Exploring Reinforcement Learning to Simulate Realistic Behavior in Autonomous Human Agents

Abstract—**Current popular methods for creating realistic autonomous human agents require complex rule sets and expertise, especially when considering Multi-Agent environments. This study examines the use of reinforcement learning for this task. By designing agents meant to take advantage of reinforcement learning theory, namely the idea of Markov Games, five soft-actor critic reinforcement learning models are trained on the task of simulating student agents with unique personalities in a classroom. Agent performance is evaluated quantitatively through the use of a behavior flagging system. The results demonstrate that reinforcement learning is a viable approach for creating autonomous human agents that demonstrate complex behavior.**

Index Terms—**Agents, Artificial Intelligence, Autonomous Agents, Autonomous Systems, Reinforcement Learning, Simulation, Soft-Actor Critic**

I. Introduction

Autonomous agents are becoming an increasingly important topic in technology, especially in the field of artificial intelligence [1]. This extends to development of realistic autonomous human agents (AHAs) in both simulations and video games. They enhance the accuracy of the former while bolstering verisimilitude in the latter. This is especially true in multi-agent environments where agents must realistically navigate a shared environment. But developing realistic AHAs, or AHAs which closely imitate the actions of the agents they are based on, is difficult. Five standard methods for doing so have previously been defined: Finite State Machines (FSM), Tree Based Methods (TBM), Rulset/Logic Based Methods, Goal-Oriented Action Planning (GOAP), and Artificial Neural Networks (NN) [2]. This paper explores an underused branch of the NN method, reinforcement learning (RL), and its application for simulating realistic AHA behavior in multi-agent environments.

Despite their potential use, realistic AHAs are rarely created with RL for simulations and video games. In a survey on AHA decision making methods in games, only 7 of the 106 studies employed NNs while only 1 of 20 games identified used NN for AHA development [2]. Developers instead often use traditional techniques. However, simulation modeling with traditional techniques is a time-consuming process that can be difficult for those not specially trained in simulation building [3]. Nontraditional approaches that involve RL often fall under the field of Multi-Agent Reinforcement Learning (MARL), or the field of simulating multiple agents using RL algorithms. Some modern MARL solutions like MADDPG, COMA, and Multi-Agent Deep Q-Network [4] [5] [6] have shown promising results. They see agents learning to navigate complete tasks through rewards and punishments rather than explicitly defined behaviors as seen in traditional techniques.

While these and similar examples exist, they are often applied to test environments such as particle environments. There are a few notable examples of RL being used for the development of AHAs. Some examples include agents in military simulators and simulations of on ramp merging [7] [8]. However, these papers explore AHA development with little focus on the ”realistic” aspect of realistic AHAs. This paper applies MARL techniques to a classroom simulation. The goal is to explore RL as a viable tool for the creation of truly realistic AHAs in simulations and games.

There is some evidence that reinforcement learning can be used to provide realistic personalities in simulations [9]. The present study's approach applies this to the problem of simulating a classroom. Students in a classroom may confidently ask questions or sit in the back to avoid getting called on. They could be hard workers, or get easily distracted. An RL solution allows required classroom goals (sit down for lectures, complete work, etc.) to be set while the agents naturally learn a variety of ways to approach the environment and complete expected goals. With the right rewards and proper training, this process creates agents with different personalities when approaching situations. The approach for this study was to train with a soft-actor critic (SAC) network. SAC has a critic examine the state/action pairs of agents from a replay buffer and learn an approximate function of the best state/action pairs for all agents to take. With this network, models can be created that give students unique 'personalities' while still accomplishing the required goals of the simulation. This approach was evaluated using a quantitative assessment flagging system.

II. Related Work

A. Reinforcement Learning Theory

RL involves allowing agents to navigate obstacles and complete a task in an environment by examining states, actions, and rewards [10]. An agent is a decision making entity that learns to navigate an environment. An example would be a student in a classroom. The environment is everything outside the agent, such as obstacles to avoid and objects to interact with. The agent can take actions in the environment, such as moving or interacting, to complete a task defined by the environment and rewards. The task for this study is simulating student activity in a class period.

The task is split into any number of timesteps. Agents learn to complete the task by taking an action and examining the resulting state/action pair at a given timestep. The agent examines their state by observing aspects of the environment, such as their own position within it or the position of obstacles. They also examine rewards, positive or negative, that were given because of an action during the timestep. These rewards are predefined by the programmer to encourage behavior. The agent tries to maximize the reward it can receive during a task. For example, if an agent receives a positive reward for sitting down at a desk, it should learn that taking the action to sit when close to a desk is ideal behavior. With the right reward schemes, more complex tasks can be accomplished.

Agent simulations can be classified as Markov Decision Processes [11]. An agent lives within an environment that contains a number of states it can enter. The agent has a number of actions it can use to move from state to state. RL teaches an agent how to move from state to state by analyzing the reward or punishment given to each state/action pair. An agent models realistic behavior by choosing the correct actions in the correct states. For example, a student agent might choose to move to a sitting state by going toward a desk. Taking the action of sitting will be rewarded or punished depending on aspects of the environment (class has started, class has ended, etc.).

This becomes complicated in MARL tasks as the environment is no longer stationary. The states can change due to the actions of other agents. This can leave an agent learning in an ever changing environment. Going back to the example, an agent may choose to sit at a desk as another agent closer to that desk also sits there. The state/action pair would be punished even though the same state/action pair would be rewarded at a different point in the training. The agent cannot converge on an optimal solution because the environment is changing.

In this context, this study frames the task of developing realistic AHAs in a classroom as a Markov game problem. Markov games have been shown to be an effective framework for MARL projects [11]. Markov games are similar to Markov chains, but rather than each agent in the environment learning a behavior separately, one overseeing model learns from the action/state pairs of every agent, stored in a replay buffer. The overseeing model has the context of all states and actions, allowing it to circumvent the issue of non-stationary environments. This paper specifically uses a SAC network to solve the task. SAC networks are off-policy actor-citic networks [12]. An actor in this network maximizes expected returns and entropy. The actor-critic model is an effective tool for MARL simulations [13]. This is shown in examples such as modeling and handling supply chain management [14]. It can also be demonstrated in unique environments such as particle environments and Starcraft II [15].

B. Other Approaches and Realistic AHAs

RL is not used for the development of realistic AHAs as often as other techniques. The main downside to these approaches compared to RL approaches is the generation of deterministic plans, either in terms of complexity or computational cost. Ruleset/Logic based methods are explicit sets of rules that an agent must follow. These methods can create some of the most realistic AHAs because a human must painstakingly design each possible encounter. Of course, this would be a lengthy process that is prone to error. As a result, these methods are rarely used [16].

