

NightFox's Lib para LibNDS.
Manual de referencia.
Edicion 20120318

<http://www.nightfoxandco.com>
<http://www.nightfoxandco.com/forum>
contact@nightfoxandco.com

Instalacion

Simplemente copia la carpeta NFLIB en la raiz de tu proyecto e incluye en tu codigo esta línea

```
#include <nf_lib.h>
```

Además, copia los archivos "makefile" y "icon.bmp" en la raíz de tu proyecto. Estos archivos puedes modificarlos según las necesidades de tu proyecto.

#include "nf_basic.h"

```
void NF_Error( u16 code,          //Codigo de error
               const char* text,  //Descripcion
               u32 value         //Info adicional
               );
```

Genera un error que detiene la ejecucion del programa, informando por pantalla del error dado. Este comando es usado internamente por la librería para generar mensajes de debug y raramente sera usado por el usuario en su codigo.

Ejemplo:

```
NF_Error(112, "Sprite", 37);
```

Genera un error con codigo 112, pasado como parametros del mismo el texto "sprite" y el valor 37.

```
void NF_SetRootFolder(const char* folder // Nombre de la carpeta raiz del proyecto
                      );
```

Define que carpeta sera la raiz del proyecto e inicializa el sistema de archivos (FAT o NitroFS).

Esto facilita poder cambiar el nombre de la carpeta que contendra todos los archivos de nuestro proyecto una vez compilado. Es obligatorio el uso de esta funcion antes de cargar ningun archivo desde la FAT.

Si deseas usar NitroFS, especifica "NITROFS" como nombre de carpeta raiz. Debes copiar el MAKEFILE adecuado en la raiz de tu proyecto para habilitar el uso de NitroFS. Todos los archivos de a cargar deberan encontrarse en la carpeta "nitrofiles".

Ejemplo:

```
NF_SetRootFolder("mygame");
```

Define la carpeta "mygame" como la raiz de nuestro proyecto, usando FAT.

Si tu flashcard no soporta ARGV, usa el Homebrew Menu para lanzar la ROM.

```
void NF_DmaMemCopy( void* destination, // Puntero de destino
                    const void* source, // Puntero de origen
                    u32 size            // Bytes a copiar
                    );
```

Funcion para la copia rapida de bloques de memoria. Realiza una copia mas rapida (de unas 4 veces) de bloques de memoria entra la RAM y la VRAM. La funcion verifica si los datos a copiar estan alineados para su copia usando el canal DMA 3. De no estarlo, realiza la copia mediante el comando memcpy();

Ejemplo:

```
NF_DmaMemCopy((void*)0x06000000, buffer, 131072);
```

Copia a la posicion de memoria 0x06000000 de la VRAM (Banco A), 131072 bytes de memoria (128kb) desde el puntero "buffer" en RAM.

```
u8 NF_GetLanguage(void);
```

Devuelve el ID del idioma del usuario.

```
0 : Japanese  
1 : English  
2 : French  
3 : German  
4 : Italian  
5 : Spanish  
6 : Chinese
```

#include "nf_2d.h"

```
void NF_Set2D( u8 screen,    // Pantalla (0 - 1)
              u8 mode      // Modo (0, 2, 5)
              );
```

Inicia el modo 2D para la pantalla seleccionada.

Mode	Configuration
0	Fondos tileados a 256 colores.
2	Fondos affine tileados de 8 bits en las capas 2 y 3
5	Fondos bitmap a 8 o 16 bits.

Ejemplo:

```
NF_Set2D(1, 0);
```

Inicia el modo 2D para fondos tileados y sprites en la pantalla 1 (inferior)

```
void NF_ShowBg(    u8 screen,    // Pantalla (0 - 1)
                  u8 layer      // Capa (0 - 3)
                  );
```

Muestra el fondo de la capa seleccionada.

Usa este comando para volver a mostrar un fondo previamente oculto.

Ejemplo:

```
NF_ShowBg(0, 2);
```

Hace visible el fondo de la capa 2 en la pantalla 0 (superior)

```
void NF_HideBg(    u8 screen,    // Pantalla (0 - 1)
                  u8 layer      // Capa (0 - 3)
                  );
```

Oculto, sin borrarlo, un fondo de la capa seleccionada.

Ejemplo:

```
NF_HideBg(0, 2);
```

Hace invisible el fondo de la capa 2 en la pantalla 0

```
void NF_ScrollBg(  u8 screen,    // Pantalla (0 - 1)
                  u8 layer,      // Capa (0 - 3)
                  s16 x,         // Posicion X
                  s16 y         // Posicion Y
                  );
```

Mueve el fondo de la capa seleccionada a las coordenadas indicadas.

Ejemplo:

```
NF_ScrollBg(0, 1, 128, 96);
```

Mueve el fondo de la capa 1 en la pantalla 0 a las coordenadas x:128, y:96

```
void NF_MoveSprite(  u8 screen,    // Pantalla (0 - 1)
                    u8 id,        // Id. del Sprite (0 - 127)
                    s16 x,        // Posicion X
                    s16 y,        // Posicion Y
                    );
```

Mueve un sprite a la posicion dada.

Ejemplo:

```
NF_MoveSprite(0, 35, 100, 50);
```

Mueve el Sprite n°35 de la pantalla 0 a las coordenadas x:100, y:50

```
void NF_SpriteLayer( u8 screen,    // Pantalla (0 - 1)
                    u8 id,        // Id. del Sprite (0 - 127)
                    u8 layer      // Capa (0 - 3)
                    );
```

Selecciona sobre que capa sera dibujado un Sprite, siendo 0 la capa mas alta y 3 la mas baja.

Ejemplo:

```
NF_SpriteLayer(1, 35, 2);
```

El sprite n°35 de la pantalla 1 sera dibujado sobre la capa n°2.

```
void NF_ShowSprite(  u8 screen,    // Pantalla (0 - 1)
                    u8 id,        // Id. del Sprite (0 - 127)
                    bool show     // Visibilidad
                    );
```

Muestra o oculta un Sprite. Si se oculta, este se vuelve invisible, sin borrarse.

Ejemplo:

```
NF_ShowSprite(0,35, false);
```

Oculta el Sprite n°35 de la pantalla 0.

```
NF_ShowSprite(1, 45, true);
```

Haz visible el Sprite n°45 de la pantalla 1, previamente ocultado.

```
void NF_HflipSprite( u8 screen,    // Pantalla (0 - 1)
                    u8 id,        // Id. del Sprite (0 - 127)
                    bool hflip    // Volteado horizontal
                    );
```

Voltea horizontalmente un Sprite. Tambien debes usar esta funcion para devolverlo a su estado normal, si esta previamente volteado.

Ejemplo:

```
NF_HflipSprite(0, 35, true);
```

Voltea horizontalmente el Sprite n°35 de la pantalla 0.

```
bool NF_GetSpriteHflip(    u8 screen,    // Pantalla (0 - 1)
                           u8 id,        // Id. del Sprite (0 - 127)
                           );
```

Obtiene el estado del volteado horizontal de un sprite.

Ejemplo:

```
estado = NF_GetSpriteHflip(0, 35);
```

Almacena en la variable "estado" si el Sprite n°35 de la pantalla 0 esta o no volteado.

```
void NF_VflipSprite(  u8 screen,    // Pantalla (0 - 1)
                     u8 id,        // Id. del Sprite (0 - 127)
                     bool vflip    // Volteado vertical
                     );
```

Voltea verticalmente un Sprite. Tambien debes usar esta funcion para devolverlo a su estado normal, si esta previamente volteado.

Ejemplo:

```
NF_VflipSprite(0, 35, true);
```

Voltea verticalmente el Sprite n°35 de la pantalla 0.

```
bool NF_GetSpriteVflip(    u8 screen,    // Pantalla (0 - 1)
                           u8 id,        // Id. del Sprite (0 - 127)
                           );
```

Obtiene el estado del volteado vertical de un sprite.

Ejemplo:

```
estado = NF_GetSpriteVflip(0, 35);
```

Almacena en la variable "estado" si el Sprite n°35 de la pantalla 0 esta o no volteado.

```
void NF_SpriteFrame(  u8 screen,    // Pantalla (0 - 1)
                     u8 id,        // Id. del Sprite (0 - 127)
                     u8 frame     // Frame a mostrar
                     );
```

Selecciona que frame de la animacion de un Sprite sera mostrado.

Ejemplo:

```
NF_SpriteFrame(0, 20, 5);
```

El Sprite n°20 de la pantalla 0 mostrara el frame n°5.

```
void NF_EnableSpriteRotScale(u8 screen,      // Pantalla (0 - 1)
                             u8 sprite,      // Id. del Sprite (0 - 127)
                             u8 id,          // RotSet que usara (0 - 31)
                             bool doublesize // Habilita el doublesize?
                             );
```

Habilita un sprite para poder rotarlo y escalarlo. Debes especificar que rotset entre los 32 diferentes usara este sprite, pudiendo usar varios sprite el mismo set de rotacion y escalado. Si doublesize esta desactivado, el tamaño maximo del Sprite sera 32x32, en caso contrario, el sprite apareceria recortado. Si habilitas la rotacion o escaldo, las funciones de volteado del sprite dejan de ser efectivas.

Ejemplo:

```
NF_EnableSpriteRotScale(1, 111, 12, false);
```

Habilita la rotacion y escalado del Sprite n°111 de la pantalla 1, usando el RotSet n°12, teniendo desactivado el "doublesize".

```
void NF_DisableSpriteRotScale(u8 screen,      // Pantalla (0 - 1)
                              u8 sprite,      // Id. del Sprite (0 - 127)
                              );
```

Deshabilita la rotacion y/o escalado de un Sprite.

Ejemplo:

```
NF_DisableSpriteRotScale(0, 46);
```

Deshabilita la rotacion y escalado del Sprite n°46 de la pantalla 0.

```
void NF_SpriteRotScale(u8 screen,      // Pantalla (0 - 1)
                       u8 id,          // N° de RotSet (0 - 31)
                       s16 angle,      // Angulo (-512 a 512)
                       u16 sx,         // Escalado X (0 a 512)
                       u16 sy,         // Escalado Y (0 a 512)
                       );
```

Rota y/o escala todos los sprites asociados a este RotSet. El angulo de rotacion y el escalado estan en base 512. Eso quiere decir que la rotacion oscila con valores de -512 y 512, siendo 0 el punto central y el escalado entre valores de 0 y 512, siendo 256 el valor central (escala 100%).

