Smooth Subdivision Surfaces over Multiple Meshes

Przemyslaw Musialski
Bauhaus-University Weimar
Weimar, Germany
musialsk@uni-weimar.de

Robert F. Tobler
VRVis Research Center
Vienna, Austria
rft@vrvis.at

Stefan Maierhofer
VRVis Research Center
Vienna, Austria
sm@vrvis.at

ABSTRACT

Standard subdivision rules, such as Catmull-Clark and Loop allow the creation of smooth surfaces with C^2 continuity over almost the whole domain except at extraordinary vertices. Normally, the subdivision schemes are limited to one mesh and need special rules for handling the boundaries of the domain. This issue leads to complications in straightforward approaches to compose objects out of multiple joined meshes. We propose a new method for stitching control meshes at common faces. The new approach uses the stitching information to smoothly subdivide the meshes across the stitching edges, while maintaining the meshes as separate units in memory. This makes it possible to compose large, complex geometries using simple components, without the necessity to subdivide the complete mesh down to the same detail level.

Keywords: subdivision surfaces, level of detail, multiple meshes

1 INTRODUCTION

Complex objects are difficult to model and render with conventional, manual modeling techniques. In order to facilitate such tasks, different methods for modeling and representing objects have been developed over the years in Computer Graphics. Using the common boundary representation in 3D space, one possibility to reduce complexity in more sophisticated shapes is to create objects composed of several simpler pieces.

Combining this idea with the subdivision surface approach results in a method for generation of smooth and complex geometry over multiple polygonal meshes. Since these objects can be defined through piecewise simple, primitive geometric elements, procedural generation of complex objects by rule based composition of the basic shapes is possible [17].

In this paper we present a method for subdividing multiple, loosely joined meshes, which we call *stitched meshes*. The complete set of all stitched meshes serves as the control domain for further subdivision to achieve smooth geometric shapes. By using the standard Catmull-Clark scheme as described by DeRose [3] we are able to create meshes with C^2 continuity over almost the complete surface and C^1 continuity at extraordinary vertices respectively.

The advantage of this method is the fact that problem regions such as the boundaries of single meshes do not have to be treated separately by using modified subdivision rules but can be handled with the standard rules for interior regions. At the same time all partial meshes of the common surface remain independent of each other, and must not be subdivided to the same level.

In section 2 we will to give a brief overview of subdivision surfaces and point out the main problems that arise when multiple subdivided surfaces need to be joined. In sections 3 and 4 we will describe the generalized subdivision and our method based on joining multiple meshes on the coarsest subdivision level. Here we will introduce *stitch faces*, which serve as a connection between different meshes and discuss the *subdivision to different levels* as a level–of–detail approach. Finally we will present some results of our method applied on models composed of several meshes.

2 RELATED WORK

Mesh subdivision is a technique for generating smooth surfaces that has been introduced quite some time ago by Catmull and Clark [2] and Doo and Sabin [4]. For a long time the theoretical foundation of the subdivision process was not as thorough as for other modeling techniques such as BSplines and the more general NURBS, and thus it took a while for subdivision methods to become widely known and used. Recently this has been rectified by the introduction of methods to analyse and evaluate subdivision surfaces at any point [12],[14], a method for extending subdivision surfaces for emulating NURBS [13], the addition of normal control to subdivision surfaces [1], and a method to closely approximate Catmull-Clark subdivision surfaces using BSpline patches [11]. A number of other extensions to subdivision surfaces [3], [7], [8], [5], [6] have established them as the modeling tool of choice for generating topologically complex, smooth surfaces.

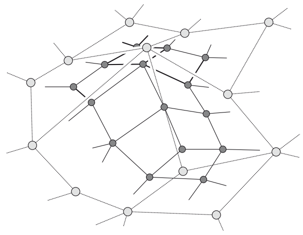


Figure 1: A step in the Catmull-Clark subdivision scheme.

The main goal of most subdivision surface techniques is the use of recursive refinement to obtain smooth surfaces out of arbitrary polygonal input. One significant disadvantage is their undefined behavior on the boundaries of geometric domains. In order to create complex shapes often several meshes have to be joined at their borders to gain satisfying visual results with sufficient complexity. Here, often continuity problems arise since smoothly joining the meshes at the boundaries is a non-trivial task. To overcome these drawbacks Biermann *et al.* [1] introduced a method for controlling the boundaries and their normals in order to join independent meshes, but his approach involves again different rules for special cases.

