MANUAL DEL PROGAMADOR

Este proyecto es una especie de Videojuego Programado en C, especialmente implementado con listas.

El videojuego es un juego clásico de laberintos compaginado con los clásicos juegos de disparos, el juego se desenvuelve en un ambiente de laberintos donde el jugador lo tiene que recorrer recolectando cosas y eliminando a los Enemigos.

Librerías Usadas

#include <stdio.h>

#include <iostream>

#include <graphics.h>

#include <time.h>

Constante usada para el tamaño del nodo de la Malla

#define TAM 45 //45tamaño del nodo de la Malla

Constante Usada para el número de Renglones en la Malla

#define NREN 14 //NUMERO DE NODOS QUE CONTENDRA LA MALLA (controla los ciclos for que crean la malla)

Constante Usada para el número de columnas de la Malla

#define NCOL 23

Constante Usada para el número de botones del menú

#define NUMBOTONES 4

Constante usada para la implementación del menú con mouse sirve para checar si se tocó o no el Botón

#define PRE 0

Constante Usada para la Implementación del menú con mouse usada para reiniciar el botón una vez que se le dio clic

#define NORMAL 1

Estructura utilizada para implementar el Menú con mouse usando un control mediante botones.

typedef struct

{

int x, y,num;

int tipo,tamx,tamy;

char texto[50];

}TBoton;

Esta estructura sirve para implementa un nodo de la malla se puede apreciar con claridad los diversos campos que contiene el registro.

typedef struct nodo

{

char info;

int existe;

int cor\_x;

int cor\_y;

struct nodo \*sig,\*ant;

struct nodo \*aba,\*arr;

struct jug \*jugador;

struct enemi \*enemigo;

}\*tpnodo;

Esta estructura sirve para la implementacion de las balas en el juego, dicha estructura es imprecindible que contenga el campo tpnodo malla.

typedef struct regbala

{

int x;

int y;

int direccion;

tpnodo nodo\_malla;

struct regbala \*ant;

struct regbala \*sig;

}\*BALA;

Estructura que sirve para la implementacion de una lista doble de balas.

typedef struct reg\_list\_doble\_bala

{

BALA inicio;

BALA final;

}\*MUNICION;

Estructura que implementa para el manejo del jugador.

typedef struct jug

{

int direccion;

tpnodo nod\_malla;

int nivel;

char nombre[20];

int muertos;

int vida;

int puntos;

MUNICION balas;

}JUGADOR;

Estructura para el manejo de los enemigos

//ESTRUCTURAS PARA LOS ENEMIGOS

typedef struct enemi

{

int x,y;

tpnodo nodo\_malla;

struct enemi \*ant,\*sig;

int band\_dir;

}\*ENEMIGO;

Con esta estructura se implementa una lista doble de Enemigos.

typedef struct malos

{

ENEMIGO inicio;

ENEMIGO final;

}\*LISTA\_ENEMIGOS;

Estructura que contiene los campos que servirán para el dibujado de los iconos básicamente guarda las dimensiones en X y Y, se define un arreglo bidimensional para la implementación delas coordenadas de los pixeles en el dibujado.

typedef struct

{

int dimx,dimy;

int \*\*img;

}imagen;//estructura de imagen guarda las dimensiones x,y de cada imagen

Estructura que contiene campos tipo imagen fue usada para el uso de cada uno de los iconos en el juego.

typedef struct

{

imagen maton;

imagen maloso;

imagen muro2,maton\_iz,maton\_aba,maton\_up;

imagen bala\_der,bala\_iz,bala\_aba,bala\_arr;

imagen explosion;

imagen muerto;

}imagenes;//Estructura que contiene las imagenes usadas para el juego

Estructura que fue usada como un registro que involucraba el uso de todos los enemigos en el juego.

typedef struct

{

imagenes enem;

LISTA\_ENEMIGOS E;

}ENEMIGOS;

Declaracion de Funciones.

void portada();

void game\_over(char nombre[], int pts);

void guardaRegistro(char nomarch[],JUGADOR r);

void leeRegistros(char nomarch[], JUGADOR registros[NR]);

void info\_jug(JUGADOR jugador);

void libera\_mem(tpnodo \*cab);

void crea\_nodo\_malla(tpnodo \*nodo,int x, int y);

void crea\_malla(tpnodo \*cab,int x,int y);

void dibuja\_malla(tpnodo lista);

char tecla();

void inicializar\_jugador(JUGADOR \*jugador);

void inicializar\_enemigo(ENEMIGOS \*Ens);

void crea\_enemigo(ENEMIGOS \*Ens, tpnodo cab);

int crea\_nodo\_ENEMIGO(ENEMIGO \*nodo,int x, int y);

void libera\_lista\_malosos(LISTA\_ENEMIGOS list);

int juego(tpnodo \*malla,int x,int y, JUGADOR \*jugador,ENEMIGOS \*enemigo,imagenes pics,int \*InoPag);

//FUNCIONES INCOLUCRADAS CON EL FUNCIONAMIENTO DEL JUEGO

void movimientos\_jugador(JUGADOR \*jug,tpnodo \*aux,tpnodo \*ant,tpnodo sig,int dir,int \*bandera\_salida,imagenes pic,tpnodo \*cabecera);

void dibuja\_jugador(int op,imagenes im,tpnodo borra,tpnodo juga);

int crea\_bala(MUNICION parke,tpnodo nodo,int dir);

int crea\_nodo\_BALA(BALA \*nodo,int x,int y);

void movimiento\_disparo\_fase1(clock\_t \*ini,clock\_t fin,JUGADOR \*jugador,ENEMIGOS E,imagenes pics,int \*cont\_muertes);

//void movimiento\_disparo\_fase1(JUGADOR \*jugador,ENEMIGOS E,imagenes pics);

void movimiento\_disparo\_fase2(tpnodo sig,tpnodo \*ant\_malla,BALA \*auxiliar\_disparo,JUGADOR \*jugador,ENEMIGOS E,int dir,imagenes pic,int \*cont\_muertes);

int elimina\_bala(MUNICION parke,BALA nod);

void dibuja\_disparo(int direccion,imagenes im,tpnodo bala\_anterior,int x,int y);

void muere(tpnodo \*aux,tpnodo \*ant, JUGADOR \*ju, tpnodo \*cabecera,int \*tecla,imagenes pic);

int elimina\_maloso(LISTA\_ENEMIGOS enem,ENEMIGO nodo);

void MUEVE(JUGADOR \*jugador,tpnodo \*aux,tpnodo \*aux\_ant,tpnodo sig,imagenes pics,int n);

void moves(JUGADOR \*jugador,tpnodo \*aux,tpnodo \*aux\_ant,char tecla);

int movimientos\_enemigos\_fase2(ENEMIGO \*aux\_maloso, tpnodo \*ant\_enemigo,tpnodo sig,ENEMIGOS \*maloson);

int movimientos\_enemigo\_fase1(clock\_t \*inicio,clock\_t final,ENEMIGOS \*maloso);

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//FUNCION QUE CARGA LOS ESCENARIOS

//void ini\_escenario(tpnodo main\_dor,imagenes pic,int nivel,ENEMIGOS maloso);

void ini\_escenario(tpnodo main\_dor,imagenes pic,int nivel,ENEMIGOS maloso,int \*cont\_enems,int \*cont\_heridos);

//FUNCIIONES INVOLUCRADAS EN LA CREACION DE LAS IMAGENES

void cargaImagenes(imagenes \*img);

void carga\_Imagen\_ENEMIGO(ENEMIGOS \*maloso);

void creaicono(char cad[],imagen \*imag);

int dibujaicono(imagen imag,int x,int y,int tam,int dir);

void ayuda(char nom\_arch[]);

void pedir\_nombre(char nombre[]);

void descartar\_input();

void llenabotones(TBoton z[]);

void llenaboton(int num, int x, int y, char texto[], TBoton \*z);

void dibujabotones(TBoton z[], int w);

void dibujaboton(TBoton z);

int checaboton(TBoton z[], int xm, int ym);

int pintamenu(TBoton Bot[]);

void portada1();

void intextxy( int x, int y, char texto[], int colf, int collet);

int menu2();

A continuación se explicara cada una de las diversas funciones.

