Así se hizo TheWormhole

Ander Dorado Bolé, Adrián Roselló Pedraza y José Carlos Zambrana Navajas

Índice

1 Descripción **3**

2 Detalles técnicos relevantes **3**
2.1 Scroll por Hardware 3
2.2 Personaje 4
2.3 Plataformas 4
2.4 Colisiones 4
2.5 Físicas 5
2.6 Enemigos 6
2.6.1 Viser 6
2.6.2 Patrol 7
2.6.3 Láser 8
2.6.4 Menciones 9
2.7 Música 9
2.8 Tilemap 10

3 Dificultades en el proceso de desarrollo **11**

4 Guiño **12**

5 Conclusión **12**

1. Descripción

The Wormhole es un juego de plataformas donde la meta es conseguir pasar todos los niveles hasta llegar al final. En el transcurso de cada nivel te enfrentas a varios enemigos que te dificultaran completar el juego.

La mecánica principal del juego es dejarse caer. Esta mecánica la hemos basado en el juego *Downwell*.

2. Detalles técnicos relevantes

2.1. Scroll por Hardware

El juego utiliza Scroll por Hardware para realizar el efecto de caída. Con este mecanismo obtenemos una mejor sensación de juego y un mejor rendimiento al no tener que repintar toda la pantalla.

Para simplificar el scroll, la pantalla se ha configurado para que sea de 32 chars de ancho.

El scroll esta configurado para que se realice cada vez que el personaje cruza un threshold preestablecido, manteniendo la posición vertical del personaje estática (posiciones de pantalla). La función encargada de realizar esta acción esta basada en este código `https://cpctech.cpc-live.com/source/hardscrl.html`.

2.2. Personaje

A la hora de pensar en el personaje protagonista, llegamos a la conclusión de que lo más acertado es un astronauta ya que todo el mundo del juego se desarrolla en el espacio, en concreto en un agujero de gusano.

Figura 1: *Personaje*

2.3. Plataformas

Las plataformas en nuestro juego se consideran entidades no renderizables. La renderización se genera cada vez que se hace un scroll, pintándolas con el sprite referenciado en el Tilemap de cada nivel.

Al mantener al personaje siempre por encima del threshold, para optimizar el rendimiento, solo se crean cuando el personaje tiene la posibilidad de colisionar con esta, nunca antes. Así evitamos el calculo de calcular colisiones con entidades que no es posible colisionarse.

2.4. Colisiones

Nuestro sistema de colisiones se encarga de comprobar las colisiones y rectificar la posición o matar al jugador. Nuestras colisiones se pueden dividir en tres tipos: colisiones con plataformas, colisiones con pinchos y colisiones con enemigos. Para las plataformas se hacen cuatro comprobaciones, una por

cada lado, de forma que si colisiona con alguno de ellos ajusta la posición en el eje correspondiente para que no atraviese la misma.

Para los pinchos, se comprueba si el personaje está dentro de su hitbox y, si lo está, mata al jugador. Cabe destacar que sabemos que la hitbox lateral de los pinchos es un poco permisiva, es algo que hemos hecho a propósito para no hacer el juego extremadamente difícil y desesperante.

Con respecto a los enemigos, se comprueba lo mismo que para los pinchos. La hitbox para estas colisiones está programada para ser más precisa para que el juego no sea demasiado fácil. En versiones anteriores a la actual, permitíamos que el jugador pudiese matar a los enemigos saltándoles en la zona superior, pero nos deshicimos de esta característica ya que consideramos que la dificulta disminuía demasiado.

2.5. Físicas

A la hora de implementar un sistema de físicas que se adecuase al estilo propuesto con el *scroll* descendente, había que implementar un sistema de físicas que tratase de una manera más realista con el movimiento del personaje por la pantalla.

En nuestro caso, optamos por añadir una componente de aceleración sobre el sistema, esto no solo permitiría hacer un salto en pocos pasos, sino que el movimiento posteriormente de los enemigos sería más dinámico, aunque, por contra, se complican otras comprobaciones a realizar, pues la velocidad a la que ahora se van a mover las entidades no es constante.

Por otro lado, tener un sistema de este tipo permite que a la hora de aplicar diferentes fuerzas, como puede ser la gravedad o la de un salto del jugador, se sumen para producir un resultado realista. Este sistema es referenciado en el libro *The Nature of Code (Daniel Shiftman)*.

