[AA3] - Práctica Bloque 2: UDP

Ítem 1: Qué juego vamos a implementar

La idea del "juego" que vamos a implementar nace de este juego concretamente ("Never Split the Party"):

https://www.youtube.com/watch?v=OX6IagIf4J4

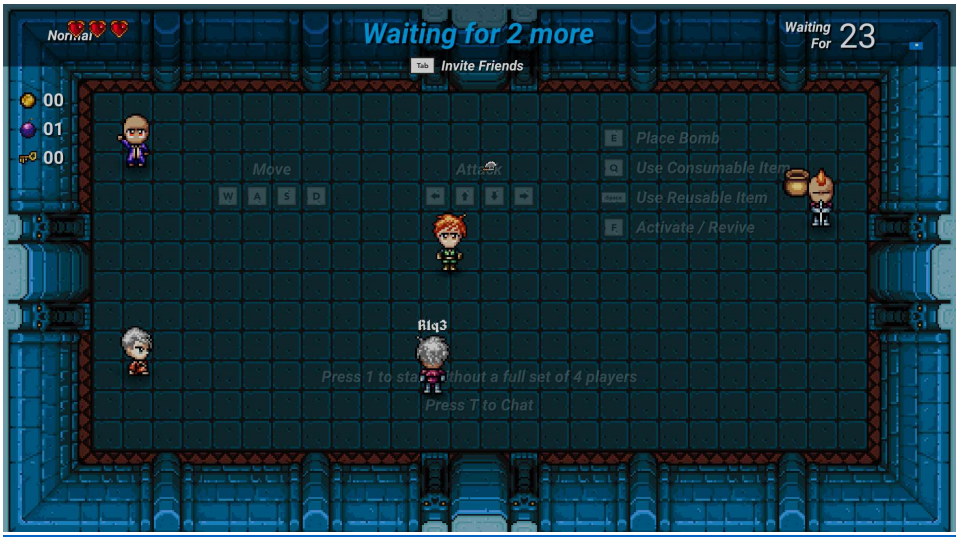

Ilustración 1: Lobby

Ítem 2: Cómo funcionará el juego

Los clientes llegan y se conectan a la primera partida disponible o bien se crea una nueva para ellos.

A medida que se incorporan en una partida, los jugadores quedan en un lobby (Ilustración 1: Lobby) en el que pueden practicar: moverse y disparar, tal y cómo se muestra en el vídeo (en la parte del principio). En esta pantalla cada cierto tiempo aparece algún ítem que pueden romper/matar para practicar su disparo.

Cuando llegan 2 jugadores a la partida, empieza una cuenta atrás de 30 segundos y al llegar a cero, se destruye la partida y se pregunta al jugador si quiere jugar de nuevo. No vamos a implementar la partida en sí. Por supuesto, el que quiera y se vea con tiempo, puede hacerlo.

Ítem 3: Traducción del punto anterior a nivel técnico

Los tres párrafos anteriores a nivel técnico implican lo siguiente:

- Establecimiento de conexión de jugadores al servidor.
- *Matchmaking*
- Detección de la desconexión
- Tratar al menos uno de los paquetes entre cliente y servidor o entre servidor y cliente como paquete crítico
- Tener constancia del *RTT*
- Compresión de datos
- Tratamiento de comandos: Movimiento y disparo

A continuación, se explican más en detalle cada uno de estos puntos:

Ítem 3.1: Establecimiento de conexión

Para implementar el establecimiento de la conexión debéis seguir las indicaciones que se dan en el ejercicio 1 de UDP: Versión de HELLO – WELCOME mejorada con challenge y client/server salt.

Podéis adaptar el mensaje de HELLO y WELCOME para pasar información que sea relevante para el juego.

Recuerda ir anotando todos los mensajes necesarios para esta fase en tu protocolo de comunicación.

Ítem 3.2: Desconexión

Para el control de la desconexión:

- Se debe implementar la desconexión "limpia". Si el cliente se desconecta, avisa al servidor y también en sentido inverso.
- Se debe implementar la desconexión por inactividad. Si el cliente está sin enviar comandos durante un tiempo (30 segundos), se le desconecta, avisando a los demás jugadores de que se le ha desconectado.

Recuerda ir anotando todos los mensajes necesarios para esta fase en tu protocolo de comunicación.

Ítem 3.3: Paquetes críticos

Se debe implementar el tratamiento de al menos un paquete crítico. Puede ser de cliente hacia servidor o de servidor hacia cliente.

Algún ejemplo de paquete crítico a tratar sería:

- El paquete que avisa a los jugadores de que acaba de conectarse un jugador nuevo.
- El paquete que marca el inicio de partida.

Decidid cuál será el paquete que vais a tratar como crítico y generad los mensajes de protocolo necesarios para que sea tratado.

Recuerda añadir el código necesario para poder simular la pérdida de paquetes y comprobar que te recuperas correctamente reenviando el paquete que se ha perdido.

Ítem 3.4: RTT Log

Nos interesa tener constancia de las variaciones que de RTT hacia los clientes.

Implementa un sistema de log para tener constancia de ello siguiendo las indicaciones del ejercicio 3 de UDP.

Ítem 3.5: Compresión de datos

Para los mensajes de protocolo que llevas definidos hasta el momento y para los que tienes que incorporar a partir de ahora, debes aplicar optimización a nivel de bit.

Debes indicar qué optimizaciones has realizado a nivel de bit. Dejo un ejemplo:

Estructura mensaje	**Tipos**	**Número de bits**	**Explicación**
HELLO_<Nick>	Int_Int_string	3_3_88888888	8 tipos de paquetes diferentes (3 bits) Máxima longitud del Nick: 8 caracteres (3 bits) Cada uno de los caracteres del Nick (8 bits)

Puedes utilizar los *includes* que se os han facilitado o implementar los vuestros.

Item 3.6: Tratamiento de los comandos

Los comandos de juego los trataremos como paquetes de los que sólo importa el estado más reciente. Así que no se solicitarán ACKs.

Sin embargo, continuamos teniendo un servidor autoritativo, con lo que toda acción que quiera realizar un jugador debe ser validada por el servidor. Como se ha visto en clase, esto aporta seguridad, pero conlleva determinados problemas que solucionaremos con las técnicas de:

- Acumulación en cliente
- Acumulación en servidor
- Predicción
- Reconciliación
- Interpolación

Los comandos a implementar son:

- Movimiento: Teclas WASD
- Disparo: Flechas de dirección

Ítem 3.7: *Matchmaking*

En la práctica de TCP, las partidas las creaba cada jugador y también era el mismo jugador el que decidía a qué partida unirse.

En *matchmaking* se incorporará a los jugadores automáticamente a una partida en el momento en que decidan que quieren jugar. De la misma forma, cuando finalice la partida no se expulsa al jugador del servidor, sino que se le pregunta si quieres volver a jugar otra partida.

El criterio para agrupar a dos jugadores en la misma partida será a partir del primer carácter del nick. Si la diferencia es menor a 10 unidades respecto al primero que entra, puede estar en esa partida, sino se le crea una nueva para él.

Entregable de la práctica → Código

1. Nos aseguramos de que el proyecto no utiliza paths absolutos para linkar las librerías.
2. Nos aseguramos de eliminar la carpeta oculta .vs.
3. Hacemos un zip de la solución
4. Lo subimos al classlife en este formato:

PracticaUDP_Nombre1Apellido1Nombre2Apellido2.zip

Entregable de la práctica → Documentación

1. Protocolo de comunicación.
2. Optimización aplicada sobre el protocolo de comunicación.