Manual Tecnico
Space Invaders 1.0

Gilberto Rosales - 201318565
Sistemas Operativos 1

Introducción

Sppace invaders esta creado para que los mas peques del hogar puedan disfrutar de su consola linux. Este juego tambien puede ser disfrutado por los mas grandes del hogar.
Lo necesario para este juego se detalla en este pequeño pero necesario manual.

Programas a instalar

Librerias Linux

Codigo Fuente:

```
#include <ncurses.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include  <sys/shm.h>
#include <stdbool.h>

/************************************** DECLARACIONES ******************************/
        int main2();
        void print_in_middle(WINDOW *win, int starty, int startx, int width, char *string);
        void gameover100(int win);
        void welcome(WINDOW *win);
        void defensor(WINDOW *win);
        void dibuja_cuadro(WINDOW *win);
        void bando(WINDOW *win);
        void pintarCuadroJuego(WINDOW *win);
        void redibujarDefensor(WINDOW *win , int posX, int lado);
        int MEDIO;
        int numAllienDebil;
        int numAllienFuerte;
        int cantidadDiparos;
        void initGlobales();
        void initGlobales2();
        int pintarInvasores(WINDOW *win, int posX, int posY);
        void jugarDefensor(WINDOW *win);
        void repintarTodo(WINDOW *win );
        void generarDisparo(WINDOW *win);
        void actualizarDisparos(WINDOW *win);
        int pintarInvasorFuerte(WINDOW *win, int posX, int posY, int xx);
        int pintarInvasorDebil(WINDOW *win, int posX, int posY, int  xx , int cantidadADibujar);
        void actualizarTimer(WINDOW *win);
        int regresarParaMJ=6;

        //int puntajeDefensor =0;
        void DestruirNave(WINDOW *win, int nave);
        void borrardePantallaYMatriz(int nave);
        void actualizarPuntajeDefensor(WINDOW *win, int tipo);
        void redibujarInvasor(WINDOW *win , int posX, int lado);
        void esperarSemaforo1(WINDOW *win);
        void esperarSemaforo2(WINDOW *win);
        void invasor(WINDOW *win);
        void jugarInvasor(WINDOW *win);
        void generarDisparoInvasor(WINDOW *win, int nave);
        void actualizarDisparoInvasor(WINDOW *win);
        void actualizarVidas(WINDOW *win);
        void InitMemoriaCompartida(WINDOW *win);
        void actualizarCompartidos(WINDOW *win);

        void dekker(WINDOW *win);
        void dekker2(WINDOW *win);
        void RegionCritica(WINDOW *win);
```

```
        /******* semaforo*/
        key_t keySema;
        int idSema;
        struct sembuf Operacion;
        union semun arg;

_
/****************globales semaforos**********************/
        #if defined(__GNU_LIBRARY__) && !defined(_SEM_SEMUN_UNDEFINED)
        #else
        union semun
        {
                int val;
                struct semid_ds *buf;
                unsigned short int *array;
                struct seminfo *__buf;
        };
        #endif
        int numProceso=0;

/****************************************** STRUCTS ***********************************/
        struct disparo {
                int yp ;
                int x , y ;
                int s; /* 1=active 0=inactive */
                char ch;
        };
        struct alien {
                int idAlien;
                int x1,x2,x3,x4,x5,x6,x7,y;
                int x1p,x2p,x3p,x4p,x5p,x6p,x7p;
                int s; /* 1=active 0=inactive */
                char ch1,ch2,ch3,ch4,ch5,ch6,ch7;
        };

        struct comandante {
                int x1,x2,x3,x4,x5,x6,x7,y;
                int x1p,x2p,x3p,x4p,x5p,x6p,x7p;
                int s; /* 1=active 0=inactive */
                int vidas;
                char ch1,ch2,ch3,ch4,ch5,ch6,ch7;
        };

         struct alien debiles[17];
         struct alien fuertes[4];
        // struct disparo disparos[10];
         struct disparo disparosInvasores[100];
         int MJ[4][70];

int valyM = 5;
//struct comandante comandante_defensor;
//struct comandante comandante_invasor;
struct structMemoria {
                int status;
           int minutos;
           int segundos;
           int puntajeDefensor;
                struct comandante comandante_defensor;
                struct comandante comandante_invasor;
                struct disparo disparos[10];
```

```
        int posYDefensor;
        int posYInvasor;
        int posXDefensor;
        int posXInvasor;
        int vidasDefensor;
        int vidasInvasor;
                int ladoDEF;
                int termino100;
                //struct Dato data2[4];
          bool p1_puede_entrar, p2_puede_entrar;
          int turno;
        };


/*****************************globales memoria**************/
        key_t       Key;
        int         ShmID;
        struct structMemoria  *punteroMemoria;

        void restaurarValoresSem() {
                                        semctl (idSema, 1, SETVAL, 0);
                                        int r2=semctl(idSema, 1, GETVAL, 1);
                                        printf("%i valor con el que muere el segundo sem:₩n", r2);
                                        semctl (idSema, 0, SETVAL, 0);
                                        r2=semctl(idSema, 0, GETVAL, 0);
                                        printf("%i valor con el que muere el primer sem:₩n", r2);
        }

                void  levantarMemoriaCompartida() {
                  Key =  ftok ("/bin/ls", 6);
                  ShmID = shmget(Key, sizeof(struct structMemoria), IPC_CREAT | 0666);
                  if (ShmID < 0) {
                    printf("*** shmget error (server) ***₩n");
                                restaurarValoresSem();
                    exit(1);
                  }
                  //printf("Server has received a shared structMemoria of four integers...₩n");
                  punteroMemoria = (struct structMemoria *) shmat(ShmID, NULL, 0);
                  if ((int) punteroMemoria == -1) {
                     printf("*** shmat error (server) ***₩n");
                         restaurarValoresSem();
                     exit(1);
                  }
                  //printf("Server has attached the shared structMemoria...₩n");
                  punteroMemoria->status=0;
                  while (punteroMemoria->status != 1) {
                     sleep(1);
                  }
                 punteroMemoria->puntajeDefensor = 0;
                }
                void eliminarMemoriaCompartida() {
                  printf("Server has detected the completion of its child...₩n");
                  shmdt((void *) punteroMemoria);
                  printf("Server has detached its shared structMemoria...₩n");
                  shmctl(ShmID, IPC_RMID, NULL);
                  printf("Server has removed its shared structMemoria...₩n");
                  printf("Server exits...₩n");
                }
```

```
        void conectarAMemoriaCompartidaYaCreada() {
            Key = ftok ("/bin/ls", 6);
            ShmID = shmget(Key, sizeof(struct structMemoria), 0666);
            if (ShmID < 0) {
                printf("*** shmget error (client) ***\n");
                restaurarValoresSem();
                exit(1);
            }
            printf("  Client has received a shared structMemoria of four integers...\n");
            punteroMemoria = (struct structMemoria *) shmat(ShmID, NULL, 0);
            if ((int) punteroMemoria == -1) {
                printf("*** shmat error (client) ***\n");
                exit(1);
            }
            printf("  Client has attached the shared structMemoria...\n");
            punteroMemoria->status = 1;
        }

        void terminarConexionAMemoriaCompartida() {
            printf("  Client has informed server data have been taken...\n");
            shmdt((void *) punteroMemoria);
            printf("  Client has detached its shared structMemoria...\n");
            printf("  Client exits...\n");
        }
/********************************************** METODOS ****************************/
void initGlobales2() {
    for (int i = 0; i < 10; ++i)
    {
        punteroMemoria->disparos[i].s = 0;
        punteroMemoria->disparos[i].yp=3;
    }
    punteroMemoria->ladoDEF= 0;
    punteroMemoria->posXDefensor = 38;
    punteroMemoria->posXInvasor = 38;
