Workshop 3 — Machine Learning & Cybernetics Implementation

Juan Diego Grajales Castillo (20221020128)
Nicolas Castro Rivera (20221020055)
Systems Sciences Introduction — Semester 2025-I
Professor: Carlos Andrés Sierra, M.Sc.

1 Introduction

This third workshop builds upon the conceptual and analytical foundations established in Workshops 1 and 2. It aims to formalize the integration of Machine Learning (ML) and Cybernetic Control principles into a simulated predator-prey system composed of two adaptive agents: one predator and one prey. These agents interact within a grid-based environment governed by discrete-time dynamics, proximity-limited perception, and a competitive objective structure.

The primary goal is to design a self-regulating system in which the agents, through reinforcement learning, develop intelligent behaviors such as pursuit, evasion, and obstacle navigation. Rather than relying on fixed rules, agents are trained using learning algorithms that allow them to optimize their decisions over time through feedback and experience. At the same time, cybernetic feedback loops ensure that behaviors remain responsive, stable, and context-sensitive, even in complex or partially observable conditions.

The document presents a comprehensive theoretical design including algorithm selection, agent architecture, feedback loop modeling, dynamic interactions, and evaluation methodology. Emphasis is placed on robustness, adaptability, and emergent complexity. Implementation details are deferred to later stages, prioritizing system design and behavior modeling over low-level programming tools. The outcome is a simulation-ready architecture capable of producing realistic, evolving behaviors in a minimal-agent setting.

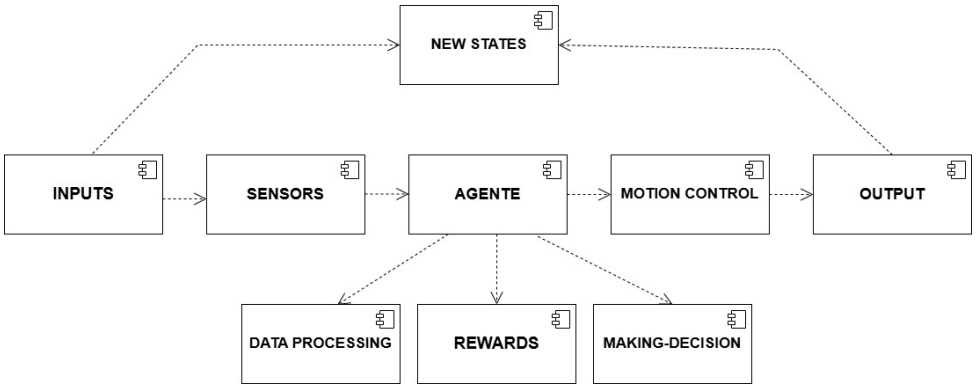

Figure 1: Diagram of system components.

2 Machine Learning Integration

The predator-prey system represents a dynamic, partially observable, and competitive environment. Machine Learning (ML), particularly Reinforcement Learning (RL), provides the theoretical and computational tools for agents to learn optimal behaviors from experience. This section outlines the rationale behind algorithm selection and the conceptual steps required to integrate RL into the system's architecture.

2.1 Algorithm Selection and Justification

Considering the discrete grid-based nature of the environment, limited perception range, and opposing goals of predator and prey, the competition and the simultaneous moves several RL algorithms were evaluated:

- **IQL:** Each agent learns its own policy using Q-Learning, unaware that the others are also learning. Can be useful in discrete systems like the Prey-Predator, but may not converge in dynamical systems.
- **Self-Play:** Use DQN and others algorithms for trains agents by letting them play against versions of themselves, improving through competition. It's ideal for competitive multi-agent scenarios like predator-prey, where agents adapt to increasingly better opponents.
- **MADDPG (Multi-Agent Deep Deterministic Policy Gradient):** Although our system has only two agents, MADDPG is particularly suited due to its centralized training with decentralized execution. Each agent learns a deterministic policy (actor) while relying on a centralized critic that uses full state information during training. This structure is especially advantageous in adversarial setups where coordination is implicit in competitive behavior.

