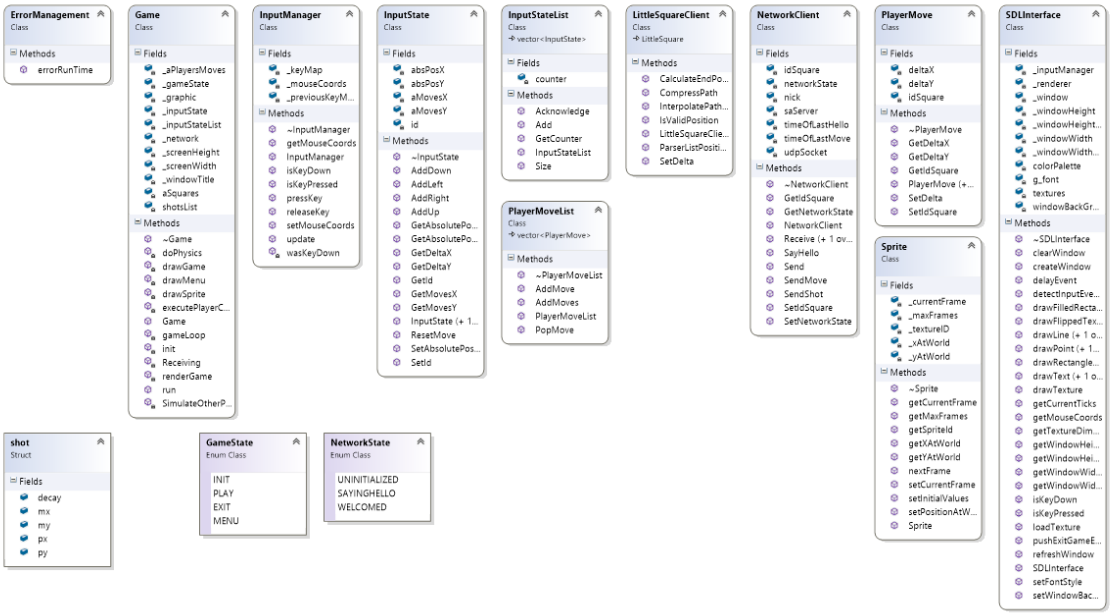
# Report: Tooty Tooty Point-and-Shooty

**Carlos Olivier**

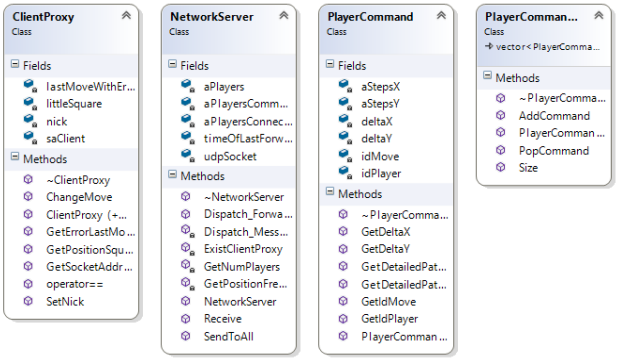
**Edu Giménez**

# DIAGRAMA DE CLASES

## Client




## Server



**PACKET TYPES**

**EMPTY ->** Guarda el int del enum de packettypes, para hacer un dispatch de los mensajes.

Como tenemos más de 7 paquetes, necesitamos utilizar 4 bits por cada packettype.

“
```
InputMemoryBitStream imbs
PacketType pt = PacketType::PT_EMPTY;
imbs.Read(&pt, 4);
```
”

**HELLO ->** Envia el nick del jugador y "pide permiso" para entrar en la partida.

“
```
OutputMemoryBitStream ombs;
ombs.Write(PacketType::PT_HELLO, 4);
ombs.WriteString(nick);
```
”

**WELCOME->** Añade el jugador a la lista de jugadores y pasa sus atributos.

“
```
OutputMemoryBitStream ombs;
ombs.Write(PacketType::PT_WELCOME, 4);
ombs.Write(freePosition, 2);
ombs.Write(numPlayers, 2);
...
```
”

**FULL->** Indica que el servidor está lleno.

“
```
OutputMemoryBitStream ombs;
ombs.Write(PacketType::PT_FULL, 4);
```
”

**TRYPOSITION->** Envía la id del jugador, los movimientos que ha realizado y la posición a la que se quiere mover (separada en el int X y el int Y).

