Texture Mapping on Doo-Sabin Subdivision Surfaces Using Multiple Images

Zhiyong Huang[1] and Chee Seng Neo[2]
[1]*Department of Computer Science, School of Computing*
National University of Singapore, Singapore 117543
E-mail: huangzy@comp.nus.edu.sg

[2]*Network infrastructure Systems Division, DSTA, Singapore 118253*
E-mail: ncheesen@dsta.gov.sg

Abstract

We propose a texture mapping method on subdivision surfaces using multiple images and have implemented it on Doo-Sabin scheme. At the beginning, one texture map is specified for each control mesh face respectively. In a subdivision process, a new face of control meshes may fall into a region of multiple texture maps. To correctly texture map on such faces, we need to further split the new faces into multiple parts so that each of them falls into a region of only one texture map. A splitting algorithm is devised. The novelty of our work is on a generalization of the method of DeRose et al. for the treatment of the use of multiple images.

Subject Descriptions:
I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - Surfaces Representation

Keywords:
Texture mapping, Subdivision surfaces, Multiple images

1. Introduction

Subdivision surfaces are important in geometric modeling [7]. Modeling with subdivision surfaces, complex models can be modeled with the efficiency of polygons and the smoothness of NURBS and other spline surfaces without trimming. Texture mapping is an important technique in computer graphics [5]. The major texture mapping method on subdivision surfaces was proposed by DeRose et al. [2] and implemented on Catmull-Clark subdivision surfaces using only one image as the texture map. In this short presentation, we extend the algorithm to texture mapping on subdivision surfaces using multiple images. We implemented our method on Doo-Sabin subdivision surfaces because their new faces of control meshes may fall into a region of multiple texture maps, where the method of DeRose et al. can not be directly applied. To correctly texture map on such faces, we need to further split the new faces into multiple parts so that each part falls into a region of only one texture map. We have addressed the problem of maintaining smoothness of texture between any adjacent faces because of the use of multiple images. The novelty of our work is on a generalization of the method of DeRose et al. for the treatment of the use of multiple images.

Problem formulation: At the beginning, multiple images (texture maps) are defined for each face of the Doo-Sabin initial control meshes respectively. In a subdivision process, a new face of control meshes may fall into a region of multiple texture maps. A splitting algorithm is devised to further split the new face into multiple parts so that each of them falls into a region of only one texture map. Then, the method of DeRose et al. can be applied.

2. Background

Subdivision surfaces have been studied for about 20 years for representing complex surfaces. The first two schemes were given by Catmull and Clark (Catmull-Clark) [1] and Doo and Sabin (Doo-Sabin) [3]. Different subdivision surfaces proposed later include Loop [6] and Butterfly scheme [4]. Recently, these techniques have received more attention in computer graphics because of the many benefits of subdivision [7].

We briefly describe the Doo-Sabin subdivision surfaces because it will introduce the problem for the texture mapping using more than one image: the new faces of control meshes can fall into a region of multiple texture maps. Doo-Sabin scheme is presented as a generalization of a recursive bi-quadratic B-splines patch subdivision algorithm. For non-rectangular meshes, it generates surfaces that reduce to a standard B-spline surface except at a small number of points, called extraordinary points. The limit surface of the scheme is C^1 continuous except at extraordinary points. The scheme works on all meshes regardless of their topology (Figure 1).

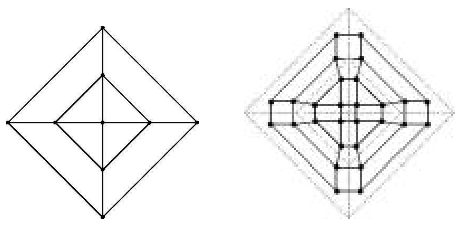


Figure 1: One refinement using Doo-Sabin scheme.

We use a cube for initial mesh (Figure 2). For each subdivision, the new vertex can be derived from (1) the average of four particular points taken in a polygon - the vertex for which the new point is being defined, (2) the two edge points (the midpoints of the edges that are adjacent to this vertex in the polygon), or (3) the face point (the average of all the points in the polygon).

