



Universitat Oberta
de Catalunya

Elias y Juan, un juego infantil para ZX Spectrum

Autor: Jose Manuel Espina Rama

Tutor: Gustau Marcos Ballester

Profesor: Joan Arnedo Moreno

Grado de Ingeniería Informática

Itinerario en Sistemas de información

Fecha de entrega: junio de 2024

Créditos/Copyright



Esta obra está sujeta a una licencia de Reconocimiento
[3.0 España de Creative Commons.](https://creativecommons.org/licenses/by/3.0/es/)

B) GNU Free Documentation License (GNU FDL)

Copyright © 2024 JOSE MANUEL ESPINA RAMA.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU Free Documentation License".

Este proyecto usa rutinas de dos terceros, que tienen licencia CC Attribution-Noncommercial-Share Alike 4.0 International.

Cuando se hable de los ficheros de código ensamblador del proyecto se detallara el origen de esas rutinas.

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Elías y Juan, un juego infantil para ZX Spectrum</i>
Nombre del autor:	<i>Jose Manuel Espina Rama</i>
Nombre del colaborador/a docente :	<i>Gustau Marcos Ballester</i>
Nombre del PRA:	<i>Joan Arnedo Moreno</i>
Fecha de entrega:	<i>06/2024</i>
Titulación o programa:	<i>Grado de Ingeniería Informática</i>
Área del Trabajo Final:	<i>TFG - Videojuegos</i>
Idioma del trabajo:	<i>Español</i>
Palabras clave	<i>retro, infantil, spectrum</i>
Resumen del Trabajo (máximo 250 palabras): <i>Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo</i>	
<p>El objetivo de este proyecto es crear un videojuego retro de “scroll” lateral. Estaría pensado para niños de 3 a 5 años. Trataría sobre recorrer unos laberintos hasta un vehículo, con la opción de coger objetos que “hacen cosas”.</p> <p>Se podrá usar con la maquina real ZX Spectrum 48 o con emuladores.</p> <p>Se desarrollara con el SpectNetIde 2.0 (Que es una extensión para el Visual Studio 2019) y GitHub.</p>	
Abstract (in English, 250 words or less):	
<p>The proposal of this project is make a retro video game with lateral scroll. It'd be though for children between 3 to 5 years old. It would be about walk many mazes until a vehicle, with the option of pick up objects than “make things”.</p> <p>I can be used with the real machine ZX Spectrum 48 o with emulators.</p> <p>It will develop with SpectNetIde 2.0 (Which is an extension for Visual Studio 2019) and GitHub.</p>	

“No cabe duda de que una de las mayores inquietudes de los aficionados al Spectrum pasa por la creación de su propio juego.” [1]



Figura 1: Recorte de la Microhobby 171 página 58.

Agradecimientos

A Santiago Romero por sus rutinas de sprites y de mapeado con scroll.

A Juan Antonio Rubio García por su taller de ensamblador para ZX Spectrum 16K – Pong y por sus tutoriales.

Abstract

The proposal of this project is make a retro video game with lateral scroll. It'd be though for children between 3 to 5 years old. It would be about walk many mazes until a vehicle, with the option of pick up objects than "make things".

I can be used with the real machine ZX Spectrum 48 o with emulators.

It will develop with SpectNetIde 2.0 (Which is an extension for Visual Studio 2019) and GitHub.

Palabras clave

Retro, Infantil, Spectrum

Notaciones y Convenciones

- Las palabras en inglés están escritas entre comillas. (Salvo scroll, engine, sprite, hardware, joystick y monster truck).
- Los nombres de juegos irán en negrita.
- Las definiciones de abreviaturas están escritas en cursiva entra paréntesis.
- Entre comillas y en cursiva van las citas directas de otros autores. Siempre se acompañaran de un numero de referencia en la bibliografía.

Índice

1. Introducción.....	12
1.1. Introducción.....	12
1.2. Descripción.....	15
1.3. Objetivos generales.....	16
1.3.1. Objetivos principales.....	16
1.3.2. Objetivos secundarios.....	16
1.4. Metodología y proceso de trabajo.....	17
1.5. Planificación.....	18
1.6. Presupuesto.....	20
1.7. Estructura del resto del documento.....	21
2. Análisis de mercado.....	22
2.1. Público objetivo y perfiles de usuario.....	22
2.2. Competencia/Antecedentes	22
2.3. Análisis DAFO.....	25
3. Propuesta.....	27
3.1. Definición de objetivos.....	27
3.2. Modelo de negocio.....	27
3.3. Estrategia de marketing.....	28
4. Diseño.....	31
4.1. Entorno de desarrollo.....	31
4.2. Requisitos técnicos del entorno de desarrollo.....	32
4.3. Inventario y breve descripción de todas las herramientas utilizadas.....	32
4.4. Inventario de elementos del juegos.....	33
4.6.1. Ficheros.....	33
4.6.2. Gráficos del mapeado.....	37
4.6.3. Gráficos del mapeado con los que se interactúa.....	39
4.6.4. Sprites elementos del juego.....	40
4.5. Arquitectura del juego.....	41

4.6. Diseño de niveles y elementos del juego.....	43
4.7. Mapa de colisiones.....	51
4.8. Dificultades.....	53
5. Implementación.....	57
5.1. Requisitos de instalación.....	57
5.2. Instrucciones de instalación.....	57
6. Demostración.....	58
6.1. Instrucciones de uso.....	58
6.2. Prototipos.....	58
6.3. Tests.....	58
7. Conclusiones y líneas de futuro.....	61
7.1. Conclusiones.....	61
7.2. Líneas de futuro.....	62
Bibliografía.....	63

Figuras y tablas

Índice de figuras

Figura 1: Recorte de la Microhobby 171 página 58.	4
Figura 2: El ordenador para niños	13
Figura 3: Gameplay	13
Figura 4: Pantalla usada para programar el motor de mapeado, movimiento de sprite y control de colisiones.	14
Figura 5: Diagrama de Gantt	19
Figura 6: Captura de https://noentiendo.itch.io/gandalf	24
Figura 7: Primer boceto del sprite del personaje con el SevenUP siguiendo una foto reducida en tamaño del dibujo de mi hijo.	33
Figura 8: Haciendo la flor a partir de una imagen de internet y reduciéndola para simplificarla.	40
Figura 9: Pantallas y niveles del juego.	41
Figura 10: Captura de como se ve el nivel 1 completo	43
Figura 11: Nivel 2 realizado con tiled.	43
Figura 12: Nivel 3 realizado con tiled.	44
Figura 13: Nivel 4 realizado con tiled.	44
Figura 14: Nivel 5 realizado con tiled.	45
Figura 15: Captura del final del nivel 5.	45
Figura 16: Nivel 10 realizado con tiled.	45
Figura 17: Personaje va a coger las uvas en nivel 10.	46
Figura 18: Tras cogerlas, los atributos de los colores se han invertido.	46
Figura 19: Nivel 11 realizado con tiled.	46
Figura 20: Nivel 12 realizado con tiled.	46
Figura 21: Nivel 13 realizado con tiled.	47
Figura 22: Nivel 14 realizado con tiled.	47
Figura 23: Nivel 20 realizado con tiled.	48
Figura 24: Personaje ante la pared falsa del nivel 20.	48
Figura 25: Tras tocar la pared falsa se puede seguir.	48
Figura 26: Nivel 21 realizado con tiled.	49
Figura 27: Personaje ante el gran huevo frito tricolor.	49
Figura 28: Tras tocarlo, aparece al otro lado del muro.	49

Figura 29: Nivel 22 realizado con tiled.	49	
Figura 30: Nivel 23 realizado con tiled.	50	
Figura 31: Nivel 24 realizado con tiled.	51	
Figura 32: Al principio las flores "rompían" el juego.		55
Figura 33: Configurando en el ZX el joystick Kempston	57	
Figura 34: Elías disfrutando del juego	59	

Índice de tablas

Tabla 1: Presupuesto	20
----------------------	----

1.Introducción

Sir Clive Sinclair no creo el Spectrum con la intención de hacer una maquina de juegos, su intención era lanzar un microordenador asequible para popularizar la programación en las escuelas y en los hogares. Previamente ya había lanzado el ZX80 [2] que fue publicitado como el primer ordenador personal por menos de 100 libras y el ZX81 [3] que vendió más de un millón y medio de unidades. Pero con el ZX Spectrum [4] se consiguió una autentica revolución vendiendo 5 millones de unidades y cambiando la vida de miles de adolescentes. (A pesar de tener la misma CPU a casi la misma velocidad, cambios en los buses internos permitían multiplicar su potencia 3 a 4 veces. [5]) Fue el ordenador más popular a mediados de 1980 en países como Reino Unido y España, alargándose mucho más su vida en países de Europa del este.

Lo que lo hizo tan popular fue su bajo precio (125 libras) y los juegos. Creo toda una industria y dio pie a la llamada “Edad de Oro del software español” [6]. Adonde adolescentes programaban sus juegos en sus dormitorios de casa de sus padres.

1.1. Introducción

Mi propuesta es hacer un juego infantil para ZX Spectrum. Quiero hacer un juego para un público infantil, para que esa sea su primera toma de contacto con los videojuegos. Considero mejor que empiece con los videojuegos con un joystick que con una pantalla táctil.

Para mi TFG quería hacer algo que me entusiasmara. Desde niño he querido hacer un juego y ya que se me ofrece la oportunidad de hacerlo, voy a aprovecharla.

Tengo un niño de 4 años al que le regalaron un viejo ordenador para niños. La pantalla era una matriz de puntos monocroma, nada de los ordenadores para niños actuales a color, me gusta que sea así. Ese viejo ordenador tenia un juego llamado "El Laberinto", que trata de guiar a un personaje a través de unas pantallas usando un “scroll” lateral, que le gusta a mi hijo.



Figura 2: El ordenador para niños



Figura 3: Gameplay

El personaje está en la parte izquierda. Aparecen dos enemigos, pero no son peligrosos. No se mueven y únicamente hay que evitar tocarlos. Mi hijo nunca se ha quejado de que pierda por ellos.

Lo que quiero hacer es un juego similar para Spectrum, para que mi hijo lo juegue y sea su primera toma de contacto con los videojuegos. Estoy usando el SpectNetIde 2.0, que es una extensión para el Visual Studio 2019.

Obviamente le quiero meter mejores gráficos a color. He pensado en 3 niveles de dificultad que vaya de los 3 años 5 años, en el último los gráficos serán más llamativos como se ve en la siguiente imagen que corresponde a unas de las primeras fases del desarrollo.



Figura 4: Pantalla usada para programar el motor de mapeado, movimiento de sprite y control de colisiones.

El personaje, en la esquina superior izquierda, está basado en un dibujo de mi hijo.

1.2. Descripción/Definición

El proyecto consiste en un juego con scroll multinivel. Está dirigido a un público infantil, mas concretamente a niños de 3 años cuyos padres disfrutaron con los juegos de 8 bits en los ochenta para que les enseñen la maquina con la que jugaban cuando eran niños.

No recuerdo ningún juego de Spectrum que fuera para un público infantil. Recuerdo el **Fumigator**, pero ese ya me parece demasiado complejo para una primera toma de contacto. El **3D Maze** consistía solo en moverse, pero sin un mapa me parece imposible para un niño pequeño. Recordemos que en los ochenta los juegos solían ser muy difíciles.

Es relevante porque lo retro está de moda. Hay eventos retro de muchas ciudades y estos años están saliendo más juegos que antes. Tras unas décadas sin juegos nuevos ahora vuelven a hacerse muchos juegos homebrew con un nivel que no tiene nada que envidiar al de los años ochenta.

El producto que se desea obtener es un juego en formato .tap que pueda ser cargado en maquinas Spectrum reales mediante un interface como divIDE [7] o jugado con un emulador.

Tengo muchas ideas y mi idea es intentar meter todo lo que me dé tiempo. Las cosas que considero imprescindibles son:

- Multinivel.
- Soporte para joystick.
- Poner objetos en forma de frutas que cambien cosas: aumentar tamaño del sprite, que aparezca otro personaje, el hermanito, que lo acompañe,...
- Al finalizar un nivel el personaje saldrá en un coche con un tamaño de sprite mayor.

1.3. Objetivos generales

1.3.1. Objetivos principales

Objetivos del producto:

- El objetivo es crear un juego infantil.
- Difundirlo por los canales retro.

Objetivos para el usuario:

- Divertirse con algo que no sea una pantalla táctil.
- Aprender a moverse por un laberinto. Distinguir entre bloques que le impiden el paso y caminos que le permiten avanzar. Conocer lo que es la meta.
- *“Los ejercicios con laberintos son un excelente recurso educativo para el trabajo de la atención con niños”. “Las fichas de laberintos son una gran opción a la hora de trabajar la falta de atención en niños. Los ejercicios de estimulación cognitiva de la atención ayudan a mejorar el TDAH, o Trastorno del Déficit de Atención e Hiperactividad.”*[8] Los laberintos reducen la impulsividad e inquietud motora de los niños con TDAH. [9]

Objetivos personales del autor del TF:

- Cumplir un sueño de la infancia.
- Hacer un producto que le pueda interesar a otros padres.
- ¡Volver a sentir unas cosquillas en el estómago, que sólo los noche-ochenteros llegaremos jamás a sentir!

1.3.2. Objetivos secundarios

Objetivos adicionales que enriquecen el TF.

- Pensar en el proyecto y desarrollar ideas y mejoras.
- Aprender desarrollo retro.

1.4. Metodología y proceso de trabajo

Para hacer un juego retro hay dos alternativas: usar unos de los “*engines*” existentes o programarlo a base de código.

Hay varios engines modernos (de los que hablare más adelante) para hacer juegos para Spectrum pero los descarte porque no permitían scroll, algo que el juego del ordenador para niños si que tenia. Programando a base de código se puede hacer en un lenguaje de alto nivel o en ensamblador.

