UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Introducción al desarrollo con SDL 1.2 para plataforma 3DS. Uso de imágenes

Apellidos, nombre	**Agustí i Melchor, Manuel** (magusti@disca.upv.es)
Departamento	**Departamento de Informática de Sistemas y Computadores (DISCA)**
Centro	Universitat Politècnica de València

1 Resumen de las ideas clave

Simple DirectMedia Layer (SDL, la Figura 1a muestra el logotipo) [1, 2] es una biblioteca de funciones de bajo consumo de recursos, con versiones para la mayoría de lenguajes de programación existentes y distribuida bajo licencia *zlib*. Fue creada por Sam Lantinga en 1998, mientras trabajaba en *Loki Software*, para realizar operaciones de carácter multimedia y multiplataforma. Fue diseñada para proporcionar acceso de bajo nivel a audio, entrada/salida (gestión de eventos producidos por teclado, ratón, *joystick*, etc. y acceso a ficheros), imágenes (en 2D y con soporte hardware para 3D como render de OpenGL), uso de temporizadores y sincronización basada en hilos. SDL está escrita en C, que ofrece un nivel común para diferentes SO y conjuga operaciones de bajo nivel con un alto nivel de portabilidad que facilitan el desarrollo multiplataforma, delegando en las interfaces nativas la implementación final de esas operaciones (Figura 1b).

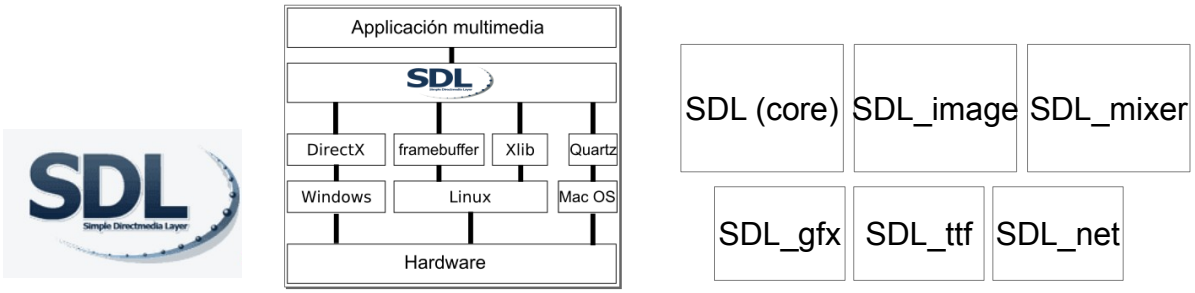


Figura 1: SDL: (a) Logotipo de SDL, (b) Niveles de capa de abstracción del SO sobre el que se ejecuta una aplicación multiplataforma y (c) Módulos de SDL: principal y extensiones oficiales. Imágenes de <https://gbatemp.net/threads/release-sdl-3ds-1-2-15-simple-directmedia-layer-for-3ds.459291/> y <http://easy-learn-computer.blogspot.com/2012/01/learning-sdl-simple-directmedia-layer.html>.

En este sentido, en cuanto a plataformas existentes hoy en día, se puede hablar de tres grupos bastante diferentes: el computador de escritorio (véase la Figura 1b) sobre los SO Linux, macOS o Windows (y su correspondientes subsistemas gráficos X11, Quartz o DirectX respectivamente), las videoconsolas (con un SO generalmente propietario y cerrado) o los *smartphone (bajo Android, iOS, etc.)*.

Entre las plataformas soportadas[1], en la versión oficial, se encuentran las correspondientes al computador de escritorio (bajo Linux, macOS, Windows y otras variantes de Unix). Y también de manera no oficial, esto es realizadas por terceras partes, existen versiones para otras plataformas como videoconsolas y, en particular para **Nintendo 3DS** (N3DS o 3DS para abreviar) y Nintendo ***Switch*** (o, simplemente, *Switch*) entre otras. SDL, actualmente, tiene dos grandes ramas y una tercera en camino[2]: SDL 1.2 (que está presente en el SDK no oficial para 3DS) y SDL 2.0 (que lo está en el de *Switch*).

En el momento de desarrollar una aplicación de características multimedia, como un videojuego, suele plantearse el tema de su **portabilidad**, esto es, si es posible reconstruir la aplicación en más de una plataforma de computador. SDL ofrece un camino interesante para resolver esta cuestión. El uso de la biblioteca **SDL,** permite realizar una versión portable, con **cambios** mínimos, al

[1] La lista actual completa se puede ver en "SDL Wiki | Installing SDL" en la URL <https://wiki.libsdl.org/SDL2/Installation>.

[2] Puede leer sobre esta futura versión en el sitio de Github: *Simple DirectMedia Layer (SDL) Version 3.0* <https://github.com/libsdl-org/SDL>.

ofrecer un nivel de abstracción común por encima de estas plataformas. Para ello, como muestra la Figura 1b fue diseñada como un API [3] a un nivel de abstracción más alto que el acceso directo a la combinación de hardware y sistema gráfico disponible en cada plataforma. Con el tiempo, nuevas plataformas han ido siendo añadidas a este abanico que puede abordar SDL.

