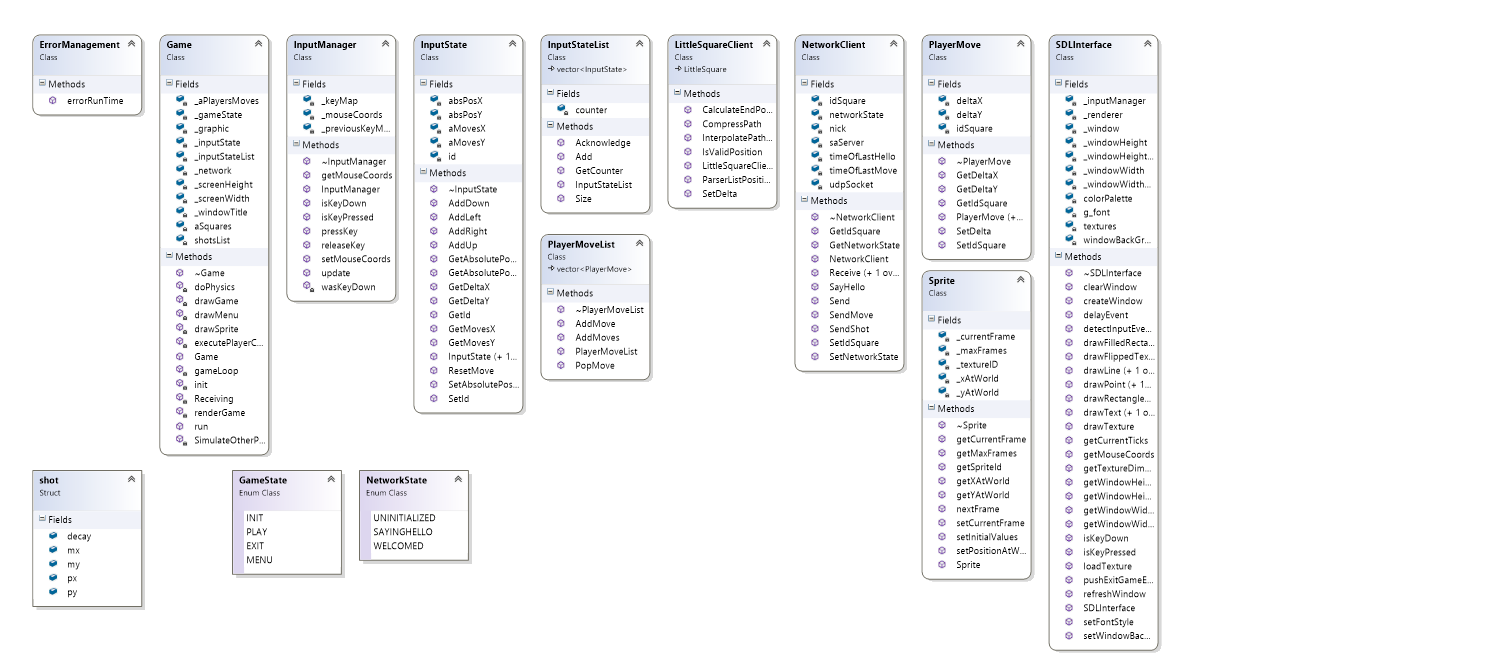
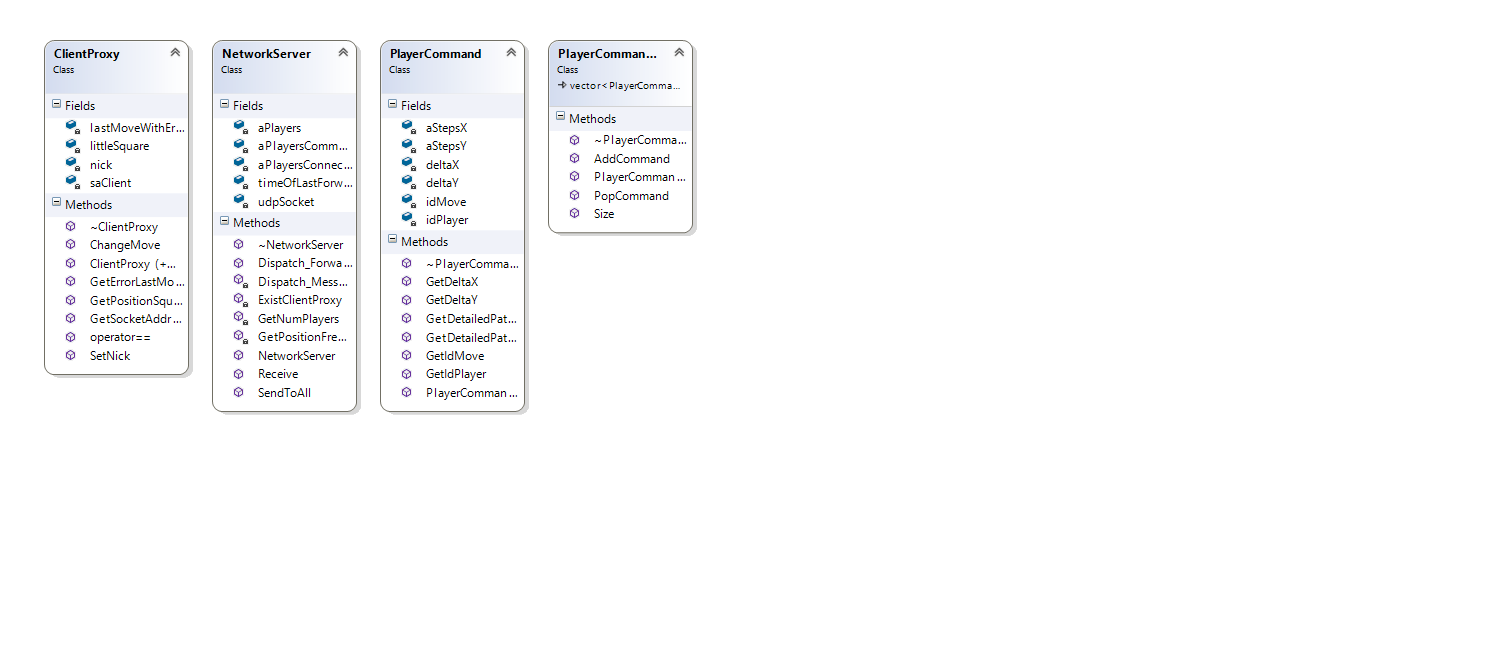
**Report: Tooty Tooty Point-and-Shooty**

