The Believability Gene in Virtual Bots

M. Polceanu[a], A.M. Mora[b], J.L. Jiménez[d], C. Buche[c], A.J. Fernández-Leiva[d]
[a]Florida International University, USA
[b]Depto. TSTC, Universidad de Granada, Spain
[c]LAB-STICC, ENIB, France
[d]Depto. LCC, Universidad of Málaga, Spain
mpolcean@cs.fiu.edu, amorag@geneura.ugr.es, josejl1987@gmail.com
buche@enib.fr, afdez@lcc.uma.es

Abstract

Video game development is not only one of the most profitable entertainment industries but also represents a very interesting field of research. Particularly in the area of Artificial Intelligence (AI), it provides many interesting (and hard to tackle) challenges. One of them consists in creating artificial bots (*i.e.*, game characters not controlled by a human) that mimic human behavior or, at least, show a believable behavior. This paper deals with this issue by describing two very different approaches that were proposed for creating believable bots and applied, with some degree of success, in the context of a first-person shooter (FPS) game. One approach is based on the idea of imitating human player behavior, and the other one consists of automatically creating bots via an interactive evolutionary algorithm. The paper analyzes the performance of these proposals and introduces different forms of hybridizing both approaches for future applications.

Introduction

Video games pose a wide variety of challenges for the field of artificial intelligence, and the generation of believable agents has been identified as one of the main research areas within this field (Yannakakis and Togelius 2014). People enjoy playing with virtual players who exhibit behavior similar to themselves, who can surprise and sometimes make mistakes. The challenge for programmers is to create a virtual player that can fool humans into thinking it is another human player. Finding good solutions to this problem has a practical impact in the development of commercial games, specially in massively multiplayer online games as virtual players can be (massively) added to the game without decreasing the motivation of human players, and this directly affects the sales of game companies.

As in the original Turing test, BotPrize challenges bots to convince human judges they are human. This competition (Hingston 2009) has been held since 2008, and its aim is to find the most humanlike bots, thus it proposes a variant of the Turing test in an Unreal Tournament 2004 (UT2K4) environment. UT2K4, is a FPS game mainly focused on the multiplayer experience, which includes several game modes for promoting this playability, the most popular being *Deathmatch*, in which players are placed in an arena and the goal is to kill the highest number of opponents (*frags*) in a limited amount of time.

We describe the two bots, *i.e. MirrorBot* and *NizorBot*, with the best performance in the 2014 Edition of the BotPrize contest, and analyze the obtained results. *MirrorBot*'s key strategy is mimicry. When the bot meets other players in the game, it observes their behavior to trigger a mirroring behavior that copies the actions of that player, including movement, shooting, weapon choice, jumping and crouching. This makes the bot mimic another player in realtime, and therefore "borrow" the humanness level of real humans. *NizorBot*'s idea is completely different, proposing an interactive evolutionary algorithm which evolves a behavior, previously defined, by modeling the knowledge with the help of a human controller. After discussing the advantages and shortcomings of the two proposals, we continue our contribution with possible ways of obtaining a superior bot, by mixing the two approaches.

MirrorBot – "the Almost-Human"

MirrorBot (Polceanu 2013) was developed in 2012, specifically for the 2K BotPrize competition, and has participated unchanged in the 2014 edition. Its design consists in two main behavior modules: *default* and *mirroring*.

The *default* behavior module, which is used more often, allows the bot to navigate in its environment, collect weapons and ammunition, shoot opponents and avoid enemy fire. Its implementation consists in a set of concurrent submodules, among which the most important are: Aiming, Navigation and Shooting. Other submodules have the role of providing data required by the main submodules, such as suitable attack or judging targets, navigation help and opponent aggressiveness.

The Aiming submodule is responsible for the orientation of the bot and has two modes: normal and confrontation. In the normal mode, which is active when no opponent is within view, the bot will orient itself towards the second navigation point in its path. This gives the observer the impression that it is following a goal and that it anticipates corners. The confrontation mode is activated whenever an opponent is in sight. It works by calculating a future location of the target, using velocity information, which results in smooth aiming and trajectory anticipation.

