**2D Racing**

This is a top-down racing game in which the player is tasked with controlling a car around a track. The goal is to complete laps as quickly as possible. The car maintains a constant forward speed, and the player can control it to turn left or right. The game ends when the car crosses the finish line.

Functional Requirements.

* Agent can take actions.
  + Pass.
  + Turn car left.
  + Turn car right.
* User can configure environment.
* User can reset environment.

Non-Functional Requirements.

* Good performance.
* Human-readable interface.

Reward Scheme.

* Reward at each step, based on the car’s distance the from center of the track. 1 at exact center, 0 at exact edge, interpolating linearly.
* Reward for crossing finish line. 1000.
* 100 for winning.

**Tic-Tac-Toe**

This is an arbitrary-size implementation, where the environment is an N⨯N grid. At each time step the agent marks an empty tile with an X, then the opponent marks an empty tile with an O. The winner is the first player to construct a line of N marks. 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). The rest of the time it takes the optimal action, calculated by mini-max search.

**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 block its movement. The location of walls is preset but the location of coins is randomly generated.

* Percept.
  + A matrix containing the contents of each square; empty, wall, agent, coin.
* Action Space.
  + Move Up.
  + Move Down.
  + Move Left.
  + Move Right.
  + Pass.
* Reward structure.
  + +1 per step.
  + +10 per coin collected.