Adaptive Tesselation of Subdivision Surfaces

Volker Settgast Kerstin Müller Christoph Fünfzig * Dieter Fellner
Institute of ComputerGraphics, Technical University of Braunschweig
Mühlenpfordtstrasse 23
38106 Braunschweig, Germany

Abstract

For a variety of reasons subdivision surfaces have developed into a prominent member of the family of freeform shapes. Based on a standard polygonal mesh a modeller can build various kinds of shapes using an arbitrary topology and special geometrical features like creases. However, the interactive display of subdivision surfaces in current scenegraph systems based on static levels of detail is unpractical, because of the exponentially increasing number of polygons during the subdivision steps. Therefore, an adaptive algorithm choosing only the necessary quads and triangles is required to obtain high-quality images at high frame rates. In this paper we present a rendering algorithm which dynamically adapts to static surface properties like curvature as well as to view-dependent properties like silhouette location and projection size. Without modifying the base mesh, the method works patchwise and tesselates each patch recursively using a new data structure, called *slate*. Besides these geometric properties the algorithm can also adapt to the graphics load in order to achieve a desired frame rate in the scenegraph system OpenSG.

Key words: Subdivision surfaces, Adaptive tesselation, Frame rate control, Virtual reality

1. Introduction

Subdivision surfaces and their interactive rendering have become an important issue in computer graphics. Particularly because of their flexibility and robustness, e.g., supporting arbitrary topology and special geometrical features like creases, subdivision surfaces are now part of well established modelling packages like Maya and 3DStudio Max. However, the interactive display of subdivision surfaces poses hard problems because of the exponentially increasing number of polygons during the subdivision steps. Thus, an adaptive algorithm, choosing only the necessary quads and triangles, is required to obtain high-quality images at high frame rates.

There are some proposals for such algorithms in the literature which can basically be grouped into two classes. The first class evaluates basis functions for each valence and configuration of special features (11). The second class employs the recursive nature of the subdivision schemes in the algorithm (9; 8; 12). Furtheron, a hardware solution was proposed in (13).

This work aims at an adaptive tesselation algo-

* Corresponding author
Email address: c.fuenfzig@cg.cs.tu-bs.de
(Christoph Fünfzig).

rithm which recursively subdivides a patch. By its nature the algorithm is easy to implement, local in its memory references and does not require any precomputation. Subdivision rules for special features can be integrated without changing the algorithm. During execution the base mesh is left unmodified, which is an essential feature for its use in a scenegraph system.

2. Preliminary Remarks

The following presentation is based on the Catmull-Clark (4) subdivision scheme. Nevertheless, the technique can easily be transfered to other subdivision schemes based on triangle or quad meshes.

We use capital letters to denote faces and patches. Normalisation of a vector is written as $\|v\|$. We assume that normal vectors are always normalised. The superscript describes the subdivision depth, e.g., cp_n^l, whereas the subscript indicates the enumeration.

2.1. *Subdivision Surfaces*

A subdivision surface is defined by a *control mesh* M^0 (see Fig. 1). By applying the subdivision rules (10) to the mesh M^0, we get a finer mesh M^1. The sequence of control meshes converges against the *limit surface* $L(M^0)$. The vertices of a control mesh M^l are called *control points* cp_i^l. The *1-neighbourhood* of cp_i^l includes each control point lying on an edge incident on cp_i^l. The *1-neighbourhood of a face* $F^l = \square cp_1^l cp_2^l cp_3^l cp_4^l$ includes all faces adjacent to any vertex of F^l and is denoted by $N(F^l)$. The *edge neighbours* of F^l are the faces having a shared edge with F^l. A *limit point* is defined as $lp_i := \lim_{l\to\infty} cp_i^l$. The *limit normal* ln_i denotes the surface normal of $L(M^l)$ at the point lp_i. The rules to calculate the limit points and the limit normals can be found in (14; 15). The 1-neighbourhood of a face, $N(F^l)$, which is a part of M^l, contains the control points $cp_1^l, \ldots, cp_n^l$. The limit surface $L(N(F^l)) = L(cp_1^l, \ldots, cp_n^l)$ is called the *patch corresponding to* F^l. This patch is a part of the limit surface $L(M^l)$. $Q(cp_1^l, \ldots, cp_n^l) = Q(N(F^l)) = \square lp_1 lp_2 lp_3 lp_4$ is the *chord quadrangle* of F^l.

