Automatically Adjusting Player Models for Given Stories in Role-Playing Games

Natham Thammanichanon
Department of Computer Engineering
Chulalongkorn University, Payathai Rd.
Patumwan Bangkok, Thailand
Tel: (+66841121819)
natham.t@student.chula.ac.th, tun.natham@gmail.com

Vishnu Kotrajaras
Department of Computer Engineering
Chulalongkorn University, Payathai Rd.
Patumwan Bangkok, Thailand
Tel: (+66890212323)
ajarntoe@gmail.com, vishnu@cp.eng.chula. ac.th

Abstract

Different kinds of players favor different game stories. Player Archetype Change Management (PACM) system is a drama management system which changes the story of role playing games according to a player model monitored during gameplay. Authors give each of his stories a matching player model. While a player plays the game, PACM selects the story that most matches current player model. However, players may not agree with a model defined for a story by its author. Players' opinions should be used to adjust the player model associated with each story. In this paper, we present the technique for adjusting the player model of each story in PACM using observed data from players. This provides the system with a more reliable player model for future playing sessions.

Keywords

Player model, Interactive narrative, Case-based planning, Commercial game.

1. Introduction

Many kinds of techniques were applied in computer games to make players enjoy their game more. Some of the techniques adjusted game elements to be more appropriate for individuals according to their playing styles. However, the most common methodology was to offer players choices during gameplay in order to make a game story progress in different directions that were prepared by its author. Not all players preferred a story given to them by such a traditional manner. Moreover, the methodology actually put a limit on some great narratives. Several works made use of player models but they usually boiled down to managing narratives in order to conserve authorial goals. Player Archetype Change Management system [1] (PACM) was a drama management system that used the player personality models to manage stories. Its drama manager translated a player's actions to the player's personality model then used the model to indicate which story should be narrated in order to gratify the player. Currently, an author defined a player model he believed to be appropriate for each of his stories. The system then matched a player's actual model obtained during play with the author-defined models. However, this matching mechanism might not be sufficient. We believed that author-defined player models for stories were too author dependent. Player models for stories should take players' satisfaction into account, as well as authors' opinions.

This paper presents a new approach for augmenting a player model for each story. Player satisfaction was used to revise a player model for a played story. Our system was created using Neverwinter Nights [2] (NWN) game environment with NWNScript, jRCEI [3] and DLModel [4].

This paper is organized as follows: Section 2 covers related works, Section 3 details PACM, Our proposed technique is explained in section 4, Experiments are discussed in section 5. Finally, section 6 summarizes the paper and discusses future work.

2. Related Works

Many researches explored the utility of enacting an interactive narrative with more appropriate and adaptable AI. Character-driven narrative was one approach, which relied on interaction between players, artificial self-determining characters and a game environment. Cavazza [5] used Hierachical Task Networks for each agent decision re-planning to create a character-driven narrative. However, a story generated by a character-driven approach usually could not generate an engaging experience.

Another approach was a plot-driven story management. Some contributions were towards story generation. Story elements were selected based on past events, character relationships and author's goals. Some works in this approach focused on plot or event generation like DINAH [6]. DINAH generated plots through the composition of narrative events from a story database according to Braginan cinematic narrative model. Façade [7] had a drama manager which built its story through player actions from primitive elements of a story, called beats. Fairclough [8] used a story director to plan a narrative by retrieving similar story cases using game information based on current game information and player actions. Some works focused on keeping players in a given plot. An interactive narrative architecture proposed by Young [9] generated plans annotated with a causal structure, monitored player actions, re-planned or prevented actions that were story threats. Magerko [10] proposed an architecture that depicted a story from a prewritten plot and autonomous characters. Its story director managed a plot by guiding non-player characters to take restorative actions if player actions were likely to have an impact to the story plot. El-Nasr [11] proposed Mirage, which utilized a player model analyzed from the player's behavior. The model was used to modify non-players' behavior to appropriately encourage players to achieve the story goal. They did not make use of player archetypes.

Sharma utilized a drama manager with a player preference model [12]. The model represented a player's interest in his played story route. This method constructed a player model by having player fill in an inquiry which represented his likes and dislikes for a game after he had finished playing it. When a new player played the game, his actions were compared with recorded actions from existing players. If a similar record was found, the game would try to steer events towards the previous player's likes and dislikes. However, each of Sharma's models was tied to an individual player. It was difficult for an author to prepare alterative stories because there was no standard model to refer to.

