I AM AI: Interactive Actor Modeling for Introducing Artificial Intelligence: A Computer Science Capstone Project

Thomas Bowersock, Victoria Kerr, Yuki Matoba, Andrew Warren, Alexandra Coman
Department of Electrical and Computer Engineering and Computer Science
Ohio Northern University
Ada, OH 45810
Email: t-bowersock.3@onu.edu, v-kerr@onu.edu, y-matoba@onu.edu, a-warren.3@onu.edu, a-coman@onu.edu

ABSTRACT

Our Senior Capstone project addresses the following challenges: (1) providing an extensive introduction to Artificial Intelligence (AI), via research and implementation, to a group of senior undergraduate students with no prior experience in the field, (2) incorporating hands-on learning of the highly-popular, commercial-level Unreal Engine development framework, (3) focusing on game-character-behavior modeling, an application of Artificial Intelligence that is current, challenging, and of considerable practical and intellectual interest, and (4) allowing students to gain experience in thinking pedagogically by designing software with educational purposes.

The student team has conducted research on AI techniques and is using Unreal Engine to implement a character-driven, non-combat-centric game-like environment populated by multiple NPCs, the behavior of which is modeled using different AI techniques. The system is being designed so as to be attractive to a varied target audience. As we strongly wish to avoid alienating potential users who do not identify as avid "gamers", we are opting for a non-combat-centric game-like environment.

INTRODUCTION

Artificial Intelligence (AI) is the academic field of study concerned with establishing the theoretical foundations and driving the development of synthetic systems that are or appear to be intelligent, whether their intelligence is human-like and general or highly specialized.

Nowadays, the reach of AI can be seen in most aspects of our lives. Its branches have extended into the medical field, where AI can be used, among others, for the monitoring of patients; in the transportation industry, with the development of smart cars; in the entertainment industry, with automated creation of music; and in the writing field, where AI "journalists" write stories about sporting events.

AI is also a constant presence in computer-game development. The general goal of the developers of game AI systems is to create believable and compelling non-player character (NPC) entities that, in the case of adversarial games, put up a solid fight against the player, while still allowing them the ability to win the game; otherwise, the entertainment purpose would be defeated. In non-adversarial gaming environments, AI techniques can be used to model the behavior and inner structure of interactive characters by emulating various aspects of the human psyche and personality: emotional states, memory, morals, social affinities, etc.

In addition, games, while seemingly for entertainment purposes only, provide an excellent test environment for different AI techniques prior to them being implemented in "real-life" environments. Current computer games are often relatively complex and allow researchers to test their ideas with little overhead cost and risk.

Our Senior Capstone student team consists of students who, at the beginning of the project, had no prior course or research experience in the field of AI. The project activities can be grouped into two general phases: (1) the research phase and (2) the design and implementation phase.

In the research phase, the students conducted research into AI techniques, with a particular focus on AI techniques for games and interactive storytelling.

As part of the implementation phase, the student team is using Unreal Engine 4 (Epic Games), a commercial-level game engine which has been used in the development of numerous acclaimed and popular computer games, to implement a character-driven, non-combat-centric game-like environment populated by multiple NPCs, the behavior of which is modeled using different AI techniques.

The AI techniques to be implemented were chosen both for their amply-demonstrated practical applications and because they lend themselves particularly well to analogy with human cognitive processes. The system is being designed so as to be attractive to a varied target audience. As we strongly wish to avoid alienating potential users who do not identify as avid "gamers", we are opting for a non-combat-centric game-like environment.

After comparative evaluation of various programming languages and other software-development tools, we have chosen to implement our game-like environment using Unreal Engine 4 by Epic Games.

In the following sections, we provide a survey of related work, including background on the AI techniques we are implementing; describe our implementation process, and end with conclusions.

LITERATURE SURVEY

In this section, we explore areas of AI research that are relevant to our topics of interest: gaming, storytelling, character-behavior modeling, and education. We also provide background on two AI techniques that we are implementing: Case-based Reasoning and Reinforcement Learning, both of which have been demonstrated in gaming contexts.

