State of the Art in Commercial AI Middleware

BÖRJE FELIPE FERNANDES KARLSSON
BRUNO FEIJÓ

ICAD/IGAMES/VisionLab - PUC-Rio, Brazil
Computer Science Department – PUC-Rio, Brazil
{borje, bruno}@inf.puc-rio.br

Abstract

This work presents the state of the art of artificial intelligence (AI) middleware through the analysis of commercial middleware. It starts with a discussion of the concept and context of an AI middleware. Then, current commercial engines' approaches to the problem are discussed. Finally, we discuss their impact, open issues to be addressed and some future directions.

Keywords: *artificial intelligence, AI engine, AI middleware, computer games, software architecture.*

1 Introduction

As the market of digital entertainment products (especially digital games) grows, these products get more and more complex and their users present higher and higher expectations, requiring quality and believability in the character behaviours. Because of these facts, artificial intelligence (AI) functionalities are no longer held in a secondary level during development. Despite of this, AI applied to entertainment products development, remains a complex domain and relatively unexplored. A great source of solutions and techniques is the academic community. Almost every game genre can benefit from what the many AI areas have to offer in the creation of more realistic and interesting environments and NPCs (Non Player Characters) or even in the control of the narrative flow.

Besides the previously cited, the application of artificial intelligence in digital games offers very interesting challenges and advantages from a research point of view. [10], for instance, presents as advantages for the AI community, the growing realism in today games and the fact that game developers create support for others to easily modify the games (building the so called mods); what turns games in a highly attractive alternative to the creation of complex simulation environments. The complexity in current games is reaching a level comparable to the real world, but on the same time allowing the researchers to focus on the AI issues and forget about real world motors and sensors. In other words, digital games provide a great test-bed for artificial intelligence techniques and models (as shown for example in, [9] and [17]).

As challenges for the AI research community, digital games offer dynamic environments that require complex decisions (often based on incomplete knowledge of the world), require real-time responses (limiting the available time for reasoning), and may have to handle resource management. Also, games provide an environment where it is possible to develop and test machine learning techniques, human-computer interfaces and interaction, knowledge representation and intelligent architectures.

With the ever growing necessity of AI functionalities and the fact that some techniques are already in use in game development, supporting tools were created in order to help with these tasks but these tools presented little flexibility [13]. Without a certain level of support, a developer will spend a great part of his time struggling with low-level details or re-implementation of common functionalities. It is believed that the next big step in the quality of AI game techniques depends on the creation of AI middleware, to alleviate developers and allow them to concentrate on creative tasks related to AI.

In the following sections this work presents a discussion of the concept and context of an AI middleware (emphasizing the relations of traditional AI areas with computer games). Then, some approaches to the problem of creating an AI middleware are presented, followed by a taxonomy regarding design methods and componentization, and related research. Finally, we discuss the impact of such middleware, open issues to be addressed and future directions.

2 Middlewares

AI middleware or AI engines for digital games are a relatively new field (both in academia and industry) in its infancy. Most of them use well know "game AI" ([28][30]) technology and some try to improve on that by bringing concept from academic AI [29], but there is still no agreement on what constitutes such piece of software and how to design one [21]. So in this work we'll try to provide a definition on what constitutes an AI middleware and provide an analysis of commercial products in this area.

2.1 Definition

In other areas related to computer games development, such as computer graphics, networking support and physics modelling, it is already a common approach to use (or at least to know about) pieces of software that help the digital games developer in the creation of the games, allowing them to focus on the creative side of the game. This is only now staring to be seen in the artificial intelligence segment of game development.

An Artificial Intelligence middleware is basically a layer of software that provides services for the game engines for performing the AI functions in a computer game. Usually also called an Artificial Intelligence Engine, AI middleware handles the process of producing the desired behavior or decision-making of intelligent agents present in the game world.

2.2 SOFTIMAGE/BEHAVIOR

A simple example of such a tool is the SOFTIMAGE/BEHAVIOR [8] toolkit, a toolkit for computerized animation that makes use of HFSMs in order to allow better representation and control of big real-time systems complexity, as complex crowds for example, allowing the animator to give intelligence to this crowd and to individual characters inside it. It also has its own visual tool for visual creation and editing of the HFSMs and supports a script language that enhances the tool flexibility. With BEHAVIOR, you can also visually debug your hierarchical finite state machines and scripts. Another service provided by the toolkit is a path-planning service, were it automatically generates waypoints and one can define behaviours for characters (object avoidance for example). The toolkit was already used in several commercial digital games, but it is too focused on animation, and most of its uses are related to producing video for games.

