Endless Run en Primer Persona

Curso 2k20/2k21

Apellidos, nombre	Clérigues Ferrer, David (daclefe1@inf.upv.es) Alcarria Lozano, Pablo (pabloal1@inf.upv.es)
Titulación	Grado de Ingeniería informática
Fecha	Abril 2021

Contenido

Índice de figuras

1 Resumen de las ideas clave

La idea principal es crear un juego tipo "endless runner" (corredor infinito), un género en el que el personaje que se controla avanza de forma continua, y generalmente debe esquivar los obstáculos mediante saltos o movimientos laterales. Se utiliza OpenGL para el escenario 3D y OpenAL para los sonidos, de forma que con el uso de las librerías se cree un entorno lo más inmersivo y frenético posible.

2 Introducción

La idea que se ha llevado a cabo en este proyecto es la creación de un juego tipo "endless runner" mediante librerías como OpenGL con ayuda de OpenAL para el sonido, estudiadas en la asignatura Sistemas Multimedia Interactivos e Inmersivos. Para su creación se ha partido de un ejemplo base de OpenGL en el que ya había creado un escenario tridimensional con un suelo, y a partir de éste se han diseñado los obstáculos, así como se ha modificado el movimiento para que avance de forma continua hacia adelante y la cámara quede fija, de forma que haya una sensación de que existen 3 carriles en el juego. Además, mediante OpenAL se han añadido canciones y efectos de sonidos para dar una mayor sensación de inmersión en el juego, y se estudian distintos tipos de controlar el personaje: con las teclas de un teclado, con el movimiento de un ratón y con el mando de la Nintendo Wii.

3 Objetivos

- Lograr entender el uso básico de OpenGL para poder desarrollar un juego interactivo.
- Ser foco de interés y objeto estudio para futuros alumnos ya que pensamos que nuestra temática puede dar juego a un gran abanico de posibilidades.
- Implementar un control intuitivo y comprender el funcionamiento de librerías externas como cwiid o simple wiimote library.
- Finalizar un proyecto final con las expectativas deseadas tanto por el profesor como por los integrantes.

4 Pasos del desarrollo

Nuestro proyecto ha sido desarrollado a partir de una base extraída de internet, dentro de **[1]** podemos encontrar el código base en el cual está inspirado nuestro juego. Muchos alumnos de este curso 2020/21 reconocerán tal programa ya que ha sido foco de estudio en la práctica 1.

4.1 Diseñar un escenario limitado para el transcurso del movimiento

Este punto no ha sido desarrollado por nosotros como tal ya que el propio programa tiene incluido un escenario generado. El escenario de opengl-3D-sample.c **[1]** contiene un grupo de muñecos de nieve situados sobre el suelo, además, para darle profundidad el fondo está pintado de azul para simular el cielo.

Para el futuro desarrollo nos interesa observar cómo está construido el escenario para poder adaptarlo a nuestras necesidades. Adjuntamos en la **figura 1** el aspecto base sobre el cual partimos.

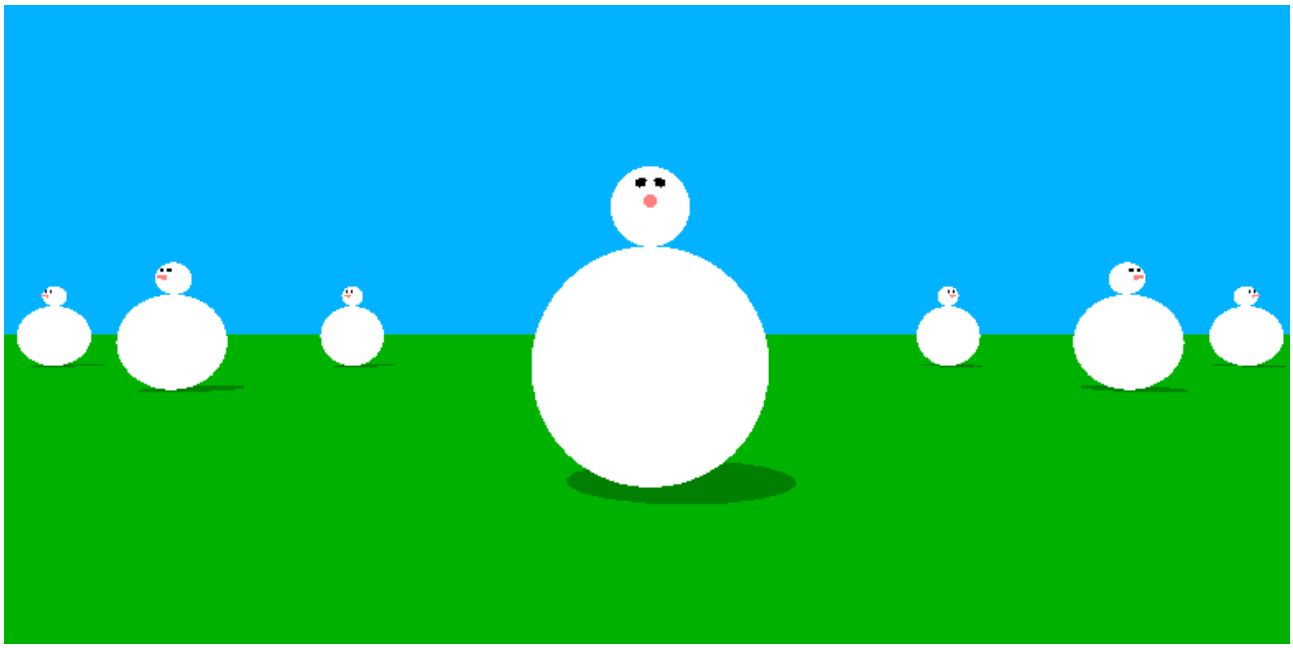

Figura 1 - Escenario base

4.2 Permitir el movimiento en el escenario de forma libre

Dentro de este punto se busca obtener un movimiento libre por el escenario, es decir, poder moverse sobre cualquier dirección de los ejes x e y. Este punto, al igual que el anterior, no ha sido un foco por desarrollar ya que, por suerte, el propio ejemplo también tenía implementado el movimiento sobre los ejes x e y junto con la actualización de la cámara. En la **figura 2** se muestran dos imágenes tomadas desde distintos ángulos.

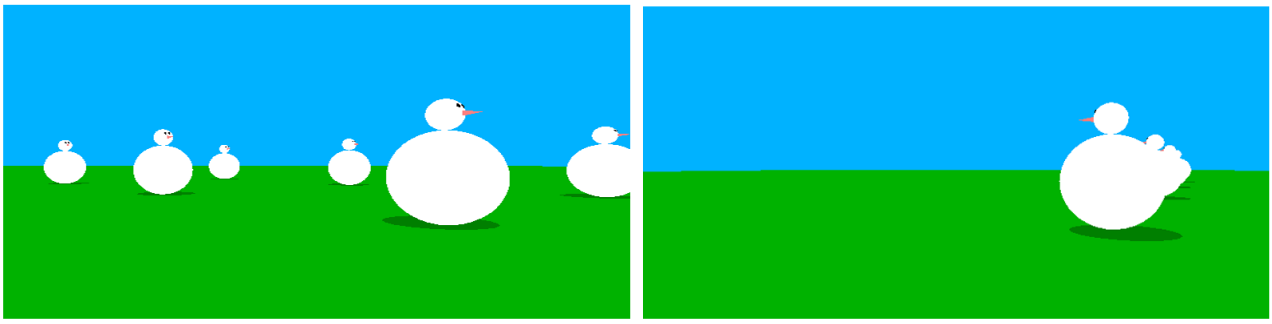

Figura 2 - Movimiento libre

4.3 Adaptar el movimiento libre a los estándares del juego

Dentro de este punto vamos a repasar y explicar cómo hemos sido capaces de modificar el código base para que se adapte a nuestras expectativas de movimiento, pero… ¿Cuáles son estas? En un principio nosotros buscamos recrear un juego del estilo Endless Run **[2]** en primera persona empleando un entorno 3D, para ello debemos tener en mente juegos populares de este género como Temple Run **[3]** o Subway Surfers **[4].**

Estos ejemplos emplean vista isométrica a diferencia de nuestro juego que busca emplear una cámara en primera persona, pero los conceptos base son los mismos. Queremos emplean tres carriles para el movimiento y dentro de estos pasarán cosas las cuales se explicarán más adelante. Como ya tenemos en mente el futuro escenario tan solo falta adaptar las funciones que se encargaran de realizar el movimiento.

El código sobre el cual partimos permite moverse libremente, es por ello, que dicha implementación va a ser borrada. Para ello, nos hemos encargado de borrar las actualizaciones a las variables de cámara lx e ly junto con los deltaMove. Una vez suprimidos estos valores tan solo nos queda disponible el movimiento en línea recta hacía adelante y hacía atrás. Como el retroceder no está contemplado en nuestro resultado final simplemente ha sido necesario suprimirlo de las funciones de movimiento.

Llegados a este punto ya tenemos una base sobre la cual trabajar, el carril central ya está implementado con dichas modificaciones entonces solo debemos centrarnos de momento en los movimientos laterales, para ello la idea empleada es añadir sobre la función "**pressSpecialKey**" unos nuevos casos donde al pulsar sobre las flechas laterales del teclado se permita realizar el movimiento hacía la izquierda o hacía la derecha, tan solo bastará actualizar el valor del eje x de momento dándole un valor positivo o negativo dependiendo el movimiento deseado.

Con toda esta implementación ya estamos cerca de finalizar el movimiento básico ya que de momento es posible moverse por tres carriles (aunque el mapa aún no está adaptado, sigue siendo el prado extenso), tras esto es necesario suprimir el avance manual y automatizarlo. Esta parte es sencilla ya que simplemente hemos suprimido el uso de esta tecla (Flecha Superior) y hemos modificado la función "**update**" para que actualice el eje en cada pasada del bucle.

Llegados a este punto tenemos el movimiento adaptado a las bases que queremos, pero hemos decidido realizar unos ajustes debido a los problemas que presenta dicho movimiento. Parece ser que la actualización de movimiento es tan rápida que podemos movernos entre extremos, pero no tenemos la opción de volver al carril central una vez se ha decidido mover el usuario, es por ello, que hemos configurado el programa de forma que cada vez que dejamos de pulsar una de flecha lateral, se consiga que la cámara vuelva al carril central. Además, hemos habilitado las teclas ESC/Q/q para salir del jugo y E/e para pausar el juego.

4.4 Modelaje de objetos con primitivas básicas de OpenGL y adaptación del escenario

En las **figuras 1 y 2** hemos mostrado el escenario base del cual se parte, pero este escenario no es el que buscamos ya que como hemos mencionado en el punto anterior necesitamos emplear tres carriles. Para ello, hemos optado por una temática del oeste al más puro estilo ruta 66.

En primer lugar, tras localizar las operaciones encargadas de dibujar el suelo hemos optado por adaptar tanto el ancho como el largo de este suelo. Buscamos formar un rectángulo que simule una carretera por lo tanto partimos de la base de que el ancho tiene que ser muchísimo más corto que la altura. Además, no tan solo queremos una carretera, sino que queremos dar profundidad al juego situando un escenario por fuera de los arcenes, siendo en este caso un paisaje arenoso.

El ancho de la carretera ha sido fijado a unas 20 unidades (Partiendo del eje 0,0 son unas 10 unidades por lado) y por otra parte, como el escenario arenoso debe sobresalir su ancho debe ser más grande por tanto, se ha fijado a unas 60 unidades (30 por lado).

