ICAPS 2006

The English Lake District, Cumbria, UK

WS4

Workshop on AI Planning for Computer Games and Synthetic Characters

Ruth Aylett
MACS, Heriot-Watt University, UK

Michael Young
Liquid Narrative Group,
University of North Carolina, USA














ICAPS 2006

The English Lake District, Cumbria, UK

Workshop on AI Planning for Computer Games and Synthetic Characters

WS4

Ruth Aylett
MACS, Heriot-Watt University, UK

Michael Young
Liquid Narrative Group,
University of North Carolina, USA

ICAPS 2006
Workshop on AI Planning for Computer Games and Synthetic Characters

Table of contents

ICAPS 2006
Workshop on AI Planning for Computer Games and Synthetic Characters

Preface

1. The scope of the workshop

The application of AI technologies in general to computer games and graphical characters is an expanding research field, as witness the growth of regular AAAI Spring Symposia sessions into the first in a series of international conferences on AI and Interactive Digital Media in June 2005, the growth of the conference Intelligent Virtual Agents (IVA), and the development of sometimes affectively-driven autonomous synthetic characters in projects in Europe, the US and Asia. At the same time, AI is being seen as a source of commercial edge by Computer Games companies, with expansion in the range of AI technologies being applied and the introduction of AI-related sessions at the Games Developers' Conference, and new tutorials on websites such as Gamasutra. Games engines have also become a standard tool for groups of AI researchers, with initiatives such as Gamebots [Kaminka et al 02]helping to make such engines more versatile and open to researchers in agent architectures.

However so far - with some notable exceptions discussed below - AI planning researchers have not been deeply involved in this work, with two unfortunate outcomes. One is a tendency for games researchers without specific experience in AI planning to reinvent wheels or use somewhat outdated approaches. The second is for planning researchers themselves to focus on techniques which are inherently less useful for these domains and to misunderstand the specific requirements of computer games and synthetic characters.

In this workshop we hope to initiate greater interaction between AI planning research and the interactive graphical environments used both for games and for more serious educational and training purposes.

2. What has been done?

The work that has been carried out so far in the use of planning for synthetic characters and in computer games can be divided into a number of different areas. The first is the sue of planning as an action-selection mechanism for intelligent characters in interactive graphical environments, not necessarily for games: often for education and training applications. Pioneering work here was carried out by Gratch [Gratch 00], whose Emile continuous planner combined AI planning technology with an agent architecture in which emotions were used to control the direction of planning, and was applied in the military training application Mission rehearsal Exercise [Gratch et al 01]. The FearNot! planner [Aylett et al 06] reported on in ther main conference is a development of this work, this tome applied to anti-bullying education for children.

Rather than use generative planning, Cavazza [Cavazza et al 01] provided intelligent graphical characters in a story environment based on friends with already-expanded plans, in the form of AND-OR trees, which were searched at execution time. These could produce a variety of narrative events, partly due to random siting of characters at initialisation, and partly to the ability of the user to remove resources needed by a character forcing them to replan by searching elsewhere in the tree. Work by Young [Riedl & Young 04] has applied generative technology not to the action-selection mechanisms of intelligent characters, but in the service of the user as a method of authoring narratives. Here planning is a way of maintaining logical coherence in a wide space of

story possibilities.

Laird is one of the best known proponents of computer games as a testbed for AI technologies in general [Laird & van Lent 01]. He equipped a games bot with planning specifically as a way of predicting what a user would do in order to counter it. This use of a theory of mind - the bot was considering what the user's plan would be in order to take advantage of it - can also be thought of as a specialised version fo the counter-planning used in two-person opponent games. It can also be applied in non-character based games, and the bridge-bidder of Smith [Smith et al 98] was a particularly successful example of this.

Finally, commercial games are beginning to move beyond the widespread use of A-based motion planning into more general applications of planning technology. The game F.E.A.R is a good example here, containing a simplified STRIPS-type planner.*

3. The role of planning

Two different types of questions are at issue in a workshop such as this. The first derives from the application areas themselves, and asks what planning can add that other technologies do not already. A games company might legitimately suggest that unless AI planning improves gameplay, it is better not included because of the extra level of risk and complexity it brings.

An interesting question here, already posed of AI technologies in general, is whether added 'intelligence' is necessarily a useful thing in a game. More intelligent actions may improve a game - the example of bridge bidding is appropriate here - but on the other hand it may not. For example, more intelligent opponents in games based on combat will not amuse the user if they are impossible to kill as a result. It appears that gameplay improvement is related to dramatic qualities such a character believability rather than to intelligence per se. It is hoped that part of the workshop will consider this question in some detail in relation to specific games.

A second question looks at the problem from the technology side: which of the technologies associated with AI planning are specifically useful for synthetic characters and computer games? Again, examining specific games can help to answer this type of question, but some generic responses do seem clear.

The first is the requirement for interactivity - batch planning of the type practised in the planning competition does not seem especially appropriate. These are domains in which interactivity is usually a basic requirement since they are organised around a user that expects an environment responsive to their actions. Continuous planning, and the interleaving of planning and execution therefore seem much more relevant than their current rather low profile in overall research suggests.

Integration is another requirement that is not always well-served by the AI planning research community. Continuous planning supposes integration with perception and execution, but in general AI planning in these domains must also interface to 3D real-time graphics and to physically-based modelling. In general, planning must sit within a much larger architecture, for which in the case of characters, goal management, emotional modelling and personality may be as important as planning competence.

4. The way forward?

At this stage, when only an embryonic group of researchers interested in applying planning to this domain exists, it seems important to consider both the barriers to developing this area of research and what needs to be done to remove them.

One barrier that this workshop might hope to start to attack is the lack of knowledge among planning researchers about the domains themselves. This produces the risk of working with inaccurate or outdated stereotypes, especially with respect to the computer games industry, which can change very quickly indeed. It also seems to be true that

combining planning with 3D interactive graphics is inherently inter-disciplinary - with an understanding of the graphics acting as the execution part of integrated systems rather essential but usually lacking amongst planning researchers.

A second barrier may lie in the fact that many of the popular research areas of AI planning technology is not all that useful for these domains in their current form. This may be one reason for a limited take-up by games practitioners so far. An event such as this is one rather small step in trying to encourage the development of more applicable planning technology, as well as to generally educate ourselves on the requirements of these domains and identify the key problems and tasks for the specific research agenda.

The Planning Competition has served the overall AI planning research community well for this purpose, but its thrust does not seem very appropriate for planning for synthetic characters or computer games for the reasons just discussed. Perhaps then a task for a workshop such as this is also to talk over whether there is a need for unifying examples or testbeds, for example based on a specific games engine and set of scenarios.

References

Aylett, R.S, Dias, J and Paiva, A. (2006) An affectively-driven planner for synthetic characters. Proceedings, ICAPS 2006

Cavazza M, Charles F, Mead S. (2001) Characters in search of an author: AI-based Virtual Storytelling. 2001; IVA 2001: 156-170

Fasciano, M (1996) Everyday-world Plan Use, Tech Report TR-96-07, Computer Science Department, U. of Chicago (1996),

Gratch J. (2000) Emile: Marshalling Passions in Training and Education. In 4th International Conference on Autonomous Agents, ACM Press, June 2000

Gratch, J; Rickel, J; & Marsalla, S. (2001) Tears and Fears, 5th International Conference on Autonomous Agents, pp113-118 2001

Kaminka, G.A; M. M. Veloso, S. Schaffer, C. Sollitto, R. Adobbati, A. N. Marshal, A. Scholer, and S. Tejada.(2002) GameBots: The ever-challenging multi-agent research test-bed. Communications of the ACM, Jan 2002.

Laird. J.(2000) It Knows What You're Going To Do: Adding Anticipation to a Quakebot, Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, Technical Report SS-00-02, AAAI Press, 2000.

Laird, J & van Lent, M (2001), Human-Level AI's Killer Application: Interactive Computer Games, AI Magazine 22(2), Summer Issue (2001)

Riedl, M & R. Michael Young (2004) An intent-driven planner for multi-agent story generation. Proceedings, 3rd International Conference on Autonomous Agents and Multi Agent Systems, July 2004.

Smith, S. J. J.; Nau, D. S.; and Throop, T. A. (1998). Computer bridge - a big win for AI planning. AI Magazine 19(2):93-106.

Organizers

- *Ruth Aylett, MACS, Heriot-Watt University*
- *Michael Young, Liquid Narrative Group,University of North Carolina*

Programme Committee

- *Marc Cavazza, Teeside University, UK*
- *Carlos Delgado-Mata, University of Bonaterra, Mexico*
- *Joao Dias, INESC-ID, Portugal*
- *Nick Hawes, Birmingham University, UK*
- *Eric Jacopin, CREC Saint-Cyr, France*
- *Gal Kaminka, Bar Ilan University, Israel*
- *Brian Magerko, Michigan State University,US*
- *Hector Munoz-Avila, Lehigh University, US*
- *Alexander Nareyek, CEO/CTO, Digital Drama Studios*
- *Jeff Orkin, MIT Media Lab, US*
- *Paolo Traverso, ITC/IRST (Italy)*
- *Mark Riedl, University of Southern California, US*
- *Jesus Savage, UNAM, Mexico*

Body Mind and Emotion

An Overview of Agent Implementation in Mainstream Computer Games

Stuart Slater, Kevan Buckley, Kamal Bechkoum

School of Computing and Information technology
Wolverhampton University
s.i.slater@wlv.ac.uk
k.a.buckley@wlv.ac.uk
k.bechkoum@wlv.ac.uk

Abstract

As both the growing demand for more immersive games continues and a widening of the buying audience increases, there is a growth in the quality and quantity of research and development in the field of "believable" computer game characters. This believability is necessary for the continued improvement in realism for computer games [Smith et al 2002] and is a key factor in an ongoing drive by developers who are creating greater immersive experiences for the gamer with each new generation of game. The purpose of this paper is to look at the elements that game developers use to give the agents in their games, human like qualities.

Introduction

Artificial Intelligence (AI) is a broad term, which for this paper applies to all work involved in making agents behave in a more challenging and believable way. This is often achieved through an agent's behaviour/actions appearing more human-like, thus creating the illusion that the gamer is playing against a real human player [Saltzman 2000].

During the early 1980's agent believability was not needed in the arcade style games being developed. Instead developers concentrated on implementing only enough simple behaviour to enhance game play. This behaviour was driven by simplistic algorithms many of which were implemented to allow simple path planning such as the ghost's movement in the *Pac Man* game or simple finite state machines (FSM's) for controlling agent actions. These were traditionally implemented in very small game environments such as single screen or tile based games and required almost no agent planning of goals but instead were more akin to scripted behaviour. Later games utilised more advanced AI techniques such as A* algorithms [Higgins 2002] which are used for agents path planning simply because virtual worlds have become considerably more complex. These techniques were implemented with limited processing resources and without regard for creating agents that can either exhibit human behaviour or look real. By the late 1990's games such as 'first person shooter' (FPS) style games were becoming more popular and as a consequence of consumer expectations in graphics, sound and AI the development became considerably more complex as the requirement for agents that could interact with each other and the player increased. This created a requirement for agents to be able to both look and behave more realistically within the very tight constraints of both developer technical skill and processor limitations. This was especially true when more realistic agents meant implementing new technologies such as realistic sensory systems, planning and more human-like agent physiology.

Agent Appearance

One of the key areas for improved agent believability has evolved from a rapid evolution of graphic processing technologies. This has resulted in higher polygon throughput and incorporation of new technologies such as vertex and pixel shaders, capable of giving surfaces a more "photo realistic look". Using pixel shaders, developers are getting closer to giving agents' skin a human like appearance without the need for complex programming. This reduction in complexity is due to the complementary programming methods and languages including NVIDIA's Cg [Fernando et al 2003] and Microsoft's High Level Shader Language (HLSL) [Fosner 2003]. These offer developers a more simplistic method of shader programming through a "scripting style" language that allows the graphic artists to implement the technologies as well as the programmers. The increase in polygon throughput has itself led to a greater dependence on packages such as *3D Studio Max 8* [Autodesk 2005a] and *MAYA 7* [Autodesk 2005b], which are used to create models and animations for cutting edge games such as *Doom 3* [Id 2004] and *Half Life 2* [Valve 2005]

Agent Physiology

With an evolving realism in the outward appearance of game characters there has been a growing amount of commercial development in game agent physiology such as inverse kinematics [Scarowicz 2004] and "ragdoll" simulation [Karma 2005]. Inverse kinematics include techniques to allow more realistic limb movements in agents, these have been coupled with graphical techniques to allow a smoother transition between actions by agents. "Ragdoll" is a term for the growing area of physics application to agent physiology such as allowing agents to

fall and move in a realistic human way.

Top selling titles such as *Half Life 2* combine both the advances in graphics, physiology and physics modeling to infuse game agents with human like movements and appearances. Animals seemingly jump at players and agents fall realistically when shot with extreme weapons. This has provided the gamer with a new level of realism and a higher level of interaction expected with each new generation of game.

Agent Decision Making

With clear visual improvements of agents, the games industry has also experienced an increase in the allocation of resources and research being carried out within project teams. This is supported by evidence from the roundtable moderator's reports for the last six years of the game developer conferences [GDC 1999-2005] that show up to around 10% CPU usage in 1999 rising to between 15 and 50% for a majority of development teams in 2003. This increase parallels the rise of many of the FPS games and strategy games such as Microsoft's Age Of Empires and Westwood Studios'Command and Conquer Games.

An area of AI that has benefited from an increase in development time is agent path-planning, as almost all agent goal-based behaviour relies on agents being able to choose between paths that may lead to the same end goal. This area of AI allows agents to traverse virtual worlds using techniques such as A* and waypoint navigation systems and have been implemented in increasing complexity for many years in some format from *Pac Man* to *Doom 3*. These navigational systems have been implemented with both cognitive modeling and goal-based reasoning, giving the agents the ability to navigate around a virtual world with a purpose, such as the goal of looking for food, as can be demonstrated with the use of the *Renderware AI* tool [Renderware 2005] or the goal of collecting weapons to fight against a human opponent.

Recent AI research has focused in two developing areas, which can be used in conjunction with path planning techniques to create more believable characters. These are sensory input processing and increased agent autonomy through methods such as real-time agent planning [Orkin 2006].

Sensory Input

A growing area of interest in both academia and industry is in the field of agent sensory systems. This has led to methods of providing agents with both the ability to "see" and "hear" items in their environments, but a serious dilemma exists for its implementation, due to the differing goals of industry and academia. Industry chooses to implement only enough technology to provide an element of game play due to processing limitations and is thus only concerned with "emulation and not simulation" [Leonard 2003] academic research on the other hand tends to focus in-depth on areas of interest. Remembering that the vast majority of commercial AI implementations use "smoke and mirrors" techniques there is a definite need to be able to scale down academic research findings so they are applicable within the processing and game play constraints of projects.

Current implementations of agent sight and hearing provide compromises between tweaking virtual worlds so that in some games the scenery broadcasts to the agents as in the *Sims* games [Orkin 2002] and/or the sensory input is driven by either polled or interrupt driven perceptions in order to limit processing load [Kirby 2002]. Therefore if an agent needs to eat, then the agent will actively seek out food using some form of planning, which blends navigation and either pre-scripted behaviour or real-time decision making. Whilst navigating, polled perceptions will provide information about the environments to the agent's sensory system on a continuous basis, if the agent is stationary and a player gets within its sensory range then an interrupt drive perception will feed the information to the agent, this gives a much less processor intensive form of implementation. This research has led to a greater scope in game play such as the ability to sneak behind enemies in games such as *Thief* [Eidos 2002], which added sophisticated auditory and visual senses to agents in the game [Leonard 2003]. *Half Life 2* and *Thief* have agents that can "seemingly" see and hear human players as they wander around their virtual world. This has meant that for the first time agents can be made aware of human players based on similar constraints to those of real human hearing or sight, or at the very least the first steps in simulating these sensory systems. The usage of sensory systems in recent games such as *Far Cry* [UBI 2004] have been used to allow not only elements of stealth, but the ability to distract agents by throwing rocks near them or sneaking past them whilst their backs are turned.

Emerging areas of interest are inter-agent communications as seen in the "walkie talkies" in *Far Cry* or cries for help from agents in *World of Warcraft* [Blizzard 2005] allowing agents to get support from other agents. Similar techniques have been used in *Call of Duty* [Activision 2003], when bullets are fired near enemy agents this causes a change of behaviour that allows the agent to dive for cover allowing human controlled game characters to advance forward to capture areas or attack.

With the implementation of agent sensory systems has come a greater use of data storage for the sensory input for agents. These storage systems are linked to agent decision making through back-end management systems and have created a new area of research in fast data access and storage mechanisms such as spatial data structures [Reynolds 2000]. These new systems and technologies are crucial to real time considerations in modern computer games.

Implementations of senses such as touch and smell though

not presently adding much to the game play are beginning to appear as in the use of smell for agents in *Half Life 2* which allows agents in the game to "sense" a player through an artificial olfactory system.

Decision Making

As mentioned previously, decision making has focused on mainly goal based reasoning and path planning. This allows agents, such as those seen in *Unreal Tournament 2004* [Atari 2004] to work co-operatively or adversarial with or against the player. Agents are given goals, such as "kill gamer", supplemented with inter-agent communication frameworks that provides co-operative team play with other agents or the gamer.

Agent decision making traditionally focuses on the use of finite state machines (FSM) which are based on deterministic programming i.e. if-then production rules that are commonly used for controlling agents' behaviour in games [Carlisle 2002].

