INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Dependencia de adscripción

Departamento de Electrónica, Sistemas e Informática

Nombre de la carrera

Ingeniería en Sistemas Computacionales

Título del trabajo

Proyecto integrador “Laberinto”

PRESENTA

Torrentera Arróniz José Luis

Sandoval López Lucio Antonio

Cabrera Cortés Abraham

Profesor: Luis Gatica

Tlaquepaque, Jalisco, 2 de diciembre de 2016.

Introducción

Durante el transcurso y elaboración del proyecto, cada uno de los integrantes del equipo debió aportar la totalidad de sus conocimientos para lograr la realización del mismo, se desarrollaron habilidades de pensamiento lógico, aprendizaje autónomo, investigación y aplicación de conocimientos y herramientas nunca antes utilizadas, aunque si, con relación al contenido temático visto durante el semestre en la asignatura. Se pretendía que cómo equipo fuera notoria la implementación de estructuras selectivas y de repetición propias de lenguaje de programación C, además del uso constante de funciones, manejo de archivos y arreglos dimensionales para finalmente llevar todo esto a la “pantalla” con ayuda de la biblioteca propia de C llamada Allegro.

Una vez terminado el proyecto, es decir, el archivo ejecutable con interfaz gráfica, el usuario podrá poner a prueba su mente y visión mediante la resolución de tres laberintos distintos, creados por nosotros mismos y con un nivel de dificultad mayor de acuerdo al laberinto. El usuario podrá divertirse y jugarlo cada uno de los tres niveles las veces que quiera, con el reto de mejorar su tiempo para solucionarlos, ya que cuenta con un cronómetro en tiempo real que marca cuando tarda en cada uno de los niveles.

Código Fuente

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <allegro5/allegro.h>
#include <allegro5/allegro_image.h>
#include <allegro5/allegro_primitives.h>
#include <allegro5/allegro_font.h>
#include <allegro5/allegro_ttf.h>


const float FPS = 60;
const int SCREEN_W = 640;
const int SCREEN_H = 640;
int BOUNCER_SIZE = 29;
enum MYKEYS {
	KEY_UP, KEY_DOWN, KEY_LEFT, KEY_RIGHT
};

typedef struct line{
	float coorX1;
	float coorX2;
	float coorY1;
	float coorY2;
}Line;
enum level{
	LVL1, LVL2, LVL3
};

int Limits_x(float bouncerCoor[4][2],float movement, bool dflag,Line
*limits, int nLines);
void ReadLines(Line * limits, int n, int begin, FILE *laberynth, float
size);


int main(int argc, char **argv)
{
	FILE *horizontal;
	FILE *vert;
	ALLEGRO_DISPLAY *display = NULL;
	ALLEGRO_EVENT_QUEUE *event_queue = NULL;
	ALLEGRO_TIMER *timer = NULL;
	ALLEGRO_TIMER *timer2 = NULL;
	ALLEGRO_BITMAP *bouncer = NULL;
	ALLEGRO_BITMAP  *image1 = NULL;
	ALLEGRO_BITMAP *image2 = NULL;
	ALLEGRO_BITMAP *image3 = NULL;
	ALLEGRO_FONT *font = NULL;
	ALLEGRO_FONT *font2 = NULL;
	float bouncerCoor[4][2];
	bouncerCoor[0][0] = 40;
	bouncerCoor[0][1] = 40;

	bool key[4] = { false, false, false, false };
	bool redraw = true;
```

```
	bool doexit = false;
	bool win = false;
	bool game = false;
	bool wait1 = false;
	bool wait2 = false;
	bool menu = true;
	int lvl = LVL1;

	float posx,posy;
	int i;
	int nLinesHz, nLinesVc;
	float size;
	float vel = 4.0;
	float time = 0;
	float time2;
	char tiempo[5];
	Line Limits[200];

	if(!al_init()) {
		fprintf(stderr, "failed to initialize allegro!\n");
		return -1;
	}

	if(!al_install_keyboard()) {
		fprintf(stderr, "failed to initialize the keyboard!\n");
		return -1;
	}
	if(!al_install_mouse()) {
		fprintf(stderr, "failed to initialize the mouse!\n");
		return -1;
	}

