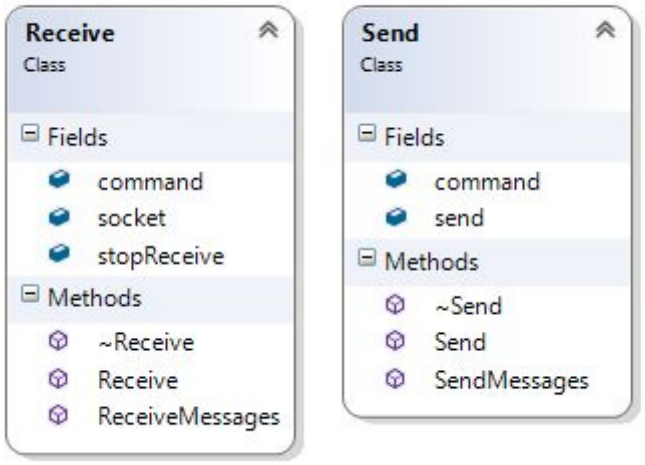
Práctica - Matchmaking

Poner en marcha el proyecto

1. Entrar en x64\release
2. Ejecutar un servidor
3. Ejecutar clientes
4. Poner puertos diferentes en los clientes (e.g. 5002, 5003, 5004).
5. Poner los wins (máximo 63, sirve para emparejar jugadores)
6. Seguir instrucciones de la consola
7. Pulsar Z para ganar

Diagrama de clases.



Reglas del juego.

Pulsar Z para ganar

Protocolo de comunicación.

Protocolo de comunicación, optimizado a nivel de bit utilizando Input y Output Memory Bit Stream. Tenemos definido los tipos de mensaje y cuanto pesa cada uno, para optimizarlo lo mejor que podamos.

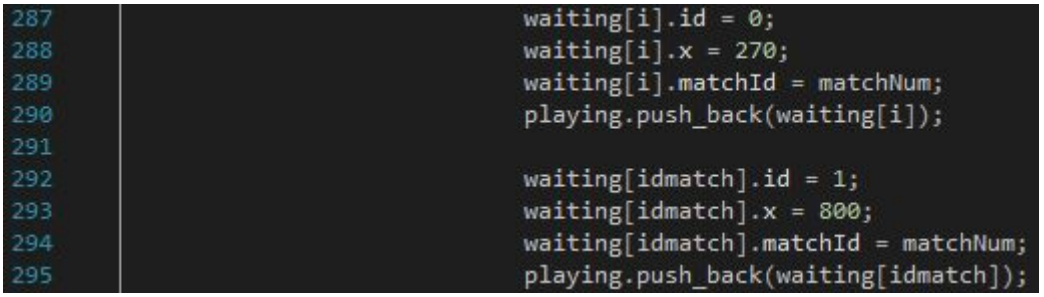
Client -> Servidor		
Estructura	Bits	Accion
HELLO_wins	4_6	El cliente pide conexion al servidor i envia sus wins
CONEXION_id	4_1	El cliente comfirma que está listo para jugar
DISCONNECT_id	4_1	El cliente avisa que es desconecta
ATTACK_id_move	4_1	El client envia que quiere ganar.
PLAY	4	El client envia que ha rebut la puntuacio i que ja torna a jugar
SEARCH_id	4_6	El client envia que vol jugar i el seu id

Servidor -> Client		
Estructura	Bits	Accion
HELLO_id	4_6	El servidor conecta al jugador o l'avisa de
CONNEXION_propia/contraria_id_x	4_1_1_12	El servidor envia la posiciones a los jugadores (propia y contraria)
DISCONNECT_id	4	El servidor envia que el jugador ha de tornar al lobby
PLAY	4	El servidor envia que començan a jugar

Múltiples Juegos

Los jugadores del servidor guardan una id de partida (los clientes no). Cuando se realiza matchmaking, los dos jugadores que van a jugar reciben un id de partida nuevo, y se crea uno nuevo que se usará para el siguiente matchmaking.

Cuando el jugador de una partida envía un comando, el servidor busca a su pareja en la lista de jugadores “playing” para realizar la acción correspondiente. De esta forma, el servidor puede recibir comandos de jugadores que no estén en la lista de “playing” o estén en una partida diferente sin interferir con los demás .



Se prepara el juego y se mueven los jugadores de waiting a playing

```
for (int i = 0; i < playing.size(); i++) // Busca el matchid i desconectar jugador
{
    if (playing[i].id == com.front().id)
    {
        matchid = playing[i].matchId;

        OutputMemoryBitStream output;
        output.Write(DISCONNECTION, TYPE_SIZE);
        sender.SendMessages(playing[i].ip, playing[i].port, output.GetBufferPtr(), output.GetByteLength());
        playing.erase(playing.begin() + i);
        break;
    }
}

for (int i = 0; i < playing.size(); i++) // Busca el contrincant i desconectarlo
{
    if (playing[i].id != com.front().id && playing[i].matchId == matchid)
    {
        OutputMemoryBitStream output;
        output.Write(DISCONNECTION, TYPE_SIZE);
        sender.SendMessages(playing[i].ip, playing[i].port, output.GetBufferPtr(), output.GetByteLength());
        playing.erase(playing.begin() + i);
    }
}
```

Cuando termina el juego, se eliminan los jugadores de la lista Playing

Dificultades y conclusiones.

El gran problema han sido los errores de principiante como condiciones con un solo ‘=’ o igualar variables con ‘==’. Referente al matchmaking, el mayor reto ha sido convertir el servidor para que funcionase con jugadores jugando y conectándose al mismo tiempo, pero gracias a tener el Receiver en un thread aparte, se ha podido solventar sin muchos problemas por el camino.