Computer-Aided Design 41 (2009) 1095–1106

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

User-assisted simplification method for triangle meshes preserving boundaries

C. González *, J. Gumbau, M. Chover, F. Ramos, R. Quirós

Departamento de lenguajes y sistemas informáticos, Universitat Jaume I. Av. de Vicent Sos Baynat, sn., 12071, Castellón, Spain

ARTICLE INFO

Article history:
Received 16 December 2008
Accepted 30 September 2009

Keywords:
Mesh simplification
CAD
User-assisted methods

ABSTRACT

This paper proposes a user-assisted mesh simplification method applied to CAD models converted to triangle meshes. This work offers the possibility of simplifying each subobject independently and at different levels of detail. This way, the user can simplify the whole model and then modify some parts, by simplifying more or by refining the desired subobjects. This can be performed while the total number of triangles in the simplified model is maintained. In this method any error metric based on an edge collapse operation can be used. Boundaries between subobjects are preserved and important attributes in the final aspect of simplified models, like normals and texture coordinates, are also considered.

1. Introduction

Nowadays 3D scenes are commonly represented with a high degree of polygonal complexity and many of the objects used in the scenes are generated from computer-aided design tools. Thus, polygonal models converted from CAD models are usually composed of a great number of polygons. This conversion to polygonal meshes (like X3D) is usually performed in order to render and manage the models in interactive applications running in a network. The current available hardware cannot however always handle all this geometry in a realistic way. Therefore, in interactive applications the accuracy of the models and the time required to process them must be taken into account.

In recent years, different solutions have been developed for interactive applications. One of these is simplification of the objects, which attempts to reduce their polygonal complexity, while maintaining the appearance of the final object as much as possible. Simplification methods allow the amount of geometry needed to represent an object to be reduced, trying to maintaining the visual quality, which benefits the performance of the GPU. Therefore, simplification methods produce objects with less geometry than the original ones. Interactive applications need to render and manage 3D scenes with a realistic framerate. Thus, the use of simplified objects can help to achieve this, because of the reduction of the geometric information. Many articles about simplification techniques have appeared in the literature and some surveys of such works can be found in [1,2], including simplification methods for CAD models [3]. We can distinguish between different criteria for mesh simplification, like geometry-based and viewpoint-driven simplification methods. Many of them are based on geometric metrics to define the order of the simplification steps. These methods are relatively fast and usually offer simplification with a good appearance. On the other hand, methods based on the user's point of view try to generate not only good geometric results, but also realistic results for the viewer by removing, for example, parts of the object that are not visible to the user. These methods are usually slower than methods based on the geometry. A simplification method makes use of a simplification operation and an error metric.

Complex shapes with several parts (subobjects) can be obtained from a design process. These CAD models are converted to polygonal meshes to be rendered and managed with a high framerate in interactive applications running in a network and subobjects are treated as submeshes. Different works for meshing objects can be found in the literature [4–6]. Simplification of polygonal meshes obtained from CAD models requires special attention. These meshes are usually composed of a great number of different submeshes that are not necessarily interconnected. This is the difference between this kind of mesh and other kinds of mesh usually used in interactive applications, such as games. Therefore, if this characteristic is not taken into account in the simplification process, undesired artifacts could be obtained with meshes obtained from CAD models, such as holes or distortion between the subobjects. Moreover, users can demand simplifications at different levels of detail of the different subobjects of the model. That is, with different percentages of simplification, usually given as the relation between the number of faces in the simplified model and the original number of faces. For example, a wheel of a car will need a greater level of detail than other parts of a car, because it usually has a rounded shape in the original model and if it is over-simplified considerable distortion could be obtained in it.

Simplification methods do not always present simplifications that satisfy the user's requirements about the total number of

* Corresponding author. Tel.: +34 964 728325; fax: +34 964 728435.
E-mail addresses: cgonzale@lsi.uji.es (C. González), jgumbau@lsi.uji.es (J. Gumbau), chover@lsi.uji.es (M. Chover), jromero@lsi.uji.es (F. Ramos), quiros@lsi.uji.es (R. Quirós).

0010-4485/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2009.09.007

triangles in the simplified model and the levels of detail of the different subobjects. For example, the user can demand a total number of triangles in the simplified model while some parts of the model are maintained or simplified to a specific percentage of simplification. A simplification process is not usually used for this purpose, because this process is done by hand by the designers. Moreover, simplification tools do not present an automatic process to work with meshes generated from CAD models with the possibility of simplifying the subobjects to different percentages of simplification. It is usually an elaborate process and these applications do not always use the best error metrics.

We present a user-assisted mesh simplification method applied to CAD models converted to triangular meshes. This method allows the different subobjects to be simplified at different levels of detail, avoiding the appearance of holes and preserving the boundaries between the subobjects. This way, the user can simplify the whole model and modify some parts, by simplifying more or by refining the desired subobjects. This can be performed while the total number of triangles in the simplified model is maintained. In the proposed method any metric based on edge collapse operation can be used. Therefore, the best and newest metrics can be integrated in the method. Textures and normals, that play an important role in the final aspect of the model, are also considered.

This is a user-assisted method, because the user can test simplifying some parts of the model or refining other simplified ones. The goal is to obtain a simplified version of the model that satisfies the user's requirements about the total number of triangles in the simplified model and the levels of detail of the different subobjects. Two different error metrics have been used in order to present the results: a geometry-based error metric (QSlim quadrics [7]) and a viewpoint-driven error metric (VMI [8]).

The main contributions of the proposed method are:

- The different parts (subobjects) of the models can be simplified with different levels of detail. This is a user-assisted method because the user can simplify the whole model and then modify the desired subobjects (by simplifying more or by refining these parts). These modifications can be done while the total number of triangles is maintained. Relative simplifications can also be performed, by maintaining a subobject always to a percentage of simplification of any other.
- Boundaries between subobjects are preserved and no distortion between subobjects is produced in simplified models. The method prevents the appearance of holes between the different subobjects in simplified models.

Models do not only consist of geometry, like vertices, edges and faces. They also usually have associated properties that allow the model to present a more realistic appearance, like normals or texture coordinates. These properties are also taken into account in this method. We recalculate texture coordinates after each simplification step and normals after the whole simplification process. Moreover, any metric based on an edge collapse operation can be used in the proposed method. The user can also choose the kind of edge collapse operation to be performed.

The rest of this paper is structured as follows. Section 2 describes the background of this research. In Section 3 we explain the proposed method. Section 4 shows some results and in Section 5 we discuss the conclusions.

2. Background

Different physics-based methods for CAD model simplification can be found in the literature [3]. These methods can be classified in different types of technique, like techniques based on surface entities [9], techniques based on volumetric entities [10], techniques based on explicit features [11] and techniques based on dimension reduction [12]. However, our method is included in the real-time rendering context and it works with meshes obtained from CAD models after a meshing process [4–6]. Over the last years many simplification methods for triangular meshes have been developed. A survey can be found in [2]. But meshes obtained from CAD models are usually composed of a great number of submeshes. These submeshes are not necessarily interconnected. Simplification methods do not always present simplification with the level of detail of the different subobjects that the user requires. Moreover, some of these methods can produce undesired artifacts with meshes obtained from CAD models, such as holes or distortion, because these models are normally composed of different subobjects without any interconnection between the subobjects. Here we present some related works on simplification methods for meshes. We can distinguish between methods on the basis of different criteria. Simplification concepts can be found in the literature [13]. A simplification method makes use of a simplification operation and an error metric. Therefore, we can classify them according to the simplification operation (Section 2.1) and to the error metric (Section 2.2). In Section 2.3 some studies about topology preserving simplification methods are presented.

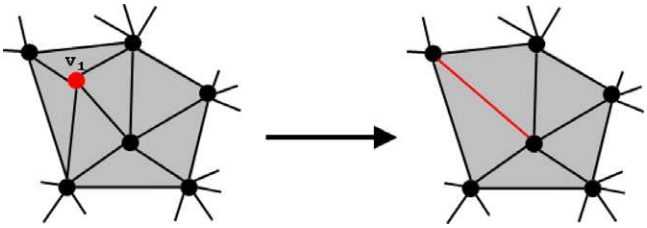


Fig. 1. Example of vertex decimation operation.