Con todo esto los ejes x han sido adaptados el único problema es que hay solapamiento en el renderizado ya que tiene el mismo eje z, es por eso por lo que la carretera tendrá un eje z con valor 0.1 y el mapa arenoso un eje z de 0. Por último, queda actualizar el eje y pero la premisa del juego es que dicho eje sea infinito es por ello que de momento, el valor del eje y será bastante grande (unas 200 unidades) pero no infinito.

Más adelante se cubrirá este apartado. Para finalizar el escenario, tan solo queda actualizar los colores dejando así un resultado como el mostrado en a **figura 3** de todo el proceso dado.

Figura 3 - Escenario Endless

Llegados a este punto hemos adaptado el escenario y el movimiento (se ha vuelto a actualizar con las medidas proporcionadas) pero aún falta renderizar distintos objetos por el mapa. La idea principal es generar unas vallas y unos cuantos cactus con las primitivas básicas de OpenGL, en este caso se va a emplear las funciones "**glutSolidCube**" y se le van a aplicar los métodos "**glScalef**" y "**glTranslatef**" para darles forma. Dentro de la **figura 4** se muestra el transcurso y el modelaje de dichos objetos. Se ha desarrollado todo apoyándonos en **[5] [6] [7] [8].**

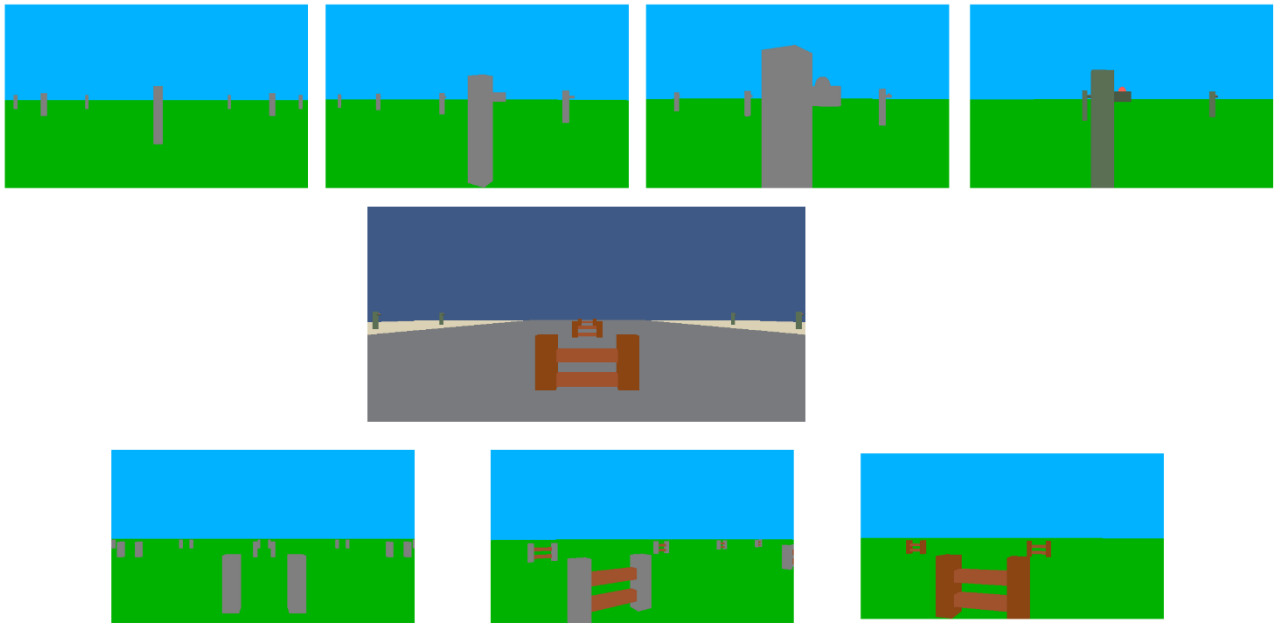

Figura 4 - Creación de los modelos

Nota: Los objetos fueron creados previamente al mapa ya que queríamos observar sus dimensiones con el movimiento libre dado por el ejemplo.

4.5 Generar los objetos dentro del mapa

Una vez explicado el desarrollo de los distintos modelos es hora de mostrar cómo ha sido posible su inclusión por el mapa, en una primera instancia simplemente se partía del bucle de renderizado aportado en la función "**renderScene**".

Dentro de este bucle hemos modificado las líneas donde dibujaban los muñecos de nieve en distintos puntos. Primero se han ajustado los valores de los ejes donde se dibujaban adaptándolos solo al carril central y seguidamente se han cambiado los muñecos por las vallas. Tras esto se ha vuelto a reajustar las posiciones empleando números aleatorios, consiguiendo de esta forma que se generen las vallas en los tres carriles a lo largo del recorrido.

El único problema es que esto queda muy poco orgánico, simple y estático es por eso por lo que adaptamos la función de forma que en cada iteración del bucle las posiciones alternen. En la **figura 5** se observa el progreso mencionado (En este punto se generan cactus fuera del mapa para tener una base de la cual partir el próximo punto "Extras"). Para la aleatoriedad nos hemos apoyado en **[9].**

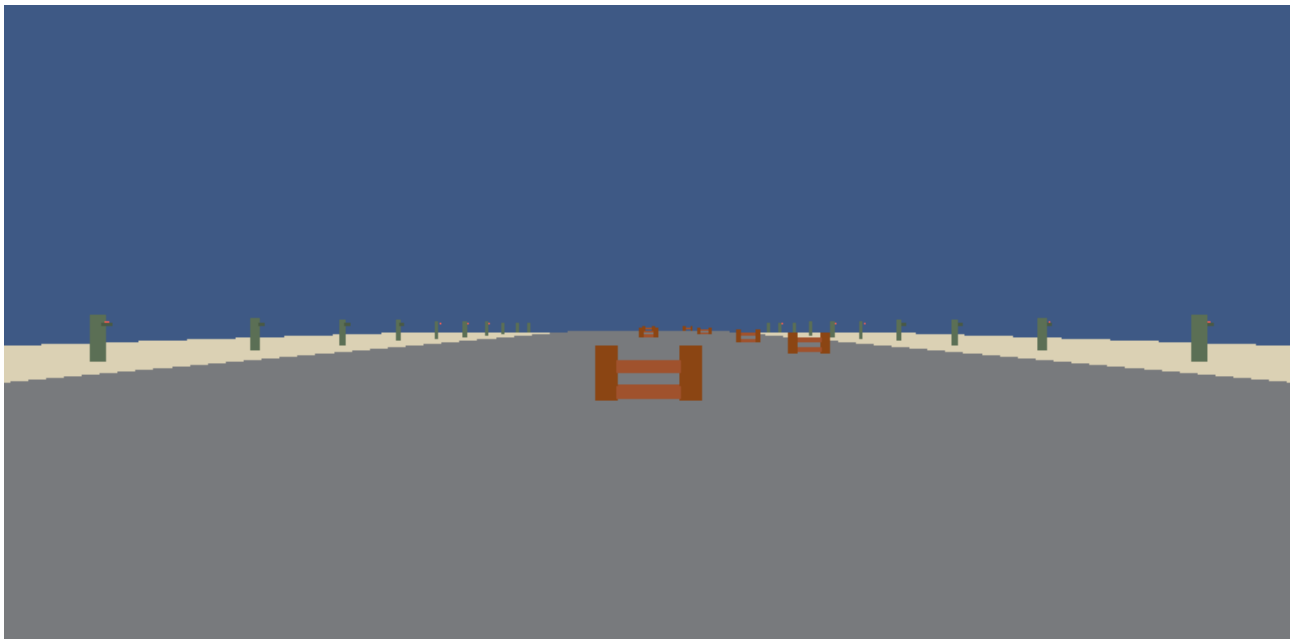

Figura 5 - Generación Aleatoria

4.6 Interfaz del juego y su contador

Para este apartado por desgracia no hemos podido lograr aquello que buscábamos. En un principio queríamos añadir un menú principal donde permitiera al jugador empezar la partida y salir de esta. Todo esto ha sido un fracaso y vamos a contar el por qué junto con los arreglos que hemos podido aplicar.

En primer lugar, la interfaz, según hemos estado observando e investigando, normalmente suelen venir dadas por librerías externas. El problema reside en que dichas librerías o bien están

desactualizadas o bien no presentan los requisitos que buscamos o bien dependen de otras librerías de gestión de ventanas las cuales no estamos implementando en nuestro proyecto.

Entonces con todos estos problemas no nos queda otra que dibujarla con primitivas básicas y gestionar las cosas mediante ratón, aunque, no ha sido implementada esta opción ya que la renderización del menú y las capturas realizadas mediante el ratón eran bastante complejas para ser adaptadas correctamente (obteníamos muchos problemas con el cambio de escenario y la captura del ratón no funcionaba como se esperaba).

Es por ello, que, con todos estos problemas se ha optado por pausar/reanudar el juego al pulsar la tecla 'E/e', para que el usuario pueda notar dicho efecto se ha decidido invertir los colores del escenario (Esto es un efecto que suele emplearse en series para denotar que el tiempo se ha parado). En la **figura 6** se muestra el resultado final.

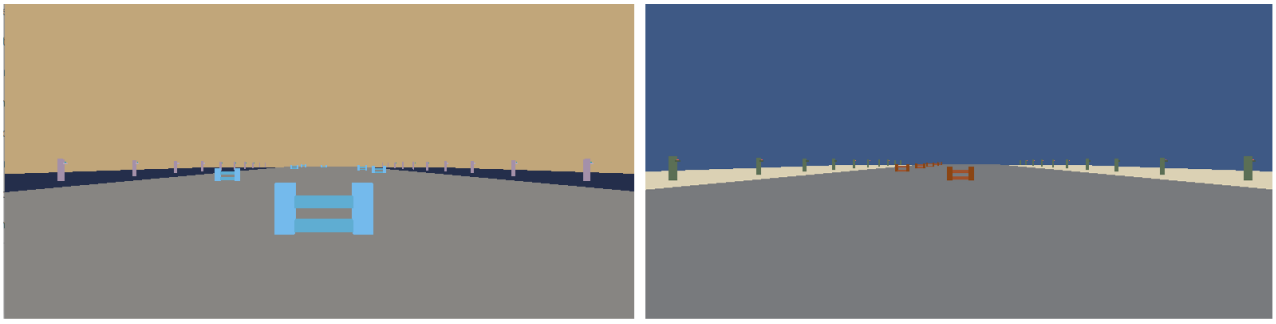

Figura 6 - Método para Pausar

Por otra parte, el contador de distancia. Para este contador se ha tratado de emplear las funciones "**glutBitmapString**" **[10]**, "**glutBitmapCharacter**" **[11]** y "**glutStrokeString**" **[12]** pero ninguna ha dado resultado.