Table 1. Game AI techniques as described by Sweetser and Wiles (2002)[28]

Technique	Description
Finite State Machines	- Commonly-used approach to modeling NPC behavior - Simple FSMs do not provide NPCs with complex inner structure, explicit goals, memory, or learning capabilities
Agents	- The agent approach gives characters greater autonomy and flexibility - Agents are capable of perceiving their environment, making decisions, and performing tasks in order to achieve a goal - There are many different types of intelligent agents, incorporating various AI techniques and approaches
Fuzzy Logic	- Can be used to make decisions based on incomplete information, much like humans often do - One can create AI NPCs that can make decisions and adjust their behavior based on fuzzy logic
Flocking	- Generating behavior for a large group of NPCs, such as a herd of cows, so that behavior for such NPCs does not have to be individually scripted
Decision Trees	- Can, for example, help characters determine what the course of action most likely to lead to the best result is, over a number of possible game-play situations
Genetic Algorithms	- Basic strategies are combined and modified over time, through a process inspired by Genetics, in order to "evolve" better strategies
Extensible AI	- Allows players of the game to create or modify game AI - Provides a flexible game experience

Sweetser and Wiles (2002)[28] describe multiple AI techniques that can be used in game contexts. Finite State Machines, Agents, Fuzzy Logic, Flocking, Decision Trees, Genetic Algorithms, Extensible AI, and combinations of the above are among the techniques discussed (Table 1).

An advanced game AI technique is showcased in the critically-acclaimed computer game F.E.A.R (Monolith Productions, 2005), which implements a type of AI Planning. AI Planning systems generate plans for reaching given goals by putting together a sequence of actions, out of a set of available actions. Action choices are made based on the action pre-conditions and post-conditions, and on heuristic evaluations (Orkin, 2006) [21].

Complex types of AI Planning used in games include HTN (Hierarchical Task Network) Planning and Belief-Space Planning.

Davis and Gorniak (2007)[7] use HTN Planning, a type of planning which involves decomposing compound tasks into primitive tasks, to create collaborative plans to be executed by groups of NPCs. Planning is conducted in real-time, during game execution. This technique allows a group of NPCs to work together, sharing knowledge similarly to the way a human team would, in order to accomplish a goal.

Macindoe et al. (2012)[16] present a type of Belief-Space Planning, partially observable Monte-Carlo cooperative planning (POMCoP), a method for planning under conditions of uncertainty and incomplete information about the environment. It is used to help enable NPC sidekicks to determine what actions can best help the player to achieve their goals in the game.

When it comes to AI systems for generating stories, one particularly influential example is that of the TALE-SPIN (Meehan, 1977)[18] program. TALE-SPIN is a story-generation program that creates stories by generating "rational behavior" of the story characters. This "rational behavior" is simulated by taking into account knowledge about things including the personality traits and needs of characters, and the relationships between characters. Cunningham and Fairclough (2002)[6] present an "interactive story engine", the story world of which is an interactive environment populated by multiple NPCs.

Case-Based Reasoning

Case-Based Reasoning (CBR), a seminal introduction to which is provided by Aamodt and Plaza (1994)[1] is an AI problem-solving approach consisting of creating solutions to newly-encountered problems by adapting solutions that have been used in the past to solve similar problems. The roots of CBR are in Psychology and Cognitive Science: it is based on problem-solving approaches commonly used by humans.

The **CBR cycle** generally consists of a series of four steps (one or more of which are sometimes omitted): **Retrieve**, **Reuse**, **Revise**, and **Retain**.

- The **Retrieve** step involves going back into "memory" to find scenarios that are similar to the current one. This "memory" is actually represented as a "case base", each component "case" of which will, in the simplest form, contain a previously-solved problem and its solution. More complex case structures are also possible. Cases are selected from the case base using a criterion (simple or composite) that is usually (but not always) a similarity metric.

- **Reusing** consists of taking the retrieved similar scenario(s) and applying its/their associated solution(s) to the current problem. Adaptation will be conducted if necessary.

- If inadequacies are found while evaluating the retrieved (and, possibly, adapted) solutions, the solutions are **revised**.

- The **Retain** step consists of storing the final solution, along with the problem it solves, back in the case base.

HaCohen-Kerner (1995)[10] shows how the principles of CBR are applicable to the game of chess. In another "real-life" game example, CBR is used by Ros et al. (2006)[24] and Lopez et al. (2007)[13] in the robotic-soccer domain.

Off the field and into virtual space, Aha, Molineaux, and Ponsen (2004)[2] explore the use of CBR-based plan retrieval in a real-time strategy game. Further exploration into the use of CBR in real-time strategy games can be seen in the work of Palma et al. (2011)[22], and Sánchez-Pelegrín, Gómez-Martín, and Díaz-Agudo (2005)[25]. Louis and Miles (2005)[15] explore the idea of combining genetic algorithms with CBR in the context of strategy games.

