A SCALABLE APPROACH TO BELIEVABLE NON PLAYER CHARACTERS IN MODERN VIDEO GAMES

A. Rankin, G. Acton, and M. Katchabaw
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7
arankin@csd.uwo.ca, gacton@csd.uwo.ca, katchab@csd.uwo.ca

KEYWORDS

Believable decision making, non player characters, performance, scalability, artificial intelligence, video games

ABSTRACT

Recognizing the importance of artificial intelligence to modern video games, considerable efforts have been directed towards creating more believable non player characters in games, complete with personalities, emotions, relationships, and other psychosocial elements. To date, research in the field has primarily focused on the quality of character behaviour, and not on aspects of performance and scalability. This is increasingly a problem as we strive for more complex, dynamic, and realistic behaviour in an ever-growing population of characters in a virtual game world.

In this paper, we explore the performance and scalability of believable non player characters in modern video games. We propose an approach to scalability that is able to improve performance and increase the ability to support a larger quantity of characters without sacrificing the believability or quality of behaviour. We then discuss a prototype implementation of this approach, as well as experiences in experiments and simulations using this prototype. Results to date have been quite positive, and show tremendous potential for continued work in the future.

INTRODUCTION

Artificial intelligence is an increasingly important aspect of modern video games, and can be a determining factor in the overall success of a game (Cass 2002; Champandard 2004; Orkin 2004; Roberts and Isbel 2007; Sweetser 2008). One of the more active areas of research in game artificial intelligence is making more believable characters, also known as non player characters (Baille-de Byl 2004; Funge 2004;Guye-Vuilleme and Thalmann 2001; Lawson 2003; Predinger and Ishizuka 2001; Rizzo et al. 1997), as this can lead to games that are ultimately more immersive, engaging, and entertaining to the player (Dias and Paiva 2005; Livingstone 2006; Sweetser 2008). Doing so, however, requires developers to reach beyond traditional approaches to game artificial intelligence, including finite state machines, rule based systems, and static scripting (Bailey and Katchabaw 2009).

The requirements for character believability tend to be quite steep (Loyall 1997). They include elements such as personality, emotion, self-motivation, social relationships, consistency, the ability to change, and the ability to maintain an "illusion of life", through having goals, reacting and responding to external stimuli, and so on (Loyall 1997).

Computationally, these believable non player characters are orders of magnitude more complex than traditional approaches to character artificial intelligence, and so they introduce the potential for serious performance problems, especially when there are a great number of them inhabiting a game world (Bailey and Katchabaw 2009). This problem is only exacerbated by the computational needs of other game sub-systems, which together put an even larger strain on often limited and over-taxed system resources. Consequently, if we are to truly take advantage of believable characters in modern video games, we must take into consideration issues of performance and scaling with artificial intelligence as it has been done with other aspects of games, such as graphics (Rankin 2009).

To improve performance and scalability of games, techniques generally involve various different kinds of pre-processing of game data and various types of run-time optimizations to reduce the processing load as the game runs. From an artificial intelligence perspective, we can adopt both techniques as well, to optimize both the believable character models used prior to execution and their use in-game at run-time.

This paper introduces and examines a scalable approach to believable non player characters that utilizes several techniques to improve in-game performance. These include dynamic importance calculation, capability scaling or reduction, and pre-emptive, priority-based character scheduling and dispatching. Through utilizing these techniques, we can ensure that scarce computational resources are allocated to characters where they are most needed, and that characters are using the decision-making processes best suited to their current state and importance to the game, while still maintaining believability.

To take advantage of these performance and scalability optimizations, this paper also introduces a new approach to believable non player characters using utility based planning and action selection combined with dynamic, emergent behaviour. To support believability, various psychosocial elements are captured in this approach, including personality, emotions, relationships, roles, beliefs, desires, intentions, and coping. This extends our earlier work in (Bailey and Katchabaw 2008; You and Katchabaw 2010), enabling richer and more compelling non player character behaviour.

The remainder of this paper is organized as follows. We begin by presenting and discussing related work in this area. We then describe our approach to performance and scalability for believable characters, outlining the various optimization techniques and strategies used in our own work. We then discuss our prototype system, including the implementation of both the non player characters and the various performance and scalability enhancements we employed. We then present and discuss our experiences from using this in a variety of experimental scenarios. Finally, we conclude this paper with a summary and a discussion of directions for future work.

RELATED WORK

To date, unfortunately, a literature survey reveals relatively little in performance and scalability specifically intended for affective artificial intelligence systems for games. That said, there is important and relevant research to note, as discussed in this section below.

Affective approaches designed for improved believability, such as the work in (Bates et al. 1994; Gratch and Marsella 2004; Imbert et al. 2005; Reilly and Bates 1992), unfortunately, make little mention of performance or scalability issues. Most frequently, this work focuses on the creation and simulation of a single character without considering the issues that arise in placing this character within a living game world, with numerous other inhabitants. While some work examines interacting characters, there are no measures of performance or scalability provided, and no mention of allocating computing resources to the characters (Rankin 2009).

