PCD Assignment 03 - Rock, paper, scissors

A cura di

Alessandra Versari - alessandra.versari2@studio.unibo.it

Lorenzo Rigoni - lorenzo.rigoni2@studio.unibo.it

Riccardo Moretti - riccardo.moretti6@studio.unibo.it

Analisi del problema

Il testo richiede di implementare in Go il famoso gioco “Sasso, carta, forbici”. In questa simulazione di gioco, sono presenti tre entità:

- due giocatori che fanno una mossa casuale tra quelle possibili;
- un arbitro che decide chi ha vinto il turno.

Nello specifico, l’arbitro richiede le mosse ai giocatori, i giocatori estraggono casualmente una possibile mossa (sasso, carta o forbici), l’arbitro riceve le mosse e calcola il risultato in base alle regole del gioco (sasso batte forbici, forbici batte carta, carta batte sasso, stessa mossa equivale ad un pareggio) per poi inviarlo ai rispettivi giocatori. Saranno poi i giocatori a stampare il risultato e il loro storico di vittorie e sconfitte aggiornato.

Design, strategia ed architettura

Per implementare questa simulazione, abbiamo utilizzato il linguaggio Go con le sue primitive che implementano lo scambio di messaggi sincroni. Per farlo, abbiamo creato due strutture che rappresentano i tipi di messaggi:

- ***MoveRequest***
- ***Result***

La struttura ***MoveRequest*** contiene al suo interno un campo di tipo *chan string*. Questa scelta è stata fatta per permettere di avere un'unica variabile che rappresentasse sia la richiesta dell'arbitro che la risposta del player. Infatti, il comportamento è il seguente:

1. l'arbitro crea un canale per l'invio della mossa;
2. l'arbitro invia al giocatore il canale incapsulato in una *MoveRequest*;
3. il giocatore calcola la sua mossa e la invia sul canale incapsulato;
4. l'arbitro riceve sul canale la mossa mandata dall'utente.

Ovviamente questi passaggi sono fatti per entrambi i giocatori.

La seconda struttura ***Result*** contiene al suo interno due campi booleani che indicano la vittoria o la sconfitta (se sono entrambi *false* significa che c'è stato un pareggio).

I due tipi di entità sono stati modellati come due funzioni.
Da una parte abbiamo il ***player***, il quale possiede un nome, un canale di richieste e un canale di risultati. Dall'altra abbiamo il ***referee***, il quale ha memorizzato i quattro canali usati dai due giocatori (due di richiesta e due di risultati).

Infine, nella funzione *main* vengono creati i quattro canali dei giocatori e lanciate le tre *goroutine* della simulazione (i due giocatori e l'arbitro). In fondo al main è stato aggiunto anche il comando *select* come "trucchetto" per bloccare il main ed evitare che la simulazione termini.