ROOMAZE

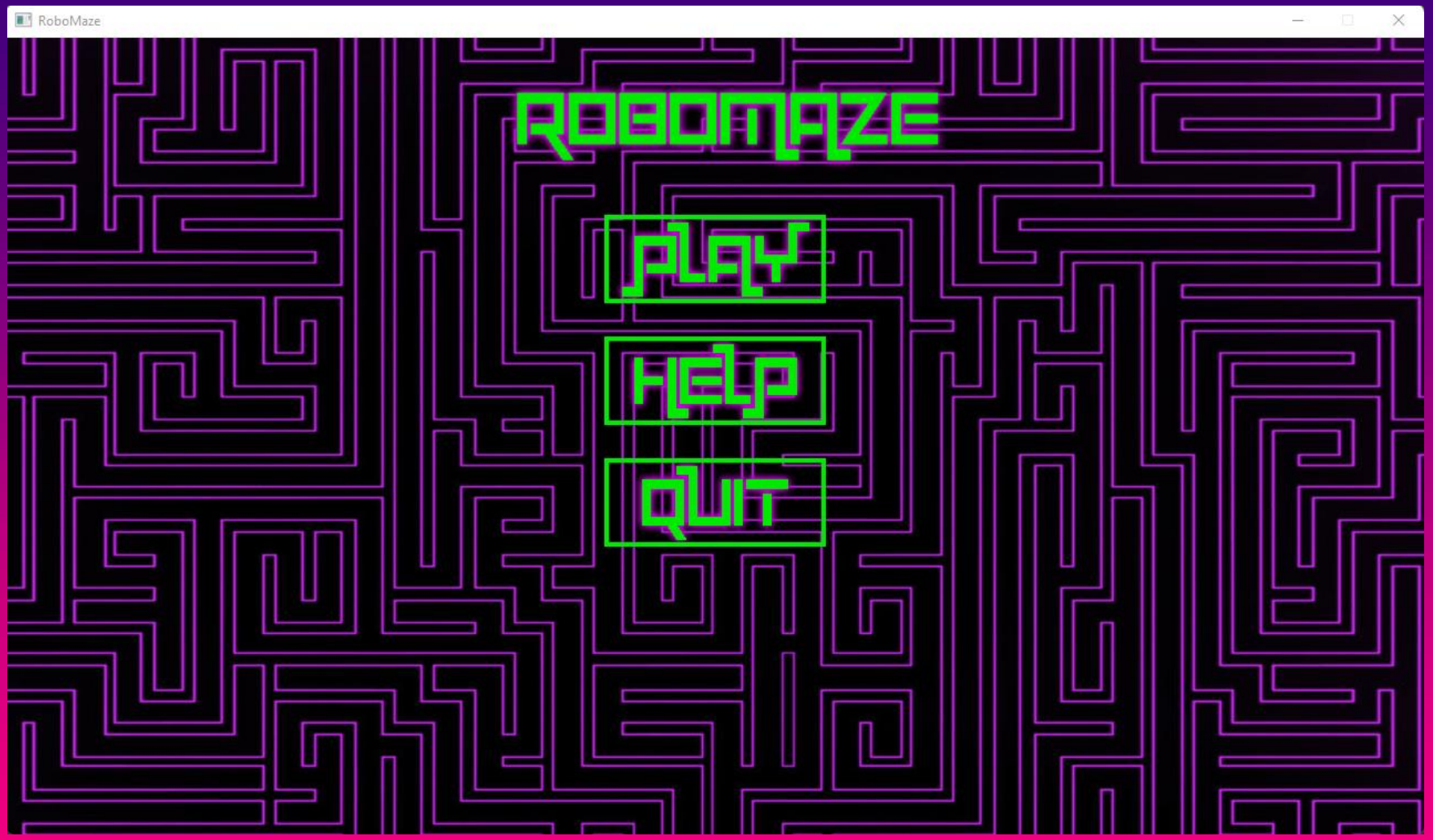
FULVIO SERRAO - 0124/002423

LUCA PEZZOLLA - 0124/002411

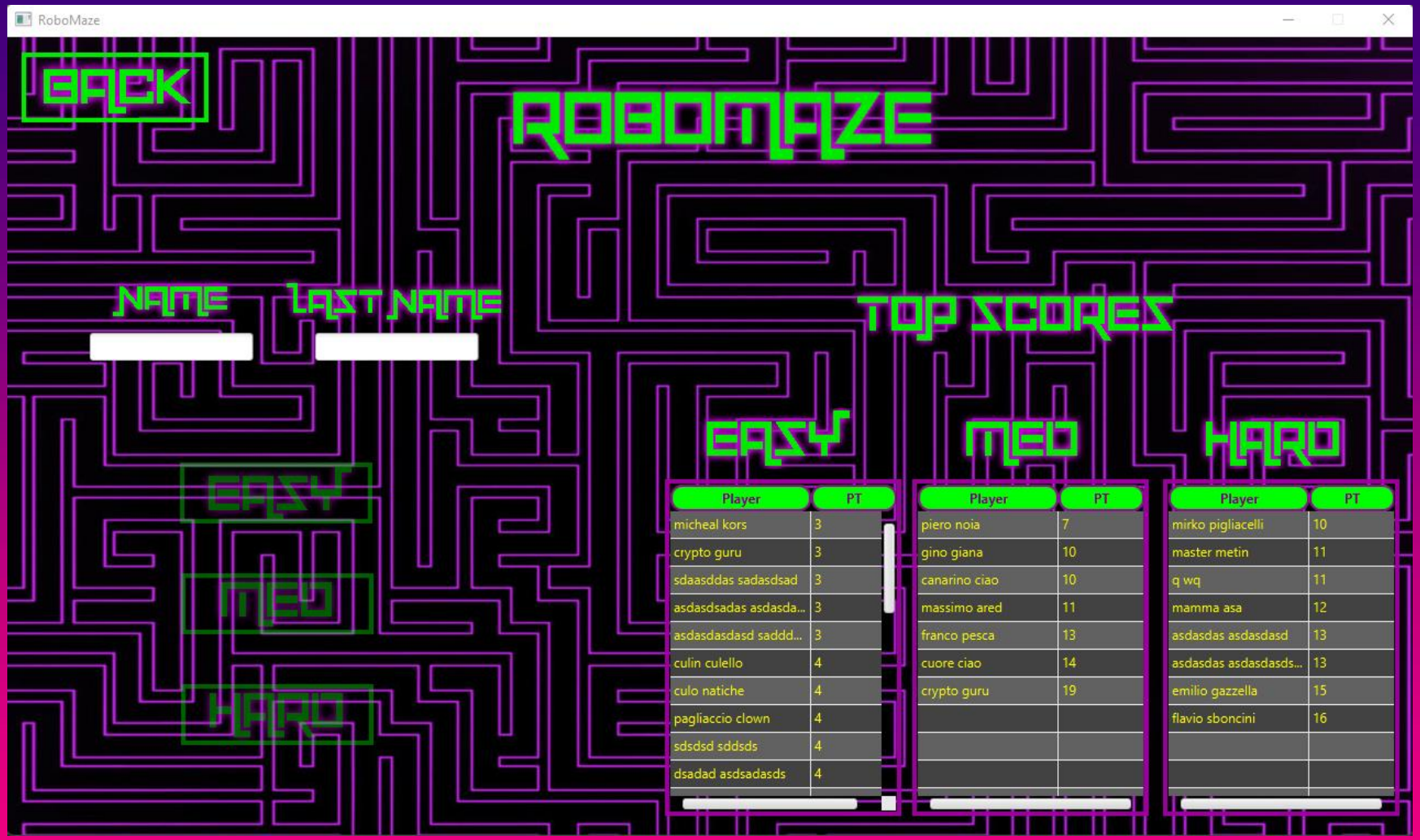
SIMONE MICILLO - 0124/002439

PRESENTAZIONE

RoboMaze è il titolo che abbiamo dato al nostro gioco con intelligenza artificiale. L'obiettivo è quello di permettere ad un Robot con nome e cognome di uscire da un labirinto. Il