main.cpp

```
#include <ctime>
#include <cstdlib>
#include <string>
#include <iostream>
#include <allegro.h>
#include "Paletas.h"
#include "Pelotas.h"
#include "Jugadores.h"
#include "juego.h"
#include "Fucionalidades.h"
#include "Dibujar.h"

#define ANCHO 1000
#define ALTO 500

int main()
{
   //INICIALIZACIONES
   inicializaciones( );

   //LOOP DEL JUEGO
   loop_del_juego( );

   //DESTRUCTORES
   destructores( );

   //SALIDA DEL JUEGO
   allegro_exit( );

return 0;
}
END_OF_MAIN( )
```

Juego.h

```
#ifndef JUEGO_H_INCLUDED
#define JUEGO_H_INCLUDED
#include <allegro.h>
#include "Fucionalidades.h"
#include "Paletas.h"
#include "Pelotas.h"
#include "Jugadores.h"
#include "Dibujar.h"

#define ANCHO 1000
#define ALTO 500

//DECLARACION DE ESTRUCTURAS
typedef struct
{
   BITMAP *mapa_de_bits;
}
bitmaps;
typedef struct
{
   SAMPLE *muestra_de_audio;
}
samples;

//FUNCIONES JUEGO
void inicializaciones( );
void loop_del_juego( );
void incremento( );
void cerrar_portada( );
void cerrar_ventana( );
void first_screen( );
void destructores( );
void setter_de_clases( );

#endif // JUEGO_H_INCLUDED
```

Juego.cpp

```
#include "juego.h"

//INVOCACION DE CLASES
Paleta<int> paletaV1;
Paleta<int> paletaV2;
Pelota<int,float> pelotaV;
Jugador<int> jugadoresV;
Funcionalidad<int,bool> funcionV;
//INVOCACION DE ESTRUCTURAS
bitmaps *bits;
samples *audio;


void setter_de_clases ()
{
   //SETTEO E INICIALIZACION DE CLASES Y FUNCIONES
   resetPelota( pelotaV , funcionV );
   pelotaV.setVelX( 5 );
   pelotaV.setVelY( 5 );
   pelotaV.setVel( 5 );
   pelotaV.setAlto( 15 );
   pelotaV.setAncho( 15 );
   paletaV1.setX( 0 );
   jugadoresV.setDisparos1( 5 );
   jugadoresV.setDisparos2( 5 );
   funcionV.setContador( 0 );
   funcionV.setPropulsion( 0 );
   paletaV1.setY( ALTO / 2 - 50 );
   paletaV1.setTamano( 15 , 94 );
   paletaV2.setX( ANCHO - paletaV1.getAncho() );
   paletaV2.setY( ALTO / 2 - 50 );
   paletaV2.setTamano( 15 , 94 );
}
void destructores ()
{
   //DESTRUCTORES DE BITMAPS
   destroy_bitmap( bits[0].mapa_de_bits );
   destroy_bitmap( bits[1].mapa_de_bits );
   destroy_bitmap( bits[2].mapa_de_bits );
   destroy_bitmap( bits[3].mapa_de_bits );
   destroy_bitmap( bits[4].mapa_de_bits );
   destroy_bitmap( bits[5].mapa_de_bits );
   destroy_bitmap( bits[6].mapa_de_bits );
   destroy_bitmap( bits[7].mapa_de_bits );
   destroy_bitmap( bits[8].mapa_de_bits );
   destroy_bitmap( bits[9].mapa_de_bits );
   destroy_bitmap( bits[10].mapa_de_bits );
   destroy_bitmap( bits[11].mapa_de_bits );
   destroy_bitmap( bits[12].mapa_de_bits );
   destroy_bitmap( bits[13].mapa_de_bits );
   destroy_bitmap( bits[14].mapa_de_bits );
   destroy_bitmap( bits[15].mapa_de_bits );

   free(bits);
   free(audio);
}
void first_screen()
{
   //FUNCION PORTADA
   play_sample( audio[0].muestra_de_audio , 200 , 150 ,
1000 , 0 );
   while( ! funcionV.getPortada() )
   {
      if( key[KEY_ENTER] )
         funcionV.setPortada( true );

      blit( bits[8].mapa_de_bits , screen , 0 , 0 , 0 , 0 , ANCHO
, ALTO );
   }
}
void inicializaciones( )
{
   //INICIALIZACIONES Y SETTERS DEL JUEGO
   allegro_init( );
   set_window_title( "Tennis de Mesa" );
   install_keyboard( );
   install_timer( );
   install_sound( DIGI_AUTODETECT , MIDI_AUTODETECT ,
NULL );
   set_volume( 230 , 200 );
   set_color_depth( 24 );
   set_gfx_mode( GFX_AUTODETECT_WINDOWED , ANCHO
, ALTO , 0 , 0 );
   install_int_ex( incremento , BPS_TO_TIMER( 70 ) );
   set_close_button_callback( cerrar_ventana );

   //DECLARACION DE SAMPLES DE AUDIO
   SAMPLE *audio1 = load_wav(
"C:/Users/marco/OneDrive/Documentos/Programas C-
C++/Ping-Pong/audio/Intro.wav" );
   SAMPLE *audio2 = load_wav(
"C:/Users/marco/OneDrive/Documentos/Programas C-
C++/Ping-Pong/audio/Hit.wav" );
   SAMPLE *audio3 = load_wav(
"C:/Users/marco/OneDrive/Documentos/Programas C-
C++/Ping-Pong/audio/Punto.wav" );
   SAMPLE *audio4 = load_wav(
"C:/Users/marco/OneDrive/Documentos/Programas C-
C++/Ping-Pong/audio/Saque.wav" );
   SAMPLE *audio5 = load_wav(
"C:/Users/marco/OneDrive/Documentos/Programas C-
C++/Ping-Pong/audio/Level-Up.wav" );
   SAMPLE *audio6 = load_wav(
"C:/Users/marco/OneDrive/Documentos/Programas C-
C++/Ping-Pong/audio/Winner.wav" );
   SAMPLE *audio7 = load_wav(
"C:/Users/marco/OneDrive/Documentos/Programas C-
C++/Ping-Pong/audio/WallHit.wav" );
   //DECLARACION DE BITMAPS
   BITMAP *buffer = create_bitmap( ANCHO , ALTO );
   BITMAP *buffer1 = create_bitmap( ANCHO, ALTO );
```

```
   //BITMAP *buffer1 =
load_bitmap("C:/Users/marco/OneDrive/Documentos/Prog
ramas C-C++/Ping-Pong/imagenes/nivel1.bmp",NULL);
   BITMAP *buffer2 = create_bitmap( ANCHO , ALTO );
   //BITMAP *buffer2 =
load_bitmap("C:/Users/marco/OneDrive/Documentos/Prog
ramas C-C++/Ping-Pong/imagenes/nivel2.bmp",NULL);
   BITMAP *buffer3 = create_bitmap( ANCHO , ALTO );
   //BITMAP *buffer3 =
load_bitmap("C:/Users/marco/OneDrive/Documentos/Prog
ramas C-C++/Ping-Pong/imagenes/nivel3.bmp",NULL);
   BITMAP *buffer4 = create_bitmap( ANCHO , ALTO );
   //BITMAP *buffer4 =
load_bitmap("C:/Users/marco/OneDrive/Documentos/Prog
ramas C-C++/Ping-Pong/imagenes/nivel4.bmp",NULL);
   BITMAP *buffer5 = create_bitmap( ANCHO , ALTO );
   //BITMAP *buffer5 =
load_bitmap("C:/Users/marco/OneDrive/Documentos/Prog
ramas C-C++/Ping-Pong/imagenes/nivel5.bmp",NULL);
   BITMAP *paleta_izq = load_bitmap(
"C:/Users/marco/OneDrive/Documentos/Programas C-
C++/Ping-Pong/imagenes/paleta1.bmp" , NULL );
   BITMAP *paleta_der = load_bitmap(
"C:/Users/marco/OneDrive/Documentos/Programas C-
C++/Ping-Pong/imagenes/paleta2.bmp" , NULL );
   BITMAP *inicio = load_bitmap(
"C:/Users/marco/OneDrive/Documentos/Programas C-
C++/Ping-Pong/imagenes/portada.bmp" , NULL );
   BITMAP *jugador1w = load_bitmap(
