Designing and Evolving an Unreal Tournament™ 2004 Expert Bot

A.M. Mora, F. Aisa, R. Caballero,
P. García-Sánchez, J.J. Merelo, and P.A. Castillo

Departamento de Arquitectura y Tecnología de Computadores.
Universidad de Granada (Spain)
{amorag,jmerelo,pgarcia,pedro}@geneura.ugr.es,
francisco_aisa@hotmail.com, rcabamo@gmail.com

Abstract. This work describes the design of a bot for the first person shooter Unreal Tournament™ 2004 (UT2K4), which behaves as a human expert player in 1 vs. 1 death matches. This has been implemented modelling the actions (and tricks) of this player, using a state-based IA, and supplemented by a database for 'learning' the arena. The expert bot yields excellent results, beating the game default bots in the hardest difficulty, and even being a very hard opponent for the human players (including our expert). The AI of this bot is then improved by means of three different approaches of evolutionary algorithms, optimizing a wide set of parameters (weights and probabilities) which the expert bot considers when playing. The result of this process yields an even better rival; however the noisy nature of the fitness function (due to the pseudo-stochasticity of the battles) makes the evolution slower than usual.

1 Introduction

A *bot* is an autonomous agent which tries to behave as a human player, normally fighting against some other humans/bots in a computer game. There are different types of bots in addition to the opponents, such as the cooperative players (try to aid the main player). They essentially refer to an Artificial Intelligence (AI) engine which is able to perform the same actions in a game as a human player.

First Person Shooter games (FPS) are one of the most profitable area in the study/implementation of bots due to their usual open-code philosophy. They are games where the player can only see the hands and the current weapon of his character, and has to fight against enemies by shooting to them. They usually have a multiplayer fighting mode placed in a limited arena.

Unreal game series follows this philosophy to make it easy the game-modding (modification), including with every copy of the game an editor (UnrealEd), an own programming language (UnrealScript), and a compiler, to add/change almost whatever the user desires: scenarios, items, characters; just with a few constraints. The state-based AI engine is also open.

Moreover, some additional tools have arisen some years ago, such as GameBots[1], a *mod* that allows the control of characters (bots) in the game through

[1] http://gamebots.sourceforge.net/

network connections to other programs. The game sends character's sensory information to the client program, which can decide the actions the bot will take. These actions are sent back to the game which interprets them for the bot movement, shooting, jumping, etc. It was initially released for the Unreal sequel *Unreal Tournament™ (UT)* and later implemented for *UT 2003*. Over the basis of that tool, it was later launched *Pogamut*[2], which defines an interface (using GameBots architecture) to program the bots externally using Java. It was implemented for Unreal Tournament™ 2004 (UT2K4).

These open-code possibilities and external tools are the main reasons why this game environment has been widely considered in the computational intelligence researching field [1–3].

This work is also embedded in the UT2K4 environment, and uses Pogamut for implementing a bot which we have named *UT Expert Bot (E-Bot)*. Its behaviour (AI) follows a shape of Finite State Machine (based in two levels of states), that describes a complex set of rules. These rules are based in the knowledge of an UT2K4 Spanish expert player and are focused on *1 vs 1 Death Match battle*. It has been modelled according to his experience, including several tricks, as the humans players do, for getting a better behaviour in the game. We have considered the rules defined in the official competitions of UT2K4 for human players, such as: there are some items forbidden (U-Damage), weapons are not respawned and the match runs for 15 minutes, not for a number of frags (number of enemies killed).

Once *E-Bot* has been completely defined and tested (it beats the default game bots in the hardest difficulty levels), we have designed some evolutionary approaches based on the application of a Genetic Algorithm (GA)[4] for optimizing the set of parameters (weights, probabilities, thresholds) which the rules of behaviour apply, following previous approaches [3, 5], but considering an own created behavioural model as the main difference with regard to these works, where the standard AI in the game was improved.

A GA is an algorithm where a population of possible solutions (called individuals) are evolved by means of selection and recombination/mutation to create a new set of candidates that compete using their fitness (quality of adaptation) with the rest of solutions, until a stop criterion (i.e. number of generations) is met. Fitness function is a quality function that gives the grade of adaptation of an individual respect the others. This function usually models the problem to solve.

2 State of the Art

Bots have been widely used in games since 1992, when Quake™ became the first known game in including autonomous characters. It presented not only the option of playing against machine-controlled bots, but also the possibility to modify them or create new ones. Some AI approaches arose such as the SOAR Quakebot [6], which modelled a human-like behaviour.