3. Tesselation on the fly with two Slates

In order to utilise computing power and memory in an optimal way, a reasonable subdivision depth is assigned to each face of the base mesh (Section 3.1). Then each face of the mesh together with its 1-neighbourhood is fetched from the mesh. The subdivision is done down to the assigned depth and the proper limit points and normals are computed (Section 3.2). If the edge neighbours of the respective base face have a higher subdivision depth, the tesselation at the border is adapted accordingly in order to avoid gaps (Section 3.3). The drawing of the resulting shape chunks is described in detail in Section 4.

Because we use a statically sized data structure, we must choose a maximum subdivision depth $maxsd$ that can be applied by our tesselation algorithm. Setting $maxsd = 5$ is sufficient for common models. $maxval$ is the maximum valence of a vertex that can occur in a mesh. In practical cases the valences are smaller than 50.

3.1. *Finding the Subdivision Depth*

For high quality images at an acceptable frame rate the following parameters are taken into account to assign the appropriate subdivision depth to each patch:

- curvature
- visibility
- membership to the silhouette
- projected size of the patch

The curvature is considered only once for a new or changed base mesh. All other parameters are evaluated frame by frame using fast approximations (instead of exact computations) to find a suitable subdivision depth for each patch.

As an approximation for the curvature we use the normal cone technique (5) for each vertex. Consider vertex cp_0^0 with valence val, its limit normal ln and the normals n_i^l of the neighbour faces $i = 1, .., val$ at subdivision depth l. We place a normal cone around ln with a given aperture angle α. Then we search for the minimal l, such that

$$\forall i = 1, .., val: \quad ln \cdot n_i^l < \cos\alpha$$

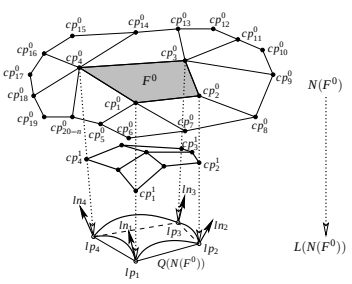

Fig. 1. Example Catmull-Clark subdivision

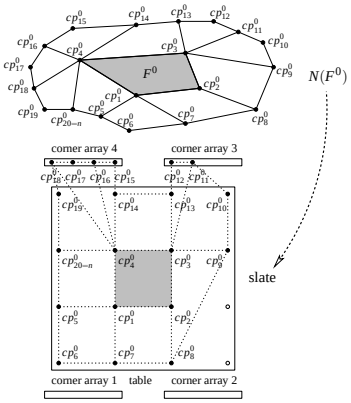

Fig. 2. Collecting the 1-neighbourhood of F from the base mesh into a slate.

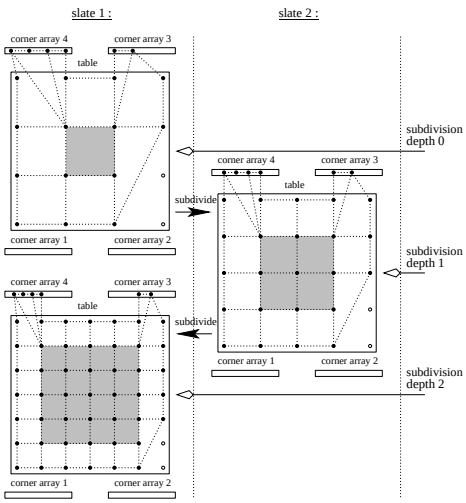

Fig. 3. Subdivision process with slates.

