Università di Pisa
Corso di laurea in Ingegneria Informatica

Documentazione Progetto di Reti Informatiche

Walter Soro - 530802

Introduzione

Il progetto si basa su un modello client-server, pensato per garantirne la modularità e la portabilità. I moduli per il client e il server sono indipendenti e organizzati in cartelle separate. Allo stato attuale, l'applicazione consente di giocare in una sola room con partite composte da due giocatori; è però possibile modificare il numero di giocatori cambiando il valore della macro corrispondente nel file ***game_utility.c*** e aggiungere facilmente nuove room inserendo le informazioni necessarie nei moduli ***scenari*** per il client e ***scenari_serv*** per il server, poiché il resto del codice è generico e progettato per funzionare su più room. Il protocollo usato per il trasporto è il protocollo TCP, scelto per garantire una trasmissione affidabile. Durante la trasmissione di un pacchetto, viene prima inviata la dimensione del messaggio in modalità binary, seguita dal messaggio in modalità text. Questa scelta è giustificata dal fatto che la maggior parte dei dati da inviare sono stringhe, il che comporta un maggior consumo di banda, limitato però dalla scelta di spostare la parte testuale dell'applicazione, come descrizioni di oggetti e messaggi di errore, sul lato client.

Implementazione Server

Dopo l'avvio, il server rimane in attesa del comando *start.* Il server si basa su l'I/O multiplexing in quanto l'applicazione è principalmente I/O bound e deve poter gestire più connessioni simultaneamente.
Lo scambio di comandi tra client e server deve essere nella forma "*comando,parametro1 parametro2 …* " utilizzando come valori dei parametri il formato esteso o, nel caso degli oggetti, l'indice associato all'oggetto all'interno della room attuale. Questa scelta evita di aumentare la complessità della ricerca tramite stringa, delegando questo onere al client che possiede in locale le informazioni necessarie. Il server riceve i comandi ed effettua un controllo sugli operandi, scartando eventuali parametri in eccesso e inviando codici di errore in caso di parametri mancanti o errati. I codici di errore sono descritti nel modulo ***errori***, presente anche nel client, in modo da mostrare al giocatore l'errore corrispondente senza necessità di inviarne il testo. Le strutture dati del server, più generiche rispetto a quelle del client, contengono solo le informazioni necessarie a gestire l'interazione tra i giocatori e gli oggetti nella room. Gli oggetti sono memorizzati in una struttura dati nel modulo ***scenari_serv*** e ogni partita contiene il proprio set di oggetti inizializzato durante la creazione; questo perché la struttura degli oggetti contiene variabili per lo svolgimento della partita in modalità cooperativa, come l'impossibilità di interagire con oggetto in uso da un altro giocatore. Il lavoro principale del server, infatti, consiste nel validare i comandi del client e comunicarne l'esito.

Implementazione Client

Il client si occupa per prima cosa del login del giocatore, mandando le richieste al server che effettua gli eventuali controlli di validità. Dopo aver inizializzato il gioco, inizia un ciclo

d’attesa basato sul valore della variabile ***in_game***, settata dopo il login e resettata opportunamente alla fine del gioco, sia per volontà dell’utente o per vittoria.
L’utente può immettere i comandi come da specifiche, con l’aggiunta di un comando, gestito interamente in locale, per mostrare nuovamente la descrizione della stanza (comando ***room***). Quando il giocatore immette il comando ***start room***, il server risponde con l’istante di inizio della partita, permettendo la gestione in locale del timer.
Il modulo ***scenari*** del client contiene le seguenti strutture dati:

- ***Object_cl***: contiene informazioni sugli oggetti (descrizione, enigmi, soluzione etc).
- ***Room:*** contiene informazioni relative alle room (quanti e quali oggetti, descrizione e codice identificativo).
- ***Partita:*** contiene il codice della room attualmente in gioco e il numero di token guadagnati dall’utente, informazione che viene utilizzata solo nel momento in cui si stampa il messaggio di vittoria.

Queste informazioni sono gestite lato client per ridurre il carico del server e l’utilizzo di banda, evitando la trasmissione da parte del server di lunghe descrizioni degli oggetti.

Funzionalità a piacere: cooperazione

La funzionalità a piacere consiste nella cooperazione completa durante il gioco: i giocatori entrano nella stanza solo insieme ad altri giocatori e, sia l’inventario che i token sono condivisi.
Gli oggetti possono essere usati da un solo giocatore per volta.
Dopo ogni azione il client manda in automatico il comando ***CHKLOG*** al server per ottenere un log, in formato testuale, delle ultime azioni o eventi degli altri giocatori. Questo log è memorizzato nel server all’interno della struttura ***Giocatore***, viene aggiornato dopo ogni evento significativo di ogni altro giocatore in partita e viene cancellato dopo l’invio.

Svolgimento del gioco:

1. Login: digitare ***s*** o ***n*** per fare login o creare un account.
2. Presentazione, in sequenza, delle descrizioni delle room disponibili nel gioco.
3. Digitare ***start 1*** per far partire il gioco nella ROOM1, unica room attualmente disponibile nel gioco. Attesa del secondo giocatore.

Gli oggetti del gioco non saranno visibili tutti nella descrizione della room ma possono essere nascosti nelle descrizioni di location o di altri oggetti.

Soluzione gioco:
Ci sono 4 token da ottenere. Si trovano risolvendo 3 enigmi e 1 azione *use* (combinata):

- Prendere la chiave (***take chiave***): soluzione 1;
- Prendere la statuetta (***take statuetta***): soluzione 2,
- Prendere la moneta (***take moneta***): soluzione 3
- Usare lo scrigno e la chiave in modo combinato (***use scrigno chiave***)

Una volta ottenuti i 4 token al client che ha ottenuto l’ultimo token viene mostrato un messaggio di vittoria, mentre al secondo giocatore verrà mostrato dopo l’azione successiva. In caso di risposta sbagliata ad un enigma il giocatore viene bloccato per 10 secondi. Il timer impostato è di 600 secondi, salvato nella macro ***TIMER_DURATION*** del modulo ***utility*** del client.