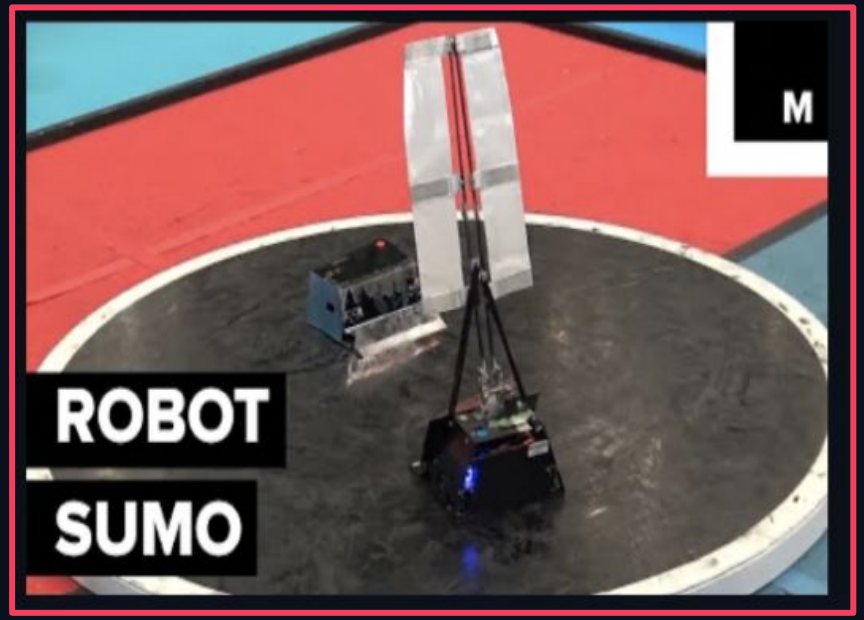
Robot Sumo Simulation

Build AI Agents that Battle, Adapt, and Work as a Team

Team: Adebare Kofo-Abayomi, Avlynn Wade, Faysal Zaki, Joshua Oziel Fernández Alvarado

Client & Faculty Advisor
Department of Electrical Engineering and Computer Science
Dr. Hanks



Goal and Motivation

Overall Goal: make Robot Sumo more accessible while creating a platform for robotics and reinforcement-learning experimentation.

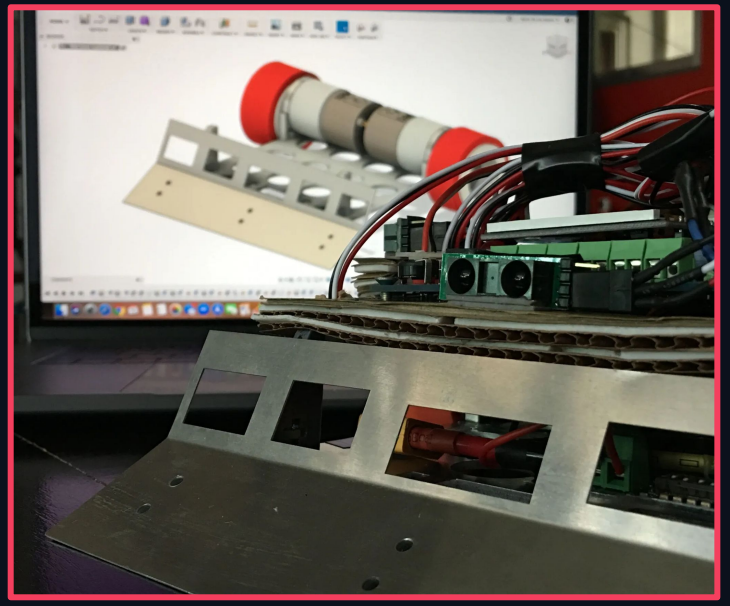
- **Accessibility:** physical Robot Sumo can be limited by cost and equipment availability. The proposed simulation gives users a virtual way to experiment with Robot Sumo.

- **Robotics Simulation:** the project aims to provide a more robust simulation environment for modeling Robot Sumo robots and their interactions.

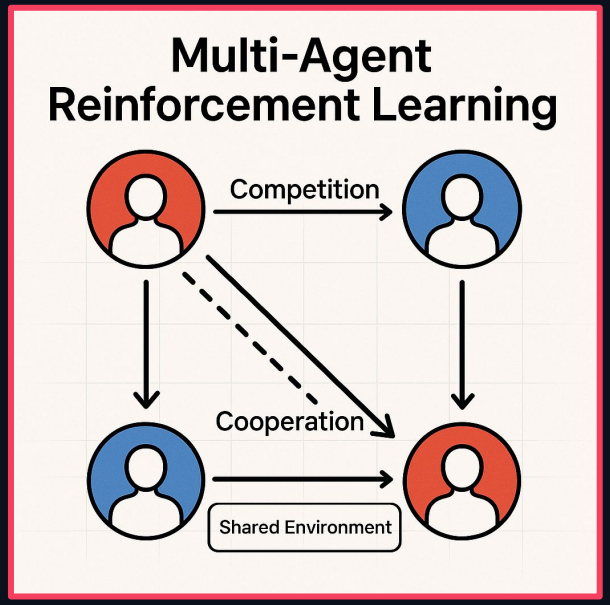
- **RL Experimentation:** The project will provide a structured environment where users can develop and evaluate reinforcement-learning algorithms under controlled conditions.

Project Approach

- **Configure Simulation:** the user can configure an environment with robot or ring options and run both visual and non-visual simulations.
- **Run RL Training:** the user can train a model with reinforcement learning algorithms through repeated non-visualized simulations.
- **Customize RL Algorithms:** the user can customize their own reinforcement learning algorithm or import an already trained agent.
- **Participate and Test:** the user can take part in the simulation to test the agent's performance or influence its training.



