Multi Agent System For Intelligent Game Cinematography

Erick Baptista Passos Esteban W. Gonzales Clua

Centro Federal de Educação Tecnológica do Piauí, Brasil
Universidade Federal Fluminense, Instituto de Computação, Brasil

Abstract

This paper describes a system to improve the visual experience of spectators in massively multiplayer online games. The proposed architecture uses distributed agents to detect actions and control the camera using cinematographic techniques for computer graphics.

Keywords: Game spectators, cinematography, multi-agent systems.

Authors' contact:
erickpassos@gmail.com
esteban@uff.br

1. Introduction

A computer game can be seen as a virtual world, thus making it as believable and attractive as possible is a challenge for both development and artistic teams. This challenge is becoming more important as games gain a new kind of audience: spectators. This audience can bring more players to the game itself or even serve as a target for in-game advertising [Drucker et al. 2002].

Carefully designed computer graphics and realistic physics play an important role in this context since they have a direct impact on players and spectators immersion. However, having good graphics is not enough, as poor camera control and unconvincing behavior of any AI controlled object can ruin this immersion. The movie industry and also academia has already demonstrated that cinematography, the methods for controlling cameras and lights in a scene, can influence the perception of moods, emotions and actions by the audience [Brown 2002; Mascelli 1965].

One has to remember that games, being interactive applications, are different from films in the sense that there's no prior knowledge or control over the actions happening in the scene. For proper use of cinematography to improve spectators experience in interactive games it's also important to recognize these actions in a way to form coherent storylines [Pinhanez 1999]. To apply these storytelling and cinematographic camera techniques to massively multiplayer online games, the ideal candidates to passive spectators, more difficulties arise:

a) With the huge amount of actions happening at the same time, how to decide what is relevant to show to each spectator without compromising performance?
b) How much of this task can and should be automated and how much control let to the spectator?
c) How to explore the, possibly under loaded, computing power of this hugely distributed grid of spectator machines to help improving their experience?

There has been little to no work so far trying to answer these questions. In this paper, we propose a multi-agent system to efficiently distribute the tasks needed by intelligent camera control for game spectators. The main goals of our project are:

a) Creating real-time cinematographic experience for game spectators;
b) Massive distribution of cinematography agents over servers and clients;

The remaining of the paper is organized as follows: section 2 shows state of the art research in related areas; section 3 explains system architecture and challenges; finally, section 4 concludes the paper and discusses future work.

2. Related Work

A good amount of research has already been done in applying cinematographic techniques and other intelligent mechanisms of virtual camera control in computer games. The majority of this research so far proposes the creation of higher level mechanisms for camera manipulation using cinema oriented constructs and language such as shots, cuts, directors, editors and cinematographers [McCabe and Kneafsey 2006; Hermann and Celes 2005; Hornung 2003; Drucker 1994; He et al. 1996]. These efforts often use the concept of film *idioms*, commonly used methods for camera positioning in recognizable scenes, like a dialog between two characters, a boxing match or a car pursuit.

Even though these are clever building blocks, they solve only part of the problem. Games are not like films, because there is no prior knowledge and control of what is going to happen in the scene. In this sense, there has to be a link with the storytelling perspective, filling these camera modules or agents with information about the actions taking place in the game virtual world. Pinhanez [Pinhanez 1999] proposed

models and algorithms to represent and detect actions in the context of a virtual world.

Hawkins [Hawkins 2002] proposed an architecture with three logical layers: directors, editors and cinematographers, comparing these with their counterparts in a real movie set, where the director is responsible to propose the film *idioms*. The editor is in charge of choosing the shots that will be in the final cut, while the cinematographer takes care of direct camera positioning. Bringing this organization to virtual cameras in computer graphics makes it possible to decouple the different modules needed by our distributed system.

We are proposing a system that is a complementary work to this previous research, where some of these techniques will be part of it, while others will not. As can be seen in the next sections, our agents are based on the architecture proposed by Halkins [Halkins 2002] but distributed over the network. Our director agent is closely related to the work done by Pinhanez [Pinhanez 1999] and also uses film *idioms* to communicate with the other types of agents.

3. System Architecture

The architecture is meant to be attached to an existing game engine, which has to provide a basic library with a scene graph, client-server communication and simulation loops. The agents communicate by using asynchronous messages and are organized in three different logical layers: directors, editors and cinematographers. Since the system is targeted to networked games with complex virtual worlds, these layers are distributed over this network. These agents and their communication are detailed in the next sections.

