PROTOCOLO Y POST-MORTEM TALLER 8

PROTOCOLO

Estructura mensaje	Sentido	Acción asociada
<PACKETTYPE::HELLO>	**C->S**	Intento de conexión con el servidor por parte del cliente
<PACKETTYPE::WELCOME>_<id>_ <sizeClients>_[<clientNId> _<clientNx>_<clientNy>]	**S->C**	Da la bienvenida a un jugador, además de darle toda la información inicial del mundo y su propio ID.
<PACKETTYPE::NEWPLAYER>_<id>_ <x> _<y>	**S->C**	Notifica a los jugadores que un jugador nuevo se ha conectado
<PACKETTYPE::DISCONNECT>_<id>	**S->C**	Notifica a los jugadores que un jugador se ha desconectado
<PACKETTYPE::ACK>_<id>_<msgID>	**C->S**	Confirma la recepción de un paquete crítico
<PACKETTYPE::PING>	**S->C**	Envía una petición de ACKPING a los clientes
<PACKETTYPE::ACKPING>	**C->S**	Envían una respuesta al PING del servidor para indicar que siguen conectados
<PACKETTYPE::ACKMOVE>_ <playerID>_<moveID>_<absX>_ <absY>	**C->S**	Manda la posición de un jugador con id playerID a todos los clientes conectados, si playerID és igual al ID del jugador que lo recibe aprovechara moveID para limpiar su lista de movimientos no “acknowledgeados”
<PACKETTYPE::MOVE>_<idMove>_< deltaX>_<deltaY>_<absX>_<absY>	**C->S**	Manda la suma de deltas acumulados en un pequeño tiempo así como la posición absoluta que tendría el jugador aplicándole ése delta
<PACKETTYPE::NOTWELCOME>	**S->C**	Niega la entrada a un cliente ya que la sala está llena

Hay más comandos preparados dentro del ENUM pero aún no están implementados.

Los clientes acumulan deltas que luego mandan con el comando MOVE, el servido acumula estos deltas y cada vez que el reloj de alguno de los clientes pasa un tiempo establecido, reenvía el movimiento con el ID más alto en caso de validarse este.

POST MORTEM

Se empezó estableciendo una acumulación de movimientos solo en servidor, pero nos dimos cuenta de que en las especificaciones del taller se pedía además una acumulación de deltas en Cliente y nos hizo tener que cambiar un poco la estructura del código.

La acumulación se hace correctamente y parece funcionar como cabía a esperar, con algo de lag.

Lo más difícil fue arreglar el código del taller anterior que tenía una serie de problemas generados por mal control de algunas de las variables, específicamente la variable clientID era asignada en algún caso que no debía serlo.

Lo que es el establecimiento de envío de movimientos ha sido bastante fácil de codificar, aunque cabe a apuntar que con la inclusión de la pérdida de paquetes uno se da cuenta de que no acaba de funcionar bien y deberíamos establecer algún método de envío de movimientos aunque el jugador no envíe inputs, las opciones que se nos han ocurrido, aunque no hemos llegado a implementar son enviar AccumMoves aunque el jugador no haga ningún input dentro de los parámetros de tiempo ya establecidos o enviar un tipo secundario de AccumMove con otro reloj aparte que tenga en cuenta el tiempo que hace que el jugador no envía inputs ni recibe ACKPINGs, o que incluso simplemente compruebe si quedan movimientos en la lista de movimientos no verificados y con otro reloj reenvie el último AccumMove.

Pérdida de paquetes

Dentro de Client.h hay un define llamado LOSSRATE que determina las posibilidades de que un paquete MOVE o ACKMOVE no se envíe.