

Flexible and Purposeful NPC Behaviors using Real-Time Genetic Control

Talib S. Hussain, *Member, IEEE* and Gordon Vidaver

Abstract—There is an increasing need in modern computer games for non-player characters (NPCs) with robust behaviors that achieve game objectives while appearing flexible and believable to the human players interacting with those NPCs. Evolutionary approaches to game artificial intelligence (game AI) have produced successful results for complex game winning strategies, realistic behavior patterns for groups of simulated entities, and more. However, there has been relatively little effort on evolutionary techniques for producing rich NPC behaviors for interaction with human players. To explore whether evolutionary mechanisms can support real-time control of NPCs to produce flexible and purposeful behavior, we present our initial efforts at integrating a genetic algorithm based robotic controller with an off-the-shelf game to control one or more NPCs dynamically. We describe the integration effort and our initial observations, and discuss our plan for achieving richer NPC control and for performing more detailed analysis of the behaviors of the resulting NPCs.

I. INTRODUCTION

IN modern computer games, human players often interact with non-player characters (NPCs) within a simulated environment. Unlike traditional computer games, where there are usually clear rules of the game, limited roles and clear objectives, many modern computer games are increasingly providing a means for entertaining the human users in an open-ended fashion. Thus, it is not the rules of the game that determine success, but rather the ability of the game and its simulated entities to attract and keep the attention of the players. As such, there is an increasing need for control mechanisms for NPCs, or “game artificial intelligence (game AI)”, that produce believable and useful behaviors without detracting from the immersive experience. A common failing of traditional techniques for game AI is that the NPCs have a limited range of behaviors and apply them in limited ways. The result is that a human user can (usually) quickly determine the limits of the NPC. While this may allow the users to exploit those limits in pursuit of their current game objectives, it also produces a perception that NPCs are predictable and not interesting. Further, when NPC-to-NPC interactions are involved, the limitations of those interactions are even more marked to a human observer. For the games of the future, there will be a strong need for NPC control mechanisms that produce flexible

behaviors under a variety of circumstances, but still allow the NPCs to play specific roles and achieve specific objectives. Additionally, recent efforts in developing game-based training systems have revealed the need for flexible NPC behaviors that support robust and realistic training events.

The pursuit of better and more believable NPC behavior has explored a number of approaches, including:

- (a) Off-line automated techniques for behavior optimization
- (b) On-line (real-time) automated techniques to adapt actions and/or learn better responses
- (c) Real-time control mechanisms to produce realistic behaviors of groups of simulated entities
- (d) Real-time control via explicit behavior models for varied yet purposeful behaviors

Evolutionary computation techniques have been used for (a-c), but rarely for (d). Recently, the use of (non-evolutionary) human behavior models for controlling NPCs has been explored [1-3], and it has been demonstrated that rich interactions may result, both between human players and the NPCs, as well as among the NPCs themselves [4]. We decided to investigate whether an evolutionary technique could be used in an analogous way to produce real-time, flexible and purposeful NPC behaviors.

In [5,6], we introduced a prototype evolutionary control mechanism for unmanned ground vehicles that was designed to produce flexible tactical behaviors under a variety of conditions. The Advocates and Critics for Tactical Behaviors (ACTB) model incorporates specific tactical behaviors, random behaviors and multiple internal operating modes, termed attitudes, that influence which and how behaviors are expressed. Thus, at a high-level, it has some similarities to a human behavior model. This paper presents our efforts to use ACTB to control an NPC within a modern computer game.

We review related research, present a brief overview of the ACTB algorithm and describe our efforts to use ACTB to control an NPC in the game *Neverwinter Nights* (Trademark of Wizards of the Coast, Inc.). Since the perception of behaviors is subjective, and these are our initial efforts, we present our results in terms of subjective observations and lessons learned in our integration effort. We conclude with our planned next steps and our thoughts on evaluating real-time genetic NPC controllers.

II. BACKGROUND

Game AI has been developed for many different

T.S. Hussain is with the Intelligent Distributed Computing department at BBN Technologies, Cambridge, MA, 02128 USA (phone: 617-873-6861; fax: 617-873-4328; e-mail: thussain@bbn.com).

