**Knowledge Transfer Document: Sector-Based Reinforcement Learning for Drone Navigation in Urban Environments**

**1. Project Overview**

This project developed a sector-based reinforcement learning (RL) framework for autonomous drone navigation in urban environments, implemented using Unity ML-Agents and the Proximal Policy Optimization (PPO) algorithm. The system divides the environment into sectors to reduce computational complexity, enabling efficient navigation through dense urban settings. The drone achieves robust performance, with high success rates and low collision rates, as validated through extensive simulations.

**2. Key Findings**

* **Sector-Based Efficiency**: Dividing the environment into sectors reduced state space complexity by up to 40%, accelerating training and improving navigation performance.
* **Reward Design**: A carefully tuned reward function balancing distance reduction, collision avoidance, and orientation stability was critical to success.
* **Curriculum Learning**: Progressive training with increasing complexity enhanced the drone's adaptability to diverse urban scenarios.
* **Scalability**: The framework is extensible to other autonomous systems and larger environments.

**3. Technical Details**

**3.1 Tools and Technologies**

* **Unity Editor**: Version 2021.3.22f1 for simulation and visualization.
* **Python**: Version 3.9 with mlagents==0.30.0 for training.
* **ML-Agents**: Unity's RL framework for integrating PPO with Unity simulations.
* **C# Scripts**: Core logic for drone control, physics, and RL (see Assets/Scripts/).

**3.2 Core Components**

* **IP\_Drone\_Agent.cs**:
  + Manages RL observations (position, velocity, target distance), actions (pitch, roll, yaw, throttle), and rewards.
  + Implements reward calculations for distance, orientation, alignment, and penalties (see RewardParameters class).
* **IP\_Drone\_Engine.cs**:
  + Simulates physics-based propulsion using Rigidbody forces.
  + Handles propeller rotation for visual feedback.
* **Normalization.cs**:
  + Normalizes observations using sigmoid and tanh functions to stabilize training.
* **IP\_Base\_RigidBody.cs**:
  + Configures drone physics (mass, drag) for realistic dynamics.
* **IP\_Drone\_Inputs.cs**:
  + Supports manual control for heuristic testing and debugging.
* **TrailManager.cs**:
  + Visualizes drone paths for performance analysis.

**3.3 Environment Setup**

* **Urban Environment**: 3D Unity scene with buildings, roads, and obstacles.
* **Sector Division**: Environment split into interconnected sectors, each with localized navigation tasks.
* **Target Selection**: Randomly selected targets from a predefined list (see IP\_Drone\_Agent.cs, SelectRandomTarget).

**3.4 Training Process**

* **Configuration**: Defined in DroneNavigation.yaml.
* **Hyperparameters**: Learning rate, batch size, and reward multipliers tuned iteratively.
* **Curriculum**: Starts with simple environments, progressively adding obstacles and sector transitions.

**4. Lessons Learned**

* **Sector Transitions**: Require smooth handling to avoid navigation disruptions; consider overlap zones in future iterations.
* **Reward Tuning**: Iterative tuning of reward parameters (e.g., distanceRewardMultiplier, collisionPenalty) was essential for balancing exploration and exploitation.
* **Curriculum Design**: Gradual complexity increase prevented training instability.
* **Debugging**: Visual tools like TrailManager and manual control (IP\_Drone\_Inputs) were critical for identifying issues.

**5. Challenges and Solutions**

* **Challenge**: High state space complexity in dense environments.
  + **Solution**: Sector-based division reduced complexity, focusing on local tasks.
* **Challenge**: Unstable training due to reward imbalances.
  + **Solution**: Fine-tuned reward parameters and normalized observations.
* **Challenge**: Sector transition errors.
  + **Solution**: Added alignment rewards to maintain directionality across sectors.

**6. Documentation and Resources**

* **Repository**: https://github.com/Prime-AI28/Unity-ML-Drone.git
* **Configuration**: Training parameters in Assets/ML-Agents/Configs/DroneNavigation.yaml.
* **Code Snippets**: Sensor definitions and reward calculations in github
* **Setup Guide**: See README.md for installation and running instructions.

**7. Next Steps**

* **Global Path Planning**: Integrate A\* or RRT\* for long-horizon navigation.
* **Dynamic Sectors**: Adapt sector sizes based on obstacle density.
* **Real-World Validation**: Test with physical drones using vision-based sensors.
* **Energy Optimization**: Incorporate energy-aware path planning.
* **Memory-Augmented RL**: Add LSTM for long-horizon tasks.