Relazione di laboratorio di Programmazione

Progetto GOP

Marco Dong - Vladut Emanuel
Cioanca Radu

A.A. 2017/2018

Introduzione

Questo progetto rappresenta un'implementazione semplificata e modificata del Gioco dell'oca.
Le specifiche di base derivano dalla consegna (link).

Principali scelte del gioco

- Giocatori: 1 - 4, questo limite è utile per non prolungare troppo il gioco, è stato anche definito un giocatore per consentire al professore un suo rapido test;
- Nome: ogni giocatore a inizio partita decide il nome che desidera utilizzare. Il nome è limitato, fino a 10 caratteri (spazi e caratteri dopo lo spazio esclusi) per avere una migliore esperienza di gioco;
- Dado: eravamo in dubbio se usare un convenzionale dado da 6 o usare uno variabile da 0 a 12. Alla fine abbiamo optato quello convenzionale per non diversificare troppo dal gioco reale;
- Effetti: per semplicità gli effetti sono uguali sia per le carte che per le caselle.
- Simboli: –▌▐, questi sono i simboli scelti, corrispondenti a fermo, un dado e la modalità di gioco normale. Sono stati presi da Code page 437,perchè supportati dalla libreria <iostream>, quindi anche dal comando "cout".
- Colori: oltre ai simboli, sono stati usati anche i colori, attraverso alcune prove, è stato scelto verde, giallo, magenta e ciano, rispettivamento per rappresentare i giocatori 1, 2, 3 e 4 poiché erano quelli più visibili con lo sfondo nero. Il bianco è il colore di default.
- Mappa: il numero di caselle varia da 40 a 80, principali difficoltà sono state trovate nello stampare la mappa in un modo leggibile. Viene suddivisa in due strati, uno astratto usato per la stampa a video e uno effettivo usato per la gestione della posizione dei giocatori e per gli effetti.
- Mazzo: È composto da diverse classi, effects, card e Deck. Effects contiene tutti gli effetti associati alle carte o alle caselle. Card è l'implementazione di una singola carta, nella quale vengono memorizzate le informazioni della singola carta, es. il numero di carta, l'effetto associato. Deck è utilizzato per

implementare il mazzo e i principali metodi per interagire con le carte, facendo anche eseguire gli effetti.

Principali scelte implementative

- COLORI: è stato usato una sequenza di stringa per il colore

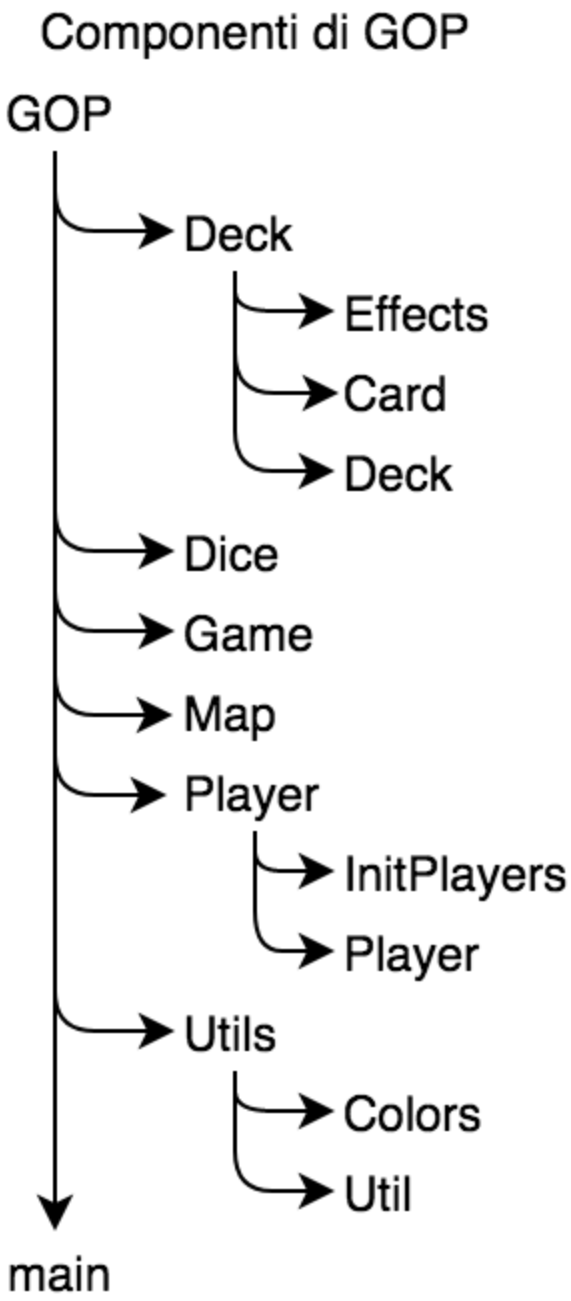
	foreground	***background***
black	***30***	***40***
red	***31***	***41***
green	***32***	***42***
yellow	***33***	***43***
blue	***34***	***44***
magenta	***35***	***45***
cyan	***36***	***46***
white	***37***	***47***

  "\033[*<BOLD>*;*<COLOR>*m", dove "BOLD" viene sostituito con 0 o 1 in base se si vuole il grassetto, e "COLOR" con il numero del colore voluto. Dopodichè qualsiasi cosa che viene mandata in output sarà di quel colore. Per ripristinare l'originale basta scegliere il bianco senza grassetto o con la stringa "\033[0m";

- Giocatore: è stata usata la libreria **<map>** per rappresentare il giocatore, che ha una *key* e un *value.* Essenzialmente il giocatore è una lista indicizzata con numeri interi di strutture, dove nome, casella e colore, per esempio, sono le sue caratteristiche;
- Per l' "animazione" è stato sfruttato la libreria **"unistd.h"** (per il tempo) e **"stdlib.h"** (per la "pulizia"), **usleep**(*<valore in microsecondi>*) e **system("CLEAR")**;
- Per comodità alcuni metodi sono stati implementati in una classe Util in modo static che ne permette una maggiore facilità di utilizzo;

- Per avere una migliore portabilità tra i sistemi Linux e Windows viene definita la variabile environment la quale permette di distinguere tra i diversi S.O. ed utilizzare i comandi adatti per la pulizia della schermata del terminale;

Metodologia di implementazione



Il progetto è stato implementato dividendolo in alcuni componenti più semplici, come ad esempio la mappa, il dado, il mazzo, ... Tali elementi sono stati poi assegnati in modo casuale ai componenti del gruppo, i quali li hanno implementati in modo autonomo e in caso di difficoltà o di idee migliori di implementazione in collaborazione con il collega.
I sotto componenti sono stati trattati come black box dal collega che non li ha implementati, sfruttando il più possibile il paradigma di programmazione ad oggetti, utilizzando metodi e interagendo con essi rimanendo all'oscuro di come siano effettivamente implementati nel dettaglio.