UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Evaluación e implementación de juegos clásicos para Nintendo DS

Proyecto Final de Carrera

Ingeniería Técnica Informática de Gestión (ITIG)

Autor: María del Carmen Sarriá Trejo
masartre@ei.upv.es

Director: Manuel Agustí Melchor

magusti@disca.upv.es

Septiembre 2012

TABLA DE CONTENIDOS

1. LISTADO FIGURAS

2. LISTA DE DEFINICIONES

Investigando sobre homebrew para NDS me he encontrado [0] con numerosos términos, a continuación expongo parte de ellos y su significado:

Backup: Copia de seguridad. Se pueden hacer backups de archivos en el ordenador, de discos de música, de juegos... etc. En el caso de la DS es el término empleado para hablar de la copia de un juego comercial.

Boot Tools: Una Boot Tool, es un dispositivo que envía la señal de arranque al Slot de GBA (Slot-2) permitiéndonos correr programas desde aquí al encender nuestra consola. (Actualmente muy poco utilizados gracias a las Nuevas Flash Cards de Slot-1).

Bug: Palabra proveniente del inglés, se utiliza para designar un "error de software". Un bug es el resultado de un fallo de programación introducido en el proceso de creación de un programa.

Cheater o chetero: Persona que acostumbra a usar cheats, generalmente en partidas multijugador, lo que les da una injusta ventaja frente a los otros.

Cheats: Son modificaciones que se pueden hacer en una ROM, generalmente para darnos ventaja en el juego. Son de lo más variopintos, como infinitas vidas o tener ya el 100% del juego.

DLDI : No es más que un estándar creado. Un formato de Homebrew compatible con TODAS las Flashcards actuales. Os dejo enlace a un documento que explica el DLDI a fondo.

DMA: El modo DMA hace que la transferencia de datos secuenciales de la ROM/Slot DS con la memoria sea más rápida, usando el controlador DMA de la DS para transferir datos directamente a la memoria.

Dump / Dumpeo : Significa volcar, descargar. Un ejemplo puede ser el traspasar la roms de un cartucho de NDS al PC.

EFS: Literalmente "Sistema de archivos Embebido". Sistema de archivos similar a FAT con la característica principal en la que va introducido dentro del ejecutable. A diferencia del método FAT, es soportado por la totalidad de Flashcards del mercado, pero puede tardar hasta varios minutos en iniciarse la primera vez. En caso de que no

arranque, es recomendable usar un lanzador externo como el DSOrganize.

Emulador: Sistema que imita el funcionamiento de otro.

Exploit : Cuando se habla de Exploits, se hace referencia a determinados fallos de seguridad, que "alguien descubre", y que permiten hacer un nuevo uso de una máquina o soporte.

FAT: Esto más que una duda de Nintendo DS, es una duda de informática. FAT es simplemente un formato de archivos, un modo de organizar la información de manera interna. Este sistema permite acceder a archivos distintos al ejecutado, pudiendo así crear archivos de guardado, entre otros usos. Se ha demostrado que la mayoría de las Flashcards R4, sean fakes o clones, sufren problemas con este tipo de formato.

Firmware: Conjunto de archivos que están en la memoria interna de nuestra Flashcard. No puede actualizarse en todas las flashcards, y en las que se puede debe hacerse con cuidado, pues podríamos estropearla.

Fix: Viene de la palabra inglesa reparar, en la scene sirve para indicar si una versión de una ROM/Homebrew tiene algún fallo (bug) arreglado. En el caso concreto de una rom, normalmente indica si está editada para saltarse la seguridad, de esta forma, la rom funciona en las Flash Cards.

FlashCards: Son accesorios que se conectan a la Nintendo DS en el Slot1 (juegos NDS) o en el Slot2 (juegos GBA) o en ambos a la vez para poder ejecutar aplicaciones y juegos homebrew, emuladores o nuestras copias de seguridad de juegos de NDS. Hoy en día son tremendamente seguras y nuestras DSs no corren ningún tipo de peligro.

Formatear: Borrar completamente el contenido de un volumen extraíble y agregar de nuevo un sistema de archivos, suele usarse cuando dicho puerto empieza a tener archivos corruptos.

Glitch: Es un error en la programación (bug) que de algún modo no afecta negativamente al juego, si no puedes aprovecharte de él.

Homebrew: Software hecho en casa, de programas y utilidades realizadas por la scene sin ningún tipo de apoyo oficial.

Homebrew Menu: Lanzador externo creado para ejecutar homebrew que usen las NitroFS y, en consecuencia, el protocolo ARGV.

Kernel: Conjunto de archivos que van en la raiz de la MicroSD, y que sirven para que nuestra flashcard funcione. Haciendo un mal símil, es como su sistema operativo.

Libnds: Librerías de programación para la DS, preparadas para usar con C/C++. Creadas por dovoto y joat y actualmente mantenidas por Wintermute.

Lua: Lenguaje de programación de uso general interpretado basado en scripts, actualmente ya no se programa con este lenguaje para la DS.

DSLua: Reproductor de scripts escritos en Lua para la NDS.

Memory Stick: Es un formato de tarjeta de memoria extraíble (memoria flash). La Memory Stick es utilizada como medio de almacenamiento de información para un dispositivo portátil. En el caso de la NDS, se utiliza para guardar diversos archivos multimedia que la NDS pueda reproducir o ejecutar. Podemos distinguir 4 tipos para la DS, siempre que tengamos una flash card:

- MicroSD: Tarjeta de memoria flash más pequeña que la MiniSD, desarrollada por SanDisk.
- MiniSD : Tarjeta de memoria flash con un tamaño intermedio entre la SD y la MicroSD.
- Secure Digital o SD: Tarjeta de memoria flash.
- CompactFlash (CF): Antiguo formato flash utilizado en flash cards arcaicos de Slot2

Modo Ad-hoc: Método de interconexión wifi en el que dos equipos están en el mismo nivel (conversación de igual a igual)

Modo infraestructura: método de conexión wifi en el que varios equipos (clientes) se conectan a un único equipo (servidor).

NitroFS: Sistema de archivos similar a FAT con la característica principal en la que va introducido dentro del ejecutable, pero a diferencia de EFS, se inicia casi al instante. Usa el protocolo ARGV para la carga de archivos. Su principal desventaja es que muy pocas Flashcards soportan este protocolo, por lo que en tal caso se debe lanzar desde el Homebrew Menu.

NFlib: Librerías de programación para la DS, preparadas para usar con C/C++. Obra del coder español NightFox.

PAlib: Librerías de programación para la DS, preparadas para usar con C/C++. Actualmente estas librerías están decayendo en uso.

Píxel: Es la menor unidad en la que se descompone una imagen digital, ya sea una fotografía, un fotograma de vídeo o un gráfico en formato de mapa de bits.

Protocolo ARGV: Sistema utilizado por las NitroFS para cargar archivos. Todo homebrew que use este protocolo necesitará ser lanzado a través del Homebrew Menu (a excepción de un par de Flashcards).

RAR: Extensión de ficheros comprimidos. Necesitaras este compresor, ya que es el utilizado, en las descargas que encontraras en NDS.scenebeta.

Release: Significa lanzar... o lanzamiento. Cuando se termina un producto, es lanzado al público.

Ripear: Llamamos ripear al procedimiento de la extracción de datos de un archivo (video, audio, idiomas etc...) generalmente para ahorrar espacio.

Roms: Lo mismo que "ISOs". Son los juegos originales de la Nintendo DS, codificados en un archivo que se importa a la tarjeta de memoria y es jugable usando nuestra Flashcard. Tienen extensión .nds. También podéis encontrar Roms para otras consolas (si instaláis un emulador en la Nintendo DS, por ejemplo, necesitareis las Roms de esa consola, las roms son los juegos).

Slot: Los Slot son las dos ranuras que tiene la NDS para introducir en ella los cartuchos de DS, GBA, Flash Cards.... Y distinguimos dos:

- Slot 1: Ranura más pequeña para cartuchos de Nintendo DS.
- Slot 2: Ranura más grande para cartuchos de GameBoy Advance (únicamente en DS Fat y Lite).

RSS (Really Simple Syndication): Es un formato de datos que es utilizado para sindicar (re difundir) contenidos a suscriptores de un sitio web.

Scene: Cuando se habla de scene se habla de un "movimiento de programadores" que se dedican a desarrollar programas para un soporte determinado. En nuestro caso Scene NDS.

SDHC: Formato nuevo de tarjetas de memoria flash SD v2.0.

Stylus: Boligrafo que inluye la Nintendo DS para tocar la pantalla táctil.

Trimmear: Extraccion de datos de un juego NDS para ahorrar espacio. No ha de confundirse con ripear. Ya que aqui se cogen datos "inservibles". Son datos vacíos, sin información. Están para que el juego funcione bien en la tarjeta original

WEP: Wireless Equivalent Privacy, sistema de encriptación Wi-Fi

Wi-Fi: Es un conjunto de normas para redes inalámbricas. Wi-Fi se creó para ser utilizada en redes locales inalámbricas, pero es frecuente que en la actualidad también se utilice para acceder a Internet. Hay varios programas para nuestra NDS que utilizan Wi-Fi.

WPA: Wi-Fi Protected Access, sistema para proteger las redes inalámbricas (Wi-Fi); creado para corregir las deficiencias del sistema previo WEP. Este tipo de sistema sólo es soportado por la DSi y la DSi XL, y sólo en juegos comerciales.

3. PRÓLOGO

Lo que pretendo con este trabajo es realizar un juego con diversos "mini juegos" utilizando las herramientas libres de las que se dispone cuando se quiere desarrollar *homebrew*, en este caso, para Nintendo DS.

Los *Homebrew* son el nombre que reciben las aplicaciones y juegos no oficiales creados generalmente para consolas de videojuegos propietarias.

En el primer apartado expondré las características de la propia consola y su hardware, que será necesario conocer para saber qué límites tenemos al programar para ella.

A continuación detallaré cómo preparar el entorno de desarrollo en sistemas Windows.

He utilizado devkitPro como kit de desarrollo de software (SDK) y explicaré cómo configurar el entorno de desarrollo integrado Eclipse para este SDK.

La librería que he elegido para la implementación de este juego es PAlib, una librería de alto nivel que utiliza devkitPro, a continuación expondré algunos datos sobre ella y como configurarla correctamente.

Palabras clave: Nintendo DS, implementación, evaluación, aplicación, juego.

4. INTRODUCCIÓN

4.1 Consolas

Nintendo DS es la videoconsola portátil de Nintendo perteneciente a la séptima generación de consolas (hasta ahora, las generaciones venían marcadas por las consolas de sobremesa). Salió al mercado en 2004 en Japón y EE.UU. llegando a Europa en marzo de 2005.

En enero de 2011, la Nintendo DS se había convertido en la videoconsola más vendida de la historia, adelantando a Sony PlayStation 2.

Hasta la fecha, han salido seis versiones de la consola (figura 1): la original; DS Lite, reduciendo el tamaño; DSi, con mejoras hardware y dos cámaras; y DSi XL, un modelo más grande de la DSi que incluye pantallas más grandes que su predecesora; la 3DS cuya atracción principal es poder mostrar gráficos en 3D sin necesidad de gafas, gracias a la autoreoscopia, la consola es retrocompatible con los modelos anteriores; la 3DSXL al igual que en el e incluirá mejoras en la duración de la batería.

Figura 1: Nintendo DS, Nintendo DS Lite, Nintendo DSi y Nintendo DSi XL,Nintendo 3DS y Nintendo 3DS XL.

4.2 Homebrew NDS

Aunque DS significa tanto Dual Screen como Developer's System, la consola no permite desarrollar libremente aplicaciones para ella.

Por lo que se necesitan herramientas para que sus aplicaciones puedan ser creadas y se ejecuten correctamente en la Nintendo DS.

4.2.1 Flashcarts

Son cartuchos creados por terceras compañías, compatibles con los slot-1 ó slot-2 de la consola. Su función es cargar binarios de homebrew o copias de seguridad mediante memoria reescribible.

Los flashcarts de slot-2, más antiguos, suelen requerir parchear los ejecutables de la consola (llamados ROMs por ser imágenes del contenido de un cartucho de memoria ROM) y usar un *Passme* (una técnica creada por la *scene* de Nintendo DS para simular la autentificación del software a cargar) o alterar el firmware de la consola.

Los flashcarts de slot-1 en cambio no suelen requerir parchear las ROMs, ni necesitan técnicas adicionales para ejecutar homebrew o copias de seguridad. Además, como ventaja adicional, deja libre el slot-2, donde pueden conectarse extras, como ampliación de la memoria de la consola.

La legalidad de los flashcarts ha sido puesta en duda por Nintendo en varias ocasiones, pero los jueces han dado siempre la razón a los vendedores/usuarios de estos cartuchos.

Prácticamente la totalidad de flashcarts en el mercado actual, tanto de slot-1 como de slot-2, permiten la carga de homebrew sin problemas.

Actualmente hay numerosas webs donde la gente sube su homebrew para que los demás disfruten de ello, eso si el código fuente no suelen subirlo.

Pero de este modo se esta creando una especie de movimiento pro juegos libres, ya no el software en general, sino juegos aplicaciones, dentro de el marco de lo legal y hechas por uno mismo, se pueden ver cosas muy interesantes, una de las paginas de este tipo es ***scenebeta,*** en la que hay muchísima gente y la verdad es que controlan mucho la legalidad, y abren temas muy interesantes, y es que la creación de homebrews esta aumentando cada vez mas, ahora están empezando a aparecer sw para la creación de homebrews para 3DS, aunque aun no están muy perfeccionados.

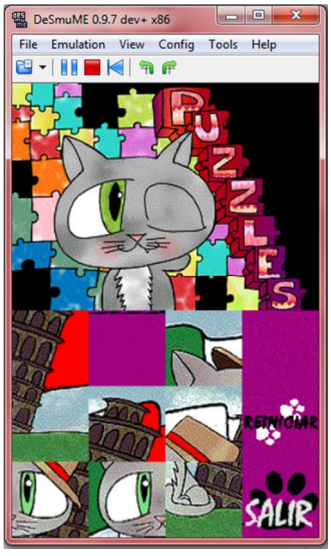
4.3 Motivación para hacer este trabajo

La idea de realizar vino de ver a los niños jugando con las consolas constantemente, solo a juegos violentos, y me puse a pensar en que cuando yo era pequeña jugábamos a juegos como el tres en raya y nos divertíamos , sin necesidad de reventar zombis ni matar a nada, y pensé que podría hacer juegos clásicos, de este modo los pequeños verían que se pueden divertir de otro modo, y los mayores podrán recordar su infancia.

4.4 Trabajo Previo

Este proyecto parte de un trabajo que he realizado este curso para la asignatura de IMD. En este trabajo también realicé la parte visual para conseguir un producto original, de este modo no trabajé solo en el código sino también en el diseño de la interfaz

Para IMD implementé únicamente dos juegos y una galería de imágenes (véase la *figura 4*) , todo táctil y con menús (como se puede apreciar en la *figura 5*), pero decidí que haría un tres en raya imitando el comportamiento humano (que hubieran algunas veces que errara, para lo que hice que algunos movimientos fueran aleatorios, véase la *figura 2*) y un puzle que no fuera el clásico puzle al que juegas una vez y ya fuera mecánico (tipo puzle de cartón de colocar fichas), de este modo hice un puzle diferente que se trata de deslizar las fichas dejando siempre un hueco libre que es al que se pueden desplazar éstas (véase la *figura 3*), para realizarlo me inspiré en un juego que había cuando yo era pequeña (véase la *figura 6*).



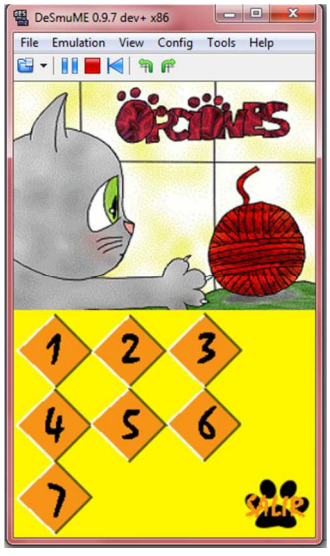


Figura 2: Tres en raya empate. *Figura 3: Puzle sin montar.* *Figura 4: Galería de imágenes.*

Figura 5: Menú de juegos. *Figura 6: Puzle aislado deslizante.*

4.4.1 Desarrollo del trabajo previo

Lo primero que hice fue la parte gráfica, a continuación muestro algunos ejemplos:

Figura 7:Bocetos

Fui programando por fases, primero probé a cargar texto en dos consolas y a centrar texto en pantalla:

Figura 8:Texto1

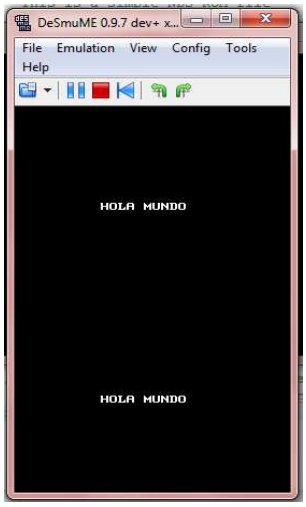


Figura 9:Texto2

Lo siguiente fue probar a cargar texto en colores diferentes:

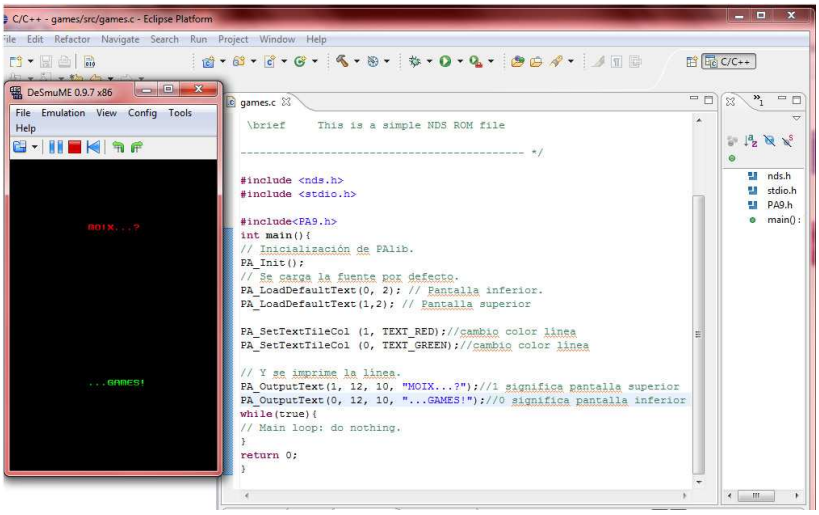


Figura 10:Texto colores

Llegando a la parte más llamativa la primera carga de fondos:

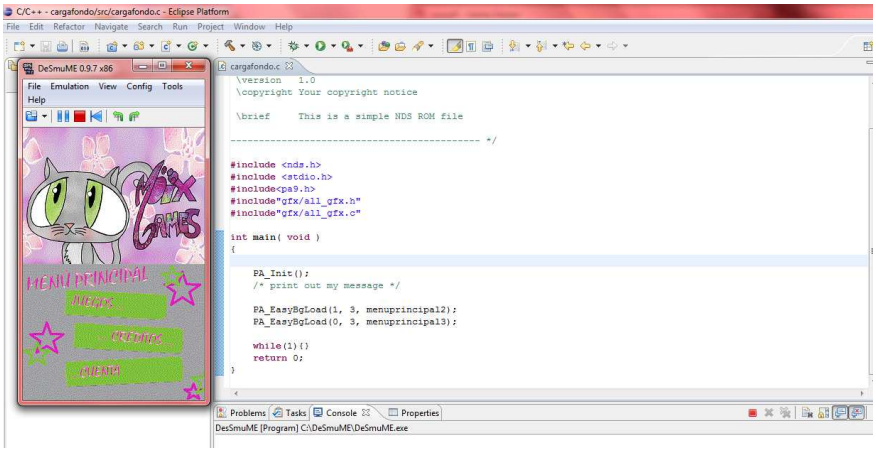


Figura 11:Carga fondos1

Lo siguiente que hice fue configurar los menús para pasar de un sitio a otro, y cuando esto ya funcionaba me puse a desarrollar el tres en raya.

Del cual me fui dando cuenta de los fallos de funcionamiento al probarlo, lo último fue un fallo al dibujar en la casilla 7 porque si yo hacía clic en la 7, la máquina tenía que dibujar su figura en la casilla 4, pero había un fallo de asignación que solucioné sin ningún problema.

En cuanto al diseño de las figuras las cree como sprites de 32x32 poniéndole el fondo como el del tablero para que parecieran fichas con borde transparente.

Figura 12:Empate3

Figura 13:Perdido3

Para decidir como realizar el tres en raya decidí hacer como una mini-encuesta a gente.

Ganó que la partida acabará porque el tablero estaba lleno porque decían que sino era muy repetitivo, lo ultimo que hice en el tres en raya fue insertar un marcador en la pantalla de arriba abajo numero de partidas ganadas, empatadas y perdidas, estos números solo se ponen a 0 cuando se sale del tres en raya.

He implementado el tres en raya de forma que la máquina no gane siempre y haga algunos movimientos aleatorios para que sea mas similar a jugar con una persona , ya que cuando hice las encuestas me dijeron que empatar o perder siempre no era divertido así que sería mejor que la

máquina se equivocara a veces pero que fuera aleatorio porque si no se le ganaría siempre con la misma jugada.

> Con respecto al puzle: tras mucho pensar al final decidí no hacer un puzle normal sino el típico puzle que va como en una cajita de plástico y se desplazan las fichas, para mayor facilidad ni si quiera hay que deslizar correctamente , solo hay que hacer click en la pieza que quieres desplazar de las que hay pegadas al agujero y esta se coloca en esa posición , hay un botón para reiniciar que te las desordena otra vez cuando quieras.

Otra cosa añadida es que la imagen consta de 9 trozos como uno tiene que ser un hueco solo salen 8 , pues bien estos 8 salen de forma aleatoria dando opción a 9 puzles diferentes con uno solo, ya que unas veces faltara un ojo, otras una pata , etc…

También desordena de forma aleatoria con lo que cada una de las fichas puede estar en cualquiera de las nueve posiciones y el agujero también , y también puede no estar (la ficha),con lo que con 9 sprites y una imagen (la montado) obtenemos mucha jugabilidad.

4.5 Objetivos

- Explicar cómo instalar los programas y cuáles usar, tanto para MS/Windows como para GNU/Linux, ya que en el trabajo de IMD, la programación la realice en el entorno MS/Windows.

- Realizar una serie de mejoras en los juegos ya existentes:
  - Tres en Raya:
    - Que sólo se dibujen tres fichas de máquina y tres de jugador, y éstas las puedas desplazar con el lápiz táctil.
    - Establecer tres niveles de dificultad, utilizando como limitadores el número de movimientos y el tiempo (además de ir mejorando la inteligencia de la máquina).
  - Puzle:
    - Poner una animación cuando se complete el puzle.
    - Cargar archivos de imágenes para usarlos como puzles.
    - Añadir más puzles.
- Crear tres juegos:
  - Pelotas: mi idea es que sea una especie de hundir la flota pero, lo que haces es esconder pelotas en diferentes lugares, en este juego se usarían las dos pantallas (en la de arriba juega la máquina y en la de abajo el jugador).
  - Parejas: el clásico juego que te muestran las parejas y luego las giran y tienes que buscarlas, con diferentes niveles de dificultad.
  - Pong: juego clásico donde los haya donde se trata el tema de las dos pantallas de una forma más difícil ya que la pelota tiene que atravesar correctamente las pantallas.

5. ENTORNO DE TRABAJO

A continuación expondré datos sobre el sw utilizado y su instalación.

5.1 DeSmuMe

La utilización de un emulador ofrece una mayor flexibilidad para la realización de pruebas del juego, ya que no será necesario volcar el juego a la tarjeta cada vez que se quiera probar, además de la posibilidad de enlazar la ejecución del proyecto desde el entorno de desarrollo con el propio emulador, véase ejemplo a continuación:

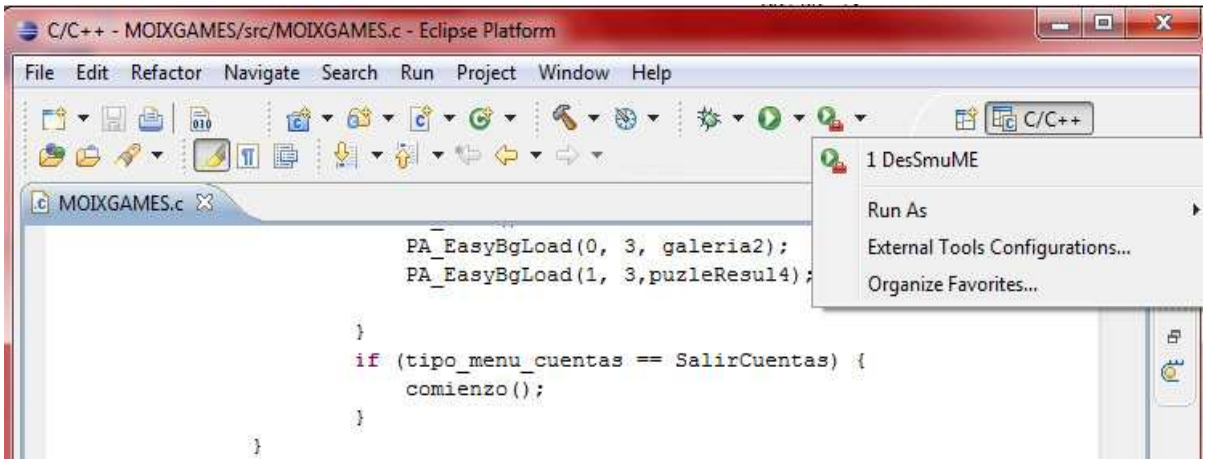


Figura14: Lanzando DeSmuME desde eclipse

Para este proyecto se ha utilizado el emulador DeSmuMe que se puede obtener de forma gratuita desde la página de los desarrolladores [5].

5.2 SDK: devkitPro Y PAlib

Nintendo no ofrece ningún tipo de herramienta para el desarrollo de software para Nintendo DS de forma gratuita, por tanto se ha utilizado devkitPro, un kit de desarrollo que también utilizado para el desarrollo de aplicaciones orientadas a Game Boy Advance, PSP GP32, GameCube o Wii.

5.2.1 Instalación de devkitPro en Windows

Existe un instalador para Windows muy fácil de usar, así que primero se descarga el ejecutable devkitPro Updater 1.5.0. Al ejecutarlo comprueba si existen actualizaciones y da la opción de instalar o descargar los archivos, seleccionando "Instalar" pregunta si se quieren conservar o eliminar los archivos que se descarguen para la instalación (aunque no es una cuestión importante). Seguidamente da a elegir los componentes a instalar, seleccionamos el compilador "DevKitArm" y "Sistema mínimo",

DevKitPPC es un compilador para GameCube y Wii y DevKitPSP para PSP, Programmer's Notepad es un IDE y Insight un debugger, así que no son necesarios ya que vamos a desarrollar para NDS y en otro IDE. A continuación se selecciona la ruta de instalación y se instala el kit de desarrollo.

