Player Modeling: What is it? How to do it?

Marlos C. Machado and Eduardo P. C. Fantini and Luiz Chaimowicz
Department of Computer Science
Federal University of Minas Gerais
Belo Horizonte, Brazil

Figure 1: *Screenshots of Civilization IV gameplay, used for some experiments in Player Modeling.*

Abstract

Player Modeling is currently a very relevant topic in game AI research. This tutorial presents the main topics related to the field, intending to be a summary and a guide of some of the most relevant works on this subject. This text starts discussing the importance of Player Modeling and its applicability. We then define and present a taxonomy of the field and discuss several application examples and common techniques used in the area. Since we intend to help a new researcher in the field to start its own work, we also list several game platforms that are suitable for experimentation and present a simple example that discusses important research topics.

Tutorial Level: Basic

Duration: 3 hours

Topics of Interests: Artificial Intelligence, Human-Computer Interaction, Development Process and Tools.

Keywords:: Player Modeling, Taxonomy, Game Platforms, Civilization IV

Author's Contact:

{marlos, fantini, chaimo}@dcc.ufmg.br

1 Introduction

The most important goal for almost every existing game is entertainment [Nareyek 2004] and this is achieved with a combination of items like graphics, story (in some game genres) and artificial intelligence (AI). All these topics can be summarized in a unique concept: immersion, that has already been suggested as a general measure of entertainment by [Manovich 2002] and [Taylor 2002].

Bakkes et al. [2009] define immersion as "the state of consciousness where an immersant's awareness of physical self is diminished or lost by being surrounded in an engrossing, often artificial environment [Nechvatal 1999]" and affirm that it is achieved with aesthetic elements like graphics and audio and held by consistency during the game.

The aesthetic elements were the industry's main concern for a long time since they are more attractive (its easier to convince a customer to buy a game because it "looks good" than because it has a complex AI that is much harder to be presented in marketing events) but this scenario is starting to change. Game AI is receiving increasing attention since it is helps keeping the player attached to the game.

Among the several AI techniques, *Player Modeling* is gaining importance as an alternative for making games more adaptable and, consequently, more challenging. Basically, Player Modeling consists in modeling the player characteristics and behaviors during the game in order to improve some game aspect. A more formal definition, discussed in [Machado et al. 2011a] and [van den Herik et al. 2005], considers Player Modeling "as an abstract description of the current state of a player at a moment. This description can be done in several ways like satisfaction, knowledge, position and strategy".

The above definition makes clear that a player model describes a game aspect. Satisfaction, knowledge, position, etc are some of the many levels of abstraction in a Player Modeling task. In the next section we are going to discuss these different abstraction levels aided by a taxonomy proposed by [Machado et al. 2011a].

This documents is organized as follows: in the next section, we discuss a Player Modeling taxonomy, presenting its main abstraction levels. Subsequently we list several works which applied Player Modeling. We discuss these works as examples of possible applications in different parts of games and instantiations of the discussed taxonomy. In Section 4 we present some techniques that are generally used to implement the modeling process, most of them related to machine learning. We also discuss, in Section 5, several game platforms that are suitable for experimentation in Player Modeling and game AI in general. We present some research possibilities in Section 6 and conclude this tutorial in Section 7 briefly discussing a simple example of Player Modeling, focusing on data extraction and analysis.

2 Player Modeling Taxonomy

Two different taxonomies have been independently developed almost at the same time: [Machado et al. 2011a] and [Smith et al. 2011a]. We can interpret this parallel development as an evidence of the necessity of an organization of the field after several published works spread in the literature.

These two taxonomies are organized in different ways. The one presented in [Smith et al. 2011a] and [Smith et al. 2011b] is an inclusive taxonomy that divides Player Modeling in four *facets*: "the *scope* of application, the *purpose* of use, the *domain* of modeled details, and the *source* of a model's derivation or motivation". It has a high level approach for classifying each research in these *facets*. On the other hand, the taxonomy proposed in [Machado et al. 2011a] is more specific, presenting more "dimensions" ("facets") than [Smith et al. 2011a]. In this tutorial we will focus on this last one.

The taxonomy presented in [Machado et al. 2011a] is summarized in Table 1. We are going to use it as guide for this presentation because it presents several levels of abstraction that, once understood, make a comprehension on the field easier.

This taxonomy can be explained through a set of questions related

Description	Categories	Goals	Applications	Methods	Implementation
Knowledge	Online Tracking	Collaboration	Speculation in Search	Action Modeling	Explicit
Position	Online Strategy Recognition	Adversarial	Tutoring	Preference Modeling	Implicit
Strategy	Off-line Review	Storytelling	Training	Position Modeling	
Satisfaction			Substitution	Knowledge Modeling	

Table 1: *Player Modeling Taxonomy Summary.*

to a research in Player Modeling. We are going use these questions as guideline for new researchers since answering them implies in an advance in their own project.

1. *What are you intending to generate with Player Modeling? What is your main goal?*

   This first question is related to the Player Modeling goal. Three different goals are presented in this taxonomy and encompass all possibilities: Collaboration, Adversarial and Storytelling. The first one is responsible for generating an AI that cooperates with the players, it can be an ally non-player character (NPC) or a helper, for example. The adversarial goal intends to generate a more challenging game. This can be done with NPCs or specific challenges, for example. Finally, the storytelling goal is related to generating characters that help the plot unfold, not necessarily playing for or against the player.

