MEPP - 3D MESH PROCESSING PLATFORM

Guillaume Lavoué [1], Martial Tola[2] and Florent Dupont[2]
[1]*Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France*
[2]*Université de Lyon, CNRS, Université Lyon 1, LIRIS, UMR5205, F-69622, France*
{*guillaume.lavoue,martial.tola,florent.dupont*}*@liris.cnrs.fr*

Keywords: Mesh processing, open source, platform.

Abstract: This paper presents MEPP, an open source platform for 3D mesh processing. This platform already contains a large set of processing tools from classical ones (simplification, subdivision, segmentation) to more technical algorithms (compression, watermarking, Boolean operation, perceptual metrics, etc.). Its main objective is to allow a quick start for both users and developers by providing highly detailed tutorials and simple integration mechanisms, through a modular architecture where components are implemented as dynamic plugins.

1 INTRODUCTION

Technological advances in the fields of telecommunication, computer graphics, and hardware design during the last decade have contributed to the emergence of three-dimensional (3D) data, often represented by polygonal meshes, in numerous industrial domains like mechanical engineering, scientific visualization, digital entertainment or cultural heritage. This emerging type of data is more complex to handle than other media such as audio signals, images or videos, and thus has brought new challenges to the scientific community, and has open new research domains like geometry processing.

To ease the development of new algorithms for 3D mesh processing, it is crucial to have some open source tools and libraries available for the scientific community. The library CGAL (The CGAL Project, 2011) (Computational Geometry Algorithms Library) introduced for several years, offers powerful geometry processing algorithms and data structures like the *Polyhedron* type for manipulating manifold polygonal meshes, however it is based on template programming and thus is not so easy to use for a beginner. Some open source softwares have also been introduced like MeshLab (Cignoni et al., 2008), Graphite (Graphite, 2003) or OpenFlipper (Möbius and Kobbelt, 2010), based on their own data structure for 3D mesh manipulation; they offer interesting geometry processing for user and also allow developers to add plugins with their own code, more or less easily.

We introduce the open source platform MEPP (3D MEsh Processing Platform) which can be seen as complementary to these existing works. It is available online [1] and its strong points regarding existing works are as follows:

- it focuses on making the use and installation very easy, whichever the operating system, by providing highly detailed user and developer tutorials; the objective is to allow a quick start for new developers,
- it simplifies the use of CGAL data structures by hiding the template programming,
- it provides not only standard modeling tools (simplification, subdivision, segmentation) but also highly technical components like progressive compression, watermarking, Boolean operation and perceptual metrics,
- it allows the visualization and processing of color meshes and dynamic mesh-sequences.

Next sections present respectively the main architecture of the platform, its manipulation and visualization functionalities and a rapid description of currently available components.

2 ARCHITECTURE

This section describes successively the goals of our platform, its kernel and related features and then the principles of component / plugin development.

2.1 Goals

MEPP provides a GUI and basic functionalities to allow development and fast integration and can also be

[1] http://liris.cnrs.fr/mepp/

easily enriched with new features and new modules. The platform provides a modular architecture through the use of components available as dynamic plugins.
As it is important that all developments have a minimal impact compared to an operating system to provide a great portability, we have directed our choices to use robust, flexible and effective C++ cross-platform open source libraries.
In order that users and developers can easily use and enhance the platform, a major effort has focused on the fact that it can be deployed quickly and easily, thanks in particular to a detailed installation documentation. The integration of new developments is facilitated by a step by step documentation on the use and enrichment of the platform.

2.2 Kernel and related features

Based on CGAL[2], Qt[3], libQGLViewer[4], OpenGL[5], Boost[6] and FFmpeg[7], the MEPP platform, written in C++ under the GNU GPL v3, runs on Windows, Linux and Mac OS X. It provides a development environment based primarily on the class *Polyhedron* of CGAL, library based on the principles of Object-Oriented Programming and especially parametric polymorphism, i.e. template.
Without any component, the MEPP kernel only allows loading, viewing and saving mesh and mesh sequences. The loading of mesh can be done by using the menu or by drag and drop from one application window or from a file browser.
MEPP allows the management of multiple objects in one or more windows and offers two types of processing and display for the meshes:

- the ”space” mode in which several objects are treated in the same scene, allowing to compare them, to observe them by coupling their rotation, or to assess the results of treatments (Figure 1),
- the ”time” mode in which several objects are seen as a sequence of meshes and visualized using a 3D+t configurable *video recorder* (step by step, loop, reverse, speed, etc.).

