

MEMORIA DEL PROYECTO DE DESARROLLO DE UNA APLICACIÓN DE E/S PARA LA NINTENDO DS

Grupo EC304

Oihan Irastorza Carrasco
oirastorza001@ikasle.ehu.eus

Eneko Pizarro Liberal
epizarro001@ikasle.ehu.eus

Franco Alejandro Valdiviezo Ríos
fvaldiviezo001@ikasle.ehu.eus

14 de mayo de 2021

Índice

1. Resumen	1
2. Introducción	2
3. Descripción del juego	2
3.1. Juego original	2
3.2. Nuestra versión	2
3.3. Estructuración del Juego	3
4. Autómata	5
5. Desarrollo del proyecto	6
5.1. Árbol de directorios	6
5.2. Tipos de datos implementados	6
5.3. Descripción general de los estados	7
5.4. Mostrar tecla pulsada por pantalla	10
5.5. Gestión de las interrupciones	11
5.6. Control del temporizador	12
5.7. Animación del logo SPACE INVADERS	12
5.8. Menú de inicio	13
5.9. <i>Setup</i> de la partida	13
5.10. Movimiento de los alienígenas	13
5.11. Movimiento del cañón láser	14
5.12. Projectiles	14
5.13. Colisiones	14
5.14. Menú de pausa	15
5.15. Fin de la partida	15
5.16. Estadísticas	15
6. Conclusiones	16
7. Gestión del proyecto	18
8. Referencias	19

1. Resumen

En esta memoria el lector se encontrará no solo el informe del desarrollo del videojuego, sino también con el proceso que este nos ha supuesto como grupo; comprender los procedimientos de implementación del mismo y algunos conceptos fundamentales relacionados con la entrada y salida de los sistemas de computación. En concreto, hablamos de la consola Nintendo DS, la cual estudiaremos con el fin de desarrollar el proyecto en forma de videojuego para la asignatura de Estructura de Computadores.

Hemos dividido la memoria en secciones para una mejor comprensión y estructuración. Al principio se podrá leer la descripción del juego original y la adaptación que nosotros hemos diseñado. El propósito de esta adaptación consiste en comprender el funcionamiento de los sistemas de entrada y salida, el sistema de interrupciones y los métodos de sincronización.

Una vez acotada la intención del juego, proseguimos con su estructuración en forma de autómatas. Aquí es donde se puede apreciar la forma que tomará el juego, y las funciones necesarias que habrá que implementar una vez pasemos al desarrollo del mismo. Es importante focalizarse en cada uno de los estados, ya que son las principales secciones en las que se dividirá el juego.

Después de realizar el autómata, el lector podrá informarse del desarrollo de cada una de las partes del proyecto. Por secciones podrá leer los pasos que hemos seguido para el cumplimiento de los requisitos del juego. Hay que tener en cuenta que hemos tenido que programar no solo el sistema de interrupciones de la Nintendo DS, sino también el controlador principal de los diferentes periféricos que se han de utilizar.

Al final del documento se hallan las conclusiones del aprendizaje, del trabajo en equipo realizado y de las dificultades que hemos superado para lograr el resultado final, que es el videojuego. También se podrá observar el seguimiento del proyecto gracias a la herramienta “Trello” y las referencias situadas en la bibliografía.

2. Introducción

El proyecto que hemos realizado ha consistido en recrear el clásico *Space Invaders* en una versión más sencilla en la Nintendo DS. El principal objetivo ha sido poner en práctica los conceptos estudiados en el tema 3, los cuales incluyen controladores, autómatas... en resumen, aquellos conceptos que abarcan los sistemas de entrada/salida vistos en la asignatura de Estructura de Computadores.

Entre los conceptos aprendidos, son destacables los métodos de sincronización, los cuales pueden ser por encuesta o por interrupción. Hemos hecho uso de estos métodos al programar el funcionamiento de la pantalla táctil, botones de la Nintendo y temporizadores.

Transversalmente, nos hemos introducido en el mundo de la programación en C [1]. Incluimos conceptos básicos como manejo de punteros, formateo de *Strings*, lectura y escritura de memoria, flujo de ejecución...vamos, *tocar el bit*.

A lo largo del proyecto hemos consultado constantemente la librería *libnds* [2] y algunos manuales de programación para la Nintendo DS [3].

3. Descripción del juego

3.1. Juego original

Space Invaders [4] es un videojuego diseñado para uno o dos jugadores. Durante el juego, se generan 55 alienígenas repartidos en 5 filas y 11 columnas. Estos invasores están constantemente moviéndose de izquierda a derecha o viceversa en la pantalla.

El objetivo del juego consiste en destruir tantos invasores como sea posible. Los enemigos avanzan paso a paso hacia el jugador mientras se desplazan de lado a lado. Las búnkeres son gradualmente destruidos tanto por el ataque de los alienígenas como por el del cañón del propio jugador. A medida que el número de enemigos decrece, la velocidad de estos aumenta.

