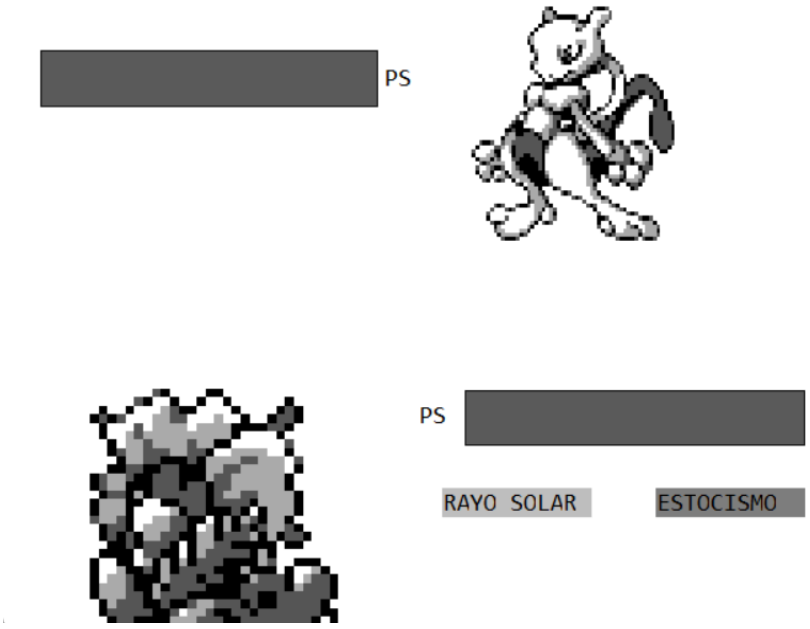
Universitat de les Illes Balears

Pokémon battle Simulator

Easy68K



Autores y DNI:

Vicent Roig : 48199890D
Alejandro Martínez Martínez: 48230272P
Hugo López García: 47432182H

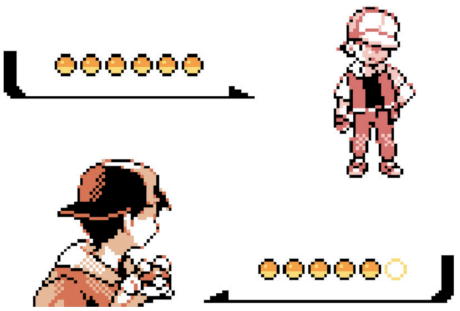
VUELTA A LOS ORÍGENES

Los primeros juegos de Pokémon, Pokémon Rojo y Pokémon Verde, fueron desarrollados por Game Freak y lanzados en Japón el 27 de febrero de 1996 para la consola portátil Game Boy. Concebidos por Satoshi Tajiri, su diseño combinaba su amor por coleccionar insectos con su fascinación por los videojuegos, dando lugar a una experiencia única que invitaba a los jugadores a capturar, entrenar y combatir con criaturas conocidas como Pokémon.

El objetivo del juego es convertirse en el Campeón de la Liga Pokémon derrotando a los entrenadores más fuertes y completando la Pokédex, un registro de las 151 criaturas originales. Estos títulos sentaron las bases del fenómeno mundial, introduciendo mecánicas como el intercambio de Pokémon entre versiones utilizando el Game Link Cable, promoviendo la cooperación y la interacción social entre jugadores.

Aunque inicialmente tuvieron un lanzamiento limitado, la creciente popularidad del juego se transformó rápidamente en un fenómeno cultural, extendiéndose a nivel global con las ediciones Pokémon Rojo y Azul en 1998 para Norteamérica, consolidando a Pokémon como una de las franquicias más exitosas en la historia de los videojuegos.

Hecha esta introducción, nuestro juego titulado (***Pokémon Battle Simulator***) recoge una de las características fundamentales que le dieron tanta fama a estos juegos, que es el combate Pokémon. Los rasgos que caracterizan a este simulador de batallas se explican en las siguientes páginas.



INTRODUCCIÓN AL JUEGO

En primer lugar cabe mencionar que este videojuego no refleja en absoluto todas las funcionalidades de los juegos iniciales de Pokémon, sinó que, como se ha mencionado anteriormente, es exclusivo para combates entre ellos.

