7th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
Suceava, Romania, May 27 – 29, 2004

VIRTUAL REALITY APPLICATIONS AND FRAME-RATE CONTROL

Laurent GRISONI, Jérémie DEQUIDT, Christophe CHAILLOU
INRIA (ALCOVE), CNRS (LIFL/IRCICA), University of Lille 1, UPRESA CNRS 8022, Bâtiment M3, 59655 Villeneuve d'Ascq Cedex, France.
Email : [laurent.grisoni|jeremie.dequidt|christophe.chaillou]@lifl.fr

Abstract. *This paper addresses the problem to guaranty frame-rate of a Virtual Reality application, without specific knowledge of the host computer power. General issues are discussed, especially regarding real-time physical simulation. Heuristic techniques are presented based on an example that dynamically adapts a model resolution to some pre-defined frame-rate.*
Keywords*: virtual collaborative environment, real-time, level of detail, animation*

Introduction

Most practical virtual reality applications share the same property, most of the time so intuitive that it is simply ignored within the development process. This common property is the fact that, in spite of all the possible optimizations and sophistications it can provide, it is not easily possible to predict, for a given complex application, the frame-rate it will provide without precise knowledge of the host hardware architecture: when such knowledge is available, devoted tests and measures are usually performed, and application is manually tuned in order to provide the desired performance. Providing applications that can guaranty some pre-set frame-rate value without any specific knowledge of the computer host is a difficult question. This is of course useful in videogames industry, but also in more specific applications, such as surgical pedagogic simulators for example, or flight simulators. Actually most of earlier works on the problem have been produced in the latter context [5,6].

This article addresses the question to know how to handle such a problem, and provide, as flexibly as possible, tools for guaranteed frame-rate manipulation. This problem actually involves several key points: in order to provide virtual reality application that can adapt their complexity to hardware context, it is of course mandatory to manipulate representations (either geometrical, numerical, or more generally, from computational point of view) that provide parameters controlling the model *resolution*: strong background on such models is hence needed in order to provide efficient frame-rate control.

This article is organized as follows: next section presents related general aspects and results, especially regarding level-of-details (LOD for short) techniques, as well as dynamic tuning theoretical tools that are potentially involved in the process. Section 3 details the classical manner to manipulate and control level-of-detail models. Section 4 proposes a framework that extends the approach discussed in 3 to animation, especially regarding physical based animation. Section 5 presents a practical example we introduced within a complex surgical simulator that dynamically adapts the rendering of human intestine model to requested frame-rate.

Related aspects

As mentioned above, the problem addressed in this article involves strong control on the models manipulated, i.e. at least a set of control parameters for each object, and algorithm: so far, most of results that can be linked to frame-rate adaptation propose control on the model itself, rather than on algorithms.

Geometrical model control involve, in its simplest version, level of detail manipulation

[12]. More sophisticated aspects can be found in the so-called *multi-resolution* objects [7,8,9], or mesh dynamic simplification [11], possibly depending on point of view [10]. We refer the reader to [1] for an excellent overview of such model manipulation. From a more theoretical point of view, many geometric models take advantage from wavelet theory [13,14,15]: such tool provides numerical framework for automatic simplification and refinement. Wavelets suffer from quite involving mathematical prerequisites, but provide stable tools, that come along with rigorous error evaluation when used for model simplification.

As a result, many theoretical and practical results exist about geometrical control of representation complexity. Next section discusses the classical use that can be done around such geometrical models.

Geometric LOD manipulation

Most existing works aiming at controlling frame-rate work the following way: considering current frame-rate measure, modifications are applied on parameters defining geometry complexity, in order either to make the drawing faster, or on the other hand more accurate. This straightforward principle is often made theoretical using the notion of *cost function*[2,3,4]. Such a function, most of the time heuristically determined, measures the "cost" displaying a given model: this cost functions evaluates the computation complexity, hardware rendering time, or virtually any "cost" that can be involved in the process. Depending on the possible evolutions of this function, modifications are accordingly made on models in order to provide optimal representation, respecting the desired frame-rate. It is to note that, classically, models are classified, and associated different priorities, depending on the viewing distance they are used. This is the most classical criteria that is used. Within animation context, there might be some others: this will be detailed in the next section, that describes the use of LOD for animation, as well as some linked aspects, including software architecture requirements.