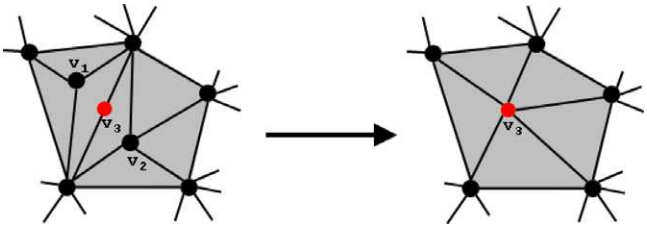


Fig. 2. Example of vertex clustering operation.

2.1. Simplification operation

Algorithms for mesh simplification can be classified according to the simplification operations. We highlight four different types:

- Vertex decimation [14,15]. These methods are based on the removal of vertices from the mesh. Once a vertex is removed, all faces using that vertex are also removed and then the hole is retriangulized. Because of the way it creates triangles, this kind of algorithm is limited to manifold meshes. An example can be seen in Fig. 1.
- Vertex clustering [16,17]. These methods are based on an inclusion box divided into several cells. All vertices that are included in a cell are collapsed into one single vertex and the triangles that share the removed vertices are updated. These methods tend to be very fast but the visual appearance of the final mesh is not relatively very good. An example can be seen in Fig. 2.
- Edge contraction [7]. These methods use an iterative selection of edges to be removed in order to decrease the level of detail. At each step, a single edge is selected for removal (or a pair of unconnected vertices). All faces sharing that edge are also removed, and the faces which share just one of the vertices of that edge are updated to cap the hole. Degenerated faces and edges are also removed. An example can be seen in Fig. 3.

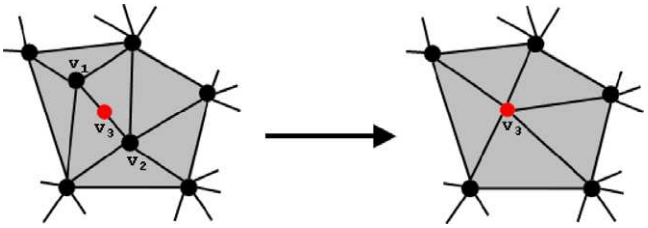


Fig. 3. Example of edge contraction operation.

- Morphological operations [18]. These methods apply morphological operations (in contrast to volumetric objects), like erosion and dilation, to decrease the level of detail of objects. They are very fast and tend to offer good results.

2.2. Error metric

The error metric in a simplification method will give the order of the simplification steps. We distinguish between methods based on the geometry and methods based on the user's point of view.

2.2.1. Geometry-based simplification methods

Methods based on geometry take only geometric information into account in order to establish the error metric. During the last years, a great number of simplification methods based on geometry have been developed, such as the works presented in [19–21]. These methods make it possible to contract edges and join vertices so that the connectivity of the mesh is preserved. A weight is assigned to each edge in a pre-process that considers the geometric importance of that edge in the simplification.

One of the most relevant improvements to the geometry-based simplification methods is the incorporation of vertex attributes, such as texture coordinates and normals, into the simplification metric. Hoppe extended his initial work [22] by incorporating color and texture coordinates [23]. The authors of the QSlim [7,24] algorithm also extended their metric to take this kind of information into account. Cohen et al. [25] developed an algorithm based on edge collapses which converts vertex positions, diffuse colors and normals into texture and normal maps. This algorithm is based on a texture deviation metric. Both the geometry of the object and also the texture frequencies were considered in [26]. To make the method more precise, pixels are subdivided into subpixels. The method presented in [27] recalculates a new texture for each simplification step, an indexing map being used to avoid a loss of precision.

Garland and Zhou [28] presented a method for simplifying simplicial complexes of any type embedded in Euclidean spaces of any dimension.

2.2.2. Viewpoint-driven simplification methods

Simplification methods based on the user's point of view take not only geometric information into account, but also visual information to compute the error metric. Therefore, these methods will remove first those parts of the object that are not visible to the user. Different viewpoint-driven simplification methods can be found in the literature, such as the works presented in [29–31].

Lindstrom et al. [32] take a visual approach to deal with the problem of visual similarity by creating a purely image-based metric. Basically, their method determines the cost of a collapse operation by rendering the model from a set of viewpoints. Then the algorithm compares the resulting images and adds the per-pixel error as an extra value for each pixel. All edges are then sorted using this error information so that the first edge collapses are those which have the least error. The main advantage of this metric is that it offers a good balance between the geometry of the object and its vertex attributes in a natural way, without the user having to assign any weight to them. Its main disadvantage is its high temporal cost.

Luebke et al. [33] presented a method for view-dependent simplification using perceptual error metrics. Zhang et al. [34] proposed a new algorithm that takes visibility into account. This work defines a new visibility function that considers the surface of the model and a set of cameras located on the surface of a virtual sphere surrounding the model. The number of cameras influences the precision and the temporal cost of the algorithm. Luebke et al. used up to 258 cameras. To guide the simplification process, they combined their visibility algorithm with Garland's quadric-based error metric [7].

Attributes were also considered in [35]. In this work Williams et al. extended the method proposed by Luebke et al. [33] to shaded and textured meshes.

Mathematical concepts, such as saliency and entropy, were used as measures for creating viewpoint-driven metrics. Lee et al. [36] introduced the saliency concept as an error metric, which was used for mesh simplification algorithms. Basically their work consists of the generation of a saliency map to be used in the QSlim algorithm as in [34] simplification algorithm. Recently, Castelló [37] presented a view-dependent method based on the entropy from a viewpoint, a concept which is taken from the "Theory of Information" [38,39]. The entropy of a viewpoint is obtained from the distribution of the projected areas of the polygons of the mesh. Projected areas are calculated by analyzing the frame buffer using a histogram.

2.3. Topology preserving simplification

In the literature we can find some works oriented to preserve the topology of the mesh in a simplification process. Topology is the branch of pure mathematics that deals with continuity and other concepts derived from it, like the properties of the objects, independently of their sizes and shapes. The genus is the number of holes of a mesh. A mesh is manifold when every edge is shared by exactly two faces. A mesh is manifold with boundaries if it has also edges that pertain only to one face.

Hoppe et al. [40] produce a simplified mesh from a given set of data points and an initial triangle mesh. They guarantee that both meshes have the same topological type. In [41] the authors introduced the link conditions. If these conditions are satisfied it is guaranteed that the complex obtained after an edge collapse is homeomorphic to the original one.

Statistical measures have also been used by Wu et al. [42] to distinguish between rough and flat parts in the models. This method marks as illegal a face constriction if a triangle pertains to a boundary or is non-manifold.

Changes in the topology after a simplification step can be detected by a test. Vivodtzev et al. [43] presented a test for changes in the topology after an edge collapse in meshes with embedded polylines. To do that they define the extended complex, which encodes both the topology of the mesh and the topology of the embedded polylines.

3. Simplification method

The proposed method receives a mesh obtained from a CAD model, generated by a meshing process. This mesh can be divided into different subobjects, treated as submeshes. The data structure of the input meshes accepted by this method is detailed in Section 3.1. In order to simplify a mesh obtained from a CAD model without losing information about the model, we store the following information:

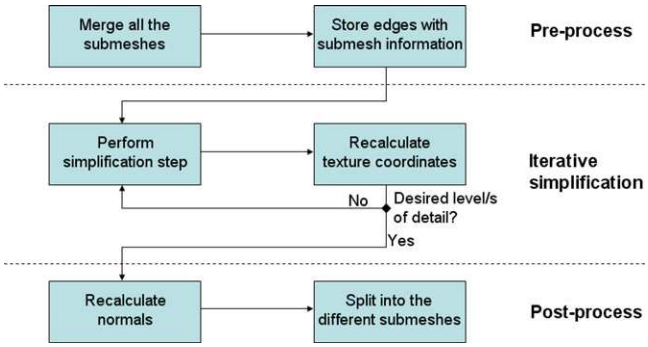


Fig. 4. Scheme of the method.

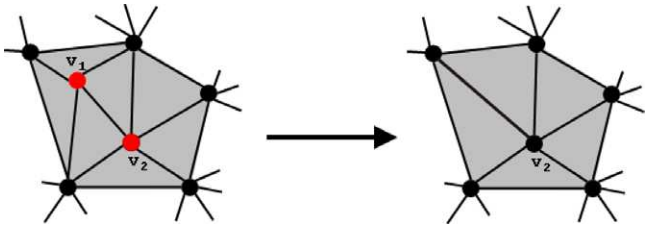


Fig. 5. Example of half edge collapse operation.

