BALL BALANCING MAZE

TOPICS IN INTELLIGENT ROBOTICS
MSC. IN ARTIFICIAL INTELLIGENCE
2024/2025

GROUP P1A:
ADRIANO MACHADO - 202105352
FÉLIX MARTINS - 202108837
FRANCISCO DA ANA - 202108762
MARTIM IGLÉSIAS - 202005380

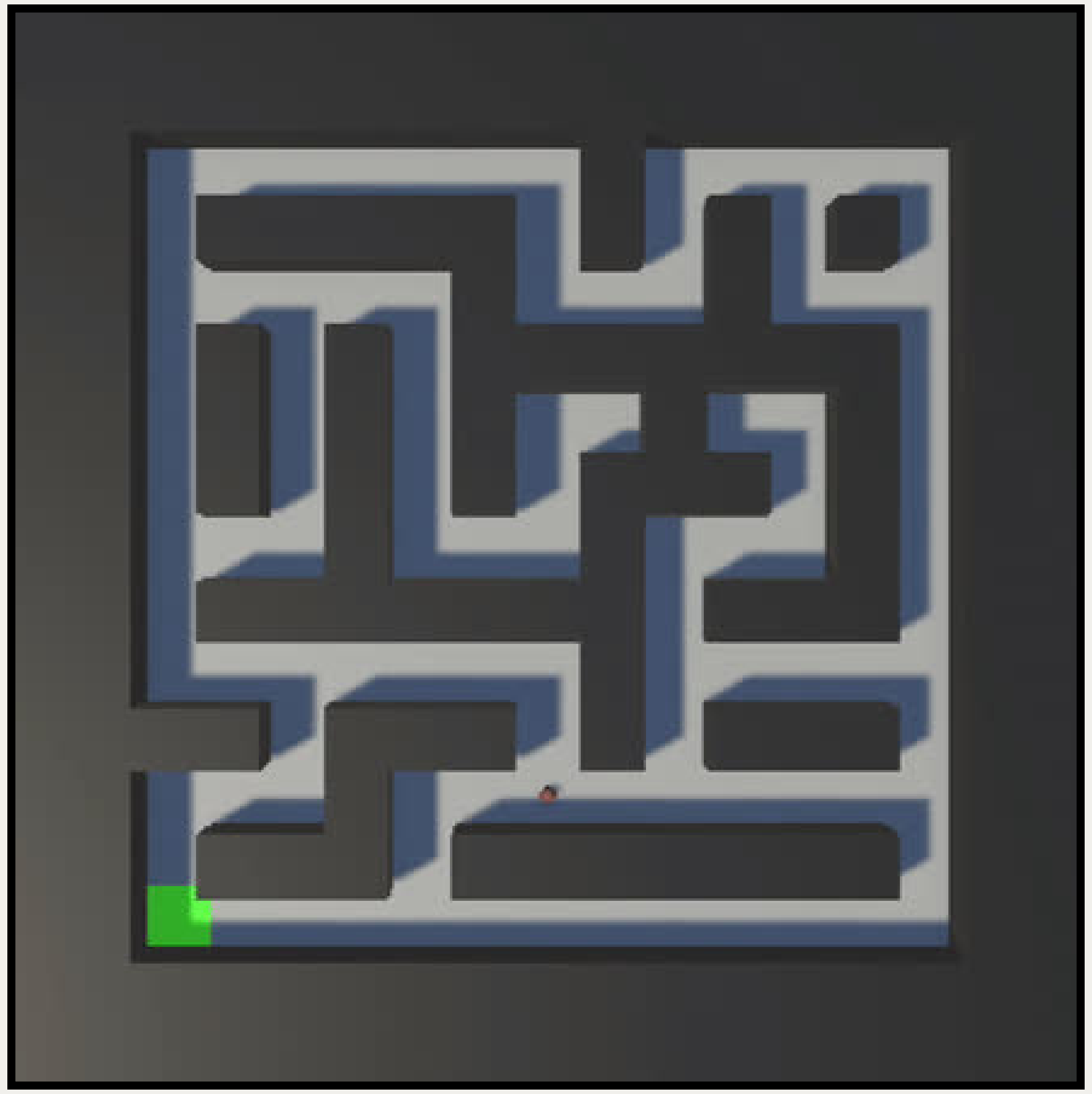
PROBLEM

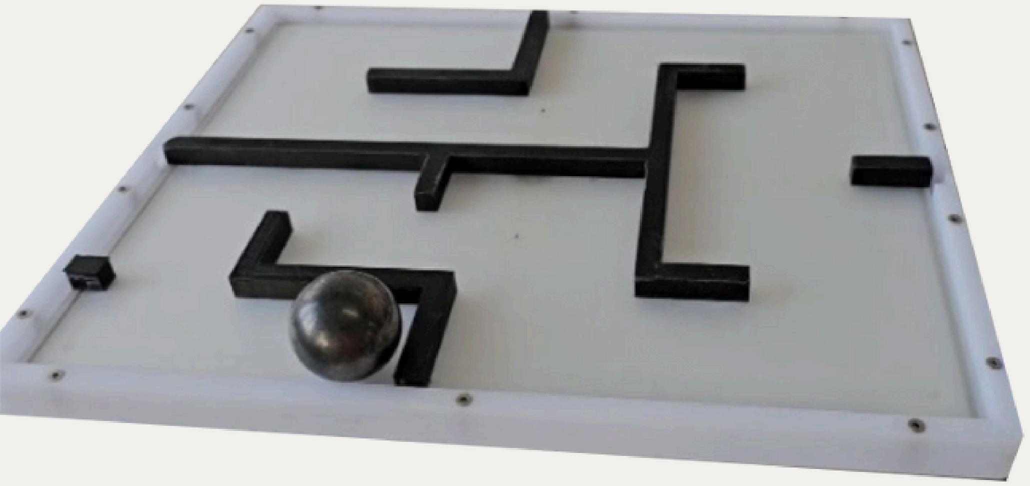
Main goal: solving a maze by moving a ball on top of a ball-balancing table

Technologies chosen for implementation:

- Unity for simulation environment
- Unity ML-Agents toolkit with Python API for RL model training

Key learning challenge: the goal is not just to solve one static maze, but to generalizes across various mazes