LOD for animation

Only geometrical aspects have been treated so far in this article. Yet, animation also benefits from multi-resolution results. [19, 20] presents a simple framework for animation level-of-detail: in this work, simple tests are used for automatic representation selection. [16] presents a theoretical framework for multi-resolution physical simulation, practically used in [17]. [18] presents a practical example of multi-resolution animation used on human organ physical simulation. It is to note that such models involve sophisticated simulation algorithms. Apart from the appealing results such a work presents, it has some immediate consequences on a software framework that would combine such a model to other simulated objects. In order to be understood, this points needs to go deeper in detail into the classical physical animation process (we refer the reader to [21] for a good introduction on that point), but can yet be explained on some points: classical physical animation involve some infinite loop, where forces are calculated (depending on geometric collisions and user interaction) and partial differential equation resulting from these forces, integrated. It is known that using implicit integration schemes provides better numerical stability [21], and allows for larger integration timesteps. Such flexibility has already been used within some multiresolution physical animation [18]: yet, it is simple to understand that involving such models in a simulation involving other, simple object, would constraint the whole to be time-step adaptive as well: several objects, each willing to command the time-step, would make the whole process very tricky.

In order to counter-balance such a problem, it is quite attractive, from algorithmic point-of-view, as well as from intellectual point-of-view, to consider each object as being alone in its simulation process. Each object would be considered as an autonomous entity that interacts with other objects, and adapts its

behavior to the constraints it faces (computation time, rendering time, speed, distance of viewing, etc…). Multi-agent based architectures hence appear quite appealing to such a context: yet, this goes at first against the well-accepted principle that physical simulation needs global synchronization, that is, simulation time is discretized the same manner for all objects. We recently proposed a multi-agent framework for simulation [23] that demonstrates it is actually possible to mix multi-agent approach with interactive physical simulation. Figure 1 shows an example of animation where each object is considered as an autonomous agent, interacting with other objects.

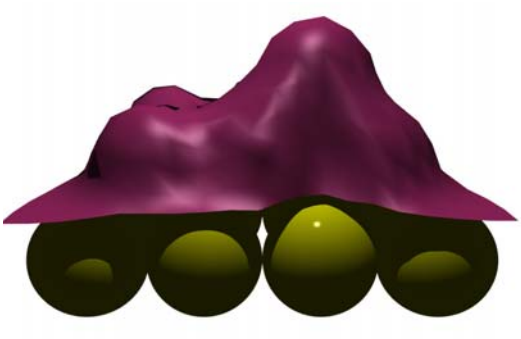

Figure 1. Example of real-time physical simulation using multi-agent framework: each object is considered as an autonomous entity.

Such an approach opens the way to true frame-rate control for animation, and makes it possible to achieve simple tasks, such as for example partial inactivation when some object simulation sets the object at equilibrium state, and no interaction is performed on the object. By comparison, such a feature is so far very hard to achieve in classical, global frameworks.

Example and results

In order to illustrate the notion presented above, we achieved simple test in some existing platform. This platform aimed at providing surgeons with pedagogic tool for minimally invasive surgery, allowing them to interactively manipulate human intestine system representation [22]. Such a model has been associated, for its rendering part, to a frame-rate control system, that is described here. We will not detail the model used for representation: we simply mention what is strictly necessary to understand the influence of the parameters on the model, and we use the functionality as a "black-box" that depends on two parameters. This rendering uses two parameters, respectively ε (floating point value) and n (positive integer value), that determine the complexity of the rendering. Their use is described below for better understanding of their consequence on the rendering.