SDL se compone de diferentes módulos (Figura 1c), siendo la "standard library" el **módulo básico** (*core/kernel*) que se encarga de la gestión del modo de vídeo y los eventos de teclado y ratón. Sobre este pueden utilizarse otros módulos (o extensiones oficiales) como *SDL_image* (para soporte de formatos gráficos), *SDL_mixer* (formatos de audio, reproducción y mezcla), *SDL_net* (operaciones de transmisión en red) o *SDL_ttf* (uso de tipos de letra *TrueType*). En este artículo nos vamos a centrar en el soporte de SDL a las imágenes por parte de los módulos **principal (*core) y SDL_image sobre la plataforma N3DS*.**

2 Objetivos

Una vez que el lector haya revisado este artículo con detenimiento y explore el código que se adjunta, dispondrá de una referencia para aplicar en el caso de abordar el desarrollo sobre la videoconsola N3DS utilizando la biblioteca SDL. En particular, será capaz de:

- Describir qué es SDL y su situación respecto al uso de las imágenes en la plataforma 3DS con SDL *core*.
- Explorar ejemplos que permitan experimentar con la funcionalidad del módulo *SDL_image* de SDL en el contexto de la plataforma N3DS.

Para no extender la longitud del artículo se ha creado un repositorio en GitHub [4] donde se alojarán los ejemplos de código que se referencian es este artículo. El resto de este documento se centrará en describir las características básicas de SDL respecto al uso de gráficos e imágenes y aplicadas a la plataforma 3DS y en explorar ejemplos de uso aplicados a los dos niveles que ofrecen los módulos SDL *core* y *SDL_image*.

3 Introducción

La disponibilidad de SDL estable, la encontramos en el conjunto de librerías PORTLIBS distribuidas con *devkitPro* [5] para 3DS. en la versión 1.2.15, aunque no incluye el módulo de red (*SDL_net*). La versión SDL 1.2 está descontinuada oficialmente por parte de SDL, pero se ha liberado el código fuente y se sigue manteniendo de forma externa a la propia SDL [4], quienes todavía mantienen la documentación, pero la disponibilidad de ejemplos en su sitio web [1] es muy baja. Como, en particular, sobre la plataforma 3DS, no hay ejemplos portados para poderlos tomar de referencia [6], es interesante disponer de un conjunto de ejemplos que aborden la funcionalidad de SDL 1.2. Antes de entrar a ver esas implementaciones, nos vamos a detener a explicar cómo funciona el sistema de dibujado en SDL y a relacionarlo los nombres de las estructuras de datos y funciones más destacadas.

En SDL 1.2 el dibujado en pantalla se realiza sobre una abstracción: la SDL_*Surface*. Este elemento es la implementación del concepto de *framebuffer* o área de memoria asociada a la pantalla: es la memoria de vídeo donde se almacenará el contenido gráfico a mostrar. La inicialización del modo de vídeo (con *SDL_SetVideoMode*, *véase la Figura 2*) de SDL 1.2 devolverá esta referencia a la pantalla, que siempre se refiere a la pantalla completa, no se puede especificar un área o ventana.