Usando un lenguaje de alto nivel hay básicamente dos alternativas. Primero tenemos el Boriel Basic del que hace unos meses salio un libro llamado “Boriel Basic para ZX Spectrum: Manual para torpes...y para los que no lo son tanto” de Juan Segura Duran.[10] La otra alternativa es Z88DK [11] para desarrollar en C que cuenta con la librería SP1, que es un engine para manejo de sprites sin parpadeo.[12]

Se hizo el desarrollo en ensamblador porque hace 3 años hice un taller de ensamblador [13] y me parece que así estoy más cerca del hardware de la maquina.

Después de una extensa búsqueda encontré unas rutinas que permitían hacer scroll sobre un mapeado. También encontré rutinas sobre dibujar y mover un sprite. Tras muchas noches muy intensas y mucho aprendizaje (errores) conseguir desarrollar mi propio mapa de colisiones, con lo que ya tenia la base del juego.

Eso me dio la confianza en que lo iba a poder realizar, aunque quedaba mucha tarea por delante. A continuación se desarrollo el multinivel y el menú inicial.

1.5. Planificación

Fechas claves de este proyecto:

24 de marzo, entrega de la PEC1

21 de abril, entrega de la PEC2

Sobre el estado del arte, conozco lo que se está haciendo actualmente para el ZX Spectrum. Había un grupo ruso, Zosya, que estaba haciendo unos juegos increíbles. (Como por ejemplo **Travel Through Time Vol.1: Northern Lights**, el mejor juego de coches que he visto para Spectrum) Pero la mayoría de los juegos usan engines accesibles al público en general y son más parecidos, los juegos van pantalla a pantalla, aunque artísticamente son muy destacables.

Estoy usando GitHub para el control de versiones. La dirección del repositorio es:

<https://github.com/jmespina/Spectrum>

19 de mayo, entrega de la PEC3.

Allí se pide una alpha y explicar todos los aspectos técnicos del trabajo. Supongo que allí congelare el desarrollo del juego y luego solo ampliare el número de niveles.

Lo más complicado de este proyecto es hacer el engine, que el personaje se mueva por donde se supone que se debe mover y no se “coma” las paredes.

16 de junio, entrega de la PEC 4

Que es la versión definitiva de la memoria. Quiero comentar todo lo posible mi código y el entorno de programación, de forma que pueda ser usado como una guía para otros aficionados a la programación retro.

30 de junio, PEC 5

Que es la defensa.

Hitos:

- Conseguir que el mapa de colisiones funcione y permita que el personaje se mueva sin entrar en las paredes. Conseguido.
- Multinivel. Conseguido.

- Soporte de joystick Kempston. Conseguido.
- Menú de inicio. Conseguido.

Vemos los hitos principales en el siguiente diagrama de Gantt:

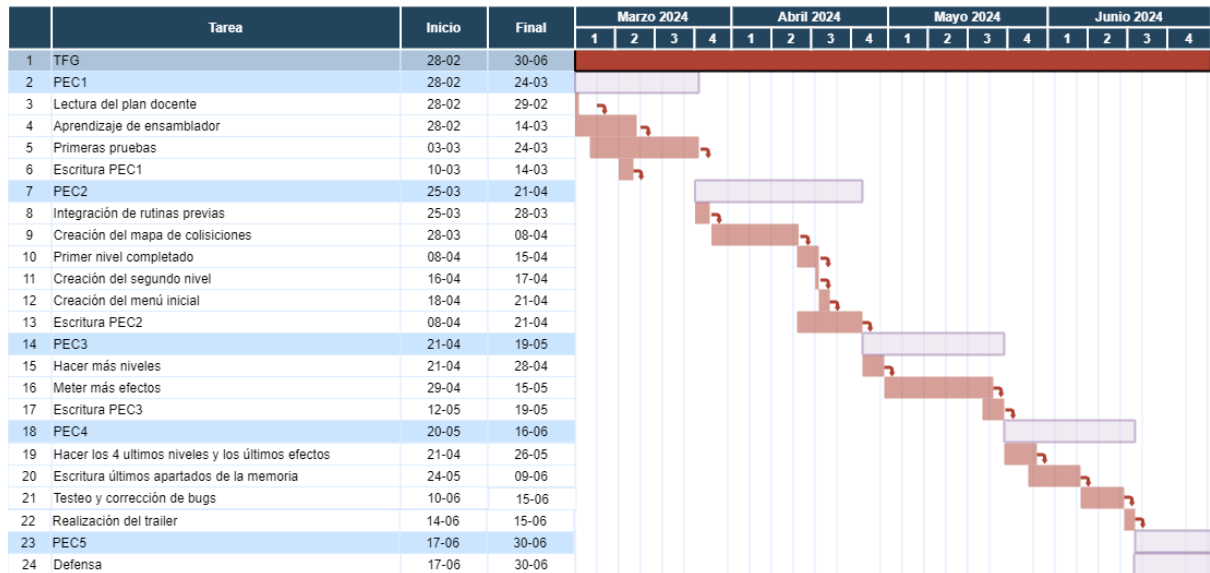


Figura 5: Diagrama de Gantt

1.6. Presupuesto

Para un proyecto de este tipo no se necesita un PC potente, de hardware quizás lo único a destacar es que recomiendo un monitor de 32" por ser más cómodo para la vista.

Hardware		
PC económico		500€
Monitor 32"		160€
Software		
SpectNetIde	IDE	0€
Visual Studio 2019	IDE	0€
GitHub	Control de versiones	0€
Equipo humano		
1 Desarrollador	2 horas diarias de 1 de marzo a 30 de junio. (122 días x 30€/hora)	3660€
Total		
		4320€

Tabla 1: Presupuesto

1.7. Estructura del resto del documento

En el apartado 2 vamos a hablar de la situación actual en el mundo retro y del estado del arte. Veremos engines y grupos de desarrolladores.

En el apartado 3 vamos a definir detalladamente este juego y el modelo de negocio.

En el apartado 4 vamos a ver el diseño del juego, con capturas de sus diferentes pantallas. También se hablara de como funciona internamente el juego.

En el apartado 5 vamos a ver como ejecutar el juego, tanto a través de un emulador como en un Spectrum real.

En el apartado 6 veremos los controles y una guía de usuario, adonde se detallara cada nivel.

En el apartado 7 vamos a hablar de las cosas que se han quedado en el tintero y posibles mejores. También de lo que he aprendido de esta experiencia y de como afrontaría el reto de realizar un juego diferente.

2. Análisis de mercado

En este capítulo vamos a situar al lector en el mundillo.

2.1. Público objetivo y perfiles de usuario

Este es un producto de nicho orientado a padres que desean que sus hijos conozcan la informática de la mano del primer ordenador que ellos mismo tuvieron. Por un lado, el público tiene que estar interesado en el mundillo retro y por el otro, no es un producto orientado a adultos, sino a niños de 3 a 5 años.

La dificultad está inspirada en el ordenador para niños que vimos antes, así que no hay “muertes” ni dificultades. Así un adulto podría pasarse el juego “del tirón” en unos 15 minutos, pero este juego supondría un reto para un niño pequeño. Así, los elementos que se encuentran en el juego: frutas, comida, “monster trucks” y bebés,... componen la vida de un niño y harían el juego interesante para ellos.

Desde el menú inicial se puede seleccionar la edad y eso nos lleva a unos niveles con diferente dificultad. Los de edad 3 son más fáciles, cortos y nada angostos. Los de edad 4 son más largos y aparecen elementos (las frutas) para llamar la atención de los niños. Los de edad 5 aparte de ser más largos y angostos tienen elementos que para un niño pueden sugerir que no se puede continuar y que necesita la ayuda de un adulto para continuar. Así el camino se encuentra bloqueado por paredes falsas o por gruesos muros que solo se pueden superar con el teletransporte.

2.2. Competencia/Antecedentes

La única competencia existente son los ordenadores para niños, pero a primera vista en los modelos que encuentro en las tiendas y en amazon estoy viendo se ha sustituido el joystick por el ratón. No sé como habrán adaptado los juegos al uso del ratón en vez de un joystick, supongo que los niños tendrán que pulsar un botón de dirección para avanzar (Como en el **Bloodwych** de 1990 de Spectrum). Creo que para ellos es más natural moverse con un joystick que con un ratón.

Eso en el ámbito de juegos para niños, sobre el estado del arte de los juegos para Spectrum permitamen explayarme pues es algo de lo que voy a disfrutar hablando.

En los ochenta existían engines para desarrollar juegos más fácilmente. La mayoría eran de uso interno de las compañías y no eran accesibles al publico. Pero algunos si que estaban en venta al publico y permitían hacer juegos comerciales con el propio Spectrum:

- PAWS (*Professional Adventure Writing System*) para hacer aventuras conversacionales. Con él, St. Bride's School desarrollo **Jack the Ripper**, el primer juego cuya venta a menores de edad estaba prohibida.
- Freescape para hacer juegos 3D como **Total Eclipse** (1988). En este caso era de uso interno y en 1991 se vendió al público con el nombre de **3D Construction Kit**.
- Filmation fue un engine para hacer juegos isométricos como **Batman** (1986) o **Head over Heels** (1987). Era de uso interno y no tengo constancia de que se abriera al público, pero un creador de juegos usando una técnica similar si que fue puesto a la venta en 1987, el **3D Game Maker**. [14]

Tras el fin de la época dorada de los 8 bits, hubo que esperar casi dos décadas hasta que en 2010 el grupo de desarrollo de videojuegos retro The Mojon Twins sacara el primer engine de creación de juegos modernos, MK1 "La Churrera" [15]. Posteriormente sacaron el MK2. En su pagina web hay unos 90 juegos gratuitos para maquinas 8 bits desarrollados con sus herramientas.

Exprimiendo al máximo al Spectrum salieron BIFROST* Engine (2012) [16] y Nirvana (2013). Originariamente el Spectrum estaba pensado para mostrar 2 colores cada 8x8 pixeles y esos engines consiguen mostrar 2 colores por cada bloque de 8x1 pixeles con el BIFROST* y 2 colores cada bloque 8x2 con el Nirvana. (16 colores por bloque de 8x8 y 8 colores por bloque respectivamente.) Así permiten crear juegos mucho más coloridos, que no parecen de Spectrum, como el **Gandalf**. [18]



Figura 6: Captura de
<https://noentiendo.itch.io/gandalf>

Aunque visualmente sea lo más avanzado creo que son muy difíciles de programar.

Uno de los engines modernos más populares es AGD (2019). Posteriormente sacaron una versión actualizada llamada MPAGD con la que en 2022 salió **Metamorphosis**, un juego visualmente muy llamativo. [19] Este juego es gratuito y a cambio piden una donación a elección del comprador.

Hace poco salió el ZX Spectrum Game Maker, del que salió la noticia en el mundo del spectrum el 9 de abril [20]. Este nuevo creador de juegos aún está en fase beta.

Todos estos engines los descarte porque no permitían scroll, algo que el juego del ordenador para niños sí que tenía. Son juegos que van pantalla a pantalla y creo que eso puede resultar confuso para el público objetivo. Del mismo modo que **La Abadía del crimen** resultaba confuso para muchos de sus jugadores por sus cambios de cámara, creo que a un niño pequeño le puede costar trabajo entender que un personaje que sale por la derecha de la pantalla aparezca de nuevo en la izquierda.

Otro juego ambicioso es **Cesare the somnambule**. [21] Un miembro de su equipo contó en un grupo de telegram de desarrollo de ensamblador para z80 [22] que los gráficos habían ocupado toda la memoria del Spectrum 128k (quedaba libre menos de 1KB), incluso usando la poderosa compresión ZX0. (Y necesitaban memoria para meter las rutinas para que se

podiera jugar con disco en un Spectrum +3. La ROM del +3 puede manejar el disco, pero para ello reserva un banco completo de memoria de 16KB.)

Están saliendo también muchas aventuras conversaciones. Este 22 de abril salió **Innsmouth**, el nuevo trabajo del grupo español Bitfans [23] que sorprendieron en 2022 con su juego **Donum**. Ambos juegos son de pago (2'5€ y 3€ su edición digital, también lo han sacado en edición física). **Donum** destacaba por sus elaborados gráficos, que fueron compuestos mediante IA (DALLE-2)

Los juegos más ambiciosos, hasta hace año y medio, venían de Rusia. Un grupo ruso, Zosya, estaba haciendo unos juegos increíbles. (Como por ejemplo **Travel Through Time Vol.1: Northern Lights** [24], el mejor juego de coches que he visto para Spectrum) Ellos sí que cobraban por sus juegos, pero con las sanciones económicas que actualmente tiene Rusia su tienda solo está abierta en su país.

Igualmente ruso, el grupo Sanchez Crew sacó en 2020 **Delta's Shadow** [25], un juego muy ambicioso que exprimía al máximo al spectrum 128k. Incluso tenía opciones de primar velocidad o gráficos. Es uno de los pocos juegos que le sacan provecho al Spectrum Next. Tenía un scroll muy curioso en el que la pantalla no avanzaba todo el tiempo sino que avanzaba cuando el personaje llegaba a un punto. Hablando en plata, parecía un juego pantalla a pantalla pero el cambio de pantallas se hacía con scroll.

Y para terminar decir que muestra de la viveza del Spectrum en Rusia es que allí se organiza desde el 2019 la Yandex Retro Games Battle [26], el más potente concurso de juegos para ordenadores retro de 8 bits con un primer premio de unos mil euros. El año pasado el ganador fue **Serafima** de Zosya, un juego muy espectacular adonde destacaban sus sprites enormes.

2.3. Análisis DAFO

Características internas del proyecto.

Debilidades:

- Conocimiento limitado del ensamblador y de las técnicas de programación.
- Tiempo limitado para hacerlo, además de que se debe compaginar con escribir este documento.
- Limitado talento como grafista.

Fortalezas:

- Me entusiasma y no me importa trasnochar con esto.

Situación externa.

Amenazas:

- Que como ya me ha pasado, que el SpectNETIDE del Visual Studio me deje de funcionar. Eso tuvo como efecto secundario que tuve que crear otro repositorio. Aunque podría usar otro IDE, ya que me he acostumbrado a este, no quiero tener que aprender otro.
- Aunque no estén orientando para el público más joven, están saliendo muchos juegos homebrew (varios cada mes) que pueden restarle visibilidad a este proyecto.

