2k Bot-Prize

AI Project

Navneeth Nagaraj
14329570
Interactive Entertainment Technology
School of Computer Science and Statistics
University of Dublin, Trinity College
nn10@tcd.ie

ABSTRACT

In this paper we try to explain the way in which a bot can be developed in unreal tournament game using unreal engine so that the bot plays like a human. The bot is developed as per the requirements of 2k bot-prize competition. This paper contains introduction about the game, following our prototype on how a bot developed in unreal engine can be used in unreal tournament.

Keywords

Inverse Reinforcement Learning(IRL);Decorators;Enums; Data-Assets;Behavioral Tree;CPLEX;Pogamut;

1. INTRODUCTION

Firstly we will give a brief introduction and background about the competition and the game. The Bot-prize Competition is an annual competition to program bots that appear human-like to other humans for the commercial video-game Unreal Tournament 2004. In the competition, computer-controlled bots and human players (judges) meet in multiple rounds of combat, and the judges try to guess which opponents are human. To win the prize, a bot has to be indistinguishable from a human player. Then we will give our background research on how bots can be developed, there advantages and dis-advantages and how Inverse Reinforcement Learning(IRL) [32] can be used to develop a human like bot.

"In the case of the BotPrize," said Schrum, a great deal of the challenge is in defining what 'human-like' is, and then setting constraints upon the neural networks so that they evolve toward that behavior. We wont be going deep into neural networks, but we will try to explain a different approach along with the existing neural networks for the bot.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

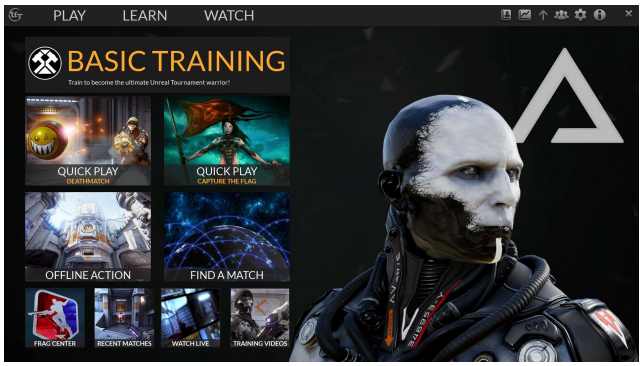

Figure 1: A Screen-Shot of Unreal Tournament Pre-Alpha

Secondly we will explain our logic on how the bot can act like human, copying the player's behavior and how IRL can be used to achieve this. We have developed two different bots based on different logic's which we will explain in detail along with this the final bot that we came up for unreal tournament. Here we are developing a Third Person Shooter(TPS) bot because there is not much of a difference between the Third Person Shooter and First Person Shooter(FPS) except the view. We are choosing the TPS bot because it is easy for us to observe the behavior of other bot with the player movements and easy to analyze it.

Later, we will explain how we developed the two bots using the state diagram and will explain each behavioral tree of our bot and different states all of which are developed using the unreal engine, we will also explain how these two characters can be integrated into one single bot to develop a bot that has human like behavior. Finally we will conclude the paper with the bot that we have developed, how can it be used in unreal tournament game and screen-shots of the same.

In sum, this paper includes

- Background research about the competition, game and AI.
- The different logic's for AI bot and what we have used.
- Implementation of bot using two different logical bots and integrating it to form an Unreal Tournament Bot.
- Problems faced and improvements.

2. BACKGROUND

2.1 About the Competition

"Computers are superbly fast and accurate at playing games, but can they be programmed to be more fun to play - to play like you and me? People like to play against opponents who are like themselves - opponents with personality, who can surprise, who sometimes make mistakes, yet don't blindly make the same mistakes over and over. The BotPrize competition challenges programmers/researchers/hobbyists to create a bot for UT2004 (a first-person shooter) that can fool opponents into thinking it is another human player. The competition has been sponsored by 2K games since 2008, with up to 7000 prize money. It was created and is organised by Associate Professor Philip Hingston, of Edith Cowan University, in Perth, Western Australia."

2K Games, Inc. is an American global developer, marketer, distributor and publisher of video games, like Borderlands, Civilization V, The Darkness, NBA 2K and more recently, WWE 2K.[13] 2K Games is a subsidiary of Take-Two Interactive. It was created on January 25, 2005 after Take-Two acquired developer Visual Concepts and its wholly owned subsidiary Kush Games from Sega for US24 million.

The game used for the competition will be based on a modified version of the DeathMatch game type for the First-Person Shooter, Unreal Tournament 2004. This modified version provides a socket-based interface (called Gamebots) that allows control of bots from an external program. In addition, several extra modifications will be made especially for the competition:
1)Chatting will be disabled. 2)Some aspects of the game play will be modified to faciltate the competition.

