A Tool for the Creation and management of level-of-detail models for 3D applications

OSCAR RIPOLLES, FRANCISCO RAMOS, MIGUEL CHOVER,
JESUS GUMBAU, RICARDO QUIROS
Departamento de Lenguajes y Sistemas Informáticos
Universitat Jaume I,
Campus de Riu Sec, 12071, Castellón
SPAIN
oripolle@uji.es, francisco.ramos@uji.es, chover@uji.es, jgumbau@uji.es, quiros@uji.es

Abstract: - Real-time visualization of 3D scenes is a very important feature of many computer graphics solutions. Current environments require complex scenes which contain an increasing number of objects composed of thousands or even millions of polygons. Nevertheless, this complexity poses a problem for achieving interactive rendering. Among the possible solutions, stripification, simplification and level of detail techniques are very common approaches to reduce the rendering cost. In this paper, we present set of techniques which have been developed for offering higher performance when rendering 3D models in real-time applications. Furthermore, we also present a standalone application useful to quickly simplify and generate multiresolution models for arbitrary geometry and for trees.

Key-Words: - Computer graphics, level of detail, simplification, triangle strips, multiresolution, tree rendering

1 Introduction

Virtual Reality and 3D environments are key technologies within the entertainment industry and multimedia in general. Apart from computer games, there are working environments having multiple applications, like real-time 3D representations of meetings [1] or the Virtual reality exposure therapy (VRET) for treating psychological disorders [2,3].

One of the main problems of interactive graphics applications is the geometric complexity of the scenes they represent. In order to solve this problem, different modeling techniques by level of detail have been developed, trying to adapt the number of polygons of the objects to their importance inside the scene. The application of these techniques is common in standards such as X3D, graphic libraries such as OpenInventor or OSG, and even in game engines such as Torque or CryEngine. Figure 1 shows the ogre head model and an approximation at a medium level of detail which would be rendered much faster.

According to Garland [4] a multiresolution model represents an object through a set of approximations at different levels of detail and allows us to recover any of them on demand. In recent years, multiresolution models have progressed substantially. In the early days, discrete models were employed in graphics applications due mainly to the low degree of complexity involved in implementing them. These models were based on a relatively small number of approximations (normally between 5 and 10) [5]. Nevertheless, the increase in realism in graphics applications made it necessary to find solutions which were more exact in their approximations, which did not call for high storage costs and which had faster visualization times. This fact led to the appearance of continuous models, where two consecutive levels of detail only differ by a few polygons. A comprehensive description of multiresolution models can be found in Ribelles et al. [6].

Fig 1. Original Ogre head model and a less detailed approximation.

Nevertheless, the multiresolution models available nowadays are not always suitable for all kind of meshes. Many of the current interactive applications such as flight simulators, virtual reality environments or computer games take place in outdoor scenes, where the vegetation is an essential

component. The lack of trees and plants can detract from their realism. Tree modeling has been widely investigated [7,8], and its representation is very realistic (see Figure 6). However, tree models are formed by such a vast number of polygons that real-time visualization of scenes with trees is practically impossible, and it is necessary to resort to some method that diminishes the number of polygons that form the object, such as multiresolution modeling. But the multiresolution models that have appeared up to now deal with general meshes and do not work properly with this kind of mesh.

Multiresolution models are widely employed in computer graphics applications as they allow us to reduce the amount of geometry information sent to the graphics system, which results in an improvement in performance. The use of triangle strips in these models offers further improvement, since it adds a compact representation of the connectivity existing in a triangle mesh and enables faster rendering. It is also worth mentioning that one of the key elements for developing a good multiresolution model is the selection of an adequate simplification algorithm to obtain the set of approximations.

In this paper we present a new tool for efficiently creating and managing multiresolution model. GeoTool is a multiplatform, portable and engine independent tool that allows us to manipulate meshes and build multiresolution models for both general meshes and trees. It can also perform more operations such as mesh simplification, (approximate meshes) or stripification (convert a mesh in triangles to triangle strips). We also describe thoroughly the different techniques included in this application for simplifying, stripifying and finally creating the multiresolution models.

2 The GeoTool basic framework

This application uses the FLTK toolkit [9] to provide a portable graphical user interface, and the OpenGL real-time rendering API, see a screenshot in Figure 1. Our application uses the Ogre3D mesh file format [10] to load and store geometry data. This file format supports mesh models composed by any number of sub-meshes. Each sub-mesh can be represented by any rendering primitive (a triangle list or a triangle strip). This is useful to store trees with a LODTree model such as [11], because the trunk must be represented by triangle strips and the leaves by triangle lists. Moreover, the Ogre file format supports bones and skeletal animations.

GeoTool allows the user to perform three different types of operations:

- Basic operations: these operations involve file, edit and render operations. The rendering primitive can be changed (wire mode, solid mode) as well as the lighting surface parameters (flat and smooth). The rendering viewpoint can also be changed in order to focus on the desired region of the model. Moreover, the application can load a previously computed LODStrips [12] or LODTree [11] model and render it.

