**Design Defense: Pirate Intelligent Agent**

**Jonathan Skop**

**Date 10/13/25**

**Project Two**

**Introduction**

Artificial intelligence (AI) enables machines to learn from data and experience in ways similar to how people make decisions. In this project, I developed an intelligent agent for a pirate character in a treasure-hunting game. This agent uses a Deep Q-learning algorithm to navigate a maze and retrieve the treasure before the human player does. This design defense provides an overview of how the intelligent agent was built, examines its behavior, evaluates the algorithm's performance, and compares machine learning strategies to human problem-solving approaches.

**Human vs. Machine Approaches to Problem Solving**

When a human tackles a maze, they rely on their perception, reasoning, and memory. For instance, a player might scan the environment for open pathways, backtrack upon hitting dead ends, and recall which routes they've already explored. Human strategies often hinge on intuition, spatial reasoning, and the capacity to adapt their decisions based on past experiences.

On the other hand, the pirate agent approaches the maze as a reinforcement learning challenge. Each state is represented by a collapsed version of the maze grid, with possible moves defined as up, down, left, or right. Rather than relying on instinct, the agent assesses each action through a Q-value function that predicts potential rewards. The model learns iteratively, fine-tuning its Q-values based on the outcomes it encounters, with experience replay helping to maintain stability.

While both approaches share the need to explore uncharted paths and adapt to feedback, the key distinction lies in how they process that feedback: humans utilize reasoning and memory, whereas machines rely on mathematical updates to a neural network.

**Purpose of the Intelligent Agent**

The pirate agent is designed to tackle a pathfinding problem in a competitive setting. It learns through reinforcement learning, balancing the need to explore new moves with the strategy of sticking to what works best based on past rewards.

For this project, we employed an ε-greedy strategy, initially setting the agent's exploration rate to 1.0 (ε) and gradually reducing it to 0.05. This way, the agent initially tried many different actions but eventually relied more on its best strategies as training progressed. This approach demonstrates how reinforcement learning agents can optimize their navigation in complex environments.

**Evaluating the Algorithm**

The pirate agent was built using Deep Q-learning (DQN). It had a neural network with two hidden layers, each with 128 neurons and ReLU activations, plus a linear output layer that shows Q-values for all the possible actions. They used the Adam optimizer with a learning rate of 0.001 and a discount factor γ of 0.95.

**The algorithm added two cool upgrades to standard Q-learning:**

**1. Experience Replay-** it keeps a record of past experiences and picks random batches for updates, which helps keep things stable.

**2. Target Network -** it updates a secondary network now and then to avoid wild fluctuations.

They trained it over 400 episodes. The win rate was relatively low overall, but the logs indicated that the agent explored less as ε decreased and began to focus more on higher-valued actions. This proves that the setup was working as it should, but with additional training or some algorithmic tweaks, it could perform better.

**Finished Results**

The pirate agent was trained using Deep Q-learning in a maze environment. Despite low win rates and incomplete checks, the model showed consistent improvement. As training progressed, the epsilon value decreased, leading to more strategic action selection, which indicated that the framework was functioning correctly. With additional training episodes and potential algorithm refinements, the agent's performance could improve significantly.

**My Conclusion**

This project illustrates the application of reinforcement learning within a pathfinding context. The pirate agent effectively used a Deep Q-learning framework, successfully balancing exploration and exploitation while refining its decision-making process via Q-value updates. Although the outcomes demonstrated some limitations in terms of consistent victories, there was clear evidence of the agent's learning progress. This underscores both the advantages and challenges of applying reinforcement learning to game environments. With further training and strategic enhancements, performance could be significantly improved, leading to more consistent success.

**References**

Chollet, F. (2017). *Deep learning with Python*. Manning Publications.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533. https://doi.org/10.1038/nature14236

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.