5.2.2 Instalación de PAlib

Para instalar PAlib tan sólo es necesario descargar la última versión y descomprimir el contenido en /devkitPro/PAlib/ en la ruta en la que se haya instalado el SDK.

Conseguir estas librerías es complicado porque en cuanto los de PAlib abren una pagina se la cierran [6], entonces hay que moverse por muchos foros.

5.2.3 Instalación de devkitPro en Linux

Para instalar DevKitPro en Linux, primero deberemos bajar todo lo necesario: devkitArm (el compilador), libnds (librería principal.), libfat-nds(librería para el manejo ficheros), libfilesystem y Default arm7, y ejemplos .nds.

Una vez los tengas, creamos la carpeta donde se alojará y descomprimimos el compilador, por ejemplo en /opt:

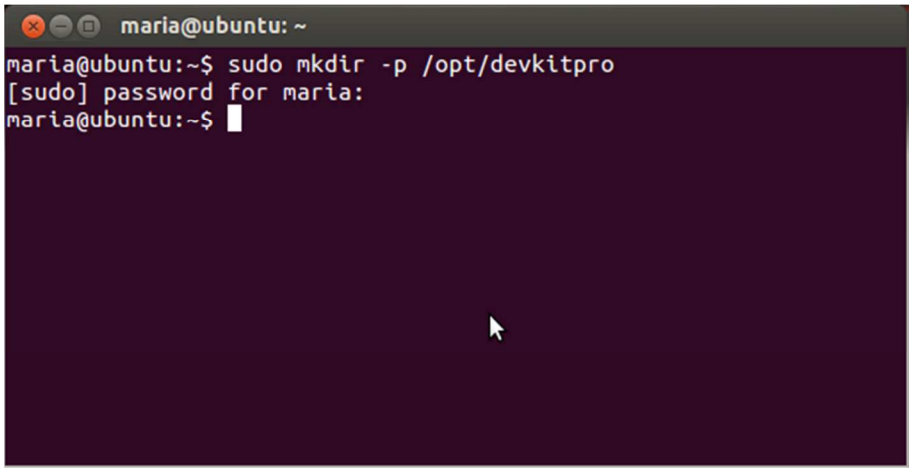


Figura 15: Creación directorio para devkitpro

Tras ello, creamos una carpeta llamada libnds en la ruta del devkitpro y descomprimimos ahí todas las librerías y los datos del ARM7:

Para los ejemplos, creamos una carpeta propia y los descomprimimos en ella:

```
mkdir libnds
cd libnds
tar -xvjf [libnds].tar.bz2
tar -xvjf [libfat-nds].tar.bz2
tar -xvjf [dswifi].tar.bz2
tar -xvjf [maxmod].tar.bz2
tar -xvjf [libfilesystem].tar.bz2
tar -xvjf [default arm7].tar.bz2
cd ..
mkdir -p examples/nds
cd examples/nds
tar -xvjf [ejemplos nds].tar.bz2
```

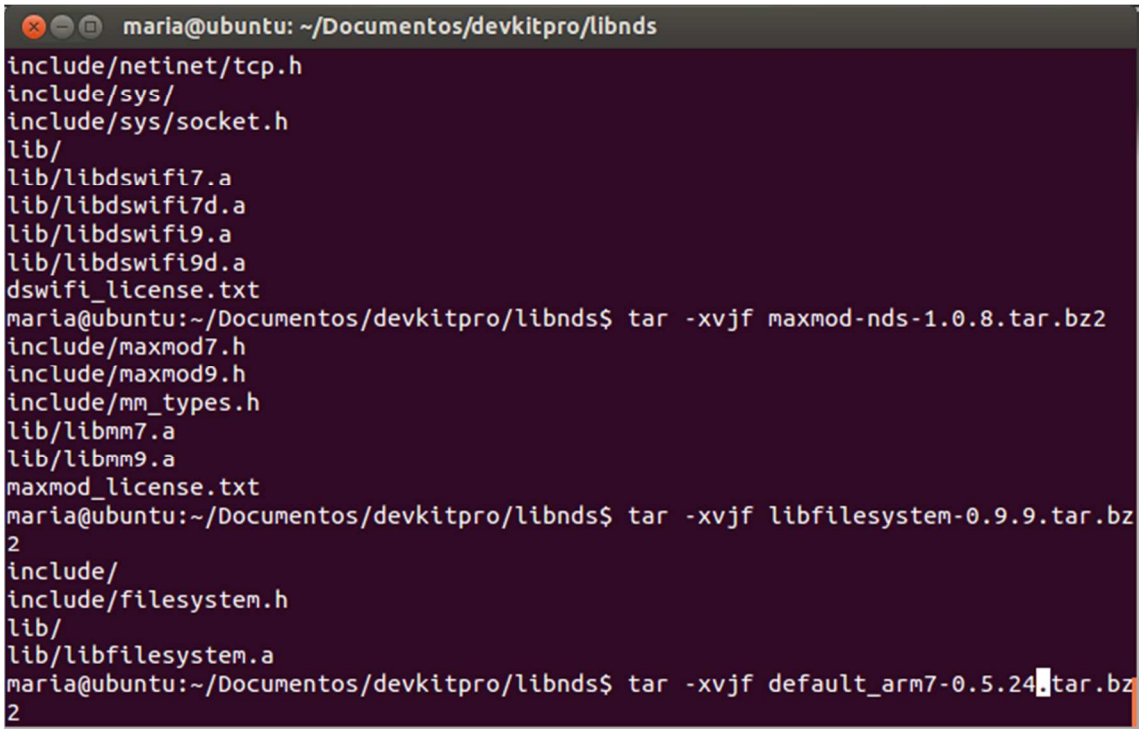


Figura 16: Descomprimir archivos

Por último, hay que añadir las variables al entorno, editando el archivo .bashrc localizado en "home" (puede estar oculto), añadiendo al final las siguientes líneas:

export DEVKITPRO=/opt/devkitpro

export DEVKITARM=$DEVKITPRO/devkitARM

Para comprobar que la instalación se ha realizado correctamente, basta con ir a $DEVKITPRO/examples/nds/nombreDeUno de los ejemplos y ejecutar la orden make.

Si todo va bien, tendremos un .nds.

5.2.4 Instalación de PAlib en Linux

La instalación de PAlib es igual en sistemas Linux y Windows, ya que solo requiere bajarse la última versión y descomprimir el contenido del paquete en /devkitPro/PAlib/.

El paquete PAlib trae además una colección de herramientas útiles. A continuación comentaré PAGfx, herramienta pensada para convertir imágenes a un formato adecuado para usar con PAlib.

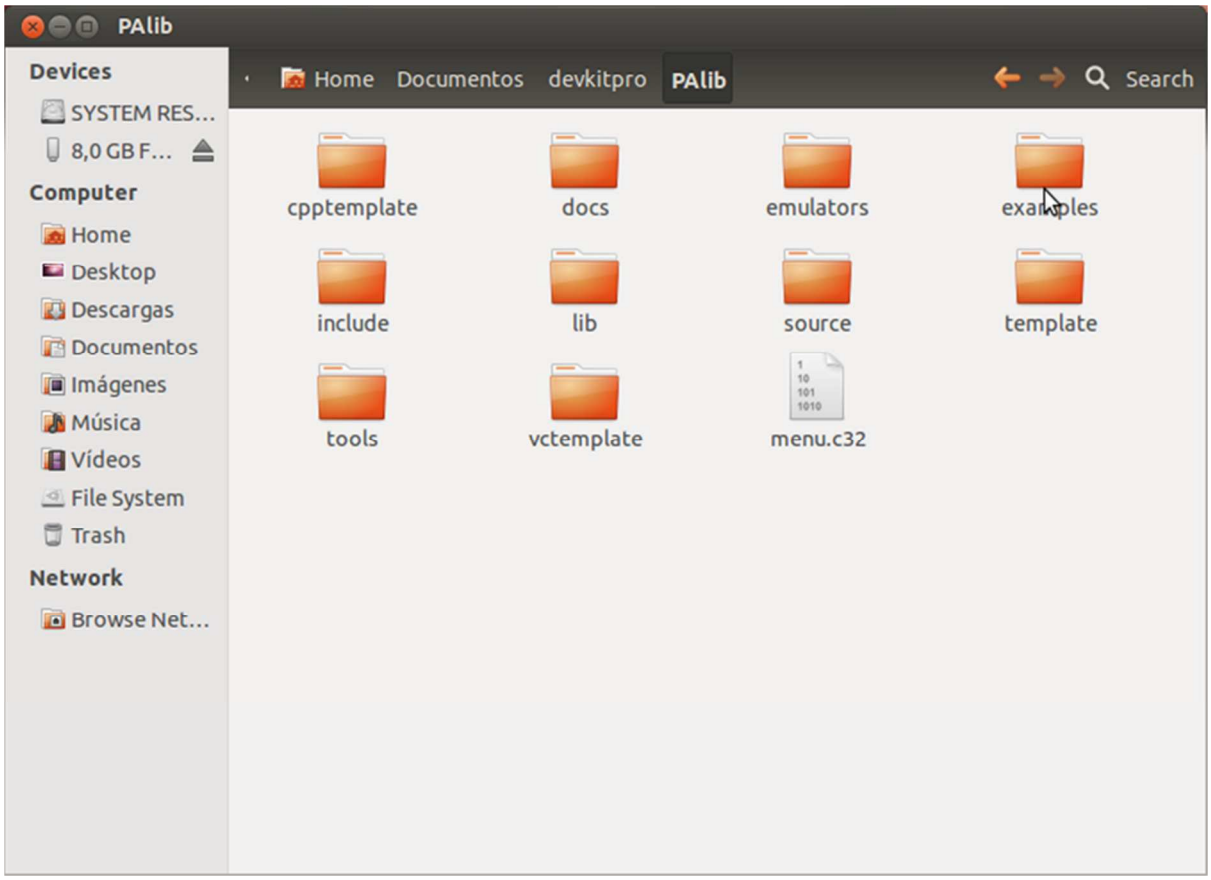


Figura 17: Estructura de los ficheros

5.3 PAGfx

PAGfx es una aplicación que se usa para convertir imágenes a RAW, para que puedan ser utilizadas en el proyecto, permite cargar sprites fondos y texturas, seleccionando el color que representa la transparencia .

Permite elegir el modo de color (256 colores por defecto), al cargar un fondo se puede seleccionar el tipo de fondo (por tiles o una fuente de letra) y para todos los tipos de imágenes se debe definir la paleta de colores.

Al hacer la conversión los ficheros resultantes aparecen en la carpeta bin, éstos incluyen una cabecera; para usarlos en el proyecto se debe mover la carpeta bin a la carpeta gfx de éste e incluir la cabecera en el código.

5.4 ECLIPSE

La razón para elegir Eclipse como entorno de desarrollo es su plugin "NDS ManagedBuilder" que permite construir proyecto de manera cómoda y rápida, éste se puede conseguir junto con el IDE en la página oficial del creador del plugin.

También cabe destacar lo bien que te marca donde están los errores y como detalla el problema.

Otra de las razones es que sirve tanto para Windows como para Linux.

Para plataformas Windows de 32 bits se puede conseguir el IDE con el plugin en la página oficial del creador del plugin. Para otras plataformas, será necesario instalar Eclipse 3.4.0 o superior e instalar el plugin mediante el actualizador del propio Eclipse usando la url de actualizaciones del NDS Managed builder (http://dev.snipah.com/nds/updater).

Para su descarga e instalación en Linux se puede conseguir utilizando el "Centro de SoftWare" de Ubuntu(en caso de utilizar Ubuntu).

Figura 18: Descarga Eclipse

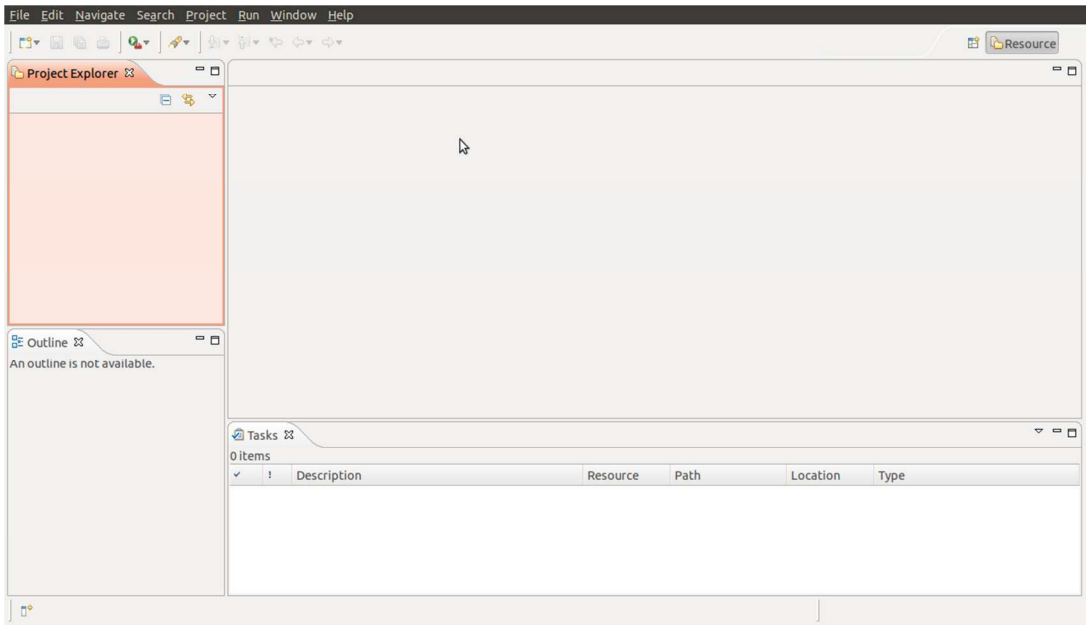


Figura 19: Eclipse en Ubuntu

Seguidamente se configura PAlib para eclipse, abriendo las propiedades del proyecto se accede a Settings en C/C++ build y, en los desplegables de devkitArm C Compiler y devkitArm C++ Compiler se añade "*${DEVKITPRO_DIR}/PAlib/include/nds*". Para enlazar las librerías hay que bajar hasta devkitArm C++ linker y se añade la ruta de búsqueda el directorio "*${DEVKITPRO_DIR}/PAlib/lib*" y en las librerías pa9 y pa7 sin el prefijo "lib" ni la extensión ".s".

Por último se configura el IDE para que ejecute el emulador, para ello se abre la configuración de herramientas externas, en la ventana emergente se añade la ruta del emulador que se quiera usar (en este caso el DeSmuMe), el entorno de trabajo será la carpeta Debug del proyecto "${workspace_loc:/${project_name}/Debug}" y como argumento el archivo generado "${project_name}.nds".

Con esto tenemos el entorno configurado para funcionar fácilmente usando "Build" para construir el proyecto y "Run>External Tools" para ejecutarlo directamente en el emulador.

6. MEJORAS

6.1 Mejoras en tres en raya

En un principio cuando realice el juego para IMD, en el tres en raya no había niveles de dificultad, y la partida terminaba cuando uno de los dos ganaba o cuando se llenaba el tablero .

Figura 18: Tres en raya empate.

Figura 19: Tres en raya Fácil

Figura 20: Tres en raya Medio

Figura 21: Tres en raya Difícil

El primer paso era que en lugar de dibujar hasta que se completara, se dibujaran solo 3 fichas por usuario. Una vez realizado esto había que conseguir mover las fichas, para ello utilizare la función:

PA_SetSpriteXY(pantalla, nombreSprite, cordenadaX, cordenadaY);

Una vez establecidas estas características, hay que proceder a crear los niveles que como ya he dicho en los objetivos previos estaban establecidos por delimitaciones en el número de movimientos y en el tiempo.

El nivel fácil : será el que ya había creado.

En el nivel medio : aumenta la inteligencia de la máquina y se delimitan el número de movimientos. Para esto solo es necesario crear un contador y cuando este valga el número deseado se parará el juego y si ninguno ha hecho línea se considerará un empate.

En el nivel difícil: vamos a contar con una restricción extra en este caso será el tiempo, disponemos de 60s para ganar, si pasado este tiempo nadie a ganado aunque el número de movimientos aun no sea igual a 0, se finaliza la partida y se considera un empate.

Para realizar la restricción del tiempo he hecho lo siguiente:

```
int milésimas,segundosCont;

        milesimas=0;

        segundosCont=60;

        milésimas++;

        if (((milesimas==60) && (segundosCont>0)) && (Bloc==0)){
              segundosCont--;
              PA_OutputText(0,23,7,"seg: %d " ,segundosCont);
              milesimas=0;
        }
```

Figura 17: contador

6.2 Mejoras en puzle

En la implementación del puzle que hice para la asignatura de IMD no había puesto nada que señalizara que se había resuelto un puzle por lo que esto se convirtió en uno de los objetivos que llevar a cabo en este proyecto.

Para poder saber que el puzle estaba resuelto, he creado un array en el que se almacena la que pieza se pinta y donde (ya que no se pintan siempre las mismas 8 piezas , porque hay 9 y con un rand se decide cuales son las que se dibujan), este array se rellenará dentro del bucle:

```
desordenar_puzle(){
        while (fichas_sin_poner > 0) {
                if(puzlOc[posicion] == 0) {
                        if (posicion == 0){
                                pieza=rand() % 9;
                                        if((pieza ==0) && ( cont11==0)){
                                                ...
                                        }
                                ...
                        }
                }
                ...
         casillasPieza[posicion]=pieza;
        }
}
```

Figura 22: Localizador piezas.

A continuación en menuPuzle() :

```
void menuPuzle(){

  PA_Init();
  if (tipo_menu_puzle == ContinuarPuzle){

  intro();
  desordenar_puzle();

        while (tipo_menu_puzle == ContinuarPuzle) {

              if (Stylus.Held) {


                    tipo_menu_puzleDib = obtener_menu_puzleDib(Stylus.X, Stylus.Y);
                          if (tipo_menu_puzleDib == PuDib11 && puzlOc[0] != 0) {
                                if (puzlOc[1] == 0) {
                                      puzlOc[1] = puzlOc[0];
                                      puzlOc[0] = 0;
                                      PA_SetSpriteXY(0, puzlOc[1], 64, 0);

                                      casillasPieza[1]=casillasPieza[0];
                                      casillasPieza[0]=10;
                                }
                                else {
                                      if (puzlOc[3] == 0) {
                                        puzlOc[3] = puzlOc[0];
                                        puzlOc[0] = 0;
                                        PA_SetSpriteXY(0, puzlOc[3], 0, 64);

                                        casillasPieza[3]=casillasPieza[0];
                                        casillasPieza[0]=10;
                                      }
                                }
                          }

                          if((casillasPieza[0]==10 && casillasPieza[1]==1 &&  ***
                          casillasPieza[8]==8) ||  ***
                          (casillasPieza[0]==0 && casillasPieza[1]==1 ***
                          casillasPieza[8]==10) ){
                            PA_CreateSprite(1,cont,(void*)hasGanado1_Sprite,
                                OBJ_SIZE_64X64,1,39,32,64);
                            cont++;
                            PA_CreateSprite(1,cont,(void*)hasGanado2_Sprite,
                                OBJ_SIZE_64X64,1,40,96,64);
                          cont++;
                          PA_CreateSprite(1,cont,(void*)hasGanado3_Sprite,
                                OBJ_SIZE_64X64,1,41,160,64);

                          for(ino=0; ino<120 ; ino++){
                                      PA_WaitForVBL();
                          }
                          PA_Init();
                          intro();
                          desordenar_puzle();

                    }
              ***
        }
```

Figura 23: Ejemplo puzle montado

Para indicar que se ha resuelto el puzle sale un cartel que se compone de 3 Sprites de 64x64 :

Figura 24: Cartel has ganado

El tener que usar tres sprites es debido a las restricciones, de la NDS, ya que esta delimita el tamaño de los sprites, los permitidos son cualquiera de estos tamaños a lo alto y ancho:

	8	16	32	64
8	8×8	16×8	32×8	
16	8×16	16×16	32×16	
32	8×32	16×32	32×32	64×32
64			32×64	64×64

Figura 25: Tamaños sprite

Una vez mostrado el cartel el puzle se reinicia automáticamente.

Otro de los objetivos que me propuse con respecto a este "mini juego" era cargar mas puzles. Para esto he creado una variable global llamada numPuzle que se encarga de cargar un puzle u otro.

He creado tres nuevos puzles , dos de ellos son unas fotos de zapatillas pintadas por mi (con lo que no hay problemas con los derechos de autor).

Se pasa de un puzle a otro dando al botón de siguiente , cuando numPuzle ==4 esta se resetea con lo que vuelve a cargar el puzle1 y así sucesivamente.

6.3 Otras mejoras

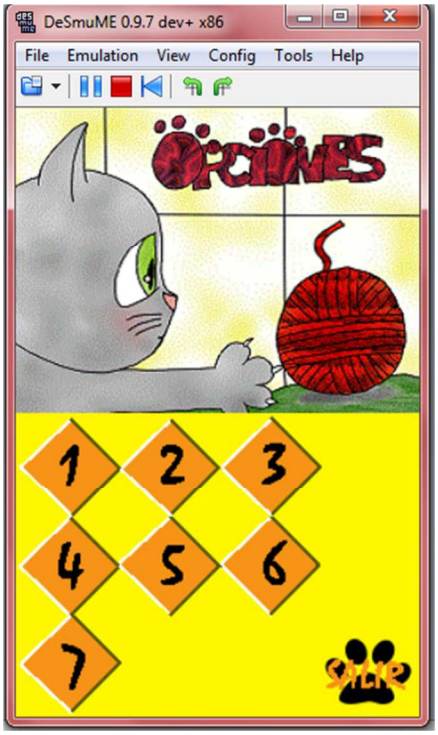
En la galería aunque no había propuesto cambiar nada , al crear imágenes nuevas para los juegos pensé que estaría bien ponerlas en la galería también pero el funcionamiento es el mismo , al hacer click en uno de los números se borra la pantalla y se carga un nuevo fondo en la pantalla 1, como se puede ver en el siguiente fragmento:

```
if (tipo_menu_cuentas == uno) {
        PA_Init();
        PA_EasyBgLoad(0, 3, galeria2);
        PA_EasyBgLoad(1, 3, enraya0);

}
```

Figura 26: Ejemplo carga fondo de galería

Parte del cambo se puede apreciar comparando las siguientes figuras:



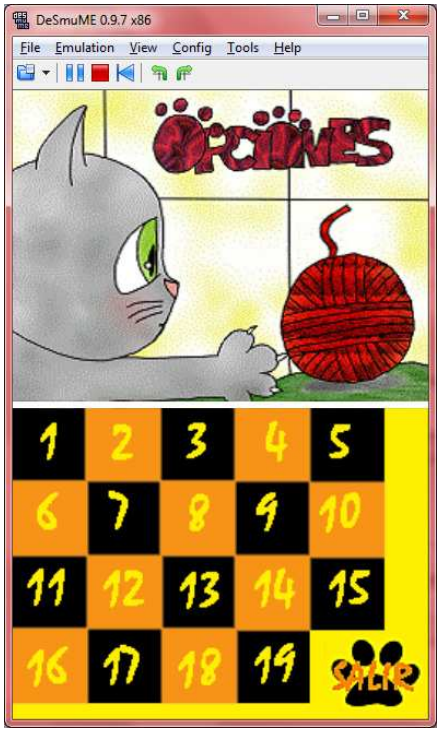


Figura 27: Galería de Imágenes

Figura 28: Galería de imágenes actual

También he cambiado la imagen de cuando arranca el juego, a continuación se puede ver el cambio:

Figura 29: Carga antigua

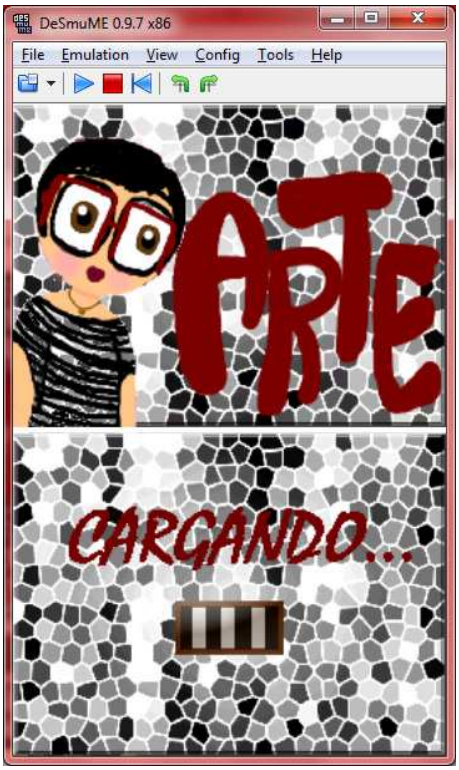


Figura 30: Carga nueva

Cabe mencionar que pone arte, porque es mi nombre de artesana así que vi adecuado ponerlo en la intro, y la muñeca del dibujo me representa a mi.

7. JUEGOS NUEVOS

A continuación expondré información sobre los juegos nuevos añadidos a la aplicación MOIXGAMES.

7.1 Pelotas (Escondite)

Este juego lo he renombrado como escondite , ya que lo que se hace es esconder, esta basado en los clásicos juegos hundir la flota y buscaminas, pero aquí se esconden pelotas.
La idea es que el jugador esconde sus pelotas , la máquina esconde las suyas y luego, la maquina busca las tuyas y tu las suyas.

Para llevar acabo esto en he creado dos arrays (casillasEsc[36] y casillasEscMaquina[36]); cuando el jugador va escondiendo las pelotas , se guarda de la siguiente forma:

casillasEsc[*posición*]=*numQueIdentificaElSprite*;

Si el jugador o la maquina encuentra una pelota se muestra una pelota, sino se muestra un tomate.

En el nivel fácil: se esconden 8 pelotas y tenemos 36 movimientos para encontrarlas, además de esto se puede ver temporalmente donde a puesto las pelotas la máquina; el que antes encuentre las 8 pelotas gana y finaliza la partida, automáticamente se reinicia todo guardando si se ha ganado o perdido y así se ven las puntuaciones.

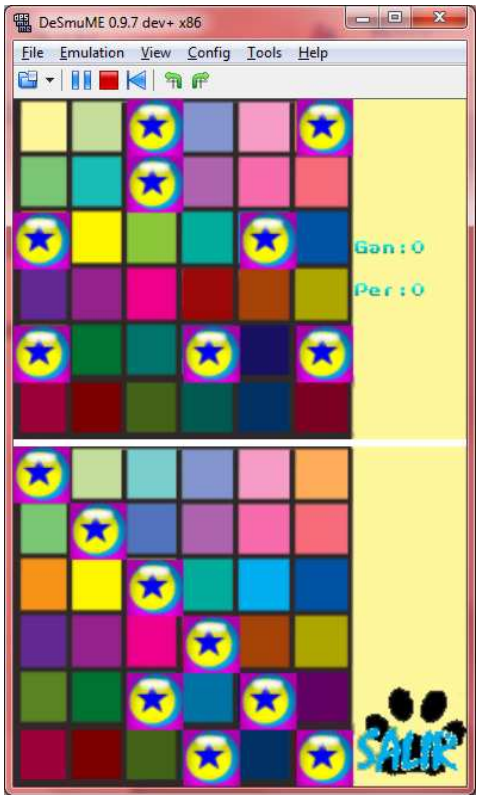


Figura 31: Escondite Fácil pelotas

Figura 32: Escondite Fácil

En el nivel medio: se esconden 4 pelotas y tenemos 25 movimientos para encontrarlas, y ya no se puede ver donde a puesto las pelotas la máquina; el que antes encuentre las 4 pelotas gana y finaliza la partida, automáticamente se reinicia todo guardando si se ha ganado , perdido o empatado .

