Mattia Vicenzotto *5CIA, 8/2/2020*

Android App Development

Technical Specification

Abstract

L'applicazione da me sviluppata è il gioco "Tic Tac Toe" (in italiano tris).
Oltre alla classica versione per 2 giocatori, nell'applicazione è possibile scegliere diverse modalità di gioco, tra le quali: "easy mode", "medium mode", "impossible mode".

General Technical Data

L'applicazione è stata realizzata con Flutter, un framework open-source creato da Google per la creazione di interfacce native per iOS e Android.
È un framework recente (maggio 2017). L'ho scelto perché lo avevo già usato in 3° superiore per realizzare un'altra app.

A differenza dello sviluppo per app Android "nativo", flutter crea applicazioni compatibili sia per Android sia per IOS. Il linguaggio di programmazione è Dart.

Per la mia applicazione ho utilizzato widget messi a disposizione dalla libreria di flutter "package:flutter/material.dart"(https://api.flutter.dev/flutter/material/material-library.html).

Oltre a questi widget ho implementato algoritmi di minimax per la gestione delle mosse da parte dei bot.

L'icona dell'applicazione e le immagini presenti nell'applicazione sono state create utilizzando il seguente sito: https://www.designer.io/en/

Ho voluto sviluppare l'applicazione su un'activity sola per facilitarne l'utilizzo e la lettura del codice.

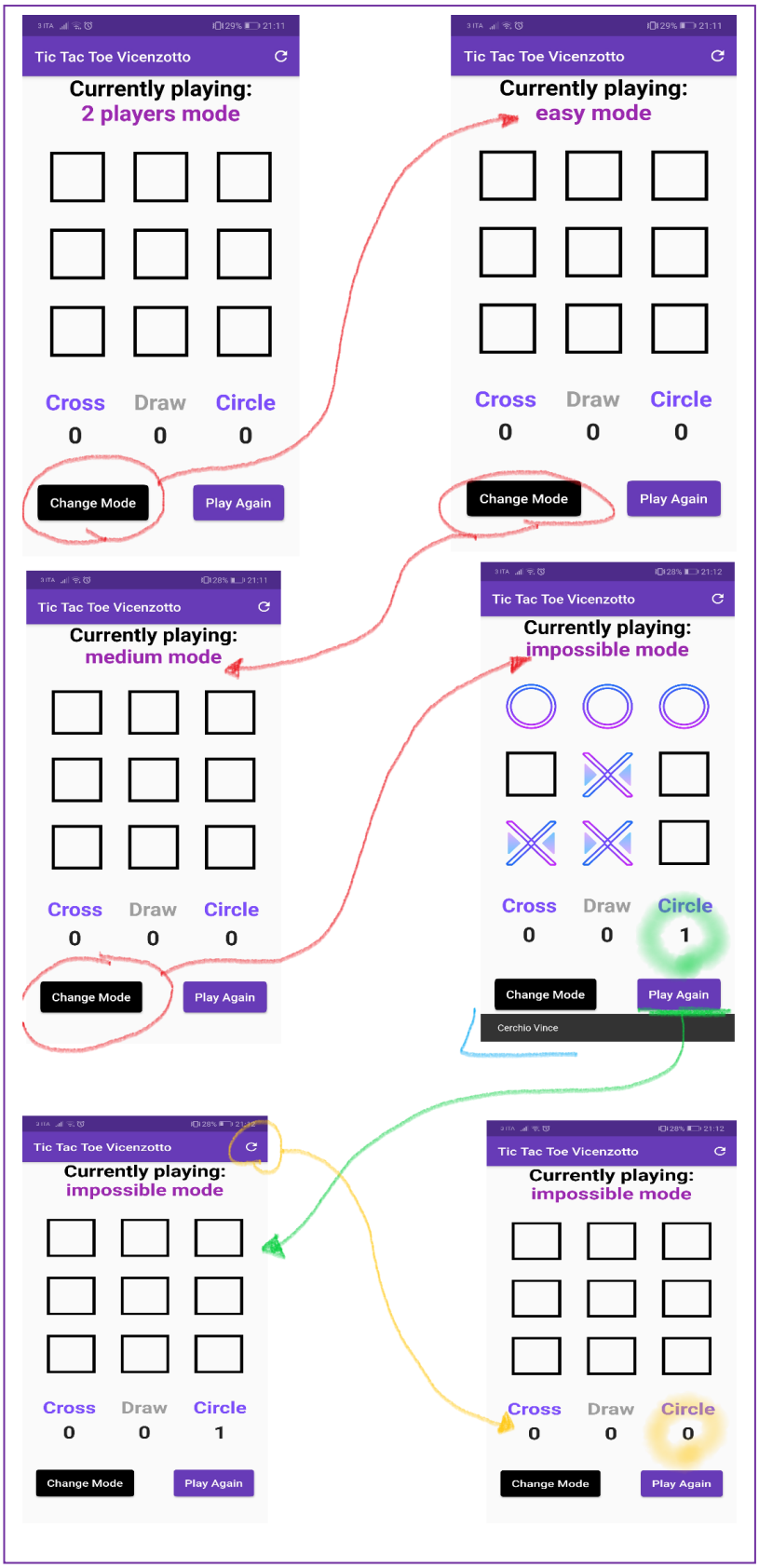
Key Features

La feature principale della mia applicazione è l'utilizzo di un'intelligenza artificiale nelle modalità "easy" ed "impossible" del gioco.

