Federico Calzolari 643496 - Documentazione del progetto

Nota: Nella scrittura del codice sono state prese delle scelte che tentano di facilitare un futuro aggiornamento del gioco implementando più mappe (scenario). Nonostante ciò, in questo progetto è possibile giocarne solo una che è co-op a due giocatori.

Nell'intero progetto **sono stati definiti alcuni standard** che hanno semplificato lo scambio di messaggi fra socket. Lo scambio di dati è sempre TCP, in quanto non è richiesta velocità da parte del server ma ci interessa che sia affidabile.

Messaggi con protocollo text:

- **Comandi**: la stringa di comando ha lunghezza massima di 6 byte.
- **Argomenti dei comandi**: lunghezza massima 10 byte.
- **Risposta**: utilizzata per la risposta agli enigmi, max 20 byte.
- **Buffer**: lunghezza massima di 256 byte, devono essere abbastanza grandi per contenere dei messaggi spiegati dopo.

Messaggi con protocollo binary:

- **Codici**: se invece un comando è gestito lato client (o non richiede un invio di messaggi di grandi dimensioni), mando un byte contenente un numero su 8 bit (uint8_t) per semplificare la send e la recv.

Strutture dati usate

Le strutture dati principali sono due e definite negli omonimi file header:

- **Account**: Questa struttura è organizzata come una lista con chiave primaria **ID.** Serve per immagazzinare le informazioni dell'utente come **email** e **password**, e lo **stato** (online, offline).
- **Sessione**: Anche questa struttura è organizzata come una lista con chiave primaria **id_account**. Serve per immagazzinare informazioni sullo stato della partita come i **token** ottenuti e i **flags** su cui si basano i comandi.

```
struct Account{
    int id;
    char email[30];
    char passw[20];
    bool status;
    struct Account* next;
};
struct Sessione{
    int id_account;
    int token;
    int flags[20];
    struct Sessione *next;
};
```

La scelta di usare delle liste è data dal fatto che è semplice scorrerle per recuperare una partita in corso e/o un account. Inoltre come possiamo vedere le sessioni delle varie mappe stanno in una lista separata per facilitare la ricerca.

```
struct Sessione* lista_sessioni[2];
```

Lista flags

0- testa messa 1- braccio messo 2- baule aperto 3-vetrina aperta
4-braccio in possesso 5-testa in possesso

Il flag settato indica che si è verificato l'evento

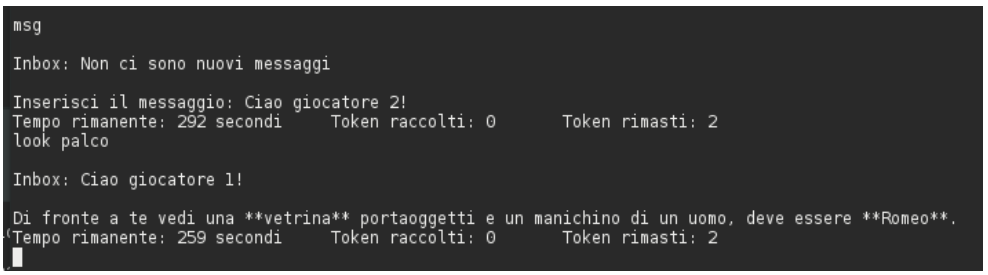
Server

Il server si accende e accetta solo comandi da standard input per farlo connettere o per spegnerlo. Una volta connesso accetterà massimo due client (in quanto gioco co-op) e dialogherà con essi per il login. Scelta la mappa dal primo dei due giocatori, l'altro si connetterà alla sessione del primo giocatore e il server accetterà e gestirà i comandi dei client fino alla vittoria o il comando da stdin "stop". Il server è basato su I/O multiplexing in quanto deve gestire più connessioni (listening socket, stdin, player 1-2) che non vengono chiuse subito ma rimangono aperte. Questo rende più facile la gestione dei comandi rispetto ad un server iterativo o concorrente.

Client

Il client ha un approccio semplice: dopo la fase di login e scelta della stanza entra in un ciclo infinito dove il giocatore dovrà immettere dei comandi per giocare. I comandi sono gestiti lato server (tranne alcuni casi) per permettere la cooperazione. Dopo l'immissione di ogni comando, **prima di mandarlo** (83-98 comandi_client.c), il client si occuperà di controllare se nella casella di posta sono presenti dei messaggi (funzionalità a piacere) e anche se il server è andato offline così da informare il giocatore e terminarsi. La scelta di non utilizzare altri metodi per una gestione in tempo reale dei messaggi che arrivano dal server è data dal fatto che si suppone che il giocatore non stia troppo fermo a pensare, essendo un gioco a tempo. Quindi, immetta comandi abbastanza velocemente da avere feedback costante sul tempo rimasto e sui messaggi in arrivo.

Funzionalità a piacere - Comunicazione tra giocatori



Lo scopo non è quello di avere una chat in tempo reale, ma una casella di posta così da rendere il gioco un po' più difficile per i giocatori. Tramite il comando 'msg' un giocatore può mandare una lettera all'altro giocatore rispettando i limiti del buffer di 256 byte. Lo scambio di messaggi passa attraverso il server che si occuperà di inoltrarli.

Come vincere

I giocatori devono recuperare il braccio del manichino di Romeo e la testa del manichino di Giulietta. Successivamente, dovranno usarli con il comando use.

Es. "use romeo braccio"

Gli oggetti sono dentro il baule e la vetrina, i giocatori per raccoglierli dovranno risolvere i due enigmi proposti.

Soluzione baule - verona

Soluzione vetrina - nome