Given the complexity and adaptiveness required for both predator and prey, we propose using MADDPG to model each agent's behavior. The shared critic facilitates stabil-

ity during learning, while decentralized execution allows realistic deployment with local sensor inputs.

2.2 Conceptual Integration into System Architecture

To embed ML into the existing predator-prey simulation, we must refactor the architecture to support interaction-based learning. The integration involves several key components:

1. **Observation Spaces:** Define a fixed-radius sensory window (e.g., 5×5 grid) centered on each agent. This localized perception limits agent awareness and makes the state space partially observable, encouraging exploration and adaptive strategies.

2. **Action Spaces:** Each agent has a discrete set of possible actions (move up, down, left, right). These are mapped directly to changes in position on the grid. MADDPG will approximate continuous policy outputs, which will then be discretized if needed.

3. **State Representation:** Each agent encodes the environment as a state vector including:

   - Relative positions of nearby entities (e.g., predator, prey, traps, walls)
   - Velocity or recent movement direction (for inertia modeling)
   - Time step (to enable time-aware policies such as prey evasion learning)

4. **Reward Functions:** The reward structure is simplified and aligned with agent objectives:

   - **Predator:** +10 for capturing the prey; 0.1 per time step without capture.
   - **Prey:** +0.1 per time step evading the predator; 5 if captured.

   These sparse but goal-aligned rewards incentivize long-term strategic planning.

5. **Learning Pipeline:** During training, the agents interact with the environment in episodes. Each step generates tuples (s_t, a_t, r_t, s_{t+1}) stored in a replay buffer. Batches of these experiences are sampled to update actor and critic networks using gradient descent. Target networks and soft updates are used to ensure stable learning, as established in the DDPG framework.

6. **Centralized Critic Design:** The critic has access to the full environment state, including both agents' positions and actions. This enables learning of interdependent value functions, essential in competitive tasks. During execution, only the actor (policy network) is used, operating from local observations.

2.3 Advantages of the Proposed Approach

The use of MADDPG for this two-agent adversarial system offers several benefits:

- **Decentralized Execution:** Agents rely solely on local information at test time, mimicking realistic perceptual constraints.
- **Centralized Stability:** The centralized critic mitigates non-stationarity caused by simultaneous learning, improving convergence.
- **Behavioral Generalization:** Neural networks enable agents to learn abstract strategies, such as anticipating movement or avoiding corners.
- **Scalability:** Although designed for two agents, the framework can be extended to support additional entities (e.g., multiple traps or hazards) with minimal architectural changes.

This integration blueprint ensures that agents evolve through interaction, balancing reactive instincts with long-term planning. The architecture supports experimentation with different policies, perception models, and training scenarios, making it suitable for a range of simulation-based research objectives.

3 Cybernetic Feedback Mechanisms

Table 1: Discrete motion equations with detailed interpretations.

Equation	Detailed Explanation
$\pi_p(t+1) = \pi_p(t) + \Delta t\, r_p(t)\, v_p(t)$	Updates predator position by adding the product of time-step Δt, predator speed $v_p(t)$, and movement direction $r_p(t)$ (a unit vector) to the current position $\pi_p(t)$.
$\pi_h(t+1) = \pi_h(t) + \Delta t\, r_h(t)\, u_h(t)$	Updates prey position using analogous terms: current position $\pi_h(t)$, time-step Δt, movement direction $r_h(t)$, and speed $u_h(t)$.
$v_p(t) = \nu_{\max} + \dfrac{\alpha}{1 + e^{-kd(t)}}$	Models predator speed adaptation: base speed $\nu_{\max}$ plus a logistic function scaled by α, where k controls sensitivity to prey distance $d(t)$, resulting in acceleration when the prey is near.
$P_{\text{evade}}(t) = P_{\text{base}}\left(1 - e^{-\beta t}\right)$	Defines probability of successful evasion for the prey: starts at zero and asymptotically approaches P_{base}, with learning rate governed by β, simulating experience-based evasion improvement.

Cybernetics provides a foundational framework for modeling systems that regulate themselves through closed feedback loops. In our predator-prey system, cybernetic mechanisms serve as the internal control structures that guide each agent's decision-making process based on dynamic interactions with the environment.