Ejemplo:

```
NF_SpriteRotScale(0, 16, 128, 256, 256);
```

Rota 90° a la derecha todos los Sprites asociodos al RotSet n°16, manteniendo la escala del 100%, en la pantalla 0.

```
NF_SpriteRotScale(1, 10, -256, 512, 256);
```

Rota 180° a la izquierda todos los Sprites asociodos al RotSet n° 10, escalando al 200% sobre la X y manteniendo al 100% la escala sobre la Y, en la pantalla 1.

#include "nf_tiledbg.h"

```
void NF_InitTiledBgBuffers(void);
```

Inicializa los buffers en RAM y las estructuras de control para la carga desde la FAT de los archivos necesarios para crear fondos "Tileados" a 256 colores.
Usa esta funcion una vez en tu codigo antes de cargar cualquier fondo.

Ejemplo:

```
NF_InitTiledBgBuffers();
```

Inicializa los buffers para cargar fondos tileados.

```
void NF_ResetTiledBgBuffers(void);
```

Reinicia los buffers y las estructuras de control para la carga desde la FAT de los archivos necesarios para crear fondos "tileados" a 256 colores.
El uso de esta funcion borra todos los datos existentes en los buffers, liberando la memoria RAM de los datos cargados e inicializa todas las variables asociadas.
Suele ser util usar esta funcion cuando se realizan cambio de nivel o pantalla, para borrar todo el contenido no necesario antes de cargar el nuevo.

Ejemplo:

```
NF_ResetTiledBgBuffers();
```

Vacia todos los buffers para fondos y las variables asociadas.

```
void NF_InitTiledBgSys(      u8 screen      // Pantalla  
                          );
```

Inicializa el sistema de fondos tileados para la pantalla seleccionada.
Inicializa todas las variables para el control de fondos, tiles, paletas y mapas.
Configura la VRAM para usar 128kb para fondos en esa pantalla.
Activa las 4 capas para usarlas con fondos Tileados
Reserva 8 bancos de 16kb para tiles (2 reservados para mapas, 6 restantes para tiles).
Reserva 16 bancos de 2kb para mapas (Se usan los 2 primeros bancos de tiles para crear estos 16 bancos de mapas).
Habilita el uso de paletas extendidas.
Total de VRAM para tiles 96kb.
Total de VRAM para mapas 32kb.
Se puede cambiar esta configuracion editando los valores de los siguientes defines:

```
#define NF_BANKS_TILES 8  
#define NF_BANKS_MAPS 16
```

Si los modificas, recuerda que por cada 8 bancos de mapas se consume 1 banco de tiles.
Salvo casos extremos, no se recomienda la modificacion de estos valores.

Esta funcion debe usarse antes del uso de cualquier fondo tileado.

Ejemplo:

```
NF_InitTiledBgSys(1);
```

Inicia el sistema de fondos "tileados" para la pantalla 1.

```
void NF_LoadTiledBg(  const char* file,      // Nombre de archivo, sin extension
                     const char* name,      // Nombre que le daras al fondo
                     u16 width,             // Ancho en pixeles del fondo
                     u16 height            // Alto en pixeles del fondo
                     );
```

Carga los archivos necesarios desde la FAT a la RAM para la creacion de fondos tileados. Los archivos deberan de tener el mismo nombre, usando IMG como extension de los archivos de tiles, MAP los archivos de mapas y PAL los archivos de paletas. Consulta la carpeta GRIT de esta libreria para mas informacion sobre la conversion de fondos. (GRIT es la utilidad de conversion de graficos que se distribuye conjuntamente con el DevKitArm).
Puedes cargar en RAM un maximo de 32 fondos a la vez.
Puedes modificar este limite editando el siguiente define:

```
#define NF_SLOTS_TBG 32
```

Ejemplo:

```
NF_LoadTiledBg("stage1/mainstage", "mifondo", 2048, 256);
```

Carga en RAM los archivos "mainstage.img", "mainstage.map" y "mainstage.pal" de la subcarpeta "stage1" y asigne el nombre de "mifondo". Informa ademas que el tamaño de este fondo es de 2048 x 256 pixeles.

```
void NF_UnloadTiledBg(const char* name      // Nombre del fondo
                      );
```

Elimina de la RAM el fondo con el nombre dado.
Puedes eliminar el fondo de la RAM cuando no lo necesites mas o una vez creado en pantalla, siempre y cuando tenga un tamaño igual o inferior a 512 x 512 pixeles. Si su tamaño es mayor, debes conservar el fondo en RAM hasta que no lo necesites mas.

Ejemplo:

```
NF_UnloadTiledBg("mifondo");
```

Elimina de la RAM el fondo con nombre "mifondo" y libera el slot que usava.

```
void NF_CreateTiledBg(u8 screen,           // Pantalla (0 - 1)
                     u8 layer,             // Capa (0 - 3)
                     const char* name      // Nombre del fondo
                     );
```

Crea un fondo en pantalla, usando los datos previamente cargados en RAM, en la pantalla y capa especificados.
Esta funcion copia los datos necesarios para este fondo desde la RAM a la VRAM. Antes de crear el fondo, carga los datos necesarios usando NF_LoadTiledBg();

Ejemplo:

```
NF_CreateTiledBg(0, 3, "mifondo");
```

Crea en la capa 3 de la pantalla 0 un fondo tileado, usando los datos con la referencia "mifondo".

```
void NF_DeleteTiledBg(u8 screen,    // Pantalla (0 - 1)
                     u8 layer      // Capa (0 - 3)
                     );
```

Borra de la pantalla el fondo de la capa indicada.
Esta funcion tambien los datos de la VRAM.

Ejemplo:

```
NF_DeleteTiledBg(0, 3);
```

Borra el fondos de la capa 3 en la pantalla 0.

```
void NF_LoadTilesForBg(const char* file,    // Nombre de archivo, sin extension
                      const char* name,    // Nombre que le daras al fondo
                      u16 width,           // Ancho en pixeles del fondo
                      u16 height          // Alto en pixeles del fondo
                      u16 tile_start,      // N° de tile inicial
                      u16 tile_end        // N° de tile final
                      );
```

Carga un tileset y su paleta desde la FAT a la RAM, de manera similar a
NF_LoadTiledBg(); pero podremos especificar el rango de tiles a cargar. Ademas, no se
cargara ningun mapa. En su lugar, se creara un mapa vacio de las medidas indicadas.
El fondo se crea posteriormente con la instrucción NF_CreateTiledBg();

Ejemplo:

```
NF_LoadTilesForBg("stagel/mainstage", "mifondo", 256, 256, 0, 23);
```

Carga en RAM los tiles del n°0 al n°23 (24 tiles en total) del archivo "mainstage.img" y
su paleta (archivo "mainstage.pal") ambos de la subcarpeta "stagel" y asigne el nombre
de "mifondo". Informa ademas que el tamaño de este fondo es de 256 x 256 pixeles, lo que
crea un mapa vacio de 32x32 tiles.

```
u16 NF_GetTileOfMap( u8 screen,    // Pantalla (0 - 1)
                    u8 layer,      // Capa (0 - 3)
                    u16 tile_x,    // Posicion X (en tiles)
                    u16 tile_y,    // Posicion Y (en tiles)
                    );
```

Obtiene el valor del tile que se encuentra en las coordenadas del mapa cargado en la
pantalla y capa especificadas.

Ejemplo:

```
u16 mytile = NF_GetTileOfMap(0, 2, 10, 20);
```

Obtiene el valor del tile de las coordenadas x:10 y:20 del mapa cargado en la pantalla
0, capa 2.

```
void NF_SetTileOfMap( u8 screen,    // Pantalla (0 - 1)
                    u8 layer,      // Capa (0 - 3)
                    u16 tile_x,    // Posicion X (en tiles)
                    u16 tile_y,    // Posicion Y (en tiles)
                    u16 tile       // Valor del tile a cambiar (0 - 16364)
                    );
```

Cambia el valor del tile que se encuentra en las coordenadas del mapa cargado en la pantalla y capa especificadas.

Ejemplo:

```
NF_SetTileOfMap(0, 2, 10, 20, 5);
```

Cambia a "5" el valor del tile de las coordenadas x:10 y:20 del mapa cargado en la pantalla 0, capa 2.

```
void NF_UpdateVramMap(u8 screen,    // Pantalla (0 - 1)
                    u8 layer       // Capa (0 - 3)
                    );
```

Actualiza el mapa de la pantalla y capa especificados. Actualiza el mapa en VRAM con la copia en RAM que ha podido ser modificada. Usa esta funcion para actualizar los cambios que realices con la funcion NF_SetTileOfMap();

Ejemplo:

```
NF_UpdateVramMap(0, 2);
```

Actualiza en la VRAM el mapa cargado en la pantalla 0, capa 2.

```
void NF_BgSetPalColor(u8 screen,    // Pantalla (0 - 1)
                    u8 layer,      // Capa (0 - 3)
                    u8 number,     // n° de color en la paleta (0 - 255)
                    u8 r,          // Valor del componente R (0 - 31)
                    u8 g,          // Valor del componente G (0 - 31)
                    u8 b           // Valor del componente B (0 - 31)
                    );
```

Cambia el valor de un color de la paleta del fondo especificado. El cambio se realiza directamente en VRAM, por lo que no recomiendo abusar de esta funcion, ya que puede causar efectos inesperados al manipular directamente la VRAM. Usala para cambiar 1 color por ciclo (por ejemplo, el color de un texto).

Ejemplo:

```
NF_BgSetPalColor(0, 3, 1, 31, 0, 0);
```

Cambia el color n°1 de la paleta de la capa 3 de la pantalla superior a color rojo. Si fuera una capa de texto con la fuente por defecto, el texto pasaria a ser rojo.

```
void NF_BgEditPalColor(    u8 screen,    // Pantalla (0 - 1)
                          u8 layer,      // Capa (0 - 3)
                          u8 number,     // n° de color en la paleta (0 - 255)
                          u8 r,          // Valor del componente R (0 - 31)
                          u8 g,          // Valor del componente G (0 - 31)
                          u8 b           // Valor del componente B (0 - 31)
                          );
```

Cambia el valor de un color de la paleta del fondo especificado. El cambio se realiza sobre la copia en RAM de la paleta, por lo que los cambios no seran visibles hasta que la actualices en VRAM usando la funcion NF_BgUpdatePalette(); Usala para realizar efectos sobre tus fondos tileados.