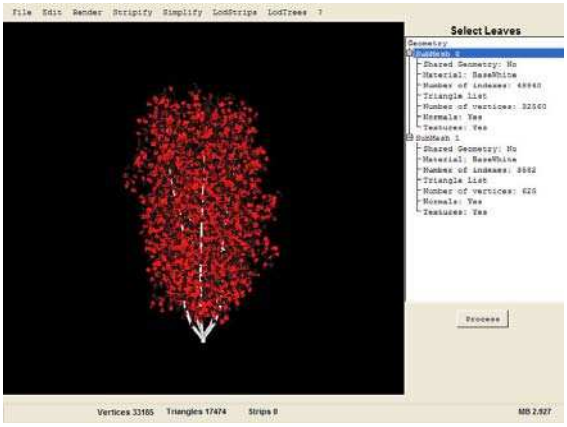

Fig 2. Screenshot of the GeoTeool application.

- Simple operations: these operations involve mesh stripification and simplification. These are catalogued as simple operations because they are done in a single step and return a transformed standalone mesh. Two different simplification approaches are available: geometry-driven simplification and viewpoint-driven simplification.
- Complex operations: these operations are LODStrip construction and LODTree construction. Internally, these complex operations perform some simple operations such as stripification, simplification and vertex reordering. It is important to note that they are much more time consuming than basic operations. They take a mesh as input, then they construct a new mesh with the associated multiresolution sequence, and save the result to disk. We will explain this process in more detail in the following sections.

3 Description of Included Algorithms

We have included two different multiresolution models: a model for general meshes LODStrips and a model for plants and trees LODTree. Both models can make a selection of the LOD in running time in

order to establish a balance between the number of polygons with which the object will be represented and the amount of time needed to visualize it. When modeling vegetation, we will need to do a separate processing. Branches, including the trunk, can be considered as general meshes and will therefore be handled by the general mesh model. Leaves, on the contrary, will be represented using their own specific model.

The construction of a multiresolution model involves the simplification of the original model, in order to obtain the sequence of different approximations that compose the model diminishing the number of polygons while maintaining their appearance. This way, we have developed several algorithms to perform the simplification of the meshes following diverse criteria. Moreover, it is important to mention that the LODStrips model works with a stripified mesh. Therefore, we will also need a method to convert polygonal meshes, which are geometrically composed of triangles, to triangle strips.

2.1 Stripification

The main problem of multiresolution models based on strips arises when, starting from a set of strips representing the initial mesh at maximum detail and applying the successive simplifications, the strips start to include a large quantity of degenerated triangles, which have no mathematical area and imply sending information for triangles that will not be rendered. An example of these low-quality strips can be observed in Figure 3, where the strip in the middle is collapsed after two simplification steps, where edges 0, 3 and 1, 2 are also collapsed.

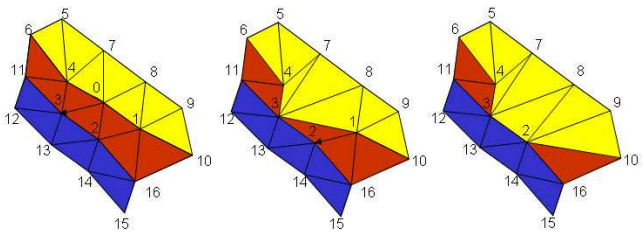

Fig 3. Collapse of a strip.

One possible way to overcome this problem is to use strips which are dynamically generated for each level of detail. Research has been conducted on this approach, and it is possible to find methods of building and maintaining a good set of triangle strips like the one proposed by Stewart [13], and also multiresolution models based on dynamic strips [14,15]. But the additional cost involved in generating the strips for every level of detail is high; therefore the use of static strips, despite their limitations, can turn out to be more suitable.

For our GeoTool application we have selected an algorithm for building triangle strips for static models which avoids working with low quality strips. It constructs the strips from the minimum to the maximum level of detail following the simplification sequence, while maintaining the original appearance of the 3D model [16].

With this intention, the algorithm starts out with the mesh simplified to the minimum level of detail, which means that it may start with just a few triangles. For every step of the algorithm it will need to know the vertices that split, the two new triangles that appear and the set of existing triangles that must be modified. It is possible to collect all this information during the simplification process of the original mesh. The two new triangles share an edge with one of the triangles that will be modified, and it will therefore be through this edge that the algorithm will locate the triangle or triangles where it will be able to insert them. Once the edge is found, it will be necessary to simply insert the new triangle taking care to choose the right side of the edge. If that edge is not found, it will be necessary to create a new strip.

To improve these results, the algorithm has been extended so that, in each step, no repeated vertices or edges and no unnecessary edges are inserted. Furthermore, it is possible to improve the results if, each time a new strip is created, the algorithm tries to insert the new triangle at the end of an already existing strip. On many occasions we have no choice but to insert a new triangle as a new strip. But, depending on how it has been inserted, the algorithm will be able to do the new insertion or not. In order to improve the chances of performing the insertion correctly, a function that predicts the usefulness of the three edges has been developed. This prediction is carried out by following their evolution throughout the remaining refinement steps.