FSM allow for agents to be in certain states that they can move between based on different actions and criteria. However, one identified downside is that their complexity increases exponentially as more states are added [2]. TBM, often represented by decision trees, allow agents to make decisions by traveling a complex set of nodes that leaf into new nodes using if/then statements. TBM are typically less complex than FSM, though this can increase the amount of fine-tuning when modeling agents with complex interactions [2]. GOAP has agents assign themselves a goal, and then chain goals together to complete more complex tasks. Each goal must be accomplished before moving to the next goal that can be accomplished. One downside to this approach is that it can also be computationally complex to calculate the next best goal [2]. More recently, Large Language Models (LLMs) have been applied to this field. One example of this is Generative Agents, where the authors simulated human behavior using calls to ChatGPT 3.5 Turbo [17]. Unfortunately, it relies on calls to the OpenAI API, which requires an internet connection and some funds.

RL approaches, with proper rewards and a sufficient NN, can learn how to interact in an environment without explicit rules or deterministic pathways. This permits the approach to circumvent the issues other approaches suffer from, making it useful for the development of realistic AHAs. Realistic AHAs are defined here as agents which closely mimic the actions and behaviors of the people they are based on. This paper uses two criteria to define realistic AHAs. The first criteria is the idea that agent 'goodness' is typically defined by optimal performance [18]. This is a subjective criteria that changes based on the game or simulation. For this project, an optimal student behavior is created and measured with a quantitative assessment. Agents learn to complete required and expected behaviors in a classroom, which are then flagged so the average rate of behavior completion can be examined. The second criteria is modeling emotions in realsitic AHAs using randomized selection models [19]. These are models that assign preset emotions to agents randomly. This study employs this method, creating agents that display one of four different personalities: shy, outgoing, curious, and disengaged.

III. Approach

The task of this experiment was the creation of realistic student AHAs in a classroom. The classroom is an environment with a number of student agents. They are able to perform actions such as sit at desks and form groups with other agents. Five RL NNs were trained using SAC so that a copy of these networks could be used by each agent at inference time. The scope included 20 agents with one of four personalities realistically navigating a typical class period in the classroom training environment.

The training environment, pictured in Fig. 1, features an exit/entrance, a teacher and the teacher's desk, 6 group tables, and 24 seats. Agents were trained in this environment over 10,000 timesteps comprising a class period. The class periods were divided into four activities: class start, lecture time, group work time, and class end. Each student was trained to perform required and expected behaviors. For example, all agents would be required to sit during the majority of lecture time and leave the class at the class end activity. These are required behaviors. An agent with the shy personality trait may choose to stay by itself during group work time. This would be an expected behavior. By providing appropriate rewards, the goal was to have agents learn realistic behaviors for students in a classroom. This was accomplished through the design of a revolving agent action technique.

A. Technique

The revolving agent action technique (Fig. 2) is a group-based, four-branch mechanism that allows an agent to perform any number of potential actions without explicitly training a model on those actions. The four branches are defined as the Group Branch, the Subgroup Branch, the Movement Branch, and the Interaction Branch. Every timestep of the training, an agent can interact with one of these four branches in a binary manner. That is, they can choose to output the code (1,2,3, or 4) for a branch to choose that branch. The training environment was tailored to this technique by grouping objects as either exits, desks, or students. Students could also target themselves to perform specific actions represented in their status (Idle, Ask Teacher, Listen, Take Notes, Use Phone, Complete Work, Talk with Friends). This technique permitted agents to learn any number of actions.

The first two branches, Group and Subgroup, allow an agent to target specific objects within the training environment. The Group branch allows the agent to cycle through groups in the training environment. For example, a classroom can be divided into groups such as desks, exits, and students. The Subgroup Branch allows agents to cycle through individual objects within those larger groups and target them. For instance, if the agent is targeting a group of desks, the subgroup branch would allow the agent to cycle through each desk and choose one as its target. Once a target is selected, an agent can use the latter two branches to interact with the target. The Movement Branch, when selected, sees an agent pathfind toward the target. For example, after selecting a desk, the agent could output code for the Movement Branch to start pathfinding toward that desk. The Interaction branch allows an agent to interact with the target. This may be restricted by certain conditions such as proximity to the target. For instance, a student would only be able to sit at a desk they targeted if they were within a certain range of that desk. In whole, the idea is that agents can learn to produce a string of action codes taken over a number of steps to perform realistic actions.

For example, assume the Group, Subgroup, Movement, and Action branches can be chosen with output of 1,2,3,4, respectively, and 0 represents the lack of an action. An agent could be trained to output the following string of actions over 6 time steps: 1,2,3,0,0,4. This represents targeting a group, targeting a member of the group, moving toward the group, taking no action for two steps, and finally interacting with the target. With this structure, all possible actions in the training environment can be represented as strings of outputs over time steps. The rationale for this approach was the view of MARL problems as Markov games. From the current state, an agent takes some action and receives some reward and ends up in a new state. When agents choose actions from their states and receive some reward, the critic of the SAC network helps determine which strings of actions provided are contributing to the highest reward. The trained policy represents the optimal string of actions an agent should take as it moves through different states.

B. The Task and Observations

The revolving agent action technique is applied here to the specific task of simulating a realistic classroom. Rewards are used to encourage agents to learn required and expected behaviors. Required behaviors represent actions that, regardless of personality, every agent should perform. Designing a "realistic" classroom is a subjective task, so these are used to represent universal behaviors that would be expected in any classroom. On the other hand, expected behaviors are meant to be completed by agents with specific personalities. These behaviors are explored below.

Two methods were tested for creating realistic student AHAs. In the first, only a single NN was trained. Agents simply observed which personality was assigned to them. While this strategy is simple, it may create a model that is the average of all the personalities. In the second strategy, a new NN was trained corresponding to each personality. This resulted in four models for the student agents. The performance of these two approaches is explored in the results section.

There were a total of fifteen observations each agent made which comprised its state. Each observation is designed to provide the critic with enough information to form meaningful state/action pairs. The observations for a state should be unique enough that the RL model can learn which chains of state/action pairs produce the highest reward:

Observation 1: If the agent's current target is a desk, is it occupied? Represented as a boolean.

Observation 2: Is the teacher currently lecturing? Represented as a boolean.

Observation 3: Is the agent currently sitting? Represented as a boolean.