La primera no ha funcionado ya que según vemos en la documentación glut ya no dispone de esta función. La segunda función directamente no mostraba nada por la pantalla y, la última opción daba problemas con los buffers y el renderizado. Finalmente, con la ayuda del ejemplo "***dinospin.c***" extraído de **[13]** hemos podido adaptar su función "**showMessage**" a nuestro código y hemos sido capaces de insertar el contador dentro del juego, en la **figura 7** se adjunta el resultado. Para la conversión del entero al string hemos seguido el tutorial de **[14].**

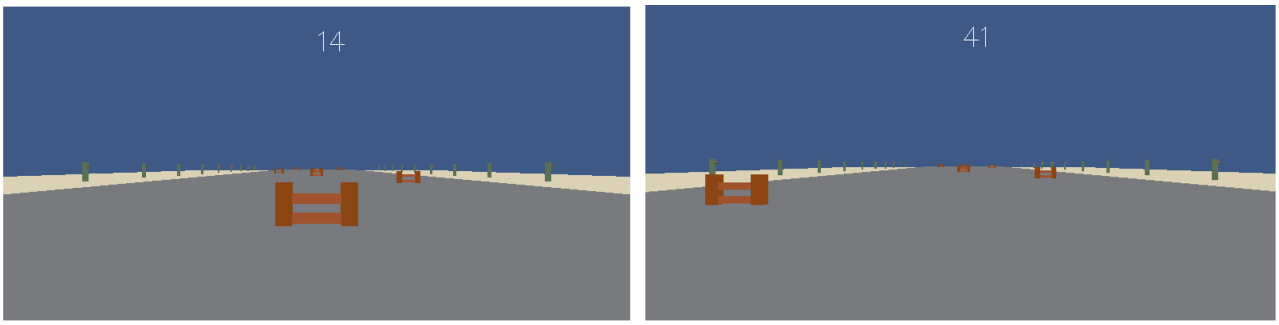

Figura 7 – Contador

4.7 Actualización de Interfaz y controles

En el punto anterior se remarca la falta de un menú, pero tras varios intentos de formar uno propio se han conseguido resultados adecuadas al contenido que deseamos alcanzar. Previamente toda información adicional a la distancia recorrida era mostrada por el terminal, pero esta característica ha sido completamente alterada, ahora se dispone de un menú propio al iniciar la partida, pausar e incluso al perder.

En este caso el menú se encarga de mostrar al jugador los controles disponibles del juego en la sección izquierda de la pantalla mientras que, por otra parte, en la sección derecha se actualizan los datos respectivos a la puntuación del usuario. En una primera instancia tenemos el valor "distancia" que indica al jugar el recorrido realizado en la partida actual. A medida que el jugador va avanzando la dificultad aumenta junto con la posibilidad de perder. Al tratarse de un juego con formato "endless" no hay niveles de dificultad ya que esta aumenta con el tiempo, juegos como templerun y subway surfers corroboran el uso de este tipo de jugabilidad.

Al perder se registra la puntuación de la vuelta anterior como "vuelta x" donde x denota el número de la partida, la interfaz es capaz de almacenar y mantener un registro de hasta 6 vueltas distintas (sin incluir la actual). En la **figura 8** se muestran los dos tipos de menú disponibles del juego tanto cuando se empieza/pausa la partida, así como cuando se pierde.

Figuraa 8 – Interfaz propuesta para el videojuego

4.8 Implementación de las colisiones

La detección de colisiones no ha sido complicada, pero ha llevado al uso de gestionar varias variables. En un primer lugar la generación realizada hasta el momento no estaba registrada en ningún lugar, es decir, los objetos se sitúan por el mapa, pero no dejan ningún rastro sobre que ejes están, es por ello por lo que no se puede detectar las colisiones de momento.

Dado este problema hemos decidido crear una estructura la cual es un array que almacena las posiciones de los objetos y su tipo, donde el tipo de momento solo son vallas y en un futuro también serán los potenciadores. Todo esto se realiza dentro de los bucles que renderizan las distintas vallas, y va actualizándose en cada iteración.

Una vez tenemos las coordenadas de todos los objetos sobre los cuales queremos operar es momento de ajustar dicha detección, en un primer lugar la detección va a trabajar sobre el eje y ya que es sobre el cual estamos avanzando. Detectar la colisión exacta tanto del objeto como

de la nuestra puede dar lugar a que nunca se detecte a pesar de que atravesemos el objeto ya que estamos trabajando con números flotantes.

Es por ello, que para saber si hemos colisionado vamos a tener que comprobar que nuestra posición está dentro de un rango determinado partiendo desde la posición del objeto, además, debemos tener en cuenta el tipo de objeto sobre el cual hemos colisionado y si estamos sobre el carril que corresponde.

Si todo esto se cumple se desactiva el escudo la primera vez (esto se implementa más adelante) y si no tenemos escudo vamos perdiendo vidas hasta que no nos queden llegando así al punto donde se reinicia el juego (tanto las vidas, como el escudo, como la distancia y la velocidad). Un objeto de orientación ha sido **[15].**

4.9 Movimiento con el uso del ratón

Como ya se ha comentado en el apartado 4.3, se dispone de un movimiento básico en el que el personaje puede moverse entre los carriles, pero tenemos el problema de la actualización, así que, por defecto, una vez se desplaza hacia uno de los lados, después vuelve al centro. La idea del uso del ratón es que el jugador se mueva hacia los lados con las funciones **glutMotionFunc()[16],** que se activa al hacer click con uno de los botones del ratón, y **glutPassiveMotionFunc()** que se activa simplemente con el movimiento del ratón.

Sin embargo, para adaptar el movimiento del juego al del ratón, dado que se parte de una base en la que se vuelve de forma automática al soltar una tecla, es bastante complicado determinar en qué momento el usuario deja de mover el ratón hacia los lados, para en ese momento devolverle al carril central.

Para no crear dos tipos de movimiento y que la sensación sea la misma jugando con el teclado que con el ratón, se debía activar algún mecanismo para que volviese al centro si no se detectaba movimiento del ratón. El problema, es que estas funciones sólo se activan si se detecta movimiento, y si intentamos hacer cualquier cambio en el método que actualiza el frame, la actualización es demasiado rápida. Por lo tanto, nos encontramos con el mismo problema, pero cuantificado dado que debemos tener en cuenta que cada jugador puede jugar con un ratón diferente, con una sensibilidad diferente y con un sensor diferente, que recoja el movimiento con distintos DPI. Siendo un problema y un poco difícil que funcionase con todas las configuraciones de ratón, fue la opción implementada hasta casi el final de la entrega hasta encontrar un modo más eficiente de determinar el carril al que debe ir la cámara.

Dado que mediante el movimiento del ratón era difícil actualizar la posición, la siguiente idea era determinar que si el ratón estaba en la parte derecha de la ventana, se mantuviese en el carril derecho, y exactamente igual con el lado izquierdo. Una vez se vuelva a llevar el ratón al centro, que la cámara también se moviera al centro. Para esto era necesario dividir la ventana en tres regiones de interés, y determinar en cuál estaba el ratón para actuar consecuentemente. Como con las funciones mencionadas obtenemos las coordenadas del ratón, nos basta con saber la posición X del ratón dentro del plano en el momento con glutPassiveMotionFunc. Con GLUT_WINDOW_WIDTH conseguimos el ancho de la pantalla, y podemos dividirla en las regiones de interés para que determine en qué lugar debe estar la pantalla. La primera vez que se probó con la variable GLUT_SCREEN_WIDTH**[17]** y produjo problemas, después de revisar la librería de

OpenGL se encontró la variable necesaria.

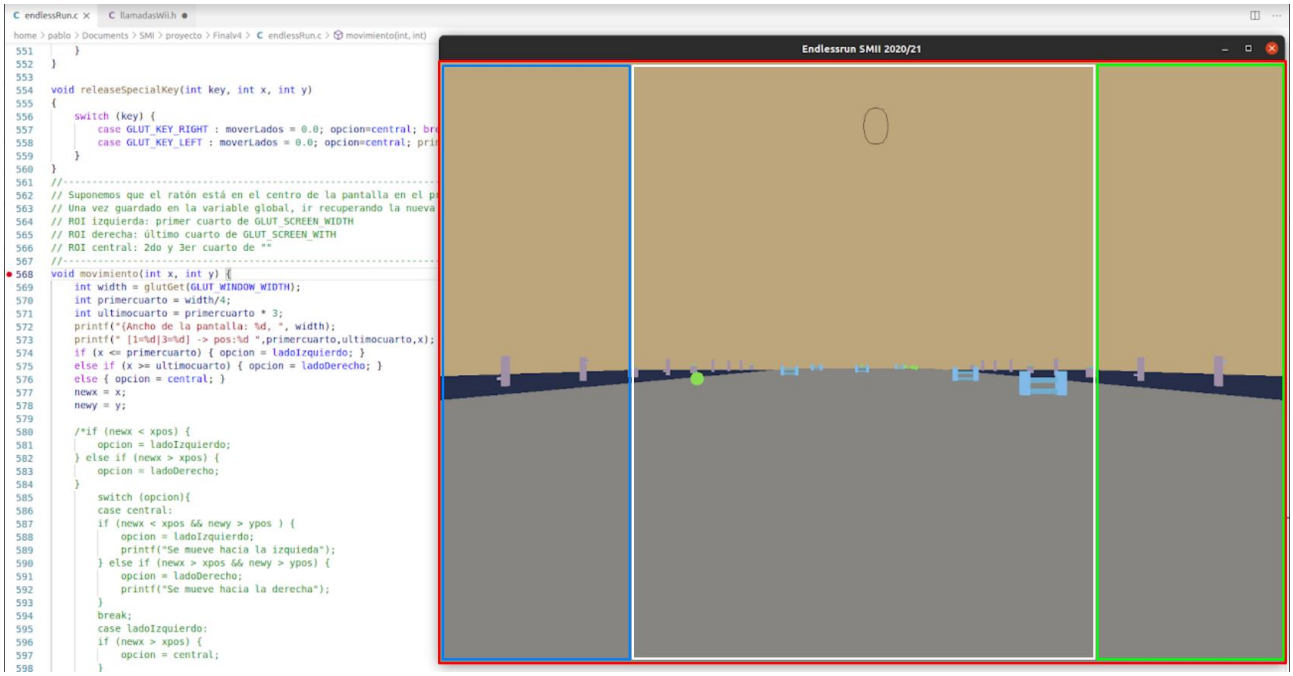

Figura 9 – Implementación del control mediante el ratón usando ROIs

Como podemos ver en la **figura 9**, el programa crea unas regiones de interés dentro de la ventana, rodeada con el marco rojo. Dentro tenemos, dibujado por encima, un marco azul y uno verde del mismo tamaño, de ¼ del ancho de la pantalla, y el resto en blanco. Si el ratón se encuentra dentro del rectángulo azul, la cámara cambia al carril izquierdo, y si se devuelve el ratón a la zona de dentro del rectángulo blanco, vuelve al centro. Ocurre igual con la parte verde, de forma que se colocaría en el carril derecho.

4.10 Librerías en C/C++ para el uso de Wii Mote

Dado que la idea principal era realizar un control con el mando de Wii Mote, **se hizo la revisión de diversos trabajos de años anteriores para documentarnos sobre cómo poder interactuar con el Wii Mote**. En concreto "Uso del Wii Mote en el computador", de Samuel Vinader, que a su vez hacía una revisión del programa wiZarra; y de "Integración Wii Mote en una aplicación OpenGL", de Ignacio Richart Ribes y Jorge Revert Enguix.