[2] `http://pogamut.cuni.cz/main/tiki-index.php`

Unreal™ appeared some years later, being as stated in the previous section, the first game in including an easily programming environment and a more powerful language, thus plenty of bots were developed. However just a few of them applied metaheuristics or complex AI techniques, and most of these bots were based in predefined hard-coded scripts.

The studies involving computer games and the improvement of some components of the characters' AI appeared some years ago [7], but the use of metaheuristics to study (and improve) specifically the behaviour of the bots inside FPSs, have arisen in the last few years. We started our research in this field in 2001, publishing our results in national conferences (in Spanish). We applied a Genetic Algorithm to improve the parameters in the bots AI core (as later several authors did [8]), and to change the way of controlling the bots by automatically redefining, by means of Genetic Programming, the standard set of rules of the main states.

Some other evolutionary approaches have been published, such as [9], where evolution and co-evolution techniques have been applied, or [2] in which an evolutionary rule-based system has been applied. The two latter developed inside the Unreal Tournament™ 2004 environment (UT2K4).

There are several studies involving other techniques, such as Self-Organizing Maps, Machine Learning, or Neural Networks, to cite a few. Another line of research involves the human-like bots (which try to imitate the human's behaviour) [1]. There is even a competition which searches for the 'most human' bot [10] in UT2K4.

Our work is devoted to present and study two ideas: first a human-like bot, based in a human expert knowledge modelling is described; second the bot is improved by means of evolutionary computation, as in previous works [3, 5], but starting from an own created AI engine, not the standard one.

3 UT2K4 Expert Bot

The design of the expert bot (*E-Bot*) AI is based in the knowledge of an expert (human) player, *Me$$!@h*, who belongs to the best Spanish UT2K4 clan, *Trauma Reactor* (*dRâ*), and who has participated in some European championships. He is one of the authors of the paper and has designed and implemented the main work flow of the bot's behaviour. As a summary, *E-Bot* is implemented considering hierarchical states (primary and secondary), a set of rules to decide the correspondent states, and a (simple) database.

Some tips have to be taken into account as part of the **expert knowledge** to model. Thus, the items and weapons are critically valuable in a *1 vs 1 Death Match*. It is even more important to consider the respawn *timing* of each of them, i.e. the time it takes to appear again once an item has been picked up. The importance grows in this mode since if one player picks one item/weapon, he/she prevents the opponent for profiting it (while he/she does, of course). Expert players must control this time in order to move to a known respawn area in the moment an item appears. These conditions (timing and locations) depend on the map, which the player should know (or learn).

The main items include: *Shields* (provide protection points), *Health packs* (health points), and *Adrenaline* (which can provide the player special states such as invisibility or faster movement).

Weapons in UT2K4 have been 'carefully' designed to be all of them equally useful and important (as a difference to other games), so there is no absolutely better weapon than the rest. Everyone has a perfect moment to use and take advantage over other weapons. The choice of this moment is an expert decision that depends on several factors such as the player's and enemy's health, the distance to enemy or the height/level where it is placed. The considered weapons are: *Shield Gun*, *Assault Rifle*, *Link Gun*, *Bio Rifle*, *Minigun*, *Rocket Launcher*, *Flak-Cannon*, *Shock Rifle*, and *Lightning Gun*.

The *movement* is another key factor in UT2K4, since it could mean the difference between win or lose. Players usually move jumping, in order to avoid being shot easily. This is a problem of our approach because Pogamut movement module does not implement the jump.

The proposed E-Bot uses a 'simple' **database**, which stores the location of every (new) item it founds while moving around the maps (searching for enemies or just exploring to learn), as the human players do the first times in a new arena. Additional information could be stored, but we think this is enough for this initial version.

The use of Pogamut implies we cannot consider the FSM defined in the UT2K4 bot's AI, thus, a complete AI engine has been designed and implemented. However, we have profited Pogamut's high-level functions, such as basic navigation from point to point in a map, or the so-called *listeners* (triggers for detecting events).

The *E-Bot*'s AI module considers two levels of states: primary and secondary. The first ones correspond to general actions to perform: Attack, Hunt, Retreat, Greedy and Camp in this initial version. The secondary states are devoted to perform additional tasks within the main action flow (offensive and defensive profiles, pickup items or weapons, among others). For instance, the bot can decide attacking to the enemy, but if it has low health search for any health pack at the same time.