Observation 4: Is the class over? Represented as a boolean.

Observation 5: Is it group work time? Represented as a boolean.

Observation 6-9: Which personality does the agent have? Represented as a one-hot encoding with four choices. The choices are shy, outgoing, curious, and disengaged.

Observation 10-12: What is the current position of the agent? Represented as a normalized 3-dimensional vector.

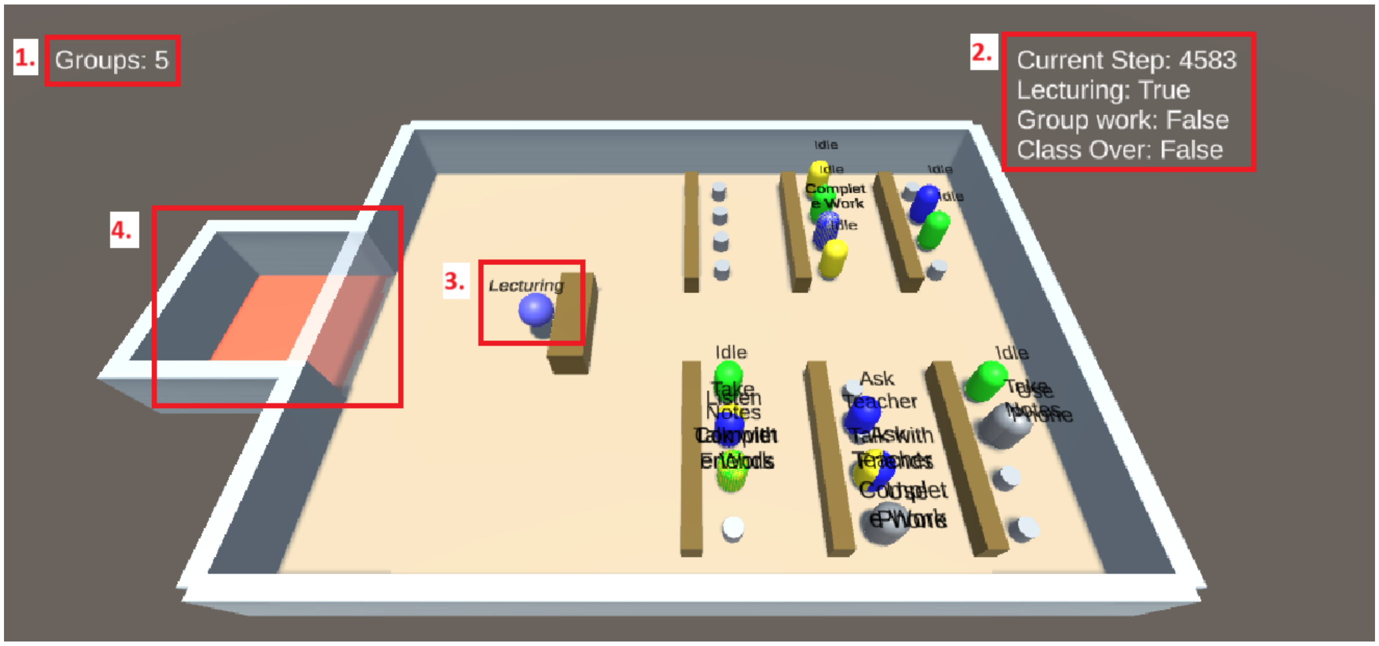

Fig. 1. The classroom environment. (1) Displays the current number of groups that are formed. (2) Displays information about the current class period. This includes the current timestep out of 10000, whether lecturing or group work is occurring, and if the class is over. (3) Is an object representing the teacher. (4) Is the exit area students use to leave the class. The personalities of students are represented by colors: shy is green, outgoing is yellow, curious is blue, and disengaged is gray.

Observation 13-15: What is the current position of the agent's target? Represented as a normalized 3-dimensional vector.

C. Rewards

Rewards can be used to encourage certain outcomes. There are 19 total rewards which encourage agents to perform the 15 required and expected behaviors for each agent. These include thirteen positive rewards and five negative rewards.

1) Required Behaviors:

Agents spend most time in class: To encourage agents to spend most of their time in class, three rewards were used: a positive reward for leaving class during the class over activity (0.01), a positive reward for grouping during the group work activity (0.01), and a negative reward for leaving class before the class over activity (-0.01).

Agents spend a majority of lecture time seated: To encourage agents to spend a majority of the lecture activity seated, two rewards were used: a positive reward for sitting during the lecture activity (0.01) and a negative reward for standing during the lecture activity (-0.01).

A majority of agents exit the classroom at the end of class: To encourage agents to exit the class during the class over activity, a positive reward was given for being at the exit during that activity (0.01).

2) Expected Behaviors:

Reluctant to group: To discourage shy agents from grouping as often, a negative reward was given for grouping during the group work activity (-0.02). This value is meant to directly offset the positive reward given to all agents to encourage grouping.

Asks less questions: To discourage shy agents from asking questions, a negative reward was given for choosing to ask questions (-0.01).

Sits near back: To encourage shy agents to sit in the back, a positive reward was given for being seated in the back row of desks (0.01).

Eager to group: To encourage outgoing agents to form groups, a positive reward was given for forming groups during the group work activity (0.01). This reward is meant to boost the positive reward given to all agents to encourage grouping.

Talk with friends: To encourage outgoing agents to talk to friends, a positive reward was given for choosing to talk to friends (0.01).

Sits near front: To encourage outgoing agents to sit in the front, a positive reward was given for being seated in the front row of desks (0.01).

Asks more questions: To encourage curious agents to ask questions, a positive reward was given for choosing to ask questions (0.01).

Does work and activities: To encourage curious agents to do work, two rewards were given: a positive reward for choosing to complete work (0.01) and a positive reward for forming groups during the group work activity (0.01).

Listen to teacher: To encourage curious agents to listen to the teacher, a positive reward was given for choosing to listen (0.01).

Leaves class: To encourage disengaged agents to leave the class, a positive reward was given for being at the exit during the lecture and group work activities (0.02). This reward is meant to offset the negative reward given to all agents to discourage leaving.

Doesn't do work: To discourage disengaged agents from completing work, a negative reward was given for choosing to complete work (-0.01).

Uses phone: To encourage disengaged agents to be distracted, a positive reward was given for choosing to use their phone (0.01).

IV. Results

A. Experimental Setup

Hardware and Software Configurations:

- 32Gb Ram at 4800 Mt/s
- 12th Gen Intel(R) Core(TM) i7-12700H at 2.30 GHz
- Unity 2022.3.4f1
- Unity ML-Agents Release 21
- Tensorboard 2.16.2.
- Windows 11 Pro
- Python 3.10.
- Torch 2.3.0.

