MEMORIA JUEGO NDS

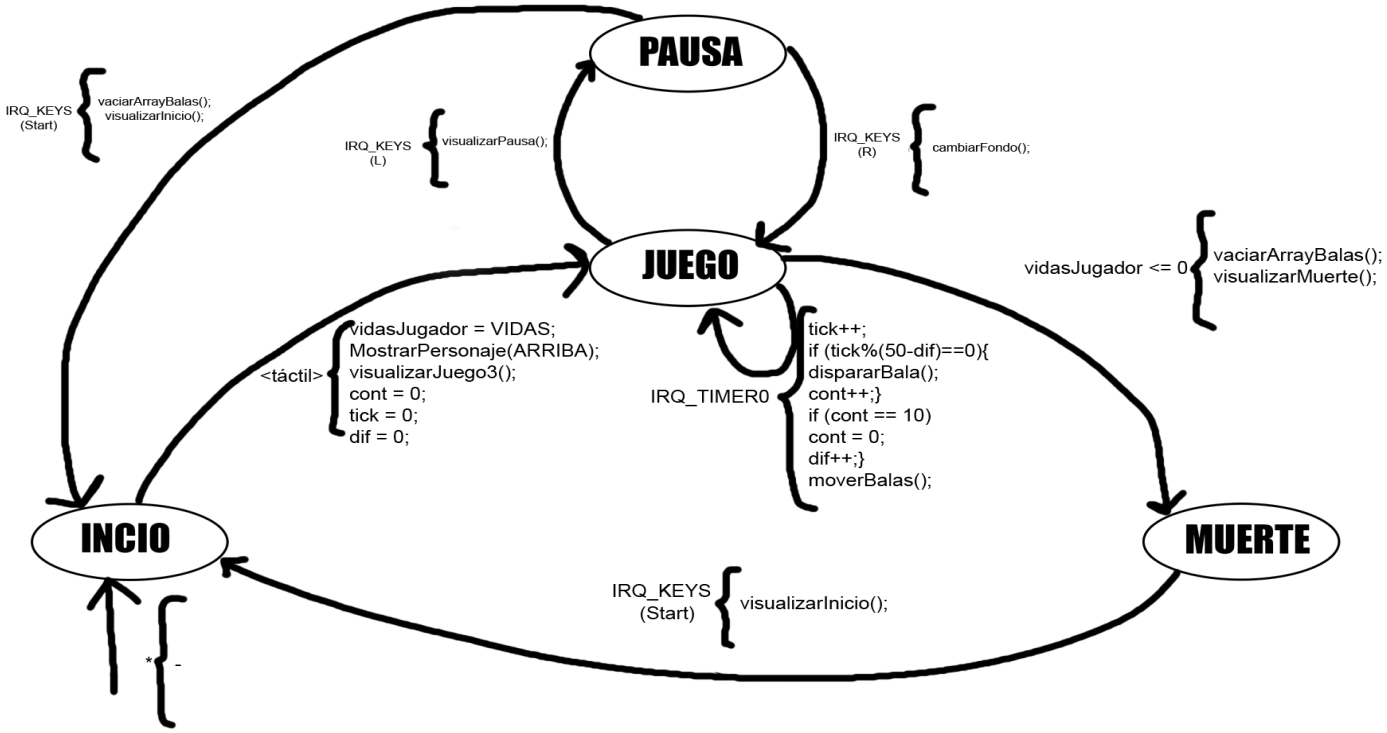
Agustin Garcia Cifuentes

ÍNDICE

INTRODUCCIÓN

Este proyecto hecho en C se ha realizado para la asignatura Estructura de Computadores 22-23 de Ingeniería Informática en la Facultad de Informática de San Sebastián. Se han aplicado los conceptos aprendidos a lo largo del tema 3 en dicha asignatura, poniendo en práctica las nociones de dispositivos de entrada/salida, además del funcionamiento del hardware de una Nintendo DS.

AUTÓMATA



DESCRIPCIÓN DEL JUEGO

Idea

He planteado este proyecto para que sea una recreación simple de uno de los minijuegos que se presentan en el videojuego *Undertale*, específicamente en la batalla contra *Undyne*. El jugador controla un corazón y debe colocar un escudo o barrera contra las flechas

dirigidas a este. Este minijuego aumenta la frecuencia y velocidad de las flechas con el tiempo, además de presentar algunos proyectiles especiales que no se verán en este proyecto.
Al aplicar esta mecánica a mi juego, he replanteado la finalidad para que sea aguantar lo máximo posible y de esta forma acumular la mayor puntuación. Cada vez que el jugador sea alcanzado por un proyectil, su número de *vidas* descenderá y la partida acabará cuando este valor llegue a 0

Controles

Por interrupción:

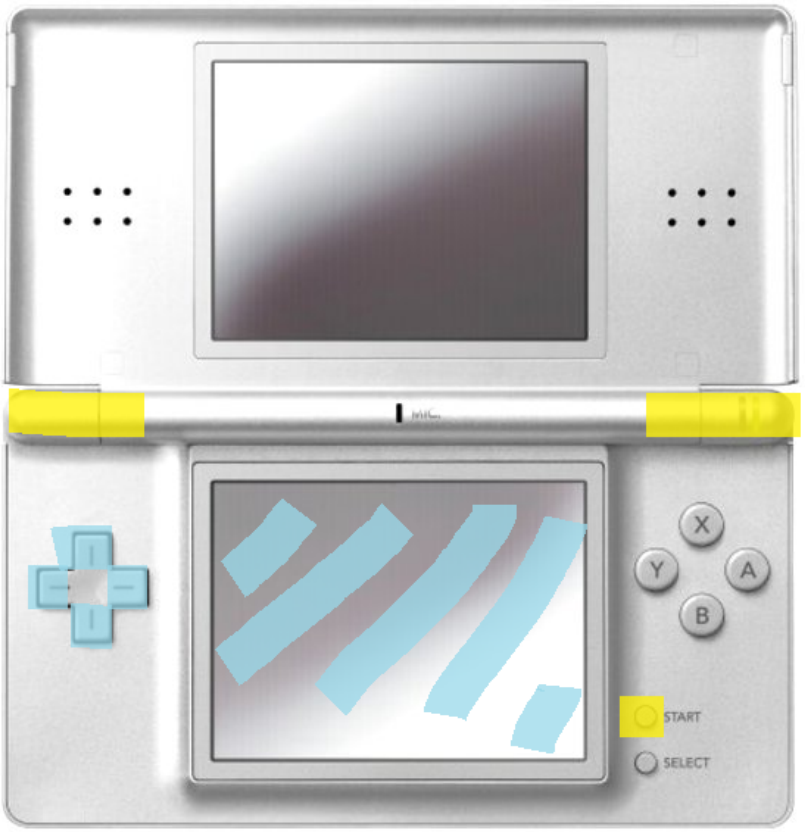
- L: Pausar el juego desde Juego
- R: Volver al juego desde Pausa
- START: Ir al menú desde *Pausa* / *Muerte*

Por encuesta:

- ARRIBA: Escudo arriba
- ABAJO: Escudo abajo
- IZQUIERDA: Escudo izquierda
- DERECHA: Escudo derecha
- TACTIL: Empezar la partida

No utilizadas

- A
- B
- X
- Y
- SELECT



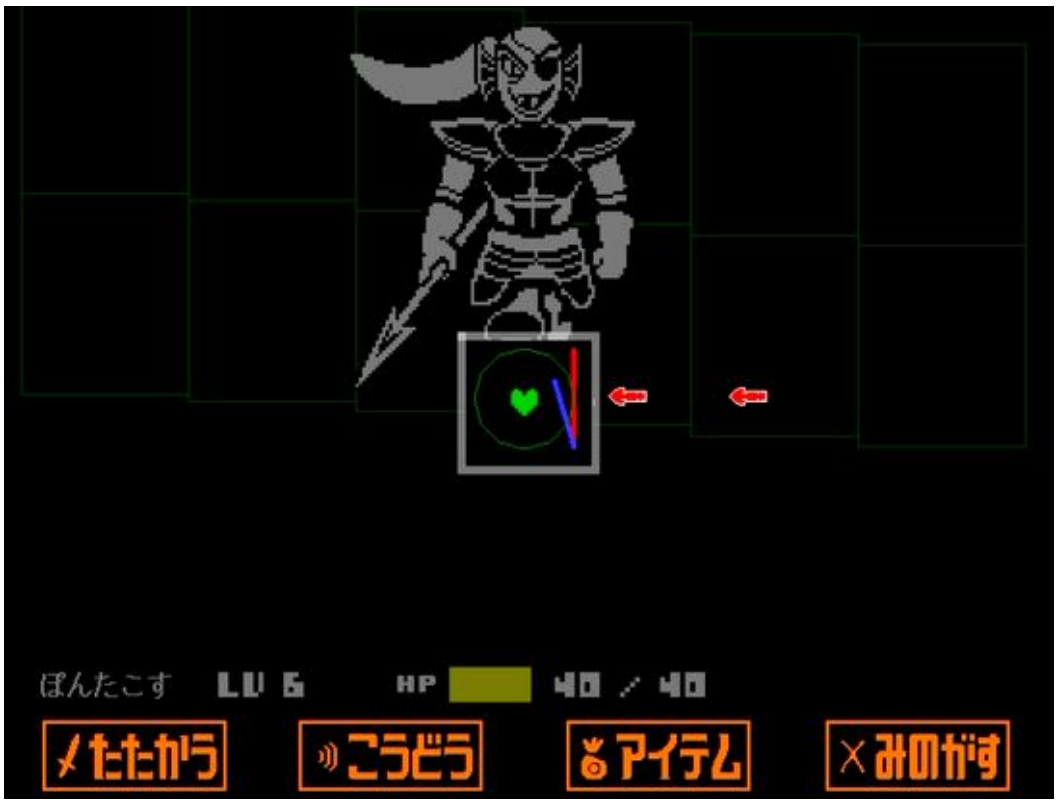
Dificultad

La dificultad del juego consiste en que cada vez las balas irán apareciendo con más frecuencia y se aproximarán al jugador con más velocidad. Este sistema se explicará más adelante en la rutina de atención del temporizador.