- Subobject information: the vertices, edges and faces that pertain to any particular subobject.
- Boundary information: we mark each edge that pertains to any boundary as a *boundary edge*.
- Connectivity: information about the neighborhood of each edge of the model.

This is the information that the method needs to work correctly. This way, the different subobjects of the model can be simplified to different percentages of simplification, while the coherence and connectivity between the subobjects are maintained.

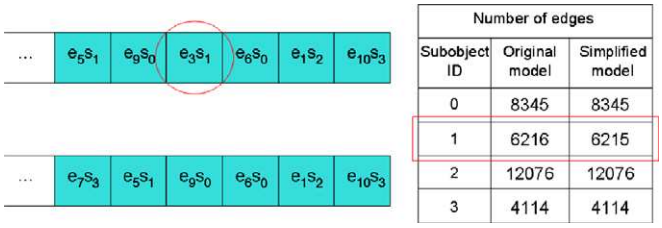
The method is divided into a pre-process that merges the model, an iterative contraction phase and a post-process to split the model into its initial subobjects. Fig. 4 shows a general scheme of the process. All these steps are detailed in Section 3.2.

3.1. Terminology

We define a mesh (M) as a set of submeshes (S). Each submesh $s \in S$ has indices to its corresponding vertices (V), edges (E) and faces (F). The set of vertices has the spatial coordinate of each vertex $v \in \Re^3$ and the information about the normals and texture coordinates associated to each vertex. Thus, each vertex $v \in V$ will have a normal $n \in \Re^2$ and a texture coordinate $t \in \Re^2$ associated to it. The set of edges E has the information about the edges with indices to each vertex of the edge and information about the neighboring polygons. And the set of faces F has the information about the faces with indices to each vertex and each edge that pertains to the face.

In the method presented here any error metric based on edge collapse operation can be used. This operation is applied at each simplification step and works by removing an edge and unifying its vertices. The final vertex can be located at a new position. However, the operation is called half edge collapse variant when one vertex is collapsed into another one, that is, no new vertices are created because we always refer to an existing vertex in order to compute the operation. As shown in Fig. 5, an edge with vertices v_1 and v_2 collapses into vertex v_2, thus producing the simplified mesh.

The proposed method makes also use of split operation. A split operation is the inverse operation of an edge collapse. That is, from one vertex a new edge will be generated.



Number of edges		
Subobject ID	Original model	Simplified model
0	8345	8345
1	6216	6215
2	12076	12076
3	4114	4114

Fig. 6. Example of simplification step. A collapse of an edge of the subobject 1 is produced.

A percentage of simplification usually refers to the relation between the number of faces in the simplified model ($tfaces_{simpl}$) and the number of faces in the original model ($tfaces_{orig}$). And it is given by $\frac{tfaces_{simpl}}{tfaces_{orig}} * 100$.

3.2. Simplification steps

The method receives the model with the information defined in Section 3.1 and the different levels of simplification of the subobjects required by the user as parameters. These levels of simplification represent the levels of detail of the subobjects. The user can demand different levels of detail on the different subobjects of the model. This will produce an automatic simplification, preserving boundaries. The user can try giving different levels of detail until a simplification that satisfies the user's requirements is obtained. The user-assisted capability of the method is explained in detail in Section 3.7.

3.2.1. Pre-process

First of all, a pre-process step is performed, which merges all the submeshes in a single virtual mesh, and stores all the information about the original connectivity of the meshes. This is a virtual mesh because it is a temporal mesh that will be deleted after the simplified model is obtained. This step is based on the distance of the vertices of different meshes in the original object. Moreover, the edges between two neighboring submeshes are marked as boundaries.

3.2.2. Iterative simplification

Next, the mesh is simplified. To do so, we store the edges in their simplification order (Fig. 6). This order is obtained by the associated cost given by the metric. Every stored edge will also have an associated identifier of subobject to know to which subobject pertains the edge. And for every subobject of the model we store the number of edges in both the original and the simplified version. This way, when we have to simplify a subobject we collapse the first edge with the identifier of the subobject. And we repeat this process until the desired level of detail of the subobject is obtained. Thus, we simplify each subobject to its associated level of detail. Moreover, after each simplification step is performed, the texture coordinates of the affected vertices are recalculated by a linear interpolation.

We have to take into account that the simplification of a submesh may affect the neighboring submeshes. That is, if edge collapse operations are produced near a boundary of a submesh, the edges of the boundaries will be moved and consequently the neighboring submeshes will also need to be moved in order to avoid generating a hole. But this will only be produced when the demanded level of detail is very low, because we perform the algorithm explained in Section 3.4 to preserve the boundaries. Consequently, these edges will be collapsed in the last steps of the simplification process.

The simplification of the subobjects at different levels of detail also offers the possibility to the user of maintaining a relative

```
vertex_to_move = None;
if (mesh.boundaries.has_vertex(v1)) then
  if (mesh.boundaries.has_vertex(v2)) then
    vertex_to_move = normal_placement(v1,v2);
  else
    vertex_to_move = v2;
  end if
else if (mesh.boundaries.has_vertex(v2)) then
  vertex_to_move = v1;
else
  vertex_to_move = normal_placement(v1,v2);
end if
```

Fig. 7. General algorithm for boundary preservation.

simplification, that is, to maintain some subobjects simplified to a multiple of the level of detail of other subobjects.

The user can also require a total number of triangles (by giving a single level of detail for the whole model), and then modifying the levels of detail of the different subobjects. The total number of triangles will automatically be preserved. This is explained in detail in Section 3.7.

3.2.3. Post-process

After simplifying the mesh, we recalculate all the normals of the model. Here we have to take into consideration that, depending on the shape of the different parts of the model, vertex normals or face normals will be needed. Therefore, depending on the angle of the affected faces, the normals are calculated per vertex or per face.

Finally, we retrieve the simplified mesh and split it into different submeshes, using the interconnectivity information about the original submeshes that was stored in the pre-process step.

3.3. Decimation computation

This method sets a decimation factor for each edge. The decimation cost reflects how much the appearance of the object will change, so that the edges with lowest values are the first to be collapsed. The decimation cost computation depends on the error metric used.

For each edge contraction $(v_1, v_2) \rightarrow v$ the following steps are performed:

- Collapse the edge that contains the vertices v_1 and v_2.
- Remove all triangles that share the edge.
- Remap all triangles shared by v_1 to v_2.
- Recalculate the cost (decimation coefficient) for the vertex based on the new connectivity information.

3.4. Boundary preservation

Using the information stored in the pre-process step, the implemented algorithm allows us to preserve the boundaries between the different subobjects. This technique is useful for any manifold model with boundaries. The method marks an edge in the model as a boundary when it only has one associated face. Two edges of different subobjects form a boundary between these subobjects, when the distance between their corresponding vertices is lower than a threshold. When an edge with a vertex in a boundary has to be remapped, the other vertex will overlap the first one. To do this, an attribute that indicates whether the vertex pertains to a boundary or is an internal vertex is stored in the data structure of the vertices. By so doing, the original shape of the boundary remains unchanged for as long as it is possible. A general scheme to this approach can be seen in Fig. 7.

The user can select what kind of edge collapse will be performed, depending on the user's requirements and limitations. As we explain in Section 3.2 we store the simplification sequence in order to be able to refine some simplified parts of the model. Thus, for example, if the user wants to store less information about the changes produced in the simplification process, a half edge collapse operation will be applied in order to not create new vertex positions. But if the user wants optimal positions for the collapsed edge, a full collapse operation will be performed, and the new vertex coordinates (or displacement relative to the originals) will be stored.

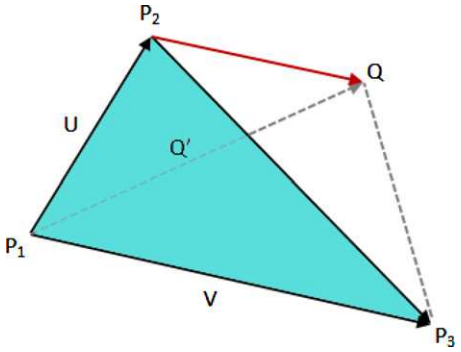


Fig. 8. Example of a modified triangle.

Our boundary preservation technique preserves the boundaries between subobjects. The boundaries of the whole model (edges of a subobject that are not connected to any other edges) are usually preserved by the metric of the simplification methods. For this consideration, this kind of edge could have assigned by the metric a high simplification cost in order to be maintained until the last simplification steps.

