DOCUMENTAZIONE PROGETTO VAGNOLI ANDREA 635788

Il server del progetto si basa su **I/O multiplexing**, per garantire la persistenza della connessione tra client e server e facilitare la gestione di più connessioni contemporaneamente. Lo scambio dei dati avviene sempre tramite **protocollo TCP**, in quanto mi interessa che lo scambio sia affidabile. Il formato dei messaggi scambiati si basa interamente su **protocollo text**, e ho implementato il seguente **protocollo di comunicazione**:

- Il client invia sempre 32 byte, comunicando con il server unicamente tramite comandi del tipo **command [obj1] [obj2]** (i parametri possono essere anche opzionali, a seconda della natura del comando). Il server, che quindi riceve sempre 32 byte, individua le tre componenti del messaggio utilizzando come delimitatore il carattere di spazio;
- Il server invia sempre 256 byte, che rappresentano la risposta al comando inviato dal client (risposta che poi viene stampata su schermo per renderla nota all'utente).

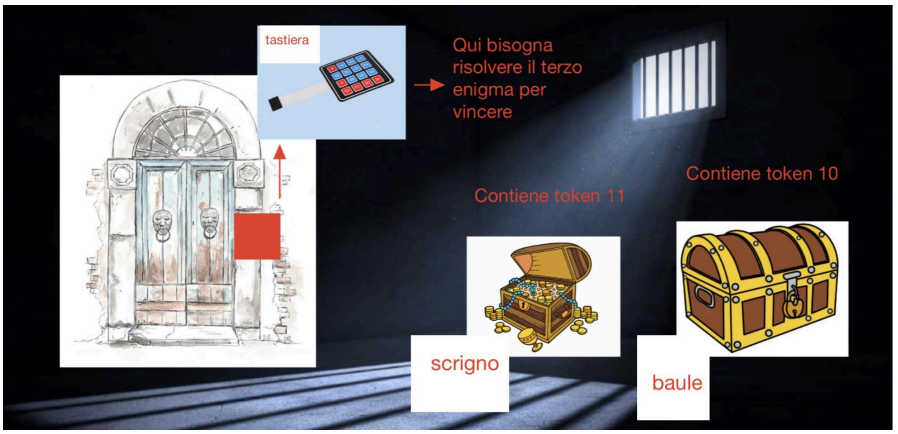
Le due grandezze (32 e 256) sono state scelte stimando, con un certo margine, la reale grandezza massima dei messaggi che era necessario inviare e ricevere. Può verificarsi che vengano inviati più byte del necessario (perché il messaggio da inviare/ricevere può effettivamente essere più breve di 32 o 256 byte), ma ho valutato essere trascurabile la quantità di traffico generato da eventuali byte "sprecati" (la valutazione è stata fatta tramite "buon senso", senza nessun criterio quantitativo). Questo accorgimento ha permesso di rendere molto più semplice il meccanismo di comunicazione, facilitando lo sviluppo dell'applicazione.

Alla fine, non ho rilevato particolari punti critici e non sono emersi particolari difetti sulle scelte progettuali fatte, potendo quindi prevedere un buon risultato anche in termini di scalabilità. Questo pensiero fa riferimento tuttavia ad un contesto di utilizzo didattico dell'applicazione, basato su un numero limitato di utenti collegati. Per semplicità, ho deciso comunque di limitare il numero massimo di socket monitorabili a 10, che comunque permettono un adeguato test dell'applicazione.

Il progetto si divide in 5 file:

- **server.c** → gestisce la connessione con i socket tramite I/O multiplexing, monitorando anche il descrittore relativo a stdin per impartire comandi al processo server;
- **client.c** → l'unico comando gestito lato client è il comando end, relativo alla chiusura della connessione. Il server ha comunque modo di accorgersi della chiusura della connessione tramite la recv che in tal caso restituisce 0 (ciò mi permette anche di "risparmiare" l'invio di un messaggio per comunicare la chiusura della connessione). Per il resto, il programma si limita a stampare ciò che riceve dal server e inviare al server l'input inserito dall'utente (i controlli vengono effettuati esclusivamente lato server).
- **manage_client_command.c** → contiene i controlli lato server che vengono effettuati sul buffer inviato dal client per analizzare il comando;
- **utilitity.c utility.h** → contengono strutture dati e funzioni necessario allo sviluppo del gioco (gestione sessioni, utenti, mosse, dinamiche di gioco etc.).