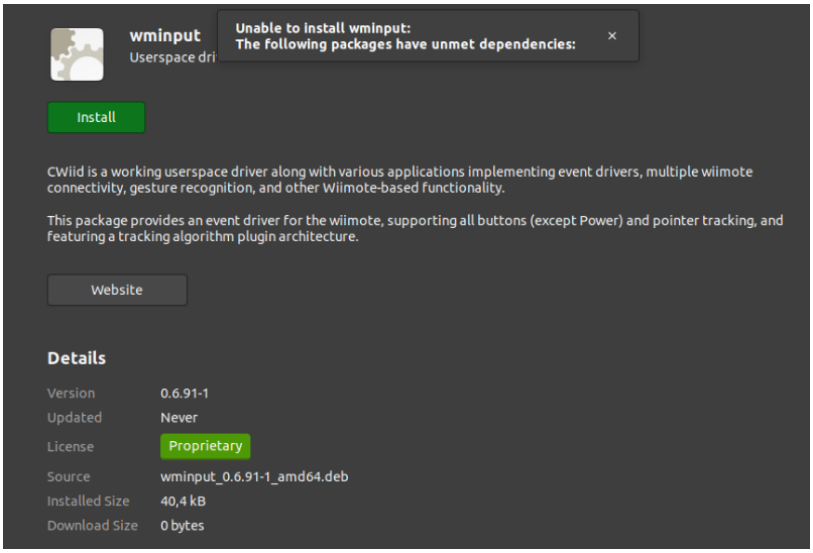

Figura 10 – Error al instalar el paquete "wminput"

Desgraciadamente, ni el trabajo de Samuel ni el del equipo de Ignacio y Jorge se pudieron compilar y ejecutar al 100%. Los dos trabajos tienen en común el uso de "wminput"**[18]**, un paquete que ya no está disponible en los repositorios de Ubuntu (distribución usada por los 2 integrantes del grupo para realizar el proyecto), y al intentar bajar directamente el paquete .deb de internet no es posible instalarse por dependencias de terceros que no puede resolver ningún gestor de paquetes. Sin embargo, teniendo la librería de cwiid**[19],** base de wminput y wmgui podíamos investigar por otros lados. A la hora de recompilar los ejemplos disponibles se acaba con algún error que no permite avanzar. Por ejemplo, haciendo la revisión de WiZarra nos encontramos con problemas con el gestor de ventanas GTK dentro del proyecto:

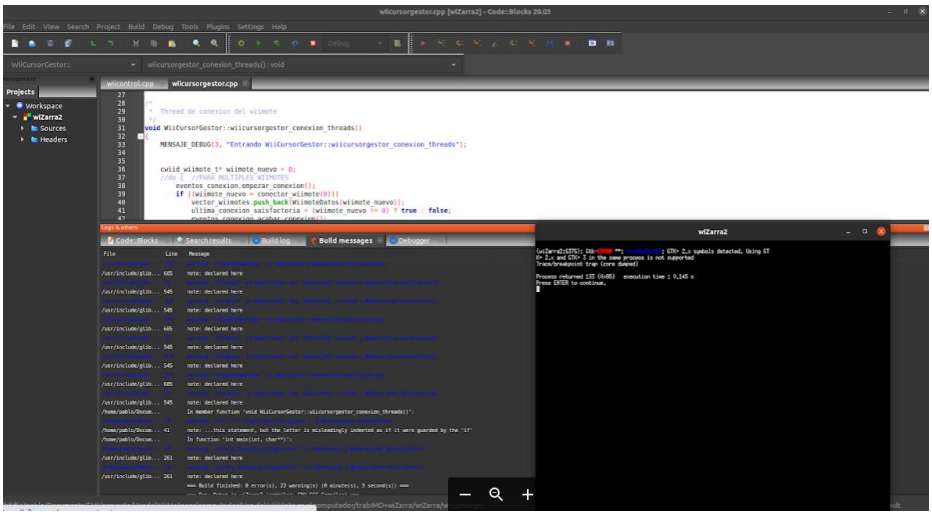

Figura 11 – Error en la compilación de wiZarra

Los problemas con las interfaces gráficas no nos importaban tanto, ya que estamos buscando cómo conectar el mando de la wii para recoger los datos, así que el trabajo de la wiZarra simplemente se dejó apartado, ya que el código encontrado en ella tenía las mismas bases que los demás proyectos, y era más productivo buscar en los otros proyectos que contaban con una documentación mejor.

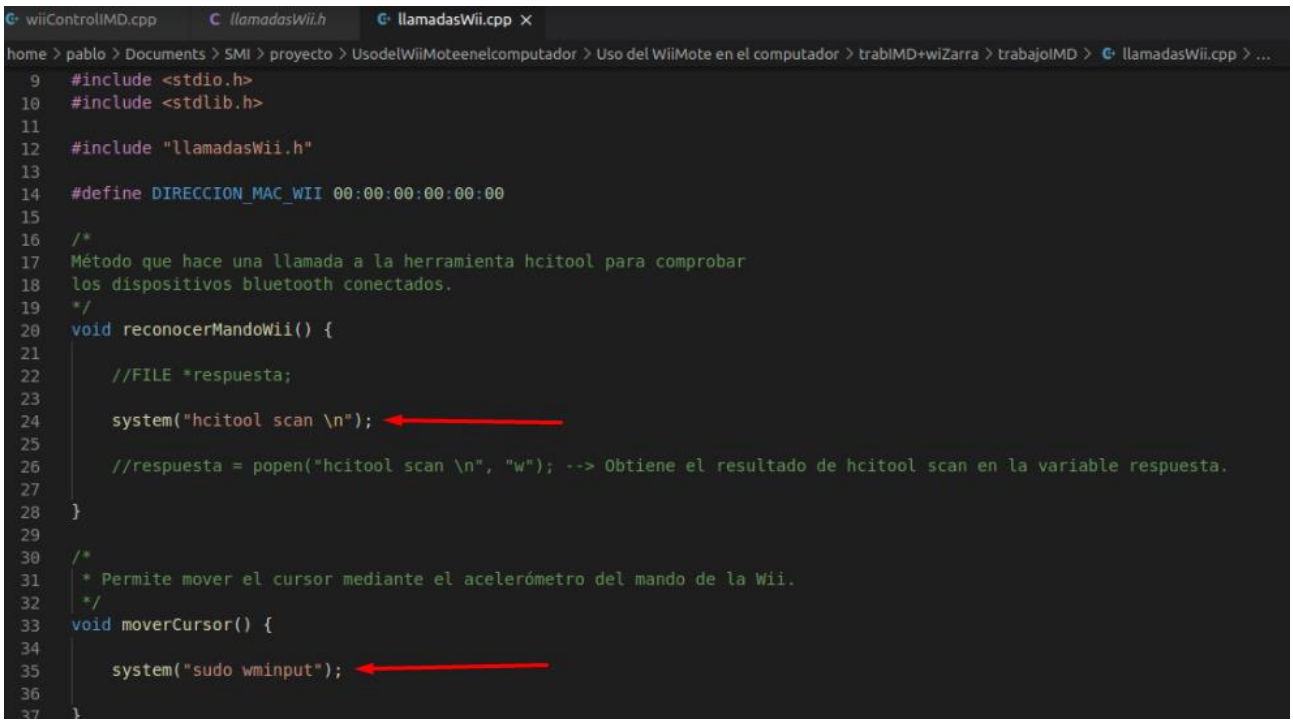

Figura 12 – Código de llamadasWii.cpp

En la **figura 12** vemos el caso de wminput, programa utilizado por los trabajos de alumnos de años anteriores. Buscando en internet se puede encontrar el paquete .deb que debería contener wminput y si se instala podríamos avanzar. Sin embargo, nos encontramos con dependencias sin arreglar, pero no tenemos información sobre estas dependencias.

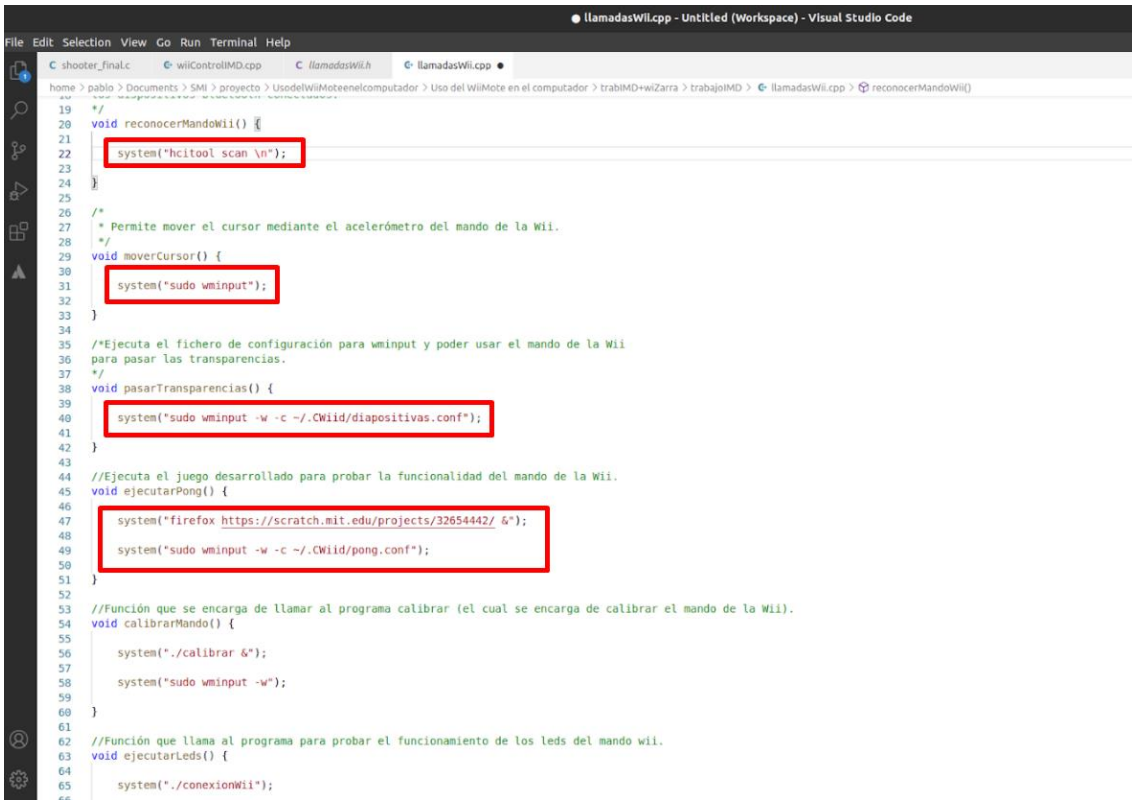

Figura 13 – Código de llamadasWii.cpp

Revisando el trabajo “wiiControlIMD” vemos que delega su uso en wmipunt, el cual ya hemos comentado que no podemos acceder a él. Lo único que podemos hacer es descubrir los dispositivos bluetooth con hcitool y conectarnos para encender los botones del WiiMote, un ejemplo que está sacado del propio paquete de github de cwiid.

Figura 14 – Conexión del mando Wii Mote

Un caso que sí se ha podido resolver es este de hcitool, que sigue disponible y sigue usándose, por lo que es posible conectarse al mando, pero al no tener ninguna librería que nos ayude a interactuar con el mando no nos sirve para nuestro propósito. Ya que la biblioteca de cwiid está disponible y se puede modificar, se intentó vincular directamente modificando el código, pero sin resultado.

