Creating Human-like Synthetic Characters with Multiple Skill Levels: A Case Study using the Soar Quakebot

John E. Laird
University of Michigan
1101 Beal Ave.
Ann Arbor, MI 48109-2110
laird@umich.edu

John C. Duchi
Upper Arlington High School
1650 Ridgeview Rd.
Upper Arlington, OH 43221
duchmannnn@aol.com

Abstract

This paper reports on a preliminary attempt to develop and evaluate synthetic characters with multiple skill levels and with human-like behavior. Our goal was to determine which aspects of behavior have impact on skill level and humanness. We developed a bot that plays the computer game Quake against a human opponent. That bot can be parameterized along four dimensions: decision time, aggressiveness, number of tactics, and aiming skill. We then played variations of the bot against a human expert. Through empirical and human judgments we then evaluated the skill level and humanness of the variations. Our results suggest that both decision time and aiming skill are critical parameters when attempting to create human-like behavior. These two parameters could also be varied to modify the skill of the bots, and for some ranges, maintain the same level of humanness.

The two primary goals of this research are to create synthetic characters with human-like behavior and varying levels of skill within the context of highly interactive tasks. The secondary goal is to develop and test a methodology for evaluating the humanness of synthetic characters in those types of tasks. Both primary goals require that we explore computational mechanisms for modeling human behavior and that we understand which behavioral parameters affect the humanness of the behavior and the skill of the behavior.

Towards these ends, we have developed a parameterized AI system (called a bot) that plays the computer game Quake against humans. In developing the bot, we have tried to build an opponent that has the same strengths and weaknesses as human players. As part of its development, we parameterized the bot along four dimensions: decision time, aggressiveness, aiming skill, and tactical knowledge. This paper describes an evaluation of variations in those parameters in terms of different levels of skill and humanness. Our approach was to build versions of the bot with different parameter values and then play them against a "gold-standard" expert human player. Our results suggest that this methodology is useful, and that we could produce a range of skill levels using a subset of these parameters.

The Soar Quakebot

The Soar Quakebot (Laird, 2000; Laird & van Lent, 1999) plays the death match version of Quake II. In a death match, players exist in a "level", which contains hallways and rooms. The players can move through the level, picking up objects, called powerups, and firing weapons. The object of the game is to "kill" the other players. Each time a player is shot or is near an explosion, its health decreases. When a player's health reaches zero, the player dies. A dead player is then "spawned" at one of a set of spawning sites within the level. Powerups, which include weapons, health, armor, and ammo, are distributed throughout the level in static locations. When a powerup is picked up, a replacement will automatically regenerate in 30 seconds. Weapons vary according to their range, accuracy, spread of damage, time to reload, type of ammo used, and amount of damage they do.

The Soar Quakebot controls a single player in the game. We have attempted to make the perceptual information and motor commands available to the bot similar to those that a human has while playing the game. For example, a bot can see only unobstructed objects in its view cone and it can hear only nearby sounds. One issue is that bots cannot sense the walls as coherent objects because they consist of many polygons that give the appearance of solid walls, open doorways, etc. To navigate, the Quakebot explores and builds up an internal map based on range data to walls. The Quakebot uses this map to know where walls, rooms, hallways, and doors are when it is running through a level. Once a map is built, it can be saved for later use when the Soar Quakebot replays the level.

The Quakebot uses Soar (Laird et al., 1987) as its underlying AI engine. The Soar Quakebot is designed based on the principles developed early on for controlling robots using Soar (Laird and Rosenbloom 1990) and then extended in our research on simulating military pilots in large scale distributed simulations (Jones, et al. 1999). All the knowledge for playing the game, including constructing and using the internal map, is encoded in Soar rules. The Soar Quakebot evaluated in this paper has 100 operators, of which 20 have substates, and 715 rules. The underlying Quake II game engine updates the world ten times a second (the graphics engine updates much more often than the game engine). On each of these cycles, all

changes to the bot's sensors are updated and any requested motor actions are initiated. In this configuration, Soar runs asynchronously to Quake II and executes its basic decision cycle anywhere from 30-50 times a second, allowing it to take multiple reasoning steps for each change in its sensors. Nominally, Soar runs as fast as possible, consuming 5-10% of the processing of a 400MHz Pentium II running Windows NT.