2. *What do you want to describe with your model?*

   Once we defined the main goal of our modeling we may define what we are going to describe with it. This decision is based on the already defined goals. The description possibilities are many but four main topics are: Knowledge, Position, Strategy and Satisfaction. These are different abstraction levels but all of them are applicable to different game genres.

   Knowledge and Satisfaction are the two items that represent a higher level description. The first one can be seen as the attempt to model what the player knows, *e.g.* a researched technology in a strategy game or how far it is from solving a puzzle, while the satisfaction description may try to maximize player entertainment, understanding which game characteristics make him happy and which bother him. A knowledge description is presented in [Cunha and Chaimowicz 2010] for example and satisfaction description is shown in [Pedersen et al. 2010].

   A description in a lower abstraction level is directly related to the players' behavior once we may want to be able to define his movement patterns or even his strategies during a match, examples are: [Hladky and Bulitko 2008] and [Spronck and den Teuling 2010], respectively.

3. *What gameplay activity are you going to improve with your model?*

   After defining your general goal and what you are intending to describe, it is necessary to precisely define the game activity that is going to be improved. While the previous discussions were general, this one is specific. At this moment there are several game possibilities and listing all of them would be impossible. We will only discuss the main four applications that are presented in Table 1.

   A first application listed in Table 1 is related to the first game AI applications, mainly concerned with state-space search. Speculation in search consists in focusing on some parts of the game search tree and leaving others that your model predicts that are not going to be explored by the player. This is very useful in situations where search is costly, even with pruning techniques and an example is [Carmel et al. 1993], that applied these techniques in chess, while it still was an intractable problem.

   Tutoring and Training are related to the game intention of improving the player ability. Tutoring intends to teach players (focusing on player preferences) and an example is [Iida and Kotani 1998]. Training intends to present challenges suited to each player, *e.g.* related to his weakness, style or strategy. Finally, substitution consists in replacing human players by the AI NPCs without other players noticing. Certainly this is the hardest task since it is related to a Turing's Test for games.

4. *What models are you intending to use to generate an understanding about the agent?*

   To describe the agents characteristics, it is necessary to observe some data to generate specific models. We need to base ourselves in something to generate these descriptions and this is the main objective of this taxonomy category. A useful method to understand this category is to use Table 1 to complete the phrase: "based on its [*methods*] we may describe its [*description*]". For example: Based on its *actions* we may describe its *knowledge*.

   We listed some common inputs that are useful to generate models. Observing players actions is the most common activity [Spronck and den Teuling 2010] but we can also make questionnaires trying to obtain the player preferences [Pedersen et al. 2010], observe his positions to model his movement [Hladky and Bulitko 2008] (this is a harder task in environments with partially observable environments) or his knowledge.

5. *When are you going to process the data? Do you have enough time for doing this online?*

   After all the previous questions we have a very good idea of our intentions and now we are able to start thinking in a more practical way. Based on this, our first concern must be with the data collection and processing. We can do these during or after the game.

   The moment data is collected is not necessarily important since it is generally a light operation, but the moment the data is processed is important. If we process data and try to predict actions all the time, this is called *online tracking*. If we process the data in batches during the game, observing a set of actions we are doing an *online strategy recognition*. Finally, if we process the whole game data afterwards we will be doing an *off-line review*.

   It is necessary to assure that the platform being used to validate the research allows the desired processing and Section 5 discusses this. Examples of online tracking can be found in [Houlette 2003], of online strategy recognition in [Laviers et al. 2009] and [Ponsen et al. 2010] and of off-line review in [Spronck and den Teuling 2010] and [Pedersen et al. 2010].

6. *What is the interface between your algorithms and the game in which your model is going to be used?*

   Finally, it is important to check what are the platforms that will be used to validate our model and how we are going to do it. If its source code is available we may do it *explicitly*, if it does not, if the platform only offers us scripting languages or other interfaces we would be doing it *implicitly*.

   This last question involves much more than scientific knowledge: it demands also a knowledge related to the suitable game platforms for experimentation. We discuss this on the Section 5. *Civilization IV*, for example, allows *implicit* modeling [Pedersen et al. 2010] while *Infinite Mario Bros* allows *explicit* [Pedersen et al. 2010].

A concern that is not encompassed in the above questions is about the techniques that may be used. Obviously this shall also be studied and decided. We discuss some AI techniques in Section 4.

We firmly believe that a successful research in this topic shall be able to easily answer these questions. We are able to exemplify this answering process with many published papers. We will use [Spronck and den Teuling 2010] as example. In [Spronck and den Teuling 2010] several *Civilization IV* matches are analyzed with machine learning algorithms trying to model a human player in terms of his strategy during the game, *i.e.*, his preferences.

We may answer all the questions we previously formulated: the authors intended to generate models that would benefit player opponents (*adversarial goal*) and they wanted to describe the player *strategy* to improve the opponent's AI (*application*). They based their description in the player *preferences* and analyzed the data after complete matches, characterizing an *off-line review*. The data collection was done with scripts running with the game *Civilization IV*, so their implementation was an *implicit* one.

