OVERHEAP

A videogame for Amstrad CPC 464

Making Of

LA IDEA

Desarrollado en gran parte en assembler Z80 y el Amstrad CPC 464 se ha emulado con Winape. Para programarlo, nos hemos apoyado en la CPCtelera, que nos ha facilitado muchísimo el desarrollo.

El desarrollo surge de la asignatura “Razonamiento Automático”, impartida en el grado de Ign. Informática, Universidad de Alicante. Los tres integrantes somos alumnos de la asignatura comentada.

Al empezar no teníamos una imagen clara del videojuego, aunque sí teníamos en mente hacer algo donde nuestro personaje pudiese disparar y además queríamos incluir scroll por hardware. Un día probando juegos de anteriores entregas del CPCretrodev y lanzamientos bastante recientes dimos con el juego Galactic Tomb. Ya con nuestro juego de referencia, decidimos adaptar Galactic Tomb, pero, en nuestro caso sólo tendríamos scroll horizontal dada la limitación de tiempo y conocimiento, donde según avanzas por los niveles del mapa, tienes que acabar con los enemigos.

PROBLEMAS ENCONTRADOS

Uno de los mayores problemas encontrados han sido las limitaciones en memoria y capacidad de cómputo del Amstrad CPC 464. Acostumbrados a usar ordenadores actuales con memoria y potencia de sobra, hasta que no empezamos a hacer pruebas con el ensamblador Z80 y escribir directamente en memoria nuestros primeros programas, no éramos del todo conscientes que tan grandes son las limitaciones hardware. Incluso hasta no tener gran parte de las mecánicas programadas e incluirlas en un prototipo del juego, no sabíamos realmente la cantidad de enemigos, mapas, y funcionalidades que podíamos usar.

Hacer scroll por hardware limitado y doble buffer para evitar parpadeos, nos ha hecho usar unos 35 KB del total de la memoria y a esto le sumamos, que al final no pudimos comprimir los niveles por falta de tiempo. Por ello hemos tenido que pensar en formas de reducir la memoria, como limitar la cantidad de sprites con transparencia, hacer un único sprite para los enemigos en vez de usar uno por cada dirección a la que mira, hacer un sistema de rondas en un mismo mapa para ahorrar memoria...

LECCIONES APRENDIDAS

Como conclusión, tras dedicarle muchas horas a entender cómo funciona el Amstrad CPC y al programar casi todo el juego en Z80, hemos adquirido mucho conocimiento acerca de cómo funciona un ordenador, más agilidad con el ensamblador y un mejor manejo del código cuando este se empieza a hacer grande. Todo ello podremos aplicarlo en futuros proyectos.

Por otro lado, el tener que entregar un producto al concurso y ponernos unos plazos de desarrollo y entrega, hemos podido comprender lo que significa tener que hacer un producto real y todo lo que conlleva conseguir que este sea usable el dia de la entrega a costa de reducir funcionalidades u otros aspectos.

CAPTURAS DURANTE EL DESARROLLO

A continuación dejamos una serie de capturas que hemos ido haciendo durante el desarrollo
En nuestra cuenta de twitter, @BastaCpc, hemos subido gifs que hemos hecho durante el desarrollo del juego. A continuación dejamos unas imágenes y el resultado final:

Imagen1. Intentando redibujar el mapa mientras una entidad se mueve.