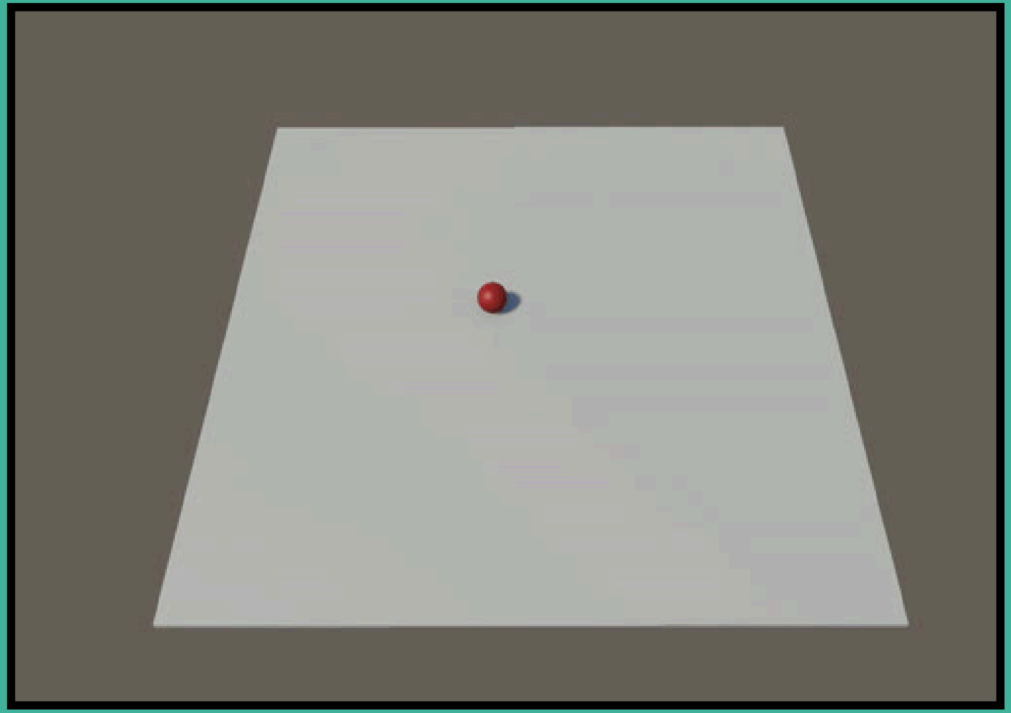


BASELINE TASKS

Objective: validate the agent's control of fundamental dynamics with simpler tasks before attempting the full maze.

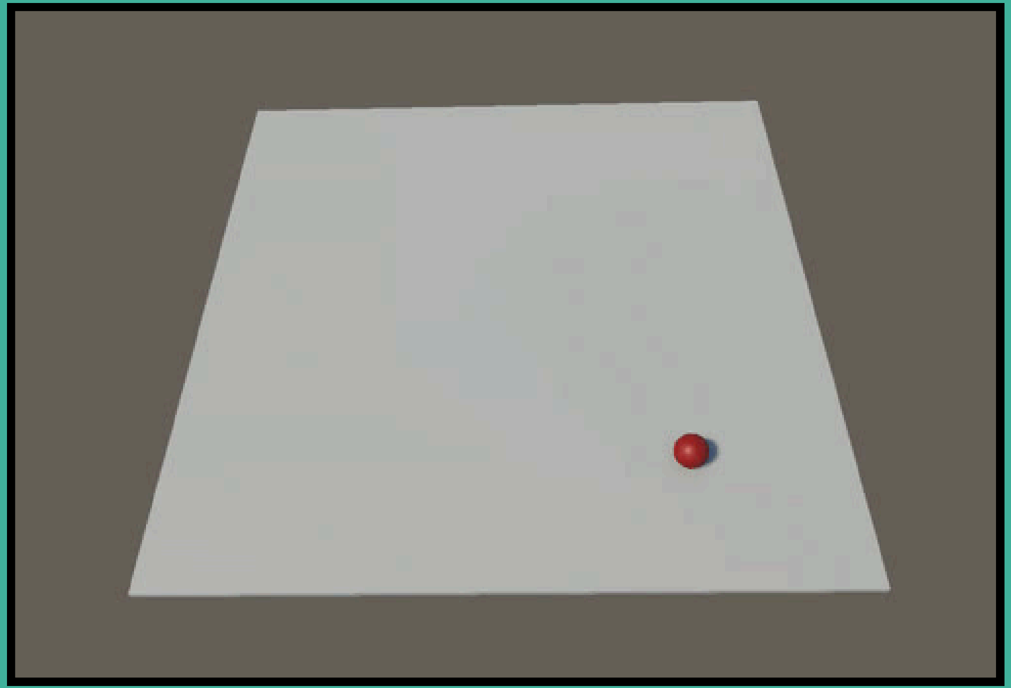
Task 1: Plate balancing

- **Goal:** Keep the ball on the plate without it falling off.
- **Reward signal:** small positive reward for each step the ball remained on the plate and a negative penalty if it fell.
- **Performance:** the agent showed exceptional performance, quickly learning a stabilization policy and achieving a near-perfect success rate.



Task 2: Balance to target

- **Goal:** Move the ball to a specific, randomly placed target coordinate on the plate and maintain its position.
- **Reward signal:** inverse of the distance to the target, encouraging the agent to get closer. A large penalty was given if the ball fell.
- **Performance:** the agent consistently learned to maneuver the ball to the target, confirming its ability to perform goal-directed navigation.



