TP Videojuegos 2

Práctica 3

Fecha Límite: `05/05/2025` a las `16:30`.

Antes de empezar

- Antes de empezar, descarga el juego ejemplo (los cazas en red) que desarrollamos en clase y estúdialo bien (está con los ejemplos de del tema **`SDL_Net`**). Leer el archivo **`README.es.md`** para ver cómo ejecutarlo, etc. Lo que tienes que hacer en esta práctica es muy parecido.
- El código que escribes en esta práctica puede hacer uso del código del juego de ejemplo, pero hay que limpiarlo (cambiar nombres de variables y métodos para hacer referencia al nuevo juego si hace falta, eliminar código que no se usa, etc).
- Para obtener la máxima nota hay que añadir la funcionalidad del apartado "funcionalidad adicional"

El juego LittleWolf sin red

LittleWolf es una modificación de la versión original[1] para que tenga varios personajes que disparan uno al otro (pero sólo uno de los jugadores es activo a la vez). Los personajes están dibujados como columnas, ya que incluir una imagen (en **`3D`**) me resultaba difícil porque todo se dibuja usando la técnica raycast.

Al ejecutar el juego **`(1)`** se carga el mapa desde **`resources/maps/little_wolf/map_0.json`** (los números positivos entre **`1`** y **`9`** representan muros de diferentes colores y el **`0`** es espacio vacío); **`(2)`** se añaden **`4`** jugadores al juego. El jugador actual puede navegar usando las siguientes teclas: **`LH`** para girar (con **`LEFT SHIFT`** lo hace más despacio), **`WS`** para avanzar y retroceder, y **`DA`** para desplazarse lateralmente. La tecla **`SPACE`** se usa para disparar. Cuando un jugador dispara a otro, el que muero se dibuja como columna con poca altura. Un jugador muerto no puede navegar ni disparar y su vista cambia a vista aérea. Usando la tecla **`N`** se puede mover al siguiente jugador, y usando la tecla **`R`** se puede reiniciar el juego (todos los muertos vuelven a la vida). **`ESC`** sale del juego.

El juego mantiene **`2`** mapas, el original que se ha cargado desde el archivo **`map_0.json`** y otro con resolución más alta para poder colocar los jugadores de manera más fácil en una casilla vacía del mapa. Para colocar un jugador con identificador **`n`** en el mapa, el código usa el número **`10+n`** (no se añade en el **`map_0.json`**, se elige la posición de manera aleatoria en el método **`addPlayer`**).

La carpeta **`resources`** que se usa en este juego está incluida en el archivo zip correspondiente.

Antes de seguir, asegúrate de que hayas estudiado bien el código de este juego, no hace falta entrar en los detalles de cómo se dibujan los muros, etc.

[1] https://glouw.com/2018/03/11/littlewolf.html

El juego LittleWolf en red

El objetivo de esta práctica es usar **`SDL_Net`** para desarrollar una versión red de LittleWolf, donde los jugadores juegan a la vez en ordenadores distintos, parecido a lo que hemos hecho con el juego de los cazas en clase. El juego tiene que tener las siguientes características:

1. Siempre hay un jugador **máster** que toma las decisiones como se indican algunos puntos a continuación. Cuando el master desconecta, el servidor asigna como master al jugador con el menor identificador. El juego tiene que seguir igual pero con el nuevo máster. El código del servidor ya lo tenéis con el juego de ejemplo.

2. El primer jugador que conecta puede navegar directamente, esperando a que otros jugadores conecten. Y cada jugador que conecta entra en el juego directamente.

3. Cuando quedan menos de **`2`** jugadores (sólo por la razón de morir, no porque unos han abandonado) se muestra el mensaje **`"The game will restart in X seconds"`** y pasados los **`5`** segundos ya empezará el juego con los mismos jugadores pero con posiciones aleatorias que las decide el máster para todos los jugadores. Durante el tiempo de espera ningún jugador puede navegar.