G. Vidaver is with the Intelligent Distributed Computing department at BBN Technologies, Cambridge, MA, 02128 USA (e-mail: gvidaver@bbn.com).

categories of computer games, including board games, robotic games, sports, theoretical games, real-time strategy (RTS), role-playing (RPGs) and first-person shooter (FPS) games. Game AI techniques can broadly be classed into deterministic and non-deterministic [7]. The former concisely define NPC behavior and typically are very predictable, while the latter facilitate learning and unpredictable gameplay as well as emergent behaviors. Traditional techniques used by game AI developers include cheating (giving the NPC more information than available to a human player), finite state automata (that allow an NPC to change general behavior based on events in the game), path finding algorithms (e.g., that provide NPCs that avoid obstacles and reach targets), scripted behaviors (that allow an NPC to be tailored to express specific behaviors under specific circumstances), expressive animations (giving the NPC the illusion of emotions), group behaviors (that provide the illusion of realistic flocks, crowds, etc). Most of these traditional techniques are deterministic, though some do use adaptive algorithms. Typical NPC behaviors that these traditional techniques provide include “chasing and evading”, following pre-determined patterns of movement, avoiding obstacles and “following the leader”.

A wide variety of evolutionary computation techniques have been explored over the years in the quest for better game AI. The choice of technique, the metrics against which to measure it and the success of the solutions vary significantly and are highly dependent upon the goals of the game-play. Further there is a tradeoff to be made between the development time required, the level of capability achieved, the stability of play, and the computing resources consumed during game-play. We broadly categorize the focus of the NPC behaviors along three interrelated dimensions:

- (a) Ability to play well
- (b) Ability to perform in an effective manner
- (c) Ability to behave in an interesting manner

Playing well typically involves achieving certain objectives or “winning” the game, as defined by specific criteria. The performance of the NPC or team of NPCs usually may be evaluated simply in terms of how well it performs against the human player or other NPCs in achieving game objectives. Evolutionary computation approaches concerned with developing NPC behaviors that “play better” typically have focused on the off-line optimization of controllers [8] and play strategies [9]. Broadly speaking, many of these approaches have achieved good levels of success in games for which the rules are fairly strict and the criteria determining success are straightforward. For example, it is possible to evolve a good checkers player [8] or backgammon player [9] that can challenge an expert human player.

Performing in an effective manner typically involves doing things “the right way”. Thus, it is not enough to win, but the NPC must do it efficiently and intelligently. For instance, effectiveness may be measured based on amount of

resources consumed, the amount of risk taken, and the variety of different tactics (of the human player or other NPCs) it can recognize and handle. Playing effectively is paramount for games in which play options tend to be highly varied, the game conditions change frequently and/or there are multiple game objectives that determine success. Evolutionary computation techniques have been applied, with varying degrees of success, to a number of such domains, including robot soccer [10,11], the board game RISK [12], robotic battles [13], real-time strategy games [14] and first-person shooters [15]. Evolutionary approaches have included both the off-line optimization of controllers and play strategies [10,11] as well as the on-line adaptation of controllers [13], plans and play strategies [12,14]. Approaches that have been most successful focus on changing the behavior of the NPC in response to the actions of the human player.

Stanley et al. [13] present the real-time Neuro Evolution of Augmenting Topologies [rNEAT] in which neural networks are used to represent agent behaviors and are adapted during gameplay in response to human player actions. The solution presented was specific to a particular game that they crafted called NERO. Within NERO, the agents were simulated robots and the behaviors taught included “avoiding an enemy”, “dodging turret fire”, and navigating complex mazes without a specific path-planning algorithm. In NERO, learning itself was indispensable to the game, and the human player had the role of a trainer for the NPC.

Louis and Miles [14] present the Case-Injected Genetic Algorithm (CIGAR) in which a genetic algorithm plans and replans actions of an NPC in an RTS game about strike force asset allocation played against a single human player. Based on the game-play of both the NPC and the human player, CIGAR builds a case-base of knowledge on how it should play. During planning and re-planning, this case-base of knowledge is used to periodically inject solutions into the population that similar to the current best member. The injection of cases biases the NPC to produce solutions that may be sub-optimal from the immediate perspective of the game objectives, but which allow it to learn, over multiple games, to identify and respond to potential traps laid by the human player. Behaviors that CIGAR captured include: “break off to attack newly discovered enemies”, “reroute to avoid new threats”, and “reprioritize to deal with changes to the game state”.

Behaving in an interesting manner is a subjective quality that reflects the player’s perception of the NPC and the behaviors it exhibits under different circumstances. Measures may include unpredictability, believability, variety of responses, realism, consistency and interpretability (i.e., can the human player sometimes “figure out” the intent behind an NPC’s action vs. does it always appear random). Some evolutionary approaches have addressed the development of “interesting” behaviors, though most have focused on generating realistic group behaviors (e.g.,

flocking behaviors [16]). These group behaviors are often relatively simple, but may show impressive variability at the aggregate level over time and under different conditions in the game. Further, the use of evolutionary computation to produce highly adaptive, yet interesting and appropriate tactical behaviors in real-time has been explored for simulated combat situations with individual and small numbers of simulated military units [5,6,17]. However, creating interesting individual NPC behaviors that are relatively complex and interact meaningfully with human players has received, by contrast, little attention in the field of evolutionary computation. Mostly this is due to the difficulty of deriving a fitness function to accommodate the subjective nature of the measures.