Methodology

Our methodology for evaluating the humanness and skill level of synthetic characters is an outgrowth of the traditional Turing test, but with extensions for interactive agents that attempt to maximize our ability to determine where its behavior falls short of human behavior. One deviation from the traditional Turing test is that instead of having a human interact with the bot, we have the human view the behavior of the bot from its perspective as it plays the game. In earlier work on military simulations, we observed that attempting to evaluate behavior while participating in the simulation/game was not very discriminating for two reasons. First, the human is also performing a task and is unable devote full attention to the evaluation. Second, the human participant is limited to seeing only the small part of the bot's behavior that is available from the player's sensors. The human sees none of the bot's behavior when it is out of view and can only guess at what information the bot has available to it. In our approach, the human evaluator sees what the bot is seeing, so that all of the bot's overt behavior is available to the human.

In practice, the methodology involves the following steps:

1. Select a set of parameters of the bot's internal structure that should have impact on the behavior, either in terms of skill level, humanness, or both. We selected four parameters, three that had three distinct values and one that had five. These parameters are described in the next section.
2. Create a reference bot that we expect will have the best overall performance. In our case, we had a single bot that we expected to be best at both skill and humanness.
3. For each parameter, create agents with different values of only that parameter while holding all of the other parameters at their reference value. We took this approach to avoid the combinatorial explosion in developing agents for every possible parameter combination, which would have required 135 agents. As it was, we developed and tested 11 distinct agents.
4. Play the bots against a gold-standard human expert. In our case, the human expert had recently played Quake II completely through twice and had played extensively in internet multiplayer games. The bots played for three minutes and we recorded the view of the game from the agent's perspective on videotape. We also kept a record of the number of times the bot killed the expert or was killed by the expert.
5. Have five humans play against the expert, each for three minutes. The humans had varying levels of experience with Quake II:
    a. Novice: two subjects had never played this type of game before, but were experienced computer users.
    b. Medium: one subject had played this type of game before, but never this game.
    c. High: two subjects had played this game before, but not recently and not to the extent that our expert had.

    We recorded the view of the game from the subjects' perspective on videotape. We also kept a record of home many times the subjects' killed the expert or was killed by the expert.
6. We then used eight human judges to evaluate the humanness and skill of the behavior recorded on videotape. Three judges saw each of the videotapes (we felt that it would be too tedious to have each judge view every videotape). The judging was blind so that the judges did not know whether they were seeing humans or bots playing the game, although they knew that there would be some combination of human and computer players. The judges included both experts and novices in Quake II.

Parameterizing the Quakebot

Our initial choice of parameters was based on the desire to have parameters that had a high probability of having a noticeable impact on skill level and humanness of the resulting behavior. These were not meant to be exhaustive, but merely illustrative of the types of parameters that could be used to modify behavior. The parameters we have investigated are:

- Decision time [5 levels]. In a standard simulation, Soar's primitive decision cycle runs as fast as possible (AFAP), meaning it attempts to make decisions and select operators as fast as the computer it is running on it allows. For the Quakebot, this averages to be about .025 seconds per decision, or 40 decisions per second. One option in Soar is to set the timing of the primitive decision cycle to a fixed time. Two complexities in changing the decision time are as follows. The first is that there is latency between the Quake simulator and the Soar Quakebot because of the network connection between them. One of our future goals is to eliminate this by running both programs on the same machine. The second issue is that the Quake world model is updated only 10 times a second so that there is a point at which speeding up Soar will have no impact. That point is not at 10 times a second because Soar often will make multiple decisions in response to changes in the world.

Overall, the expectation was that slowing decision time would monotonically decrease the ability of the AI system - especially for tasks requiring high interaction with the environment. We also wanted to evaluate if changes in decision time influenced the perceived humanness of the behavior. We expect that it would because there is a growing body of work suggesting that human behavior in these highly interactive tasks is best modeled with the primitive decision making function taking 50-100 msec. (If we were accurately modeling perception and motor actions, the expected value would be 50 msec.; however, there is no explicit modeling of the time for those processes in the current Quakebot so that 100 msec is a better estimate.)

- Aggressiveness [3 levels]. We varied a set of parameters that determined how likely it is to attack and what range it attacks from.
- Complexity of tactics [3 levels]. We created three groupings of tactics to correspond to different skill levels.
- Level of expertise of specific tactically critical skill [3 levels]. This involved using different levels for a specific but critical tactical skill that is known to have impact on performance - in the case of the domain it is an aiming skill. Here we evaluated three levels. One that we expected to be worse than human, one similar to what we expect most humans to use (shoot at the current location of the enemy), and one that is superhuman because it uses information (very accurate depth and velocity perception) and calculations that are usually unavailable to human players (shoot at the point where the enemy is predicted to be when the projectile gets to that range).

