Dynamic Progressive Triangle-Quadrilateral Meshes

Stefan Wundrak
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt, Germany
stefan.wundrak@igd.fraunhofer.de

Thomas Henn
Technical University Darmstadt
Fraunhoferstr. 5
64283 Darmstadt, Germany
thomas.henn@gris.informatik.tu-darmstadt.de

André Stork
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt, Germany
andre.stork@igd.fraunhofer.de

ABSTRACT

We present an extension of the original progressive mesh algorithm for large dynamic meshes that contain a mix of triangle and quadrilateral elements. The demand for this extension comes from the visualisation of dynamic finite element simulations, such as car crashes or metal sheet punch operations. These methods use meshes, which consist mainly of quadrilaterals, due to their increased numerical stability during the simulation. Furthermore, these meshes have a dynamic geometry with about 25 to 100 animation steps. Accordingly, we extend the original progressive mesh algorithm in two aspects: First, the edge collapse operation is extended for meshes with a mixture of triangle and quadrilateral elements. Second, we present an algorithm on how to extend quadric error metrics for the simplification of large dynamic meshes with many animation steps. The results are dynamic progressive triangle-quadrilateral meshes – a progressive multi-resolution mesh structure that has two interactive degrees of freedom: simulation time and mesh resolution. We show that our method works on meshes with up to one million vertices and 25 animation steps. We measure the approximation error and compare the results to other algorithms.

Keywords

Progressive Meshes, Level of Detail, Crash Simulation, Animation, Quadrilaterals, Dynamic Meshes.

1. INTRODUCTION

For dynamic finite element simulations in structural mechanics, such as car crashes or sheet metal forming, analysts prefer quadrilateral meshes over triangular meshes (Figure 1), since three-noded constant strain triangles behave poorly in bending [MG97]. The results of these simulations are meshes with a constant mesh topology, which contain between 1 and 5 million vertices, and a dynamic geometry with about 25 to 100 animation steps. In addition, during the optimization process hundreds of variants are simulated and stored, leading to huge amounts of data. Tools to visualise the three-dimensional simulation results help the engineer to interpret the crash behaviour and to optimise the car body. These tools have to efficiently deal with large time dependent data sets [Som03]. Even though for finite element simulations the model has to be highly tessellated over the complete mesh, during post-processing analysis, data base browsing, or interpolation of simulation results, it is often sufficient to display the animated mesh at a reduced granularity first and refine the animation on demand only. Furthermore, only the mesh sections with a high deformation need to be displayed at full resolution. This approach reduces the amount of data that needs to be transmitted and displayed. The base for these methods will be a progressive data structure that supports dynamic triangle-quadrilateral meshes.

Short Communications proceedings, ISBN 80-86943-05-4
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.

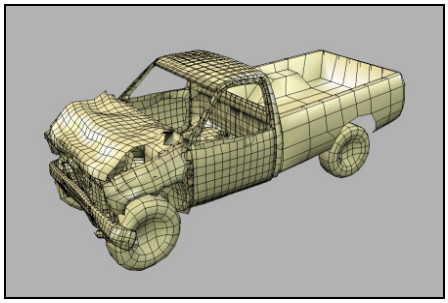


Figure 1. Example of a small triangle-quadrilateral mesh used for crash simulations.

Goals and Contribution

We will show how Dynamic Progressive Triangle-Quadrilateral Meshes can easily be generated out of common dynamic meshes as a pre-processing step to visualisation.

Accordingly, we extend the original progressive mesh algorithm of Hoppe [Hop96] in two aspects: First, the edge collapse and vertex split operations are extended for meshes with a mixture of triangle and quadrilateral elements. Therefore, a new constraint is added to avoid the creation of degenerated meshes. Second, we present an algorithm that extends quadric error metrics [GH97] for the simplification of large dynamic meshes with many animation steps. The result is a progressive multi-resolution mesh structure that has two interactive degrees of freedom: simulation time and mesh resolution. We show that our method works on meshes with up to one million vertices and 25 animation steps, and compare the approximation error to other algorithms.

