

Simulating Sensory Perception in 3D Game Characters.

Fabio Zambetta
School of CS&IT, RMIT University
GPO Box 2476V
Melbourne, Victoria, 3001
fabio.zambetta@rmit.edu.au

ABSTRACT

Extensive simulation of sensory perception for NPCs (Non Playing Characters) or bots in 3D games has been quite rare if not absent until recently. However, a few games have proven that proper simulation of senses can lead to interesting and novel gameplay, and it is likely that the trend towards more sophisticated simulation will continue. In this paper we analyze the existing techniques to simulate virtual senses, highlight their weaknesses and propose some ideas to improve over the main existing approaches. The work presented here is part of an ongoing research on 3D learning characters, funded by the RMIT Emerging Researchers Grant.

Keywords

3D Characters, Sensory Perception, Scene Graph.

1. INTRODUCTION AND RELATED WORK

Extensive simulation of sensory perception for NPCs (Non Playing Characters) or bots in 3D games has been quite rare until recent times. However, a few games such as Thief: The Dark Project [3] and Splinter Cell [2] have proven that proper sense simulation can lead to novel gameplay. Moreover, intelligent software used in military simulation or the film industry also adopted various models of perception to drive complex crowd behaviors. A notorious example is given by Massive, the software behind crowd animation in feature films such as Lord of the Rings, Happy Feet, Eragon or the recent 300. Massive employs a decision making strategy based on fuzzy logic to drive the action selection of characters in a crowd. The fuzzy rules would not be able to operate without built-in artificial senses: Massive currently implements hear and vision using a variation of a region sense manager citegameai, which is also the standard approach in games or military simulations with sense simulation.

Region sense managers accumulate sensory perceptions in

three stages:

1. *Aggregation* stage: Potential sensors are found in the virtual environment;
2. *Testing* stage: Potential sensors are tested to see if their signal reached any character;
3. *Notification* stage: Characters who registered for sensory events notification receive information that got through.

This approach suffers from a severe limitation though: The geometry of the game level or the virtual environment is not taken into account whereas distance is the only factor that discriminates whether a specific perception is sensed by a character. This can cause inaccuracies especially in indoor action games where contiguous rooms may be separated by thick walls or barriers.

Even though modified versions of this approach exist that try to handle many special problematic cases, a more comprehensive solution improving over a region sense manager is presented in [8] under the name of FEM (Finite Element Method) sense manager. This approach relies on breaking a game level down into a finite number of discrete constituents. A sense graph is built in analogy to a scene graph [11]: Each node in the directed acyclic graph represents a region of space where the signals can pass around unhindered, and as signals traverse the scene their effect is attenuated. This approach seems to be tying into the idea of a multi-view scene graphs [1] that advocates the need for different views of a scene graph, which is solely useful for rendering or animation tasks. Inaccuracies in the region based approach are readily solved by the FEM, at the expense of considerable computational power. Also, the need for sophisticated design tools emerges to let game designers mark the areas of the environment where specific sensory signals can travel. A coarse approximation can be provided by the level geometry, but this will result unsatisfactory in many practical cases. Besides, the type of approximation provided to the sense manager will largely affect the quality of an NPC's senses. The remainder of the paper is organized as follows: We first describe some core ideas behind the architecture of our own sense management system in Sections 2; Section 3 describes the prototype we are currently developing, and finally Section 4 highlights our future work.

2. A NOVEL SENSORY MANAGEMENT SYSTEM

Before we analyze the basic building blocks encompassed in the architecture of our sensory management system we will give an overview of the main principles underpinning its design.

First and foremost, we wanted our sensor manager to be able to output fuzzy predicates, because the decision making architecture that we have in mind will use fuzzy logic, as well. The decision making approach we will implement is based on a fuzzy extension of [4] [14], where an architecture for 3D learning embodied agents is described, using an XCS classifier system [12] for action selection. While an FCS (Fuzzy Classifier System) [5] has clear advantages in terms of scalability over a standard FIS (Fuzzy Inference System) [13], classifier systems tend to be more complex to manage. Our future research in the decision making process will put emphasis on deriving a fuzzy rule based system that is simple enough to work with, scalable but that can capture at the same time the complexity of the problems we are set to solve.

