Game Designers Training First Person Shooter Bots

Michelle McPartland and Marcus Gallagher

University of Queensland
{michelle,marcusg}@itee.uq.edu.au

Abstract. Interactive training is well suited to computer games as it allows game designers to interact with otherwise autonomous learning algorithms. This paper investigates the outcome of a group of five commercial first person shooter game designers using a custom built interactive training tool to train first person shooter bots. The designers are asked to train a bot using the tool, and then comment on their experiences. The five trained bots are then pitted against each other in a deathmatch scenario. The results show that the training tool has potential to be used in a commercial environment.

Keywords: Reinforcement learning, interactive training, first person shooters, game artificial intelligence.

1 Introduction

Over the past decade there has been a dramatic increase in using computer games for Artificial Intelligence (AI) research. In particular, first person shooter (FPS) games are an increasing popular environment in which to test traditional machine learning techniques due to their similarities to robotics and multi-agent systems (MAS). For example, FPS game agents, termed bots, are able to sense and act in their environment, and have complex, continuous movement spaces. FPS bot AI generally consists of hard-coded techniques such as finite state machines, rule-based systems, and behaviour trees [1][2]. These techniques are generally associated with drawbacks such as predictable behaviours [3] and time consuming tuning of parameters [4].

Reinforcement Learning (RL) is a class of machine learning algorithms which allow the agent to build a map of behaviours by sensing and receiving rewards from the environment. An extension to the standard RL algorithm is interactive RL, or interactive training, a method that allows human users to interact with the learning algorithm by providing rewards or punishments instead of using a fixed reward function.

While there is an increasing amount of research into using machine learning techniques for FPS bot AI [5-7], this research removes the game designers from the development loop. While automating the learning of AIs may appear to be an advantage, it is hard to control the direction the algorithm learns in [8], and is also hard to fine tune to commercial standards. This research attempts to bridge this gap by allowing game designers to interact with the underlying learning algorithm in order to direct what behaviours the bot learns. There are a number of advantages of using

M. Thielscher and D. Zhang (Eds.): AI 2012, LNCS 7691, pp. 397–408, 2012.

interactive training for FPS bot AI. It gives designers control over the types of bots that are made. The underlying code of the behaviours can be modularised and reused thereby decreasing bugs and coding time. Finally, iteration time designing bots is decreased as the designers can see in real-time the effects of the bot behaviours while the game is being played.

This paper extends previous work in interactive training [9] with the aim of further investigating its suitability for commercial first person shooter games. The research is continued through experiments involving commercial computer game designers using the tool to train a bot. This aim will be achieved by examining the results from the training session, to see if they match the intention of the designers. The secondary aim is to prove that diverse types of bots can be created by different users.

The contribution of this paper is twofold. Investigating how human users interact with learning algorithms is an interesting and novel idea, and may provide insight into the underlying algorithm itself. The results from the paper also benefit the game industry as a new method for designing and training bot AIs may be established.

This paper is organized as follows. Section 2 will provide background on FPS games and relevant research. An introduction to RL and interactive training will then be given. Section 3 will outline the game test bed used for the research, including the interactive training tool interface. Section 4 presents the data gathered from game designers' training sessions and results from playing the trained bots against each other. The final section concludes the paper with ideas on future work.

2 Background

FPS games are one of the most popular types of game being played in the current market [10]. FPSs are categorized for their combat nature and fast paced action. These games are generally made up of bots that navigate the environment, shoot at enemies, and pick up items of interest. AI research using FPS games has gained considerable attention in the research [6][11].

During the last decade, research into FPS games has continued to increase. A Neural Network (NN) was used to train the weapon selection module of Unreal Tournament bots [5]. The results showed that the bots trained against the base AI had improved performance, and the bots trained against the harder AI were not as competitive, but had improved slightly. In a similar environment an evolutionary NN was used to train hierarchical controllers for FPS bots [6]. Three controllers for shooting, exploring and path following were evolved individually, and then an evolutionary algorithm (EA) was trained to decide when to use each of the controllers. The results showed that bots using the evolved controllers were not able to outperform the hard-coded full knowledge bots, but they were able to play the game quite well. A NN was also used to learn sub-controllers for the movement of an FPS bot and found that it was able to outperform controllers using decision trees and Naïve Bayes classifiers [11].