2.3 SimBionic

SimBionic [16] developed by Stottler Henke presents another approach (arguably richer) to the design of AI support software that matches somewhere in between FSMs and RBSs, by providing a framework for defining the objects that display behavior within the game world. This framework is very state-oriented, supporting the creation of complex hierarchical state systems.

These state systems have several components that [22] can be classified as descriptors and declarations. Descriptors are identifiers used to represent objects and behaviors that exist in the game world. SimBionic also describes attributes of objects as descriptors, for example, a Weapon descriptor could have two son descriptors Revolver and Rifle, Rifle in turn could have a son AR-15, and so on; and the gun attributes as weight, ammunition, would also be represented as descriptors. Declarations are symbolic associations used by the SimBionic project.

These associations consist of actions, predicates (built-in functions that provide access and evaluation services), behaviors and variables. A behaviour implies the selection of either another behaviour or an action, and is responsible for the “decision making” in SimBionic.

The SimBionic runtime engine provides simple C++ and Java application programming interfaces (APIs), and one basically needs to notify the engine of each clock tick and include some header and C++ implementation files with the game project to let it run.

SimBionic also provides an interactive debugger and a visual editor that greatly improve productivity and are also relatively non-tech friendly, allowing for authoring almost with no programming.

Besides that, the toolkit also lets entities communicate data and status information between them, either using group messaging or virtual blackboards, a feature not explored in most middleware.

2.4 AI.implant

From BioGraphic Technologies, the AI.implant [1] tool is an example of the usage of decision trees we have, an animation control engine that was designed to introduce AI to the computer game and video media character development process. Essentially, AI.implant provides autonomous character control for the game engine or animation engine offering pre-defined behaviors (like "Avoid Obstacles" for example) are assigned to the “agent” to implement a desired set of actions (preventing a guard from walking into a wall). Sensors are created for the “agent”, letting it perceive events in the game world, and based on those events use binary decision trees (BDT) to choose a course of action. The BDT can be used to create complex decisions of arbitrary depth. It is even possible to construct a finite state machine (FSM) using the BDT appropriately and agents can have default behaviors assigned that execute in the absence of behaviors determined by the BDT. As AI.implant work closely with 3D modeling software (through plug-in interface to Maya and 3ds Max), it also provides functionalities for the visual placement of waypoints in the terrain to help the characters path-finding.

There is also a C++ SDK for calling AI.implant functions from within a game.

It supports hierarchical pathfinding and by using the plug-ins one can automatically create a waypoint network which can be subsequently edited as needed [20].

2.5 DirectIA

Developed by Mathematiques Appliquees S.A., it is different from most of the common approaches as it uses a biologic/evolutionary approach [4]. DirectIA [3] provides tools and support for the creation of agents, with behaviours ranging from basic reactions to deep world state analysis. DirectIA is an agent-centric tool, which makes significant considerations of how and why a character makes a decision, with inspiration in the modeling of human and animal behaviours. Using a motivational model, i.e., an action selection mechanism that mimics the mechanism in animals. The DirectIA mechanism handles stimuli, emotion, states, motivations, behaviours and actions.

Such systems may be seen as a set of motivations that compete until it is decide which one must be applied to the situation, given the internal and external states, resulting in an emergent behaviour. The intelligent agents can weight tradeoffs, learn from their own experience and present behaviours nor specifically programmed by the game developers.

DirectIA, that was already presented above, provides different components tightly grouped, a motivational engine to model the emotions and needs of the intelligent agents, a behavioral engine to model the agent's decision processes, a communication engine that supports communication between the agents, a perception engine, an action engine, and a knowledge engine to store each agent's representation of the game world. But it may also be considered as a layered approach, as the high level functionalities provide a agent-based approach, and the lower level functionalities provide support for commonly used techniques as path-finding and steering.

DirectIA offers real-time decision and action behavior modeling tools, in the form of a high-level tool suite (that support complex agents and reactive agents) and a low-level tool suite (focused on techniques as hierarchical pathfinding and steering tools).