The main contribution of this algorithm is the fact that it improves on the results offered by previous stripification algorithms, since all of them offer low quality for levels of coarser detail.

2.1 Simplification

The applications that can be benefited by using image-based simplification are those in which the main requirement is visual similarity. Examples of such applications are video games, vehicle simulations, building walk-throughs, etc.

In addition, the construction of a multiresolution representation is based on the simplification sequence

that permits the generation of the different levels of detail. This section deals with the simplification methods that GeoTool uses to construct the multiresolution models LODStrips and LODTree.

There are many simplification methods that we could apply [17,18,19] and many metrics could be used [20,21]. The most extended and accurate methods used for surface simplification, as for example [22,23], use techniques based on iterative edge contractions to simplify models, and maintains surface error approximations using quadric metrics. These methods allow us to merge vertices and modify the edges that use these vertices in order to keep the connectivity with the other vertices of the edges.

The GeoTool application includes three different kinds of simplification: geometry-based, viewpoint-based and leaves simplification.

2.3.1 Geometry-based simplification

Among the simplification methods included in our application, we have selected an edge-collapse approach which presents good results for meshes of real applications, which often are composed of many textured submeshes [24].

The methods in the literature offer efficient algorithms to simplify surfaces with a unique mesh. However, if a surface is composed by a set of individual submeshes without any physical interconnection information, the simplification process tends to produce artifacts like holes between the patches. This way, the final simplified model does not preserve the appearance from the original object.

The included method presents also the possibility of a local simplification of the model. That is, if the object is composed by more than a single mesh, each mesh could be simplified to a different level of detail. It must be considered that simplifying a submesh produces a minimum simplification of the neighbor meshes in order to avoid creating holes in the surface.

Using the information stored in the pre-process step, this simplification algorithm allows boundaries preservation, so that when an edge with a vertex in a boundary has to be remapped, the other vertex will overlap the first one. This way, the original shape of the virtual boundary remains unchanged while possible.

Furthermore, at each simplification step texture coordinates for each modified vertex are 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.

In Figure 4 we can see how our implementation improves the quality of the simplified model when the model is composed of a lot of small patches. The image on the left is the original model. The image in the center is the model produced by the QSlim [23] algorithm at 30% of polygons. We can see the holes and artifacts produced during the simplification. The image on the right shows the same model simplified with our improved technique. We can see how all the previous artifacts have disappeared, and the mesh has a smooth transition through all the patches.

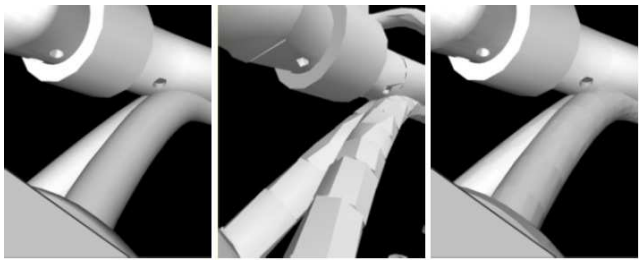

Fig 4. Example of simplification to 30%. From left to right: original model, QSlim[23] simplification and simplification with the selected method.

2.3.2 Viewpoint-based simplification

Most common simplification methods use some technique based on a geometric distance as a quality measure between an original mesh and the one obtained from simplification. By using these methods we can obtain meshes that are visually very similar to the original. The image-based simplification tries to carry out a simplification guided by differences between images more than by geometric distances. The goal is to create simplified meshes that appear similar to a human observer.

By using the image-based simplification we can achieve meshes with a maximum simplification in hidden zones. The information theory has been used as a basis for the elaboration of metrics that have been successfully applied in image-based modeling. For example: the viewpoint entropy that is based on Shannon entropy has been used to compute the best viewpoints of an object.

In this second simplification method that GeoTool includes, the goal is to use new metrics that can be used to evaluate the cost of an edge collapse [20]. To address this, the mutual information has been proposed as a new metric for the elaboration of image-based simplification algorithms. In order to compute the mutual information, we need the number of pixels covered for each visible triangle from a particular camera position. This number will give us the projected area [25].

We compute the error induced by each edge collapse as the difference between the mutual

information of the model before simplifying with the mutual information of the model after simplifying. Then we choose the edge collapse that has the least cost. This error has been calculated as a sum of differences in absolute value of the mutual information for each viewpoint. The initial measurements have been made with 6, 12 and 20 viewpoints. We have checked that the accuracy of the results is better with many viewpoints than with just a few, although the computational cost is higher.

The current implementation completely recalculates in each iteration the errors induced by all the edge collapses. Thus, it always chooses the optimum.

In Figure 5 we can distinguish how the fish keeps the tail shape while several holes appear in the model simplified by Qslim [23].