Thereafter, we needed our sensor manager to describe the world in real-time: this has profound implications in terms of the scalability and performance of the approach we will adopt. Probably, the amount of resources currently available to AI developers will limit the accuracy of the solution that can be provided. Nevertheless, the current trend for PCs is to adopt multi-core CPU architectures, and the same holds for next-generation consoles such as the Xbox 360 and the PS3. Therefore, it is likely to expect that the quality and accuracy of NPCs sensory simulation will increase accordingly.

Finally, care must be taken in not oversimplifying the representation of the world, as this risks introducing aliasing i.e., the inability to learn the appropriate concepts because the perceptual representations cannot distinguish between the things they need to learn about [7].

We will now describe how our sense management system produces sensory percepts that will be later used in the decision making process (see Figure 1).

The operations performed by the system are executed in three specific sub-stages:

1. The scene graph (or graphics database) is queried using the spatial information of the NPC under consideration. An approximation of the scene graph is derived, which is going to be further processed. We will refer to this as the NPC view, in analogy with the graphics database metaphor;
2. The NPC view is processed giving rise to a set of percepts;
3. The percepts are expressed using fuzzy predicates (e.g., Table1 IS Very Close, etc.);
4. All the relevant percepts are collected and sent to the decision making module.

The first stage does not present a great deal of problems as long as the underlying implementation of the scene graph includes an operator to perform fast spatial queries. This is

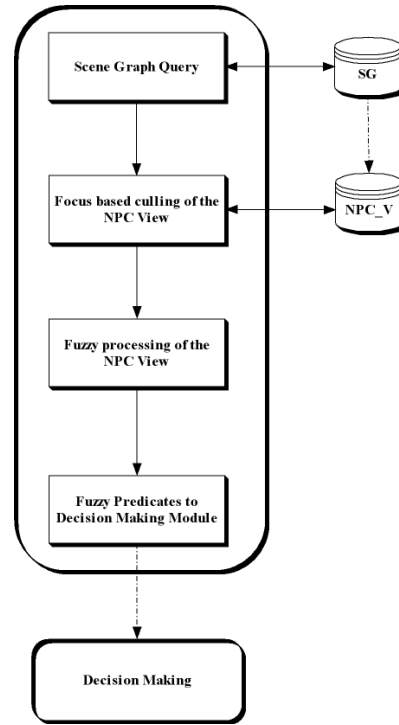


Figure 1: The Sensor Manager Architecture.

generally the case regardless of the data representation used, and even more so if the scene graph is combined with spatial subdivision techniques (BPS trees, kd-trees, octrees [9]) or a portal system [9] [10]. The type of query involved will extract an NPC view of the scene graph based on its spatial features (position, orientation, bounding volume, etc.).

The second stage performs a culling operation on the NPC view, using an estimate of the NPC level of attention. A naïve implementation of the level of attention would rely solely on predetermined values given by game designers as attributes of the NPCs. A better strategy for sight could involve for instance fine tuning such values after eye-tracking data is gathered from players' activity. Clearly, culling is performed to reduce the amount of sensory information that needs to be processed, but also to mimic the typical limitation of human sensory apparatuses.

The third stage is problematic, but it is our opinion that if a good implementation is built then the whole approach will be successful. The rationale for this stage stems from an obvious observation: If we want to have human-like decision making for our NPCs then we need to provide information to the decision making system that better resembles the noisy and uncertain type of information human beings manage. In other words, we do not instinctively perceive an object as "3.576 meters away", but we would say that it is "more or less close" to us. Only if we started to measure the distance between our position and the object's would we be able to refer to precise numerical data. This is at odds with the classic approaches in Computer Graphics and Artificial Intelligence where we usually try to achieve the best possible level of performance, but ironically this problem postulates the simulation of both human deficiencies and strengths at the same time. Besides, the use of fuzzy predicates to describe

sensory perceptions (e.g., “close”, “big”, “smelly”, “noisy”, “fast”, etc.) brings another clear advantage: Different fuzzy membership functions can be used by different 3D characters. Such an approach enables fine tuning sensory perceptions for each different human being, as different perceptions of *close* or *noisy* may be given. Another possibility would be designing a non-realistic perception system: What kind of perception system would be adequate for a dragon, an ogre, or finally an alien such as the ones found in Starcraft? Once again, hyperrealism may not lead to the best results in a video game scenario.