If a vertex in the edge is classified as a boundary to be preserved, the algorithm showed in Fig. 7 is executed. If both vertices are in a boundary (or different boundaries) we compute a normal vertex placement, that is, the final vertex will be located to the position given by the metric. This position is the one that minimizes the contraction cost. If only one vertex is in a boundary, the other vertex of the edge is the one to be moved. Otherwise, if no vertices are on a boundary, we also compute a normal vertex placement. This way, if the user wants to modify the level of detail of any subobject, the algorithm will perform the simplification operations, preserving automatically the boundaries between subobjects.

We must distinguish between this possibility and the option offered by some simplification methods, like QSlim [7], that allows borders to be preserved. Borders and boundaries are not the same concept. A border is an edge that pertains to only one face of the model. However, a boundary between two submeshes is an edge that pertains to two different faces, because we have merged the whole model in a pre-process before the simplification steps are performed.

The proposed method preserves the boundaries, however the topology preservation depends on the metric used.

3.5. Texture coordinates recalculation

At each simplification step texture coordinates for each modified vertex need to be recalculated in order to maintain the mapping appearance. The new texture coordinate is calculated based on the displacement of the mapped vertex using a linear interpolation. An example of a modified triangle is shown in Fig. 8, where the vertex P_2 is moved to the position Q.

To obtain the offset that must be applied to the texture coordinates of the modified vertex, we propose the following system of equations:

$$Q'_x = \alpha U_x + \beta V_x + \gamma N_x \quad (1)$$

$$Q'_y = \alpha U_y + \beta V_y + \gamma N_y \quad (2)$$

$$Q'_z = \alpha U_z + \beta V_z + \gamma N_z \quad (3)$$

Fig. 9. Original ball model (a), ball model simplified to 30% without texture coordinate interpolation (b), and ball model simplification to 30% with application of texture coordinate interpolation.

having $\vec{Q}' = \vec{Q} - \vec{P^1}$, $\vec{U} = \vec{P^2} - \vec{P^1}$, $\vec{V} = \vec{P^3} - \vec{P^1}$, where $\vec{P^1}$, $\vec{P^2}$, $\vec{P^3}$ are the three vertices of a modified triangle, $\vec{Q}$ is the new position of the modified vertex and $\vec{N}$ is the triangle normal. α, β and γ are the coordinates of $\vec{Q}$ expressed in the triangle coordinate system. They also express how much the modified vertex has been moved in triangle coordinate system units, so that they can be used to calculate the perturbed texture coordinate for the modified vertex. We use the following formula:

$$T_u^{res} = \alpha\left(T_u^2 - T_u^1\right) + \beta\left(T_u^3 - T_u^1\right) + T_u^1 \tag{4}$$

$$T_v^{res} = \alpha\left(T_v^2 - T_v^1\right) + \beta\left(T_v^3 - T_v^1\right) + T_v^1 \tag{5}$$

where $\vec{T_1}$, $\vec{T_2}$ and $\vec{T_3}$ are the original texture coordinates of the three vertices, $\vec{T^{res}}$ is the new texture coordinate for the modified vertex and α and β are the coefficients calculated from the Eqs. (1)–(3).

Note that we only use α and β because texture coordinates are two-dimensional vectors contained in the plane formed by the triangle. As γ is the displacement along the normal vector of the triangle, we do not need it.

In Fig. 9 we can see an example of the use of this texture coordinate interpolation. It can be observed that using texture recalculation more accurate textures are obtained.

3.6. Normals recalculation

After simplifying the model we recalculate all the normals of the model. Depending on the shape of the parts of the object, the normals will be calculated per face or per vertex. This is because some parts of the models need to use per-vertex normals (for example smooth and rounded parts) and other ones need to use per-face normals (for example corners). To do this, we perform the following algorithm:

- We expand the model, so that each face will have different indices of normals. Thus, we obtain the number of vertices as the number of faces multiplied by three.
- The face normal is assigned to each vertex.
- For each vertex we review all the other vertices of the model. If any vertex is located at the same spatial coordinate and the angle between their respective faces is less than a threshold angle given by the user (we use 30 degrees) we add the normal of this vertex and the indices of normals will be the same.

An example of the use of normals recalculation is shown in Fig. 10. It shows that a combination of per-face and per-vertex normals produce the best results.

3.7. User-assisted simplification

The proposed method is user-assisted because the user can test simplifying different parts of the model. Simplifying the whole model with an unique level of detail does not always produce a simplification that satisfies the user's requirements about the total number of triangles in the simplified model and the levels of detail of the different subobjects. Our method gives total control to the user for giving different levels of detail to the different subobjects. This will produce an automatic simplification that preserves the boundaries between the subobjects, maintaining the coherence in the model. The user can change the different levels of details until a desired simplification of the model is obtained.

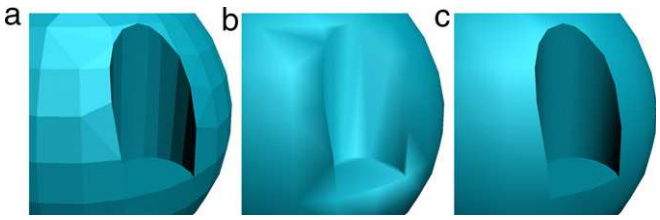


Fig. 10. Part of a model with the per-face normals (a), with the per-vertex normals (b) and taking into account the angle between the faces (c).

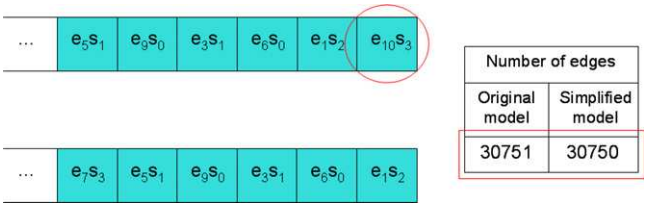


Fig. 11. Example of simplification step. A collapse of an edge is produced.

A possible strategy for the user to simplify the model could be the following: if the user wants a maximum number of triangles in the simplified model to be rendered, it can be limited by giving a level of detail for the whole model. The model will be simplified to this level of detail. After that, the user can change the level of detail of any subobject of the model. The user can give a greater level of detail of a simplified subobject or simplify it more. This can be done until a simplified model that satisfies the user's requirements about the number of triangles in the subobjects is obtained. If a level of detail for the whole model was given, a change in a local level of detail will automatically affect to the other local levels of detail in order to maintain the final number of triangles. This will not affect to the subobjects that have an associated level of detail introduced by the user, because these subobjects have exactly the number of polygons that the user wants for them. Examples are discussed in Section 4.

In order to simplify the whole model with a single level of detail preventing the appearance of holes between the different subobjects and preserving the boundaries, we follow the same steps explained in Section 3.2, but without considering the subobject information associated to the edges (Fig. 11). We store the edges in the order of their simplification costs given by the error metric. We then simplify normally with the simplification method extracting the edges in a sequential order.

When the user demands to simplify a specific subobject more, this simplification will be performed normally with the subobject simplification explained in Section 3.2. After this, the algorithm will add more faces to the simplified model in order to maintain the required level of detail for the whole model. To do that, the stored simplification sequence is used. Hence, the algorithm will perform vertex split operations with the collapsed edges of the subobjects that have not a required level of detail introduced by the user. With this operation the vertex obtained after a collapse will produce the edge again and the affected faces will be retriangulized. This is performed until the total level of detail is obtained. A pseudo-code of this algorithm is presented in Fig. 12. In this figure we can see that the vertex split operations are performed in order to obtain the desired total level of detail. The extraction of a simplification step from the simplification sequence will return the last collapse performed. Thus, the vertex split operations will undo the last

```
obtain_totalLevelofDetail(Model m, LOD total_lod, SIMPL_SEQUENCE simpl_seq)
{
if (totalLevelofDetail(Model)<total_lod)
  while (gobalLevelofDetail(Model)<>total_lod)
    simpl_step= extract_last_simpl_step(simpl_seq)
      if (simpl_step.submesh has not local level of detail assigned by the user)
      performSplit(simpl_step.info, m)
    end if
  end while
else if (totalLevelofDetail(Model)>total_lod)
  while (gobalLevelofDetail(Model)<>total_lod)
    edge= extract edge of subobject without associated level of detail
    performCollapse(edge, m)
   end while
end if
}
```

Fig. 12. Pseudo-code for reobtaining the desired total number of triangles after a modification in any submesh.