Oportunidades:

- El público debe estar receptivo a probar un nuevo juego gratuito.
- Puedo consultar dudas técnicas en el grupo de Telegram anteriormente mencionado. (Como por ejemplo el diverso comportamiento de los emuladores con el joystick kempston.)

3.Propuesta

La propuesta del proyecto es sencilla, un juego para los niños de padres que quieran que sus hijos experimenten los 8 bits como ellos.

Recuerdo una vez que salio un Amstrad CPC 6128 en El Precio Justo: [27] “El futuro ya esta aquí es algo fácil de comprobar al ver como cualquier niño maneja con habilidad la que, para nosotros pueden resultar complicadas maquinas...”

En los ochenta muchos aprendimos a programar (al menos lo justo como para meter los listados de los cargadores de la Micromania), mientras que ahora hay universitarios que “*no conocen los fundamentos básicos sobre cómo funciona un ordenador.*”[28]

Supongo que siempre ha habido gente interesada en los ordenadores y gente interesada en otras cosas, pero creo que en general la sociedad ha involucionado. Este sera mi granito de arena para fomentar que a los niños les interese la informática.

3.1. Definición de objetivos/especificaciones del producto

Es un juego para que los más pequeños descubran la informática y se interesen por ella de una forma que tiendo más sana que mediante los dichosos teclados táctiles.

3.2. Modelo de negocio

En este proyecto he usado rutinas (la de dibujar el mapa y la de dibujar los sprites) bajo licencia “CC Attribution-Noncommercial-Share Alike 4.0 International”. Mi otra fuente de rutinas también usa la misma licencia. Así que para vender el juego tendría que hacer mis propias rutinas.

Es una tarea muy compleja que no voy a llevar a cabo. La rutinas de mapeado por bloques tiene en su bibliografía enlaces a la vieja revista Microhobby adonde explican esa técnica y suministran código. Pero me cuesta entender lo que explicaba la Microhobby, creo que es una técnica distinta y que el autor solo la tomo como vaga referencia.

Los grupos rusos vendían sus juegos, pero una alternativa que creo que me estaría permitida por la licencia es pedir donaciones, que es lo que hacia el juego, anteriormente mencionado, **Metamorphosis**.

Otra posibilidad es la de vender modificaciones del juego. En el juego pienso incorporar dos fotografías de mis hijos y podría pedir una pequeña cantidad a cambio de meter en el juego fotografías de los usuarios.

3.3. Estrategia de marketing

Tratare de venderle el juego a youtubers retro como El SPECTRUMERO Javi Ortiz [29] para que hable de él. También tratare de que hablen de él en la web de El Mundo del Spectrum [30] y lo publicare en el grupo de Telegram.

Mensaje para El SPECTRUMERO Javi Ortiz:

Buenos días,

soy un subcriptor de tu canal y he hecho un juego multinivel con scroll para Spectrum en ensamblador. Lo he hecho especialmente para mi hijo que tiene 4 años, pero creo que le puede interesar a otros padres para enseñar a sus hijos pequeños la maquina con la que jugaban cuando eran niños. Lo he hecho como TFG y en GitHub está todo el código y la memoria del TFG (que creo que podría ser de interés para los aspirantes a desarrollador). He usado con su permiso rutinas de <https://wiki.speccy.org/cursos/ensamblador/indice> y <https://espamatica.com/>

Es un juego infantil sin muertes y sin tiempo, inspirado en un juego del ordenador para niños "El ordenador de Bob".

El objetivo del juego es recorrer unos laberintos. Hay 15 niveles de dificultad ascendiente y se dividen en 3 grupos divididos por la edad, que

se seleccionan desde el menú inicial.

Recomiendo jugarlo con ZX Spin o Retro Virtual Machine. Se puede jugar con un joystick Kempston. En el menú inicial solo hay que seleccionar la edad introduciendo "3", "4" o "5" y luego se juega con las clásicas "QAOP".

Es un juego en el que he querido meter muchas sorpresas. En los primeros niveles solo nos encontramos con el coche, que es la meta. Pero luego aparecerán diferentes comidas que originan sorpresas. Las pongo al final del email por si quieres jugar sin conocerlas.

Puedes ver el trailer en: <https://youtu.be/O4ewmNfgltk>

Puedes descargar el .tap en:

[https://github.com/jmespina/Spectrum/blob/master/TapeFiles/EliasYJuan1](https://github.com/jmespina/Spectrum/blob/master/TapeFiles/EliasYJuan1.tap)
.tap

Te invito a que lo pruebes y le des difusión.

Muchas gracias,

JM

- coche con efecto flash, es la meta.
- uvas, le pone efecto flash a las paredes.
- chupete, hace que aparezca el hermanito.
- plátano, cambia TODOS los atributos, hasta los del fondo negro.
- pera, hace que aparezcan gradualmente flores por el mapeado.
- huevo frito, teletransportador.
- seta, para volver al inicio del nivel.
- muro falso.

4. Diseño

4.1. Entorno de desarrollo.

El entorno de desarrollo que he elegido es SpectNetIDE, una extensión para Visual Studio 2019.

Previamente había hecho un taller de programación en ensamblador sin un IDE (Codificando con el Notepad++, compilando con el Pasmó y testeando y debuggeando con el emulador ZEsarUX), pero usar un IDE tiene una serie de ventajas, como tener “a golpe de clic” la compilación, el debug y el GitHub. (Además de que permite usar otras extensiones como para contar automáticamente cuantos ciclos de reloj necesita la ejecución de una rutina, que no es importante en mi proyecto, pero para otros proyectos es crítico porque si se tarda demasiado no se pueden aplicar ciertos efectos como el color 8x1).

SpectNetIDE funcionaba bien hasta que a mitad del proyecto dejó de funcionar. Conseguí que volviera a funcionar pero entraba en conflicto con el repositorio que ya tenía y tuve que crear otro. Creo que el fallo fue por una actualización de windows que dejara de funcionar el sistema de .NET Framework. También tuve infinidad de problemas cuando intenté meterle las rutinas del sintetizador de voz de la MicroHobby (Uno de ellos era que el debug no hacía nada, pasaba por una orden de “LD A,56” y no cambiaba el valor del registro A.)

Una alternativa es Klive IDE [31] que es del mismo creador del SpectNetIDE. Lo probé e incluso me permitió detectar porque fallaba una cosa (Ver dificultades con fecha 22 de abril), pero dejé de usarlo porque no tiene la pseudo instrucción de división y modulo. Las pseudo instrucciones son muy populares desde el principio de la programación, cuando cada compilador tenía sus propias “particularidades” (En un código impreso en un libro de los ochenta los comentarios venían separados por una coma, lo que a primera vista parece como que instrucciones como “LD” están aceptando tres operandos cuando en realidad solo puede aceptar dos.). Así algunos compiladores aceptan instrucciones que realmente no existen e internamente las cambian por instrucciones válidas. En una rutina que he copiado de una web uso el símbolo “/” y el “%” para hacer esas operaciones. Eso el SpectNetIDE lo acepta, pero el Klive IDE no.

Podría haber programado esas operaciones matemáticas (Encontré una web con el código para hacerlas [32]), pero ya me había acostumbrado al SpectNetIDE y dice el dicho “más vale malo conocido que bueno por conocer”. Otra de sus ventajas es que permite añadir pantallas de carga automáticamente, es muy fácil.

Descubrí otro IDE similar (una extensión para Visual Studio 2022 llamada Felix [33]), pero no sabía como hacer debug. Contacte con el creador que me dijo que era muy fácil y que buscara información en internet. Ante esa dificultad lo deseché. (Es curioso como mucha gente hace lo más difícil que es crear un programa, pero luego no elaboran ningún manual ni graban en video un tutorial.)

4.2. Requisitos técnicos del entorno de desarrollo.

Visual Studio necesita una CPU de 64 bits. Eso es lo único que destacaría.

4.3. Inventario y breve descripción de todas las herramientas utilizadas.

Son todos gratuitos:

- SpectNetIde.[34] Que es el entorno de desarrollo con el que codifico, hago debug y subo a GitHub. Es una extensión para Visual Studio 2019.
- Tiled.[35] Que es un mapa para crear de manual visual mapas de juegos. Permiten generar unos mapas que se pueden integrar fácilmente en el proyecto. Crea los mapas con números y eso se puede copiar directamente en el SpectNetIde. (Hay que eliminar una coma final que el IDE no acepta)
- SevenUP [36]. Para hacer los sprites. Genera unos ficheros con números que se pueden cortar y pegar en el SpectNetIde.
- ZX-Paintbrush [37]. Para pasar imágenes .JPG a formato spectrum, para convertir la portada del juego.
- ZXSpin [38], emulador principal que he usado. No se actualiza desde 2009, pero es simple de manejar y funciona mejor que otros emuladores.
- Retro Virtual Machine [39], otro emulador más moderno. Carga el .tap a la antigua, o sea, tarda alrededor de dos minutos y va imitando los sonidos de carga. Creo que es uno de los emuladores que más se usan actualmente.

- ES.ppectrum [40], emulador para hacer debug del .tap del sintetizador de voz de la MicroHobby. Su depurador era fácil de usar y me permitía copiar las instrucciones del sintetizador a mi código. No he probado el juego en este emulador.
- scr2tap [41], para pasar un archivo de imagen .scr a .tap.
- Paint .Net [42], para ayudarme a crear los sprites y pasar las fotografías.

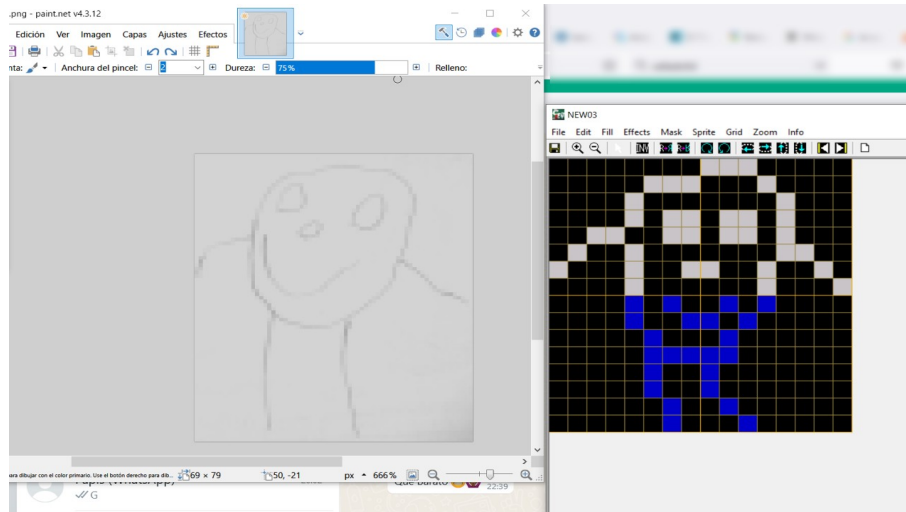


Figura 7: Primer boceto del sprite del personaje con el SevenuP siguiendo una foto reducida en tamaño del dibujo de mi hijo.

4.4. Inventario de elementos del juegos.

4.4.1. Ficheros.

A continuación voy a describir por lo alto los ficheros de código que hay en el repositorio (solo se mencionaran las rutinas más importantes):

Fichero Z80CodeFiles/EliasYJuan.z80asm.

Es el punto de entrada en el código. Contiene el bucle del menú y el del juego. Lee el teclado y el joystick. Mueve el personaje y el scroll. Contiene la definición de las variables. Este fichero contiene algunas rutinas de terceros, intente moverlas a rutinasPrevias.z80asm pero fallaban.

Rutina start:	El punto de entrada en el programa, por donde se empieza la ejecución.
---------------	--

bucleMenu:	Muestra el menú y espera a que se seleccione un nivel.
redraw:	Es el bucle principal del juego. Imprime el mapeado (rutina DrawMap_16x16_Map) y al personaje (rutina DrawSprite_16x16_LD). También llama a las rutinas frutasEspeciales para ver si hay que hacer algo más especial.
bucle:	Es la rutina que se ocupa de leer el teclado y el joystick. Rutina de https://wiki.speccy.org/cursos/ensamblador/gfx5_mapeados Modificada por mí para que antes de hacer un movimiento compruebe si este es posible.
leerJoystick:	Rutina de https://espamatica.com/0x0a-ensamblador-zx-spectrum-marciano-joystick/#Joystick
podemosIrDer:	Hay 4 rutinas para movernos en cada una de las 4 direcciones posibles. Como el juego consiste en avanzar hacia la derecha, ésta fue la primera rutina desarrollada. Si se ha pulsado la tecla “p” (o esa dirección en el joystick) esta rutina se ocupara de comprobar si el personaje se puede mover hacia la derecha examinando el mapa de colisiones.
moverBichoDerecha:	Si el personaje se puede mover a la derecha llegaremos a esta rutina. Comprueba si el personaje ha llegado al limite para moverse hacia la derecha (constante SHIP_TOP_R) y si es así en vez de mover al personaje llama a mover el mapa a la derecha haciendo scroll. (rutina MueveMapaIX)

Fichero Z80CodeFiles/rutinasPrevias.z80asm

Es el fichero adonde están las rutinas principales del juego de mapeado y dibujado de sprites. Casi todo el código proviene de dos autores, la página web de sus tutoriales son:

- <https://wiki.speccy.org/cursos/ensamblador/indice> [43]
- <https://espamatica.com/> [44]

Más detalles tras la descripción de cada rutina.