An example is

"IF player in view AND gun loaded then FIRE"

The limitations of FSM's for computer games is that they rely on conditional *true* or *false* variables resulting in actions that are predictable and could be perceived as being limited. To enhance FSM's the Boolean variables can be replaced with fuzzy variables that have a much larger range of values. This allows for a more complex set of actions i.e.

"IF player in sensory range AND gun has enough bullets THEN fire weapon ELSE look for ammunition"

which would mean that an agent might not fire the gun if they do not have enough bullets to kill the player, thus they might go instead to retrieve ammunition. This allows developers to expand agent goal options by using linguistic rules to define behaviour in conjunction with tiered goal systems involving primary and sub goals to allow a breaking down of complex tasks [Waverren 2001]. This tiered approach to tasks enhances the behaviour options of agents to offer them choices depending on the primary task set. A difficulty with giving agents sub tasks is that if the agent's option path is highly varied, then a situation may arise where conflicting goals will need to be carefully managed to avoid a gridlocked agent response.

Making more adversarial players is not the only way that AI has been improved. Games such as *Creatures* [CyberLife 1996] blend techniques such as neural networks, and aspects of biochemistry to create agents with unique behaviours that can interact and mutate into new agents with unique behaviours [Stern 1999].

Many developers have serious concerns about agents that could try and move beyond the constraints of the game architecture if they exhibited unpredictable behaviour. Therefore developers rely on a more scripted-behaviour approach to avoid any kind of adverse emergent behaviour. Another concern of games developers is that there are serious concerns with AI adversely affecting game play due to both the processor time required and the speed of the response. This may be due to the software waiting for the next agent action or simply overly complex AI that interrupts the player's immersion. In some cases this has led to developer's reducing agent capabilities such as in *Ultima Online* [Stern 1999]. Techniques to decrease system load and responses times for complex AI include AI handlers running as separate threads and/or using level of detail AI architectures dependant on situation and processing availability [Woodcock 2003].

Agent Emotions

Commercial implementations of intelligent agents continue to provide a reasonable challenge to the gamer, but most games still lack any implementation of agent emotion's and thus agents can appear devoid of emotion, which could appear to the gamer that the agent is lifeless and shallow. Agents in commercial games currently cannot get annoyed by failed goals, show satisfaction for a kill against a tough enemy, or run away in fear.

Some developer's script facial animations to appear on agents' faces at intervals, to give the gamer the illusion of agent emotion i.e. when a player is killed by an agent the facial emotions might show a smile. The game *Halo* [Bungie 2003] featured simple finite state machines for the emotions surprise, anger and awe and upon activation of a particular state the agent would flee in terror, go berserk and attack, or retreat into a defensive position, this was complemented with suitable facial animations.

Conclusions

Recent games featuring large-scale environments such as *World of Warcraft* have virtual worlds filled with hundreds of agents and millions of gamers across multiple servers. The agents in these worlds need to appear to behave as realistically as possible to provide a satisfactory degree of immersion to the gamer and therefore are programmed with some awareness of their environments and a level of autonomy. These agents seemingly make decisions on goals given to them by the developer and thus appear to act in a similar way to a real player. Some games even implement "planning models" such as in F.E.A.R [Orkin 2006] to allow agents to choose options in real-time, based on their goals.

Currently agents can be equipped with sophisticated sensory systems such as sight and sound (hearing) and are able to traverse virtual worlds. They can plan, choose between options to complete goals and thus create the illusion of autonomy. The impact on the developer for implementing these new technologies is the increased processing and resource implications for this new generation of games. This has meant that developers have

been forced to look at ways of optimising games in such a way as to maintain a consistent game experience for the user whilst creating ever evolving virtual agents for the gamer to interact with. In part this has been achieved through a greater use of more intelligent environments, some of which can broadcast information about useful items in "view" to the agents, such as those found in the *Sims* games.

This evolving of both environmental design and AI architectures is necessary for the continued development of more sophisticated behavioural models for agents, in order that these agents are able to affect and interact with the player and their environments [Todd, P et al 1997]. These optimisation techniques along with a growth in visual improvements of agents is currently leading developers to look at other areas of improving agent believability. One area of interest is in the use of human emotion modeling.

If agents can be developed with more sophisticated cognitive and goal architectures then it is feasible that they could exhibit a level of emotions such as the universal emotions [Damasio 1999], fear, happiness, sadness, anger, surprise and disgust. These emotions could be linked to the agents' goal structure [Johnson-Laird 1989]:

- Happiness that a goal has been achieved or progress made.
- Sadness if a goal not completed or loss of a goal.
- Anger if goal challenged or failed because of an external entity
- Disgust if a goal is violated.
- Anxiety if a goal or the goal of self-preservation is threatened by a future event.

Plus

- Fear if an immediate goal is in danger or the goal of self- preservation is immediately threatened.
- Surprise – unexpected successful completion of a goal.

This may initially be limited to both a blend of facial [Ekman 2004], physical i.e. posture, voice and emotional state storage, but could subsequently be expanded to actually affect primary goals and the choosing of sub goals that might best satisfy any emotional needs. This could be the agent experiencing rage at a player killing his comrade and then choosing to punish the player by killing his teammate rather than the player. This of course would open up the possibility that an agent may become unpredictable, but would certainly offer the gamer a much more human like opponent.

Looking ahead there are several hurdles to overcome in implementing agent emotions amongst which are:

1. Constraining agents to virtual world architectures thus their cognitive model and domain knowledge would need to be carefully developed which might require extensive resources and time.
2. Limitations due to the current state of the art in agent decision making that may not be advanced enough for the realization of complex emotional architectures.
3. A possible side effect of adding 'human like' emotions to agents is that this unpredictability could be hard to replicate for testing and potentially memory and processor intensive.
4. The defining of a suitable emotion architecture that can adequately model human emotion within the constraints of developer resources and current state of the art in AI games technologies.

Web Reference

[Activision 2003] Call of Duty, 2003. Activision. www.callofduty.com

[Atari 2004] Unreal Tournament 2004. Atari. www.unrealtournament.com

[Autodesk 2005a] 3D Studio Max 8. www.discreet.com

[Autodesk 2005b] Maya 7. Autodesk formerly Alias. www.alias.com

[Blizzard 2005] World of Warcraft, 2005. Blizzard. www.worldofwarcraft.com

[Bungie 2003] Halo . Bungie Studios. http://www.microsoft.com/games/halo/default.asp

[CyberLife 1996] Creatures, CyberLife Technologies,1996.

[Eidos 2002] Thief. Eidos Interactive 2002.http://www.eidos.co.uk/games/info.html?gmid=34

[GDC 2001-2005] archive section of the Games Developers conferences 2001-2005. www.gdconf.com

[Id 2004] Doom 3, Id Software, 2004. www.doom3.com

[Karma 2005] Karma Physics Engine http://udn.epicgames.com/Two/IntroToKarma

[Renderware 2005] Renderware A.I game engine http://www.renderware.com/

[UBI 2004] Far Cry 2004. UBI soft. www.ubisoft.com

[Valve 2005] Half Life 2, Valve Software, 2005. www.valvesoftware.com

Reference

Carlisle, P. 2002.*Designing a GUI Tool to Aid in the Development of Finite State Machines* (pg 71-77), AI Game Programming Wisdom: Charles River Media.

Damasio, A. 1999. *The Feeling of What Happens.*. Published by Vintage.

Ekman,P. 2004. *Emotions Revealed.* Published in the UK by Phoenix

Fernando, R; Kilgard, M.J. 2003. *The Cg Tutorial*. Addisson Wesley.

Fosner,R. 2003. *Real-Time Shader Programming*. Morgan Kaufmann Publishers.

Funge, J.D. 1999. *AI for Games & Animation – A Cognitive Modelling Approach*. AK PETERS LTD. Natick, Massachusetts.

Higgins,D. 2002.*How to achieve Lightning Fast A**. AI Game programming Wisdom. Charles River Media.
Johnson-Laird, P.N. & Oatley,K. (1989). The language of emotions: An analysis of a semantic field. Cognition & Emotion, 3, 81-123.
Kirby, N. 2002 *Moderator's report – AI Roundtables*. Game Developers Conference 2002. San Jose, California.
Leonard, T .2003. *Building AI Sensory Systems – Lessons from Thief and Half-Life*. Game Developers Conference 2003. San Jose, California.
Orkin, J. 2002. *Tips from the Trenches*, AI Game programming Wisdom. Charles River Media.
Orkin, J. 2006. *Three States and a Plan: The A.I. of Fear*. Game Developer's Conference 2006. San Jose, California.
Reynolds, C. 2000. *Interaction with Groups of Autonomous Characters*, in the proceedings of the Game Developer Conference 2000, San Francisco, California.
Saltzman, M.1999. Secrets of the Sages. Chapter 9. Macmilan Digital Publishing USA.
Scarowicz,A. 2004. *Artificial Intelligence for Animated Autonomous Agents*. PhD Thesis Kingston University.
Smith, P; Hjelstrom, G. 2002. *Polygon soup for the programmer's soul: 3D Pathfinding*. Game Developers Conference 2002. San Jose. California.
Stern, A .1999. *AI Beyond Computer Games*. (www.lggwg.com/wolff/aicg99/stern.html
Todd, P.;Geoffrey F. 1997. How Cognition shapes cognitive evolution. IEEE EXPERT: Intelligent Systems and their applications, 12(4), 7-9.
Waverren, JMP.2001. *The Quake 3 Arena Bot*. University of Technology, Delft Faculty, ITS
Woodcock, S. 2003. *AI Roundtables Moderators' Report*. Game Developers Conference 2003. San Jose, California.

Biography

Stuart Slater is a senior lecturer in IT & Computing at Wolverhampton University. His current research involves the development of an "agent emotion architecture" for use with commercial AI solutions.

Planning for an AI based virtual agents game*

Susana Fernández, Roberto Adarve, Miguel Pérez, Martín Rybarczyk and **Daniel Borrajo**
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. Leganés (Madrid). Spain†

Abstract

Computer games have become a big software industry. From the early days of the use of AI to solve classical games, such as chess or checkers, we are now seeing an intensive use of graphics to attract gamers. Currently, AI in games, normally refers to either the designing the behaviour of stock AI agents, like Bots (automated player characters), or refers to custom AI agents such as Non-Player Characters (NPCs). We present here our on-going work on building a game, AI-LIVE, that is oriented towards the intensive use of AI controlled Bots. The game borrows the idea from the popular THE SIMS, but with a strong focus on building characters based on different AI techniques. More particularly, we present the work on applying planning techniques for building one such agent.

Introduction

Games has always been a challenging domain for testing AI techniques. In the beginning of AI the focus was on classical games, such as chess (Newell, Simon, & Shaw 1972) or checkers (Samuel 1963). The type of AI-based techniques that were used for solving these games were mainly search and, sometimes, machine learning. Then, in the 80's and 90's the work on these classical games continued by intensive use of faster machines with more memory, such as the work on Deep Blue (Hsu *et al.* 1990) or CHINOOK (Schaeffer *et al.* 1996). Recently, video games have produced a renewed interest from the AI community on applying its techniques into games. They normally refer to automated players (Bots), either opponents or teammates, and NPCs. Bots are agents that act as if controlled by a human player. They are stock AI characters that will follow designed behaviours. Non-Player Characters are any artificial agent that is not a player, like Monsters that act only as an enemy to all players. A notable example of AI in games is the project called FEAR, which stands for Flexible Embodied Animat aRchitecture.[1] This is a framework for creating AI controlled systems for synthetic characters. The project includes reusable AI components, a portable framework, and interfaces to realtime 3D games. A similar system is Arianne, although it is not designed specifically for AI.[2] Arianne is a multiplayer online engine to develop turn based and real time games providing a simple way of creating the game server rules and game clients. There have also been several tasks in games that have been solved using a variety of AI techniques. Examples are production systems for QUAKE (van Lent *et al.* 1999), planning for bridge (Smith, Nau, & Throop 1998) or Real Time Strategy games (Chung, Buro, & Shaeffer 2005) and in Full Spectrum Command,[3] or genetic approaches in Blondie24 (Fogel 2001) (see (Rabin 2002) for some reported work). Perhaps, the most used technique has been different versions of the A* algorithm for path-planning purposes.

The idea inspiring AI-LIVE has been the commercial game THE SIMS[4] and how it could be generalised and modularised so that AI clients could be designed to play a game together with humans. Our goal is to build an architecture similar to the one proposed in (Buro & Furtak 2004) for Real Time Strategy games. Each AI client is developed as an architecture using one AI technique. So far, we have built a rule-based AI client and a planner one in AI-LIVE. However, it could be augmented to incorporate any other planner or AI technique. Furthermore, all agents can interact, leading to a more complex system that integrates social and psychological models in order to obtain believable emergent behaviours, as the work by Silverman (Pelechano *et al.* 2005).

*This work has been partially supported by the Spanish MCyT under project TIC2002-04146-C05-05, MEC project TIN2005-08945-C06-05, and CAM-UC3M project UC3M-INF-05-016.

†The use of planning for this domain appeared in the discussions carried out within the Artificial Intelligence Standards Committee of the International Game Developers Association, Working Group on Goal Oriented Planning. We would like to thank its members.

[1] http://sourceforge.net/projects/fear
[2] https://sourceforge.net/projects/arianne
[3] http://www.ict.usc.edu/content/view/56/108
[4] http://en.wikipedia.org/wiki/The_Sims

AI-LIVE architecture

AI-LIVE is a client/server application running over TCP/IP. It works similarly to modern online games such as Ragnarok Online,[5] World of Warcraft[6] or Guild Wars[7], where various users connect to a central server to play in a shared world , with a key difference: in our case, human clients share the game with AI controlled characters using a variety of AI-oriented techniques, playing in the same world. At this point in development, a basic universe has been implemented consisting of a simple room with objects to pick up, together with two different AI clients and a graphics renderer. Figure 1 shows a high level view of the architecture.

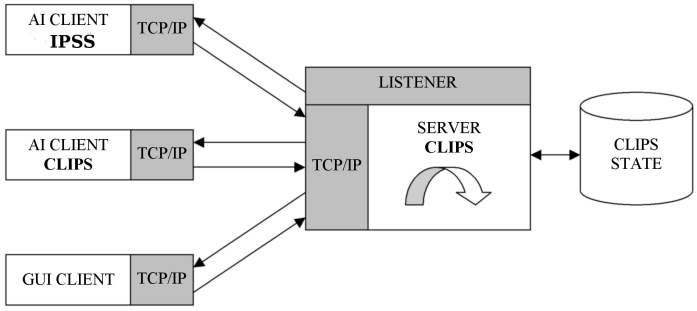

Figure 1: High level view of AI-LIVE architecture.

The server holds the state of the universe, which is divided in separate stages or realms where clients play. Each stage is made of objects in a cell-based 3D space. These objects are instances of classes from an ontology that is shared among all clients. In the future, clients will be able to travel from one stage to another, as players do in online games.

AI clients connect to the server to control an actor object in a specific stage each, while GUI clients connect to the server to open a window and display a graphical representation of a particular stage. In the future, GUI-based clients will allow human players to play and interact with AI clients.

The greyed application modules in Figure 1 are responsible for communication between both ends. All these modules are written in C. The server intelligence is done using the CLIPS tool for building knowledge-based systems.

After the server is initialised, it listens for incoming connections and runs a round-robin loop over the list of connected clients (we will define in the future asynchronous behaviour):

- Each AI client receives the CLIPS state corresponding to its stage from the AI server and gets a turn with unlimited time to decide on what to do. All other AI clients are paused as this happens. When the AI client is done, the server receives an action, which is passed to the AI engine and executed.
- Clients get all their turns during the game in strict order of connection.
- All GUI clients receive the CLIPS state corresponding to their stages from the AI servers in every turn the server executes, together with each AI client.

Clients connect to the server using a simple binary network protocol with little overhead, and at this stage of the project, they synchronise with the server simply by waiting for incoming data. Currently, there are two AI clients implemented, using CLIPS and IPSS (Rodríguez-Moreno *et al.* 2004) (an integrated planner and scheduler based on PRODIGY (Veloso *et al.* 1995)) respectively, as well as a 2D GUI client.

Given that we would like AI-LIVE to grow in the future and in order to be flexible, we have defined an ontology that is shared by all modules, that is described in the next subsection. Next subsections describe in more detail the ontology, the server and the CLIPS clients, while next section describes the IPSS clients.

The ontology

Figure 2 shows AI-LIVE class model. All objects in AI-LIVE universe are instances of one of these classes. A set of basic physical properties define all entities, and actors add personality properties and relationships with other entities (actors or not).

The main classes in the ontology are:

- `Stage`: it represents the different stages where the game can take place.
- `Entity`: abstract class to represent any possible entity in the stage. An stage is a collection of entities. There are four entity subclasses:
  - `Actor`: it represents game actors together with their personality and emotions. Now, we are not reasoning about this type of knowledge, but we would like to focus on it in the next future.
  - `Wall`: it represents walls that cannot be traversed.
  - `Object`: it represents any general object.
  - `ContentCapability`: it represents objects with capacity properties as a `Container` or an `Actor`.
- `Cell`: it represents the atomic space portions inside a stage.
- `Relationship`: it represents relationships among actors and objects.
- `ClientAction`: each action supported in the game has an associated class. So far, they are `MoveAction`, `PickUpAction`, `PutDownAction` and `AddClient`.