Imagen 2. Primer prototipo del sistema de disparos funcionando

Imagen 3. Probando el scroll hardware con un ejemplo de la CPCtelera

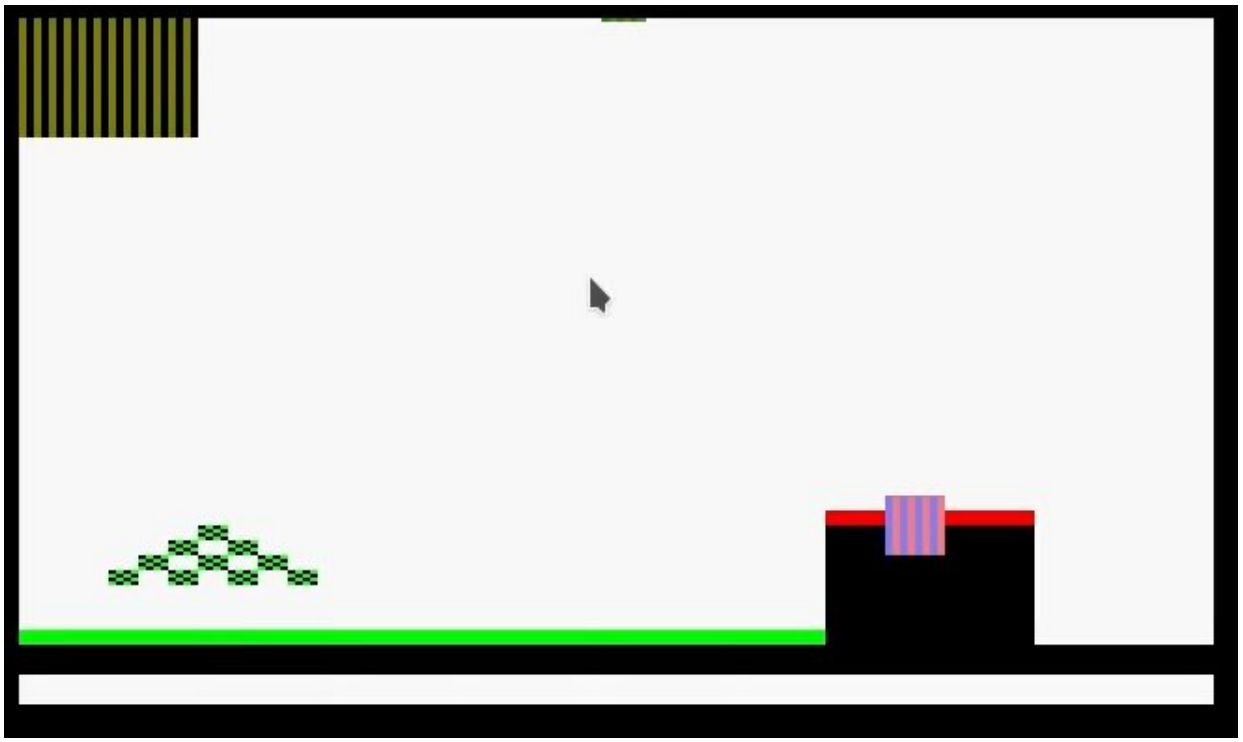

Imagen 4. Colisión de una entidad (cuadrado) con el mapa (rojo)