3 Application of Player Modeling

As we previously discussed, Player Modeling can improve several game aspects. In this section we present works that improved different game characteristics showing the applicability of Player Modeling.

3.1 Game Design

The main idea of Player Modeling related to *Game Design* is to generate environments that are best suited to each player, who is unique and has unique requirements.

An interesting work that used Player Modeling to assist game designers is [Drachen et al. 2009]. In this paper, authors used tools to extract gameplay information (*implicit* implementation) from the game *Tomb Raider: Underworld* and processed these data with neural networks (*off-line review*) to obtain playing styles (*Pacifist, Runner, Veteran* and *Solver*). During this process the authors extracted several information to make this classification, like *cause of death, completion time* and *number of deaths*. This information is applicable to the optimization of game design features [Drachen et al. 2009].

Another recent work that studied game design improvement based on adaptive analysis is [Pedersen et al. 2010]. They looked for correlations between players satisfaction and level characteristics of the game *Infinite Mario Bros*, such as *presence of gaps, blocks* and *enemies*. This allowed the authors to maximize player satisfaction. The implementation was *explicit* and it was an *off-line review*.

3.2 Interactive Storytelling

Interactive Storytelling is "a story-based experience in which the sequence of events that unfolds is determined while the player plays" [Thue et al. 2007]. This feature explicitly demands adaptive behavior since it is interactive, but not necessarily implements a personalization for player preferences.

Thue et al. [2007] proposed PaSSAGE, a method that models interactive storytelling as a decision process that is influenced by different *weights* that characterize each player. The authors define their own method as "an interactive storytelling system that uses player modeling to automatically learn a model of the player's preferred style of play, and then uses that model to dynamically select the content of an interactive story" [Thue et al. 2007]. The used platform was *Neverwinter Nights*. Besides the applicability of Player Modeling in *Interactive Storytelling*, [Thue et al. 2007] affirmed that there were not many works with this combination and this scenario still holds nowadays.

3.3 Game Artificial Intelligence

This is a very general topic but we intend to discuss here the application of improving opponents' AI during a game. The great majority of works related to Player Modeling are in this topic. One of them is [Charles and Black 2004] that proposed a framework for player-centered games and [Ponsen et al. 2010] that created a Poker player with Monte-Carlo Tree Search and used Player Modeling to focus on relevant parts of the tree.

A very important paper is [Houlette 2003], one of the first works to really describe a Player Modeling approach in digital games that were not board games. In this paper the author presents a general approach for player modeling, more specifically how to characterize player behaviors and implement one.

A more recent paper is [Laviers et al. 2009], that "evaluates the competitive advantage of executing a *play switch* based on the potential of other plays to increase the yardage gained and the similarity of the candidate plays to the current play" in the game *Rush 2008*.

3.4 Mimicking Human Players

Finally, this last topic, as we already said, is the most difficult since it implies computers passing on a simpler version of the Turing Test. As far as we know no paper was successful on this task.

An interesting approach was [Spronck and den Teuling 2010] and [den Teuling 2010] where the authors tried to fit humans in a group of NPCs, characterizing their preferences expecting that, after correctly classifying them, they would be able to select an appropriate preprogrammed behavior in the game *Civilization IV*.

Since this is a very hard task, there are not many works on this topic. Despite the very attractive application, to be able to replace human players by artificial agents, most of research is focused on improving game AI, not in mimicking human players. But we truly believe this is a very promising research topic.

4 Common Techniques

Several techniques may be used to generate player models and a large number of them are discussed in [van den Herik et al. 2005]. In this text we are going to focus our discussion in the techniques that were already used, listing the papers which used them.

4.1 Kernel Machines

This technique was used by [Spronck and den Teuling 2010] and [den Teuling 2010] to predict human player preferences based on several observations of NPCs playing. The authors tried to classify the player's preference using data extracted on each turn of the game *Civilization IV*. The authors did not use "pure" SVM[1] but an optimization called SMO – Sequential Minimal Optimization.

Laviers et al. [2009] used SVMs in the *Rush 2008* game to "recognize the defensive play as quickly as possible in order to maximize our [their] team's ability to intelligently respond with the best offense".

4.2 Evolutionary Algorithms

A recent work that used artificial evolution to achieve learning in the preference modeling task is [Martínez et al. 2010]. The technique that really did the classification was a neural network but artificial evolution was used to relate "the difference between the subject's reported affective preferences and the relative magnitude of the corresponding model (ANN) output" [Martínez et al. 2010]. The platform they used was developed by them and is called *MazeBall*. The combination of evolutionary algorithms and neural networks is not uncommon, *e.g.* [Pedersen et al. 2010] also used it.

Another recent work that applied these techniques to adaptive behavior though modeling players is [Tan et al. 2011] with ideas of evolutionary computation and reinforcement learning in a car racing game. The authors implemented several game strategies and encoded these strategies in *chromosomes* as binary numbers where each chromosome is responsible for its activation (1) or not (0). They also trained neural networks as a controller.