Our approach uses standard Catmull-Clark rules which are applied to multiple meshes, by joining the meshes to a virtual single mesh.

3 GENERALIZED SUBDIVISION SURFACES

The standard subdivision process starts out with a mesh $M^{(0)}$ composed of vertices, edges, and faces that serves as the base for a sequence of refined meshes $M^{(0)}, M^{(1)}, M^{(2)}, \ldots$ which converges to a limit surface, called the subdivision surface.

The process for generating submesh $M^{(n+1)}$ of a specific mesh $M^{(n)}$ in the sequence can be split up into two operations. The first operation, which we will call *mesh refinement*, is the logical introduction of all the vertices in the submesh. This operation yields all the

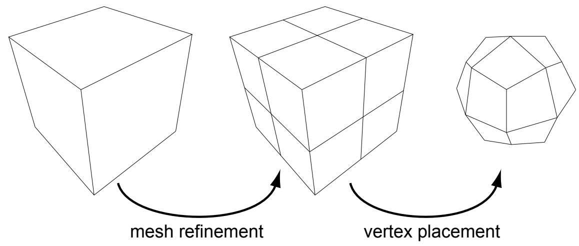


Figure 2: Generalized subdivision.

connectivity information for the vertices of the submesh without specifying the positions of these newly introduced vertices. The second operation, which we will call *vertex placement*, is the computation of the actual vertex positions. Standard subdivision schemes use specific rules for generating the new vertex positions, that ensure that the limit surface of the subdivision process satisfies certain continuity constraints, e.g. C^1 or C^2 continuity.

To obtain maximum flexibility in generating subdivision surfaces, we propose to separate the two operations of *mesh refinement* and *vertex placement*, and make it possible to independently specify both of these operations.

4 SUBDIVISION OVER MULTIPLE MESHES

General subdivision rules ensure C^2 continuity between interior mesh polygons and C^1 at extraordinary vertices. At the boundary of a mesh different subdivision rules have to be applied to achieve considerable smoothness. To connect two meshes at their borders, both pieces have to have exactly the same border definition. Unfortunately, depending on the chosen subdivision scheme, it is not always straightforward to do as mentioned in section 2.

To handle these problems we suggest to stitch meshes at selected faces (referred as *stitch faces*) prior to the subdivision process with the effect that standard interior subdivision rules can be applied at the borders to a neighbor mesh. Furthermore, since the connection between the meshes is established basically on the logical level, only the *vertex placement* step has to be applied over multiple meshes — the topological subdivision proceeds independently.

To achieve the connection, only small amounts of information about the neighbors have to be held in each mesh additionally. Since control meshes are rather sparse and easy to control, common borders between two can be constructed quite easily. Moreover, due to this simplicity, the stitch faces can be either user selected or automatically generated by routines to construct complex geometry as proposed by Tobler *et al.* in [17].

Nevertheless, each of the partial meshes remain independent from any higher located common control surface and can be rendered, edited or subdivided separately.

4.1 Stitch Faces

A logical connection between two meshes can be established by defining *stitch faces* (Figure 3). Any of the adjacent meshes has to contain such a face pointing to its neighbor. Corresponding stitch faces have to be identical and tangent to each other; only the vertex order must be reversed. These regions can be seen as shared polygons which contain connection data bidirectionally, such that a way from a mesh to its neighbor and *vice versa* can be found in order to obtain all needed geometric information.

For the connection the following information about the neighbor mesh is kept in each stitch face:

- *reference to the neighbor mesh*
  via the reference the whole structure of the neighbor can be accessed

- *stitch face index in the neighbor mesh*
  with this index the corresponding face within the neighbor mesh can be identified

- *face–vertex–index in the stitched face*
  with this index the offset in the vertex order in the face can be specified, such that geometrically corresponding vertices are matched to each other

Stitch faces can be simple defined by adding the auxiliary information to particular elements i.e. in form of a hashtable, that uses the face-index as a key.