[5] http://en.wikipedia.org/wiki/Ragnarok_Online
[6] http://www.worldofwarcraft.com
[7] http://www.guildwars.com/

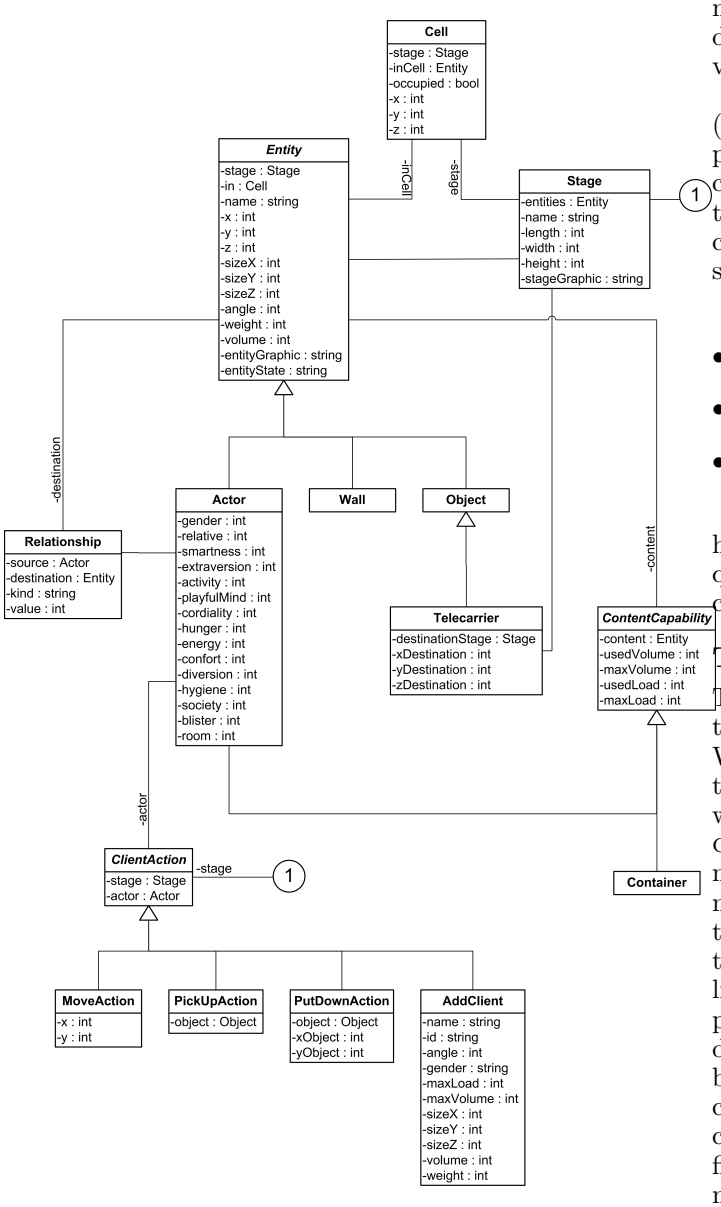

Figure 2: AI-LIVE ontology.

The server

The server is the central part of the game. It holds the state of the game with all of its defined stages and objects. It is in charge of maintaining a list of connected clients to serve states to and receive actions from in a round-robin basis. These actions are verified and executed against the state, producing a new state for the next client. The server is written in C language embedding the CLIPS production system to control the state, verify and execute rules.

When a client of any kind connects to the server (which is listening for incoming connections), both parts will identify. On success, the server will add the client to the loop. In this loop, all AI clients get one turn in strict order of connection, while GUI clients receive a copy of the state in every turn, but they do not send an action.

The actions currently supported by AI-LIVE are:

- Move an actor from one position to another
- Pick up an object as requested by an actor
- Put down an object as requested by an actor

To execute these actions, the CLIPS code in the server has a series of rules, that check for validity of the requested operations and alter the state. As the state changes, clients actions change.

The CLIPS client

The CLIPS client is an AI-LIVE client implemented using the CLIPS tool for building knowledge-based systems. We have integrated CLIPS with the rest of client code that handles, as in the case of the IPSS client, all networking operations. At this stage of the project, each CLIPS client playing the game will have to pick up as many objects as possible, considering its actor maximum capacity. As in the case of the IPSS clients, setting up the goals of the AI characters is, perhaps, one of the most interesting parts of the project that we would like to work on. Given that IPSS is a backward chaining planner and CLIPS works in forward mode, there will be differences in how goals will be defined and pursued in both types of clients. In the case of CLIPS clients, they can afford having a more reactive behaviour, while IPSS clients will have to be goal oriented (in its simplest configuration). We are also considering adding other planners that might have a different behaviour, introducing new ways of looking at this problem.

To decide on which action to run, the CLIPS client will use the stage received from the server as state, matching every object and property (such as class, position, size, angle, weight, bulk...) with conditions of rules.

The GUI client

The first graphical client for AI-LIVE is a conventional 2D sprite-based renderer named CREND. It is modelled after successful 2D engines still in use by some of today

video-games, and is written from scratch in pure C, using the portable SDL low-level multimedia framework for graphics.

CREND connects to an AI-LIVE server and obtains the state for a stage as an AI client would, but it does not send any action. Instead, it draws a graphical representation of the stage in a window where users can see what is happening in the server. Objects are drawn in a particular parallel perspective derived from traditional 2D scrolling, and used by most 2D role-playing games such as Tales of Phantasia.

The IPSS client

The IPSS client is the one that uses planning technology for deciding at each step which action is best to execute. It uses the IPSS system that integrates planning and scheduling (Rodríguez-Moreno *et al.* 2004). One can use any other planner for implementing this planning step, though there some features of IPSS that have been useful for building these clients (some of them will be described later). The client is divided in two modules:

- Main module: its tasks are to connect to the server, deal with the network, and invoke the AI module. This module is in charge of the low-level networking, receiving each state and interacting with the AI module to obtain an action, which is then forwarded to the server. It is written in C. The client can be configured with the following parameters: the server address and port; the stage the client is to play in; and the controlled actor profile.
- AI module: its main task is to decide the actor actions. The planning tool chosen for this task is the domain-independent planner IPSS. It is used to find a plan for the AI-LIVE domain and problem supplied by the main module. The main module receives the state with all the objects for the current stage from the server and parses it, translating it into IPSS description language (quite similar to PDDL2.1 (Fox & Long 2002)). The parsed state and goals form the input problem to the planner. The client domain is defined as a set of operators designed to be counterparts of the set of implemented actions the server can execute. At this stage of the project, the AI-LIVE actions are supported as operators: move, pick-up and put-down. These two last operators use the individual capacity constrains of characters. Then, IPSS provides a potential plan to achieve the goals from the current state, and outputs only the first operator in the plan. This operator is translated back to AI-LIVE actions scheme, and returned to the server.

Now, we will comment on two specific issues that have to be considered for applying planning to game playing in general and how we have solved them within AI-LIVE: how goals are generated, and how to solve efficiently the problem of selecting paths to go from one place to another in a given map, how to integrate path planning with task planning.

The first issue concerns selecting a goal to work on. Currently, AI-LIVE selects to maximize the number of objects taken by the actor, so goals consist on having the actor the objects that are in the room. Traditional approaches to planning assume that goals are given as input to the planners. However, we believe that, from a planning perspective, setting up the goal of an actor in this type of domains is precisely one of the key challenges of using AI planning here. Therefore, we want to study different types of goals generation schemes for this type of games. A related problem is the oversubscription in planning that occurs when agents do not have enough resources to achieve all of their goals. This requires finding plans that satisfy only a subset of them (van den Briel, Sánchez, & Kambhampati 2004).

The second issue relates to the use of grids/maps in planning domains (that appear in most games), as it is the case of AI-LIVE. For many planners, reasoning about how to go from one place to another can easily make planning intractable, as it is the case of IPSS. This is specially true if we want to optimize the cost of the path to go from one place to another according to a quality measure. There are many articles in AI games about applying heuristics to path planning. However, planning domains pose the added problem of path planning integration with the operator definitions. For example, to define an action for moving a synthetic character is necessary to know how the character can move in the world. We have used two approaches. The first approach, which is based on a careful manual knowledge engineering of the domain, exploits one of IPSS main features: the support for user-defined heuristics, to efficiently guide the search. These domain-dependent heuristics are defined as control rules (if-then structures), that help the planner taking directions, checking adjacency, and deciding on a position for an actor to pick up an object from the stage. Figure 3 shows an example of one of these rules for selecting bindings for the `move` operator. Suppose that the current planner goal is to have an actor on a `goal-cell` (the one that has an object which the actor wants to pick up, with coordinates `(x,y)`). If the actor is currently at another cell, this rule selects the best adjacent cell to the `goal-cell`, `origin-cell` (coordinates `(x1,y1)`), to which the actor should move first. This decision is needed given that IPSS is a backward chaining planner. The meta-predicate `(adjacent-p x y x1 y1)` is true if position `(x,y)` is adjacent to position `(x1,y1)`. The meta-predicate `(best-cell-p x y x1 y1 x2 y2)` is true if `(x1,y1)` is the best adjacent position for reaching `(x,y)` starting from `(x2,y2)` (where the actor is). Repeated use of this control rule guides the actor within the map directly from its initial position to the goal cell.

This approach for solving the path-planning problem within the task planning requires to define by hand the appropriate control rules. Obviously, this depends on the user that defines the right knowledge. The second approach integrates the task planner (IPSS) and a

```
(control-rule select-cell-for-MOVE
  (if (and (current-goal (cell-inCell <goal-cell> <actor_id>))
           (current-operator move)
           (true-in-state (cell-x <goal-cell> <x>))
           (true-in-state (cell-y <goal-cell> <y>))
           (true-in-state (cell-x <origin-cell> <x1>))
           (true-in-state (cell-y <origin-cell> <y1>))
           (adjacent-p <x> <y> <x1> <y1>)
           (true-in-state (cell-inCell <actor-cell> <actor_id>))
           (true-in-state (cell-x <actor-cell> <x2>))
           (true-in-state (cell-y <actor-cell> <y2>))
           (best-cell-p <x> <y> <x1> <y1> <x2> <y2>)
           (or (true-in-state (cell-occupied <origin-cell> false))
               (true-in-state (cell-inCell <origin-cell> <actor_id>)))))
  (then select bindings ((<cell_id> . <origin-cell>))))
```

Figure 3: Example of a hand crafted control rule for selecting bindings for the `move` operator.

path planner (a standard implementation of the A* algorithm), by interleaving their execution following the ideas in (Fox & Long 2001). When IPSS needs to find a path during the search for the task planning solution, it calls the path planner. If there is a path between the current position of the actor and the goal position, the path planner returns a solution (together with all its associated quality metrics), that can use that information while solving the problem.

In relation to the integration of task and path planning, IPSS has two useful features:

- Functions can be called within the definition of variables on operators. If we want to know whether there is a solution and its quality between two nodes of a path-planning problem, we can define a variable `distance` in the preconditions of the `move` operator whose value is the result of calling the path planner function.
- Different cost metrics can be defined at each operator. In this paper, we are mainly interested in the distance quality metric, but both planners (task and path planners) can also use other quality metrics, and obtain *good* solutions according to them.

For describing the overall planning problem we need two separate files. A problem file for the task planner, in which we have abstracted the information on the map/path graph, and a problem file for the path planner with information about the map/path graph. Nodes of the path graph will also appear in the task planner problem file so that there is a connection between these two processes. In the next subsections we describe this process in more detail.

Path planner

The input to the path planner is a path-planning problem composed of an initial node, a final node, a quality metric, and a graph. In order to automatically specify the graph for each problem given to the IPSS client, we extract it from the problem definition. Given that we wanted the approach to be as domain independent as possible, for each domain we only need to specify the problem predicates from which the system will create the graph nodes and the predicates from which the system will create the graph arcs. Each arc can have a set of quality metrics defined. In our experiments, we have only used one: distance. But in other applications, such as planning tourist visits in the SAMAP architecture (Arias, Sebastiá, & Borrajo 2005), we have used others such as price (cost of the transportation method), utility (a subjective value that can represent user/agent preferences such as *I prefer to use the bus when possible*), distance, and time. Then, when calling the path planner we can specify the cost metric to be minimized.[8] The output of the path planner is a list of move actions in the form of: initial node, final node, and the values of the cost metrics for that arc (distance).

Given that the task planner can call the path-planner for solving the same path planning problem many times during the search (due to symmetries in the task planning search tree), the path-planner provides a caching mechanism. Every solution for the path planning problems is stored the first time. Then, in case of solving the same path planning problem is needed, it retrieves the previous solution. This assumes that the graph does not change between two calls to the path-planner. In case it can change, such as domains in which agents can act on the arcs, the nodes, or their quality metric features, then the caching mechanism will not be useful. For instance, if the actor leaves objects in the floor when building a plan, it creates obstacles in the graph. Therefore, a previous solution to go from one place to another in a previous call of the path planner can be

[8] In the case of maximizing cost metrics, such as utility, we can always convert them to minimization problems.

made invalid.

This approach can be applied to many other domains, such as DRIVERLOG, SOKOBAN, ... The DRIVERLOG domain, for instance, has a graph for drivers and a graph for trucks. So, our implementation of the path planner admits a set of graphs to be defined. Then, for each domain operator, the appropriate path-planning problem file will be selected.

Task planner

IPSS uses a backward search to solve planning problems. The types of decisions that it makes during search are: goal to work on, operator name that can achieve the selected goal, bindings for that operator, and decide whether to continue subgoaling or execute an applicable operator. In order to integrate it with the path planner, we redefined the `move` operator as specified in Figure 4.

Variables appear between brackets. The preconds section defines first the variables that are used in the operator, as well as the operator preconditions. The effects section defines other variables that are used only in the effects, as well as the postconditions. Finally, the cost section defines new variables only used for the costs computation, and the operator costs depending on the quality metric used by the task planner when solving a specific problem. In this case, we have defined one quality metric: steps (distance). The `gen-from-pred` function accesses the current state to provide values for variables. For instance, the room (stage) where the actor currently is, or the position it is on.

The connection with the path planner is done through the `path-planning-distance` function whose input parameters are the initial node and end node, that will also appear in the path-planner graph description. It returns `false` in case of no solution, or a numeric value with the total distance to go from the initial node to the final node. This value is assigned to the <steps> variable.

Conclusions

We have presented the first steps of building AI-LIVE, a game inspired in the popular game THE SIMS. We have defined and built an architecture based on a server and three types of clients: a rule-based one, a planning-based one, and a GUI. From a planning perspective, we have defined and provided initial solutions to two planning problems related to this type of domains: goal selection and integration with path-planning. Both problems are common to the application of planning technology to many games.

Currently, we are dealing with a simplified domain in terms of actions covered, though we have defined the architecture to be easily augmented with many more actions. So, we are using an ontology that can cope with the knowledge needed for many of the potential new actions. For instance, in the next future, AI-LIVE is expected to shift towards the social relationships among actors and the reasoning about psychological aspects, that have already been considered in the ontology. We also want to change our 2D cell-based world representation to a full 3D with dynamic physical objects.

References

Arias, J.D.; Sebastiá, L.; and Borrajo, D. 2005. Using ontologies for planning tourist visits. In *Working notes of the ICAPS'05 Workshop on Role of Ontologies in Planning and Scheduling*, 52–59. Monterey, CA (EEUU): AAAI.

Buro, M., and Furtak, T. 2004. RTS games and real-time AI research. In *Proceedings of the Behavior Representation in Modeling and Simulation Conference (BRIMS)*, 51–58.

Chung, M.; Buro, M.; and Shaeffer, J. 2005. Monte carlo planning in RTS games. In Kendall, G., and Lucas, S., eds., *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG'05)*, 117–124. Essex (UK): IEEE.

Fogel, D.B. 2001. *Blondie24: Playing at the Edge of AI*. Morgan Kaufmann.

Fox, M., and Long, D. 2001. Hybrid stan: Identifying and managing combinatorial optimisation subproblems in planning. In *Proceedings of IJCAI'01*.

Fox, M., and Long, D. 2002. *PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains*. University of Durham, Durham (UK).

Hsu, F.; Anantharaman, T.; Campbell, M.; and Nowatzyk, A. 1990. *Computers, Chess, and Cognition*. Springer. chapter Deep Thought, 55–78.

Newell, A.; Simon, H.A.; and Shaw, J. 1972. *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Rabin, S., ed. 2002. *AI Game Programming Wisdom*. Charles River Media.

Rodríguez-Moreno, M.D.; Oddi, A.; Borrajo, D.; Cesta, A.; and Meziat, D. 2004. IPSS: A hybrid reasoner for planning and scheduling. In de Mántaras, R.L., and Saitta, L., eds., *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, 1065–1066. Valencia (Spain): IOS Press.

Samuel, A. 1963. Some studies in machine learning using the game of checkers. In Feigenbaum, E., and Feldman, J., eds., *Computers and Thought*. New York, NY: McGraw-Hill.

Schaeffer, J.; Lake, R.; Lu, P.; and Bryant, M. 1996. Chinook, the world man-machine checkers champion. *AI Magazine* 17(1):21–29.