Some of the software versions, such as Python 3.10., were not the most recent versions at the time of the experiment. This is because the Unity ML-Agents package requires specific versions of some libraries.

B. Training and Hyperparameter Tuning

Five neural networks were trained for this experiment using two methods: The mixed models method with separate networks for the four personalities (shy, outgoing, curious, disengaged), and the single model method containing every personality. The rewards for each network were tuned using Tensorboard and tested using a quantitative evaluation metric. Agents were trained in the classroom environment over 500,000 timessteps, where 10,000 timesteps comprised a full class period. Training each model took about 30-45 minutes, and a total of 50 class periods occurred during each training.

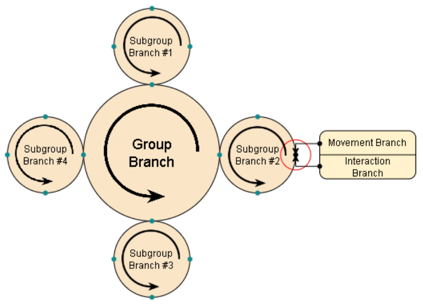

Fig. 2. Visualization of the revolving agent action technique. The Group branch can hold any number of Subgroup branches These branches contain any number of individual targets. The Group and Subgroup branches are rotated by the agent to select a target (circled in red). The agent can then select the Movement branch to move toward the target or the Interaction branch to interact with it.

TABLE I
Hyperparameters

Learning Rate		**Tau**	
High	0.001	High	0.01
Low	0.00001	Medium	0.005
Base	0.0003	Base	0.001
Recommended	0.0003	Recommended	0.005
Buffer Size		**IEC**	
High	300K	High	0.5
Medium	100K	Low	0.05
Base	50K	Base	0.01
Recommended	100K	Recommended	0.05
Hidden Units		**Layers**	
High	128	High	4
Medium	32	Low	3
Base	20	Base	2
Recommended	128	Recommended	3

For each hyperparameter listed in Table I, three values were tested by training a model and looking at the Cumulative Reward (Should increase steadily), Policy Loss (Should converge smoothly), and Entropy (Should decrease as the agent learns more confident actions), using Tensorboard. The three values for each hyperparameter were based on ranges given from the ML-agents documentation for training SAC networks [20].

The first hyperparameter tested was learning rate at values of 0.001, 0.00001, and 0.0003. Cumulative reward was highest with a value of 0.001. Policy loss converges smoothly with values of 0.001 and 0.00001. Entropy decreased evenly with a values of 0.001 and 0.0001. The second hyperparameter tested was Buffer Size at values of 300K, 100K, and 50K. Cumulative reward was highest with a value of 100K. Policy loss converges smoothly with values of 100K and 300K. Entropy decreases evenly at all values. The third hyperparameter tested was Hidden Units at values of 128, 32, and 20. Cumulative reward was highest with a value of 128. Policy loss converges smoothly with values of 128 and 32. Entropy decreases evenly at all values.

The fourth hyperparameter tested was Tau at values of 0.01, 0.005, and 0.001. Cumulative reward was highest with a value of 0.005. Policy loss converges smoothly with a value of 0.005. Entropy decreases evenly at all values. The fifth hyperparameter tested was Initial Entropy Coefficient (IEC) at values of 0.5, 0.05, and 0.001. Cumulative reward was highest with a value of 0.05. Policy loss does not converge smoothly on any values. Entropy decreases evenly at all values. The final hyperparameter tested was Layers at values of 4, 3, and 2. Cumulative reward was highest with a value of 3. Policy loss converges smoothly with values of 3 and 4. Entropy decreases evenly at all values.

The recommended values were tuned on the single model method, but used for both the single model and mixed models methods. This was because the personality observation represented a minor change to the state.

C. Analysis

Initial results demonstrated two issues: poor reward balance and overtraining. Agents had become too committed to specific actions and were not rewarded enough to complete other actions. To combat this, a few rewards were re-tuned and the number of training steps was reduced to 150,000. All hyperparameters were kept the same. This resulted in five models that performed much better than the initial models. A behavioral flagging system was set up that marked which required and expected behaviors an agent completed during a class period. It tracked 12 activities that corresponded to the 15 required and expected behaviors. The below graphs show the average values over 50 trials for each student and the average values for each personality.

In Class: Students could be in class for all 10,000 steps. Students were required to spend the majority of time in class, and it was expected that disengaged students would spend the least amount of time in class. With the single model (Fig. 3), students spent most of the class in the classroom, and disengaged students on average spent the least amount of time in class. With the mixed models (Fig. 4), students spent a majority of class in the classroom. However, disengaged students on average spent less time outside of class compared to outgoing and curious students.

Seated for Lecture: Students could be seated for the lecture activity for up to 4000 steps. It was required that students spend a majority of lecture time seated. With the single model (Fig. 5), students successfully spent a majority of the lecture activity seated. Only three students failed to meet the requirement. With the mixed models (Fig. 6), students were less successful at meeting the requirement. 10 students were able to do so, while the remaining 10 could not. The latter student were solely outgoing and curious students.

Exits Class at End: Rather than steps, this metric was measured as an average rate at which a student exited the class at the end of the class period. With the single model (Fig. 7), all students successfully exited the class a majority of the time. The lowest rate for leaving was 70 percent of the time. With the mixed models (Fig. 8), almost all students successfully exited the class a majority of the time, with the exception of student 4.

In Group: Because of how grouping was tied to rewards, student grouping was tracked every 50 steps as an agent took an action. The largest total steps a student could be grouped for was once every 50 steps over 2000 steps, or 40 steps. Students should spend a majority of the group work activity grouped, with shy students grouping the least. With the single model (Fig. 9), all students successfully spend a majority of the group work activity grouped, and shy students are on average the least likely to group. With the mixed models (Fig. 10), all but five students spend a majority of the group work activity grouped. Outgoing are the least likely to group even though they should be the most likely to group.