Los alienígenas de las 2 filas valen 10 puntos, los de la mitad 20 y los de las dos últimas 30. Por eliminar todos los enemigos de un nivel se puede obtener un máximo de 990 puntos. Se puede obtener una puntuación aleatoria adicional de 50, 100, 150 o 300 puntos disparando a los OVNI's que cruzan la pantalla horizontalmente por encima de los alienígenas. Cuando se obtienen 1500 puntos, se obtiene una vida adicional (marcada por un pitido).

3.2. Nuestra versión

El videojuego está diseñado para un único jugador. Se generan 30 alienígenas en 9 columnas y 4 filas sobre un fondo animado de estrellas. Al igual que en el

juego original [5], los alienígenas se mueven de izquierda a derecha y viceversa, acercándose cada vez más a la Tierra.

El objetivo en nuestra versión consiste en matar todos los alienígenas antes de que acabé el tiempo. A diferencia de la versión clásica, nuestro cañón no tiene búnkeres para protegerse. Tampoco se generan OVNI's. De hecho, se obtienen 10 puntos por sendos alienígenas.

Hay 4 formas posibles de finalizar la partida.

- Se agota el tiempo de partida para matar a los invasores.
- Los invasores llegan finalmente a la posición del jugador, esto es, a la Tierra, destruyendo el cañón láser (con su respectiva animación).
- Los enemigos disparan aleatoriamente al cañón láser y este es destruido al ser alcanzado por tres disparos.
- El usuario decide terminar la partida presionado el botón <A>.

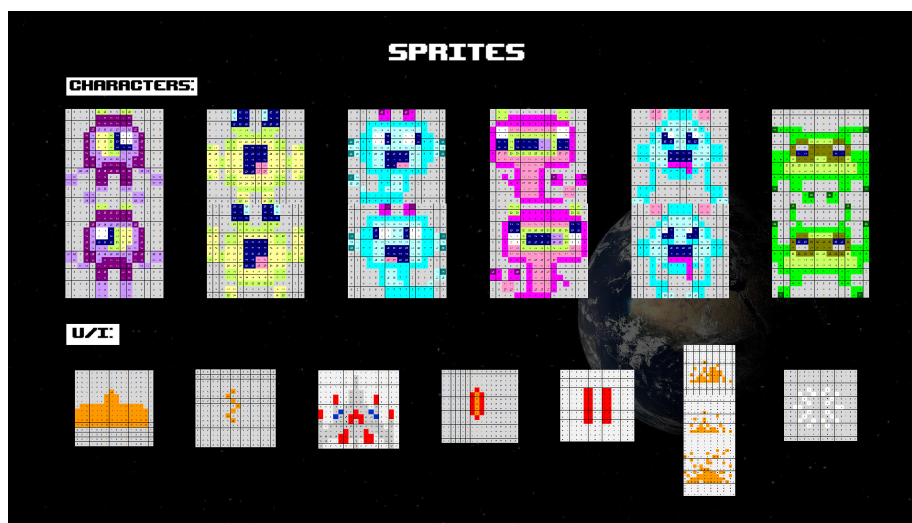


Figura 1: Algunos de los *sprites* diseñados para el proyecto

3.3. Estructuración del Juego

El juego nos da la bienvenida con una animación del logotipo original de “Space Invaders” en la pantalla inferior de la Nintendo DS, mientras que en la superior se mostrará tanto información sobre los creadores del juego y el grupo al que pertenecen como algunas indicaciones para el jugador.

En general, la pantalla inferior se utilizará para albergar los gráficos y la partida en sí. En cambio, el único propósito de la pantalla superior será mostrar datos relacionados con el estado del juego. Por ejemplo, mostrarle al jugador las teclas que está presionando y el modo de lectura de las mismas; bien por sincronización por encuesta o bien por interrupción. También la puntuación que el usuario ha conseguido a lo largo de la partida y si ha finalizado la misma, mostrando las diferentes causas de fin de la partida.

Al inicio del juego, si el usuario pulsa la tecla <SELECT> será direccionado al menú principal del juego, donde podrá elegir entre comenzar la partida (“play”), ir a la configuración del juego (“options”) o finalizar el mismo (“quit game”).

Si el usuario decide comenzar la partida seleccionando la opción “play” (pulsando <START>), la pantalla procederá a cambiarse, mostrando un fondo animado de estrellas brillando sobre la Tierra. En ese mismo instante aparecerán los invasores en pantalla, junto con el cañón láser sobre el cual el jugador tomará el control.