For example, one specific dimension of “interesting” pertains to achieving “believable” human-like performance. Livingstone [18] recently examined the issue of assessing the believability of game AI. The main instruments used for collecting data currently seem to be surveys and questionnaires for players and observers. He suggests three broad categories of NPC behaviors that pertain to believability: Plan, Act and React, and provides the following set of criteria against which players and observers may judge NPCs.

A believable NPC will

1. Plan
 - a. with some degree of strategy and/or tactics
 - b. with the ability to coordinate actions with the player or another AI
 - c. so that it does not repeatedly attempt a previous, failed plan or action
2. Act
 - a. with human-like reaction times and abilities
3. React
 - a. to players’ presence and actions appropriately
 - b. to changes in their local environment
 - c. to the presence of foes and allies

For example, NPCs that act with extremes of behavior (i.e., too fast, too slow, too good) tend to be viewed as less human-like (2.a).

In recent years, the use of computational human behavior models (HBMs) to control NPCs within a game has been explored, with some success [1-4]. An HBM differs from a typical game AI or evolved set of behaviors in that a real-time decision of what behavior to express is made based on a combination of game rules, current situations, current internal operating modes, and internal behavior logic. Key here is the internal operating mode. For example, an HBM may model emotion states and produce different behavior when experiencing different emotions. As one may expect, the techniques tend to provide NPCs with behavior that appears quite believable. However, the ability of the HBM to also perform well against game winning conditions and to perform effectively while trying to win is not always an easy task to accomplish.

The field of game AI and the evolutionary computation approaches to game AI have addressed a wide variety of solutions to the problem of robust NPC behaviors. A continuing challenge is the development of NPC controllers that play well, effectively and in an interesting manner.

III. ACTB OVERVIEW

We recently developed a prototype genetic algorithm (GA) based controller, called the Advocates and Critics for Tactical Behaviors (ACTB), for unmanned ground vehicles (UGVs). ACTB was developed with the intent of controlling one or more UGVs so that they demonstrated meaningful tactical behaviors under dynamic conditions. Further, it was desired that new behaviors could be easily added to the system. The details of the ACTB are presented in [5,6]. We briefly summarize the salient features of ACTB, indicate the subset of behaviors used in our integration process, and discuss what made us consider it for controlling NPCs.

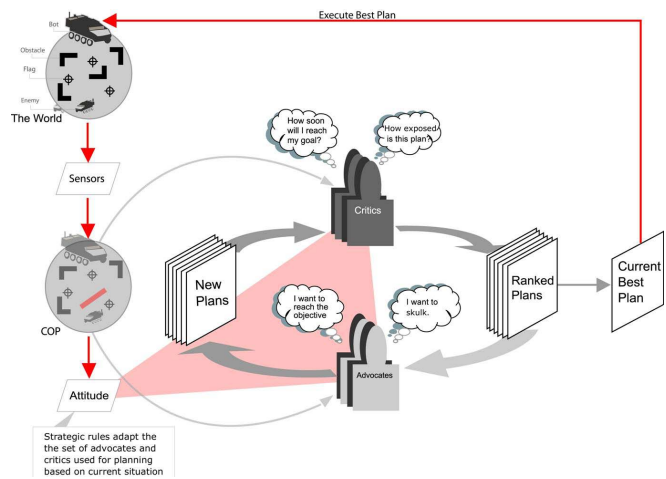


Figure 1: Continual planning cycle of ACTB

Within ACTB, real-time planning and re-planning is performed in a continual cycle that is influenced by changes in the state of knowledge about a simulated world as well as by the current planning emphasis of the system (see Figure 1). A genetic algorithm performs a search over the space of possible ‘plans’, where a plan indicates locations to move to as well as actions to perform at those locations (represented as sequence of waypoints and actions). A typical goal of a plan is to visit a number of flag locations while avoiding or fighting any enemies encountered. A plan may be for a single UGV or for multiple, collaborative UGVs [5].

Domain-specific operators termed advocates suggest changes to be made to the plans, as well as the more typical random genetic operators of crossover and mutation. The advocates may makes probabilistic changes anywhere on the genome, or, alternatively, may focus their efforts to specific parts of the genome (such as immediate-term actions, or parts of the plan that have been flagged as problematic). For the purposes of this integration effort, we selected advocates that primarily performed path planning (e.g., obstacle

avoidance), safety (e.g., flee a nearby enemy, avoid a known enemy that is not an immediate threat, skulk near walls), and aggression (e.g., maneuver to attack the enemy), as well as the random operators.

