Laboratório de IA e CD Report

- **Implementation of Attax**

The game finishes when:
- The board is full;
- A player is stuck (meaning he can't make any move);
- The threefold repetition rule verifies, that is the board state repeat itself three times during the game, which means the players can't make any improvements.

The winner is always the one who has the most territory, meaning has the most pieces in the board. If they both have the same territory, it is a draw.
In case of 'stuck', we count territory having in mind the territory that wasn't fill by the only player who could do it.

- **Implementation of Go**

The game finishes when:
- When players pass their turn consecutively.

The winner is always the one who has the most territory(subtracting self captured pieces) according to the Japanese rules (Addition of Komi to white pieces points as it plays in second). By having the same points, it is a draw.

- **Game modes and Interface**

The interface in both games is mainly used to see the results more widely and to make the type of game changing: human vs human, human vs agent, agent vs agent.

About the human, we relate the mouse click to the game move: we first click in the piece we want to move and then click in the space we want to move the piece to (Attax); we only click in the "space" (by space we mean the intersection of the 4 lines) we want to put the piece in (Go).

About the agent, is an autonomous agent who does the best move according to the MCTS algorithm.

- **Socket-based communication**

Although it wold be better, the communication is not independent from interface, because we couldn't interrupt the main event loop of the interface to extract necessary info for communication

Faced with this problem, we define separately a class Client which is given to the interface if we want to play with communication.

- **AlphaZero (Neural Network + MCTS)**

AlphaZero uses the MCTS algorithm to explore and analyze potential moves in the game.

In AlphaZero, the MCTS doesn't have the simulation step. Instead, it uses a neural network - ResNet - to evaluate states and possible moves. It takes a state as an input and outputs a value and a policy.

The model generates data by playing games against itself - Self Play - starting randomnly and getting more knowledge by the iterations.

Our model as temperature implemented as to encourage exploration. We also added noise to the prior probabilities in the root node. This again promotes exploration and avoids overfitting to specific sequence moves.