TP Videojuegos 2

Práctica 3

Fecha Límite: 01/06/2020 a las 09:00.

El objetivo de esta práctica es usar `SDL_Net` para desarrollar un juego en red entre dos jugadores. El juego consiste en dos cazas (controlados por dos jugadores) que pueden disparar uno al otro hasta que al menos uno de ellos gana 3 puntos. ***Es muy importante estudiar el ejemplo `Ping Pong` y entenderlo muy bien antes de empezar con esta práctica.***

La primera parte vale el `80%` y la segunda el `20%`. No es obligatoria hacer la segunda parte.

PARTE I (80%)

Junto a este enunciado vais a recibir una versión del juego que no usa `SDL_Net`, es decir los dos jugadores juegan en la misma ventana a la vez usando teclas distintas (para saber que teclas ver el método `init()` en `FightersSystem.cpp`). Hay que modificar esta versión para que los dos jugadores jueguen en red. ***Antes de modificar el código hay que estudiarlo muy bien para ver qué sistemas y entidades tiene, cómo funcionan, etc.***

El código ya tiene todas las clases necesarias (que usamos en el ejemplo *`Ping Pong`*) para convertirlo en juego en red:

1. **`NetWorking`**: incluye el servidor y todo lo necesario para la comunicación entre clientes.
2. **`SDLGame`**: ya tiene el método para obtener una instancia de la clase `NetWorking`.
3. **`Manager`**: ya tiene el mecanismo de mensajes, etc. Ver `messages.h` también.
4. **`NetworkingSystem`**: un *proxy* para enviar/recibir mensajes entre clientes. Recuerda que para cada mensaje que defines en `messages.h`, hay que definir un caso correspondiente en el método `update()` de `NetworkingSystem.cpp`

Después de haber entendido cómo funciona el juego, hay que convertirlo en un juego en red siguiendo lo que hemos hecho para el juego *`Ping Pong`*. En particular:

1. La forma de ejecutar el juego tiene que ser como la del *`Ping Pong`*. Se puede copiar el `main.cpp` del *`Ping Pong`* y modificarlo para que use el nuevo juego.

2. Como el *Ping Pong,* el texto "*Waiting for the other player*" tiene que aparecer hasta que el otro jugador conecta (o cuando un jugador desconecta)
3. El mecanismo para empezar el juego tiene que ser como el del *Ping Pong*, es decir, sólo el cliente 0 puede empezar el juego (enviando un mensaje adecuado) y el cliente 1 puede pedir al cliente 0 que lo empieza (enviando un mensaje adecuado).
4. Para evitar conflictos, sólo el *cliente 0* comprueba colisiones y comunica toda la información necesaria al cliente 1.
5. Los dos jugadores tienen que usar las flechas para mover el caza y espacio para disparar.
6. Cuando acaba el juego, tiene que aparecer un mensaje adecuado para cada jugador (`YOU LOSE`, `YOU WIN`, `DRAW`), depende quien ha ganado -- gana el que llega a tres puntos antes y hay empate si los dos tienen 3 puntos).
7. Modifica el juego para que compruebe la collision entre los dos cazas (en este caso los dos mueren, pero ninguno gana un punto).

PARTE II (20%)

Modifica el juego para que el jugador pueda introducir su nombre como parámetro al empezar el juengo

```
../bin/TPV2Debug.exe client localhost Calvin
```

El nombre es una cadena de caracteres sin espacios, es opcional, si no aparece hay que usar "`Anonymous`". El nombre tiene que tener longitud máxima de 10 caracteres (comprobar usando `std::strlen`), en otro caso muestra un mensaje de error. Hay que pasar el nombre a la constructora de la clase `StarWars` (como `host` y `port`) y después en `initGame()` pasarlo a la clase `Manager` para poder consultarlo desde cualquier parte del juego (añadir un método "`const char* getName()`" a `Manager`)

Durante el juego, los nombres de los jugadores tienen que aparecer en la parte superior de la pantalla, uno a la izquierda y otro a la derecha. El nombre del jugador local tiene que aparecer con un fondo blanco. Para que un cliente comunique su nombre al otro, el mensaje correspondiente incluirá un atributo "`char name[11]`" al que hay que copiar el nombre del jugador usando el método `std::strcpy`.