Estructura de Computadores
Facultad de Ingeniería Informática
UPV/EHU

Memoria

StayAtHome-NDS

Iyán Álvarez
Aleina Pelipian
Unai Roa

5 de mayo de 2020

Índice general

1 Introducción **1**

2 Resumen **2**

3 Conceptos generales **3**

3.1 Entrada de datos . . . 3

3.1.1 Sincronización por encuesta . . . 3

3.1.2 Sincronización por interrupción . . . 3

3.2 Gráficos . . . 4

3.2.1 Fondos . . . 4

3.2.2 Sprites . . . 6

3.3 Temporizadores . . . 8

4 Desarrollo del proyecto **9**

4.1 Entorno de desarrollo . . . 9

4.2 Control de versiones . . . 9

4.3 Trabajo en grupo . . . 10

5 Resultados **11**

5.1 Descripción de estados . . . 11

5.2 Estructura de datos . . . 15

5.2.1 Estado . . . 15

5.2.2 Objetos . . . 16

5.3 Programa principal . . . 17

5.3.1 Inicialización . . . 17

5.3.2 Juego . . . 18

5.3.3 Finalización . . . 19

6 Conclusión **20**

1. Introducción

Este proyecto consiste en implementar un juego muy sencillo para la consola Nintendo DS. El objetivo del mismo es manejar los conceptos estudiados en el Tema 3 de Entrada/Salida de la asignatura Estructura de Computadores impartida el primer año de Ingeniería Informática en la UPV/EHU. Este juego debe llevar a cabo el control de sus periféricos mediante los dos tipos de sincronización estudiados: encuesta e interrupción. Debe hacer uso, al menos, del teclado y de la pantalla táctil y debe realizar algún tipo de control de temporizadores.

La Nintendo DS ha sido la consola que ha marcado la infancia de una generación. A diferencia de la gran mayoría de consolas, esta no tiene un sistema operativo. Esto nos da la opción de trabajar a un nivel más bajo donde, por ejemplo, podemos acceder a la información de los registros.

El objetivo principal de este trabajo es realizar un juego para la Nintendo DS, donde trabajaremos a bajo nivel con el lenguaje de programación C.

Esta consola dispone de dos procesadores, un ARM7 y un ARM9. Cada uno de estos se ocupa de la entrada y salida de determinados periféricos. Ya que la comunicación entre estos procesadores es un apartado complejo para tratar a estas alturas de nuestra formación, hemos utilizado la librería *libnds* y trabajaremos con el control de un único procesador.

Figura 1.1: Consola Nintendo DS

2. Resumen

Este proyecto se ha realizado con el objetivo de poner en práctica diferentes conocimientos sobre el subsistema de E/S de la consola Nintendo DS. Para ello, se han ido adquiriendo dichos conocimientos mediante las prácticas y laboratorios realizados durante el curso. Este proyecto es la prueba de ello, donde hemos utilizado y aplicado los temas aprendidos durante la asignatura.

Nuestro proyecto esta basado en la crisis sanitaria que estamos viviendo a día de hoy. Está basado en 4 niveles determinados por tiempo, es decir cada uno tiene una duración de 60 segundos y el objetivo es aguantar el mayor tiempo posible y eliminar al mayor número de virus.

Otro objetivo de este trabajo era fomentar la comunicación y el trabajo en grupo a la hora de desarrollar el proyecto y, por otro lado, superar las dificultades con las que podamos encontrarnos en este proceso.

Además, hemos trabajado con la herramienta Git y la plataforma GitHub, la cual permite trabajar a varios individuos con un mismo proyecto de manera sencilla y eficaz. Consiste en ir añadiendo al proyecto principal apartados concretos una vez comprobado su funcionamiento, de manera que dicho proyecto va tomando forma de una manera escalonada y ordenada.

3. Conceptos generales

3.1 Entrada de datos

Para el buen funcionamiento del juego, hemos decidido realizar sincronización del teclado utilizando ciertas teclas por encuesta y otras por interrupción. El funcionamiento de la pantalla se realiza mediante encuesta.

