SKY MESSIONS

Machine Learning & Game Development Project

By

Hadi Rida

Ahmad Ayoub

Submitted to the School of Arts & Science of the

Lebanese International University Beirut, Lebanon

COMPUTER SCIENCE

Supervised by: Dr. Ali Ballout

Fall 2025-2026

Abstract

This project combines **Reinforcement Learning** and **Unity game development** to create a drone that autonomously navigates a dynamic environment. The drone is trained using ML-Agents to reach targets and avoid obstacles, while the game allows player interaction and comparison with the AI’s behavior. The work demonstrates the integration of AI-driven agents into interactive gameplay.

1. Introduction

Modern games are increasingly using **artificial intelligence** to create dynamic and realistic interactions. **Reinforcement Learning (RL)** allows agents to learn optimal behaviors through trial and error, making it a powerful tool for simulating autonomous entities in a virtual environment.

This project focuses on designing a **drone agent** that learns to navigate a Unity environment, reach designated and moving targets, and avoid obstacles. The goal is to apply RL techniques to solve a complex navigation task while exploring the integration of AI into gameplay.

The trained drone is incorporated into a **playable game**, where players can either observe the AI's behavior or interact directly through manual control. This approach demonstrates how **ML-driven agents** can enhance game experiences and highlights the process of developing, training, and integrating AI within a game engine.

2. Machine Learning (Reinforcement Learning)

2.1. Background

Reinforcement Learning (RL) is a type of machine learning where an agent learns optimal behavior through interaction with an environment. The agent receives **observations**, performs **actions**, and gets **rewards** based on the outcome. Over time, it learns a **policy** that maximizes cumulative reward.

In this project, RL is used to train a **drone agent** in Unity to autonomously deliver packages by navigating to target rooms while avoiding obstacles. The RL agent learns behaviors that are challenging to program manually, making it suitable for dynamic game environments.

2.2. Problem Formulation

The RL task for the drone is defined as follows:

- **Agent(DroneBrain):** Drone controlled via Rigidbody physics.

- **Environment:** 3D Unity scene with multiple rooms, moving or static targets, a home base, and obstacles.

- Observations (inputs):

  - Drone's position and velocity
  - Current target position
  - Home base position
  - Relative vector to target
  - Distance to target

- Drone’s height relative to min/max limits
- Movement status (returning home or delivering)
- Target velocity if moving
- Obstacle direction, density, distance, and danger level (via Physics.OverlapSphere)

- Actions (outputs):

  - Continuous 3D movement: X, Y, Z forces applied to Rigidbody

- Reward Function:
  - Positive reward for reaching target or successfully returning home
  - Progress-based reward for getting closer to the target
  - Penalty for collisions or violating height limits
  - Small penalty per timestep to encourage efficiency
  - Bonus for speed (fewer steps per successful delivery)

- Episode Termination:
  - Drone reaches target or home base
  - Drone collides with obstacles
  - Maximum steps reached

2.3.1. Environment Design

The Unity environment was designed with the following elements:

- **Drone GameObject:** Rigidbody physics, velocity/rotation control, collision detection.
- **Targets:** Room-based targets, which can be static or moving (linear or circular motion).
- **Home Base:** Randomized spawn area for deliveries and drone start positions.
- **Obstacles:** Detected using `Physics.OverlapSphere` with configurable sensing radius.
- **Sensors:** Ray perception replaced with Physics.OverlapSphere for obstacle detection.

2.3.2. Target Randomization and Obstacle Avoidance Strategy

To improve training efficiency and generalization, both the target position and the home base are randomly spawned at the beginning of each episode. This prevents the agent from memorizing fixed paths and encourages learning a generalized navigation policy.

Obstacle avoidance is implemented using a **sphere-based sensing approach**, where nearby obstacles are detected using a spherical overlap region. This method has several advantages: it is computationally efficient, simple to implement, and does not suffer from blind spots that can occur with limited ray-based sensing.

An alternative approach is **ray-cast–based obstacle detection**, which simulates real-world drone ultrasonic sensors. Ray-based sensing provides directional awareness and helps the agent identify optimal paths around obstacles. However, it may introduce blind spots depending on ray density and configuration.

Through experimentation, a **hybrid approach** combining sphere-based detection and ray-based sensing was found to be the most effective. The sphere ensures complete obstacle awareness, while ray-based sensing enables better path planning toward the target.

2.4. RL Algorithm and YAML Configuration

- **Algorithm:** Proximal Policy Optimization (PPO) using Unity ML-Agents
- **Reason for choice:** PPO is stable for continuous action spaces and widely used in game environments.
- Key hyperparameters from YAML:

Parameter	Value
batch_size	2048
buffer_size	20480
learning_rate	0.00025
beta	0.005
epsilon	0.2
lambda	0.95
num_epoch	3
learning_rate_schedule	linear
hidden_units	256
num_layers	3
gamma	0.99
reward strength	1.0
max_steps	20,000,000
time_horizon	1000

2.5. Training Process

- Training was conducted inside the Unity Editor using the above PPO configuration.

- The agent trained for several million steps until the cumulative reward stabilized.

- Curriculum learning was optionally enabled to gradually increase difficulty (smaller delivery zones, moving targets).