The comparisons and decisions required to choose the state are quite complex, because they consider several factors and situations that the human expert knows. For instance, the comparison of weapons depends on several parameters and weights, because of the existent balance between the power and usefulness of all of them, which could mean that a weapon is the best in a specific situation (different levels, hidden opponent, close/far enemy), but that weapon would be the worst on a different position.

The AI engine (composed by a huge system of rules) at a time chooses the primary state, but while the bot is performing the actions associated to it, the engine continues checking conditions (for instance the timing of every item), receiving sensory information, checking the bot status, etc. This way, a secondary state can be also set. However the engine can stop and change, at any time, both the primary and/or the secondary state, depending of the conditions, received information and status, giving an extra 'flexibility' level to the FSM.

As a general summary, the bot tends to be defensive if its health is lower than the enemy's, unless the latter has a critical health level, so the bot will attack him to win a frag. In similar health conditions the attitude depends on the weaponry (in comparison with the enemy's weapons). If the bot's health is good, it is quite aggressive.

The value of the expert bot has been tested in an experiment, consisting in four different *1 vs 1 - Death Match* combats against a standard UT2K4 game bot, held in two different maps. The bots have been fighting, respecting championship rules, during 15 minutes. The results of every match as well as the average are in Table 1.

Map	*E-Bot*	*StdBot*
DM-Ironic	13 (m1), 26 (m2)	6 (m1), 8 (m2)
DM-Idoma	23 (m1), 25 (m2)	15 (m1), 8 (m2)
Average	21.75	9.25

Table 1. Scores (number of frags) of the test fights between *E-Bot* and a Standard UT2K4 bot in the hardest difficulty (*StdBot*). Match 1 (m1) and 2 (m2) in every map.

As it can be seen *E-Bot* outperforms the standard game bot, even in the hardest difficulty level, which is a challenge for a medium level player.

4 Evolution of the UT2K4 Expert Bot

Following previous approaches [8, 3, 5], the expert bot will be improved. To this end, the set of parameters which act as thresholds, weights, priorities and conditions, and which finally determine the final behaviour of the bot, will be optimized by means of a Genetic Algorithm (GA)[4], which can evolve their values searching for the best combination, i.e. the set of values which performs better in a battle. Following this idea, we have designed and tested a GA-based bot, named *GE-Bot*[3], following different approaches.

4.1 First approach: chromosome-143

In this approach the *chromosome scheme* has 143 genes, grouped in six different blocks: having 2 genes for health levels, 3 genes for distance, 5 genes to value the risk, 1 gene for time to see an enemy, 6 genes for items priority, and 126 genes related to weapons selection.

The *fitness function* has been defined as:

$$f = \begin{cases} 3 + (damP/damR) & \textit{if (frags + 1) = deads} \\ (2 \cdot frags - deads) + (damP/damR) & \textit{if frags > deads} \\ (3/2) + (damP/damR) & \textit{if frags = deads} \\ frags/deads & \textit{if frags < deads} \end{cases} \tag{1}$$

[3] The source code of both *E-Bot* and *GE-Bot* can be downloaded from `https://github.com/franaisa/EvolutionaryBot` under a GPL license.

Where *frags* is the number of enemy kills the bot has obtained, *deads* is the number of own deads, *damP* is the total damage produced by the bot, and *damR* is the total damage it has received. This function rewards the individuals with a positive balance (more frags than deads) and a high number of frags. In addition individuals which perform a high amount of damage to the enemies are also rewarded, even if they have not got a good balance.

The *evaluation of an individual* consists in setting the values of the chromosome in the *E-Bot* AI engine, then a 1 vs 1 combat is launched between this and a standard *E-Bot*. After the time limit defined, the fitness value is computed for the individual. There is a high pseudo-stochastic component in these battles, since the results do not depend completely on our bot, but also on the enemy's actions which we cannot control. Thus, the fitness function is considered as noisy [11], since an individual could be valued as good in one combat, but yield very bad results in another match.

An *elitist selection mechanism* has been applied, remaining the best four individuals in the next population. The rest of population is considered as parents.

The *uniform crossover* operator (every gene of a descendent has the same probability of belonging to each one of the parents) has been applied in two steps: on the one hand it is used with the elite, yielding two descendents by combining the best individual with a random one in the best four chosen. On the other hand, the rest of the offspring is formed as a random combination of the rest of parents (in pairs). Finally, four random individuals are included in the population (they substitute the four worse) to add diversity.

The *mutation mechanism* is performed on every descendent. It changes with a probability $1/chromosome\_size$ the value of every gen in a $\pm10\%$ rate.

