Towards more human-like computer opponents

Michael Freed, Travis Bear, Herrick Goldman, Geoffrey Hyatt,
Paul Reber, and Joshua Tauber

NASA Ames Research Center
mfreed@mail.arc.nasa.gov

Current-generation online games typically incorporate a "computer" opponent to train new players to compete against human opponents. The quality of this training depends to a large degree on how similar the computer's play is to that of an experienced human player. For instance, inhuman weaknesses in computer play encourage new players to develop tactics, prediction rules and playing styles that will be ineffective against people. Game designers often compensate for weaknesses in the computer's play by providing it with superhuman capabilities such as omniscience. However, such abilities render otherwise important tactics ineffective and thus discourage players from developing useful skills.

These differences are especially pronounced in "real-time strategy" games such as Starcraft where tactics are often designed to take advantage of specific human limitations. An informal survey of experienced Starcraft players reveals numerous play-critical differences between human and computer performance. In this paper, we identify several of these differences, and then discuss a prototyping tool for constructing appropriately human-like software agents.

Play-relevant human characteristics

In real-time strategy games, players carry out a variety of fast-paced, combat-related activities including: attacking opponents, producing and positioning combat units, gathering resources needed to produce units, defending production facilities and collecting information about opponent units and facilities (scouting). Individually and in combination, these tasks are demanding in ways that push the limits of human performance. For example, dexterity-imposed limitations on control of the interface (keyboard and mouse) increase the time taken to bring groups of units into action. Players can exploit dexterity limits using "raiding" tactics: units make a brief, disruptive attack, and then depart before the opponent can command an effective response. In Starcraft, the computer can command a response without dexterity-imposed delays, enabling it to repel raids reliably. Consequently, the computer opponent provides little opportunity to learn and practice raids.

An informal survey of experienced Starcraft players was used to identify significant differences between human and computer play. These differences, summarized below, fall into general categories including: fine-motor control, visual field-of-view, and visual attention.

Fine motor control. One critical difference between a human player and a typical computer player is the speed with which commands can be issued. While this speed can be essentially unlimited for the computer, human speed depends on dexterity in controlling the interface. Limited dexterity has numerous play-relevant implications in a game such as Starcraft where micro-management of units at critical phases of combat often determines the outcome. One implication, as already discussed, is to create a vulnerability to raiding tactics. Another is to create vulnerability to "forks" in which the opponent initiates multiple, concurrent attacks. If effective defense requires carefully manipulating defending units, responding to one attack may entail suffering substantial loss against others. Similarly, the time required to control units trades off against other potential uses of that time such as bringing idle units into battle, maintaining production, and gathering information. Experienced players take advantage of this by constantly skirmishing and probing opponent defenses, thereby encouraging the opponent to interrupt and delay other important tasks.

An additional effect of limited dexterity is to reduce the value of powerful units whose attacks must be manually targeted. In Starcraft, using such a unit requires (1) selecting the unit (which may be small, moving, and partly obscured) with the mouse, (2) employing mouse/menu or keyboard shortcut to select what kind of attack it should make, and then (3) using the mouse to specify a target for the attack. The time required to execute this sequence, often several seconds for experienced players, limits the rate at which manually targeted attacks can be made. This, in turn, limits the number of such units that can be used effectively in a given battle and affects strategic choices about the production of these units compared with others that are less powerful but easier to manage.

Field of view. In Starcraft, as in many similar games, the main display window shows a portion of the playing area in detail, while a small accompanying window coarsely depicts the full map geography and provides crude situation information. The inability of a human player to view the full map in detail[1] produces vulnerability to a range of deceptive tactics.

For example, a player can "feint," attacking one location with a small force in an attempt to trick the opponent into thinking that a major attack at that location is imminent. If the opponent responds by moving units into position for the expected attack (meanwhile wasting time that could be better used for other purposes), this creates an opportunity for a real attack elsewhere. Clearly, the success of a feint depends on the opponent not knowing where one's units are actually located.

A second tactic that depends on limited view is to "lure" units out of position. In this case, the opponent is encouraged to chase a small number of units with a larger force, either to enable an ambush or simply to move opponent units away from an effective position. A third tactic is to mislead an opponent about the overall composition of one's forces by showing an unrepresentative sample. This encourages the opponent to misallocate unit production resources to counter the perceived force composition, thus increasing vulnerability to one's actual forces.