The Navigation submodule manages the paths taken by

the bot within the environment. MirrorBot's navigation has two modes: normal and emergency. The normal mode is a variant of standard graph navigation, featuring an improvement to how the bot follows navigation points. This improvement consists in generating random locations around each navigation point, which perturbs the overall trajectory and therefore hiding botlike movement. In cases where MirrorBot gets stuck, either due to graph inconsistencies or accidents like falling off a bridge, the Navigation submodule switches to emergency mode. This activates a ray-tracing system, that allows it to "squeeze" itself out of difficult terrain, and avoid cliffs.

The Shooting submodule decides whom and how to attack. It takes into account weapon type, splash damage and distance to targets to find an appropriate direction to shoot at. It also considers collateral targets which can be attacked with minimum effort while focusing on a main target.

The second behavior module, *mirroring*, is only activated when an opponent is considered as unaggressive. An opponent is by default unaggressive, but this status changes as soon as MirrorBot detects that the given player shoots directly at it with a dangerous weapon; *i.e.*, a weapon that does or can inflict damage. More specifically, receiving damage is not the only criterion for deciding aggressiveness. Due to MirrorBot's dodging ability, even though enemy fire does not reach it, an opponent targeting it with a potentially dangerous weapon is also classified as a threat, and therefore excluded from the mirroring candidate list. For example, shooting the link gun (which deals zero damage in the BotPrize competition) or not shooting at all, gives the ideal mirroring candidate.

When the mirroring behavior is activated for a target, MirrorBot will begin recording all observable low-level actions of the opponent: aim, movement, fire, jumping, crouching and weapon choice. These are stored as frames in a sequence, which are to be replayed by MirrorBot itself. The orientation is inverted and movement maintains a constant distance to the target; *i.e.*, aiming and moving left/right, and moving forward/backward are swapped. Due to client-server connection latency, frames may be received with a delay or even lost. Average communication latency is approximately 300 milliseconds. In addition, a delay is introduced, so that the total time before action frames are reproduced amounts to the natural visual perception delay of humans.

NizorBot – "the Semi-Human"

NizorBot (Jiménez, Mora, and Fernández-Leiva 2015) is based on the idea shown in ExpertBot (Mora et al. 2015), which modeled the behavior of an expert human player (one of the authors), who participated in international UT2K4 contests. It included a two-level finite state machine (FSM), in which the main states define the high level behavior of the bot (such as attack or retreat), while the secondary states (or substates) can modify this behavior in order to meet immediate or necessary objectives (such as taking health packages or a powerful weapon that is close to the bot's position). The decisions about the states are made by an expert system, based on a huge amount of rules that take into consideration almost all conditions (and tricks) that a human expert would do. It also uses a knowledge database (bot's memory) to retain important information about the fighting arena, such as the position of items and weapons, or advantageous locations. This information is stored once the bot has discovered them, as a human player would do.

NizorBot is an implementation over ExpertBot by applying an Interactive Evolutionary Algorithm (IEA) (Takagi 2001; Parmee, Abraham, and Machwe 2008), in which human experts guide the optimization of the bot's parameters in order to obtain a humanlike bot. The basic idea is to let the experts rule out those candidate solutions (*i.e.*, individuals) that perform *subjectively* worse than others from the point of view of humanness. More specifically, every **individual** in the IEA is a chromosome with 26 genes, divided into 6 blocks of information. Each block represents the behavior of a specific feature of the bot.

The *distance selection block* [Genes 0-2] controls the distance ranges that influence weapon choice, attacking and defending. The *Weapon selection block* [Genes 3-11] controls the priority assigned to each weapon. The *Health control block* [Genes 12-13] manages the offensive, defensive or neutral behavior of the bot, based on the level of health. The *Risk block* [Genes 14-18] controls the amount of health points that the bot could risk. The *Time block* [Gene 19] defines the amount of time which the agent considers to decide that the enemy is out of vision/perception. The *Item control block* [Genes 20-25] sets the priority assigned to the most important items and weapons.