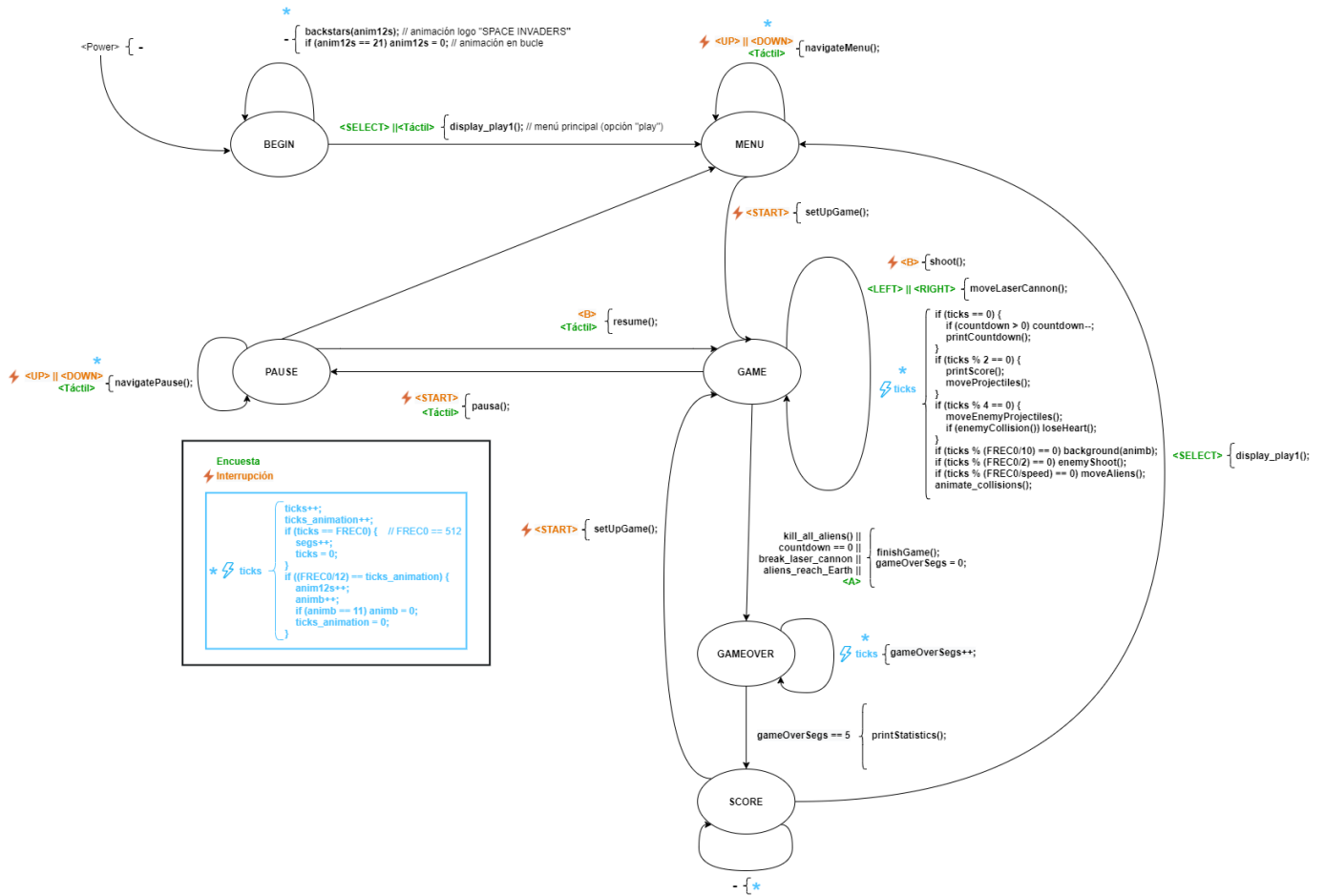
Los invasores están formados por 4 tipos, de los 5 diseños posibles, de alienígenas animados ordenados en 4 filas, con un tipo de alienígena en cada una. Estas filas irán bajando hacia el usuario a medida que las mismas van desplazándose de derecha a izquierda colisionando con el borde de la pantalla.

El objetivo del juego será destruir todos los invasores disparando hacia ellos gracias a pulsar la tecla , la cual puede disparar hasta 10 láseres en cada momento. Si la puntería del usuario es la correcta, moviendo el láser de izquierda a derecha mediante las flechas de la consola, colisionarán con un alienígena cada vez, explotándolo, mostrando la animación propia y añadiendo al marcador del jugador 10 puntos por cada disparo acertado.

Si finaliza el juego con alguno de los casos mencionado anteriormente, se mostrarán las estadísticas de la partida en la pantalla secundaria, además de la opción de empezar una nueva o volver al menú principal.

El usuario será capaz de pausar la partida pulsando la tecla <START>. Se mostrará un menú en el cual se verá una Nintendo DS más pequeña que indica al jugador las funciones de cada botón. Desde aquí se puede continuar la partida presionando el icono “Resume” situado en la pantalla inferior del diseño de la Nintendo DS pequeña, o finalizar la partida pulsando el icono “Quit”, situado justo debajo.

4. Autómata



5. Desarrollo del proyecto

5.1. Árbol de directorios

- **source.** Aquí se encuentran todos los archivos `.c`, donde está implementado la mayoría del código fuente. Los programas están divididos en 7 ficheros:
 - *main.c.* El programa principal. Acciones a realizar por encuesta.
 - *controllers.c.* Gestión de interrupciones.
 - *keypad_screen.c.* Todo lo relacionado con las rutinas de atención al teclado y la pantalla táctil.
 - *timers.c.* Manejo de los temporizadores, animaciones y todo aquello que tenga que ver con el movimiento de los *sprites*.
 - *sprites.c.* Administración de los *sprites*. Desde el diseño de cada uno de ellos, pasando por la asignación en memoria, hasta las funciones correspondientes para mostrarlos por pantalla.
 - *backgrounds.c.* Funciones pertinentes para cargar los fondos en memoria.
 - *functions.c.* Aquí es donde guardamos todas las funciones auxiliares para el juego.
- **include.** En este directorio podemos encontrar las declaraciones de las funciones implementadas en los archivos del directorio *source*. También podemos encontrar el fichero *defines.h*, donde tenemos declaradas la mayoría de las constantes, variables globales y estructuras de datos.
- **gfx.** Es donde hemos almacenado todos los fondos que hemos usado en nuestro juego, cada uno con sus respectivos archivos `.PNG` y `.grit`.
- **build.** El directorio de salida de compilación.

