José Alejando Monzón Paz – 21560

Pablo Javier Peña Echeverría – 21210

Electrónica digital 2: Proyecto No.2 – videojuego

Descripción:

El juego esta principalmente basado en Battle City, el cual es un juego de tanques en el que hay que defender la base aliada de tanques enemigos. El juego se alteró para hacerlo más sencillo de producir y solo tiene un nivel.

Entre otras características tenemos el menú el cual da la opción de uno o dos jugadores mientras suena la canción de inicio a través del buzzer. Y un control de PlayStation4 que se comunica con un ESP32 el cual se comunica con la Tiva C.

Código:

A cada jugador y enemigo se le asignó un conjunto de variables para controlar su posición, su estado, en el caso de los enemigos su velocidad, la posición de su bala y el estado de su bala. Esto es importante porque básicamente el resto del código son comparaciones entre estas variables para determinar si hay colisión y si es game over.

```
int enemy22Life = 1;
int enemy22x = 218;
int enemy22y = 0;
int enemy22Pace = 0;
int xEne22Bullet = 0;
int yEne22Bullet = 0;
int enemy22BulletPace = 0;

int enemy23Life = 1;
int enemy23x = 266;
int enemy23y = 0;
int enemy23Pace = 0;
int xEne23Bullet = 0;
int yEne23Bullet = 0;
int enemy23BulletPace = 0;

int tripleWave5 = 0;

int xP1 = 0;
int yP1 = 209;
int P1facing = 0;
int triggerP1 = 0;
int P1bullDirection = 0;
int xP1bullet = 0;
int yP1bullet = 0;
int P1Life = 0;

int xP2 = 290;
int yP2 = 209;
int P2facing = 0;
int triggerP2 = 0;
int P2bullDirection = 0;
int xP2bullet = 0;
int yP2bullet = 0;
int P2Life = 0;
```

Dentro del loop principal lo primero que se hace es leer los joysticks que se usaron como controles, los cuales son básicamente dos potenciómetros y un botón. El loop tiene tres estados: menú, un jugador y dos jugadores. Al iniciar la Tiva C el estado por default es 0, y con el joystick del jugador uno se puede elegir entre estado 1 y 2, pero solo se dará el cambio de estado el apachar el botón del jugador uno.

```
//*****************************************************************************
// Loop Infinito
//*****************************************************************************
void loop() {
  //int i =30;
  int yPlayer1 = analogRead(PE_2);
  int xPlayer1 = analogRead(PE_3);
  int buttonP1 = analogRead(PE_5);

  int yPlayer2 = analogRead(PE_4);
  int xPlayer2 = analogRead(PE_0);
  int buttonP2 = 0;

  int animy = (yP1/5)%2;
  int animx = (xP1/5)%2;

  int animy2 = (yP2/5)%2;
  int animx2 = (xP2/5)%2;

  if(state == 0){
    if(yPlayer1 > 2300){//down
      LCD_Bitmap(97, 130, 13, 13, pick);
      selection = 2;
      for(int i = 0; i < 13; i++){
          H_line(97, 115+i, 13, 0x00);
      }
    }
    if(yPlayer1 < 1700){//up
      LCD_Bitmap(97, 115, 13, 13, pick);
      selection = 1;
      for(int i = 0; i < 13; i++){
        H_line(97, 130+i, 13, 0x00);
      }
    }
    if(buttonP1 < 10 && selection == 1){
      state = 1;
    }else if(buttonP1 <10 && selection == 2){
      state = 2;
    }
  }else if(state == 1){
```

