A Game Architecture for Emergent Story-Puzzles in a Persistent World

Federico Peinado

Dpto. Sistemas Informáticos y Programación (UCM)
C/ Profesor José García Santesmases s/n
28040 Madrid (Spain)
(+34) 91 394 7646
fpeinado@fdi.ucm.es

Pedro Pablo Gómez-Martín

Dpto. Sistemas Informáticos y Programación (UCM)
C/ Profesor José García Santesmases s/n
28040 Madrid (Spain)
(+34) 91 394 7624
pedrop@sip.ucm.es

Marco Antonio Gómez-Martín

Dpto. Sistemas Informáticos y Programación (UCM)
C/ Profesor José García Santesmases s/n
28040 Madrid (Spain)
(+34) 91 394 7621
marcoa@sip.ucm.es

ABSTRACT

This paper presents the design of an architecture for narrative games with story-puzzles like classic graphic adventures. The system is able to create new short stories in each session, combining a basic set of narrative elements in an emergent way but maintaining coherency with the storyline of previous sessions. As a test-bed of this proposal we use a simple detective game inspired on the famous *Cluedo*'s characters.

Keywords

Game Architecture Design, Game AI, Interactive Storytelling, Automatic Generation of Puzzles

INTRODUCTION

There was a time when hundreds of adventure games were in the market. Most of them were just pieces of funny use-and-throw-away interactive literature, good for business but not so good for customers' wallets. Nowadays, a new generation of 3D highly replayable games dominate the market. There are still some remaining instances of classic adventures, but only a small number of them try to tell a more replayable story that changes and becomes slightly different for one session to another. *Blade Runner* [10] is a good example of this kind of games in which computer controlled characters do not always act in the same way, generating a sort of "emergent

plot". Unfortunately, the same *general* plot structure is reused in every particular generation. Nevertheless, this is not considered a big disadvantage, because the whole game takes a long time to be solved (along several sessions), so the average player is not expected to finish completely the game more than once or twice.

Generally speaking, games like *Blade Runner* work using a *predesigned* long-play storyline, which the system "makes up" with slight changes during each game. While this approach is, as said before, an easy alternative for prolonged games, it is not an option for shorter and more interactive and replayable games.

In this paper we present the architecture of a computer game that automatically generates short linked stories that take place in a small and persistent virtual universe. Persistence requirements force the system to maintain coherence *between all the stories.* We exemplify our architecture in the context of a simplified game where the player plays the role of a detective solving mysterious crimes. All the stories take place in the same Victorian mansion, sharing atmosphere and characters with the well-known board game *Cluedo* [8] (*Clue* in the USA version).

Besides of their emergent generation, coherence between crimes, culprits and victims has to be preserved. We are not included "serious crimes" like murder or rape to make possible for players to reuse their accumulated knowledge about characters as *clues* for finding out the solution of each new mystery. Clues are represented by *subtle displays of emotions* during the social interaction between characters and dialogues between each character and the detective, talking about other suspects (see table 1.1).

Stories are generated coherently by the system, establish the crime, the victim the culprit and his motive during the initialisation of each game session. The main part of the story unfolds during the interaction between the detective and the other characters, based on the actions and reactions of the virtual actors that implement the non-player characters. Their behaviours add useful noise to the fictional world, but the mysterious crime is always there in the background, and the final goal of the player is to uncover *who* committed and *why*.

Table 1.1: Hidden clues in character dialogue

Emotion	Text of the character
Balanced	"I don't know if Mrs. Peacock is jealous of Miss Scarlett."
Weak attraction	"Mrs. Peacock has no reasons to be jealous of Miss Scarlett."
Moderated trust	"I cannot think of Mrs. Peacock as the culprit."
Strong hate	"It's obvious, that conceited witch is absolutely jealous!"

STORY GENERATION

Automatic construction of story plots has always been a golden dream in the entertainment industry, especially in commercial videogames that needs a large number of short story plots with no necessity of high novelty or high artistic quality. Of course, nowadays computers are not able to generate automatically complex stories like the ones that human writers can create. But now it is possible to take advantage of the combinatory power of computers to build some simple story-puzzles.

The main problem with computer-generated stories for games is the *interactive dilemma*. Usually game designers don't want to lose all the control about how the plot unfolds: they prefer to *drive* the story. Unfortunately, if designers completely guide the plot, there is no place for computer-generated variations in the story. On the other hand, if the plot is completely entrusted to the system, it is not easy to assure that the result will be valuable. That is the reason why we propose to find a balance –in real time and adapted to each experience– between designer's and player's roles. There is a good explanation of the problems of interactive storytelling in [4].

