5
Conclusions

This chapter presents the contributions and future work needed for the techniques proposed in this thesis.

5.1 Main Contributions

The literature on parallel game engines is scarce. Furthermore, most of the work on parallel rendering is focused on PC clusters and/or on global aspects of parallel rendering, rather than on multicore systems as required by the game industry. This thesis presents some contributions for the area of parallel game engines with multiprocessors.

Firstly, the thesis presents a comprehensive analysis of parallel game engine architectures and gives an orientation towards the most adequate architecture. The contribution here is to provide an orientation that we cannot find in the literature. The Fully Parallel Architecture, although simple, is very scalable and we could not find any source material presenting a similar architecture. In order to create a game engine that is truly scalable, it is necessary to make use of parallel versions of standard algorithms. As shown in the introduction, games today simply divide the workload by separating single threaded tasks amongst the different processing cores. This remains effective only for the current computers that have a small amount of cores. In a near future, when computers will have a great number of cores, this old method will face great difficulties, because there will be a great disparity between the great number of cores and a few tasks to be distributed. The Fully Parallel Architecture is presented as a scalable and flexible architecture for videogames.

Secondly, this thesis presents parallel implementations of classic techniques in game development. It proposes a new and effective method for parallel octree culling and sorting for multicore systems, using counting sort and based on a new balancing algorithm called Adaptive Two-Step Static Balancing (A2SSB). The complexity of A2SSB is constant, and if we have different octrees in the same computer architecture (*i.e.* the number of cores is fixed), we have a cost that is proportional to 8^h (equation 1), where h is the

Algorithm	Time (milliseconds)
Full A2SSB algorithm	4.0
A2SSB using RAM	7.1
Classic Depth-first Strategy (single processor)	15.5

Figure 5.1: Results for the octree node processing with 299,593 nodes and 4 processors.

height of the full octree.

$$C \propto 8^h \tag{1}$$

The results (Figure 5.1) show the importance of avoiding the access to RAM by using a cache friendly strategy, an issue not being discussed in the literature on game engines. The comparison between the proposed algorithm with its version that uses the RAM reveals that the performance is almost 1.8 times better. The improvement in relation to the classic single processor is almost 4.0 times for a computer with 4 cores.

As a third main contribution, the present thesis proposes a new parallel algorithm for broad-phase collision detection for multicore systems, which is based on hierarchical grids. Also a narrow-phase load balancing is proposed. As far as the author is aware, there is no work on hierarchical grid on multicore systems in the literature.

5.2 Future Works

This thesis needs comparison tests to better analyze how the proposed methods and techniques stand against alternative methods (such as Binary Space Partition Trees with Potencially Visible Sets for culling [Teller92] and Sweep and Prune for broad phase collision detection [Cohen95]). Since source materials for such alternative parallel techniques are not available, alternative methods need to be implemented.

Also a whole game engine based on the proposed methods and algorithms should be done as part of the future work. This thesis focused on parallel culling, sorting, and collision detection, but there's still others engine parts that need research such as "Artificial Intelligence" and "Streaming and Decompression". For example, we could research a parallel version of the A* search algorithm allowing us improve path finding performance on parallel CPUs.

As a challenging proposal for future work, the parallel algorithm for the

broad-phase collision detection should be extended to other types of cells and deformable objects.