To keep track of the connection, all edges around the stitch face are marked. Whenever a marked edge is encountered during processing of a face, the respective

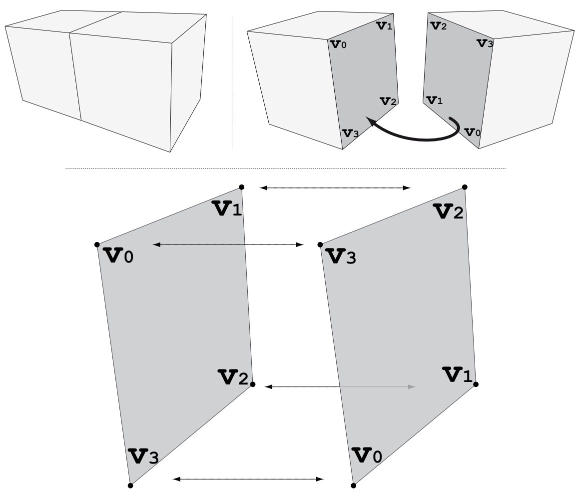


Figure 3: Top: Multiple stitched meshes, Bottom: Corresponding stitch faces

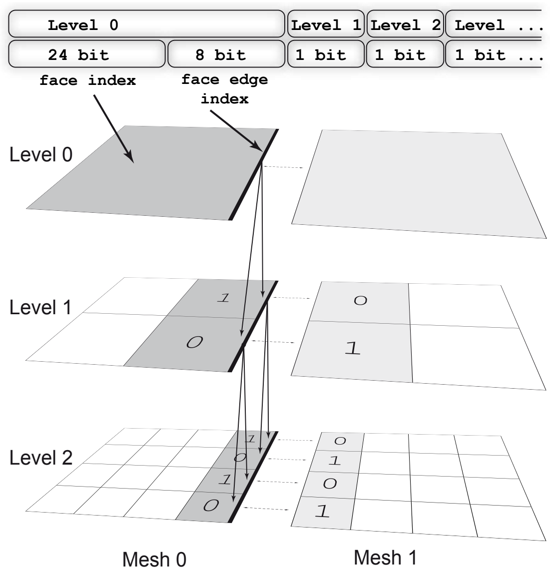


Figure 4: Encoding the subsequent face location path along a stitch edge.

stitch face can be accessed. Through the provided reference to an adjacent mesh and a face–vertex–index, a corresponding face in the neighbor mesh can be exactly detected. Now a standard subdivision rule can be applied at border edges and vertices and the geometric information behind the boundary is retrieved from the neighbor. The same procedure has to be applied on the opposite mesh, which finally produces a continuous surface as if there were no junction, yet both pieces are only loosely connected with a single reference between the control meshes.

4.2 Further Subdivision Steps

Subdivision rules are based on recursive refinement of given initial control meshes. Control meshes in our case contain connection information about each other, but these are not propagated to the following subdivision steps in the same manner. In fact, the topology of the subdivided meshes changes in comparison to its parent, since elements previously defined as stitch faces disappear in the subsequent mesh. The stitch faces have been introduced to simplify modeling the original base mesh and to keep an unambiguous connection between meshes as described in the section above, but they do not serve any geometric purpose. In the subdivided meshes, each stitch face is not present anymore and results in a topological hole. Since each hole fits exactly to a corresponding one in the neighbor mesh, the common surface defined by the subdivision process is continuous.

Due to this fact only the control mesh holds the necessary connection to its neighbors but the successive generated submeshes do not. We overcome this problem by keeping the whole subdivision hierarchy acces-

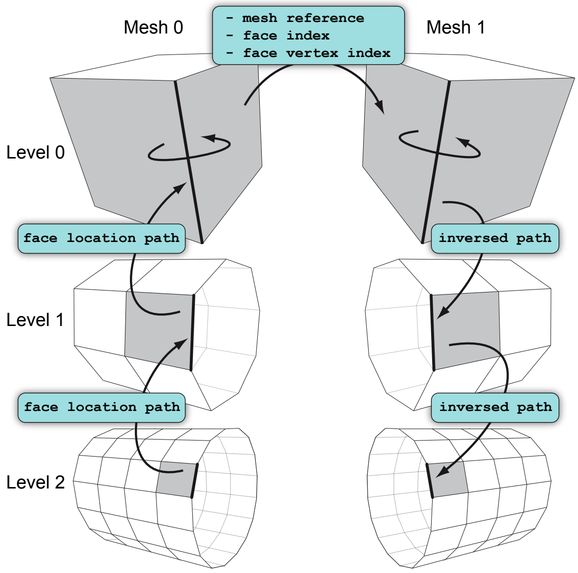


Figure 5: Traversing the subdivision hierarchy along the face location path.

sible in storage and by marking the edges around each hole as *stitch edges* (bold painted edges in Figure 4,5 and 6).

