AISB 2011

AI & Games

Editors:
Dimitar Kazakov &
George Tsoulas

THE UNIVERSITY of York

Foreword from the Convention Chairs

The AISB'11 call for symposium proposals particularly encouraged events drawing more strongly on the cognitive science aspect of the AISB remit. The result is a coherent programme with a very strong interdisciplinary character, which is also matched in the choice of plenary speakers. The three symposia looking at the interaction between Computing and Philosophy, the prospect of machine consciousness and the quest for a new, comprehensive intelligence test, form a coherent unit where the eternal questions of who we are and what makes us so are asked from a dual Human-Machine perspective. The Symposia on Active Vision, Computational Models of Cognitive Development and Human Memory for Artificial Agents demonstrate how better understanding of the nature and basis of cognitive processes can advance work on Artificial Intelligence and, inversely, how computational models of these processes can help better to understand them. The prominent multi-agent design and modelling paradigm links the Symposium on Social Networks and Multi-agent Systems with the one on AI and Games. Finally, the Symposium on Learning Language Models from Multilingual Corpora, which brings together some of the first attempts in this area, can also be seen through the prism of such a general notion in Philosophy and Linguistics as semiosis, and the dual role of sign and interpretant that text plays in translations.

We are delighted that after another ten successful years in its long history, the AISB convention is returning to the University of York. The 2011 convention takes place on the brand-new Heslington East campus, the result of a multi-million pound expansion that is now the new home of the Department of Computer Science, and hosts the Excellence Hub for Yorkshire and Humber, a new incubator for interdisciplinary research and interaction between academia and industry. The last few years have seen a strong involvement of the Computer Science Department in such interdisciplinary collaboration through the York Centre for Complex Systems Analysis (YCCSA), and we hope that this convention will provide a boost for more synergy between York departments, with other institutions conducting AI-related research in the region, and beyond. As the programme shows, we have also made an effort to promote cooperation with industry and use the convention to support school outreach. The convention format makes it perfect for establishing dialogue and collaboration in new areas of research, as well as across disciplines, and we hope that this year, it will play again this role to the full. We want to thank everyone who has contributed to it or otherwise made this event possible and wish all participants a fruitful and enjoyable time in York.

Dimitar Kazakov and George Tsoulas

4th Symposium on AI & Games

A symposium at AISB 2011.
http://www.dcs.shef.ac.uk/~daniela/AISB11/

Programme Chairs

David C. Moffat, Glasgow Caledonian University

Daniela M. Romano, Sheffield University

Introduction

Computer games form an important sector of the digital economy, and they are sophisticated in many ways. The need for better AI in games is deeply felt, however, and recognised by the industry. Conversely, games offer new challenges and excellent application domains for AI technology and research. They are increasingly used for education, serious games or game-based learning, where story and AI techniques create a believable, engaging experience for learners.

This symposium focuses on the application of artificial intelligence, or intelligent techniques, frameworks and theories, to create interactive, engaging, intelligent games.

There are academic papers on AI algorithms, including Monte-Carlo search techniques, and the application of Natural Language Processing to giving better tutorial advice situated in a game context. Other papers, from academics as well as from leading AI developers working in the industry, are about techniques to make more intelligent game characters and environments.

Paolo Busetta (who is Principal Architect of the *AOS Group*, Cambridge) gives the keynote speech, about the CoJACK system, which adds to beliefs, desires and intentions, more psychological primitives to simulate agents with greater believability.

Finally there are some more speculative papers that propose ways to apply emotion brain-computer interfacing to virtual reality environments, and an application of established AI technologies to serious games for cultural education.

We would like to thank the Programme Committee members for their interest, their reviews of the submissions, and their helpful suggestions to the authors. Thanks also to the AISB convention organising team, at the University of York.

Programme Committee:

Industry day

The AI & Games Symposium this year is happy to include an AI & Games Industry session, at which companies from industry are invited to showcase their games and which also includes a presentation from a venture capital fund. The purpose of the session is to help academia and industry to network together and commercialise new ideas. The Industry session has been organised with the assistance of Dimitar Kazakov (University of York).

Sponsorship by Namaste

We are grateful to Namaste Entertainment Pvt. Ltd., of London and Kathmandu, for supporting two students to attend the Symposium. Thanks to Rodolfo Rosini in particular for his efforts and interest in AI & Games.

http://namasteevents.com/

Contents

Keynote speech by: ***Paolo Busetta***

An introduction to CoJACK

CoJACK is an extension of JACK, a computer language for the development of multi-agent systems, used in simulation and training especially to model sophisticated behaviors and tactics. CoJACK extends the BDI paradigm implemented by JACK with a set of primitives (interpreted by or embedded into its execution engine and controlled via configurable mathematical formulae and parameters) to represent "sub-rational" aspects of human performance - e.g. speed and order of memory recall, errors and confusion during recall, different utility of alternative courses of action, neglecting of commitments, and so on. These primitives can be used to model certain non-rational aspects - fatigue, emotions, drug intake, personality traits and level of experience - without directly affecting the representation of "ideal" behavior, e.g. tactics written according to doctrine. CoJACK has been recently inserted in a list of reference cognitive architectures published in 2010 (see http://bicasociety.org/cogarch/) by the Biologically Inspired Cognitive Architectures Society.

In this seminar, we will provide a quick introduction to CoJACK and show some examples of VR-based training systems for a specific domain (anti-terrorism training).

The first part of the talk will provide background information on the BDI (Belief – Desire – Intention) paradigm, will motivate the need for cognitive extension, and will introduce the architecture of CoJACK.

The second part will show examples built using VBS2 by Bohemia Interactive. VBS2 is a virtual reality / serious games platform, popular for military training; we used CoJACK for animating the avatars involved in the game.

Paolo Busetta
Principal Architect
AOS Group
Wellington House, East Road Cambridge CB1 1BH
T: +44 1223 308 000
www.aosgrp.co.uk

Procedural Quests: A Focus for Agent Interaction in Role-Playing-Games

John Grey and **Joanna Bryson** [1]

Abstract.

In current videogames non-player characters' (NPCs') abilities to be active, dynamic agents are typically constrained to a bare minimum. Agents have very local behaviours to deal with actions, which can combine in limited ways with global game mechanics to deal with repeated behaviours. Here we present a systems-AI approach to designing NPCs. The proposed NPC design is capable of dynamic dialog, with context generated from both episodic memory and emotional valence towards previous social interactions. The NPCs can be allowed to run independently of users to develop a believable social network of friendships and grudges, with memories supporting such opinions. Additionally, NPCs can spread information in a more realistic manner than the current standard, global mechanisms. This information fomrs a culture, which then serves as the motivation for quests offered to other characters and the user that encounters these societies.

1 Introduction

In Role-Playing-Games (RPGs) gameplay is typically structured through quests. Moira Brown's task to kill some raiders and loot a local mall in *Fallout 3* is one example of a quest. Quests serve the purpose of directing player action, while also advancing a narrative. Side quests are types of quests with reduced narrative complexity, which serve to advance the game in relatively basic ways, for example by giving a character more experience or weapons. However, in many games the narrative component of side quests is not just reduced, but missing entirely. Without a narrative context, such sidequests can be unmotivated and tedious for players. Additionally quests often limit the methods of interaction between the player and the world, and between the player and the non-player characters (NPC) they encounter. Players' choices are constrained, and Quest Arcs (collections of quests that create a larger narrative) offer choice only at branch points in the narrative. Such choices do not affect the general world that surrounds the player, or, when they do, this generally takes the form of global morality meters — single variables that indicate a value known to every character in the game. As such, in a large number of modern RPGs, killing an NPC will instantly reduce the player's global moral standing in the world, even with NPCs who could not possibly know of the relevant action. All police (or whatever the game equivalent may be) will converge to arrest or fight the player. This is so prevalent a form of interaction in games that it has entered into videogame humour as:

The Guy in the Street Rule:. *No matter how fast you travel, rumours of world events always travel faster. When you get to anywhere, the people on the street are already talking about where you've been. The stories of your past experiences will spread even if no witnesses were around to see them. [20]*

Believable Social Agents (BSAs) offer a mechanism for reducing the constraints and rigidity of side quests and increasing the level of player engagement in computer games without substantially increasing the cost of their production. Such agents autonomously generate complex social networks of friendships and grudges leading to a more interesting social landscape when approached by the player. This gives both the player and NPC more to talk about, automatically generates variation in game play, and provides more narrative motivation to the quests needed by the player to increase their character's status.

Here we present a methodology for developing Believable Social Agents, as an option to add context to games without the author intensive requirements. We begin with a brief review of quests and believability. Then we describe our approach and a quest generator completed under that approach. Finally we describe the developer's experience of building in our system, with mention to how we have extended on the current state of the art for AI representations underlying quest generation.

2 Believability and Narrative Quests

We have developed the idea of generative procedural quests from a variety of literatures relating to games and AI, including story generation and computational creativity. Of particular importance are the areas of Quests, both 'traditional' and procedural, and AI, specifically the typical characteristics of videogame AI and suitable methods for videogames. Quests are most usefully described by Jeff Howard as the intersection between gameplay and narrative [24]. In this way, particular patterns of action are combined with a story that gives context to that action, typically killing someone or something (a *kill quest*), or getting an item (a *fetch quest*).

2.1 Agents

Artificial intelligence has been used in videogames in a wide variety of forms for many years. These uses of artificial intelligence are generally concerned with the control of enemies, in the form of individual enemies in First-Person Shooter games, or overall teams in strategy games. In academic research meanwhile, artificial intelligence has dealt with a wide range of issues, from learning to natural language processing to robotics planning. However, NPC artificial intelligence has generally been a lot simpler, with World of Warcraft's [7] NPCs being prime examples. In World of Warcraft, NPCs stand in one spot, an exclamation mark above their head indicating they have

[1] University of Bath, UK, email: johngrey4296@gmail.com, jjb@cs.bath.ac.uk

a quest available, and are otherwise non-reactive. NPCs of this form are one form of 'obviously stupid' agents that are identified by Bates as 'perhaps the primary impediment to fully suspending disbelief' for a virtual world [5, p2].

Bates [6] goes on to note that there is a difference between the requirements of traditional artificial intelligence and BSAs, in that one focuses on high competence and realistic behaviour, while the other merely requires that an agent 'not be clearly stupid or unreal' [6]. This shift of perspective is a very important difference between interactive fiction perspectives and more traditional AI perspectives. Blumberg [8] points out that the classic era of animation contained many characters that are *believable* — that is, immersive and emotionally engaging — without being even slightly realistic. They communicate emotional state with grossly exagerated gestures and actions, yet maintain our identification and sympathy. This description is congruent with Mateas' [25, p8], and also Barros arguments [4] which highlight a very specific difference between the two styles. In traditional AI planning of behaviours, the emphasis is generally on the optimality of the solution, as discussed in [9]. In 'expressive AI' [25, p5] there are 'softer' dramatic constraints [4, p35]. Current agents in games that will stand in the same place for the entire game, or run screaming for 2 metres when the player kills someone, but then turn around and continue calmly on their way, is very clearly stupid and unreal, thus supporting Bates. In the present work, we generate side quests procedurally, based on both local events and local knowledge of historic events. In so doing, we utilize research into agents and AI in a constrained context, making the actions and behaviours of NPCs emerge naturally from the dynamics of the game, in contrast to either isolated or overly global supplements. This increase believability.

The approaches available for implementing the sorts of agents needed for quests are divided into two main categories by Mateas [25], who makes a distinction between 'Classical' agent approaches, and 'Interactionist' approaches. Classical approaches attempt to model mental processes, while Interactionist, or Behaviour Based approaches focus on the results of intelligence. Mateas and others have concluded that interactionist approaches are more useful in application to believable game and story agents, both because of their dynamic nature — responding in a natural way to environmental contingencies, and because they are easier to develop in [28, 8, 33, 13].

The interactionist approach — dynamic, behaviour based or sometimes 'reactive' AI — is sometimes denigrated as being too simple to carry a narrative plot, because the individual agents 'only' react to their environment and the opportunities is presents. In both individual agents and game AI more generally, many have felt a need to reintroduce the structure of a more formal planning system "on top" of the behaviour based system, in order to guarantee structured, coherant plans [14]. This approach allows the combination of low-level behaviours and dynamic / reactive plans that do not require computationally expensive searching, and computationally expensive constructive planners which run less frequently but perform the global reasoning for an overall plan. Examples of this approach include interactive story generators such as Mateas' Facade [25] or Hayes-Roth's woggles [22]. These include a drama manager that closely resembles a typical constructive planner, but also include individual character agents. Drama managers are also simiarl to Game Masters in traditional Pen and Paper RPGs [3]. These views, combined with Brom's work in Emohawk [10] suggest that autonomous characters require more than just basic behaviours and reactive plans at the individual agent level to create sufficient dynamic worlds, despite the variety of character-focused methods of story generation. On a related note, Bryson [13] has previously argued that the best approach to game design is to in fact facilitate authors in creating characters that will drive the narrative for themselves.

Despite their emphasis on narrative, Mateas and Aylett both focus on engaging interactive stories, and thus miss the side-quest as an avenue for more structured game narratives. We believe that in fact the narrative needs of a side quest provide enough context for a meaningful dynamic world *without* requiring the encumberment of an overarching managers that resemble the story generation programs mentioned earlier. This is the approach taken in the present work.

2.2 Procedural Quests and Social Agents

Actual attempts at, and considerations of, procedural quests are few and far between. Calvin Ashmore's work on *Charbitat* is an important exception, generating procedural worlds with spatial quests of a key-lock form [2]. However, that work does not deal with creating meaning, either in Howard's or Salen's conception. Anne Sullivan's work on the GrailGM takes an alternative path from that of this work, and uses a Quest manager instead of agents [35]. Although this approach can work, it should be noted that procedural quests are similar to computer-based story generation, in that both may take either story or character-centric approaches [32]. Story-centric managers can provide greater narrative coherence, while character-centric approaches can enable more believable and understandable actions by characters.

Meanwhile, there have been experiments with procedural quests in videogames. Notably the game *Yoda Stories*, which generates both the worlds and the story to be followed in each play through of the game. Similarly *Din's Curse* enables enemies that live long enough to become unique, which then triggers quests to kill that particularly-important enemy. Additionally, *Din's Curse* operates through having archetypes of characters within a town, rather than named characters, allowing quests interrupted by a death of a character, to trigger a new quest of finding a new instance of that archetype.

Related work deals with Believable Social Agents in interactive contexts, but this is also relatively limited. Per Persson deals with interactivity, but the requirements of interactive narrative would appear to be greater than those needed for the creation of contexts for quests, and as such would unnecessarily complicate matters [30]. Michael Mateas' work, both on *Facade* [27], and *The Prom* [26] are the primary works concerning social behaviours as a part of game mechanics. However, these place relatively complex agents at the forefront of a game's dynamics, and are as such described as *Social Games*, in that the main game mechanics revolve around social interactions, which is quite different from the intended genres of this current work, which is RPGs.

3 NPC Design for Believable Social Agents

In this section we introduce our approach to creatin gthe social angents that will in time generate a beliveable society to engage the player. The principle design elements for Believable Social Agents are:

1. A set of general-purpose priorities for the agent.
2. Individual memory and perception.
3. Conversational ability.

An implemented Believable Social Agent can then instantiate quests as necessary, which requires particular considerations for the design of such quests.

Figure 1. A typical view in the prototype game

3.1 General Prioritised Action Selection

NPC and Agent design is a broad area that may utilise a number of approaches. Recently particular emphasis has been given to the use of Behaviour Trees and Hierarchical Plans for agents in games [23]. Although neural nets have been used in some games such as *Creatures* [18], Behaviour based approaches appear to be favoured in modern videogames due to their modularity, ease of creation, revision, and their relatively light processing requirements [19]. We have chosen to use the Behaviour Oriented Design (BOD) approach to develop our NPC [17]. BOD provides a set of heuristics and an iterative development strategy for creating both character behaviour and a hierarchical control the Prioritised, Ordered, Slip-stack Hierarchical (POSH) dynamic plans, which are similar to behaviour trees. BOD has previously been applied to video games and has been featured in some game AI design tools [12, 1].

BOD addresses the requirements of BSA in the following way:

1. *A set of general-purpose priorities for the agent.* We encode these in terms of BOD's POSH plans, although any hierarchical AI planning structure could be used.
2. *Individual memory and perception.* Under BOD (like OOD [21]) the capacity for memory is specified in behaviour modules which also provide methods for generating and accessing that memory, and further methods for acting on it.
3. *Conversational ability.* BOD does not provide these specifically, but rather we customised a set of general-purpose language abilities, utilising the memory capabilities of the agent extensively. These are described below.

$$\uparrow \left\langle \begin{array}{r} (\text{Strong / Scary Enemy Nearby}) \Rightarrow \textit{Flee} \\ (\text{Suitable Enemy Nearby}) \Rightarrow \textit{Designate to Attack} \\ (\text{Target Designated}) \Rightarrow \textit{Attack} \\ (\text{NPC Designated}) \Rightarrow \textit{Talk} \\ (\text{Item Designated}) \Rightarrow \textit{Pickup} \\ (\text{Suitable NPC Nearby}) \Rightarrow \textit{Designate to Talk} \\ (\text{Suitable Item Nearby}) \Rightarrow \textit{Designate to Pick up} \\ (\text{Quest in Memory}) \Rightarrow \textit{Quest Action Designation} \\ () \Rightarrow \textit{Process Memories} \end{array} \right\rangle \quad (1)$$

Figure 2. The BSA Drive Collection

$$\uparrow \left\langle \begin{array}{r} (\text{Strong / Scary Enemy Nearby}) \Rightarrow \textit{Flee} \\ (\text{Target Designated}) \Rightarrow \textit{Attack} \\ (\text{NPC Designated}) \Rightarrow \textit{Talk} \\ (\text{Item Designated}) \Rightarrow \textit{Pickup} \\ (\text{Suitable Enemy Nearby}) \Rightarrow \textit{Designate to Attack} \\ (\text{Suitable NPC Nearby}) \Rightarrow \textit{Designate to Talk} \\ (\text{Suitable Item Nearby}) \Rightarrow \textit{Designate to Pick up} \\ (\text{Quest in Memory}) \Rightarrow \textit{Quest Action Designation} \\ () \Rightarrow \textit{Process Memories} \end{array} \right\rangle \quad (2)$$

Figure 3. The original, easily interruptible, BSA Drive Collection

In BOD, the top of the plan hierarchy is the Drive Collection, which determines which of the behaviours available to an agent should be executed at any particular instant. In this article we illustrate single levels of a POSH hierarchy, which from this perspective are very similar to a STRIPS plan or Nilsson's Teleo-Reactive Plans [16, 29]. Within each sub-component of the hierarchy, each possible action is guarded both by a necessary precondition which the agent must sense, and by a priority. If more than one action can be executed, the highest-priority one of these is executed.

Figure 2 shows the drive collection for all the BSAs in our society. What differs between agents are not their abstract priorities, but rathere their experience, memory, and physical location. The actions derived from the drive collection form the basic abilities from which quests can be constructed. These actions are sequenced by the interaction of the POSH planner and its environment — the plan elements are arranged in an order such that they will generate action and change in both the character and the world. As the current focus for quests are killing and fetching, this short list of behaviours is sufficient. The general pattern for quest activities is to talk to an NPC, receive a quest, then move to a target location, either kill an NPC or pickup an item, and then return to the original NPC. Thus, the *Quest Action Designation* behaviour merely queries a quest and, according to the state of the quest, deals with the target, or returns to the original giving NPC.

The ordering of behaviours, placing the designation of talking and item targets lower in the priority of the plan as a group, while having attack designation higher than the attack action, is to ensure believable behaviour. If the drives were arranged as in Figure 3, agents could move across the world to attack a particular individual, all the while being under attack from others enemies. This is undesirable for combat, but slightly more desirable for communication and item behaviours.

3.2 Episodic Memory, Emotions and Perception

The memory and emotional capabilities of the NPCs are used to provide context for generated quests. The general concepts of the design are based on Brom *et al*'s episodic memory [11]. Memories are a particular data structure, based on a *Memory Primitive* (MP) that holds any information that the system may want to recall, such as the NPCs involved in the action, the type of action, and any items involved in the action. Specialising the memory in this way may seem restricted compared to general-purpose Cognitive Architectures such as SOAR [31], but such restriction helps keep the design clear and the game AI light-weight. The essential element of a MP is that it has an *ac-*

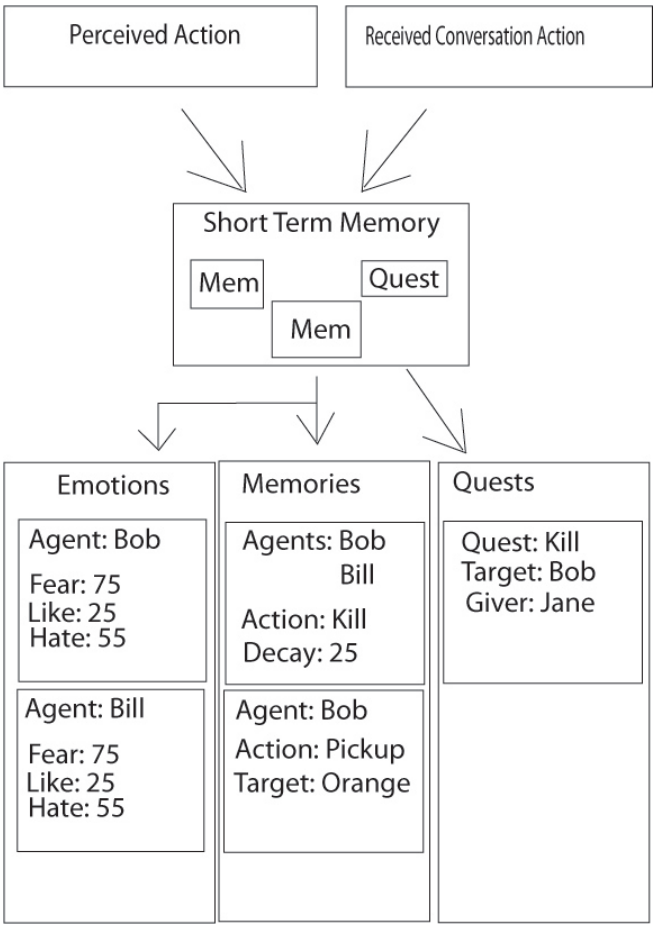


Figure 4. Structure of Agent Memory

tion variable which can describe whatever high level actions an agent can take. Thus, for all main drives in the drive collection (attacking, fleeing, talking, picking up items) there is an enumeration that can be inserted into the *action* variable. Coupled with variables to store the actors in that memory, there arises a simple data structure that can describe anything (Bob killed Alice, Bill picked up orange etc) an agent can do quite simply. As capabilities are added to the agents Drive Collection, the enumeration of actions grows as well. Add a drink ability, there is a need for a drink enumeration and so forth.

Emotions form another data structure in the overall memory design. Each NPC has one instantiation of the emotion data structure associated with every other individual character it knows. They hold basic integer values to record particular strengths of opinion, such as fear and hate. In this system, *emotions* are not a transient state as often understood in research [36]. Rather here, emotions are labels for long-term valence directed toward a particular individual, using such emotional terms as hate, friendliness, and fear. As events are perceived (described below), the emotional values for the perceived actors are adjusted as necessary.

The actual structure for the memory is relatively simplistic, with the entire memory shown in Figure 4. Memories are received through a regular perception of local events, or generated based on the agents' own behaviour, and added into a short term memory (STM) object.

$$\uparrow \left\langle \begin{array}{r} (\text{Turn to Speak}) \Rightarrow \textit{Speak} \\ (\text{Turn to Listen}) \Rightarrow \textit{Listen} \\ (\text{Turn to Wait}) \Rightarrow \textit{Wait} \\ () \Rightarrow \textit{Sort Conversation Turn} \end{array} \right\rangle \quad (3)$$

Figure 5. The *Conversation* Competence

$$\downarrow \left\langle \begin{array}{r} (\text{Moving and Close to NPC}) \Rightarrow \textit{Stop Moving} \\ (\text{Has said something previously}) \Rightarrow \textit{Clear Text} \\ (\text{Has nothing selected to say}) \Rightarrow \textit{Select Something to Say} \\ (\text{Has something selected to say}) \Rightarrow \textit{Set Speech Text} \\ () \Rightarrow \textit{Finish} \end{array} \right\rangle \quad (4)$$

Figure 6. The *Speak* Sequence

The low priority element of the Drive Collection *Process Memories*, seen in Figure 2, assesses any MPs in the stack. Memories that occurred recently are processed sooner, enabling potential functionality for if too many things happen, older memories that have not been processed are removed and forgotten without being processed. The assessment of items in the STM does three separate things:

- Changes the emotion values of all involved parties of the MP based on general rules of behaviour (eg: random attacks will decrease friendliness and increase fear, giving items will increase friendliness).
- Adds the MP to the long term memory if it is of a type important enough.
- Removes the assessed MP from the STM.

Additionally, when added to the long term memory, each MP initialises a *decay* value, which is regularly decremented if it is not accessed. If a memory is accessed and used, its decay value is reset. If an MP's decay value reaches zero, the memory is removed, and thus 'forgotten'. Memories provide, among other things, specific reasons for the creation of quests ("Bob killed Bill so I want Bob killed"), while emotions provide a more general context ("I really hate Bob so I want him killed"). Additionally the two may be combined ("Bob killed Bill, who I liked, so I want Bob dead").

The emotion system gives agents reasons for actions, without having specific memories to support such quests. This serves a twofold purpose. Firstly, it can provide initial conditions for the generation of quests prior to the generation of memories. Secondly, it allows memories to be forgotten, reducing actual memory requirements and latency of the program, while keeping the effects of the actions through the emotion variables.

3.3 Communication

Communication between agents, or between the player and an agent, is an interaction using the conversation competence, which is contained in the Drive Collection of the NPC. The BOD elements necessary for conversation are the competence seen in Figure 5, and the memory of the participants of the conversation (described above). The method of conversation is separated into the passing of underlying, *conversational primitives* (CP), and the conversion of those

primitives into human understandable text. CPs are passed between agents in a 'Pull' model, from the speaker to the listener, in the *Listen* action in Figure 5. Meanwhile, to make conversation understandable to humans, CPs are converted into text in the *Set Speech Text* action in Figure 6.

CPs are essentially a subclass of the primary *Memory Primitive* with additional variables to define the type of statement, and allow conversation to refer to particular memories, other conversational primitives, additional agents and items etc. The conversion process from Conversational Primitive to human understandable text is straightforward. Based on the type of conversational primitive (e.g. *Greeting*), a particular string is retrieved. Then, any variables in that string (such as *'NPC_I_AM_TALKING_TO'*) are replaced by the appropriate value (like the name *'Bob'*), eventually resulting in the human readable text (*'Good Morning Bob'*), which, in the prototype game, is displayed above the agent's sprite. This basic level of variability in statements can provide a surprising amount of freedom with a minimum amount of work.

Although BOD has previously been proposed as a mechanism for dialog planning [15], the present work represents the first application to our knowledge of POSH to conversational agents. Typically, conversation in computer games take the form of Conversation Trees, as can be seen in the *Fallout 3 GECK*. In the present work, conversation is ordered into two POSH competences: one for giving and one for receiving. Receiving is the simpler capability, in that it takes the last CP from the agent being talked to, and stores the information in the short term memory. The giving capability deals with selecting a new CP to offer up for the talk partner to receive, and add the relevant details to it. This decision process can be surprisingly simple through use of nested competences. The current implementation though has just a limited number of statement possibilities in a single layered plan, seen in Figure 7.

3.4 Quest Design

For the agents to be able to give and complete quests, the archetypes of quests from which instantiations are built need to be designed appropriately. In this work, there are just two quest types, but with additional game mechanics to utilize, additional quests archetypes can be designed. The essential parts of the quest design is to ensure:

- Quest Archetypes are understandably linked with the contexts that justify them (hating someone results in wishing them dead, not wanting them to have an apple pie).
- Quest Archetypes have defined structures (go there, kill/pickup that, return here) which can be understood and are utilised in the agents' quest fulfilment drive.
- Quest Archetypes deal with general events, with variables that can be filled as needed (so a KILL_QUEST has a variable of X that designates the target).

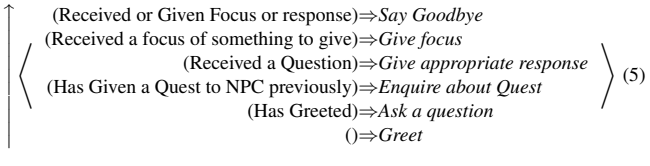


Figure 7. The *Speech Selection* competence

4 Example — Generating and Communicating Quests in a Virtual World

This work was done in a prototype game of a top-down, *Legend of Zelda* style, called *Shadow Quest* that was originally created by Prageeth Silva [34]. An example screenshot is shown in Figure 1.

We describe how such agents can be used to enable generative dynamic quest systems. Additionally we offer potential pathways to increase the flexibility of the current design very easily.

4.1 Designing Generative Quests: The Developer's Perspective

There are five main elements to the process by which a developer could add additional quests to the described quest generator. In the following sections these elements shall be described, and highlighted through the example of the addition of a 'Make Poison' quest type. The five stages of the implementation would be designing the base quest, implementing any supporting systems for the quest, implementing base level plans for the NPC agents, defining the related actions and quests in the memory, and finally adding the various conversation options into the conversation plan.

4.1.1 Choosing the Base Quest design

This starting stage essentially is the point at which the quest, from which all related plans, conversations, and memories etc, will be derived from. Thus, a quest could be designed for a 'Make Poison' quest, with a generic understanding of the quest consisting of:

- A quest of this type being given to an individual that is liked rather than hated or feared
- The quest receiver needing to get a number of items, go to a particular location, and combine them to create the poison
- The quest receiver returning to give the created poison to the quest giver

This quest type would be a 'subclass' as it were, of the typical 'fetch item' quest type. From this it is recognised that there would need to be some sort of game mechanic that would allow items to be combined to create other items. This leads to the second stage of the implementation process.

4.1.2 Creating the Base system to enable the quest

Following the generic understanding of the quest, any supporting game mechanics would need to be implemented. This is why the main design described above only considers 'kill' and 'fetch' quests, as others would require additional game mechanics. An alternative is to create the general game mechanics, or work with a system with a set of game mechanics already, and then create quests designed for those mechanics. This is merely a position determined by the stage of development, underlying technology of the development, and preference of the developer. As such, in this example, the imaginary game that the 'Make Poison' quest shall be added to will need an alchemy game mechanic, available to the player, and imitable by any NPCs. This would merely require that NPCs could perform any observable actions that the player can in the course of using alchemy: getting items, going to a workbench, creating a large puff of smoke, losing the correct ingredients, and receiving the right poison.

4.1.3 Creating Definitions in memory

Before an agent can *do* the elements of a quest, agents need to be able to understand the quest in terms of whether it is good or bad, whether it is something you would do to or for a friend or an enemy etc. Thus, memory structures would need to be created to describe the *alchemise* action(s). This would actually consist of adding to an enumerated list of actions available, and adjusting any perception routines as necessary. So instead of agents perceiving other agents of doing nothing while standing at a work bench, they can retrieve the 'alchemise' enumeration, and add that into an action memory along with the agent's name, and maybe the resulting poison that was created from doing the alchemise action.

4.1.4 Creating the Plan for the Agent

The next stage of development would require that there be the various plans created for the agents to enable use of alchemy. There would generally be at least two, possibly three, different plans at this stage to create:

- The Drive to *Alchemise* items to create the poison
- The Competence in the *Quest Action Designation* to describe the order of actions to complete the quest
- Any memory processing needs to place quests in memory and alter emotions.

Thus, the Drive collection (figure 2) would have an *alchemise* drive, that takes the agent to the workbench, and transfers the items to the workbench, calls the game mechanic to convert the items into the poison, and then picks up the poison. The competence for completing the entire quest would not only designate the target of an *Alchemise* action, and designate who to return to when finished, but also a repeatable section of the competence to retrieve as many items as were needed for the *alchemise* action to be successful, utilising a reuse of the already created 'fetch' quest type, or a 'subclass' of it, to ensure the agent went and found all the ingredients. Meanwhile changing memory processing rules etc merely adds an additional rule or set of rules to deal with the particular quest. So creating a poison would possibly be something to be scared of the agent it is made for, or something to like the agent for if the design of conversations later on tie the intention of making a poison to a potential target, who is disliked.

4.1.5 Creating Conversation Options

Once agents can perceive others as doing the 'alchemise' action, they have memory structures for the quest. Thus, they merely need to have any appropriate string(s) created to be inserted in the already existing conversational strings. There would be very little aspects of conversation that would be created entirely from scratch. Agents could, instead of asking 'Would you please *KILL BOB* for me', they can ask 'Would you please *MAKE A POISON* for me', or other such permutations of basic sentences. In a similar way, gossip and memory passing sentences such as 'Did you hear that *BOB KILLED BILL*' could become 'Did you hear that *BOB CREATED A POISON*' or even 'Did you hear that *BOB CREATED A POISON FOR JILL*'. There would be no need to create whole new structures for the conversations, as most elements would already exist and would not need specific context, such as acknowledgements ('yes, i did hear that'), and acceptance and refusal of the proposed quest.

4.2 Procedural Quests from the Player's perspective

From the player's perspective, the added quest can fit into gameplay and interaction with NPCs reasonably fluidly. Thus, in the imaginary new game, where alchemy and the alchemy quest has been implemented, and an expanded 'kill' quest has been created (of the form: 1) get give quest, 2) make, take, or ask for poison, 3) kill target with poison 4) return to original quest giver), the following sequence of events could occur:

- Bob dislikes Bill (due to random initialised emotion, or an actual memory)
- Bob asks Jill to kill Bill, because of the memory or emotion.
- Jill also dislikes Bill, and so accepts.
- Jill uses the expanded kill quest form, asking Jack to make her a poison.
- Jack, liking Jill, accepts, and performs the quest; getting ingredients, making the poison and then returning to Jill.
- Jill, now with the poison, goes and fulfils her quest by killing Bill with it. She returns to Bob, who is pleased with her.

The above, preliminary, sequence of events also has a number of secondary effects:

- To get the ingredients, Jack takes something from Bob. Bob then dislikes Jack slightly.
- Jack, who liked Bill, hears about, or witnesses, Jill killing Bill. He then hates Jill.
- Upon meeting the player, Jack gives the player a quest to kill Jill.
- The Player kills Jill, Bob hears and hates the player, asking Jack to kill the Player. Jack, liking the Player alot, refuses. Bob then attacks both the Player and Jack.

The secondary effects of the original kill quest produces a range of different actions. In this example quite extreme results, but further quest design and conversation options can allow for reproach, lies, and refusals to help in other situations, possibly even watching others getting attacked and killed without helping. However, what the effects, the quests given, and the conversations and gossip that occur in the game, all begin to tie each individual in the game together. Additionally, although there can be numerous effects of people liking some people, others hating others, and so on, it does not get confusing for the player, because in all conversations the NPCs can state their position. They will not just go 'kill Bob, because I say so', they will command 'kill Bob, *because* I hate him/ he killed my brother', or 'I don't like Bill, he stole my MacGuffin, go get it back for me'.

5 Discussion

We have proposed a suitable design for NPCs that can perceive actions around them, communicate those perceived actions between each other, and use such memories as the context for various side-equests in RPGs. Although currently the design is relatively basic, there are sufficient strengths to warrant further development. In particular, there are various aspects of the design that mean it can be easily used in videogames, and expanded upon.

5.1 Creating a continuum from Local to Global effects of actions

Due to the agent's capability to perceive actions nearby, remember them, and then transmit them in 'conversation' to other agents, there

can arise a fluid continuum of effects. Locals effects, as seen currently, of NPCs running screaming from a murder remains. However, now, instead of an automatic global morality value that is effected by the murder, the NPCs can pass information between them, which can be used to colour interactions. Eventually, it can reach a state where most NPCs know about the action, which can approximate the global value. Furthermore, as time goes by, the individual action is 'forgotten' in the NPC memory structures, leaving only the long term emotional valence. Whether this is preferable to the current situation would require a large amount of testing in a more aesthetically complete game, rather than a prototype.

5.2 Suitability for Use in Games

The proposed design is reasonably suitable for use in videogames in a number of ways. In terms of memory and processor use, the design is not currently optimised, but hints at opportunities in that area. POSH action selection is designed to reduce the combinatorial bottleneck of tree searches, and is unlikely to be more processor intensive than a similar Behaviour Tree implementation, which, as has been noted previously, is a method rapidly gaining popularity for videogame AI. Additionally, the memory design of the agents is suited to working with limited resource situations, as memories can decay, leaving only an emotional valence that can still be used to justify quests. Admittedly there could still be a certain amount of resources required for holding the emotion variables for each NPC, but again, these could be easily designed to decay if not used. Quests meanwhile need only be instantiated if an NPC accepts the quest, while could be automatically rejected if there is not enough available memory.

In terms of actual use in videogames, this proposed design only deals with the creation of quests, and the surrounding context for them. It does not however deal with the creation of *Quest Spaces* in which to perform quests. As such, this system would need to be able to be combined with some amount of procedural level generator [2], or an AI 'Director' to populate a static space with challenges in a similar way to Valve's *Left 4 Dead*.

Additionally, through use of this design, additional NPCs can be created with no increase of authorial effect. Once a single NPC is complete, the only areas that need to be added to are the list of basic strings which form sentences (there does not need to be a one to one relationship between Conversational Primitive and string. EG: There can be multiple ways to greet someone, not just 'Good morning X'), quest types, and names. To create differences, the NPCs just need to have randomised starting emotions to some NPCs, and be left to run for a while.

5.3 Freedom for expansion of the NPC design

The design, having been developed using BOD, errs on the side of simplicity and ease of redesign. As such, although the current design only deals with a limited number of behaviours, conversational options and quests, it is relatively trivial to implement additional capabilities. Of particular interest are the following:

5.3.1 Meta-Level Agents

Meta-level agents can, in this context, encompass a range of possibilities for more complex NPC interactions. With a slight addition, there can be non-physical 'Gods' that talk to only one individual, or put particular individuals on quests. Additionally, Meta-level agents can serve the purpose of creating factions, allowing groups of individuals to perform the same quests, or have telepaths, common goals, or even specialised behaviours (see below). By Meta-Level Agents, it is meant a non-physical agent that can interact with other agents, or an agent that serves as the aggregate of many agents memories or emotions.

5.3.2 Expanded Quest Design

Although at the moment there are just two types of quests accounted for in the design, kill quests and fetch quests, two points need to be considered. Firstly, Kill quests and Fetch quests are so prevalent in games due to the variety they can achieve. There can be assassinations (kill quests with stealth requirements), protection quests (kill everything *apart* from the target), stealing quests (Fetch quest with stealth requirements), information quests (Fetch quest but with the requirement being information instead of an item) and so on. There are a great variety of quests that are descended from Kill and Fetch quests, and that is without considering Quest arcs and the alternative conception of quests offered by Wibroe [37]. All of these can be easily implemented into the proposed design, merely adding the necessary behaviours to the agent, creating new quest archetypes, and memory or conversational primitives.

Additionally, large amounts of variation could be achieved through changing the focus of quests to become procedural behaviours. Such that instead of an NPC performing a quest once, he checks that he is constantly performing the quest. A trivial example of this would be creating a behaviour archetype that says the agent has to hop on one foot constantly. Combining this with the Meta-Level Agents described above, can easily create factions with particular idiosyncratic behaviours (eg: a warrior clan that hops on one foot all the time). It would then be a trivial matter to use perceptions of other agents *Not* performing that behaviour as an issue, which would result in a location based 'law' stating that an agent would need to hop on one foot when in the camp of those particular warriors. Again, although this example is humorous, the underlying possibilities are both easily adaptable and an expansion of the proposed NPC design.