[1] Support Vector Machine

Togelius et al. [2007] also worked on a similar problem using "evolutionary algorithms to evolve racing tracks that maximized the entertainment value to particular human players" [Tan et al. 2011].

4.3 Neural Networks

Neural Networks and evolutionary algorithms are not an unusual combination. Tan et al. [2011] focused on evolutionary algorithms but they also trained neural networks as a controller for the car racing game to generate adaptive behavior. Martínez et al. [2010] also used neural networks, more specifically self-organization maps (SOM) and emergent SOMs (ESOM), to identify player types on a prey/predator game called *MazeBall*, while Drachen et al. [2009] used ESOM networks to cluster players intending to distinguish different behaviors in the game *Tomb Raider: Underworld*.

Neural networks applications different from clustering are prediction and classification, and the techniques used are also different. An example of prediction with neural networks is [Pedersen et al. 2010] who used multi-layer perceptrons to predict players' satisfaction in the game *Infinite Mario Bros*.

Clustering and prediction/classification are not exclusive approaches, Thurau et al. [2003] used multi-layer perceptrons and self organizing maps in the *Quake II* game, for example.

The application of these techniques to each research problem is not simple to be explained since it involves a specific modeling, we invite the interested readers to consult the original papers.

4.4 Overview

Research in game AI is a relatively new and multidisciplinary topic. Because of it several researchers who actually work on computer games had already worked on other fields and may apply unusual techniques to games. We have not covered all the most common AI techniques not discussing Hidden Markov Models or Evaluation Functions, neither papers with "innovative" techniques as the application of compiler principles to game AI [Orkin et al. 2010].

Based on this superficial covering and our previous experience we were able to observe that most learning techniques used in these researches are from natural computing, *i.e.* evolutionary algorithms and neural networks. We believe the main reason for this is the fact that these are older techniques, more widespread in the whole computing community.

5 Suitable Game Platforms

This section is derived from our previous work [Machado et al. 2011a] and is also available in our research group website[2]. In this previous work we listed several available game platforms and Player Modeling research opportunities in each game genre. We will present a summary of this discussion here.

There are three conventional ways to access game mechanics to use them as testbed platforms: open source game, game modification tools and game emulators. We discuss each one below.

- **Open Source Game** is a video game in which its source code is open source. They are often freely distributed and sometimes cross-platform compatible. As consequence, many are included in Linux distributions. Open source games which are free software and contain exclusively free content are called *free games*. Most free games are open source, but not all open source games are free software; some open source games contain proprietary non-free content. This class (open source game) generally generates explicit player models.
- **Game Mod** is a term applied to personal computer games and are more common in first-person shooters, role-playing games and real-time strategy games. Mods are not standalone software and require the original release in order to run them. They may be created by the general public or developers and offer limited control of the game, generally allowing the inclusion of new items, weapons, characters, enemies, artificial intelligence, models, textures, levels, story lines, music, game modes, among others. Mods that add new content to the underlying game are often called *partial conversions*, mods that create an entirely new game are called *total conversions* and mods that fix bugs are called *unofficial patches*. Because of their nature they are more common in commercial games. Nowadays we can find tools and documentation to assist mod makers for games developed by *Valve Software*, *id Software*, *Bethesda Softworks*, *Firaxis*, *Crytek*, *The Creative Assembly* and *Epic Games*. This approach generates implicit or explicit player models as it depends on the interface offered by the game.
- **Game Emulator** is a software that duplicates (or emulates) the functions of an original game in a second system, intending to generate a behavior that closely resembles the behavior of the original system. They are generally used to play older video games on personal computers or modern video game consoles, but they are also used to translate games into other languages, to modify existing games, and to develop home brew demos and new games for older systems. Generally this also generates an explicit player modeling.

We divided the games listed here by its genre, intending to generate a better organization.

5.1 Action Games

5.1.1 Valve Software SDKs

Valve Software is the developer with most support for game modifications, providing complete documentation and official tools. The Source Engine and its associated SDK provide one of the most efficient, complete, and powerful action game development package on the market. The games cited below are all first-person shooters.

The Source SDK [Valve d] includes: Half-Life 2 C++ source code, Softimage XSI—EXP (for model building), The Hammer (level editor), FacePoser (facial expressions and lip-syncing) and Half-Life Model Viewer. It allows directly edition of the following action/first-person shooter games: Half-Life: Source, Half-Life 2, Counter-Strike: Source, Day of Defeat: Source, Team Fortress 2 and Portal.

Alien Swarm [Valve a] has exclusive tools, called Alien Swarm Authoring Tools, that allow developers to create their own missions, weapons, enemies and other gameplay elements. This tool pack includes: Updated Hammer (level editor), suite of command line compiling utilities (such as studiomdl and map compiling tools), Particle editor, Faceposer (facial expressions and lip-syncing), Example campaign, Tutorial maps and C++ source code to the Alien Swarm client and server dlls.

Valve also made available a set of software utilities that allows developers to create their own levels, campaigns, common enemies, and other gameplay elements in the games Left 4 Dead [Valve b] and Left 4 Dead 2 [Valve c].