    punteroMemoria->posYDefensor=20;
    punteroMemoria->posYInvasor=3;

    punteroMemoria->comandante_defensor.vidas = 5;
    punteroMemoria->comandante_defensor.ch1 = '<';
    punteroMemoria->comandante_defensor.ch2 = '<';
    punteroMemoria->comandante_defensor.ch3 = '=';
    punteroMemoria->comandante_defensor.ch4 = '=';
    punteroMemoria->comandante_defensor.ch5 = '=';
    punteroMemoria->comandante_defensor.ch6 = '>';
    punteroMemoria->comandante_defensor.ch7 = '>';
    punteroMemoria->comandante_defensor.x1p=3;
    punteroMemoria->comandante_defensor.x2p=3;
    punteroMemoria->comandante_defensor.x3p=3;
    punteroMemoria->comandante_defensor.x4p=3;
    punteroMemoria->comandante_defensor.x5p=3;
    punteroMemoria->comandante_defensor.x6p=3;
    punteroMemoria->comandante_defensor.x7p=3;
    punteroMemoria->comandante_defensor.y=20;

    punteroMemoria->comandante_invasor.vidas = 5;
    punteroMemoria->comandante_invasor.ch1 = '<';
    punteroMemoria->comandante_invasor.ch2 = '<';
    punteroMemoria->comandante_invasor.ch3 = '=';
    punteroMemoria->comandante_invasor.ch4 = '=';
```

```
	punteroMemoria->comandante_invasor.ch5 = '=';
	punteroMemoria->comandante_invasor.ch6 = ')';
	punteroMemoria->comandante_invasor.ch7 = ')';
	punteroMemoria->comandante_invasor.x1p=3;
	punteroMemoria->comandante_invasor.x2p=3;
	punteroMemoria->comandante_invasor.x3p=3;
	punteroMemoria->comandante_invasor.x4p=3;
	punteroMemoria->comandante_invasor.x5p=3;
	punteroMemoria->comandante_invasor.x6p=3;
	punteroMemoria->comandante_invasor.x7p=3;
	punteroMemoria->comandante_invasor.y=3;
	punteroMemoria->termino100= 0;
}

void initGlobales() {

	MEDIO = 38;

	cantidadDiparos = 0;

	for (int i = 0; i < 10; ++i)
	{
		disparosInvasores[i].s = 0;
	}
	numAllienDebil=0;
	numAllienFuerte=0;
	for (int i = 0; i < 4; ++i)
	{
		fuertes[i].idAlien = 0;
		fuertes[i].s = 1;
		fuertes[i].y= 3;
		fuertes[i].x1p= 3;
		fuertes[i].x2p=3;
		fuertes[i].x3p=3;
		fuertes[i].x4p=3;
		fuertes[i].x5p=3;
		fuertes[i].x6p=3;
		fuertes[i].x7p=3;
	}
	for (int i = 0; i < 17; ++i)
	{
		debiles[i].idAlien = 0;
		debiles[i].s = 1;
		debiles[i].y= 3;
		debiles[i].x1p= 3;
		debiles[i].x2p=3;
		debiles[i].x3p=3;
		debiles[i].x4p=3;
		debiles[i].x5p=3;
	}

	for (int i = 0; i < 4; ++i)
	{
		for (int i2 = 0; i2 < 70; ++i2)
		{
			MJ[i] [i2]  = 0;
		}
	}
```

```
}

unsigned int input;
int main2() {
        /*semaforo*/
        keySema = ftok ("/bin/ls", 4);
        if (keySema == (key_t)-1)
        {
                printf("No puedo conseguir key de semaforo₩n");
                exit(0);
        }
        printf("%i₩n",keySema );

        idSema = semget (keySema, 10, 0600 | IPC_CREAT);
        if (idSema == -1)
        {
                printf("No puedo crear semaforo₩n");
                exit (0);
        }
        /*pinta*/
        initscr();
        clear();
        noecho();
        cbreak();
        nodelay(stdscr,1);
        keypad(stdscr,1);
        srand(time(NULL));
        initGlobales();
        welcome(stdscr);
        endwin();
        /*termina de pintar*/

}
int main(int argc, char *argv[])
{
        main2();
        semctl (idSema, 1, SETVAL, 0);
        int r2=semctl(idSema, 1, GETVAL, 1);
        //printf("%i valor con el que muere el segundo sem:₩n", r2);
        semctl (idSema, 0, SETVAL, 0);
        r2=semctl(idSema, 0, GETVAL, 0);
        ///printf("%i valor con el que muere el primer sem:₩n", r2);
        if(numProceso==1) {
                eliminarMemoriaCompartida();
        }else{
                terminarConexionAMemoriaCompartida();
        }
  return 0;
}

void print_in_middle(WINDOW *win, int starty, int startx, int width, char *string)
{       int length, x, y;
        float temp;

        if(win == NULL)
```

```
        win = stdscr;
    getyx(win, y, x);
    if(startx != 0)
        x = startx;
    if(starty != 0)
        y = starty;
    if(width == 0)
        width = 80;

    length = strlen(string);
    temp = (width - length) / 2;
    x = startx + (int)temp;
    mvwprintw(win, y, x, "%s", string);
    refresh();
}

/*
void gameover100(int win) {
  nodelay(stdscr, 0);
  if (win == 0) {
    clear();
    move((LINES/2)-1,(COLS/2)-5);
    addstr("Perdiste");
    move((LINES/2),(COLS/2)-11);
    addstr("PRESS ANY KEY TO EXIT");
    move(0,COLS-1);
    refresh();
    getch();
  }

  else if (win == 1) {
    clear();
    move((LINES/2)-1,(COLS/2)-5);
    addstr("Ganaste!");
    move((LINES/2),(COLS/2)-11);
    addstr("PRESS ANY KEY TO EXIT");
    move(0,COLS-1);
    refresh();
    getch();
  }
}
*/

void gameover100(int win) {

  int key;
 while(1) {
  nodelay(stdscr, 0);
  if (win == 0) {
    clear();
    move((LINES/2)-1,(COLS/2)-5);
    addstr("Perdiste");
    move((LINES/2)-3,(COLS/2)-27);
    addstr("Presiona Enter si deseas continuar Jugando o e para salir");

      move((LINES/2)+2, 10);
      addstr("Estadisticas:");
    mvwprintw(stdscr, (LINES/2)+3, 10, "Tiempo: %d : %d ",punteroMemoria-
)minutos,punteroMemoria->segundos);
```

```
        mvwprintw(stdscr, (LINES/2)+4, 10, "Puntaje Defensor: (score) %d", punteroMemoria->puntajeDefensor);
        mvwprintw(stdscr, (LINES/2)+5, 10, "Vidas Defensor: %i" ,punteroMemoria->comandante_defensor.vidas);
        mvwprintw(stdscr, (LINES/2)+6, 10, "Vidas Comandante:%i", punteroMemoria->comandante_invasor.vidas);
    refresh();
     key = getch();
                if(key == 10){
                                    break;
                }else if(key == 'e'){
                                    break;
                }
  }
  else if (win == 1) {
    clear();
    move((LINES/2)-5,(COLS/2)-5);
    addstr("Ganaste!");
    move((LINES/2)-3,(COLS/2)-27);
    addstr("Presiona Enter si deseas continuar Jugando o e para salir");

          move((LINES/2)+2, 10);
          addstr("Estadisticas:");
          mvwprintw(stdscr, (LINES/2)+3, 10, "Tiempo: %d : %d ",punteroMemoria->minutos.punteroMemoria->segundos);
        mvwprintw(stdscr, (LINES/2)+4, 10, "Puntaje Defensor: (score) %d", punteroMemoria->puntajeDefensor);
        mvwprintw(stdscr, (LINES/2)+5, 10, "Vidas Defensor: %i" ,punteroMemoria->comandante_defensor.vidas);
        mvwprintw(stdscr, (LINES/2)+6, 10, "Vidas Comandante:%i", punteroMemoria->comandante_invasor.vidas);


  /*

        mvwprintw(win, 1, 10, "%s", ":");


        ***/

    refresh();
    key = getch();
                if(key == 10){
                                    break;
                }else if(key == 'e'){
                                    break;
                }
  }
   else if (win == 3) {
    clear();
    move((LINES/2)-1,(COLS/2)-5);
   // addstr("Ganaste!");
    move((LINES/2),(COLS/2)-7);
    addstr("VUELVE PRONTO");
    move(0,COLS-1);
     move((LINES/2)+2, 10);
          addstr("Estadisticas:");
   addstr("Estadisticas:");
```

```
        mvwprintw(stdscr, (LINES/2)+3, 10, "Tiempo: %d : %d ",punteroMemoria->minutos,punteroMemoria->segundos);
        mvwprintw(stdscr,(LINES/2)+4, 10, "Puntaje Defensor: (score) %d", punteroMemoria->puntajeDefensor);
        mvwprintw(stdscr, (LINES/2)+5, 10, "Vidas Defensor: %i" ,punteroMemoria->comandante_defensor.vidas);
        mvwprintw(stdscr, (LINES/2)+6, 10, "Vidas Comandante:%i", punteroMemoria->comandante_invasor.vidas);

    refresh();
    key = getch();
                if(key == 10) {
                        break;
                }else if(key == 'e') {
                                break;
                }
   }
}

        semctl (idSema, 1, SETVAL, 0);
        int r2=semctl(idSema, 1, GETVAL, 1);
        //printf("%i valor con el que muere el segundo sem:₩n", r2);
        semctl (idSema, 0, SETVAL, 0);
        r2=semctl(idSema, 0, GETVAL, 0);
        ///printf("%i valor con el que muere el primer sem:₩n", r2);
        if(numProceso==1) {
                eliminarMemoriaCompartida();
        }else{
                terminarConexionAMemoriaCompartida();
        }


        if(key == 10) {
                main2();
                }

}


void dibuja_cuadro(WINDOW *win) {                       // y - 24 , x -80
        mvwprintw(win, 0, 0, "%s", "*");
        for (int i = 1; i < 80-1; ++i)
        {
                mvwprintw(win, 0, i, "%s", "-");
        }
        mvwprintw(win, 0, 80-1, "%s", "*");
        for (int i = 1; i < 24-1; ++i)