El fichero Principal.z80asm también contiene algunas rutinas de terceros. En el resto de ficheros todo el código es de mi autoría.

rutina	Lee el teclado.
LEER_TECLADO:	Rutina de https://wiki.speccy.org/cursos/ensamblador/teclado
DrawMap_16x16_Map:	Dibuja el mapeado según unos datos gráficos y un mapa de posiciones. También realiza el mapa de colisiones. Contiene código para marcar los objetos que hay en el mapeado. (Esas son modificaciones mías.) Rutina de https://wiki.speccy.org/cursos/ensamblador/gfx5_mapeados
MueveMapalX:	Cita del autor: <i>“Incrementar la variable DM_MAPX para scrollear a la derecha.”</i> Rutina de https://wiki.speccy.org/cursos/ensamblador/gfx5_mapeados
DrawSprite_16x16_LD:	Imprime un sprite de 16x16 pixeles. Rutina de https://wiki.speccy.org/cursos/ensamblador/gfx3_sprites_lowres
DrawSprite_MxN_LD:	Imprime un sprite mayor, se usa para los Monster trucks gigantes (hasta 72x48 pixeles). Rutina de https://wiki.speccy.org/cursos/ensamblador/gfx3_sprites_lowres#rutina-generica-trazado-de-sprites-multicaracter
PrintStringFF:	Para imprimir los textos que aparecen en el juego. Rutina de https://espamatica.com/0x09-ensamblador-zx-spectrum-marciano-partida/
PlaySound:	Para reproducir un sonido cuando el personaje choca contra una pared. Rutina de https://espamatica.com/0x0a-ensamblador-zx-spectrum-pong-sonido/#Sonido

Fichero Z80CodeFiles/cadenasTexto.z80asm

Contiene todas las cadenas de texto del juego. No tiene ninguna rutina.

Fichero Z80CodeFiles/frutasEspeciales.z80asm

Están las rutinas de efectos especiales que se provocan cogiendo frutas (y el huevo frito aunque no sea una fruta.)

FrutasEspeciales:	Rutina que cambia los atributos para que tengan efecto flash.
MostrarBebe:	Rutina que reutiliza la rutina de mostrar al personaje para mostrar al bebe. Para ello llama a la rutina, pero antes hay que cambiar los valores para que apunte a los datos gráficos del bebe y luego volverlos a cambiar por los del personaje.
comprobarDestino:	Comprueba lo que hay en la dirección de la tecla pulsada para realizar diversos efectos.
meterRetardo:	Ralentece el juego con 7 instrucciones halt para que no sea demasiado rápido. Cogiendo algunas frutas como las uvas se desactiva y el personaje va a toda velocidad.
frutasEspeciales4:	Pone las flores, su ejecución es diferente en el nivel 14 y en el nivel 24 por la diferencia de tamaño.

Fichero Z80CodeFiles/graficos.z80asm

Contiene todos los datos gráficos de todos los sprites del juego, desde las rocas del fondo a los personajes. También contiene los datos gráficos de la imagen final. No tiene ninguna rutina.

Fichero Z80CodeFiles/imprimirMapaColisiones.z80asm

Contenía las rutinas para mostrar el mapa de colisiones mientras lo iba desarrollando, pero tras conseguir que funcione bien, ya no tiene ninguna utilidad.

Fichero Z80CodeFiles/mapas.z80asm

Contiene todos los datos gráficos de todos los mapas del juego. No tiene ninguna rutina.

Fichero Z80CodeFiles/menu.z80asm

Contiene las rutinas para mostrar el menú de inicio y mandar a un nivel dependiendo de la

edad selecciona. La “Ñ” y el “¿” lo tuve que codificar como carácter creado por el usuario (UDG) para que se mostrara en emuladores con ROM inglesa. La fuente de caracteres lo saque de: https://fontstruct.com/fontstructions/show/118432/sinclair_zx_spectrum_es	
menu:	Abre el canal superior (muy importante en el Spectrum) y muestra los textos.
leerEdad:	Lee la edad proporcionada por el jugador.

Fichero Z80CodeFiles/meta.z80asm Contiene la rutina para mostrar la animación del final de cada nivel.	
meta:	Muestra los sprites de los enormes Monster trucks recorriendo la pantalla. Finalmente restaura los atributos si se han modificado por una fruta.










Fichero Z80CodeFiles/niveles.z80asm Selecciona el nivel del juego.	
seleccionarNivel:	Dependiendo del valor de la variable DM_NIVEL carga los datos de dimensiones y de mapa de cada nivel.

4.4.2. Gráficos del mapeado.

Al hacer el juego, lo primero es dibujar el mapeado, o sea, las paredes fijas que conforman el entorno adonde se va a desarrollar el juego. Esos gráficos se definen en 16x12 (la pantalla del Spectrum es formato 4:3) bloques de 2x2 *sprites*.

Cada sprite se define mediante ocho bytes más un bytes de atributos de color. 4 sprites conforman un bloque y una pantalla se define como un rectángulo de 16x12. Así una nueva pantalla solo consume 192 bytes de memoria. Aparte están los 36 bytes para definir cada bloques (8 bytes * 4 + 4 bytes de atributos)... 324 bytes en total. De esa forma los casi 7KB de datos necesarios para mostrar una pantalla se reducen muchísimo y consumiendo unos 35KB se pueden hacer unas 100 pantallas. (Este juego presenta 85 pantallas y quedan unos 10KB de memoria libre.)


Los bloques para mapeado que se han definido en el juego son (con esos números se han definido en los mapas):

	Bloque 1, que corresponde al terreno más denso, más profundo. Pensé que era lo más apropiado para las rocas más profundas.
	Bloque 2, con más huecos negros. Como el Spectrum permite darle brillo o no a un sprite de 8x8 lo use y se aprecia fácilmente que los 2 sprites superiores son de un azul más brillante que los dos inferiores.
	Bloque 3, con menos densidad.
	Bloque 4, con forma de saliente, para poder algunos en los bordes del mapeado.
	Bloque 5, con poca densidad en la parte superior, para poner en el borde inferior del mapeado.
	Bloque 6, que muestra una mínima densidad. Así la roca deja paso a la superficie por donde se mueve el jugador.
	Bloque 7, que es idéntico al bloque 1, pero que corresponde a una pared falsa.
	Bloque 17, similar al bloque 1, pero en dos de sus sprites el color negro se ha cambiado al rojo.
	Bloque 18, similar al bloque 1, pero en dos de sus sprites el color negro se ha cambiado al magenta.

Los bloques 1 al 7 son los principales del juego. Posteriormente definí los sprites de los elementos del juego: el personajes jugador y los elementos con los que interactúa. Finalmente para dar mas variedad al mapeado añadí dos nuevos bloques para los niveles de edad 4 y 5 que se diferencia en que tienen algunos pixeles de color rojo y magenta. Además, a esos dos últimos bloques no se le aplica el efecto flash cuando se cogen las uvas porque resultaría demasiado llamativo. (El efecto flash consiste en que el color del sprite y el color del fondo se van alternando.)

4.4.3. Gráficos del mapeado con los que se interactúa.



Estos gráficos están en el mapeado, pero no forman parte de las paredes y el jugador puede interactuar con ellos.

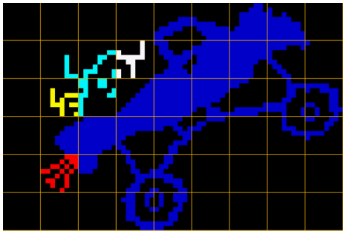

	<p>Bloque 8, coche meta.</p> <p>Es elemento que marca el final del nivel y para recalcar que es un elemento especial tiene efecto flash en sus atributos. Así sus colores rojo y azul se intercambian con el negro del fondo. Cuando se alcanza se pone en marcha auténticamente el final del nivel.</p>
	<p>Bloque 9, uvas.</p> <p>Al cogerlas desaparecen y a todos los bloques del mapeado se les aplica un efecto flash. Salvo el bloque 7 (pared falsa) que así se revela fácilmente y los bloques 17 y 18, que si se les aplicara el flash desentonaría mucho con el resto del mapeado.</p> <p>Me costo mucho hacerlas. En un principio intente copiar unas que venían en una captura en la Microhobby, pero no quedaba bien. Mi esposa me sugirió la forma triangular y de la Microhobby conserve el que la parte derecha este un pixel levantada.</p>
	<p>Bloque 10, chupete.</p> <p>Al cogerlo desaparece y aparece el hermanito para acompañar al personaje del juego.</p> <p>Me inspire en un dibujo de la invitación para el baby shower de mi hijo.</p>
	<p>Bloque 11, plátano.</p> <p>Al cogerlo se cambia el color de los atributos del mapeado. Así el jugador puede jugar a cambiarle el color al mapa, incluyendo al fondo.</p>
	<p>Bloque 12, pera.</p> <p>Al cogerlo desaparece y aparecen unas flores aleatoriamente por el mapeado.</p>
	<p>Bloques 13, 14, 15 y 16. Huevo frito.</p> <p>Es un teletransportador, lo que permite continuar el juego cuando frente al jugador hay una pared de bloques infranqueable. Se ha hecho de 4x4 bloques porque así podía hacerlo con 3 colores: el negro del fondo, el blanco de la clara y el amarillo de la yema.</p>

	<p>Bloque 19, seta.</p> <p>Para volver al inicio cuando se coge un teletransportador que te lleva a un camino cortado.</p>
	<p>Bloque 20, flores.</p> <p>Al coger la pera aparecen progresivamente durante 8 movimientos del personaje. Es pseudo aleatorio, pues sus posiciones son calculadas según la posición en los ejes X e Y del personaje. No hacen nada al cogerlas y no desaparecen. Se pueden traspasar porque sino bloquearía el camino del personaje.</p>  <p><i>Figura 8: Haciendo la flor a partir de una imagen de internet y reduciéndola para simplificarla.</i></p>

4.4.4. Sprites elementos del juego.

Estos son los únicos elementos móviles del juego:

	<p>Personaje.</p> <p>Basado en un dibujo de mi hijo.</p>
	<p>Monster truck.</p> <p>Es un sprite de 6x4 bloques (realmente 7x4, con sprites negros a la izquierda para borrar su propio rastro.)</p>

	<p>Monster truck con bebe.</p> <p>Es el sprite anterior pero rotado 30° y retocado. Se le ha añadido el hermanito y fuego rojo de un motor a reacción. Como se ha rotado es más grande que el anterior, 9x6 bloques.</p>
	<p>Bebe.</p> <p>Al coger el chupete aparecerá y acompañara al personaje.</p> <p>Diseño original que intenta transmitir que es un bebe gateando.</p>

4.5. Arquitectura del juego.

El juego se inicia en la pantalla de menú, donde se pregunta al jugador por su edad. Eso sirve de selector de dificultad. Al introducir una edad empezara el juego y el jugador recorrerá los niveles uno tras otro según el siguiente gráfico:

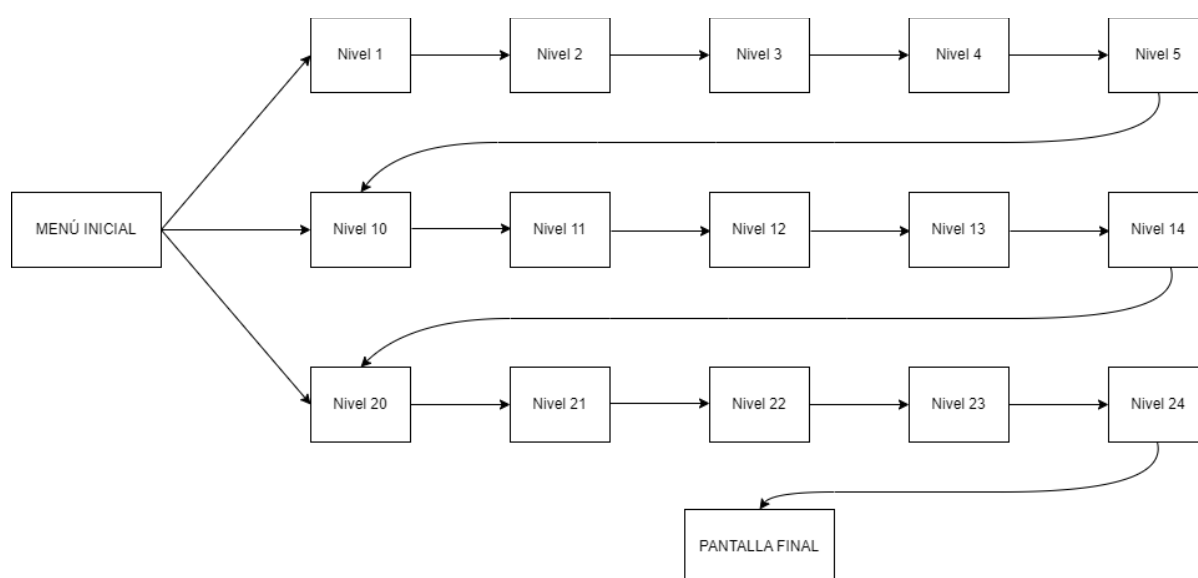


Figura 9: Pantallas y niveles del juego.

Quizás debería haber empezado con el nivel 0, pero así está más claro.

El juego funciona mediante bloques de 8x8 píxeles. Por eso el scroll no es suave. Algunos juegos de los ochenta usaban scroll a cada pixel o a cada dos o a cada cuatro (o un scroll de 8x8 para el fondo y a pixel para el personaje y enemigos.), pero yo únicamente encontré un código de mapeado y scroll mediante bloques de 8x8 píxeles. Trabajar por bloques hace que el programa sea más tosco, pero facilita mucho la programación porque la forma de trabajar del Spectrum en el modo de píxeles es bastante complicada. Pues divide la pantalla en tercios, líneas y columnas.

Este código de mapeado y scroll es primitivo. El principal problema es un parpadeo cada vez que el personaje se mueve. (Si se va rápido parpadea mucho.) La pantalla se dibuja enteramente en cada iteración del bucle principal del juego y luego encima se pinta el personaje. Para aumentar la velocidad considere implementar la técnica “*Adaptive tile refresh*”[45] en la que únicamente los elementos que cambiaran se redibujaran (el fondo negro no se redibujaría), pero me pareció que actualmente estaba fuera de mi alcance. La librería SP1 mencionada anteriormente usa esa técnica, pero sin scroll porque textualmente “sería algo bastante costoso”. [46]

Creo que en los ochenta había tres técnicas de dibujados de sprites principalmente. En este proyecto se ha usado la más sencilla. Otras técnicas que evitaban el problema del parpadeo era dibujar los sprites con “xor” y otra aún más avanzada era dibujar con máscaras. En los juegos exclusivos para los modelos de 128K se podía usar la memoria extra para crear una segunda representación de la pantalla. Esa técnica se llamaba “*shadow screen*”. Mientras una pantalla era mostrada, la otra se estaba dibujando y se alternaba con una frecuencia de 50Hz, la misma que la frecuencia de refresco de los televisores PAL. Así se evitaba el parpadeo.