Ejemplo:

```
NF_BgEditPalColor(0, 3, 1, 31, 0, 0);
```

Cambia el color n°1 de la paleta de la capa 3 de la pantalla superior a color rojo.

```
void NF_BgUpdatePalette(    u8 screen,    // Pantalla (0 - 1)
                           u8 layer      // Capa (0 - 3)
                           );
```

Actualiza en VRAM la paleta del fondo especificado con la copia que se encuentra en RAM.

Ejemplo:

```
NF_BgUpdatePalette(1, 2);
```

Actualiza la paleta del fondo de la capa 2 en la pantalla inferior.

```
void NF_BgGetPalColor(u8 screen,    // Pantalla (0 - 1)
                     u8 layer,      // Capa (0 - 3)
                     u8 number,     // n° de color (0 - 255)
                     u8* r,         // Componente R (0 - 31)
                     u8* g,         // Componente G (0 - 31)
                     u8* b          // Componente B (0 - 31)
                     );
```

Obtiene los valores RGB de un color de la paleta que se encuentra cargada en RAM, del fondo y pantalla especificados.

Ejemplo:

```
u8 rojo;
u8 verde;
u8 azul;
NF_BgGetPalColor(1, 3, 200, &rojo, &verde, &azul);
```

Obtiene el valor RGB del color n°200 del fondo 3 de la pantalla inferior y guardalo en las variables "rojo", "verde" y "azul".

```
extern u8 NF_GetTilePal(    u8 screen,    // Pantalla (0 - 1)
                           u8 layer,      // Capa (0 - 3)
                           u16 tile_x,    // Posicion X del tile (en tiles)
                           u16 tile_y,    // Posicion Y del tile (en tiles)
                           );
```

Obtiene el numero de paleta extendida que esta usando un tile en el fondo especificado. Por defecto, todos los tiles usan la paleta extendida n°0.

Ejemplo:

```
paleta = NF_GetTilePal(0, 3, 20, 10);
```

Obtiene la paleta extendida que esta usando el tile en la posicion 20, 10 del fondo en la capa 3, pantalla superior.

```
void NF_SetTilePal(    u8 screen,    // Pantalla (0 - 1)
                      u8 layer,      // Capa (0 - 3)
                      u16 tile_x,    // Posicion X del tile (en tiles)
                      u16 tile_y,    // Posicion Y del tile (en tiles)
                      u8 pal         // n° de paleta extendida (0 - 15)
                      );
```

Cambia la paleta extendida que usara el tile especificado. La paleta tiene que estar cargada en la VRAM, ademas, los cambios no seran visibles hasta que se ejecute la funcion NF_UpdateVramMap(); ya que todas las operaciones se realizan sobre la copia en RAM del mapa.

Ejemplo:

```
NF_SetTilePal(0, 3, 20, 10, 2);
```

Indica que el tile en la posicion 20, 10 de la capa 3 en la pantalla superior, use la paleta extendida n°2.

```
void NF_LoadExBgPal(    const char* file,    // Archivo (extension .pal)
                       u8 slot              // n° de slot en RAM (0 - 127)
                       );
```

Carga en RAM un archivo de paletas para poderlo usar mas tarde como paleta extendida de fondos.

Ejemplo:

```
NF_LoadExBgPal("bg/sunset", 3);
```

Carga la paleta del archivo "bg/sunset.pal" del sistema de archivos al slot n°3 en RAM.

```
void NF_UnloadExBgPal(u8 slot);    // n° de slot (0 - 127)
```

Borra de la memoria RAM una paleta cargada anteriormente, liberando el espacio usado.

Ejemplo:

```
NF_UnloadExBgPal(5);
```

Borra de la RAM la paleta cargada en el slot n°5. Si la paleta esta transferida en VRAM, esta podra seguir siendo usada.

```
void NF_VramExBgPal( u8 screen,    // Pantalla (0 - 1)
                    u8 layer,      // Capa (0 - 3)
                    u8 id,         // n° de slot de la paleta en RAM (0 - 127)
                    u8 slot       // n° paleta extendida (0 - 15)
                    );
```

Transfiere desde la RAM a la VRAM una paleta para ser usada como paleta extendida.

Ejemplo:

```
NF_VramExBgPal(0, 3, 100, 10);
```

Transfiere la paleta del slot en RAM n°100 a la memoria VRAM del fondo n°3 de la pantalla superior, para ser usada como paleta extendida n°10.

```
void NF_SetExBgPal(  u8 screen,    // Pantalla (0 - 1)
                    u8 layer,      // Capa (0 - 3)
                    u8 pal        // Paleta extendida (0 - 15)
                    );
```

Cambia la paleta extendida que usara el fondo de la capa y pantalla especificados.

Ejemplo:

```
NF_SetExBgPal(0, 3, 5);
```

El fondo de la capa 3 de la pantalla superior, usara la paleta extendida n° 5.

```
void NF_SetTileHflip( u8 screen,    // Pantalla (0 - 1)
                     u8 layer,      // Capa (0 - 3)
                     u16 tile_x,    // Posicion X del tile (en tiles)
                     u16 tile_y    // Posicion Y del tile (en tiles)
                     );
```

Invierte el estado del volteado (FLIP) horizontal de un tile del mapa.

Ejemplo:

```
NF_SetTileHflip(0, 1, 10, 20);
```

Invierte el tile de la posicion x10, y20 del mapa de la capa 1 de la pantalla superior.

```
void NF_SetTileVflip( u8 screen,    // Pantalla (0 - 1)
                     u8 layer,      // Capa (0 - 3)
                     u16 tile_x,    // Posicion X del tile (en tiles)
                     u16 tile_y    // Posicion Y del tile (en tiles)
                     );
```

Invierte el estado del volteado (FLIP) vertical de un tile del mapa.

Ejemplo:

```
NF_SetTileVflip(0, 1, 10, 20);
```

Invierte el tile de la posicion x10, y20 del mapa de la capa 1 de la pantalla superior.

```
void NF_RotateTileGfx(u8 slot,      // Slot
                     u16 tile,      // n° de tile
                     u8 rotation    // Orientacion de la rotacion
                     );
```

Rota el grafico del tile en la direccion indicada. Se rota el tile especificado del buffer en ram con el slot indicado.

Rotaciones:

1 - 90° derecha
2 - 90° izquierda
3 - 180°

Ejemplo:

```
NF_RotateTileGfx(3, 76, 2);
```

Rota 90° a la izquierda el tile n°76 del buffer del slot 3.

#include "nf_sprite256.h"

```
void NF_InitSpriteBuffers(void);
```

Inicializa los buffers y estructuras de control para almacenar los datos necesarios para la creacion de Sprites a 256 colores. Los buffers se usaran para almacenar los datos cargados desde la FAT.

Debes usar esta funcion 1 vez antes de cargar ningun dato para sprites.

Ejemplo:

```
NF_InitSpriteBuffers();
```

Inicializa el sistema de buffers para sprites.

```
void NF_ResetSpriteBuffers(void);
```

Reinicia el sistema de buffers para datos de sprites, vaciando los buffers y reiniciando las variables asociadas. Usalos en cambios de nivel, para vaciar la RAM usada antes de cargar los nuevos datos.

Ejemplo:

```
NF_ResetSpriteBuffers();
```

Vacia todos los buffers para sprites y reinicia las variables asociadas.

```
void NF_InitSpriteSys( u8 screen          // Pantalla
                      [u8 vram_mapping]  // Modo de mapeado de VRAM (64 o 128)
                      );
```

Inicializa el sistema de Sprites para la pantalla seleccionada.

Asigna 128kb de VRAM para graficos y paletas.

Habilita el uso de las paletas extendidas.

El parametro VRAM_MAPPING es optional, si no se especifica, se asume que es 64.

Pueden cargarse en VRAM hasta 1024 tiles de 64 bytes (Mapping 64) o de 128 bytes

(Mapping 128) y 16 paletas. El uso de Mapping 64 limita la VRAM usable a 64kb.

En el caso de usar el Mapping 128, no es posible usar Sprites de 8x8 pixeles.

Inicializa el OAM con los parametros por defecto.

Ejemplo:

```
NF_InitSpriteSys(0);
```

```
NF_InitSpriteSys(1, 128);
```

Inicializa el sistema de sprites a 256 colores en la pantalla 0 en modo "Map64" y en la pantalla 1 en modo "Map128".

```
void NF_LoadSpriteGfx(const char* file, // Nombre de archivo
                      u16 id,           // n° de slot (0 - 255)
                      u16 width,        // Ancho del grafico (en pixeles)
                      u16 height,       // Altura del grafico (en pixeles)
                      );
```

Carga desde la FAT a la RAM un grafico para usarlo posteriormente en la creacion de un Sprite. El nombre de archivo se especifica sin extension. El grafico que cargues debera de usar la extension IMG. Debes de especificar en que slot deseas almacenar el grafico

en RAM, de los 256 disponibles (0 - 255), así como las medias del gráfico. En caso de tratarse de un gráfico con varios frames para una animación, especifica el tamaño del frame (en píxeles).

Si deseas variar el número de slots disponibles, edita el siguiente define:

```
#define NF_SLOTS_SPR256GFX 256
```

Ejemplo:

```
NF_LoadSpriteGfx("stage3/nave", 100, 64, 32);
```

Carga el archivo de gráficos "nave.img" de la subcarpeta "stage3" y almacenalo en el slot nº100 de la RAM. Este gráfico tiene unas medidas de 64 x 32 píxeles.

```
void NF_UnloadSpriteGfx(    u16 id            // nº de slot
                           );
```

Borra de la RAM el gráfico del slot indicado y márcalo como disponible.

Puedes borrar de la RAM el gráfico una vez creado el Sprite, siempre que no lo necesites más o, si es animado, hayas transferido a la VRAM todos los frames.

Ejemplo:

```
NF_UnloadSpriteGfx(100);
```

Borra de la RAM el gráfico contenido en el slot nº100 y márcalo como disponible.

```
void NF_LoadSpritePal(const char* file,    // Nombre del archivo
                     u8 id                // nº de slot (0 - 63)
                     );
```

Carga desde la FAT a la RAM una paleta para poder asignársela a un Sprite.

El nombre de archivo debe introducirse sin extensión. Debes darle al archivo de paletas la extensión PAL.

