Universidad de Valladolid

EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE

Alumno: Jorge García Flores

Tutor: Fernando Díaz Gómez

Agradecimientos

Ante todo quisiera agradecer el apoyo mostrado por mi familia para la realización y consecución de esta titulación, con el consiguiente aguante que han tenido que tener. Segundo, a mis amigos, en especial a Víctor Daniel Santos Gonzálvez, David Bravo y a Luis Miguel Martín Rodríguez. Además quisiera acordarme de ese amigo que me dio un poco de sabiduría en el mundo de la informática que es Sergio Ortega (este hombre es una fiera tanto en sistemas como en programación). También a mi tutor de Proyecto de Fin de Carrera, Fernando Díaz Gómez por toda su ayuda y que me ha asombrado con lo que puede llegar a saber sobre Informática en general. Y por último, aunque no por ello menos importante, a esos grandes compañeros que he conocido en la Escuela de Informática de Segovia y de lo cual estoy muy orgulloso de poder decir que son nuevas amistades para mí (Fátima Serna García, Yésica Valle, Manuel Rico López, etc.). Muchas gracias a todos vosotros.

Cualquier sueño que merezca soñarse, es un sueño por el que luchar.

Índice

PARTE 1:

INTRODUCCIÓN AL SISTEMA

1.- INTRODUCCIÓN

1.1.- IDENTIFICACIÓN DEL PROYECTO

Título: EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE

Autor: Jorge García Flores

Tutor: Fernando Díaz Gómez

Departamento: Informática

Área: Ciencias de la Computación e Inteligencia Artificial

1.2.- ORGANIZACIÓN DE LA MEMORIA

La documentación del presente proyecto se va estructurar basándose en tres partes diferenciadas. Dichas partes serán Introducción al Sistema, Desarrollo del Sistema y Conclusiones, junto con una documentación del usuario (compuesto por el Manual de instalación y el Manual del usuario). Se justifica esta organización, ya que cada parte o sección tiene unos presentes (en este caso una audiencia) que es claramente diferenciada para cada caso (público en general, personal técnico –programadores, desarrolladores, diseñadores- y usuarios de la aplicación desarrollada), representándose como un documento con entidad propia, compuesto por otros documentos con un índice propio a ellos.

La Parte 1 muestra los detalles del Proyecto a los usuarios generales de la aplicación, así como los costes de dicho proyecto en forma de presupuesto con los gastos que ha ocasionado.

- ✓ Una descripción general del Proyecto Fin de Carrera (PFC), tratando los objetivos que se han perseguido por dicho proyecto.
- ✓ Inclusión de una interfaz de usuario (IU), en el que se incluye una serie de diseños de las pantallas preliminares,

así como un diseño de alto nivel de abstracción de la interfaz con sus explicaciones detalladas.

- ✓ Se incluye una información detallada de las funcionalidades realizadas en el Proyecto Fin de Carrera.
- ✓ Se describirán una serie de cuestiones consideradas como importantes en las que se explicarán una serie de nomenclaturas utilizadas.
- ✓ Explicación de la metodología usada para el Proyecto Fin de Carrera desarrollado.
- ✓ Descripción de la tecnología empleada para la realización del proyecto.
- ✓ Estudio económico del proyecto, en el cual se hará un presupuesto general de gastos incurridos en dicho proyecto, explicándose de forma detallada las fases del proyecto que se han realizado, así como una descripción de los recursos humanos y materiales utilizados en cada fase del proyecto.

La Parte 2 del Proyecto Fin de Carrera aporta los detalles que se creen suficientes y necesarios, pero perfectamente explicados y detallados a los programadores, diseñadores y desarrolladores. Esto es debido para que se realice una alta comprensión del diseño e implementación de la aplicación desarrollada, todo ello desde un análisis riguroso, aceptado y supervisado hasta el más mínimo detalle.

- ✓ Análisis del Sistema
    - o Se incluye una información detallada y sesgada del modelo estructurado para la aplicación usando para ello Diagramas de Paquetes y Diagramas de Estructura, explicándose el por qué se han utilizado éstos y no otros y justificándolo a su vez.
- ✓ Diseño del Sistema

- En este apartado se incluye el diseño de la arquitectura del sistema elegida, una explicación de por qué no se ha diseñado e implementado una base de datos para la aplicación, el estudio del entorno de desarrollo y explotación de la aplicación, así como una justificación de las tecnologías utilizadas en el desarrollo del Proyecto Fin de Carrera.

✓ Diagrama de Requisitos del Sistema

- Inclusión detallada de los requisitos de información, requisitos funcionales, casos de uso, requisitos no funcionales, definición de actores, así como una matriz de rastreabilidad, un glosario de términos y un índice de tablas para un perfecto conocimiento de los desarrollado.

✓ Pruebas del Sistema

- Se incluye una batería de pruebas realizada tanto con caja negra como caja blanca. Además se incluye en qué equipo se han realizado las pruebas de instalación de la aplicación.

Por último, en la Parte 3 mostramos una serie de conclusiones que se han obtenido una vez realizado el Proyecto Fin de Carrera, todo ello de forma totalmente detallada.

✓ Se realiza una evaluación de la aplicación dependiendo del rendimiento y la robustez del producto realizado en el Proyecto Fin de Carrera, todo ello explicado de forma amplia.

✓ Valoración de los objetivos iniciales que se han detallado en la Parte 1 y consecución de ellos mismos en el desarrollo del Proyecto Fin de Carrera.

✓ Explicación de los nuevos conocimientos adquiridos al realizar el Proyecto Fin de Carrera en general, y en particular con la aplicación desarrollada.

- ✓ Detalle sesgado de posibles ampliaciones del producto para nuevas versiones.

- ✓ Relación de la Bibliografía utilizada en el desarrollo del Proyecto Fin de Carrera.

Añadido a estas tres partes en las que se ha desglosado la documentación presente, se encuentran los Apéndices del Manual de instalación y el Manual de usuario. El fin de esta sección es formar al usuario de la aplicación desarrollada en su uso presentándose de forma totalmente intuitiva, al ser una guía gráfica (How to). Dichos manuales ya se han explicado cuáles son, pero de forma general:

- ✓ El Manual de instalación se detalla para explicar la instalación de la aplicación, así como los programas y equipo necesario para su instalación y correcto funcionamiento.

- ✓ El Manual de usuario describe todas las funcionalidades que ofrece la aplicación desarrollada en el Proyecto Fin de Carrera, así como una explicación de ellas.

2.- DESCRIPCIÓN GENERAL

En este segundo apartado de la Parte 1 se va a detallar la motivación por la cual se ha realizado el presente Proyecto Fin de Carrera, los objetivos planteados y funcionalidades del producto en forma de tabla [**APUNTES INGENIERÍA DEL SOFTWARE 1 – FRANCISCO JOSÉ GONZÁLEZ CABRERA**], descripción del producto, desglosado a su vez en el diseño de la interfaz de usuario y el diseño de la estructura de la aplicación, una descripción de la nomenclatura utilizada debido a que hay una serie de palabras específicas que se deben explicar, una descripción del entorno de ejecución de la aplicación con el hardware que se necesita, así como los programas necesitas para su correcto funcionamiento y ejecución, la tecnología empleada para desarrollar el Proyecto Fin de Carrera, qué

metodología se ha usado en el desarrollo del presente proyecto y un presupuesto totalmente detallado de dicho proyecto.

- Se pretende crear una aplicación para la videoconsola *PlayStation* Portable de *Sony* que ejecute un emulador de la videoconsola *Master System 2* de *Sega*.

- Se pretende crear para dicha aplicación un contenedor que mostrará una serie de funcionalidades que se describirán más adelante.

- Dicho contenedor permitirá la visualización de los videojuegos a los que el usuario jugará, así como cambiar una serie de parámetros, tanto en la visualización, como en el sonido, así como el rendimiento de la *PlayStation Portable* podrá ser cambiado.

- El contenedor no será otra cosa más que una especie de ventana a modo de ventana como las de los sistemas operativos actuales en las que habrá un menú con una serie de opciones, y a partir de dichas opciones ocurrirán unas cosas, u otras.

- Se crearán todas las herramientas posibles para que un usuario de forma totalmente intuitiva pueda cambiar los parámetros de configuración y rendimiento, así como que pueda cambiar de juego sin necesidad de apagar la videoconsola *PlayStation Portable*, ni tenga que salir de la videoconsola emulada *Master System 2*.

- Se podrán añadir todos los juegos que el usuario quiera, pero atendiendo a tres aspectos, que los videojuegos se encuentren comprimidos (archivos con extensión *.rar* o *.zip*), que los videojuegos sean para la videoconsola *Master System 2* y que exista una tarjeta de memoria *Memory Stick* que permita el almacenamiento de dichos videojuegos sin que la memoria se haya agotado.

- Se creará la infraestructura necesaria para que los videojuegos que permitan un juego a dos jugadores, puedan jugar mediante conexión a internet con la metodología ad-hoc a dicho videojuego con dos videoconsolas PlayStation Portable.

2.1.- MOTIVACIÓN

El proyecto realizado es debido a que cada vez más, en el ámbito lúdico, los usuarios demandan aplicaciones con las que poder recrear sus computadoras de años atrás, en el aspecto de que son computadoras de generaciones anteriores, siendo conocida como generación anterior una computadora con menor capacidad de procesamiento (según los bits del registro) con la que poder pasar unos momentos de ocio. Todo esto ha provocado una gran escena de desarrollo de aplicaciones en dicho ámbito conocida como *homebrew*, que no es otra que el desarrollo de aplicaciones para videoconsolas de la actual generación (por ejemplo *Nintendo 3ds*, *PlayStation Portable*, *PlayStation 3*, *XBOX 360*, *Nintendo Wii*, etc.), aunque también se realizan para videoconsolas de generaciones anteriores (*NES*, *Master System 2*, *Mega Drive*) e incluso ordenadores de 8-16 bits (*Commodore 64*, *MSX*, *Amstrad*, *Spectrum*, *Amiga*, etc.) de forma independiente a las grandes empresas y de una forma totalmente altruista para que usuarios de dicho hardware, puedan disfrutar de nuevas aplicaciones totalmente distintas para lo que están diseñadas estas máquinas. Los objetivos fundamentales pueden ser distintos, ya que aunque sean máquinas de propósito específico, cada vez más se van acercando a máquinas de propósito general y ya se pueden visualizar imágenes, fichero *.pdf*, escuchar música o visualizar películas. En el presente proyecto, el objetivo fundamental es acercar a los usuarios más jóvenes, la forma en que los que tenemos más edad podíamos divertirnos en breves períodos de tiempo con unas máquinas mucho más simples que ahora, por lo cual se desarrolla un emulador con un contenedor que hará las veces de menú y visualización para poder jugar a videojuegos realizados para la máquina a emular.

Es por tanto, que ya que al realizar un emulador de una computadora, eligiera para ello la primera videoconsola que tuve, y que fue la *Master System 2* de *Sega*. De sus detalles se hablará más adelante para explicar su funcionamiento al igual de qué se compone dicha máquina.

Otra de las motivaciones es que quería probarme a mí mismo para realizar un proyecto de esta envergadura, teniendo en cuenta que no es lo que se suele hacer como proyecto en la **Escuela de Ingeniería Técnica de Informática de Gestión** cuando un alumno propone un proyecto a un tutor. Siempre me ha gustado intentar superar mis límites y ver de lo que soy capaz de hacer.

2.2.- OBJETIVOS

El objetivo del proyecto a realizar es la construcción de un emulador de la videoconsola *Master System 2* de la empresa *Sega*, la cual fue lanzada al mercado en el año 1985, para poder ser ejecutada en una videoconsola *PlayStation Portable* (*PSP*) de la empresa Sony, la cual fue puesta a la venta en el año 2004.

Como consecuencia de un análisis concienzudo, se tienen los siguientes objetivos concretos:

- Ejecutar el emulador de la máquina *Master System 2* con la videoconsola *PSP* (*PlayStation Portable*) desde una tarjeta de memoria *Memory Stick*.

- Ejecutar los videojuegos disponibles de dicha videoconsola mediante emulación, realizando la carga de ellos. Dichos juegos se guardarán también en la *Memory Stick* en formato *.rar*.

- Cambiar distintos parámetros en las opciones del emulador como puede ser la frecuencia de reloj de la *PSP* para poder ofrecer una buena emulación, controles, visualización de frames por segundo, etc.

Dicho de esta forma, es una explicación general, pero a continuación, se mostrará de forma totalmente detallada los objetivos del sistema [**APUNTES INGENIERÍA DEL SOFTWARE 1 – FRANCISCO JOSÉ GONZÁLEZ CABRERA**]:

OBJ-01	***Ejecutar emulador Master System 2 en PSP***
Versión	*1.0*
Autores	*Jorge García Flores*
Fuentes	
Descripción	*El sistema Sony Playstation Portable deberá ejecutar el emulador de la videoconsola Master System 2.*
Subobjetivos	*Ejecutar juegos en el emulador* *Configuración del emulador*
Importancia	*Imprescindible*
Urgencia	*Alta*
Estado	*Validado*
Estabilidad	*PD*
Comentarios	Ninguno

OBJ-02	***Ejecutar juegos en el emulador***
Versión	*1.0*
Autores	*Jorge García Flores*
Fuentes	
Descripción	*El emulador de la videoconsola Master System 2 deberá ejecutar las ROMS.*
Subobjetivos	
Importancia	*Imprescindible*
Urgencia	*Alta*
Estado	*Validado*
Estabilidad	*PD*
Comentarios	Ninguno

OBJ-03	***Configuración del emulador***
Versión	*1.0*
Autores	*Jorge García Flores*
Fuentes	
Descripción	*Se podrá configurar el emulador para características de vídeo, sonido, controles, etc.*
Subobjetivos	
Importancia	*Imprescindible*
Urgencia	*Alta*
Estado	*Validado*
Estabilidad	*PD*
Comentarios	Ninguno

2.3.- FUNCIONALIDADES DEL PRODUCTO

A continuación, de forma detallada, se presentan y enumeran las funcionalidades disponibles por parte del producto desarrollado para el Proyecto Fin de Carrera.

Dichas funcionalidades pueden ser divididas en dos grandes apartados dependiendo de a quién afecte. Esto es debido a que existen *ROMS* con las que se podrá realizar una serie de cosas y el contenedor para ejecutar el emulador, en el que se podrán realizar cambios en él. Además, existen una serie de consideraciones especiales que tratan de las informaciones que se crean que también serán descritas. En este momento se detallan de forma general, mostrándose más adelante una forma mucho más detallada mediante tablas [**APUNTES INGENIERÍA DEL SOFTWARE 1 –FRANCISCO JOSÉ GONZÁLEZ CABRERA**].

ROMS:

Las funcionalidades que atañen a una *ROM* son las que se detallan a continuación de forma general.

- ✓ Ejecutar *ROM*.

- ✓ Cargar partida de una *ROM* ejecutada previamente.
- ✓ Guardar partida de una *ROM* ejecutada previamente.
- ✓ Cambiar parámetros de una *ROM* ejecutada previamente.
    - o Parámetro de Audio (diferentes frecuencias).
    - o Parámetro de Vídeo.
    - o Parámetro de Controles.
    - o Parámetro de Sistema.
    - o Parámetro de System.

CONTENEDOR DEL EMULADOR:

Las funcionalidades que se corresponden con el contenedor del emulador son las que se describen a continuación de una forma también general.

- ✓ Menú principal.
- ✓ Opciones del contenedor del emulador.
    - o Parámetro de Vídeo.
    - o Parámetro de Entrada (botones de la videoconsola PlayStation Portable).
    - o Rendimiento de la videoconsola PlayStation Portable a través del contenedor para ejecutar el emulador.
    - o Botón de cancelación de la consola PlayStation portable (hay que recordar que dependiendo de la región de la videoconsola (Asia-Norteamérica, Europa, el botón de cancelar será el botón X -equis- o el botón O -círculo-).

INFORMACIÓN:

Respecto a las funcionalidades que se corresponden a la información de la ROM, también se tratan de forma general en este apartado. Dichas funcionalidades son las siguientes.

- ✓ Información de *ROMS*.
- ✓ Información de los controles.
- ✓ Información de las opciones.
- ✓ Información de partida guardada correspondiente a una *ROM*.
- ✓ Información del sistema.
- ✓ Número máximo de partidas guardadas.
- ✓ Cambio de opciones sin resetear el emulador.

2.4.- DESCRIPCIÓN DEL PRODUCTO

En el actual apartado se realizará una descripción del producto que se ha desarrollado como Proyecto Fin de Carrera, tratado como un subapartado a su vez, el cual el Diseño de la Interfaz de Usuario. Respecto al Diseño de la interfaz de usuario se explicarán una serie de consideraciones que hay que tener en cuenta en toda interfaz que se desarrolle, y a su vez también se detallará cómo se encuentra estructurada la aplicación desarrollada para el Proyecto Fin de Carrera.

2.4.1.- DISEÑO DE LA ESTRUCTURA DE LA APLICACIÓN

Lo primero que hay que hacer es explicar qué es una interfaz de usuario. Según [Wikipedia], la interfaz de usuario se define como el medio con que el usuario se comunica con una máquina, equipo o computadora (en nuestro caso la videoconsola *PlayStation Portable*), y comprende todos los puntos de contacto entre el usuario y el equipo. Por norma general suelen ser intuitivas (fáciles de entender) y fáciles de accionar.

En el producto desarrollado para el Proyecto Fin de Carrera se presenta una interfaz básica que será gráfica (**GUI**, *Graphic User Interface*) que incluye un menú con una serie de opciones mostrado todo a través del contenedor implementado y que podrá ser visualizado a través de la videoconsola *PlayStation Portable*. La interfaz usada es una interfaz de **Software-Hardware**, que establece un puente entre la máquina (*PlayStation Portable)* y el usuario, permitiendo a la máquina entender las instrucciones y al usuario el código binario de forma legible. Además hay que decir que en realidad la interfaz no se comunica solamente con la *PlayStation Portable* como máquina, sino también con la videoconsola emulada *Master System 2*, que hace las veces de un nuevo hardware, aunque se encuentre emulado mediante software.

Añadir, que el diseño de las interfaces de usuario se han de centrar en la experiencia del usuario con las aplicaciones y los conocimientos que tengan adquiridos, por lo que pueden existir diversos tipos de usuarios y que engloban a un gran conjunto que va desde usuarios totalmente experimentados hasta usuarios no han tenido nunca una relación con un software, independientemente del tipo de aplicación que se realice. Por todo ello, una interfaz de usuario debe de cumplir una serie de requisitos, los cuales son los siguientes que se van a detallar de una forma breve.

- **Puesta en marcha y apagado:** el producto desarrollado permite una puesta en marcha y apagado del emulador sin tener que apagar el sistema bajo el que está corriendo, por lo que está realizado de una forma totalmente eficiente sin que el usuario tenga la molestia de apagar la *PlayStation Portable* para poder ejecutar otra aplicación

- Control de las funciones manipulables del equipo**: en realidad lo que se manipula son las funciones del** contenedor, no las funciones de la máquina emulada, ya que sería algo prácticamente imposible mediante software (de forma física sí que se podría, como por ejemplo introducir en un sistema físico *Master System 2* una visualización

mediante *RGB* y no analógico como se encontraba implementado).

- **Manipulación de archivos y directorios:** se permite *CRUD* (crear, leer, modificar y borrar) directorios, así como *ROMS*, pero en este caso se tiene que hacer conectando la *PlayStation Portable* a un equipo (ordenador de sobremesa, notebook, laptop), ya que todo se encuentra en la tarjeta de memoria *Memory Stick*.

- **Herramientas de desarrollo de aplicaciones:** en nuestro caso esto no es necesario, ya que lo único que se va a hacer es ejecutar una máquina emulada (*Master System 2*) a través de un contenedor y que se ejecutará en una *PlayStation Portable* para poder ejecutar a su vez dicha máquina emulada videojuegos creados para ella.

- **Comunicación con otros sistemas:** nuestra aplicación se comunicará en realidad con dos sistemas, uno físico (*PlayStation Portable*) y otro lógico (*Master System 2)*, que a su vez es pseudofísico por las características que contiene. Además hay que indicar que la máquina física que ejecuta el software, puede comunicarse con otras máquinas a través de internet mediante *wi-fi* con una conexión del tipo *ad-hoc* y también con ordenadores mediante cable USB.

- **Información de estado:** el software desarrollado permite una información acerca del estado de la batería de la *PlayStation Portable*, la temperatura que tiene y la hora del sistema.

- **Configuración de la propia interfaz y entorno:** la interfaz puede configurarse según una serie de parámetros que modifican el contenedor de la aplicación.

- **Intercambio de datos entre aplicaciones:** se realiza un intercambio de datos entre aplicaciones, ya que mediante el contenedor de la aplicación podemos elegir una *ROM* y ser

ejecutada en otra aplicación, que en este caso es el de la máquina emulada *Master System 2*.

- **Control de acceso:** no se ha considerado necesario un control de acceso a la aplicación por dos motivos. El primero es debido a la naturaleza del desarrollo de la aplicación en la escena *homebrew* y segundo, no existe diferenciación de usuarios.

- **Sistema de ayuda interactivo:** existen una serie de mensajes que aparecen de forma dinámica en caso de realizar unas acciones por parte del usuario y que permiten una ayuda para tener una experiencia totalmente satisfactoria de la aplicación.

Según todo esto, se puede ver que el usuario no va a tener ningún problema con la aplicación, aunque se haya desarrollado para una máquina de propósito específico como es la videoconsola *PlayStation Portable*, ya que contiene las siguientes consideraciones:

- ✓ Presenta una claridad y una estética acordes debido a su sencillez de menús y desplazamiento a través de ellos.

- ✓ Fácil de utilizar con los controles de PlayStation Portable.

- ✓ Es perfectamente funcional, haciendo lo que tiene que hacer y siendo diseñado para ello.

- ✓ Presenta una legibilidad clara en todas las opciones que presenta y el sistema de directorios que se ha creado, por lo que hay una probabilidad muy pequeña de perderse, sobre todo una vez que el usuario se ha familiarizado con el producto.

- ✓ Es rápido a la hora de cargar la aplicación, así como a la hora de cargar los videojuegos la consola emulada, ya que todo se encuentra en una tarjeta de memoria, al contrario que los videojuegos originales para *PlayStation Portable*

que se encuentran en un *UMD*, por lo que su acceso es mucho más rápido que una unidad lectora.

- ✓ Todo lo que se ha creado en la interfaz es suficientemente significativo para un usuario, tanto los mensajes que se crean de ayuda, como la visualización de la propia interfaz.

Como conclusión, decir que la interfaz creada presenta todas las especificaciones que han sido definidas como un estándar.

2.5.- DESCRIPCIÓN DE LA NOMENCLATURA UTILIZADA

En el apartado actual se debe presentar una descripción acerca de una serie de términos que se han ido utilizando a lo largo del documento para una familiarización con ellos a modo de glosario. Esto es debido a que existen términos específicos que corresponden a aspectos no tratados a lo largo de la enseñanza de la Ingeniería Técnica de Informática de Gestión y deben ser explicados para que todos los usuarios de la documentación, así como de la aplicación sepan de qué se está hablando.

Esta nomenclatura se desarrollará para una forma más cómoda del lector en un orden alfabético, presentándose en una tabla que contendrá dos columnas. La primera corresponde al término y la segunda será su definición correspondiente de una manera amena, pero a la vez rigurosa.

TÉRMINO	DESCRIPCIÓN
Master System 2	Videoconsola de tercera generación, la cual es de 8 bits, de sobremesa y es producida por *SEGA*.
Memory Stick	Tarjeta de memoria flash cuyo funcionamiento permite guardar datos, en este caso de partidas de juegos de la consola *Playstation Portable*, y emuladores y los propios juegos de los

	emuladores, además de fotos, vídeos, música, etcétera para ser ejecutados dentro de la *Playstation Portable*.
Playstation Portable	Videoconsola de séptima generación, la cual es de 32 bits, portátil y es producida por *SONY*.
ROM	Representa la memoria en la que está almacenado un juego que podrá ser ejecutado por la *Master System 2*.
iROM	Juego comenzado que se caracteriza por ser una *ROM* más el estado en ejecución de una partida jugada anteriormente y que ha sido guardada.

2.6.- DESCRIPCIÓN DEL ENTORNO DE EJECUCIÓN DE LA APLICACIÓN

El sistema desarrollado se basa en una Arquitectura de Máquinas Virtuales. Los motivos de por qué se utiliza este tipo de arquitectura y no otra los iré detallando a lo largo del presente apartado, pero antes de hacerlo, primero se debe definir y explicar qué es una máquina virtual para entender las razones de esta arquitectura elegida.

La definición de máquina virtual según [**Wikipedia**] es un software que simula una computadora y puede ejecutar programas como si fuese una computadora real. Una característica esencial de las máquinas virtuales es que los procesos que ejecutan están limitados por los recursos y abstracciones proporcionadas por ellas, es decir, que estos procesos no pueden escaparse de esta "máquina virtual".

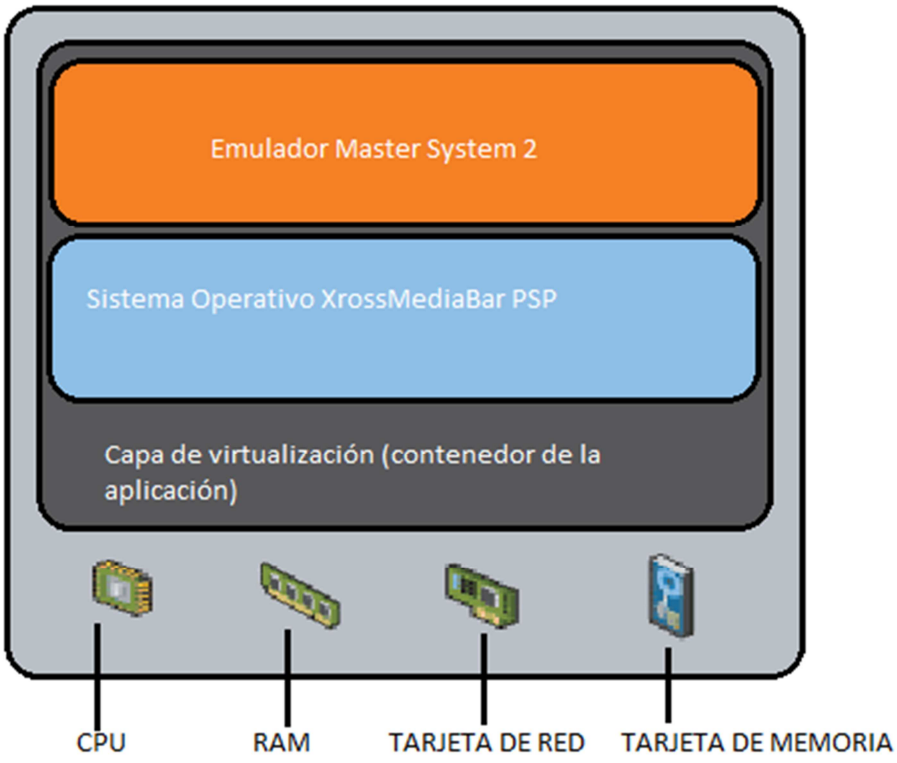
Por parte de la empresa **vmware** nos dice que una máquina virtual es un contenedor de software perfectamente aislado que puede ejecutar sus propios sistemas operativos como si fuera un ordenador

físico, comportándose como un ordenador físico que contiene su CPU virtual, memoria, disco duro y tarjeta de red.

A VMware virtual machine

Estos primeros conocimientos que se han definido nos indican ya una primera aproximación de la arquitectura que deberemos usar para abordar el desarrollo del producto que ofreceremos como Proyecto Fin de Carrera.

Según el producto desarrollado y teniendo en cuenta la arquitectura a utilizar, el modelo de nuestra arquitectura será como el que especifica el siguiente gráfico:



Visto el gráfico, pasaré a enumerar y detallar cada componente que contiene:

- CPU: como ya se sabe, es la unidad central de proceso, en el caso de la *PlayStation Portable* es de multipropósito *Sony Allegrex* basado en MIPS de velocidad configurables 1/333 MHz, incluyendo una CPU basada en MIPS32 Rk-4 de 32 bits, una FPU (unidad de coma flotante) y una VFPU (cálculo vectorial). También incluye un procesador conocido como *Media Engine*, teniendo las mismas características que el procesador Allegrex para manejar audio/vídeo y un DSP (procesador digital de señal).

- RAM: 32 MB de memoria principal

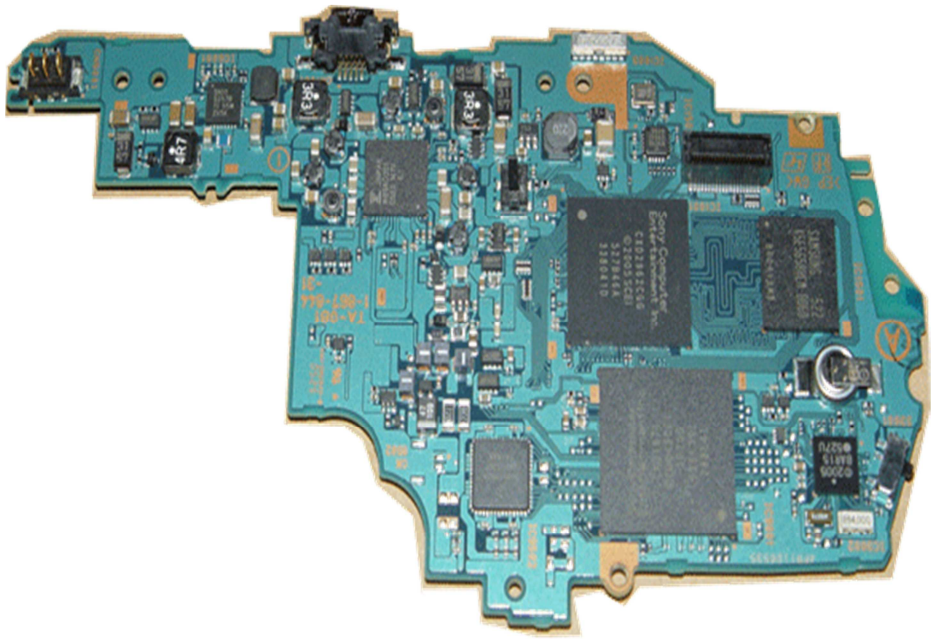
- Tarjeta de red: wi-fi IEEE 802.11b permitiendo conexión ad-hoc y por infraestructura (en el caso del producto desarrollado será por ad-hoc la conexión).

- Tarjeta de memoria: *Memory Stick* que permite el almacenamiento de datos, que va desde los 32 MB hasta varios GB (por norma general entre 4 y 8).

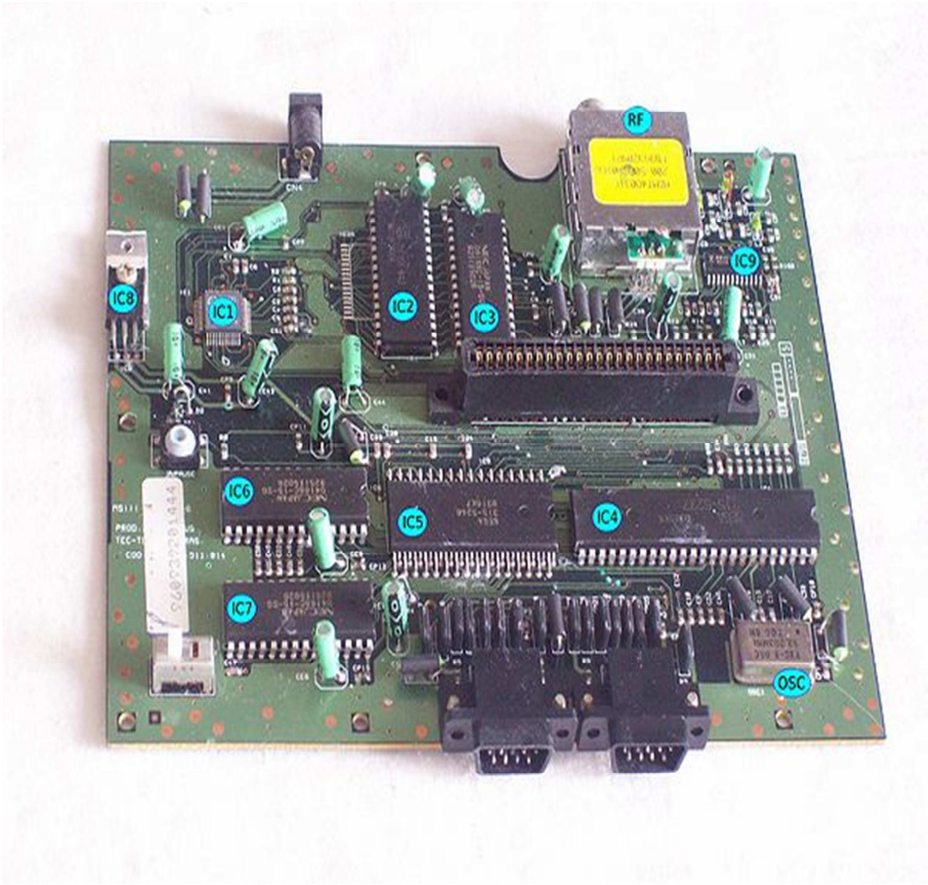
- Capa de virtualización: capa que permite la virtualización de un sistema o componentes, como un procesador, memoria, o un dispositivo de entrada/salida, mapeando los recursos visibles dentro de la interfaz en los recursos del sistema real posiblemente diferente (en el caso del producto desarrollado, en una videoconsola *PlayStation Portable*). Por consiguiente, el verdadero sistema aparece como un sistema virtual diferente (emulador *Master System 2*).

- Sistema Operativo XrossMediaBar PSP: interfaz gráfica de usuario a modo de sistema operativo que usa la *PlayStation Portable*, así como más productos de la marca *Sony*. Se basa en un menú de iconos expuestos de forma horizontal navegando a través de ellos y desplegando nuevos iconos en cada opción del menú.

- Emulador Master System 2: creación por medio de software del hardware que se realizó de la videoconsola de 8 bits, *Master System 2,* con todas sus características, incluyendo ejecución de juegos sean de región *PAL* o *NTSC*.



Placa base de la videoconsola *PlayStation Portable*



Placa base de la videoconsola *Master System 2*

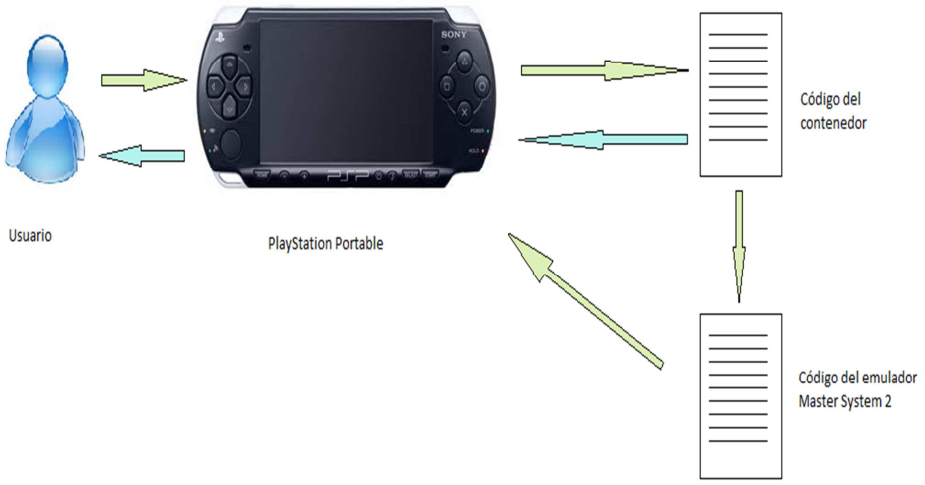
El sistema está basado en los componentes software y hardware que se detallan a continuación:

- ✓ PlayStation Portable: videoconsola de séptima generación, la cual es de 32 bits, portátil y es producida por *SONY*. Esta es la máquina que nos permitirá poder ejecutar nuestra aplicación desarrollada para el Proyecto Fin de Carrera.

- ✓ Contenedor (aplicación): aplicación realizada para el Proyecto Fin de Carrera que consiste en una interfaz gráfica de usuario (*GUI*) y que será la encargada de contener el emulador y poder ejecutarlo como un proceso aparte.

- ✓ Master System 2: videoconsola de tercera generación, la cual es de 8 bits, de sobremesa y es producida por *SEGA*. Esta es la máquina a emular mediante software para poder realizar su función principal que es la ejecución de videojuegos creados para sus características.

Por lo tanto, un diagrama sencillo que especifique todo esto a modo de arquitectura de máquina virtual sería como el que sigue en el siguiente gráfico:



Visto todo esto, es cuando se puede comprobar el por qué se ha usado esta arquitectura y no otra. Lo primero que hay que tener en cuenta es que se va a desarrollar un contenedor que hará las veces de capa de virtualización y que será el encargado de ejecutar la máquina emulada (*Master System 2*) en la máquina física (*PlayStation Portable*) siguiendo una interfaz gráfica de usuario como se ha explicado detalladamente anteriormente. El emulador es la aplicación que permitirá ejecutar juegos (*ROMS*) y todo ello siendo visualizado en una *PlayStation Portable* usando su CPU, memoria RAM, tarjeta de red y tarjeta de memoria con los recursos asignados y que dicho emulador puede ser ejecutado en segundo plano por la capa de virtualización (contenedor) sin necesitar resetearlo cuando se realice algún cambio.

Otra consideración a tener en cuenta es por qué se usa la conexión entre dos máquinas *PlayStation Portable* mediante ad-hoc. Esto es debido a que *PlayStation Portable* tiene una conexión inalámbrica wi-fi 802.11b y es la forma más fácil de conectarla a una red sin necesidad de usar un punto de acceso, por lo que en cuanto una máquina *PlayStation Portable* se conecta a la red, puede compartir la conexión con los demás equipos de esa red de una manera tradicional, mientras que una conexión por infraestructura requiere de un punto de acceso.

