Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

`IE-0117` Programación bajo plataformas abiertas

Reporte de proyecto: Space Invaders

Belinda Brown Ramírez - `B61254`
Alexander Calderón Torres - `B61325`
Carolina Urrutia Núñez - `B77833`

II ciclo
15 de diciembre de 2018

Tabla de contenidos

---

1. Reseña del programa

Se desarrolló el juego de arcade Space Invaders, el cual fue diseñado por Toshihiro Nishikado, de Taito Corporation y distribuido por Bally Midway en 1978. Space Invaders consiste principalmente en defender al planeta de una invasión de alienígenas, esto mediante una nave controlada por el usuario que dispara contra las naves invasoras, por cada nave derribada se obtiene cierto puntaje. Este fue un juego que cautivó a las personas, pues se volvió uno de los primeros en implementar la función de guardar y mantener en memoria los puntajes obtenidos más altos, generando así un factor de competitividad por obtener el primer puesto y el mejor puntaje, lo que creó un aumento de interés y adicción hacia el juego. Además, Space Invaders impulsó la industria de los videojuegos, ya que generó que tanto niños como adultos se quisieran involucrar en este mundo de entretenimiento y recreación. También fue uno de los primeros juegos en hacer referencia a los alienígenas, lo que lo hacia un juego aún más interesante y único.

2. Funcionamiento

El juego proporciona la funcionalidad básica que el juego de Space Invaders original ofrecía. Al abrir el juego, el usuario es recibido con la pantalla de inicio, de la cual se detallará proximamente.

El lenguaje de programación utilizado fue C, además, para la construcción de la interfaz gráfica se usaron las bibliotecas de SFML y Simple DirectMedia.

El programa está compuesto de una serie de archivos llamados: invaders.h , inicio.c , juego.c , main.c y scores.c. En cada uno de ellos se encuentra el código que hace posible el correcto funcionamiento del programa.

2.1. Pantalla de Inicio

En esta, se muestran los siguientes elementos:

- **Título del juego**
- **Información sobre los enemigos:**Se muestra el puntaje obtenido al eliminar a los distintos enemigos presentes en el juego.
- **Botón PLAY:** Es el botón con el que el jugador da inicio a una partida individual. Una vez presionado, se salta a la pantalla de juego.

2.2. Pantalla de juego

Esta es la pantalla en la que se da lugar a la partida y en la que el jugador pasa la mayor parte del tiempo. Mientras esta pantalla esté activa, el usuario puede hacer uso de las teclas esc o P, la primera para salir de la partida y la segunda para pausar esta. Sus elementos se detallan a continuación:

- **Puntajes:** En la parte superior de esta pantalla se encuentra un texto en el que se detalla el puntaje que ha sido obtenido hasta el momento por el jugador (Score <1 >), además del texto del puntaje más alto (Hi-Score).
- **Alienígenas:** Son los enemigos a derrotar. Estos personajes presentan repetitivamente un movimiento horizontal (todos en conjunto). Estas naves tienen también la capacidad de disparar, y cuando una de las balas logra impactar la nave del jugador, este pierde una vida o

cuando impactan las barreras, estas son dañadas. Al ser impactados por una bala del jugador, estos explotan, desaparecen y suman su respectivo puntaje al jugador.

- **Barreras protectoras:** Estas barreras funcionan como una ayuda para el jugador, al impedir que las balas enemigas hagan daño a su nave. Su vida se prolonga solo durante algunos impactos, por lo que después de un tiempo estas barreras desaparecen, dejando al descubierto y un más vulnerable al jugador.
- **Nave principal:** Es la nave controlada por el usuario y la herramienta que se posee para eliminar a los alienígenas. Se controla mediante las teclas de dirección del teclado, y presenta un movimiento solamente horizontal; además, utilizando la tecla de espacio, la nave puede disparar una bala a la vez, y tiene un límite de 350 balas.
- **Cantidad de vidas disponibles:** Se poseen tres vidas inicialmente, que se pierden al recibir impactos de las naves enemigas. Una vez que las tres se acaban, la partida llega a su fin.
- **Credit:** Elemento decorativo.

2.3. Pantalla de *Game Over*

Esta pantalla se muestra una vez que la partida ha acabado, y en ella se muestran algunas imágenes y texto decorativo, así como una lista de las cinco puntuaciones más altas que se han registrado en el juego. Mientras se muestra esta pantalla, basta presionar cualquier tecla para volver a la pantalla de inicio, y jugar una nueva partida si se desea.

3. Experimentos realizados

Se probó que el código no tiene errores de sintáxis por lo que se logró compilar, sucesivamente se hicieron las modificaciones necesarias para que ejecutara adecuadamente las instrucciones generales, como lo son: el movimiento lateral, el disparo y la recepción de misiles, conteo de vidas, señal de inicio de partida y finalización del mismo.

Con lo que respecta al sistema operativo en el que funciona, es ejecutable sobre cualquier plataforma que compile mediante gcc. El programa recibe las instrucciones realizadas por medio del teclado con el fin de volver más accesible el juego, ya que al correrlo en una computadora es indispensable el uso del teclado, y por este motivo no será necesario un mouse para jugarlo solo para seleccionar el botón de play.

Por otra parte, respecto a las variables de respuesta como posición del jugador, puntaje, posición del disparo, correcta identificación y asignación de puntaje respecto al impacto sobre los distintos alienígenas, fue esencial la creación y consumación de pruebas experimentales de forma que se validó mediante test unitarios. Es necesario descomponer las funciones del juego en comportamientos cuantificables, los cuales fueron medidos discretamente haciendo uso de un análogo de gtest.