Puedes almacenar en RAM hasta 64 paletas (0 - 63). Si necesitas ajustar el número de paletas en RAM, edita el siguiente define:

```
#define NF_SLOTS_SPR256PAL 64
```

Ejemplo:

```
NF_LoadSpritePal("stage3/player", 34);
```

Carga el archivo de paleta "player.pal" de la subcarpeta "stage3" en el slot 34.

```
void NF_UnloadSpritePal(    u8 id            // nº de slot (0 - 63)
                           );
```

Borra de la RAM la paleta del slot indicado y márcalo como disponible. Puedes borrarla si ya no es necesaria o ya está transferida a la VRAM.

```
void NF_VramSpriteGfx(u8 screen,          // Pantalla (0 - 1)
                     u16 ram,             // Slot en RAM del gráfico (0 - 255)
                     u8 vram,             // Slot en VRAM del gráfico (0 - 127)
                     bool keepframes      // Copia a la VRAM solo el primer frame
                     );
```

Copia un grafico de la RAM a la VRAM de la pantalla indicada, para poder usarlo en un Sprite. Debes indicar la pantalla de destino, el slot de origen en la RAM (0 - 255), el slot de destino en la VRAM (0 - 127) y en el caso de ser animado, si se deben de mantener los frames adicionales en la RAM (true) o copiarlos todos a la VRAM (false).

Ejemplo:

```
NF_VramSpriteGfx(1, 160, 23, false);
```

Copia el grafico del slot 160 de la RAM al slot 23 de la VRAM de la pantalla 1, copiando todos los frames, si este es animado.

```
void NF_FreeSpriteGfx(u8 screen,    // Pantalla (0 - 1)
                     u16 id        // Slot en VRAM (0 - 127)
                     );
```

Borra de la VRAM el grafico del slot seleccionado de la pantalla indicada. Evita borrar el grafico si algun sprite lo esta usando, ya que este se volviera invisible o aparecera corrompido.

Ejemplo:

```
NF_FreeSpriteGfx(1, 34);
```

Borra de la VRAM de la pantalla 1 el grafico del slot n°34.

```
void NF_VramSpriteGfxDefrag( u8 screen    // Pantalla (0 - 1)
                             );
```

Desfragmenta la libre VRAM usada por los graficos de los Sprites. Este proceso se ejecuta automaticamente cuando la VRAM libre fragmentada es superior en un 50% de la VRAM libre total. No es necesario que nunca ejecutes manualmente este comando. Puedes consultar el estado de la VRAM para sprites leyendo el valor de las siguientes variables:

```
NF_SPRVRAM[u8 screen].free      <- Memoria VRAM total libre
NF_SPRVRAM[u8 screen].fragmented <- Memoria VRAM libre fragmentada
NF_SPRVRAM[u8 screen].inarow    <- Memoria VRAM libre, ultimo bloque mas grande
NF_SPRVRAM[u8 screen].lost      <- Memoria VRAM libre no usable por fragmentacion
```

Ejemplo:

```
NF_VramSpriteGfxDefrag(1);
```

Desfragmenta la VRAM libre para Sprites de la pantalla 1.

```
void NF_VramSpritePal(u8 screen,    // Pantalla (0 - 1)
                     u8 id,        // Slot en RAM de la paleta (0 - 64)
                     u8 slot      // Slot en VRAM de la paleta (0 - 15)
                     );
```

Copia la paleta de la RAM al Slot de paletas extendidas de la VRAM. Si el slot esta en uso, el contenido se sobrescribe.

Ejemplo:

```
NF_VramSpritePal(1, 56, 8);
```

Copia la paleta del slot en RAM n°56 al slot de paletas extendidas n°8 de la pantalla 1.

```
void NF_CreateSprite( u8 screen,    // Pantalla (0 - 1)
                    u8 id,         // n° de Sprite (0 - 127)
                    u16 gfx,       // Slot del grafico que usara (0 - 127)
                    u8 pal,       // Slot de la paleta que usara (0 - 15)
                    s16 x,        // Coordenada X donde se creara
                    s16 y,        // Coordenada Y donde se creara
                    );
```

Crea un Sprite en la pantalla indicada, usando el grafico y paletas dadas. Debes especificar un numero unico de Sprite ente 0 y 127, así como las coordenadas donde deseas crearlo. El numero de sprite asigna la prioridad del mismo, siendo 0 la mas alta (se dibuja el ultimo, quedando encima).

Ejemplo:

```
NF_CreateSprite(0, 12, 30, 1, 100, 50);
```

Crea un Sprite en la pantalla 0, usando el grafico del slot 30, la paleta del slot 1 y dandole como identificador el n°12. El sprite se creara en las coordenadas x:100, y:50.

```
void NF_DeleteSprite( u8 screen,    // Pantalla (0 - 1)
                    u8 id,         // n° de Sprite
                    );
```

Borra de la pantalla indicada el sprite con el numero dado. El grafico usado y la paleta no se borran de la VRAM, solo se elimina el Sprite.

Ejemplo:

```
NF_DeleteSprite(0, 12);
```

Borra de la pantalla 0 el sprite con el identificador n°12.

```
void NF_SpriteOamSet( u8 screen    // Pantalla (0 - 1)
                    );
```

Copia los datos del OAM temporal usado por esta libreria al REAL de las libnds. Usa este comando justo antes del `swiWaitForVBlank()`;
El OAM debe de actualizarse durante el refresco vertical (VBLANK), es decir, justo despues del ejecutarse el `swiWaitForVBlank()`;
Para ello, ejecuta `oamUpdate(&oamMain);` o `oamUpdate(&oamSub);` dependiendo de si es la pantalla principal (superior) o la secundaria (inferior).

Ejemplo:

```
NF_SpriteOamSet(0);
```

Actualiza los datos del OAM de la pantalla 0 (principal).

```
void NF_SpriteSetPalColor(  u8 screen,    // Pantalla (0 - 1)
                           u8 pal,        // Paleta (0 - 15)
                           u8 number,     // n° de color en la paleta (0 - 255)
                           u8 r,          // Valor del componente R (0 - 31)
                           u8 g,          // Valor del componente G (0 - 31)
                           u8 b,          // Valor del componente B (0 - 31)
                           );
```

Cambia el valor de un color de la paleta de sprites especificada. El cambio se realiza directamente en VRAM, por lo que no recomiendo abusar de esta funcion, ya que puede causar efectos inesperados al manipular directamente la VRAM. Usala para cambiar 1 color por ciclo (por ejemplo, el color de un texto).

Ejemplo:

```
NF_SpriteSetPalColor(0, 3, 1, 31, 0, 0);
```

Cambia el color n°1 de la paleta de sprites n°3 de la pantalla superior a color rojo.

```
void NF_SpriteEditPalColor( u8 screen,    // Pantalla (0 - 1)
                           u8 pal,        // Paleta (0 - 15)
                           u8 number,     // n° de color en la paleta (0 - 255)
                           u8 r,          // Valor del componente R (0 - 31)
                           u8 g,          // Valor del componente G (0 - 31)
                           u8 b,          // Valor del componente B (0 - 31)
                           );
```

Cambia el valor de un color de la paleta de sprites especificada. El cambio se realiza sobre la copia en RAM de la paleta, por lo que los cambios no seran visibles hasta que la actualices en VRAM usando la funcion NF_SpriteUpdatePalette(); Usala para realizar efectos sobre tus sprites.

Ejemplo:

```
NF_SpriteEditPalColor(0, 3, 1, 31, 0, 0);
```

Cambia el color n°1 de la paleta de sprites n°3 de la pantalla superior a color rojo.

```
void NF_SpriteUpdatePalette( u8 screen,    // Pantalla (0 - 1)
                             u8 pal        // Paleta (0 - 15)
                             );
```

Actualiza en VRAM la paleta de sprites especificada con la copia que se encuentra en RAM.

Ejemplo:

```
NF_SpriteUpdatePalette(1, 2);
```

Actualiza la paleta de sprites n°2 en la pantalla inferior.

```
void NF_SpriteGetPalColor(  u8 screen,    // Pantalla (0 - 1)
                           u8 pal,        // Paleta (0 - 15)
                           u8 number,     // n° de color (0 - 255)
                           u8* r,         // Componente R (0 - 31)
                           u8* g,         // Componente G (0 - 31)
                           u8* b,         // Componente B (0 - 31)
                           );
```

Obtiene los valores RGB de un color de la paleta de sprites que se encuentra cargada en VRAM, de su copia original en la RAM.

Ejemplo:

```
u8 rojo;
u8 verde;
u8 azul;
NF_SpriteGetPalColor(1, 3, 200, &rojo, &verde, &azul);
```

Obtiene el valor RGB del color n°200 de la paleta n°3 de la pantalla inferior y guardalo en las variables "rojo", "verde" y "azul".

#include "nf_text.h"

```
void NF_InitTextSys( u8 screen    // Pantalla (0 - 1)
                    );
```

Inicializa el motor de texto para la pantalla dada.

Es imprescindible inicializar los buffers y el motor de fondos tileados antes de poder usar el motor de texto. Consulta la seccion [#include "nf_tiledbg.h"](#) para mas informacion acerca de las funciones [NF_InitTiledBgBuffers\(\)](#); y [NF_InitTiledBgSys\(\)](#); Esta funcion tambien reinicia el sistema de texto, vaciando los buffers y liberando la RAM usada.

Ejemplo:

```
NF_InitTextSys(1);
```

Inicializa el motor de texto en la pantalla inferior.

```
void NF_LoadTextFont( const char* file,    // Nombre del archivo
                     const char* name,    // Nombre asignado a la fuente
                     u16 width,           // Ancho de la capa de texto (pixeles)
                     u16 height,          // Alto de la capa de texto (pixeles)
                     u8 rotation          // Rotacion de la fuente (0 - 2)
                     );
```

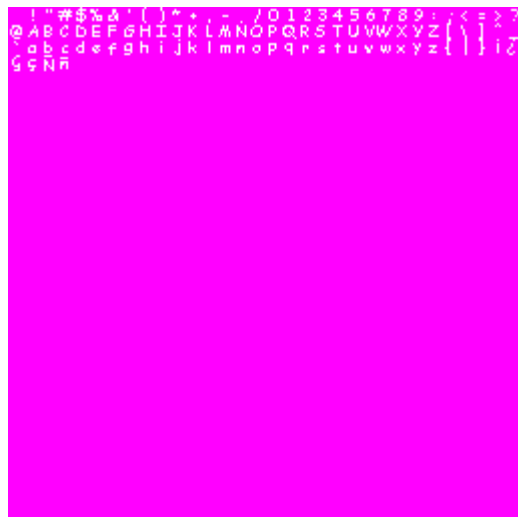
Carga a la RAM desde la FAT un archivo de fuentes.