Since we are using Catmull-Clark rules, each quadrangle is successively subdivided into four pieces. In our approach corresponding faces of two stitched meshes touch each other at a single edge. Therefore at each subdivision level a single bit can be used to encode the location of a sub–edge along a parent edge. Concatenating the bits along the subdivision hierarchy, a path to each face along the stitch edge can be defined. The only exception provide the elements in the base level, where the face index and corresponding face–edge–index indicates an non–ambiguous stitch edge. In a memory conservative implementation it is not necessary to store this path, as it can always be computed on the fly. Figure 4 shows how this path is constructed.

For any face along the stitch edge this unique path allows to locate it in the hierarchy and furthermore, because of mirror symmetry, it is easily possible to locate the corresponding face in the subdivision hierarchy of the neighbor mesh by exchanging the level–zero information and simple bitwise inversion of the path as shown in Figures 4 and 5.

It should be mentioned that other subdivision methods, like i.e. Loop [15] can also be fitted in this or similar scheme.

4.3 Recursive Stitched Subdivision

The stitched subdivision method described above performs with two joined objects. Indeed there is no limitation in the number of adjacent elements which could be affected. Since every border face can hold as many unique paths as its edge count, more meshes could be connected to it. The procedure then recursively proceeds by moving from one mesh to the next as shown in Figure 6. In the showed case the top face of mesh M_2 holds the unique paths at the particular edges stitched to the two neighbors M_1 and M_3. In the corner with all three meshes the algorithm retrieves first the stitch face from M_2 to M_3, but since it is again a stitch face it proceeds recursively to next mesh in the same manner. The two stitch faces in M_2 and respectively one per M_1 and M_3 are removed after the first subdivision step.

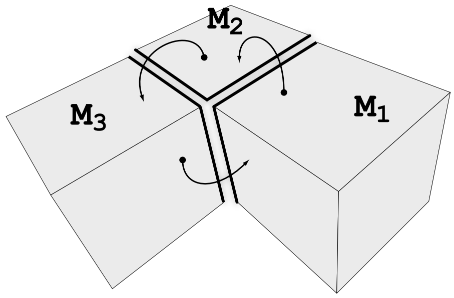


Figure 6: Subdivision over three meshes.

4.4 Subdivision to different levels

As mentioned, due to the two phase nature of our subdivision method, only the vertex placement operation needs access to neighboring meshes in a stitched mesh. As with all subdivision methods, the vertex positions of a new level are computed from the positions of the elements in the previous subdivision level. Thus a subdivision step of a single mesh can be performed, while all neighboring stitched meshes remain not subdivided (see Figure 7). This, of course is possible with only one step difference between adjacent meshes, which is still sufficient for a level–of–detail approach. In a rendering application this can be exploited as follows: only subdivide meshes that are currently viewed in a close up to a certain level, and remove subdivision levels that are not visible or not in the near field anymore (see Figure 15).

To solve the problem of discontinuity at a junction of two (or more) connected meshes at different resolutions, we propose to force the border vertices of the higher sampled mesh to their positions of the lower level. The newly generated edge–vertices in the higher level can be linearly placed in the middle of each edge, which produces T–triangles (see Figure 16, 17). While the meshes are basically joined, indeed, during the rendering small micro holes in the surface can still appear at the T–junctions. This issue can be enforced by introducing zero–area–triangles at the T–junctions similar as proposed in [10].

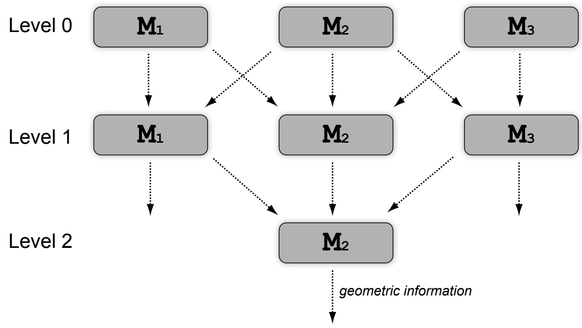


Figure 7: Different subdivision levels and geometric information flow.