Results

There are two sets of results, one in terms of skill and the other in terms of humanness. The skill results come from an empirical analysis of the head-to-head competition of the human and agents against our expert and from judgments of people watching the play of the bots. The humanness results come from judgments of people watching the play of the bots. We will concentrate on results for three of the four parameters: decision time, complexity of tactics, and aiming skill. The different levels of aggressiveness did not lead to any notable results.

One overall note on the results is that although there are interesting trends in the data, these results are only preliminary - this is essentially a pilot study. There were often moderately high standard deviations in the judgments of the viewers of videotapes, due in part because of the range of experience in our judges.

Skill Levels

To compute a skill level, we recorded the number of times the subject or bot scored a kill and the number of times the expert scored a kill for each trial. We then computed a success ratio, which was the number of subject/agent kills divided by the number of expert kills in that trial. For example, if a subject scored one kill and the expert scored ten kills that would give a success ratio of 1/10 = .1 and if there were equal kills it would give a success ratio of 1.

To test the stability of our human expert, we played the reference agent against the expert four times throughout the testing. The ratios for these four trials were 1.2, .6 , 1.1, .9 with an average of 0.93.

We also checked to ensure that the measure of skill corresponded to the level of experience in our subjects who played the games. The performance of our five human players against our expert did improve with their level of experience:

- Novices (2) = 1/28 = .036
- Medium (1) = 2/8 = .25
- High (2) = 7/19 = .37

This gives us some confidence that significant differences in the success ratios of the bots corresponded to differences in skill.

We can also compare the skill scores given by judges to these levels of experience. These rankings are on a scale of 1 to 10, with 10 being highest skill:

- Novice (2) = 2.5
- Medium (1) = 4.3
- High (2) = 4.8

This suggests that there was consistency across the three measures of human performance - experience, success, and judges' evaluations.

Skill vs. Decision Time

Decision time is the amount of time in seconds that the bot has to make a decision. As we vary the decision time of the agents, we expect to see improvement in skill, and this is exactly the result we get:

- .50 = 0.0
- .25: = 0.23
- .10: = 0.625
- .05: = 0.71
- AFAP: = 0.93

There is a monotonic improvement of skill with decrease in decision time, with the fastest decision time giving performance close to the skill level of our expert. However, this also shows that in order to achieve human-level expert skill additional improvements are needed in other skill areas because the decision time for AFAP is probably superhuman. We need to improve the time

modeling of sensing and motor actions and probably improve other aspects of skill of the bot if we run the bot at the appropriate decision time of .1 or .05 seconds/decision.

The second way to evaluate skill is via the skill ratings of our judges. As before, these rankings are on a scale of 1 to 10, with 10 being highest skill:

- .5 = 1.0
- .25 = 2.3
- .1 = 6.3
- .05 = 5.7
- AFAP = 6.0

These results show two things. First, that the human judges do not base their decisions strictly on the success ratio and second, that they perceived that the agents with a decision time of .1 seconds had the highest skill.

Skill vs. Tactics

As we increase the complexity of the tactics available to the bot, we expect that its skill will improve. The results:

- Low Tactics = 1.0
- Medium Tactics = 0.875
- Full Tactics = 0.9

The results suggest that there is no impact of the tactics on skill. This could be because the different levels of skill we implemented were not really significant for the specific trials we ran with our expert. Surprisingly, our judges did notice some difference, although these are not statistically significant.

- Low Tactics = 3
- Medium Tactics = 4.25
- Full Tactics = 4.25

One possibility is that we need to create more extreme versions of the agents in terms of tactical knowledge.

Skill vs. Aiming Skill

We predicted that aiming skill would have a significant impact on the success of the bots and this prediction was borne out in the results.

- Poor = 0.0
- Good = 0.56
- Best = 0.93

The human judges had similar evaluations.

- Poor = 4.0
- Good = 5.7
- Best = 8.0

Humanness Levels

The humanness results are based on the ratings of our judges. We asked the judges to rate the humanness of the behavior they saw on videotape on a scale from 1 to 10 and to make a binary decision as to whether they were seeing a human or a computer player (they were told that they would see a mixture of human and computer players). The binary judgments of the human players were: 16 human votes and 2 computer votes. The binary judgments of the computer players were: 21 human votes and 27 computer votes. Only one of our judges was perfect in picking human vs. computers - and this was the judge with the most experience in the game.