There is also a SDK for older Valve Games, like Half-Life and Counter-Strike, but no technical support is available.

5.1.2 Id Software open source games

Id Software is a traditional action/first-person shooter game developer that released the world famous Doom and Quake game series. Id Software does not provide support for game mods, but the complete C/C++ source-code of Doom, Quake, Quake 2, Quake 3, Quake Wars and Wolfstein 3D is available on the FTP link [id Software] of the company. This initiative eliminates problems like platform restrictions since we are able to completely alter the game AI by having the source code.

[2] http://www.j.dcc.ufmg.br/platforms.html

5.1.3 Unreal Tournament Game Bots

GameBots is an Unreal Tournament (first-person shooter game) modification that allows software agents to play the game. The Unreal Tournament game server feeds the player's sensory information to the agent and it feeds actions back to the game. A JavaBot API [JavaBot] can be used to provide a higher-level interface to Unreal protocol. This is clearly an *implicit* modeling possibility that not necessarily reduces the game usefulness.

5.2 Platform Games

5.2.1 Infinite Mario Bros

Infinite Mario Bros [Persson] is an open-source 2D game written in Java that mimics the classical action/platform game Super Mario Bros. It was made for programming contests, like Mario AI Championship [Championship]. A similar open-source project is Secret Maryo Chronicles [Secretmaryo.org], written in C++. Both allow complete game modification and the research possibilities are only restricted by the game genre, not by the game platform.

5.2.2 Open Sonic

Open Sonic [Martins] [Sonic] is an open-source 2D action/platform game which emulates and enhances the experience of the "Sonic the Hedgehog" universe. It introduces a different style of gameplay called cooperative play in which it's possible to simultaneously control three different characters.

Open Sonic provides a map editor to create new levels. New items, enemies and bosses can be created or modified by Object Scripts based on finite state machines. The game mechanics can be modified only by source code, written in C. This set of possibilities allows game modifications ranging from level design alterations through game AI.

5.3 Simulation Games

5.3.1 Flight Gear

Flight Gear [Solutions] is an open-source multi platform project which offers a sophisticated framework for realistic 3D flight simulations. It runs on Windows, Mac and Linux and was developed by skilled volunteers around the world. The source code in C++ is available and licensed under the GNU License. Customizations can be done by source code or 3rd party extensions.

5.3.2 The Open Racing Car Simulator (TORCS)

TORCS [Bernhard] is a multi platform 3D car racing simulation. It can be used as an ordinary car racing game, as an AI racing game or as a research platform. It runs on Linux, FreeBSD, MacOSX and Windows. Its source code is written in C/C++ and is licensed under the GPL License.

5.4 Sports Games

5.4.1 Super Tux Kart

Super Tux Kart [Jenkins] is a multi platform and open-source racing game, similar to Super Mario Kart. It runs under Linux and Windows. New vehicles configurations can be inserted from XML add-ons, new levels can be made in Blender and the AI customization can be done by source code modification, written in C++. As most of the previous discussed game this combination allows level design modifications, AI research and consequently, player modeling research.

5.5 Strategy Games

5.5.1 Civilization IV SDK

Civilization IV [Games] is a turn-based strategy game developed by Firaxis Games. The XML interface offers the possibility to configure several of its parameters. Another possibility is to use the Civ4 SDK, that allows developers to completely rewrite or modify (by C++ code) their game, recompiling DLLs [Firaxis] that replace the original one.

While the first possibility generates an *implicit* modeling the second one generates an *explicit* modeling. The XML interface does not offer AI programming possibilities but the other approaches do and, since Civilization IV is an extremely complex game, it is very attractive for research due to the simplicity of changing its AI.

5.5.2 Open Real-Time Strategy (ORTS)

ORTS [Buro] is a real-time strategy engine implemented in a client/server architecture with 2D or 3D graphics. It was created for the study of real-time AI problems such as path finding, imperfect information problems, scheduling and planning in the domain of RTS games. ORTS runs on Windows, Linux and Mac. The server side loads scene configurations and waits for clients connections. The units AI shall be written in C++ in the client side. This platform was used from 2006 to 2009 in the ORTS Game AI Competition promoted by the AIIDE Conference.

5.5.3 StarCraft: Brood War API

StarCraft Brood War is a real-time strategy game developed by Blizzard Entertainment. There is an unofficial C++ API for StarCraft, called Brood War API [bwapi], that allows AI modules creation which has been used in StarCraft AI Competitions[3] [4].

5.5.4 Wargus

Wargus [Pali b] is a cross-platform open source project which allows unofficial modifications of Warcraft II, a real-time strategy game released by Blizzard Entertainment. Wargus runs on Windows and Linux and is organized in two levels: the Stratagus Engine [Pali a] (an emulator of Warcraft II core engine) and a higher layer for game logics implemented in Lua Script [Rio].

5.6 Role-Playing Games

5.6.1 Baldur's Gate and Icewind Dale series

Baldur's Gate and Icewind Dale series are role-playing games released by BioWare. These games were developed with the Infinity Game Engine. There is an unofficial tool called Near Infinity [Hauglid] which combines browsing and editing to allow games modification based on BioWare's Infinity Engine. It supports both Baldur's Gate 1 & 2 and Icewind Dale 1 & 2. Another tool to modify Baldur's Gate is Weidu [Weimer].