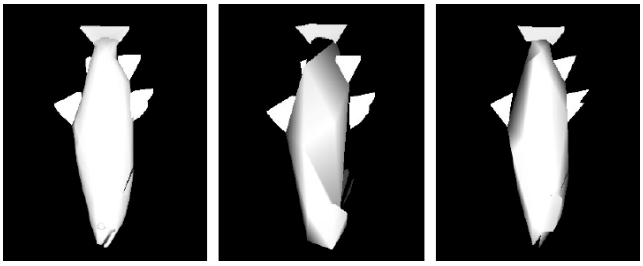

Fig 5. Example of simplification. From left to right: original model, QSlim [23] simplification and simplification with the presented method.

2.3.3 Leaves simplification

Many automatic simplification methods have appeared up to now, as those presented in the previous sections. Nevertheless, when applying them to trees, these obtain acceptable results with the meshes of polygons that represent the trunk and the branches. However, they do not work properly with the foliage. The existing simplification methods generally eliminate polygons, so that the appearance of the crown after an automatic process of simplification is that it has been pruned. The number of leaves is less than before, so the tree appears less leafy. The images obtained with these methods are not very realistic and for this reason it is necessary to introduce new solutions.

The method included for the automatic simplification of foliage diminishes the number of polygons that form the crown, while maintaining its leafy appearance. The key to the algorithm is leaf collapse. Two leaves are transformed into a single one, so that the area of the new leaf is similar to the area formed by the two leaves initially. An error function is the way of determining the pair of leaves that will be simplified to create a new one. [26]

This method of simplification repeatedly selects a pair of leaves, which minimizes an error function. These leaves disappear and a new one is obtained. The collapsed leaves are eliminated from the list of candidates, and next, the new leaf is evaluated with the leaves that remain in the foliage.

The simplification method is characterized by two elements:

- The measurement that specifies the cost of collapsing two leaves, combining distance and planarity between the pair of evaluated leaves.
- The position of the vertices that form the newly created leaf, selecting two vertices of each of the collapsed leaves. More precisely, the two vertices of each leaf that are furthest from the other leaf would be chosen. This method will allow us to maintain an area similar to the two original leaves.

Figure 6 presents an image composition of three simplified tree versions according to the distance to the viewer. It can be observed that the images obtained maintain the appearance although their leaf number diminishes.

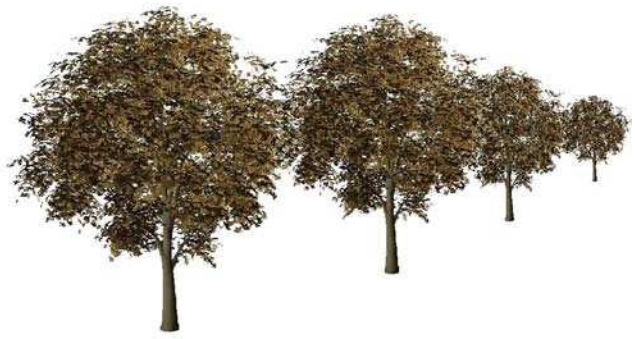

Fig 6. Composition of different approximations of the *Aesculus Hippocastanum* following the distance-to-the-viewer criteria.

2.3 Multiresolution modeling for general meshes

The continuous uniform resolution model we include in GeoTool noticeably improves existing models in terms of storage and visualization costs. The model is based entirely on optimized hardware primitives, triangle strips, and it is conceived in such a manner that mesh updating is fast and efficient [12].

The construction of this multiresolution model is made up of a set of different processes. First of all, as we are working with a model based on triangle strips, we will need to have a stripified mesh. We will also need to choose a simplification method for our 3D

model, since we need the simplification information to generate the different levels of detail. The stripification method has been explained above, and different approaches for simplifying a mesh have also been addressed. With the information about the strips and the simplification sequence, the real construction of the model begins by filling the data structures with all the required information.

In this process, vertices are reordered according to the simplification order, that is, the first vertex to be collapsed will be zero, the second will be one, and so on. Obviously, it will also be necessary to modify the strips according to the changes made. This step also stores the ordered vertices and the triangle strips within the model data structures.

It is also important to note that the construction process computes and stores the strips that change for each level of detail and the exact location of the vertex to be simplified in every strip. This allows the levels of detail in the model to be crossed rapidly, offering optimum performance. This is the information that enables a fast level of detail extraction time and makes this multiresolution model quick and efficient. Finally, the method takes into account the existence of degenerated triangles and applies filters to avoid their appearance in the triangle strips.

The presented LODStrips model offers a fast LOD extraction which allows us to obtain smooth transitions between LODs, as well as very good rendering times, because extraction is usually an important part of the total rendering time.