Al entrar en el estado uno lo primero que se hace es borrar el fondo, pero solo en el primer ciclo, y luego las lecturas del joystick se usan para determinar en qué dirección se moverá el jugador uno, y en qué dirección se moverán los proyectiles. Para determinar la dirección de disparo se guarda la última dirección en la que el tanque del jugador estaba apuntando al moverse. Cabe destacar que el jugador no puede disparar para atrás y que tiene que esperar que su bala se salga del mapa o choque con un enemigo antes de volver a disparar.

```
if(yPlayer1 > 2300 && P1Life == 0){//down
  yP1++;
  P1facing = 3;
  if(yP1 == 211){
    yP1 = 210;
  }
  LCD_Sprite(xP1, yP1, 30, 30, tankP1Backward, 2,animy, 0, 0);
  H_line(xP1, yP1-1, 30, 0x00);
}
if(yPlayer1 < 1700 && P1Life == 0){//up
  yP1--;
  P1facing = 1;
  if(yP1 == -1){
    yP1 = 0;
  }
  LCD_Sprite(xP1, yP1, 30, 30, tankP1Forward, 2,animy, 0, 0);
  H_line(xP1, yP1+30, 30, 0x00);
}
if(xPlayer1 > 2300 && P1Life == 0){//left
  xP1--;
  P1facing = 2;
  if(xP1 == -1){
    xP1 = 0;
  }
  LCD_Sprite(xP1, yP1, 30, 30, tankP1LR, 2, animx, 1, 0);
  V_line(xP1+30, yP1, 30, 0x00);
}
if(xPlayer1  < 1700 && P1Life == 0){//right
  xP1++;
  P1facing = 4;
  if(xP1 == 291){
    xP1 = 290;
  }
  LCD_Sprite(xP1, yP1, 30, 30, tankP1LR, 2, animx, 0, 0);
  V_line(xP1-1, yP1, 30, 0x00);
}
```

```
if(buttonP1 < 10 && P1Life == 0){
  if(triggerP1 == 0 && P1facing == 1){//shooting up
    triggerP1 = 1;
    P1bullDirection = 1;
    xP1bullet = xP1 + 9;
    yP1bullet = yP1 - 10;
  }
  if(triggerP1 == 0 && P1facing == 4){//shooting right
    triggerP1 = 1;
    P1bullDirection = 2;
    xP1bullet = xP1 + 30;
    yP1bullet = yP1 + 10;
  }
  if(triggerP1 == 0 && P1facing == 2){//shooting left
    triggerP1 = 1;
    P1bullDirection = 3;
    xP1bullet = xP1 - 10;
    yP1bullet = yP1 + 10;
  }
}
```

Para los enemigos se construyó una misma estructura para todos, lo único que cambia es el momento en el que son activados y las variables donde se guardan sus estados.

Primero se revisa si el enemigo está vivo, luego se revisa su posición en “y” para determinar si tiene que disparar o no. Los primeros enemigos se mueven una posición cada 6 ciclos y sus balas cada 2, en comparación con el jugador que se mueve cada ciclo al igual que su bala.

```
//Enemy #1 code
//*********************************************************************************************
  if(enemy1Life == 0){
    if(enemy1y == 0){
      xEne1Bullet = enemy1x + 8;
      yEne1Bullet = enemy1y + 30;
    }
    if(enemy1y == 90){
      xEne1Bullet = enemy1x + 8;
      yEne1Bullet = enemy1y + 30;
    }
    int animEne1 = (enemy1y/5)%2;
    LCD_Sprite(enemy1x, enemy1y, 28, 30, enemyTankForward, 2, animEne1, 0, 0);
    H_line(enemy1x, enemy1y-1, 28,0x00);
    LCD_Bitmap(xEne1Bullet, yEne1Bullet, 10, 10, bulletDown);
    H_line(xEne1Bullet, yEne1Bullet, 10, 0x00);
    if(enemy1Pace == 0){
      enemy1y++;
      enemy1Pace = 1;
    }else if(enemy1Pace == 1){
      enemy1Pace = 2;
      yEne1Bullet++;
      if(yEne1Bullet >= 250){
        yEne1Bullet = 245;
      }
    }else if(enemy1Pace == 2){
      enemy1Pace = 3;
    }else if(enemy1Pace == 3){
      enemy1Pace = 4;
      yEne1Bullet++;
      if(yEne1Bullet >= 250){
        yEne1Bullet = 245;
      }
    }else if(enemy1Pace == 4){
      enemy1Pace =  5;
    }else if(enemy1Pace == 5){
      enemy1Pace =  0;
      yEne1Bullet++;
      if(yEne1Bullet >= 250){
        yEne1Bullet = 245;
      }
    }
```

