Organización del Computador II

1er Cuatrimestre de 2008

Trabajo Práctico N°1: "Rescatando a la Princesa Peach"

Introducción

Se nos informó que la Princesa Peach está atrapada en el Reino Champiñón, y se nos ha encomendado ir a rescatarla.

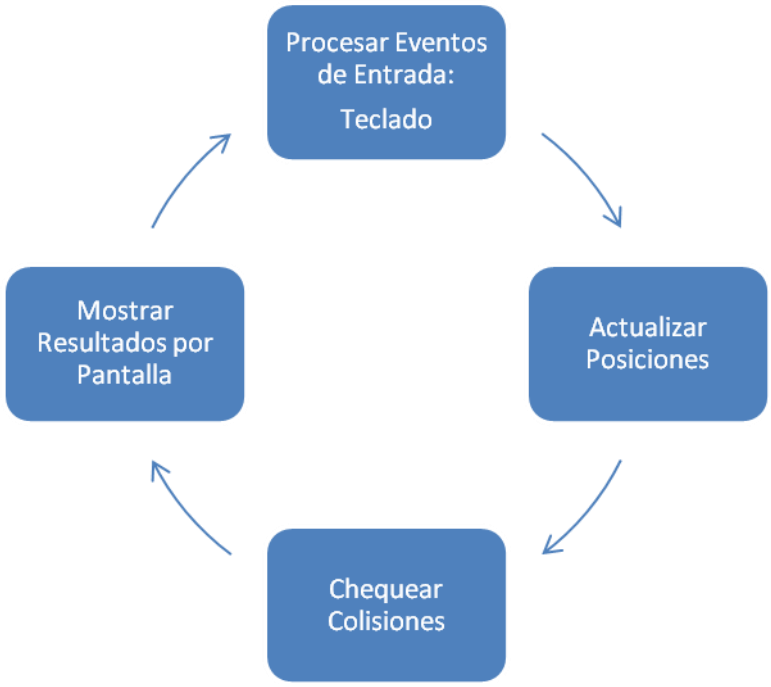
Para tan arriesgada misión, deberemos con anterioridad realizar una simulación por computadora de cómo será el rescate, para que nada salga mal.

Dicho "simulador" al que llamaremos de ahora en mas Super Mario Hnos. correrá sobre la nueva consola Nientiendo Güi, convirtiéndose así en el más novedoso juego sacado por esta firma en este año, la versión 114 del Super Mario.

El TP consiste en programar en ASM las funciones encargadas de generar ciertos efectos visuales de este video juego como solapar los *Sprites* y chequear las *colisiones*.

Para llevar a cabo esto veamos primero como será el funcionamiento.

Un video juego no es más que un programa que contiene los siguientes tres estados generales: la inicialización, el ciclo general, y la finalización. El primero como su nombre lo indica, es el encargado de reservar los recursos e inicializar las variables que sean necesarias, el tercero por su parte, será el encargado de liberar los recursos una vez que el juego termine, y el segundo, que es el que más nos importa, es un ciclo que cumple con el siguiente esquema:

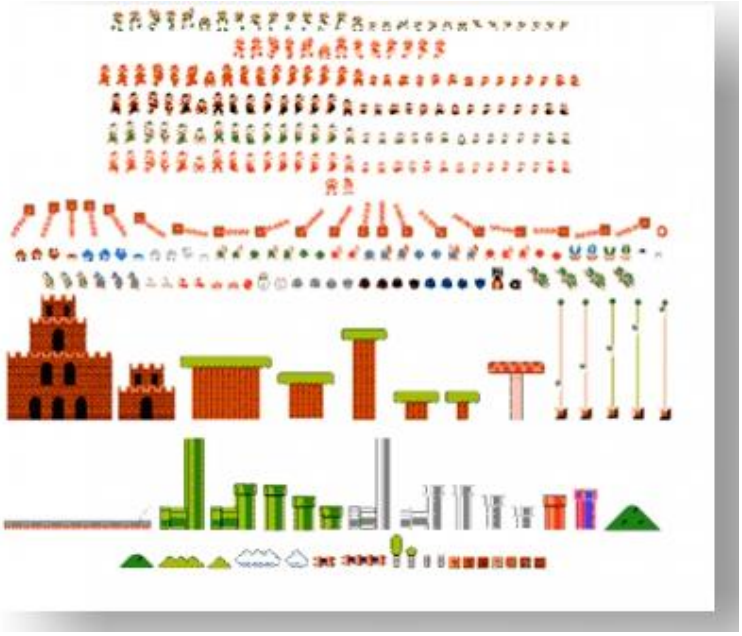


En este TP prestaremos más atención al segundo y tercer estado de este ciclo general, en donde deberán realizar las actualizaciones de las posiciones y chequear colisiones entre los *Sprites*.