gioco consiste in tre livelli di difficoltà, rispettivamente: Easy, Medium e Hard. Il Robot si muove in base al colore della cella su cui transita utilizzando tre strategie differenti.



Schermata iniziale del gioco



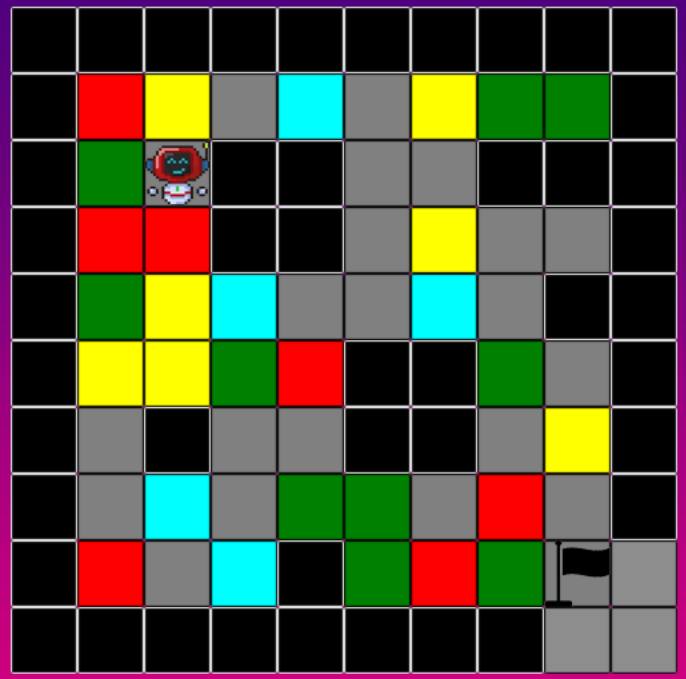
Schermata in cui l'utente può scegliere il livello di difficoltà, dopo aver inserito nome e cognome.