4. Resultados obtenidos

El juego se logra compilar mediante el uso adecuado de un archivo Makefile. Todo el código se encuentra en gitlab asignado al principio del curso, se muestra en el siguiente link: *https://gitlab.com/UCR-EIE/IE-0117/II-2018/g2/tree/master/Proyecto%20final%20* .Después de obtener los archivos, es

necesario descomprimir el zip que se descarga. A partir de ahí, se debe posicionar dentro de la carpeta de space invaders generada y digitar *make* en la línea de comandos obteniendo como resultado:

```
belindabrown@LinuxMint:~/Desktop/spaceinvaders$ make
gcc -g -I./incl -D_GNU_SOURCE   -c -o src/main.o src/main.c
gcc -g -I./incl -D_GNU_SOURCE   -c -o src/inicio.o src/inicio.c
gcc -g -I./incl -D_GNU_SOURCE   -c -o src/juego.o src/juego.c
gcc -o bin/SpaceInvaders src/main.o src/inicio.o src/juego.o -lcsfml-graphics -lcsfml-audio
```

Figura 1. Demostración de make

De la misma manera hacemos uso del Makefile para poder ejecutar el programa, en la línea de comandos digitamos make run:

```
belindabrown@LinuxMint:~/Desktop/spaceinvaders$ make run
./bin/SpaceInvaders
```

Figura 2. Demostración de make run

Seguidamente se desplegará una pantalla con el programa listo para ser manipulado:

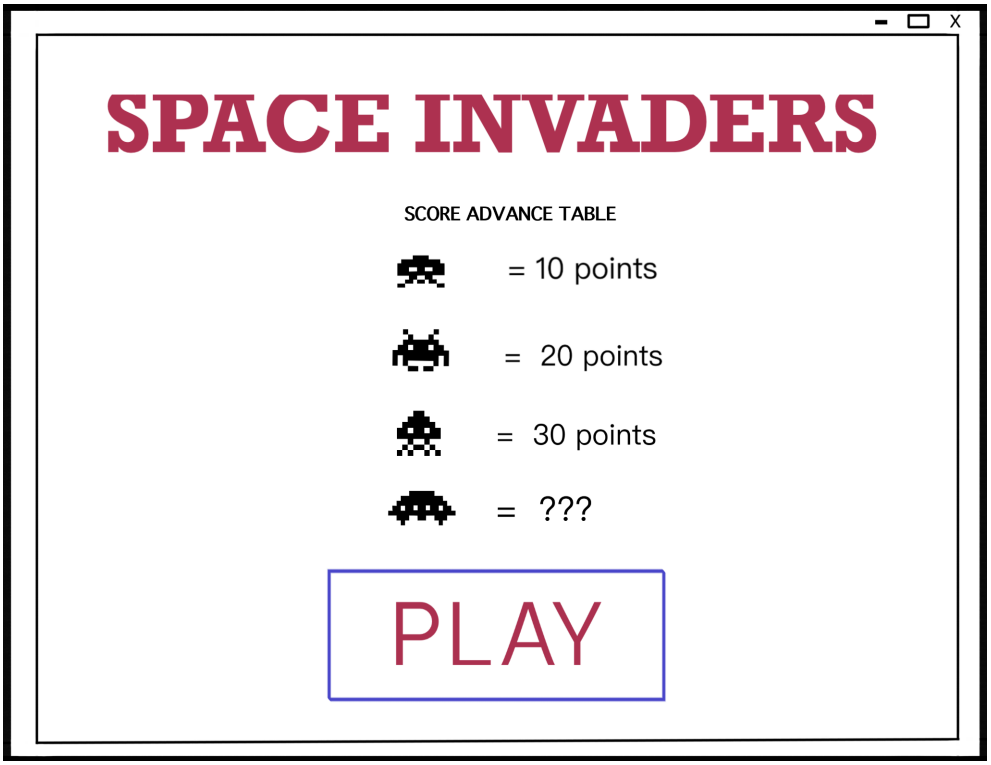

Figura 3. Pantalla de inicio desplegada

5. Conclusiones

El objetivo general de este proyecto era contribuir teórica y prácticamente al entendimiento y comprensión de la lógica de programación. Para lograr desarrollar el programa se hizo una división detallada y concreta de las necesidades que implicaba cada función del programa con el fin de cumplir la misión. Tomamos en cuenta una serie de bibliotecas que aportaban funciones importantes

para nosotros. Hicimos una deshilación con respecto a lo que es la composición del juego en su totalidad, partiendo de las pantallas y lo que se requería en cada una de ellas.

Además, el proyecto permitió hacer uso de herramientas como GitLab y herramientas de investigación. Como parte de los obstáculos del proyecto, resultó ser más complejo de lo esperado por lo que algunas de las funciones como multijugador y el despligue de los scores planteados en la propuesta inicial, no fueron posibles de realizar, sin embargo, se considera que el resto del programa funciona óptimamente.

Para concluir, para la realización de un proyecto de este calibre cualidades como la persistencia y el trabajo en equipo fueron necesarias.

Referencias

[1] Classic Gaming. (2018).*History of Space Invaders*. Recuperado de: http://www.classicgaming.cc/classics/space-invaders/history.php

[2] Cabrera, J. (2015).*A la vieja escuela, reseña Space Invaders*. Recuperado de: http://radiobuap.com/2015/02/a-la-vieja-escuela-resena-space-invaders/.

[3] GENBETA. (2013).*SFML,biblioteca para desarrollo de juegos*. Recuperado de: www.genbeta.com/desarrollo/sfml-2-biblioteca-para-el-desarrollo-de-videojuegos

[4] EcuRed. (2018).*Lenguaje de programación C*. Recuperado de: https://www.ecured.cu/Lenguaje_de_Programaci%C3%B3n_C.

[5] Make 8bit art (s.f). *Figuras*. Recuperado de: https://make8bitart.com/.

[6] Stack Overflow (2018). *Consultas varias*. Recuperado de: https://es.stackoverflow.com/.

