Progetto Reti informatiche 2024

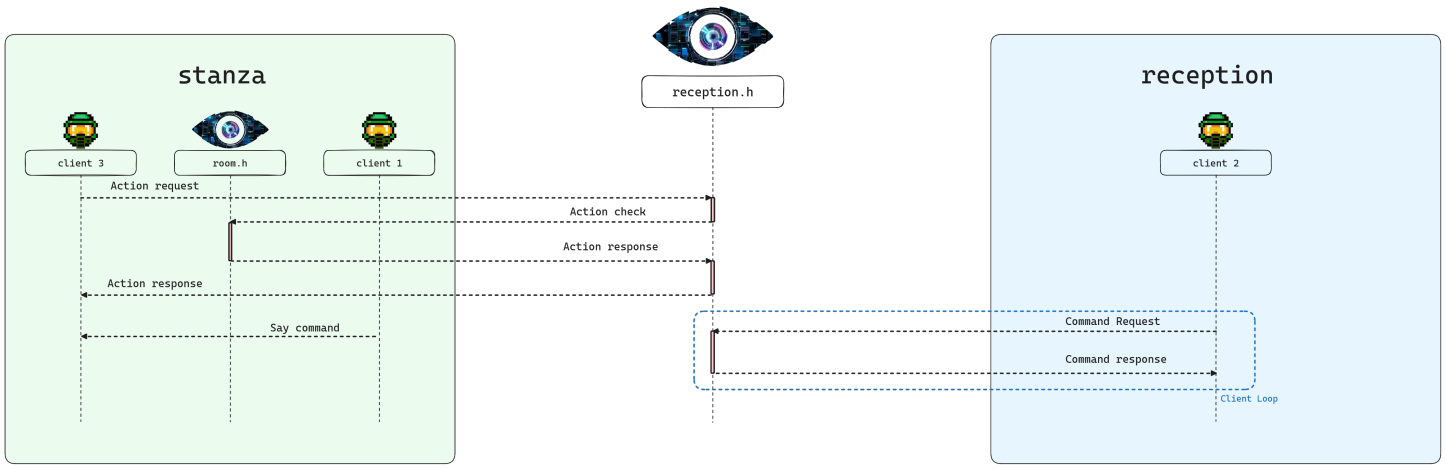
Francesco Mignone 569223

L'applicazione utilizza I/O multiplexing e socket di tipo TCP bloccanti. È stato scelto di usare il protocollo TCP in quanto non si necessita di un'eccessiva velocità di trasmissione ma si necessita di affidabilità. Una volta inizializzata la connessione essa rimane attiva per tutta la sessione, per questo l'overhead dell'handshake TCP non fa diminuire di troppo le prestazioni.

Lo stato di un client (gamer) viene suddiviso in tre stati: non autenticato, autenticato gestito dal server tramite il modulo `reception.h`, e in gioco gestito all'interno della room dal modulo `room.h` e `reception.h`. Si pensi come a un cliente che entri in una Escape-Room: all'esterno l'utente non ha significato, all'interno viene autenticato e registrato e successivamente entra in un a stanza, si noti come alcune funzioni dei `reception.h` e `room.h` si scambino informazioni volutamente in quanto un giocatore all'interno di una stanza può parlare con la reception tipicamente tramite un interfono o la reception può interagire con la stanza conoscendo lo stato tramite sistemi di sorvelgianza.

Il modulo network.h permette di definire server modulari, che utilizzino I/O multiplexing, con chiamate a funzioni in vari stati della `listen_server()`: `TickFunction()` eseguita a inizio ciclo dopo la select, `AcceptFunction()` eseguita dopo che il server ha eseguito l'`accept()` sul socket in ascolto, `ResponseFunction()` eseguita a seguito della `_receive()` e seguita da una `_send()` in modo tale da poter iterpretare il messaggio ricevuto secondo le esigenze di chi instanzia il server, `InputFunction()` per gestire il messaggio letto dal socket dello standard input, `DisconnectFunction()` chiamata in caso di errore su di un socket per gestirne la corretta disconnesione. Sono inoltre fornite funzioni per inizializzare le strutture client e server e le annesse operazioni di rete come `conect()` e `listen()`.

Formato Messaggi: I messaggi inviati e ricevuti sono tutti dello stesso formato `message`, varia solo la tipologia del messaggio secondo i tipi `msgType` e `cmdType` presenti in `protocol.h` e un campo di testo per i parametri, i messaggi non hanno una lunghezza predefinita e essendo delle strutture vengono serializzate e deserializzate rispettivamente prima dell'invio e della ricezione, per questo sono state implementate le funzioni `_send()` e `_receive()` per inviare e ricevere la corretta lunghezza e formato del messaggio. Viene utilizzato il Text protocol visto che i messaggi sono prevalentemente di testo a discapito del consumo di banda maggiore.