Smith, S. J.; Nau, D.S.; and Throop, T.A. 1998. Computer bridge - a big win for AI planning. *AI Magazine* 19(2):93–106.

van Lent, M.; Laird, J.E.; Buckman, J.; Hartford, J.; Houchard, S.; Steinkraus, K.; and Tedrake, R. 1999. Intelligent agents in computer games. In *AAAI/IAAI*, 929–930.

```
(OPERATOR move
   (params <actor_id> <cell_id> <destination_cell_id>)
   (preconds
     ((<actor_id> ACTOR)
      (<stage_id> (and STAGE (gen-from-pred (stage-entities <stage_id> <actor_id>))))
      (<cell_id> (and CELL (gen-from-pred (cell-inCell <cell_id> <actor_id>))))
      (<destination_cell_id> (and CELL (diff <cell_id> <destination_cell_id>)))
      (<steps> (and STEPS (path-planning-distance <cell_id> <destination_cell_id> <steps>)))
      (<xx> (and COORDINATE (gen-from-pred (cell-x <destination_cell_id> <xx>))))
      (<yy> (and COORDINATE (gen-from-pred (cell-y <destination_cell_id> <yy>))))
      (<x> (and COORDINATE (gen-from-pred (cell-x <cell_id> <x>))))
      (<y> (and COORDINATE (gen-from-pred (cell-y <cell_id> <y>)))))
     (and (cell-inCell <cell_id> <actor_id>)
          (cell-occupied <destination_cell_id> false)))
    (effects ()
     ((del (cell-inCell <cell_id> <actor_id>))
      (del (cell-occupied <destination_cell_id> false))
      (del (cell-occupied <cell_id> true))
      (del (actor-x <actor_id> <x>))
      (del (actor-y <actor_id> <y>))
      (add (cell-inCell <destination_cell_id> <actor_id>))
      (add (cell-occupied <destination_cell_id> true))
      (add (cell-occupied <cell_id> false))
      (add (actor-x <actor_id> <xx>))
      (add (actor-y <actor_id> <yy>))))
    (costs ()
      ((steps <steps>))))
```

Figure 4: Example of the `move` operator for the AI-LIVE domain in the IPSS language.

Veloso, M.; Carbonell, J.; Pérez, A.; Borrajo, D.; Fink, E.; and Blythe, J. 1995. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical AI* 7:81–120.

Champandard, A.J. 2003. *Synthetic Creatures with Learning and Reactive Behaviors*. New Riders Games.

Pelechano, N.; O'Brien, K.; Silverman, B.; and Badler, N. 2005. Crowd simulation incorporating agent psychological models, roles and communication. In *First International Workshop on Crowd Simulation (V-CROWDS '05)*.

van den Briel, M.; Sánchez, R.; and Kambhampati, S. 2004. Over-Subcription in Planning: A Partial Satisfaction Problem. In *ICAPS Workshop on Integrating Planning into Scheduling*.

Author in the Loop: Using Mixed-Initiative Planning to Improve Interactive Narrative

James M. Thomas
Liquid Narrative Group
Department of Computer Science
Box 7535, NC State University
Raleigh, NC 27695 USA
jmthoma5@ncsu.edu

R. Michael Young
Liquid Narrative Group
Department of Computer Science
Box 7535, NC State University
Raleigh, NC 27695 USA
young@csc.ncsu.edu

Abstract

This paper describes a foundation for an interface to allow non-technical human authors to collaborate with an automated planning system to design interactive narrative. Drawing from research in advisable and mixed-initiative planning, a domain metatheory is presented that allows for qualitative elaborations of narrative domains. The authors describe a graphical user interface that exploits this metatheory to specify authorial preferences. Specific constructs related to interactive narrative are considered to demonstrate how the preferences of the human author may be used to define and control the possible user experiences of an interactive narrative.

Introduction

"Interactive narrative" describes the stories that develop within virtual worlds in which human users interact with one or more computer controlled agents. The most well known examples of interactive narrative are computer games, but also included are intelligent tutoring systems, embodied conversational agents, virtual environments, and training simulators. A persistent challenge for such systems is the narrative paradox: "how to reconcile the needs of the user who is now potentially a participant rather than a spectator with the idea of narrative coherence." (Aylett 2000).

Few systems attempt to reconcile these goals dynamically at run-time. Those favoring strong plot coherence often restrict the depth of the computer-controlled characters, and/or the human user's available palette of interactions with these characters, reducing character believability. Systems with interesting and believable characters often lack any automated mechanism to coerce these 'emergent' bots to meaningfully contribute to a story. Although many useful and commercially successful systems have been built with these limitations, none has yet fully met the goal of *Hamlet on the Holodeck* (Murray 1998, Cavazza et. al, 2000).

One approach for the balancing of these competing goals is the Mimesis system (Riedl, Saretto, and Young 2003). Their algorithm generates plans for actions of story world characters based on hierarchical task decompositions and discrete causal requirements. Although Mimesis simultaneously solves for plot coherence and character believability, the authors acknowledge (Riedl and Young 2004) that a primary limitation is the lack of a search space heuristic that would allow the system to judge the relative "goodness" of one plan over another. In other words, there is no mechanism to ensure that particular narrative qualities such as "suspense", "surprise" or "romance" will be produced in resulting plans.

One might attempt to define a generalized heuristic function in terms of universally accepted narrative ideals, but most planners lack a sufficiently powerful model to make associations between such generalized ideals and the semantics of a specific problem domain and plan space. Also, no set of heuristics has yet been identified that guarantees "good" narrative even when applied by skilled and motivated humans. As author Somerset Maugham quipped, "There are three rules for writing the novel. Unfortunately, no one knows what they are".

An alternative approach is to involve the human author in defining heuristic functions for each interactive narrative based on that author's preferences of setting and plot. For the system to capture these preferences and report them to the planner, it must have an integrated understanding of the definitions of actions and entities in the problem domain (the setting) and the effects that the constraints on those actions have in defining the topology of the plan space (plot experiences). A reasonable approach for gaining that understanding is to keep the author "in the loop" throughout the plan construction process. This paper describes the foundations for the design of such a collaborative authoring environment for interactive narrative. The first stage of this environment is being implemented as part of the Zócalo system of planning services at North Carolina State University (NCSU).

Planning For Interactive Narrative

Planning for interactive narrative offers special challenges and opportunities. Even in systems that do not attempt the dynamic generation of narrative structure, it is difficult to maintain clear knowledge or control of what can happen at run-time. Two examples from a sub-genre of narrative, massively multiplayer online games, illustrate this point. An early such game (1985) was Lucasfilm's Habitat. One of the first game-wide campaigns planned inside Habitat was a treasure hunt called the "D'nalsi Island Adventure". Habitat designers Morningstar and Farmer recall:

> It took us hours to design, weeks to build (including a 100-region island), and days to coordinate the actors involved. It was designed much like the puzzles in an adventure game. We thought it would occupy our players for days. In fact, the puzzle was solved in about 8 hours by a person who had figured out the critical clue in the first 15 minutes. Many of the players hadn't even had a chance to get into the game. The result was that one person had had a wonderful experience, dozens of others were left bewildered, and a huge investment in design and setup time had been consumed in an eyeblink. (Benedikt 1990)

This lack of predictability was not simply an artifact of the times. In January of 2006, Jeff Kaplan, a lead designer of World of Warcraft (the most popular on-line game to date at the time of the article with 5.5 million subscribers) was interviewed by the New York Times. In the interview he was asked how long it would take until the top boss in one realm of the game (Ahn'Qiraj) would be defeated:

> My estimates are in the one-to-two-month range, but my expectation is that it could happen today. I've learned that as soon as something is in the game, you have to expect that it's going to be beaten. (Shiesel, 2006)

In essence, the designers of this type of interactive narrative have given up on predicting the run-time possibilities of their work. Instead, these Game-Masters (Louchart and Aylett 2003) are kept busy developing the next episodic installment of narrative while tweaking its predecessor to accommodate the unforeseen actions of users.

Despite the difficulty of constructing shrink-wrapped games that solve the narrative paradox, designers continue to make the attempt. Marc Laidlaw, lead designer of the commercially successful, narrative-rich game Half Life 2 for Valve Entertainment, asserted the importance of plot in at the Austin Game Writer's Conference in late 2005:

> Game designers should be in love with plot. It's the engine of the story. It's the core tech, that little mathematical nub that everyone can point at and make the subject of proofs and axioms and corollaries. None of these technical considerations serve us very well when we start arguing about meaning, but if you get plot right, then meaning inevitably follows.
>
> I like to talk about plot because it can be directly implemented in a game. Plot is the sum of cause and effect. Whether the cause is something in the player's control, or something the designers force upon the player, it's measurable. It can map directly to gameplay decisions, and lead to an outcome or outcomes that are equally tangible.
>
> A thorough discussion of plot gives you everything you need to build your story, and your game. (Laidlaw 2005)

Automated planning manages "the sum of cause and effect" that defines plot. Beyond its desirability to a few risk-takers in the game industry, ensuring that a coherent narrative achieves particular goals is an important requirement for educational and training applications. The task for planning systems in interactive narrative reaches well beyond finding a single complete and consistent plan. Authors are interested in understanding how unplanned user actions may affect story goals. This in turn raises issues about the variability of narrative experiences that are possible with each construction and how those possibilities shift as authors make changes. Compared to other plan authors, those building interactive narrative are probably more likely to work with the planner to make incremental refinements to the planning problem through multiple iterations.

Including Domain Knowledge in Planning

Traditional automated planners are not designed specifically to facilitate iterative collaboration with the plan author. Research into collaborative planning methodologies has generally been referred to as advisable or mixed-initiative planning. Advisable planning (Myers 1996b) attempts to shape the behavior of the planner by adding additional information to the definition of the planning problem prior to invocation of the planners. Mixed-initiative planners allow for the iterative and incremental construction of the plan with both the user and the planner capable of proposing or initiating requests to change aspects of the problem or solution. Thus, advisable planning is effectively a special case of mixed-initiative planning where the initiative is first taken by the human, then by the planning system. "Configurable" planning (Nau 2005) is the combination of domain-independent planning engines with higher-level abstractions like hierarchical task networks that capture and exploit domain knowledge. Each of these research threads has application toward collaborative planning of interactive narrative.

Interactive Narrative as a Planning Domain

Much of the motivation for configurable planners is based on the gulf between the real world and restrictive experimental domains descended from "blocks world. Where Nau's "configurable" planners represent an architectural middle ground, interactive narrative represents a domain of similarly intermediate complexity between the "blocks world" and the real world.

Because interactive narrative takes place in a virtual world, its domains are both fully knowable and fully malleable. An advantage for planning research is that these domains may be amended or contracted to suit the requirements of the planning problem. In fact, the plan author may be responsible not only for the domain representation, but also may be involved in the construction of the domain itself. As interactive narrative planning is a component within this larger creative process, there are possibilities and requirements for experimentation and exploration than are not found working with real world domains. This affords researchers the freedom to investigate intricate relationships between the domain, its representation, the planning problem and the resulting plan spaces. Integrating these concepts into an authoring tool can benefit both the interactive narrative and the planning research communities.

Mixed-Initiative Planning Research

Mixed-initiative techniques have long been associated with several prominent planning research projects. Ferguson and Allen (1998) have studied extensively aspects of mixed-initiative in their TRIPS and TRAINS projects In their estimation "far more attention needs to be paid to the gap between the abilities of automated reasoners and the needs of human decision makers (Allen and Ferguson 2002). The systems Allen and Ferguson have built rely on human-computer interfaces based on natural human dialog. Their focus is on building a dialog system intermediary between the human plan author and group of back-end agents. A key challenge they have addressed is the mapping of individual communicative utterances of the user to the most appropriate plan editing action. They bias this intention recognition toward those candidates suggested by recency and those that will minimize plan churn. Another challenge they have addressed is the resolution of ambiguities about the scope of an intended change. Is the requested change to be performed on the problem goal or the proposed solution? Is the solution to be modified, extended or rejected? To perform these types of reasoning, the authors employ a collaborative interaction model compatible with the SharedPlans formalism of Grosz and Kraus (1996) and realized as an inter-agent communication protocol. The application of Allen and Ferguson's work to interactive narrative is limited by two factors. First, they rely on a domain representation assumed to be complete and accurate, where these are very much in flux during the authoring process of interactive narrative. Second, much of their focus is on the interpretation of spoken natural language statements about plans and plan goals in order to make the appropriate changes to the plans, where interactive narrative inputs are likely to be text with formally constrained syntax and semantics. A key contribution of their work that can help interactive narrative is modeling the problem solving state at multiple levels of abstraction, from a high-level hierarchy of objectives, to a compact summary of a class of possible concrete solutions, to the intermediate world states of particular solutions (Ferguson and Allen 2002).

Rich and Sidner (1998) also leverage discourse interpretation and SharedPlans in COLLAGEN. COLLAGEN, like TRIPS and TRAINS, is a few steps beyond the immediate challenges of authoring interactive narrative. COLLAGEN constrains search through a detailed model of interaction history. This includes intentional structure (partial SharedPlans), linguistic structure (hierarchical groupings of actions into segments), and attentional structure (a "focus stack" of segments). This model is used to generate context-dependent natural language formulations from which the user may choose. Rich and Sidner believe that, in contrast to the weakly structured interaction histories in most interactive systems, the interaction history in COLLAGEN "reflects the user's problem solving process". This idea of the system asserting informed choices of actions to the plan author could be used to guide the authors of interactive narrative toward decisions that have the best utility relative to their goals.

Tate, Dalton, Levine (1998) introduced the <I-N-OVA> Model (for Issues, Nodes, Orderings / Variables / Auxiliary) abstraction to allow for "plans to be manipulated and used separately from the environments in which they are generated." In Tate's system, the user and planning system work to refine the sets of constraints under which the planner must operate. Amant, et. al., (2001) have built mixed initiative interfaces for plan visualization and navigation. Blythe, et. al. (2001) have investigated representing plan structures in ways interpretable by humans as business processes. These systems focus on mapping plan representations to natural language correlates within the domain.

Advisable Planning

The idea of an advisable problem solving system goes back as far as John McCarthy's proposed "Advice Taker" program (McCarthy 1959). McCarthy's first example problem for a 'program with common sense' was a planning problem. Advice continued to have a prominent role in research into automating common sense including projects such as Cyc (Lenat 1995). Meanwhile, as the field of automated planning developed specialized knowledge representations and reasoning methods it became separated

from McCarthy's more general strain of commonsense reasoning work. However, Myers and her colleagues (Myers 1996b), have recently investigated the application of user-supplied advice within the context of modern planning techniques. Myers' advisable planner employs a model where abstract advice specifications provided by the user are compiled into a language of constraints common to traditional planning algorithms. Myers distinguishes between three "idioms" of advice. Task advice identifies the goals and actions to be included in solution. Strategic Advice recommends how goals and actions are to be accomplished relative to parameter values. "Evaluational" Advice puts constraints on some metric defined for the overall plan (e.g., resource usage, execution time or solution quality).

In Myers' work, the advice an author gives the planner is grounded in a domain metatheory, an abstract representation independent of underlying planning technologies. A domain metatheory is intended to enhance user directability of the planning process, aid in the generation of qualitatively different plans, and aid plan summarization. Myers (2000a) proposes a model built on three constructs: *roles*, *features*, and *measures*. *Roles* describe the function of an object within an operator, *features* are attributes that differentiate operators, and *measures* are partial orderings of features with respect to some criterion.

For example, the feature "**Air**" might be associated with the operator "AirMail(loc1, loc2, item)" while the feature "**Land**" might be associated with "BicycleMessage(loc1, loc2, item)". Related features may be grouped into feature categories, e.g., **Transport-Media** could be a category containing both **Air** and **Land**.

A *measure* is an ordering (possibly partial) of features within a feature category. For example, the measure **AFFORDABILITY** might be defined over feature category **Transport-Media** as to rank **Land** higher than **Air**, where the measure **COMFORT** might be defined over feature category **Transport-Media** as to rank **Air** over **Land**. A role-fill specifies explicit object instances or constraints over a set of instances relative to a given operator role.

Measures may be extended to describe object instances within the domain through the assignment of measure values. For example, if the measure **AFFORDABILITY** has measure values defined as (**Cheap**, **Moderate**, **Expensive**) the object instance *Lear Jet* would have the **AFFORDABILITY** measure value *Expensive* while the object instance *Subway* would have the **AFFORDABILITY** measure value *Cheap*.

Strategic advice is specified through the metatheoretic elements of Activities, Roles, Role-Fills and Measures, which in turn are simpler and closer to the natural cognitive models employed by human experts than the lower level planning constructs of goals, operators, variables and bindings. Strategic advice consists of prescriptions and restrictions of roles, fillers (a.k.a. role-fills), relative to specific activities. This advice takes two forms. Role Advice designates which object-role specifications (role-fills) are required or restricted in specific activities. For example, a role template of "Stay in <Accommodation> while vacationing in <Location> might be instantiated as "Stay in 3-star hotels while vacationing in Scotland" where the role of <Location> is filled by "Scotland" and the filler "3-star hotel" is prescribed for the role <Accommodation> (Myers 1996b). This is an example of an target activity with a feature of Vacation. In contrast, Method Advice operates at a higher level, as it prescribes or prohibits the use of specific activities within the plan.

Advice For Qualitative Differences

Once the planner becomes knowledgeable of the advice associations of its elements, it is possible to direct it toward solutions that have particular qualities relative to that advice. Many planners can generate different plans for the same problem, but extracting and summarizing the meaning of those differences is difficult. Furthermore, the particular differences of interest will vary from user to user and task to task. A deeper problem is the assumed accuracy and completeness of the domain and problem representations. Because much of Myers' work has been situated in the application of planning military operations for the real world, the domain representation is often seen to be incomplete or imperfect by the human experts who use the system because they have experience and knowledge over a vast number of real world exceptions. Therefore, considerable effort is devoted to eliciting more complete descriptions of the domain and problem representations.

A goal for advisable planning systems is that they create novel plans that are qualitatively different from one another (Myers and Lee 1999) a goal that is especially relevant for interactive narrative. To achieve this goal, the plan author nominates a subset of measures from the domain metatheory to serve as criteria for evaluating chosen properties of plans. Myers introduces an evaluation function that maps feature measures into categories on which measurements normalized over the interval [0, 1] can be applied. A set of k evaluation criteria thus define a k-dimensional space in which the Euclidean distances can be measured between the locations of each plan relative to each of these dimensions as measured by the evaluation functions. Myers' recent work (Myers 2005) uses the metatheory to summarize plan content, and uses a type hierarchy to reason about differences based on which objects are bound to different features of the plan.

