertex, computed as the average of the triangle vertices: $\mathbf{p}_a$, $\mathbf{p}_b$, and $\mathbf{p}_c$.

[7]The name stems from the fact that while Loop's and the MB schemes divide each edge into two new edges per subdivision step, Kobbelt's scheme creates three new edges per two subdivision steps. Thus the name $\sqrt{3}$-subdivision.

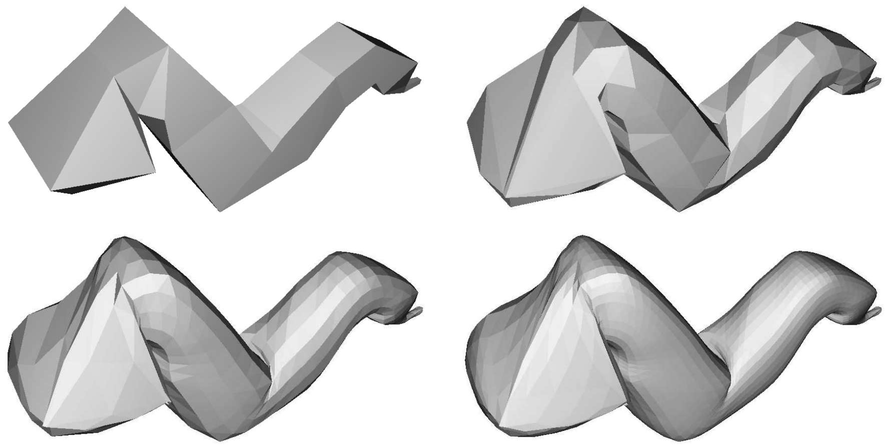


Figure 13.43: A worm is subdivided three times with the modified butterfly scheme. Notice that the vertices are interpolated at each subdivision step.

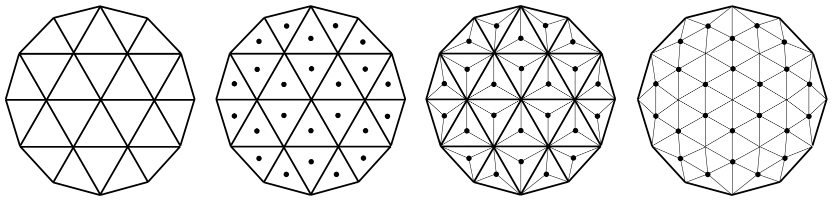


Figure 13.44: Illustration of the $\sqrt{3}$-subdivision scheme. A 1-to-3 split is performed instead of a 1-to-4 split as for Loop's and the modified butterfly schemes. First, a new vertex is generated at the center of each triangle. Then, this vertex is connected to the triangle's three vertices. Finally, the old edges are flipped. *(Illustration after Kobbelt [505].)*

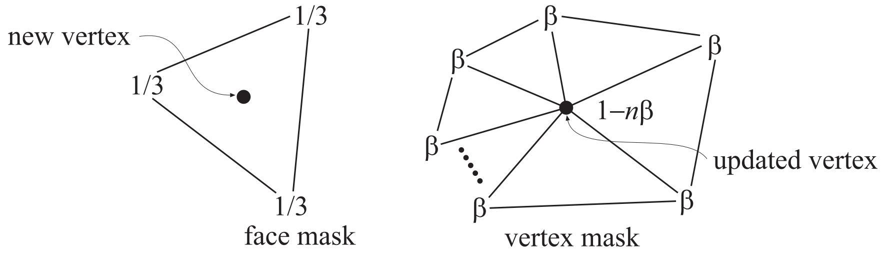


Figure 13.45: The masks for the $\sqrt{3}$-subdivision scheme. As can be seen, the face mask gives minimal support, since it uses only the three vertices of the triangle. The vertex mask uses all the vertices in the ring, called the 1-ring, around the vertex.