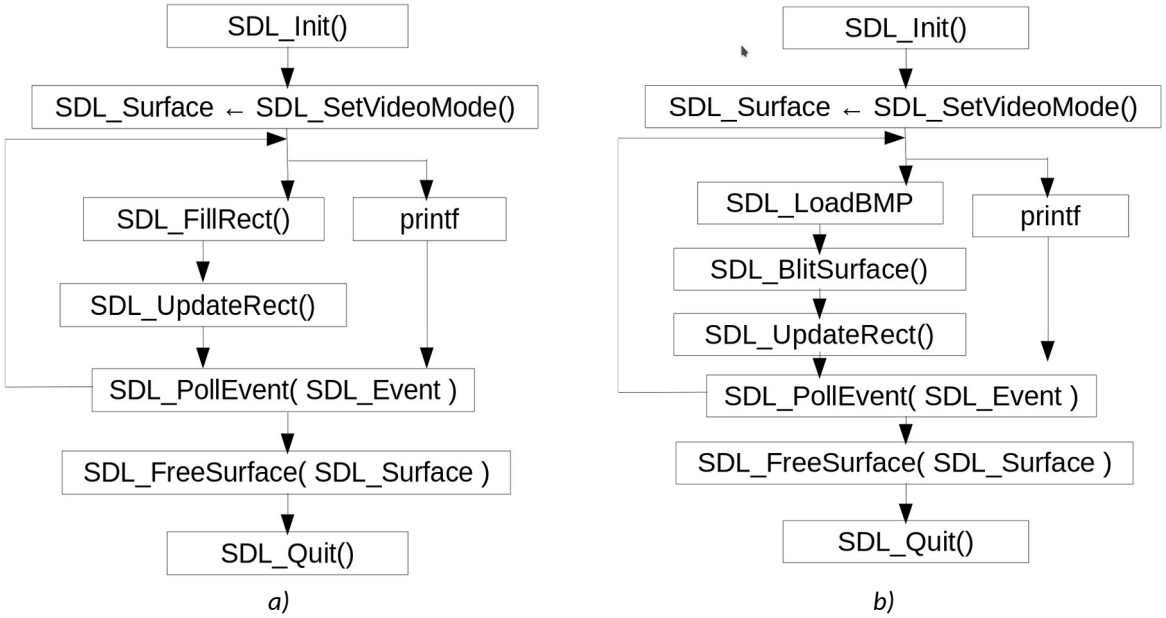


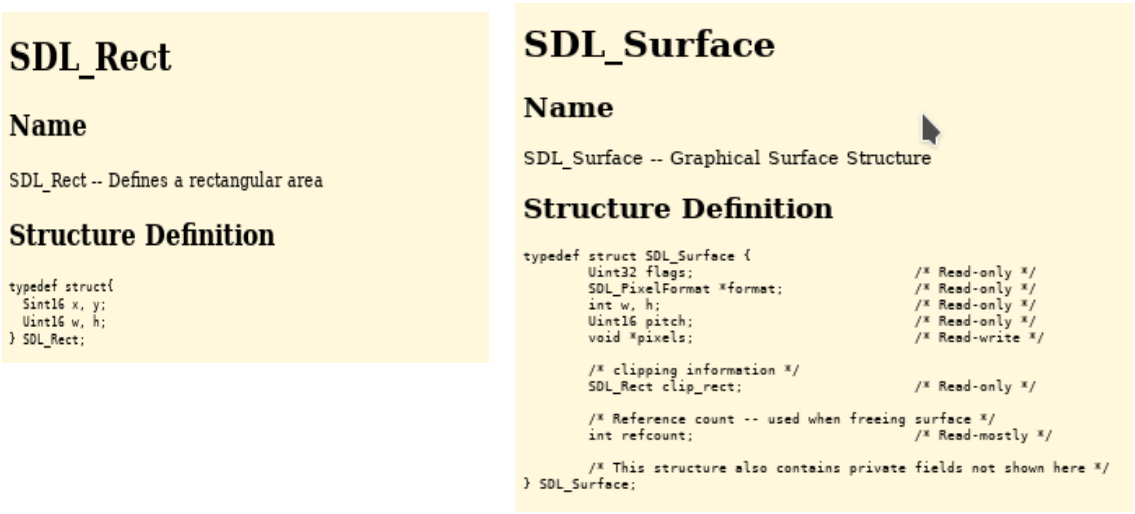
Figura 2: Secuencia de pasos para dibujar en utilizando SDL (core) en modo gráfico y en modo texto: (a) rectángulos y (b) imágenes.

La Figura 2 muestra de forma esquemática la secuencia típica de trabajo en SDL tanto en modo gráfico como en modo texto (representado por la instrucción *printf*). El paso previo es igual en ambos casos. El bloque central es diferente si se utiliza el acceso en modo rectángulo o a través de una superficie. Posteriormente se comprueban las acciones de usuario y, al terminar la aplicación, se han de liberar los recursos, lo que implica liberar la superficie de la pantalla y al resto de "superficies" que se hubieran creado al acceder a imágenes. Es en ese bloque central donde se ve la diferencia entre los modos de dibujado. Podemos realizar operaciones de dibujado a tres niveles:

- Píxel a píxel. En este modo de acceso directo se puede cambiar el contenido en pantalla[3], modificando individualmente cada uno de los valores del *framebuffer* (la *Surface* que define toda la pantalla) directamente.
- Rectángulos (que corresponde al tipo *SDL_Rect* definido en la Figura 3a). Para dibujar los rectángulos, véase Figura 2a, se especifica un área plana por sus coordenadas iniciales (superior izquierda), ancho y alto. Esta zona se ha de explicitar cómo se ha de pintar, indicando el color a utilizar para pintar todos los píxeles de la misma con *SDL_FillRect*. Tras lo cual hemos de solicitar que se actualice el contenido de esa zona en pantalla con *SDL_UpdateRect*.
- Imagen (que corresponde al tipo *SDL_Surface* definido en la *Figura 3b*). Cuando no es homogéneo el color de un área se utiliza una imagen en mapa de bits para definir el contenido de un área rectangular. Por compatibilidad con el modo de acceso a la memoria de vídeo, SDL 1.2 utiliza también una *SDL_Surface* para almacenar el contenido de las imágenes y se dibuja con *SDL_BlitSurface*, actualizando directamente una parte del contenido en pantalla, un área rectangular de NxM píxeles, descartando lo que se sale de la "pantalla". La Figura 2b muestra la secuencia de dibujado típica en ese caso, cada imagen que se defina en la aplicación se indicará en qué posición de la pantalla (la *surface* definida con *SDL_SetVideoMode*) se "pinta", en realidad se copia (puesto que ambas son del mismo tipo). A esta operación se la denomina *bliting*[4]. Después de lo cual, cuando se quiera que se haga visible (quizás se quiere pintar alguna cosa más antes) hay que indicar que se refresque la pantalla con

[3] Aunque es poco habitual por la carga de trabajo que supone, pero si tiene interés en ello puede consultar las funciones getpixel y putpixel de la documentación de SDL [3].

SDL_UpdateRect[5] (toda, si las coordenadas son todas cero o, parcialmente, en caso contrario).



