

NGN_SOUND.H

```
NGN_AudioClip* PlaySfx(  
    NGN_AudioClipData* sound,  
        // Clip de audio  
    int32_t volume = 100,  
        // Volumen  
    int32_t panning = 0,  
        // Panning (-100 a 100)  
    bool loop = false,  
        // Loop ?  
    uint8_t mixer_channel = MIXER_EFFECTS_CH  
        // Canal por defecto en el mixer  
);
```

Reproduce un clip de audio cargado previamente en RAM con el comando `ngn->load->AudioClip()`; Devuelve la referencia de la instancia creada y lo añade a la cola de reproducción.

```
NGN_AudioClipData* coin_sfx = ngn->load->AudioClip("data/wav/coin.wav");  
ngn->sound->PlaySfx(coin_sfx);    // Reproducción simple  
NGN_AudioClip* my_sound = ngn->sound->PlaySfx(coin_sfx, 100, false);    // Avanzada  
ngn->sound->SfxVolume(my_sound, 50);
```

```
void ResumeSfx(NGN_AudioClip* sound);
```

Continúa la reproducción de un efecto de sonido.

```
ngn->sound->ResumeSfx(my_sound);
```

```
void PauseSfx(NGN_AudioClip* sound);
```

Pausa un efecto de sonido.

```
ngn->sound->PauseSfx(my_sound);
```

```
void StopSfx(NGN_AudioClip* sound);
```

Detén la reproducción de un efecto de sonido. Esto lo eliminara de manera automática de la cola de reproducción.

```
ngn->sound->StopSfx(my_sound);
```

```
void SfxVolume(NGN_AudioClip* sound, int32_t volume = 100);
```

Cambia el nivel de volumen de un sonido (0 - 100).

```
ngn->sound->SfxVolume(my_sound, 50);
```

int32_t SfxGetVolume(NGN_AudioClip* sound);

Devuelve el nivel actual de volumen de un sonido.

```
uint32_t vol = ngn->sound->SfxGetVolume(my_sound);
```

void SfxPitch(NGN_AudioClip* sound, float pitch = 1.0f);

Cambia la velocidad de reproducción y frecuencia de un sonido. (Usar 1.0f como valor nominal).

```
ngn->sound->SfxPitch(my_sound, 1.2f);
```

float SfxGetPitch(NGN_AudioClip* sound);

Devuelve el valor actual del PITCH de un sonido.

```
float my_pitch = ngn->sound->SfxGetPitch(my_sound);
```

void SfxLoop(NGN_AudioClip* sound, bool loop = true);

Establece si un sonido debe repetirse al finalizar.

```
ngn->sound->SfxLoop(my_sound, true);
```

bool SfxGetLoop(NGN_AudioClip* sound);

Devuelve el estado de repetición de un sonido.

```
bool repeat = ngn->sound->SfxGetLoop(my_sound);
```

void SfxPanning(NGN_AudioClip* sound, int32_t panning = 0);

Establece el efecto “panning” de un sonido (-100 izquierda, 0 centro, 100 derecha).

```
ngn->sound->SfxPanning(my_sound, -66);
```

int32_t SfxGetPanning(NGN_AudioClip* sound);

Devuelve nivel de “panning” de un sonido.

```
int32_t pan = ngn->sound->SfxGetPanning(my_sound);
```

bool SfxIsPlaying(NGN_AudioClip* sound);

Devuelve el estado de reproducción de un sonido.

```
bool play = ngn->sound->SfxIsPlaying(my_sound);
```

```
bool SfxIsAlive(NGN_AudioClip* sound);
```

Devuelve si aún existe un sonido.

```
bool alive = ngn->sound->SfxIsAlive(my_sound);
```

```
NGN_MusicClip* OpenMusic(  
    const char* filepath,  
        // Archivo de audio  
    bool auto_start = true,  
        // Reproducción automática  
    int32_t volume = 100,  
        // Volumen  
    bool loop = true,  
        // Loop ?  
    uint8_t mixer_channel = MIXER_MUSIC_CH  
        // Canal por defecto en el mixer  
);
```

Abre y opcionalmente, reproduce un archivo de audio por streaming en formato WAV, OGG o FLAC. Devuelve la referencia de la instancia creada y lo añade a la cola de reproducción. Opcionalmente, se puede asignar el canal del mezclador de audio.

```
NGN_MusicClip* bgm = ngn->sound->OpenMusic("data/ogg/stage01.ogg");
```

```
NGN_MusicClip* OpenMusic(  
    const char* filepath,  
        // Archivo de audio  
    int32_t loop_start,  
        // Inicio del loop (milisegundos)  
    int32_t loop_end = NGN_DEFAULT_VALUE,  
        // Fin del loop (milisegundos)  
    bool auto_start = true,  
        // Reproducción automática  
    int32_t volume = 100,  
        // Volumen  
    uint8_t mixer_channel = MIXER_MUSIC_CH  
        // Canal por defecto en el mixer  
);
```