5.2. Tipos de datos implementados

Para diferenciar unos *sprites* de otros, hemos creado dos tipos de datos, cada uno con sus respectivos miembros:

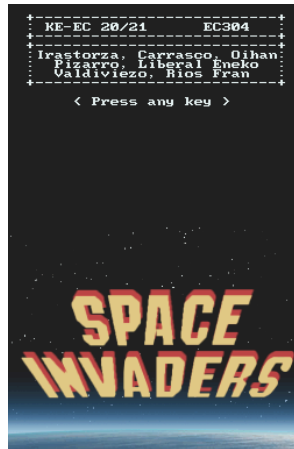
- `t_sprite`. Para identificar tanto a los alienígenas como al cañón láser.
 - `x`. Posición en el eje X.
 - `y`. Posición en el eje Y.
 - `hp` (*health points*). Los puntos de vida.
 - `characterN`. Número que identifica el tipo de alienígena.
 - `spriteN`. Número oscilante entre 1 y 2 que define la animación del *sprite*.

- cont. Contador para gestionar la animación de colisión.
- t_projectiles. Se encarga de estructurar los proyectiles, tanto para el cañón láser como para los enemigos.
 - x. Posición en el eje X.
 - y. Posición en el eje Y.
 - active. Número que evalúa cuándo el proyectil está activo.
 - sprite. Número que identifica el diseño de *sprite* correspondiente al proyectil.

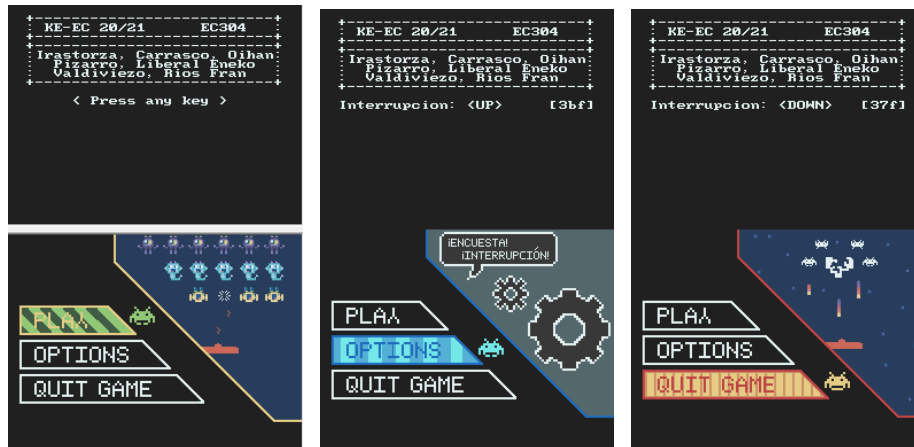
5.3. Descripción general de los estados

En la implementación de nuestro juego hemos distinguido entre 6 estados distintos:

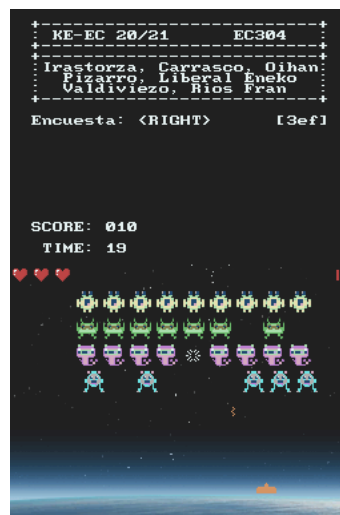
- BEGIN. El estado inicial, donde se muestra la pantalla de inicio con el logotipo de Space Invaders.



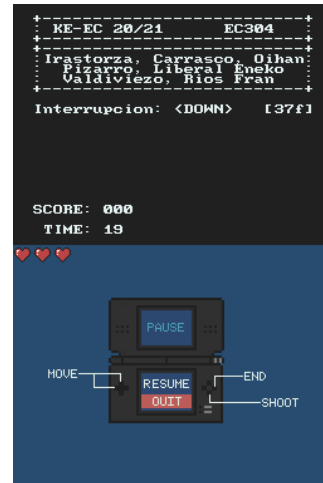
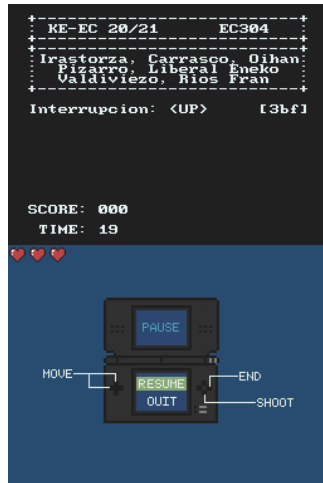
- MENU. Podremos elegir si empezar a jugar con el botón “play”, configurar el juego en el apartado “options” o abandonar el juego con el botón “quit game”.