**1. Diagrama de clases**

**Client**



**Server**



**2. Detalle de todos los mensajes que intercambian cliente y servidor**

**a. Si se está optimizando la serialización explicar en qué campos y porqué.**

PacketTypes:

EMPTY, HELLO, WELCOME, FULL, TRYPOSITION, POSITION, DISCONNECT, SHOOT, INIT

Para cada mensaje:

**EMPTY ->** estructura packettype utilizada para guardar 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 ->** es un write de un const char utilizado para enviar el Nick.

“ OutputMemoryBitStream ombs;

ombs.Write(PacketType::PT\_HELLO, 4);

ombs.WriteString(nick);

”

**WELCOME->** Utilizado para añadir al jugador a la lista de jugadores y pasar sus atributos.

“

OutputMemoryBitStream ombs;

ombs.Write(PacketType::PT\_WELCOME, 4);

ombs.Write(freePosition, 2); //Freeposition es la posición del vector donde no hay jugador

ombs.Write(numPlayers, 2);

...

”

**FULL->** Es un mensaje para indicar que el servidor está lleno.

“

OutputMemoryBitStream ombs;

ombs.Write(PacketType::PT\_FULL, 4);

”

**TRYPOSITION->** Se envía con la id del jugador 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); //hasta 10 ya que la pantalla

ombs.Write(aMovesY); // llega a hacer 700

”

**POSITION->** Se envía con la posición corregida al cliente, separada igual que tryposition y con la id.

**DISCONNECT->** Se utiliza para indicar la desconexión. Se envía la id del que se desconecta.

“

OutputMemoryBitStream ombs;

ombs.Write(PacketType::PT\_DISCONNECT, 4);

ombs.Write(\_network.GetIdSquare());

”

**SHOOT->** Se envía la id del jugador que dispara, la posición del media de este y la posición del mouse al efectuar el 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->** Para reinicializar a un jugador, se pasa la position del vector de jugadores y la posición.

“

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);

”

**3. Qué información se guarda en servidor por cada cliente y porqué.**

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.

**4. Justificación de los tiempos en ms escogidos para enviar los eventos de cliente y el tiempo de respuesta del servidor.**

100 Carlos

**5. De qué forma y con qué estructuras de datos simula el cliente su movimiento.**

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.

1. **Qué ocurre a nivel de estas estructuras cuando el servidor valida las acciones.**

Rompemos el pair int en su first y second y los asignamos a los vectores de X e Y respectivamente. Entonces de estos hacemos la interpolación en los ejes y en las diagonales.

1. **Qué ocurre cuando el servidor le corrige.**

El jugador corregido salta desde la posición incorrecta a la correcta, corregida por la interpolación en los ejes como la diagonal.

1. **Explicar cómo se puede comprobar que el servidor corrige al jugador cuando intenta hacer algo mal.**

Un jugador puede intentar salirse de uno de los bordes de la pantalla, por ejemplo.

**6. Explicar la interpolación que se ha utilizado. Estructuras de datos utilizadas. Cómo y cuándo se simula el movimiento de los otros jugadores.**

Carlos, help. Carlos

**7. Si se implementa pérdida de paquetes:**

Pérdida de paquetes no implementada.

**a. Explicar en qué comunicación se intenta controlar la pérdida de paquetes.**

**b. Explicar si se utiliza técnica de ACKs o de control del estado más reciente.**

**c. Explicar cómo se ha implementado.**

**8. Si se implementa matchmaking:**

Matchmaking no implementado

**a. Explicar que estructuras de datos y clases habéis añadido de nuevo para implementarlo.**

**b. Explicar el tipo de matchmaking que implementáis.**

**9. Breve manual de usuario**

Para disfrutar de la pre-pre-alpha 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.