3.1.1 Sincronización por encuesta

La sincronización por encuesta se basa en la consulta continua o periódica sobre los registros del controlador del periférico, para saber si este se encuentra o no disponible.

- **Pantalla táctil:** para comenzar a jugar e imprimir las instrucciones.
- **Tecla <UP>:** para dirigir el movimiento del desinfectante hacia arriba.
- **Tecla <SELECT>:** para finalizar el juego y apagar la consola.

3.1.2 Sincronización por interrupción

La sincronización por interrupción es un suceso asíncrono, es decir, la CPU es interrumpida en cuanto se detecta una señal externa y así estará lista para iniciar la transferencia de datos.

- **Tecla <DOWN>:** para dirigir el movimiento del desinfectante hacia abajo.
- **Tecla <A>:** para disparar las gotas de desinfectante.
- **Tecla <START>:** para activar y desactivar el modo pausa y reiniciar la partida.

3.2 Gráficos

La Nintendo DS dispone de dos pantallas gráficas de tipo LCD, de dimensiones 256x192 píxeles, que se pueden aprovechar para mostrar tanto fondos, como sprites, etc. Para cada implementación de gráfico, dicha imagen será traducida a contenido binario y trasladado al banco de memoria VRAM.

En nuestro caso, vamos a utilizar la pantalla superior para realizar trazas y la inferior para mostrar fondos y sprites. De esta manera, podemos controlar y tener información del estado de algunas variables del juego y mostrar información relevante para el jugador, al mismo tiempo que ofrecemos una buen experiencia de juego. La funcionalidad de trazado también nos ha permitido 'debuggear' el juego y comprobar el correcto funcionamiento de determinadas funciones.

3.2.1 Fondos

Dado nuestro proyecto y la idea principal, los gráficos se tratan de un entorno basado en la crisis sanitaria que estamos sufriendo a día de hoy. El fin es proteger a cinco personas que se encuentran en la calle predispuestas al virus COVID-19. Para esto, hemos creado dos fondos de pantalla adecuados.

- **Inicio:** Una vez abierto el juego, nos aparecerá en pantalla un fondo de inicio (figura 3.1), el cual nos indicará las opciones que tenemos antes de iniciar la partida. Nos podemos informar sobre las instrucciones del juego o bien empezar la partida directamente, con tan sólo un toque en la pantalla.

Figura 3.1: Fondo Inicio

- **Calle:** Dada la situación actual y el entorno del juego, como fondo del mismo hemos decidido representar el medio más adecuado: una calle vacía (figura 3.2), donde las señales indican el estado de alarma y el riesgo al que se están exponiendo las personas al encontrarse en un espacio abierto.

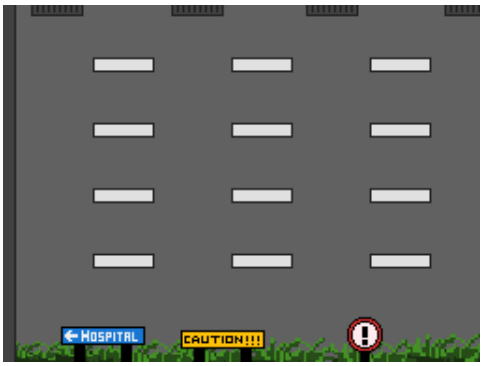

Figura 3.2: Fondo Tráfico

- **Fondo negro:** Una vez finalizada la partida y no queremos volver a jugar, se mostrará un fondo negro (figura 3.3) simulando que hemos apagado la consola.

Figura 3.3: Fondo negro

3.2.2 Sprites

Los sprites son una de las partes más importantes del juego, ya que podemos interactuar con ellos y ejecutar diferentes acciones tanto usando la entrada/salida de la Nintendo DS o automatizando determinados procesos.