Inspiraciones

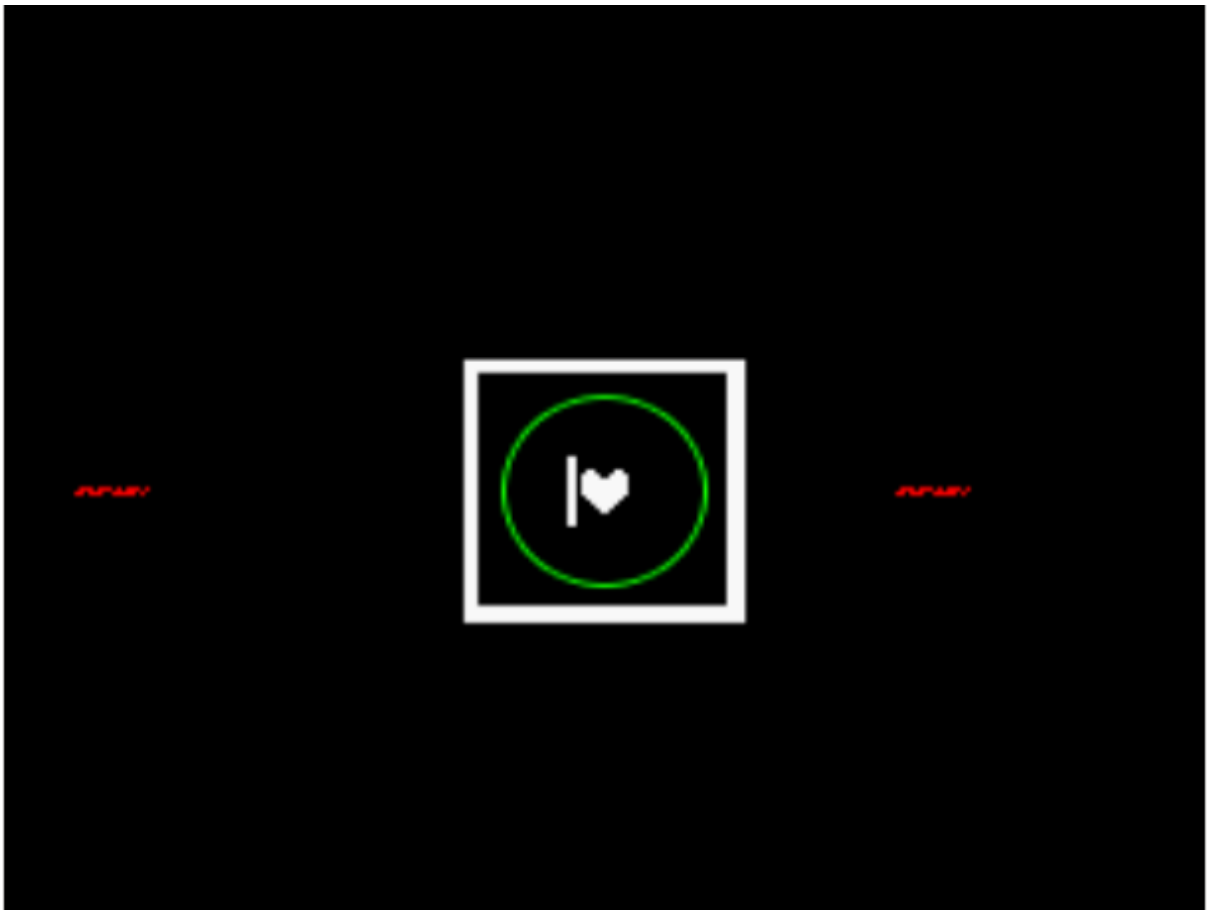
Como se ha mencionado en la introducción, este juego está altamente inspirado en *Undertale* y su minijuego contra el personaje *Undyne*. Se pueden apreciar las similitudes en el apartado artístico.

Juego original



Recreación del proyecto


UNDYNETALE
Pulsa la pantalla para empezar



DESARROLLO

Árbol de directorios

- build: Archivos compilados
- gfx: Recursos visuales y gráficos
    - Puerta.png
    - PuertaAbierta.png
    - Inicio.png
    - Juego1.png
    - Juego2.png
    - Juego3.png
    - Muerte.png
    - Pausa.png
- include: Interfaces de las clases / métodos abstractos de las clases
    - definiciones.h
    - fondos.h
    - graficos.h
    - juego.h
    - perifericos.h
    - rutinas.h
    - rutinasAtencion.h
    - sprites.h
- source: Clases con métodos implementados.
    - fondos.c
    - graficos.c
    - juego.c
    - main.c
    - perifericos.c
    - rutinas.c
    - rutinasAtencion.c
    - sprites.c

Variables globales

Estas están definidas en */include/definiciones.h*
Sólo se comentarán las añadidas a las existentes en la plantilla.