Questions Asked: Students can ask questions up to 10000 steps. Curious students are expected to do so the most, and shy students the least. With the single model (Fig. 11), shy students

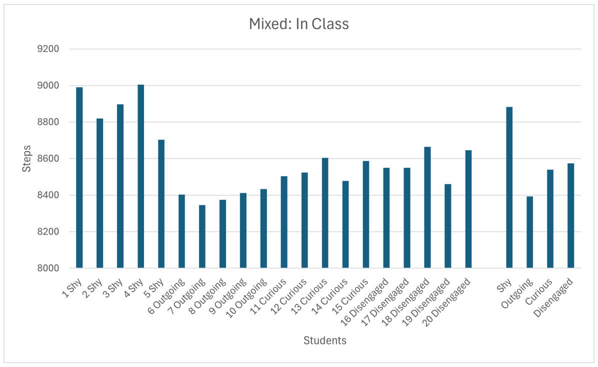

Fig. 4. Mixed Models: In Class Averages

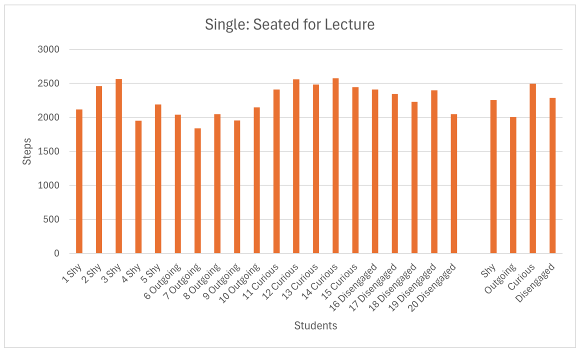

Fig. 5. Single Model: Seated for Lecture Averages

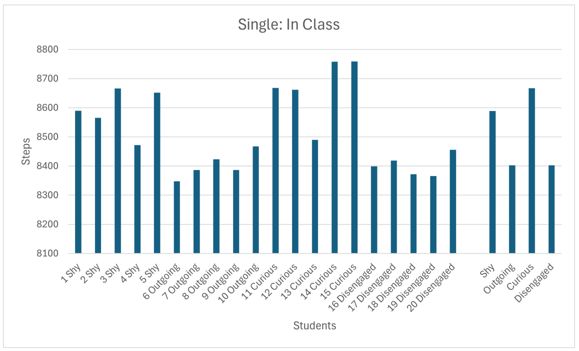

Fig. 3. Single Model: In Class Averages

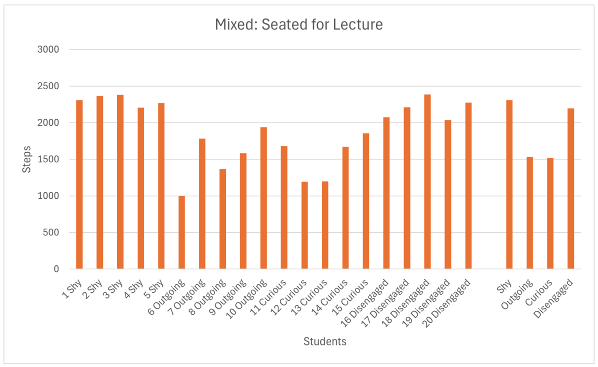

Fig. 6. Mixed Models: Seated for Lecture Averages

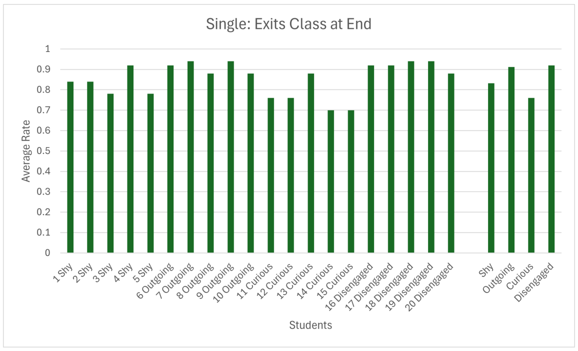

Fig. 7. Single Model: Exits Class at End Averages

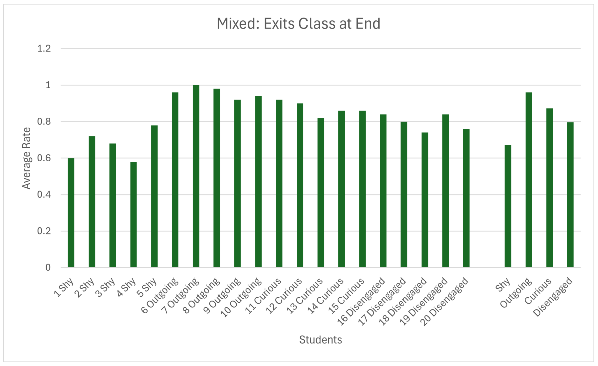

Fig. 8. Mixed Models: Exits Class at End Averages

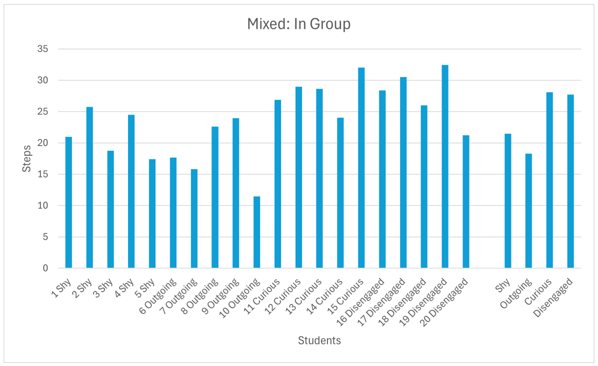

Fig. 10. Mixed Models: In Group Averages

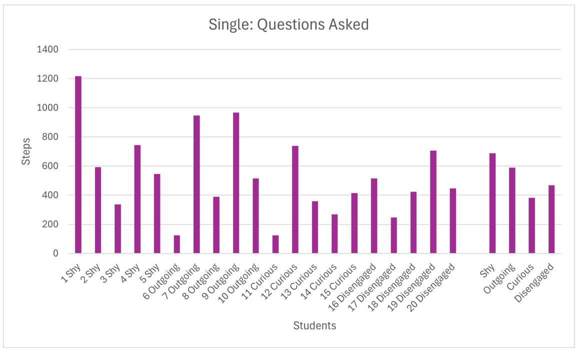

Fig. 11. Single Model: Questions Asked Averages

were the most likely to ask questions while curious students were the least likely. With the mixed models (Fig. 12), the curious students successfully asked the most questions while shy students successfully asked the least questions.

Sat in Back: Students could sit in the back row for all 10000 steps. Shy students were expected to sit in the back more than other students, especially curious students. With the single model (Fig. 13), shy students sat in the back at a similar rate to outgoing students. Interestingly, curious students sat in the back more often than both groups. With the mixed models (Fig. 14), shy students sat in the back more often than curious students, but not as much as disengaged students.