Figura 33: Escondite Medio

En el nivel difícil: se esconden una pelota y tenemos 20 movimientos para encontrarla, el que antes encuentre la pelota gana y finaliza la partida, automáticamente se reinicia todo guardando si se ha ganado, perdido o empatado .

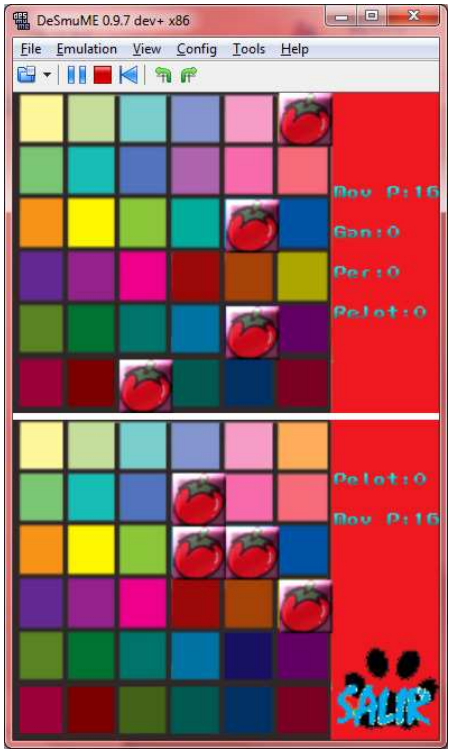


Figura 34: Escondite Difícil

Dependiendo de si se gana se pierde o se empata se muestra, una imagen en la parte de arriba ,y abajo aparece un cartel (nuevamente compuesto por tres sprites de 64x64):

Figura 35:Carteles

Figura 36: Escondite Perdido

Figura 37: Escondite Ganado

Figura 38: Escondite Empatado

7.2 Parejas

Es el clásico juego de las parejas, pero para que resulte un poco mas interesante (y ya que en la pantalla no caben muchas "cartas", porque hay que tener en cuenta las restricciones del tamaño de los sprites, y no quería que fueran imágenes muy pequeñas, dado que mi idea es que la aplicación pueda ser usada por gente de cualquier edad), las cartas no se giran al principio porque 8 cartas se memorizan enseguida, sino que se giran solo cuando haces click en ellas y se vuelven a girar si no has hecho pareja, originariamente el juego tenía 8 cartas, esto se ha mantenido para el nivel fácil:

Figura 39: Parejas Fácil1

Figura 40: Parejas Fácil2

Figura 41: Parejas Fácil resuelto

En el nivel medio: ya hay 10 cartas, y el funcionamiento es el mismo que en el fácil:

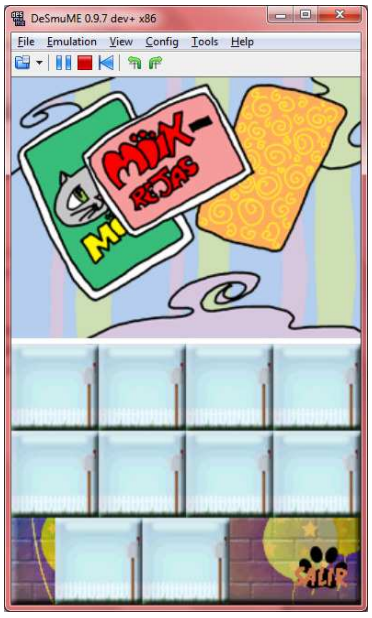


Figura 42: Parejas Medio1

Figura 43: Parejas Medio2

Figura 44: Parejas Medio resuelto

En el nivel difícil :hay 10 cartas, el funcionamiento es igual que el de los otros, pero con la restricción añadida de que hay limite de clicks:.

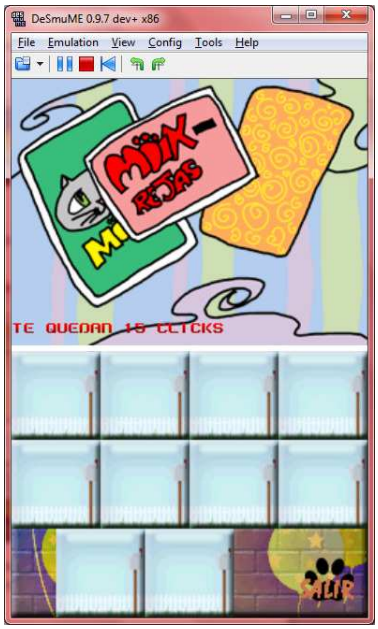


Figura 45: Parejas Dificil1

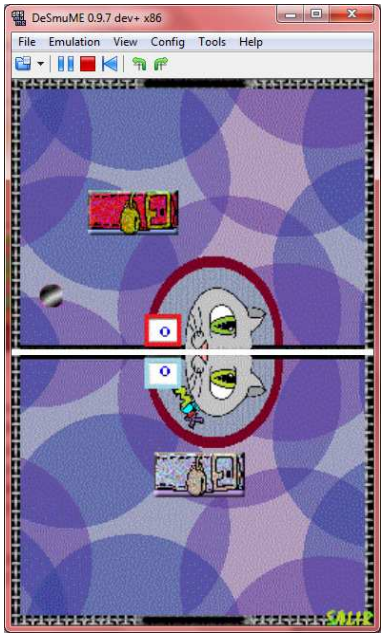
Figura 46: Parejas Difícil 2

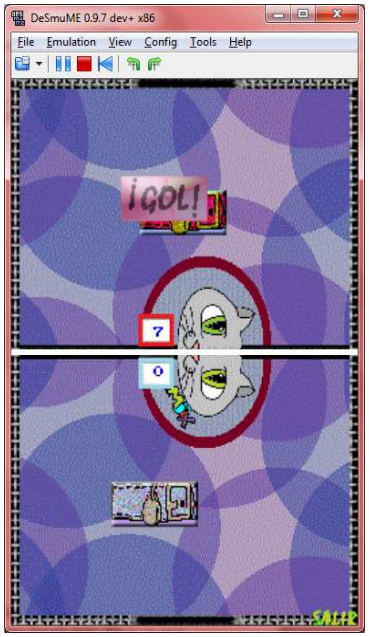
Figura 47: Parejas Difícil resuelto

7.2 Pong (Poinx)

El ultimo juego es el pong, en este caso renombrado como poinx, en esta versión el usuario mueve un collar de gato para dar a la pelota, este se desplaza con el stylus, con lo que s se puede mover libremente por la pantalla, mientras que la maquina al principio esta arriba y se desplaza al centro de la pantalla 1 cada vez que la pelota entra en la pantalla 0, el movimiento de la maquina es horizontal. En este "mini juego" también distinguimos tres niveles:

En el nivel fácil: la pelota se desplaza despacio y las dos porterías se encuentran en las coordenadas X 92 y 167.





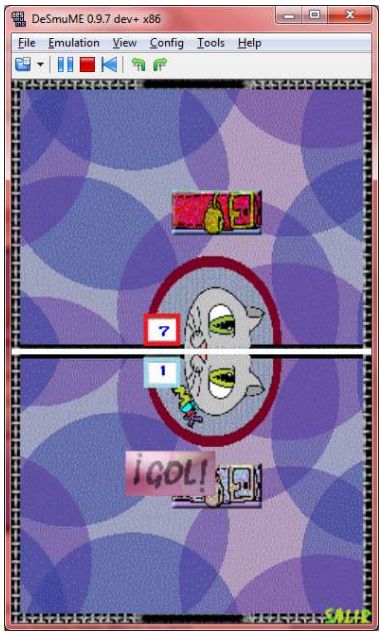


Figura 48: Poinx fácil *Figura 49:Poinx fácil gol jugador* *Figura 50:Poinx fácil gol máquina*

En el nivel medio: la pelota va el doble de rápido y la portería del jugador es mas grande, comprendiendo en este caso entre x>=79 y x<=191.

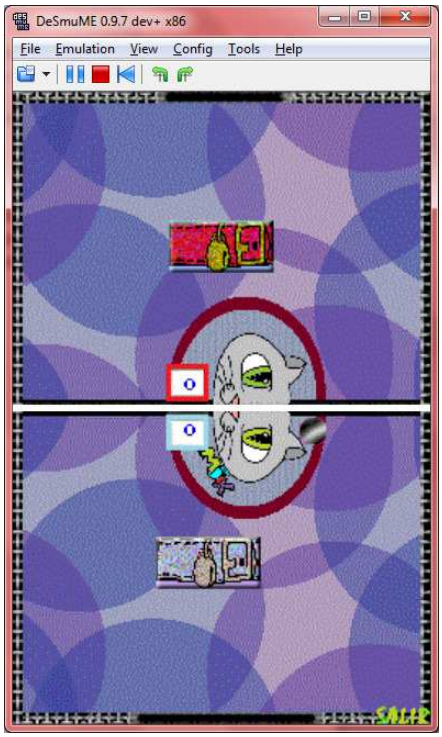


Figura 51:Poinx Medio

En el nivel difícil: la pelota va aun más rápido y la portería del jugador es aun mas grande, con x entre 14 y 218.

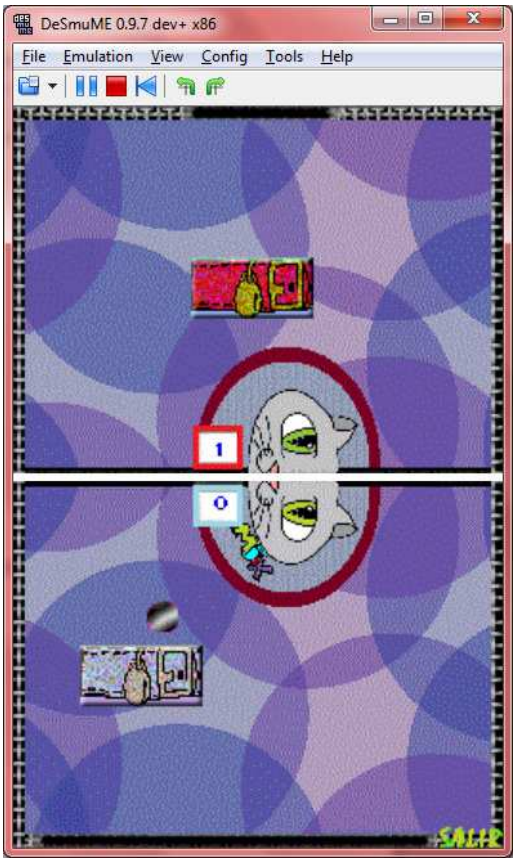


Figura 52:Poinx Difícil

8. IMPLEMENTACIÓN

A continuación voy a ha analizar la implementación de los diferentes partes del juego:

Lo primero que hay que hacer es incluir las librerías necesarias, y a continuación se han de hacer todas las declaraciones de "#define", que vayan a utilizar los métodos obtener, en total han sido necesarios 143 "#define:

```
#include <nds.h>
#include <stdio.h>
#include<pa9.h>
#include"gfx/all_gfx.h"
#include"gfx/all_gfx.c"
```

```
//MENU PRINCIPAL
#define JUEGOS 1
#define CREDITOS 2
#define CUENTAS 3
//MENU JUEGOS
#define TresEnRaya 4
```

El siguiente paso es declarar las variables globales, tales como los contadores, etc.:

```
//declaraciones
u32 tresOc [9];
u32 puzlOc[9];
u32 localCarta[8];//cartas bocaabajo
u32 CartaArriba[8];//cartas bocaabajo
u32 pintCarta[8];//para comprobar que ya se ha pintado una carta en esa pos
u32 cartaGira[2];//las dos cartas boca arriba
int ContG;
int ContP;
//parejas med y dif
u32 localCartaMD[10];//cartas bocaabajo
u32 CartaArribaMD[10];//cartas bocaabajo
u32 pintCartaMD[10];//para comprobar que ya se ha pintado una carta en esa pos
int contMovPar;
int cont4;
int cont1;
int cont5;
int cont2;
int cont6;
```

Posteriormente se declaran los métodos obtener, véase ejemplo:

```
//Menu 3 en raya
u8 obtener_menu_parejas(s32 xStylus, s32 yStylus);
u8 tipo_menu_subParejas= SalirSubParejas;
u8 obtener_menu_subParejas(s32 xStylus, s32 yStylus);
u8 tipo_menu_subParejasMD= SalirSubParejas;
u8 obtener_menu_subParejasMD(s32 xStylus, s32 yStylus);
//Tipo menu creditos
u8 tipo_menu_creditos= SalirCreditos;
//Menu creditos
u8 obtener_menu_creditos(s32 xStylus, s32 yStylus);
```

Estos se utilizaran para recoger los datos de la pantalla táctil como ya explicaré a posteriori.

Lo siguiente es ha analizar la implementación de los diferentes métodos del juego.

8.1 main ()

```
int main (void) {

	PA_Init();
	PA_InitVBL();

	int i;

	PA_LoadSpritePal(0,12,(void*)carg10_Pal);
	PA_LoadSpritePal(0,13,(void*)carg11_Pal);

	PA_EasyBgLoad(1, 3, arte);
	PA_EasyBgLoad(0, 3, carga);

	PA_CreateSprite(0,cont,(void*)carg10_Sprite, OBJ_SIZE_64X32,1,12,96,100);

	bool noSal = 1;
		while (noSal) {
			noSal = (Stylus.Held || Pad.Newpress.Start);
				for(i=0; i<30 ; i++){
					PA_WaitForVBL();
				}
			PA_CreateSprite(0,cont,(void*)carg11_Sprite,
OBJ_SIZE_64X32,1,13,96,100);
				for(i=0; i<30 ; i++){
					PA_WaitForVBL();
				}
			PA_CreateSprite(0,cont,(void*)carg10_Sprite,
OBJ_SIZE_64X32,1,12,96,100);
				for(i=0; i<30 ; i++){
					PA_WaitForVBL();
				}
			PA_CreateSprite(0,cont,(void*)carg11_Sprite,
OBJ_SIZE_64X32,1,13,96,100);
				for(i=0; i<30 ; i++){
					PA_WaitForVBL();
				}
			PA_CreateSprite(0,cont,(void*)carg10_Sprite,
OBJ_SIZE_64X32,1,12,96,100);
				for(i=0; i<30 ; i++){
					PA_WaitForVBL();
				}
			PA_CreateSprite(0,cont,(void*)carg11_Sprite,
OBJ_SIZE_64X32,1,13,96,100);
				for(i=0; i<30 ; i++){
					PA_WaitForVBL();
				}
		}
		PA_Init();

		PA_EasyBgLoad(1, 3, moix);
		PA_EasyBgLoad(0, 3, tocaIni);
```

```
while (1) {
		if (Stylus.Held) {
			a_jugar = obtener_a_jugar(Stylus.X, Stylus.Y);
			comienzo();
		}
		PA_WaitForVBL();
	}
}//fin main
```

PA_Init (): Esto es para que se inicien las palib con todas sus funciones, con esta función hacemos un reset de la pantalla se borra todo y podemos cargar otras imágenes.

PA_InitVBL (): Inicia las rutinas necesarias para que la DS realice ciertas tareas, tales como obtener la posición del stylus.

PA_LoadSpritePal(Pantalla , numero a la paleta(entre 0 y 15), (void*)sprite_Pal): se hace para cargar la paleta que usará el sprite, en este caso son las paletas de el icono de carga, cargar la paleta es necesario hacerlo antes de crear el sprite, ya que sino el esprite no aparece en la pantalla o sale dibujado mal.

Figura 53: imagen sin paleta y con paleta

PA_EasyBgLoad(u8 screen, u8 bg, PA_BgStruct img):* son, primero la pantalla en la que cargamos el fondo (1 pantalla superior y 0 pantalla inferior), el fondo de la pantalla donde va nuestra imagen (entre 0 y 3) y por último, el puntero a la estructura del fondo, creado con PAGfx.

PA_CreateSprite(pantalla, número identificador del sprite(entre 0 y 127), (void*) nombreSprite, OBJ_SIZE_16X16, numero colores(puede ser 0 ->16 colores o 1->256 colores), numeroDePaleta,coordenada coordenada y): esta función es utilizada, para crear los sprites.

Los bucles for los he utilizado para hacer que se mantenga 30 iteraciones cada imagen, de este modo se consigue crear la sensación de movimiento, para llevar a cabo esto hago uso también de la función PA_WaitForVBL,

PA_WaitForVBL(); Sirve para sincronizar el juego con la velocidad de dibujar la pantalla, con esto esperamos a que acabe de dibujar la pantalla. Sucede cada 16,5 milisegundos.

Tras hacer esto, nuevamente hacemos el PA_Init() y cargamos dos nuevos fondos.

Y para finalizar, creamos un bucle while(1), en el que introducimos un if en el cual si pulsamos pantalla llama a obtener_a_jugar(cordenadaX del Stylus, coordenadaY del Stylus) y se llama al método comienzo().

Stylus.Held : esto quiere decir que el Stylus está pulsando la pantalla.

8.2 comienzo()

```
//COMIENZO
int comienzo(){

	PA_Init();
	PA_EasyBgLoad(1, 3, menuprincipal2);
	PA_EasyBgLoad(0, 3, menuprincipal3);

	bool noSalir = 1;

	while (noSalir) {
		noSalir = (Stylus.Held || Pad.Newpress.Start);
		PA_WaitForVBL();
	}

	while (1) {
		if (Stylus.Held) {
			tipo_menu_prin = obtener_menu_principal(Stylus.X, Stylus.Y);
				if (tipo_menu_prin > 0 && tipo_menu_prin<4) {
					menuPrincipal();
				}
		}
		PA_WaitForVBL();
	}
```

Este método se parece bastante al anterior porque este es un método lanzadera por así decirlo, es el encargado de llamar al menuPricipal.

Comenzamos vaciando la pantalla y cargando dos nuevos fondos, que son el menú principal.

Volvemos a crear las dos while, y en el segundo cuando pulsemos con el stylus llamamos a menuPrincipal, tipo_menu_prin<4 , quiere decir que llamaremos a menuPrincipal si tipo_menu_prin es un define cuyo valor es menor a 4, este valor se obtiene haciendo el obtener_menu_principal(X,Y), que dependiendo de donde pulsemos en la pantalla nos devolverá un número u otro.

8.3 menuPrincipal()

Este método ya tiene un poco mas de código, lo voy a dividir en 3 partes:

```
//MENU PRINCIPAL
void menuPrincipal(){
        PA_Init();
        //JUGAR
        if (tipo_menu_prin == JUEGOS){

                PA_EasyBgLoad(1, 3, juegos2);
                PA_EasyBgLoad(0, 3, juegos4f);

                bool noSalJu = 1;
                while (noSalJu) {
                        noSalJu = (Stylus.Held || Pad.Newpress.Start);
                        PA_WaitForVBL();
                }

   while (tipo_menu_prin == JUEGOS){
        if (Stylus.Held) {
                tipo_menu_juegos = obtener_menu_juegos(Stylus.X, Stylus.Y);
        if (tipo_menu_prin == JUEGOS && tipo_menu_juegos > 3) {
                menuJuegos();
           }
        }PA_WaitForVBL();}
 }//fin jugar
```

Comenzamos vaciando la pantalla como siempre, luego si tipo_menu_prin es JUEGOS cargamos los fondos del menú de juegos.

Creamos un bucle que haga que mientras tipo_menu_prin sea JUEGOS, si pulsamos con el stylus , llamamos a obtener tipo menú juegos.

Si tipo_menu_prin es igual a JUEGOS y tipo_menu_juegos es mayor que 3, llamamos a menuJuegos con el tipo obtenido.

Continuamos con los créditos:

```
//CREDITOS
if (tipo_menu_prin == CREDITOS){

        PA_EasyBgLoad(1, 3, menuprincipal2);
        PA_EasyBgLoad(0, 3, creditos1);
        bool noSalCre = 1;

while (noSalCre) {
        noSalCre = (Stylus.Held || Pad.Newpress.Start);
        PA_WaitForVBL();
}

while (tipo_menu_prin == CREDITOS) {
        if (Stylus.Held) {
                tipo_menu_creditos = obtener_menu_creditos(Stylus.X, Stylus.Y);
                if (tipo_menu_prin == CREDITOS && tipo_menu_creditos ==
SalirCreditos) {
                        comienzo();
                }
        }
        PA_WaitForVBL();
        }
}//finCreditos
```

Cargamos los fondos y creamos el bucle de si tipo_menu_prin igual a CREDITOS, si tocamos la pantalla llamamos a obtener_menu_creditos y si tipo_menu_prin es igual a CREDITOS y tipo_menu_creditos es SalirCreditos, llamamos al método comienzo (ósea salimos de aquí).

Ahora continuare con el último que es cuentas, aunque en verdad es la galería.

```
        //CUENTAS
if (tipo_menu_prin == CUENTAS) {

        PA_EasyBgLoad(1, 3, opciones2);
        PA_EasyBgLoad(0, 3, galeria2);
                bool noSalCue = 1;
                while (noSalCue) {
                        noSalCue = (Stylus.Held || Pad.Newpress.Start);
        PA_WaitForVBL();
                }

                while (tipo_menu_prin == CUENTAS) {
                        if (Stylus.Held) {
                        tipo_menu_cuentas = obtener_menu_cuentas(Stylus.X, Stylus.Y);
                                if (tipo_menu_prin == CUENTAS && tipo_menu_cuentas > 38){

                                menuCuentas();

                                }
                        }
                        PA_WaitForVBL();
        }
        }//fin cuentas
}//fin menu principal
```

Cargamos los fondos y creamos el bucle de si tipo_menu_prin igual a CUENTAS, si tocamos la pantalla llamamos a obtener_menu_cuentas y si tipo_menu_prin es igual a CUENTAS y tipo_menu_ccuentas mayor que 38 llamamos a menuCuentas.

8.4 menuCuentas()

```
void menuCuentas(){
    //CONTINUAR
        if (tipo_menu_prin == CUENTAS){
            //GALERIA
            while (tipo_menu_prin == CUENTAS) {
                if (Stylus.Held) {
                    tipo_menu_cuentas = obtener_menu_cuentas(Stylus.X, Stylus.Y);
                        if (tipo_menu_cuentas == uno) {
                            PA_Init();
                            PA_EasyBgLoad(0, 3, galeria2);
                            PA_EasyBgLoad(1, 3, enraya0);
                        }
                        if (tipo_menu_cuentas == dos) {
                        if (tipo_menu_cuentas == tres) {
                        if (tipo_menu_cuentas == cuatro) {
                        if (tipo_menu_cuentas == cinco) {
                        if (tipo_menu_cuentas == seis) {
                        if (tipo_menu_cuentas == siete) {
                        if (tipo_menu_cuentas == ocho) {
                        if (tipo_menu_cuentas == nueve) {
                        if (tipo_menu_cuentas == diez) {
                        if (tipo_menu_cuentas == once) {
                        if (tipo_menu_cuentas == doce) {
                        if (tipo_menu_cuentas == trece) {
                        if (tipo_menu_cuentas == catorce) {
                        if (tipo_menu_cuentas == quince) {
                        if (tipo_menu_cuentas == dieciseis) {
                        if (tipo_menu_cuentas == diecisiete) {
                        if (tipo_menu_cuentas == dieciocho) {
                        if (tipo_menu_cuentas == diecinueve) {
                        if (tipo_menu_cuentas == SalirCuentas) {
                            comienzo();
                        }
                }

            }

        }
}
```

Si tipo_menu_prin es CUENTAS, mientras que tipo_menu_prin es CUENTAS, si pulsamos con el stylus, llamamos a obtener_menu_cuentas para, hay 19 casos posibles como se ve en la imagen, en cada caso se hace lo mismo se vacía la pantalla y cargamos los nuevos fondos.

8.5 menuJuegos()

De este menú vamos a tres en raya, puzles y a menuJuegosDos:

```
//MENU JUEGOS
void menuJuegos(){
    //3 EN RAYA

    if (tipo_menu_juegos == TresEnRaya){

        PA_EasyBgLoad(1, 3, enraya0);
        PA_EasyBgLoad(0, 3, enrayaNiv);

        bool noSalRAY = 1;

        while (noSalRAY) {
            noSalRAY = (Stylus.Held || Pad.Newpress.Start);
            PA_WaitForVBL();
        }

            while (tipo_menu_juegos == TresEnRaya) {
                    if (Stylus.Held) {
                            tipo_niv_raya = obtener_niv_raya(Stylus.X, Stylus.Y);
                                if (tipo_menu_juegos == TresEnRaya && tipo_niv_raya > 49) {
                                    menuRaya();
                                }
                    }
            PA_WaitForVBL();
            }
    }//fin jugar
```

Si tipo_menu_juegos es igual a TresEnRaya, cargamos los dos fondos nuevos, y llamamos a obtener_niv_raya, si tipo_menu_juegos es TresEnRaya y tipo_niv_raya mayor que 49 (cualquier valor inferior no se coge así es como conseguimos delimitar los pasos de una pantalla a otra), llamamos a menuRaya.

```
  //PUZZLES
  if (tipo_menu_juegos == Puzle){

        PA_EasyBgLoad(1, 3, puzzles2);
        PA_EasyBgLoad(0, 3, puzzles41);

        bool noSalPUZ = 1;

        while (noSalPUZ) {
            noSalPUZ = (Stylus.Held || Pad.Newpress.Start);
            PA_WaitForVBL();
        }

        while (tipo_menu_juegos == Puzle) {
                if (Stylus.Held) {
                        tipo_menu_puzle = obtener_menu_puzle(Stylus.X, Stylus.Y);
                            if (tipo_menu_juegos == Puzle && tipo_menu_puzle > 8) {
                                numPuzle=0;
                                menuPuzle();
                            }
                }
        PA_WaitForVBL();
        }
//fin jugar
```

Si tipo_menu_juegos es igual a Puzle, cargamos los dos fondos nuevos, y llamamos a obtener_menu_puzle, si tipo_menu_juegos es Puzle y tipo_menu_puzle mayor que 8 (cualquier valor inferior no se coge), ponemos la variable numPuzle a 0 para empezar por el puzle 0, cabe mencionar que numPuzle esta definido como variable global.

La última parte del método menuJuegos es la siguiente:

```
    if (tipo_menu_juegos == MenuDos){

        PA_EasyBgLoad(0, 3,juegos4);

        while(tipo_menu_juegos==MenuDos){
            if (Stylus.Held) {
                tipo_menu_juegos_dos = obtener_menu_juegos_dos(Stylus.X, Stylus.Y);
                    if (tipo_menu_juegos == MenuDos && tipo_menu_juegos_dos > 54) {

                        menuJuegosDos();

                    }
            }
            PA_WaitForVBL();
        }
    }//fin jugar

    //SALIR
    if (tipo_menu_juegos == Salir) {
        comienzo();
    }
}//fin menu juegos
```

Si tipo menuJuegos es igual MenuDos, cargamos el nuevo fondo en la pantalla 0 (ósea la inferior), y mientras tipo_menu_juegos sea MenuDos,si pulso con el stylus llamamos a obtener_menu_juegos_dos, si tipo_menu_juegos es MenuDos y tipo_menu_juegos_dos es mayor que 54, llamamos a menuJuegosDos que es el encargado de llamar a escondite, parejas y poinx.