At any time, users can easily switch from one mode to another.
The platform offers an OpenGL accelerated display with ”Display Lists” in order to allow the use of all graphics cards. The rendering is possible as cloud of points, wireframe and conventional 3D with or without reinforcement of points and edges. Three color modes are available: ”vertex color”, ”face color” or ”material”.
Meshes related to OBJ (Wavefront), OFF (Object File Format), PLY (Polygon File Format), SMF (3D World Studio) and X3D (XML-based free format) can be read. For now, only the formats OBJ (Wavefront) and OFF (Object File Format) can be written. Image and video capture (H.264/MPEG-4 AVC) are also available.
The management of the mesh is also done within the MEPP kernel, ie its memory storage and access but also basic functionalities associated (number of components and boundaries, normals, bounding box, degree, valence, genus, tags, etc.). The kernel also provides to all components a ”Viewer” object that manages the display and behavior of the object ”Scene” according to the mode (”normal”, ”space” or ”time”), itself containing the object or objects *Polyhedron* (Figure 2).
The structure of underlying data used in the platform to represent a mesh is based on the concept of semi-oriented edges, ie half-edges, with relations of incidence and adjacency. It restricts the class of representable mesh with those of type manifold, with or without borders; we call it a polyhedron.
One of the big advantages of MEPP, however, is the fact that the definition of this polyhedron only appears rarely to the developer eyes, in order to best abstract the CGAL library.

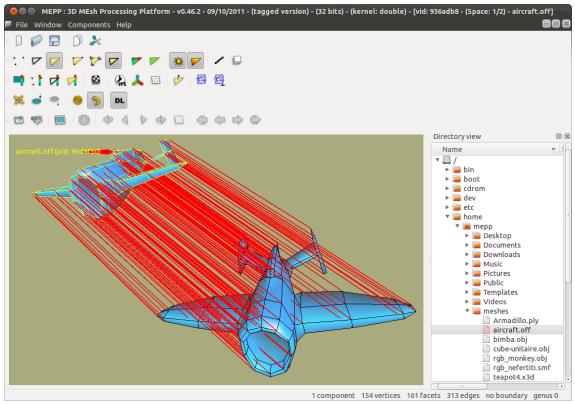

Figure 1: The interface of MEPP, here in ”space” mode.

2.3 Component / plugin development

Through the principle of inheritance, the polyhedron and its geometric items (vertices, edges, facets) can be enhanced, new classes are derived from these existing classes, which gives the developer the opportunity to add his own data and associated functions.

[2] http://www.cgal.org/
[3] http://qt.nokia.com/
[4] http://www.libqglviewer.com/
[5] http://www.opengl.org/
[6] http://www.boost.org/
[7] http://ffmpeg.org/

Given this behavior and for modularity, the solution that we have chosen for MEPP is the multiple ”conditional” inheritance, each component can thus enrich the geometric items (vertices, half-edges, facets) and the polyhedron itself.
The resulting enriched polyhedron and its geometrical items, therefore inherit of all the enhancements brought by the inheritance of each respective components implemented as plugins (.dll / .so / .dylib) with selection at compilation and detection and automatic loading at runtime.
This principle of plugin also allows us to enhance the MEPP GUI (menus and toolbars) and at the same time to resolve the problem of heterogeneous licenses (free component, private component with more restrictive license, etc.). In order to manage interactions, events related to the interface (pre-draw, post-draw, mouse movements and clicks, key press, etc.) are transmitted from the kernel to components through a signal mechanism specific to Qt (Figure 2). MEPP is entirely gen-