5.6.2 Neverwinter Nights series

Neverwinter Nights series are role-playing games developed by BioWare. For game modding there is an official tool called Aurora Neverwinter Toolset that allows the modification of several game aspects. The tools include a visual tile-based terrain editor, a script editor (NWScripts [Nwn2Scripting] [Gaming a] [Gaming b] [BioWare]), a conversation editor and an object editor.

6 Research Possibilities

Game AI is a fantastic research field because it is the perfect platform to seek for an AI in a human level [Laird and Lent 2000].

[3] http://skat.dnsalias.net/mburo/sc2011/

[4] http://ls11-www.cs.uni-dortmund.de/rts-competition/starcraft-cig2011

Player modeling is an attractive sub area and presents several research possibilities – if we combine Description, Goals and Methods possibilities discussed in Section 2 we already generate more than forty possibilities of research in this subject, while [Smith et al. 2011b] mention 64 possibilities in their taxonomy, evidencing how promising is this field.

We have discussed several research possibilities during this text but we will condensate it on this section. An interesting and immediately applicable-to-industry research topic is the personalized procedural generation of scenes, defining types, quantities and positions of units in a terrain.

In a lower level it would be interesting to generate artificial agents that "mimics" the player's gameplay. This activity requires a gameplay identification (with, at least, pattern recognition techniques). Reducing game granularity we could also look for an agent's action predictor, ranging from immediate actions to movement or strategy.

It is certainly impossible to discuss all research topics, even topics we already mentioned during the text as interactive storytelling and opponents AI improvement could be better explored.

Player Modeling is not restricted to general computer games as RTS or FPS, but could also be used in more "traditional" games as *chess* [Carmel et al. 1993] or *Go*. A final research topic that could be interesting is the application of Player Modeling techniques in *Go*, where machines are not able to defeat the best human players because of the size of its search space. The focus on some parts of it may be promising.

7 Player Modeling in Civilization IV

In this section we will briefly discuss a research in *Player Modeling* applied to the game *Civilization IV*. This section aims to show that we can clearly see behavior differences between distinct agents in this game. These differences are shown comparing data gathered from several matches of agents with different preferences. This section is a summary of [Machado et al. 2011b].

Civilization IV is a turn based strategy game that has six different victory conditions and dozens of agents with different predefined preferences that directly infer on their behavior. The different victory conditions, that can be "pacific" or not, created a very rich environment for experimentation since agents have very distinct behaviors.

There is an interface in the game *Civilization IV* that allows players to generate scripts for the game. We used *AIAutoPlay*, a script generated by [Spronck and den Teuling 2010] that allows human beings to generate games that are going to be played only by the game AI and to collect data of these matches each turn.

The intuition of our previous research was that different preferences would generate different behaviors during the game, to validate it we collected information of 40 matches of each agent (see [Machado et al. 2011b] for details) and analyzed several game metrics for agents with high preference and no preference on some game aspects.

Figure 2 contains two different curves that are the analysis of the culture generated by each agent. We extracted its 5th root because we modeled culture as a polynomial of degree five [Machado et al. 2011b].

The two analyzed agents are *Alexander, the Great*, and *Hatshepsut*. The first agent has no preference for culture while the second has a high preference on this attribute. As expected, we can observe that Hatshepsut's average curve is bigger than Alexander's curve, characterizing different agents (with a confidence of 99%).

To conclude this discussion we can answer the questions proposed on the second section: We intended to generate models that could imply in better opponents for human players (adversarial) and we were able to describe agent's strategy with our modeling. As already discussed, these models are the first step for mimicking human players (substitution) and our models were derived from players' actions and status. We did it offline and implicitly.

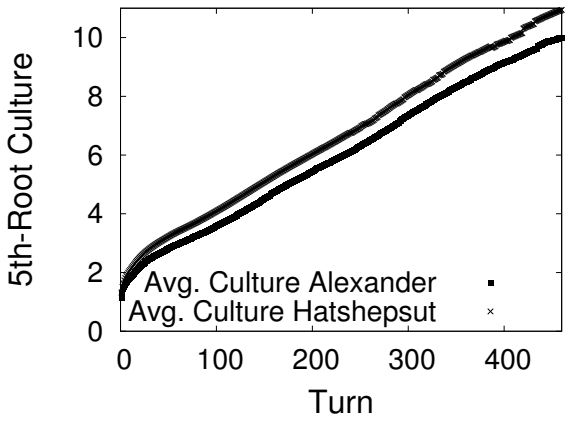

Figure 2: *Comparison of Culture between different agents*

This was a very brief discussion just presenting one of the several results of our most recent work on player modeling [Machado et al. 2011b], we invite interested readers to read the original paper, that contains a much deeper discussion.

Acknowledgments

We would like to thank Dr. Pieter Spronck who kindly shared with us the *AIAutoPlay* script discussed on Section 7.

This work is partially supported by CAPES, CNPq and Fapemig.