Related Work

Progressive Meshes, first introduced by Hoppe [Hop96], are a progressive data structure for triangular meshes based upon the *edge collapse* operation. By means of this operation it is possible to reduce the complexity of a given triangle mesh by iteratively removing edges and thus deleting the adjacent triangles resulting in the base mesh M^0:

$$M^n \rightarrow (\text{ecol}_{n-1}) \rightarrow \ldots \rightarrow (\text{ecol}_1) \rightarrow M^1 \rightarrow (\text{ecol}_0) \rightarrow M^0$$

The inverse *vertex split* operation allows undoing these changes and restoring the removed mesh items. This allows for storage and transmission of the original mesh M^n as multi-resolution representation consisting of M^0 and a sequence of n vertex split operations:

$$M^0 \rightarrow (\text{vsp}_0) \rightarrow M^1 \rightarrow (\text{vsp}_1) \rightarrow \ldots \rightarrow (\text{vsp}_{n-1}) \rightarrow M^n$$

Ramsey et al. [RBH03] describe an extension to the edge collapse operation for meshes composed of non-planar multi-sided polygons. However, extending the necessary preconditions for legal collapse operation was not covered.

Gumhold [Gum04] introduced the *remove edge* and *join edges* operations to simplify arbitrary polygonal meshes (Face Clustering). The remove edge operation joins two faces by removing its shared edge and thus creates polygons of higher complexity.

To create the sequence of operations from the original mesh one has to define an error metric. Selecting the error measurement is a trade-off between the quality of the simplified mesh and the performance of the simplification process. For a discussion of the different methods see [PS97].

Garland and Heckbert [GH97] introduced an error metric based on *quadrics* that accumulates the error during simplification. It approximates the maximum error of the geometric distance and the deviation of surface normals. A quadric is assigned to each vertex of the mesh, which represents a weighted combination of all faces adjacent to the vertex. To estimate the approximation error of an edge collapse operation the quadrics of the two adjacent vertices are added and evaluated. This leads to a fast algorithm with low memory needs and a good approximation quality. In [GH98] a generalization of the quadric error metric is presented that also considers surface properties, such as colours, texture coordinates, or surface normals. We will extend this approach in this work to support dynamic meshes with many animation steps.

In [KEH97] Kuschfeldt has described how to convert quadrilateral meshes into triangles and how to display the original element boundaries using texture mapping. In combination with progressive meshes this method has several disadvantages. First, inherent information about the element boundaries is lost and has to be stored additionally, which increases the amount of data and complicates the method. Second, storing twice as many triangles increases the connectivity data in many representations. Thus, we decided to include quadrilaterals as basic elements into the progressive mesh algorithm which leads to a simple and elegant solution.

2. PROGRESSIVE TRIANGLE-QUADRILATERAL MESHES

Our aim is to extend progressive meshes to support a mix of triangle and quadrilateral elements. We start with some definitions related to meshes that will be used throughout the paper.

A polygonal mesh M can be denoted by a tuple (V, F), where $V = \{v_0,\ldots,v_n\}$ is a set of vertex positions defining the shape of the mesh in R^3, and $F = \{f_0,\ldots,f_i\}$ is a set of closed faces. The set of edges is denoted by $E = \{e_0,..,e_k\}$. An animation with a constant topology is defined by m sets of vertex positions $A = \{V_0,\ldots,V_{m-1}\}$.

An edge is a *boundary edge* if it is part of only one face. A vertex is a *boundary vertex* if it is part of a boundary edge. A vertex is an *inner vertex* if it is not a boundary vertex.

For the representation of a legal surface, polygonal meshes have to fulfil the following conditions (cp. [Gum04] [FDF+90] [BSBK02]):

(M1) *The mesh is manifold with boundary.*
(M2) *The minimum valence of an inner vertex is three.*
(M3) *The minimum degree of a face is three.*

Our work is based on the OpenMesh library [BSBK02] that internally works with a half-edge data structure. Non-manifold meshes are initially converted to manifold meshes.

Extending the Collapse and Split Operation for Mixed Meshes

Figure 2 shows the half-edge collapse operation used for triangle meshes collapsing the edge $\{v_0; v_1\}$ into the vertex v_1. The adjacent faces f_l and f_r degenerate and vanish in this process. Also the vertex v_0 is removed from the mesh.