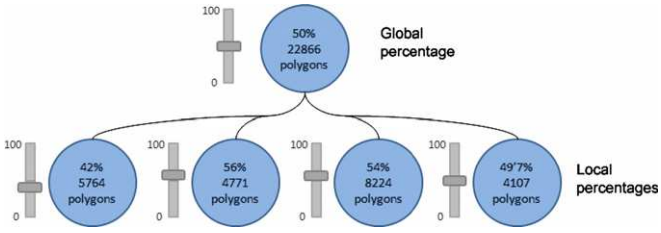


Fig. 13. Levels of detail of a model simplified to 50% of its original geometry.

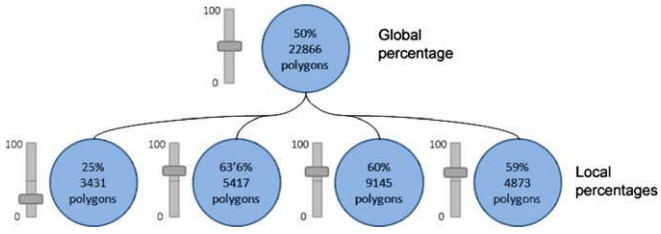


Fig. 14. Levels of detail of a model simplified to 50% of its original geometry, simplifying the first subobject to 25%.

collapse operations. The vertex split operation is not performed if the extracted simplification step pertains to a submesh that has an associated level of detail introduced by the user, because this submesh has exactly the number of polygons that the user wants for it.

On the other hand, when the user refines a simplified subobject, this refinement will produce split operations in this subobject. For this purpose, the simplification sequence will be used. This way, the last collapse operations stored with the information of this subobject will be undone, generating the edges that had been collapsed. This is performed until the desired level of detail in the subobject is obtained. And more edge collapse operations from other subobjects that do not have an associated local level of detail must be performed in order to maintain the desired total number of triangles in the final object. This is also considered in Fig. 12.

An example is shown in Figs. 13–15. Fig. 13 shows the levels of detail of a model simplified with a total 50% percentage. The edges have been collapsed in the order of their simplification costs, without taking the subobject information into account. In Fig. 14 we can see that the first subobject has been simplified to 25%. The levels of detail of the other subobjects have to be modified in order to maintain the total number of triangles. Therefore, the other subobjects have been refined with the information stored in the simplification sequence. On the other hand, in Fig. 15 the first subobject has been refined to 100% of its original geometry. Thus, the other subobjects have been simplified in order to maintain the total number of triangles.

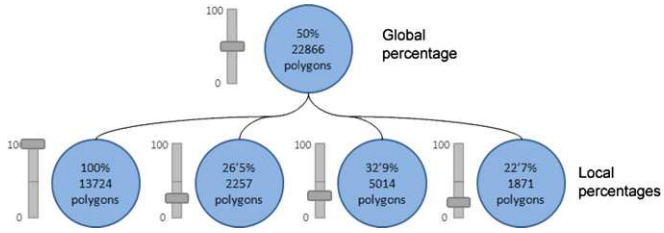


Fig. 15. Levels of detail of a model simplified to 50% of its original geometry, maintaining the level of detail of the first subobject at 100%.

Table 1
Details of the models presented in the results subsections.

Model	Subfigure	Triangles	Vertices	Subobjects
Casino	16(a)	315	357	10
Yacht	16(b)	532	352	11
Speaker	16(c)	5 202	5 276	184
X-wing	16(d)	8 716	10 026	1004
Cellular	16(e)	11 728	14 259	1541
Ball	16(f)	12 264	8 272	22
Racing car	16(g)	42 964	41 620	1579
Toy car	16(h)	59 589	56 442	2007

4. Results

We have tested several models with our method and some examples are presented here. The different characteristics of the method are divided into subsections. The models used to present the results are introduced in Fig. 16 and their geometric details exposed in Table 1. The subobjects of the models are represented with different colors. All the percentages of simplification are relative to the number of triangles of the models.

4.1. Simplification of subobjects

The proposed method allows subobjects to be simplified at different levels of detail. This way the user can simplify more some parts of a simplified model or refine other ones by giving them a greater level of detail. Some examples are shown in this subsection. In Fig. 17 we can see the simplification of a subobject of the speaker model. Figs. 18 and 19 show a refinement of some parts of the model in a simplified version. Fig. 18 shows the racing car model simplified to 10% and a refinement of the subobjects that make up the wheel at different levels of detail. And Fig. 19 shows the toy car model simplified to 10% and a refinement of the subobjects that make up the steering wheel at different levels of detail. In Fig. 20 we can see a wheel of a model refined with different levels of detail. In Fig. 20a the original model is shown, in Fig. 20b the whole model has been simplified to 10% of its original geometry, and in Fig. 20c and Fig. 20d a refinement of the wheel has been performed.

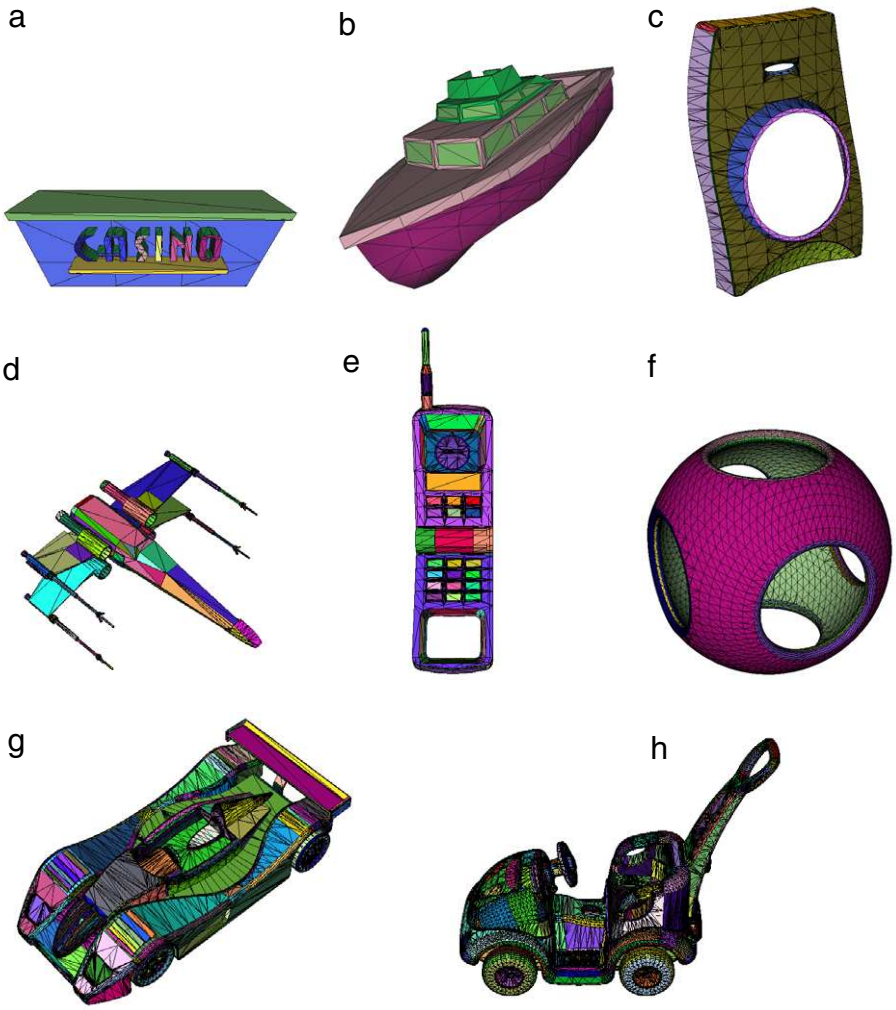


Fig. 16. Models presented in the results subsections.

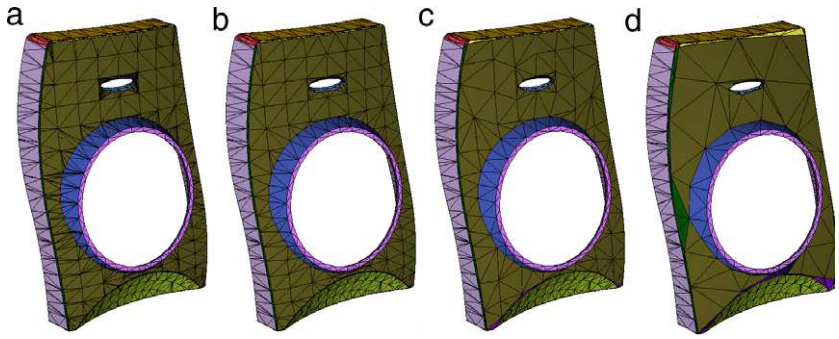


Fig. 17. Original speaker model (a) and simplifications of the front subobject represented by a brown color to 50% (b), 30% (c) and 10% (d).