	timer = al_create_timer(1.0 / FPS);
	if(!timer) {
		fprintf(stderr, "failed to create timer!\n");
		return -1;
	}

	timer2 = al_create_timer(1.0);
	if(!timer2) {
		fprintf(stderr, "failed to create timer!\n");
		al_destroy_timer(timer);
		return -1;
	}

	display = al_create_display(SCREEN_W, SCREEN_H);
	if(!display) {
		fprintf(stderr, "failed to create display!\n");
		al_destroy_timer(timer);
		al_destroy_timer(timer2);
		return -1;
	}
	al_init_image_addon();
	bouncer =
al_load_bitmap("C:/Users/joseluis/ProyectosC/Proyecto_Fundamentos/Release
/cabecita.png");
	if(!bouncer) {
```

```
            fprintf(stderr, "failed to create bouncer bitmap!\n");
            al_destroy_display(display);
            al_destroy_timer(timer);
            al_destroy_timer(timer2);
            return -1;
      }


      image1 =
al_load_bitmap("C:/Users/joseluis/ProyectosC/Proyecto_Fundamentos/Release
/image_1.png");
      if(!image1)
      {
            fprintf(stderr, "failed to load image!\n");
            al_destroy_display(display);
            al_destroy_timer(timer);
            al_destroy_timer(timer2);
            al_destroy_bitmap(bouncer);
            return -1;
      }

      image2 =
al_load_bitmap("C:/Users/joseluis/ProyectosC/Proyecto_Fundamentos/Release
/image_2.jpeg");
      if(!image2)
      {
            fprintf(stderr, "failed to load image!\n");
            al_destroy_display(display);
            al_destroy_timer(timer);
            al_destroy_timer(timer2);
            al_destroy_bitmap(bouncer);
            return -1;
      }

      image3 =
al_load_bitmap("C:/Users/joseluis/ProyectosC/Proyecto_Fundamentos/Release
/image_3.jpeg");
      if(!image3)
      {
            fprintf(stderr, "failed to load image!\n");
            al_destroy_display(display);
            al_destroy_timer(timer);
            al_destroy_timer(timer2);
            al_destroy_bitmap(bouncer);
            return -1;
      }

      al_init_font_addon();
      al_init_ttf_addon();
      ALLEGRO_PATH *path = al_get_standard_path(ALLEGRO_RESOURCES_PATH);
      al_set_path_filename(path, "CHILLER.ttf");
      font = al_load_ttf_font(al_path_cstr(path, '/'),20,0 );

      if (!font){
            fprintf(stderr, "Could not load 'pirulen.ttf'.\n");
            return -1;
      }
```

```
	font2 = al_load_ttf_font(al_path_cstr(path, '/'),72,0 );

	if (!font){
		fprintf(stderr, "Could not load 'pirulen.ttf'.\n");
		return -1;
	}


	al_init_primitives_addon();

	al_set_target_bitmap(al_get_backbuffer(display));

	event_queue = al_create_event_queue();
	if(!event_queue) {
		fprintf(stderr, "failed to create event_queue!\n");
		al_destroy_bitmap(bouncer);
		al_destroy_display(display);
		al_destroy_timer(timer);
		al_destroy_timer(timer2);
		return -1;
	}

	al_register_event_source(event_queue,
al_get_display_event_source(display));

	al_register_event_source(event_queue,
al_get_timer_event_source(timer));

	al_register_event_source(event_queue,
al_get_timer_event_source(timer2));

	al_register_event_source(event_queue,
al_get_keyboard_event_source());

	al_register_event_source(event_queue, al_get_mouse_event_source());

	al_clear_to_color(al_map_rgb(0,0,0));

	al_flip_display();

	al_start_timer(timer);


	ALLEGRO_EVENT ev;

	while(!doexit){
		al_wait_for_event(event_queue, &ev);
		if(menu ){
			if(ev.type == ALLEGRO_EVENT_DISPLAY_CLOSE)
			{
				doexit = true;
			}

			else if(ev.type == ALLEGRO_EVENT_MOUSE_BUTTON_DOWN)
			{
				posx = ev.mouse.x;
```

```
                        posy = ev.mouse.y;