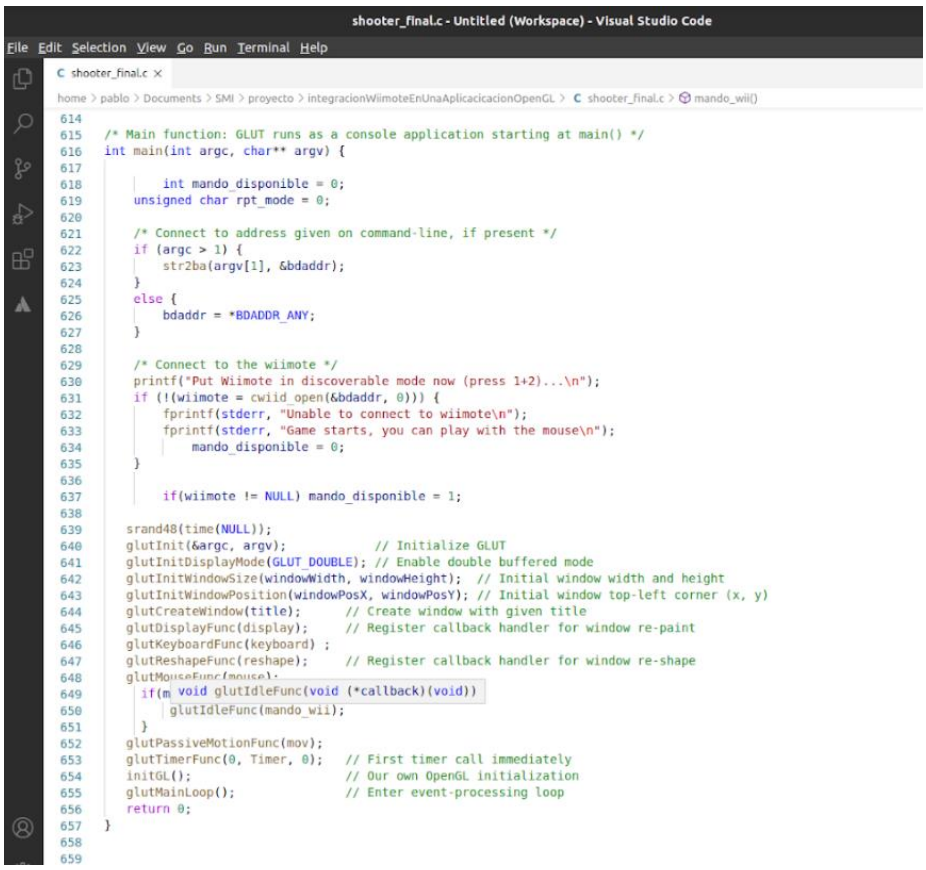

Figura 15 – Ejemplo de uso de la conexión al WiiMote con cwiid (no funciona)

Pasando al trabajo del otro grupo, también se dedicaron recursos para entender su funcionamiento y conocer si podíamos extraer alguna idea o código directamente para poder utilizarlo en nuestro proyecto. Nos encontramos con que utiliza cwiid directamente para conectarse, pero nos encontramos con un proyecto inestable que nos devuelve una segmentación del núcleo en tiempo de ejecución una vez recompilado.

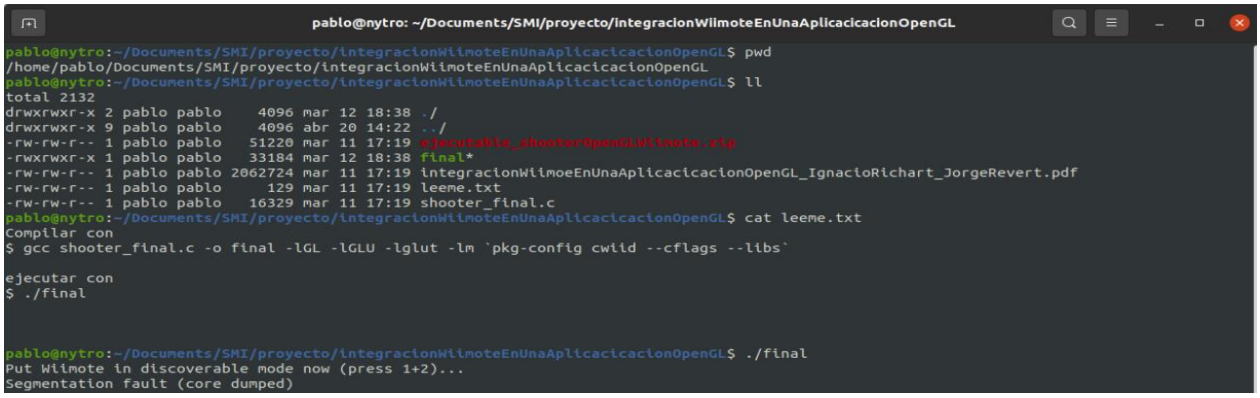

Figura 16 – Ejecución del código mostrado en la figura 15

Como podemos ver en la imagen de arriba, se muestra el error mencionado, pero en la figura anterior a ésta vemos cómo podemos darle directamente la dirección del dispositivo bluetooth para que intente conectarse. Esto resulta con un error del socket. Pensábamos que como la creación de sockets requiere permisos especiales, podría ser tema de privilegios de usuario. Sin

embargo, ejecutando la aplicación con "sudo" delante, tampoco podemos ejecutarla con una conexión exitosa.

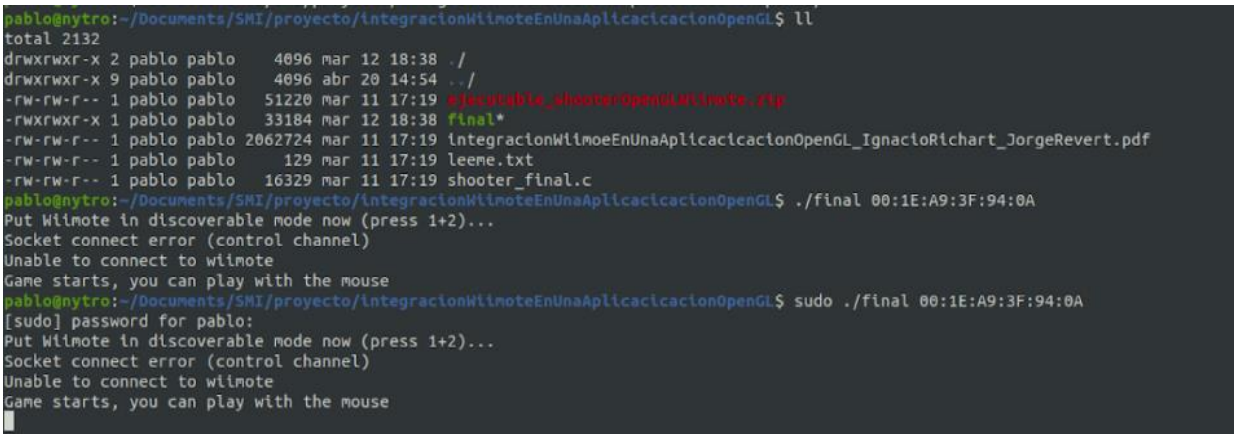

Figura 17 – Error en tiempo de ejecución que persiste incluso ejecutándolo con sudo

Revisando los trabajos de años anteriores podemos concluir con se conectan mediante hcitool o lo utilizan para descubrir el dispositivo, pero luego delegan el uso de su aplicación directamente en wminput, así que el código o bases sólidas con las que esperábamos comenzar a experimentar con el mando Wii Mote no nos sirven por haber utilizado directamente llamadas al sistema delegando el uso a wminput, no conseguimos recuperar nada de éstos, y no nos aportan nada con lo que trabajar.

En general, este tipo de proyectos y librerías tan especializadas sin un soporte demasiado continuado acaban quedando desactualizadas, y la librería utilizada por ambos proyectos era cwiid, así que nos centramos en esta. Independientemente de la librería escogida, todas tienen bastantes años de antigüedad, de forma que es costoso recuperarlas para que trabajen en entornos de diez años después, sobre todo si hay librerías y paquetes de las que depende que han sido modificadas y ya no funcionan con nuevas, o simplemente no están disponibles ya.

Además de las complicaciones encontradas para intentar compilar los ejemplos de años anteriores, para poder sacar en claro qué tecnologías o librerías podíamos utilizar debido a que las librerías en sí ya no se pueden utilizar correctamente, luego se habría tenido que dedicar mucho tiempo a investigar el funcionamiento de la librería y a la integración en nuestro programa, donde. Una vez se consiguiese vincular un dispositivo, luego debería implementarse el reconocimiento de los botones y los sensores (osciloscopio, giroscopio) para que nuestro programa pudiese reconocerlos e interactuar con la librería de cwiid, por lo que estimamos que el trabajo que conllevaría conseguir esto en relación con la envergadura del tamaño sería demasiado, ya que durante semanas se han encontrado problemas, uno tras otro, para intentar poner en funcionamiento una librería desactualizada. La idea de utilizar el mando de Wii Mote viene fuertemente motivada desde las primeras clases de teoría, viendo todas las posibilidades de interacción con distintos dispositivos, pero debido a las complicaciones encontradas se ha decidido descartar esta parte y dedicarle más recursos a mejorar el programa en general.

Por lo tanto, esta parte del proyecto, debido a tantas complicaciones que se han ido encadenando a la hora del trabajo, se ha decidido descartar porque peligraba la integridad de este, de forma que podría quitarnos más tiempo del previsto. Como la planificación y el ritmo de trabajo fue bueno, fue posible **añadir nuevas funcionalidades** que suplen la falta del uso del Wii

Mote y puedan mejorar las características del programa **mientras se buscaban alternativas al Wii Mote**. En concreto, se añadió un segundo modo control a parte del uso del teclado, que no estaba previsto en un principio en nuestros objetivos: el uso del ratón mediante regiones de interés explicado en el apartado 4.8; y se añadieron efectos de sonido y una banda sonora, así como el uso de power-ups, los escudos.

4.11 Instalación DS4DRV, Uso e implementación de un Dualshock 4.

Una de las fortalezas y bellezas de nuestro proyecto era la inclusión y uso del mando WiiMote para dar un toque de interactividad extra. Dadas las circunstancias mostradas en el anterior punto dicha idea ha sido descartada y suplida por el uso del ratón tal y como se ha mencionado, pero con el paso de los días no nos quedamos muy satisfechos con el resultado es por eso por lo que hemos decidió incluir o, mejor dicho, hacer uso del mando Dualshock 4. En este caso hemos hecho uso de este dispositivo mediante cableado ya que su uso por Bluetooth no ha podido ser posible, aunque este apartado se detallará más adelante.

Para hacer uso de un artilugio Dualshock 4 es necesario instalar el controlador DS4DRV junto con sus dependencias, dentro de su página oficial **[20]** hay una guía que explica cómo se instala, pero aun así nosotros vamos a explicar nuestro proceso. En un primer lugar, hay que cubrir las dependencias más básicas instalando los paquetes/librerías: **Python (2.7 o 3.3+), Python-setuptools y hcitool.**

Nosotros hemos partido de un sistema Ubuntu 20.04 por tanto no hemos tenido que preocuparnos por instalar Python3 ya que con dicha versión del SO ya viene esta característica instalada, pero sí debemos afirmar que hemos hecho uso de "**pip**" el cual es un paquete que ha requerido ser instalado con la orden "**sudo apt install -y python3-pip**". Para comprobar que tanto python3 como pip están instalados correctamente bastará con ejecutar lar órdenes "**python3 --version**" y "**pip3 --help**". Si todo está correctamente configurado e instalado tendremos un resultado similar al de las **figuras 18 y 19**.

```
david@daclefe1:~/Desktop/SMI/trabajo/EndlessRun$ python3 --version
Python 3.8.5
```

Figura 18 – Comprobación de la versión de Python instalada

```
david@daclefe1:~/Desktop/SMI/trabajo/EndlessRun$ pip3 --help

Usage:
  pip3 <command> [options]

Commands:
  install                     Install packages.
  download                    Download packages.
  uninstall                   Uninstall packages.
  freeze                      Output installed packages in requirements format.
```

Figura 19 – Comandos disponibles con pip3

La librería "**setuptools**" en el momento que hemos realizado la instalación no tenía su página web **[21]** disponible, pero hemos podido instalar el paquete desde la página **[22]** tal y como se indica, empleando el comando "**pip3 install setuptools**".