6. Anexos

6.1. Código fuente:

main.c:

```
1
2
3 #include "invaders.h"
4
5 int main(){
6     sfEvent e;
7     enum pantallas pantalla_actual = INICIO;
8     enum pantallas siguiente_pantalla = INICIO;
9
10    // Cargar pantallas
11    cargar_inicio();
12    cargar_juego();
13
14    // Crear pantalla
15    sfVideoMode pp = {800, 600 , 32};
16    sfRenderWindow* w;
17    w = sfRenderWindow_create(pp, "SPACE INVADERS", sfClose, NULL);
18
19    dibujar_inicio(w);
20    while(sfRenderWindow_isOpen(w)){
21      switch(pantalla_actual) {
22        case JUEGO:
23          animacion_juego(w);
24          break;
25      }
26      while(sfRenderWindow_pollEvent(w, &e)){
27        if(e.type == sfEvtClosed){
28          printf("Cerrando\n");
29          sfRenderWindow_close(w);
30        } else {
31          switch(pantalla_actual) {
32            case INICIO:
33              siguiente_pantalla = eventos_inicio(w, &e);
34              break;
35            case JUEGO:
36              siguiente_pantalla = eventos_juego(w, &e);
37              break;
38            case RESULTADOS:
39              pantalla_actual = eventos_resultados(w, &e);
40              break;
41          }
42
43          if (pantalla_actual != siguiente_pantalla) {
44            sfRenderWindow_clear(w, sfWhite);
45            switch(siguiente_pantalla) {
46              case INICIO:
47                dibujar_inicio(w);
48                dibujar_inicio(w);
49                break;
50              case JUEGO:
51                reset_juego();
52                dibujar_juego(w);
53                dibujar_juego(w);
54                //dibujar_scores();
```

```
                    break;
                case RESULTADOS:
                    dibujar_scores(w);
                    break;
            }
            pantalla_actual = siguiente_pantalla;
          }
        }
      }
    }

    return 0;
}