Luego de esto se revisa si la nueva posición del tanque enemigo llegó al final de la pantalla (la base aliada en este caso) o la posición del sprite del enemigo interseca con la posición del sprite del jugador. En el caso de que esto pase, el enemigo muere, el jugador muere, se coloca un bitmap de explosión sobre la posición del jugador y se coloca bitmap que dice “GAME OVER”. Luego de un delay de tres segundos se resetean todas las variables que pudieron haber cambiando mientras se corría el juego y el estado del loop se cambia a 0 lo que nos regresa al menú principal.

Esto mismo se hace con la bala del enemigo con la diferencia de que en este caso el juego no se acaba si la bala del enemigo llega al final de la pantalla, pero con esa excepción es lo mismo.

```
if((enemy1y == 209)||((xPl+29)>enemy1x && (xPl+1)<(enemy1x+28) && yPl<(enemy1y+30) && (yPl+30)>enemy1y)){
  enemy1Life = 1;
  PlLife = 1;
  LCD_Sprite(xPl, yPl, 30, 30, dead, 2, 1, 0, 0);
  mapeo_SD("gameOver.txt", 67, 75, 186);
  delay(3000);
  LCD_Clear(0x00);
  enemy1Life = 0;
  enemy1y = 0;
  enemy2y = 0;
  enemy3y = 0;
  enemy4y = 0;
  enemy5y = 0;
  enemy6y = 0;
  enemy7y = 0;
  enemy8y = 0;
  enemy9y = 0;
  enemy10y = 0;
  enemy11y = 0;
  enemy12y = 0;
  enemy13y = 0;
  enemy14y = 0;
  enemy15y = 0;
  enemy16y = 0;
  enemy17y = 0;
  enemy18y = 0;
  enemy19y = 0;
  enemy20y = 0;
  enemy21y = 0;
  enemy22y = 0;
  enemy23y = 0;
  tripleWave1 = 0;
  tripleWave2 = 0;
  tripleWave3 = 0;
  tripleWave4 = 0;
  tripleWave5 = 0;
  PlLife = 0;
  xPl = 0;
  yPl = 209;
  state = 0;
  turn = 0;
  mapeo_SD("menu.txt", 32, 25, 256);
```

Este mismo código se repitió para los 23 enemigos que se pueden encontrar durante el primer nivel, la diferencia obviamente es que en lugar de realizar las comparaciones con las variables del enemigo uno se cambió a las variables del enemigo correspondiente.

Luego de esta sección se encuentra el código para la colisión de la bala del jugador. Básicamente se revisa si el espacio ocupado por el bitmap de la bala interseca el espacio del enemigo. En el caso de que si, el enemigo muere, su bala muere, se deja correr una corta animación de explosión, se borra esta explosión, se resetea la posición de la bala del jugador y se activa la siguiente oleada de tanques enemigos. Esto significa que al principio del juego todos los tanques están “muertos” con excepción del enemigo 1, y cuando este muere se activa al siguiente y así hasta que todos mueren.