Visual Attention. Limits on people's ability to pay attention to all available visual stimuli has a variety of effects. In Starcraft, two effects stand out as especially important. First, players often fail to detect units which are visible but non-salient. For instance, a unit may be camouflaged by similar background, partially obstructed by another object, or visible only as a blurring or darkening of background terrain. Players take advantage of their opponent's limited visual attention to sneak units into position and to hide them "in plain sight."

A second effect is to delay situation assessment. In particular, understanding the nature of an attack or of a defensive position may require taking account of numerous visual objects (units, terrain features, fortifications,..). Since human attention mechanisms demand time for each object to be examined, a visually complex situation forces a player to either act prematurely (i.e. without having taken account of all available information) or to delay action.

Numerous play-relevant human characteristics may be added to those described. For example, limits on human auditory attention has effects analogous to those on visual attention. Constraints on memory performance make it difficult to maintain situation awareness by requiring repeated observations, and may enhance the disruptive effect of interruptions as players forget to resume after interruption. This discussion focuses on general human characteristics which can be taken advantage of by specific tactics. Other interesting characteristics may be idiosyncratic rather than general (e.g. susceptibility to fatigue, boredom, or overconfidence), or may result in broad effects on play style without producing specific tactical vulnerabilities.

Agent Architecture

The previous section identified common playing tactics that depend on the opponent having specific human characteristics. A computer player that lacks these characteristics will not be vulnerable to the associated tactics, at least partly undermining its value as a training tool. Having identified at least some of the human qualities that could profitably be incorporated into a computer opponent, it is worth considering how these may be incorporated effectively. In our view, a successful approach should satisfy the following criteria:

1. Since many games emphasize similar aspects of human performance, human characteristics underlying performance should be represented in a highly reusable agent architecture.
2. The main consideration in building a computer player is enabling it to play the game effectively. The agent architecture should facilitate constructing capable agents by incorporating sophisticated AI mechanisms for selecting and controlling action.
3. The architecture should emphasize limitations and temporal characteristics of human performance that tend to be game-relevant.
4. Games will differ in which aspects of human performance are worth representing. When developing any particular game agent, it should be easy to "turn-off" or ignore human attributes not currently relevant.

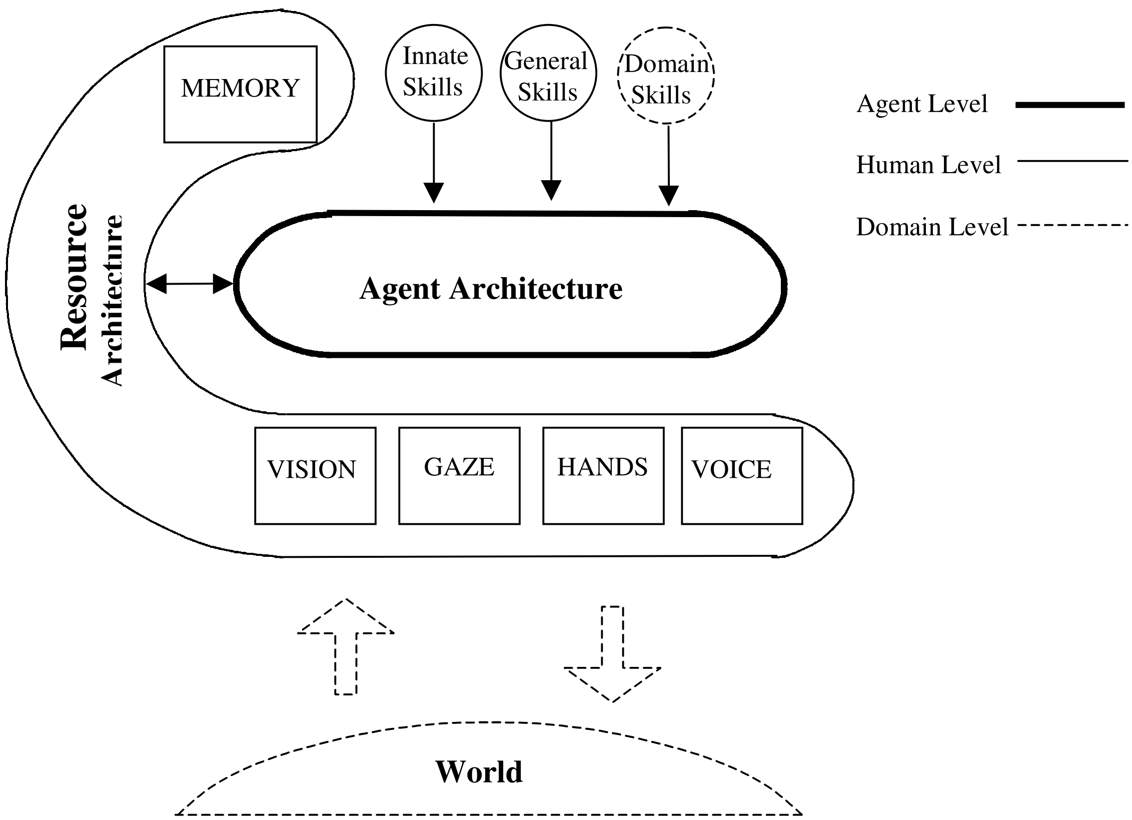
In the remainder of this paper, we will describe an agent architecture called APEX that satisfies these criteria. APEX was developed to help simulate human commercial jet pilots, air traffic controllers, and other highly skilled

