Forza4

Giacomo Di Loreto, Mattia Giordano

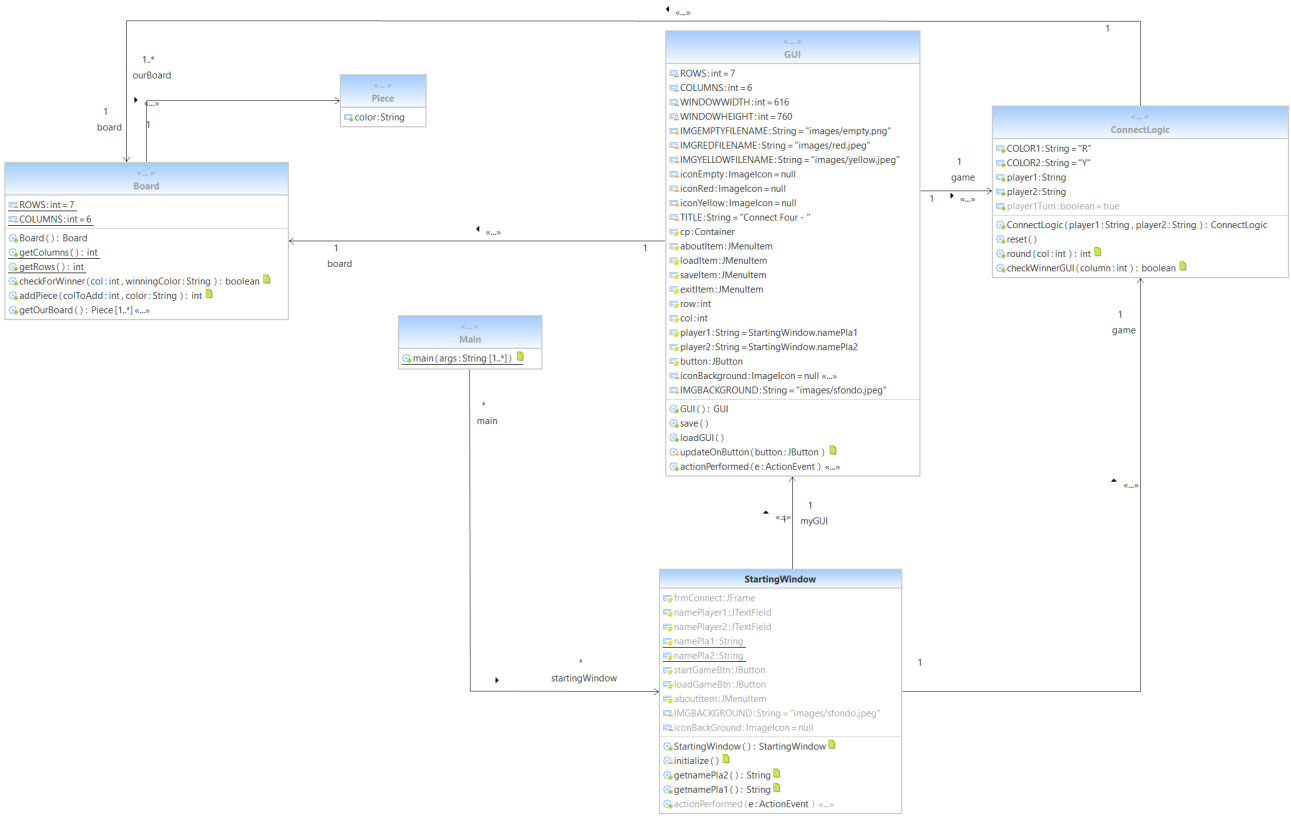
9 febbraio 2023

Indice

1 Introduzione

Per gettare le basi di questo progetto abbiamo iniziato cercando dei giochi di Forza4 online per capire le funzionalità' fondamentali e quelle che invece avremmo potuto implementare. In una prima realizzazione del gioco siamo partiti dalla GUI cercando di disegnare con il metodo paintComponent la griglia e le pedine; ci veniva però difficile e scomodo aggiornare ad ogni mossa la griglia. Inoltre partendo dalla GUI abbiamo realizzato dopo la logica e questo è stato un approccio secondo noi sbagliato, in quanto ogni errore era da ricercare sia nella GUI che nella logica. Abbiamo optato quindi, dopo vari tentativi, di cambiare approccio ricominciando da capo. Questa volta siamo partiti cercando di realizzare il gioco funzionante sulla console, sviluppando la logica del gioco, suddividendo i vari oggetti in classi, per poi sviluppare l'interfaccia grafica in un secondo momento. Dopo vari aggiustamenti, avevamo ora un punto di partenza solido per sviluppare la GUI: a questo punto però abbiamo deciso di non utilizzare il metodo paintComponent e la classe Graphics e Graphics2D, per i problemi avuti con la prima bozza di progetto. Un approccio più semplice e con cui avevamo più confidenza lo abbiamo trovato utilizzando una griglia di bottoni. Inoltre e' stata creata una pagina di start dove gli utenti devono inserire il nome dei due giocatori e poi iniziare la partita e avviare quindi il gioco. Le immagini utilizzate sono tutte CC. Nello sviluppo delle classi ci siamo divisi il lavoro per cui Giacomo Di Loreto ha sviluppato le classi StartingWindow, Piece e ConnectLogic, Mattia Giordano ha sviluppato le classi Main, GUI e Board, naturalmente poichè è stato usato il polimorfismo c'è stata collaborazione tra noi nella creazione di tutte le classi. Fondamentale per la collaborazione tra di noi è stato l'utilizzo di GitHub, il quale ci ha permesso di lavorare sulle stesse versioni del codice contemporaneamente. Abbiamo personalizzato il gioco applicando il logo della nostra universita' sulle pedine.
Confrontandoci alla fine del progetto, entrambi concordiamo sull'utilità del progetto nel porci di fronte a piccole sfide grazie alle quali siamo riusciti a trovare sempre metodi efficaci per risolverle anche quando all'inizio ci sembrava molto difficile.