8.6 menuJuegosDos()

```
void menuJuegosDos(){
    //Escondite
    PA_EasyBgLoad(1, 3, enraya0);
    PA_EasyBgLoad(0, 3,juegos4);

    if (tipo_menu_juegos_dos == Escondite){

        PA_EasyBgLoad(1, 3, escondit);
        PA_EasyBgLoad(0, 3, esconditeNiv);

        bool noSalES = 1;

        while (noSalES) {
            noSalES = (Stylus.Held || Pad.Newpress.Start);
            PA_WaitForVBL();
        }

        while (tipo_menu_juegos_dos == Escondite) {
            if (Stylus.Held) {
                tipo_niv_ESC = obtener_niv_ESC(Stylus.X, Stylus.Y);
                    if (tipo_menu_juegos_dos == Escondite && tipo_niv_ESC > 118) {
                        menuESCbase();
                    }
            }
            PA_WaitForVBL();
        }
    }//fin jugar
```

Tras todo lo explicado anteriormente al ver este ejemplo de código nos damos cuenta de que siempre seguimos la misma estructura en cuanto a los menús.

No aparece ninguna función de PA_lib que no haya aparecido antes y no tiene excesiva complejidad.

```
//Poinx
if (tipo_menu_juegos_dos == Poinx){

    PA_EasyBgLoad(1, 3, poinx);
    PA_EasyBgLoad(0, 3, poinxNiv);

    bool noSalPO = 1;

    while (noSalPO) {
        noSalPO = (Stylus.Held || Pad.Newpress.Start);
        PA_WaitForVBL();
    }

    while (tipo_menu_juegos_dos == Poinx) {
        if (Stylus.Held) {
            tipo_niv_POI= obtener_niv_POI(Stylus.X, Stylus.Y);
                if (tipo_menu_juegos_dos == Poinx && tipo_niv_POI > 122) {
                    menuPOIbase();
                }
        }
        PA_WaitForVBL();
    }
fin jugar
```

```
//parejas
if (tipo_menu_juegos_dos == Parejas){

    PA_EasyBgLoad(1, 3, parejas);
    PA_EasyBgLoad(0, 3, parejasNiv);

    bool noSalPA = 1;

    while (noSalPA) {
        noSalPA = (Stylus.Held || Pad.Newpress.Start);
        PA_WaitForVBL();
    }

    while (tipo_menu_juegos_dos == Parejas) {
        if (Stylus.Held) {
            tipo_niv_PAR= obtener_niv_PAR(Stylus.X, Stylus.Y);
                if (tipo_menu_juegos_dos == Parejas && tipo_niv_PAR > 126) {
                    menuPARbase();
                }
        }
        PA_WaitForVBL();
    }
fin jugar

if (tipo_menu_juegos_dos == Anterior) {

    menuPrincipal();
}
```

Al comparar los tres if principales de este método podeos apreciar que el comportamiento es siempre el mismo.

Con lo que una vez te das cuenta de esto la complejidad a la hora de crear menús desaparece.

8.7 menuESCbase()

```
void menuESCbase(){

    if (tipo_niv_ESC == ESCFac){

        PA_EasyBgLoad(1, 3, escondit);
        PA_EasyBgLoad(0, 3, escondit3);

        bool noSalESC = 1;

                while (noSalESC) {
                    noSalESC = (Stylus.Held || Pad.Newpress.Start);
                    PA_WaitForVBL();
                }
                    while (tipo_niv_ESC == ESCFac) {
                            if (Stylus.Held) {
                                    tipo_menu_ESC = obtener_menu_escondite(Stylus.X, Stylus.Y);
                                        if (tipo_niv_ESC == ESCFac && tipo_menu_ESC > 59) {

                                            contPuestas=0;
                                            contEmpEsc=0;
                                            contGanEsc=0;
                                            contPerEsc=0;
                                            menuESCFac();
                                        }
                            }
                    PA_WaitForVBL();
                    }

            }//fin raya fac
```

```
//raya med
if (tipo_niv_ESC == ESCMed){

    PA_EasyBgLoad(1, 3, escondit);
    PA_EasyBgLoad(0, 3, escondit3);

    bool noSalESC2 = 1;

            while (noSalESC2) {
                noSalESC2 = (Stylus.Held || Pad.Newpress.Start);
                PA_WaitForVBL();
            }

                while (tipo_niv_ESC == ESCMed) {
                        if (Stylus.Held) {
                                tipo_menu_ESC = obtener_menu_escondite(Stylus.X, Stylus.Y);
                                    if (tipo_niv_ESC == ESCMed && tipo_menu_ESC > 59) {

                                        contPuestas=0;
                                        contEmpEsc=0;
                                        contGanEsc=0;
                                        contPerEsc=0;
                                        menuESCMed();
                                    }
                        }
                PA_WaitForVBL();
                }

        }//fin med
```

```
if (tipo_niv_ESC == ESCDif){

    PA_EasyBgLoad(1, 3, escondit);
    PA_EasyBgLoad(0, 3, escondit3);

    bool noSalESC3 = 1;

            while (noSalESC3) {
                noSalESC3 = (Stylus.Held || Pad.Newpress.Start);
                PA_WaitForVBL();
            }

                while (tipo_niv_ESC == ESCDif) {
                        if (Stylus.Held) {
                                tipo_menu_ESC = obtener_menu_escondite(Stylus.X, Stylus.Y);
                                    if (tipo_niv_ESC == ESCDif && tipo_menu_ESC > 59) {

                                        contPuestas=0;
                                        contEmpEsc=0;
                                        contGanEsc=0;
                                        contPerEsc=0;
                                        menuESCDif();
                                    }
                        }
                PA_WaitForVBL();
                }

        }//fin raya dif
if (tipo_niv_ESC== SalirESC2) {
    menuPrincipal();
}
```

Es como el método anterior no añade nada nuevo, salvo que aquí se ve como se resetea el valor de las variables.

Este es el método que llamará a un nivel del juego escondite dependiendo de en que zona de la pantalla pulsemos.

8.8 menuESCFac()

```
void menuESCFac(){
    PA_Init();
    //CARGAMOS LAS PALETAS COMO YA HE DICHO ANTERIORMENTE
    if (tipo_menu_ESC== ContinuarESC){

        PA_EasyBgLoad(1, 3, esconfond32Maquina);
        PA_EasyBgLoad(0, 3, esconfond32usu);
        int jugBu,maqBu;
        jugBu=0;
        maqBu=0;
        int ran;
        ran=0;
        int cuentaTomates;
        cuentaTomates=0;
        int cartel;
        cartel=124;
        int valCas;
        for (valCas=0;valCas<36;valCas++){
            casillasEsc[valCas]=0;
        }
        for (valCas=0;valCas<36;valCas++){
            casillasEscMaquina[valCas]=0;
        }
        cont=1;
        contMaqEsc=1;
        bool noSalescc = 1;
        int noPulsa;
        noPulsa=0;
        int borra;
        borra=0;
        int contJ,contM;
         int movJ;
         movJ=36;
         int movM;
         movM=36;
        PA_LoadDefaultText(0,2);
        PA_SetTextTileCol(0,5);
        PA_OutputText(0,1,12, "ESCONDE LAS 8 PELOTAS");
```

```
PA_LoadDefaultText(1,2);
PA_SetTextTileCol(1,5);
PA_OutputText(1,24,10,"Gan:%d " ,contGanEsc );
PA_OutputText(1,24,13,"Per:%d " ,contPerEsc );

int FIN;
for (FIN = 0; FIN  < 120; FIN++){

    PA_WaitForVBL();
}
PA_ClearTextBg(0);

while (noSalescc) {
noSalescc = (Stylus.Held || Pad.Newpress.Start);
PA_WaitForVBL();
}
 //tomates jug y tomates Maq
 int tomatesM;
 tomatesM=0;
 int tomatesJ;
 tomatesJ=0;
 int textM;
gana
 int ganaMac;
 ganaMac=0;
 int ganaJug;
 ganaJug=0;


    int noRepite;
    noRepite=0;

    int finalJ;
    finalJ=0;
    int final;
    final=0;
```

En este caso, el ya no es un menú, sino que es la versión fácil del juego del escondite.

En los dos primeros fragmentos de código podemos ver que el orden del desarrollo es siempre el mismo, PA_Init(), comprobar donde estamos, cargar fondo, etc.

Pero vemos alguna función que no e explicado anteriormente.

PA_LoadDefaultText(u8 screen, int background): con esta función, se carga en el fondo background (valores entre 0 y 3) de la pantalla screen (por defecto, con valor 0 se indica la pantalla inferior y 1 la superior) los bitmaps de la fuente por defecto con la que escribir el texto.

PA_SetTextTileCol (u8 screen, u8 color);
Permite cambiar el color de la fuente, el primer argumento indica la fuente de qué pantalla y el segundo el color. Se pueden poner los colores en textoTEXT_WHITE, TEXT_RED, etc..., o con números como en el ejemplo anterior.

```
//CONTINUAR esC
while (tipo_menu_ESC == ContinuarESC) {

    if (Stylus.Held  ) {

        tipo_menu_subEsc = obtener_menu_subEsc(Stylus.X, Stylus.Y);
        if(noPulsa==0){

        if(cuentaTomates<9){
            finalJ=0;
            if(finalJ==0){

                if(tipo_menu_subEsc==casEsc1 && casillasEsc[0]==0){
                    PA_CreateSprite(0,cont,(void*)peloEsc_Sprite, OBJ_SIZE_32X32,1,4,0,0);
                    casillasEsc[0]= cont;
                    cont++;
                    cuentaTomates++;
                    finalJ=1;
                }
                if(tipo_menu_subEsc==casEsc2 && casillasEsc[1]==0){ ...}

                    SE HACE LO MISMO PARA LAS 36 CASILLASPOSIBLES

            if(tipo_menu_subEsc==casEsc36 && casillasEsc[35]==0){...}
            noPulsa=1;

                if(tipo_menu_subEsc == SalirSubEsc ){
                PA_Init();
                menuESCbase();
                cuentaTomates++;
            }
        }
    }
```

En este fragmento de código nos encontramos, con un caso concreto. La variable contador (cuentaTomates) es menor que nueve luego no se han dibujado aun todos los tomates (actualmente sustituidos por pelotas), si final vale 0, entonces aun estamos jugando, esta imagen muestra lo que hace la consola, cuando se presiona en la posición 0, estando en las condiciones anteriormente mencionadas.

Lo primero que hace es mirar si la casilla esta ocupada, si no es así crea un sprite en ella y actualiza el valor de casillasEsc en la posición 0, se incrementa contador para que los sprites creados a continuación no tengan el mismo identificador, se incrementa el contador de tomates y final se pone a uno (así no se dibuja repetidas veces la ficha mientras estas pulsando).

Esto hay que hacerlo para todas las posiciones posibles.

```
//MAQUINA PONE
//A JUGAR
if (cuentaTomates==8){

    //MAQUINA PONE
    while(contMaqEsc<9){

        ran= rand() % 36;

        if(ran==0 && casillasEscMaquina[0]==0){
            PA_CreateSprite(1,contMaqEsc,(void*)peloEsc_Sprite, OBJ_SIZE_32X32,1,4,0,0);
            casillasEscMaquina[0]= contMaqEsc;
            contMaqEsc++;
            cuentaTomates++;
        }
        if(ran==1 && casillasEscMaquina[1]==0){
```

SE HACE LO MISMO PARA LAS 36 CASILLAS POSIBLES

```
    }//fin maq pone ficha
```

Aquí se comprueba lo siguiente, si el jugador ya ha puesto los 8 tomates, se ejecutara esto, mientras que la maquina no haya puesto los 8 tomates.

Para que sea aleatorio usamos la función rand, posteriormente comprobamos si el número que nos ha salido no esta ya ocupado , y si no esta ocupado creamos el sprite, actualizamos el valor de casillasEsMaquina, e incrementamos contMaqEsc (contador de numero de pelotas puesta por la máquina) y cuentaTomates (que almacenará el total de tomates puestos).

```
while(contMaqEsc==9 &&(movJ>0 || movM>0) && final==0){

    if(borra==0){
      //Pausa
        int fines;
        for (fines = 0;fines < 300; fines++){
            PA_WaitForVBL();
        }
        //fin pausa

       //borrarSprites
        contJ=cont;
        contM=contMaqEsc;
            while(contM>0){
                PA_DualDeleteSprite(contM);
                contM--;
            }
            while(contJ>0){
                PA_DualDeleteSprite(contJ);
                contJ--;
            }//fin quitar sprites

            //texto
                PA_InitText(0, 0);
                PA_SetTextTileCol(0,5);
                PA_OutputText(0,0,12,"ENCUENTRA LAS 8 PELOTAS");
                int textE;
                for (textE = 0; textE  < 120; textE++){

                    PA_WaitForVBL();
                }
                PA_ClearTextBg(0);
            //fin texto

                borra=1;
    }
```

Hacemos un bucle para qué el usuario tenga tiempo para tratar de memorizar donde ha puesto la máquina las pelotas, y borramos los sprites para evitar pasarnos del limite de sprites.

PaDualDeleteSprite(numeroSprite): esta función dado un número de sprite te borra ese sprite concreto(de ahí que yo en este caso creara dos bucles, de este modo borraba todos los sprites).

PA_ClearTextBg(pantalla): elimina todo el texto que hay en la pantalla que le pongamos(1 arriba, 0 abajo).

```
//no acierta
else {
    if(tipo_menu_subEsc==casEsc1 && casillasEscMaquina[0]==0 && casillasEscMaquina[0]!=9999){
        PA_CreateSprite(0,contJ,(void*)escon5_Sprite, OBJ_SIZE_32X32,1,14,0,0);//tomate
        contJ++;

        casillasEscMaquina[0]=9999;
        //maquina pone
        maqBu=1;
        jugBu=1;
        noRepite=0;
        noPulsa=1;
        movJ--;
    }
    if(tipo_menu_subEsc==casEsc2 && casillasEscMaquina[1]==0 && casillasEscMaquina[1]!=9999 ){

    SE HACE LO MISMO PARA LOS 36 CASOS
```

En este fragmento podemos ver un ejemplo de que se hace cuando el jugador falla, primero comprueba que casilla se ha pulsado, luego si en esa casilla hay alguna pelota y por último si esta ya ha sido revisada(cuando una casilla ya a sido revisada se dibuja el tomate y sele asigna el valor 9999 en el array).

```
//GANADORES
if(tomatesJ==8 && final==0){
    contGanEsc++;
    PA_OutputText(1,24,10,"Gan:%d " ,contGanEsc );
    PA_OutputText(1,24,13,"Per:%d " ,contPerEsc );
    PA_OutputText(1,24,7,"Mov P:%d " ,movM);
    PA_OutputText(1,24,16,"Pelot:%d " ,tomatesM);
    PA_OutputText(0,24,4,"Pelot:%d " ,tomatesJ);
    int tiemi2;
    for(tiemi2=0; tiemi2<100 ; tiemi2++){
        PA_WaitForVBL();
    }
    PA_Init();
    final=1;
    PA_EasyBgLoad(1, 3, gatoPierde);
    PA_EasyBgLoad(0, 3, esconfond32usu);
    if(final==1){
        PA_LoadDefaultText(0,2);
        PA_SetTextTileCol(0,5);
        PA_LoadDefaultText(1,2);
        PA_SetTextTileCol(1,5);
        PA_OutputText(0,24,7,"Mov P:%d " ,movJ );
        PA_OutputText(0,24,4,"Pelot:%d " ,tomatesJ);
        PA_CreateSprite(0,cartel,(void*)hasGanado1_Sprite, OBJ_SIZE_64X64,1,5,0,64);//
        cartel++;
        PA_CreateSprite(0,cartel,(void*)hasGanado2_Sprite, OBJ_SIZE_64X64,1,6,64,64);//
        cartel++;
        PA_CreateSprite(0,cartel,(void*)hasGanado3_Sprite, OBJ_SIZE_64X64,1,7,128,64);
        cartel++;
        for(tiemi2=0; tiemi2<200 ; tiemi2++){
            PA_WaitForVBL();
        }
        PA_Init();
        menuESCFac();
    }
}
//fin ganadores
```

```
    if(tipo_menu_subEsc == SalirSubEsc ){
        PA_Init();
        menuESCbase();
    }
//Maquina busca
    if(maqBu==1 && movM>=0){
        for (textM = 0; textM  <40; textM++){

            PA_WaitForVBL();
        }
        if(tipo_menu_subEsc == SalirSubEsc ){
            PA_Init();
            menuESCFac();
        }
        while(noRepite==0){
            ran=rand() % 36;
            PA_OutputText(1,24,10,"Gan:%d " ,contGanEsc );
            PA_OutputText(1,24,13,"Per:%d " ,contPerEsc );
            PA_OutputText(1,24,7,"Mov P:%d " ,movM);
            PA_OutputText(1,24,16,"Pelot:%d " ,tomatesM);
            PA_OutputText(0,24,4,"Pelot:%d " ,tomatesJ);

            if(ran==0 && casillasEsc[0]!=0 && casillasEsc[0]!=9999){
                PA_CreateSprite(1,contM,(void*)peloEsc_Sprite, OBJ_SIZE_32X32,1,4,0,0);
                contM++;
                tomatesM++;
                casillasEsc[0]=9999;
                noRepite=1;
                //Jug busca

                    jugBu=0;
                    maqBu=0;
            }
            else if(ran==1 && casillasEsc[1]!=0 && casillasEsc[1]!=9999){
```

LO MISMOPARA LOS 36 CASOS POSIBLES

```
        else{
            if(ran==0 && casillasEsc[0]==0 || ran==0 && casillasEsc[0]!=9999){
                PA_CreateSprite(1,contM,(void*)escon5_Sprite, OBJ_SIZE_32X32,1,14,0,0
                contM++;
                noRepite=1;
                casillasEsc[0]=9999;
                //Jug busca
                    jugBu=0;
                    maqBu=0;
            }
            else if(ran==1 && casillasEsc[1]==0 || ran==1 && casillasEsc[1]!=9999){
```

LO MISMOPARA LOS 36 CASOS POSIBLES

Mirando los fragmentos anteriores, podemos ver como se realiza la comprobación para saber si alguien a ganado y quien ha sido.

Y mostramos el texto por pantalla con la función:

PA_OutputText (u8 screen, u16 x, u16 y, const char * text);
Ésta es la función encargada de imprimir el texto. El primer argumento indica la pantalla como en la función anterior. Con x e y indicamos la posición del texto, en tiles. Y el último argumento es el texto a imprimir, aceptando además argumentos extra al estilo printf().

```
if(noRepite==1){
    movM--;
    PA_OutputText(1,24,10,"Gan:%d " ,contGanEsc );
    PA_OutputText(1,24,13,"Per:%d " ,contPerEsc );
    PA_OutputText(1,24,7,"Mov P:%d " ,movM);
    PA_OutputText(1,24,16,"Pelot:%d " ,tomatesM);
    PA_OutputText(0,24,4,"Pelot:%d " ,tomatesJ);
        //GANADORES
        if(tomatesM==8 && final==0){
            contPerEsc++;
            PA_OutputText(1,24,10,"Gan:%d " ,contGanEsc );
            PA_OutputText(1,24,13,"Per:%d " ,contPerEsc );
            PA_OutputText(1,24,7,"Mov P:%d " ,movM);
            PA_OutputText(1,24,16,"Pelot:%d " ,tomatesM);
            PA_OutputText(0,24,4,"Pelot:%d " ,tomatesJ);
            int tiemi;
            for(tiemi=0; tiemi<100 ; tiemi++){
                PA_WaitForVBL();
            }
            PA_Init();
            final=1;
            PA_EasyBgLoad(1, 3, gatoGana);
            PA_EasyBgLoad(0, 3, esconfond32usu);
            if(final==1){
                PA_LoadDefaultText(0,2);
                PA_SetTextTileCol(0,5);
                PA_LoadDefaultText(1,2);
                PA_SetTextTileCol(1,5);
                PA_OutputText(0,24,7,"Mov P:%d " ,movJ );
                PA_OutputText(0,24,4,"Pelot:%d " ,tomatesJ);
                PA_CreateSprite(0,cartel,(void*)hasPerdido1_Sprite, OBJ_SIZE_64X64,1,8,0,64);//t
                cartel++;
                PA_CreateSprite(0,cartel,(void*)hasPerdido2_Sprite, OBJ_SIZE_64X64,1,9,64,64);//
                cartel++;
                PA_CreateSprite(0,cartel,(void*)hasPerdido3_Sprite, OBJ_SIZE_64X64,1,10,128,64);
                cartel++;
                for(tiemi=0; tiemi<200 ; tiemi++){
                    PA_WaitForVBL();
                }

                PA_Init();
                menuESCFac();
            } SE CIERRAN TODAS LAS LLAVES Y FIN DEL MÉTODO
```

Tanto el menú del nivel medio como el de nivel difícil son muy similares, solo que el contador de movimientos tiene un numero de menor valor, y que en lugar de 8 bolas son 4 y en el difícil una, y estos también contemplan la existencia del empate, este se produce cuando los movimientos de ambos se han acabado pero ninguno a encontrado todas las pelotas.

8.9 menuPOIbase()

```
void menuPOIbase(){
    if (tipo_niv_POI == POIFac){

        PA_EasyBgLoad(1, 3, poinx);
        PA_EasyBgLoad(0, 3, juegos3);

        bool noSalPOI = 1;

                while (noSalPOI) {
                    noSalPOI = (Stylus.Held || Pad.Newpress.Start);
                    PA_WaitForVBL();
                }
                    while (tipo_niv_POI == POIFac) {
                            if (Stylus.Held) {
                                    tipo_menu_poinx = obtener_menu_poinx(Stylus.X, Stylus.Y);
                                        if (tipo_niv_POI == POIFac && tipo_menu_poinx > 63) {
                                            menuPOIFac();
                                        }
                            }
                    PA_WaitForVBL();
                    }
            }//fin raya fac
    //raya med
    if (tipo_niv_POI == POIMed){
    if (tipo_niv_POI == POIDif){
    if (tipo_niv_POI== SalirPOI2) {
        menuPrincipal();
    }

}//fin menu ESCONDITE
```

Como podemos apreciar es igual que los menús anteriores, este es el encargado de llevarnos a los diferentes niveles del poinx.

8.10 menuPOIFac()

```
void menuPOIFac(){
    //CONTINUAR
    PA_Init();
    if (tipo_menu_poinx == ContinuarPOI){
        PA_EasyBgLoad(1, 3, campo2);
        PA_EasyBgLoad(0, 3, campo1);
        int numP;
        numP=0;
        int collarM;
        collarM=0;
        cont=0;
        int numGol;
        numGol=cont;
        PA_LoadSpritePal(0,5,(void*)gol_Pal);
        PA_LoadSpritePal(1,6,(void*)gol_Pal);
        cont++;
        int yc1,yc2,xc1,xc2;
        yc1=160;
        yc2=yc1+32;
        xc1=30;
        xc2=xc1+64;
        PA_LoadSpritePal(0,3,(void*)collar22_Pal);
        PA_CreateSprite(0,cont,(void*)collar22_Sprite, OBJ_SIZE_64X32,1,3,xc1,yc1);
        int collarJugador ;
        collarJugador= cont;
        cont++;
        int ycm1,xcm1;
        ycm1=1;
        xcm1=30;
        PA_LoadSpritePal(1,4,(void*)collar11_Pal);
        PA_CreateSprite(1,cont,(void*)collar11_Sprite, OBJ_SIZE_64X32,1,4,xcm1,ycm1);
        collarM=cont;
        cont++;
        PA_LoadSpritePal(0,1,(void*)pelota_Pal);
        PA_LoadSpritePal(1,2,(void*)pelota_Pal);
```

DECLARACION DE TODAS LAS VARIABLES A USAR

```
//CONTINUAR esC
while (tipo_menu_poinx == ContinuarPOI) {
    if((xc1!=Stylus.X-32 && yc1!=Stylus.Y-16) || (xc1!=Stylus.X-32 || yc1!=Stylus.Y-16)  ){
        xc1=Stylus.X-32;
        yc1=Stylus.Y-16;
        PA_SetSpriteXY(0, collarJugador, xc1, yc1);
    }
    PA_InitText(0, 0);
    PA_InitText(1, 0);
    PA_SetTextTileCol(1,TEXT_BLUE);
    PA_SetTextTileCol(0,TEXT_BLUE);

    PA_OutputText(0,13,1,"%d",contGJ);
    PA_OutputText(1,13,22,"%d",contGM);

    while((posIni==0 && contPos==0) ){
        PA_MoveSprite(collarJugador);
        posIni= rand() % 3;
        //posIni=1;
        if(posIni==1){
            contPos=1;
        }
        else if(posIni==2){

            origenP0=4;
            contPos=1;
            pantalla=1;
            posIni=6;
        }
        else{
            posIni=0;
            contPos=0;
        }
    }
```

```
//dif casos
if(posIni>0 && pantalla==0){
//collar
    if(contclickP==0){
        Stylus.X=62;
        Stylus.Y=176;
        contclickP=1;
    }
    xc1=(Stylus.X)-32;
    yc1=(Stylus.Y)-16;
    xc2=xc1+64;
    yc2=yc1+32;
    int posx2,posy2;
    posx2=0;
    posx2=posx+16;
    posy2=0;
    posy2=posy+16;
    PA_InitText(0, 0);
    PA_InitText(1, 0);
    PA_SetTextTileCol(1,TEXT_BLUE);
    PA_SetTextTileCol(0,TEXT_BLUE);
    PA_OutputText(0,13,1,"%d",contGJ);
    PA_OutputText(1,13,22,"%d",contGM);
```

Este juego fue uno de mis grandes problemas, me costó muchísimo hacer bien los choques y ajustar los movimientos.

Aquí encontramos una función nueva:

PA_InitText(0,0): inicializa el texto en la ventana 0.

En el while vemos como se decide la dirección inicial de la bola.

posIni es la variable encargada de indicar la orientación de la bola.

