UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Introducción al desarrollo con SDL 1.2 para plataforma 3DS. Modos de vídeo y eventos

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, el logotipo se muestra en la Figura 1a) [**1**] 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, 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.

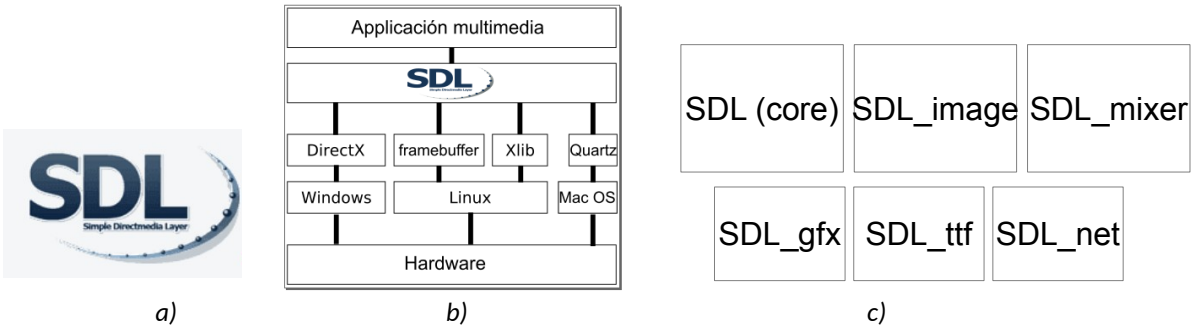


Figura 1: SDL: (a) Logotipo de SDL, (b) Niveles de capa de abstracción del SO sobre el que se ejecuta una aplicación multiplataforma y (c) Módulos de SDL: principal y extensiones oficialesImágenes de <https://gbatemp.net/threads/release-sdl-3ds-1-2-15-simple-directmedia-layer-for-3ds.459291/> y <http://easy-learn-computer.blogspot.com/2012/01/learning-sdl-simple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

En 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. Actualmente, SDL 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*).

[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>.

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 ofrecer un nivel de abstracción común por encima de estas plataformas. Para ello, como muestra la Figura 1c fue diseñada como un API 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 para el que puede abordar un desarrollo con SDL.

SDL se compone de diferentes módulos, 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 la funcionalidad del **módulo principal o *core 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 a la plataforma 3DS.
- Explorar ejemplos que permitan experimentar con la funcionalidad del módulo principal de SDL en el contexto de la plataforma N3DS.

Para no extender la longitud del artículo se ha creado un repositorio en GitHub [5] donde se alojarán los ejemplos de código que se referencian en este artículo. El resto de este documento se centrará en describir las características generales de SDL aplicadas a la plataforma 3DS y en explorar ejemplos de uso aplicados a los modos de vídeo y eventos susceptibles de ser utilizados en dicha plataforma.

3 Introducción

Es posible encontrar varios trabajos de adaptación del código fuente de SDL para la plataforma 3DS como los realizados por *Rikku2000*[3] (2024), sobre el que *nop90*[4] (2017) dice con-

[3] Véase "LibSDL 4 Nintendo 3DS" <https://gbatemp.net/threads/sdl-for-3ds.374519/>.

[4] Puede encontrarse el trabajo de "SDL-3DS - SDL 1.2 for 3DS" en el repositorio <https://github.com/nop90/SDL-3DS>.

tinuar, sobre el cual es posible encontrar el de zephray[5] (2017) y el de Pierre Wendling[6]. La disponibilidad de SDL estable, la encontramos en el conjunto de librerías PORTLIBS distribuidas con *devkitPro* [2] 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 han liberado los fuentes 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 [2], es interesante disponer de un mínimo conjunto de ejemplos que aborden la funcionalidad de SDL 1.2.