En nuestro caso, hemos utilizado sprites de 16x16, ya que debido al entorno de desarrollo y herramientas utilizadas no podíamos hacer sprites de mayor tamaño.

Aun así, tuvimos la necesidad de diferenciar en cuestiones de tamaño a una persona de un desinfectante, por lo que para generar el sprite de la persona utilizamos dos sprites 16x16, uno encima del otro. De esta manera, logramos simular un único sprite aunque a la hora de programar y ejecutar determinadas acciones hay que tener en cuenta que son dos diferentes, por lo tanto cada uno ocupa un espacio de memoria distinto en la Nintendo DS.

- **Desinfectante:** Este sprite juega el papel más importante de la partida, ya que evitará que los humanos sean infectados por los virus. El desinfectante podrá ser dirigido por el jugador utilizando las teclas <ARRIBA> y <ABAJO>. Además, al pulsar la tecla <A> disparará una gota que tiene la capacidad de matar a un virus.

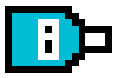

Figura 3.4: Desinfectante

- **Gota:** Las gotas tendrán una dirección fija, horizontal, hacía la derecha. Son lanzadas por el desinfectante y tienen una velocidad de recarga de medio segundo, por lo que necesitaremos una mayor precisión a la hora de combatir los virus.

Figura 3.5: Gota

- **Persona:** Las personas tendrán dos estados: normal e infectada. Al comienzo de la partida, todas las personas están sanas, pero cuando un virus impacte a una persona esta se infectará. Cuando todas las personas estén infectadas, el juego se dará por finalizado, perdiendo la partida.

Figura 3.6: Persona sana

Figura 3.7: Persona infectada

- **Virus:** Si un virus alcanza a una persona, la infectará. En cambio, si una gota alcanza un virus, lo eliminará.

  Dependiendo de la dificultad, los virus tendrán una velocidad y una trayectoria distintas. Esta cambia cada 60 segundos hasta llegar a Extrema, que se mantiene hasta perder.

  - Fácil: estos aparecerán cada 2 segundos y la trayectoria será recta y horizontal, hacía la izquierda.
  - Media: estos aparecerán cada 2 segundos y la trayectoria será aleatoria, es decir, se moverán ligeramente hacia arriba y abajo.
  - Difícil: estos aparecerán cada segundo y la trayectoria será recta y horizontal, hacía la izquierda.
  - Extrema: estos aparecerán cada segundo y la trayectoria será aleatoria, es decir, se moverán ligeramente hacia arriba y abajo.

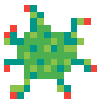

Figura 3.8: Virus

3.3 Temporizadores

Los temporizadores de la Nintendo DS son contadores que funcionan a una frecuencia dada y se utilizan para controlar el tiempo. Cada procesador de la Nintendo DS dispone de cuatro temporizadores. Estos generan un número de ticks determinados en un segundo. Para poder estipular esto en la Nintendo DS, tendremos que aplicar la siguiente fórmula:

$$latch = 65536 - \frac{1}{frec_{interrupcion}} frec_{cuenta}$$

Estos datos los guardaremos así:

- TIMERx_DAT = $frec_{interrupcion}$ (formato hexadecimal)
- TIMERx_DAT = $frec_{cuenta}$ (formato decimal).

En nuestro proyecto hemos decidido utilizar dos temporizadores.

TIMER0 (60 ticks/seg) El temporizador TIMER0 funciona a 60 ticks por segundo y desde él controlamos el funcionamiento principal del juego. Está dividido en dos partes: en las llamadas cada tick y en las llamadas cada segundo.

Nos parecía importante actualizar la información de una manera óptima y recibir un feedback inmediato en las tareas visuales, por eso estas se hacen cada tick. Por ejemplo, actualizar la posición del Spray *updateSpray()* o de los viruses *updateVirus()*, actualizar la información de los virus que se han matado en la partida *printVirusKilled()* y detectar el final del juego *detectGameFinish()*. En cambio otras como calcular la dificultad *calculateDifficulty()* o imprimir los segundos *printSegs()* se realizan cada segundo (60 ticks).