Judging will be done using the same in-game judging system that was used in 2011. The system is based on a modification to the Link Gun. The primary mode is intended for tagging BOTs, while the secondary mode is for tagging HUMAN-controlled opponents. During the game, when a player believes he/she has identified an opponent as a BOT (respectively a HUMAN), he/she shoots the opponent using the primary (respectively secondary) mode of the Link Gun. The shot has no effect on the opponent, but the player will see a tag "BOT" (or "HUMAN") attached to the opponent to remind them of which opponents have been judged. If the player changes his/her mind, the opponent can be shot using the other mode to reverse the judgement, as often as he/she desires.

The final tag for each opponent at the end of the round constitutes that player's judgement on that opponent. At the conclusion of all the judging rounds, each player's or bot's humanness rating will be the percentage of times it has been judged human over all the times it has been judged.

Judges will be encouraged to play the game as normal, in addition to judging. As incentives for judges to do this, there will be awards for best judge, and for the judge with the highest combined score over the judging rounds, and for the judge with the highest humanness rating.

To win the major prize of $7000 cash, a bot must achieve a humanness rating equal to or above the average humanness ratings of the competition judges. If more than one bot achieves this rating, then the prize will be shared equally. If no bot achieves this rating, then the minor prize of $2000 cash will be shared by the bots with the highest humanness rating.

2.2 About the Game

Figure 2: A Screen-Shot of Unreal Tournament-2004

Unreal Tournament 2004 is a first-person shooter video game developed by Epic Games and Digital Extremes. It is part of the Unreal series, specifically the sub-series started by the original Unreal Tournament; the sequel to 2002's Unreal Tournament 2003.

Unreal Tournament 2004 was built with Unreal Engine 2.5 and the content of its predecessor, Unreal Tournament 2003. The game was developed by multiple studios, with Epic Games leading the project.

Unreal Tournament 2004 is a first-person shooter representing a fast-paced extreme sport of the future. The game, designed primarily for multi-player game-play, offers multiple ways of movement including double-jumping, dodge-jumping, wall-dodging and shield-jumping.[3] UT2004 also features an extensive array of weapons, all of which come with a secondary fire. Some of them were designed specifically for use in vehicle-based game-types, and typically appear only in those game-types, such as the Anti-Vehicle Rocket Launcher (AVRiL) and the Grenade Launcher. More than 100 maps are included in the game for all new and existing game-types.

3. RELATED WORK

The 2k Bot-prize competition [4] has stimulated much of the related work in this area. Known as the "Bot Turing Test", the Bot-prize competition pits autonomous bots vs. a panel of expert human judges drawn from the gaming industry. The agents fight a modified death match game in Unreal Tournament against a human player and a judge; to win the

major prize, the bot must convince four out of five judges of its humanness.

In the 2010 competition no bot was able to win the major prize and as in previous competitions the most human bot did not manage to outscore the least "human" human player. There are, however, promising strategies that have emerged for topics such as item acquisition, steering, and firing techniques. Ideas such as circular strafing, imprecise aiming, and more complex item seeking strategies have been demonstrated by various bots (e.g, [4]).

The top-scoring bot in 2010 was Conscious-Robots from Carlos III University in Madrid. In many of the submissions, human-like characteristics were implemented for navigation. Relational databases have been used to provide the bot with long term memory of hot-spots where actions commonly occurred in the past, instead of going to the last place an enemy was seen [4]. In contrast, the bot uses IRL and expert human demonstrations to learn key locations for resource acquisition.

The bots, like humans, have different objectives depending on their state. The ICE 2008/2008 bot had two behavior states: item collection and combat [4]. The 2008 FALCON bot implemented four states allowing for more variability in action: running around, collecting items, escape, and engage [9] [3] notes that two of the most historically common failures among competition bots have been overly accurate shooting and lack of aggression compared to human players; our approach was specifically designed to address these shortcomings. Note that due to the rules of the competition, it would not be possible to use inverse reinforcement learning to learn an exploration policy directly on the competition map, but it remains a feasible approach for attack policies which are not map-specific.

3.1 Inverse Reinforcement Learning

To learn attack and exploration policies from expert player demonstrations, we should first use inverse reinforcement learning to learn a reward model from a set of player demonstrations. First the expert's demonstration is converted into a set of linear programming constraints over states, actions, and re- wards. The goal is to find a reward model that effectively explains the player's action choices. This is done by as- summing that the observed policy maximizes the selected re- ward model, and searching for reward values that minimize the difference between an optimal policy (under that reward model) and the observed set of demonstrations. To solve the resulting set of equations, we should use the CPLEX solver offline.