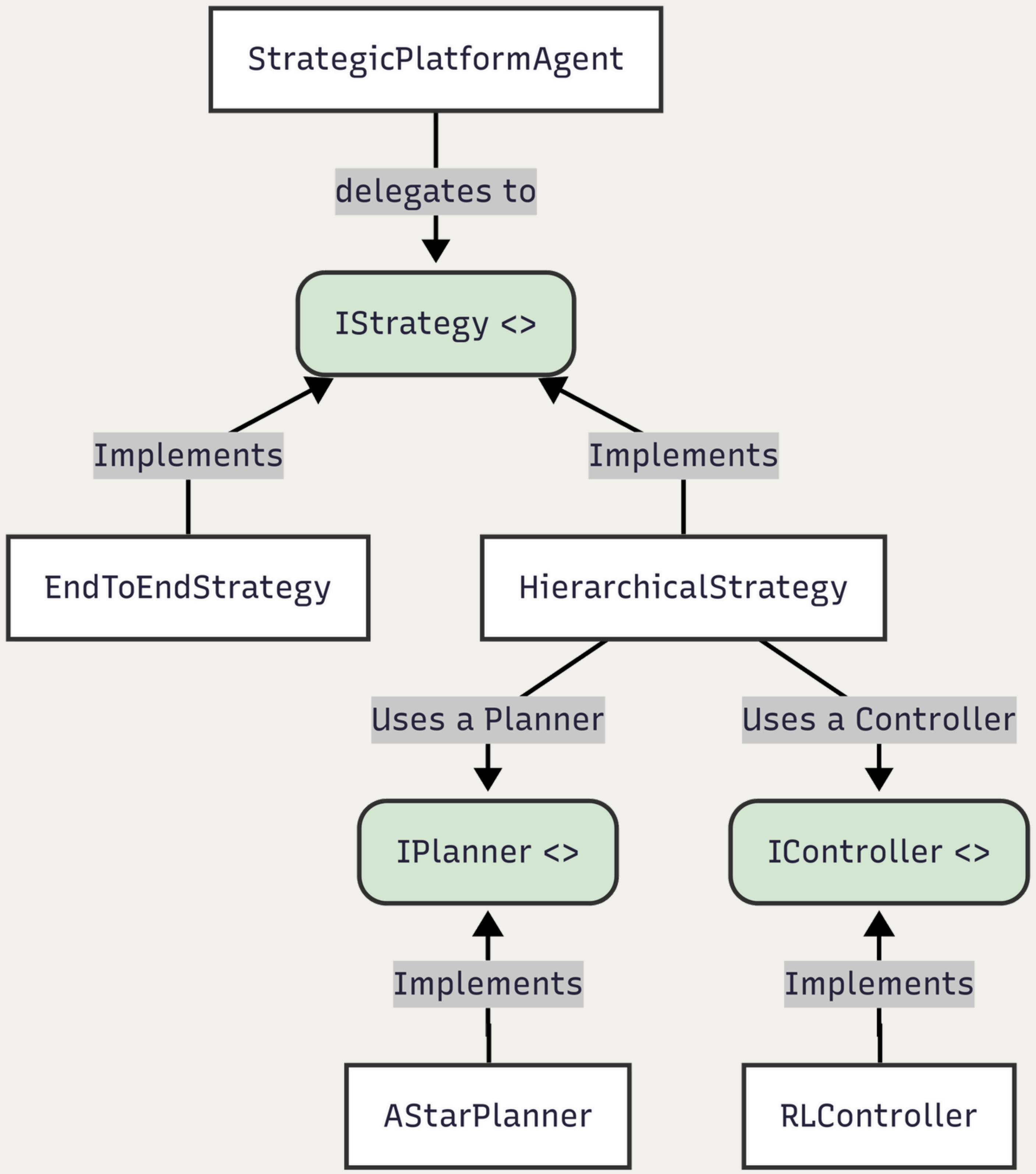
OUR SOLUTION

Strategy Pattern implementation

- Decouples agent from decision-making logic
- This modular design allows for different solution approaches to be seamlessly integrated and switched without altering the core agent's code.
- Interfaces *IStrategy*, *IPlanner* and *IController* ensure the design is reusable and easily extendable for testing new strategies.