2.7.- TECNOLOGÍA EMPLEADA

Se ha usado una serie de herramientas para desarrollar la aplicación en el presente Proyecto Fin de Carrera y que son las siguientes:

- Hardware:
    - Ordenador de sobremesa.
    - Sony PlayStation Portable.
    - Tarjeta de memoria Memory Stick.
- Software
    - DevC++ IDE como entorno de desarrollo (IDE) para programar en C y su correspondiente compilador.
    - PSPsdk como kit de desarrollo para PlayStation Portable y sus correspondientes librerías.
    - Kit de Programación PSP 3.0 que es un compilador para crear ejecutables *EBOOT.PBP* de *PlayStation Portable* y además contiene:
        - Oracle VM Virtual Box 4.2.12.
            - Distribución Debian.

        - Memoria Base: 4GB.
        - Procesadores: 8.
        - Memoria de vídeo: 12 MB.
        - Puerto SATA: Normal, 14GB.
- PSPdev Win32 + SDK
- Librería devkitARm + SDK.
- devkitPSP Revisión 9.
- Librerías:
    - SDL.
    - SDL_MIXER.
    - JPEG.
    - LIBpsp2d.
    - PSPGL.
    - LibbulletML.
    - Lib_Tremor.
    - LibPng.
    - Zlib.
    - LibMad,
    - OSlib.
    - Freetype.
    - OGG.
    - PSPToolChain.

- StarUML 5.0 para el modelado del diseño de la aplicación usando el lenguaje unificado de modelado UML.
- GoldWave 4.26: para realizar el sonido de la aplicación en el menú del Sistema Operativo de la videoconsola *PlayStation Portable*, XMB (*Xross Media Bar*).
- Atrac3: librería para la generación de sonido con una compresión de formato at3, el cual es sólo compatible con sistemas Sony y que es cerrado.
- Oracle VM Virtual Box 4.2.12: aplicación de Oracle para la ejecución de diferentes Sistemas Operativos mediante una máquina virtual.
- Debian: distribución Linux para la compilación de los ficheros implementados con el SDK para la videoconsola PlayStation Portable.

2.8.- VISIÓN GLOBAL DE LA MASTER SYSTEM 2

La videoconsola *Master System 2* alberga dentro de ella una serie de procesadores de propósito específico, junto con un conjunto de elementos hardware para acondicionar las señales de comunicaciones entre todos sus componentes.

La *CPU* de la máquina es el microprocesador *Z80A*, que para ser liberado de carga, la *Master System 2* dispone de otros dos procesadores de propósito específico.

El primero de ellos es el procesador de vídeo, variante del *TMS9918* de *Texas Instruments*, encargado de realizar varios efectos por hardware como son escalados, detección de colisiones, doble capa de dibujado, etc.; proporcionando una resolución de 256x192 píxeles.

El segundo procesador es el generador de sonido que corresponde a un chip *SN76489* de *Texas Instruments*, capaz de generar

tres canales independientes de sonido y un cierto canal especializado en ruido.

Aunque la videoconsola *Master System 2* sólo incorpora 8 Kbyte de memoria *RAM*, los cartuchos pueden llegar a albergar hasta un Mbyte de memoria, la cual será direccionable por medio de mapeados de memoria.

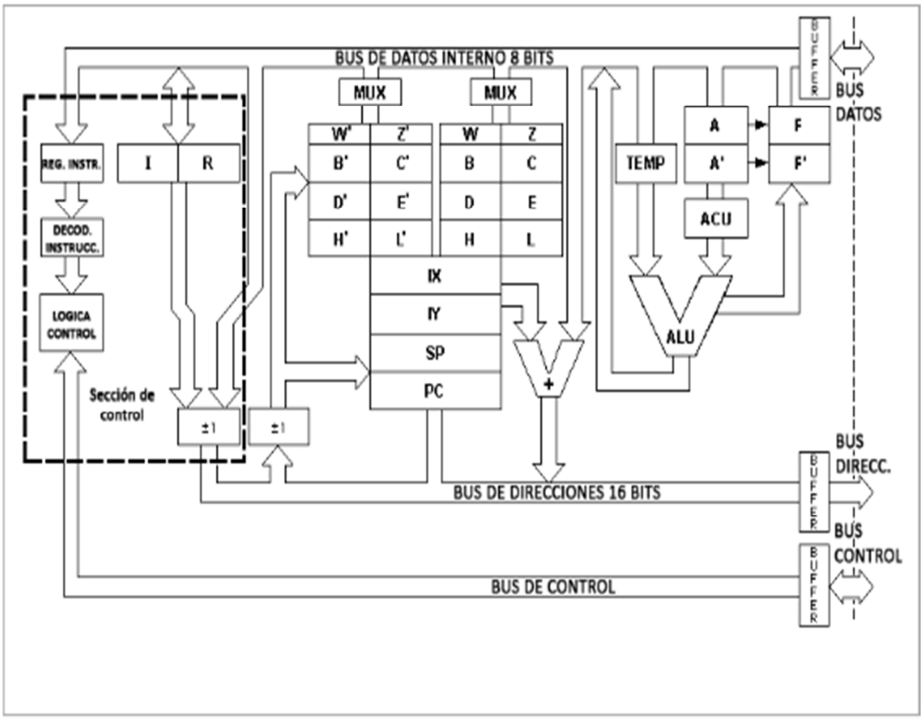
Por último, de forma externa a la Master System 2, es posible conectar una serie de dispositivos controladores (periféricos) para interactuar con la misma, como son los *control pad*, *pistolas de luz*, *gafas tridimensionales* y otros dispositivos variados. Todo esto se puede consultar en [**http://www.smspower.org/Development/Documents**]

2.8.1.- CPU Z80

En este apartado, se detallará la arquitectura del microprocesador de la *Master System 2*, el *Z80A*.

Éste es un procesador *CISC* (Complex Set Instruction Code) diseñador para ser compatible con las aplicaciones realizadas para el procesador *8080* de *Intel*, compartiendo el mismo juego de instrucciones, conteniendo los mismos códigos de operación y al cual añade una serie de instrucciones para nuevos programas para que no tengan que mantener una compatibilidad.

En el caso de la videoconsola *Master System 2*, está conectado a un reloj externo que funciona a 3.579545 MHz, llevando la temporización del resto de componentes del hardware de la videoconsola.



En la figura se puede observar que el microprocesador incorpora un bus de datos de 8 bits, limitando el tamaño de los datos intercambiados con los periféricos y memoria de un byte, y un bus de direcciones de 16 bits que supone un máximo de 65.536 palabras de memoria direccionables. Por otro lado, tiene dos espacios de direccionamiento de memoria y de E/S, pudiéndose direccionar la misma cantidad de unidades en cada uno de ellos.

Los 26 registros que contiene son los siguientes:

- B, C, D, E, H, L, B', C', D', E', H', L': registros de 8 bits que pueden agruparse de dos en dos para formar registros de 16 bits de la forma BC, DE, HL, B'C', D'E' y H'L'. Sólo son accesibles a la vez 6 de ello, pues se encuentran divididos en dos bancos intercambiables por medio de unas instrucciones específicas de cambio de banco.

- A, F, A', F': registros de 8 bits que sólo son accesibles al mismo tiempo dos de ellos. El registro A es un registro de tipo acumulador el cual siempre será uno de los operandos

de la *ALU* (Unidad Aritmético Lógica). En el registro F se encuentran los flags de la máquina para poder guardar su estado.

- PC: contador de programa que apunta a la dirección de la siguiente instrucción a ejecutar, siendo de 16 bits.
- SP: puntero de pila, la cual apunta a la primera dirección libre de la pila de ejecución y que es también de 16 bits.
- IX, IY: registros de 16 bits usados como dirección base para instrucciones que hacen uso de vectores. Existe una *mini ALU* que sólo sirve para sumarle un dato de 8 bits a estos registros y volcar el resultado al bus de direcciones.
- I: registro de 8 bits usado para el tratamiento de interrupciones.
- R: registro de 8 bits usado como salida para refrescar memorias externas y también como generador de números pseudoaleatorios.
- W, Z: registros invisibles para el programador utilizados para almacenar resultados temporales al ejecutar ciertas instrucciones del procesador.

2.8.2.- SISTEMA DE ENTRADA/SALIDA

Como se ha comentado anteriormente, el microprocesador Z80A tiene un bus de direcciones de 16 bits, pudiendo direccionar hasta 65.536 palabras de un byte de longitud. Por ello, el Z80A es capaz de seleccionar en un momento dado, mediante una señal, si comunicarse con dispositivos de E/S o memoria., por lo que permite el uso de todo el espacio direccionable para cada uno de los dos subsistemas.

A pesar del espacio de E/S tan grande del que se dispone y que proporciona el microprocesador *Z80A* de la videoconsola *Master*

System 2, sólo hace uso de una pequeña cantidad de estos puertos de E/S, donde se encuentran conectados una serie de periféricos externos al microprocesador.

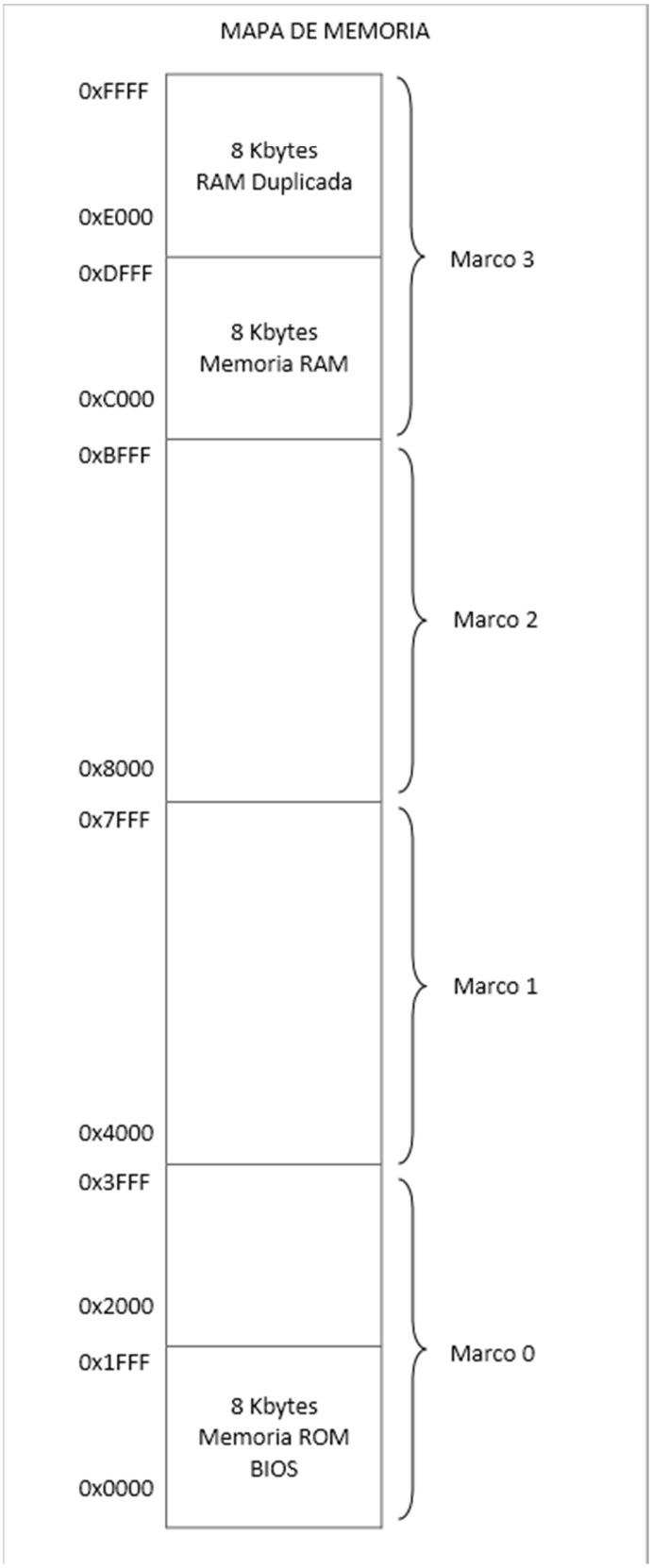
Puerto	Dirección	Función
0x3E	Salida	Activa la memoria.
0x3F	Salida	Registro de control del joystick.
0x7E	Entrada	Contador vertical del VDP.
0x7F	Entrada	Contador horizontal del VDP.
	Salida	Registro de control del PSG.
0xBE	Entrada/Salida	Registro de datos del VDP.
0xBF	Entrada/Salida	Registro de control/estado del VDP.
0xC0	Entrada	Registro de estado del joystick.
0xC1	Entrada	Registro de estado del joystick.
0xDC	Entrada	Registro de estado del joystick.
0xDD	Entrada	Registro de estado del joystick.

2.8.3.- SISTEMA DE MEMORIA

La videoconsola *Master System 2* dispone de 8 Kbyte de memoria *RAM* que se encuentran mapeados a partir de la dirección *0xc000*, pero que también son accesibles desde la dirección *0xE000* (conocido como *RAM* duplicada). El sentido de tener la memoria *RAM* duplicada es que las cuatro últimas direcciones de memoria (*0xFFFC-0xFFFF*) tienen en ellas mapeados registros de sólo escritura (registros de marco) que no pueden leerse por sí mismos, pero sí es posible leer direcciones equivalentes en la otra zona de la memoria *RAM* (*0xDFFC-0xDFFF*) para obtener su valor.

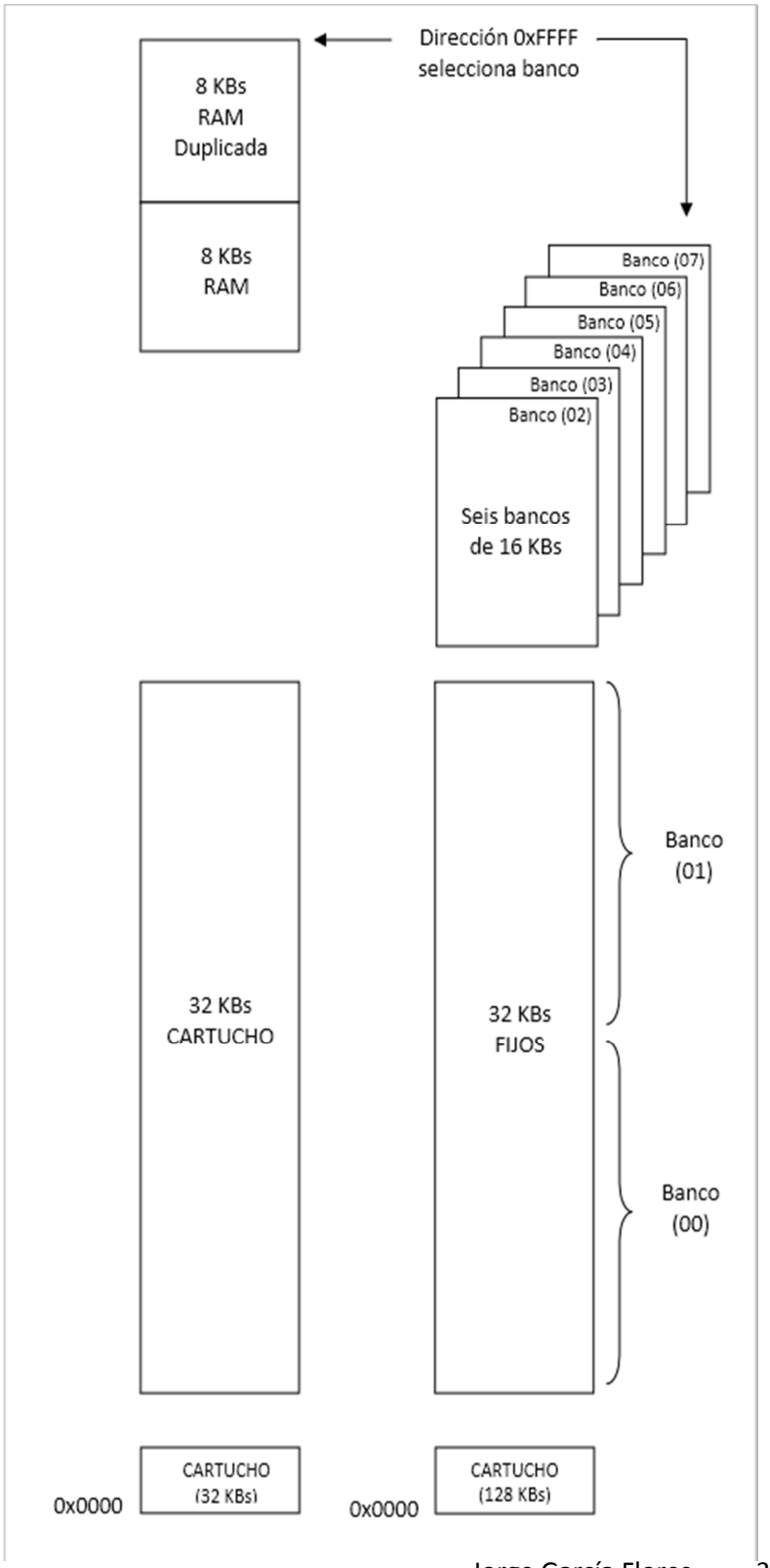
Al iniciarse la *Master System 2*, en la zona inferior de su memoria se encuentra una pequeña *ROM* de 8 Kbyte que contiene la *BIOS* y que será encargada de inicializar ciertos registros y copiarse a sí misma a la memoria *RAM*. Pero si encuentra un cartucho introducido en la ranura del mismo, le cede el control, mapeando la *ROM* del cartucho a partir de la dirección *0x0000*.

Para cargar la *ROM* del programa sólo se dispone de 48 Kbyte de memoria direccionables. Esto es así físicamente y de este modo se cargan los programas de hasta 32 Kbyte de memoria, pero como hay programas que pueden llegar hasta el Mbyte de memoria, ahí entran en juego los mapeadores de memoria.


MAPA DE MEMORIA
0xFFFF
8 Kbytes
RAM Duplicada
0xE000
0xDFFF
8 Kbytes
Memoria RAM
0xC000
Marco 3
0xBFFF
Marco 2
0x8000
0x7FFF
Marco 1
0x4000
0x3FFF
0x2000
0x1FFF
8 Kbytes
Memoria ROM
BIOS
0x0000
Marco 0

Al usar mapeadores de memoria se dividen los 64 Kbyte direccionables por el procesador en 4 marcos de 16 Kbyte cada uno numerados de 0 a 3 (de direcciones bajas a altas). Así mismo, la ROM del programa en cuestión también se dividirá en tantos bancos de 16 Kbyte como sea necesario para cubrir todo su tamaño.

Mediante los registros de marco, el programa podrá seleccionar qué banco de la ROM mapear en cada momento en qué marco de memoria (del 0 al 2, a excepción del 3 que se encuentra ocupado por la memoria RAM), posibilitando de este modo el acceso a cualquier tamaño de memoria.


Dirección 0xFFFF
selecciona banco
8 KBs
RAM
Duplicada
8 KBs
RAM
Banco (07)
Banco (06)
Banco (05)
Banco (04)
Banco (03)
Banco (02)
Seis bancos
de 16 KBs
32 KBs
CARTUCHO
32 KBs
FIJOS
Banco
(01)
Banco
(00)
0x0000
CARTUCHO
(32 KBs)
0x0000
CARTUCHO
(128 KBs)

2.8.4.- PROCESADOR DE VÍDEO (VDP)

La videoconsola *Master System 2* contiene un procesador de vídeo dedicado conocido como *VDP* (*Visual Display Processor*), cuya función es liberar al microprocesador *Z80A* de computar gran parte de la carga gráfica.

Este procesador se basa en un *TMS9918A* de la compañía *Texas Instruments* pero contiene algunos aspectos mejorados.

Aunque el *VDP* soporta 16 modos de vídeo distintos, tan sólo uno de ellos (el modo 4) se encuentra documentado por la compañía *SEGA*, siendo de una resolución de 256x192 píxeles.

El *VDP* cuenta con dos pequeñas memorias *RAM* internas en el propio chip.

- ✓ *CRAM* (*Color RAM*): encargada de almacenar las paletas de colores que se pueden usar en pantalla y que tiene un tamaño de 32 bytes, por lo que contiene 32 colores *RGB* distintos de 6 bits cada uno de un total de 64.
- ✓ *VRAm* (*Video RAM*): tiene un tamaño de 16 Kbyte y se encuentra dividida en tres zonas.
    - o Mapa de pantalla (1.792 bytes): zona dedicada a almacenar un mapa con la información de los tiles que forman el fondo (32x24 tiles de 8x8 píxeles).
    - o Tabla de atributos (256 bytes): zona dedicada a almacenar una tabla con los atributos de un máximo de 64 sprites simultáneos.
    - o Generador de caracteres (14.336 bytes): zona de memoria donde se almacenan los caracteres, es decir pequeños gráficos de 8x8 píxeles de los que están formados los tiles y los sprites.

Por otro lado, el *VDP* cuenta con 12 registros internos de 8 bits que sirven para determinar su comportamiento.

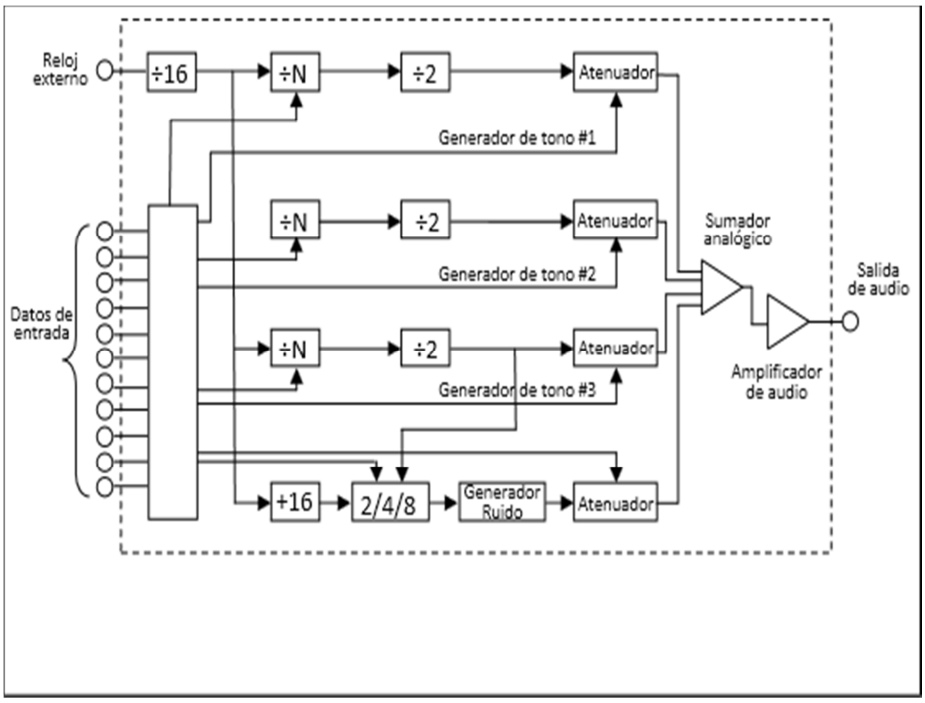
Para finalizar, hay que indicar que el *VDP* se comunica con el *Z80A* a través de dos puertos de E/S.

2.8.5.- GENERADOR DE SONIDO PROGRAMABLE (PSG)

La videoconsola *Master System 2* incorpora un chip dedicado exclusivamente a las funciones de sonido conocido como *PSG* (*Programmable Sound Generator*).

Este chip se basa en el *SN76489* de *Texas Instruments* e incorpora cuatro canales de sonido, de los cuales tres son generadores de tono y uno de ruido. Cada canal tiene asociado a sí mismo un atenuador para poder variar el volumen de forma independiente.

El *PSG* se encuentra conectado al reloj central de la videoconsola *Master System 2*, dividiendo su frecuencia por 16 y usando la frecuencia resultante (233 KHz) como base para configurar la frecuencia de los distintos canales.



Cada uno de los tres canales de tono que incorpora el chip *PSG* se usará para producir sonidos armónicos (músicas, voces digitalizadas, efectos sonoros, etc.), mientras que el canal de ruido, producirá a su vez ruido blanco o periódico para generar percusiones, explosiones, interferencias, etc.

Una vez que se produzca la salida de los cuatro canales y se haya modificado su volumen con los atenuadores, pasarán a través del mezclador, el cual es un sumador lógico, cuya función es sumar las cuatro ondas resultantes, generadas por los cuatro canales, para producir la onda final. Luego pasa a través de un amplificador de audio para producir la salida final que se enviará a través de las salidas que incorpora la videoconsola *Master System 2* para ser reproducida.

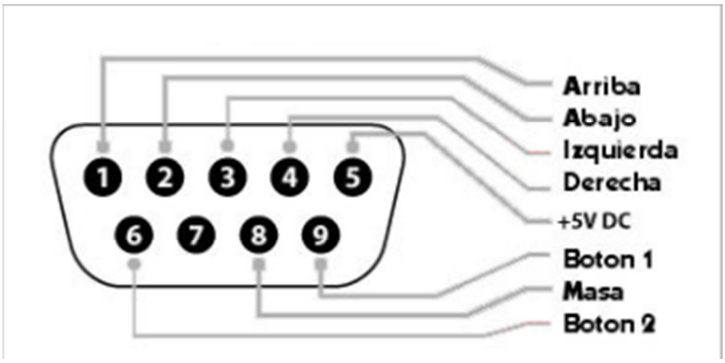
El *PSG* cuenta con 8 registros, de los cuales, cuatro son registros de 4 bits para el volumen, tres registros de 10 bits para la frecuencia de los canales de tono y un último registro de 4 bits para el canal de ruido.

El microprocesador *Z80A* podrá comunicarse con el *PSG* a través de sus puertos de E/S de forma unidireccional.

2.8.6.- DISPOSITIVOS CONTROLADORES

Al ser la *Master System 2* una videoconsola, todos sus dispositivos controladores están diseñados específicamente para responder al control de los videojuegos.

Todos los dispositivos de control (tanto de la propia *SEGA* como de terceras compañías) tenían un interfaz físico común.



Los dispositivos de control se comunican con el microprocesador *Z80A* a través de dos puertos de E/S.

Estos puertos funcionan con lógica inversa. La pulsación de uno de los contactos no es excluyente, es decir, se pueden pulsar de forma simultánea varios contactos del controlador a la vez.

3.- METODOLOGÍA EMPLEADA

Se ha desarrollado el Proyecto Fin de Carrera de nombre "Emulador de Master System 2 corriendo bajo PlayStation Portable" como un modelo en cascada en el ciclo de vida clásico de un proyecto software. El por qué se ha utilizado este modelo y no otro es por la razón de que no podremos pasar de fase hasta que no se hayan cumplido todos los objetivos de la anterior y se haya especificado todo correctamente. Es decir, hasta que no se hayan definido todos los requisitos de forma totalmente correcta no podremos pasar a la fase de diseño, debido a que puede haber cosas que hayan pasado por alto. Es importante señalar que en esta etapa se debe consensuar todo lo que se requiere del sistema y será aquello lo que seguirá en las siguientes etapas, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software.

De la misma manera, no podremos pasar de esta fase a la de implementación, ya que tenemos que tener modelado perfectamente el

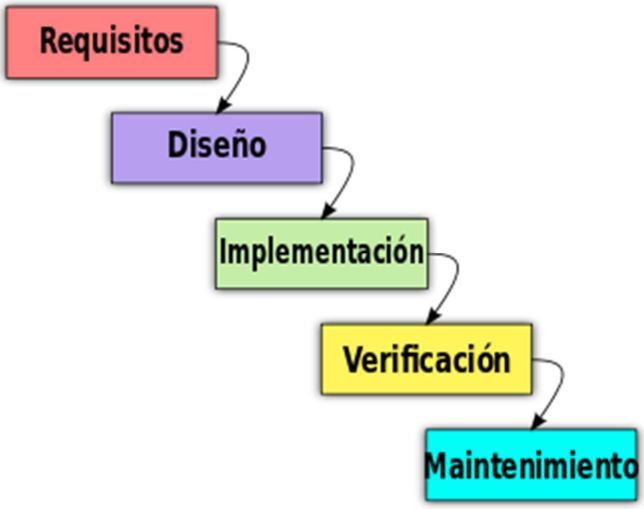
diseño de la aplicación, más cuando la arquitectura usada es una **arquitectura de Máquina Virtual** y tiene que estar detallado hasta el más mínimo detalle para no tener imprevistos que de otra manera podrían resultar insalvables.

Hecha y supervisada la fase de diseño y habiéndose dado paso a la fase de diseño, es cuando debemos comenzar la fase de implementación del código de la aplicación. Esta es la fase más laboriosa por lo que se tiene que abordar, ya que es una programación estructurada usando no sólo un lenguaje de alto nivel como es C, sino que también se debe usar lenguaje ensamblador para realizar una serie codificaciones que se explicarán más adelante y de forma mucho más detallada. Además, hay que indicar que va a haber una serie de componentes que van a interaccionar entre ellos, tanto a nivel software, como a nivel hardware a bajo nivel.

Terminada esta fase, se pasa a la fase de pruebas y verificación. En esta fase, los elementos ya programados se ensamblan para comprobar que el sistema funciona adecuadamente y que cumple con los requisitos realizando una serie de pruebas para testear el producto desarrollado y que no contiene fallos y cumple todos los objetivos y funcionalidades especificados. Una vez hecho, en la verificación es donde el usuario ya ejecuta el sistema implementado. En el caso de que no sea así, habrá que volver a la fase anterior y no pasar de ella hasta que se vuelva a corregidos los errores y volver a pasar por la batería de pruebas realizada.

Cuando la fase de pruebas ha terminado, se puede abordar ya la fase de mantenimiento del producto con su instalación y ejecución final en la máquina para el que está desarrollado (*PlayStation Portable*) y nuevas funcionalidades que se puedan hacer.

Por todo esto, se ha considerado que la mejor metodología en cuanto a un ciclo de vida que se pudiera escoger es el ciclo de vida en cascada.



Otras con consideraciones a tener en cuenta en la fase de diseño es que se sigue una guía por medio de casos de uso, permitiendo el desarrollo de conceptos definidos con anterioridad en la fase de requisitos y posterior desarrollo del software a implementar.

La desventaja es que se destina un 75% de recursos al mantenimiento y que cualquier error detectado en la etapa de prueba conduce necesariamente al rediseño y una nueva implementación del código, aumentando los costes del desarrollo del proyecto.

4.- PRESUPUESTO DE GASTOS

En el actual apartado se realiza una descripción de la estimación del coste económico asociado a la realización del desarrollo del Proyecto Fin de Carrera de nombre “Emulador de Master System 2 corriendo bajo PlayStation Portable”, pero primero se debe hacer un inciso en que los sistemas software tienen unas características especiales que les hacen distintos del resto de proyectos de otras ingenierías. Para ello, este apartado se divide a su vez en otros tres apartados que serán descritos a continuación.

4.1.- CARACTERÍSTICAS DE LOS SISTEMAS SOFTWARE

Según se ve en [**APUNTES INGENIERÍA DEL SOFTWARE 1 –FRANCISCO JOSÉ GONZÁLEZ CABRERA**] y [**APUNTES INGENIERÍA DEL SOFTWARE 2 –LUIS EZEQUIEL HERNANZ ALBERTOS, ENRIQUE CARBALLO**] es que el software no se fabrica, sino que se desarrolla. Esto quiere decir que los costes del software no se deben a la fabricación del producto, ya que el software es intangible, sino a las horas empleadas en la creación del producto desarrollado.

Otra característica es que el software al ser intangible no se degrada, es decir que no se rompe, pero sí que se vuelve anticuado, por lo que hay que crear nuevas modificaciones al software ya desarrollado, conllevando a una serie de errores y aumento de costes debido a que aumenta la complejidad en el desarrollo del software, sobre todo en aquellos productos que han sido desarrollados con un fuerte acoplamiento, por lo que su cohesión tiende a ser menor.

Una nueva característica que difiere a un proyecto de ingeniería clásica es el mantenimiento, pues al no romperse el software, no necesita de “nuevas piezas físicas”, pero el fallo en el software puede llegar a ser más grave, ya que puede ser debido a un error en una etapa temprana del desarrollo y que no se ha tenido en cuenta hasta mucho más tarde. Es por ello, que el mantenimiento es mucho más complejo en el software que en cualquier otro producto realizado, aumentando su coste enormemente.

4.2.- PRINCIPALES FASES EN LA ELABORACIÓN DEL SOFTWARE

En este apartado se abordan las principales fases que contiene el desarrollo de todo producto software.

- Objetivos necesarios: se detallan los objetivos que abordará el producto software y lo que aportará a los usuarios,

definiendo funcionalidades del sistema y la forma de utilizar dicho sistema.

- Especificaciones del Sistema: delimitación precisa del sistema y descripción del uso del mismo de distintas maneras por parte de los usuarios finales.

- Análisis: proceso automatizado para analizar el comportamiento del software. Con ello se pretende mejorar el software en cuestiones de correctitud, optimización y seguridad, ya sea mediante análisis dinámico o análisis estático del software.

- Diseño: determinación de la manera de resolver el problema planteado y que ha sido estudiado en la fase de análisis, proponiendo para ello soluciones modelando el comportamiento del sistema y posterior implementación.

- Implementación: creación de las estructuras y algoritmos desarrollados en la fase de diseño mediante la codificación con algún lenguaje de programación.

- Pruebas y verificación: fase en la que se asegura que el software se comporta de acuerdo a las especificaciones y objetivos que se han detallado sin cometer errores en el diseño y en la implementación.

- Documentación: fase en la que se redacta de forma clara y concisa todo lo referente al desarrollo del producto software, así como los manuales destinados a los diferentes usuarios del mismo producto software.

4.3.- ESTUDIO ECONÓMICO DEL PROYECTO

El presente apartado se centra en el estudio económico, con su estimación de costes mediante un estudio realizado de forma orientativa con el que poder hacer frente a los costes que se producen en el desarrollo del producto software, siendo este caso el Proyecto Fin de

Carrera con el nombre de “Emulador de Master System 2 corriendo bajo PlayStation Portable”.

Para ello, se desglosará el estudio mediante diferentes etapas en la elaboración del desarrollo del proyecto, mostrándose la influencia de las etapas en el coste total del producto.

Así mismo, se detallará mediante los puntos de función y el modelo constructivo de costes COCOMO la estimación del coste de dicho proyecto en esfuerzo y tiempo requerido.

Se ha elegido un equipo de tres personas compuesto por los siguientes miembros:

- Jefe de Proyecto que se encargará de una serie de tareas que se verán en la calendarización del proyecto.
- Analista, encargándose de las fases de diseño y análisis.
- Programador que se encarga de la codificación del proyecto.

✓ **Parte 1- Estimación utilizando Puntos de Función:**

La métrica del punto de función es un método en ingeniería del software para medir el tamaño del software. Pretende medir la funcionalidad entregada al usuario independientemente de la tecnología utilizada para la construcción y explotación del software, y también ser útil en cualquiera de las fases de vida del software, desde el diseño inicial hasta la explotación y mantenimiento. Existen diferentes metodologías de medición, de las cuales la más popular es la mantenida por el International Function Point Users Group (IFPUG).

Tradicionalmente se ha medido el tamaño del software mediante distintas métricas: recuento de las líneas de código, número de programas fuente, o técnicas similares, que no resultan aceptables como una buena práctica profesional, porque:

- Su resultado depende fuertemente del entorno técnico y el lenguaje de programación utilizado
- Varía en función de la pericia de cada programador y del uso de normas y metodologías

- No resultan significativas al usuario ni a la dirección

Cuando se trata de establecer métricas de productividad y calidad en la construcción de software, o realizar estimaciones de coste y duración, es imprescindible disponer de una medida fiable y comprensible del tamaño de lo que se construye.

La tabla que usaremos para realizar la práctica será una que se usó en la asignatura de **Ingeniería del Software 1** y que se muestra a continuación:

Factores de Complejidad	**0-5**	**Factores de Complejidad**	**0-5**
Comunicación de datos	3	Funciones distribuidas	0
Rendimiento	5	Configuraciones fuertemente utilizadas	2
Frecuencia de transacciones	4	Entrada on-line de datos	0
Diseño para la eficiencia del usuario final	3	Actualizaciones on-line	0
Procesos complejos	1	Reusabilidad	1
Facilidad de instalación	5	Facilidad de operación	2
Instalación en múltiples lugares	0	Facilidad de cambio	1

Parámetro	**Complejidad baja**	**Complejidad media**	**Complejidad alta**
Entradas	x3	x4	x6
Salidas	x4	x5	x7
Ficheros internos	x7	x10	x15
Ficheros externos	x5	x7	x10
Consultas externas	x3	x4	x6

Para hallar los puntos de función, debemos seguir una serie de pasos:

1.- Identificar el número de elementos y complejidad:

- Entradas:
  - Número máximo de partidas guardadas
  - Cambio de opciones sin resetear el emulador
  - Cambiar sistema
  - Cambiar audio
  - Cambiar vídeo
  - Cambiar system
- Salidas:
  - Información de la ROM
  - Información de los controles
  - Información de las partidas guardadas
  - Información de batería, temperatura y hora actuales
- Ficheros externos:
  - Cargar ROM
  - Guardar Juego
- Ficheros internos:
- Cuestiones externas:

2.- Hallamos los puntos de función sin ajustar (PFNA):

- Total de entradas = 6
- Total de salidas = 4
- Total de ficheros externos = 2
- Total de ficheros internos = 0
- Total de cuestiones externas = 0

Existen seis entradas, cuyas complejidades son las que se explican a continuación. El **número máximo de partidas guardadas** nos indica que podremos guardar hasta 10 partidas de una misma *ROM*, siendo este proceso de complejidad alta, ya que hay que interactuar con el estado en el que se encuentra la partida y guardarlo en una tarjeta de memoria *Memory Stick* para poder acceder al estado de la partida cuando se quiera más adelante. La complejidad del proceso **cambio de opciones sin resetear el emulador** consiste en cambiar una serie de parámetros del contenedor del emulador y guardarlo en la tarjeta de memoria *Memory Stick*, pero el emulador no se debe apagar, sino que

en cuanto se cambien las opciones, debemos seguir en el mismo lugar en el que nos encontráramos anteriormente al realizar los cambios, por lo que es alta su complejidad. A continuación, el proceso de cambiar sistema tiene una complejidad media, ya que cambiamos los parámetros del contenedor del emulador y deben ser guardados en la *Memory Stick*. Tanto **cambiar audio**, como **cambiar vídeo** y **cambiar system** tienen la misma complejidad media, debido a que lo que se hace es interactuar con la PlayStation Portable y el contenedor para que esas modificaciones se reflejen en las *ROMS* y el contenedor del emulador y también deben guardarse sus cambios en la *Memory Stick*.