Each of the old vertices, $\mathbf{p}^k$, are updated using the formula in the second line, where $\mathbf{p}_i^k$ $(i = 0 \ldots n-1)$ denotes the immediate neighbors of $\mathbf{p}^k$, and n is the valence of $\mathbf{p}^k$. The subdivision step is denoted by k as before.

$$\begin{aligned}\mathbf{p}_m^{k+1} &= (\mathbf{p}_a^k + \mathbf{p}_b^k + \mathbf{p}_c^k)/3 \\ \mathbf{p}^{k+1} &= (1 - n\beta)\mathbf{p}^k + \beta \sum_{i=0}^{n-1} \mathbf{p}_i^k\end{aligned} \tag{13.57}$$

Again, β is a function of the valence n, and the following choice of $\beta(n)$ generates a surface that is C^2 continuous everywhere except at irregular vertices $(n \neq 6)$, where the continuity is at least C^1 [505].

$$\beta(n) = \frac{4 - 2\cos(2\pi/n)}{9n} \tag{13.58}$$

The masks, which are of minimum size, for the $\sqrt{3}$-scheme are shown in Figure 13.45.

The major advantage of this scheme is that it supports adaptive subdivision in a more natural way. See Kobbelt's paper [505] for details. Some other advantages of this scheme are smaller masks, and slower triangle growth rate than Loop's and the MB scheme. The continuity of this scheme is the same as Loop's. Disadvantages include that the edge flip introduces a little complexity, and that the first subdivision step sometimes generates unintuitive shapes due to the flip. In Figure 13.46, a worm is subdivided with the $\sqrt{3}$-scheme, and in Figure 13.38 on page 403, a tetrahedron is subdivided.

13.5.4 Catmull-Clark Subdivision

The two most famous subdivision schemes that can handle polygonal meshes (rather than just triangles) are Catmull-Clark [133] and Doo-Sabin [206].[8] Here, we will only briefly present the former. Catmull-Clark surfaces have been used

[8]Incidentally, both were presented in the same issue of the same journal.

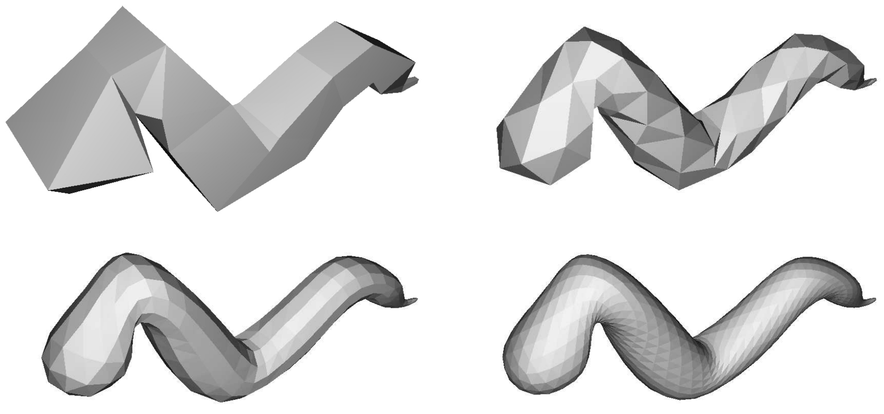


Figure 13.46: A worm is subdivided three times with the $\sqrt{3}$-subdivision scheme.

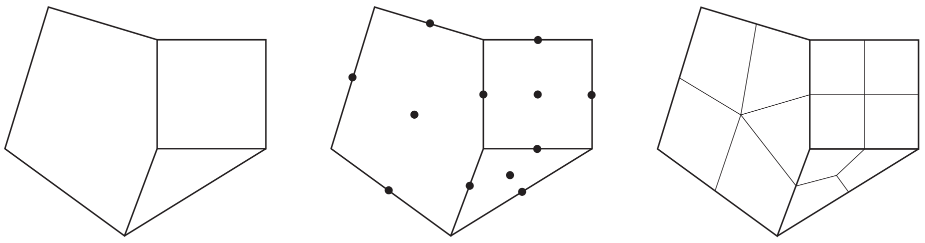


Figure 13.47: The basic idea of Catmull-Clark subdivision. Each polygon generates a new point, and each edge generates a new point. These are then connected as shown to the right. Weighting of the original points is not shown here.

in Pixar's short film *Geri's Game* [189] and in *Toy Story 2*, and in all subsequent feature films from Pixar. As pointed out by DeRose et al. [189], Catmull-Clark surfaces tend to generate more symmetrical surfaces. For example, an oblong box results in a symmetrical ellipsoid-like surface, which agrees with intuition.