As Myers moved toward a mixed-initiative model in which the user makes many of the decisions necessary to create the final plan, a new problem was introduced. At some

points in the creation of a complex plan there may be hundreds or thousands of unresolved issues. The system must rank these decisions based on importance so that the user has a chance to complete the plan. As many as five different methods for this type of prioritization were considered and three were implemented in a system called PASSAT (Wolverton 2004). The exploratory nature of interactive narrative construction is likely to produce similarly complex plan spaces. The prioritization methods pioneered in PASSAT would be useful in making optimal use of the finite attention of human narrative authors.

Domain Elaboration Framework

To leverage the results of advisable and mixed-initiative planning, this paper introduces DEF, the Domain Elaboration Framework. DEF is an adaptation of Myers' domain metatheory that allows authors to add detail to classical planning domains to enable expressive problem definition and reasoning about plans.

The basis of DEF is a STRIPS-stye (Fikes and Nilsson 1971) planning domain characterized by objects, conditions and operators. More formally, an object symbol provides a unique name for an entity in the world. All object instances are predefined by the plan author. A condition is a conjunction of function-free literals composed of a unique name identifying a relation and a set of placeholder variable terms or object instances. These terms are also referred to as condition parameters. An operator is defined by the set of literals stating the preconditions that must hold before it can be invoked, the set of literals stating the effects that will hold following its invocation, and a parameter list that may be applied to designate variables in these sets of literals.

Where the metatheory introduced by Myers relies on roles, role-fills, features, and measures, DEF uses an alternate grammar of **types**, **dimensions**, **weights**, and **measurements**. A type is a symbolic name of a node in a global hierarchy of author-defined types with a unique root node named "***anyThing***". Every operator, parameter, and object instance is required to have at least one associated type. Although type can be seen as an implicit concept in Myers' original metatheory. it is not until her recent work (Myers 2005) that one can find an explicit representation of type. In the example of the move operator whose *loc1* parameter was assigned the role of origin the type might be inferred to be location. It would seem obvious to a human author of the move operator that the *loc1* should only bind to objects of type location, but without explicit constraints a planner could just as easily fill the origin role with a cat, a cake, or a comb.

Because every parameter of an operator or condition and every object has a **type** associated with it, the type hierarchy can be used to guide the planner in assuring that the authorial intentions for bindings are maintained. In fact, an interactive narrative creation tool built on the DEF framework could communicate type constraints on parameters and objects through extending the set of preconditions for each operator and for the initial and goal states.

For example, in the case of the *loc1* parameter within the move operator the type constraint isalocation(*loc1*) could be added to the operator's list of preconditions, and when object instances of type location are created, corresponding isalocation(*newobject*) conditions could be added to the initial state of the planning problem. Some planners allow the nomination of a special subset of preconditions (sometimes referring to these as constraints) whose truth values can be computed directly from the assignments to the initial state, allowing for faster processing. For these planners, type constraints may actually help speed the plan search process by reducing the set of objects the planner must consider for bindings to parameters of operators and literals.

Types are also associated with operators, enabling the author to use a portion of the type hierarchy to encompass entities much like features in Myers' formulation. Every operator, parameter, and object instance has one or more associated types, and zero or more associated **measurements**. A **measurement** consists of a **dimension** and a **weight**. A **dimension** is a symbolic name selected from a global list of unique author-defined dimensions. A **weight** specifies a relative intensity of the dimension normalized on the interval [-1, 1]. The default weight '0' represents a neutral intensity, -1 is maximally negative and 1 is maximally positive.

Expressive Power of DEF

The **dimension** construct in DEF corresponds to the **measure** of Myers' metatheory. Both are symbolic values chosen from an author-defined list, e.g., *affordability, comfort,* or *magic*. A key difference is that where DEF uses numeric **weights** to gauge instances on each **dimension**, Myers uses **measure values** from a set of symbols that are defined for each feature category and ordered by the plan author for each **measure**.

For example, in DEF, object instance *Lear Jet* may have the **measurement** {*affordability*, -0.98}, where *affordability* is a **dimension**, and **weight** is near the minimal value of -1 on the scale [-1, 1].

With Myers' metatheory, object instance *Lear Jet* may be assigned the measure value *Expensive* for the measure *affordability*, from the ordered set of measure values {*Cheap*, *Moderate*, *Expensive}*.

At the operator level, DEF allows types and dimensions to describe operators in the same way they describe object instances. Myers uses **features** to describe operators at a

higher level of abstraction than DEF. The strategy chosen with DEF is to use a reduced low-level vocabulary to elaborate the problem domain description and defer their aggregation into more complex abstractions like **features** and **feature categories** to higher level user interfaces. Hopefully, this will allow for abstractions of arbitrary complexity at the interface level, while preserving an underlying representation that facilitates efficient reasoning about the qualities of individual plans and the qualitative differences between plans.

To make qualitative judgments about plans, Myers' measures are converted to proportionally distributed values over the interval [0, 1]. DEF requires explicit normalization of weights over an interval [-1, 1] (chosen to facilitate a default neutral weight of 0). Clearly, this shifts some responsibility to the interface to ensure that human authors assign weights with this normalization in mind. An interface using DEF can provide abstractions such as symbolic ranges, {*Cheap*, *Moderate*, *Expensive}* and convert these values to proportional internal representations. However, an interface is not precluded from allowing more precise or non-proportional numerical representations when appropriate.

An expressive advantage of DEF is that types and measurements are applicable to every operator, operator parameter, and condition parameter. Authorial goals are often articulated in terms of the types of actions contained in a story. The knowledge to support this type of reasoning can be represented through measurements applied to operators. Suppose, as in the film *The Princess Bride*, a young boy would like to make sure that the story does not contain too much kissing. *Kissing* could be introduced as a dimension and every operator associated with the act of kissing could be assigned measurements on the order of (*kissing*, .95). Other operators could have neutral values of 0, or negative values. *Kissing* could be selected as an evaluation criterion and the plans whose evaluation functions return low values of *kissing* could be favored.

Object instances could also have attributes that are directly derived from authorial goals. Perhaps the author would like to favor stories that contain a lot of enchanted objects. A dimension of "*magical*" could be created and applied with high levels to magic rings, scrolls, and potions, and low levels to chewing gum wrappers and socks. A planning heuristic that takes these measurements as an input can offer a high-degree of fidelity to discrimination between candidate plans.

Higher-level narrative constructs will necessitate the use of more complex representations. Suppose the author wants a 'happy' story. Is give-money(giving-player, receiving-player) a 'happy' action? It might be happy for receiving-player but not for giving-player depending on the state of the world. Using parameter-level measurements in DEF, a default positive measurement of happiness could be given to the receiving-player and a default negative measurement to the giving-player. Still, what if the measurement value of one parameter may depend on the bindings to other parameters in the same operator? For example, the giving-player might be happy to give money to her child, but unhappy to give the money to a thief.

One approach would be to recognize that these two situations describe actions that differ from the perspective of drama (mothering vs. mugging), even if they may have the same add and delete list from the perspective of classical planning. Thus, the action can be split into a give-moneyto-child action where ischildof(player2, player1) is added as a precondition, and a give-money-to-thief action where isthief(player2) is added as a precondition. Then the happiness of player1 can be assigned different values in each action. It may be possible for a tool using DEF could create cloned actions like these when the user indicates that happiness is a function of the sub-types of player1 and player2 and use a more compact representation to solicit and display such preferences. Still, the role the operator plays within the larger context of the plan may also significantly effect the user's evaluation of the plan. DEF merely serves as a starting point for reasoning about interactive narrative.

To summarize, DEF associates a set of one or more types and zero or more measurements with every operator, operator parameter, condition parameter, and object instance. DEF is a domain-independent representation intended to by leveraged by a user interface for use with any planner that can work with a STRIPS-style domain description.

Qualitative Reasoning With DEF

One motivation for DEF is to provide a general framework for elaborations of the plan author's preferences for the objects and actions in the domain over a variety of criteria. It is left to the planner and whatever interfaces are put between the human and the planner to make use of these preferences to influence plan reasoning. An evaluation function can be easily constructed from the **measurements** in DEF to apply the qualitative reasoning power of Myers' work to resulting plans, simply by transforming the **weights** from their interval [-1, 1] to the interval [0,1] employed by Myers' algorithms. This function can be used to iteratively refine and navigate through the plan space, or it can influence can be in the heuristics that are applied by the planner to direct search, perhaps in conjunction with other DEF constructs.

Another mechanism for reasoning about types recently introduced by Myers (2005) is also easily applicable in the DEF context. Myers defines a function MinSuperType(V) which finds the most specific super-type common to a set of elements V . This allows the author to characterize the differences between plans or parts of plans, through the five distinct set relationships that correspond to particular

subsets of their typed elements. Set arithmetic functions are described that help pinpoint key strategic differences between plans and show areas where plans are not as different as they might seem.

Incorporating DEF in a Planning System

While DEF supplies the raw materials for qualitative reasoning about plan structures, it requires an interface to allow non-technical authors to apply it to a planning system. This interface should represent the problem solving state at multiple levels of abstractions, similar to the four-layer model employed by Allen and Ferguson (2002). Their model allowed the user to move from high-level hierarchical objectives, through task structures that summarized classes of concrete solutions, to more primitive descriptions of particular plan fragments and world states.

Implementation of such an interface has begun with a program called Bowman, which is currently part of the Zócalo suite of planning tools available at NCSU at http://zocalo.csc.ncsu.edu. Bowman provides a GUI that allows authors to describe types, objects, operators, conditions and the initial and goal state of a planning problem. Bowman seamlessly passes an XML representation of the planning problem to a planning web-service to generate plans. The planner interface supports requests for the next N plans, planning for N seconds, or simply until a complete plan is found.

Bowman shows not only individual plans but also the entire plan space through scalable vector graphics (SVGs) that can be navigated through mouse clicks. Bowman can depict the plan space as a tree of nodes, where each node is a partial plan with zero or more plan flaws to be resolved. A plan flaw is an open precondition, a threatened causal link, or a flawed decomposition. Plan nodes with zero flaws are shown in green and plans with one or more flaws are shown in progressively more pale shades of yellow.

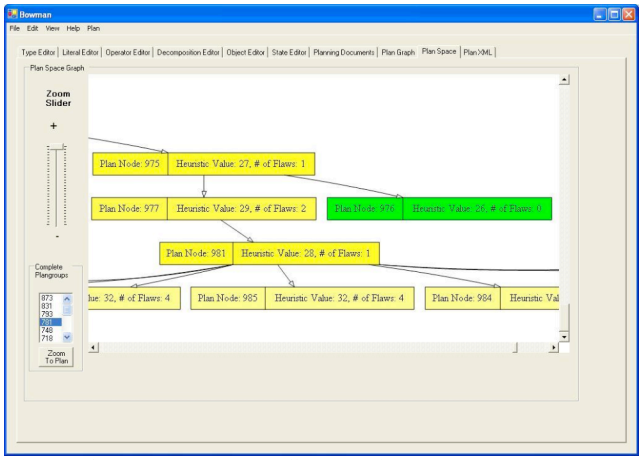

Figure 1 Bowman Plan Space View

The Bowman user can navigate the plan space and individual plans either through direct manipulation with the mouse or through dialog-based search.

Application to Interactive Narrative

Bowman can work with any planner that supports its straightforward XML representation of the planning and conforms to a simple web service interface. However, to provide support for interactive narrative domains, Bowman assumes a particular type of planner is being used. In its initial deployment, Bowman assumes that this is the Zócalo planner, based on Longbow, a decompositional (HTN) partial order causal link planner described by Young, Pollack and Moore (1994). Riedl, Saretto, and Young (2003) extended Longbow to support what they called *narrative mediation* to manage and respond to user actions in interactive narrative. Mediation policies are invoked in response to unplanned user actions that would threaten conditions in the world required for a planned future action. When such an action occurs, a mediation policy may nominate alternative actions called *failure modes* that may be substituted at run-time for the threatening user action.

To leverage narrative mediation, Bowman must be able to represent the particular mediation policies in effect for particular actions. Since failure modes are simply lists of operators, it would be possible to provide guidance on the subset of operators that are good candidates for failure modes through DEF constructs. In addition, DEF may be used to inform Bowman of abstract specifications of the characteristics of failure modes required to resolve plan bottlenecks. A first step is to use DEF to differentiate between types of agents.

Agent Types

For Bowman to be useful in addressing narrative mediation, it must contain in its representation of operators the types of agents that fill various roles. Thus a first step toward supporting narrative mediation is to distinguish between user-controlled agents and system-controlled agents (often called NPCs, or Non Player-controlled Characters). This distinction can be accomplished through a convention applied to the population of the type hierarchy of DEF. A subtree of the hierarchy can be fixed to contain "agent" and "inanimate". "Agent" can be subdivided into "NPC" and "User". A generalized mechanism can be realized in Bowman to allow the author to designate the subtree of the global type tree that is associated with the user (the default convention being "User" above) and that which is associated with NPC agents.

For interactive planning domains, the definition of operator in DEF is then extended to *require* specification of the type of agent that is capable of invoking the action ("User", "NPC", or the un-committed "Agent" as a default). Also,

individual object instances are extended to have a required agent type specification.

Mediation Strategies

As described in by Riedl, Saretto, and Young (2003) the planner is responsible for detecting user actions that could threaten the story plan. For each of these *exceptional* actions, the system must determine if changing part of the unexecuted portion of the plan can *accommodate* the action or if an *intervention* is required. An *intervention* requires that the requested action does not execute. Instead an instance of a non-threatening action, called a *failure mode* is substituted for the requested action in real-time. For example, if the user tries to shoot a character that is required to achieve a narrative goal later in the plan, a *failure mode* of "shoot-and-miss" or "jamming-shoot" might be substituted for the threatening "shoot(?gun, ?target, ?victim)" action. For this substitution to occur in real-time while as the user invokes the action, the intervention policies must be communicated to the story world system as soon as the initial story plan is created and whenever that story plan is altered.

How does the system know which actions can be invoked by the user? With DEF and Bowman, the author must explicitly specify the type of agent capable of invoking any particular action. Bowman provides this information to the planner, and also uses it to advise the human author of the set of actions for which failure modes are appropriate. For example, if the author at some point in the construction of the narrative has specified that the "shoot(?gun, ?target, ?victim)" action is one that can be invoked by an agent that can be a human user, Bowman can later take the initiative to advise the human author this action is contained in the set of actions for which failure modes may be useful. This indication could be provided prior to the invocation of the planner, or following invocation of the planner, after which the set may have been reduced by the set of actions for which the accommodation strategy proved sufficient. Bowman can further aid the human author in creating failure modes by reminding the author of the preconditions and effects of the underlying operators, highlighted with the planner's understanding of particular threatening effects if the planner has provided such guidance. Finally, the plan space depiction can be enhanced with explicit representation of the impact of mediation strategies.

Narrative "Macro" Libraries

Bowman allows for libraries of planning "macros" to be made available for plan authors. These sets of related literals and operators can assist plan authors to achieve typical planning goals with less total work. As a motivating example, consider the issue of inter-agent relationships and their effect on character believability.

Narrative systems have been built with rich emotional models of agents (Gratch and Marsella 2004). But in most commercial games the attitudes of NPCs are modeled with a single bit of memory – "friend" vs. "foe". Bowman proposes a modest extension of this model to enhance the believability of characters through three-valued attitudes defined by the narrative author. These attitudinal values can shift based on actions chosen by the agents in the world to enhance the component of character believability that is inferred by characters changing their behaviors based on social history. Examples of attitudes could be "ally/foe", "trust", "niceness", "sincerity", or "goal-directedness". Bowman predefines a set of literals that handle initialization of attitudes and transitions from one attitude to another. These attitude "macros" can be invoked through simpler constructs that the author manipulates. This may seem an overly shallow modeling of social relationships, but it does match up quite well with the emotional "bank account" metaphor pioneered by psychologists like George Bach and later popularized by Stephen Covey (1989).

For example, suppose the author would like to model the ally/foe attitude. Assume that we name this quality "*ally*". The following literals express possible initial states for the "*ally*" attitude between *?agent-a* and *?agent-b*:

- *attitude-negative*(*ally*(*?agent-a, ?agent-b*)) means that *?agent-a* is a foe of *?agent-b*
- *attitude-neutral*(*ally*(*?agent-a, ?agent-b*)) means that *?agent-a* is neither an ally nor a foe of *?agent-b*
- *attitude-positive*(*ally*(*?agent-a, ?agent-b*)) means that *?agent-a* is an ally of *?agent-b*

The pre-defined library of helper operators that support attitude maintenance are:

attitude-up-to-neutral(?attitude-name, ?a, ?b)

Preconditions:

attitude-negative(?attitude-name(?a, ?b))

incr(?attitude-name, ?a, ?b)

Effects:

attitude-neutral(?attitude-name(?a, ?b))

^incr(?attitude-name, ?a, ?b)

attitude-up-to-positive(?attitude-name, ?a, ?b)

Preconditions:

attitude- neutral (?attitude-name(?a, ?b))

incr(?attitude-name, ?a, ?b)

Effects:

attitude- positive (?attitude-name(?a, ?b))

^incr(?attitude-name, ?a, ?b)

A reciprocal set of "*attitude-down*" actions that reference the "*decr*" literal are also automatically added to the domain description. Then, the only thing the plan author needs to do is establish initial attitude values for relationships that need to be modeled and insert "*incr*" or "*decr*" attitude literals as the effects of actions that make sense in the world. The planner will find the appropriate "*attitude-up*" or "*attitude-down*" actions to insert in the plan to meet attitudinal goals specified later in the plan (if any).