3.1 Características de SDL en la plataforma 3DS

Como la versión actual para 3DS es la perteneciente a la rama SDL 1.2, solo se puede gestionar una ventana, centrada y a pantalla completa, en el caso del computador de escritorio. Aquí, en la plataforma 3DS esto se traduce en una serie de posibles combinaciones entre la pantalla superior y la inferior. Como muestra la Figura 2, se pueden utilizar valores de profundidad de color o bits por pixel (bpp) de 32 (con 1 byte para cada canal de color RGB y para el canal de transparencia *Alpha*), 24 (solo RGB), 16 y 8 (que son modos para mantener la compatibilidad con la NDS).

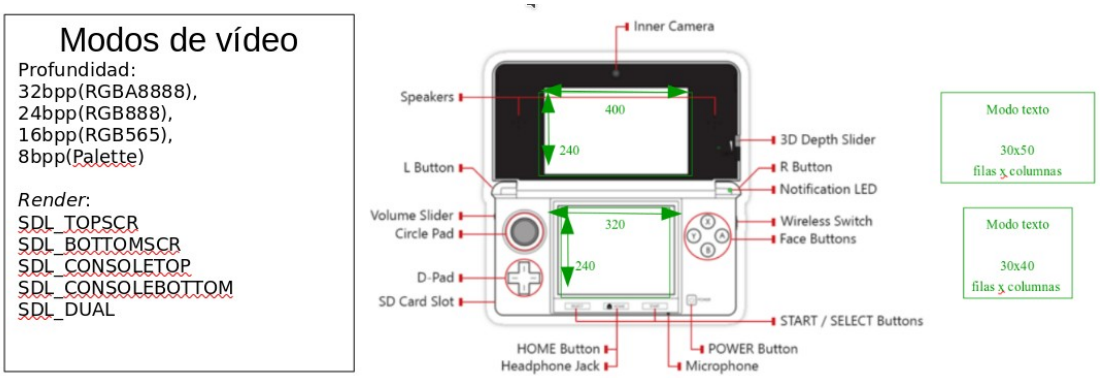


Figura 2: Resumen de modos gráficos en SDL 1.2 para 3DS.
Imagen extraída de <https://en.wikipedia.org/wiki/Nintendo_3DS>.

Además, la resolución de pantalla, es diferente (Figura 2) tanto en modo gráfico como en modo texto, entre ambas pantallas, condicionado por el tamaño físico de ambas. Por lo que es posible tener diferentes combinaciones que, en la terminología de SDL para 3DS, se definen como:

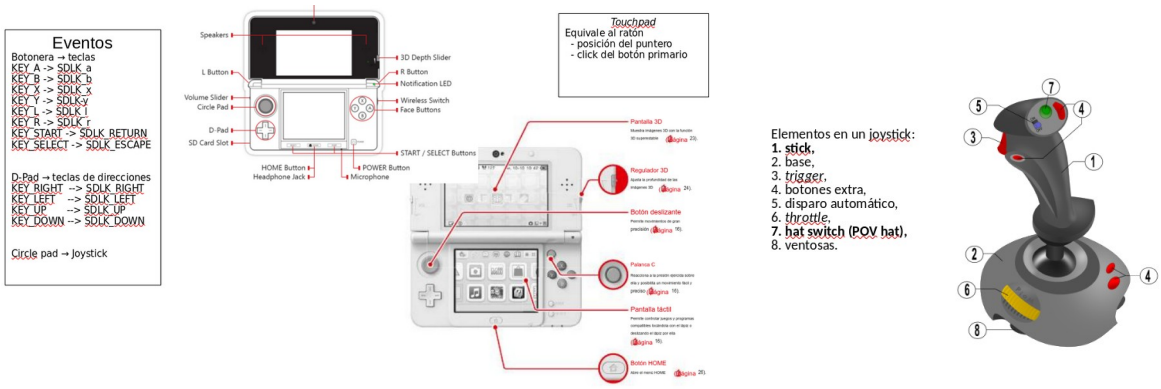
- *SDL_TOPSCR* (la opción por defecto) o *SDL_BOTTOMSCR* que seleccionan la pantalla superior o la inferior, respectivamente, como dispositivo de salida (render). Y *SDL_CONSOLETOP* o *SDL_CONSOLEBOTTOM* que hacen lo propio en modo texto o consola. Por lo que podemos escoger por separado el modo de cada una de las dos pantallas.
- SDL_DUAL, es un modo específico para poner las dos pantallas en modo gráfico.