An extension to the half-edge collapse is the edge collapse operation, which allows optimal placement of the remaining vertex v_1, but will not be used in this work. Although the usage of the edge collapse operation usually increases the quality of the decimated mesh it also increases the amount of data that has to be stored in a progressive mesh as we will describe later.

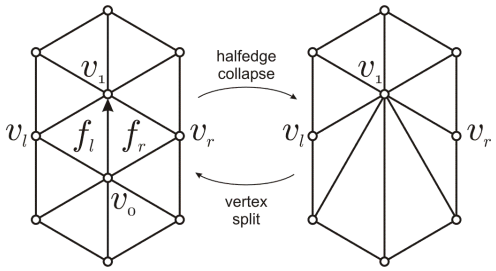


Figure 2. Half-edge collapse and vertex split operation in triangle mesh.

Figure 3 shows the new extended half-edge collapse operation for quadrilateral meshes. In this case, collapsing the edge $\{v_0; v_1\}$ into the vertex v_1 does not lead to degenerated faces. Instead, the quadrilaterals are transformed into triangles. Only the vertex v_0 is removed.

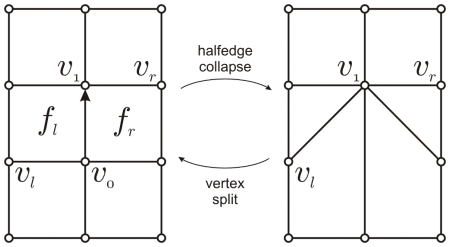


Figure 3. Half-edge collapse in quad mesh.

How these half-edge collapse and vertex split operations are stored in a data structure is described in section 4.

Legal Collapses for Triangle Edges

Depending on the topology of the mesh an edge collapse operation may produce a degenerated mesh. To avoid this one has to validate legal operations using a topology test. Hoppe et al. introduced three preconditions in order to collapse an edge $\{v_0; v_1\} \in E$ [HDD+93].

Definition: *Two vertices v_i and v_j are neighbours, if an edge $\{v_0; v_1\} \in E$ exists.*

(P1) *For each vertex $v_i \in V$, that is a neighbour of v_0 and v_1, v_i shares a face with v_0 and v_1.*
(P2) *If v_0 and v_1 are both boundary vertices, $\{v_0; v_1\}$ is a boundary edge.*
(P3) *M has more than 4 vertices if neither v_0 nor v_1 are boundary vertices, or M has more than 3 vertices if either v_0 or v_1 are boundary vertices.*

While (P2) avoids the separation of M into two meshes that are only connected by a single vertex, (P3) terminates the simplification. Precondition (P1) avoids the creation of vertices with a valence of two, which are only allowed at the boundary of the mesh. Figure 4 shows a half-edge collapse operation that violates the first precondition.

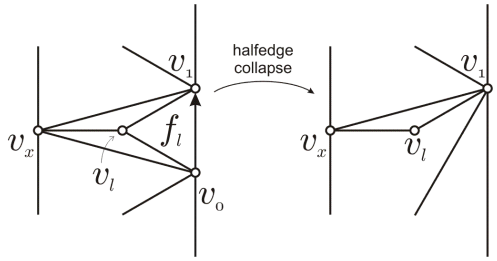


Figure 4. Illegal half-edge collapse operation that creates a vertex with a valence of two. This case is avoided, because (P1) is violated by v_x.

Legal Collapses for Quadrilateral Edges

For mixed meshes containing triangles and quadrilaterals the precondition (P1) is not sufficient and has to be extended. It is necessary to differ between edges that are adjacent to a triangle or to quadrilaterals only. We define that a *triangle edge* is adjacent to at least one triangle, whereas a *quadrilateral edge* is adjacent to quadrilaterals only.

The collapse of a quadrilateral edge needs no further tests. As seen in Figure 3, the valence of each involved vertex is unchanged or increased. The valence of the adjacent faces is decreased by one, since the quadrilaterals are turned into triangles. Only precondition (P2) is necessary in this case.