5 RESULTS

We have compared the performance of our method on two scenes showing the same model. In the first one we used the model of a triple–cross built out of one coherent mesh. In the second scene we used a model composed of five pieces as shown in Figure 10.

All observed subdivision timings are presented in Figures 8 and 9 and were measured on a Intel Centrino Duo 1.83 Mhz machine with 1GB main memory and 2MB CPU–Cache.

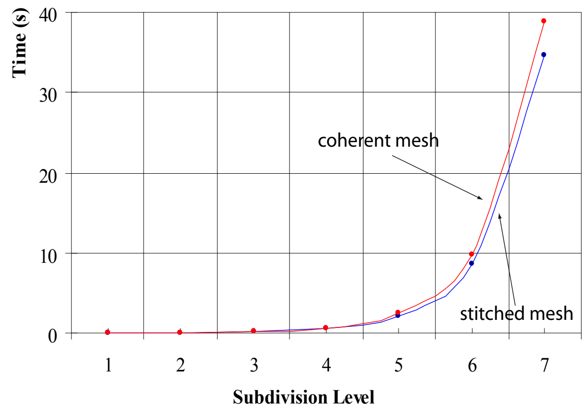
Figure 9 shows the comparison of the run times of both models. One can observe, that our approach has nearly exactly the same performance as the classical one of subdividing a complete mesh. Indeed, the stitched meshes were computed even faster in higher subdivision steps, probably due to cache effects.

	$M_1 \dots M_4$		M_5		M_Σ	
	time(s)	space(kb)	time(s)	space(kb)	time(s)	space(kb)
$M^{(1)}$	0,002	1	0,005	1	0,014	5
$M^{(2)}$	0,007	2	0,013	4	0,042	12
$M^{(3)}$	0,025	8	0,049	16	0,148	48
$M^{(4)}$	0,092	31	0,185	63	0,563	187
$M^{(5)}$	0,362	124	0,725	249	2,183	745
$M^{(6)}$	1,443	494	2,865	989	8,657	2.965
$M^{(7)}$	5,765	1.972	11,518	3.944	34,560	11.832
Σ	7,696	2.633	15,36	5.266	46,167	15.798

Figure 8: Runtime and memory usage of the partial meshes $M_1 \dots M_5$ and their sum M_Σ (see model in Figure 10).

6 CONCLUSION AND FUTURE WORK

We have implemented a new method for stitched meshes, that makes it possible to perform the subdivision of large, complex geometric objects in a piecewise fashion, while retaining the continuity



M_Σ	0,014	0,042	0,148	0,563	2,183	8,657	34,560
M	0,010	0,040	0,150	0,603	2,409	9,703	38,882

Figure 9: Runtime comparison between partial meshes M_Σ and coherent mesh M in seconds.

properties of standard subdivision. Other necessary tasks like i.e. vertex normal interpolation or texture coordinate mapping as well as crease edges handling [3] can be facilitated by this method just as in standard subdivision. This is possible, because at the stitches the access to any necessary neighborhood information takes place just like in a standard mesh.

This new approach has been shown to have no significant performance impact on the actual subdivision process while opening up a number of optimization possibilities for level-of-detail rendering.

Additionally, the new method is optimally suited as a basis for procedural generation of complex objects. In procedural based subdivision, the stitch-faces correspond to symbols which are used in a L-system grammar that builds more complex objects.

We are currently working on a system for vegetation generation, that combines these two approaches in order to generate highly detailed plants models as well as on a GPU–supported implementation of this approach, which will make a real-time subdivision at higher resolutions possible.

ACKNOWLEDGMENTS

Most of this work has been done at the VRVis research center in Vienna, Austria, which is partially founded by the Austrian government research program Kplus. Parts of this work have also been funded by the Austrian Science Fund (FWF) under project P17260-N04.