4.6. Diseño de niveles y elementos del juego.

El juego se compone de 15 niveles:

Nivel 1 (edad 3 primer nivel) de 48x12 bloques (3 pantallas)

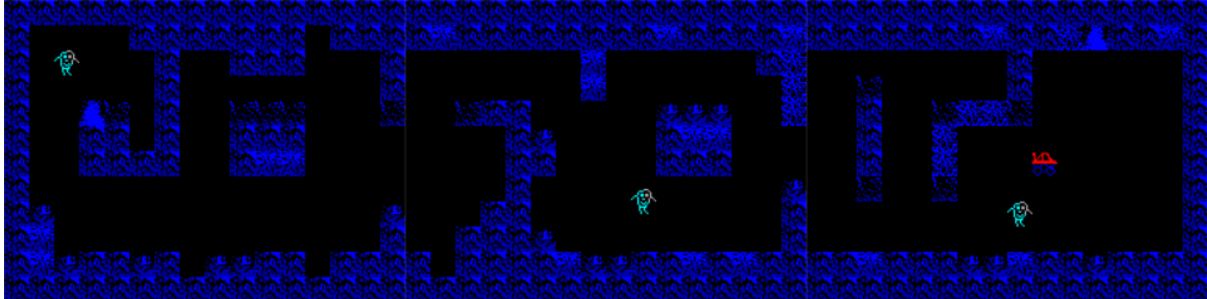


Figura 10: Captura de como se ve el nivel 1 completo

Este nivel (como el segundo y el tercero) está basado en el primer nivel del juego infantil que se muestra en el apartado 1.1. No tiene ninguna dificultad, pero su diseño es complejo, con dos bloques aislados que obligan a rodearlos y que crean bifurcaciones que complican el camino.

Nivel 2 (edad 3 segundo nivel) de 3 pantallas.

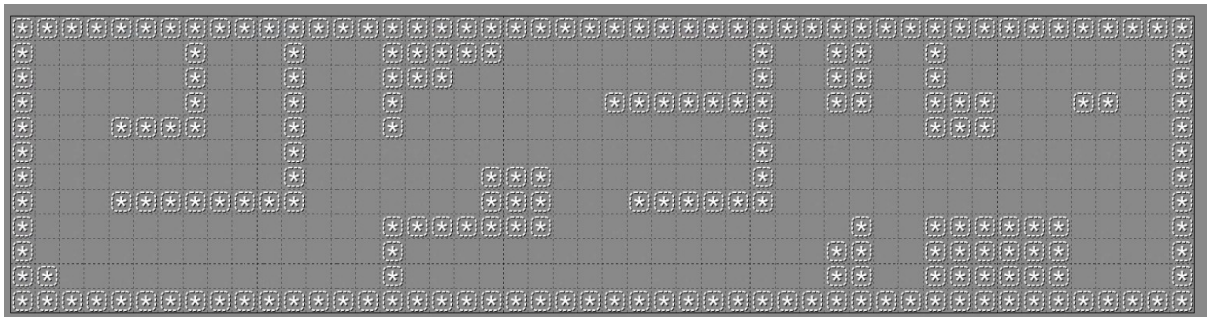


Figura 11: Nivel 2 realizado con tiled

Nada que comentar.

Nivel 3 (edad 3 tercer nivel) de 3 pantallas.

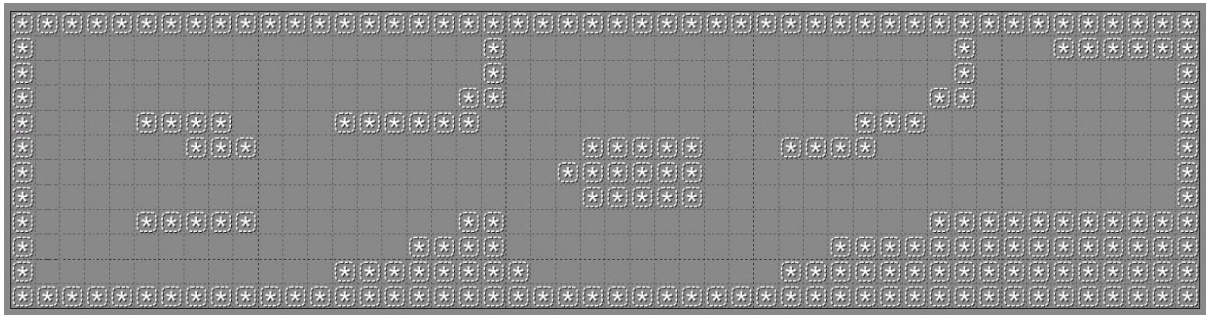


Figura 12: Nivel 3 realizado con tiled

Nada que comentar.

Nivel 4 (edad 3 cuarto nivel) de 3 pantallas.

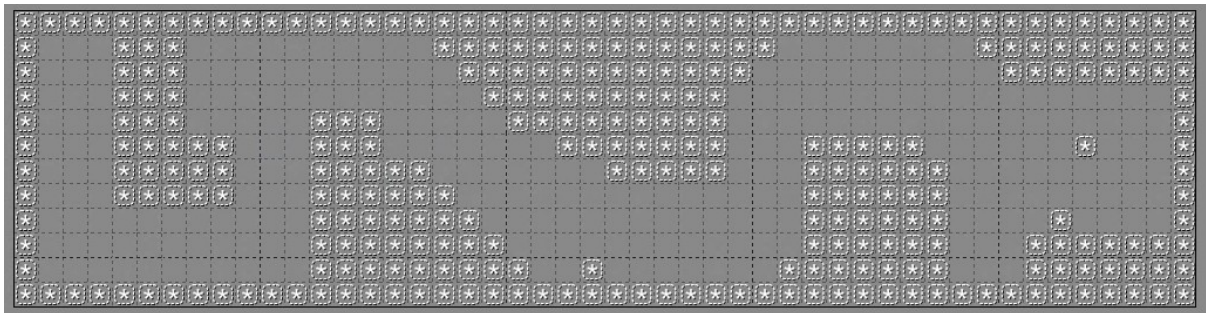


Figura 13: Nivel 4 realizado con tiled.

Primer nivel creado por mí sin basarme en el juego infantil. Es camino es muy sencillo y amplio. El proceso creativo ha sido muy simple, me imagine un túnel con subidas y bajas.

Nivel 5 (edad 3 quinto nivel) de 3 pantallas.

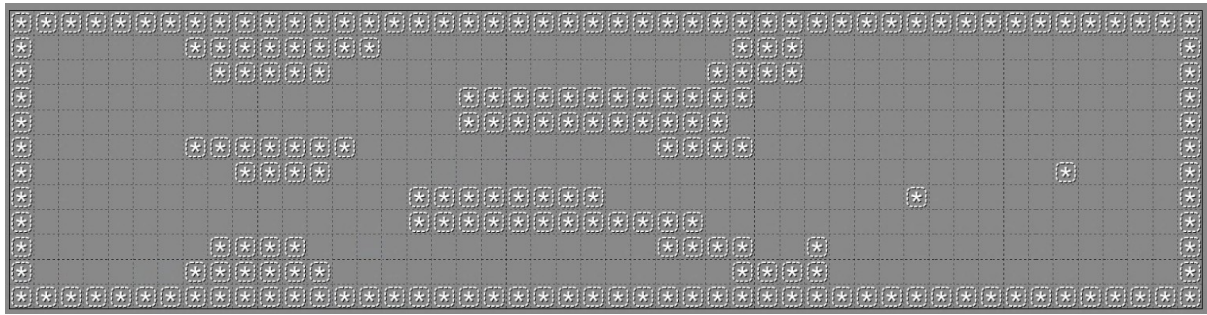


Figura 14: Nivel 5 realizado con tiled

Primer nivel con sorpresa. La estrellita aislada que está mas abajo se corresponde con el chupete. Cuando lo coja aparecerá el hermanito bebe:



Figura 15: Captura del final del nivel 5.

Nivel 10 (edad 4 primer nivel) de 60x12 bloques (3'75 pantallas).

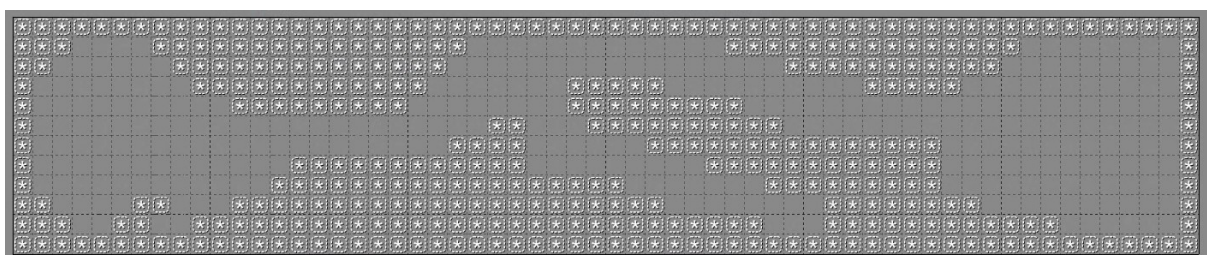


Figura 16: Nivel 10 realizado con tiled

En los niveles de edad 4 y 5 los caminos son más angostos. Como novedad de los niveles de edad 4, aparecerán algunos bloques con pixeles de color rojo o magenta. Aparte del tono extra de color, hay unas uvas que cuando se cogen le ponen efecto flash a todos los primeros bloques y además aumenta al máximo la velocidad del personaje. (Aunque en las

imágenes no se aprecia bien el efecto, en el juego el cambio es enorme.) En este nivel ese cambio únicamente es estético, pero en niveles más avanzados ayudara al jugador a descubrir una pared falsa (que no tiene efecto flash.).

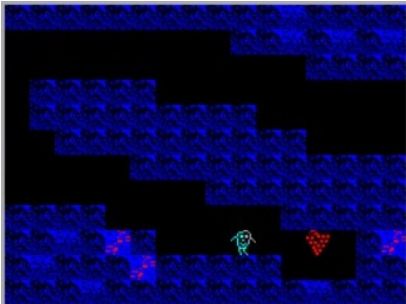


Figura 17: Personaje va a coger las uvas en nivel 10.

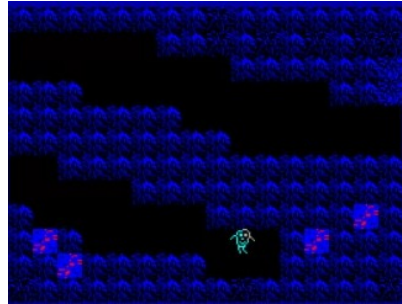


Figura 18: Tras cogerlas, los atributos de los colores se han invertido.

Nivel 11 (edad 4 segundo nivel) de 80x12 bloques (5 pantallas).

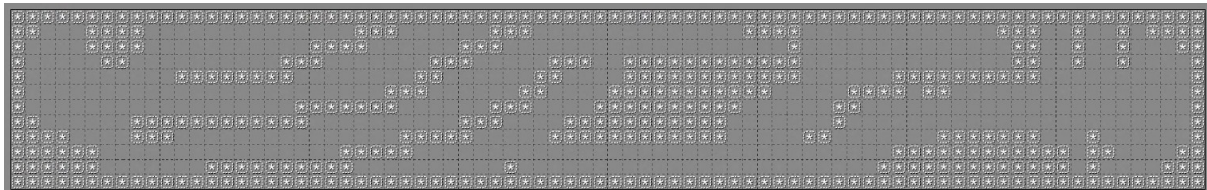


Figura 19: Nivel 11 realizado con tiled.

Se buscaba ofrecer algo más complicado así que el camino se complica con bifurcaciones dobles y triples que obligaran al jugador a dar la vuelta.

Nivel 12 (edad 4 tercer nivel) de 5 pantallas.

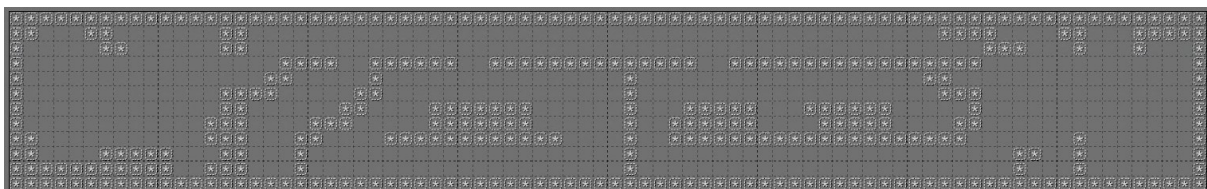


Figura 20: Nivel 12 realizado con tiled.

Los primeros caminos que se abren conducen a callejones sin salida.

Nivel 13 (edad 4 cuarto nivel) de 10 pantallas.

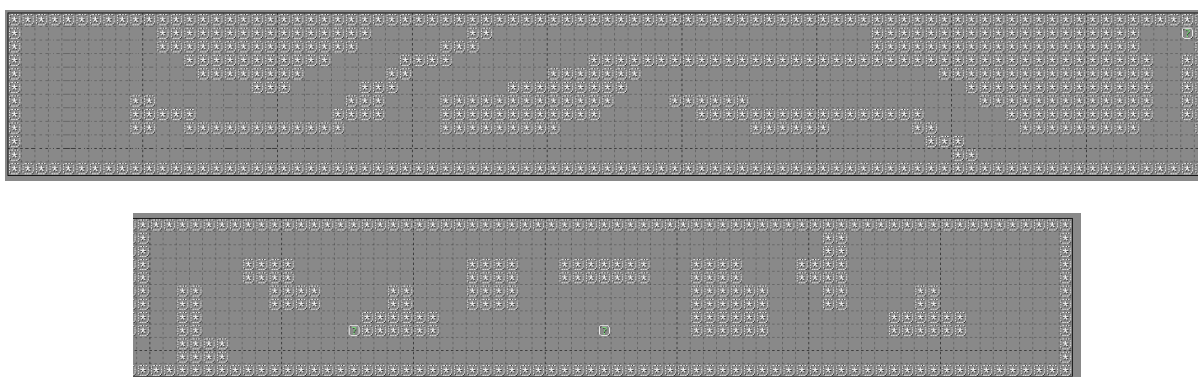


Figura 21: Nivel 13 realizado con tiled.