On the other hand, the collapse of a triangle edge within a quadrilateral mesh needs further attention. Similar to an edge collapse in a triangle mesh a vertex with a valence of two might be created.

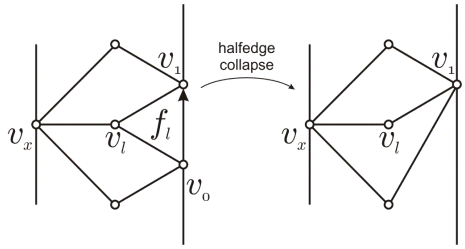


Figure 5. Creation of a vertex with a valence of two in a quadrilateral mesh. This case is not avoided, because (P1) is not violated by v_x.

In Figure 5 the shown half-edge collapse operation leads to a vertex with a valence of two, even though precondition (P1) is not violated. Thus, precondition (P1) has to be extended in order to avoid this case.

Definition: *Two vertices v_i and v_j are neighbours of second order, if a face $f \in F$ exist for that $v_i \in f$ and $v_j \in f$ holds true.*

Corollary: *Two vertices v_i and v_j that are neighbours are also neighbours of second order.*

Precondition (P1) can be reformulated as follows:

(P1*) *For each vertex $v_i \in V$, that is a neighbour of second order of v_0 and v_1, v_i shares one face with v_0 and v_1.*

Precondition (P3) for termination is not changed, since successive collapses of quadrilaterals always end up in triangles. The more costly precondition (P1*) is only needed for quadrilateral meshes. A proof for the extended preconditions is given in the Section 7.

Extending the Quadric Error Metric for Mixed Meshes

As described in [GH97] fundamental quadrics represent the triangles of the original mesh. Each fundamental quadric is defined by a plane, which in turn is defined by a triangle of the mesh. Each quadric is based on a set of these fundamental quadrics. However, in a quadrilateral mesh the fundamental quadrics are only well-defined for planar quadrilaterals. For skew quadrilaterals with non-coplanar vertices a plane has to be defined that represents the quadrilateral best. We propose to compute the plane as follows:

To define the plane a normal vector n and a distance d are needed. At first, a normal vector is computed for each vertex of the quadrilateral (cp. [RBH03]). Then n is set to the mean of these normals, and d is defined as the centre of gravity of the quadrilateral multiplied by n. This leads to a fundamental plane that has the same distance to all vertices of the quadrilateral.

3. LARGE DYNAMIC PROGRESSIVE MESHES WITH MULTI QUADRICS

In [GH98] a generalization of the quadric error metric was presented that considers surface properties, such as colours, texture coordinates or surface normals. This approach can be extended to support animated meshes with fixed topology, since the characteristic property of a dynamic model is the changed geometry during each animation step.

There are two principle approaches for the extension of quadrics to consider multiple properties (e.g. A and B) during simplification [GH98]:

(1) The generation of multi quadrics Q_A and Q_B for each vertex and the definition of the error as $Q_A(v_A) + Q_B(v_B)$.

(2) The generation of one higher dimensional quadric Q_{AB} for each vertex and the definition of the error as $Q_{AB}(v_{AB})$.

While in method (1) the memory and computation costs rise linearly with the number of attributes, method (2) shows a quadratic behaviour.

In the case of dynamic progressive meshes the number of attributes corresponds to the number of animation steps. A mesh with 25 animation steps would thus lead to a factor of $25^2 = 625$ in computation time and memory consumption compared to the static mesh. This disqualifies method (2) for our work. Using multi quadrics instead of higher dimensional quadrics means however, that optimal vertex placement is not easily possible.

In contrast to the simplification of polygonal meshes with colour or texture attributes a dynamic mesh with m animation steps has additional geometry data in form of m positions for each vertex. The summation of the error values in method (1) however doesn't fit the needs of dynamic meshes very well. Thus, we choose a modified method (1*) that better preserves the geometric variation over time:

(1*) The generation of multi quadrics Q_A and Q_B for each vertex and the definition of the error as $max(Q_A(v_A) , Q_B(v_B))$.