Game AI has many parameters which are usually balanced or fine tuned by developers once all the features of the game are complete. The job of tuning parameters can be time consuming due to the large number of them. Some researchers have attempted

to tune these parameters using genetic algorithms [4][7][12][13]. An FPS environment was used with the parameters being the behaviour of the bots [12][13][4]. Bots were tuned from Unreal Tournament (Epic Games) and found the tuned bots were better, in terms of kills, than the standard AI in the game [12]. The popular Counter Strike (Valve) game has been used as a test bed for FPS research [13]. They found that evaluation times were extremely long as the rendering could not be "turned off", and therefore only 50 generations were completed. Results showed that after only 50 generations the evolved bots had a slight advantage. Other research evolved the behaviour of bots in an open sourced FPS engine called Cube [4]. Evaluation was performed by the author manually playing against the bots with an evaluation function consisting of the bots health, kills and deaths. Results showed that the bots evolved into capable enemies from initial useless ones.

Reinforcement learning (RL) is a type of action selection mechanism where an agent learns through experience interacting with the environment [14]. The field of RL is widespread but is mainly focused in robotics and multi agent systems [15][16]. Sarsa(λ) is a type of RL algorithm where the policy is updated after the next action is selected using the rewards obtained from the current and next action. The Sarsa(λ) algorithm has been successfully applied to multi-agent systems using a computer game environment [17][18].

Previous work has been performed in using RL for learning individual bot behaviors in a shooter style environment [8][19]. Both of these research papers show that RL can successfully be used to learn navigation, item collection and combat behaviors. A simplified FPS test bed has also been used in other research, where the map was broken into square areas, and each area represented one of the states, along with enemy visible and a set of plans which denoted a planned path through the areas [19]. This work looked at using RL for learning bot behaviours in FPS team scenarios.

Interactive RL is an extension to the standard RL algorithm as it allows a human user to interact with the reward process and the action selection process. There is little research in the literature on interactive RL, and none in an FPS environment. An interactive RL algorithm was used in a simple 2d environment to train a synthetic dog how to behave [20]. The user was able to guide the dog by using the mouse to lure the dog into positions such as sitting or begging. An extension to this work is seen in [21] which used a more complex environment to teach a virtual character how to make a cake. Reward values were represented with a sliding bar, and guide actions were used in the form of clicking an object in the environment. The research presented here extends previous work on interactive RL [9], which performed a preliminary investigation into using the algorithm in an FPS environment. From the success of the previous work, this paper will test the interactive training tool on five commercial game designers, and will compare the results of the trained bots.

3 Method

A purpose built FPS game environment was used for the interactive training experiments as full control was needed over the game update loop, the user interface and the

CPU cycles. The game environment consisted of the basic components of an FPS game: bots that can navigate the environment at different speeds, shoot weapons, pick up items, strafe, and duck behind cover. See Figure 1 for a screenshot of the game environment with four bots currently in combat. For more details on the game environment refer to [9].

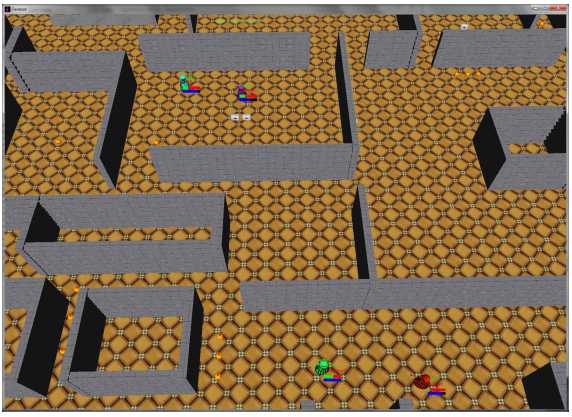

Fig. 1. Screenshot of game environment with bots playing against each other

The state sensors of the bot were designed to capture local information that the bot can use to sense the environment. The input states for the bot are as follows:

- Health: Low (0), Medium (1), High (2)
- Ammo: Low (0), Medium (1), High (2)
- Enemy: NoEnemy (0), Melee Range (1), Ranged (2), Out of Attack Range (3)
- Item: None (0), Health (1), Ammo (2)

The output states were the actions the bot can perform in the world as follows: Melee (0), Ranged (1), Wander (2), Health Item (3), Ammo Item (4), Dodge (5), and Hide (6). Therefore the number of state action pairs in the policy table is 756.