For example, if the author intends for character *?agent-a* to become an ally of *?agent-b*, the author could add condition *attitude-neutral(?ally, ?agent-a, ?agent-b)* to the initial state, *attitude-positive(?ally, ?agent-a, ?agent-b*) to the goal state and create an operator like *join-up(?ally, ?agent-a, ?agent-b)* that has *incr(?ally, ?agent-a, ?agent-b)* as effect. A planner could add the *attitude-up-to-positive* operator to establish the *attitude-positive* goal condition. Then an operator like *join-up* would be selected to establish the *incr* condition.

Creating an inter-agent relationship library of macros is just one example of what might be done with a sufficiently general interface like Bowman. Bowman can serve as a test bed for assessing the usefulness of other macro-like libraries of related operators and literals.

Research Ambitions

DEF and Bowman can aid in narrative construction and qualitative reasoning about plans. At the point of this writing, Bowman works with the Zócalo planner to construct planning domains and planning problems and navigate through plan spaces and individual plans. However, only the "*type"* component of the DEF framework is realized in the current version of Bowman. To achieve the ambitions outlined in this paper, much work remains.

First, the full DEF framework must be implemented in Bowman, so that agent types, mediation strategies and narrative "macros" can be specified. Second, authors must be given an ability to articulate their narrative preferences through DEF constructs. Third, because authors are engaged in a creative process, it will be important for Bowman to provide a rich interaction history as in COLLAGEN (Rich and Sidner 1998) to allow exploration of alternate approaches. Plan summarization techniques will be needed to annotate the plan space with DEF constructs and increase the effectiveness of the "navigation" paradigm. The qualitative metrics proposed by Myers should overlay all these capabilities to provide a clear understanding of the differences between plans.

In such a system, the author could use "world manipulation sliders" like a sound engineer uses a mixing board to control the relative levels of different components in a production. In the case of interactive narrative, these components may be primitive constructs like DEF dimensions, or more complex features like the number of different execution paths or the amount of "conflict" or "happiness" in the story. A tight coupling with the planning system could allow the Bowman to opportunistically highlight areas of the plan that would best benefit from the attention of the human author. To conserve that attention, Bowman would benefit from an internal model of the author's intentions and apply them automatically to situations that are deemed similar. The result would be a system whose behavior is interesting enough to hold the interest of people who create interesting characters for a living.

Conclusion

This paper introduced a general planning domain metatheory called DEF and a general plan-authoring interface called Bowman, currently under development at North Carolina State University. These tools are being used to support author-preference realization in interactive narrative. As these tools grown into high-level interfaces accessible to non-technical authors, new avenues of planning research may become accessible as well.

Acknowledgements

The authors wish to thank the members of the Liquid Narrative Group at NC State University for their assistance in developing Zocalo. This is material is based upon work supported under a National Science Foundation Graduate Research Fellowship and National Science CAREER award 0092586.

References

Allen, J. F., and Ferguson, G. 2002. Human-Machine collaborative planning. *In Proceedings of the Third International Workshop on Planning and Scheduling for Space*.

Aylett, R.S. 2000. Emergent Narrative, Social Immersion And "Storification" *Proceedings, Narrative Interaction for Learning Environments*, Edinburgh.

Benedikt, M. ed. 1990. The Lessons Of Lucasfilm's Habitat. *In Cyberspace: First Steps*. MIT Press, Cambridge, Massachusetts, p. 288.

Blythe, J., Kim, J., Ramachandran, S., and Gil, Y. 2001. An Integrated Environment For Knowledge Acquisition. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces* (New York, NY, USA, 2001), ACM Press, pp. 13–20.

Cavazza, M., 2000. Interactive Storytelling In Virtual Environments: Building The Holodeck. presented at the 6

International Conference on Virtual Systems and Multimedia (VSMM), 2000, October 4-6, Gifu, Japan.

Covey, S. 1989. *The Seven Habits of Highly Effective People*. Simon & Schuster, New York, NY, USA, pp. 188-199.

Ferguson, G., and Allen, J. F. 1998. TRIPS: An Integrated Intelligent Problem-Solving Assistant. In *AAAI/IAAI* pp. 567–572.

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intellligence,* 2, pp. 189-208.

Gratch, J. and Marsella, A. 2004. A Domain-independent framework for modeling emotion, *Journal of Cognitive Systems Research,* Volume 5, Issue 4, pp. 269-306

Grosz, B. J., and Kraus, S. 1996. Collaborative Plans For Complex Group Action. *Artificial Intelligence* 86, 2, 269–357.

Lenat, D. B. 1995. CYC: A Large-Scale Investment In Knowledge Infrastructure. *Communications of the ACM* 38, 11, 33–38.

Louchart, S. and Aylett, R. 2003. Solving The Narrative Paradox In VEs - Lessons From Rpgs In *Intelligent Virtual Agents, 4th International Workshop IVA2003* eds T. Rist, R.Aylett, D.Ballin,

McCarthy, J. 1959. Programs With Common Sense. *In Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, Her Majesty's Stationary Office, pp. 75–91.

Murray, J. 1997. *Hamlet on the Holodeck: The Future of Narrative in Cyberspace,* MIT Press, Cambridge, Massachusettes.

Myers, K. Strategic Advice For Hierarchical Planners. 1996a. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR96)*, L. C. Aiello, J. Doyle, and S. C. Shapiro, Eds., Morgan Kaufmann Publishers, pp. 112–123.

Myers, K. 2000a. Domain Metatheories: Enabling User-Centric Planning. In *Proceedings of the AAAI Workshop on Representational Issues for Real-World Planning Systems*, K. Myers and Y. Gil, Eds.

Myers, 2000b. K. Planning With Conflicting Advice. In *Proceedings of the Fifth International Conference on AI Planning Systems* (AIPS 2000).

Myers, K., and Lee, T. J. 1999. Generating Qualitatively Different Plans Through Metatheoretic Biases. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence,* AAAI Press.

Myers, K. L. 1996b. Advisable Planning Systems. In *Advanced Planning Technology*, A. Tate, Ed. AAAI Press, Menlo Park, CA.

Myers, K. L. 2005. Metatheoretic Plan Summarization And Comparison. In *Proceedings of the ICAPS-05 Workshop on Mixed-initiative Planning and Scheduling* Monterey, CA.

Nau, D. S. 2005. May All Your Plans Succeed! *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, July 2005. Invited talk.

Rich, C., and Sidner, C. L. 1998. COLLAGEN: A Collaboration Manager For Software Interface Agents. *User Modeling and User-Adapted Interaction* 8, 3-4, 315–350.

Riedl, M., Saretto, C., and Young, R. M. 2003. Managing Interaction Between Users And Agents In A Multiagent Storytelling Environment. *In Proceedings of the Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-03).*

Riedl, M., and Young, R. M. 2004. An Intent-Driven Planner For Multi-Agent Story Generation. *In Proceedings of the Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04)*.

Riedl, M. O., and Amant, R. S. 2002. Toward Automated Exploration Of Interactive Systems. *In IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces* (New York, NY, USA, 2002), ACM Press, pp. 135–142.

Schiesel, Seth. 2006. "Kill the Big, Bad Dragon (Teamwork Required)." *New York Times*, 28 January 2006.

Tate, A., Dalton, J., and Levine, J. 1998. Generation Of Multiple Qualitatively Different Plan Options. *In AIPS*, pp. 27–35.

Tate, A. 1977. Generating Project Networks. In *Proceedings of IJCAI-77*, pp. 888-893.

Wolverton, M. 2004. Prioritizing Planning Decisions In Real-World Plan Authoring. In *Proceedings of the ICAPS04 Workshop on Connecting Planning Theory with Practice.*

Young, R.M., Pollack, M.E., and Moore, J.D. (1994). Decomposition and causality in partial-order planning. In *Proceedings of the Second International Conference on AI and Planning Systems*, 188-193, Chicago, IL, 1994.

Applying a Plan-Recognition / Plan-Generation Paradigm to Interactive Storytelling

Börje Karlsson, Angelo E. M. Ciarlini[1], Bruno Feijó, Antonio L. Furtado

PUC-Rio – Departamento de Informática, R. Marquês de São Vicente, 225, Rio de Janeiro, Brazil, 22453-900
[1]UniRio – Departamento de Informática Aplicada, Av Pasteur, 458 – Térreo, Rio de Janeiro, Brazil, 22290-240
borje@inf.puc-rio.br, angelo.ciarlini@uniriotec.br, bruno@inf.puc-rio.br, furtado@inf.puc-rio.br

Abstract

A key issue in interactive storytelling is how to generate stories which are, at the same time, interesting and coherent. On the one hand, it is desirable to provide means for the user to intervene in the story. But, on the other hand, it is necessary to guarantee that user intervention will not introduce events that violate the rules of the intended genre. This paper describes the usage of a plan recognition / plan generation paradigm in LOGTELL, a logic-based tool for the interactive generation and dramatization of stories. We focus on the specification of a formal logic model for events and characters' behaviour and on how the tool helps the interactive composition of plots through the adaptation of fully or partially generated plots. Based on the model, the user can interact with the tool at various levels, obtaining a variety of stories agreeable to individual tastes, within the imposed coherence requirements. The system alternates stages of goal inference, planning, plan recognition, user intervention and 3D visualization. Our experiments have shown that the system can be used not only for entertainment purposes but also, more generally, to help in the creation and adaptation of stories in conformity with a specified genre.

Introduction

In recent years, the convergence of games and filmmaking has been seen as an opportunity to create storytelling systems in which authors, audience, and virtual agents engage in a collaborative experience. The resulting systems can be useful for many different purposes, such as story board production, education and training, and, of course, entertainment. Different approaches have been proposed, using techniques and concepts from many areas such as Computer Graphics, Artificial Intelligence, Cognitive Science, Literature and Psychology. The suitability of each approach depends on the goal of each application.

A first decision to be made before implementing a storytelling system is whether it should be able to actually create stories or only enable the user to tell different stories based on previously computed sequences of actions. In the former case, the opportunities of interaction and the variety of different stories tend to be greater, but a coherent chaining of actions is more difficult to attain.

A second important point corresponds to the focus of the story models. The focus can be either on characters or on plots. In a character-based approach, the storyline usually results from the real-time interaction among virtual autonomous agents. The main advantage of a character-based model is the ability of anytime user intervention, which means that the user may interfere with the ongoing action of any character in the story, thereby altering the plot as it unfolds. Although powerful in terms of interaction, such an extreme interference level may lead the plot to unexpected situations or miss essential predefined events. Additionally, there is no guarantee that narratives emerging from the interaction of autonomous agents will be complex enough to create an interesting drama. By contrast, in plot-based models, characters should follow more rigid rules, specifying the intended plot structures. A fundamental inspiration for plot-based approaches has been the seminal work of Vladimir Propp in the field of literary theory (Propp 1968). Propp observed that significant events within a narrative of a given genre (in his case, Fairy Tales) can be associated with a fixed repertoire of functions, and that these occur in certain typical sequences. In a pure plot-based approach, user intervention might be more limited, but it is usually easier to guarantee coherence and a measure of dramatic power.

A third decision is whether stories should be told using a first- or a third-person viewpoint - cf. the notion of focalization in narratology studies (Bal 2002). First-person tends to be particularly suitable for applications closer to digital games, whereas third-person is more appropriate for those involving filmmaking.

Finally, it is necessary to choose between a *reactive* and a *deliberative* behaviour for the characters. In the first option efficiency is the main advantage, but modeling an intelligent behaviour is more complicated and the alternatives for the agents are somewhat limited. In the second, planning and reasoning techniques are usually applied to simulate an intelligent behaviour, but performance is often affected, especially if the story generation occurs at real-time.

LOGTELL is based on modeling and simulation. The idea behind LOGTELL is to try to express the basic structure of a genre through a temporal logic model, and then verify what kind of stories can be generated by

simulation, combined with user intervention. In this way, we focus not simply on different ways of telling preexisting stories, but on the dynamic creation of plots. The model includes *typical events* and *goal-inference rules*. Plots are generated by successive cycles of goal-inference, planning, plan recognition and user intervention.

Specifically, we try to conciliate both plot-based and character-based modeling. On the one hand, we borrowed from Propp's ideas, but tried to extend his rather informal notion of function. In our treatment, typical events are described by parameterized operations with pre-conditions and post-conditions, so that planning algorithms can be used for plot generation. On the other hand, the goal-inference rules model the behaviour of the various actors, thus providing some character-based features. The rules declaratively specify how situations can bring about new goals for each character.

Our objective is not to create an immersive experience in which the user takes part in the story as one of the characters. We endeavour, instead, to explore the possibilities of generating a large variety of coherent stories by means of a plan-recognition/plan-generation paradigm. For this reason, our stories are told with a third-person viewpoint. User intervention is always indirect. During the simulation, the user can intervene either passively, just letting the partially-generated plots that seem interesting to be continued, or, in a more active way, trying to force the occurrence of situations and events. These are rejected by the system whenever it finds no valid way to change the story to accommodate the intervention.

Plot dramatization can be activated for exhibiting the final as also the partially generated plots. For dramatization, characters are represented by actors in a 3D-world. During the performance of an event, low-level planning is used to detail the tasks involved in each event. We decided to implement our own graphical engine, so that we could better guarantee the compatibility between the logical model of our plots and the corresponding graphical dramatization.

The next section describes related work in the area of storytelling. Section 3 presents LOGTELL's overall architecture. Section 4 describes the main features of the Interactive Plot Generator (IPG), which is the kernel of the system. Section 5 illustrates how the user can interact with LOGTELL to generate stories. Section 6 shows how the generated plots are dramatized. Section 7 illustrates the use of the tool with an example. Section 8 contains concluding remarks.

Related Work

The approach adopted in the DEFACTO project (Sgouros 1999) uses successive evaluations of rules to control the generation of an interactive story where the user is the protagonist. The interaction among characters' goals is explicitly represented and an Aristotelian conception of plot is used to lead the story to a climax and then resolve it. The chaining of events, however, is not explained by pre- and post-conditions, making the control of what can and what cannot occur rather complex. Additionally, it does not allow the use of planning algorithms to develop sequences of events for the achievement of goals. The need of user intervention seems to be high if one wishes to generate a complete plot. Goals are inferred by means of rules analyzing the current situation, but the choice of actions to achieve goals appears to be more reactive than deliberative.

The approach described in (Cavazza, Charles, and Mead 2002) adopts a character-based model to make user interventions at any possible time. Characters are autonomous agents, executing plans to achieve their goals, and, from their interactions, it is expected that a narrative will eventually emerge. Users are spectators but can "physically" interact with the context and even advise characters, affecting their decisions and the resulting stories. In order to decide, at real-time, the actions to be performed, characters consult a Hierarchical Task Network (HTN), corresponding to pre-compiled plans. In this way, the system does not have to pay the price of using problem-solving planners while presenting a 3D animation. It might demand more effort to model the behaviour of the characters, but it makes sense if one does not consider maximizing the alternatives as a requirement. The main doubt about pure character-based approaches is to what extent dramatic and engaging narratives may actually result. The task seems to be easier with genres like sitcoms, wherein the climax of a story is not so clearly distinguishable.

The use of Propp's ideas in pure plot-based approaches leads to systems more concerned with the guidance of interactive stories than with their generation (Spierling et al. 2002). For each "Proppian" function within a story of a certain genre, such systems present alternatives to be chosen by the users. Still, we claim that to obtain an effective method to generate stories, it is necessary to extend Propp's ideas, adding semantics to the functions (and to their specializations), so that preconditions, effects and goals can be fully expressed.

Reference (Paiva, Machado, and Prada 2001) presents the Teatrix environment, where Propp's functions are used to model synthetic characters that interact with other characters, directed by children, in a virtual world. Each child directs one character and the synthetic characters are autonomous. All characters have a role in the story, specifying the functions in which they can take part. Synthetic characters have goals that change according to the situation. They plan and try to execute actions (i.e. functions) according to their roles. The approach seems interesting for education, but the control of the consistency of actions and goals and the generation of dramatic situations are not guaranteed. Additionally, the use of pre-defined plans in the planning process can enhance the performance, but might limit the amount of different stories that can be generated.

The interactive drama FAÇADE (Mateas and Stern 2000) is an effort to build an interactive system that

integrates characteristics of both plot-based and character-based approaches. A drama manager is responsible for maintaining the story state. Characters have autonomy most of the time, but their goals and their behaviour can be changed by the drama manager, in order to move the plot forward. The interactive story has the user as the protagonist. The drama manager automatically selects scenes to be played. Scenes are composed of beats, which define the granularity of the interaction between characters and plots. The user can directly interfere in the execution of a beat, determining how the rest of the scene will be played. The approach clearly separates higher-level goals, important for the story, from lower-level goals, more specific of the autonomous behaviour of the characters. Such separation can also be found in LOGTELL. The generation phase deals only with higher-level goals, which are essential for the creation of plots. Lower-level goals are assigned to actors when they have to dramatize an event. The main differences between LOGTELL and FAÇADE result from the objectives of each system. In FAÇADE, the focus is on letting the user experience a story from a first-person perspective. As a consequence, the interaction occurs at real-time, at the level of the beats. In LOGTELL, we focus on the generation of a maximum of different and coherent stories with a third-person viewpoint. The interaction basically occurs during the generation phase. The user is not allowed to interfere in the dramatization phase.

The Erasmatron system (Crawford 1999) is intended to support the authoring process of interactive stories. It tries to balance plot-based and character-based approaches by using the notions of verbs and sentences. Actions are represented by verbs with roles assigned to characters to form sentences. Such a proposal is close to the way we extended Propp's functions in LOGTELL. Functions are implemented as logical operations, with parameters, pre- and post-conditions.

