Universidad de Costa Rica

EIE

Escuela de Ingeniería Eléctrica

IE-0117 **Programación bajo plataformas abiertas**

Reporte Proyecto: Videojuego Duck Hunt

Ana Chaves Matamoros - B61982

Marlon Lazo Coronado - B43717

Alejandro Castillo Sequeira - B81787

Docente

M. Sc. Ricardo Román-Brenes

ricardo.roman@ucr.ac.cr

Contenido

Índice

Introducción

El presente proyecto consiste en el desarrollo de un videojuego ya que se considera que de esta manera se demuestra y se aplican de forma satisfactoria los conocimientos adquiridos en el curso IE-0117 Programación Bajo Plataformas Abiertas. Para llevar a cabo este propósito se utilizó el lenguaje de programación **C**.

Además, se hizo uso de distintas bibliotecas, entre las cuales destaca Simple and Fast Multimedia Library (SFML), debido a que se ha recomendado para programar de manera mas rápida y sencilla el proyecto. Esta biblioteca cuenta con funciones para gráficos, manipulación de imágenes, texto, sonidos, dispositivos de entrada (teclado, ratón y mandos de juego), así como rutinas para aritmética de punto fijo y acceso al sistema de archivos.[1].

El videojuego seleccionado es Duck Hunt. Este, cumple con las indicaciones y requerimientos del proyecto. Además, su desarrollo es adecuado al utilizar la biblioteca antes mencionada, mostrando una programación interesante, que de como resultado un videojuego entretenido y operable.

Figura 1: Videojuego Duck Hunt [2]

Reseña del Programa

El impacto que han tenido los videojuegos en las últimas cinco décadas aún no ha sido investigado con la debida profundidad por los científicos sociales. Es en esta categoría donde se puede apreciar el rápido avance de la tecnología. Ya que existe un cambio considerable desde el primer videojuego creado en 1952 llamado Nought and crosses a los que los usuarios utilizan en tiempos contemporáneos.

Duck Hunt es un videojuego creado y desarrollado por la empresa Nintendo en 1984 para la consola Nintendo Entertainment System (NES) [3].

Este videojuego es del tipo de temática cacería, consiste en disparar, mediante el uso de una pistola electrónica llamada Nintendo Zapper, a los objetivos que se muestran por pantallas (estos pueden ser patos o discos dependiendo del nivel en que se encuentre).

Al iniciar el videojuego se muestra una pantalla con 3 opciones de jugabilidad: Game A (1 duck), Game B (2 ducks) y Game C (Clay shooting).

Figura 2: Menú principal [2]

Si selecciona el juego A, esta opción despliega un pato a la vez. En cada turno, el pato se moverá por la pantalla de manera aleatoria y el objetivo del juego es dispararle al pato haciendo uso del Nintendo Zapper. Este permanecerá en la pantalla por un tiempo definido, si en este tiempo se logra acertar entonces el pato caerá y será capturado por un perro. Caso contrario, si no se logra acertarle al pato, este volará y ya no estará disponibile para dispararle, en este caso el perro se reirá por el desacierto y saldrá otro pato para el siguiente turno.

Por turno se tienen un máximo de 3 disparos, por lo tanto, si ninguno de los disparos es acertado el tiempo pasará y el pato se irá. Por cada pantalla hay un total de 10 patos que saldrán en total, así cada pantalla posee un nivel mínimo de aciertos que hay que completar para pasar a la siguiente. En total se tienen 99 pantallas y cada vez que se sube de nivel los patos se mueven más rápido por lo que la dificultad aumenta.

Esta opción A también está disponible para dos jugadores, de este modo uno de ellos es capaz de darle dirección al pato mediante el uso del control y el otro debe de acertar con el Zapper.

La opción B es igual a la opción A pero este posee la diferencia de que por cada turno aparecen 2 patos en la pantalla en vez de uno solo. Igual se cuenta con la misma cantidad de disparos por turno y con la misma cantidad de pantallas. Esta opción solo está disponible para un jugador.

El modo C es parecido a los anteriores solo que este posee discos en vez de patos, por cada turno dos discos son lanzados al mismo tiempo al aire. Estos se alejarán con el tiempo hasta llegar a desaparecer de la pantalla. Al igual que los modos anteriores, se tienen 3 disparos por cada turno y un total de 99 pantallas.

La puntuación obtenida depende de la cantidad de patos o discos acertados. Además,

existen patos de diferentes colores y entre menos comunes sean estos, mayor será la puntuación si se acierta. En caso de que en una pantalla se logren todos los aciertos, se le agrega un bonus a la puntuación. En el cuadro 1 se muestra la puntuación asignada por cada rubro por nivel y en el cuadro 2 se muestra la cantidad de aciertos necesarios para pasar al siguiente nivel.

	Niveles 1-5	**Niveles 6-10**	**Niveles 11-15**	**Niveles 16-20**	**Niveles 21-99**
Pato negro	500	800	1000	1000	1000
Pato azul	1000	1500	2000	2000	2000
Pato rojo	1500	2400	3000	3000	3000
Disco	1000	1500	2000	2000	2000
Bonus	10000	10000	15000	20000	30000

Cuadro 1: Sistema de puntuación [2]

Niveles	**Aciertos**
1 al 10	6/10
11 al 12	7/10
13 al 14	8/10
15 al 19	9/10
20 al 99	10/10

Cuadro 2: Cantidad de aciertos necesarios [2]

Funcionamiento del programa

Este proyecto tiene como objetivo alcanzar una gran similitud con el videojuego original, sin embargo, se realizaron distintas modificaciones para hacer más fácil su implementación. La principal diferencia con el juego original consiste en que en este programa se utiliza el mouse para poder realizar los disparos en vez de utilizar un Nintendo Zapper. Se utilizan además varios botones del teclado para poder seleccionar el modo de juego en el menú principal.

En el menú de bienvenida se habilitaron solamente las opciones A y B del videojuego original para poder jugar con uno o dos patos por turno respectivamente. Se utilizaron la misma cantidad de patos (10) por nivel con los mismos colores del videojuego original (negro, azul y rojo). Se definió una probabilidad de 70 % para la aparición del pato azul, 20 % para el pato negro y 10 % para el pato rojo. Esto se generó mediante la función rand(), la cual se implementó para que generara un número aleatorio de 0 a 100 y de acuerdo al número obtenido se le asigna una textura distinta al pato. De esta manera, si el número generado es menor a 70, se retorna la imagen del pato azul; si el número está entre 70 y 90 se retorna la imagen del pato negro y para los demás casos se retorna la imagen del pato rojo.