The fitness of a plan is evaluated via a weighted sum of diverse fitness components termed critics. A critic evaluates the plan against specific objectives and flags parts of the genome that are problematic given that critics criteria. Critics used in this integration effort include:

Critic	Metric
ObjectiveSuccess	Does planned path reach all objectives?
Safety	How much of my path is in visual range of the enemy?
Traversability	How much of the planned path goes through obstacles?
Duration	How long does it take to traverse the entire path?
TimeToObjective	How long to reach the first objective?
ComplexPath	How many segments are in the path?
Skulk	How visually exposed is the NPC?

During the development of ACTB, we identified the need for a higher-level organizing principle. If too many advocates and critics were used, the system became unable to react in real-time to events since it had too many factors to balance and significant changes to the population would take too long. As such, a consistent set of advocates, critics and their relative importance is grouped into an Attitude, which represents a different behavioral emphasis triggered under different environmental conditions (see Figure 2). The full set of ACTB attitudes were in this effort, including Cautious, Brave, Scared, and Ambitious. For instance, when Cautious, a high weight is given to the Skulk critic, thereby rewarding plans that limit potential exposure, but when Scared, the Safety critic has a higher weight, thereby rewarding plans that move away from the enemy quickly.

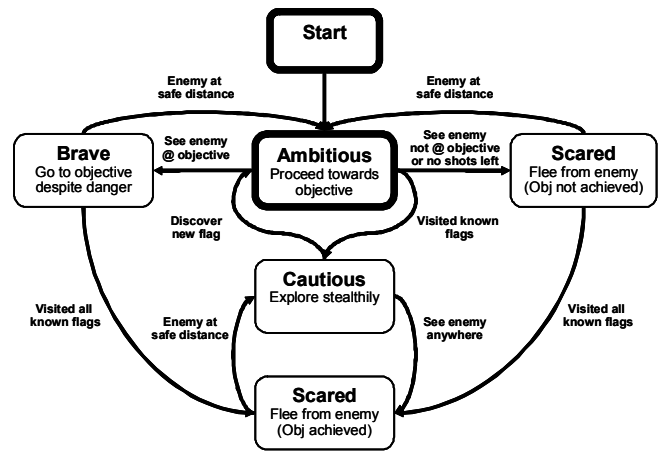
The use of attitudes was a key factor that led us to considering ACTB as a genetic NPC controller. As with human behavior models, ACTB changes the way it performs depending upon the conditions it experiences. Further, the use of random genetic operators and probabilistic advocates leads to a continual variation in the plans generated.

A second key reason that led us to considering ACTB as a genetic NPC controller was more subjective. ACTB was initially developed to “perform in an effective manner” by using tactical behaviors appropriate to a situation. However, in previous results in simulation [6], we were struck by the ability of the ACTB controller to recover from unexpected circumstances, to respond in an interesting manner to the actions of other simulated UGVs, and to generally behave in a manner that was easy to anthropomorphize.

Figure 2: Attitude transitions based on current conditions

IV. GENETIC NPC CONTROLLER INTEGRATION

Controlling NPCs within a game using an external process



requires the capability to

- communicate information about changes in the game state to the NPC controller on a frequent, regular basis
- communicate information about the terrain and what the NPCs can see
- issue commands from the NPC controller to the game that result in the indicated actions being taken by an NPC in a timely manner
- receive information about other entities in the game in a timely manner

The game Neverwinter Nights (NWN) provides a rich, easy-to-use scripting environment for developing new game scenarios. A third-party tool, Shadow Door (copyright 2003 Phillip Saltzman and Robert Zubek), provides an NPC control interface for NWN that allows an external LISP process to receive information from and send information to a running instance of the game via a port on the host computer. A simple string-based format is used for specifying commands and for communicating information between the external process and an NPC within the game. For example, “speak!#Hello” will issue a command that causes the NPC to utter ‘Hello’. Shadow Door provides commands to cause a single NPC to move to an object in the game, attack a player’s character (PC), speak a statement or perform a specific animation. It proactively provides information to the external process about what is said to it and by whom it is attacked (e.g., “attacked-by#player-name”).

To integrate ACTB as an NPC genetic controller, we developed an ACTB-NWN bridge based upon Shadow Door, with the external part of the bridge programmed in Java 1.4, and the complementary internal part of the bridge written in NWN’s scripting language. To satisfy the basic requirements of the advocates, critics and attitude transitions used, as well as enable external control of multiple NPCs, we added the ability to

- target specific commands for specific NPCs.
- obtain information about an NPC’s current location, heading and state of health (alive/dead)
- obtain location, heading and health information about the players and other NPCs that can be seen by an NPC.
- issue a command to move an NPC to a specific location

Due to limitations in the NWN scripting language, we were unable to obtain information about what terrain was visible to the NPCs, so we provided complete information about the terrain to the NPC controller a-priori.