Debes especificar el nombre de archivo (sin extension), el nombre con el que quieres referenciar la fuente, el ancho y alto de la capa de texto en pixeles y la rotacion de la misma. El valor de rotacion es 0: Ninguna, 1: Derecha, 2: Izquierda.

La fuente se compone de 2 archivos, el de tiles (.FNT) y el de paleta (.PAL).

Debes cargar una fuente por cada capa de texto que quieras crear.

Usa esta plantilla como referencia para crear tus fuentes:



Ejemplo:

```
NF_LoadTextFont("stage4/default", "titulo", 256, 256, 2);
```

Carga la fuente "default" de la carpeta "stage4" y dale el nombre "titulo". La rotacion "2" indica que se cargara la seccion con la fuente rotada a la izquierda. La capa de texto sera de 32x32 tiles (256x256 pixeles).

Cada fuente cargada usara un slot de fondos tileados.

```
void NF_UnloadTextFont(      const char* name      // Nombre de la fuente
                             );
```

Borra de la RAM la fuente con el nombre dado.

Ejemplo:

```
NF_UnloadTextFont("titulo");
```

Borra la fuente con el nombre "titulo".

```
void NF_CreateTextLayer(      u8 screen,           // Pantalla (0 - 1)
                              u8 layer,           // Capa (0 - 3)
                              u8 rotation,        // Rotacion (0 - 2)
                              const char* name    // Nombre de la fuente
                              );
```

Crea un fondo tileado especial para poder escribir texto en el.

Debes especificar el numero de pantalla y la capa donde quieres crear la capa de texto, la rotacion del texto y el nombre de la fuente que quieres usar en esa capa.

La rotacion es 0: Ninguna, 1: Derecha, 2: Izquierda.

Ejemplo:

```
NF_CreateTextLayer(1, 0, 2, "titulo");
```

Crea una capa de texto en la pantalla inferior, en la capa 0, con la fuente rotada a la izquierda, usando la fuente con nombre "titulo".

```
void NF_DeleteTextLayer(      u8 screen,           // Pantalla (0 - 1)
                              u8 layer           // Capa (0 - 3)
                              );
```

Borra una capa de texto.

Debes especificar la pantalla y n° de capa que quieres eliminar.

Ejemplo:

```
NF_DeleteTextLayer(1, 0);
```

Borra la capa de texto de la pantalla inferior, capa 0.

```
void NF_WriteText(    u8 screen,          // Pantalla (0 - 1)
                    u8 layer,            // Capa (0 - 3)
                    u8 x,                // Posicion X
                    u8 y,                // Posicion Y
                    const char* text     // Texto a mostrar
                    );
```

Muestra un texto por pantalla, en las coordenadas dadas. Debes especificar la pantalla, capa y coordenadas donde mostrar el texto. El texto no se escribe directamente en la pantalla, si no en un buffer temporal. Para mostrar en pantalla, debes de ejecutar la instrucción `NF_UpdateTextLayers()`; El motivo de hacer esto es para minimizar los accesos a VRAM durante la escritura del texto.

Si deseas usar texto con formato (mostrar variables, etc) usa el comando `printf()`;

Ejemplo:

```
NF_WriteText(1, 0, 1, 1, "Hola Mundo!");
```

Manda al buffer temporal de la capa nº0 de la pantalla inferior el famoso texto "Hola Mundo!"

Ejemplo 2:

```
char text[32];
u16 myvar = 10;
sprintf(text, "Hola mundo %d veces", myvar);
NF_WriteText(1, 0, 1, 1, text);
```

Manda al buffer temporal de la capa nº 0 de la pantalla inferior el texto "Hola mundo 10 veces".

```
void NF_UpdateTextLayers(void);
```

Copia el contenido de los buffers temporales de texto a la VRAM de las pantallas correspondientes. Solo se actualizan los buffers de las capas que han sido modificadas desde la ultima actualizacion.

Esta funcion actualiza los buffers necesarios de ambas pantallas.

Ejemplo:

```
NF_UpdateTextLayers();
```

Copia los datos de los buffers temporales de texto a la VRAM.

Esto provoca que los textos se muestren en pantalla.

Solo se copian los buffers de la pantalla y capa que se han modificado desde la ultima actualizacion.

```
void NF_ClearTextLayer(    u8 screen,    // Pantalla (0 - 1)
                          u8 layer      // Capa (0 - 3)
                          );
```

Borra el contenido de una capa de texto, poniendo a 0 todos los bytes.

Ejemplo:

```
NF_ClearTextLayer(0, 2);
```

Borra el contenido de la capa de texto de la pantalla 0, capa 2.

```
void NF_DefineTextColor(    u8 screen,    // Pantalla (0 - 1)
                           u8 layer,      // Capa (0 - 3)
                           u8 color,      // n° de color (0 - 15)
                           u8 r,          // Componente R (0 - 31)
                           u8 g,          // Componente G (0 - 31)
                           u8 b           // Componente B (0 - 31)
                           );
```

Define un color basado en los valores RGB para poder ser usado mas tarde en una capa de texto. El color se guardara en el slot especificado. Para que esta funcion funcione, la fuente original debe de estar indexada de 2 colores (Magenta/Blanco).

Ejemplo:

```
NF_DefineTextColor(0, 0, 13, 15, 31, 15);
```

Define el color n°13 de la capa de texto n°0 de la pantalla superior como verde claro.

```
void NF_SetTextColor( u8 screen,    // Pantalla (0 - 1)
                     u8 layer,      // Capa (0 - 3)
                     u8 color       // Color (0 - 15)
                     );
```

Especifica con que color sera escrito el texto de la capa indicada a partir de ese momento. No altera el color del texto ya escrito hasta ese momento.

Ejemplo:

```
NF_SetTextColor(0, 0, 3);
```


Indica que el texto que se escriba a partir de ese momento en la capa 0 de la pantalla superior, sera del color definido en el slot n°3.

#include "nf_text16.h"

Las funciones incluidas aquí son las específicas para mostrar texto con fuentes de 8x16 pixeles, siendo las demás funciones de texto compatibles con este modo.

`void NF_LoadTextFont16() ;`

Como en `NF_LoadTextFont()`; pero con soporte para fuentes de 8x16 pixeles.
Usa esta plantilla como referencia para crear tus fuentes:



```
!"#$%&'()*+,-./0123456789:;<=>?  
@ABCDEFGHIJKLMNopqrstuvwxyz[\]^_  
'abcdefghijklmnopqrstuvwxyz{|}~  
ÇçÑñ
```

`void NF_CreateTextLayer16();`

Como `NF_CreateTextLayer()`; pero con soporte para fuentes de 8x16 pixeles.

`void NF_WriteText16();`

Como `NF_WriteText()`; pero para capas que usan fuentes de 8x16 pixeles.

`void NF_ClearTextLayer16();`

Como `NF_ClearTextLayer()`; pero para capas que usan fuentes de 8x16 pixeles.

#include "nf_colision.h"

```
void NF_InitCmapBuffers(void);
```

Inicializa los buffers para almacenar los archivos de mapas de colisiones.
Debes usar esta funcion una vez antes de cargar ningun mapa de colision.

```
void NF_ResetCmapBuffers(void);
```

Reinicia los buffers de mapas de colision, borrando todos los datos almacenados en la RAM. Es util usar esta funcion en los cambio de nivel, ya que elimina facilmente todos los datos y libera la RAM con una sola funcion.

```
void NF_LoadColisionMap(    const char* file,    // Nombre del archivo
                           u8 id,          // Slot donde almacenarlo
                           u16 width,      // Ancho del mapa (en pixeles)
                           u16 height     // Altura del mapa (en pixeles)
                           );
```

Carga un archivo de mapas de colisiones en la ram, en el slot indicado. Debes especificar el ancho y alto del mapa en pixeles. Recuerda que tu mapa debe ser 8 pixeles mas alto que tu fondo y usar esa primera fila de 8 pixeles para definir el tileset del mapa de colisiones.

Usa el comando "Convert_CMaps.bat" de la carpeta GRIT para convertir tu mapa de colisiones. Solo es necesario copiar el archivo ".cmp".

```
void NF_UnloadColisionMap(u8 id);
```

Descarga de la RAM el mapa de colisiones del slot indicado.
La ram usada por el mapa queda disponible, asi com el n° de slot.

```
u16 NF_GetTile(u8 slot,    // n° de Slot
               u16 x,      // Coordenada X en pixeles
               u16 y       // Coordenada Y en pixeles
               );
```

Devuelve el n° de tile (del tileset que has especificado en la primera fila del mapa) que se encuentra en las coordenadas dadas del mapa de colisiones cargado en el slot indicado.

```
void NF_SetTile(      u8 slot,      // n° de Slot
                     u16 x,        // Coordenada X en pixeles
                     u16 y,        // coordenada Y en pixeles
                     u16 value     // Valor a escribir (0 - 255)
                     );
```

Cambia el valor de un tile en el mapa de colisiones cargado en el slot seleccionado.

```
void NF_LoadColisionBg(      const char* file,      // Archivo
                             u8 id,                // Slot (0 - 31)
                             u16 width,            // Ancho del fondo
                             u16 height           // Altura del fondo
                             );
```

Carga un fondo de colisiones en la ram, en el slot indicado. Debes especificar el ancho y alto del mapa en pixeles. Recuerda que tu fondo de colisiones debe ser 8 pixeles mas alto que tu fondo real y usar esa primera fila de 8 pixeles para definir el tileset con los colores del fondo de colisiones.

Usa el comando "Convert_CMaps.bat" de la carpeta GRIT para convertir tu fondo de colisiones. Debes copiar los archivos ".cmp" y ".dat".

```
void NF_UnloadColisionBg(u8 id);
```

Descarga de la RAM el fondo de colisiones del slot indicado.
La ram usada por el fondo queda disponible, asi com el n° de slot.

```
u8 NF_GetPoint(      u8 slot,      // n° de slot (0 - 31)
                     s32 x,        // Coordenada X en pixeles
                     s32 y         // Coordenada Y en pixeles
                     );
```

Devuelve el n° de color (0 - 255) del pixel dado del fondo de colisiones especificado.
Si se especifica una coordenada fuera del mapa, devuelve 0.