Respecto a las salidas, todas las complejidades son medias. La **información de la ROM** contiene el nombre de la *ROM*, además deberá ser guardado todo en una tarjeta de memoria *Memory Stick*. La **información de los controles** guarda la información correspondiente al mapeo de los botones para poder manejar las *ROMS* y movernos por el menú, teniendo que estar todo guardado en la *Memory Stick*. La **información de las partidas guardadas** nos guarda en la *Memory Stick* la información de una *ROM* que tiene un estado de juego guardado. Por último, la *información de la batería, temperatura y hora actuales* nos muestra información acerca del estado de la batería (medida en percentiles), la temperatura y la hora actual realizando llamadas al sistema de la *PlayStation Portable* que nos dice esa información.

A continuación tenemos los ficheros externos. En nuestro caso existen dos procesos que son **Cargar ROM** y **guardar Juego**. La complejidad de las entradas es alta para **cargar ROM** y **guardar ROM**, ya que son procesos muy complejos de realizar al tener que interactuar con la tarjeta de memoria *Memory Stick*, el emulador de la *Master System 2* y la consola *PlayStation Portable*. Además **cargar ROM** sirve tanto para ejecutar un juego la primera vez como para seguir una partida desde el lugar donde nos encontrábamos anteriormente en el juego. Respecto a **guardar Juego**, también su complejidad es alta porque tenemos que guardar el estado de donde nos encontremos en la *ROM* con toda la información que conlleva.

Hay que tener en cuenta que no tenemos ficheros lógicos internos, ni consultas externas, ya que no existe una base de datos en nuestro sistema y no hay transacciones con ella al no existir, por lo que al no existir una base de datos no hay ficheros lógicos internos y al no haber dicha base de datos, no se realizan consultas externas.

Calculamos el PFNA:

$PFNA= 2_{in}x6 + 4_{in}x4 + 4_{out}x5 + 2_{fle}x10 = 68$

Ahora calculamos el factor de ajuste, siendo ΣFi la suma de los factores de complejidad:

$FA= (0'01x\Sigma Fi) + 0'65 = (0'01x27) + 0'65 = 0'92$

Por último, calcularemos los puntos de función ajustados usando los PFNA y el FA de la siguiente manera:

$PF= FAxPFNA = 0'92x68 = 62'56$

- ✓ **Parte 2- Estimación usando COCOMO:**

El Modelo Constructivo de Costes (o COCOMO, por su acrónimo del inglés COnstructive COst MOdel) es un modelo matemático de base empírica utilizado para estimación de costes de software. Incluye tres submodelos, cada uno ofrece un nivel de detalle y aproximación, cada vez mayor, a medida que avanza el proceso de desarrollo del software: básico, intermedio y detallado.

Este modelo fue desarrollado por Barry W. Boehm a finales de los años 70 y comienzos de los 80, exponiéndolo detalladamente en su libro "Software Engineering Economics".

Pertenece a la categoría de modelos de subestimaciones basados en estimaciones matemáticas. Está orientado a la magnitud del producto final, midiendo el "tamaño" del proyecto, en líneas de código principalmente.

Podremos usar uno de los tres siguientes modelos:

- Modelo orgánico: un pequeño grupo de programadores experimentados desarrollan software en un entorno familiar. El tamaño del software varía desde unos pocos miles de

líneas (tamaño pequeño) a unas decenas de miles (medio).

- Modelo semilibre: corresponde a un esquema intermedio entre el orgánico y el rígido; el grupo de desarrollo puede incluir una mezcla de personas experimentadas y no experimentadas.
- Modelo rígido: el proyecto tiene fuertes restricciones, que pueden estar relacionadas con la funcionalidad y/o pueden ser técnicas. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

Al ser un proyecto el que se va a realizar muy complejo existiendo fuertes restricciones y una gran innovación tecnológica, así como una interacción entre un sistema que no es una computadora de propósito general, sino una de propósito específica (*PlayStation Portable*) con un sistema que emula otra computadora de propósito específica (*Master System 2*), que en este caso es de mayor antigüedad y que tiene un contenedor que hace posible el poder usar dicho emulador, el modelo que se usa es el rígido:

MODO	**a**	**b**	**c**	**d**
Orgánico	2'4	1'05	2'5	0'38
Semilibre	3	1'12	2'5	0'35
Rígido	3'6	1'2	2'5	0'32

La tabla de líneas de código usando un lenguaje se muestra también a continuación:

Lenguaje	**LDC/PF**
Ensamblador	320
C	**150**
Cobol	106
Pascal	91
Basic	64
PCL	64
Java	53
C++	29

Sabiendo que los puntos de función ajustados nos han dado el resultado de 62’56 y que el lenguaje que se utilizará será C, debido a que es un lenguaje con el que se puede interactuar a un nivel casi bajo, aun siendo de alto nivel dicho lenguaje de programación y es de los pocos que nos permite una programación en lenguaje ensamblador para poder interactuar con los sistemas en cuestión que se relacionan entre ellos, hallaremos el número de líneas de código del programa:

LDC= Líneas de código del lenguaje x PF = 150x62’56 = 9384
LDC ≈ 9’384 KLDC

	Valor de los factores					
FACTORES	Muy bajo	Bajo	Medio	Alto	Muy alto	Extra
Fiabilidad requerida	0’75	0’88	1’00	1’15	**1’4**	
Tamaño de la base de datos		0’94	1’00	1’08	1’16	
Complejidad del software	0’70	0’85	1’00	1’15	1’30	**1’65**
Restricciones de tiempo de ejecución			1’00	1’11	**1’30**	1’66
Restricciones de memoria			**1’00**	1’06	1’21	1’56
Volatilidad del hardware		**0’87**	1’00	1’15	1’30	
Restricciones de tiempo de respuesta		**0’87**	1’00	1’07		
Calidad de los analistas	1’46	1’19	**1’00**	0’86	0’71	
Experiencia con el tipo de aplicación	**1’29**	1’13	1’00	0’91	0’82	
Experiencia con el hardware	**1’21**	1’10	1’00	0’90		
Experiencia con el lenguaje de programación	1’14	1’07	**1’00**	0’95		
Calidad de los programadores	1’42	1’17	**1’00**	0’86	0’70	
Técnicas modernas de programación	1’24	**1’10**	1’00	0’91	0’82	
Empleo de herramientas	1’24	1’10	1’00	**0’91**	0’83	
Restricciones a la duración del proyecto	1’23	1’08	1’00	1’04	**1’10**	

Se han elegido estos valores ya que se necesita una alta fiabilidad requerida de la aplicación para que se vea que funcione de

forma correcta, el tamaño de la base de datos no se presenta (no existe una base de datos). El software es de complejidad extra, ya que debe interactuar la PlayStation Portable con el emulador de la Master System 2 para poder ejecutarlo, así como el contenedor para el emulador para que el usuario pueda interactuar con ello. Las restricciones del tiempo de ejecución serán del valor muy alto al necesitar una rapidez de ejecución por parte de la aplicación. No habrá muchas restricciones de memoria, ya que con una Memory Stick de 32 MB podemos guardar la aplicación, así como las partidas guardadas de las ROMS y las ROMS de los videojuegos ejecutados por el emulador será más que suficiente. No habrá volatilidad del hardware. La calidad de los analistas será media al ser éstos el propio alumno y tener una experiencia demostrada a lo largo de la titulación. Al ser el alumno quien realiza la aplicación y no estar familiarizado con el tipo de aplicación, su experiencia es baja, así como su experiencia con el hardware en cuanto a la programación para dicho hardware. El nivel de experiencia con el lenguaje de programación será medio, al ser visto dicho lenguaje en las prácticas de Sistemas Operativos y estar ya familiarizado con dicho lenguaje. La calidad como programador es media, al estar habituado a realizar programas durante bastantes años y haberse incorporado al mundo laboral. Sin embargo, las técnicas modernas de programación no se pueden poner en práctica en el presente proyecto, al ser un proyecto no genérico y que se realiza muy pocas veces (en especial para un hardware distinto a una computadora), de ahí que su nivel sea muy bajo. El empleo de herramientas será también de un nivel alto, al saber manejar las herramientas necesitadas y una profunda lectura y comprensión de las librerías utilizadas para la realización del proyecto. Por último, las restricciones a la duración del proyecto deben ser de un valor alto al querer presentarse el proyecto en el presente año.

Se muestra en negrita los datos que se cogen para calcular el peso del factor de ajuste, el cual es el siguiente.

$m(x) = \prod m(x_j) =$
$1'4 \cdot 1'65 \cdot 1'30 \cdot 1'00 \cdot 0'87 \cdot 0'87 \cdot 1'00 \cdot 1.29 \cdot 1'21 \cdot 1'00 \cdot 1'00 \cdot 1'10 \cdot 0.91 \cdot 1'10 = 3'9065$

Teniendo ya el peso del factor de coste x_j calculado, podemos calcular el esfuerzo realizado de la aplicación en personas/mes:

$E= a*Kloc^b= 3'6 * 9'384^{1'2}= 52'86$ personas/mes

$T= cE^d= 2'5 * 52'86^{0'32}= 8'89$ meses

- ✓ **Parte 3- Presupuesto:**

En esta última parte realizaremos el presupuesto de nuestro proyecto llamado "EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE". Para ello, tendremos en cuenta la duración del proyecto, los recursos humanos y los recursos materiales utilizados.

HARDWARE	**USO (%)**	**COSTE TOTAL (€)**	**COSTE (€)**
Ordenador personal	300%	1356'00 €	116'23 €
Conexión a internet	100%	214'14 €	24'90 €
Impresora	100%	119'90 €	119'90 €
PlayStation Portable	100%	250'00 €	

SOFTWARE	USO (%)	COSTE TOTAL (€)	COSTE (€)
Windows 7	300%	909’30 €	129’90 €
StarUML	100%	0’00 €	0’00 €
Microsoft Office 2007	300%	839’30 €	119’90 €
DevC++	100%	0’00 €	0’00 €
Oracle VM Virtual Box 4.2.12	100%	0’00 €	0’00 €
Debian	100%	0’00 €	0’00 €
SDK PSP v 3.0	100%	0’00 €	0’00 €
PSPToolChain	100%	0’00 €	0’00 €

	TIEMPO	COSTE
Jefe de proyecto	480 horas	20’00 €/Hora
Analista	72 horas	15’00 €/Hora
Programador	600 horas	10’00 €/Hora

A continuación se muestra una información detallada de ello:

- Recursos humanos:
  - Jefe de proyecto:
    - Tasa estándar: 20'0 €/hora
    - Tasa sobretiempo: 23'0 €/hora
    - Costo por uso: 0 €
  - Analista:
    - Tasa estándar: 15'0 €/hora

        - Tasa sobretiempo: 18'0 €/hora
        - Costo por uso: 0 €
    - Programador:
        - Tasa estándar: 10'0 €/hora
        - Tasa sobretiempo: 13'0 €/hora
        - Costo por uso: 0 €
    - Recursos materiales:
        - Ordenador:
            - Costo por uso: 116’23 €
            - Cantidad: 3
            - Cantidad: 1
        - Impresora:
            - Costo por uso: 119’90 €
            - Cantidad: 1
            - Cantidad: 1
        - Conexión a Internet:
            - Costo por uso: 116’23 €
            - Cantidad: 3
            - Cantidad: 1
        - PlayStation Portable:
            - Costo por uso: 66’00 €
            - Cantidad: 3
            - Cantidad: 1
    - Duración:
        - Documentación:
            - Duración: 60 días
            - Persona que se ocupa: Jefe de proyecto
            - Horas por día: 8 horas

- Fase de análisis
  - Duración: 4 días
  - Persona que se ocupa: Analista
  - Horas por día: 8 horas
    - Objetivos del sistema:
      - Duración: 1 días
      - Persona que se ocupa: Analista
      - Horas por día: 8 horas
    - Requisitos de información:
      - Duración: 1 días
      - Persona que se ocupa: Analista
      - Horas por día: 8 horas
    - Requisitos funcionales:
      - Duración: 1 días
      - Persona que se ocupa: Analista
      - Horas por día: 8 horas
    - Casos de uso:
      - Duración: 1 días
      - Persona que se ocupa: Analista
      - Horas por día: 8 horas
    - Definición de actores:
      - Duración: 1 días
      - Persona que se ocupa: Analista
      - Horas por día: 8 horas
    - Requisitos no funcionales:
      - Duración: 1 días
      - Persona que se ocupa: Analista
      - Horas por día: 8 horas

- Fase de Diseño
  - Diagrama de Casos de Uso:
    - Duración: 3 días
    - Persona que se ocupa: Analista
    - Horas por día: 8 horas
  - Diagrama de Paquetes:
    - Duración: 1 días
    - Persona que se ocupa: Analista
    - Horas por día: 8 horas
  - Diagrama de estructura:
    - Duración: 1 días
    - Persona que se ocupa: Analista
    - Horas por día: 8 horas
- Implementación:
  - Duración: 60 días
  - Persona que se ocupa: programador
  - Horas por día: 8 horas
- Pruebas:
  - Duración: 15 días
  - Persona que se ocupa: programador
  - Horas por día: 8 horas

Por lo tanto, el costo del proyecto resultante será:

- Documentación= 9386’13 €
- Especificación de requisitos= 381’03 Euros
  - Objetivos del sistema= 0 €
  - Requisitos no funcionales= 0 €

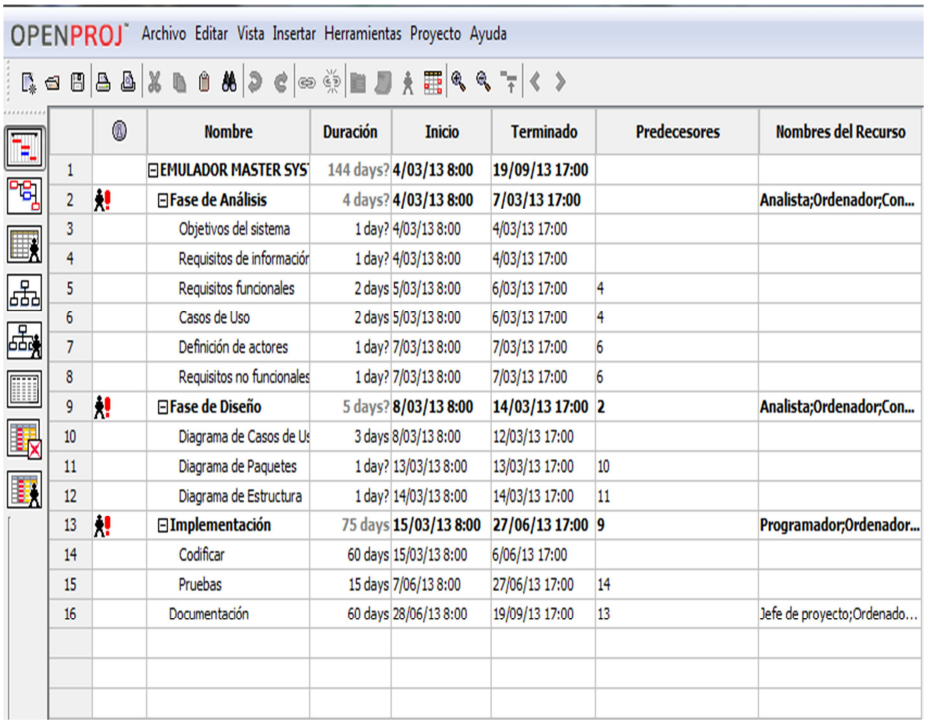
        - Requisitos de información= 0 €
        - Requisitos funcionales= 0 €
    - Diseño= 381’03 €
        - Diagrama de Clases= 1008'0 €
        - Diagrama de Secuencia= 1872'0 €
        - Diagrama de Casos de Uso= 864'0 €
        - Diagrama de Estados= 720'0 €
        - Diagrama de Objetos= 720'0 €
    - Implementación= 407’03 €
        - Pruebas= 0 €
    - Ordenadores= 116’23x3 = 348’69 €
    - Conexión a Internet: 24’90x5 = 124’50 €
    - Impresora: 119’00 €
    - PlayStation Portable: 66’00 €
    - Windows 7:129’90x3 = 389’70 €
    - StarUML: 0’00 €
    - Microsoft Office 2007: 119’90x3 = 359’70 €
    - DevC++ IDE: 0’00 €
    - Oracle VM Virtual Box 4.2.12: 0’00 €
    - Debian: 0’00 €
    - Kit de Programación PSP 3.0: 0’00 €
    - Total= 12462’81 €

4.4.- CALENDARIZACIÓN DEL PROYECTO

En este apartado se mostrará el desarrollo de las fases del proyecto, así como los recursos asignados a las distintas fases. Todo ello ha sido realizado con la herramienta de *Open Project*.

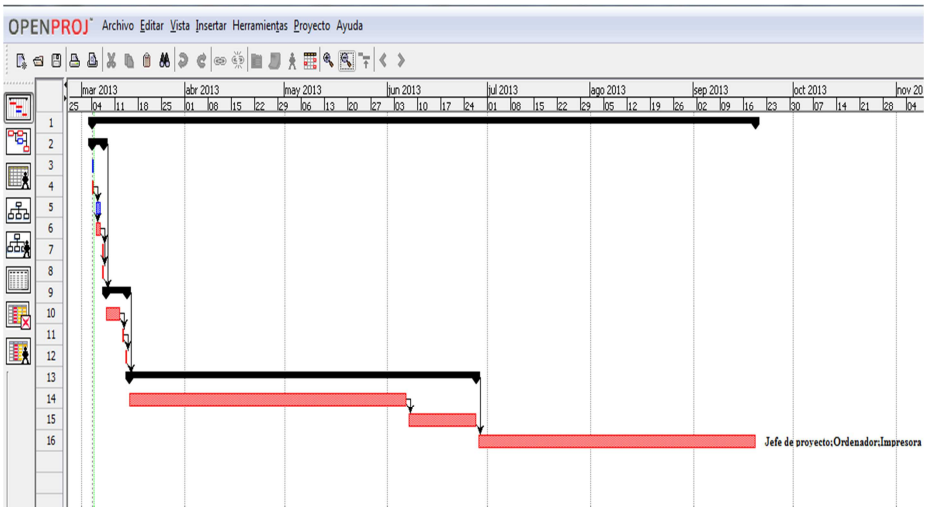
Primero se verá el *Diagrama de Gantt* con las fases realizadas

en una tabla, para a continuación ver los recursos y por último mostrar el *Diagrama de Pert*.

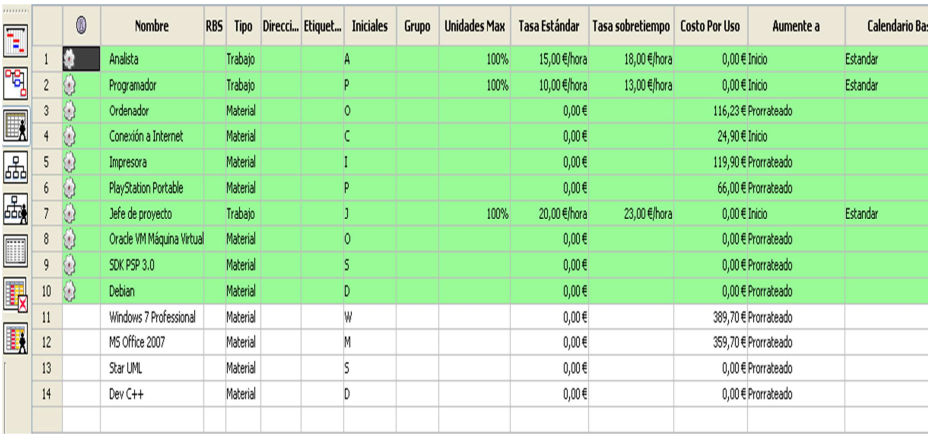


		Nombre	Duración	Inicio	Terminado	Predecesores	Nombres del Recurso
1		EMULADOR MASTER SYS	144 days?	4/03/13 8:00	19/09/13 17:00		
2		Fase de Análisis	4 days?	4/03/13 8:00	7/03/13 17:00		Analista;Ordenador;Con...
3		Objetivos del sistema	1 day?	4/03/13 8:00	4/03/13 17:00		
4		Requisitos de informació	1 day?	4/03/13 8:00	4/03/13 17:00		
5		Requisitos funcionales	2 days	5/03/13 8:00	6/03/13 17:00	4	
6		Casos de Uso	2 days	5/03/13 8:00	6/03/13 17:00	4	
7		Definición de actores	1 day?	7/03/13 8:00	7/03/13 17:00	6	
8		Requisitos no funcionales	1 day?	7/03/13 8:00	7/03/13 17:00	6	
9		Fase de Diseño	5 days?	8/03/13 8:00	14/03/13 17:00	2	Analista;Ordenador;Con...
10		Diagrama de Casos de Us	3 days	8/03/13 8:00	12/03/13 17:00		
11		Diagrama de Paquetes	1 day?	13/03/13 8:00	13/03/13 17:00	10	
12		Diagrama de Estructura	1 day?	14/03/13 8:00	14/03/13 17:00	11	
13		Implementación	75 days	15/03/13 8:00	27/06/13 17:00	9	Programador;Ordenador...
14		Codificar	60 days	15/03/13 8:00	6/06/13 17:00		
15		Pruebas	15 days	7/06/13 8:00	27/06/13 17:00	14	
16		Documentación	60 days	28/06/13 8:00	19/09/13 17:00	13	Jefe de proyecto;Ordenado...

Se puede ver en la primera parte del *Diagrama de Gantt* las fechas perfectamente en las que el Proyecto Fin de Carrera comenzó y termina, así como la duración de cada fase y los recursos asignados a cada fase.

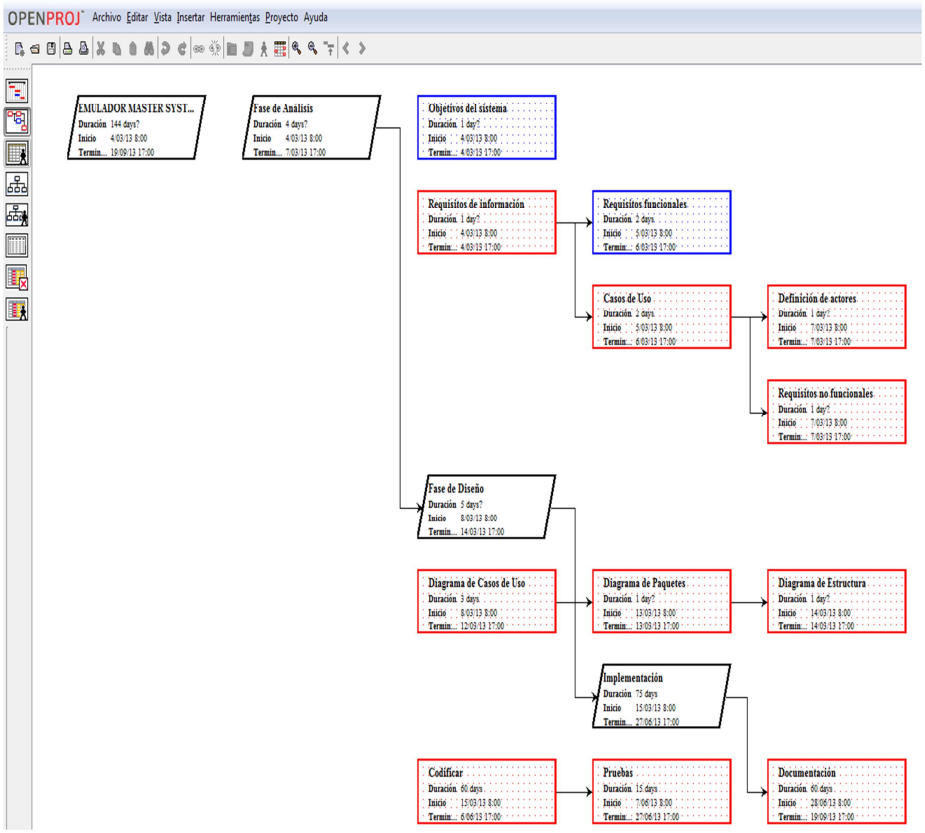


En la segunda parte del Diagrama de Gantt vemos la evolución de las fechas según las tareas de las fases que se realizan, así como los recursos que han sido asignados, todo ello mediante barras.



	Nombre	RBS	Tipo	Direcci...	Etiquet...	Iniciales	Grupo	Unidades Max	Tasa Estándar	Tasa sobretiempo	Costo Por Uso	Aumente a	Calendario Ba
1	Analista		Trabajo			A		100%	15,00 €/hora	18,00 €/hora	0,00 €	Inicio	Estandar
2	Programador		Trabajo			P		100%	10,00 €/hora	13,00 €/hora	0,00 €	Inicio	Estandar
3	Ordenador		Material			O			0,00 €		116,23 €	Prorrateado	
4	Conexión a Internet		Material			C			0,00 €		24,90 €	Inicio	
5	Impresora		Material			I			0,00 €		119,90 €	Prorrateado	
6	PlayStation Portable		Material			P			0,00 €		66,00 €	Prorrateado	
7	Jefe de proyecto		Trabajo			J		100%	20,00 €/hora	23,00 €/hora	0,00 €	Inicio	Estandar
8	Oracle VM Máquina Virtual		Material			O			0,00 €		0,00 €	Prorrateado	
9	SDK PSP 3.0		Material			S			0,00 €		0,00 €	Prorrateado	
10	Debian		Material			D			0,00 €		0,00 €	Prorrateado	
11	Windows 7 Professional		Material			W			0,00 €		389,70 €	Prorrateado	
12	MS Office 2007		Material			M			0,00 €		359,70 €	Prorrateado	
13	Star UML		Material			S			0,00 €		0,00 €	Prorrateado	
14	Dev C++		Material			D			0,00 €		0,00 €	Prorrateado	

En esta tabla se ven todos los recursos existentes en la realización del proyecto, tanto materiales como humanos, con sus costes por uso en el caso de los recursos materiales y las tasas estándar y de sobretiempo en los recursos humanos.



En este diagrama, el cual es llamado Diagrama de Pert o Diagrama de red se pueden ver todas las fases, con sus fechas y la duración de cada una de ellas y cómo se encuentran relacionadas entre ellas.

PARTE 2:

DESARROLLO DEL SISTEMA

5.- REQUISITOS DEL SISTEMA

El análisis del sistema es el proceso del estudio de las necesidades de los usuarios para llegar a una definición de requisitos del sistema, así como el refinamiento de dichos requisitos. Dicho esto, la definición de requisito es la que sigue a continuación

Un requisito es una condición o capacidad que necesita el usuario para resolver un problema o conseguir un determinado objetivo.

El modelo de análisis constituye una abstracción resumida y precisa de lo que debe hacer el sistema que se va a desarrollar y no la forma en cómo se desarrollará, es decir, nos dice el qué hacer pero no el cómo hacerlo. Esto es importante, ya que al no incluirse en esta parte estructuras de la implementación, un buen modelo puede ser visto por personas expertas de la aplicación a desarrollar que no sean programadores.

Al proponer el proyecto llamado “EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE” se fijaron una serie de requisitos funcionales para la integración de la aplicación, así como de requisitos no funcionales, casos de uso, requisitos de información y definición de actores que se pasará a desglosar a continuación.

5.1.- REQUISITOS DE INFORMACIÓN

Los requisitos de información que se encuentran presentes en el proyecto corresponden a dos grupos diferenciados. Estos grupos son:

- ✓ Información que atañe a las ROMS.
- ✓ Información acerca del contenedor del sistema.

Como se puede ver en las siguientes tablas, se desglosa de forma detallada y de acuerdo al estándar exigido en la asignatura de **Ingeniería del Software 1** los siguientes requisitos de información.

IRQ-01	***Información de las ROMS***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador.	
Requisitos asociados	UC–1_01 Menú_Principal UC–1_02 Cargar_ROM UC–1_03 Cargar_Guardar_Juego UC–1_10 Sistema UC–1_11 Audio UC–1_12 Vídeo2 UC–1_13 System	
Descripción	El sistema deberá almacenar la información correspondiente a las ROMS, así como la carpeta en la que se encuentran. ROM: representa la memoria en la que está almacenado un juego que podrá ser ejecutado por la Master System 2.	
Datos específicos	*Nombre de la ROM* *Carpeta en que se encuentran las ROMS*	
Tiempo de vida	**Medio**	**Máximo**
	7 años	*10 años*
Ocurrencias simult.	**Medio**	**Máximo**
Importancia	*Alta*	
Urgencia	*Media*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	Ninguno	

IRQ-02	***Información de los controles***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–03 Configuración del emulador.	
Requisitos asociados	UC–1_09 Controles	
Descripción	El sistema deberá mostrar y almacenar la información correspondiente a os controles con los que se podrá manejar los juegos ejecutados por el emulador, así como los controles que manejan el contenedor del emulador.	
Datos específicos	*Nombre de los controles*	
Tiempo de vida	**Medio**	**Máximo**
	7 años	*10 años*
Ocurrencias simult.	**Medio**	**Máximo**
Importancia	*Alta*	
Urgencia	*Media*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	Ninguno	

IRQ-03	***Información de opciones***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–03 Configuración del emulador.	
Requisitos asociados	UC–1_04 Opciones UC–1_05 Vídeo UC–1_06 Entrada UC–1_07 Rendimiento UC–1_08 Menú	
Descripción	El sistema deberá mostrar y almacenar la información correspondiente a la configuración existente del emulador. Las opciones de configuración realmente lo son del contenedor (proceso PSP) en el que se ejecuta el emulador.	
Datos específicos	*Tamaño de la pantalla* *Ratio del manejador de la entrada* *Rendimiento del emulador* *Otras opciones de menú*	
Tiempo de vida	**Medio**	**Máximo**
	7 años	*10 años*
Ocurrencias simult.	**Medio**	**Máximo**
Importancia	*Alta*	
Urgencia	*Media*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	Ninguno	

IRQ-01_01	***Información de partidas guardadas***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador.	
Requisitos asociados	UC–1_01 Cargar_ROM UC–1_03 Cargar_Guardar_Juego	
Descripción	El sistema deberá mostrar y almacenar la información correspondiente a la partida guardada de una ROM. Esto se conoce como suspender/activar la ejecución de un programa de la PSP con su estado de ejecución actual.	
Datos específicos	*Imagen del juego* *Fecha de la partida guardada* *Hora de la partida guardada*	
Tiempo de vida	**Medio**	**Máximo**
	7 años	*10 años*
Ocurrencias simult.	**Medio**	**Máximo**
Importancia	*Alta*	
Urgencia	*Media*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	Ninguno	

IRQ-01_02	***Información del sistema***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador.	
Requisitos asociados	UC–1_01 Cargar_ROM UC–1_10 Sistema UC–1_11 Audio UC–1_12 Vídeo2 UC–1_13 System	
Descripción	El sistema deberá mostrar y almacenar la información correspondiente a una ROM jugada en ese momento concreto, siendo éstos una imagen mostrada del juego, si soporta sonido mediante la emulación FM del chip de Yamaha emulado, etc..	
Datos específicos	*Imagen del juego al que se juega (screenshoot)* *Opciones de audio* *Opciones de vídeo*	
Tiempo de vida	**Medio**	**Máximo**
Ocurrencias simult.	*7 años* **Medio**	*10 años* **Máximo**
Importancia	 *Alta*	
Urgencia	*Media*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	Ninguno	

CRQ-01	***Número máximo de partidas guardadas***
Versión	*1.0*
Autores	*Jorge García Flores*
Fuentes	
Objetivos asociados	OBJ–02 Ejecutar juegos en el emulador.
Requisitos asociados	IRQ–01_01 Información de partidas guardadas UC–1_01 Cargar_ROM UC–1_03 Cargar_Guardar_Juego
Descripción	*El emulador será capaz de guardar hasta 10 partidas simultáneas de distintos juegos*
Importancia	*Media*
Urgencia	*Baja*
Estado	*Validado*
Estabilidad	*Alta*
Comentarios	Ninguno

CRQ-02	***Cambio de opciones sin resetear emulador***
Versión	*1.0*
Autores	*Jorge García Flores*
Fuentes	
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–03 Configuración del emulador.
Requisitos asociados	UC–1_04 Opciones UC–1_05 Vídeo UC–1_06 Entrada UC–1_07 Rendimiento UC–1_08 Menú UC–1_10 Sistema UC–1_11 Audio UC–1_12 Vídeo2 UC–1_13 System
Descripción	*El usuario podrá cambiar las opciones de la configuración del emulador sin necesidad de resetear el sistema para ello.*
Importancia	*Media*
Urgencia	*Baja*
Estado	*Validado*
Estabilidad	*Alta*
Comentarios	Ninguno

5.2.- REQUISITOS FUNCIONALES

El objetivo del presente proyecto es que un usuario de una *PlayStation Portable* pueda ejecutar un emulador de *Master System 2* para poder jugar a una serie de *ROMS*. Tanto el emulador como las *ROMS* deberán ser almacenadas en una tarjeta de memoria *Memory Stick* de mínimo 32 MB de memoria.

El presente proyecto se encuentra divido en dos partes claramente diferenciadas que son el emulador de la *Master System 2* y el contenedor de la aplicación, el cual será una interfaz de usuario gráfica y que hará de capa de virtualización.

El usuario podrá realizar una serie de opciones que consisten en jugar a una *ROM* que se encuentre previamente guardada en una *Memory Stick*, guardar y cargar una partida de una *ROM* que se encuentre ejecutándose en el emulador *Master System 2* y cambiar distintos parámetros de configuración, tanto del contenedor, como de las *ROMS*.

A continuación se presenta de forma totalmente detallada y desglosada los requisitos funcionales que han sido detectados para la realización del proyecto.

RF-01	***Emulador guardado en Memory Stick***
Versión	*1.0*
Autores	*Jorge García Flores*
Fuentes	
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador.
Requisitos asociados	IRQ–02 Información de los controles IRQ–03 Información de opciones IRQ–01_02 Información del sistema
Descripción	*La información almacenada por el sistema deberá satisfacer la siguiente restricción: el emulador deberá ser almacenado en una Memory Stick de mínimo 32Mb para ser ejecutado.*
Importancia	*Imprescindible*
Urgencia	*Alta*
Estado	*Validado*
Estabilidad	*Alta*
Comentarios	Ninguno

RF-02	***ROMS guardadas en Memory Stick***
Versión	*1.0*
Autores	*Jorge García Flores*
Fuentes	
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–03 Configuración del emulador.
Requisitos asociados	IRQ–01 Información de las ROMS IRQ–01_01 Información de partidas guardadas IRQ–01_02 Información del sistema
Descripción	*La información almacenada por el sistema deberá satisfacer la siguiente restricción: el emulador deberá ser almacenado en una Memory Stick de mínimo 32Mb para ser ejecutado.*
Importancia	*Imprescindible*
Urgencia	*Alta*
Estado	*Validado*
Estabilidad	*Alta*
Comentarios	Ninguno

Una vez definidos los requisitos funcionales se pasa a definir los requisitos no funcionales de la aplicación.

5.3.- REQUISITOS NO FUNCIONALES

Se consideran los requisitos no funcionales aquellos que deben tenerse en cuenta para que la aplicación funcione correctamente presentándose una optimización en el diseño del proyecto. En el caso del proyecto desarrollado hay que tener en cuenta dos aspectos.

- ✓ El relacionado con el sistema de almacenamiento de la videoconsola *PlayStation Portable.*
- ✓ El relacionado con las *ROMS.*

Dicho esto, se pasa a detallar de forma desglosada los requisitos no funcionales encontrados en la etapa de búsqueda de requisitos, presentándose también en forma de tabla.

NFR–*01*	***Memory Stick de mínimo 32 Mb***
Versión	*1.0*
Autores	*Jorge García Flores*
Fuentes	
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador. OBJ–03 Configuración del emulador.
Requisitos asociados	IRQ–01 Información de las ROMS IRQ–01_01 Información de partidas guardadas IRQ–01_02 Información del sistema IRQ–02 Información de los controles IRQ–03 Información de opciones
Descripción	La Memory Stick será de mínimo 32 Mb para poder guardar el emulador en ella y ser ejecutado correctamente.
Importancia	*Imprescindible*
Urgencia	*Media*
Estado	*Validado*
Estabilidad	*Alta*
Comentarios	*Ninguno*

NFR–*02*	***Rapidez en la ejecución del emulador***
Versión	*1.0*
Autores	*Jorge García Flores*
Fuentes	
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador. OBJ–03 Configuración del emulador.
Requisitos asociados	IRQ–01 Información de las ROMS IRQ–01_01 Información de partidas guardadas IRQ–01_02 Información del sistema IRQ–02 Información de los controles IRQ–03 Información de opciones
Descripción	El emulador deberá ser ejecutado lo más rápido posible para permitir un uso eficiente de él.
Importancia	*Imprescindible*
Urgencia	*Media*
Estado	*Validado*
Estabilidad	*Alta*
Comentarios	*Ninguno*

NFR–*03*	***Rapidez en la ejecución de las ROMS***
Versión	*1.0*
Autores	*Jorge García Flores*
Fuentes	
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador. OBJ–03 Configuración del emulador.
Requisitos asociados	IRQ–01 Información de las ROMS IRQ–01_01 Información de partidas guardadas IRQ–01_02 Información del sistema IRQ–02 Información de los controles IRQ–03 Información de opciones
Descripción	El emulador deberá ejecutar las ROMS lo más rápido posible y que el usuario no espere más de lo necesario.
Importancia	*Imprescindible*
Urgencia	*Media*
Estado	*Validado*
Estabilidad	*Alta*
Comentarios	*Ninguno*

Como se puede ver, existe un requisito no funcional relacionado con el sistema de almacenamiento de la videoconsola *PlayStation*

Portable que es que se necesita un tamaño mínimo de la tarjeta de memoria *Memory Stick*.

Además, la aplicación debe ser rápida a la hora de ejecutar el emulador *Master System 2* y la ejecución de las *ROMS* para que el usuario tenga una experiencia satisfactoria con el producto.

5.3.1.- DEFINICIÓN DE ACTORES

Como actor, me refiero a aquella persona susceptible de ser usuario y que interactúe con el sistema desarrollado mediante la videoconsola *PlayStation Portable* y pueda usar el producto.