We need to use the IRL formulation defined by [6] and maximize:

$$\sum_{i=1}^{N} \min_{a \in A} \{ \big(P_{a1}(i) - P_a(i)\big) \big(I - \lambda P_{a1}\big)^{-1} r - \gamma ||r||_1 \}$$

shows that, $\{ \big(P_{a1}(i) - P_a(i)\big) \big(I - \lambda P_{a1}\big)^{-1} r \} \geq 0 \forall a \in A \backslash a_1$

$$|r_i| \leq r_{max} i = 1, ..., N$$

where a_1 is the action index of the policy for current state,P is the state transition matrix, λ is a discount factor, and γ is the penalty coefficient. r is the reward vector and r_{max} is set to 1. The max-min formulation can be converted to a problem of only maximization by adding new variables, y_i, to the problem as follows:

$$y_i \leq \{(P_{a1} - P_a)(I - \lambda P_{a1})^{-1} r\}$$

Maximizing y_i is equivalent to minimizing the right-hand side part of the objective function. By adding the sum over ρ_i to convert the first-norm of rewards into linear form, the overall linear programming formulation becomes:

$$maximize \sum_i^N (y_i - \lambda \rho_i)$$

shows that,

$$y_i \leq \{ \big(P_{a1}(i) - P_a(i)\big) \big(I - \lambda P_{a1}\big)^{-1} r \} \forall a \in A \backslash a_1$$

$$r_i \leq \rho_i$$

$$-r_i \leq \rho_i$$

$$r_i \leq r_{max}$$

If we define N as the number of states (128 for exploration and 12 for attack mode) and A as the number of possible actions (4 for both exploration and attack) we have 3N variables and (A-1)N +3N = (A+2)N constraints. Hence, we have 384 variables, 768 constraints for exploration mode, and 36 variables, 72 constraints for attack mode; due to our simple state space representation, CPLEX can calculate values for all the reward variables within a few seconds.

By using IRL, we will be able to avoid having to hand-code rewards for particular states based on domain knowledge and instead can directly leverage the experience of expert players. After IRL returns the reward model, we should run value iteration offline to learn a policy. During game-play, the bot simply performs a policy lookup based on the current game state which can be executed very rapidly.

4. OUR LOGIC

The first idea was to make the bot record other players at runtime instead of having a database of movements. This way, if the bot sees a non-violent player (shooting at the bot but around it, or shooting with a non-dangerous weapon) it would trigger a special behavior, mirroring. This makes the bot mimic another player in real-time, and therefore "borrowing" the humanness level. We thought that if our bot would meet a human, then it would seem human itself. We know this idea is not too new, actually it was inspired from some "how to be a salesman" articles we skimmed, which said that if not too obvious, mimicking can make a peer more comfortable with a conversation. The bot records keyframes of the target's actions, and plays them back with a small delay, and without full fidelity, so that it appears somewhat independent (mimic, not copy).

Due to the lack of long-term memory and the real-time nature of the mirroring module, we were obliged to use classic graph navigation, which we customized in order to hide

traces of bot-like movement such as the brief stops on navigational points, aiming behavior and elevator rides. The bot's movement and aim are completely separate, so that it can concentrate its aim to what requires attention while moving freely.

The bot also has the ability to remember its target, follow it when out of sight and dodging behavior based on the firing direction of its opponent. Also inspired from how human players generally play, the bot will forget its target if another opponent is more aggressive.

The complex game-play and 3-D environments of "Unreal Tournament 2004" require that bots mimic humans in a number of ways, including moving around in 3-D space, engaging in chaotic combat against multiple opponents and reasoning about the best strategy at any given point in the game. Even displays of distinctively human irrational behavior can, in some cases, be emulated.

"People tend to tenaciously pursue specific opponents without regard for optimality," said Schrum. "When humans have a grudge, they'll chase after an enemy even when it's not in their interests. We can mimic that behavior."

In order to most convincingly mimic as much of the range of human behavior as possible, the team takes a two-pronged approach. Some behavior is modeled directly on previously observed human behavior, while the central battle behaviors are developed through a process called neuroevolution, which runs artificially intelligent neural networks through a survival-of-the-fittest gauntlet that is modeled on the biological process of evolution.

"If we just set the goal as eliminating one's enemies, a bot will evolve toward having perfect aim, which is not very human-like. So we impose constraints on the bot's aim, such that rapid movements and long distances decrease accuracy. By evolving for good performance under such behavioral constraints, the bot's skill is optimized within human limitations, resulting in behavior that is good but still human-like."

"Miikkulainen said that methods developed for the BotPrize competition should eventually be useful not just in developing games that are more entertaining, but also in creating virtual training environments that are more realistic, and even in building robots that interact with humans in more pleasant and effective ways"

There was very successful game in which bots like these were created where in the bots acted like humans and it was a very successful branch as shooting game and also it was called an ego shooter games, one of them were Quake, which was a survival game. We have taken a part of this logic where in they use Inverse Reinforcement Learning(IRL). The main idea of this technique is that Reinforcement technique provides a powerful solution for sequential decisions making problems under uncertainty, whereas in IRL it directly tackles the problem by learning the reward function through the demonstrations, here demonstrations is the strategy of other players. We will show how this IRL helped in developing human like bot.