3.1 Director agent

In our system, directors are distributed and independent agents, being responsible for detecting actions based on real time events generated by game objects. They run on *player* machines to monitor locally visible objects, like the main character, bots or even passive physics entities. Being unable to control the story, their only responsibility is to detect and keep record of the actions involving the monitored objects. We use the term action as proposed by Pinhanez [Pinhanez 1999], where they are meant to represent the events that happen in the virtual world and its context. By distributing these directors to client machines it's possible to simultaneously keep track of different storylines, each one comprising a set of related actions, without increase the load on servers. Figure 1 shows a diagram for the player client.

Each action may consist of several smaller ones, each one described by a simple verb, representing the event, and the list of game objects involved and the type of involvement. Since a single object can be involved in several actions, these are also indirectly connected, making it possible to follow the storyline of a character. Director agents keep records about past actions detected in local memory and persistent storage for future use, as explained in the next section. This history information is kept as a time-oriented graph, with causality information whenever possible, with current actions at one end. Based on this history registry, the director maintains a list of possible film *idioms* for each storyline's current actions.

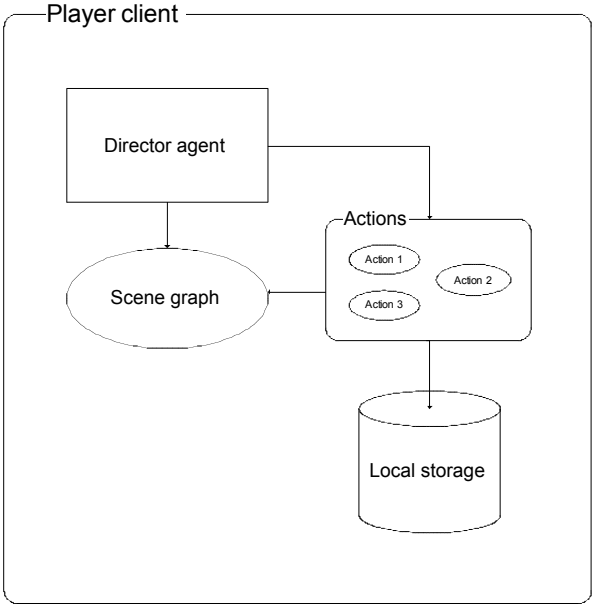

Figure 1: player client

3.2 Editor agent

The editor is an agent that runs locally on the *spectator* machine and is responsible for collecting action information from remote directors. The editor will get a list of active director agents from a server and choose one storyline to follow. The editor then connects to that particular director and gets the information related to the current actions and suggested shots. It chooses the appropriate shot and passes this information to the next layer, the cinematographer, to properly position the camera, based on the suggested idiom.

The editor is an intelligent semi-automated interface with the *spectator*. When logging on to the game, one can inform the editor the wanted mood/theme, being it full of fight action or more complex dialogs. With more complex criteria, the editor can make detailed queries to the servers and get a list of more specific active directors on the network. It's also possible to create a bookmark of favorite characters and storylines for each *spectator*. Combining this editor information with a real time voting system and the actions history from the directors, servers can automatically choose interesting highlights to record permanently, without compromising performance and using real time

feedback from the audience. The spectator client diagram is shown in figure 2.

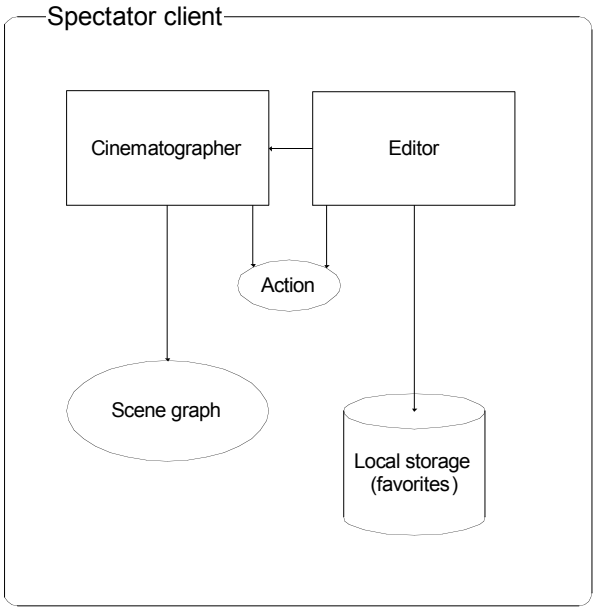

Figure 2: spectator client diagram

3.3 Cinematographer agent

The cinematographer agent also runs locally on *spectator* machines, and is responsible for directly controlling the camera, following the shot rules chosen by the editor, which in turn received them from the director. The architecture does not specify any *idiom* format for this shot information, neither the algorithms used for directly positioning the camera. Occlusion detection and line of action coherence are all responsibilities of the local cinematographer. One important decision is to develop this camera positioning algorithms based only on the action and suggested shot information, making it possible to reproduce a whole storyline without keeping more persistent information.