El jugador dispone de tres pokémon iniciales; *Venusaur, Charizard* y *Blastoise* los cuales deberá utilizar para derrotar a una serie de Pokémon legendarios (Pokémon con unas características que los normales) para alzarse con la victoria. En particular, el jugador deberá derrotar de forma consecutiva a 5 de ellos seleccionados de forma pseudoaleatoria, mediante los movimientos que conocen sus Pokémon. Tal y como sucede en los videojuegos originales, existen movimientos llamados "*super efectivos"* que se caracterizan por hacer un daño generalmente más grande a los Pokémon adversarios. El jugador deberá descubrir cuál de entre los que tiene disponibles, causa un mayor daño al Pokémon rival para así tener más posibilidades de alzarse con la victoria.

Nótese que la CPU también conoce las debilidades de los Pokémon del jugador, sin embargo, estos no siempre responderán con los movimientos más efectivos sinó que usarán otros que tengan a su disposición.

Otro rasgo característico de los combates Pokémon es la melodía de fondo, y este simulador no es una excepción. Cada Pokémon rival presenta su propia melodía que irá cambiando a medida que vayan siendo derrotados. Las melodías que han sido utilizadas provienen de los videojuegos de Pokémon de quinta generación (***Pokémon Blanco*** y ***Pokémon Negro***). La melodía que suena al inicio dando a presentar el juego proviene de la instrucción de los juegos originales lanzados en 1996.

Finalmente comentar que en caso de que el jugador resulte vencedor, se le recompensará con un mensaje de felicitaciones en el que se muestran sus compañeros Pokémon. Esto es un guiño a lo que en los juegos principales se conoce como *Hall de la fama* y es la máxima condecoración que cualquier entrenador Pokémon pueda llegar a tener.

EXPLICACIÓN DEL JUEGO

En este apartado se va a proceder a la descripción de forma explícita y brevemente detallada de cómo se ha estructurado el código para llevar a cabo el desarrollo del videojuego.

En primer lugar se ha implementado un fichero llamado *"SYSTEM.X68"* el cual es el encargado de inicializar todo el sistema. Esto incluye la instalación de las distintas subrutinas que dan servicio a posibles interrupciones, como podría ser la lectura del teclado por ejemplo, o el limpiado de la pantalla correspondiente a una llamada al sistema. Las constantes que han sido requeridas por el fichero anteriormente mencionado se encuentran en "*SYSCONST.X68"* y contiene, entre otras informaciones, los valores hexadecimal de cada tecla utilizada en el videojuego. Del mismo modo las variables del sistema que son usadas se almacenan en el fichero "*SYSVARS.X68",* que contiene, por ejemplo, variables para registrar el estado de la tecla pulsada.

Uno de los ficheros más importantes (si no el más importante) de todos es el fichero "*SYSBATTLE.X68" .* Aquí se encuentran todas las rutinas que permiten un correcto funcionamiento de los sistemas de batalla Pokémon. Es importante mencionar, que los Pokémon no dejan de ser unos *sprites* con una dimensión considerablemente pequeña, en definitiva, originalmente una imágenes PNG que son convertidas en un fichero binario más amigable para el lenguaje ensamblador. La explicación de la implementación de cómo se lleva a término este proceso de conversión se explica en la página 6. Asimismo el fichero "*CONST.X68"* contiene todas las constantes requeridas por el videojuego. Explicar detalladamente cada punto sería laborioso, en consecuencia, consideraremos que junto "*VARS.X68"* constituyen los pilares del sistema de batalla.

En "*SYSBATTLE.X68"* una de las rutinas más importantes es *DRAWPKM,* que, como su propio nombre indica, es el encargado del dibujado de los *sprites* de los Pokémon por pantalla. Los *sprites* se encuentran almacenados secuencialmente en dos *buffers* (uno para el rival y otro para el jugador). Una vez obtenido un puntero al comienzo del *sprite* del Pokémon, se van leyendo 4 bytes cada vez, lo que se corresponde a un píxel por pantalla. Realizando el número adecuado de iteraciones en x en y, de acuerdo con el tamaño de los *sprites,* es posible la visualización de dicho Pokémon por pantalla.

Otra característica a tener en cuenta, es que anteriormente se ha mencionado que la generación de los Pokémon se hace de forma pseudoaleatoria. Esto es posible grácias al fichero "*RANDOM.X68"* . Dicho fichero implementa un rutina que calcula un número aleatorio basándose en el algoritmo LCR (*Linear Congruential*

Generator) y obteniendo como semilla generadora el tiempo en milisegundos del computador mediante una llamada al sistema TRAP #15.