The ratings of the human players as humans were high with the following averages across our five human players: 7.3, 8.25, 8.0, 7.75, and 6.83. The highest rating for a bot was 6.7. A more detailed look at the results gives us some ideas as to where to improve the bot to make it appear more human.

Humanness vs. Decision Time

Our hypothesis is that using a decision time close to the predicted decision time of humans should maximize the humanness of the bot.

- .50 = 4.3
- .25: = 2.6
- .10: = 6.3
- .05: = 5.6
- AFAP: = 5.0

These results confirm our hypothesis with the two highest rating for the decision times closest to human decision time. The bot with a decision time of .10 seconds was unanimously judged to be human on the binary scale by human judges (this bot was not judged by the judge with the most experience in Quake). It should be noted that at the slowest decision times (.50 and .25), the bots' behavior degrades noticeably as it is unable to avoid crashing into walls.

Humanness vs. Tactics

Recall that we saw little impact of tactics on our computed success ratings and only a small trend in the skill ratings of our judges. The results for humanness judgments are:

- Low Tactics = 3.25
- Medium Tactics = 4.0
- Full Tactics = 5.0

Although there were high standard deviations for these results, it is interesting that the trend is that more complete tactics lead to more human behavior.

Humanness vs. Aiming Skill

In developing the different levels of aiming skill, we created the best level to be superhuman.

- Poor = 6.7
- Good = 6.7
- Best = 5.0

It would appear that the judges recognized that the best aiming skill was unlikely to be possible by a human.

Conclusions

The preliminary nature of this study does not let us draw any hard and fast conclusions about modeling the skill level or humanness in the Soar Quakebot; however, it does identify some trends. It also indicates that we can use this methodology to explore these issues and that there are areas to explore in more detail.

The most interesting trends in the data concern the role of decision time and aiming skill in the humanness and skill of the bots. Variations in decision time show changes in ratings of humanness, with the best performance coming with a decision time similar to that hypothesized for humans. Small variations near that time also show variations in skill without large decreases in humanness. An area for future study is to explore more carefully the impact of small variations of decision time.

Variations in aiming skill show changes in both skill and humanness. The judgments of humanness suggest that variations of skill at the lower end can be achieved by manipulating this parameter.

There are a number of lessons we learned from this research that we will use in future developments and experiments.

First, there are artifacts in the bot's behavior that are distinctive. It turns much faster than any of the human players and its aim is much better. Such issues need to be addressed in the underlying bot implementation.

Second, we need to improve the bot's overall skill in playing the game to compensate for changing its aiming skill. We need to also broaden the range of behaviors we add and remove when we test variants in the complexity of its tactics. For example, all of the bots use a map to navigate the world and know where to find powerups. This is more knowledge than many commercial bots have, so it would be appropriate to create bots with fewer tactics and less knowledge than our lowest bot evaluated in this study.

Third, we need to model the decision times for perception and action incorporated into our bots.

Fourth, we need to select the most human bot (.1 second decision time and medium aiming skill) as our reference bot. Even with the flaws in our bots, the results suggest that we are in the ballpark for achieving human-like behavior. The results suggest that we could immediately improve the humanness of all of the bots by using a more "human" approach to aiming. It will be interesting to see the judgments of a bot with decision time of .1 seconds, full tactics, and good aiming skill.

Finally, this research demonstrated that we could apply the Turing test more broadly to interactive dynamic environments with the goal of improving the humanness of AI systems.

Acknowledgments

The authors are indebted to the many University of Michigan students who have worked on the Soar/Games project, most notably Michael van Lent, Steve Houchard, Joe Hartford, and Kurt Steinkraus.

References

Jones, R.M., Laird, J.E., Nielsen, P.E., Coulter, K.J., Kenny, P.G., and Koss, F.V. (1999) Automated Intelligent Pilots for Combat Flight Simulation, AI Magazine, 20(1), 27-42.

Laird, J. E. (2000) It Knows What You're Going to Do: Adding Anticipation to a Quakebot. AAAI 2000 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment, March 2000: AAAI Technical Report SS-00-02.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987), Soar: An architecture for general intelligence. Artificial Intelligence, 33(3), 1-64.

Laird, J. E. and Rosenbloom, P. S. (1990) Integrating Execution, Planning, and Learning in Soar for External Environments. In *Proceedings of National Conference of Artificial Intelligence*, Boston, MA, 1022-1029.

Laird, J. E. and van Lent, M. (1999) Developing an Artificial Intelligence Engine. In *Proceedings of the Game Developers Conference*, San Jose, CA, 577-588.