Para la opción A, no se habilitó la opción de un segundo jugador que controle la dirección del pato si no que esta dirección es aleatoria y solo se cuenta con un jugador en todos los modos.

Además, se redujo la cantidad de niveles de modo que en este caso sean 5 pantallas en total y no 99 como en el videojuego original.

Se incorporó además el movimiento y visualización de la mira del arma, música de animación, sonido de botones, registro de la puntuación más alta y registro del número de aciertos por nivel, todo esto mediante una interfaz gráfica.

Se utilizó la biblioteca SFML para incorporar los sonidos, imágenes, texto y demás variables importantes para la animación de la interfaz. Además, se empleó la biblioteca time para permitir un control de los tiempos que los patos permanecen en pantalla y los tiempos que transcurren entre cada round.

Se utilizaron los siguientes rubros para poder pasar de nivel y para las puntuaciones logradas tomando como referencia las del videojuego original.

	Niveles 1-2	**Niveles 3-4**	**Nivel 5**
Pato azul	500	800	1000
Pato negro	1000	1500	2000
Pato rojo	1500	2400	3000
Bonus	10000	15000	20000

Cuadro 3: Sistema de puntuación

Nivel	**Aciertos**
1	6/10
2	7/10
3	8/10
4	9/10
5	10/10

Cuadro 4: Cantidad de aciertos necesarios

Para poder implementar estos rubros se realizaron distintas funciones. La función score recibe como argumentos la imagen del pato generada, las texturas con los 3 del colores de los patos, una variable boolena p1 que indica si hubo un acierto y el nivel en el cual se encuentra el usuario. Lo primero que se hace es verificar si hubo un acierto, esto mediante el argumento p1. En caso de que se cumpla esta condición, se compara el color del pato con las distintas texturas para verificar el color del pato y mediante

estructuras if se realiza la puntuación de acuerdo al color y al nivel. En caso de que la variable p1 sea falsa, se retorna un 0.

La función aciertosN se encarga de recibir como argumento el nivel en el que se encuentra el usuario y retorna la cantidad de aciertos necesarios para pasar al siguiente nivel.

Una vez finalizado cada nivel, se utiliza la función bonus para verificar si se obtuvieron los 10 aciertos y de esta manera se retorna el valor de acuerdo a los valores propuestos.

Para realizar la comparación de la mira con el pato, se tomó en consideración el centro de la imagen que representa la mira. Si este centro toca la imagen en alguno de sus puntos se genera un acierto mediante una variable booleana. Para esto se emplearon las funciones sfMouse_getPosition (para la posición de la mira) y sfSprite_getPosition (para el pato). A estas posciones se le sumaron los valores adecuados tanto en x como en y para tomar en consideración el área total de la imagen (76 x 62 pixeles) y el centro de la mira.

El movimiento del pato fue generado de forma que se obtuviera la posición de la imagen y a esta posición se le sumara un pequeño número aleatorio en ambas coordenadas para que el movimiento del pato fuera lo más suave posible.

Los niveles y los rounds fueron controlados mediante ciclos y bucles, principalmente haciendo uso del bucle for. Para los rounds, se cronometró el tiempo destinado a cada uno. Para los niveles había que verificar si los aciertos obtenidos eran mayores o igual a los requeridos para pasar al siguiente nivel, esto se logró mediante la condición if.

Además, se implementaron la funciones GameA y GameB de manera que la primera utiliza las funciones anteriores y está configurada para generar un pato por round y la segunda esta configurada para generar dos patos por round.

Experimentos realizados

Para verificar el correcto funcionamiento del programa se realizaron varias pruebas.

En la primer prueba se comenta el código que le da movimiento a los patos de manera que estos aparezcan estáticos en la pantalla, y que así sea más fácil acertar. Se verifica entonces en la terminal que las puntuaciones coincidan con los valores del cuadro 3, según el color del pato acertado.

Por otra parte, este experimento nos permite afirmar que las coordenadas de la mira están dentro del rango de las coordenadas de la imagen del pato.

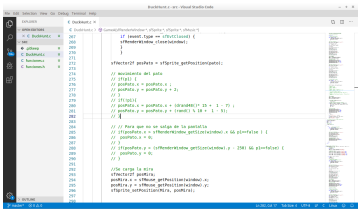

Figura 3: Código de movimiento comentado

Luego de realizar la prueba y obtener los resultados, se evidencia como el programa funciona correctamente, brindando una puntuación de 500 al acertar un pato azul, una puntuación de 1000 para el caso de los patos negros, y para los rojos, 1500 puntos. Lo anterior se puede apreciar en la Figura 7, en los aciertos 5, 9 y 6 respectivamente.

Figura 4: Pato azul acertado

Figura 5: Pato negro acertado

Figura 6: Pato rojo acertado

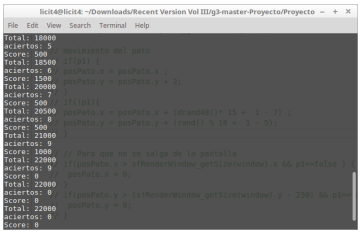

Figura 7: Análisis de resultados en terminal

Para la segunda prueba, se mantiene el pato en movimiento y se verifica que no se sube de nivel si no se logran los aciertos descritos en el cuadro 4. Esta prueba se realiza tanto en el modo A y B.

Al ejecutar el juego, y dejarlo correr sin disparar ni obtener puntos durante los diez rounds del primer nivel, nos damos cuenta que el programa no permite avanzar al nivel 2, sino que muestra el mensaje "Game Over", y lleva al jugador a la pantalla del menú principal.

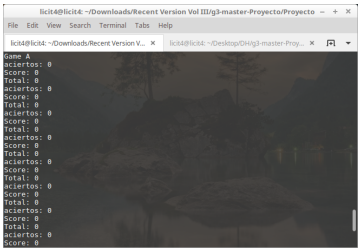

Figura 8: Resultados Game A

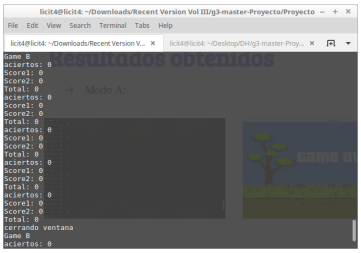

Figura 9: Resultados Game B

Figura 10: Mensaje de Game Over

Estas pruebas son suficientes para verificar además los demás requerimientos del juego: color del pato aleatorio, música deseada, etc.