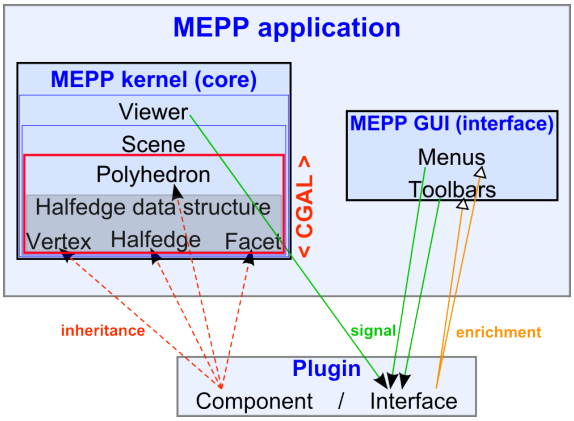

Figure 2: MEPP architecture.

erated by the cross-platform open-source build system CMake[8]. On Linux and Mac OS X, a prerequisite of packages (specific to the distribution for Linux, Homebrew[9] or MacPorts[10] for Mac OS X) is required. On Windows, these packages are provided as ”binary kits” precompiled for 32 and 64 bit architectures. A Linux virtual machine ”ready to start” is also available, offering to start even faster in developing.

3 COMPONENTS

This section provides an overview of currently available components of our platform.

[8]http://www.cmake.org/
[9]http://mxcl.github.com/homebrew/
[10]http://www.macports.org/

3.1 Basic processings

Some basic processing operations are provided:

- basic mesh manipulation (triangulation, noise addition, smoothing, scaling, rotation, translation),
- subdivision algorithms (Loop, Sqrt3, Doo-Sabin, Catmull-Clark, Quad triangle),
- simplification algorithms (Lindstrom-Turk implementation from CGAL and a canonical vertex removal algorithm).

Figure 3 illustrates a result of simplification.

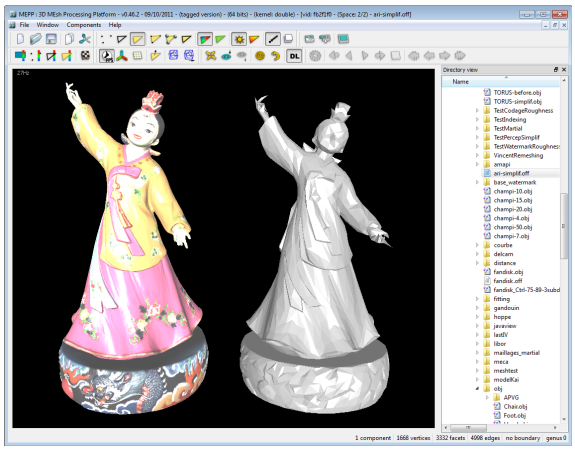

Figure 3: Two objects in the same space: the original colored mesh (1,2M vertices) and result after simplification.

3.2 Curvature and segmentation

Our platform provides the curvature calculation implementation of Cohen-Steiner and Morvan (Cohen-Steiner and Morvan, 2003), with a geodesic neighborhood integration. It computes both min-max curvature values and directions. A segmentation algorithm is also provided, implementing the Variational Shape Approximation from Cohen-Steiner et al. (Cohen-Steiner et al., 2004). Results from curvature calculation and segmentation are illustrated in figure 4.

3.3 Boolean operation

A fast and exact Boolean operation algorithm between 3D meshes is implemented (Leconte et al., 2010); this algorithm is able to compute the union, intersection and difference between two 3D meshes. One of the key feature is the speed, indeed for instance this algorithm is able to compute the intersection, union or difference between two 80K vertices meshes in about 2.5 seconds on a 2GHz processor. Figure 5 illustrates some Boolean operations between two 3D models.

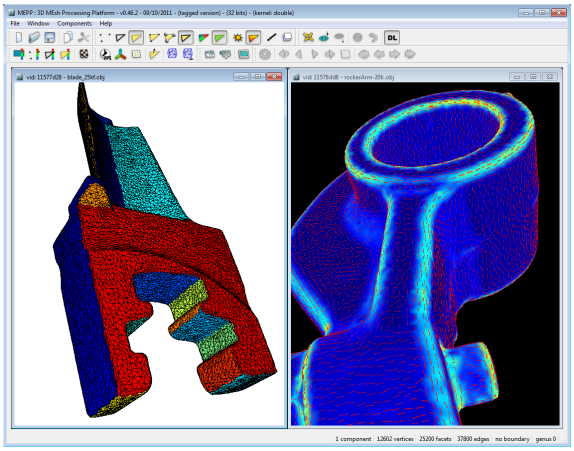

Figure 4: Segmentation of the Blade model and curvature values and directions of the RockerArm model.