**FUNCION MAIN**

Función principal del programa en donde se centra el funcionamiento general del juego aquí se definen e inicializan las diversas variables usadas a lo largo del programa como está trabajando con memoria puramente dinámica hay que tener especial cuidado en el manejo de estas ya que de lo contrario el programa podría colapsar fácilmente.

En esta función se tienen dos Bucles principales, el primero que es el que mantiene el ciclo principal del programa que sirve especialmente para mantener el menú. Según se la opción elegida en el menú que regresara un paramento entero para entrar en switch que servirá para ingresar a casa opción que muestra el menú.

int main()

{

initwindow( getmaxwidth(),getmaxheight(), "ZAMARRIPA RESCUE");

tpnodo inicio=NULL;//Inicializacion del la malla

srand(time(NULL));

int cor\_x=100;

int cor\_y=0;

int res,i;

int InoPag=1;

int band=1;

int op;

char nom[20];

char record[100];

imagenes imags;

JUGADOR jug;

JUGADOR registros[NR];

ENEMIGOS E;

portada();

inicializar\_jugador(&jug);//LISTO

cleardevice();

do{

op=menu2();

///FUNCION PRINCIPAL

switch(op)//parametro switch//OP

{

case 0:

cleardevice();

descartar\_input();

pedir\_nombre(nom);

// pedir\_nombre(jug.nombre);

// strcpy(jug.nombre,nom);

do

{

crea\_malla(&inicio,cor\_x,cor\_y);

inicializar\_enemigo(&E);//CHECAR FUNCION

carga\_Imagen\_ENEMIGO(&E);//LISTO

cargaImagenes(&imags);

setactivepage(InoPag);

cleardevice();

jug.muertos=0;

res=juego(&inicio,cor\_x,cor\_y,&jug,&E,imags,&InoPag);

if(res)

{

if(jug.nivel==4)

{

band=1;

}

else

band=0;

}

else

band=1;

cleardevice();

if(InoPag)

InoPag=0;

else

InoPag=1;

}while(band!=1 && jug.vida>0);

game\_over(nom, jug.puntos);

jug.puntos=0;

jug.nivel=1;

break;

case 1:

descartar\_input();

ayuda("ayuda.txt");

getch();

cleardevice();

break;

case 2:

descartar\_input();

cleardevice();

leeRegistros("records", registros);

moveto(200, 60);

do{

for(i=0; i<5; i++)

{

sprintf(record, "%d) %s %d pts\n", i+1, registros[i].nombre, registros[i].puntos);

settextstyle(1,0,4);

setcolor(WHITE);

outtextxy(((getmaxx()/2\*0)+100),(50)\*i+10,record);

settextstyle(1,0,4);

setcolor(RED);

outtextxy(((getmaxx()/2\*0)+100),(50)\*i+10,record);

delay(100);

}

}while(!kbhit());

getch();

cleardevice();

break;

}

}while(op!=3);

closegraph( );

return( 0 );

}

**FUNCION JUEGO**

Esta función bien podría ser la principal del funcionamiento medular del programa ya que es aquí donde se centra el funcionamiento más complejo que logra tener el programa, aquí es donde se hace el dibujado del escenario hacienda un llamado a la función que se encarga de esa tarea. Aquí es donde inicia el uso fuerte de los apuntadores de la lista MALLA que servirá como base para la manipulación del todo el juego.

int juego(tpnodo \*malla,int x,int y, JUGADOR \*jugador,ENEMIGOS \*enemigo,imagenes pics,int \*InoPag)

{

int salida=0;

int bala;

int cont\_en=0;

int con\_herido=0;

int cont\_enems\_archivo=0;

int cont\_herido\_archivo=0;

int tecla;

clock\_t inicio,iniDis;

clock\_t fin;

tpnodo aux=\*malla;

tpnodo aux\_ant=\*malla;

ENEMIGOS \*Aux\_En=enemigo;//AUXILIAR ENEMIGOS

jugador->nod\_malla=\*malla;//SE APUNTA EL JUGADOR A LA MALLA

jugador->nod\_malla->jugador=jugador;

//ini\_escenario(\*malla,pics,3,\*enemigo,&cont\_enems\_archivo,&cont\_herido\_archivo);//SE DIBUJA EL ESCENARIO.

ini\_escenario(\*malla,pics,jugador->nivel,\*enemigo,&cont\_enems\_archivo,&cont\_herido\_archivo);//SE DIBUJA EL ESCENARIO.

printf("contador enemigos= %d\n",cont\_enems\_archivo);

printf("contador heridos= %d\n",cont\_herido\_archivo);

getch();

dibujaicono(pics.maton,jugador->nod\_malla->cor\_x,jugador->nod\_malla->cor\_y,1,0);

setvisualpage(\*InoPag);

inicio=clock();//inicializacion de las varibles temporizadoras

iniDis=clock();

do//CICLO PRINCIPAL

{

fflush(stdin);

if(kbhit())

{

tecla=getch();

if(tecla==0)

{

tecla=getch();

}

switch(tecla)

{

case 72://ARRIBA

//checa\_y\_mueve(Ju,&aux,&aux\_ant,aux->arr,2,Es,list,&ban\_sal,&tecla);

movimientos\_jugador(jugador,&aux,&aux\_ant,aux->arr,1,&tecla,pics,malla);

break;

case 80: //ABAJO

movimientos\_jugador(jugador,&aux,&aux\_ant,aux->aba,2,&tecla,pics,malla);

break;

case 75://IZQUIERDA

movimientos\_jugador(jugador,&aux,&aux\_ant,aux->ant,3,&tecla,pics,malla);

break;

case 77://DERECHA

movimientos\_jugador(jugador,&aux,&aux\_ant,aux->sig,4,&tecla,pics,malla);

break;

case 32://DISPARO JUGADOR

if(jugador->direccion==4 && aux->sig!=NULL && aux->sig->info!='m' )//DERECHA

{

bala=crea\_bala(jugador->balas,aux->sig,6);

if(bala)

{

dibujaicono(pics.bala\_der,jugador->balas->final->nodo\_malla->cor\_x,jugador->balas->final->nodo\_malla->cor\_y,1,0);//PENDIENTE

}

}

if(jugador->direccion==3 && aux->ant!=NULL && aux->ant->info!='m' )//IZQUIERDA

{

bala=crea\_bala(jugador->balas,aux->ant,7);

if(bala)

{

dibujaicono(pics.bala\_iz,jugador->balas->final->nodo\_malla->cor\_x,jugador->balas->final->nodo\_malla->cor\_y,1,0);//PENDIENTE

}

}

if(jugador->direccion==1 && aux->arr!=NULL && aux->arr->info!='m')//ARRIBA

{

bala=crea\_bala(jugador->balas,aux->arr,8);

if(bala)

{

dibujaicono(pics.bala\_aba,jugador->balas->final->nodo\_malla->cor\_x,jugador->balas->final->nodo\_malla->cor\_y,1,0);//PENDIENTE

}

}

if(jugador->direccion==2 && aux->aba!=NULL && aux->aba->info!='m')//ABAJO

{

bala=crea\_bala(jugador->balas,aux->aba,9);

if(bala)

{

dibujaicono(pics.bala\_arr,jugador->balas->final->nodo\_malla->cor\_x,jugador->balas->final->nodo\_malla->cor\_y,1,0);//PENDIENTE

}

}

break;

printf("contador de muertes=%d\n",cont\_en);

}

info\_jug(\*jugador);

}

fin=clock();

if(Aux\_En->E->inicio != NULL)

{

if(movimientos\_enemigo\_fase1(&inicio,fin,Aux\_En))

{

//PENDIENTE FUNCION MUERE

muere(&aux,&aux\_ant,jugador,malla,&tecla,pics);