El primer if que aparece nada mas entrar en el bucle es el encargado de dibujar la bola donde pulse con el stylus mientras que mas tarde vemos otra nueva función de movimiento que afecta al mismo sprite, esta es la siguiente:

PA_MoveSprite(numeroSprite):esta función hace que cuando hagas click sobre un sprite y mantengas puedas ir moviéndolo con el stylus.

```
//MOVER COLLAR MAQUINA
if(posIni<5){
    if( (ycm1<79) && (xcm1<95) ){
        ycm1=ycm1+incyMaq;
        xcm1=xcm1+incxMaq;
        PA_SetSpriteXY(1, collarM, xcm1, ycm1);
    }
    else if( (ycm1<79) && (xcm1>95) ){
        ycm1=ycm1+incyMaq;
        xcm1=xcm1-incxMaq;
        PA_SetSpriteXY(1, collarM, xcm1, ycm1);
    }
    else if( (ycm1>79) && (xcm1<95) ){
        ycm1=ycm1-incyMaq;
        xcm1=xcm1+incxMaq;
        PA_SetSpriteXY(1, collarM, xcm1, ycm1);
    }
    else if( (ycm1>79) && (xcm1>95) ){
        ycm1=ycm1-incyMaq;
        xcm1=xcm1+-incxMaq;
        PA_SetSpriteXY(1, collarM, xcm1, ycm1);
    }
    else if( (ycm1==79) && (xcm1<95) ){
        xcm1=xcm1+incxMaq;
        PA_SetSpriteXY(1, collarM, xcm1, ycm1);
    }
    else if( (ycm1==79) && (xcm1>95) ){
        xcm1=xcm1-incxMaq;
        PA_SetSpriteXY(1, collarM, xcm1, ycm1);
    }
    else if( (ycm1<79) && (xcm1==95) ){
        ycm1=ycm1+incyMaq;
        PA_SetSpriteXY(1, collarM, xcm1, ycm1);
    }
    else if( (ycm1>79) && (xcm1==95) ){
        ycm1=ycm1-incyMaq;
        PA_SetSpriteXY(1, collarM, xcm1, ycm1);
    }
}
```

Aquí podemos ver como movemos el collar de la máquina cuando la bola esta en la pantalla del jugador, esto lo que hace es que cuando la

bola cruza automáticamente el collar (porque los golpeadores con collares de gato) dela máquina se pone en el centro de su pantalla.

```
      //choca arriba
      if( ((((posx+16)>=xc1)&&(posx<=xc2))&& (posy2==yc1)) ||
(posx+16)>=xc1)&&(posx<=xc2))&&((posy2>yc1) && (posy2<yc1+14)) ){
        //viene de la derecha
          if(posIni==3){
              arribaJ=1;
              posIni=4;
          }
        //viene de la izquierda
          else if(posIni==1){
              arribaJ=2;
              posIni=2;
          }
           //
          else {
              posIni=4;
          }
      }

      //choca izquierda
      else if( (((posx+16)==xc1) && ((posy2>yc1) && (posy<yc2)) ) ||
 (((posx+16)>=xc1) &&(posx+16)<=xc1+14) && ((posy2>yc1) && (posy<yc2)) ) {
          //viene de la arriba
              if(posIni==1){
                  posIni=3;
              }
           //viene de abajo
              else if(posIni==2){
                  posIni=4;
              }
              //
              else {
                  posIni=4;
              }
      }
```

```
    //choca derecha
    else if( (((posx)==xc2) && ((posy2>yc1) && (posy<yc2))  ) ||
(((posx)>=xc1+40) &&(posx)<=xc2) && ((posy2>yc1) && (posy<yc2)) ) {

          //viene de la arriba
              if(posIni==3){
                  posIni=1;
              }
           //viene de abajo
              else if(posIni==4){
                  posIni=2;
              }
              //
              else {
                  posIni=2;
              }
    }

    //choca abajo
    else if( (((posx+16>=xc1)&&(posx<=xc2))&& (posy==yc2) ) ||
( ((posx+16>=xc1)&&(posx<=xc2)) && ((posy<yc2) && (posy>yc1+18)) ) ){
          //viene de la derecha
              if(posIni==4){
                  posIni=3;
              }
           //viene de la izquierda
              else if(posIni==2){
                  posIni=1;
              }
          }//fin collar
```

Aquí vemos como se detectan los choques de la bola contra la paleta del jugador y que se hace cuando esto pasa. Las coordenadas de el jugador son xc1 e yc1, mientras que las de la pelota son posx y posy, a la hora de los choques hay que tener en cuenta que las coordenadas que definen a un sprite son las de la esquina superior izquierda.

```
  else if(pantalla==1){
      if(contPAbajo==0){
          posy=170;

          PA_CreateSprite(1,numP,(void*)pelota_Sprite, OBJ_SIZE_16X16,1,2,posx,posy);
      }
      contPAbajo=1;
      if(posIni==5){
          //x+ y+/
              posx=posx+(incrementarx );
              posy=posy+(incrementary );
              //pantalla,sprite,posx,posy
              PA_SetSpriteXY(1, numP, posx, posy);
              //muevo collar máquina
              if( ((ycm1<158)&&(ycm1>0))&& ((xcm1<190)&&(xcm1>0))){
                  origenP0=5;
                  if(xcm1-incxMaq>0){
                      xcm1=xcm1-incxMaq;
                      PA_SetSpriteXY(1, collarM, xcm1, ycm1);
                  }
                  //golpes
                  if( ((((posx+16)>=xcm1)&&(posx<=xcm1+64))&& (posy+16==ycm1)) ||
(((posx+16)>=xcm1)&&(posx<=xcm1+64))&&((posy+16>ycm1) && (posy+16<ycm1+14)) ){
                      origenP0=5;
                      posIni=6;
                  }//choca arriba
                  else if( (((posx+16)==xcm1) && ((posy+16>ycm1) && (posy<ycm1+32)) ) ||
     (((posx+16)>=xcm1) &&(posx+16)<=xcm1+14) && ((posy+16>ycm1) && (posy<ycm1+32)) ) {
                      origenP0=5;
                      posIni=7;
                  }//choca IZQUIERDA
              }
```

```
      if(posy>=170 &&( posx<234 && posx>7)){
          if(origenP0>4){
              origenP0=5;
              posIni=1;
              pantalla=0;
              PA_DualDeleteSprite(numP);
              contPArriba=0;
          }
      }
      else if(posy>=170 && posx>=234){
          if(origenP0>4){
              origenP0=5;
              posIni=3;
              pantalla=0;

              PA_DualDeleteSprite(numP);
              contPArriba=0;
          }
      }
      else if((posy>6 && posy<170 )&& posx>=234){
              origenP0=5;
              posIni=7;
      }
  }//fin posIni==0

  else if(posIni==6){

      else if(posIni==7 ){
          else if(posIni==8 ){
```

Ejemplo de tratamiento bola y movimiento collar de máquina en la pantalla 1. Hay que tener en cuenta que cada caso es diferente pero con ver uno se entiende el resto.

Cuando la bola cruza de pantalla hay que crearla de nuevo en la otra pantalla, pero hay que cambiar las coordenadas de posy y mantener la de posx, ya que cuando sale esta en la y 0 de la pantalla 0, pero en la pantalla 1 entra y la y de abajo es 192, por lo que hay que reajustarla.

También vemos en el fragmento de la derecha como se detectan los goles, básicamente este juego depende muchísimo de las coordenadas.

Fin del menú de poinx fácil, el medio y el difícil son bastante parecidos pero aumentando la velocidad de la bola y la de la máquina, también aumentaba el tamaño de la portería del jugador.

```
if(contPArriba==0){
    posy=0;
    PA_CreateSprite(0,numP,(void*)pelota_Sprite, OBJ_SIZE_16X16,1,1,posx,posy);
}
contPArriba=1;
if(posIni==1){
else if(posIni==2 ){
        //x+ y-
            posx=posx+(incrementarx );
            posy=posy-(incrementary );
            //pantalla,sprite,posx,posy
            PA_SetSpriteXY(0, numP, posx, posy);

                if((posx>7 && posx<234)&& posy<=0){
                    origenP0=2;
                    posIni=6;
                    pantalla=1;

                    PA_DualDeleteSprite(numP);
                    contPAbajo=0;
                }
                else if(posx>=234 && posy<=0){
                    origenP0=2;
                    posIni=8;
                    pantalla=1;

                    PA_DualDeleteSprite(numP);
                    contPAbajo=0;
                }
                else if(posx>=234 && (posy>0 && posy<170)){
                    origenP0=2;
                    posIni=4;
                }
        }

        else if(posIni==3 ){
            else if(posIni==4 ){
```

Ejemplo de tratamiento de la bola en la pantalla 0, cada uno de los 4 casos es diferente.

8.11 menuPARbase()

```
void menuPARbase(){
    if (tipo_niv_PAR == PARFac){
        PA_EasyBgLoad(1, 3, parejas);
        PA_EasyBgLoad(0, 3, juegos3);
        bool noSalPAR = 1;
                while (noSalPAR) {
                    noSalPAR = (Stylus.Held || Pad.Newpress.Start);
                    PA_WaitForVBL();
                }
                    while (tipo_niv_PAR == PARFac) {
                            if (Stylus.Held) {
                                    tipo_menu_parejas = obtener_menu_parejas(Stylus.X, Stylus.Y);
                                        if (tipo_niv_PAR == PARFac && tipo_menu_parejas > 68) {
                                            menuPARFac();
                                        }
                                    }
                    PA_WaitForVBL();
                    }
            }//fin raya fac
    //raya med
    if (tipo_niv_PAR == PARMed){
    if (tipo_niv_PAR == PARDif){
    if (tipo_niv_PAR== SalirPAR2) {
        menuPrincipal();
    }

}//fin menu parejas
```

Es igual a los menús anteriores, pero este es el encargado de llevarnos a los diferentes niveles del juego de las parejas.

8.12 menuPARFac()

```
//PAREJAS
void menuPARFac(){
    //CONTINUAR
    PA_Init();
    if (tipo_menu_parejas == ContinuarPAR){
        PA_LoadSpritePal(0,0,(void*)gato0_Pal);
        PA_LoadSpritePal(0,1,(void*)gato1_Pal);
        PA_LoadSpritePal(0,2,(void*)gato2_Pal);
        PA_LoadSpritePal(0,3,(void*)gato3_Pal);
        PA_LoadSpritePal(0,4,(void*)gato4_Pal);
        PA_LoadSpritePal(0,5,(void*)gato5_Pal);
        PA_LoadSpritePal(0,6,(void*)gato6_Pal);
        PA_LoadSpritePal(0,7,(void*)gato7_Pal);
        PA_EasyBgLoad(1, 3, parejas);
        PA_EasyBgLoad(0, 3, tablero);
        cont=0;
        ContG=0;
        bool noSalpac = 1;
        int aux;
        aux=0;
        while (noSalpac) {
        localCarta[0]=0;
EL ARRAY TIENE 8 POS Y TODAS SE PONEN A 0
        CartaArriba[0]=0;
EL ARRAY TIENE 8 POS Y TODAS SE PONEN A 0
        pintCarta[0]=20;
EL ARRAY TIENE 8 POS Y TODAS SE PONEN A 20
        int ran;
        ran=0;
        cont=1;
```

```
        PA_CreateSprite(0,cont,(void*)gato0_Sprite,
        OBJ_SIZE_64X64,1,0,0,0);
        localCarta[0]=cont;
        cont++;
SE HACE LO MISMO PARA LAS 8 CARTAS

        //CONTADOR COLORES
        cont4=0;
        cont1=0;
        cont2=0;
        cont5=0;
        int yaCarta;
        yaCarta=3;
        cartaGira[0]=0;
        cartaGira[1]=0;
        int carta;
        carta=0;
        int casilla[8];
        casilla[0]=0;
EL ARRAY TIENE 8 POS Y TODAS SE PONEN A 0
//contador para dibujar solo un sprite en una pos
        int temp;
        temp=0;
//contador para que al hacer dos clicks se giren luego
        int cuentaClicks;
        cuentaClicks=0;
```

```
while (tipo_menu_parejas == ContinuarPAR) {
    if (Stylus.Held ){
        if(temp==0) {
        tipo_menu_subParejas = obtener_menu_subParejas(Stylus.X, Stylus.Y);
        if((tipo_menu_subParejas == Carta1) && (casilla[0]==0) ){//linea1
        //Carta 1 primmera vez
        while(  pintCarta[0]==20){
            PA_DualDeleteSprite(localCarta[0]);
            ran=rand() %5;
            if(ran>0){
            if((ran==1) && (cont1<2)){
            PA_CreateSprite(0,cont,(void*)gato1_Sprite, OBJ_SIZE_64X64,1,1,0,0);
                pintCarta[0]=ran;
                cont1++;
                carta=1;
            }
            else if((ran==2) && (cont2<2)){
            else if((ran==3) && (cont5<2)){
            else if((ran==4) && (cont4<2)){
                yaCarta=1;
                casilla[0]=1;
            }
        }
```

```
        //Carta 1 siguientes veces
        if( (pintCarta[0]!=20) && (yaCarta==0) && (casilla[0]==0) ){
        PA_DualDeleteSprite(localCarta[0]);
        if(pintCarta[0]==1){
        PA_CreateSprite(0,cont,(void*)gato1_Sprite, OBJ_SIZE_64X64,1,1,0,0);
        carta=1;
        }
        else if(pintCarta[0]==2){
        else if(pintCarta[0]==3){
        else if(pintCarta[0]==4){
        casilla[0]=1;
        }
        CartaArriba[0]=cont;
        cont++;
        temp++;
        cuentaClicks++;
        yaCarta=0;
    if(cuentaClicks==1){
        cartaGira[0]=carta;
    }
    else{
        cartaGira[1]=carta;
    }
}
```

Este juego me resulto bastante problemático con un error con un sprite que aparecía sin tener que aparecer finalmente logre solventar el problema.

Dentro de el while comprobamos que estamos pulsando, a continuación gracias alas coordenadas del stylus, la función obtener_menu_subParejas nos devuelve en que carta estamos pulsando, por ejemplo si es la carta numero 1, se hace un rand y se decide de este modo que dibujo va a parecer, el random solo se ejecuta la primera vez, de este modo garantizamos que el juego sea diferente el mayor numero de veces, como podéis ver, la imagen de la izquierda muestra como se dibuja la primera vez(es el caso que he mencionado anteriormente), mientras que a la derecha vemos como se dibuja las siguientes veces.

También se puede ver como se le asigna el valor al array cartaGira[i], almacenando de este modo que dos cartas están giradas.

Luego hay que hacer esto para todas las cartas

```
//dejan de hacer click
if(!Stylus.Held){
    temp=0;
    //dos clicks
    if(cuentaClicks==2){
        int ti;
        for (ti = 0; ti < 100; ti++){

            PA_WaitForVBL();
        }
        //si son iguales
        if(cartaGira[0]==cartaGira[1]){
            int cartaQuieta[2];
            cartaQuieta[0]=0;
            cartaQuieta[1]=0;
            int posi;
            posi=0;
            int j;
            for(j=0;j<8;j++){
                if((CartaArriba[j]!=0) && (CartaArriba[j]!=3)){
                    cartaQuieta[posi]=j;
                    posi++;
                }
            }
            int c0,c1;
            c0=0;
            c1=0;
            c0=cartaQuieta[0];
            c1=cartaQuieta[1];
            CartaArriba[c0]=3;
            CartaArriba[c1]=3;
        }
        //si no son iguales
        if((cartaGira[0]!=cartaGira[1]) && ((cartaGira[0]>0 && cartaGira[1]!=0) )){
            if(CartaArriba[0]>4 ){
                PA_CreateSprite(0,localCarta[0],(void*)gato0_Sprite, OBJ_SIZE_64X64,1,0,0,0);
                PA_DualDeleteSprite(CartaArriba[0]);
                CartaArriba[0]=0;
                casilla[0]=0; ESTO HAY QUE HACERLO PARA TODAS LAS CARTAS
            }
            }
            cuentaClicks=0;
            cartaGira[0]=0;
            cartaGira[1]=0;
        }//fin 2 clicks
        //SI HAS GANADO
        if((CartaArriba[0]==3) && (CartaArriba[1]==3) && (CartaArriba[2]==3)
&& (CartaArriba[3]==3) && (CartaArriba[4]==3) && (CartaArriba[5]==3)
&& (CartaArriba[6]==3) && (CartaArriba[7]==3) ){
            int FIN;
            for (FIN = 0; FIN  < 60; FIN++){

                PA_WaitForVBL();
            }
            ContG++;
            PA_LoadDefaultText(1,2);
            PA_SetTextTileCol(1,TEXT_BLUE);
            if(ContG==1){
                PA_OutputText(1,0,23,"HAS GANADO, %d VEZ", ContG);
            }
            else{
                PA_OutputText(1,0,23,"HAS GANADO, %d VECES", ContG);
            }

            for (FIN = 0; FIN  < 120; FIN++){

                PA_WaitForVBL();
            }
            //borar todos los sprites
            while(cont>0){
                PA_DualDeleteSprite(cont);
                cont--;
            }
            menuPARFac();
        }//FIN SI HAS GANADO
    }//fin
    PA_WaitForVBL();
}
```

Aquí encontramos como se lleva a cabo la comprobación de si las cartas giradas son iguales, y vemos que si no son iguales estas se giran y ya esta.

A la derecha esta la comprobación de que se ha montado el puzle completamente y aparece un mensaje de texto diciéndotelo.

En el nivel difícil también te saldrá si has perdido, ya que se juega con límite de movimientos y si transcurridos estos no lo has logrado resolver pierdes.

Después de esto esta la parte de código de ayuda y salir, pero como no parecen funciones nuevas ni métodos en si diferentes, no es necesario que lo vuelva a explicar.
Si bien esto es para la opción fácil, en las opciones nivel medio y nivel difícil no aparecen funciones de PAlib que no se hayan visto anteriormente, si bien se añaden cosas, cambian valores de variables, aparece una pareja mas en pantalla, y en el nivel difícil un limite de movimientos, no es necesario explicarlo todo de nuevo, ya que con lo citado aquí ya se puede entender todo el código.

8.13 menuRaya()

```
//menu raya para seleccionar nivel
void menuRaya(){
    if (tipo_niv_raya == rayaFac){
        PA_EasyBgLoad(1, 3, enraya0);
        PA_EasyBgLoad(0, 3, enraya2);
        bool noSalRAY = 1;
            while (noSalRAY) {
            while (tipo_niv_raya == rayaFac) {
                if (Stylus.Held) {
                    tipo_menu_raya = obtener_menu_raya(Stylus.X, Stylus.Y);
                        if (tipo_niv_raya == rayaFac && tipo_menu_raya > 11) {
                            menuRayaFac();
                        }
                }
                PA_WaitForVBL();
            }
    }//fin raya fac
    //raya med
    if (tipo_niv_raya == rayaMed){

    if (tipo_niv_raya == rayaDif){
    if (tipo_niv_raya == salirRaya) {
        menuPrincipal();
    }
}//fin menu raya
```

Como podemos ver sigue siendo igual a los menús anteriores, este es el encargado de llevarnos a los diferentes niveles del tres en raya.

8.14 menuRayaDif()

En este caso he creído oportuno explicar el difícil ya que tiene bastantes mas cosas que el nivel fácil.

Tras hacer el PA_Init(), cargar las paletas, crear e inicializar las variables necesarias y cargar los fondos empieza lo difícil.

```
while (tipo_menu_raya == ContinuarRaya) {
    PA_LoadDefaultText(1,2);
    PA_SetTextTileCol(1,TEXT_GREEN);
    PA_OutputText(1,0,23, "EMPATE:%d,GANADO:%d,PERDIDO:%d", contEmp,contGan,contPer);
    PA_LoadDefaultText(0,2);
    PA_SetTextTileCol(0,3);
    PA_OutputText(0,23,3,"%d de 10",contMov);
    PA_SetTextTileCol(0,5);
    PA_OutputText(0,23,7,"seg: %d " ,segundosCont);
    milesimas++;
    if (((milesimas==60) && (segundosCont>0)) && (Bloc==0)){
        segundosCont--;
        PA_OutputText(0,23,7,"seg: %d " ,segundosCont);
        milesimas=0;
    }
            if (Stylus.Newpress) { //para que selec se ejecute despues de dibujar maq y jug
                tipo_menu_rayaDib = obtener_menu_rayaDib(Stylus.X, Stylus.Y);
                //seleccionar ficha1 jug
                if((tipo_menu_rayaDib == Dib11) && ( saltaDiv==0 ) && (tresOc[0]==1) && (Bloc==0) && (contJ==3) ){
                    nomF=tresNom[0];
                    contJ--;
                    dibOmov=1;
                    continue;//sal
                }//
```

A continuación explicaré que hace línea por línea:

Mientras que estamos en ContinuarRaya, se inicializa el texto, se le asigna un color, se muestra el texto ,se repiten estas operaciones para la pantalla inferior.

Se incrementa la variable milésimas, cuando esta vale 60 significa que ha pasado un segundo , se comprueba que segundosCont tiene un valor superior a 0 y que el juego no ha finalizado ya por otro motivo, entonces se le resta uno a segundosCont y se muestra por pantalla(este será nuestro contador, si pasan 60 segundos finaliza el juego este como este la partida).

Aquí aparece una nueva función:

Stylus.Newpress : Cuando se pulsa la pantalla táctil, vale 1 durante 1 frame.

Y luego obtenemos en que casilla hemos pulsado y el jugador selecciona su ficha.

```
//dibujar ficha1 jug
if((tipo_menu_rayaDib == Dib11) && ( saltaDiv==0 ) && (tresOc[0]==0) && (Bloc==0) && (contJ<3) ){
    if(dibOmov==2){
        PA_CreateSprite(0,cont,(void*)fichaJug_Sprite, OBJ_SIZE_32X32,1,0,21,21);
        tresNom[0]=cont;//se guarda el nom del sprite
        cont++;
    }
    if(dibOmov==1){
        PA_SetSpriteXY(0, nomF, 21, 21);
        for(i=0;i<9;i++){
            if(tresNom[i]==nomF){
                tresNom[i]=9;
                tresOc[i]=0;
            }
        }
        tresNom[0]=nomF;//se guarda el nom del sprite
        contMov++;//incrementar movimiento
    }
    tresOc[0]=1;//ocupada por jugador
    contJ++;
    saltaDiv=1;

}  ESTO SE HACE PARA TODAS LAS POSICIONES
```

El siguiente paso es dibujar la ficha, comprobamos que hemos seleccionado esta casilla, que le toca poner al jugador, que la casilla esta libre y por último que contJ sea menor que 3. Luego dependiendo del valor de dibOmov dibujare la ficha o la moveré.

Esto hay que hacerlo para todos los casos posibles.

Comprobando resultado:

```
//EMPATE?
if((((contMov==10)&& (Bloc==0)) ||((segundosCont==0)&& (Bloc==0))){
    //PA_LoadDefaultText(0,2);
    PA_SetTextTileCol(0,TEXT_BLUE);
    PA_OutputText(0,10,22, "EMPATE");
    for(i=0; i<90 ; i++){
        PA_WaitForVBL();
    }
    contEmp++;
    Bloc=1;
}//fin empate
//Gana maquina?
if((Bloc==0) && ((tresOc[0]==2 && tresOc[1]==2 && tresOc[2]==2 ) ||
    (tresOc[0]==2 && tresOc[4]==2 && tresOc[8]==2) ||
    (tresOc[0]==2 && tresOc[3]==2 && tresOc[6]==2) ||
    (tresOc[1]==2 && tresOc[4]==2 && tresOc[7]==2) ||
    (tresOc[2]==2 && tresOc[4]==2 && tresOc[6]==2) ||
    (tresOc[2]==2 && tresOc[5]==2 && tresOc[8]==2) ||
    (tresOc[3]==2 && tresOc[4]==2 && tresOc[5]==2) ||
    (tresOc[6]==2 && tresOc[7]==2 && tresOc[8]==2))){
    PA_LoadDefaultText(0,2);
    PA_SetTextTileCol(0,TEXT_RED);
    PA_OutputText(0,10,22, "HAS PERDIDO");
    for(i=0; i<90 ; i++){
        PA_WaitForVBL();
    }
    Bloc=1;
    contPer++;
}//fin Gana máquina
```

```
//Gana jugador?
if((Bloc==0) && ((tresOc[0]==1 && tresOc[1]==1 && tresOc[2]==1 ) ||
    (tresOc[0]==1 && tresOc[4]==1 && tresOc[8]==1) ||
    (tresOc[0]==1 && tresOc[3]==1 && tresOc[6]==1) ||
    (tresOc[1]==1 && tresOc[4]==1 && tresOc[7]==1) ||
    (tresOc[2]==1 && tresOc[4]==1 && tresOc[6]==1) ||
    (tresOc[2]==1 && tresOc[5]==1 && tresOc[8]==1) ||
    (tresOc[3]==1 && tresOc[4]==1 && tresOc[5]==1) ||
    (tresOc[6]==1 && tresOc[7]==1 && tresOc[8]==1))){
    PA_LoadDefaultText(0,2);
    PA_SetTextTileCol(0,TEXT_GREEN);
    PA_OutputText(0,10,22, "HAS GANADO");
    for(i=0; i<90 ; i++){
        PA_WaitForVBL();
    }
    Bloc=1;

    contGan++;
}//fin jugador
```

Ejemplo de máquina seleccionando ficha

```
//dibujar maquina
 if((saltaDiv == 1) && ( tresOc[0]==0 || tresOc[1]==0 || tresOc[2]==0 || tresOc[3]==0 || tresOc[4]==0 || tresOc[5]==0 ||
tresOc[6]==0 || tresOc[7]==0 || tresOc[8]==0) && (Bloc==0) ){
    //para ver como pone la maquina
    for(i=0; i<30 ; i++){
        PA_WaitForVBL();
    }
    nomF=9;//para resetear nomF
    //si estan las 3 decide cual mover
    if(contM==3){
        //seleccionar 0
        if((tresOc[0]==2)&&( ((tresOc[1]==0)&&(tresOc[4]==2)&&(tresOc[7]==2)) || ((tresOc[1]==2)&&(tresOc[4]==0)&&(tresOc[7]==2)) ||
                ((tresOc[1]==2)&&(tresOc[4]==2)&&(tresOc[7]==0)) || ((tresOc[3]==0)&&(tresOc[4]==2)&&(tresOc[5]==2)) ||
                ((tresOc[3]==2)&&(tresOc[4]==0)&&(tresOc[5]==2)) || ((tresOc[3]==2)&&(tresOc[4]==2)&&(tresOc[5]==0)) ||
                ((tresOc[3]==0)&&(tresOc[4]==2)&&(tresOc[5]==2)) || ((tresOc[6]==0)&&(tresOc[7]==2)&&(tresOc[8]==2)) ||
                ((tresOc[6]==2)&&(tresOc[7]==0)&&(tresOc[8]==2)) || ((tresOc[6]==2)&&(tresOc[7]==2)&&(tresOc[8]==0)) ||
                ((tresOc[6]==0)&&(tresOc[4]==2)&&(tresOc[2]==2)) || ((tresOc[6]==2)&&(tresOc[4]==0)&&(tresOc[2]==2)) ||
                ((tresOc[6]==2)&&(tresOc[4]==2)&&(tresOc[2]==0)) || ((tresOc[2]==0)&&(tresOc[5]==2)&&(tresOc[8]==2)) ||
                ((tresOc[2]==2)&&(tresOc[5]==0)&&(tresOc[8]==2)) || ((tresOc[2]==2)&&(tresOc[5]==2)&&(tresOc[8]==0)) )){

            dibOmovM=1;
            nomF=tresNom[0];
        }
```

Desarrollar la inteligencia de la maquina fue algo bastante costoso ya que había que tener en cuenta muchísimos casos, arriba podemos ver uno de ellos.

