Robot Sumo Simulation:
Build AI Agents that Battle, Adapt, and Work as a Team

Team Members
Adebare Kofo-Abayomi: akofoaabyomi2022@my.fit.edu
Avlynn Wade: awade2024@my.fit.edu
Faysal Zaki: fzaki2023@my.fit.edu
Joshua Oziel Fernández Alvarado: jfernandezal2023@my.fit.edu

Client & Faculty Advisor
Department of Electrical Engineering and Computer Science
Dr. Hanks, thanks@fit.edu

Section 1.0 Date of Meetings with Advisor (and Client) for Plan Development

Meeting Purpose	Date and Time
Introductions and Initial Project Plan Evaluation	Tuesday, August 25th, 2026 ● 11:00 am - 12:00 pm
Final Project Plan Evaluation Before Submission	Friday, August 28th, 2026 ● 12:00 pm - 12:30 pm

Section 2.0 Goal and Motivation

The overall goal of the Robot Sumo Simulation project is to create a more accessible and interactive platform for users interested in Robot Sumo, general robotics, and reinforcement learning (RL). Currently, access to physical Robot Sumo robots can be limited by factors such as cost and equipment availability. The proposed simulation could help mitigate these challenges by allowing users to experiment with Robot Sumo virtually, reducing the need for physical robots. Additionally, from a robotics perspective, the project aims to provide a more robust simulation environment for modeling Robot Sumo robots and their interactions. Existing systems may lack advanced physics, user engagement, and overall Robot Sumo-specific functionality. Contrastingly, from an RL standpoint, the project will provide users with a structured platform to develop and evaluate different RL algorithms within a controlled environment. Overall, the Robot Sumo Simulation project seeks to make Robot Sumo more accessible and also provide an accessible platform for specialized robotics and RL experimentation.

Section 3.0 Project Approach

Section 3.1 Configuration for the Physics Simulation

The user can configure a Robot Sumo environment with multiple settings for robot options, ring options, and environmental effects to influence the simulation. After configuring the simulation, the user can run the simulation in order to train or test an agent in customizable conditions. The simulation can be visualized, or it can be run only through physics updates to save computational power and speed up training.

Section 3.2 Run Experiments with Existing Reinforcement Learning Algorithms

The user can utilize already established reinforcement learning algorithms and apply them to the simulation environment in order to train an agent to control a robot. The training consists of multiple non-visualized simulations running sequentially with positive and negative feedback based on the robot's performance within the specified Robot Sumo objectives.

Section 3.3 Import Custom Agents and Reinforcement Learning Algorithms

The user can customize their own reinforcement learning algorithm with their preferred weights and parameters in order to train the agent. Additionally, the user can import an already trained agent in order to test it, or train it further with updated parameters. However, custom reinforcement learning algorithms and agents must comply with the simulation's given parameters and API in order to execute properly.

Section 3.4 Participate in the Simulation to Test or Influence the Agent

The user can run a visualized instance of the simulation and control a robot through inputs. By participating in the simulation, the user can subjectively test the performance of an agent, or take part in the training process in order to influence its training and give it certain patterns if desired. The user can take part in a visualized simulation and compete against agents to test the simulation or explore configurations for the training process.

Section 4.0 Novel Features and Functionalities

The Robot Sumo Simulation project will showcase several novel features and functionalities. To start, the system will allow users to upload and evaluate their own reinforcement learning algorithms within a 3D Robot Sumo environment. Even though there are other existing RL-based robotics frameworks that provide similar functionality, like the Python library Gymnasium-Robotics, which offers a "collection of robotics simulation environments" for testing RL frameworks [2], the proposed system's novelty lies in applying user-defined RL algorithms specifically within a Robot Sumo environment. Overall, the goal of this outlined feature is to allow the Robot Sumo Simulation to function as an interactive RL testbed for both robotics and RL-focused users.

Moving on, another notable and novel feature will be the system's support for multi-agent reinforcement learning (MARL), an area that remains fairly open to further research and experimentation [1]. Specifically, these MARL techniques will be applied within the dedicated Robot Sumo environment. Which, in turn, will allow users to experiment with multiple competing agents and explore how different agents learn to interact with one another.

Finally, the proposed project will combine physics-based Robot Sumo simulation, RL training, and interactive visualization within one cohesive system. Existing Robot Sumo simulators and RL methodologies demonstrate that these concepts can be implemented on their own. However, the proposed system aims to integrate them into an accessible environment, one that includes RL experimentation as a major capability. Therefore, the project's novelty lies in its combination of Robot Sumo-specific simulation and flexible RL experimentation.