Looking at other game artificial intelligence research, some attention has been given to issues of performance. The work in (Wright and Marshall 2000) is notable for providing a flexible general-purpose framework for artificial intelligence processing in games. While taking an “egocentric” approach, this work, unfortunately, provides no specific insight or performance optimizations for characters with psychosocial systems for believable behaviour, as in our current work. Work towards scalable crowd behaviour, such as (Pettré et al. 2006; Sung et al. 2004), is also relevant, but tends to focus on the believable simulation of groups, as opposed to the believable simulation of a large number of individuals, which can be very different problems.

We can also view believable non player characters as a form of agent, and games themselves to therefore be a kind of multi-agent system (Rankin 2009). Performance and scaling in multi-agent systems is still not a thoroughly explored area of research, however, but there are works of interest that borrow heavily from such diverse areas as operations research (Baker 1998; Haupt 1989; Holthaus and Rajendran 1997), distributed computing (Rana and Stout 2000), and operating system process scheduling (Ramamritham and Stankovic, 1994; Tanenbaum 2008). While not directly involving games, this work provides insight into the issues at hand, as well as potential solutions relevant to this problem domain.

While there is a lack of literature focused on performance and scalability in affective artificial intelligence for video games, there are tremendous opportunities for research in this area. Fortunately, insight and experience is available to be borrowed from other aspects of gaming research, as well as other disciplines, and can applied to this problem in a rather unique and innovative fashion, as we have done as described in this paper.

GENERAL APPROACH

As discussed earlier, we take advantage of several techniques in unison to improve performance and scalability of non player characters in video games. In this section, we present these various techniques.

Overview

We begin by recognizing that not every non player character in a game has equal importance to the player and the game at any point in time. Some characters are the focus of attention, are engaged in significant activities, or are otherwise critical to what is unfolding in the game. Others are of less importance, and their activities will largely, if not completely, go unnoticed to the player. As the player moves through the game world and plays the game, the importance of the various characters changes, with some becoming more important and others less so.

Furthermore, we note that characters that are not visible to the player, or otherwise not important to the player, do not need the same richness, detail, and fidelity in their behaviour as those that are the focus of attention, or are otherwise important, to maintain believability. In those cases, a game can safely do less with those characters, in many cases significantly so, with no perceptible difference to the player’s experience. For example, suppose we have two non player characters in a game that are hungry; one, Alice, is in the same room as the player, while the other, Bob, is in a different locale far across the game world, outside of the player’s ability to sense. To maintain believability, Alice must recognize her hunger, formulate a plan to satisfy her hunger considering her current psychosocial and physiological state and surroundings, and then execute this plan. After all, the player is in the same room and is able to see and hear everything she does; the slightest out of place action could sacrifice believability and adversely affect the player’s experience. On the other hand, Bob would not need to do anything except simply continue to live to maintain believability. In the player’s absence, it is likely reasonable to believe that he could provide himself with the necessities of life, without actually needing to do anything about it. In fact, the game would not even need to track Bob’s hunger level to maintain the same level of believability. (That said, Bob would still need to progress in his life in the absence of the player, as it is likewise unreasonable to believe that he would do absolutely nothing at all when the player is not around. How this should be done to maintain believability is discussed below.)

With this in mind, we can safely scale or reduce the capabilities of a non player character according to their current visibility and importance to the player while still maintaining believability across all characters in the game, as shown in Figure 1. In doing so, we can achieve tremendous performance savings in characters with reduced capabilities as they require significantly less processing and this processing is far less complicated than characters operating at full capacity. This, in turn, allows the game to support a much larger number of characters without requiring additional resources to be dedicated to artificial intelligence processing.

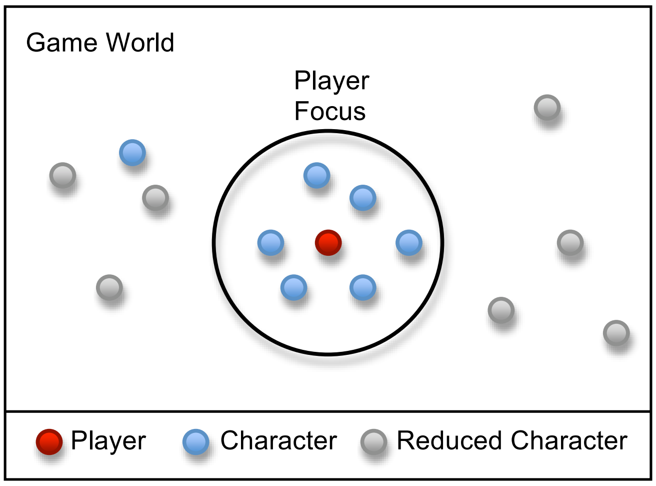

Figure 1: Characters in the Game World

To support this approach, each non player character in a game is a separately schedulable and executable entity, enabling computational resources to be allocated by a scheduling and dispatching subsystem on a character-by-character basis according to their importance. Furthermore, each character has access to a range of decision-making and processing algorithms, allowing their capabilities to be scaled or reduced based on their importance. Lastly, character importance is calculated and updated regularly based on a variety of factors to ensure that scheduling, dispatching, and capability adjustment are all carried out using the best available information. Each of these activities is discussed in further detail in the sections that follow below.