Sat in Front: Students could sit in the front row for all 10000 steps. Curious students were expected to sit in the front more than other students, especially shy students. With the single model (Fig. 15), curious students on average successfully sat in the front more often, they were tailed by shy students. With the mixed models (Fig. 16), curious students sat in the front on average more than shy students, but less than disengaged students.

Work Done: Students could complete work for all 10000 steps. It was expected that curious students would complete the most work while disengaged students would complete the least work. With the single model (Fig. 17), curious students completed the least amount of work, even less than disengaged

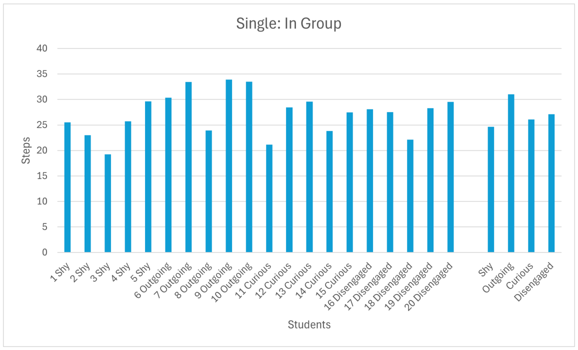

Fig. 9. Single Model: In Group Averages

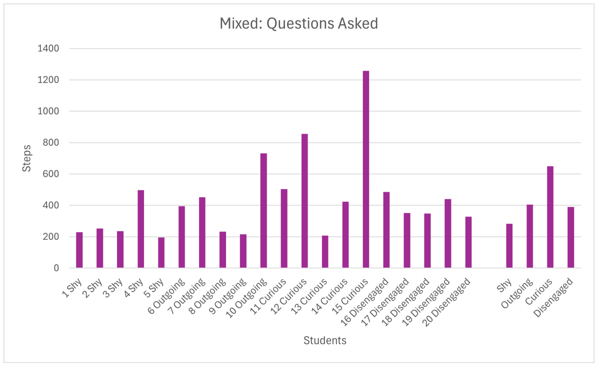

Fig. 12. Mixed Models: Questions Asked Averages

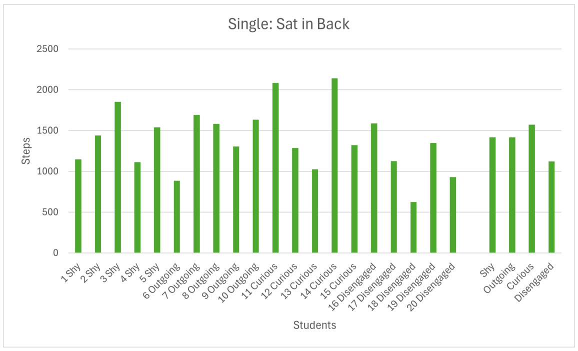

Fig. 13. Single Model: Sat in Back Averages

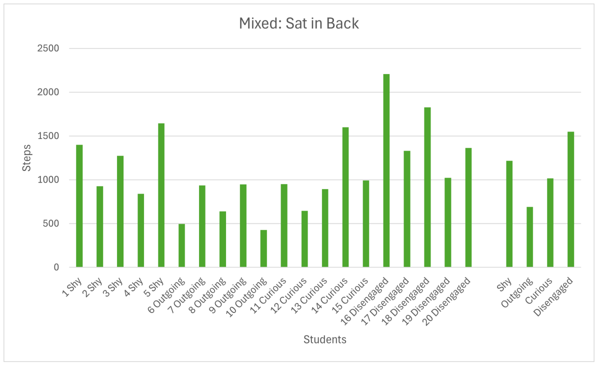

Fig. 14. Mixed Models: Sat in Back Averages

students. With the mixed models (Fig. 18), curious students successfully completed the most amount of work, though disengaged students completed the second most amount of work on average.

Listen to Lecture: Students could listen to the lecture for 4000 steps. Curious students were expected to listen to the lecture the most often. With the single model (Fig. 19), outgoing and disengaged students on average listened to the lecture more than curious students. With the mixed models (Fig. 20), disengaged students on average listened to the lecture more than curious students.

Leave Class: Students could leave the class for up to 10000 steps. It was expected that disengaged students would leave the class more often than other students. With the single model (Fig. 21), outgoing and disengaged students left the class at the same rate. With the mixed models (Fig. 22), outgoing and curious students left the class more often than disengaged students on average.

Using Phone: Students could use their phones for up to 10000 steps. Disengaged students were expected to use their phones more than other students. With the single model (Fig. 23), outgoing and shy students used their phones on average more than disengaged students. With the mixed models (Fig. 24), disengaged students successfully used their phones more often than other students.