4. Cada jugador tiene la vista normal (lo que ve por delante), pero cuando muere la vista cambia a una vista aérea para poder ver lo que está pasando con los otros jugadores. Esta funcionalidad ya existe en la versión sin red. Si quieres, puedes permitir que los jugadores cambien a vista aérea cuando están vivos también, pulsando alguna tecla.

5. Los jugadores pueden abandonar el juego en cualquier momento pulsando la tecla **`ESC`**, el resto de jugadores siguen jugando igual. Para eso los jugadores tienen que *avisar con un mensaje adecuado al salir del bucle principal* (como en el juego de ejemplo). Recuerda que cuando el master abandona habrá un nuevo máster, y cuando conecta uno nuevo con identificador menor del máster actual habrá un cambio de máster (esto lo decide el servidor ya, tú tienes que tenerlo en cuenta al nivel del juego exactamente como en el juego de ejemplo).

6. Para sincronizar los jugadores se puede usar la siguiente estrategia. Después de cada movimiento cada jugador avisa a todos la información del movimiento (posición anterior, posición actual, velocidad actual, rotación actual, etc). Cuando el master recibe ese mensaje tiene que comprobar que no haya conflictos, es decir en el mapa del master ese movimiento es válido (el jugador estaba en su posición anterior y la nueva posición – si es distinta – no está ocupada), en caso contrario avisa a todos los jugadores las posiciones actuales (como en su mapa) que tienen que tener para resolver el conflicto. Puedes usar otra estrategia si quieres, pero es muy importante tener los conflictos en cuenta.

7. Cuando un jugador dispara solo envía un mensaje diciendo que se ha disparado y el máster tiene que decidir si el disparo choca con algún jugador y avisar a todos quien haya muerto. El jugador que ha disparado no toma la decisión.

8. Añadir sonidos de disparo y grito de dolor cuando un jugador muere, no solo en la parte del jugador que haya disparado/muerto, sino en todos los clientes con volumen relativo a la distancia del jugador que haya disparado/muerto.

9. Eliminar las siguientes características del juego original (ya que no tienen sentido en el juego en red): pulsar **R** para reiniciar, pulsar **N** para cambiar de jugador.

Funcionalidad adicional

Para obtener la nota máxima hay que añadir las siguientes características al juego:

1. Hacer que los jugadores tengan un nivel de vida, y en cada disparo que reciben pierden un porcentaje depende de la distancia del disparo y mueren cuando quedan con cero vida. Hay que mostrar la vida de todos los jugadores.
2. Añadir marcadores, cuando un jugador dispara a otro gana un punto. Los marcadores no se resetean cuando empieza un nuevo juego. Hay que mostrar los marcadores de todos los jugadores.
3. Cuando empieza el juego, pregunta al usuario su nombre usando **`std::cin`** o pasándolo como parámetro por consola (hasta **`10`** caracteres), y muestra todos los nombres al lado (o en lugar) de los identificadores. Para que uno envíe su nombre a los otros jugadores, se puede añadir un atributo "`char name[11]`" al mensaje de petición de conexión (usamos `11` porque la última posición es para el `0` que indica el final de una cadena de caracteres estilo `C`). Se puede usar los siguientes métodos para convertir entre `std::string` y cadenas de caracteres estilo `C` de ese tamaño:

```
void string_to_chars(std::string &str, char c_str[11]) {

      auto i = 0u;
      for (; i < str.size() && i < 10; i++) c_str[i] = str[i];
      c_str[i] = 0;
}

void chars_to_string(std::string &str, char c_str[11]) {
      c_str[10] = 0;
      str = std::string(c_str);
}
```

   Para la "serialización" y "deserialización" de un array **`x`** con tamaño **`11`** se puede usar

```
_IMPL_SERIALIAZION_WITH_BASE_(...,x,11u,...)
```

4. ¡Puedes añadir cualquier funcionalidad adicional que deseas!