Scheduling and Dispatching of Characters

To allocate computational resources to non player characters in a game, a scheduling and dispatching subsystem is added to the game. The goal of this subsystem is rather simple—choosing the most appropriate characters to run at each game tick or frame of game execution.

Producing an optimal schedule for each tick is itself a computationally expensive task, and so we take a simpler, more heuristic approach using a system of priorities assigned to each non player character in the game. (Priorities are derived from the perceived importance of the characters, as discussed earlier in the paper, and below in further detail.) With this in mind, the most appropriate characters to execute in any tick are simply those with the highest priorities. Resource starvation is averted in characters of low importance through an aging mechanism built into the calculation of priorities.

Each game tick, the *x* characters with highest priorities are selected to run, where *x* is a positive integer configurable and tunable at run-time, based on a number of factors including the number of available processor cores, the workload being generated by other game subsystems, and so on. Each of the *x* selected characters is allocated one or more update cycles, depending on their relative importance. Each update cycle allows a character to execute for a period of time. This execution can be pre-empted, and does not necessarily allow the character to complete the task on which it was working. If the task is not completed when the character's update cycle ends, the character is paused and its priority is adjusted to reflect the work in progress.

Depending on the number of characters in the game, the priority system could be realized as either a single list sorted by priority, or a series of priority queues. While the mechanics of each approach is different, they can both deliver the same pre-emptive, priority-driven scheduling and dispatching of non player characters in a game.

Capability Scaling or Reduction

As discussed earlier, not every non player character in a game needs the highest levels of richness, detail, and fidelity in their behaviour in order to be perceived by the player as believable. Indeed, some characters could function believably with greatly reduced capabilities, depending on the state of the player, the game, and the various characters within it.

To better understand how capability scaling or reduction can be accomplished, we first outline how a believable non player character must function in the first place. We assume at least a basic set of psychosocial/cognitive processing elements, as without some form of these elements, it is extremely difficult for a character to deliver behaviour that could be truly considered believable (Acton 2009). This results in the high level character decision-making process shown in Figure 2. Each of the stages from this process is discussed below.

- **Event:** A notification of activity in the game world, whether it is from other characters, the world itself, or simply the passage of time.

- **Appraisal:** The event is examined to see if it is of interest to the character, and if so, does the event or its consequences have sufficient importance to the character to warrant further consideration. Deadlines for action/reaction may also be set, depending on the event.

- **Coping:** The character reflects on the event and updates its internal psychosocial and physiological state based on the event. This adjusts the character's mood, emotional memory, and so on.

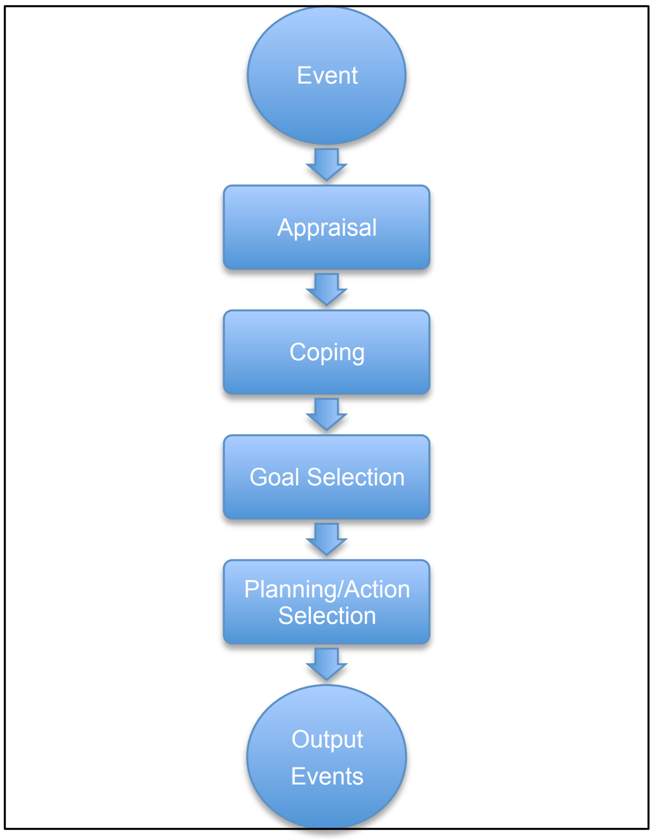

Figure 2: High Level Character Decision-Making Process

- **Goal Selection:** Given the current state of the character, its surroundings, and recent events, the character determines the goals that it should be trying to meet. Goals might be immediate, short-term, medium-term, and long-term.