Resultados obtenidos

Los resultados obtenidos cumplieron con los objetivos propuestos para el proyecto, primero se logró desplegar una ventana en la cual se pudo pintar las imágenes, segundo, se genero eventos tales como cerrar la ventana, interacciones con el teclado, seleccionar una imagen, tercero, utilización del mouse para generar sonidos o mover la ventana

Además, se cumplio el trabajo de animar el juego de diversas formas tales como mover imágenes de forma pseudoaleatorias, imprimir texto en la ventana, cambiar el cursor del mouse por una mira, seleccionar una imagen y generar más eventos, incorporar sonido, contar e imprimir los aciertos y su puntuación, cambiar la imagen al ser elegida y generar sonidos, salir del programa mediante un botón, niveles de dificultad para el juego, modos del juego

De manera específica uno de los objetivos era imitar el juego Duck Hunt original de una forma satisfactoria, esto ha sido posible porque logró que los patos aparezcan y se mueven de formas y frecuencias diferentes para animar el juego y generar diferentes niveles de dificultad

Además de esto se ha logrado darle a las imágenes de los patos cierta limitación al hacerlos que al llegar a un extremo de la ventana estos aparezcan en el otro extremo y de esa forma hacer el juego más atractivo.

Por último, el cursor del mouse fue reemplazado por una imagen que genera la mira con la que se dispara, al coincidir el área de la imagen del pato y la de la mira y dar clic sobre esta, se han generado eventos tales como el sonido de la mira el del pato y se cambia la imagen del pato original por la del pato muerto y se le da el efecto de caída.

Además de esto, se han creado dos dos modos funcionales del juego, el modo A y el modo B. Estos representan diferentes niveles de dificultad. Además de esto se ha logrado generar diferentes tipos de puntuaciones dependiendo del tipo de pato que se acierta e imprimirlos sobre la ventana.

Gracias a los dos experimentos realizados, se verificó que todas las puntaciones y bonus coincidieran con las propuestas en el cuadro 3. Además, para la segunda prueba se observó que en ambos modos de juego, si no se cumplen los aciertos necesarios en cada nivel, la pantalla despliega el mensaje *Game Over* y se devuelve al menú principal.

Conclusiones

- Se logró la correcta implementación del videojuego propuesto, el cual posee gran similitud con el videojuego original.
- La biblioteca SFML permite crear aplicaciones multiplataformas.
- Los distintos módulos que incorpora SFML, permiten desarrollar con mayor facilidad videojuegos sencillos.
- La compatibilidad de SFML con gran variedad de formatos de fuente, audio e imagen, permiten un desarrollo eficaz del programa.
- SFML es una biblioteca multimedia, no un motor para hacer videojuegos, representa solo la base sobre la que construir un videojuego o un motor.
- La biblioteca time sirve para manejar tiempo de manera sencilla en la implementación del programa.
- Los ciclos y bucles fueron fundamentales para la ejecución del programa.

Referencias

[1] SFML, “Documentación CSFML,” Recuperado de : https://www.cs.rit.edu/doc/libcsfml-doc/html/globals_func.htm, s.f.

[2] StrategyWiki, “Duck Hunt,” Recuperado de : https://strategywiki.org/wiki/Duck_Hunt, s.f.

[3] Nintendo of America Inc, “DUCK HUNT: Instruction Booklet,” Recuperado de : http://www.thealmightyguru.com/Wiki/images/6/6b/Duck_Hunt_-_NES_-_Manual.pdf, 1984.

Anexos

En esta sección se presentan los códigos empleados para la implmentación del video-juego.

```
#include <SFML/Audio.h>
#include <SFML/Graphics.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include<time.h>
#include <stdbool.h>

#include "./funciones.h"

void main()
{
    //Se crea la ventana
    sfVideoMode mode = {1000, 600, 32};
    sfRenderWindow *window = sfRenderWindow_create(mode, "Duck Hunt", sfResize
        | sfClose, NULL);

    //Se crea el texto "Duck Hunt"
    sfFont *style = sfFont_createFromFile("res/pd.ttf");
    sfText *text = sfText_create();
    sfText_setFont(text, style);
    sfText_setString(text, "Duck Hunt");
    sfText_setColor(text, sfWhite);
    sfText_setCharacterSize(text, 200);
    sfVector2f v;
    v = sfText_getPosition(text);
    v.x=v.x+70;
    v.y=v.y+10;
    sfText_setPosition(text, v);

    //
    sfText *text6 = sfText_create();
    sfText_setFont(text6, style);
    sfText_setString(text6, "__________");
    sfText_setColor(text6, sfBlue);
    sfText_setCharacterSize(text6, 142);
    sfVector2f v6;
    v6.x=75; v6.y=70;
    sfText_setPosition(text6, v6);

    //Se crea el texto "Game A"
    sfText *text2 = sfText_create();
    sfText_setFont(text2, style);
    sfText_setString(text2, "Game A");
    sfText_setColor(text2, sfWhite);
    sfText_setCharacterSize(text2, 60);
    sfVector2f v2;
    v2.x=200; v2.y=300;
    sfText_setPosition(text2, v2);

    sfText *text02 = sfText_create();
    sfText_setFont(text02, style);
    sfText_setString(text02, "(press key A)");
    sfText_setColor(text02, sfCyan);
    sfText_setCharacterSize(text02, 25);
    sfVector2f v02;
    v02.x=400; v02.y=325;
    sfText_setPosition(text02, v02);

    //Se crea el texto "Game B"
    sfText *text3 = sfText_create();
    sfText_setFont(text3, style);
    sfText_setString(text3, "Game B");
    sfText_setColor(text3, sfWhite);
    sfText_setCharacterSize(text3, 60);
    sfVector2f v3;
    v3.x=200; v3.y=400;
    sfText_setPosition(text3, v3);

    sfText *text03 = sfText_create();
    sfText_setFont(text03, style);
    sfText_setString(text03, "(press key B)");
    sfText_setColor(text03, sfCyan);
    sfText_setCharacterSize(text03, 25);
```

```
    sfVector2f v03;
    v03.x=400; v03.y=425;
    sfText_setPosition(text03, v03);

    //Se carga la imagen de fondo
    sfVector2f V;
    sfSprite* bg = sfSprite_create();
    sfTexture* t = sfTexture_createFromFile("res/background.jpg", NULL);
    sfSprite_setTexture(bg, t, sfTrue);
    float x = (float)sfTexture_getSize(t).x;
    float y = (float)sfTexture_getSize(t).y;
    V = (sfVector2f){sfRenderWindow_getSize(window).x/x,
        sfRenderWindow_getSize(window).y/y};
    sfSprite_setScale(bg, V);

    //Se carga la imagen de la mira
    sfSprite* Mira = sfSprite_create();
    sfSprite_setTexture(Mira, sfTexture_createFromFile("res/mira.png", NULL),
        sfTrue);
    V = (sfVector2f){0.8, 0.8};
    sfSprite_scale(Mira, V);

    //Se carga la m sica
    sfMusic* tiro = sfMusic_createFromFile("res/tiro.ogg");
    sfMusic* title = sfMusic_createFromFile("res/title.ogg");

    //Se empieza el evento
    sfEvent event;
    sfMusic_play(title);

    while (sfRenderWindow_isOpen(window))
    {
        //Se esconde el cursor
        sfRenderWindow_setMouseCursorVisible(window, sfFalse);

        while (sfRenderWindow_pollEvent(window, &event))
        {
            //Se sale si se presiona la X
            if (event.type == sfEvtClosed)
            {
                printf("cerrando ventana\n");
                sfRenderWindow_close(window);
            }

            //Se crea la opci n Game A (solo un pato)
            if (event.key.code == sfKeyA)
            {
                sfMusic_pause(title);
                printf("Game A\n");

                //Suena la musica
                GameA(window, bg, Mira, tiro);
            }

            //Se crea la opci n Game B (dos patos)
            if (event.key.code == sfKeyB)
            {
                printf("Game B\n");
                //Suena la musica
                sfMusic_pause(title);
                GameB(window, bg, Mira, tiro);
            }
        }

        sfRenderWindow_clear(window, sfBlack);
        sfRenderWindow_drawText(window, text, NULL);
        sfRenderWindow_drawText(window, text2, NULL);
        sfRenderWindow_drawText(window, text3, NULL);
        sfRenderWindow_drawText(window, text02, NULL);
        sfRenderWindow_drawText(window, text03, NULL);
        sfRenderWindow_drawText(window, text6, NULL);
        sfRenderWindow_display(window);
    }
    return 0;
}
```