"C:/Users/marco/OneDrive/Documentos/Programas C-
C++/Ping-Pong/imagenes/player1_wins.bmp" , NULL );
   BITMAP *jugador2w = load_bitmap(
"C:/Users/marco/OneDrive/Documentos/Programas C-
C++/Ping-Pong/imagenes/player2_wins.bmp" , NULL );

   //ALMACENADO DE BITMAPS EN MEMORIA DINAMICA
   bits = ( bitmaps * ) malloc( sizeof ( bitmaps ) * 11 );
   bits[0].mapa_de_bits = buffer;
   bits[1].mapa_de_bits = buffer1;
   bits[2].mapa_de_bits = buffer2;
   bits[3].mapa_de_bits = buffer3;
   bits[4].mapa_de_bits = buffer4;
   bits[5].mapa_de_bits = buffer5;
   bits[6].mapa_de_bits = paleta_izq;
   bits[7].mapa_de_bits = paleta_der;
   bits[8].mapa_de_bits = inicio;
   bits[9].mapa_de_bits = jugador1w;
   bits[10].mapa_de_bits = jugador2w;

   //ALMACENADO DE SAMPLES DE AUDIO EN MEMORIA
DINAMICA
   audio = ( samples * ) malloc( sizeof ( samples ) * 7 );
   audio[0].muestra_de_audio = audio1;
   audio[1].muestra_de_audio = audio2;
   audio[2].muestra_de_audio = audio3;
   audio[3].muestra_de_audio = audio4;
   audio[4].muestra_de_audio = audio5;
   audio[5].muestra_de_audio = audio6;
   audio[6].muestra_de_audio = audio7;

   //VIÑETA1
   setter_de_clases( );

   //VIÑETA2
   first_screen( );
}
void loop_del_juego ( )
{
   //CICLO DEL JUEGO
      //FUNCION VENTANA (ESC = SALIR)
   while( ! funcionV.getVentana() )
   {
      if( key[KEY_ESC] )
         funcionV.setVentana( true );

      //COLOR DE BITMAPS FUENTE
      clear_to_color( bits[1].mapa_de_bits , 0x003800 );
      clear_to_color( bits[2].mapa_de_bits , 0x370000 );
      clear_to_color( bits[3].mapa_de_bits , 0x060739 );
      clear_to_color( bits[4].mapa_de_bits , 0x363300 );
      clear_to_color( bits[5].mapa_de_bits , 0x000000 );

      //SET DE TIEMPO E INICIALIZACION DEL JUEGO
      while( funcionV.getTiempo() > 0 )
      {
         if( funcionV.getResultado() == 1 )
         {
            if( key[KEY_SPACE] )
            {
               play_sample( audio[3].muestra_de_audio , 200 ,
150 , 1000 , 0 );
               funcionV.setResultado( 0 );
               funcionV.setMensaje( 0 );
            }
            funcionV.setTiempo( 0 );
            continue;
         }

         //DECLARACION DE VARIABLES, MARCA DE
COORDENADAS Y POSICIONES
         int posy1 = ALTO - paletaV1.getAlto();
         int posy2 = ALTO - paletaV2.getAlto();
         int y1 = paletaV1.getY();
         int y2 = paletaV2.getY();
         int movSpeed = 3;

         //MOVIMIENTOS
         if ( key[KEY_W] )
         {
            if( paletaV1.getY() >= 0 && y1 <= posy1 )
            {
               paletaV1.setY( paletaV1.getY() - movSpeed );
            }

            else
            {
```

```
            paletaV1.setY( ( paletaV1.getY() < 0 ) ?0 : posy1
);
          }
        }
        else if( key[KEY_S] )
        {
          if( paletaV1.getY() >= 0 && y1 <= posy1 )
          {
            paletaV1.setY( paletaV1.getY() + movSpeed );
          }
          else
          {
            paletaV1.setY( ( paletaV1.getY() < 0 ) ?0 : posy1
);
          }
        }
        if ( key[KEY_UP] )
        {
          if( paletaV2.getY() >= 0 && y2 <= posy2 )
          {
            paletaV2.setY( paletaV2.getY() - movSpeed );
          }
          else
          {
            paletaV2.setY( ( paletaV2.getY() < 0 ) ? 0 : posy2
);
          }
        }
        else if( key[KEY_DOWN] )
        {
          if( paletaV2.getY() >= 0 && y2 <= posy2 )
          {
            paletaV2.setY( paletaV2.getY() + movSpeed );
          }
          else
          {
            paletaV2.setY( ( paletaV2.getY() < 0 ) ?0 : posy2
);
          }
        }

        //FUNCIONES VARIAS
        moverPelota( pelotaV , paletaV1 , paletaV2 ,
jugadoresV , funcionV , bits[0].mapa_de_bits ,
bits[9].mapa_de_bits , bits[10].mapa_de_bits ,
audio[4].muestra_de_audio , audio[5].muestra_de_audio ,
audio[6].muestra_de_audio , audio[1].muestra_de_audio ,
audio[2].muestra_de_audio );

        //TIEMPO - 1
        funcionV.setTiempo( funcionV.getTiempo() - 1 );

      }

      //FUNCIONES Y SETTERS DE PALETAS, PELOTAS, MESA
      dibujar_mesa( bits[0].mapa_de_bits );
      dibujar_pelota( bits[0].mapa_de_bits , pelotaV );

      //TABLERO
      tablero( bits[0].mapa_de_bits , jugadoresV , funcionV );

      //PALETAS (DOBLE OPCION)
      masked_blit( bits[6].mapa_de_bits ,
bits[0].mapa_de_bits , 0 , 0 , paletaV1.getX() ,
paletaV1.getY() , 15, 94 );
      //draw_sprite( bits[0].mapa_de_bits
,bits[6].mapa_de_bits , paleta_1.getX(), paleta_1.getY()) ;
      masked_blit( bits[7].mapa_de_bits ,
bits[0].mapa_de_bits , 0 , 0 , paletaV2.getX() ,
paletaV2.getY() , 15 , 94 );
      //draw_sprite( bits[0].mapa_de_bits
,bits[7].mapa_de_bits , paleta_2.getX(), paleta_2.getY()) ;

      //INVOCACION DE BITMAPS - CAMBIO DE NIVEL
      if( funcionV.getContador() < 4 )
      {
        funcionV.setNivel( 1 );
        bits[0].mapa_de_bits = bits[1].mapa_de_bits;
        blit( bits[0].mapa_de_bits , screen , 0 , 0 , 0 , 0 ,
ANCHO , ALTO );
      }
      else if( funcionV.getContador() < 8 )
      {
        if( funcionV.getMensaje() == 1 &&
funcionV.getContador() == 4 )
        {
          textprintf_ex( bits[0].mapa_de_bits , font , 340 ,
250 , 0xD1D1D1 , -1 , "CAMBIO DE NIVEL!! VELOCIODAD
AUMENTADA!!" );
        }
        funcionV.setNivel( 2 );
        bits[0].mapa_de_bits = bits[2].mapa_de_bits;
        blit( bits[0].mapa_de_bits , screen , 0 , 0 , 0 , 0 ,
ANCHO , ALTO );
      }
      else if( funcionV.getContador() < 12 )
      {
        if( funcionV.getMensaje() == 1 &&
funcionV.getContador()==8 )
        {
          textprintf_ex( bits[0].mapa_de_bits , font , 340 ,
250 , 0xD1D1D1 , -1 , "CAMBIO DE NIVEL!! VELOCIODAD