Los primeros 3 enemigos vienen uno por uno, pero después empiezan a venir en oleadas de 3, luego de 4 y la oleada final es de 6 la cual cubre toda la pantalla. Para esto se tuvo que crear una variable contadora para saber cuándo todos los tanques de una oleada estuvieran muertos.

```
//Player #1 bullet collision
//*****************************************************************************************************************************
  if(yPlbullet <= (enemy1y+28) && (xPlbullet+9)>enemy1x && (xPlbullet+1)<(enemy1x+28) && enemy1Life == 0){//Enemy #1
    enemy1Life = 1;
    enemy2Life = 0;
    LCD_Sprite(enemy1x, enemy1y, 30, 30, dead, 2, 1, 0, 0);
    LCD_Sprite(xEne1Bullet-10, yEne1Bullet-10, 30, 30, dead, 2, 0, 0, 0);
    delay(15);
    for(int i = 0; i<31; i++){
      H_line(enemy1x, enemy1y+i-1, 30, 0x00);
      H_line(xEne1Bullet-10, yEne1Bullet-11+i, 30, 0x00);
    }
    for(int i = 0; i <= 11; i++){
      V_line(xPlbullet+10-i, yPlbullet, 10, 0x00);
    }
    triggerPl = 0;
    xPlbullet = -10;
    yPlbullet = -10;
  }

  if(yPlbullet <= (enemy2y+28) && (xPlbullet+9)>enemy2x && (xPlbullet+1)<(enemy2x+28) && enemy2Life == 0){//Enemy #2
    enemy2Life = 1;
    enemy3Life = 0;
    LCD_Sprite(enemy2x, enemy2y, 30, 30, dead, 2, 1, 0, 0);
    LCD_Sprite(xEne2Bullet-10, yEne2Bullet-10, 30, 30, dead, 2, 0, 0, 0);
    delay(15);
    for(int i = 0; i<31; i++){
      H_line(enemy2x, enemy2y+i-1, 30, 0x00);
      H_line(xEne2Bullet-10, yEne2Bullet-11+i, 30, 0x00);
    }
    for(int i = 0; i <= 11; i++){
      V_line(xPlbullet+10-i, yPlbullet, 10, 0x00);
    }
    triggerPl = 0;
    xPlbullet = -10;
    yPlbullet = -10;
  }



if(yPlbullet <= (enemy4y+28) && (xPlbullet+9)>enemy4x && (xPlbullet+1)<(enemy4x+28) && enemy4Life == 0){//Enemy #4
  enemy4Life = 1;
  tripleWave1++;
  if(tripleWave1 == 3){
    tripleWave1 = 0;
    enemy7Life = 0;
    enemy8Life = 0;
    enemy9Life = 0;
  }
  LCD_Sprite(enemy4x, enemy4y, 30, 30, dead, 2, 1, 0, 0);
  LCD_Sprite(xEne4Bullet-10, yEne4Bullet-10, 30, 30, dead, 2, 0, 0, 0);
  delay(15);
  for(int i = 0; i<31; i++){
    H_line(enemy4x, enemy4y+i-1, 30, 0x00);
    H_line(xEne4Bullet-10, yEne4Bullet-11+i, 30, 0x00);
  }
  for(int i = 0; i <= 11; i++){
    V_line(xPlbullet+10-i, yPlbullet, 10, 0x00);
  }
  triggerPl = 0;
  xPlbullet = -10;
  yPlbullet = -10;
}
```