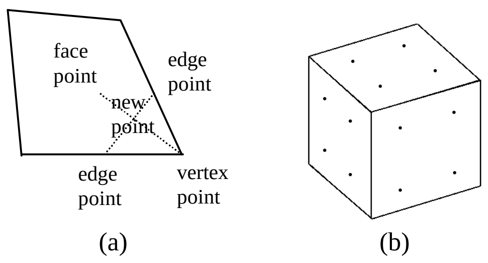


Figure 2: (a) New point is the average of 2 edge points, 1 face point, and 1 vertex point. (b) New points derived on the cube.

There are three types of faces formed from subdivision: F-faces, E-faces, and V-faces.

F-faces (Figure 3): For each n-sided face F in the original polyhedron, linking the new vertices of F forms a new n-sided face.

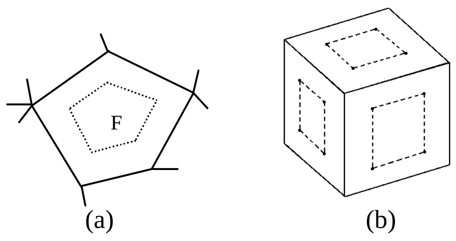


Figure 3: (a) Forming an F-face on a pentagon face. (b) Forming F-faces on the cube.

E-faces (Figure 4): For each edge E common to two faces F and F', a new 4-sided face is made by linking the images of the end vertices of E on the faces F and F'.

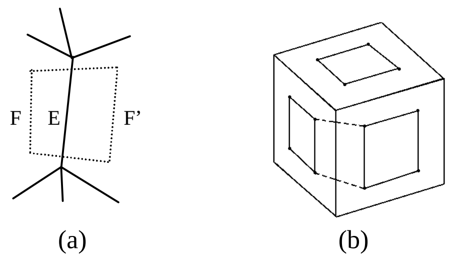


Figure 4: (a) Forming an E-face along an edge. (b) Forming an E-face on the cube.

V-faces (Figure 5): For each n-spoked (n>2) vertex V, where n faces meet, a new n-sided face is formed by linking the new vertices formed by V on the faces meeting at V.

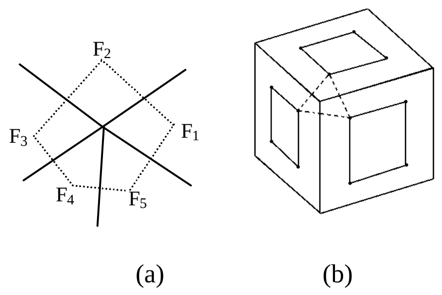


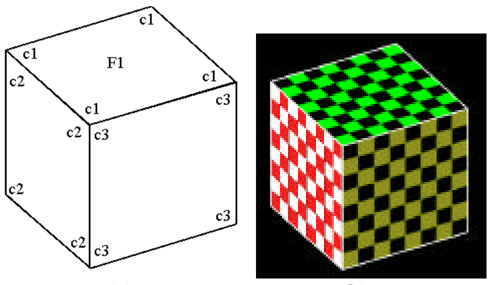
Figure 5: (a) Forming a V-face around a vertex. (b) Forming a V-face on the cube

Now, we brief the texture mapping method of DeRose et al. [2]. The goal is the construction of smooth texture coordinates for Catmull-Clark surfaces. They have proved that smoothly varying texture coordinates result if the texture coordinates(s, t) assigned to the control vertices are subdivided using the same subdivision rules as used for the geometric coordinates (x, y, z). In other words, control point positions and subdivision can be thought of as taking place in a 5-space consisting of (x, y, z, s, t) coordinates.

3. Our Work

In this section, we describe our work of texture mapping on Doo-Sabin subdivision surfaces using multiple images. When a new face falls into a region of one texture map completely, the texture coordinates of the new vertices are derived from the texture coordinates of the old vertex coordinates using the same subdivision rule, i.e., the method of DeRose et al. [2] can be applied directly. Thus, we only need to focus on the case where a new face falls into a region with multiple texture maps.

Without the loss of generality, a cube is used as the initial shape throughout all subsections. Each face is mapped with six chessboard images shown in different colors (Figure 6). For example, in the initial polygon mesh, the top, left, and front control meshes are mapped with texture maps c_1, c_2, and c_3 respectively.