- **Planning/Action Selection:** Considering the goals in place, a plan is created to achieve the goals with the desired results and acceptable side effects and consequences. If a plan already exists and is in process, it might be revised to reflect changes in goals, character state, and so on. With the plan in mind, appropriate actions are selected to have the plan implemented. Note that if events indicate a quick reaction is required, the current plan may be temporarily bypassed or abandoned to carry out alternate actions. (This may be the case, for example, to ensure self-preservation.)

- **Output Events:** Based on the actions selected, new events are generated and propagated into the game accordingly.

When this approach to character decision-making is considered, scaling a non player character can be accomplished in several ways. Generally, these methods can be grouped into either data reductions or processing reductions.

The purpose of data reductions is to limit the amount of information that is used in the various stages of the decision-making process. When done properly, this can greatly collapse decision spaces into smaller problems, making them far more efficient and less costly to work with. With care, these reductions can be carried out with little to no perceivable change in character believability. Data reductions can themselves take many forms.

One such form is a model or state reduction. As discussed earlier, in our approach, every character has a psychosocial model associated with them, defining their personalities, emotions, relationships, roles, values, and so on. A fully populated model could have a character's state defined by dozens of traits, all of which need to be maintained, and all of which could affect decisions made by the character. While rich, expressive, and useful during design to fully define a character, this can be very expensive computationally to use at run-time, as these factors must be consulted during appraisal, coping, goal selection, and planning/action selection. A careful reduction of this design model to a few key traits can produce a run-time model that is orders of magnitude more efficient, and far less costly to use within a game. This process is shown in Figure 3.

Another form of data reduction is event reduction. In this case, the number of events used to trigger decision-making or provide information during decision-making is limited, by reducing the types of events of interest, the importance of the various events or their consequences, or the frequency of their reporting to the character. This results in either fewer events reaching the character, fewer events moving past the appraisal stage (as the events are now deemed irrelevant, unimportant or inconsequential), or simpler decisions throughout the various stages of processing with fewer variables and factors to consider. All of these alternatives have the potential to improve performance substantially.

Processing reductions, on the other hand, generally involve altering a character's decision-making processes by changing algorithms or omitting aspects or complete stages of decision-making to improve performance. Again, if done with care, these performance improvements can be achieved without sacrificing believability. Possible processing reductions include the following.

- **Use of Defaults.** To accelerate the various stages of decision-making, default strategies can be used instead of analyzing and developing them dynamically on demand. For coping, events could have default impacts on character state, instead of determining this impact from the current state and other factors. For goal selection, characters could be assigned default goals to meet instead of developing them from scratch. For planning and action selection, default plans could be provided for each possible goal, complete with prescribed actions so that they do not need to be developed at run-time. While the defaults used will not do as well at reflecting the current state of characters or the game, the performance savings can be substantial even if this approach is used only for out of focus characters, or those that are unimportant.

- **Use of Randomization.** Much like through the use of defaults, randomization can be used to replace various aspects of behaviour, although doing so likely makes sense only with unimportant characters in the game. It

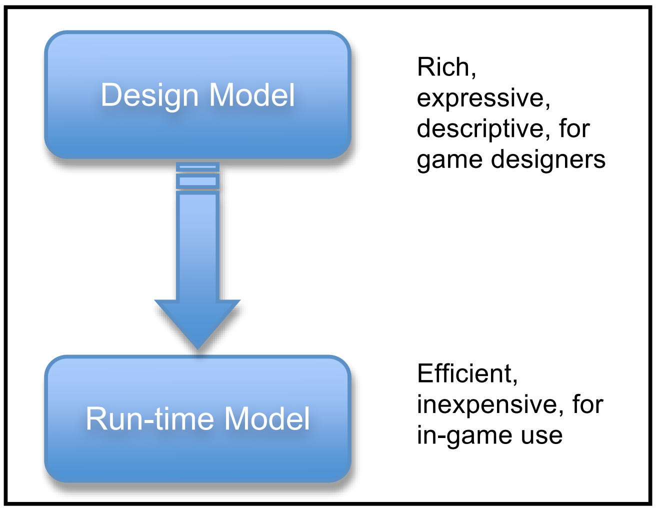
Design Model
Rich, expressive, descriptive, for game designers
Run-time Model
Efficient, inexpensive, for in-game use

Figure 3: Character Model Reduction

is likely also wise to constrain randomness to ensure that characters are still acting believably. This can be accomplished in a variety of ways, such as constraining randomness to choose from a pre-defined set of defaults, or by permitting some level of initial processing develop sensible options that are chosen from randomly, instead of using a more expensive algorithm to make a more optimal choice.