El usuario es persistente a lo largo del ciclo de duración del sistema a lo largo de todo su tiempo de vida, siendo un receptor de datos proporcionados por la salida de la aplicación, influyendo en ellos mediante la elección de una serie de diversas de opciones dadas por el sistema.

Nuestro usuario no es de ningún tipo especial pues no se pide un registro ni necesitando una seguridad para que pueda acceder a ella según unos permisos establecidos, sino que es un usuario normal que puede acceder a todas las funcionalidades de la aplicación que ésta le propone y se encuentran diseñadas e implementadas.

A continuación en la siguiente tabla se detalla al actor en cuestión.

ACT–*01*	Usuario
Versión	*1.0*
Autores	*Jorge García Flores*
Fuentes	
Descripción	Este actor representa a los usuarios que ejecutan el emulador en una Playstation Portable para jugar.
Comentarios	Ninguno

5.3.2.- CASOS DE USO

Un caso de uso es la descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Existen una serie de personajes que participarán en algún caso de uso y que se llaman actores. Como se ha visto anteriormente, el actor ha sido definido e identificado.

Dicho esto, y con una definición que proviene de la ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollan entre un sistema y sus actores respecto a un evento que inicia un actor principal sobre el propio sistema.

En el proyecto que se presenta, un usuario es el que desencadenará la ejecución de todo y que como se podrá ver a continuación en el Diagrama de Casos de Uso podrá acceder a una serie de funcionalidades cuando ocurra algún evento en particular.

Así mismo, se detallan de una forma totalmente desglosada en forma de tabla como se ha visto en la asignatura de **Ingeniería del Software 1**.

UC-01	***Menú_Principal***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador. OBJ–03 Configuración del emulador.	
Requisitos asociados	IRQ–01 Información de las ROMS IRQ–02 Información de los controles IRQ–03 Información de opciones	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario ejecute la aplicación del emulador.	
Precondición	*El emulador ha sido cargado correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario enciende su Playstation Portable.
	2	El usuario ejecuta el emulador.
	3	Carga el emulador.
	4	Se accede al menú principal.
Postcondición	*El usuario elige entre las distintas opciones del menú principal.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador no se ejecuta correctamente, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*5 s*
	2	*20 ms*
	3	*20 ms*
	4	*5 ms*
Frecuencia	*1 vez / ejecución*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	

UC-02	***Cargar_ROM***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador.	
Requisitos asociados	IRQ–01 Información de las ROMS UC–1_03 Cargar_Guardar_Juego UC–1_10 Sistema UC–1_11 Audio UC–1_12 Vídeo2 UC–1_13 System	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario cargue una ROM en el emulador.	
Precondición	*El emulador ha sido cargado correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema la lista de ROMs disponibles.
	2	El sistema muestra las ROMs disponibles.
	3	El usuario elige la ROM.
	4	El emulador carga la ROM.
	5	El usuario comienza la partida de dicha ROM.
Postcondición	*El usuario comienza una partida a dicha ROM desde cero.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador no carga la ROM correctamente, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*5 s*
	2	*20 ms*
	3	*20 ms*
Frecuencia	*1 vez / ejecución de ROM*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	

UC-03	***Guardar_Juego***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador.	
Requisitos asociados	IRQ–01 Información de las ROMS IRQ–01_01 Información de partidas guardadas IRQ–01_02 Información del sistema UC–1_02 Cargar_ROM UC–1_10 Sistema UC–1_11 Audio UC–1_12 Vídeo2 UC–1_13 System	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario guarde la partida de una ROM en el emulador.	
Precondición	*El emulador y la ROM han sido cargados correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema la lista de ROMs disponibles.
	2	El sistema muestra las ROMs.
	3	El usuario elige la ROM.
	4	El emulador carga la ROM.
	5	El usuario comienza la partida de dicha ROM.
	6	El usuario carga o guarda una partida de dicha ROM.
Postcondición	*El usuario continúa la partida de dicha ROM en el lugar que se había quedado anteriormente.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador no carga la ROM correctamente, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*5 s*
	2	*20 ms*
	3	*20 ms*
	4	*1 s*
Frecuencia	*1 vez / ejecución de ROM*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	

UC-04	***Cargar_Juego***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador.	
Requisitos asociados	IRQ–01 Información de las ROMS IRQ–01_01 Información de partidas guardadas IRQ–01_02 Información del sistema UC–1_02 Cargar_ROM UC–1_10 Sistema UC–1_11 Audio UC–1_12 Vídeo2 UC–1_13 System	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario cargue la partida de una ROM en el emulador.	
Precondición	*El emulador y la ROM han sido cargados correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema la lista de ROMs disponibles.
	2	El sistema muestra las ROMs disponibles.
	3	El usuario elige la ROM.
	4	El emulador carga la ROM.
	5	El usuario comienza la partida de dicha ROM.
	6	El usuario carga o guarda una partida de dicha iROM.
Postcondición	*El usuario continúa la partida de dicha ROM en el lugar que se había quedado anteriormente. En ese momento se juega a una iROM, la cual es una ROM más el estado en ejecución de la partida guardada anteriormente.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador no carga la ROM correctamente, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*5 s*
	2	*20 ms*
	3	*20 ms*
	4	*1 s*
Frecuencia	*1 vez / ejecución de ROM*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	

UC-05	***Opciones del emulador***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–03 Configuración del emulador.	
Requisitos asociados	IRQ–03 Información de opciones UC–1_01 Menú_Principal UC–1_05 Vídeo UC–1_06 Entrada UC–1_07 Rendimiento UC–1_08 Menú	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario elige opciones del emulador (vídeo entrada, rendimiento, configurar botón de cancelar en PSP) en el menú de la interfaz del emulador.	
Precondición	*El emulador ha sido cargado correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	Carga el emulador.
	2	El usuario elige entre las distintas opciones.
	3	El sistema establece las nuevas opciones de configuración para el contenedor del emulador.
Postcondición	*El usuario ha modificado las opciones disponibles del emulador.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador no carga, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*20 ms*
	2	*10 ms*
Frecuencia	*1 vez / ejecución*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	

UC-06	***Vídeo***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–03 Configuración del emulador.	
Requisitos asociados	IRQ–03 Información de opciones UC–1_04 Opciones	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica las opciones de vídeo del emulador.	
Precondición	*El emulador ha sido cargado correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	Carga el emulador.
	2	El usuario elige la opción de Vídeo.
	3	El usuario modifica la visualización del contenedor del emulador
Postcondición	*El usuario ha modificado las opciones disponibles del emulador.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador no carga, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*20 ms*
	2	*10 ms*
	3	*5 s*
Frecuencia	*1 vez / ejecución*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	

UC-07	***Entrada***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–03 Configuración del emulador.	
Requisitos asociados	IRQ–03 Información de opciones UC–1_04 Opciones	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica las opciones de entrada del emulador.	
Precondición	*El emulador ha sido cargado correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	Carga el emulador.
	2	El usuario elige la opción de Entrada.
	3	El usuario modifica la entrada del emulador
Postcondición	*El usuario ha modificado las opciones disponibles del emulador.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador no carga, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*20 ms*
	2	*10 ms*
	3	*5 s*
Frecuencia	*1 vez / ejecución*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	

UC-08	***Rendimiento***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–03 Configuración del emulador.	
Requisitos asociados	IRQ–03 Información de opciones UC–1_04 Opciones	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica las opciones de rendimiento del emulador respecto a la Playstation Portable.	
Precondición	*El emulador ha sido cargado correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	Carga el emulador.
	2	El usuario elige la opción de Rendimiento.
	3	El usuario modifica el rendimiento del emulador
Postcondición	*El usuario ha modificado las opciones disponibles del emulador.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador no carga, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*20 ms*
	2	*10 ms*
	3	*5 s*
Frecuencia	*1 vez / ejecución*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	

UC-09	***Setup botón cancelar PSP***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–03 Configuración del emulador.	
Requisitos asociados	IRQ–03 Información de opciones UC–1_04 Opciones	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica las opciones de manejar el menú del emulador teniendo en cuenta si la Playstation Portable es japonesa o del resto del mundo, es decir, los botones X y O.	
Precondición	*El emulador ha sido cargado correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	Carga el emulador.
	2	El usuario elige la opción de Menú.
	3	El usuario modifica el manejo del menú del emulador
Postcondición	*El usuario ha modificado las opciones disponibles del emulador.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador no carga, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*20 ms*
	2	*10 ms*
	3	*5 s*
Frecuencia	*1 vez / ejecución*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	

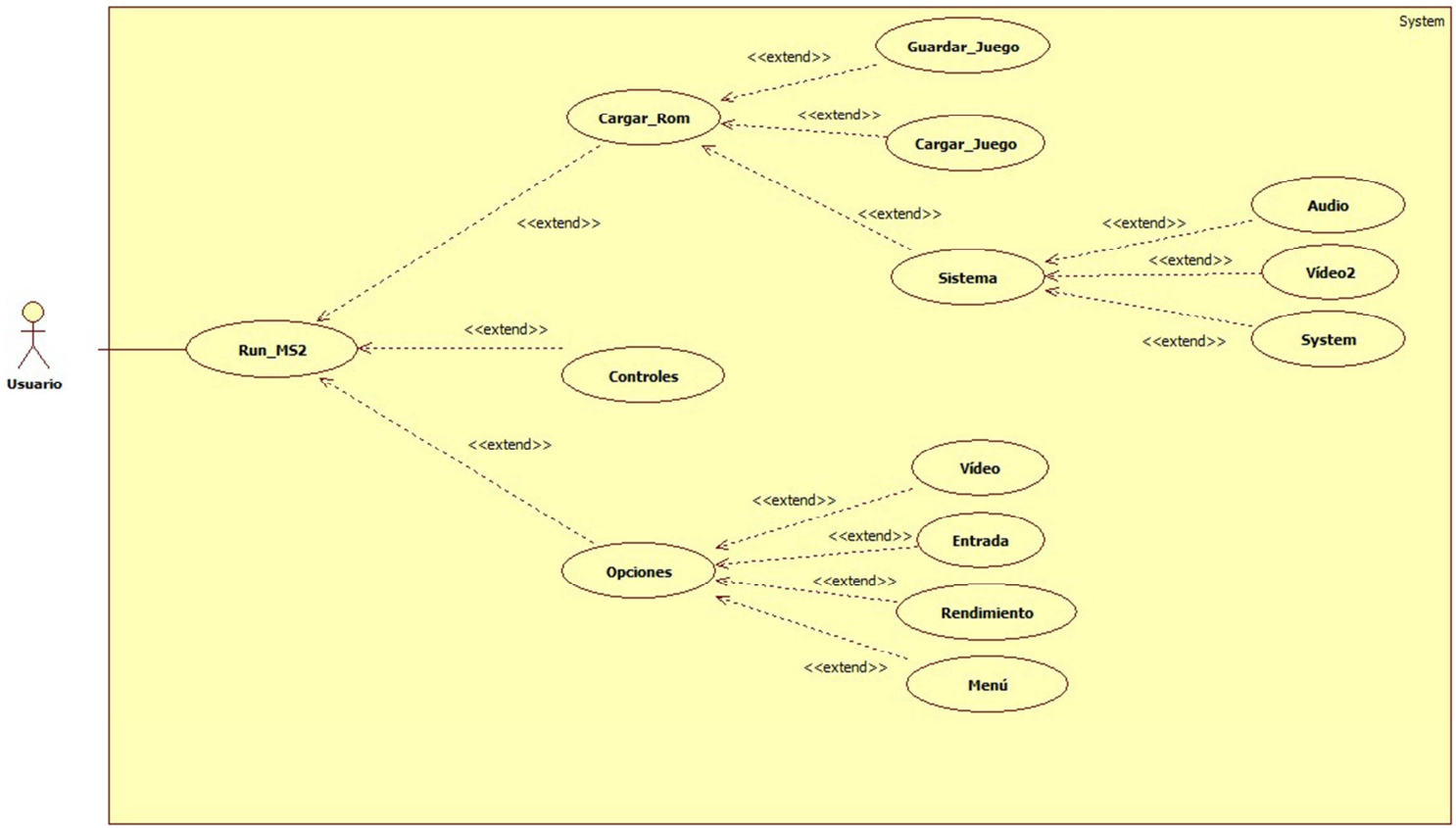
UC-10	***Controles***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–03 Configuración del emulador.	
Requisitos asociados	IRQ–02 Información de los controles UC–1_01 Menú_Principal	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica los controles cuando realiza una partida a una ROM.	
Precondición	*El emulador ha sido cargado correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	Carga el emulador.
	2	El usuario elige la opción de Controles.
	3	El usuario modifica los controles para jugar a las ROMS
Postcondición	*El usuario ha modificado las opciones disponibles del emulador.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador no carga, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*20 ms*
	2	*10 ms*
	3	*5 s*
Frecuencia	*1 vez / ejecución*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Configurar mapping (asociación) entre los botones de la PSP y los controles de la Master System 2.*	

UC-11	***Sistema***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador.	
Requisitos asociados	IRQ–01 Información de las ROMS UC–1_11 Audio UC–1_12 Vídeo2 UC–1_13 System	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica los parámetros de una ROM.	
Precondición	*El emulador ha sido cargado correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema la lista de las ROMs disponibles.
	2	El sistema muestra las ROMs disponibles.
	3	El usuario elige la ROM.
	4	El emulador carga la ROM.
	5	El usuario modifica los parámetros de la ROM.
Postcondición	*El usuario ha modificado las opciones disponibles del emulador.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador o la ROM no cargan, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*5 s*
	2	*10 ms*
	3	*5 s*
Frecuencia	*1 vez / ejecución*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	

UC-12	***Audio***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador.	
Requisitos asociados	IRQ–01 Información de las ROMS UC–1_10 Sistema	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica los parámetros de audio una ROM.	
Precondición	*El emulador ha sido cargado correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema la lista de las ROMs disponibles.
	2	El sistema muestra las ROMs disponibles.
	3	El usuario elige la ROM.
	4	El emulador carga la ROM.
	5	El usuario modifica los parámetros de audio de la ROM.
Postcondición	*El usuario ha modificado las opciones disponibles del emulador.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador o la ROM no cargan, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*5 s*
	2	*10 ms*
	3	*5 s*
Frecuencia	*1 vez / ejecución*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	

UC-13	***Vídeo2***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador.	
Requisitos asociados	IRQ–01 Información de las ROMS UC–1_10 Sistema	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica los parámetros de vídeo una ROM, en este caso una barra vertical en la zona izquierda de la pantalla.	
Precondición	*El emulador ha sido cargado correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema la lista de las ROMs disponibles.
	2	El sistema muestra las ROMs disponibles.
	3	El usuario elige la ROM.
	4	El emulador carga la ROM.
	5	El usuario modifica los parámetros de vídeo de la ROM.
Postcondición	*El usuario ha modificado las opciones disponibles del emulador.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador o la ROM no cargan, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*5 s*
	2	*10 ms*
	3	*5 s*
Frecuencia	*1 vez / ejecución*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	

UC-14	***System***	
Versión	*1.0*	
Autores	*Jorge García Flores*	
Fuentes		
Objetivos asociados	OBJ–01 Ejecutar emulador Master System 2 en PSP. OBJ–02 Ejecutar juegos en el emulador.	
Requisitos asociados	IRQ–01 Información de las ROMS UC–1_10 Sistema	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica los parámetros de sistema una ROM, pudiendo resetear la ROM o hacer una captura de pantalla de la ROM.	
Precondición	*El emulador ha sido cargado correctamente.*	
Secuencia normal	**Paso**	**Acción**
	1	El usuario solicita al sistema la lista de las ROMs disponibles.
	2	El sistema muestra las ROMs disponibles.
	3	El usuario elige la ROM.
	4	El emulador carga la ROM.
	5	El usuario modifica los parámetros de sistema de la ROM.
Postcondición	*El usuario ha modificado las opciones disponibles del emulador.*	
Excepciones	**Paso**	**Acción**
	1	Si el emulador o la ROM no cargan, a continuación este caso de uso queda sin efecto.
Rendimiento	**Paso**	**Cota de tiempo**
	1	*5 s*
	2	*10 ms*
	3	*5 s*
Frecuencia	*1 vez / ejecución*	
Importancia	*Imprescindible*	
Urgencia	*Alta*	
Estado	*Validado*	
Estabilidad	*Alta*	
Comentarios	*Ninguno*	


System
Guardar_Juego
<<extend>>
Cargar_Rom
<<extend>>
Cargar_Juego
<<extend>>
<<extend>>
Audio
<<extend>>
Sistema
<<extend>>
Vídeo2
<<extend>>
System
Usuario
Run_MS2
<<extend>>
Controles
<<extend>>
Vídeo
<<extend>>
Entrada
<<extend>>
Opciones
<<extend>>
Rendimiento
<<extend>>
Menú

Como se puede ver en el Diagrama de Casos de Uso, el actor **Usuario** es el que inicia todo el proceso de ejecutar la aplicación una vez que se ha encendido la videoconsola *PlayStation Portable*, siendo lo primero ejecutar el contenedor y una vez hecho eso, mediante relaciones *extend* que nos indican que son opciones que pueden realizarse o no, podremos ejecutar una *ROM*, cambiar la configuración de los controles o cambiar una serie de parámetros del contenedor y la propia *PlayStation Portable*.

En el caso de que se ejecute una *ROM* se relaciona este caso de uso (**Cargar_Rom**) con otros tres casos de uso mediante relaciones extend. Estos casos de uso son **Guadar_Juego**, **Cargar_Juego** y **Sistema**, y a su vez el caso de uso Sistema se relaciona mediante relaciones *extend* con los casos de uso **Audio**, **Vídeo2** y **System**.

También decir que el caso de uso Opciones se relaciona de igual manera mediante relaciones *extend* con los casos de **Vídeo**, **Entrada**, **Rendimiento** y **Menú**.

En este momento no se considera el volver a describir los casos de uso, pues ya se encuentran definidos de forma rigurosa en las tablas anteriores.

5.4.- MATRIZ DE RASTREABILIDAD

La matriz de rastreabilidad se usa para poder identificar de forma rápida y concisa todos los requisitos y casos de uso asociados al proyecto y cuáles dependen unos de otros.

En la siguiente tabla se puede observar de forma detallada lo explicado.

	OBJ-01	**OBJ-02**	**OBJ-03**
IRQ-01	•	•	

IRQ-02	•		•
IRQ-03	•		•
IRQ-01_01	•	•	
IRQ-01_02	•	•	
UC-01	•	•	•
UC-02	•	•	
UC-03	•	•	
UC-04	•	•	
UC-05	•		•
UC-06	•		•
UC-07	•		•
UC-08	•		•
UC-09	•		•
UC-10	•		•
UC-11	•	•	

UC-12	•	•	
UC-13	•	•	
UC-14	•	•	
CRQ_01		•	
CRQ_02	•	•	
NFR-01	•	•	•
NFR-02	•	•	•
NFR-03	•	•	•

5.5.- RESUMEN

Este resumen es acerca de todas las tablas que se ha ido viendo a lo largo del apartado de requisitos del sistema.

A continuación se muestra la tabla con el resumen de todo ello.

TIPO	**ID**	Descripción
OBJETIVOS	OBJ–01	*El sistema Sony Playstation Portable deberá ejecutar el emulador de la videoconsola Master System 2.*
	OBJ–02	*El emulador de la videoconsola Master System 2 deberá ejecutar las ROMS.*
	OBJ–03	*Se podrá configurar el emulador para características de vídeo, sonido, controles, etc.*

REQUISITOS INFORMACION	IRQ–01	El sistema deberá almacenar la información correspondiente a las ROMS, así como la carpeta en la que se encuentran.
	IRQ–02	El sistema deberá mostrar y almacenar la información correspondiente a las ROMS, así como la carpeta en la que se encuentran.
	IRQ–03	El sistema deberá mostrar y almacenar la información correspondiente a la configuración existente del emulador.
	IRQ–01_01	El sistema deberá mostrar y almacenar la información correspondiente a la partida guardada de una ROM.
	IRQ–01_02	El sistema deberá mostrar y almacenar la información correspondiente a una ROM jugada en ese momento concreto.
REQUISITOS FUNCIONALES	UC–1_01	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario ejecute la aplicación del emulador.
	UC–1_02	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario cargue una ROM en el emulador.
	UC–1_03	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario cargue una ROM en el emulador.
	UC–1_04	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario elige opciones en el menú de la interfaz del emulador.
	UC–1_05	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica las opciones de vídeo del emulador.
	UC–1_06	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica las opciones de entrada del emulador.
	UC–1_07	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica las opciones de rendimiento del emulador respecto a la Playstation Portable.

	UC–1_08	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica las opciones de manejar el menú del emulador teniendo en cuenta si la Playstation Portable es japonesa o del resto del mundo, es decir, los botones X y O.
	UC–1_09	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica los controles cuando realiza una partida a una ROM.
	UC–1_10	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica los parámetros de una ROM.
	UC–1_11	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica los parámetros de audio una ROM.
	UC–1_12	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica los parámetros de vídeo una ROM, en este caso una barra vertical en la zona izquierda de la pantalla.
	UC–1_13	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un usuario modifica los parámetros de sistema una ROM, pudiendo resetear la ROM o hacer una captura de pantalla de la ROM.
REQUISITOS NO FUNCIONALES	NFR–01	La Memory Stick será de mínimo 32 Mb para poder guardar el emulador en ella y ser ejecutado correctamente.
	NFR–02	El emulador deberá ser ejecutado lo más rápido posible para permitir un uso eficiente de él.
	NFR–03	El emulador deberá ejecutar las ROMS lo más rápido posible y que el usuario no espere más de lo necesario.
CONFLICTOS		

5.6.- GLOSARIO DE TÉRMINOS

El glosario de términos se realiza debido a que existen una serie de términos que deben ser descritos para entender el perfecto funcionamiento del proyecto, presentándose también forma de tabla. Aunque exista redundancia y se hayan visto en el apartado de **nomenclatura de términos**, se ha considerado que debido a que en el DRS deben ser especificados, deban ponerse de nuevo.

Nombre	**Descripción**
Master System 2	Videoconsola de tercera generación, la cual es de 8 bits, de sobremesa y es producida por SEGA.
Memory Stick	Tarjeta de memoria flash cuyo funcionamiento permite guardar datos, en este caso de partidas de juegos de la consola Playstation Portable, y emuladores y los propios juegos de los emuladores, además de fotos, vídeos, música, etcétera para ser ejecutados dentro de la Playstation Portable.
Playstation Portable	Videoconsola de séptima generación, la cual es de 32 bits, portátil y es producida por SONY.
ROM	Representa la memoria en la que está almacenado un juego que podrá ser ejecutado por la Master System 2.
iROM	Juego comenzado que se caracteriza por ser una ROM más el estado en ejecución de una partida jugada anteriormente y que ha sido guardada.

5.7.- ÍNDICE DE TABLAS

Por último, se incluye un índice de todas las tablas que se han ido detallando a lo largo de la documentación del presente proyecto.

6.- ANÁLISIS DEL SISTEMA

El sistema simulará el comportamiento de la videoconsola *Master System 2* emulada mediante software, así como se simulará la capa de virtualización de una máquina virtual mediante el contenedor que nos permitirá manejar de forma satisfactoria la aplicación. Indicar que no se va a requerir una base de datos como tal para el desarrollo del producto, si bien es necesario definir una serie de tipos abstractos de datos (TAD), sobre los que descansan la implementación del sistema y que soportan la gestión de ficheros, de imágenes, de ejecución de tareas en segundo plano, de partidas, de sonido, etc.; y que se detallan en el correspondiente apartado de la documentación técnica, relacionado con el diseño del sistema.

6.1.- DIAGRAMA DE PAQUETES

Primero se va a definir qué son los paquetes, para a continuación explicar el por qué se ha utilizado este tipo de diagramas para la realización del proyecto.

En UML, el paquete es un mecanismo de propósito general para organizar elementos de modelado de grupos.

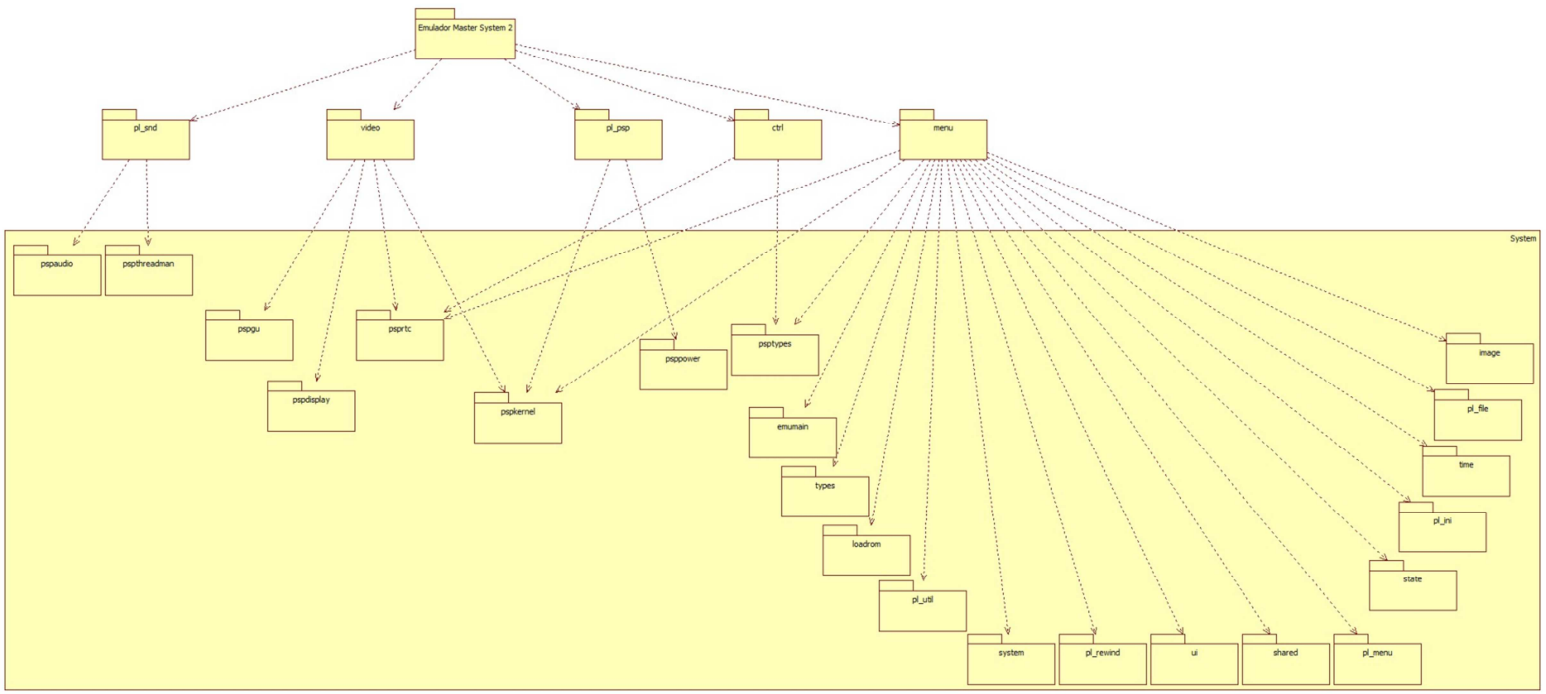
Los paquetes se utilizan para organizar los elementos de modelado en partes mayores que se pueden manipular como un grupo. La visibilidad de estos elementos puede controlarse para que algunos sean visibles fuera del paquete mientras que otros permanecen ocultos. Los paquetes también se pueden emplear para presentar diferentes vistas de la arquitectura del sistema [**EL LENGUAJE UNIFICADO DE MODELADO –BOOCH, RUMBAUGH, JACOBSON**].

Debido a esta definición se muestra claramente el por qué es necesario modelar el sistema mediante un Diagrama de Paquetes.

Dicho esto, se puede observar por todo lo explicado que existen una serie de paquetes que son los que se ven en la siguiente página en la figura del Diagrama de Paquetes.

La arquitectura general del emulador está compuesto de 28 paquetes que son los necesarios para comprender la implementación del modelo arquitectónico y que se ha seguido con una serie de funcionalidades, las cuales se detallarán de una forma específica más adelante, ya que el presente Proyecto Fin de Carrera no sigue la guía de estilo que proporciona la **Escuela de Informática de Segovia** al ser dicho proyecto muy técnico y específico.

Cada paquete corresponde a las unidades que se ven en el gráfico que se muestra a continuación. Además se hace uso de las librerías proporcionadas por el *PSPsdk*, pero que no serán detalladas al haber ya una documentación propia de ellas.


Emulador Master System 2
pl_snd
video
pl_psp
ctrl
menu
System
pspaudio
pspthreadman
pspgu
psprtc
pspdisplay
pspkernel
psppower
psptypes
emumain
types
loadrom
pl_util
system
pl_rewind
ui
shared
pl_menu
image
pl_file
time
pl_ini
state

6.1.1.-UNIDAD PL_SND

- ✓ Objetivo: funcionalidades que permiten manejar el sonido de la *Master System 2* por medio de la videoconsola *PlayStation Portable*.

- ✓ Descripción general: la emulación del chip de sonido *PSG* de la videoconsola *Master System 2* se realiza mediante la librería pl_snd.

- ✓ Descripción técnica: los atributos y componentes del paquete pl_snd de los que se harán uso en la implementación se encuentran en el archivo pl_snd.h, el cual es una librería que contiene dichos métodos y atributos pero al no tener eventos ni manejadores, no es necesario definirlo mediante una tabla y con dar una relación de las operaciones se considera totalmente definidas la unidad.

Nombre de la función
pl_snd_stereo_sample
pl_snd_mono_sample
pl_snd_sample

6.1.2.-UNIDAD VIDEO

- ✓ Objetivo: funcionalidades que permiten manejar el vídeo de la *Master System 2* por medio de la videoconsola *PlayStation Portable*.

- ✓ Descripción general: la emulación del chip de video *VDP* de la videoconsola *Master System 2* se realiza mediante la librería pl_snd.

- ✓ Descripción técnica: los atributos y componentes del paquete video de los que se harán uso en la implementación se encuentran en el archivo video.h, el cual es una librería que contiene dichos métodos y atributos pero al no tener eventos ni manejadores, no es necesario definirlo mediante una tabla y con dar una relación de las operaciones se considera totalmente definida la unidad.

Nombre de la función
pspVideoPrint
pspVideoPrintCenter
pspVideoPrintN
pspVideoPrintClipped
pspVideoPrintNRaw
pspVideoPrintRaw
pspVideoPutImage
pspVideoPutImageAlpha

6.1.3.- UNIDAD PL_PSP

- ✓ Objetivo: funcionalidades que permiten manejar la aplicación que contendrá a la *Master System 2* por medio de la videoconsola *PlayStation Portable*.

- ✓ Descripción general: la virtualización del contenedor que contendrá a *Master System 2* se realiza mediante la librería pl_psp.

- ✓ Descripción técnica: los atributos y componentes del paquete pl_psp de los que se harán uso en la implementación se

encuentran en el archivo pl_psp.h, el cual es una librería que contiene dichos métodos y atributos pero al no tener eventos ni manejadores, no es necesario definirlo mediante una tabla y con dar una relación de las operaciones se considera totalmente definidas la unidad.

Nombre de la función
pl_psp_register_callback

6.1.4.- UNIDAD CTRL

- ✓ Objetivo: funcionalidades que permiten manejar los controles con los que emular los control pad y poder manejar los juegos de la *Master System 2* por medio de la videoconsola *PlayStation Portable*.

- ✓ Descripción general: la virtualización del contenedor que contendrá a *Master System 2* se realiza mediante la librería ctrl.

- ✓ Descripción técnica: los atributos y componentes del paquete ctrl de los que se harán uso en la implementación se encuentran en el archivo ctrl.h, el cual es una librería que contiene dichos métodos y atributos que se muestran a continuación en la tabla expuesta. Al incluirse la librería pspctrl, la documentación se encuentra en ésta, no creyéndose necesario más para ser explicada esta unidad.

6.1.5.- UNIDAD MENU

- ✓ Objetivo: funcionalidades que permiten la creación de un menú para el contenedor que contendrá a la *Master System 2* por medio de la videoconsola *PlayStation Portable*.

- ✓ Descripción general: contendrá las diversas opciones con las que el usuario podrá jugar a la *Master System 2* y que se realiza mediante la librería menu.

- ✓ Descripción técnica: los atributos y componentes del paquete menú de los que se harán uso en la implementación se encuentran en el archivo menu.h, el cual es una librería que contiene dichos métodos y atributos que se muestran a continuación en la tabla expuesta. Al ser un menú simple, no contendrá componentes ni eventos ni manejadores como el resto de las unidades.

6.1.6.- UNIDAD EMUMAIN

- ✓ Objetivo: funcionalidades que permiten la ejecución de la videoconsola *Master System 2* por medio de la videoconsola *PlayStation Portable*.

- ✓ Descripción general: permite que se lance mediante la capa de virtualización el emulador de *Master System 2*, además de ejecutarlo y poder eliminarlo de memoria y que se realiza mediante la librería emumain.

- ✓ Descripción técnica: los atributos y componentes del paquete emumain de los que se harán uso en la implementación se encuentran en el archivo emumain.h, el cual es una librería que contiene dichos métodos y atributos que se muestran a continuación en la tabla expuesta. Al ser una unidad que lanzará, ejecutará y borrará de memoria los datos, no contendrá componentes ni eventos ni manejadores como el resto de las unidades.

6.1.7.- UNIDAD PL_UTIL

- ✓ Objetivo: funcionalidades que permiten guardar y cargar partidas de juegos ejecutados en la *Master System 2* por medio de la videoconsola *PlayStation Portable*.

- ✓ Descripción general: contendrá las opciones de guardar y cargar una partida de un juego ejecutado previamente y que esté ejecutándose en ese mismo instante en la *Master System 2* y que se realiza mediante la librería pl_util.

- ✓ Descripción técnica: los atributos y componentes del paquete pl_util de los que se harán uso en la implementación se encuentran en el archivo pl_util.h, el cual es una librería que contiene dichos métodos y atributos pero al no tener eventos ni manejadores, no es necesario definirlo mediante una tabla y con dar una relación de las operaciones se considera totalmente definidas la unidad.

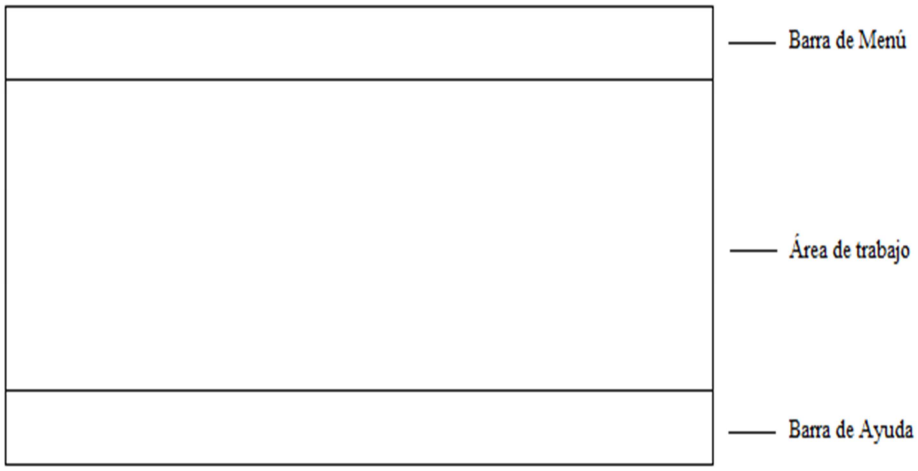
Nombre de la función
pl_util_save_image_seq

6.1.8.- UNIDAD UI

- ✓ Objetivo: funcionalidades que permiten la creación de una interfaz de usuario para el menú y poder manejar la *Master System 2* por medio de la videoconsola *PlayStation Portable*.

- ✓ Descripción general: el layout general es el que se muestra en la siguiente imagen y que será igual independientemente del lugar en el que se encuentre el usuario en el menú.

- ✓ Descripción técnica: los atributos y componentes del paquete ui de los que se harán uso en la implementación se encuentran en el archivo ui.h, el cual es una librería que contiene dichos métodos y atributos que se muestran a continuación en la tabla expuesta.


Barra de Menú
Área de trabajo
Barra de Ayuda

Componente pl_ui	Evento	Manejador
PspUiFileBrowser	OnRender	pspUiOpenBrowser
	OnOk	pspUiOpenBrowser
	OnButtonPress	pspUiOpenBrowser
PspUiMenu	OnRender	pspUiOpenMenu
	OnOk	pspUiOpenMenu
	OnCancel	pspUiOpenMenu
	OnButtonPress	pspUiOpenMenu
	OnItemChanged	pspUiOpenMenu
PspUiGallery	OnRender	pspUiOpenGallery
	OnOk	pspUiOpenGallery
	OnCancel	pspUiOpenGallery
	OnButtonPress	pspUiOpenGallery
PspUiSplash	OnRender	pspUiSplashScreen
	OnCancel	pspUiSplashScreen
	OnButtonPress	pspUiSplashScreen
	OnGetStatusBarText	pspUiSplashScreen

6.1.9.- UNIDAD PL_MENU

- ✓ Objetivo: funcionalidades que permiten la creación del menú de la aplicación.

- ✓ Descripción general: conjunto de las diversas opciones de las que dispondrá el menú para que el usuario pueda manejar de forma correcta la aplicación.

- ✓ Descripción técnica: los atributos y componentes del paquete pl_menu de los que se harán uso en la implementación se encuentran en el archivo pl_menu.h, el cual es una librería que contiene dichos métodos y atributos pero al no tener eventos ni manejadores, no es necesario definirlo mediante una tabla y con dar una relación de las operaciones se considera totalmente definidas la unidad.

Nombre de la función
pl_menu_update_option
pl_menu_remove_item
pl_menu_set_item_caption
pl_menu_set_item_help_text
pl_menu_append_option
pl_menu_find_option_by_value
pl_menu_select_option_by_index
pl_menu_select_option_by_value
pl_menu_find_option_by_index
pl_menu_create
pl_menu_destroy
pl_menu_clear_items
pl_menu_append_item
pl_menu_find_item_by_index
pl_menu_remove_item
pl_menu_get_item_count
pl_menu_clear_options
pl_menu_find_item_by_id

6.1.10.- UNIDAD PL_FILE

- ✓ Objetivo: funcionalidades que permiten el manejo de los directorios.