[1] Limited field of view arises both from interface design and from innate human limits. In particular, innately limited human visual acuity effectively limits field of view on a detailed scene by requiring a person to stay close to the display. This puts much of the scene out of view or in the visual periphery.

operators in complex real-world environments. These tasks and environments are similar to those in many games, including real-time strategy games. Thus, we expect that APEX's success in simulating operators in these domains will translate to success at simulating human gamers.

As software, the APEX consists of two main components. The *agent architecture* provides a set of powerful AI mechanisms that enable an APEX agent to operate capably in demanding task environments. This component is not intended to be particularly human-like, and could in fact be used to simulate (or control) a wide range of agent types including animals and robots. The *human resource architecture* consists of modules, each representing a human cognitive, perceptual, or motor faculty (resource); these give the agent component human characteristics.

Agent architecture. The core of the agent architecture is a reactive planner (Firby, 1989; Simmons, 1994; Gat, 1996; Pell, et al., 1997; Freed and Remington, 1997), an algorithm used to generate competent behavior in dynamic,



time-pressured and otherwise demanding task environments. Reactive planners select action based on a library of stored plans which together represent the agent's expertise in a particular environment. Different reactive planners use different plan notations and incorporate somewhat different capabilities. The following describes an APEX plan for attacking a fortified position:

```
(procedure
  (index (attack ?fortification with ?artillery-grp))
  (step s1 (select staging-ground near ?fortification => ?stage))
  (step s2 (select support-group => ?supp-grp))
  (step s3 (move units ?artillery-grp to ?stage) (waitfor ?s1))
  (step s4 (move units ?supp-grp to ?stage) (waitfor ?s1 ?s2))
  (step s5 (interpose ?supp-grp btwn ?artillery-grp ?fortification)
          (waitfor ?s3 ?s4))
  (step s6 (target ?fortification with ?artillery-grp) (waitfor ?s5))
  (step s7 (terminate) (waitfor (destroyed ?fortification))))
```

When a plan's INDEX clause matches an active goal, it is retrieved from the plan library. For instance, a goal of the form
(attack base7 with company5) would match the plan above and cause it be retrieved. Plan STEPs corresponding to low-level "primitive" actions such as keypresses and gaze shifts are handled by signaling the appropriate module of the resource architecture to carry out the action. Non-primitive steps are treated as goals and (recursively) decomposed into subgoals using other stored plans.

The APEX planner emphasizes capabilities for managing concurrency, repetition and multitask interactions. For instance, to support specification of concurrent behavior, parallel execution of plan is assumed unless order is specified. For example, steps s3 and s4 above are each constrained to wait until step s1 has completed; s4 has to wait for s2 also. However, the two steps are not ordered with respect to one another. For this particular plan, this has the useful consequence that group has to wait (senselessly) for the other to arrive at its destination before beginning to travel.