Ejemplo de máquina dibujando

```
//poner en la casilla 4
if((saltaDiv == 1) && (
(tresOc[0]==1 && tresOc[1]==0 && tresOc[2]==0 && tresOc[3]==0 && tresOc[4]==0 && tresOc[5]==0 && tresOc[6]==0 && tresOc[7]==0 && tresOc[8]==0) ||
(tresOc[0]==0 && tresOc[1]==1 && tresOc[2]==0 && tresOc[3]==0 && tresOc[4]==0 && tresOc[5]==0 && tresOc[6]==0 && tresOc[7]==0 && tresOc[8]==0) ||
(tresOc[0]==0 && tresOc[1]==0 && tresOc[2]==1 && tresOc[3]==0 && tresOc[4]==0 && tresOc[5]==0 && tresOc[6]==0 && tresOc[7]==0 && tresOc[8]==0) ||
(tresOc[0]==0 && tresOc[1]==0 && tresOc[2]==0 && tresOc[3]==1 && tresOc[4]==0 && tresOc[5]==0 && tresOc[6]==0 && tresOc[7]==0 && tresOc[8]==0) ||
(tresOc[0]==0 && tresOc[1]==0 && tresOc[2]==0 && tresOc[3]==0 && tresOc[4]==0 && tresOc[5]==1 && tresOc[6]==0 && tresOc[7]==0 && tresOc[8]==0) ||
(tresOc[0]==0 && tresOc[1]==0 && tresOc[2]==0 && tresOc[3]==0 && tresOc[4]==0 && tresOc[5]==0 && tresOc[6]==1 && tresOc[7]==0 && tresOc[8]==0) ||
(tresOc[0]==0 && tresOc[1]==0 && tresOc[2]==0 && tresOc[3]==0 && tresOc[4]==0 && tresOc[5]==0 && tresOc[6]==0 && tresOc[7]==1 && tresOc[8]==0) ||
(tresOc[0]==0 && tresOc[1]==0 && tresOc[2]==0 && tresOc[3]==0 && tresOc[4]==0 && tresOc[5]==0 && tresOc[6]==0 && tresOc[7]==0 && tresOc[8]==1) ||
(tresOc[4]==0 && tresOc[0]==2 && tresOc[8]==2) || (tresOc[4]==0 && tresOc[1]==2 && tresOc[7]==2) || (tresOc[4]==0 && tresOc[2]==2 &&tresOc[6]==2) ||
(tresOc[4]==0 && tresOc[3]==2 && tresOc[5]==2) || (tresOc[4]==0 && tresOc[0]==1 && tresOc[8]==1) || (tresOc[4]==0 && tresOc[1]==1 &&tresOc[7]==1) ||
(tresOc[4]==0 && tresOc[2]==1 && tresOc[6]==1) || (tresOc[4]==0 && tresOc[3]==1 && tresOc[5]==1))){

    if((contM<3) && (dibOmovM==0)){
        PA_CreateSprite(0,cont,(void*)fichaMac_Sprite, OBJ_SIZE_32X32,1,1,76,70);
        tresNom[4]=cont;//se guarda el nom del sprite
        cont++;
        contM++;
    }

    if((contM==3) && (dibOmovM==1)){
        PA_SetSpriteXY(0, nomF, 76, 70);
        PA_LoadDefaultText(0,2);
        PA_SetTextTileCol(0,TEXT_GREEN);
        PA_OutputText(0,10,22, "4");
        for(i=0;i<9;i++){
            if(tresNom[i]==nomF){
                tresNom[i]=9;
                tresOc[i]=0;
            }
        }
        tresNom[4]=nomF;//se guarda el nom del sprite
        contMov++;//incrementar movimiento
    }
    saltaDiv=0;
    tresOc[4]= 2;//ocupada por maquina

}//fin pone 4
```

Como se puede ver arriba, la maquina también contempla si esta dibujando o moviendo.

Y al darle a reiniciar, se resetea el valor de todas las variables.

8.15 menuPuzle()

Tras hacer el PA_Init(), comprobar que tipo_menu_puzle es igual ContinuarPuzle, cargamos paletas inicializamos variables, y proseguimos con el desarrollo.

```
intro();
desordenar_puzle();
PA_LoadDefaultText(1,2);
PA_SetTextTileCol(1,TEXT_RED);
int ino;
//CONTINUAR PUZZLE
while (tipo_menu_puzle == ContinuarPuzle) {
    if (Stylus.Held) {
        tipo_menu_puzleDib = obtener_menu_puzleDib(Stylus.X, Stylus.Y);
            if (tipo_menu_puzleDib == PuDib11 && puzlOc[0] != 0) {
                if (puzlOc[1] == 0) {
                    puzlOc[1] = puzlOc[0];
                    puzlOc[0] = 0;
                    PA_SetSpriteXY(0, puzlOc[1], 64, 0);

                    casillasPieza[1]=casillasPieza[0];
                    casillasPieza[0]=10;
                }
                else {
                    if (puzlOc[3] == 0) {
                        puzlOc[3] = puzlOc[0];
                        puzlOc[0] = 0;
                        PA_SetSpriteXY(0, puzlOc[3], 0, 64);

                        casillasPieza[3]=casillasPieza[0];
                        casillasPieza[0]=10;
                    }
                }
            }
```

Aquí podemos ver el ejemplo de mover una pieza del puzle.

Ejemplo comprobación puzle resuelto:

```
            if((casillasPieza[0]==10 && casillasPieza[1]==1 && casillasPieza[2]==2 && casillasPieza[3]==3 && casillasPieza[4]==4 &&
    casillasPieza[5]==5 && casillasPieza[6]==6 && casillasPieza[7]==7 && casillasPieza[8]==8) ||
   (casillasPieza[0]==0 && casillasPieza[1]==10 && casillasPieza[2]==2 && casillasPieza[3]==3 && casillasPieza[4]==4 && casillasPieza[5]==5 &&
   casillasPieza[6]==6 && casillasPieza[7]==7 && casillasPieza[8]==8) ||(casillasPieza[0]==0 && casillasPieza[1]==1 && casillasPieza[2]==10 &&
   casillasPieza[3]==3 && casillasPieza[4]==4 &&casillasPieza[5]==5 && casillasPieza[6]==6 && casillasPieza[7]==7 && casillasPieza[8]==8) ||
   (casillasPieza[0]==0 && casillasPieza[1]==1 && casillasPieza[2]==2 && casillasPieza[3]==10 && casillasPieza[4]==4 &&
   casillasPieza[5]==5 && casillasPieza[6]==6 && casillasPieza[7]==7 && casillasPieza[8]==8) ||(casillasPieza[0]==0 && casillasPieza[1]==1 &&
  casillasPieza[2]==2 && casillasPieza[3]==3 && casillasPieza[4]==10 && casillasPieza[5]==5 && casillasPieza[6]==6 && casillasPieza[7]==7 &&
  casillasPieza[8]==8) ||(casillasPieza[0]==0 && casillasPieza[1]==1 && casillasPieza[2]==2 && casillasPieza[3]==3 && casillasPieza[4]==4 &&
   casillasPieza[5]==10 && casillasPieza[6]==6 && casillasPieza[7]==7 && casillasPieza[8]==8) ||(casillasPieza[0]==0 && casillasPieza[1]==1 &&
 casillasPieza[2]==2 && casillasPieza[3]==3 && casillasPieza[4]==4 && casillasPieza[5]==5 && casillasPieza[6]==10 && casillasPieza[7]==7
 && casillasPieza[8]==8) ||(casillasPieza[0]==0 && casillasPieza[1]==1 && casillasPieza[2]==2 && casillasPieza[3]==3 && casillasPieza[4]==4 &&
   casillasPieza[5]==5 && casillasPieza[6]==6 && casillasPieza[7]==10 && casillasPieza[8]==8) ||(casillasPieza[0]==0 && casillasPieza[1]==1 &&
casillasPieza[2]==2 && casillasPieza[3]==3 && casillasPieza[4]==4 && casillasPieza[5]==5 && casillasPieza[6]==6 && casillasPieza[7]==7
&& casillasPieza[8]==10) ){
                PA_CreateSprite(1,cont,(void*)hasGanado1_Sprite, OBJ_SIZE_64X64,1,39,32,64);//tomate
                cont++;
                PA_CreateSprite(1,cont,(void*)hasGanado2_Sprite, OBJ_SIZE_64X64,1,40,96,64);//tomate
                cont++;
                PA_CreateSprite(1,cont,(void*)hasGanado3_Sprite, OBJ_SIZE_64X64,1,41,160,64);//tomate
                    for(ino=0; ino<120 ; ino++){
                            PA_WaitForVBL();
                    }
                    PA_Init();
                    intro();
                    desordenar_puzle();
            }
```

Hay que tener en cuenta los nueve casos posibles, para poder hacer esto cree el array que vemos arriba, este se rellena en el método desordenar, y aquí se le va actualizando el valor cada vez que se mueve una pieza, de esta forma cuando posición y pieza coincida en todo el array el juego finaliza.

```
//reiniciar puzle
if(tipo_menu_puzleDib == ReiniciarDibuPuzle){
    PA_Init();
    intro();
    desordenar_puzle();
}
//SIGUIENTE puzle
if(tipo_menu_puzleDib == SiguienteDibuPuzle){
    PA_Init();
    numPuzle++;
    if(numPuzle==4){
        numPuzle=0;
    }
    intro();
    desordenar_puzle();
}
//Salir Puzle
if(tipo_menu_puzleDib == SalirDibuPuzle){

    PA_Init();
    menuJuegos();
}
```

Aquí podemos ver que se hace cuando se pulsa el botón reiniciar, al pulsar siguiente y al pulsar salir.

Lo ultimo que añadí, fue el pulsar siguiente dependiendo de si se pulsa o no se cargara un puzle u otro y cuando numPuzle valga 0 de nuevo se vuelve a cargar el mismo puzle y así sucesivamente, para darle fluidez al juego.

8.16 desordenar_puzle()

Este es uno delos métodos que mas me costó desarrollar, no ya por su complejidad sino porque no se me ocurría muy bien como crearlo, ya que es un código bastante extenso (y repetitivo) voy a tratar solo una parte de él.

```
if(numPuzle==0){
    //puzle1
        PA_LoadSpritePal(0,3,(void*)CaPuz11_Pal);
        PA_LoadSpritePal(0,4,(void*)CaPuz12_Pal);
        PA_LoadSpritePal(0,5,(void*)CaPuz13_Pal);

        PA_LoadSpritePal(0,6,(void*)CaPuz21_Pal);
        PA_LoadSpritePal(0,7,(void*)CaPuz22_Pal);
        PA_LoadSpritePal(0,8,(void*)CaPuz23_Pal);

        PA_LoadSpritePal(0,9,(void*)CaPuz31_Pal);
        PA_LoadSpritePal(0,10,(void*)CaPuz32_Pal);
        PA_LoadSpritePal(0,11,(void*)CaPuz33_Pal);

    while (fichas_sin_poner > 0) {
        posicion = rand() % 9;

        if(puzlOc[posicion] == 0) {
            if (posicion == 0){

                pieza=rand() % 9;

                if((pieza ==0) && ( cont11==0)){
                    PA_CreateSprite(0,cont,(void*)CaPuz11_Sprite, OBJ_SIZE_64X64,1,3,0,0);
                    puzlOc[posicion] = cont;
                    cont11=1;
                    cont++;
                    fichas_sin_poner--;
                }
                if((pieza ==0) && ( cont11==1) && ((cont-1) != puzlOc[posicion])){
                    pieza=rand() % 9;
                }   ESTO HAY QUE HACERLO PARA TODAS LAS PIEZAS Y TODAS LAS POSICIONES, A SU VEZ TAMBIÉN,
                    PARA TODOS LOS VALORES DE NUMPUZLE
```

Como podemos apreciar, tanto la carga de paletas como el funcionamiento básico de este método están dentro de un if, bien, pues es que dado las restricciones de la consola NDS con respecto a que solo permite cargar 16 paletas y con eso no me llegaba para las de todos los puzles, cada vez que le toca a un puzle se cargan las paletas de este, así no hay problemas como los que me ocurrieron hasta que lo descubrí.

Bien este método básicamente mediante un random decide en que posición va poner la ficha, y una vez decidida la posición si esta esta vacía, hace otro random para decidir que pieza dibujar, comprueba que esta no haya sido dibujada con anterioridad, esto hay que hacerlo para todas las piezas y para todas las posiciones.

8.17 intro()

```
void intro(){
    PA_Init();
   if(numPuzle<4){
        int ino;
        //MOSTRAR IMAGEN COMPLETA
        PA_EasyBgLoad(1, 3, puzzles2);
        PA_EasyBgLoad(0, 3, puzzles4);
        PA_LoadDefaultText(0,2);
        PA_SetTextTileCol(0,TEXT_BLUE);
        PA_OutputText(0,5,10, "MIRA LA IMAGEN");

        for(ino=0; ino<90 ; ino++){
                PA_WaitForVBL();
        }
        PA_Init();
        PA_EasyBgLoad(1, 3, puzzles2);
        if(numPuzle==0){
            PA_EasyBgLoad(0, 3, puzleResul);
        }
        else if(numPuzle==1){
            PA_EasyBgLoad(0, 3, puzleResul2);
        }
        else if(numPuzle==2){
            PA_EasyBgLoad(0, 3, puzleResul3);
        }
        else if(numPuzle==3){
            PA_EasyBgLoad(0, 3, puzleResul4);
        }
        for(ino=0; ino<180 ; ino++){
            PA_WaitForVBL();
        }

          //A JUGAR
          PA_Init();
          PA_EasyBgLoad(1, 3, puzzles2);
          PA_EasyBgLoad(0, 3, puzzles4);
          PA_LoadDefaultText(0,2);
          PA_SetTextTileCol(0,TEXT_GREEN);
          PA_OutputText(0,7,10, "A JUGAR");
          for(ino=0; ino<180 ; ino++){
              PA_WaitForVBL();
          }
          //A JUGAR
          PA_Init();
          PA_EasyBgLoad(1, 3, puzzles2);
          PA_EasyBgLoad(0, 3, puzzles4);
          PA_LoadDefaultText(0,2);
          PA_SetTextTileCol(0,TEXT_GREEN);
          PA_OutputText(0,7,10, "A JUGAR");
          for(ino=0; ino<90 ; ino++){
              PA_WaitForVBL();
          }
    }
}   //fin intro
```

Este es un método muy sencillos su función se aprecia a simple vista, es el método que se encarga de mostrar la imagen que habrá que componer con el haciendo el puzle.

Se hace un se comprueba que numPuzle sea menor a cuatro, y se carga la imagen que corresponda dependiendo del valor de numPuzle.

8.19 Métodos obtener

Son todos iguales, solo cambia lo la variable que devuelven y las coordenadas que se pulsan , son los que nos permiten crear los menús.

A continuación tenemos un ejemplo:

```
u8 obtener_menu_subParejasMD(s32 xStylus, s32 yStylus) {
    if (pulsado_boton_con_stylus(xStylus, yStylus, 0, 0, 63, 60)) {
            return Carta1;
    }
    else if (pulsado_boton_con_stylus(xStylus, yStylus,67, 5, 122, 60)) {
            return Carta2;
        }
        else if (pulsado_boton_con_stylus(xStylus, yStylus, 130, 5,189, 60)) {
               return Carta3;
            }
            //linea2
            else if (pulsado_boton_con_stylus(xStylus, yStylus, 3, 67,60, 124)) {
                    return Carta4;
                }
                  else if (pulsado_boton_con_stylus(xStylus, yStylus,66, 66, 123, 123)) {
                         return Carta5;
                    }
                    else if (pulsado_boton_con_stylus(xStylus, yStylus, 131, 67, 188, 125)) {
                           return Carta6;
                        }
                        //linea3
                        else if (pulsado_boton_con_stylus(xStylus, yStylus, 35, 130, 91, 188)) {
                               return Carta7;
                            }
                              else if (pulsado_boton_con_stylus(xStylus, yStylus, 98, 132, 157, 188)) {
                                    return Carta8;
                                }
                              else if (pulsado_boton_con_stylus(xStylus, yStylus, 193, 5, 252, 60)) {
                                    return Carta9;
                                }
                              else if (pulsado_boton_con_stylus(xStylus, yStylus, 193, 67, 252, 124)) {
                                    return Carta10;
                                }
    else if (pulsado_boton_con_stylus(xStylus, yStylus, 208, 157, 254, 190)) {
            return SalirSubParejas;
        }
        else {
            return 0;
        }
}//fin menu parejas?
```

Este es el obtener que utilizan menuPARMed y menuPARDif para decidir que hacer, como se puede apreciar arriba el funcionamiento es muy simple.

9. PROBLEMAS EN EL DESARROLLO

9.1 Puzle

Al añadir los puzles me ocurrió el siguiente problema, al cargar los fragmentos había algunos que salían como si la paleta no funcionaba, buscando información, vi que era porque me estaba saltando las restricciones de la NDS y es que esta acepta hasta 16 paletas de 256 colores, el modo recomendado por la relación memoria-calidad.

Este problema lo resolví metiendo la declaración de paletas dentro del if (numPuzle == *elNumeroDePuzle*){…}, así cada vez solo cargo las paletas de el puzle que estoy dibujando.

Figura 54: Puzle1

Figura 55: Puzle2

Figura 56: Puzle3

Figura 57: Puzle4

9.2 Parejas

A la hora de programar este juego , Problema con la capacidad de la RAM , no me dejaba poner todas las cartas, la solución me la dio una página web [1]:

Poniendo: vramSetBankE(VRAM_E_MAIN_SPRITE); en menuParejas.

Pero esta solución solo era parcial mirando en paginas al final descubrí como borrarlas, obtuve bastante información en [2] , [3] y [4] , así descubrí que con la función *PA_DualDeleteSprite(aux);* se borra el sprite que quiera.

Luego el problema lo solucione con :

```
if (Stylus.Held && temp==0) {
    tipo_menu_subParejas = obtener_menu_subParejas(Stylus.X,
    Stylus.Y);

    if(tipo_menu_subParejas == Carta1){ //linea1.....}

if(!Stylus.Held){
    temp=0;}
```

Para que las cartas no se pongan siempre en la misma posición usaremos la función rand, para obtener un numero aleatorio (aux= rand() % 9)

Cuando ya funcionaba todo "aparentemente", empezó a aparecerme la cara del gato en la posición 0, cuando dibujaba otra carta:

Figura 58: Puzle Fallo1

Figura 59: Puzle Fallo2

Falla al jugar n partidas seguidas donde n> = 6 empieza bien y llega un momento en el que no dibuja los sprites y luego al salir y meterme una vez falla el tablero aparece siempre vacío.

Figura 60: Puzle Fallo3

Figura 61: Puzle Fallo4

Pero se pueden jugar bien 5 partidas y detecta bien que el jugador ha ganado. Intente solucionar lo de el click sobre una misma casilla con un array llamado casilla me dice si la he pulsado ahora , que casilla, y si esta pulsada no se puede volver a pulsar.

Al apretar la casilla 1 y luego la dos y después la tres y luego la 2 la 2 no se muestra y la tres se queda cuando debería girarse, el problema parece afectar solo a la tercera casilla.

Figura 62: Puzle Fallo5

El fallo resulto ser un error simple, sin darme cuenta puse casilla[0]==0 dentro del while en lugar de en el if entonces salía del while sin que tuviera que salir.

Mas tarde vuelve ha ocurrir el problema de que se queda la casilla 3 con el dibujo, sigo probando

El juego vuelve a pararse porque la casilla3 coge un valor diferente del que debiera y por lo tanto no se detecta el fin ya la casilla 3 lleva con la imagen fija desde el principio. El fallo volvió a ser un error de

programación no se le asignaba valor a la variable yaCarta en la parte de dibujo de la carta 3.

Pero seguía apareciendo la cara del gato en la posición 0, y aunque mostré los valores por la pantalla para ver si algo se cargaba mal, no cargaba nada mal, así que seguía sin entender que pasaba.

Tras muchas pruebas seguía fallando tras unos ajustes como un fallo que había en una pos de un array cada vez fallaba menos, rara vez se pintaba el gato pero seguía pintándose y puedo jugar 6 partidas bien pero tras recorrer muchísimas paginas seguía igual y por alguna razón no se puede pasar de las 6 partidas , y se seguía dibujando el gato en la casilla 0 .

Traza uno hare muchas pruebas de pos para ver que pasa en la casilla 0, y de este modo comprobar si falla con cualquier casilla.

Figura 63: Puzle fallo cara de gato

Al final resulta que se cargaban unos sprites sobre otros así que poniendo la variable cont=0 nada más empezar en el juego, y antes de reiniciar la siguiente partida hago un delete de todos los sprites de la siguiente forma:

```
while(cont>0){
        PA_DualDeleteSprite(cont);
                cont--;
}
```

Tras esto ya me carga bien todas las partidas que quiera en este screen se puede ver que ya llego a la partida 9

Figura 64: Puzle pasando de 6 partidas

Al final se me ocurrió que como el fallo era dibujar alguna de las casillas la primera vez , pensé que en el bucle de cuando se pintan por primera vez comprobando que no se ha dado la vuelta a la casilla0 , volvía a cargar la parte trasera de la carta y se solucionaría el problema.

Una vez hecho esto ya no aparece el gato en la casilla 0.

9.3 Escondite

El primer problema que tuve con este juego fue el siguiente, aparece el cartel" pon los tomates" puedo poner los 18 tomates, entonces la maquina pone los suyos(de forma provisional para ver que los pone he hecho que se muestren),entonces se hace un wait, y se borran todos los sprites, sale el cartel de busca los tomates y entonces , mis clicks no son obedecidos por la consola, sin embargo la maquina si que busca los tomates, también resulta que si se mete en el bucle de Jugador busca y cambia los valores de la variables maqBu y jugBu, por lo tanto la maquina juega sus 36 intentos

Y el Jugador no lo hace, mas como ya he dicho cambia los valores de esas variables y se va actualizando su contador de movimientos.

Estoy tratando de que solo busque en una casilla una vez muestra

Figura 65: Escondite fallo1

Posibles escondites y resultados:

Figura 66: Escondite Ejecución fallo1

Figura 67: Escondite Ejecución fallo2

Al final tras muchos días de dar vueltas descubrí el problema, resulta que había una línea :

else if(ran==3 && casillasEsc[3]!=0 || ran==3 && casillasEsc[3]!=9999){

En lugar de :

else if(ran==3 && casillasEsc[3]==0 || ran==3 && casillasEsc[3]!=9999){

Ahora la máquina encuentra ya todos los tomates siempre:

Figura 68: Escondite máquina encuentra los tomates

Llegados a este punto la maquina funciona perfectamente así que llego el momento de arreglar el oyente para que detecte mis clickcs.

Mirando desde donde si que me cierra el juego para ver donde si que escucha el click, y tras poner un obtener en búsqueda de jugador, y unos cuantos cambios mas, logro que me detecte los clicks pero tengo un problema con que el contador se descuenta al poner la ficha la maquina y eso que en buscar maquina no hago movJ--.Vease:

Figura 69: Escondite fallo contador

Pasa si clico en la casilla 25 ,casilla 3, el error era que en algunos casos ponía casillasEscMaquina[24]==9999 en lugar de casillasEscMaquina[24]=9999; Tras estos cambios ya funciona

Ahora tratando de poner texto de ganado y de perdido, y resulta el texto se dibuja debajo de los sprites:

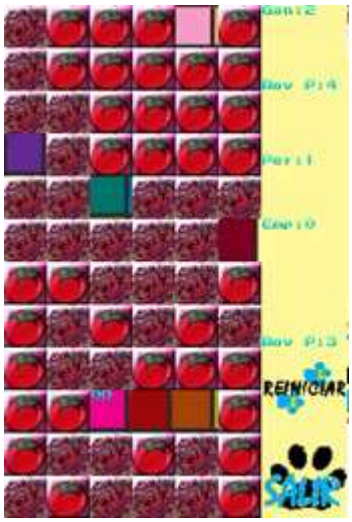


Figura 70: Escondite fallo texto

Pensándolo mucho llegue a la conclusión de que la solución más fácil y vistosa era poner unos cartelitos con el resultado:

Pensándolo mejor quedaría mejor con Sprites,

Figura 71:Carteles

Al estar hablando de la versión beta : vacío la pantalla y cargo los Sprites de "HAS GANADO " o "HAS PERDIDO ", ya que en esta versión del juego el empate no hace falta (en las versiones posteriores nivel medio y nivel difícil si hace falta).

Figura 72: Escondite has ganado

Figura 73: Has perdido

Un ejemplo sería:

```
PA_Init();
PA_EasyBgLoad(1, 3, gatoPierde);
PA_EasyBgLoad(0, 3, esconfond32usu);
…

…
…
PA_CreateSprite(0,cartel,(void*)hasGanado1_Sprite,
OBJ_SIZE_64X64,1,5,0,64);
cartel++;

PA_CreateSprite(0,cartel,(void*)hasGanado2_Sprite,
OBJ_SIZE_64X64,1,6,64,64);
cartel++;

PA_CreateSprite(0,cartel,(void*)hasGanado3_Sprite,
OBJ_SIZE_64X64,1,7,128,64);
cartel++;
```

Como que escondiera tomates me parecía un poco ilógico, para que quede mejor en lugar de esconder tomates esconderé pelotas que es mas apropiado ya que el juego es de un gato.

Y usar los tomates para cuando fallan haciendo un símil con la expresión "tomatazo" que es como decir fallo.