To enable the system to be used as a testbed for exploring alternative behaviors and configurations of the NPC controller, we added the ability to

- dynamically define a scenario through the creation of flags in random locations that the NPCs and PCs must visit in order to win the game
- allow an external process to initiate new scenarios without bringing down the server
- automatically track which flags have been visited and terminate the scenario upon completion
- provide location information about flags to the NPCs *a priori* or not, to enable testing of different conditions

As inspiration for this testbed, and to facilitate integration with ACTB, many of these characteristics were similar to those used in a previous simulator used to test ACTB [6].

The integrated system works as follows. A NWN game is initiated and the genetic NPC controller started, each as their own process and possibly on different machines. The human player character and a specific number of NPCs each appear at random locations in the game world. The NPC controller is given information about any objectives (flags) that are visible or known to the NPCs it is controlling. The controller randomly generates an initial population of movement plans, and immediately evaluates them based on its default attitude (i.e., ambitious). It then starts to evolve in a steady state manner, always applying advocates and critics based on the most recent state information available. Specifically, the controller receives information about the location of the NPCs and any visible enemies or flags every two seconds by polling the game state via the bridge. As key events occur (i.e., discovering an enemy, reaching a flag), the attitude is adjusted accordingly.

Continual planning is used in which the best plan in the population is always adopted as the current execution plan. The current plan is executed by moving through each planned location in the path in sequence. Specifically, for each location in the planned path of each NPC, a movement command is issued to the NWN bridge, which results in the corresponding NPC in the game moving towards that point. The controller waits until the NPC has reached the requested point before issuing the next movement command. The controller may be forced to reissue a command if the NPC doesn't move in response to a movement command. If, after reissuing a command, the NPC remains immobile for a time, the next node in the path is sent. Via this mechanism, the NPC controller accommodates the physics of the game and the unexpected, interactive nature of game situations by not assuming that commands that are issued are perfectly executed.

For our integration efforts, we applied most of the ACTB behaviors, including: avoiding being seen, skulking close to walls, and going around obstacles and enemies. Several ACTB behaviors, including visual awareness (e.g. advocates to plan where to look, peek around corners, peek and then jump back and a critic that rewards looks that maintain time-

decaying situational awareness) required the ability to independently control where the NPC was looking in NWN. However, in NWN, an NPC only faces the direction in which it is moving. Using the scripting capabilities of NWN, we developed a virtual 'look' capability that caused the NPC to scan in directions other than the one it was facing. This allowed us to integrate and test look-based behaviors. However, since there was no corresponding animation available, a player could not tell where the NPC was looking (which detracted from realistic interactions since it produced an "eyes in the back of the head" effect).

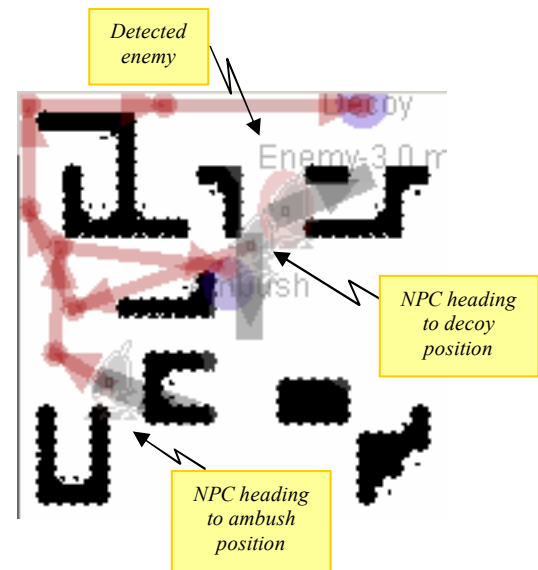


Figure 3: Illustration of new ambush behavior

To demonstrate that interesting collaborative behaviors for interaction with a human player are feasible, we developed a new 'Ambush' behavior. Upon detection of an enemy and entering a Brave attitude, this behavior advocates for plans that have two NPCs approach the enemy from different directions. Figure 3 illustrates an application of the ambush behavior. The image is a screenshot observed via our graphical user interface. The map indicates obstacles in black and current ambush objectives as small circles with labels (i.e., ambush near middle of image and decoy near top right). The light arrows show the path segments planned for each NPC, and the small dark arrow originating at an NPC shows the current direction it is looking in. The NPC controller has generated a plan that will take one NPC around behind the enemy while the other NPC approaches a frontal position.

