NGN_RESOURCES.H

El sistema de recursos de N'gine permite una gestión rápida y eficiente, mediante la creación de repositorios, de todos los recursos que puede necesitar nuestro programa. Una vez creado un repositorio, se pueden cargar en él todos los recursos necesarios mediante un listado en formato de texto plano. Un ejemplo de este archivo sería:

```
# Resources loading template file


# Usage: Regular files
# TYPE; INTERNAL_NAME; FILE_PATH
# Example:
# TEX; logo; data/logo_overlay.png

# Usage: Font files
# FNT; INTERNAL_NAME; FILE_PATH; FONT_HEIGHT; [ANTIALIAS]; [BASE_COLOR];
[BORDER_SIZE]; [BORDER_COLOR]
# Example:
# FNT; mono_16; data/monofonto.ttf; 16
# FNT; mono_24; data/monofonto.ttf; 24; 1; 0xFFFFFF; 2; 0x202020

# Admited filetypes
# TEX  Textures (.png)
# TBG  Tiled backgrounds (.tbg)
# SPR  Sprites (.spr)
# MAP  Collision maps (.map)
# SFX  Sound Effects (.wav) (.ogg)
# TXT  Plain text files (.txt)Ç
# FNT  Font files (.ttf) (.otf)



# --- Resources loading ---


# Textures
TEX; logo; data/logo_overlay.png

# Sprites
SPR; aim; data/aim.spr
SPR; bird; data/bird_small.spr

# Tiled backgrounds
TBG; bg0; data/bg0.tbg
TBG; bg1; data/bg1.tbg
TBG; bg2; data/bg2.tbg

# Sound Effects
SFX; shutter; data/shutter.wav

# Fonts
FNT; mono_16; data/monofonto.ttf; 16; 1; 0xFFFFFF; 1; 0x202020
```

Se puede usar el carácter “#” para introducir comentarios en el archivo.
El máximo número de repositorios simultáneos es de 127.

void AddRepository(std::string repo_name);

Crea un nuevo repositorio con el nombre indicado. Si se intenta crear un repositorio con un nombre ya existente, la orden se ignorará.

```
ngn->resources->AddRepository("local");
```

void RemoveRepository(std::string repo_name);

Elimina el repositorio con el nombre indicado, así como los datos que pueda contener. Si el repositorio no existe, la orden se ignorará.

```
ngn->resources->RemoveRepository("local");
```

void Clear(std::string repo_name);

Elimina los datos contenidos en repositorio con el nombre indicado. Si el repositorio no existe, la orden se ignorará.

```
ngn->resources->Clear("local");
```

bool Load(std::string repo_name, std::string filelist);

Carga los recursos indicados en un listado en formato de texto plano en el repositorio indicado. Si el proceso de carga falla, devuelve FALSE.

```
if (!ngn->resources->Load("local", "data/resources.txt")) return false;
```

NGN_TextureData* GetTexture(
    std::string repo_name,
    std::string resource_name
);

NGN_TiledBgData* GetTiledbg(
    std::string repo_name,
    std::string resource_name
);

NGN_SpriteData* GetSprite(
    std::string repo_name,
    std::string resource_name
);

```
NGN_CollisionMapData* GetCmap(
    std::string repo_name,
    std::string resource_name
);

NGN_AudioClipData* GetSfx(
    std::string repo_name,
    std::string resource_name
);

std::vector<std::string> GetTxt(
    std::string repo_name,
    std::string resource_name
);

NGN_TextFont* GetTypeface(
    std::string repo_name,
    std::string resource_name
);
```

Devuelve el puntero de memoria al recurso con el nombre solicitado, o un vector con el contenido en el caso de los archivos de texto. Si el recurso o el repositorio no se encuentran, devuelve NULL o un vector vacío.

```
NGN_Texture* bg = new NGN_Texture(ngn->resources->GetTexture("local", "background"), 0, 0);
NGN_SpriteData* bullet = ngn->resources->GetSprite("local", "arrow");
```

Se han añadido, además, las siguientes sobrecargas en las clases NGN_Texture, NGN_TiledBg, NGN_Sprite, NGN_Sound y NGN_TextLayer para el uso directo de repositorios en la creación de los objetos de dichas clases.

```
NGN_Texture(
    std::string repo_name,            // Nombre del repositorio
    std::string resource_name,        // Nombre del recurso
    int32_t position_x,               // Posición X inicial
    int32_t position_y,               // Posición Y inicial
    uint32_t texture_width,           // Ancho de la textura
    uint32_t texture_height           // Altura de la textura
);

NGN_TiledBg(
    std::string repo_name,            // Nombre del repositorio
    std::string resource_name,        // Nombre del recurso
    int32_t position_x,               // Posición X del fondo
    int32_t position_y                // Posición Y del fondo
);
```

```
NGN_Sprite(
    std::string repo_name,        // Nombre del repositorio
    std::string resource_name,    // Nombre del recurso
    int32_t position_x,           // Posición X inicial
    int32_t position_y,           // Posición Y inicial
    uint32_t sprite_width,        // Ancho del Sprite
    uint32_t sprite_height,       // Altura del Sprite
    uint32_t box_width,           // Ancho de la caja de colisiones
    uint32_t box_height,          // Alto de la caja de colisiones
    int32_t box_offset_x,         // Offset horizontal de la caja
    int32_t box_offset_y          // Offset vertical de la caja
);

NGN_TextLayer(
    std::string repo_name,        // Nombre del repositorio
    std::string resource_name,    // Nombre del recurso
    std::string bg_name,          // Nombre de la textura de fondo
    int32_t position_x,           // Posición X
    int32_t position_y,           // Posición Y
    uint32_t width,               // Ancho de la capa
    uint32_t height,              // Alto de la capa
    bool filtering                // Filtrado del contenido?
);

NGN_AudioClip* PlaySfx(
    std::string repo_name,        // Nombre del repositorio
    std::string resource_name,    // Nombre del recurso
    int32_t volume,               // Volumen
    int32_t panning,              // Panning
    bool loop,                    // Loop ?
    uint8_t mixer_channel         // Canal por defecto en el mixer
);
```