For this subdivision depth l all normal vectors of the neighbour faces are inside the normal cone. So l is a subdivision depth guaranteeing a maximum curvature in the considered vertex by an user defined $NormalConeAperture = \alpha$.

The following steps are done once per frame. First, we classify(6) each vertex cp^0 of the base mesh with limit point lp and limit normal ln to be of type

$$\begin{array}{ll} \text{back vertex} & : \quad \varepsilon < ln \cdot camray \\ \text{front vertex} & : -\varepsilon > ln \cdot camray \\ \text{silhouette vertex} & : -\varepsilon \leq ln \cdot camray \leq \varepsilon \end{array}$$

with $camray = \|lp - cameraposition\|$ and a user defined $VertexClassifier = \varepsilon$. Afterwards, we traverse each face F^0 of the base mesh and decide if F^0 belongs to

$$\begin{array}{ll} \text{back} & : \text{all its vertices are back vertices,} \\ \text{front} & : \text{all its vertices are front vertices,} \\ \text{silhouette} & : \text{otherwise} \end{array}$$

Back faces will not be subdivided at all. Front faces are assigned a subdivision depth equal to the maximum of their vertex depths, assigned by curvature. Silhouette faces obtain a subdivision depth equal to the average of $maxsd$ and the maximum of their vertex depths, assigned by curvature.

Finally, we estimate the projected size of each front face and each silhouette face and adapt their subdivision depths, respectively. This ensures that no unnecessary refinement will be done and prevents the presentation from being too coarse, e.g., in highlighted areas. We use a bounding sphere around the considered patch and estimate the radius of the projected sphere onto the image plane. For user defined $ProjectedSizeMin$ and $ProjectedSizeMax$, the maximum and minimum radius R_{min} and R_{max} of the desired subpatches are computed dependent on the distance to the camera, the field-of-view of the camera and the image resolution. If the radius R according to subdivision depth l, that means $R \cdot 1/2^l$, is greater than R_{max}, s further refinements are neccessary until $R \cdot 1/2^{l+s} \leq R_{max}$. We obtain $l+s$ as the new subdivision depth. If in contrast $R \cdot 1/2^l < R_{min}$, then a smaller subdivision depth is sufficient. We search for the minimum s such that $R \cdot 1/2^{l-s} \geq R_{min}$ and obtain $l - s$ as the new subdivision depth. In the case of $R_{max} \geq R \cdot 1/2^l \geq R_{min}$ no change of the depth is necessary.

For frame rate control we measure the time between two frame update calls. This time is compared to a given time window we want to achieve in the current frame (*Feedback Mechanism*(16)).

Starting with the tesselation depths assigned from geometric properties, patches are moved to the neighbouring lower depth for decreasing detail and moved to the neighbouring higher depth for increasing detail, respectively. We always start by adjusting the maximum depth $maxsd$ for rapid adjustment. The amount of adjustment is decreased as the frame rate gets closer to the target frame rate.

3.2. *Subdivision Work with Slates*

For each face of the base mesh, we now tesselate the corresponding patch with the chosen subdivision depth. Thereby the base mesh is not modified, instead we use a separate data structure consisting of two slates. A *slate* is composed of a two-dimensional array *table* of size $(2^{maxsd}+3)^2$ and four one-dimensional arrays $corner_{1,2,3,4}$ of size $(maxval-4)\cdot 2$. The slates are allocated statically and can be reused for each face that has to be tesselated. Therefore two of this slates are sufficient to compute all subdivisions, and the total number of 3D vectors necessary is $2\cdot(2^{maxsd}+3)^2+16\cdot maxval-64$.

