Universidad del Valle de Guatemala
IE3054 Electrónica Digital 2
Sección 10
Andrés Lemus 21634

Proyecto 2
Videojuego

Introducción

El proyecto 2 consistía en hacer un videojuego mediante la modificación de una pantalla LCD con un controlador ILI9341 y microprocesador de TIVA tm4c123gh6pm. La programación de este proyecto fue realizada en una IDE llamada Energía, que funciona para las placas de desarrollo de la empresa TIVA. Para poder programar la pantalla de una manera más fácil se utilizó una librería hecha y proporcionada por el catedrático donde se tenían funciones para mostrar figuras y sprites.

El proyecto debía tener controles, música y utilizar una memoria SD. El control para el juego debía ser de un microcontrolador externo, en este caso se utilizó un microcontrolador ESP32. Este estaba alimentado por medio de la placa de TIVA y como se tenían tres botones en configuración de pull-up, se utilizaron tres cables para poder comunicarse con la TIVA y poder utilizar las interrupciones y que el presionado sea lo más rápido posible. En el caso de la música también se utilizó un ESP32 controlando sus PWM para generar tonos y reproducirlos mediante un buzzer pasivo. La memoria SD se utilizó un lector externo que se comunica por medio del protocolo SPI con la TIVA.

Programación y Lógica

El programa se puede dividir en cuatro etapas que son: menú, juego, perdedor y ganador. El programa comienza cargando el menú, donde se carga una imagen de fondo y se muestra un texto. Cuando se termina de cargar la imagen el programa se queda polleando a que el usuario presione el botón de start. Al momento de que el jugador presiona el botón de start, el programa pasa al juego en sí. El juego está dividido por turnos, primero donde el enemigo tiene que atacar y el jugador tiene que esquivar, una vez el jugador haya esquivado un set de enemigos le tocará atacar. En el turno de ataque de jugador se mostrará en pantalla un rectángulo grande con colores, en el siguiente orden: rojo, anaranjado, amarillo, verde, amarillo, anaranjado y rojo de nuevo. Entonces, en el turno de ataque un rectángulo blanco se estará moviendo entre estos colores, y el jugador tiene que intentar que cuando presione el botón el rectángulo blanco este en el color verde ya que si presiona en el color verde el enemigo perderá 20 puntos de vida. El color amarillo son 15 puntos, el anaranjado 10 y el rojo 5. El enemigo tiene un total de 200 puntos de vida, y hay 10 sets específicos de enemigos, así que el jugador si cada turno le quita 20 puntos de vida podrá terminar el juego, sin que tenga que repetir sets de enemigos. Durante el turno de esquivar el jugador puede perder cuando choca con uno de los enemigos y en ese momento se acaba la etapa del juego, y se entra en la de perdedor. En esta etapa de igual forma carga una imagen con un fondo del jugador siendo atrapado por las telarañas, y se queda polleando al botón de start para volver al menú. Si el jugador logra llegar a 0 la vida del enemigo entonces se habrá ganado y cargará una foto de ganador y de igual

forma se pollea el botón para poder volver al menú principal. En eso se basa el juego, un juego por turnos donde se esquiva y se tiene el turno para hacerle daño al enemigo.

Funciones Importantes

Función de colisión

```
bool Collision(int x1, int y1, int w1, int h1, int x2, int y2, int w2, int
h2){
  return (x1 < x2 + w2) && (x1+ w1 > x2) && (y1 < y2 + h2) && (y1 + h1 > y2);
//Chequear las coordenas, anchos y altos
}
```

Está función tiene como parámetros las coordenadas en x y y, el ancho y alto de dos sprites que se quiere revisar si hay colisión. En esta función se revisan las coordenadas, el ancho y alto del jugador contra las del enemigo para así retornar un true o false si hay una colisión.