}

}

movimiento\_disparo\_fase1(&iniDis,fin,jugador,\*enemigo,pics,&cont\_en);

if(cont\_en==cont\_enems\_archivo && cont\_herido\_archivo==jugador->muertos)

{

salida=1;

}

delay(12);

}while(tecla!= 27 && salida==0);

libera\_lista\_malosos(enemigo->E);

libera\_mem(malla);

if(jugador->vida>0 && salida==1)

{

jugador->nivel++;

return 1;

}

return 0;

}

**FUNCION MOVIMIENTOS JUGADOR**

Esta función es la encargada del movimiento del jugador aquí, se hace un intercambio de ligas entre los apuntadores, todo en base del apuntador a la Malla. Dentro de esta función se hacen las distintas validaciones de los diversos movimientos que tiene el jugador.

void movimientos\_jugador(JUGADOR \*jug,tpnodo \*aux,tpnodo \*ant,tpnodo sig,int dir,int \*bandera\_salida,imagenes pic,tpnodo \*cabecera)

{

//if(sig!=NULL && sig->obstaculo!=1 && sig->enemigo==NULL && sig->puerta==NULL)

if(sig!=NULL && sig->info != 'm' && sig->enemigo==NULL)//añadir Condicion de Salida

{

\*ant=\*aux;

(\*aux)->jugador=NULL;

\*aux=sig;

(\*aux)->jugador=jug;

jug->nod\_malla=\*aux;

jug->direccion=dir;

dibuja\_jugador(dir,pic,\*ant,jug->nod\_malla);

if((\*aux)->info =='j')

{

jug->muertos+=1;

printf("MUERTOS=%d",jug->muertos);

(\*aux)->info='0' ;

}

switch(dir)

{

case 1: //ARRIBA

if(sig !=NULL && sig->arr !=NULL && sig->arr->enemigo !=NULL)

muere(aux,ant,jug,cabecera,bandera\_salida,pic);

//void muere(tpnodo \*aux,tpnodo \*ant, JUGADOR \*jugador, tpnodo \*cabecera, int \*tecla,imagenes pic)

break;

case 2://ABAJO

if(sig !=NULL && sig->aba !=NULL && sig->aba->enemigo !=NULL)

muere(aux,ant,jug,cabecera,bandera\_salida,pic);

break;

case 3://IZQUIERDA

if(sig !=NULL && sig->ant !=NULL && sig->ant->enemigo !=NULL)

muere(aux,ant,jug,cabecera,bandera\_salida,pic);

break;

case 4://DERECHA

if(sig !=NULL && sig->sig !=NULL && sig->sig->enemigo !=NULL)

muere(aux,ant,jug,cabecera,bandera\_salida,pic);

break;

}

}

}

**FUNCION PINTA JUGADOR**

Esta función pinta o muestra en pantalla al jugador en la siguiente posición del movimiento por obvias razones se le pasan como parámetro el apuntador apuntando al siguiente nodo de la malla que contiene las coordenadas actuales donde se tendrá que dibujar el jugador, un apuntador apuntando al nodo anterior de la malla para hacer el borrado de la imagen anterior del jugador, la variable que contiene ya en memoria las diversas imágenes usadas en el juego y un parámetro entero para indicar en que orientación se debe redibujar el jugador.

void dibuja\_jugador(int op,imagenes im,tpnodo borra,tpnodo juga)

{

switch(op)

{

case 1://ARRIBA

dibujaicono(im.maton\_up,juga->cor\_x,juga->cor\_y,1,0);

setfillstyle(1,BLACK);

bar(borra->cor\_x,borra->cor\_y,borra->cor\_x+TAM,borra->cor\_y+TAM);

floodfill(borra->cor\_x+2,borra->cor\_y+2,BLACK);

break;

case 2://ABAJO

dibujaicono(im.maton\_aba,juga->cor\_x,juga->cor\_y,1,0);

setfillstyle(1,BLACK);

bar(borra->cor\_x,borra->cor\_y,borra->cor\_x+TAM,borra->cor\_y+TAM);

floodfill(borra->cor\_x+2,borra->cor\_y+2,BLACK);

break;

case 3://IZQUIERDA

dibujaicono(im.maton\_iz,juga->cor\_x,juga->cor\_y,1,0);

setfillstyle(1,BLACK);

bar(borra->cor\_x,borra->cor\_y,borra->cor\_x+TAM,borra->cor\_y+TAM);

floodfill(borra->cor\_x+2,borra->cor\_y+2,BLACK);

break;

case 4://DERECHA

dibujaicono(im.maton,juga->cor\_x,juga->cor\_y,1,0);

setfillstyle(1,BLACK);

bar(borra->cor\_x,borra->cor\_y,borra->cor\_x+TAM,borra->cor\_y+TAM);

floodfill(borra->cor\_x+2,borra->cor\_y+2,BLACK);

break;

}

}

**FUNCION CREA BALA**

Esta función lo que hace es crear un bala en memoria e introducirla a una lista de balas. Básicamente es una inserción al final de una lista doble.

int crea\_bala(MUNICION parke,tpnodo nodo,int dir)

{

int res;

BALA nuevo;

res=crea\_nodo\_BALA(&nuevo,nodo->cor\_x,nodo->cor\_y);

if(res)

{

if(parke->inicio==NULL)

{

parke->inicio=nuevo;

parke->final=nuevo;

parke->inicio->nodo\_malla=nodo;

parke->inicio->direccion=dir;

}

else

{

parke->final->sig=nuevo;

nuevo->ant=parke->final;

parke->final=nuevo;

parke->final->nodo\_malla=nodo;

parke->final->direccion=dir;

}

}

return(res);

}

**Funcion que crea un nodo de la bala .**

Esta función lo que hace es crear un nodo de la bala del jugador.

int crea\_nodo\_BALA(BALA \*nodo,int x,int y)

{

int band=0;

\*nodo=(BALA)malloc(sizeof(struct regbala));

if(\*nodo)

{

(\*nodo)->x=x;

(\*nodo)->y=y;

(\*nodo)->ant=NULL;

(\*nodo)->sig=NULL;

(\*nodo)->direccion=6;

(\*nodo)->nodo\_malla=(tpnodo)malloc(sizeof(struct nodo));

if((\*nodo)->nodo\_malla)

band=1;

}

return(band);

}

**FUNCION MOVIMIENTO BALA FASE UNO.**

Esta función lo que hace básicamente es recorrer la lista doble de balas al recorrer la lista esta permite que al llamar una función dentro del while que recorre la lista doble de balas que es la que le da el movimiento real a las balas en esta función lo que hace es pasar como parámetro la liga siguiente de la bala actual que se está en la lista doble de balas.

void movimiento\_disparo\_fase1(clock\_t \*ini,clock\_t fin,JUGADOR \*jugador,ENEMIGOS E,imagenes pics,int \*cont\_muertes)

{

int seg;

seg=(fin-(\*ini))%CLK\_TCK;

BALA auxiliar\_balas=jugador->balas->inicio;

tpnodo ant;

if(seg>100)

{

while(auxiliar\_balas !=NULL)

{

if(auxiliar\_balas->direccion ==6)//DERECHA

{

movimiento\_disparo\_fase2(auxiliar\_balas->nodo\_malla->sig,&ant,&auxiliar\_balas,jugador,E,6,pics,cont\_muertes);

}

if(auxiliar\_balas->direccion ==7)//IZQUIERDA

{

movimiento\_disparo\_fase2(auxiliar\_balas->nodo\_malla>ant,&ant,&auxiliar\_balas,jugador,E,7,pics,cont\_muertes);

}

if(auxiliar\_balas->direccion ==8)//ARRIBA

{

movimiento\_disparo\_fase2(auxiliar\_balas->nodo\_malla->arr,&ant,&auxiliar\_balas,jugador,E,8,pics,cont\_muertes);

}

if(auxiliar\_balas->direccion ==9)//ABAJO

{

movimiento\_disparo\_fase2(auxiliar\_balas->nodo\_malla->aba,&ant,&auxiliar\_balas,jugador,E,9,pics,cont\_muertes);