Para la última oleada se sigue usando una variable contadora y cuando esta llega a 6 en lugar de liberar la siguiente oleada se muestra el mensaje de “VICTORY” y, similar al código en el que se revisa la colisión entre el jugador y el enemigo o su bala, se reinician todas las variables y se regresa el menú de inicio.

```
if(yPlbullet <= (enemy18y+28) && (xPlbullet+9)>enemy18x && (xPlbullet+1)<(enemy18x+28) && enemy18Life == 0){//Enemy #18
  enemy18Life = 1;
  tripleWave5++;
  LCD_Sprite(enemy18x, enemy18y, 30, 30, dead, 2, 1, 0, 0);
  LCD_Sprite(xEne18Bullet-10, yEne18Bullet-10, 30, 30, dead, 2, 0, 0, 0);
  delay(15);
  for(int i = 0; i<31; i++){
    H_line(enemy18x, enemy18y+i-1, 30, 0x00);
    H_line(xEne18Bullet-10, yEne18Bullet-11+i, 30, 0x00);
  }
  for(int i = 0; i <= 11; i++){
    V_line(xPlbullet+10-i, yPlbullet, 10, 0x00);
  }
  triggerPl = 0;
  xPlbullet = -10;
  yPlbullet = -10;
  if(tripleWave5 == 6){
    tripleWave5 = 0;
    LCD_Print(text1, 105, 100, 2, 0xffff, 0x00);
    delay(3000);
    LCD_Clear(0x00);
    enemy1Life = 0;
    enemy1y = 0;
    enemy2y = 0;
    enemy3y = 0;
    enemy4y = 0;
    enemy5y = 0;
    enemy6y = 0;
    enemy7y = 0;
    enemy8y = 0;
    enemy9y = 0;
    enemy10y = 0;
    enemy11y = 0;
    enemy12y = 0;
    enemy13y = 0;
    enemy14y = 0;
    enemy15y = 0;
    enemy16y = 0;
    enemy17y = 0;
    enemy18y = 0;
    enemy19y = 0;
    enemy20y = 0;

     enemy21y = 0;
     enemy22y = 0;
     enemy23y = 0;
     PlLife = 0;
     xPl = 0;
     yPl = 209;
     state = 0;
     turn = 0;
     mapeo_SD("menu.txt", 32, 25, 256);
   }
 }
```

Y al final del código para el estado uno se tiene la sección donde se actualiza la posición de la bala del jugador y se revisa si se sale del mapa.

```
//*****************************************************************************************************************************************
//Player #1 bullet direction
//*****************************************************************************************************************************************
  if(triggerPl==1 && PlbullDirection == 1){//Pl bullet moving up
    LCD_Bitmap(xPlbullet, yPlbullet, 10, 10, bulletUp);
    H_line(xPlbullet, yPlbullet+10, 10, 0x00);
    yPlbullet--;
    LCD_Bitmap(xPlbullet, yPlbullet, 10, 10, bulletUp);
    H_line(xPlbullet, yPlbullet+10, 10, 0x00);
    yPlbullet--;
    if(yPlbullet <= 0){
      for(int i = 10; i >= 0; i--){
        H_line(xPlbullet, yPlbullet + i, 10, 0x00);
      }
      triggerPl = 0;
      xPlbullet = -10;
      yPlbullet = -10;
    }
  }else if(triggerPl==1 && PlbullDirection == 2){//Pl bullet moving right
    LCD_Bitmap(xPlbullet, yPlbullet, 10, 10, bulletRight);
    V_line(xPlbullet-1, yPlbullet, 10, 0x00);
    xPlbullet++;
    LCD_Bitmap(xPlbullet, yPlbullet, 10, 10, bulletRight);
    V_line(xPlbullet-1, yPlbullet, 10, 0x00);
    xPlbullet++;

    if((xPlbullet+10) >= 320){
      for(int i = 0; i <= 11; i++){
        V_line(xPlbullet+9-i, yPlbullet, 10, 0x00);
      }
      triggerPl = 0;
      xPlbullet = -10;
      yPlbullet = -10;
    }
  }else if(triggerPl==1 && PlbullDirection == 3){//Pl bullet moving left
    LCD_Bitmap(xPlbullet, yPlbullet, 10, 10, bulletLeft);
    V_line(xPlbullet+10, yPlbullet, 10, 0x00);
    xPlbullet--;
    LCD_Bitmap(xPlbullet, yPlbullet, 10, 10, bulletLeft);
    V_line(xPlbullet+10, yPlbullet, 10, 0x00);
    xPlbullet--;

     if(xPlbullet <= 0){
       for(int i = 0; i <= 11; i++){
         V_line(xPlbullet+i, yPlbullet, 10, 0x00);
       }
       triggerPl = 0;
       xPlbullet = -10;
       yPlbullet = -10;
     }
  }else{
     triggerPl = 0;
     xPlbullet = -10;
     yPlbullet = -10;
  }
```

Luego de esto se repite el ciclo hasta que el jugador gana o muere o uno de los enemigos llegan al final.