5.3.3 Personality and Speech

Finally, there is no consideration of personality and other idiosyncrasies in the proposed design. These would also be very easy to implement into the design if necessary. Most RPGs have a large number of statistics to create individual differences, ranging from strength to charisma and intelligence. These are perfectly situated to enable individual differences in the proposed NPC design. Combining these statistics with basic signal processing concepts such as compression, expansion, and transfer functions would easily allow the opinions of other NPCs to be affected by base personality stats. The result would be that NPCs could have varying levels of emotional responses, so that a murder next to them does not effect them, or someone saying hello to them instantly makes them loath the NPC. Additionally, a simple addition of an emotional variable to the Conversational Primitive design would allow variation in the generated human readable text of NPCs, allowing greetings to easily range from 'Good morning X!', to 'oh, its you', with only the required change to the CP, the speech selection action, and the addition of meta data to the selectable strings available for speech.

6 Conclusion and Future Work

We have presented a design and basic methodology for easily implementable NPCs for use in RPGs. These NPCs have the capability to observe others, remember actions, and communicate remembered actions with other NPCs. They can use such memories to justify the request or offering of performing particular quests. Additionally, we have shown how the design is easily expandable to deal with other common elements of RPGs such as factions and personality.

Future work can take a number of distinct pathways. We plan to address the theoretical nature of this paper by implementing the AI design into a custom made, aesthetically complete game. We then intend to run a number of tests to investigate the effects of such 'capable' NPCs on player experience. It may be that only some proportion of a crowd should be 'interesting', not all of it. Regardless of this, we think Social Agents of this type could be a powerful creative use of AI in videogames. We also intend to investigate their application into the field of dynamic, computer generated music.

ACKNOWLEDGEMENTS

We would like to thank the referees for their comments and suggestions which helped improve this paper.

REFERENCES

[1] A. Armstrong. The Behaviour-Oriented Design of Modular Agent Intelligence. http://aigamedev.com/open/reviews/behavior-oriented-design-modular-agent/, 03 2008.

[2] C. Ashmore and M. Nitsche, 'The Quest in a Generated World', in *Proc. 2007 Digital Games Research Assoc.(DiGRA) Conference: Situated Play*, pp. 503–509. Citeseer, (2007).

[3] R Aylett, S Louchart, A Tychsen, M Hitchens, R Figueiredo, and C D Mata, 'Managing Emergent Character-Based Narrative', in *Proceedings of The Second International Conference on Intelligent Technologies for Interactive Entertainment*, Cancun, Mexico, (2008).

[4] L M Barros and S R Musse, 'Introducing Narrative Principles Into Planning-Based Interactive Storytelling', in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, pp. 35–42, Valencia, Spain, (2005).

[5] J Bates. The Nature of Character in Interative Worlds and The Oz Project, 1992.

[6] J Bates, B Loyall, and W S Reilly, 'Broad Agents', in *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, Boulder, Colorado, (1992).

[7] Blizzard Entertainment. World of Warcraft, 2005.

[8] Bruce Mitchell Blumberg, *Old Tricks, New Dogs: Ethology and Interactive Creatures*, Ph.D. dissertation, MIT, September 1996. Media Laboratory, Learning and Common Sense Section.

[9] C Brom. Personal Communication, 2010.

[10] C Brom, M Bida, J Gemrot, R Kadlec, and T Plch, 'Emohawk: Searching for a "Good" Emergent Narrative', in *Interactive Storytelling: Second Joint International Conference on Interactive Digital Storytelling*, eds., I A Iurgel, N Zagalo, and P Petta, pp. 86–91, Guimaraes, Portugal, (2009). ICIDS 2009.

[11] C. Brom, K. Pešková, and J. Lukavskỳ, 'What does your actor remember? towards characters with a full episodic memory', *Virtual Storytelling. Using Virtual Reality Technologies for Storytelling*, 89–101, (2007).

[12] Cyril Brom, Jakub Gemrot, Michal Bída, Ondrej Burkert, Sam J. Partington, and Joanna J. Bryson, 'POSH tools for game agent development by students and non-programmers', in *The Ninth International Computer Games Conference: AI, Mobile, Educational and Serious Games*, eds., Qasim Mehdi, Fred Mtenzi, Bryan Duggan, and Hugh McAtamney, pp. 126–133. University of Wolverhampton, (November 2006).

[13] Joanna J. Bryson, 'Creativity by design: A behaviour-based approach to creating creative play', in *AISB'99 Symposium on Creativity in Entertainment and Visual Art*, ed., Frank Nack, pp. 9–16, Sussex, (1999). The Society for the Study of Artificial Intelligence and the Simulation of Behaviour.

[14] Joanna J. Bryson, 'Cross-paradigm analysis of autonomous agent architecture', *Journal of Experimental and Theoretical Artificial Intelligence*, **12**(2), 165–190, (2000).

[15] Joanna J. Bryson, 'Making modularity work: Combining memory systems and intelligent processes in a dialog agent', in *AISB'00 Symposium on Designing a Functioning Mind*, ed., Aaron Sloman, pp. 21–30, (2000).

[16] Joanna J. Bryson and Lynn Andrea Stein, 'Architectures and idioms: Making progress in agent design', in *The Seventh International Workshop on Agent Theories, Architectures, and Languages (ATAL2000)*, eds., C. Castelfranchi and Y. Lespérance, pp. 73–88. Springer, (2001).

[17] Joanna J. Bryson and Lynn Andrea Stein, 'Modularity and design in reactive intelligence', in *Proceedings of the* 17^{th} *International Joint Conference on Artificial Intelligence*, pp. 1115–1120, Seattle, (August 2001). Morgan Kaufmann.

[18] A. J. Champandard. Evolving with creatures' ai: 15 tricks to mutate into your own game. http://aigamedev.com/open/highlights/creatures-ai/, October 2007.

[19] A. J. Champandard. This year in game ai: Analysis, trends from 2010 and predictions for 2011. http://aigamedev.com/open/editorial/2010-retrospective/, January 2011.

[20] Aaron et al. The Grand List of Console Role Playing Game Cliches, 2010.

[21] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley, Reading, MA, 1995.

[22] Barbara Hayes-Roth and Robert van Gent, 'Story-making with improvisational puppets', in *Proceedings of the First International Conference on Autonomous Agents*, ed., W. Lewis Johnson, pp. 1–7. ACM press, (February 1997).

[23] C. Hecker. My liner notes for spore/spore behavior tree docs. http://chrishecker.com/My_Liner_Notes_for_Spore/Spore-_Behavior_Tree_Docs, April 2009.

[24] Jeff Howard, *Quests: Design, Theory, and History in Games and Narratives*, A.K. Peters, 2008.

[25] M Mateas, *Interactive Drama, Art and Artificial Intelligence*, Ph.D. dissertation, Carnegie Mellon University, 2002.

[26] M. Mateas, 'The authoring challeng in interactive storytelling', in *Proceedings of Interactive Digital StoryTelling, ICIDS*, eds., R. Aylett, M. Y. Lim, S. Louchart, P. Petta, and M. Riedl, p. 1. Springer, (2010).

[27] M. Mateas and A. Stern, 'Façade: An experiment in building a fully-realized interactive drama', in *Game Developers Conference, Game Design track*. Citeseer, (2003).

[28] Michael Mateas, 'An oz-centric review of interactive drama and believable agents', Technical Report CMU-CS-97-156, School of Computer Science, Carnegie Mellon University, (June 1997).

[29] Nils J. Nilsson, 'Teleo-reactive programs for agent control', *Journal of Artificial Intelligence Research*, **1**, 139–158, (1994).

[30] Per Persson, Jarmo Laaksolahti, and Peter Lönnqvist, 'Understanding socially intelligent agents - a multilayered phenomenon', *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, **31**(5), 349–360, (2001).

[31] P.S. Rosenbloom, A. Newell, and J.E. Laird, *Soar Papers: Research on Integrated Intelligence*, MIT Press Cambridge, MA, USA, 1993.

[32] M.L. Ryan, *Possible worlds, artificial intelligence, and narrative theory*, Indiana Univ Pr, 1991.

[33] Phoebe Sengers, 'Do the thing right: An architecture for action expression', in *Proceedings of the Second International Conference on Autonomous Agents*, eds., Katia P Sycara and Michael Wooldridge, pp. 24–31. ACM Press, (1998).

[34] Prageeth Silva. Shadow Quest. http://shadowquest.thenewcoders.net/, 2010.

[35] Wardrip-Fruin N. Sullivan, A. and Mateas M., 'Rules of engagement: Moving beyond combat-based quests', in *Proceedings of Foundations of Digital Games, Intelligent Narrative Technologies Workshop*, (2010).

[36] Emmanuel Tanguy, Philip Willis, and Joanna J. Bryson, 'Emotions as durative dynamic state for action selection', in *Proceedings of the* 20^{th} *International Joint Conference on Artificial Intelligence*, pp. 1537–1542, Hyderabad, (January 2007). Morgan Kaufmann.

[37] M. Wibroe, KK Nygaard, and P.B. Andersen, 'Games and stories', *Virtual Interaction: Interaction in Virtual Inhabited 3D Worlds*, 166–181, (2001).

Playing Tetris Using Bandit-Based Monte-Carlo Planning

Zhongjie Cai and **Dapeng Zhang** and **Bernhard Nebel**[1]

Abstract.

Tetris is a stochastic, open-ended board game. Existing artificial Tetris players often use different evaluation functions and plan for only one or two pieces in advance. In this paper, we developed an artificial player for Tetris using the bandit-based Monte-Carlo planning method (UCT).

In Tetris, game states are often revisited. However, UCT does not keep the information of the game states explored in the previous planning episodes. We created a method to store such information for our player in a specially designed database to guide its future planning process. The planner for Tetris has a high branching factor. To improve game performance, we created a method to prune the planning tree and lower the branching factor.

The experiment results show that our player can successfully play Tetris, and the performance of our player is improved as the number of the played games increases. The player can defeat a benchmark player with high probabilities.

1 INTRODUCTION

In 1985, Alexey Pajitnov et al. first invented the puzzle video game Tetris. It has now spread to almost all game consoles and desktop systems.

The standard Tetris is a stochastic, open-end board game. Players control the falling pieces in the game field and try to build fully occupied rows, which are removed in each turn. The standard Tetris field contains 10×20 cells, and the pieces are randomly chosen from 7 pre-defined shapes of blocks. In addition to the current piece, the information of the next piece is provided to the player. Removing rows in a single turn has certain rewards, which in our approach is given as 0.1 for a single row, and 0.3, 0.6, 1.0 for two to four removed rows. This encourages the players to remove multiple rows instead of only one row in one turn. The challenge of this game is that, the player needs to place the pieces in the proper positions in the game field in order to best accommodate the next pieces and remove as many rows as possible. An inappropriate placement of one piece often results in a bad situation of the game, and causes the player to spend more time to deal with. The game is over if the top of the game field is occupied by blocks and the next piece cannot be placed onto the field.

In the two or more players' competitions, if one player has removed $n(n > 1)$ rows in one turn, all other players will receive an attack of $(n - 1)$ rows of blocks, adding to the bottom of their game fields. Each attack row would contain $(n - 1)$ empty cells in random positions. Thus removing multiple rows in single turns brings even more benefits than rewards. Highly skilled human players prefer to plan and remove three or even four rows using a single falling piece, while beginners and many of the existing Tetris artificial players tend to remove rows as soon as possible in each turn to survive the game. The game is over when only one player is still alive in the competition, and of course the last player is the winner.

Researchers have created many artificial players for the Tetris game using various approaches[12]. To the best of our knowledge, most of the existing players rely on evaluation functions, and the search methods are usually given less focus. The Tetris player developed by Szita et al. in 2006[11] employed the noisy cross-entropy method, but the player had a planner for only one piece. In 2003, Fahey had developed an artificial player that declared to be able to remove millions of rows in a single-player Tetris game using a two-piece planner[6]. Later in 2005, genetic algorithms were introduced to the Tetris players by Böhm et al.[3], in which a heuristic function was evolved by using an evolutionary algorithm. In our previous work, we have developed an artificial player with a one-piece planner by using learning by imitation[14] that could successfully play against Fahey's player in the Tetris competitions.

Yet most of the existing artificial players known are based on a planner for only one- or two-piece. This paper was motivated by creating an artificial player based on the planning of a long sequence of pieces. We modeled our player in Tetris planning problem with the Monte-Carlo planning method. In order to balance the exploration-exploitation trade-offs in the planning process, we employed the bandit algorithm to guide the planning process. As for state revisiting, we created a method to store the visited game states in a specially designed database. We also created a hash function to quickly locate and operate the information of a given game state in the database. In order to reduce the branching factor of Tetris planning, we created an intuitive evaluation function and combined it with the UCT algorithm.

The highlights of this paper can be summarized as follows:

- We modeled the artificial player of Tetris using the UCT algorithm.
- Our method of the database of the visited states provided support to UCT and improved the performance of the planner.
- By pruning the planning tree, the player can defeat the artificial player developed by Fehey, which is regarded as the benchmark.

This paper is structured in the following manner. First in section 2, we present our solution on modeling the Tetris planning problem with the bandit-based Monte-Carlo planning method. Our method to design the knowledge database and store the information of the visited game states is presented in section 3. The idea of combining the evaluation function to the UCT algorithm is discussed in section 4. The experiments and the results are shown and analyzed in detail in section 5. In the final section 6, we draw the conclusion and discuss the future work.

[1] Universtiy of Freiburg, Germany, email: caiz, zhangd, nebel@informatik.uni-freiburg.de

1.1 Related Works

To create an artificial player for a board game, the general components are the search method and the evaluation function. The board games which are solvable by brute-force methods, such as Othello, have already been dominated by game programs using various search methods, such as LOGISTELLO [5]. Board games such as Checkers are solvable using knowledge database combined with search methods, one such example is the program named CHINOOK [10]. Many board games, e.g. Chess and Go, are currently unsolvable, thus are still challenging tasks for artificial intelligence researchers. To improve the performance of the artificial players for these board games, one of the tasks for the researchers is to balance the trade-offs between the search depths and evaluation functions [2].

The Monte-Carlo planning method (MCP) has offered a new solution to artificial players of board games. In 1993, Bernd first modeled the board game Go with the MCP algorithm [4], and his Go player had a playing strength of about 25 kyu[2] on a 9×9 board. Soon the MCP method was successfully applied in other board games, such as Backgammon[8]. In 2006, Levente Kocsis and Csaba Szepesvri developed a new search technique named UCT, which stands for *Upper Confidence Bound applied to Trees* [7], and proved that UCT to be more efficient than its alternatives in several domains. Instead of uniform sampling of the game actions, UCT uses the multi-armed bandit algorithm to guide the action selection of the planning process. Later applications using the technique, such as MoGo[3], demonstrated that this technique can be successfully applied to the game of Go.

Learning techniques have also been applied to improve the perfomance of artificial players of board games. The first such approach was the one by Samuelson in 1959 [9]. He was able to show how a program can learn to play Checkers by playing against itself. In 2010, Takuma Toyoda and Yoshiyuki Kotani suggested the idea of using previous simulated game results to improve the performance of the original Monte-Carlo Go program [13], and their work announced positive results on the larger Go board. In Tetris, Böhm et al. used genetic algorithms for the heuristic function, and our previous work had introduced learning by imitation to the artificial player of multi-player Tetris games[3].

Yet to the best of our knowledge, UCT has not been applied in artificial players for Tetris.

2 PLANNING TETRIS USING UCT

In this Section, we discuss how we model the Tetris planning problem using the UCT algorithm.

There are two possible values for every cell in the game field, e.g. occupied and unoccupied, so the standard Tetris search space consists of 2^{200} game states. The branching factor is 162 for a given game state without the piece information, which is the sum of all possible placements of actions from the 7 different pieces. The large branching factor brings us to the idea of using the Monte-Carlo planning method in our solution to the artificial Tetris player. The core feature of the Monte-Carlo planning is to sample as many future states as possible from all actions of the given state of the game for a certain period of time, and for each episode evaluate only the leaf state using a fast evaluation function. In the end, the algorithm takes the action with the best evaluated reward in the planning as the result of the algorithm.

[2] In Go, the rank of $30 - -20$ kyu refers to a *Beginner* level.
[3] Website: http://www.lri.fr/ gelly/MoGo.htm

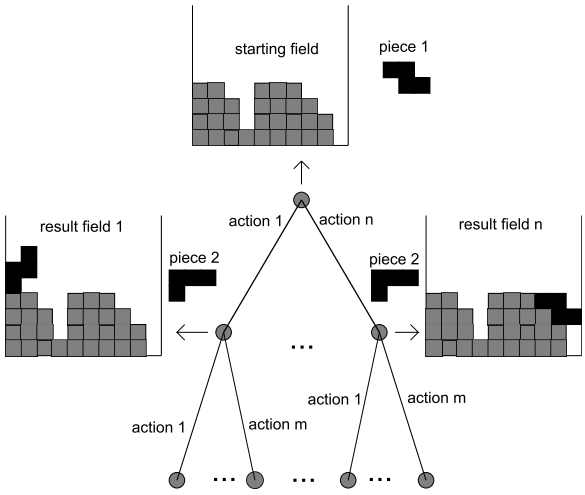


Figure 1: Node of game field state and piece in planning tree

A sample of the planning tree is shown in the Figure 1. In our model of the Tetris planning, we consider each state of the game field, together with a given piece, as a single node in the planning tree. For instance, the root node consists of a game field, which is the rectangular area with gray square blocks inside, and a piece, in this case a "Z" shaped piece displayed by four black square blocks. The fields in the nodes are the so-called "cleared fields", which means that no fully occupied, removable rows are contained in such fields.

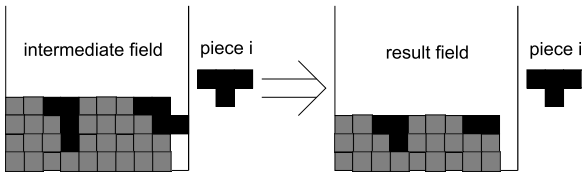


Figure 2: Procedure of removing fully occupied rows

The paths connecting the nodes of the planning tree represent the actions associated to the given piece. By following a path of a starting node, the piece is added to the game field according to the index-encoded action, and an intermediate field is generated. The field will then be checked for removable rows, and if there are rows removed in the field, such rows are removed and the reward will be given according to the game rules. This procedure is described in Figure 2. Then the result field, together with the next piece in the sequence, form a new node, which is the child node of the starting node.

2.1 Planner Structure

The pseudo code describing our planner is displayed in Algorithm 1 and 2.

In the beginning of a planning episode, a state of the game field and a sequence of pieces are given as the inputs to the planner. The planner initiates the planning tree and starts its planning phases. The ranks of the paths in the planning tree are calculated directly by using the rewards gained from performing the actions and removing the fully occupied rows in the field. The rewards are based on the game rules discussed in Section 1.The planner continues to run as many phases as possible until a certain time out rule is reached, and returns the action by which the highest reward is gained. The subtree of the node following the path of the selected action is preserved for future

```
Input: state, list of pieces
Output: action
initialization;
while not time out do
    search(state, first piece);
end
action←selectBest(state, pieces);
updateTree();
return action;
```

Algorithm 1: One Planning Episode : Function **doPlanning**

planning, and other nodes are deleted at the end of each planning episode to reduce memory consumption.

```
Input: state, piece
Output: reward
type←stateType(state, piece);
switch type do
    case type == normal node
        action←selectAction(state, piece);
        state,reward←performAction(state, piece,
        action);
        updateTree(state, piece, action, reward);
        return reward;
    case type == leaf node
        return 0;
    case type == terminal node
        return -1;

endsw
```

Algorithm 2: One Planning Phase : Function **search**

In each planning phase, the planner selects and performs one of the actions for each piece in the given sequence. Each performed action results in a child node in the planning tree, together with a reward accordingly. A leaf node is reached once all the pieces in the given sequence have been performed with an action. The planner then sums up the total reward gained in the current search path, and updates the reward information in each node in the search path backwards from the leaf to the root.

In Tetris, a common game state usually does not have any information on the winning chance. To simplify the evaluations on the leaf node, we consider it to be with zero reward, which means leaf nodes do not have any influence on their parent nodes in the search path. One exception is that, if a node contains a state of the field which is, according to Tetris rule, a terminal state, the reward is always set to a big negative value. Such behavior makes the actions leading to terminal states much less likely to be chosen by the planner during the planning phases.

The method for sampling the actions is the key feature of the Monte-Carlo planning. Comparing to the traditional uniformed and randomized sampling methods, the multi-armed bandit algorithm has advantages in balancing the trade-offs between explorations and exploitations during the planning process, and is proved to be more efficient than other methods in many domains[7].

2.2 Bandit Algorithm

To model the action selection using the bandit algorithm, we consider each state of the Tetris game field, together with a given piece, as a separate K-armed bandit machine, where K is the number of possible actions for the piece given.

The action selection is performed for each visiting node in the current search path. According to the algorithm UCB1[1], the action selection is based on the upper confidence bound of the reward and bandit score of every arm (action) of the bandit machine (game state), which is described by Formula 1.

$$I = \underset{i \in 1,\ldots,K}{\arg\max}\{R_i + c_i\} \tag{1}$$

The R_i in the formula is the reward from performing the action i and removing the rows in the result game field. The c_i is a bias sequence chosen as:

$$c_i = \lambda\sqrt{\frac{\ln s}{t}} \tag{2}$$

where λ is a constant factor manually chosen for balancing the exploration-exploitation trade-offs. Higher λ values result in higher chances of randomized explorations based on the bandit scores, while smaller λ values lead to higher possibilities of selective exploitations according to the rewards. The number of visits s to every node in the search tree and the number of visits t to each action path of the node are kept and updated throughout the planning episode.

In the procedure of action selection, the bandit score of each action of the given piece is first calculated by using the number of visits of the node and the action. Then, the gained reward is added to the bandit score, and the sums of the two are used to rank the actions. The action with the highest sum is chosen to be the final result of the procedure. If multiple actions have the same highest sum, the result is chosen randomly from the list of such actions. Notice that the formula 2 will be invalid for the nodes in which some of the actions have never been visited before. For such nodes, an action is selected randomly from the unvisited actions.

In the standard UCT algorithm, only the sub-tree of the node following the selected action path will be kept for the next planning episode. For board games where game states are less likely to be revisited in the future plays of the game, such behavior would have little influence on the future planning process. But the state revisiting happens quite often in Tetris because of the game rules. Therefore, the information of the explored states in previous planning episodes would play an important role in the planning. In the next section, we will discuss the state revisiting of Tetris.

3 STATE REVISITING

Before designing our database for the visited game states, we think of the information that is useful for the future planning episodes. First, we want to start each planning episode of a root node from scratch. So the information of the number of visits to nodes and actions is not to be stored, because such information is a bias to the given sequence of pieces of the previous planning episodes. The immediate rewards and targeting states of the actions associated to one node can be easily and fast computed in the planning phases, thus the information can also be ignored.

In our method of planning, a node in the planning tree is made of a state of the game field and a given piece, and the information of the node consists of the following components which is necessary to be stored:

1. The highest reward among all the actions, and
2. The reward of each action.

The information of the item 1 is the key to our idea of storing and reusing the information of the explored game states, because it represents the summarized results of its associated previous planning episodes, and can be easily combined with the results of any future planning episodes. The information of the item 2 can be abstracted to a list of actions that matches the highest reward, which can be combined easily with the future planning results.

Considering the planner of the artificial player plays 100 pieces in a single game, and for each planning episode, the planner explores $1,000$ phases. Then the total number of explored states of a single game is approximately $100,000$. Assume that one quarter of these explored states are revisited states, then in the end there are $75,000$ newly explored states in such a game. Rather than saving every single node in the planning tree, we store only the root node in every planning episode. In this way, the size of the stored nodes is significantly reduced, whilst the most useful information of each planning episode is preserved. Since the root node would only be queried once per episode, the time cost for system operations of the database is ignorable.

Now that we have our information stored in a database, the final task is to find a fast and easy way to load and save such information. In the following section, we will present our design of the database using hash functions.

3.1 Hashing In Database

Although not all of the game states will be explored and stored in our approach, locating a specific state in a huge amount of data is not an easy task. One of the possible options for locating a state is to use an existing database management systems. However, such systems are inappropriate for our approach, as the data we want to store are small in single sizes and have little relation to each other.

In our early approach, we tried to store the data in a single file, which is easy to implement. But locating the data of a certain game state in the file is time-consuming. One of the possible methods is to use the "sparse file" system for storage, and every state is stored to a certain position in the "sparse file", where positions are calculated using a hash function. Since there is an "offset limit" in the size of a single "sparse file", it is difficult to create a perfect hash function to generate positions for the 2^{200} states in Tetris without collisions.

Another option is to use rather a simple file system with each state stored in a separate file. Like the "sparse file" system, the simple file system is also dependent on the operating system to locate the entry of a certain file. The difference is that the hash function can easily be created for the simple file system, as there are few limits to the name and path of a file. Also because game states are stored in separate files, there will be no "offset limit" of a single file, and thus the collisions of positions are avoided.

For any file system, the key issue is to balance the number of files and folders in one folder, the maximum depth of folders, and the size of each file containing the data. Too many files or subfolders in one folder could require more time for the operating system to locate the entry of a target file in the disk. The size of each file directly affects the computation time to store and retrieve the data for the program.

In our approach, each state of the game field generates a file name according to the values of the field encoded by a vector of integers. In this way, every state would have a unique file name, and collisions are avoided. Another advantage is that, the data of the game field is hidden inside the name of the file. And from another point of view, this hashing method reduces the size of the storage.

Then, all such files are separated into different folders in a folder tree of 5 depths. In each depth of the folder, there are up to 16 subfolders. This scattering method is used to avoid too many subfolders or files in a single folder. Our later experiment on random sequences of pieces showed that, for the Linux operating system, the computational time had less than 1% differences in runtime from the beginning to the end of the experiment, where the number of the saved states in the database increased from zero to $100,000$.

```
Input: state, pieces
Output: action
1 initialization;
2 while not time out do
3 |   search(state, pieces, 0);
4 end
5 combineKnowledge(state, pieces);
6 action←selectBest(state, pieces);
7 updateTreeEx();
8 return action;
```

Algorithm 3: One Planning Episode : Modified Function **doPlanningEx**

In each planning episode, the planner starts planning from scratch, using only the information of the game state, the piece sequence, and the current planning tree. Then after the planning phases are all completed, the planner retrieves the information of the root node from the database. Such information is combined with the newly explored information of the root node. The combination rule is simple. If the highest reward of the root node in the database is bigger than that in the planning tree, the information in the database will completely override the information in the planning tree, and vice-versa. If the two rewards are the same, then the lists of actions matching the highest reward of both the database and the planning tree will be merged. Algorithm 3 shows a modification to its previous version discussed in Section 2.1.

4 PRUNING THE PLANNING TREE

The previously introduced method is based on the sampling of all possible actions of the given piece in the given game state. However, many of the actions are not worth exploring, because they often lead to unwanted or even bad game states. In this section, we created a method to prune the planning tree to reduce the number of actions that need to be sampled, and improved the performance of the developed player.

From records of many Tetris games played by artificial and human players, we studied that one of the most important features of a proper placement of a given piece in the game field is to avoid creating the "*hole*" shapes in the field, as most of the skilled human players would usually do. Thus, it is intuitive to distinguish between a good game state and a bad one by examine the number of the "*hole*" shapes contained in such game fields.

Since many of the possible actions of a given piece would create the "*hole*" shapes in their result game fields, we created a method to prune these actions from the planning tree. The pruning is based on the increment of holes from the original game field to the result game field from performing an action. In addition to the reward and the bandit score of each action discussed in Section 2.2, the number

of holes created by the action is considered as a parameter with a negative effect on the sum of the former two. This behavior results in a lower possibility of an action to be chosen during the action selection process, if there are more "*hole*" shapes created from the action. The Equation 1 is then modified as:

$$I = \underset{i \in 1,\ldots,K}{\arg\max}\{R_i + c_i + P_i\} \tag{3}$$

where P_i is a negative value defined according to the number of holes created, and is defined by the following equation:

$$P_i = \gamma H_i, (-1 \leq \gamma \leq 0) \tag{4}$$

where H_i is the number of holes created by performing the action i to the game field, and γ is a negative factor according to the number of rows removed from the result game field. The reason for γ being different is that, unlike the actions that can only create holes in the field, the actions that can both remove multiple rows and create some holes may still lead to a good result game state, and thus should have a certain chance to be explored.

5 EXPERIMENTS

We have conducted three experiments for our artificial Tetris player.

The first experiment was meant to test the validity of our method. In order to measure the performances of the developed player when using the database of the visited game states to support the standard UCT algorithm, we started the experiment on an empty database, and let the player repeatedly play Tetris on a fixed sequence of pieces. Two parameters of the game are to be evaluated: a) the final score of the game, and b) the ratio of the roll-outs in one planning episode of our approach comparing to the standard UCT algorithm. The former parameter stands directly for the performance of the developed player, and the latter indicates the effectiveness of the database in the future planning process.

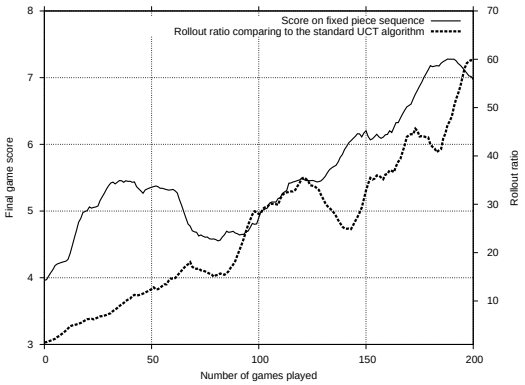


Figure 3: Experiment on a fixed piece sequence

The results of the first experiment are shown in Figure 3. The solid line in the figure displays the final score of each game. The game score is based on the given game rule described in Section 1. We can see that the final score grows as the number of played games increases. This shows that our method to store the information of the visited game states and reuse it in future planning process can successfully support the standard UCT algorithm and improve the performance.

Comparing to the standard UCT algorithm as a basis for the number of roll-outs per planning episode, the roll-out ratio of each planning process of our approach is shown in the figure with the thick dashed line. The number of roll-outs of a state is piled up when the state is revisited in the future planning episodes. The results show that the knowledge database helps the standard UCT algorithm to do more roll-outs during the planning process when the states are revisited, and hence improves the performance.

One observation in the experiment is that, although the trend of the two parameters is going in a growing manner, there exist some falls of scores and ratios at some point of the experiment. After analyzing this phenomenon, we found out the reason is that at some point of the game, some newly explored actions produced rewards which are higher than those in the previous planning episodes. This resulted in the changes of choices of the actions for such pieces, and lead to some future states which are brand new to the player's knowledge database. We can also see in the figure that after some more game plays, the final scores went up again as such states were covered by the player's database.

The second experiment was designed to analyze the total size of the database and the coverage percentage of the visited game states. Unlike the first experiment on a fixed piece sequence, we let our player continually playing the Tetris games on randomly generated piece sequences. The main idea is to let the player explore as many unvisited game states as possible, and analyze the experiment results.

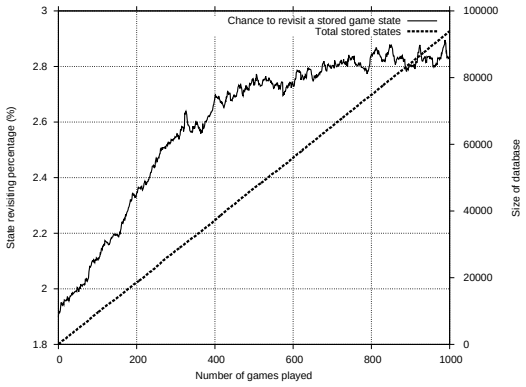


Figure 4: Experiment on random piece sequences

From the results of the second experiment displayed with the dashed line in Figure 4, we can see that the total size of the stored game states in the knowledge database is growing with a constant factor. A conclusion can be drawn that given randomly generated piece sequences, the player is able to increase its knowledge of the Tetris game.

Shown with the solid line in the figure, as more visited game states are stored in the database, the coverage percentage of the revisited game states is increasing. This means that it is more likely to revisit a previously explored game state, even when the game has a completely different piece sequence. On the other hand, the percentage is still low, which means that currently the database is not big enough to cover the most useful game states in Tetris.

Combining the results of the first two experiments provides evidence that our artificial Tetris player can successfully play the standard Tetris game, and has the ability to learn from the played games and improve its future performance with the help of the knowledge database of the previously visited game states. By repeatedly playing the Tetris games using randomly generated piece sequences, the performance of the player will improve, as more game states will be covered by its knowledge database.

The third experiment was the competitions against Fahey's benchmark player[6] using different approaches. The winning percentages

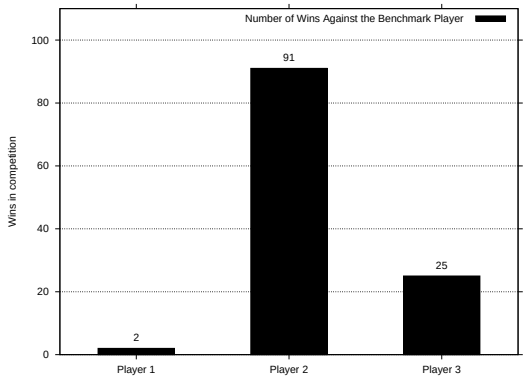


Figure 5: Competitions between players

are calculated on a basis of 100 rounds of games, and the results are shown in Figure 5. The three columns represent the different approaches used to develop the player: 1) the UCT based player with only the knowledge database, 2) the UCT based player with both the database and the pruning method, and 3) the SVM based pattern player in our previous work[14].

As we can see in the figure, with only the knowledge database, the player's performance against the benchmark is very low. With the support of the pruning method, the performance of the player is significantly improved. As shown in Figure 5, the number of wins increases from 2 to 91 over 100 games, and is competitive comparing to our previous work of the SVM based pattern player, which has a result of 25 wins out of 100 games in the competition.

There is a trade-off in including a pruning method in our approach. The complexity of the included method affects the efficiency of the planner in both the action selection and the roll-out sampling. According to the statistics of the experiments, the number of roll-outs per planning episode dropped by $1/2$ when the pruning method is included, while the number of pieces successfully played per game raised by 25 times. Conclusion can be drawn that such method is suitable in our approach to improve the performance of the developed player, with little loss of efficiency.

6 CONCLUSIONS

In this paper, we have developed an artificial Tetris player using the bandit-based Monte-Carlo planning method. Different from many existing artificial Tetris players, our player is built on the ten-piece planner. Our idea is to use the Monte-Carlo planning method to sample the possible actions of the given pieces in the game field, and use the bandit algorithm to balance the exploration-exploitation trade-offs and guide the planning process.

One of the key challenges of our work is to find a good solution to make use of the information of the visited states during the planning process, as such information is not kept and reused in the standard UCT algorithm. We created a method to store the information of the visited game states in a specially created database file system. The information can be loaded and reused in the future planning episodes when the stored states are revisited, and such a scheme provides the developed player with the learning ability.

The high branching factor causes the planner to spend much of its time exploring possible actions, while many of such actions are useless and often lead to unwanted game states. We created a method to prune the planning tree during the planning process to reduce the number of actions to be explored, and improved the game performance of the player.

The experiment results show that our player can successfully play the Tetris game. By using the stored information of the visited game states as a support to the standard UCT algorithm, the results of the experiments show that the performance of the player improves as the number of played games increases. The player could explore the unvisited Tetris game states using randomly generated piece sequences and improves its game performance. With the pruning method, the developed player has significantly higher chance to win a multi-player Tetris game in competition against the benchmark of the Tetris players.

6.1 Future Work

The results of our second experiment on randomly generated piece sequences showed that our database has not yet covered a high percentage of useful game states. In the next step, we will continue the experiment on exploring unvisited game states for the database, and analyze the effects of larger database on the Tetris games.

Currently we use an intuitive method to prune the planning tree. Although the overall performance of the developed player is improved, careful studies are needed to analyze the trade-offs between more complex pruning methods and the changes in the performance. This is another interesting topic be in our future plans.

REFERENCES

[1] Peter Auer and Jyrki Kivinen, 'Finite-time analysis of the multiarmed bandit problem', in *Machine Learning*, pp. 235–256, (2002).

[2] Hans J. Berliner, Gordon Goetsch, Murray Campbell, and Carl Ebeling, 'Measuring the performance potential of Chess programs', *Artif. Intell.*, **43**(1), 7–20, (1990).

[3] Niko Böhm, Gabriella Kókai, and Stefan Mandl. An evolutionary approach to Tetris, 2005. In Proceedings of the sixth metaheuristics international conference (MIC2005).

[4] Bernd Brügmann. Monte-Carlo go, 1993. Unpublished technical report.

[5] Michael Buro, 'From simple features to sophisticated evaluation functions', in *Computers and Games, Proceedings of CG98, LNCS 1558*, pp. 126–145. Springer-Verlag, (1999).

[6] Colin Fehey. Tetris AI. `http://www.colinfahey.com/tetris/tetris_en.html`, 2003. www accessed on 02-August-2010.

[7] Levente Kocsis and Csaba Szepesvri, 'Bandit based Monte-Carlo planning', in *In: ECML-06. Number 4212 in LNCS*, pp. 282–293. Springer, (2006).

[8] François Van Lishout, Guillaume Chaslot, and Jos W.H.M. Uiterwijk, 'Monte-Carlo tree search in Backgammon', *Computer Games Workshop*, 175–184, (2007).

[9] Arthur L. Samuel, 'Some studies in machine learning using the game of Checkers', *IBM Journal of Research and Development*, **3**(3), 210–229, (1959).

[10] Jonathan Schaeffer and Robert Lake, 'Solving the game of Checkers', in *Games of No Chance*, pp. 119–136. Cambridge University Press, (1996).

[11] István Szita and András Lőrincz 2, 'Learning Tetris using the noisy cross-entropy method', *Neural Computation*, **18**, 2936–2941, (2006).

[12] Christophe Thiery and Bruno Scherrer, 'Building controllers for Tetris', *International Computer Games Association Journal*, **32**, 3–11, (2009).

[13] Takuma Toyoda and Yoshiyuki Kotani, 'Monte Carlo Go using previous simulation results', *Technologies and Applications of Artificial Intelligence, International Conference on*, **0**, 182–186, (2010).

[14] Dapeng Zhang, Zhongjie Cai, and Bernhard Nebel, 'Playing Tetris using learning by imitation', in *GAMEON*, pp. 23–27. EUROSIS, (2010).

Determinization in Monte-Carlo Tree Search for the card game Dou Di Zhu

Edward Powley[1], **Daniel Whitehouse**[1], **and Peter Cowling**[1]

Abstract. *Monte-Carlo Tree Search (MCTS)* is a class of game tree search algorithms that have recently proven successful for deterministic games of perfect information, particularly the game of Go. *Determinization* is an AI technique for making decisions in stochastic games of imperfect information by analysing several instances of the equivalent deterministic game of perfect information. In this paper we combine determinization techniques with MCTS for the popular Chinese card game Dou Di Zhu. In determinized MCTS, there is a trade-off between the number of determinizations searched and the time spent searching each one; however, we show that this trade-off does not significantly affect the performance of determinized MCTS, as long as both quantities are sufficiently large. We also show that the ability to see opponents' hidden cards in Dou Di Zhu is a significant advantage, which suggests that inference techniques could potentially lead to much stronger play.