These mechanisms do not replace reinforcement learning but rather complement it by embedding regulation, stability, and adaptive control principles into the behavior architecture. The agents sense their environment, evaluate internal state, and actuate responses accordingly, forming continuous perception-action cycles.

3.1 Feedback Loop Structure

Each agent operates under a closed-loop feedback system composed of the following components:

- **Sensor Input (S_t):** Real-time environmental observations (e.g., positions of obstacles, traps, predator/prey) captured within a finite sensory range.
- **Internal State (I_t):** Encodes variables like velocity, last movement, decision confidence, and survival time.
- **Control Logic (C):** A learned or predefined mechanism (e.g., neural network policy) that processes input and state to determine an action.
- **Actuator Output (A_t):** Concrete movement decisions (e.g., direction change), modifying the agent's position in the environment.
- **Feedback Update (F_t):** The new environmental and internal states after the action, which then serve as input for the next time step.

This loop repeats continuously during simulation, providing a robust structure for responsive, adaptive behavior.

3.2 Predator Feedback Loop

The predator's goal is to minimize the time required to capture the prey. Its feedback loop incorporates proximity-based urgency and environmental awareness.

- **Sensory Perception:** The predator's sensors detect the relative position of the prey, nearby traps, and walls within a 5×5 local grid.
- **Speed Regulation:** Predator manege a constant speed if doesn't see the prey directly and increases the speed if detects it with sensor $d(t)$:

  where v_{base} is the minimum speed, α controls the acceleration range, and k determines sensitivity to distance.
- **Movement Direction:** The predator selects a direction using a policy trained to maximize capture probability, avoiding traps when needed.

3.3 Aroma Matrix as an Environmental Cue

To further assist the predator in tracking the prey in a biologically inspired manner, we introduce an **aroma matrix**—a dynamic, two-dimensional grid that stores the decaying scent trail left behind by the prey over time.

Each cell in the environment holds a scalar value $A_{i,j}(t)$ representing the current intensity of the prey's scent at that location (i, j) and time t. The scent evolves based on the following update rule:

$$A_{i,j}(t+1) = \begin{cases} \gamma, & \text{if prey is at position } (i,j) \\ \lambda \cdot A_{i,j}(t), & \text{otherwise} \end{cases}$$

where:

- γ is the maximum scent intensity deposited by the prey when it visits a cell.
- $\lambda \in (0, 1)$ is a decay factor simulating the evaporation of scent over time.

The predator uses the aroma matrix as an additional sensory layer, perceiving nearby scent gradients within its observation radius. This mechanism supports:

- **Direction inference:** The predator can follow increasing gradients of scent to approximate the prey's path.
- **Search reinforcement:** In the absence of visual contact, the predator can continue pursuit based on environmental clues.
- **Behavior modulation:** The aroma signal is combined with other stimuli (e.g., direct line-of-sight) to prioritize hunting strategies.

This enhancement transforms the environment into a stateful participant in the simulation, improving realism and enabling more nuanced predator behavior, especially when direct prey detection is not possible.

This feedback loop allows the predator to transition between search and chase modes depending on real-time inputs.

3.4 Predator Motion Dynamics

The predator's movement over time can be formalized using a discrete-time dynamic equation derived in Workshop 2:

$$\pi_p(t+1) = \pi_p(t) + \Delta t \, r_p(t) \, v_p(t)$$

where:

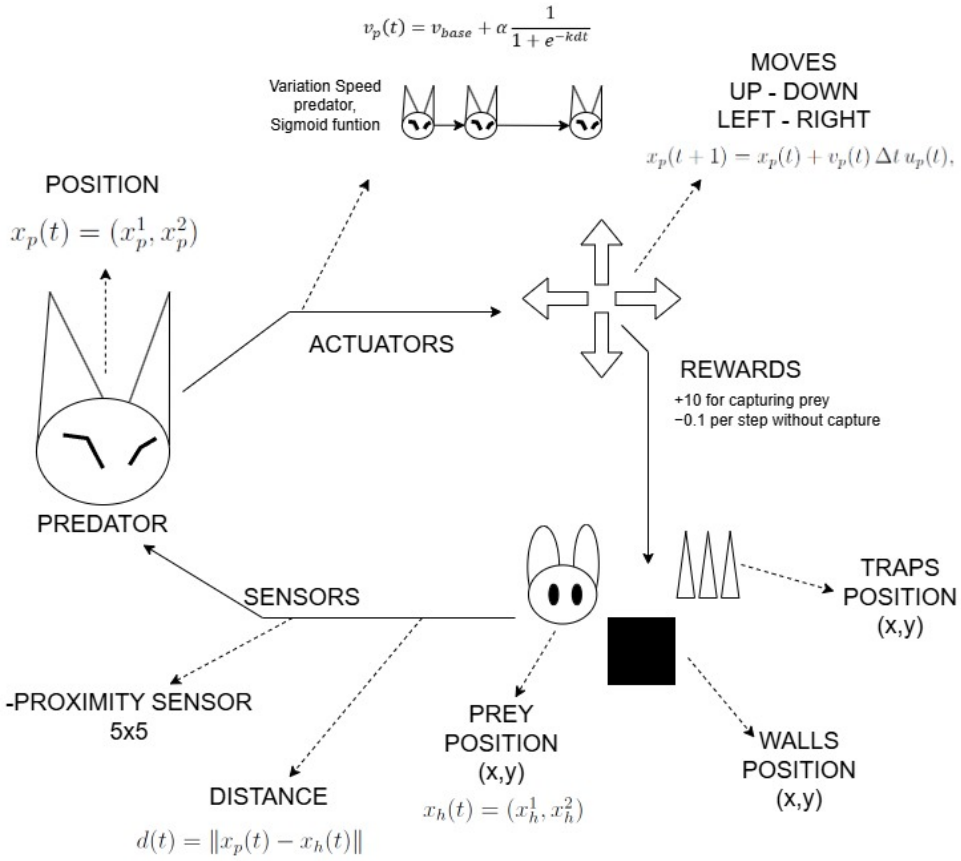

Figure 2: Phase Portrait of the predator agent.

- $\pi_p(t)$ represents the predator's position vector at time step t, typically in (x, y) coordinates within the grid.
- Δt is the discrete time step interval used in the simulation. In most cases, $\Delta t = 1$ to match the tick rate of the environment, but it is explicitly included to generalize the model for different time scales.
- $r_p(t)$ is a unit vector indicating the direction of the predator's intended movement at time t. It is determined by the output of the control logic (e.g., learned policy or sensor-based decision).
- $v_p(t)$ is a scalar representing the predator's velocity at time t, which may be modulated dynamically through cybernetic feedback (e.g., logistic function based on prey proximity).

This equation defines how the predator updates its position based on its current location, directional intent, and speed. Importantly, both $r_p(t)$ and $v_p(t)$ may be influenced by internal states and external stimuli, such as proximity to prey, obstacles, or scent gradients.

By embedding this motion model into the cybernetic feedback loop, the predator is not only reactive but also responsive to varying spatial conditions and learning-derived

strategies. It also allows for smooth integration into reinforcement learning frameworks, where both direction and velocity can be outputs of the policy network.

3.5 Prey Feedback Loop

The prey's objective is to maximize evasion time while minimizing the risk of entrapment. Its feedback loop prioritizes safety, exploration, and learning from threats.

- **Sensory Perception:** The prey senses the predator's proximity, traps, and map boundaries within its perception grid.
- **Evasion Learning:** A time-dependent evasion probability function reflects accumulated experience:

  $$P_{\text{evade}}(t) = P_0 \cdot (1 - e^{-\beta t})$$

  where P_0 is the asymptotic evasion skill and β controls the learning rate.
- **Decision Modulation:** If the predator is near, evasive actions are prioritized. If not, the prey may explore or rest. Its movement direction is influenced by prior successes (e.g., successful escapes).

This dynamic model encapsulates the prey's responsive nature: as the environment changes (e.g., predator proximity or trap detection), so does $r_h(t)$ and potentially $u_h(t)$, allowing the prey to adapt its motion accordingly.