Thue proposed PaSSAGE [13]. It applied player modeling to learn each player's playing style, then used it to select an event to be narrated in the story. However, only story events were allowed to update a player model. In real games, other events, such as how a player killed monsters, were also important to the player model.

PACM used player personality models analyzed from player actions to manage game stories. However, each story and its player model compatibility only relied on its author's opinion. Our approach extended PACM such that the player model for each story was influenced not only by its author, but also by players who played the story. This allowed better model matching against players.

3. Player Personality Modeling in PACM

Our approach extended the player modeling module in PACM. Its architecture is shown in Figure 1. The player personality model in PACM was based on Bartle's "player category" [14] which classified player characteristics as {achiever, explorer, socializer, killer}. It was a combination of the percentage of these categories and its confidence value which represented how much a current player was trying to continue the story and how his actions were consistent with the model. An example is shown in Figure 2. When a player started to play a game, PACM initialized the player model from his character status and selected a story from the initial model. When the player carried out an action, his model was updated. This player model was then compared with the model for a current story. If the difference between both models was more than a given value, the confidence value decreased. Otherwise, the confidence value increased. A single unintended action that did not match the story selected model could not make an immediate impact on the confidence value. When the confidence value was below a predetermined threshold value, the drama manager attempted to change the story.

For example, a player personality model used to select a current story was {achiever 10%, explorer 50%, socializer 40%, killer 0%} and its confidence value was 70. When the player went into an inn and talked to some non-story-relate NPCs, his talking actions would update his personality model to {achiever 6%, explorer 52%, socializer 42%, killer 0%}. The distance value between the updated model and the model defined for the current story did not become more than a predefined value. Hence the confidence value of the player personality model was increased to 72. On the other hand, if the player talked with other characters very frequently, his personality model might become {achiever 0%, explorer 30%, socializer 70%, killer 0%}. This model differed from the model defined for the story more than our given limit. Therefore the confidence value would decrease instead.

4. Our Approach for Enhancing a Player Model Defined for a Story

A story should have its player model constructed from its author and its players. In our attempt to extend PACM with the addition of players' view, we had 2 steps, observing and updating. In the observing step, gameplay was monitored and information was collected in the form of a "story log". A story log contained all stories which a player had played and personality model used to select those stories. An overview procedure of our approach is shown in Figure 3.