AUMENTADA!!" );
          textprintf_ex( bits[0].mapa_de_bits , font , 350 ,
270 , 0xD1D1D1 , -1 , "RESET DE PUNTOS - 5 PARA CADA
JUGADOR" );
        }

        if ( funcionV.getContador() == 8)
        {
          jugadoresV.setDisparos1(5);
          jugadoresV.setDisparos2(5);
        }
        funcionV.setNivel( 3 );
        bits[0].mapa_de_bits = bits[3].mapa_de_bits;
```

```
        blit(bits[0].mapa_de_bits , screen , 0 , 0 , 0 , 0 ,
ANCHO , ALTO );
      }
      else if( funcionV.getContador() < 16 )
      {
         if( funcionV.getMensaje() == 1 &&
funcionV.getContador() == 12 )
         {
            textprintf_ex( bits[0].mapa_de_bits , font , 340 ,
250 , 0xD1D1D1 , -1 , "CAMBIO DE NIVEL!! VELOCIODAD
AUMENTADA!!" );
         }
         funcionV.setNivel( 4 );
         bits[0].mapa_de_bits = bits[4].mapa_de_bits;
         blit( bits[0].mapa_de_bits , screen , 0 , 0 , 0 , 0 ,
ANCHO , ALTO );
      }
      else if( funcionV.getContador() < 21)
      {
         if( funcionV.getMensaje() == 1  &&
funcionV.getContador() == 16 )
         {
            textprintf_ex( bits[0].mapa_de_bits , font , 340 ,
250 , 0xD1D1D1 , -1 , "CAMBIO DE NIVEL!! VELOCIODAD
AUMENTADA!!" );
         }
         funcionV.setNivel( 5 );
         bits[0].mapa_de_bits = bits[5].mapa_de_bits;
         blit( bits[0].mapa_de_bits , screen , 0 , 0 , 0 , 0 ,
ANCHO , ALTO );
      }

      //TIEMPO INTERLOCUCION DE CPU (20)
      rest( 20 );

      //LIMPIEZA DE BITMAPS
      clear_bitmap( bits[0].mapa_de_bits );
   }
}
void cerrar_portada()
{
   funcionV.setPortada( true );
}
void cerrar_ventana()
{
   funcionV.setVentana( true );
}
void incremento()
{
   funcionV.setTiempo( funcionV.getTiempo() + 1 );
}
```

Jugadores.h

```
#ifndef JUGADORES_H_INCLUDED
#define JUGADORES_H_INCLUDED
#include <iostream>

//TEMPLATE CLASE, JUGADOR
template <class type1>//type1 = int
class Jugador
{
   type1 puntaje1;
   type1 puntaje2;
   type1 disparos1;
   type1 disparos2;

   public:

   Jugador() {
      puntaje1 = 0;
      puntaje2 = 0;
      disparos1 = 0;
      disparos2 = 0;
   }

   ~Jugador () {

   }

   void setPuntaje1 ( type1 Puntaje1 ) {
      this->puntaje1 = Puntaje1;
   };

   void setPuntaje2 ( type1 Puntaje2 ) {
      this->puntaje2 = Puntaje2;
   };

   void setDisparos1( type1 disparos1 ) {
      this->disparos1 = disparos1;
   }

   void setDisparos2( type1 disparos2 ) {
      this->disparos2 = disparos2;
   }

   type1 getDisparos2 () {
      return disparos2;
   }
   type1 getDisparos1 () {
      return disparos1;
   }

   type1 getPuntaje1 () {
      return puntaje1;
   }

   type1 getPuntaje2 () {
      return puntaje2;
   }
};

#endif // JUGADORES_H_INCLUDED
```

Paletas.h

```
#ifndef PALETAS_H_INCLUDED
#define PALETAS_H_INCLUDED
#include <allegro.h>
#include <iostream>
#include "Jugadores.h"

//TEMPLATE CLASE PALETA. HERENCIA DE JUGADOR
template <class type1>//type1 = int
class Paleta: public Jugador<int>
{
   type1 X;
   type1 Y;
   type1 alto;
   type1 ancho;

   public:
   Paleta () {
      X = 0;
      Y = 0;
      alto = 10;
      ancho = 2;
   }

   ~Paleta () {

   }

   void setTamano( type1 ancho , type1 alto ) {
      this->alto = alto;
      this->ancho = ancho;
   }

   void setX( type1 x ) {
      this->X = x;
   }

   void setY( type1 y ) {
      this->Y = y;
   }

   type1 getX() {
      return X;
   }

   type1 getY() {
      return Y;
   }

   type1 getAlto() {
      return alto;
   }

   type1 getAncho() {
      return ancho;
   }
};

#endif // PALETAS_H_INCLUDED
```

Pelotas.h

```
#ifndef PELOTAS_H_INCLUDED
#define PELOTAS_H_INCLUDED
#include <allegro.h>
#include <iostream>
#include "Fucionalidades.h"
#include "Paletas.h"


#define ANCHO 1000
#define ALTO 500

//TEMPLATE CLASE PELOTA. HEREDADA DE PALETA
template <class type1,class type2>//type1 = int - type2 =
float
class Pelota: public Paleta<int>
{
   type1 X;
   type1 Y;
   type1 alto;
   type1 ancho;
   type1 dirX;
   type1 dirY;
   type2 velocidadX;
   type2 velocidadY;
   type1 velocidad;

   public:

   Pelota() {
      X=0;
      Y=0;
      dirX=0;
      dirY=0;
   }

   ~Pelota () {

   }

   void setVel( type1 velocidad ) {
      this->velocidad = velocidad;
   }

   void setVelX ( type2 velocidadX ) {
      this->velocidadX = velocidadX;
   }

   void setVelY ( type2 velocidadY ) {
      this->velocidadY = velocidadY;
   }

   void setX( type1 x) {
      this->X = x;
   }

   void setY ( type1 y) {
      this->Y = y;
   }

   void setAlto ( type1 alto) {
      this->alto = alto;
   }

   void setAncho ( type1 ancho ) {
      this->ancho = ancho;
   }

   void setDirX ( type1 dirX) {
      this->dirX = dirX;
   }

   void setDirY ( type1 dirY ) {
      this->dirY = dirY;
   }

   type1 getVel () {
      return velocidad;
   }

   type2 getVelX () {
      return velocidadX;
   }

   type2 getVelY () {
      return velocidadY;
   }

   type1 getX () {
      return X;
   }

   type1 getY () {
      return Y;
   }

   type1 getAncho () {
      return ancho;
   }

   type1 getAlto () {
      return alto;
   }

   type1 getDirX () {
      return dirX;
   }

   type1 getDirY () {
      return dirY;
   }
};

//SOBRECARGA DE FUNCIONES
```

```
void resetPelota( Pelota<int,float> &pelota ,
Funcionalidad<int,bool> &funciones );
void resetPelota( Pelota<int,float> &pelota , Paleta<int>
paleta , int a, Jugador<int> &jugadores ,
Funcionalidad<int,bool> &funciones, BITMAP *buffer ,
BITMAP *jugador1 , BITMAP *jugador2 , SAMPLE *level_up
, SAMPLE *winner );
//MOVIMIENTO DE PELOTA
void moverPelota( Pelota<int,float> &pelota , Paleta<int>
&paleta1 , Paleta<int> &paleta2 , Jugador<int> &jugadores
, Funcionalidad<int,bool> &funciones , BITMAP *buffer ,
