OpenTissue - An Open Source Toolkit for Physics-Based Animation

Kenny Erleben[1], Jon Sporring[1], and Henrik Dohlmann[1,2]

[1] Department of Department of Computer Science, University of Copenhagen, Denmark, {kenny,sporring}@diku.dk,

[2] 3D-Lab School of Dentistry, Dept. of Pediatric Dentistry, University of Copenhagen, Denmark, henrikd@3dlab.dk.

Abstract. OpenTissue is a multidisciplinary, open source programming toolkit for physics-based simulation, collision detection, scientific visualization, and medical imaging. The toolkit was initiated in 2001, is driven by academia, and is used by several universities. It implements several cutting edge simulators using cutting edge programming techniques. OpenTissue is available under Windows and Linux, and it implements almost all algorithms presented in the accompanying book [1]. This presentation will give an overview of OpenTissue: what it can do, how it is done, and where the toolkit is going.

1 History of OpenTissue

OpenTissue goes back to November 2001, where K. Erleben, H. Dohlmann, J. Sporring, and K. Henriksen collected code-pieces of their own work. The ambition was to ease project collaboration and teaching efforts. In the beginning, OpenTissue served as a playground for our experiments, but soon OpenTissue also provided a valuable tool for student projects and for communication with colleagues. Hence, OpenTissue was released in August 2003 under the terms of the GNU Lesser General Public License (gnu.org) with the original four authors as moderators. We specifically chose the Lesser General Public License, since we wish to make the OpenTissue available to industry partners in the medical, gaming, and film industry. Today OpenTissue works as a foundation for research, student projects, and industry collaborations in physics-based animation at the Department of Computer Science, University of Copenhagen, and the toolkit is freely available through our servers (www.opentissue.org) for Windows and Linux.

2 Toolkit Overview

Due to the experimental nature of OpenTissue, OpenTissue contains a wide variety of algorithms and data structures. Some code-pieces are merely wrappers of third party software ensuring that the programmers can use these tools within the common framework. Other code-pieces consist of tools extracted from our own research projects in soft tissue simulation and medical image data acquisition. Finally much code is direct implementation of theoretical concepts discussed in a recent book [1], and have been used for several years to teach physics-based. Examples of some of the more advanced implementations included in OpenTissue are:

Multibody Animation:

- Constraint Based Animation: [1,7], see Figure 1(a)
- Shock Propagation: [1,7], see Figure 1(b)
- Complementarity Problem Solvers: [1,7], see Figure 1(c)

Deformable Objects:

- Finite Differences: [1–3], see Figure 1(d)
- Finite Element Methods: [1,4], see Figure 1(e)
- Particle Systems: [1,5], see Figure 1(f)
- Computational Fluid Dynamics: [1,6], see Figure 1(g)

Collision Detection:

- Spatial Coherence and self-intersection: [1,7], see Figure 1(h)

Geometry Generation:

- Thin-shell generation: [8], see Figure 1(i)

Visualization and Segmentation:

- Voxel visualization [9], see Figure 1(j)

Segmentation:

- Chan-Vese [10], see Figure 1(k)

An increasing amount of students and collaborators are contributing with their own code and are helping greatly in maintaining and improving OpenTissue.

3 Coding Style

OpenTissue is written in C++ and strongly relies on templates using the coding standards suggested by Alexandescu *et al.* [11–14]. Templates gives the flexibility to generate object classes that are type independent, or more precisely the types of the in and out data for classes are first decided by the compiler at compile time. The gained generic structure of classes strongly extends the usability of each class and greatly reduces the amount of code that needs to be maintained. The down-side is that template programming is still not very well supported by compilers, and syntax error messages can at times be puzzling. Metaprogramming literally means a language in a language, which for C++ is completely realized as a programming style, and does not rely on added libraries or special compilers. The advantage of metaprogramming is, that many calculations that normally would be performed at runtime, are evaluated at compiler time, hence reducing runtime and runtime errors. Unfortunately, like templates, metaprogramming is not well supported by typical compilers, and compiler errors take longer time to correct.

