Programmazione

Gioco dell’oca pazza

Contents

I

GOP

Chapter 1

Regole del gioco

Il gioco dell'oca è gioco da tavolo nel quale si segue un percorso estremamente semplice. Il movimento della pedina è deciso dal lancio casuale di due dadi.

Il numero minimo di giocatori è 2, inizia il giocatore più giovane di età.
Lo scopo del gioco è di arrivare per primo alla fine del tabellone senza superare l'ultima casella.

1.1 Gioco dell'oca Pazza

Il nostro è un gioco dell'oca pazza interattivo scritto in c++, seguirà delle regole un po' particolari diverse da solito gioco dell'oca.

1.1.1 Menù

Il menù iniziale è composto da "`Nuova Partita`" , "`Opzioni`" e "`Esci dal gioco`".
Per interagire nel menù iniziale basta che si inseriscono i numeri associati ad ogni funzione. Se viene inserito un altro carattere che non è presente nelle possibili scelte da fare, il sistema restituirà un errore e darà un altra possibilità di inserimento.

1.1.2 Nuova Partita

Nuova partita sarà la nostra prima scelta del menù. Se si inserisce 1 il programma farà iniziare il gioco con le opzioni di default. Per modificare le opzioni di default basta andare nella sezione opzioni e cambiare le modalità di gioco.

Appena si inizia una nuova partita il programma chiede il numero di giocatori che prenderanno parte al gioco. Se non viene inserito un numero valido il numero di giocatori sarà impostato automaticamente a 2.

Successivamente verrà richiesto il nome di ogni giocatore e la sua età di modo da far iniziare il più giovane per primo e poi il resto in ordine crescente d'età.

Infine verrà stampata la mappa del gioco, e le informazioni di log relative all'identità, alla posizione di ogni giocatore e un segnalino ad indicare il turno. Il movimento dei giocatori è ben scandito con il riempimento delle caselle, vince il giocatore che arriva per primo alla fine senza superare la fine. Se un giocatore arriva all'ultima casella superandola, tornerà indietro del numero di caselle in eccesso.

1.1.3 Opzioni

Per entrare nella selezione "`Opzioni`" basterà inserire il numero 2.

Le impostazioni di Default sono le seguenti:
- numero di giocatori: 2 ;
- modalità di gioco: facile ;

Se si vogliono modificare le opzioni basterà entrare all'interno di esse.
Il menù opzioni sarà impostato in questo modo:
- Difficoltà
- Regole
- Credits

- Menù Principale

Esistono invece due tipi di **difficoltà** easy e hard. Per easy basterà inserire 0 invece 1 per hard. Se il valore inserito non sarà consono, rimarrà impostato il valore di default.
Se si sceglie la modalità easy la mappa è di una lunghezza variabile tra 40 e 63 caselle e può essere riempita con caselle vuote tra il 51% e il 65% delle caselle totali, le restanti sono caselle azione generate casualmente.
Nella modalità hard la mappa ha una lunghezza tra 64 e 90 caselle ed è essere riempita allo stesso modo esposto di sopra, tenendo conto che le caselle vuote sono tra il 35% e il 50% per incrementare la difficoltà.
Anche la disposizione delle caselle è generata casualmente.

Diverse saranno le funzioni Regole e Credits. La prima stampa a video un file dove sono inserite tutte le regole del gioco la seconda stampa un file dove sono inserite le informazioni di chi ha sviluppato il gioco.

Selezionando "`Menù Principale`" si torna indietro al menù iniziale. Prima di rivedere il menu principale verranno stampate le opzioni con le eventuali modifiche effettuate.

1.1.4 Esci dal gioco

Inserendo il numero 3 il programma si interrompe e si esce.

Chapter 2

Oggetti del gioco

L'implementazione delle carte e delle caselle speciali è dovuta alla scelta di rendere il gioco più dinamico.

2.1 Mazzo

Le carte sono raggruppate in un mazzo di 40 mischiato casualmente dal codice.
Il mazzo è formato nel modo seguente:
- carte vuote (10);
- avanza di una casella (10);
- salta un turno (5);
- ritira i dadi e vai indietro (5);
- ritira i dati e vai avanti (5);
- manda il giocatore che gioca al prossimo turno alla casella di partenza (5).

