DOI: https://doi.org/10.61762/pjstvol5iss2art20979

PLATFORM - A Journal of Science & Technology

Received: 22 November 2022, Accepted: 22 December 2022, Published: 31 December 2022, Publisher: UTP Press, Creative Commons: CC BY 4.0

ADAPTABLE ADVERSARY BEHAVIOUR FOR GAME OF TAG BASED ON PLAYER BEHAVIOUR USING MACHINE LEARNING

Muhammad Iqbal Asim

Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Malaysia

Email: Muhammad_18001179@utp.edu.my

ABSTRACT

Adversaries in video games would seldom be created to rely on predefined behaviour trees to respond to various player behaviours and fulfil simple goals in order to deliver an experience designed by the video game developer. While this is adequate for most games, immersion and the intended experience of the video game would be compromised when the adversary is faced with unexpected player behaviour, which would usually end up confusing the adversary's artificial intelligence (AI). This occurrence is known as a "sequence break", which is an event that occurs when the behaviour tree is poorly designed and contains known exploits that could either significantly reduce the difficulty of dealing with the adversaries or allow the player to bypass them entirely. Allowing adversaries to adapt to player behaviour as the game progresses using adversarial reinforcement learning would provide a dynamically evolving experience in which the user would be continuously challenged depending on their cleverness, forcing them to play the game as intended without the use of exploits. Inadequate adversarial AI in video games can make a game more boring or frustrating depending on the set difficulty level determined by the game developer. In this study, past and current implementations of AI in video games are explored, including the unique innovations which can be found throughout the video game industry regarding video game AI adversaries.

Keywords: *artificial intelligence, adversarial reinforcement learning, video games, behaviour tree*

INTRODUCTION

In this research, Adversarial Reinforcement Learning is used to train an agent in the game of tag. The agent is then tested in a live setting against player-controlled agents to gauge its proficiency. This study is classified as machine learning research and is thus under the umbrella of artificial intelligence. In machine learning, data is acquired or generated and then used by algorithms to decide on an action to be taken. The consequences of the action taken are then used as a metric to measure the success of that decision and are factored into the next decision made.

This study pursues the use of Adversarial Reinforcement Learning, which pits two machine learning agents against each other in repeating rounds of the game of tag. Hence, it would teach the agent how to traverse the game environment based on the behaviour of the other player agent. The agent will learn to maximise its rewards through set reward structures which correspond to its state of being tagged or not and deciding on a set of basic movement actions the agent can make to achieve this goal. More complex actions, such as creating distance between the agent and its adversary and pursuing an untagged agent, will be left to the agent to teach itself as these behaviours should not be solidly defined to allow the agent flexibility to adapt to its adversary depending on their unique behaviour.

PROBLEM STATEMENT & OBJECTIVES

Artificial intelligence adversaries in video games rely on designed predetermined behaviour trees, allowing them to respond to various player behaviours

and achieve simple goals to deliver an experience designed by the video game developer. While this is sufficient for most games, the immersion is immediately hampered or broken once the behaviour becomes stale through repetition, or much worse, when the behaviour tree is not well designed and has known exploits which could either significantly reduce the difficulty of dealing with such enemies or allow the player to bypass them entirely.

As an adversary in a video game exists solely to create an obstacle for the player to overcome, predictable adversaries immediately decline in difficulty as soon as the player has more than a few encounters with them. The player's progression in terms of personal ability is usually acknowledged in certain video games, usually Esports titles such as Counter-Strike: Global Offensive or Dota 2, where the objective is to best your human opponents. But for single-player titles where the intended dynamics of the game are designed around the player interacting with the video game environment, the most common case is that it would usually break the game for such a player as their journey would be lacking the "struggle" aspect of the video game which is highly valued especially in single player games. This is because the experience of struggling through a section within a level pushes the player to the beginning of the natural journey of progression the player would be able to share with their player character as they level up and gain abilities which would alleviate their struggles from earlier in the game. In turn, by going through the struggle phase of the game, the player is given a sense of value for the upgrades and abilities which would be granted later on in the game, hence artificially lengthening play time and value.