Two main strategies implemented

- **Hierarchical:** Decomposes the problem into high-level pathfinding using the A∗ algorithm for pathfinding (*AStarPlanner*) and low-level control using a RL-based agent (*RLController*).
- **End-to-End:** Handles the entire maze navigation task with a single, monolithic policy, implemented in *EndToEndStrategy*



HIERARCHICAL CONTROL

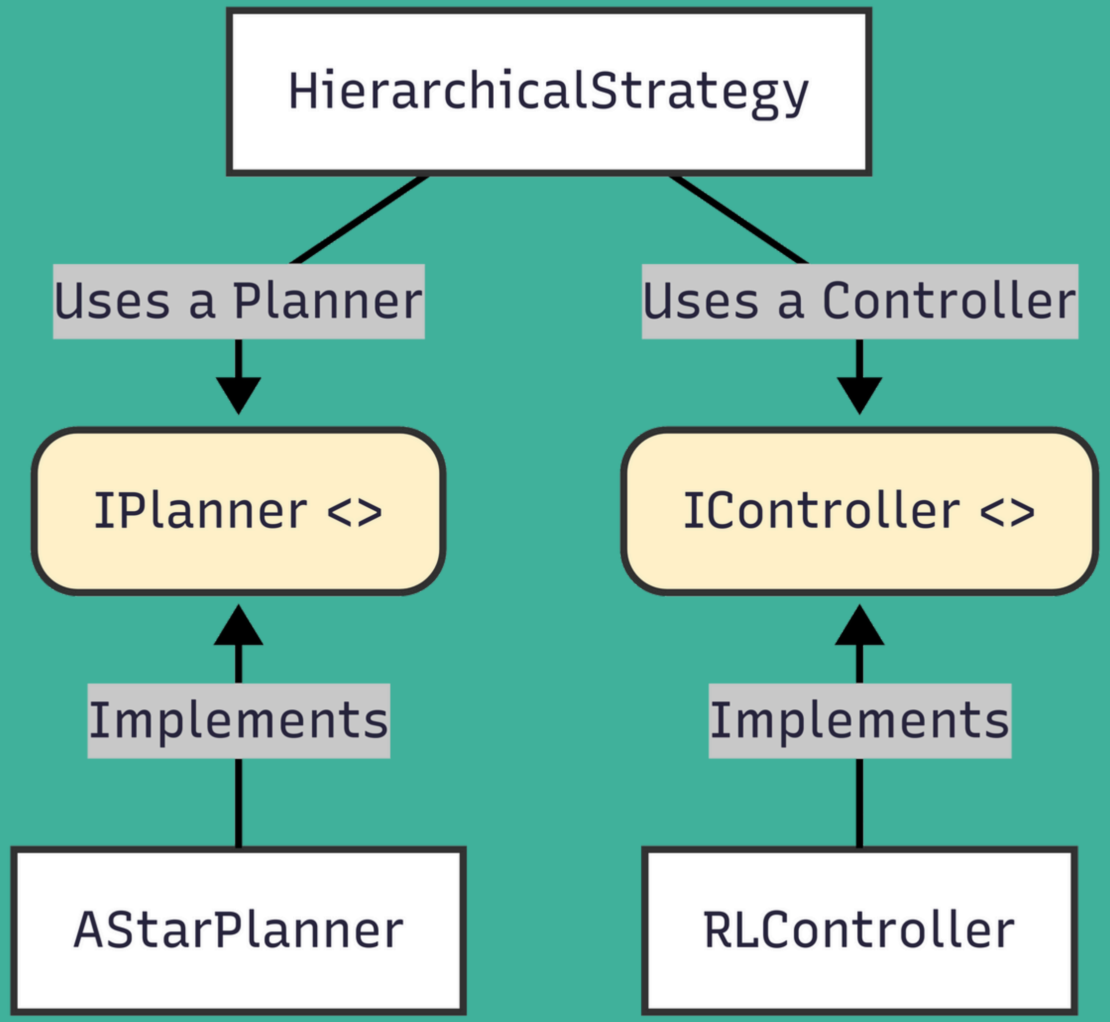
Problem Decomposition Philosophy: Complex maze navigation broken into two distinct, manageable and interconnected subproblems