- Challenges during training:
  - Balancing the reward function to prevent shortcuts
  - Avoiding the drone getting stuck at boundaries
  - Handling moving targets with circular motion or linear trajectories

2.6.1. Training Challenges: Catastrophic Forgetting

During training, a challenge related to **catastrophic forgetting** was observed. After introducing obstacle avoidance, the agent occasionally forgot how to efficiently reach the target. This occurred because the agent could receive positive rewards for avoiding obstacles even when moving temporarily farther from the target, which conflicted with the primary navigation objective.

This conflict caused instability in learning, where improvements in obstacle avoidance negatively affected target-reaching performance.

To address this issue, **reward shaping** was applied. When an obstacle is detected, the agent is temporarily discouraged from prioritizing target proximity rewards. Instead, the agent focuses on safely avoiding the obstacle before resuming target navigation. Once the obstacle is cleared, the original distance-based reward toward the target is restored.

2.6.2. Training Metrics and Logs

Training performance was monitored using logged metrics including cumulative reward, mean reward, and episode length (steps). Both successful and unsuccessful training phases were recorded to analyze learning behavior.

These logs demonstrate the effect of reward shaping and obstacle avoidance on learning stability and performance improvement over time.

A. Early Training Instability (Before Reward Shaping)

```
[INFO] DroneBrain. Step: 20000. Time Elapsed: 32.450 s. Mean
Reward: -19.115. Std of Reward: 1.514. Training.
[INFO] DroneBrain. Step: 30000. Time Elapsed: 44.789 s. Mean
Reward: -20.815. Std of Reward: 2.735. Training.
[INFO] DroneBrain. Step: 40000. Time Elapsed: 53.159 s. Mean
Reward: -18.592. Std of Reward: 2.250. Training.
[INFO] DroneBrain. Step: 50000. Time Elapsed: 65.829 s. Mean
Reward: -85.027. Std of Reward: 112.324. Training.
[INFO] DroneBrain. Step: 60000. Time Elapsed: 74.607 s. Mean
Reward: -19.802. Std of Reward: 1.292. Training.
[INFO] DroneBrain. Step: 70000. Time Elapsed: 86.731 s. Mean
Reward: -19.169. Std of Reward: 0.596. Training.
[INFO] DroneBrain. Step: 80000. Time Elapsed: 95.685 s. Mean
Reward: -20.372. Std of Reward: 0.416. Training.
[INFO] DroneBrain. Step: 90000. Time Elapsed: 108.472 s. Mean
Reward: -19.480. Std of Reward: 1.756. Training.
```

These logs demonstrate severe reward collapse caused by conflicting objectives between obstacle avoidance and target reaching. High reward variance indicates unstable learning behavior.

B. Stabilized Training (After Reward Shaping)

```
[INFO] DroneBrain. Step: 2010000. Time Elapsed: 42.917 s. No
episode was completed since last summary. Training.
[INFO] DroneBrain. Step: 2020000. Time Elapsed: 56.026 s. Mean
Reward: 233.197. Std of Reward: 1.777. Training.
[INFO] DroneBrain. Step: 2030000. Time Elapsed: 71.284 s. Mean
Reward: 236.942. Std of Reward: 0.422. Training.
[INFO] DroneBrain. Step: 2040000. Time Elapsed: 85.764 s. Mean
Reward: 241.370. Std of Reward: 26.538. Training.
[INFO] DroneBrain. Step: 2050000. Time Elapsed: 105.690 s. Mean
Reward: 210.853. Std of Reward: 38.163. Training.
[INFO] DroneBrain. Step: 2060000. Time Elapsed: 118.830 s. Mean
Reward: 230.695. Std of Reward: 4.204. Training.
[INFO] DroneBrain. Step: 2070000. Time Elapsed: 137.467 s. Mean
Reward: 215.717. Std of Reward: 0.000. Training.
[INFO] DroneBrain. Step: 2080000. Time Elapsed: 150.544 s. Mean
Reward: 238.122. Std of Reward: 12.257. Training.
[INFO] DroneBrain. Step: 2090000. Time Elapsed: 173.076 s. Mean
Reward: 230.324. Std of Reward: 31.593. Training.
[INFO] DroneBrain. Step: 2100000. Time Elapsed: 184.635 s. Mean
Reward: 227.224. Std of Reward: 16.564. Training.
```

After applying reward shaping, the agent demonstrated consistent improvement in mean reward and reduced variance, confirming successful mitigation of catastrophic forgetting.

2.6.3. Results and Analysis

- Training curves showed steady improvement in cumulative reward over episodes.

- The drone learned to:

  - Navigate efficiently toward the target
  - Avoid obstacles
  - Return home after deliveries when required

- Observed behaviors:

  - Some targets' circular motion introduced slight learning delays
  - Occasional collisions when navigating near complex obstacles

- Success rate improved with curriculum learning over time.

2.7. Discussion

- Future improvements:

  - More complex or randomized obstacle layouts
  - Dynamic target priorities
  - Multi-agent training for simultaneous drones
  - Experimenting with other RL algorithms like SAC for faster convergence

2.8. Reinforcement Learning Workflow Diagram

```
Environment (Unity Scene)
   ├── Targets
   ├── Obstacles
   ├── Home Base
        ↓ Observations
Drone Agent (ML-Agents)
        ↓ Actions
Unity Physics (Rigidbody Forces)
        ↓
New State + Reward
        ↓
PPO Algorithm (Training)
        ↺ (Loop)
```

- At the beginning of each episode, both the target and home base are randomly spawned to improve generalization and prevent overfitting to fixed locations. Obstacle avoidance is handled using a sphere-based sensing system, which provides continuous environmental awareness without blind spots.