Talk with Friends: Students could talk with friends over

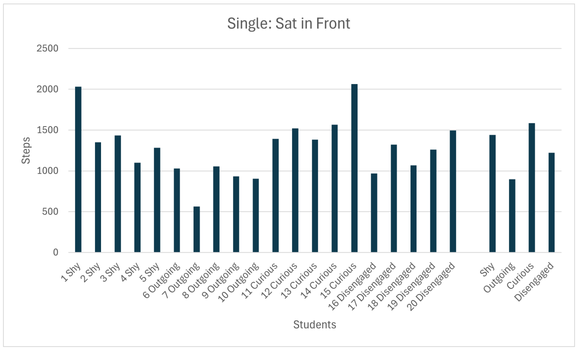

Fig. 15. Single Model: Sat in Front Averages

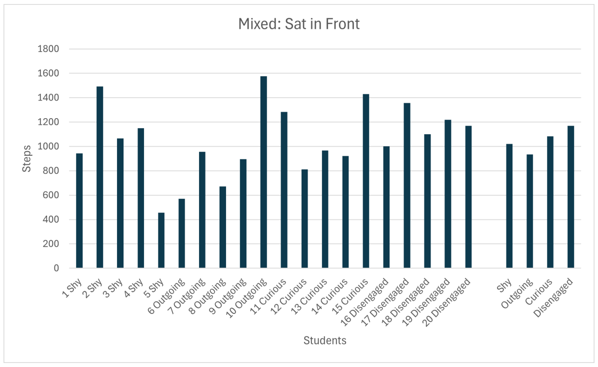

Fig. 16. Mixed Models: Sat in Front Averages

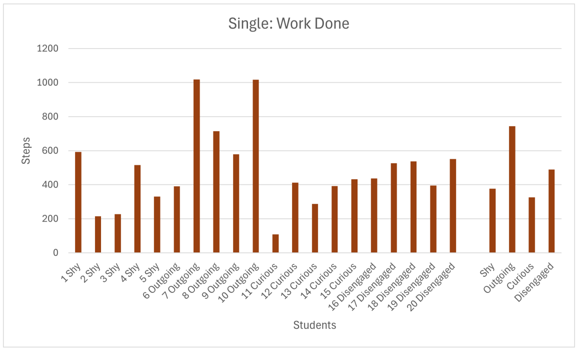

Fig. 17. Single Model: Work Done Averages

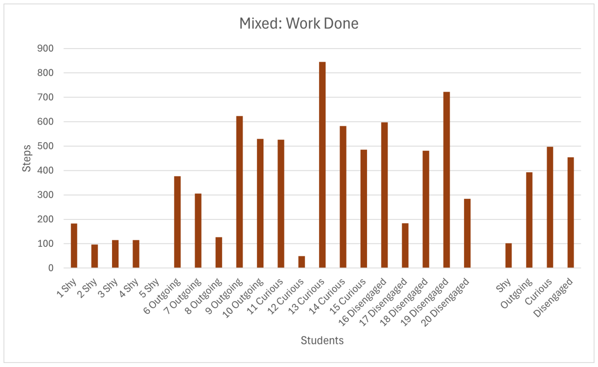

Fig. 18. Mixed Models: Work Done Averages

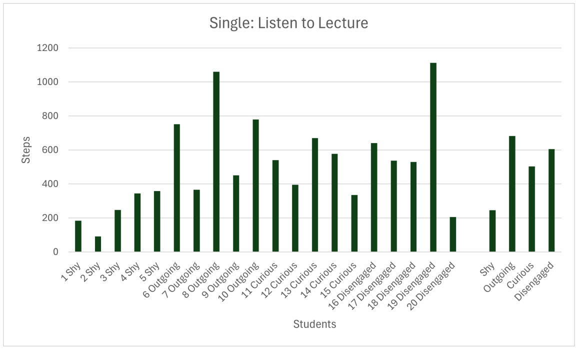

Fig. 19. Single Model: Listen to Lecture Averages

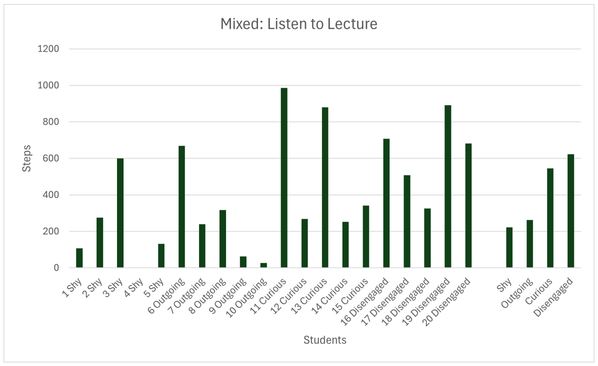

Fig. 20. Mixed Models: Listen to Lecture Averages

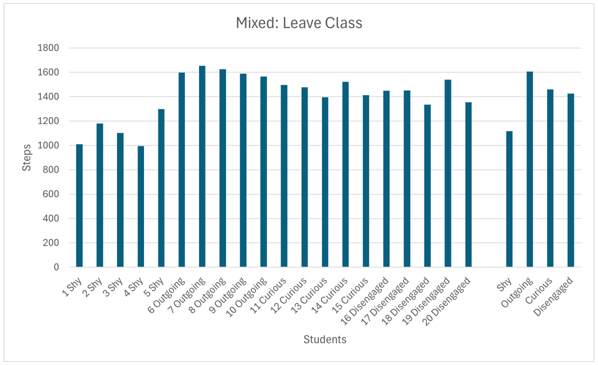

Fig. 22. Mixed Models: Leave Class Averages

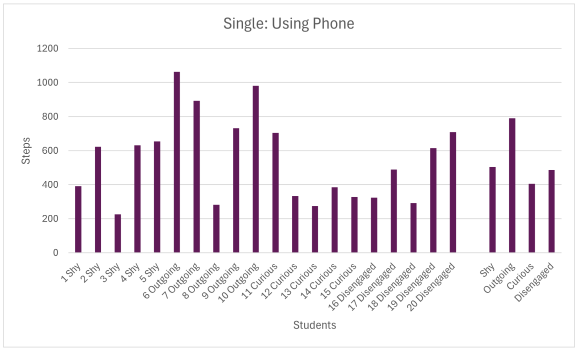

Fig. 23. Single Model: Using Phone Averages

all 10000 steps. Outgoing students were expected to talk to friends more often than other students. With the single model (Fig. 25), disengaged students talked with friends more often than outgoing students. With the mixed models (Fig. 26), disengaged students on average spent more time talking to friends than outgoing students.

Overall, the single model performed better than the mixed models, especially in the required activities. However, both were unable to fully capture all of the expected behaviors. It would appear that the agents had trouble distinguishing between personalities, especially when choosing activities on the choosing actions such as using the phone. However, while agents didn't necessarily align with their assigned personality, they still were able to perform a variety of actions to simulate students in a classroom. The results show that both the single model method and the mixed models method can produce agents that follow required activities while performing other expected activities.

V. Summary and Future Work

To further explore the burgeoning field of autonomous agents, this study examined how RL could be applied to the development of realistic AHAs. Current approaches to developing realistic AHAs rely on traditional methods that can be costly in terms of both computation and complexity, and often produce prescribed results. RL enables developers