Abre y opcionalmente, reproduce un archivo de audio por streaming en formato WAV, OGG o FLAC. Es posible establecer los puntos de “loop” de inicio y final (en milisegundos) si es necesario. Devuelve la referencia de la instancia creada y lo añade a la cola de reproducción. Opcionalmente, se puede asignar el canal del mezclador de audio.

```
NGN_MusicClip* bgm = ngn->sound->OpenMusic("data/ogg/stage01.ogg", 13721);
```

void CloseMusic(NGN_MusicClip* music);

Cierra un stream de audio y lo elimina de la cola de reproducción.

```
ngn->sound->CloseMusic(bgm);
```

void PlayMusic(

```
    NGN_MusicClip* music,      // Clip de música
    int32_t volume = 100,      // Volumen
    bool loop = true           // Loop ?
);
```

Inicia o continua la reproducción de un archivo de audio abierto previamente, pudiendo especificar nuevos valores de volumen y repetición.

```
ngn->sound->PlayMusic(bgm, 50, false);
```

void ResumeMusic(NGN_MusicClip* music);

Continúa con la reproducción de una música pausada previamente. Si no estaba en pausa, se inicia la reproducción desde el principio.

```
ngn->sound->ResumeMusic(bgm);
```

void PauseMusic(NGN_MusicClip* music);

Pone en pausa una música indicada.

```
ngn->sound->PauseMusic(bgm);
```

void StopMusic(NGN_MusicClip* music);

Detiene la reproducción de la música indicada. Esta no será eliminada de cola de reproducción.

```
ngn->sound->StopMusic(bgm);
```

void MusicVolume(

```
    NGN_MusicClip* music,
    int32_t volume = 100
);
```

Cambia el nivel de volumen (0 - 100) de la música indicada.

```
ngn->sound->MusicVolume(bgm, 75);
```

int32_t MusicGetVolume(NGN_MusicClip* music);

Devuelve el nivel actual de la música indicada.

```
uint32_t vol = ngn->sound->MusicGetVolume(bgm);
```

void MusicPitch(NGN_MusicClip* music, float pitch = 1.0f);

Cambia la velocidad de reproducción y frecuencia de una música. (Usar 1.0f como valor nominal).

```
ngn->sound->MusicPitch(bgm, 0.75f);
```

float MusicGetPitch(NGN_MusicClip* music);

Devuelve el valor actual del PITCH de una música.

```
float pitch = ngn->sound->MusicGetPitch(bgm);
```

void MusicLoop(NGN_MusicClip* music, bool loop = true);

Cambia el estado del LOOP (repetición) de una música.

```
ngn->sound->MusicLoop(bgm, false);
```

bool MusicGetLoop(NGN_MusicClip* music);

Devuelve el estado del LOOP de una música.

```
bool loop = ngn->sound->MusicGetLoop(bgm);
```

bool MusicIsPlaying(NGN_MusicClip* music);

Devuelve el estado de reproducción de una música.

```
bool play = ngn->sound->MusicIsPlaying(bgm);
```

bool MusicIsAlive(NGN_MusicClip* music);

Devuelve si aún existe una música.

```
bool alive = ngn->sound->MusicIsAlive(bgm);
```

void PauseAll();

Pausa todos los SFX (efectos de sonido) y músicas que estén en reproducción en ese momento. Los sonidos y músicas reproducidos después de la ejecución de esta orden no se verán afectados.

```
ngn->sound->PauseAll();
```

void ResumeAll();

Continúa la reproducción de todos los SFX y músicas puesta en pausa por la instrucción PauseAll().

```
ngn->sound->ResumeAll();
```

void StopAll();

Detiene la reproducción de todos los SFX activos y cierra todos los streams de música abiertos.

```
ngn->sound->StopAll();
```

void Update();

Actualiza el estado de todos los SFX y músicas existentes en la cola de audio, eliminando los que ya no son necesarios automáticamente. Esta instrucción debe ejecutarse una vez por frame con el fin de mantener la cola de sonidos actualizada.

```
ngn->sound->Update();
```

void SetMixerLevel(uint8_t channel, int32_t level);

Cambia el nivel de volumen de unos de los canales del mezclador de sonido. Los canales disponibles son MIXER_MASTER_CH (canal maestro), MIXER_MUSIC_CH (música), MIXER_EFFECTS_CH (efectos de sonido), MIXER_AUX1_CH (canal auxiliar 1) y MIXER_AUX2_CH (canal auxiliar 2). El valor del volumen es un número entero entre 0 y 100.

```
ngn->sound->SetMixerLevel(MIXER_MASTER_CH, 75);
```

int32_t GetMixerLevel(uint8_t channel);

Devuelve el valor actual del volumen del canal especificado del mezclador de sonido. El valor devuelto es un número entero entre 0 y 100. En el caso de especificar un canal inexistente, devolverá 0.

```
int32_t vol = ngn->sound->GetMixerLevel(MIXER_MASTER_CH);
```