

# NGN\_LOAD.H

**NGN\_TextureData\* Texture(std::string filepath);**

Carga una textura en formato PNG.

```
NGN_TextureData* bg_grid = ngn->load->Texture("data/png/grid.png");
```

**\* Nota:** El tamaño máximo de textura es de 8192x8192 píxeles.

**NGN\_TiledBgData\* TiledBg(std::string filepath);**

Carga un fondo de tiles en formato .tbg (Ver utilidades adjuntas).

```
NGN_TiledBgData* tiles_bg_front = ngn->load->TiledBg("data/bg/bg_front.tbg");
```

**\* Nota:** El tamaño máximo del tileset es de 8192x8192 píxeles.

**NGN\_SpriteData\* Sprite(std::string filepath);**

Carga un Sprite en formato .spr (Ver utilidades adjuntas).

```
NGN_SpriteData* wizard_sprite = ngn->load->Sprite("data/spr/wizard.spr");
```

**\* Nota:** El tamaño máximo de cada fotograma es de 8192x8192 píxeles.

**NGN\_CollisionMapData\* CollisionMap(std::string filepath);**

Carga un mapa de colisiones en formato .map (Ver utilidades adjuntas)

```
NGN_CollisionMapData* collision_map = ngn->load->CollisionMap("data/collision/mainmap.map");
```

**NGN\_AudioClipData\* AudioClip(std::string filepath);**

Carga un archivo de audio en formato WAV, FLAC o OGG para usarlo como efecto de sonido. Devuelve FALSE en caso de error en la carga del archivo.

```
NGN_AudioClipData* coin_sfx = ngn->load->AudioClip("data/wav/coin.wav");
```

```

NGN_TextFont* TrueTypeFont(
    std::string filepath,      // Archivo a cargar
    uint32_t height,         // Altura de la fuente (en pixeles)
    bool antialias = true,    // Antialias?
    uint32_t font_color = 0xFFFFFFFF,    // Color base
    uint32_t outline = 0,     // Borde? (en pixeles)
    uint32_t outline_color = 0x000000 // Color del borde
);

```

Carga y convierte a BITMAP un archivo de fuente tipográfica TRUE TYPE con la altura y propiedades especificadas. Devuelve NULL en caso de no poder cargar o convertir la fuente.

```

NGN_TextFont* font = ngn->load->TrueTypeFont("data/consolas.ttf", 24);

```

```

NGN_RawImage* PngAsRaw(std::string filepath);

```

Carga una imagen en formato PNG y devuelve los pixeles de la misma en formato RAW. Devuelve NULL en caso de error.

```

NGN_RawImage* pixels = ngn->load->PngAsRaw("data/clouds.png");

```

```

NGN_RawImage* SpriteAsRaw(
    std::string filepath,      // Archivo a cargar
    uint32_t frame = 0         // Fotograma a convertir
);

```

Carga un sprite y devuelve los pixeles del fotograma especificado (el primero por defecto) formato RAW. Devuelve NULL en caso de error.

```

NGN_RawImage* pixels = ngn->load->SpriteAsRaw("data/coin.spr", 3);

```

```

bool SpriteAsRawVector(
    // Archivo a cargar
    std::string filepath,
    // Vector de destino con los frames
    std::vector<NGN_RawImage*> &raw_frames,
    // Frame inicial (0 por defecto)
    uint32_t first_frame = 0,
    // Frame final (ultimo por defecto)
    uint32_t last_frame = NGN_DEFAULT_VALUE
);

```

Carga un sprite y almacena los pixeles de los fotogramas especificados (todos los fotogramas por defecto) formato RAW dentro de un vector. Devuelve TRUE en caso de éxito y FALSE en caso de error.

```

std::vector<NGN_RawImage*> frames;
ngn->load->SpriteAsRawVector("data/coin.spr", frames, 2, 5);

```

```

int32_t LoadFile(
    std::string filepath,           // Archivo
    std::vector<uint8_t> &data      // Vector de datos
);

```

Lee el archivo de la ruta especificada desde el sistema de archivos del dispositivo o desde un paquete de archivos (ver SetDisk() y SetPackage()) y coloca los datos en el vector dado. Si se lee el archivo desde un paquete y este está encriptado, se desencriptará el contenido. El método devuelve el número de bytes leídos o -1 en caso de error.

```

std::vector<uint8_t> buffer;
int32_t file_length = ngn->load->LoadFile("data/stage.bin", buffer);

```

```

void SetDisk();

```

A partir de ese momento, establece el sistema de archivos del dispositivo como el origen de los datos en los métodos de carga.

```

ngn->load->SetDisk();

```

```

bool SetPackage(std::string pkg_file, std::string key = "");

```

A partir de ese momento, establece un paquete de archivos creado con la herramienta NGN\_FileSystem como el origen de los datos en los métodos de carga. Si los archivos del paquete están encriptados, deberá de proporcionarse la clave de encriptación en el segundo parámetro.

```

ngn->load->SetPackage("gamedata.pkg", "myawesomekey");

```