En la esquina superior derecha se encuentra un “?” que se corresponde con el plátano. Cuando se toca, éste no desaparece y cambiar el color de los atributos del mapeado. Como no desaparece se puede cambiar 128 veces de combinación de color de los atributos de papel y tinta, aparte de otras 128 veces con flash.

Centrado hay otro “?” que se corresponde con el chupete para que aparezca el hermanito. Este nivel empieza como un túnel, pero luego recordé como eran los niveles del juego bob y me acorde del tetris, así que en la segunda parte puse fichas del tetris. Hay dos unidas, en el juego se ve con diferente color.

Nivel 14 (edad 4 quinto nivel) de 10 pantallas.

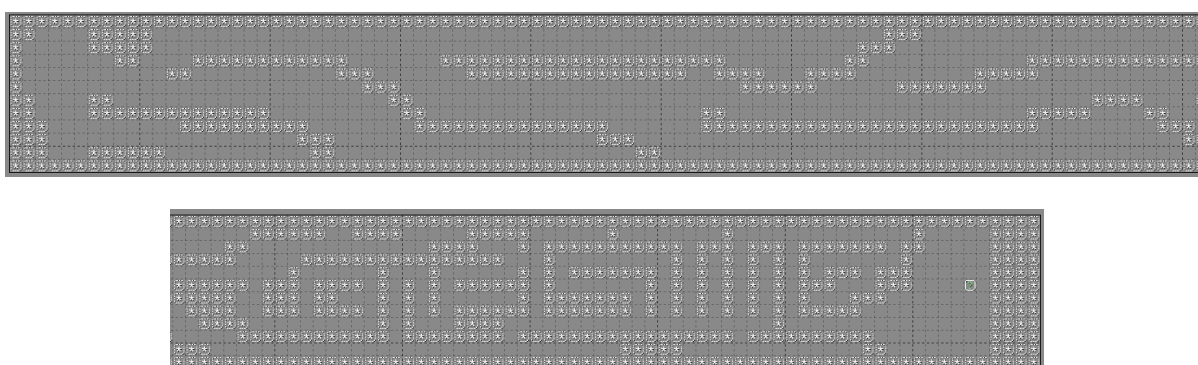


Figura 22: Nivel 14 realizado con Tiled.

Último nivel de edad 4, empieza con túneles normales y luego aparecen unos angostos túneles. Me inspire en la película Tron, adonde los corredores deben tomar rectas y ángulos rectos a toda velocidad. Hay dos posibles caminos hacia la meta.

Nivel 20 (edad 5 primer nivel) de 32x12 bloques (2 pantallas).

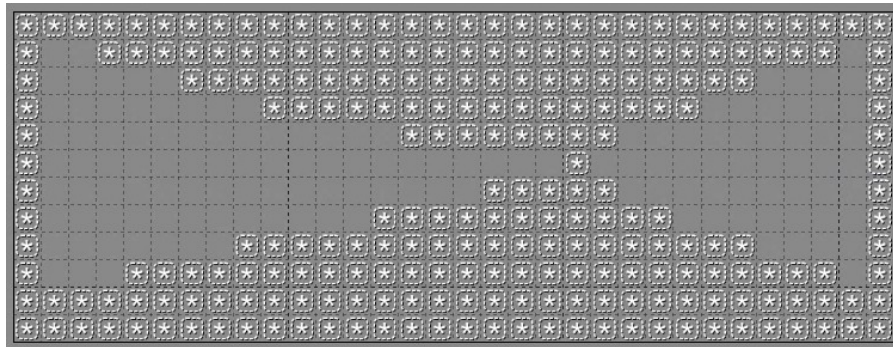


Figura 23: Nivel 20 realizado con tiled.

Aparece una pared falsa. Aunque estos retos siempre han sido discretos (Y solían provocar que el jugador se pusiera a golpear con la espada paredes sospechosas o usar magia de revelación) preferí poner un ejemplo muy evidente para que un niño pequeño (o un padre poco experimentado) lo descubriera.

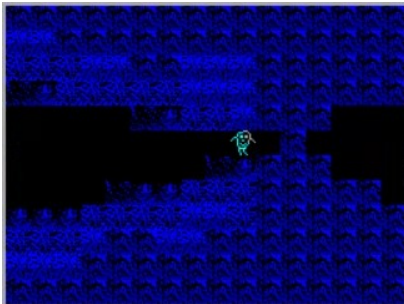


Figura 24: Personaje ante la pared falsa del nivel 20.

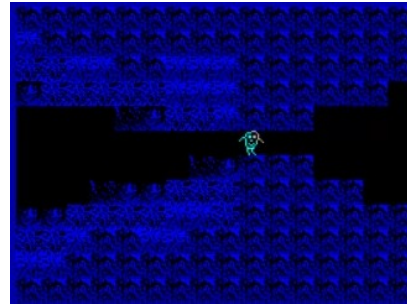


Figura 25: Tras tocar la pared falsa ya se puede seguir.

Nivel 21 (edad 5 segundo nivel) de 48x12 bloques (3 pantallas).

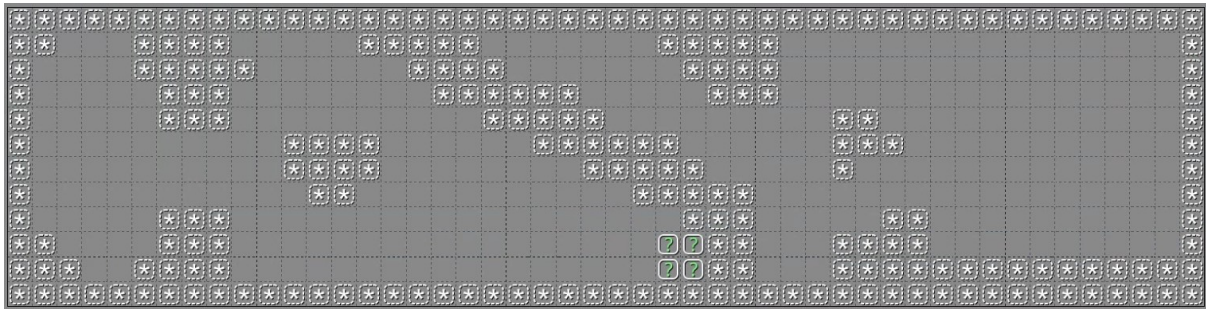


Figura 26: Nivel 21 realizado con tiled.

En este nivel aparece el huevo frito teletransportador (representado en los 4 bloques con “?”). Cuando el jugador lo toca se traslada a la parte superior lo que le permite culminar el nivel.

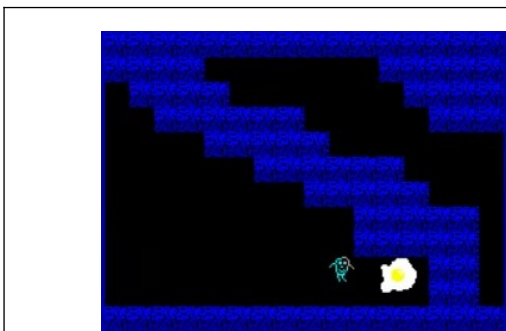


Figura 27: Personaje ante el gran huevo frito tricolor.



Figura 28: Tras tocarlo, aparece al otro lado del muro.

Nivel 22 (edad 5 tercer nivel) de 80x12 bloques (5 pantallas).

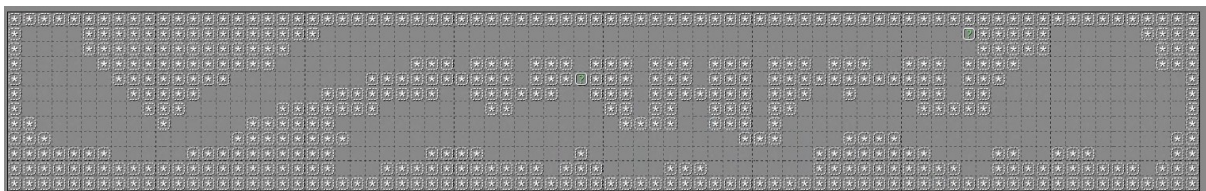


Figura 29: Nivel 22 realizado con tiled.

Durante el proceso creativo pensé: “¿cómo presentar el mayor reto?” Pues con el mayor número de puertas falsas posibles, aunque el jugador tendrá ayuda.

Así que hay 10 posibles paredes falsas, ¿pero como saber cual es la correcta sin tener que tocarlas todas? Las uvas (representada en el mapa con “?”) ponen efecto flash a todos los muros salvo a la pared falsa.) Una forma de aumentar la dificultad seria que la pared falsa fuera aleatoria y no estuviera siempre en la misma posición. Pero en el Spectrum es difícil conseguir números aleatorios. Se intento y no se consiguió.

Nivel 23 (edad 5 cuarto nivel) de 160x12 bloques (10 pantallas).

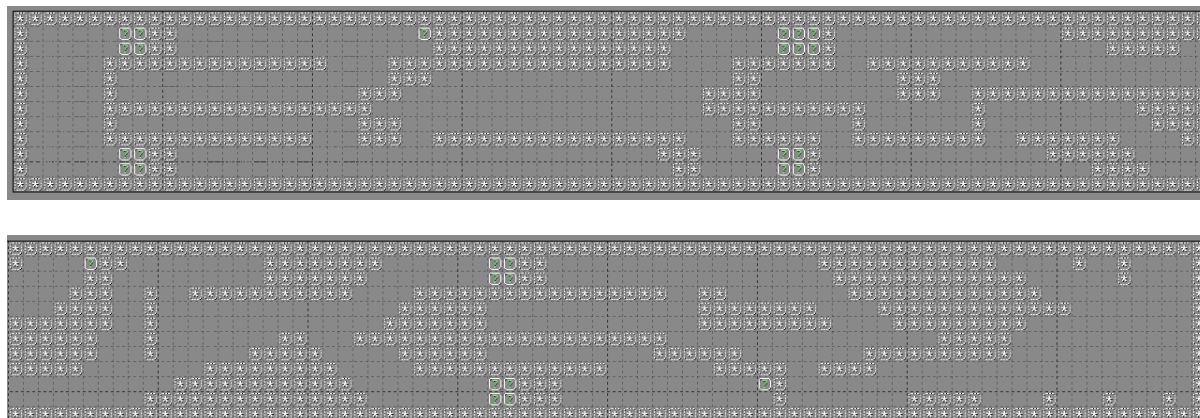


Figura 30: Nivel 23 realizado con tiled

Aquí el reto es la presencia de varios teletransportadores (representados por bloques de 4 “?”). La mayoría conduce a caminos sin salida, así que para que no haya que resetear la maquina, se ha puesto una nueva fruta (una seta) al final de esos caminos para volver al inicio.

Nivel 24 (edad 5 quinto nivel) de 80x36 bloques (15 pantallas).

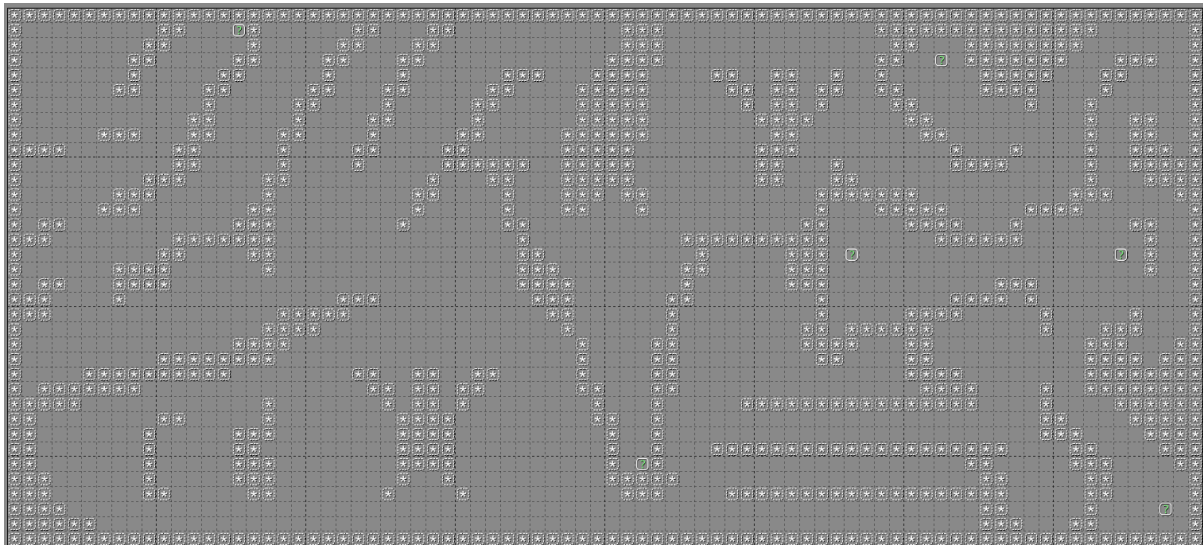


Figura 31: Nivel 24 realizado con tiled

El nivel final, se planea con la dificultad de el camino no sea solo hacia la derecha sino también hacia arriba y abajo.

Esta inspirado en el laberinto que se ve en la primera temporada de la serie WestWorld, incluso hay como dos personas. Hay multitud de frutas y dos posibles caminos para la meta, el "?" situado a la mitad del extremo derecho.

4.7. Mapa de colisiones

El mayor reto en este trabajo ha sido implementar el mapa de colisiones. Tiene dos funciones: impedir que el jugador se "coma" un muro (sino el juego sería simplemente avanzar hacia delante) y permitir que el jugador coja "comidas".

En las webs adonde me he documentado no había ninguna rutina para eso y he tenido que crearla por mi cuenta. Este ha sido el mayor reto en este trabajo. Tenia una vaga idea de como podía hacerse, haciendo un mapa de todo el nivel o de solo lo que se estuviera mostrando en ese momento en pantalla.