By integrating this model into the cybernetic feedback framework, the prey becomes capable of producing nuanced, situational behaviors. These may include accelerating when cornered, slowing to reduce noise or trace (in case of scent-based tracking), or optimizing paths to safe zones. This reinforces the system's realism and enhances the prey's role as a learning, adaptive agent.

This feedback mechanism enables the prey to adapt over time and develop robust survival strategies based on contextual cues. In phase portrait of the prey, is visible the inputs that can affect it movement, prey have a constant speed, but the direction can vary.

3.6 Prey Motion Dynamics

The movement of the prey agent is governed by a similar discrete-time equation, capturing its evolution over time in the environment:

$$\pi_h(t+1) = \pi_h(t) + \Delta t \, r_h(t) \, u_h(t)$$

where:

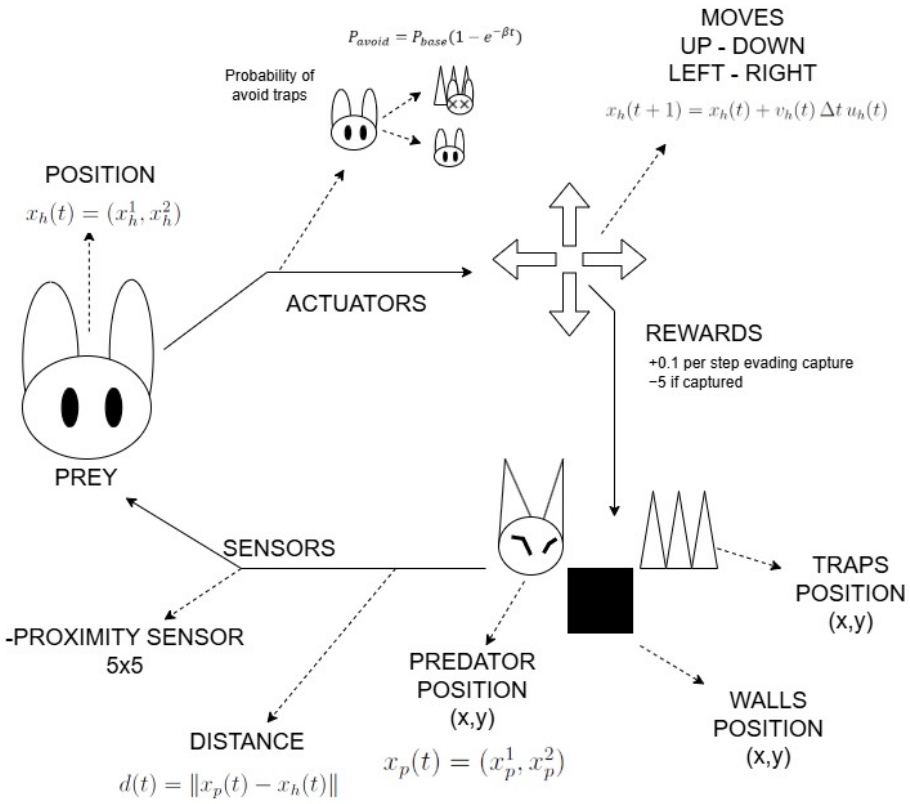

Figure 3: Phase Portrait of the prey agent.

- $\pi_h(t)$ is the position vector of the prey at time t, representing its spatial coordinates on the grid.
- Δt denotes the simulation time step. As with the predator model, this factor can be set to 1 for simplicity but retains generality for temporal scaling.
- $r_h(t)$ is a unit vector indicating the direction chosen by the prey at time t, based on its current perception and decision logic. This direction may result from evasive strategy, exploration behavior, or a learned policy output.
- $u_h(t)$ is a scalar representing the prey's instantaneous speed. Unlike the predator's speed, which may be driven by aggression, the prey's velocity can vary based on proximity to threats, internal evasion functions, or energy conservation strategies.

3.7 Integration with Reinforcement Learning

Cybernetic feedback mechanisms are not replacements for learning but operate synergistically with RL policies:

- Inputs from sensors and internal states are used to construct the observation vector for each agent's policy network.

- Feedback-based rules (e.g., urgency modulation, trap avoidance) can be embedded as prior knowledge or encoded via reward shaping.
- Control outputs from the policy network become the actuator commands within the cybernetic loop.