The use of planning in (Riedl and Young 2004) to create plots has many similarities with the decisions made while implementing LOGTELL. In both approaches, a non-linear, least-commitment planner is used to create plots, conciliating actions of many different characters. The main difference is that LOGTELL does not assume the existence of one goal for the story as a whole. Instead, at the beginning of the story and after each planning phase, we use goal-inference rules (defined in a temporal modal logic) to consider new goals induced, for the various characters, by situations arising from the part of the plot so far generated. On the other hand, plans generated according to (Riedl and Young 2004) incorporate information explaining the intention of the actions, which can be useful to help in the dramatization of a plot, in particular to choose a convincing order of events. In LOGTELL, it is up to the user to choose a compatible total order of events to be dramatized.

The LOGTELL Architecture

LOGTELL comprises a number of distinct modules to provide support for generation, editing and visualization of interactive plots, as shown in Figure 1. The arrows represent the dataflow. The general architecture can be seen as a pipeline, where data is transformed from morphological functions into real-time 3D animations dramatized by virtual actors and handled by a graphical engine. Consequently, each module has specific input and output data.

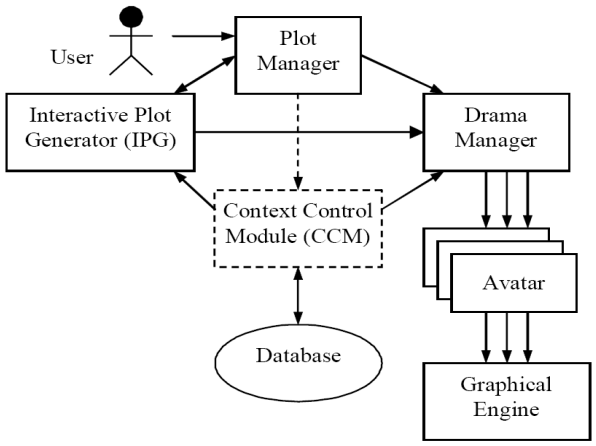

Figure 1: LOGTELL Architecture

The user interfaces with the system through the Plot Manager. The generation of plots by the Interactive Plot Generator (IPG) is started by the Plot Manager, which receives the partial plots generated so far and allows the user to intervene in the generation process. In order to visualize the dramatization of a plot (final or partial), the user chooses a total order of events, compatible with the partially ordered sequence generated by IPG, and asks the Plot Manager to activate the Drama Manager.

The Drama Manager is responsible for controlling the dramatization of the plot. In order to do that, it controls actors for each character in a 3D environment running on our game engine. During the dramatization, the Drama Manager consults IPG to keep the coherence between logical and graphical representations of the plot.

For the time being, the context of the stories to be generated and told is directly accessed by the modules and there is a certain replication of data. IPG uses files directly specifying the logical context in Prolog and the Drama Manager uses its own graphical and logical data. In order to eliminate compatibility problems, we are currently implementing the Context Control Module (CCM) to store all data in a single database. CCM will control the access to the data and format the data items to be used by the other modules. We are also extending our interface to help the user specify the context via the Plot Manager.

Plot Generation

IPG (Ciarlini 1999) semi-automatically generates plots of narratives of a specific genre. Narratives could be both of literary genres and of more mundane ones, such as the context of a business information system. In its use for entertainment, the focus is on checking the logical coherence of a genre and its characters and exploring the variety of stories that can be generated.

The context for the creation of stories comprises the following items:

- a set of facts (state), introducing the characters and their initial situation, as well as the description of the scenarios and other static features needed for the generation of stories;
- a set of logical rules, to infer goals to be pursued by each character, as certain situations arise in the course of plots; and
- a limited repertoire of pre-defined operations (typical of the chosen genre) in which characters can take part.

Examples of possible facts in a simple swords-and-dragons context, using a Prolog notation, are listed below:

- dragon('Draco').
- strength('Draco',45).
- affection('Brian','Marian',100).

The facts at a current state change as a consequence of the occurrence of events, which result from the execution of operations by the various characters. For each operation, the following data is supplied:

- a list of arguments, indicating the characters involved in the event, locations, etc.;
- a list of pre-conditions, specifying facts that should or should not hold prior to the execution of the operation;
- a list of post-conditions (effects), specifying facts that hold or cease to hold immediately after the execution of the operation;
- its representation, specifying details about the exhibition of an event caused by the operation.

An example of an operation in the fairy tale context is "kidnap", having a "villain" as agent and a "victim" as patient. Usual pre-conditions are that "the victim should presently be fragile" and that "both the victim and the villain should be present at the victim's current location". Post-conditions are that "the victim will be a captive of the villain" and "both the villain and the victim will be at the villain's home". The representation of events based on this operation would involve the specification of smaller-grain actions, such as: the villain getting closer to the victim, grasping the victim and taking him/her to the villain's home.

During the generation phase, plots are represented by partially-ordered sets of events. Partial rather than total ordering is a consequence of the use of non-linear planning during the simulation, establishing temporal constraints only when necessary, which makes the conciliation of goals easier. As a consequence, the truth of a fact at a certain time might depend on the final total order that will be chosen later. For instance, suppose there are two events without a predefined order between them: "the knight gets stronger" and "the knight fights the dragon". Depending on the order, the knight has different strength levels at the time he fights the dragon.

For each class of characters, there are goal-inference rules, specifying, in a temporal modal logic formalism (Ciarlini, Veloso, and Furtado 2000), the goals that the characters of the class will have when certain situations occur during a narrative. The rules use the following meta-predicates to speak about the occurrence of an event or the truth value of a literal (a fact or a negation of a fact) at certain times:

- h(T,LITERAL): LITERAL is necessarily true at time T;
- p(T,LITERAL): LITERAL is possibly true at time T;
- e(T,LITERAL): LITERAL is established at time T; and
- o(T,EVENT): EVENT occurred at time T.

In order to express constraints relating variables, there are two additional meta-predicates:

- h(CONSTRAINT): CONSTRAINT is necessarily true; and
- p(CONSTRAINT): CONSTRAINT is possibly true.

An example of goal-inference rule appropriate to the present context is: "when the victim becomes fragile, the villain will regard that as an opportunity and will have the goal of kidnapping the victim". Another possible rule is that "when the victim is kidnapped, the hero will feel motivated to free the victim". This last rule, is represented in our logic as follows:

$$\forall(\text{VIC},\text{T1},\text{VIL}) \wedge (\text{T1},\text{kidnapped}(\text{VIC},\text{VIL}) \rightarrow$$

$$\exists \text{T2}\ \text{h}(\text{T2},\text{not}(\text{kidnapped}(\text{VIC},\text{VIL}))) \wedge \text{h}(\text{T2}>\text{T1})$$

It is important to notice that the rules do not determine the specific reaction of a character. They only indicate goals to be pursued somehow. The events that will eventually achieve the goals are determined by the planning algorithm.

The generation of a plot starts by inferring goals of characters from the initial configuration. Given this initial input, the system uses a planner that inserts events in the plot in order to allow the characters to try to fulfill their goals. When the planner detects that all goals have been either achieved or abandoned, the first stage of the process is finished. The partial plot then generated is presented to the user by means of the Plot Manager and can optionally be dramatized. If the user does not like the partial plot, IPG can be asked to generate another alternative. If the user accepts the plot generated so far, the process continues by inferring new goals from the situations generated in the first stage. If new goals are inferred, the planner is activated again to fulfill them. The process alternates goal-inference, plan generation/recognition and user interference until the moment the user decides to stop or no new goal is inferred.

Notice that, in this process, we mix forward and backward reasoning. In the goal-inference phase, we adopt forward reasoning, so that situations in the past generate goals to be fulfilled in the future. In the planning phase, an event inserted in the plot for the achievement of a goal might have unsatisfied pre-conditions, to be handled

through backward reasoning. Also, to establish them before the event, the planner might insert previous events with further unfulfilled pre-conditions, and so on, recursively.

The user can also force the occurrence of events at certain times. For instance, the user could well insert "the wedding of the knight with the princess". It is also possible to specify that some situations should be true at certain times along the narrative, leaving to the system the job of planning the events that bring about such situations. It should be possible to say, for instance, that "the knight will be weaker than the dragon at a certain time". This kind of intervention is allowed both at the beginning of the process and at the pauses occurring between two simulation cycles. The planner tries to conciliate both inferred goals and user specified events and situations.

Our planning tool is a non-linear planner implemented in Prolog, adapted from (Yang, Tenenberg, and Woods 1996) with extensions. The use of a non-linear planner, as suggested before, seems more suitable because it uses a least-commitment strategy. Constraints (including the order of events) are established only when necessary, making easier the conciliation of various goals. Features to permit the abandonment of goals were included, and also constraint programming techniques for dealing with numerical pre-conditions.

Our plots are not restricted to incorporating only successful plans. In trying to provide adequate means for handling negative interactions happening along a plot, we realized that the solution of conflicts and competitions sometimes requires the presence of totally or partially failed plans, which conventional plan generators reject. When a goal is abandoned, events occurring prior to the moment of abandonment must be kept as part of the narrative, and thus influence its continuation.

We use two main mechanisms to handle goal abandonment and competitive plan execution: conditional goals and limited goals. A conditional goal has attached to it a survival condition, which the planner must check to determine whether the goal should still be pursued. Limited goals are those that are tried once only, and have an associated limit (expressed as a natural number). The limit restricts the number of new events that can be inserted to achieve the goal.

Composing by plan recognition

An alternative way to derive plans for goals is to take, from a conveniently structured library, a pre-existing typical plan, adapting it if necessary to specific circumstances. We have been using a structure for such libraries of typical plans that also allows plan-recognition by a method proposed by Kautz (Kautz 1991), and which has been implemented as a complementary feature of IPG. The method consists of matching observed events against the plan definitions (also called complex operations) stored in the library, trying to find one or more plans of which these events may be part.

A structured library with these typical plans (complex operations) is shown in Figure 2. Single arrows denote composition (part-of link) and double arrows denote generalization (is-a link).

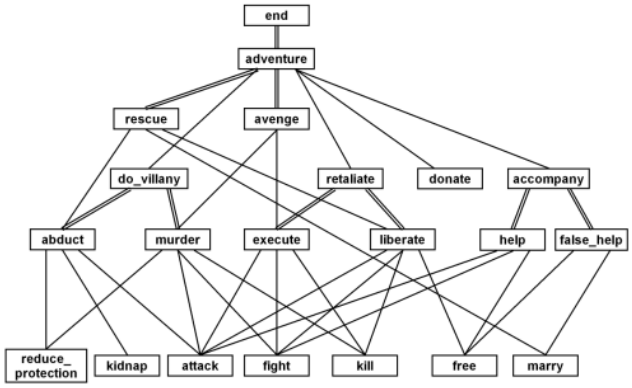

Figure 2: Typical plan hierarchy

These complex operations have the same syntax shown for (basic) operations, if the complex operation results from a composition of other possibly complex and/or basic operations, there will be two more parameters, respectively, a list of the component operations, and a list declaring any order requirements holding between them.

Complex operations formed by generalization are also represented, branching down to specialized operations corresponding to alternative ways to reach the same main effects; clauses is_a(<more-specialized-operation>,<more-general-operation>) declare this structural link.

The first step of the plan recognition algorithm is the generation of explanation graphs for the observed (or selected) events. An explanation graph for an event describes in which way this event can be used as part of some end-plan. After the graphs for all observed events are created, they are unified. The final graph will contain all the end-plans where the observed events fit.

Using this approach in LOGTELL, the user can select a group of events and request the possible complex operations that contain them. The system will then insert the complex operations components (if any) in the original plan. More details about this mode of interaction will be provided in the next section.

User Interaction

People who have no special talent for literary composition, like ourselves, find it difficult to invent interesting plots. Storytelling researchers (Glassner 2004) repeatedly point out that there may be problems when users participating in a game are prompted to function as "authors". But we usually do not feel so uncomfortable if asked to adapt an existing plot, by introducing small modifications in a gradual fashion.

The underlying philosophy of the system consists of providing the user with efficient means for exploring coherent alternatives that the story may allow at a given

state, and for guiding the plot at the level of events and characters' goals.

In the LOGTELL tool, the user has direct control only over the Plot Manager. This module, in turn, communicates with IPG to execute plot generation and enforce coherence, and with the Drama Manager to control plot visualization. The Plot Manager comprises the user graphical interface (implemented in Java), whereby the user can participate in the choice of the events that will figure in the plot and decide on their final sequence (Figure 3). Each event is represented by a rectangular box that may assume a specific color according to its current status.

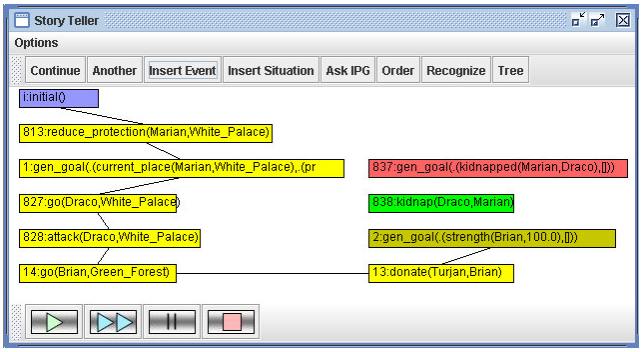

Figure 3: Plot Manager Interface

The user neither has direct control over the scene, nor over the characters themselves. Moreover, user intervention is always indirect, in the sense that any user intervention must be validated by IPG before being incorporated to the current plan.

Plot generation and dramatization are two separate processes, in contrast to pure character-based approaches, where user interaction affects plot structuring at real-time. This means that only during the simulation process the user has an opportunity to intervene in the creation of the plot.

As explained in the previous section, plots are created in an attempt to fulfill goals that the characters aim to achieve. At each simulation step, new goals may be inferred and automatically added to the plot, which causes the insertion of a new set of events. The events inserted in the plot so far are sent to the graphical interface for user intervention via the Plot Manager, which offers two commands for automatic plot generation: **another** and **continue**. The command **another**, requests from IPG an alternative solution to achieve the same goals of the step just finished. The command **continue** asks IPG to try to infer new goals and continue the simulation process.

These two commands provide a form of *weak user intervention*. The user merely selects partially-generated plots that seem interesting from his/her perspective to proceed with the simulation. This weak form of intervention usually leads the plot to situations that the author of the story has devised beforehand.

The Plot Manager offers, in addition, two complementary means for *strong user intervention* in the creation of more personalized stories. Firstly, the command **insert situation** allows users to specify situations that should occur at specific times along the plot by inserting some additional goal to be reached. The specific details of how the goal will be accomplished are left to IPG, which is charged to find a solution, if one exists, using the planning algorithm. It must be noted that, in view of performance considerations, a valid computable plan may fail to be obtained if the search limits currently configured in IPG are exceeded. As in the purely automatic generation, the user may confirm the solution (by indicating **continue**) or request an alternative (**another**), which (as said before), is a case of weak intervention. Secondly, at a lower interaction level, the user is allowed to explicitly insert events into the plot with the command **insert event**. To validate the insertions, the user must invoke IPG through the **continue** command. At this moment, all user defined operations are submitted to IPG, which runs the planning algorithm to check whether or not they are consistent with the ongoing plot. If not, IPG tries to fulfill possible unsatisfied constraints by inserting further new operations in a specific order. The user may also remove user defined operations that were not yet incorporated to (or were rejected by) the planner.

Besides these interaction modes, the user can also use two other commands, **tree** and **recognize**. The **tree** command displays the available hierarchy of typical plans and can be used, by itself, as a clue to be taken into consideration when inserting new events in the story. Figure 4 shows the hierarchy for our swords-and-dragons example; blue edges denote composition (part-of link) and red edges denote generalization (is-a link).

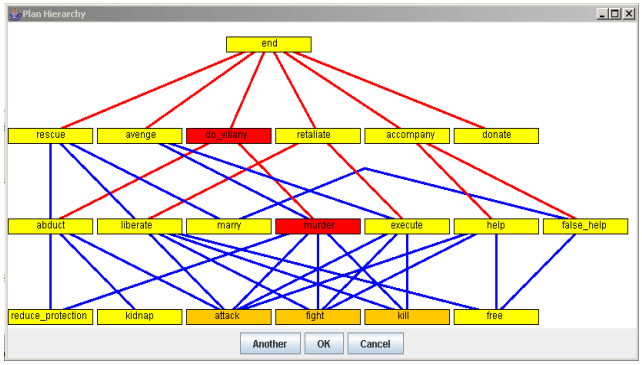

Figure 4: Plan Hierarchy Interface

When using the **recognize** command (which is supported by the plan-recognition feature of IPG) the user needs to mark one or more events already inserted and/or being considered for insertion in the Plot Manager interface and the system will try to match these events, as observations, against the library in an attempt to identify one or more typical plans subsuming them.

The system will then show the typical Plan Hierarchy representing the story genre in use with the complex operation found (if any) marked in red and its components marked in orange. The user can then choose if the complex operation found is an interesting one or try to change it into another one that fits the intended story. For example, the

list of observations [attack('Brian', 'Red_Castle'), kill('Brian', 'Draco')] fits in both rescue and avenge plans and thus suggests two alternative ways to structure the narrative from which the user may draw his preferences. Upon selecting the desired partial plan, its component events will be inserted in the Plot Manger interface.

The usage of plan hierarchies can be much enriched if literary indices are made available. For folktales, for example, there is the monumental index compiled by Aarne and Thompson (Aarne 1964). Their identified *themes* and *motifs* have always been an inexhaustible source of inspiration for novice and even experienced authors. Treated as fragments of typical plans, they could then be retrieved, to become part of user-composed plots.

