NGN_CAMERA.H

void CreateLayers(uint32_t layers);

Define el número de capas de las que dispondrá la cámara.

```
ngn->camera->CreateLayers(4);
```

void SizeOfLayer(uint32_t layer_number,
uint32_t width,
uint32_t height
);

Define el tamaño la capa de sprites (por defecto, todas las capas tienen el tamaño del fondo de esa capa).

```
ngn->camera->SizeOfLayer(3, 4000.0f, 2000.0f);
```

void Setup(uint32_t world_width,
uint32_t world_height,
NGN_Sprite* target_sprite = NULL
);

Inicializa la cámara. Debe especificarse el tamaño del mundo y opcionalmente, el sprite al que seguirá la cámara.

```
ngn->camera->Setup(4000, 1024, player->sprite);
```

int32_t PushBackground(uint32_t layer_number,
NGN_TiledBg* background
);

int32_t PushBackground(uint32_t layer_number,
NGN_Texture* texture
);

Añade un fondo a la capa y devuelve su índice en la lista. En caso de error, devuelve -1.

```
ngn->camera->PushBackground(2, bg_clouds);
```

int32_t PushVirtualBg(
uint32_t layer_number,
NGN_Texture* texture,
uint32_t bg_width,
uint32_t bg_height,
uint32_t loop_x,
uint32_t loop_y,
float auto_x = DEFAULT_VALUE,
float auto_y = DEFAULT_VALUE);

int32_t PushVirtualBg(
uint32_t layer_number,
NGN_TiledBg* background,
uint32_t bg_width,
uint32_t bg_height,
uint32_t loop_x,
uint32_t loop_y,
float auto_x = DEFAULT_VALUE,
float auto_y = DEFAULT_VALUE);

Añade un fondo con un tamaño “virtual” a la capa y devuelve su índice en la lista. En caso de error, devuelve -1. Debe especificarse el tamaño virtual del fondo, los puntos de loop en “X” e “Y” y opcionalmente los valores de auto-scroll de este fondo. El uso de este método sobrescribe el valor del tamaño de capa para los sprites.

```
ngn->camera->PushVirtualBg(0, bg0, 100000, 720, 2560, 0, 1, 0);
```

int32_t PushSprite(uint32_t layer_number, NGN_Sprite* sprite);

Añade un sprite a la capa y devuelve su índice en la lista. En caso de error, devuelve -1.

```
ngn->camera->PushSprite(3, player->sprite);
```

void LookAt(NGN_Sprite* target_sprite);

Indica a la cámara que debe seguir a un sprite concreto.

```
ngn->camera->LookAt(player->sprite);
```

void LookAt(uint32_t position_x, uint32_t position_y);

void LookAt(Vector2I32 pos);

Indica a la cámara que debe colocarse en una posición específica.

```
ngn->camera->LookAt(1000, 768);
```

void Update();

Actualiza la vista de la cámara. Esta función debe llamarse una vez por frame.

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

int32_t RemoveBackground(NGN_TiledBg* background);

int32_t RemoveBackground(NGN_Texture* texture);

Busca y elimina un fondo de la cámara. En caso de no encontrar ese fondo en la lista, devuelve -1.

```
ngn->camera-> RemoveBackground(bg_clouds);
```

int32_t RemoveSprite(NGN_Sprite* sprite);

Busca y elimina un sprite de la cámara. En caso de no encontrar ese sprite en la lista, devuelve -1.

```
ngn->camera-> RemoveSprite(player->sprite);
```

int32_t ChangeLayer(NGN_Sprite* sprite, uint32_t layer_number);

Cambia de capa un sprite. En caso de no encontrar ese sprite en la lista, devuelve -1.

```
ngn->camera-> ChangeLayer(player->sprite, 2);
```

void Reset();

Reinicia la camara, eliminando todas las capas y las referencias contenidas.

```
ngn->camera-> Reset();
```

Size2I world

Almacena el tamaño actual del mundo.

Vector2I position

Almacena la posición actual de la cámara en el mundo.

bool animation_pause

Pausa la animación de todos los sprites si su valor es TRUE.