```

juego.c:

```
#include "invaders.h"

int score = 0;
int vidas = 3;

sfTexture* naveText;
sfTexture* naveIzquierda;
sfTexture* naveDerecha;
sfTexture* naveCentro;
sfTexture* balaText;
sfSprite* balaSprite;
sfSprite* naveSprite;
sfText* primerscore;
sfText* segundoscore;
sfText* hiscore;
sfText* credit;
sfText* life;
sfText* gameover;
sfRectangleShape* c;
sfSprite* barrier1;
sfTexture* texture;
sfSprite* barrier2;
sfTexture* texture2;
sfSprite* barrier3;
sfTexture* texture3;
sfSprite* barrier4;
sfTexture* texture4;
sfText* paused;

sfSprite* enemy1Sprite;
sfTexture* enemy1Texture;
sfSprite* enemy2Sprite;
sfTexture* enemy2Texture;
sfSprite* enemy3Sprite;
sfTexture* enemy3Texture;
sfSprite* enemy4Sprite;
sfTexture* enemy4Texture;

//?
int naveSpriteX=380;
int naveSpriteY=480;

bool pausado=false;

struct barrera {
  int x;
  int y;
  int vida;
};
struct barrera b[4] = {
              [0].x = 120, [0].y = 430, [0].vida = 3,
              [1].x = 280, [1].y = 430, [1].vida = 3,
              [2].x = 440, [2].y = 430, [2].vida = 3,
              [3].x = 600, [3].y = 430, [3].vida = 3};

void pausar(){
```

```
128
129   pausado=!pausado;
130
131 }
132
133 int cargar_juego(void) {
134   //Sprites y texturas de la nave y balas
135
136   //texto SCORE <1>
137   primerscore = sfText_create();
138   sfText_setString(primerscore, "SCORE <1>");
139   sfText_setCharacterSize(primerscore, 30);
140   sfText_setPosition(primerscore, (sfVector2f){40,50});
141   sfText_setFont(primerscore, sfFont_createFromFile("resources/232MKRL_.TTF"));
142   sfText_setColor(primerscore, sfBlack);
143
144   //texto SCORE <2>
145   segundoscore = sfText_create();
146   sfText_setString(segundoscore, "SCORE <2>");
147   sfText_setCharacterSize(segundoscore, 30);
148   sfText_setPosition(segundoscore, (sfVector2f){550,50});
149   sfText_setFont(segundoscore, sfFont_createFromFile("resources/232MKRL_.TTF"));
150   sfText_setColor(segundoscore, sfBlack);
151
152   //texto HI-SCORE
153   hiscore = sfText_create();
154   sfText_setString(hiscore, "HI-SCORE");
155   sfText_setCharacterSize(hiscore, 40);
156   sfText_setPosition(hiscore, (sfVector2f){300,50});
157   sfText_setFont(hiscore, sfFont_createFromFile("resources/232MKRL_.TTF"));
158   sfText_setColor(hiscore, sfBlack);
159
160   //texto credit
161   credit = sfText_create();
162   sfText_setString(credit, "Credit 00");
163   sfText_setCharacterSize(credit, 20);
164   sfText_setPosition(credit, (sfVector2f){650,550});
165   sfText_setFont(credit, sfFont_createFromFile("resources/232MKRL_.TTF"));
166   sfText_setColor(credit, sfBlack);
167
168   life = sfText_create();
169   sfText_setString(life, "Lives 3");
170   sfText_setCharacterSize(life, 20);
171   sfText_setPosition(life, (sfVector2f){50,550});
172   sfText_setFont(life, sfFont_createFromFile("resources/232MKRL_.TTF"));
173   sfText_setColor(life, sfBlack);
174
175   //fix
176   gameover = sfText_create();
177   sfText_setString(gameover, "GAME OVER");
178   sfText_setCharacterSize(gameover, 60);
179   sfText_setPosition(gameover, (sfVector2f){250,330});
180   sfText_setFont(gameover, sfFont_createFromFile("resources/Athletic.TTF"));
181   sfText_setColor(gameover, sfRed);
182
183   //linea
184   c = sfRectangleShape_create();
185   sfRectangleShape_setFillColor(c, sfBlack);
186   sfRectangleShape_setPosition(c, (sfVector2f){ 50, 545 });
```

```
187  sfRectangleShape_setSize(c, (sfVector2f){ 710, 3 });
188
189  //Protecci n
190  texture = sfTexture_createFromFile("resources/barrier.png", NULL);
191  barrier1 = sfSprite_create();
192  sfSprite_setTexture(barrier1, texture, sfTrue);
193  sfSprite_setPosition(barrier1, (sfVector2f){120, 430});
194  sfSprite_setScale(barrier1, (sfVector2f){0.25,0.25});
195
196  texture2 = sfTexture_createFromFile("resources/barrier.png", NULL);
197  barrier2 = sfSprite_create();
198  sfSprite_setTexture(barrier2, texture2, sfTrue);
199  sfSprite_setPosition(barrier2, (sfVector2f){280, 430});
200  sfSprite_setScale(barrier2, (sfVector2f){0.25,0.25});
201
202  texture3 = sfTexture_createFromFile("resources/barrier.png", NULL);
203  barrier3 = sfSprite_create();
204  sfSprite_setTexture(barrier3, texture3, sfTrue);
205  sfSprite_setPosition(barrier3, (sfVector2f){440, 430});
206  sfSprite_setScale(barrier3, (sfVector2f){0.25,0.25});
207
208  texture4 = sfTexture_createFromFile("resources/barrier.png", NULL);
209  barrier4 = sfSprite_create();
210  sfSprite_setTexture(barrier4, texture4, sfTrue);
211  sfSprite_setPosition(barrier4, (sfVector2f){600, 430});
212  sfSprite_setScale(barrier4, (sfVector2f){0.25,0.25});
213
214  enemy1Texture = sfTexture_createFromFile("resources/enemy1.png", NULL);
215  enemy1Sprite = sfSprite_create();
216  sfSprite_setTexture(enemy1Sprite, enemy1Texture, sfTrue);
217  sfSprite_setScale(enemy1Sprite, (sfVector2f){0.10,0.10});
218
219  enemy2Texture = sfTexture_createFromFile("resources/enemy2.png", NULL);
220  enemy2Sprite = sfSprite_create();
221  sfSprite_setTexture(enemy2Sprite, enemy2Texture, sfTrue);
222  sfSprite_setScale(enemy2Sprite, (sfVector2f){0.10,0.10});
223
224  enemy3Texture = sfTexture_createFromFile("resources/enemy3.png", NULL);
225  enemy3Sprite = sfSprite_create();
226  sfSprite_setTexture(enemy3Sprite, enemy3Texture, sfTrue);
227  sfSprite_setScale(enemy3Sprite, (sfVector2f){0.10,0.10});
228
229  enemy4Texture = sfTexture_createFromFile("resources/enemy4.png", NULL);
230  enemy4Sprite = sfSprite_create();
231  sfSprite_setTexture(enemy4Sprite, enemy4Texture, sfTrue);
232  sfSprite_setScale(enemy4Sprite, (sfVector2f){0.10,0.10});
233
234  paused = sfText_create();
235  sfText_setString(paused, "PAUSED");
236  sfText_setCharacterSize(paused, 60);
237  sfText_setPosition(paused, (sfVector2f){300,330});
238  sfText_setFont(paused, sfFont_createFromFile("resources/Athletic.TTF"));
239  sfText_setColor(paused, sfRed);
240
241  //Carga de las imagenes de la nave
242
243  naveSprite = sfSprite_create();
244  balaSprite= sfSprite_create();
245
```

```
246   naveCentro= sfTexture_createFromFile("resources/jugador.png", NULL);
247   if (!naveCentro){
248     return 1;
249   }
250
251
252   naveIzquierda= sfTexture_createFromFile("resources/jugadorizq.png", NULL);
253   if (!naveIzquierda){
254     return 1;
255   }
256
257   naveDerecha= sfTexture_createFromFile("resources/jugadorder.png", NULL);
258   if (!naveDerecha){
259     return 1;
260   }
261
262   balaText = sfTexture_createFromFile("resources/bala.png", NULL);
263   if (!balaText){
264     return 1;
265   }
266   sfSprite_setTexture(balaSprite, balaText, sfTrue);
267   sfSprite_setScale(balaSprite, (sfVector2f){0.6,0.6});
268
269
270   return 0;
271 }
272
273 #define MAX_BALAS 350
274 struct bala {
275   int x;
276   int y;
277   int pasos;
278   int activa;
279 };
280
281 struct bala balas[MAX_BALAS];
282 int nbalas;
283
284 struct bala balas_enemigas[MAX_BALAS];
285 int nbalas_enemigas;
286
287 void agregar_bala(int x) {
288   int i;
289   for (i = 0; i < MAX_BALAS; i++) {
290     if (balas[i].activa == 0) {
291       balas[i].activa = 1;
292       balas[i].x = x;
293       balas[i].pasos = 0;
294       nbalas++;
295       return;
296     }
297   }
298 }
299
300
301
302 struct enemigo {
303   int x;
304   int y;
```

```
  int tipo;
  int vivo;
} enemigos[60];

void agregar_bala_enemiga(int x, int y) {
  int i;
  for (i = 0; i < MAX_BALAS; i++) {
      if (balas_enemigas[i].activa == 0) {
      balas_enemigas[i].activa = 1;
      balas_enemigas[i].x = x;
      balas_enemigas[i].y = y;
      nbalas_enemigas++;
      return;
      }
  }
}

int mover_bichos = 0;
int movimiento = -20;

bool todos_muertos(){
  int vivos=0;
  for (int i = 0; i < 61; ++i)
  {
    if (enemigos[i].vivo==1)
    {
      ++vivos;
    }
  }