RoboMaze

STEPS: 3

ANGELA
INTERELLA

ROBOT STATE: EVADE



RoboMaze

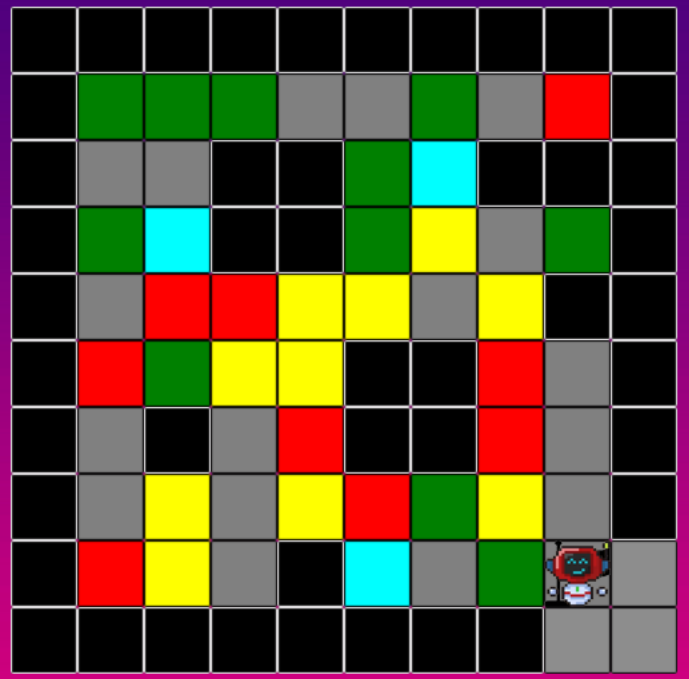
STEPS: 14

ANGELA INTERELLA

CONTINUE

ROBOT STATE: SEEK

GAME OVER



Player	PT
piero noia	7
gino giana	10
canarino ciao	10
massimo ared	11
franco pesca	13
cuore ciao	14
Angela INTerella	14
crypto guru	19

ANT COLONY OPTIMIZATION

L'ACO è un algoritmo di intelligenza artificiale che simula il comportamento di una colonia di formiche; nel nostro caso, simula il comportamento assunto da una colonia di formiche nel raggiungere un punto desiderato dal formicaio (problema di cammino minimo da una sorgente ad una destinazione, nel caso in cui gli archi non abbiano pesi).

ROBOMAZE

- Factory Method
- State
- Strategy
- Observer
- Command
- Proxy

FACTORY METHOD

Il pattern è stato utilizzato al fine di creare il labirinto corretto, nascondendo di fatto la creazione dello stesso e creando una vera e propria interfaccia tra la richiesta dell'oggetto e la sua effettiva realizzazione. In futuro sarà possibile, senza troppi patemi, aggiungere altri livelli semplicemente estendendo ulteriormente sia la classe Labyrinth che la classe LabCreator.

STATE

Questo pattern ha lo scopo di rappresentare i comportamenti del Robot, del tutto identici a quelli di un automa a stati finiti. La scelta è stata quasi doverosa, per via del ruolo che il pattern assume, e per come sembra calzare a pennello per il nostro gioco.

STRATEGY

Utilizzato all'interno di ogni State, Strategy definisce la strategia di movimento del Robot in base allo stato, tramite l'utilizzo di tre diverse strategie:

AntColonyOneCell (movimento con ACO di 1 cella - PursuitState/SeekState)

AntColonyTwoCells (movimento con ACO di al piu' 2 celle - FleeState)

RandomMove (movimento casuale in una delle possibili celle adiacenti - EvadeState).

OBSERVER

Il pattern Observer è stato usato per aggiornare la rappresentazione visiva della scena sull'effettivo stato del labirinto. In particolare, la classe GameSub si occupa di incapsulare i dati relativi allo stato del labirinto e del robot, che le arrivano grazie alla funzione notifySubscribers chiamata dalla classe Game.

COMMAND

Tale pattern è stato fondamentale per parametrizzare la richiesta inviata dall'utente tramite pressione di un bottone (scelta della difficoltà del labirinto - easy/medium/hard) e farla arrivare sino alla classe LabController, che si occupa della rappresentazione visiva del labirinto.

PROXY

L'ultimo pattern utilizzato, Proxy, è stato scelto per creare un layer tra il client (l'applicazione) e i files (le scoreboard), controllando a tutti gli effetti l'accesso ad esse tramite funzioni preimpostate dall'interfaccia IFile (read e write, rispettivamente).