2 Diagramma UML



3 Le classi

Per la creazione delle classi abbiamo definito ogni oggetto con una classe. Le classi utilizzate sono le seguenti: Main (fa partire e da inizio alla creazione del gioco); StartingWindow crea la schermata iniziale, fa partire la classe GUI, crea la schermata del gioco vero e proprio che a sua volta genera la logica alla base del gioco, ossia la classe ConnectLogic. Questa fa anche utilizzo della classe Board (ossia della griglia), e della classe Piece (ossia delle pedine utilizzate). Tutte queste classi sono racchiuse nel pacchetto connectFour.

3.1 Main

La classe Main è la classe che funge da launcher e creerà il gioco con effetto a catena: difatti fa partire la classe StartingWindow. Naturalmente questa è l'unica classe dotata di un metodo main.

3.2 StartingWindow

La classe StartingWindow è un'interfaccia grafica che servirà da launcher, la quale fa ampiamente uso del pacchetto Swing. In questa classe sono definiti un frame, dei campi di testo, delle etichette, dei bottoni, una barra dei menù, e delle immagini. Per semplicità e rapidità questa classe è stata costruita con il plug-in WindowBuilder, che da anche la possibilità di vedere e posizionare direttamente sul frame i vari oggetti.

3.2.1 Attributi

- frmConnect: frame dove e' costruita la schermata iniziale
- namePlayer1: campo dove inserire il nome del primo giocatore
- namePlayer2: campo dove inserire il nome del secondo giocatore
- NamePla1: nome del primo giocatore
- namePla2: nome del secondo giocatore
- startGameBtn: bottone che consente di cominciare una partita, dopo aver inserito i nomi di entrambi i giocatori
- aboutItem:
- IMGBACKGROUND: immagine di sfondo della schermata iniziale
- IconBackGround: icona della finestra

3.2.2 I metodi della classe

1. StartingWindow():

   Costruttore utilizzato per far partire la schermata iniziale

2. initialize():

   Metodo che costruisce la schermata iniziale sul frame frmConnect

3. getNamePla1():

   Ritorna il nome inserito nel campo di testo del giocatore 1

4. getNamePla2():

   Ritorna il nome inserito nel campo di testo del giocatore 2

5. actionPerformed(ActionEvent e):

