Available online at www.sciencedirect.com
ScienceDirect
Computers & Graphics 32 (2008) 307–319
COMPUTERS & GRAPHICS
www.elsevier.com/locate/cag

Technical Section

Optimizing the management of continuous level of detail models on GPU

O. Ripollés*, M. Chover

Dept. Lenguajes y Sistemas Informáticos, Universitat Jaume I, Castellón, Spain

Received 27 July 2007; received in revised form 14 December 2007; accepted 12 February 2008

Abstract

In this paper we present a new continuous multiresolution framework which has been developed in view of the outstanding evolution of hardware. Our interest not only focuses on exploiting GPUs possibilities, but also on making the best possible use of the capabilities offered by new bus technologies. On the one hand, we store the geometry, based on triangle strips, in high-performance memory on the GPU, offering fast rendering time. On the other hand, we have designed our level-of-detail extraction algorithm in order to make the most of current PCI Express bus characteristics, by sending the minimum information in the most appropriate way while taking into account the appearance of degenerate triangles. The results section shows that our model improves the efficiency of previously existing solutions. Its easy integration and its short extraction time make it suitable for game engines and graphics libraries which often resort to discrete models when it comes to selecting a multiresolution technique.

Keywords: Multiresolution modeling; Level of detail; Graphics hardware

1. Introduction

In recent years, the tendency has been to include geometrically complex scenes in interactive graphics applications, such as computer games or virtual reality environments. These highly realistic scenarios often involve many polygonal meshes made up of a high number of triangles, which poses a problem for maintaining a high frame rate. One of the possible solutions to this problem is the use of level-of-detail techniques, which represent an object through a set of approximations at different levels of detail to allow the recovery of any of them on demand [1]. Nowadays, this can be considered as a compulsory feature. In this sense, graphics libraries like OpenInventor or OpenSceneGraph, and game engines such as Torque or Ogre, introduce multiresolution models to easily reduce the amount of geometry that must be rendered in a scene, thus resulting in an improvement in performance.

The first multiresolution models that were developed were based on a relatively small number of approximations (usually between 5 and 10) [2], and were known as discrete multiresolution models. These discrete models suffer from popping artifacts that appear when switching between the different levels of detail, causing noticeable and visually disturbing effects. Later, continuous multiresolution models appeared with the aim of improving discrete models, as they offered a wide range of different approximations to represent the original object.

Despite the better features shown by continuous models, traditional solutions usually involve discrete multiresolution models. The reasons behind this decision are quite simple: discrete models are more easily integrated and they also offer an easier and more straightforward level of detail update. Many authors consider that using continuous models is not worth the effort, as in an interactive application the viewer keeps moving all the time and this would entail updating the whole scene continuously, which would lower overall performance. In this sense, it is easier to discard one model and use another one (which happens with discrete models) and accept the popping artifacts.

*Corresponding author. Tel.: +34 964 728337; fax: +34 964 728435.
E-mail addresses: oscar.ripolles@uji.es (O. Ripollés), chover@uji.es (M. Chover).

0097-8493/$ - see front matter
doi:10.1016/j.cag.2008.02.003

A further improvement on continuous multiresolution models are those that present view-dependent capabilities, which enable an object to include different resolutions in different areas at the same time. These models, although they offer better granularity, present important time limitations as their extraction process is usually more complex and they need to obtain some extra information of the scene conditions. These time-consuming characteristics make them less interesting for graphics libraries and game engines.

As we all know, graphics hardware has improved outstandingly over recent years. Performance is doubling every six months [3], in contrast to microprocessors which grow by approximately 40% every year [4]. Thus, the possibility of taking maximum benefit from its computational power, its tremendous memory bandwidth, the possibility of parallel programming, as well as the alleviation of CPU load, has increased the interest in using GPUs as a computational solution, not only for computer graphics, but also for general-purpose routines. Furthermore, the development of the PCI Express bus has boosted the performance of AGP buses, making the traffic of information between the CPU and the GPU much more efficient.

Since hardware was improving and new technologies were presented, we decided to develop a new continuous multiresolution framework to take advantage of these new features. Our intention is to provide a competitive solution in comparison with discrete models.

The solution presented in this paper has been devised from the experience obtained after analyzing different multiresolution approaches, and in a more precise way, after the implementation of LodStrips [5,6]. This continuous model was entirely based on optimized hardware primitives, triangle strips, and dealt with the creation of degenerate triangles applying pre-calculated filters. Nevertheless, this multiresolution model presented important limitations. The main drawback was the extraction process, which entailed updating the strips by performing costly random insertions and resizes.

Thus, our main objective was to develop a level-of-detail update routine which maximized the efficiency of the extraction process and the data traffic through the bus. The approach presented in this paper, Speed Strips, exploits the new hardware capabilities in two ways. On the one hand, the possibility of storing geometry information in the graphics hardware by means of vertex buffer objects vastly improves the visualization time. On the other hand, the PCI Express bus supports isochronous data transfers and different QoS levels, which guarantees that the data arrive at their destination in a given time. The use of multiple isochronous virtual channels per lane presents a perfect solution for applications which require real-time data transfer [7]. With these two features in mind, we have developed a triangle strips updating routine which changes the strips in one single step by working directly with the information stored in the GPU and sending the minimum amount of information in the most appropriate way.

Speed Strips presents the following features:

- Easy implementation, as it offers integration into an application with little effort.
- Low memory cost, if compared with discrete models which involve having to store different approximations of the same model.
- Short extraction time, as it is able to compete with the static mesh substitution approach.
- Based on triangle strips, which not only offer a compact representation of the connectivity existing in a triangle mesh but also a faster rendering.

Fig. 1 presents a scene of the Ogre rendering engine where our multiresolution model has been integrated. More than 15 million polygons are rendered in real time. To adjust the detail, we have considered the distance to the viewer criterion. We have also color-coded the different models to represent the level of detail at which they are rendered.

This paper presents the following structure. Section 2 contains a study of the work previously carried out on multiresolution modeling. Section 3 presents the general framework in which this model has been developed. Section 4 provides thorough details of the proposed method. Section 5 offers different versions of the original method which have been developed for testing purposes. Section 6 includes a comparative study of our algorithm against other possible solutions. Lastly, Section 7 contains comments on the results obtained and a brief outline of future lines of work.

Fig. 1. Color-coded LOD scene with many Speed Strips models inside the Ogre rendering engine.