        {
                mvwprintw(win, i, 0, "%s", "|");
        }
        mvwprintw(win, 24-1, 0, "%s", "*");
        for (int i = 1; i < 80-1; ++i)
        {
                mvwprintw(win, 24-1, i, "%s", "-");
        }
        mvwprintw(win, 24-1, 80-1, "%s", "*");
        for (int i = 1; i < 24-1; ++i)
        {
                mvwprintw(win, i, 80-1, "%s", "|");
```

```
        }

        refresh();
}

void welcome(WINDOW *win) {
        while(1) {
                if(has_colors() == FALSE)
                {       endwin();
                        printf("Your terminal does not support color₩n");
                        exit(1);
                }
                start_color();                  /* Start color                  */
                init_pair(1, COLOR_RED, COLOR_BLACK);  // mensaje inicial
                init_pair(2, COLOR_YELLOW , COLOR_RED); // tiempo punteo vidas
                init_pair(3, COLOR_BLUE , COLOR_BLACK); // defensor
                init_pair(4, COLOR_YELLOW  , COLOR_BLACK); // balas defensor
                init_pair(5, COLOR_GREEN , COLOR_BLACK); // balas defensor
                attron(COLOR_PAIR(1));

                move((20/2)-5,(80/2)-8);
          addstr("Space Invaders");
                attroff(COLOR_PAIR(1));
                attroff(COLOR_PAIR(1));

                attron(COLOR_PAIR(2));
                move((20/2)+5,(80/2)-20);
          addstr("Bienvenido presiona enter para ingresar!");
          move((20/2)+6,(80/2)-14);
          addstr("Gilberto Rosales - 201318565");
                attroff(COLOR_PAIR(2));

                dibuja_cuadro(win);
                int key = wgetch(win);
                if(key == 10) {
                                break;
                }
        }
        bando(win);
        //clear();
}
void bando(WINDOW *win) {
        int key;
        clear();
        while(1) {
                dibuja_cuadro(win);
          move((20/2)-5,(80/2)-14);
          addstr("Bienvenido Selecciona Bando!");
          move((20/2)-2,(80/2)-15);
          addstr("Presiona 1 para ser defensor...");
          move((20/2),(80/2)-14);
          addstr("Presiona 2 para ser invasor...");
          key = wgetch(stdscr);
                if(key == 49) {
                        break;
                }else if (key == 50) {
                        break;
                }
        }
```

```
        if(key == 49){
                esperarSemaforo1(win);
        }else if(key == 50){
                esperarSemaforo2(win);
        }
}
void esperarSemaforo1(WINDOW *win){
        int seConecto=0;
        clear();
        move((20/2)-2,(80/2)-15);
        addstr("Seleccionaste defensor: 1");
        move((20/2),(80/2)-14);
        //printf("Esperando jugador 2₩n");
        addstr("Espera mientras se conecta...");
        int re=semctl(idSema, 1, GETVAL, 2);
        //      mvwprintw(win, 10, 10, "val sem2: %i" , re);
        //printf("valor del sem2: %i₩n", re);
        refresh();
        dibuja_cuadro(win);
        while(1) {

          /*key = wgetch(stdscr);
                if(key == 10){
                        break;
                }*/

                if(re==2){
                        Operacion.sem_num = 0;
                        Operacion.sem_op = 1;
                        Operacion.sem_flg = 0;
                        //printf("saco del Semáforo ₩n");
                        numProceso = 2;
                        semop (idSema, &Operacion, 1);
                        sleep (1);
                        break;
                        //
                }else{
                        /*** para esperar al jugador */
                        arg.val = 0;int r=0;
                        semctl (idSema, 0, SETVAL, &arg);
                        Operacion.sem_num = 0;
                        Operacion.sem_op = -1;
                        Operacion.sem_flg = 0;
                        numProceso = 1;
                        while (1)
                        {
                                semctl (idSema, 1, SETVAL, 2);
                                semop (idSema, &Operacion, 1);
                                //printf("Se conecto el jugador₩n");
                                seConecto =1;
                                break;
                        }

                }
                if(seConecto==1){
                        break;
                }
        }
        InitMemoriaCompartida(win);
```

```
	defensor(win);
}
void esperarSemaforo2(WINDOW *win){
	int seConecto=0;
	clear();
	move((20/2)-2,(80/2)-15);
	addstr("Seleccionaste: 2");
	move((20/2),(80/2)-14);
	printf("Esperando jugador 1₩n");
	addstr("Espera mientras se conecta...");
	int re=semctl(idSema, 1, GETVAL, 2);
	//mvwprintw(win, 10, 10, "val sem2: %i" , re);
	//	printf("valor del sem2: %i₩n", re);
	refresh();
	dibuja_cuadro(win);
	while(1) {
		if(re==2){
			Operacion.sem_num = 0;
			Operacion.sem_op = 1;
			Operacion.sem_flg = 0;
			//printf("saco del Semáforo ₩n");
			numProceso = 2;
			semop (idSema, &Operacion, 1);
			sleep (1);
			break;

		}else{
			/*** para esperar al jugador */
			arg.val = 0;int r=0;
			semctl (idSema, 0, SETVAL, &arg);
			Operacion.sem_num = 0;
			Operacion.sem_op = -1;
			Operacion.sem_flg = 0;
			numProceso = 1;
			while (1)
			{
				semctl (idSema, 1, SETVAL, 2);
				semop (idSema, &Operacion, 1);
				//printf("Se conecto el jugador₩n");
				seConecto =1;
				break;
			}

		}
		if(seConecto==1){
			break;
		}
	}
	InitMemoriaCompartida(win);
	invasor(win);
}

void InitMemoriaCompartida(WINDOW *win){
	if(numProceso==1){
		levantarMemoriaCompartida();
		initGlobales2();
	}else{
		conectarAMemoriaCompartidaYaCreada();
	}
```

```
}

void defensor(WINDOW *win) {
        clear();
        dibuja_cuadro(win);
        pintarCuadroJuego(win);
        actualizarVidas(win);
        //redibujarDefensor(win,MEDIO,0);
        //redibujarInvasor(win,MEDIO,0);
        pintarInvasores(win,valyM,5);
        jugarDefensor(win);
}

int valyMinvasor = 5;
void invasor(WINDOW *win) {
        clear();
        dibuja_cuadro(win);
        pintarCuadroJuego(win);

        actualizarVidas(win);
        //redibujarDefensor(win,MEDIO,0);
        //redibujarInvasor(win,MEDIO,0);
        pintarInvasores(win,valyMinvasor,5);
        jugarInvasor(win);

  //getch();
}

void jugarInvasor(WINDOW *win) {
        punteroMemoria->comandante_invasor.vidas=5;
        int key=0;
        int lado = 0;
        int contadorTimer = 20;
        char ladoc = 'd';
        generarDisparoInvasor(win,17);
        while(1) {
                /** CODIGO DEKKER*/
                punteroMemoria->p2_puede_entrar = true;
    while( punteroMemoria->p1_puede_entrar ) {
        if( punteroMemoria->turno == 1 ) {
            punteroMemoria->p2_puede_entrar = false;
            while( punteroMemoria->turno == 1 ) {}
            punteroMemoria->p2_puede_entrar = true;
        }
    }
    /*************************************************************region critica*/
    redibujarInvasor(win,punteroMemoria->posXInvasor,lado);
    redibujarDefensor(win,punteroMemoria->posXDefensor,punteroMemoria->ladoDEF);
    punteroMemoria->turno = 1;
    punteroMemoria->p2_puede_entrar = false;
                if(contadorTimer ==20) {
                        actualizarTimer(win);
                        contadorTimer=0;
                        if(valyMinvasor==30) {
                                ladoc='i';
                        }else if(valyMinvasor==5) {
                                ladoc='d';
                        }
                        pintarInvasores(win,valyMinvasor,5);
```

```
                    if(ladoc=='d'){
                        valyMinvasor++;
                    }else{
                        valyMinvasor--;
                    }
                }else{
                    actualizarCompartidos(win);
                }

        //      usleep(1000000/20);
                ++contadorTimer;
                key = getch();
                //redibujarDefensor(win,punteroMemoria->posXDefensor,punteroMemoria->ladoDEF);

                if (key == 'o'){//izquierda 99-c
                if(punteroMemoria->posXInvasor>2){
                    --punteroMemoria->posXInvasor;
                    ///lado = 1;
                }
        }else if (key == 'p'){ // derechar 118- v