[5] Disponible en "SDL-1.2-N3DS" <https://github.com/zephray/SDL-1.2-N3DS >.

[6] En "SDL Wiki - Nintendo 3DS" <https://wiki.libsdl.org/SDL2/README/n3ds>.

Por otra parte, es posible recibir eventos de entrada desde, véase Figura 3:

- La botonera, que corresponde con la teclas A, B, X, Y, *Start*, *Select* y el *D-Pad* (la cruceta con *Left*, *Right*, *Up* y *Down*).
- La pantalla táctil, *touchpad*, simulada como un ratón con un solo botón.
- El *Circle Pad* y el *C-Stick* (solo en la New 3DS, véase el detalle en la Figura 3a) que están mapeados como un *Joystick*. Este es un elemento muy complejo, como muestra la Figura 3b, que en este caso se ha hecho corresponder el movimiento del *stick* con el *Circle Pad* y el *hat switch* con el *C-Stick*.



a) *b)*

Figura 3: Eventos de entrada SDL 1.2 en 3DS: (a) resumen del hardware en la 3DS y (b) Joystick. Imágenes de <https://en.wikipedia.org/wiki/Nintendo_3DS>, <https://www.manualslib.es/manual/121502/Nintendo-New-Nintendo-3Ds.html> y <https://en.wikipedia.org/wiki/Joystick>.

4 Desarrollo

En este punto se abordan los detalles que pertenecen a la implementación de ejemplos de código que se pueden realizar sobre los temas que nos ocupan. Para ello vamos a empezar por abordar la configuración del modo de vídeo y después entraremos a ver la gestión de los eventos.

4.1 Establecer el modo de vídeo en SDL para 3DS

En este apartado comentamos un ejemplo de código, véase Listado 1, que configura la salida gráfica de la 3DS. SDL 1.2 basa el dibujado en pantalla en una única zona, a pantalla completa, en lo que denomina un objeto de tipo *SDL_Surface*. Para ello, la secuencia de pasos habitual es:

- Declarar e inicializar una variable para dibujar. En la línea 10 vemos que utiliza el tamaño de la pantalla superior para definir esta en modo gráfico y la inferior en modo texto, véase la Figura 4a. Utilizamos 8 bits por pixel para el color, pero podrían se 16, 24 o 32. En la línea 12 hemos ocultado el cursor, porque habitualmente no se utiliza en 3DS.

- Después tenemos el bucle principal encargado de dibujar y recoger las acciones del usuario. Entre las líneas 14 a la 19 se define un rectángulo (de dimensiones y color aleatorio) para mostrar algo gráfico en la pantalla y que se hace efectiva con la línea 18, de manera que se podrían agrupar varias operaciones para que fueran procesadas juntas y optimizar tiempos. Habrá visto que no se indica la pantalla, como tampoco se hace antes del "printf" (línea 19), según el tipo de salida de la instrucción va a una pantalla física u otra de la 3DS. Pero observe, Figura 4b, que si se declaran las dos en modo gráfico y se utiliza "printf" aparece mezclado en la pantalla inferior.
- Entre las líneas 20 a 22 se leen las pulsaciones de teclas al modo 3DS, en este ejemplo, simplemente para ver si hay que detener la aplicación.
- Al terminar se liberan recursos (línea 24) y se indica al servidor de SDL (línea 25).