  if (vivos==0)
  {
    return true;
  }

  return false;
}

int reset_juego(void) {
  int i = 0;
  naveSpriteX=380;
  naveSpriteY=480;
  nbalas = 0;
  memset(balas, 0, sizeof(struct bala) * 200);
  naveText = naveCentro;
  score = 0;
  vidas = 3;
  b[0].vida =3;
  b[1].vida = 3;
  b[2].vida =3;
  b[3].vida=3;
  mover_bichos = 0;
  movimiento = -20;

  for (i = 0; i < 60; i++) {

    enemigos[i].x = ((i % 12) * 55) + 70;
    enemigos[i].y = ((i / 12) * 45) + 100;

    if ((i / 12) == 0) {
```

```
364        enemigos[i].tipo = 3;
365      } else if (((i / 12) == 1) || ((i / 12) == 2)) {
366        enemigos[i].tipo = 2;
367      } else {
368        enemigos[i].tipo = 1;
369      }
370      enemigos[i].vivo = 1;
371    }
372
373    return 0;
374  }
375
376  int bala_y(int pasos) {
377    return 460 - (pasos * 20);
378  }
379
380  int dibujar_juego(sfRenderWindow* w){
381    int i, balas_dibujadas = 0;
382    char *string;
383    char *string2;
384
385    sfRenderWindow_clear(w, sfWhite);
386    sfRenderWindow_drawText(w,segundoscore, NULL);
387    sfRenderWindow_drawText(w, primerscore, NULL);
388    sfRenderWindow_drawText(w, hiscore, NULL);
389    sfRenderWindow_drawText(w, credit, NULL);
390
391    asprintf(&string, "Score <1>    %d", score);
392    sfText_setString(primerscore, string);
393    free(string);
394
395    //if(multijugador == true){
396     //asprintf(&string2, "Score <2>    %d", score);
397     //sfText_setString(segundoscore, string2);
398     //free(string2);
399    //}
400
401
402
403
404    asprintf(&string, "Lives %d", vidas);
405    sfText_setString(life, string);
406    free(string);
407    sfRenderWindow_drawText(w, life, NULL);
408
409    if (vidas == 0 || todos_muertos() == true){
410      sfRenderWindow_drawText(w, gameover, NULL);
411    }
412
413    if(b[0].vida > 0){
414      sfRenderWindow_drawSprite(w, barrier1, NULL);
415    }
416    if(b[1].vida > 0){
417      sfRenderWindow_drawSprite(w, barrier2, NULL);
418    }
419    if(b[2].vida > 0){
420      sfRenderWindow_drawSprite(w, barrier3, NULL);
421    }
422    if(b[3].vida > 0){
```

```
    sfRenderWindow_drawSprite(w, barrier4, NULL);
  }
  if (pausado){
    sfRenderWindow_drawText(w, paused, NULL);
  }


  sfRenderWindow_drawRectangleShape(w, c, NULL);

  sfSprite_setPosition(naveSprite, (sfVector2f){naveSpriteX, naveSpriteY});
  sfSprite_setTexture(naveSprite, naveText, sfTrue);
  sfSprite_setScale(naveSprite, (sfVector2f){0.4,0.4});
  sfRenderWindow_drawSprite(w, naveSprite, NULL);

  for (i = 0; i < 60; i++) {
    if (enemigos[i].vivo) {
      switch (enemigos[i].tipo) {
        case 1:
          sfSprite_setPosition(enemy1Sprite, (sfVector2f){enemigos[i].x, enemigos[i
    ].y});
          sfRenderWindow_drawSprite(w, enemy1Sprite, NULL);
          break;
        case 2:
          sfSprite_setPosition(enemy2Sprite, (sfVector2f){enemigos[i].x, enemigos[i
    ].y});
          sfRenderWindow_drawSprite(w, enemy2Sprite, NULL);
          break;
        case 3:
          sfSprite_setPosition(enemy3Sprite, (sfVector2f){enemigos[i].x, enemigos[i
    ].y});
          sfRenderWindow_drawSprite(w, enemy3Sprite, NULL);
          break;
        case 4:
          sfSprite_setPosition(enemy4Sprite, (sfVector2f){enemigos[i].x, enemigos[i
    ].y});
          sfRenderWindow_drawSprite(w, enemy4Sprite, NULL);
          break;
      }
    }
  }

  balas_dibujadas = 0;
  for (i = 0; i < MAX_BALAS; i++) {
    if (balas_dibujadas == nbalas)
      break;
    if (balas[i].activa == 1) {
      sfSprite_setPosition(balaSprite, (sfVector2f){balas[i].x, bala_y(balas[i].
    pasos)});
      sfRenderWindow_drawSprite(w, balaSprite, NULL);
      balas_dibujadas++;
    }
  }

  balas_dibujadas = 0;
  for (i = 0; i < MAX_BALAS; i++) {
    if (balas_dibujadas == nbalas_enemigas)
      break;
    if (balas_enemigas[i].activa == 1) {
      sfSprite_setPosition(balaSprite, (sfVector2f){balas_enemigas[i].x,
```

```
482     balas_enemigas[i].y});
483       sfRenderWindow_drawSprite(w, balaSprite, NULL);
484       balas_dibujadas++;
485     }
486   }
487
488   sfRenderWindow_display(w);
489
490   return 0;
491 }
492
493 int impacto_jugador(int x, int y) {
494   if ((naveSpriteX < x) &&
495       (naveSpriteX+80 > x) &&
496       (naveSpriteY < y) &&
497       (naveSpriteY+80 > y)){
498     return 1;
499   }
500
501   return 0;
502 }
503
504 int impacto_barrera(int x, int y){
505
506   for(size_t i = 0; i < 4; ++i){
507     if(b[i].vida > 0){
508       if ((b[i].x < x) &&
509           (b[i].x+80 > x) &&
510           (b[i].y < y) &&
511           (b[i].y+80 > y)){
512
513           --b[i].vida;
514           return 1;
515
516         }
517     }
518   }
519
520   return 0;
521
522 }
523
524 int impacto_enemigo(int x, int y) {
525   int i = 0;
526
527   for (i = 0; i < 60; i++) {
528     if (enemigos[i].vivo) {
529       if ((enemigos[i].x < x) &&
530           (enemigos[i].x+40 > x) &&
531           (enemigos[i].y < y) &&
532           (enemigos[i].y+35 > y)){
533         enemigos[i].vivo = 0;
534         switch (enemigos[i].tipo) {
535           case 1:
536             score += 10;
537             break;
538           case 2:
539             score += 20;
540             break;
```

```
          case 3:
            score += 30;
            break;
          case 4:
            score += 40;
            break;
        }
        return 1;
      }
    }
  }

  return 0;
}

int animacion_juego(sfRenderWindow* w) {
  if (todos_muertos())
  {
      for (int i = 0; i < 60; i++) {

    enemigos[i].x = ((i %12) * 55) + 70;
    enemigos[i].y = ((i / 12) * 45) + 100;

    if ((i / 12) == 0) {
      enemigos[i].tipo = 3;
    } else if (((i / 12) == 1) || ((i / 12) == 2)) {
      enemigos[i].tipo = 2;
    } else {
      enemigos[i].tipo = 1;
    }
    enemigos[i].vivo = 1;
  }
  }
  int i = 0, j;
  int soy_primero;
  int y = 0;
  int balas_actualizadas = 0;
  usleep(100000);

  if (vidas == 0) {
    return 0;
  }

  // Procesar Bichos
  mover_bichos++;
  if (mover_bichos == 5) {
    mover_bichos = 0;
    movimiento *= -1;
  }
  for (i = 0; i < 60; i++) {
    if (mover_bichos == 0) {
      enemigos[i].x += movimiento;
    }
    soy_primero = 1;
    // Soy primero de mi fila?
    for (j = i + 12; j < 60; j += 12) {
      if (enemigos[j].vivo) {
        soy_primero = 0;
        break;
```

```
      }
    }
    if (soy_primero) {
      // Debo disparar?
      if ((rand() %100) < 1) {
        if(!pausado && enemigos[i].vivo){
        agregar_bala_enemiga(enemigos[i].x+20, enemigos[i].y+40);
        }
      }
    }
  }

  // Procesar Balas
  for (i = 0; i < MAX_BALAS; i++) {
    if (balas_actualizadas == nbalas)
      break;
    if (balas[i].activa == 1) {
      balas[i].pasos++;
      if (impacto_enemigo(balas[i].x, bala_y(balas[i].pasos))) {
        // Finalizar bala
        balas[i].activa = 0;
        nbalas--;
        continue;
      }
      if (balas[i].pasos == 18) {
        balas[i].activa = 0;
        nbalas--;
      } else {
        balas_actualizadas++;
      }
    }
  }

  for (i = 0; i < MAX_BALAS; i++) {
    if (balas_actualizadas == nbalas_enemigas)
      break;
    if (balas_enemigas[i].activa == 1) {
      balas_enemigas[i].y += 20;

      // Actualiza la vida de las barreras, revisa si hubieron colisiones
      if (impacto_barrera(balas_enemigas[i].x, balas_enemigas[i].y)){
        balas_enemigas[i].activa = 0;
        nbalas_enemigas--;
      }
      else if (impacto_jugador(balas_enemigas[i].x, balas_enemigas[i].y)) {
        vidas--;
        balas_enemigas[i].activa = 0;
        nbalas_enemigas--;      }


      if (balas_enemigas[i].y > 480) {
        balas_enemigas[i].activa = 0;
        nbalas_enemigas--;
      } else {
        balas_actualizadas++;
      }
    }
  }