Los diferentes estados del juego

```
1  #define INICIO 0
2  #define JUEGO 1
3  #define MUERTE 2
4  #define PAUSA 3
```

El máximo posible de proyectiles en pantalla a la vez

```
1  #define MAX_BALAS 1
```

El objeto tipo Bala con su índice identificador, posición, si está presente o no y su ubicación

```
1  struct Bala
2  {
3      int index;
4      int posX;
5      int posY;
6      int viva;
7      int ubi;
8  };
```

La última orientación del jugador

```
1  extern int ultimaPos;
```

Una colección de balas

```
1  extern struct Bala balas[MAX_BALAS];
```

Las vidas del jugador

```
1  extern int vidasJugador;
```

La posición en pantalla del jugador

```
1  extern int X; //PlayerX
2  extern int Y;//PlayerY
```

La “dificultad” del juego

```
1  extern int dif;
```

La contador para aumentar la dificultad del juego

```
1  extern int cont;
```

La cuenta de ticks

```
1  extern int tick;
```

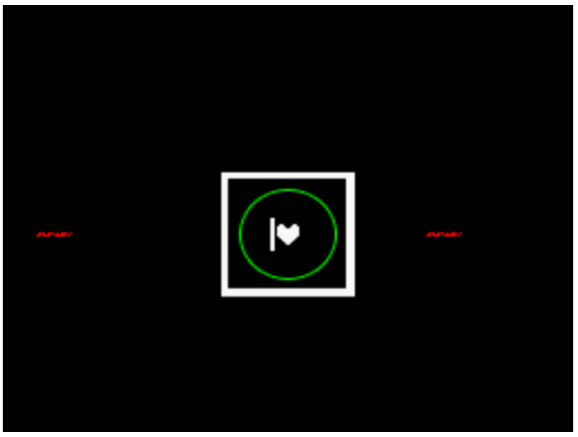
Estados del juego

INICIO: 0

Este es el estado mostrado al abrir el juego, el jugador será bienvenido con una pantalla indicando que debe pulsar la pantalla táctil para comenzar a jugar.

JUEGO: 1

Este es el estado donde transcurre la partida, aquí se hace la encuesta de las teclas de dirección.



MUERTE: 2

Si durante el estado anterior, las vidas del jugador se agotan, pasará a este estado y será avisado mediante un texto en la pantalla superior, indicando que debe pulsar start para volver al menú (Estado inicial)

PAUSA: 3

Este estado se da si el jugador interrumpe la partida pulsando el botón L. El juego se parará por completo hasta que el jugador pulse el botón R, donde proseguirá donde lo dejó.

Gestión de interrupciones

Desde el programa principal, inicialmente se habilitan todas las interrupciones

```
ConfigurarTeclado(0×400C); // Configurar el teclado.
ConfigurarTemporizador( 56798,  0×00C1); // Configurar el temporizador.
HabilitarIntTeclado(); // Habilitar las interrupciones del teclado.
HabilitarInterrupciones();
HabilitarIntTempo(); // Habilitar las interrupciones del temporizador.
EstablecerVectorInt(); // Habilitar interrupciones.
```

Teclado

Se le indica al controlador del teclado que debe gestionar las teclas L,R y START por interrupción (bits 3,4 y 16).
También se implementa lo que debe hacer cada tecla en función del estado en el que se encuentre el juego, este funcionamiento se puede seguir desde el autómata del juego

```
void RutAtencionTeclado ()
{
    int tecla = TeclaPulsada();
    switch (tecla)
    {
    case START:
        if (ESTADO == PAUSA)
        {
            consoleClear();
            vaciarArrayBalas();
            visualizarInicio();
            ESTADO = INICIO;
        } else if(ESTADO == MUERTE){
            consoleClear();
            visualizarInicio();
            ESTADO = INICIO;
        } break;

    case L:
        if (ESTADO == JUEGO)
        {
            consoleClear();
            iprintf("\x1b[01;00HPAUSA");
            iprintf("\x1b[02;00HPULSA R PARA SEGUIR O START PARA VOLVER AL MENU");
            visualizarPausa();
            ESTADO = PAUSA;
        }
        break;

    case R:
        if (ESTADO == PAUSA)
        {
            consoleClear();
            iprintf("\x1b[04;00HVidas: %d",vidasJugador);
            iprintf("\x1b[05;05H——CONTROLES——");
            iprintf("\x1b[06;00HTeclas de direccion: dirigir escudo");
            iprintf("\x1b[007;00HL: PAUSA");
            cambiarFondo();
            ESTADO = JUEGO;
        } break;
    }
}
```

Reloj

Desde esta parte del código se controlan varios aspectos fundamentales del funcionamiento, tales como: el cambio de fondos, el movimiento y aparición de las balas y la puntuación del juego.