BITMAP *jugador1 , BITMAP *jugador2 , SAMPLE *level_up
, SAMPLE *winner , SAMPLE *wall ,  SAMPLE *hit , SAMPLE
*punto );
void golpe_especialp1( Pelota<int,float> &pelota ,
Paleta<int> paleta , BITMAP *buffer , int propulsion ,
Funcionalidad<int,bool> &funciones , int diry , Jugador<int>
&jugadores );
void golpe_especialp2( Pelota<int,float> &pelota ,
Paleta<int> paleta , BITMAP *buffer , int propulsion ,
Funcionalidad<int,bool> &funciones , int diry , Jugador<int>
&jugadores );

#endif // PELOTAS_H_INCLUDED
```

Pelotas.cpp

```
#include "Pelotas.h"

void resetPelota( Pelota<int,float> &pelota ,
Funcionalidad<int,bool> &funciones )
{
     //SETTERS DE PELOTA
     pelota.setX( ANCHO / 2 );
          pelota.setY( ALTO / 2 );
          srand( time( 0 ) );
          int direccionX = ( rand() % 2) + 1; //DIRECCION X
RAND 0/1
          int direccionY = ( rand() % 2) + 1; //DIRECCION Y
RAND 0/1
          direccionX = ( direccionX == 1 ) ? - 1 : 1;//1 = -1 , 2
= 1
          direccionY = ( direccionY == 1 ) ? - 1 : 1;//1 = -1 , 2 =
1
          pelota.setDirX( direccionX );//SETTER
          pelota.setDirY( direccionY );//SETTER
          funciones.setResultado( 1 );//MARCA DE INICIO
}

void resetPelota( Pelota<int,float> &pelota , Paleta<int>
paleta , int a , Jugador<int> &jugadores ,
Funcionalidad<int,bool> &funciones , BITMAP *buffer ,
BITMAP *jugador1 , BITMAP *jugador2 , SAMPLE *level_up
, SAMPLE *winner)
{
  if ( a == 1 )
  {
  //GANA PALETA2
  //PELOTA SETTEADA EN PALETA1
     pelota.setX( paleta.getX() + paleta.getAncho() +
pelota.getAncho() - 3 );
     pelota.setY( paleta.getY() + paleta.getAlto() / 2 );
     pelota.setDirX( 1 );
     jugadores.setPuntaje2( jugadores.getPuntaje2() + 1 );
  }
  else
  {
  //GANA PALETA1
  //PELOTA SETTEADA EN PALETA2
     pelota.setX( paleta.getX() - paleta.getAncho() / 2 -
pelota.getAncho() + 7 );
     pelota.setY( paleta.getY() + paleta.getAlto() / 2 );
     pelota.setDirX( -1 );
     jugadores.setPuntaje1( jugadores.getPuntaje1() + 1 );
  }

  if( funciones.getContador() == 3 )
  {
     //CAMBIO DE VELOCIDAD1 - NIVEL2 - MENSAJE DE
CAMBIO
     play_sample( level_up , 200 , 150 , 1000 , 0 );
     pelota.setVel( 7 );
  }
  else if( funciones.getContador() == 7 )
  {
     //CAMBIO DE VELOCIDAD2 - NIVEL3 - MENSAJE DE
CAMBIO
     play_sample( level_up , 200 , 150 , 1000 , 0 );
     pelota.setVel( 9 );
  }
  else if( funciones.getContador() == 11 )
  {
     //CAMBIO DE VELOCIDAD3 - NIVEL4 - MENSAJE DE
CAMBIO
     play_sample( level_up , 200 , 150 , 1000 , 0 );
     pelota.setVel( 11 );
  }
  else if( funciones.getContador() == 15 )
  {
     //CAMBIO DE VELOCIDAD4 - NIVEL5 - MENSAJE DE
CAMBIO
     play_sample( level_up , 200 , 150 , 1000 , 0 );
     pelota.setVel( 13 );
  }

  if( jugadores.getPuntaje1() == 10 )
  //GANADOR PALETA1/PLAYER1
  {
     play_sample( winner , 200 , 150 , 1000 , 0 );
     while( ! key[KEY_ENTER] )
     {
        //CASE ENTER: SETTER E INICIALIZACION DE JUEGO
DESDE 0
        blit( jugador1 , screen , 0 , 0 , 0 , 0 , ANCHO , ALTO );
        jugadores.setPuntaje1( 0 );
        jugadores.setPuntaje2( 0 );
        funciones.setContador( 0 );
        pelota.setVelX( 5 );
        pelota.setVelY( 5 );
        pelota.setVel( 5 );
        jugadores.setDisparos1( 5 );
        jugadores.setDisparos2( 5 );
        funciones.setNuevo( true );
        resetPelota( pelota , funciones );
     }
     if( key[KEY_ESC] )
     {
        //CASE ESC: EXIT GAME
        allegro_exit();
     }
  }
  else if( jugadores.getPuntaje2() == 10 )
  {
  //GANADOR PALETA2/PLAYER2
     play_sample( winner , 200 , 150 , 1000 , 0 );
     while( ! key[KEY_ENTER] )
     {
        //CASE ENTER: SETTERS E INICIALIZACION DE JUEGO
DESDE 0
        blit( jugador2 , screen , 0 , 0 , 0 , 0 , ANCHO , ALTO );
        jugadores.setPuntaje1( 0 );
```

```
        jugadores.setPuntaje2( 0 );
        funciones.setContador( 0 );
        pelota.setVelX( 5 );
        pelota.setVelY( 5 );
        pelota.setVel( 5 );
        jugadores.setDisparos1( 5 );
        jugadores.setDisparos2( 5 );
        funciones.setNuevo( true );
        resetPelota( pelota , funciones );
      }
      if( key[KEY_ESC] )
      {
        //CASE ESC: EXIT GAME
        allegro_exit();
      }
    }
    if( funciones.getNuevo() == true )
    {
      //CASE ENTER: NUEVO = FALSE
      funciones.setNuevo( false );
    }

    //SETTER DE VELOCIDAD LUEGO DEL RESTART,
CONTADOR DE TURNOS, MARCA DE
RESULTADO/INICIALIZACION
    pelota.setVelX( pelota.getVel() );
    pelota.setVelY( pelota.getVel() );
    funciones.setContador( funciones.getContador() + 1 );
    funciones.setResultado( 1 );
}
void moverPelota( Pelota<int,float> &pelota , Paleta<int>
&paleta1 , Paleta<int> &paleta2 , Jugador<int> &jugadores
, Funcionalidad<int,bool> &funciones , BITMAP *buffer ,
BITMAP *jugador1 , BITMAP *jugador2 , SAMPLE *level_up
, SAMPLE *winner , SAMPLE *wall ,  SAMPLE *hit , SAMPLE
*punto)
{
    //DECLARACION DE COORDENADAS Y DIRECCIONES
            int dirx = pelota.getDirX( );
            int diry = pelota.getDirY( );
            int px = pelota.getX( );
            int py = pelota.getY( );
            int y1 = paleta1.getY( );
            int y2 = paleta2.getY( );

            if ( px <= paleta1.getAncho() + pelota.getAncho() /
2 )
    //GOLPE BORDE INTERIOR DE PALETA1
            {
              if ( ( py + pelota.getAncho() / 2 ) >= y1 && ( py -
pelota.getAncho() / 2 ) <= y1 + paleta1.getAlto() )
      //GOLPE LIMITES LARGO DE PALETA
      {
        if( py + pelota.getAncho() / 2 >= y1 && py <= y1 +
paleta1.getAlto() - 63 )
        //GOLPE ESQUINA SUPERIOR DE PALETA1. EJE Y>X
        {
          if( diry == 1 )
            {
              diry = -1;