SPRITES:

La gran mayoría de los juegos en 2D, manejan los gráficos utilizando bitmap's, que no es más que una imagen BMP en donde se encuentran todas las cosas que se mostrarán por pantalla.

Veamos un Ejemplo:



Como se ve, este BMP contiene todas las imágenes que se van a mostrar por pantalla, desde el cuerpo de Mario y Luigi en todas sus posiciones y tamaños, hasta las nubes que se ven en el cielo.

Cada uno de estos dibujos se los denomina “sprite”, y se colocan en un solo BMP, en vez de tener uno para cada uno, esto se hace así por una cuestión de optimización, así no hay que abrir tantas imágenes cada vez que se quiere cargar una nube, o mover a Mario.

Si prestan atención, el suelo que se ve en este tipo de video juegos, siempre repite un *patrón*, y esto se debe a que para formar ese suelo, lo que se hace es repetir constantemente un mismo sprite, uno al lado del otro, que da la apariencia de ese suelo continuo.

Por ejemplo, para generar el piso de nuestro video juego, repetiremos un Sprite como se ve a continuación:

De esta manera, un video juego se podría resumir a simplemente ir pegando estas figuritas (Sprites), de manera ordenada e ir cambiándolas de acuerdo a lo que se lea de la entrada y salida, y así ir produciendo la sensación de movimiento en los personajes, y las diferentes animaciones.

Desarrollo

Para desarrollar esto, deberán programar en lenguaje ensamblador las siguientes funciones, respetando la convención C.

```
void generarFondo(char* screen, int ws, int hs, char* piso, int wp, int hp);
```

Recibe como parámetros un puntero al lugar donde está la pantalla y sus dimensiones en píxeles (**ws** de ancho y **hs** de alto), un puntero al lugar de memoria donde está el Sprite correspondiente al piso y sus dimensiones en píxeles (**wp** de ancho y **hp** de alto). Se deberá rellenar la parte inferior de la pantalla repitiendo el Sprite del piso y rellenar la parte superior de la pantalla con el color azul, cuyas componentes RGB son (255, 150, 0)

La pantalla es simplemente una matriz de píxeles, donde cada píxel son tres bytes que representan las componentes Rojo, Verde y Azul (RGB) del color a mostrar.

Las matrices de pixeles de la imagen que representa el piso, y de todo el resto de las imágenes con las que se trabajará en este TP, se guardan de la siguiente manera:

El primer pixel de la parte superior izquierda es el que esta apuntado por el puntero piso, de ahí en más, se avanza de izquierda a derecha y de arriba hacia abajo, tomando a la matriz como un vector gigante. Tomar en cuenta que al igual que en los BMP, si una 'fila' de la imagen tiene una longitud en pixeles que no es múltiplo de cuatro, en memoria se extiende con bytes 'basura' en modulo cuatro la cantidad de bytes de cada fila de la matriz.

Ejemplo:

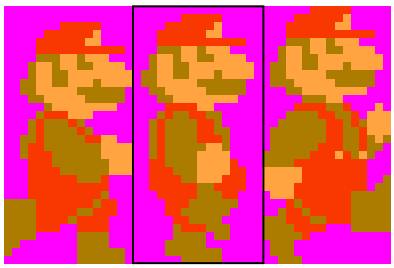
Una imagen de tres pixeles de ancho, cada fila debería tener 9 bytes, pero como no es múltiplo de cuatro, cada fila en la matriz que represente dicha imagen tendrá una longitud de 12 bytes. Esto solo pasa con las imágenes, la matriz que representa la pantalla se almacena intuitivamente.

```
void recortar(char *imagen, int paso, int wp, int w, int h, char *res);
```

Como vimos antes, en una imagen se pueden guardar varios Sprites que serán utilizados en el video juego. Por lo tanto, eventualmente tendremos que recortar una imagen de este que necesitemos para pegar en un determinado momento en el video juego. Esta función es la encargada de recortar un Sprite de una determinada imagen. Recibe como parámetros un puntero a la imagen, y sus dimensiones en píxeles (**w** de ancho y **h** de alto). El entero **paso** es el número de Sprite dentro de la imagen a recortar, y el entero **wp** es el ancho de cada uno dentro de la imagen (tienen todos el mismo ancho).

Finalmente, **res** es el puntero al lugar donde deberá copiarse el Sprite recortado (que tiene tamaño **wP** por **h** píxeles).