The use of CBR in gaming is not limited to strategy games, however. CBR has also been used in a text-based adventure game to dynamically tailor aspects of the game to the current player, based on a technique called "player modeling" (Sharma et al., 2007)[26].

Outside of the realm of gaming, CBR has also been applied to the generation of stories. CBR for story-generating is, for example, explored by Ontañón et al. (2012)[20] in their paper on the system GENA (Generation of Audio-Visual NArative), which generates narratives based on annotated content.

CBR is also being used extensively in numerous other fields, including the medical field (Montani at al., 2013)[19] and computer-synthesized music (Lopez et al., 2005)[14].

Reinforcement Learning

Reinforcement Learning (RL), a type of machine learning, is another branch of AI with demonstrated uses in game contexts. RL has two main characteristics: (1) agents receive positive and negative rewards for their actions, and (2) agents act so as to maximize long-term, cumulative reward.

Amato and Shani (2010)[3] demonstrate reinforcement learning in strategy games using the game Civilization IV, a "very large and complex" strategy game, as a test-bed. The reinforcement learning approaches used in their work are Q-learning and model-based Q-learning. A variant of Q-learning, called Frequency Adjusted Q-learning (FAQ-learning), is presented by Kaisers and Tuyls (2011)[12].

Gusmao and Raiko (2011)[9] introduce a RL algorithm called Exlos, which is tailored specifically to the large and complex environments of real-time strategy games.

Artificial Intelligence and Education

There are two different broad areas of AI in education: (1) using AI to teach, and (2) teaching AI. Our project is intended to cover both of these aspects.

Bernardini, Porayska-Pomsta, and Sampath (2013)[4] describe an interactive intelligent agent intended to help children with autism develop their social skills. The system uses AI Planning techniques.

Gómez-Martín, Gómez-Martín, and González-Calero (2005)[8] describe a game-based tutoring system created using CBR techniques. They point out that potentially "difficult" and "tedious" tasks "can be engaging and fun when they are part of a good story".

Redmond and Phillips (1997)[23] present a case-based tutoring system called CECELIA based on the learning approach of having students observe iterative problem-solving processes conducted by experts and attempt to predict and explain the experts' actions.

Further exploration into the possibilities of teaching through AI is presented by Cirillo, Micarelli, and Papagni (1997)[5]: a CBR-based system for authoring training applications and other educational applications. The system allows training-session personalization, so as to better suit each individual user.

At the other end of the spectrum, an example of a project using AI to teach AI is Jim (Heilemann, Mrowczynski, and Selkowitz, 2014 [11]), a student-designed robot puppet created using a Lego mindstorm kit. The project allows AI topics to be taught in an accessible manner.

IMPLEMENTATION

Selecting Software-Development Tools

The first step towards implementation was selecting, based on evaluation of various options, (1) a suitable programming language for implementing the underlying AI techniques of the NPCs, and (2) tools for developing the graphical user interface (GUI).

For the former purpose, we examined the differences between general-purpose languages, such as Java, C#, and C++, and the special-purpose language for creating interactive-story-character behavior ABL.

ABL ("A Behavior Language", Mateas and Stern, 2002)[17] is a programming framework for designing agents to act as NPCs in games referred to by the authors as "interactive, dramatic story [worlds]". ABL allows one to define, among others, goals, subgoals, and behavior for characters, actions that can be taken in a given game world, sensors for capturing information from the environment, and joint behavior allowing multiple character agents to coordinate.

One of the advantages of general-purpose languages is that they are not tailored toward a specific type of AI, so they are less limiting. They also have the advantage of familiarity, as all our team members have used them. On the downside, however, general-purpose languages may require us to recreate game/AI-specific functionality that is readily available in special-purpose languages.

Table 2. Tools for Creating the GUI

Programming Language/Development Tool	Pros	Cons
Scratch (Programming Language)	- easy to get started - good for creating simple games	- too simplistic to be usable for creating a game with complex AI
OpenGL (API)	- supported by multiple operating systems - usable with multiple programming languages - widely used, hence there are many educational resources	- learning curve involved

Programming Language/Development Tool	Pros	Cons
Unreal Engine (Game Engine)	- used for the development of numerous commercial-level games - cross-platform - advanced 3D graphical capabilities	- highly complex, thus challenging to learn

Table 3. Programming Languages for Implementing AI Techniques

Programming Language/API	Pros	Cons
General Purpose (Java, C#, C++)	- less limiting, not tailored toward a specific form of AI - team is familiar with these programming languages - expansive collection of plug-ins and libraries	- effort required to recreate specific functionality readily available in special-purpose languages
Special Purpose (ABL)	- built-in agent functionality - tailored specifically towards at least one character-modeling AI technique	- limited to certain types of built-in AI - lower availability of Internet resources and support network

Special-purpose languages such as ABL have their own sets of advantages and disadvantages. On the plus side, ABL has built-in agent functionality. It is tailored to at least one set of AI techniques, but, unfortunately, this may mean that it lacks the flexibility required to handle all techniques that we plan to implement. Another disadvantage is that Internet resources and support networks are not as readily available as for general-purpose languages.