```

```
  dibujar_juego(w);
}

enum pantallas eventos_juego(sfRenderWindow* w, sfEvent *e){
  if(vidas == 0){
    return RESULTADOS;
  }

  if(e->type == sfEvtKeyPressed){
    switch(e->key.code){

      case sfKeyP:
        pausar();
      break;

      case sfKeyEscape:
        return INICIO;

      case sfKeyLeft:
      if(!pausado){
        if (naveSpriteX > 30) {
          naveSpriteX -=20;
          naveText = naveIzquierda;
        }
      }
        break;

      case sfKeyRight:
      if(!pausado){
        if (naveSpriteX < 730) {
          sfSprite_move(naveSprite, (sfVector2f){+20, 0});
          naveSpriteX +=20;
          naveText = naveDerecha;
        }
      }
        break;

      case sfKeySpace:
      if(!pausado){
        agregar_bala(naveSpriteX+22);
      }
      break;

    }

    sfSprite_setPosition(naveSprite, (sfVector2f){naveSpriteX, naveSpriteY});
    dibujar_juego(w);

    naveText = naveCentro;
  }

  return JUEGO;
}
```

inicio.c:

```
1
2         #include "invaders.h"
3
4 sfText* SPACE;
5 sfText* scoretable;
6 //Imagenes de los enemigos y texto de los puntos
7 sfTexture* texture1;
8 sfText* score1;
9 sfTexture* texture2;
10 sfText* score2;
11 sfTexture* texture3;
12 sfText* score3;
13 sfTexture* texture4;
14 sfText* score4;
15 sfSprite* enemy1;
16 sfSprite* enemy2;
17 sfSprite* enemy3;
18 sfSprite* enemy4;
19 //boton PLAY
20 sfSprite* bplay;
21 sfTexture* texture5;
22 //boton MUltijugador
23 sfSprite* bmultip;
24 sfTexture* texture6;
25
26
27 int cargar_inicio(void) {
28   SPACE = sfText_create();
29    sfText_setString(SPACE, "SPACE INVADERS");
30    sfText_setCharacterSize(SPACE, 80);
31    sfText_setPosition(SPACE, (sfVector2f){100,50});
32    sfText_setFont(SPACE, sfFont_createFromFile("resources/Athletic.TTF"));
33    sfText_setColor(SPACE, sfColor_fromRGB(144,12,63) );
34
35    scoretable = sfText_create();
36    sfText_setString(scoretable, "SCORE ADVANCE TABLE");
37    sfText_setCharacterSize(scoretable, 20);
38    sfText_setPosition(scoretable, (sfVector2f){300,170});
39    sfText_setFont(scoretable, sfFont_createFromFile("resources/Athletic.TTF"));
40    sfText_setColor(scoretable, sfBlack);
41
42    texture1 = sfTexture_createFromFile("resources/enemy2.png", NULL);
43    enemy1 = sfSprite_create();
44    sfSprite_setTexture(enemy1, texture1, sfTrue);
45    sfSprite_setPosition(enemy1, (sfVector2f){300, 220});
46    sfSprite_setScale(enemy1, (sfVector2f){0.1,0.1});
47
48    score1 = sfText_create();
49    sfText_setString(score1, "= 10 points");
50    sfText_setCharacterSize(score1, 20);
51    sfText_setPosition(score1, (sfVector2f){380,230});
52    sfText_setFont(score1, sfFont_createFromFile("resources/232MKRL_.TTF"));
53    sfText_setColor(score1, sfBlack);
54
55    texture2 = sfTexture_createFromFile("resources/enemy1.png", NULL);
56    enemy2 = sfSprite_create();
57    sfSprite_setTexture(enemy2, texture2, sfTrue);
```

```
  sfSprite_setPosition(enemy2, (sfVector2f){295, 270});
  sfSprite_setScale(enemy2, (sfVector2f){0.12,0.12});

  score2 = sfText_create();
  sfText_setString(score2, "= 20 points");
  sfText_setCharacterSize(score2, 20);
  sfText_setPosition(score2, (sfVector2f){380,290});
  sfText_setFont(score2, sfFont_createFromFile("resources/232MKRL_.TTF"));
  sfText_setColor(score2, sfBlack);

  texture3 = sfTexture_createFromFile("resources/enemy3.png", NULL);
  enemy3 = sfSprite_create();
  sfSprite_setTexture(enemy3, texture3, sfTrue);
  sfSprite_setPosition(enemy3, (sfVector2f){293, 330});
  sfSprite_setScale(enemy3, (sfVector2f){0.13,0.13});

  score3 = sfText_create();
  sfText_setString(score3, "= 30 points");
  sfText_setCharacterSize(score3, 20);
  sfText_setPosition(score3, (sfVector2f){380,350});
  sfText_setFont(score3, sfFont_createFromFile("resources/232MKRL_.TTF"));
  sfText_setColor(score3, sfBlack);

  texture4 = sfTexture_createFromFile("resources/enemy4.png", NULL);
  enemy4 = sfSprite_create();
  sfSprite_setTexture(enemy4, texture4, sfTrue);
  sfSprite_setPosition(enemy4, (sfVector2f){300, 395});
  sfSprite_setScale(enemy4, (sfVector2f){0.17,0.17});

  score4 = sfText_create();
  sfText_setString(score4, "= 40 points ");
  sfText_setCharacterSize(score4, 20);
  sfText_setPosition(score4, (sfVector2f){380,410});
  sfText_setFont(score4, sfFont_createFromFile("resources/232MKRL_.TTF"));
  sfText_setColor(score4, sfBlack);

  texture5 = sfTexture_createFromFile("resources/playbut.jpg", NULL);
  bplay = sfSprite_create();
  sfSprite_setTexture(bplay, texture5, sfTrue);
  sfSprite_setPosition(bplay, (sfVector2f){300, 460});
  sfSprite_setScale(bplay, (sfVector2f){0.3,0.3});

//  texture6 = sfTexture_createFromFile("resources/multiplayerbut.jpg", NULL);
//  bmultip = sfSprite_create();
//  sfSprite_setTexture(bmultip, texture6, sfTrue);
//  sfSprite_setPosition(bmultip, (sfVector2f){325, 530});
//  sfSprite_setScale(bmultip, (sfVector2f){0.4,0.4});

  return 0;
}

int dibujar_inicio(sfRenderWindow* w){
  sfRenderWindow_clear(w, sfWhite);
  sfRenderWindow_drawText(w,SPACE, NULL);
  sfRenderWindow_drawText(w, scoretable, NULL);
  sfRenderWindow_drawSprite(w, enemy1, NULL);
  sfRenderWindow_drawText(w, score1, NULL);
  sfRenderWindow_drawSprite(w, enemy2, NULL);
  sfRenderWindow_drawText(w, score2, NULL);
```

```
  sfRenderWindow_drawSprite(w, enemy3, NULL);
  sfRenderWindow_drawText(w, score3, NULL);
  sfRenderWindow_drawSprite(w, enemy4, NULL);
  sfRenderWindow_drawText(w, score4, NULL);
  sfRenderWindow_drawSprite(w, bplay, NULL);
  sfRenderWindow_drawSprite(w, bmultip, NULL);
  sfRenderWindow_display(w);
}

enum pantallas eventos_inicio(sfRenderWindow* w, sfEvent *e){
  switch(e->type) {
    case sfEvtMouseButtonPressed:
      // Boton de play?
      if (e->mouseButton.x > 300 &&
          e->mouseButton.x < 495 &&
          e->mouseButton.y > 460 &&
          e->mouseButton.y < 515) {
        return JUEGO;
      }
      if (e->mouseButton.x > 325 &&
          e->mouseButton.x < 470 &&
          e->mouseButton.y > 530 &&
          e->mouseButton.y < 570){
          return JUEGO;
        }
      break;
  }

  return INICIO;
}

//
//    while(sfRenderWindow_isOpen(w)){
//      while(sfRenderWindow_pollEvent(w, &e)){
//
//        if(e.type == sfEvtClosed){
//          printf("Cerrando\n");
//          sfRenderWindow_close(w);
//        }else if(e.type ==sfEvtKeyPressed){
//
//          //if boton jugar
//          // if boton multijugador
//          // if
//
//        }
//      }
//
//      sfRenderWindow_clear(w, sfWhite);
//      sfRenderWindow_drawText(w,SPACE, NULL);
//      sfRenderWindow_drawText(w, scoretable, NULL);
//      sfRenderWindow_drawSprite(w, enemy1, NULL);
//      sfRenderWindow_drawText(w, score1, NULL);
//      sfRenderWindow_drawSprite(w, enemy2, NULL);
//      sfRenderWindow_drawText(w, score2, NULL);
//      sfRenderWindow_drawSprite(w, enemy3, NULL);
//      sfRenderWindow_drawText(w, score3, NULL);
//      sfRenderWindow_drawSprite(w, enemy4, NULL);
//      sfRenderWindow_drawText(w, score4, NULL);
//      sfRenderWindow_drawSprite(w, bplay, NULL);
```

```
890 //      sfRenderWindow_drawSprite(w, bmultip, NULL);
891 //      sfRenderWindow_display(w);
892 //
893 //    }
894 //
895 //
896 //    sfText_destroy(SPACE);
897 //    sfText_destroy(scoretable);
898 //    sfSprite_destroy(enemy1);
899 //    sfSprite_destroy(enemy2);
900 //    sfSprite_destroy(enemy3);
901 //    sfSprite_destroy(enemy4);
902 //    sfSprite_destroy(bmultip);
903 //    sfSprite_destroy(bplay);
904 //    sfText_destroy(score4);
905 //    sfText_destroy(score1);
906 //    sfText_destroy(score2);
907 //    sfText_destroy(score3);
908 //    sfRenderWindow_destroy(w);
909 //
910 //    return 0;
911 // }
912
913
```