Esta es la información de TIMER0_CNT = 0x00C1 y TIMER0_DAT = 56798.

TIMER1 (100 ticks/seg) El temporizador TIMER1 funciona a 100 ticks por segundo y desde el controlamos también el funcionamiento principal del juego. A diferencia del anterior, este no esta dividido en dos partes. Su único papel es el de actualizar las coordenadas y el estado de las gotas (updateDrop()).

Podríamos haber utilizado un único temporizador, pero nos parecía necesario para la dinámica del juego que las velocidades de movimiento fueran diferentes. Debido a esta necesidad decidimos implementar este temporizador.

Esta es la información de TIMER1_CNT = 0x00C1 y TIMER1_DAT = 60293.

4. Desarrollo del proyecto

4.1 Entorno de desarrollo

El código fuente del juego ha sido realizado en el entorno de Ubuntu, un sistema operativo de software libre y código abierto. Esta ha sido la primera toma de contacto con este sistema operativo para nosotros. Hemos aprendido operaciones básicas con el terminal (*ls, cd, rm, apt get,...*), por ejemplo, para compilar, simular o hacer commits. Para compilar nuestro juego utilizábamos el comando *make clean; make*, esto eliminaba los archivos de la última compilación del juego y creaba los nuevos cuando no había ningún problema en el código; en el caso de haber errores nos lo mostraba. Para simular el juego, hemos utilizado el emulador NO$GBA, este es un emulador diseñado para Windows, pero como nosotros hemos trabajado en Ubuntu principalmente hemos tenido que utilizar el programa Wine para poder ejecutarlo. Esto nos ha dado algunos problemas ya que iba a velocidades muy rápidas o no era del todo estable. Hemos intentado utilizar Windows para probar el juego y por eso recomendamos encarecidamente su ejecución en Windows utilizando el emulador NO$GBA (figura 4.1):

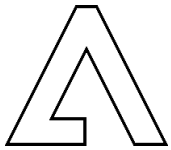

Figura 4.1: NO$GBA

4.2 Control de versiones

Para un mejor rendimiento del proyecto y una mejor organización a la hora de su desarrollo, hemos estado utilizando el entorno Git, un software de control de versiones. Su objetivo es facilitar el mantenimiento de un proyecto, coordinando el trabajo que varias personas realizan sobre archivos compartidos.

Hemos utilizado diferentes ramas *git branch* para trabajar en la implementación de diferentes funcionalidades y solventar diferentes bugs.
Tras cada cambio en el código realizábamos un *git commit -m "mensaje"*, para que quedara constancia de los cambios realizados junto a una pequeña descripción de estos. Una vez hecho esto y los cambios eran definitivos utilizábamos GitHub para hacer un *merge* entre la rama de la implementación y master.

4.3 Trabajo en grupo

Para la realización de este proyecto en grupo nos hemos ayudado de dos grandes apoyos que son GitHub y Discord.

GitHub se trata de una plataforma que permite alojar proyectos utilizando el sistema de control de versiones Git. Lo hemos utilizado para trabajar con diferentes ramas del trabajo (*virus, spray, drop, typedefs, pause...*) creando el código fuente correspondiente a cada una. Ha sido de gran ayuda y comodidad puesto que cada uno podía centrarse y trabajar en un apartado concreto y una vez comprobado su correcto funcionamiento, lo añadíamos a la versión master, a la que todos teníamos acceso. Además siempre podíamos acceder a lo que se estaba trabajando en otras ramas diferentes a las nuestras ya que una vez hechos los commits pertinentes hacíamos *git push* y todo el contenido de nuestra rama se subía a la plataforma.