- **Streamlining of Coping.** Integrating the impact of events into a character's internal state can be expensive, especially with a complex psychosocial model in use. To improve performance we can perform coping only in response to a limited set of critical events when characters are outside of the focus of the player. While this will result in characters whose moods and emotions change only in extreme circumstances, the player will be largely unaware of this.

- **Disabling of Goal Changing.** Whenever a character changes goals, any existing plan must be discarded and a new plan must be formulated, which can be quite expensive. To improve performance, characters can be prevented from modifying their goals while a plan is executing to avoid re-planning, unless very exceptional circumstances arise. While this will result in characters sticking with plans when they should likely be changed, this should not have too large an impact on their believability, provided that they are outside the focus of the player.

- **Automatic Achievement of Goals.** As mentioned above, planning and action selection can be computationally expensive tasks. When a character is outside the focus of the player, these tasks can be avoided entirely by simply allowing the character's goals to be achieved automatically. After all, if a goal is achievable by the character, and the player is not present to witness the goal actually being achieved, planning and action selection and execution are not required for the player to believe what has happened. It is important to ensure, however, that the goal is likely to be achieved by the character, and that the time required to meet the goal is properly taken into account; otherwise believability may be inadvertently sacrificed.

- **Disabling of All Processing.** If a character is relatively unimportant and is someone with whom the player has had no prior personal contact or knowledge thereof, it is possible to disable all, or nearly all, decision-making in the character. After all, the player would have little to no expectation of the character's mood, goals, or actions and so it is believable for the character to be in their initial state when first encountered by the player. The player has no way of knowing that the character was largely inactive up until when they entered the focus of the player.

  This strategy might also be applicable to important characters or characters that have been previously encountered, provided that they are out of the player's focus and will remain out of their focus until the next break in the game, such as a cut-scene, level transition, and so on. If important events are recorded, their effects on the character, as well as the character's goals, plans, and actions can all be simulated during the break, so that they are up-to-date when the player next encounters them.

When we combine the various forms of data and processing reductions together, we have great flexibility in the amount of capability scaling or reduction available to a game. If taken too far, this can eventually impact the believability of the game, but if done with care in an intelligent fashion, we can achieve tremendous performance savings with little to no effect on the believability perceived by the player.

Character Importance and Priority Calculation

The process of scheduling and dispatching, as well as the process of capability scaling or reduction both use a measure of a non player character's importance as a factor that ultimately affects both resource allocation and performance. Capability scaling or reduction uses a measure of importance directly, adjusting the capabilities of a character accordingly. Scheduling and dispatching use importance in the form of a priority in determining which characters are run in each game tick. Below, we examine how each measure is computed.

The importance of a non player character is determined by a collection of factors, with one calculating the importance, *i*, of a character as:

$$i = (\alpha f + \beta d + \gamma r + \delta c) / 4$$

where α, β, γ, and δ are weights between 0 and 1.0 to tune and balance the equation. The factor f is a measure of player focus on the character, which takes into consideration the distance between the player and the character, whether the character is within range of the player's senses, and the strength of relationships between the player and the character. The factor d is a designer-imposed measure of importance of the character, usually with respect to the story of the game. The factor r is a measure of importance defined by the currently active roles of the character in question, as some roles in the game are inherently more important than others. Lastly, the factor *c* is a measure of importance that comes from character interactions. If a given character is involved with other, more important characters, their own importance might require a boost to put the characters on more equal footing. (For example, if the two are fighting each other, an inherently more important character could enjoy an unfair advantage over less important characters because the seemingly more important character has more capabilities and better access to computational resources.) The factors *f*, *d*, *r*, and *c* all range between 0 and 1.0 and so after scaling, the importance of a character also lies between 0 and 1.0.

With this in mind, the priority, *pri,* of a non player character can be computed as follows:

$$pri = \varepsilon i + \zeta s - \eta rc + \theta p$$

where ε, ζ, η, and θ are also weights between 0 and 1.0 to tune and balance the equation. (Carefully setting of these weights can also result in various scheduling policies, such as fair, least-slack-time, and so on (Rankin, 2009).) The factor *i* is the importance of the character as defined above. The factor *s* is a starvation factor that increases at a certain rate for each game tick that the character is not run; by doing this, even an unimportant character's priority will eventually exceed the most important character, allowing the unimportant character to run and avoid starvation. The factor *rc* is a run counter used to ensure that a single character is not over-allocated update cycles despite its importance. Lastly, the factor *p* is a progress measure that approaches 1.0 as the character approaches completion of its task at hand, to allow scheduling to clear out near-complete tasks from characters. All of factors *i*, *s*, *rc*, and *p* are normalized to between 0 and 1.0.