La memorizzazione degli utenti registrati e le relative sessioni vengono gestite tramite liste (in utility.h sono forniti più dettagli sul significato delle singole variabili e delle strutture dati).



- schema della mappa implementata nel gioco -

Il **gioco** si sviluppa nel seguente modo: l'utente, dopo aver effettuato registrazione e login (le registrazioni non vengono salvate su file, quindi vengono perse ad ogni spegnimento del server), può decidere in quale stanza giocare (è stata implementata una sola stanza, "Enigma01", ma l'applicazione è pensata per poter supportare anche più di 1 mappa).

Successivamente viene mostrata la **funzionalità a piacere** che ho deciso di implementare: un giocatore può giocare in una stanza nel ruolo di "protagonista" o di "oracolo". La possibilità di giocare in modalità oracolo è permessa solo se esiste almeno un altro giocatore che già stia giocando in modalità protagonista nella stessa stanza. In tal caso, l'utente oracolo può decidere quale giocatore "aiutare" (viene specificato successivamente nella documentazione cosa si intende con il termine "aiutare"), ricevendo così l'oggetto "bacchetta", che può essere utilizzata attraverso il comando "use bacchetta". L'oracolo può tenere traccia poi del risultato del giocatore che ha aiutato (vittoria, sconfitta) mettendosi in attesa e utilizzando il comando look, che lo informa sullo stato della partita del giocatore protagonista.

L'utente che accede alla stanza nel ruolo di protagonista invece, come da specifiche, può utilizzare i seguenti comandi: "look", "take", "objs" ed "end". Ho deciso di aggiungere anche il comando "lay", che permette all'utente di lasciare un oggetto che ha in mano (può tenere al massimo 1 oggetto per volta). Il comando lay si può usare da solo, o insieme all'oggetto che si vuole lasciare (in tal caso lascerà comunque l'oggetto in mano, qualsiasi esso sia).

L'utente per vincere, deve sbloccare due token, presenti all'interno di due oggetti bloccati (scrigno e baule) entrambi da un enigma ciascuno, composto da una parola da inserire. In seguito deve sbloccare il terzo oggetto (tastiera) per uscire dalla stanza tramite un terzo enigma, questa volta di tipo numerico. Per poter sbloccare il terzo oggetto è necessario aver risolto i primi due indovinelli e che si siano verificate una o entrambe delle seguenti condizioni:

- Il tempo rimanente è minore di 5 minuti;
- Un oracolo ha usato il comando use bacchetta per aiutare il protagonista;

Si capisce quindi che l'utilità del giocatore oracolo è quello di permettere all'utente di poter pensare per più tempo alla soluzione del terzo indovinello. Il tempo massimo, entro il quale si può vincere è 30 minuti, scaduto il quale, al primo inserimento di un nuovo comando da parte del giocatore viene notificata la sconfitta. In caso di vittoria, anch'essa essa viene notificata e l'utente può successivamente scegliere se iniziare una nuova partita su una mappa (che sia la stessa od una nuova, anche in ruoli diversi da quello utilizzato precedentemente, nei limiti delle regole esposte sopra). L'oracolo invece può decidere se rimanere in attesa del compagno aiutato per scoprire l'esito della partita o uscire dal gioco e poi eventualmente effettuare di nuovo il login.

Il giocatore riceve inoltre un aiuto a schermo in seguito a comandi inviati in maniera erronea.

SOLUZIONE ENIGMI: - "orologio" sblocca lo scrigno - "vocali" sblocca il baule - "100" sblocca la tastiera e quindi permette di vincere (**100** è il naturale successivo rappresentato in binario dei numeri che compaiono nel gioco, sempre in binario, "Enigma**01**" - "Token**10**" - "Token**11**").