- GAME. La partida en sí.



- PAUSE. Se pausa la partida actual y se despliega un menú en pantalla con dos opciones, una para continuar con la partida (“resume”) y otra para abandonarla (“quit”).



- GAMEOVER. Es el estado al que se pasa al finalizar la partida. La pantalla secundaria indica si se ha ganado o se ha perdido. En caso haber perdido la partida, se muestra el motivo de la derrota. Este estado dura siempre 5 segundos.



- **SCORE.** Tras el estado GAMEOVER, en la pantalla secundaria se muestran las estadísticas generales del juego. Además, se proporciona al usuario la opción de reiniciar la partida o volver al menú principal.



5.4. Mostrar tecla pulsada por pantalla

Inicialmente, tuvimos que mostrar un mensaje en la pantalla secundaria que indicaba la tecla pulsada, el valor hexadecimal del registro de datos *KEYS_DAT* y el método de sincronización utilizado (véanse figura 2 y figura 3).

```
Encuesta: <LEFT>          [3df1]
```

Figura 2: Tecla <A> detectada por encuesta

```
Interrupcion: <START>     [3f71]
```

Figura 3: Tecla <START> detectada por interrupción

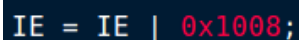
La función de mostrar el mensaje se llama desde el archivo *main.c* o *keypad.screen.c*, dependiendo de si el método utilizado para la sincronización es por encuesta o interrupción, respectivamente.

5.5. Gestión de las interrupciones

Antes de nada, es importante mencionar que la Nintendo DS no posee un hardware específico para administrar las interrupciones. En cambio, este trabajo lo realiza un programa llamado *Interrupt Dispatcher* [6], que es el responsable de leer los registros de interrupción. Es este pequeño software quien decide cuales son las interrupciones aceptadas y, en caso de estar permitidas, accede a la tabla de interrupciones para obtener las rutinas de atención correspondientes.

Este apartado lo hemos desarrollado desde el archivo *controllers.c*, del cual es importante mencionar algunas de sus funciones.

- *enable_interrupts*. Utilizada para habilitar las interrupciones. Hemos activado el bit 12 del registro IE (Interrupt Enable) para activar la línea de interrupción de las teclas (*IRQ_KEYS*) y el bit 3 para el temporizador *timer0*. Para activar estos bits y dejar los demás en su estado actual, hemos aplicado la máscara 0x1008 con un operador binario *OR* [7] (véase figura 4).



```
IE = IE | 0x1008;
```

Figura 4: Modificación del registro IE

- *configure_control_registers*. Aquí es donde programamos los registros de control de los periféricos.

Por una parte, hemos configurado el registro de control *KEYS_CNT* para que nos detecte las siguientes teclas por interrupción: , <START>, <A>, <DOWN> y <UP>. El valor del registro en valor hexadecimal se queda como 0x40CA.

Por otra parte, hemos configurado el registro de control del temporizador *TIMER0_CNT* para que la frecuencia de interrupción sea de 512 *ticks/s*. Para ello, basta con dejar los dos últimos bits del registro a 0 y establecer el *latch* a 0 en con el registro de datos del temporizador *TIMERS_DAT*. También es importante que el bit 6 del registro *TIMER0_CNT* esté encendido para que genere interrupción al desbordarse, y el bit 7 para que el temporizador funcione. Ergo asignamos al registro el valor en hexadecimal 0x00C0.

- *define_interrupt_vector_table*. Esta función es la que se encarga de inicializar el vector de interrupciones. Cuando llega una petición de interrupción, el gestor mirará aquí cual es la rutina de atención correspondiente (véase figura 5).

```
/*
 * Rutina de Atencion al Teclado
 */
irqSet(IRQ_KEYS, keypad_handler);
```

Figura 5: Interrupción detectada y rutina a ejecutar

- *interrupts_settings*. Es la función que engloba las mencionadas anteriormente. Es llamada por el programa principal en el archivo *main.c*.

5.6. Control del temporizador

Este apartado lo hemos administrado desde el fichero *timers.c*. Hablaremos más en profundidad sobre las funciones definidas en este archivo en los siguientes apartados, pero es importante destacar que aquí se encuentra la rutina de atención al temporizador *keypad_handler* (véase figura 6), que en nuestro caso se ejecuta 512 veces por segundo, acorde con lo mencionado anteriormente. Desde aquí llevaremos el control de los *ticks* por segundo, la cuenta atrás, FPS de las animaciones, segundero...

```
/*
 * Rutina de atención a la interrupción del temporizador
 */
void timer_handler() {
    ticks++;
    if (ticks == FREC0) {
        segs++;
        ticks = 0;
    }
}
```

Figura 6: Fragmento de la rutina de atención al temporizador que administra el segundero

5.7. Animación del logo SPACE INVADERS

Hemos diseñado un conjunto de 21 fondos [8] que conforman la animación inicial del logotipo. Estos fondos se muestran a 12FPS en bucle, hasta que el usuario decida pulsar la pantalla táctil o el botón <SELECT>. El programa detecta si se ha pulsado la pantalla táctil gracias a la función *is_the_screen_touched* definida en el archivo *keypad_handler.c*.