Il gioco: Il giocatore deve prima effettuare il login o una registrazione. Una volta effettuato l'accesso può avviare una stanza tra quelle disponibili, che possono essere mostrate con il comando `show rooms`. All'avvio il giocatore viene posizionato all'ingresso della stanza e può interagire con le locazioni o oggetti come da specifiche, si puó interagire solo con gli oggetti presenti in una locazione come mostrato dalla descrizione e non con oggetti presenti al di fuori di essa, bisogna prima spostare la locazione sulla locazione di interesse, se si vuole tornare a vedere la stanza principale si puó digitare `look room`. Quando viene risolto un indovinello viene mostrato nella "chat" di tutti i giocatori connessi alla stanza, la sincronizzazione tra chi risolve l'enigma é assente, chi risolve per primo l'indovinello ottiene l'oggetto ad esso connesso, mentre i token sono assegnati alla stanza e il gioco viene considerato concluso quando tutti i token sono stato o raccolti o sbloccati, il gioco é considerato concluso e quindi perso quando i

giocatori non riscono a ottenere tutti i token entro il tempo limite. Sono stati implementati solo 2 tipi di puzzle: nel momento in cui si interagisce con un oggetto potrebbe essere sichiesto un indovinello per essere preso, oppure un uggetto richiede un'altro oggetto per essere sbloccato. É stato implementato il comando `drop` per rilasciare un oggetto dal proprio inventario. Come funzionalitá a piacere é stato implementato il comando `say` per mandare un messaggio a tutti gli utenti della room in modalitá P2P con index centrale. Il server può accettare un numero generico di client (il limite é solo la memoria e le capienze delle stanze).

Server: Il server viene messo subito in ascolto delle richieste dei client. Una volta autenticati viene istanziata una struttura `gamer` con i relativi dati ad un giocatore e vengono lasciati ad aspettare nella reception finché non viene ricevuto il comando di richiesta per una stanza. Ricevuta la richiesta il server controlla se é disponibile una stanza con la stessa mappa e se non é presente il server si occupa di caricare e instanziare una struttura `game_room`, la mappa creata da un individuo é caricata dalla cartella `./room_template/` dove al nome della mappa corrisponde una relativa cartella, all'interno si possono trovare dei file[1] che identificano il nome della locazione e all'interno troviamo: come prima riga la descrizione della locazione con gli oggetti con cui si puó interagire indicati come da specifiche, si noti come non é stato implementata la possibilitá di avere una locazione dentro una locazione (puó comunque essere aggiunta per come é strutturato il codice). Il server si occupa di gestire la disconnessione improvvisa o per comando `end` del client rimuovendo la struttura `gamer` e chiudendo il socket, l'inventario del giocatore viene rilocato coerentemente a dove erano stati presi gli oggetti.

Client: Il client si sviluppa tutto tramite la gestione dell'input da tastiera tramite la `InputFunction()` la quale si occupa di controllare il comando inserito e chiamare la funzione che richiede il comando con annessi parametri se presenti al server e gestisce la risposta e se necessario stampata a video. Il client dispone anche di un server in modo tale da poter ricevere e gestire messaggi dal server o eventualmente da altri giocatori, in caso del server vengono gestiti i comandi di vittoria e sconfitta della partita. Il client richiede uno scambio di messaggi con la funzione `request()` la quale ha il semplice compito di eventualmente creare la connessione tra un client e un server e successivamente inviare il messaggio tramite `_send()` e attendere la risposta tramite la `_receive()`.

In tutto il progetto qundo si scrivono stringhe sia terminale che nei file evitare l'utilizzo del carattere `'_'` viene utilizzato come separatore e potrebbe rompere qualcosa.

1. La struttura di una mappa é cosi composta: all'interno della cartella si deve trovaer un file chiamato `room` il quale si occupa di impostare la stanza base che viene trattata come una location, il formato del file é descritto in seguito insieme alle location. I successivi file sono le location che compongono la stanza, il formato del file deve essere il seguente: la prima riga deve contenere la descrizione della location, in seguito in ogni riga si puó aggiungere un oggetto separando ogni campo che corrisponde all'oggetto da un `'_'`, l'ordine dei campi é lo stesso della struttura presente in `room.h` nel quale é anche presente tramite commenti una descrizione dei vari campi(evitare spazi nel nome dell'oggetto e ovviamente evitare di specificare una locazione di memoria del prossimo item, garantisco che non funziona a meno che non si voglia gestire la lista a mano), fare comunque riferimento alle mappe già presenti per avere un idea. Inoltre i file delle location devono essere senza estensione e le locazioni non devono essere omonime.

2. La funzione per annullare la registrazione di un utente é cancellarlo dal file, semplice e funzionale. Consultare il file `credentials.txt` per gli utenti già registrati.