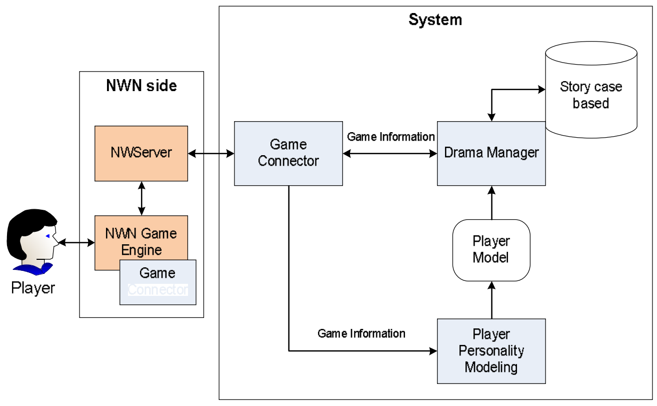

Figure 1. PACM components

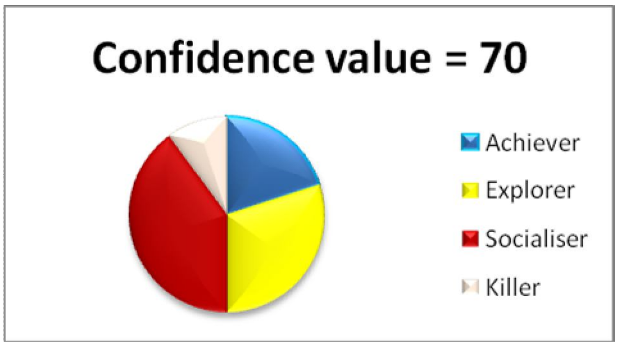

Figure 2. An example of a player personality model

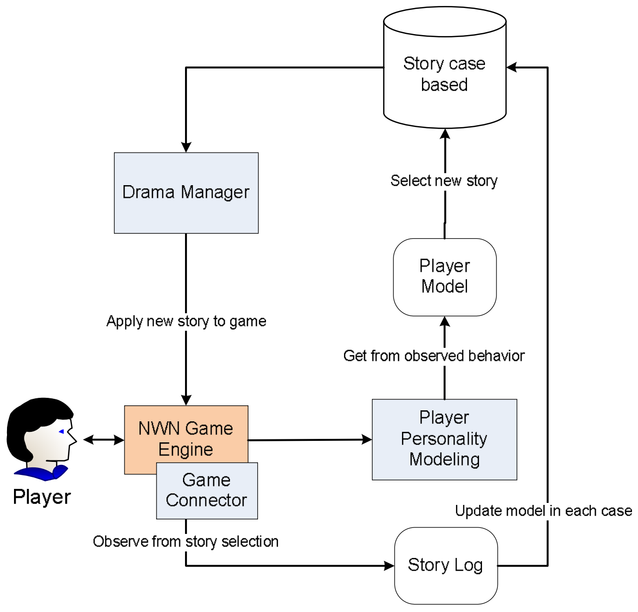

Figure 3: An overview of our updating procedure

The updating step began after the player finished his current game. The story log was used. For stories that the player could not finish, their player models were updated such that they were more unlikely to be selected next time by the same player (or players with similar personality model). For a story that the player finished, its player model was updated such that they were more likely to be selected next time. How the player model for a story and its confidence value changed is defined in equation (1), (2) and (3).

$$score_s^c = (P_s^c * confidence_s + w_p * (P_{ps}^c - P_s^c) * confidence_{ps})/100 \quad (1)$$

$$confidence'_s = (1 + w_c) * confidence_s \quad (2)$$

$$P'^c_s = score_s^c / confidence'_s * 100 \quad (3)$$

Where	C	= {*achiever, explorer, socializer, killer*}.	
	c	= a member of set C.	
	P^c_s	= the percentage of personality c of the model defined for story s, before updating.	
	P'^c_s	= the percentage of personality c of the model defined for story s, after updating.	
	P^c_{ps}	= the percentage of personality c of the player model used to select story s.	
	$confidence_s$	= confidence value of the model defined for story s, before updating.	
	$confidence_s$	= confidence value of the model defined for story s, after updating.	
	$confidence_{ps}$	= confidence value of the player personality model used to select story s.	
	w_p	= 0.5	if story s finished.
	w_p	= -0.5	if story s did not finish.
	w_c	= w_{cf}	if story s finished.
	w_c	= -0.05	if story s did not finish.
	w_{cf}	= 0.1	if $confidence_{ps} >= confidence_s$
	w_{cf}	= 0.1 * $confidence_{ps}/confidence_s$	if $confidence_{ps} < confidence_s$
	w_{cf}	= 0	if $confidence_{ps} < 0$

5. Experiments and Discussions

We conducted an experiment with 11 participants to test our approach. Stories used in this experiment were D&D adventures adapted from Dungeon magazines. We first asked each participant to evaluate himself according to Bartle's model. Each of them assigned a percentage score to each Bartle's category and PACM generated an initial player personality model for each player using information from the character creation screen. These are shown in table 1. Because it was difficult to find participants who shared similar playing styles, we asked each participant to play two games with the same character. Each game was played until a story was completed (story could change in-between) with same character for each participant.

If our approach worked, for each player whose playing style did not match his given story (stories), the player model for the story should be updated such that it became more different from the player's own model. Each participant should prefer the story in his 2nd gameplay to the story in his 1st gameplay. If not, at least the participant should love both stories equally.

The result could be classified into 2 groups. The first group was the results from players whose profiles matched the originally given story. These players finished the story without the system attempting to change it. The model for each story was modified to obtain more chance to be selected in the second gameplay. Therefore the players in this category got the same story in their second gameplay. This was good for participants (P2, P3, P5, P6 and P9) who enjoyed the initial story in the first gameplay because they could still enjoy it in their second gameplay. Figure 4 shows 2 graphs. Each graph represents the distance between players' own personality models and a story model for players who played through story S3 (most players in this group played this story). The red line shows the distance when our adaptation technique was not used. The blue line shows the distance when our adaptation technique was used.

However, there were some participants (P1, P4 and P10) who did not like to play their initial stories but finished their stories with no story changing attempt from the system. We discovered that this was because the game environment did not have enough content for allowing them to change their playing archetypes. Our system interpreted that these players enjoyed their initial stories and updated the stories' models accordingly. Therefore they had to play the same story in their second gameplay. For future tests, all of our stories would have to be modified to include enough elements of different player profiles to allow change.