            }
            play_sample( hit , 200 , 150 , 1000 , 0 );
            pelota.setVelX( pelota.getVel() - 1 );
            pelota.setVelY( pelota.getVel() + 1 );
        }
        else if( py >= y1 + 62 && ( py - pelota.getAncho() / 2 )
<= y1 + paleta1.getAlto() )
        //GOLPE ESQUINA INFERIOR DE PALETA1. EJE Y>X
        {
          if( diry == -1 )
          {
            diry = 1;
          }
          play_sample( hit , 200 , 150 , 1000 , 0 );
          pelota.setVelX( pelota.getVel() - 1 );
          pelota.setVelY( pelota.getVel() + 1 );
        }
        else if( py >= y1 + 31 && py <= y1 + paleta1.getAlto()
- 32 )
        //GOLPE EN EL MEDIO DE LA PALETA. Y=X
        {
          play_sample( hit , 200 , 150 , 1000 , 0 );
          pelota.setVelX( pelota.getVel() );
          pelota.setVelY( pelota.getVel() );
        }
        dirx *= -1;
      }
      else
      //PALETA1 PIERDE
      {
        if( ( py > y1 + paleta1.getAlto() ) && ( px >
paleta1.getAncho() + pelota.getAncho() / 2 ) && ( diry == -1
) && ( px > 0 ) )
        //PALETA1 REBOTE 0
        {
          play_sample( hit , 200 , 150 , 1000 , 0 );
          diry = -1;
        }
        else if( ( py < y1 ) && ( px > paleta1.getAncho() +
pelota.getAncho() / 2 ) && ( diry == 1 ) && ( px > 0 ) )
        {
          play_sample( hit , 200 , 150 , 1000 , 0 );
          diry = 1;
        }
        if( px <= 0 )
        {
          funciones.setMensaje( 1 );
          play_sample( punto , 200 , 150 , 1000 , 0 );
          resetPelota( pelota , paleta1 , 1 , jugadores ,
funciones , buffer , jugador1 , jugador2 , level_up , winner);
          return;
        }
      }
          }
    else if ( px >= ANCHO - paleta2.getAncho() -
pelota.getAncho() / 2 )
```

```
  //GOLPE BORDE INTERIOR DE PALETA2
          {
            if ( ( py + pelota.getAncho() / 2 ) >= y2 && ( py -
pelota.getAncho() / 2 ) <= y2 + paleta2.getAlto())
        //GOLPE LIMITES LARGO DE PALETA
        {
          if( ( py + pelota.getAncho() / 2 ) >= y2 && py <= y2
+ paleta2.getAlto() - 63 )
          //GOLPE ESQUINA SUPERIOR. VELOCIDAD DE EJE
Y>X
          {
            if( diry == 1 )
            {
              diry = -1;
            }
            play_sample( hit , 200 , 150 , 1000 , 0 );
            pelota.setVelX( pelota.getVel() - 1 );
            pelota.setVelY( pelota.getVel() + 1 );
          }
          else if( py >= y2 + 62 && ( py - pelota.getAncho() /
2 ) <= y2 + paleta2.getAlto() )
          //GOLPE ESQUINA INFERIOR. VELOCIDAD DE EJE
Y>X
          {
            if( diry == -1 )
            {
              diry = 1;
            }
            play_sample( hit , 200 , 150 , 1000 , 0 );
            pelota.setVelX( pelota.getVel() - 1 );
            pelota.setVelY( pelota.getVel() + 1 );
          }
          else if( py >= y2 + 31 && py <= y2 +
paleta1.getAlto() - 32 )
          //GOLPE MEDIO DE LA PALETA
          {
            play_sample( hit , 200 , 150 , 1000 , 0 );
            pelota.setVelX( pelota.getVel() );
            pelota.setVelY( pelota.getVel() );
          }
          dirx *= -1;
        }
        else
        //PIERDE PALETA2
        {
          if(( ( py - pelota.getAncho() / 2 ) > y2 +
paleta2.getAlto() ) && ( px > ANCHO - paleta2.getAncho() -
pelota.getAncho() / 2 ) && ( diry == -1 ) && ( px < ANCHO ) )
          //PALETA2 REBOTE 0
          {
             play_sample( hit , 200 , 150 , 1000 , 0 );
             diry = 1;
          }
          else if( ( ( py + pelota.getAncho() / 2 ) < y2 ) && ( px
> ANCHO - paleta2.getAncho() - pelota.getAncho() / 2 ) &&
( diry == 1 ) && ( px < ANCHO ) )
          {
            play_sample( hit , 200 , 150 , 1000 , 0 );
            diry = -1;
          }
          if( px >= ANCHO )
          {
            //PUNTO
            funciones.setMensaje( 1 );
            play_sample( punto , 200 , 150 , 1000 , 0 );
            resetPelota( pelota , paleta2 , 2 , jugadores ,
funciones , buffer , jugador1 , jugador2 , level_up , winner);
            return;
          }
        }
          }
          else if (( diry < 0 && py <= 0 ) || ( diry > 0 &&  py
>= ( ALTO-pelota.getAlto() ) ))
  //REBOTE PARED
  {
             play_sample( wall , 200 , 150 , 1000 , 0 );
    diry *= -1;
          }

          if( ( pelota.getX() <= paleta1.getAncho() +
pelota.getAncho() / 2 )  && ( pelota.getY() >= paleta1.getY()
) && ( pelota.getY() <= paleta1.getY() + paleta1.getAlto() )
&& ( key[KEY_ALT] ) )
  //PROPULSION PALETA1
  {
    funciones.setPropulsion( 1 );
    jugadores.setDisparos1( jugadores.getDisparos1() - 1 );
  }
  if( ( funciones.getPropulsion() == 1 ) && (
jugadores.getDisparos1 () >= 0 ) )
  {
    golpe_especialp1( pelota , paleta1 , buffer ,
funciones.getPropulsion() , funciones , diry , jugadores );
  }

  //PROPULSION PALETA2
  if( ( pelota.getX() >= ANCHO - paleta2.getAncho() -
pelota.getAncho() / 2 ) && ( pelota.getY() >= paleta2.getY()
) && ( pelota.getY() <= paleta2.getY() + paleta2.getAlto() )
&& ( key[KEY_ALTGR] ) )
  {
    funciones.setPropulsion( 2 );
    jugadores.setDisparos2( jugadores.getDisparos2() - 1 );
  }
  if( funciones.getPropulsion() == 2 &&
jugadores.getDisparos2() >= 0  )
  {
    golpe_especialp2( pelota , paleta2 , buffer ,
funciones.getPropulsion() , funciones , diry , jugadores );
  }

  //MOVIMIENTOS CONSTANTES DE PELOTA
  pelota.setX( pelota.getX() + pelota.getVelX() * dirx );
  pelota.setDirX( dirx );
  pelota.setDirY( diry );
  pelota.setY( pelota.getY() + pelota.getVelY() * diry );
```

```
   pelota.setDirX( dirx );
   pelota.setDirY( diry );
}
void golpe_especialp1( Pelota<int,float> &pelota ,
Paleta<int> paleta , BITMAP *buffer , int propulsion ,
Funcionalidad<int,bool> &funciones , int diry , Jugador<int>
