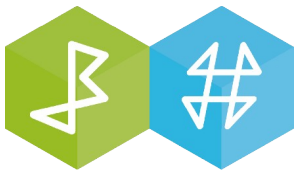
ABEL Léna
DUBOL Lucas
PICOT Nivolas
VALENNE Nathan
TOILLON Samuel
Group 5

SAE S2.126:

Fox and Geese



BUT Informatique 2021/2022

Table of contents

Introduction

This report documents the milestone #1 results of the '*SAÉ 201 – Développement IHM'.* For this project, we choose to implement the game ***'Fox and Geese' (Fag)*** which consists of one board game,13 yellow pawn geese and 1 red pawn fox.

The objective of the fox is to capture the geese by jumping over them in a straight line while the geese aim at blocking the movement of the fox through the benefit of their numbers.

Because Fox and Geese is a game of inequality, we propose to implement different decisions strategies for the fox and the geese, for a total of 4 strategies.

1 User interface – How to play the game

At the end of the project the view and the control of the game will be 100% graphical oriented. Before these features are ready, we use command line arguments to configure the game options such as the modes, the name of the players, the team membership and selection of the AI algorithms. For example, among the modes of operations one can select one of the 4 following modes:

- Human against Human
- Human against Computer
- Computer against Computer (1 game)
- Profiler mode (computer against computer, multiple games)

As a result, the program can be run with the following parameters, as reported by the command '***java Fag -h***':

```
Initializing the random generator seed to: 42
Setting the gaming mode in HUMAN-COMPUTER mode.
Setting name of player-1 to Lena
Setting team of player-1 to Fox.

Usage: FoxAndGeese [OPTIONS]
List of options:
 -m|--mode   <Single|DoubleHuman|DoubleComputer> The gaming mode (HUMAN-COMPUTER, HUMAN-HUMAN, COMPUTER-COMPUTER).
 -n1|--name1 <NameOfPlayer1> The name of player #1.
 -n2|--name2 <NameOfPlayer2> The name of player #2.
 -t1|--team1 <Fox|Geese>     The team of player #1.
 -t2|--team2 <Fox|Geese>     The team of player #2.
 -a1|--algo1 <0|1|2>         The algorithm strength of computer #1.
 -a2|--algo2 <0|1|2>         The algorithm strength of computer #2.
 -s|--seed  <LONG> The seed of the random generator. If 0, (TODO-TBD).
Enjoy the game...
```

Fig 1: Helper output for java Fag --help

2 Decisions strategies

We propose 5 algorithms called ***GeeseAI0***, ***GeeseAI1, GeeseAI2, FoxAI0 and FoxAI1***, with the postfix '0-1-2', reflecting the increasing complexity of the five algorithms.

2.1 GeeseAI0

Implements a very basic and naive algorithm which main purpose is to test the functionalities and the requirements of the game during the development phase. The behavior of this algorithm is to take a random available goose and move it to a random valid place. This algorithm is not considered as a deliverable but we will use it as a reference for the measurements of the execution time and the winning strategies of the fox algorithms.

2.2 GeeseAI1

With *GeeseAI1* we try to come up with an algorithm that remains simple and fast. The idea is to try to win on the long run by continuously minimizing the risk of loosing. The strategy is as follows:

- First the algorithm plays defense. It collects the geese which are at risk to be eaten by the fox. It then tries to protects them with another goose while trying to close any gap that would permit the fox to jump over one of the geese at risk.

- Next, if none of the geese is in danger the algorithm plays attack. The goal is to try to move the remaining geese towards the fox in order to corner it. While playing this attack, the geese always try to remain grouped as much as possible because it brings the maximum protection to the team.

2.3 GeeseAI2

For the *GeeseAI2,* we intend to build a decision tree and walk through it with a *MinMax* algorithm which goal is to find the highest gain that a player can be sure to get without knowing the actions of the other player. This technique is typically used for AIs in chess games and is particularly suitable here for the Fox and Geese game in which all the game states are known to both players at any time (a.k.a. full information games).

MinMax is a recursive algorithm that is relatively easy to implement but its main drawback is its long execution time related to the depth of the decision tree. Therefore, we will tune the depth of the search tree to keep it in acceptable response time. We will also try some pruning techniques if time permits.

2.4 FoxAI0

Implements a very basic and naive algorithm which main purpose is to test the functionalities and the requirements of the game during the development phase. The behavior of this algorithm is to take the fox and move it to a random valid place. This algorithm is not considered as a deliverable but we will use it as a reference for the measurements of the execution time and the winning strategies of the geese algorithms.

2.5 FoxAI1

For FoxAI1, we try to avoid moving the fox too deep into the branches of the cross. In other words, the fox will try to stay in the middle of the board as much as possible. Otherwise, if the fox ends up in a branch of the cross it will avoid eating a goose if the AI think it’s a trap. So the fox will predict a flaw in the geese player/AI gameplay.