The basic idea for Catmull-Clark surfaces is shown in Figure 13.47, and an actual example of Catmull-Clark subdivision is shown in Figure 13.32 on page 398. As can be seen, this scheme only generates faces with four vertices. In fact, after the first subdivision step, only vertices of valence 4 are generated, thus such vertices are called ordinary or regular (compared to valence 6 for triangular schemes).

Following the notation from Halstead et al. [364] (see Figure 13.48), let us focus on a vertex $\mathbf{v}^k$ with n surrounding edge points $\mathbf{e}_i^k$, where $i = 0 \dots n-1$. Now, for each face, a new face point $\mathbf{f}^{k+1}$ is computed as the face centroid, i.e., the mean of the points of the face. Given this, the subdivision rules are [133,

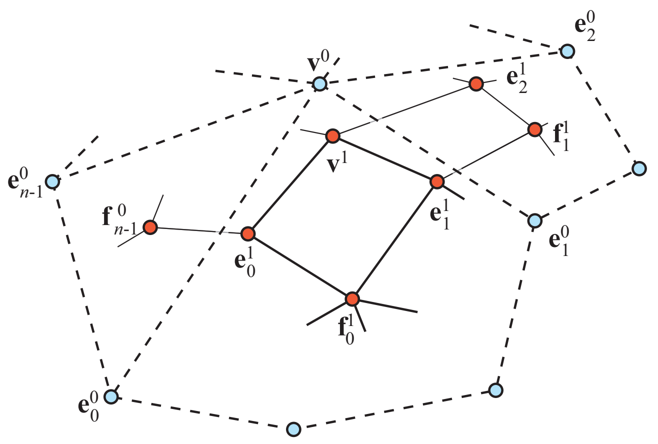


Figure 13.48: Before subdivision, we have the blue vertices and corresponding edges and faces. After one step of Catmull-Clark subdivision, we obtain the red vertices, and all new faces are quadrilaterals. Illustration after Halstead et al. [364].

364, 1024]:

$$\begin{aligned}\mathbf{v}^{k+1} &= \frac{n-2}{n}\mathbf{v}^k + \frac{1}{n^2}\sum_{j=0}^{n-1}\mathbf{e}_j^k + \frac{1}{n^2}\sum_{j=0}^{n-1}\mathbf{f}_j^{k+1},\\ \mathbf{e}_j^{k+1} &= \frac{\mathbf{v}^k + \mathbf{e}_j^k + \mathbf{f}_{j-1}^{k+1} + \mathbf{f}_j^{k+1}}{4}.\end{aligned} \tag{13.59}$$

As can be seen, new edge points are computed by the average of the considered vertex, the edge point, and the two newly created face points that have the edge as a neighbor. On the other hand, the vertex is computed as weighting of the considered vertex, the average of the edge points, and the average of the newly created face points.

The Catmull-Clark surface describes a generalized bicubic B-spline surface. So, for a mesh consisting only of regular vertices we could actually describe the surface as a B-spline surface.[9] However, this is not possible for irregular settings, and being able to do this using subdivision surfaces is one of the scheme's strengths. Limit positions and tangents are also possible to compute [364]. See Section **??** for a highly efficient technique on how to render Catmull-Clark subdivision surfaces on graphics hardware with tesseltation shaders.

13.5.5 Piecewise Smooth Subdivision

In a sense, curved surfaces may be considered boring because they lack detail. Two ways to improve such surfaces are to use bump or displacement maps

[9]See the SIGGRAPH course notes for more on this topic [1024].