Some experiments were conducted to test this approach for the *GE-Bot*. All of them (and next) have been conducted in the map DM-Ironic, since it is one of the most used maps in competitions, and has a design that offers almost any possible fighting situation: different levels, hidden zones, viewpoints and camping areas.

Firstly, the algorithm is tested considering as parameter setting: *30 generations, 30 individuals, and 15 minutes every evaluation*, taking 10 running days.

Figure 1 shows the best and average fitness evolution in the run, considering one (left side) and three (right side) evaluations per individual.

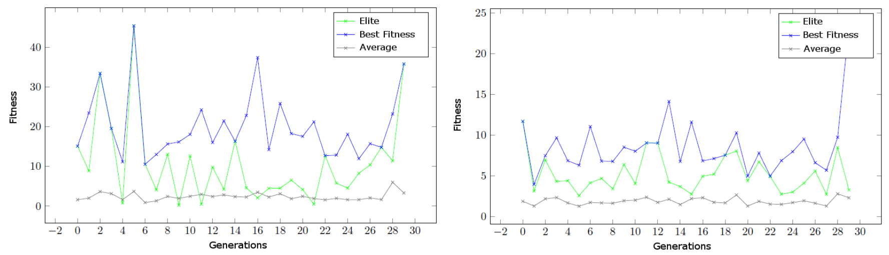

Fig. 1. Fitness evolution of the best, elite and average in 30 generations for the first approach of GE-Bot (143 genes chromosome, 1 evaluation (left) and 3 evaluations (right) per individual).

As it can be seen on the left subfigure, the evolution is highly noisy, as commented when the fitness function was defined, due to the pseudo-stochasticity of combats. In order to deal with this effect, some evaluations (different matches) per individual have been performed, as it is recommended [11]. This second experiment has taken one month of computational time and is plotted on the right side subfigure. An improvement in the fitness evolution tendency is shown.

The best fitness graphs proved the noise nature of the function, so even considering three matches for every evaluation, and having followed an elitist scheme, the best values oscillate without a clear tendency. The average results show a lightly improvement tendency, which let us think in a good (but slow) evolution and improvement of individuals (parameters) or results. The reasons of these light tendencies and high oscillations might be two: first reason could be the chromosome length, since the evolution of 143 genes might take much more generations; the second reason could be the inclusion of diversity enhancement mechanisms, such as the uniform crossover or the random generation of some individuals (which substitute the four worst).

4.2 Second approach: chromosome-26

Since the chromosome length implies a high number of generations (and so, huge computational time) are required for a good optimization process, the *chromosome scheme* was redefined considering 26 genes, by reducing the weapons block to just 9 genes (one associated to the priority of each weapon). This representation is less precise from the expert's AI performance point of view, but with a more approachable size. The rest of the algorithm terms and operators remained the same. Results are presented in Figure 2 (two runs of ten), with a different parameter setup, due to the expected computational time reduction, so, in order to perform a more complete evolution, *the number of generations has been increased to 50, being the population again 30 individuals and considering this time 5 minutes for evaluating every individual.* The running time of this experiment has been 5 days per run.

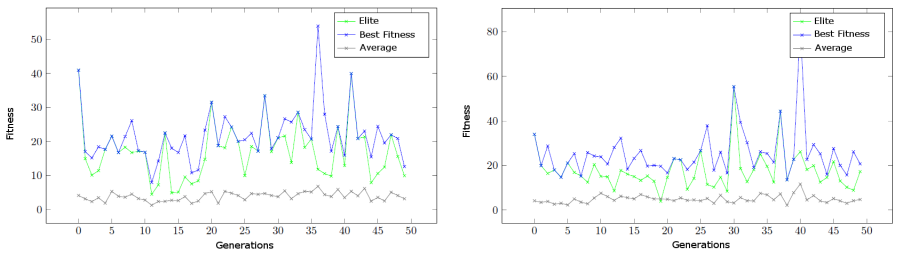

Fig. 2. Fitness evolution of the best, elite and average in 50 generations for the second approach of GE-Bot (26 genes chromosome). Two different runs.

As it is shown, there are again fluctuations and light improving tendencies. The oscillations seem to be lighter, but what is happening is that the resulting values have been reduced (due to the shorter battle time). The average fitness tendency on the other hand is less clear this time. Thus, the reduction in the evaluation time just has meant an improvement in the computational cost.