   Metodo che assegna dei comandi a determinate azioni

3.3 GUI

La classe GUI, che e' una estensione di JFrame (osservando quindi il principio di ereditarietà) e' il cuore del progetto. Oltre a contenere il metodo per la creazione della griglia di gioco composta da bottoni (senza il quale non esisterebbe il "campo" da gioco), regola tutte le azioni dei giocatori: in base al click del mouse del giocatore, ci sono delle funzioni che posizionano le pedine nella colonna selezionata. Nella classe GUI, inoltre, viene impostata la barra del menu' con i suoi bottoni, potendo modificare le azioni che l'applicativo compie dopo averli premuti. Con la classe URL (importata attraverso java.net.URL) creiamo degli oggetti di questo tipo, attraverso i quali vengono posizionate le pedine dei giocatori semplicemente sostituendo l'immagine della "pedina bianca" (quindi una posizione vuota) con quella della pedina del colore corrispondente al turno del giocatore. I vari errori che possono presentarsi durante la partita (colonna piena, griglia piena) sono controllati da questa classe. Anche l'esito della partita viene dato richiamando una funzione dalla classe ConnectLogic.

3.3.1 Attributi

- ROWS: le righe della griglia
- COLUMNS: le colonne della griglia
- WINDOWWIDHT: larghezza finestra
- WINDOWHEIGHT: altezza finestra
- IMGEMPTYFILENAME: immagine tassello vuoto
- IMGREDFILENAME: immagine tassello rosso
- IMGYELLOWNAME: immagine tassello giallo
- imgBackGround: immagine background
- TITLE: titolo
- iconBackground: immagine icona finestra
- iconEmpty: icona pedina vuota
- iconRed: icona pedina rossa
- iconYellow: icona pedina gialla
- cp: Container
- aboutItem: tasto about del menùbar
- loadItem: tasto load del menùbar
- saveItem: tasto save del menùbar
- exitItem: tasto exit del menùbar
- player1: nome del primo giocatore
- player2: nome secondo giocatore
- game: logica di gioco

Abbiamo poi il costruttore GUI che crea un'istanza del gioco, ossia un oggetto chiamato game di tipo ConnectLogic, passandogli in input i nomi dei giocatori inseriti precedentemente nei JTextField in StartingWindow. Viene poi creato il menù bar e inserite le immagini. Successivamente creo l'array di bottoni che avranno come immagine una casella vuota. A questi bottoni è stato poi aggiunto un ActionListener che ad ogni click li aggiornerà usando il metodo updateOnButton che verrà spiegato in seguito.

3.3.2 I metodi della classe

1. updateOnButton(JButton button):

   Prende in input un bottone e calcola di quale bottone si tratta nella griglia. Usando il metodo getPlayer1Turn della classe ConnectLogic, vede se tocca al primo o al secondo giocatore, così da settare il titolo nella finestra del frame in base al turno a meno di errori questo metodo, aggiorna il bottone selezionato, seguendo anche il turno del giocatore e quindi impostando la pedina del colore corretto. Ad ogni inserimento di pedina controlla se c'è un vincitore con il metodo checkWinnerGUI della classe Board, per aprire poi un pannello in caso di vittoria.

2. actionPerformed(ActionEvent e):

   Questo metodo esegue delle azioni solo al presentarsi di un evento, in questo caso la pressione dei tasti del menùbar della GUI

3. save():

   Questo metodo salva lo stato attuale della GUI alla pressione del tasto save tramite serializzazione, creando un file chiamato "GUI.ser". Questo è possibile grazie alle classi FileOutputStream e ObjectOutputStream le quali salveranno su file lo stream di byte dello stato attuale della GUI. Successivamente sarà stampato "Saved" in console.

4. loadGUI():

   Questo metodo esegue il caricamento di una partita precedentemente salvata andando a leggere il file di salvataggio chiamato "GUI.ser". Questo è possibile tramite deserializzazione, in particolare con le classi FileInputStream e ObjectInputStream le quali andranno a trasformare un file contentente uno strem di byte, nello strem di byte stesso, così facendo verrà rispistinato lo stato della GUI precedentemente salvato. In caso di errori verrà stampato in console "Class not found".

