GAMES AND ARTIFICIAL INTELLIGENCE

ASSIGNMENT 3

Name: Syed Ahsan Ali
Student ID: S3736294
Due Date: 11/06/2023
Instructor: Dr. Michael Dann

DESIGN AND INTENTION

For this assignment involving the application of machine learning techniques, I have opted for a template game. The primary reason for this choice is my desire to train a model using reinforcement learning, and the existing framework of this game serves as the perfect groundwork. The provided setup allows me to invest my efforts primarily in training and a methodical trial-and-error process to practically apply the theoretical concepts we've learned in class.

I've titled this template game "EscapeQuest: Vital Sustenance". The title is a combination of two central themes: 'EscapeQuest' stands for the player's objective to evade enemies and make their way to the level's exit, and 'Vital Sustenance' signifies the need to preserve health and stamina. As the title suggests, the player's goal is to reach the exit of each level while maintaining their health and avoiding enemies. The player's game ends only when their health is completely depleted. There are several elements in the game environment that can diminish a player's health:

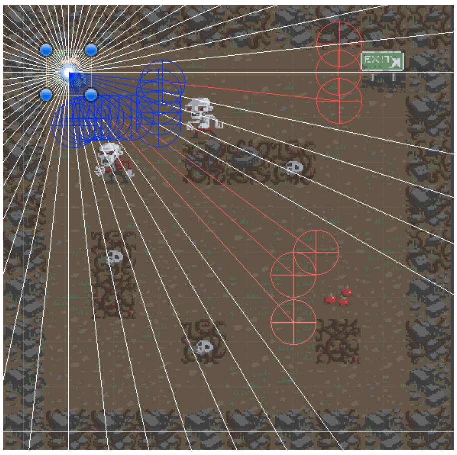
- Enemy Contact: If a player comes into contact with an enemy, they lose 10 or 20 health points, depending on the enemy type.
- Wall Collisions: The game environment features walls. Crashing into these results in a loss of 1 health point.
- Environmental Navigation: Simply moving around the game environment reduces the player's health. Each step taken deducts 2 health points.

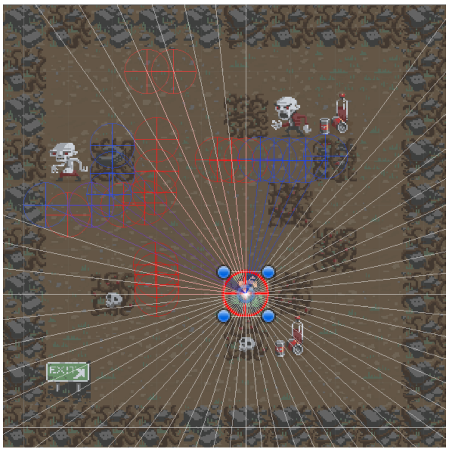
To counterbalance these health-eroding factors, the game includes food and drinks for consumption to boost health. Food replenishes 10 health points, while soda restores 20 points.

IMPLEMENTATION AND SCIENTIFIC EVIDENCE

The game's code was already provided, and my task was to make adjustments to the Player Agent file. We utilized the MLAgent in the training process, with a yaml file serving to configure and define various MLAgent toolkits. I began training with a rudimentary rule - the player's aim is to reach the exit, receiving one point as a reward. This initial setup didn't include any hindrances, adversaries, or health configurations.

To observe the environment, objects, goals, and enemies, I decided to equip the player with a ray perception sensor. This enabled the player to perceive the environment's state and location. I made the enemy-related rays blue, unlike the other red rays, to better identify the most significant threats. This is what the ray cast appears like within the game.



