WP Artificial Intelligence and Software Agents

LaserTag Group Competition: Assignment

MARS Group

February 21, 2023

LaserTag: Game Description

LaserTag is a multi-agent game simulation in which three or four agent teams, each made up of three agents, compete 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.

Your Assignment

1. **Set up communication:** Please join the channel # lasertag in MARS Explorers on Slack. All announcements, communication, and technical discussions about LaserTag and the competition will occur in this channel.
2. **Set up the project:** LaserTag is developed and written in MARS. Please check out the project's GitHub repository and set up the project in Rider. The directory **LaserTagBox** contains the game. See the section *Project Setup* in the LaserTag documentation for more details.
3. **Review the documentation:** The LaserTag documentation (PDF) can be found in the **Documentation** directory of the repository. Please use it to familiarize yourself with the model's concepts and mechanisms while designing your AI.
4. **Study the code:** The 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 get a feeling for what the possibilities are and how agents behave when calling different methods.
5. **Write your agent logic:**
   - Create a class that inherits from `AbstractPlayerMind`. Your agents' mind will be written inside this class. From here, it can access a `PlayerBody` (via an interface `IPlayerBody`), which represents the agent's physical form. Call methods defined in `PlayerBody` to shape the agent's physical behaviors and routines. Here is the list of available methods:
     - `ChangeStance2(Stance)`
     - `ExploreBarriers1() : List<Position>`
     - `ExploreDitches1() : List<Position>`
     - `ExploreHills1() : List<Position>`

  - `ExploreEnemies1() : List<EnemySnapshot>`
  - `ExploreTeam() : List<FriendSnapshot>`
  - `GetDistance(Position) : int`
  - `GoTo(Position) : bool`
  - `HasBeeline1(Position) : bool`
  - `Reload3()`
  - `Tag5(Position) : bool`

  The numbers at the end of some of the methods indiciate the number of `ActionPoints` required to execute the method. Please see the LaserTag documentation for more details.

- Your code must meet the following requirements:

  a) Follow all rules listed in the section *Constraints for Developers* of the LaserTag documentation.

  b) No simulation-external information may be loaded into the simulation at runtime (your agent must have an empty constructor).

  c) No loops that are known not to terminate within a reasonable time (e.g., `while(true)`).

  d) The use of `PropertyDescription` tags (for loading external information into your agents at runtime) is not allowed.

6. **Submit your AI:** The deadline for submitting your code for the competition is HH:mm on DDMMYYYY. Only your class inheriting from `AbstractPlayerMind` (and all associated classes, if any) will be considered. Please submit your code via LOCATION. To-do: Update date, time, location

7. **Attend the competition:** The competition will begin HH:MM on DDMMYYYY. The competition/tournament style will be discussed and decided with you in the Slack channel. To-do: Update date, time

Goal of the Game

The objective of the game will be discussed and decided with you in the Slack channel.

Questions and Feedback

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

Happy coding and good luck! :)