}

auxiliar\_balas=auxiliar\_balas->sig;//Siguiente bala

}

\*ini=clock();

}

}

**FUNCION MOVIMIENTO FASE2**

Esta función hace el movimiento del disparo aquí se hacen las validaciones prudentes que necesita el disparo para darle movimiento y también elimina un enemigo si es que este lo toca la bala.

void movimiento\_disparo\_fase2(tpnodo sig,tpnodo \*ant\_malla,BALA \*auxiliar\_disparo,JUGADOR \*jugador,ENEMIGOS En,int dir,imagenes pic,int \*cont\_muertes)

{

if(sig!=NULL && sig->info != 'm' &&sig->info !='j' && (\*auxiliar\_disparo)->nodo\_malla->enemigo ==NULL)

{

\*ant\_malla=(\*auxiliar\_disparo)->nodo\_malla;

(\*auxiliar\_disparo)->nodo\_malla=sig;//apunta la bala al siguiente nodo de la malla

dibuja\_disparo(dir,pic,\*ant\_malla,sig->cor\_x,sig->cor\_y);

}

else

{

if((\*auxiliar\_disparo)->nodo\_malla->enemigo!=NULL)//SI ES ENEMIGO

{

setfillstyle(1,BLACK);

bar((\*auxiliar\_disparo)->nodo\_malla->cor\_x,(\*auxiliar\_disparo)->nodo\_malla->cor\_y,(\*auxiliar\_disparo)->nodo\_malla->cor\_x+TAM,(\*auxiliar\_disparo)->nodo\_malla->cor\_y+TAM);

floodfill((\*auxiliar\_disparo)->nodo\_malla->cor\_x+2,(\*auxiliar\_disparo)->nodo\_malla->cor\_y+2,BLACK);

dibujaicono(pic.explosion,(\*auxiliar\_disparo)->nodo\_malla->cor\_x,(\*auxiliar\_disparo)->nodo\_malla->cor\_y,1,0);

delay(300);

elimina\_maloso(En.E,(\*auxiliar\_disparo)->nodo\_malla->enemigo);

(\*auxiliar\_disparo)->nodo\_malla->enemigo=NULL;

jugador->puntos+=5;

\*cont\_muertes=\*cont\_muertes+1;

printf("CONTADOR MUERTES %d\n",\*cont\_muertes);

//LIBERA ENEMIGO

}

setfillstyle(1,BLACK);

bar((\*auxiliar\_disparo)->nodo\_malla->cor\_x,(\*auxiliar\_disparo)->nodo\_malla->cor\_y,(\*auxiliar\_disparo)

>nodo\_malla->cor\_x+TAM,(\*auxiliar\_disparo)->nodo\_malla->cor\_y+TAM);

floodfill((\*auxiliar\_disparo)->nodo\_malla->cor\_x+2,(\*auxiliar\_disparo)->nodo\_malla->cor\_y+2,BLACK);

elimina\_bala(jugador->balas,\*auxiliar\_disparo);//LIBERAR MEMORIA DISPARO

}

}

Esta Funcion lo que hace Basicamente es liberar el nodo de la bala o eliminarlo de la lista doble de balas.

int elimina\_bala(MUNICION parke,BALA nod)

{

BALA aux=parke->inicio;

BALA aux\_ant=parke->inicio;

if(aux !=NULL)

{

if(aux==nod)

{

if(aux->sig!=NULL) //SI ES EL PRIMERO

{

parke->inicio=aux->sig;

parke->inicio->ant=NULL;

free(aux);

}

else

{

parke->inicio=NULL;

parke->final-NULL;

free(aux);

}

return 1;

}

aux=parke->final;

if(aux==nod)//SI ES EL ULTIMO

{

if(aux->ant!=NULL)

{

parke->final=aux->ant;

parke->final->sig=NULL;

free(aux);

}

else

{

parke->inicio=NULL;

parke->final=NULL;

free(aux);

}

return 1;

}

else

{

aux=parke->inicio->sig;

aux\_ant=parke->inicio;

while(aux !=parke->final)

{

if(aux==nod)

{

aux\_ant->sig=aux->sig;

aux\_ant->sig->ant=aux\_ant;

free(aux);

return 1;

}

aux=aux->sig;

aux\_ant=aux\_ant->sig;

}

}

}

}

**FUNCION ELIMINA MALOSO**

Esta función básicamente hace una simple eliminación en la lista doble de enemigos que se tiene desde luego esta función solo se manda llamar cuando el enemigos es tocado por una bala.

int elimina\_maloso(LISTA\_ENEMIGOS enem,ENEMIGO nodo)

{

ENEMIGO aux=enem->inicio;

ENEMIGO aux\_ant=enem->inicio;

if(aux !=NULL)

{

if(aux==nodo)

{

if(aux->sig!=NULL)

{

enem->inicio=aux->sig;

enem->inicio->ant=NULL;

free(aux);

}

else

{

enem->inicio=NULL;

enem->final=NULL;

free(aux);

}

return 1;

}

aux=enem->final;

if(aux==nodo)

{

if(aux->ant !=NULL)

{

enem->final=aux->ant;

enem->final->sig=NULL;

free(aux);

}

else

{

enem->inicio=NULL;

enem->final=NULL;

free(aux);

}

return 1;

}

else

{

aux=enem->inicio->sig;

aux\_ant=enem->inicio;

while(aux !=enem->final)

{

if(aux==nodo)

{

aux\_ant->sig=aux->sig;

aux\_ant->sig->ant=aux\_ant;

free(aux);

return 1;

}

aux=aux->sig;

aux\_ant=aux\_ant->sig;

}

}

}

}

**Función Dibuja Disparo**

Esta función dibuja el disparo en una determinada orientación que está dada por el parámetro de direccionamiento que se lo da el jugador.

void dibuja\_disparo(int direccion,imagenes im,tpnodo borra,int x,int y)

{

switch(direccion)

{

case 6://DERECHA

//printf("ENTRO FUNCION DIBUJA DISPARO con un%d \n\n",6);

dibujaicono(im.bala\_der,x,y,1,0);

setfillstyle(1,BLACK);

bar(borra->cor\_x,borra->cor\_y,borra->cor\_x+TAM,borra->cor\_y+TAM);

floodfill(borra->cor\_x+2,borra->cor\_y+2,BLACK);

//dibujaicono(im.maton\_up,juga->cor\_x,juga->cor\_y,1,0);

break;

case 7://IZQUIERDA

dibujaicono(im.bala\_iz,x,y,1,0);

setfillstyle(1,BLACK);

bar(borra->cor\_x,borra->cor\_y,borra->cor\_x+TAM,borra->cor\_y+TAM);

floodfill(borra->cor\_x+2,borra->cor\_y+2,BLACK);

break;

case 8://ARRIBA

dibujaicono(im.bala\_aba,x,y,1,0);

setfillstyle(1,BLACK);

bar(borra->cor\_x,borra->cor\_y,borra->cor\_x+TAM,borra->cor\_y+TAM);

floodfill(borra->cor\_x+2,borra->cor\_y+2,BLACK);

break;

case 9://ABAJO

dibujaicono(im.bala\_arr,x,y,1,0);setfillstyle(1,BLACK);

bar(borra->cor\_x,borra->cor\_y,borra->cor\_x+TAM,borra->cor\_y+TAM);

floodfill(borra->cor\_x+2,borra->cor\_y+2,BLACK);

break;

}

}

**FUNCION MOVIMIENTO ENEMIGO FASE 1**

Esta función funciona exactamente igual que la función mueve disparo fase 1, la única diferencia es el parámetro enemigos que contiene la un campo que alberga las cabeceras de la lista doble de enemigos.

int movimientos\_enemigo\_fase1(clock\_t \*inicio,clock\_t final,ENEMIGOS \*maloso)

{

int seg;