REFERENCES

[1] Henning Biermann, Adi Levin, and Denis Zorin. Piecewise smooth subdivision surfaces with normal control. In *SIGGRAPH '00: Proceedings of the 27th*

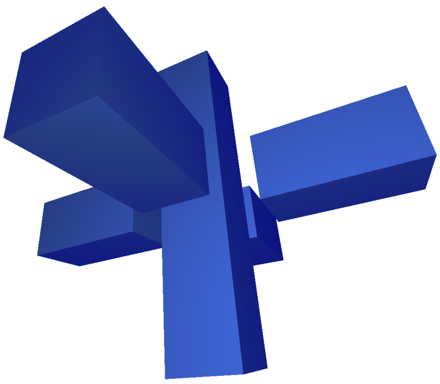


Figure 10: Test model at base level, composed of five meshes. We refer to them as $M_1 \ldots M_4$ to the four small pieces (each composed of six faces), M_5 to the bigger one (composed of 14 faces) and M_Σ to whole object.

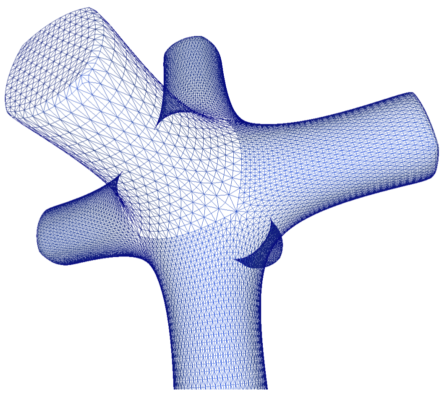


Figure 11: Stitched model from Figure 10 subdivided partially to 4th and 5th level.

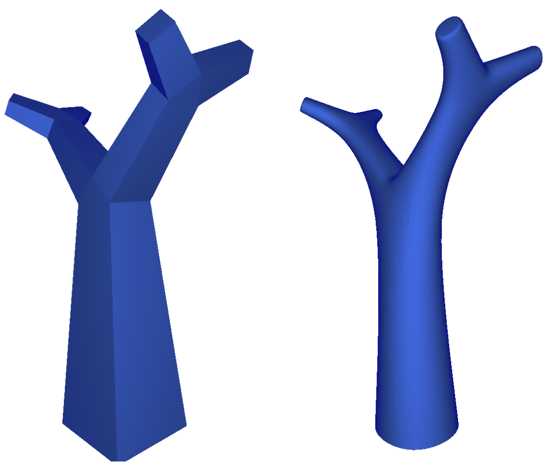


Figure 12: Tree trunk model composed of several simple meshes.

Figure 13: A simple hand model composed of 14 meshes.

Figure 14: Hand model subdivided without stitching information.

annual conference on Computer graphics and interactive techniques, pages 113–120, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

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

[3] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 85–94, New York, NY, USA, 1998. ACM Press.

[4] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10:356–360, September 1978.

[5] Igor Guskov, Wim Sweldens, and Peter Schröder. Multiresolution signal processing for meshes. In Alyn Rockwood, editor, *Proceedings of the Conference on Computer Graphics (Siggraph99)*, pages 325–334, N.Y., August8–13 1999. ACM Press.

[6] Igor Guskov, Kiril Vidimče, Wim Sweldens, and Peter Schröder. Normal meshes. In *SIGGRAPH '00:*