A GUI testing environment is also provided with DirectIA [23], but the toolkit is programmer oriented and the developer must create several C++ classes in order to use it. One can also define several behaviours via scripting and parameter files that are initialized and loaded into the DirectIA engines at run time.

2.6 Pensor

Most artificial intelligence engines (or so called ones) can be categorized as packs of similar functionalities, layered approaches or semi-independent components. Some of these packages could be: planners, path-finding, decision making, actions, sensing, infra-structure, learning, communication, emotions, motivations, coordination and tactical analysis.

One example of artificial intelligence middleware that is organized as sets of functionalities is Pensor [11] that is composed of a set of algorithms/techniques/technologies re-combinable to each individual project. Pensor has six planners, three of them being optimized path-finders (path-planners) [24] that take into account terrain analysis information when calculating the paths; a decision module that implements finite state machines and fuzzy FSMs, rules and a shell [25]; an actuator module that implements basic physics modeling and several steering behaviours; a perception module that provides several kinds of sensors and an infra-structure module that provides support for task priorities and resource managing for each task, as well as a scripting language interpreter.

Mindlathe used to provide the technologies that formed the basis of Pensor, but now it seems not to be available anymore.

2.7 RenderWare AI

An Yet another commercial artificial intelligence middleware is RenderWare AI [14], by Criterion Inc., that presents a more purely layered design focused on the developers, providing a architectural layer, a services layer, an agents layer (actually it is a behaviour layer) and a decision layer [27].

The Agents layer being a set of behaviours that can be instantiated by a game character (behaviours such as flee, ranged attack, etc.),

The Decisions layer supports a “brain object”. Every thinking entity has an associated brain object and this brain performs the decision-making process

The Services layer basically consists of a number of managers (path manager, entity manager, sound source manager, etc.) and the Architecture layer deals with initialization, updating and termination of the RWAI layers under the control of the game engine. Regarding path planning, RWAI also has a dynamic topology analyzer to help NPCs identify obstacles, enemies, locate sounds, etc.

Renderware AI offers fully open modifiable source code, a set of C++ classes that the developers can extend and tweak to create their own agents and also offers FSMs and neural networks for the characters decisions processes.

As almost every middleware available, it splits the game objects into two categories, thinking entities and passive entities to facilitate their representation inside the game world model.

One of the advantages of RWAI is that it can be used in conjunction with the other RenderWare platform components, with coupled with the available source code makes it more appealing to developers.

2.8 Emotion AI

A new engine is Emotion AI [18] by Neon AI; it tries to provide a new framework for game AI by simulating human emotional behavior to a reasonable degree. Only non-cognitive emotions in the first version of the engine [12] but can simulate a range of different motivational needs, and these needs having their current “satisfaction” levels adjusted.

The Emotion AI Engine conceptually divides emotions into three layers of behavior. At the top level are “reactions” or “momentary emotions”; these are the behaviors that someone displays briefly in reaction to events. The next level down are “moods”, these are prolonged emotional states caused by the cumulative effect of momentary emotions, that is signals of punishment and reward (i.e., are generated in response to incoming events and decay at rates dependent on the personality). Underlying both of these layers and always present is “personality”; this is the behavior that generally is displayed when no momentary emotion or mood overrides.

The brain model proposed is also affected by “motivational needs” (inspired by Maslow need hierarchy [31]) that are used as a range of agent needs currently requiring the most attention. For example: needs requiring the most urgent attention, needs (if satisfied) that will provide the largest reward, needs (if not satisfied) that will provide the largest punishment (negative reward) [12].

The environment input enter the brain model, is influenced by the need list, the brain model itself and then decides wich actions to perform and changes the current brain state.

With this model, the Emotion AI Engine Middleware SDK can provide more elaborate character behaviours, similar to the ones in Lionhead’s Black and White.

3 Other considerations

Apart from the approaches presented above, one can not forget approaches that make use of machine learning techniques integrated into the artificial intelligence engine in order to obtain new (possibly emergent) behaviours [2].

Other important considerations are: special attention should be paid to the knowledge base containing the goals, tactics and independent behaviours of a computer game, is its most important feature [17]. Without that, the second part, the inference engine (the component responsible for taking the actual decisions), would not be useful at all. And [7] states that it is extremely valuable to construct a good set of sensors as a priceless asset in order to improve the player experience interacting with the game world allowing the creation of much richer environments by the game developers and designers. Another approach to the problem of embedding AI in games is Spark! [15] that is a fuzzy logic editor created to help the use of this technique in games.