#include "nf_sound.h"

```
void NF_InitRawSoundBuffers(void);
```

Iniciliaza los buffers y estructuras de datos para cargar sonidos en formato RAW. Debes de ejecutar esta funcion una vez antes de cargar ningun sonido en este formato. Recuerda de inicializar el motor de sonido de la DS con el comando de Libnds `soundEnable()`;

```
void NF_ResetRawSoundBuffers(void);
```

Reinicia todos los buffers de sonido. Esta funcion es util para vaciar todos los datos en un cambio de pantalla, etc.

```
void NF_LoadRawSound( const char* file,    // Nombre del archivo
                     u16 id,              // Slot donde se cargara (0 - 31)
                     u16 freq,            // Frecuencia de los samples (en Hz)
                     u8 format            // Formato de muestra (0 - 2)
                     );
```

Carga un archivo RAW a la RAM desde la FAT o EFS. Debes especificar el nombre del archivo sin extension, el slot donde lo guardaras (0 - 31), la frecuencia del sample (en Hz, por ejemplo 11025) y el formato de muestra (0 -> 8 bits, 1 -> 16 bits, 2 -> ADPCM).

Ejemplo:

```
NF_LoadRawSound("music", 1, 22050, 0);
```

Carga el archivo "music.raw" en el slot nº1. Este archivo esta codificado a 22050hz y 8 bits.

Para convertir los archivos a "RAW" os recomiendo el programa "Switch"
<http://www.nch.com.au/switch/plus.html> , el cual es gratuito. Os recomiendo convertir los archivos a formato "RAW", 8 bits signed a 11025hz o 22050hz. Eso si, en "Mono".

```
void NF_UnloadRawSound(u8 id);
```

Borra de la RAM el archivo cargado en el slot indicado. Debes especificar el slot a borrar (0 - 31).

```
u8 NF_PlayRawSound( u8 id,          // nº de slot donde esta cargado el sonido
                   u8 volume,       // Volumen (0 - 127)
                   u8 pan,           // Balance (0 - 64 - 127)
                   bool loop,        // Bucle ? (true / false)
                   u16 loopfrom      // Punto donde empieza el bucle
                   );
```

Reproduce un archivo de sonido cargado en el slot especificado. Debes especificar el volumen, el balance, si quieres que se repita automaticamente y de ser asi, el sample desde el que se repetira.
Esta funcion ademas devuelve el nº de canal que se le ha asignado al sonido.

Ejemplo:

```
NF_PlayRawSound(1, 127, 64, true, 0);
```

Reproduce el archivo de sonido del slot nº1, con volumen al 100% (127), con el balance centrado (64), que se repetirá automáticamente desde el primer sample.

Puedes usar el resto de funciones de sonido (pausa, stop, volumen, etc) que tiene Libnds directamente, ya que son lo suficientemente fáciles de usar.

<http://libnds.devskitpro.org/a00099.html>

#include "nf_bitmapbg.h"

```
void NF_Init16bitsBgBuffers(void);
```

Inicializa los buffers de almacenamiento de fondos de 16 bits.
Usa esta funcion una sola vez antes de usar estos buffers.

```
void NF_Reset16bitsBgBuffers(void);
```

Reinicia los buffers de 16 bits, vaciando su contenido de la RAM. Es util para asegurarse que se borra todo el contenido en cambios de nivel, etc.

```
void NF_Init16bitsBackBuffer(u8 screen    // Pantalla (0 - 1)
                             );
```

Inicializa el backbuffer de 16 bits de la pantalla indicada. Usa esta funcion una vez antes de habilitar el backbuffer.

```
void NF_Enable16bitsBackBuffer(    u8 screen    // Pantalla (0 - 1)
                                   );
```

Habilita el backbuffer para la pantalla seleccionada. Si el backbuffer ya estaba habilitado, el contenido del mismo sera borrado.

```
void NF_Disble16bitsBackBuffer(    u8 screen    // Pantalla (0 - 1)
                                   );
```

Deshabilita el backbuffer de la pantalla seleccionada, borrando el contenido del buffer y liberando la ram usada (128kb);

```
void NF_Flip16bitsBackBuffer(u8 screen    // Pantalla (0 - 1)
                             );
```

Copia el contenido del Backbuffer a la VRAM de la pantalla seleccionada, mostrando la imagen guardada en el.

```
void NF_InitBitmapBgSys(    u8 screen,    // Pantalla (0 - 1)
                           u8 mode      // Modo de color (0 - 1)
                           );
```

Inicializa la pantalla indicada de la DS en modo "bitmap", con la profundidad de color indicados (8 o 16 bits). El motor 2D debe estar inicializado en modo 5.
0 - 8 bits (256 colores)
1 - 16 bits

Ejemplo:

```
NF_InitBitmapBgSys(0, 1);
```

Inicializa la pantalla superior en modo "bitmap" a 16 bits de profundidad de color.

```
void NF_Load16bitsBg( const char* file,    // Archivo
                     u8 slot            // Slot donde guardarlo (0 - 15)
                     );
```

Carga desde la FAT o EFS un archivo de imagen de 16 bits en formato binario (*.img) de hasta 256x256 pixeles de tamaño (128kb). Debes convertir el archivo usando la siguiente linea de comandos en GRIT:

```
grit.exe archivo.ext -gb -gB16 -ftb
```

Puedes cargar como maximo el numero de archivos definidos en `#define NF_SLOTS_BG16B`.

Ejemplo:

```
NF_Load16bitsBg("bmp/bitmap16", 0);
```

Carga el archivo "bitmap16.img" en el slot n° 0

```
void NF_Unload16bitsBg(      u8 slot      // Slot (0 - 15)
                           );
```

Borra de la RAM la imagen cargada anteriormente en el slot indicado.

Ejemplo:

```
NF_Unload16bitsBg(0);
```

Borra de la RAM la imagen cargada en el slot 0. Es util cuando ya has copiado la imagen al backbuffer o a la VRAM y ya no la necesitas mas.

```
void NF_Copy16bitsBuffer(  u8 screen,      // Pantalla (0 - 1)
                          u8 destination,  // Destino (0 - 1)
                          u8 slot         // Slot (0 - 15)
                          );
```

Copia la imagen contenida en el buffer del slot indicado a la VRAM o BackBuffer de la pantalla seleccionada. Como destino, indica 0 para la VRAM o 1 para el BackBuffer.

Ejemplo:

```
NF_Copy16bitsBuffer(0, 1, 0);
```

Copia la imagen del slot 0 al BackBuffer de la pantalla superior.

```
void NF_Init8bitsBgBuffers(void);
```

Inicializa los buffers de almacenamiento de fondos de 8 bits.
Usa esta funcion una sola vez antes de usar estos buffers.

```
void NF_Reset8bitsBgBuffers(void);
```

Reinicia los buffers de 8 bits, vaciando su contenido de la RAM. Es util para asegurarse que se borra todo el contenido en cambios de nivel, etc.

```
void NF_Load8bitsBg(  const char* file,      // Archivo
                     u8 slot                // Slot donde guardarlo (0 - 15)
                     );
```

Carga desde la FAT o EFS un archivo de imagen de 8 bits en formato binario (*.img) de hasta 256x256 pixeles de tamaño (64kb) y su correspondiente paleta (*.pal). Debes convertir el archivo usando la siguiente linea de comandos en GRIT:

```
grit.exe archivo.ext -gb -gu8 -gB8 -pu8 -ftb -fh! -gTFF00FF
```

o si tienen que compartir la paleta

```
grit.exe archivo.ext -gb -gu8 -gB8 -pul6 -pS -ftb -fh! -Omypal.pal -gTFF00FF
```

Si quieres mostrar mas de un fondo en la misma pantalla, ambos deberan de compartir la paleta.

Puedes cargar como maximo el numero de archivos definidos en `#define NF_SLOTS_BG8B`.

Ejemplo:

```
NF_Load8bitsBg("bmp/bitmap8", 0);
```

Carga el archivo "bitmap8.img" y su paleta "bitmap8.pal" en el slot nº 0

```
void NF_Unload8bitsBg(u8 slot          // Slot (0 - 15)
                     );
```

Borra de la RAM la imagen cargada anteriormente en el slot indicado.

Ejemplo:

```
NF_Unload8bitsBg(0);
```

Borra de la RAM la imagen cargada en el slot 0. Es util cuando ya has copiado la imagen al backbuffer o a la VRAM y ya no la necesitas mas.

```
void NF_Copy8bitsBuffer(  u8 screen,      // Pantalla (0 - 1)
                          u8 destination,  // Destino (0 - 2)
                          u8 slot         // Slot (0 - 15)
                          );
```

Copia la imagen contenida en el buffer del slot indicado a la VRAM o BackBuffer de la pantalla seleccionada. Como destino, indica 0 para la capa 2 en VRAM, 1 para la capa 3 en VRAM o 2 para el BackBuffer.

Ejemplo:

```
NF_Copy8bitsBuffer(0, 1, 0);
```

Copia la imagen del slot 0 a la capa 3 de la pantalla superior.

```
void NF_Init8bitsBackBuffer( u8 screen      // Pantalla (0 - 1)
                             );
```

Inicializa el backbuffer de 8 bits de la pantalla indicada. Usa esta funcion una vez antes de habilitar el backbuffer.

```
void NF_Enable8bitsBackBuffer(      u8 screen      // Pantalla (0 - 1)
                                   );
```

Habilita el backbuffer para la pantalla seleccionada. Si el backbuffer ya estaba habilitado, el contenido del mismo sera borrado.

```
void NF_Disble8bitsBackBuffer(      u8 screen      // Pantalla (0 - 1)
                                   );
```

Deshabilita el backbuffer de la pantalla seleccionada, borrando el contenido del buffer y liberando la ram usada (64kb);

```
void NF_Flip8bitsBackBuffer( u8 screen,           // Pantalla (0 - 1)
                             u8 destination      // Capa de destino (0 - 1)
                             );
```

Copia el contenido del Backbuffer a la VRAM de la pantalla seleccionada, mostrando la imagen guardada en el. Puedes mandarlo a la capa 2 (0) o a la capa 3 (1), según el valor que le des al parametro "destination".

```
void NF_Load16bitsImage(      const char* file,    // Archivo
                              u8 slot,             // Slot (0 - 15)
                              u16 size_x,          // Ancho de la imagen (256 max)
                              u16 size_y          // Altura de la imagen (256 max)
                              );
```

Carga un archivo de imagen de 16 bits (*.img) de un maxmimo de 256x256 pixeles, en un slot en RAM. Especifica tambien las medidas de la imagen. La imagen se cargara en un slot de fondos de 16 bits.