This structure creates a self-regulating, adaptive system in which agents learn not only to react, but to stabilize and optimize their behaviors within a dynamic, uncertain environment.

4 Agent Interaction Dynamics

The interaction between the predator and the prey forms the core dynamic of the system, constituting a closed, two-agent adversarial loop. Unlike systems involving populations or cooperative swarms, this environment isolates the interaction to a single predator and a single prey agent, creating a tightly coupled competitive setting. Each agent is independently trained but operates under mutual influence, giving rise to co-evolutionary behavior patterns.

4.1 Asymmetric Objectives and Learning Goals

The agents pursue fundamentally opposing objectives, which directly shape their learning processes:

- **Predator:** Learns to capture the prey as quickly and efficiently as possible. It must navigate around environmental constraints such as walls and traps while minimizing pursuit time and maximizing the probability of successful interception.
- **Prey:** Learns to evade the predator for as long as possible. Its training focuses on survival tactics—such as obstacle use, trap avoidance, path randomization, and timing escapes—that make capture difficult even for an optimized predator.

Both agents use reinforcement learning policies to maximize their own expected returns, based on reward functions defined in terms of capture or evasion. These asymmetric roles simulate predator-prey dynamics observed in natural systems, providing ecological realism and a fertile ground for emergent behavior.

4.2 Training Paradigm

Although there is only one predator and one prey, their policies are developed in parallel using a competitive learning paradigm. We use a form of **self-play**, where:

- The predator is trained against increasingly evasive prey strategies.

- The prey is trained against increasingly efficient predator pursuit patterns.

This iterative training loop forces both agents to adapt continually, avoiding static or exploitable strategies. As a result, the agents evolve behaviors that are not explicitly programmed but instead emerge from the need to outperform the opponent.

4.3 Information Constraints and Observability

To maintain realism and avoid overfitting to perfect information:

- Agents are restricted to a localized observation radius (e.g., 5 $times$5 grid cells).
- Each agent's policy network receives only the partial state observable from its position.
- No direct communication exists between agents; all coordination or response is emergent from learned behavior.

This partial observability introduces uncertainty and compels agents to learn predictive and probabilistic strategies rather than reactive or deterministic ones.

4.4 Interaction Outcomes and Metrics

The competition between predator and prey results in diverse outcomes based on starting positions, learned policies, and map complexity. These outcomes are evaluated based on:

- **Time to Capture:** Number of steps taken by the predator to successfully intercept the prey.
- **Survival Duration:** Time survived by the prey before being captured or reaching maximum episode length.
- **Trap Utilization:** Whether traps are successfully used by either agent as part of their strategy.
- **Path Diversity:** Degree to which agents vary their movement across episodes, indicating adaptability.

4.5 Emergent Behaviors and Adaptive Complexity

One of the strengths of the proposed architecture is its capacity to yield emergent behaviors, such as:

- **Ambushes:** The predator learns to anticipate the prey's trajectory and intercept at choke points.

- **Escape Loops:** The prey uses walls or traps to trap the predator in suboptimal paths and escape.
- **Learning to Lure:** The prey may exploit predator speed or aggression to guide it into traps.

These behaviors are not explicitly coded, but arise as side effects of reward maximization, proving the effectiveness of reinforcement learning combined with cybernetic feedback in shaping complex strategies from simple rules.

5 Evaluation Strategy

Evaluating a predator-prey simulation driven by reinforcement learning and cybernetic control requires a multidimensional approach. The system must be assessed not only in terms of agent performance (i.e., success in capturing or evading) but also with regard to stability, learning progression, and emergent behavior. This section presents a structured evaluation framework incorporating both quantitative metrics and qualitative insights.