“
```
ombs.Write(PacketType::PT_TRYPOSITION, 4);
ombs.Write(_inputState.GetId());
ombs.Write(aMovesX);
ombs.Write(aMovesY);
```
”

**POSITION->** Después de recibir Tryposition, devuelve la posición corregida a todos los clientes, separada igual que tryposition y con la id para que cada cliente actúe en consecuencia dependiendo de si debe simular un jugador ajeno o corregir su propia posición.

**DISCONNECT->** Indica desconexión de un cliente al servidor.

“
```
OutputMemoryBitStream ombs;
ombs.Write(PacketType::PT_DISCONNECT, 4);
ombs.Write(_network.GetIdSquare());
```
”

**SHOOT->** Se envía la id del jugador que dispara, y las posiciones inicial y final de la representación gráfica del disparo.

“

```
OutputMemoryBitStream ombs;
ombs.Write(PacketType::PT_SHOOT, 4);
ombs.Write(_inputState.GetId());
ombs.Write(_posSquareX, 10);
ombs.Write(_posSquareY, 10);
ombs.Write(_posShotX, 10);
ombs.Write(_posShotY, 10);
```

”

**INIT->** Al entrar en partida, se manda a todos los jugadores ya en ella para que actualicen tu posición inicial.

“

```
OutputMemoryBitStream ombs;
ombs.Write(PacketType::PT_INIT, 4);
ombs.Write(freePosition, 2);
ombs.Write(aPlayers[freePosition].GetPositionSquare().first, 10);
ombs.Write(aPlayers[freePosition].GetPositionSquare().second, 10);
```

”

**SERVIDOR**

El servidor almacena para cada jugador:

Su posición: Para comprobar si están dentro de los límites, si están recibiendo daño del enemigo, etc.

Su id: Para saber cuál de los jugadores es.

Su munición y vida: Atributos esenciales en el juego que afectaran directamente a este, por lo tanto, deben ser comprobados por el servidor.

La actualización global se manda a los jugadores cada 100 ms (mismo lapso entre envíos de acumulación de movimientos desde estos al servidor). El tiempo de respuesta es amplio pero la simulación por cliente hace que se vea fluido igualmente y aparte está ajustada al tiempo de recarga de armas.
Durante la partida es importante saber si tu disparo ha llegado inmediatamente, así que los cálculos y envío de paquetes para representación gráfica se hacen automáticamente. Las armas tienen un tiempo de recarga amplio, siempre superior a medio segundo, lo que hace que con un espacio entre mensajes de 100 ms nunca se sobresature el servidor, aun cuando todos los jugadores maximicen el numero de disparos por segundo que pueden realizar.

**SIMULACIÓN EN CLIENTE**

La posición de los jugadores es un pair<int, int> ya que se mueven en 2D y los movimientos se almacenan en 2 vectores de int distintos: uno para el eje X y otro para el eje Y.

Los movimientos realizados por cada cliente se almacenan en un vector de deltas local doble repartido entre los movimientos en X e Y. Cada posición de los vectores corresponde a un movimiento. Al llegar al tiempo de envió, se manda ambos vectores y la posición final al servidor.

Cuando el servidor recibe los datos, junta de nuevo los movimientos X e Y y los valida como uno solo. Si un jugador realiza un movimiento ilegal, se le devuelve automáticamente a la última posición legal, efectivamente haciendo saltar desde la posición incorrecta a la correcta.

**MANUAL DE USUARIO**

Para disfrutar de la versión actual de TootyTootyPointAndShooty abra el servidor y 4 clientes.

Tendrá usted por pantalla 4 cuadrados que no pueden salir de los bordes y que, si toca la barra espaciadora, efectúan un disparo en línea recta hacia donde tenga el cursor del mouse.

Se mueven con las flechas.