Allowing enemies in video games to adapt to player behaviour as the game progresses would allow for an unpredictable experience where the player would be consistently challenged based on their ingenuity and force them to play the game as intended without abusing exploits. On the other hand, this circumvents the issue of difficulty mismatch for players who choose a higher difficulty without being able to cope with the difficulty level. By allowing enemies to reactively adapt their playing strategies to player behaviour and performance, less skilled players would have a chance and properly enjoy the game as it is.

The objectives of this study are to explore the data sciences and utilise the knowledge gained to develop a competent AI video game and video game AI adversary that can beat the player in the least amount of time possible and pose as a competitive adversary that can adapt its playing strategy and behaviour based on the behaviour of its adversary, which is the in-game player. Besides, to validate that the developed AI adversary can create a fun and engaging experience for the player by providing a challenge that is easy enough to be engaging and hard enough but still possible to overcome.

SCOPE OF STUDY

This study will be split into two development phases: the game environment and the machine learning agent. Phase 1 of the study will utilise the agile development pipeline since it involves the development of the entire video game environment within which the machine learning model will be trained, tested, and deployed. This includes the development of the video game environment and the agents the machine learning model will take control of interacting with its environment. Both the machine learning model and the player will be given control of the same agent. The player agent will be developed with the ability to move in 4 directions (forward, backward, rightward, and leftward) and rotate their body 360 degrees around the y-axis. Each action is controlled through input devices which are read as a float value between -1 and +1. Due to the nature of this player agent control scheme, integration of either human player or machine learning model controls are as straightforward as pairing the source of input floats with the input structures from each player, whether it be mouse and keyboard or the action states of the machine learning model represented by a bank of float values. The training environment is a square arena enclosed by walls and populated with obstacles which would ideally obscure a straight line between each player agent spawned within the arena. The environment would only have 2 players spawned on one of the 4 corners of the arena and the centre. Once the game commences, both players can roam freely throughout the environment.

Phase 2 of the study entails the development of the machine learning model, which will take control of the adversary agent within the developed video game environment. This phase will involve the entire

machine learning development pipeline, including data extraction, development, training, testing, evaluation, and deployment. This study will be trained using the Adversarial Neural Network (ANN) model and utilising the Unity ML-Agents Toolkit to marry the machine learning model and training environment hosted in the Unity Game Engine. This is to ensure that the integration of the machine learning model and the training environment is done seamlessly and guarantees full control of the training environment variables, including the study's reward structure and training environment. The main workload of this study phase is balancing the game of tag rule set and tweaking the reward structure for the machine learning model training. The reward structure for the model should ideally push the agent to avoid being tagged as "it" and run as far as possible from the player who is "it".

Figure 1 Optimal game of Tag behaviour flow

LITERATURE REVIEW

The use of pathfinding algorithm, A* and Finite State Machines (FSM) is an industrywide standard according to Orkin [1]. Any video game with Non-Player Characters (NPC) that uses AI would use A* for pathfinding and some form of FSM for behaviour management. Innovating on this status quo, Orkin [1] mentions that an FSM dictates the video game the NPCs in F.E.A.R. with the A* algorithm integrated as some form of planning algorithm to create an action sequence plan. This is an unconventional use of common algorithms, as the A* algorithm is usually used exclusively in the video game industry as the baseline pathfinding algorithm. Implementing the algorithm into NPCs' behaviour planning in F.E.A.R yielded unexpected yet interesting results.

The NPCs of F.E.A.R. are built off three-state FSM. The fundamental states used are Goto, Animate, and UseSmartObject. UseSmartObject is a data-driven state derived from the Animate state. But instead of selecting an animation to play directly, this state picks an animation from the derived SmartObject from their game database [1]. This means that the FSM powering NPCs in F.E.A.R. are actually running on a dual-state FSM instead. According to Orkin [1], the development team for F.E.A.R.'s FSM have decided to boil down the FSM for their Ais due to the realisation that any actions taken by the NPCs could be simplified to basically going from point A to point B and playing an animation. Although, the difficult part is figuring out what to do and when to do it.