scores.c:

```
#include "invaders.h"


int f_pantallaendgame(){


  sfVideoMode pj = {800, 600 , 32};
  sfRenderWindow* w3;
  w3 = sfRenderWindow_create(pj, "SPACE INVADERS", sfClose, NULL);

  sfEvent e3;

  //texto ENDGAME
  sfText* endgame;
  endgame = sfText_create();
  sfText_setString(endgame, "END GAME");
  sfText_setCharacterSize(endgame, 70);
  sfText_setPosition(endgame, (sfVector2f){240,50});
  sfText_setFont(endgame, sfFont_createFromFile("resources/Athletic.TTF"));
  sfText_setColor(endgame, sfColor_fromRGB(144,12,63) );


  //texto SCORES
  sfText* scores;
  scores = sfText_create();
  sfText_setString(scores, "SCORES");
  sfText_setCharacterSize(scores, 40);
  sfText_setPosition(scores, (sfVector2f){330,140});
  sfText_setFont(scores, sfFont_createFromFile("resources/Athletic.TTF"));
  sfText_setColor(scores, sfBlack);

  sfTexture* texture1;
  texture1 = sfTexture_createFromFile("resources/enemy2.png", NULL);
  sfSprite* enemy1;
  enemy1 = sfSprite_create();
  sfSprite_setTexture(enemy1, texture1, sfTrue);
  sfSprite_setPosition(enemy1, (sfVector2f){110, 526});
  sfSprite_setScale(enemy1, (sfVector2f){0.14,0.14});

  sfTexture* texture2;
  texture2 = sfTexture_createFromFile("resources/enemy1.png", NULL);
  sfSprite* enemy2;
  enemy2 = sfSprite_create();
  sfSprite_setTexture(enemy2, texture2, sfTrue);
  sfSprite_setPosition(enemy2, (sfVector2f){357, 522});
  sfSprite_setScale(enemy2, (sfVector2f){0.15,0.15});

  sfTexture* texture3;
  texture3 = sfTexture_createFromFile("resources/enemy3.png", NULL);
  sfSprite* enemy3;
  enemy3 = sfSprite_create();
  sfSprite_setTexture(enemy3, texture3, sfTrue);
  sfSprite_setPosition(enemy3, (sfVector2f){580, 520});
  sfSprite_setScale(enemy3, (sfVector2f){0.16,0.16});