Figura 4.2: GitHub

Discord se trata de una aplicación diseñada para la comunicación por audio y vídeo entre varios integrantes. Hemos utilizado mucho esta plataforma para comunicarnos, nos ha permitido tanto hacer llamadas, vídeo-llamadas, pasar archivos o incluso compartir la pantalla para que los demás puedan ver lo que se esta haciendo en streaming.

Figura 4.3: Discord

Gracias a estas aplicaciones y herramientas hemos logrado ayudarnos estando en nuestras propias casas y lograr los objetivos del proyecto.

5. Resultados

5.1 Descripción de estados

Estado INIT: en este estado se espera la decisión del jugador. Dando un click en pantalla, podemos empezar a jugar, o bien ver las instrucciones del juego, tal y como se puede ver en la siguiente captura (figura 5.1):

Si seleccionamos el recuadro *PLAY* en nuestra pantalla inferior, el juego comenzará y podremos empezar a defender a las personas del temido COVID-19.

Figura 5.1: Pantalla inferior INIT

En cambio, si es la primera vez que jugamos o no recordamos cómo era la dinámica del juego, podemos tocar en el recuadro *INSTRUCTIONS* de la pantalla inferior de la Nintendo DS. Una vez activado, se mostrarán por pantalla las instrucciones y dinámica del juego.

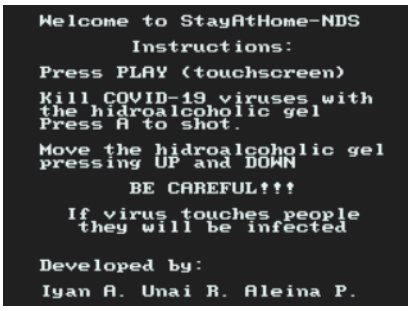

Figura 5.2: Pantalla superior instrucciones

Estado GAME: una vez seleccionado *PLAY* en la pantalla de INIT, aparecerán tanto las personas, como el bote desinfectante y los viruses comenzaran a spawnearse (figura 5.3).

El objetivo del juego es proteger a las 5 personas del virus COVID-19.

Podremos mover nuestro desinfectante hacia arriba y abajo pulsando las teclas <ARRIBA> y <ABAJO> respectivamente. En este estado, también podemos lanzar gotas desinfectantes hacía los viruses, pulsando la tecla <A>, estas tendrán un tiempo de recarga de medio segundo. Como se puede observar en la figura 5.3, al ser infectadas, las personas empezarán a encontrarse mal y el color de su piel pasará a ser verde claro. Para poner el juego en modo PAUSE, habrá que tocar la tecla <START>.

Figura 5.3: Pantalla juego

Estado PAUSE: una vez estamos en el estado PAUSE, el juego se detendrá. En pantalla aparecerá un mensaje informativo, para poder volver a jugar (figura 5.4):

Figura 5.4: Pantalla pausa

Estado RESTART: cuando las cinco personas han sido infectadas, el juego entra en modo RESTART. Desde aquí, podemos decidir si vamos a reiniciar el juego, pulsando la tecla <START> volviendo al estado GAME, o acabar la partida, pulsando la tecla <SELECT> que entrará en el estado END y simulará el apagado de la consola.

Figura 5.5: Pantalla reinicio

Estado END: una vez acabada la partida y pulsada la tecla <SELECT>, la consola se apagará.

Figura 5.6: Consola apagada

5.2 Estructura de datos

A la hora de programar nos dimos cuenta que para el manejo y control de varios estados y objetos era necesaria la creación de estructuras de datos, ya que utilizábamos una gran cantidad de variables globales y esto dificultaba el mantenimiento y la comprensión de los algoritmos. Para lograr esto hemos utilizado la instrucción *typedf struct* del lenguaje de programación C.

Basándonos en la idea que teníamos decidimos centralizar todas estar variables en dos variables globales: Estado y Objetos.