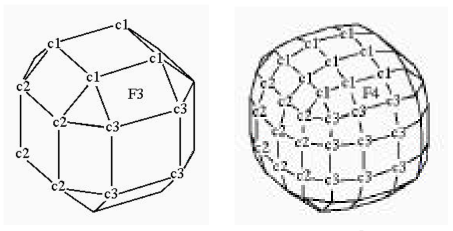
(a) (b)

Figure 6: Faces of the control mesh are mapped with different texture maps.

Our method is based on the different treatments, i.e., splitting, to F-face, E-face and V-face, the only three types of the faces after each subdivision. The details of the splitting are described from 3.1 to 3.3.

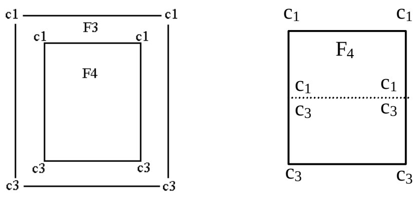
3.1 Texture mapping on F-faces

The case of a new F-face falling into a region of two texture maps is illustrated in Figure 7 and 8. To derive the texture coordinates of the vertices of the new F-face, we need to split it, e.g., F_4 in the illustration, into two parts so that each part falls into a region of only one texture map (Figure 8 (b)). Then, the texture coordinates of the new vertices are derived from the texture coordinates of the old vertex coordinates using the same subdivision rule separately with two texture maps c_1 and c_3.



(a) (b)

Figure 7: After one refinement, F3 will form F4.



(a) (b)

Figure 8: (a) F_4 is formed by F_3. (b) Splitting of the new E-face F_4 into two equal parts.

We describe how a face is split now. In Figure 9, each vertex is represented as (geometric coordinates, texture coordinates) pair. Coordinates u_0 to u_3 (v_0 to v_3) are defined in the texture map c_1 (c_2) with the texture coordinates s_0 to s_3 (t_0 to t_3).

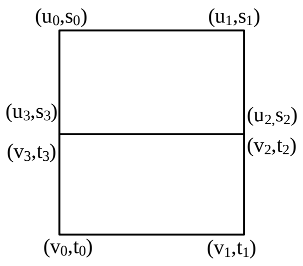


Figure 9: Each point is represented as (geometric coordinates, texture coordinates) pair.

After splitting, we can interpolate in each face to derive s'_0, s'_1, t'_0, and t'_1 (Figure 10 (a)) using the texture map c_1 and c_2. Now, we still have to find four more texture coordinates (represented by '?" in Figure 10 (b)), two for each texture map in order to map two texture maps on this face.

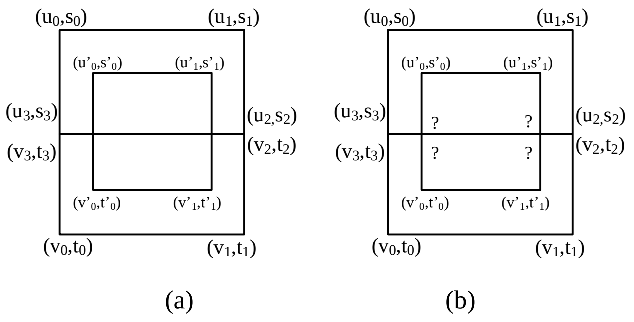


Figure 10: (a) Four new texture coordinates formed. (b) Two more texture coordinates for each texture are needed.

The computing for s_3' is as follows:

$a= s_2 – s_3$, $b=s_0'-s_3$,
$s_3'= s_3 + (|a\cdot b|/|a|^2)\ a$

The computing is similar for s_2' and other two texture coordinates t_3' and t_2' from another texture map (Figure 11).

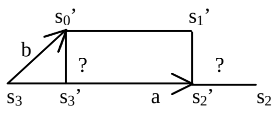


Figure 11: Computing diagram for s_3' and s_2'.

The above computing is correct for regular texture mapping (that means the texture coordinates form a rectangle). The technique will fail for irregular texture mapping as shown in Figure 12. Hence we need to adjust the texture-coordinates to achieve smooth and continuous texture mapping on the face.

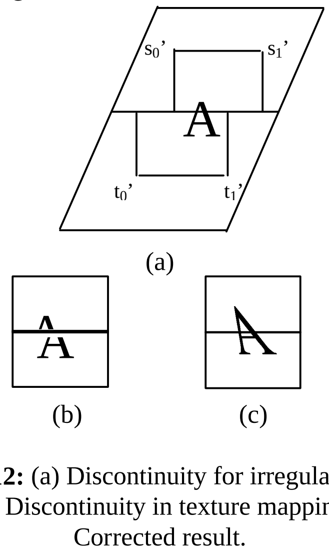


Figure 12: (a) Discontinuity for irregular texture mapping. (b) Discontinuity in texture mapping on F4. (c) Corrected result.