5.8. Menú de inicio

En el caso del menú, tenemos una colección de 6 fondos divididos en grupos de 2 para poder animar cada una de las opciones (“play”, “options” y “quit game”). Para navegar por el menú, el programa espera bien por interrupción las teclas <UP> y <DOWN> o bien por encuesta que se pulse la pantalla táctil.

La posición de la pantalla es registrada con las funciones auxiliares *getX* y *getY* definidas en el archivo *keypad_handler.c*. La opción del menú se almacena en la variable *menuState*. Si su valor es 0 (corresponde a la opción “play”) y se pulsa el botón “play” en la pantalla o el botón <START>, se dará comienzo a una nueva partida.

5.9. Setup de la partida

La función *setUpGame* es aquella que se ejecuta al comenzar un nuevo juego o al reiniciar la partida una vez terminada. Es de vital importancia remarcar la ejecución de esta función, ya que es la que prepara el escenario del juego e inicializa los parámetros necesarios para el comienzo. Algunos de sus cometidos son:

- Generar el cañón láser y los alienígenas (función *generateNPC* en *functions.c*).
- Establecer la velocidad de movimiento de los alienígenas.
- Preparar el cargador de disparos (función *generateNPC*).
- Iniciar la cuenta atrás.
- Inicializar la puntuación a 0.
- ...

5.10. Movimiento de los alienígenas

Los enemigos se mueven de manera conjunta, realizando pequeñas olas por filas. Cuando uno de los alienígenas llega al borde del escenario, espera a que la última fila termine de desplazarse para descender todos simultáneamente. Entonces, se invierte la dirección y los alienígenas empiezan a desplazarse hacia el lado opuesto. Así sucesivamente hasta llegar a la Tierra.

El código del movimiento se puede encontrar en la función *moveAliens* en el archivo *timers.c*.

5.11. Movimiento del cañón láser

El cañón se mueve por encuesta pulsando los botones <LEFT>, <RIGHT> o cualquiera de estas dos teclas más la de disparar, es decir, la tecla , gracias a la función *moveLaserCannon* del archivo *functions.c*. Si toca alguno de los dos bordes del escenario, el cañón se detiene.

Para conseguir la detección de múltiples teclas, hemos modificado la función *pressed_key_value* en el archivo *keypad_screen.c*, para que envíe una señal distinta en caso de pulsar <LEFT+B> o <RIGHT+B>.

5.12. Projectiles

Distinguimos dos tipos de disparos, el del cañón láser y el de los alienígenas.

Por una parte, el cañón puede tener un total de 10 disparos simultáneamente en pantalla. Cuando se ejecuta la función *shoot* del archivo *functions.c*, se activa uno de los disparos. Hemos programado la rutina de movimiento de los disparos *moveProjectiles* de tal forma que se muevan en vertical y hacia arriba todos aquellos disparos que estén activos, una vez cada 2 *ticks*. La bala es desactivada si atraviesa la pantalla o colisiona con algún alienígena.

Por otra parte, tenemos los disparos enemigos. La frecuencia de disparo es de 2 veces por segundo. Se escoge un alienígena aleatorio, y este generará un disparo que se moverá verticalmente y hacia abajo, hasta llegar a la Tierra o colisionar con el cañón láser. Las funciones involucradas en el proceso son similares a las anteriores, *moveEnemyProjectiles* en *timers.c*, y *getRandomAlien* y *enemyShoot* en *functions.c*.

5.13. Colisiones

Para este apartado es necesario también distinguir entre dos tipos de colisiones.

Por un lado, la colisión de los disparos del cañón láser se comprueba cada vez que un disparo se mueve. Se pueden analizar las condiciones de colisión [9] en la función *collision* del archivo *functions.c*. En caso de detectarse colisión, se elimina el alienígena con la rutina *removeAlien* de *functions.c* y se inicializa un contador intrínseco del alienígena, utilizado exclusivamente para mostrar la animación de colisión.

Por otro lado, la colisión del disparo enemigo es más sencilla, puesto que no tiene ningún tipo de animación. Al chocar con el cañón láser, el disparo desaparece y el cañón pierde una vida. Las rutinas utilizadas en esta acción son *enemyCollision* y *loseHeart* en *functions.c*.

5.14. Menú de pausa

Es una sencilla interfaz que consta de 2 fondos, uno para el botón de reanudar la partida y otro para abandonarla. Podemos navegar por el menú con las teclas <UP>, <DOWN> y o con la pantalla táctil.

Cuando se accede al menú de pausa, se ejecuta la función *pause* de *functions.c*, que se encarga de detener y esconder todos los *sprites* que haya en pantalla. Si se reanuda el juego, se llama a la función *resume* para revertir el efecto.