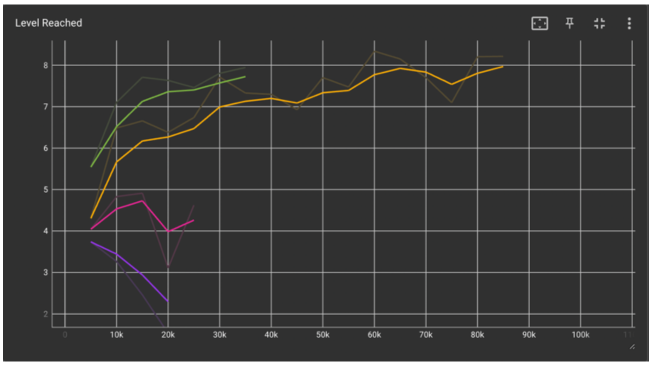


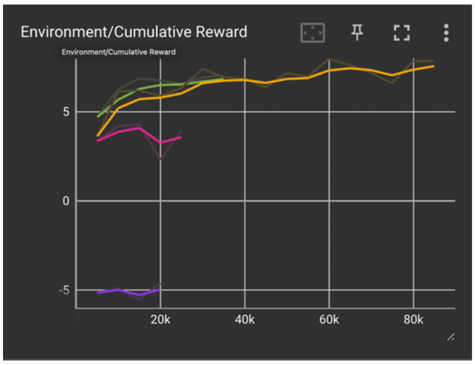
Next, I began to reward the player positively or negatively based on its impact on health, observing how this influenced the ultimate objective of progressing to the highest levels. Initially, I allocated 0.3 and 0.5 points respectively for consuming food and soda, penalized movement with -0.3 points, and applied negative points for losing to enemies. However, allocating high rewards to food and soda ended up distracting the player from the main goal - exiting the level. Given the model's propensity to optimize reward accumulation, it started focusing on health improvement. To balance this, I performed trial-and-error iterations to comprehend this function's impact on the game and derived the most suitable reward value, striking a balance between reaching the exit and maintaining health.

Penalizing the player for movement and enemy encounters was proving counterproductive, as it discouraged the player from reaching the exit point due to the fear of losing rewards. Therefore, I contemplated implementing a variable reward system. Under this system, if the player's health was above 50 points, health reduction would be proportional to the damage taken from enemies relative to the remaining health. Otherwise, a maximum health reduction would be applied. Given the reward system's relative nature, I decided not to penalize the player for movement to prevent excessive reward-loss aversion.

Moreover, I introduced a higher reward for consuming food and soda when the player's health was critically low. This way, the player would strive to avoid substantial penalties and, when health was low, would be incentivized by larger rewards to regain health. These modifications significantly enhanced the game performance at advanced levels. The graphical representations illustrate the distinct improvements achieved as a result of these changes when compared to the previous outcomes.

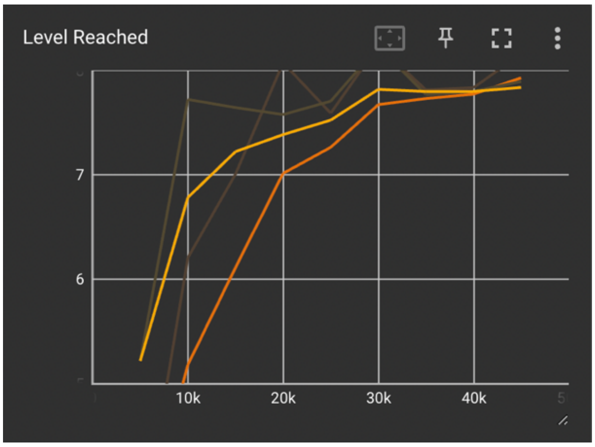
The yellow graph represents the updated progression with the recent modifications. The green graph denotes the initial progression, where no rewards were implemented, except for reaching the exit level. The pink graph illustrates a model preoccupied with collecting rewards from soda and food, thereby making no advancement in levels. The purple graph indicates a model that receives negative rewards and becomes overly cautious to avoid these negative rewards, resulting in a lack of progression through the levels.