Another paper addressed using AI in video games in a competitive setting. According to Berner et al. [2], OpenAI Five became the first AI-powered system to win a match against the Dota 2 world champions when writing the study. Dota 2 is a Multiplayer Online Battle Arena (MOBA) Esports title where two teams of 5 battle it out in a closed arena with various obstacles and a central base target that their consecutive teams need to defend from attacks.

The AI system was developed for the specific purpose of beating the Dota 2 world champions is named OpenAI Five. Berner et al. [2] mention that Dota 2 was the specific choice for this study as it introduces significant challenges like long time horizons, partial or imperfect information, and continuously complex state-action spaces. Conquering these challenges is noteworthy as they would be the main feature set for more proficient AI systems moving forward. This study is also one of the earliest implementations of AI systems in competition with human counterparts for an esports game. Esports titles are known for their heavy emphasis on teamwork and coordination to execute an agreed-upon strategy

More experimental applications of this research are also addressed with a paper by Baker et al. [3], where the study pits two adversarial Long-Short-Term Memory (LSTM) AIs against each other in order to observe any integration of tool usage strategies towards their playing strategies. The LSTMs are split into two teams

Seekers and Hiders. Both teams are provided access to movable tiles coloured yellow on each level. Both teams can interact with the tiles, although once a tile is locked, it cannot be moved by any member other than the team the tile has been locked by.

According to Baker et al. [3], through deeper iterations of the training phase, the adversarial LSTMs began to develop more intricate playing strategies to best one another. At first, the agents started moving around at random. At a certain point, the agents began chasing each other. Then, the hiders started using the box tiles to block entrances to keep themselves hidden. After that, the seekers figured out that the ramp tile was useful for going over walls. Through this study, we can deduce the capabilities of the ML-Agents Toolkit provided by Unity as this study flawlessly integrates their agents into their game environment using the aforementioned package.

METHODOLOGY

The ambiguity of the general end goal of each player in a game of Tag presents an issue in defining terminal states for the machine learning agent to train under during reinforcement learning. Because of this, the specific rule set used within this study will be modified as such:

1. Each player is spawned on preset locations within the arena, and one random player is designated as "it".
2. Once the game commences, each player can roam wherever within the map's bounds.
3. Once the "it" player touches or tags another player, that player is now frozen for 2 seconds before being allowed to move again to allow the previous "it" player a head start.
4. The "it" player loses once they are tagged for more than 10 seconds, including time frozen.

The modifications occur in rules 3 and 4, where additional circumstances are handled to make the game fair and provide a tangible goal for both players to pursue aside from chasing each other within a closed arena. Rule 3 is added to prevent recent "it" players from instantaneously tagging the previous "it" player. A situation which is unfair as it disallows the other player to get away from the newly tagged "it" player. Rule 4 is added to create an end goal that seeks to make the overall duration of the game finite.

This study utilises two process methodologies employed for both phases of the study. Phase 1 would use the Agile development method, and phase 2 would use the CRISP-DM framework.

The game environment developed for this study will be the training environment for the following phase. Due to this, its completion will have to be performed in advance to ensure a smooth transition to phase 2, which is the machine learning model development. The game environment can be broken into several main components: Player Movement, Player Controller, Game Manager, and Game Arena.

The player movement component is what marries the player controls to the virtual 3D player agent within the game arena. This component handles the general movement of the player agent and its interactions with the game environment, such as gravity, physical boundaries, and player agent states, such as frozen, tagged, and untagged. The player agent's states are what determine its available actions during gameplay. Being frozen means, the agent will not be able to move or look around for the entire duration it is frozen. Being tagged would mean the agent would have to chase the other untagged player agent, and being untagged would mean the opposite, which means the player agent would have to run away from the tagged player agent. The player movement component also provides several actions available to the player agent during gameplay. Some actions available to player agents are moving across the X and Y axes and look rotation along the Y axis.