2.4 Multiresolution modeling for trees

The construction of the multiresolution model for a tree entails two different processes. On the one hand, the trunk of the tree is formed by a continuous mesh, classified inside the arbitrary surfaces, and its multiresolution model will be built using the LODStrips model we have just presented above. On the other hand, as tree foliage is represented by a set of isolated polygons, most of the existing multiresolution models are not appropriate for its visualization [27,28]. The multiresolution model for foliage has been included in order to allow the interactive visualization of the crown of trees in real-time applications [11].

In order to change the level of detail, two geometric operations have been defined, as shown in Figure 7:

- Leaf collapse: Decreases the level of detail of a given representation. This operation replaces two leaves by a new one, diminishing the number of leaves of the crown. Following the example in Figure 7, leafs l_t and l_u will be replaced with l_s in the new level of detail.
- Leaf split: It is the inverse operation of the previous one. It replaces a leaf by the two leaves that it represents, increasing the level of detail. Following the same example, l_s will be replaced with l_t and l_u.

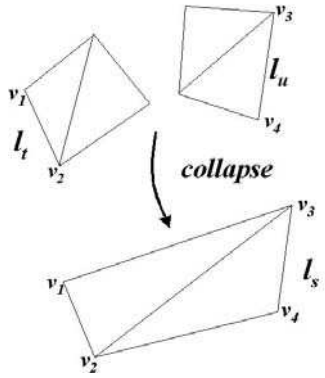

Fig 7. Example of leaf collapse and split operation.

The data of the multiresolution model are arranged in *forests* of binary trees. This structure reflects the way the simplification operation works, which establishes a hierarchical relation between leaves. As shown in Figure 7, each new leaf, l_s, is created from two existing ones, l_t and l_u.

The data structure of LODTree allows us to visualize many trees with different levels of detail, sharing the same data. The multiresolution model for foliage has proven to be useful to render foliage with uniform resolution, where the level of detail can be changed in real time and the number of leaves can be adjusted to meet the requirements of the application.

4 The GeoTool application

Once the different techniques included in the application have been described, we can start with providing thorough details of the GeoTool. The user interface of GeoTool has been designed to be easy to use. The menu bar across the top manages all the operations that can be performed on a mesh.

The main window in the center shows the current render state, which can be changed using the Render menu. The panel on the right shows a more detailed view of the current selected action. For example, when the simplify option becomes selected, the panel on the right shows more information and options about the selected action.

The status bar on the bottom of the application shows some information about the loaded model, such as its vertices, strips and triangle count.

4.1 Basic Operations

- File Menu (see Fig. 8)

  Open: shows a dialog to open an Ogre mesh file and load it into the application.

  Save (As): Saves the current mesh into an Ogre mesh file.

  Load Textures: Allows selecting the texture of the entire model. In addition, it allows us to select the texture of for a single submesh.

  Quit: Terminates the application.
- Edit Menu

  Undo: Gets the current mesh back to its previous state.

  Fit: Modifies the current view to fit the loaded mesh inside the screen.

  Rotate/Pan: Selects the action to be taken when the user drags the mouse pointer.

  Mesh info: Configures the right panel to show mesh information, such as its vertex and triangle counts, the rendering primitive type and its sub-mesh count.

  Select leaves: Configures the right panel to show a sub-mesh selector. This allows the user to select the sub-mesh that represents the treetop. Pushing the process button the foliage is selected.
- Render Menu (see Fig. 9)

  Wire / Solid: Selects the geometry rendering mode: wireframe or solid.

  Flat / Smooth: Selects the surface **shading mode**: flat or smooth (Gouraud).

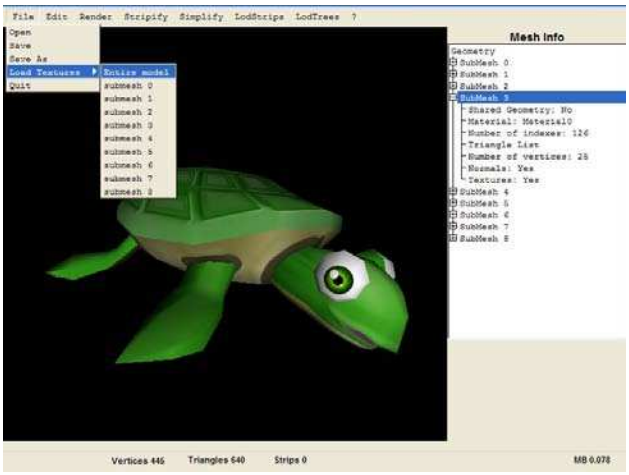

Fig 8. The File Menu.

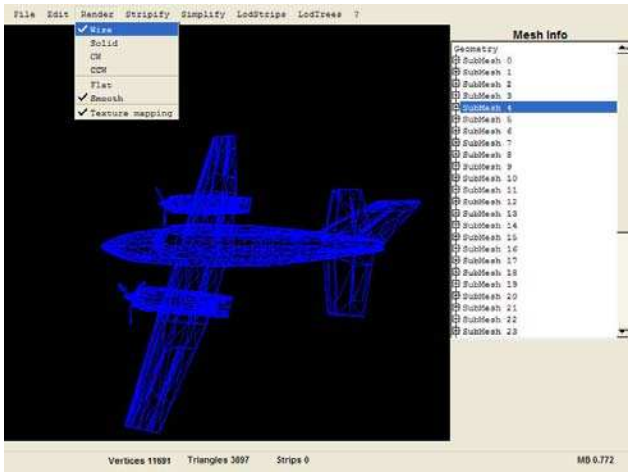

Fig 9. The Render Menu.