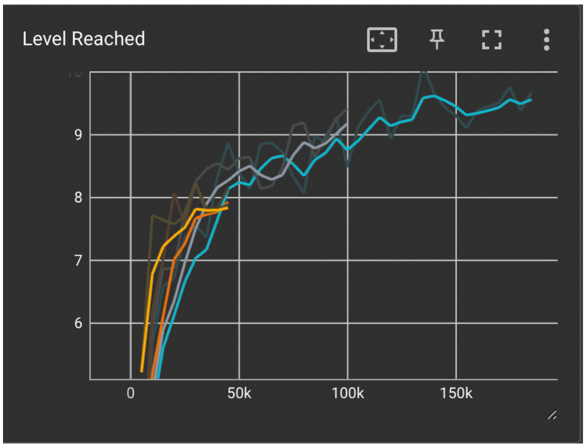




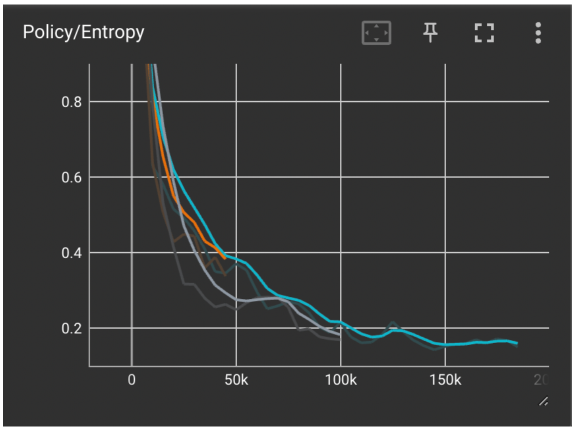
While the outcomes had shown some improvement, the rewards plateaued, and progress seemed to cap around level 7. I wanted to ensure that the training model sustained its learning rate and continually advanced through the levels. To this point, I hadn't altered the Collect Observation function, merely monitoring the player's transform position. I then considered tracking the distance to various elements such as objects, food, the goal, and enemies. I individually incorporated these observations into the model, anticipating performance enhancement.

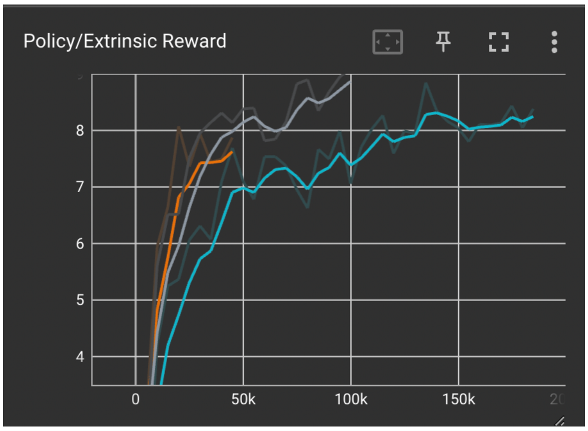
Indeed, this change propelled the learning curve to approximately level 8. I compared this with earlier models, which showed similar outcomes at 45000 steps. However, I noticed a difference - the older models plateaued, while the levels achieved by the newly trained model continued to rise, as indicated by the graph. Recognizing this model's potential, I decided to train it again, this time for an extended period. This led to a significant increase, with the model reaching about 10 levels. The yellow graph is the plateaued one whereas the orange one shows potential. The second includes the blue graph which is an orange graph trained for an extended period. The grey is final graph. At 100k steps, the grey crossed the level 9 mark while the blue is just short of that.





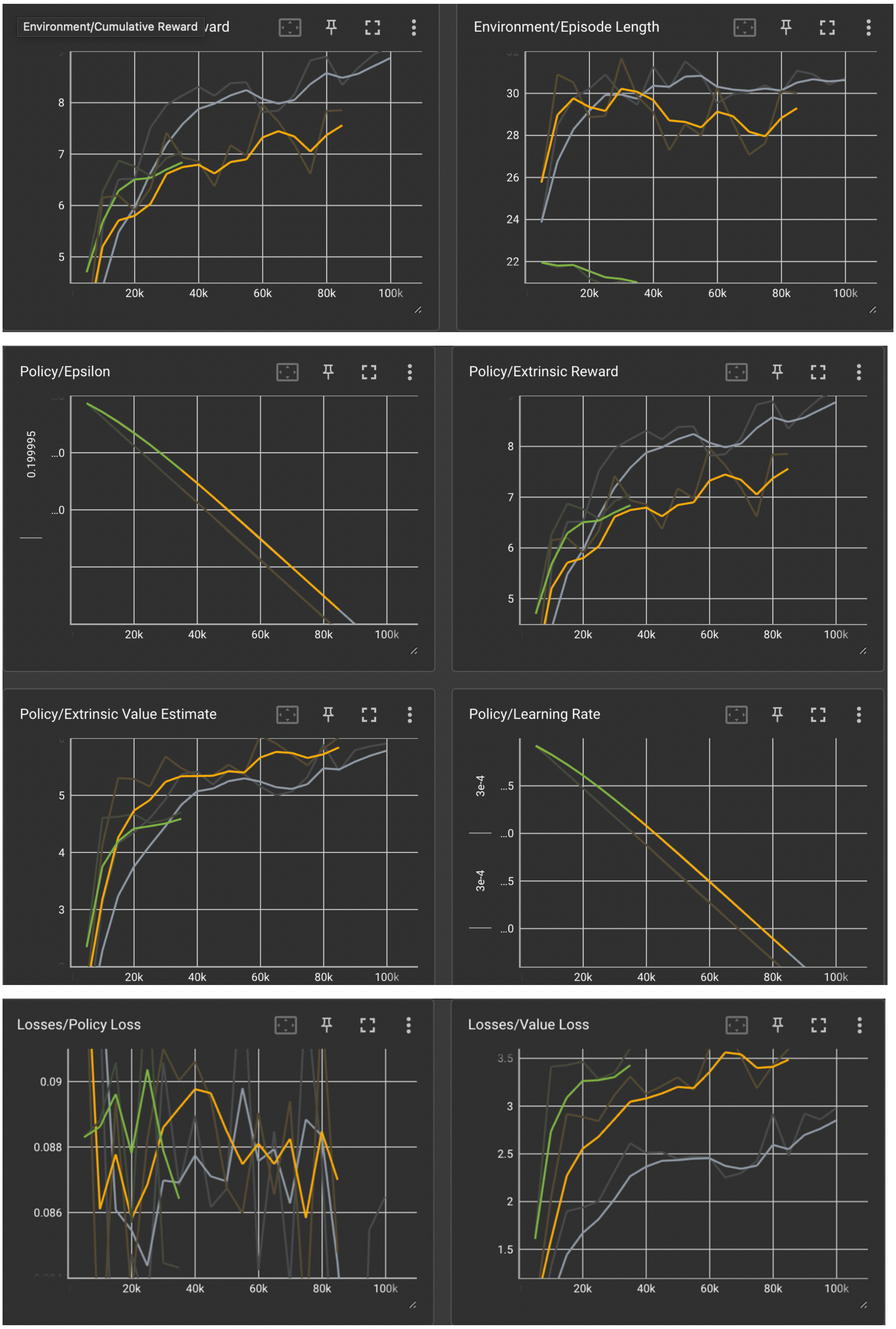
In addition, the entropy reduced more rapidly. During training, I noticed that the highest level achieved spiked to 22, a notable increase from the prior maximum of 15 I had observed. Here is the graph of policy/entropy and policy/rewards for the last and most efficient models. The grey one is the final trained model.