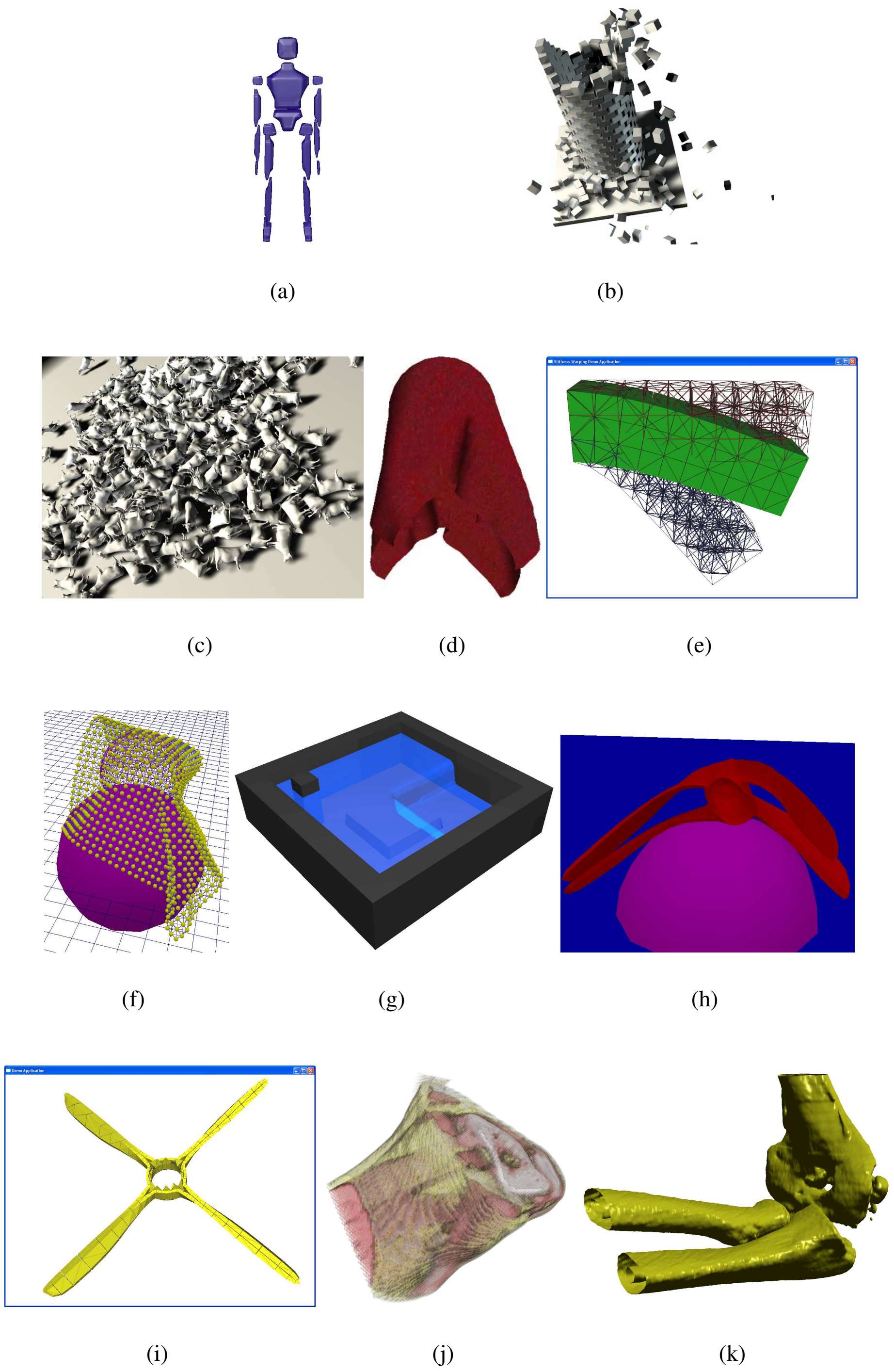

Fig. 1. Examples of graphical output from OpenTissue demos.

4 Pros and Cons of OpenTissue

OpenTissue is an Open Source Toolkit driven by academia research, implementing the latest, cutting edge technology, and accompanied with scientific publications explaining methods in full depth, e.g. [1, 7, 8].

There exists several systems for geometry, simulation, and visualization, such as Matlab (`www.mathworks.com`), VTK (`public.kitware.com/VTK/`), and Open Dynamics Engine (`ode.org/`), OpenFEM (`www-rocq.inria.fr/OpenFEM/`), etc., but none of these are both Open Source and integrates all aspects of analysis, simulation, and visualization. On the other hand, OpenTissue includes many modules such as collision detection, physics based simulation, medical imaging, and scientific visualization, and they are easily integrable. A major advantage of the toolkit is that it provides alternative algorithms for many problems such that a user is encouraged to experiment with the algorithms to find the one best suited for a given problem. It is further strongly inspired by our collaborators including university and industry, and has been used as one of our teaching tools for several years.