Tables 2 and 3 summarize the advantages and disadvantages that were identified for tools that could be used for creating the graphical user interface (Table 2) and for programming languages that could be used to implement the underlying AI techniques (Table 3).

Our final decision was to use Unreal Engine 4 (Epic Games), a cross-platform, commercial-level game engine that is free to use for educational purposes. The engine offers complex three-

dimensional graphics and can be used in conjunction with C++, with which our entire team is familiar. In addition, since it is a game engine, it is ideal for the type of interactive environment that we are creating.

Implementation of Artificial Intelligence Techniques

Prior to choosing Unreal Engine 4, in order to familiarize ourselves with varied AI techniques, we implemented a CBR algorithm and an instance of A* search in a general-purpose programming language.

The implemented CBR algorithm is a variant of Bounded Greedy (Smyth and McClave, 2001)[27]. Bounded Greedy is an algorithm that was introduced in the context of CBR recommendation systems, which recommend products and services to users based on the users' previous purchases, ratings, social-media information, etc. Bounded Greedy addresses the need to generate recommendations that not only match user preferences but are also diverse among themselves, so as to represent genuine choices.

A* search is a heuristic-search algorithm used extensively in game AI, especially for path-finding tasks. It can also be incorporated in AI Planning (Orkin, 2006)[21].

Next, we describe the algorithms implemented in Unreal Engine 4.

Development in Unreal 4 was conducted using a combination of C++ code and the Unreal 4 "Blueprint" visual scripting system.

The first type of AI that we are showcasing in Unreal Engine is a Finite-State Machine (FSM), a commonly-used mechanism for automating non-player characters in games. A character based on an FSM is, at any given time, in a specific state. When a given event occurs, the character transitions from its current state to another state. Unreal Engine provides tools for building FSMs for modeling NPC behavior.

An FSM modeling the behavior of an NPC in a combat-based game might indicate that the NPC should attack when an adversary is seen and patrol a defended location once enemies are no longer in sight. The two states would be "attacking" and "patrolling". Transitions to these states would be triggered by an enemy being sighted and by the enemy no longer being in sight.

The FSM that was implemented for our project does not provide combat-related behavior: instead, it simulates a character reacting to user interaction by displaying various emotions: calm, happiness, and anger. Input received from the user signifies the type of intended interaction: flattery, insult, or stating that the flattery/insult was a joke. The input causes the character to

"respond" by transitioning to a state corresponding to a new "emotion" or remaining in the same state.

The inner workings of the FSM are made visible on the screen so as to show how the underlying AI "clockwork" works: in addition to seeing a display of emotion from the user, the user also sees, on screen, the underlying FSM state transition that was conducted on receipt of user input.