Further we thought to provide the bot with a strong strategy where in the strategy is taken our as a union of professional player movements, we achieved this by setting the way points as we set the route for the bot. We also added a feature wherein when a bot dies at a places more than two times the bot changes its strategy by marking the place as danger zone, we implemented this by setting a flag to zero when it dies more than two times and the strategy changes based on this. We wanted the bot not to be overloaded with the decision making, so we made a behavioral tree in a well organized way so that the bot acts quickly based on the situations.

In order to implement all this we thought of developing two bots wherein one bot just follows the way points and the other bot searches the whole map until they encounter enemies and then shoot, on careful observations we discovered that the bot following the way points had more efficiency than the bot searching randomly so we have taken the way point approach and along with this we have added controller which we will see later which helps in human like behavior.

In order to test all this we have created a zombie like bots where in the bot stays at a place and reacts if it gets hit or if the enemy is within the visual range, we also added a feature in which, whenever a bot spots an enemy irrespective of the movements the bot fires at the enemy, which makes the bot more accurate at aiming.

5. IMPLEMENTATION

We will explain in order how we implemented the bot in unreal engine.

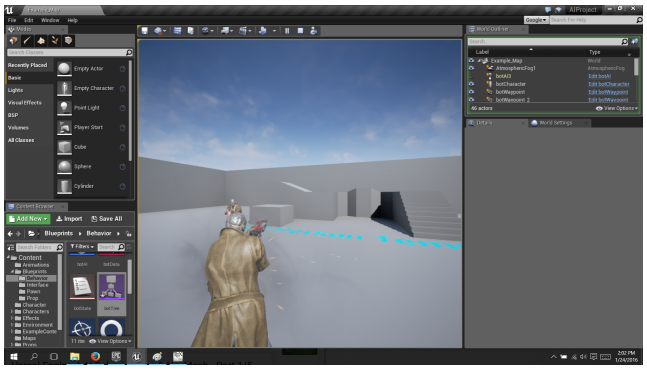

Figure 3: A Screen-Shot of Bot developed by using Unreal-Engine

5.1 AI and Nav Mesh

The first step is Navigation mesh which is a behavioral tree which is the back bone to any AI. We created way points, data assets, enums which defines the state of a bot, and then behavioral tree which is what controls the bot, task for the trees, decorators and services.

The map mesh should cover the entire area where you want to generate the bot. In order to create your AI bot we need use the pawn the player uses and then modify it. We are using "owen" (Bot Model) from the content example pack and have added an auto-gun. As we have mentioned before we will be creating a third person shooter since its all the same, only the view changes.

We then created two events enemy found and enemy lost. When the enemy found event is activated the bot shoots the weapon, else it will keep searching for another player. Changing animation is easy, get-mesh play animation and add the necessary animation. We have added all the necessary animation for the AI bot like movement, changes, reload. Now we need to create AI controller. We created a botAI controller and we want to create an Event-begin and we have added runBehaviourTree which is the bot runs at the beginning.Then we created Enumerator, this is user defining options or states which is in struct. Our bot contains Patrol, Search and Attack. Then we create data asset. This will allow bot to store information that the functions can reference to. We created targetPoint which will tell the bot the next location to goto, state of AI, Enemy actor which tells the bot who the enemy is and route which tell the next way point to patrol. We will create behavioral tree. AI way points which is modified target points.

5.2 AI Navigation

In second step we created tasks for navigation of the bot and finally navigating it to the enemy. Here we create different way points which is route through which the bot patrols until he encounters an enemy else the cycle repeats.

Even though we created way points we added extra controls so that it acts more like a human.

We have three different functions lets call it nodes as shown in Figure 4. Selector which call everything below it. There is another node Sequence calls functions in order i.e.this node will only call next function if only its previous function is successful and another one is Simple Parallel which calls one function while you call another. We will see all these in more detail later on.

In Figure 4 we can see that there is Patrol, Attack and Destroy and so on which can be see in blue, all these are called Decorators. These decorators are in the sense, the conditions that you add on top of your task.

5.3 AI Attack

In step three we created instructions for bot on how to attack the enemy.

The logic here is patrol the bot until he encounters the enemy, if found shoot. So we created bot to patrol from the beginning when the target is encountered there will be change of state and the bot shoots. The bot continues attacking until the bot loses the sight of the enemy. When the enemy is lost the bot again patrols and continue it as a cycle.