First, the 1-neighbourhood of the considered face F^0 is collected into the slate (see Fig. 2). The vertices of F^0 and the vertices of its edge neighbour faces are stored in the table. If one of the vertices of F^0 has $valence > 4$, then the remaining vertices of $N(F^0)$ are stored in the dedicated corner arrays.

The algorithm *tesselate* starts with the chosen subdivision depth and slate 1 as arguments (see Fig. 3). The total running time is linear in the number of limit points (3).

```
tesselate (subdivision depth, slate i)
  if subdivision depth > 0 then
    perform one subdivision step and
    save the new points in slate j = (i + 1) mod 2;
    tesselate (subdivision depth − 1, slate j);
  else
    compute limit points and normals
    from slate i;
  end if;
```

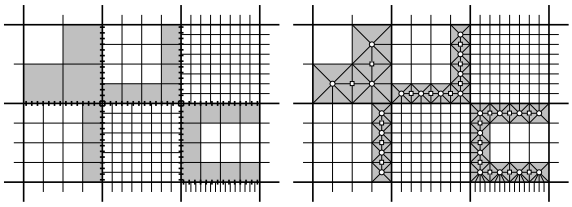

Fig. 4. Schematical presentation of 6 patches at different subdivision depths: With gaps (left image) and without gaps (right image).

3.3. *Gap prevention*

Due to the fact that we use an adaptive presentation of the surface, neighbour patches can have different subdivision depths. An example is shown in Figure 4, six patches with different depths are illustrated schematically. Gaps can occur between the differently tesselated patches depicted as dashed lines. To avoid these gaps, the tesselation of the patches with lower subdivision depth must be modified. For this purpose, we replace the chord quadrangles lying at the border to a further subdivided patch (grey marked in Fig. 4) with suitable triangle fans.

Such a triangle fan is built from the limit point of the new face point of the next subdivision step (marked by a small circle in Fig. 4), the vertices of the quads and the points lying at the border to the deeper tesselated neighbour. If neighbour quadrangles of the same patch are replaced like this, the shared edge is subdivided and the resulting limit points (marked by a small box in Fig. 4) are added to the triangle fan. The additional subdivision step is essential to prevent undesired sharp bends and buckles on the tesselated surface as illustrated in Figure 10.

4. Rendering with OpenSG

A scenegraph node in OpenSG (1) consists of the tree structure in the *Node* class and the functionality in the *NodeCore* fieldcontainer. In order to execute the adaptive algorithm in OpenSG, we need a derived NodeCore class *DynamicSubdivisionCC*, which contains the base mesh (2), the tesselator object and the user parameters NormalConeAperture, ProjectedSizeMin/Max, VertexClassifier.

Every time the base mesh is altered, the tesselator object collects the curvature information (Section 3.1). The Position and Normal fields are resized to a static size sufficient for the maximum subdivision depth *maxsd*. These fields cache the limit points and limit normals between frames. They can be reused as long as the subdivision depth is not increased. The Indices, Types and Lengths fields have to be updated every frame. The inner part of the surface is built of quad strips only. For the outer part a mixture of quad strips and triangle fans is used as described in Section 3.3.

5. Results

To show the effect of the backface culling heuristic, we use another camera and have a look at the back parts (see Fig. 8). The reasonable assignment of subdivision depths is visible in the wireframe in Figure 11. Gap prevention is done between the differently tesselated patches.

For our tests we picked a scene with 40 instances of one model with 68 base faces (see Fig. 12). The camera performs several rotations around the y–axis, facing the scene center. All benchmarks were performed on a standard PC with Intel P4 1.7 GHz, 512 MB, GeForce 4 TI 4600.

Figure 5 provides the resulting triangle counts for the adaptive and uniform tesselation. In Figure 6 it is detailed how the patches without gap prevention are composed of quads at depth 0 to 4. In contrast, the uniform tesselation takes all triangles at a given fixed depth, including the backfacing ones.