The m animation steps can be interpreted as m distinct meshes $\{(V_0,F),...,(V_{m-1},F)\}$. For each vertex v_i, m quadrics $Q_{t,i}$ will be generated and initialised using the geometry of the associated animation step (V_t,F). During simplification the meshes of the animation steps can be processed in a parallel manner. Thus, an edge collapse will be rated considering the $2*m$ associated quadrics.

The collapse of an edge (v_i,v_j) into vertex v_k needs m quadric additions $Q_{t,k} = Q_{t,i} + Q_{t,j}$, generating m

quadrics, one per animation step. The overall approximation error for this operation is defined by the maximum of these *m* quadrics:

$$\max_{t=0..m}\left(Q_{t,k}(v_k)\right)$$

For large models with many animation steps it is possible to further optimise the error metric. In the case of crash results, one can notice that the geometric difference between two successive animation steps is relatively small. Therefore, it is possible to take into account only selected animation steps. For crash simulations we achieved good results by selecting the first, the last, and one animation step in between, preferably the one with the maximum vertex displacements. A generalization to arbitrary animations would be to automatically detect the animation steps with the strongest deformation. Restriction to a small number of animation steps allows for reduced computation effort. Measurements are discussed in section 5.

4. DATA STRUCTURE

As described in [Hop96] progressive meshes are stored in two parts. First the small base mesh, second a stream of *n* vertex split operations, which are the inverse of the *n* edge collapse operations generated during the simplification process. This structure may then be used for Level-of-Detail or progressive streaming of the mesh.

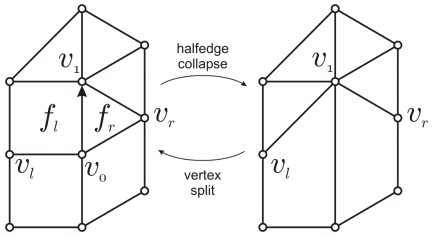


Figure 6. The vertex split operation as inverse operation of edge collapse allows for the refinement of the base mesh.

As shown in Figure 2 for a pure triangle mesh it is sufficient to know the references of the vertices v_1; v_l; v_r as well as the position of the vertex v_0. If optimal vertex placement was used during simplification the new position of v_1 has also to be stored. If the vertex has additional properties, such as colour or displacement vectors, one additionally has to store the properties of the vertex v_0. The references to v_l and v_r indicate the triangles f_l and f_r.

To define a vertex split operation within a mixed mesh one also has to save whether the half-edge collapse operation did remove any triangles. For example Figure 6 shows the half-edge collapse removing triangle f_r. Thus, the inverse operation of the half-edge collapse operation $hec(v_0;v_1)$ is well-defined by the vertex split operation $vsp(p_0; v_1; v_l; v_r; t_l; t_r)$ with the following parameters:

v_1; v_l; v_r	*vertex reference*
p_0	*position of removed vertex* v_0
t_l; t_r	*bool, true if* f_l / f_r *triangle*

5. RESULTS

Evaluation

To evaluate the approximation error of a simplified mesh we compare it to the original mesh. We used Metro [CRS96] to measure the two-sided Hausdorff distance. Tools such as Metro do not offer the comparison of dynamic simplified meshes. As a workaround we compared the single animation steps manually and calculated an overall approximation error as average of the all animation steps.

In addition, Metro is only able to compare triangle meshes, thus, before measuring, the mixed meshes had to be triangulated. To avoid approximation errors due to non well-defined triangulation we used the star-shaped triangulation as described in [Gum04] that uses the centre of gravity as a new vertex position.

In the following we express the two-sided Hausdorff distance in percentage of the diagonal of the meshes bounding box. Each simplification was computed using a PC with a Pentium 4 processor with 2.0 GHz and 1GB of RAM running Windows 2000.

Results Static Mixed Meshes

First, we test our algorithm against QSlim, an implementation of the quadric error metric using edge collapse simplification with optimal vertex placement as described in [GH97].