If desired, we can add a fifth factor to priority calculations to reflect the amount of capability reduction being applied to a particular non player character. Doing so may be reasonable since a character with its capabilities reduced by data or processing reductions will require fewer computational resources and therefore can cope with its schedule being reduced as well. Ordinarily, this would be accomplished using the importance factor *i,* as a low importance would trigger both capability and schedule reduction simultaneously. If importance and capability reduction were not so closely linked, a separate factor indicating reduction would then be necessary. (This can occur, for example, when there is a very large number of non player characters needing to be managed; in such a case, even the capabilities of fairly important characters would need reduction despite their importance in order to maintain game performance at an acceptable level.)

PROTOTYPE IMPLEMENTATION

As a proof of concept, we started with the development of a foundation framework for believable non player characters. This foundation was designed to be extended with modules for character scheduling and dispatching, as well as capability scaling or reduction, as discussed earlier in this paper. This prototype was developed for the various Microsoft Windows platforms using a combination of managed and unmanaged C++ using Microsoft Visual Studio 2008 as a development environment.

At the core of this prototype is a character system based on the high-level decision making process shown in Figure 2. Character state is composed of a psychosocial model integrating aspects of personality, emotions, relationships, roles, beliefs, desires, intentions, and coping. The personality model is derived from Reiss' theory of basic desires (Reiss 2004), as this approach presents personality in a fashion well suited to goal selection and consequences of actions. The emotion model selected is based on Ekman's universal emotion model (Ekman et al. 1972), a veritable standard in this area. Roles were developed using role theory from (Guye-Vuilleme and Thalmann 2001) as a basis. Aspects of appraisal and coping were adapted from (Gratch and Marsella 2004), while goal selection and planning/action selection were driven by standard utility based processes. A further discussion of the non player character system used as a foundation in this work can be found in (Acton 2009).

A scheduler and dispatcher module was added to the prototype to allocate computational resources to non player characters from the character system. This was based on a simple serial sort and search algorithm to determine the next characters to run based on priorities as described in the previous section. Capability scaling or reduction was implemented with multiple levels of reduction. A character running with full capabilities uses the complete character system described above. The first level of reduction uses rudimentary partial planning in which planning is carried out over several update cycles, with actions selected from partial plans in earlier update cycles while the current cycle continues to refine the plan. The second level of reduction uses full appraisal, coping, and goal selection capabilities, but then uses default plans and actions associated with goals selected, instead of carrying out a full planning/action selection stage. Finally, the third level of reduction uses randomization to select a goal and select a plan and actions capable of achieving this goal. While this is not the most realistic of approaches, it can still be appropriate for non critical characters in the game.

To assess the operation and performance of the prototype system, detailed logs are collected. These logs show all non player character state and activity at each tick of simulated game time, and contain performance information related to the scheduling and capability level of each character in the system. These logs are valuable to experimentation with the prototype system, as discussed at length in the next section of this paper.

RESULTS AND EXPERIENCES TO DATE

To assess the effectiveness of our approach to scalable believable non player characters, we conducted a series of experiments using our prototype system. In this section, we discuss highlights of our results. A complete presentation of experimental results and experiences can be found in (Rankin 2009).

Experimental Environment and Configuration

All experimentation was executed on an Intel Core 2 Duo system with a clock speed of 2.0Ghz and 4.0GB of RAM. The 64-bit variant of Windows Vista was used as the system's operating system. This configuration provided more than enough power for the experimentation we conducted.

The prototype system was configured to use the psychosocial model described in the previous section, with characters having access to 4 roles, 5 goals, and 8 actions during processing. While a typical game would have more possibilities open to its characters, this configuration on its own was sufficient to demonstrate the effectiveness of our approach. Time in the system was simulated so that 4 characters could run each game tick, there were 30 milliseconds between game ticks, and each action consumed one tick for execution. While actions would ordinarily take longer and have varied lengths in reality, this accelerated experiments and simplified analyses, as it was easier to confirm that factors such as importance were being properly handled by the system. Lastly, for simplicity and balancing, all weights used in calculating importance and priority were set to 1.0, except during starvation experiments. It is possible that better (or worse) results could be obtained through the fine-tuning of these weights. Additional experiments are currently under way, and others are planned to explore these and other issues more fully in the future.

Initial Experiments

Prior to more rigorous experimentation, initial testing was conducted to assess the basic operation of our prototype system. From this, we were able to verify:

- Equal fixed importance and priority resulted in an even distribution of resources to characters and equal opportunity for execution.
- Increased importance and priority translated into an increase in resource allocations to characters and a corresponding increase in execution time.
- Starvation of characters with low importance was effectively prevented by our approach to scheduling, and would be a serious issue if these measures were disabled or not provided in the first place.
- Characters with reduced capabilities required fewer resources to execute than characters with full capabilities intact.
- The prototype system could handle several characters of varying importance well, adjusting scheduling and capabilities accordingly without difficulty.