Por último, el paquete "**hcitool**" como bien dice la página, suele venir incluido al instalarse añadidos como BlueZ o similares, aunque como algunos miembros del equipo no disponíamos de estos paquetes hemos procedido a la instalación de BlueZ con la ayuda de **[23]** empleando los comandos "**sudo apt-get purge bluez***" y "**sudo apt-get install bluez-utils**". Queremos remarcar que nosotros no sufrimos ningún percance como el mostrado en la discusión e insistimos en la gran importancia de las dependencias de librerías.

Cabe destacar que el propio foro remarca que hay otras dos librerías llamadas "**pyudev**" y "**Python-evdev**" que serán necesarias para el uso de este driver pero que no nos debemos preocupar ya que se instalarán al ejecutar el script de DS4DRV pero como bien se dice, más vale prevenir que curar, es por eso que con la ayuda de **[24]** y **[25]** hemos podido instalar estas librerías antes de ejecutar el script con los comandos "**pip3 install pyudev**" y "**sudo pip3 install evdev**" tal como se muestra en la **figura 20** .

```
david@daclefe1:~/Desktop/SMI/trabajo/EndlessRun$ pip3 install pyudev
Requirement already satisfied: pyudev in /usr/local/lib/python3.8/dist-packages
(0.22.0)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from pyude
v) (1.14.0)
david@daclefe1:~/Desktop/SMI/trabajo/EndlessRun$ sudo pip3 install evdev
Requirement already satisfied: evdev in /usr/local/lib/python3.8/dist-packages (
1.4.0)
```

Figura 20 – Instalación de pyudev

Una vez se han suplido todas las dependencias es momento de instalar el controlador, para ello nos guiamos de las instrucciones del propio foro ejecutando el comando "sudo pip install ds4drv". Si la instalación ha sido un éxito se tiene que obtener algo similar a la **figura 21**.

```
david@daclefe1:~/Desktop/SMI/trabajo/EndlessRun$ sudo pip3 install ds4drv
Requirement already satisfied: ds4drv in /usr/local/lib/python3.8/dist-packages
(0.5.1)
Requirement already satisfied: pyudev>=0.16 in /usr/local/lib/python3.8/dist-pac
kages (from ds4drv) (0.22.0)
Requirement already satisfied: evdev>=0.3.0 in /usr/local/lib/python3.8/dist-pac
kages (from ds4drv) (1.4.0)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from pyude
v>=0.16->ds4drv) (1.14.0)
```

Figura 21 – Instalación de ds4drv

Podemos comprobar que dichos paquetes han sido instalados ejecutando la orden "**python3 -m pip list**". Ahora solo falta localizar los paquetes correspondientes tal y como se muestra en las **figuras 22 y 23**.

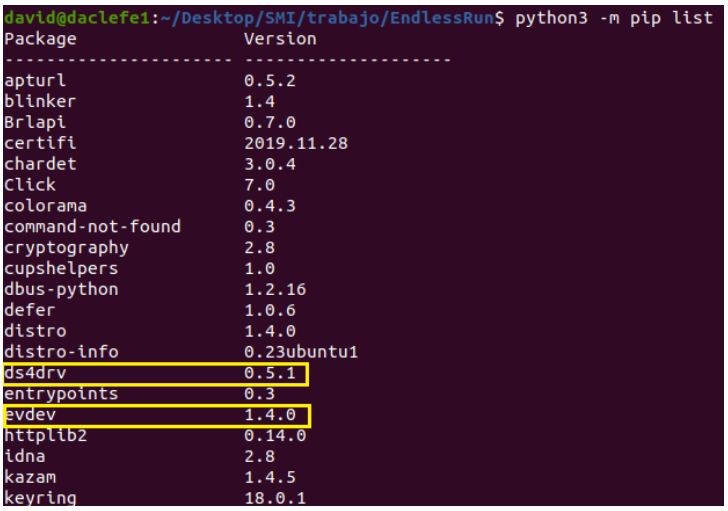

Figura 22 – Versión de ds4drv y evdev

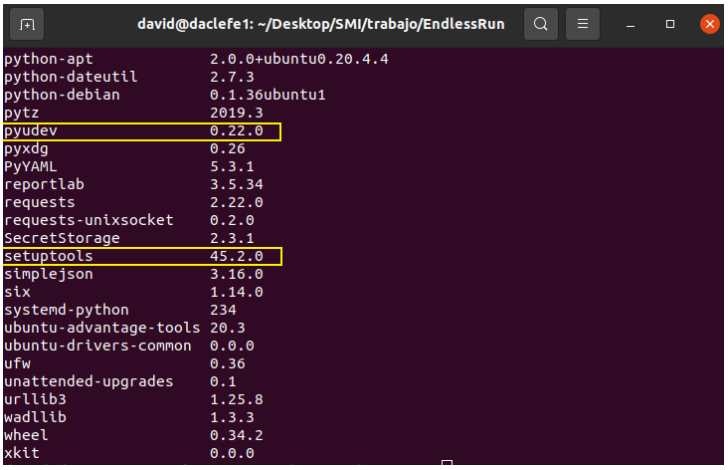

Figura 23 – Versión de pyudev y setuptools

Llegados a este punto se ha completado con éxito la instalación a partir de este punto se va a explicar el funcionamiento que le hemos dado. Dicho controlador permite hacer uso tanto del cable como del propio bluetooth, pero nos hemos encontrado con un problema que nos impide hacer uso del bluetooth. Este problema viene dado por los componentes del propio ordenador ya que no poseen de este mecanismo, al intentar usar el comando "**ds4drv**" para ejecutar el controlador recibimos un error relacionado con las librerías del bluetooth donde nos indica que hemos de inicializar el dispositivo del propio ordenador mediante la orden "**hciconfig hciX up**" pero al emplear dicha orden como se puede observar en la **figura 24**, no se encuentra ningún componente de bluetooth integrado.

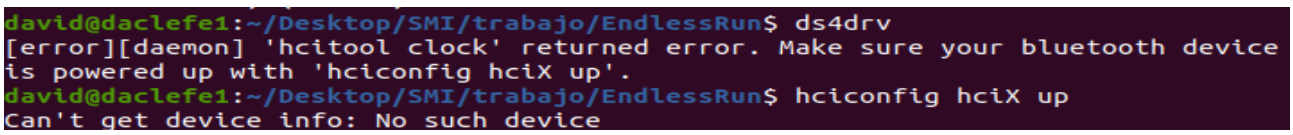

Figura 24 – Error en el dispositivo bluetooth

Aunque no se haya podido conectar por bluetooth si se intenta emplear el comando "**ds4drv --hidraw**" se puede observar en la **figura 25** como hay un error, no se puede crear un fichero para almacenar los valores del mando y así emparejarlo.

```
david@daclefe1:~/Desktop/SMI/trabajo/EndlessRun$ ds4drv --hidraw
[error][controller 1] Failed to create input device: "/dev/uinput" cannot be ope
ned for writing
```

Figura 25 – Error de permisos en la escritura por ds4drv

Si seguimos leyendo el perfil oficial del controlador se puede observar que este problema suele ser común cuando no hay permisos suficientes, para corregir esto basta con descargar el fichero que se nos proporciona en **[26]** y tras esto bastará con copiarlo en **"/etc/udev/rules.d/"** tal y como se ve en la **figura 26**. Importante ejecutar los comandos **"sudo udevadm control --reload-rules"** y "**sudo udevadm trigger**" tras estos pasos.

```
david@daclefe1:~$ sudo cp 50-ds4drv.rules /etc/udev/rules.d/
[sudo] password for david:
david@daclefe1:~$
```

Figura 26 – Solución de los permisos de escritura

Con todos estos pasos establecidos volvemos a probar el comando "**ds4drv --hidraw**" y esta vez, a diferencia de antes se reconoce el mando y el propio programa sigue escuchando por si se quieren añadir más mandos. Este comportamiento se puede observar en las **figuras 27 y 28**.

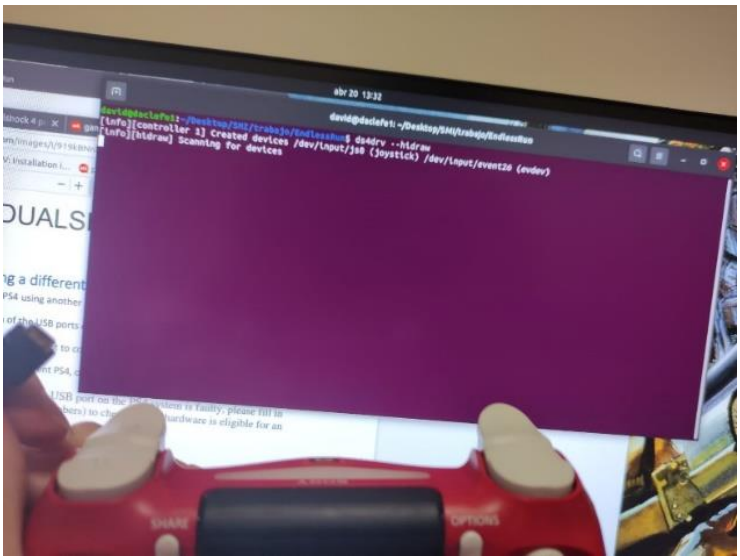

Figura 27 – Búsqueda de dispositivos por ds4drv

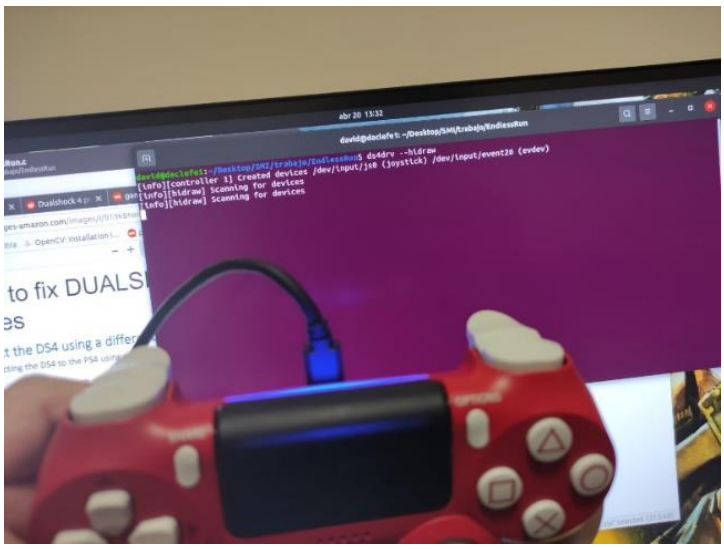

Figura 28 – Búsqueda de dispositivos por ds4drv pese a estar el mando conectado

El emparejado entre el Dualshock 4 y el propio PC ha sido un éxito rotundo, ahora solo falta probar su funcionamiento. Dentro de la carpeta "doc/src" se ha adjuntado un video titulado "pruebaMandoPS4tactil.mp4" donde se muestra el funcionamiento de la pantalla táctil sobre el proyecto sin tener que haber configurado nada, esto se debe a que predeterminadamente la pantalla táctil del dualshock está mapeada como si fuera un ratón y al tener nosotros implementado dicho dispositivo sobre el juego es posible satisfacer esta interacción.