5.2.1 Estado

Dentro de la variable Estado de tipo "tEstado"guardamos la información referente al estado actual del juego:

- **estado:** Se guarda el estado acutal del juego: INIT, GAME, PAUSE...
- **segs0:** Se guardan los segundos transcurridos desde el inicio del juego, esta labor la hace el temporizador TIMER0.
- **numVirus:** Se guarda el índice del último virus spawneado del array Virus[numVirusT].
- **numGota:** Se guarda el índice de la última gota spawneada del array Gota[numGotaT].
- **dificultad:** Se guarda la dificultad del juego referente al tiempo transcurrido desde el inicio del juego.
- **viruskilled:** Se guarda el número de virus totales que se han eliminado en la partida.
- **infectedpeople:** Se guarda el número de personas infectadas en la partida.
- **canshot:** Se guarda si el Spray esta listo para disparar o no.
- **initdone:** Se guarda si se han hecho las instrucciones de inicio o no.
- **instructdone:** Se guarda si se han mostrado por pantalla las instrucciones del juego o no.
- **restartdone:** Se guarda si se ha reiniciado el juego, es decir si una vez que hemos perdido una partida volvemos a jugar o no.

5.2.2 Objetos

- **Spray:** Representa una variable de tipo tSpray, en esta se guardan las coordenadas 'x' e 'y' del objeto, y la dirección en la que se esta moviendo.
- **Virus[numVirusT]:** Representa un array de objetos "tVirus", su cantidad es la especificada en la variable "numVirusT", nosotros la hemos fijado a 6, ya que los virus se reutilizan. Por otro lado en el tipo "tVirus"guardamos las coordenadas x e y del objeto, si es visible o no y el índice de memoria.
- **Gota[numGotaT]:** Representa un array de objetos "tGota", su cantidad es la especificada en la variable "numGotaT", nosotros la hemos fijado a 8, ya que las gotas, al igual que los virus se reutilizan. Por otro lado en el tipo "tGota"guardamos las coordenadas x e y del objeto, si es visible o no y el índice de memoria.
- **Persona[numPersonaT]:** Representa un array de objetos "tPersona", su cantidad es la especificada en la variable "numPersonaT", nosotros la hemos fijado a 5, en este caso esta no se puede modificar ya que las personas no quedarían centradas o se saldrían de la pantalla. Por otro lado en el tipo "tPersona"guardamos las coordenadas x e y del objeto, si esta infectada o no y como es un sprite 32x16 necesitamos guardar dos índices de la memoria, uno para la parte superior de la persona y otro para la inferior.

5.3 Programa principal

Una vez explicados los conceptos básicos que hemos desarrollado en el juego, vamos a explicar las secciones del programa principal y los resultados obtenidos:

5.3.1 Inicialización

En primer lugar, abrimos la consola e iniciamos el juego, encontrándonos en el estado inicial, INIT. En este instante, observamos en la pantalla un recuadro donde pone *PLAY*, donde comenzaríamos el juego, pasando al estado GAME; además de otro recuadro más pequeño donde pone *INSTRUCTIONS*, el cual nos llevará a una breve explicación de como jugar.

A continuación se muestra la parte del autómata de esta primera parte del proyecto, donde visualizamos los dos estados mencionados previamente, junto con alguna instrucción:

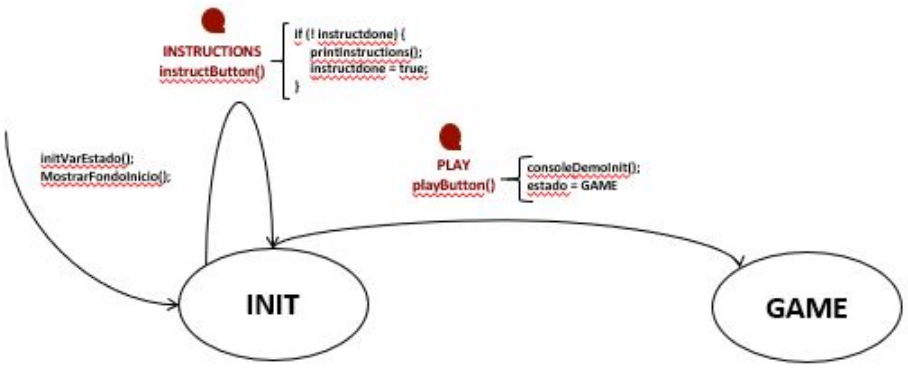

Figura 5.7: Inicialización del juego

5.3.2 Juego

Por lo tanto, pulsando el recuadro que contiene *PLAY* pasamos al siguiente estado, GAME. En este transcurso se mostrará el fondo del juego (*printGameScreen()*) con las personas y el desinfectante; y se inicializaran las variables (*initVarGameScreen()*). Habiendo comenzado la partida, tendremos varias opciones plasmadas en la siguiente imagen que explicaremos a continuación:

Durante la partida, en la pantalla superior vemos un temporizador que nos marca el transcurso del tiempo. En cuanto a las opciones para jugar, está la flecha hacia arriba (<ARRIBA>), que va por encuesta (recordamos que ésto significa que el procesador consulta los registros del controlador del periférico viendo así la disponibilidad del mismo); y por otra parte, la flecha hacia abajo (<ABAJO>), que va por interrupción (recordamos que en este caso el procesador es avisado por el controlador del periférico sobre la disponibilidad del mismo). Ambos botones realizan la misma operación (mover el spray a una velocidad constante), pero lo hacen en direcciones opuestas.