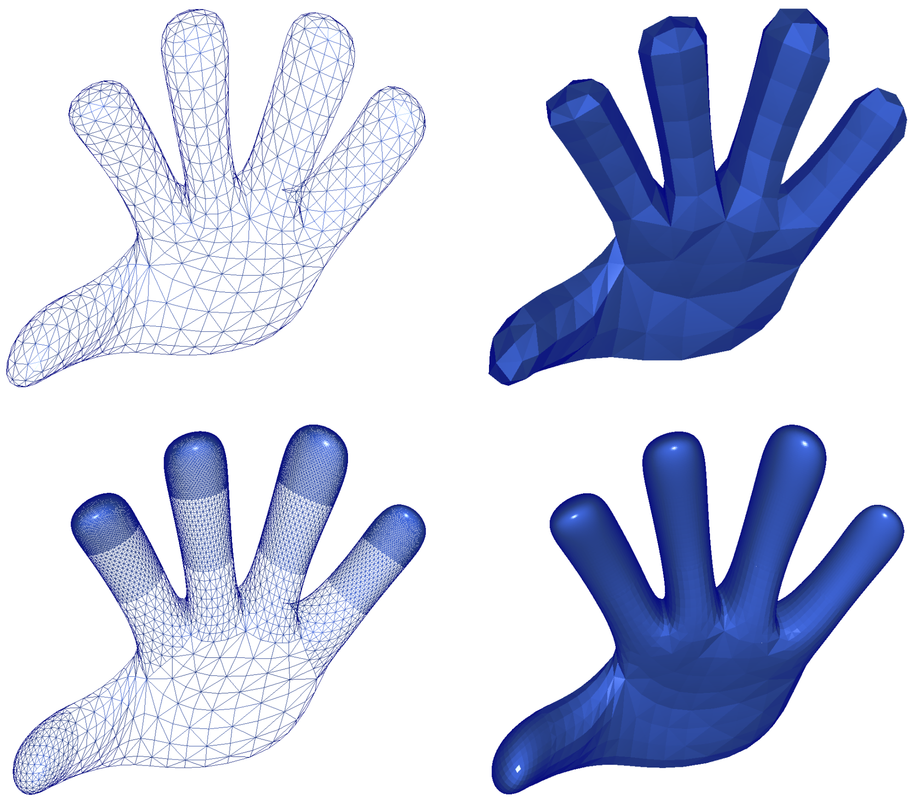


Figure 15: A simple hand model. Top: uniform subdivision. Bottom: multi–resolution subdivision with forced border vertices.

Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pages 95–102, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[7] Leif Kobbelt. $\sqrt{3}$ - subdivision. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 103–112, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[8] Aaron Lee, Henry Moreton, and Hugues Hoppe. Displaced subdivision surfaces. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 85–94, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[9] Charles Loop. Triangle mesh subdivision with bounded curvature and the convex hull property. Technical Report MSR-TR-2001-24, Microsoft Research (MSR), January 2001.

[10] Frank Losasso and Hugues Hoppe. Geometry clipmaps: terrain rendering using nested regular grids. *ACM Trans. Graph.*, 23(3):769–776, 2004.

[11] Jörg Peters. Patching catmull-clark meshes. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 255–258, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[12] Ulrich Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12(2):153–174, 1995.

[13] Thomas W. Sederberg, Jianmin Zheng, David Sewell, and Malcolm Sabin. Non-uniform recursive subdivision surfaces. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 387–394, New York, NY, USA, 1998. ACM Press.

[14] Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 395–404, New York, NY, USA, 1998. ACM Press.

[15] Jos Stam and Charles T. Loop. Quad/triangle subdivision. *Comput. Graph. Forum*, 22(1):79–86, 2003.

[16] Robert F. Tobler and Stefan Maierhofer. A mesh data structure for rendering and subdivision. In *WSCG 2006 - Short communication Proceedings, ISBN 80-86943-05-4*, pages 157–162, 2006.

[17] Robert F. Tobler, Stefan Maierhofer, and Alexander Wilkie. A multiresolution mesh generation approach for procedural definition of complex geometry. In *SMI '02: Proceedings of the Shape Modeling International 2002 (SMI'02)*, page 35, Washington, DC, USA, 2002. IEEE Computer Society.

