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]*

[1] University of Bath, UK, email: johngrey4296@gmail.com

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 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.

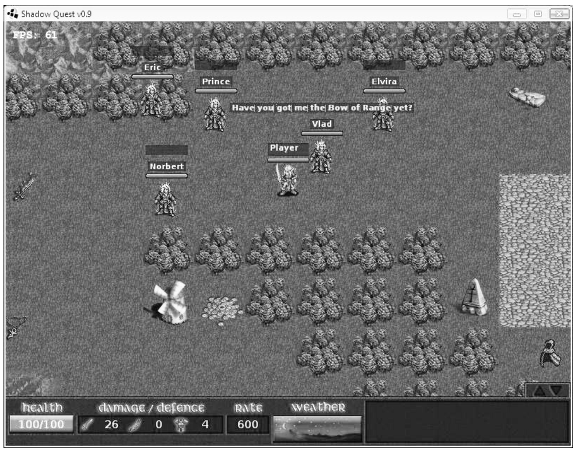

Figure 1. A typical view in the prototype game

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.

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

Figure 2. The BSA Drive Collection

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

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-*

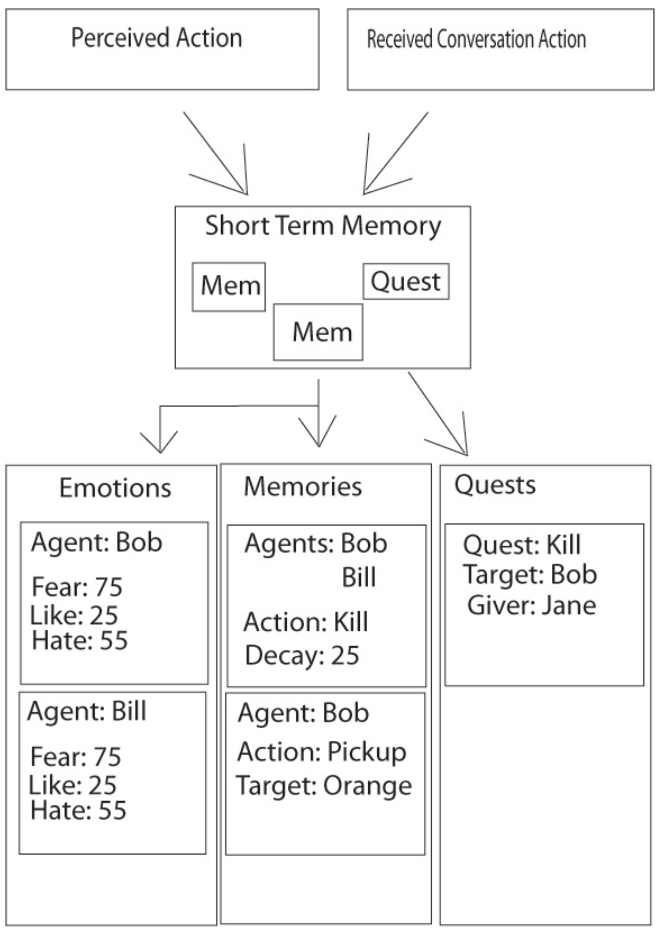

Figure 4. Structure of Agent Memory

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.

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

Figure 5. The *Conversation* Competence

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

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).

$$\uparrow \left\langle \begin{array}{r} \text{(Received or Given Focus or response)} \Rightarrow \textit{Say Goodbye} \\ \text{(Received a focus of something to give)} \Rightarrow \textit{Give focus} \\ \text{(Received a Question)} \Rightarrow \textit{Give appropriate response} \\ \text{(Has Given a Quest to NPC previously)} \Rightarrow \textit{Enquire about Quest} \\ \text{(Has Greeted)} \Rightarrow \textit{Ask a question} \\ () \Rightarrow \textit{Greet} \end{array} \right\rangle \tag{5}$$

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 sidequests 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. Additionally, the proposed design goes some way to address the types of meaning described in section 2.

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).

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 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.

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).