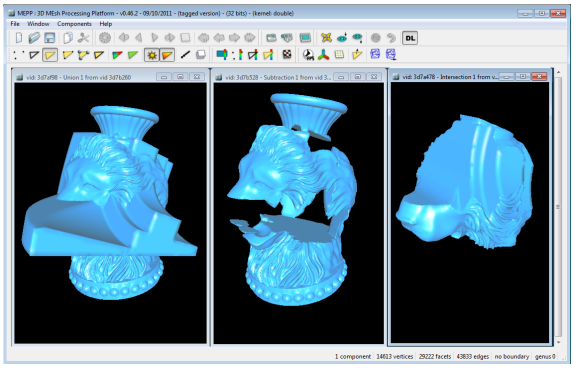

Figure 5: Results after union, difference and intersection between the LionVase and the Fandisk objects.

3.4 Perceptual quality metrics

Recent perceptual metrics are implemented in the platform (Lavoué, 2011); given a distorted 3D shape and a reference one, they compute a score that predicts the perceived distortion between them, as well as a distortion map. These perceptual quality metrics may be more relevant than classical root mean square distance to evaluate or drive processing operations. To our knowledge, no open platform currently proposes such tools.

3.5 Compression and watermarking

The MEPP platform proposes a recent progressive compression algorithm applying for colored meshes (Lee et al., 2011b), and a join watermarking scheme (Lee et al., 2011a). The compression method provides high compression ratio and allows a decompression by Level-of-Details (LoD). When the watermarking method is activated, a secret bit string is hidden in each level-of-details, it allows the protection of the decoded LoD.

3.6 Minkowski sum

Finally, our platform provides an algorithm for Minkowski sum of convex polyhedra (Barki et al., 2009). The algorithm is exact and quite fast regarding its state-of-the-art counterparts.

4 CONCLUSIONS

We have presented the Mesh Processing Platform MEPP; its main objective is to ease the development of new processing algorithms for 3D meshes. This platform is mostly based on the *Polyhedron* type from CGAL, however one of the main advantages of MEPP, is the fact that the definition of this polyhedron structure is hidden to the developer eyes; hence this will allow us, in the next future, to integrate other data structures, for instance supporting non-manifold meshes; we also plan to integrate the management of textured meshes.

REFERENCES

Barki, H., Denis, F., and Dupont, F. (2009). Contributing vertices-based Minkowski sum computation of convex polyhedra. *Computer-Aided Design*, 41(7):525–538.

Cignoni, P., Callieri, M., and Corsini, M. (2008). Meshlab: an open-source mesh processing tool. In *Eurographics Italian Chapter Conference*.

Cohen-Steiner, D., Alliez, P., and Desbrun, M. (2004). Variational shape approximation. In *ACM Siggraph*, pages 905–914.

Cohen-Steiner, D. and Morvan, J. (2003). Restricted delaunay triangulations and normal cycle. In *19th Annu. ACM Sympos. Comput. Geom.*

Graphite (2003). http://www.loria.fr/ levy/Graphite/index.html.

Lavoué, G. (2011). A Multiscale Metric for 3D Mesh Visual Quality Assessment. *Computer Graphics Forum*, 30(5):1427–1437.

Leconte, C., Barki, H., and Dupont, F. (2010). Exact and Efficient Booleans for Polyhedra. Technical report.

Lee, H., Dikici, C., Lavoué, G., and Dupont, F. (2011a). Joint reversible watermarking and progressive compression of 3D meshes. *The Visual Computer*, 27(6-8):781–792.

Lee, H., Lavoué, G., and Dupont, F. (2011b). Rate-distortion optimization for progressive compression of 3D mesh with color attributes. *The Visual Computer*.

Möbius, J. and Kobbelt, L. (2010). OpenFlipper: An Open Source Geometry Processing and Rendering Framework. In *Curves and Surfaces*.

The CGAL Project (2011). *CGAL User and Reference Manual*. CGAL Editorial Board, 3.9 edition. http //www.cgal.org/.