Función de Ataque del jugador

```
void attack(){
  detachInterrupt(PE_3); //Se desactivan las interrupciones de los botones de
movimiento
  detachInterrupt(PA_7); //Se desactivan las interrupciones de los botones de
movimiento
  //Si muffet no está en la pantalla mostrarla
  if (set < 5 || set > 9){

  }
  else {
    LCD_Sprite_Zoom(134, 25, 26, 35, muffet, 5, 5, 0, 2, 0x0000); //Mostrar a
muffet en la pantalla
  }
  //Relleanar el fondon para la preparación del ataque
  FillRect(0, 98, 320, 142, 0x0000);
  FillRect(58, 98, 204, 80, 0xFFFF);
  FillRect(60, 100, 200, 76, 0x0000);
  FillRect(128, 188, 64, 25, 0xDF61);
  FillRect(130, 190, 60, 21, 0x0000);
  LCD_Print("ATTACK", 135, 195, 1, 0xDF61, 0x0000);
  if (set == 0){
    LCD_Print("I THINK PURPLE", 80, 110, 1, 0xFFFF, 0x0000);
    LCD_Print("LOOKS GOOD ON YOU", 80, 140, 1, 0xFFFF, 0x0000);
  }
  //Pintar fondo de los rectángulos de colores para el turno de ataque
  while (attack_flag == 0){ //Revisar bandera del fondo
    if (digitalRead(PF_4) == LOW){ //Pollear el botón
      //Pintar fondo
      FillRect(18, 98, 282, 120, 0xFFFF);
      FillRect(20, 100, 278, 116, 0x0000);
      FillRect(20, 100, 40, 116, 0xE841);
      FillRect(60, 100, 40, 116, 0xEA41);
```

```
      FillRect(100, 100, 40, 116, 0xDF61);
      FillRect(140, 100, 40, 116, 0x1705);
      FillRect(180, 100, 40, 116, 0xDF61);
      FillRect(220, 100, 40, 116, 0xEA41);
      FillRect(260, 100, 38, 116, 0xE841);
      attack_flag = 1; //Encnder bandera para que no se esté pintando el
fondo
    }
  }

  //Si ya se pintó el fondo quedarse en un loop
  while (attack_flag == 1){
    if (x_attack < 294){ //Que avance el rectángulo blanco de ataque mientras
no haya llegado al límite
      attachInterrupt(digitalPinToInterrupt(PF_4), hit, FALLING); //Activar
interrupción del botón de ataque
      FillRect(x_attack, 105, 5, 100, 0xFFFF); //Pintar rectángulo
      x_attack = x_attack + 1; //Ir avanzando un espacio
      //Dependiendo del color de fondo en que se encuentre se limpiará el
rastro con el respectivo color
      if (x_attack <= 60 && x_attack >= 25){
        FillRect(x_attack - 5, 105, 5, 100, 0xE841); //Rojo
      }

      else if (x_attack <= 100 && x_attack >= 65){
        FillRect(x_attack - 5, 105, 5, 100, 0xEA41); //Anaranjado
      }

      else if (x_attack <= 140 && x_attack >= 105){
        FillRect(x_attack - 5, 105, 5, 100, 0xDF61); //Amarillo
      }

      else if (x_attack <= 180 && x_attack >= 145){
        FillRect(x_attack - 5, 105, 5, 100, 0x1705); //Verde
      }

      else if (x_attack <= 220 && x_attack >= 185){
        FillRect(x_attack - 5, 105, 5, 100, 0xDF61); //Amarillo
      }

      else if (x_attack <= 260 && x_attack >= 225){
        FillRect(x_attack - 5, 105, 5, 100, 0xEA41); //Anaranjado
      }

      else if (x_attack <= 298 && x_attack >= 265){
        FillRect(x_attack - 5, 105, 5, 100, 0xE841); //Rojo
      }

    }

     //Si el jugador no presiono, lo detecta como si solo haya presionado en
el color rojo
     else {
```

```
      FillRect(x_attack, 105, 5, 100, 0xE841);
      hp_muffet = hp_muffet - 5; //Restar 5 de HP
      if (hp_muffet < 0){ //Si es menor a 0
          hp_muffet = 0; //Igualar a 0
        }
      LCD_Print("-5", 200, 50, 2, 0xEA41, 0x0000);
      attack_flag = 0; //Terminar ataque
      detachInterrupt(PF_4); //Desactivar interrupción de ataque
      x_attack = 20; //Regresar rectángulo de ataque a su posición
      background = 2; //Reinciar variable del fondo para el juego
      sprintf(buffer, "HP %03d/200", hp_muffet); //Convertir en string la
vida para mostrar en la pantalla
      LCD_Print(buffer, 20, 20, 1, 0xFFFF, 0x0000); //Mostrar en la pantalla
      FillRect(20, 40, 100, 10, 0x6800);
      if (hp_muffet == 0){ //Si la vida es 0
          hp_bar = 1; //Mostrar un poquita de la barra
        }
        else {
          hp_bar = map(hp_muffet, 0, 200, 0, 100); //Mapear valores para la
barra de vida
        }
      FillRect(20, 40, hp_bar, 10, 0x16A3); //Mostrar barra de vida
      delay(500); //delay de 50ms
    }
  }
}
```

En está función primero el jugador tiene que presionar el botón de start para confirmar que está lista para poder atacar. Así que el programa se queda polleando el botón para la confirmación. Una vez presionado el botón se carga el fondo con los rectángulos de colores y se enciende la interrupción del botón start para que cuando se esté moviendo el rectángulo sea más rápida la transición cuando se presiona. Vemos al final que si no se presiona se detectará que el jugador presionó el botón en el color rojo y únicamente se reducirán 5 HP del enemigo.