5.1 Quantitative Metrics

To objectively measure system performance and agent improvement over time, the following metrics will be employed:

- **Episode Reward:** Total cumulative reward received by each agent during an episode. This metric reflects how well the agent's actions align with its learning objective (e.g., capture for predator, survival for prey).
- **Time to Capture / Survival Duration:** Measures the number of steps before the prey is captured or until the maximum episode length is reached. This provides a direct indicator of predator efficiency and prey evasiveness.
- **Success Rate:** Ratio of successful captures by the predator versus total episodes. For the prey, it corresponds to episodes survived without being captured.
- **Learning Curve Stability:** Graphs of average reward over training episodes. A stable curve with upward trends indicates successful policy optimization, while volatility may signal instability or inadequate exploration.
- **Exploration Entropy:** Degree of diversity in state-action choices over time. This ensures agents avoid converging prematurely to suboptimal deterministic behaviors.

These metrics will be collected across multiple training runs with varied initial conditions and random seeds to ensure robustness and generalizability of results.

5.2 Qualitative Analysis

In addition to numerical metrics, it is critical to interpret behaviors qualitatively to identify emergent phenomena or design flaws:

- **Emergent Behaviors:** Observation of unprogrammed but intelligent actions such as path prediction, trap usage, or luring techniques. These signal that agents have learned strategies beyond basic rule-following.
- **Failure Modes:** Analysis of repeated failure patterns (e.g., prey cornering itself, predator looping inefficiently) can reveal weaknesses in policy architecture, reward shaping, or environment structure.
- **Adaptation to Environment:** Evaluation of how well agents adjust to new map configurations, changes in trap distribution, or altered reward magnitudes.
- **Behavioral Smoothness:** Visually inspecting whether agent movement appears erratic or fluid. Smoothness often correlates with stable, confident policy convergence.

Episodes of interest will be recorded and annotated to illustrate key behaviors, which will be useful for both debugging and presentation of results.

5.3 Stability and Convergence Analysis

From a systems theory perspective, it is essential to analyze the long-term behavior of agent interactions. We propose:

- **Phase Space Trajectories:** Plotting agent positions over time in a 2D grid to identify convergence patterns, loops, or chaotic behavior.
- **Equilibrium Analysis:** Identifying scenarios where agent behavior stabilizes (e.g., predator consistently captures within fixed steps), or where unstable dynamics emerge (e.g., prey exploits a loophole to never be caught).
- **Bifurcation Behavior:** Testing how small changes in initial conditions (e.g., agent spawn distance) affect outcomes—useful for diagnosing sensitivity and resilience of learned strategies.
- **Regulation Oscillations:** Analyzing how cybernetic feedback parameters (e.g., speed scaling, evasion probability) affect the system's convergence to stable behaviors or divergence into oscillatory or chaotic dynamics.

In phase portrait of predator is visible how the different inputs can affect the decision that predator will take, it can move in different directions and in different speed depending of the perception that you have obtained from the environment.

In phase portrait of prey is visible how can affect the different inputs to the decision of the direction, it has constant speed, but the direction will change with the information received.

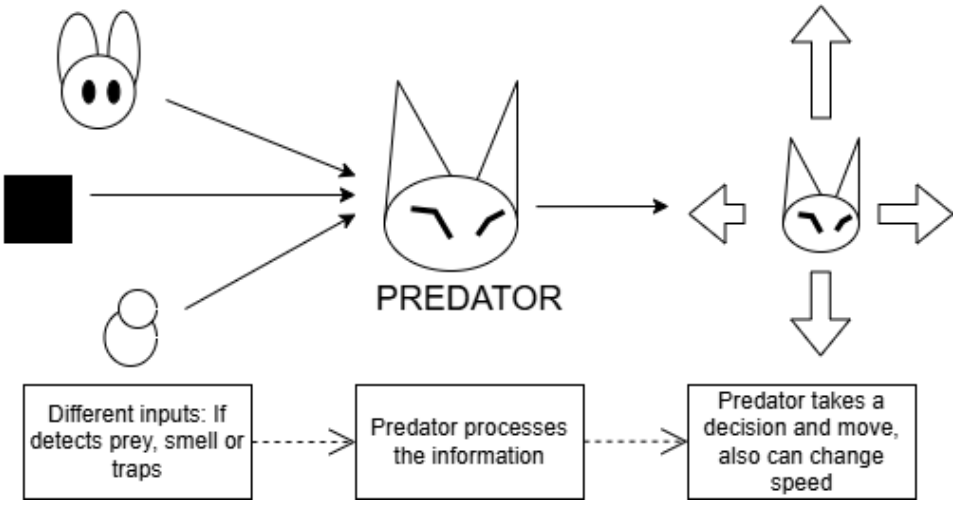

Figure 4: Phase Portrait of the predator agent.