References

BAKKES, S., SPRONCK, P., AND VAN DEN HERIK, J. 2009. Rapid and Reliable Adaptation of Video Game AI. *IEEE Transactions on Computational Intelligence and AI in Games 1*, 2 (june), 93 –104.

BERNHARD. Torcs. `http://torcs.sourceforge.net/`.

BIOWARE. Neverwinter Nights 2 - NwN2 Builders - Scripting — BioWare Social Network. `http://social.bioware.com/forum/1/category/164/index`.

BURO, M. ORTS Homepage. `http://skatgame.net/mburo/orts/`.

BWAPI. bwapi - An API for interacting with Starcraft: Broodwar. `http://code.google.com/p/bwapi/`.

CARMEL, D., CARMEL, D., MARKOVITCH, S., AND MARKOVITCH, S. 1993. Learning Models of Opponent's Strategy in Game Playing. In *In Proceedings of the AAAI Fall Symposium on Games: Planning and Learning*, The AAAI Press, 140–147.

CHAMPIONSHIP, M. A. Mario AI Championship. `http://www.marioai.org/`.

CHARLES, D., AND BLACK, M. 2004. Dynamic Player Modelling: A Framework for Player-centred Digital Games. In *International Conference on Computer Games: Artificial Intelligence, Design and Education*, 29–35.

CUNHA, R. L. D. F., AND CHAIMOWICZ, L. 2010. An Artificial Intelligence System to Help the Player of Real-Time Strategy Games. In *Proceedings of the 2010 Brazilian Symposium on Games and Digital Entertainment*, IEEE Computer Society, Washington, DC, USA, SBGAMES '10, 71–81.

DEN TEULING, F. 2010. *Player Modelling in Civilization IV*. Master's thesis, Faculty of Humanities of Tilburg University, the Netherlands.

DRACHEN, A., CANOSSA, A., AND YANNAKAKIS, G. N. 2009. Player Modeling using Self-Organization in Tomb Raider: Un-

derworld. In *Proceedings of the 5th international conference on Computational Intelligence and Games*, IEEE Press, Piscataway, NJ, USA, CIG'09, 1–8.

FIRAXIS. CvGameCoreDLL. http://www.firaxis.com/downloads/Patch/CvGameCoreDLL_v161.zip.

GAMES, K. Civilization IV. http://www.2kgames.com/civ4/.

GAMING, W. Neverwinter Script directory - NWNWiki, the Neverwinter Nights Wiki - your guide to the game of NWN. http://nwn.wikia.com/wiki/Neverwinter_Script_directory.

GAMING, W. Neverwinter2 Script directory - NWN2Wiki, the Neverwinter Nights 2 wiki - Races, classes, skills and more. http://nwn2.wikia.com/wiki/Neverwinter2_Script_directory.

HAUGLID, J. O. Near Infinity - An Infinity Engine Browser & Editor. http://www.idi.ntnu.no/~joh/ni/.

HLADKY, S., AND BULITKO, V. 2008. An Evaluation of Models for Predicting Opponent Positions in First-Person Shooter Video Games. In *Proceedings of IEEE 2008 Symposium on Computational Intelligence and Games (CIG)*, 39–46.

HOULETTE, R. 2003. *Player Modeling for Adaptive Games*. Charles River Media, Dec.

ID SOFTWARE. Índice de /idstuff/source/. ftp://ftp.idsoftware.com/idstuff/source/.

IIDA, H., AND KOTANI, Y. 1998. Tutoring Strategies in Game-Tree Search. *Games of no chance: combinatorial games at MSRI 18*, 4, 433–435.

JAVABOT. JavaBot for Unreal Tournament. http://utbot.sourceforge.net/.

JENKINS, T. Main Page - Supertuxkart. http://supertuxkart.sourceforge.net/.

LAIRD, J. E., AND LENT, M. V. 2000. Human-Level AI's Killer Application: Interactive Computer Games. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, AAAI Press, 1171–1178.

LAVIERS, K., SUKTHANKAR, G., MOLINEAUX, M., AND AHA, D. 2009. Improving Offensive Performance through Opponent Modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence for Interactive Digital Entertainment*, 58–63.

MACHADO, M. C., FANTINI, E. P. C., AND CHAIMOWICZ, L. 2011. Player Modeling: Towards a Common Taxonomy. In *Proceedings of 16th International Conference on Computer Games*, CGAMES 2011 USA, 50–57.

MACHADO, M. C., ROCHA, B. S. L., AND CHAIMOWICZ, L. 2011. Agents Behavior and Preferences Characterization in Civilization IV. In *Proceedings of the 2011 Brazilian Symposium on Games and Digital Entertainment*, IEEE Computer Society, Salvador, Brazil, SBGAMES '11.

MANOVICH, L. 2002. *The Language of New Media (Leonardo Books)*. The MIT Press, Mar.

MARTÍNEZ, H. P., HULLETT, K., AND YANNAKAKIS, G. N. 2010. Extending Neuro-evolutionary Preference Learning through Player Modeling. In *Proceedings of IEEE 2010 Symposium on Computational Intelligence and Games CIG'10*.

MARTINS, A. Open Sonic The Hedgehog. http://opensnc.sourceforge.net/.