While these tests verified the correct operation of the prototype system, we still needed to assess the improved performance and scalability enabled by our approach. This is accomplished through experimentation outlined in the next section.

Stress Testing

To assess performance and scaling improvements, we used the prototype system to manage hundred of characters simultaneously. In these experiments, we executed three scenarios with 100, 200, and 800 characters respectively. Each scenario was itself run three times, once with all characters at full capability, once with all characters at the second level of reduction (as described in the previous section), and once with all characters at the third level of reduction.

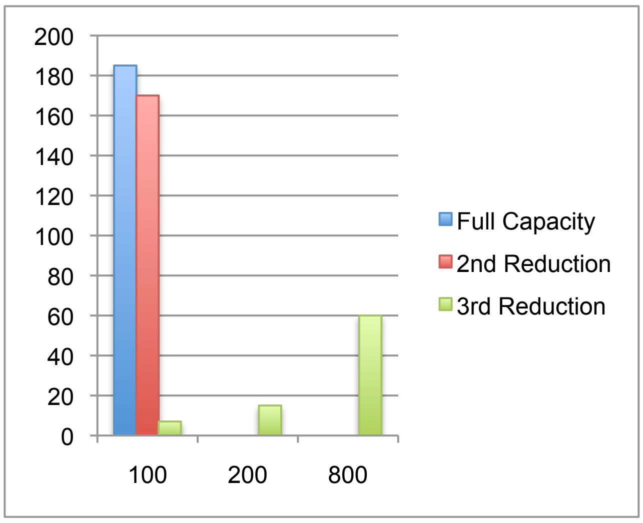

Figure 4: Character Stress Testing

Results from this experimentation are shown in Figure 4, measuring the time to completion of 200 game ticks in seconds. With 100 characters executing, performance under full capacity suffered greatly. There was a slight improvement under the second reduction, but performance was still unacceptable. (The small difference achieved with this reduction is because even under full capacity, the planner is somewhat primitive and incomplete. With a complete planner, full capacity characters would suffer worse, and there would be a bigger improvement achieved through this reduction.) With the third reduction, performance improvements were substantial. As the number of characters increased, only the third reduction characters were able to complete. While their time to completion increased, performance was still improved dramatically through this reduction.

It is important to note that while full capacity and slightly reduced characters had performance issues, the system would never be expected to support this many at a time. Through dynamic adjustments to capabilities, only a few would run at these capability levels at a time, depending on the game, with the others reduced further. This experiment was to solely demonstrate performance improvements through our approach.

From these results, we can see great improvements in the performance delivered by our approach to scalable believable non player characters. We are able to deliver a collection of characters that can adapt to various computational requirements through proper scheduling and capability adjustment.

CONCLUSIONS AND FUTURE WORK

This paper introduced a scalable approach to believable non player characters in modern video games. Through a combination of importance determination, prioritized scheduling and dispatching, and capability scaling or reduction, we can adjust the level of functioning of characters to adhere to computational constraints while maintaining believability. Experimental results with our prototype system have been both positive and quite promising.

In the future, there are many avenues for continued work. We plan to continue experimentation and further tune, scale, and explore the capabilities of our approach. We will continue the development of our prototype approach, adding both more functionality to our characters and additional capability reduction techniques. Lastly, we plan to embed our approach within a complete game or game engine to fully assess both its performance and its sustained believability through extensive user testing.

REFERENCES

Acton, G. 2009. *Playing the Role: Towards an Action Selection Architecture for Believable Behaviour in Non Player Characters and Interactive Agents*. Masters Thesis, Department of Computer Science, The University of Western Ontario.

Bailey, C. and Katchabaw, M. 2008. "An Emergent Framework For Realistic Psychosocial Behaviour In Non Player Characters". *Proceedings of FuturePlay 2008*. (Toronto, Canada, November 2008.)

Baille-de Byl, P. 2004. *Programming Believable Characters In Games*. Charles River Media.

Baker, A. 1998. "A Survey of Factory Control Algorithms that Can Be Implemented in a Multi-agent Heterarchy: Dispatching, Scheduling, and Pull". *Journal of Manufacturing Systems*, *Volume 17, Number 4*. Elsevier.

Cass, S. 2002. "Mind Games". *Appeared in IEEE Spectrum*, 39(12).

Bates J., Loyall A., and Reilly W. 1994. "An Architecture for Action, Emotion, and Social Behavior." *Lecture Notes in Computer Science, 830:55-68*.

Champandard, A. 2004. *AI Game Development: Synthetic Creatures with Learning and Reactive Behaviours*. New Riders.