In summary, we used NWN and Shadow Door to develop a proof-of-concept control interface and testbed that enables our ACTB control software to receive state information from the game and to issue commands to one or more NPCs. We integrated most of the controls and behaviors available in ACTB, though there were several limitations in the information that could be provided (readily) by NWN.

V. OBSERVATIONS

For this initial effort, we present our observations based on the authors playing against the NPC controller a number of times. As we expected from our experience with the original ACTB system, our genetic NPC controller is able to express several basic game AI capabilities, such as evading, chasing, following pre-specified patterns of movement, and path planning to avoid obstacles and achieve objectives.

It also incorporated some basic steps towards demonstrating a personality (via the weights on the critics and attitude state transitions) and emotions (via the set of attitude states used), but these were hard to observe from the perspective of the human player in the game. For instance, when an NPC runs past the player, it may “disappear” and take time to find again due to the limited view of the player. As such, many interesting NPC behaviors are missed.

Subjectively, our observations (as players) were that:

1. The ACTB software was able to control up to three NPCs in NWN.
2. Time-lags in the NPCs’ responses were not obviously noticeable
3. The play style of an NPC was somewhat predictable to us since we knew the underlying algorithms.
4. The play was not repetitive and the NPCs rarely made the exact same maneuver twice. Though they would, as expected, make similar maneuvers with random variations, those appeared visually different to the player due to the limited view.
5. The NPC controller played well. In fact it was often difficult to catch up to and attack an NPC.
6. It was possible to determine that the attitude of an NPC had changed, though, surprisingly, it was not easy to determine which attitude was current. This is probably due to the fact that most of our attitude transitions centered around the interactions of the NPCs with the human player, so every time we viewed the enemy, the NPCs were in the process of changing attitudes.
7. There was some “back-and-forth” as the NPC controller made up its mind, but usually it would quickly resolve after a couple iterations and commit to an action.

Using the original ACTB graphical user interface, we were able to gain an “overview” of the game world state, as known to the NPC. Thus, we could see the full map, the location of the NPC, the player and what they were doing (see Figures 3, 4 and 5). At this level, the tactics used and the interactions that occurred were more apparent to an observer. Overall, the challenge of the scenario we had designed was not very high, and since the speed of the NPC and the player were the same, it was easy for the NPC to evade the player.

Table 1 presents some preliminary results that illustrate the relative timeliness of the NPC controller. All runs were

conducted on a single processor Pentium M 1.7 GHZ with 1 GB of RAM. For each condition, two scenarios were used, each with the same map and three flags in random locations. The table shows the resulting times and averages. We tested the NPC controller with varying numbers of NPCs, and under two conditions: whether it had a priori knowledge of the location of the flags (informed) or not (naïve). The preliminary results show that the NPC controller was comparable in performance to a human player for a single, naïve NPC, and was definitely able to exploit collaboration among the UGVs to reduce its time to completion.

Table 1: Time to visit all flags for single and multiple NPCs

	Player (Naïve)	One NPC		Two NPCs		Three NPCs	
		Informed	Naïve	Informed	Naïve	Informed	Naïve
Completion	451	162	218	76	61	49	74
Times (sec)	315	147	475	94	138	62	84
Average	383	154	346	85	100	56	79

Figure 4 shows screenshots for two different informed runs. Figure 4a illustrates a detailed, stealthy plan for a single NPC that accomplishes all goals while remaining close to walls and minimizing exposure in open space (this run took 147 seconds to plan and execute). Figure 4b illustrates three NPCs collaborating to achieve the same mission (this run took 49 seconds to plan and execute).

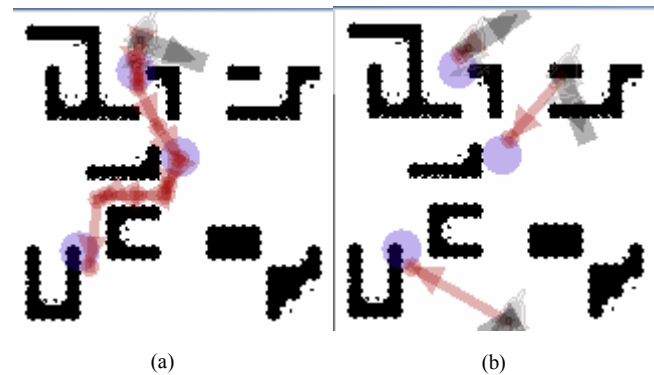


Figure 4: (a) Single NPC being stealthy and (b) Three NPCs collaborating

Figure 5a illustrates a run of a single naïve NPC which has discovered one flag and is exploring. The large circles indicate target locations that the NPC is considering during its exploration. These are generated by the controller to encourage the NPC to visit new areas in its quest to find flags. Figure 5b illustrates successful discovery of a new flag (this run took 475 seconds to plan and execute).