4.2 Simple Operations

The simple operations are performed in a single step of the Geometry Game Tools Library and modify the given mesh.

- Stripify Menu. (See Fig. 10). This menu has no popup menu associated. Instead, it immediately opens the Stripification panel. The process button will start the stripification. The progress bar will show the stripification status. Only manifold meshes can be stripified. If a mesh is not manifold an error message is shown.
- Simplify Menu. (See Fig. 11). The simplification of a mesh object can be accomplished with one of the two following simplification modes:

  Mesh simplification: Performs edge collapse simplification. Moreover, there are two ways to perform simplification in edge collapse.

  Geometry-based: Fast method of simplification. Simplify mesh objects based on geometry relation between triangles.

  Viewpoint-based: Simplify the mesh object based on image processing.

  Leaves simplification: Performs leaves collapse simplification. Only simplifies the mesh that was chosen as foliage with the Edit/Select Leaves menu.

In addition, the mesh reduction factor can be chosen as a percentage value or as a number of vertices.

4.3 Complex Operations

The complex operations visually need several of simple geometry operations.

4.3.1 LODStrips

- **Generate:** Generate a mesh with its correspondent LOD (Level Of Detail) sequence. The process to accomplish this is similar to the simplification, but a new step is added. In the right panel appear a new Build button that performs the stripification and the build process. The build process gets the simplification sequence done by the simplification and the stripified model, then generates the new mesh with its correspondent LOD sequence (see Fig. 12).
- **Visualize:** This allows us to see the result of the LODStrip process. The level of detail of the object can be changed with a slide bar that appears in right panel (see Fig 13).

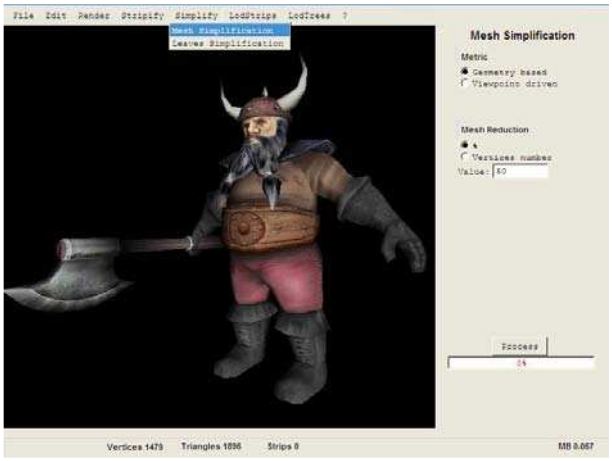

Fig 10. The Simplify Menu.

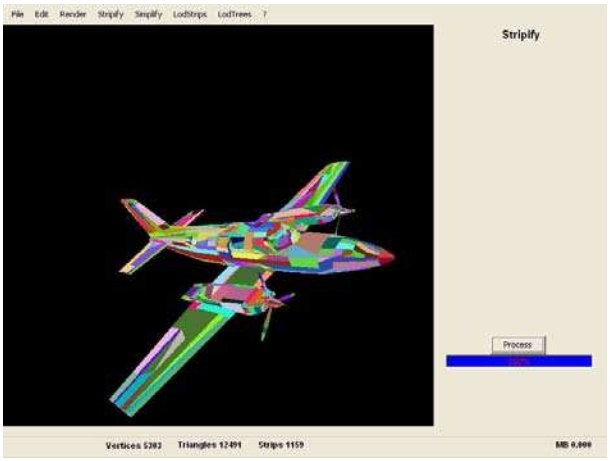

Fig 11. The Stripify Menu.

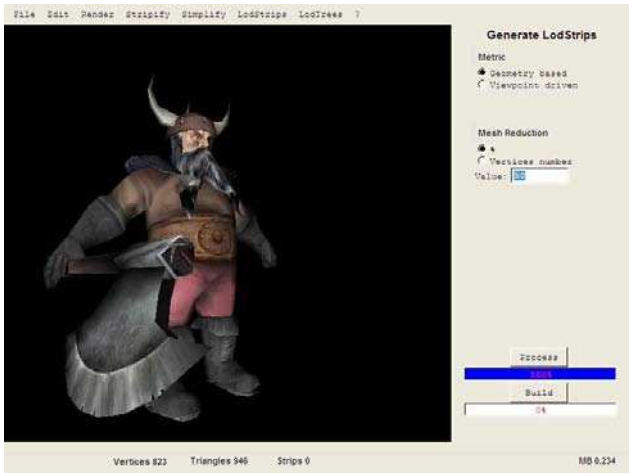

Fig 12. Generation of the multiresolution model LODStrips.