                        if ((posx<=445 && posx>187) && (posy>335 &&
posy<433))
                        {
                              game = true;
                              menu = false;
                        }

                        else if ((posx<=443 && posx>189) && (posy>463 &&
posy<558))
                        {
                              doexit = true;
                        }
                  }

                  al_draw_bitmap(image1, 0, 0, 0);

                  al_flip_display();
            }
            if(wait1 ){
                  if(ev.type == ALLEGRO_EVENT_DISPLAY_CLOSE)
                  {
                        doexit = true;
                  }

                  else if(ev.type == ALLEGRO_EVENT_MOUSE_BUTTON_DOWN)
                  {
                        posx = ev.mouse.x;
                        posy = ev.mouse.y;

                        if ((posx<=166 && posx>42) && (posy>339 &&
posy<377))
                        {
                              menu = true;
                              wait1 = false;
                        }
                        else if ((posx<=613 && posx>478) && (posy>275 &&
posy<406))
                        {
                              game = true;
                              wait1 = false;
                        }
                  }

                  al_draw_bitmap(image2, 0, 0, 0);
                  itoa((int)time2,tiempo,10);
                  al_draw_text(font2, al_map_rgb(255,255,255),210, 111,
0, tiempo);

                  al_flip_display();
            }
            if(wait2 ){
                  if(ev.type == ALLEGRO_EVENT_DISPLAY_CLOSE)
                  {
                        doexit = true;
                  }
```

```
                  if(ev.type == ALLEGRO_EVENT_MOUSE_BUTTON_DOWN)
                  {
                        posx = ev.mouse.x;
                        posy = ev.mouse.y;
                        if ((posx<=166 && posx>42) && (posy>339 &&
posy<377))
                        {
                              menu = true;
                              wait2 = false;
                        }
                        else if ((posx<=613 && posx>497) && (posy>326 &&
posy<370))
                        {
                              doexit = true;
                        }
                  }

                  al_draw_bitmap(image3, 0, 0, 0);
                  itoa((int)time2,tiempo,10);
                  al_draw_text(font2, al_map_rgb(255,255,255),210, 111,
0, tiempo);
                  al_flip_display();
            }
            if(game )
            {
                  if(lvl == LVL1){
                        size = 60;
                        horizontal =
fopen("C:/Users/joseluis/ProyectosC/Proyecto_Fundamentos/Release/Laberint
o_horiz.txt","r");
                        if (horizontal == NULL){
                              printf("ERROR FILE");
                              return -1;
                        }
                        vert =
fopen("C:/Users/joseluis/ProyectosC/Proyecto_Fundamentos/Release/Laberint
o_vert.txt","r");
                        if (vert == NULL){
                              printf("ERROR FILE");
                              return -1;
                        }

                        nLinesHz = 24;
                        nLinesVc = 30;

                  } else if(lvl == LVL2){

                        size = 40;
                        horizontal =
fopen("C:/Users/joseluis/ProyectosC/Proyecto_Fundamentos/Release/Lab_lvl2
_horiz.txt","r");
                        if (horizontal == NULL){
                              printf("ERROR FILE");
                              return -1;
                        }
```

```
                        vert =
fopen("C:/Users/joseluis/ProyectosC/Proyecto_Fundamentos/Release/Lab_lvl2
_vert.txt","r");
                        if (vert == NULL){
                              printf("ERROR FILE");
                              return -1;
                        }

                        nLinesHz = 56;
                        nLinesVc = 57;


                  }else if(lvl == LVL3){
                        size = 33;
                        horizontal =
fopen("C:/Users/joseluis/ProyectosC/Proyecto_Fundamentos/Release/Laberint
o3_horizontal.txt","r");
                        if (horizontal == NULL){
                              printf("ERROR FILE");
                              return -1;
                        }
                        vert =
fopen("C:/Users/joseluis/ProyectosC/Proyecto_Fundamentos/Release/Laberint
o3_vertical.txt","r");
                        if (vert == NULL){
                              printf("ERROR FILE");
                              return -1;
                        }

                        nLinesHz = 78;
                        nLinesVc = 75;