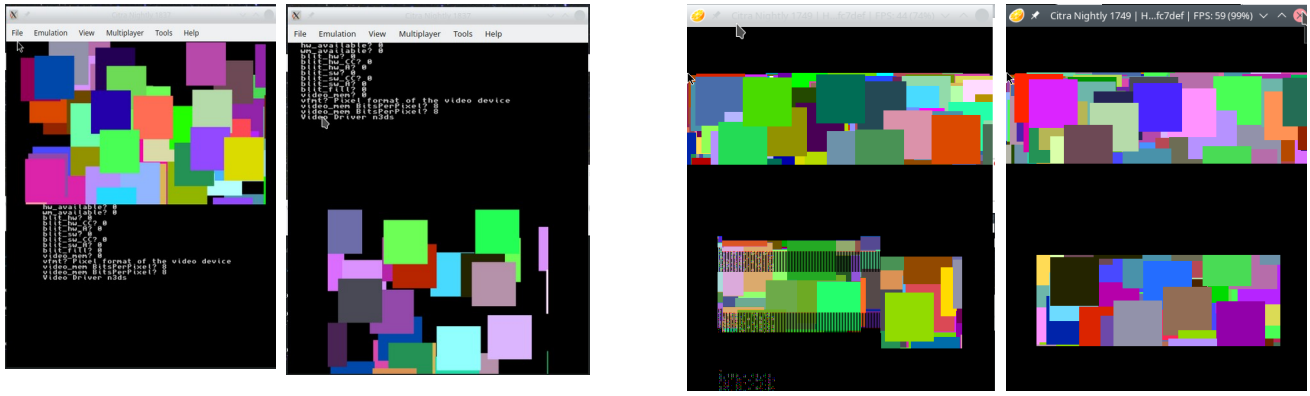
```
1.  #include <time.h>
2.  #include <unistd.h>
3.  #include <SDL.h>
4.  #include <3ds.h>
5.  int main(int argc, char** argv) {
6.    SDL_Surface *pantalla;
7.    SDL_Rect r;
8.    srand(time(NULL));
9.    SDL_Init(SDL_INIT_VIDEO);
10.   pantalla = SDL_SetVideoMode(400, 240, 8,
11.                               SDL_TOPSCR | SDL_CONSOLEBOTTOM);
12.   SDL_SetCursor( NULL );
13.   while ( aptMainLoop() ) {
14.      r.x = rand()%400;     r.y = rand()%240
15.      r.w = 64;             r.h = 64;
16.      SDL_FillRect(pantalla, &r,
17.    SDL_MapRGB(pantalla->format, rand()%255, rand()%255, rand()%255) );
18.      SDL_UpdateRect(pantalla, 0, 0, 400, 240);
19.      printf("rect %d,%d,%d,%d\n", r.x, r.y, r.w, r.h );
20.      hidScanInput();
21.      u32 kDown = hidKeysDown();
22.      if ( (kDown & KEY_START) || (kDown & KEY_A) ) { break; }
23.   }
24.   SDL_FreeSurface( pantalla );
25.   SDL_Quit();
26.   return 0;
27. }
```

Listado 1: Código desarrollado para explorar los modos de vídeo de SDL 1-.2 en N.3DS

Así podemos obtener salidas como las que se muestran en la Figura 4 que muestran una pantalla en modo texto y la otra en modo gráfico o las dos en modo gráfico, siempre con el valor de ancho a 400 y el de alto a 240. Fíjese que la Figura 4b muestra que no se llenan las pantallas. El uso de la pantalla en modo SDL_DUALSCR[7], véase Figura 5a, permite establecer el modo gráfico en las dos pantallas a la vez, lo cual se gestiona (internamente) como dos áreas gráficas de coordenadas consecutivas en filas: así que es posible dibujar elementos que pasan de una zona a otra fácilmente y hay que llevar cuidado porque no es necesario activar una u otra para dibujar (como sucede en otros contextos) puesto que son las dos un mismo *framebuffer*. La resolución efectiva en este caso es de 320x480 píxeles (de ancho y

[7] Puede encontrar más detalles sobre SDL_DUAL en <https://gbatemp.net/threads/release-sdl-3ds-1-2-15-simple-directmedia-layer-for-3ds.459291/page-10>. .