                    if(punteroMemoria->posXInvasor<71){
                ++punteroMemoria->posXInvasor;
                    //lado = 0;
                }
        } else if (key == 'e'){
                break;
        }else if(key=='1'){
                    generarDisparoInvasor(win,17);
        }else if(key=='2'){
                    generarDisparoInvasor(win,18);
        }else if(key=='3'){
                    generarDisparoInvasor(win,19);
        }else if(key=='4'){
                    generarDisparoInvasor(win,20);
        }

                /************************************ termina region critica************/
                punteroMemoria->turno = 1;
    punteroMemoria->p2_puede_entrar = false;
    usleep(1000000/20);
                if(punteroMemoria->termino100 ==1|| punteroMemoria->comandante_invasor.vidas==0){
                    break;
                }
  }
  if(punteroMemoria->termino100 ==1 || punteroMemoria->comandante_invasor.vidas==0){
                gameover100(1);
  }
   if(key=='e'){
                    gameover100(3);
       }

}

void jugarDefensor(WINDOW *win){
        punteroMemoria->comandante_invasor.vidas=5;
        int key=0;
        int lado = 0;
        //int lado= 0;//izquierda
        int contadorTimer = 20;
```

```
        char ladoc = 'd';
        while(1) {

                /* DEKKER PARA UN PROCESO*/
                punteroMemoria->p1_puede_entrar = true;
    while( punteroMemoria->p2_puede_entrar ) {
        if( punteroMemoria->turno == 2 ) {
            punteroMemoria->p1_puede_entrar = false;
            while( punteroMemoria->turno == 2 ) {}
            punteroMemoria->p1_puede_entrar = true;
        }
    }
                /*Region critica*/
    redibujarDefensor(win,punteroMemoria->posXDefensor,punteroMemoria->ladoDEF);
    redibujarInvasor(win,punteroMemoria->posXInvasor,lado);

    if(contadorTimer ==20){
                        actualizarTimer(win);
                        contadorTimer=0;
                        if(valyM==30){
                                ladoc='i';
                        }else if(valyM==5){
                                ladoc='d';
                        }
                                pintarInvasores(win,valyM,5);
                        if(ladoc=='d'){
                                valyM++;
                        }else{
                                valyM--;
                                //pintarInvasores(win,valyM,5);
                        }

                }else{
                        actualizarDisparos(win);
                        actualizarCompartidos(win);
                }
        //      usleep(1000000/20);
                ++contadorTimer;
                key = getch();

                if (key == 'x'){//izquierda 99-c
                if(     punteroMemoria->posXDefensor>2){
                        --      punteroMemoria->posXDefensor;
                        //redibujarDefensor(win,posXDefensor,lado);
                        punteroMemoria->ladoDEF= 1;
                }
        }else if (key == 118){ // derechar 118- v
                        if(     punteroMemoria->posXDefensor<71){
                ++      punteroMemoria->posXDefensor;

                        punteroMemoria->ladoDEF = 0;
                }
        } else if (key == 'e'){
                break;
        }else if(key=='c'){
                        generarDisparo(win);
        }

    punteroMemoria->turno = 2;
```

```
    punteroMemoria->p1_puede_entrar = false;
    usleep(1000000/20);
    mvwprintw(win, 10, 10, "%d", punteroMemoria->termino100);
    if(punteroMemoria->termino100 ==1|| punteroMemoria->comandante_invasor.vidas==0){
                    break;
                    }


  }

      if(punteroMemoria->termino100 ==1 || punteroMemoria->comandante_invasor.vidas==0){
              gameover100(1);
  }
  if(key=='e'){
                    gameover100(3);
      }
}

void dekker(WINDOW *win){
  while( true ){
    /*[REALIZA_TAREAS_INICIALES]*/

    //[REALIZA_TAREAS_FINALES]
    break;
  }
}

void dekker2(WINDOW *win){
  while( true ){
    /*[REALIZA_TAREAS_INICIALES]*/

    //[REALIZA_TAREAS_FINALES]
    break;
  }
}

void generarDisparoInvasor(WINDOW *win, int nave){
      int hayEspacio=0;
      int i;
      for (i = 0; i < 10; ++i)
      {
            if(disparosInvasores[i].s ==0){
                  hayEspacio = 1;
                  break;
            }
      }

      for (i = 0; i < 4; ++i)
      {
                  if(fuertes[i].idAlien==nave){
                        break;
                  }

      }

      if(hayEspacio==1){
            disparosInvasores[i].x = fuertes[i].x4;
            disparosInvasores[i].y = fuertes[i].y+1;
            disparosInvasores[i].s = 1;
```

```
                disparosInvasores[i].ch = '|';

    attron(COLOR_PAIR(5));
                move(disparosInvasores[i].y,disparosInvasores[i].x);
    addch(disparosInvasores[i].ch);
                attroff(COLOR_PAIR(5));
                //refresh();
        }
}

void actualizarDisparoInvasor(WINDOW *win){
        for (int i = 0; i < 10; ++i)
        {
                if(disparosInvasores[i].s == 1){
                        if(disparosInvasores[i].y==19){
                                if(disparosInvasores[i].x==     punteroMemoria->comandante_defensor.x1p)
{
                                }else if (disparosInvasores[i].x==      punteroMemoria-
>comandante_defensor.x2p){
                                        --punteroMemoria->comandante_defensor.vidas;
                                        actualizarVidas(win);
                                }else if (disparosInvasores[i].x==      punteroMemoria-
>comandante_defensor.x3p){
                                        --punteroMemoria->comandante_defensor.vidas;
                                        actualizarVidas(win);
                                }else if (disparosInvasores[i].x==      punteroMemoria-
>comandante_defensor.x4p){
                                        --punteroMemoria->comandante_defensor.vidas;
                                        actualizarVidas(win);
                                }else if (disparosInvasores[i].x==      punteroMemoria-
>comandante_defensor.x5p){
                                        --punteroMemoria->comandante_defensor.vidas;
                                        actualizarVidas(win);
                                }else if (disparosInvasores[i].x==      punteroMemoria-
>comandante_defensor.x6p){
                                        --punteroMemoria->comandante_defensor.vidas;
                                        actualizarVidas(win);
                                }else if (disparosInvasores[i].x==      punteroMemoria-
>comandante_defensor.x7p){
                                        --punteroMemoria->comandante_defensor.vidas;
                                        actualizarVidas(win);
                                }
                                disparosInvasores[i].s=0;
                                move(disparosInvasores[i].y,disparosInvasores[i].x);
                        addch(' ');
                        }else{
                                if(disparosInvasores[i].y<19){
                                        attron(COLOR_PAIR(5));
                                        move(disparosInvasores[i].y,disparosInvasores[i].x);
                                addch(' ');
                                        disparosInvasores[i].y = disparosInvasores[i].y+1;
                                        move(disparosInvasores[i].y,disparosInvasores[i].x);
                                addch(disparosInvasores[i].ch);
                                        attroff(COLOR_PAIR(5));
                                }else{
                                        move(disparosInvasores[i].y,disparosInvasores[i].x);
                                addch(' ');
                                        disparosInvasores[i].s = 0;
                                }
```

```
                    }
                }
            }
        }
}

void redibujarDefensor(WINDOW *win , int posX, int lado) {
            attron(COLOR_PAIR(3));
                    move(punteroMemoria->comandante_defensor.y, punteroMemoria->comandante_defensor.x1p);
                    addch(' ');