Usar el mismo mapa que uso para representar el nivel para esta tarea se me atojaba una tarea complicada. Una dificultad es que necesitaría un registro (o variable) para ir marcando

la posición y estas CPUs de 8 bits andas cortas de registros. (Incluso se tuvo la mayor dificultad de todo este trabajo porque durante un tiempo se uso el registro IY y este no se puede usar si se usan rutinas de la ROM, incluso las interrupciones pueden cambiarlo y provocar bugs.) Si usara una variable el problema seria de que las operaciones serian más lentas.

La segunda opción me pareció más sencilla, pues ya tengo una rutina que recorre la memoria seleccionando que mostrar y fue fácil insertar mi código para que creara un mapa con la representación de lo que se esta mostrando en pantalla.

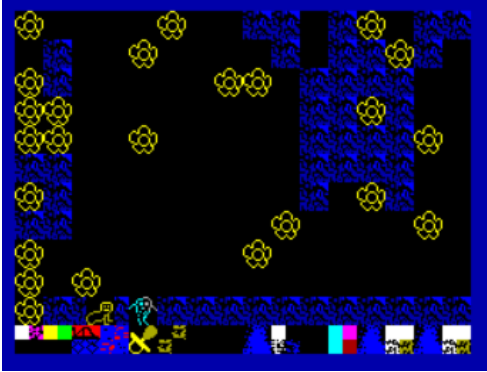
Una dificultad técnica es que mientras la pantalla tiene un tamaño de 32x24 bloques de 8x8 pixels, mi personaje es un sprite de 16x16, pero se mueve en saltos de 8. Así tenia un mapa de colisiones con unas dimensiones de 16x12, pero el personaje podría estar entre un bloque y un espacio libre. Así que tenia que detectar si estaba en una posición intermedia y en ese caso mirar dos veces en el mapa de colisiones.

4.8. Dificultades durante el desarrollo

Este trabajo detalla la construcción de un juego y para ayudar a cualquier desarrollador retro que lo lea, creo que es importante detallar algunas de las dificultades acontecidas y como se solventaron: (los primeros meses no fui tomando notas, pero hubo más dificultades)

Fecha	Descripción	Solución
22/04	<p>Gran problema: la pantalla del ZX Spectrum se divide en dos partes (que algunos llaman "canales"), la parte superior con 22 líneas y la línea de comandos inferior con 2 líneas.</p> <p>Al escribir en la pantalla sin especificar el canal se provocaban fallos gráficos graves. Lo curioso es que en el IDE se veía mal pero en un emulador (FUSE) se veía bien, por lo que llegue a pensar que se trataba de un bug del IDE.</p>	Al crear un nuevo proyecto en la versión actual del SpectNetIDE se escribe automáticamente un Hola Mundo y allí activaban el canal superior antes de escribir en él. Así descubrí como arreglarlo, activando el canal superior.
	Problemas con el joystick. kemspton, el personaje se iba para arriba.	Tras consultarlo con el grupo de Telegram revise mi código y lo corregí. También aprendí que el FUSE también da problemas porque no implementa bien el joystick, por eso deje de usar ese emulador. Empece a usar el RVM.
14/05	No se reseteaban bien los atributos al terminar un nivel.	Estaba usando una misma variable para dos cosas, cree una nueva variable para indicar que hay que resetear los atributos.

23/05	<p>La noche anterior había avanzado mucho, pero ésta fue horrible. Todo han sido problemas:</p> <ul style="list-style-type: none"> -Han salido bugs de bloques invisibles que no me dejan continuar cuando pensaban que eso ya lo había dejado solido. -Fracase al intentar poner un nuevo elemento en el menú de inicio para el joystick. Pero es que ademas el desactivas el joystick, que debería ser trivial, hace que el personaje se mueva solo. -Los 3 teletransportadores funcionan, pero no se puede continuar. 	<ul style="list-style-type: none"> -La noche siguiente hice debug del código y localice el bug, una línea que debía de haberse borrado. Es raro. -Desactive temporalmente el joystick. -Se ha cambiado la pareja de tres teletransportadores del nivel 24 por un trio de dos teletransportadores.
27/05	<p>Al aplicarse el efecto pera (llenar de flores la pantalla) se corrompen algunos gráficos e incluso se resetea la maquina.</p>	<p>Como en otras ocasiones en este trabajo, se peco de optimismo y el resultado fue que se escribía en zonas fuera de lo pensado.</p> <p>Tras muchos intentos variando unos parámetros conseguí un efecto bueno y que no daba problemas.</p> <p>Haciendo debug vi para mi sorpresa que mientras la variable que controla la posición del Y tiene un valor bajo (de 1 a 20), la del eje X tenia un valor muy rápido. Como era un registro de 16 bits lo arregle poniendo a cero el registro de 8 bits que ocasionaba el valor alto.</p> <p>Finalmente rediseñe la rutina para que nunca escribiera afuera de la zona de memoria del mapa del nivel.</p>

31/05	<p>Las flores del efecto pera aparecen por toda la pantalla y llegan a romper el juego. Como permiten pasar a través de ellas, si están en un muro permiten saltárselo. El problema es que también permiten saltarse el límite inferior del nivel e ir a zona de la memoria que no controlo.</p>  <p><i>Figura 32: Al principio las flores "rompían" el juego.</i></p>	<p>Antes de machacar el bloque con la flor, miro que es lo que hay y si es distinto de 0 (que representa el bloque vacío) la rutina vuelve sin machacarlo.</p> <p>Así hay menos flores, por que se ha aumentado a 8 el número de pasos en los que aparecen flores.</p>
2/06	<p>El juego tiene un fallo de diseño y es que es capaz de mostrar hasta 5 bloques de otro nivel si uno se va al extremo de la derecha sin coger el coche.</p> <p>Por eso el coche nunca está cerca del borde, pero en el nivel final uno puede pasar por la zona extremo derecha, ver esos otros caminos y no ser capaz de llegar.</p>	<p>Le añadido a cada nivel 5 bloques a la derecha. (No lo he añadido a los mapas de los niveles porque considero que no forman parte del mapeado y que normalmente ni se verían.)</p>

5/05	<p>Con todo hecho vamos a por la épica: el sintetizador de voz.</p> <p>Me ha costado mucho extraer los 2204 bytes del .tzx de la revista. Finalmente puedo ejecutarlo y haberle debug con el emulador ES.Pectrum. Durante hora y media he copiado la rutina que produce el sonido de la vocal "o", pero con el SpectNetIDE no se reproduce sonido.</p>	<p>Voy a hacer debug del código original y de mi copia comparando el valor de los registros a cada paso, a ver si descubro que pasa.</p> <p>Tras 3 horas he conseguido que mi copia emita el mismo sonido que el emulador. Los problemas eran dos:</p> <ul style="list-style-type: none"> -Muchos valores estaban en decimal y tenían que estar en hexadecimal. -Se cogía valores de la memoria que estaban asignados en el código original, pero no en mi copia.
09/05	<p>Sigo con la voz, el SpectNetIDE se comporta mal. Cuando le meto las rutinas de las dos primeras letras va bien, pero con la tercera en el debug no ejecuta las instrucciones que debe y se va a la posición inicial de la RAM, resetea el emulador.</p>	<p>Pensé que el SpectNetIDE se había corrompido. Lo volví a instalar en una maquina virtual pero el problema persistía. Es cosa del código de las rutinas.</p> <p>Tras mucho estudio encontré que el culpable era un CALL sin su RET (que había sustituido por un JP sin pensar en que eso dejaba un salto en la pila).</p> <p>Conseguí que pronunciara una palabra bien, con dos palabras se fastidiaba.</p> <p>Pero el problema fue que al integrarlo en el juego, cuando terminaba un nivel, en vez de ir al siguiente iba al menú de inicio.</p> <p>Desisto.</p>
10/5	<p>El SpectNetIDE permite añadir automáticamente pantallas de carga al juego cuando genera el archivo .tap, pero no me aceptaba el archivo .tap con la pantalla.</p>	<p>No sé la razón, pero no me aceptaba el .tap generado con el ZX-Paintbrush.</p> <p>Tuve que generar un archivo .scr y convertirlo a .tap con el scr2tap para que lo aceptara.</p>

5. Implementación

5.1. Requisitos de instalación

No es necesaria una instalación de la manera en la que normalmente se entiende. El emulador ZX Spin tiene muchos años, es del 2009 y sus requisitos de hardware son muy modestos. Se puede descargar de <https://www.zophar.net/sinclair/zx-spin.html>

Se descomprime el archivo comprimido .zip y se puede ejecutar directamente el ejecutable ZXSpin.exe

Por otro lado se puede meter el archivo .tap del juego en una tarjeta compact flash y cargarla en un spectrum real con el interface Divide.

5.2. Instrucciones de instalación

Es muy sencillo, únicamente abrir el emulador y arrastrar el archivo .tap encima. En el ZX Spin es muy fácil mapear el joystick Kempston a otras teclas. (Pulsando F8 y haciendo click en “*Controllers*” y luego en “*Customise...*”. Se pulsa la flecha que se quiere redefinir y luego la tecla que se quiere usar para esa dirección.)

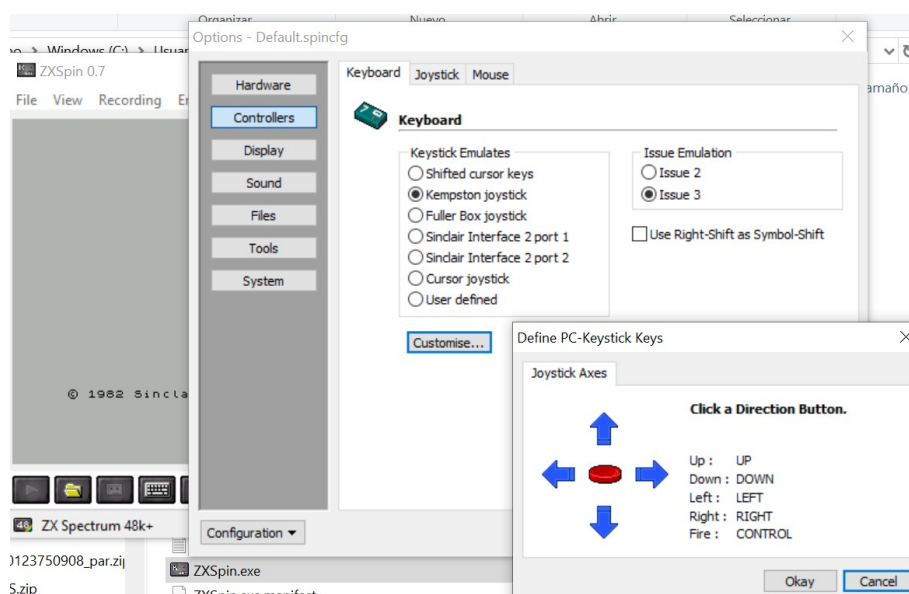


Figura 33: Configurando en el ZX el joystick Kempston

6. Demostración

6.1. Instrucciones de uso

Solo hay que introducir la edad ("3", "4" ó "5") en el menú inicial y ya se te lleva al juego.

Para mover al personaje los controles son los habituales:

- Q (Arriba)
- A (Abajo)
- O (Izquierda)
- P (Derecha)

6.2. Prototipos

El desarrollo del juego ha sido continuo desde el principio. Implementando en un principio las rutinas de mapeado con scroll para aprender como funciona y haciendo modificaciones sucesivamente para irle añadiendo nuevas funcionalidades y elementos. Por eso no ha habido prototipos.

Por otro lado anteriormente se ha hablado de todos los elementos del juego (mapas, sprites, diagrama de niveles,...) y creo que no habría nada nuevo que aportar en este apartado.

6.3. Tests

Se ha testado con los emuladores ZXSpin 0.7 y Retro Virtual Machine 2.1.11. También se ha testado con un ZX Spectrum 48k (gomas), ZX Spectrum+ y ZX Spectrum+2A 128K negro. Se ha testado con un joystick Sinclair emulando un Kempston mediante un interface.

Sobre los emuladores, decir que pocos son 100% compatibles. Por ejemplo, hablando del sintetizador de voz del Cobra's Arc dicen: "*why only with Spin and not with Real Spectrum, Spectaculator, Klive, X128, EmuZWin*".[47] Lo importante es que testado mi juego con la maquina real y allí funciona correctamente.

Un testeador ha sido un amigo informático con emulador, él ha estado testeando durante todo el desarrollo. Otro fueron un padre y un hijo con las maquinas reales. Aparte de bugs, la principal aportación fue que viendo al hijo jugar me sorprendió lo rápido que iba. Así que decidí meterle un retardo al juego para hacerlo más pausado (Se puede ir a toda velocidad cogiendo las uvas u otras frutas.)

Mi propio hijo también ha sido testeador con mi ZX Spectrum+. Le encanto y no quería dejar de jugar. Aprendí que él esperaba encontrar el bebe en todos los niveles, seguramente sea esa la primera mejora que le haga al juego. También que acerté con los muros de color azul, pues cuando podía cambiar de colores, jugaba a cambiarlos de color, pero a la hora de seguir jugando quería que los muros fueran de color azul.



Figura 34: Elías disfrutando del juego

Lamentablemente mi joystick Kempston no funciona y tuvo que jugarlo con las teclas QAOP.

Finalmente le pase el .tap y las instrucciones anteriores a otro amigo que no había tenido contacto con el proyecto previamente para hacerle un pequeño test. Su perfil es de un jugador adulto que empezó con las consolas de 16 bits. Solo se le paso las instrucciones que le mande a El SPECTRUMERO Javi Ortiz.

Se le envió el siguiente cuestionario y recibí las siguientes respuestas:

- ¿Cuál ha sido tu impresión general?
La impresión general ha sido muy buena mejor de lo que esperaba.
- ¿Cómo crees que le parecerá a un niño que nunca ha jugado a un videojuego?
Depende mucho del niño, hay niños que sentirán interés inmediatamente y otros no le llamara mucho la atención.
- ¿Qué es lo que más te ha gustado?
Los laberintos. Me ha gustado mucho tener que orientarme y buscar la salida era como un pequeño desafío.
- ¿Qué es lo que menos te ha gustado?
Que no haya música.
- ¿Se te ha resistido algún nivel?
El nivel final era el más difícil porque no se sabe que hay justo abajo. El mapa es mas grande y es más difícil intuir el camino.
- ¿Que nivel te ha gustado más?
Había un nivel donde parecía no haber salida pero uno de los muros era un muro falso y ahí estaba la solución.
- ¿Habrías añadido algo más?
Me hubiera gustado oír alguna melodía o algo de música.
- ¿Quieres comentar alguna cosa más? ¿Algo que mejorar?
Aparte de lo de la música felicidades por este gran juego. Espero que Juan y Elías lo redescubran de mayores! seguro que les hará mucha gracia e ilusión.