Por otra parte, no sabemos el porqué, pero nos ha sido imposible emplear los controles. Hemos creado un fichero de configuración tal y como se nos indica en el foro oficial del controlador, tan solo hemos copiado el fichero situado en **[27]** y lo hemos copiado sobre un fichero creado con gedit sobre **"~/.config/ds4drv.conf**". De esta forma se observa en la **figura 29**.

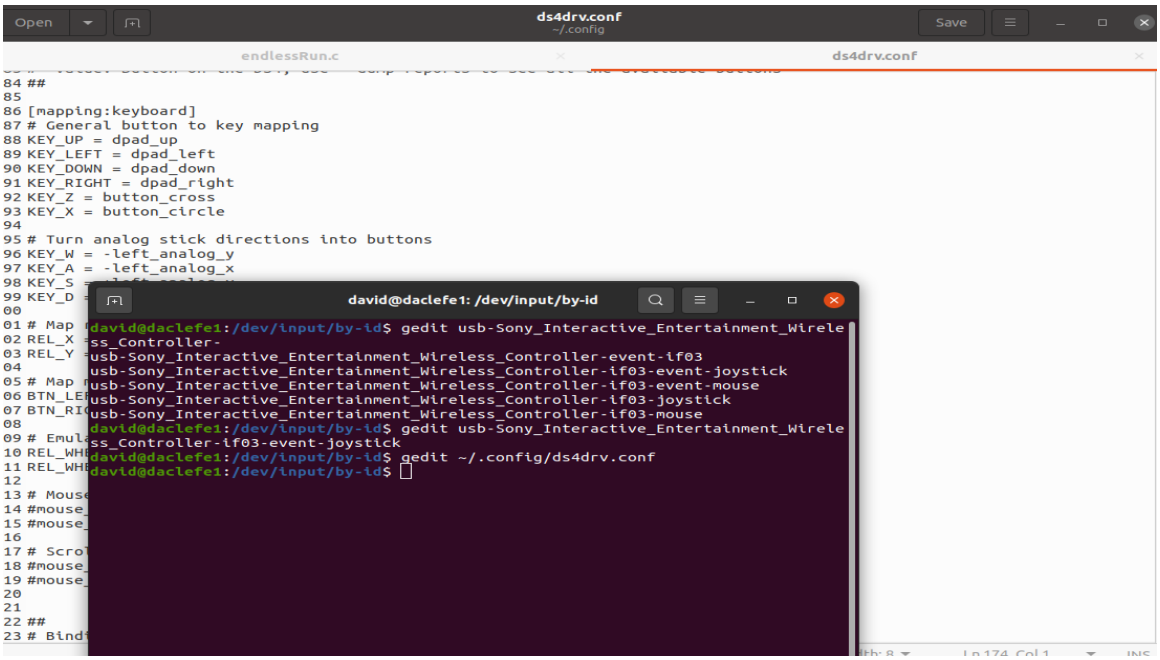

Figura 29 - Mapeado de configuración de ds4drv para Dualshock 4

A pesar de haber asignado las teclas, tal y como se ve en la figura anterior, el mapeado no ha funcionado correctamente o no se detecta, nosotros creemos que esto se debe a la falta de emparejamiento del bluetooth ya que si se intenta usar algún comando distinto a "**ds4drv --hidraw**" salta una excepción sobre el bluetooth idéntica a la de la **figura 25**.

Es posible que el mando WiiMote tuviera un encanto mayor debido al giroscopio y sus otras funcionalidades pero creemos que la adaptación del dualshock 4 es un detalle innovador que añade interactividad entre el entrono real y el virtual al involucrar al usuario y a un dispositivo ajeno al ordenador, creemos que la adaptación de los botones hubiera sido un plus pero dadas las limitaciones de hardware a las cuales nos enfrentamos tenemos una respuesta bastante positiva al poder usar el teclado táctil del propio mando.

4.12 Extras

Dentro del apartado extras tenemos varias ideas que presentar. En primer lugar, se introduce el concepto endless dentro del juego. Como no queremos que el juego utilice una gran cantidad de cómputo para estar siempre generando un mapa totalmente nuevo lo que hemos hecho es que al llegar a x punto sobre el eje y, se reinicie esta variable a su valor por defecto, es decir, una vez llegamos a la distancia 120 (por ejemplo ya que puede ser modificable) se restablecerá la posición del jugador al inicio (un valor de 0) aunque se incrementará la velocidad para dar dinamismo.

Al estar constantemente cambiando la generación de vallas se consigue el efecto endless, ya que nos estamos moviendo en un mapa fijo, pero al reiniciar la posición con nuevas vallas parece un escenario totalmente nuevo además, esto añade un factor aleatorio donde puede ser beneficioso para el jugador o completamente catastrófico. Esta premisa nos parece bastante interesante.

Hay otra idea contemplada para este proyecto, y es la propia inserción de objetos o "power ups". No queríamos que el juego fuera muy complico ni que tuviera muchas mecánicas avanzadas, por esta razón simplemente hemos añadido esferas por el mapa las cuales nos apartaran un escudo para evitar el daño sufrido, que en la **figura 30** se pueden observar dichos objetos.

Otro factor que considerar era añadir más formas de moverse como saltar o agacharse, pero viendo la forma en la que teníamos implementadas ya las bases no eran muy factibles. Es por eso por lo que decidimos añadir sonido en vez de nuevo movimiento. De esta forma se descarta añadir más funcionalidades de OpenGL para dar paso a otra herramienta muy versátil llamada OpenAL.

Durante el transcurso del juego sonará la canción "80's Synthwave by MOKKA [No Copyright Music] / Synthetic Pleasures" extraída de **[28],** y luego dependiendo las interacciones que tengamos sonarán otros sonidos extraídos de **[29],** estos sonidos suelen aparecer al recibir un golpe, al coger un escudo, al perder la partida y al pausar el juego. Como nosotros hemos cambio los nombres de los efectos para adaptarlos al código se adjuntan los nombres originales por si a alguien le interesan*: falling on undergrowth, vibration hit transition, water bubble,Trumpets and strings off beat*. Queremos remarcar que toda operación de sonido ha sido realizada siguiendo **[30] [31] [32].**

Aunque para facilitar el entendimiento de este último bloque mencionado vamos a dejar unos pequeños recordatorios sobre que sonidos existen y que los provocan. En primer lugar, como sonido base en formato estéreo tenemos el fichero "sounds/fondo.wav" que hace referencia a **[33]** donde nos acompañará durante toda la aventura, el sonido ha sido apaciguado ya que mantener el mismo volumen que los otros no era factible ya que sobrepasaba los efectos de estos y resultan inaudibles para el usuario. Por otra parte, al empezar/pausar la partida el jugador escuchará el sonido "sounds/pausa.wav" acompañando como experiencia los sonidos "sounds/golpe.wav" al ser golpeado por una valla y "sounds/powerup.wav" al recoger un potenciador de escudo. Por último, el sonido "sounds/muerte.wav" será reproducido al perder la partida.

Con el paso de tiempo nos hemos dado cuenta de que a pesar de tener unos cuantos sonidos incluidos aún no tenemos suficiente inmersividad ya que hay ciertas características que le quitan organismo al proyecto como por ejemplo, cuando se recibe un golpe y se tiene el escudo activado al perderlo no se escucha ningún tipo de sonido además, al pasar unos cuantos metros y llegar al límite para aumentar la velocidad no se da ningún tipo de incentivo. Dados estos problemas hemos decidió añadir otros dos nuevos sonidos los cuales cubran estos fallos de inmersión. Se pueden localizar dentro de la carpeta "sounds" del proyecto con los nombres "romperescudo.wav" y "masvel.wav".

Queremos remarcar que hemos añadido un último sonido final llamado "lastchance.wav" sacado del mismo banco de sonidos. Este se reproducirá cuando el usuario alcance su última vida. Hay una cosa que no se ha podido corregir y es dar la sensación de una ambientación 3D, es decir, el juego ha optado por potenciar aún más la experiencia inmersiva pero debido a algunos problemas los cuales desconocemos ha sido difícil aplicar el sonido en 3D. Este último sonido tenía como fin añadir una sensación de peligro al usuario ya que se activa cuando este está en sus últimas, queríamos meter al usuario dentro del juego y que este notará la presencia amenazante de una bestia detrás de él pero no ha podido ser así. Hemos intentado modificar las funciones del sonido para situarlo detrás de la posición del usuario, pero por mucho que se modificarán los valores con números negativos el sonido seguía reproduciéndose como si estuviera en la misma posición x, y, z del usuario. Así que como resultado tenemos un sonido amenazante para el usuario, pero no tan inmersivo como esperábamos.

Para finalizar, la última idea es añadir detalle al mapa. De momento el mapa es capaz de generar una línea uniforme de cactus sobre la arena, pero queremos que dichos cactus sean generados aleatoriamente partiendo de tres modelos diferentes, al modificar las funciones que se encargan de esto añadiendo valores fijos se ha obtenido el resultado mostrado en la **figura 30**.

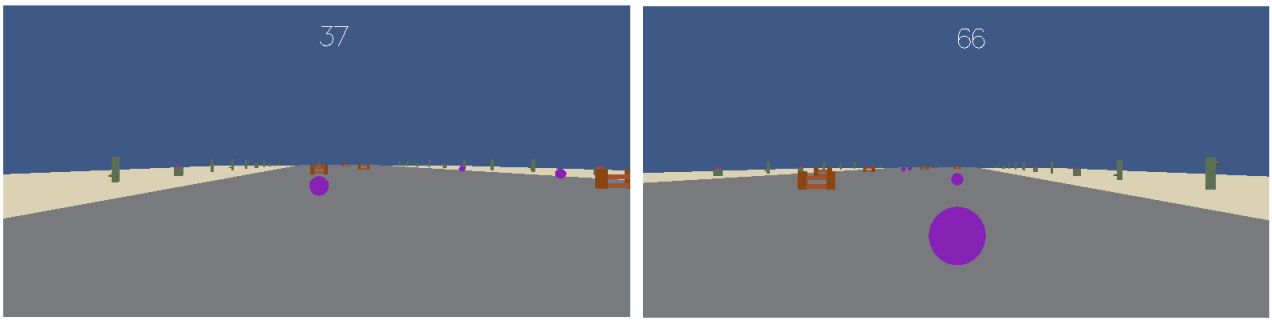

Figura 30 - Escudo & Cactus

Antes de dar paso a la siguiente sección queremos entrar en detalle ante el descarte de las mecánicas tanto de salto como la de agacharse, en un principio la vista que tiene el jugador está configurada dentro del método "**renderScene**", específicamente se ajusta con la función "**gluLookAt**". Para incluir dichas mecánicas es necesario modificar el eje z ya que es sobre el cual se debe trabajar. El problema es que como mínimo hay que incrementar o reducir el valor del eje z en 2 para que pueda observarse un cambio notable, el problema viene dado al trabajar sobre una plantilla ya que esta está configurada de cierta forma para que todos los elementos que se visualicen en ella estén perfectamente posicionados. Si intentamos saltar o agacharnos el juego pierde completamente el sentido orgánico ya que como se puede apreciar en la **figura 31** los efectos visuales de agacharse son devastadores ya que nos sacan del propio mapa y al saltar la vista es muy poco realista y saca al usuario de la inmersión, además en determinadas distancias emplear dicha mecánica puede provocar inconsistencias visuales.