La lectura de los *sprites,* así como la lectura de las distintas imágenes requeridas, se lleva a cabo mediante *"FILEREADER.X68"* , el cual contiene también una rutina que permite el dibujado de cualquier imágen (a excepción de los *sprites)* en la pantalla de visualización de 640x480 píxel por píxel.
Luego tenemos el fichero "*MENU.X68*". Aquí se encuentran todas las rutinas que permiten un correcto funcionamiento del menú de combate (movimientos pokemon). Es importante indicar que en total tendremos tres diferentes menús ("*MENUVEN"*, "*MENUCHAR"*, "*MENUBLAS"*), en el que cada uno tendrá diferentes movimientos. Es importante aclarar que el esqueleto del menú ha sido proporcionado por *Antonio Burguera.*

La rutina "*MENUINIT"* inicializa el sistema de menú y, "*MENUPD"* gestiona la actualización del menú, para mover el botón seleccionado en el menú. Si el usuario no se mueve entre el menú y pulsa "z", se mirará qué movimiento pokemon se ha seleccionado.

"MENUPLOT" dibuja las opciones del menú en la pantalla, el botón seleccionado estará en color gris mientras que el no seleccionado en gris oscuro.
Después un fichero que guarda relación con el menú es "*SYSMENU.X68*", en él encontramos dos subrutinas: "*SELMENU"* y "*CHGMENU"*.
"SELMENU" se encargá de ver que tipo de menú tiene que cargar de los tres pokemons para hacer la inicialización, actualización y dibujar la pantalla.
"CHGMENU" cambia el menú si nuestro pokémon ha muerto para así utilizar el menú apropiado en cada pokémon.

Finalmente vamos a comentar algunas de las dificultades que se nos han presentado durante el desarrollo del proyecto. En primer lugar, la actualización del menú nos ha generado bastantes problemas, por lo que al cambio de Pokémon se refiere, y es que no se actualizaban correctamente los movimientos correspondientes para cada Pokémon del jugador. El sincronismo ha sido otro punto en contra y es que, por algún motivo que desconocemos, al esperar un determinado tiempo al pulsar un tecla, el menú dejaba de funcionar y esto en consecuencia hacía el juego injugable.

CONVERSIÓN PNG A BINÁRIO

El programa utilizado para convertir imágenes PNG a binario ha sido **Python** usando el editor de texto **Visual Studio Code.** Los módulos necesarios han sido *cv2* instalado mediante el gestor de paquetes y dependencias *pip* de Python y *struct. cv2* es una herramienta poderosa que permite tareas como leer y escribir imágenes, aplicar filtros, detectar bordes, reconocer patrones, y trabajar con video.
Por otro lado, *struct* es la biblioteca que realiza la magia, permitiendo la conversión entre datos de Python y representaciones binarias (bytes). Es útil para leer y escribir datos binarios, como manipular archivos binarios o trabajar con redes y protocolos.

Este programa obtiene el valor en formato BGR por cada píxel en las coordenadas x e y de la imágen leída. Tras ello, se escribe en el fichero una estructura en la que “BBBB” indica que se trata de 4 bytes, siendo el primer byte ”*0*”, representando el canal alfa y a continuación los bytes “*b*” (blue), “*g*” (green) y “*r*” (red) en cada caso. Finalmente tras la escritura se cierra el fichero. El resultado de la ejecución de este programa garantiza la obtención de un fichero binario apto para Eas68K.

```python
import cv2
import struct

# Cargar la imagen
imagen = cv2.imread('logo.png')

# Obtener las dimensiones de la imagen
alto, ancho, _ = imagen.shape

#abrir archivo binario para escribir el código bgr
with open("pixel_logo.bin", "wb") as archivo:
    # Iterar sobre cada píxel y obtener el valor en formato BGR
    for y in range(alto):
        for x in range(ancho):
            # Obtener el valor BGR del píxel en (x, y)
            b, g, r = imagen[y, x]
            archivo.write(struct.pack("BBBB", 0, b, g, r))
    archivo.close()
print("Finalizado")
```

CONCLUSIONES

El desarrollo del proyecto “Pokemon Battle Simulator” ha sido un proceso tedioso que nos ha hecho utilizar recursos externos para transformar PNG a formato binario y técnicas en ensamblador novedosas para nosotros.

Nuestro proyecto ha conseguido abordar el sistema de combates y la carga de imágenes.

A pesar de los desafíos que nos hemos encontrado y haber reformado ideas, el proyecto ha sido una gran oportunidad para aprender programación en ensamblador y a gestionar el juego en un entorno limitado.