- ✓ Descripción general: conjunto de las funcionalidades para poder moverse de forma totalmente eficiente a través de un sistema de directorios.

- ✓ Descripción técnica: los atributos y componentes del paquete pl_file de los que se harán uso en la implementación se encuentran en el archivo pl_file.h, el cual es una librería que contiene dichos métodos y atributos pero al no tener eventos ni manejadores, no es necesario definirlo mediante una tabla y con dar una relación de las operaciones se considera totalmente definidas la unidad.

Nombre de la función
pl_file_get_file_list
pl_file_destroy_file_list
pl_file_get_file_list_count

6.1.11.- UNIDAD PL_INI

- ✓ Objetivo: funcionalidades que permiten el manejo de los ficheros.

- ✓ Descripción general: conjunto de las funcionalidades que permiten el correcto funcionamiento del sistema de ficheros.

- ✓ Descripción técnica: los atributos y componentes del paquete pl_ini de los que se harán uso en la implementación se encuentran en el archivo pl_ini.h, el cual es una librería que contiene dichos métodos y atributos pero al no tener eventos ni manejadores, no es necesario definirlo mediante una tabla y con

dar una relación de las operaciones se considera totalmente definidas la unidad.

Nombre de la función
pl_ini_create
pl_ini_destroy
pl_ini_save
pl_ini_get_int
pl_ini_set_int
pl_ini_get_string
pl_ini_set_string

6.1.12.- UNIDAD PL_REWIND

- ✓ Objetivo: funcionalidades que permiten el manejo de los estados de las *ROMS*.
- ✓ Descripción general: conjunto de las funcionalidades que permiten el correcto funcionamiento de los estados de una *ROM* guardada.
- ✓ Descripción técnica: los atributos y componentes del paquete pl_rewind de los que se harán uso en la implementación se encuentran en el archivo pl_rewind.h, el cual es una librería que contiene dichos métodos y atributos pero al no tener eventos ni manejadores, no es necesario definirlo mediante una tabla y con dar una relación de las operaciones se considera totalmente definida la unidad.
- ✓

Nombre de la función
pl_rewind_init
pl_rewind_realloc
pl_rewind_destroy
pl_rewind_reset
pl_rewind_save
pl_rewind_restore

6.1.13.- UNIDAD TYPES

- ✓ Objetivo: tipos definidos para poder manejar funcionalidades de otras unidades.
- ✓ Descripción general: conjunto de tipos manejados por la librería psptypes.
- ✓ Descripción técnica: los atributos y componentes del paquete types de los que se harán uso en la implementación se encuentran en el archivo types.h, el cual es una librería que contiene dichos atributos. Al ser una unidad que maneja librerías de la videoconsola PlayStation Portable, dichas funcionalidades se encuentran documentadas en el propio Pspsdk.

6.1.14.- UNIDAD LOADROM

- ✓ Objetivo: tipos definidos para poder manejar funcionalidades para cargar y guardar estados de una *ROM*.

- ✓ Descripción general: conjunto de tipos manejados por loadrom.c para permitir guardar el estado de una *ROM* y continuar la partida desde el mismo lugar posteriormente.

- ✓ Descripción técnica: los atributos y componentes del paquete loadrom de los que se harán uso en la implementación se encuentran en el archivo loadrom.h, el cual es una librería que contiene dichos atributos. Al ser una unidad que maneja librerías de la videoconsola PlayStation Portable, dichas funcionalidades se encuentran documentadas en el propio Pspsdk.

6.1.15.- UNIDAD STATE

- ✓ Objetivo: métodos para poder manejar el estado de una *ROM* correctamente.

- ✓ Descripción general: conjunto de métodos manejados por state.c para permitir un uso adecuado del estado de una *ROM* cuando se guarde o cargue una partida más adelante.

- ✓ Descripción técnica: los atributos y componentes del paquete state de los que se harán uso en la implementación se encuentran en el archivo state.h, el cual es una librería que contiene dichos atributos. Al ser una unidad que maneja librerías de la videoconsola PlayStation Portable, dichas funcionalidades se encuentran documentadas en el propio Pspsdk.

7.- DISEÑO DEL SISTEMA

A continuación se mostrarán en este apartado una serie de pantallas a modo de ejemplo, siendo éstas un boceto que más tarde serían utilizadas para la creación del contenedor de la aplicación. Dependiendo del lugar en el que nos encontremos en el contenedor, se podrán hacer una serie de aspectos u otros y que serán detallados uno

por uno. Más adelante, se presentarán las mismas pantallas, pero ejecutándose bajo la videoconsola *PlayStation Portable*.

7.1.- DEFINICIÓN DEL TAD

En el presente apartado se va a definir los TAD (tipo abstracto de datos) que se ha usado en la implementación para el Proyecto Fin de Carrera "Emulador de Master System 2 corriendo bajo PlayStation Portable" dela forma que se ve en la asignatura [**ESTRUCTURA DE DATOS –PILAR GRANDE GONZÁLEZ-**]

Un Tipo Abstracto de Dato (TAD), es un tipo de dato en el que sólo se definen los rasgos esenciales, sin importar los detalles específicos de implementación. Son tipos de datos no primitivos en cuyo diseño se han considerado tres principios que son:

- ✓ Separación.
- ✓ Encapsulamiento.
- ✓ Ocultación de información.

Las estructuras de datos se pueden documentar como un TAD. El documentar de un TAD se realiza por medio de una especificación lógica. Además hay que considerar dos aspectos en la realización del TAD.

- ✓ Externo: interfaz donde se especifican el nombre, las propiedades del tipo y las operaciones que se pueden aplicar.
- ✓ Interno: implementación que comprende los algoritmos mediantes los que se realizan las operaciones.

PERTENENCIA DE UNIDAD	NOMBRE DEL TIPO	ATRIBUTOS DEL TIPO	OPERACIONES
pl_file	pl_file_t	name : char[] attrs : char[] next : pl_file_t	pl_file_get_parent_directory
	pl_file_list_t	files : pl_file_t	pl_file_get_filename
			pl_file_get_extension
			pl_file_get_file_size
			pl_file_exists
			pl_file_rm
			pl_file_open_directory
			pl_file_is_directory
			pl_file_is_root_directory
			pl_file_is_of_type
			pl_file_mkdir_recursive
			pl_file_get_file_list
			pl_file_destroy_file_list
			pl_file_get_file_list_count
Descripción del TAD	Estructura de datos y operaciones que permite el manejo de ficheros en PlayStation Portable.		

OPERACIONES	DEFINICIÓN
pl_file_get_parent_directory	Obtener el directorio padre en el sistema de ficheros creado.
pl_file_get_filename	Obtener el nombre del fichero.
pl_file_get_extension	Obtener la extensión del fichero.
pl_file_get_file_size	Obtener el tamaño del fichero.
pl_file_exists	Saber si el fichero existe.
pl_file_rm	Eliminar un fichero existente.
pl_file_open_directory	Abrir un directorio.
pl_file_is_directory	Saber si es un directorio.
pl_file_is_root_directory	Saber si el directorio es el raíz.
pl_file_is_of_type	Saber el tipo del fichero.
pl_file_mkdir_recursive	Crear un directorio de forma recursiva.
pl_file_get_file_list	Obtener la lista de los ficheros.
pl_file_destroy_file_list	Eliminar la lista de los ficheros.
pl_file_get_file_list_count	Contador de los ficheros de la lista.

PERTENENCIA DE UNIDAD	NOMBRE DEL TIPO	ATRIBUTOS DEL TIPO	OPERACIONES
pl_ini	pl_ini_section_t		pl_ini_create
	pl_ini_file_t	head : pl_ini_section_t	pl_ini_destroy
			pl_ini_load
			pl_ini_save
			pl_ini_get_int
			pl_ini_set_int
			pl_ini_get_string
			pl_ini_set_string
Descripción del TAD	Estructura de datos y operaciones que permite el manejo de ficheros respecto a la creación y destrucción de ellos.		

OPERACIONES	DEFINICIÓN
pl_ini_create	Crear una partida para que sea guardada.
pl_ini_destroy	Eliminar una partida guardada.
pl_ini_load	Cargar una partida.
pl_ini_save	Guardar una partida.
pl_ini_get_int	Obtener una partida guardada.
pl_ini_set_int	Establecer una partida guardada.
pl_ini_get_string	Obtener la cadena de la partida guardada.
pl_ini_set_string	Establecer la cadena de la partida guardada.

PERTENENCIA DE UNIDAD	NOMBRE DEL TIPO	ATRIBUTOS DEL TIPO	OPERACIONES
pl_menu	pl_menu_option_def_t	text : char[] value: void*	pl_menu_create
	pl_menu_def_t	id : int caption : char[] help_text: char[] options: pl_menu_option_def	pl_menu_destroy
			pl_menu_clear_items
			pl_menu_append_item
			pl_menu_find_item_by_index
			pl_menu_find_item_by_id
			pl_menu_remove_item
			pl_menu_get_item_count
			pl_menu_set_item_help_text
			pl_menu_clear_options
			pl_menu_append_option
			pl_menu_find_option_by_index
			pl_menu_find_option_by_value
			pl_menu_select_option_by_index

			pl_menu_select_option_by_value
			pl_menu_update_option
Descripción del TAD	Estructura de datos y operaciones que permite el manejo de imágenes en PlayStation Portable.		

OPERACIONES	**DEFINICIÓN**
pl_menu_create	Crear el menú.
pl_menu_destroy	Eliminar el menú.
pl_menu_clear_items	Inicializar ítems del menú.
pl_menu_append_item	Añadir un ítem del menú al final.
pl_menu_find_item_by_index	Encontrar un ítem del menú por el índice.
pl_menu_find_item_by_id	Encontrar un ítem del menú por el identificador.
pl_menu_remove_item	Eliminar un ítem del menú.
pl_menu_get_item_count	Obtener el número de ítems del menú.
pl_menu_set_item_help_text	Establecer el texto de ayuda de un ítem del menú.
pl_menu_clear_options	Inicializar opciones del menú.
pl_menu_append_option	Añadir una opción del menú al final.
pl_menu_find_option_by_index	Encontrar una opción del menú por el índice.
pl_menu_find_option_by_value	Encontrar una opción del menú por el identificador.
pl_menu_select_option_by_index	Seleccionar una opción por su índice.
pl_menu_select_option_by_value	Seleccionar una opción dependiendo de su valor.
pl_menu_update_option	Actualizar una opción.

PERTENENCIA DE UNIDAD	NOMBRE DEL TIPO	ATRIBUTOS DEL TIPO	OPERACIONES
pl_psp	pl_callback_type	PSP_EXIT_CALLBACK	pl_psp_init
	ExitPSP	int	pl_psp_shutdown
			pl_psp_set_clock_freq
			pl_psp_get_app_directory
			pl_psp_register_callback
			pl_psp_start_callback_thread
Descripción del TAD	Estructura de datos y operaciones que permite la llamada en segundo plano de la videoconsola Master System 2 respecto a los de la videoconsola PlayStation Portable que emulará.		

OPERACIONES	DEFINICIÓN
pl_psp_init	Inicializar una llamada en segundo plano del emulador.
pl_psp_shutdown	Apagar el emulador.
pl_psp_set_clock_freq	Establecer la frecuencia de reloj de la PlayStation Portable.
pl_psp_get_app_directory	Obtener el directorio de la aplicación.
pl_psp_register_callback	Registrar la llamada en segundo plano del emulador.
pl_psp_start_callback_thread	Comenzar el hilo de ejecución del emulador de Master System 2 mediante una llamada en segundo plano.

PERTENENCIA DE UNIDAD	NOMBRE DEL TIPO	(ATRIBUTOS DEL TIPO	OPERACIONES
pl_rewind	pl_rewind	state_data_size : int state_count : int start : rewind_state current : rewind_state save_state : int load_state : int get_state_size : int	pl_rewind_init
			pl_rewind_realloc
			pl_rewind_destroy
			pl_rewind_reset
			pl_rewind_save
			pl_rewind_restore
Descripción del TAD	Estructura de datos y operaciones que permite el manejo de retroceso de una partida de una ROM.		

OPERACIONES	DEFINICIÓN
pl_rewind_init	Inicializar el retroceso.
pl_rewind_realloc	Realojar en memoria el retroceso de la partida guardada.
pl_rewind_destroy	Destruir el retroceso de una partida guardada anteriormente.
pl_rewind_reset	Resetear el retroceso de una partida guardada anteriormente.
pl_rewind_save	Guardar el retroceso de una partida.
pl_rewind_restore	Restaurar el retroceso de una partida guardada.

PERTENENCIA DE UNIDAD	NOMBRE DEL TIPO	ATRIBUTOS DEL TIPO	OPERACIONES
pl_snd	pl_snd_stereo_sample	l : short r : short	pl_snd_callback
	pl_snd_mono_sample	ch : short	pl_snd_init
	pl_snd_sample	stereo : pl_snd_stereo_sample mono : pl_snd_stereo_sample	pl_snd_set_callback
			pl_snd_pause
			pl_snd_resume
			pl_snd_shutdown
Descripción del TAD	Estructura de datos y operaciones que permite el manejo del sonido de la videoconsola Master System 2.		

OPERACIONES	DEFINICIÓN
pl_snd_callback	Llamada en segundo plano del sonido.
pl_snd_init	Iniciar el sonido.
pl_snd_set_callback	Establecer la llamada en segundo plano del sonido.
pl_snd_pause	Parar el sonido momentáneamente.
pl_snd_resume	Restaurar el sonido que se ha parado anteriormente.
pl_snd_shutdown	Apagar el sonido.

PERTENENCIA DE UNIDAD	NOMBRE DEL TIPO	ATRIBUTOS DEL TIPO	OPERACIONES
ui	PspUiMetric	Background : PspImage Font : PspFont CancelButton OkButton Left : int Top : int Right : int Bottom : int ScrollbarColor ScrollbarBgColor ScrollbarWidth : int TextColor SelectedColor SelectedBgColor StatusBarColor MenuFps : int DialogFogColor BrowserFileColor BrowserDirectoryColor BrowserScreenshotDelay BrowserScreenshotPath : char[] GalleryIconsPerRow : int GalleryIconMarginWidth : int MenuItemMargin : int MenuSelOptionBg	pspUiGetButtonIcon

		MenuOptionBoxColor MenuOptionBoxBg MenuDecorColor TitlePadding : int TitleColor TabBgColor Animate : int	
	PspUiFileBrowser	Filter : char[] Userdata	pspUiOpenBrowser
	PspUiMenu	menu : pl_menu	pspUiOpenGallery
	PspUiGallery	Userdata menu : pl_menu	pspUiOpenMenu
	PspUiSplash		pspUiSplashScreen
	UiMetric	PspUiMetric	pspUiAdhocHost
			pspUiAdhocJoin
			pspUiConfirm
			pspUiYesNoCancel
			pspUiAlert
			pspUiFlashMessage
			pl_menu_item
			pspUiFadeout
Descripción del TAD	Estructura de datos y operaciones que permite el manejo de la interfaz de usuario.		

OPERACIONES	DEFINICIÓN
pspUiGetButtonIcon	Obtener los botones creados como iconos para la interfaz de usuario.
pspUiOpenBrowser	Abrir el navegador de ficheros de la interfaz de usuario.
pspUiOpenGallery	Abrir la galería de imágenes de la interfaz de usuario.
pspUiOpenMenu	Abrir el menú de la interfaz de usuario.
pspUiSplashScreen	Crear imagen de una ROM en la interfaz de usuario.
pspUiAdhocHost	Crear una partida en red en la interfaz de usuario.
pspUiAdhocJoin	Unirse a una partida en red por medio de la interfaz de usuario.
pspUiConfirm	Botón de confirmar de la interfaz de usuario.
pspUiYesNoCancel	Botón de cancelar de la interfaz de usuario.
pspUiAlert	Mensaje de alerta de la interfaz de usuario.
pspUiFlashMessage	Mensaje rápido de la interfaz de usuario.
pl_menu_item	Crear ítem del menú en la interfaz de usuario.
pspUiFadeout	Eliminar la interfaz de usuario.

PERTENENCIA DE UNIDAD	NOMBRE DEL TIPO	ATRIBUTOS DEL TIPO	OPERACIONES
video	PspVertex	color : int x : short y : short z : short	pspVideoInit
			pspVideoShutdown
			pspVideoClearScreen
			pspVideoWaitVSync
			pspVideoSwapBuffers
			pspVideoBegin
			pspVideoEnd
			pspVideoPrint
			pspVideoPrintCenter
			pspVideoPrintN
			pspVideoPrintClipped
			pspVideoPrintNRaw
			pspVideoPrintRaw
			pspVideoPutImage
			pspVideoPutImageAlpha
			pspVideoGlowRect
			pspVideoShadowRect
			pspVideoGetVramBufferCopy
			pspVideoBeginList
			pspVideoCallList

			pspVideoAllocateVramChunk
			pspVideoGetVSyncFreq
Descripción del TAD	Estructura de datos y operaciones que permite el manejo del vídeo.		

OPERACIONES	DEFINICIÓN
pspVideoInit	Inicializar vídeo.
pspVideoShutdown	Apagar vídeo.
pspVideoClearScreen	Limpiar la pantalla de vídeo.
pspVideoWaitVSync	Sincronización vertical del vídeo.
pspVideoSwapBuffers	Buffers de intercambio para el vídeo.
pspVideoBegin	Comenzar el vídeo.
pspVideoEnd	Finalizar el vídeo.
pspVideoPrint	Imprimir el video.
pspVideoPrintCenter	Imprimir de forma centrada el vídeo.
pspVideoPrintN	Imprimir N veces el vídeo.
pspVideoPrintClipped	Imprimir cortado el vídeo.
pspVideoPrintNRaw	Impirmir N filas el vídeo.
pspVideoPrintRaw	Imprimir una fila del vídeo.
pspVideoPutImage	Poner una imagen en el video.
pspVideoPutImageAlpha	Poner una imagen alpha en el vídeo.
pspVideoGlowRect	Establecer un brillo en el vídeo.
pspVideoShadowRect	Establecer una sombra en el vídeo.
pspVideoGetVramBufferCopy	Obtener una copia del buffer de la memoria de vídeo.
pspVideoBeginList	Comenzar la lista de vídeo.
pspVideoCallList	Llamar a la lista de vídeo.
pspVideoAllocateVramChunk	Alojar en la memoria de vídeo una parte.
pspVideoGetVSyncFreq	Obtener la frecuencia de vídeo mediante sincronización vertical

7.1.1- CONSIDERACIONES ESPECIALES DE TAD

En este apartado se documentan los TAD que no contienen operaciones, o tipos asociados a él, por lo que se describirán mediante sus tablas correspondientes.

PERTENENCIA DE UNIDAD	OPERACIONES	ATRIBUTOS DEL TIPO
ctrl	PspCtrlInit	Char[]
	PspCtrlGetPollingMode	Int
	PspCtrlSetPollingMode	int
	PspCtrlPollControls	
Descripción del TAD	Estructura de datos y operaciones que permite la correspondencia de los controladores de la videoconsola PlayStation Portable respecto a los de la videoconsola Master System 2 que será emulada.	

PERTENENCIA DE UNIDAD	OPERACIONES
pl_util	pl_util_save_image_seq
	pl_util_save_vram_seq
	pl_util_date_compare
	pl_util_compute_crc32_buffer
	pl_util_compute_crc32_fd
	pl_util_compute_crc32_file
Descripción del TAD	Estructura de datos y operaciones que permite el manejo de diversas utilidades del emulador.

PERTENENCIA DE UNIDAD	OPERACIONES
loadrom	load_rom
	loadzip
Descripción del TAD	Estructura de datos y operaciones que permite la carga de una ROM.

PERTENENCIA DE UNIDAD	OPERACIONES
state	system_save_state
	system_load_state
	save_state_to_mem
	get_save_state_size
	load_state_from_mem
Descripción del TAD	Estructura de datos y operaciones que permite guardar o cargar el estado de una ROM ejecutada.

PERTENENCIA DE UNIDAD	OPERACIONES
menu	InitMenu
	DisplayMenu
	TrashMenu
Descripción del TAD	Estructura de datos y operaciones que permite guardar o cargar el estado de una ROM ejecutada.

PERTENENCIA DE UNIDAD	OPERACIONES
ctrl	InitMenu
	DisplayMenu
	TrashMenu
Descripción del TAD	Estructura de datos y operaciones que permite guardar o cargar el estado de una ROM ejecutada.

PERTENENCIA DE UNIDAD	NOMBRE DEL TIPO	ATRIBUTOS DEL TIPO
emumain	EmulatorOptions	ShowFps : int ControlMode : int ClockFreq : int DisplayMode : int VSync : int UpdateFreq : int Frameskip : int VertStrip : int SoundEngine : int SoundBoost : int AutoFire : int RewindSaveRate : int RewindReplayDelay : int
	ButtonConfig	ButtonMap[MAP_BUTTONS] : int
Descripción del TAD	Estructura de datos y operaciones que permite el manejo del inicio del emulador.	

Una vez vistas estas tablas, el resto que no se detallan es debido a que pertenecen al propio sdk (software development kit) de la PlayStation Portable y que he considerado que no es necesario el explicarlo.

Indicar también que para más detalle de todo este apartado, ir directamente al apartado de **Análisis del Sistema**, en el que se verán todos los paquetes en un propio Diagrama de Paquetes, así como los objetivos, descripciones generales y descripciones técnicas de los paquetes.

7.2.- DIAGRAMA DE ESTRUCTURA

Los diagramas de estructura son representaciones gráficas de la jerarquía existente entre los módulos existentes de un programa y las estructuras de programación utilizadas para controlar la operación de esos módulos.

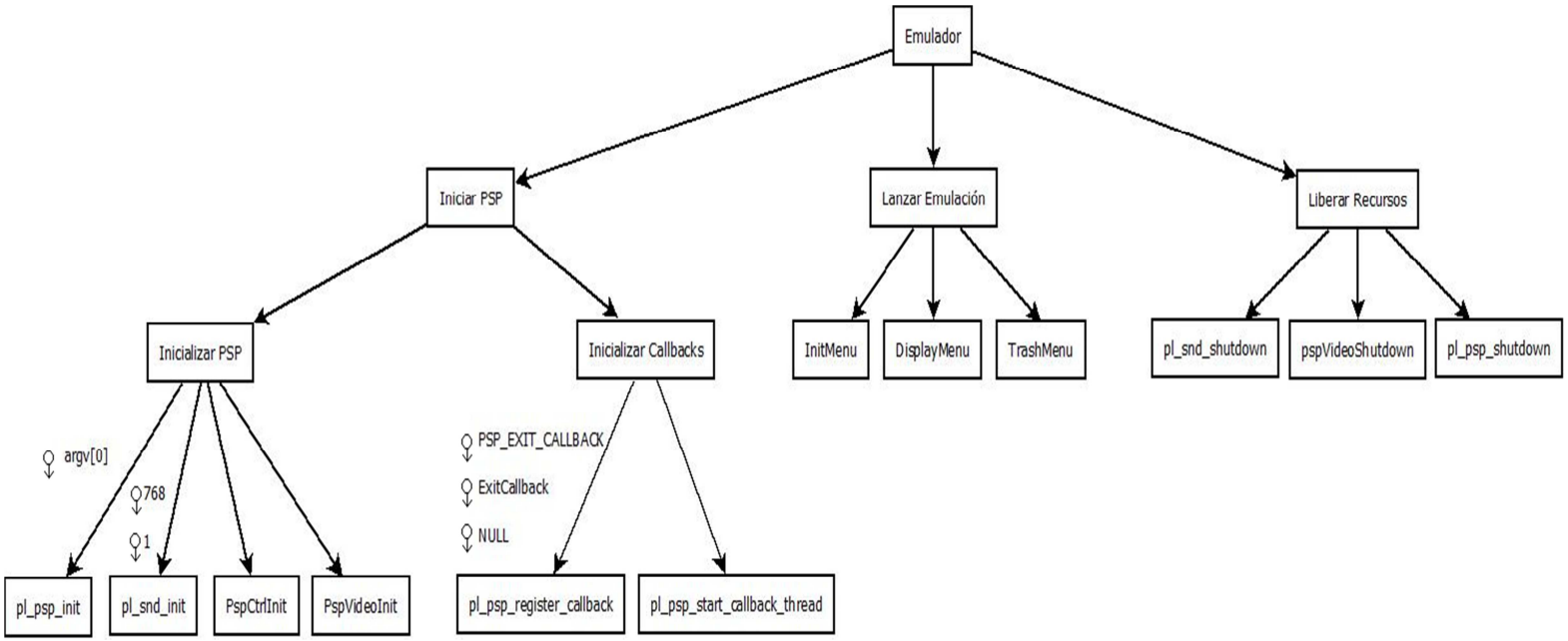
Cada módulo se representa como un rectángulo. Los módulos a su vez se pueden componer de otros, o ser módulos primitivos. Un módulo primitivo se encarga de desarrollar el procesamiento de los datos, mientras que un módulo no primitivo se encarga de administrar a otros módulos.

Para elaborar un diagrama de estructura hay que considerar los siguientes puntos:

- Descomponer el problema en una jerarquía de módulos (cada módulo debe desarrollar una única tarea).
- La ejecución de varios módulos en secuencia se indica de derecha a izquierda y de arriba abajo.
- La ejecución condicionada de varios módulos se indica mediante el rombo, señalándose de derecha a izquierda el orden en que se evalúan las condiciones.
- La iteración se indica mediante una flecha que indica los módulos a repetir.

Visto todos estos puntos, se puede deducir que aunque la programación pueda ser orientada a objetos, se puede estructurar como si fuera una programación estructurada debido a la complejidad del sistema, arquitectura y la implementación en ensamblador de algunos componentes.

En el siguiente diagrama, se puede ver de hecho, cómo se encuentra el programa principal que es conocido como main.c y que se encuentra implementado.


Emulador
Iniciar PSP
Lanzar Emulación
Liberar Recursos
Inicializar PSP
Inicializar Callbacks
InitMenu
DisplayMenu
TrashMenu
pl_snd_shutdown
pspVideoShutdown
pl_psp_shutdown
argv[0]
768
1
PSP_EXIT_CALLBACK
ExitCallback
NULL
pl_psp_init
pl_snd_init
PspCtrlInit
PspVideoInit
pl_psp_register_callback
pl_psp_start_callback_thread

Este diagrama muestra el módulo Emulador que es el main.c de nuestra aplicación. El módulo llama a tres módulos distintos en la siguiente secuencia:

- **Iniciar PSP**: módulo que sirve para iniciar la videoconsola PlayStation Portable y que cargue la aplicación en sus sistema de ficheros.

- **Lanzar Emulación**: módulo que permite la ejecución de la aplicación desarrollada para el Proyecto Fin de Carrera.

- **Liberar Recursos**: módulo que sirve para liberar de la memoria de la videoconsola *PlayStation Portable* todos los recursos que se hayan necesitado en la ejecución de la aplicación.

A su vez, el módulo Iniciar PSP está compuesto por otros cuatro módulos:

- **pl_psp_init**: módulo que es el que inicia la aplicación desde la ruta en que se encuentra guardado en la videoconsola *PlayStation Portable*. Se pasan cero argumentos a través de argv[0] para indicar la ruta.

- **pl_snd_init**: módulo que sirve para iniciar el sonido de la videoconsola *PlayStation Portable* para emular la *Master System 2*. Se pasan como argumentos 768 y 1 que corresponde a sample_count y stereo respectivamente.

- **PspCtrlInit**: módulo que permite la inicialización de los controles de la *PlayStation Portable* para poder manejar la *Master System 2*.

- **PspVideoInit**: módulo que permite la inicialización del vídeo de la *PlayStation Portable* para poder manejar la *Master System 2*.

Respecto al módulo Inicializar Callbacks, se compone de dos módulos:

- **pl_psp_register_callback**: módulo para realizar la ejecución de la aplicación en segundo plano guardándose en un registro.

- **pl_start_callback_thread**: módulo para comenzarla ejecución de la aplicación en segundo plano en un nuevo hilo de ejecución.

Por su parte, el módulo Lanzar Emulación se compone a su vez de otros tres módulos, los cuales son:

- **InitMenu**: módulo para inicializar el menú que se mostrará en la videoconsola *PlayStation Portable*.

- **DisplayMenu**: módulo para la visualización por pantalla del menú de la aplicación.

- **TrashMenu**: módulo para liberar recursos asignados al menú de la aplicación.

Por último, Liberar recursos es un módulo que contiene tres módulos, que son:

- **pl_snd_shutdown**: módulo que libera de recursos asociados a la memoria que tienen que ver con la emulación del sonido de la *Master System 2*.

- **pspVideoShutdown**: módulo que libera de recursos asociados a la memoria que tienen que ver con la emulación del vídeo de la *Master System 2*.

- **pl_psp_shutdown**: módulo que libera de recursos asociados a la memoria cuando se cierra la aplicación al terminar su ejecución.

7.4.- DISEÑO DE LA INTERFAZ DE USUARIO

INTERFAZ:

Parte muy importante de la aplicación y que se ha explicado anteriormente qué funcionalidades debe tener. De todas maneras, en este apartado se vuelve a explicar y de forma detallada las pantallas que contiene la aplicación. Hay que recordar que la interfaz es una de las partes más importantes de una aplicación, pues su buen diseño permite que una aplicación sea intuitiva, o por el contrario hace que la aplicación pueda llegar a ser un auténtico desastre y que el usuario deje de usar la aplicación. Debido a la sencillez con la que ha sido realizada y gracias a las números explicaciones que se dan en las pantallas a modo de ayuda y cambio de parámetros todos explicados, se puede decir que la interfaz se encuentra bien estructurada y facilita la navegación en la aplicación por parte del usuario, sabiendo que la máquina para la que ha sido realizada la aplicación no es una computadora y no será manejada con teclado y ratón, sino con una serie de controles más diferenciados como son cruceta, joystick analógico y una serie de botones. Por lo tanto y a modo de resumen, se puede decir que la interfaz es como sigue:

- ✓ Totalmente intuitiva.
- ✓ Presenta una gran accesibilidad a todo tipo de usuarios.
- ✓ Gran usabilidad por la facilidad con que las personas pueden utilizar la herramienta.

El producto se divide en una serie de módulos que se explicarán de una forma totalmente de tallada gracias a las pantallas que se muestran a continuación. Dichos módulos están diseñados en forma de compartimentos, los cuales son tres y que se enumeran a continuación.

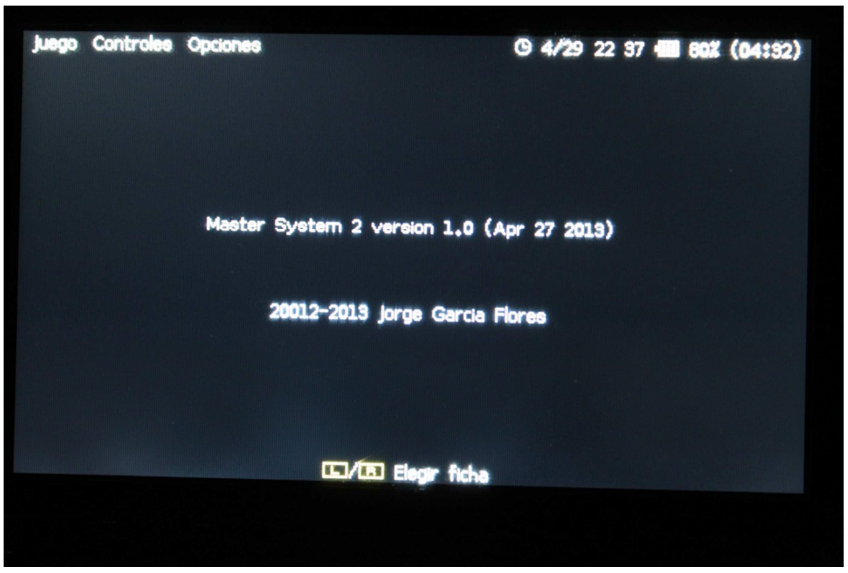
- ✓ Opciones del menú que se encuentran en la parte superior.
- ✓ Mensajes de ayuda de la aplicación encontrándose en la parte inferior.

- ✓ Bloque de navegación que nos muestra diferentes características y se encuentra entre medias de los dos anteriores.

Existen seis pantallas distintas, las cuales son la pantalla principal, la pantalla de los controles, la pantalla de las opciones, la pantalla con el sistema de directorios, la pantalla de cargar o guardar una *ROM* ejecutada previamente y la pantalla del sistema que permite las modificaciones de una *ROM* ejecutada previamente.

En este momento se pasa a describir cada una de las pantallas de la interfaz de usuario creada para la aplicación del Proyecto Fin de Carrera:

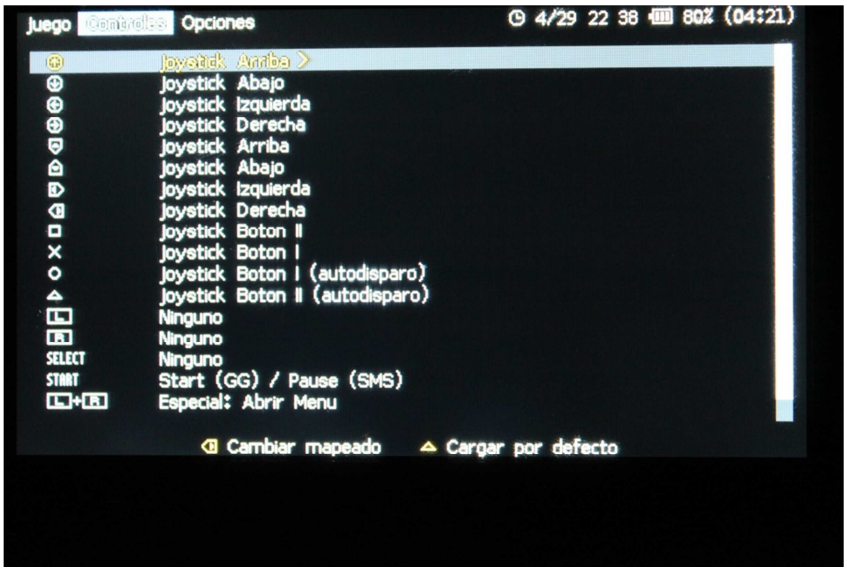
Pantalla Principal:



En este boceto se puede ver la forma principal del contenedor de la aplicación para la interfaz de usuario. Se puede comprobar cómo se encuentran divididos perfectamente los tres compartimentos. En la parte superior se encuentran las tres principales opciones del menú una vez que se ha ejecutado la aplicación. Dichas opciones son **Juego**, **Controles** y **Opciones** (esto se irá explicando en cada pantalla). También se puede ver en la esquina superior derecha una información importante para el usuario como es la fecha y hora, el porcentaje de la

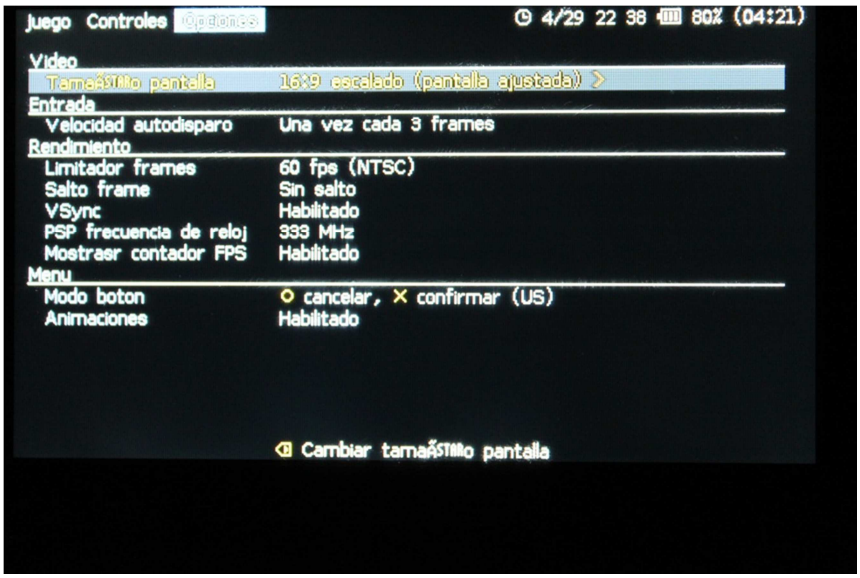
batería y las horas restantes que puede seguir usando la máquina *PlayStation Portable* sin que se conecte a la luz, es decir funcionando mediante la batería. La zona intermedia en este momento no contiene nada hasta que no se comience a navegar por el menú principal, por lo que no hay nada más que señalar en este compartimento. Por último, en la parte inferior se puede ver que es la zona de los mensajes de ayuda, y en este caso se nos indica que para navegar a través de las pestañas del menú, se usarán los botones **R** y **L** de la videoconsola *PlayStation Portable*.

Controles:



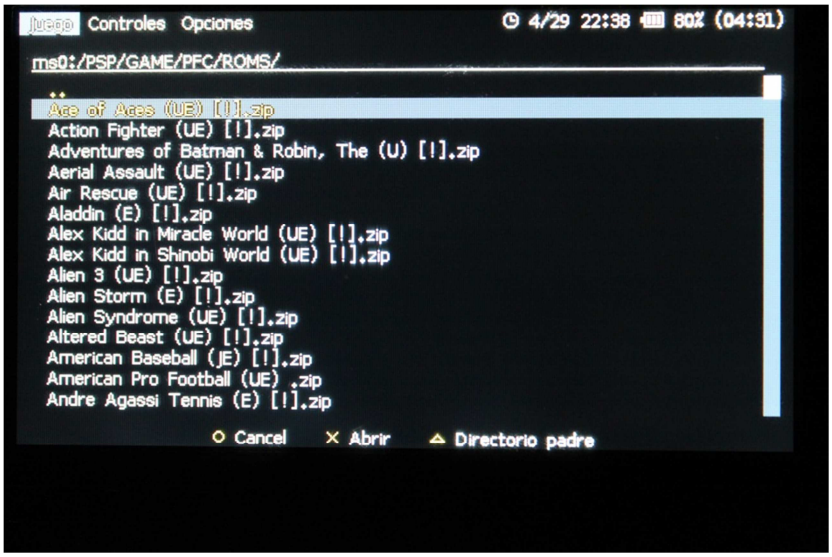
En este segundo boceto realizado, lo que se puede ver es que se sigue utilizando los mismos módulos que en la pantalla anterior, por lo que sigue la tónica general de lo que será la interfaz de usuario del contenedor. En este momento nos encontramos en el apartado de los **Controles**. El asterisco de la parte superior nos indica que se seguirán mostrando las mismas opciones que en la pantalla de menú principal (**Juego**, **Controles**, **Opciones**) y que en la parte inferior se nos indica mediante un mensaje de ayuda lo que podemos hacer. En esta pantalla, el contenedor intermedio contiene los controles que podemos utilizar para navegar por la interfaz, así como una barra de desplazamiento en la parte de la derecha para poder mostrar todo el contenido.