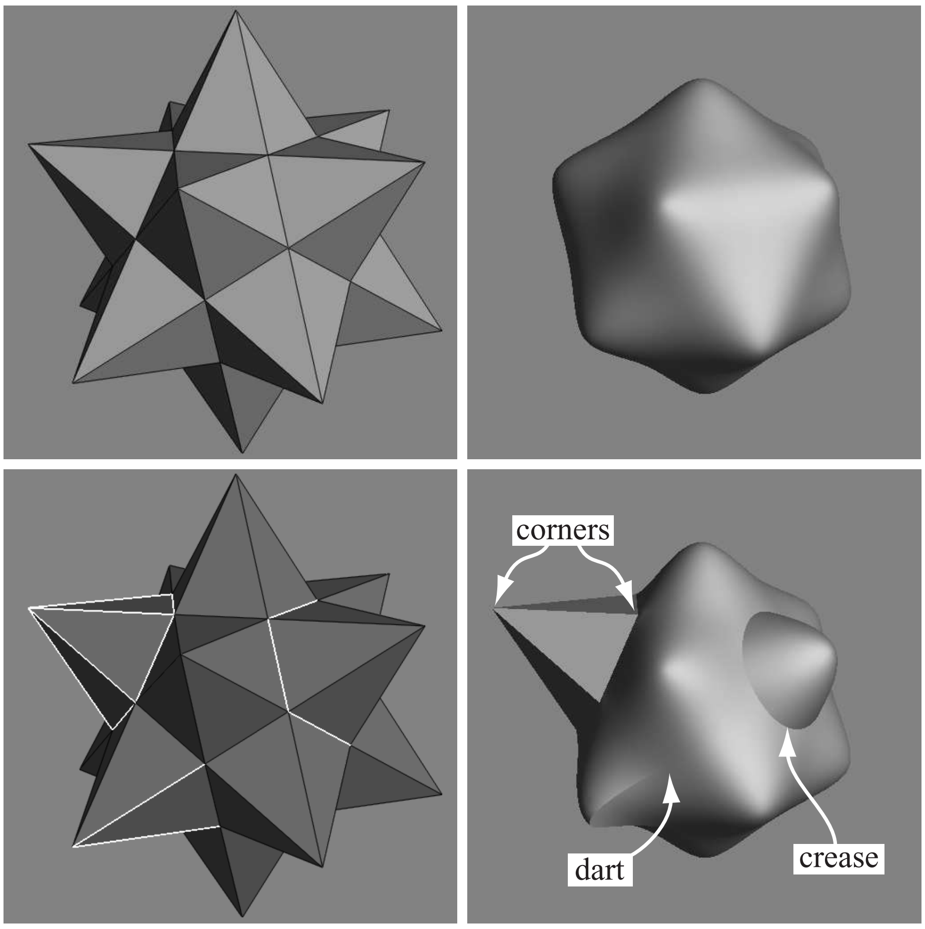


Figure 13.49: The top row shows a control mesh, and the limit surface using the standard Loop subdivision scheme. The bottom row shows piecewise smooth subdivision with Loop's scheme. The lower left image shows the control mesh with tagged edges (sharp) shown in a light gray. The resulting surface is shown to the lower right, with corners, darts, and creases marked. *(Image courtesy of Hugues Hoppe.)*

(Section 13.5.6). A third approach, *piecewise smooth subdivision*, is described here. The basic idea is to change the subdivision rules so that *darts*, *corners*, and *creases* can be used. This increases the range of different surfaces that can be modeled and represented. Hoppe et al. [418] first described this for Loop's subdivision surfaces. See Figure 13.49 for a comparison of a standard Loop subdivision surface, and one with piecewise smooth subdivision.