Figure 7 shows the frame rates which can be achieved for a uniform subdivision depth of 2, 3 and 4 of approximately 31, 17 and 7 fps. In comparison the adaptive tesselation with dynamic frame rate control allows to switch from a given frame rate of $14 - 15$ fps to $39 - 40$ fps within 70 frames.

6. Conclusion

We have shown that an adaptive tesselation algorithm using two statically sized slates performs very good in practice. A classification scheme detects visually important parts of the surface for additional refinements to obtain high quality images. A frame rate control is also easy to integrate.

The extension to other kinds of subdivision surfaces based on quad meshes just requires the implementation of the subdivision rules. For subdivision schemes using a triangle mesh (e.g. Loop (7)), an additional pairing of triangles into quadrangles is necessary to work with the slates.

In conclusion, the algorithm is simple and efficient, easy to implement and also easy to extend to other subdivision rules. Therefore it is a reasonable choice for rendering subdivision surfaces in a scenegraph system like OpenSG.

This work has been supported by the OpenSG PLUS project funded by the German Federal Ministry of Education and Science (bmb+f) under grant 01 IRA 02G and by the ModNav3D project funded by the German Research Council (DFG) under grant Fe-431/4-3.

References

[1] Dirk Reiners, Gerrit Voß and Johannes Behr. OpenSG – Basic Concepts. *Proc. OpenSG Symposium 2002*

[2] Mario Botsch, Stephan Steinberg, Stephan Bischoff and Leif Kobbelt. OpenMesh – a generic and efficient polygon mesh data structure. *Proc. OpenSG Symposium 2002*

[3] Volker Settgast, Kerstin Müller, Christoph Fünfzig and Dieter Fellner. Adaptive Tesselation of Subdivision Surfaces in OpenSG. *Proc. OpenSG Symposium 2003*

[4] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, **10**(6):350–355, 1978.

[5] Leon A. Shirman and Salim S. Abi-Ezzi. The Cone of Normals Technique for Fast Processing of Curved Patches. *EUROGRAPHICS 93 Conf. Proc.*, **12**(3):261–272, 1993.

[6] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. *SIGGRAPH 97 Conf. Proc.*, pp. 199–208, 1997

[7] C. Loop. Smooth subdivision surfaces based on triangles. *Master thesis, University of Utah*, 1987.

[8] K. Müller and S. Havemann. Subdivision Surface Tesselation on the Fly using a versatile Mesh Data Structure. *Computer Graphics Forum (Eurographics 2000 Proc.)*, **19**(3):151–159, 2000.

[9] K. Pulli and M. Segal. Fast Rendering of Subdivision Surfaces. *Eurographics Rendering Workshop*, pp. 61–70, 1996.

[10] D. Zorin and P. Schröder. Subdivision for modeling and animation. *SIGGRAPH 99 Course Notes*, 1999.

[11] J. Bolz and P. Schröder. Rapid Evaluation of Catmull-Clark Subdivision Surfaces. *Web3D'02 Conf. Proc.*, pp. 11–17, 2002.

[12] S. Havemann. Interactive rendering of Catmull/Clark surfaces with crease edges. *The Visual Computer*, **18**:286–298, 2002.

[13] S. Bischoff, L. Kobbelt and H. P. Seidel. Towards Hardware Implementation of Loop Subdivision. *Eurographics Workshop on Graphics Hardware*, pp. 41–50, 2000.

[14] H. Biermann, A. Levin and D. Zorin. Piecewise Smooth Subdivision Surfaces with Normal Control. *SIGGRAPH 2000 Conf. Proc.*, pp. 113–120, 2000

[15] H. Halstead, M. Kass and T. DeRose. Efficient, Fair Interpolation using Catmull–Clark Surfaces. *SIGGRAPH 93 Conf. Proc.*, pp. 35–44, 1993

[16] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments. *SIGGRAPH 93 Conf. Proc.*, pp. 247–254, 1993.