Para el modo de dos jugadores se tiene que elegir el modo de dos jugadores. El cual es exactamente el mismo código, pero todas las comparaciones entre jugador-enemigo, bala jugador-enemigo, bala enemigo-jugador se realizan dos veces porque ahora hay un nuevo conjunto de variables para el jugador 2. Por ejemplo, la comparación entre el enemigo 1 y su posición ahora se vería así:

```
if((enemyly == 209)||((xP1+29)>enemylx && (xP1+1)<(enemylx+28) && yP1<(enemyly+30) && (yP1+30)>enemyly)
||((xP2+29)>enemylx && (xP2+1)<(enemylx+28) && yP2<(enemyly+30) && (yP2+30)>enemyly)){
  enemylLife = 1;
  P1Life = 1;
  P2Life = 1;
  LCD_Sprite(xP1, yP1, 30, 30, dead, 2, 1, 0, 0);
  LCD_Sprite(xP2, yP2, 30, 30, dead, 2, 1, 0, 0);
  mapeo_SD("gameOver.txt", 67, 75, 186);
  delay(3000);
  LCD_Clear(0x00);
  enemylLife = 0;
  enemyly = 0;
  enemy2y = 0;
  enemy3y = 0;
  enemy4y = 0;
  enemy5y = 0;
  enemy6y = 0;
  enemy7y = 0;
  enemy8y = 0;
  enemy9y = 0;
  enemy10y = 0;
  enemy11y = 0;
  enemy12y = 0;
  enemy13y = 0;
  enemy14y = 0;
  enemy15y = 0;
  enemy16y = 0;
  enemy17y = 0;
  enemy18y = 0;
  enemy19y = 0;
  enemy20y = 0;
  enemy21y = 0;
  enemy22y = 0;
  enemy23y = 0;
  tripleWave1 = 0;
  tripleWave2 = 0;
  tripleWave3 = 0;
  tripleWave4 = 0;
  tripleWave5 = 0;
  P1Life = 0;
  P2Life = 0;

  xP1 = 0;
  yP1 = 209;
  xP2 = 290;
  yP2 = 209;
  state = 0;
  turn = 0;
  mapeo_SD("menu.txt", 32, 25, 256);
}
```

Solo se duplicaron las comparaciones en el if y se hicieron con las coordenadas del jugador 2. Entonces el ciclo es el mismo hasta que todos los enemigos mueren o uno de los dos jugadores mueren o uno de los enemigos llega al final y luego se regresa al menú de inicio.

En este caso para la imagen que se usa en el menú y la imagen de “Game Over” se usó la SD. Se convirtió ambas imágenes con el lcd_image_converter y se guardaron dentro de un .txt. Con la función mapeo_SD se lee línea por línea del .txt y se grafica como si fuera la función para bitmaps.

```
void mapeo_SD(char doc[], int x, int y, int w){
  myFile = SD.open(doc, FILE_READ);
  int hex1 = 0;
  int val1 = 0;
  int val2 = 0;
  int mapear = 0;
  int vertical = 0;
  unsigned char maps[w*2];

  if(myFile){
    while(myFile.available()){
      mapear = 0;
      while(mapear < w*2){
        hex1 = myFile.read();
        if(hex1 == 120){
          val1 = myFile.read();
          val2 = myFile.read();
          val1 = ascii2hex(val1);
          val2 = ascii2hex(val2);
          maps[mapear] = val1*16 + val2;
          mapear++;
        }
      }
      LCD_Bitmap(x, vertical+y, w, 1, maps);
      vertical++;
    }
    myFile.close();
  }else{
    Serial.println("No se puedo abrir la imagen");
    myFile.close();
  }
}
```

Solo hay que indicarle el nombre del archivo, las coordenadas xy y el ancho de la imagen.