He usado la táctica de que uno parezca en relieve y el otro hundido, que es un clásico del buscaminas

Figura 74: Buscaminas

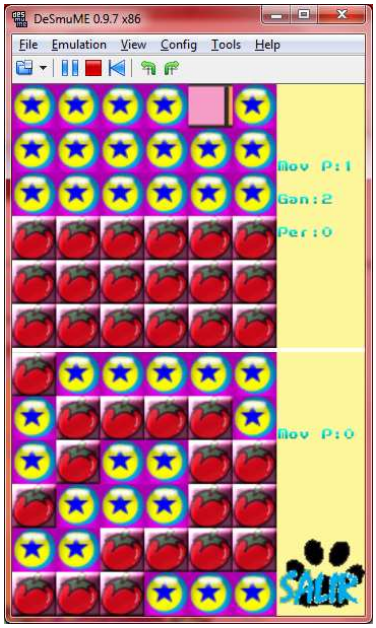


Figura 75: Escondite pelotas

Para mejorarlo ahora pongo un contador de pelotas para que el jugador sepa cuantas pelotas ha encontrado

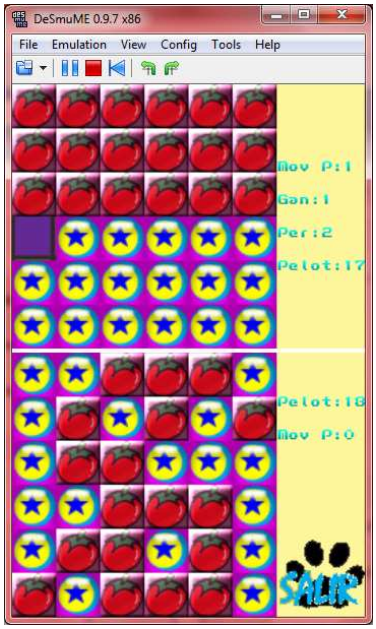


Figura 76: contador pelotas

9.4 Poinx

Una vez esta cargado el fondo y tenemos los tres sprites que necesitamos, que son collarM , collarJ y la pelota, a continuación comenzaremos con el movimiento de la bola

Primera fase hacer que se mueva "sola":

Para movimiento bola estableceré que cada dirección es un número

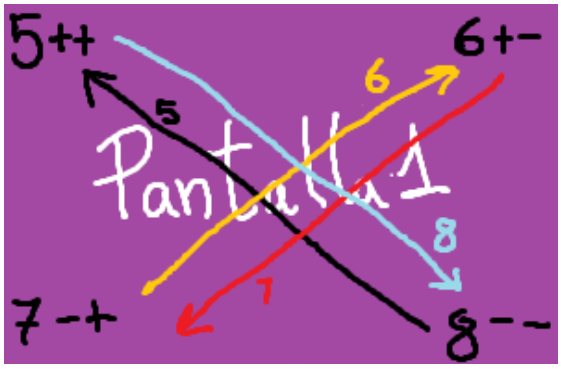


Figura 77: Direcciones bola pantalla 1

Figura 78: Direcciones bola pantalla 0

Lo siguiente es encontrar los casos posibles:

Pantalla 0:

Orientación 1 y 2:

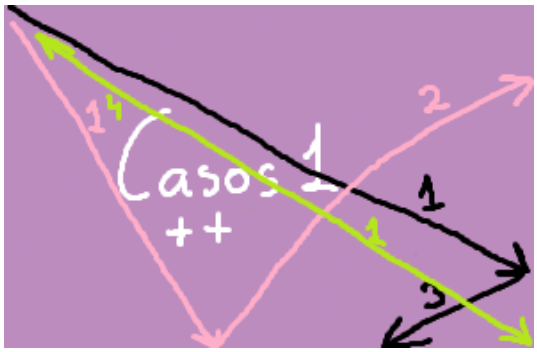


Figura 79: Caso1

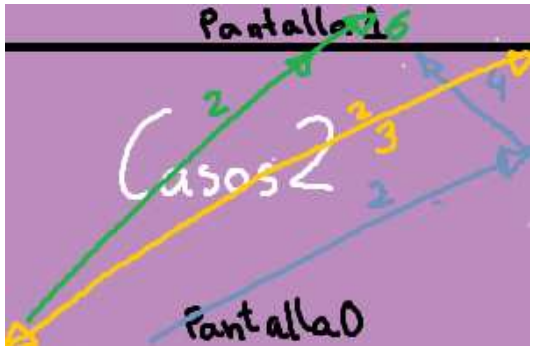


Figura 80:Caso2

Orientación 3 y 4:

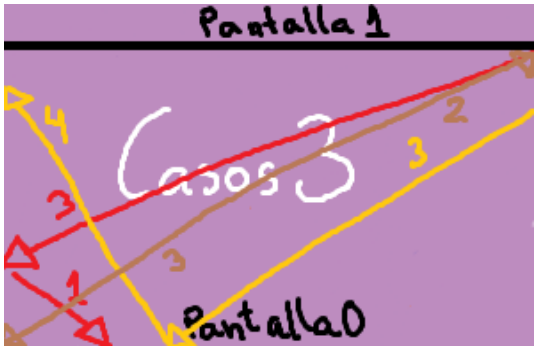


Figura 81:Caso3

Figura 82:Caso4

Pantalla 0:

Orientación 5 y 6:

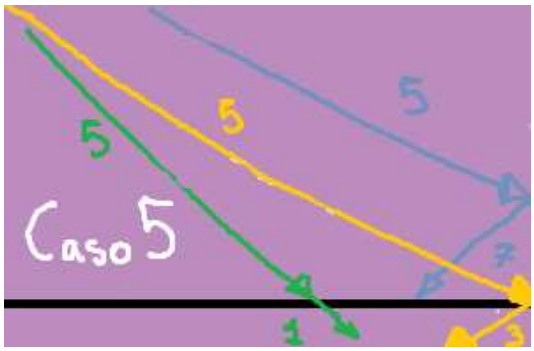


Figura 83:Caso5

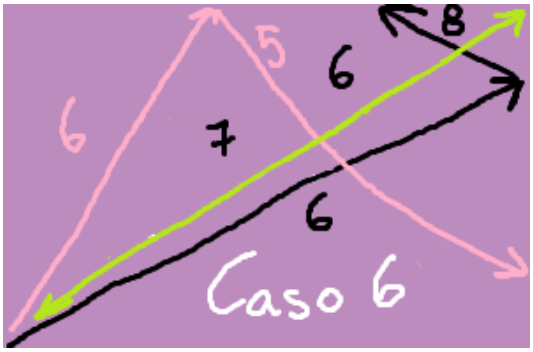


Figura 84:Caso6

Orientación 7 y 8:

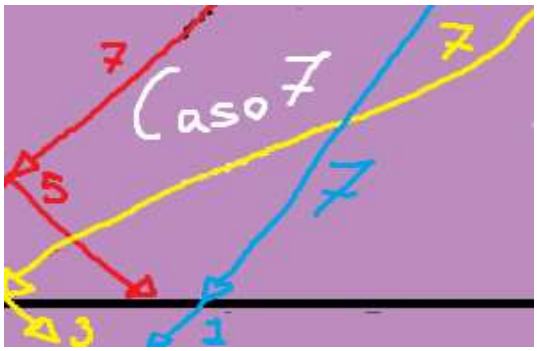


Figura 85:Caso7

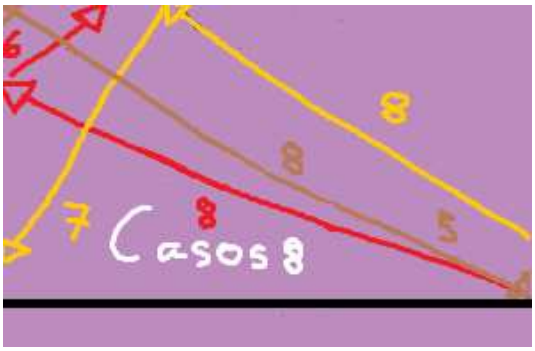


Figura 86:Caso8

Tuve muchos problemas para ajustar los choques tanto con las paredes como con los collares:

Traza de la bola:

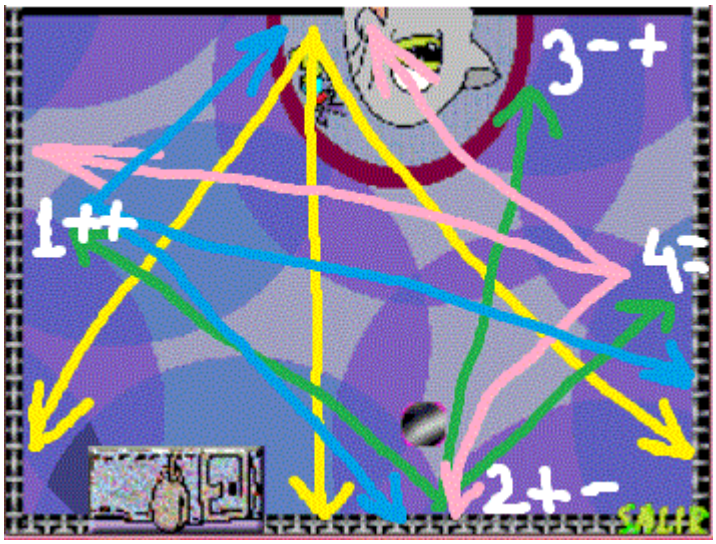


Figura 87: Traza bola pantalla0

Ahora que ya rebota la bola hay que estudiar los casos del choque con el collar:

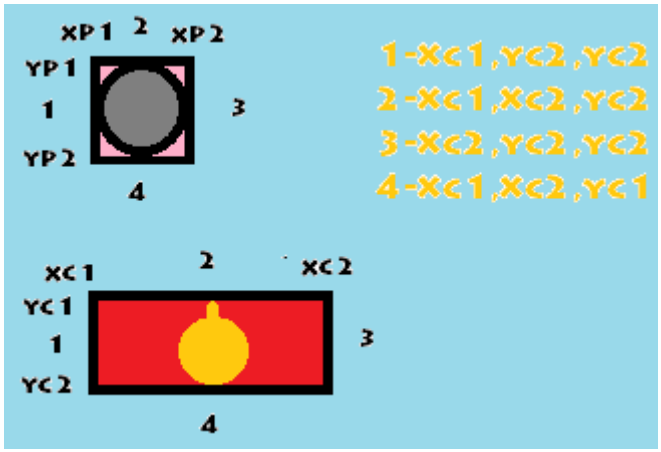


Figura 88: datos bola y collar

Hay que obtener las coordenadas del sprite collar, en este caso uso las coordenadas del stylus, por eso cuando se ejecuta el juego el centro del collar siempre esta debajo de la punta del stylus. Uso coordenadas Stylus *Stylus.X y Stylus.Y[7]*

Choque arriba:

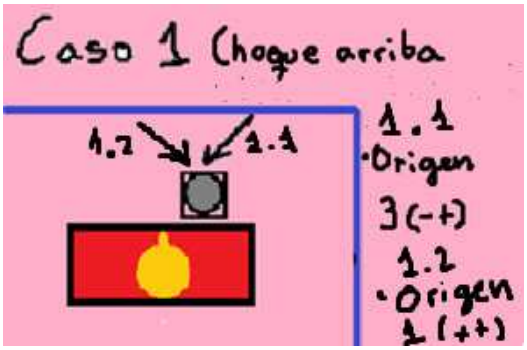


Figura 89: choque arriba1

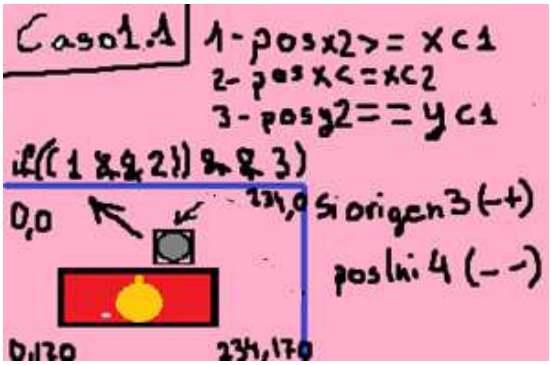


Figura 90: choque arriba1.1

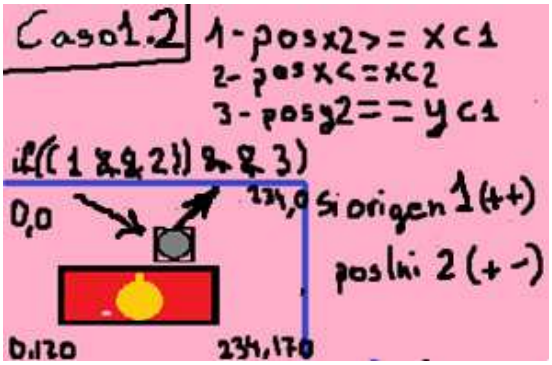


Figura 91: choque arriba1.2

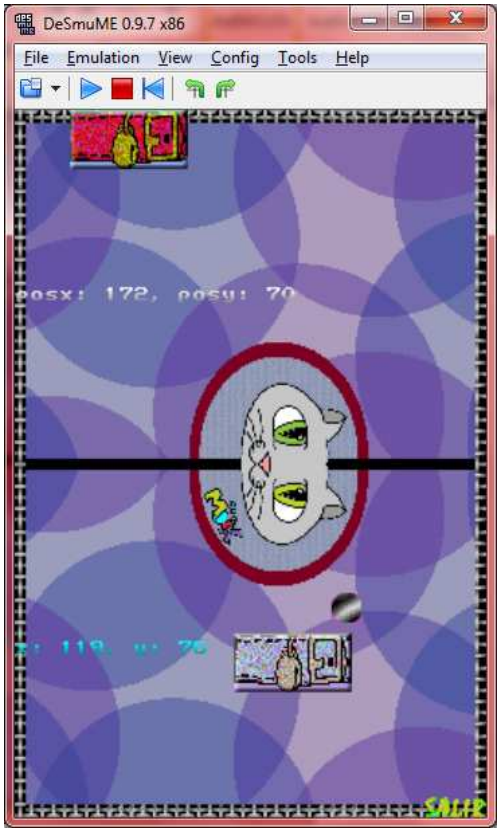


Figura 92: choque tablero1

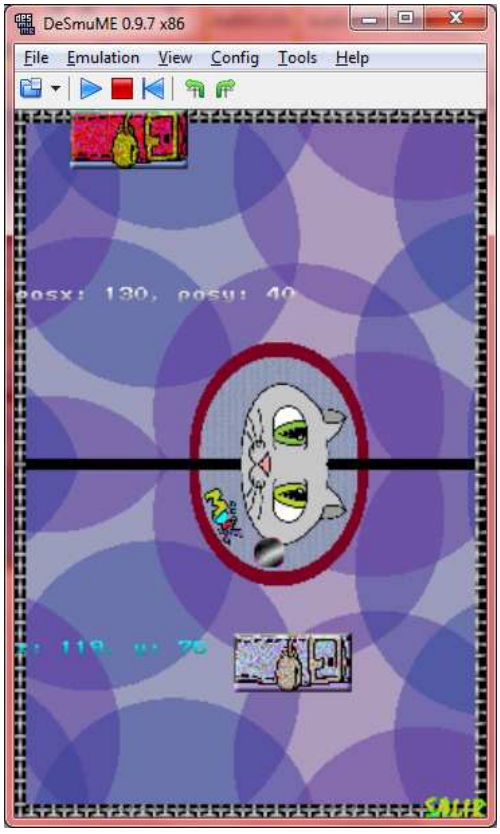


Figura 93: choque tablero2

Choque izquierda:

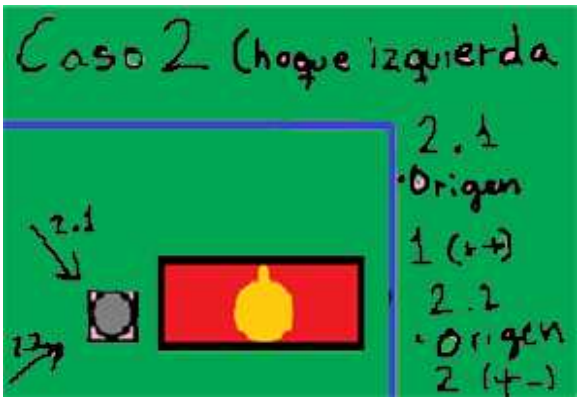


Figura 94: choque izquierda2

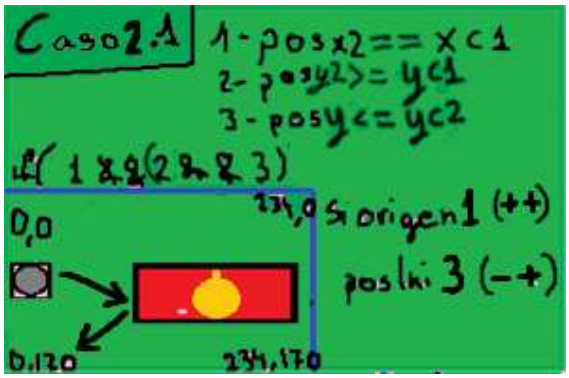


Figura 95: choque izquierda2.1

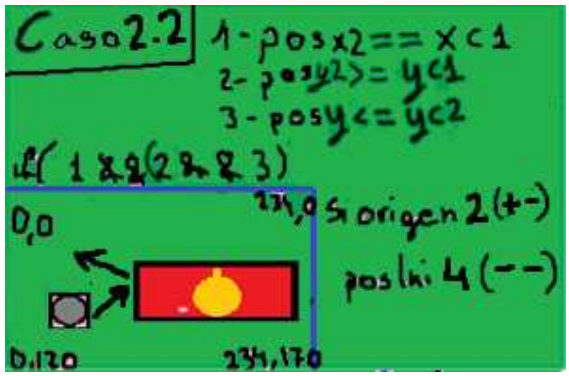


Figura 96: choque izquierda2.2

Figura 97: choque tablero3

Figura 98: choque tablero4

Choque derecha:

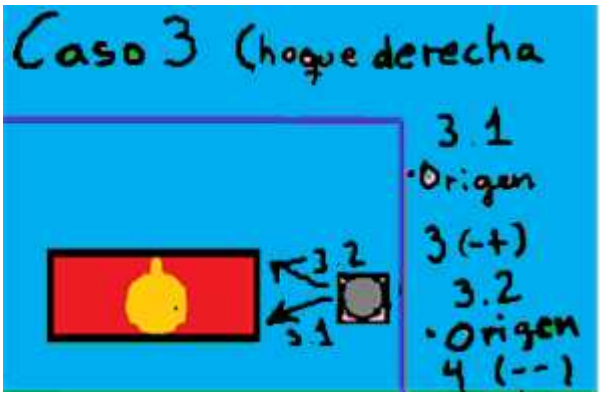


Figura 99: choque derecha3

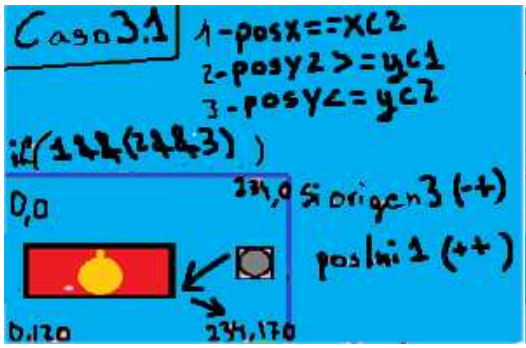


Figura 100: choque iderecha3.1

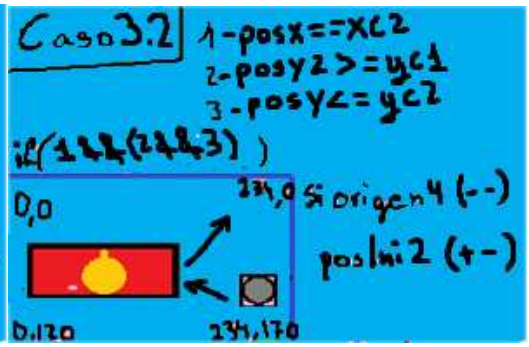


Figura 101: choque derecha3.2

Choque abajo:

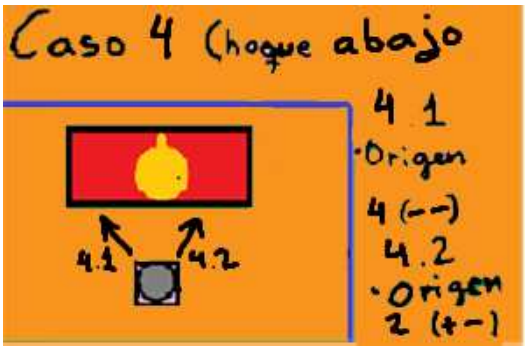


Figura 102: choque abajo4

Figura 103: choque abajo4.1

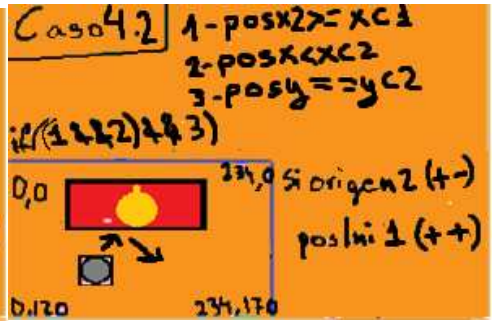


Figura 104: choque abajo4.2

Ya que el movimiento del collar del jugador depende de la posición del stylus, al arrancar el juego se detectaba que el stylus estaba arriba (coincidiendo con la posición del botón continuar) ,por lo que lo arregle de la siguiente forma:

```
if(contclickP==0){
        Stylus.X=62;
        Stylus.Y=176;
        contclickP=1;
}
```

Pero seguían habiendo fallos al chocar, por lo que aun hubo quehacer muchos mas diagramas (todos ellos en papel).

Pese a todo , hay que realizar el siguiente paso , los goles.

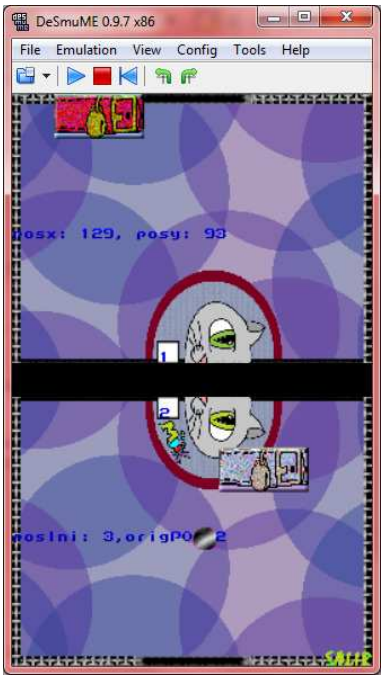


Figura 105: tablero con goles

```
//GOLES EJEMPLO
if(posy==170 &&(posx>=93 && posx2<=163)){
        contGM++;
        PA_OutputText(0,13,1,"%d",contGJ);
        PA_OutputText(1,13,23,"%d",contGM);

}
```

También con un rand%3 , consigo hacer que la bola unas veces salga hacia arriba y otras veces hacia abajo.

Ahora sale la bola unas veces hacia arriba y otras hacia abajo.

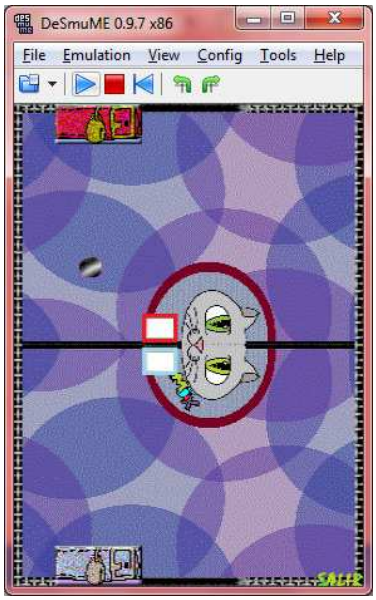


Figura 106: Bola hacia arriba

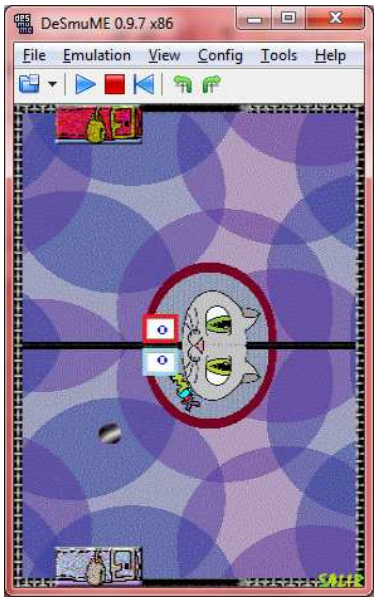


Figura 107: Bola hacia abajo

Si se produce un gol se borra bola, pausa y sale la bola en dirección al goleado. Así que si el gol lo sufre el jugador la bola sale hacia abajo y si el gol lo sufre la maquina la bola sale hacia arriba.

Es la hora de comenzar a mover la de la máquina, su inteligencia se basará en los siguientes esquemas:

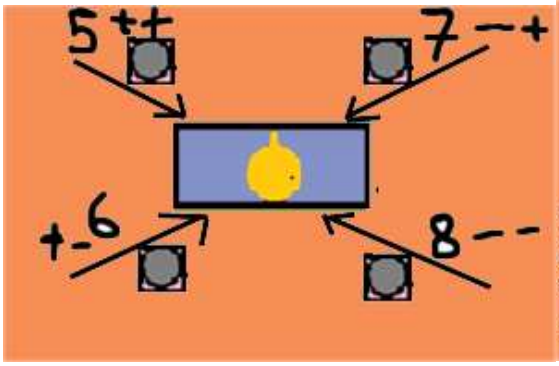


Figura 108: Choques collar máquina

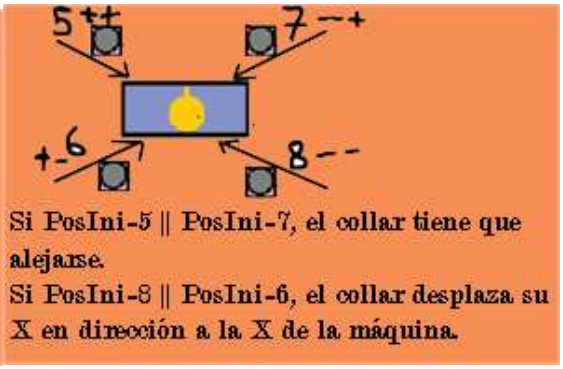


Figura 109: Casos choques collar máquina

10. Ejecución paso a paso

En este apartado voy a realizar una ejecución completa explicando como lo hago.

10.1 Arranque

La primera imagen es una captura de lo que aparece nada mas cargar el juego, esto se realiza en el main() y no hay que tocar nada es automático.

La segunda imagen es lo que aparece a posteriori, es por así decirlo el comienzo del juego, hagas donde hagas click en la pantalla va al menú principal.

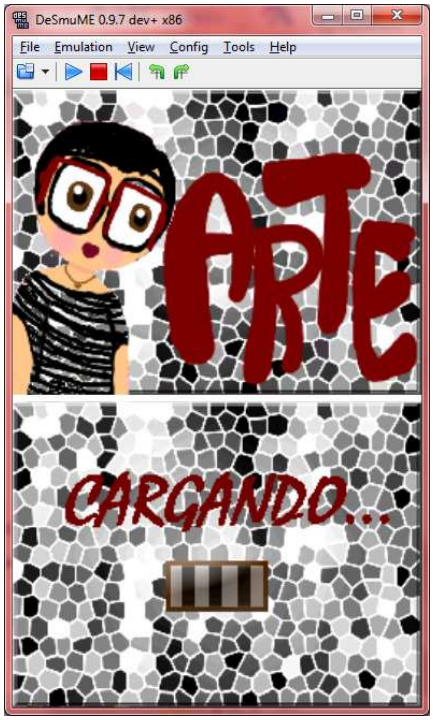


Figura 110: Carga

Figura 111: Comienzo.

10.2 Menu principal

Desde este podemos ir a los juegos, los créditos o la galería dependiendo de en que zona de la pantalla haga click.

Figura 112: Menu principal

10.2.1 Créditos

Al hacer un click sobre el botón créditos del menú principal se muestra lo siguiente:

Figura 113: Créditos

Para salir de los créditos simplemente hay que pulsar en cualquier parte de la pantalla.

10.2.2 Galería

En este apartado, lo que hacemos es cargar un fondo diferente cada vez que hagamos un click en un uno de los cuadrados con numero (tal y como se puede apreciar en las imágenes de abajo), para salir y volver al menú principal solo hay que hacer click en el botón salir.