Per l'implementazione ho utilizzato l'algoritmo di minimax, scritto in dart per l'occasione dal sottoscritto.

App structure overview

Wireframe



Funzionamento applicazione

- Tramite il pulsante "Change Mode" è possibile cambiare modalità. Le modalità si susseguono nel seguente ordine:
  -2 players mode
  -easy mode
  -medium mode
  -impossible mode
  (se viene premuto ancora si ritorna alla "2 players mode")
- Quando un giocatore oppure un bot vince appare una "snackbar" che annuncia il vincitore ed il punteggio viene aggiornato.
- Premendo il pulsante "play again" la board di gioco viene pulita.
- È possibile resettare il punteggio premendo l'icona in alto a destra.

Code fragments

Minimax algorithm

pseudocodice di algoritmo minimax:

```
function minimax(nodo)
    SE nodo è un nodo terminale
        return il valore euristico del nodo
    SE l'avversario deve giocare
        α := +∞
        PER OGNI figlio di nodo
            α := min(α, minimax(figlio))
    ALTRIMENTI dobbiamo giocare noi
        α := -∞
        PER OGNI figlio di nodo
            α := max(α, minimax(figlio))
    return α
```

IL MIO CODICE:

-*funzione chiamante*

LA FUNZIONE SI CHIAMA "BESTMOVE" E RESTITUISCE LA MIGLIORE MOSSA POSSIBILE DATA UNA CONFIGURAZIONE DI GIOCO

```
static List<int> bestMove(List<List<int>> boardCopy) {
  //variabili di calcolo
  List<int> botTempCords;
  int tempScore;
  int player = data.isCross;

  //risultati
  int bestScore;        //miglior punteggio ottenibile
  List<int> bestCords;  //coordinate che portano al miglior punteggio ottenibile

  //genero tutte le possibili configurazioni future possibili della board
  for (int i = 0; i < 9; i++) {
    botTempCords = utility.convertToCords(i);
    if (boardCopy[botTempCords[0]][botTempCords[1]] == 0) {
      List<List<int>> botBoard; //creazione di una board di copia
      botBoard = [
        [boardCopy[0][0], boardCopy[0][1], boardCopy[0][2], boardCopy[0][3]],
        [boardCopy[1][0], boardCopy[1][1], boardCopy[1][2], boardCopy[1][3]],
        [boardCopy[2][0], boardCopy[2][1], boardCopy[2][2], boardCopy[2][3]],
        [boardCopy[3][0], boardCopy[3][1], boardCopy[3][2], boardCopy[3][3]]
      ];
      botBoard = utility.insertInBoard(botBoard, botTempCords, player);
      if(checkScore(botBoard) == player)  //se si vince con la mossa appena effettuata le coordinate attuali sono le migliori
      {return botTempCords;}
      //l'algoritmo di minimax restituisce l'esito della partita
      //l'algoritmo sceglie sempre la mossa migliore per il player e per il bot
      //in questo modo gli è possibile sapere come finirà una partita
      tempScore = minimax(botBoard, false, player);
      if (bestScore == null) {
        bestScore = tempScore;
        bestCords = botTempCords;
      }
      if (tempScore < bestScore && player == -1){
        bestScore = tempScore;
        bestCords = botTempCords;
      }
      if (tempScore > bestScore && player == 1){
        bestScore = tempScore;
        bestCords = botTempCords;
      }
    }
  }
  return bestCords;
}
```