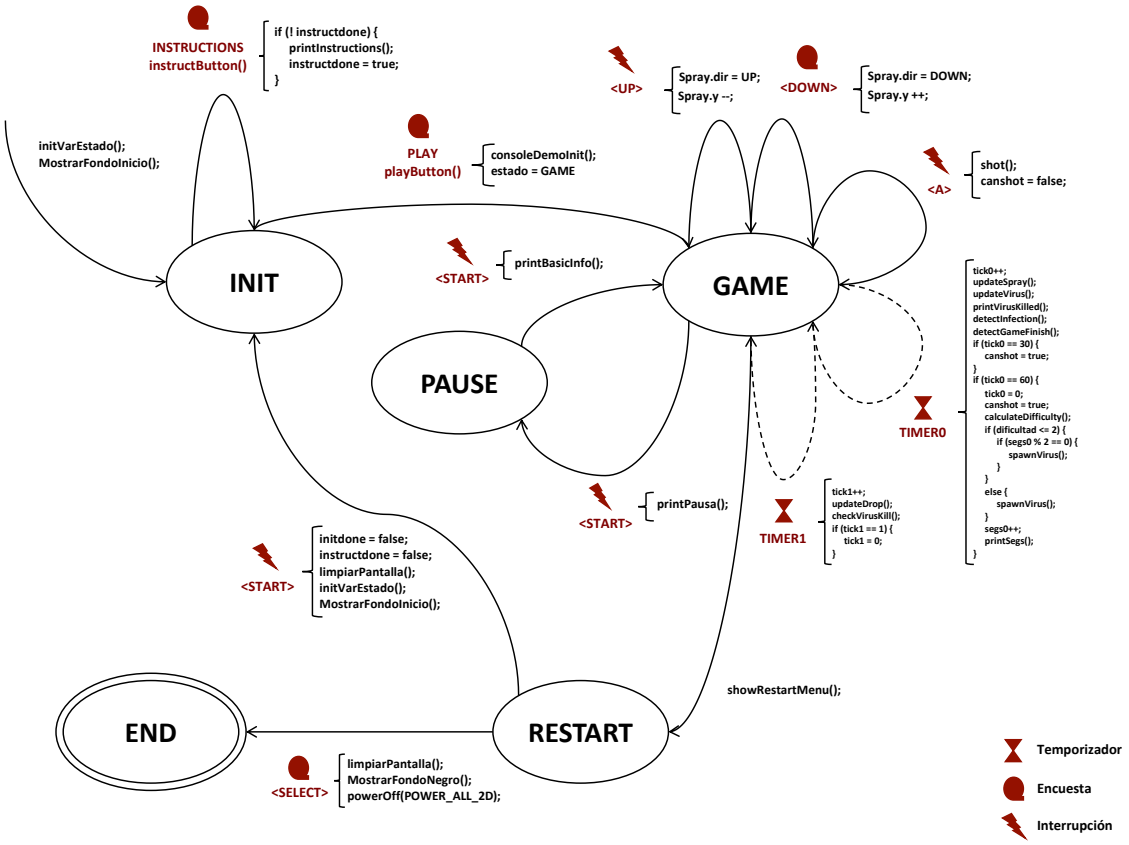

Figura 5.8: Autómata del juego

Por otra parte, tenemos la opción de pausar el juego pulsando la tecla <START>, esto cambiaría el estado a PAUSE. De esta manera, se detendrán los virus y no podremos seguir jugando hasta volver a pulsar la tecla <START>, que cambiará el estado del juego de nuevo a GAME.

Otro de los estados disponibles en este juego es RESTART, al cual se pasará una vez estando todas las personas infectadas. Para ello, los viruses deberán de moverse hasta ellas sin tocar ninguna de las gotas disparadas por el spray. Cuando esto suceda se activara la función (*detectInfection()*). En consecuencia, en el momento que se detecten estas circunstancias, el estado del juego pasará a ser RESTART. A su vez, el objetivo del juego es evitar esto. Por ello, el desinfectante disparará dichas gotas mediante la tecla <A>(*shot()*). En el momento que estas gotas contacten con los viruses, éstos se eliminarán (*checkVirusKill()*)).

5.3.3 Finalización

Una vez que el juego a detectado la muerte de todas las personas, se pasará al estado RESTART, desde este tendremos dos opciones. La primera, si pulsamos la tecla <START>, volveríamos a jugar nuevamente. Esto cambiaría el estado a INIT.
Por otro lado, en el estado RESTART también tenemos la opción de finalizar el juego totalmente (*detectGameFinish()*)) y apagar la consola pulsando la tecla <SELECT>. Una vez aquí, no podemos volver a jugar hasta volver a iniciar la consola.

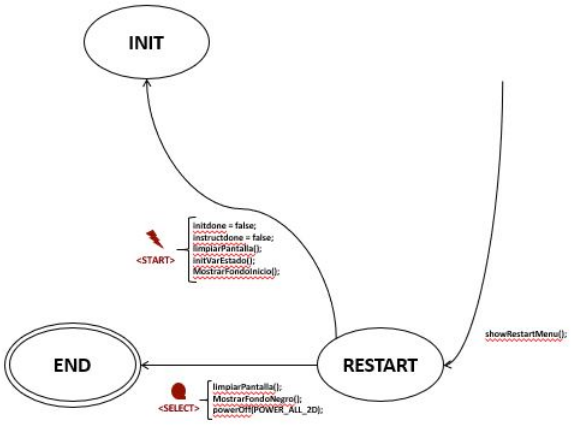

Figura 5.9: Reinicio y final del juego

6. Conclusión

En un principio, cuando nos introdujeron el proyecto, nos pareció bastante complejo. Esto se debía a que no solo teníamos que aprender a programar en C y conocer el sistema de entrada y salida de la Nintendo DS, sino que también tuvimos que introducirnos en un sistema operativo nuevo para todos, Ubuntu, y adaptarnos a las circunstancias de desarrollar el proyecto de manera conjunta confinados en nuestras casas.

A medida que pasó el tiempo y fuimos realizando laboratorios de diferentes conceptos, poco a poco desarrollamos nuestros conocimientos y viendo pequeños avances fuimos animándonos cada vez más. Gracias a esto hemos disfrutado con la realización del juego y nos ha quedado como esperábamos.

En general, nos ha parecido un proyecto completo en el cuál hemos aprendido mucho.