2.6. Enemigos

Los enemigos funcionan coexisten con el sistema de físicas que retroalimenta el estilo de juego de *The Wormhole*. El hecho de tener que descender progresivamente permite que los enemigos actúen sobre esta intención y genere comportamientos muy curiosos.

En la práctica, se han desarrollado tres comportamientos específicos aunque en un inicio formaban parte de este grupo otros muchos comportamientos que al final se descartaron y que mencionaré posteriormente. Dichos tres comportamientos presentes en el juego se tratan de las entidades: Viser, Patrol y Laser.

2.6.1. Viser

Este enemigo de 6x6 píxeles presenta un comportamiento curioso, se mantiene en un estado *stand by* hasta que el jugador desciende por debajo del mismo. En ese momento, se activa y pasa a perseguir al jugador. El jugador, entonces, deberá esquivarlo pues al tocarlo morirá instantáneamente, siendo su única escapatoria matar al enemigo bajando lo suficiente como para que el desplazamiento de la pantalla lo mate.

Cabe señalar que, intencionadamente, el movimiento de este enemigo no es completamente preciso, aunque es más veloz que el jugador, se puede engañar haciendo diferentes movimientos de los que esperamos que el usuario medio haga uso como una mecánica más en los niveles propuestos.

Además, y como última característica de este enemigo, no colisiona con su entorno, esto provoca. En parte por las características de este proyecto, sería inviable en este momento crear un enemigo con colisiones con el entorno y, más importante, a la hora de jugar, los enemigos no supondrían una amenaza ya que enseguida colisionarían.

También, añadir que este enemigo no fue inicialmente propuesto de esta manera, sino que al comienzo del desarrollo se plantearon enemigos que no esperasen al enemigo y ya entrasen en fase de persecución desde un inicio,

aunque en ese momento del desarrollo los enemigos también se podían matar saltado sobre ellos.

Pero nos percatamos que al combinar estas dos características el juego perdía encanto, pues al perseguir desde un inicio, siendo su origen la parte inferior de la pantalla, el jugador no tenía que realizar mucho esfuerzo para eliminar al enemigo, siendo este hecho el causante de que eliminásemos la posibilidad de matar enemigos y de que comenzase a perseguir desde la parte superior de la pantalla.

Figura 2: *Viser*

2.6.2. Patrol

Este enemigo de 6x6 píxeles presenta una rutina de movimiento constante, inicialmente aparece en la posición destinada de la pantalla y, a continuación, según uno de sus atributos que funciona como la distancia de movimiento, se calcula cual es su posición objetivo (*target*) - que se aloja en otro atributo de la entidad - y se procede a seguir una rutina de movimiento entre dicha posición objetivo y la origen.

Una cuestión a señalar de este enemigo es que la intención con este enemigo inicialmente era que su intervalo de movimiento se pudiera alterar, pudiendo cambiar la longitud y dirección de la patrulla del enemigo. No obstante, la escasez de tiempo hizo mella y tuvimos que dejar esa característica para un futuro del proyecto.

Figura 3: *Patrol*

2.6.3. Láser

Este último enemigo incorporado se trata de un láser que se activa y se desactiva según un temporizador. Cuando el láser se encuentra activado, como se puede suponer, el jugador morirá al entrar en contacto con el haz, en cambio, el jugador podrá pasar a través de esta entidad cuando se encuentre desactivada, este estado se puede visualizar claramente mediante las animaciones que acompañan a la entidad.

Este enemigo también se pensó hacerlo de tamaño variable, como sucedió con el patrol, pero a las alturas que se desarrolló esta entidad ya tomamos la decisión de no hacer variable la ruta del anterior enemigo, por lo que en este caso no pudimos actuar de otra manera y mantener fijo el tamaño del láser.

Figura 4: *Laser*

2.6.4. Menciones

Otros enemigos se plantearon en su inicio, diferentes versiones del viser se probaron como se ha mencionado anteriormente. No obstante, hay un enemigo que completamente se eliminó por su poca compatibilidad con la forma de juego y que nos gustaría mencionar.

Un enemigo al que apodamos como *frog* y cuyo objetivo era saltar hacia la dirección del jugador. Y, aunque se completó el comportamiento del enemigo, finalmente no se acabo de incluir en los mapas finales, la forma de cargar los mapas junto con (nuevamente) el límite de tiempo nos instó a recortar a este enemigo del presupuesto inicial.

