An abstract spatial device for human behavior models, 
user performance data collection, and game control user interface

Bruno Emond 
National Research Council Canada 
Institute for Information Technology 
1200 Montreal road, 
Ottawa, ON. K1A 0R6 
1-613-991-5471 
bruno.emond@nrc-cnrc.gc.ca

Keywords: Game engine, Cognitive models, Distributed systems, ACT-R.

ABSTRACT: This paper presents a system in which cognitive models embedded in an abstract spatial device communicate with a game engine. The communication is decoupled by the introduction of a message broker. One benefit of the decoupling allows messaging resources, developed for controlling avatars in the game engine with cognitive models, to be reused as building blocks of a performance observer, and an end-user application for design and real time control of game entities. The system is presented in terms of the distribution of representations and processes across system components to support cognitive models and avatars' visual perceptions and movement in space. The system also supports an end-user interface independent of the game engine editor for the creation of environments such as rooms with various textures and sizes, for the insertion of objects and avatars, as well as changing objects’ position during game execution.

1. Introduction

The technical and cultural boundaries between modeling, simulation, and games are increasingly blurred (Narayanasamy, Wong, Fung, & Rai, 2006), providing broader access to capabilities in modeling and simulation and further credibility to game-based applications. The recent broad availability of game engine application programming interfaces, as well as the wide deployment of computer networks, and multi-core desktop computers, which make high-performance computing (HPC) more affordable for running complex simulations, provide an interesting platform for the development of a new generation of environments combining HPC and gaming technology (National_Research_Council, 2010). For example, graphical processing units, a core technology for video-intensive games, are greatly outpacing commodity processors with GPU chips including up to 240 cores (National_Research_Council, 2010)

The realm of game engines is being extended outside the immediate location of traditional game consoles and computers to pervasive games (Wietrzyk & Radenkovic, 2007), as an essential component in bi-directional cross-reality merging of real world sensors, actuators, and virtual worlds (Paradiso & Landay, 2009). There is also a growing interest for the application of game engines outside the traditional entertainment market, such as models for robot simulation (Faust, Simon, & Smart, 2006), emergency preparedness (Persson & Wide, 2007), and serious games aiming at providing a skill acquisition context using similar conditions to be found in the environment where the skill is to be performed (Narayanasamy, et al., 2006).

Serious games development can benefit from the availability of existing game engines as long as they can be tailored to fit learning objectives. In addition to the learning requirements, multi-player networking capabilities often need to be extended to allow serious games to be inserted in a federated infrastructure of simulations, building models, terrain databases, and human behavior models.

In spite of excellent demonstrations that human behavior models can interact with game engines (Best & Lebiere, 2003, 2006; Wray, Laird, Nuxoll, Stokes, & Kerfoot, 2005) and virtual worlds (Veksler, 2009), the integration of human behavior models in simulation federations is still a challenge, in particular when considering the specification of standardized interface rules so that human behavior models can be integrated (RTO/NATO, 2010). Bringing together autonomous, but yet interoperable systems, is certainly challenging, and involves issues ranging from network transport, protocols, and agreed upon message formats.

Aside from shared and agreed upon information representations, a useful method to allow autonomy and interoperability is decoupling (Aier & Winter, 2009). Decoupling is a general system development
and analysis strategy to reduce complexity as well as a method for building service oriented systems (Narayanan, 2009).

The paper presents an overview of a system in which cognitive models embedded in an abstract spatial device communicate with a game engine (Terathon_Software, 2010). The communication is decoupled by the introduction of a message broker (Apache_Software_Foundation, 2010). This simple solution was favored over HLA (IEEE, 2000a, 2000b, 2000c) in order to reduce development complexity while still working in a publish and subscribe framework. The abstract spatial device plays a central role in the system by issuing and receiving messages from the game engines. Interestingly, the same messaging resources used for the cognitive models-game engine interactions can be reused for a performance observer, and an end-user application for designing and real time control of game entities, such as the insertion of objects and avatars, as well as changing objects’ position during game execution.

The system architecture is described in the next section in terms of the distribution of representations and processes across system components to support avatars’ visual perceptions and movement in space. Then, the functionalities of the end-user interface, and the performance observer are briefly presented.

2. System Overview

The paper is essentially concerned by the interaction between cognitive models and a game engine used in close battle and firearm training (Emond, Fournier, & Lapointe, 2010a, 2010b). The first element required for this integration is a device with which ACT-R models can interact models (Anderson, 2007; Anderson & Lebiere, 1998; Bothell, 2010). The abstract spatial device (ASD) plays this role in support of running both constructive simulations (Emond, 2010), and virtual simulations with a user player interacting with the game engine. The other application of the abstract spatial device is to interface between the game engine and the performance observer module, and the design and control user interface. The architecture (Figure 1) is very similar to the separation in robotics between sensors-effectors, spatial support (symbolic), and cognitive layers (Kennedy et al., 2007). The sensor-effectors layer corresponds to the game engine, the spatial support to the abstract spatial device, and the cognitive layer to the ACT-R cognitive models.