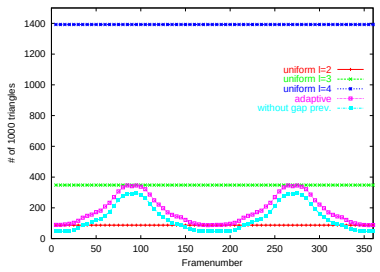

Fig. 5. Resulting triangle counts for adaptive and uniform tesselation of testscene I.

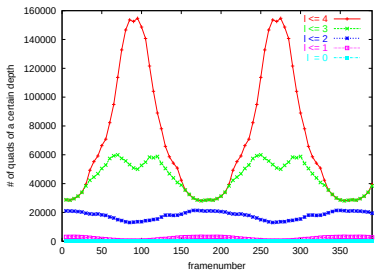

Fig. 6. Distribution of the resulting quadrangles (without gap prevention) to certain subdivision depths for testscene I.

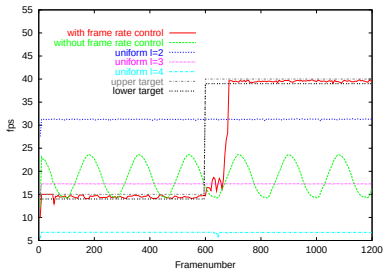

Fig. 7. Frame rates for testscene I with and without framerate control. The adaptive algorithm uses depths 0 – 4.

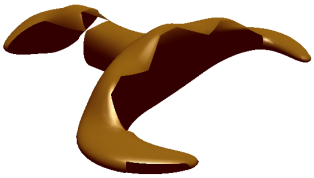

Fig. 8. Effect of the backface culling heuristic: The camera is in front of the spaceship.

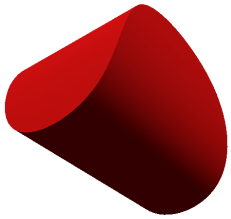

Fig. 9. Subdivision Surface with 6 base faces (box) and 8 sharp edges.

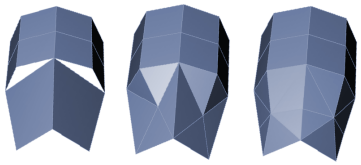

Fig. 10. Different gap filling strategies in an example of four patches. The simple method (middle image) creates some visible buckles, which are prevented by our method (right image).

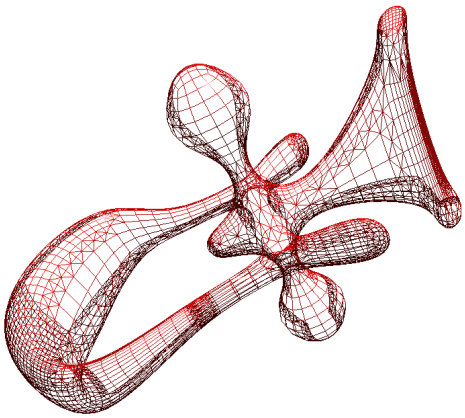

Fig. 11. Wireframe view to show the adaptive tesselation.

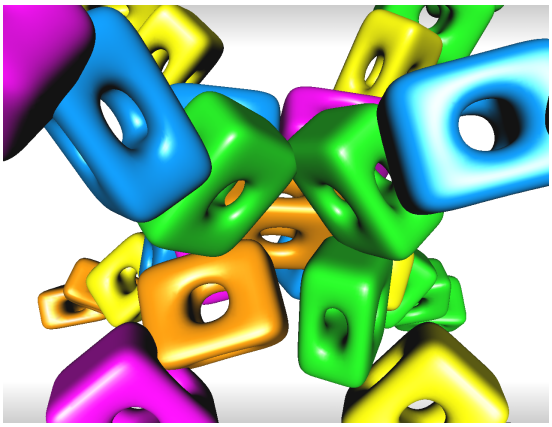

Fig. 12. Testscene I for frame rate (40 instances with 68 base faces each using depths 0 – 4).