High-level planner: A* algorithm

- Guarantees an optimal path using the discrete maze structure.
- Computes the entire sequence of waypoints before exeuction begins.
- Uses full knowledge of the maze for strategic planning.
- Provides a deterministic and interpretable navigation plan.

Low-level controller: PPO RL agent

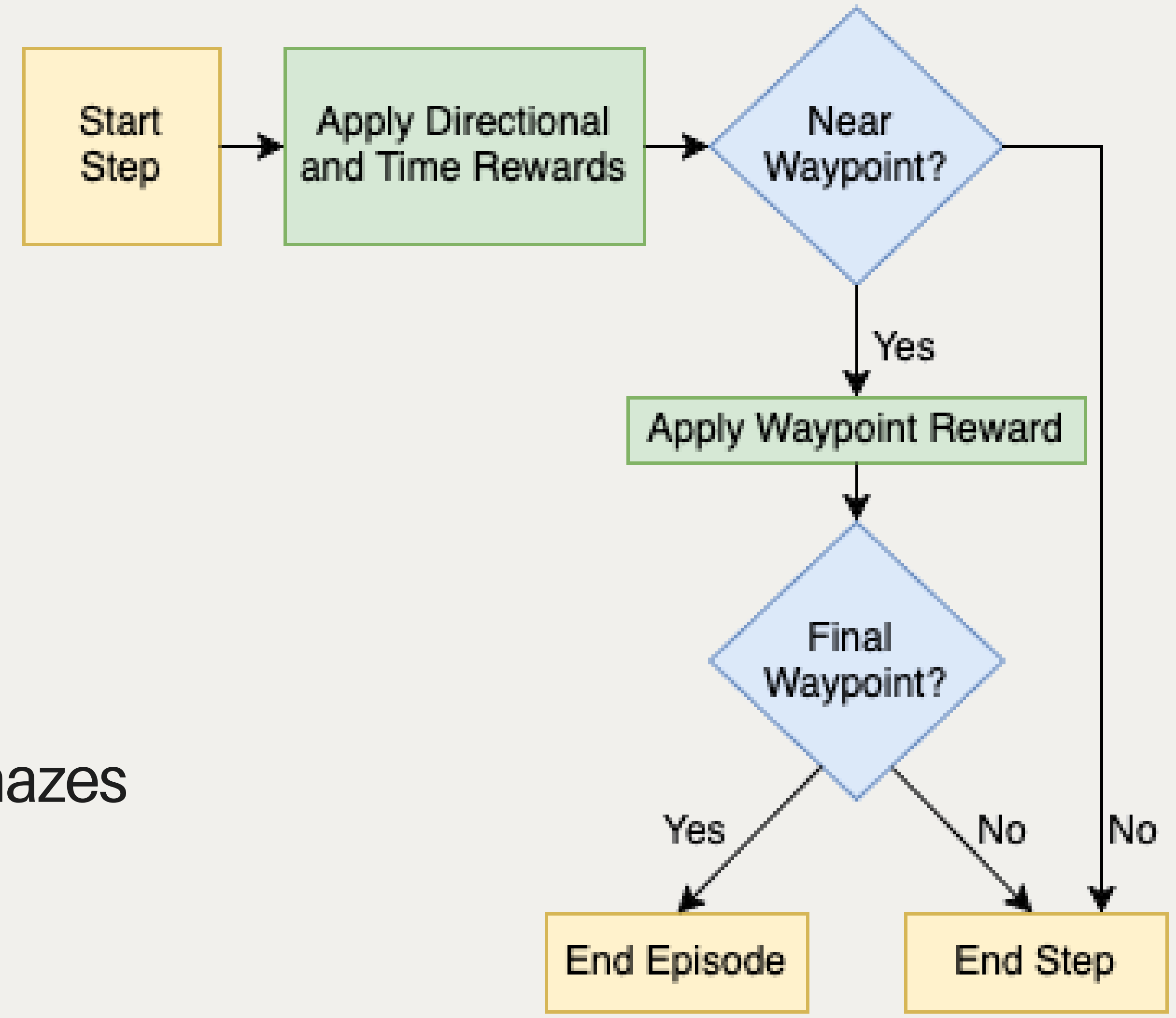
- Learns complex, physics-based ball manipulation via trial and error.
- Handles nonlinear dynamics and plate movement.
- Adapts to different starting positions and waypoint locations.
- Focuses only on point-to-point navigation, ignoring overall maze complexity.