4.3.2 LODTrees

- **Generate**: This process gets a tree object with the LODStrip of the trunk. The foliage should be selected the *Edit/Select Leaves* menu. The user interface operation is the same that the LODStrips one. First, simplify the foliage selecting the percent or the number of triangles desired. Finally pushing the build button the LODTree is generated.
- **Visualize**: Two slidebars appears in right panel. One to change the level of detail of the trunk and the other to change the level of detail of the foliage (see Figure 14).

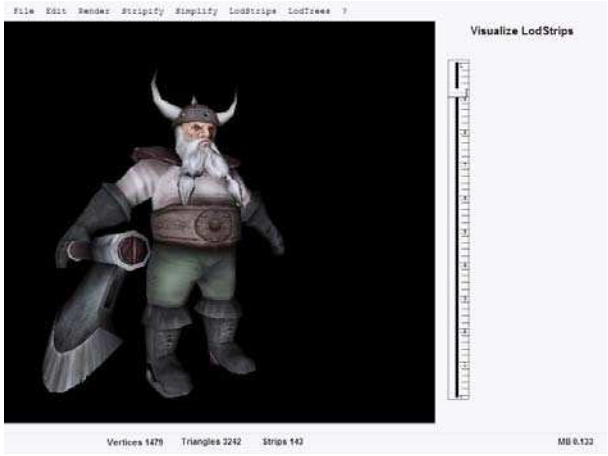

Fig 13. Visualization of LodStrips.

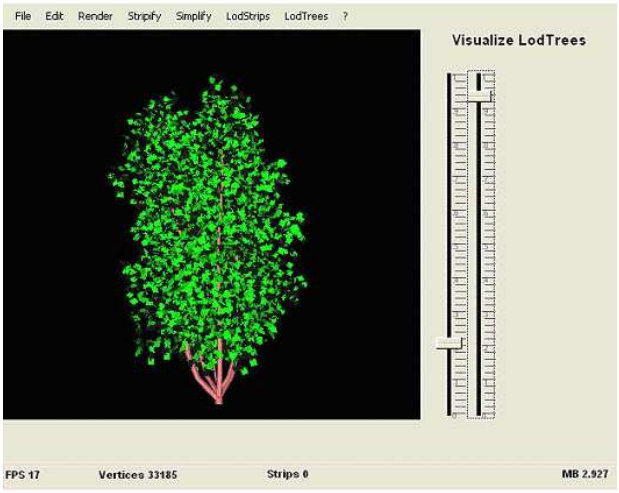

Fig 14. Visualization of LODTree.

4.4 Level of Detail Generation

This section explains the steps that are necessaries to build a LOD object. The LOD generation involves mesh stripification and mesh simplification.

The first step is to open a mesh object. We do this with the **Open** option of the **File** menu. Next step is select option **Generate** of the **LODStrip** menu to begin the generation process (see Fig. 12). At right appears the **Generate LODStrips** panel, in which we can choose the simplification method between **Geometry based** and **Viewpoint Driven**. In addition, we must specify mesh reduction by percent or by the number of triangles to obtain. Pushing the **Process** button the simplification is performed and its result can be viewed at screen. At this point, if the result of simplification doesn't convince us, we can undo this step and change simplification options. In the other hand, if the results of the simplification are OK, we can build the LOD object pushing the **Build** button and filling the name of this one. If we can view the result of build process can do this by selecting the option **Visualize** of the menu **LODStrips** that makes a slide bar appear in the right panel (see Fig. 13). By moving the slide bar we can change the Level Of Detail of the object.

First of all to perform **LODTree** we must select the **Select Leaves** option of the Edit menu (see Fig. 2). This makes that the **Select Leaves** panel appears. In this panel there is a mesh info browser that shows the submeshes in which is divided the object and relevant geometric information. Clicking a submesh in the browser paints it in red in the view window. We must choose the treetop submesh and push the button Process to make the selection of leaves. Next we can go to do the LODStrip procress as described before (see Fig. 12). Finally the result of the LODStrip process is used as input of the LODTree one. Using the **LODTree** menu we can perform the LODTree building following the same steps than a LODStrip process.

5 Conclusions

We have presented a tool for efficient geometry processing of meshes. It allows us to simplify and convert a mesh into triangle strips. Moreover, it is capable of generating a multiresolution model for arbitrary geometry and for trees and plants.

The different algorithms included in this application have also been described, in order to offer details of the resulting meshes we can obtain from them. Furthermore, the selection of the different approaches for simplifying, stripifying or developing multiresolution meshes has also been justified.

Acknowledgements

This work has been supported by grant P1 1B2007-56 (Bancaixa), the Spanish Ministry of Science and Technology (Contiene Project: TIN2007-68066-C04-02) and FEDER funds.