Usa la funcion NF_Unload16bitsBg(); para borrarla de la RAM.

Ejemplo:

```
NF_Load16bitsImage("bmp/character", 1, 64, 128);
```

Carga el archivo "character" en el slot nº 1, el cual tiene unas dimensiones de 64 x 128 pixeles.

```
void NF_Draw16bitsImage(      u8 screen,          // Pantalla (0 - 1)
                              u8 slot,            // Slot (0 - 15)
                              s16 x,              // Posicion X
                              s16 y,              // Posicion Y
                              bool alpha          // Color 0xFF00FF transparente?
                              );
```

Dibuja la imagen cargada en el Slot seleccionado, en el BackBuffer de la pantalla seleccionada, en las coordenadas indicadas. Si el parametro "alpha" es verdadero, los pixeles de color 0xFF00FF (magenta) no seran dibujados.

Ejemplo:

```
NF_Draw16bitsImage(1, 1, 100, 50, true);
```

Dibuja la imagen del Slot nº1 en el BackBuffer de la pantalla inferior, en las coordenadas x:100, y:50.

```
#include "nf_media.h"
```

```
void NF_LoadBMP(      const char* file,      // Archivo
                      u8 slot                // Slot de 16 bits donde cargar la imagen
                      );
```

Carga un archivo BMP de 8, 16 o 24 bits en un slot de imágenes de 16 bits. Para cargar y mostrar la imagen, debes iniciar el modo de 16 bits, los backbuffers correspondientes y usar la función `NF_Draw16bitsImage()`; para mandarla del slot en RAM al BackBuffer. Todos los píxeles fuera de la pantalla, serán ignorados.

Ejemplo:

```
NF_LoadBMP("bmp/lostend", 0);
```

Carga el archivo "lostend.bmp" en el slot de 16 bits nº 0.

#include "nf_affine.h"

```
void NF_InitAffineBgSys(    u8 screen    // Pantalla (0 - 1)
                          );
```

Inicializa el sistema de fondos "Affine" (rotacion y escalado) para la pantalla seleccionada. Este modo es exclusivo, solo se pueden cargar fondos del tipo affine una vez inicializado y solo en las capas 2 y 3. Ademas estos fondos no pueden tener mas de 256 tiles cada uno y deben compartir la paleta, con un maximo de 256 colores. El motor 2D debe estar inicializado en modo 2.

Ejemplo:

```
NF_InitAffineBgSys(0);
```

Inicializa el modo Affine para la pantalla superior.

```
void NF_LoadAffineBg( const char* file,    // Archivo
                     const char* name,    // Nombre del fondo
                     u16 width,           // Ancho en pixeles
                     u16 height           // Alto en pixeles
                     );
```

Carga un fondo "affine" en la memoria RAM desde la FAT o NitroFS. Es imprescindible inicializar los buffers de fondos tileados antes de poder cargar un fondo "affine". Consulta la seccion `#include "nf_tiledbg.h"` para mas informacion acerca de la funcion `NF_InitTiledBgBuffers()`;

Los fondos "affine" tienen que ser obligatoriamente de 256x256 o 512x512 pixeles y con un tileset de 256 tiles maximo. Todos los fondos para una misma pantalla deben de compartir la paleta. Usa el bat `Convert_Affine.bat` de la carpeta GRIT para convertir tus fondos.

Ejemplo:

```
NF_LoadAffineBg("bg/waves512", "waves", 512, 512);
```

Carga el fondo "waves512" de la carpeta bg, asigne el nombre "waves" y especifica que el fondo es de 512 x 512 pixeles.

```
void NF_UnloadAffineBg(    const char* name    // Nombre del fondo
                          );
```

Borra de la RAM el fondo affine especificado. Es una simple llamada a la funcion `NF_UnloadTiledBg()`;

Ejemplo:

```
NF_UnloadAffineBg("waves"); borra de la RAM el fondo "waves".
```

```
void NF_CreateAffineBg(    u8 screen,           // Pantalla (0 -1)
                          u8 layer,             // Capa (2 - 3)
                          const char* name,     // Nombre
                          u8 wrap                // Wrap (0 - 1)
                          );
```

Crea un fondo affine en la pantalla y capa especificadas, usando los graficos previamente cargados en RAM. Debes especificar si quieres que el fondo sea infinito (Wrap 1) o no (Wrap 0).

Ejemplo:

```
NF_CreateAffineBg(0, 3, "waves", 1);
```

Crea un fondo en la pantalla 0, capa 3, usando los graficos del fondo "waves", con la opcion "wrap arround" (fondo infinito) habilitada.

```
void NF_DeleteAffineBg(    u8 screen,    // Pantalla (0 - 1)
                          u8 layer      // Capa (2 - 3)
                          );
```

Elimina de la VRAM el fondo de la pantalla y capas especificado.

Ejemplo:

```
NF_DeleteAffineBg(0, 3);
```

Borra el fondo de la pantalla superior en la capa 3.

```
void NF_AffineBgTransform(  u8 screen,    // Pantalla (0 - 1)
                           u8 layer,    // Capa (2 - 3)
                           s32 x_scale,  // Escala X (0 - 256 - >512)
                           s32 y_scale,  // Escala Y (0 - 256 - >512)
                           s32 x_tilt,   // Inclinacion X (0 - >512)
                           s32 y_tilt    // Inclinacion Y (0 - >512)
                           );
```

Modifica la matriz de transformacion del fondo especificado con los parametros dados. Pues modificar la escala en los ejes X e Y, asi como la inclinacion de dichos ejes.

Ejemplo:

```
NF_AffineBgTransform(0, 3, 512, 512, 0, 0);
```

Zoom del fondo de la pantalla superior, capa 3, al 50% de su tamaño.

```
void NF_AffineBgMove( u8 screen,    // Pantalla (0 - 1)
                     u8 layer,    // Capa (2 - 3)
                     s32 x,        // Posicion X
                     s32 y,        // Posicion Y
                     s32 angle     // Angulo de rotacion (-2048 / 2048)
                     );
```

Mueve el fondo affine a la posicion especificada. Puedes tambien especificar la rotacion de este fondo (entre -2048 a 2048). Los fondos affine no se pueden mover con la funcion NF_ScrollBg();

Ejemplo:

```
NF_AffineBgMove(0, 3, 128, 96, 256);
```

Mueve el fondo de la pantalla superior en la capa 3 a las coordenadas X128, Y96 y rotalo 45° a la derecha.

```
void NF_AffineBgCenter(    u8 screen,    // Pantalla (0 - 1)
                           u8 layer,    // Capa (2 - 3)
                           s32 x,        // Posicion X
                           s32 y        // Posicion Y
                           );
```

Define el centro de rotacion del fondo affine especificado.

Ejemplo:

```
NF_AffineBgCenter(0, 3, 128, 128);
```

Define el centro de rotacion del fondo affine de la pantalla superior, capa 3, en las coordenadas X128, Y128.

```
#include "nf_3d.h"
```

```
void NF_Set3D( u8 screen,    // Pantalla (0 - 1)
              u8 mode      // Modo (0, 2, 5)
              );
```

Inicia el modo 3D para la pantalla seleccionada.

Mode	Configuration
0	Fondos tileados a 256 colores.
2	Fondos affine tileados de 8 bits en las capas 2 y 3
5	Fondos bitmap a 8 o 16 bits.

Los objetos 3D se renderizaran en la capa 0. Si se especifica la pantalla 1 para 3D, los numeros de pantalla para todos los objetos 2D (fondos, sprites, etc) se invierten, siendo la pantalla superior la 1 y la inferior la 0.
Es necesario usar esta funcion antes de poder usar los 3dSprites.

Ejemplo:

```
NF_Set3D(1, 0);
```

Inicia el modo 3D para fondos tileados y sprites en la pantalla 1 (inferior).

```
void NF_InitOpenGL(void);
```

Inicializa y configura el OpenGL para el uso de las funciones 3dSprites de la librería. Esta funcion es llamada automaticamente por la funcion NF_Init3dSpriteSys(); por lo que no es necesario que la uses.

#include "nf_sprite3d.h"

Estas funciones son especiales, dado que usan el motor 3d de la consola para crear sprites a base de poligonos texturizados. Solo se pueden usar en una pantalla a la vez, perdemos la capa 0 de fondos, pero en contrapartida podemos crear hasta 256 sprites de un tamaño maximo de 1024x1024, pudiendo usar cualquier tamaño en base 2, y usar un maximo de 32 paletas simultaneamente.

Para la carga de graficos y paletas, se usaran las mismas funciones que los sprites 2D. Puedes convertir las imágenes indexadas de 256 colores en texturas para usarlas como 3dSprites con el siguiente comando de grit:

```
grit.exe imagen.bmp -gb -gu8 -gB8 -pu8 -ftb -fh! -gTFF00FF
```

O usando los bats para convertir bitmaps de 8 bits.

```
void NF_Init3dSpriteSys(void);
```

Inicializa el sistema de 3dSprites.
Asigna 128kb de VRAM para texturas y 16kb para paletas.
Habilita el uso de las paletas extendidas.

Ejemplo:

```
NF_InitSpriteSys();
```

Inicializa el sistema de 3dSprites a 256 colores.

```
void NF_Vram3dSpriteGfx(u16 ram,          // Slot en RAM del grafico (0 - 255)
                        u16 vram,         // Slot en VRAM del grafico (0 - 255)
                        bool keepframes    // Copia a la VRAM solo el primer frame
                        );
```

Copia un grafico de la RAM a la VRAM, para poder usarlo en un 3dSprite. Debes indicar el slot de origen en la RAM (0 - 255), el slot de destino en la VRAM (0 - 255) y en el caso de ser animado, si se deben de mantener los frames adicionales en la RAM (true) o copiarlos todos a la VRAM (false).

