WP Artificial Intelligence and Software Agents

LaserTag Group Competition: Assignment

MARS Group

February 23, 2023

LaserTag: Game Description

LaserTag is a multi-agent game simulation written in MARS. The game consists of three or four teams – each made up of three agents – competing against each other. Team members need to coordinate with each other and be careful not to become overpowered by agents from other teams. To play the game well, agent behavior needs to be designed intelligently such that each agent interacts with its environment, its team members, and its opponents to work towards the common goal of the team.

Initial Setup

Before beginning to work on your assignment, please complete the following steps.

- **Communication:** Join the channel # lasertag in the Slack workspace MARS Explorers. All communication about LaserTag and the competition will occur in this channel.
- **Repository:** Clone the LaserTag GitHub repository.
- **Documentation:** The LaserTag documentation (PDF) is in the `Documentation` directory of the repository. Please read it thoroughly to learn about the game's concepts and mechanisms.
- **Project Setup:** Set up LaserTag in JetBrains Rider.

  See the section *Project Setup* of the LaserTag documentation for more details.
- **Code:** The LaserTag source code includes the setup for the three or four agent teams that will compete in one simulation. Review and experiment with the code to learn about the game and how agents behave when calling different methods.

Goal of the Game

The objective of the game will be discussed and decided with you in the # lasertag Slack channel (see **Communication** above for the link to the Slack workspace).

Your Assignment

Your assignment is to create two agent types. One will be a *rule-based* agent type and the other will be a *learning* agent type. Your agents will compete in the LaserTag competition against agents designed by other groups. The goal of this assignment is to

1. program rule-driven and learning-driven agent behaviors and
2. observe and analyze differences in rule-based and learning-based agent behavior.

Rule-based agent

In this behavior modeling approach, your main task as developers is to define *rules* (e.g., in the form of `if-else` statements) for your agents to follow. These rules will determine under which circumstances your agents call the methods that are available via the `PlayerBody`.

Learning agent

In this behavior modeling approach, your main task as developers is to define:

- *states* that your agents can be in and
- *rewards* that your agents receive for their *actions*.

The states will be passed to a reinforcement learning (RL) algorithm that returns a corresponding *action* for your agent to perform. We recommend table-based *Q*-learning. Through this learning approach, your agent can store its experiences for state-action pairs in a table. By querying the table, the agent can let its experiences influence its behaviors in the present and future.

The MARS Framework provides some base classes that you can use for your implementation:

- `Mars.Components.Services.Learning` for basic learning algorithm and data structures
- `Mars.Components.Services.Explorations` for action selection policies

You can also implement from scratch or use external libraries. Here are some examples:

- Introduction to Q-Learning Using C#
- Implement Q-Learning Algorithm in C#
- Introduction to Machine Learning with C# and ML.NET

Constraints

All implementations must satisfy the following constraints:

- Your agent type's main class (e.g., `MyAgentMind`) must inherit from the class `AbstractPlayerMind`. In `MyAgentMind`, you can define your agent type's mind. From here, your agent type can access a `PlayerBody` (via an interface `IPlayerBody`), which represents its physical form.

  See the section *Project Structure* of the LaserTag documentation for the project's class structure.

  See the section *Agent properties and methods* of the LaserTag documentation for a list of properties and methods that are available via the `PlayerBody`.

- Satisfy all constraints listed in the section *Constraints for Developers* of the LaserTag documentation.

Submission and Competition

- **Submit your agents:** Your agent types' source code must be submitted for the competition via LOCATION by HH:mm on DDMMYYYY. Submissions must include only your main class inheriting from `AbstractPlayerMind` and its associated classes, if any. To-do: Update date, time, location
- **Attend the competition:** The competition will begin on DDMMYYYY at HH:MM. The competition/tournament style will be discussed and decided with you in the LaserTag Slack channel (see above). To-do: Update date, time

Questions and Feedback

Please submit any questions, feedback, feature requests, and bug reports via the LaserTag Slack channel. We (Nima Ahmady-Moghaddam and Jonathan Ströbele) will do our best to address them as soon as possible.

Happy coding and good luck! :)