Here we use sequence route as described before. First we

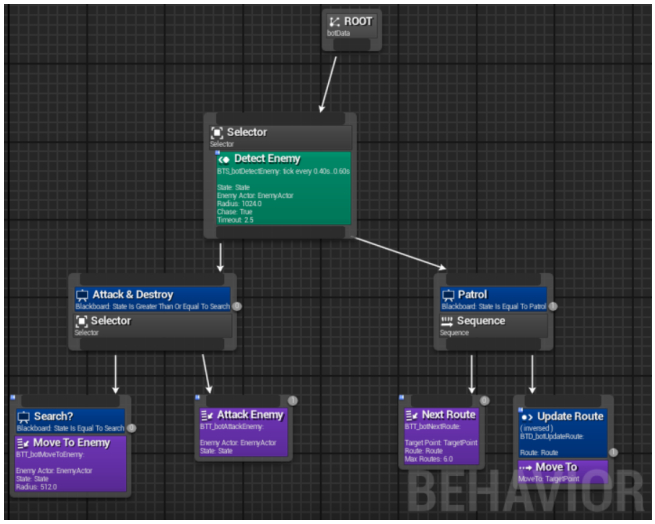

Figure 4: Behavioral Tree of our Bot

create an event receive execution where in it notifies enemy to change animation for attack. Based on these event the bot state changes between Patrol, Search and Attack. Here we have made the bot to attack continuously and follow the player so that the bot does not lose sight of the player and chances for the bot to act like human is more because once found he tries to kill the player.

For example in the place where we have added route, the bot should not repeat the same route because it has already been checked unless it checks the whole map, wherein we need an update which is taken care by services.

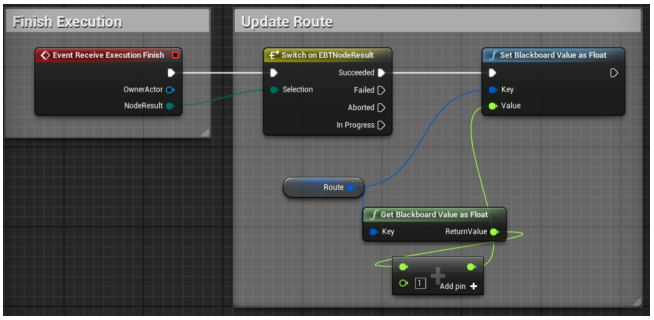

Figure 5: Bot Update Route

5.4 AI Brain

In step four we created the brain essential decision maker in a behavioral tree. In this step we detect the enemy change the state in order for the bot to attack the enemy. In Figure 4 we have added services, the role of services here is to tell the bot whether to Patrol or Search. The green indicator Detect Enemy in Figure 4 is the Service.

5.5 AI Modifications

Let us recall what we have done till now. We started with a logic where the bot follows the setup way points, if the en-

emy is found then the bot fires and if the enemy is not found the bot continues to patrol the map. Here we have initially set the way points where this way points will work only for a specific map because way points were set for a specific map and does not apply to all the maps available. In order to over come this we did develop another bot by considering the movement of zombie where the zombie moves only when the enemy is in sight. Here the combination is that we have set way points for the bot together we have added this feature where in when the bot does not know the map, it will follow the enemy when the enemy is in sight. We have developed this considering that no player in unreal tournament stays at one point more than five seconds, which was obtained on the research we did about the professional players.

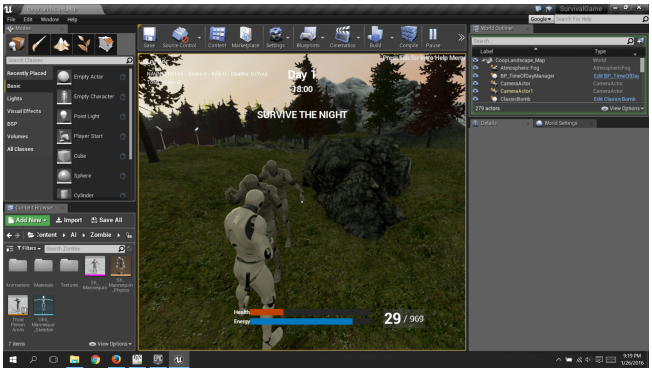

Figure 6: A Screen-Shot of 2nd bot that has zombie nature.

By considering the fact and the the logic we seen we developed bots like zombie where-in when the player enters the sight radius of the bot the bot comes near the player.

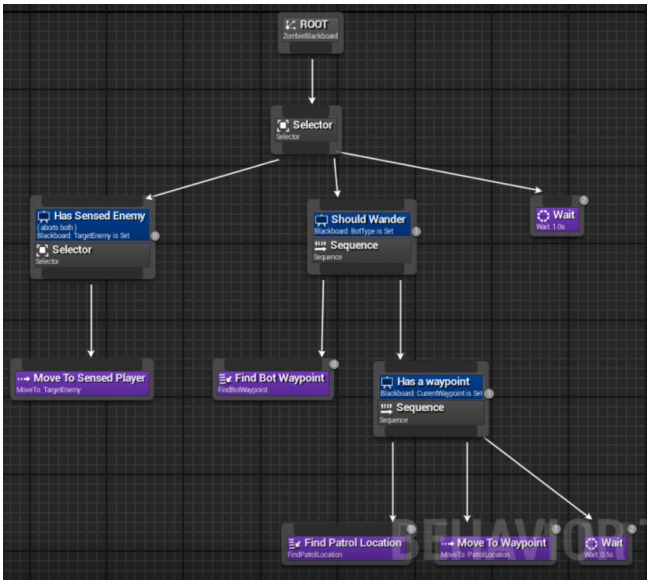

Figure 7: Behavioral Tree of Zombie Bot

Here we can observe that there are two selectors one is sensing the enemy and the other one is should wander. When the bot senses the enemy the bot moves close to the player and attacks else it will move to way-point, if the way-point is not found then it will find a patrol location which we have given in terms of way-points, where-in it moves towards it and then it waits for another event to occur.