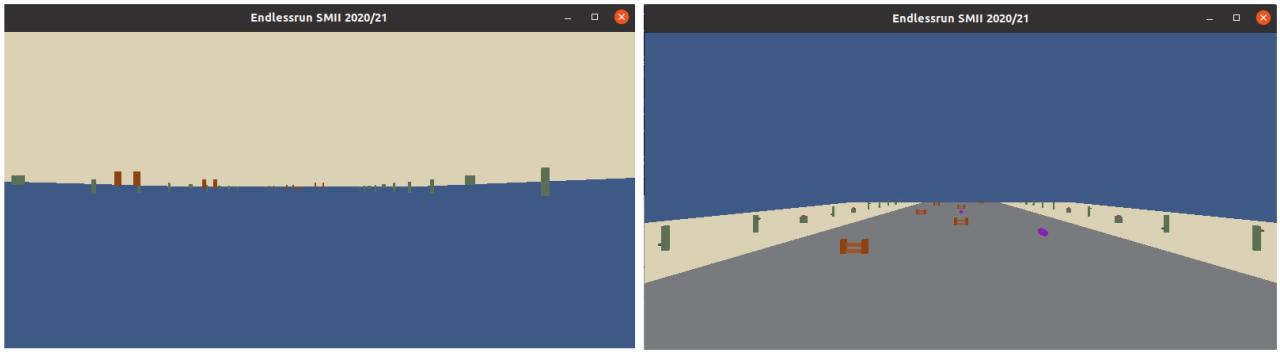

Figura 31 - Visualización mecánica Agacharse/Saltar

5 Trabajos futuros y anotaciones

Para hablar sobre los trabajos futuros, primero es necesario hacer un balance de la ejecución del proyecto paso a paso. Podemos encontrar que extras propuestos como saltar o agacharse no han sido implementados. Este extra iba a ser algo implementado una vez creadas las bases del juego y consiguiendo ya un juego funcional, y tal como teníamos el funcionamiento de la cámara, no era muy factible. Respecto al control de la cámara con el mando de Wii Mote, en el apartado 4.10 se explican las complicaciones y problemas que han imposibilitado el uso del controlador, habiendo llegado a un punto en el que las opciones para seguir intentando implementarlo se estaban agotando.

Dado que estos extras no han sido implementados, se han introducido nuevas funcionalidades que crean un programa más complejo y completo. Respecto al movimiento de la cámara mediante saltar y agacharse, se optó por añadir una biblioteca de sonidos mediante OpenAL, que incluye en el juego una banda sonora y efectos de sonidos para las colisiones y los power-ups, entre otros. Esta decisión nos pareció más acertada que rediseñar el movimiento de la cámara, es decir, deshacer el trabajo hecho ya. Así, hemos implementado la librería OpenAL dentro de nuestro proyecto, dándole una mayor variedad de temas tratados dentro de éste. Como trabajos futuros, una revisión de los posibles movimientos de cámara junto a una implementación de estos saltos podría ser una buena extensión del proyecto, siempre que no modifiquen en exceso la experiencia de juego. Por ejemplo, durante la fase de diseño se dejó como extra el salto porque eso podría permitir al jugador apretar continuamente el botón de salto consiguiendo esquivar prácticamente todos los obstáculos, por lo que debería regularse la cantidad de saltos, o el tiempo en el que se pudiese saltar después de realizar un salto. Esto también podría añadirse como un power-up temporal, de forma que al conseguir este bonus, durante un breve periodo de tiempo tuviese la habilidad de saltar.

En cuanto al control por el mando Wii Mote, se ha cambiado por el uso del ratón para añadir una nueva forma de mover la cámara junto la aplicación del mando dualshock 4. El ratón es un elemento básico que cualquier usuario de un ordenador tiene, por lo que esta funcionalidad puede ser disfrutada por cualquier usuario del videojuego, y no necesita comprar un Wii Mote para probarlo. Aunque nos hubiera gustado que funcionase con este controlador, como trabajos futuros pensamos que puede ser más beneficioso el uso de controladores más modernos y con un mayor uso por la comunidad de videojuegos, como los mandos de PS Move, Xbox One o Dualshock 4 (ajustar el mapeado de los botones) ya que éstos disponen de mayor compatibilidad en los sistemas operativos, existiendo programas que permiten mapear sus botones para realizar ciertas acciones dentro del sistema, como por ejemplo DS4**[33]** e InputMapper**[34].** Ya que se ha conseguido conectar un mando de PS4 por cable para que pueda usarse la entrada táctil, puede que sea un punto de partida para una mejora a este proyecto, de forma que se consigan mapear todos los botones del controlador en sistemas Linux, ya que en MS Windows existen programas que funcionan perfectamente.

Otro extra futuro ideado también podría ser la implementación de un hacha, equipada por el jugador, de forma que se vea sólo la parte de arriba simulando que la sujeta por la parte baja del mango. Esta hacha podría utilizarse para destruir un nuevo tipo de vallas, identificadas por un color distinto, por ejemplo, que diesen más puntos al jugador, por lo que se debería implementar también una animación de destrucción de vallas y del movimiento del hacha cuando está caminando y cuando se utiliza para golpear a la valla, dando mayor sensación de realismo.

Otra funcionalidad propuesta es añadir una tienda en el juego, de manera que a medida que avanzas en tus habilidades y consigues recorrer más metros, desbloqueas en la tienda nuevas

apariencias para los obstáculos. Los obstáculos que se podrían añadir, siempre que tuviesen sentido dentro de la estética del videojuego, son casi infinitos. Esto implicaría crear más figuras mediante OpenGL pero se podrían reutilizar las colisiones.

Por último, también se podría implementar el uso de gafas de realidad virtual, de forma que la cámara en primera persona que se ve en la pantalla del jugador, y si este visor de realidad virtual tuviera sensores de movimiento o de rotación, poder usarlos para que se moviese de carril en base al movimiento del jugador, consiguiendo una experiencia más inmersiva aún.

6 Conclusión

Como conclusión, este proyecto ha resultado muy enriquecedor durante el desarrollo gracias a todos los ámbitos trabajados a lo largo de la realización del Endless Run partiendo de una base sencilla, pero con mucho potencial para desarrollar nuevas cosas. Gracias a la generación de los obstáculos y su colocación en el plano, además de la implementación de las colisiones, se hace un recorrido sobre la creación de figuras en escenarios 3D en OpenGL, junto a la revisión de las acciones disponibles para controlar a un personaje en primera persona.

Como autocrítica, se podría haber dedicado más recursos a la interacción con el sistema dados los problemas de paquetes que han imposibilitado el uso del mando de Nintendo Wii como control, pero como se hizo una buena separación de los objetivos de forma que avanzaba el desarrollo de forma iterativa, se han podido presentar alternativas para terminar con un trabajo totalmente funcional con el uso del teclado y del movimiento del ratón además de la alternativa propuesta con el controlador Dualshock 4. Dado que los objetivos estaban claros y bien distribuidos, se pudo reestructurar la planificación, por lo que bajo nuestro punto de vista hubo una buena organización en el proyecto.

Se le proponen al lector como mejora al juego base diferentes propuestas expuestas en el apartado 5, como añadir la interacción con algún mando de videoconsola, donde puede que los mandos de PlayStation o Xbox puedan tener mayor compatibilidad; nuevos power-ups, la posibilidad de romper la valla para obtener una mayor puntuación, o adaptar este juego para la realidad virtual.

7 Bibliografía

[1] Opengl-3D-sample.c, código base del proyecto. URL: <CMSC 425 (umd.edu)>

[2] Definición Endless Run. URL: <Endless runner | Definición en GamerDic>

[3] Temple Run. URL < Temple Run - Apps on Google Play>

[4] Subway Surfers. URL <Subway Surfers - Apps on Google Play>

[5] Figura Geométricas Glut. URL <11 Geometric Object Rendering (opengl.org)>

[6] glTranslate Doc. URL<glTranslate (khronos.org)>

[7] glScaled Doc. URL <glScale (khronos.org)>

[8] Transformaciones básicas en OpenGL. URL <Basic Transformations in OPENGL - GeeksforGeeks>

[9] Generar números aleatorios. URL <C library function - rand() - Tutorialspoint>

[10] glutBitmapString Doc. URL <glutbitmapstring(3) - Linux man page (die.net)>

[11] glutBitmapCharacter Doc. URL <10.1 glutBitmapCharacter (opengl.org)>

[12] glutStrokeString Doc. URL <glutstrokestring(3) - Linux man page (die.net)>

[13] OpenGL Examples, Dinospin.c. URL <OpenGL - Examples>

[14] Conversión Int a String. URL <How to convert integer to string in C? - Stack Overflow>

[15] Referencia sobre Colisiones. URL <collision.pdf (peroxide.dk)>

[16] GlutMotionFunc. URL <https://www.opengl.org/resources/libraries/glut/spec3/node51.html>

[17] GlutGet. URL <https://www.opengl.org/resources/libraries/glut/spec3/node70.html>

[18] WMINPUT. URL <https://ubuntu.pkgs.org/20.10/ubuntu-universe-amd64/wminput_0.6.91-1_amd64.deb.html>

[19] cwiid. URL <https://github.com/abstrakraft/cwiid>

[20] DS4DRV Driver Official. URL <GitHub - chrippa/ds4drv: A Sony DualShock 4 userspace driver for Linux>

[21] Setuptools Página Principal. URL <pythonhosted.org>

[22] Setuptools Página Alternativa. URL <setuptools · PyPI>

[23] Instalación BlueZ y corrección de errores. URL <apt - Install bluetooth or bluez-utils packages fails using bluez-5.37 (or bluez-4.101) - Ask Ubuntu>

[24] pyudev. URL <pyudev – pure Python libudev binding — pyudev 0.21.0 documentation>

[25] Python-evdev URL <Introduction — Python-evdev>

[26] Udev rules file. URL: <ds4drv/50-ds4drv.rules at master · chrippa/ds4drv · GitHub>

[27] DS4DRV Configuración. URL < ds4drv/ds4drv.conf at master · chrippa/ds4drv · GitHub>

[28] Synthetic Pleasures – Mokka. URL <80's Synthwave by MOKKA [No Copyright Music] / Synthetic Pleasures - YouTube>

[29] Librería de sonidos Gratuita. URL <Download Free Sound Effects for Videos | Mixkit>

[30] Ajustar volumen en OpenAL. URL <iphone - How to adjust the volume of a sound in OpenAL? - Stack Overflow>

[31] OpenAL Guía. URL <Complete Guide to OpenAL with C++ Part 2 | IndieGameDev>

[32] Práctica 3 SMII, Audio Posicional con OpenAL 2020/21 & Ejercicio ejemplo openal_escena.c. Disponible en el poliformaT de la asignatura.

[33] DS4 Para Windows. URL <http://ds4windows.com/>

[34] Input mapper. URL <https://beta.inputmapper.com/ >