Los fondos del estado juego se cambian desde aquí, porque es aquí donde se buscan las colisiones cada vez que se mueven las balas, de esta forma el cambio se hace cada vez que hay una colisión y no cada vez que se entra al estado==JUEGO en el juego(); optimizando mejor los recursos y mejorando el rendimiento del juego.

Desde aquí se gestiona la aparición de balas y su desplazamiento.

Se ha seguido la recomendación de configuración del documento guía del proyecto, de modo que el latch se ha configurado mediante la siguiente fórmula:

Se busca que el latch actúe 5 tics / segundo, de modo que como parámetro X se ha escogido 256, el cual da un latch de 39321

En el controlador, se han activado el bit 7 y 6 para activar el temporizador y las interrupciones, y en los bit 0 y 1 se ha colocado el decimal 2 (10) para dividir la frecuencia entre el parámetro anteriormente mencionado.

```
1  void RutAtencionTempo()
2  {
3      if (ESTADO==JUEGO)
4      {
5          tick++;
6
7          if (tick%(50-dif)==0) //La frecuencia de aparición de balas
8          {
9              dispararBala();
10             cont++;
11         }
12         if (cont == 10)
13         {
14             cont = 0;
15             dif++;
16         }
17         moverBalas();
18     }
19 }
```

Pantalla táctil

La pantalla táctil es usada sólamente en el estado inicial del juego. Al pulsarla se dará inicio a la partida

```
1  int tactilTocada(){
2      touchPosition pos_Pantalla;
3      touchRead(&pos_Pantalla);
4      return !(pos_Pantalla.px==0 && pos_Pantalla.py==0);
5  }
```

Se comprueba mediante encuesta en la función principal del juego, desde el estado de INICIO

```
if(tactilTocada()){
    consoleClear();
    MostrarPersonaje(ARRIBA);
    vidasJugador = VIDAS;
    iprintf("\x1b[04;00HVidas: %d",vidasJugador);
    iprintf("\x1b[06;07H———CONTROLES———");
    iprintf("\x1b[07;00HTeclas de direccion: dirigir escudo");
    iprintf("\x1b[08;00HL: PAUSA");
    visualizarJuego3();
    cont = tick = dif = 0;
    ESTADO = JUEGO;
}
```

Sprites

Existe un sprite para cada orientación del jugador, además de dos sprites para los proyectiles, uno para los del eje horizontal y otro para los del eje vertical.

No se ha creado un sprite para cada orientación en los proyectiles con el fin de ahorrar memoria y recursos.

Jugador derecha

0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3
0	0	0	0	3	3	0	0	0	0	3	3	0	0	3	3
0	0	0	3	3	3	3	0	0	3	3	3	3	0	3	3
0	0	0	3	3	3	3	3	3	3	3	3	3	0	3	3
0	0	0	3	3	3	3	3	3	3	3	3	3	0	3	3
0	0	0	3	3	3	3	3	3	3	3	3	3	0	3	3
0	0	0	3	3	3	3	3	3	3	3	3	3	0	3	3
0	0	0	0	3	3	3	3	3	3	3	3	0	0	3	3
0	0	0	0	0	3	3	3	3	3	3	0	0	0	3	3
0	0	0	0	0	0	3	3	3	3	0	0	0	0	3	3
0	0	0	0	0	0	0	3	3	0	0	0	0	0	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3

Jugador abajo

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	3	3	0	0	0	0	3	3	0	0	0	0
0	0	0	3	3	3	3	0	0	3	3	3	3	0	0	0
0	0	0	3	3	3	3	3	3	3	3	3	3	0	0	0
0	0	0	3	3	3	3	3	3	3	3	3	3	0	0	0
0	0	0	3	3	3	3	3	3	3	3	3	3	0	0	0
0	0	0	3	3	3	3	3	3	3	3	3	3	0	0	0
0	0	0	0	3	3	3	3	3	3	3	3	0	0	0	0
0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
0	0	0	0	0	0	3	3	3	3	0	0	0	0	0	0
0	0	0	0	0	0	0	3	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