DuckHunt.c

```c
#include <SFML/Audio.h>
#include <SFML/Graphics.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include<time.h>
#include <stdbool.h>

#include "./funciones.h"

// Convierte un n mero entero en texto
char *itoa(int i){
  char *res = malloc(8*sizeof(int));
  sprintf(res, "%d", i);
  return res;
}

// Genera un pato aleatorio
sfSprite* patoAleatorio(sfTexture* tP1,sfTexture* tP2, sfTexture* tP3){
  
  int numero = rand() %101;
  sfSprite* pato= sfSprite_create();

  //70 %de probabilidad de que el pato sea azul
  if (numero < 70)
  {sfSprite_setTexture(pato, tP1, sfTrue);}

  //20 %de probabilidad de que el pato sea negro
  else if (70<numero && numero<90)
  {sfSprite_setTexture(pato, tP2, sfTrue);}

  //10 %de probabilidad de que el pato sea rojo
  else
  {sfSprite_setTexture(pato, tP3, sfTrue);}
  return pato;
}


// Genera la puntuaci n
int score(sfSprite* pato, sfTexture* tPM1,sfTexture* tPM2, sfTexture* tPM3,
    bool p1, int nivel){
int score;

sfTexture* texture = sfSprite_getTexture(pato);

if(p1){
  if(texture == tPM1 && (nivel ==1 || nivel ==2) ){
  score = 500;
  }
  else if(texture == tPM2 && (nivel ==1 || nivel ==2) ){
  score = 1000;
  }
  else if(texture == tPM3 && (nivel ==1 || nivel ==2) ){
  score = 1500;
  }
  else if(texture == tPM1 && (nivel ==3 || nivel ==4) ){
  score = 800;
  }
  else if(texture == tPM2 && (nivel ==3 || nivel ==4) ){
  score = 1500;
  }
  else if(texture == tPM3 && (nivel ==3 || nivel ==4) ){
  score = 2400;
  }
  else if(texture == tPM1 && (nivel == 5) ){
  score = 1000;
  }
  else if(texture == tPM2 && (nivel == 5) ){
  score = 2000;
  }
  else if(texture == tPM3 && (nivel == 5) ){
  score = 3000;
}}

else{
  score = 0;
}
return score;
}
```

```
80 //_Aciertos_necesarios_para_pasar_al_siguiente_nivel
   int_aciertosN(int_nivel){
82 int_N;
   if(nivel_==1){
84 N_=_6;}␣

86 else_if(nivel_==2){
   N_=_7;}␣
88
   else_if(nivel_==3){
90 N_=_8;}␣

92 else_if(nivel_==4){
   N_=_9;}␣
94
   else{
96 N=10;
   }
98
   return_N;
100 __}

102 _int_bonus(int_nivel){
   ␣␣_int_N;
104 if(nivel_==1_||__nivel_==2_){
   N_=_10000;}␣
106
   else_if(nivel_==_3_||__nivel_==_4_){
108 N_=_15000;}␣

110 else{
    N_=_20000;}␣
112
    return_N;
114 _}

116 //_Creacion_del_pato_muerto
    sfSprite*_patomuerto(sfSprite*_pato,_sfTexture*_tPM1,sfTexture*_tPM2,sfTexture*
        _tPM3,_sfTexture*_tP1,sfTexture*_tP2,sfTexture*_tP3){
118 sfTexture*__texture_=_sfSprite_getTexture(pato);
    if(texture_==_tP1_){
120 sfSprite_setTexture(pato,_tPM1,_sfTrue);
    }
122 else_if(texture_==_tP2){
    sfSprite_setTexture(pato,_tPM2,_sfTrue);
124 }␣
    else_if(texture_==_tP3)_{
126 sfSprite_setTexture(pato,_tPM3,_sfTrue);
    }
128 return_pato;
    }
130

132 //_Opci n_de_juego_A_:_Un_pato_por_turno
    void_GameA(sfRenderWindow_*_window_,_sfSprite*_bg_,_sfSprite*_Mira_,_sfMusic*_
        tiro_){
134 ␣␣␣␣␣sfEvent_event;
    ␣␣␣␣␣//_Variables_importantes
136 ␣␣␣␣␣int_total=_0;
    ␣␣␣␣␣bool_sigNivel=true;
138 ␣␣␣␣␣_sfMusic*_hit_=_sfMusic_createFromFile("res/hit.ogg");
    ␣␣␣␣␣_sfMusic*_music_=_sfMusic_createFromFile("res/music.ogg");
140 ␣␣␣␣␣
    ␣␣␣␣␣_//Imagenes_de_los_patos
142 ␣␣␣␣␣sfTexture*_tP1_=_sfTexture_createFromFile("res/pato1.png",_NULL);
    ␣␣␣␣␣sfTexture*_tP2_=_sfTexture_createFromFile("res/pato2.png",_NULL);
144 ␣␣␣␣␣sfTexture*_tP3_=_sfTexture_createFromFile("res/pato3.png",_NULL);
    ␣␣␣␣␣
146 ␣␣␣␣␣sfTexture*_tPM1_=_sfTexture_createFromFile("res/patoM1.png",_NULL);
    ␣␣␣␣␣sfTexture*_tPM2_=_sfTexture_createFromFile("res/patoM2.png",_NULL);
148 ␣␣␣␣␣sfTexture*_tPM3_=_sfTexture_createFromFile("res/patoM3.png",_NULL);␣␣␣␣

150 ␣␣␣
    ␣␣␣for(_int_Nivel_=_1;_Nivel_<_6;_Nivel++){
152 ␣␣␣␣␣if(sigNivel){
    ␣␣␣␣␣int_aciertos_=_0;
154 ␣␣␣␣␣sfMusic_play(music);␣␣
    ␣␣␣␣␣//_Tiempo_de_espera(2_segundos)
156 ␣␣␣␣␣//_Se_indica_el_nivel_en_el_que_se_encuentra
```

```
    time_t timeT = time(NULL);
    while (time(NULL) - timeT) < 2){
        if(sfRenderWindow_pollEvent(window, &event)){
        if (event.type == sfEvtClosed) {
            sfRenderWindow_close(window);
        }
        }

        //Se crea el texto "Level x"
        sfFont *style = sfFont_createFromFile("res/pd.ttf");
        sfText *textLevel = sfText_create();
        sfText_setFont(textLevel, style);

        char *textN = itoa (Nivel);

        sfText_setString(textLevel, textN);

        free(textN);

        sfText_setCharacterSize(textLevel, 100);
        sfVector2f vL;
        vL= sfText_getPosition(textLevel);
        vL.x=vL.x+630; vL.y=vL.y+150;
        sfText_setPosition(textLevel, vL);

        sfText *textOL = sfText_create();
        sfText_setFont(textOL, style);
        sfText_setString(textOL, "Level");
        sfText_setCharacterSize(textOL, 100);
        sfVector2f vOL;
        vOL= sfText_getPosition(textOL);