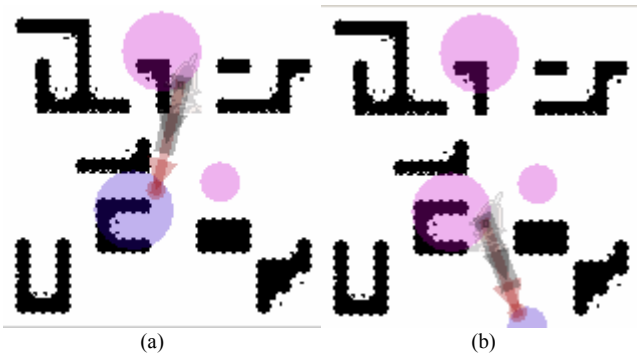


Figure 5: Single NPC exploring (a) seeking a new flag and (b) finding it

VI. DISCUSSION

Our NPC controller exhibited many of the seven criteria for believable game AI discussed earlier. It would plan in a tactical manner (1.a) and would not repeatedly attempt a failed plan or action (1.b). It did not act too fast or too well (2.a), though the occasionally back-and-forth during re-planning would make it appear less-than-believable. It would react to the players' presence in a meaningful manner (3.a), though all players were considered enemies in this effort. We did not explicitly test its reaction to changes in the environment (since there were none in our scenario, only the a priori terrain map), though this is a known capability of the core ACTB (all advocates and critics always operate based on the most recent information about the environment). Further, we demonstrated that our controller could be used to control multiple NPCs in a collaborative fashion (though it does not have the ability to explicitly coordinate with the player or with another AI).

Our genetic NPC controller exhibits similar tactical behaviors as rNEAT and CIGAR, though using a very different technical approach. Compared to rNEAT, our underlying algorithm is quite different since we do not learn on-line, but instead adapt rapidly based on static planning capabilities (i.e., advocates, critics and attitudes do not change). Compared to CIGAR, we use a continual GA planning cycle, with additional event-driven changes to both operators and fitness, but not to the population, while CIGAR re-plans using a new genetic search with a new population (but containing some previous and stored individuals) in response to each event, and only provides a new plan after the search completes. The re-planning in CIGAR thus appears to be relatively infrequent compared to our "real-time" re-planning. Further, CIGAR is based upon a static fitness function, while ours adapts in response to events, based on attitude transitions.

We were able to incorporate most of the original ACTB capabilities into our NPC controller, as well as a new ambush behavior. A limiting factor to the complexity of the controller was the limited types of information that could be transmitted from *Neverwinter Nights* using its internal scripting capabilities. For example, the lack of ability to capture information about the speed of NPCs and players

makes it difficult to model the enemy and determine interceptions.

VII. CONCLUSIONS AND NEXT STEPS

We have presented our initial efforts at using a real-time genetic mechanism for controlling NPC behaviors in an interactive game. To first order, the outcome was positive – real time control for somewhat believable behaviors appears feasible. Further integration efforts are required to incorporate additional planning behaviors and to extend the complexity of the game objectives, the environment and the possible NPC actions.

We plan to explore different ACTB-based NPC controllers by varying high-level strategy (i.e., which attitudes and transitions they use, what their initial attitude is) and low-level planning behaviors (i.e., different subsets of critics and advocates). Our belief is that we can achieve highly "interesting" and believable NPC behaviors through the use of an increasingly rich set of attitudes, advocates and critics. Ideally, in the long-term, we would like to develop critics that explicitly reward more believable plans, though this will require new measures and ways of reflecting about the NPC activities.

In addition to further development efforts, we plan to identify an evaluation methodology that may be used to validate the flexibility, purposefulness and believability of the NPC behaviors. Although it is difficult to quantify these dimensions, we can set up subjective experiments where human observers measure different NPCs against one another and against human players using criteria similar to those of [18]. For instance, the observers could be asked to rate each character along a number of positive and negative plausible characteristics components of believability. Positive characteristics may include: 1) Surprise (can an NPC ambush me), 2) Timely response (can an NPC get away from me when I attack), 3) Decisiveness (can the NPC come up with a new consistent plan quickly), 4) Mission capability (does it achieve its goals). Negative qualities observers could rate may include: 1) Predictable movement, 2) Slow response, 3) Thrashing (going back and forth without getting closer to the goal), 4) Remaining exposed for a long period, and 5) Ignoring threats. Such an evaluation methodology could confirm our suspicion that rich ACTB-based NPC controllers are better (e.g., be judged to have higher positives and lower negatives), as well as be used to compare genetic NPC controllers with other game AI techniques.