ENEMIGO auxiliar\_enemigos;

tpnodo anterior;

auxiliar\_enemigos=maloso->E->inicio;

seg=(final-(\*inicio))%CLK\_TCK;

if(seg > 500)

{

//INICIA RECORRIDO DE LA LISTA ENEMIGOS

while(auxiliar\_enemigos)

{

switch(auxiliar\_enemigos->band\_dir)

{

case 0:

//void movimientos\_enemigos\_fase2(ENEMIGO \*aux\_maloso, tpnodo \*ant\_enemigo,tpnodo sig,int dire,ENMIGOS \*maloson)

if(movimientos\_enemigos\_fase2(&auxiliar\_enemigos,&anterior,auxiliar\_enemigos->nodo\_malla->arr,maloso))

return 1;

break;

case 1:

if(movimientos\_enemigos\_fase2(&auxiliar\_enemigos,&anterior,auxiliar\_enemigos->nodo\_malla->aba,maloso))

return 1;

break;

case 2:

if(movimientos\_enemigos\_fase2(&auxiliar\_enemigos,&anterior,auxiliar\_enemigos->nodo\_malla->sig,maloso))

return 1;

break;

case 3:

if(movimientos\_enemigos\_fase2(&auxiliar\_enemigos,&anterior,auxiliar\_enemigos->nodo\_malla->ant,maloso))

return 1;

break;

}//Llave switch

//AVANZE EN LA LISTA DE ENEMIGOS AL SIGUIENTE ENEMIGO

auxiliar\_enemigos=auxiliar\_enemigos->sig;

}//Llave While

\*inicio=clock();

}

return 0;

}//Llave Funcion

**FUNCION MOVIMIENTOS ENEMIGOS FASE 2**

Esta función tiene el mismo funcionamiento que la función mueve disparo fase 2 aquí es donde se hacen las validaciones prudentes para el movimiento de los enemigos y además se hace el dibujado del enemigo en la siguiente posición, además aquí se toma en cuenta también la condición del jugador ya que si este se encuentra en un nodo adyacente donde se situara el enemigo en su próximo movimiento el jugador se elimina.

int movimientos\_enemigos\_fase2(ENEMIGO \*aux\_maloso, tpnodo \*ant\_enemigo,tpnodo sig,ENEMIGOS \*maloson)

{

//CONDICIONES PARA QUE SE HAGA EL AVANZE DEL ENEMIGO

//if(sig!=NULL && sig->info != 'm' && sig->info!='j' && sig->enemigo==NULL)//SUPER IFCondicion para que muera solo si toca al jugador

if(sig!=NULL && sig->info != 'm' && sig->info!='j' && sig->jugador==NULL && sig->enemigo==NULL)//SUPER IF

{

\*ant\_enemigo=(\*aux\_maloso)->nodo\_malla;//se guarda la posicion anterior del nodo enemigo que apunta a la malla;

(\*aux\_maloso)->nodo\_malla->enemigo=NULL;//SE PONE EN ANTERIOR EL ENEMIGO ANTERIOR

(\*aux\_maloso)->nodo\_malla=sig;

(\*aux\_maloso)->nodo\_malla->enemigo=\*aux\_maloso;

(\*aux\_maloso)->x=(\*aux\_maloso)->nodo\_malla->cor\_x;

(\*aux\_maloso)->y=(\*aux\_maloso)->nodo\_malla->cor\_y;

dibujaicono(maloson->enem.maloso,(\*aux\_maloso)->x,(\*aux\_maloso)->y,1,0);

setfillstyle(1,BLACK);

bar((\*ant\_enemigo)->cor\_x,(\*ant\_enemigo)->cor\_y,(\*ant\_enemigo)->cor\_x+TAM,(\*ant\_enemigo)->cor\_y+TAM);

floodfill((\*ant\_enemigo)->cor\_x+2,(\*ant\_enemigo)->cor\_y+2,BLACK);

//VALIDAR SI EL JUGADOR ESTA AL LADO

if(sig->sig!=NULL && sig->sig->jugador!=NULL)

return 1;

if(sig->ant!=NULL && sig->ant->jugador!=NULL)

return 1;

if(sig->arr!=NULL && sig->arr->jugador!=NULL)

return 1;

if(sig->aba!=NULL && sig->aba->jugador!=NULL)

return 1;

}

else

{

(\*aux\_maloso)->band\_dir=rand()%4;

}

return 0;

}

**FUNCION INICIALIZAR ENEMIGO**

Esta función inicializa los campos necesarios para el manejo de la lista doble de enemigos.

void inicializar\_enemigo(ENEMIGOS \*Ens)

{

Ens->E=(LISTA\_ENEMIGOS)malloc(sizeof(struct malos));

Ens->E->inicio=NULL;

Ens->E->final=NULL;

}

**FUNCION CREA ENEMIGO**

Esta función básicamente es una inserción a una lista doble de enemigos

void crea\_enemigo(ENEMIGOS \*Ens, tpnodo cab)

{

int res;

ENEMIGO nuevo;

res=crea\_nodo\_ENEMIGO(&nuevo,cab->cor\_x,cab->cor\_y);

if(res)

{

if(!Ens->E->inicio)

{

Ens->E->inicio=Ens->E->final=nuevo;

Ens->E->inicio->nodo\_malla=cab;

cab->enemigo=Ens->E->inicio;

}

else

{

Ens->E->final->sig=nuevo;

nuevo->ant=Ens->E->final;

Ens->E->final=nuevo;

Ens->E->final->nodo\_malla=cab;

cab->enemigo=Ens->E->final;

}

}

}

**FUNCION CREA NODO ENEMIGO**

Esta función crea un nodo tipo enemigo (para el manejo de los enemigos en la malla) para su posterior inserción en la lista doble de enemigos.

int crea\_nodo\_ENEMIGO(ENEMIGO \*nodo,int x, int y)

{

int res=0;

\*nodo=(ENEMIGO)malloc(sizeof(struct enemi));

if(\*nodo)

{

(\*nodo)->x=x;

(\*nodo)->y=y;

(\*nodo)->ant=NULL;

(\*nodo)->sig=NULL;

(\*nodo)->band\_dir=rand()%4;

(\*nodo)->nodo\_malla=(tpnodo)malloc(sizeof(struct nodo));

res=1;

}

return(res);

}

**FUNCION INICIALIZAR JUGADOR**

Esta función lo que hace básicamente es inicializar los campos de una variable tipo jugador para su posterior uso en el programa. Como se puede apreciar las distintas ligas que se usaran se inicializan así como también se hace su reservación de memoria para un posterior uso de estas.

void inicializar\_jugador(JUGADOR \*jugador)

{

jugador->nod\_malla=NULL;

jugador->nivel=1;

jugador->direccion=4;

strcpy(jugador->nombre,"NOIVAZMAR");

jugador->muertos=0;

jugador->vida=2;

jugador->puntos=0;

jugador->balas=(MUNICION)malloc(sizeof(struct regbala));

jugador->balas->inicio=NULL;

jugador->balas->final=NULL;

}

**FUNCION CREA MALLA**

Esta función hace un armado de locaciones de memoria enlazándolas una a una en forma de malla, para que el uso de esta haga más fácil, más cómodo y más dinámico la implementación de la memoria en el programa.

void crea\_malla(tpnodo \*cab,int x,int y)

{

int i,j,reinicio\_x=x;

tpnodo nuevo=NULL;

tpnodo auxren=NULL,auxcol=NULL;

//AUXCOL auxiliar usado para situarse en las columnas de la lista

//AUXREN auxliar usado para situarse en los renglones de la lista

for(i=0;i<NREN;i++)//FOR QUE CREA LAS FILAS DE LA MALLA(RENGLONES)