The player controller is the component which translates player or model input to usable values, which the player movement component can use to interpret into tangible actions. A human player would be interacting with the game through the use of keyboard and mouse input devices. The input values of different devices are registered in Unity through its Player Controller component, which can interpret directional buttons as float values representing vertical or horizontal axis movement or individual Boolean values for each button. With that in mind, the configuration of the machine learning agent would be to configure simply it to provide float or discrete values representative of the number of input actions required. Using this component, a human player and machine learning model would use the same input schematic.

The game manager is the main component that controls the handling of game states and enforces the game's rules within the game environment. The game manager is a single source of truth for player agents to retrieve characteristics such as reward modifiers, game rule modifiers, and arena size. The game manager is also responsible for resetting the state of the game environment once a winner has been decided.

The game arena is the virtual 3D game arena in which the entire game of Tag takes place. The agents will be playing within several variations of the same game arena, which allows the agents to learn sequentially with higher complexity with each run of the training phase. This is performed in this method due to the time constraints for the study. The variations of the game environment configurations available are as follows:

1. Random spawn locations
2. Random beginning "it" player
3. Game time limit
4. Tag time limit

The second phase of development will be centred around the configuration of the training environment to fit the requirements of the study and tuning the hyperparameters of the machine learning model training.

To train the model, this study uses an Artificial Neural Network (ANN) model with adversarial reinforcement training. The training in phase 2 will use a reinforcement training method where two versions of the model will be pitted against each other, and its ELO ranking will be decided by the end of the episode if the model wins.

The training of machine learning agents will go through all of the arena variations to ensure thorough coverage for its playing strategy. The inclusion of obstacles should allow for more variation in gameplay, as the obstacles can obstruct a direct line between two players or create a complex routing strategy to catch up or extend the gap separating both players.

The agent controller is the component responsible for determining the rewards for each player agent throughout an episode of training. The reward structure for an agent is configured around the importance of tagging the other player if tagged and staying untagged as long as possible. Due to this, the reward structure has been configured as follows:

1. Untagged agents would have their cumulative rewards set to zero, and tagged agents would receive a punishment.
2. An agent ending an episode as untagged would receive a big reward, while a tagged agent would end the episode with the initial punishment amount.

RESULTS

After extensive training runs, the agent appears to have learned the general layout of the game arena while also having several basic strategies in its arsenal. Some of these include navigating in an unorthodox pattern which may confuse its opponent on where it is headed. Both agents seldom use this strategy alongside following the quickest route towards each other. This creates a diverse variety of gameplay situations where one game could end in a quick exchange of close-quarter tagging or a continuous pursuit throughout the game arena, ending with a winner. This behaviour is described in Figure 2.

Figure 2 The trained agent navigating through the maze of the game arena to reach the target agent, which is hiding behind multiple walls

The agent has also learned to create the most spaces through a complex route between itself and the tagged agent, as shown in Figure 3. In addition, the agent has learnt that when its opponent is tagged and frozen, it would be the best time to create the most space by moving quickly through the maze of the game arena. This means it is utilising the grace period provided by

the frozen delay and allowing itself an escape strategy by going towards the outside of the arena, which has the most escape routes.

Figure 3 The agent following an exit routine to create space between itself and the tagged player

Other than that, the agent can quickly respond to close-quarter opponents who tag each other in quick succession. In this situation, the agent would move rapidly in a circular motion to halt the other player from escaping through the closest doors whilst spinning itself to tag the other player if they were to try to sneak past the rapid movement. A visual of this behaviour is shown in Figure 4.

Figure 4 The agent spinning around and moving rapidly within a choke point to exhaust exit options for the target agent

The walls used for the game arena have a very thin thickness, allowing only the agent's hands to come through since its capsule collider is only limited to its main body. Due to this, the agents have learnt to tag each other through the walls if the other agent is within reach on the other side, as shown in Figure 5. This was a quirk of the game arena that the agents abused heavily during gameplay since they were not punished for performing this exploit.