To actually be able to use such features on the surface, the edges that we want to be sharp are first tagged, so we know where to subdivide differently. The number of sharp edges coming in at a vertex is denoted s. Then the vertices are classified into: smooth ($s = 0$), dart ($s = 1$), crease ($s = 2$), and corner ($s > 2$). Therefore, a crease is a smooth curve on the surface, where the continuity across the curve is C^0. A dart is a nonboundary vertex where a crease ends and smoothly blends into the surface. Finally, a corner is a vertex where three or more creases come together. Boundaries can be used by marking each boundary edge as sharp.

After classifying the various vertex types, Hoppe et al. use a table to determine which mask to use for the various combinations. They also show how to compute limit surface points and limit tangents. Biermann et al. [70] present several improved subdivision rules. For example, when extraordinary vertices are located on a boundary, the previous rules could result in gaps. This is avoided with the new rules. Also, their rules make it possible to specify a normal at a vertex, and the resulting surface will adapt to get that normal at that point. DeRose et al. [189] present a technique for creating soft creases. Basically, they allow an edge to first be subdivided as sharp a number of times (including fractions), and after that, standard subdivision is used.

13.5.6 Displaced Subdivision

While parametric surfaces and subdivision surfaces are great for describing smooth surfaces, they are sometimes not that useful because the surfaces lack detail. One solution would be to use bump mapping (see Section 6.8). However, this is just an illusionary trick that changes the normal in each pixel, and thus, the shading. The silhouette of an object looks the same with or without bump mapping. The natural extension of bump mapping is *displacement mapping* [155], where the surface is displaced. This is usually done along the direction of the normal. So, if the point of the surface is $\mathbf{p}$, and its normalized normal is $\mathbf{n} = \mathbf{n}'/||\mathbf{n}'||$, then the point on the displaced surface is:

$$\mathbf{s} = \mathbf{p} + d\mathbf{n}. \tag{13.60}$$

Here, the scalar d is the displacement at the point $\mathbf{p}$. The displacement could also be vector-valued [512].

In this section, the *displaced subdivision surface* [556] will be presented. The general idea is to describe a displaced surface as a coarse control mesh that is subdivided into a smooth surface that is then displaced along its normal using a scalar field. In the context of displaced subdivision surfaces, $\mathbf{p}$ in Equation 13.60 is the limit point on the subdivision surface (of the coarse control mesh), and $\mathbf{n}$ is the normalized normal at $\mathbf{p}$, computed as:

$$\begin{aligned} \mathbf{n}' &= \mathbf{p}_u \times \mathbf{p}_v, \\ \mathbf{n} &= \frac{\mathbf{n}'}{||\mathbf{n}'||}. \end{aligned} \tag{13.61}$$

In Equation 13.61, $\mathbf{p}_u$ and $\mathbf{p}_v$ are the first-order derivative of the subdivision surface. Thus, they describe two tangents at $\mathbf{p}$. Lee et al. [556] use a Loop subdivision surface for the coarse control mesh, and its tangents can be computed

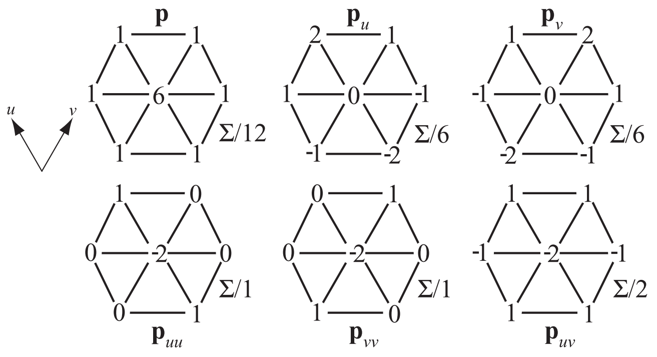


Figure 13.50: The masks for an ordinary vertex in Loop's subdivision scheme. Note that after using these masks, the resulting sum should be divided as shown. *(Illustration after Lee et al. [556].)*

using Equation 13.51. Note that the notation is slightly different here; we use $\mathbf{p}_u$ and $\mathbf{p}_v$ instead of $\mathbf{t}_u$ and $\mathbf{t}_v$. Equation 13.60 describes the displaced position of the resulting surface, but we also need a normal, $\mathbf{n}_s$, on the displaced subdivision surface in order to render it correctly. It is computed analytically as shown below [556]:

$$\begin{aligned}
\mathbf{s}_u &= \frac{\partial \mathbf{s}}{\partial u} = \mathbf{p}_u + d_u\mathbf{n} + d\mathbf{n}_u,\\
\mathbf{s}_v &= \frac{\partial \mathbf{s}}{\partial v} = \mathbf{p}_v + d_v\mathbf{n} + d\mathbf{n}_v,\\
\mathbf{n}_s &= \mathbf{s}_u \times \mathbf{s}_v.
\end{aligned} \tag{13.62}$$

To simplify computations, Blinn [81] suggests that the third term can be ignored if the displacements are small. Otherwise, the following expressions can be used to compute $\mathbf{n}_u$ (and similarly $\mathbf{n}_v$) [556]:

$$\begin{aligned}
\bar{\mathbf{n}}_u &= \mathbf{p}_{uu} \times \mathbf{p}_v + \mathbf{p}_u \times \mathbf{p}_{uv},\\
\mathbf{n}_u &= \frac{\bar{\mathbf{n}}_u - (\bar{\mathbf{n}}_u \cdot \mathbf{n})\mathbf{n}}{||\mathbf{n}'||}.
\end{aligned} \tag{13.63}$$

Note that $\bar{\mathbf{n}}_u$ is not any new notation, it is merely a "temporary" variable in the computations. For an ordinary vertex (valence $n = 6$), the first and second order derivatives are particularly simple. Their masks are shown in Figure 13.50. For an extraordinary vertex (valence $n \neq 6$), the third term in rows one and two in Equation 13.62 is omitted.

The displacement map for one triangle in the coarse mesh is a scalar field, that is, a heightfield. The storage requirements for one triangle is one half of $(2^k + 1) \times (2^k + 1)$, where k is the number of subdivisions that the displacement map should be able to handle, which depends on the desired accuracy. The displacement map uses the same parameterization as the underlying subdivision mesh. So, for example, when one triangle is subdivided, three new points

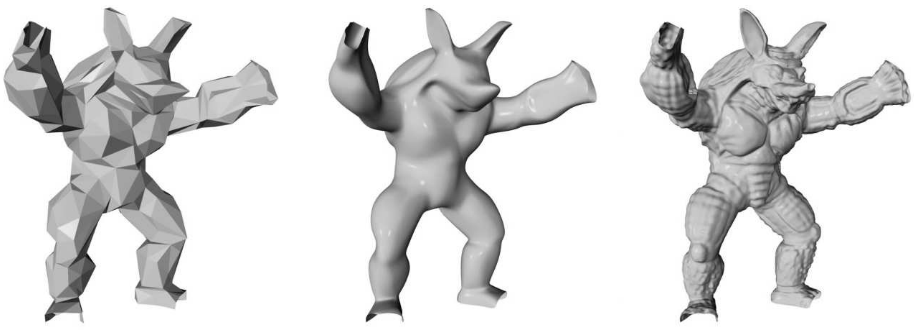


Figure 13.51: To the left is a coarse mesh. In the middle, it is subdivided using Loop's subdivision scheme. The right image shows the displaced subdivision surface. *(Image courtesy Aaron Lee, Henry Moreton, and Hugues Hoppe.)*

are created. Displacements for these three points are retrieved from the displacement map. This is done for k subdivision levels. If subdivision continues past this maximum of k levels, the displacement map is subdivided as well, using Loop's subdivision scheme. The subdivided displacement, d, is added using Equation 13.60. When the object is farther away, the displacement map is pushed to the limit points, and a mipmap pyramid of displacements is built as a preprocess and used. The resulting displaced subdivision surface is C^1 everywhere, except at extraordinary vertices, where it is C^0. Remember that after sufficiently many subdivision steps, there is only a small fraction of vertices that are extraordinary. An example is shown in Figure 13.51.