And one of the most important design issues when building an AI package is defining who the target user for the tool is. Tools like SimBionic and AI.implant for instance are targeted towards visual animators and game designers (especially AI.implant), presenting easy to use visual tools often integrated into 3D modeling software packages and using approaches that do not require programming language expertise. On the other hand, tools like DirectIA and Renderware AI are clearly focused on the game developers (especially Renderware AI) requiring the developers to have a good understanding of the lower level implementations of the game and often offering source code to be modified by the developers (not the case for DirectIA though).

4 Conclusion

Many AI middleware solutions are flourishing but little is known about the real possibility of creating a set of interface standards to ease the creation of artificial intelligence middleware and how these standards will relate to implementation issues.

Most of the current commercial AI middleware and AI engines propose their own way of doing things and only cover a relatively small subset of AI capabilities. Cooperation among the computer games industry and the academy is crucial in this endeavour for the creation of such standards even if the goals for each community are not the same.

Much more thought will have to be dedicated (and currently is being dedicated) to the question of standardizing artificial intelligence interfaces in computer games. With this in mind, the International Game Developers Association (IGDA) has set up the AI Interface Standards Committee to develop such interface standards, the initiative being a joint effort of game AI developers, middleware representatives and academics [9][10]. It is expected that as standards evolve, AI middleware can really gain widespread acceptance.

	DirectIA	**AI.implant**	**RWAI**	**SimBionic**	**Emotion. AI**	**Pensor**	**Behavior**
Decision support	Motivated decision graphs	BDTs	FSM and NN	FSM-like	Brain / emotion model	FSM and rules	FSM
Services	Pathfinding	Auto path generation, pathfinding	Auto path generation, pathfinding, other modules.	Agent Comm.	N/A	Pathfinding and basic physics	Path-planning
Behaviours	Templated	Pre-packaged	Pre-packaged	User developed	Moods / Temporary reactions / Personalities	Pre-packaged	N/A
Source code	No	Some	Yes	Some	N/A	No longer available	No
Tools	Scripts, templates, GUI	Maya & 3DS plug-ins	Skeleton code	Visual editor and debugger	N/A	N/A	Debugger and visual editor.

Table 1: A comparison between the different engines (inspired by [6])

Even with these issues still open, artificial intelligence in games will have a growing priority in the digital game development process for a long time, because it is still and area little explored but that already showed that can bring great advances in game design and gameplay. And AI middleware will be essential like graphics engines.

However, the "not invented here" syndrome (that is very common in the digital games industry), the fear of not having complete control over the game and a possible performance hit that may occur due to the AI middleware, "engine" or library are some of the obstacles for the massive adoption of AI middleware by game development companies. Some claim that middleware that does not have source code licenses adds great risks and that has been a big basis for not using certain libraries [5]. Also, attention must be paid so that the learning curve for implementing and integrating the AI middleware into a digital game title might be too high.

But on the bright side, AI engines are a reality and are being used more and more. One shall really evaluate the game needs and what the AI middleware is supposed to provide. This is for sure the most important step, because not understanding the game's requirements probably will lead you to make a poor decision [19].

This work is part of a bigger research that is trying to design a more academic AI middleware but that can be used in commercial games and is currently in a prototyping state [26].

5 References

[1] AI.implant - *Advanced AI for Games, Animation and Simulation*. URL: http://www.ai-implant.com/ (25/08/2004).

[2] Ding, Z. *Designing AI Engines with Built-in Machine Learning Capabilities*. In Proceedings of the Game Developers Conference 1999, San Jose, USA, 1999.

[3] Direct IA. URL: http://www.directia.com (19/08/2004).

[4] Donnart, J., Jakobi, N., Kodjabachian, J., Meyer, C., Meyer, A., Trullier, O. *Industrial Applications of Biomimetic Adaptative Systems*. Proceedings of HCP'99 - Human Centered Processes. ENST Bretagne Pub, Brest, France, September 1999.