I also attempted to modify the yaml file, experimenting with changes to the batch and buffer size, learning rate, beta value, number of epochs, and the number of hidden layers and their units. However, the graphs didn't indicate any significant enhancement towards the intended goal. In fact, increasing the batch and buffer size seemed to decelerate the rate and reduce the model's efficiency. Increasing the number of epochs sped up the step count, but it didn't assist in achieving higher levels. Consequently, I decided to stick with the initial configurations, given that all previous trainings were based on them. During my research, I explored the concept of curiosity, which takes the agent's current and subsequent observations, encodes them, and uses the encoding to predict the intervening action. Despite tweaking its strength, gamma, and encoding size, there wasn't any substantial impact on the final outcome.

The graphs show the comparison in environment, loss, and policy of initial, mid and final models.



CONCLUSION

While the initial performance was not in line with my original predictions, the iterative process of refinement led to substantial improvements. My original prediction was that a simple reward and penalty system, coupled with basic environment observations, would suffice in guiding the AI agent towards achieving its primary goal. However, the reality was more complex, as the agent's behaviour was heavily influenced by the reward system's specifics and the degree of information obtained from its observations.

Yes, I was ultimately able to achieve the behaviour I wanted, but not without significant adjustments. The initial design of the reward system unintentionally prompted the AI agent to prioritize health restoration overreaching the exit, which was unexpected. By iteratively adjusting the rewards and penalties associated with health-related actions and introducing more nuanced observations, the agent's behaviour evolved to balance health maintenance and progress towards the exit, which was the intended objective.

The most unexpected behaviour was how the AI agent became overly focused on accumulating rewards when over-incentivized to consume food and soda. This necessitated a careful balancing act, adjusting the reward values associated with different actions to guide the agent towards a more optimal behaviour. Regarding the implementation of this approach in a commercial game, I believe there is significant potential. While challenges were encountered, the process of iteratively adjusting the reward system and agent observations demonstrated the flexibility and adaptability of reinforcement learning. The lessons learned from this project regarding fine-tuning AI behaviour could be valuable in creating more sophisticated and engaging gameplay experiences in a commercial setting.

However, it is important to note that reinforcement learning approaches require careful tuning and adjustment, making them more time-consuming and resource-intensive than more deterministic AI approaches. Additionally, while the results were promising, they were achieved in a specific game environment, and similar results may not be directly translatable to all types of games or scenarios. As such, I would advocate for this approach as one tool among many in a game developer's toolkit, to be used when the game's design and objectives align well with the strengths of reinforcement learning.