NGN_SPRITE.H

```
NGN_Sprite(
      NGN_SpriteData* sprite,
      // Objeto de la clase Sprite Data
      int32_t position_x = DEFAULT_VALUE,
      // Posición X inicial (oculto por defecto)
      int32_t position_y = DEFAULT_VALUE,
      // Posición Y inicial (oculto por defecto)
      uint32_t sprite_width = DEFAULT_VALUE,
      // Ancho del Sprite (por defecto, el de la textura)
      uint32_t sprite_height = DEFAULT_VALUE,
      // Altura del Sprite (por defecto, la de la textura)
      uint32_t box_width = DEFAULT_VALUE,
      // Ancho de la caja de colisiones
      uint32_t box_height = DEFAULT_VALUE
      // Alto de la caja de colisiones
      int32_t box_offset_x = NGN_DEFAULT_VALUE,
      // Offset horizontal de la caja de colisiones
      int32_t box_offset_y = NGN_DEFAULT_VALUE
      // Offset vertical de la de colisiones
);
```

Crea un nuevo Sprite, usando los datos previamente cargados.

```
NGN_Sprite* player = new NGN_Sprite(wizard_sprite, 300, 300, 256, 256, 160, 256);
```

void Position(float position_x, float position_y);
void Position(Vector2 pos);

Posiciona el sprite en la coordenada dada.

```
player->Position(1200, 900);
```

void Translate(float speed_x, float speed_y);
void Translate(Vector2 spd);

Mueve el sprite en la dirección y velocidades dadas.

```
player->Translate(5f, 0f);
```

void Size(float w, float h);

Cambia el tamaño del sprite.

```
player->Size(64, 48);
```

void Scale(float w, float h);
void Scale(float scale);

Escala el sprite, según el factor dado. Según la sobrecarga usada, escalara los ejes en conjunto o por separado. La escala por defecto es 1.0f.

```
player->Scale(1.5f);
player->Scale(2.0f, 0.75f);
```

void Rotate(double degrees);

Rota el sprite cada frame el número de unidades dado, en grados.

```
player->Rotate(1.2f);
```

void SetCenter(float x, float y);

Especifica, en coordenadas relativas y desde el centro real del sprite, donde se ubicará el centro de rotación del sprite.

```
player->SetCenter(-10, -5);
```

```
int32_t AddCollider(
    std::string name,
    // Nombre del colisionador
    float offset_x,
    // Offset X respecto al centro real del sprite
    float offset_y,
    // Offset Y respecto al centro real del sprite
    float width,
    // Ancho del colisionador
    float height
    // Altura del colisionador
);
```

Añade un colisionador (collider) adicional al sprite, con el nombre, tamaño y posición indicados. Pueden añadirse tantos colisionadores adicionales como sea necesario. Estos colisionadores serán detectados por la instrucción “HitBox” de la clase “NGN_Collisions” de manera automática. La rotación o escalado del sprite no afecta a estos colisionadores. Esta función devuelve 0 si el colisionador se añade con éxito.

```
player->AddCollider("bottom", -16.0f, 80.0f, 32.0f, 96.0f);
player->AddCollider("left", -64.0f, -80.0f, 64.0f, 32.0f);
```

int32_t GetColliderId(std::string name);

Devuelve la ID (posición) en el vector de colisionadores del colisionador con el nombre dado. Si no se encuentra, devuelve -1.

```
Int32_t id = player->GetColliderId(“left”);
```

int32_t ColliderEnabled(std::string name, bool status);

Habilita o deshabilita el colisionador con el nombre dado. Esta función devuelve 0 si tiene éxito el cambio de estado.

```
player->ColliderEnabled(“left”, false);
```

int32_t RemoveCollider(std::string name);

Elimina el colisionador con el nombre especificado. En caso de éxito, esta función devuelve 0.

```
player->RemoveCollider(“left”);
```

int32_t AddAnimation(
    std::string name,
    // Nombre de la animación
    uint32_t first_frame,
    // Primer fotograma de la animación
    uint32_t last_frame,
    // Ultimo fotograma de la animación
    uint32_t loop,
    // Fotograma "punto de loop”
    uint32_t frame_duration
    // Duracion en ciclos de cada fotograma
);

Añade una nueva animación al Sprite creado. Devuelve 1 en caso de error.

```
player->AddAnimation(“walk”, 1, 7, 1, 5);
```

int32_t SetAnimation(std::string name = "");

Selecciona una animación con el nombre dado para su reproducción. Devuelve 1 en caso de error.

```
player->SetAnimation(“walk”);
```

void PlayAnimation();

Reproduce la animación seleccionada actualmente.

```
player->PlayAnimation();
```

Vector2 position
Vector2 screen

Posición del Sprite (global o en pantalla).

float width
float height

Tamaño del Sprite.

float box.width
float box.height
float box.offset.x
float box.offset.y

bool box_enabled

Propiedades de la caja principal de colisión del Sprite. Puede habilitarse o deshabilitarse mediante el flag “box_anabled”.

int32_t frame

Fotograma del Sprite que debe mostrarse (El primer frame es el 0).

int32_t total_frames

Su valor indica el número total de fotogramas que contiene el sprite.

[string] current_animation.id
[uint32_t] current_animation.first_frame
[uint32_t] current_animation.last_frame
[uint32_t] current_animation.loop
[uint32_t] current_animation.frame_duration

Parámetros de la animación en curso.

bool animation_pause

Si el valor de esta propiedad es TRUE, la animación actual se pausa.

bool visible

Indica si el Sprite es o no visible.

int32_t alpha

Nivel de transparencia del Sprite, entre 0 y 255.

SDL_BlendMode blend_mode

Modo de mezcla de color del Sprite. Los modos disponibles son: *NGN_BLENDMODE_NONE*, *NGN_BLENDMODE_ALPHA*, *NGN_BLENDMODE_ADDITIVE* y *NGN_BLENDMODE_MODULATE*. El valor por defecto de esta propiedad es **NGN_BLENDMODE_ALPHA**.

bool on_screen

Flag para indicar si está o no en pantalla. El uso no es automático y deberá actualizar este flag manualmente.

double rotation

Rotación del Sprite, en grados.

bool flip_h
bool flip_v

Volteado vertical y horizontal del Sprite.