The ITRL algorithm used in this paper was loosely based on the work on interactive synthetic dog [20], and human training [21]. The algorithm runs as normal when no human input is recorded using a pre-defined reward function. The reward function can be modified by the user at any stage of the training through edit boxes in the User Interface (UI) (see Figure 2). A button was also available to clear all reward values, which will disable the pre-defined reward function and only user rewards are used.

Fig. 2. User Interface design for the interactive training tool

Table 1 lists the steps of the interactive training algorithm. If the user selected a guide action, the algorithm will use this input to override the RL action selection method. For a complete description of the algorithm and user interface design see [9]. The learning rate was represented by $\alpha = 0.1$, reduced over time to 0.05. The decay factor was $\gamma = 0.4$, the eligibility trace was $\lambda = 0.8$, and the exploration rate was $\varepsilon = 0.2$. The end game condition is the terminal state which was decided by the designers at any point during the training. Trained bots can be saved and loaded for continuation of training as well.

Table 1. Interactive Sarsa(λ) Algorithm

1:	Initialize $Q(s,a)$ arbitrarily, set $e(s,a)$=0 for all s, a
4:	Repeat for each update step t in the game
5:	$g \leftarrow$ guidance object
6:	If guidance received then
7:	$a \leftarrow g$
8:	Else
9:	$a \leftarrow$ action select a from policy, Q, using ε-greedy selection
10:	End if
11:	Execute a
12:	$hr \leftarrow$ user reward or penalty
13:	If user reward received then
14:	$r \leftarrow g$
15:	Else
16:	Observe r
17:	End if
18:	$\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
19:	$e(s,a) \leftarrow 1$
20:	For all s, a:
21:	$Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
22:	$e(s,a) \leftarrow \gamma\lambda e(s,a)$
23:	$s \leftarrow s'$, $a \leftarrow a'$
24:	Until s is terminal

The interactive training algorithm updates when a state change occurs and when the user has selected a guide action or reward. When the user selects a guide action, it immediately overrides the current action. The user chosen action continues until it either succeeds or fails, or the user selects another action.

Emails were sent to five game designers working in the computer games industry that have worked on commercial shooter games. The designers were asked to train a bot using the supplied training tool, and to email the results back along with answers to some questions regarding their training experience. Data was recorded for all user actions, including reward and penalty frequencies, and whether the guide actions failed or succeeded after the action was pressed.

4 Results

This section looks at the results from the five game designers using the interactive training tool to train bots to play a first person shooter game. Feedback was gathered from the users to find out what type of bot they tried to train. The first section will compare the feedback with the data from the training phase. The second section will investigate the results from the five user trained bots playing against each other.

4.1 Training Phase

Figure 3 displays the state-action value functions represented by a colour scale visualisation to show the similarities and differences between the trained bots. User 3 clearly has the most active policy with the majority of states having adjusted values, and very few areas where the state action pairs have not been visited. This activity