        vOL.x=vOL.x+350; vOL.y=vOL.y+150;
        sfText_setPosition(textOL, vOL);

        //Se carga la mira
        sfVector2f posMira;
        posMira.x = sfMouse_getPosition(window).x;
        posMira.y = sfMouse_getPosition(window).y;
        sfSprite_setPosition(Mira, posMira);

        //Se imprime la imagen
        sfRenderWindow_clear(window, sfBlack);
        sfRenderWindow_drawSprite(window, bg, NULL);
        sfRenderWindow_drawText(window, textLevel, NULL);
        sfRenderWindow_drawText(window, textOL, NULL);
        sfRenderWindow_drawSprite(window, Mira, NULL);
        sfRenderWindow_display(window);
    }

    //Se empiezan los Rounds
    for(int i = 0; i < 10; i++){

        //Se carga el color del pato aleatoriamente
        sfSprite* pato = patoAleatorio(tP1,tP2, tP3);

        //variables para el pato muerto
        bool p1=false;

        //Tiempo de espera (2 segundos)
        time_t time1 = time(NULL);
        while(time(NULL) - time1) < 2){
        if(sfRenderWindow_pollEvent(window, &event)){
        if (event.type == sfEvtClosed) {
            sfRenderWindow_close(window);
        }
        }

        //Se crea el texto "Round x"
        sfFont *style = sfFont_createFromFile("res/pd.ttf");
        sfText *text4 = sfText_create();
        sfText_setFont(text4, style);

        char *textRound = itoa (i + 1);

        sfText_setString(text4, textRound);

        free(textRound);

        sfText_setCharacterSize(text4, 100);
        sfVector2f v4;
```

```
    sfVector2f v4=sfText_getPosition(text4);
    v4.x=v4.x+630; v4.y=v4.y+150;
    sfText_setPosition(text4, v4);

    sfText *text5 = sfText_create();
    sfText_setFont(text5, style);
    sfText_setString(text5, "Round");
    sfText_setCharacterSize(text5, 100);
    sfVector2f v5;
    v5=sfText_getPosition(text5);
    v5.x=v5.x+350; v5.y=v5.y+150;
    sfText_setPosition(text5, v5);

    //Se carga la mira
    sfVector2f posMira;
    posMira.x = sfMouse_getPosition(window).x;
    posMira.y = sfMouse_getPosition(window).y;
    sfSprite_setPosition(Mira, posMira);

    //Se imprime la imagen
    sfRenderWindow_clear(window, sfBlack);
    sfRenderWindow_drawSprite(window, bg, NULL);
    sfRenderWindow_drawText(window, text4, NULL);
    sfRenderWindow_drawText(window, text5, NULL);
    sfRenderWindow_drawSprite(window, Mira, NULL);
    sfRenderWindow_display(window);
    }

    //Se crea el texto "Score: x"
    sfFont *style = sfFont_createFromFile("res/pd.ttf");
    sfText *text6 = sfText_create();
    sfText_setFont(text6, style);

    char *textScore = itoa(total);

    sfText_setString(text6, textScore);

    free(textScore);

    sfText_setCharacterSize(text6, 20);
    sfVector2f v6;
    v6=sfText_getPosition(text6);
    v6.x=v6.x+870; v6.y=v6.y +550;
    sfText_setPosition(text6, v6);

    sfText *text7 = sfText_create();
    sfText_setFont(text7, style);
    sfText_setString(text7, "Score: ");
    sfText_setCharacterSize(text7, 20);
    sfVector2f v7;
    v7=sfText_getPosition(text7);
    v7.x=v7.x+800; v7.y=v7.y+550;
    sfText_setPosition(text7, v7);

    //Se crea el texto "Hits: x"
    sfText *text8 = sfText_create();
    sfText_setFont(text8, style);

    char *textHits = itoa(aciertos);

    sfText_setString(text8, textHits);

    free(textHits);

    sfText_setCharacterSize(text8, 20);
    sfVector2f v8;
    v8=sfText_getPosition(text8);
    v8.x=v8.x+870; v8.y=v8.y +520;
    sfText_setPosition(text8, v8);

    sfText *text9 = sfText_create();
    sfText_setFont(text9, style);
    sfText_setString(text9, "Hits: ");
    sfText_setCharacterSize(text9, 20);
    sfVector2f v9;
    v9=sfText_getPosition(text9);
    v9.x=v9.x+800; v9.y=v9.y+520;
    sfText_setPosition(text9, v9);

    // Se mueve el pato (5 segundos)
```

```
    time1 = time(NULL);
    while (time(NULL) - time1) < 5){

        if(sfRenderWindow_pollEvent(window, &event)){
        if (event.type == sfEvtClosed) {
                sfRenderWindow_close(window);
        }
        }

    sfVector2f posPato = sfSprite_getPosition(pato);

    //movimiento del pato
    if(p1) {
    posPato.x = posPato.x;
    posPato.y = posPato.y + 2;
    }
    if(!p1){
            posPato.x = posPato.x + (drand48()* 15 + 1 - 7);
    posPato.y = posPato.y + (rand() %10 + 1 - 5);
    }

    //Para que no se salga de la pantalla
    if(posPato.x > sfRenderWindow_getSize(window).x && p1==false ) {
        posPato.x = 0;
    }
    if(posPato.y > (sfRenderWindow_getSize(window).y - 250) && p1==false) {
        posPato.y = 0;
    }

    //Se carga la mira
    sfVector2f posMira;
    posMira.x = sfMouse_getPosition(window).x;
    posMira.y = sfMouse_getPosition(window).y;
    sfSprite_setPosition(Mira, posMira);

  //Si se presiona el mouse
    if ((event.type == sfEvtMouseButtonPressed)) {
        sfMusic_play(tiro);