SDL_Rect

Name

SDL_Rect -- Defines a rectangular area

Structure Definition

```
typedef struct{
  Sint16 x, y;
  Uint16 w, h;
} SDL_Rect;
```

SDL_Surface

Name

SDL_Surface -- Graphical Surface Structure

Structure Definition

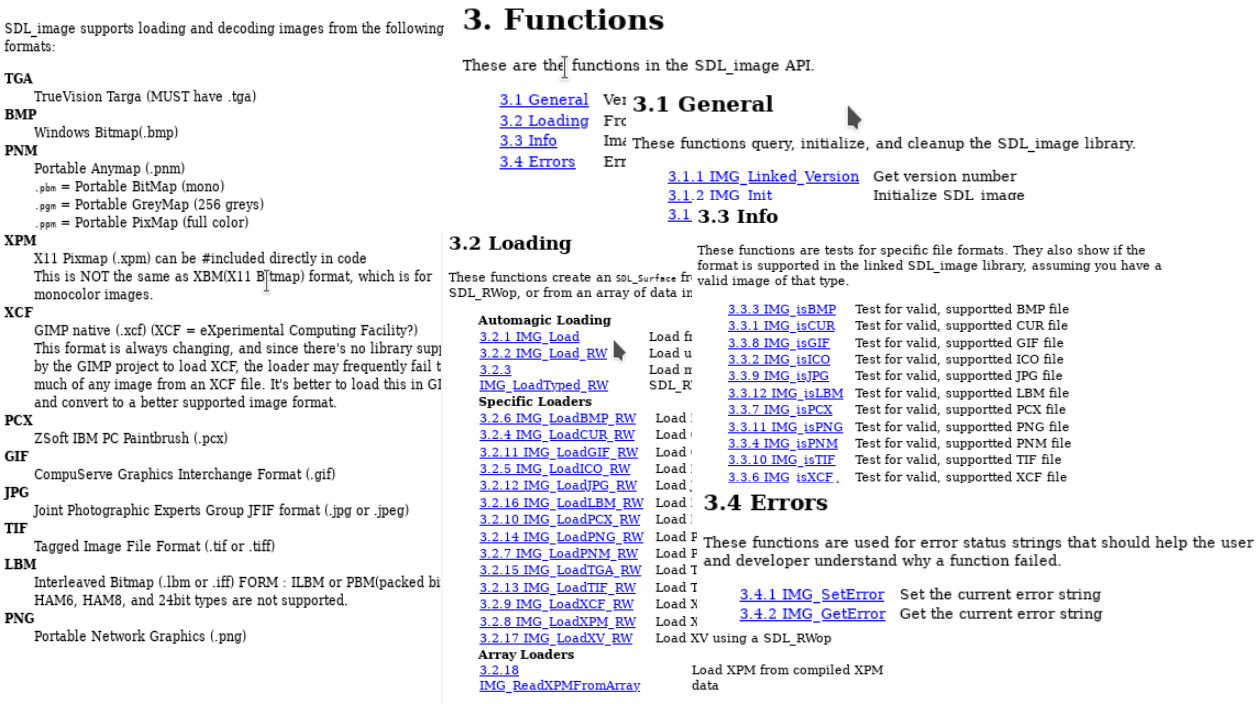
```
typedef struct SDL_Surface {
        Uint32 flags;                           /* Read-only */
        SDL_PixelFormat *format;                /* Read-only */
        int w, h;                               /* Read-only */
        Uint16 pitch;                           /* Read-only */
        void *pixels;                           /* Read-write */

        /* clipping information */
        SDL_Rect clip_rect;                     /* Read-only */

        /* Reference count -- used when freeing surface */
        int refcount;                           /* Read-mostly */

        /* This structure also contains private fields not shown here */
} SDL_Surface;
```

Figura 3: Definiciones de conceptos básicos de dibujo en SDL (core): SDL_Rect y SDL_Surface [3].



SDL_image supports loading and decoding images from the following formats:

TGA
TrueVision Targa (MUST have .tga)
BMP
Windows Bitmap(.bmp)
PNM
Portable Anymap (.pnm)
.pbm = Portable BitMap (mono)
.pgm = Portable GreyMap (256 greys)
.ppm = Portable PixMap (full color)
XPM
X11 Pixmap (.xpm) can be #included directly in code
This is NOT the same as XBM(X11 Bitmap) format, which is for monocolor images.
XCF
GIMP native (.xcf) (XCF = eXperimental Computing Facility?)
This format is always changing, and since there's no library supp by the GIMP project to load XCF, the loader may frequently fail t much of any image from an XCF file. It's better to load this in GI and convert to a better supported image format.
PCX
ZSoft IBM PC Paintbrush (.pcx)
GIF
CompuServe Graphics Interchange Format (.gif)
JPG
Joint Photographic Experts Group JFIF format (.jpg or .jpeg)
TIF
Tagged Image File Format (.tif or .tiff)
LBM
Interleaved Bitmap (.lbm or .iff) FORM : ILBM or PBM(packed bi HAM6, HAM8, and 24bit types are not supported.
PNG
Portable Network Graphics (.png)

3. Functions

These are the functions in the SDL_image API.

3.1 General
3.2 Loading
3.3 Info
3.4 Errors

3.1 General

These functions query, initialize, and cleanup the SDL_image library.

3.1.1 IMG_Linked_Version — Get version number
3.1.2 IMG_Init — Initialize SDL image

3.3 Info

These functions are tests for specific file formats. They also show if the format is supported in the linked SDL_image library, assuming you have a valid image of that type.