Por ejemplo, una imagen que utilizaremos es la que almacena los Sprites que corresponden a Mario caminando hacia la derecha, o sea:



Esta imagen tiene 96 pixeles de ancho por 64 de alto, y contiene 3 Sprites, donde cada uno tiene 32 pixeles de ancho. Entonces, si queremos recortar el del medio, debemos invocar a la función recortar con los siguientes parámetros:

```
recortar(&imagen, 1, 32, 96, 64, &res);
```

```
void blit(char *sprite, int w, int h);
```

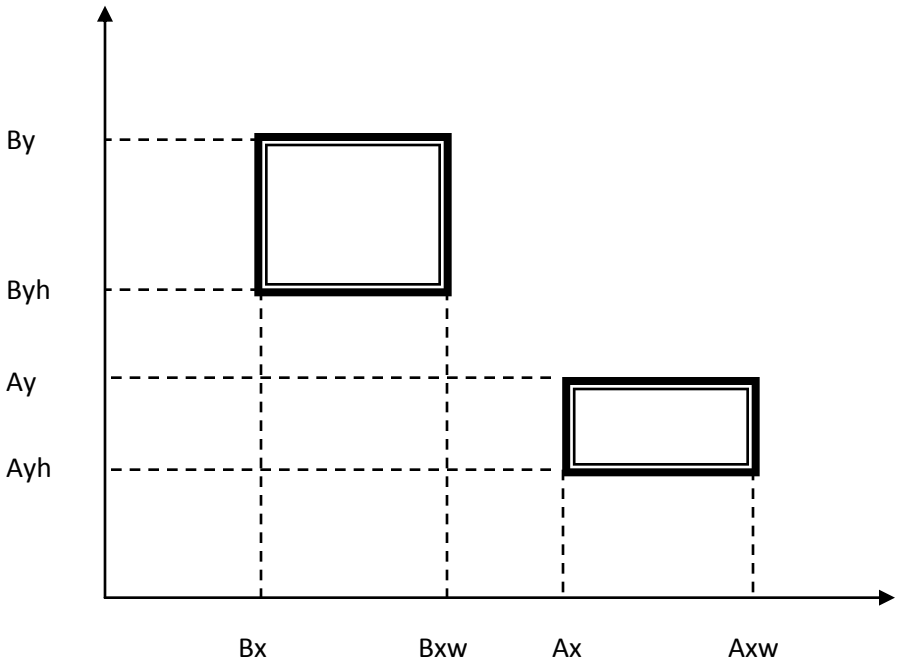
Esta función es la encargada de solapar un Sprite sobre el fondo de manera de que sólo se vea el dibujo, y no el color off. El parámetro **Sprite**, es un puntero a un área de memoria, en donde está almacenado, **w** es el ancho y **h** el alto del mismo. El color off es el fucsia cuyas componentes RGB son (255, 0, 255) y el color del fondo con el que se debe reemplazar es siempre el azul cuyas componentes RGB son (0, 150, 255).

```
int checkcolision
      (int Ax, int Ay, int Axw, int Ayh, int Bx, int By, int Bxw, int Byh);
```

Esta función chequea si dos rectángulos A y B se solapan. Esta función es útil porque los Sprites son simplemente rectángulos y el video juego tiene que ser capaz de saber por ejemplo si Mario está chocando contra una pared o algún otro objeto.

Recibe como parámetro las coordenadas del rectángulo A y las coordenadas del rectángulo B, y devuelve un 1 si se solapan y un 0 si no.

Las coordenadas del rectángulo vienen dadas de la siguiente manera:



En donde:

Ax : Posición x del objeto A
Ay : Posición y del objeto A
Axw : Posición x + ancho de A
Ayh : Posición y + alto de A

```
int apagar(char *sprite, int w, int h, int *cont);
```

Cuando la cabeza de nuestro personaje golpea sobre unas ciertas cajitas, del interior de estas salta una moneda que se difumina en el aire. Esta función es la que se encarga de difuminar la moneda. Para esto recibe un puntero al Sprite de la moneda, su ancho y su alto y un puntero a un contador. El efecto se logra decrementando únicamente el color amarillo de la moneda, esto se realiza así para simplificar la función.

Para eso debe ir cambiando el color de los píxeles que corresponden a la moneda **sin modificar** los píxeles que corresponden al color off del Sprite (como ya mencionamos, el color off es [255, 0, 255]).

