PROCEEDINGS OF SYMPOSIUM ON COMPUTER GEOMETRY SCG' 2004, VOLUME 13,
pp. xy -xz

BASIC SUBDIVISION SCHEMES ON TRIANGULAR MESHES

Martin Samuelčík[1]

[1]FMPI UK, Mlynská dolina, 812 31 Bratislava, SR, e-mail: samuelcik@fmph.uniba.sk

Abstract. Subdivision curves and surfaces are very useful parts of current geometric modeling. In this paper we present two basic subdivision schemes for triangular meshes, Loop scheme and modified butterfly scheme. Main part of this work is data structure for storing triangular mesh that saves memory and subdivision schemes can be performed fast using this structure. We also give steps how to perform described schemes on that structure.

Keywords: subdivision, triangular mesh, Loop scheme, modified butterfly scheme

1. Subdivision schemes

Subdivision approach for geometric modeling is not a new part in this area. For example de Casteljau algorithm for Bézier curves produces two control polygons that are a better approximation of Bézier curve than the previous control polygon. And this is the basic idea of subdivision. Given initial polygon or mesh, we try to construct sequence of polygons or meshes that converge to some limit curve or surface. In each step we get better approximation of limit object. Such process is described by rules, which say how to perform one step (iteration) of the sequence. Good example is Chaikin's algorithm for creating smooth curve with basic subdivision rules. This so-called "corner cutting" algorithm produces in each step two new vertices on each line of polygon, then removes old vertices and connect new vertices. Special rules are applied at the first and last line. Figure 1 illustrates this process.

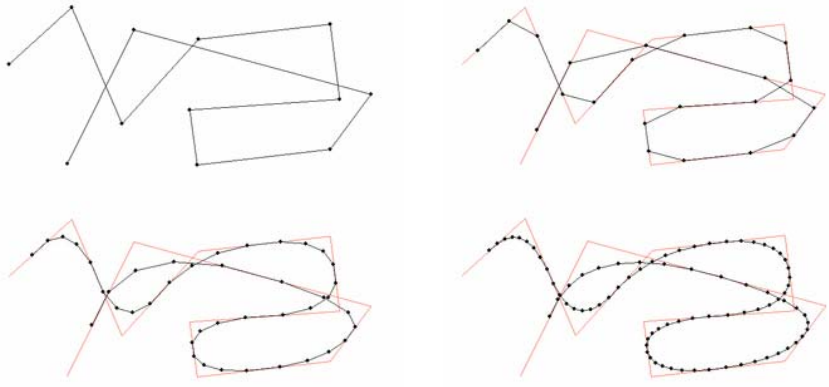

Figure 1 Chainkin subdivision curve.

There exist different types of subdivision schemes. The two fundamental types are approximating and interpolating schemes. Approximate scheme produces object that approximates initial polygon or mesh, interpolating scheme in each step contains vertices from previous step so limit object interpolates vertices of initial object. We can also differ schemes by smoothness of limit object.

We will focus now on the subdivision surfaces, especially on the surfaces described as triangular meshes. Now we are going to present two most simple schemes for creating approximating and interpolating limit surfaces. These two schemes are local so to change the

position of old vertices and to create new ones we will use only old vertices from local neighborhood of that vertex. Each description of the scheme consists of three parts:

1. How to create new vertices, its geometric position (coordinates)
2. How to change geometric position of old vertices in mesh
3. How to connect new vertices with old and other new vertices (topology)

These rules can be displayed graphically and are called mask of the scheme. We will see, that these masks are different for regular vertices and extraordinary vertices. Regular vertices are for triangular meshes with valence (number of neighbors) equal to 6 in the middle of mesh and 4 on the boundary, extraordinary vertices are the others. More precise description of subdivision process can be found in [2].

1.1. Loop scheme

Loop scheme is approximating scheme, which produces C^1 smooth surface near extraordinary vertices, and C^2 smooth surface elsewhere [1]. Figure 2 shows masks of this simple scheme.