5.15. Fin de la partida

Se trata de un estado que dura 5 segundos, gestionado con la variable *gameOverSegs*. Cuando la partida concluye, todos los *sprites* de la pantalla principal son eliminados. Si acabamos con todos los alienígenas, un mensaje parpadeante con la palabra “VICTORY” se mostrará en la pantalla secundaria. En caso darse una de las condiciones de derrota ya mencionadas, el mensaje será “GAME OVER” seguido por la razón de la derrota. El mensaje a mostrar se evalúa mediante la función *printEndReason* en *functions.c*.

5.16. Estadísticas

Tras los 5 segundos de fin de partida, la rutina *printStatistics* mostrará en la pantalla secundaria una pequeña tabla indicando la cantidad de alienígenas destruidos, el tiempo sobrevivido y la puntuación total. Después de otros 5 segundos (gestionados por la variable *scoreSegs*) un mensaje parpadeante advertirá al usuario de su próxima acción. En caso de pulsar <START>, se ejecuta la función *setUpGame* y se reinicia la partida. En cambio, si se oprime la tecla <SELECT>, el jugador volverá al menú principal.

6. Conclusiones

A pesar de las dificultades que nos han ido surgiendo a lo largo del desarrollo, hemos conseguido implementar con éxito todos los requisitos encomendados de cara al proyecto. Entre otros, la opción a concluir la partida por distintas razones, el movimiento de los alienígenas y el cañón, los disparos y todo lo que engloba ello (*hitboxes* y colisiones, además de todas las funciones para la generación y muestra de sprites), el sistema de puntuación y un largo etcétera.

Además, hemos aportado al proyecto mejoras no obligatorias, entre las cuales podemos mencionar:

- Sprites adicionales para los enemigos y el cañón.
- Animaciones. Desde el logotipo del inicio del juego, pasando por los menús de inicio y de pausa, hasta el movimiento de los alienígenas y la colisión, el fondo estrellado de la partida o la destrucción de la nave.
- La posibilidad de que los enemigos disparen de vuelta.
- La integración de la pantalla táctil tanto en la interfaz de inicio como en la de pausa.
- ...

Sin embargo y a pesar de haber logrado estos objetivos, han quedado algunas ideas en el tintero, por ejemplo la generación del “jefe final”, los búnkeres de la versión clásica o añadir efectos de sonido y música ambiental.

Por último, cabe destacar la modificación de algunos apartados del enunciado original:

“Inicialmente se debe escribir en la pantalla superior un mensaje que indica al usuario que debe de pulsar la pantalla táctil para comenzar a jugar una partida. Se mostrará un mensaje del estilo a: ‘Para comenzar toque la pantalla táctil’”.

En nuestro caso, el mensaje es “press any key”. El jugador podrá pulsar cualquier tecla o la pantalla táctil para continuar.

“El movimiento del cañón se realiza con la flecha izquierda detectada por interrupción y con la flecha derecha por encuesta.”

Hemos decidido sincronizar estas dos teclas mediante encuesta por dos razones. La primera y la de más peso, es que la encuesta nos permite movernos manteniendo la tecla pulsada, mientras que la interrupción no. La segunda razón es la coordinación; resulta más sencillo equilibrar la velocidad de movimiento hacia ambas direcciones si estas funcionan mediante el mismo método de sincronización.

Las teclas <UP> y <DOWN> también las hemos detectado por interrupción. Al fin y al acabo, su único cometido es moverse por el menú, y no nos interesa mantener la tecla pulsada o que se pueda detectar accidentalmente una pulsación como varias.

“El movimiento de cada extraterrestre es autónomo”.

El movimiento de los alienígenas es coordinado por olas en nuestra versión. Los extraterrestres se mueven por filas una tras otra, y cuando se termina de mover la última, la secuencia se vuelve a repetir desde la primera.

Hemos optado por este tipo de movimiento por varias razones. Primeramente porque nos atraía más el hecho de mover los alienígenas con algo de sentido, haciendo que parecieran una tropa organizada más que un grupo de enemigos sin rumbo. Por otra parte, supusimos que el aprendizaje sería mayor para nosotros si intentáramos crear un algoritmo de movimiento por ondas en vez de simplemente asignar un movimiento aleatorio a cada uno de los *sprites*.

“Los extraterrestres no disparan, sólo pretenden llegar a la Tierra”.

En nuestro caso, dado que nuestro cañón láser dispara con una frecuencia mucho más rápida que en el juego original, hemos decidido aumentar un poco la dificultad añadiendo disparos enemigos, además de tres vidas al cañón láser.

“Por cada crab_alien liquidado el jugador obtendrá un punto y aparecerá otro nuevo en otra posición al azar en la mitad superior de la pantalla principal”.

En el sistema implementado, todos los alienígenas valen 10 puntos, al igual que los primeros enemigos la versión clásica. Además, matarlos no hace que se generen más alienígenas. Eliminar a todos es la forma de ganar en nuestra versión.

“La partida puede terminar por tres motivos: (1) cuando un extraterrestre llega a la Tierra, (2) cuando haya pasado el tiempo estipulado o (3) cuando se pulse la tecla <A> durante el juego”.

Hemos añadido un cuarto final que será provocado si la nave es alcanzada por disparos enemigos 3 veces.