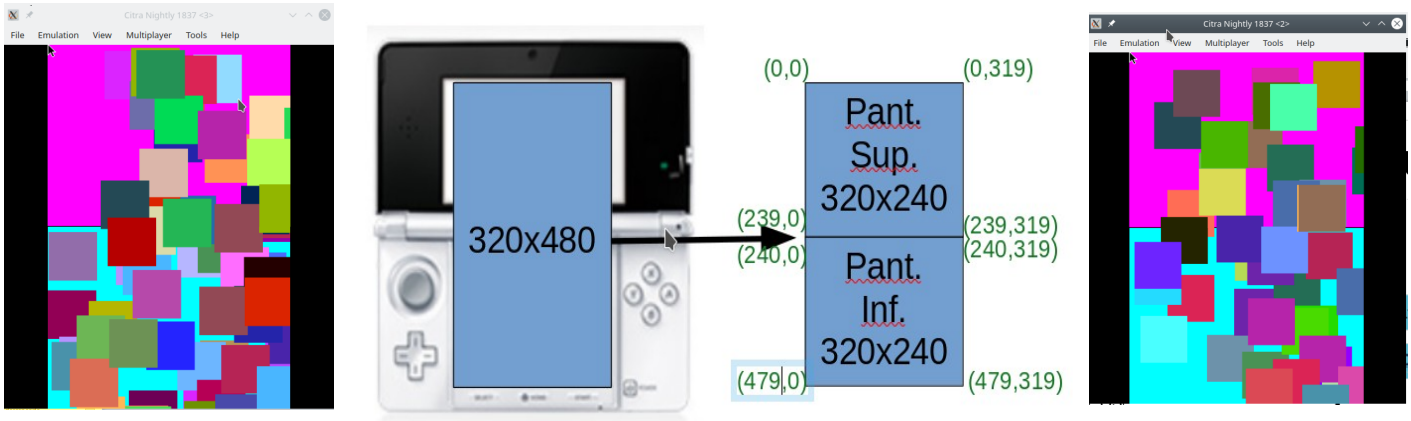
alto), otra cosa es que el hardware permite adaptarlo a la pantalla y así “estirar” la parte de arriba para cubrir el mayor tamaño de pantalla que tiene la superior.



a) *b)*

Figura 4: Ejemplos de salida de modos gráficos de 400x240: (a) gráfico y texto y (b) solo gráficos.

Es posible obtener el mismo resultado con las dos pantallas en modo gráfico (SDL_TOPSCR | SDL_BOTTOMSCR) si fijamos también las dimensiones a 320x480 píxeles, como muestra la Figura 5b. Tenga cuidado con las dimensiones y qué modo se activa porque el resultado podría ser que las dos pantallas se quedaran en negro.