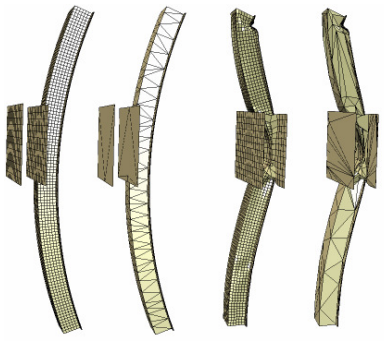


Figure 7. The pillar mesh at animation step t0 and t15 and in resolutions of 100% and 5% of original vertices.

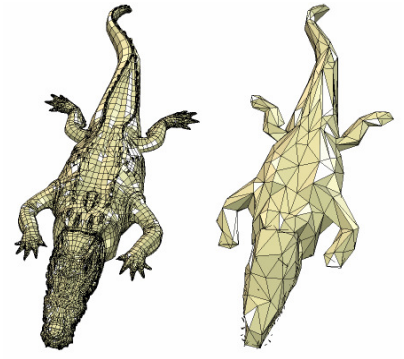


Figure 8. The croco mesh in resolutions of 100% and 5% of original vertices.

We use two different animation steps (t0, t15) of a technical model (Figure 7) and a non-technical model (Figure 8) to evaluate the quality of the simplification of our algorithm compared to QSlim. The results are shown in Table 1.

Mesh	Tris	Quads	5%	5% QSlim	Time [sec]
pillar.t0	0	6190	0,08	0,03	0,7
pillar.t15	0	6190	0,68	0,68	0,7
croco	9358	12523	1,78	1,75	2,1

Table 1. Results static meshes.

The results are generally comparable to QSlim, even though no optimal vertex placement algorithm was used. One thing to notice is that for the non-deformed mesh (pillar.t0) the approximation error is significantly lower than to the deformed mesh (pillar.t15). Meshes prepared for finite element analysis are generally tessellated in a high resolution, even for plane faces. For an undeformed mesh these triangles can be eliminated easily without increasing the approximation error. For deformed meshes however, these triangles are needed to represent the deformed geometry with the needed accuracy. This explains the difference between the two pillar meshes.

Results Dynamic Mixed Meshes

In a second experiment we measure the quality of our algorithm for dynamic meshes. We used the front part of the large neon mesh (Figure 9) as a subset, since this part is deformed most. The Metro tool does not support dynamic meshes, thus we compared the single animation steps for the neon.front mesh as if they were static meshes and calculated an overall approximation error afterwards. The results are shown in Table 2. Above 4% of the original vertices, the maximum overall approximation error falls below 1%.

Percent of vertices	t0 [%]	t12 [%]	t24 [%]	Max. t0...t24
1%	2,06	1,63	1,52	2,06
2%	1,21	1,35	1,30	1,35
4%	0,89	0,78	0,80	0,89
11%	0,26	0,27	0,27	0,27

Table 2. Approximation error for dynamic mesh neon.front, at various resolutions and animation steps.

Figure 9 shows the resulting meshes subject to the two degrees of freedom in mesh resolution and simulation time.

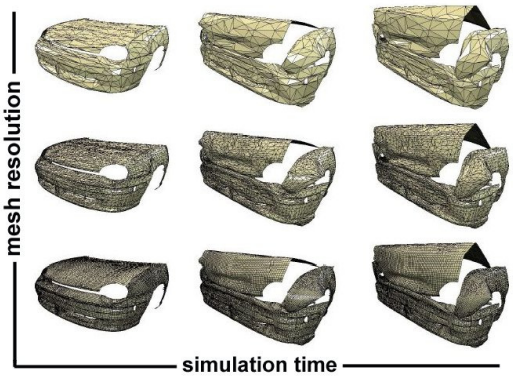


Figure 9. This picture shows the two degrees of freedom, with resolution 4%, 18%, and 100% and animation steps t0, t6, and t12.

One thing to notice is that for dynamic meshes the use of quadric error metrics leads to a higher mesh resolution for the areas with a high deformation.

Our last experiment analyses if it is sufficient for the error metric to consider only a subset of animation steps for meshes that are typically produced by crash simulations, in order to save computation time during the simplification process.

Considered Animation Steps	1%	4%	6%	Time [sec]
All	1,60	0,73	0,38	36:52
t0,t6,t12	1,67	0,68	0,34	3:33
t0	3,92	1,97	1,24	2:16
t12	1,80	0,93	0,57	2:16

Table 3. Mean approximation errors for dynamic mesh neon.front.