In order to solve the problem, we need to compute the texture coordinates s_{3n}', s_{2n}', t_{3n}', and t_{2n}'as shown in Figure 13.

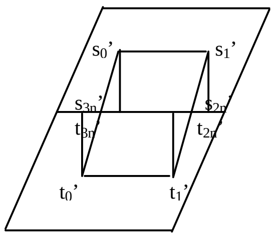


Figure 13: Position of new texture coordinates to maintain continuity.

The computing for s_{3n}' is as follows (Figure 14):

$u = (s_2 – s_3)/||s_2, s_3||$, $v = (t_2 – t_3)/|| t_2, t_3||$
$\text{norm\_dist}(s_0', s_3') = || s_0', s_3'||/ || s_2, s_3||$,
$\text{norm\_dist}(t_0', t_3') = || t_0', t_3'||/ || t_2, t_3||$,
$\text{norm\_dist}(s_3', t_3') = || s_3,t_3'||/ ||t_2, t_3||– || s_3, s_3'||/ || s_2,s_3||$,
$\text{norm\_dist}(s_3', t_{3n}') =$
$(\text{norm\_dist}(s_0',s_3')*\text{norm\_dist}(s_3', t_3')) /$
$(\text{norm\_dist}(s_0', s_3')+\text{norm\_dist}(t_0', t_3')$,
$s_{3n}' = s_3'+u*\text{norm\_dist}(s_3', t_{3n}')*||s_2, s_3||$,
$t_{3n}' = t_3'–v*(\text{norm\_dist}(s_3', t_3')-\text{norm\_dist}(s_3',t_{3n}'))* ||t_2,t_3||$,

where $||x, y||$ is the Euclidean distance between x and y.

The computing is similar for s_{2n}', t_{3n}', and t_{2n}'.

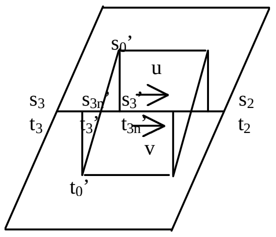


Figure 14: Computing diagram for t_{3n}'.

Finally, we discuss the case of an F-face falling into a region of multiple texture maps (Figure 15). To derive the texture coordinates for the new vertices, we split the new F-face into three parts as shown in Figure 16 (b).

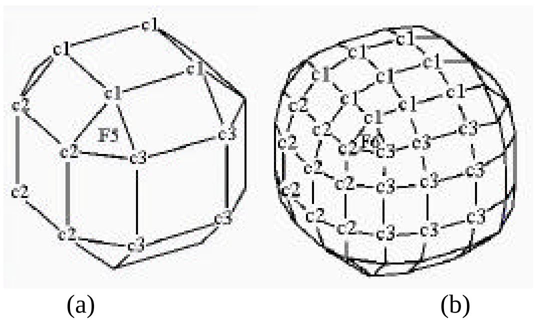


Figure 15: After one refinement, F5 will form F6.

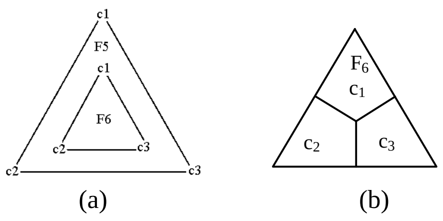


Figure 16: (a) F6 is formed by F5. (b) Splitting of new V-face into three parts.

The computation of the texture coordinates is illustrated in Figure 17. s_1 and s_3 can be computed, as they are points on the E-Faces. The texture coordinates for this faces is derived as follows:

$s_0' = 0.25*(s_0 + s_1 + s_2 + s_3)$,
$s_2' = s_2$.

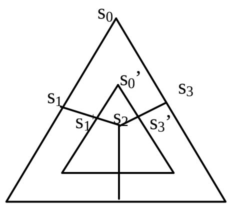


Figure 17: Computing diagram for s_2'.