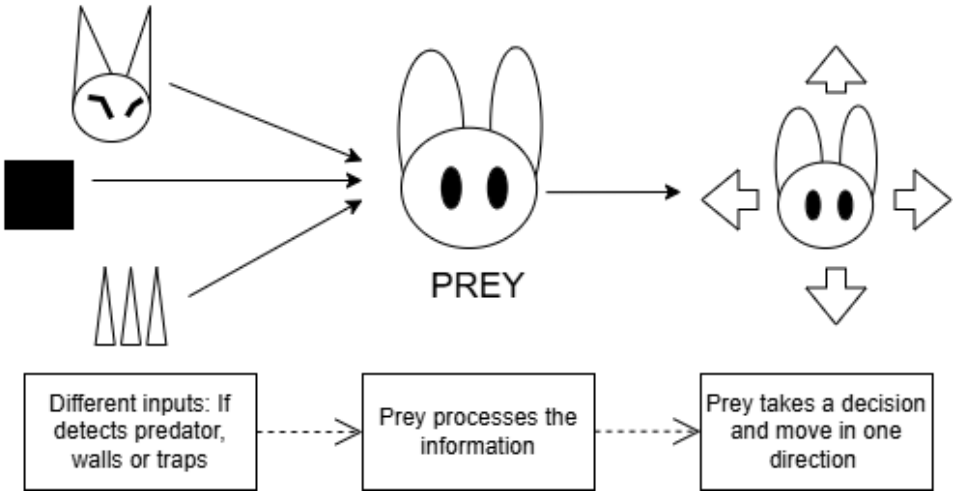

Figure 5: Phase Portrait of the prey agent.

5.4 Generalization Testing

To ensure the agent policies do not overfit to specific maps or conditions:

- **Unseen Maps:** Evaluate agent performance on grid layouts not used during training.
- **Dynamic Obstacles:** Introduce moving traps or shifting terrain to test policy flexibility.

- **Noise Injection:** Add observational or action noise to measure policy robustness under uncertainty.

5.5 Evaluation Summary

This multifaceted evaluation strategy—combining learning metrics, system-level analysis, and emergent behavior observation—ensures a comprehensive understanding of agent performance, system adaptability, and the effectiveness of learning and feedback mechanisms. It also provides a foundation for iterative refinement and supports the eventual goal of producing stable, realistic, and intelligent autonomous behavior.

6 Conclusion

This workshop presents a complete theoretical framework for the design of an adaptive, cybernetic predator-prey simulation using modern reinforcement learning techniques. By focusing on a two-agent competitive scenario—composed of a single predator and a single prey—the system isolates core behavioral dynamics and enables deep analysis of agent interaction, learning progression, and stability.

The use of MADDPG (Multi-Agent Deep Deterministic Policy Gradient) is justified by its ability to support decentralized decision-making and centralized training, making it well-suited for adversarial environments. Cybernetic feedback mechanisms further enrich the agents' behavior by embedding context-aware regulation and real-time adaptation, ensuring that learning is not only effective but also interpretable and stable.

A multidimensional evaluation strategy has been proposed to validate agent performance, system convergence, and the emergence of complex strategies. This includes quantitative metrics such as reward accumulation and survival time, as well as qualitative observations of behavioral smoothness and adaptability. The addition of phase space analysis and sensitivity testing aligns the system with principles of dynamical systems and control theory.

Ultimately, the design developed in this workshop lays a strong foundation for future implementation and experimentation. It provides both the structural clarity and the theoretical depth required to produce intelligent, evolving agents capable of simulating realistic predator-prey dynamics in a controlled, research-oriented simulation environment.

References

1. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O., & Mordatch, I. (2017). Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Advances in Neural Information Processing Systems, 30.*

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

3. Mnih, V., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.

4. Foerster, J., et al. (2017). Stabilising experience replay for deep multi-agent reinforcement learning. *Proceedings of ICML*.