DOCUMENTAZIONE PROGETTO RETI INFORMATICHE

Anno 23-24 - Kevin Giannandrea (matricola: 625722)

Lo sviluppo del gioco è basato sull'utilizzo dei socket TCP, in quanto è necessario avere affidabilità dei messaggi scambiati e una gestione della connessione tra il server e il client, senza necessità di avere una velocità elevata nella trasmissione. Lo scambio dei messaggi è basato sull'invio della dimensione del messaggio (in formato *binary protocol*), e infine del messaggio (in formato *text protocol*), questa scelta progettuale (invio della dimensione e poi messaggio effettivo) è basata su dei criteri di affidabilità: Il destinatario sa quanti byte dovra' ricevere quindi semplifica la ricezione del messaggio effettivo e la gestione del buffer (evitando buffer-overflow), inoltre conoscendo la dimensione puo' confrontarlo con la dimensione effettiva dei dati ricevuti. Il client e server inoltre comunicano mediante l'invio di un byte, ad esempio se il client vuole effettuare una operazione (take, use, ecc) lo comunichera' mediante un byte relativo al tipo di operazione, il server risponderà anch'esso con tale formato di comunicazione (il server invia un byte nei casi di errore o fine del gioco), nello specifico è definito nel file "*cmdClientServer.h*" i possibili byte di invio (il file header è incluso sia nel client e nel server).

SERVER

Il server è implementato usando l'I/O multiplexing poiché per soddisfare le richieste di un client non è necessario che il server attenda (quindi non è necessario avere un processo che gestisca appositamente una richiesta di un client), esso viene messo in ascolto all'inizio inserendo la stringa 'start' da linea di comando. Dopo aver avviato il server i client possono connettersi ed è richiesto di autenticarsi o nel caso di primo accesso di registrarsi. Le informazioni dei client che si registrano vengono memorizzate nel file "*utenti.txt*", quindi ad ogni operazione di accesso viene verificata l'autenticazione. Inoltre è richiesto di specificare il tipo di ruolo che il cliente può avere, giocatore o aiutante. Il server gestisce le strutture dati per la gestione dei giocatori tramite una lista. Dopo la fase di accesso, al client è richiesto di specificare la stanza di gioco, e dopo aver ricevuto la specifica il server si occupa di allocare le risorse per istanziare una nuova partita. Nello specifico il server si occupa di gestire solo la lista dei giocatori perché la logica del gioco è implementata mediante il modello 'macchina a stati'. Il giocatore nella sua struttura dati ha un attributo "stato" che permette di gestire le operazioni che puo' fare. Anche gli oggetti hanno un proprio stato, ad esempio se il giocatore vuole ottenere un oggetto che ha nel suo descrittore uno stato maggiore a quello del giocatore, allora esso dovrà risolvere un enigma. Il giocatore puo' aumentare lo stato risolvendo enigmi o usando oggetti combinati tra loro. L'aumento dello stato del giocatore consente di avanzare nel gioco e di arrivare al completamento. Ogni qual volta il client vuole eseguire una operazione invierà al server il byte relativo al tipo di operazione, il server riceve il comando e in base al byte identifica il tipo di operazione. Il server identifica se

l’operazione puo’ essere eseguita (ad esempio se una take puo’ essere fatta, in base allo stato del giocatore o se ad esempio l’oggetto è gia’ presente nell’inventario). Ogni qual volta che viene completata l’esecuzione di un comando il server invia al client le informazioni sul tempo rimanente e sui token raccolti. A ogni risoluzione di un enigma viene aumentato il numero di token al giocatore. Il server oltre a gestire le operazioni nelle specifiche implementa anche l’operazione ‘drop’ che consente al client di liberare un oggetto dal suo inventario. Il server per gestire il gioco si serve del file “gameF6.h”, dove sono definite le costanti e le strutture dati necessarie a descrivere la stanza di gioco ‘F6’, ad esempio “MAX_TIME”, “MAX_TOKEN”. Inoltre ogni locazione e oggetto è descritto mediante dei campi nel vettore ‘elementi’, ad esempio un oggetto puo’ contenere un enigma e la relativa risposta e ulteriori campi necessari alla sua descrizione. Quando il giocatore aumenta il proprio stato, l’operazione di look può restituire una descrizione diversa (ad esempio una porta chiusa che poi viene aperta). Se il giocatore termina entro il tempo limite allora il server lo avviserà, quindi il client si disconnette. Il server gestisce la chiusura delle connessioni da parte dei client tramite la funzione “gestione_end”, infatti anche in caso di disconnessioni improvvise del client la funzione viene chiamata in modo da provvedere a deallocare le risorse e a eliminare il socket da quelli monitorati.

CLIENT

Il client all’avvio attende che il server si accenda e a quel punto viene mostrata una schermata per l’autenticazione. Inoltre è necessario specificare il ruolo e la stanza di gioco che si vuole giocare. A questo punto la stanza è inizializzata e il client puo’ inserire da stdin un'operazione che vuole effettuare. L’input del client è gestito dalla funzione *“lettura_comando*” che si occupa di leggere una stringa valida e di inviare al server il byte in base al tipo di operazione. Il client non è a conoscenza dello stato del gioco, esso puo’ solo specificare il tipo di operazione e attendere la risposta da parte del server. Quando il giocatore digita una operazione da linea di comando ad esempio una take, il codice del client si occupa di verificare che sia una stringa valida prima di inviare il comando al server (in modo da evitare di sovraccaricare il server inutilmente), le stringhe valide iniziano con il tipo di operazione (look, use, take, ecc) seguite dal parametro (nel caso della use anche due parametri). Infine il server consente a un client di connettersi è di specificare il ruolo ‘aiutante’ nel gioco, cioè il client genera in modo pseudo randomico un numero intero compreso tra 1 e 60 che lo invierà al server, a questo punto il server prende un giocatore in modo casuale che sta giocando e gli aumenta il tempo rimanente con il valore inviato dal giocatore ‘*aiutante*’ in modo da aiutarlo.