The **fitness function** is defined as:

$$f = \begin{cases} (fr - d) + s2 + \frac{s1}{2} \\ +log((dmgG - dmgT) + 1) & if \;\; fr \geqslant d \\ \frac{fr}{d} + s2 + \frac{s1}{2} \\ +log((dmgG - dmgT) + 1) & if \;\; fr < d \end{cases}$$

Where fr is the number of enemy kills the bot has obtained (frags), d is the number of own deaths, $dmgG$ is the total damage dealt by the bot, and $dmgT$ is the total damage it has received. $s1$ and $s2$ refer respectively to the number of Shields and Super Shields the bot has picked up (very important item). This function rewards the individuals with a positive balance (more frags than deaths) and a high number of frags. In addition, individuals which deal a high amount of damage to enemies are also rewarded, even if they have not got a good balance. The logarithms are used due to the magnitude that the damages take, being around the thousand. We add 1 to avoid negative values.

The *evaluation of an individual* consists in setting the values of the chromosome in the NizorBot AI engine, then a 1 vs 1 combat is launched between this and a standard UT2K4 bot at its maximum difficulty level. Once the time defined for the match is finished, the summary of the individual (bot) performance regarding these values is considered for the fitness computation.

A *probability roulette wheel* has been used as **selection mechanism**, considering the fitness value as a proportion of this probability. The *elitism* has been implemented by replacing the five worst individuals of the new population by the five best of the current one.

The **uniform crossover** operator was used, so that every gene of a descendant has the same probability of belonging to each one of the parents.

The **interaction of the game expert** has been conducted by pausing the game at some specific points of the evolution (synchronously in some generations), where a form with several questions is presented to the human. This form shows the data about the best individual of the current generation (and thus, the best overall due to elitism). The expert then performs a third person assessment (TPA) by watching a video of the corresponding bot playing a match.

Afterwards, the evaluator must fill in the form to identify those specific features (e.g. distance selection, weapon selection, etc.) that he/she considers more humanlike.

Then, the gene blocks associated to the selected features are 'blocked' so that they are not altered by the genetic operators during the evolution. This affects the rest of the population when this individual combines and spreads its genetic information. In addition, this interaction reduces the search space as only the non-frozen blocks are evolved, so that the algorithm performance is improved.

Obtained Results and Analysis

To evaluate the adequacy of the bots, both of them competed in the BotPrize competition (edition 2014).

As illustrated in Table 1, the winner of the 2014 edition was MirrorBot, which also was one of the two bots that passed the Turing test adapted to games in the previous edition of the BotPrize competition, and can therefore be considered the state-of-the-art. However, as result of the new (and harder) evaluation system, it does not reach the value for being considered human according to the competition rules (*i.e.*, 0.5) although it is relatively close to it and by only 3% below the most humanlike human player in the contest.

NizorBot also showed a very good performance finishing as the runner-up and with a humanity factor relatively close to be considered as human, at a close tie with the 2nd and 3rd ranked humans.

We argue that the requirement that a bot should be considered truly humanlike only if it passes the 50% barrier does not reflect an appropriate measure of humanness. The main reason for this claim is that, by applying this rule, *none* of the four human judges in the 2014 edition can be considered as human (according to the results shown in Table 1). Furthermore, in the previous edition of the contest, two human judges were also ranked well below the humanness barrier of 50%. This same issue generated controversies amongst the general public, regarding the 2012 competition results, where bots where deemed "more human than human".

Interestingly, the perceived level of humanness varies significantly with the conditions in which it is evaluated. For example, in the 2014 edition, one of the used game maps featured low gravity; in this case, the perceived behavior of players tended to be drastically more botlike, even in the case of real human players. This was primarily due to the fact that most of the time, the players were gliding in the air, which left little time to analyze more complex movement patterns, and easily predictable behavior was most obvious.