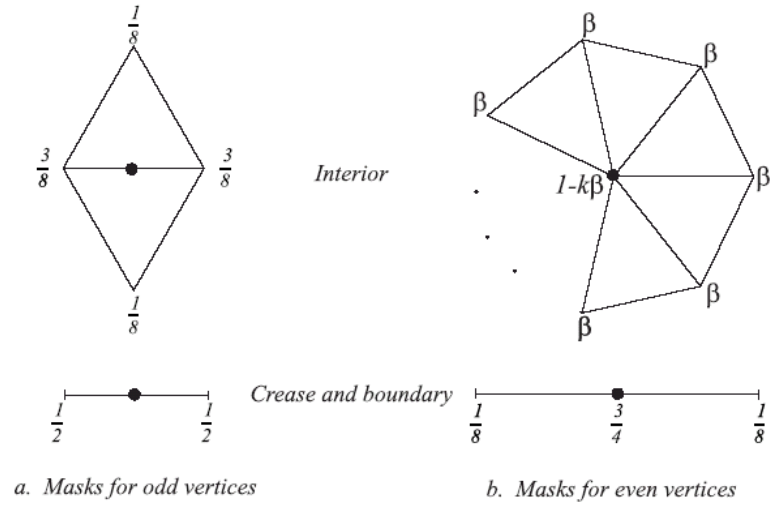

Figure 2 Masks for Loop scheme.

We give an explanation of these masks. As we mentioned before, the masks show how to create new vertices and move the old ones. In the case of Loop scheme we have:

1. Mask for creating new vertices (called odd vertices) is on the left side. It means that the new vertex is created between two old vertices as barycentric combination of the four nearest old vertices with barycentric coefficients given by the mask. For new vertices on the boundary or the crease there is also a simple mask with only two old vertices.
2. Mask on the right side shows how to change position of old vertices (called even vertices). Again the new position of old vertex is computed as a barycentric combination of its old position and the old position of its neighbors. The barycentric coefficients are written in the mask, as well as the rule for moving old vertices on boundary or crease. Here k is the valence of the current vertex and for β we can use two approaches. One approach is to put

$\beta = \frac{1}{k}(\frac{5}{8} - (\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{k})^2)$, the other approach is $\beta = \frac{3}{16}$ for $k=3$ and $\beta = \frac{3}{8k}$ for $k>3$.

3. The last step is to change the topology of mesh and to include the new vertices. Simply for each old triangle all edges are removed, between two old vertices connected with old edge new vertices are inserted and these new vertices are connected with each other. Figure 3 shows this refinement.

Figure 3 Topology refinement for one triangle and inserting new vertices.

To perform one step of the subdivision process, new vertex should be created for each edge, each old vertex should be moved to the new position and each edge should be removed and the new edges should be used to create more precise topology.

1.2. Modified butterfly scheme

This is interpolating scheme that produces C^1 smooth surfaces except the neighborhood of the extraordinary vertices, where it creates artifacts [3]. There are special rules for neighborhood of the regular and the extraordinary vertices. Masks are showed in the Figure 4.

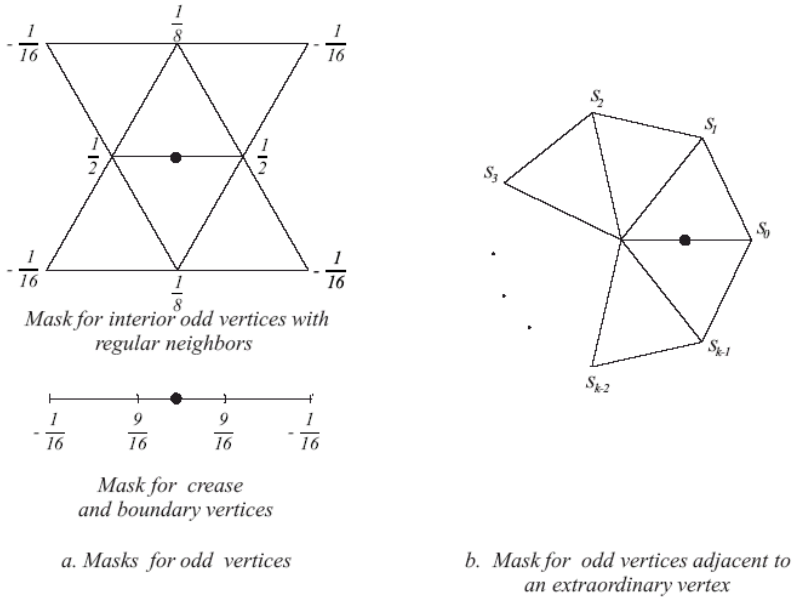

Figure 4 Masks for modified butterfly scheme

This scheme is working on the triangular mesh and its rules are similar to Loop scheme, the explanation of masks is following:

1. Creating of new vertices is the most difficult part of rules. For each edge we want to insert new vertex between end vertices of edge. We have three situations here:
   a. Both end vertices are regular. Then we use mask on the left side of Figure 4. All coefficients are used as in Loop scheme, also for boundary and crease vertices.
   b. If one of the end vertices is an extraordinary vertex, then we use mask on the right side of Figure 4. As we see, we use for creation of the new vertex only the vertices from neighborhood of the extraordinary vertex without that vertex. The coefficients s_i are $\frac{1}{k}(\frac{1}{4}+\cos\frac{2i\pi}{k}+\frac{1}{2}\cos\frac{4i\pi}{k})$ for $k>4$. For k=3, $s_0=\frac{5}{12}, s_{1,2}=-\frac{1}{12}$; for $k = 4$ $s_0=\frac{3}{8}, s_2=-\frac{1}{8}, s_{1,3}=0$. The current vertex has weight (barycentric coefficient) equal to 0,75.
   c. If both end vertices are extraordinary, we use the mask for end vertices separately as in case b. and we get the final result as barycentrum of these two computed vertices.
2. Because this scheme is interpolating, we leave the old vertices at the old position so we don't move them.
3. The topology refinement is again equal to the refinement of Loop scheme.

2. Data structure

Our data structure for triangular mesh is very simple and stores only the nearest neighborhood of one vertex. So we have only one table (vertex table) with the records for vertices (number of rows in table is number of vertices in mesh). This vertex record contains the following:

- Geometric position (coordinates of vertex).
- Valence of vertex.
- List of indexes of vertex neighbors. These indexes are counterclockwise sorted equally to the order of neighbors around vertex, so we have also information about the orientation in the vertex. Index is number of record (line) in table.
- Additional custom data.

This structure does not need much memory. For exact amount of used memory we have formula:

*memory = number_of_vertices * (3 * sizeof(float) + sizeof(integer) + sizeof(custom_data)) + number_of_edges * 2 * sizeof(integer)*

So for triangle mesh we have memory usage equal to *O(n)*. But we want to use this structure to perform subdivision process on stored triangle mesh. Description of this process is part of the next section.

2.1. Performing Loop scheme

For this subdivision process we have three main steps that need to be performed, so we will describe them now in the case of our structure.

1. This part is simple, we are going through all vertices and for current vertex we are going through all his neighbors. If current neighbor of current vertex was not processed, we can insert a new vertex between them. The mask for this new vertex is using only four vertices from neighborhood of current vertex so access to them is very easy. At the end of this process we get all new vertices.
2. This part can be done with one pass of all vertices and using the given mask, which contains only vertices from vertex neighborhood.
3. Last part is topology refinement. If we are inserting new vertex, we have to remove old edge and replace it with two new edges connecting new vertex and two endpoints of the old edge. This means that we have to change one number in the list of neighborhood vertices for both endpoint vertices and replace it with the index of new created vertex. To connect the newly created vertices in the triangles, we only need to connect the newly created vertices when passing the neighbors of the current vertex.

In each of these three steps, we are going through all vertices, so we can do all these steps in one pass, so time complexity is also *O(n)*. but then we have to remember the old position of the old vertices.

2.2. Performing modified butterfly scheme

Because modified butterfly and Loop subdivision are different only in masks for the old and new vertices, so if we want to perform this scheme, it is very similar to previous one. We only have to add new steps when creating the new vertices, but we don't need to move the old vertices. The topological refinement is the same.