There are basically two approaches in the automatic plot generation: *centralized* and *distributed generation*. The first one is based on an intelligent system that controls all the details of the plot development, like Minstrel [9], that creates new tales combining pieces of Arthurian legends. The second one is based on a *multi-agent system* in which the plot *emerges* from the behaviour of autonomous characters, because there is no main plot to follow, like I-Storytelling [2].

The implementation of the centralized approach is quite complex and it lacks a truly emergent behaviour. In contrast, consistency management is a hard problem in the distributed generation, especially if the model grows during the design process and becomes more and more sophisticated.

THE GAME ARCHITECTURE

We propose a double layer architecture. The so-called *narrative layer* has an abstract plot generator that sets the initial game state, the location of objects and characters and the social relations between them. All these avatars inhabit in the persistent world, known as *simulation* or *agent layer*. This layer controls the simulation progress, using the guidelines imposed by the narrative layer.
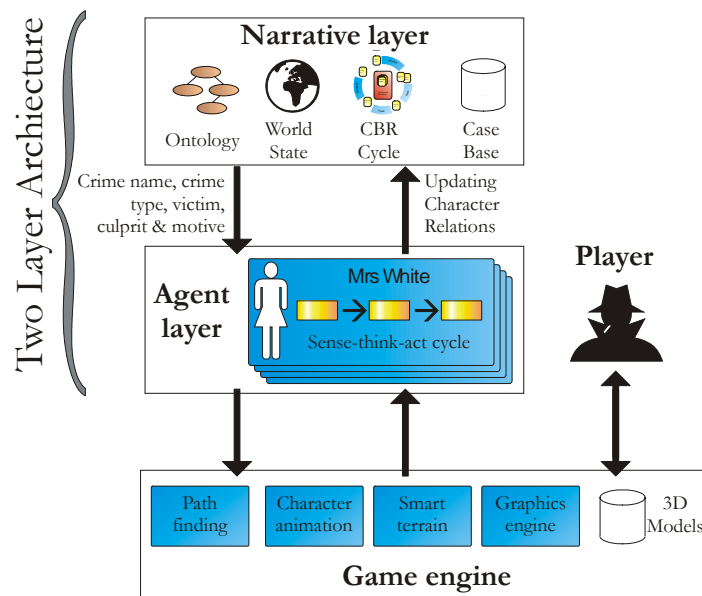
Figure 1.1: Double-layer architecture overview

This architecture is a mixed approach to the problem of generating story plots from a set of narrative components analysed in terms of Description Logics (a subset of the well-know First

3

Order Logic [1]) and simple numerical attributes. These relations are based in OCC, a cognitive model of emotions for agents [7]. An AI-process is defined to generate plots from the player actions, the initial *multiform plot* and a random initial setting for the story, using a narrative ontology (based on well-known narrative theories [5]) to measure the semantic distance between elements taking part in the story development. Table 2.1 presents a summary of the algorithm for the story puzzle generation.

Table 2.1: Algorithm of Story-Puzzle generation

Step	Explanation
Random seed generation	Create a new crime (type, victim, culprit, motive)
Retrieve similar situation	CBR retrieval + Narrative ontology
Adapt old situation to current state of characters	CBR adaptation + Heuristic rules (TRAMs)

Formal logics and ontologies allow the system to maintain coherence and structure in the global puzzle. For instance, a crime with a similar motive can be retrieved and then, the system has to adapt the emotional relations between characters adding facts to the plot.

The game architecture proposed here allows the designer to have indirect control over two different layers. Firstly, abstract generation of each story plot can be controlled in the narrative layer establishing the theme, mood and background of the game, the set of possible crimes, characters and their motivations. In the second place, the autonomous characters also have a set of different personalities (coded behaviours) for the designer to combine and create emergent and interesting events.

Narrative Layer

The narrative layer is the more abstract component of the architecture dedicated to the management of the plot. This layer of the system generates the story-puzzle for each game using a creative Case-Based Reasoning (CBR) process [3]. This is a problem-solving method based on a four-stages cycle. When solving a new problem, the first stage is to retrieve (*remember*) a similar old problem from the case base (*episodic memory*); the second stage is to adapt the old solution of the retrieved case to the new one, creating a new solution; the third stage is to test that new solution and finally there is a last stage for learning the new pair problem-solution as a case that will be integrated in the case base.