La valoración que hago es que he hecho bien mi trabajo, he hecho un buen juego. Destacaría dos cosas: sobre la música, él creció con los 16 bits, cuando la música ya era parte de los juegos. Con los 8 bits la música solía limitarse a los choques o explosiones y el único juego de Spectrum 48 que conozco con música durante el juego era el Manic Miner. (Como la música era generada por la propia CPU, mientras la música suena la ejecución del juego se para.)

También destaco como supo superar el nivel del muro falso. (Aunque parece olvidar que en el primer nivel de edad 5 ya se encontró con un muro falso muy evidente.)

7. Conclusiones y líneas de futuro

7.1. Conclusiones

Lo que más he aprendido de este trabajo es que se debe pensar cuidadosamente una rutina (o incluso programar en papel como en aquella época) antes de codificarla. El debug ha sido fundamental para detectar los errores y poder construir unas rutinas solidas.

El tener que programar en ensamblador conlleva tener que usar un número limitado de registros (No como en los lenguajes de alto nivel adonde se puede usar una infinidad de variables). Muchas veces me ha sorprendido ver en un registro un valor que no esperaba.

Otra conclusión es que el ensamblador permite hacer algunas cosas muy fácilmente, mientras que otras (los sprites por software) son muy complicadas. En ese sentido se ve que para algunas cosas el ensamblador es muy potente y se podían hacer maravillas para los ordenadores de 8 bits.

Hemos conseguido todos los objetivos generales descritos en el apartado 1.3. Especialmente las tareas de programación nocturna han sido muy satisfactorias y me han quitado algunas décadas.

Más técnicamente hay una cosa que me gustaría haber conseguido y no pude. Conseguir la aleatoriedad, que es muy complicada de conseguir en el Spectrum. Probé con llamadas a la ROM, leer de los registros "I" y "R" (que según leí se pueden usar para obtener números aleatorios), busque rutinas,... Al final para conseguir la aleatoriedad del efecto pera (es decir, que aparezcan flores por la pantalla) use la posición del personaje tras coger la pera. Así si el personaje se mueve en una dirección distinta, durante los 8 movimientos en los que van apareciendo flores, estas aparecerán en posiciones distintas.

Se ha seguido la planificación, pero reseñaría que empece dos meses antes. Hubo noches en que avance poco, pero si no hubiera empezado al menos con un mes de antelación no me hubiera sido posible hacer el juego completo

7.2. Líneas de futuro

Hay varias cosas que considero que se pueden mejorar:

- Por demanda de mi hijo, le meteré el chupete para que salga el hermanito en casi todos los niveles.
- Creo que el salto más grande sería contar con un gráfista para hacer los gráficos de los muros y darle más variedad. Aparte se podrían dar diferentes temáticas a cada grupo de niveles.
- Otra mejora importante sería cambiar la forma de pintar al personaje en la pantalla, pasando a imprimir con XOR. Así se eliminaría el parpadeo.
- El juego no tiene enemigos. Se podría añadir un contador de tiempo para aumentar la dificultad o añadir enemigos móviles. Lo primero sería fácil de implementar y lo segundo me parece mucho más difícil. El juego se mueve conforme se mueve el personaje, cambiar eso a que haya enemigos que se muevan todo el tiempo es complicado. Seguramente hubiera que utilizar las interrupciones del procesador.
- Cuando publicite el juego en el grupo de Telegram preguntare a alguien tiene experiencia con las rutinas del sintetizador de voz de la Microhobby. Quizás con ayuda pueda finalmente meterle la voz.

Una sensación de desasosiego que me deja este proyecto es que se puede ser considerado por algunos como un juego simplón. Yo lo hice así para mis hijos. Pero si volviera a hacer otro juego lo haría con lo que se considera más estándar, con enemigos y elementos móviles. Habría que usar las interrupciones de la CPU para hacer que los enemigos se muevan y no depender de que el personaje se mueva.

Pero si volviera a hacer un juego lo primero sería contar con un grafista. A mí me gusta programar, los gráficos mejor que lo haga otra persona. Me gustaría meterle más colorido. Sería cuestión de aprender a manejar el BIFROST* Engine o el NIRVANA Engine.

Y ya por fantasear, un scroll más suave... Música en el juego... eso ya no es Spectrum 48k (Aparte de que necesitaría a alguien me he hiciera la música, porque de eso voy peor que de grafista.)

Bibliografía

En este documento se ha seguido la norma UNE-ISO 690:2013.

[1] Cómo se hace un juego. Ogerox. Microhobby 171 página 58. [en línea] [fecha de consulta: 16 de abril de 2024]. Disponible en: https://microhobby.speccy.cz/mhf/171/MH171_58.jpg

[2] ZX80 [en línea] [fecha de consulta: 16 de abril de 2024]. Disponible en: <https://en.wikipedia.org/wiki/ZX80>

[3] ZX81 [en línea] [fecha de consulta: 16 de abril de 2024]. Disponible en: <https://en.wikipedia.org/wiki/ZX81>

[4] ZX Spectrum [en línea] [fecha de consulta: 16 de abril de 2024]. Disponible en: https://en.wikipedia.org/wiki/ZX_Spectrum

[5] 16K ZX81 vs 16K ZX Spectrum [en línea] [fecha de consulta: 16 de abril de 2024]. Disponible en: <https://www.youtube.com/watch?v=mL0K5u1zkko>

[6] Edad de oro del software español [en línea] [fecha de consulta: 16 de abril de 2024]. Disponible en: https://videojuegos.fandom.com/es/wiki/Edad_de_oro_del_software_espa%C3%B1ol#:~:text=La%20llamada%20Edad%20de%20oro,Spectrum%2C%20detr%C3%A1s%20de%20Reino%20Unido.

[7] divide [en línea] [fecha de consulta: 16 de abril de 2024]. Disponible en: <https://divide.speccy.cz/>

[8] Estimulación cognitiva para niños - Laberintos para imprimir. [en línea] [fecha de consulta: 3 de mayo de 2024]. Disponible en: <https://www.edufichas.com/estimulacion-cognitiva/laberintos/>

[9] Laberintos [en línea] [fecha de consulta: 4 de mayo de 2024]. Disponible en: <http://afantdah.org/index.php/laberintos>

[10] Segura Duran, Juan. *Boriel Basic para ZX Spectrum: Manual para torpes...y para los que no lo son tanto*. Autopublicado en Amazon. 2023. ISBN-13. 979-8871546543.

[11] An Introduction to Z88DK [en línea] [fecha de consulta: 16 de abril de 2024]. Disponible en: <https://github.com/z88dk/z88dk/wiki>

[12] ANN: SP1 for Z88DK (aka splib3, Sprite Pack v3.0) [en línea] [fecha de consulta: 15 de mayo de 2024]. Disponible en: <https://worldofspectrum.org/forums/discussion/11729/redirect/p1>

[13] Taller de ensamblador para ZX Spectrum 16K – Pong ►► Sesión 1 - Hola Mundo [en línea] [fecha de consulta: 16 de abril de 2024]. Disponible en: <https://www.youtube.com/watch?v=sH3Qhoj8Cjg>

[14] Hazte tu propio «Filiation» [en línea] [fecha de consulta: 16 de abril de 2024]. Disponible en: <https://programbytes48k.wordpress.com/2011/05/05/hazte-tu-propio-filiation/>

[15] ¡Liberado! Mojon Twins Engine MK1 v5.0 (La Churrera) [en línea] [fecha de consulta: 20 de abril de 2024]. Disponible en: <https://www.elmundodelspectrum.com/liberado-mojon-twins-engine-mk1-v5-0-la-churrera/>

[16] BIFROST* Engine [en línea] [fecha de consulta: 20 de abril de 2024]. Disponible en: https://sinclair.wiki.zxnet.co.uk/wiki/BIFROST*_Engine

[17] NIRVANA Engine [en línea] [fecha de consulta: 20 de abril de 2024]. Disponible en: https://sinclair.wiki.zxnet.co.uk/wiki/NIRVANA_Engine

[18] Gandalf [en línea] [fecha de consulta: 20 de abril de 2024]. Disponible en: <https://noentiendo.itch.io/gandalf>

[19] Metamorphosis Final game [en línea] [fecha de consulta: 20 de abril de 2024]. Disponible en: <https://itnl-team.itch.io/metamorphosis>

[20] ZX Spectrum Game Maker, crea tu juego para Spectrum sin conocimientos técnicos [en línea] [fecha de consulta: 20 de abril de 2024]. Disponible en: <https://www.elmundodelspectrum.com/zx-spectrum-game-maker-crea-tu-juego-para-spectrum-sin-conocimientos-tecnicos/>

[21] Cesare the somnambule [en línea] [fecha de consulta: 20 de abril de 2024]. Disponible en: <https://zxamaze.itch.io/cesare-the-somnambule>

[22] Enlace de invitación al grupo de telegram Ensamblador Z80 (ZX Spectrum, Amstrad, MSX y más): <https://t.me/EnsambladorZXSpectrum>

[23] Bitfans [en línea] [fecha de consulta: 20 de abril de 2024]. Disponible en: <https://bitfans.itch.io/>

[24] Travel Through Time Volume 1: Northern Lights [en línea] [fecha de consulta: 15 de mayo de 2024]. Disponible en: <https://www.zosya.net/2021/06/24/travel-through-time-volume-1-northern-lights/>

[25] ¡Nuevo! Delta's Shadow de Sanchez [en línea] [fecha de consulta: 15 de mayo de 2024]. Disponible en: <https://www.elmundodelspectrum.com/nuevo-deltas-shadow-de-sanchez/>

[26] Yandex Retro Games Battle v3 [en línea] [fecha de consulta: 15 de mayo de 2024]. Disponible en: <https://rgb.yandex.ru/>

[27] Amstrad CPC 6128 en El Precio Justo [en línea] [fecha de consulta: 20 de abril de 2024]. Disponible en: <https://www.youtube.com/watch?v=g8hED2JS094>

[28] Los jóvenes no comprenden cómo funciona el sistema de archivos: "¿Carpetas? ¿Unidades? ¿De qué me estás hablando?" [en línea] [fecha de consulta: 20 de abril de 2024]. Disponible en: <https://www.genbeta.com/almacenamiento/jovenes-no-comprenden-como-funciona-sistema-ficheros-carpetas-unidades-que-me-estas-hablando>

[29] El SPECTRUMERO Javi Ortiz [en línea] [fecha de consulta: 20 de abril de 2024]. Disponible en: <https://www.youtube.com/@ElSpectrumeroJaviOrtiz>

[30] El mundo del Spectrum [en línea] [fecha de consulta: 20 de abril de 2024]. Disponible en: <https://www.elmundodelspectrum.com/>

[31] Dotneteer / kliveide [en línea] [fecha de consulta: 4 de mayo de 2024]. Disponible en: <https://github.com/Dotneteer/kliveide>

[32] Advanced Math [en línea] [fecha de consulta: 4 de mayo de 2024]. Disponible en: <http://z80-heaven.wikidot.com/advanced-math>

[33] Felix [en línea] [fecha de consulta: 4 de mayo de 2024]. Disponible en: <https://github.com/adigostin/felix/issues/1>

[34] SpectNet IDE. Visual Studio 2017/2019 integrated ZX Spectrum IDE for the Community [en línea] [fecha de consulta: 30 de abril de 2024]. Disponible en: <https://dotneteer.github.io/spectnetide/>

[35] Tiled [en línea] [fecha de consulta: 15 de mayo de 2024]. Disponible en: <https://www.mapeditor.org/>

[36] SevenUP [en línea] [fecha de consulta: 17 de mayo de 2024]. Disponible en: <https://metalbrain.speccy.org/>

[37] ZX-Paintbrush [en línea] [fecha de consulta: 15 de mayo de 2024]. Disponible en: <https://sourcesolutions.itch.io/zx-paintbrush>

[38] ZXSpin [en línea] [fecha de consulta: 15 de mayo de 2024]. Disponible en: <https://www.zophar.net/sinclair/zx-spin.html>

[39] Retro Virtual Machine [en línea] [fecha de consulta: 15 de mayo de 2024]. Disponible en: <https://www.retrovirtualmachine.org/>

[40] ES.spectrum [en línea] [fecha de consulta: 11 de junio de 2024]. Disponible en: <https://www.habisoft.com/spectrum/>

[41] juananZX / scr2tap [en línea] [fecha de consulta: 14 de junio de 2024]. Disponible en: <https://github.com/juananZX/scr2tap>

[42] paint .net [en línea] [fecha de consulta: 14 de junio de 2024]. Disponible en: <https://www.getpaint.net>

[43] Curso de Ensamblador Z80 de Compiler Software [en línea] [fecha de consulta: 17 de mayo de 2024]. Disponible en: <https://wiki.speccy.org/cursos/ensamblador/indice>

[44] ESPAMÁTICA [en línea] [fecha de consulta: 17 de mayo de 2024]. Disponible en: <https://espamatica.com/>

[45] Adaptive tile refresh [en línea] [fecha de consulta: 30 de abril de 2024]. Disponible en: https://en.wikipedia.org/wiki/Adaptive_tile_refresh

[46] Z88DK y SPLIB (SP1) [en línea] [fecha de consulta: 15 de mayo de 2024]. Disponible en: [https://wiki.speccy.org/cursos/z88dk/splib?s\[\]=colisi%C3%B3n](https://wiki.speccy.org/cursos/z88dk/splib?s[]=colisi%C3%B3n)

[47] No speech in Cobra's Arc? A mystery tale from pain... [en línea] [fecha de consulta: 11 de junio de 2024]. Disponible en: <https://worldofspectrum.org/forums/discussion/4243/>