Con todo esto, mencionar que dicho enemigo habría sido un incentivo para el jugador de no descender en la pantalla muy rápidamente, dando variedad al contenido ya existente.

2.7. Música

El sistema de música del juego funciona en combinación con el sistema de interrupciones, encontrándonos con un manejador de interrupciones y otro de música. De esta manera cada seis interrupciones se llama a una nota diferente de la canción que se está reproduciendo.

Esto se realiza de esta manera debido a que, como hemos forzado que las interrupciones se llamen siempre en el mismo intervalo de tiempo, si llamamos a tocar una nota de la canción cada seis interrupciones estaremos tocando la música a 50Hz, que es la frecuencia base de la que partía toda la música que hemos realizado.

Por otro lado, dentro del manejador de música hemos hecho varias funciones que permiten alternar entre diferentes pistas de audio dependiendo del nivel que se cargue en el juego, de esta manera, conforme avancemos en el juego diferentes piezas musicales sonarán. Incluir, por último, que dichas piezas musicales al cambiarse se irán difuminando para introducir a la siguiente canción, es decir, hemos implementado un sistema de música

con *fade in/out*.

Por último, en cuanto a los efectos especiales, hemos diseñado efectos especiales para el salto, la muerte del jugador y pasar entre niveles. Dichos efectos especiales coexisten con la música simultáneamente. La tecla M ha sido habilitada para silenciar el sonido, no obstante, funciona en algunas ocasiones y hay que presionarla varias veces si no silencia el sonido, esto creemos que puede ser porque se silencia y des-silencia con la misma pulsación, pero no hemos tenido tiempo de investigarlo.

2.8. Tilemap

Los mapas de cada nivel están especificados como Tilemaps. La lectura del tilemap se hace linea a linea. La función de lectura del tilemap se encarga tanto de leer el contenido renderizable, por ejemplo el fondo y de crear las entidades, tanto enemigos como plataformas. El tilemap mantiene dos punteros en todo momento a la dirección de memoria. La primera es a la siguiente linea a renderizar, y la segunda es a la siguiente linea de plataformas que hay que crear. Esto se utiliza para mantener solo las plataformas colisionables mencionadas en el apartado de Plataformas.

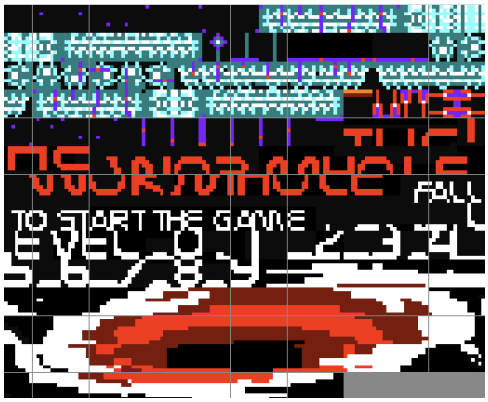

Figura 5: *Tiles utilizados*

3. Dificultades en el proceso de desarrollo

En enemigos, eliminar la posibilidad de eliminarlos y enemigos en menciones.

Tomar la decisión de realizar el scroll por hardware y la modificación del formato de pantalla provoco que tuviéramos que redefinir muchas funciones de Cpctelera para que se adaptaran a nuestro caso.

El error más notable que no hemos podido solucionar a tiempo se da en casos específicos del renderizado de entidades en el limite de pantalla. Este error no ocurre siempre, solo en algunos casos, y es por eso que no hemos podido identificar a tiempo el origen de ese bug. Como consecuencia es posible visualizar el renderizado de una entidades durante un instante en una posición errónea.

4. Guiño

El Guiño que hemos implementado en el juego se encuentra en el nivel 5. En este nivel, hemos modificado la paleta del juego y el diseño de las plataformas ajustándola a las que encontramos en *Larcena's Legacy*. Teniendo en cuenta que el juego que hemos implementado es de tipo plataformas, hemos creído que utilizar las plataformas que hay en *Larcena's Legacy* era correcto.

Figura 6: *Plataformas de Larcena's Legacy ingame*

5. Conclusión

Como conclusión de todos los integrantes del grupo, la creación de este juego ha sido una gran experiencia que es muy probable que repitamos en otra edición, ahora sí, con más tiempo.