Car Racing through the Streets of the Web: a High-Speed 3D Game over a Fast Synchronization Service

Stefano Cacciaguerra, Stefano Ferretti, Marco Roccetti, Matteo Roffilli

Department of Computer Science, University of Bologna,
Mura Anteo Zamboni 7, 40127 Bologna Italy
{scacciag, sferrett, roccetti, roffilli}@cs.unibo.it

ABSTRACT

The growth of the Internet brought a new age for game developers. New exciting, highly interactive Massively Multiplayer Online Games (MMOGs) may be now deployed on the Web, thanks to new scalable distributed solutions and amazing 3D graphics systems plugged directly into standard browsers. Along this line, taking advantage of a mirrored game server architecture, we developed a 3D car racing multiplayer game for use over the Web, freely inspired to Armagetron. Game servers are kept synchronized through the use of a fast synchronization scheme which is able to drop obsolete game events to uphold the playability degree while preserving the game state consistency. Preliminary results confirm that smart 3D spaces may be created over the Web where the magic of gaming is reproduced for the pleasure of a huge number of players. This result may be obtained only by converging highly accurate event synchronization technologies with 3D scene graph based rendering software.

Categories and Subject Descriptors

K.8.0 [Computing Milieux]: Personal Computing – Games.

General Terms: Algorithms, Design, Performance.

Keywords: MMOG, Synchronization, Scene Graph.

1. INTRODUCTION

The global connectivity provided by the Internet and the enormous advances of game technologies promote new exciting gaming scenarios, where large numbers of users participate together in highly interactive Massively Multiplayer Online Games (MMOGs). Keys to success of MMOGs are as follows: i) game interfaces have to respond to the issue of emulating a sensation of a realistic participation; ii) games must be deployed over a scalable architecture able to timely provide a compelling experience shared amongst many simultaneous players. With respect to i), a new genre of networked gaming applications is emerging that belongs to the realm of fiction. In this context, it is under the responsibility of game designers to develop systems that make available to players the look and feel of exciting gaming sensations. Game interfaces are thus needed that provide full sensorial experiences involving all human senses like vision and hearing [1]. To this aim, today's graphics processor units offer means to create accurate virtual worlds thanks to 3D visual representations and positional audio. Modern graphical interfaces permit to integrate 3D multiplatform libraries, such as OpenSceneGraph for example, directly on a standard browser [2].

Copyright is held by the author/owner(s).
WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

According to this approach, a scene is described as a set of hierarchical data structures (i.e., trees) that are used to manage efficiently the rendering activities of 3D objects. Using this 3D visualization technology for developing game interfaces, it turns out that each game event generated by a given player produces a corresponding modification of the rendered scene graph at the player display. As to the problem of distributing MMOGs over the Web, an efficient architectural solution has been recently proposed which rests upon the use of a *mirrored game server* architecture [3]. In essence, such solution deploys over the network a constellation of communicating replicated game servers, locally maintaining a redundant copy of the game state. Each game server receives game events generated by its connected players and periodically notifies them with new game states that correspond to modifications of the scene graphs that have to be rendered at the player side. Obviously, game servers synchronize themselves in order to maintain a uniform view of the game evolution within the entire system. These synchronization activities must be accomplished as fast as possible to guarantee an adequate interactivity degree among players [4]. In this paper, we present a new 3D car racing MMOG, deployed over the Web, which is able to speed up the rate of the exchanged scene updates at the player side based on the required interactivity degree. This result is obtained by resorting to a fast event synchronization scheme that exploits the notion of obsolescence to drop out-of-date events while preserving game consistency at each mirrored game server. As network QoS is not yet a tangible product to ameliorate gaming problems due to network service disruptions, our work shows that only the convergence of fast synchronization technologies with modern 3D rendering software can be a viable solution to improve the entertaining experience for the Web gaming community.

2. DESIGN ISSUES

A fundamental requirement of MMOGs is to provide a shared experience that appears consistent across all the players. A consequent problem is concerned with the latency needed to notify all participants with new scene graphs representing modifications of the game state. In particular, fast-paced games need high values of the *Scene Update Rate* (SUR) for a full player satisfaction. Unfortunately, the typical network behavior can be detrimental to meeting even a minimal *Scene Update Rate* (mSUR) value for an acceptable playability degree. Based on the scientific literature [5], mSUR is typically set equal to 8 scene updates per second (sps), roughly equivalent to an interactivity degree of 125 ms. It is worth noticing that such SUR value is independent of the rate of frames rendered by the graphics processor unit at the client side. Indeed, this latter parameter is concerned with the graphical rendering of the actual scene. SUR, instead, is concerned with the pace of delivering scene updates (i.e., how fast game state updates are notified to all players).

Recent studies demonstrated that the semantics of the game may be exploited to speed up the synchronization activities among game servers, thus improving the responsiveness of the system while maintaining the game state consistency. In particular, a notion of *obsolescence* has been introduced that permits to discard those game events that during the game evolution lose their importance [4]. In substance, the idea of obsolescence rests upon the fact that fresher events (and corresponding scene updates) can make irrelevant the previous ones. With this in view, as soon as a game event (and its corresponding scene update) becomes obsolete, due to the generation of a newer one, then it can be discarded for an augmented interactivity. In the following, we will show how this concept may be applied in the development of a highly interactive 3D car racing game for use over the Web.

Table. 1: Game Events.