Dias, J. and Paiva, A. 2005. "Feeling and Reasoning: A Computational Model for Emotional Characters". *Lecture Notes in Computer Science, 3808*.

Ekman, P., Friesen, W., and Ellsworth, P. 1972. Emotion in the Human Face: Guidelines for Research and an Integration of Findings. Pergamon.

Funge, J.D. 2004. *Artificial Intelligence For Computer Games*. A K Peters.

Gratch, J. and Marsella, S. 2004. "A Domain-Independent Framework for Modeling Emotion". *Cognitive Systems Research, 5(4)*. (December 2004).

Guye-Vuilleme, A., and Thalmann, D. 2001. "A High-level Architecture For Believable Social Agents". *VR Journal, 5*.

Haupt, R. 1989. "A Survey of Priority Rule-based Scheduling". *OR Spectrum, 11(1)*.

Holthaus, O. and Rajendran, C. 1997. "New Dispatching Rules for Scheduling in a Job Shop - An Experimental Study". *The International Journal of Advanced Manufacturing Technology, 13(2)*.

Imbert, R., De Antonio, A., and De Informatica, F. 2005. "COGNITIVA: A Context Independent Cognitive Architecture for Agents Combining Rational and Emotional Behaviours". *In 5th. WSEAS Int. Conf. on Multimedia, Internet and Video Technologies*. (Corfu, Greece, 2005.)

Lawson, G. 2003. "Stop Relying On Cognitive Science In Game Design - Use Social Science". *Accessed June 2010 from http://www.gamasutra.com/php-bin/letter_display.php?letter_id=647.*

Livingstone, D. 2006. "Turing's Test And Believable AI In Games". *Computers in Entertainment (CIE), 4(1).*

Loyall, A. 1997. *Believable Agents: Building Interactive Personalities*. PhD Dissertation, Stanford University.

Orkin, J. 2004. "Symbolic Representation of Game World State: Toward Real-Time Planning in Games". *In Proceedings of the AAAI Workshop on Challenges in Game Artificial Intelligence*.

Pettré, J., de Heras Ciechomski, P., Maïm, J., Yersin, B., Laumond, J., and Thalmann, D. 2006. "Real-Time Navigating Crowds: Scalable Simulation and Rendering". *Computer Animation and Virtual World (CAVW) Journal - CASA 2006 Special Issue.*

Prendinger, H., and Ishizuka, M. 2001. "Social Role Awareness In Animated Agents". *Proceedings of the International Conference on Autonomous Agents.*

Ramamritham, K. and Stankovic, J. 1994. "Scheduling Algorithms and Operating Systems Support for Real-time Systems". *Proceedings of the IEEE, 82(1).*

Rana, O. and Stout, K. 2000. "What is Scalability in Multi-agent Systems?" *Proceedings of the Fourth International Conference on Autonomous Agents*. (Catalonia, Spain, June 2000.)

Rankin, A. 2009. *Scalability and Performance of Affective Multi-Agent Systems*. Masters Thesis, Department of Computer Science, The University of Western Ontario.

Reilly, W. and Bates, J. 1992. *Building Emotional Agents*. Technical Report CMU-CS-92-143, School of Computer Science, Carnegie Mellon University. (Pittsburgh, PA, May 1992.)

Reiss, S. 2004. "Multifaceted Nature of Intrinsic Motivation: The Theory of 16 Basic Desires". *Review of General Psychology*, 8(3).

Rizzo, P., Veloso, M., Miceli, M., and Cesta, A. 1997. "Personality-driven Social Behavior In Believable Agents". *Proceedings of the AAAI Fall Symposium on Socially Intelligent Agents*.

Roberts, D. and Isbell, C. 2007. "Desiderata For Managers Of Interactive Experiences: A Survey Of Recent Advances In Drama Management". *In the Proceedings of the First Workshop on Agent-Based Systems for Human Learning and Entertainment.*

Sung, M., Gleicher, M., and Chenney, S. 2004. "Scalable Behaviors for Crowd Simulation". *Computer Graphics Forum, Vol. 23, No. 3.*

Sweetser, P. 2008. *Emergence in Games*. Charles River Media, Game Development Series.

Tanenbaum, A. 2008. *Modern Operating Systems*, Third Edition. Prentice Hall.

Wright, I. and Marshall, J. 2000, "Egocentric AI Processing for Computer Entertainment: A Real-time Process Manager for Games". *Proceedings of the First International Conference on Intelligent Games and Simulation (GAME-ON 2000).* (London, United Kingdom, November 2000.)

You J. and Katchabaw, M. "A Flexible Multi-Model Approach to Psychosocial Integration in Non Player Characters in Modern Video Games". *Proceedings of FuturePlay 2010.* (Vancouver, Canada, May 2010.)