The first parameter ε determines how many parameter intervals will be used in the tessellation of the skeleton curve representing the intestine (see Figure 2 for visual example). The second parameter n defines the complexity of the primitive cylinder used for tessellation (we used hardware-based skinning for better result, see [22] for details). It determines the quality of the overall shape, i.e., the coarser the primitive, the coarser the resulting deformation. In our implementation, we actually use several versions of the tesselation primitive. The number of polygons in each version is $O(2^i)$, for i=0,…,n. Using skinning, raising the number of vertices on the tessellation primitive produces smoother interpolation with only a small measured computational overhead.

The principle of the frame-rate control algorithm is fairly simple. In fact, it can be seen as a feedback control loop: considering the distance between the measured frame rate and the required value, tuning functions are applied to ε and n. Fig. 2 illustrates this. Precisely speaking, the iterations from one display configuration (ε_k, n_k) to the next one (ε_{k+1}, n_{k+1}) uses the following relations:

$$\begin{cases} \varepsilon_{k+1} = \varepsilon_k + \alpha(f_0 - f_k) \\ \rho_{k+1} = \rho_k + \beta(\varepsilon_{k+1} - \varepsilon_k) \\ n_{k+1} = \lfloor \rho_{k+1} \rfloor \end{cases}$$

where ρ_k is a floating point version of the integer variable n_k, α and β are two arbitrary constants, f_k is the current measured frame rate, and f_0 the

desired one. In our implementation, we used $\alpha=10^{-4}$ and $\beta=5$. These values were experimentally determined, and practically appeared to provide satisfactory results: indeed, using these values, the tessellation process stabilizes itself at the desired frame rate in less than a second.

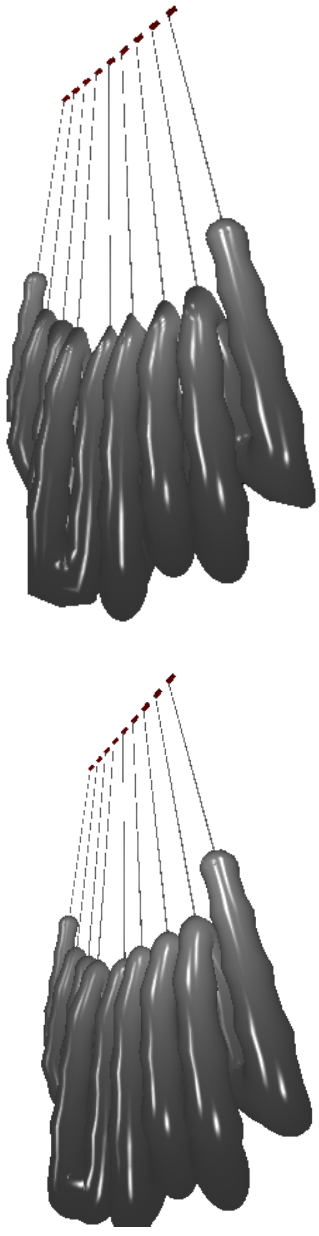

Figure 2: Low quality tessellation (top) and high quality tessellation (bottom) obtained using our automatic frame rate adaptation system at 900 and 460 frames/sec respectively, on a GeForce 4 based system.

Conclusion

This article presented some heuristic approach to guarantying frame-rate. The presented technique allows for fast adaptation to required frame-rate, and can easily be generalized to other specific 3D animated objects. A lot yet remains to be done before such frame-rate control could be flexibly adapted within any virtual reality applications: we strongly believe in the potential of agent-based architecture for the wide-spreading of such tools. So far, only very few has been done on the question to provide generic tools for cost function generation: this point would also be interesting from a research point of view.

References

[1] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, R. Huebner (2002), *Level of Detail for 3D Graphics*, Morgan Kaufmann ed., ISBN 155-860-8389.

[2] D.G. Aliaga, A. Lastra (1999) *Automatic image placement to provide a guaranteed frame rate*, Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH), 307—316.

[3] T. A. Funkhouser, C. H. Séquin (1993) *Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments*, Computer Graphics(27), Annual Conference Series, pp. 247-254.