In Fig. 20c the wheel is recovered at 50% and in Fig. 20d at 100% of its original geometry. It can be appreciated that the boundaries between the simplified parts are preserved. In Fig. 21more than one subobject of the ball model are simplified at different levels of detail. And Fig. 22 shows a relative simplification in the ball model, where a subobject is maintained to the half level of detail than other one.

4.2. Boundary preservation

In this subsection we justify why we unify the model and take boundaries into account. If the model is not unified and boundaries are not taken into account holes and distortion can be produced. We present some examples of simplifying the models by applying QSlim [7] without unifying the models and the simplifications applying our method (Figs. 23 and 24). We can appreciate that the boundaries between subobjects have been preserved with our method.

4.3. Texture coordinates recalculation

Textures play an important role in the final appearance of a simplified model. Therefore, a recalculation of the texture coordinates

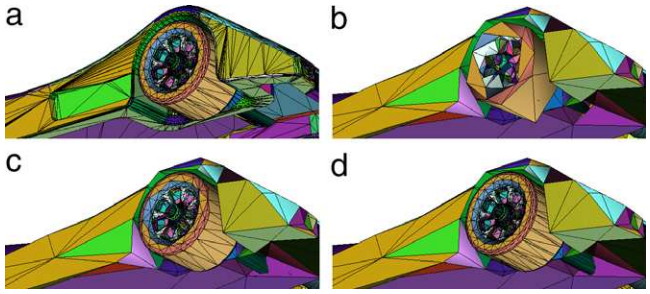


Fig. 18. Original racing car model (a), model simplification to 10% (b) and refinements of the subobjects that make up the wheel at 50% (c) and 100% (d).

of vertices that have been affected after a simplification step is performed. An example of this recalculation was presented in Fig. 9. In this figure we can see a simplification applying the texture coordinates recalculation and a simplification without applying it. It can be observed that the textures are more accurate with the texture coordinate interpolation.

4.4. Normals recalculation

Normals are also an important attribute to be taken into account in order to achieve a final appearance of the simplified model that is as similar as possible to the original one. We have presented a post-process method that considers the angle between faces in order to calculate per-vertex or per-face normals. In Fig. 25 we can see two points of view of an object in which the normals have been recalculated. In the parts of the model with angles between faces higher than the threshold, normals have been calculated per face, as for example in corners or the back lights of the model. And in the parts of the model with low angles between faces normals have been calculated per vertex, as for example in the wheels.

4.5. Using different error metrics

In the proposed method any metric based on an edge collapse operation can be used. Here two different error metrics have been used in order to present the results: a geometry-based error metric (QSlim quadrics [7]) and a viewpoint-driven error metric (VMI [8]). Both metrics are detailed in Appendix. Figs. 26 and 27 show simplifications of the casino model and yacht models using both metrics. We can observe that the simplified models maintain the subobject information and the boundaries between them are preserved. Table 2 shows some statistics of these simplifications.

Table 2
Models presented in the results subsections.

Model	Triangles removed	Error metric	Time (s)
Casino	189	QSlim	0.016
		VMI	5.062
Yacht	266	QSlim	0.032
		VMI	7.25

4.6. Temporal cost

The temporal cost introduced by this method during the simplification is negligible. We state this because:

- The possibility of simplifying the different submeshes of the model with different levels of detail does not introduce any temporal cost. The only difference is the order in which the edges are extracted.
- The computational and temporal cost introduced to maintain the boundaries is only the cost introduced by conditional operations.
- The computational and temporal cost introduced to recalculate the texture coordinates is only the cost introduced by a linear interpolation.

5. Conclusions

A user-assisted simplification method for triangle meshes generated from CAD models has been presented. This kind of mesh normally consists of different and independent subobjects. These subobjects are not necessarily interconnected. Thus, simplifications produced by some of the existing methods would usually produce distortion and holes between the subobjects of the models. Simplification methods do not always present simplifications that

Fig. 19. Original toy car model (a), model simplification to 10% (b) and refinements of the subobjects that make up the steering wheel at 50% (c) and 100% (d).

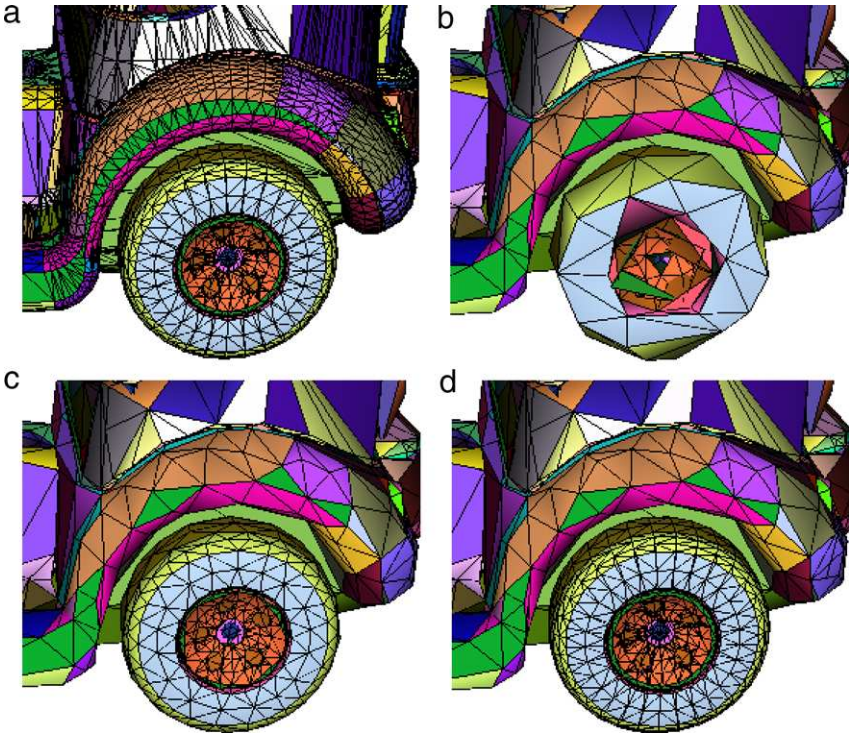


Fig. 20. Example of subobject simplification with our method. In (a) the original model is shown. In (b) the whole model is simplified to 10%. And in (c) and (d) the wheel of the model is refined to 50% and 100% respectively of its original geometry.

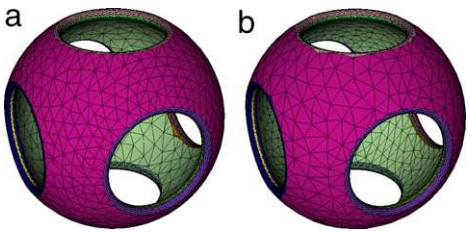


Fig. 21. Simplifications of the subobjects represented by the pink and green colors. In (a) the pink subobject is simplified to 50% and the green subobject is simplified to 30%. In (b) the pink subobject is simplified at 30% and the green subobject is simplified to 50%. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

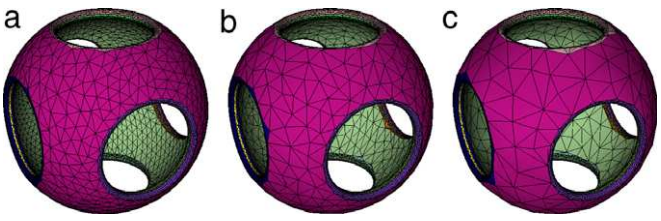


Fig. 22. Relative simplification. The subobject represented by the green color is simplified to 80%, 40% and 20%, while the subobject represented by pink color is always maintained at the half level of detail, that is, 40%, 20% and 10%, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

satisfy the users' requirements about the total number of triangles in the simplified model and the levels of detail of the different subobjects. Users can demand different levels of detail in each submesh. Therefore, some parts of the models have to be modified. A simplification process is not usually used for this purpose, because this process is done by hand by the designers. Moreover, simplification tools do not offer an automatic process to manage this kind of mesh with the possibility of simplifying the subobjects to different percentages of simplification.

