Taller 3

Programació de Jocs en Xarxa

Pol Casamiquela Vázquez

Eloi Farriol Cristóbal

3r Grau en Continguts Digitals Interactius

enti
ESCOLA DE NOVES TECNOLOGIES INTERACTIVES

Centre adscrit a:
UNIVERSITAT DE BARCELONA

1. Indicar cómo poner en marcha el proyecto

El proyecto se puede poner en marcha simplemente abriendo la solución con Visual Studio y ejecutando el código en modo *release* o *debug*.

2. Diagrama de clases utilizado

En este proyecto no hemos implementado ninguna clase.

3. Explica de qué forma aceptas las conexiones en servidor e indica en qué parte del código puedo encontrarlo.

Para comprobar si hay algún cliente que intenta conectarse al chat, creamos un socket para comprobar su estado constantemente en el loop principal del main. Cuando su estado es *sf::Socket::Done* sabemos que un cliente intenta comunicarse con el servidor y añadimos un nuevo socket al vector *socketList*.
Esto podemos encontrarlo en la línea 52 del código.

4. Indica dónde se crea el socket de cliente.

El socket del cliente se crea junto con el resto de variables en el inicio de la función *main()* y se declara como *nonblocking*.

5. Indica dónde tienes definida e inicializada la lista de mensajes del cliente y el servidor.

En ambos casos la lista de mensajes se crean en el inicio de la función *main()*, con *std::vector<std::string> aMensajes,* y se actualiza solo cuando entra un mensaje por el socket, con un push_back. No obstante, el servidor no emplea la lista de mensajes en ningún momento, simplemente se trata de un registro de los mensajes que hemos querido mantener por si fuera necesario hacer un “track” de los mensajes. En el caso del cliente, utilitza la lista de mensajes para printarlos en pantalla y mostrarlos al usuario.

6. Indica dónde está el código que recibe mensajes en el cliente y en el servidor.

Tanto en el servidor como en el cliente los mensajes se reciben en el Loop principal del main(), donde se comprueba el estado del socket y si es *sf::Socket::Done* se añade el mensaje recibido en la lista de mensajes.

Esto lo encontramos en la línea 141 del cliente y en la línea 66 del servidor.

---

7. Indica dónde está el código que envía mensajes en el cliente y en el servidor.

En el caso del cliente se llama a la función *SendFunction()* cuando el estado actual del socket es *sf::Socket::NotReady*. En esa función se gestionan los inputs del usuario y cuando se envían los mensajes al servidor.

En el caso del servidor solo se envían mensajes cuando éste recibe un mensaje de un cliente, que lo reenvía a todos los demás clientes incluyendo el remitente.

---

8. Indica si estás utilizando sockets nonblocking y si lo estás haciendo, en qué lugar del código los desbloqueas y porqué.

Estamos utilizando *sockets nonblocking* para poder enviar y recibir mensajes a la vez sin que se bloquee la ejecución. El servidor crea uno al inicio del *main()* y lo desbloquea a continuación, y cuando se confirma la conexión con el cliente crea otro para estar preparado por si sucede otra conexión con otro cliente (línea 52) y lo desbloquea inmediatamente. El cliente crea su socket al inicio del *main()* y lo desbloquea poco después dentro de la misma función.

---

9. Indica dónde desconectas los sockets

El cliente desconecta su socket después de pulsar *Esc* o escribir */exit*. En ese momento, el Servidor detectaría que en uno de los sockets que tenga abiertos se ha producido una desconexión y por lo tanto también cerraría el suyo. Por parte del cliente, esto se gestiona desde la *SendFunction()* y por parte del servidor se gestiona desde el Loop principal del *main()*.

---

10. Postmortem: Dificultades que has encontrado y conclusiones.

En esta práctica no hemos encontrado ninguna dificultad particular a parte de tener que plantear cómo actúa el servidor a la hora de disponer de capacidad suficiente para albergar las conexiones entrantes. Por otro lado, hemos podido observar que las conexiones cliente servidor son costosas para la CPU y por ello si añadíamos un pequeño "delay" (*sf::sleep*) en los loops, el rendimiento mejoraba considerablemente sin apenas ser apreciable por el ojo humano.