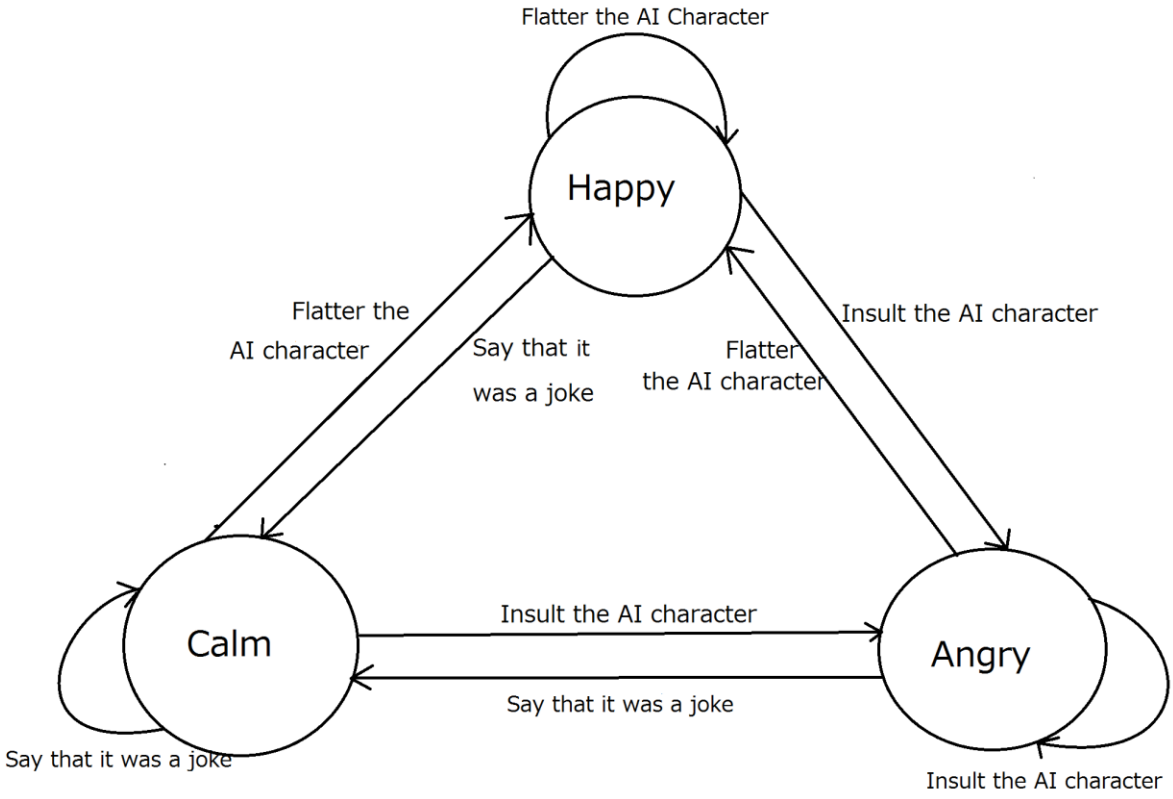

Fig. 1. Visual Representation of the Implemented Finite State Machine

After having previously been implemented in a general-purpose programming language, a variant of the Bounded Greedy algorithm was incorporated into the Unreal Engine implementation.

The Bounded-Greedy-based recommender system we have implemented is intended to simulate interaction with a waiter who recommends a dish based on the taste preference expressed by the user.

As mentioned before, the key characteristic of Bounded Greedy is that it balances similarity to the user's preference (the "customer" should, after all, receive a dish that they like, based on their expressed taste preferences) with diversity of provided options (a selection of choices that are too similar to one another would likely not be particularly exciting for the user).

The case base contains 15 dishes with recorded taste properties. Each dish has a different degree of saltiness, sweetness, bitterness, etc. Each food is defined as a combination of tastes, each taste with an associated degree.

The pseudocode and description of the Bounded Greedy Algorithm are provided by Smyth and McClave (2001)[27].

Here are several implementation details of the variant of Bounded Greedy implemented in our system:

- A set of 5 dishes is retrieved based on both similarity and diversity.
- Similarity and diversity are computed based on the taste properties of dishes.
- The retrieval criterion is the weighted sum of the similarity to the preference expressed by the user and the assessed diversity of the set that would be obtained by adding the current candidate choice to the set of previously-retrieved choices (Equation (5) of Smyth and McClave (2001)[27]).
- The weight α is set to 0.7. Therefore, the selection criterion consists 70% of similarity and 30% of diversity. These weight values were chosen because, while we want to balance between the two, in such a context, similarity is more important than diversity: we do want to cater to the user's expressed taste preference, and too much variation might defeat that purpose. Still, the user is provided with diverse choices to a certain extent.

We are also currently implementing a reinforcement learning algorithm to allow an AI character to learn how to navigate a maze.

CONCLUSIONS

Our Senior Capstone student team consists of students who, at the beginning of the project, had no prior course or research experience in the field of AI.

The objective was for the students to be provided with an extensive introduction to Artificial Intelligence (AI), via research and implementation, including hands-on learning of the Unreal

Engine 4 development framework. The focus is on the game-character-behavior-modeling application of Artificial Intelligence.

The project activities can be grouped into two general phases: (1) the research phase and (2) the design and implementation phase.

In the research phase, students conducted research into AI techniques, with a particular focus on techniques for games and interactive storytelling.

As part of the implementation phase, the student team is using Unreal Engine 4 to implement a game-like environment populated by multiple non-player characters. The behavior of each character is to be modeled using a different Artificial Intelligence technique. The AI techniques to be showcased are finite state machines, case-based reasoning, and reinforcement learning.