3.2 Texture mapping on E-faces

When the new E-face falls into a region of two different texture maps, e.g., c1 and c3 for F3 in Figure 18 and E3 in Figure 19. To compute the texture coordinates of the new vertices, we need to split the new E-face into two parts (Figure 20).

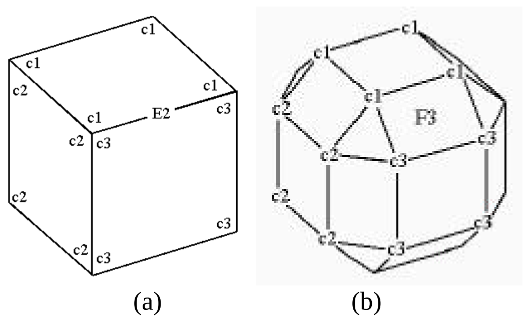


Figure 18: After one refinement, E2 will form F3.

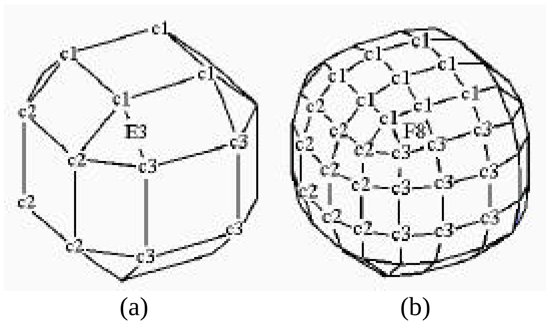


Figure 19: After one refinement, E3 will form F8.

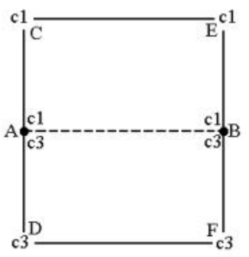


Figure 20: Splitting of new E-face into two parts.

As illustrated in Figure 21, for the new vertices a, b, c, d, e, and f, texture coordinates s_0', s_1' and s_2' can be derived directly. Then, the computing for s_3' is as follows:

$$s_3' = s_0' + \|u_0', u_3'\| / \|u_0, u_3\| (s_3 - s_0).$$

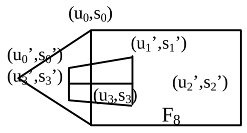


Figure 21: Computing diagram for s_3'.

3.3 Texture mapping on V-faces

V-face will not fall into a region with two texture maps for a closed object. If it falls into a region with more than two texture maps, e.g., three texture maps c1, c2, and c3 for F5 as shown in Figure 22, to compute the texture coordinates of the new vertices, we need to split the new v-face into three parts as shown in Figure 23. The same subdivision is applied for each part using the related texture map.

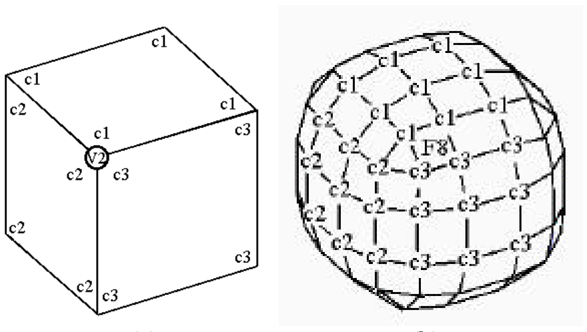


Figure 22: After one refinement, V2 will form F5.

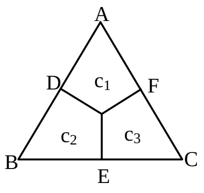


Figure 23: Splitting of new V-face into three parts.

s_2 is computed from F-faces (Figure 24). s_1 and s_3 are computed from E-faces. Hence we need to find out s_0.

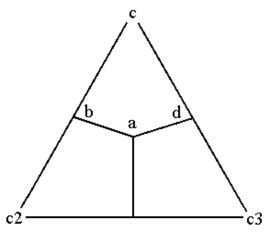


Figure 24: Computing diagram for s_2, where a refers to s_0, b refers to s_1, c refers to s_2, and d refers to s_3.

The number of multiple-textures faces is a consistent for any number of subdivisions, which is equal to the number of vertices in the initial starting mesh. For example, the cube has 8 vertices initially. After a subdivision refinement, the number of multiple-textured faces is 8. After the second refinement, the number of multiple-textured faces is still 8. With this property, we can store the initial texture coordinates s_0, t_0 and w_0 in Figure 25 (a). These coordinates will be used as the center coordinates for multiple-texture face as shown in the figure 25 (b). Hence we can use this value to compute s_0'.