3.4 ConnectLogic

Questa classe regola la logica del gioco. Vengono dichiarate le variabili che verranno poi utilizzate dalle altre classi, come quelle dei giocatori. Contiene il metodo reset, che serve a resettare una partita per cominciarne un'altra (ma solo se si gioca da console, altrimenti il metodo di reset per la GUI e' implementato nella classe GUI), e i metodi di check della vittoria (uno per la vittoria in console, l'altra per la vittoria in GUI), opportunamente contraddistinti.

3.4.1 Attributi

- BOARD: oggetto di tipo board
- PLAYER1: nome del primo giocatore
- PLAYER2: nome del secondo giocatore
- COLOR1: colore del primo giocatore
- COLOR2: colore del secondo giocatore
- player1Turn: variabile booleana che segna se il turno e' del giocatore 1

3.4.2 I metodi della classe

1. ConnectLogic:

   costruttore che prende in input i nomi dei giocatori e costruire un oggetto chiamato "board" di tipo Board

2. reset():

   Metodo che resetta la partita costruendo un nuovo oggetto chiamato "board" di tipo Board

3. getPlayer1Turn():

   Ritorna il valore della variabile privata player1Turn

4. round(int col):

   Metodo che regola i turni dei giocatori basandosi sul valore della variabile player1Turn

5. checkWinnerGUI(int column):

   Metodo che ritorna la colonna e il colore del vincitore passandoli in input al metodo checkForWinner nella classe Board

3.5 Board

Come si evince dal nome, questa classe crea a livello logico la griglia di gioco, formata da un array di tipo Piece, il quale viene inizialmente inizializzato a null dal costruttore. In questa classe viene effettuato anche il controllo a livello logico dei vincitori e inoltre viene aggiunta una pedina nell'array logico ogni qualvolta il giocatore inserisce una pedina nella GUI, facendo anche controlli se la colonna è piena o se la pedina non si può mettere in un determinato punto.

3.5.1 Attributi

- ROWS: righe della griglia logica
- COLUMNS: colonne della griglia logica
- ourBoard: griglia, ossia un array, di oggetti di tipo pedine

3.5.2 I metodi della classe

1. checkForWinner(int col, String winningColor):

   Questo metodo prende in input il numero della colonna e il colore del vincitore e controlla nelle varie direzioni, ossia in alto, in basso, a destra, a sinistra e sulle diagonali se c'è una vittoria, ossia 4 pedine di fila.

2. addPiece(int colToAdd, String color):

   Inserisce la pedina nella colonna selezionata dal giocatore e la mette del colore del giocatore. Mi ritorna la riga su cui è stato aggiunto il pezzo, in caso di errore torna -1 e stampa un messaggio di errore.

3.6 Piece

Nella classe Piece e' presente una sola variabile privata, alla quale posso accedere le altre classi attraverso i metodi getter e setter, osservando cosi il principio di incapsulamento.

3.6.1 Attributi

- color: variabile che rappresenta il colore del pezzo

3.6.2 I metodi della classe

1. getColor():

   Ritorna il valore della variabile privata COLOR

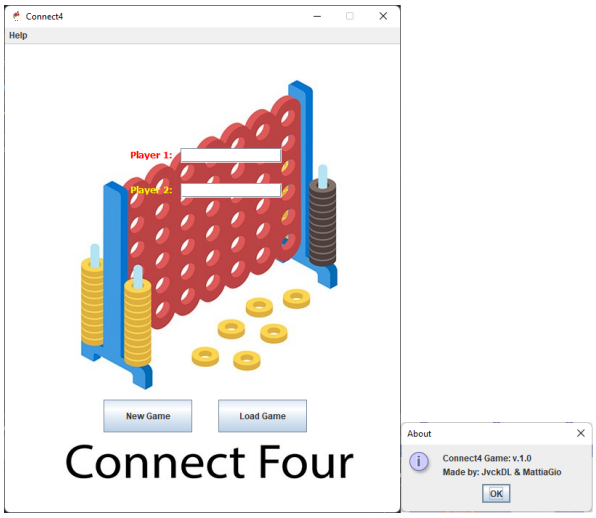
2. setColor():