6. EVALUATION

The first bot that we developed will start to run as soon as the game begins and will follow the way points that we had set, as soon as the bot encounters an enemy it follows the enemy and shoots at the enemy until the enemy is lost. This bot function works efficiently without any errors, when the bot loses the sight of the enemy it again follows the way point that we had set until it encounters the enemy. We had this tested and what we have observed is that when all the players are initialized, the bot follows the way points and if the enemy is in sight, it comes to close range and then shoots the enemy as shown in Figure 3.

On the other hand, we developed a bot that attacks when the enemy is in certain radius which we called it a zombie bot. This bot also attacks when the enemy is in the range of bot view. Here the bot follows the enemy more efficiently than the other bot, which is good. It was seen that the bot does not attack when the other players are already in a fight, which is to take advantage of the situation. This makes the bot to act more like human.

Further, we have combined all the advantages of both the bots and made into one single bot which we used in unreal tournament. Here as the game begins the bot gets a new weapon and switches to the most powerful weapon and then follows the way points and when more than one bot is encountered it attacks the enemy that is in close range and we have many successful wins of our bot.

In sum, the integrated bot acts more like human and has a good characteristics of human behavior.

7. PROBLEMS ENCOUNTERED

It was seen that when many enemies attack our bot, our bot was confused whom to shoot, so we made two choices wherein the bot either runs or shoots the bot nearby. It was difficult to understand the neural network so we have not made any changes to it. When the enemy is behind an object and the bot cannot see the enemy but the enemy can attack the bot, we for now solved this by making our bot move and not stop at a single position. And also many of the previous bot developers had used pogamut which is the old version of the unreal tournament and we have to use these code for the later integration and the additional packages dint not have clear explanations and we were not able to test it with the old tools and also now we have unreal tournament-2014.

It was bit tricky to integrate both the bots and form an unreal bot, since the packages were different. There maybe some errors which we have not encountered yet.

8. CONTRIBUTIONS

Both of us have contributed equal parts. From the beginning we analyzed the working of game, then we listed out the possible solutions to encounter the problem, then we made discussions on providing the best solutions and available logic. We then chose the platform that we want to develop the bot on, and then we worked on different machines to develop a bot that we had planned early as per the requirements. After the bot being developed we tested it by playing with the bots that we developed, later we made modifications for the bot and then we integrated it with to make the main bot and then imported it to the unreal tournament game.

Figure 8: Screen-Shot of player taking fire from bot.

9. CONCLUSION

It was asked to develop a bot which acts like human i.e the bot plays like a human player, in order to achieve this we developed two bots one that follows the player with characteristics that was pre-defined that is it fires when it encounters an enemy and the other bot was developed where the bot's state(action) changes when the enemy is in sight. We then integrated these two techniques to develop a bot for unreal tournament that acts like human.

Rather than following usual approach we took a different approach and different technique using which we have developed the bot as described in the paper. We have developed a powerful bot that satisfies most of the requirements needed for the competition. We also have shown how a bot can be developed using Inverse Reinforcement Technique, and different methods which can be used to develop a bot.

We have proposed and implemented a technique where-in the bot acts more like human to resolve mismatched valuations by having our bot directly learn reward functions from expert human player demonstrations.

10. LIMITATIONS AND FUTURE WORK

Most of the movements of our bot are manually programmed, we will try to make the bot to act on its own based on different situations it will encounter. We have not tested our bot for different maps, we will test our bot for different maps available. We were not able to incorporate all the AI techniques in the bot, we will add all possible techniques to optimize our bot when we make changes in neural networks.

Because of limited time we were not able to develop the complete bot that we wanted to, which we will be doing. We have not made any modifications in the neural networks, we will analyze the neural networks and will make necessary modifications to make the bot act more like human. Also the unreal tournament game competition was with the technologies available during the year 2004, we will further improve the bot with the latest technologies rather than developing the bot considering only the requirements.

11. ACKNOWLEDGMENTS

We would like to thank Liliana Paola Mamani Sanchez and University of Dublin, Trinity College for providing us an opportunity for developing a bot as a coursework of Artificial Intelligence. We would also like to thank our friends for their valuable feedback on the bot we developed.

12. REFERENCES

[1] Cassell, J.; Sullivan, J.; Provost, S.; and Churchill, E., eds.2000.Embodied Conversational Agents. MIT Press.