3.3.3 IMG_isBMP — Test for valid, supportted BMP file
3.3.1 IMG_isCUR — Test for valid, supportted CUR file
3.3.8 IMG_isGIF — Test for valid, supportted GIF file
3.3.2 IMG_isICO — Test for valid, supportted ICO file
3.3.9 IMG_isJPG — Test for valid, supportted JPG file
3.3.12 IMG_isLBM — Test for valid, supportted LBM file
3.3.7 IMG_isPCX — Test for valid, supportted PCX file
3.3.11 IMG_isPNG — Test for valid, supportted PNG file
3.3.4 IMG_isPNM — Test for valid, supportted PNM file
3.3.10 IMG_isTIF — Test for valid, supportted TIF file
3.3.6 IMG_isXCF — Test for valid, supportted XCF file

3.2 Loading

These functions create an SDL_Surface fr SDL_RWop, or from an array of data in

Automagic Loading
3.2.1 IMG_Load — Load f
3.2.2 IMG_Load_RW — Load u
3.2.3 IMG_LoadTyped_RW — Load n SDL_R
Specific Loaders
3.2.6 IMG_LoadBMP_RW — Load
3.2.4 IMG_LoadCUR_RW — Load
3.2.11 IMG_LoadGIF_RW — Load
3.2.5 IMG_LoadICO_RW — Load
3.2.12 IMG_LoadJPG_RW — Load
3.2.16 IMG_LoadLBM_RW — Load
3.2.10 IMG_LoadPCX_RW — Load
3.2.14 IMG_LoadPNG_RW — Load P
3.2.7 IMG_LoadPNM_RW — Load P
3.2.15 IMG_LoadTGA_RW — Load T
3.2.13 IMG_LoadTIF_RW — Load T
3.2.9 IMG_LoadXCF_RW — Load X
3.2.8 IMG_LoadXPM_RW — Load X
3.2.17 IMG_LoadXV_RW — Load XV using a SDL_RWop
Array Loaders
3.2.18 IMG_ReadXPMFromArray — Load XPM from compiled XPM data

3.4 Errors

These functions are used for error status strings that should help the user and developer understand why a function failed.

3.4.1 IMG_SetError — Set the current error string
3.4.2 IMG_GetError — Get the current error string

Figura 4: SDL_image: formatos y API [3].

[4] En otros contextos se dibuja en diferentes niveles (*layers*) y el hardware o software gráfico, renderiza (combina en el caso de 2D con transparencia o calcula el valor de cada pixel en caso de 3D) la imagen a mostrar en pantalla.

[5] En caso de inicializar SDL en modo *SDL_DOUBLEBUF*, se mantienen dos áreas diferentes: una para refrescar el contenido en pantalla, mientas que en otra se realizan las operaciones de dibujado. Cuando se ejecuta *SDL_Filp* se intercambian (*fliping*). Esta función es equivalente a *SDL_UpdateRect* (con las cuatro coordenadas a cero).

Para complementar el uso de las imágenes, el módulo *SDL_image* proporciona operaciones para cargar imágenes desde ficheros con formato ICO(Icon)/CUR(Cursor)/BMP, PNM (PPM/PGM/PBM), XPM, LBM(IFF ILBM), PCX, GIF, JPEG, PNG, TGA, TIFF y miniaturas de XV. En algunos casos de forma nativa y, en otros, a través de diferentes bibliotecas externas (como en el caso de JPEG, PNG, TIFF y ZLIB en los casos que utilizan este esquema de compresión).

Las herramientas disponibles en *devkitPro* (*sdl-config* y *pkg-config*) nos permiten caracterizar las dependencias de SDL y *SDL_image* en este contexto, véase el Listado 1. Utilizaremos esta información cuando hablemos de las dependencias de un proyecto con SDL 1.2 en 3DS para generar el distribuible final, para lo que necesitaremos saber cuáles son y cómo indicarlas.

```
$ /opt/devkitpro/portlibs/3ds/bin/sdl-config --version
1.2.15
$ /opt/devkitpro/portlibs/3ds/bin/sdl-config --cflags
-I/opt/devkitpro/portlibs/3ds/include/SDL -D_GNU_SOURCE=1 -ffunction-sections
-fdata-sections -march=armv6k -mtune=mpcore -mfloat-abi=hard -mword-
relocations -I/opt/devkitpro/libctru/include -DARM11 -D_3DS
$ /opt/devkitpro/portlibs/3ds/bin/sdl-config --libs
-L/opt/devkitpro/portlibs/3ds/lib -march=armv6k -mfloat-abi=hard
-L/opt/devkitpro/portlibs/3ds -lSDL -L/opt/devkitpro/libctru/lib -lcitro3d -
lctru -lm
$
$ /opt/devkitpro/portlibs/3ds/bin/arm-none-eabi-pkg-config sdl --modversion
1.2.15
$ /opt/devkitpro/portlibs/3ds/bin/arm-none-eabi-pkg-config sdl  --cflags
-D_GNU_SOURCE=1 -ffunction-sections -fdata-sections -march=armv6k -
mtune=mpcore -mfloat-abi=hard -mword-relocations -DARM11 -D_3DS
-I/opt/devkitpro/portlibs/3ds/include/SDL -I/opt/devkitpro/libctru/include
$ /opt/devkitpro/portlibs/3ds/bin/arm-none-eabi-pkg-config sdl  --libs
-L/opt/devkitpro/portlibs/3ds/lib -L/opt/devkitpro/portlibs/3ds
-L/opt/devkitpro/libctru/lib -L/opt/devkitpro/portlibs/3ds
-L/opt/devkitpro/libctru/lib -march=armv6k -mfloat-abi=hard -lSDL -lcitro3d -
lctru -march=armv6k -mfloat-abi=hard -lSDL -lcitro3d -lctru -lm
$
$ /opt/devkitpro/portlibs/3ds/bin/arm-none-eabi-pkg-config SDL_image --
modversion
1.2.12

$ /opt/devkitpro/portlibs/3ds/bin/arm-none-eabi-pkg-config SDL_image --cflags
-D_GNU_SOURCE=1 -ffunction-sections -fdata-sections -march=armv6k -
mtune=mpcore -mfloat-abi=hard -mword-relocations -DARM11 -D_3DS
-I/opt/devkitpro/portlibs/3ds/include/SDL
-I/opt/devkitpro/portlibs/3ds/include/libpng16
-I/opt/devkitpro/portlibs/3ds/include
-I/opt/devkitpro/portlibs/3ds/include/SDL -I/opt/devkitpro/libctru/include

$ /opt/devkitpro/portlibs/3ds/bin/arm-none-eabi-pkg-config SDL_image --libs
-L/opt/devkitpro/portlibs/3ds/lib -L/opt/devkitpro/portlibs/3ds
-L/opt/devkitpro/libctru/lib -L/opt/devkitpro/portlibs/3ds
-L/opt/devkitpro/libctru/lib -lSDL_image -lpng16 -lz -lm -lctru -lz -
lturbojpeg -march=armv6k -mfloat-abi=hard -lSDL -lcitro3d -lctru -
march=armv6k -mfloat-abi=hard -lSDL -lcitro3d -lctru -lm
```