Ejemplo:

```
NF_Vram3dSpriteGfx(160, 23, false);
```

Copia el grafico del slot 160 de la RAM al slot 23 de la VRAM, copiando todos los frames, si este es animado.

```
void NF_Free3dSpriteGfx(u16 id           // Slot en VRAM (0 - 255)
                        );
```

Borra de la VRAM el grafico del slot seleccionado.
Evita borrar el grafico si algun sprite lo esta usando, ya que este se volvera invisible o aparecera corrompido.

Ejemplo:

```
NF_Free3dSpriteGfx(34);
```

Borra de la VRAM el grafico del slot n°34.

```
void NF_Vram3dSpriteGfxDefrag(void);
```

Desfragmenta la libre VRAM usada por los graficos de los Sprites. Este proceso se ejecuta automaticamente cuando la VRAM libre fragmentada es superior en un 50% de la VRAM libre total. No es necesario que nunca ejecute manualmente este comando.

Puedes consultar el estado de la VRAM para sprites leyendo el valor de las siguientes variables:

```
NF_TEXVRAM.free           <- Memoria VRAM total libre
NF_TEXVRAM.fragmented    <- Memoria VRAM libre fragmentada
NF_TEXVRAM.inarow        <- Memoria VRAM libre, ultimo bloque mas grande
NF_TEXVRAM.lost          <- Memoria VRAM libre no usable por fragmentacion
```

```
void NF_Vram3dSpritePal(    u8 id,           // Slot en RAM de la paleta (0 - 64)
                           u8 slot         // Slot en VRAM de la paleta (0 - 31)
                           );
```

Copia la paleta de la RAM al Slot de paletas extendidas de la VRAM. Si el slot esta en uso, el contenido se sobrescribe.

Ejemplo:

```
NF_VramSpritePal(56, 8);
```

Copia la paleta del slot en RAM n°56 al slot de paletas extendidas n°8.

```
void NF_Create3dSprite(    u8 id,           // n° de Sprite (0 - 255)
                           u16 gfx,         // Slot del grafico que usara (0 - 255)
                           u8 pal,         // Slot de la paleta que usara (0 - 31)
                           s16 x,         // Coordenada X donde se creara
                           s16 y         // Coordenada Y donde se creara
                           );
```

Crea un 3dSprite en la pantalla, usando el grafico y paletas dadas. Debes especificar un numero unico de Sprite ente 0 y 255, asi como las coordenadas donde deseas crearlo. El numero de sprite asigna la prioridad del mismo, siendo 0 la mas alta (se dibuja el ultimo, quedando encima), siempre y cuando reordenes la cola de 3dSprites.

Ejemplo:

```
NF_Create3dSprite(12, 30, 1, 100, 50);
```

Crea un Sprite en la pantalla, usando el grafico del slot 30, la paleta del slot 1 y dandole como identificador el n°12. El sprite se creara en las coordenadas x:100, y:50

```
void NF_Delete3dSprite(u8 id);
```

Borra de la pantalla indicada el 3dSprite con el numero dado. El grafico usado y la paleta no se borran de la VRAM, solo se elimina el Sprite.

Ejemplo:

```
NF_Delete3dSprite(12);
```

Borra de la pantalla el sprite con el identificador n°12.

```
void NF_Sort3dSprites(void);
```

Reordena el orden de dibujado de los 3dSprites creados por su ID, siendo la mas baja la que tiene prioridad.

```
void NF_Set3dSpritePriority( u16 id,          // ID del Sprite (0 - 255)
                             u16 prio        // Prioridad (0 - 255)
                             );
```

Cambia la prioridad de dibujo del 3dSprite con la ID indicada, siendo la mas baja la que tiene mas prioridad.

```
void NF_Swap3dSpritePriority(u16 id_a,        // ID Sprite A
                             u16 id_b        // ID Sprite B
                             );
```

Intercambia la prioridad entre dos 3dSprites.

```
void NF_Move3dSprite( u8 id,                // Id. del Sprite (0 - 255)
                      s16 x,                // Posicion X
                      s16 y,                // Posicion Y
                      );
```

Mueve un 3dSprite a la posicion dada.

Ejemplo:

```
NF_Move3dSprite(35, 100, 50);
```

Mueve el 3dSprite n°35 a las coordenadas x:100, y:50

```
void NF_Show3dSprite( u8 id,                // Id. del Sprite (0 - 255)
                      bool show            // Visibilidad
                      );
```

Muestra o oculta un 3dSprite. Si se oculta, este se vuelve invisible, sin borrarse.

Ejemplo:

```
NF_Show3dSprite(35, false);
```

Oculta el 3dSprite n°35.

```
NF_Show3dSprite(45, true);
```

Haz visible el 3dSprite n°45, previamente ocultado.

```
void NF_Set3dSpriteFrame(   u8 id,          // Id. del Sprite (0 - 255)
                             u8 frame        // Frame a mostrar
                             );
```

Selecciona que frame de la animacion de un 3dSprite sera mostrado.

Ejemplo:

```
NF_Set3dSpriteFrame(20, 5);
```

El Sprite n°20 mostrara el frame n°5.

```
void NF_Draw3dSprites(void);
```

Dibuja en la pantalla todos los 3dSprites creados.
Es necesario usar esta funcion una vez por ciclo para que se muestren en pantalla los 3dSprites.

Este es el codigo basico para mostrarlos en pantalla:

```
// Dibuja los 3D Sprites
NF_Draw3dSprites();
// Actualiza la escena 3D, si no lo haces, no se mostrara en pantalla
glFlush(0);
// Espera al sincronismo vertical
swiWaitForVBlank();
```

```
void NF_Update3dSpritesGfx(void);
```

Actualiza si es necesario las texturas de los 3dSprites animados.
Usala si algunos de los 3dSprites tienen el flag KEEPPFRAMES == TRUE.
Usa esta funcion justo despues de swiWaitForVBlank();

```
void NF_Rotate3dSprite(    u16 id,        // ID del Sprite (0 - 255)
                          s16 x,          // Rotacion X (-512/0/512)
                          s16 y,          // Rotacion Y (-512/0/512)
                          s16 z,          // Rotacion Z (-512/0/512)
                          );
```

Rota el 3dSprite sobre los 3 ejes disponibles. Puedes especificar una rotacion de entre -512 y 512, siendo 0 el punto central (sin rotacion).

```
void NF_Scale3dSprite(u16 id,        // ID del Sprite
                     u16 x,          // Escala X (0/64/512)
                     u16 y,          // Escala Y (0/64/512)
                     );
```

Escala el 3dSprite sobre los ejes X e Y. El rango de escalado va de 0 a 512, siendo 64 el punto central (escala 100%).

```
void NF_Blend3dSprite(u8 sprite,     // ID del Sprite (0 - 255)
                     u8 poly_id,     // ID de poligono (1 - 62)
                     u8 alpha        // Transparencia (0 - 31)
                     );
```

Habilita y cambia el nivel de alpha de el sprite 3d indicado. Para que la transparencia sea efectiva entre Sprites, debes especificar un poly_id diferente para cada sprite (entre 1 y 62). El rango de alpha es de 0 a 31, siendo 31 opaco. Para eliminar la transparencia, selecciona un valor para alpha de 31 o especifica como poly_id el n° 0.

```
void NF_3dSpritesLayer(    u8 layer    // Capa
                          );
```

Selecciona la capa en la que se dibujaran los Sprites 3D. (0 - 3)
En realidad los Sprites 3D siempre se dibujan sobre la CAPA 0, esta funcion solo cambia la prioridad de esta capa sobre las demas.

```
void NF_3dSpriteEditPalColor(u8 pal,      // Paleta (0 - 31)
                             u8 number,    // n° de color en la paleta (0 - 255)
                             u8 r,        // Valor del componente R (0 - 31)
                             u8 g,        // Valor del componente G (0 - 31)
                             u8 b,        // Valor del componente B (0 - 31)
                             );
```

Cambia el valor de un color de la paleta de sprites especificada. El cambio se realiza sobre la copia en RAM de la paleta, por lo que los cambios no seran visibles hasta que la actualices en VRAM usando la funcion NF_3dSpriteUpdatePalette(); Usala para realizar efectos sobre tus sprites.

Ejemplo:

```
NF_3dSpriteEditPalColor(3, 1, 31, 0, 0);
```

Cambia el color n°1 de la paleta de sprites n°3 a color rojo.

```
void NF_3dSpriteUpdatePalette(      u8 pal      // Paleta (0 - 31)
                                   );
```

Actualiza en VRAM la paleta de sprites especificada con la copia que se encuentra en RAM.

Ejemplo:

```
NF_3dSpriteUpdatePalette(2);
```

Actualiza la paleta de sprites n°2.

```
void NF_3dSpriteGetPalColor( u8 pal,      // Paleta (0 - 31)
                             u8 number,    // n° de color (0 - 255)
                             u8* r,        // Componente R (0 - 31)
                             u8* g,        // Componente G (0 - 31)
                             u8* b,        // Componente B (0 - 31)
                             );
```

Obtiene los valores RGB de un color de la paleta de sprites que se encuentra cargada en VRAM, de su copia original en la RAM.

Ejemplo:

```
u8 rojo;
u8 verde;
u8 azul;
NF_3dSpriteGetPalColor(3, 200, &rojo, &verde, &azul);
```

Obtiene el valor RGB del color n°200 de la paleta n°3 y guardalo en las variables "rojo", "verde" y "azul".

```
void NF_3dSpriteSetDeep(      u8 id,      // n° de Sprite (0 - 255)
                              s16 z      // Profundidad (-512/0/512)
                              );
```

Establece la profundidad en el eje Z para el 3dSprite seleccionado, siendo -512 el punto mas cercano, 0 el centro por defecto y 512 el punto mas alejado. Cambiar la profundidad tiene como fin evitar que el sprite se entrecruce con otros sprites al aplicarle rotacion o zoom. Cambiar la profundidad del Sprite tambien altera a la prioridad que este tiene en pantalla.

```
#include "nf_mixedbg.h"
```

```
void NF_InitMixedBgSys(u8 screen);    // Pantalla (0 - 1)
```

Inicializa el modo mixto para fondos (Tiled BG + Bitmap 8 bits)

Capas 0 a 2 - Tiled (64kb, 48kb para tiles, 16kb para mapas).

Capa 3 - Bitmap 8 bits (64kb).

Se pueden usar todas las funciones de los dos tipos de fondos.