Player name	**FPA**	**TPA**	**H**
Xenija	0.17139763	0.8235294	0.4974635
MirrorBot	0.20164771	0.7333333	**0.4674905**
Player	0.19328127	0.6315789	0.4124301
tmchojo	0.17757519	0.6470588	0.4123170
NizorBot	0.11821633	0.7058824	**0.4120493**
BotTracker	0.20070203	0.5909091	0.3958056
CCBot	0.06214746	0.7058824	0.3840149
Juan_CVC	0.12372294	0.6190476	0.3713853
OvGUBot	0.10545765	0.6086957	0.3570767
ADANN	0.08351664	0.4761905	0.2798536

Table 1: Final Results of BotPrize 2014 (taken from the competition website). 6 Virtual Bots (marked in *italic*) and 4 human judges. (F/T PA = First/Third Person Assessment)

Therefore, the measurement of humanness is more appropriate when done in a relative frame of reference than in an absolute one, as there exist scenarios in which not even humans are perceived to behave in a humanlike fashion over 50% of the times.

BotPrize 2014 was the first edition that featured both first and third person assessments of humanness, the results of which have not been analyzed before. TPA deemed the players $5.2\ times$ as humanlike as the FPA, on average (*i.e.*, calculated as the average of TPA/FPA ratios of each player). This is also evidence that humanness is context-specific, and may be better evaluated as such. Taken separately, the average FPA scores of humans and bots are 16.6% and 12.8% respectively, while the average TPA scores are 68.0% and 63.6% respectively. This shows that humans are, on average, still more humanlike. However, in FPA, MirrorBot has the highest score in the contest followed closely by another machine called BotTracker. In TPA, MirrorBot and NizorBot have the highest scores for bots, with just one human (Xenija) who did better.

Finally, when using an absolute threshold for humanness, this condition may become unreachable (even by humans) due to the subjective nature of voting.

Regardless of the exact evaluation metric, we can identify an objective set of advantages and drawbacks of the two bots. In the past two editions of BotPrize, MirrorBot has demonstrated the advantage of online mirroring, smooth navigation and simultaneous activation of several behavior modules which led to complex behavior using simple rules. However, the rules were a result of a lengthy fine-tuning process by its creator. Therefore, improving its behavior or porting it to different scenarios will be cumbersome using the same approach.

On the other hand, NizorBot was obtained using minimal expert intervention, by using an automated evolution process. Starting from botlike behavior, NizorBot evolved to be a competitive alternative to MirrorBot. However, unlike MirrorBot, it only exploited a limited set of techniques and did not focus on online interaction.

To sum up, the two approaches are able to complement each other towards the goal of obtaining an even more humanlike bot which could also be applied to contexts other than UT2K4.

Future Lines of Improvement for Obtaining a 'Human' Bot

There are several lines of improvement in both bots.

A direct first approach would be the *application of evolutionary computation (EC) methods to MirrorBot* in order to generalize its currently context-specific implementation. Considering that MirrorBot has many 'hard-coded' parameters, such as aiming speed, view distance, weapon and target selection policies or mirroring candidate selection, this would try to optimize their values. In the near future we intend to study the possibility of optimizing these parameters using an Interactive Evolutionary Algorithm (IEA) as done in NizorBot.

Another EC technique that could be implemented is a Genetic Programming (GP) (Koza 1992) approach, due to the additional flexibility factor that this method offers, *i.e.*, GP is able to create new sets of behavioral rules, given a set of possible inputs (conditions) and a set of possible actions.

However, one of the main problems of an IEA is the fatigue of the human user that is produced by the continuous feedback that the subjacent EC technique demands (Cotta and Fernández Leiva 2011). A form to deal with it is to promote the *algorithmic proactivity* in the sense that there would be a mechanism that predicts or infers the further user interactions with the goal of reducing the requirement of user interventions. To this aim, historic decisions of the expert are considered.