2.2 Mappa

La classe mappa viene generata casualmente in base alla modalità di gioco selezionata.
I tipi di caselle sono le seguenti:
- start (casella iniziale);
- bridge (ripete il movimento);
- inn (fermo 1 turno);
- prison (fermo 3 turni);
- labirinth (torna alla casella in cui eri all'inizio del turno);
- skull (torna a start);
- end (casella finale).

II

Il codice

Chapter 3

Il codice scritto è composto da diverse parti, ogni una delle quali gestisce un modulo del programma.

`main`

All'interno del main inizialmente viene creato il menù sul quale vengono fatte le scelte.
Se viene scelta "`Nuova Partita`" viene istanziato un oggetto `game` che gestirà i giocatori il mazzo la mappa e l'esecuzione del gioco.

Il `while` presente cicla finché non si esce da gioco.
Quando la partita finisce si torna nel menù principale, quindi l'unico modo per uscire dal gioco è quella di selezionare "`Esci dal gioco`".

3.1 Menù

Il `menu` è una classe.
Nella sezione protetta abbiamo:
- due variabili intere `Player_n` (che sarà il numero di giocatori) e `x` (che gestisce la scelta all'interno del menù);
- una variabile di tipo bool `Mode` (gestisce la difficolta del gioco);
- una variabile ifstream `file_in` che servirà per la gestione dei file in input;
- una variabile di tipo char "`parser`" che serve per la scansione delle righe dei file da stampare.

Nella sezione pubblica abbiamo:
- i due costruttori, uno di default e uno che prende in input i valori che vengono generati tramite l'interfaccia di opzioni;
- metodi set e get per le variabili sopra elencate;
- la funzioni di tipo void come `display` che stampa a video il menù , `choice` che fa scegliere dove spostarsi, `setOptions` che setta le opzioni all'interno del menù e `parseFile` stampa sul terminale i file .txt.

Da questo momento in poi si tenga conto della scelta implementata di gestire gli input utente con `getline` facendo poi una conversione al tipo desiderato. In questo modo gestiamo sia stringhe con più parole sia gli errori causati da input non permessi.

3.1.1 Game

Game è una sottoclasse di menù (eredita tutti i suoi metodi ed attributi).
Inoltre sono presenti tuitti i metodi che gestiscono il mazzo i giocatori, la mappa e anche le esecuzioni del gioco.

Nella sezione protetta abbiamo una matrice (vettore a due dimensioni) di stringhe per la rappresentazione grafica della mappa.

Invece nella sezione pubblica abbiamo:
- puntatore alla mappa;
- puntatore al mazzo;
- puntatore a giocatore.

Infine la classe implementa i seguenti metodi:
- i costruttori:
  - `Game ()` costruttore di default che richiamerà il costruttore della classe madre menù;
  - `Game (int p bool mo)` costruttore con parametri richiamato nel main per inizializzare una nuova partita e che metterà i parametri player e modalità seguendo il costruttore della classe madre menù. Dopodiché provvederà alla costruzione del mazzo della mappa e della lista dei giocatori richiamando gli appositi metodi;
- `createPalyers` crea la lista circolare dei giocatori ordinati per età, a partire dalla sentinella;
- `sortInsert` inserisce un giocatore nella lista con sentinella tenendo conto dell'età (funzione ausiliaria per l'inserimento ordinato dei giocatori in lista);
- `displayPlayers` mostra l'elenco dei giocatori;
- `parseMap` scorre la lista che rappresenta la mappa inserendone nelle celle le varie caselle implementate (la rappresentazione grafica della mappa avrà la forma di una serpentina);
- `printMap` stampa il vettore di stringhe bidimensionale (la rappresentazione della mappa);
- `initMap` inizializza la matrice con `"  "`;
- `isBusy` controllo su casella per verificare se sia occupata o meno;
- `gameStart` gestisce i turni, cicla finché un giocatore non vince, quindi stampa il vincitore;
- `prntLog` stampa il log contenente nome e numero di casella di un giocatore (il giocatore che muove nel turno è indicato con un * stampato a destra).

3.2 Mappa

La mappa è una lista bidirezionale gestita da una classe.
Nella sezione privata abbiamo delle variabili intere `length` (lunghezza mappa), `avg` (la percentuale di caselle vuote) `nEmpty` (numero di caselle vuote).