Función de ataque cuando se presiona el botón

```
void hit(){
  delay(10);
  if (x_attack <= 60 && x_attack >= 20){
        hp_muffet -= 5;
        if (hp_muffet < 0){
          hp_muffet = 0;
        }
        LCD_Print("-5", 200, 50, 2, 0xEA41, 0x0000);
        attack_flag = 0;
        x_attack = 20;
        background = 2;
        detachInterrupt(PF_4);
        sprintf(buffer, "HP %03d/200", hp_muffet);
        LCD_Print(buffer, 20, 20, 1, 0xFFFF, 0x0000);
```

```
  FillRect(20, 40, 100, 10, 0x6800);
  if (hp_muffet <= 0){
    hp_bar = 1;
  }
  else{
    hp_bar = map(hp_muffet, 0, 200, 0, 100);
  }
  FillRect(20, 40, hp_bar, 10, 0x16A3);
  delay(500);
}

else if (x_attack <= 100 && x_attack >= 60){
  hp_muffet -= 10;
  if (hp_muffet < 0){
    hp_muffet = 0;
  }
  LCD_Print("-10", 200, 50, 2, 0xEA41, 0x0000);
  attack_flag = 0;
  x_attack = 20;
  background = 2;
  detachInterrupt(PF_4);
  sprintf(buffer, "HP %03d/200", hp_muffet);
  LCD_Print(buffer, 20, 20, 1, 0xFFFF, 0x0000);
  FillRect(20, 40, 100, 10, 0x6800);
  if (hp_muffet <= 0){
    hp_bar = 1;
  }
  else{
    hp_bar = map(hp_muffet, 0, 200, 0, 100);
  }
  FillRect(20, 40, hp_bar, 10, 0x16A3);
  delay(500);
}

else if (x_attack <= 140 && x_attack >= 100){
  hp_muffet -= 15;
  if (hp_muffet < 0){
    hp_muffet = 0;
  }
  LCD_Print("-15", 200, 50, 2, 0xEA41, 0x0000);
  attack_flag = 0;
  x_attack = 20;
  background = 2;
  detachInterrupt(PF_4);
  sprintf(buffer, "HP %03d/200", hp_muffet);
  LCD_Print(buffer, 20, 20, 1, 0xFFFF, 0x0000);
  FillRect(20, 40, 100, 10, 0x6800);
  if (hp_muffet <= 0){
    hp_bar = 1;
  }
  else{
    hp_bar = map(hp_muffet, 0, 200, 0, 100);
  }
```

```
  FillRect(20, 40, hp_bar, 10, 0x16A3);
  delay(500);
}

else if (x_attack <= 180 && x_attack >= 140){
  hp_muffet -= 20;
  if (hp_muffet < 0){
    hp_muffet = 0;
  }
  LCD_Print("-20", 200, 50, 2, 0xEA41, 0x0000);
  attack_flag = 0;
  x_attack = 20;
  background = 2;
  detachInterrupt(PF_4);
  sprintf(buffer, "HP %03d/200", hp_muffet);
  LCD_Print(buffer, 20, 20, 1, 0xFFFF, 0x0000);
  FillRect(20, 40, 100, 10, 0x6800);
  if (hp_muffet <= 0){
    hp_bar = 1;
  }
  else{
    hp_bar = map(hp_muffet, 0, 200, 0, 100);
  }
  FillRect(20, 40, hp_bar, 10, 0x16A3);
  delay(500);
}

else if (x_attack <= 220 && x_attack >= 180){
  hp_muffet -= 15;
  if (hp_muffet < 0){
    hp_muffet = 0;
  }
  LCD_Print("-15", 200, 50, 2, 0xEA41, 0x0000);
  attack_flag = 0;
  x_attack = 20;
  background = 2;
  detachInterrupt(PF_4);
  sprintf(buffer, "HP %03d/200", hp_muffet);
  LCD_Print(buffer, 20, 20, 1, 0xFFFF, 0x0000);
  FillRect(20, 40, 100, 10, 0x6800);
  if (hp_muffet <= 0){
    hp_bar = 1;
  }
  else{
    hp_bar = map(hp_muffet, 0, 200, 0, 100);
  }
  FillRect(20, 40, hp_bar, 10, 0x16A3);
  delay(500);
}

else if (x_attack <= 260 && x_attack >= 220){
  hp_muffet -= 10;
  if (hp_muffet < 0){
```

```
        hp_muffet = 0;
      }
      LCD_Print("-10", 200, 50, 2, 0xEA41, 0x6800);
      attack_flag = 0;
      x_attack = 20;
      background = 2;
      detachInterrupt(PF_4);
      sprintf(buffer, "HP %03d/200", hp_muffet);
      LCD_Print(buffer, 20, 20, 1, 0xFFFF, 0x0000);
      FillRect(20, 40, 100, 10, 0x6800);
      if (hp_muffet <= 0){
        hp_bar = 1;
      }
      else{
        hp_bar = map(hp_muffet, 0, 200, 0, 100);
      }
      FillRect(20, 40, hp_bar, 10, 0x16A3);
      delay(500);
    }

    else if (x_attack <= 298 && x_attack >= 260){
      hp_muffet -= 5;
      if (hp_muffet < 0){
        hp_muffet = 0;
      }
      LCD_Print("-5", 200, 50, 2, 0xEA41, 0x0000);
      attack_flag = 0;
      x_attack = 20;
      background = 2;
      detachInterrupt(PF_4);
      sprintf(buffer, "HP %03d/200", hp_muffet);
      LCD_Print(buffer, 20, 20, 1, 0xFFFF, 0x0000);
      FillRect(20, 40, 100, 10, 0x6800);
      if (hp_muffet <= 0){
        hp_bar = 1;
      }
      else{
        hp_bar = map(hp_muffet, 0, 200, 0, 100);
      }
      FillRect(20, 40, hp_bar, 10, 0x16A3);
      delay(500);
    }
}
```