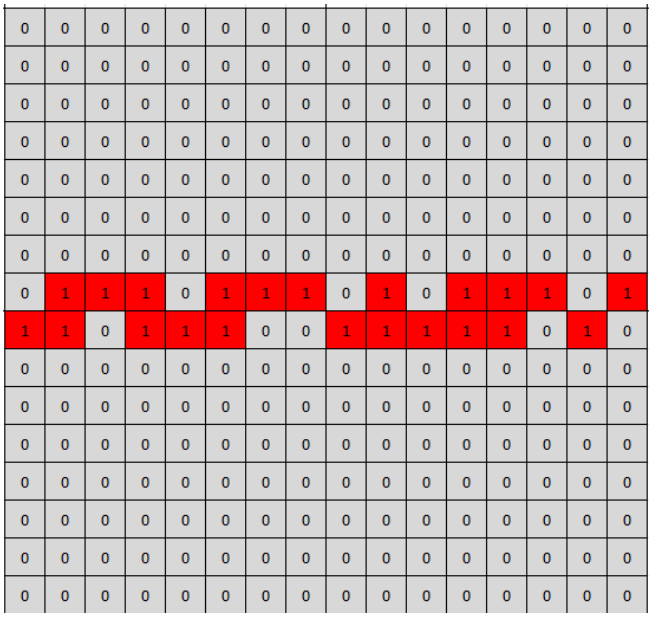
Jugador izquierda

3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	3	0	0	3	3	0	0	0	0	3	3	0	0	0	0
3	3	0	3	3	3	3	0	0	3	3	3	3	0	0	0
3	3	0	3	3	3	3	3	3	3	3	3	3	0	0	0
3	3	0	3	3	3	3	3	3	3	3	3	3	0	0	0
3	3	0	3	3	3	3	3	3	3	3	3	3	0	0	0
3	3	0	3	3	3	3	3	3	3	3	3	3	0	0	0
3	3	0	0	3	3	3	3	3	3	3	3	0	0	0	0
3	3	0	0	0	3	3	3	3	3	3	0	0	0	0	0
3	3	0	0	0	0	3	3	3	3	0	0	0	0	0	0
3	3	0	0	0	0	0	3	3	0	0	0	0	0	0	0
3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0

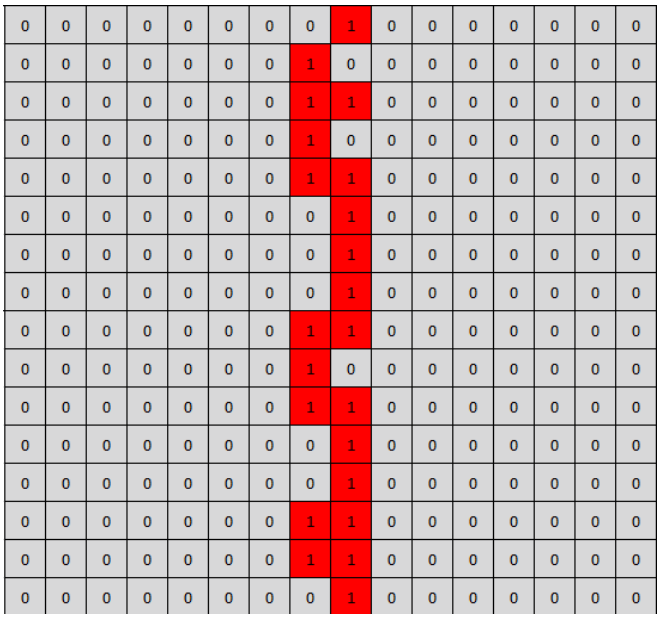
Jugador arriba

3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	3	3	0	0	0	0	3	3	0	0	0	0
0	0	0	3	3	3	3	0	0	3	3	3	3	0	0	0
0	0	0	3	3	3	3	3	3	3	3	3	3	0	0	0
0	0	0	3	3	3	3	3	3	3	3	3	3	0	0	0
0	0	0	3	3	3	3	3	3	3	3	3	3	0	0	0
0	0	0	3	3	3	3	3	3	3	3	3	3	0	0	0
0	0	0	0	3	3	3	3	3	3	3	3	0	0	0	0
0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
0	0	0	0	0	0	3	3	3	3	0	0	0	0	0	0
0	0	0	0	0	0	0	3	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Proyectil horizontal



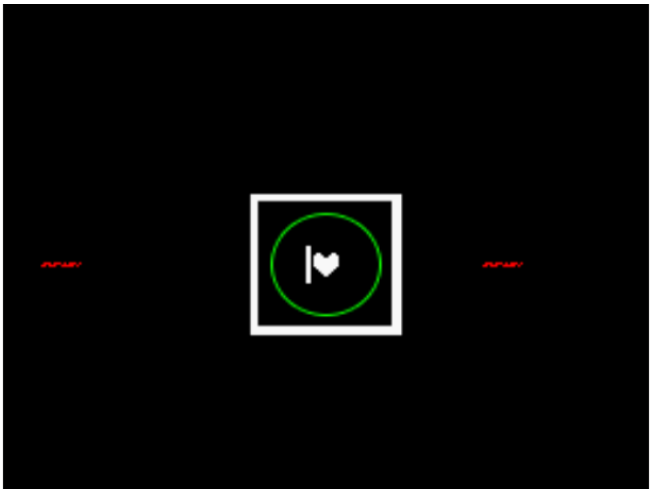
Proyectil vertical



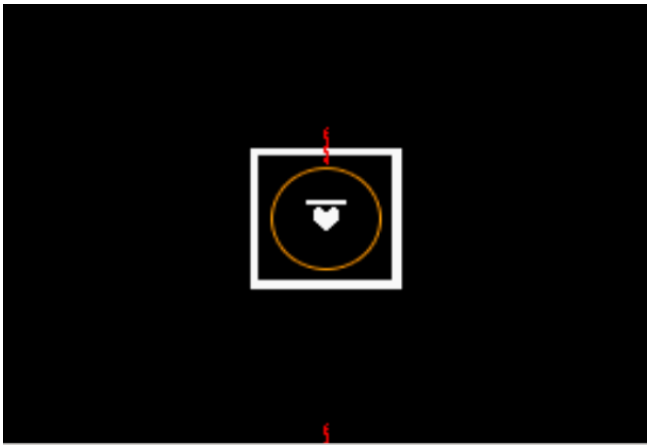
Fondos

Hay un fondo para cada estado del juego, estos se pueden apreciar en el apartado de estados, Sin embargo, el estado Juego, con el fin de implementar la mecánica de la vida de una forma original, el estado JUEGO cuenta con 3 posibles fondos; uno para cada vida.