The fourth and final stage has just the function of collecting the percepts and sending them to the decision making module.

3. A PRELIMINARY PROTOTYPE

We are currently in the process of developing a prototype using TGE (Torque Game Engine) [6]. TGE contains some pre-packaged starter kits that allow developers to create and customize applications very fast. At the moment we have chosen to customize the FPS starter kit (see Figure 2), but of course additions to the existing code base have been necessary, and will also be in the future. The goals of the experiments we are planning on implementing will be:

- Testing the actual performance of the approach: How many queries can be realistically performed in any AI computation cycle on a standard PC, without compromising the game’s overall performance?
- Testing the scalability of the approach: As a consequence of the previous point, is it likely that the approach presented here can scale to different orders of magnitude? If so, we will need to quantify that.
- Testing a scenario integrating the complete sensory system: Even though the approach may be computationally sound, we still need to find gameplay scenarios conducive to players’ satisfaction. These experiments will require play testing and questionnaires filling.

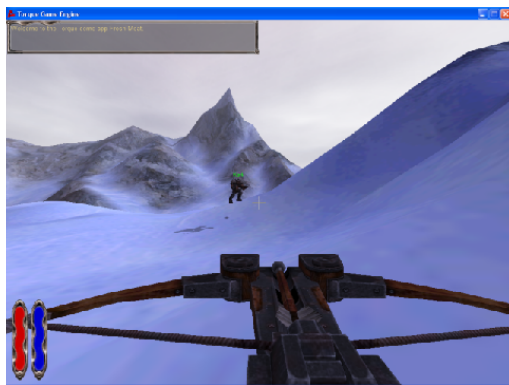


Figure 2: The current version of the prototype running in TGE.

4. FUTURE WORK

Our efforts are currently aimed at implementing a prototype that will completely address the points raised in Section 2.

The next step will clearly involve testing the experimental hypothesis described in Section 3. The future of sensory perception simulation in games seems to be quite a promising and exciting one. That will hopefully lead to new and interesting developments in the area of game design and development.

5. ACKNOWLEDGEMENT

This work is supported in part by RMIT University under the **RMIT Emerging Researchers Grant** (2006). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect those of RMIT. Moreover, we would like to thank Garage Games for donating a license of their **Torque Game Engine**.

6. REFERENCES

- [1] Scenegraps: Past, present, and future. <http://www.realityprime.com/scenegraps.php>.
- [2] Splinter cell. www.splintercell.com/.
- [3] Thief: The dark project. [http://en.wikipedia.org/wiki/Thief_\(computer_game\)](http://en.wikipedia.org/wiki/Thief_(computer_game)).
- [4] F. Abbattista, A. Paradiso, G. Semeraro, and F. Zambetta. An agent that learns to support users of a web site. *Applied Soft Computing*, 4(1):1–12, 2004.
- [5] A. Bonarini. Anytime learning and adaptation of fuzzy logic behaviors. *Adaptive Behavior*, 5(3–4):281–315, 1997.
- [6] K. C. Finney. *Advanced 3D Game Programming All in One*. Course Technology PTR, Boston, 2005.
- [7] A. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, Department of Computer Science, University of Rochester, 1995.
- [8] I. Millington. *Artificial Intelligence for Computer Games*. Morgan Kaufmann, San Francisco, 2006.
- [9] T. Moller, E. Haines, and T. Akenine-Moller. *Real-Time Rendering*. AK Peters, Wellesley, second edition, 2002.
- [10] A. Perez. Peeking through portals. *Game Developer Magazine*, 5(3):44–48, 21998.
- [11] J. Wernecke. *The Inventor mentor : programming Object-oriented 3D graphics with Open Inventor, release 2*. Addison-Wesley Professional, Boston, 1994.
- [12] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
- [13] L. Zadeh. Outline of a new approach to the analysis of complex systems. *IEEE Transactions on Man, Systems and Cybernetics*, 3:28–44, 1973.
- [14] F. Zambetta and F. Abbattista. The design and implementation of SAMIR. In R. Khosla, R. J. Howlett, and L. C. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems, 9th International Conference, Melbourne, Australia, Proceedings Part II*, Lecture Notes in Computer Science, pages 768–774. Springer, 2005.