                    move(punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x2p);
                    addch(' ');
                    move( punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x3p);
                    addch(' ');
                    move( punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x4p);
                    addch(' ');
                    move( punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x5p);
                    addch(' ');
                    move( punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x6p);
                    addch(' ');
                    move( punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x7p);
                    addch(' ');
                            punteroMemoria->comandante_defensor.x1 = posX;
                            punteroMemoria->comandante_defensor.x2 = posX+1;
                            punteroMemoria->comandante_defensor.x3 = posX+2;
                            punteroMemoria->comandante_defensor.x4 = posX+3;
                            punteroMemoria->comandante_defensor.x5 = posX+4;
                            punteroMemoria->comandante_defensor.x6 = posX+5;
                            punteroMemoria->comandante_defensor.x7 = posX+6;
                    move( punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x1);
                    addch( punteroMemoria->comandante_defensor.ch1);
                    move( punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x2);
                    addch( punteroMemoria->comandante_defensor.ch2);
                    move( punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x3);
                    addch( punteroMemoria->comandante_defensor.ch3);
                    move( punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x4);
                    addch( punteroMemoria->comandante_defensor.ch4);
                    move( punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x5);
                    addch( punteroMemoria->comandante_defensor.ch5);
                    move( punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x6);
                    addch( punteroMemoria->comandante_defensor.ch6);
                    move( punteroMemoria->comandante_defensor.y,  punteroMemoria->comandante_defensor.x7);
                    addch( punteroMemoria->comandante_defensor.ch7);
```

```
                punteroMemoria->comandante_defensor.x1p = punteroMemoria->comandante_defensor.x1;
                punteroMemoria->comandante_defensor.x2p = punteroMemoria->comandante_defensor.x2;
                punteroMemoria->comandante_defensor.x3p = punteroMemoria->comandante_defensor.x3;
                punteroMemoria->comandante_defensor.x4p = punteroMemoria->comandante_defensor.x4;
                punteroMemoria->comandante_defensor.x5p = punteroMemoria->comandante_defensor.x5;
                punteroMemoria->comandante_defensor.x6p = punteroMemoria->comandante_defensor.x6;
                punteroMemoria->comandante_defensor.x7p = punteroMemoria->comandante_defensor.x7;

        attroff(COLOR_PAIR(3));

}

void redibujarInvasor(WINDOW *win , int posX, int lado) {
        //repintarTodo(win);
        attron(COLOR_PAIR(5));
        move( punteroMemoria->comandante_invasor.y,  punteroMemoria->comandante_invasor.x1p);
                addch(' ');
                move( punteroMemoria->comandante_invasor.y,  punteroMemoria->comandante_invasor.x2p);
                addch(' ');
                move( punteroMemoria->comandante_invasor.y,  punteroMemoria->comandante_invasor.x3p);
                addch(' ');
                move( punteroMemoria->comandante_invasor.y,  punteroMemoria->comandante_invasor.x4p);
                addch(' ');
                move( punteroMemoria->comandante_invasor.y,  punteroMemoria->comandante_invasor.x5p);
                addch(' ');
                move( punteroMemoria->comandante_invasor.y,  punteroMemoria->comandante_invasor.x6p);
                addch(' ');
                move( punteroMemoria->comandante_invasor.y,  punteroMemoria->comandante_invasor.x7p);
                addch(' ');
                        punteroMemoria->comandante_invasor.x1 = posX;
                        punteroMemoria->comandante_invasor.x2 = posX+1;
                        punteroMemoria->comandante_invasor.x3 = posX+2;
                        punteroMemoria->comandante_invasor.x4 = posX+3;
                        punteroMemoria->comandante_invasor.x5 = posX+4;
                        punteroMemoria->comandante_invasor.x6 = posX+5;
                        punteroMemoria->comandante_invasor.x7 = posX+6;
                move( punteroMemoria->comandante_invasor.y,  punteroMemoria->comandante_invasor.x1);
                addch( punteroMemoria->comandante_invasor.ch1);
                move( punteroMemoria->comandante_invasor.y,  punteroMemoria->comandante_invasor.x2);
                addch( punteroMemoria->comandante_invasor.ch2);
                move( punteroMemoria->comandante_invasor.y,  punteroMemoria->comandante_invasor.x3);
                addch( punteroMemoria->comandante_invasor.ch3);
```

```
                move( punteroMemoria->comandante_invasor.y,     punteroMemoria->comandante_invasor.x4);
                addch( punteroMemoria->comandante_invasor.ch4);
                move( punteroMemoria->comandante_invasor.y,     punteroMemoria->comandante_invasor.x5);
                addch( punteroMemoria->comandante_invasor.ch5);
                move( punteroMemoria->comandante_invasor.y,     punteroMemoria->comandante_invasor.x6);
                addch( punteroMemoria->comandante_invasor.ch6);
                move( punteroMemoria->comandante_invasor.y,     punteroMemoria->comandante_invasor.x7);
                addch( punteroMemoria->comandante_invasor.ch7);
                                punteroMemoria->comandante_invasor.x1p =
        punteroMemoria->comandante_invasor.x1;
                                punteroMemoria->comandante_invasor.x2p =
        punteroMemoria->comandante_invasor.x2;
                                punteroMemoria->comandante_invasor.x3p =
        punteroMemoria->comandante_invasor.x3;
                                punteroMemoria->comandante_invasor.x4p =
        punteroMemoria->comandante_invasor.x4;
                                punteroMemoria->comandante_invasor.x5p =
        punteroMemoria->comandante_invasor.x5;
                                punteroMemoria->comandante_invasor.x6p =
        punteroMemoria->comandante_invasor.x6;
                                punteroMemoria->comandante_invasor.x7p =
        punteroMemoria->comandante_invasor.x7;
                attroff(COLOR_PAIR(5));
}

void generarDisparo(WINDOW *win){
        int hayEspacio=0;
        int i;
        for (i = 0; i < 10; ++i)
        {
                if(punteroMemoria->disparos[i].s ==0){
                        hayEspacio = 1;
                        break;
                }
        }
        if(hayEspacio==1){
                punteroMemoria->disparos[i].x = punteroMemoria->posXDefensor + 3;
                punteroMemoria->disparos[i].y = punteroMemoria->posYDefensor-1;
                punteroMemoria->disparos[i].yp = punteroMemoria->disparos[i].y;
                punteroMemoria->disparos[i].s = 1;
                punteroMemoria->disparos[i].ch = '*';
                //
    attron(COLOR_PAIR(4));
                move(punteroMemoria->disparos[i].y,punteroMemoria->disparos[i].x);
    addch(punteroMemoria->disparos[i].ch);
                attroff(COLOR_PAIR(4));
                refresh();
        }
}

void actualizarDisparos(WINDOW *win){
        for (int i = 0; i < 10; ++i)
        {
        //punteroMemoria->disparos[i] =       punteroMemoria->punteroMemoria->disparos[i];
                if(punteroMemoria->disparos[i].s == 1){
```

```
                    if(punteroMemoria->disparos[i].y==4){
                        if(punteroMemoria->disparos[i].x==punteroMemoria->comandante_invasor.x1p){
                        }else if (punteroMemoria->disparos[i].x==punteroMemoria->comandante_invasor.x2p){
                            --punteroMemoria->comandante_invasor.vidas;
                            actualizarVidas(win);
                        }else if (punteroMemoria->disparos[i].x==punteroMemoria->comandante_invasor.x3p){