        //Se compara la posicion de la mira con la del pato
        if((posMira.x+25)>posPato.x && (posPato.x + 76)>(posMira.x+25)){
        if((posMira.y+25)>posPato.y && (posPato.y + 62) >(posMira.y+25)){

        sfSprite* patoX = patomuerto(pato, tPM1,tPM2,tPM3, tP1, tP2,tP3);
        sfMusic_play(hit);
            p1=true;
            pato = patoX;
        }}
        }

    //Se imprime la imagen
    sfSprite_setPosition(pato, posPato);
    sfRenderWindow_drawSprite(window, bg, NULL);
    sfRenderWindow_drawSprite(window, pato, NULL);
    sfRenderWindow_drawSprite(window, Mira, NULL);
    sfRenderWindow_drawText(window, text6, NULL);
    sfRenderWindow_drawText(window, text7, NULL);
    sfRenderWindow_drawText(window, text8, NULL);
    sfRenderWindow_drawText(window, text9, NULL);
    sfRenderWindow_display(window);
    }
    if(p1){
        aciertos ++;
    }
    printf("aciertos: %d  \n", aciertos);

    int puntuacion = score(pato, tPM1, tPM2, tPM3, p1, Nivel);
    total= total + puntuacion;
    printf("Score: %d  \n", puntuacion);
    printf("Total: %d  \n", total);
  }

  //Si se cumplieron los aciertos, se pasa al siguiente nivel
  int Necesarios =aciertosN(Nivel);
  if(aciertos >= Necesarios){
  sigNivel = true;}
  else{
  sigNivel = false;}
```

```
  //Bonus de perfect
398   if(aciertos ==10){
      total = total + bonus(Nivel);
400   }
  }
402   else{
          //Tiempo de espera(5 segundos)
404       time_t timeT = time(NULL);
          while((time(NULL) - timeT) <5){
406           if(sfRenderWindow_pollEvent(window, &event)){
              if (event.type == sfEvtClosed) {
408               sfRenderWindow_close(window);
              }
410           }

412       //Se crea el texto "Game Over"
          sfFont *style = sfFont_createFromFile("res/pd.ttf");
414       sfText *textOver = sfText_create();
          sfText_setFont(textOver, style);
416       sfText_setString(textOver, "Game Over");
          sfText_setCharacterSize(textOver, 100);
418       sfVector2f vO;
          vO= sfText_getPosition(textOver);
420       vO.x=vO.x+300; vO.y=vO.y+150;
          sfText_setPosition(textOver, vO);
422
          //Se imprime la imagen
424       sfRenderWindow_clear(window, sfBlack);
          sfRenderWindow_drawSprite(window, bg, NULL);
426       sfRenderWindow_drawText(window, textOver, NULL);
          sfRenderWindow_display(window);
428       }

430   return;
  }
432 }
  return;
434 }


436
  // Opci n de juego B : Dos patos por turno
438 void GameB(sfRenderWindow *window, sfSprite* bg, sfSprite* Mira, sfMusic*
      tiro ){
      sfEvent event;
440   //Se inicia el juego
      int total=0;
442   bool sigNivel=true;
      sfMusic* hit = sfMusic_createFromFile("res/hit.ogg");
444   sfMusic* music = sfMusic_createFromFile("res/music.ogg");

446   //Imagenes de los patos
      sfTexture* tP1 = sfTexture_createFromFile("res/pato1.png", NULL);
448   sfTexture* tP2 = sfTexture_createFromFile("res/pato2.png", NULL);
      sfTexture* tP3 = sfTexture_createFromFile("res/pato3.png", NULL);
450
      sfTexture* tPM1 = sfTexture_createFromFile("res/patoM1.png", NULL);
452   sfTexture* tPM2 = sfTexture_createFromFile("res/patoM2.png", NULL);
      sfTexture* tPM3 = sfTexture_createFromFile("res/patoM3.png", NULL);
454


456   for(int Nivel = 1; Nivel < 6; Nivel++){
          if(sigNivel){
458       int aciertos = 0;
          sfMusic_play(music);
460
          //Tiempo de espera(2 segundos)
462       //Se indica el nivel en el que se encuentra
          time_t timeT = time(NULL);
464       while((time(NULL) - timeT) <2){
              if(sfRenderWindow_pollEvent(window, &event)){
466           if (event.type == sfEvtClosed) {
                  sfRenderWindow_close(window);
468           }
              }
470
          //Se crea el texto "Level x"
472       sfFont *style = sfFont_createFromFile("res/pd.ttf");
          sfText *textLevel = sfText_create();
474       sfText_setFont(textLevel, style);
          
```

```
476     char *textN = itoa(Nivel);
478     sfText_setString(textLevel, textN);
480     free(textN);
482     sfText_setCharacterSize(textLevel, 100);
484     sfVector2f vL;
486     vL = sfText_getPosition(textLevel);
488     vL.x = vL.x + 630; vL.y = vL.y + 150;
490     sfText_setPosition(textLevel, vL);
492     sfText *textOL = sfText_create();
494     sfText_setFont(textOL, style);
496     sfText_setString(textOL, "Level");
498     sfText_setCharacterSize(textOL, 100);
500     sfVector2f vOL;
502     vOL = sfText_getPosition(textOL);
504     vOL.x = vOL.x + 350; vOL.y = vOL.y + 150;
506     sfText_setPosition(textOL, vOL);
508     //Se carga la mira
510     sfVector2f posMira;
512     posMira.x = sfMouse_getPosition(window).x;
514     posMira.y = sfMouse_getPosition(window).y;
516     sfSprite_setPosition(Mira, posMira);