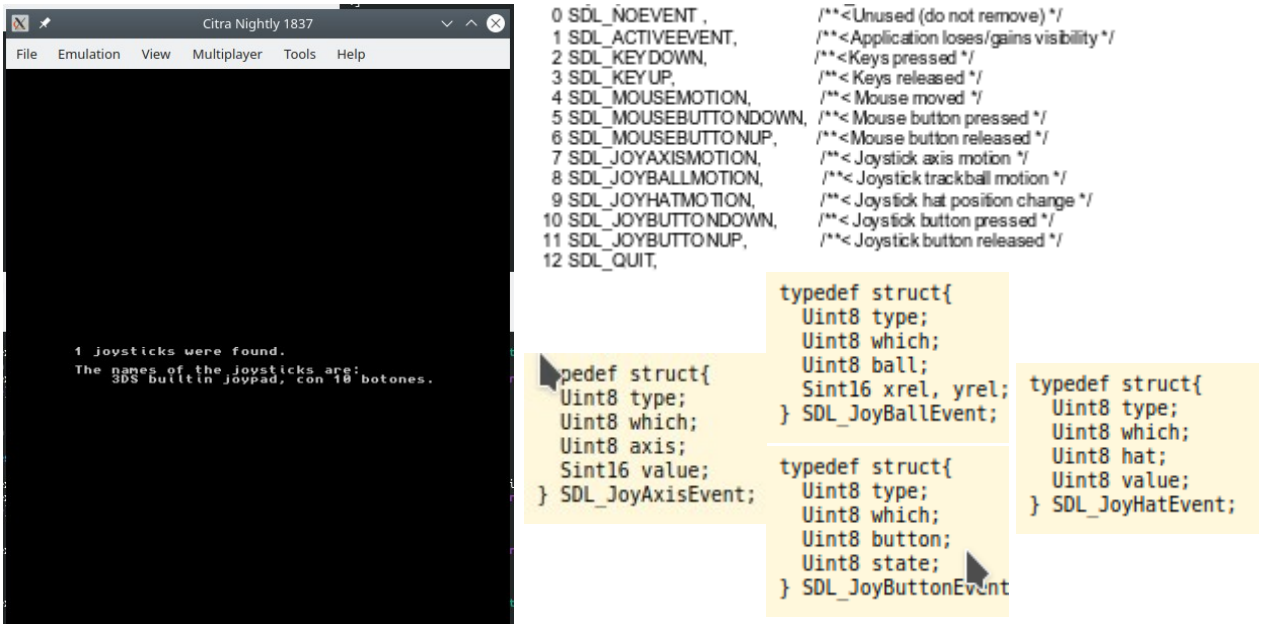
a) *b)*

Figura 5: Modo gráfico (a) SDL_DUAL: 320x480 mapeados a dos pantallas , para las que se muestran las coordenadas de las esquinas de ambas áreas como parejas de índices (fila, columna) y (b) equivalente con SDL_TOPSCR | SDL_BOTTOMSCR con 320x480

4.2 Gestión de eventos en SDL para 3DS

La “botonera” de la 3DS genera eventos como el teclado en el computador de escritorio, pero el resto de elementos de la interfaz física de la 3DS no están en el computador. Para procesar esos nuevos eventos hay que atender a otros tipos. Para utilizar el resto de controles y la pantalla táctil hemos de recurrir a otros eventos diferentes de las pulsaciones de teclas que se utilizan en SDL en el computador. Recuerde que hemos adelantado ya esta si-

tuación al hablar de cómo se corresponden las características de la 3DS a la funcionalidad de SDL (revise el apartado 3.1y la Figura 3). Vamos a ver ejemplos de uso de los diferentes eventos y a qué controles están asociados. El Listado 2 que se muestra en este apartado no está completo por brevedad en la exposición, pero lo encontrará en el repositorio [5] creado para completar este artículo.



a) *b)*

Figura 6: Eventos de SDL: (a) capturas de la salida de sdl1_2_modoTXT_i_events.c y (b) tipos de eventos y algunas estructuras de datos SDL para 3DS [4].

A diferencia del ejemplo del Listado 1, en este caso se inicializa (línea 42 del Listado 2) el sistema gráfico y también el *joystick* (con SDL_INIT_JOYSTICK), para poder configurar SDL a que atienda a los eventos del resto de controles. La configuración del modo de vídeo a modo texto, en las líneas 43 y 44, configura las dos pantallas; aunque se verá en la ejecución (Figura 6a) que todo el texto que generan los "printf" va solo a la pantalla inferior. Entre las líneas 45 a la 55 consultamos cuántos *joysticks* y qué nombre tienen. dejaremos abierto uno, el único que sabemos que habrá en esta plataforma (línea 54) y lo comprobamos (Figura 6a), obteniendo que solo se detecta uno de nombre "3DS builtin joypad" que nos dice que tiene 10 botones.