Table 3 shows a summary of the results. Including only a subset of three animation steps leads to nearly

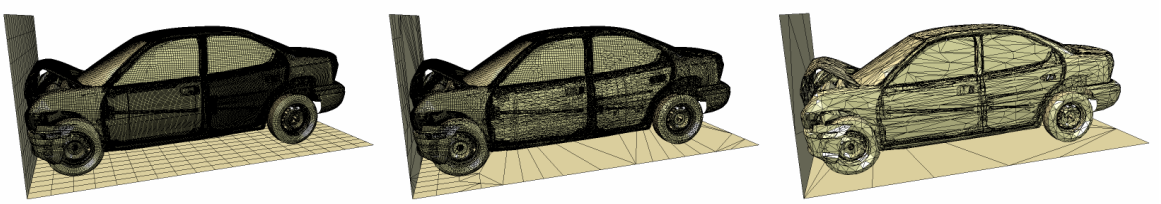


Figure 10. The complete neon mesh at resolutions 100%, 50%, and 8% of the original vertices.

no increase of the approximation error but significantly reduces the computation time for the simplification. Using only one animation step is only acceptable if the animation step with the maximum deformation (t12) is chosen.

Finally, Figure 10 shows the complete finite element mesh of a neon car body in various resolutions. The mesh contains more than 1 million vertices and 25 animation steps. The second mesh shows the mix of triangular and quadrilateral elements.

6. SUMMARY AND CONCLUSION

We have extended the original progressive mesh algorithm in two aspects: First, to support meshes with a mixture of triangle and quadrilateral elements. Second, we defined an error metric for the simplification of large dynamic meshes with many animation steps.

Together we have created a multi-resolution mesh structure that has two interactive degrees of freedom: simulation time and mesh resolution. In future work extensions towards selective refinement and optimal vertex placement shall be researched.

7. PROOF OF PRECONDITIONS

Let *M* be a *valid* mixed mesh which means the rules (M1) to (M3) are true. Let *col* be an edge collapse that fulfils the preconditions (P1*), (P2) and (P3) on *M*.

We want to proof that *col(M)* also is a valid mesh.

We use a proof by contradiction and presume the opposite: "*col(M)* is an invalid mesh". Thus, the mesh violates one of the rules (M1) to (M3).

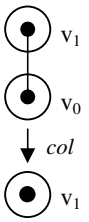
We will show that this presumption leads to a contradiction.

There are three reasons why the produced mesh *col(M)* may be invalid, namely the violation of (M1), (M2), or (M3).

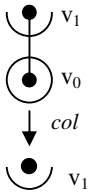
Case 1: *col(M)* violates (M1)

Since *M* is manifold with boundary there are three cases:

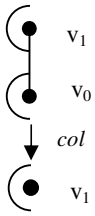
1) The vertices v_0 and v_1 are inner vertices. Since *M* is manifold the topology around v_0 and v_1 are homeomorphous to discs. After edge-collapse *col(M)*, v_1 still has a disc topology, all other vertices have an unchanged topology. Thus, *col(M)* is manifold and does not violate (M1). Contradiction!



2) The vertex v_0 is an inner vertex and v_1 is a boundary vertex. Since *M* is manifold the topology around v_0 is homeomorphous to a disc and the topology around v_1 to a half-disc. After *col*, the topology around v_1 is homeomorphous to a half-disc, all other vertices have an unchanged topology. Thus, *col(M)* is manifold and does not violate (M1). Contradiction!

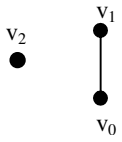


3) The vertices v_0 and v_1 are boundary vertices. Due to (P2) this means $\{v_0,v_1\}$ is a boundary edge. After *col*, the topology around v_1 is homeomorphous to a half-disc. All other vertices have an unchanged topology. Thus, *col(M)* is manifold and does not violate (M1). Contradiction!