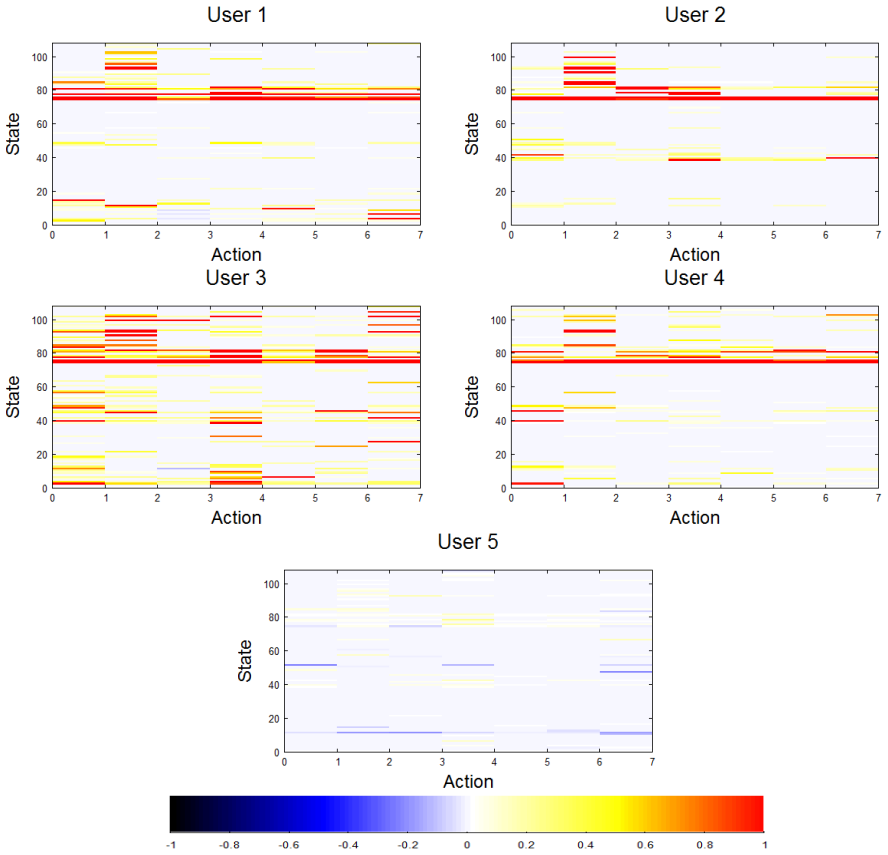

Fig. 3. Clockwise from top left to bottom right, the state-action value functions represented with a colour scale for user 1, user 2, user 3, user 4 and user 5

indicates that user 3 spent a lot of time training the bot and therefore the bot, in theory, should be more experienced at playing the game than the other bots. The next most active landscape is from user 5, although this is not immediately clear from Figure 3 as it appears very flat all over. The reason for the decrease in values is that user 5 turned off the automatic reward distribution at the beginning of the training session. Therefore all reward values were manually given by the user causing the smaller values. However, despite the small values, a good spread of clusters are seen in the landscape, with some flat areas but less than seen in the policies of users 1, 2 and 4.

Users 1, 2 and 4 have similar clusters; although their values are varied over the three users. For example, user 2 has higher peaks in the states over 100, and very low peaks in the states less than 20. Whereas, user 1 had high values in states less then 20, and small peaks in the ones greater than 100. User 4 generally had lower values, but across more state-actions, indicating a broader training experience with less repetition on similar states than other uses.

Table 2. Guide actions successes (S) and failures (F) for user training

Action	**S1**	**F1**	**S2**	**F2**	**S3**	**F3**	**S4**	**F4**	**S5**	**F5**
Melee	6	1	3	21	15	13	4	2	9	0
Ranged	2	8	2	3	3	1	3	5	6	15
Wander	0	0	0	0	0	0	1	0	0	0
Health Item	11	1	16	5	64	6	9	0	31	0
Ammo Item	6	0	1	3	0	0	2	2	14	0
Dodge	8	0	0	0	0	0	3	0	0	0
Hide	1	2	0	0	0	0	0	1	6	2
TOTAL	***34***	***12***	***22***	***32***	***82***	***20***	***22***	***10***	***66***	***17***

User 1 tried to create an item collecting bot that shot at range then moved into melee. Table 2 shows an even number of ranged and melee actions being used, with melee (six actions) being more successful than the ranged action (two successes). The health and ammo item actions were also evenly used with 11 health and six ammo item successes. Therefore, overall a general type bot was attempted to be trained.

User 2 tried to create an aggressive melee combatant and this can clearly be seen by the number of melee actions that were selected. Unfortunately a very high number of these guide actions failed, indicating that either the failure condition for the melee action was unreasonable (i.e. having to kill the opponent), or that the range for using the melee action was not clear to the user.

User 3 attempted a health collecting ranged/melee combination bot that favoured melee. The figures in Table 2 reflect what the user attempted to do. The melee action was focussed on with 15 successes and 13 failures, and the ranged action was selected less frequently with three successes and one failure. The health item action was used as a guide 64 times successfully, and six times unsuccessfully. These figures show that user 3 performed more interactive training than all the other users, which were also seen by the height field representation of the policy in Figure 3 as the landscape was more active, compared to the others.

User 4 aimed to create a bot that fled the stronger enemies but attacked the weak ones. The data does not reflect this training as the hide action was only selected once

by the user. However, this failure may be an indication of why the user felt the training did not work well for them. Improvements need to be made on the hide behaviour so that it is useful for the intended purpose as user 4 assumed.

The user with the second most active training session was user 5. Their feedback quoted them trying to create a bot that was primarily ranged but also used melee attacks, and attempted to collect health and ammo items. The guide action data back up this statement as the ranged attack was focused on with nine successes and 15 failures, while the melee attack was used nine times successfully. Health and ammo items were selected 31 and 14 times successfully successively. User 5 tried training the hide action more than the rest of the users with six successful attempts and two failures.