Another approach in this line consists in *computing the decisions as a function of the similarity of individuals*. Thus, the expert would assign a humanness score to an individual, for instance, and the score would be computed for the rest of the population considering the similarity to that individual. This could be also done with regard to the parts blocked in the chromosomes according to the expert's criteria.

A different improvement (and thus, research) line intends to obtain a *more comprehensive evaluation* by considering multiple aspects of humanness, and to study several evolutionary solutions using a larger behavior repository (including mirroring behavior) which may result in superior bot versions for various contexts.

Last but not least, defining what is "humanlike" might be harder than to recognize what is "botlike". By using techniques applied in NizorBot to evolve a bot that avoids botlike behavior, which was the actual development process of MirrorBot, we may partially remove the need for human-based evaluation in constructing humanlike bots.

Anyway, also considering the human interaction in the loop, we could let the evaluators the capability of identify or 'mark' somehow in a video showing a bot's behavior the exact moments when he/she has recognized a bot-like behavior. Combining this tool with a very deep log file storing all the information about the bot and the environment we could identify the botlike actions and potentially design an automatic learning system for avoiding doing them.

Acknowledgements

This paper has been partially supported by project V17-2015 of the Microprojects program 2015 from CEI BioTIC Granada, Junta de Andalucía within the project P10-TIC-6083 (DNEMESIS), by Ministerio español de Economía y Competitividad under project TIN2014-56494-C4-1-P (UMA-EPHEMECH), and Universidad de Málaga. Campus de Excelencia Internacional Andalucía Tech. This material is based upon work supported by the ENIB mobility program.

Conclusions

We have presented two models to build a believable bot: one based on mirroring the behavior of the human, and another based on interactive genetic algorithms. These approaches have been evaluated in the BotPrize competition as benchmark. Design features of each approach were correlated with the results they obtained. We have argued that the current methodology for evaluating humanness can still benefit from further research, and that context is a critical factor in this type of assessment. We have also discussed the advantages and shortcomings of MirrorBot and NizorBot, and concluded that a mixed approach has the potential of producing a more humanlike bot, which may also be transferable to scenarios other than the BotPrize competition.

References

Cotta, C., and Fernández Leiva, A. J. 2011. Bio-inspired combinatorial optimization: Notes on reactive and proactive interaction. In J. Cabestany et al., ed., *IWANN'11*, volume 6692 of *LNCS*, 348–355. Springer.

Hingston, P. 2009. A turing test for computer game bots. *Computational Intelligence and AI in Games, IEEE Transactions on* 1(3):169–186.

Jiménez, J. L.; Mora, A. M.; and Fernández-Leiva, A. J. 2015. Evolutionary interactive bot for the FPS unreal tournament 2004. In *Proceedings 2nd Congreso de la Sociedad Española para las Ciencias del Videojuego, Barcelona, Spain, June 24, 2015.*, 46–57.

Koza, J. R. 1992. *Genetic Programming: On the programming of computers by means of natural selection*. Cambridge, MA: MIT Press.

Mora, A. M.; Aisa, F.; García-Sánchez, P.; Castillo, P. Á.; and Guervós, J. J. M. 2015. Modelling a human-like bot in a first person shooter game. *IJCICG* 6(1):21–37.

Parmee, I. C.; Abraham, J. A. R.; and Machwe, A. 2008. User-centric evolutionary computing: Melding human and machine capability to satisfy multiple criteria. In *Multi-objective Problem Solving from Nature*, Natural Computing Series. Springer Berlin Heidelberg. 263–283.

Polceanu, M. 2013. Mirrorbot: Using human-inspired mirroring behavior to pass a turing test. In *CIG'13*, 1–8. IEEE.

Takagi, H. 2001. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE* (9):1275–1296.

Yannakakis, G., and Togelius, J. 2014. A panorama of artificial and computational intelligence in games. *Computational Intelligence and AI in Games, IEEE Transactions on.*