{

crea\_nodo\_malla(&nuevo,x,y);

if(!\*cab)//CHECA SI LA LISTA ESTA VACIA

\*cab=auxren=auxcol=nuevo;//SI LA LISTA ESTA VACIA TODO APUNTA AL PRIMER NODO ES DECIR EL NODO RECIEN CREADO

else

{

auxcol=auxren;//VUELVE A SITUAR A AUXCOL AL INICIO DEL RENGLON LO REGRESA DEL ULTIMO NODO AL PRIMERO

auxcol->aba=nuevo;

auxren=auxren->aba;//\*\*

auxren->arr=auxcol;//INDICA QUE YA HAY UNA FILA

auxcol=auxren;//\*\*\*ESTA INSTRUCCION DA INICIO A LA FILA SIGUIENTE DE LA MALLA(<<SIGUIENTE RENGLON>>)

}

for(j=0;j<NCOL;j++)//FOR QUE CREA LAS COLUMNAS DE LA MALLA

{

x+=TAM;

crea\_nodo\_malla(&nuevo,x,y);

auxcol->sig=nuevo;//

nuevo->ant=auxcol;//NUEVO APUNTA->ANTERIOR APUNTA A LO QUE AUXCOL ESTA APUNTADO

if(auxren->arr)

{

auxcol->sig->arr=auxcol->arr->sig;

auxcol->arr->sig->aba=auxcol->sig;

}

auxcol=auxcol->sig;

}

y+=TAM;//AUMENTA COORDENADAS EN Y para generar nuevas coordenadas

x=reinicio\_x;//reinicia las coordenadas en X para iniciar con la nueva FILA

}

}

**FUNCION CREA NODO MALLA**

Función que crea un nodo de la malla.

void crea\_nodo\_malla(tpnodo \*nodo,int x, int y)

{

\*nodo=(tpnodo)malloc(sizeof(struct nodo));

if(\*nodo)

{

(\*nodo)->info='0';//Se inicializa los valores para la posterior utilizacion del Nodo

(\*nodo)->existe=0;//Se inicializa los valores para la posterior utilizacion del Nodo

(\*nodo)->cor\_x=x;

(\*nodo)->cor\_y=y;

(\*nodo)->arr=NULL;

(\*nodo)->aba=NULL;

(\*nodo)->ant=NULL;

(\*nodo)->sig=NULL;

(\*nodo)->jugador=NULL;

(\*nodo)->enemigo=NULL;

}

}

**FUNCION LIBERA MEM**

Esta función lo que hace es liberar la memoria usada en la implementación de la malla libera uno a uno cada nodo de la malla.

void libera\_mem(tpnodo \*cab)

{

tpnodo auxren,auxcol;

auxren=auxcol=\*cab;

while(auxren)

{

auxren=auxren->aba;

while(auxcol)

{

free(auxcol);

auxcol=auxcol->sig;

}

auxcol=auxren;

}

\*cab=NULL;

}

**FUNCION TECLA**

Esta función lo que hace es descartar si se ha presionado una tecla con anterioridad

char tecla()

{

char c;

c=getch();

if(c==0)

{

c=getch();

}

return(c);

}

**FUNCION CAARGA IMAGEN ENEMIGO**

Esta función lo que hace es cargar en memoria la información de las imágenes para su posterior dibujado. Aquí se hace un llamado a la función que carga el archivo con la información del icono (Coordenadas, número de pixeles, color)

void cargaImagenes(imagenes \*img)//carga todas las imagenes a la estructura

{

creaicono("maton.edb",&img->maton);

creaicono("matonizquierda.edb",&img->maton\_iz);

creaicono("mariod.edb",&img->maton\_aba);

creaicono("matonup.edb",&img->maton\_up);

creaicono("pared2.edb",&img->muro2);

creaicono("maloso.edb",&img->maloso);

creaicono("bala2\_der.edb",&img->bala\_der);

creaicono("bala\_izquierda2.edb",&img->bala\_iz);

creaicono("bala\_aba.edb",&img->bala\_aba);

creaicono("bala\_arr.edb",&img->bala\_arr);

creaicono("explosion.edb",&img->explosion);

creaicono("muerto.edb",&img->muerto);

}

**FUNCION CREA ICONO**

Función que se encarga de leer el archivo con la información del icono y que a su vez lo carga en memoria (Arreglo Bidimensional) para su posterior dibujado.

void creaicono(char cad[],imagen \*imag)//funcion que crea iconos leyendolo de un archivo y guardandolos en una variable por referencia

{

int i,j,x;

FILE \*fp;

fp = fopen(cad,"rb");//se abre el archivo segun la cadena cad[]

fread(&imag->dimy,sizeof(int),1,fp);//se guardan las dimensiones de la imagen

fread(&imag->dimx,sizeof(int),1,fp);

imag->img = (int\*\*)malloc(sizeof(int\*)\*imag->dimy);//se reserva memoria para la imagen

for(i=0;i<imag->dimy;i++)//reserva memoria

\*((imag->img)+i) = (int\*)malloc(sizeof(int)\*imag->dimx);

for(i=0;i<imag->dimy;i++)//se guarda cada pixel en un arreglo

for(j=0;j<imag->dimx;j++)

{

fread(&x,sizeof(int),1,fp);

imag->img[i][j] = x;

}

fclose(fp);//se cierra el archivo

}

**FUNCION DIBUJA ICONO**

Esta función lo que hace es imprimir el icono lo hace con la ayuda de un arreglo bidimensional que contiene las coordenadas del pixel y el color del pixel.

int dibujaicono(imagen imag,int x,int y,int tam,int dir)//funcion que dibuja un icono en las coordenadas x y,tam: tamaño del icono, dir: rotacion del icono

{

int i,j;

float o;

o=0.5;

for(i=0;i<imag.dimy;i++)//ciclo para leer cada color de pixel del arreglo de la imagen

for(j=0;j<imag.dimx;j++)

{

if(imag.img[i][j] != 67043583)//si el color es diferente a ROSA se imprime en pantalla

{

setfillstyle(1,imag.img[i][j]);

switch(dir)

{

case 0://0 grados

bar(x+(j\*tam),y+(i\*tam),x+(j\*tam)+tam,y+(i\*tam)+tam);

break;

case 1://90 derecha

bar(x+imag.dimy-(i\*tam),y+(j\*tam),x+imag.dimy-((i\*tam)+tam),y+(j\*tam)+tam);

break;

case 2://180 derecha

bar(imag.dimx+imag.dimx/2+(x-(j\*tam)),imag.dimy+(y-(i\*tam)),imag.dimx+imag.dimx/2+(x-(j\*tam)+tam),imag.dimy+(y-(i\*tam)+tam));

break;

case 3://270 derecha

bar(x+(i\*tam),y+imag.dimx+imag.dimx/2-(j\*tam),x+((i\*tam)+tam),y+imag.dimx+imag.dimx/2-((j\*tam)+tam));

break;

}

}

}

}

**INICIA ESCENARIO**

Esta función lo que hace es inicializar el escenario leyendo el archivo que contiene los parámetros necesarios para dibujar en cada nodo de la malla la imagen deseada y para inicializar la información de los nodos que servirán como parámetros para el funcionamiento del juego

void ini\_escenario(tpnodo main\_dor,imagenes pic,int nivel,ENEMIGOS maloso,int \*cont\_enems,int \*cont\_heridos)

{

tpnodo aux;//\*\*\*\*\*AUXILIAR PARA EL RECORRIDO DE LA MALLA

aux=main\_dor;

char dat;

FILE \*f;

nivel==1 ? f=fopen("nivel1.txt","r") : nivel==2 ? f=fopen("nivel2.txt","r") : f=fopen("nivel3.txt","r");

//f=fopen("nivel1.txt","r");

while(!feof(f)) //feof(f)==NULL es igual a !feof(f) feof(f)!=NULL es igual a feof(f)

{

dat=fgetc(f);

if(dat=='1')

{

dibujaicono(pic.muro2,aux->cor\_x,aux->cor\_y,1,0);

aux->info='m';

}

if(dat=='4')

{

dibujaicono(pic.muro2,aux->cor\_x,aux->cor\_y,1,0);

aux->info='m';

}

if(dat=='2')