REWARDS FOR THE HIERACHICAL AGENT

Multi-Component Reward Design:

- Success depends on carefully engineered reward function with three key components
- Composite formula: *Rtotal* = *Rdir* + *Rtime* + *Rwaypoint*
- Each component addresses different behavioral objectives for optimal learning



Directional Reward (*Rdir*): Movement Guidance

- Uses dot product of ball velocity and target direction (positive rewards for correct direction, negative for wrong direction)

Dynamic Time Penalty (*Rtime*): Speed Incentive

- Small negative reward at each timestep to encourage efficient navigation
- Adaptively scaled by maze complexity to ensure fair evaluation across different mazes

Waypoint Achievement Reward (*Rwaypoint*): Success Signal

- +1.0 reward for reaching each waypoint, 0.0 otherwise
- Provides strong learning signal for sub-goal completion

END-TO-END LEARNING

Single Policy Architecture:

- One unified RL agent handles entire maze navigation task autonomously
- More ambitious approach, undifferentiating high level plan and low-level control

Convolutional Neural Network (CNN): Spatial Vision

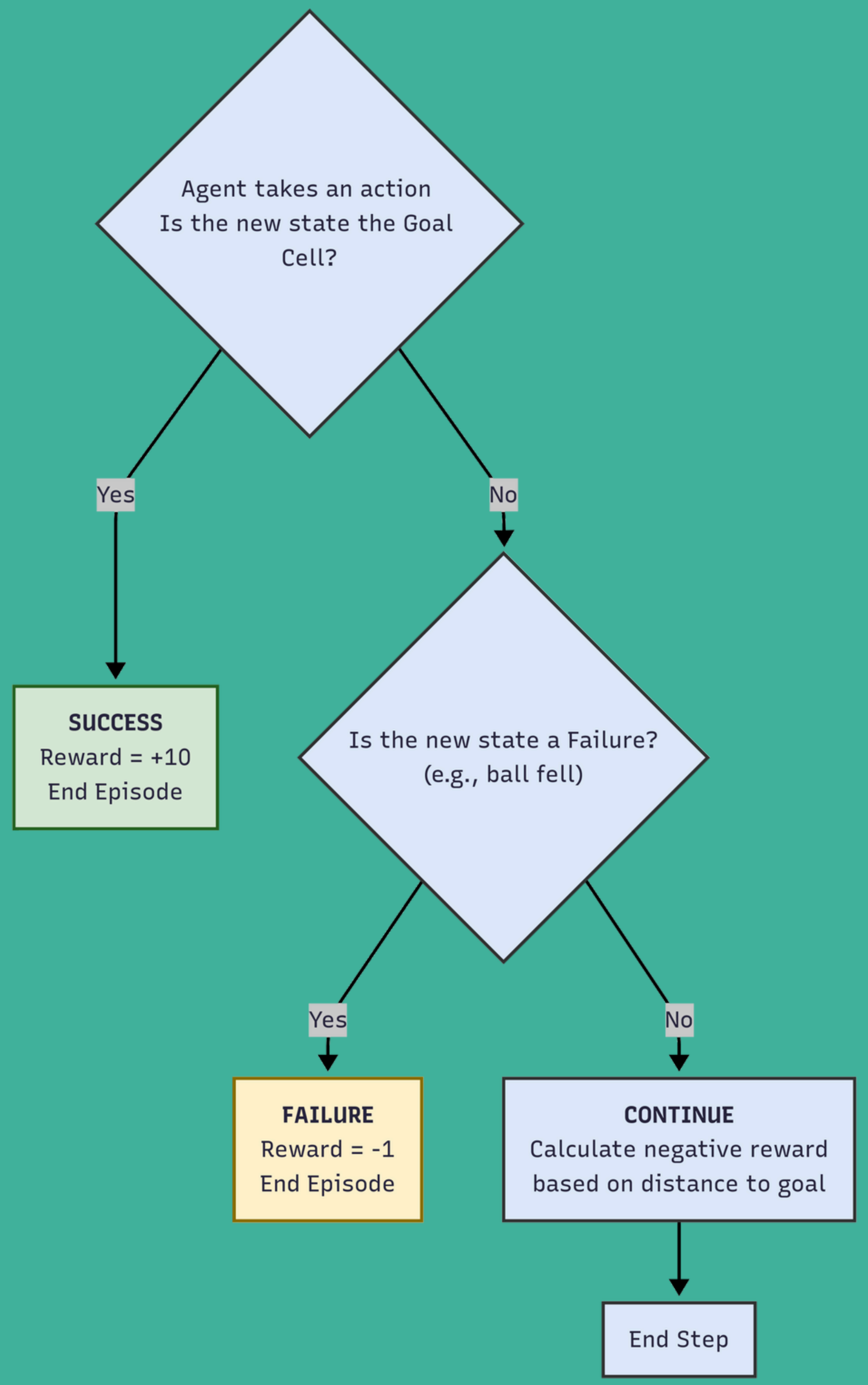
- Processes 20x20 grid of wall presence around the ball's current position
- Provides agent with local "maze vision" to understand navigable paths and extract spatial features for decision-making

Long Short-Term Memory (LSTM): Temporal Context

- Gives agent short-term memory to track movement history