This section has shown that the users were able to use the interactive training tool to train the types of bots they wanted. The policy visualisations showed that there were three distinct types of bots that were trained, where user 1, 2 and 4 had similar trends, and user 3 and 5 had distinctly different trends. The next section will continue investigating the varied nature of the bots from the different users.

4.2 Simulation Phase

This section looks at the results from a game played with five trained bots, one from each user, fighting against each other. No AI controlled bots were included in the games. The simulations were run for 12000 game ticks or iterations. An iteration was a complete update cycle of the game and was used to be consistent over all replays. Due to there being multiple RL bots, using RL iterations would only be relevant to one of the five bots. 50 games were played and the results averaged. Figure 1 shows the five user trained bots playing against each other.

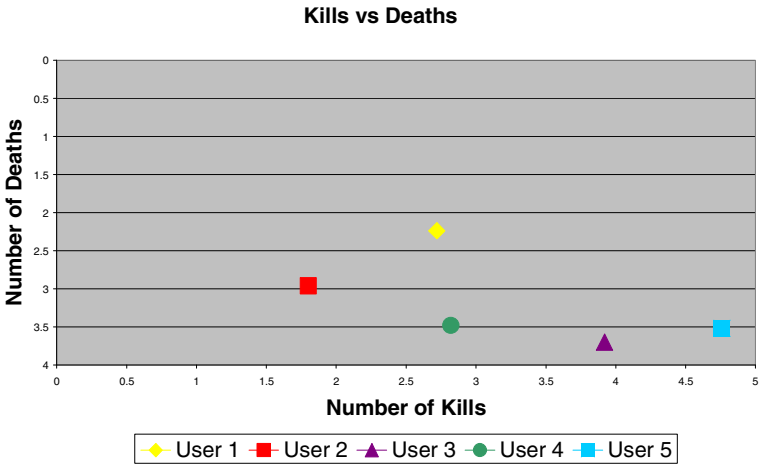

Fig. 4. Number of kills versus deaths for each user

Figure 4 maps the number of kills versus deaths to represent an overall combat strategy based on maximising kills and minimising deaths. The number of deaths scale on the Y axis was reversed as the best strategy has the lowest death count. The figure indicates that user 5 has the best combat strategy bot, being the only one in the first wave on the Pareto front. User 5 had the highest number of average kills of 4.8, and although they did not have the lowest number of deaths of 3.5, they were still able to dominate the other bots in combat. The next front had user 3 in it with an average of 3.9 kills and 3.7 deaths, with user 1 in the next front with 2.7 kills and the lowest number of deaths of 2.2. Users 2 and 4 were dominated by the other three bots with average kills of 1.8 and 2.8, and deaths of 3.0 and 3.5 respectively.

Observation of the games showed that user 1 was extremely competent in health item collection, and often avoided ammo items in favour of health. This behaviour is also reflected in the values recorded for health and ammo collection. User 1 was the second best at collecting health items with an average of 18.0 health items per game, whereas they were the second lowest in ammo collection at 8.7 items per game (see Figure 5a). This bot rarely used the wander behaviour, which corresponds to feedback from user 1 that they wanted their bot to always try to move with an intention.

User 3 followed a similar strategy to user 1 of favouring health items over ammo items. They were successful in this goal, and achieved the highest health collecting bot with an average of 23.0 health items per game. User 3 not only trained a very good health item collecting bot, but also a competitive combat bot, proving that the extensive training they did produced a bot that was well rounded in all the game objectives.

Users 2, 4 and 5 produced bots with a similar health collecting ability with averages of 6.0, 10.7 and 7.7 respectively. While these bots were capable at health item collection, they frequently chose different actions during non-combat states such as wander, and ammo collection.