```

```
972
973
974  while(sfRenderWindow_isOpen(w3)){
975    while(sfRenderWindow_pollEvent(w3, &e3)){
976
977      if(e3.type == sfEvtClosed){
978        printf("Cerrando\n");
979        sfRenderWindow_close(w3);
980      }//else if{
981
982        //obtener datos del file donde se guardan los puntajes
983        //hacer la comparacion de datos
984        //agregar nuevo puntaje en caso de
985        //imprimirlos en pantalla
986
987      }
988
989      sfRenderWindow_clear(w3, sfWhite);
990      sfRenderWindow_drawText(w3,endgame, NULL);
991      sfRenderWindow_drawText(w3, scores, NULL);
992      sfRenderWindow_drawSprite(w3, enemy1, NULL);
993      sfRenderWindow_drawSprite(w3, enemy2, NULL);
994      sfRenderWindow_drawSprite(w3, enemy3, NULL);
995      sfRenderWindow_display(w3);
996
997    }
998
999
1000   sfText_destroy(endgame);
1001   sfText_destroy(scores);
1002   sfSprite_destroy(enemy1);
1003   sfSprite_destroy(enemy2);
1004   sfSprite_destroy(enemy3);
1005   sfRenderWindow_destroy(w3);
1006
1007
1008
1009
1010   return 0;
1011 }
1012
1013
```

Makefile:

```
CC=gcc

# Files
BASEDIR=.
INCLDIR=$(BASEDIR)/incl
SRCDIR=$(BASEDIR)/src
BINDIR=$(BASEDIR)/bin
DOCDIR=$(BASEDIR)/doc

# Compiler options and flags
CFLAGS=-g -I$(INCLDIR) -D_GNU_SOURCE
LIBRARIES=-lcsfml-graphics -lcsfml-audio
PROGRAM=SpaceInvaders

########################################
# Targets
########################################

all: $(BINDIR)/$(PROGRAM)

$(BINDIR)/$(PROGRAM): src/main.o src/inicio.o src/juego.o #src/scores.o
  $(CC) -o $@ $^ $(LIBRARIES)

clean:
  rm -rf $(BINDIR)/$(PROGRAM) src/*.o

run: $(BINDIR)/$(PROGRAM)
  $(BINDIR)/$(PROGRAM)

.PHONY: all clean
```

README:

```
Space Invaders
2018, Nov 14
Belinda Brown Ram rez - timna.brown@ucr.ac.cr
Alexander Calderon Torres - alexander.calderontorres@ucr.ac.cr
Carolina Urrutia Nu ez - ana.urrutia@ucr.ac.cr

Este programa permite jugar al Space Invaders
Primero se debe poner el comando en la terminal: make
Despu s, vamos a correr: make run
Se va a generar una pantalla en donde al hacerle click sobre el bot n de "PLAY" nos
    redirecciona a otra ventana en donde se inicia el juego. Si se desea reiniciar
    el juego se preciona la tecla de "Esc". Por otro lado, para mover la nave se usa
    las flachas del teclado de izquierda y derecha respectivamente, para disparar
    misiles se usa la tecla de "espacio".
```