3.4 Communication

The proposed system involves three machine types: servers, player and spectator clients. Three different agent classes were also described: directors, editors and cinematographers. The servers don't need to run any agent. The only extra task for them is to keep databases of active and inactive directors and related storylines for querying purposes, and also for the spectators feedback system. Notice that game simulation servers, not described here, are still needed, since they are still responsible for coordinating the simulation between the player clients. Player clients run the director agents while spectator clients run the editors and cinematographers. This architecture is shown in figure 3.

As can be seen in figure 1, directors periodically send summarized information about current storylines to the servers, which are constantly queried by editors. The editors, after choosing the desired storyline, start a communication with the director, for receiving actions and suggested shots. This information needs to be exchanged once per cut only, not compromising bandwidth. Communication between editors and cinematographer is local. Both spectator and player machines rely on the simulation servers for updated scene information of game objects.

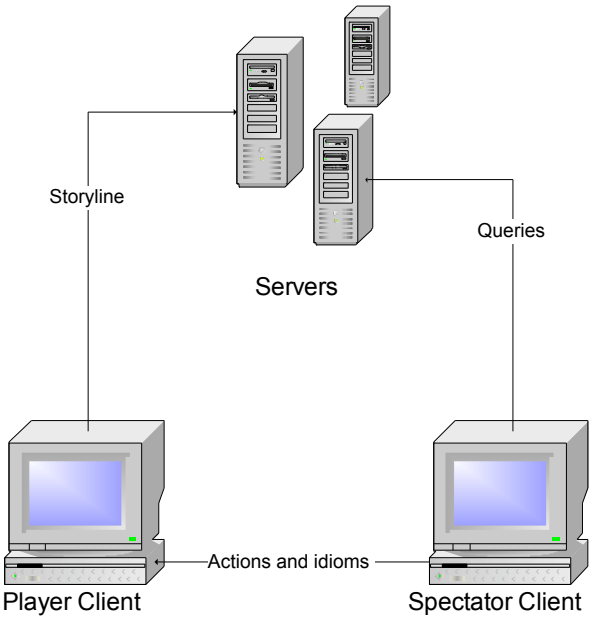

Figure 3: system architecture

4. Conclusion

We've proposed a new architecture for a multi-agent system to improve the experience of spectators in massively multiplayer games. This architecture is strongly based on previously researched building blocks, but presents a distribution model that relies mainly on client machines, not imposing more computing complexity or relevant network traffic to the already overloaded simulation servers.

This project is still a work in progress and we expect to have experiments results with the prototype soon. We're also investigating the use of this architecture to distribute other storytelling related tasks, like the detection of future events by the director based on physics engine prediction. This can also improve spectators, and also players experience.

References

BROWN, B. 2002. Cinematography: image making for cinematographers, directors and videographers. Oxford: Focal

DRUCKER, S.M., HE, L., COHEN, M., WONG, C. AND GUPTA, A., 2002. Spectator games: a new entertainment modality for networked multiplayer games [online] Microsoft Research. Available from: research.microsoft.com/~sdrucker/papers/spectator.pdf [Acessed 17 July 2007]

DRUCKER, S., 1994. *Intelligent camera control for graphical environments*. PhD Thesis, Massachusetts Institute of Technology

HALKINS, B. 2002. Creating an event driven cinematic camera. *Game Developer Magazine*. Sep/Nov 2002.

HE, L., COHEN, M. AND SALESIN, D., 1996. The virtual cinematographer: a paradigm for automatic real-time camera control and direction. *In: Proceedings of SIGGRAPH '96*. p 217-224, 1996

HERMANN, R., CELES, W., 2005. Posicionamento automático de cameras em ambientes virtuais dinâmicos. *In: Proceedings of IV workshop on games and digital entertainment of the Brasilian Simposium on Computer Games and Digital Entertainment*, 2005

HORNUNG, A. 2003. *Autonomous real-time camera agents in interactive narrative and games*. MsC Dissertation, Kunsthochschule für Medien Köln [Academy of Media Arts]

MASCELLI, J.V. 1965. The five c's of cinematography. Los Angeles: Silman-James Press

MCCABE, H. AND KNEAFSEY, J. 2006. A virtual cinematography system for first person shooter games. *In: Proceedings of iDig – international digital games conference. 2006*

PINHANEZ, C.S. 1999. *Representation and recognition of actions in interactive spaces*. PhD thesis, Massachusetts Institute of Technology