1 INTRODUCTION

From the inception of the field of game AI until relatively recently, the vast majority of research has focussed on games that are *deterministic* (do not have chance events) and have *perfect information* (all players can observe the exact state of the game). Chess [1] and checkers [2] are two well-known and well-studied examples of such games.

One recent innovation in AI for deterministic games of perfect information is *Monte-Carlo Tree Search (MCTS)*. MCTS iteratively builds a partial search tree, using random simulated games to evaluate nonterminal states. This makes it well-suited to games with long trajectories and high branching factor, as well as games where good heuristic state evaluation functions are difficult to find. The game of Go exemplifies both of these properties; MCTS has proven particularly successful for Go [3], and it is this success that has fuelled much recent interest in MCTS.

Work on AI for games that are *stochastic* (have chance events) or have *imperfect information* (the state of the game is only partially observable) is comparatively recent. One popular approach to such games is *determinization*. The game is reduced to several instances of a deterministic game of perfect information (called *determinizations*), by randomly fixing the values of future chance events and hidden information. Each of these determinizations is analysed by AI techniques for deterministic games of perfect information and the results are combined to yield a decision for the original game. Although it is not a fool-proof approach for all games, determinization has proven successful in games such as bridge [4] and Klondike solitaire [5], as well as the domain of probabilistic planning [6].

There is an important trade-off to be made in determinization, between the number of determinizations, and the amount of effort spent on analysing each one. However, in this paper we present experimental results suggesting that this trade-off is not as important as it may appear at first: as long as both parameters are sufficiently large, adjusting the balance between them does not have a significant effect on playing strength.

It is common in games of imperfect information for certain pieces of information to be visible to some players but hidden from others. An important technique in such games is the ability to infer this information by observing the actions of other players. An upper bound can be placed on the performance gain that can be achieved by inference by measuring the performance of a player able to observe the exact state of the game, effectively modelling a player who is able to instantly and perfectly infer any hidden information. Using this method, we show that the potential benefit of inference is significant: if two cooperating players can observe the hidden information belonging to each other and to their opponent, then they can improve their win rate by around 21%.

The structure of this paper is as follows. Section 2 reviews existing work on MCTS and on AI for stochastic games of imperfect information. Section 3 introduces the card game Dou Di Zhu, which serves as our application domain for the remainder of this paper. Section 0 describes the version of determinized MCTS we use, and addresses some technical issues of how this is applied to Dou Di Zhu. Section 5 presents experimental results, including measurement of the variance in win rates for Dou Di Zhu (Section 5.1), the trade-off between number of determinizations and number of MCTS iterations per determinization (Section 5.2), and the potential increase in playing strength obtainable through inference of hidden information (Section 5.3). Finally, Section 6 gives some concluding remarks and directions for future work.

2 LITERATURE REVIEW AND BACKGROUND

2.1 Preliminaries

This section introduces briefly the notions of *uncertainty* (stochasticity and/or imperfect information) in games. For more details we refer the reader to a standard textbook on game theory, e.g. [7].

A *game* can be thought of as a multi-agent Markov decision process; that is, a Markov process where, at each state, one of the agents (or *players*) is allowed to make a decision which influences

[1]Artificial Intelligence Research Centre, School of Computing, Informatics and Media, University of Bradford, UK. Email: {**e.powley**, **d.whitehouse1**, **p.i.cowling**}**@bradford.ac.uk**.

the transition to the next state. A game is said to be *deterministic* if taking a particular action from a given state always leads to the same state transition. If a game is not deterministic, i.e. there is some element of randomness (or *chance*) to the transitions, the game is said to be *stochastic*. A game has *perfect information* if all players are able to observe the current state of the game. If a game does not have perfect information, i.e. the underlying Markov process is partially observable, the game is said to have *imperfect information*. For example, many card games are stochastic (because they are played with a shuffled deck) with imperfect information (because no player is able to observe the cards held by the other players). Where there are stochastic state transitions, we may also regard this as a deterministic game where chance outcomes (e.g. the order of cards in a shuffled deck) are decided in advance, but are hidden from all players. This way of thinking gives rise to the determinization approach we will discuss later.

In a game of imperfect information, the states as observed by each player are partitioned into *information sets*, where an information set is defined as a set of states that are indistinguishable from the player's point of view. During the game, the player cannot observe the current state, but can observe the current information set.

2.2 Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS) is a class of game tree search algorithms that make use of simulated games to evaluate non-terminal states. Simulated games select random actions until a terminal state is reached and the reward is averaged over multiple simulations to estimate the strength of each action. MCTS algorithms have gained in popularity in recent years due to their success in the field of computer Go [3]. In particular the *UCT* algorithm [8] proposed in 2006 has led to the recent upsurge of interest in MCTS algorithms. UCT enables a selective search to be performed where only the most promising areas of the tree are searched. This is particularly useful in domains with a large branching factor.

MCTS algorithms build a sub-tree of the entire decision tree where usually one new node is added after every simulation. Each node stores estimates of the rewards obtained by selecting each action and an improved estimate is available after every simulation step. In the case of the UCT algorithm, each decision in the tree is treated as a Multi-Armed Bandit problem where the arms are actions, and the rewards are the results of performing a Monte-Carlo simulation after selecting that action. The UCB algorithm [9] is used to guide the selection of actions at each decision in the tree. The UCB algorithm ensures that the tree will be selectively explored, performing a deeper search in more promising areas of the tree.

There has been some research on modifying UCT to achieve better performance for particular domains (such as Go), with the RAVE heuristic [10] being an example of such a modification. Also the algorithm can be executed in parallel [11] allowing MCTS agents to exploit modern multi-core and multi-processor hardware. UCT has been successfully applied to a wide variety of challenging domains such as General Game Playing [12], where the rules of the game are not known in advance and an AI player must use a very general approach which does not depend upon specific game knowledge. UCT requires very little domain knowledge and is therefore useful in domains where no good heuristics are known.

The structure of MCTS algorithms is generally quite similar: a discrete number of iterations are performed, after which an action from the root node is selected according to statistics collected about each action. Each simulation step performs four operations on the sub-tree built by the algorithm, namely Selection, Expansion, Simulation and Back-Propagation.

2.2.1 Selection

Starting from the root node, an action is selected at each node in the current partial game tree, according to some criterion. This mechanism is applied iteratively until it reaches either a terminal game state or a node which is not in the current partial game tree.

In the case of the UCT algorithm the mechanism used for traversing the tree is the UCB algorithm. After initially selecting every action once, the UCB algorithm selects the action which maximizes

$$\frac{\sum_{t=1}^{v_j(n)} X_{j,t}}{v_j(n)} + k\sqrt{\frac{\ln(n)}{v_j(n)}} \qquad (1)$$

where $X_{j,t}$ denotes the reward obtained by selecting arm j on trial t, n is the number of times the current node has been visited previously, $v_j(n)$ is the number of these visits in which action j was selected, and k is a constant controlling the balance between exploration and exploitation (often $k = \sqrt{2}$). The UCB algorithm balances exploration of untried moves with exploitation of known good moves. This enables UCT to find the most promising areas of the tree and search these further.

2.2.2 Expansion

If the state reached is terminal then there is no need to expand the tree or perform a simulation and the final result of the game is Back-Propagated. If the state is not terminal then it is added to the tree.

2.2.3 Simulation

Starting from the node added in the expansion phase (or the selected leaf node if no node was added), the game is played out to completion by choosing random moves for each player. The result of this simulated game is then used to update the statistics in the tree during Back-Propagation. Choosing an effective method for quickly selecting random moves which give rise to unbiased game simulations, and which are also realistic enough to be strongly correlated with the eventual winner from a position, appears crucial to strong play in MCTS, particularly for computer Go [13].

2.2.4 Back-Propagation

Each node visited during selection has its statistics updated. In the case of the UCT algorithm the number of times the node has been visited and the cumulative reward of simulations passing through that node (from the point of view of the player selecting the action at the node) are recorded. These are then used in turn by the UCB algorithm during the next selection phase.

2.3 AI for games with uncertainty

This section briefly surveys research on AI for games with stochasticity and/or imperfect information.

2.3.1 Determinization

One approach to designing AI for games with stochasticity and/or imperfect information is *determinization*, also known as *perfect information Monte Carlo (PIMC)*. For an instance of a stochastic game with imperfect information, a *determinization* is an instance of the equivalent deterministic game of perfect information, in which the current state is chosen from the AI agent's current information set, and the outcomes of all future chance events are fixed and known. For example, a determinization of a card game is an instance of the game where all players' cards, and the shuffled deck, are visible to all players. We then create several determinizations from the current game state, analyse each one using AI techniques for deterministic games of perfect information, and combine these decisions to yield a decision for the original game.

Ginsberg's GIB system [4] applies determinization to create an AI player for the card game Bridge which plays at the level of human experts. GIB begins by sampling a set D of card deals consistent with the current state of the game. (The question of how D is sampled is rather more subtle than it first appears, and indeed Ginsberg [4] does not give complete details). For each of these deals $d \in D$ and for each available action a, the perfect information ("double dummy") game is searched to find the score $\mu(a, d)$ resulting from playing action a in determinization d. The search uses a highly optimised exhaustive search of the double dummy Bridge game tree. Finally, GIB chooses the action for which the sum $\sum_{d \in D} \mu(a, d)$ is maximal.

A domain closely related to (single-player) stochastic games of imperfect information is that of probabilistic planning. Yoon et al's FF-Replan [6] and FF-Hindsight [14] systems apply determinization to probabilistic planning problems. Despite the simplicity of its approach, FF-Replan outperformed more complex systems to win the IPPC-04 and perform well in the IPPC-06 probabilistic planning competitions.

Bjarnason et al [5] present a variant of UCT for stochastic games, called *Sparse UCT* and apply it to the single-player card game of Klondike Solitaire. In a stochastic game, a single state-action pair can lead to multiple successor states (corresponding to the possible outcomes of the chance event); Sparse UCT handles this by allowing multiple child nodes for each action from each state. The selection phase selects actions using UCB, but the traversal to child nodes is stochastic, as is the addition of child nodes during expansion. Sparse UCT imposes an upper limit w on the number of child nodes for each state-action pair.

Bjarnason et al [5] also study an *ensemble* version of Sparse UCT, in which several search trees are constructed independently and their results (the expected rewards of actions at the root) are averaged. They find that ensemble variants of UCT often produce better results in less time than their single-tree counterparts. The ensemble case with $w = 1$, which Bjarnason et al call *HOP-UCT*, is equivalent to a straightforward application of determinization (more specifically, *hindsight optimisation* [15]) with UCT as deterministic solver, in which the determinization is constructed lazily as UCT encounters each chance event.

Bjarnason et al [5] treat Klondike Solitaire as a stochastic game of perfect information: rather than being fixed from the start of the game, the values of face down cards are determined as chance events at the moment they are revealed. This works for single-player games where the hidden information does not influence the game until it is revealed, but generally does not work for multiplayer games where the hidden information influences the other players' available and chosen actions from the beginning of the game. Hence the specific methods of Sparse UCT and lazy determinization are not immediately applicable to multiplayer games, but the general ideas are transferable.

Bjarnason et al show that Sparse UCT is able to win around 35% of Klondike Solitaire games, which more than doubles the estimated win rate for human players. Determinized MCTS also shows promise in games such as Phantom Go [16] and Phantom Chess (Kriegspiel) [17], among others.

Despite these successes, determinization is not without its critics. Russell and Norvig [18] describe it (somewhat dismissively) as "averaging over clairvoyance". They point out that determinization will never choose to make an information gathering play (i.e. a play that causes an opponent to reveal some hidden information) nor will it make an information hiding play (i.e. a play that avoids revealing some of the agent's hidden information to an opponent). Ginsberg [4] adds weight to this claim by making the same observations about GIB specifically. One explanation for this is that, from the point of view of the decision-making process in each determinization, there is no hidden information to gather or hide.

Russell and Norvig's criticisms of determinization are valid but equally valid are the experimental successes of determinization. Frank and Basin [19] identify two key problems with determinization:

1. *Strategy fusion.* An AI agent can obviously not make different decisions from different states in the same information set (since, by definition, it cannot distinguish such states); however, the deterministic solvers can and do make different decisions in different determinizations.
2. *Non-locality.* Some determinizations may be vanishingly unlikely (rendering their solutions irrelevant to the overall decision process) due to the other players' abilities to direct play away from the corresponding states.

Building on the work of Frank and Basin, Long et al [20] identify three parameters of game trees and show that the effectiveness of determinization has some dependence on the game's position in this parameter space. The parameters measure the ability of a player to influence the outcome of a game in its late stages (*leaf correlation*), the bias in the game towards a particular player (*bias*) and the rate at which hidden information is revealed (*disambiguation*). Long et al [20] demonstrate how these parameters can be used to predict whether determinization is an appropriate method for a given game.

2.3.2 Other approaches

One alternative approach to tree search for stochastic games is *expectimax search* [18]. This is a modification of the well-known minimax algorithm to game trees containing chance nodes. The value of a chance node is the expected value of a randomly chosen child (i.e. the sum of the values of its children weighted by the probabilities of the corresponding chance outcomes). Unfortunately, expectimax is not well-suited to the type of card games studied in this paper, where there is a single chance node with a very large branching factor (corresponding to the initial shuffling of the deck), as the resulting game tree is too large to search even with MCTS.

As noted previously, the field of game AI focussed on deterministic games of perfect information until relatively recently. The same is not true in the field of game theory, where stochastic games of imperfect information have been well studied [7]. Thus a popular approach to AI for such games is to compute (or approximate) a Nash equilibrium strategy; examples of this approach include Gala [21] and counterfactual regret [22]. While there is undeniable appeal in finding a strategy that is provably optimal, we feel that searching for Nash equilibria is often not the

most appropriate approach to building strong game AI. The definition of Nash equilibrium requires only that the strategy is optimal against other optimal (Nash) strategies, so Nash strategies often fail to fully exploit suboptimal opponents.

2.3.3 Inference

In games of imperfect information, it is often possible to infer hidden information by observing the actions of the other players, according to some model of the other players' decision processes. This type of inference has frequently been applied to the game of poker [23-25], but also to other games such as Scrabble [26] and the card game Skat [27] which has similarities to the card game Dou Di Zhu which we study in this paper. Developing an inference engine for Dou Di Zhu is a subject of current and future work. Section 5.3 discusses the increase in playing strength we expect to see from a strong inference engine.

3 DOU DI ZHU

3.1 History and Popularity of Dou Di Zhu

Dou Di Zhu [28] is a 3-player gambling card game which originated in China. It falls into the class of Climbing Games but also has similar elements to Trick Taking Games. The name Dou Di Zhu translates into English as "Fight The Landlord" and is a reference to the class struggle during the Cultural Revolution in China where peasants were authorized to violate the human rights of their Landlords. In the original version of the game two players play compete together against a third player, the Landlord. There are other versions of the game involving four and five players but these are less popular.

The game was only played in a few regions of China until quite recently, when versions of the game on the Internet have led to an increase in the popularity of the game throughout the whole country. Today Dou Di Zhu is played by millions of people online, although almost exclusively in China, with one website reporting 1,450,000 players per hour. In addition there have been several major Dou Di Zhu tournaments including one in 2008 which attracted 200,000 players.Dou Di Zhu is interesting from an AI perspective as it necessitates both competition (between the Landlord and the other two players) and cooperation (between the two non-Landlord players).

3.2 Rules

Dou Di Zhu[2] uses a standard 52 card deck with the addition of a black joker and a red joker. Suit is irrelevant but the cards are ranked with 3 being the lowest rank and 2 being the highest rank (higher than Ace). The jokers are ranked higher than a two, with the red joker ranked higher than the black joker. Each player is dealt a hand of 17 cards from a shuffled deck and the remaining three cards are placed faced down on the table.

3.2.1 Bidding Phase

Each player takes turns to bid on their hand with the possible bids being 1, 2 or 3 chips. Bids must be strictly higher than the current bid but each player has the option to pass. This continues until two of the players pass consecutively. If any player bids 3 chips then the bidding phase immediately ends. If all three players initially pass, the cards are shuffled and dealt again. The winner of the bidding phase is designated as the *Landlord* and this player adds the three extra cards on the table into their hand, and plays first. The winning bid determines the *stake* for the game.

3.2.2 Card Play Phase

The goal of the game is to be the first to get rid of all cards in hand. If the Landlord wins, the other two players must each pay the stake to the Landlord. However if either of the other two players wins, the Landlord pays the stake to both opponents. This means the two non-Landlord players must cooperate to beat the Landlord. The Landlord always plays first and then play moves around the table in a fixed direction. At the end of the game the stake is doubled if a player has failed to remove any cards from their hand.

The card play takes place in a number of rounds until a player has no cards left. Whoever plays first can play any group of cards from their hand provided this group is a member of one of the legal *move categories* (see Table 1). The next player can play a group of cards from their hand provided this group is in the same category and has a higher rank than the group played by the previous player. If a player has no compatible group they must pass. This continues until two players pass, at which point the next player wins that round and may start a new round by playing a group of cards from any category.

There are a few special rules for some of the categories. If the move being played is a straight run (of singles, pairs or trios) then the next player must play a straight run of the same type and length, ending on a higher ranked card. Straight runs can contain cards of any rank from 3 to Ace, but not cards of rank 2 or jokers. A Bomb or a Nuke (also known as a Rocket) may be played at any point but doubles the stake of the game. A Bomb may be followed by another Bomb of higher rank but not a Quadplex.

Some categories allow extra *kicker* cards to be played with the group which have no effect on the category or rank of the move being played. A kicker can be any card provided it is of different rank to all the cards in the main group. If the kicker cards are single cards they must be of different rank and if the kicker cards are pairs they must be differently ranked pairs. Also a Nuke cannot be used as a kicker. If a move with kickers is played, the next player must play a move in the same category with the same number of kickers, although the ranks of the kicker cards are ignored.

Table 1 summarises the categories of moves in Dou Di Zhu.

Table 1. Description of the different move categories in Dou Di Zhu

Name	Description
Solo	Any individual card, for example A or 2. It is also possible to play runs of sequential cards with length at least 5, for example 345678 or 89TJQKA.
Pair	Any pair of identically ranked cards for example 55 or 77. It is possible to play runs of sequential pairs with length at least 3, for example 334455 or TTJJQQKK.
Trio	Any three identically ranked cards for example AAA or 888. It is possible to play runs of sequential trios of any length, for example 444555 or TTTJJJQQQ. Each trio may also have a kicker attached, for example 444555TJ or 999QQ.
Quadplex	Any four identically ranked cards with two kickers attached, for examples 4444TJ or 999955KK.
Bomb	Any four identically ranked cards, for example 5555 or 2222.
Nuke	The red joker and the black joker together.

[2] There are different versions of the rules of Dou Di Zhu. The rules used in this work were taken from [28].

3.3 Basic Strategy

Dou Di Zhu is a game which requires substantial levels of skill from expert human players. The player who wins the bidding phase usually has a high confidence they were dealt a good hand although they have three extra cards to discard. Often a hand contains lower ranked individual cards that cannot form part of a bigger group and it is possible to play these by using them as kicker cards. In general most plays should reduce the number of moves needed to get rid of all cards. Bombs and Nukes are powerful since they can be played at any point, but they double the stake of the game. For this reason playing them early in the game is risky.

Starting a new category is a good position to be in, allowing a player to choose a category where he holds multiple groups, or holds a high-ranking group that opponents are unlikely to be able to play on. Hence inference is an important aspect of Dou Di Zhu especially concerning the location of the high ranked cards in opponents' hands. The two non-landlord players also need to work together since they either both win or both lose. This can be achieved by making plays that allow the other non-Landlord player to play cards or prevent the Landlord from making further plays.

4 METHODOLOGY

In order to focus only on the card play phase, it is possible to remove the bidding phase and designate that an arbitrarily chosen player gets the extra cards and is the Landlord.

4.1 Perfect Information Dou Di Zhu

It is possible to modify Dou Di Zhu to be a game of perfect information by playing with all cards face up at all times. This removes the need to make inferences about other player's cards and allows the exact consequences of every action to be studied when searching the game tree. This version of the game we call *Perfect Information Dou Di Zhu.*

In order to compare the results of playing multiple games of Dou Di Zhu, we count numbers of wins instead of cumulative rewards, ignoring the effects of player bidding and bombs/nukes. We award a player 1 point for winning a dealt hand and 0 points for a loss. Since the UCB algorithm has been well studied with rewards in $\{0,1\}$ we can benefit from the work of others in setting the parameter k which trades off exploration and exploitation in the UCB formula (1).

4.2 Imperfect Information Dou Di Zhu

By hiding from each player the cards held in the other players' hands during a game of Perfect Information Dou Di Zhu we create the game of *Imperfect Information Dou Di Zhu.* Note that the number of cards held by each player and the history of cards played so far are still visible to all players, only the ranks of the held cards are hidden. Since there are unknown cards, the game tree has a huge branching factor for every possible combination of cards each player could hold. Instead of searching this tree we apply a determinization approach similar to the approach Ginsberg [4] uses for Bridge. This involves searching multiple determinizations of the game at each decision step. However, where Ginsberg uses minimax search, we use the UCT algorithm.

At each decision our agent samples multiple determinizations by randomly distributing the hidden cards between the other players. Each of these determinizations is a game of Perfect Information Dou Di Zhu, to which the UCT algorithm is then applied. The actions available in the first layer of the tree are the same for each determinization since the cards our agent holds are the same in each case. The visits to each action for each determinization are summed and our agent selects the action with the highest total number of visits.

5 EXPERIMENTS

In the experiments described in this section we use determinized UCT, where d denotes the number of determinizations (i.e. d independent search trees) and s denotes the number of UCT iterations per determinization.

5.1 Variation in Dou Di Zhu win rates

Although the strength of decisions made by each player has a significant effect on the outcome of a game of Dou Di Zhu, some random deals may favour one player over another, whereas others may be much more sensitive to the players' decisions. In an effort to reduce the variance of subsequent results and thus allow them to be compared more easily, we begin by choosing a set of 1000 Dou Di Zhu deals to be used for the remainder of the experiments. The practice of specifying deck ordering in advance is common in Bridge and Whist tournaments between human players, to minimise the effect of luck when comparing players. This set of deals is chosen such that when played by a set of identical AI agents (in this case determinized UCT), the number of wins for a particular player is as close as possible to the mean number of wins for 1000 random deals. In order to choose such a set, we must first determine what that mean is.

We generated 100 sets of 1000 random Dou Di Zhu deals, and for each deal we played a single game with three determinized UCT players, each with $d = 50$ determinizations and $s = 250$ UCT iterations per determinization. For each set, we recorded how many of the 1000 games were won by player 1. Figure 1 shows a histogram of these numbers of wins.

From these results we find that the number of wins appears to be normally distributed. The mean is $\mu = 433.47$; thus we choose a set of deals for which player 1 achieved exactly 433 wins in this experiment as our "representative" set for the remainder of this paper. The standard deviation is $\sigma = 16.27$ and so a 95% confidence interval for the mean number of wins for player 1 is [433.37, 433.57].

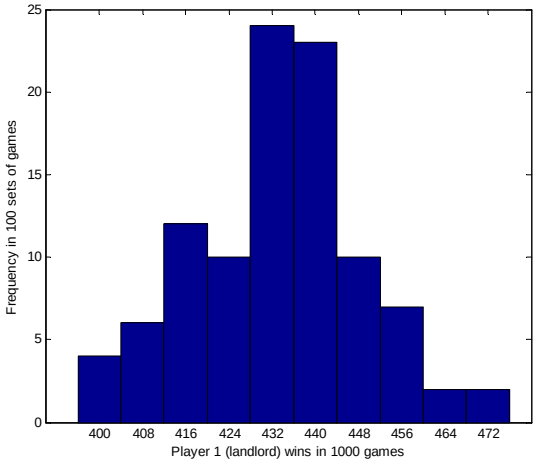


Figure 1. Histogram of win rates for the landlord player in 100 sets of 1000 Dou Di Zhu games.

5.2 Effect of d and s on playing strength

In these experiments, we played a number of games for each of our 1000 deals. In each game, players 2 and 3 used determinized UCT with 40 determinizations and 250 UCT iterations per determinization, whereas player 1 used determinized UCT with d determinizations and s iterations per determinization, each game with a different value for d and/or s. For each combination of d and s, we counted how many games out of the 1000 trials were won by player 1.

5.2.1 Varying d while s remains fixed

In this first experiment, we fixed four values for s, namely 50, 100, 250 and 500. For each of these, we used a number of values for d, ranging from 1 to 100.

Figure 2 plots the number of wins against d. We see that playing strength increases rapidly with $d < 20$, with rapidly diminishing returns for $d \geq 20$. However, there seems to be slightly more benefit to increasing the number of determinizations beyond 30 when the number of UCT iterations is low.

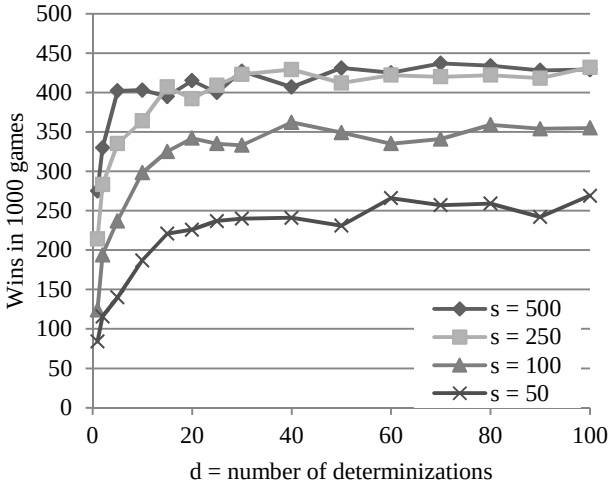


Figure 2. Plot of number of landlord (player 1) wins against number of determinizations, for fixed numbers of UCT iterations per determinization.

5.2.2 Varying s while d remains fixed

The conditions for this experiment were similar to those for the previous experiment, with the exception that we fixed four values of d (namely 5, 10, 25 and 40) and varied s (from 25 to 1000). The results are plotted in Figure 3. For $s \leq 300$ the playing strength increases logarithmically with the number of simulations, levelling off for $s > 300$.

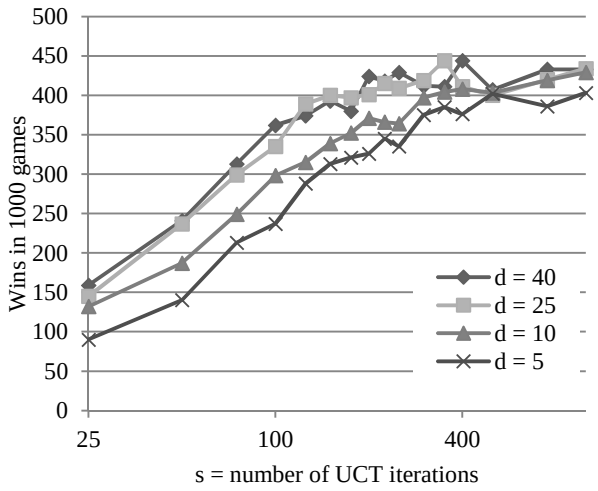


Figure 3. Plot of number of landlord (player 1) wins against number of UCT iterations per determinization, for fixed numbers of determinizations.

5.2.3 Varying d and s while ds remains fixed

Arguably the fairest comparison of the relative strength of different AI agents is to allocate them the same length of time (or the same number of CPU cycles) to make their decisions. However, this is not ideal from a scientific point of view since such measurements are inherently noisy and depend on issues of implementation and hardware. Instead, we simulate this by allocating a fixed number of UCT iterations per decision, making the (not unreasonable) assumptions that the time for a single iteration is roughly constant and the overall decision time is roughly linear in the number of iterations.

In this experiment we fixed four values of a constant Σ (namely 2500, 5000, 10000 and 15000), varied d from 1 to 100, and took $s = \left\lfloor \frac{\Sigma}{d} \right\rfloor$ for each value of d (so that $ds \approx \Sigma$). With our current implementation and hardware $\Sigma = 10000$ equates to approximately 1 CPU second of computation per move. Note that in this experiment (as in the preceding two experiments), players 2 and 3 have $ds = 40 \times 250 = 10000$.

Figure 4 plots the number of wins for player 1 against d. Given the results of the preceding experiments, it is not surprising that determinized UCT is weaker when d or s is too small, nor that its strength is somewhat independent of these parameters when both are sufficiently large.

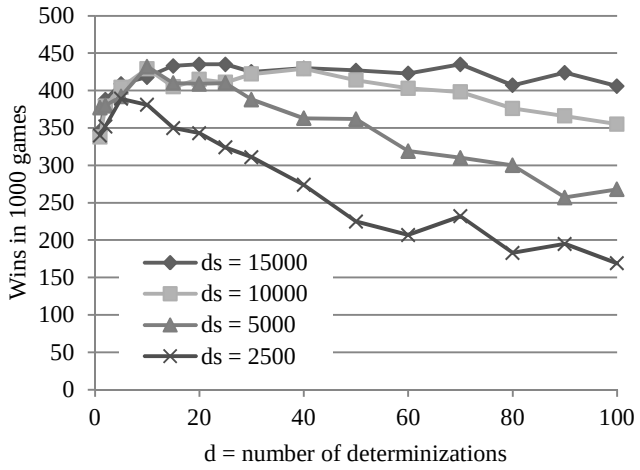


Figure 4. Plot of number of landlord (player 1) wins against number of determinizations, for a fixed number of UCT simulations divided equally among all determinizations.

5.3 Benefit of observing hidden information

As noted in Section 2.3.3, it is often useful in games of imperfect information to make inferences about the hidden information. However, it is not always clear how useful this inference may be. Richards and Amir [26] demonstrate the usefulness of inference in Scrabble by playing an AI agent with no inference engine against an AI agent with the ability to "cheat" and observe directly its opponent's hidden rack of letters. The benefit of direct access to the hidden information is an upper bound on the possible benefits of inference.

In our experiment for Dou Di Zhu, we played two types of AI agent against each other in various combinations: the determinized UCT player used in previous experiments (with $d = 40$ and $s = 250$) and a perfect information player in which the 40 determinizations are replaced by 40 copies of the actual state of the game. (This player still uses multiple UCT trees, but all trees have the same root state, which is the actual state of the game). Specifically, we take as a baseline the number of wins when each player uses determinized UCT (i.e. has only imperfect information) and measure the increase in numbers of wins when various players instead use perfect information.

Table 2 and Figure 5 show the results of this experiment. We see that knowledge of the other players' hidden cards increases the number of wins by around 7-12% (the benefit appears to be slightly less for the landlord, and slightly greater for the player preceding the landlord). Furthermore, if both non-landlord players can observe the hidden information, their win rate increases by around 21%.

When all three players have perfect information, players 2 and 3 win 12.7% more games than if all three have imperfect information. This suggests that perfect information is more beneficial to the non-landlord players than to the landlord.

Table 2. The increase in numbers of wins over 1000 games when the specified player(s) gain perfect information, and all other player(s) have imperfect information. For example, if player 2 has perfect information and players 1 and 3 have imperfect information, player 2 wins 86 more games than if all three players have imperfect information. Note that a win for player 2 is also counted as a win for player 3, and vice versa.

Player(s) with perfect information	Increase in number of wins (1000 games)
Player 1	69
Player 2	86
Player 3	119
Players 2 and 3	210

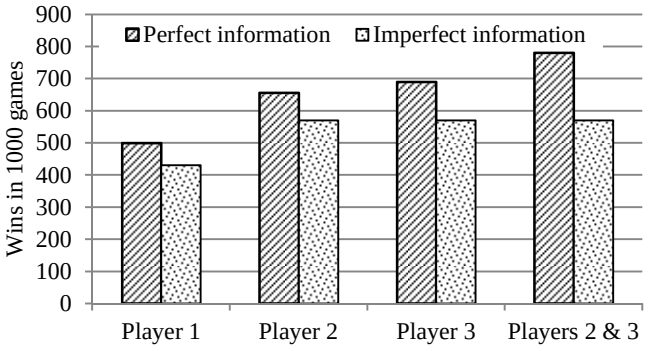


Figure 5. Number of wins for players with perfect versus imperfect information. Each pair of bars shows the numbers of wins for the specified player(s) when they have perfect or imperfect information and all other players have imperfect information.

6 CONCLUSION

In this paper we presented the hugely popular Chinese card game Dou Di Zhu as an interesting game for investigation. We applied a determinization approach using the UCT algorithm to search each determinization. Although we have yet to accurately determine the playing strength of the resulting AI agent (not least due to the lack of other strong AI for Dou Di Zhu against which to test) our informal experiments (playing against this paper's authors) suggest that it is on a par with human players. The nature of Dou Di Zhu suggests that the effects of some of the problems with determinization outlined in Section 2.3.1, are unlikely to be major: in particular, strategies such as information hiding or bluffing do not generally play an important role.

We showed that the performance of determinized UCT is not particularly sensitive to the trade-off between the number of determinizations and the number of UCT iterations per determinization. When the number of determinizations is fixed, performance scales approximately logarithmically with respect to the number of UCT iterations; when the number of simulations is fixed, performance increases rapidly with the number of determinizations until some threshold (approximately 20 in our experiments) is reached, after which increasing the number of determinizations does not significantly increase the playing strength. We do not see any obvious reason why these results should be specific to Dou Di Zhu; however we plan to repeat these experiments for other stochastic games of imperfect information to investigate this further.

We have shown that the increase in playing strength from an accurate inference engine is potentially large. One of our future

aims is to find an inference method for Dou Di Zhu that achieves some of this increase. Conceptually, it is easy to combine an inference model with determinization: instead of sampling determinizations at random, use determinizations that the inference model identifies as being close to the actual state. Our next step is to design an inference model that can identify such determinizations.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their helpful comments. This work is funded by grant EP/H049061/1 of the UK Engineering and Physical Sciences Research Council (EPSRC).

REFERENCES

[1] C.E. Shannon, "Programming a computer for playing chess," *Philosophical Magazine (Series 7)*, vol. 41, 1950, p. 256–275.

[2] A.L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, vol. 3, Jul. 1959, pp. 210-229.

[3] C.S. Lee, M.H. Wang, G. Chaslot, J.B. Hoock, A. Rimmel, O. Teytaud, S.R. Tsai, S.C. Hsu, and T.P. Hong, "The computational intelligence of MoGo revealed in Taiwan's computer Go tournaments," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, 2009, p. 73–89.

[4] M.L. Ginsberg, "GIB: Imperfect information in a computationally challenging game," *Journal of Artificial Intelligence Research*, vol. 14, 2002, p. 313–368.

[5] R. Bjarnason, A. Fern, and P. Tadepalli, "Lower bounding Klondike solitaire with Monte-Carlo planning," *Proc. ICAPS-2009*, 2009, p. 26–33.

[6] S. Yoon, A. Fern, and R. Givan, "FF-Replan: A Baseline for Probabilistic Planning," *17th International Conference on Automated Planning and Scheduling (ICAPS-07)*, 2007, p. 352–359.

[7] R.B. Myerson, *Game Theory: Analysis of Conflict*. Harvard University Press, 1991.

[8] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," *Machine Learning: ECML 2006*, 2006, p. 282–293.

[9] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, 2002, pp. 235-256.

[10] S. Gelly and D. Silver, "Combining online and offline knowledge in UCT," *Proceedings of the 24th international conference on Machine learning*, ACM, 2007, p. 273–280.

[11] G. Chaslot, M. Winands, and H. van Den Herik, "Parallel Monte-Carlo tree search," *Computers and Games*, 2008, p. 60–71.

[12] H. Finnsson, "CADIA PLAYER : A Simulation-Based General Game Player," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, 2009, pp. 4-15.

[13] G. Chaslot, C. Fiter, J.B. Hoock, A. Rimmel, and O. Teytaud, "Adding expert knowledge and exploration in Monte-Carlo Tree Search," *Advances in Computer Games*, 2010, p. 1–13.

[14] S. Yoon, A. Fern, R. Givan, and S. Kambhampati, "Probabilistic Planning via Determinization in Hindsight," *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008, pp. 1010-1017.

[15] E.K.P. Chong, R.L. Givan, and H. Chang, "A framework for simulation-based network control via hindsight optimization," *Proceedings of the 39th IEEE Conference on Decision and Control*, 2000, pp. 1433-1438.

[16] J. Borsboom, J.-takeshi Saito, G. Chaslot, and J. Uiterwijk, "A comparison of Monte-Carlo methods for Phantom Go," *Proc. 19th Belgian–Dutch Conference on Artificial Intelligence–BNAIC*, Utrecht, The Netherlands, 2007.

[17] P. Ciancarini and G.P. Favini, "Monte Carlo tree search in Kriegspiel," *Artificial Intelligence*, vol. 174, Jul. 2010, pp. 670-684.

[18] S.J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2009.

[19] I. Frank and D. Basin, "Search in games with incomplete information: a case study using Bridge card play," *Artificial Intelligence*, 1998, pp. 87-123.

[20] J. Long, N. Sturtevant, and M. Buro, "Understanding the Success of Perfect Information Monte Carlo Sampling in Game Tree Search," *AAAI Conference on Artificial Intelligence*, 2009.

[21] D. Koller and A. Pfeffer, "Representations and solutions for game-theoretic problems," *Artificial Intelligence*, vol. 94, 1997, p. 167–215.

[22] M. Zinkevich, M. Johanson, and M. Bowling, "Regret minimization in games with incomplete information," *21st Annual Conference on Neural Information Processing Systems (NIPS 2007)*, 2007.

[23] M. Ponsen, J. Ramon, T. Croonenborghs, K. Driessens, and K. Tuyls, "Bayes-Relational Learning of Opponent Models from Incomplete Information in No-Limit Poker," *Proceedings of the Twenty-third National Conference on Artificial Intelligence (AAAI-08)*, 2008, pp. 1485-1487.

[24] M. Ponsen, G. Gerritsen, and G. Chaslot, "Integrating Opponent Models with Monte-Carlo Tree Search in Poker," *Proceedings of Interactive Decision Theory and Game Theory Workshop at the Twenty-Fourth Conference on Artificial Intelligence (AAAI-10)*, 2010, pp. 37-42.

[25] R.J.S. Baker and P.I. Cowling, "Bayesian opponent modeling in a simple poker environment," *IEEE Symposium on Computational Intelligence and Games*, 2007, pp. 125-131.

[26] M. Richards and E. Amir, "Opponent modeling in Scrabble," *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007, p. 1482–1487.

[27] M. Buro, J.R. Long, T. Furtak, and N. Sturtevant, "Improving state evaluation, inference, and search in trick-based card games," *Proc. 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, Morgan Kaufman, 2009, pp. 1407-113.

[28] J. McLeod, "Dou Dizhu," 2010. http://www.pagat.com/climbing/doudizhu.html. Accessed 4th February 2011.

From game tutorials to game partners using natural language generation techniques

Luciana Benotti
PLN Group, FAMAF
National University of Córdoba
Córdoba, Argentina
luciana.benotti@gmail.com

Nicolás Bertoa
PLN Group, FAMAF
National University of Córdoba
Córdoba, Argentina
nicobertoa@gmail.com

Abstract

The goal of this work is to design and implement an agent which generates hints for a player in a first person shooter game. The agent is a computer-controlled character which collaborates with the player to achieve the goal of the game. Such agent uses state of the art reasoning techniques from the area of artificial intelligence planning in order to come up with the content of the instructions. Moreover, it applies techniques from the area of natural language generation to generate the hints. As a result the instructions are both causally appropriate at the point in which they are uttered and relevant to the goal of the game.