                            --punteroMemoria->comandante_invasor.vidas;
                            actualizarVidas(win);
                        }else if (punteroMemoria->disparos[i].x==punteroMemoria->comandante_invasor.x4p){
                            --punteroMemoria->comandante_invasor.vidas;
                            actualizarVidas(win);
                        }else if (punteroMemoria->disparos[i].x==punteroMemoria->comandante_invasor.x5p){
                            --punteroMemoria->comandante_invasor.vidas;
                            actualizarVidas(win);
                        }else if (punteroMemoria->disparos[i].x==punteroMemoria->comandante_invasor.x6p){
                            --punteroMemoria->comandante_invasor.vidas;
                            actualizarVidas(win);
                        }else if (punteroMemoria->disparos[i].x==punteroMemoria->comandante_invasor.x7p){
                            --punteroMemoria->comandante_invasor.vidas;
                            actualizarVidas(win);
                        }
                        punteroMemoria->disparos[i].s=0;
                        move(punteroMemoria->disparos[i].y,punteroMemoria->disparos[i].x);
                    addch(' ');
                    }else{
                        if(punteroMemoria->disparos[i].y>4){
                            attron(COLOR_PAIR(4));

                            move(punteroMemoria->disparos[i].yp,punteroMemoria->disparos[i].x);
                        addch(' ');
                            punteroMemoria->disparos[i].y = punteroMemoria->disparos[i].y-1;
                            punteroMemoria->disparos[i].yp=punteroMemoria->disparos[i].y;

                            if(punteroMemoria->disparos[i].y==5){
                                if(MJ[0] [punteroMemoria->disparos[i].x-regresarParaMJ]==0){
                                    move(punteroMemoria->disparos[i].y,punteroMemoria->disparos[i].x);
                                    addch(punteroMemoria->disparos[i].ch);
                                    }else{
                                    //printf("%i¥n",MJ[0] [punteroMemoria->disparos[i].x-regresarParaMJ]);
                                    punteroMemoria->disparos[i].s=0;
                                    DestruirNave(win,MJ[0] [punteroMemoria->disparos[i].x-regresarParaMJ]);
                                }
                            }else if(punteroMemoria->disparos[i].y==7){
                                if(MJ[1] [punteroMemoria->disparos[i].x-regresarParaMJ]==0){
                                    move(punteroMemoria->disparos[i].y,punteroMemoria->disparos[i].x);
```

```
                                addch(punteroMemoria->disparos[i].ch);
                            }else{
                                //printf("%i¥n",MJ[1][punteroMemoria->disparos[i].x-regresarParaMJ]);
                                punteroMemoria->disparos[i].s=0;
                                DestruirNave(win,MJ[1][punteroMemoria->disparos[i].x-regresarParaMJ]);
                            }
                        }else if(punteroMemoria->disparos[i].y==9){
                            if(MJ[2][punteroMemoria->disparos[i].x-regresarParaMJ]==0){
                                move(punteroMemoria->disparos[i].y,punteroMemoria->disparos[i].x);
                                addch(punteroMemoria->disparos[i].ch);
                            }else{
                                //printf("%i¥n",MJ[2][punteroMemoria->disparos[i].x-regresarParaMJ]);
                                punteroMemoria->disparos[i].s=0;
                                DestruirNave(win,MJ[2][punteroMemoria->disparos[i].x-regresarParaMJ]);
                            }
                        }else if(punteroMemoria->disparos[i].y==11){
                            if(MJ[3][punteroMemoria->disparos[i].x-regresarParaMJ]==0){
                                move(punteroMemoria->disparos[i].y,punteroMemoria->disparos[i].x);
                                addch(punteroMemoria->disparos[i].ch);
                            }else{
                                //printf("%i¥n",MJ[3][punteroMemoria->disparos[i].x-regresarParaMJ]);
                                punteroMemoria->disparos[i].s=0;
                                DestruirNave(win,MJ[3][punteroMemoria->disparos[i].x-regresarParaMJ]);
                            }
                        }else{
                            move(punteroMemoria->disparos[i].y,punteroMemoria->disparos[i].x);
                    addch(punteroMemoria->disparos[i].ch);
                        }
                        attroff(COLOR_PAIR(4));

                    }else{
                            move(punteroMemoria->disparos[i].yp,punteroMemoria->disparos[i].x);
                    addch(' ');
                            punteroMemoria->disparos[i].s = 0;
                    }
                }


            }
        }
}

void DestruirNave(WINDOW *win, int nave){
    if(nave<17){//debiles
        for (int i = 0; i < 16; ++i)
        {
            if(debiles[i].idAlien==nave){
```

```
                    debiles[i].s=0;
                    move(debiles[i].y, debiles[i].x1p);
                    addch(' ');
                    move(debiles[i].y, debiles[i].x2p);
                    addch(' ');
                    move(debiles[i].y, debiles[i].x3p);
                    addch(' ');
                    move(debiles[i].y, debiles[i].x4p);
                    addch(' ');
                    move(debiles[i].y, debiles[i].x5p);
                    addch(' ');
                    borrardePantallaYMatriz(nave);
                    break;
                }

            }
            actualizarPuntajeDefensor(win,0);
        }else{//fuertes
            for (int i = 0; i < 4; ++i)
            {
                if(fuertes[i].idAlien==nave){
                    fuertes[i].s=0;
                    move(fuertes[i].y, fuertes[i].x1p);
                    addch(' ');
                    move(fuertes[i].y, fuertes[i].x2p);
                    addch(' ');
                    move(fuertes[i].y, fuertes[i].x3p);
                    addch(' ');
                    move(fuertes[i].y, fuertes[i].x4p);
                    addch(' ');
                    move(fuertes[i].y, fuertes[i].x5p);
                    addch(' ');
                    move(fuertes[i].y, fuertes[i].x6p);
                    addch(' ');
                    move(fuertes[i].y, fuertes[i].x7p);
                    addch(' ');
                    borrardePantallaYMatriz(nave);
                    break;
                }

            }
          actualizarPuntajeDefensor(win,1);
        }

}
void borrardePantallaYMatriz(int nave){
        for (int i = 0; i < 4; ++i)
                {
                        for (int i2 = 0; i2 < 70; ++i2)
                {
                                if(MJ[i][i2]==nave){
                                        MJ[i][i2] =0;
                }

                }
                }
}

void pintarCuadroJuego(WINDOW *win){
```

```
        for (int i = 1; i < 80-2; ++i)
        {
                mvwprintw(win, 2, i, "%s", "-");
        }
        for (int i = 1; i < 80-2; ++i)
        {
                mvwprintw(win, 24-3, i, "%s", "-");
        }

        attron(COLOR_PAIR(2));
        mvwprintw(win, 1, 1, "%s", "Tiempo:");
        mvwprintw(win, 1, 60, "%s", "Vidas invasor:");
        mvwprintw(win, 24-2, 1, "%s", "Punteo:");

        mvwprintw(win, 24-2, 80-20, "%s", "Vidas defensor:");
        attroff(COLOR_PAIR(2));
        attron(COLOR_PAIR(4));
        mvwprintw(win, 24-2, 8, "%d", punteroMemoria->puntajeDefensor);
        attroff(COLOR_PAIR(4));

        refresh();
}

void repintarTodo(WINDOW *win ) {
        clear();
        dibuja_cuadro(win);
        pintarCuadroJuego(win);
}

int pintarInvasores(WINDOW *win, int posX, int posY) {
        int xx = 0;
        //************************************ PRIMERA LINEA
        xx =  pintarInvasorDebil(win,posX,posY,xx,2);
        xx = pintarInvasorFuerte(win,posX, posY, xx );
        xx =pintarInvasorDebil(win,posX,posY,xx,2);
        //************************************ SEGUNDA LINEA
        xx = 0; ++posY; ++posY;
        xx =  pintarInvasorDebil(win,posX,posY,xx,4);
