##

Kirby Dreamland for PIC18 Ver. 1.0

##

Bienvenido al Kirby Dreamland para PIC18 del Grupo E formado por Daniel Mejias y Biel Pérez, aquí un par de cosas que te pueden ser de ayuda para el uso de este.

Manual de Uso

Al iniciar el programa estaremos en el nivel 1 *Greens Greens* , podemos mover a kirby tanto por el teclado como por los botones del PIC18.

	Teclado	PIC18
Izquierda:	a	RA1
Derecha:	d	RA0
Saltar (o flotar):	k	RA5
Absorber:	j	RA3
Escupir:	j	RA3

Además también podemos activar tanto absorber y escupir mediante el sensor analógico de temperatura. Para hacer esto deberemos soplar a este así aumentando su temperatura y absorbiendo si no tenemos nada en la boca o escupiendo de otra forma. Para la inmersión también se activa el ventilador representando así la potencia de los pulmones de Kirby.

El objetivo es derrotar a Whispy Woods el cual se encuentra al final del nivel y asi completar el juego. (En futuras versiones se añadirán más niveles)

Código y Problemas

El código del juego está dividido principalmente en 4 archivos: main, picengine, animation y gamedata.

Main contiene el comportamiento principal del juego: Físicas, controles, interacciones, enemigos *etc*… y la mayoría de variables no constantes

Explicación de las colisiones:
Las colisiones usan colisiones libres cuadradas para los enemigos y colisiones cuadriculadas fijas para el fondo. Las libres básicamente comparan un punto con un cuadrado que podría estar en cualquier posición, mirando si el punto está entre los márgenes horizontales y verticales de este. La colisión cuadriculada se usa para comprobar colisiones con el fondo, ya que es bastante sencilla, simplemente divide la posición entre el tamaño fijo de los bloques (8 píxeles) y comprueba si el bloque en el que el punto se sitúa es de tipo sólido.

Explicación de los enemigos:
El comportamiento de los enemigos es complejo y depende del enemigo, hay 4 *slots* donde puede ponerse la información de un enemigo, esto se ve en enemyDat, cada slot tiene 8 bytes donde se guarda información de ese enemigo, la primera representa la posición horizontal del enemigo, la segunda la posición vertical, la tercera el tipo de enemigo (se usa

para determinar su comportamiento), la cuarta la imagen que usa en ese momento, si este byte es 128 el enemigo se considera muerto y el *slot* liberado, del quinto al séptimo byte son de uso libre según cada enemigo y el último se usa para determinar su id global del área (principalmente para evitar que se duplique por accidente).

El comportamiento general del juego en cada refresco es actualizar y mover todos los enemigos según su comportamiento, matar alguno si el jugador lo ha eliminado y comprobar si se debería crear algún enemigo según la posición de la pantalla y la información del área, aparte de esos factores también tendrá en cuenta que haya algún slot libre y que el enemigo que se quiere crear no exista ya en el mapa (con la id global).

Pic engine contiene todas las funciones relacionadas con mostrar la imágen en la izquierda de la pantalla, esta fue de las partes más complicadas de programar, ya que necesitábamos tener un fondo desplazable y modificable e imágenes desplazables por encima (sprites) de manera que procesar toda esa información y mostrarla fuera rápido.

Explicación del Pic Engine:
La variable fullImage contiene toda la información de cómo se verá la imagen cuando se ponga en la GLCD y es la que se va actualizando en cada refresco de pantalla. Primero se eliminan los sprites buscando donde están, borrando esa sección de la imagen y reemplazandola por el fondo que tendría que haber ahí.

Luego se añaden secciones de fondo según sea necesario, esto se hace porque el juego funciona en un “espacio circular”, cuando la cámara se mueve lo suficiente está en realidad está pasando por el mismo espacio otra vez, entonces se da la sensación de avance creando el mundo por delante de la pantalla a medida que se avanza sin que se note.

Después de borrar los sprites que habían y añadir la nueva sección del nivel necesaria (si hace falta) se vuelven a dibujar los sprites por encima del fondo. Una vez hecho esto, el fullImage está listo, se usa la variable camX como offset y se dibuja la información en la GLCD. Este proceso se hace a lo largo del main con asistencia del Pic Engine.

Animation contiene la información relacionada con las animaciones complejas y funciones para ejecutarlas, en el nivel disponible solo se usa para la estrella warp star que usa kirby al final de la primera área.

Finalmente, está Game Data que solo es un header, ya que no contiene funciones y se usa exclusivamente para la información del juego: Imágenes (dividido en móviles y estáticas), bloques (conjuntos de tiles), información de cada área y el contenido de estas.