**General Statements**

All three environments share several non-functional requirements;

* Good performance.
* Machine-readable, and Human-readable interfaces.
* Configurable difficulty.

The last requirement is important because it became clear early in the project that training time would be limited, and that it would not become clear how quickly the agents would train until late in the development lifecycle. As such was essential to have tuneable parameters that could reduce the difficulty of the environments if they proved too generally challenging for all algorithms. If every method failed to improve as a result of poor environment design, that would not allow for any meaningful comparison of the methods themselves.

**Tag**

The agent is tasked with controlling an avatar in a 2D environment. The goal is to prevent the agent avatar (the runner) from contacting hostile agents (the seekers). At the beginning of each epoch, the seekers are placed in a circle around the seeker, each has a randomly chosen distance and angle. At each step, the seekers move directly towards the runner. The epoch ends when the seeker is caught or moves off the edge of the game area. The epoch will also end if the runner successfully evades the seekers for a set number of steps (configurable).

Functional Requirements.

* Player can take actions.
  + Turn Left.
  + Continue Straight.
  + Turn Right.
* User can configure environment.
  + Number of seekers.
  + Speed ratio between runner and seeker.
  + Size of arena (height & width).
  + Number of time steps until epoch termination.
* User can reset environment.

The percept is a 1D vector. It contains the position of the agent, the rotation of the agent, and the positions of each seeker.

Reward Scheme.

* +1 per time step.

**Tic-Tac-Toe**

This is an arbitrary-size version of the game. The environment is an N⨯N grid (where N is a configurable parameter of the environment). At each time step the agent marks an empty tile with its symbol, then the opponent marks an empty tile with its symbol. The winner is the first player to construct a line of N symbols. By default the opponent follows a ε-Greedy policy. This means that it takes a random action ε% of the time (ε is a configurable parameter of the environment), and the rest of the time it takes the optimal action.

Functional Requirements.

* Player can take actions.
  + Turn Left.
  + Continue Straight.
  + Turn Right.
* User can configure environment.
  + Dimensions of board.
  + Opponent strategy.
* User can reset environment.

Reward scheme.

* +1 per time step.
* +2n for a length n line of symbols.
* +1000 for winning a game.
* -100 for takin an invalid move.

**Maze**

In this environment, the world is a grid of squares. The agent controls an avatar, and is tasked with moving it around the grid to collect coins. There are empty squares that the agent can move through, and solid walls that block its movement. The location of walls is preset but the location of coins is randomly generated. The agent’s starting position can be randomly chosen or preset.

Functional Requirements.

* Player can take actions.
* Up.
* Down.
* Left.
* Right.
* User can configure environment.
  + Maze configuration.
  + Number of coins.
  + Initial placement of agent (set position or random).
* User can reset environment.

Non-Functional Requirements.

* Good performance.
* Machine-readable, and Human-readable interfaces.

The percept is a 1D vector containing the contents of each square; empty, wall, agent, coin.

Reward scheme.

* +1 per step.
* +10 per coin collected.