        xx = pintarInvasorFuerte(win,posX, posY, xx );
                //************************************ TERCERA LINEA
        xx = 0; ++posY; ++posY;
        xx =  pintarInvasorDebil(win,posX,posY,xx,3);
        xx = pintarInvasorFuerte(win,posX, posY, xx );
        xx =  pintarInvasorDebil(win,posX,posY,xx,1);

                //************************************ CUARTA LINEA
        xx = 0; ++posY; ++posY;
        xx =  pintarInvasorDebil(win,posX,posY,xx,1);
        xx = pintarInvasorFuerte(win,posX, posY, xx );
        xx =pintarInvasorDebil(win,posX,posY,xx,3);
        numAllienDebil = 0;
        numAllienFuerte = 0;
}


int pintarInvasorDebil(WINDOW *win, int posX, int posY, int  xx , int cantidadADibujar) {
        for (int i = 0; i < cantidadADibujar; ++i)
                {
                        int i2=numAllienDebil;
```

```
                    if(debiles[i2].s == 1){
                        int columnas = 0;
                        if(posY==5){
                            columnas=0;
                        }else if(posY==7){
                            columnas=1;
                        }else if(posY==9){
                            columnas=2;
                        }else if(posY==11){
                            columnas=3;
                        }
                        if(debiles[i2].x1-regresarParaMJ>=0){
                            MJ[columnas][debiles[i2].x1p-regresarParaMJ] = 0;
                            MJ[columnas][debiles[i2].x2p-regresarParaMJ] = 0;
                            MJ[columnas][debiles[i2].x3p-regresarParaMJ] = 0;
                            MJ[columnas][debiles[i2].x4p-regresarParaMJ] = 0;
                            MJ[columnas][debiles[i2].x5p-regresarParaMJ] = 0;
                        }
                        move(debiles[i2].y, debiles[i2].x1p);
                        addch(' ');
                        move(debiles[i2].y, debiles[i2].x2p);
                        addch(' ');
                        move(debiles[i2].y, debiles[i2].x3p);
                        addch(' ');
                        move(debiles[i2].y, debiles[i2].x4p);
                        addch(' ');
                        move(debiles[i2].y, debiles[i2].x5p);
                        addch(' ');
                        debiles[i2].idAlien = numAllienDebil+1;
                        debiles[i2].ch1= '₩₩';
                        debiles[i2].ch2= '-';
                        debiles[i2].ch3= '.';
                        debiles[i2].ch4= '-';
                        debiles[i2].ch5= '/';
                        debiles[i2].y = posY;
                        debiles[i2].x1= posX+xx; ++xx;
                        debiles[i2].x1= posX + xx ;++xx;
                        debiles[i2].x2= posX + xx ;++xx;
                        debiles[i2].x3= posX + xx ;++xx;
                        debiles[i2].x4= posX + xx ;++xx;
                        debiles[i2].x5= posX + xx ;
                        xx=xx+3;
                        /* ESCRIBIR */
                        attron(COLOR_PAIR(5));
                        move(debiles[i2].y, debiles[i2].x1);
                        addch(debiles[i2].ch1);
                        move(debiles[i2].y, debiles[i2].x2);
                        addch(debiles[i2].ch2);
                        move(debiles[i2].y, debiles[i2].x3);
                        addch(debiles[i2].ch3);
                        move(debiles[i2].y, debiles[i2].x4);
                        addch(debiles[i2].ch4);
                        move(debiles[i2].y, debiles[i2].x5);
                        addch(debiles[i2].ch5);
                        /* guardar la posAnterior*/
                        debiles[i2].x1p= debiles[i2].x1;
                        debiles[i2].x2p= debiles[i2].x2;
                        debiles[i2].x3p= debiles[i2].x3;
                        debiles[i2].x4p= debiles[i2].x4;
```

```
                    debiles[i2].x5p= debiles[i2].x5;
                    /*actualizar matriz de valores*/
                    MJ[columnas] [debiles[i2].x1-regresarParaMJ] = debiles[i2].idAlien;
                    MJ[columnas] [debiles[i2].x2-regresarParaMJ] = debiles[i2].idAlien;
                    MJ[columnas] [debiles[i2].x3-regresarParaMJ] = debiles[i2].idAlien;
                    MJ[columnas] [debiles[i2].x4-regresarParaMJ] = debiles[i2].idAlien;
                    MJ[columnas] [debiles[i2].x5-regresarParaMJ] = debiles[i2].idAlien;

                    attroff(COLOR_PAIR(5));
                }else{
                    move(debiles[i2].y, debiles[i2].x1p);
                    addch(' ');
                    move(debiles[i2].y, debiles[i2].x2p);
                    addch(' ');
                    move(debiles[i2].y, debiles[i2].x3p);
                    addch(' ');
                    move(debiles[i2].y, debiles[i2].x4p);
                    addch(' ');
                    move(debiles[i2].y, debiles[i2].x5p);
                    addch(' ');
                    debiles[i2].idAlien = numAllienDebil+1;
                    debiles[i2].ch1= ' ';
                    debiles[i2].ch2= ' ';
                    debiles[i2].ch3= ' ';
                    debiles[i2].ch4= ' ';
                    debiles[i2].ch5= ' ';
                    debiles[i2].y = posY;
                    debiles[i2].x1= posX+xx; ++xx;
                    debiles[i2].x1= posX + xx ;++xx;
                    debiles[i2].x2= posX + xx ;++xx;
                    debiles[i2].x3= posX + xx ;++xx;
                    debiles[i2].x4= posX + xx ;++xx;
                    debiles[i2].x5= posX + xx ;
                    xx=xx+3;
                    /* ESCRIBIR */
                    attron(COLOR_PAIR(5));
                    move(debiles[i2].y, debiles[i2].x1);
                    addch(debiles[i2].ch1);
                    move(debiles[i2].y, debiles[i2].x2);
                    addch(debiles[i2].ch2);
                    move(debiles[i2].y, debiles[i2].x3);
                    addch(debiles[i2].ch3);
                    move(debiles[i2].y, debiles[i2].x4);
                    addch(debiles[i2].ch4);
                    move(debiles[i2].y, debiles[i2].x5);
                    addch(debiles[i2].ch5);
                    /* guardar la posAnterior*/
                    debiles[i2].x1p= debiles[i2].x1;
                    debiles[i2].x2p= debiles[i2].x2;
                    debiles[i2].x3p= debiles[i2].x3;
                    debiles[i2].x4p= debiles[i2].x4;
                    debiles[i2].x5p= debiles[i2].x5;

                }
                numAllienDebil++;
            }
            return xx;
}
```

```
int pintarInvasorFuerte(WINDOW *win, int posX, int posY, int xx) {
                int i= numAllienFuerte;
                if(fuertes[i].s == 1) {
                        int columnas = 0;
                        //printf("%i₩n",posY);
                        if(posY==5) {
                                columnas=0;
                        }else if(posY==7) {
                                columnas=1;
                        }else if(posY==9) {
                                columnas=2;
                        }else if(posY==11) {
                                columnas=3;
                        }
                        MJ[columnas][fuertes[i].x1p-regresarParaMJ] = 0;
                        MJ[columnas][fuertes[i].x2p-regresarParaMJ] = 0;
                        MJ[columnas][fuertes[i].x3p-regresarParaMJ] = 0;
                        MJ[columnas][fuertes[i].x4p-regresarParaMJ] = 0;
                        MJ[columnas][fuertes[i].x5p-regresarParaMJ] = 0;
                        MJ[columnas][fuertes[i].x6p-regresarParaMJ] = 0;
                        MJ[columnas][fuertes[i].x7p-regresarParaMJ] = 0;

                        move(fuertes[i].y, fuertes[i].x1p);
                        addch(' ');
                        move(fuertes[i].y, fuertes[i].x2p);
                        addch(' ');
                        move(fuertes[i].y, fuertes[i].x3p);
                        addch(' ');
                        move(fuertes[i].y, fuertes[i].x4p);
                        addch(' ');
                        move(fuertes[i].y, fuertes[i].x5p);
                        addch(' ');
                        move(fuertes[i].y, fuertes[i].x6p);