Even when a plan specifies no order on a pair of steps, constraints may emerge during execution that require order to be imposed dynamically. For instance, subgoals generated in the process of carrying out s3 and s4 may come into conflict when each needs the dominant hand to manipulate the mouse. To resolve this, the planner needs to temporarily suspend effort at either s3 or s4, thus managing access to a limited resource by imposing order on otherwise concurrent activities. Detecting and resolving such conflicts is the responsibility of multitask coordination mechanisms incorporated into the APEX planner; further information on these mechanisms can be found in (Freed, 1998b).

Resource architecture. The function of the resource architecture is to give the agent human-like qualities, particularly performance-limiting and temporal characteristics. For example, the module representing a human hand/arm specifies that completing a targeted action such as a grasp, button push, or mouse movement requires an amount of time determined by Fitts Law (Fitts and Peterson, 1964). It also represents attributes such as limited strength. Strength, completion time and other characteristics are automatically considered whenever the agent component commands the hand/arm to take an action.

For example, if a lifting action is commanded, the hand/arm will first check the object's weight against strength limits, causing the action to fail if it exceeds the strength threshold. Next, if the action is to succeed, the hand/arm resource determines how much time should elapse before completion. After this interval, the hand/arm signals the world (game application) to indicate that the action has taken place.

We have identified 6 general characteristics that apply to any all resource modules (Freed, 1998a). Using the hand/arm to illustrate, these are: *capacity* (limited lifting strength); *precision* (maximum fine motor control); *bias* (handedness); *fatigue* (muscle weariness); *unique state* (a hand can only be in one place at a time); and *time* (every action requires some). Some categories may apply in several different ways (e.g. capacity also applies to hand/arm as a limit on graspable volume). We are still far from having incorporated the complete set of human characteristics implied by this taxonomy, although significant progress has been made in modeling certain resources, particularly vision.

A key feature of the resource architecture is the ability to turn off characteristics; for example, one could prevent the hand/arm from enforcing limits on strength, allowing objects of any weight to be lifted. The importance of this feature can be seen when one considers the diverse and often subtle means by which people cope with or circumvent their limits. For instance, people compensate for limited visual acuity (loss of detail about objects in the periphery of vision) by shifting gaze – scanning and searching as needed to maintain a relatively current picture of the whole visual field. Since people rely on domain-specific expertise to scan effectively, new scanning plans must be created for any new game agent. Developers who do not want to create scanning plans can turn off acuity limits, allowing the visual field to be examined in detail without scanning.

Conclusion

More human-like computer opponents could serve a valuable role in training new players, and may have other benefits such as providing players with an improved testing ground for new tactical and strategic ideas. In our view, the problem of creating such agents involves two main problems. First, a game designer must be able to identify which aspects of human performance are most relevant to gameplay – i.e. which are most significant in determining how well various tactics would work against a human opponent. Second, the designer requires software and methodological support for developing capable, human-like agents. In this paper, we present progress on both problems.

References

Fitts, P.M. and Peterson, J.R. 1964. Information capacity of discrete motor responses. *Journal of Experimental Psychology*, 67, 103-112.

Firby, R.J. 1989. Adaptive Execution in Complex Dynamic worlds. Ph.D. thesis, Yale University.

Freed, M. 1998a. Simulating human performance in complex, dynamic environments. Ph.D. Dissertation, Department of Computer Science, Northwestern University.

Freed, M. 1998b. Managing multiple tasks in complex, dynamic environments. *In Proceedings of the 1998 National Conference on Artificial Intelligence*. Madison, Wisconsin.

Freed, M. & Remington, R.W. 1997. Managing Decision Resources in Plan Execution. *In Proceedings of the Fifteenth Joint Conference on Artificial Intelligence*, Nagoya, Japan.

Gat, Erann. 1996. The ESL User's Guide. Unpublished. Available at: www-aig.jpl.nasa.gov/ home.gat/esl.html

Pell, B., Bernard, D.E., Chien, S.A.., Gat, E., Muscettola, N., Nayak, P.P., Wagner, M., and Williuams, B.C. 1997. An autonomous agent spacecraft prototype. *Proceedings of the First International Conference on Autonomous Agents*, ACM Press.

Simmons, R. 1994. Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation*. 10(1).