

# NGN\_CANVAS.H

## *(Funciones de La capa de dibujado)*

```
NGN_Canvas(  
    int32_t position_x = 0,  
        // Posicion X (0 por defecto)  
    int32_t position_y = 0,  
        // Posicion Y (0 por defecto)  
    uint32_t _width = DEFAULT_VALUE,  
        // Ancho de la capa (Toda la pantalla por defecto)  
    uint32_t _height = DEFAULT_VALUE,  
        // Alto de la capa (Toda la pantalla por defecto)  
    bool _filtering = false  
        // Filtrado bilinear del contenido?  
);
```

Crea un nuevo canvas, usando los parámetros especificados.

```
NGN_Canvas* canvas = new NGN_Canvas(100, 50, 200, 64);
```

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

Posiciona el canvas en la coordenada dada.

```
canvas->Position(10, 20);
```

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

Mueve el canvas en la dirección y velocidades dadas.

```
canvas->Translate(0.0f, -2.0f);
```

```
void Size(float w, float h);
```

Cambia el tamaño del canvas.

```
canvas->Size(640, 480);
```

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

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

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

### **void Rotate(double degrees);**

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

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

### **void SetCenter(float x, float y);**

Específica, en coordenadas relativas y desde el centro real canvas, donde se ubicará el centro de rotación del mismo.

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

### ***Vector2 position***

Posición del canvas en pantalla.

### ***float width***

### ***float height***

Tamaño del canvas.

### ***bool visible***

Indica si el canvas es o no visible.

### ***int32\_t alpha***

Nivel de transparencia del canvas, entre 0 y 255.

### ***SDL\_BlendMode blend\_mode***

Modo de mezcla de color del canvas. 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 filtering***

Activa o desactiva el filtrado bilineal del contenido del canvas.

### ***double rotation***

Rotación del canvas, en grados.

***bool flip\_h***

***bool flip\_v***

Volteado vertical y horizontal del canvas.

Nota:

Los cambios de tamaño o escala no afectan al tamaño original del contenedor, solo se cambia el tamaño del contenido al representarse en la pantalla.

## ***(Funciones de dibujado)***

**void Cls(uint32\_t color = 0x00000000);**

Borra el contenido del canvas y si se especifica, lo rellena con el color dado.

```
textbox->Cls(0x0080FFFF);      // RRGGBBAA
```

**void Point(int32\_t x, int32\_t y, uint32\_t color);**

Dibuja un punto de 1x1 pixels del color especificado en las coordenadas del canvas dadas.

```
canvas->Point(100, 50, 0xFF00FF);
```

```
void Line(  
    int32_t x1, int32_t y1,      // Punto A  
    int32_t x2, int32_t y2,      // Punto B  
    uint32_t color              // Color (RGBA)  
);
```

Dibuja una línea entre dos puntos con el color especificado.

```
canvas->Line(10, 10, 200, 200, 0xFF0000FF);
```

```

void Box(
    int32_t x1, int32_t y1,          // Vértice superior izquierdo
    int32_t x2, int32_t y2,          // Vértice inferior derecho
    uint32_t color,                  // Color (RGBA)
    bool paint = false               // Relleno?
);

```

Dibuja una caja entre los vértices especificados con el color dado. Puede dibujarse con o sin relleno.

```

canvas->Box(10, 10, 200, 200, 0xFF00FFFF, true);
canvas->Box(10, 10, 200, 200, 0xFFFFFFFF);

```

```

void Circle(
    int32_t cx, int32_t cy,          // Coordenadas del centro
    int32_t r,                       // Radio horizontal
    uint32_t color,                  // Color (RGBA)
    int32_t ry = DEFAULT_VALUE,      // Radio vertical
    bool paint = false               // Ángulo inicial (RAD)
);

```

Dibuja un círculo con los parámetros especificados. Si no se especifica el radio vertical, se usará el horizontal en su lugar. El parámetro paint establece si el círculo es o no con relleno.

```

canvas->Circle(320, 240, 32, 0xFFFFFFFF);
canvas->Circle(320, 240, 32, 0x804080FF, 64, true);

```

```

void Arc(
    int32_t cx, int32_t cy,          // Coordenadas del centro
    int32_t r,                       // Radio horizontal
    double start_angle,              // Ángulo inicial (rad)
    double end_angle,                // Ángulo final (rad)
    uint32_t color,                  // Color (RGBA)
    int32_t ry = DEFAULT_VALUE,      // Radio vertical
    uint8_t close = 0                // Cerrar el arco?
);

```

Dibuja un arco con los parámetros especificados. Si no se especifica el radio vertical, se usará el horizontal en su lugar. Los ángulos deben de especificarse en RADIANTES. El parámetro close permite cerrar el arco por sus extremos (0 = no lo cierra, 1 = cierra los dos extremos, 2 = cierra los extremos con el centro).

```

canvas->Arc(250, 360, 200, 0.0f, (PI * 2.0f), 0xFFFFFFFF, 200);
canvas->Arc(640, 360, 100, 0.3f, 4.0f, 0xFFFFFFFF, 100, 1);
canvas->Arc(640, 580, 100, 0.8f, 5.0f, 0xFFFFFFFF, 100, 2);

```

**uint32\_t GetPixelColor(int32\_t x, int32\_t y);**

Devuelve el color en formato RGBA8888 (RRGGBBAA) del pixel en las coordenadas proporcionadas. Si las coordenadas se encuentran fuera de los límites del canvas, se devuelve 0x00000000.

```
uint32_t color = canvas->GetPixelColor(345, 123);
```

**Rgba GetPixelRgba(int32\_t x, int32\_t y);**

Devuelve el color en formato RGBA (color.r, color.g, color.b, color.a) del pixel en las coordenadas proporcionadas. Si las coordenadas se encuentran fuera de los límites del canvas, se devuelve 0 en los 4 componentes.

```
Rgba color = canvas->GetPixelRgba(345, 123);
```