References:

[1] Nijholt, A., 2005. Meetings in the virtuality continuum: Send your avatar. Proceedings International Conference on CYBERWORLDS. Singapore, pages 75–82. IEEE Computer Society.

[2] Regenbrecht, H. et al 2006. Collaborative mixed reality exposure therapy. Proceedings International Conference on CYBERWORLDS. EPFL Lausanne, Switzerland, pp. 25–32. IEEE Computer Society.

[3] Nazrita Ibrahim, Mustafa Agil Muhamad Balbed and Azmi Mohd Yusof, Virtual Reality Approach in Treating Acrophobia: Simulating Height in Virtual Environment, WSEAS Transactions on Computers, vol 5(7), pp. 511-518, 2008.

[4] Garland, M. Multiresolution modeling: Survey & Future opportunities. State of the Art Reports of EUROGRAPHICS'99, pp 111-131, 1999.

[5] Evans, F., Skiena, S., Varshney, A. Efficiently generating triangle strips for fast rendering. Technical report, Department of Computer Science, State University of New York at Stony Brook, 11794-4400, 1997.

[6] Ribelles, J., López, A., Belmonte, O., Remolar, I., Chover, M. Multiresolution modeling of arbitrary polygonal surfaces: a characterization. Computers

& Graphics, vol. 26:3, pp. 449-462, ISSN 0097-8493, 2002.
[7] P. Prusinkiewicz and A. Lindenmayer. The algorithmic beauty of plants. Springer-Verlag New York, Inc., 1990.
[8] B. Lintermann and O. Deussen. Interactive modeling of plants. IEEE Computer Graphics Application, 19(1):56–65, 1999.
[9] Fast Light Toolkit. http://www.fltk.org/.
[10] Ogre3D: http://www.ogre3D.org/.
[11] I. Remolar, C. Rebollo, M. Chover, J. Ribelles, Real-Time Tree Rendering, ICCS 2004, vol. 3039, pp. 173-180, June, 2004.
[12] F. Ramos, M., Chover. LodStrips. Lecture notes in Computer Science, Proc. of Computational Science ICCS 2004, Springer, ISBN/ISSN 3-540-22129-8, Krakow (Poland), vol. 3039, pp. 107-114.
[13] J. Stewart, Tunneling for Triangle Strips in Continuous Level-of-Detail Meshes. Graphics Interface, 2001, pp. 91-100.
[14] Hoppe, H. Progressive messhes. ACM SIGGRAPH 1996, pp 99-108.
[15] M. Shafae, R., Pajarola, Dstrips: Dynamic Triangle Strips for Real-Time Mesh Simplification and Rendering. Proceedings Pacific Graphics Conference 2003.
[16] O. Ripollés, M. Chover, J. F. Ramos, Quality Strips for Models with Level of Detail, Proc. of Visualization, Imaging, and Image Processing (VIIP), 2005.
[17] D.P. Luebke. A developer's survey of polygonal simplification algorithms. IEEE Computer Graphics and Applications, 3(24):24–35, 2001.
[18] P. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. Technical report, Multiresolution Surface Modeling Course Notes of SIGGRAPH'97, 1997.
[19] Pai-Feng Lee, Bin-Shyan Jong, Point-based Simplification Algorithm, WSEAS Transactions on Computer Research, vol 1(3), pp. 61-66, 2008.
[20] P. Castelló, M. Sbert, M. Chover, M. Feixas, Techniques for Computing Viewpoint Entropy of a 3D Scene, ICCS 2006, University of Reading, UK, May, 2006.
[21] R. Alvarez, L. Tortosa, J. F. Vicent, A. Zamora, Error measurements and parameters choice in the GNG3D model for mesh simplification, WSEAS Transactions on Information Science & Applications, vol. 5 (5), pp. 579-588, 2008.
[22] E. Puppo and R. Scopigno. Simplification, lod and multiresolution - principles and applications. Tutorial Notes of EUROGRAPHICS'99, 16(3), 1997.
[23] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, pp. 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.
[24] C. Gonzalez, J. Gumbau, M. Chover, and P. Castello. Mesh simplification for interactive applications. In WSCG, 2008.
[25] P. Castelló, M. Sbert, M. Chover, M. Feixas, Viewpoint Entropy-Driven Simplification, WSCG, ISBN 978-80-86943-01-5, 2007.
[26] I. Remolar, M. Chover, Ó. Belmonte, J. Ribelles, C. Rebollo, Geometric Simplification of Foliage, Eurographics'02 (Short Presentations), ISBN/ISSN 1017-4565, pp. 397-404, 2002
[27] O. Deussen, C. Colditz, M. Stamminger, and G. Drettakis. Interactive visualization of complex plant ecosystems. In VIS '02: Proceedings of the conference on Visualization '02, pages 219–226. IEEE Computer Society, 2002.
[28] A. Jakulin. Interactive vegetation rendering with slicing and blending. In A. de Sousa and J.C. Torres, editors, Proc. Eurographics 2000 (Short Presentations), 2000.