Listado 1: Versiones y dependencias de SDL con sdl-config y pkg-config para 3DS.

4 Desarrollo

Veamos ahora ejemplos de código comentados sobre el uso de operaciones relativas al uso de imágenes. Distinguiremos dos niveles:

- Las operaciones sobre rectángulos.
- Las operaciones sobre imágenes.
- Las operaciones que ofrece el módulo *SDL_image*.

4.1 Ejemplo de uso de dibujado con rectángulos en SDL *core*

Ya hemos visto parte del API de SDL para el dibujado, veamos ahora un ejemplo práctico de código que podrá utilizar para explorar e inventar y que está basado en el uso de rectángulos, como muestra la Figura 5.

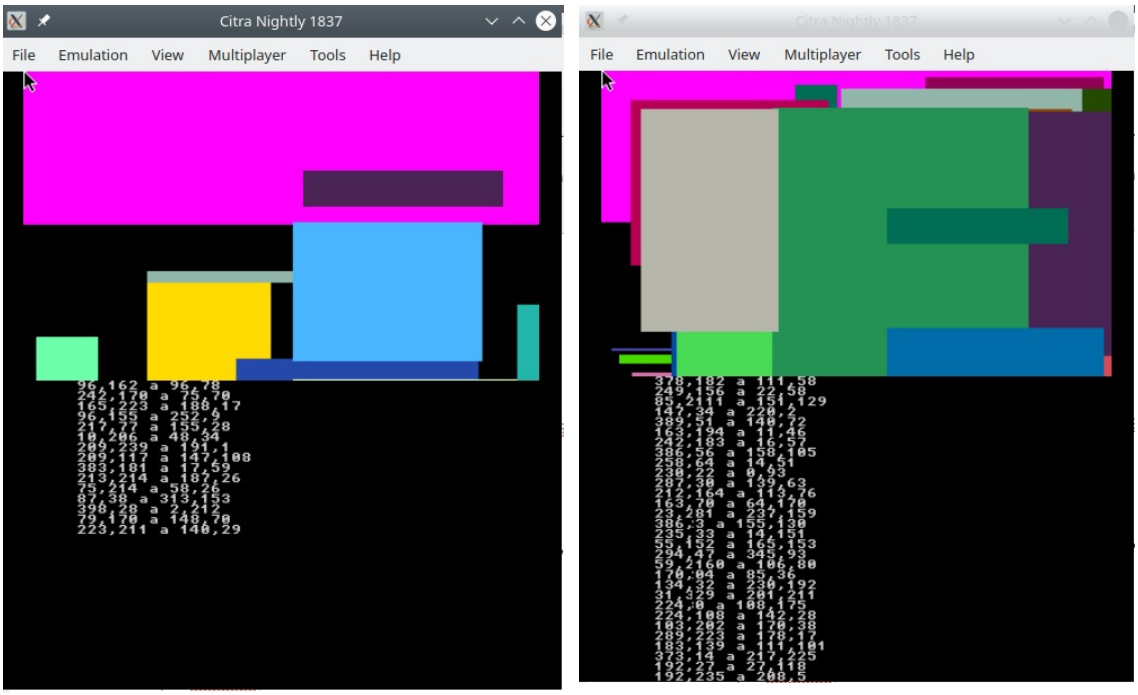


Figura 5: Primitivas de dibujo básico en SDL: ejemplos de uso de rectángulos.

El Listado 2 muestra el código completo de este ejemplo que utiliza las dos pantallas de la 3DS, para ello la configuración (entre las líneas 14 a la 16) establecen el formato de la superior en modo gráfico y la inferior en modo texto.

Tras lo cual se pinta un primer rectángulo a modo de fondo, con tamaño fijo (líneas 18 a la 20) y, posteriormente, en el bucle, se van dibujando nuevos rectángulos con posiciones y tamaños aleatorios (líneas 22 a 35) y mostrando en la pantalla inferior un mensaje con esos valores calculados para el último rectángulo pintado.