REFERENCES

- [1] M. van Lent, J. Laird, J. Buckman, J. Hartford, S. Houchard, K. Steinkraus, and R. Tedrake, "Intelligent agents in computer games," *Proceedings of the National Conference on Artificial Intelligence*, Orlando, FL, July 1999, pp. 929-930.
- [2] B. Magerko, J.E. Laird, M. Assanie, A. Kerfoot, D. Stokes, "AI characters and directors for interactive computer games," *Proceedings of the 2004 Innovative Applications of Artificial Intelligence Conference*, San Jose, CA, July 2004.

- [3] B. Silverman, *More Realistic Human Behavior Models for Agents in Virtual Worlds: Emotion, Stress, and Value Ontologies*, Philadelphia: U of Penn/ACASA Technical Report, 2001.
- [4] D. McDonald., A. Leung, W. Ferguson, and T. Hussain, "An abstraction framework for cooperation among agents and people in a virtual world," *Proceedings of the 2006 Conference on Artificial Intelligence and Interactive Digital Entertainment*, Marina del Rey, CA, June 20-23 2006.
- [5] T. Hussain, D. Montana and G. Vidaver, "Evolution-based deliberative planning for cooperating unmanned ground vehicles in a dynamic environment," *Genetic and Evolutionary Computation - GECCO 2004, Part II. Lecture Notes in Computer Science 3103*, K. Deb et al (Eds.) Berlin: Springer-Verlag, Seattle, WA, June 26-30, 2004, pp. 1017-1029.
- [6] T. Hussain, G. Vidaver and J. Berliner, "Advocates and critics for tactical behaviors in UGV navigation," G.R. Gerhart, C.M. Shoemaker and D.W. Gage (Eds.), *Unmanned Ground Vehicles VII Conference: Proceedings of SPIE, Volume 5804* Bellingham, WA: SPIE, Orlando, FL, March 29-31, 2005, pp. 255-266.
- [7] D.M. Bourg and G. Seemann, *AI for Game Developers*. Cambridge, MA: O'Reilly Media, Inc., 2004.
- [8] K. Chellapilla and D.B. Fogel "Evolving an expert checkers playing program without using human expertise," *IEEE Transactions on Evolutionary Computation*, 5(4), pp.422-428, 2001.
- [9] Y. Azaria and M. Sipper, "GP-Gammon: Using genetic programming to evolve backgammon players," *Proceedings of 8th European Conference on Genetic Programming (EuroGP2005)*, M. Keijzer, A. Tettamanzi, P. Collet, J. van Hemert, and M. Tomassini, Eds. 2005, vol. 3447 of Lecture Notes in Computer Science, pp. 132-141, Springer-Verlag, Heidelberg, 2005.
- [10] S. Luke, C. Hohn, J. Farris, G. Jackson and J. Hendler, "Co-evolving soccer softbot team coordination with genetic programming," *Proceedings of the First International Workshop on RoboCup, at the International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, 1997.
- [11] S. Whiteson, N. Kohl, R. Miikkulainen and P. Stone, "Evolving RoboCup keepaway players through task decomposition," *GECCO 2003: Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 356-368, July 2003.
- [12] J.M. Vaccaro and C.C. Guest, "Planning an endgame move set for the game RISK: A comparison of search engines," *IEEE Transactions on Evolutionary Computation*, 9(6), pp. 641-652, 2005.
- [13] K.O. Stanley, B.D. Bryant. and R. Miikkulainen, "Real-time neuroevolution for the NERO video game," *IEEE Transactions on Evolutionary Computation*, 9(6), p. 653-668, 2005.
- [14] S.J. Louis and C. Miles, "Playing to learn: Case-injected genetic algorithms for learning to play computer games," *IEEE Transactions on Evolutionary Computation*, 9(6), p. 669-681, 2005.
- [15] N. Cole, S.J. Louis and C. Miles, "Using a genetic algorithm to tune first-person shooter bots," *Proceedings of the International Congress on Evolutionary Computation*, Portland, Oregon, p.139-145, 2004.
- [16] C.W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model, in computer graphics," *Proceedings of SIGGRAPH '87*, pp. 25-34, 1987.
- [17] V.W. Porto, D.B. Fogel and L.J. Fogel, "Generating novel tactics through evolutionary computation," *SigArt Bulletin*, ACM Press, Fall 1998, pp. 8-14.
- [18] D. Livingstone, "Turing's test and believable AI in games", *ACM Computers In Entertainment*, 4(1), 2006.