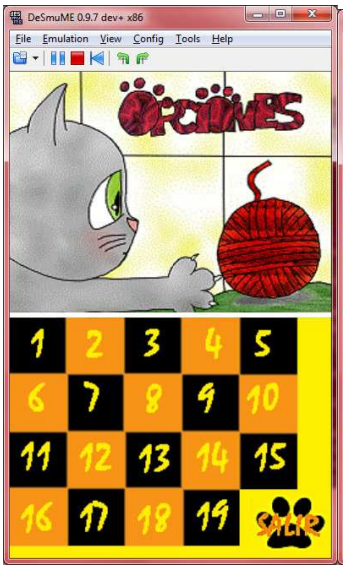


Figura 114 : Galería1

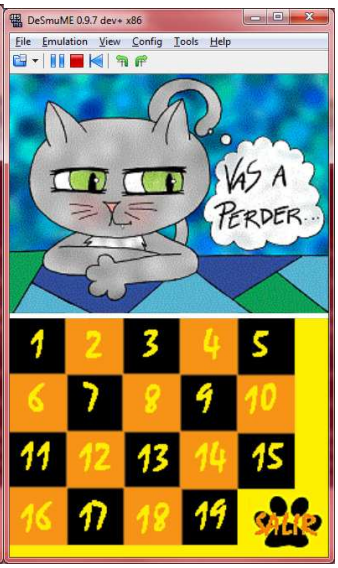


Figura 115 : Galeria2

10.2.3 Menu juegos

Este es el menú de juegos principal, de aquí podemos ir al tres en raya pulsando en el primer botón, a los puzles, salir al menú principal y por ultimo si pulsamos siguiente al segundo menú de juegos.

Figura 116: Menú juegos

10.2.3.1 Ejecución 3 en raya

Bien, una vez hacemos un click en el botón 3 en raya del menú de juegos nos aparece la imagen 117, que nos da opción a elegir a que nivel del juego queremos jugar.

Cliquemos en la opción que cliquemos siempre nos aparecerá la imagen 118, y cuando pulsemos ayuda siempre veremos la imagen 119.

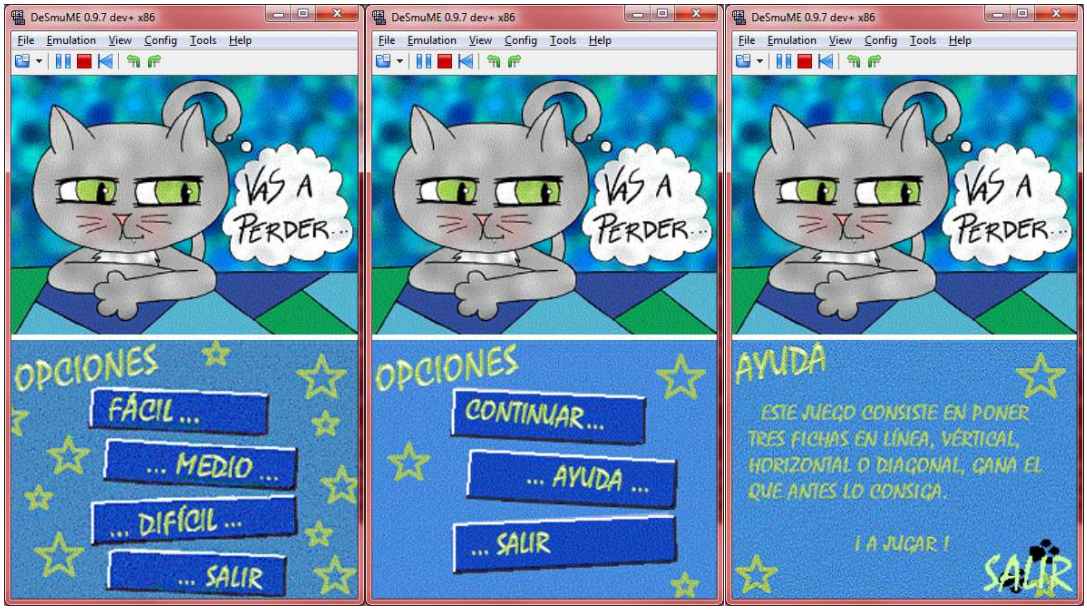


Figura 117 niveles 3 en raya *Figura 118 menú tres en raya* *Figura 119 Ayuda 3 en raya*

Pues, si pulsamos en la opción fácil nos aparecerá la imagen 120, en la cual en la pantalla inferior solo tenemos el botón de reiniciar y el de salir, si le damos a reiniciar se borraran las fichas pero, no se reseteara el contador de puntuaciones, esto solo ocurre al salir de 3 en raya fácil y volverse a meter. Pulsando salir salimos del juego.

En la imagen 121, vemos una partida empezada, pues bien la 122 es lo que pasa si en esa partida, yo hago un click sobre mi ficha situada el la posición 9 (así la selecciono), levanto el stylus y pulso en la posición 6.

Figura 120 raya fácil *Figura 121 raya fácil1* *Figura 122 raya fácil 2*

Al pulsar la opción medio nos aparecerá la imagen 123, en la cual en la pantalla inferior tenemos el botón de reiniciar , el de salir y un contador de movimientos, si le damos a reiniciar se borraran las fichas pero, no se reseteara el contador de puntuaciones, esto solo ocurre al salir de 3 en raya medio y volverse a meter. Pulsando salir salimos del juego.

En la imagen 124, vemos una partida empezada, pues bien la 125 es lo que pasa si en esa partida, yo hago un click sobre mi ficha situada el la posición 3 (así la selecciono), levanto el stylus y pulso en la posición 9, sinos fijamos en el cuadro de movimientos vemos que el numero a cambiado, cuando llegue a 10 independientemente del estado de la partida esta terminara.

Figura 123 raya medio

Figura 124 raya medio 1

Figura 125 raya medio 2

Al pulsar la opción difícil nos aparecerá la imagen 126, en la cual en la pantalla inferior tenemos el botón de reiniciar , el de salir, un contador de movimientos y un cronometro que nos muestra los segundos que nos quedan, si le damos a reiniciar se borraran las fichas pero, no se reseteara el contador de puntuaciones, esto solo ocurre al salir de 3 en raya difícil y volverse a meter. Pulsando salir salimos del juego.

En la imagen 126, vemos una partida empezada, pues bien la 127 es lo que pasa si en esa partida, yo hago un click sobre mi ficha situada el la posición 9(así la selecciono), levanto el stylus y pulso en la posición 6, si nos fijamos en el cuadro de movimientos vemos que el numero a cambiado, cuando llegue a 10 independientemente del estado de la partida esta terminara, pero no solo esto, sino que también a cambiado el numero del cronometro, cuando este sea 0 la partida finalizara también.

Figura 126: raya difícil

Figura 127: raya dificil1

Figura 128: raya dificil2

Figura 129: raya difícil 3

10.2.3.2 Ejecución puzle

Si la opción del menú juegos que hemos pulsado es puzle, aparecerá la figura 130, que nos da a elegir si pasar al juego(continuar), ver la ayuda o salir.

Si pulsamos ayuda se muestra la figura 131, si le damos a continuar aparece la figura 132, bien como podemos apreciar esta tiene el botón siguiente, reiniciar y salir, si pulso siguiente cargara un nuevo puzle(imágenes 133,134,135), si doy a reiniciar las piezas se colocaran en otra posición y pueden no salir las mismas piezas, si doy a salir salgo del juego.

Si estamos en el cuarto puzle y doy a siguiente volvemos al puzle 0.

Figura 130: menú puzle

Figura 131: ayuda puzle

Figura 132: puzle 1

Figura 133: puzle 2

Figura 134: puzle 3

Figura 13:5 puzle 4

10.2.3.3 Menu juegos2

A través de este menú podemos ir a el juego escondite, al poinx , parejas y volver al menú juegos uno.

Figura 136: Menú juegos 2

10.2.3.3.1 Escondite

Bien, una vez hacemos un click en el botón escondite del menú de juegos dos nos aparece la imagen 137, que nos da opción a elegir a que nivel del juego queremos jugar.

Cliquemos en la opción que cliquemos siempre nos aparecerá la imagen 138, y cuando pulsemos ayuda siempre veremos la imagen 139.

Figura 137: niveles escondite

Figura 138: Menú escondite

Figura 139: Ayuda escondite

Pues, si pulsamos en la opción fácil nos aparecerá la imagen 140, donde nos pide que escondamos nuestras pelotas, una vez hecho esto la máquina pone las suyas y como estamos en la opción fácil podemos verlas durante un corto periodo de tiempo.

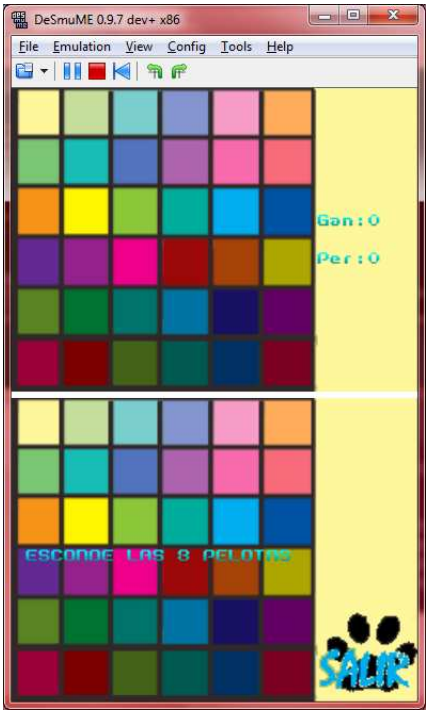


Figura 140: escondite fácil

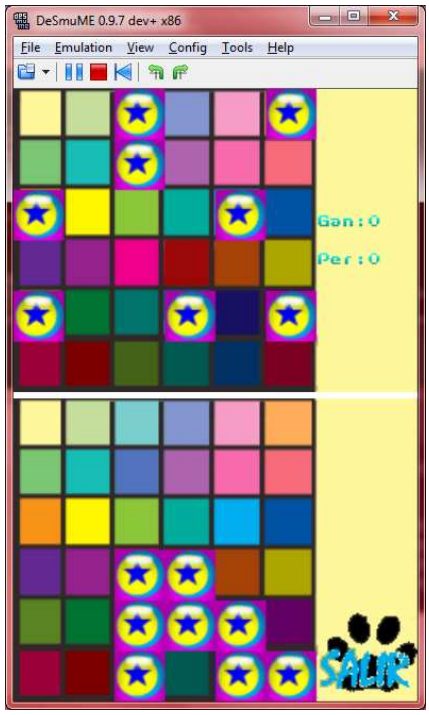
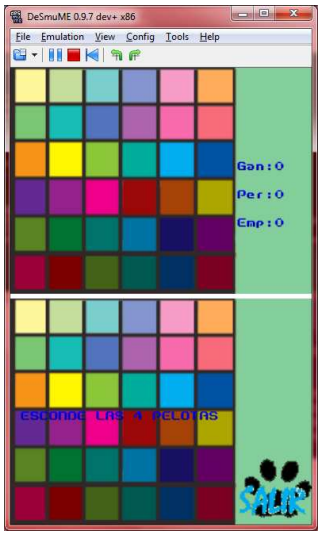


Figura 141: escondite fácil2

Figura 142: escondite fácil3

Figura 143: escondite fácil resultado

Pues, si pulsamos en la opción medio nos aparecerá la imagen 144, donde nos pide que escondamos nuestras pelotas, una vez hecho esto la máquina pone las suyas y ya no podemos ver las fichas que ha puesto la maquina durante un corto periodo de tiempo.



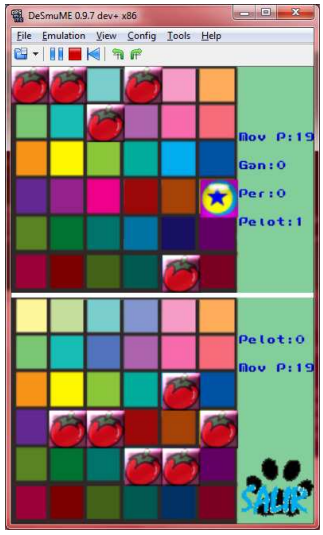
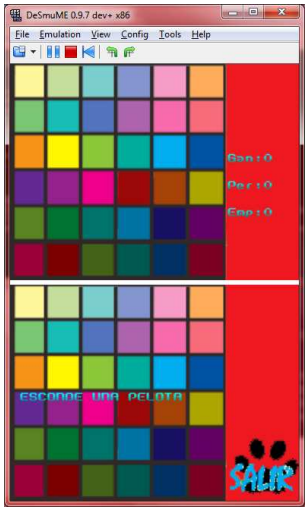


Figura 144: escondite medio *Figura 145: escondite medio 2* *Figura 146: escondite medio 3*

Pues, si pulsamos en la opción difícil nos aparecerá la imagen 147, donde nos pide que escondamos nuestras pelotas, una vez hecho esto la máquina pone las suyas y ya no podemos ver las fichas que ha puesto la maquina durante un corto periodo de tiempo, además de que el numero de movimientos se ha reducido bastante solo tenemos que encontrar y buscar una bola.



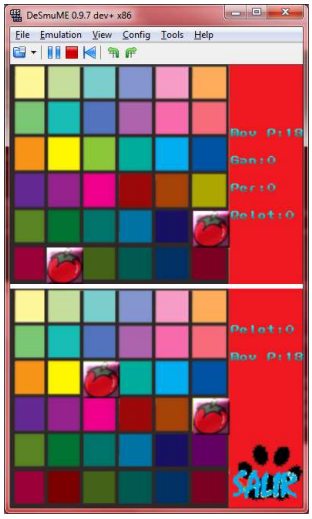


Figura 147: escondite difícil *Figura 148: escondite dificil2* *Figura 149: escondite dificil3*

10.2.3.3.2 Poinx

Bien, una vez hacemos un click en el botón poinx del menú de juegos dos nos aparece la imagen 150, que nos da opción a elegir a que nivel del juego queremos jugar.

Cliquemos en la opción que cliquemos siempre nos aparecerá la imagen 151, y cuando pulsemos ayuda siempre veremos la imagen 152.

Figura 150: menú niv poinx

Figura 151: menú poinx

Figura 152: ayuda poinx

Pues, si pulsamos en la opción fácil nos aparecerá la imagen 153, cuando se mete un gol aparece un cartel y los contadores se incrementan como vemos en la imagen 154.

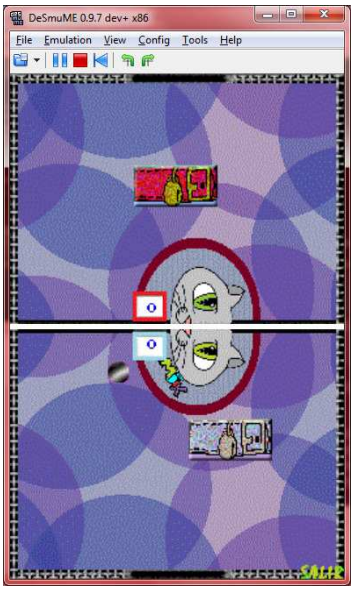


Figura 153: poinx fácil

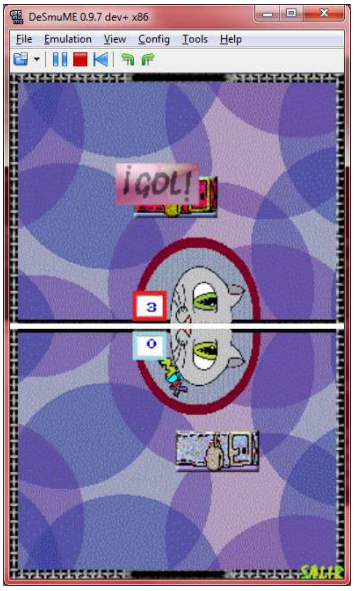


Figura 154: poinx fácil2

Pues, si pulsamos en la opción medio nos aparecerá la imagen 155,cuando se mete un gol aparece un cartel y los contadores se incrementa como vemos en la imagen 156. Pero si nos finamos vamos vemos que nuestra portería a crecido, y aunque en la imagen no se vea tanto la máquina como la bola van mas rápido.

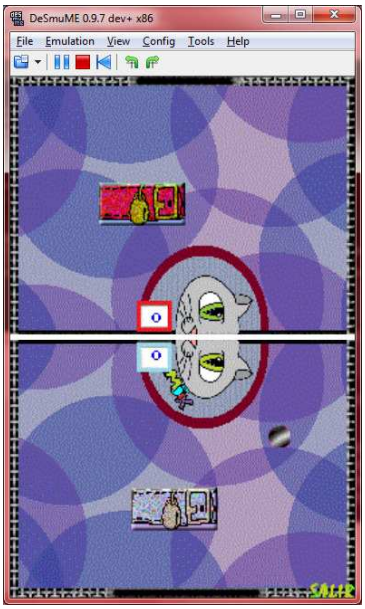


Figura 155: poinx medio

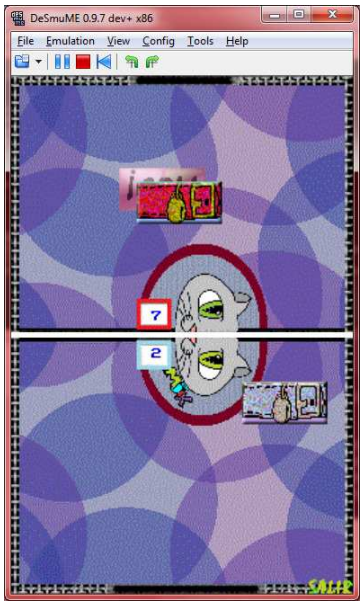


Figura 156: poinx medio2

Pues, si pulsamos en la opción difícil nos aparecerá la imagen 157,cuando se mete un gol aparece un cartel y los contadores se incrementa como vemos en la imagen 158. Pero si nos finamos vamos vemos que nuestra portería a crecido, y aunque en la imagen no se vea tanto la máquina como la bola van mas rápido.

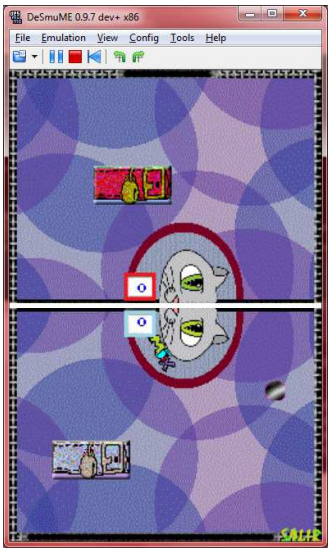


Figura 157: poinx dificil1

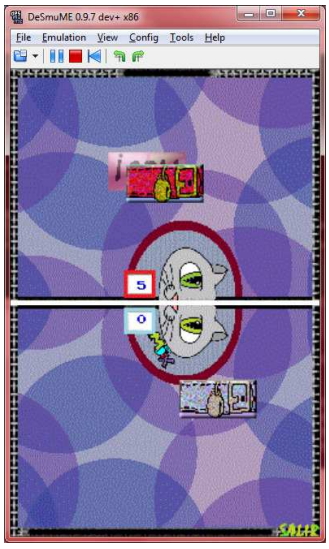


Figura 158: poinx difícil 2

10.2.3.3.3 Parejas

Bien, una vez hacemos un click en el botón parejas del menú de juegos dos nos aparece la imagen 159, que nos da opción a elegir a que nivel del juego queremos jugar.

Cliquemos en la opción que cliquemos siempre nos aparecerá la imagen 160, y cuando pulsemos ayuda siempre veremos la imagen 161.

Figura 159: menú niv parejas

Figura 160: menú parejas

Figura 161: ayuda parejas

Pues, si pulsamos en la opción fácil nos aparecerá la imagen 162,en la imagen 163 vemos un ejemplo de girar dos cartas, y en la 164 el tablero resuelto.

Figura 162: parejas fácil1

Figura 163: parejas fácil 2

Figura 164: parejas fácil3

Pues, si pulsamos en la opción medio nos aparecerá la imagen 165 donde podemos ver que ya hay una pareja mas de cartas ,en la imagen 166 vemos un ejemplo de girar dos cartas, y en la 167 el tablero resuelto.

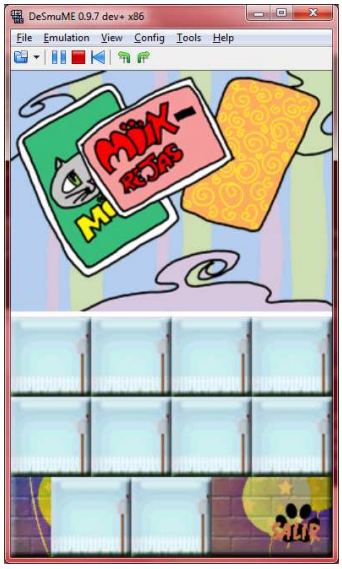


Figura 165: parejas medio 1

Figura 166: parejas medio2

Figura 167: parejas medio 3

Pues, si pulsamos en la opción difícil nos aparecerá la imagen 168 donde podemos ver que ya hay el mismo numero de cartas que en el medio ,en la imagen 169 vemos un ejemplo de girar cartas, y en la 170 el tablero si perdemos, ya que en el nivel difícil hay un contador de movimientos, si llegas al limite pierdes y se reinicia la partida.

Figura 168: parejas difícil 1

Figura 169: parejas difícil 2

Figura 170: parejas difícil 3

11. CONCLUSIÓN

Ha sido un trabajo muy interesante, difícil, ya que muchas veces me encontraba limitada, como si no tuviera suficientes conocimientos, lo mas difícil es decidir el como hacer las cosas, en algunos momentos no me veía capaz pero me ha enseñado mucho (al menos eso creo).

En definitiva creo que se asemeja bastante a lo que quería hacer y lo que mas me ha llamado la atención ha sido ver una aplicación que funciona en algo que no es un ordenador y que la he hecho yo, aunque en algunos momentos creía que no lo lograría.

Creo que lo que mas me ha gustado es ver como otras personas jugaban con el sin que supieran que lo había hecho yo y ver sus caras cuando se lo dije.

Quiero también darle las gracias a Manolo por permitirme hacer algo así y aceptar ser mi director de proyecto para la ampliación de este trabajo, y que gracias a esto MOIXGAMES a crecido ya mucho desde su comienzo y aun crecerá mas.

De lo que establecimos como objetivos en un principio no he realizado la carga de imágenes desde la consola y no encontré como, leyendo el trabajo de "miguelDavid_DSiPod" he visto que ellos también querían cargar las imágenes y tampoco pudieron.

En contraposición he de decir, que según las solicitud solo tenia que hacer niveles en el tres en raya y en el juego de las parejas, al ver que no podía hacerla carga de fotos, hice los niveles en el escondite y en el poinx.

Este ha sido un trabajo muy gratificante y que me ha hecho aprender mucho, y verme capaz de mas de lo que yo creía.

12. BIBLIOGRAFíA

[0] http://nds.scenebeta.com/tutorial/definiciones-del-mundo-nds

[1]http://www.topproducerwebsite.com/users/10749/downloads/Manual.pdf?id=0.9605311

[2] http://nds.scenebeta.com/node/16789

[3] http://nds.scenebeta.com/node/5738

[4] http://www.elotrolado.net/hilo_el-misterioso-caso-de-los-sprites-peta-memorias_1142706_s10

[5] http://sourceforge.net/projects/desmume/files/

[6] http://palib-dev.com/

[7] http://www.nightfoxandco.com/forum/index.php?topic=224.0

[8] http://palib.info/wiki/doku.php?id=day1es

[9] http://www.neoteo.com/tutorial-como-hacer-temas-para-nintendo-ds-lite

[10] http://simianzombie.com/?tag=palib

[11]http://nds.scenebeta.com/tutorial/indexando-im%C3%A1genes-para-el-pagfx

[12] http://nds.scenebeta.com/tutorial/ponte-al-dia-en-10-minutos-la-scene-nds

[13] http://www.mygnet.net/codigos/csharp/juegos/juego_de_puzzle.28

13

[14]http://foro.elhacker.net/programacion_cc/puzzle_en_dev_c-t311129.0.html

[15]http://www.elrincondelc.com/nuevorincon/index.php?pag=codigos&id=20

[16]http://nds.scenebeta.com/tutorial/indexando-im%C3%A1genes-para-el-pagfx

[17] http://nds.scenebeta.com/node/13789

[18] http://www.elotrolado.net/hilo_problema-pagfx_1356571

[19] http://nds.scenebeta.com/tutorial/programaci-n-tipos-de-variables-y-primer-contacto-con-las-funciones-palib

[20] http://www.elotrolado.net/hilo_programacion-en-nds-recopilacion-de-tutoriales-palib-y-voc_905471_s180

[21] http://devnintendods.blogspot.com/2009/03/nueva-version-del-tetris.html

[22] http://raultecnologia.wordpress.com/nds/16

[23] http://devnintendods.blogspot.com/2008/03/cargar-fondos.html

[24] http://devnintendods.blogspot.com/search/label/PALib

[25] http://sites.google.com/site/devnintendods/Home/tetris

[26] http://mundogeek.net/archivos/2007/01/17/desarrollo-con-cc-en-eclipse/

[27] http://www.neverbyte.net/?f

[28] http://todo-redes.com/

[29] http://blogs.gamefilia.com/knightfox/20-03-2009/20618/tutorial-de-programacion-de-nintendo-ds-dia-1-carga-de-fondos-desde-la-fa

[30] http://nds.scenebeta.com/tutorial/creando-un-icono-para-nuestros-homebrews-o-juegos-comerciales

[31] http://nds.scenebeta.com/noticia/romer

[32] http://www.elotrolado.net/hilo_nds-programacion-con-palib-4-texto-color-tamano-y-fuentes_901046

[33] http://desmume.softonic.com/linux

[34] http://sourceforge.net/projects/devkitpro/

[35] http://palib-dev.com/

[36] http://snipah.com/

[37] http://sourceforge.net/projects/desmume/files/

[38] Miguel A.Del Agua Teba y David Fresneda Bayarri (2011).“DSiPod”.Trabajo presentado para IMD. Consultado para la realización del trabajo de IMD en el curso 2011/2012. http://www.superbob.net63.net/archivos

COMO CREAR UN CONTADOR PARA NDS

[39] http://www.elotrolado.net/hilo_Homebrew-DS_591847

[40] http://www.elotrolado.net/hilo_nds-programacion-con-palib-10-menus-funciones-y-archivo_1018696

[41]http://www.elotrolado.net/hilo_duda-palib-el-texto-se-me-crea-infinitamente_1272397

PAUSAR AL CERRAR LA CONSOLA

[41] http://nds.scenebeta.com/node/12583

[42] http://nds.scenebeta.com/node/4635

Falsa solución va descontando pero esta bloqueada la pantalla

TUTORIALES

[43] http://www.espalteam.com/foros/showthread.php?t=182

Tutorial instalación y configuración del codeblocks

[44] http://ccpssolutions.com/nogdusforums/index.php?topic=630.0

[45] Cambio de nombre de pelotas a escondite, cambio nombre de pong a poinx

PARA MOVER EL PALO DEL PONG

[45] http://www.elotrolado.net/hilo_problema-con-un-codigo-de-palib-necesito-ayuda_1265444

[46] Iván López Rodríguez (2011). “Guía de desarrollo para Nintendo DS”. Proyecto final de carrera. Consultado para la realización del trabajo de IMD en el curso 2011/2012.

[47]María del Carmen Sarriá Trejo (2012). "MoixGames". Mi trabajo de IMD. Realizado en el curso 2011/2012.