Opciones:



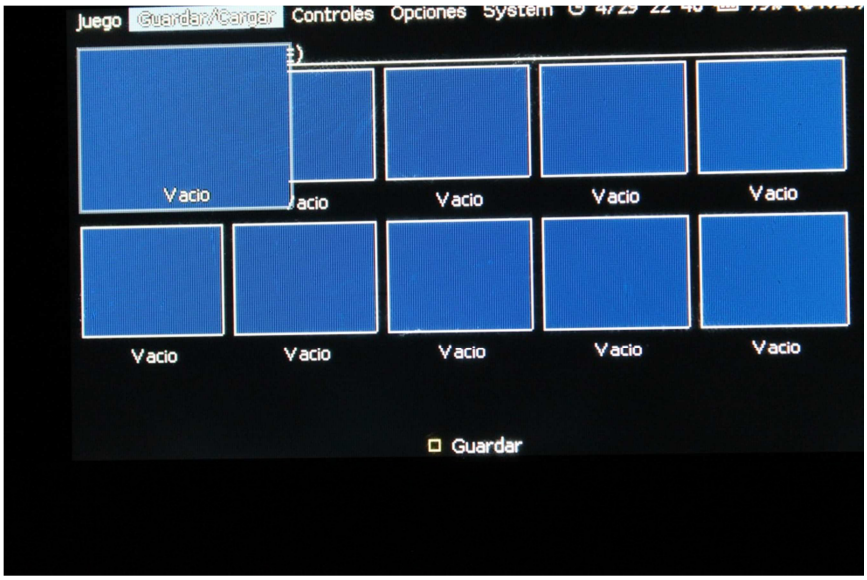
En la tercera pantalla que se muestra mediante un boceto, se ve que es el apartado de las **Opciones**. Se sigue manteniendo la misma estructura que en las anteriores pantallas, por lo que no rompe con la tónica general del diseño que se desea implementar. La parte superior sigue conteniendo lo mimo que la pantalla principal, de ahí el asterisco (**Juego**, **Controles**, **Opciones**) y en la parte inferior se nos irá mostrando los mensajes de ayuda dependiendo de dónde nos encontremos para cambiar las opciones existentes. El módulo intermedio se encuentra los parámetros que se pueden cambiar, tanto para el emulador, como para el contenedor, así como para la propia máquina física.

Juego:



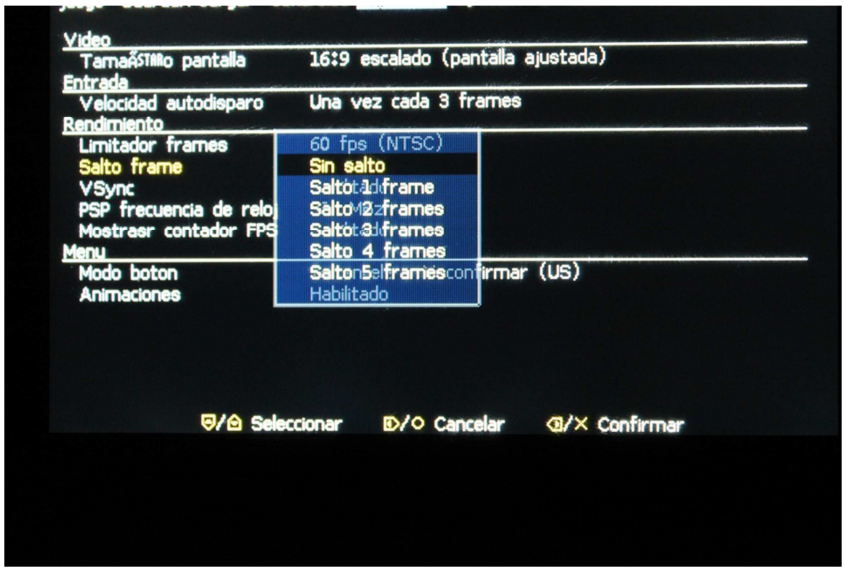
Esta cuarta pantalla nos muestra mediante el boceto lo que será la pantalla de **Juego**, refiriéndose a éste como el sistema de directorios que se ha implementado para la aplicación. Sigue manteniendo la misma estructura que el resto de las pantallas. La parte superior sigue teniendo la misma información que la pantalla principal. En el módulo o compartimento inferior se muestran los mensajes de ayuda para poder movernos en la navegación de directorios. Lo más importante de esta pantalla en la interfaz de usuario es el módulo intermedio. En él no sólo veremos las *ROMS* almacenadas en la tarjeta de memoria *Memory Stick*, sino también el directorio en el que nos encontramos actualmente, por lo que ofrece una manera fácil e intuitiva de poder moverse por la interfaz para cualquier tipo de usuario. Se muestra también una barra de desplazamiento en el caso de que el usuario contenga una gran cantidad de *ROMS* guardadas. Otra cosa importante que se puede ver es una imagen del juego en el que se encuentra el usuario seleccionando una *ROM* en el caso de que haya sido guardada anteriormente. La visualización de la imagen se realiza con transparencias para permitir visualizar de forma correcta los nombres, ya que es lo que se considera más importante a la hora de navegar por el directorio.

Salvar/Cargar:



Esta pantalla contiene una serie de consideraciones especiales que deben ser explicadas y tenidas en cuenta por parte del usuario. Lo primero que hay que indicar es que estamos en la pantalla de **Salvar/Cargar** un juego. Para ello, primero se debe haber ejecutado un juego (*ROM*) previamente y una vez hecho e ir de nuevo a la interfaz de usuario, nos aparecerán nuevas pestañas relacionadas con el juego. La primera de ellas es ésta. Aunque no rompa la estructura del contenido de la interfaz, se puede ver en la parte superior las nuevas opciones del menú en la parte superior, así como en la parte superior el mensaje de ayuda que corresponde a esta pantalla. La parte intermedia nos muestra en este momento 8 casillas distintas donde almacenar un juego guardado con su estado para ser recuperado posteriormente (al final se consiguió albergar hasta 10 partidas distintas por juego), mostrándose la casilla en la que nos encontremos con una tamaño algo mayor al resto, con una imagen del juego en el lugar en el que se encuentra guardado y la fecha y hora a la que se guardó para posteriormente reanudar su partida.

Sistema:



La última pantalla que se presenta a modo de boceto también depende de si hemos ejecutado previamente una *ROM*. En este caso nos encontramos en la opción de **Sistema**, la cual nos permite cambiar una serie de parámetros de configuración acerca de la ROM presentado una relación de dependencia con el emulador de la *Master System 2* ejecutado en ese mismo instante. Sigue la misma estructura que la pantalla de Salvar/cargar en el compartimento superior del contenedor y la misma estructura en la parte inferior que en el resto de las pantallas para el sistema de ayuda. La parte intermedia es la que nos permite cambiar los parámetros de una *ROM*, así como ciertos aspectos del emulador.

EVENTOS:

Los eventos son una serie de acciones que ocurren cuando un usuario realiza una acción. Nuestro producto contiene una serie de eventos que deben ser explicados de forma detallada para saber qué ocurre en los momentos en que un usuario realiza una acción Hay que tener en cuenta, que aunque la implementación sea una programación estructurada y no una programación orientada a eventos, sí que se debe explicar qué ocurre en el sistema.

En el momento de ejecutar la aplicación, el emulador de la videoconsola *Master System 2* no se encuentra ejecutándose todavía. Esto es así para que no consuma recursos, además solamente el emulador se ejecutará cuando un usuario cargue una *ROM* (en este caso cargar una *ROM* sería introducir un juego de forma física en la ranura de la videoconsola física). En ese momento ocurre el evento de ejecutarse el emulador de *Master System* 2, el cual se encontrará corriendo bajo la videoconsola *PlayStation Portable* como un nuevo hilo de ejecución, por lo que en este momento ocurren dos cosas totalmente diferenciadas. La primera es que la aplicación del contenedor sigue ejecutándose, y a su vez el emulador también se ejecuta, parándose cuando un jugador cargue una nueva *ROM* (ya sea la misma o distinta), comenzando otra vez el proceso de emulación.

Otro evento que ocurre es cuando un usuario se encuentra ya ejecutando una *ROM* en el emulador y desea guardar una partida o cambiar los parámetros de esa *ROM* o del propio contenedor, siguiendo la ejecución del emulador sin necesidad de “resetearlo” o “apagarlo”.

Un nuevo evento existe cuando un usuario ha guardado una partida anteriormente de una *ROM* (y que llamaremos *iROM* a modo de imagen de una *ROM*) y más adelante carga una partida desde el mismo punto, habiendo ejecutado previamente la ROM original.

Por otra parte, se considera también un nuevo evento cuando se permite la conexión mediante wi-fi por medio de ad-hoc a otra máquina *PlayStation Portable* ejecutando su propio emulador de *Master System 2* (el emulador debe ser el mismo en las dos máquinas evidentemente) para poder realizar una partida a dos jugadores teniendo en cuenta que la ROM debe ser la misma, es decir el mismo juego.

8.- CUESTIONES DE IMPLEMENTACIÓN

El objetivo en toda aplicación que se desarrolle es el de lograr proyectos homogéneos, ya sea en las tecnologías utilizadas para el

desarrollo de los proyectos como en las herramientas escogidas en la documentación generada.

Se deben utilizar herramientas de desarrollo libres, de código abierto, debido a que se evita la dependencia de empresas del sector privado, así como una reducción considerable del presupuesto en los costes de desarrollo.

Otra característica es que las herramientas que se deban utilizar deben ser portables siguiendo dos aspectos importantes para ello:

- ✓ Ser independientes del sistema operativo utilizado.
- ✓ Ser independientes de la plataforma hardware para la que se desarrollen.

La aplicación ha sido desarrollada utilizando el lenguaje *C/C++*. Dicho lenguaje es uno de los más utilizados y de los que se encuentra más documentación. A pesar de ser un lenguaje de alto nivel, se dice que es el de más bajo nivel de los existentes por su antigüedad, por lo que para el presente proyecto ha servido, al poder utilizar también lenguaje ensamblador, además de usar las típicas instrucciones propias de un lenguaje de alto nivel como son iteraciones, estructuras de control, etc.

La construcción del proyecto consta de una carpeta llamada *psplib* que contiene todos los ficheros implementados del contenedor que hace de capa de virtualización. Estos ficheros se relacionan con otros que se encuentran ya implementados que son el propio emulador de la videoconsola *Master System 2* y ficheros implementados para la propia videoconsola *PlayStation Portable* y que se encuentran en el *Pspsdk* (el kit de desarrollo de software de la PlayStation Portable).

Por lo tanto, también es necesario el uso de dicho kit de desarrollo y que se encuentra perfectamente documentado.

La interfaz de usuario es muy simple, a la vez que útil y homogénea, por lo que todas las pantallas siguen la misma estructura establecida y que se ha mostrado anteriormente en el apartado [**2.4.2 Diseño de la estructura de la aplicación**]. Gracias a esto, se ha

conseguido el mayor grado de usabilidad posible que se buscaba por la sencillez que muestra.

8.1.- JUSTIFICACIÓN DE LAS TECNOLOGÍAS UTILIZADAS

En este apartado se justifica el uso de todas las tecnologías que se han usado en el desarrollo del Proyecto Fin de Carrera "EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE".

C/C++:

El lenguaje de programación *C/C++* es un lenguaje de programación de estilo clásico que permite el uso de bucles, estructuras de control, así como estructuras de datos más complejas como arrays, registros, ficheros, etc. El lenguaje C es un lenguaje de programación estructurado, mientras que C++ es su evolución a un lenguaje de programación orientado a objetos.

Las características que tiene este lenguaje de programación son las siguientes:

- ✓ Un núcleo del lenguaje simple, con funcionalidades añadidas importantes, como funciones matemáticas y de manejo de archivos, proporcionadas por bibliotecas.
- ✓ Es un lenguaje muy flexible que permite programar con múltiples estilos. Uno de los más empleados es el estructurado "no llevado al extremo" (permitiendo ciertas licencias de ruptura).
- ✓ Un sistema de tipos que impide operaciones sin sentido.
- ✓ Usa un lenguaje de preprocesado, el preprocesador de C, para tareas como definir macros e incluir múltiples archivos de código fuente.

- ✓ Acceso a memoria de bajo nivel mediante el uso de punteros.
- ✓ Interrupciones al procesador con uniones.
- ✓ Un conjunto reducido de palabras clave.
- ✓ Por defecto, el paso de parámetros a una función se realiza por valor. El paso por referencia se consigue pasando explícitamente a las funciones las direcciones de memoria de dichos parámetros.
- ✓ Punteros a funciones y variables estáticas, que permiten una forma rudimentaria de encapsulado y polimorfismo.
- ✓ Tipos de datos agregados (struct) que permiten que datos relacionados (como un empleado, que tiene un id, un nombre y un salario) se combinen y se manipulen como un todo (en una única variable "empleado").

Pero a pesar de todas estas ventajas, también dispone de una serie de carencias o inconvenientes:

- ✓ Recolección de basura nativa, sin embargo se encuentran a tal efecto bibliotecas como la "libgc" desarrollada por Sun Microsystems, o el Recolector de basura de Boehm.
- ✓ Soporte para programación orientada a objetos, aunque la implementación original de C++ fue un preprocesador que traducía código fuente de C++ a C.
- ✓ Funciones anidadas, aunque GCC tiene esta característica como extensión.
- ✓ Soporte nativo para programación multihilo.

Pspsdk:

Kit de desarrollo de software para la videoconsola *PlayStation Portable*. Este sdk no es el oficial, por lo que se debe tener en cuenta que existe poca documentación. A pesar de ello sigue el sdk oficial proporcionado por Sony, por lo que la documentación existente por parte del oficial es válida para este sdk.

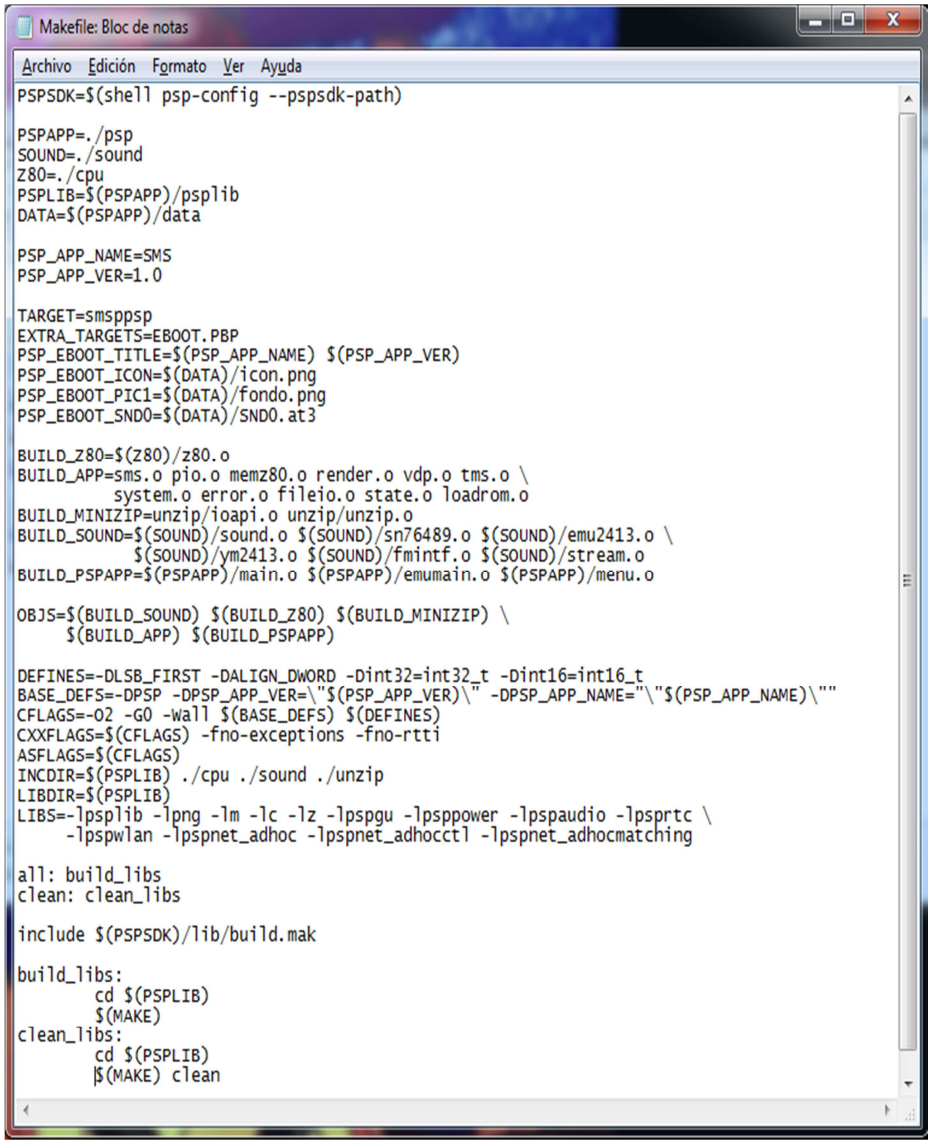
Se necesita para poder realizar el proyecto, ya que el sistema hardware no es un *PC*, sino una máquina de propósito específico como es *PlayStation Portable*.

Kit de Programación PSP 3.0:

Kit de desarrollo para poder crear ejecutables para la videoconsola *PlayStation Portable*, ya que esta videoconsola no ejecuta *.exe*, sino archivos con extensión *.PBP* y cuyos nombres de los archivos ejecutables por convenio suele ser *EBOOT*.

8.2- GENERACIÓN MAKEFILE

En este apartado se detalla la generación de un archivo makefile para la compilación de la aplicación del Proyecto Fin de Carrera "EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE". Lo primero es definir qué es un archivo makefile, el cual es un tipo de fichero para ser usado de forma rápida y eficiente en la compilación de programas cuando no se necesita un IDE (Entorno de Desarrollo Integrado). Este archivo no necesita ningún tipo de extensión, puede ser editado con el bloc de notas y para compilar el programa usando el archivo makefile, mediante la consola de comandos, se debe introducir la palabra "make".



En la imagen se encuentran una serie de líneas con sus comandos correspondientes, los cuales se explican a continuación.

La primera línea indica que en la variable PSPSDK el shell utilizado, con la configuración necesitada y dónde se encuentra la tura del sdk.

La segunda línea muestra la ruta en donde se encuentra la aplicación, lo mismo ocurre para la variable de sonido, la variable de la CPU y dónde se encuentran las librerías desarrolladas (PSPLIB). Por

último, en la variable DATA, se indica la ruta de donde tenemos los datos guardados al icono, fondo utilizado y sonido para la aplicación en el menú de la *PlayStation Portable*.

PSP_APP_NAME y **PSP_APP_VER** son variables para introducir el nombre de la aplicación y la versión.

TARGET es el nombre dado por defecto a nuestra aplicación para el compilador, mientras que **EXTRA_TARGETS** indica el nombre y extensión del fichero (**EBOOT.PBP**) que será reconocida por la videoconsola *PlayStation Portable*.

Las cuatro siguientes líneas nos dicen que son variables en donde introducimos el nombre y versión de la aplicación (**PSP_EBOOT_TITLE**), el icono que utilizamos (**PSP_EBOOT_ICON**), el fondo utilizado para la aplicación (**PSP_EBOOT_PIC1**) y el sonido que acompaña a la aplicación (**PSP_EBOOT_SND0**).

Las variables **BUILD_Z80**, **BUILD_APP**, **BUILD_MINIZIP**, **BUILD_SOUND** y **BUILD_PSPAPP** sirven para construir los .o (ficheros obj que interpreta la PlayStation Portable) de la librerías creadas.

La variable **OBJS** lo que hace es meter en ella todo lo que exista en la creación de los .o.

DEFINES, **BASE_DEFS**, **CFLAGS**, **CXXFLAGS**, **ASFLAGS**, **INCDIR**, **LIBDIR** y **LIBS** sonvariables que conviene no tocar, ya que sirven para la generación del archivo en la compilación.

La palabra all sirve para construir todas las librerías que hayamos realizado e introduciro en el makefile, mientras que la palabra clean, limpia de la caché todas las librerías que se hayan creado, compilado y formado el fichero .o. Respecto a include, con ello se pretende incluir el fichero **build.mak** del pspsdk.

Por último, **build_libs** y **clean_libs**, lo que hacen es llamar a la creación y limpieza de las librerías respectivamente.

9.- DISEÑO DE LAS PRUEBAS DEL SISTEMA

Se ha optado por utilizar el modelo de caja negra para la realización de la batería de pruebas.

Esto es así, ya que la aplicación contiene una cantidad enorme de variables, que en el caso de que fuera el modelo de caja blanca, se haría inviable la batería de pruebas, además existen otros asuntos como son los gráficos y el sonido, además de los controles usados por la aplicación (botones y cruceta en la videoconsola PlayStation Portable) que tampoco nos permite la realización del modelo de caja blanca.

Es por toda esta justificación, que se ha optado por el modelo de caja negra, el cual sigue la siguiente estandarización:

- ✓ Pruebas de instalación.

Prueba <máquina>
Características
Sistema Operativo:
Resultado de la instalación de la aplicación

- ✓ Pruebas del sistema.

Caso de prueba	
Objetivo	
Descripción	
Entrada	
Salida	
Resultado esperado	
Resultado obtenido	

9.1.- PRUEBAS DE INSTALACIÓN

Las pruebas de instalación sirven para comprobar la correcta instalación del software desarrollado en la máquina objetivo, en este caso siendo la máquina objetivo una videoconsola *PlayStation Portable*.

Prueba *PlayStation Portable*
Características: PlayStation Portable modelo PSP-1000
Sistema Operativo: Xross Media Bar (XMB)
Resultado de la instalación de la aplicación

9.2.- PRUEBAS DEL SISTEMA

Las pruebas del sistema se realizan para comprobar el perfecto funcionamiento de la aplicación desarrollada para el Proyecto Fin de Carrera "EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE".

Las pruebas consisten en el modelo de caja negra y se probarán los siguientes funcionamientos de la aplicación.

- Ejecución de la aplicación.
- Moverse por el menú.
- Comprobar batería.
- Seleccionar opción Controles.
- Seleccionar opción Opciones.
- Seleccionar opción Juego.
- Cargar *ROM*.
- Guardar partida de *ROM*.

- Cargar partida de *ROM*.
- Seleccionar opción Sistema de *ROM*.
- Visualizar fps (frames por segundo).
- Visualizar estado de la batería.
- Comprobar fecha en el sistema.
- Comprobar funcionamiento de sonido.
- Seleccionar opción Opciones.
- Cambiar parámetros de Opciones.
- Realizar captura de pantalla.
- Finalizar aplicación.

Con todas estas pruebas se pasa en el presente apartado a mostrar la batería de pruebas en el modelo de caja negra:

Caso de prueba: **Ejecución de la aplicación**	
Objetivo	Comprobar que la aplicación se ejecuta correctamente en la videoconsola PlayStation Portable.
Descripción	Al cargar la aplicación desde el XMB (Xross Media Bar) se debe ejecutar la aplicación.
Entrada	Pulsar botón X en el icono de la aplicación.
Salida	El contenedor de la aplicación.
Resultado esperado	Comienza la ejecución de la aplicación.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Moverse por el menú**	
Objetivo	Comprobar que el usuario puede moverse por las diferentes opciones del menú de la aplicación.
Descripción	El usuario con los botones de la videoconsola L y R navega por el menú de la aplicación.
Entrada	Pulsar botón L o R.
Salida	Cambiar de opción.
Resultado esperado	Se cambia de opción yendo a la derecha si se pulsa el botón R o a la izquierda pulsando el botón L.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Comprobar batería**	
Objetivo	Comprobar el tiempo de uso restante de la batería, tanto en horas como en nivel de batería.
Descripción	El usuario puede ver estando en el menú el tiempo de batería restante.
Entrada	Nivel de batería y tiempo.
Salida	Nivel de batería actual y tiempo restante.
Resultado esperado	El nivel de batería disminuye a lo largo del tiempo que se utilice la videoconsola *PlayStation Portable* sin que se conecte a la red eléctrica.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Seleccionar opción Controles**	
Objetivo	Seleccionar la opción de Controles y cambiar los parámetros.
Descripción	El usuario puede cambiar los controles con los que manejará los videojuegos a modo de *control pad* o *joystick*.
Entrada	Controles por defecto.
Salida	Nuevos controles.
Resultado esperado	Los videojuegos se podrán manejar con los controles que se han definido de nuevo.
Resultado obtenido	Se obtiene el resultado esperado.
Caso de prueba: **Seleccionar opción Juego**	
Objetivo	Seleccionar la opción de Juego y navegar por el sistema de archivos.
Descripción	El usuario puede navegar por el sistema de ficheros creado y ver las *ROMS* que se encuentran guardadas en el directorio que se encuentren en la tarjeta de memoria *Memory Stick*.
Entrada	Seleccionar la opción Juego.
Salida	Las *ROMS* existentes que se encuentren guardadas.
Resultado esperado	Se muestran las *ROMS* guardadas en el directorio en el que se encuentren.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Cargar *ROM***	
Objetivo	Seleccionar una *ROM* guardada en el sistema de archivos.
Descripción	El usuario elige una *ROM* para poder jugar a ella en el emulador de la videoconsola *Master System 2*.
Entrada	Seleccionar *ROM*.
Salida	El juego se ejecuta y el usuario puede jugar a él.
Resultado esperado	Se carga la *ROM* y el usuario comienza a jugar a dicha *ROM*.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Guardar partida de *ROM***	
Objetivo	Guardar el estado de una *ROM* para continuar desde él más adelante.
Descripción	El usuario guarda la partida de una *ROM* para continuarla más delante con dicha partida.
Entrada	Guardar partida.
Salida	Se guarda el estado de la partida con la fecha y hora.
Resultado esperado	Se guarda el estado de la *ROM*.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Cargar partida de *ROM***	
Objetivo	Cargar el estado de una *ROM* para continuar con la partida.
Descripción	El usuario carga la partida de una *ROM* que ha sido guardada previamente.
Entrada	Cargar partida.
Salida	Se carga el estado de la partida y el usuario puede continuar con la partida desde donde estuviera.
Resultado esperado	Se carga el estado de la *ROM*.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Seleccionar opción Sistema de *ROM***	
Objetivo	Realizar acciones que afectan a una *ROM*.
Descripción	El usuario selecciona la opción de sistema para cambiar parámetros de una *ROM* o hacer una captura de pantalla.
Entrada	Seleccionar opción Sistema.
Salida	Se cambian parámetros que afectan a la *ROM*.
Resultado esperado	Los parámetros de la *ROM* han sido cambiados.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Visualizar fps (frames por segundo)**	
Objetivo	Mostrar los fps a los que funciona el emulador de la *Master System 2*.
Descripción	Se muestran los fps a los que el emulador ejecuta una *ROM*.
Entrada	Los fps a los que se visualizan las *ROM*.
Salida	Los fps actuales a los que se visualizan las *ROM*.
Resultado esperado	Los fps actuales.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Visualizar estado de la batería**	
Objetivo	Mostrar el estado de la batería de la *PlayStation Portable*.
Descripción	Se muestran un gráfico de la batería, el porcentaje restante de batería y el tiempo que queda de batería antes de que se apague la videoconsola PlayStation Portable sin que ésta se encuentre conectada a la red eléctrica.
Entrada	Estado de la batería.
Salida	Gráfico, porcentaje y tiempo restante de batería.
Resultado esperado	Estado actualizado de la batería en tiempo real.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Comprobar fecha y hora en el sistema**	
Objetivo	Mostrar la fecha que se encuentra guardada en el estado del sistema *PlayStation Portable*.
Descripción	Se muestra la fecha y hora actual del sistema.
Entrada	Fecha y hora del sistema.
Salida	Fecha y hora actualizadas.
Resultado esperado	Estado actualizado de la fecha y hora del sistema.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Comprobar funcionamiento de sonido**	
Objetivo	El sonido generado por el emulador debe funcionar correctamente.
Descripción	Cada ROM tiene unas melodías propias y deben poder escucharse por el sistema de sonido de la videoconsola *PlayStation Portable*.
Entrada	Carga de *ROM*.
Salida	Sonido de la *ROM* generado por el sistema de sonido de la *PlayStation Portable*.
Resultado esperado	Se escucha el sonido de forma acorde.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Seleccionar opción Opciones**	
Objetivo	Seleccionar la opción de opciones una vez cargada una *ROM*.
Descripción	Al seleccionar la opción de Opciones de una *ROM*, se podrá cambiar parámetros que afectan al contenedor que visualiza las *ROM*.
Entrada	Seleccionar Opciones.
Salida	Todas las opciones disponibles a ser cambiadas en Opciones.
Resultado esperado	Se ha seleccionado correctamente Opciones.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Cambiar parámetros de opciones**	
Objetivo	Cambiar los parámetros disponibles que afectan al contenedor que visualiza la *ROM* en Opciones.
Descripción	Se podrán cambiar los parámetros que afectan al contenedor visualizador de la ROM como son el tamaño de la pantalla, la entrada, límite de fps, saltos de fps, sincronización vertical (VSync), frecuencia de reloj de la *PlayStation Portable* (desde 222 MHz hasta 333 MHz), mostrar el contador de fps, modo de botones americano o japonés y animaciones para los menús.
Entrada	Estado de la batería.
Salida	Gráfico, porcentaje y tiempo restante de batería.
Resultado esperado	Estado actualizado de la batería en tiempo real.
Resultado obtenido	Se obtiene el resultado esperado.

Caso de prueba: **Finalizar aplicación**	
Objetivo	Terminar la aplicación y volver al sistema operativo de la *PlayStation Portable*.
Descripción	Salir de la aplicación de forma satisfactoria.
Entrada	Pulsar el botón *home* de la videoconsola *PlayStation Portable*.
Salida	Volver al sistema operativo *XMB* (Xross Media Bar).
Resultado esperado	Se vuelve al XMB de la *PlayStation Portable*.
Resultado obtenido	Se obtiene el resultado esperado.

PARTE 3:

CONCLUSIONES

10.- CONCLUSIONES Y FUTUROS TRABAJOS

A continuación en este apartado se hablará acerca del trabajo desarrollado de forma personal por parte del Proyecto Fin de Carrera "EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE", así como la explicación de las consideraciones fundamentales de la aplicación a través de los casos de prueba anteriormente planteados.

10.1.- EVALUACIÓN

Una vez realizadas las pruebas expuestas en el capítulo anterior (**8.2.- PRUEBAS DEL SISTEMA**), se considera que se debe hacer una evaluación sobre el funcionamiento de la aplicación desarrollada para el Proyecto Fin de Carrera "EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE" dentro del sistema de la videoconsola PlayStation Portable.

Con esta aplicación se busca una serie de características que se han considerado fundamentales al ahora de desarrollar la aplicación, y teniendo en cuenta éstas, se pasan a enunciar diferentes puntos sobre las pruebas realizadas en la aplicación.

10.1.1- EVALUACIÓN DEL RENDIMIENTO

Las pruebas realizadas respecto a la evaluación del rendimiento, son las relativas a la velocidad de carga de la aplicación, así como de la carga de *ROM* y los estados de partidas salvadas para una futura carga de estado desde un punto ya establecido con anterioridad.

La carga de la aplicación es lo que va a tardar más, 8 segundos exactos, ya que al no tener la videoconsola *PlayStation Portable* un disco duro al que acceder, debe ir primero a la tarjeta de memoria *Memory Stick*, acceder al lugar donde se encuentra la aplicación y por último cargar la aplicación. A pesar de que pueda parecer un tiempo alto, en realidad es corto por las especificaciones tenidas en cuenta,

tanto del lugar donde se encuentra guardada la aplicación como que es una máquina virtual lo que se ha hecho en realidad.

Una vez que ya ha cargado la aplicación, la navegación a través del menú es rápida con una respuesta prácticamente al momento de las órdenes transmitidas por los botones de la videoconsola *PlayStation Portable*, llegando a alcanzar velocidades menores a 1 milisegundos, es decir prácticamente imperceptibles para el ojo humano, así como para el cambio de todos los parámetros existentes en la aplicación que pueden ser configurados.

Respecto a la carga de las diferentes *ROM* el tiempo requerido es de dos segundos, un tiempo muy pequeño para la carga de un programa, más siendo como es una máquina virtual lo que se ha desarrollado. Hay que incidir en que la aplicación se basa en una videoconsola *Master System 2*, la cual usaba cartuchos *ROM* como videojuegos, por lo que un usuario debía introducir un cartucho y encender la consola, algo que no es requerido en la aplicación desarrollada.

El manejo de las *ROM* a la hora de jugar es absolutamente momentáneo, respondiendo perfectamente a las órdenes transmitidas por el usuario por medio de los botones de la *PlayStation Portable*.

Por último, decir que el tiempo de respuesta a la hora de cargar y guardar el estado de cualquier *ROM* es también instantáneo, por lo que al usuario no se le hace esperar a la hora de realizar las citadas acciones.

Como consideración, aclarar que el tamaño de la aplicación es de 630 Kb, por lo que su rendimiento, respecto al tamaño de lo que ocupa es excelente debido a la optimización de los recursos que se han tenido que hacer, ya que la consola *PlayStation Portable* no tiene el mismo tamaño de memoria *RAM* que otras computadoras de propósito general o incluso de propósito específico como las videoconsolas de sobremesa, mucho más avanzadas en este aspecto.

10.1.2.- EVALUACIÓN DE LA ROBUSTEZ

El sistema es lo suficientemente robusto para no poseer caídas en los diferentes servicios que ofrece. Estos servicios son los que dependen desde un primer momento a la carga de la aplicación por parte de la videoconsola *PlayStation Portable*, no dejando al usuario esperando durante un tiempo muy amplio, que haga que su experiencia con la aplicación sea totalmente insatisfecha.

Lo mismo sucede para la carga de cualquier *ROM*, ya que en caso de que el sistema se quedara parado, el usuario no podría hacer uso de la aplicación desarrollada y teniendo que desechar el producto.

Esta explicación se puede aplicar de la misma manera cuando un usuario salva y carga el estado de una *ROM* para continuar la partida en cualquier otro momento y el sistema se muestra lo suficientemente robusto como para no tener ninguna caída, así como que el frame rate que poseen los juegos de las *ROM* continúan siendo de 30 fps, por lo que la experiencia del usuario en este caso va a ser satisfecha.

En último lugar, decir que cualquier cambio en los parámetros, tanto del contenedor, como de la visualización y sonido por parte de las *ROM*, así como acerca del rendimiento ofrecido por la videoconsola *PlayStation Portable* no hará que el sistema tenga ninguna pérdida y pueda continuar con la partida que tuviera.

10.2.- CONSECUCIÓN DE LOS OBJETIVOS PLANTEADOS

Se han conseguido los objetivos fundamentales planteados desde un principio, los cuales eran ejecutar emulador *Master System 2* en *PlayStation Portable*, ejecutar las *ROM* de la videoconsola *Master System 2* en la videoconsola *PlayStation Portable* y la configuración de los parámetros que afectan al contenedor del emulador.

Todo ello ha sido realizado de con una interfaz de usuario simple, pero a la vez de forma intuitiva y amigable para el usuario y que puede ser utilizada la aplicación desarrollada en todo momento, al

ser la videoconsola *PlayStation Portable* una videoconsola portátil, que lo único que le afecta es tener un nivel de batería aceptable para poder estar jugando un tiempo relativamente elevado.

También se ha conseguido utilizar software libre para el desarrollo de la aplicación, por lo que el coste económico se reduce drásticamente, sobre todo teniendo en cuenta para qué plataforma está desarrollado y el uso de unos kits de desarrollo proporcionados por la empresa *Sony* hubieran disparado el coste de forma exponencial siempre y cuando se hubieran podido adquirir, por lo que se decidió usar kits de desarrollo realizados por gente ajena que los distribuyen sin ánimo de lucro.

10.3.- ADQUISICIÓN DE NUEVOS CONOCIMIENTOS

Durante el desarrollo del Proyecto Fin de Carrera "EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE" se han tenido que utilizar tecnologías totalmente nuevas, con lo cual ha sido necesario el aprendizaje previo por mi parte para el empleo de las tecnologías utilizadas, además de una lectura profunda de toda la documentación encontrada y que se detalla más adelante en el apartado de **BIBLIOGRAFÍA**, y aunque se ha tenido desde un principio de forma totalmente clara las especificaciones acerca del producto a desarrollar, ha sido necesario pensar el porqué del uso de esas tecnologías y no otro.

Esto es debido a que el desarrollo de software en el mundo *hombrew* para la videoconsola *PlayStation Portable* es muy específico y no se puede usar otro tipo de software.

Al desarrollar el Proyecto Fin de Carrera "EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE", se ha conseguido la familiarización con el mundo de desarrollo de aplicaciones para videoconsolas, más específicamente el desarrollo de emuladores de máquinas antiguas que puedan ser usadas en otras más modernas, en especial en la videoconsola *PlayStation Portable*, por lo que la sensación obtenida es que puedo acceder a un

mundo laboral orientado al servicio de software lúdico en el cual el desarrollo de aplicaciones es muy complejo debido a las tecnologías y componentes que hay que usar y la integración entre ellas.

En la adquisición de los nuevos conocimientos, se ha invertido un mayor tiempo y esfuerzo en el aprendizaje de lo que se desconoce totalmente como puede ser una programación en un lenguaje de programación como es **C**, unido a lenguaje ensamblador, generación de ficheros *makefile*, los kits de desarrollo de *PlayStation Portable* y otros programas como *GoldWave* y la librería de sonido Atrac3 para la generación de música usada en la aplicación.