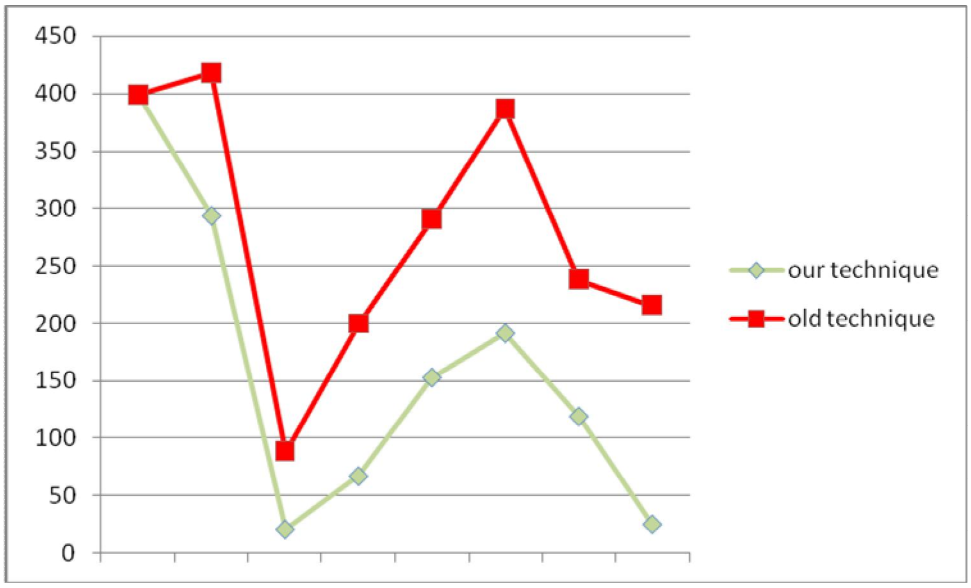

Figure 4. Comparison of distance between players' models and S3 story's model

The other group included participants who while playing, the system attempted to change their stories. There were three of these players in our experiment (P7, P8 and P11). When such attempts took place, the player models for their initial stories were modified to be less likely to be obtained again.

It was discovered, however, that although those models of the stories were updated, our prototype could not adjust story models enough for more suitable stories to be selected in the second gameplay. For player P8, although the system had an attempt to change his story model in both games, player P8's own personality model was still most suitable for story S3. The system therefore selected story S3 for participant P8 again. He eventually finished it. This resulted in story S3 being adjusted twice, to be less likely to be selected (when an attempt to change the story occurred) and to be more likely to be selected (when the player finished the story). The changing of the personality model of his story is shown in Figure 5. For player P7 and P11, their stories actually altered. Each of these players finished his altered story in his first gameplay, causing his finished story's model to become more likely to be selected in his second gameplay and his initial story to become less likely to be selected. However, player P7's own personality model was still closer to story S5 (his initial story from his first game) than story S2 (the story he finished in his first game). This was probably due to the huge effect created by the update mechanism of PACM during play. Following current story even slightly caused the story to become considerably harder to change. Player P7 might follow story S5 differently in his second game, thus making the system unable to bring the story model closer to S2. Therefore the system selected story S5 for him in his second play, without changing the story. For player P11, his own personality model's confidence value was too low to cause his finished story's model to be selected in his second gameplay. In order to achieve a more effective story change, our system would need to scale each update further.