El cambio que debe realizarse a los píxeles de la moneda es asignarles 0 a la componente roja, y el valor del contador a la componente azul y verde (amarillo). También se deberá decrementar el contador en 8 para que en el próximo ciclo del juego, se le asigne un valor menor a las componentes azul y verde y así ir formando la animación. El contador es siempre menor a 255 y mayor a 0.

```
void salta (int *saltando,
            int *bajando,
            int*posy,
            int *velocidad,
            int *crecerVelocidad,
            int cotaAbajo);
```

Cada vez que el usuario del video juego presione la tecla de salto, el personaje deberá saltar de una forma esperada, la misma corresponde al trazo de una parábola, en donde, la velocidad se irá decrementando hasta llegar a cero, y luego volverá a crecer de la misma manera hasta que el personaje sea frenado por alguna superficie. Esta misma parábola es la que describe la trayectoria de la moneda que salta de la caja al ser golpeada por Mario.

Los parámetros **saltando** y **bajando** indican si Mario (o la moneda) están subiendo (si **saltando** está en 1), bajando (si **bajando** está en 1) o no se mueve en dirección vertical en este paso (si **saltando** y **bajando** están ambos en 0). El parámetro **posy** indica la posición vertical del objeto (Mario o la moneda) que este procedimiento debe modificar dependiendo de si está subiendo o bajando. El parámetro **velocidad** indica en cuánto hay que modificar la posición **posy**, y el parámetro **crecerVelocidad** indica si hay que incrementar la velocidad para el próximo paso o decrementarla. Esto se realiza para simular el efecto parábola. Finalmente, el parámetro **cotaAbajo** indica la posición donde debe frenarse al caer, ya sea el piso, o un objeto.

Para aplicar el efecto mencionado, se aplicara el siguiente algoritmo.

```
void salta(saltando, bajando, posy, velocidad, crecerVelocidad, cotaAbajo)
{
       si (saltando = 1)
       {
            posy = posy - velocidad/c1;// estoy subiendo, la posy se decrementa.
              si (crecerVelocidad)
              {
                     velocidad++;
                     si(velocidad >= c2) crecerVelocidad=0;
              }si no
              {
                     velocidad--;
              }

              si (llegue al techo o velocidad=0)
              {
                     si (llegue o pasé el techo) posy=techo //techo en y = 0
                     saltando=0   //cambio estado de saltar a bajar
                     bajando=1
                     velocidad=1  //reseteo la velocidad en 1
              }
     }

     si (bajando = 1)
     {
              posy = posy + velocidad/c1;//estoy bajando, la posy se incrementa.
              velocidad++; //cuando bajo, la velocidad siempre se incrementa
              si (llegué o pasé la cotaAbajo)
              {
                     posy = cotaAbajo;
                     bajando= 0;
                     velocidad = vel_ini;
                     crecerVelocidad =1;
              }
     }
}
```

Hay tres constantes (**c1**, **c2** y **vel_ini**) que deberán probar hasta encontrar un valor que produzca el efecto de parábola deseado.

Otras consideraciones

Para que puedan probar el trabajo práctico, les entregaremos un **main** realizado en lenguaje **C** que utiliza estas funciones, todas las imágenes que contienen los Sprites a utilizar en este video juego y la librería **SDL** que es necesaria para la compilación y ejecución del mismo.

Informe

Debe reflejar el trabajo hecho para obtener el resultado, las decisiones tomadas (con el estudio de sus alternativas), las estructuras de datos usadas (con gráficos y/o dibujos si ayudan a clarificar), las pruebas que hayan hecho para tomar decisiones o al final para buscar errores en el producto final, etcétera. Debe contar como mínimo con los siguientes capítulos: introducción, desarrollo, discusión y conclusiones. Estar estructurado top-down o sea leyendo la introducción se debe saber qué se hizo y cuáles son las partes más importantes. Después de leer los primeros capítulos se debe saber cada cosa que se hizo y como se hicieron las más importantes. En el resto de los capítulos se debe poder leer el detalle de todo lo hecho.

Además, el informe debe incluir:

- Carátula con número del grupo y los nombres de los integrantes con número de libreta y email
- Manual del usuario
- Instrucciones para el corrector, por ejemplo como ensamblar los archivos fuente para obtener el ejecutable.
- Lista de todos los archivos entregados.

Entrega

La fecha de entrega de este trabajo es el martes **20 de mayo**, en el horario de clase (de 17 a 20 hs). No se aceptarán trabajos pasada esa fecha. Para poder contar con una instancia de recuperación del trabajo, es necesario haber entregado una primera versión en la fecha antes mencionada.

La entrega se realizará en un CD que debe incluir, los ejecutables, todos los archivos fuentes necesarios para crearlos y el informe en formato digital. Si desarrollaron prototipos en lenguaje C para resolver el trabajo primero en alto nivel, deben entregarlos e incluir los resultados obtenidos en el informe.