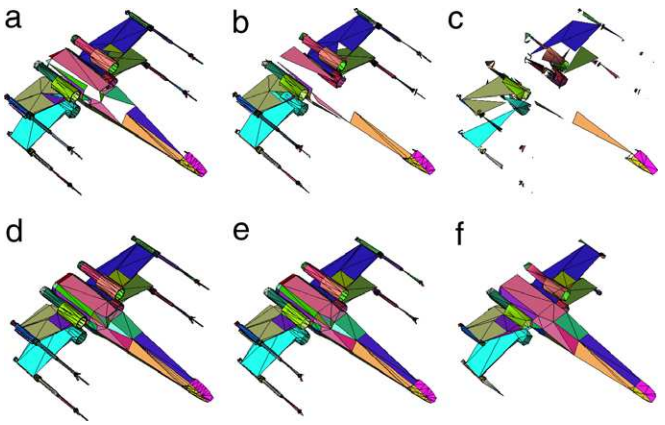


Fig. 23. Simplifications of X-wing model with QSlim to 50% (a), 30% (b) and 10% (c) and with our method to 50% (d), 30% (e) and 10% (f).

The proposed method allows the different subobjects of the meshes obtained from CAD models to be simplified with different levels of detail. It also presents the possibility of simplifying whole models with a single level of detail. Therefore, the user can simplify the whole model and then modify some parts, by simplifying more or by refining the desired subobjects. This can be performed while the total number of triangles is maintained. This way, the user can try with different levels of detail of the subobjects to find a simplification that satisfies the user's requirements, while the total number of triangles in the final object required by the user is maintained. Moreover, simplifications with proportions between the levels of detail of different subobjects can be performed. Boundaries are always preserved and no holes are produced in the simplified objects. This method also takes into account the information of texture coordinates and normals, which are necessary to obtain an accurate visual simplification. Any metric based on an edge collapse operation and any kind of edge collapse operation can be used in this method.

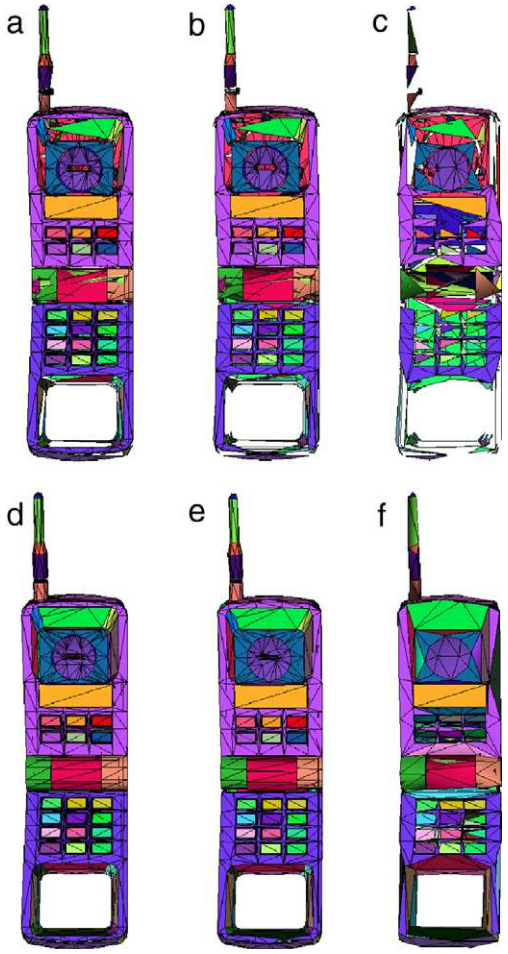


Fig. 24. Simplifications of cellular model with QSlim to 50% (a), 30% (b) and 10% (c) and with our method to 50% (d), 30% (e) and 10% (f).

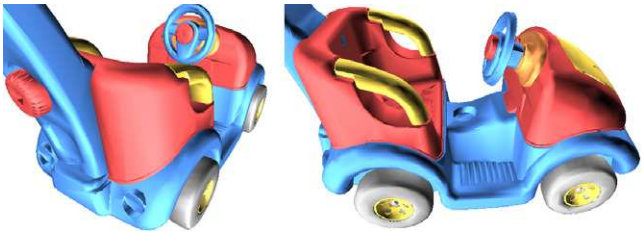


Fig. 25. Toy car model simplified to 30%, in which the normals have been recalculated.

Probably, one limitation of the method is that the memory necessary to store the simplification sequence and the information of connectivity depends on the complexity of the models. However, the main limitation of the method is that it is limited for metrics based on an edge collapse operation. Hence, a possible option for future work is to extend this work in order to be able to do any simplification operation.

We can see in the results set out in this paper how our method accurately preserves the boundaries and no holes are produced when we simplify either the whole model or independent subobjects with different levels of detail. We also present the final appearance of simplified models with texture coordinates and normals recalculation.

Moreover, the temporal cost introduced by this extension during simplification can be considered to be negligible in comparison to the total simplification time.

Acknowledgements

This work has been supported by the Spanish Ministry of Education and Science (TIN2007-68066-C04-02), Caja Castellón-Bancaja Foundation (P1-1B2007-56) and the Jaume I University (PREDOC/2005/12).

Appendix

A.0.1. QSlim quadrics

QSlim [7] computes the error metric by associating a set of planes with each vertex of the model. The decimation cost of an edge is obtained by the quadratic error metric, that measures the total squared distance of a vertex to the two sets of planes formed by the planes adjacent to the vertices of the edge. This way, an edge is replaced with a new vertex which minimizes the contraction cost. A plane can be represented with a 4D vector p, which contains the plane normal and the distance to the origin. The squared distance of a vertex v to a plane p equals $v^{\mathrm{T}}(pp^{\mathrm{T}})v$. Therefore, the error function for a vertex v is:

$$\Delta(v) = \sum_{p \in planes(v)} v^{\mathrm{T}}(pp^{\mathrm{T}})v = v^{\mathrm{T}} \left(\sum_{p \in planes(v)} pp^{\mathrm{T}} \right) v. \tag{A.1}$$

This value will represent how much that contraction would change the shape of the mesh. It is important to note that a contraction pair is not restricted to a triangle edge, but to a pair of vertices that is closer than a given distance.

A.0.2. Viewpoint mutual information

In [8] a viewpoint-driven simplification method is presented. Its objective is to provide approximations so that the resulting simplified mesh achieves an appearance as similar as possible to the original mesh. To do this, it applies an information-theoretic measure called viewpoint mutual information (VMI) [44,45]. Viewpoint mutual information was introduced to select the best views. An information channel $V \rightarrow O$, called a viewpoint information channel. V and O represent, respectively, a set of viewpoints and the set of polygons of an object. Viewpoints will be indexed by v and polygons by o. The conditional probabilities of $p(o|v)$ are given by the relative area of the projected polygons over the sphere of directions centered at viewpoint v:

$$p(v) = \frac{1}{N_v} p(o) = \sum_{v \in \mathcal{V}} p(v)p(o|v) = \frac{1}{N_v} \sum_{v \in \mathcal{V}} p(o|v). \tag{A.2}$$

The mutual information for a given point is formulated as:

$$I(v, O) = \sum_{o \in \mathcal{O}} p(o|v) \log \frac{p(o|v)}{p(o)}. \tag{A.3}$$

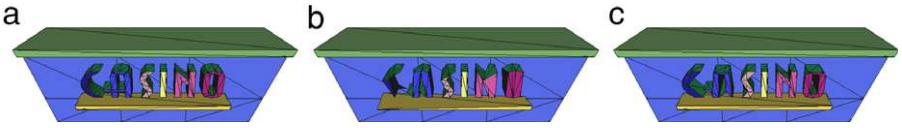


Fig. 26. Original casino model (a) and simplifications of the model to 60% with our method using QSlim quadrics (b) and VMI (c).

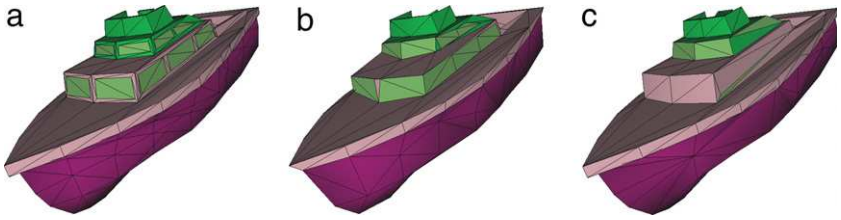


Fig. 27. Original yacht model (a) and simplifications of the model to 50% with our method using QSlim quadrics (b) and VMI (c).