{

crea\_enemigo(&maloso,aux);

aux->enemigo=maloso.E->final;

dibujaicono(maloso.enem.maloso,aux->enemigo->x,aux->enemigo->y,1,0);

aux->info='E';

\*cont\_enems=\*cont\_enems+1;

printf("CONTADOR ENEMIGOS=%d\n",\*cont\_enems);

}

if(dat=='3')

{

dibujaicono(pic.muerto,aux->cor\_x,aux->cor\_y,1,0);

aux->info='j';

\*cont\_heridos=\*cont\_heridos+1;

printf("\nCONTADOR HERIDOS=%d\n",\*cont\_heridos);

}

if(!feof(f))

{

aux=aux->sig;

if(!aux)

{

main\_dor=main\_dor->aba;

aux=main\_dor;

}

}

}

fclose(f);

// \*cont\_enems=cont;

}

**FUNCION PORTADA**

Esta función imprime la portada inicial del programa.

void portada()

{

int display\_x,display\_y;

display\_x=getmaxx();

display\_y=getmaxy();

setcolor(4);

settextstyle(1,0,5);

outtextxy(display\_x\*0+100 ,display\_y\*0+25, "Universidad Autonoma de San Luis Potosi" );

setcolor(1);

outtextxy(display\_x\*0+100 ,display\_y\*0+75, "Facultad de Ingenieria" );

outtextxy(display\_x\*0+100 ,display\_y\*0+125, "Area de Computacion e Informatica" );

outtextxy(display\_x\*0+100 ,display\_y\*0+175, "Estrucuturas de Datos y Algoritmos B");

outtextxy(display\_x\*0+200 ,display\_y-100, "Elaborado por: Jesus Aurelio Hernandez Aguilar");

outtextxy(display\_x\*0+200 ,display\_y-200, "Profesor Miguel Angel de la Cruz Lopez");

settextstyle(0,0,8);

setcolor(4);

outtextxy(display\_x\*0+200 ,display\_y/2-50, "ZAMARRIPA RESCUE");

getch();

}

**FUNCION INFO JUG**

Esta función imprime durante el juego la información (Puntos y Vidas) del jugador durante el juego.

void info\_jug(JUGADOR jugador)

{

char vidas[2];

char puntos[5];

char heridos[2];

int x=50;

int y=10;

setcolor(LIGHTBLUE);

settextstyle(3,0,2);///2

outtextxy(x-45,y,"PUNTOS");

sprintf(puntos,"%d",jugador.puntos);

outtextxy(x-30,y+30,puntos);

settextstyle(3,0,2);///2

setcolor(LIGHTBLUE);

outtextxy(x-45,y+60,"HERIDOS");

sprintf(heridos,"%d",jugador.muertos);

outtextxy(x-30,y+90,heridos);

setcolor(LIGHTBLUE);

outtextxy(x-45,y+120,"VIDAS");

sprintf(vidas,"%d",jugador.vida);

setcolor(RED);

outtextxy(x-30,y+150,vidas);

/\*

outtextxy(getmaxx()-280+x,10+y,ju.nombre);

sprintf(puntos,"%d", ju.puntos);

outtextxy(getmaxx()-80+x,40+y,puntos);

outtextxy(getmaxx()-280+x,40+y,"Puntos:");

\*/

}

**FUNCION MUERE**

Función que se encarga de regresar al jugador al inicio del laberinto y de restarle vidas al jugador, además de dibujar una icono que simula la explosión del jugador

void muere(tpnodo \*aux,tpnodo \*ant, JUGADOR \*jugador, tpnodo \*cabecera, int \*tecla,imagenes pic)

{

\*ant=\*aux;

(\*aux)->jugador=NULL;

jugador->nod\_malla=\*aux;

delay(500);

jugador->vida--;

setfillstyle(1,BLACK);

bar(jugador->nod\_malla->cor\_x,jugador->nod\_malla->cor\_y,jugador->nod\_malla->cor\_x+TAM,jugador->nod\_malla->cor\_y+TAM);

floodfill(jugador->nod\_malla->cor\_x+2,jugador->nod\_malla->cor\_y+2,BLACK);

dibujaicono(pic.explosion,jugador->nod\_malla->cor\_x,jugador->nod\_malla->cor\_y,1,0);

delay(450);

setfillstyle(1,BLACK);

bar(jugador->nod\_malla->cor\_x,jugador->nod\_malla->cor\_y,jugador->nod\_malla->cor\_x+TAM,jugador->nod\_malla->cor\_y+TAM);

floodfill(jugador->nod\_malla->cor\_x+2,jugador->nod\_malla->cor\_y+2,BLACK);

if(jugador->vida==0)

{

\*tecla=27;

}

else

{

delay(1000);

jugador->nod\_malla=\*cabecera;

dibujaicono(pic.maton,jugador->nod\_malla->cor\_x,jugador->nod\_malla->cor\_y,1,0);

\*aux=\*cabecera;

(\*aux)->jugador=jugador;

\*ant=\*cabecera;

jugador->direccion=4;

}

}

**FUNCION LIBERA LISTA MALOSOS**

Función que se encarga de liberar la memoria usada en la implementación de la lista doble de enemigos.

void libera\_lista\_malosos(LISTA\_ENEMIGOS list)

{

ENEMIGO aux\_elimina;

ENEMIGO aux;

if(list !=NULL)

{

if(list->final==list->inicio)

{

free(list);

}

else

{

while(list->final==list->inicio)

{

aux=list->final->ant;

aux\_elimina=list->final;

list->final=aux;

free(aux\_elimina);

}

free(list);

}

}

}

**FUNCION MENU2**

Esta función dibuja el menú implementado mediante Mouse. Regresa como parámetro el número de botón al que se le ha dado clic.

int menu2()

{

TBoton botones[NUMBOTONES];

int x,y,bot;

portada1();

pintamenu(botones);

clearmouseclick(WM\_LBUTTONDOWN);

while (!ismouseclick(WM\_LBUTTONDOWN))

delay(50);

getmouseclick(WM\_LBUTTONDOWN, x, y);

bot=checaboton(botones,x,y);

botones[bot].tipo=PRE;

dibujaboton(botones[bot]);

delay(100);

botones[bot].tipo=NORMAL;

dibujaboton(botones[bot]);

return(bot);

}

**FUNCION PORTADA1**

Esta función imprime una segunda portada al momento de aparecer el menú.

void portada1()

{

int display\_x,display\_y;

display\_x=getmaxx();

display\_y=getmaxy();

setcolor(LIGHTBLUE);

line(display\_x\*0+5,display\_y\*0+5,display\_x-5,display\_y\*0+5);

line(display\_x\*0+5,display\_y\*0+5,display\_x\*0+5,display\_y-5);

line(display\_x\*0+5,display\_y-5,display\_x-5,display\_y-5);

line(display\_x-5,display\_y-5,display\_x-5,display\_y\*0+5);

line(display\_x\*0+10,display\_y\*0+10,display\_x-10,display\_y\*0+10);

line(display\_x\*0+10,display\_y\*0+10,display\_x\*0+10,display\_y-10);

line(display\_x\*0+10,display\_y-10,display\_x-10,display\_y-10);

line(display\_x-10,display\_y-10,display\_x-10,display\_y\*0+10);

line(display\_x\*0+15,display\_y\*0+15,display\_x-15,display\_y\*0+15);

line(display\_x\*0+15,display\_y\*0+15,display\_x\*0+15,display\_y-15);

line(display\_x\*0+15,display\_y-15,display\_x-15,display\_y-15);

line(display\_x-15,display\_y-15,display\_x-15,display\_y\*0+15);

setcolor(2);

settextstyle(1,0,5);

outtextxy(display\_x\*0+200 ,display\_y-100, "infocomp");

outtextxy(display\_x\*0+200 ,display\_y-200, "Facultad de Ingeniera");

outtextxy(display\_x\*0+200 ,display\_y-300, "UASLP");

settextstyle(0,0,8);

setcolor(2);

outtextxy(display\_x\*0+20 ,display\_y/2-50, "ZAMARRIPA AL RESCATE");

settextstyle(0,0,4);

outtextxy(display\_x-400 ,display\_y-50, "E.D.A 'B'");

