NGN_GRAPHICS.H

bool Init(

```
    std::string window_name,            // Nombre en la ventana
    uint32_t native_width,              // Resolución Nativa del juego
    uint32_t native_height,
    int8_t scr_mode = NGN_SCR_WINDOW    // Modo de pantalla
    bool bilinear_filter = false,       // Filtro bilinear activado?
    bool sync = true                    // VSYNC activo?
```

);

Inicia el modo gráfico con los parámetros especificados. Devuelve FALSE en caso de error.

```
ngn->graphics->Init("Ngine Demo", 1280, 720);
```

```
Los modos de pantalla disponibles son:
NGN_SCR_WINDOW          1       Modo ventana x1
NGN_SCR_WINDOW_X2       2       Modo ventana x2
NGN_SCR_WINDOW_X3       3       Modo ventana x3
NGN_SCR_WINDOW_X4       4       Modo ventana x4
NGN_SCR_WINDOW_FULL     0       Modo ventana a pantalla completa
NGN_SCR_FULLSCREEN      -1      Modo pantalla completa
```

void SetMode(int8_t mode);

Cambia el modo gráfico al indicado. Los cambios de modo gráfico pueden provocar que las capas de texto o canvas deban de redibujarse por completo. Usar la variable “force_redraw” de la clase para detectar si debe realizarse.

```
ngn->graphics->SetMode(NGN_SCR_WINDOW_X2);
```

void SetFiltering(bool enabled);

Establece el estado del filtrado bilinear de la escena.

```
ngn->graphics->SetFiltering(true);
```

void SetVerticalSync(bool enabled);

Establece el estado de la sincronización vertical de la ventana.

```
ngn->graphics->SetVerticalSync(true);
```

void Update();

Intenta actualizar la escena a 60fps. Llamar una vez por frame.

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

```
void SetViewportClip(
                     int32_t x,    // Posición X
                     int32_t y,    // Posición Y
                     int32_t w,    // Ancho
                     int32_t h     // Alto
                     );
```

Define el área visible de la pantalla.

```
ngn->graphics->SetViewportClip(100, 100, 250, 250);
```

void ShowMouse(bool visible);

Muestra u oculta el cursor del ratón.

```
ngn->graphics->ShowMouse(false);
```

```
void OpenViewport(
                  uint8_t id,        // ID del VIEWPORT (0-7)
                  int32_t pos_x,     // Posición del viewport
                  int32_t pos_y,
                  uint32_t width,    // Ancho del viewport
                  uint32_t height,   // Alto del viewport
                  // Resolución del render en el viewport
                  uint32_t h_res = NGN_DEFAULT_VALUE,
                  uint32_t v_res = NGN_DEFAULT_VALUE,
                  uint32_t local_filter = false
);
```

Abre uno de los 8 viewports disponibles con los parámetros facilitados.
Se puede cambiar el estado del filtrado local con el parámetro “local_filter”. Esto no afecta al estado del filtrado general de la escena.

```
ngn->graphics->OpenViewport(3, 480, 240, 320, 240, 160, 120);
ngn->graphics->OpenViewport(3, 480, 240, 320, 240, 160, 120, true);
```

void CloseViewport(uint8_t id);

Cierra el viewport con el ID facilitado.

```
ngn->graphics->CloseViewport(3);
```

void SelectViewport(uint8_t id);

Selecciona el viewport con el ID facilitado. A partir de ese momento todos los comandos de ngn->render se enviarán a ese viewport. Todos los elementos enviados al viewport no se mostraran en pantalla hasta que se ejecute el comando ngn->render->Viewports(); el cual renderiza el contenido de los viewports en ese momento.

```
ngn->graphics->SelectViewport(3);
```

void ViewportPosition(uint8_t id, int32_t x, int32_t y);
void ViewportPosition(uint8_t id, Vector2I32 position);

Posiciona en las coordenadas facilitadas el viewport solicitado. El punto de referencia se sitúa en la coordenada superior izquierda.

```
ngn->graphics->ViewportPosition(3, 100, 200);
```

void ViewportLocalFilter(uint8_t id, bool status);

Cambia el estado del filtrado local del viewport seleccionado.

```
ngn->graphics->ViewportLocalFilter(3, true);
```

void DefaultViewport();

Devuelve el destino del render a la pantalla principal.

```
ngn->graphics->DefaultViewport();
```

NGN_Sprite* CloneSprite(NGN_Sprite* sprite);

Clona un sprite con las propiedades actuales. El Sprite resultante es una instancia totalmente nueva.

```
NGN_Sprite* spr_clone = ngn->graphics->CloneSprite(spr);
```

Size2I32 GetDesktopResolution();

Devuelve la resolución del escritorio en el momento de iniciar el programa.

```
Size2I32 desktop_res = ngn->graphics->GetDesktopResolution();
```

uint32_t GetFps();

Devuelve el número de fotogramas renderizados en el transcurso del último segundo.

```
uint32_t fps = ngn->graphics->GetFps();
```

```
void ScreenShot(
    std::string path,                   // Ruta de destino
    NGN_TextureData* overlay = NULL,    // Textura opcional
    uint8_t alpha = 0xFF                // Nivel de opacidad
);
```

Guarda una captura del contenido del renderer del frame actual en formato PNG en el archivo y rutas especificados. De manera adicional, puede especificarse una textura para superponerla en la captura y pudiendo además especificar su nivel de opacidad.

```
ngn->graphics->ScreenShot("captures/myscreen1.png");
ngn->graphics->ScreenShot("captures/myscreen2.png", overlay_tex, 0XD0);
```

bool vsync

Indica si el sincronismo vertical está activo.

bool filtering

Indica si el filtro bilinear está activo.

int32_t native_w
int32_t native_h

Almacena la resolución nativa del juego.

std::string window_caption

Guarda el texto del título de la ventana.

SDL_Rect cliparea

Almacena los valores del área de recorte.

bool force_redraw

Indica si debido a un cambio de modo de pantalla, debe redibujarse la escena.