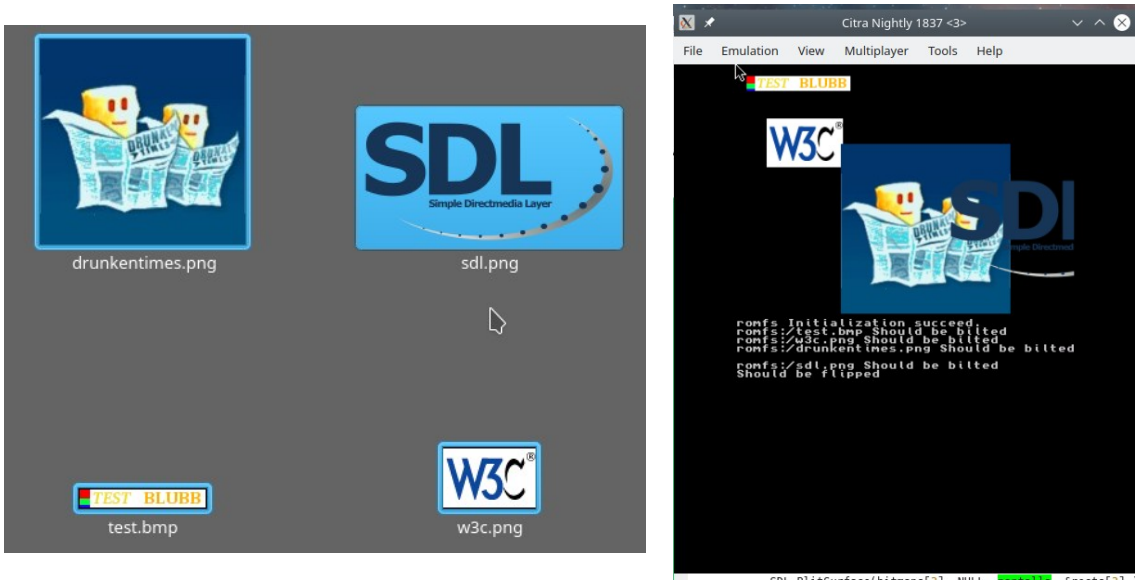
En cada iteración, se comprueba si el usuario quiere terminar en algún momento. Al terminar, se liberan recursos en las líneas 36 y 37.

```
1.  #include <time.h>
2.  #include <SDL.h>
3.  #include <3ds.h>
4.
5.  int main(int argc, char** argv) {
6.      int exit_requested = 0, w, h;
7.      SDL_Surface *pantalla;
8.      SDL_Rect r;
9.
10.     srand(time(NULL));
11.
12.     SDL_VideoDriverName(msg, 1024);  printf("Video Driver %s\n", msg);
13.     w=400; h = 240;
14.     if (SDL_Init(SDL_INIT_VIDEO ) < 0) { return -1;   }
15.     pantalla = SDL_SetVideoMode(w, h, 8,
16.                                 SDL_TOPSCR | SDL_CONSOLEBOTTOM);
17.     SDL_SetCursor( NULL );
18.     SDL_Rect f = {0, 0, w, h/2-1}; // x,y, anple, alt
19.     SDL_FillRect(pantalla, &f,  SDL_MapRGB(pantalla->format, 255, 0,
    255) );
20.     SDL_UpdateRect(pantalla, 0, 0, w, h);
21. ;
22.     while (!exit_requested && aptMainLoop() ) {
23.       r.x = rand()%w;
24.       r.y = rand()%h;
25.       r.w = rand()%w;
26.       r.h = rand()%h;
27.       SDL_FillRect(pantalla, &r,  SDL_MapRGB(pantalla->format, rand()
    %255, rand()%255, rand()%255) );
28.       SDL_UpdateRect(pantalla, 0, 0, w, h); //320, 480);
29.
30.       printf("%d,%d a %d,%d\n", r.x, r.y, r.w, r.h);
31.
32.       hidScanInput();
33.       u32 kDown = hidKeysDown();
34.       if ( (kDown & KEY_START) || (kDown & KEY_A) ) { break; }
35.     } // while (!exit_requested && aptMainLoop() )
36.     SDL_FreeSurface( pantalla );
37.     SDL_Quit();
38.     return 0;
39. }
```

Listado 2: Ejemplo de SDL 1.2: básico con SDL (core).

4.2 Ejemplo de uso de imágenes en *SDL_image*

Sobre un ejemplo de Zephray [7] hemos construido una ampliación para ver cómo se utilizan las funciones del API de *SDL_image*, recuérdese la Figura 4, para hacer uso de diferentes imágenes en formatos de archivos que no son BMP, que es el único soportado en SDL *core*. Vamos a cargar cuatro imágenes (se pueden ver en la Figura 6a) y el resultado de mostrarlas en la 3DS (en la pantalla superior las imágenes, junto a unos comentarios de texto en la inferior), como muestra la Figura 6b.



a) *b)*

Figura 6: Ejemplo de SDL 1.2: básico con SDL_image.: (a) imágenes y (b) resultado

El Listado 3 muestra la secuencia de operaciones, evitando la repetición de los tres ficheros de formato PNG para los que se repite la misma serie de instrucciones.