The main problem the system aims at solving is one of integration of cognitive models with other parts of the simulator system such as laser tracking (Lapointe & Godin, 2005). Decoupling the system components with the use of a message broker allows an easy way to inter-connect autonomous systems through a publish-and-subscribe mechanism, with message topics as a means to specify device specific or general communication channels.

Figure 1. System overview.

3. Embedded Cognitive Models

3.1 Abstract spatial device

Simulation networks can be comprised of many elements such as games, physics engines, flight simulators, terrain and building databases. The main function of simulation networks is to allow complex simulations to be built out of smaller independent components. Military simulations are typical simulation networks with many types of simulator endpoints exchanging information. The abstract spatial device is one of these endpoints and provides a common interface for embedded cognitive models to interact with other simulation endpoints. In the current system the abstract spatial device is also a mediator between the game engine and both the performance observer, and the end-user application for designing and controlling the game engine.

The intention of the abstract spatial device is to provide an adequate level of abstraction from the details of a game engine while allowing for a cognitive model to perceive and act deliberately. The focus of the current work is therefore at a relatively high level of spatial categorization and assumes that some simulation/game objects of perception, action and planning are already labeled in some way, or that the computation operating at the abstract spatial device level generates categorized chunks to cognitive models. The abstract spatial device isolates the cognitive model from the game engine. This isolation not only allows the construction of ACT-R chunks using specialized
methods for creating visual location and visual object chunks (adding a Z coordinate), but the device can also execute processes that compute perceptual conditions (calculating object occlusion) and motor actions (changing avatar locations) which then are synchronized with game object instances through message passing.

The abstract spatial device holds an environmental frame of reference. All reference frames are defined as a combination of a point of origin and one or more axes (Zacks & Michelon, 2005). From a reference frame, the spatial properties of objects can be specified. Zacks and Michelon identify three major classes of reference frames: object-based, relative to an object internal spatial structure; egocentric, relative to the self; and environmental, relative to some fixed feature or features of the environment (Zacks & Michelon, 2005). Object-based reference frames assign axes to objects such as top-down or front-back properties. An egocentric frame of reference is in reference to the self-orientation such as a coarse distinction between the sagittal (division of left and right), coronal (division of back and front), and transverse (division of upper body and lower body) planes. More refined egocentric reference frames are required to capture the cognitive representation of space from an egocentric perspective, such as eye-centered, body-centered, and body effectors centered. Finally, the environmental reference frames locate things relative to axes defined with respect to a fixed space, such as the principal axes of a rectangular room or the cardinal directions in geography (north, south, east, west) (Zacks & Michelon, 2005). At any moment, the origins of each reference frame maintained by an organism are located relative to the other reference frames (Zacks & Michelon, 2005). For example, an agent’s bearing, could be the origin of all object locations from an egocentric perspective, but this bearing will have a different value from the environmental frame of reference.

3.2 Visual and spatial representations

The abstract spatial device holds an environmental frame of reference with a point of origin set as longitude and latitude coordinates as well as North-South, East-West and altitude axes. Even though these axes can be important for navigation, they are not based on the self as a point of origin. There is general understanding that in an egocentric reference frame, locations are represented with respect to the particular perspective of a perceiver, whereas an allocentric reference frame locates points within a framework external to the holder of the representation and independent of his or her position (Klatzky, 1998).

The abstract spatial device environmental frame of reference must be distinguished from allocentric representations. The environmental frame of reference is an objective specification of the spatial position of objects of perception, including the agent or avatar that embodies a cognitive model. Allocentric spatial representations are cognitive approximations of the environmental frame of reference.

Egocentric and allocentric spatial representations are located in the cognitive models, not in the abstract spatial device. Recent findings from spatial cognition and cognitive neuroscience indicate that one system is centred on the processing of immediate sensory information for the determination of an egocentric perspective, while the second system relates to allocentric spatial information, or a more enduring and less transient spatial characteristics of the environment (Avraamides & Kelly, 2008).