Being a dynamically evolving toolkit, using OpenTissue takes some effort on behalf of the user. First, the heavy use of template programming and the large diversity in algorithms and data structures combined with the slightly delayed documentation causes a steep learning curve. Second, the toolkit has third party dependencies the main of which are: Boost (`www.boost.org`), Atlas (`math-atlas.sourceforge.net`), Qhull (`www.thesa.com/software/qhull/`), TinyXML (`www.grinninglizard.com/tinyxml/`), and Nvidia SDK (`developer.nvidia.com`),. These cannot be entirely eliminated, however, they are included as binaries in order to ease usage and reduce the number of compiler problems. Not all of OpenTissue needs all third party software, and the users need only concern him or herself with the third party libraries actually needed for the parts of OpenTissue actually used. Third, there is no clickable drag and drop GUI. It is all highly optimized C++ code, and the library is a low-level application programmers interface (API), which means that OpenTissue makes all the data structures and algorithms available, but the user must tie everything together. This requires an experienced C++ programmer that is not afraid of template programming. Finally, our aim is to complete the transition of all the code into a completely generic programming framework; we are currently halfway.

Open Source implies that no implementation details are hidden, and the user is encouraged to tweak and change the code to fit their specific needs, without having to wait on OpenTissue developers. The toolkit is designed using modern programming paradigms of generic and metaprogramming, which in practice implies that most of the code is implemented in headers, thus minimizing linker problems. Another advantage is that the user only needs to include the parts of OpenTissue in his or hers binaries that are actually used. Documentation is based on Doxygen. Although this is often lacking behind on the more recent additions, luckily there are plenty inline source comments fully explaining implementation details.

References

1. Erleben, K., Sporring, J., Henriksen, K., Dohlmann, H.: Physics-based Animation. Charles River Media (2005)
2. Kelager, M., Fleron, A., Erleben, K.: Area and volume restoration in elastically deformable solids. Electronic Letters on Compuer Vision and Image Analysis (ELCVIA) **5** (2005) 32–43 Special Issue on Articulated Motion, `http://elcvia.cvc.uab.es`.
3. Terzopoulos, D., Platt, J., Barr, A., Fleischer, K.: Elastic deformable models. In: Computer Graphics. Volume 21. (1987) 205–214
4. Müller, M., Gross, M.: Interactive virtual materials. In Balakrishnan, R., Heidrich, W., eds.: Proceedings of Graphics Interface 2004, Waterloo, Canada, Canadian Human-Computer Communications Society (2004) 239–246
5. Jakobsen, T.: Advanced character physics. Presentation, Game Developer's Conference (2001) `www.ioi.dk/Homepages/thomasj/publications/AdvancedPhysics_files/v3_document.htm`.
6. Layton, A.T., Panne, M.v.d.: A numerically efficient and stable algorithm for animating water waves. The Visual Computer **18** (2002) 41–53
7. Erleben, K.: Stable, Robust, and Versatile Multibody Dynamics Animation. PhD thesis, Department of Computer Science, University of Copenhagen (DIKU), Universitetsparken 1, DK-2100 Copenhagen, Denmark (2005) `http://www.diku.dk/~kenny/thesis.pdf`.
8. Erleben, K., Dohlmann, H., Sporring, J.: The adaptive thin shell tetrahedral mesh. Journal of WSCG (2005) 17–24
9. Engel, K., Hadwiger, M., Kniss, J.M., Rezk-Salama, C.: High-quality volume graphics on consumer pc hardware. SIGGRAPH course notes (2002)
10. Chan, T., Vese, L.: Active contours without edges. IP **10** (2001) 266–277
11. Alexandescu, A.: Moder C++ Design: Generic Programming and Design Patterns Applied. C++ In-Depth Series – Bjarne Stroustrup. Addison Wesley (2005)
12. Abrahams, D., Gurtovoy, A.: C++ Template Metraprogramming. C++ In-Depth Series – Bjarne Stroustrup. Addison Wesley (2005)
13. Sutter, H., Alexandescu, A.: C++ Coding Standards. C++ In-Depth Series – Bjarne Stroustrup. Addison Wesley (2005)
14. Sutter, H.: Exceptional C++ Style. C++ In-Depth Series – Bjarne Stroustrup. Addison Wesley (2005)