Nella sezione pubblica:
- due variabili di tipo puntatore `init` (che punta all'inizio della mappa) ed `end` (che punta alla fine della mappa);
- il costruttore `Map` il cui parametro è un booleano che indica la modalità di gioco;
- metodi set e get per le variabili sopra elencate;
- funzioni che prendono come parametro la modalità di gioco seleziona dall'utente, `generateMap` (crea la mappa in base alla modalità selezionata) `calcNBox` (calcola il numero di caselle vuote) e infine `HowEmpty` (decide la percentuale di caselle vuote casualmente);
- una funzione di tipo Box `*genBox` (ritorna un puntatore di tipo Box alla nuova casella anch'essa generata casualmente).

3.2.1 Box

La classe Box serve a generare e gestire gli oggetti casella di cui la mappa è composta.

La sezione protetta della classe contiene nome descrizione (stringhe) e id (intero).
La sezione pubblica invece:
- puntatori `prev` (di tipo Box che punta alla casella precedente) e `next` (di tipo Box che punta alla casella precedente);
- due costruttori, `Box()` di default ed `Box(string n, string d, int id)` parametrico (in ordine sono stati passati nome descrizione ed identificativo);
- metodi set e get per le variabili sopra elencate;
- `display` gestisce la stampa del nome e della descrizione della casella corrente;

Le sottoclassi descrivono le caselle elencate nel paragrafo **2.2**. Ognuna di esse contiene un suo costruttore che richiama il costruttore della sovra classe box grazie al quale imposta i parametri per ogni casella.

3.3 Giocatore

L'elenco dei giocatori è gestito da una lista circolare con sentinella (implementata in game), quest'ultima serve per ordinarli per età dal giovane al più anziano.
Nella sezione privata di giocatore troviamo:
- una variabile di tipo string `name` (nel quale sarà salvato il nome del giocatore);
- quattro variabili di tipo intero, `age` (l'eta del giocatore) , `turn` (utilizzata per gestire il blocco dei turni), `nBox` (memorizza il numero di casella su cui si trova il giocatore) e `d` (memorizza l'ultimo tiro di dado).

Nella sezione pubblica abbiamo:
- due puntatori `next` (di tipo player e gestisce la lista di giocatori) e `position` (di tipo box che indica la posizione corrente per quel giocatore);
- due costruttori di player uno di default e uno che prende come parametro il nome del giocatore, l'età e la posizione corrente all'interno della mappa (inizializzata a start).
- metodi get e set per le variabili sopra elencate;
- i metodi del gioco come `dice` (funzione per il lancio del dado) e `Turn` (gestisce il turno del giocatore corrente);
- il metodo `move (int x, bool v)` che gestisce lo spostamento del giocatore sulla mappa di x posizioni in avanti o indietro a seconda di v;
- `handleCard` che prende come parametro un oggetto Carte che attiva l'effetto della carta passata.

3.4 Mazzo

Il mazzo è una classe che ha come membri privati un array di puntatori a carte di 40 oggetti Carte e un segnalino utilizzato come contatore dai metodi.

In private abbiamo un vettore di puntatori a carte che fanno parte del mazzo e una variabile di tipo intero segnalino che serve come contatore alla carta corrente.

Nel dettaglio, il mazzo, ha in public i seguenti metodi:
- `Mazzo` il costruttore che alloca all'interno del vettore di carte un numero predefinito di puntatori a carte. Ovvero genera un mazzo ordinato;
- `Mischia` è una funzione che simula l'azione di mescolamento delle carte;
- `Pesca` di tipo Carte incrementa il segnalino e ritorna la carta corrente.

3.4.1 Carte

La classe Carte contiene come elementi privati gli attributi della carta `Name`, `Description` e una di tipo intero `Id` (identificatore del tipo di carta).
Per ogni tipo di carta è stata implementata una sottoclasse che eredita i parametri e ridefinisce i costruttori passando come parametri Nome, Descrizione e Id a seconda del tipo di carta.

Chapter 4

Autori

Andrea D'arpa	andrea.darpa@studio.unibo.it	0000803520
Davide Balestra	davide.balestra2@studio.unibo.it	0000789079
Matteo Celani	matteo.celani@studio.unibo.it	0000804303

`https://github.com/BaL97/GOP/`