                  }

                  ReadLines(Limits,nLinesHz,0,horizontal,size);
                  ReadLines(Limits,nLinesHz+nLinesVc,nLinesHz,vert,size);

                  fclose(horizontal);
                  fclose(vert);



                  if(ev.type == ALLEGRO_EVENT_TIMER) {
                        time++;

                        if(key[KEY_UP] && bouncerCoor[0][1] >= vel) {
                              if(Limits_x(bouncerCoor,-
vel,0,Limits,nLinesHz+nLinesVc))
                                    bouncerCoor[0][1] -= vel;
                        }

                        if(key[KEY_DOWN] && bouncerCoor[0][1] <= SCREEN_H
- BOUNCER_SIZE - vel) {

      if(Limits_x(bouncerCoor,vel,0,Limits,nLinesHz+nLinesVc))
                                    bouncerCoor[0][1] += vel;
                        }
```

```
                        if(key[KEY_LEFT] && bouncerCoor[0][0] >= vel ) {
                              if(Limits_x(bouncerCoor,-
vel,1,Limits,nLinesHz+nLinesVc))
                                    bouncerCoor[0][0] -= vel;
                        }

                        if(key[KEY_RIGHT] && bouncerCoor[0][0] <= SCREEN_W
- BOUNCER_SIZE - vel) {

      if(Limits_x(bouncerCoor,vel,1,Limits,nLinesHz+nLinesVc))
                                    bouncerCoor[0][0] += vel;
                        }
                        redraw = true;
                  }
                  else if(ev.type == ALLEGRO_EVENT_DISPLAY_CLOSE) {
                        break;
                  }
                  else if(ev.type == ALLEGRO_EVENT_KEY_DOWN) {
                        switch(ev.keyboard.keycode) {
                        case ALLEGRO_KEY_UP:
                              key[KEY_UP] = true;
                              break;

                        case ALLEGRO_KEY_DOWN:
                              key[KEY_DOWN] = true;
                              break;

                        case ALLEGRO_KEY_LEFT:
                              key[KEY_LEFT] = true;
                              break;

                        case ALLEGRO_KEY_RIGHT:
                              key[KEY_RIGHT] = true;
                              break;
                        }
                  }
                  else if(ev.type == ALLEGRO_EVENT_KEY_UP) {
                        switch(ev.keyboard.keycode) {
                        case ALLEGRO_KEY_UP:
                              key[KEY_UP] = false;
                              break;

                        case ALLEGRO_KEY_DOWN:
                              key[KEY_DOWN] = false;
                              break;

                        case ALLEGRO_KEY_RIGHT:
                              key[KEY_RIGHT] = false;
                              break;

                        case ALLEGRO_KEY_LEFT:
                              key[KEY_LEFT] = false;
                              break;

                        case ALLEGRO_KEY_ESCAPE:
                              doexit = true;
```

```
                        break;
                    }
                }

                if(redraw && al_is_event_queue_empty(event_queue)) {
                    redraw = false;

                    al_clear_to_color(al_map_rgb(0,0,0));

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

    al_draw_filled_rectangle(Limits[i].coorX1,Limits[i].coorY1,Limits[i
].coorX2+10,Limits[i].coorY2+10, al_map_rgb(255, 255, 255));
                    for(i = nLinesHz; i < nLinesHz+nLinesVc; i++)

    al_draw_filled_rectangle(Limits[i].coorX1,Limits[i].coorY1,Limits[i
].coorX2+10,Limits[i].coorY2, al_map_rgb(255, 255, 255));
                    al_draw_bitmap(bouncer, bouncerCoor[0][0],
bouncerCoor[0][1], 0);

                    itoa((int)time/60,tiempo,10);
                    al_draw_text(font, al_map_rgb(255,255,255),10, 0,
0, tiempo);
                    al_flip_display();

                    if(bouncerCoor[0][1] > 600){
                        win = true;
                        game = false;
                        time2 = time/60;
                        time = 0;
                        key[KEY_DOWN] = false;
                        key[KEY_UP] = false;
                        key[KEY_LEFT] = false;
                        key[KEY_RIGHT] = false;
                    }
                }

                switch(lvl){
                case LVL1:
                    if(win){
                        lvl = LVL2;
                        wait1=true;
                        bouncerCoor[0][0] = 35;
                        bouncerCoor[0][1] = 28;
                        BOUNCER_SIZE -= 4;
                    }
                    break;
                case LVL2:

                    if(win){
                        lvl = LVL3;
                        wait1 = true;
                        bouncerCoor[0][0] = 35;
                        bouncerCoor[0][1] = 28;
                        BOUNCER_SIZE -= 8;
                    }
                    break;
```