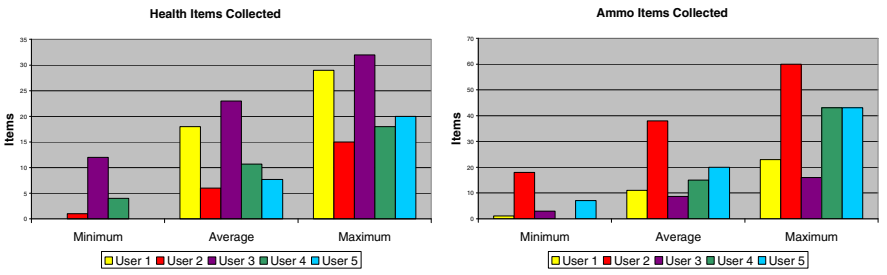

Fig. 5. (a) Average health items collected for user trained bots and (b) Average ammo items collected for user trained bots

Although user 2 had the lowest health collection rate, they scored extremely high in the ammo item collection task with an average of 38.1, almost double that of the next highest scoring bot from user 5 who had an average of 19.9 (see Figure 5b). Observation of one game showed user 2 spending a lot of time wandering around an area of the map with number of ammo items, which could be attributed to this very high number.

The increased activity seen in the policy height field representations of user 3 and 5, seemed to have paid off as these bots stood out in combat. Observation of the combat strategy of user 3 and 5's bots showed intelligible behaviours, and they would only break out of combat to collect health items. Bots from users 1, 2 and 4 had more erratic behaviour with rapid state changes and selection of strange actions during combat. For example, in the replay, user 1 is fighting user 2 and user 1 breaks out of the combat and wanders away even though the enemy is still in sight. They re-engage in combat after a time, but the behaviour looks erratic and is not what would be expected of a commercial FPS bot.

5 Discussion

The user trained bots have shown greater diversity in their policy landscapes and behaviours than in previous research of automatically trained bots [8]. The policy landscapes were especially varied in user 3 and 5's bots, both of which spent more time training than the other three users. The bots produced from the varied policies appeared better in their behaviours than the other three bots. An example of the diversity in bots can be seen by the bot user 1 trained, which was very good at health collection but not as competitive in combat, whereas user 3 also produced a very good health collecting bot and was also very good in combat.

One of the major concerns with the combat system was that the bots did not have the ability to shoot and move to a designated position at the same time. User 4 was not able to train the type of bot that they wanted to due to this restriction and the limited actions that could be performed in combat. A solution to this issue is to add guide actions which are able to add to the combat experience. These actions could include a flee to health item action, an action which allowed the bot to kite by staying in ranged attack distance, and a separate ranged action which moved into melee range.

The results showed that user 5 only using manual rewards seemed to perform better than using the automatic reward function in regards to training the bot that they wanted and seeing immediate feedback during the training session. Forcing manual rewards only should improve some of the issues with training not seeming to be working for some types of bots, especially those that differ from the path the automatic reward system steered them towards. This issue is clearly seen in the policy landscapes of the trained bots. Users 1, 2 and 4 did the least amount of training (as seen in the training results listed in Table 2) and the policy landscapes were all very similar. User 3 performed extensive training using automatic rewards, and was able to produce a more varied landscape for their trained bot. User 5 had an extremely different policy landscape to the other users due to only manual rewards being used, and they were the most successful in creating a bot that they wanted in accordance to their feedback. These results imply that the automatic training reward feature is too forceful for allowing full customisation for user guided training.

6 Conclusion

This paper has clearly shown that interactive training is a viable option for designing bots in an FPS game. All the users, who have experience working on big budget, high quality FPS games, felt that the tool had potential to be used during the development of a FPS game. The secondary aim of this paper was to show that a diverse set of bot types could be trained by different users. The results showed that the bots were all different from each other, and particularly the bot which was trained with manual rewards only was unique from the others and performed well in the game objectives.

A number of improvements have been identified based on the feedback from the users which will make the tool more suitable for commercial FPS game needs. A number of the users made points about the inadequacy of the wander behaviour, and commented on how it is not what bots should be doing, rather they should be moving with intention around the level to known item positions and known pathways. To address this issue bot patrol paths will replace the wander behaviour.

The difference between what the bot knows and what the user knows caused some frustration with one of the users. To address this issue, items and enemies that are visible will be marked so that the user can immediately see what the bot can see. Also the bot's vision will be modified from a distance based system to a line of sight based system to be closer to what a human player could see. Similarly the ranges for the ranged and melee attack behaviours were not obvious. Some of the users failed the melee action many times during training and this could be improved by having clear ranges visible on screen. In addition to this visual feedback, the actions that are not available (i.e. would fail due to parameter constraints) will be disabled on the UI. Further improvements will also be made to allow the designers to hand initialise the policy before training commences.