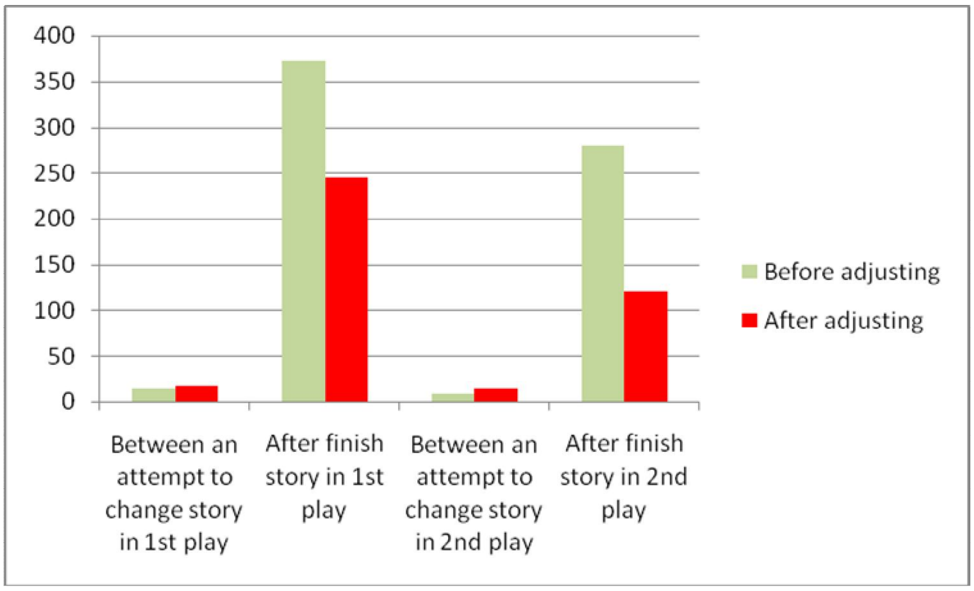

Figure 5. Distance between P8's player model and the model of his played story, S3

6. Conclusion and future works

In this paper, we had proposed a technique for adjusting the player model of each story using observing data from players. We extended PACM, a drama management system, with such feature. Our technique was able to adjust the player personality model of each story so that stories enjoyed by players became more likely to be selected and stories not enjoyed by players became less likely to be selected. Each story used in our experiment still did not have quite enough elements for players to play around with, which limited changes in the story, causing some players to get stuch with stories they did not like. A better design for each story can fix this problem. Due to time limitation, we could only have each player played 2 games. The updating procedure could not steer stories that players did not like away from being selected in the second game. To solve this problem, the score of each update needs to be re-scaled. This is only the problem of adjustment, however.

For future works, we plan to adjust confidence value when updating model for stories that players did not like. Moreover, we plan to provide a better initial player's personality model for each player. Instead of deriving the initial model from the player's statistics, which may cause the model to be unsuitable for some players, we plan to derive the initial model from the player's behavior during the starting phase of the game.

References

[1] N. Thammanichanon and V. Kotrajaras, PACM: Player Archetype Change Management System in Role-playing Games, *Proceedings of 13 th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia, Educational & Serious Games* (2008), Wolverhampton, UK.

[2] Bioware, Neverwinter Nights, http://nwn.bioware.com, 2008.

[3] F. Peinado, RCEI: An API for Remote Control of Narrative Environments. , *Proceedings of the 4th International Conference on Virtual Storytelling,* 2007.

[4] F. Peinado, DLModel, a tool for dealing with description logics, http://federicopeinado.com/projects/dlmodel, 2008.

[5] M. Cavazza, F. Charles and S. J. Mead, Character-Bsed Interactive Storytelling, *IEEE Intelligent Systems* (2002), 17-24.

[6] D. Ventura and D. Brogan, Digital Storytelling with DINAH: dynamic, interactive, narrative authoring heuristic, *Proceedings of the International workshop on Entertainment Computing (IWEC)* (2002), pp. 91-99.

[7] M. Mateas and A. Stern, Integrating plot, character, and natural language processing in the interactive drama Façade, *Proceedings of 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment (TISDE-03)* (2003).

[8] C. R. Fairclough and P. Cunningham, AI structuralist storytelling in computer games, *Proceedings of the International conference on Computer Games: Artificial Intelligence, Design and Education* (2004).

[9] R. M. Young, M. Riedl, M. Branly, A. Jhala, R. Martin and C. Sagretto, An architecture for integrating plan-based behavior generation with interactive game environments, *Journal of Game Development* vol. 1 (2004).

[10] B. Magerko, J. Laird, M. Assanie, A. Kerfoot and D. Stokes, AI characters and directors for interactive computer games, *Proceedings of the 2004 Innovative Applications of Artificial Intelligence Conference* (2004).

[11] M. S. El-nasr, A User-Centric Adaptive Story Architecture: Borrowing from Acting Theories, *Proceedings of the ACM SIGCHI International Conference on Advances in computer entertainment technology* (2004).

[12] M. Sharma, S. Ontanon, C. Strong, M. Metha and A. Ram, Towards player preference modeling for drama management in interactive stories, *Proceedings of the Twentieth International FLAIR Conference on Artificial Intelligence (FLAIR)* (2007), AAAI Press, 571-576.

[13] D. Thue, V. Bulitko, M. Spetch and E. Wasylishen, Interactive storytelling: A player modeling approach, *Proceedings of the 3rd conference on Artificial Intelligence and Interactive Digital Entertainment* (2008), Stanford, California, USA, 43-48.

[14] R. A. Bartle, *Designing Virtual Worlds* (2004), New Riders Publishing.