}

**FUNCION PINTA MENU**

Esta función lo que hace es mandar llamar a las funciones que se encargaran de armar el menú .

int pintamenu(TBoton Bot[])

{

settextstyle(TRIPLEX\_FONT, HORIZ\_DIR, 2);

llenabotones(Bot);

dibujabotones(Bot,4);

}

**FUNCION CHECA BOTON**

Función que verifica que botón del menú se ha presionado.

int checaboton(TBoton z[], int xm, int ym)

{

int i;

int reg=-1,x1,y1,x2,y2;

for(i=0;i<NUMBOTONES && reg==-1;i++)

{

x1=z[i].x;

x2=z[i].x+z[i].tamx;

y1=z[i].y;

y2=z[i].y+z[i].tamy;

if(xm>=x1&& xm<=x2&&ym>y1&&ym<=y2)

reg=i;

}

return(reg);

}

**FUNCION DIBUJA BOTON**

Función que se encarga del dibujado de cada botón del menú.

void dibujaboton(TBoton z)

{

int color;

color=z.tipo==NORMAL?LIGHTGRAY:DARKGRAY;

setfillstyle(SOLID\_FILL, 2);

bar(z.x,z.y,z.x+z.tamx,z.y+z.tamy);

setcolor(color);

line(z.x,z.y,z.x+z.tamx,z.y);

line(z.x,z.y,z.x,z.y+z.tamy);

color=z.tipo==NORMAL?DARKGRAY:LIGHTGRAY;

setcolor(color);

line(z.x,z.y+z.tamy,z.x+z.tamx,z.y+z.tamy);

line(z.x+z.tamx,z.y,z.x+z.tamx,z.y+z.tamy);

setcolor(15);

//settextjustify(1,1);

outtextxy(z.x+z.tamx/2,z.y+z.tamy/2,z.texto);

}

**FUNCION DIBUJA BOTONES**

Función que se encarga de llamar el número de botones que se requiere dibujar para armar el menú.

void dibujabotones(TBoton z[], int w)

{

int i;

for(i=0;i<w;i++)

dibujaboton(z[i]);

}

**FUNCION LLENA BOTON**

Esta función se encarga de imprimir la información que tendrá el botón del menú.

void llenaboton(int num, int x, int y, char texto[], TBoton \*z)

{

z->num=num;

z->x=x;

z->y=y;

strcpy(z->texto, texto);

z->tipo=NORMAL;

z->tamx=textwidth(texto)\*2;

z->tamy=textheight(texto)\*2;

}

**FUNCION LLENA BOTONES**

Función que se encarga de cargar la información (texto) de cada Botón.

void llenabotones(TBoton z[])

{

llenaboton(1,190,80,"JUEGO", &z[0]);

llenaboton(2,190,120,"AYUDAS", &z[1]);

llenaboton(3,190,160,"RECORDS", &z[2]);

llenaboton(4,190,200,"SALIR", &z[3]);

}

void pedir\_nombre(char nombre[])

{

//descartar\_input();

cleardevice();

int x = getmaxx()/2;

int y = getmaxy()/2;

settextjustify(LEFT\_TEXT, TOP\_TEXT);

//settextstyle(TRIPLEX\_FONT, HORIZ\_DIR, 2);

settextstyle(0, HORIZ\_DIR, 4);

setcolor(WHITE);

outtextxy(x-textwidth("nombre:"), y, "nombre:");

intextxy(x, y, nombre, BLACK, WHITE);

}

**FUNCION LEE AYUDA**

Esta función lo que hace es leer un archivo

void ayuda(char nom\_arch[])

{

FILE \*f;

int x=0,y=0;

char texto[100];

f=fopen(nom\_arch,"r");

if(f==NULL)

{

outtextxy(400,600,"Archivo Inexistente");

getch();

}

else

{

cleardevice();

setcolor(15);

settextstyle(0,0,3);

while(!feof(f))

{

fgets(texto,100,f);

outtextxy(x,y,texto);

y+=textheight(texto);

//if(y>=450)

//{

//outtextxy(400,y,"Presiona ENTER para continuar");

//cleardevice();

//getch();

//y=100;

//}

}

}

}

**FUNCION LEE REGISTROS**

Esta función lo que hace básicamente es leer el archivo que contiene los registros de los jugadores.

void leeRegistros(char nomarch[], JUGADOR registros[NR])

{

int i;

FILE \*f;

for(i=0;i<NR;i++)

{ strcpy(registros[i].nombre," ");

registros[i].puntos=0;

}

f=fopen(nomarch,"rb+");

if(f==NULL)

{

f=fopen(nomarch,"wb");

fwrite(registros,sizeof(JUGADOR),NR,f);

}

else

fread(registros,sizeof(JUGADOR),NR,f);

fclose(f);

}

**FUNCION INT TEXT XY**

Esta función indica y muestra el texto que se está ingresando que a su vez es insertado en una cadena.

void intextxy( int x, int y, char texto[], int colf, int collet)

{

char car[2];

int i=0;

do{

while(!kbhit())

{

setcolor(collet);

outtextxy(x,y,"\_");

delay(50);

setcolor(colf);

outtextxy(x,y,"\_");

delay(50);

}

car[0]=getch();

car[1]='\0';

if(car[0]!='\b' && car[0]!='\r')

{

setcolor(collet);

outtextxy(x,y,car);

x+=textwidth(car);

texto[i++]=car[0];

}

else if(car[0]=='\b' && i>0)

{

car[0]=texto[--i];

car[1]='\0';

x-=textwidth(car);

setcolor(colf);

outtextxy(x,y,car);

}

} while(car[0]!='\r');

texto[i]='\0';

}

**FUNCION GAME OVER.**

Esta función lo que hace es mostrar el letrero que indica el fin del juego así como mandar llamar la función que guarda en archivo un registro ordenado de las puntuaciones correspondientes a cada jugador.

void game\_over(char nombre[], int pts)

{

JUGADOR jugador;

strcpy(jugador.nombre,nombre);

jugador.puntos = pts;

//setactivepage(0);

//setvisualpage(0);

cleardevice();

do{

setcolor(RED);

settextstyle(4,0,50);

outtextxy(getmaxx()/2-250,getmaxy()/2,"GAME OVER ");

setcolor(WHITE);

outtextxy(getmaxx()/2-250,getmaxy()/2,"GAME OVER ");

setcolor(12);

delay(85);

}while(!kbhit());

guardaRegistro("records",jugador);

//guardaRegistro("records", jugador);

delay(800);

}

**Función Guarda Registros**

Esta función lo que hace básicamente es guardar en archivo el nombre del jugador y su puntación así como la ordenación de estos recibiendo como parámetros el nombre del jugador y los puntos en una variable tipo jugador.

void guardaRegistro(char nomarch[],JUGADOR r)

{

JUGADOR registros[NR];

int i;

FILE \*f;

printf("jugador GAME %s",r.nombre);

printf("PUNTOS%d",r.puntos);

for(i=0;i<NR;i++)

{ strcpy(registros[i].nombre," ");

registros[i].puntos=0;

}

f=fopen(nomarch,"rb+");

if(f==NULL)

{ printf("No se encontro el archivo");

registros[0]=r;

f=fopen(nomarch,"wb");

fwrite(registros,sizeof(JUGADOR),NR,f);

fclose(f);

}

else

{

fread(registros,sizeof(JUGADOR),NR,f);

for(i=NR-1;i>=0;i--)

{ if(registros[i].puntos<=r.puntos && i+1<NR)

registros[i+1]=registros[i];

else

if(registros[i].puntos>=r.puntos && i+1<NR)

{ registros[i+1]=r;

i=-1;

}

}

if(registros[0].puntos<=r.puntos)

registros[0]=r;

rewind(f);

fwrite(registros,sizeof(JUGADOR),NR,f);

fclose(f);

}

}