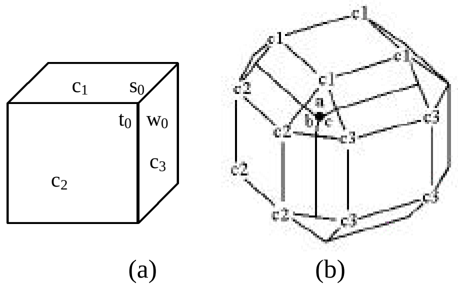
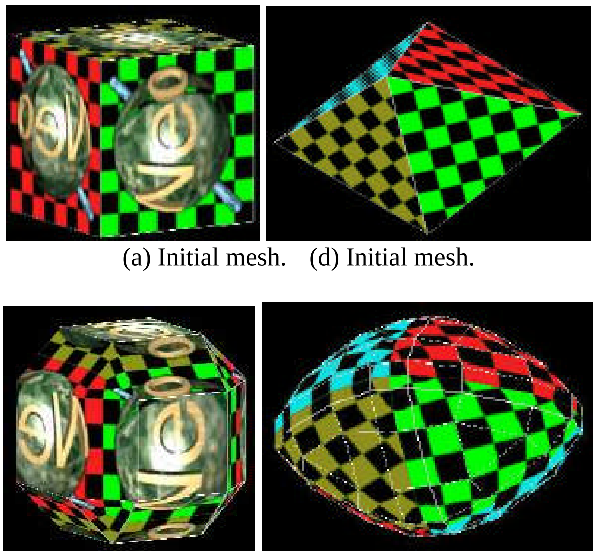


Figure 25: (a) Initial texture coordinates for initial mesh. (b) Center texture coordinates for multiple texture mapped face, where a refers to s_0, b refers to t_0, and c refers to w_0.

4. Implementation and Results

The coding is done on a Pentium III PC running on Windows NT using MS Visual C++ 6.0 and OpenGL libraries. We have implemented the texture-mapping algorithm for Doo-Sabin subdivision surfaces described in Section 3.

From examples shown in Figure 26 to 28, each mesh gets smoother after each refinement. We can see that the textures are continuous across the boundaries of the control meshes after each refinement. We show our method also works for texture mapping using only one texture map (Figure 29).



(a) Initial mesh. (d) Initial mesh.

(b) After one refinement. (e) After two refinements.

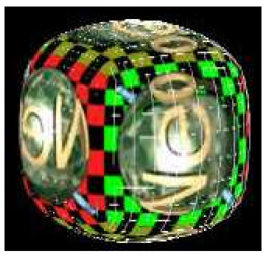
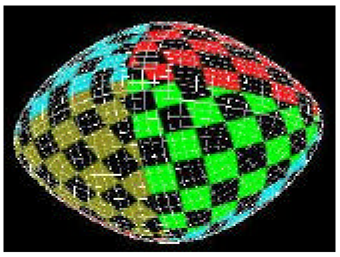
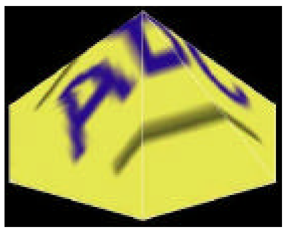
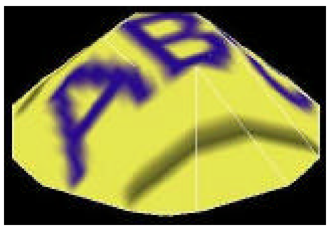
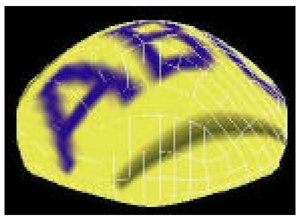


(c) After two refinements. (f) After four refinements.

Figure 26: Six texture maps are used for (a)-(c) and eight are used (d)-(f).


(a) Initial mesh.


(b) After two refinements.


(c) After four refinements.