   Setta il colore della variabile privata COLOR

4 Funzionalità

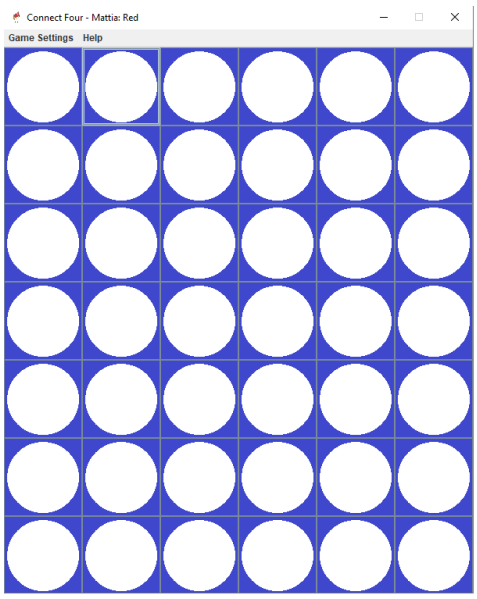
4.1 StartingWindow

In questa finestra troviamo due campi JTextField dove gli utenti devono inserire i propri nomi, che verranno poi ricordati durante i turni e saranno indicati come titolo della finestra. Troviamo anche un tasto "New Game" che quale servirà per iniziare una nuova partita e un tasto "Load Game" per caricare una partita già iniziata. Inoltre è presente anche un menùbar con una sezione "Help" e il tasto "About" che aprirà una scheda con alcune informazioni dell'applicazione.

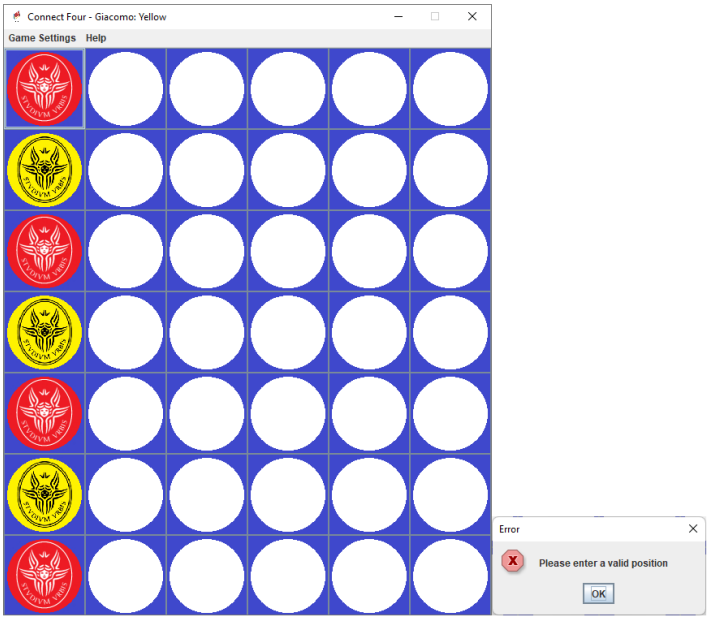


4.2 Connect4-GUI

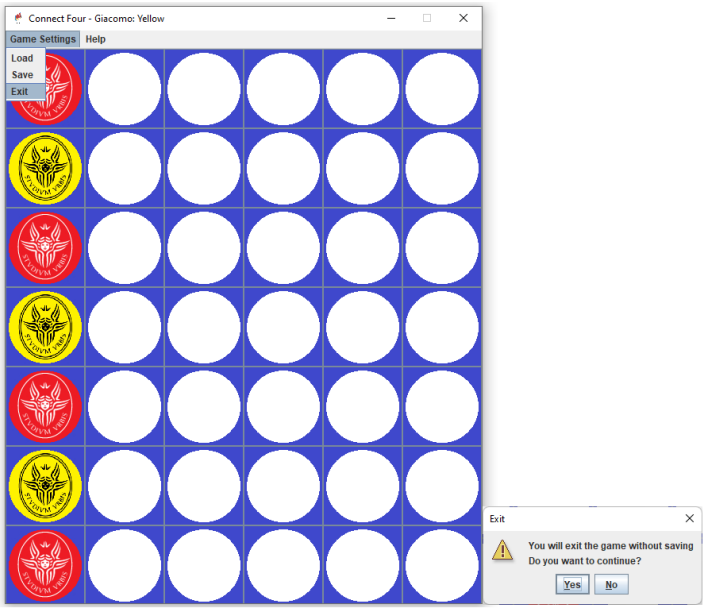
Dopo aver selezionato il bottone "New Game" il launcher si chiuderà e si aprirà la schermata come di seguito in figura.