- Enables understanding of sequential decisions and compensating for partial observability in complex maze environments

Complex Observation Space

- Requires agent to integrate multiple information sources for optimal navigation



RESULTS & COMPARATIVE ANALYSIS

- **Hierarchical approach:** Is 8.3x **more sample efficient** (6M vs 50M training steps), achieves 5.4% **higher success rate** and 4.6% **faster completion,** but requires domain knowledge (A∗ algorithm)

- **End-to-End:** Fully autonomous learning, but demands significantly more training

- Both approaches show consistent performance across different maze configurations

Metric	Hierarchical Strategy	End-to-End Strategy
Avg. Success Rate	85.8%	80.4%
Avg. Steps to Completion	992	1040
No. of Training Steps	6M	50M

FINDINGS & NEXT STEPS

Key Challenges:

- Complex Unity simulation setup and configuration
- Balancing physics fidelity with performance and ML-Agents integration

Conclusions:

- Developed modular, Strategy-Pattern framework for rapid prototyping and comparison
- Demonstrated effectiveness of hierarchical and end-to-end deep RL for nonlinear robotic control

Future Work:

- **Physical Integration:** Implement inverse kinematics for multi-axis robotic arm control
- **Real-World Sensing:** Deploy physical sensors for ball tracking and dynamic maze recognition
- **Enhanced Physics:** Transfer to high-fidelity simulators or use domain randomization
- **Hardware Validation:** Test policies on physical ball-balancing platforms