[5] Keith, C., *From the Ground Up: Creating a Core Technology Group*, 2003, URL: http://www.gamasutra.com/features/20030801/keith_01.shtml (01/08/2004)

[6] Dybsand, E., *AI Middleware: Getting into Character. Conclusion*, 2003, URL: http://www.gamasutra.com/features/20030725/dybsand_01.shtml (20/08/2004)

[7] Leonard, T. *Building AI Sensory Systems*. In Proceedings of the Game Developers Conference 2003, San Jose, USA, 2003.

[8] Softimage Behaviour. URL: http://www.softimage.com/products/behavior/ (21/07/2003).

[9] Nareyek, A., Karlsson, B., Wilson, I., Chady, M., Mesdaghi, S., Axelrod, R., Porcino, N., Combs, N., El Rhalibi, A., Wetzel, B. and Orkin, J. (eds). *The 2004 Report of the IGDA's Artificial Intelligence Interface Standards Committee*. IGDA, June 2004. URL: http://www.igda.org/ai/report-2004/report-2004.html (27/08/2004)

[10] Nareyek, A., Knafla, B., Fu, D., Long, D., Reed, C., El Rhalibi, A. and Stephens, N. (eds). *The 2003 Report of the IGDA's Artificial Intelligence Interface Standards Committee*. IGDA, June 2003. URL: http://www.igda.org/ai/report-2003/report-2003.html (27/08/2004)

[11] Pensor. URL: http://www.pensor.net (10/06/2003).

[12] Wilson, I., *The Artificial Emotion Engine*, 2000 Spring Spymposuim on AI and Interactive Entertainment, USA.

[13] Rabin, S. *Designing a General Robust AI Engine*. In Game Programming Gems, Charles River Media, 2000.

[14] Renderware AI. URL: http://www.renderware.com/renderwareai.htm (11/08/2004).

[15] Spark!. URL: http://www.louderthanabomb.com/ (10/06/2003).

[16] SimBionic. URL: http://www.shai.com/products/ (20/07/2004).

[17] Laird, J., van Lent, M., *Developing an Artificial Intelligence Engine*. In Proceedings of the Game Developers Conference 1999, San Jose, USA, 1999.

[18] Emotion AI. URL http://www.emotion.ai/ (20/07/2004)

[19] Macris, A. *Effective Middleware Evaluation*. In Game Developer Magazine, May, 2003.

[20] BGT. *AI.implant for Games*, 2003, Whitepaper, BGT BioGraphic Technologies, Montreal Canada.

[21] Karlsson, B. *Issues and approaches in Artificial Intelligence middleware development for digital games and entertainment products*. In proceedings of the International Digital Games Research Conference 2003, Utrecht, The Netherlands, November, 2003.

[22] Fu, D. and Houlette, R. *Putting AI in Entertainment: An AI Authoring Tool for Simulation and Games*, 2002, IEEE Intelligent Systems, Vol. 17, No. 4, pp. 81-84.

[23] Masa. *Direct IA Datasheet*, 2002, Whitepaper, The Masa Group, Paris, France.

[24] Pensor. *Path planning to massive worlds*, 2002, Pensor whitepaper series, wp-2002-5-1, Mindlathe Ltd., Coventry, United Kingdom.

[25] Pensor. *Decision inertia an AI stability*, 2002, Pensor whitepaper series, wp-2002-4-2, Mindlathe Ltd., Coventry, United Kingdom.

[26] Karlsson, B. *Prototyping a Simple Layered Artificial Intelligence Engine for Computer Games*. In Proceedings of the 4th Intl. Conf. on Intelligent Games and Simulation (GAME-ON 2003), London, UK, November, 2003.

[27] Pontevia, P. and Williams, G. *AI Middleware, A Powerful New Multi-Platform Approach For Game Development: Introducing RenderWare AI, powered by Kynogon,* 2002, Whitepaper, Criterion Software Inc., Austin, USA.

[28] Dybsand, E., *A Finite-State Machine Class*, Game Programming Gems, Charles River Media, 2000.

[29] Mateas, M. and Stern, A., *Beyond Finite State Machines: Managing Complex, Intermixing Behavior Hierarchies*, In Proceedings of the Game Developers Conference2004, San Jose, USA, 2004.

[30] Christian, M. *A Simple Inference Engine for a Rule-Based Architecture*, in AI Game Programming Wisdom, 2002

[31] Maslow, A. *Personality and motivation*. Harper and Collins. 1970.