Game Event	Description	Key
Accelerate	Each key press increases the speed of the car	↑
Brake	Each key press decreases the speed of the car	↓
Turn Left	The more the key is pressed the higher the left turn	←
Turn Right	The more the key is pressed the higher the right turn	→

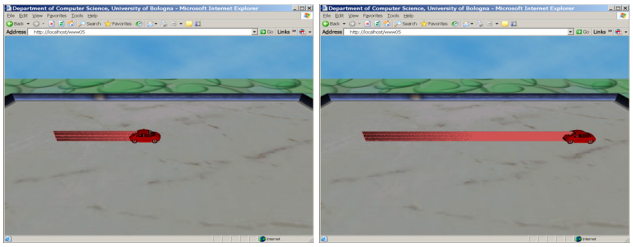

Figure 1. *Accelerate*: the rightmost event makes obsolete the leftmost one.

Figure 2. *Turn*: the rightmost event does not make obsolete the leftmost one.

3. 3D CAR RACING

We developed a 3D car racing game, freely inspired to Armagetron [6]. According to the game, each player drives a car within a synthetic arena. As long as each car moves within the arena, a wall is built up after its passage. The goal of the game is to make the opponents crash into a wall. The car can turn left or right in a range of 0-179 degrees. The user can accelerate/decelerate the speed of its car while going in the same direction. The speed of the car is bound between two specific values: *MIN_SPEED* and *MAX_SPEED*. If the user turns the car, its speed is decreased, following a linear law based on the amplitude of the turn. Game events that a player may generate are reported in Table 1. Each time the user presses a key, the corresponding game event is generated and sent to the server that, in turn, notifies other servers and updates its game state accordingly. A game event e is encoded through the tuple (X, ρ, θ), where X is the actual position of the car in the virtual arena, while ρ and θ represent respectively the radial and angular coordinates of the new speed vector. Needless to say, each generated game event corresponds to a scene graph modification that must be delivered to all players. In this scenario, game events may become obsolete based on a straight-line acceleration/deceleration of the car. In simple words, an event $e_1 = (X_1, \rho_1, \theta_1)$ may be made obsolete by a subsequent event $e_2 = (X_2, \rho_2, \theta_2)$, generated by the same player, if $\theta_1 = \theta_2$ (Fig. 1). Indeed, in this situation dropping e_1 does not alter the final computed game state, as the two events simply amount to drawing walls along the same direction. Instead, if two events e_1 and e_2 are generated such that $\theta_1 \neq \theta_2$ (Fig. 2), then the previous event e_1 cannot be considered as obsolete and must be sent to all the mirrored servers for final delivery to all the players. Further, according to the game rules, two walls cannot cross each other. Hence, knowing the exact instant at which the wall is built up by a car (say A) may become important to determine if a further car (say B) crashes into the wall or cuts in front of A. Indeed, in the former case A wins, in the latter A loses. Summing up, obsolescence cannot be applied when it becomes necessary to settle collision detection problems.

4. RESULTS AND CONCLUSIONS

We conducted a preliminary study based on the use of the 3D car racing game we developed. We implemented different gaming scenarios where the number of communicating mirrored game servers deployed over the Web varies from 4 to 7. Figure 3 reports the performances of the 3D car racing MMOG when our obsolescence-based mechanism is either activated or not (ON or OFF). Our mechanism promotes the fluent rendering of the game evolution as it guarantees that, in average, experienced SUR values are always over mSUR (Fig. 3). We claim that "the need for speed" is a mandatory requirement to guarantee reckless racing cyber-contests on the Web. The paper has showed that an opponent rapidly running away (through the streets of the Web) may be caught provided that our approach is adopted.

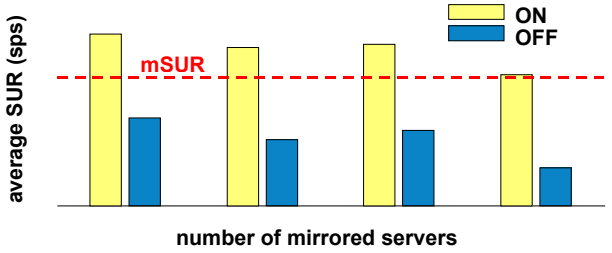

Figure 3. Average SUR values.

5. REFERENCES

[1] Cacciaguerra S., Roccetti M., Roffilli M., Lomi A., "A Wireless Software Architecture for Fast 3D Rendering of Agent-Based Multimedia Simulations on Portable Devices", *Proc. IEEE Consumer Communications and Networking Conf.*, 2004.

[2] OpenSceneGraph Web Site: www.openscenegraph.org/, 2005.

[3] Mauve M., Vogel J., Hilt V., Effelsberg W., "Local-lag and timewarp: Providing consistency for replicated continuous applications", *IEEE Trans. on Multimedia*, 6(1):47–57, 2004.

[4] Palazzi C.E., Ferretti S., Cacciaguerra S., Roccetti M., "On Maintaining Interactivity in Event Delivery Synchronization for Mirrored Game Architectures", *Proc. IEEE Int. Conf. on Networking Issues in Multimedia Entertainment*, 2004.

[5] Pantel L., Wolf L.C., "On the Impact of Delay on Real-Time Multiplayer Games", *Proc. 12th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2002.

[6] Armagetron: http://armagetron.sourceforge.net/, 2005.