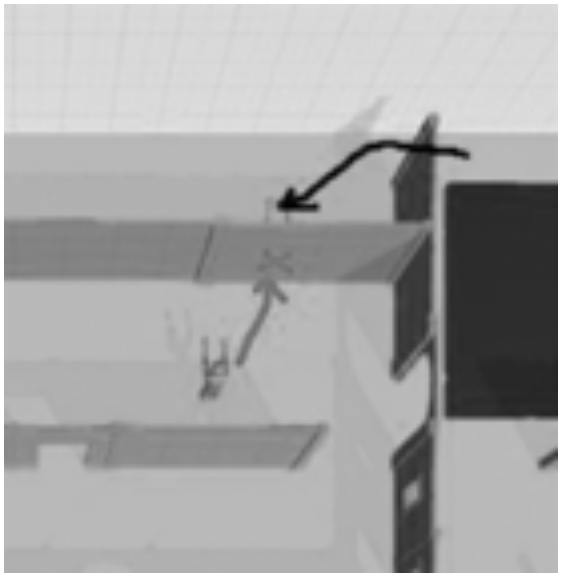

Figure 5 The opposing agent tagging through the walls, indicated by the 'X' when the agent tries to escape its path

Although, the agent still has some flaws in its pathfinding as it can still get stuck in dead ends, corners, or doorframes. On some occasions, the agents use this flaw to their advantage. Seeing that both agents have the same flaw, the agents would navigate close to wall corners and ends to ensure that the other agent has a chance of catching themselves stuck in a corner or on the wrong side of a wall. One such event has been documented in Figure 6, as the

Figure 6 The agent is trapped in the corner of the doorframe geometry and is stuck

target agent moves rapidly away and downwards from the doorway, which confuses the agent into moving into the doorway corner, which renders it stuck. This strategy is unique to this implementation of Tag as it is a quirk that only occurs with these specific agents.

CONCLUSION AND RECOMMENDATION

With the examination of previous implementations of behaviour trees in video games and more experimental implementations with machine learning agents within games, extensive insight was provided, which helped supercharge the development and training phases of the study. With the simple reward structures stemming from terminal actions within the game of Tag, the machine learning model can create various scenarios from the adversarial reinforcement learning process competition.

Following a plan that has been put in place preliminarily helps exponentially speed up the progress of the study as it allows for a smooth implementation throughout the development of the entire study. By employing time management skills and study management knowledge, the study could be completed within the given timeframe and meet its deadline.

Since even the model itself acknowledges this quirk of its own making and uses it to its advantage against the adversarial model, longer and more extensive training sessions should reorient this behaviour towards pathfinding which is more resilient as it should be more familiar with the topology of the game arena itself with more time training within it. Other than that, a more diverse variety of game arenas should also be used to train the model to provide it with a wide array of environments to train within and reinforce its pathfinding algorithm.

In conclusion, machine learning agents can serve as adequate video game adversaries given the training environment is properly configured and the model is trained extensively on a diverse array of states. The main objective of this study was to explore the viability of machine learning agents as video game opponents, given they have access to the same control structure as the player. The result of this study shows that the agents can play competently with the basic rules of the game of Tag and create various playing strategies based on the game's quirks, arena environment, and their own pathfinding.

ACKNOWLEDGEMENT

The authors would like to acknowledge the Department of Science and Information Technology, Universiti Teknologi PETRONAS, Malaysia, for supporting and providing research facilities to conduct this research.

REFERENCES

[1] J. Orkin, "Three States and a Plan: The AI of FEAR," In *Game developers conference*, SanJose, California: CMP Game Group, March. 2006, vol. 2006, p. 4.

[2] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison,... & S. Zhang, "Dota 2 with large scale deep reinforcement learning," arXiv preprint arXiv:1912.06680, 2009.

[3] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, & I. Mordatch, "Emergent Tool Use from Multi-Agent Interaction," OpenAI, Feb. 11 2020. [Online]. Available: https://openai.com/blog/emergent-tool-use/ (accessed Feb. 10, 2022)