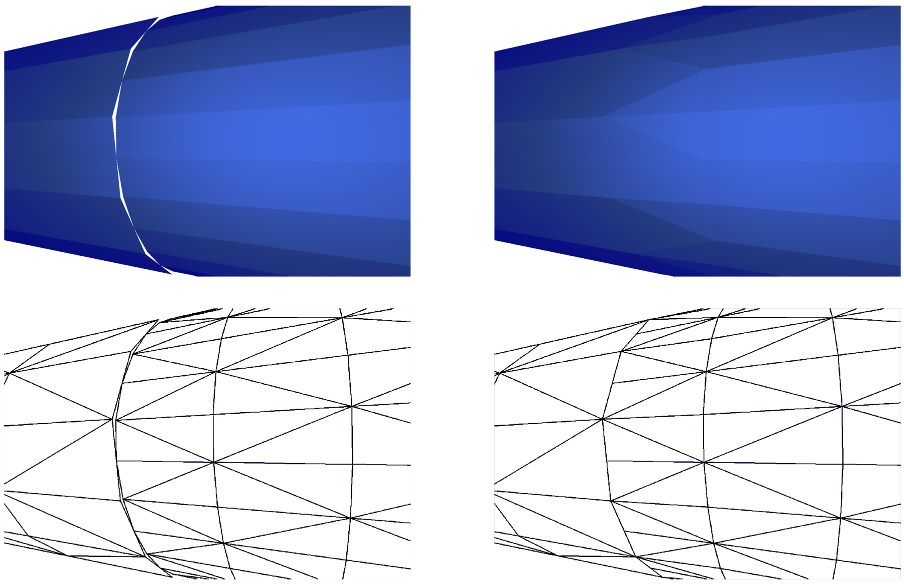


Figure 16: A junction between two meshes at different LODs. Left: without adjustment, Right: with forced border vertices.

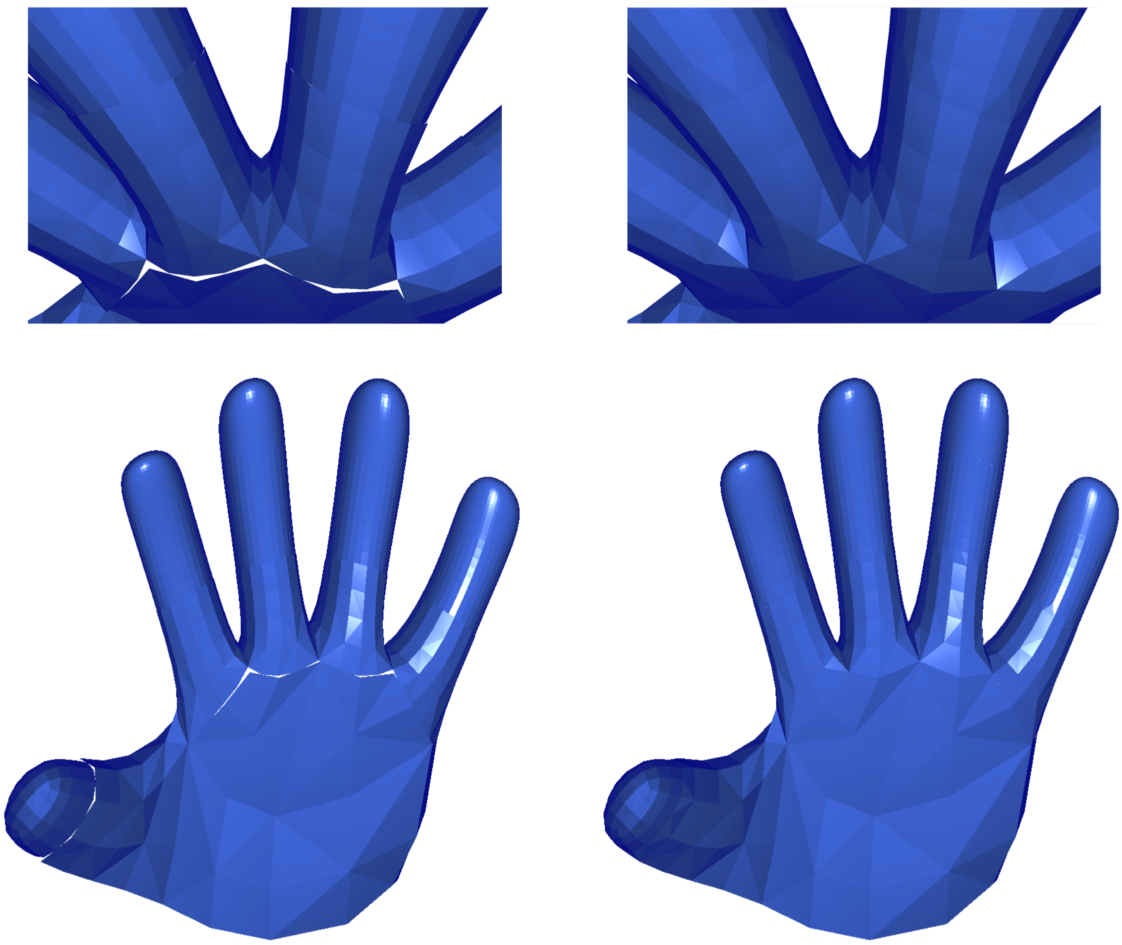


Figure 17: Hand model at different LODs. Left: without junction adjustment, right: with forced border vertices.