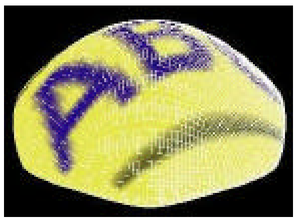
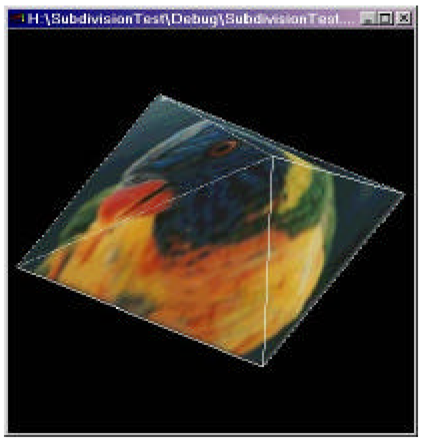
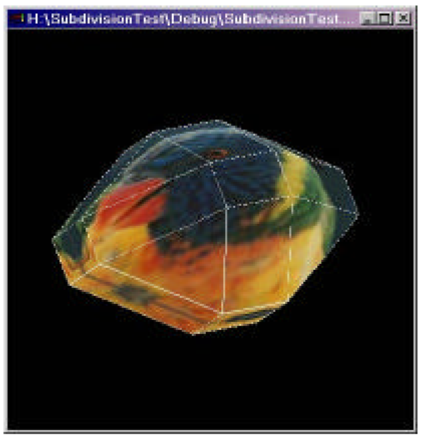

(d) After six refinements.

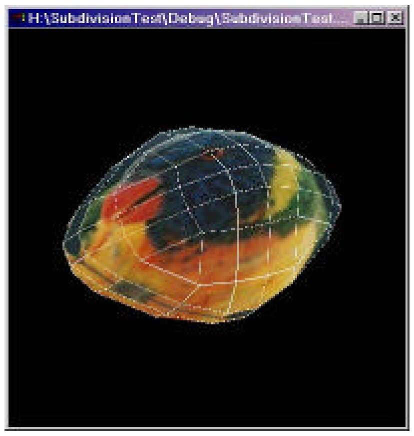
Figure 27: Refinement done on a hexagon-based pyramid. Six texture maps are used.



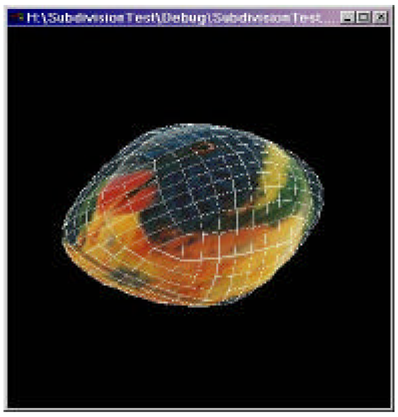
(a) Initial mesh: one image is cut to five parts and mapped to each face respectively.



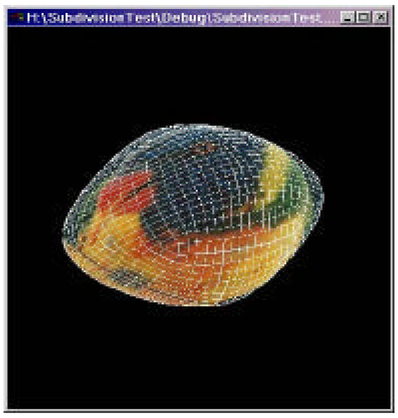
(b) After one refinement.



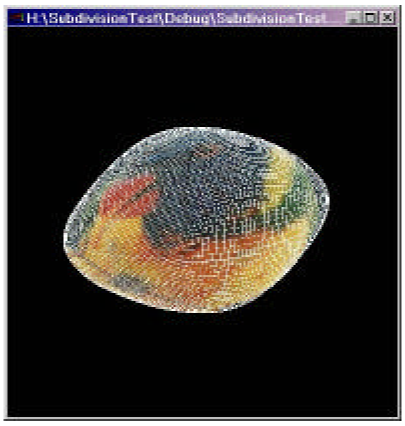
(c) After two refinements.



(d) After three refinements.

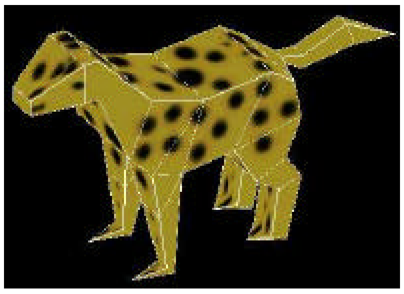


(e) After four refinements.