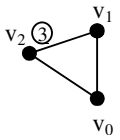
Case 2: *col(M)* violates (M2)

Let the vertices v_0 and v_1 be the inner vertices involved in *col*. Let v_2 be the inner vertex that violates (M2), such that $val(v_2)<3$ in *col(M)*.

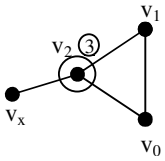


The degree of an inner vertex can only be decreased by *col* if it is a neighbour of v_0 and v_1 (proof left to the reader). The valence of v_2

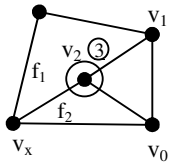
must be 3. Since $val(v_2)<3$ would violate already (M2) and $val(v_2)>3$ would not lead to $val(v_2)<3$ in $col(M)$.



Since $val(v_2)=3$, v_2 must have exactly one additional neighbour v_x. In addition, v_2 is an inner vertex and M is manifold, thus the topology around v_2 must be homeomorphous to a disc.



Since the topology around v_2 is homeomorphous to a disc and $val(v_2)=3$, there must be f_1 and f_2 with $\{v_x, v_2, v_1\} \in f_1$ and $\{v_x, v_0, v_2\} \in f_2$ to close the circle around v_2.



Due to f_1 and f_2, v_x is a neighbour of 2nd order to v_0 and v_1 but does not share one common face with v_0 and v_1. This means that *col* already violates (P1*). Contradiction!

Case 3: *col(M)* violates (M3)

This case is trivial since all faces with degree < 3 are removed after an edge collapse *col*. Thus, *col(M)* can never violate (M3), which also leads to a contradiction in this case.

Since all three cases lead to a contradiction, this means the presumption *col(M) is invalid* must be wrong. Thus, *col(M)* must be valid. *Q.E.D.*

8. ACKNOWLEDGMENTS

This work has been partially supported by the European Network of Excellence AIM@SHAPE, IST project 506766.

9. REFERENCES

[BSBK02] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt. OpenMesh. A generic and efficient polygon mesh data structure. OpenSG Symposium, 2002.

[CRS96] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno. Metro: measuring error on simplified surfaces. Technical report, Paris, France, 1996.

[FDF+90] James Foley, Andries van Dam, Steven Feiner, and John Hughes. Computer Graphics: Principle and Practice, Second Edition. Page 473. Addison-Wesley Publishing Company, Reading, Massachusetts, 1990.

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pages 209-216, ACM Press, New York, NY, USA, 1997.

[GH98] Michael Garland and Paul S. Heckbert. Simplifying surfaces with colour and texture using quadric error metrics. In VIS '98: Proceedings of the conference on Visualization '98, pages 63-269, IEEE Computer Society Press, Los Alamitos, CA, USA, 1998.

[Gum04] Stefan Gumhold. Progressive polygonal meshes. Technical Report WSI. 2004.4, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, July 2004.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pages 19-26, ACM Press, New York, NY, USA, 1993.

[Hop96] Hugues Hoppe. Progressive meshes. In SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 99-108, ACM Press, New York, NY, USA, 1996.

[KEH97] S. Kuschfeldt, T. Ertl, and M. Holzner. Efficient visualization of physical and structural properties in crash-worthiness simulations. In IEEE Visualization '97, IEEE Computer Society Press, pp.487-490,583, 1997.

[MG97] Anish Malanthara and Walter Gerstle. Comparative study of unstructured meshes made of triangles and quadilaterals. Proc. of 6th Intl. Meshing Roundtable, pp. 437-447, 1997.

[PS97] Paolo cignoni, Claudio Montani, Enrico Puppo, and Roberto Scopigno. Multiresolution representation and visualization of volume data. IEEE Transactions on Visualization and Computer Graphics, pp. 352–369, October–December 1997.

[RBH03] Shaun D. Ramsey, Martin Bertram, and Charles Hansen. Simpli_cation of arbitrary polyhedral meshes. In IASTED Computer Graphics and Imaging 2003, pp. 117-222, 2003.

[Som03] Ove Sommer. Interaktive Visualisierung von Strukturmechaniksimulationen. PhD-Thesis, Universität Stuttgart, 2003. URN: urn:nbn:de:bsz:93-opus-15600.