[2] Gemrot, J.; Kadlec, R.; Bida, M.; Burkert, O.; Pibil, R.; Havlcek, J.; Zemck, L.; Lovic, J.; Vansa, R.; Tolba, M.;Plch, T.; and Brom, C. 2009. Pogamut 3 can assist developers in building AI (not only) for their videogame agents. In Dignum, F.; Bradshaw, J.; Silverman, B.; and van Doesburg, W., eds.,Agents for Games and Simulations, Lecture Notes in Computer Science. Springer Berlin / Heidelberg.

[3] Hingston, P. 2010b. A new design for a turing test for bots.In IEEE Computational Intelligence and Games (CIG).

[4] Hirono, D., and Thawonmas, R. 2009. Implementation of a Human-Like Bot in a First Person Shooter: Second Place Bot at BotPrize 2008. In Proceedings of Asia Simulation Conference 2009 (JSST 2009).

[5] Laviers, K.; Sukthankar, G.; Molineaux, M.; and Aha, D. 2009. Improving offensive performance through opponent modeling. In Proceedings of Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE).

[6] Ng, A. Y., and Russell, S. 2000. Algorithms for inverse reinforcement learning. In Proceedings of 17th International Conference on Machine Learning, 663-670.

[7] Schaeffer, J.; Bulitko, V.; and Buro, M. 2008. Bots get smart. IEEE Spectrum.

[8] Sutton, R. S., and Barto, A. G. 1998.Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning). The MIT Press.

[9] Wang, D.; Subagdja, B.; Tan, A.-H.; and Ng, G.-W. 2009. Creating Human-like Autonomous Players in Real- time First Person Shooter Computer Games. In Twenty-First Annual Conference on Innovative Applications of Artificial Intelligence (IAAI'09).

[10] Wray, R., and Laird, J. 2003. Variability in human behavior modeling for military simulations. In

Proceedings of Behavior Representation in Modeling and Simulation Conference (BRIMS).

[11] Gemrot, J., Kadlec, R., Bida, M., Burkert, O., Pibil, R., Havlicek, J., Zemcak, L., Simlovic, J., Vansa, R., Stolba, M., Plch, T., Brom C.: Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents. In: Agents for Games and Simulations, LNCS 5920,2009,pp. 1-15.

[12] Bida, M., Cerny, M., Gemrot, J., Brom, C.: Evolution of GameBots project. In: Herrlich, M., Malaka, R., Masuch, M. (eds.) ICEC 2012. LNCS, vol. 7522, pp. 397–400. Springer, Heidelberg, 2012.

[13] https://wwe.2k.com/

[14] Learning Policies for First Person Shooter Games Using Inverse Reinforcement Learning, Bulent Tastan,and Gita Sukthankar,Department of EECS, University of Central Florida Orlando, FL 32816, U.S.A

[15] P. Abbeel and A.Y. Ng. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the 21st international conference on Machine learning (ICML 2004), 2004.

[16] Edwin V. Bonilla, Shengbo Guo, and Scott Sanner. Gaussian process preference elicitation. In NIPS 2010, 2010.

[17] C. Boutilier. A POMDP formulation of preference elicitation problems. In AAAI 2002, pages 239-246, 2002.

[18] Constantin A. Rothkopf. Modular models of task based visually guided behav- ior. PhD thesis, Department of Brain and Cognitive Sciences, Department of Computer Science, University of Rochester, 2008.

[19] Umar Syed and Robert E. Schapire. A game-theoretic approach to apprenticeship learning. In Advances in Neural Information Processing Systems, volume 10, 2008.

[20] Umnar Syed and Robert E. Schapire. A reduction from apprenticeship learning to classifcation. In NIPS 2010, 2010.

[21] Brian D. Ziebart, J. Andrew Bagnell, and Anind K. Dey. Modelling interaction via the principle of maximum causal entropy. In Proceedings of the 27th International Conference on Machine Learning (ICML 2010), Haifa, Israel, 2010.

[22] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In in 20th Int. Joint Conf. Arti cial Intelligence, volume 51, pages 2856-2591, 2007.

[23] D Ramachandran, 2010. Personal communication.

[24] Marting L. Puterman. Markov Decision Processes : Discrete Stochastic Dynamic Programming. John Wiley and Sons, New Jersey, US, 2005.

[25] P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In ICML 2006, pages 697-704. ACM Press New York, NY, USA, 2006.

[26] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In in Proc. 17th International Conf. on Machine Learning, pages 663-670. Morgan Kaufmann, 2000.

[27] Shengbo Guo and Scott Sanner. Real-time multiattribute bayesian preference elicitation with pairwise comparison queries. In AISTATS 2010, 2010.

[28] Peter D. Grunwald and A. Philip Dawid. Game theory, maximum entropy, minimum discrepancy, and robust bayesian decision theory. Annals of Statistics, 32(4):1367-1433, 2004.

[29] Thomas Furmston and David Barber. Variational methods for reinforcement learning. In AISTATS, pages 241-248, 2010.