[4] Maciel, P. W. C., and Shirley, P. (1995) *Visual Navigation of Large Environments Using Textured Clusters*. In Proceedings Symposium on Interactive 3D Graphics, pp. 95—102.

[5] Bruce Schachter (ed.) (1983), *Computer Image Generation*, John Wiley and Sons.

[6] Mueller, Carl (1995) "*Architectures of Image Generators for Flight Simulators*", University of North Carolina Computer Science Technical Report TR95-015.

[7] A. Finkelstein, D. Salesin (1994), *Multiresolution curves*, Proc. of the 21st annual conference on Computer graphics and interactive techniques, pp. 261-268.

[8] D. Zorin, P. Schröder, T. Derose, L. Kobbelt, A. Levin, W. Sweldens (2002), *Subdivision for*

Modeling and Animation, course note in the 29th annual conference on Computer graphics and interactive techniques, july.
[9] I. Guskov, K. Vidimce, W. Sweldens, P. Schröder (2000), *Normal meshes*, Computer Graphics Proceedings (SIGGRAPH 2000), pp 95-102.
[10] D. Luebke (2000), *Advanced Issues in Level of Detail: Robust View-Dependant Simplification for Very Large Scale CAD Visualization*, SIGGRAPH course Notes.
[11] P. Cignoni, C. Montani, R. Scopigno (1998) *A Comparison of Mesh Simplification Algorithms*, Computers & Graphics, Pergamon Press, Vol. 22(1), pp. 37-54.
[12] J.H. Clark (1976) *Hierarchical geometric models for visible surface algorithm,* Communications of the ACM, 19(10), pp.547-554, Oct.
[13] W. Sweldens, P. Schröder (1996), *Building your own wavelets at home,* ACM SIGGRAPH Course Notes.
[14] L. Grisoni, C. Blanc, C. Schlick (1999) *Hermitian B-splines*, Computer Graphics Forum, 18(4), pp. 237-248, december.
[15] P. Schröder, W. Sweldens (1995) *Spherical wavelets: Efficiently representing functions on a sphere*, Computer Graphics Proceedings (SIGGRAPH 95), pp. 161-172, 1995
[16] E. Grinspun, P. Krysl, P. Schröder, (2002) *CHARMS: a simple framework for adaptive simulation*, ACM TOG, Proc. of the 29th annual conference on Computer graphics and interactive techniques, 21(3), pp. 281-290, july.
[17] S. Capell, S. Green, B. Curless, T. Duchamp, Z. Popovic (2002) *A multiresolution framework for dynamic deformations*. In Proc. of ACM SIGGRAPH/Eurographics Symp. on Computer animation, ACM Press, pp. 41–47.
[18] G. Debunne, M. Desbrun, M.-P. Cani, A.H. Barr (2001), *Dynamic real-time deformations using space and time adaptive sampling*, Computer Graphics Proceedings (Aug), Annual Conference Series, ACM Press / ACM SIGGRAPH. Proc. SIGGRAPH'01.
[19] D.A. Carlson, J.K. Hodgins (1997, *Simulation Levels of Detail for Real-time Animation*, Proc. Graphics Interface '97, pp 1-8.
[20] D. C. Brogan, J. K. Hodgins (2002) *Simulation level of detail for multiagent control*, Proc. of the first international joint conference on Autonomous agents and multiagent systems, pp. 199-206.
[21] A. Witkin, D. Baraff, M. Kass (1997) *An Introduction to Physically Based Modeling*, ACM SIGGRAPH Course Notes.
[22] L. Raghupathi, L. Grisoni, F. Faure, D. Marchal, M.-P. Cani, C. Chaillou (2004) *An Intestine Surgery Simulator: Real-Time Collision Processing and Visualization*, accepted for publication in IEEE Trans. On Visualization And Comp. Graphics.
[23] J. Dequidt, L. Grisoni, C. Chaillou (2004), *Asynchronous interactive physical simulation*, submitted for publication.