Progetto

“indovina la posizione della pallina”

Brunasso Vittorio

**Descrizione**

Si vuole realizzare un gioco di strategia in cui due giocatori (che corrispondono al client e al server) si sfidano a nascondere e indovinare la posizione di una piccola pallina all'interno di una griglia 5x5. L'obiettivo principale del gioco è indovinare la posizione della pallina nascosta del tuo avversario mentre cerchi di nascondere la tua in modo astuto.

**Regole del Gioco:**

1. Ogni giocatore ha una griglia 5x5 sulla quale posizionare una pallina nascosta.

2. A turno, i giocatori inseriscono le coordinate per indovinare la posizione della pallina avversaria e per nascondere la propria.

3. Le coordinate valide vanno da 0 a 4 sia per le righe che per le colonne.

4. Il giocatore guadagna punti quando indovina correttamente la posizione della pallina avversaria.

5. Il gioco si svolge in tre round, e il punteggio totale alla fine determina il vincitore.

**Requisiti tecnici**

La scelta del linguaggio di programmazione è ricaduta su Python, che solitamente è una scelta popolare per lo sviluppo di giochi.

Python offre una vasta gamma di librerie e framework che semplificano lo sviluppo di applicazioni. Nel caso specifico, abbiamo utilizzato la libreria “standard socket” per gestire la comunicazione tra il client e il server, e “pickle” per serializzare e deserializzare i dati.

**Per quanto riguarda C**

Se avessi scelto di implementare il progetto in C anziché in Python, avrei dovuto affrontare alcune differenze chiave:

-Gestione della Memoria: In C, avrei dovuto gestire manualmente l'allocazione e la deallocazione della memoria per vari oggetti, come stringhe e strutture dati. Questo comporta una maggiore responsabilità e attenzione per evitare errori di gestione della memoria.

Stringhe: La gestione delle stringhe in C è più dettagliata rispetto a Python. Python gestisce automaticamente la memoria per le stringhe, mentre C utilizza le funzioni della libreria standard di C, come strcpy e strcat.

Socket Programming: L'implementazione delle comunicazioni socket in C richiederebbe l'uso di funzioni di sistema come socket, bind, listen, accept, send, e recv. La gestione degli errori e la sincronizzazione tra client e server sarebbero più esplicite.

Input/Output: Le operazioni di input/output in C sono meno esplicite e meno concise rispetto a Python. Avrei dovuto utilizzare funzioni come scanf e printf per gestire l'input e l'output.

Serializzazione dei Dati: In C, avrei dovuto implementare manualmente la serializzazione e deserializzazione dei dati, poiché non esiste un modulo incorporato come pickle di Python.

In sintesi, idealmente, il passaggio a C avrebbe richiesto una maggiore attenzione ai dettagli e una conoscenza più approfondita dei concetti di basso livello rispetto a quanto richiesto con Python.

**Struttura**

**Server (server.py)**

- Inizializza un socket per l'ascolto delle connessioni.

- Accetta connessioni dai client.

- Comunica con i client, gestendo il flusso di gioco.

- Gestisce le regole del gioco e mantiene lo stato globale del gioco (posizione della pallina, punteggi, ecc.).

**Client (client.py)**

- Inizializza un socket e si connette al server.

- Comunica con il server, inviando e ricevendo dati.

- Partecipa al gioco seguendo le istruzioni del server e inserendo le coordinate per nascondere e indovinare la posizione della pallina.

- Visualizza le informazioni sulla griglia e gli esiti delle mosse.

**Funzioni Comuni (functions.py)**

- Contiene funzioni comuni utilizzate sia dal server che dai client, come la visualizzazione della griglia e la spiegazione delle regole.

**Serializzazione e Deserializzazione (pickle)**

- Utilizza il modulo “pickle” per convertire i dati Python in un formato adatto alla trasmissione attraverso la rete (serializzazione) e per ricostruire i dati originali ricevuti (deserializzazione).

**Struttura di Controllo e Flusso (main)**

- Il main del server gestisce l'inizializzazione del gioco, le connessioni dei client e il flusso del gioco.

- Il main del client coordina le azioni del giocatore durante il gioco, inclusa l'interazione con il server.

**Gestione degli Errori**

- Implementa controlli di errore per affrontare scenari imprevisti durante l'esecuzione del gioco, come input non validi o disconnessioni improvvise.

**Spiegazione del Gioco (spiega\_regole)**

- Fornisce una spiegazione delle regole del gioco.

**Visualizzazione della Griglia (visualizza\_griglia)**

- Mostra la griglia con le informazioni sulla posizione della pallina e le mosse del client.

**Gestione del Punteggio**

- Registra e aggiorna il punteggio dei giocatori durante il gioco.

La struttura del progetto, quindi, è organizzata per fornire una base solida per il gioco, con divisione delle responsabilità tra server e client, utilizzo di funzioni comuni e gestione dei dati attraverso la serializzazione.

**Comunicazione**

La comunicazione tra il client e il server è implementata utilizzando i socket, che sono endpoint per la comunicazione bidirezionale attraverso una rete. Ecco una panoramica del flusso di comunicazione:

**Inizializzazione**:

- Il server inizia ascoltando su una porta specifica.

- Il client si connette al server utilizzando l'indirizzo IP e la porta del server.

**Messaggi Iniziali**

- Il server invia un messaggio iniziale di benvenuto e spiegazione delle regole al client.

**Turno di Gioco**

- Il gioco procede a turni alternati tra il server e il client.

- In ogni turno, il server e il client si scambiano informazioni sulla posizione della pallina o le coordinate per indovinare.

**Serializzazione e Deserializzazione**

- Prima di inviare i dati attraverso il socket, i dati vengono convertiti in un formato adatto alla trasmissione utilizzando il modulo `pickle` (serializzazione).

- Quando i dati vengono ricevuti, vengono ricostruiti nel loro formato originale (deserializzazione).

**Comandi e Risposte**

- Il server emette comandi al client, come "Nascondi la tua pallina" o "Indovina la posizione della pallina".

- Il client risponde con le coordinate o le azioni richieste.

**Aggiornamento del Gioco**

- Durante ogni turno, il server e il client aggiornano lo stato del gioco, controllano la correttezza delle mosse e calcolano eventuali punteggi.

**Fine del Gioco**

- Dopo un numero predefinito di turni(3), il gioco termina.

- I punteggi finali vengono inviati dal client al server.

- Il server determina il vincitore o se c'è una parità.

**Chiusura Connessione**

- Alla fine del gioco, il client e il server chiudono le rispettive connessioni.

Quindi, in sintesi, la comunicazione avviene attraverso lo scambio di messaggi strutturati tra il client e il server utilizzando i socket. I messaggi sono serializzati prima di essere trasmessi attraverso la rete e deserializzati al momento della ricezione per mantenere la consistenza dei dati.