1 Introduction

Nowadays, most game tutorials make the player follow a fixed script. As a result having a good or bad tutorial is still an art that depends on how good the script is at faking some freedom (for instance, by foreseeing possible reactions of the player) in order to make it more entertaining. *Natural language generation* can offer game designers with techniques that, in the future, may transform good tutorial design not in an art but in a science by providing insights and efficient algorithms for calculating the players knowledge and dealing with the players reactions in a dynamic way.

The goal of this work is to design and implement an agent (a computer-controlled character) which is able to generate hints that help a player advance in a game situated in a 3D virtual world. The agent knows information that is necessary to win the level and is not known by the player, hence the player needs to collaborate with the agent in order to finish the level successfully. Such agent uses state of the art reasoning algorithms such as *planning* in order to come up with the content of the instructions which are relevant at each point in the interaction. Furthermore, it applies techniques from situated natural language generation such as *common ground management* to generate the context-aware hints.

We believe that our contribution is interesting for the game industry as well as for the players. For players, this work is a first step towards more flexible, effective and entertaining tutorials. In order to sustain our claims we performed a human based evaluation. The results of the evaluation, though preliminary due to the amount of subjects, are indeed encouraging. For the game industry, good tutorials are expensive to develop because it is costly to script all the appropriate game behaviors caused by different potential player reactions (some of which might never be triggered). Our approach can be seen as a way of automatically producing scripts that change according to the unpredictable player behavior and the changing environment, reducing the burden on the developer.

The paper proceeds as follows. In Section 2 we briefly introduce the area of natural language generation. Section 3 describes the game environment where our game partner lives, and discusses the process of planning problem generation from the game environment. Section 4 presents the process used to generate an instruction according to a plan. Section 5 describes the integration of our generated instructions into the game interaction as well as the management of common ground between the player and the agent. Section 6 presents the results of our human evaluation. Section 7 concludes the paper.

Figure 1: The player's view in the GIVE Challenge.

2 Related work

Natural language generation (NLG) is a subfield of Artificial Intelligence (AI) and Computational Linguistics. It is concerned with the construction of computer systems which can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information. NLG systems combine knowledge about language and the application domain to automatically produce documents, reports, explanations, help messages, and other kinds of texts.

The area of NLG is a new area of research, with less than a couple of decades of experience (Reiter and Dale, 2000). However, it is a rapid developing field that has put forward promising techniques in the last few years, in particular thanks to the shared challenges proposed for comparing current NLG technologies. One of such challenges is called GIVE (Giving Instructions in Virtual Environments). GIVE (Koller et al., 2010) is particularly relevant for our work since it applies NLG techniques to the problem of generating natural language instructions in a 3D virtual world. As a result several techniques investigated by GIVE's research community are directly applicable to this work. In the GIVE task, human players try to solve a treasure hunt in a virtual 3D world that they have not seen before. The computer has a complete symbolic representation of the virtual world. The challenge for the NLG system is to generate, in real time, natural-language instructions that can guide the player to the successful completion of their task. Only the player can effect any changes in the world, by moving around, manipulating objects, etc. Figure 1 shows a screen-shot of the user's view on the 3D world. On the top of the picture, the current instruction given by the NLG system is displayed.

The NLG tasks and techniques that are relevant for GIVE and for this work are content determination through planning, situated generation of referring expressions and common ground management; we will briefly discuss each of them.

The task of content determination consists in deciding which information to communicate to the player such that the information is appropriate at the current point in the interaction. Such task can be implemented using the inference technique of planning (Benotti, 2010). As a result, the instructions are both relevant to the goal of the game and causally appropriate at the point in which they are uttered.

The situated generation of referring expressions area (Reiter and Dale, 2000) provides algorithms for coming up with descriptions of objects so that the player can identify such objects

(e.g. "the door in front of you") taking into account both static (e.g., color) and dynamic properties (e.g., visibility) of an object.

Finally, common ground management is the task of generating grounding acts when appropriate according to the behavior of the player. Grounding acts are utterances that, in a strict sense, do not add new information to the discourse but instead they reinforce old information. For example, if the NLG system tells the player "take the left green kit" and the player turns slightly left to face an object, the system may generate a positive grounding act such as "yes" if that was the right object, or "no" otherwise (Traum, 1994).

These three techniques are useful for generating hints that convey appropriate information to the player and that consider the player reaction in order to reinforce the information. In (Garoufi and Koller, 2010) the authors propose to cast all three tasks as a planning problem. Such integration is proven to be successful for small and discrete game worlds, however it does not scale to continuous game worlds in which the player can move freely (and continuously) in 3 dimensions. As a result in this paper we propose to use planning for handling only content determination while we use more traditional NLG methods for the other two tasks.

3 Finding plans from a game state

The design of game characters decision making is one of the most challenging tasks when designing a game; the expert game partner is, by no means, an exception to this rule. In most games, the characters decision making is either implemented in an ad-hoc way which is scripted for that particular game, or designed as a state machine which needs to be kept small because of scalability issues (Millington and Funge, 2009). A believable game partner can not get away with a decision making process that is either scripted or small. First, the agent needs to react appropriately to infinitely many player reactions which cannot be predicted and scripted. And second, it needs to reason over complex game states in a goal directed way in order to be able to give instructions that are both causally appropriate at the point in which they are uttered and relevant to the goal of the game.

Our approach to implement the decision making process of the expert game partner (a.k.a content determination in the NLG area) is to use an off-the-shelf automated planner. The planner that we use is FastForward (FF). FF (Hoffmann, 2005) can handle big planning problems (with over a million objects) and return plans of thousands of actions in seconds and can handle our game environment (with a couple of hundreds of objects and plans with a maximum length of 250 actions) in a few milliseconds. Planning problems are specified in PDDL (Gerevini and Long, 2005).

Automated planners are ready to be used in industry applications. They have reached a maturity level in which they can be used in real time applications even if the need of re-planning is high. In our setup there is a high need for re-planning because the behavior of the player is non-deterministic and unpredictable. Moreover, planners are domain independent, in order to come up with a plan which wins the game they just need an specification of the actions that are possible in the game and a discretized representation of what the agent knows about the state of the game. The agent knowledge about the region locations and adjacencies is discretized using the game waypoints (Millington and Funge, 2009).

The game environment for which we have implemented our game partner lives is first person shooter game (FPS). The game scenario, called Igor[1], was developed using the Irrwizard framework[2] and extended in C++. As in most FPS games the player is situated in a 3D world where he can perform several actions such as

[1] https://sites.google.com/site/nicolasbertoa/igor

[2] http://irrwizard.sourceforge.net/

walk, jump, climb stairs, shoot and pick up different items which have different effects. The goal of the game is to kill a creature that is wandering in the 3D world. The creature cannot be killed only by shooting at it because it has a self healing mechanism that needs to be turned off first by deactivating a series of power rays in a given order. The NLG agents knows which is the right sequence of rays as well as the position of each of these rays. It is also able to recognize other items (such as poison or health) and is aware of their effect. All this information is stored in the specification that is sent to the planner and the planner returns a sequence of actions that, if executed in the current state of the game, would achieve the goal of the game.

The agent must extract information from the environment and represent it in the planner language: PDDL (Gerevini and Long, 2005). PDDL is intended to express the physics of a domain, that is, what facts are true in the world, what actions are possible, and what the preconditions and effects of actions are. The agent represents the number of rays that the player needs to pick up and its order. Also, the agent codifies in the planning problem the waypoints and adjacencies which are used by the planner to find the nearest path between the player and the items to pick up. Moreover, the agent needs information about the player and the enemy position and state. The agent constantly verify the health level of the player and enemy and updates its condition (attacking, wandering, dead etc) in the planning problem.

Once the agent has joined all the information about the problem, it generates a planning problem and sends to the planner. Then the planner generates a plan and the agent uses the plan to generate the instructions (see next section). The planner takes, in average, 8.61 milliseconds to find a plan for our game planning domain (with a standard deviation of 7.46 milliseconds). The maximum length of a plan in our game environment is 250 actions (notice that plans and plan length varies depending of the current state of the interaction). Once a plan was obtained, the agent needs information about the environment to check if the plan was accomplished or not, and to know the situations in which must replan (we address these issues in Section 5).

4 Using plans to decide what to say

This section explains how the agent uses the plan obtained from the planner. The plan is composed by a series of steps that the player must follow to meet the plan goal. When the agent has a new plan, it must decide what instruction to say next. Suppose that the planner give us the plan steps that shown in Figure 2. This plan contains two types of actions, *MoveTo* and *TakeKey*. The first type of action indicates that the player has to move from one way-point to another, and the second indicates the ray that the player hast to pick as well as the way-point where it is located.

```
MoveTo  w4    w5
MoveTo  w5    w6
MoveTo  w6    w7
MoveTo  w7    w8
MoveTo  w8    w9
MoveTo  w9    w10
MoveTo  w10   w11
TakeKey blueKey w11
```

Figure 2: Plan steps generated by the planner.

These steps a plan to pick up the ray *blueKey* as illustrated in Figure 3. Now, the agent has the problem of deciding which plans steps to verbalize, that is it needs to do content determination based on the plan steps. Notice that the naive approach of verbalizing all plan steps would result in a very repetitive and over restrictive game partner. Therefore we made the content determination process of our agent dependent of the actions whose waypoints are directly visible to the player or visible by turning around on his current position, we will say that this waypoints are *visible 360*.

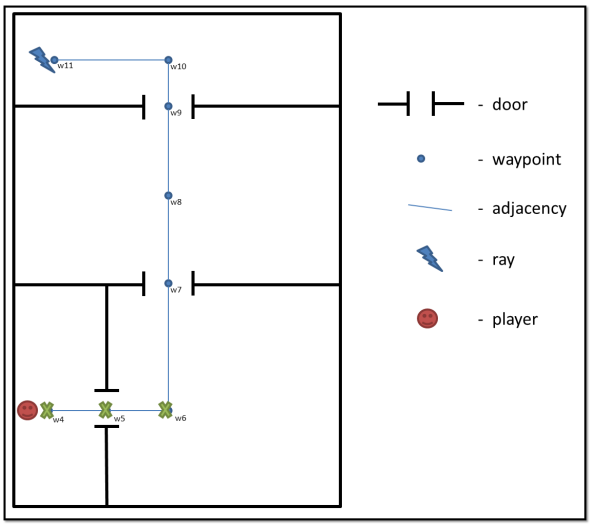


Figure 3: Path of the plan in Figure 2 with waypoints visible 360 highlighted (one uniquely referable).

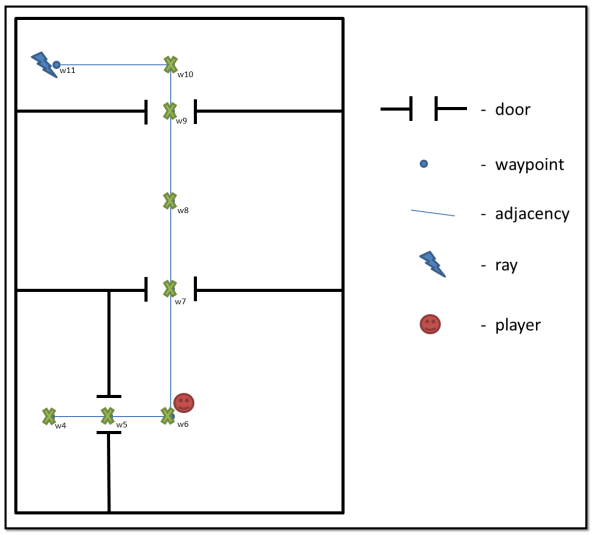


Figure 4: Path of the plan in Figure 2 with waypoints visible 360 highlighted (many uniquely referable).

The Figure 3 shows that there are only three visible waypoints of the plan at 360 degrees from the position of the player. These are the waypoints 4, 5 and 6. The others are blocked by the environment. From the visible waypoints we verbalize the one that has a referring expression which uniquely identifies it. In our case, the waypoints 4 and 6 are waypoints that are in the middle of rooms, so, there are no referring expressions to unambiguously describe them. But in the case of the waypoint 5, there is a door, so we can use a referring expression to uniquely identify the object involved. As a result, in the situation illustrated in Figure 2 the instruction "go through the door in front of you" is generated.

But, what happens in the case when there are more than one object that can be uniquely identified? The Figure 4 shows this case. There are seven visible waypoints of the plan at 360 degrees from the position of the player, and three of them have an object that can be uniquely identified: 5 (has a small door), 7 (has a big door), and 9 (has some stairs). In this case, the agent generates an instruction which refers to the the furthest visible object in the path of the plan and verbalizes "see those stairs in front of you?" before saying "take them" (these strategy is used when referring to objects that are far away from the player, see Section 5 for details).

The screen-shots in Figure 5 illustrate the process we just explained. In the screen-shot we have drawn the visible waypoints and the paths between them.The arcs between waypoints illustrate the actions in the plan. In the screen-shot in the top of Figure 5 the planned actions could be verbalized as "go forward, go through the door, go forward, go forward, do you see those stairs in front of you?, take them, go forward". As we said, we have decided to verbalize the last action in the sequence which contains a referring expression which uniquely identifies the object involved in the action. In this example, this is the case for the action "do you see those stairs in front of you?" since "those stairs" are a distinguishing referring expression (given the current position of the player) for the stairs that are further away. We say that the first actions, namely "go forward, go through the door, go forward, go forward" are *left tacit* (Benotti, 2007) and they are expected to be inferred by the player given

Figure 5: Instructions generated using the plan and the player's visibility

the causal constraints of the world (in simpler terms, in order to see "those stairs" better the player will need to approach them). The screenshot at the bottom of Figure 5 ("Turn left"), illustrates an instruction whose goal is to make directly visible a waypoint that is visible 360.

5 Interaction and common ground

Once the agent generates the instruction, it will show that instruction to the player as illustrated in Figure 5. Of course, the player may decide to follow the instruction or to do something else. A good game partner should handle both situations robustly; we discuss our strategy for doing this in Section 5.1. Furthermore, not only the player but also our environment can behave in a non deterministic way. The game partner should be able to sense and react to a changing environment in an appropriate way; Section 5.2 addresses this issue. In both of this situations grounding acts can be use to reinforce instructions that were already communicated but were not successfully accomplished yet; we illustrate how positive and negative grounding acts are used by our agent in Section 5.3.

5.1 Dealing with unpredictable behavior

Our strategy for dealing with the player's unpredictable behavior is to find a new plan (that is, to replan) every time the player gets *too far away* from the current plan. The crucial point here is to define what it means to get too far from the current plan. Since we base our content determination procedure in the player visibility we also use the same strategy to decide when to replan. The agent will replan when all waypoint in the plan are no longer visible 360 for the player.

Let's illustrate our strategy by an example. Figure 6a shows an example scenario and the waypoints of the plan which are visible 360.

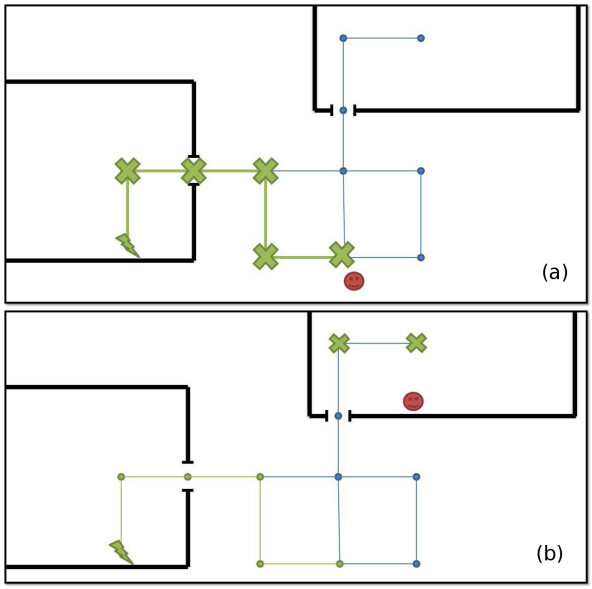


Figure 6: Re-planning example.

Now suppose that the player does not follow the instructions and ends up in the situation illustrated in Figure 6b. In this new situation re-

planning is needed because none of the waypoints in the plan is visible anymore from the current player position.

Summing up, the agent will re-plan when there are not visible waypoints of the plan, because is very difficult that the agent achieves the reorientation of the player towards the goal. This strategy results effective (and not overly restrictive) while guiding the player to the game goal as shown by our evaluation results in Section 6.

5.2 Dealing with a changing environment

Suppose we have 4 ray items: blue, green, violet and red and the correct sequence to deactivate the enemy's defense mechanism is red, blue and red. The Figure 7 shows the plan that the agent obtained from the planner (the plan is simplified not to show move actions for presentation simplicity). What the planner cannot handle is the fact that, when the player picks up some object of the game, the object repositions randomly in another way-point. For example, in the Figure 8 we can see that the obtained plan is incorrect because the red ray was repositioned at a different way-point. The planner assumes a deterministic environment, we deal with a non deterministic environment, such as our game, by means of replanning. That is, when the position the environment changes in a non deterministic way (for example when a ray changes its position), the agent replans.

5.3 The importance of the grounding acts

Grounding acts are utterances that, in a strict sense, do not add new information to the discourse but instead they reinforce old information. Grounding acts are not required from an informational point of view of communication, however, they play an important role in order to cope with changing environments, and the unpredictable behavior of the player.

The Figure 9 shows a case of the positive grounding act "Get this ray" which follows the instruction "Turn left to see the ray" when the

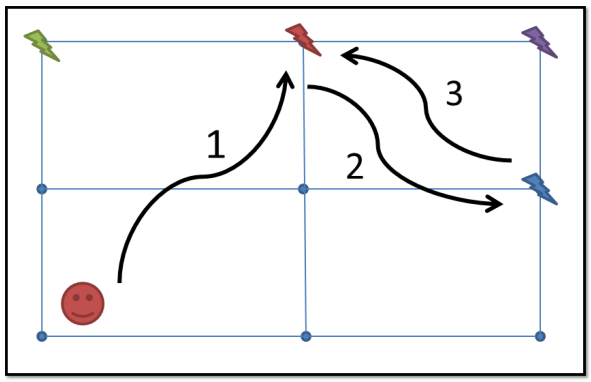


Figure 7: Steps to get the sequence of rays.

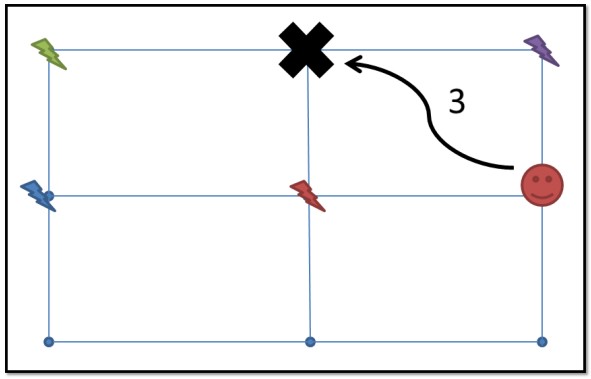


Figure 8: The plan was incorrect due dynamic properties of rays.

ray becomes visible. This utterance is a positive grounding act because, it does not add new information to the discourse. The player should be able to figure out that he should pick up this ray, otherwise, why would have the agent guided him to it? However, natural language is ambiguous and the player may not draw this conclusion so the reinforcing achieved by the grounding act is indeed useful.

The other screen-shot in the Figure 9 illustrates a case of negative grounding acts with "This is not the ray you need", which follows the instruction "we need to find the green ray", when the player hovers the mouse pointer on the red ray. With this instruction, the agent prevents the player from re-activating the protection mechanism of the creature as a result of

Figure 9: Grounding acts that can be generated as a reaction to the player actions

Figure 10: Multi-utterance instructions which create and use the common ground

picking a ray in the wrong sequence. Again, in a strict sense, the negative grounding act is not necessary because the player has already been told that the next ray that is needed is green. However, the player may not remember this and may try to take the red ray which is directly in front of him.

The screen-shots in Figure 10 illustrates a typical example of common ground creation between the agent and the player. The screen-shot in the top shows the instruction "Do you see that ray in front of you?". With this instruction the agent wants that the player to focus his attention in a ray with is in front of him. As a result it is to be expected that the player gets closer to the ray. In this new situation, the agent generates the instruction "Pick it up". It is clear from this instruction that the agent wants that the player picks up the blue ray in front of him, but we can see that neither the ray nor its position are included in the instruction. This can be done because this information is already in the the common ground between the agent and the player.

6 User evaluation

In this section we describe the results of the human evaluation of our agent. You can see a gameplay video at https://sites.google.com/site/nicolasbertoa/ai4games_video.

For our evaluation we used objective and subjective metrics. Objective metrics were collected by logging the player behavior and the subjective metrics by asking players to complete a questionnaire after finishing the tutorial level. We used the same metrics that are used

in the GIVE Challenge (Koller et al., 2010) to evaluate systems in terms of the effectiveness, naturalness of instructions and the engagement of the interaction. We gathered 10 volunteers for the evaluation. Each of them played the game in his own. The demographic characteristics of the volunteers that we collected are the following: they were all male, the average age was 20 years old and all of them were gamers. This subject population is the target market of the kind of game we implemented.

The results that we obtained using the objective metrics are shown in Figures 11 and 12.

The Figure 11 shows the number of successful instructions, minor faults and serious faults. An instruction is considered successful if the player did exactly what the system asked him to do, an instruction is considered a minor fault if the player deviated from the instruction without causing a replanning, and an instruction is a serious fault if the agent had to replan after uttering it. Their averages are 44.1, 28.9 and 4.4, and their standard deviations are 9.74, 17.04 and 3.92, respectively. We can see that there were very few serious faults, this suggests that the agent was successful in the task of guiding the player. Also, we can see that players deviated from 40% of the instructions and, according to the game logs, this is due mainly to the presence of enemy which obviously causes the player to not follow the plan. Finally, approximately 60% of the instructions were directly successful, the player did exactly what the agent asked him to do in more than half of the cases.

The Figure 12 shows the average of the other objective metrics we collected. The successful instructions were completed quickly, which suggests that the instructions were easy to understand and execute. Also, players were able to complete the level quickly without visiting all the map waypoints (they only visited, in average, 70% of the map).

The Figure 13 shows the results of the subjective metrics. On the one hand, the players considered that the instructions were too repetitive.

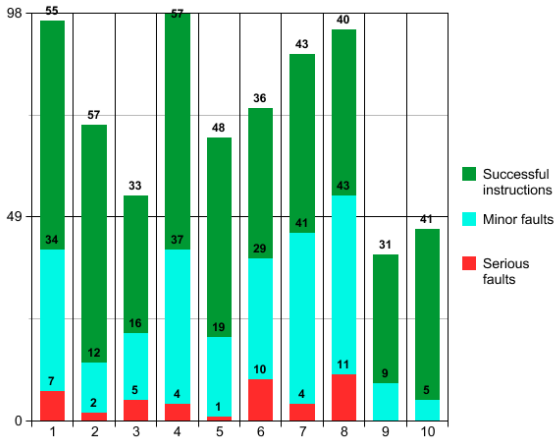


Figure 11: Objective metrics: Successful instructions, minor and serious faults per player.

	Average	Standard deviation
Time of completeness per instruction (seconds)	28.9	17.04
Waypoints traveled	65.5	18.76
Playing time (minutes)	2.4	1.29

Figure 12: Objective metrics: average times and distance traveled.

Also, they considered that the instructions were few time at the screen and that the instructions were generated too early. Also, some players said that sometimes they were confused about the direction to go in. On the other hand, the statements that are related to entertainment obtained very high percentages, which suggests that the player had fun, that is the main objective of any game. Another strong point is that the instructions were very clearly worded and were understood. Also, players perceived that most of the time the agent helped to the players when they were confused.

We consider that these results are encouraging, in particular because one of the main goals of our agent was to be able to participate in an engaging interaction while still providing useful instructions whenever the player needed help. We are considering different techniques to improve the agent in those characteristics in which it did not get such good results (for example, by doing corpus based generation in order to avoid being repetitive (Gandhe and Traum, 2007)).

Number	Statement	%
1	The system used words and phrases that were easy to understand	80
2	I had to re-read instructions to understand what I needed to do	70
3	The system gave me useful feedback about my progress	50
4	I was confused about what to do next	75
5	I was confused about which direction to go in	80
6	I had no difficulty with identifying the objects the system described for me	80
7	The system gave me a lot of unnecessary information	10
8	The system gave me too much information all at once	75
9	The system immediately offered help when I was in trouble	80
10	The system sent instructions too late	50
11	The system's instructions were delivered too early	10
12	The system's instruction were visible long enough for me to read them	35
13	The system's instructions were clearly worded	90
14	The system's instruction were repetitive	90
15	I really wanted to kill that creature	95
16	I lost track of time while solving the overall task	80
17	I enjoyed solving the overall task	90
18	Interacting with the system was really annoying	55
19	I would recommend this game to a friend	70
20	The system was very friendly	70
21	I felt I could trust the system's instructions	60

Figure 13: Results of the subjective metrics.

7 Conclusions

In this paper we have presented an agent which is able to generate hints that help a player win a level in first person shooter game. Such agent uses state of the art reasoning such as planning in order to come up with the content of the instructions and state of the art techniques from natural language generation to context-aware generate referring expressions and grounding acts.

Currently, most game tutorials make the player follow a fixed script. It must contemplate as many situations as possible. To achieve that, the developers often use state machines, which needs to be kept small because of scalability issues (Millington and Funge, 2009). Also, it is very difficult to predict all the player's reactions, therefore, when a reaction is contemplated, the developer must add new source code. Our approach is a way of automatically producing scripts that change according to the unpredictable player behavior and the changing environment, reducing the burden of the developer. This paper is a first step to motivate the interaction between the game industry and the research area of generation of natural language.

References

Luciana Benotti. 2007. Incomplete knowledge and tacit action in a dialogue game. In *Workshop on the Semantics and Pragmatics of Dialogue*, pages 17–24, Italy.

Luciana Benotti. 2010. *Implicature as an Interactive Process*. Ph.D. thesis, Université Henri Poincaré, INRIA Nancy Grand Est, France. Supervised by P. Blackburn. Reviewed by N. Asher and B. Geurts.

Sudeep Gandhe and David Traum. 2007. Creating spoken dialogue characters from corpora without annotations. In *Proceedings of Interspeech*, Belgium.

Konstantina Garoufi and Alexander Koller. 2010. Automated planning for situated natural language generation. In *Proceedings of the 48th ACL*, Uppsala.

Alfonso Gerevini and Derek Long. 2005. Plan constraints and preferences in PDDL3. Technical Report R.T. 2005-08-47, Università degli Studi di Brescia, Italy.

Jörg Hoffmann. 2005. Where 'ignoring delete lists' works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research*, 24:685–758.

Alexander Koller, Kristina Striegnitz, Andrew Gargett, Donna Byron, Justine Cassell, Robert Dale, Johanna Moore, and Jon Oberlander. 2010. Report on the second nlg challenge on generating instructions in virtual environments (give-2). In *Proceedings of the International Natural Language Generation Conference (INLG)*, Dublin.

Ian Millington and John Funge. 2009. *Artificial Intelligence for Games*. Morgan Kaufmann, Burlington, MA, USA.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press, New York, NY, USA.

David Traum. 1994. *A Computational Theory of Grounding in Natural Language Conversation*. Ph.D. thesis, Computer Science Dept., U. Rochester, USA. Supervised by James Allen.

Representing Personality Traits as Conditionals

Richard Evans[1]

Abstract. This paper compares two approaches to representing personality traits in synthetic agents. It proposes a set of goals that any computational implementation of personality should satisfy. It describes the personality trait system used in *The Sims 3*. Then an alternative system is described, in which traits are represented as conditionals relating world state to emotional state. It is shown that the conditionals model does a better job of satisfying the desiderata.

1 INTRODUCTION

The more highly evolved the species, the more one individual's behaviour differs from another's within that species [1, 2]. If we want to build evolved synthetic characters, they too must express individual personality.

Modelling individual personalities enables the following novel sort of player interaction: first the player makes some synthetic characters and chooses their personalities. (Perhaps, for example, he chooses an irascible old man, and his sweet forgiving daughter). Then he drops them into a particular social situation. (Perhaps, for example, he makes them homeless). Then *he just sits back and watches the emergent drama unfold*: the social situation they are in presents them with problems and their different personalities give them unique solutions to those personalities.

The Sims 3 has a model of individual personality that has allowed many players to experiment with exactly this sort of interaction [3]. One notable example is documented in the very popular Alice & Kev blog. Robin Burkinshaw [4] created two Sims with very different personalities: an irascible old man and his sweet, forgiving daughter. He dropped them into a particularly challenging social situation: he gave them no money, no job, and made them homeless. Then he sat back, recording the events that unfolded autonomously:

"I have attempted to tell my experiences with the minimum of embellishment. Everything I describe in here is something that happened in the game. What's more, a surprising amount of the interesting things in this story were generated by just letting go and watching the Sims' free will and personality traits take over".

In Burkinshaw's emergent unfolding story, Kev (the irritable old homeless man) is always trying to find a place to stay the night. But whenever he finds people who are kind enough to have him, he ends up arguing with them. Eventually, he irritates them so much that they ask him to leave, and he is back on the streets again. It is a hard life, and he takes it out on his daughter. But she is remarkably sweet and always forgives him.

Part of what makes Burkinshaw's blog so compelling is the heartfelt description of the homeless situation. But it is the *combination* of the social situation with the unique personalities that makes it dramatic. As Schopenhauer once wrote:

"The revelation of the idea of man is accomplished chiefly by two means: by accurate drawings of significant characters, and by the invention of poignant situations in which they reveal themselves." [5]

This paper outlines two different computational models of individual personality. It proposes a set of goals for any representation of personality. Then it sketches how *The Sims 3* models personality, and evaluate how well it meets the desiderata. Next an alternative model of personality traits is proposed, in which each trait is modelled as a declarative conditional, relating world state to emotional state. It is shown that this alternative model does a better job of meeting the desiderata.

2 DESIDERATA FOR A COMPUTATIONAL MODEL OF PERSONALITY

Before evaluating different models of personality, we need a list of explicit requirements and goals that we can use to help us adjudicate.

The first requirement on any computational model of personality is that a personality be composed of atomic units, which can be reused in a variety of different personalities. Satisfying this *compositionality requirement* is essential if we want to be able to generate a wide variety of personalities cheaply. As Chris Crawford puts it, we should "apply the ideas of vector analysis to the problem and look for a complete set of vectors that span the vector space of the problem" [15].

The second major requirement is that each personality trait has a distinct and obvious effect on autonomous behaviour. If you create a character that is a foul-mouthed extrovert, and leave him to his own devices, he should go out meeting new people and swearing at them. Further, the way he autonomously manifests his personality should be *transparent* to the player: the player shouldn't have to take careful notes of every action the character does for many days, compile them in a spreadsheet, in order to notice that this character is 11.3% ruder. Manifestation of personality has to be obvious in individual behaviour without recourse to statistical patterns.

The third major requirement on our model of personality is that it explains how personality connects with *emotion*. We all know that different people are differently affected by the same external stimuli. Our model of personality must handle this.

The fourth requirement is a practical authoring requirement when building a large multi-agent system. If we are

[1] RichardPrideauxEvans@gmail.com Thanks to Michael Mateas for feedback on a previous draft. Thanks also to Matt Brown, Ray Mazza, Peter Ingebretson, Tom Wang, Eric Holmberg-Weidler and Matt Goss for many illuminating discussions on these isssues.

going to build a world with a large number of personality-traits, we will have to minimize the amount of authoring needed when adding a new trait. We must minimize the amount of code and data that needs to be touched when adding a new trait. If there are *n* affordances and *m* personality traits, we need an authoring approach which requires considerably less content than *n* * *m*.

The remaining goals are nice to have, but not as important as the preceding.

The fifth goal is that our computational mechanism for generating personalities is sufficiently compositional that we can generate an *indefinite* number of personalities. A system that can generate a large, but finitely bounded, set of personalities is not as rich as a recursive generative system.

Personality traits seem to have varying degrees of resolution and specificity. For example: some people are aggressive (tout court), others become aggressive *if humiliated*. Our sixth goal is that our model of traits should allow us to specify traits at different levels of specificity.

Some traits are incompatible with others. For example: a character cannot both have a good sense of humour, and also be completely humourless. Now trait incompatibilities such as these could be hand-authored, or (preferably), the incompatibility between traits could be a derived fact, entailed by the description of the traits themselves. Our seventh goal is that incompatibility between traits should be *automatically derivable* in the model (rather than having to be painstakingly hand-coded).

One's past affects one's personality. Different people narrate early life-events differently, and their interpretation of those events determines their understanding of their current possibilities of action. Our final goal is that the system can express the way *personal narratives explain personality*.

To summarize, the computational model of personality should satisfy the following goals:

1. Each personality must be **decomposable** into atomic units which can be reused in other personalities
2. Personality trait must affect **autonomous** action. When we leave the characters alone, we want them to autonomously behave in-character
3. Each personality trait must affect **emotion**
4. Requirements for **authoring**: minimize the number of places we need to touch in order to add a new trait
5. The model should be able to make an **indefinite** number of personalities
6. It should provide for the idea that some traits are fine-**grained refinements** of others
7. The model should provide the means for automatically deriving which pairs of traits are **incompatible**
8. The model should make it possible for **personal narratives** to explain personality

3 HOW *THE SIMS 3* REPRESENTS PERSONALITY

The Sims 3 has a model of individual personality based on simple traits. Each Sim can have up to five traits from a pool of 80. This means there are 80-choose-5 (240 million) distinct personalities.

Personality traits affected autonomous behaviour in three main ways. (1) For each trait there was a unique motive associated with it. For example: a mean-spirited Sim was given an extra desire to undermine the self-worth of other people. This extra motive affected his autonomous behaviour, so he behaved in character. Actions that were specially suited to that personality trait were tagged as satisfying the corresponding motive. (2) When a Sim was interacting with objects, traits affected emotional state via a large number of ad-hoc if-thens scattered throughout the code. Naturally, these were rather difficult to find, and difficult to maintain. (3) When a Sim was interacting with other Sims, traits affected emotional state via a set of if-then rules expressed in a simple declarative language (horn-clauses with a small fixed set of free variables). For example:

- If my interlocutor makes a joke, then find it amusing.
- If my interlocutor makes a joke, but I have no sense of humour, find it boring.
- If my interlocutor makes a joke, but I have no sense of humour, but we are good friends, then find it friendly.

Trait-specific conditionals would override the more general-purpose conditionals, so the person would respond in character.

The implementation of traits in The Sims 3 involved a variety of approaches. (1) and (3) were strongly data-driven approaches, whereas (2) involved a lot of procedural code. One of the motivations for the traits-as-conditionals approach, which this paper proposes in the next section, is precisely to replace this heterogeneous hodgepodge with a *uniform* representation where all the data is in one place.

Traits in *The Sims 3* satisfied some of our core goals. Personality traits affected autonomy via a distinct motive associated with each trait. Further, different personalities did have different emotional responses to the same stimulus: Sims who disliked children would get irritable in the presence of children, while family-oriented Sims would enjoy their company.

The trait system was a major step forward from previous versions of *The Sims*, and many reviewers noticed how the trait system in *The Sims 3* made the characters richer: "The Sims themselves are now powered by much more sophisticated psychological systems than found in earlier games… Traits are designed to reflect how people describe themselves in the real world and are so eerily portrayed in their behavior that *The Sims 3* feels like an anthropology study with teeth" [19].

But a number of our other goals were not satisfied. To start with, adding a new personality trait meant making a wide number of different types of changes before it was manifest in behaviour: you needed to add an associated motive, define a variety of trait-specific social interactions, define how that personality-trait responded to social interactions initiated by others. If you wanted that personality-trait to have different emotional responses, you had to sprinkle the code with if-statements. Each of these aspects of trait manifestation was expressed in a different representation, so there was a substantial authoring burden when adding a new trait.

Although you could make a large number of personality types in The Sims 3, the number was finitely bounded. There was no concept of describing personality traits in a language with recursive structure, allowing an infinite number of possible traits.

Further, the personality traits were simple atomic objects (elements in an enumeration). This meant there could be no explanation in the model of why one trait was incompatible

with another, so incompatibility between traits had to be authored by hand. This was time-consuming and error-prone.

Because traits were atomic objects, there could be no understanding of why personal events could determine personality. Intuitively, if a character suffered a traumatic early event involving a dog, this would explain a subsequent fear of dogs. But in *The Sims 3*, the personality trait of fearing-dogs had no constituent structure – it had no understanding that fearing-dogs involved *dogs*, so there was no way to connect the event (being traumatized by a dog) with the trait (fearing dogs).

4 AN ALTERNATIVE: REPRESENTING PERSONALITY TRAITS AS CONDITIONALS

This paper describes a richer model of personality traits than that used in *The Sims 3*. Instead of a trait being a simple atomic object, whose effects are scattered through the code and data, now a personality trait is represented by a declarative[3] conditional specifying the condition under which the character has an emotional state. For example, Jealousy could be represented as:

If my partner talks to another → Anxiety

Thrill-seeking could be represented as:

If I perform a risky action → Excitement

Tearful could be represented as:

If something gets the better of me → Upset

Honest:

If I say something false → Shame

Compliant:

If I don't do what somebody tells me to do → Shame

The general pattern is that the left-hand-side of the conditional is a world-state, and the right-hand side is an emotional state. The emotional states are intrinsically motivating: the agent wants to achieve some, and avoid others. So by specifying emotional consequences we are *indirectly* specifying what the agent wants.

Some personality traits are represented by a cluster of conditionals. For example, Argumentativeness could be represented by a pair:

- If somebody contradicts something I say → Angry(Contradiction)
- If Angry(Contradiction) /\ I prove someone else wrong → Anger dissipates

The first specifies the condition under which the argumentative person becomes annoyed. The second describes the conditions under which that anger dissipates: by proving somebody else wrong.

Being Vengeful can also be represented as a pair of conditionals:

- If x harms me → Vengeance(x)
- Vengeance(x) and I take revenge on x → Vengeance(x) dissipates

(Note that this example requires emotional state with *constituent structure*: the Vengeance emotion is directed towards a specific individual, x. In my implementation, this structure is encoded naturally in Exclusion Logic [10]).

5 AN AGENT ARCHITECTURE WHICH SUPPORTS TRAIT-CONDITIONALS

In *The Sims 3*, actions are tagged directly with the trait-motives that they satisfy. This involves a tight coupling between the set of actions and the set of traits:

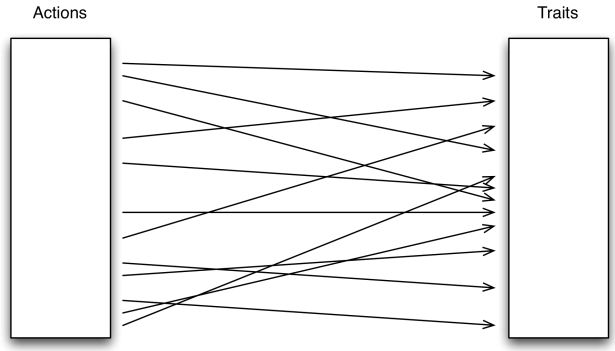
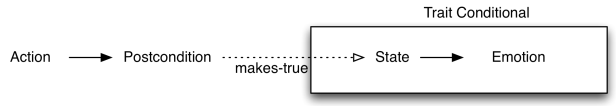


Figure 1. Tight Coupling between Actions and Traits

The action promises that if you perform it, it will satisfy the motive in question. A large number of autonomy bugs came from places where this promise is broken: places where an action claimed to satisfy a motive, but in fact did not do so for one reason or another, because of failure or unforeseen conflict. In these cases, the Sim is unmasked: continually repeating the same action sequence over and over again, falsely believing that this time he will get the satisfaction that has so far eluded him. (Another set of problems from the opposite direction stem from the fact that the advertised motives specify what *typically* should happen when the action is performed. But if this *particular* instance of the action in this particular situation would also satisfy *another* motive, this fact will be entirely lost on the Sim. Sims will never, in other words, be opportunistic because satisfied motives are assigned to action-types at design-time, rather than assigned differently to different action-instances at run-time).