Before dramatization, there must be – as said before – one additional user interaction that is actually mandatory, namely the conversion of the partially-ordered generated plan into a strict sequence, thereby completing the composition of a proper plot. Notice that, if the simulation is resumed afterwards, this addition of new temporal constraints is also an intervention, because it can affect the inference of new goals. To determine the sequence, the user connects the events in a sequential order of his/her choice, respecting the temporal constraints supplied by IPG. The plot's configuration emerges as the user moves the cursor to draw edges linking the operation boxes, starting from the root. To help the user in this process, we utilize colors to distinguish operations that are already connected (yellow), operations that – in view of the temporal constraints – can be immediately connected (green), or cannot yet be connected (red). The starting root is blue and the current operation being rendered is cyan. To connect two operation boxes, the user must click with the mouse over the source box and drag over the destination box (the same process is used to remove a link between two operations). Once the current plot (or part of it) is thus connected into a linear sequence, it can be dramatized by invoking the Drama Manager with the **render** command.

The tool also offers a facility for querying the IPG module about the state of any element of the narrative at a specific time Ti, using our temporal modal logic. This feature allows advanced users to find out, for instance, why an operation or goal is not being allowed, and helps authors to revise and tune the story requirements.

Dramatization

We have developed our own engine to support the graphical representation of the plots. It is implemented in C++ and uses the OpenGL graphical API to support real-time rendering of the 3D elements. Characters in a generated plot are regarded as actors for the dramatization.

The graphical engine does not have to perform any intelligent processing. It is merely responsible for rendering, at each frame, the scene and the current actors' aspect and movements, resulting from real-time interactions with the scene and, occasionally, with other actors. In doing that, it follows the ordered sequence of events generated at the previous stages of simulation. The Drama Manager is the module that synchronizes characters' actions and the overall graphical representation.

The Drama Manager's job is not limited to assigning the actions that specific characters must perform. It translates symbolic operations into fully realized 3D visual graphical animations. And it must guarantee the synchronism and logical coherence between the intended world and its graphical representation. Figures 5 and 6 show some snapshots of the dramatization of the generated plots.

As received from IPG, the plot is organized as a sequence of events, each one associated with a discrete time instant. The simulation occurs in continuous real-time and the duration of an operation rendering is not previously known. Variable attributes change as the event is dramatized. In order to make logical and graphical representations compatible, the values of the variables before the dramatization of each event must agree with the pre-conditions of the event and the values at the end with its post-conditions.

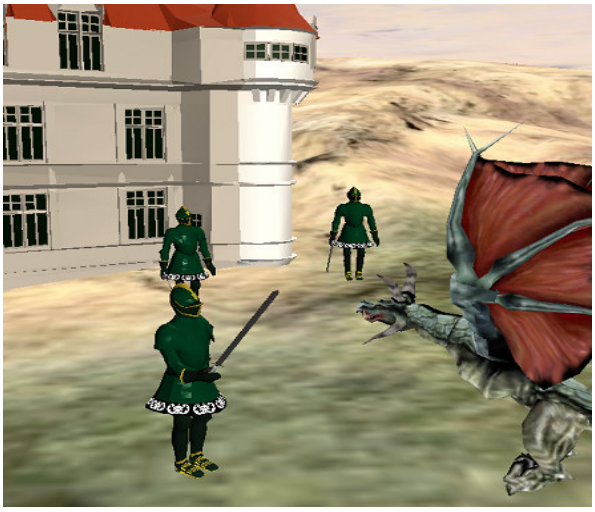

Figure 5: Draco attacking Marian's castle.

Figure 6: Hoel meeting Marian before getting married.

The dramatization starts by the selection of a specific event and the execution of the command **render** in the Plot Manager. All subsequent chained events from this point to the end are visualized, unless the user interrupts the process. When an event is activated for rendering, the engine uses the current values of the pertinent attributes as a starting point for the representation.

The user can alternate between plot generation and dramatization. In this case, after a dramatization, new events and time constraints can be added either by the user or by IPG. If dramatization is activated again, it can start only at events that occur before the modifications.

The Drama Manager converts all events into actions, which are delegated to specific actors, at specific times, according to the plot order of events. Whenever an event finishes, the Drama Manager asks the Plot Manager to give it the next event. If none exists, the dramatization stops.

The dramatization of an event ends when the involved actors(s) finish enacting the associated graphical representation. In our experiments, this may take from a few seconds to about one minute, depending on the kind of operation and on the scenario features.

Scene and Actors

For the graphical representation of the plots, according to the genre of the story being represented, the engine loads a specific scenario. The scenario is represented by a 3D environment that is suitable for the events and characters that the story is supposed to contain, taking into consideration the conventions of the genre (e.g. the presence of castles).

Because most events have an association with the place where they are performed, actors should be constrained, while moving through the scene, to maintain graphical coherence with respect to how they follow the plot directions. Buildings, such as castles and other genre-related objects, serve, more than as an ornament, as a conditioning factor to orient the displacements of the characters, the absolute and relative position where an action is to be executed, and the form to treat collisions. We make use of terrain reasoning and path-planning based on waypoints (Pozzer et al. 2004).

Actors have a geometric structure amenable to graphical representation, and are provided with a minimum of planning capabilities, at a low level of detail. Since actors are expected to play the assigned roles achieving an adequate performance, some rudimentary planning resources are indispensable, so that, in real-time, an actor be able to make decisions and to schedule the necessary micro-actions. In general, simple path-finding algorithms and direct inter-agent communication schemes are sufficient. Each actor must also incorporate behaviours for interacting with the physical environment and with the other actors. Contrary to the generality of the IPG planner, the local planning of each actor must be simplified to ensure short response times.

During graphical representation of the plot, all control of the actions each actor is supposed to perform is made by the Drama Manager. It acts as a director that coordinates sequences of actions performed by the whole cast. It continuously monitors the representation process, activating new tasks whenever the previous ones have been finished. As a director, it also controls the positioning of the (virtual) camera, which an option of LOGTELL permits to be transferred to the user. The manual option provides zooming, rotation, and vertical and horizontal shifting; some users have found particularly entertaining to look at the scene from a bird's eye perspective, watching the plot unfold with all locations in view.

For IPG, as the number of characters increase, the computational effort required to control such characters and their interactions may become prohibitive. However, the use of fewer characters – a small number of actors, consequently – may lead to poor graphical representations. The test scenario used as an example in this paper, based on swords-and-dragons tales, features two heroes, one villain, one victim and a magician. To enhance the diversity and liveliness of plots, but also to turn the representation more realistic, we introduced a supporting cast, consisting of groups of soldiers (guardians) in charge of the protection of locations where the leading actors live, and where events take place. As opposed to the leading actors, whose actions are predetermined by the plot, these extras are endowed with a higher although still limited level of behavioural autonomy.

For the purposes of our example IPG totally ignores and not even distinguishes individual extras, since only as groups they have some influence over the plot conduction. For instance, when the plot is being represented, the graphical engine queries IPG about the current protection level of each location. At this moment, a proportional number of guardians is inserted into the scenario, together with the leading characters. We feel that, either as partially or fully autonomous graphical entities, supporting actors positively contribute plot visualization.

The degree of autonomy conceded to the extras leaves them free to perform certain actions randomly, such as walking in different directions; this feature is being improved with the integration of an AI middleware (Karlsson and Feijó 2005) into the Drama Manager.

When the actors are required to participate in some plot event, which has always a higher priority, the Drama Manager makes them interrupt momentarily whatever they were doing. So, the autonomous actions are not allowed to interfere with the execution of the plot; for instance, the guardians cannot inadvertently kill a leading actor.

Test Scenario

The test scenario currently in use for LOGTELL corresponds to a small sub-class of the popular swords-and-dragons genre. The possible events were modeled by just a few parameterized operations, which can nevertheless generate a considerable variety of different plots. The specified operations were the following:

- go(CH,PL): character CH goes to place PL;

- reduce_protection(VIC,PL): the protection of place PL (represented by the number of guardians) is spontaneously reduced by the prospective victim VIC;
- kidnap(VIL,VIC): the villainous character VIL kidnaps VIC;
- attack(CH,PL): character CH attacks place PL (fighting the guardians);
- fight(CH1,CH2): character CH1 fights character CH2;
- kill(CH1,CH2): character CH1 kills character CH2;
- free(HERO,VIC): character HERO frees character VIC, raising the degree of affection of VIC for HERO;
- marry(CH1,CH2): the two characters get married;
- donate(CH1, CH2): strength level of character CH2 is raised by the magical powers of CH1; and
- bewitch(CH1,CH2): the double effect of this operation is to instill an evil nature into CH2 and, at the same time, make him or her much stronger.

Besides these basic operations, a hierarchy of complex operations (structured by is-a or part-of links) was added:

- rescue, avenge - these are the two species of adventure. The rescue variety has components: abduct, liberate, marry, accompany, donate. The other variety, avenge, has components: murder, execute, accompany, donate.
- do villainy, retaliate, accompany - do villainy specializes into: abduct or murder; retaliate specializes into: liberate or execute; accompany specializes into: help or false help.
- abduct, murder, execute, liberate, help, false help. Abduct has components: reduce protection, attack, kidnap; murder has components: reduce protection, attack, fight, kill; liberate has components: attack, fight, kill, free; execute has components: attack, fight, kill; help has components: attack, fight, free; false help has components: free, marry.

We left out two basic operations from this hierarchy. As operation go is in fact a component of practically all others, it is therefore assumed to be always present. And bewitch was deliberately excluded, since any plot including it should not be considered typical in the context of our genre (a sort of tolerated *transgression* of the conventions).

The model of the genre was completed by the following goal-inference rules, presented here in English for simplicity:

- If a character plays the role of a victim, this character will spontaneously do something that puts her/him in a less protected situation.
- If the strongest character playing an heroic role is still weaker than the villain, this character will want to get stronger.
- If the protection level of a victim is reduced, the villain will want to kidnap the victim.
- If a victim is kidnapped, a hero will want to free her.
- If the affection levels of two characters vis-à-vis each other exceeds a threshold, they will want to marry.
- If a victim is killed, a hero will want to avenge her

As one of the possible starting configurations, we defined an initial state including the following information:

- Marian is a princess, living in a palace (the victim).
- Brian and Hoel are knights (the heroes).
- Turjan is a forest-dwelling magician (a donor, in Propp's sense).
- Draco is a dragon whose lair is in a red castle (the villain).
- The princess, the dragon, and the magician have protecting guardians around their homes.
- Each character is endowed with a certain strength level for fighting.
- The two heroes have a high affection for the princess, which is not reciprocated by her.
- Turjan is neutral with respect to all the others.

Examples of interactive step-wise plot composition

Using the tool, it is possible to generate many different plots. An example plot tells the classical happy-ending story: "The protection of Marian's castle is reduced. Draco regards that as an opportunity to kidnap her. Draco then goes to Marian's Castle, attacks the castle and kidnaps Marian. As a noble knight, Brian feels compelled to save her. But, before that, he needs to ask for Turjan's magic to raise his strength. He then goes to Draco's Castle, attacks the castle and fights Draco. He kills Draco and frees Marian, who starts loving her saviour. Motivated by their mutual affection, Brian and Marian go to the church and marry each other."

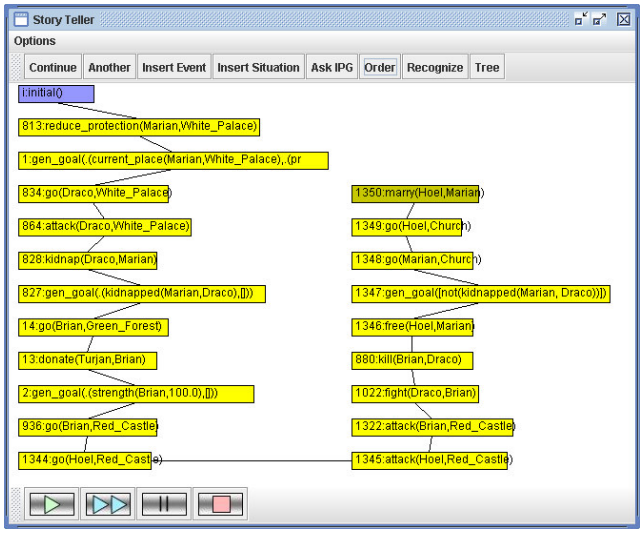

Figure 7: An example of a generated plot.

The plot in Figure 7 follows the same course until the point where Marian is kidnapped, but, after that, it can be summarized as follows: "The two knights, Brian and Hoel, propose to save the princess. They both go to Draco's castle and attack the guardians. But Brian alone fights Draco, and finally defeats and kills it. Hoel then is seen to free Marian, causing her to fall in love with him and become his wife. In spite of doing most of the effort to save Marian, Brian is not able to marry the princess."

Concluding Remarks

Having implemented and extended an initial version of LOGTELL, we have been running a number of experiments, which seem to demonstrate that combining goal inference, plan generation/recognition and user participation constitutes a promising strategy towards the production of plots which are both entertaining and coherent. Moreover, our modeling method, based on temporal logic, has proved adequate to capture the conventions of genres encompassing stories with a high degree of regularity, such as fairy tales (as one could foresee, on the basis of Propp's pioneering work) and, consequently, simple swords-and-dragons narratives.

On the negative side, we must admit that modern and post-modern genres, with their emphasis on a more radical transgression of any conventions should not be so easy to formalize in a systematic way.

Also, plan generation is unfortunately limited by computational complexity considerations. There is however a continuing research effort to improve its efficiency, and we intend to look into that, to try to upgrade the performance of the IPG planning algorithms. What we have already verified is that an interactive regime, with the intervention of the user at various stages and at different levels, as our methods and implemented tools favour, does much to expand such bounds. A particularly effective help to this interaction is provided by using plan-recognition over libraries of typical plans, which offer expert advice to all kinds of users.

A specific topic for our future research is how to alter the LOGTELL approach in order to offer more advanced dramatization resources, such as investing more on affective computing (Izard 1991, Velázquez 1997) and improving automatic camera control.

To explore the range of applications of LOGTELL is yet another objective of our project. The system could be used, for example, to generate side quests in MMORPGs. Our efforts are now mainly concentrated on the continuing development of our tool, so as to cope with genres involving more sophisticated forms of communication among the characters and a deeper treatment of drives and emotions (Gratch and Marsella 2004).

Acknowledgments

This work has been partly sponsored by CNPq, CAPES and FINEP.

References

Aarne, A. 1964. *The Types of the Folktale: A Classification and Bibliography*. Translated and enlarged by Thompson, S., FF Communications.

Bal, M. 2002. *Narratology - Introduction to the Theory of Narrative*. University of Toronto Press.

Cavazza, M., Charles, F., and Mead, S. 2002. Character-based interactive storytelling. *IEEE Intelligent Systems*, sp. issue on AI in Interactive Entertainment, 17(4):17-24.

Ciarlini, A. 1999. Geração interativa de enredos. PhD thesis, Depto. de Informática, PUC-Rio, Brazil.

Ciarlini, A., Veloso, P., and Furtado, A. 2000. A Formal Framework for Modelling at the Behavioural Level. In *Proc. of the Tenth European-Japanese Conference on Information Modelling and Knowledge Bases*, Saariselkä, Finland.

Crawford, C. 1999. Assumptions underlying the Erasmatron storytelling system. In *Working Notes of the 1999 AAAI Spring Symposium on Narrative Intelligence*.

Glassner, A. 2004. *Interactive Storytelling.* A K Peters.

Gratch, J. and Marsella, S. 2004. A domain independent framework for modelling emotion. In *Journal of Cognitive Systems Research*, 5(4):269-306.

Karlsson, B. and Feijó, B. 2005. AI Middleware as Means for Improving Gameplay. In *Proceedings of the ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, Valencia, Spain.

Kautz, H. 1991. A Formal Theory of Plan Recognition and its Implementation. In *Reasoning about Plans*. J. F. Allen et al. (eds.). Morgan Kaufmann, San Mateo, EUA.

Izard, C. E. 1991. *The psychology of emotions*. New York: Plenum Press, New York.

Mateas, M., and Stern, A. 2000. Towards integrating plot and character for interactive drama. In *Socially Intelligent Agents: the Human in the Loop*, AAAI Fall Symposium, technical report, p. 113-118, Menlo Park, USA.

Paiva, A., Machado, I., and Prada, R. 2001. Heroes, villians, magicians, ...: Dramatis personae in a virtual story creation environment. In *Proc. Intelligent User Interfaces 2001*: 129-136, Santa Fe, USA.

Pozzer, C. T., Feijo, B., Ciarlini, A. et al. 2004. Managing Actions and Movements of Non-Player Characters in Computer Games. In *Proc. of the Brazilian Symposium on Computer Games and Digital Entertainment*.

Propp, V. 1968. *Morphology of the Folktale*, Laurence Scott (trans.), Austin: University of Texas Press.

Riedl, M.; Young, M. 2004. An intent-driven planner for multi-agent story generation. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multi Agent Systems*, New York, USA.

Spierling, U., Braun, N., Iurgel, I., and Grasbon, D. 2002. Setting the scene: playing digital director in interactive storytelling and creation. *Computers&Graphics*, 26:31-44.

Sgouros, N. M. 1999. Dynamic generation, management and resolution of interactive plots. *Artificial Intelligence*, 107(1):29-62.

Velázquez, J. D. 1997. Modeling emotions and other motivations in synthetic agents. In *AAAI-97: Proceedings of The Fourteenth National Conference on Artificial Intelligence*, p. 10-15, Menlo Park, USA. AAAI Press.

Yang, Q., Tenenberg, J. and Woods, S. 1996. On the Implementation and Evaluation of Abtweak. In *Computational Intelligence Journal*, 12(2):295-318.