Section 5.0 Potentially Useful Algorithms and Tools

There are numerous potential algorithms and software tools that could be useful in the development of the Robot Sumo Simulation project. First and foremost, in order to promote simple and well-defined experimentation, the reinforcement learning component of the project

will conform to existing Multi-Agent Reinforcement Learning (MARL) APIs. For example, APIs such as Gymnasium, PettingZoo, MOMAland, and the Vectorized Multi-Agent Simulator (VMAS) could be utilized as convenient MARL development toolkits [3][4]. Additionally, for constructing the actual Robot Sumo simulation, various game engines such as Pygame, Godot Engine, and Unity could be helpful in creating the 2D and 3D environments, and for implementing the actual Robot Sumo models.

Moving on, several types of reinforcement learning algorithms could also be beneficial to the development of the intended Robot Sumo multi-agent system. A few of these algorithms are outlined below (Sections 5.0.1 through 5.0.3). It is worth noting that these potential RL algorithms are not an exhaustive list. After further research is conducted throughout Milestone 1 (as outlined in Section 9), other RL algorithms and techniques may be evaluated and selected.

5.0.1 The Standard "Single Agent" Approach

The first potential algorithm is the standard "single agent" RL approach, where an agent "interacts with an environment of unknown dynamics in a trial-and-error fashion" and receives rewards/feedback based on the actions it performs [6]. For the Robot Sumo project in particular, a standard single-agent RL algorithm could provide baseline results for evaluating individual agent behaviors, before moving to the more complex MARL techniques.

5.0.2 The Evolutionary Approach

The second potential algorithm is the evolutionary approach, which is an almost hybrid method that combines evolutionary computing (EC) and traditional RL practices. Specifically, evolutionary reinforcement learning (EvoRL) "involves maintaining a population of agents" that are evaluated and improved over successive generations. This technique may be useful to the project, as this method has shown promising results in handling multi-agent systems, especially those with "conflicting objectives for rewards" such as in competitive environments like that of Robot Sumo [5].

5.0.3 The Centralized Training Decentralized Execution (CTDE) Approach

The next approach that could be useful to the project is CTDE, or Centralized Training Decentralized Execution. In this technique, agents have a "centralized" training scheme, which means that they "...utilize shared computational facilities…to exchange information during training" [6]. However, agents ultimately make decisions independently during implementation, hence the "decentralized" term [6]. This method could be beneficial for the outlined Robot Sumo project because information about the other robots could be used during training to develop expected behaviors. However, each robot would still operate independently once actually deployed in a competitive setting.

5.0.4 The Imitation Learning Approach

The final approach that could be utilized within the project would be the Imitation Learning technique. In this approach, an agent "is trained to perform a task from demonstrations by learning a mapping between observations and actions" [8]. In short, the agents learn from analyzing examples of the intended behavior, for example, watching chess games. This approach could be practical for the Robot Sumo task, as within this specific RL research domain "special attention is given to learning methods in robotics and games" [8]. Meaning that these robotics-based Imitation Learning methodologies could be easily researched and applied to the proposed task.

Finally, the Florida Institute of Technology supercomputing resource, AI.Panther, could be employed to handle the substantial computational task associated with reinforcement learning of this scale. This tool would be incredibly useful to prevent any unnecessary stress on developers' personal computers.

Section 6.0 Technical Challenges

The development of the Robot Sumo Simulation project presents numerous technical challenges. To start, developers will have to navigate the complexities of reinforcement learning (RL) with limited prior experience in this domain. Specifically, the team will have to design optimal training algorithms, with adequate separation between the actual simulation environment and the visualization. This separation is critical for implementing the desired "headless" training mode, in which the simulation can run without rendering visual output, reducing computational overhead [7]. Additionally, developers will have to research and select best-practice RL algorithms for the proposed task to ensure the simulation works as expected.

Moving on, the team may encounter integration challenges associated with running the training simulation in the AI.Panther environment. Similar to the difficulties described above, the team currently has limited experience in using and running programs with this resource. As a result, additional exploration and learning will be required to utilize this proposed tool correctly.

Finally, arguably one of the most important technical challenges is reproducibility. This concept is especially crucial, yet difficult to achieve, within the outlined experimental environment, which contains numerous moving parts and sources of variability. To mitigate this challenge, developers will need to maintain diligent data management and organizational practices. Without these practices, important results and experimental outcomes could be lost, potentially leading to setbacks in overall project development. Along with this, another important consideration pertaining to reproducibility will be controlling sources of randomness, primarily through seeding techniques. In short, using consistent random seeds will make experiments more reproducible and make it easier to compare different training runs.

Section 7.0 Itemized Tasks for Milestone 1 (Sep 28)

1. Multiple demos will be developed utilizing the potential tools outlined in Section 5.0 in order to evaluate their respective benefits and shortcomings. The results of said evaluation will

determine the final set of tools with which to proceed for the rest of the project's development.

a. For the simulation, the three primary tools that will be evaluated are Pygame, Godot, and Unity. These tools will be evaluated primarily on how accessible they are for developers to use, how well suited they are for the development of the proposed simulation, how easily RL algorithms and APIs can be integrated, and how optimizable they are for continuous training.