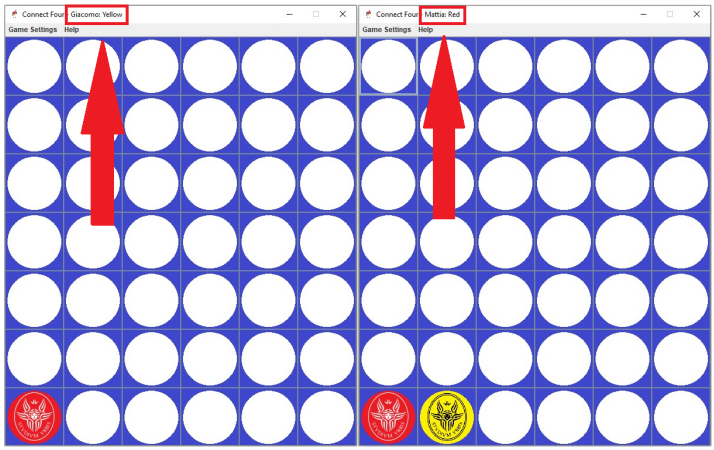
Cliccando su un qualsiasi bottone la pedina scenderà sul fondo, fino a toccare il fondo della griglia o un'altra pedina. In caso si dovesse riempire una colonna intera e l'utente provasse a inserire un'altra pedina, comparirà un messaggio di errore accompagnato da un suono e stampa su linea di comando.



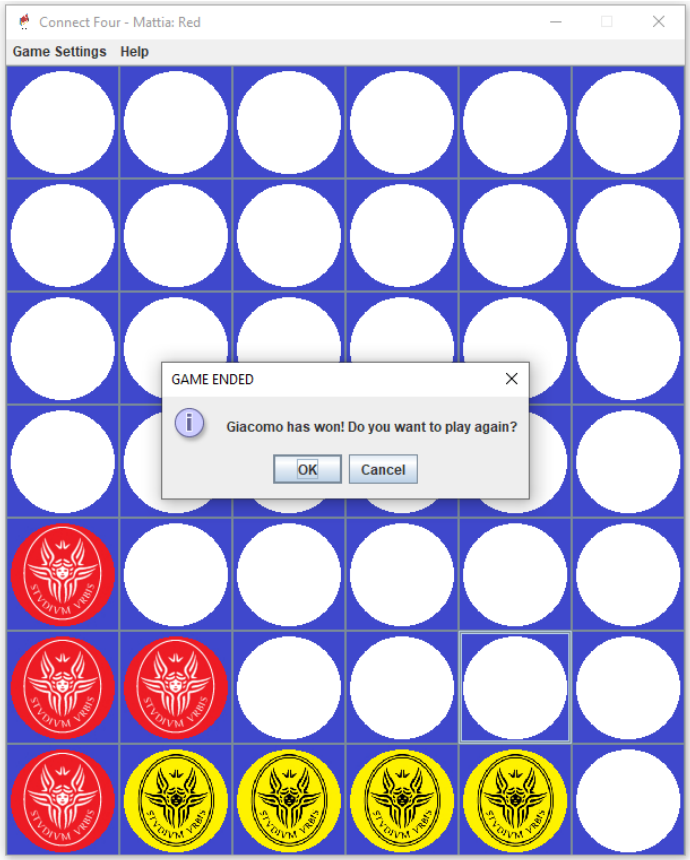
In questa finestra è anche presente l'opzione Game Settings, la quale permette di salvare e caricare le partite oppure di uscire dal gioco, una volta cliccato quest'ultimo messaggio uscirà un pannello ricordandoci di salvare e chiedendo se siamo sicuri di voler uscire.



Ad ogni turno il titolo verrà aggiornato in base al turno del giocatore, il quale indicherà anche il colore a lui corrispondente. Il rosso inizia sempre per primo.