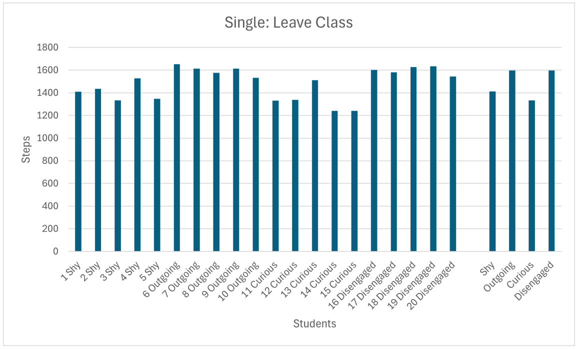

Fig. 21. Single Model: Leave Class Averages

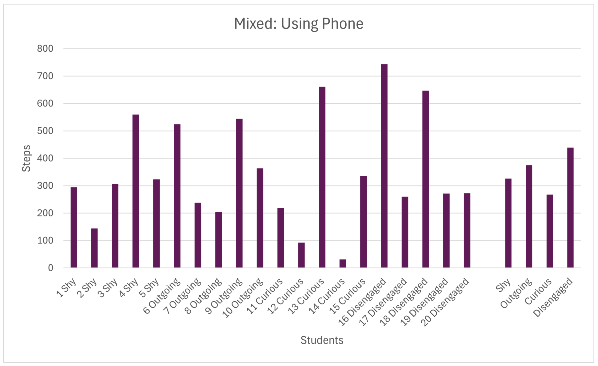

Fig. 24. Mixed Models: Using Phone Averages

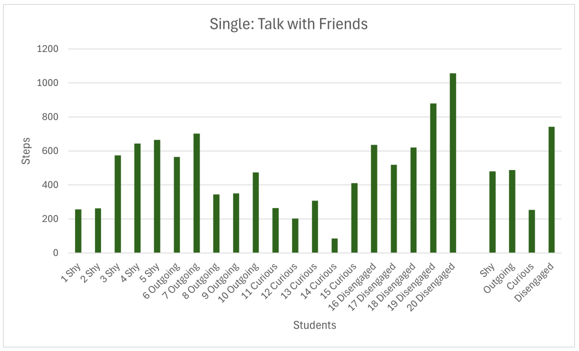

Fig. 25. Single Model: Talk with Friends Averages

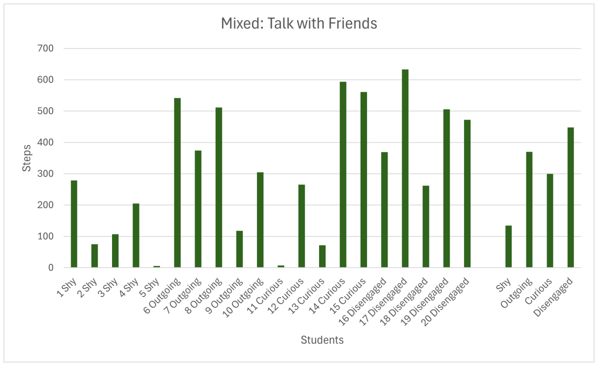

Fig. 26. Mixed Models: Talk with Friends Averages

to outline goals for agents so a NN can be taught to complete complex tasks without those costs. While RL has been used to develop AHAs, little work has focused on the development of realistic AHAs using RL.

This study defined realistic AHAs as agents with personalities that perform required and expected action. Through the design of a revolving agent action technique, it was shown that SAC networks could be used to train agents as students with unique personalities to simulate a class period. By examining the results of a single model and mixed models approach, it was determined both approaches could create agents which met required actions. However, both approaches struggled to create agents with consistent personalities, resulting in agents performing expected actions inconsistently. Regardless, this produced agents that performed actions of students with different personalities, even if those actions weren't consistent with predetermined personalities. This shows RL is a promising path for the development of realistic AHAs.

It was demonstrated in this study that designing RL agents to complete complex tasks as realistic AHAs poses unique challenges. The current RL agents are designed to learn singular state/action pairs. Future work could explore the idea of learning chains of consecutive state/action pairs. The revolving agent action technique used in this study could also be expanded to more complex tasks, like simulating students during an entire school day. The current model uses four branches, but a more complex task would require additional sub branches.

References

[1] Paul, K. (2024, December 12). Autonomous agents and profitability to dominate AI agenda in 2025, executives forecast — Reuters. Autonomous agents and profitability to dominate AI agenda in 2025, executives forecast. https://www.reuters.com/technology/artificial-intelligence/autonomous-agents-profitability-dominate-ai-agenda-2025-executives-forecast-2024-12-12/.

[2] Uludağlı, M. Ç., and Oğuz, K. (2023). Non-player character decision-making in computer games. Artificial Intelligence Review, 56(12), 14159-14191.

[3] Banks, J. (1999, December). Introduction to simulation. In Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1 (pp. 7-13).

[4] Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems, 30.

[5] Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. (2018, April). Counterfactual multi-agent policy gradients. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).

[6] Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H., Kohli, P., and Whiteson, S. (2017, July). Stabilising experience replay for deep multi-agent reinforcement learning. In International conference on machine learning (pp. 1146-1155). PMLR.

[7] Roessingh, J. J., Toubman, A., van Oijen, J., Poppinga, G., Hou, M., and Luotsinen, L. (2017, October). Machine learning techniques for autonomous agents in military simulations—Multum in Parvo. In 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 3445-3450). IEEE.

[8] Wang, Z., Kim, B., Kobayashi, H., Wu, G., and Barth, M. J. (2018). Agent-based modeling and simulation of connected and automated vehicles using game engine: A cooperative on-ramp merging study. arXiv preprint arXiv:1810.09952.

[9] Starzyk, J. A., Graham, J., and Puzio, L. (2016). Needs, pains, and motivations in autonomous agents. IEEE Transactions on Neural Networks and Learning Systems, 28(11), 2528-2540.

[10] Sutton, R. S. (2018). Reinforcement learning: An introduction. A Bradford Book.

[11] Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In Machine learning proceedings 1994 (pp. 157-163). Morgan Kaufmann.

[12] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., ... and Levine, S. (2018). Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.

[13] Li, K., Jia, Q. S., and Yan, J. (2022). An actor-critic method for simulation-based optimization. IFAC-PapersOnLine, 55(11), 7-12.

[14] Sai, D. G. S. M., Venkatraman, K., Chellammal, P., Natarajan, B., and Sridevi, R. (2023, November). A Novel Reinforcement Learning Based Optimization Approach for Supply Chain Management. In 2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS) (pp. 655-661). IEEE.

[15] Li, W., Jin, B., Wang, X., Yan, J., and Zha, H. (2023). F2a2: Flexible fully-decentralized approximate actor-critic for cooperative multi-agent reinforcement learning. Journal of Machine Learning Research, 24(178), 1-75.

[16] Jagdale, D. (2021). Finite state machine in game development. International Journal of Advanced Research in Science, Communication and Technology, 10(1), 384-390.

[17] Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. ArXiv. arXiv preprint ArXiv:2304.03442.

[18] Chang, Y. H., Maheswaran, R., Levinboim, T., and Rajan, V. (2011, October). Learning and evaluating human-like NPC behaviors in dynamic games. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (Vol. 7, No. 1, pp. 8-13).

[19] Hudlicka, E., and Broekens, J. (2009, September). Foundations for modelling emotions in game characters: Modelling emotion effects on cognition. In 2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops (pp. 1-6). IEEE.

[20] Juliani, A. (2018). Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627.