La botonera estará asociada a los eventos SDL_KEY, la pantalla táctil a los SDL_MOUSE y los *joysticks* tienen asociados cinco tipos de eventos (Figura 6b): SDL_JOYAXIS, SDL_JOYBALL, SDL_JOYHAT, SDL_JOYBUTTON y los dos SDL_JOYBUTTON. Por lo que el bucle principal (que empieza en la línea 56) va a ir recibiendo eventos (línea 57) y, en este caso, generando mensajes de texto para ver cómo acceder a las estructuras de datos correspondientes. Por lo que, tras los habituales en otras plataformas SDL_KEYDOWN y SDL_QUIT (líneas 60 a la 72), vemos entre las líneas 73 a la 81 y 82 a la 86 (del Listado 3) cómo gestionar el uso de la pantalla táctil como un ratón, del que se puede consultar su posición y la pulsación de un botón.

```
28. #include <time.h>
29. #include <unistd.h>
30. #include <SDL.h>
31. #include <stdio.h>
32. #include <3ds.h>
33. #define SCREEN_W 320
34. #define SCREEN_H 240
35.
36. int main(int argc, char** argv) {
37.     int exit_requested = 0, wait = 25,i=0;
38.     SDL_Surface *pantalla;
39.     SDL_Event event;
40.     SDL_Joystick *joystick;
41.
42.    SDL_Init(SDL_INIT_VIDEO | SDL_INIT_JOYSTICK);
43.    pantalla = SDL_SetVideoMode(320, 249, 8,
44.                                SDL_CONSOLETOP | SDL_CONSOLEBOTTOM);
45.     printf("%i joysticks were found.\n\n", SDL_NumJoysticks() );
46.     if (SDL_NumJoysticks() > 0) {
47.         printf("The names of the joysticks are:\n");
48.         for( i=0; i < SDL_NumJoysticks(); i++ )    {
49.             joystick = SDL_JoystickOpen(i);
50.             printf("    %s, con %d botones.\n", SDL_JoystickName(i),
51.                                    SDL_JoystickNumButtons(joystick));
52.             SDL_JoystickClose( joystick );
53.         }
54.        joystick = SDL_JoystickOpen(0);
55.     } // if (SDL_NumJoysticks() > 0) {
56.     while (!exit_requested && aptMainLoop() ) {
57.     if(SDL_PollEvent(&event))   {
58.         printf("event.type %d\n", event.type);
59.         switch(event.type)    {
60.             case SDL_KEYDOWN: // Análogo para SDL_KEYUP
61.                 printf( "SDL_KEYDOWN, %s\n",
62.                        SDL_GetKeyName( event.key.keysym.sym ) );
63.                 printf( "type %u8 state %u8. keysym %d\n",
64.                                 event.key.type, event.key.state, 0);
65.                 if(event.key.keysym.sym == SDLK_a)
66.                   printf("SDLK_A\n");
67. ...
68.                 break;
69.             case SDL_QUIT:
70.                 printf( "SDL_QUIT\n" );  exit_requested = 1;
71.                 break;
72.
73.             case SDL_MOUSEMOTION:
74.                 printf( "SDL_MOUSEMOTION %dx%d\n",
75.                         event.motion.x, event.motion.y );
76.                 break;
77.             case SDL_MOUSEBUTTONUP:
78.                  printf( "SDL_MOUSEBUTTONUP type %d button %d %dx%d\n",
79.                         event.button.type, event.button.button,
80.                         event.button.x, event.button.y );
81.                  break;
```