                        addch(' ');
                        move(fuertes[i].y, fuertes[i].x7p);
                        addch(' ');

                        fuertes[i].ch1= '(';
                        fuertes[i].ch2= '/';
                        fuertes[i].ch3= '-';
                        if(numAllienFuerte == 0) {
                                fuertes[i].ch4= '1';
                        }else if (numAllienFuerte == 1) {
                                fuertes[i].ch4= '2';
                        }else if (numAllienFuerte == 2) {
                                fuertes[i].ch4= '3';
                        }else if (numAllienFuerte == 3) {
                                fuertes[i].ch4= '4';
                        }
                        fuertes[i].ch5= '-';
                        fuertes[i].ch6= '₩₩';
                        fuertes[i].ch7= ')';
                        fuertes[i].idAlien = numAllienFuerte + 17; //+17
                        fuertes[i].y = posY;
                        fuertes[i].x1= posX+xx; ++xx;
                        fuertes[i].x1= posX + xx ;++xx;
                        fuertes[i].x2= posX + xx ;++xx;
                        fuertes[i].x3= posX + xx ;++xx;
                        fuertes[i].x4= posX + xx ;++xx;
```

```
                    fuertes[i].x5= posX + xx ;++xx;
                    fuertes[i].x6= posX + xx ;++xx;
                    fuertes[i].x7= posX + xx ;;

                    xx=xx+3;
                    /* ESCRIBIR */
                    attron(COLOR_PAIR(5));
                    move(fuertes[i].y, fuertes[i].x1);
                    addch(fuertes[i].ch1);
                    move(fuertes[i].y, fuertes[i].x2);
                    addch(fuertes[i].ch2);
                    move(fuertes[i].y, fuertes[i].x3);
                    addch(fuertes[i].ch3);
                    move(fuertes[i].y, fuertes[i].x4);
                    addch(fuertes[i].ch4);
                    move(fuertes[i].y, fuertes[i].x5);
                    addch(fuertes[i].ch5);
                    move(fuertes[i].y, fuertes[i].x6);
                    addch(fuertes[i].ch6);
                    move(fuertes[i].y, fuertes[i].x7);
                    addch(fuertes[i].ch7);
                    fuertes[i].x1p= fuertes[i].x1;
                    fuertes[i].x2p= fuertes[i].x2;
                    fuertes[i].x3p= fuertes[i].x3;
                    fuertes[i].x4p= fuertes[i].x4;
                    fuertes[i].x5p= fuertes[i].x5;
                    fuertes[i].x6p= fuertes[i].x6;
                    fuertes[i].x7p= fuertes[i].x7;
                    //if(numAllienFuerte==0){
                            MJ[columnas][fuertes[i].x1-regresarParaMJ] = fuertes[i].idAlien;
                            MJ[columnas][fuertes[i].x2-regresarParaMJ] = fuertes[i].idAlien;
                            MJ[columnas][fuertes[i].x3-regresarParaMJ] = fuertes[i].idAlien;
                            MJ[columnas][fuertes[i].x4-regresarParaMJ] = fuertes[i].idAlien;
                            MJ[columnas][fuertes[i].x5-regresarParaMJ] = fuertes[i].idAlien;
                            MJ[columnas][fuertes[i].x6-regresarParaMJ] = fuertes[i].idAlien;

                            MJ[columnas][fuertes[i].x7-regresarParaMJ] = fuertes[i].idAlien;
                    //}

                    attroff(COLOR_PAIR(5));
                }else{
                    move(fuertes[i].y, fuertes[i].x1p);
                    addch(' ');
                    move(fuertes[i].y, fuertes[i].x2p);
                    addch(' ');
                    move(fuertes[i].y, fuertes[i].x3p);
                    addch(' ');
                    move(fuertes[i].y, fuertes[i].x4p);
                    addch(' ');
                    move(fuertes[i].y, fuertes[i].x5p);
                    addch(' ');
                    move(fuertes[i].y, fuertes[i].x6p);
                    addch(' ');
                    move(fuertes[i].y, fuertes[i].x7p);
                    addch(' ');
                    fuertes[i].ch1= ' ';
                    fuertes[i].ch2= ' ';
                    fuertes[i].ch3= ' ';
                    fuertes[i].ch4= ' ';
```

```
                fuertes[i].ch5= ' ';
                fuertes[i].ch6= ' ';
                fuertes[i].ch7= ' ';
                fuertes[i].idAlien = numAllienFuerte + 17; //+17
                fuertes[i].y = posY;
                fuertes[i].x1= posX+xx; ++xx;
                fuertes[i].x1= posX + xx ;++xx;
                fuertes[i].x2= posX + xx ;++xx;
                fuertes[i].x3= posX + xx ;++xx;
                fuertes[i].x4= posX + xx ;++xx;
                fuertes[i].x5= posX + xx ;++xx;
                fuertes[i].x6= posX + xx ;++xx;
                fuertes[i].x7= posX + xx ;;

                xx=xx+3;
                /* ESCRIBIR */
                attron(COLOR_PAIR(5));
                move(fuertes[i].y, fuertes[i].x1);
                addch(fuertes[i].ch1);
                move(fuertes[i].y, fuertes[i].x2);
                addch(fuertes[i].ch2);
                move(fuertes[i].y, fuertes[i].x3);
                addch(fuertes[i].ch3);
                move(fuertes[i].y, fuertes[i].x4);
                addch(fuertes[i].ch4);
                move(fuertes[i].y, fuertes[i].x5);
                addch(fuertes[i].ch5);
                move(fuertes[i].y, fuertes[i].x6);
                addch(fuertes[i].ch6);
                move(fuertes[i].y, fuertes[i].x7);
                addch(fuertes[i].ch7);
                fuertes[i].x1p= fuertes[i].x1;
                fuertes[i].x2p= fuertes[i].x2;
                fuertes[i].x3p= fuertes[i].x3;
                fuertes[i].x4p= fuertes[i].x4;
                fuertes[i].x5p= fuertes[i].x5;
                fuertes[i].x6p= fuertes[i].x6;
                fuertes[i].x7p= fuertes[i].x7;
            }
            numAllienFuerte++;
            return xx;
}

/************************ compartidos ******/

  // int minuts= 0;
  // int seconds= 0;
    void actualizarTimer(WINDOW *win) {
        attron(COLOR_PAIR(4));
            if(punteroMemoria->segundos<59) {
                ++punteroMemoria->segundos;
            }else{
                punteroMemoria->segundos = 0;
                ++punteroMemoria->minutos;
            }
        mvwprintw(win, 1, 9, "%d", punteroMemoria->minutos);
        mvwprintw(win, 1, 10, "%s", ":");
        mvwprintw(win, 1, 11, "%d", punteroMemoria->segundos);
            attroff(COLOR_PAIR(4));
```

```
                refresh();
        }

        void actualizarPuntajeDefensor(WINDOW *win, int tipo){
                if(tipo==0){
                        punteroMemoria->puntajeDefensor = punteroMemoria->puntajeDefensor + 10;
                        attron(COLOR_PAIR(4));
                        mvwprintw(win, 24-2, 8, "%d", punteroMemoria->puntajeDefensor);
                        attroff(COLOR_PAIR(4));
                }else{
                        punteroMemoria->puntajeDefensor = punteroMemoria->puntajeDefensor + 15;
                        attron(COLOR_PAIR(4));
                        mvwprintw(win, 24-2, 8, "%d", punteroMemoria->puntajeDefensor);
                        attroff(COLOR_PAIR(4));
                }
        }

        void actualizarVidas(WINDOW *win){
                attron(COLOR_PAIR(4));
                mvwprintw(win, 1, 75, "%i", punteroMemoria->comandante_invasor.vidas);
                mvwprintw(win, 24-2, 76, "%i" , punteroMemoria->comandante_defensor.vidas);
                attroff(COLOR_PAIR(4));
        }

void actualizarCompartidos(WINDOW *win){
        attron(COLOR_PAIR(4));
        mvwprintw(win, 1, 9, "%d", punteroMemoria->minutos);
        mvwprintw(win, 1, 10, "%s", ":");
  mvwprintw(win, 1, 11, "%d", punteroMemoria->segundos);
        mvwprintw(win, 24-2, 8, "%d", punteroMemoria->puntajeDefensor);
        mvwprintw(win, 24-2, 76, "%i" ,punteroMemoria->comandante_defensor.vidas);
        mvwprintw(win, 1, 75, "%i", punteroMemoria->comandante_invasor.vidas);
        if( punteroMemoria->puntajeDefensor)=100){
                punteroMemoria->termino100=1;
        }else if(punteroMemoria->comandante_invasor.vidas==0){
                punteroMemoria->termino100=1;
        }
        attroff(COLOR_PAIR(4));
//      redibujarDefensorSiempre(win);
}

/************************/
```