&jugadores )
{
   int i;
   int j = rand() % 80; //EJE X
   int k = rand() % 50; //EJE Y ( Y*2 = 100 )


   if( pelota.getX() < 700 && diry > 0 )
   //DIBUJOS ESTELA DE PELOTA PALETA1. DIRECCION DE Y
POSITIVA
   {
      for( i=0 ; i<1000 ; i++ )
      {
         line( buffer , pelota.getX() - j , pelota.getY() - k ,
pelota.getX() - j , pelota.getY() - k , 0xFFFFFF);
         line( buffer , pelota.getX() - j - 5 , pelota.getY() - ( k +
3 ) , pelota.getX() - j - 5 , pelota.getY() - ( k + 3 ) , 0xFFFFFF );
         line( buffer , pelota.getX() - j - 15 , pelota.getY() - ( k +
7 ) , pelota.getX() - j - 15 , pelota.getY() - ( k + 7 ) , 0xFFFFFF
);
         line( buffer , pelota.getX() - j - 3 , pelota.getY() - ( k +
3 ) , pelota.getX() - j - 3 , pelota.getY() - ( k + 3 ) , 0xFFFFFF );
         line( buffer , pelota.getX() - j - 10 , pelota.getY() - ( k +
7 ) , pelota.getX() - j - 10 , pelota.getY() - ( k + 7 ) , 0xFFFFFF
);
         line( buffer , pelota.getX() - j - 7 , pelota.getY() - k ,
pelota.getX() - j - 7 , pelota.getY() - k , 0xFFFFFF );
         line( buffer , pelota.getX() - j - 12 , pelota.getY() - ( k +
1 ) , pelota.getX() - j - 12 , pelota.getY() - ( k + 1 ) , 0x00FCFE
);
         line( buffer , pelota.getX() - j - 30 , pelota.getY() - ( k+
8 ) , pelota.getX() - j - 30 , pelota.getY() - ( k + 8 ) , 0x00FCFE
);
         line( buffer , pelota.getX() - j - 32 , pelota.getY() - ( k+
10 ) , pelota.getX() - j - 32 , pelota.getY() - ( k + 10 ) ,
0x00FCFE );
         line( buffer , pelota.getX() - j - 37 , pelota.getY() - ( k +
1 ) , pelota.getX() - j - 37 , pelota.getY() -( k + 1 ), 0x00FCFE
);
         line( buffer , pelota.getX() - j - 40 , pelota.getY() - ( k +
8 ) , pelota.getX() - j - 40 , pelota.getY() -( k + 8 ), 0x00FCFE
);
         line( buffer , pelota.getX() - j - 14, pelota.getY() - ( k +
10 ), pelota.getX() - j - 14 , pelota.getY() -( k + 10 ), 0x00FCFE
);
      }
   }
   else if( pelota.getX() < 700 && diry < 0 )
   //DIBUJOS ESTELA DE PELOTA PALETA1. DIRECCION DE Y
NEGATIVA
   {
      for( i=0 ; i<1000 ; i++ )
      {
         line( buffer , pelota.getX() - j , pelota.getY() + k ,
pelota.getX() - j , pelota.getY() + k , 0xFFFFFF );
         line( buffer , pelota.getX() - j - 5 , pelota.getY() + k + 3
, pelota.getX() - j - 5 , pelota.getY() + k + 3 , 0xFFFFFF );
         line( buffer , pelota.getX() - j - 15 , pelota.getY() + k +
7 , pelota.getX() - j - 15 , pelota.getY() + k + 7 , 0xFFFFFF );
         line( buffer , pelota.getX() - j - 3 , pelota.getY() + k + 3
, pelota.getX() - j - 3 , pelota.getY() + k + 3 , 0xFFFFFF );
         line( buffer , pelota.getX() - j - 10 , pelota.getY() + k +
7 , pelota.getX() - j - 10 , pelota.getY() + k + 7 , 0xFFFFFF );
         line( buffer , pelota.getX() - j - 7 , pelota.getY() + k ,
pelota.getX() - j - 7 , pelota.getY() + k , 0xFFFFFF );
         line( buffer , pelota.getX() - j - 12 , pelota.getY() + k +
1 , pelota.getX() - j - 12 , pelota.getY() + k + 1 , 0x00FCFE );
         line( buffer , pelota.getX() - j - 30 , pelota.getY() + k +
8 , pelota.getX() - j - 30 , pelota.getY() + k + 8 , 0x00FCFE );
         line( buffer , pelota.getX() - j - 32 , pelota.getY() + k +
10 , pelota.getX() - j - 32 , pelota.getY() + k + 10 , 0x00FCFE
);
         line( buffer , pelota.getX() - j - 37 , pelota.getY() + k +
1 , pelota.getX() - j - 37 , pelota.getY() + k + 1, 0x00FCFE );
         line( buffer , pelota.getX() - j - 40 , pelota.getY() + k +
8 , pelota.getX() - j - 40 , pelota.getY() + k + 8, 0x00FCFE );
         line( buffer , pelota.getX() - j - 14 , pelota.getY() + k +
10 , pelota.getX() - j - 14 , pelota.getY() + k + 10, 0x00FCFE );
      }
   }
   else if ( pelota.getX() > 700 )
   //PROPULSOR SET 0, PARA ASI DEJAR DE TENER ESTELA
   {
      funciones.setPropulsion( 0 );
   }

   if( pelota.getX() <= paleta.getAncho() + pelota.getAncho()
/ 2 )
   //ACUMULADOR DE VELOCIDAD
   {
      pelota.setVelX( pelota.getVelX() + 1 );
      pelota.setVelY( pelota.getVelY() + 1 );
   }
}
void golpe_especialp2( Pelota<int,float> &pelota ,
Paleta<int> paleta , BITMAP *buffer , int propulsion ,
Funcionalidad<int,bool> &funciones , int diry , Jugador<int>
&jugadores)
{
   int i = 0;
   int j = rand() % 80; // EJE X
   int k = rand() % 50; // EJE Y ( Y*2 = 100)

   //DIBUJOS ESTELA DE PELOTA PALETA2. DIRECCION DE Y
POSITIVA
   if( pelota.getX() > 300 && diry > 0 )
   {
      for( i=0 ; i<1000 ; i++ )
      {
```

```
        line( buffer , pelota.getX() + j , pelota.getY() - k ,
pelota.getX() + j , pelota.getY() - k , 0xFFFFFF );
        line( buffer , pelota.getX() + j + 5 , pelota.getY() - ( k +
3 ) , pelota.getX() + j + 5 , pelota.getY() - ( k + 3 ) , 0xFFFFFF
);
        line( buffer , pelota.getX() + j + 15 , pelota.getY() - ( k
+ 7 ) , pelota.getX() + j + 15 , pelota.getY() -( k + 7 ) ,
0xFFFFFF );
        line( buffer , pelota.getX() + j + 3 , pelota.getY() - ( k +
3 ) , pelota.getX() + j + 3 , pelota.getY() - ( k + 3 ), 0xFFFFFF );
        line( buffer , pelota.getX() + j + 10 , pelota.getY() - ( k
+ 7 ) , pelota.getX() + j + 10 , pelota.getY() - ( k + 7 ) ,
0xFFFFFF );
        line( buffer , pelota.getX() + j + 7 , pelota.getY() - k ,
pelota.getX() + j + 7 , pelota.getY() - k , 0xFFFFFF );
        line( buffer , pelota.getX() + j + 12 , pelota.getY() - ( k
+ 1 ) , pelota.getX() + j + 12 , pelota.getY() - ( k + 1 ) ,
0x00FCFE );
        line( buffer , pelota.getX() + j + 30 , pelota.getY() - ( k
+ 8 ) , pelota.getX() + j + 30 , pelota.getY() - ( k + 8 ) ,
0x00FCFE );
        line( buffer , pelota.getX() + j + 32 , pelota.getY() - ( k
+ 10 ) , pelota.getX() + j + 32 , pelota.getY() - ( k + 10 ) ,
0x00FCFE );