4.3 Third approach: fitness and operators redefinition

According to the second conclusion reached by the end of Subsection 4.1, the evolution in this problem is quite difficult, so it would be recommended the implementation of mechanisms to increase the exploitation of solutions rather than increasing diversity (as it has been done).

Firstly, the fitness function has been redefined in order to take into account more elements of the individual/bot performance during a match, i.e. the most important items for survive and the best weapons for our bot (the most useful according to the defined FSM). They have been included in some terms of the evaluation function as follows:

$$f = \begin{cases} (frags - deads) + s2 + \frac{s1}{2} + \frac{tS/10}{d+1} \\ + \frac{tL/10}{d+1} + \log((damP - damR) + 1) \quad \textit{if } frags \geq deads \\ \\ \frac{frags}{deads} + s2 + \frac{s1}{2} + \frac{tS/10}{d+1} \\ + \frac{tL/10}{d+1} + \log((damP - damR) + 1) \quad \textit{if } frags < deads \end{cases} \tag{2}$$

Where $frags$, $deads$, $damP$ and $damR$ are the same as in Equation 1. $s1$ and $s2$ refers respectively to the number of Shields and Super Shields the bot has picked up. tS is the time the bot has used the Shock Rifle, and tL refers to the time it has used the Lightning Gun. The term frags is the most important and the rest are weighted to have lower relevance.

The GA scheme has been changed to a stationary approach in order to increase the selective pressure and get a higher convergence factor. Thus the best 15 individuals are considered for the new population. They also are one of the parents for generating the offspring and the other parents are selected using a probability roulette wheel from the rest of population. The crossover and mutation operators remains the same as in the other approaches.

Figure 3 shows the results of two runs (of ten), considering the same parameter configuration as in the last approach.

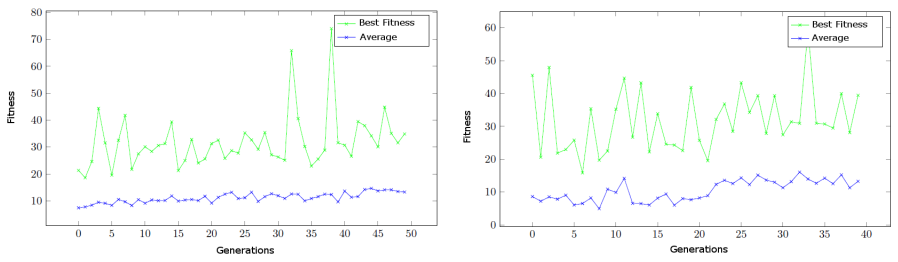

Fig. 3. Fitness evolution of the best, and average in 50 generations for the third approach of GE-Bot (26 genes chromosome, complex fitness function and stationary scheme).

As it can be seen, the results are better than in previous approaches, showing a 'softer' fitness tendency (with less fluctuations), and a clear improving, with the generations, in the average results.

5 Expert Bot versus Evolved Bots

A last test has been performed confronting the best individuals of the *GE-Bot* approaches against *E-Bot*, since they have been optimized for beating it (they have fought against it during the evolution). The average results of four matches are shown in Table 2. They have been held in the same two maps and for 15 minutes each.

	Score		
GE-Bot: 143chr-30gen-3evals	4.2	13.2	E-Bot
GE-Bot: 26chr-50gen-1evals	15.3	7.8	E-Bot
GE-Bot: 26chr-50gen-1evals-complexfitness	17.5	6.2	E-Bot

Table 2. Average scores (number of frags) of the test fights between the three approaches of *GE-Bot* versus *E-Bot*. 4 battles in two maps: DM-Ironic and DM-Idoma.

The results show that the *GE-Bots* with a higher number of generations and smaller chromosome perform better than the one with the higher length. This result should be expected, since the chromosome length is a key factor in the evolution, because it defines the search space size, so long individuals need a higher number of generations. In this case, we just have performed (due to the huge computational time required) 30 generations for 143 genes which are clearly insufficient, according to the experiments results. The evolved for longer *GE-Bots* (50 generations) yield a very good performance beating *E-Bot* in average. Finally, the last approach considering the complex fitness function performs better than the rest, because it is less influenced by the noisy nature of the evaluation function due to the consideration of additional (and important)factors to compute it.

6 Conclusions and Future Work

In this work, a human-like bot for the First Person Shooter game Unreal Tournament™ 2004 (UT2K4) has been described. It models the expert knowledge of a Spanish high-level player so, its has been named Expert-Bot (*E-Bot*). Its behaviour is based in a shape of finite state machine with two levels of states (primary and secondary), each of them with an associated priority for performing actions. This work flow is flexible, so the AI engine can alter/change it depending on the conditions or status (bot's or enemy's). Moreover, the bot uses a database for 'learning' the maps, as a human player would do the first time it navigates around a new arena. *E-bot* has been designed for fighting in 1 vs 1 death match mode, considering the official competition rules.