Figure 1. Example Engine Comparison Matrix

Engine	**2D/3D Support**	**Developer Familiarity**	**Simulation Fit**	**RL Training and APIs**	**Optimization**
Pygame	2D				
Godot	2D + 3D				
Unity	2D + 3D				

b. The four reinforcement learning approaches mentioned in Section 5.0 (single agent, evolutionary, CTDE, and imitation) will be evaluated on their feasibility as the project progresses and scope allows. The outlined RL approaches will be integrated and tested as the simulation and APIs evolve, and as the complexity of the overall project increases. The single-agent approach, in particular, will be used for the demos in order to lower scope, and more complicated approaches (MARLs) will be implemented once appropriate.

2. Resolve Technical Challenges:
   a. Get access to Florida Tech's supercomputing resource, AI.Panther, for later training.
   b. Research and understand the pros and cons associated with each potential engine.

3. Select and Test Collaboration Tools:
   a. Source Control: GitHub
      i. Will be used for source control, branches, pull requests, issue tracking, and integration with project-management tools in order to organize the project and allow multiple people to complete tasks concurrently.
   b. Software Development: SCRUM
      i. Iterative design techniques (SCRUM) will be utilized to organize project development, as well as manage scope with continuous testing between the simulation and RL training APIs.
   c. Documents and Presentations: Google Drive, Google Docs, and Google Slides
      i. Google Workspace will be used for centralized file storage and simultaneous collaborative editing to keep files organized and easy to access.

d. Team Communication: Discord and WhatsApp
    i. Discord will be used for work sessions and collaborative communication regarding specific tasks during milestones.
    ii. WhatsApp will be used for general communication and organization.
e. Task Management and Calendar: Shared Google Calendar
    i. Selected for easy organization of meetings and major milestone deadlines.

4. Create Requirement Document
5. Create Design Document
6. Create Test Plan

Section 8.0 Itemized Tasks for Milestone 2 (Oct 26)

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

Section 9.0 Itemized Tasks for Milestone 3 (Nov 23)

- 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 controls for agent testing.
- Define an RL training algorithm to improve an agent iteratively.
- Define weights and parameters for RL training.
- Finalize API integration with engine of choice.
- Save training iterations as reproducible data.

Section 10.0 Task Matrix for Milestone 1

Task	**Avlynn**	**Joshua**	**Faysal**	**Adebare**
Create demos with all engines	10%	50% (Godot, Pygame)	30% (Unity)	10%
Research API integration	40% (Gymnasium, Vectorized Multi-Agent	10%	10%	40% (PettingZoo, MOMAland)

	Simulator (VMAS))			
Requirement Document	40%	20%	20%	20%
Design Document	20%	20%	40%	20%
Test Document	20%	40%	20%	20%

Section 11.0 References

[1] Z. Ning and L. Xie, "A Survey on Multi-Agent Reinforcement Learning and Its Application," *Journal of Automation and Intelligence*, vol. 3, no. 2, Feb. 2024, doi: 10.1016/j.jai.2024.02.003.

[2] "Gymnasium-Robotics Documentation," Farama.org. Accessed: Aug. 25, 2026. [Online]. Available: https://robotics.farama.org/index.html

[3] M. Bettini, R. Kortvelesy, J. Blumenkamp, and A. Prorok, "VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning," arXiv.org. Accessed: Aug. 25, 2026. [Online]. Available: https://arxiv.org/abs/2207.03530

[4] "MOMAland Documentation," Farama.org. Accessed: Aug. 26, 2026. [Online]. Available: https://momaland.farama.org/index.html

[5] S. Gronauer and K. Diepold, "Multi-Agent Deep Reinforcement Learning: A Survey," *Artificial Intelligence Review*, Apr. 2021, doi: 10.1007/s10462-021-09996-w.

[6] H. Bai, R. Cheng, and Y. Jin, "Evolutionary Reinforcement Learning: A Survey," *Intelligent Computing*, Apr. 2023, doi: 10.34133/icomputing.0025.

[7] V. M. Varier, D. K. Rajamani, F. Tavakkolmoghaddam, A. Munawar, and G. S. Fischer, "AMBF-RL: A Real-Time Simulation Based Reinforcement Learning Toolkit for Medical Robotics," *2022 International Symposium on Medical Robotics (ISMR)*, pp. 1–8, Apr. 2022, doi: 10.1109/ismr48347.2022.9807609.

[8] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation Learning," *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–35, Jun. 2017, doi: 10.1145/3054912.