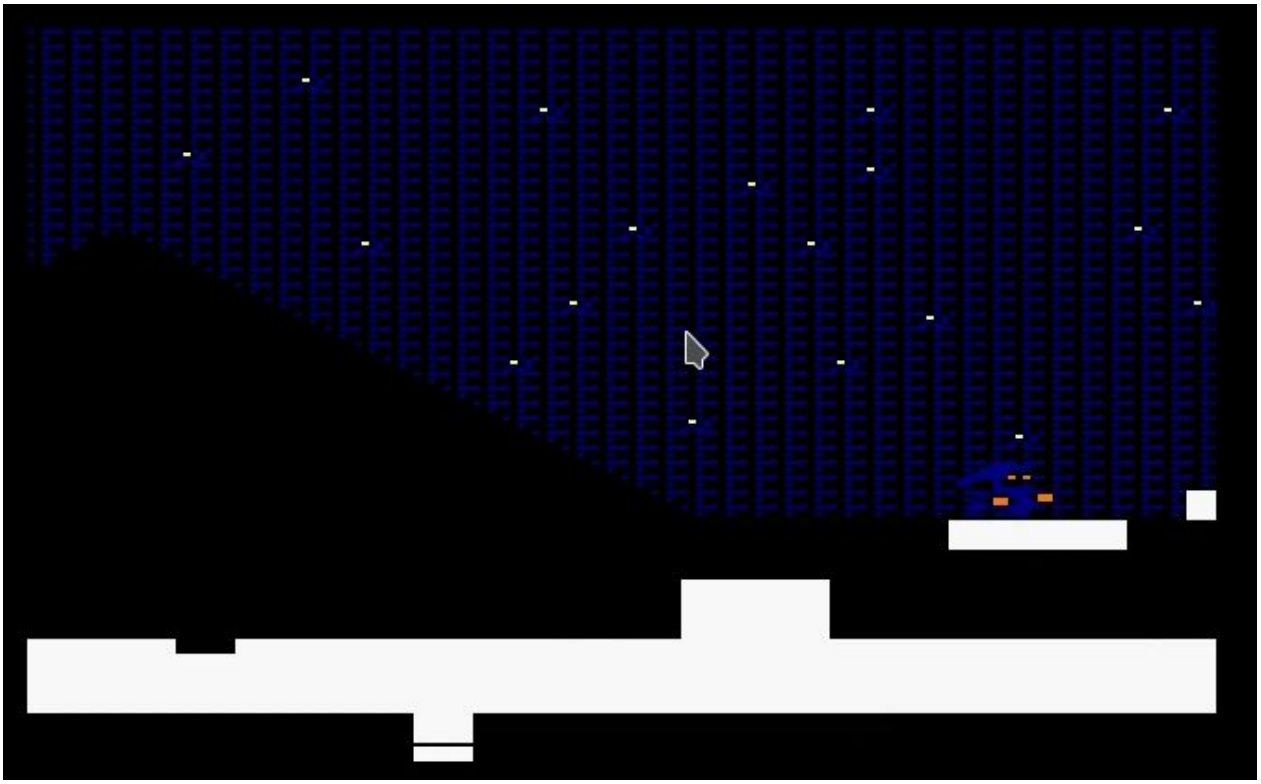

Imagen 5. Sprite del hero + Scroll Hardware + Mejoras en las colisiones

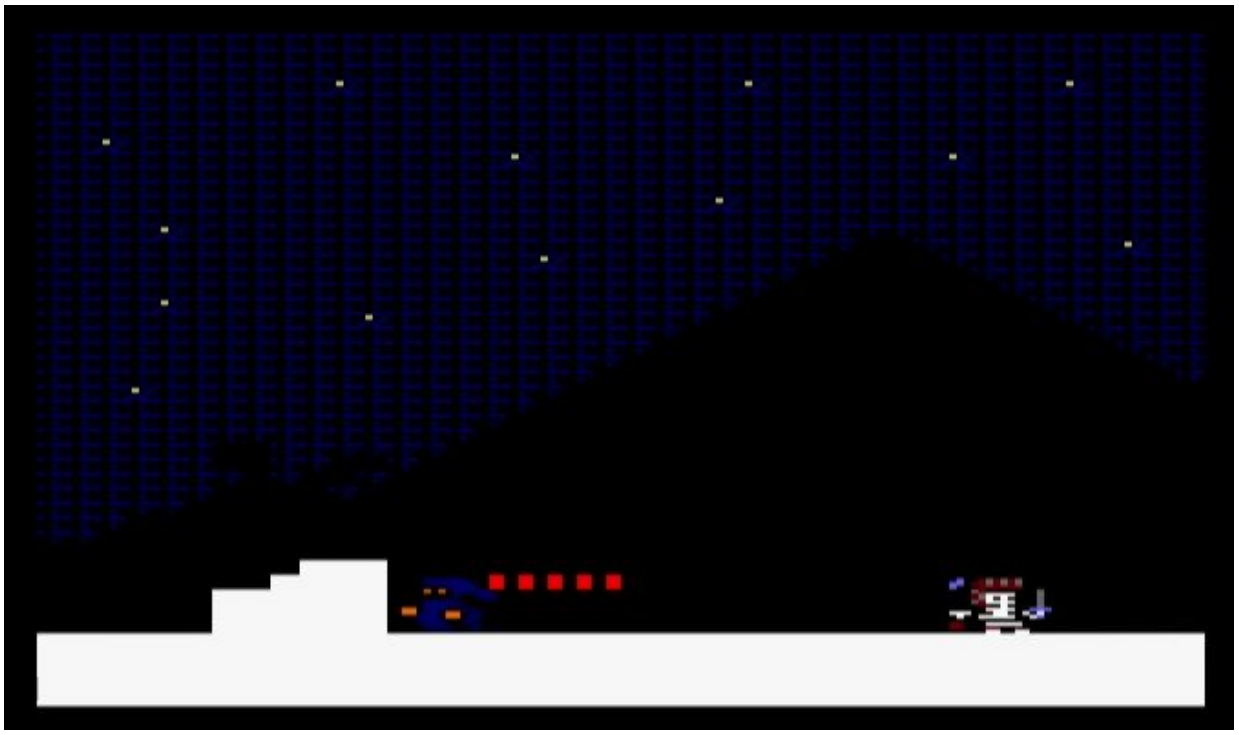

Imagen 6. Sistema de disparos

Imagen 7. Resultado final del juego

Herramientas utilizadas

- **Gimp** - para el tileset de los gráficos
- **Tiled** - para hacer los mapas y sprites
- **Arkos Tracker -** para la música del menú
- **Visual Studio Code** - para la programación en ASM
- **CDT2WAV** - para cargar el juego en un Amstrad CPC
- **CPCTelera -** Game Engine para Amstrad CPC
- **Wine** - para emular Winape en Linux
- **Manjaro** - para el sistema operativo durante el desarrollo
- **Google Docs** - Usados para hacer la documentación
- **Trello** - Organizar las tareas
- **GitHub** - Repositorio de código

Créditos

Juan López Quiles	jlq2@alu.ua.es https://github.com/psjuan97
José Luis Gómez Antón	Jlga10@alu.ua.es
Alejandro Aliaga Hyder	ajah1@alu.ua.es https://github.com/alihyder1