Then a Genetic Algorithm (GA) has been implemented to improve the parameters (thresholds, weights, priorities) that the bot's AI considers in its behavioural rule system. The evolved expert bot, named *GE-Bot* has been tested considering different approaches (chromosome length, number of generations, scheme, fitness function, crossover), along with an evaluation consisting in repeated battles which are rated once they have finished, for assigning an average fitness value to every individual in the population. The aim is to deal with the noisy nature of the fitness function, since it depends on the results of the battles, which can vary for the same individual from time to time, regarding to the situation in the map or the enemy's actions which we cannot predict.

Some experiments have been conducted, firstly proving that *E-Bot* outperforms the standard UT2K4 bots, even in the hardest difficulty level. The result-to-effort ratio for *GE-Bot* is not very good, since the extremely high computation time that every experiment takes (from 5 to 30 days), has limited the number of generations and runs performed, so the improvement is not substantial. However, the results yielded show that the best approaches are those which consider a smaller chromosome size, and a higher number of generations, rather than wasting time in repeated evaluations. Moreover, the approach following a stationary scheme and a fitness function that considers items and weapons use gets the best performance and results. Both approaches have improved the results of *E-Bot* and have beaten it in a set of test battles. However, the evolution is not as good as desired and some repetitions of the evaluations could definitely help to improve this tendencies. We will consider this as the first future work line.

According to the expert's opinion, the bots do not perform as well as expected (from the behavioural point of view). This way, another line of research will be the improvement of *E-Bot* in its main flaw, the movement. Thus, the possibility of jumping must be included in its actions, in order to get a harder opponent (more difficult to kill). Once the behaviour has been improved, several research could be conducted with regard to the evolutionary approaches, since the results in this paper can be considered as preliminary. A better noisy fitness dealing could be an interesting issue to address.

Finally, another option to study could be the optimization of *E-Bot* for multiplayer and team modes, which are so far the most famous among the players.

References

1. Soni, B., Hingston, P.: Bots trained to play like a human are more fun. In: IEEE International Joint Conference on Neural Networks, IJCNN'08. (2008) 363–369
2. Small, R., Bates-Congdon, C.: Agent Smith: Towards an evolutionary rule-based agent for interactive dynamic games. In: IEEE Congress on Evolutionary Computation 2009 (CEC'09). (2009) 660–666
3. Mora, A.M., Montoya, R., Merelo, J.J., García-Sánchez, P., Castillo, P.A., Laredo, J.L.J., Martínez, A., Esparcia, A.I.: Evolving bot AI in Unreal. In et al., C.D.C., ed.: Applications of Evolutionary Computing, Part I. Volume 6024 of LNCS., Springer (2010) 170–179
4. Goldberg, D.E.: Genetic Algorithms in search, optimization and machine learning. Addison Wesley (1989)
5. Mora, A.M., Moreno, M.A., Merelo, J.J., Castillo, P.A., Arenas, M.G., Laredo, J.L.J.: Evolving the cooperative behaviour in unreal; bots. In Yannakakis, G.N., Togelius, J., eds.: CIG, IEEE (2010) 241–248
6. Laird, J.E.: It knows what you're going to do: Adding anticipation to a quakebot. AAAI 2000 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment SS-00-02 (2000)
7. Laird, J.E.: Using a computer game to develop advanced AI. Computer (2001) 70–75
8. Cole, N., Louis, S.J., Miles, C.: Using a genetic algorithm to tune first-person shooter bots. In: Proceedings of the IEEE Congress on Evolutionary Computation 2004. (2004) 139–145

9. Priesterjahn, S., Kramer, O., Weimer, A., Goebels, A.: Evolution of human-competitive agents in modern computer games. In: IEEE World Congress on Computational Intelligence 2006 (WCCI'06). (2006) 777–784
10. 2K-Games: The 2k botprize competition (2012) http://www.botprize.org.
11. Mora, A.M., Fernández-Ares, A., Merelo, J.J., García-Sánchez, P.: Dealing with noisy fitness in the design of a rts game bot. In et al., C.D.C., ed.: EvoApplications. Volume 7248 of Lecture Notes in Computer Science., Springer (2012) 234–244