References

1. Sanchez-Crespo Dalmau, D.: Core Techniques and Algorithms in Game Programming. New Riders, Indianapolis (2003)
2. Isla, D.: Handling Complexity in the Halo 2 AI. In: Proceedings of the Games Developers Conference. International Game Developers Association, San Francisco (2005)
3. Jones, J.: Benefits of Genetic Algorithms in Simulations for Game Designers. School of Informatics. University of Buffalo, Buffalo (2003)
4. Overholtzer, C.A., Levy, S.D.: Adding Smart Opponents to a First-Person Shooter Video Game through Evolutionary Design. In: Artificial Intelligence and Interactive Digital Entertainment. AAAI Press, USA (2005)
5. Petrakis, S., Tefas, A.: Neural Networks Training for Weapon Selection in First-Person Shooter Games. In: Diamantaras, K., Duch, W., Iliadis, L.S. (eds.) ICANN 2010, Part III. LNCS, vol. 6354, pp. 417–422. Springer, Heidelberg (2010)
6. van Hoorn, N., Togelius, J., Schmidhuber, J.: Hierarchical Controller Learning in a First-Person Shooter. In: Computational Intelligence and Games, pp. 294–301. IEEE Press, Milano (2009)
7. Spronck, P.: Adaptive Game AI. Dutch Research School of Information and Knowledge Systems. University of Maastricht, Maastricht (2005)

8. McPartland, M., Gallagher, M.: Reinforcement Learning in First Person Shooter Games. In: Computational Intelligence and AI in Games, pp. 43–56. IEEE Press, Perth (2011)
9. McPartland, M., Gallagher, M.: Interactive Training For First Person Shooter Bots. In: Computational Intelligence in Games. IEEE Press, Granada (2012)
10. First-Person Shooter Games Prove to be Most Popular at MTV Game Awards. Entertainment Close - Up (2011)
11. Geisler, B.: An Empirical Study of Machine Learning Algorithms Applied to Modelling Player Behavior in a First Person Shooter Video Game. University of Wisconsin, Madison (2002)
12. Mora, A.M., Montoya, R., Merelo, J.J., Sánchez, P.G., Castillo, P.Á., Laredo, J.L.J., Martínez, A.I., Espacia, A.: Evolving Bot AI in Unreal^{TM}. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcazar, A.I., Goh, C.-K., Merelo, J.J., Neri, F., Preuß, M., Togelius, J., Yannakakis, G.N. (eds.) EvoApplicatons 2010, Part I. LNCS, vol. 6024, pp. 171–180. Springer, Heidelberg (2010)
13. Cole, N., Louis, S.J., Miles, C.: Using a Genetic Algorithm to Tune First-Person Shooter Bots. In: Congress on Evolutionary Computation, pp. 139–145. IEEE Press, Portland (2004)
14. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
15. Busoniu, L., Babuska, R., De Schutter, B.: A Comprehensive Survey of Multiagent Reinforcement Learning. Systems, Man and Cybernetics Part C: Applications and Reviews, 156–172 (2008)
16. Suh, I.H., Lee, S., Young Kwon, W., Cho, Y.-J.: Learning of Action Patterns and Reactive Behaviour Plans via a Novel Two-Layered Ethology-Based Action Selection Mechanism. In: International Conference on Intelligent Robot and Systems, pp. 1799–1805. IEEE Press, Edmonton (2005)
17. Bradley, J., Hayes, G.: Group Utility Functions: Learning Equilibria Between Groups of Agents in Computer Games By Modifying the Reinforcement Signal. In: Congress on Evolutionary Computation, pp. 1914–1921. IEEE Press, Edinburgh (2005)
18. Nason, S., Laird, J.E.: Soar-RL: Integrating Reinforcement Learning with Soar. Cognitive Systems Research 6(1), 51–59 (2005)
19. Patel, P.G., Carver, N., Rahimi, S.: Tuning Computer Gaming Agents using Q-Learning. In: Computer Science and Information Systems, pp. 581–588. IEEE Press, Szczecin (2011)
20. Blumberg, B., Downie, M., Ivanov, Y.A., Berlin, M., Johnson, M.P., Tomlinson, B.: Integrated Learning for Interactive Synthetic Characters. ACM Transactions on Graphics 21(3), 417–426 (2002)
21. Thomaz, A.L., Breazeal, C.: Reinforcement Learning with Human Teachers: Evidence of Feedback and Guidance with Implications for Learning Performance. In: Proceedings of the 21st National Conference on Artificial Intelligence, pp. 1000–1005. AAAI, USA (2006)