To support traits-as-conditionals, we need a different agent architecture – one which involves a looser coupling between actions and traits. In this alternative agent-architecture, actions specify declarative post-conditions. They specify what will be true when the action has been performed – not what goals will be satisfied when the action has been performed. These post-conditions in turn make true the antecedents of the trait-conditionals:

[3] Isn't it redundant to say that the conditional is declarative? Aren't all conditionals declarative? No – some conditionals are *deontic*: they specify what *should* happen under certain circumstances, not what *will* happen under certain circumstances. I previously considered an alternative model of personality traits based on these deontic conditionals [11]. This is why I stress the *declarative* aspect of the conditionals used here – to distinguish them from *deontic* conditionals.



Instead of actions directly tagged with motivating factors, there are two extra levels of indirection: actions are tagged with postconditions, and those postconditions make the antecedents of the conditionals true, which in turn activate the emotional response. This involves a looser coupling between the actions and the emotions:

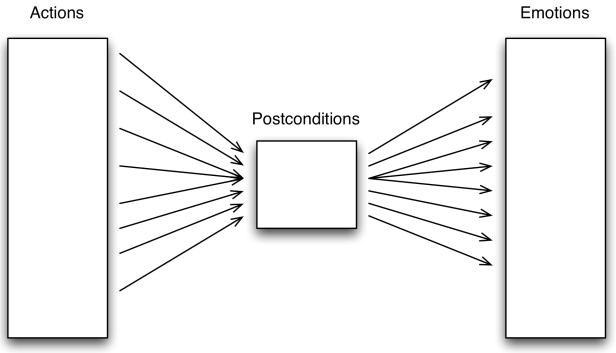


Figure 2. Looser Coupling between Actions and Emotions

In this alternative architecture, the trait conditionals are performing *double-duty*:

- They are used by the planner to decide what action to perform: the agent calculates the emotional state he would be in if he performed the action. The emotional state is intrinsically attractive or unattractive.
- They are used by the simulator to update the emotional state of the agent. When he performs an action, the postconditions are added to the database. This in turn makes the conditional's antecedent true, which in turn updates the agent's emotional state.

6 EXAMPLES OF TRAITS REPRESENTED AS CONDITIONALS

The conditional model can naturally express a wide variety of personality characteristics. For example, it can express the so-called Big Five traits (a statistical agglomeration of a very large number of traits). For each of the big five, there is a pair representing positive and negative values of the trait:

Open to Experience:

If I receive a new experience → Excitement

Closed to Experience:

If I receive a new experience → Anxiety

Conscientious:

If I do a job properly → Satisfaction
If I do a job badly → Shame

Careless:

If I do a job badly → Satisfaction

Extroverted:

If I meet a new person → Excitement

Shy:

If I meet a new person → Anxiety

Agreeable:

If I am friendly to somebody → Satisfaction

Disagreeable:

If I am mean to somebody → Satisfaction

Neuroticism comes in a variety of flavours. We give one example - neuroticism with respect to bodily contact:

If I am touched by somebody → Repulsion

Well-adjusted (the opposite of neurotic):
Again, we only give one example: being well-adjusted with respect to bodily contact with someone I trust:

If I am touched by somebody who I trust → Warmth

This model can also accommodate the more specific personality traits of *The Sims 3*. For example:

- Childish: Performing childish action → Enjoyment
- Commitment issues: being in committed state → Anxiety
- Couch-Potato: Exercise → Anxiety; Eating → Enjoyment
- Coward: Doing brave action → Anxiety
- Dislikes children: Being around children → Irritation
- Excitable: Unremarkable event occurs→ Excited
- Family-Oriented: Giving nurturance/help to children → Enjoyment
- Flirty: Receiving attention from attractive men → Enjoyment

7 TRAITS-AS-CONDITIONALS SATISFY THE DESIDERATA ABOVE

(1) Decomposition. This model satisfies the decomposability requirement, just as *The Sims 3* does: a personality is a bundle of independent personality traits. But now each trait is represented by a conditional with constituent structure, rather than by an atomic unit.

(2) Autonomy. In the traits-as-conditionals model, traits directly affect emotion, and *indirectly* affect autonomy. The trait conditional specifies an emotional state on the right-hand-side. This emotional state is intrinsically motivating: he either wants to be in it or wants to avoid it. So the conditional indirectly tells him what he should do.

(3) Personality affecting emotion. The conditional does double-duty in this architecture. It is used by the planner to decide what to do, and it is also used by the simulator to update emotional state: when the conditional fires, the emotional state is updated directly.

(4) Minimize the authoring requirements to add a new trait. In *The Sims 3*, traits were linked directly to action. If there were *n* actions and *m* traits, there was a sparse matrix with *n* * *m* entries. In the conditional model proposed here, by contrast, there is a small finite intermediary between the world state and the traits: a list of emotional states. So adding a new trait is significantly less burdensome in this conditional model than in *The Sims 3*.

(5) Supporting an indefinite number of personalities. One significant advantage of the traits-as-conditionals model is that it can express an *indefinite* number of traits. A trait is just a conditional, expressed as a horn-clause. There are just as many possible personality traits as there are possible horn clauses.

(6) Supporting various levels of granularity. Traits-as-conditionals make it very natural to express traits at varying levels of specificity. We can define a generally mean-spirited character as:

> If I am mean to somebody → Enjoyment

We can define somebody who is mean-spirited to women by adding an extra conjunct on the left-hand side:

> If I am mean to somebody female → Enjoyment

We can keep adding conjuncts on the left-hand side, without end, to make more and more specific conditionals.

(7) Supporting the idea that some traits are incompatible. The traits-as-conditional model also provides an explanation of why certain traits are incompatible. Two traits are incompatible if the left-hand-side of one is entailed by the left-hand-side of another, but the emotional state on the right-hand side of one is different from the emotional state of the other. For example, Good-Sense-of-Humour could be characterized as:

> If somebody tells a joke → Amused

No-Sense-of-Humour could be described as:

> If somebody tells a joke → Bored

The constituent structure of the conditionals makes it possible for the machine to *automatically detect* which traits are incompatible, rather than (as was the case in *The Sims 3*) having to hand-author all incompatibility-pairs by hand.

(8) Supporting the idea that past history can explain personality. Finally, because the traits-as-conditionals approach treats a trait as a declarative sentence with structure, it can naturally accommodate the idea that personal narratives explain traits. For example: it is easy to see how, after being traumatized by a dog in infancy, to add a conditional:

> If I see a dog → Anxious

If traits are conditionals, particular traumatic or transformative moments could be turned dynamically into traits that have been generated on the fly by the situation:

> If I am in a situation which has aspect F, and I am having a traumatic / transformative experience, then add a trait conditional: If the situation satisfies F→ Anxiety / Enjoyment

This last suggestion is largely programmatic. I have brushed over the considerable issue of how the agent chooses *which aspects of the traumatic situation merited the anxious response*. If the agent was traumatised when standing in front of a barking dog on a sunny Tuesday, which trait conditional does he add:

> If I see a dog → Anxious
> If it's sunny → Anxious
> If it's a Tuesday→ Anxious

Nevertheless, acknowledging that this architecture does not directly answer this question - if we *do* separately find a good answer to it, then the traits-as-conditionals architecture is well placed to support the ability to learn traits on the fly based on past experiences.

8 RELATED WORK

Many games, RPGs in particular [6], have used individual personality traits. But the personality traits that are chosen in these games merely affect the *stats* of his avatar - not the *autonomous behaviour* of all the NPCs. What is distinctive about the approaches described in this paper is that personality deeply affects autonomous behaviour. This is what allows autonomous improvisation.

Some previous systems [8] had a model where traits affected autonomy. But in these early systems, each agent had the same set of personality aspects. The only thing that differed was the *numeric value* of each aspect:

> "The Universe program uses a trait-based personality model. Each story world character is represented by a *person frame* which stores information about that character such as the character's name, stereotypes, traits, interpersonal relationships with other characters, and the character's history. Traits, such as intelligence, moodiness, and promiscuity whose values range in integral value from 0 to 10, are continuous dimensions and the degree to which a character manifests a trait is stored as an integer value. Traits such as intelligence and moodiness and promiscuity have ranges between 0 and 10. Traits such as guile, self-confidence, and niceness have ranges between -10 and 10 where a negative value indicates that that character has the opposite of the trait".

In *Universe*, the personality differences are *quantitative* differences (the difference between having a 5/10 and a 8/10) as opposed to the *qualitative difference* you get if a trait is modelled by having an element that would otherwise *not be there at all*. *The Sims 1 & 2* similarly had a personality model based on a

small number of quantitative differences [12], rather than a large pool of qualitatively distinct elements.

The approach described in this paper is clearly related to cognitive appraisal theory [13]. Cognitive appraisal theory is a psychological theory that explains why people have the emotions they do, and how the same stimulus can elicit different emotional responses in different people. The explanation of an emotional response involves two types of judgement: the *primary* appraisal is the agent's judgement whether the outcome is in line with her desires and goals. The *secondary* appraisal is her judgement whether she is able to affect the outcome, or to what extent she is helpless. In the traits-as-conditionals approach described here, the fact that the situation causes the emotion is *unexplained*: it is just taken as given that this sort of person will respond to this sort of situation with this sort of emotional response. The cognitive appraisal theory is a deeper theory in that it attempts to *unpack* why this conditional is true by appealing to primary and secondary appraisals. The cognitive appraisal theory is more *cognitive* than my trait-conditionals in that it attempts to explain the emotion in terms of the agent's *judgment* about discrepancies between what is and what should be the case: it is the discrepancy between the agent's judgment of how the situation has turned out and how it *should* have turned out which *explains* the emotional upset.

Cognitive appraisal theory has been used to update the emotional state of the agent after action has been performed. But it has not been used to determine action-selection. In the traits-as-conditionals approach described in this paper, a trait-conditional performs *double-duty*. It is both used to update the emotional state based on what *did* happen, and also used to anticipate the emotional consequences of what *might* happen in order to decide what the agent should do next.

There are a number of other systems that use personality to influence action-selection [16, 17]. In these systems, personality affects action-selection in the following direct way: the action is tagged directly with the personality-trait or emotional state that it satisfies. For example: the action of eating chocolate is tagged with the choc-aholic personality-type. The model described here, by contrast, is much more truly simulationist in that, instead of specifying the consequences of the type of action directly in terms of emotional state or personality state, we are specifying the consequences of that particular action in terms of world-consequence, and then, as a separate step, we compute what the emotional update of that consequence is in the current context. For example: the consequence of eating chocolate is that chocolate is consumed. The personality-trait of being a choc-aholic means that consuming chocolate is particularly pleasurable. In this particular case, the consequence is the same, but the extra level of indirection gives us the ability to be *sensitive to the specificities of the situation*. E.g. the consequences of moving a pawn forward in a particular chess situation depend on the *precise state of the board*.

The traits-as-conditionals approach proposed here is based on the personality model developed by Walter Mischel [7]. Mischel was a situationist and interactionist who developed a powerful critique of the big-five trait model, and eventually produced a constructive alternative based on situation-sensitive conditionals. But one major different is that the conditionals Mischel considered were *deontic* conditionals, relating world-state to the action the agent *should do* – rather than conditionals relating world-state to *emotional state*, as proposed here.

9 IMPLEMENTATION

The traits-as-conditionals approach described here has been implemented in a multi-agent simulation. In one scenario, two agents are playing tic-tac-toe. They both want to win, but one of the agents has a personality trait of being a bad loser: losing is particularly upsetting for him. The other has a trait of being sensitive to the other's feelings: seeing that the other is feeling upset means that she also feels upset. In this situation, when the sympathetic player is about to win, she will anticipate that her winning will upset the other, and sees that him being upset will also upset herself. So she deliberately avoids winning, and aims for a draw, to spare his feelings.

Initial results suggest that the authoring burden is significantly lighter when specifying traits as conditionals. This is precisely because the conditional does *double-duty* in determining both emotional effects and action-selection.

10 CONCLUSIONS

This paper has contrasted two ways of implementing autonomous personality traits in synthetic characters: the trait model in *The Sims 3*, and the traits-as-conditionals approach. This paper proposed a set of requirements and goals that any implementation should satisfy. It has been argued that, although *The Sims 3* does a reasonable job of satisfying these goals, the traits-as-conditionals approach does a better job:

	The Sims 3	**Traits-as-Conditionals**
Personality decomposable into traits	Yes	Yes
Personality affects autonomous action	Yes	Yes
Personality affects emotion	Yes	Yes
Minimal authoring for adding new trait	No	Yes
Indefinite number of personalities	No	Yes
Some traits are	No	Yes

refinements of others		
Model explains trait incompatibility	No	Yes
Personal narratives can explain traits	No	Yes

Perhaps the major advantage of the traits-as-conditionals approach is that, because one conditional does double-duty in determining both emotional update and action-selection, the authoring burden is lighter. This consideration becomes increasingly important as we scale up from academic proof-of-concept implementations to industrial-size implementations, with hundreds of personality traits and tens of hundreds of different types of action.

REFERENCES

[1] C. Darwin. *The Origin of Species*.

[2] G. Miller. *The Evolution of Adaptive Unpredictability in Competition and Courtship*. Machiavellian Intelligence II. Cambridge University Press.

[3] Maxis. *The Sims 3*. Electronic Arts.

[4] R. Burkinshaw. *Alice and Kev*. http://aliceandkev.wordpress.com/

[5] A. Schopenhauer.*The World as Will and Representation*. Dover (1966).

[6] Black Isle Studios. *Fallout*. Interplay (1997).

[7] W. Mischel & Y.Shoda. A cognitive-affective system theory of personality. *Psychological Review, 102.* (1995)

[8] Lebowitz, M. (1984). Creating characters in a story-telling universe. *Poetics*, 13, 171-194.

[9] Mccoy, J., Mateas, M., and Wardrip-fruin, N. Comme il Faut : A System for Simulating Social Games Between Autonomous Characters. *Proccedings of the 8th Digital Art and Culture Confernce (DAC 2009)*, (2009).

[10] R. Evans. *Introducing Exclusion Logic as a Deontic Logic*. DEON (2010).

[11] R. Evans. *Artificial Intelligence in The Sims 3*. Sims 3 website. http://sims3.fr/view/pages/blog.jsp?author=Richard%20Evans#ArtificialIntelligenceinTheSims3

[12] K.Forbus & W.Wright. *Some Notes on programming objects in The Sims*. http://www.qrg.northwestern.edu/papers/files/programming_objects_in_the_sims.pdf

[13] S.Folkman & R.Lazarus et al. *Dynamics of a Stressful Encounter: Cognitive Appraisal, Coping and Encounter Outcomes*. Journal of Personality and Social Psychology 1986.

[14] A. Ortony et al. *The Cognitive Structure of Emotions*. Cambridge University Press.

[15] C. Crawford. *Personality Modelling for Interactive Storytelling*. http://www.ieconference.org/ie2004/proceedings/003%20crawford.pdf

[16] K. Perlin & A. Goldberg. *Improv: A system for scripting interactive characters.* SIGGRAPH 1996.

[17] M. Lim, R.Aylett & C. Jones. *Affective guide with attitude*. First International Conference on Affective Computing and Intelligent Interaction. 2005

[18] E. Andre et al. *Integrating models of personality and emotion into lifelike characters*. Proceedings of the Workshop on Affect in Interactions.

[19] J. Ocampo. *The Sims 3 Review*. IGN. http://uk.pc.ign.com/articles/988/988108p1.html

Social Objects – A framework for social interactions between videogame characters

Phil Carlisle, Dr Steve Manning, Dr Mark Grimshaw
University of Bolton

Abstract. We present social objects, a method of enabling social interactions between videogame characters based on an extension of the smart object concept. Social objects provide a mechanism for the control of one or more videogame characters in a coordinated group social interaction. Our motivation being the reproduction of behaviour seen in any street, park, town square or shopping mall setting. Primarily for use as "ambient characters" to populate a city environment within a videogame.

1 INTRODUCTION

Videogames have become capable of rendering many hundreds of characters (often called NPC's, a shorter term for non-player characters) at interactive framerates. This opens up new potential for crowds to become part of games in a manner that was technically impossible previously. A recent example of crowd use in a videogame context is within the game "Heavy Rain" [1] which used a potential field approach similar to [2] to simulate a dense population within a shopping mall context. Whilst the simulation in Heavy Rain serves its purpose (to obscure the player's view of their son during an interactive narrative sequence) it does not provide a convincing crowd scenario for the general case. Similarly a less dense crowd is simulated in the game "Grand Theft Auto IV" [3] as a means to provide an ambient background cast to the city within which the player is situated. Again, this simulation fails to model many of the social interactions one would expect to see in any observation session in a similar environment in reality. It is reasonable to assume that the focus of the game is primarily on the player controlled character and their interaction with the mechanisms of the game rather than on the social simulation, however this brings up a question. What if the social simulation were part of the game mechanisms?

It was this question that motivated the research presented within this paper. Interestingly a recent game "Spy Party" [4] has been receiving press attention, primarily because it focuses on this aspect of social behaviour. To quote the creator Chris Hecker "SpyParty is a **spy game** about **human behavior, performance, perception, and deception**".

This paper is organised as follows: in section 2 we will cover current research relevant to our motivation, in section 3 we will present the social object model and the framework within which it functions, in section 4 we will address the issue of evaluation, finally in section 5 we will conclude with a discussion of the current shortcomings of the model and the ongoing development intended to address them.

2 RELATED RESEARCH

Allbeck [5] developed an architecture called CAROSA (Crowds with Aleatoric, Reactive, Opportunistic, and Scheduled Actions), which along with the parameterized action representation (PAR) demonstrated an ability to coordinate agents in a variety of tasks in a social setting. The PAR concept can be closely related to the social object model as one of the core purposes of both is to constrain and control the interactions of agents. Each system is designed to model the functionality of an item or action and to represent the preconditions, failure conditions and semantics of interaction. However the focus of Social Objects is the proxemic and animation control of a (virtual) bodily interaction between multiple agents in a social context, which is an area that the CAROSA architecture does not currently address.

Matteas & Stern demonstrate coordinated social behaviours in Facade [6] which uses a behaviour tree language called ABL, a descendant of HAP in order to coordinate characters within a three dimensional world [7]. The focus of Facade is the interplay between the player and two characters, within the context of the characters' apartment. The player is challenged to act as a mitigating force during a conflict between the two characters who are a married couple with a troubled relationship. Whilst the Social Object model was very much inspired by the concepts in Facade, such as the use of a reactive behaviour tree for agent control, the focus of Social Objects is the proxemic and animation control of such interactions. The assertion being that psychologically we are attuned to social interactions involving coordinated acts such as touching, posture, gesture and gaze changing and that in order to portray social interactions convincingly we should pay specific attention to these areas. Essentially the goal of Social Objects is to provide a subset of the work presented in Facade but with a focus on the those aspects of social interaction that provide the most "human like" behaviours for the least amount of development effort. The thesis being that a focus on proximity and animation is sufficient to reproduce believable human like performances of the type typically required for social crowd scenes as previously described.

Pedica [8] dicusses social proxemic behaviours in work developed by the CADIA research lab at the University of Reyjkavic, although this work was focussed on the social proxemic organisation within group discussion Pedica also describes the importance of gaze and posture changes during conversation and incorporates the notions of interpersonal distance of Hall [9] Scheflen [10] Kendon [11] and Goffman [12]. Similarly the work of Jan & Traum [13] focuses on

adapting crowd like social forces models to group discussion scenes. Both offer insight into the requirement for proxemic and orientation control during group conversation. However both of these works address group conversation without consideration of a greater framework to use for different forms of group social activity.

The work of Arinbjarnar [14] is related as it offers the notion of "schemas", which are reactive passages of narrative that are formed by applying constraints against an abstract authored narrative passage. The advantage being that narrative emerges from matching current situations to multiple potential schemas. Whilst the focus of such work is on creation of narrative actions, it is easy to understand that such work might be applied to social interactions. This approach also allows for concurrency as schema's are all considered together within the constraints that have been applied.

Within videogames, work by Evans [15] Isla [16] and Orkin [17,18] have all demonstrated coordinated human-like behaviour. While they have informed and inspired the current approach, the specific goal of coordinating proxemic and animation behaviour has not been clearly addressed in the available videogame literature.

Navigation

Crowd simulation in research tends to be focussed on the simulation of large scale crowds, often for the purpose of studying crowd flow, but also for the study of crowd behaviour. A good overview of the available literature is provided by Pelechano et al [5]. Pelechano describes the available literature in terms of microscopic and macroscopic approaches. Macroscopic approaches such as [2] tend to model crowds as aggregates and are suitable for simulations of large crowds. Microscopic approaches [19],[20] tend to model individual agents, often with models of personality, emotion and memory. These microscopic approaches limit the scale of crowd that can be simulated at interactive rates, yet they are more suited to the type of representation required for videogame use. Typically, crowd related literature is concerned with crowd flows [21] or aggregate crowd behaviour [22], but these are generally based around the requirements of accurate simulation. For the purposes of videogames the goal is not to provide "accuracy" but instead to provide "believability".

In order to do this, we must express commonly observable crowd dynamics such as lane formation, stacking at points of contention such as doorways, alongside more social interactions like group discussions, people in a hurry, browsing in shop windows and outdoor social activities such as playing a ball game.

The core of any crowd simulation is the navigation model [23]. Aggregate models such as potential fields [2] tend to treat space as a grid and agent movement is determined by a cost function within adjacent cells of the grid. Whilst such models achieve considerable speed advantages as they can be easily computed, they tend to fail when observed on an individual agent level. The discreet nature of the grid structure is an inherent problem which has also been reproduced in many grid-based approaches used in videogames. The problem being that most NPC's are expected to follow relatively straight lines of movement, however anything other than horizontal or vertical movement on a grid is liable to cause a "stairstep" aliasing effect, as new positions are sampled along a the straight line.

Another common approach to navigation is the use of a discreet network of world space positions connected by edges which represent straight line movements. Typically the straight line segments are sampled with simple rays to detect collisions along the line. Such naive sampling may result in potential collision artefacts as obstacles may be situated anywhere within the world space. A more robust sampling mechanism uses a swept volume intersection test to determine any potential collisions along the volume between the nodes.

However even such a relatively complex test can only determine collision intersections with static objects. In videogames with crowd simulations it is reasonable to assume that there will be many dynamic obstacles (i.e. the crowd agents) moving at any given time-step. In order to solve this problem, current videogames typically use a navigation mesh. This is a set of polygons (typically convex polygons or rectangular areas) which represent navigable space and rather than using line segments or grids, offer an area within which it can be deemed "safe" for an agent to move. Dynamic collision avoidance then becomes a matter of solving the positions of all agents such that they remain within the bounds of the collision free space of the navigation mesh whilst also not intersecting other agents.

Agent to agent intersection is prevented using either a simple force based model [20] or a geometric model such as reciprocal velocity obstacles [24].

Assuming the ability to navigate around the world as a precondition, we then must consider the actual behaviour of agents in the world, how they interact and the qualities of their behaviour as seen from a player perspective.

Facade provides a compelling environment in which the AI driven non-player characters interact meaningfully with the player. However they do not interact convincingly with each other. One of the primary aims of this work, is to address those shortcomings. We aim to address the need for coordination between multiple NPC's, paying specific attention to the three areas of proxemics, coordination and animation.

Proxemics

Hall [9] and others within the field of social psychology [25-27], have identified that humans tend to observe a set of social "distances" which can be viewed as a set of concentric rings around the subject of interaction. The study of this area led to the field of proxemics and has an impact on the representation within the social object model. In effect, proxemic distance and the angle between interacting NPC's must be closely monitored and controlled if we are to reproduce many social interactions. Kendon [28] identifies these proxemic notions in terms of "f-formations", these formations represent social spaces, especially among groups of individuals involved in discussions and are

dynamic in nature dependant on the disposition of the group and its environment.

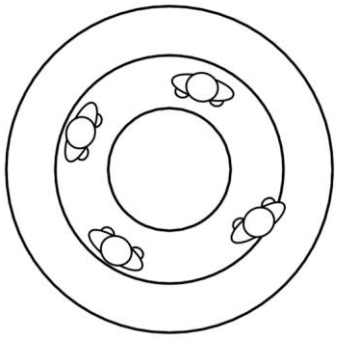


Figure 1 – Proxemic distances visualised as concentric rings.

Coordination

Most social interactions require coordination between two or more individuals. These interactions require that both individuals are aware of and can be considered participants in the interaction. Many non-verbal signals are sent between the participants in anticipation of the interaction in order to coordinate the appropriate reaction [8]. For instance two friends might approach each other and smile before mutually sharing handshakes or hugging each other.

Animation

Specifically for videogames, one of the biggest issues in reproducing social behaviour is the issue of animation selection, playback and coordination. This is an issue because animation data sets are limited and visual errors such as animated body parts inter-penetrating are visually jarring. The use of motion blending and inverse kinematic models [29] to some extent can help alleviate some of these issues, however there remains a problem of processing cycles being available to compute the expensive inverse kinematic constraints for a large number of interactive characters.

3 THE SOCIAL OBJECT MODEL

The social object model can be considered an extension of the "smart object" model [30] employed in games such as "The Sims" [31][32]. The purpose of the smart object model in the game was to coordinate the position and animation of an NPC such that it could perform tasks related to virtual objects within the game environment. A typical example is that the use of a shower would require the NPC (called a "sim" in the game) to navigate the environment to stand in front of the shower, perform an animation to open the shower door, step inside the shower, close the shower door, switch the shower on and perform an animation whilst showering. The most interesting feature of this approach and the reason it is called smart objects, is that the functionality for controlling the NPC lay in the scripts of the objects being used. In the previous case, the shower would have a script that describes how the NPC would use it. Each object then propagates through the system its availability and function.

This external control is a key feature of the social object model. An interesting observation regarding social group behaviour is that the duration of many social groupings extend beyond the participation of the initiating individuals [11]. Consider the case where a couple start a conversation in a public place. Later a friend joins them, then another. Soon a small group forms, each member participating in the group discussion. Then the original couple decide to continue with their previous plan and leave the group. Often the group will continue for some time, even though the originators of the discussion have now left.

It is the realisation that social groupings exist outside of the participants of the group that motivates this work. In practical terms, the social object is a multi-agent coordination architecture, which can be used to mimic the creation of social interactions between two or more agents.

Social objects work by implementing a simple architecture that propagates positional, role and animation data to agents accepted into the social group. Each social object is defined by a behaviour tree [7] which acts to keep the group coordinated by performing role allocations during interaction. Individual agents are coordinated using a shared blackboard [33] which represents position and orientation data for each agent. An event system is used to request appropriate animations for each agent in the group at the required time. The social object itself is defined as a C++ class which contains a behaviour tree and blackboard. This class is typically instantiated when an agent reaches a given threshold for participation in any particular social activity. For example, an agent who perceives themselves to be "bored" will then construct a social object which serves to coordinate some group social entertainment activity. We term the creator of the social object the "proposer", although the social object construction can also be performed without a proposer, as is the case where an entertainment situation is represented in the game world outside of any given agent. For example, a ballgame might require that NPCS's enter a given play area before they can be considered playing the game. Hence the social object would essentially be a token in the game world that is described by the bounds of the playfield and would reactively respond to any NPC's that enter the area of the playfield in order to coordinate the gameplay.

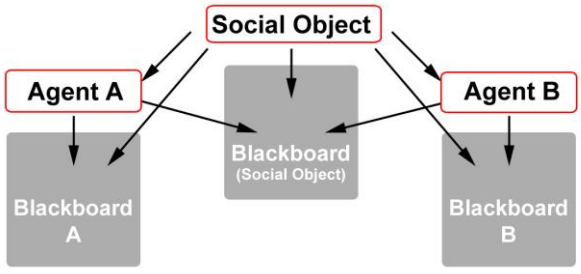


Figure 2 – The social object acts as a shared blackboard and coordination agent. It propagates events to all participants and can alter participating agents' blackboards in order to coordinate activity.

Acceptance into the social group

Normally, the social object would track the location of the proposer and be represented in all agents sensory model such that they can perceive the availability of the social object. Upon the perception of the social object, a second NPC would then send an event to the social object that it is available to perform the social interaction. The social object places the NPC into a potential participants list and responds with an event which contains a set of constraints. These constraints are generally proxemic, time and/or animation based. For instance the proxemic constraint would require that any potential participant would be within a maximum distance/time to allow the social interaction to take place within a timely manner. Animation constraints would require that potential participants are capable of playing particular animations required for the interaction. For example if the social interaction is "greeting with a hug", then the animation constraint would require that the "hug" or "be hugged" animation be available to the participant.

These constraints serve to reduce the number of potential participants while ensuring that the social object is not required to be fully aware of their capabilities.

If the potential participant is capable of fulfilling the constraints for the interaction, it sends an event to the social object to that effect. At this point the social object either accepts the event and adds the NPC into the social group, or rejects the request (for example if the NPC is too far away to realistically join the group). The current system is implemented such that all constraints must be satisfied before a social interaction is performed, however it is worth considering that partial constraint satisfaction may be appropriate in specific cases where the social interaction is reasonably complex and yet a simplified version of the interaction would be suitable.

Once a social group is formed such that there are enough accepted NPC's to perform the interaction, the social object first confirms the availability of each agent to ensure they have not entered another social group, it then proceeds to coordinate the interaction.

Coordinate frames

One of the most important aspects of the social interaction is maintaining the appropriate proxemic position and orientation for each agent. In order to allow for more accurate animation synchronization, especially during interactions where multiple agents touch each other, a coordinate frame is selected for all proxemic position and orientation values. This coordinate frame is typically relative to a specific agent, however it may also be relative to the social object itself, another object, or a world origin. This allows for accurate creation of animation sequences, with known reference distances required between them, for example one agent patting another on the back or hugging requires a specific orientation and distance between both characters if it is not to display inter-penetration artefacts when the two characters play their responding animations.

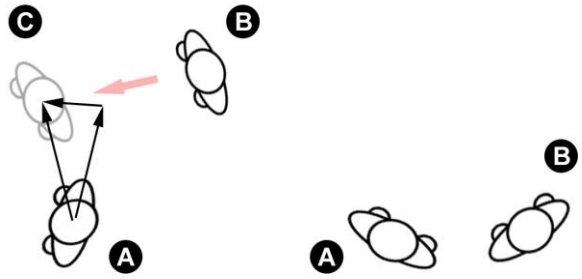


Figure 3 – Left, character B is moving to position and orientation C, which is defined relative to the transform of character A. Right, character B has moved into position as seen from another viewpoint.

Proxemic distance and orientation is typically coordinated using a simple force based model similar to [20], where position and orientation are loosely defined, however more specific constraints can be applied for more demanding interactions. In general, social interactions of the type seen in a city environment require loose proxemic constraints, with the occasional hard constraint for interactions such as touch-based greetings.

Sequencing interactions

Interaction sequences are specified within the branches of the behaviour tree. Each sequence has a number of constraints which are required of the participants, such that the sequence may or may not be prioritized for consideration when the tree is evaluated. These constraints act as a filter for individual sequences of interaction and work much in the same way as the constraints were when choosing participants for the group.

Typically each sequence has a decorator applied which modifies the sequence priority after the sequence has been executed, which allows for other sequences to be applied during the social interaction, for example a number of variations of turn taking signals may be displayed during a conversation.

The behaviours within each sequence correspond to the phases of interaction and are modelled as atomic actions within the tree. Actions such as "mark NPC <id> as performer", "mark NPC <id> as listener", "play animation <id>", "modify NPC blackboard" etc serve to coordinate NPC behaviour. Typically they modify the data within the individual agent blackboard such that the NPC's own behaviour tree responds to the new data. In this way, the social object behaviour tree can be thought of as a method of enabling or inhibiting individual NPC behaviour when within the social context.

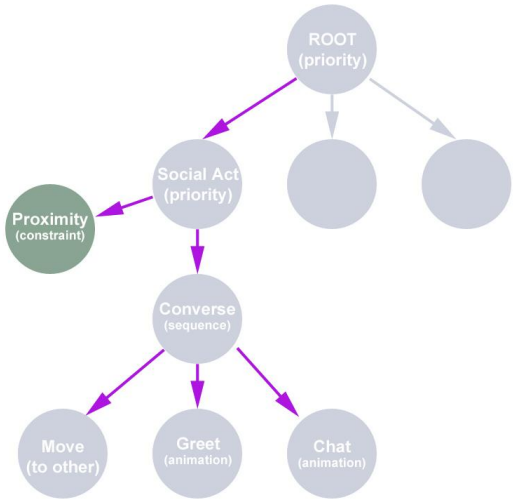


Figure 3 – Example behaviour tree showing potential social act. Darker node represents a proximity constraint which must be satisfied before the social act may be considered for activation. Darker arrows represent path of execution when this is true.

Gaze control is coordinated such that NPC's track the current nearest "performer" within the group. Individuals in the group are labelled as performer or listener depending on the social situation being modelled. Multiple performers and listeners are possible to allow for a variety of social contexts.

Animations are similarly coordinated, with performer and listener each animating differently based on their current role and individual personality. It should be noted that each NPC within the group is notified of their role, but is responsible for the performance of that role. Thus animation selection is the responsibility of the individual NPC allowing variation in individual displays for the same social context.

As each NPC perceives the threshold of desire for the current social interaction to drop (usually in response to the satiation of need) they leave the social group by informing the social object of their withdrawal. The social object then monitors the remaining participants within the group and once the number of group participants drops below a given threshold the social object informs any remaining participants of the cessation of group activity.

4 EVALUATION

Evaluation of the proposed model is an ongoing problem which we have not yet fully addressed in our current work. Typical crowd research tends to use observation of particular crowd phenomenon in both a real world situation (normally from film) and a simulated environment as a means of evaluating the simulation. This is in many ways the approach used in social psychology under the name context analysis [11], the so-called "natural history" approach. The specification of such subtle animation behaviours such as gaze shifts during a conversation and extraction of meaningful data from example video suggests that automated extraction of such events is a desirable goal. Motion capture techniques have been used in crowd modelling [34] and virtual character behaviour modelling [35] as a means of providing source data for statistical analysis and examples of model behaviours. In terms of ease of acquisition and invasiveness motion capture currently has many drawbacks, however the field of markerless motion capture [36] and the introduction of low cost technologies associated with it, such as Microsoft's "Natal" may offer a useful approach in the near future.

Another form of analysis used in virtual humanoid research is Laban Motion Analysis [37] [38] which uses a notation (Labonotation) to describe the Body, Space, Shape, Effort, and Relationship of movements. Developed by Rudolph von Laban and extended by Bartenieff and others, this provides a method of classifying and annotating human bodily movement. It is exactly this bodily movement that is key to social interactions.

Hall devised a notation called "Proxemic notation" to describe the positional and orientation relationships between participants in social interactions. Using this notation, observers were able to annotate live observation sessions between humans in a natural setting, recording their relative distance and orientation in a manner that allowed the observer to capture the changing proxemic relationships.

The concept of a notation that is derived from observation, such as those seen in LMA and Proxemics, is likely to be a key factor in evaluation. Ideally, sequences of actual interactions between humans would be automatically annotated from sequences captured either on film or via motion capture, thus forming a corpus which could be statistically compared to simulated examples of similar interactions. A key advantage to such a technique would be that there would be no differences between observers perception of the interaction, observations could be performed more frequently and with a much reduced workload for the researcher. Unfortunately there is still significant work to be completed in the area of automated capture and annotation for this specific purpose.

The question of measuring the players perception of such social simulations is a significant one. Work by Bailenson [27,39] may offer some insight, however the greater task of measuring engagement and immersion suggests that work by Nacke [40] on the psycho-physiological measurement of immersion would be appropriate.

5 CONCLUSIONS & FUTURE WORK

This paper shows how a relatively simple extension of the smart object model can allow for coordinated social group behaviour within crowds of NPC's. In the architecture and implementation presented so far, there are a number of shortcomings which we are working to overcome. One limitation is that each NPC can only be a participant in one social group. Whilst this seems like a reasonable approach, in reality humans are often constrained by several social groups at once. For instance an individual may be part of a social group "at a children's party" whilst

simultaneously being part of another social group "watching the clown perform". Both groups apply constraints on the behaviour of the participant. Another is that social groups do not take into account the desire of agents within the group to continue participation in the activity beyond the minimum threshold needed to continue the activity. For example a group playing a football game requires a given number of agents, however in reality the game can continue with fewer and fewer agents if the desire to play remains with the participants. A third limitation is that currently the behaviours described focus on NPC interactions between themselves rather than with the player. Being a reactive system, we expect the interaction with players to be relatively easy to incorporate, however there are significant usability issues surrounding the concept of player acceptance of social activity, specifically how to ascertain that the player intended to become a part of a social group activity and the corresponding timing of when the player is no longer participating. Finally there remains the issue of evaluation of the behaviours created using this approach. Evaluation of the actual behaviours produced seems reasonably straightforward if labour intensive, however the key aspect of player immersion along with the question of how social interactions of this type can effect player perception remains an issue. We hope to address these shortcomings with further development of the model and test implementation.

REFERENCES

[1] Quantic Dream, "Heavy Rain," 2010.

[2] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM Transactions on Graphics*, vol. 25, Jul. 2006, p. 1160.

[3] Rockstar Games, "Grand Theft Auto IV," 2008.

[4] C. Hecker, "Spy Party," 2011.

[5] N. Pelechano, J. Allbeck, and B. Norman, *Virtual Crowds: Methods, Simulation and Control*, Morgan & Claypool, 2008.

[6] M. Mateas, A. Stern, and G. Tech, "Façade : An Experiment in Building a Fully-Realized Interactive Drama," *Narrative*, vol. 2.

[7] M. Mateas and a Stern, "A behavior language for story-based believable agents," *IEEE Intelligent Systems*, vol. 17, Jul. 2002, pp. 39-47.

[8] C. Pedica and H. Vilhjalmsson, "Social Perception and Steering for Online Avatars," *Design*, pp. 104-116.

[9] E.T. Hall, "Proxemics -- The Study of Man's Spatial Relations and Boundaries," *Mans Image in Medicine and Anthropology*, International Universities Press, 1963, pp. 422-445.

[10] A.E. Scheflen, *Human Territories: how we behave in space-time*, Prentice-Hall, 1976.

[11] A. Kendon, *Conducting Interaction: Patterns of Behavior in Focused Encounters (Studies in Interactional Sociolinguistics)*, Cambridge University Press, 1990.

[12] E. Goffman, *Frame Analysis: An Essay on the Organization of Experience*, Northeastern University Press, 1974.

[13] D. Jan and D.R. Traum, "Dynamic movement and positioning of embodied agents in multiparty conversations," *Computational Linguistics*, 2007, p. 1.

[14] M. Arinbjarnar and D. Kudenko, "Schemas in Directed Emergent Drama."

[15] R. Evans, "Social Activities: Implementing Wittgenstein," *Gamasutra*, 2002.

[16] D. Isla, "Building a Better Battle: HALO 3 AI Objectives," *online*, 2008.

[17] J. Orkin, "Simple Techniques for Coordinated Behavior," *AI Wisdom 2*, S. Rabin, ed., Hingham: Charles River Media, 2003, pp. 199-206.