518     //Se imprime la imagen
520     sfRenderWindow_clear(window, sfBlack);
522     sfRenderWindow_drawSprite(window, bg, NULL);
524     sfRenderWindow_drawText(window, textLevel, NULL);
526     sfRenderWindow_drawText(window, textOL, NULL);
528     sfRenderWindow_drawSprite(window, Mira, NULL);
530     sfRenderWindow_display(window);
532     }
534     //Se empiezan los Rounds
536     for(int i = 0; i < 5; i++){
538     //Tiempo de espera (2 segundos)
540     time_t time1 = time(NULL);
542     while(time(NULL) - time1) < 2){
544     if(sfRenderWindow_pollEvent(window, &event)){
546     if (event.type == sfEvtClosed) {
548     sfRenderWindow_close(window);
550     }
552     }
554     //Se crea el texto "Round x"
556     sfFont *style = sfFont_createFromFile("res/pd.ttf");
558     sfText *text4 = sfText_create();
560     sfText_setFont(text4, style);
562     char *textRound = itoa(i + 1);
564     sfText_setString(text4, textRound);
566     free(textRound);
568     sfText_setCharacterSize(text4, 100);
570     sfVector2f v4;
572     v4 = sfText_getPosition(text4);
574     v4.x = v4.x + 630; v4.y = v4.y + 150;
576     sfText_setPosition(text4, v4);
578     sfText *text5 = sfText_create();
580     sfText_setFont(text5, style);
582     sfText_setString(text5, "Round");
584     sfText_setCharacterSize(text5, 100);
586     sfVector2f v5;
588     v5 = sfText_getPosition(text5);
590     v5.x = v5.x + 350; v5.y = v5.y + 150;
592     sfText_setPosition(text5, v5);
594     //Se carga la mira
596     sfVector2f posMira;
598     posMira.x = sfMouse_getPosition(window).x;
600     posMira.y = sfMouse_getPosition(window).y;
602     sfSprite_setPosition(Mira, posMira);
604     //Se imprime la imagen
606     sfRenderWindow_clear(window, sfBlack);
608     sfRenderWindow_drawSprite(window, bg, NULL);
```

```
    sfRenderWindow_drawText(window, text4, NULL);
    sfRenderWindow_drawText(window, text5, NULL);
    sfRenderWindow_drawSprite(window, Mira, NULL);
    sfRenderWindow_display(window);
    }

    //Se carga el color del pato aleatoriamente
    sfSprite* pato1 = patoAleatorio(tP1,tP2, tP3);
    sfSprite* pato2 = patoAleatorio(tP1,tP2, tP3);


    sfVector2f posInicialPato1 = {0,100};
    sfVector2f posInicialPato2 = {0,300};
    sfSprite_setPosition(pato1, posInicialPato1);
    sfSprite_setPosition(pato2, posInicialPato2);

    //variables para el pato muerto
    bool p1=false;
    bool p2=false;

    //Se mueve el pato (5 segundos)
    time1 = time(NULL);
    while (time(NULL) - time1) < 5){

        if(sfRenderWindow_pollEvent(window, &event)){
        if (event.type == sfEvtClosed) {
            sfRenderWindow_close(window);
        }
        }

        sfVector2f posPato1 = sfSprite_getPosition(pato1);
        sfVector2f posPato2 = sfSprite_getPosition(pato2);

        //movimiento del pato
        if(p1==true) {
        posPato1.x = posPato1.x;
        posPato1.y = posPato1.y + 2;
        }
        if(p1 == false){
            posPato1.x = posPato1.x + (drand48()* 15 + 1 - 7);
        posPato1.y = posPato1.y + (rand() %10 + 1 - 5);
        }

        if(p2==true) {
        posPato2.x = posPato2.x;
        posPato2.y = posPato2.y + 2;
        }
        if(p2 == false){
            posPato2.x = posPato2.x + (drand48()* 15 + 1 - 7);
        posPato2.y = (rand() %10 + 1 - 5);
        }

        //Para que no se salga de la pantalla
        if(posPato1.x > sfRenderWindow_getSize(window).x && p1==false) {
            posPato1.x = 0;
        }
        if(posPato1.y > (sfRenderWindow_getSize(window).y - 250) && p1==false) {
            posPato1.y = 0;
        }

        if(posPato2.x > sfRenderWindow_getSize(window).x && p2==false) {
            posPato2.x = 0;
        }
        if(posPato2.y > (sfRenderWindow_getSize(window).y - 250) && p2==false) {
            posPato2.y = 0;
        }

        //Se carga la mira
        sfVector2f posMira;
        posMira.x = sfMouse_getPosition(window).x;
        posMira.y = sfMouse_getPosition(window).y;
        sfSprite_setPosition(Mira, posMira);

        //Si se presiona el mouse
        if (event.type == sfEvtMouseButtonPressed) {
            sfMusic_play(tiro);

            //Se compara la posicion de la mira con la del pato
            if((posMira.x+25)>posPato1.x  && (posPato1.x + 76)>(posMira.x+25)){
                
```

```
            if((posMira.y+25)>posPato1.y__&&_(posPato1.y_+_62)_>(posMira.y+25))
                {
636 ____
_____sfSprite*_patoM1_=_patomuerto(pato1,_tPM1,tPM2,tPM3,_tP1,_tP2,tP3);
638 _____sfMusic_play(hit);__
_____________p1=true;
640 _____________pato1_=_patoM1;
_____}}
642 ____
____if((posMira.x+25)>posPato2.x__&&_(posPato2.x_+_76)>(posMira.x+25)){_
644 ____________if((posMira.y+25)>posPato2.y__&&_(posPato2.y_+_62)_>(posMira.y+25))
                {