When a displaced surface is far away from the viewer, standard bump mapping could be used to give the illusion of a displaced surface. This is especially advantageous if the rendering bottleneck is geometry processing. Some bump mapping schemes need a tangent space coordinate system at the vertex, and the following can be used for that: $(\mathbf{b}, \mathbf{t}, \mathbf{n})$, where $\mathbf{t} = \mathbf{p}_u/||\mathbf{p}_u||$ and $\mathbf{b} = \mathbf{n} \times \mathbf{t}$.

Lee et al. [556] also present how adaptive tessellation and backpatch culling can be used to accelerate rendering. More importantly, they present algorithms to derive the control mesh and the displacement field from a detailed polygon mesh.

13.5.7 Normal, Texture, and Color Interpolation

In this section, we will present different strategies for dealing with normals, texture coordinates and color per vertex.

As shown for Loop's scheme in Section 13.5.1, and the MB scheme in Section 13.5.2, limit tangents, and thus, limit normals can be computed explicitly. This involves trigonometric functions that are expensive to evaluate. However, a very short look-up table can do the job efficiently. Another approach is to compute limit normals (that are exact) at the vertices of the control mesh, and then use the same subdivision scheme used for the vertices to subdivide the normals as well [459]. However, this increases the storage need during subdivision, and it is not obvious whether this is faster. Also, with this scheme, the exact normals will not be generated in the end.

Assume that each vertex in a mesh has a texture coordinate and a color. To be able to use these for subdivision surfaces, we also have to create colors and texture coordinates for each newly generated vertex, too. The most obvious way to do this is to use the same subdivision scheme as we used for subdividing the polygon mesh. For example, you can treat the colors as four dimensional vectors (RGBA), and subdivide these to create new colors for the new vertices [841]. This is a reasonable way to do it, since the color will have a continuous derivative (assuming the subdivision scheme is at least C^1), and thus abrupt changes in colors are avoided over the surface The same can certainly be done for texture coordinates [189]. A sophisticated scheme for texturing subdivision surfaces is given by Piponi and Borshukov [741].

13.6 Efficient Tessellation

To actually use curved surfaces in a real-time rendering context, we often need to create triangles on the surface. This process is known as *tessellation.* The simplest form of tessellation is called *uniform tessellation.* Assume that we have a parametric Bézier patch, $\mathbf{p}(u, v)$, as described in Equation 13.27. We want to tessellate this patch by computing 11 points per patch side, resulting in $10 \times 10 \times 2 = 200$ triangles. The simplest way to do this is to sample the *uv*-space *uniformly.* Thus, we evaluate $\mathbf{p}(u, v)$ for all $(u_k, v_l) = (0.1k, 0.1l)$, where both k and l can be any integer from 0 to 10. This can be done with two nested `for`-loops. Two triangles can be created for the four surface points $\mathbf{p}(u_k, v_l)$, $\mathbf{p}(u_{k+1}, v_l)$, $\mathbf{p}(u_{k+1}, v_{l+1})$, and $\mathbf{p}(u_k, v_{l+1})$.

While this certainly is straightforward, there are faster ways to do it. Instead of sending tessellated surfaces, consisting of many triangles, over the bus from the CPU to the GPU, it makes more sense to send the curved surface representation to the GPU, and let it handle the data expansion. In the following five subsections, we will describe tessellation hardware, fractional tessellation, the evaluation shader, and how to render Catmull-Clark surfaces and displacement mapped surfaces with such hardware. This type of tessellaion is supported by the ATI Radeon HD 2000 series and also by the XBOX 360. Techniques for adapative tesselation is described in Section 13.6.4.

13.6.1 Hardware Tessellation Pipeline

An efficient way of providing inexpensive data expansion of geometry, is to send higher order surfaces to the GPU, and let it tessellate the surface. This can be done with a rather small, but important, change in the rendering pipeline. This is illustrated in Figure 13.52.

The tessellator uses a fractional tessellation technique, which is described in the subsequent section. This basically tessellates a triangle or a quadrilateral