3. Implementation & results

We implemented the described structure for the triangular meshes and also both subdivision processes. We use this implementation to show how fast can the subdivision process be using this structure. We implemented it using Visual Studio .NET with OpenGL library for displaying meshes. We tested implementation on the PC with AthlonXP 1700+, 348 MB memory and GeForce 2 MX graphics card. Our implementation also contained a part for loading meshes from .ply format.

Figures below show few steps of subdivision process on some meshes. In table we can see time and memory complexity of meshes needed for performing subdivision process.

Loop scheme	Vertices	Edges	Triangles	Time	Memory
0	6789	20361	13574	0 s	380136 B
1	27105	81444	54296	0.609 s	1520352 B
2	108594	325776	217184	1.437 s	6081216 B
3	434370	1303104	868736	3.360 s	24324672 B
4	1737474	5212416	3474944	12.015 s	97298496 B

Table 1 Time and memory complexity for Loop scheme performed on the screwdriver model.

Figure 5 The original model of screwdriver and the model after one step of Loop subdivision process.

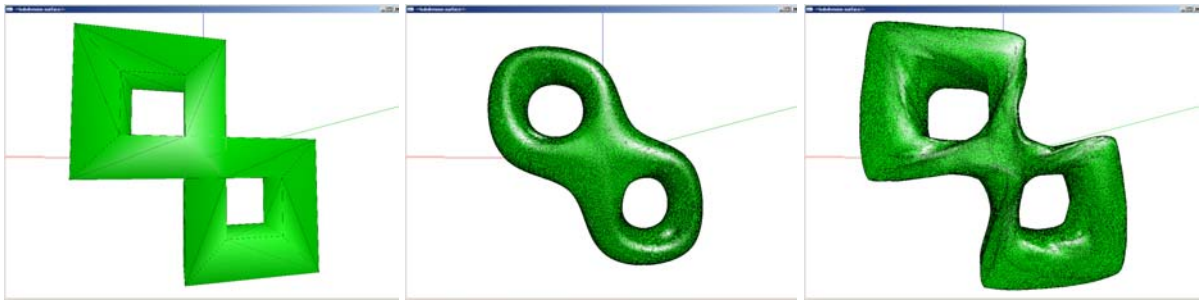

Figure 6 Model of a double torus and its approximating and interpolating subdivision surfaces.

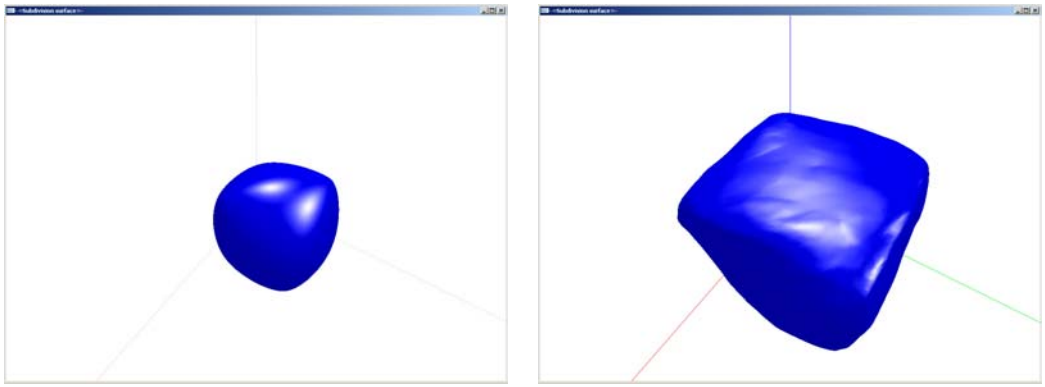

Figure 7 Model of a cube with six steps of Loop (left) and modified butterfly (right) subdivision.

References

[1] LOOP, C. Smooth Subdivision Surfaces Based on Triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[2] ZORIN, D., SCHROEDER, P., AND SWELDENS, W. Interpolating Subdivision for Meshes with Arbitrary Topology. Computer Graphics Proceedings (SIGGRAPH 96) (1996), pp. 189–192.

[3] Subdivision for Modeling and Animation, SIGGRAPH 2000 course, http://mrl.nyu.edu/~dzorin/sig00course/