        line( buffer , pelota.getX() + j + 37 , pelota.getY() - ( k
+ 1 ) , pelota.getX() + j + 37 , pelota.getY() -(k + 1 ) ,
0x00FCFE );
        line( buffer , pelota.getX() + j + 40 , pelota.getY() - ( k
+ 8 ), pelota.getX() + j + 40 , pelota.getY() -(k + 8 ) ,
0x00FCFE );
        line( buffer , pelota.getX() + j + 14 , pelota.getY() - ( k
+ 10 ), pelota.getX() + j + 14 , pelota.getY() -(k + 10 ) ,
0x00FCFE );
      }
    }
    else if( pelota.getX() > 300 && diry < 0 )
    //DIBUJOS ESTELA DE PELOTA PALETA2. DIRECCION DE Y
NEGATIVA
    {
      for( i=0 ; i<1000 ; i++ )
      {
        line( buffer , pelota.getX() + j , pelota.getY() + k ,
pelota.getX()+j , pelota.getY() + k , 0xFFFFFF );
        line( buffer , pelota.getX() + j + 5 , pelota.getY() + k +
3 , pelota.getX() + j + 5 , pelota.getY() + k + 3 , 0xFFFFFF );
        line( buffer , pelota.getX() + j + 15 , pelota.getY() + k
+ 7 , pelota.getX() + j + 15 , pelota.getY() + k + 7 , 0xFFFFFF );
        line( buffer , pelota.getX() + j + 3 , pelota.getY() + k +
3 , pelota.getX() + j + 3 , pelota.getY() + k + 3, 0xFFFFFF );
        line( buffer , pelota.getX() + j + 10 , pelota.getY() + k
+ 7 , pelota.getX() + j + 10 , pelota.getY() + k + 7, 0xFFFFFF );
        line( buffer , pelota.getX() + j + 7 , pelota.getY() + k ,
pelota.getX() + j + 7 , pelota.getY() + k , 0xFFFFFF );
        line( buffer , pelota.getX() + j + 12 , pelota.getY() + k
+ 1 , pelota.getX() + j + 12 , pelota.getY() + k+ 1 , 0x00FCFE );
        line( buffer , pelota.getX() + j + 30 , pelota.getY() + k
+ 8 , pelota.getX() + j + 30 , pelota.getY() + k + 8 , 0x00FCFE
);
        line( buffer , pelota.getX() + j + 32 , pelota.getY() + k
+ 10 , pelota.getX() + j + 32 , pelota.getY() + k+ 10 ,
0x00FCFE );
        line( buffer , pelota.getX() + j + 37 , pelota.getY() + k
+ 1 , pelota.getX() + j + 37 , pelota.getY() + k + 1 , 0x00FCFE
);
        line( buffer , pelota.getX() + j + 40 , pelota.getY() + k
+ 8 , pelota.getX() + j + 40 , pelota.getY() + k + 8 , 0x00FCFE
);
        line( buffer , pelota.getX() + j + 14 , pelota.getY() + k
+ 10, pelota.getX() + j + 14 , pelota.getY() + k + 10 ,
0x00FCFE );
      }
    }
    else if ( pelota.getX() < 300 )
    //PROPULSOR SET 0. PARA ASI DEJAR DE TENER ESTELA
    {
      funciones.setPropulsion( 0 );
    }

    if( pelota.getX() >= ANCHO - paleta.getAncho() -
pelota.getAncho() / 2 )
    //ACUMULADOR DE VELOCIDAD
    {
      pelota.setVelX( pelota.getVelX() + 1 );
      pelota.setVelY( pelota.getVelY() + 1 );
    }
}
```

Dibujar.h

```
#ifndef DIBUJAR_H_INCLUDED
#define DIBUJAR_H_INCLUDED
#include <allegro.h>
#include "Pelotas.h"
#include "Jugadores.h"
#include "Fucionalidades.h"

#define ANCHO 1000
#define ALTO 500

//FUNCIONES DIBUJAR
void dibujar_mesa( BITMAP *Buffer );
void tablero( BITMAP *Buffer , Jugador<int> &jugadores ,
Funcionalidad<int,bool> &funciones );
void dibujar_puntos( BITMAP *Buffer );
void dibujar_pelota ( BITMAP *Buffer , Pelota<int,float>
&pelota );

#endif // DIBUJAR_H_INCLUDED
```

Dibujar.cpp

```
#include "Dibujar.h"

void dibujar_mesa( BITMAP* buffer )
//DIBUJAR MESA
{
	int i = 0;
	int j = 20;

	while( j <= ALTO )
  {
		rectfill( buffer , ANCHO/2 , i , ANCHO/2 , j , 0xFFFFFF );
		i = j + 10 ;
		j = i + 20 ;
	}
	line( buffer , ( ANCHO / 2) / 2 , ALTO / 2 , ( ANCHO / 2 ) * 1.5 , ALTO / 2 , 0x666666 );
	line( buffer , ( ANCHO / 2) / 2 , ALTO - ALTO , ( ANCHO / 2 ) / 2 , ALTO , 0x666666 );
	line( buffer , ( ANCHO / 2) * 1.5 , ALTO - ALTO , ( ANCHO / 2 ) * 1.5 , ALTO , 0x666666 );

	//MESA GASTADA (?)
	dibujar_puntos( buffer );

}
void tablero( BITMAP *Buffer , Jugador<int> &jugadores , Funcionalidad<int,bool> &funciones )
//DIBUJAR TABLERO
{
  line( Buffer , ANCHO - 140 , 0 , ANCHO - 140 , 80 , 0xFFFFFF );
  line( Buffer , ANCHO - 142 , 0 , ANCHO - 142 , 82 , 0xFFFFFF );
  line( Buffer , ANCHO - 142 , 82 , ANCHO , 82 , 0xFFFFFF );
  line( Buffer , ANCHO - 140 , 80 , ANCHO , 80 , 0xFFFFFF );

  line( Buffer , 140 , 0 , 140 , 80 , 0xFFFFFF );
  line( Buffer , 142 , 0 , 142 , 82 , 0xFFFFFF );
  line( Buffer , 142 , 82 , 0 , 82 , 0xFFFFFF );
  line( Buffer , 140 , 80 , 0 , 80 , 0xFFFFFF );

  textprintf_ex( Buffer , font , 10 , 20 , 0xD1D1D1 , - 1 , "Puntos P1: %d" , jugadores.getPuntaje1() );
  textprintf_ex( Buffer , font , 880 , 20 , 0xD1D1D1 , - 1 , "Puntos P2: %d" , jugadores.getPuntaje2() );
  if( jugadores.getDisparos1() >= 0)
  {
    textprintf_ex( Buffer , font , 10 , 40 , 0xD1D1D1 , - 1 , "Disparos 1: %d" , jugadores.getDisparos1() );
  }
  else
  {
    textprintf_ex( Buffer , font , 10 , 40 , 0xD1D1D1 , - 1 , "Disparos 1: 0" );
  }
  if( jugadores.getDisparos2() >= 0)
  {
    textprintf_ex( Buffer , font , 880 , 40 , 0xD1D1D1 , - 1 , "Disparos 2: %d" , jugadores.getDisparos2() );
  }
  else
  {
    textprintf_ex( Buffer , font , 880 , 40 , 0xD1D1D1 , - 1 , "Disparos 2: 0" );
  }
  textprintf_ex( Buffer , font , 900 , 480 , 0xD1D1D1 , - 1 , "LEVEL %d" , funciones.getNivel() );

}
void dibujar_puntos ( BITMAP *buffer )
//DIBUJAR PUNTOS
{
  line( buffer , 20 , 100 , 20 ,100 , 0xD1D1D1 ); line( buffer , 30 ,70 , 30 , 70 , 0xD1D1D1 ); line( buffer , 54 , 343 , 54 , 343 , 0xD1D1D1 );
  line( buffer , 56 , 250 , 56 ,250 , 0xD1D1D1 ); line( buffer , 67 , 120 , 67 , 120 , 0xD1D1D1 ); line( buffer , 12 , 123 , 12 , 123 , 0xD1D1D1 );
  line( buffer , 87 , 150 , 87 ,150 , 0xD1D1D1 ); line( buffer , 113 , 230 , 113 , 230 , 0xD1D1D1 ); line( buffer , 76 , 222 , 76 , 222 , 0xD1D1D1 );
  line( buffer , 92 , 300 , 92 ,300 , 0xD1D1D1 ); line( buffer , 143 , 430 , 143 , 430 , 0xD1D1D1 ); line( buffer , 101 , 111 , 101 , 111 , 0xD1D1D1 );
  line( buffer , 105 , 200 , 105 ,200 , 0xD1D1D1 ); line( buffer , 152 , 300 , 152 , 300 , 0xD1D1D1 ); line( buffer , 123 , 43 , 123 , 43 , 0xD1D1D1 );