_____sfSprite*_patoM2_=_patomuerto(pato2,_tPM1,tPM2,tPM3,_tP1,_tP2,tP3);
646 _____sfMusic_play(hit);__
_____________p2=true;
648 _____________pato2_=_patoM2;
_____}}
650 ____________}
____
652 _____//Se_crea_el_texto_"Score:_x"
_____sfFont_*style_=_sfFont_createFromFile("res/pd.ttf");
654 _____sfText_*_text6_=_sfText_create();
_____sfText_setFont(text6,_style);
656 ____
_____char_*__textScore_=_itoa_(total);
658 __
_____sfText_setString(text6,_textScore);
660 ____
_____free(textScore);
662 ____
_____sfText_setCharacterSize(text6,_20);
664 _____sfVector2f__v6;
_____v6=_sfText_getPosition(text6);
666 _____v6.x=v6.x+870;_v6.y=v6.y_+550;
_____sfText_setPosition(text6,_v6);
668 ____
_____sfText_*_text7_=_sfText_create();
670 _____sfText_setFont(text7,_style);
_____sfText_setString(text7,_"Score:_");
672 _____sfText_setCharacterSize(text7,_20);
_____sfVector2f__v7;
674 _____v7=_sfText_getPosition(text7);
_____v7.x=v7.x+800;_v7.y=v7.y+550;
676 _____sfText_setPosition(text7,_v7);
____
678 _____//Se_crea_el_texto_"Hits:_x"
_____sfText_*_text8_=_sfText_create();
680 _____sfText_setFont(text8,_style);
____
682 _____char_*__textHits_=_itoa_(aciertos);
__
684 _____sfText_setString(text8,_textHits);
____
686 _____free(textHits);
____
688 _____sfText_setCharacterSize(text8,_20);
_____sfVector2f__v8;
690 _____v8=_sfText_getPosition(text8);
_____v8.x=v8.x+870;_v8.y=v8.y_+520;
692 _____sfText_setPosition(text8,_v8);
____
694 _____sfText_*_text9_=_sfText_create();
_____sfText_setFont(text9,_style);
696 _____sfText_setString(text9,_"Hits:_");
_____sfText_setCharacterSize(text9,_20);
698 _____sfVector2f__v9;
_____v9=_sfText_getPosition(text9);
700 _____v9.x=v9.x+800;_v9.y=v9.y+520;
_____sfText_setPosition(text9,_v9);____

702
_____//_Se_imprime_la_imagen
704 _____sfSprite_setPosition(pato1,_posPato1);
_____sfSprite_setPosition(pato2,_posPato2);
706 _____sfRenderWindow_clear(window,_sfBlack);
_____sfRenderWindow_drawSprite(window,_bg,_NULL);
708 _____sfRenderWindow_drawSprite(window,_pato1,_NULL);
_____sfRenderWindow_drawSprite(window,_pato2,_NULL);
710 _____sfRenderWindow_drawSprite(window,_Mira,_NULL);
_____sfRenderWindow_drawText(window,_text6,_NULL);
712 _____sfRenderWindow_drawText(window,_text7,_NULL);
```

```
        sfRenderWindow_drawText(window, text8, NULL);
        sfRenderWindow_drawText(window, text9, NULL);
        sfRenderWindow_display(window);
    }
    if(p1){
        aciertos++;
    }

    if(p2){
        aciertos++;
    }
    printf("aciertos: %d\n",aciertos);

    int puntuacion1 = score(pato1, tPM1, tPM2, tPM3, p1, Nivel);
    int puntuacion2 = score(pato2, tPM1, tPM2, tPM3, p2, Nivel);
    total= total + puntuacion1 + puntuacion2;
    printf("Score1: %d\nScore2: %d\n",puntuacion1, puntuacion2);
    printf("Total: %d\n",total);

  }

  //Si se cumplieron los aciertos, se pasa al siguiente nivel
  int Necesarios =aciertosN(Nivel);
  if(Necesarios >= aciertos ){
    sigNivel = false;}

  //Bonus de perfect
  if(aciertos == 10){
    total = total + bonus(Nivel);}

  }
  else{

    sfMusic* GameOver = sfMusic_createFromFile("res/gameover.ogg");
    sfMusic_play(GameOver);

    //Tiempo de espera(5 segundos)
    time_t timeT = time(NULL);
    while((time(NULL) - timeT) < 5){
      if(sfRenderWindow_pollEvent(window, &event)){
        if (event.type == sfEvtClosed) {
          sfRenderWindow_close(window);
        }
      }
    }

    //Se crea el texto "Game Over"
    sfFont *style = sfFont_createFromFile("res/pd.ttf");
    sfText *textOver = sfText_create();
    sfText_setFont(textOver, style);
    sfText_setString(textOver, "Game Over");
    sfText_setCharacterSize(textOver, 100);
    sfVector2f vO;
    vO= sfText_getPosition(textOver);
    vO.x=vO.x+300; vO.y=vO.y+150;
    sfText_setPosition(textOver, vO);

    //Se imprime la imagen
    sfRenderWindow_clear(window, sfBlack);
    sfRenderWindow_drawSprite(window, bg, NULL);
    sfRenderWindow_drawText(window, textOver, NULL);
    sfRenderWindow_display(window);
    }
    return;
  }
}
return;
}
```

funciones.c

```
#ifndef FUNCIONES_H
#define FUNCIONES_H

#include <SFML/Audio.h>
#include <SFML/Graphics.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include<time.h>
#include <stdbool.h>
```

```
11
char *itoa(int i);
13 sfSprite *patoAleatorio(sfTexture *tP1,sfTexture *tP2, sfTexture *tP3);
int score(sfSprite *pato, sfTexture *tPM1,sfTexture *tPM2, sfTexture *tPM3,
    bool p1, int nivel);
15 int aciertosN(int nivel);
int bonus(int nivel);
17 sfSprite *patomuerto(sfSprite *pato, sfTexture *tPM1,sfTexture *tPM2,sfTexture*
    tPM3, sfTexture *tP1,sfTexture *tP2,sfTexture *tP3);
void GameA(sfRenderWindow *window, sfSprite *bg, sfSprite *Mira, sfMusic *
    tiro);
19 void GameB(sfRenderWindow *window, sfSprite *bg, sfSprite *Mira, sfMusic *
    tiro);

21 #endif
```

funciones.h