-*algoritmo di minimax*

LA FUNZIONE SI CHIAMA "BESTMOVE" E RESTITUISCE LA MIGLIORE MOSSA POSSIBILE DATA UNA CONFIGURAZIONE DI GIOCO

```
static int minimax(List<List<int>> boardCopy, bool isMaximizing, int pValue) {
  List<int> botTempCords;
  int score = checkScore(boardCopy); // 1 = player | -1 = bot | 0 = pareggio | -100 = non concluso
  int tempScore = 0;
  if (score != -100) {
    return score;
  }

  isMaximizing = !isMaximizing; //cambio il massimizzante in minimizzante e viceversa
  pValue *= -1; //cambio giocatore

  for (int i = 0; i < 9; i++) {
    botTempCords = utility.convertToCords(i);
    if (boardCopy[botTempCords[0]][botTempCords[1]] == 0) {
      //creare copie della board è necessario
      //se si passa la lista di base come valore di una funzione viene passata per indirizzo
      //creando una copia identica il problema è risolto
      List<List<int>> botBoard;
      botBoard = [
        [boardCopy[0][0], boardCopy[0][1], boardCopy[0][2], boardCopy[0][3]],
        [boardCopy[1][0], boardCopy[1][1], boardCopy[1][2], boardCopy[1][3]],
        [boardCopy[2][0], boardCopy[2][1], boardCopy[2][2], boardCopy[2][3]],
        [boardCopy[3][0], boardCopy[3][1], boardCopy[3][2], boardCopy[3][3]]
      ];

      botBoard = utility.insertInBoard(botBoard, botTempCords, pValue);
      tempScore = minimax(botBoard, isMaximizing, pValue);

      if (score == -100) {
        score = tempScore;
      } else {
        if (isMaximizing)
          score = max(score, tempScore);
        else
          score = min(score, tempScore);
      }
    }
  }
  return score;
}
```

-*funzione insertInBoard*

PRENDE IN INGRESSO UNA BOARD DI GIOCO, DELLE COORDINATE ED IL VALORE DEL PLAYER ATTUALE

```
static List<List<int>> insertInBoard(List<List<int>> boardCopy, List<int> myTempCords, int pValue) {
  boardCopy[myTempCords[0]][myTempCords[1]] = pValue; //inserisco il valore sulla board
  boardCopy[myTempCords[0]][3] += pValue; //win righe
  boardCopy[3][myTempCords[1]] += pValue; //win colonne
  if (myTempCords[0] == myTempCords[1]) boardCopy[3][3] += pValue; //diagonale
  return boardCopy;
}
```

-*funzione checkScore*

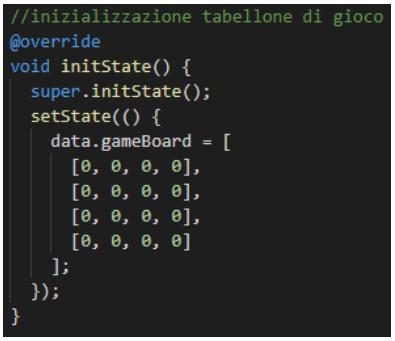
RITORNA IL PUNTEGGIO DELLA PARTITA IN CORSO

```
//ritorna il punteggio --> 1 = vince croce | -1 = vince bot | 0 = pareggio | -100 = non concluso
static int checkScore(List<List<int>> myBoard) {
  bool tie = true;
  for (int i = 0; i < data.side; i++) {
    for (int j = 0; j < data.side; j++) {
      if (myBoard[i][j] == 0) tie = false;
    }
  }

for(int i = 0; i<data.side; i++)
{
  if (myBoard[i][data.side].abs() > 2) { if (myBoard[i][data.side] > 0) { return 1;} else {return -1;}}
  else if (myBoard[data.side][i].abs() > 2) { if (myBoard[data.side][i] > 0) { return 1;} else {return -1;}}
}

if (myBoard[data.side][data.side].abs() > 2) {if (myBoard[data.side][data.side] > 0) {return 1;} else {return -1;}}
else if (myBoard[0][2] == 1 && myBoard[1][1] == 1 && myBoard[2][0] == 1) {return 1;}
else if (myBoard[0][2] == -1 && myBoard[1][1] == -1 && myBoard[2][0] == -1) {return -1;}
else if (tie == true) { return 0;}
else {return -100;}
}
```

Inizializzazione board:



```dart
//inizializzazione tabellone di gioco
@override
void initState() {
  super.initState();
  setState(() {
    data.gameBoard = [
      [0, 0, 0, 0],
      [0, 0, 0, 0],
      [0, 0, 0, 0],
      [0, 0, 0, 0]
    ];
  });
}
```

Development

Target API level: 28

Minimum API level: 16

IDE: vsCode (flutter)

Man-hours: 40

Problems and difficulties

Durante il periodo di sviluppo ero impegnato nello studio scolastico ed impegni personali. Avrei voluto dedicare più ore al progetto.

Reported Bugs

/

Further development

Può essere implementata una modalità per giocare online, l'opzione di iniziare per secondi. Può essere migliorata la grafica ed aggiunti degli effetti quando un giocatore vince.

Self-rating

4 Stars

★★★★

References

Libreria per il material design:
https://api.flutter.dev/flutter/material/material-library.html

Tic Tac Toe AI with Minimax Algorithm (utilizzando p5.js)
https://youtu.be/trKjYdBASyQ

Immagini/icona applicazione:
https://www.designer.io/en/

IDE: https://code.visualstudio.com/
flutter installation/configuration: https://flutter.dev/docs/get-started/install/windows

Come trovare il vincitore nel tris (complessità computazionale minore possibile)
https://stackoverflow.com/questions/4198955/how-to-find-the-winner-of-a-tic-tac-toe-game-of-any-size/34478665