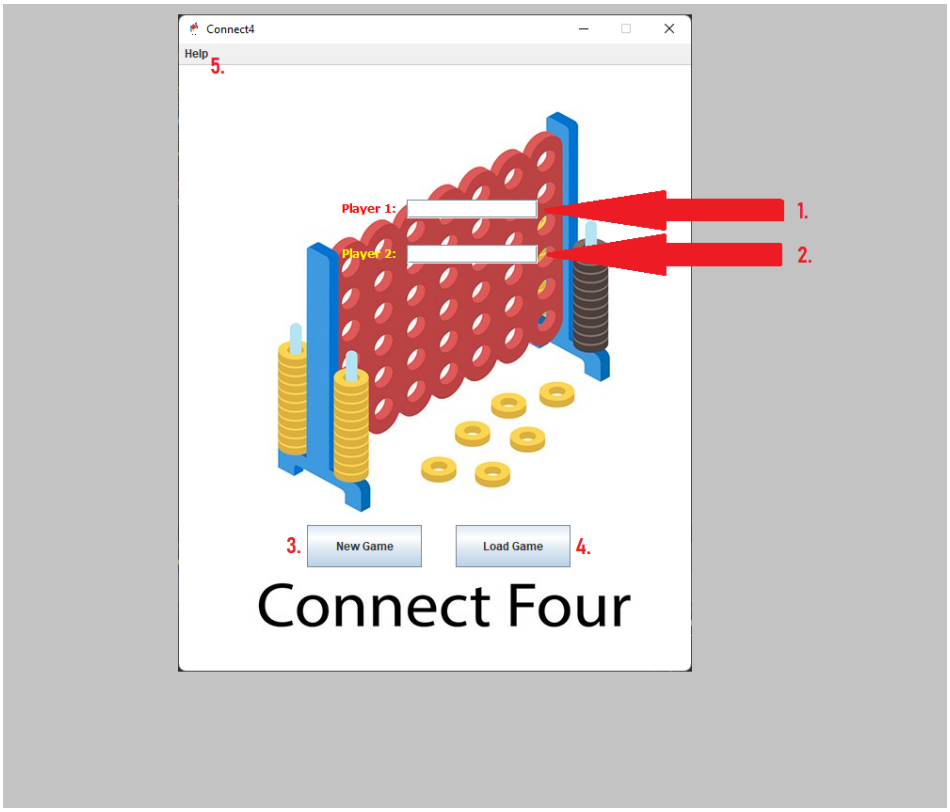


Quando un giocatore vince comparirà un pannello con un messaggio che chiederà se si vuole rigiocare o meno.



5 Manuale della GUI

5.1 Avvio di una partita

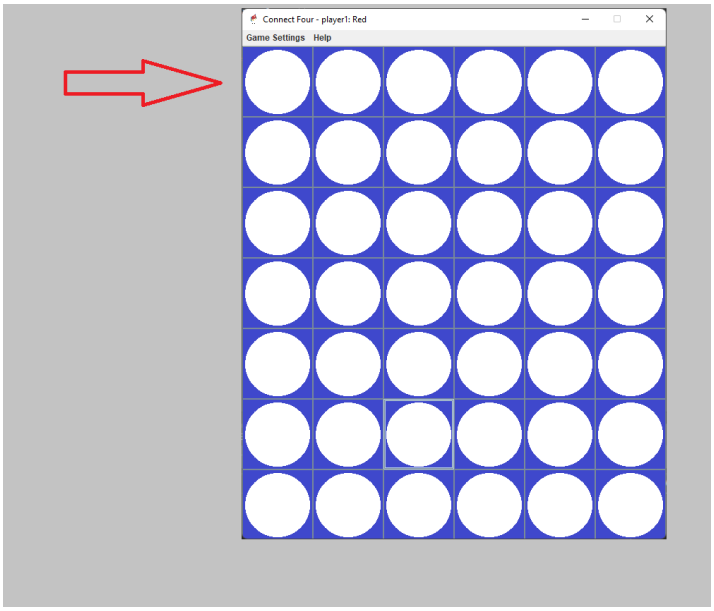


Di seguito verranno illustrati nel dettaglio i passaggi per l'avvio di una nuova partita.