Listado 2: sdl1_2_modoTXT_i_events.c (primera parte).

```
82.                 case SDL_MOUSEBUTTONDOWN:
83.                    printf( "SDL_MOUSEBUTTONDOWN type %d button %d %dx%d\n",
84.                     event.button.type, event.button.button, event.button.x,
85.                     event.button.y );
86.                      break;
87.                 case SDL_JOYAXISMOTION:
88.                     printf("SDL_JOYAXISMOTION which %d axis %d val %d\n",
89.                   event.jaxis.which, event.jaxis.axis, event.jaxis.value );
90.                     if ( ( event.jaxis.value < -3200 ) ||
91.                           (event.jaxis.value > 3200 ) )   {
92.                         if( event.jaxis.axis == 0)  {
93.                            printf( "Left-right movement code goes here\n");
94.                         }
95.                         if( event.jaxis.axis == 1) {
96.                             printf("Up-Down movement code goes here\b");
97.                         }
98.                     }
99.                     break;
100.                case SDL_JOYHATMOTION:
101.                     printf("SDL_JOYHATBUTTO which %d hat %d val %d\n",
102.                       event.jhat.which, event.jhat.hat, event.jhat.value);
103.                     break;
104.                case SDL_JOYBUTTONDOWN:
105.                   printf("SDL_JOYBUTTONDOWN which %d button %d state %d\n",
106.          event.jbutton.which, event.jbutton.button, event.jbutton.state);
107.                       if (event.jbutton.button == 0) // Start en la 3DS
108.                           printf("Botón Start presionado\n");
109.                       else if (event.jbutton.button == 1) // A
110.                           printf("Botón A presionado\n");
111.                       else if (event.jbutton.button == 2)  // B
112.                           printf("Botón B presionado\n");
113....
114.                       else if (event.jbutton.button == 9)  //  ZR
115.                           printf("Botón y presionado\n");
116.                       break;
117.               case SDL_JOYBUTTONUP:
118.                       printf("SDL_JOYBUTTONUP, botó %d\n",
119.                           event.jbutton.button );
120.                       if (event.jbutton.button == 1)  // A en la 3DS
121.                           printf("Botón A soltat\n");
122.                       break;
123.                case SDL_JOYBALLMOTION:  /* Handle Joyball Motion */
124.                    printf("SDL_JOYBALLMOTION %d, \n", event.jball.ball );
125.                    if( event.jball.ball == 0 ) { /* ball handling */ }
126.                    break;
127.                default: /* Report an unhandled event */
128.                    printf("I don't know what this event is!\n");
129.            }// switch
130.        } //    if/while(SDL_PollEvent(&event))
131.            SDL_Delay(wait);
132.        }
133.        if (joystick != NULL ) SDL_JoystickClose( joystick );
134.        SDL_Quit();
135.        return 0;
136.    }
```

Listado 3: sdl1_2_modoTXT_i_events.c (segunda parte).

Para poder unificar la gestión de eventos, la Figura 6b muestra las estructuras de datos que dan soporte a los eventos de *joystick*, que se diferencian en el tipo de valores que pueden recibir. Entre las líneas 87 a la 99, vemos cómo identificar el movimiento de la palanca (elemento 1 de la Figura 3b), el *hat* (elemento 7 de la Figura 3b) entre las líneas 100 y la 103 para el *Circle Pad* y el *C-Stick*, de la línea 104 a la 122 para los 10 (*Start* , A, B, X, Y, L, R, *Select*, L, R, ZL y ZR) botones y gatillos.

Al terminar el bucle principal, las líneas 133 a la 135 se ocupan de liberar los recursos y terminar la gestión de tareas encargadas a SDL.

5 Conclusión y cierre

A lo largo de este objeto de aprendizaje hemos visto cómo se gestiona el modo gráfico y los eventos que genera el interfaz de la 3DS. Gracias al uso de SDL es posible portar esta aplicación a otra plataforma, como el PC, con las obligadas diferencias por el hardware que utilizan. En el caso de plataformas más similares en hardware, los cambios se reducen y, si además se utiliza la misma versión de SDL en ambas, el número de cambios puede llegar a ser muy próximo a cero e, incluso, cero.

Ahora, con estas reflexiones es cuestión de proponerse un ejemplo propio. ¿Qué tal si aprovechamos la riqueza de eventos para modificar los parámetros de los recuadros que se crean a voluntad o dejamos caer pequeños cuadrados, en la gama de verdes a blancos, atravesando las dos pantallas “estilo matrix”. ¿Te te animas, estimado lector?

6 Bibliografía y referencias

[1] SDL. Sitio web. <https://www.libsdl.org/>.

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

[3] nop90. SDL 1.2 for 3DS. <https://github.com/nop90/SDL-3DS>.

[4] Documentación de SDL 1.2.15 <https://www.libsdl.org/release/SDL-1.2.15/docs/html/guidevideo.html>.

[5] Repositorio de los ejemplos de este artículo <https://github.com/magusti/3DS/3DS_SDL_1_2_modosDeVideo_Eventos>.