Como se ha dicho, el desarrollo de la aplicación ha sido muy duro y largo, debido a que se ha necesitado una lectura profunda de la documentación encontrada, por lo que no se comenzó a programar hasta bien avanzado el tiempo en el desarrollo del Proyecto Fin de Carrera. Sin embargo, todo ese tiempo que se estableció para el aprendizaje de conocimientos por medio de la lectura, me ha proporcionado una forma de programar en la que la optimización ha sido uno de los recursos fundamentales. Esto ha sido así desde un primer momento, por lo que el tamaño de aplicación podría llegar a ocupar incluso en un diskette, ya que su tamaño ocupa tan sólo 630 Kb con todas las funcionalidades que se han llegado a ofrecer de cara al usuario.

En último lugar, decir que gracias a todo lo que he llegado a aprender, puedo optar a un puesto laboral privado en la que es ahora mismo la mayor industria del ocio existente, superando al cine o la música, por lo que a medio-largo plazo se realizarán inversiones cada vez mayores, con un *feedback* de cara a los usuarios mucho más importante y con unos sistemas de financiación totalmente distintos al resto de las industrias del ocio, por lo que está en una continua expansión y que es cambiante de la noche a la mañana dependiendo de los gustos de los usuarios.

Esto se ve sobre todo a que cada vez más, existe más gente que quiere recordar o saber cómo comenzó la industria del videojuego, y aunque el emulador desarrollado sea de una videoconsola de tercera

generación como es la *Master System 2*, sí se puede empezar a comprender la evolución de la industria del videojuego y hasta dónde se ha llegado en términos de potencia de computación y que algo mucho más básico puede poseer las mismas dosis de diversión que algo mucho más moderno y más potente.

También decir, que a pesar de adquirir nuevos conocimientos, se han potenciado otros existentes como son los siguientes:

- ✓ Ingeniería del Software: usado para todo el proceso de análisis y diseño del producto a la hora de realizar el Proyecto Fin de Carrera.
- ✓ Arquitectura de un sistema: usado para comprender qué tipo de tecnologías y arquitectura se debía usar en el presente proyecto.
- ✓ Programación: usado para el proceso de implementación del proyecto obviamente.

10.4.- POSIBLES AMPLIACIONES

A continuación se exponen las posibles ampliaciones como mejoras para el Proyecto Fin de Carrera "EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE":

- ✓ Nuevos filtros de renderizado de imágenes para mejorar los gráficos (2xSAI –Kreed-).
- ✓ Creación de un contenedor para poder emular diferentes videoconsolas de la misma generación y anteriores generaciones (Nintendo NES, Game Boy,…).
- ✓ Compartir capturas de pantalla con Facebook mediante un sistema gestor de usuarios con login y password.
- ✓ Externalizar ROM a un servidor en la nube, creando un sistema cliente-servidor mediante una arquitectura de máquina virtual.

- ✓ Externalizar partidas guardadas por usuario en un servidor en la nube.

11.- BIBLIOGRAFÍA

- LIBROS:
    - Apuntes de Ingeniería Técnica de Informática de Gestión
        - PILAR GRANDE GONZÁLEZ. *Apuntes Estructura de datos*
        - FRANCISCO JOSÉ GONZÁLEZ CABRERA. *Apuntes Ingeniería del software 1*
    - Libros de consulta
        - BOOCH, RUMBAUGH, JACOBSON. *El Lenguaje Unificado de Modelado*. **Editorial Pearson 2ª edición 2005**
        - IAN SOMMERVILLE. Ingeniería del Software. **Editorial Pearson 7ª edición 2004**
        - FRANCISCO JAVIER CEBALLOS. *C/C++ Curso de programación*. **Editorial Ra-Ma 2ª edición 2002**
    - Miscelánea
        - ISRAEL LÓPEZ FERNÁNDEZ. *Trabajo Fin de Carrera Emulador de Master System 2 en C++*.

- DIRECCIONES WEB:
    - Master System 2
        - http://www.smspower.org/Development/Documents
        - http://www.smspower.org/Development/OfficialDocumentation
        - http://www.smspower.org/Development/SMSPagingChips
        - http://www.smspower.org/uploads/Development/chipsym2413am.pdf

        - http://www.smspower.org/uploads/Development/msvdp-20021112.txt
        - http://www.smspower.org/uploads/Development/smstech-20021112.txt
        - http://www.smspower.org/uploads/Development/SN76489-20030421.txt
        - http://www.smspower.org/uploads/Development/z80.pdf

    - Máquina Virtual
        - http://www.vmware.com/es/virtualization/virtualization-basics/virtual-machine.html
        - https://es.wikipedia.org/wiki/Maquina_virtual

    - Miscelánea
        - https://es.wikipedia.org/wiki/Master_System
        - https://es.wikipedia.org/wiki/PlayStation_Portable

APÉNDICES

APÉNDICE I: MANUAL DE INSTALACIÓN

1.- COMPONENTES NECESARIOS

Los componentes que necesitaremos para la instalación de la aplicación son los siguientes:

- ✓ Videoconsola *PlayStation Portable*.
- ✓ Tarjeta *Memory Stick.*
- ✓ Cable *USB.*

La videoconsola *PlayStation Portable* debe ser el modelo mínimo *PSP-1000* y contener **Custom Firmware 5.00 M33-4**, que nos permite ejecutar cualquier aplicación homebrew, es decir, no oficial. Se puede descargar desde [**http://psp.scenebeta.com/**] (hay que registrarse para ello) y para instalarlo seguir las instrucciones del tutorial que se encuentra en el sitio web [**http://psp.scenebeta.com/tutorial/downgrade-universal-de-cualquier-firmware-psp-o-psp-slim-sin-usar-otra-psp]**.

Respecto a la tarjeta de memoria debe ser una *Memory Stick* de al menos 32 MB que será en la que se guardará la aplicación y será ejecutada por la *PlayStation Portable*.

Por último, se necesitará un cable *USB* (modelo *shielded 1700 2.0*) para poder guardar en la tarjeta de memoria la aplicación desarrollada.

A continuación las imágenes de los componentes que se necesitarán.

Videoconsola PlayStation Portable

Memory Stick 32 MB

Cable USB shielded 1700 2.0

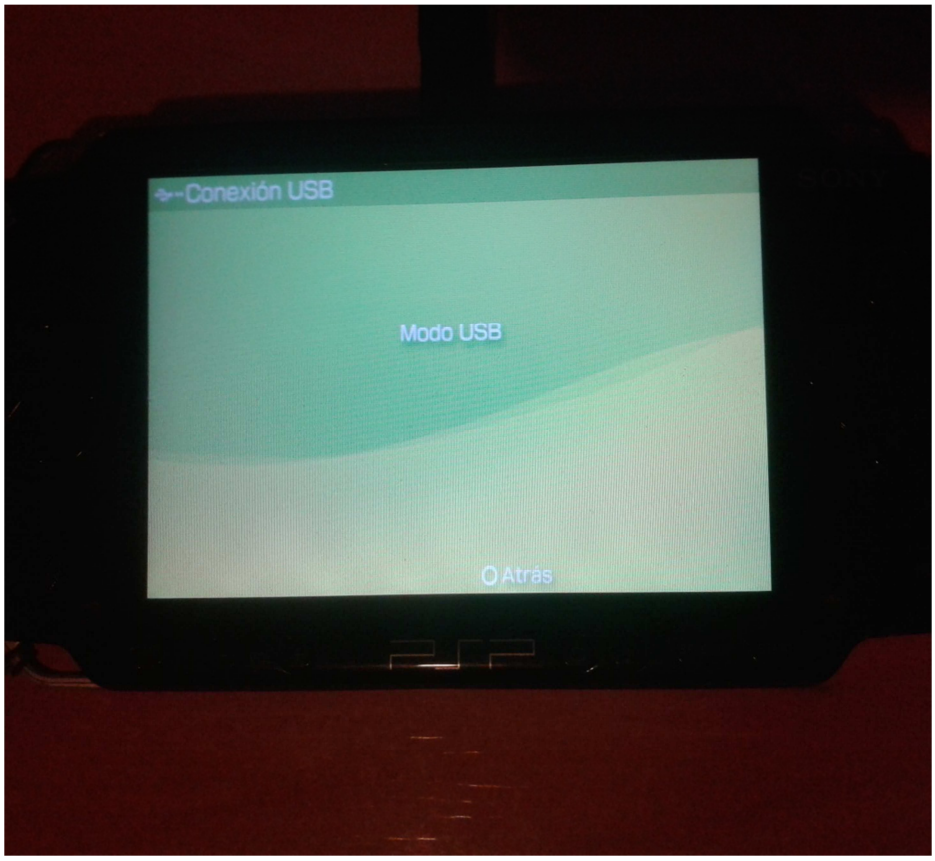
Una vez especificados los componentes necesitados, la aplicación se deberá instalar del siguiente modo, siguiendo una serie de pasos. Dichos pasos serán acompañados de unas imágenes a modo ilustrativo de cómo se debe realizar la instalación del producto.

- Lo primero que se debe hacer es introducir la tarjeta de memoria Memory Stick para que la aplicación sea guardada en ella.

- Inserte el cable USB y conecte la videoconsola PlayStation Portable a un PC y diríjase a donde tiene guardada la aplicación.

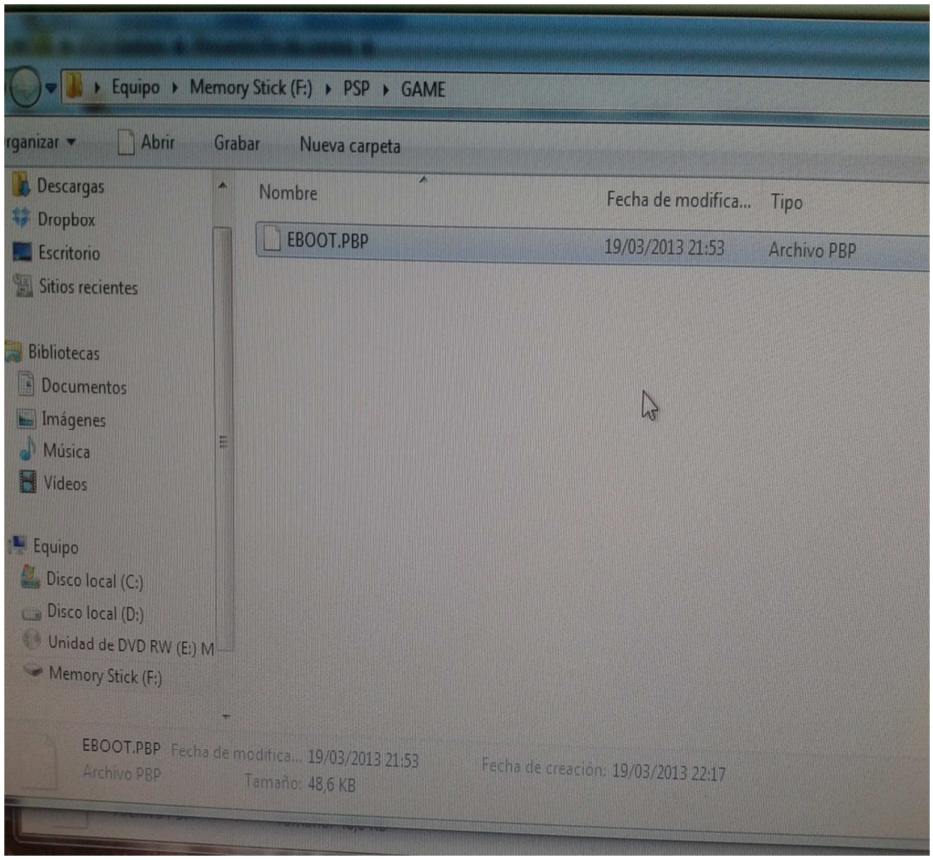
- A continuación se enciende la videoconsola PlayStation Portable y se coloca en Ajustes → Conexión USB.

- Realice la conexión de USB pulsando el botón X. Verá en la pantalla un mensaje que le indica que su videoconsola PlayStation Portable se encuentra conectada al PC.



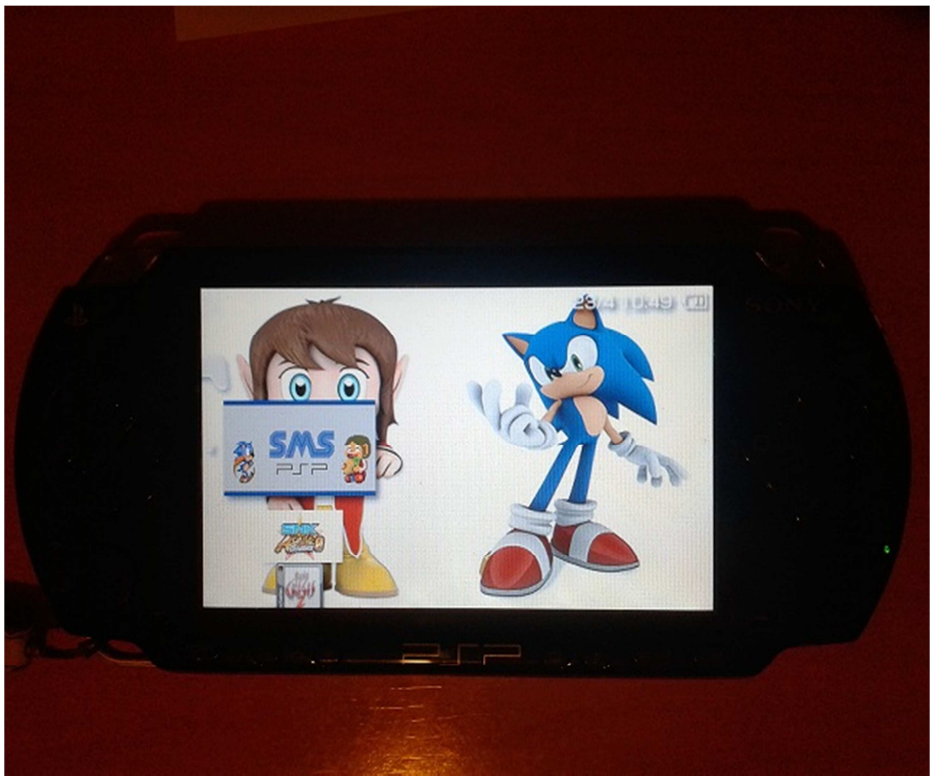
- Cree desde el PC una nueva carpeta en Memory Stick (F:/ en este caso) → PSP → GAME

- Copie y pegue el fichero EBOOT.PBP en la carpeta de destino creada.



- Pulse el botón O y vaya en el menú a Juego → Memory Stick y a continuación pulse el botón X.

Si ha seguido todos los pasos correctamente (tener en cuenta que F:/ puede ser otra unidad, dependiendo de en qué puerto USB del PC haya conectado la videoconsola PlayStation Portable), verá una imagen como la que sigue.



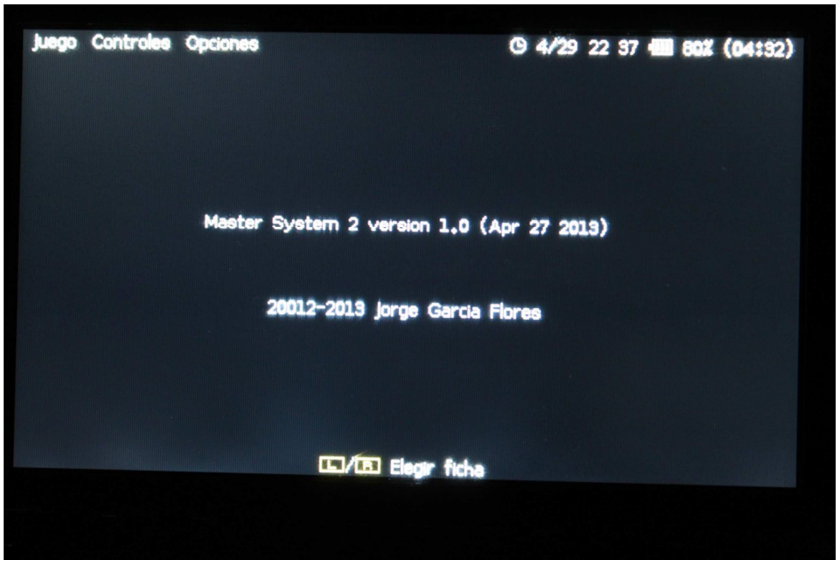
APÉNDICE II: MANUAL DE USUARIO

1.- Cómo empezar

Una vez se haya instalado la aplicación en la videoconsola PlayStation Portable (**APÉNDICE I: MANUAL DE INSTALACIÓN**), el usuario deberá ir en el menú *XMB* (*Xross Menú Bar*) al apartado Juego y a continuación dentro de él, elegir la opción *Memory Stick*^{TM} y pulsar el botón X de la videoconsola *PlayStation Portable*.

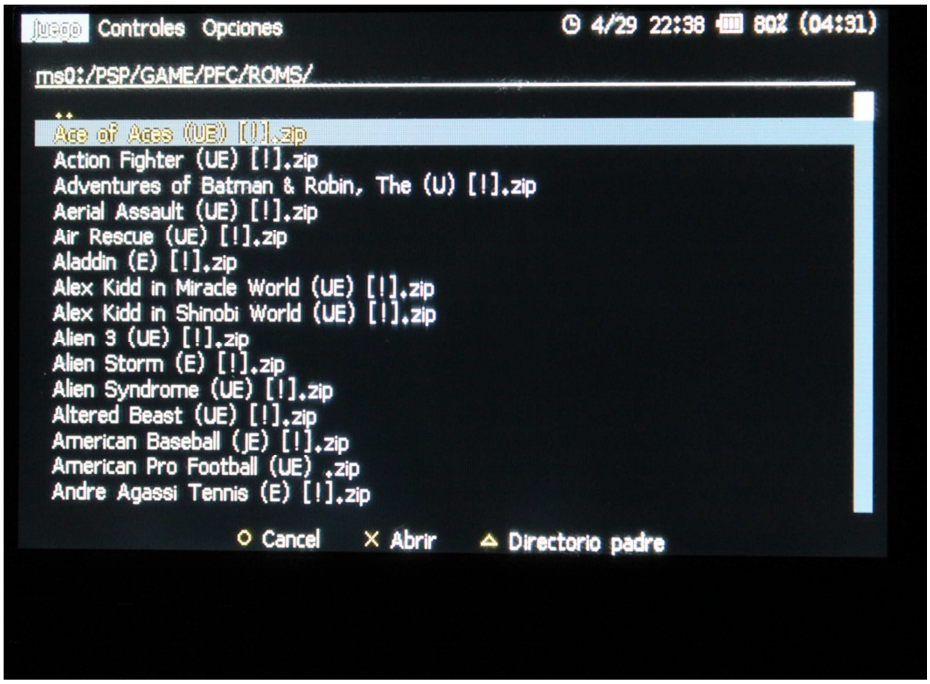
En cuanto se haya realizado esta operación, aparecerán a continuación todos los ficheros ejecutables que se encuentren en la tarjeta de memoria *Memory Stick* y se selecciona la aplicación instalada pulsando de nuevo el botón *X*.

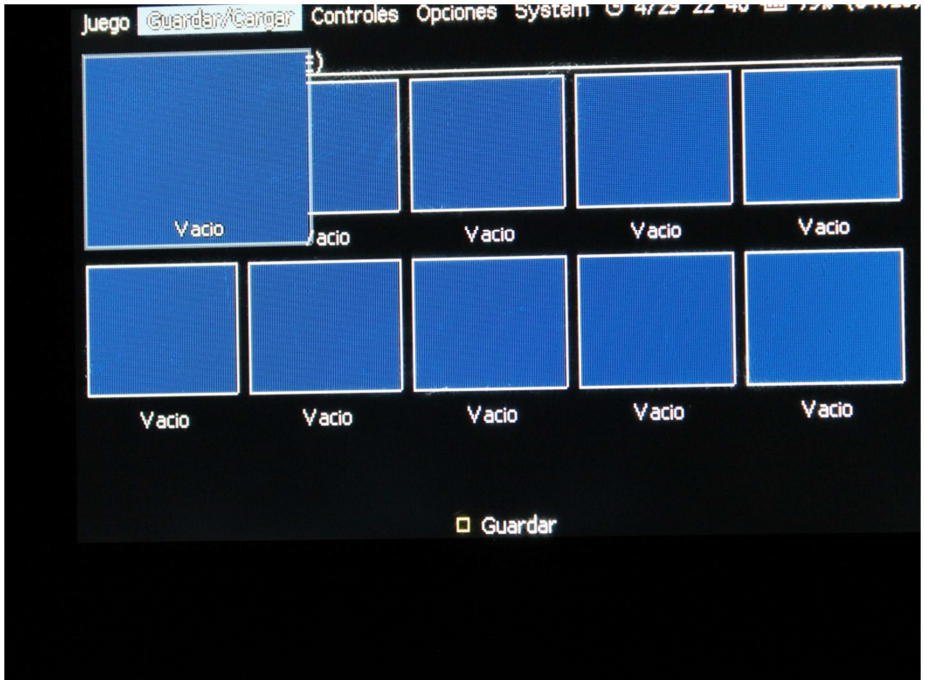
Esperamos unos segundos a que cargue la aplicación y en cuanto haya cargado, nos aparecerá una pantalla en la que se nos dice que pulsemos los gatillos *R* y *L* de la videoconsola *PlayStation Portable*. Una vez pulsados, ya se podrá acceder al menú de la aplicación.

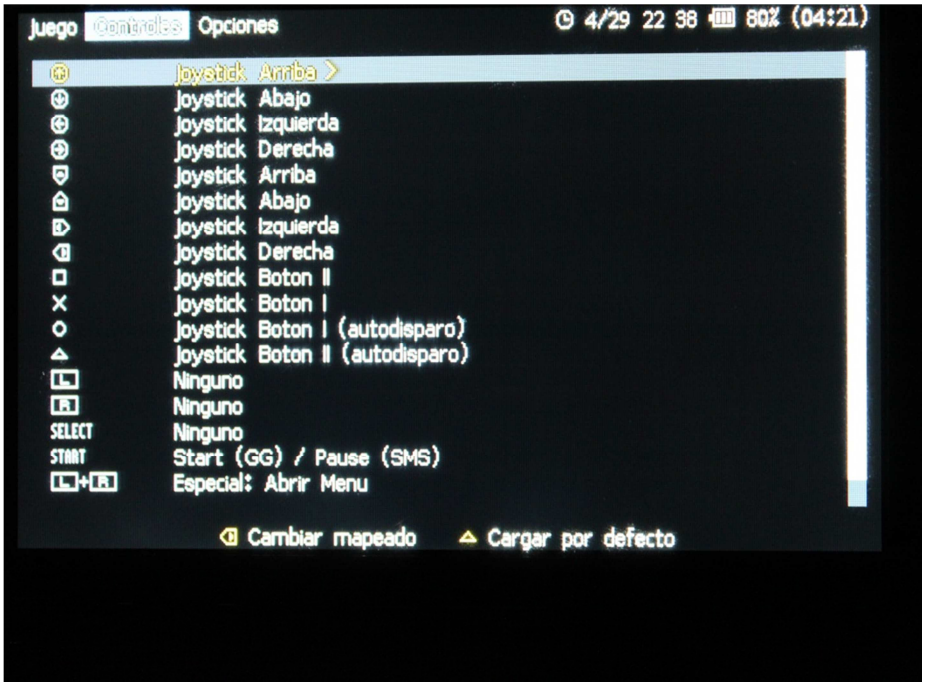


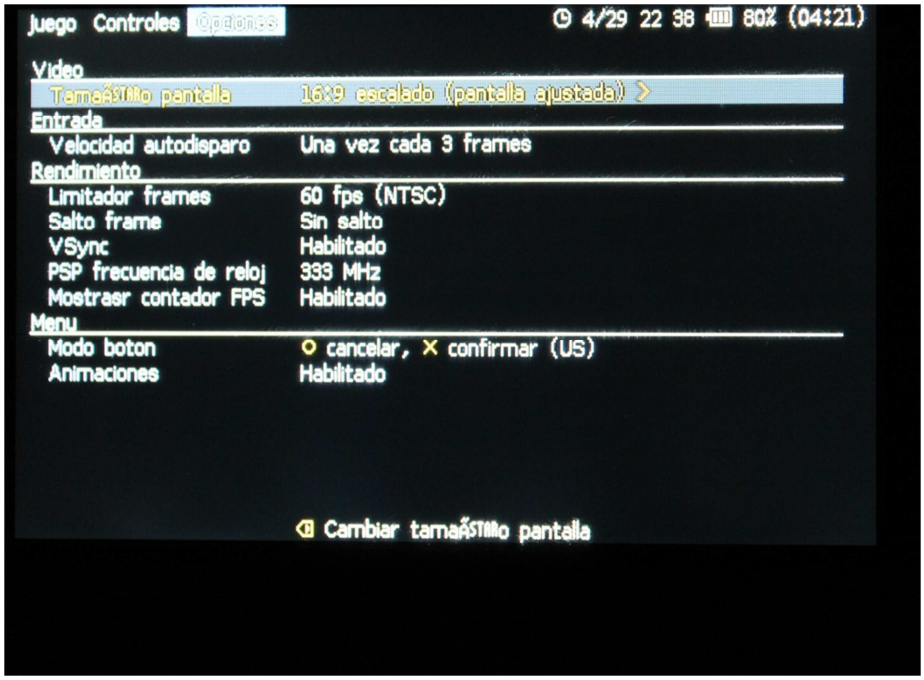
2.- Menús del juego

Cuando empieza un usuario a manejar la aplicación "EMULADOR DE MASTER SYSTEM 2 CORRIENDO BAJO PLAYSTATION PORTABLE", la primera pantalla que aparece es el Menú Principal. Desde él se puede acceder a empezar una partida de una *ROM*, los Controles para cuando el jugador juegue a la *ROM* y las *Opciones* generales de la aplicación para configurar los parámetros existentes.




Juego
Guardar/Cargar
Controles
Opciones
System
Vacio
Vacio
Vacio
Vacio
Vacio
Vacio
Vacio
Vacio
Vacio
Vacio
Guardar


Juego
Controles
Opciones
4/29 22 38 80% (04:21)
Joystick Arriba
Joystick Abajo
Joystick Izquierda
Joystick Derecha
Joystick Arriba
Joystick Abajo
Joystick Izquierda
Joystick Derecha
Joystick Boton II
Joystick Boton I
Joystick Boton I (autodisparo)
Joystick Boton II (autodisparo)
Ninguno
Ninguno
SELECT
Ninguno
START
Start (GG) / Pause (SMS)
Especial: Abrir Menu
Cambiar mapeado
Cargar por defecto



Todo esto se explicará de forma más detallada en los siguientes apartados del presente manual.

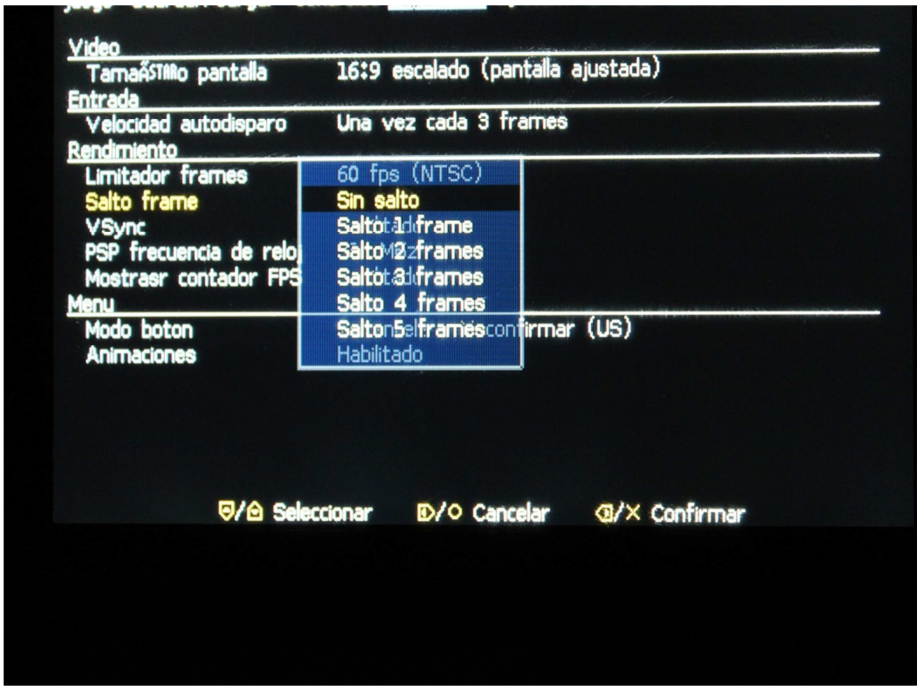
Para salir de la aplicación, se deberá pulsar el botón *HOME* de la videoconsola *PlayStation Portable* y una vez que pregunte si desea salir, elija la opción de Sí y pulse el botón *X* para aceptar.

3.- Navegar por el menú

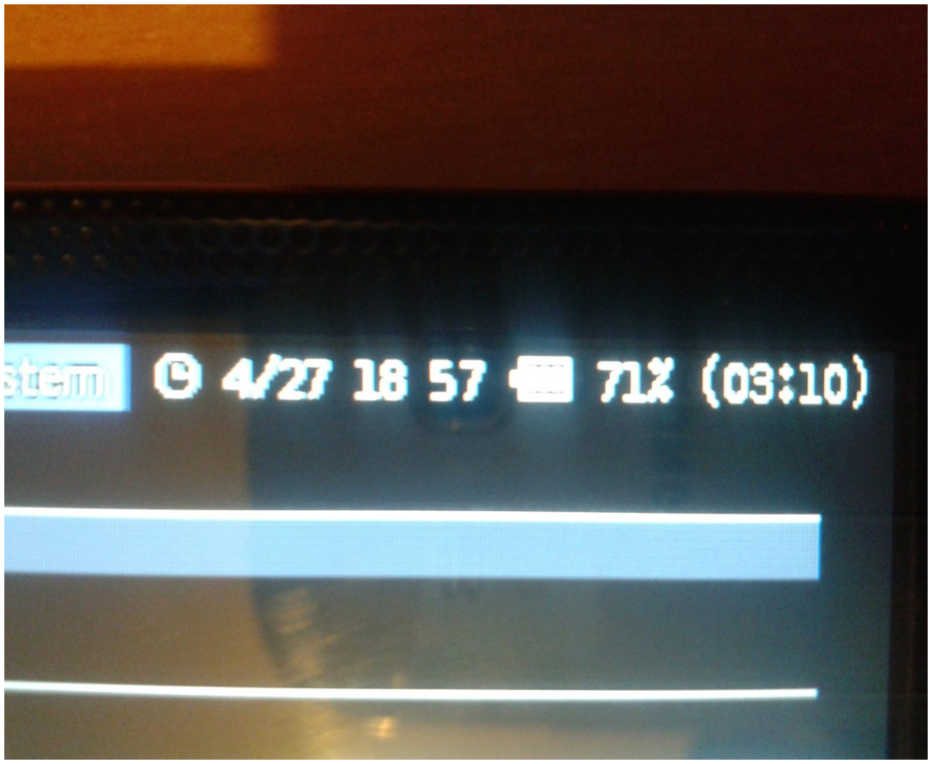
Para poder navegar por el menú principal se usarán la cruceta y los botones *L* y *R* de la videoconsola *PlayStation Portable*.

Los botones *L* y *R* se usarán para poder cambiar entre las distintas opciones principales del menú principal que aparecen (*Juego*, *Salvar*/*Guardar*, *Controles*, *Opciones* y *Sistema*) a modo de ir hacia la izquierda o ir hacia la derecha.

Para elegir una opción se pulsará el botón *X* de la videoconsola *PlayStation Portable* y se accederá a la opción elegida por parte del usuario. En este momento, se podrá usar la cruceta para cambiar entre los distintos parámetros de las opciones con los botones *arriba* y *abajo* y para acceder al parámetro elegido se deberá pulsar el botón *derecha* y luego en el caso de cambiar el parámetro, pulsar el botón *X*.



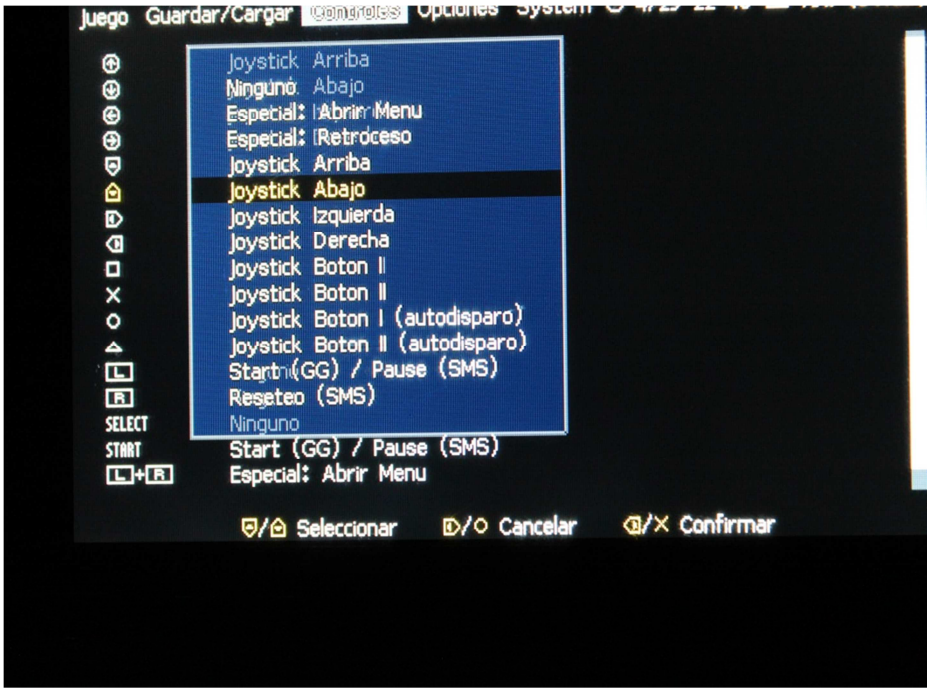
También se puede ver a modo de información en la pantalla, la fecha y hora del sistema, así como el porcentaje, tiempo restante y un dibujo del estado de la batería de forma actualizada en todo momento.



4.- Controles

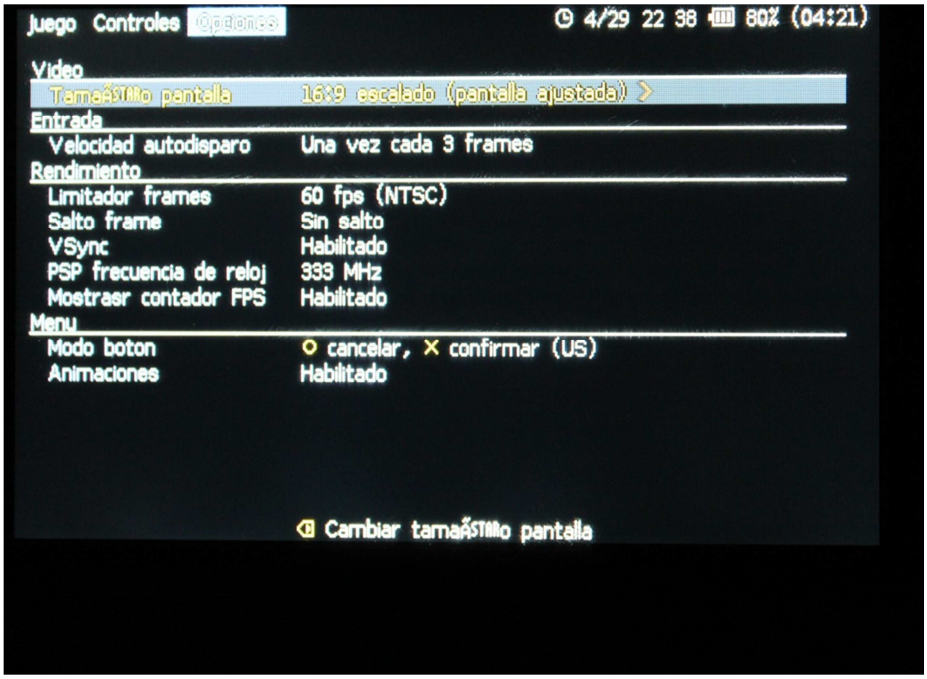
En la opción de *Controles* se permite al usuario cambiar los controles elegidos por defecto para poder manejar las partidas de las diferentes *ROM*. Para manejar cualquier videojuego, se usará la cruceta o el *stick analógico* para mover al personaje del videojuego al que se esté jugando, mientras que los botones *X* y *Cuadrado* serán los botones *1* y *2* de los control pad de la videoconsola *Master System 2*.

Todo esto se podrá cambiar al gusto del usuario y en caso de que quiera volver a utilizar los controles cargados por defecto, no deberá hacer ninguna modificación por su propia y cuenta y sólo deberá pulsar el botón *Triángulo* para que se carguen los controles por defecto.



5.- Opciones

Si el usuario accede a Opciones dentro del menú principal podrá cambiar una serie de parámetros que afectan tanto a la pantalla, como a la aplicación desarrollada y al sistema PlayStation Portable.



A continuación se explican detalladamente cada uno de los parámetros que son susceptibles de ser modificados por parte del usuario.

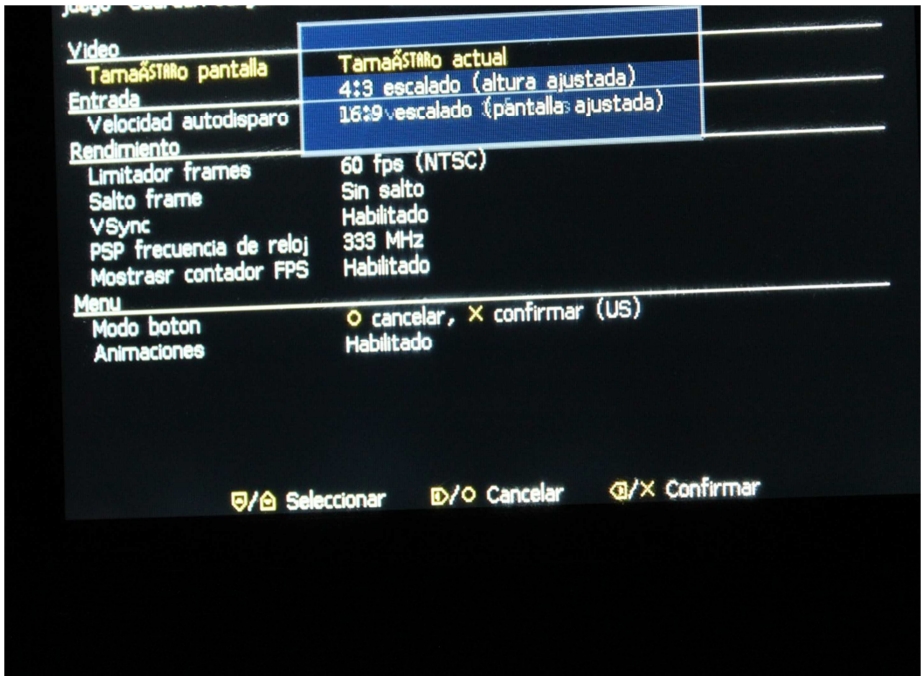
5.1.- Vídeo

En este parámetro si el usuario lo desea, puede cambiarlo para poder ver la visualización de las ROM a la hora de jugar. Las opciones disponibles que existen son las siguientes:

- Tamaño actual: presenta una visualización de tamaño pequeño que ocupa aproximadamente una tercera parte de la pantalla de forma centrada de la videoconsola *PlayStation Portable*.