Significant efforts have been made to develop ACT-R spatial modules, and models of spatial representations, and navigation (ACT-R_Web_Site, 2010). The current implementation has no added spatial ACT-R module and is only using the embedded visual module (Byrne & Anderson, 2001) of ACT-R 6 with extended visual-location and visual-object chunks with additional slots. The additional slots include a third spatial coordinate, Z, as well as distance, size, and bearing from self (Gunzelmann & Lyon, 2007). The interpretation of the X, Y, ad Z coordinates correspond respectively to the sagittal, transverse, and coronal axes. The abstract spatial device, containing the embedded cognitive models, generates the visual-location chunks as the cognitive model’s field of view content.

At this point in time the models are meant to perceive and act in a small room with little demand for building an allocentric spatial representation for navigation planning. No spatial module like those of Gunzelmann or Robbins et al. are included in the architecture (Robbins, Carruth, & Morais, 2009).

3.3 Messages, synchronization and movements in space

There are three main issues when looking at the integration of software agents and games: information representation, synchronization, and communication (Dignum, Westra, van Doesburg, & Harbers, 2009). The first two will be discussed in the present section, while the issue of inter-agent communication is for future work, and has not been addressed in the current cognitive modeling implementation.

The paired combinations of the abstract spatial device with cognitive models, the performance observer, and the end-user application to design and control game
settings form three autonomous system components. As was outlined in the system overview section, a message broker mediates the communication between the game engine and the cognitive modeling resources. This is in contrast with most ACT-R systems for communicating with and external application, which use a direct socket connection between the ACT-R device and the application. The messages scheme is simple and uses key-value pairs.

There are two types of messages exchanged between the abstract spatial device and the game engine related respectively to perception and motor actions. The motor messages are issued to the game engine when a cognitive model makes a request to a body module. The body module is a simple ACT-R module extension that recognizes chunk specifications for changing the heading and/or the position of an avatar. When a cognitive model formulates a movement in space request, it is applied in the abstract spatial device by launching a process or thread that executes until either the process ends autonomously, or the process is ended by another cognitive model request. During the execution of a process in the abstract spatial device, messages are sent to the game engine to move a corresponding avatar to the same coordinates. Continuous movements in the abstract spatial device are approximated as a series of discreet movements in the game engine. Both, the game engine and the abstract spatial device share the same coordinate system, with distance in meters and a point of origin as longitude latitude, and altitude values.

The abstract spatial device is not always completely synchronized with the game engine world model. Only the subset of the game engine world relevant to the perceptual field of cognitive models is updated. The distribution of computational resources between cognitive models, the abstract spatial device and the game engine could even be extended to the point where cognitive models could only deal with high-level goals, while the AI embedded in the game engine could deal with path finding (Hart, Nilsson, & Raphael, 1968; MacGregor & Leung, 2009).

4. Reuse of Simulation Resources

An unforeseen consequence of the decoupling the abstract spatial device and the game engine with a message broker was the possibility of reusing some of the messages needed for cognitive models for other purposes. In particular, movement in space messages generated by the cognitive models' body module are used as a basis for an application to design and control game scenarios in real time.

The design and control user interface provides end-users or instructors with a set of tools to design scenarios and instruction levels as needed. From the application, an instructor can insert avatars controlled by cognitive models, or floating geometric figures for early shooting skill and judgment learning. The drag and drop functionality is supported by the same messages produced by the cognitive models body module. A user can grab a moving object using the user interface and change its properties on the fly while someone is playing the game. This is particularly useful when an instructor wants to modify the conditions in the middle of a game. In addition, the user interface supports the design and creation of room elements such as the size of the room, presence of doors, or columns.

The messages to support perception are also reused for a performance observer, which collects information about the performance of user players on shooting tasks. The performance observer can set shooting objectives that are used to control the state of objects depending on the number of shots on target. In addition, because of the presence of the message broker, any device subscribing to the data-performance topic can receive data from user actions such as a module to build learners' profiles.

4. Conclusion

The paper presented an overview of a system in which cognitive models embedded in an abstract spatial device communicate with a game engine. The communication is decoupled by the introduction of a message broker. One benefit of the decoupling allowed messaging resources, developed for controlling avatars in the game engine with cognitive models, to be reused as building blocks of a performance observer, and an end-user application for design and real time control of game entities. The system is presented in terms of the distribution of representations and processes across system components to support cognitive models and avatars' visual perception and body movements. The system also supports an end-user interface independent of the game engine editor for the creation of environments such as rooms with various textures and sizes, for the insertion of objects and avatars, as well as changing objects’ position during a game execution.

Future developments of the current work requires a system validity evaluation, and could include the integration of terrain and building standards to the abstract spatial device, and the exploration of game-AI as a means to distribute intelligence throughout the system.

5. References


Author Biographies

BRUNO EMOND is senior research officer at the National Research Council of Canada with interests in the application of cognitive modeling technology in training simulators, as well as learning and performance in multimedia and broadband e-learning environments.