7. Gestión del proyecto

Hemos dividido las horas dedicadas al proyecto de la siguiente forma:

Horas invertidas	
Laboratorios (+ horas extra)	15 (10.5 + 4.5)
Desarrollo del juego fuera del laboratorio	30
Redacción de la memoria	5
Tutorías	3
Total	53

A lo largo del desarrollo del proyecto, nos hemos valido de la herramienta de administración de proyectos “Trello” (véase figura 7).

Hemos separado las tareas en 3 tableros distintos: “To Do”, “Doing” y “Done” [10]. De esta forma, los integrantes del grupo hemos sabido en todo momento las tareas que estaban en proceso, los trabajos disponibles, los objetivos conseguidos... además, hemos creado un par de tableros más; el de “Dudas”, para apuntar todas aquellas cuestiones que requieran de alguna reunión grupal o tutoría con el profesor; y el de “Repositorio”, para tener un acceso directo a Drive donde gestionábamos los directorios de nuestro videojuego.

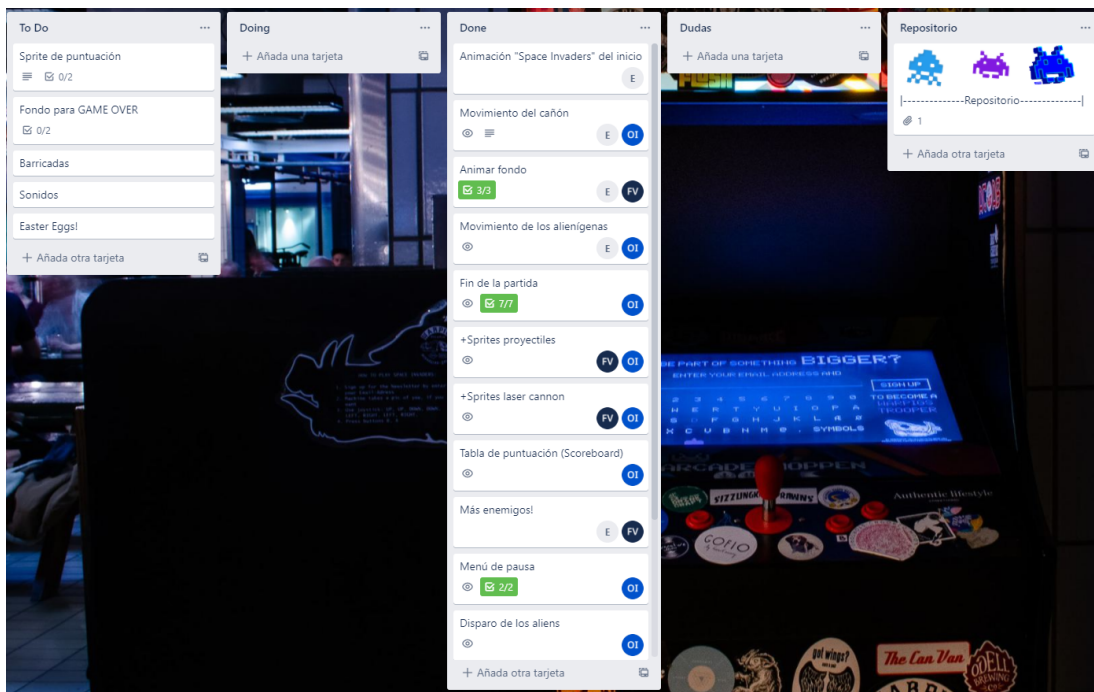


Figura 7: Visión general del proyecto en la herramienta “Trello”

8. Referencias

- [1] J. Y. Trevis Rothwell, “The gnu c reference manual,” 2020. [Online]. Available: <https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html#Credits>
- [2] Doxygen, “Libnds documentation,” 2020. [Online]. Available: <https://patater.com/files/projects/manual/manual.html>
- [3] J. Amero, “Introduction to nintendo ds programming,” 2006. [Online]. Available: <https://patater.com/files/projects/manual/manual.html>
- [4] T. Corporation, *Space Invaders - Arcade - Manual*. Taito, 1978. [Online]. Available: https://oldgamesdownload.com/wp-content/uploads/SpaceInvaders_Manual_Arcade_EN.pdf
- [5] G. ARCHIVE, “Space invaders 1978 - arcade gameplay,” 2015. [Online]. Available: <https://www.youtube.com/watch?v=MU4psw3ccUI>
- [6] Doxygen, “interrupts.h file reference,” 2020. [Online]. Available: https://libnds.devkitpro.org/interrupts_8h.html
- [7] “C - operators,” 2021. [Online]. Available: https://www.tutorialspoint.com/cprogramming/c_operators.htm
- [8] “Gba image transmogrifier (.grit),” Nov 29, 2008. [Online]. Available: <https://www.coranac.com/man/grit/html/grit.htm>
- [9] “2d collision detection,” 2021. [Online]. Available: https://developer.mozilla.org/es/docs/Games/Techniques/2D_collision_detection
- [10] M. Lage Junior, M. Godinho Filho *et al.*, “Variations of the kanban system: Literature review and classification,” *International Journal of Production Economics*, vol. 125, no. 1, pp. 13–21, 2010.