The creative CBR process of the narrative layer includes the first two stages. Firstly, a retrieval of a case from the case base with a crime similar to a random generated one. Secondly, an adaptation of the current characters situation based on the retrieved characters situation, using heuristics rules to change the emotional relations into characters, like "if someone has an affair with somebody you loved, you get angry with him/her". In the next section of this paper there is a more detailed explanation of the CBR process that takes place in this layer.

Agent Layer

The agent layer is in charge of the fictional world to be presented to the player. This layer has to control every autonomous agent that inhabits in that world. At the very beginning of the plot, the narrative layer illustrates the characters with the relation that each one has with the others.

The agents use the well-known *sense-think-act* cycle. The first stage is implemented using the smart terrain ideas from Will Wright. The environment is populated by tagged objects that broad their properties to the rest of the objects and characters. The agents can intercept these properties in the *sense* stage to know which options they have at every moment. In order to give coherence to the implementation, we consider also each autonomous agent as a tagged object, to make possible other characters to detect its presence and interact with it.

The think stage decides which of the available objects the character will interact with. The decision has to be guided using some human concepts as ownership or responsibility [6]. In some cases the decision process may require the aided of the narrative layer, in order to detect the relevance of the possible action in the development of the storyline.

Once the next action to be performed has been chosen by the think stage, the agent uses a low-level IA *sublayer* that establishes how it should be executed. In that sense, using the information of the selected tagged object, the layer decides whether the character has to move towards the object or not (calling the path finding routines if needed), and chooses the suitable character animation to be sent to the underlying graphic engine.

CASE-BASED REASONING PUZZLE GENERATION

To explain the ideas of this paper, we use a basic set of story elements: Cluedo's characters (Mrs. Peacock, Colonel Mustard, Reverend Green, Professor Plum, Miss Scarlett and Mrs. White), several types of crimes (abuse, arson, assault, burglary, coercion, disorderly conduct, extortion, forgery, fraud, harassment, theft, threat and vandalism) and different motives for the culprit (anger, avarice, envy, jealousy, lack of money and revenge).

Each character has an emotional relation vector with three components or relations to the others: (attraction/disgust, trust/distrust, love/hate) with integer values in [-10,10]. A component is considered *balanced* for the value 0, *weak* for absolute values in [1,3], *moderated* for absolute values in [4,6] and *strong* for absolute values in [7,10].

Excluding the first time, before each generation process, there is always a starting point for the characters relation, described as a matrix of vectors (see table 3.1). Then, the system generates a random crime, a vector with five components: crime name, type, victim, culprit and motive, like the example of table 3.2.

Table 3.1: Previous world state

Characters	toPeacock	toMustard	toGreen	toPlum	toScarlett	toWhite
Mrs. Peacock	-	(3,0,0)	(0,0,0)	(-1,0,0)	(0,5,5)	(0,0,0)
Colonel Mustard	(1,0,0)	-	(0,0,0)	(0,0,0)	(5,0,0)	(0,0,0)
Reverend Green	(0,0,0)	(0,0,0)	-	(0,0,0)	(0,0,0)	(0,0,0)
Professor Plum	(0,0,0)	(0,0,0)	(0,0,0)	-	(0,0,0)	(0,0,0)
Miss Scarlett	(0,0,0)	(2,0,0)	(2,0,0)	(2,0,0)	-	(0,0,0)
Mrs. White	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	-

Table 3.2: Random generated crime

Crime name	Crime type	Victim	Culprit	Motive
"New"	Theft	Miss Scarlett	Mrs. Peacock	Jealousy

After the generation of the crime, an instance of the case base is retrieved comparing the crime vector. A similar crime is retrieved, using the distance between concepts in a simple ontology. In this example (table 3.3), the crime type and the motive of the culprit are exactly the same so the similarity is high. The other part of the retrieved case is the *recommended* world state to start the story and setup the game ready for the detective to uncover this mystery (table 3.4).