Está función es la que se llama cuando el rectángulo de ataque es está moviendo y el jugador presiona el botón, chequea la coordenada de ataque del rectángulo y dependiendo de su coordenada en ese instante el jugador habrá hecho cierta cantidad de daño al enemigo.

Función de lectura de la memoria microSD

```
void mapeoSD(){
  int hex1 = 0; //Variable para agarrar lo leído de la SD
  int val1 = 0; //Variable para guardar decena del valor hexadecimal
  int val2 = 0; //Variable para guardar la unidad hexadecimal
  int mapear = 0; //Iniciar con la posición del arreglo en 0
  int vertical = 0; //Iniciar con la coordenada en y en 0
  unsigned char maps[640]; //Definir un arreglo de 640 espacios

  if (myFile){ //Si se encontró un archivo con el nombre definido previamente
ejecutar
    while (myFile.available()){ //Siempre que haya contenido por leer en la
SD ejecutar
      mapear = 0; //Iniciar la posición de mapear en 0
      while (mapear < 640){ //Mientras se llegue
        hex1 = myFile.read(); //Leer del archivo de texto
        if (hex1 == 120){ //Si se leyó una "x", significa que vienen dos
valores en hexadecimal que si hay que guardar
          val1 = myFile.read(); //Leer y almacenar en una variable
          val2 = myFile.read(); //Leer y almacenar en una variable
          val1 = asciitohex(val1); //Convertir el valor de ASCII a
hexadecimal
          val2 = asciitohex(val2); //Convertir el valor de ASCII a
hexadecimal
          maps[mapear] = val1*16 + val2; //Guardar el valor de mapeo en el
arreglo
          mapear++; //Aumentar la posición del arreglo
        }
      }

      LCD_Bitmap(0, vertical, 320, 1, maps); //Mostrar una fila de contenido
de la imagen
      vertical++; //Aumentar la línea vertical
    }

    myFile.close(); //Cerrar el archivo si ya hay contenido por leer
  }
  else {
    myFile.close(); //Si no se encontró el arhivo cerrar para evitar algún
error
    Serial.println("NO SE ABRIO");
  }
}
```

En está función tenemos que definir distintas variables como lo indica la función en sí. El primer paso es definir el objeto de la librería SD con un nombre como se muestra a continuación:

```
#include <SD.h>

File myFile;
```

Luego, debemos iniciar la comunicación SPI:

```
pinMode(PA_3, OUTPUT); //Se define PA_3 como salida y se convierte en slave
select

SPI.setModule(0); //Utilizar el SPI Module 0

if (!SD.begin(PA_3)){ //Verificar que el slave select este en 0, y se
responda un 1
    Serial.println("initialization failed!"); //Si no falló la inicialización
    return;
  }
  Serial.println("initialization done."); //Se inició la comunicación
```

Ahora sí para leer un archivo debemos asignar el objeto de la libreía SD que definimos previamente como myFile de la siguiente manera:

```
myFile = SD.open("muf.txt");
mapeoSD();
```

Donde el argumento de la función SD.open es el nombre del archivo incluido su tipo entre comillas, para luego llamar a la función.