- 4:3 escalado (altura ajustada): presenta una visualización de pantalla mayor ocupando aproximadamente tres cuartas partes de la pantalla de forma centrada, pero ocupando todo el alto de la pantalla de la videoconsola *PlayStation Portable*.

- 16:9 escalado (pantalla ajustada): presenta una visualización que ocupa el 100% de la pantalla de la videoconsola *PlayStation Portable*.



5.2.- Entrada

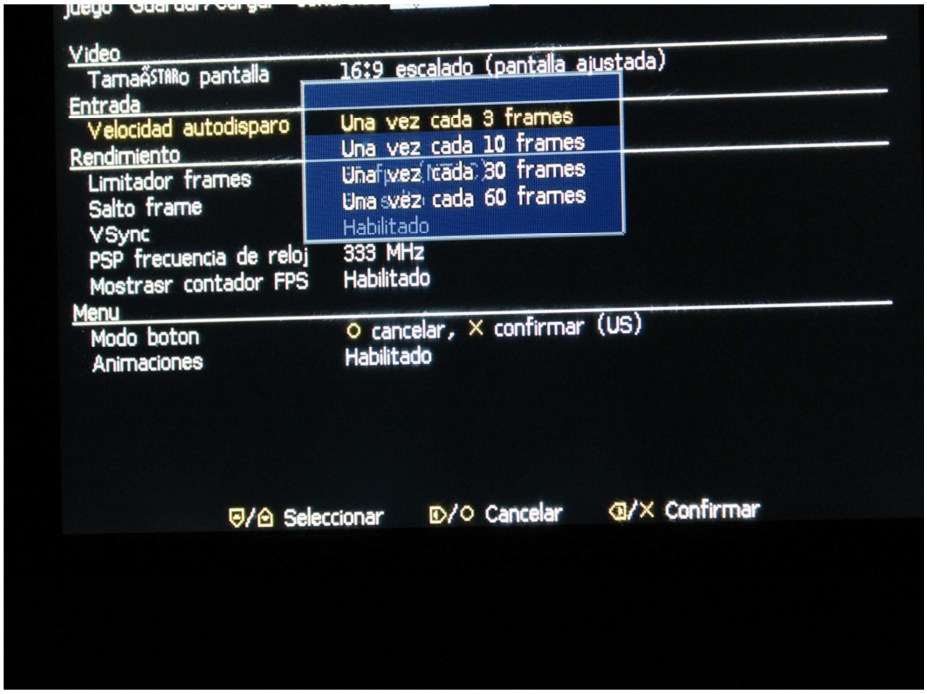
Este parámetro se puede ajustar para que el usuario en caso de escoger el botón de autodisparo, indique cuántas pulsaciones se haría por cada frame, siendo las posibilidades las que se enumeran a continuación:

- Una vez cada 3 frames.

- Una vez cada 10 frames.

- Una vez cada 30 frames.

- Una vez cada 60 frames.



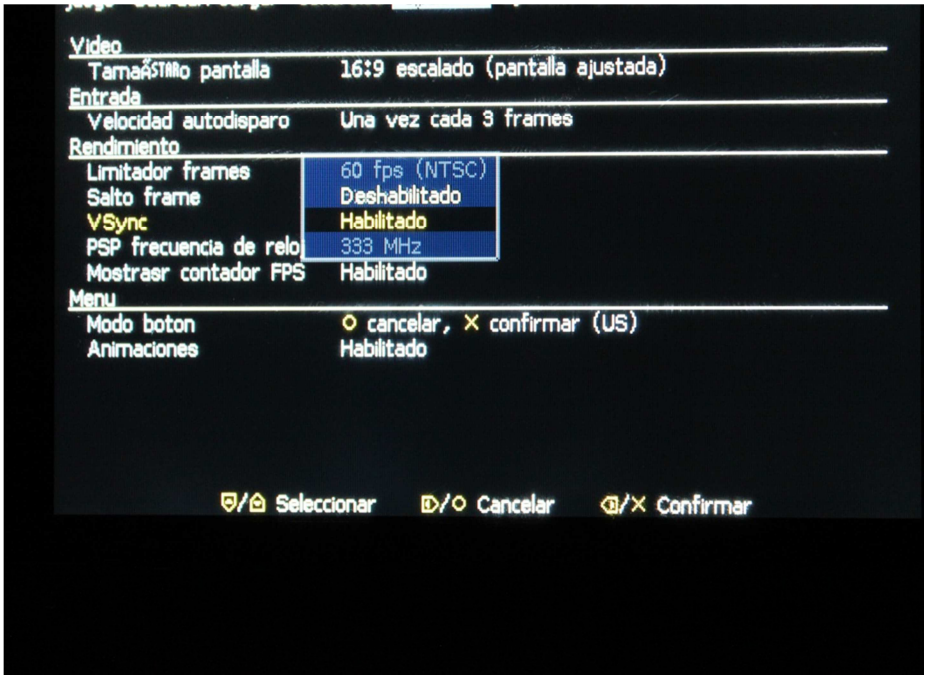
5.3.- Rendimiento

Este parámetro afecta al rendimiento tanto del emulador, como de la visualización y del sistema PlayStation portable, presentando las opciones disponibles para ser cambiadas:

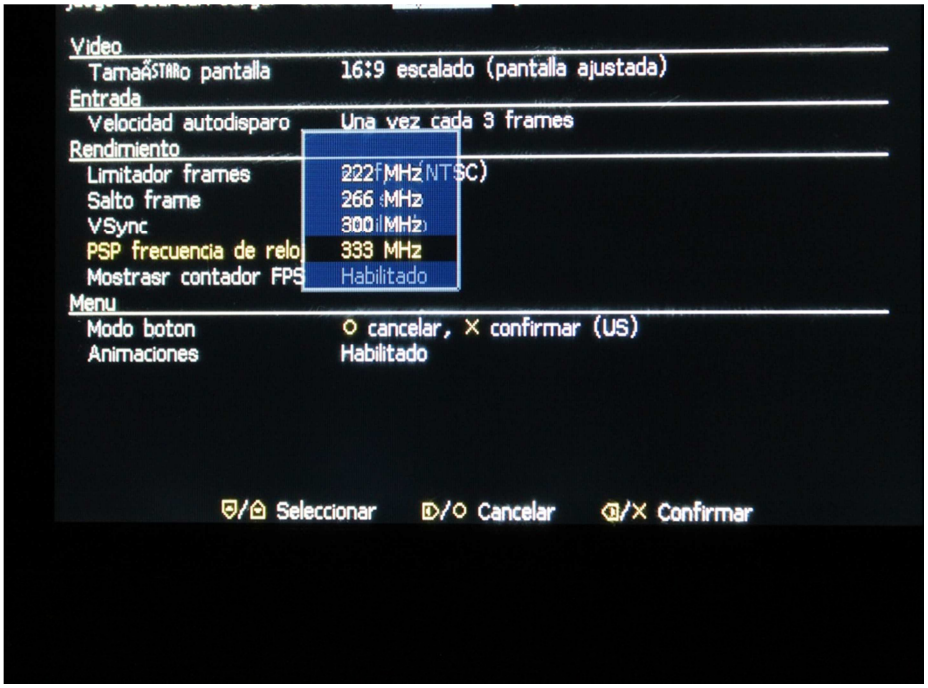
- Límite de frames: para el caso de que el juego sea de región *NTSC*.

- Salto de frames: indica el salto de frames, siendo éstos de 1 a 5 o que no tenga ningún salto.

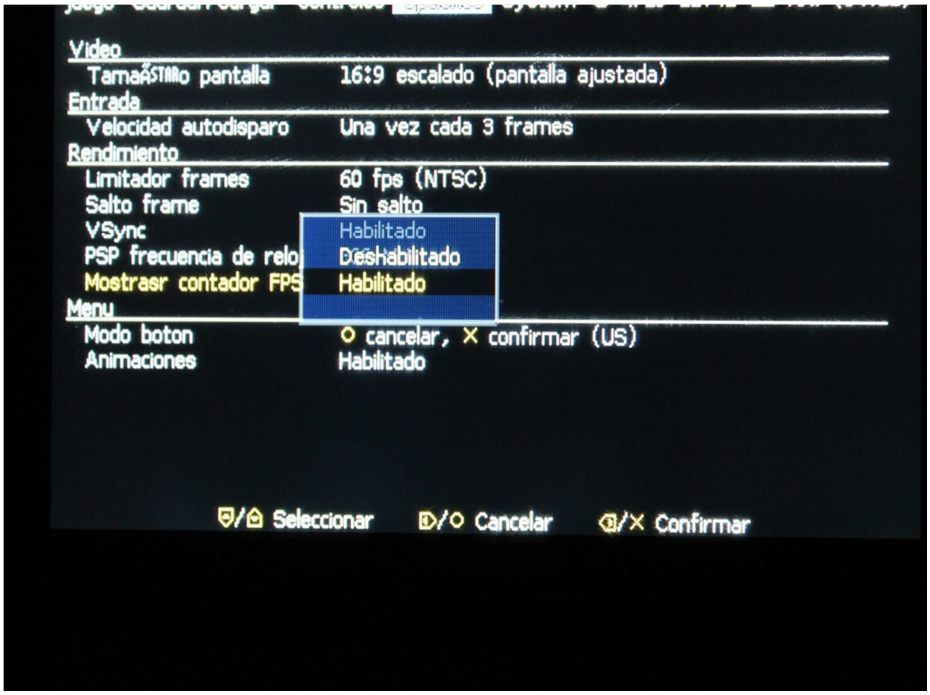
- VSync: sincronización vertical, para cuando exista scroll vertical y que la pantalla no parezca “partida”.



- Frecuencia de reloj PSP: cambio de la frecuencia de reloj de la videoconsola *PlayStation Portable* para 222 MHz, 266 MHz, 300 MHz y 333 MHz. A mayor frecuencia, mayor rendimiento y menos gasto de batería debido a la optimización conseguida del firmware de la *PlayStation Portable*.


Video
Tamaño pantalla
16:9 escalado (pantalla ajustada)
Entrada
Velocidad autodisparo
Una vez cada 3 frames
Rendimiento
Limitador frames
222 MHz (NTSC)
Salto frame
266 MHz
VSync
300 MHz
PSP frecuencia de relo
333 MHz
Mostrasr contador FPS
Habilitado
Menu
Modo boton
○ cancelar, ✕ confirmar (US)
Animaciones
Habilitado
/ Seleccionar
/○ Cancelar
/✕ Confirmar

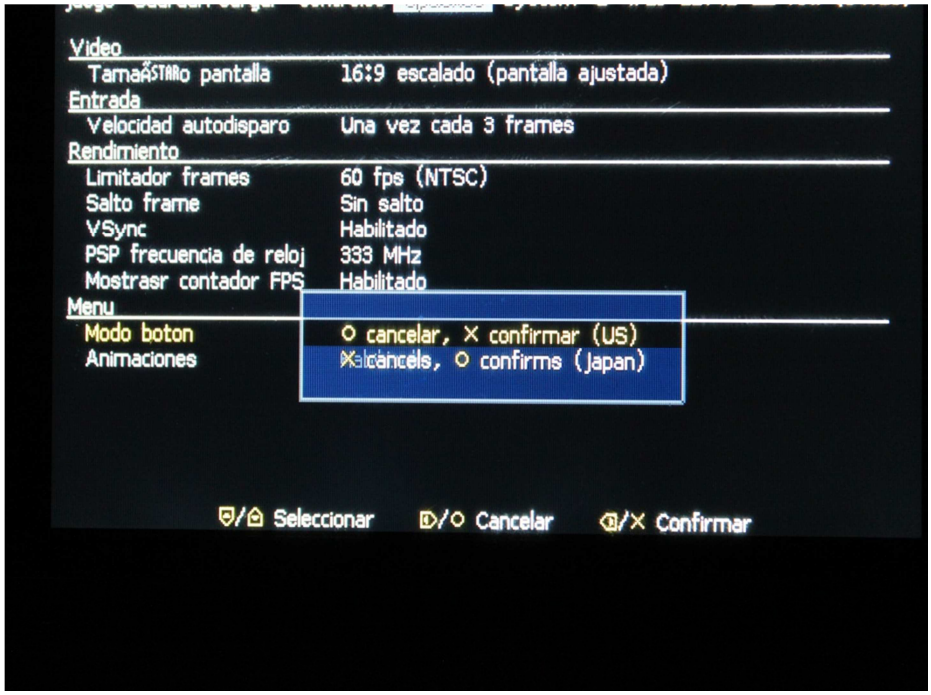
- Mostrar contador de frames: muestra una visualización en todo momento de a cuántos frames por segundo va el juego de la *ROM* que hayamos escogido.



5.4.- Menú

Este último parámetro del menú de opciones tiene a su vez otras dos opciones que pueden ser modificadas, que son las que se indican a continuación:

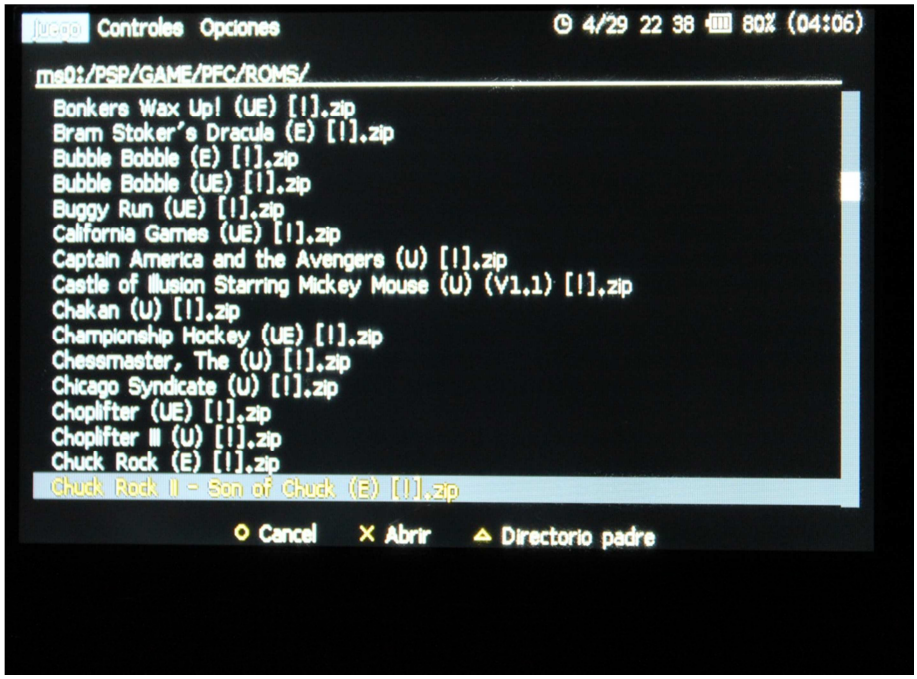
- Modo de botón: su cambio permite manejar la videoconsola *PlayStation Portable* de la misma forma que en Estado Unidos y Europa o por el contrario de la misma forma que en Japón.



- Animaciones: permite mostrar animaciones en la navegación por el menú de la aplicación por parte del usuario.

6.- Ejecutar una ROM

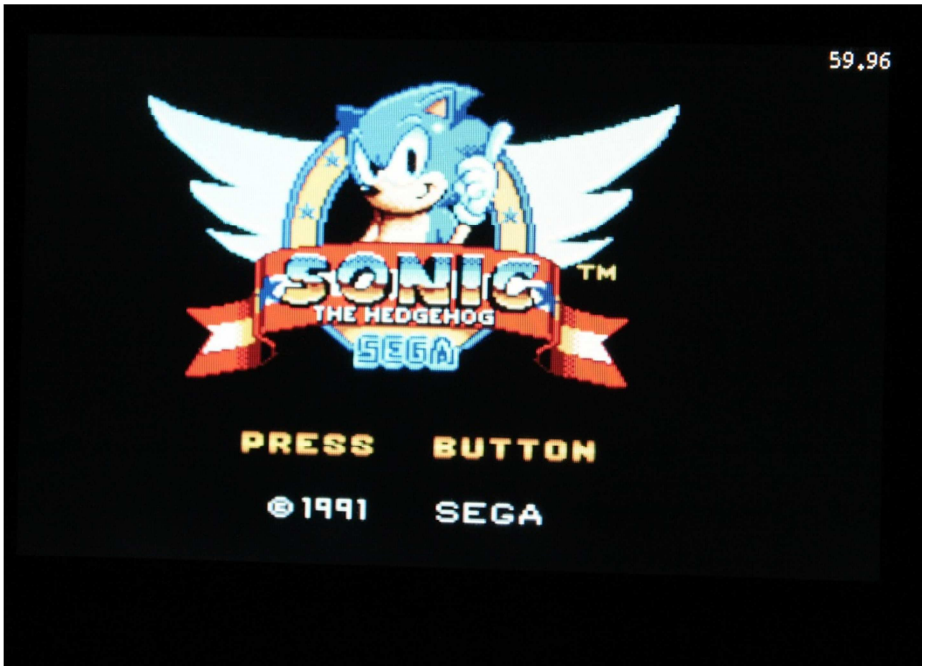
Para ejecutar una *ROM* por parte del usuario, se deberá acceder a la opción de *Juego*. En el caso de que las *ROM* estén en alguna carpeta, deberá ser seleccionada la carpeta con el botón *X* de la videoconsola *PlayStation Portable*.



Una vez seleccionada la carpeta donde se encuentran las *ROM*, el usuario podrá seleccionar la *ROM* que tenga guardada en la tarjeta de memoria *Memory Stick*.

En cuanto la *ROM* esté seleccionada, se deberá pulsar el botón *X* de la videoconsola *PlayStation Portable*.

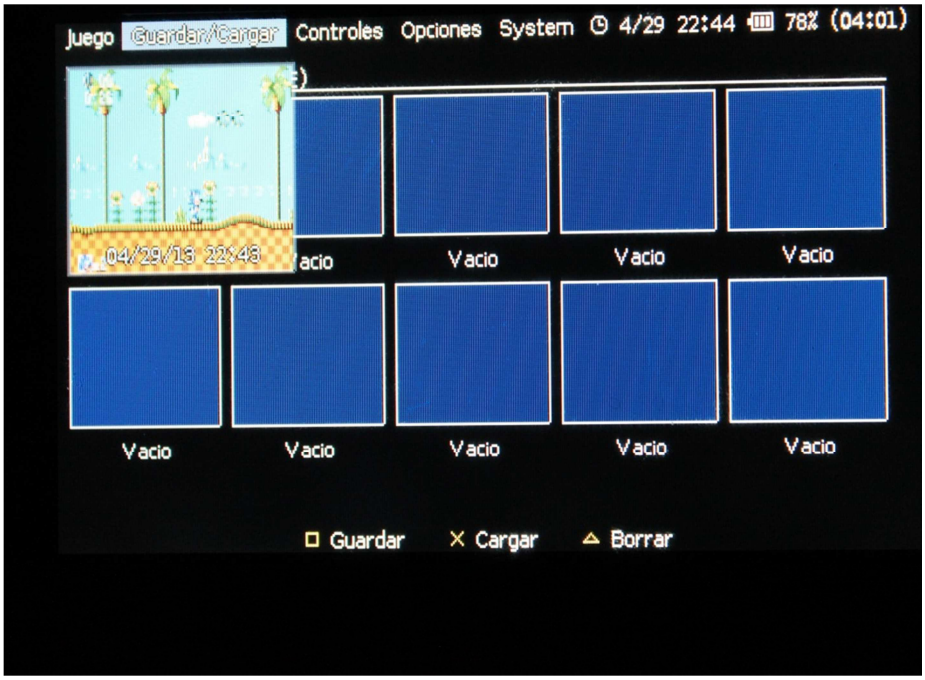
En el caso de que el usuario decida cambiar de *ROM* estando en una partida, tan sólo deberá ir a donde se encuentren las *ROM* y seleccionar la *ROM* que quiera sin tener que salir de la aplicación, ya que se ejecutará de forma totalmente automática.


59.96
SONIC
THE HEDGEHOG
SEGA
TM
PRESS BUTTON
©1991 SEGA

7.- Guardar y Cargar partida

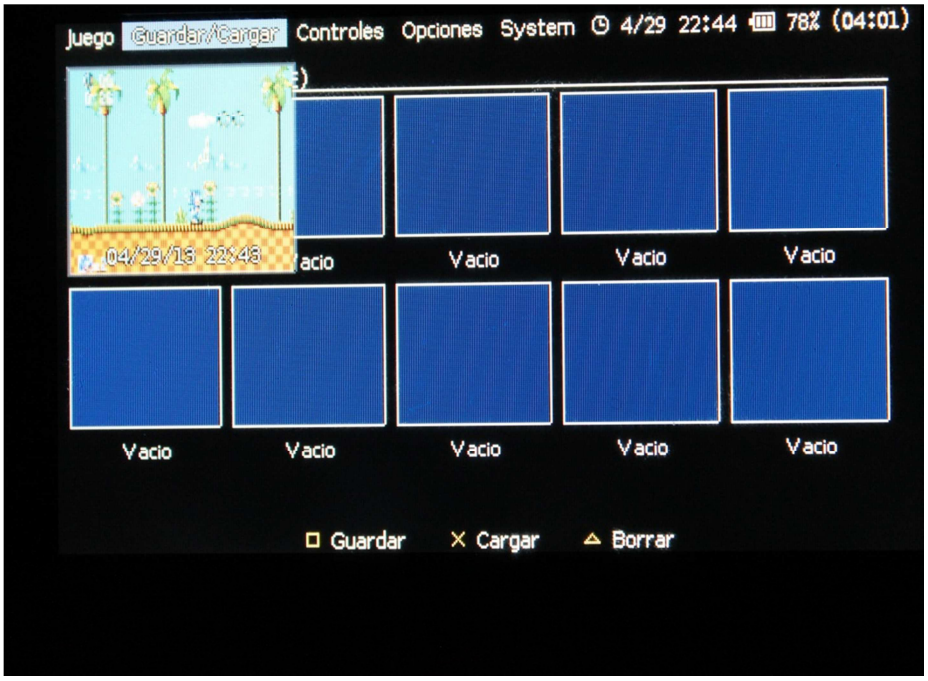
Para guardar una partida es indispensable que el usuario haya cargado previamente una *ROM* del juego al que quiera jugar. Una vez que lo haya hecho, cuando quiera guardar la partida deberá pulsar a la vez los botones *L* y *R* de la videoconsola *PlayStation Portable* y seleccionar la opción nueva que aparece que se llama *Guardar/Cargar* y pulsar el botón *X*.

En cuanto se haya hecho esto, se podrá elegir hasta 10 slots distintos para poder guardar una partida pulsando el botón Cuadrado de la videoconsola *PlayStation Portable*. En este momento, aparecerá una imagen en donde se encuentra el jugador, con información acerca de la fecha y hora en la que ha guardado la partida.



En el caso de que el usuario deseé cargar una partida, primero deberá seleccionar la *ROM* de ese juego y a continuación pulsar los botones *L* y *R* para acceder al menú principal. Se deberá elegir la opción de *Guardar/Cargar* con el botón *X* y seleccionar la partida deseada con el botón *X*.


59.95
00
0:33
x2


Juego
Guardar/Cargar
Controles
Opciones
System
4/29 22:44
78% (04:01)
04/29/13 22:48
Vacio
Vacio
Vacio
Vacio
Vacio
Vacio
Vacio
Vacio
Vacio
Guardar
Cargar
Borrar

Una vez hecho esto, el jugador puede continuar la partida desde el lugar en el que estaba con todo su estado.

8.- Sistema

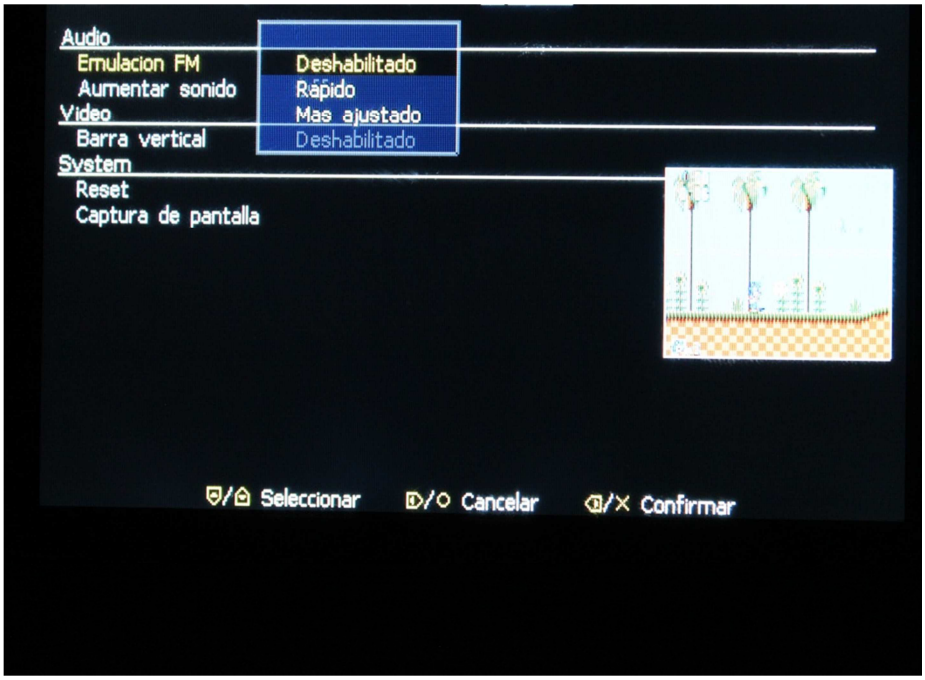
Para acceder a esta opción, el usuario deberá haber cargado previamente una *ROM* como se ha explicado anteriormente en (**Ejecutar una ROM**). En cuanto se haya hecho esto, el usuario deberá pulsar de forma simultánea los botones *L* y *R* de la videoconsola *PlayStation Portable* para poder acceder al menú principal.

En cuanto se acceda a la opción de *Sistema*, se deberá pulsar el botón *X* para acceder a los parámetros existentes que pueden ser modificados por el usuario y que se corresponden al propio emulador de *Master System 2*, visualización y nuevas características del sistema, los cuales se explican a continuación:

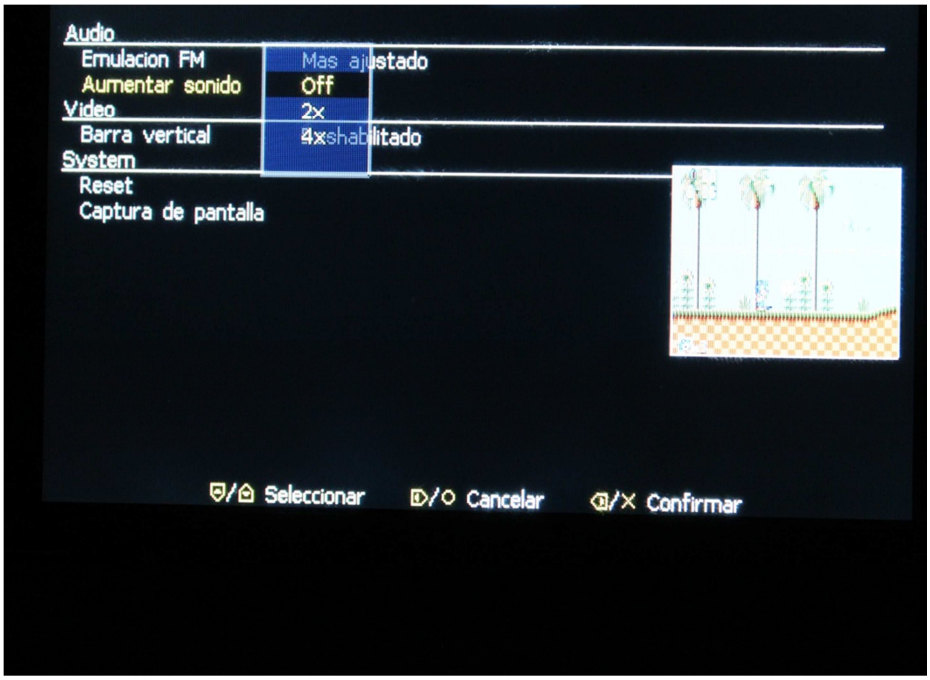
8.1.-Audio

Permite la modificación del sistema de sonido del emulador mediante los dos parámetros siguientes:

- Emulación FM: permite que los juegos que lo posean, utilicen el chip de sonido *Yamaha YM2413*, presentando una mayor calidad de sonido.

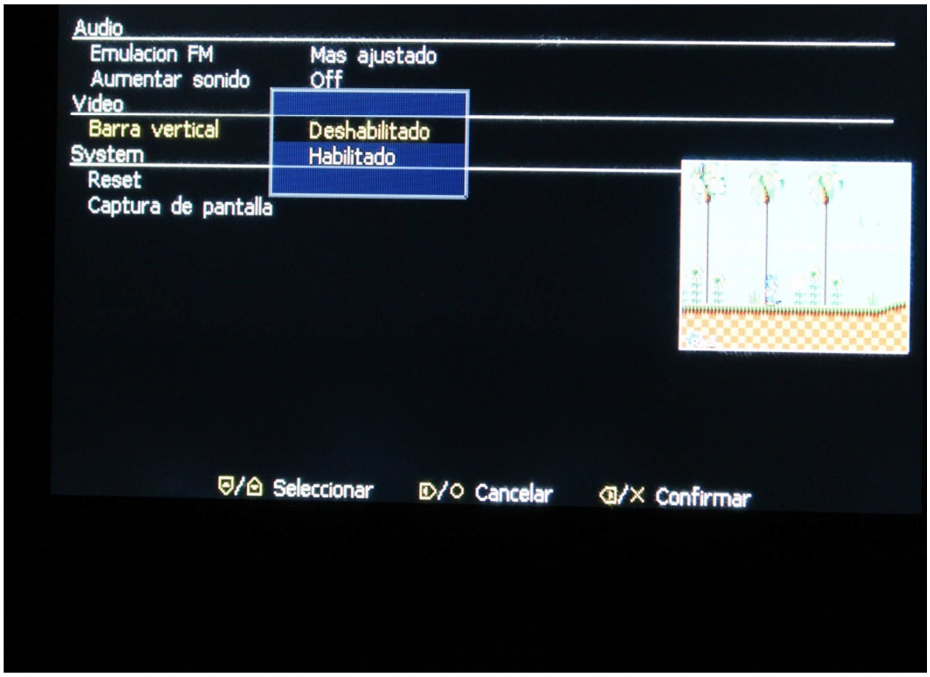

Audio
Emulacion FM
Aumentar sonido
Video
Barra vertical
System
Reset
Captura de pantalla
Deshabilitado
Rápido
Mas ajustado
Deshabilitado
Seleccionar
Cancelar
Confirmar

- Aumento de sonido: parámetro que permite que el sonido sea aumentado hasta el cuádruple de lo permitido por el sistema *PlayStation Portable*.



8.2.- Vídeo

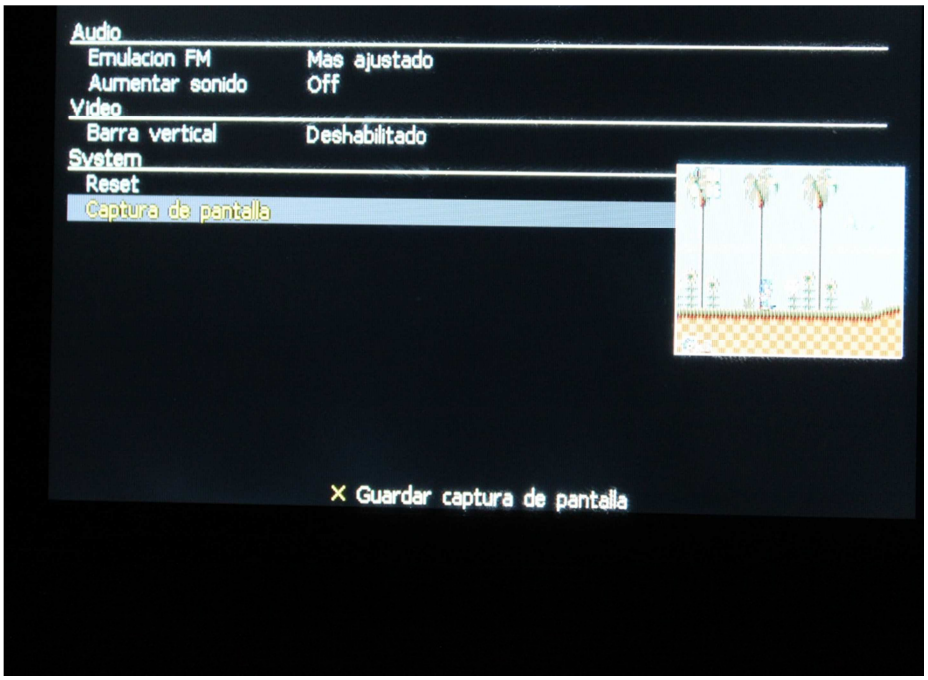
Modificar este parámetro permite mostrar una barra vertical en la parte izquierda de la pantalla que ocupa toda la altura.



8.3.- System

Cuenta con dos nuevas opciones que son las que se explican a continuación

- Resetear: parámetro que resetea la videoconsola emulada *Master System 2*.



- Captura de pantalla: permite hacer una captura de pantalla que se mostrará en el sistema de ficheros para las *ROM* con su nombre.


Juego Controles Opciones
4/29 22 38 80% (04:06)
ms0:/PSP/GAME/PFC/ROMS/
Back to the Future Part III (UE) [!].zip
Basket Ball Nightmare (UE) [!].zip
Batman Returns (UE) [!].zip
Battle Out Run (UE) [!].zip
Battleship (U) [!].zip
Black Belt (UE) [!].zip
Bomber Raid (UE) [!].zip
Bonanza Bros (UE) [!].zip
Bonkers Wax Up! (UE) [!].zip
Bram Stoker's Dracula (E) [!].zip
Bubble Bobble (E) [!].zip
Bubble Bobble (UE) [!].zip
Buggy Run (UE) [!].zip
California Games (UE) [!].zip
Captain America and the Avengers (U) [!].zip
Castle of Illusion Starring Mickey Mouse (U) (V1.1) [!].zip
CASTLE OF ILLUSION
PRESS START BUTTON
Starring
MICKEY MOUSE
Cancel
Abrir
Directorio padre

9.- Consideraciones especiales

A pesar de no tratarse en el presente proyecto y no tener ninguna competencia sobre ello, se debe indicar el manejo de las *ROM* de los diferentes videojuegos que se guarden en la *Memory Stick*.

Aunque existan aproximadamente en la actualidad 523 juegos (de diferente región) distintos para la videoconsola *Master System 2*, todos se manejan de la misma manera, es decir, con el *control pad* básico que se puso en comercialización a la vez que la videoconsola. El **control pad** cuenta con una cruceta de 8 direcciones (arriba, abajo, izquierda, derecha y las respectivas diagonales) y dos botones (**botón 1** y **botón 2**), los cuales tendrían funciones distintas para cada videojuego, por lo que el manejo de cada uno de ellos no se va a tratar, pero sí que hay que explicar el funcionamiento del **control pad** implementado en la videoconsola *PlayStation Portable*.

El manejo del movimiento de los personajes, menús, etc. de los diferentes videojuegos se realiza con la **cruceta** o el **stick analógico** indistintamente de la videoconsola *PlayStation Portable*.

Para las diferentes acciones con los botones **botón 1** y **botón 2**, se realizarán con los botones **cuadrado** y **equis** de la videoconsola *PlayStation Portable*.

El manejo tanto de la **cruceta**, **stick analógico** y botones **cuadrado** y **equis** podrán ser configurados y cambiados por parte del usuario en el menú de Controles como ya se ha especificado anteriormente en el presente manual de usuario.

Por último, indicar que el botón de Pause se encontraba en la misma videoconsola *Master System 2* y no en el **control pad**, por lo que se ha optado por representarlo por el botón **start** de la videoconsola *PlayStation Portable*, aunque es posible el cambiarlo por el botón que quiera el usuario del mismo modo que la **cruceta**, **stick analógico** y los botones **cuadrado** y **equis**.

10.- FAQ

- No se instala la aplicación en la tarjeta de memoria Memory Stick.
    - El tamaño de la aplicación es de 1’2 MB, por lo que puede deberse a dos motivos principalmente.
        - Primero: no existe suficiente espacio en la **Memory Stick**.
        - Segundo: la aplicación debe guardarse en la carpeta **GAME**, dentro de la carpeta **PSP** de la **Memory Stick**, tal como se indica en el manual de instalación.
- Aparece como corrupto el archivo en el XMB de la PlayStation Portable.
    - El emulador no ha sido guardado correctamente tal como indica el manual de instalación, ya que ha sido probado en varias **PlayStation Portable** con firmware distintos y modelos distintos.
- El emulador carga, pero no ejecuta ninguna ROM.
    - El emulador sólo ejecuta **ROM** con extensión **.sms** y que se encuentren comprimidas con un formato **.zip**, por lo que otro tipo de extensión en la **ROM** u otro formato de compresión de datos no está permitido.
- No aparece ninguna ROM.
    - El usuario debe crear una carpeta donde haya guardado el emulador que se llame **Juegos** o **ROMS** por ejemplo y todas las **ROM** deberán ser guardadas en esa carpeta para poder verse en el sistema de directorios creado.