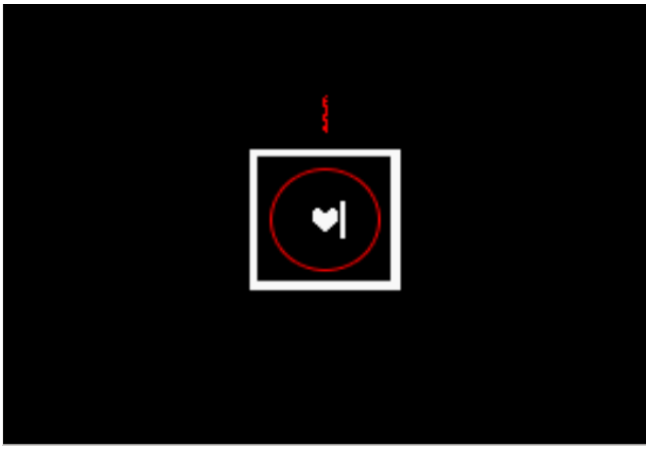
3 Vidas



2 Vidas



1 Vida



Como se puede apreciar, el círculo verde decorativo del juego original, es aprovechado aquí como indicador de la vida del jugador, de esta forma no es necesario levantar la vista a la pantalla superior y dar pie a perder una vida por no prestar atención.

Rutinas propias

Se han desarrollado diferentes métodos para el funcionamiento del juego, estas se encuentran en rutinas.c

```
1  void BorrarPersonaje();
2  void MostrarPersonaje(int pos);
3  void dispararBala();
4  void cambiarFondo();
5  void vaciarArrayBalas();
6  void moverBalas();
```

BorrarPersonaje()

Esta rutina simplemente ejecuta el método borrar de todas las orientaciones del personaje, eliminandolo de forma segura de la pantalla.

```
1  void BorrarPersonaje(){
2      BorrarArriba(PlayerID,X,Y);
3      BorrarAbajo(PlayerID,X,Y);
4      BorrarDerecha(PlayerID,X,Y);
5      BorrarIzquierda(PlayerID,X,Y);
6  }
```

MostrarPersonaje(int)

Muestra al personaje en la orientación del parámetro siendo:

- 0: Arriba/Norte
- 1: Abajo/Sur
- 2: Derecha/Oeste
- 3: Izquierda/Este

```
void MostrarPersonaje(int pos){
    switch (pos)
    {
    case ARRIBA:
        MostrarArriba(PlayerID, X, Y);
        break;

    case ABAJO:
        MostrarAbajo(PlayerID, X, Y);
        break;

    case DERECHA:
        MostrarDerecha(PlayerID, X, Y);
        break;

    case IZQUIERDA:
        MostrarIzquierda(PlayerID, X, Y);
        break;
    }
}
```

vaciarArrayBalas()

Recorre todo el array de balas colocando cada posición a 0 y ejecutando el método BorrarProy.

```
void vaciarArrayBalas(){
    int j;
    for (j = 0; j < MAX_BALAS; j++)
    {
        balas[j].viva = 0;
        if (balas[j].ubi == ARRIBA || balas[j].ubi == ABAJO)
        {
            BorrarProyV(j,balas[j].posX,balas[j].posY);
        } else{
            BorrarProyH(j,balas[j].posX,balas[j].posY);
        }
    }

}
```

dispararBalas()

Este método se usa para generar proyectiles en pantalla. Primero se genera un número random del 0 al 3 incluido para determinar su orientación/ubicación, tras esto se busca la primera posición libre en el array de balas cuyo proyectil no haya sido disparado, tras esto se coloca a 1 su vida y se inicializan sus coordenadas y se ejecuta el método MostrarProy

```
void dispararBala(){
    int i = 0;
    int j = 0;

    int r = rand() % 4;
    while (j < 30)
        {
            if (!balas[j].viva)
            {
                i = j;
                break;
            }
            j++;
        }
    int random = r+4;
    balas[i].ubi=random;
    switch (random)
        {
            case ARRIBA: //Bala arriba
                balas[i].posX = 120;
                balas[i].posY = 0;
                MostrarProyV(i,balas[i].posX,balas[i].posY);
                break;
            case ABAJO: //Bala abajo
                balas[i].posX = 120;
                balas[i].posY = 191;
                MostrarProyV(i,balas[i].posX,balas[i].posY);
                break;
            case DERECHA: //Bala derecha
                balas[i].posX = 255;
                balas[i].posY = 88;
                MostrarProyH(i,balas[i].posX,balas[i].posY);
                break;
            case IZQUIERDA: //Bala izquierda
                balas[i].posX = 0;
                balas[i].posY = 88;
                MostrarProyH(i,balas[i].posX,balas[i].posY);
                break;
            default:
                break;
        }
    balas[i].viva = 1;
}
```

moverBalas()