NAREYEK, A. 2004. AI in Computer Games. *Queue 1* (February), 58–65.

NECHVATAL, J. 1999. *Immersive ideals/critical distances. A study of the affinity between artistic ideologies based in virtual reality and previous immersive idioms*. PhD dissertation, Centre Adv. Inquiry Interactive Arts (CAiiA), Univ. Wales College, Wales, U.K.

NWN2SCRIPTING. Nwn2 Scripting - Home. http://www.nwn2scripting.com/.

ORKIN, J., SMITH, T., AND ROY, D., 2010. Behavior Compilation for AI in Games.

PALI. Stratagus. https://launchpad.net/stratagus.

PALI. Wargus - Warcraft II. https://launchpad.net/wargus.

PEDERSEN, C., TOGELIUS, J., AND YANNAKAKIS, G. N. 2010. Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games 2*, 1, 54–67.

PERSSON, M. Infinite Mario Bros! http://www.mojang.com/notch/mario/.

PONSEN, M., GERRITSEN, G., AND CHASLOT, G., 2010. Integrating Opponent Models with Monte-Carlo Tree Search in Poker.

RIO, P. The Programming Language Lua. http://www.lua.org/.

SECRETMARYO.ORG. Secret Maryo Chronicles. http://www.secretmaryo.org/.

SMITH, A. M., LEWIS, C., HULLETT, K., SMITH, G., AND SULLIVAN, A. 2011. An Inclusive View of Player Modeling. In *Proceedings of the 6th International Conference on the Foundations of Digital Games*, ACM, New York, NY, USA, FDG '11.

SMITH, A., LEWIS, C., HULLETT, K., AND SMITH, G. 2011. An Inclusive Taxonomy of Player Modeling. Tech. rep.

SOLUTIONS, S. Home - Flight Gear. http://www.flightgear.org/.

SONIC, A. R. O. API Reference. http://opensnc.sourceforge.net/wiki/index.php/API_Reference.

SPRONCK, P., AND DEN TEULING, F. 2010. Player Modeling in Civilization IV. In *Proceedings of the 6th Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, 180–185.

TAN, C., TAN, K., AND TAY, A. 2011. Dynamic Game Difficulty Scaling using Adaptive Behavioural Based AI. *Computational Intelligence and AI in Games, IEEE Transactions on PP Issue:99*, 99, 1–1.

TAYLOR, L. N. 2002. *Video games: Perspective, point-of-view, and immersion*. M.S. thesis, Grad. Art School, Univ. Florida, Gainesville, FL.

THUE, D., BULITKO, V., SPETCH, M., AND WASYLISHEN, E. 2007. Interactive Storytelling: A Player Modelling Approach. In *In Proceedings of the third Artificial Intelligence and Interactive Digital Entertainment conference (AIIDE' 07)*.

THURAU, C., BAUCKHAGE, C., AND SAGERER, G. 2003. Combining self organizing maps and multilayer perceptrons to learn bot-behaviour for a commercial game. In *GAME-ON*.

TOGELIUS, J., DE NARDI, R., AND LUCAS, S. 2007. Towards Automatic Personalised Content Creation for Racing Games. In *IEEE Symposium on Computational Intelligence and Games (CIG) 2007*, IEEE, 252–259.

VALVE. Authoring Tools/SDK (Alien Swarm). http://developer.valvesoftware.com/wiki/Alien_Swarm_SDK.

VALVE. Authoring Tools/SDK (Left 4 Dead). http://developer.valvesoftware.com/wiki/Authoring_Tools/SDK_(Left_4_Dead).

VALVE. Authoring Tools/SDK (Left 4 Dead 2). http://developer.valvesoftware.com/wiki/Authoring_Tools/SDK_(Left_4_Dead_2).

VALVE. Valve Developer Community. http://developer.valvesoftware.com/wiki/Main_Page.

VAN DEN HERIK, H. J., DONKERS, H. H. L. M., AND SPRONCK, P. H. M. 2005. Opponent Modelling and Commercial Games. In *Proceedings of IEEE 2005 Symposium on Computational Intelligence and Games (CIG'05)*, G. Kendall and S. Lucas, Eds., 15–25.

WEIMER, W. WeiDU.org: Infinity Engine Utilities and Mods . http://www.weidu.org/.

Marlos C. Machado is a graduate student in the Computer Science Department at Federal University of Minas Gerais. His research interests include machine learning and artificial intelligence for games, in particular player modeling. The work described in this paper is related with his Master's research.

Eduardo P. C. Fantini is a graduate student in the Computer Science Department at Federal University of Minas Gerais. His research interests include multi-agent coordination, chatterbots and general artificial intelligence for games.

Luiz Chaimowicz is an assistant professor in the Department of Computer Science and has a CNPq Productivity Grant (level 2). His PhD and postdoctoral were developed in the robotics field, more specifically in the development of artificial intelligence algorithms for multiple robots coordination. Since 2005 he is also working in the games field. He was one of the SBGames08 general coordinators and committee coordinator of the Computing Track in SBGames09. He is a committee member of the SBGames editions from 2007 to 2011. He was also one of the project coordinators of the game "Estrada Real Digital", funded by Finep in the Educational Games Edict.