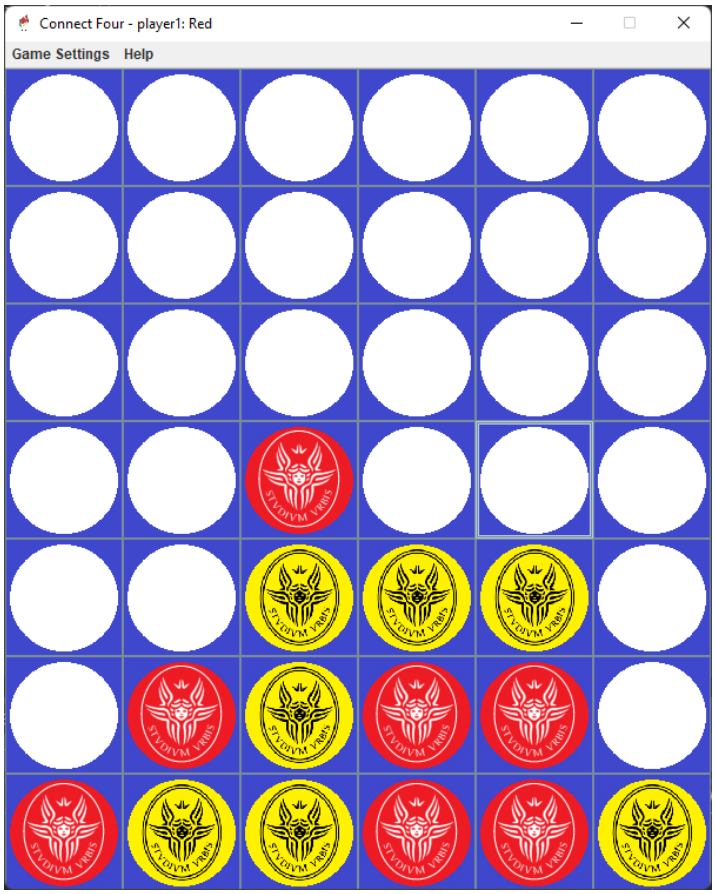
1. Nome nel primo giocatore
2. Nome del secondo giocatore
3. Tasto di avvio della partita
4. Menu

5.2 Posizionamento delle pedine

Una volta inseriti i nomi dei giocatori e cliccato sul tasto "New Game", verrà visualizzata la griglia dove inserire le pedine. Basta cliccare in una qualsiasi riga della colonna desiderata e le pedine scenderanno dall'alto poggiandosi in fondo se la colonna e' vuota, altrimenti sopra alle pedine già presenti.

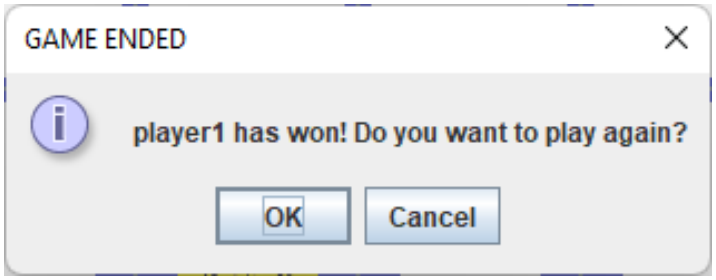


Questo e' il risultato dopo aver posizionato diverse pedine.



5.3 Inizio nuova partita

Una volta che la partita si conclude, comparirà' un messaggio che chiederà' se si vuole iniziare una nuova partita (portano nuovamente alla prima schermata e facendo scrivere i nomi dei giocatori), oppure se uscire e chiudere l'applicativo.



6 Referenti di sviluppo

Entrambi abbiamo seguito reciprocamente il lavoro, anche per avere una maggiore coesione tra le classi e i metodi necessari. Nello specifico Giacomo Di Loreto(1845290) si è concentrato maggiormente sulle classi: StartingWindow, Piece e ConnectLogic. In particolare per la classe StartingWindow si intende il launcher del gioco, mentre le classi Piece e ConnectLogic la logica del gioco. Mattia Giordano(1884283) si è focalizzato sulle classi: Main, Board e GUI. In particolare la classe Main è la classe che una volta eseguita da inizio al gioco, mentre la classe GUI è la classe che crea l'interfaccia grafica dove l'utente interagira' con il gioco. La classe Board infine riporta a livello logico quello che avviene sulla GUI.