En primer lugar, líneas 12 a la 15, se definen las rutas y posiciones de las imágenes a mostrar. Y se cargarán las imágenes indicadas en las posiciones predefinidas. La imagen BMP se carga con SDL *core*, líneas 25 a 31.

En cambio las imágenes PNG se cargan haciendo uso del API de *SDL_image*, la primera se carga entre las líneas 33 a la 41 y las otras dos se hace de manera similar, pero se ha obviado por brevedad en la exposición.

Se puede observar que la imagen del logo de SDL (*sdl.png*), que tiene el fondo transparente, aparece aplicada esa propiedad al pintarse sobre el resto de operaciones anteriores. Además, aparece recortada porque su tamaño es mayor que el área de dibujo y se aplica también esta condición de forma interna,

```
1.  #include <3ds.h>
2.  #include <stdio.h>
3.  #include <SDL.h>
4.  #include <SDL_image.h>
5.  #define SCREEN_W 320
6.  #define SCREEN_H 240
7.
8.  int main(int argc, char **argv) {
9.      SDL_Init(SDL_INIT_VIDEO);
10.     SDL_Surface *pantalla;
11.     SDL_Surface *bitmaps[4];
12.     char *rutes[4] = {"romfs:/test.bmp", "romfs:/w3c.png",
13.                       "romfs:/drunkentimes.png", "romfs:/sdl.png"};
14.     SDL_Rect rects[4] = {{10,10, 0,0}, {30, 50, 0,0},
15.                          {100,75, 0,0}, {200,100, 0,0}};
16.
17.     pantalla = SDL_SetVideoMode(SCREEN_W, SCREEN_H, 32,
18.                         SDL_SWSURFACE | SDL_TOPSCR | SDL_CONSOLEBOTTOM);
19.
20.     Result rs = romfsInit();
21.     if (rs)
22.         printf("romfsInit: %08lx\n", rs);
23.     else {
24.         printf("romfs Initialization succeed.\n");
25.         bitmaps[0] = SDL_LoadBMP( rutes[0] );
26.         if (bitmaps[0] == NULL)
27.             printf("Open Bitmap %s failed!\n", rutes[0] );
28.         else{
29.             SDL_BlitSurface(bitmaps[0], NULL, pantalla, &rects[0]);
30.             printf("%s Should be bilted\n", rutes[0] );
31.         }
32.        // load sample.png into image
33.         SDL_RWops *rwop;
34.         rwop=SDL_RWFromFile(rutes[1], "rb");
35.         bitmaps[1] = IMG_LoadPNG_RW(rwop);
36.         if (bitmaps[1] == NULL)
37.             printf("Open Bitmap %s failed!\n", rutes[1] );
38.         else {
39.             SDL_BlitSurface(bitmaps[1], NULL, pantalla, &rects[1]);
40.             printf("%s Should be bilted\n", rutes[1] );
41.         }
42.         // La secuencia para los otros dos PNGs es idéntica a este
43.     } // else de if (rs)
44.
45.     SDL_Flip(pantalla);
46.     printf("Should be flipped\n");
47.     SDL_UpdateRect(pantalla, 0, 0, SCREEN_W, SCREEN_H);
48.     SDL_Delay(5000);
49.
50.     SDL_FreeSurface( bitmaps[0] );  SDL_FreeSurface( bitmaps[1] );
51.     SDL_FreeSurface( bitmaps[2] );  SDL_FreeSurface( bitmaps[3] );
52.
53.     SDL_Quit();
54.     return 0;
55. }
```

Listado 3: Ejemplo de SDL 1.2 utilizando el módulo SDL_image.

5 Conclusión y cierre

A lo largo de este objeto de aprendizaje hemos visto cómo se gestiona el uso de las imágenes en SDL desde sus operaciones básicas en el módulo *core* de SDL hasta la carga de formatos de ficheros gráficos complejos que ofrece el módulo *SDL_image*. Por supuesto que hay más operaciones relacionadas con la imagen, si tiene la curiosidad debe revisar la documentación.

Ahora, con estas reflexiones es cuestión de proponerse un ejemplo propio. ¿Qué te parece si aprovechamos las opciones básicas exploradas para hacer un juego estilo Solitario o *Simon*[6]. ¿Te te animas, estimado lector?

6 Bibliografía y referencias

[1] Sitio web de SDL. Disponible en: <https://www.libsdl.org/>,

[2] Wikipedia - Simple DirectMedia Layer. Disponible en <https://en.wikipedia.org/wiki/Simple_DirectMedia_Layer>.

[3] Manual de referencia de SDL1.2.15. "SDL Reference". Disponible en la URL <https://www.libsdl.org/release/SDL-1.2.15/docs/html/reference.html >.

[4] Repositorio de los ejemplos de este artículo <https://github.com/magusti/3DS/3DS_SDL_1_2_imageBasic >.

[5] devkitPro / 3DS examples. <https://github.com/devkitPro/3ds-examples>.

[6] Manual de usuario de SDL1.2.15. "SDL Guide". Disponible en la URL <https://www.libsdl.org/release/SDL-1.2.15/docs/html/guide.html >.

[7] zephray. SDL-1.2-N3DS Public – Examples <https://github.com/zephray/SDL-1.2-N3DS>.

[6] Puede leer sobre la historia de este legendario juego en <https://es.wikipedia.org/wiki/Simon_(juego)>.