High values mean high degree dependence ("'highly coupled view"') and low values low degree dependence ("'more representative view"').

The error metric is defined as the sum of variations of viewpoint mutual information for all viewpoints V:

$$c = \sum_{v \in \mathcal{V}} \left| I_v - I'_v \right| . \tag{A.4}$$

References

[1] Cignoni P, Montani C, Scopigno R. A comparison of mesh simplification algorithms. Computer Graphics 1998;22(1):37–54.
[2] Luebke D. A developer's survey of polygonal simplification algorithms. IEEE Computer Graphics and Applications 2001;21(3):24–35.
[3] Thakur A, Banerjee AG, Gupta SK. A survey of CAD model simplification techniques for physics-based simulation applications. Computer-Aided Design 2009;41(2):65–80.
[4] Lorensen WE, Cline HE. Marching Cubes: A high resolution 3D surface construction algorithm. Computer Graphics 1987;21(4):163–9.
[5] Lund C.. The SUPERMESH: Run-time meshing from coarse cad models using MSGMESH. In: The MSC 1991 world users' conf. proc., vol. II. 1991. Paper no. 42.
[6] White DR, Saigal S, Owen SJ. Meshing complexity of single part CAD models. In: Proceedings of the 12th international meshing roundtable, Sandia National Laboratories. 2003. p.121–34.
[7] Garland M, Heckbert P. Surface simplification using quadric error metrics. In: SIGGRAPH'97: 24th annual conference on Computer graphics and interactive techniques. 1997. p. 209–16.
[8] Castelló P, Sbert M, Chover M, Feixas M. Viewpoint-driven simplification using mutual information. Computers and Graphics 2008;32(4):451–63.
[9] Lee K. Geometric detail suppression by the Fourier transform. Computer-Aided Design 1998;30(9):77–93.
[10] Lee SH. Feature-based multiresolution modeling of solids. ACM Transactions on Graphics 2005;24(4):1417–41.
[11] Date H, Kanai S, Kishinami T, Nishigaki I. Flexible feature and resolution control of triangular meshes. In: Proceedings of the sixth IASTED international conference on visualization, imaging and image processing. 2006.
[12] Sud A, Foskey M, Manocha D. Homotopy-preserving medial axis simplification. In: Proceedings of the 2005 ACM symposium on solid and physical modeling. 2005.
[13] Luebke D, Reddy M, Cohen JD, Varshney A, Watson B, Huebner R. Level of detail for 3D graphics. San Francisco (CA): Morgan Kaufmann Publishers; 2002.
[14] Ciampalini A, Cignoni P, Montani C, Scopigno R. Multiresolution decimation based on global error. Technical report, Centre National de la Recherche Scientifique. Paris (France); 1996.
[15] Schroeder WJ. A topology modifying progressive decimation algorithm. In: VIS'97: 8th conference on visualization'97. Los Alamitos (CA, USA): IEEE Computer Society Press; 1997. p. 205–12.
[16] Low K, Tan T. Model simplification using vertex-clustering. In: SI3D'97: Proceedings of the 1997 symposium on interactive 3D graphics. ACM Press; 1997. p. 75–81.
[17] Rossignac J, Borrel P. Multiresolution 3D approximations for rendering complex scenes. In: Falcidieno B, Kunii T, editors. Modeling in computer graphics: Methods and applications. Springer-Verlag; 1993. p. 455–65.
[18] Nooruddin F, Turk G. Simplification and repair of polygonal models using volumetric techniques. IEEE Transactions on Visualization and Computer Graphics 2003;9(2):191–205.
[19] Garland M. Multiresolution modeling: Survey & future opportunities. In: State of the art reports of EUROGRAPHICS'99, 1999; vol. 14(4), p. 111–31.
[20] Heckbert P, Garland M. Survey of polygonal surface simplification algorithms. In: Technical report, Multiresolution surface modeling course notes of SIGGRAPH'97, 1997.
[21] Puppo E, Scopigno R. Simplification, lod and multiresolution — Principles and applications. In: Tutorial notes of EUROGRAPHICS' 97, 1997; vol. 16(3).
[22] Hoppe H. Progressive meshes. In: SIGGRAPH'96: Proc. of the 23rd annual conference on Computer graphics and interactive techniques. 1996. p. 99–108.
[23] Hoppe H. New quadric metric for simplifying meshes with appearance attributes. In: VIS'99: Proc. of the conference on visualization'99. IEEE Computer Society Press; 1996. p. 59–66.
[24] Garland M, Heckbert PS. Simplifying surfaces with color and texture using quadric error metrics. In: VIS'98: Proc. of the conference on visualization'98. 1998. p. 263–69.
[25] Cohen J, Olano M, Manocha D. Appearance preserving simplification. In: SIGGRAPH'98, vol. 32. 1998. p. 115–22.
[26] Xu A, Sun S, Xu K. Texture information driven triangle mesh simplification. Computer Graphics and Imaging 2005;43–8.
[27] Chen CC, Chuang JH. Texture adaptation for progressive meshes. In: EUROGRAPHICS'06, Computer graphic forum 2006. vol. 25(3), p. 343–50.
[28] Garland M, Zhou Y. Quadric-based simplification in any dimension. ACM Transactions on Graphics 2005;24(2):209–39. Draft preprint available as Tech report UIUCDCS-R-2004-2450.
[29] Luebke D, Erikson C. View-dependent simplification of arbitrary polygonal environments. In: SIGGRAPH'97. 1997. p. 199–208.
[30] El-Sana J, Varshney A. Generalized view-dependent simplification. Computer graphics forum: Eurographics Association, vol. 18. Blackwell Publishers Ltd; 1999. C83–C94.
[31] Hoppe H. View-dependent refinement of progressive meshes. In: SIGGRAPH'97. 1997. p. 189–98.
[32] Lindstrom P, Turk G. Image-driven simplification. ACM Transactions on Graphics 2000;19(3):204–41.
[33] Luebke DP, Hallen B. Perceptually-driven simplification for interactive rendering. In: Proc. of the 12th Eurographics Workshop on Rendering Techniques, London (UK); 2001, p. 223–34.
[34] Zhang E, Turk G. Visibility-guided simplification. Proceedings of IEEE Visualization 2002;31:267–74.
[35] Williams N, Luebke D, Cohen JD, Kelley M, Schubert B. Perceptually guided simplification of lit, textured meshes. In: Proc. of the 2003 symposium on interactive 3D graphics. ACM Press; 2003. p. 113–21.
[36] Lee CH, Varshney A, Jacobs DW. Mesh saliency. ACM Transactions on Graphics 2005;24(3):659–66.
[37] Castelló P, Sbert M, Chover M, Feixas M. Viewpoint Entropy-driven Simplification. In: Proc. of 15th international conference in central europe on computer graphics, visualization and computer vision 2007. p. 249–56.
[38] Shannon CE. A mathematical theory of communication. CSLI Publications; 1948.
[39] Cover TM, Joy AT. Elements of information theory. 2nd ed Wiley-Interscience; 1991.
[40] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W. Mesh optimization. In: Proceedings of SIGGRAPH'93. 1993. p. 19–26.
[41] Dey T, Edelsbrunner H, Guha S, Nekhayev D. Topology preserving edge contraction. Technical report RGI-Tech-98-018, North California, Raindrop Geomagic Inc., Research Triangle Park; 1998.
[42] Wu JH, Hu SM, Sun JG, Tai CL. An effective feature-preserving mesh simplification scheme based on face constriction. In: Proceedings of the 9th pacific conference on computer graphics and applications, 2001. p. 12.
[43] Vivodtzev F, Bonneau GP, Le Texier P. Topology preserving simplification of 2D non-manifold meshes with embedded structures. The Visual Computer 2005; 21:679–88.
[44] Viola I, Feixas M, Sbert M, Gröller E. Importance-driven focus of attention. IEEE Transactions on Visualization and Computer Graphics 2006;12(5):933–40.
[45] Feixas M, Sbert M, González F. A unified information-theoretic framework for viewpoint selection and mesh saliency. ACM Transactions on Applied Perception 2009;6(1):1–23.