[18] J. Orkin, "Constraining Autonomous Character Behavior with Human Concepts," *AI Wisdom 2*, S. Rabin, ed., Hingham: Charles River Media, 2003, pp. 189-197.

[19] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, 1995, pp. 4282-4286.

[20] C.W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Computer Graphics*, vol. 21, Aug. 1987, pp. 25-34.

[21] S. Belbasi and M.E. Foulaadvand, "Simulation of traffic flow at a signalized intersection," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, 2008, p. P07021.

[22] R. Narain, A. Golas, S. Curtis, and M.C. Lin, "Aggregate dynamics for dense crowd simulation," *ACM Transactions on Graphics*, vol. 28, Dec. 2009, p. 1.

[23] B. Yersin, "Real-Time Motion Planning, Navigation, and Behavior for Large Crowds of Virtual Humans," *Work*, vol. 4401, 2009.

[24] J. van den Berg and D. Manocha, "Reciprocal Velocity Obstacles for real-time multi-agent navigation," *2008 IEEE International Conference on Robotics and Automation*, May. 2008, pp. 1928-1935.

[25] W.J. Ickinger, "Psychological characteristics and Interpersonal Distance," pp. 1-17.

[26] E. Mcdaniel, P.A. Andersen, and P.P. Journals, "International patterns of interpersonal tactile communication: A field study," *Psychology*, 1998.

[27] J.N. Bailenson, J. Blascovich, A.C. Beall, and J.M. Loomis, "Interpersonal Distance in Immersive Virtual Environments," *Society*, 1984, pp. 1-15.

[28] A. Kendon, "Movement coordination in social interaction: Some examples described."

[29] M. Sung, M. Gleicher, and S. Chenney, "Scalable behaviors for crowd simulation," *Computer Graphics Forum*, vol. 23, Sep. 2004, pp. 519-528.

[30] J. Ciger and D. Thalmann, "Planning with Smart Objects," *Virtual Reality*.

[31] K.D. Forbus and W. Wright, "Some notes on programming objects in The Sims," *Group*, 2001.

[32] W. Wright, "The Sims," 2000.

[33] D.D. Corkill, "Blackboard systems," *AI Expert*, vol. 2, 1991, pp. 40-47.

[34] S. Paris, J. Pettré, and S. Donikian, "Pedestrian Reactive Navigation for Crowd Simulation: a Predictive Approach," *Computer Graphics Forum*, vol. 26, Sep. 2007, pp. 665-674.

[35] P.N. Magnenat-thalmann, "Real-time Animation of Interactive Virtual Humans," 2006.

[36] B. Michoud, E. Guillou, H. Briceno, and S. Bouakaz, "Real-Time Marker-free Motion Capture from multiple cameras," *2007 IEEE 11th International Conference on Computer Vision*, Oct. 2007, pp. 1-7.

[37] R. Von Laban, *Mastery of Movement*, Princeton Book Company Publishers, 1971.

[38] D.M. Chi, M. Costa, N.I. Badler, D. Chi, L. Zhao, and N. Badler, "Center for Human Modeling and Simulation The EMOTE Model for Effort and Shape The EMOTE Model for Effort and Shape," *Techniques*, 2000, pp. 173-182.

[39] N. Yee, J.N. Bailenson, M. Urbanek, F. Chang, and D. Merget, "The unbearable likeness of being digital: the persistence of nonverbal social norms in online virtual environments.," *Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society*, vol. 10, 2007, pp. 115-21.

[40] L.E. Nacke, "Affective Ludology: Scientific Measurement of User Experience in Interactive Entertainment," 2009.

Influence Landscapes - From Spatial to Conceptual Representations

Luke Dicken and John Levine[1]

Abstract. In this paper we will present the concept of the ”Influence Landscape”, an extension to the current ”Influence Map” or ”Artificial Potential Field” that is commonly used in AI for Games. Influence Maps have previously only been used purely in relation to a spatial representation of the world and have been used to guide the movement of agents, avoiding dangerous areas and being drawn to rewarding areas. We will describe an extension to this principle that allows the computationally efficient process to be applied across a conceptual space, allowing the agent to perform basic reasoning in this efficient manner.

1 INTRODUCTION

1.1 Motivation

Artificial Intelligence research has previously demonstrated a robust ability to solve a diverse range of problems. We now have techniques that are capable of very sophisticated reasoning, for example, Deep Blue which is able to play Chess at a near-grandmaster level. However, in order to do this, Deep Blue required enough processing power to evaluate 200,000,000 chess positions per second, making it the 259th most powerful supercomputer of its time[11]. That is to say, AI techniques can produce very high quality results given enough time or processing power. At the other end of the spectrum we can find techniques that require much less power in order to produce decision making, such as the Brooks Subsumption Architecture [3], which is designed to be both light-weight and robust in its decision making by reacting to specific subsets of stimulus within the environment. In this way, decisions are made very rapidly but in a highly reactive manner; there is little regard given to the longer term objectives of the agent. This simplification is where such an approach gains its speed, but it also the cause of the drop in quality of solutions.

As AI researchers, this puts us in something of a dilemma. We have tools that are capable of either good decision making at the cost of speed, or fast decision making at the cost of quality. However, there are any number of applications that require both fast and good decisions and increasingly, emphasis is being placed on finding better compromises between these two. This paper will present an extension to the well proven technique known as the Influence Map or Artificial Potential Field. Our aim is to show how, by using techniques from Automated Planning, the concept can be adapted to provide a fast and powerful approach to a wider class of problem than it has previously been applied to.

[1] Strathclyde AI and Games Research Group, University of Strathclyde. Glasgow, UK. (Corresponding Author : Luke Dicken email luke@cis.strath.ac.uk)

1.2 Paper Outline

The remainder of the paper is structured as follows : In Section 2 we will discuss the basic underpinnings of the Influence Map along with a synopsis of recent work that has involved the technique, we will also describe the field of Automated Planning, and introduce the concepts that drive our extension. In Section 3 we will outline the manner in which the current approach to Influence Maps can be reformulated slightly without altering its function, and then demonstrate how using the techniques we previously introduced we can extend the concept of the Influence Map into a conceptual representation, thereby generating what we term Influence Landscapes. In Section 4 we will present two worked examples of this process and show that although the information being handled is now much richer than an Influence Map would typically represent, the fundamental concepts remain unchanged. Finally, in Section 5 we will present our conclusions and discuss future work we feel would be applicable to further develop this approach.

2 RELATED WORK

2.1 Influence Maps

The Influence Map is a mainstay of the AI Developers arsenal. In broad terms, it is a heuristic evaluation over a (typically 2 dimensional) map, giving the perceived value of every location within this plane (or space). The main idea is that different elements within the world can be assigned either a positive or negative influence and this influence can be used to guide an agent around the world, attempting to avoid the negative influences and be drawn to the positives. In order for the influence of an object to be perceived across a distance, the influence radiates from the point of interest across the remainder of the map. No particular function is specified for this radiation process, but common techniques include Gaussian distributions centered on the point, or propagation according to an adaptation of the inverse square law. Visual representations of this may resemble a heat map, with the colouring of the map being representative of the influence being exerted on the world. Figure 1 shows an example of this, with the image on the left being taken from the game Ms. Pac-Man and the right showing the threatening influence that the ghost characters are exerting on the environment.

2.2 Automated Planning

Automated Planning, by contrast, is a much more deliberative or symbolic approach to AI. The aim of AP is to allow agents to achieve complex sequences of goals without direct intervention by a human.

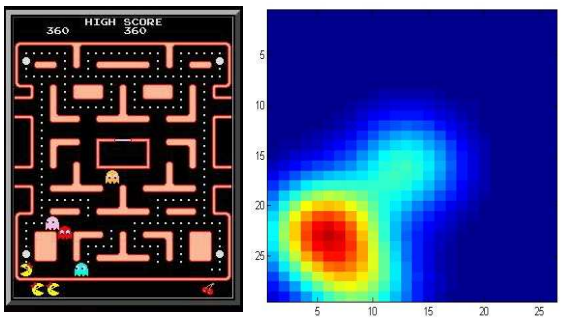


Figure 1. Screenshot of Ms. Pac-Man game and associated Influence Map

This is accomplished by defining three elements and then performing reasoning. In no particular order these elements are :

- A complete description of the manner in which the environment of the agent works, a description of the types of object within the environment and the actions that are possible in the environment, with associated conditions that must be true before an action can be applied (for example, it would not be possible to load a package onto a truck if both these objects were not at the same location) and the effects that the action will have in the world.
- A full description of the initial state of the world that the agent will be acting in.
- A partial description of the key facts that must hold in the end state, in other words, a list of the goals to be achieved.

In recent times the Planning Community has standardised on a single language with which to describe the domains that are being worked with, the Planning Domain Description Language or PDDL [12]. PDDL relies on a fact-based representation of the environment which is to say facts can be asserted to be true or false. For ease of writing, it takes a predicate calculus approach to modeling the world rather than requiring a purely propositional description be supplied, and also generally uses a hierarchical notion of typing to allow for more finegrain control over the expansion of the predicate calculus into grounded propositions. So to re-use the example from the previous paragraph, package, truck and location would be distinct types and packages and trucks might also be extensions to a physical_object type. This allows the definition of two predicates :

```
at(a physical_object, location)
in(package, truck)
```

Extensions to the basic version of PDDL have been built to allow the use of numeric values (Fluents) in the calculus, to allow for metrics to be specified to give better assessments of the quality of plans produced and to allow actions to have duration rather than be instantaneous [7]. There is also now support to allow for facts to be asserted or deasserted at specific times independent of the planning system, allowing a basic - and fully observable - model of a changing environment to be created [5]. In the currently most advanced form, known as PDDL+ [6], an wide array of features are supported, leading to a very rich modeling language, although in general, uptake of this variant within the community has so far been limited and PDDL 2.1 remains the de facto standard.

Using the descriptions provided by PDDL, planners attempt to navigate the state space of the world and find a sequence of actions that connect the initial state to one of possibly many states in which the goal propositions hold. The aim is to find the least cost - whether that cost be in terms of time or some other measure of quality - path, although often any path is deemed sufficient. In general, the task of searching this space for valid action sequences is particularly complex. The PDDL specification is capable of representing problems that belong to complexity class PSPACE-Complete (although it should be noted that the benchmark human-solvable problems tend to be restricted to NP-Hard) [9]. In order to adequately search this space, the majority of planning research focuses on search strategies and heuristic guidance. A good example of this is the planner Fast Forwards [10], which uses Enforced Hill Climbing as its search strategy (best first, failing into breadth-first when an improvement over the current node is not available) as well as the Relaxed Plan Graph heuristic, which ignores the negative effects of actions, meaning that all facts that become true remain true. Relatively informally, this gives an overly optimistic view of the interactions with the world as goals will be believed to be met before they necessarily have been due to conflicts between actions (consider a problem in which a crane is empty and must move two containers - in the first step it may pick the first up, in the second step the crane is still empty as this fact cannot be deasserted, so it may pick up the second as well). This simplification has proven powerful as it reduces the complexity of the task dramatically, and gives a much more easily computable heuristic estimate of the remaining work required to get from a given state to the goal.

3 METHODOLOGY

3.1 Abstract Model

As was previously mentioned, Influence Maps typically describe a continuous two dimensional world, and highlight the perceived value of this continuous space to the agent based on points of interest in the world. The first modification we propose is simply to move from a continuous world to a discrete world consisting of tiled, abutting locations. This is a simple enough change that does not drastically alter the concept of the Influence Map. For each tile, the value perceived for that tile is equivalent to the function evaluation at a single point contained within the tile if this were still in the continuous space, for ease of discussion we will assume that this point is chosen to be the center of the tile but this is not a requirement. Effectively all we have done is produce the same heat map style representation, but by discretisation we have greatly reduced the granularity of change that the map reflects and introduced distinct steps.

Into this model we introduce barricades, which do not take up space within the map but are impassable to agents within the world and consequently also impassable to influence propagation. In essence, although our tiles still adjoin geographically, we have created a maze of passages and barricades which will focus and direct the agents mobility in this world, as well as the perceived influence that points of interest will exert - there is little sense in an agent being threatened by an enemy geographically adjacent if the shortest path between the two traverses thirty or more other tiles. This can be visualised as a grid of tiles, with all tiles connected to their neighbors to the North, South, East and West. Where a barricade exists, there is no connection as shown in Figure 2.

3.2 Implementation

We previously explained PDDL as the standard language in which planning problems were described. Due to its propositional nature,

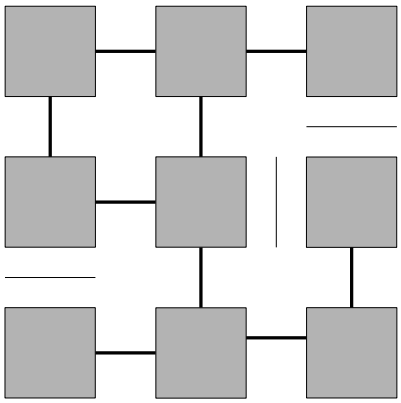


Figure 2. Example of disconnected tiles highlighting the presence of the "barricades".

this gives a representation that can be thought of as having a very high dimensionality, with each dimension only capable of representing a true or false value. This is a highly complex representation that does not give an easy method to see how the dimensions relate to each directly (except by inspection of the labels placed on the propositions, which is to an extent cheating). Importantly, this does not give a representation particularly conducive to exploration of a space, but highly suited to parameter tuning.

SAS+ [2] is an alternative formalism to PDDL and rather than capturing the state of the world as a series of true or false statements, it utilises a collection of multivalued variables to represent the current state. Again to refer to the example of packages and trucks, we noted that in PDDL this would give rise to propositions of the form at(Package1, Location1) or in(Package1, Truck1) which could be true or false. In SAS+, these would be condensed into a single variable reflecting the current position of Package1 :

$$\texttt{location\_package1} \in \{\texttt{Location1, Truck1, ...}\}$$

This generates a representation with a much lower dimensionality, but with more values than the true or false options of the propositional format. Although this is not an ideal formulation for typical styles of search, it generates a nice representation that it is much more accessible and human-readable. It is also worth noting that thanks to work on the planner Downwards by Helmert, automatic reformulation from PDDL to SAS+ is possible [8], meaning that the two can be used interchangeably without requiring any intervention or domain analysis by hand. This is done principally by analysis of mutually exclusive propositions in the PDDL - although there is no explicit link between propositions such as at(Package1, Location1) and at(Package1, Location2) it is intuitive that these two facts can never both be true simultaneously and analysis of the actions within the domain can reveal this kind of relation and allow propositions to be grouped and translated into the multi-valued variables of SAS+.

Furthermore, the same techniques will also discover the manner in which these variables alter their values by analysis of how the potential actions within a domain can affect the state of the variables. Thus it might become apparent that in order to move a package between two locations, it must first be loaded onto a truck. This allows the SAS+ formalism to not only define the values that the variable can take, but also a sense of the ordering of these values and their mutual adjacency. It is common to represent this as a Domain Transition Graph (DTG), which essentially lays out the values of the variable as nodes within a graph, and actions that allow transitions between two value pairs defining the edges between nodes. This gives rise to the structure shown in Figure 3 which formalises the recurrent example.

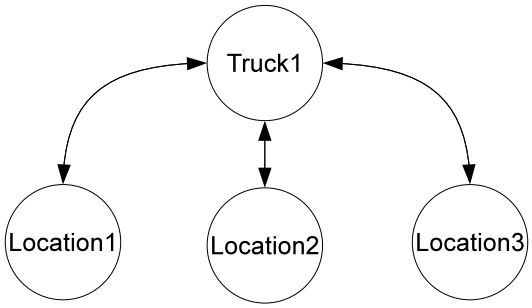


Figure 3. Example of a Domain Transition Graph. Packages may move between locations by first being loaded into the truck

As was mentioned in the description of the definition, actions have preconditions that must be satisfied before the action becomes applicable and in SAS+ this is captured by the Causal Graph (CG) and Causal Links (CL). Again, this information can be generated automatically without any inspection based on an existing PDDL definition, or it can be defined directly in SAS+ as part of a model being created without PDDL. The CG captures the dependencies between the types of objects found in the environment, for the example we can see that the actions we can take to manipulate the location of the package are dependent on both the state of the truck and the location which the package is currently at.

Causal Links are the actual embodiment of the dependencies highlighted by the CG, and more or less enforce that a specific set of SAS+ variables must take specific values before an action can be taken, or put another way, the current state of the world must be set to specific nodes within certain DTGs before an edge between two values can be traversed. This means that the entire state representation of the world can be thought of as a cross-section through each DTG, and the actions that can be taken at this particular time point are exactly those edges for which the CLs are satisfied.

This presents us with a model very similar to that which we described previously, namely the disconnected discrete Influence Map shown in Figure 2, but rather than dealing with a representation of spatial coordinates, the DTG/CG space provides the same basic structure but represents practically any set of discrete concepts.

3.3 Influence Landscapes

We will now show how this structure can be used to form what we term "Influence Landscapes". In the same manner as with IMs, we designate points of interest within a DTG, and these are the origin of the influence that the agent perceives. As with IMs, influence may be positive or negative. As with IMs, we do not formally define the manner in which influence can be propagated across the graph, however we will outline a basic technique below in order to demonstrate the process as a whole.

3.3.1 Influence Propagation

In a continuous coordinate system, influence propagation is relatively straightforward, since influence is typically exerted as a mathematical function. To evaluate the level of influence at a given point, one

simply supplies the coordinates to the function and receives a resultant value representing the level of perceived influence at that point. In the Influence Landscape, this is not possible as no such coordinate system exists. Instead we rely entirely on the abstract concept of edge-distance [4] to propagate influence across the graphs. We have used a simple reward sharing algorithm to replace the function definition. In the initial stage of the algorithm, influence is assigned to the points of interest in the world and these nodes are added to a list. Then, while the list is not empty the following algorithm is executed - a node is chosen from the list, one is subtracted from its influence value and this is then divided by the number of predecessors this node has in the graph, and the resultant value is assigned to these nodes if it is higher than their current influence value. Any nodes that have had their value updated in this manner are then added to the list. This stage is repeated until the list is empty, at which point propagation has converged and the influence landscape has been generated. This landscape can now be used to guide the behaviour of the agent.

3.3.2 Traversing the Influence Landscape

It is important to note the manner in which we anticipate the Influence Landscape technique to be used. Again, we do not define a particular algorithm as being a part of the IL technique, but include the information here for completeness. We have experimented with three different approaches. Strict Hill Climbing was the first and most obvious, in which the agent will move to the node which is most attractive, or stay put if the current node is perceived to have higher value than any of the alternatives. We found that in certain domains this produced traps at local maxima that were not intended, so we next considered a hill climbing algorithm which did not take into account the value of the current node, essentially mandating that the agent always take some action. Finally, we found that in a class of domains, looking at solely the next action was not appropriate, and we moved on to consider a neighborhood-bounded implementation of A* search which was capable of finding the most promising point within the nearby area of the graph. This was appropriate in cases where the agent was forced to take an undesirable choice in the short-term to reach a more desirable node in the longer-term. Of course, the addition of a look-ahead search, even one as straightforward as a limited A* would add to the processing required to select an action, but again this serves to highlight the flexibility of the IL approach, as it can be adapted to suit the needs of most domains.

Regardless of how the next action is selected, this action will now be used to traverse the graph. In order for that to be possible, the CLs attached to that edge in the graph must be satisfied. Our lightweight approach to this is simply to solve any CLs that are currently violated and then traverse the edge. This is by no means fool-proof and a more detailed discussion of the flaws with this approach is included in a later section of this paper. To solve the CL, the agent transfers its focus to the DTG that is currently not set to the correct value and uses the IL method there with the goal now being to achieve the correct value. When the CL is resolved, the agent can transfer focus back to the original DTG and check for further violations. As the edges are traversed, both the internal state of the agent is updated, and the relevant affector is activated to cause the action to be triggered in the world.

In this way, the agent is able to use the notion of influence to navigate through a space of abstract concepts, as well as spatial locations. Preconditions of actions can be highlighted and met before an action is taken, and influence allows the agent to avoid dangers in its environment and find ways to reach the desirable locations.

4 EXAMPLES

We will now demonstrate the power of the Influence Landscape technique by introducing two example domains that are both more complex than the previous explanatory example, and more oriented towards the area of AI in Games research. The first domain we term the Tank Domain and is inspired by work by Avery, Louis and Avery [1] as well as work by Thompson, Levine and Hayes [13]. In this domain, tanks move around a battlefield with the agent under AI control being one such tank. The agent can decide which direction to move and whether to be in a ”firing” state, a ”neutral” state or a ”shielded” state. The Flight Simulator Domain involves the agent attempting to land an aircraft. The agent is able to control its direction of movement, and whether the following are deployed - airbrake, flaps, landing gear. This second example domain resembles the first, but highlights the extensibility of the technique to handle a full three dimensional representation of the world (albeit still using discrete intervals) and has multiple state variables to control.

4.1 Tank Domain

The Tank Domain is a fairly straightforward example of a scenario in which IMs are quite applicable for moving the agent’s tank around whilst avoiding the enemy tanks. However, we have added the concept of state to the scenario as mentioned above. This takes the scenario out of the realm of the Influence Map, as this type of reasoning cannot be handled. However, it is now very suited to being dealt with by the Influence Landscape approach. We can see that locations become a grid representing the battlefield, and for the sake of diagrammatical simplicity we will again constrain the possible movement directions to North, South, East and West. An open battlefield is relatively uninteresting, so we will add some obstructions to the world. An example of such a battlefield is shown in Figure 4.

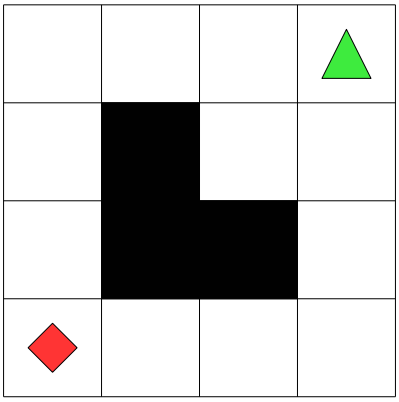


Figure 4. Example of a Tank Domain battlefield. The triangle is the agent, the diamond is an enemy tank.

Representing this in the DTG form that the Influence Landscape requires is relatively straightforward and is shown in Figure 5. We have labelled the nodes representing locations within the world by their grid coordinates, and the state variable notation is as follows: ”Sh.” for the shielded state, ”Ne.” for the neutral state, ”Fi.” for the firing state.

For the purposes of this example, we will make the goal of the example to reach the bottom left corner, and we will leave the enemy static occupying the goal location. It is simple enough to see that movement alone will not solve this problem, and in the neutral

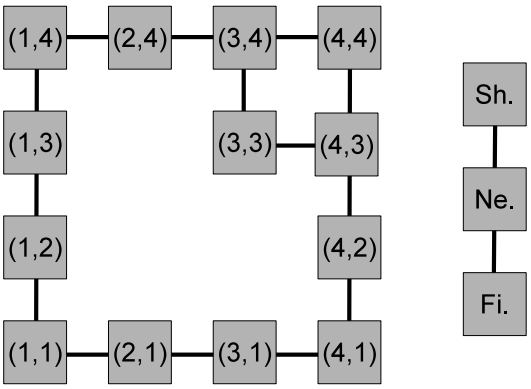


Figure 5. Example of a Tank Domain DTG space. On the left is the DTG representing locations, on the right the DTG represents the current state of the agent's tank.

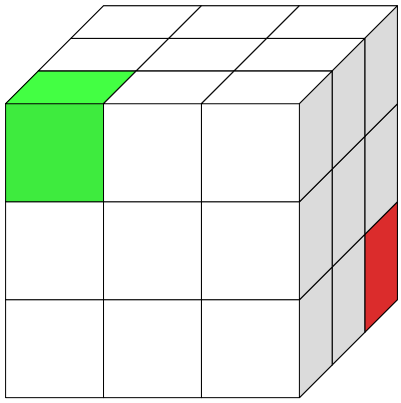


Figure 6. Visualising the example Flight Simulator problem.

state the agent will experience a negative influence radiating from the enemy and not be able to reach its goal. Equally it is easy to see that although going into the defensive state would be a viable solution, this will not allow the agent to reach the final tile and so is not going to allow the agent to achieve its goal. Therefore, the only way to complete the task assigned is to switch into the firing state and destroy the enemy, thereby clearing the path for the agent to reach the objective safely.

4.2 Flight Simulator Domain

The Flight Simulator Domain deals with the problem of landing a plane at an airport. This requires a three dimensional representation of the environment since height is obviously a major factor in this task. For the sake of simplicity we have abstracted several key aspects out of the model in order to maintain simplicity, most important among these are the orientation of the plane and its airspeed - in a real-world situation these are of course essential elements. We assume that a plane is capable of moving North, South, East and West as well as up and down. During the landing process it will be necessary at certain points to deploy the planes flaps, to lower the landing gear and to deploy the airbrake. These will be triggered at different points, and each will have different tolerances on the earliest point at which it may be triggered as well as a hard deadline on the latest point it must be triggered by. As you can see, this domain presents a much more complex and reasoning-centric problem, which can again be represented as an Influence Landscape. For simplicity we will consider a very small representation of a space of 3x3x3 possible locations for the plane to be in. The plane starts out in a corner with altitude, so would have position (1,1,3) and must reach the ground level in the opposing corner or (3,3,1). This scenario is visualised in Figure 6, with the planes initial position marked in green, and the target landing site marked in red.

Represented as a DTG, this generates the set of graphs shown in Figure 7. Again the initial point and the goal point have been marked in green and red respectively. The left hand side again represents the graph of the location variable for the agent, with each 3x3 grid being a cross-section layer for a given altitude. The three graphs representing the state of the properties of the plane are labelled on the right hand side of the diagram.

As can be seen, it is relatively trivial given this formulation to adopt the Influence Landscape technique and create a system which can not only be guided to its goal in terms of location, but also ensure

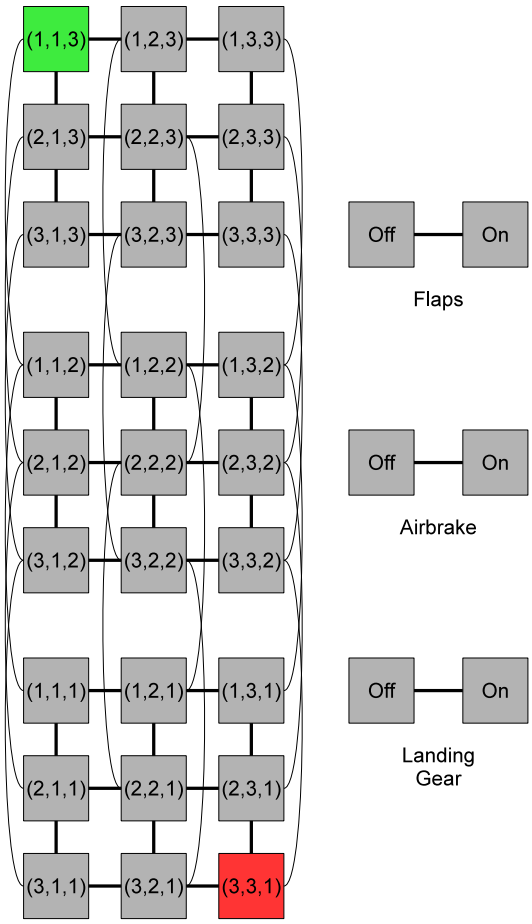


Figure 7. Visualising the DTGs generated from the Flight Simulator Domain problem

that the correct settings within the plane are activated at the correct time.

5 CONCLUSIONS AND FUTURE WORK

In this paper we have presented a technique that can be used to adapt "Influence Maps" from being a technique that functions purely across a spatial representation of the world and indicates good and bad areas of this world, to being a technique capable of providing the same level insight at a conceptual level. We do not propose that this is a complete substitution for the need to perform deliberative reasoning about the world, however it does offer an efficient alternative in situations where the overheads involved in full reasoning outweigh the complexity of the world being reasoned about.

Work with this technique is ongoing, and it currently forms the basis for a complete agent architecture being developed by the authors aimed at providing a better unification between reactive and deliberative AI techniques in a single architecture. As such, several open questions remain, most importantly with the manner in which Causal Links can be resolved. As laid out above we take a very basic approach at this juncture of satisfying the demands of the edge we wish to traverse and then traversing it - this is provably not a sound approach when domains become more complex. The Planning domain Driverlog resembles the early example we used but includes the notions of drivers for the trucks - in order for a truck to move between locations it must have a driver onboard. It is relatively easy to see that this quickly leads to an issue using naive CL resolution: in order to move a truck we need a driver onboard, in order for the driver to board the truck it must be at the same location as the driver, but in order to move the truck to the same location as the driver it needs a driver and so forth. For more complex domains in which reasoning needs to be able to identify cyclical dependencies and avoid them, this approach is not adequate, and part our ongoing work is in developing a more refined technique that can inform this process without relying on a fully deliberative process.

With that said, it is our contention that as an extension to the current Influence Map technique, our work is currently mature enough to be suitable for adoption in a number of game-related domains. This is particularly the case for those domains that were already somewhat suited to the IM approach as these tend to not require the complex types of reasoning that a more robust CL resolution system would provide, but in many cases would benefit from the ability to do basic reasoning. We believe that the Influence Landscape technique has the potential to serve as a useful tool for the Game AI developer.

REFERENCES

[1] P Avery, S Louis, and B Avery, 'Evolving Coordinated Spatial Tactics for Autonomous Entities using Influence Maps', *IEEE Symposium on Computational Intelligence and Games*, (2009).

[2] C Bäckström and B Nebel, 'Complexity results for SAS+ planning', *Computational Intelligence*, (1995).

[3] R Brooks, 'A Robust Layered Control System for a Mobile Robot', *IEEE journal of robotics and automation*, (1986).

[4] E Dijkstra, 'A Note on Two Problems in Connexion with Graphs', *Numerichse Mathematik*, 279–271, (1959).

[5] S Edelkamp and J Hoffman, 'PDDL 2.2: the language for the classical part of IPC-04', *Proceedings of the International Conference on Automated Planning and Scheduling*, (2004).

[6] M Fox and D Long, 'PDDL+ level 5: An extension to PDDL2. 1 for modelling planning domains with continuous time-dependent effects', *Technical Report, University of Durham*, (2001).

[7] M Fox and D Long, 'PDDL2. 1: An extension to PDDL for expressing temporal planning domains', *Journal of Artificial Intelligence Research*, (2003).

[8] M Helmert, 'A Planning Heuristic Based on Causal Graph Analysis', *Proceedings of the Fourteenth International Conference on*, (2004).

[9] M Helmert, *Understanding Planning Tasks : Domain Complexity and Heuristic Decomposition*, Springer, 2008.

[10] J Hoffmann and B Nebel, 'The FF Planning System: Fast plan Generation Through Heuristic Search', *Journal of Artificial Intelligence Research*, (2001).

[11] F Hsu, *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*, Princeton University Press, 2002.

[12] D McDermott, M Ghallab, A Howe, C Knoblock, A Ram, M Veloso, D Weld, and D Wilkins, 'PDDL The Planning Domain Definition Language', *Yale Center for Computational Vision and Control Technical Report*, (1998).

[13] T Thompson, J Levine, and G Hayes, 'EvoTanks: Co-Evolutionary Development of Game-Playing Agents', *IEEE Symposium on Computational Intelligence and Games*, (2007).

SPREE : The Strathclyde Poker Research Environment

Luke Dicken, Nicky Johnstone, John Levine and Phil Rodgers[1]

Abstract. This paper presents the Strathclyde Poker Research Environment (SPREE) which has been developed as a tool to aid research into the game of Poker, both by enhancing our understanding of the way human players play the game and providing a standard environment for autonomous agents to play. We will justify the need for this tool, outline the principal components and demonstrate potential uses for the system.

1 INTRODUCTION

1.1 Motivation

When designing and developing stronger Artificial Intelligence (AI) algorithms, researchers often look to classical multiplayer games as being microcosmic examples of the decision making process. Although games such as Go, Chess or Poker have simple rules, meaning that they can be described to an AI system relatively easily, they still remain complex challenges[7][4]. Of these, Poker is arguably one of the more interesting to AI as it a game with incomplete information; whereas with games such as Go and Chess all of the pieces are visible, and you can therefore make informed assumptions about your opponents' future moves, in Poker you have no knowledge of the cards your opponent is holding and must try to make inferences based on previous observations of that opponent which influence your decisions.

Poker research typically focuses on the "Texas Hold 'Em" variant, in which each player is dealt two cards which are hidden to other players, and subsequently a round of betting takes place based on the perceived strength of these "hole" cards. Three cards are then dealt face up and those players who still remain in the pot must now bet based on the strength of their two cards combined with the three on the table to make a five card Poker hand. When this is complete, a further card is dealt face up, followed by a round of betting, now based on the best five card hand each player can make from the six cards available to them. A final card is then dealt and a round of betting concludes the game, with those players still in the game revealing their hands and the player with the highest ranked hand taking the entire amount wagered by all players. This variant of Poker itself comes in one of two sub-variants, either "Limit" or "No-Limit". In both cases, players always have the option to "fold", forfeiting any wager made so far, and players may "call", or meet the amount currently being wagered by other players. They may also "raise", increasing the amount being bet. In Limit this raise is by a fixed amount, in No-Limit it is of a variable amount. Both are valuable mechanisms as Limit allows the researcher to deal with a more tractable problem, as only three actions are available at each decision point, whilst No-Limit allows a player to telegraph a lot more information about the perceived value of their hand which an AI system could use to influence its own decisions.

Many approaches to creating an automatic player (or "agent") for Poker have been based around the field of machine learning [3][6], using data about previous games to infer the likely strength of opponents, as well as decision theoretic techniques based on the observable components of the game. However, there is a common weakness to these systems in that both the quantity and quality of training data available from which to learn is not sufficient. Typically this data is obtained from players at online casinos, who observe and record the actions of their opponents using specialist software such as Poker Tracker[2], and subsequently trading or selling this data within the community. A sample training set, such as that used in a recent piece of work exploring the application of Monte Carlo tree search techniques to Poker play [5], consists of around a million games of Poker that have been observed. However, because this data has been collected by players at an online casino, only the information available to that player is recorded and able to be used by an AI system based on the data. As a result of this, the amount of information available - particularly about certain hand configurations - is quite limited. For instance, consider a player who is dealt a very bad hand initially. That player will probably fold immediately, and his hand will never be shown to the other players, so there is never an explicit connection (except in those cases where the player making the decision is the one recording the data) to link a poor initial hand to the fold action, it must instead be inferred from the frequency with which poor initial hands are seen at a showdown. Trying to rectify this large a priori bias in the dataset is the motivation for requiring such a large number of example games, but it does not address the fundamental issue that the partially observable nature of the game introduces.

Additionally, when such Poker agents are created by researchers they typically play Poker internal to themselves against either other agents created by the researchers (perhaps based on the work of others) or against a hypothetical model of a "human" player based on the data, rather than an actual human. The results of this play are then compared against other agents to look for improvements in performance according to some metric. The assumption underlying this kind of analysis is that for a large enough sample of games, two agents will experience an equivalent set of circumstances and thus be comparable. However, if you assume a full ten player game of Poker, there are 20 hole cards to be dealt and 5 community cards, and the number of possible permutations for dealing 25 cards from a deck is $7.41 \text{x} 10^{39}$, and this is before you consider the differences introduced by the behaviour of each agent's opponents when faced with the same situation. As such, it is effectively impossible to offer a meaningful comparison between two agents on the basis of independent experimentation with contrasted results, as the agents will not have experienced identical circumstances.

The Strathclyde Poker Research Environment (SPREE) aims to

[1] Strathclyde AI and Games Research Group, University of Strathclyde. Glasgow, UK. (Corresponding Author : Luke Dicken email luke@cis.strath.ac.uk)

overcome both of these obstacles to Poker research by providing a casino-like environment from which complete data can be extracted, enabling researchers to get a much more accurate picture of the decisions players make and the situations that lead to these decisions. In addition, SPREE provides an experimental platform on which analysis can be performed to properly compare different agent implementations by offering proper experimental practices to be followed, such as replicating all conditions including card order and opponent play-style between tests of two separate agents. Developing agents is a relatively simple task, with a complete framework for this being available in the Java language, and the communications protocol to the SPREE server being available for development in other languages.

1.2 Paper Outline

The remainder of this paper is structured as follows. In Section 2, we will give an outline of the principal components of the SPREE system, namely the server implementation, the current GUI client for human players, the administration interface and the framework for developing agents. In Section 3 we will discuss the uses of SPREE with specific reference to two use cases, one in which SPREE is used to gather information on human Poker playing patterns and another in which SPREE is used as a tool to comparatively evaluate two different Poker agents. Finally, in Section 4 we will summarise the contribution that SPREE makes to the field.

2 SPREE OVERVIEW

The SPREE system is broken into a number of distinct components, each of which will be outlined below. An overview of the interactions of each component is presented in Fig. 1. All components interact by TCP/IP except where otherwise noted.

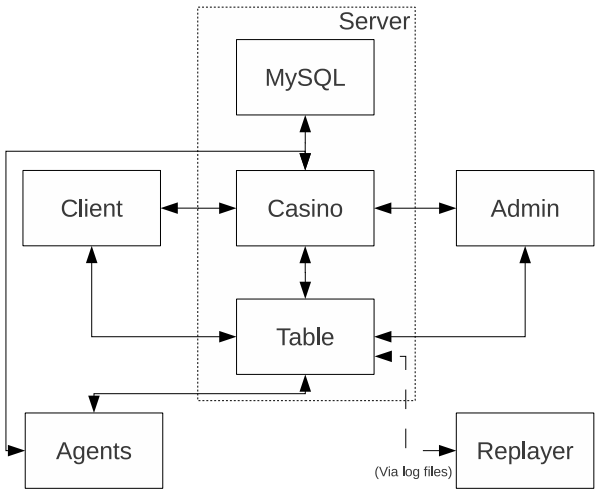


Figure 1. Representation of the interactions of the different components of the SPREE system.

2.1 Server

The core of SPREE is the server, which is used to control the flow of the poker game, including dealing cards and handling the betting options available to each player as play progresses. At its heart, the SPREE server relies on basic technologies, it is implemented in Java and utilises standard TCP/IP communications for negotiating with players. It also requires access to a MySQL server which holds configuration settings and details relating to user accounts.

2.1.1 Casino

The casino component of the server is responsible for controlling what games are on offer and logging players in. This is the component that users initially connect to in order both to authenticate themselves (or register as a new player) and to retrieve a list of Poker tables currently active on the server, the configurations of these tables and their location. This information is stored in a MySQL database (meaning that it is persistent between server restarts). The casino is also responsible for managing a player's "bankroll", or the amount of virtual money they have associated with their account. This amount can be altered by administrators of the SPREE server (more on this below). When a player connects to the casino their current bankroll is passed to the client, when they sit at a table they must choose an amount to take to the table which is deducted from their bankroll. They may then gamble with this amount, and when they leave the table the database is updated, adding the amount they have remaining from this back into their total bankroll.

From a user's perspective, the principal use of the casino is to retrieve the list of available tables. The user can then use this list to connect to the table and communicate with it directly.