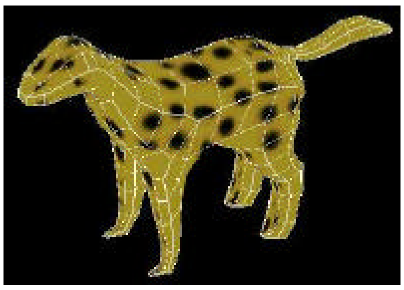


(f) After six refinements.

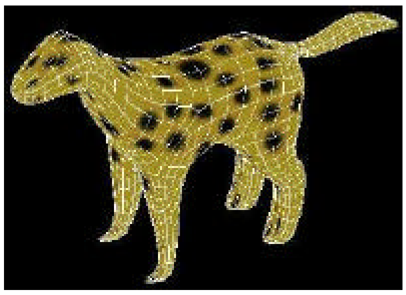
Figure 28: Use a natural image but cut it into 5 pieces to map on the initial control mesh.



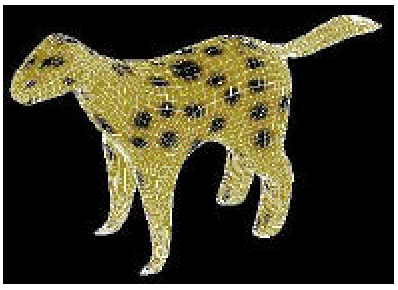
(a) Initial mesh.



(b) After one refinement.



(c) After two refinements.



(d) After three refinements.

Figure 29: Refinement done on an animal. Only one texture map is used.

The split algorithm is integrated with the subdivision process. The time complexity of each split is constant time because only a limited number of new faces will be created as described in the treatments of F-face, E-face and V-face described in the previous section. Thus, the time complexity of texture mapping algorithm is same as that of the subdivision algorithm. Similarly, we can get the same conclusion for the space complexity. For the example shown in Figure 4.4, it took 0.001sec, 0.023sec, and 0.107sec for the three times subdivision without applying texture mapping. It took 0.003sec, 0.026sec, and 0.127sec with texture mapping. It confirmed our analysis, so did other examples.

Our method of further splitting will not change the nature of the subdivision scheme. From operation's point of view, this is an extension to Doo-Sabin (and other dual based subdivision schemes). This is not required for primal subdivision since faces do not get shifted around with the dual operation. Our solution has arrived at introducing another split when the dual is computed hindering the smoothing algorithm. However, introducing the new split on specific polygons will not affect the nature of the sub-division because the new control mesh faces resulted from splitting are contained and aligned with the original ones. Their union is the original one.

5. Summary

We have proposed and implemented a method of texture mapping on Doo-Sabin subdivision surfaces using multiple images based on the idea of the further splitting. We have shown and discussed the results of our method.

6. Acknowledgement

We would like to thank the reviewers of Eurographics 2002 Short Presentations for the comments. We have updated the paper accordingly. This work was done when Chee Seng Neo was studying as an undergraduate student in Department of Computer Science, School of Computing, NUS. It was partly supported by NUS research grant R252-000-051-112.

7. References

[1] E. Catmull and J. Clark. Recursively Generated B-spline Surfaces On Arbitrary Topological Meshes. Computer-Aided Design, 10:350-355, 1978.

[2] T. DeRose, M. Kass, and T. Truong. Subdivision Surfaces in Character Animation. Computer Graphics (SIGGRAPH '98 Proceedings)(1998), 85-94.

[3] D. Doo and M. Sabin. Behaviour Of Recursive Division Surfaces Near Extraordinary Points. Computer-Aided Design, 10:356-360, 1978.

[4] N. Dyn, D. Levin, and J. A. Gregory. A Butterfly Subdivision scheme for Surface Interpolation with Tension Control. ACM Trans. Gr. 9, 2 (April 1990), 160169.

[5] D. Hearn and M. P. Baker. Computer Graphics. Second Edition. Prentice Hall, Inc. 1994. 553-560.

[6] C. Loop. Smooth Subdivision Surfaces Based On Triangles. Master's thesis, University of Utah, Dept. of Mathematics, 1987.

[7] J. Warren. Subdivision methods for geometric design. Pre-print: http://www.cs.rice.edu/~jwarren/.