```
                case LVL3:
                        if(win){
                                wait2 = true;
                        }
                }
                win = false;
        }
    }


    al_destroy_bitmap(bouncer);
    al_destroy_timer(timer);
    al_destroy_timer(timer2);
    al_destroy_display(display);
    al_destroy_event_queue(event_queue);

    return 0;
}

int Limits_x(float bouncerCoor[4][2], float movement, bool dflag, Line
*limits, int nLines){

    int i, mx = 0, my = 0, j;

    if(dflag){
        mx = movement;
    }else{
        my = movement;
    }

    bouncerCoor[1][0] = bouncerCoor[0][0] + BOUNCER_SIZE;
    bouncerCoor[1][1] = bouncerCoor[0][1];
    bouncerCoor[2][0] = bouncerCoor[0][0];
    bouncerCoor[2][1] = bouncerCoor[0][1] + BOUNCER_SIZE;
    bouncerCoor[3][0] = bouncerCoor[0][0] + BOUNCER_SIZE;
    bouncerCoor[3][1] = bouncerCoor[0][1] + BOUNCER_SIZE;

    for(j = 0; j < nLines; j++)
        for(i = 0; i < 4; i++)
            if(bouncerCoor[i][0] + mx >= limits[j].coorX1 &&
                    bouncerCoor[i][0] + mx <= limits[j].coorX2 +
10.0 &&
                    bouncerCoor[i][1] + my >= limits[j].coorY1
&&
                    bouncerCoor[i][1] + my <= limits[j].coorY2 +
10.0
            ){
                return 0;
            }

    return 1;
}

void ReadLines(Line * limits, int n,int begin,FILE *laberynth, float
size){
    int i;
```

```
      for(i = begin;i < n;i++){
            fscanf(laberynth,"%f %f %f
%f",&limits[i].coorX1,&limits[i].coorX2,&limits[i].coorY1,&limits[i].coor
Y2);
            limits[i].coorX1 = (limits[i].coorX1 * size)+20.0;
            limits[i].coorX2 = (limits[i].coorX2 * size)+20.0;
            limits[i].coorY1 = (limits[i].coorY1 * size)+20.0;
            limits[i].coorY2 = (limits[i].coorY2 * size)+20.0;
      }
}
```

Conclusiones

La elaboración de archivos ejecutables con interfaz gráfica sin duda fueron un gran reto para nosotros como estudiantes, ya que para lograrlo se debía de hacer uso de todas las herramientas que conocíamos de C, e incluso las que no. Fue interesante y satisfactorio haber aprendido a manejar gran parte de las herramientas que nos ofrece la librería de Allegro para crear “juegos” con gráficos, interacción mediante el uso de las teclas y el mouse, además del manejo de imágenes “bitmaps”.

Ahora contamos con una perspectiva del trabajo que conllevan este tipo de proyectos, donde pudimos ver una panorámica de lo que se viene en semestres posteriores. Ahora sabemos el tiempo que se requiere y el trabajo en equipo que debe es primordial en este tipo de proyectos, ir a las asesorías fue una gran herramienta que nuestro profesor nos facilitó para lograr comprender cómo es que se podía hacer nuestro proyecto (laberinto), solucionar las dudas que tenías al respecto e incluso la corrección de problemas específicos de nuestro código.

Este tipo de proyectos independientemente de si son muy cansados, difíciles y tediosos, son de gran beneficio para nosotros como estudiantes pues con base a ellos, podemos darnos cuentas que tanto aprendimos y que temas nos hace falta reforzar antes de ingresar al próximo semestre. Practicar lo aprendido es una manera de lograr ser los mejores, además de saber implementar la programación para la vida cotidiana que al fin de cuentas eso es lo que hace un Ingeniero en Sistemas Computacionales.