2.1.2 Table

Each table runs as a self-contained process initiated by the casino. The table component runs a single table of Poker with a specific rules configuration. The basic game rules are set out using an XML-based game definition, in a format similar to that used by the AAAI Annual Computer Poker Competition[1]. The current version of SPREE can only handle the Texas Hold 'Em variant of the game due primarily to limitations imposed by the client, but by using a formalism such as this for the server, SPREE is quite extensible to other variants of Poker. The game definition used for SPREE's Limit version of Texas Hold 'Em is shown below.

```
<GameDef>
    <Description>Texas Hold'Em</Description>
    <Rounds>4</Rounds>
    <MinPlayers>2</MinPlayers>
    <MaxPlayers>10</MaxPlayers>
    <MinBet>2</MinBet>
    <SmallBlind>1</SmallBlind>
    <BlindStructure>1|2</BlindStructure>
    <PrivateCards>2|0|0|0</PrivateCards>
    <PublicCards>0|3|1|1</PublicCards>
    <BetsPerRound>3|4|4|4</BetsPerRound>
    <BetStructure>1|1|2|2</BetStructure>
</Gamedef>
```

In this example, the Description element gives a text description of the game, which is used by the server to advertise what type of game this is. The Rounds element indicates how many rounds of both dealing and betting occur in the game. MinPlayers and MaxPlayers respectively define the minimum players required to make a game and maximum. MinBet defines the base bet amount for the table. The SmallBlind is traditionally set at half of this bet amount.

The BlindStructure element reflects the positions at the table and the amounts that they must pay prior to the game begining, relative to the SmallBlind amount, so in this case there are two blinds required, with the first being equivalent to one SmallBlind and the other being equivalent to two SmallBlinds. The manner in which cards are dealt to players is determined by the PrivateCards element, which in the example shows that two cards are dealt to each player in the initial round and none for the remaining rounds. PublicCards reflects the manner in which community cards are dealt, and for Texas Hold 'Em this follows the pattern of none in the first round, three in the second and then one each for the third and fourth rounds. BetsPerRound sets a cap on the number of times a bet or raise is allowed, in Limit play this is typically capped at three raises per round, and as the initial bet would not be counted and the blinds are not counted, this is achieved by specifying a three bet cap in the first round and four in the others. BetStructure allows for a common mechanic in which the minimum bet amount increases in the later rounds of the game, this usually doubles, and as the example shows, the way of signalling this in the game definition is by stipulating the bet amount in each round with reference to the MinBet value. In rounds one and two this is equivalent to 1*MinBet, increasing in rounds three and four to 2*MinBet.

When the table is started it does nothing but wait for connections. A game does not begin until two players are seated at the table. At this point, the server begins to deal the cards to the players according to the game definition. It maintains the current state of the hand and contacts each client using a TCP/IP message called a `gameMessage`. There are three possible types of `gameMessage`, a `stateMessage` which details the current state of the game, an `optionsMessage` that offers the client the opportunity to choose one of the available options and an `actionMessage` that is used as a partial update to indicate the action taken by a specific player. This means that at the begining of each round of betting a `stateMessage` is generated which contains the state of the game after the server has dealt cards and otherwise processed the necessary steps between rounds of betting. The `stateMessage` contains information about each player, but replaces cards that the client being contacted is not able to view (such as other players' hole cards) with 'x' to show that information is being hidden. Note that below we will describe an administrative view of this data which is based on this same `stateMessage` data without such obfuscation.

2.1.3 Exported Data

The data that the table records is stored in two formats. The first is a "Recording", which contains the `stateMessage` transmissions that the server has generated during the game. These are used by the replayer component (described below) to step through the game move by move and visualise exactly what happened during that specific game.

The second style of exported data which the server generates is a "Hand History" file, which is a descriptive account of what has occurred during the game. This style of data is used by existing tools such as Poker Tracker[2]. The aim of specifically exporting this data is to allow games played within SPREE to be analysed in the same manner as other games that have been played on other systems, and to maximise interoperability between SPREE and other tools in use both by players and by researchers.

Additionally, note that this export system is designed to be highly modular, and implementations that output the data in a different format can be easily developed if it is found necessary to extract specific data that cannot be parsed from the current structure.

2.2 Client

Our client implementation allows a user to connect to a SPREE Casino and either authenticate to an existing player account or create a new one. From there, a listing of the active tables is retrieved and displayed to the user. An example of this is shown in the lower portion of Fig. 2, where it can be seen that the casino has only one table currently active. Details about the tables are made available to the user to aid them in choosing an appropriate place to play, in much the same way as would be experienced in a real online casino. Specifically, the client displays the name of the table, the particular variant of Poker being played, whether the table is playing Limit or No-Limit, the size of the initial amount wagered, the current number of players at the table, the minimum and maximum players allowed and the minimum and maximum buy-in amounts allowed. The user then has the option to "sit" or "watch" the table. Sitting means that the user intends to play at this table, whilst watching is simply observing the game currently in progress without taking an active part in the game. Additionally, users may request more information about the table.

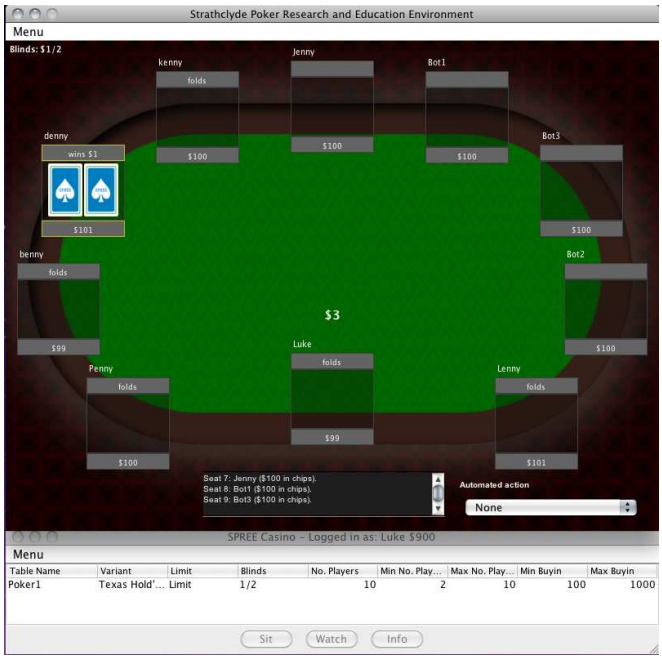


Figure 2. Screenshot of SPREE Client. Luke is playing against 9 opponents, Denny won the previous hand.

When a table is selected by the player, the table view portion of the GUI opens, as shown in the upper portion of Fig. 2. This is influenced heavily by the look and feel of a variety of graphical Poker systems, and is relatively representative of a traditional table game of Poker. Fig. 3 shows a game in progress, in which the community cards can be see. Also in this image, the options available to the player are clearly visible. Lenny has the option to fold check or bet. It is also possible to use the GUI to "queue up" an action in advance, meaning that they can make decisions when they first see their cards and decide what they will do (such as fold) before it is their turn. This is a common feature of many casino interfaces as it allows the

player to focus their attention elsewhere rather than forcing them to wait patiently for their turn.

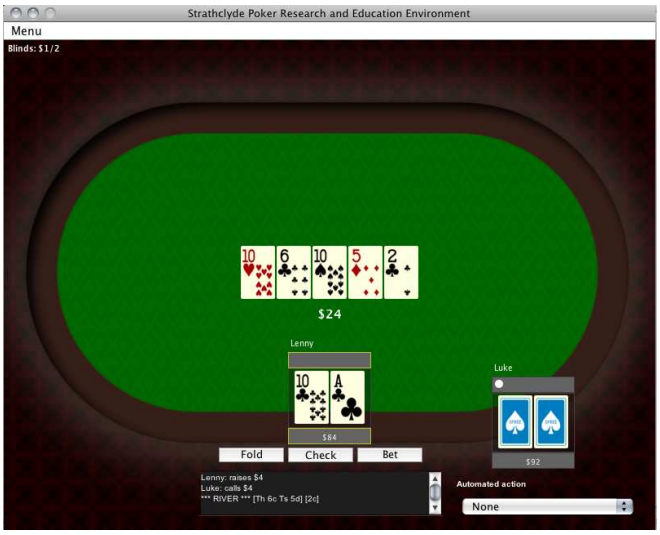


Figure 3. Screenshot of a Poker game in progress. Lenny is playing against Luke. The game has reached its final phase, with Lenny to bet next.

In all respects, the design of the client tried to closely emulate the manner in which online casinos present the game to players, on the assumption that since these casinos succeed in getting players to play in their free time, the features that they offer to players must be of use. Additionally, it is hoped that those already familiar with playing in a commercial environment will find the SPREE implementation roughly equivalent to the experience they are used to.

2.3 Administration Interface

The SPREE administration interface is designed to allow users of the SPREE system who are flagged as administrators to control and configure an active SPREE server. This gives them the opportunity to alter the set of tables currently used by the server by deleting existing ones or adding new ones, and to specify the parameters and game type of these tables as they are added. Administrators are also able to edit user accounts, both by changing usernames and passwords as well as altering the bankroll of a user. It is possible for administrators to make other users into administrators or remove a user's administrator status. It is possible to watch a game in progress at a table from within the administration interface, which provides a real-time perfect information account of the table, including each player's hole cards. This is obviously open to abuse by players who are also administrators, therefore we have enforced that a player who is playing at a table cannot also view that table using the administration interface.

As a separate part of the administration tools, we have also created a "replayer" system which is able to parse the generated hand history files and play back the game step by step, allowing for a view similar to that seen in the administration interface itself, but after the fact rather than in real time. This allows the administrators to inspect the actions that occur within a game, either to look for irregularities in play or perhaps to analyse an agent's behaviour under certain circumstances more closely. This tool operates offline based entirely from the file generated, with no requirement of access to a SPREE server.

2.4 Poker Agent Framework

The SPREE Agent Framework is derived from the codebase of the client component, with the interactive components removed. Like the client, the Agent Framework is built to maintain an internal model of the current Poker game in progress, which is kept current by the receipt of update messages from the server.

When a decision is required of an agent, the `makeMove` routine of the framework is triggered and is passed the possible options that the agent can choose between. This allows the agent developer to insert whatever algorithm they choose into their implementation and send their choice back to the server through the `PokerTableController` library provided. This provides a complete solution for researchers to create Poker playing agents and deploy them by removing any necessity for development on components not directly related to the AI research.

3 POSSIBLE APPLICATIONS

3.1 Data Gathering

As the SPREE system allows researchers to gather complete information about games of Poker that have been played, one of the primary uses of the system is to expand the amount of data available for machine learning to take advantage of, and to do so using these complete hand histories rather than the partial information variants. In order to achieve this, human players need to play Poker within the SPREE environment. Despite the information being complete, a large amount of data will still need to be collected to be of use to the kind of AI techniques that could benefit from it.

The exact method by which this data will be gathered will vary from project to project, however it is important to note when generating such data by playing games involving humans that the perceived value of the virtual money being wagered must be taken into consideration. It is a common phenomenon that when wagering something that has no intrinsic worth to the player (referred to as "play money"), they will typically behave in a quite different manner to players who are wagering something of minor value, such a "low stakes" game, who play as different again from players wagering something they consider to be of significant value, such as in a "high stakes game". Because of this, it may be necessary to somehow offer incentives during the data gathering such that the players involved take it seriously. In any event, when using the SPREE system for data gathering, this issue must be considered as it could potentially have a significant impact on the worth of the data. In the case of machine learning, not accounting for this variance in play style could lead to an agent learning specifically how to beat play money humans, which would be of limited use in a wider context. It is also worth observing that the SPREE environment would be an ideal tool in order to analyse and attempt to quantify exactly what effect these differing stakes have on players' behaviour.

3.2 Agent Evaluation

Because the SPREE system allows for specific scenarios to be established, it is a very good test bed for comparing two different Poker playing agents. Typically, the strength of a Poker player is rated as "Big Bets per 100 Hands", which analyses the player's winnings (in terms of the minimum stake at the table) over time. A standard method of comparison is to use this (or some other "bottom line" metric) on the assumption that over time, two agents will have encountered a similar range of circumstances. As was noted earlier,

there are $7.41\text{x}10^{39}$ possible ways of dealing out the cards required for a game of Poker. While a number of these configurations are functionally equivalent (as suits are not ranked in Poker) this still makes for an incredibly large number of circumstances to test agents in, in order for them to have seen a comparable set of aggregate circumstances, and therefore making it difficult to minimise the impact of these differing circumstances on the agents' performance.

SPREE allows researchers to take a different tack by allowing for the dealt cards to be established in advance. This allows specific scenarios to be engineered, meaning that the agents can be tested under specific conditions, but it also means that the cards can be dealt identically for two different agents, so they can each face the same opponents, holding the same cards allowing for a true comparison to be made between the agents, rather than a statistical analysis that attempts to factor out the inherent variance in the results. This feature may also be of use when developing agents, as it allows for a specific scenario in which the agent performs poorly to be identified, and then those cases to be specifically retested as the agent is refined, allowing for a much more targeted analysis of whether these changes are improving the performance of the agent.

4 DISCUSSION AND SUMMARY

At this time we feel that SPREE is quite a mature project, however it is far from complete and there are a range of features that would be of benefit if they were incorporated. Currently, all games that are handled are of a "cash" variety, in which players sit, play for virtual money and may at any time leave the game, retaining the virtual money they had remaining at that point in the game. This is different from the "tournament" style of play, in which a player wagers their money to enter the tournament, receives a number of chips as part of this "buy in" process and then plays until eliminated with no opportunity to leave without forfeiting the game and their wager. It is currently possible to play pseudo-tournaments by placing meta-rules on the players external to the environment itself to achieve the same effect, but this would be significantly more elegant if it was possible within SPREE, and would open up further avenues for analysis since a player's betting style in this format differs significantly to cash games due to the emphasis being more on eliminating players from the tournament rather than ensuring a steady return on wagers.

One of the most obvious features that is currently not implemented within SPREE is a chat system to allow players to communicate with each other. Although this is very common on commercial poker systems, it has deliberately been omitted from the early work developing SPREE as we felt that the presence of a chat system was likely to result in an environment in which detecting agents was trivial as these would be the players not communicating. We wanted the emphasis in designing agents to remain firmly on the algorithmic side, rather than allow the introduction of a Turing Test via a chat system. This may be introduced at a later date, but would probably be optional on a table by table basis to allow agent-based experiments to be conducted, for example one in which humans attempt to detect bots based purely on play style.

Poker research is a very active topic, with many questions outstanding in a variety of areas from developing stronger AI routines to creating agents that can pass themselves off as human players. We have developed SPREE as a tool for our own use to aid us in our work towards answering these questions, and in this paper we introduce it to the wider community in the hopes that will be of similar use to others.

REFERENCES

[1] 'AAAI Annual Computer Poker Competition', *http://www.computerpokercompetition.org/*.

[2] 'Poker Tracker', *http://www.pokertracker.com/*.

[3] Darse Billings, *Algorithms and Assessment in Computer Poker*, Ph.D. dissertation, University of Alberta, 2006.

[4] B Bouzy, 'Computer Go: An AI oriented survey', *Artificial Intelligence*, **132**(1), 39–103, (October 2001).

[5] Guy Van Den Broeck, Kurt Driessens, and Jan Ramon, 'Monte-Carlo Tree Search in Poker using Expected Reward Distributions', *ACML*, 1–15, (2009).

[6] Richard G Carter, 'An Investigation into Tournament Poker Strategy using Evolutionary Algorithms', *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games*, (2007).

[7] Fenghsiung Hsu, *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*, Princeton University Press, 2002.

Games based learning for Exploring Cultural Conflict

Lynne Hall[1], Syaheerah Lutfi[2], Asad Nazir[3], John Hodgson[1], Marc Hall[1], Christopher Ritter[3], Susan Jones[1], Samuel Mascarenhas[4], Bridget Cooper[1], Ana Paiva[4] and Ruth Aylett[3].

Abstract. In this paper we discuss the early stage design of MIXER, a technology enhance educational application focused at supporting children in learning about cultural conflict, achieved through the use of a game with an effective embodied AI agent. MIXER is being developed re-using existing technology applied to a different context and purpose with the aim of creating an educational and enjoyable experience for 9-11 year olds. This paper outlines MIXER's underpinning technology and theory. It presents early stage design and development, highlighting current research directions.

1 INTRODUCTION

Creating interesting and enjoyable role-play games for a serious purpose provides considerable challenges to developers. Role play games are notoriously expensive, with most successful games the result of large teams and considerable development time. Role play games typically include a cast of characters who need to act in a credible and believable way that engage the user and provide the essential information permitting the user to succeed in the game. Whilst games engines such as UNITY ensure that games mechanics, graphical display and essential functionality are relatively easy to incorporate, achieving complex and sophisticated cognitive and affective character behaviour typically requires significant development.

This paper outlines research being conducted as part of the European funded FP7 project, eCUTE (education in Cultural Understanding, Technology Enhanced). The aim of the project is to research and develop computer based innovative techniques to make users aware of cultural and group differences around them. Conventional role-play and game-based simulations such as Barnga! [29] are widely used with the aim of creating safe environments in which participants can be exposed to emotional states such as culture shock and those arising from intercultural conflict and then reflect on their own experience. In eCUTE, we aim to provide such game-based learning, with applications under development that are aimed at using role-play based intelligent software to engage user with affective synthetic characters which simulates cultural differences based on theories in Cultural and Social Psychology.

eCUTE focuses on culturally specific expressive emotional behaviour using autonomous synthetic characters that display behaviour representing a synthetic culture. The project draws upon theories in social psychology, emotion and in inter-cultural communication to create these characters. The characters will be virtual actors that embody models of culturally-specific behaviour; through various interactions with the game, children and young adults, differences in cultures experienced and explored.. Affective and narrative engagement of learners in these scenarios is seen as an important motivating mechanism for meeting the pedagogical goals of the system.

Within eCUTE, we are developing an application for 9-11 year olds focused at culture-related conflict. MIXER (ModeratIng Cross-Cultural Empathic Relationships) focuses on enabling the children to identify, explore and understand differences between cultures. It was developed with two main criteria:

- To ensure that users not only learn but that they also have enjoyment and fun as part of their interactive experience. To achieve this we needed to provide a scenario where we could highlight cultural conflict, but where the interaction engaged the children in a context that would be both interesting and fun for them.
- To re-use FearNot! [6] to provide the existing architecture, technology framework, look & feel, characters and environment, thus massively reducing required development.

This paper presents our early stage design and development activities aiming to meet this criteria. Section 2 provides an overview of the context, focusing on cultural conflict. Section 3 provides an overview of our technological context provided by FearNot! Section 4.1 outlines our use of Hide and Seek as a cultural conflict scenario, with section 4.3 detailing how we intend to incorporate cultural dimensions into the synthetic cultures. Section 5 discusses our approach and outlines current and future research directions. Finally, in section 6 we present brief conclusions.

2 MIXER's Purpose: Experiential Learning of Cultural Conflict

Europe has become the centre of global diversity, populated by a huge diversity of economic, political and social immigrants and migrants and annually visited by millions of tourists. In 21st century Europe, many cultural, ethnic and religious groups must live and work together. However, such integration is not always a smooth process and cultural differences can lead to social stresses and sometimes outright conflict. Providing an educational experience where such issues are explored is difficult in the traditional classroom environment. However, AI and games offer considerable potential for experiential and enjoyable experiences that could impact upon children's cultural understanding.

[1] Department of Computing and Technology, University of Sunderland, UK. Email: {lynne.hall, john.hodgson, marc.hall, susan.jones, bridget.cooper}@sunderland.ac.uk;
[2] Departamento de Ingeniería Electrónica E.T.S.I. Telecomunicación, Universidad Politécnica de Madrid, Spain. Email: syheerah@die.upm.es;
[3] School of Maths and Computer Science, Heriot-Watt University, UK. Email {a.nazir, c.ritter, ruth}@macs.hw.ac.uk;
[4] INESC-ID, Portugal. Email: {Samuel.mascarenhas anna.paiva}@inesc-id.pt

Exploring cultural conflict essentially relates to perceptions of group membership. Such membership bolsters self-importance and boosts self-esteem of individual members [3]. Thus, children (and adults) connect themselves to others through the evaluative implications of a set of common physical [27] or moral traits [4]. Children however, consistently rationalize differences and categorize individuals primarily based on physical characteristics (i.e.: skin or hair colour, body size, language etc.) [1, 19] above and beyond gender [27] and moral traits [21, 8]. Consequently, they use this attributional information to decide on potential friendship (see [27]). Whereas in-groupers are favoured and perceived "as different as snowflakes" [8] p.34, – similar but positively distinctive, out-groupers are often denigrated and stereotyped as being all much the same. While there is some debate as to the age at which in-group preference and out-group prejudice begin to decline as children develop better cognitive abilities (where they start examining the individual characteristics of members of out-group rather than stereotyping the members as a whole), in-group bias is prevalent throughout the primary school years - 7-12 years old [7, 31] and can increase during these years [24].

A child's experience is dominated by his or her affective-perceptual processes [1] that are associated to fear of the unknown and familiarity attachment. Thus to avoid uncertainties, a child most likely attaches him or herself to a similar group and usually considers out-group members as a threat [27, 23,], or to some extent, inferior [21, 20]. Although children's preference for similar group is determined primarily by physical attributes, several findings [21, 3, 4, 20] point that discrimination towards members of out-group is also based on status, consistent with the Social Identity Theory [28] – a widely accepted theory accounting to social prejudice in adults. Nesdale (2004) [21] asserts that children as young as 3 years start to develop awareness about which groups carry better image, and prefer memberships with groups that are regarded highly or considered superior.

The moral circle theory [4] makes a similar assertion. The theory posits that people identify themselves with a particular group that exhibits a set of moral traits of equal 'standard' (moral identity). Anyone who is outside this circle is viewed as inferior. In children (and adults), the tendency towards prejudice will increase as tension and conflict increases between ingroup-outgroup and will reach its peak when the "inferior" group threatens the social standing of the "superior" one [4, 21].

Insufficient information about those outside the group causes insecurity in children and threatens their social identity (group status). This evokes the need to restore a good self-image in order to maintain self-esteem [4], and an effective way to achieve this is by negatively evaluating the out-group members [25, 26].

Often, prejudice in children is seen as a mirror that reflects the society's attitudes and values [20, 21], regularly transmitted by the closest people who daily interact with them. However, there is a wealth of evidence that shows that correlations between children's prejudice and prevailing societal norms have been between low and nonexistence (see [2]). This shows that children do not just sponge up dominant ethnic attitudes by the community but also seek to understand and process their experiences through active participation in their interpersonal worlds, but this depends on whether they have acquired sufficient information to be able to engage in proper moral reasoning.

One way to do this is through extended contact [16, 32] – where an in-group member becomes an active participant in the activities of an out-group member, gets to understand the latter's values and rituals and consequently disconfirms the negative beliefs about the whole out-group. In other words, the out-group member is seen as a model whose positive exemplar is extended to the group as a whole. Studies by Wright and colleagues (see [32]) confirmed that an in-group member that has friendship with an out-group member leads to more tolerant and positive intergroup attitudes. A similar study was replicated by Liebkind and McAlister [16] in promoting tolerance between native and non-native Finnish children showed favourable attitude changes when a particular child from both groups are brought into contact with each other.

Hence, it is not necessary to completely dispel existing group boundaries or forcing them to reach a mutual compromise in order to engage children in intergroup play and friendship, but rather to keep it less salient while concurrently establishing ways to facilitate some sort of contact [13]. This is where an application such as MIXER plays a role - as a plausible platform in enhancing children's intergroup attitudes in an anxiety-free environment, by engaging children in new cultural experiences through active participation with out-group synthetic agent's representative(s) and facilitating the generalisation of the positive effects towards out-group peers as a whole. Among major advantages of such application include training children to combat negative preconceptions by looking at things through the perspective of out-group members (i.e. why certain things are done in certain ways) and making them experience the feelings of such children - which will subsequently enhance empathy skills. Additionally, social training of this sort may provide a better solution to the problem discussed as observing interactions among synthetic agents (or directly interacting with those agents) does not evoke anxiety in the user.

3 MIXER technology: ReUsing FearNot!

In eCUTE the pragmatic decision to base MIXER on FearNot! (Fun with Empathic Agents Reaching Novel Outcomes in Teaching) will significantly reduce development time. Re-using this architecture enables development effort to focus on the extension of FearNot! to incorporate cultural factors.

FearNot! is a school-based Virtual Learning Environment (VLE) consisting of synthetic characters representing the various actors in a scenario related to bullying issues. FearNot! uses emergent narrative to create episodes with those characters. The goal of FearNot! is to enable children to explore bullying issues, and coping strategies, by interacting with characters to which they become affectively engaged. User empathy is triggered by the different properties of the characters such as their appearance, behaviours and emotions.

3.1 What does FearNot! look like?

FearNot! engages children's interest by letting the children role-play as an advisor (invisible friend) to the bullying victim. The episodes introduce different characters and then show some bullying incidences and then the user interacts with the victim

character and advises the character what he/she should do to cope with the bullying situation. And then the story then emerges from there. The following screenshots show the interface for FearNot! application.

Figure 1: Bullying Episode

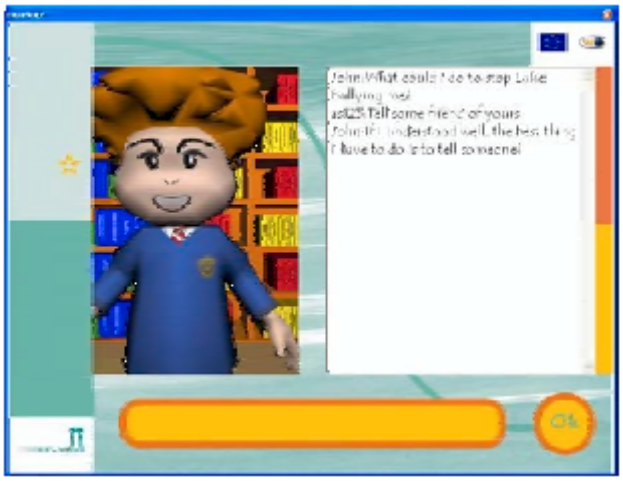


Figure 2: User Interaction with Victim.

3.2 FearNot! Architecture

Autonomous agents running with FATiMA architecture work as the character minds to generate affective behaviour for the characters as the story goes on [6]. Although FearNot! is driven by emergent narrative it is very important to keep the learning goals in context and the story to run towards these goals. A story facilitator [5] in the architecture keeps check on the emergence and guides the story and interaction. The following diagram shows the FearNot! architecture:

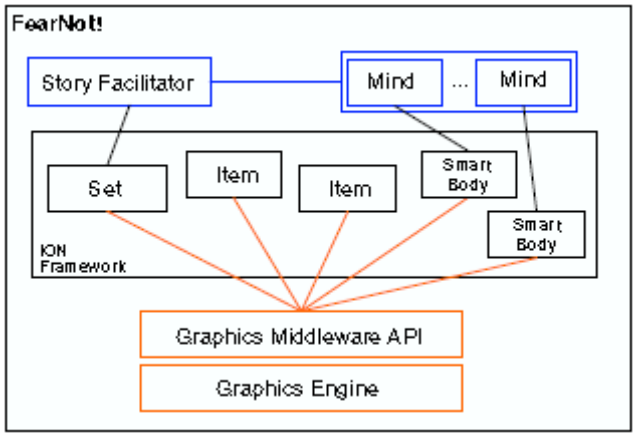


Figure 3 – FearNot! Architecture

The layered architecture that FearNot! uses consists of three layers: The application layer, the world model and the Graphics layer. Application layer combines the user-interface, world model with the FATiMA architecture which is the architecture for the Affective characters in FAtiMA [11] where the character minds are running and also initiates the story facilitator. [5]. The world model consists of the ION framework which includes symbolic representations of entities in the application, The ION framework [30] is used create abstraction between to communicate between two entities. And finally the last layer consists of graphics engine and the graphical objects.

3.3 FAtiMA (FearNot Affective Mind Architecture)

FAtiMA is used to build the affective agents in FearNot!. FAtiMA presents two main layers for the appraisal and coping processes. Emotional Reactions and reactive behaviours are formulated in the reactive layer, while the goal-oriented behaviour is the outcome of the deliberative layer. It's also composed by two main memory components: the Knowledge Base that stores semantic knowledge such as properties about the world and relations. The autobiographic memory stores episodic information concerning previous events and the personal experience. Figure 4 shows the major components of FAtiMA architecture.

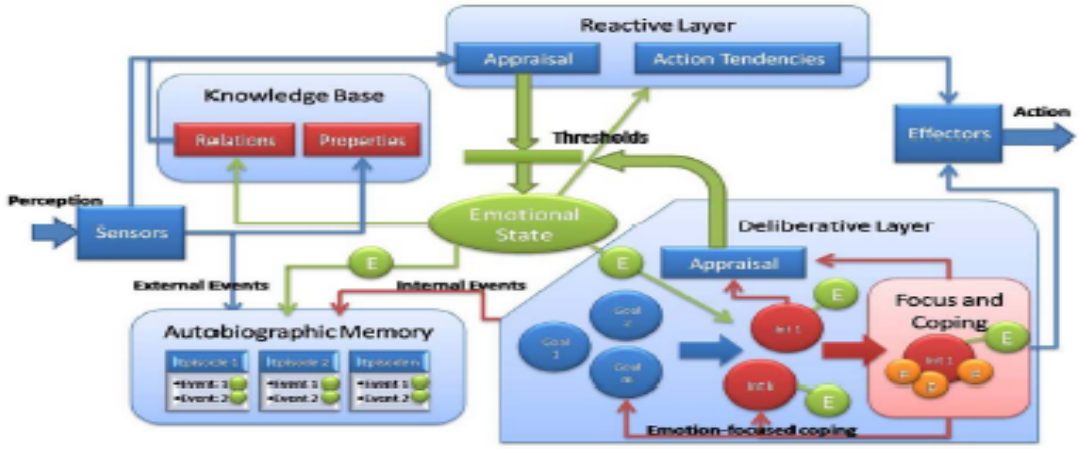


Figure 4: FAtiMA's architecture.

After perception of an event in the world appraisal is done at the reactive layer which results in possible generation of a set of emotions (emotional concepts in FAtiMA are based on the OCC model [22]) memories are updated simultaneously memories are updated with the change in the world values and the events are stored in to the autobiographic memory of the agent. The perceived event is then used to initiate the goal activation process. After a goal is selected the reactive and deliberative layers use the information stored in the memory which also includes emotional information to decide what action to take next. Then the effectors are used to send the selected action to the world.

3.4 Maximizing the Use of existing software

We know from previous projects that the FearNot! approach is plausible and can be quite useful with children of the target age [12]. The development of MIXER aimed to develop scenarios and interactions that made the maximum use of the existing FearNot! software architecture and also identified ways it provides to enhance the ability of FearNot!

4 MIXER's DESIGN

With MIXER, we had a number of significant constraints. Firstly, that whilst we were looking for a cultural conflict situation, however, that cultural conflict was not to be based on race, religion or politics rather it was to be based on a synthetic culture. We wanted our scenarios to be realistic, yet we wanted our cultures to be synthetic (mainly not being translatable to any one ethnic group)

Secondly, as we were using FearNot! we had to follow an episodic structure where the role of the user would be that of an invisible friend or advisor. And finally, we really wanted our users to have a fun interaction, something that sometimes seems to be forgotten in the development of serious games.

4.1 Why Hide & Seek?

Our initial design ideas with MIXER focused on an episodic soap opera style format (much as FearNot! had been), where conflict between groups would be provided through a storyline about an in and out group. Trying to determine who those groups should be and why there should be cultural conflict provided us with somewhat earnest scenarios. It was readily apparent that such serious content would be of little interest to the intended users. The fact that we were meant to be providing games based learning did suggest that somehow there should be some element of fun in the application.

Our user needed to be more than a commentator about a situation within which they had little buy-in and possibly wouldn't really understand. Rather they had to be able to envision themselves in the situation of the characters. Returning to the basic fact that we were meant to be creating a game we decided to explore the games that children play and examined their potential to provide an opportunity to explore cultural conflict.

There are a few games that are played in almost every culture by the majority of children. One of the most typical of such games is Hide and Seek, played everywhere by both genders. The rules for Hide and Seek are not dependant on race, nationality or politics, rather they are handed down and modified based on children's experience and agreement in the game space. Children are often aware that others (even in the next street) may have different rules for Hide and Seek.

Typically where rules are not the same, conflict will occur, with cultural expectations (e.g. the rules of hide and seek) not being adhered to. Whilst older children will generally define rules before starting to play, late primary children will generally only discover the difference in rules when conflict occurs, often with game abandonment and shouts of "its not fair" and "I don't want to play any more."

4.2 The Scenario

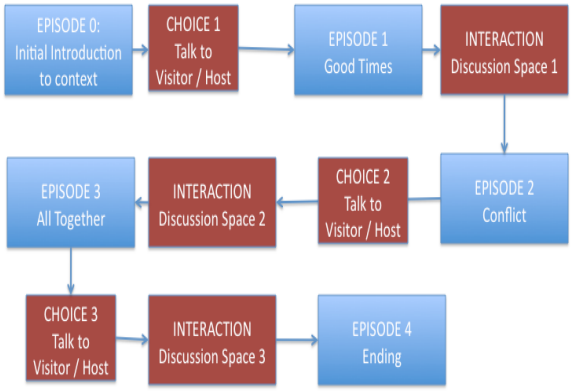


Figure 5: Mixer General Episodic Outline

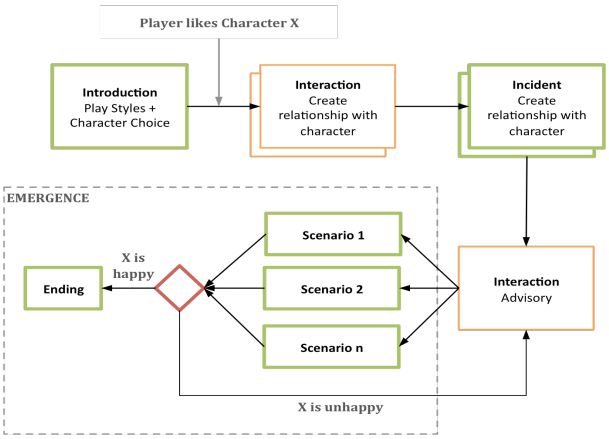


Figure 6: Intergrating Emergence in MIXER

As figure 5 outlines, MIXER is composed of 5 episodes where the user decides who to interact with and has 3 interaction points where the user talks to their selected character. In Episode 0 (figure 7), the user is introduced to the context with simple back story being presented, that 2 schools are attending an activity week. One of these schools is the host school and the other are visitors. At this point (Choice 1) the user can decide which school they want to watch further. The aim of the first episode and interaction is for the user to start to make friends with character in a positive, non-conflict oriented way, so that user is focused towards being an everyday friend rather than shoulder to cry on.

In the next episode (figure 8) the user sees a game of Hide and Seek where only 1 child is from the school not selected by the user. The rules of the Hide and Seek will not match and the user will watch conflict as the children fail to agree how to play the game, followed by its abandonment. Again at the end of this scene the user can select who they talk to and then discusses what has happened. The user will be asked to suggest what the agent can do to help deal with the conflict that involves cognitive, emotion and behavioural elements (figure 9). This part is zoomed out in figure 6, where the user's suggestion or advice combined with the agent's decision will influence the emergence of the events.

In Episode 3, both schools play together, it may be that more conflict ensues or perhaps the agent that the child has talked to will ensure that rules are clear early on to avoid misunderstanding. The agent could explore different suggestions at different times as trial and error in order to see which one works best, until he is finally happy (figure 6). As the architecture will permit MIXER to exhibit emergent behaviour the focus of this episode is not predictable. In the final episode a positive message will be given to the child by their selected character, either that they will continue to try to improve the situation or that the situation is now resolved.

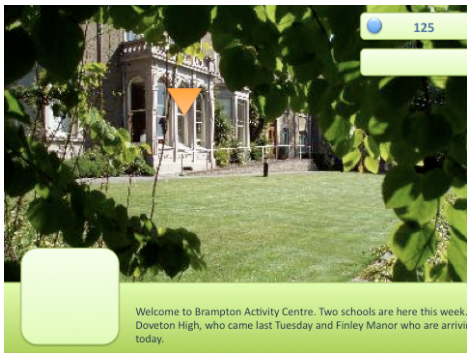


Figure 7: Episode 0

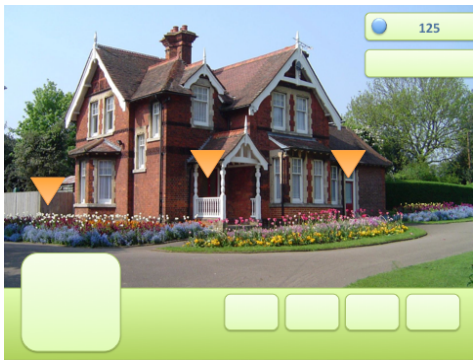


Figure 8: Possible hiding places

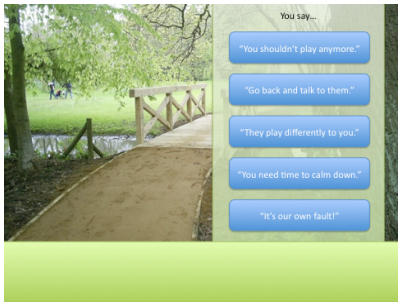


Figure 9: Discussion point with user

4.3 Synthetic Cultures: Dimensions of Hide and Seek

Synthetic cultures are simplified notion of real-world cultures, inspired by human cultural dimensions, but reflect the behavioural tendencies related to a specific extreme of a particular dimension [18]. For instance, an event may reflect the extreme side the individual/collectivistic dimension, instead of introducing elements of all the dimensions, as in real cultures. Our approach to synthetic culture is based on that of Hofstede [14, 15], who provides a 5-dimensional model. Using this as a framework has enabled us to highlight the extremes 3 of the 5 dimensions within the context of Hide & Seek. Table 1 identifies these dimensions and figure 10 an initial indication of how they could be represented in MIXER.

Dimension	**One Cultural Extreme**	**Other Cultural Extreme**
Identity	Indivualist	Collectivist
Hierarchy	High power distance	Low power distance
Gender	Masculine	Feminine
Truth	Uncertainty Avoidance	Uncertainty tolerant
Virtue	Long term orientations (Shotor)	Short term orientation (Lotor)

Table 1: taken from [15]

Hide and Seek has a myriad of potential rules, for example, as figure 9 reveals, when the rule "has been caught" is applied, what happens next? In some games, being caught means that the game is finished and the hider returns to base. Or it can mean that the hider joins the seeker and assists them in finding hiders. In other instances, "has been caught" means that the user is "frozen" and can no longer move. In some variants, frozen remains the state until the game ends, whilst in other approaches, hiders can be unfrozen by other hiders when the seeker moves on to another location.
These differences in rules can be mapped onto Hofstede's dimensions. For example, a more feminine culture is one in which the user can be saved (e.g. unfrozen by a fellow hider), whilst in a masculine culture, the more extreme position would be that once caught the games ends for the player. In an uncertainty tolerant culture, seekers might risk being caught, if they were aware that they could be re-engaged in the game. Although these mappings are relatively simplistic, they do allow a game as well known and easy to understand as Hide and Seek to provide synthetic cultures that can represent levels of extremism on Hofstede's dimensions.