  line( buffer , 520 , 406 , 520 ,406 , 0xD1D1D1 ); line( buffer , 187 , 465 , 187 , 465 , 0xD1D1D1 ); line( buffer , 322 , 54 , 322 , 54 , 0xD1D1D1 );
  line( buffer , 400 , 300 , 400 ,300 , 0xD1D1D1 ); line( buffer , 200 , 343 ,  200 , 343 , 0xD1D1D1 ); line( buffer , 876 , 12 , 876 , 12 , 0xD1D1D1 );
  line( buffer , 300 , 356 , 300 ,356 , 0xD1D1D1 ); line( buffer , 221 , 303 , 221 , 303 , 0xD1D1D1 ); line( buffer , 654 , 98 , 654 , 98 , 0xD1D1D1 );
  line( buffer , 320 , 743 , 320 ,743 , 0xD1D1D1 ); line( buffer , 234 , 222 , 234 , 222 , 0xD1D1D1 ); line( buffer , 634 , 76 , 634 , 76 , 0xD1D1D1 );
  line( buffer , 380 , 765 , 380 ,765 , 0xD1D1D1 ); line( buffer , 256 , 123 , 256 , 123 , 0xD1D1D1 ); line( buffer , 342 , 343 , 342 , 343 , 0xD1D1D1 );
  line( buffer , 380 , 220 , 380 ,220 , 0xD1D1D1 ); line( buffer , 298 , 76 , 298 , 76 , 0xD1D1D1 ); line( buffer , 654 , 434 , 654 , 434 , 0xD1D1D1 );
  line( buffer , 420 , 160 , 420 ,160 , 0xD1D1D1 ); line( buffer , 303 , 54 , 303 , 54 , 0xD1D1D1 ); line( buffer , 761 , 422 , 761 , 422 , 0xD1D1D1 );
  line( buffer , 900 , 150 , 900 ,150 , 0xD1D1D1 ); line( buffer , 376 , 230 , 376 , 230 , 0xD1D1D1 ); line( buffer , 982 , 477 , 982 , 477 , 0xD1D1D1 );
```

```
   line( buffer , 870 , 120 , 870 ,120 , 0xD1D1D1 ); line(
buffer , 354 , 333 , 354 , 333 , 0xD1D1D1 ); line( buffer , 422
, 223 , 422 , 223 , 0xD1D1D1 );
   line( buffer , 630 , 240 , 630 ,240 , 0xD1D1D1 ); line(
buffer , 331 , 232 , 331 , 232 , 0xD1D1D1 ); line( buffer , 500
, 342 , 500 , 342 , 0xD1D1D1 );
   line( buffer , 333 , 230 , 333 ,230 , 0xD1D1D1 ); line(
buffer , 400 , 276 , 400 , 276 , 0xD1D1D1 ); line( buffer , 744
, 211 , 744 , 211 , 0xD1D1D1 );
   line( buffer , 443 , 87 , 443 , 87 , 0xD1D1D1 ); line( buffer ,
442 , 290 , 442 , 290 , 0xD1D1D1 ); line( buffer , 433 , 155 ,
433 , 155 , 0xD1D1D1 );
   line( buffer , 657 , 97 , 657 , 97 , 0xD1D1D1 ); line( buffer ,
412 , 360 , 412 , 360 , 0xD1D1D1 ); line( buffer , 765 , 165 ,
765 , 165 , 0xD1D1D1 );
   line( buffer , 823 , 170 , 823 , 170 , 0xD1D1D1 ); line(
buffer , 456 , 387 , 456 , 387 , 0xD1D1D1 ); line( buffer , 333
, 187 , 333 , 187 , 0xD1D1D1 );
   line( buffer , 723 , 150 , 723 , 150 , 0xD1D1D1 ); line(
buffer , 476 , 456 , 476 , 456 , 0xD1D1D1 ); line( buffer , 223
, 165 , 223 , 165 , 0xD1D1D1 );
   line( buffer , 354 , 245 , 354 , 245 , 0xD1D1D1 ); line(
buffer , 490 , 410 , 490 , 410 , 0xD1D1D1 ); line( buffer , 254
, 187 , 254 , 187 , 0xD1D1D1 );
   line( buffer , 987 , 322 , 987 , 322 , 0xD1D1D1 ); line(
buffer , 501 , 402 , 501 , 402 , 0xD1D1D1 ); line( buffer , 987
, 376 , 987 , 376 , 0xD1D1D1 );
   line( buffer , 965 , 467 , 965 , 467 , 0xD1D1D1 ); line(
buffer , 520 , 489 , 520 , 489 , 0xD1D1D1 ); line( buffer , 543
, 465 , 543 , 465 , 0xD1D1D1 );
   line( buffer , 654 , 321 , 654 , 321 , 0xD1D1D1 ); line(
buffer , 550 , 123 , 550 , 123 , 0xD1D1D1 ); line( buffer , 656
, 423 , 656 , 423 , 0xD1D1D1 );
   line( buffer , 876 , 460 , 876 , 460 , 0xD1D1D1 ); line(
buffer , 860 , 410 , 860 , 410 , 0xD1D1D1 ); line( buffer , 760
, 410 , 760 , 410 , 0xD1D1D1 );
   line( buffer , 769 , 322 , 769 , 322 , 0xD1D1D1 ); line(
buffer , 932 , 402 , 932 , 402 , 0xD1D1D1 ); line( buffer , 987
, 376 , 987 , 376 , 0xD1D1D1 );
   line( buffer , 930 , 467 , 930 , 467 , 0xD1D1D1 ); line(
buffer , 965 , 489 , 965 , 489 , 0xD1D1D1 ); line( buffer , 840
, 465 , 840 , 465 , 0xD1D1D1 );
   line( buffer , 950 , 321 , 950 , 321 , 0xD1D1D1 ); line(
buffer , 870 , 276 , 870 , 276 , 0xD1D1D1 ); line( buffer , 760
, 423 , 760 , 423 , 0xD1D1D1 );
}
void dibujar_pelota( BITMAP *Buffer , Pelota<int,float>
&pelota )
//DIBUJAR PELOTA
{
        circlefill ( Buffer , pelota.getX() , pelota.getY() ,
pelota.getAncho() / 2 , 0xFFFFFF );
        circlefill ( Buffer , pelota.getX() , pelota.getY() ,
pelota.getAncho() / 6 , 0x000000 );
}
```

Funcionalidades.h

```
#ifndef FUCIONALIDADES_H_INCLUDED
#define FUCIONALIDADES_H_INCLUDED
#include <iostream>

//TEMPLATE CLASE, FUNCIONALIDAD
template <class type1,class type2>
class Funcionalidad //type1 = int, type2 = BOOL
{
   type1 contador;
   type1 resultado;
   type1 nivel;
   type1 tiempo;
   type1 disparos;
   type1 propulsion;
   type1 mensaje;
   type2 ventana;
   type2 portada;
   type2 nuevo;

   public:
   Funcionalidad () {
      contador = 0;
      resultado = 0;
      nuevo = false;
      nivel = 1;
      ventana = false;
      portada = false;
      tiempo++;
   }

   ~Funcionalidad () {

   }

   void setNivel ( type1 nivel ) {
      this->nivel = nivel;
   }

   void setContador ( type1 contador ) {
      this->contador = contador;
   }

   void setResultado ( type1 resultado ) {
      this->resultado = resultado;
   }

   void setNuevo ( type2 nuevo ) {
      this->nuevo = nuevo;
   }

   void setVentana ( type2 ventana ) {
      this->ventana = ventana;
   }

   void setPortada ( type2 portada ) {
      this->portada = portada;
   }

   void setTiempo ( type1 Tiempo ) {
      this->tiempo = Tiempo;
   }

   void setPropulsion ( type1 Propulsion ) {
      this->propulsion = Propulsion;
   }

   void setMensaje ( type1 Mensaje ) {
      this->mensaje = Mensaje;
   }

   type1 getMensaje () {
      return mensaje;
   }

   type1 getPropulsion () {
      return propulsion;
   }

   type1 getContador () {
      return contador;
   }

   type1 getResultado () {
      return resultado;
   }

   type1 getTiempo () {
      return tiempo;
   }

   type1 getNivel ( ) {
      return nivel;
   }

   type2 getVentana () {
      return ventana;
   }

   type2 getNuevo () {
      return nuevo;
   }

   type2 getPortada () {
      return portada;
   }
};

#endif // FUCIONALIDADES_H_INCLUDED
```