Table 3.3: Crime vector of the retrieved case

Retrieved Crime	Crime	Victim	Culprit	Motive
"The Thief of the Indian Pipe"	Theft	Colonel Mustard	Professor Plum	Jealousy

Table 3.4: Recommended world state matrix of the retrieved case

Characters	toPeacock	toMustard	toGreen	toPlum	toScarlett	toWhite
Mrs. Peacock	-	(3,0,0)	(0,0,0)	(-1,0,0)	(0,5,5)	(0,0,0)
Colonel Mustard	(2,0,0)	-	(0,0,0)	(0,0,0)	(2,0,0)	(2,0,0)
Reverend Green	(0,0,0)	(0,0,0)	-	(0,0,0)	(0,0,0)	(0,0,0)
Professor Plum	(-1,0,0)	(0,5,5)	(0,0,0)	-	(5,0,0)	(0,0,0)
Miss Scarlett	(0,0,0)	(7,0,0)	(0,0,0)	(0,0,0)	-	(0,0,0)
Mrs. White	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	-

The adaptation of the current world state to fit the recommended world state is the most difficult task that the system has to achieve. Using specific heuristic rules, the system modifies the current world state based on the differences between both. The idea is inspired on the adaptation process of Minstrel [9], the story generator that uses TRAMs (Transform-Recall-Adapt Methods) to create new story events reusing cases, e.g.: transforming an affair between Colonel Mustard and Miss Scarlett in a romance between Mrs. Peacock and Professor Plum.

The result is the initial state of the character´s relations in the new game, a reinterpretation of the recommended state based on the current state of the persistent world (see table 3.5).

Table 3.5: Adapted world state matrix

Characters	toPeacock	toMustard	toGreen	toPlum	toScarlett	toWhite
Mrs. Peacock	-	(**5**,0,0)	(0,0,0)	(-1,0,0)	(0,5,5)	(0,0,0)
Colonel Mustard	(**0**,0,0)	-	(0,0,0)	(0,0,0)	(**7**,0,0)	(0,0,0)
Reverend Green	(0,0,0)	(0,0,0)	-	(0,0,0)	(0,0,0)	(0,0,0)
Professor Plum	(0,0,0)	(0,0,0)	(0,0,0)	-	(0,0,0)	(0,0,0)
Miss Scarlett	(0,0,0)	(2,0,0)	(2,0,0)	(2,0,0)	-	(0,0,0)
Mrs. White	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	-

CONCLUSIONS

The architecture presented here allows the system to create a new combination of game content in each game session, presenting each time a new story-puzzle coherently linked to the main storyline of the persistent world. Depending on the player skills solving the puzzle and also the emergent interaction between characters, emotional relations between characters *evolve*, creating a different initial situation for each game. To sum up, using this architecture, designers of narrative games will improve the replay factor of their products, reusing static material to create more dynamic stories; at the same time player's experiences are enriched with the participation of more deeply interrelated characters.

The next step of this research is to implement this architecture in an up-to-date game engine to translate abstract terms to concrete models of characters and immersive interaction. This will allow the evaluation of results from the point of view of the experience of regular gamers.

ACKNOWLEDGEMENTS

REFERENCES

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. The Description Logic Handbook: Theory, Implementation and Applications, 2003.

2. Cavazza, M., Charles, F., Mead, S.J. Character-based Interactive Storytelling. IEEE Intelligent Systems, special issue on AI in Interactive Entertainment, 2002 pp. 17-24.

3. Gervás, P., Díaz-Agudo, B., Peinado, F., and Hervás, R. Story Plot Generation based on CBR, 24th Annual International Conference of the British Computer Society's Specialist Group on Artificial Intelligence (SGAI). Applications and Innovations in Intelligent Systems XII, Ed.: Macintosh, A., Ellis, R., Allen, T. Cambridge, UK. Springer, WICS series, 2004 pp. 33-46.

4. Mateas, M. "An Oz-Centric Review of Interactive Drama and Believable Agents", 1999.

5. McKee, R. Story: Substance, Structure, Style and The Principles of Screenwriting. Regan Books, 1st edition (December 17, 1997).

6. Orkin, J. "Constraining Autonomous Character Behavior with Human Concepts". In S. Rabin (Ed.), IA Game Programming Wisdom 2

7. Ortony, A., Clore, G.L., Collins, A. "The Cognitive Structure of Emotions", Cambridge University Press, 1988

8. Parker. "Cluedo. The Classic Detective Game", 2004.

9. Turner, S.R. Minstrel: A Computer Model of Creativity and Storytelling. Report of the Computer Science Department, University of California. Los Angeles, CA, 1992.

10. Westwood Studios. "Blade Runner", 1997.