El código que se ejecuta cuando se quieren mover las balas en pantalla. Recorre todo el array de balas y entrará al algoritmo sólo si su atributo “viva” es 1; lee su ubicación en pantalla y primero verifica si hay colisión con el jugador; si existe, si corresponde a un bloqueo o impacto. Si no hay colisión, hace avanzar a la bala modificando las coordenadas.

```
void moverBalas(){
    int i;
    for(i = 0; i<MAX_BALAS; i++)
        {
            if (balas[i].viva)
            {
                if (balas[i].ubi==ARRIBA) //ARRIBA → y++
                {
                    if (ultimaPos==ARRIBA && balas[i].posY==70) //el jugador ha bloqueado el proyectil
                    {
                        balas[i].viva = 0;
                        BorrarProyV(i,balas[i].posX,balas[i].posY);
                    }
                    else if (ultimaPos≠ARRIBA && balas[i].posY > 70) //La bala le ha dado al jugador==No estaba bloqueandola
                    {
                        vidasJugador--;
                        balas[i].viva = 0;
                        BorrarProyV(i,balas[i].posX,balas[i].posY);
                        cambiarFondo();
                    }
                    else //La bala sigue su recorrido
                    {
                        balas[i].posY++;
                        MostrarProyV(i,balas[i].posX,balas[i].posY);
                    }
                }
                if (balas[i].ubi==ABAJO) //ABAJO → y--
                {
                    if (ultimaPos==ABAJO && balas[i].posY==105) //el jugador ha bloqueado el proyectil
                    {
                        balas[i].viva = 0;
                        BorrarProyV(i,balas[i].posX,balas[i].posY);
                    }
                    else if (ultimaPos≠ABAJO && balas[i].posY < 105) //La bala le ha dado al jugador==No estaba bloqueandola
                    {
                        vidasJugador--;
                        balas[i].viva = 0;
                        BorrarProyV(i,balas[i].posX,balas[i].posY);
                        cambiarFondo();
                    }
                    else //La bala sigue su recorrido
                    {
                        balas[i].posY--;
                        MostrarProyV(i,balas[i].posX,balas[i].posY);
                    }
                }
                if (balas[i].ubi==DERECHA) //DERECHA → x--
                {

                    if (ultimaPos==DERECHA && balas[i].posX==137) //el jugador ha bloqueado el proyectil
                    {
                        balas[i].viva = 0;
                        BorrarProyH(i,balas[i].posX,balas[i].posY);
                        cambiarFondo();
                    }
                    else if (ultimaPos≠DERECHA && balas[i].posX < 137) //La bala le ha dado al jugador==No estaba bloqueandola
                    {
                        vidasJugador--;
                        balas[i].viva = 0;
                        BorrarProyH(i,balas[i].posX,balas[i].posY);
                        cambiarFondo();
                    }
                    else //La bala sigue su recorrido
                    {
                        balas[i].posX--;
                        MostrarProyH(i,balas[i].posX,balas[i].posY);
                    }

                }
                if (balas[i].ubi==IZQUIERDA) //IZQUIERDA → x++
                {
                    if (ultimaPos==IZQUIERDA && balas[i].posX==103) //el jugador ha bloqueado el proyectil
                    {
                        balas[i].viva = 0;
                        BorrarProyH(i,balas[i].posX,balas[i].posY);
                    }
                    else if (ultimaPos≠IZQUIERDA && balas[i].posX > 103) //La bala le ha dado al jugador==No estaba bloqueandola
                    {
                        vidasJugador--;
                        balas[i].viva = 0;
                        BorrarProyH(i,balas[i].posX,balas[i].posY);
                        cambiarFondo();
                    }
                    else //La bala sigue su recorrido
                    {
                        balas[i].posX++;
                        MostrarProyH(i,balas[i].posX,balas[i].posY);
                    }

                }
                iprintf("\x1b[04;00HVidas: %d",vidasJugador);
                if (vidasJugador ≤ 0) //El jugador ha perdido la partida
                {
                    vaciarArrayBalas();
                    consoleClear();
                    visualizarMuerte();
                    ESTADO = MUERTE;
                }

            }
        }

}

```

cambiarFondo()

Este método lee la vida del jugador y coloca el fondo de juego correspondiente a dicho valor.

```
void cambiarFondo(){
    switch (vidasJugador)
        {
        case 3:
            visualizarJuego3();
            break;
        case 2:
            visualizarJuego2();
            break;
        case 1:
            visualizarJuego1();
            break;
        default:
            break;
        }
}
```