[30] Milton Friedman and Leonard J. Savage. The expected-utility hypothesis and the measurability of utility. The Journal of Political Economy, 60(6): 463, 1952.

[31] Michael O'Gordon Duff. Optimal Learning Computational Procedures for Bayes-adaptive Markov Decision Processes. PhD thesis, University of Massachusetts at Amherst, 2002.

[32] Preference elicitation and inverse reinforcement learning Constantin A. Rothkopf and Christos Dimitrakakis ,Frankfurt Institute for Advanced Studies, Frankfurt, Germany.

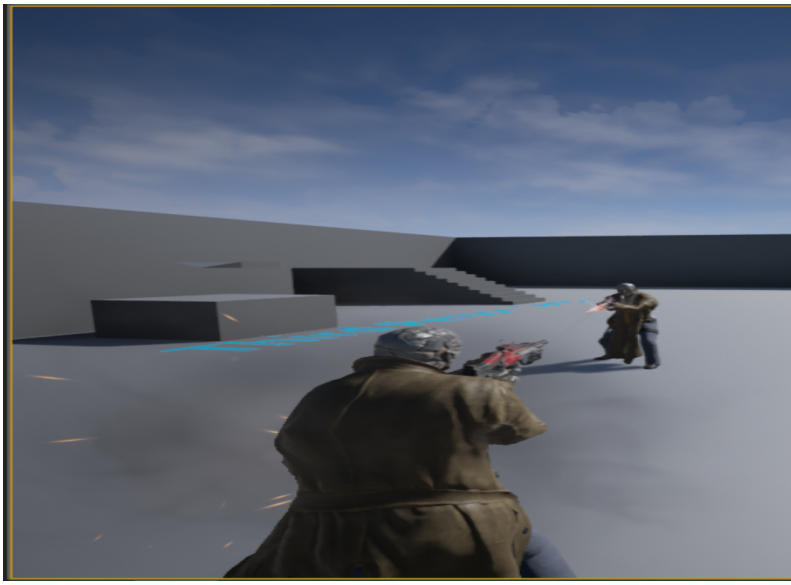

Figure 9: Screen-Shot of player taking fire from bot. The bot is the one in front of the player.

Figure 10: Screen-Shot of bot following the player and attacking.

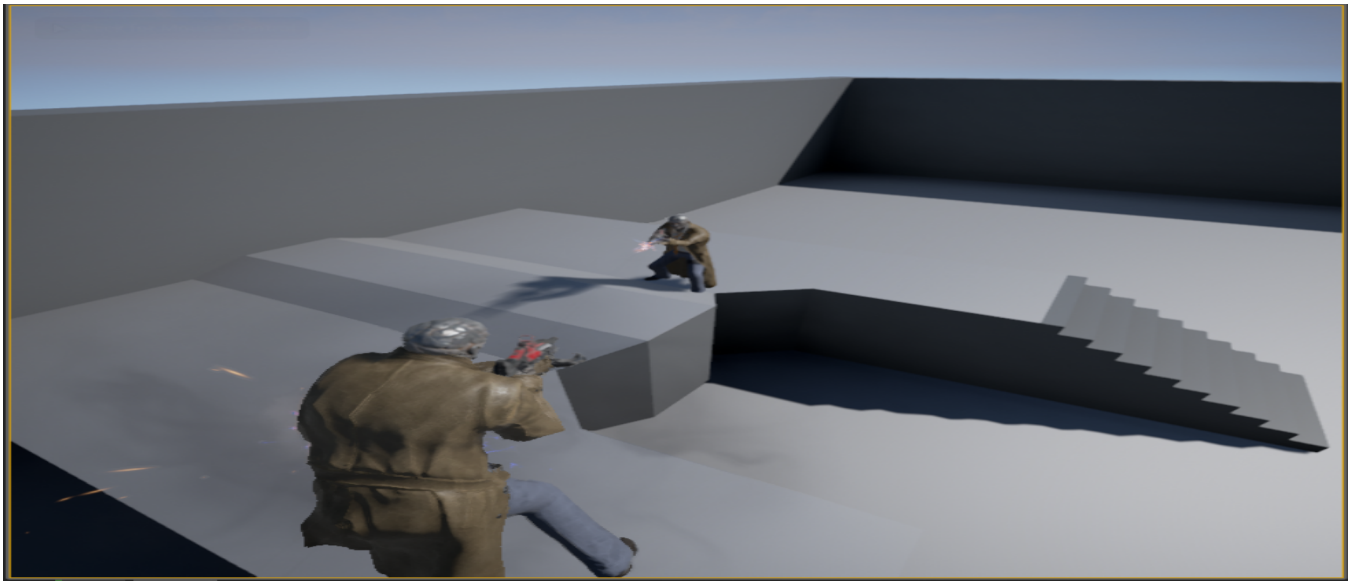

Figure 11: Screen-Shot of bot firing when player is moving.