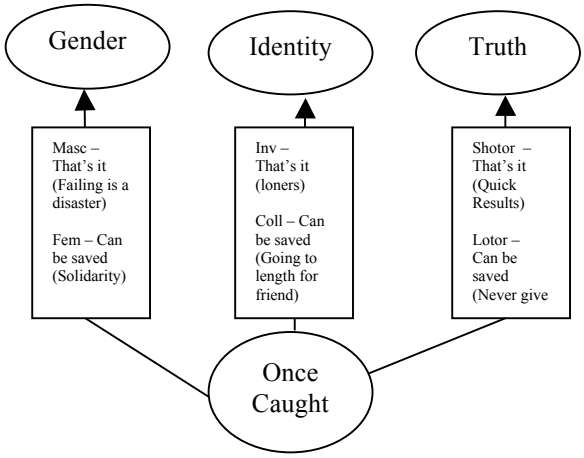


Figure 10: A rule mapped to cultural identities

4.4 Possible changes to existing Architecture

The prototype being worked on serves multiple purposes, initially it will serve as an experiment to implement and evaluate the new social framework with the existing architecture. This evaluation will act as a requirement analysis tool for possible changes required to the existing architecture. It is this implementation that will help us understand whether we need to change any particular components or concepts within the FAtiMA and ION framework.
Although, possible changes to the architecture will be clearer after the prototype has been testing, there are certain concepts we envisage in future implementations. One concept is 'Theory of Mind', since these scenarios may require a character to influence the goals and actions of others. 'Theory of the Mind' would allow the agent to perceive the possible responses or reactions of other agents.

Another possible change would enable certain characters in the environment to form a group dependent upon certain parameters, which serve to maintain relationships and hierarchy inside the group providing greater uniformity.

Other adaptations we are currently working are the replacement of the graphics environment OGRE with Unity 3D, considerably reducing the authoring effort by using Unity's easier interface and asset management. We also aim to make the implementations platform independent through the use of Unity. This will allow us to implement the game through a web browsers or, if required, on mobile phones. The intention here is a reduction in the Authoring time.

This approach appears to be viable and works, but as with any other innovative technology it requires iterative approach with this prototype being used to point the required modifications which will improve not only the working of FearNot! architecture but also the scope of social concepts that will be applied with in it.

4.5 Next Steps: Evaluating the Early Stage Mixer

Currently, we are engaged in two separate activities. Firstly, we are involved in user testing ensuring that the scenario is appropriate, understandable and enjoyable. This very early stage user involvement will use a low tech solution, where children and teachers are asked to comment on a version of MIXER provided through storyboarding software (on the screen, but largely passive).

Secondly, we are authoring the identified episodes and interaction points in a mid-tech prototype of MIXER. This mid-tech prototype will use aspects of the FearNot! technology and will have the FearNot! look and feel, with characters, sets and animations developed to provide the appropriate backdrop to the interaction. Although MIXER will use emergent narrative, we will take a scripted approach with this mid-tech prototype as this enables us to quickly evaluate it with users.

The user-centric approach taken in eCUTE requires that these scenarios and ideas are evaluated with the intended user group. This evaluation not only needs to explore children's reactions to MIXER itself, but additionally needs to identify if children can recognize cultural dimensions. The approach we will take to the early stage design evaluation of MIXER will be based on the use of certain techniques used as a line of enquiry that match the recommendations for pedagogy and practice of UK government in the area of 'Engagement Activities' [9].

One of the broad principles of 'Engagement Activities' is based upon are 'Directed activities related to text' (DARTs) developed by Lunzer and Gardner in the 1970s and 1980s [17]. In this text not only relates to the written word, but also diagrammatic representations or pictures.

These engagement actives, such as the 'Odd one out' develops thinking skills as they describe reasoning for the difference along with similarity; but are also fun as they allow the child to think allowed and exchange ideas without the fear of being wrong [9, 10]. Activities that have been developed, based upon the principle of 'Engagement Activities' have been developed by teaching professionals and educational consultants, and are starting to become a common part of a teachers repertoire, with it being possible to map them to research tools found in Interaction Design or Evaluation with children.

With MIXER, we will use a range of engagement techniques, including card sorting, the use of a 'Thinking Box' (where children enter up to 9 words related to a specific question), discussion groups and "living questionnaires" where children take many paces forward or back depending upon their view, or a continuum in which pupils negotiate their position along a line of pupil or place them self at a given point in a line. Making the evaluation add value rather than a burden on stakeholders and user, which is a key aim of eCUTE.

5 DISCUSSION

This paper has outlined the early stage development of MIXER, with MIXER currently just about to be evaluated with the intended user group. Our aims with this evaluation are two-fold, firstly to ensure that the user experience is fun, but still enhances learning about cultures. Secondly, to identify essential extensions to the underlying architecture and the digital assets (e.g. sets, character animations) to allow us to appropriately display the dimensions and to ensure characters behave in an expected manner.

Early and informal discussions with children and teachers suggest that Hide and Seek is a good scenario choice: inclusive, popular and easy to understand. Currently, we are focusing on how we can incorporate the cultural dimensions into Hide and Seek in such a way as to make them visible but not intrusive or inappropriate. We are also focusing on how we can evaluate the users' awareness of these dimensions in a manner appropriate for 9-11 year olds. As detailed we intend to use engagement techniques and we are engaged in crafting this approach basing this on best practice information disseminated by the UK Department of Education and verifying with teacher input.

MIXER has been both enhanced and constrained by the project decision to re-use FearNot! to provide the existing architecture, technology framework, look & feel, characters and environment. Reusing the existing components of the FearNot! architecture will save huge amount of effort and time that goes into developing such complex AI systems. Autonomous agents are the most important part of this scenario and FAtiMA architecture provides a very powerful and workable solution for designing authored or autonomous agents

The existing FearNot! Architecture has the ability of combining pedagogically motivated emergent narrative produced using both autonomous agents and user input. The designed scenario has the potential to enhance the emergent narrative output by giving the user the option of choosing agents or group to be friends with and advise.

The overall architecture in FearNot! is connected and communicates with different components using ION integration framework which brings together the different ends of the software: the graphical appearance of the environment and agents, and the user input together to be used both by the graphics engine and the FAtiMA based agents. Although sets, props and animations have to be created and incorporated this is a relatively simple task with the content separated from the architecture thus readily permitting the incorporation of new content.

We have also changed the user interaction for MIXER. In FearNot! interaction was limited to free text entered as the user interacting as the invisible friend with a child character (this child was the victim in a bullying scenario). In MIXER we are aiming to give the user more choice. Thus children will be able to decide which group do they want to interact with, whose side of the story do they want to hear, etc. These choices not only extend the interaction and perspective of the user, but are also useful information for the development team, particularly the psychologists and cultural theorists. In this scenario since the interaction will be choice based, it makes it easier for both the user and the system to operate and communicate. It helps us in avoiding the complex lexical analysers and glossary of inputs words and then constructing meaningful events for the agents to understand and also, it's very time consuming and difficult for most children in this age group to type text in.

6 CONCLUSIONS

This paper has outlined the early stage development of MIXER, a game-based learning application that will provide users with the opportunity to engage with characters in synthetic cultures. Hide and Seek provides an ideal game with which to provide synthetic cultures, providing a context where cultural difference is not based on race, religion or politics, but rather on the application of the rules of a well known game. MIXER has been developed using existing technology thus rapidly speeding up the development process. A low tech prototype of MIXER is just about to be evaluated with 9-11 year olds. A mid tech prototype is currently being authored and will be tested in early March 2011.

This work was partially supported by European Community (EC) and is currently funded by the ECUTE project (ICT-5-4.2 257666). The authors are solely responsible for the content of this publication. It does not represent the opinion of the EC, and the EC is not responsible for any use that might be made of data appearing therein.

REFERENCES

1. Aboud, F.E., Children and Prejudice. 1988: Oxford: Basil Blackwell.
2. Aboud, F. and A. Doyle, Parental and Peer Influences on Children's Racial Attitudes. International Journal of Intercultural Relations, 1996. 20: p. 371-383.
3. Aquino, K.F. and A. Reed II, The Self-Importance of Moral Identity. Journal of Personality and Social Psychology, 2002. 86(3): p. 1423-1440.
4. Aquino, K.F. and A. Reed II, Moral Identity and the Expanding Circle of Moral Regard toward out-Groups. Journal of personality and social psychology, 2003. 84(6): p. 1270-1286.
5. Aylett, R., Figueiredo, R., Louchart, S., Dias, J., Paiva, A.: "Making It Up as You Go Along - Improvising Stories for Pedagogical Purposes". In Proceedings of IVA 2006, pp.304-315 (2006)
6. Aylett, R.S, et al. FearNot! – An Emergent Narrative Approach to Virtual Dramas for Anti-bullying Education Virtual Storytelling. Using Virtual Reality Technologies for Storytelling Lecture Notes in Computer Science, 2007, Volume 4871/2007, 202-205, DOI: 10.1007/978-3-540-77039-8_19
7. Banks, W.C. and W.J. Rompf, Evaluative Bias and Preference Behaviour in Black and White Children. Child Development, 1973. 44: p. 776-783.
8. Corenblum, B., What Children Remember About Ingroup and Outgroup Peers: Effects of Stereotypes on Children's Processing of Information About Group Members. Journal of Experimental Child Psychology, 2003. 86: p. 32-66.
9. Department for Education Skills and Children. Active engagement techniques. Pedagogy and Practice. 2004. Available at: http://education.gov.uk/publications/eOrderingDownload/DfES 0434-2004G PDF.pdf.
10. Department for Education Skills and Children. Leading in Learning. Pedagogy and Practice. 2004. Available at: http://education.gov.uk/publications/eOrderingDownload/DfES%200439-2004G%20PDF.pdf
11. J. Dias and A. Paiva. Feeling and reasoning: a computational model for emotional agents. In Proceedings of 12th Portuguese Conference on Artificial Intelligence, EPIA 2005, pages 127–140. Springer, 2005.
12. Hall1, L, Woods, S, Aylett, R. FearNot! Involving Children in the Design of a Virtual Learning Environment. 2006. International Journal of Artificial Intelligence in Education. Volume 16, Number 4/2006. Pages 327-351.
13. Hewstone, M. and R.J. Brown, Contact Is Not Enough: An Intergroup Perspective on the Contact Hypothesis, in Contact and Conflict in Tergroup Encounters, M. Hewstone and R.J. Brown, Eds. 1986, Blackwell: Oxford. p. 1-44.
14. Hofstede, G, Hofstede G, J, and Minkov, M. Cultures and Organizations: Software for the Mind, Third Edition. McGraw-Hill; 3 edition (May 3, 2010). ISBN-10: 0071664181.
15. Hofstede G, J, Exploring Culture: Exercises, Stories and Synthetic Cultures. Intercultural Press (August 2, 2002). ISBN-10: 1877864900
16. Liebkind, K. and A.L. McAlister, Extended Contact through Peer Modelling to Promote Tolerance in Findland. European Journal of Social Psychology, 1999. 29: p. 765-780.
17. Lunzer, E. and Gardner, K. (1984) Learning from the written word. Oliver and Boyd. ISBN: 0050037676.
18. Mascarenhas, S., et al. Using Rituals to Express Cultural Differences in Synthetic Characters. in Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1. 2009. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.
19. Malik, K., The Meaning of Race. 1996, London: MacMilland.
20. Nesdale, D., The Development of Prejudice in Children, in Understanding Prejudice, Racism and Racial Conflicts, Augoustinos and Reynolds, Eds. 2001, Sage: London. p. 57-73.
21. Nesdale, D., Social Identity Processes and Ethnic Prejudice in Children, M. Bennet and F. Sani, Eds. 2004, NY: Psychology Press. p. 219-245.
22. Ortony, A., Clore, G., and Collins, A. (1988). The Cognitive Structure of Emotions. New York, NY: Cambridge University Press.
23. Quillian, L., Prejudice as a Response to Perceived Group Threat: Population Composition and Anti-Immigrant and Racial Prejudice in Europe. Sociological Review, 1995. 60: p. 586-611.
24. Rice, A.S., R.A. Ruiz, and A.M. Padilla, Person Conception, Self-Identity, and Ethnic Group Preference in Anglo, Black and Chicano Preschool and Third Grade Children. Journal of Cross-cultural Psychology, 1974. 5(1): p. 100-108.
25. Steele, C.M., The Psychology of Self-Affirmation: Sustaining the Integrity of the Self, in Advances in

Experimental Social Psychology, L. Berkowitz, Editor. 1988, Academic Press: NY. p. 261-302.
26. Steele, C.M., S.J. Spencer, and M. Lynch, Self-Image Resilience and Dissonance: The Role of Affirmational Resources. Journal of Personality and Social Psychology, 1993. 64(6): p. 885-896.
27. Targowska, A.U., Exploring Young Children's 'Racial' Attitudes in an Australian Context - the Link between Research and Practice. 2001.
28. Tajfel, H. and J.C. Turner, An Integrative Theory of Integroup Conflict, in The Social Psychology of Intergroup Relations, W.G. Austin and S. Worchel, Eds. 1979, Brooks-Cole: Monterey, CA.
29. Thiagarajan, S. (1990) Barnga: A Simulation Game on Cultural Clashes. Intercultural Press
30. Vala, M et al. ION Framework – A Simulation Environment for Worlds with Virtual Agents, Intelligent Virtual Agent Lecture Notes in Computer Science, 2009, Volume 5773/2009, 418-424, DOI: 10.1007/978-3-642-04380-2_45
31. Weiland, A. and R. Coughlin, Self-Identification and Preference. A Comparison of White and Mexican-American First and Third Graders. Journal of Cross-cultural Psychology, 1979. 10(3): p. 356-365.
32. Wright, S.C., et al., The Extended Contact Effect: Knowledge of Cross-Group Friendships and Prejudice. Journal of Personality and Social Psychology, 1997. 73(1): p. 73-90.

Recognition of Emotional Brain Activities in Virtual Reality Environment: A Position Paper

F. Abbattista[1], G. Attolico[2], V. Carofiglio[1], F. De Felice[1] and G. Dimauro[1]

Abstract. Emotion recognition is one of the key steps towards emotional intelligence in Human Computer interaction (HCI). Using a brain-computer interface (BCI) to detect and identify emotions could improve the quality and effectiveness of HCI. Researchers have shown that it is possible to extract emotional cues from electroencephalography (EEG) measurements that become a way to investigate the emotional activity of a subject beyond his conscious and controllable behaviours. In this view, we aim at designing, building and testing a flexible system for the recognition of emotions of users on the base of their brain activity. This main goal involves several topics of investigation: (1) How to design such a system? Which architecture and techniques are more suitable? Which is the most proper kind of interaction? (2) Which is the best way (instrument, set-up) to acquire EEG data with relevant information for evaluating the emotional state of the user? (3) What processing techniques and classification methodologies are best suited to recognize emotion from EEG data? (4) How self-induced emotions could be used in a BCI paradigm (real-time data processing)?

1 INTRODUCTION

Researchers have recognised that humans show emotions when interacting with a computer [1]. This has been largely exploited for therapeutic purposes (a good example among many is [2]), in educational situations (especially combined with "serious games" [3]) or for motivational purposes (see for instance the large literature in the recent research area of "persuasive computing" [4]). Typically these works concentrate on automatic recognition of emotions from speech [5,6,7,8], facial expressions [9,10,11] or their combination. Recently, the development of haptic technology [12] has enabled the investigation about the relationship between emotions and virtual tactile stimuli [13, 14, 15]. Another method for measuring human emotion is the acquisition and processing of physiological signals [16]. Several researchers [17, 18, 19] have shown that it is possible to extract emotional cues from electroencephalography (EEG) measurements, which become a way to investigate the emotional activity of a subject beyond his conscious and controllable behaviours. The neurons of the brain produce a constantly present rhythmic signal that can be divided into several bands (namely alfa, beta, theta, delta), based on frequency. One of the most apparent results states that different emotional states generate different peak frequencies in the alpha band [19]. Another important distinction, in the psychology literature, is made between two dimensions of emotion: the valence (ranging from negative to positive) and the arousal (ranging from calm to excited) [20]. Cognitive researchers have investigated how changes along these two dimensions modulate the EEG signals and have determined that the position of an emotion in this two dimensional plane can be derived from EEG data [8, 21, 22]. The advantage of recognising emotion using EEG is that brain activity provides direct information about emotion that does not depend on user controllable behaviour such as speaking and facial expressions. However, the use of EEG signals to this aim still needs a lot of improvement. Earlier research has shown that human computer interfaces, which exploit this kind of signals do not perform very well and are able to recognize emotion correctly in about 60-70% of the examined cases [8]. Further research is needed: The theory and the methodologies must be validated using a wide range of quite different applications.

Using EEG techniques to recognize user's emotions by using a commercial low-cost device could open many other possibilities. In the medical field, the automatic recognition of emotions from clues other than words or facial expressions could be used to objectively assess the emotional state of people (by therapists and/or psychologists, by physician to evaluate the results of clinical tests whose results are affected by the emotional state of the patient, by the people as a bio-feedback to learn the control of their internal state, etc.). Another application is to help people that suffer from severe muscle diseases to communicate their emotions using proper brain-computer interfaces [20]. In the field of human-machine interaction this technique could be applied in many different situations such as software adaptation, educational games or tutoring systems. Moreover, a brain computer interface could add emotional information to computer mediated human-human conversation (e.g. instant messaging, online games, chat rooms).

In this paper we will focus on how Brain Computer Interface (BCI, [23]) could be used to detect and identify emotions in Virtual Reality Environments (VRE). In doing this, we will propose a kind of VRE only as medium in which the BCI-based emotion's recognition system will be tested. Section 2 includes related work on emotions and on the correlation between them and brain activity, in BCI applications. Section 3 provides an overview of the BCI-based system we plan to develop and a brief description of the Virtual Reality Environment domain we choose to link to it. Section 4 gives a step-by-step description of the development phases: Data acquisition, data processing and classification and test on real-time applications. An hypothesis for performing a preliminary data acquisition is in Section 5. Section 6 provides some basic reflections about how to exploit recognized EEG pattern of emotions in a BCI-based VRE. Some final considerations are in section 7.

[1] Dept. of Informatica, Univ. of Bari, Italy. Email: {fabio, carofiglio, fabio.defelice, dimauro }@di.uniba.it.

[2] Institute of Intelligent Systems for Automation – National Research Council, Bari, Italy. Email: attolico@ba.issia.cnr.it.

2 RELATED WORKS

For recognizing emotion from brain activity, we need information about how the brain handles emotion, and information on how to measure the brain activity. Several methods of measuring brain activity exist. Among them we plan to exploit EEG signals due to the fact this kind of signals can be measured at any moment and by cheap devices.

Moreover, emotion is a phenomenon that is difficult to grasp. Psychologists worked at decoding emotions for decades, by focusing on two main questions: (i) How can emotions be classified?; (ii) Which is their functioning?, i.e. how are they triggered? How do they affect behaviour? Which is the role played by cognition? Two points of view prevailed: The first one assumes that a limited set of basic emotions exists, while the second one considers emotions as a continuous function of one or more dimensions: see, e.g., the "circumplex model" of affect [20]. Russell's circumplex model has two "axes" that might be labeled as displeasure/pleasure (horizontal axis) and low/high arousal (vertical axis). For example, "relaxed" is on the `pleasure' side of the first axis, but has a low value on the second (arousal) axis whereas ``bored'' also has low arousal but is located on the displeasure portion of the first axis (figure 1). Due to its simplicity (it is easy to express an emotion in terms of arousal and valence, whereas it is much more difficult to decompose an emotion into basic emotions) and universality (there is little controversy about the first two dimensions of the model.), this model is used in most studies.
A lot of work has be done on the correlation between emotions and brain activity, in Brain-Computer Interface applications (BCI, [23]): One of the earliest attempts to prove that EEG signals can be used for emotion detection is proposed by Chanel et al [21].

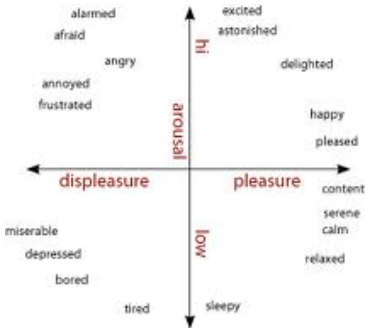


Figure 1. Circumplex model's of emotion – Russell

The authors tried to recognize the arousal dimension of emotion from both EEG and peripheral signals. The accuracy of combining both types of signals resulted in a boost of accuracy that reached up to 72%. They employed a 64-channel EEG device, as a consequence the huge amount of data to be processed makes the system heavy in real-time applications. Kostyunina et al [24] used 10 different electrodes in order to differentiate between four emotions that are joy, anger, fear and sorrow. They reached the conclusion that joy and anger emotions result in an increase in the peak frequencies of the alpha band whereas the case of fear and sorrow emotions result in a decrease in the peak frequencies of the alpha band. M.Mikhail et al [25], propose an approach that analyzes highly contaminated EEG data. They also use a feature selection mechanism to extract features that are relevant to the emotion detection task based on neuroscience findings and reached an average accuracy of 51% for joy emotion, 53% for anger, 58% for fear and 61% for sadness. Murugappan et al [18] investigated the possibility of using visual and audio-visual stimulus for detecting the human emotion by measuring electroencephalogram (EEG). They designed visual and audiovisual stimulus based protocols to acquire the EEG signals using 63 bio-sensors that were placed all over the surface of the scalp. They analyzed the EEG signals for classifying five emotions (disgust, happy, surprise, sad and anger) and reached a conclusion that the audiovisual stimulus based emotion recognition gives better classification accuracy over visual stimulus. In [26] a method for single trial classification using both EEG and peripheal physiological signals is presented. Regarding the EEG, according to the authors the obtained classification rate was, on average, 55.7% for arousal and 58.8% for valence.

The general approach for any system that rely on brain signals is a layered approach: There are three main stages that the signals have to pass through in order to reach a final decision which are signal pre-processing, feature extraction and classification. Signal pre-processing is the stage during which the signal is passed through a number of filters for artifact removal and for getting the signal ready for the next stages. A number of approaches for artifact removal exists (e.g. [27], [28]); after removing the artifacts, signals pass through the feature extraction stage. Feature extraction is the process of selecting features that are representative to the specific cognitive (or emotion) state and selective from other extracted features so that the system will not suffer from redundant features. This stage normally generates a large number of features. This requires extracting relevant and distinctive features for the classification task. There are several methods used for reducing the number of features [29]. One of the most commonly used techniques is principal component analysis (PCA) [30]. A classifier is a sort of a function that is able to learn the relationship between features and their classes: It can infer to which class it belongs. A good review of existing BCI classifier is in [31].

3 EMOTIONAL STATES RECOGNITION IN VIRTUAL REALITY ENVIRONMENTS

We plan to design, build and test a flexible system for the recognition of user's emotions from the brain activity. This system will be verified inside different real-time applications. The system will be made from three main components: a Virtual Reality Environment (VRE), a BCI interface (to capture the user's reaction during his/her interaction with the VRE) and a module for emotion recognition (ERC).

In figure 2, the user interacts with a virtual reality environment. During the interaction, the EEG device records the brain activity of the user and collects EEG data that are supplied to the Emotion Recognition Component. The recognized user's emotional state is fed to the VRE that use this information to

adapt the interaction and/or to guide the user feelings toward certain emotions.

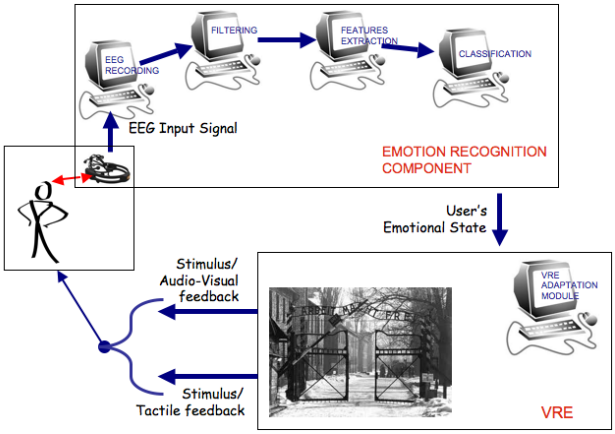


Figure 2. Example of the integration of a BCI-based module for emotion recognition in a Real-Time Application

To verify and validate our system we have chosen to implement the following scenario:

After sixtyfive years the possibilities to listening live attestation of the Nazi's horrors from the voice of the survivors to the extermination camps are getting fewer as they getting older. Sooner the only way to preserve the remembrance of that terrible phase of the European (and Worldwide) history will entrusted on indirect documentation in the form of videos, images and texts reporting interviews to last witnesses. However next generations pupils could not totally comprehend how terrible the Shoah was if relating only to those media. A way to maintaining alive the dramatic meaning of that experience could be to reconstruct a 3D virtual environment of one of those camps, such as Auschwitz. Users enter the camp by the main entrance under the sadly notorious motto "Arbeit macht frei", users can navigate among the prisoner's barracks, activating links to videos, photos documenting the Jewish and Gipsy's lifestyle in the 1940-1945 period, or by hearing songs that some prisoners composed during their permanence. A digital character representing a prisoner guides users through different parts of the camp, users could interact with him by asking, with vocal query, questions about his personal experiences.

The BCI is aimed at capturing the players' reaction to the presented contents to allow the virtual prisoner to dynamically adapt the visit to the user, avoiding media too upsetting for the user's sensibility. Examples of player's reactions are emotions which can be induced with different levels of arousal, valence or dominance: Disgust for bloody scenes, anger for abuses or sadness for suffering of others, etc. In our view, the main goal of the application will be to maintain alive the dramatic meaning of that experience without disturbing too much the more sensitive users. This means that the system should not avoid the player feels emotions such as disgust, anger or sadness, but it should let the player feels that emotions with an intensity below a given threshold. If the player feels a too intense emotion (either negative or positive) the system should catch his/her attention in order to divert him/her from the content that most likely triggered the too intense (according to the threshold) emotion.

Moreover, the player's emotional state can change the multimodal rendering of virtual objects inside the virtual environment. In this way users could be guided along well-defined emotional and informative paths.

4 SYSTEM'S DESIGN AND IMPLEMENTATION

The system design will benefit of earlier work on an emotion recognition system. During this phase, the architecture of the system and the techniques to solve the different problems related to its goal will be identified and justified. Since the ability to recognize emotions depends on how well the EEG features can be mapped onto the chosen emotion representation, two important topics will be investigated (1) the optimal emotion representation for the application at hand and (2) the way to build correspondences between EEG data and emotions. Different types of applications need different representations of the emotions of interest (basic emotions vs. dimensional model of emotions).

The research will deal with three main tasks: data acquisition, data processing and classification and test on real-time applications.

(a) *Data acquisition*: this task will investigate a proper way to acquire EEG data about the brain activity. The selected set-up must contain relevant information to evaluate the emotional state of the subject. We will use data from public database (e.g. IADS, IAPS, [32], [33]) as a reference for the development of algorithms and will integrate them with data acquired using a commercial device whose performance on the application at hand will be checked and experimentally assessed.

(b) *Data processing and classification:* this task will deal with the data processing needed to improve the quality of the data and with the analysis required to extract relevant features and to correlate them to the emotions of interest. The data processing will be mainly oriented to the reduction of noise and of the artifacts while preserving the significant signal available in the data: beside filtering, techniques to detect, measure and remove the regular part of the data will be considered to emphasize the emotional variations. The feature selection will aim at reducing the dimensionality of the data and at make more explicit the information of interest for recognizing the emotional state of the users. In this part of the activity the use of standard databases will be fruitful providing well-defined and reliable examples. The features identified in this phase will therefore be correlated to the emotions of interest: Fourier coefficients or stroke analysis are examples of characteristics of the data that can make more explicit the information contents of the EEG measures. Supervised learning approaches will be applied to automatically extract the mapping between features and emotions. The expected result is a process that can associate the proper emotions to the EEG patterns exhibited by the users. A first validation of the approach using standard measures for learning systems will be done.

(c) *Test on real-time applications*: this final task will use the methodologies and techniques identified in the previous task to develop a system to recognize the emotional state of the user in real-time. Such a system will be integrated inside an application based on the user interaction with a virtual environment. The emotional reactions of the user will be used to understand the impact of contents and to select the ones best suited to the specific sensitivity of each person. Even the kind of interaction with the contents will be guided by the feedback collected using the emotion recognition system. A new verification of the system will be done: beside a further check of the correct recognition of the user's emotions, a qualitative evaluation of the importance of empathy for the effectiveness of the virtual experience will be done. Further experiments on changing some parameters (such as the set of emotions, the measures and features analyzed, the interval of time of observation) will be done to identify the best working set-up.

5 PRELIMINARY DATA ACQUISITION

The first phase of the research will concern data acquisition and will involve several experiments on different subjects, using a variety of experimental settings. In this preliminary phase, we will use the commercial EEG device Emotiv Systems headset (figure 3, http://www.emotiv.com/), which has been selected on the base of factors such as the placement of its electrodes as well as on matters more properly concerned with data (for example is the problem of format conversion of the data provided by a proprietary device). This headset is able to detect and classifying mental states. This device uses 16 electrodes (see figure 3) and covers the four main regions of the brain. The acquired signal is prepared through noise filtering and artifact removal (using for example techniques such as Independent Component Analysis). The following step is feature extraction to select and identify which parts of the data can be used to reduce the dimensionality of the data and keep the relevant information. Techniques such as PCA [30], Partial Least Squares or the same Independent Component Analysis will be verified to deal with this problem. The last step will be the association between features and mental states, which include emotions such as instantaneous excitement, long-term excitement and boredom/engagement. These different mental states and emotions affect the signal power of one or more of the measured bio-signals.
To increase classification accuracy, the Emotiv System allows each new user to train the system on his/her signals. This is done by a method of calibration of the signatures. The training of the system using the Emotiv headset will involve the selection of a set of emotions of interest over a significant group of subjects whose EEG activity will be measured while they are exposed to emotional stimulation for a few seconds. The pictures and sounds used to induce the desired emotion will be selected from two public databases: from the International Affective Picture System (IAPS) and the International Affective Digitized Sound system (IADS) (see [32], [33]). Because the images from IAPS or the sounds from IADS could induce emotions that are different from the expected ones, each user will be asked to rate his/her own emotional state with respect to a predefined standard system [34].

The acquisition set-up will be further checked by comparing data and consequent analysis and classification with measures provided by more accurate EEG equipments such as those used in the clinical practise. That phase will give more insight into the performance of the commercial device in terms of number and locations of electrodes as far as on the accuracy and reliability of the subsequent analysis.

Figure 3. EEG device Emotiv Systems headset

6 ANALYSIS OF REPRODUCIBILITY OF EEG PATTERNS

In this step, we will focus on the effective use of self-induced emotions in a BCI paradigm (real-time data processing).
In this context, by self-induced emotion we mean emotion induced in the user as an effect of the feedback from the virtual reality environment. To elicit a given emotion in the user, the VRE should be able to make a *plan* by selecting a behavior among several possibilities: The recognition of user's emotions is intended to provide a feedback to the VRE system enabling a continuous and progressive adaptation of the planned stimuli to guide the emotional state toward the desired EEG patterns.
In the VRE about the Shoah, the main goal of the application will be to *maintain alive the dramatic meaning of that experience without disturbing too much the more sensitive users.* To reach this goal the system must be able to acquire and recognize the player's reactions to the presented contents: On the base of the evaluation of his level of interest and on the presence of unpleasant emotions, the system will be able to change the displayed contents and their rendering to keep the emotional engagement at a significant but acceptable level. In this perspective, the detailed and correct interpretation of EEG data and their mapping into a sufficiently analytic representation of emotions provide the necessary tool to guide the user experience along well-defined and effective emotional and informative paths.

7 CONCLUSIONS

Working with the EEG signals is rather complicated due to their high subjectivity, therefore it is very difficult to draw a priori conclusions. Literature describes very different results in terms of performance, obviously depending also on the number of subjects involved in the experimental settings. In order to verify and validate our system we will implement extremely different application scenarios. In this way we aim at evaluate in a more reliable way the level of empathy that could be activated in the users of a wide range of applications. Specially designed questionnaires will enable us to attain a subjective evaluation of

the method. Some simple questions (e.g [35]) about the emotions the player experienced while using the selected application will enable us to easily compare the reported player experience with the one supposed by the system during the interaction. Since an objective measure of the performance of the method is needed too, we aim at building a database of EEG data associated to metadata (such as Fourier coefficients that have been successfully applied to many applications of pattern classification) that can represent a reference for comparing different approaches to the recognition of emotions and for evaluating possible changes of basal EEG features. We are confident to develop a software system that will be able to correctly classify at least 95% of the patterns already in the database: that would be a significant enhancement with respect to the state of the art in the field and a serious base for its practical application in real life applications.

REFERENCES

[1] R.W. Picard. *Affective Computing*. MIT Press, Cambridge (1997).

[2] K. Blocher and R.W. Picard. Affective Social Quest: Emotion Recognition Therapy for Autistic Children. In Socially Intelligent Agents -Creating Relationships with Computers and Robots, K. Dautenhahn, A. Bond, L. Canamero and B. Edmonds (Eds.), Kluwer Academic Publishers, The Netherlands (2002).

[3] D. R. Michael and S. L. Chen. Serious Games: Games That Educate, Train, and Inform. Muska & Lipman/Premier-Trade. (2005).

[4] B.J Fogg. Persuasive technology: Using computers to change what we think and do. Morgan Kaufmann. (2003).

[5] V. Carofiglio, F. de Rosis and N. Novielli. Cognitive Emotion Modeling In Natural Language Communication. In Affective Information Processing. Jianhua Tao (Ed.). Springer. (2010).

[6] M.W. Bhatti, Y. Wang, and L. Guan. A neural network approach for human emotion recognition in speech. In Proceedings of the 2004 International Symposium on Circuits and Systems (ISCAS '04). Vancouver, CA (2004).

[7] R. Cowie, E. Douglas-Cowie, N. Taspatsoulis, G. Votsis, S. Kollias, W. Fellenz, and J.G. Taylor. Emotion recognition in human-computer interaction. Signal Processing Magazine, IEEE, 18:32-80 (2001).

[8] F. Dellaert, T. Polzin, and A. Waibel. Recognizing emotion in speech. In Proceedings of International Conference on Spoken Language Processing ICSLP96, Philadelphia, USA. (1996).

[9] B. Fasel and J. Luettin. Automatic facial expression analysis: A survey. Pattern recognition, 36:148-275, (2003).

[10] M. Pantic and L.J.M. Rothkrantz. Automatic analysis of facial expressions: The state of the art. In IEEE Transactions on Pattern Analysis and Machine Intelligence, 22:1424-1445, (2000).

[11] I. Maglogiannis, D. Vouyioukas and C. Aggelopoulos. Face detection and recognition of natural human emotion using Markov random fields. Personal and Ubiquitous Computing archive, 13:95-101, (2009).

[12] K. Salisbury, F. Conti, and F. Barbagli. Haptic rendering: introductory concepts. In IEEE Computer Graphics and Applications, 2:24-32, (2004).

[13] V.V. Kryssanov, E.W. Cooper, H. Ogawa and I. Kurose. A Computational Model to Relay Emotions with Tactile Stimuli. In 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops, Amsterdam, NL, (2009).

[14] V.V. Kryssanov, S. Kumokawa, I. Goncharenko, and H. Ogawa. Perceiving the Social: A Multi-Agent System to Support Human Navigation in Foreign Communities. International Journal of Software Science and Computational Intelligence, 2:24-37, (2010).

[15] J.N. Bailenson, N. Yee, S. Brave, D. Merget, and D. Koslow. Virtual Interpersonal Touch: Expressing and Recognizing Emotions Through Haptic Devices. Human-Computer Interaction, 22:325-353, (2007).

[16] K.H. Kim, S.W. Bang, and S.R. Kim. Emotion recognition system using short term monitoring of physiological signals. Medical and Biological Engineering and Computing, 42:419 427, (2004).

[17] A. Choppin. EEG-based human interface for disabled individuals: Emotion expression with neural networks. Master's thesis, Tokyo Institute of Technology, Tokyo, Japan, (2000).

[18] M. Murugappan, M. Rizon, R. Nagarajan, S. Yaacob, I. Zunaidi, and D. Hazry. EEG feature extraction for classifying emotions using FCM and FKM. In Proc. Of ACACO. China. (2007).

[19] D.O. Bos. EEG-based Emotion Recognition: the influence of visual and auditory stimuli (online). http://hmi.ewi.utwente.nl/verslagen/capita-selecta/CS-OudeBos-Danny.pdf

[20] J.A. Russell. Core affect and the psychological construction of emotion. Psychological Review, 110:145-172, (2003).

[21] G. Chanel, J. Kronegg, D. Grandjean, and T. Pun. Emotion assessment: Arousal evaluation using EEG's and peripheral physiological signals. Technical Report 05.02, Computer Vision Group, Computing Science Center, University of Geneva, Geneva, Switzerland, (2005).

[22] W. Heller, J.B. Nitschke, and D.L. Lindsay. Neuropsychological correlates of arousal in self-reported emotion. Neuroscience letters, 11:383-402, (1997).

[23] J.R.Wolpaw. Brain-computer interfaces (BCIs) for communication and control: a mini-review. Suppl. Clin. Neurophysiol, 57:607-613, (2004)

[24] M. Kostyunina and M. Kulikov. Frequency characteristics of EEG spectra in the emotions. Neuroscience and Behavioral Physiology, 26:340-343, (1996).

[25] M. Mikhail, K. El-Ayat, R. El Kaliouby, J. Coan, and J.J.B.Allen. Emotion detection using noisy EEG data. In Proceedings of the 1st Augmented Human International Conference (AH '10). New York, NY, USA, 2010

[26] S. Koelstra A.Yazdani, M.Soleymani, C.Mühl, J.Lee, A.Nijholt, T.Pun, T.Ebrahimi, I.Patras. Single Trial Classification of EEG and Peripheral Physiological Signals for Recognition of Emotions Induced by Music Videos. Brain Informatics. 89-100 (2010)

[27] T.P. Jung, C. Humphries, T.W. Lee, S. Makeig, M.J. McKeown, V. Iragui, and T.J. Sejnowski. Extended ICA removes artifacts from electroencephalographic recordings. In Advances in neural information processing systems. M. J. Kearns, S. A. Solla, D. A. Cohn (Eds.), Colorado, USA, (1998).

[28] J.C. Woestenburg, M.N. Verbaten & J.L. Slangen. The removal of the eye-movement artifact from the EEG by regression analysis in the frequency domain. Biological Psychology 16:127-147, (1983).

[29] D.J. McFarland, et al. BCI meeting 2005-Workshop on BCI Signal Processing: Feature extraction and translation, IEEE Trans. Neur. Syst. Rehab. Eng., 14:135-138, (2006).

[30] I.T. Jolliffe. Principal Component Analysis, Series: Springer Series in Statistics, 2nd ed., Springer, NY, (2002).

[31] F. Lotte, M. Congedo, A. Lécuyer, Lamarche, and B. Arnaldi. A review of classification algorithms for EEG-based brain-computer interfaces. Journal of neural engineering, 4:1-13, (2007).

[32] P.J. Lang, M.M. Bradley, and B.N. Cuthbert. International affective picture system (IAPS): Affective ratings of pictures and instruction manual. Technical Report A-6, University of Florida, Gainesville, FL, (2005).

[33] M. M. Bradley and P.J. Lang. *International affective digitized sounds (IADS): Stimuli, instruction manual and affective ratings*. Tech. Rep. No. B-2. The Center for Research in Psychophysiology, University of Florida, Gainesville, FL, (1999).

[34] M. M. Bradley, P. J. Lang. Measuring emotion: The self-assessment manikin and the semantic differential. J Behav Ther Exp Psychiatry, 25:49-59, (1994).

[35] C.L. Lisetti, F. Nasoz. Using noninvasive wearable computers to recognize human emotions from physiological signals. EURASIP Journal on Applied Signal Processing, 11:1672-1687,(2004).

Proceedings of AISB '11: AI & Games
Dimitar Kazakov and George Tsoulas (eds.)
ISBN 978-1-908187-01-7

Published by the Society for the Study of Artificial Intelligence and the Simulation of Behaviour
Printed by the University of York, York, UK