Novel Features and Functionalities



- **Dedicated RL Environment:** although there are other types of robotics simulations for RL training, this project provides a dedicated Robot Sumo environment where users can create, and test their own RL algorithms.
- **Robot Sumo Simulation:** combines physics-based simulation with Robot Sumo-specific interactions and competition structure.
- **Multi Agent Reinforcement Learning:** the project aims to use MARL techniques for training, which is a current area of research and experimentation.

Tools Under Consideration

Simulation Engines

Pygame | Godot | Unity

RL / MARL APIs & Toolkits

Gymnasium | PettingZoo | MOMAland | VMAS

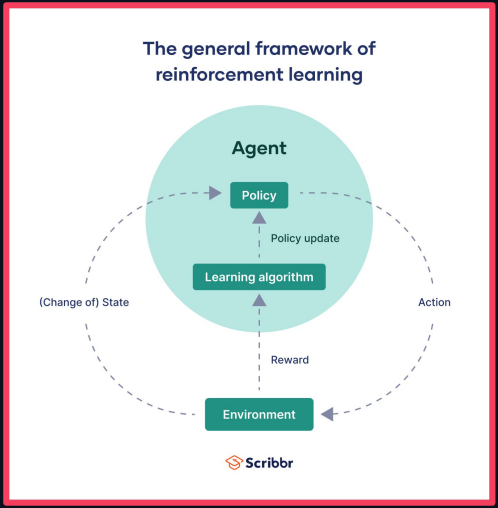
RL Approaches Being Evaluated

Single-agent | Evolutionary | CTDE | Imitation learning

Engine Evaluation Criteria:

- Developer accessibility / familiarity
- Suitability for the simulation
- Ease of RL algorithm and API integration.
- Optimization for continuous training
- 2D / 3D support

Reinforcement-Learning Approaches



- **Single-agent:** Baseline approach for initial demos, using trial-and-error and reward feedback to train an agent [6].
- **Evolutionary RL:** Improves a population of agents over generations [5].
- **CTDE (Centralized Training Decentralized Execution):** Uses shared information during training while allowing agents to make decisions independently during the process [6].
- **Imitation learning:** Trains agents to choose actions based off of demonstrations of desired behavior [8].

Technical Challenges

- **RL learning curve:** Limited prior team experience with reinforcement learning requires research and careful algorithm selection.
- **Simulation/Visualization separation:** The simulation must support non-visualized training so computation is not wasted on rendering [7].
- **Algorithm & computing integration:** Integration between the supercomputing resources, RL APIs and the simulation may prove difficult.
- **Reproducibility & data management:** Training runs need organized data and controlled randomness so the results can be repeated and compared.

Milestone 1 - September 28

- **Tool Demos:** develop simulation demos with each engine option to gauge their value to the project in accordance with the guidelines.
- **Research and AI.Panther:** conduct research regarding already established RL APIs and their accessibility to the engines. Get access to AI.Panther for computing power.
- **Write Documents:** write out the Requirements, Design, and Test Documents in order to have a clear vision going forward.

Collaboration Tools

Source Control	**GitHub**	Project management tool allowing for multiple collaborators.
Software Development	**SCRUM**	Iterative development and continuous testing between simulation and RL APIs.
Documents / Presentations	**Google Drive, Docs, Slides**	Centralized storage and simultaneous collaborative editing.
Communication	**Discord, WhatsApp**	Discord for development conversations and whatsapp for general coordination.
Task management / Meetings	**Shared Google Calendar**	Organization for meetings and deadlines.

Milestone 1 Task Matrix

Task Matrix: current percentage split of tasks, as outlined in the project plan.

Task	Avlynn	Joshua	Faysal	Adebare
Create demos with all engines	10%	50% (Godot, Pygame)	30% (Unity)	10%
Research RL API integration	40% (Gymnasium, VMAS)	10%	10%	40% (PettingZoo, MOMAland)
Requirement Document	40%	20%	20%	20%
Design Document	20%	20%	40%	20%
Test Document	20%	40%	20%	20%

Milestone 2 — October 26

Goal: Basic Simulation

- Implement robot models with corresponding hitboxes.
- Give robots the ability to interact with the environment and set start and end conditions for a sumo match.
- Give exposable inputs for the robot controller.
- Provide readable full state feedback for RL training.

Goal: RL Research

- Research RL training API integration with engine of choice.
- Define API interface and implement it into the simulation

Milestone 3 — November 23

Goal: Simulation MVP

- Abstract the physics simulation into a step function with feedback.
- Implement non-visualized simulations for training.
- Optimize simulation physics and components to facilitate RL training.
- Implement player input for agent testing.
- Define an RL training algorithm to improve an agent iteratively.
- Define weights and parameters for RL training.
- Finalize API integration with the engine of choice.
- Save training iterations/sessions for reproducibility

QUESTIONS?

References

[1] Ning & Xie (2024), “A Survey on Multi-Agent Reinforcement Learning and Its Application.”
[2] Gymnasium-Robotics Documentation, Farama.org.
[3] Bettini et al., “VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning.”
[4] MOMAland Documentation, Farama.org.
[5] Gronauer & Diepold (2021), “Multi-Agent Deep Reinforcement Learning: A Survey.”
[6] Bai, Cheng & Jin (2023), “Evolutionary Reinforcement Learning: A Survey.”
[7] Varier et al. (2022), “AMBF-RL: A Real-Time Simulation Based Reinforcement Learning Toolkit for Medical Robotics.”
[8] Hussein et al. (2017), “Imitation Learning.”