Bibliography

1. Aamodt, Agnar, and Plaza, Enric. "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches". Artificial Intelligence Communications 7, no. 1 (1994): 39-52.
2. Aha, David, Marc Ponsen and Matthew Molineaux. "Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game." Springer (2004).
3. Amato, Christopher and Guy Shani. "High-level Reinforcement Learning in Strategy Games." AAMAS (2010): 75-82.
4. Bernardini, Sara, Kaska Porayska-Pomsta and Harini Sampath. "Designing an Intelligent Virtual Agent for Social Communication in Autism." AAAI (2013).
5. Cirillo, Vincenzo, Marco Papagni and Alessandro Micare. "Ocram-CBR: A Shell for Case-Based Educational Systems." Springer (1997).
6. Cunningham, Pádraig and Chris Fairclough. "An interactive story engine." Springer (2002): 171-176.
7. Davis, Ian and Peter Gorniak. "SquadSmart: Hierarchical Planning and Coordinated Plan Execution for Squads of Characters" AAAI (2007).
8. Gómez-Martín, Marco Antonio, Gómez-Martín, Pedro Pablo, and González-Calero, Pedro A.. "Game-Based Learning as a New Domain for Case-Based Reasoning." Universidad Complutense de Madrid (2005): 175-184.
9. Gusmao, Antonio and Tapani Raiko. "Reinforcement Learning In Real-Time Strategy Games." Aalto (2011).
10. HaCohen-Kerner, Yaakov. "Learning strategies for explanation patterns: Basic game patterns with application to chess." Springer (1995).
11. Heilemann, Michael, Robert Selkowitz and Jon Mrowezynski. "Jim: A Platform for Affective AI in an Interdisciplinary Setting" AAAI (2014).
12. Kaisers, Michael and Karl Tuyls. "FAQ-Learning in Matrix Games: Demonstrating Convergence Near Nash Equilibria, and Bifurcation of Attractors in the Battle of Sexes." AAAI (2011).
13. Lopez, Ramon, et al. "Team Playing Behavior in Robot Soccer: A Case-Based Reasoning Approach." Springer (2007).
14. Lopez, Ramon, Petra Perner and Padraig Cunningham. "Emergent Case-Based Reasoning Applications." Cambridge Univ Press (2005).

15. Louis, Sushil J. and Miles, Chris “Combining Case-Based Memory with Genetic Algorithm Search for Competent Game AI”. Springer (2005)
16. Macindoe, Owen, Leslie Pack Kaelbling and Tomas Lozano-Perez Lozano-Perez. "Belief Space Planning for Sidekicks in Cooperative Games." POMCoP (2012).
17. Mateas, Michael and Stern, Andrew. “A Behavior Language for Story-based Believable Agents”. AAAI Press (2002).
18. Meehan, James R. “TALE-SPIN – An Interactive Program That Writes Stories”. IJCAI (1977).
19. Montani, Stefania, et al. "Mining and Retrieving Medical Processes to Assess the Quality of Care." Springer (2013): 233-240.
20. Ontañón, Santiago, et al. "GENA." A Case-Based Approach to the Generation of Audio-Visual Narratives (2012): 297-311.
21. Orkin, Jeff. "Three States and a Plan: The A.I. of F.E.A.R." Monolith Productions (2006).
22. Palma, Ricardo, et al. "Combining Expert Knowledge and Learning from Demonstration in Real-Time Strategy Games." Springer (2011).
23. Redmond, Michael and Susan Phillips. "Encouraging Self-Explanation Through Case-Based Tutoring: A Case Study." Springer (1997).
24. Ros, R., Veloso, M.M., de Mantaras, R.L., Sierra, C., Arcos, J.L. “Retrieving and Reusing Game Plays for Robot Soccer”. Springer (2006).
25. Sánchez-Pelegrín, Rubén Marco, Gómez-Martín, Antonio, Díaz-Agudo, Belén: “A CBR Module for a Strategy Videogame”. Springer (2005).
26. Sharma, Manu, Ontañón, Santiago, Mehta, Mehta, and Ram, Ashwin. “Drama Management and Player Modeling for Interactive Fiction Games”. Computational Intelligence Journal (2007).
27. Smyth B., and McClave, P. “Similarity vs. Diversity”. Springer (2001).
28. Sweetser, P. and J. Wiles. "Current AI in Games." A Review. Australian Journal of Intelligent Information Processing Systems 8.1 (2002): 24-42.