NASH_API_REFERENCE.md

NASH v1.01 API Reference

by Nicholas Musurca (nick.musurca@gmail.com)

Initialization

`NASH:New(width, height)`

Creates a new canvas at `width x height` dimensions. The canvas is transparent by default, with all pixels set to `RGBA(0,0,0,0)`. If you can, try to reuse the canvases you make.

```
-- example: create a 640x480 canvas
local my_canvas = NASH:New(640, 480)

print("width: "..my_canvas.width)
print("height: "..my_canvas.height)
```

`RGB(red, green, blue)`

Creates an RGBA color with the elements (red, green, blue, 255). Note that valid ranges for color elements are [0-255].

```
-- example: the color yellow
local color_yellow = RGB(255, 255, 0)
```

`RGBA(red, green, blue, alpha)`

Creates an RGBA color with the elements (red, green, blue, alpha). Note that valid ranges for color elements are [0-255].

```
-- example: the semi-transparent color yellow
local color_yellow = RGBA(255, 255, 0, 127)

-- note that a color is just a table of 4 numbers. The following line is equivalent:
color_yellow = {255, 255, 0, 127}
```

Drawing

`NASH:BlendMode(mode)`

Changes the current blend mode to `mode`. The blend modes are:

- `NASH.BLEND_NORMAL`
- `NASH.BLEND_ADD`
- `NASH.BLEND_SUBTRACT`
- `NASH.BLEND_MULTIPLY`
- `NASH.BLEND_SCREEN`
- `NASH.BLEND_ALPHA`

```
-- example: set the blend mode to NORMAL, the default mode
```

```
my_canvas:BlendMode(NASH.BLEND_NORMAL)
-- this replaces whatever is at (64, 32) with a red point
my_canvas:Point(64, 32, RGB(255, 0, 0) )

-- now we set the blend mode to ALPHA
my_canvas:BlendMode(NASH.BLEND_ALPHA)
-- this blends the point at (64, 32) with a new color, using the alpha value
my_canvas:Point(64, 32, RGBA(0, 127, 255, 127) )
```

NASH:Box(x0, y0, x1, y1, color)

Draws a box from top-left (`x0` , `y0`) to bottom-right (`x1` , `y1`) in the specified `color` .

```
-- example: draw a yellow box from (0, 0) to (100, 100)
my_canvas:Box(0, 0, 100, 100, RGB(255, 255, 0) )
```

NASH:BoxFill(x0, y0, x1, y1, color)

Draws a filled box from top-left (`x0` , `y0`) to bottom-right (`x1` , `y1`) in the specified `color` .

```
-- example: draw a filled yellow box from (0, 0) to (100, 100)
my_canvas:BoxFill(0, 0, 100, 100, RGB(255, 255, 0) )
```

NASH:Circle(x0, y0, radius, color)

Draws a circle at (`x0` , `y0`) with `radius` in the specified `color` .

```
-- example: draw a yellow circle at (50, 50) with radius 25
my_canvas:Circle(50, 50, 25, RGB(255, 255, 0) )
```

NASH:CircleFill(x0, y0, radius, color)

Draws a filled circle at (`x0` , `y0`) with `radius` in the specified `color` .

```
-- example: draw a filled yellow circle at (50, 50) with radius 25
my_canvas:CircleFill(50, 50, 25, RGB(255, 255, 0) )
```

NASH:Clear(color)

Fills the entire canvas with the specified `color` .

```
-- example: fill the canvas with opaque white
my_canvas:Clear( RGB(255, 255, 255) )
```

NASH:Invert()

Inverts all of the elements of the canvas colors.

NASH:InvertRGB()

Inverts the RGB elements of the canvas colors, but leaves the alpha element untouched.

NASH:Line(x0, y0, x1, y1, color)

Draws a line from (`x0` , `y0`) to (`x1` , `y1`) in the specified `color` .

```
-- example: draw a yellow line from (0, 0) to (100, 100)
my_canvas:Line(0, 0, 100, 100, RGB(255, 255, 0) )
```

NASH:OvalFill(x0, y0, radius_x, radius_y, color)

Draws a filled oval at (`x0` , `y0`) with x-dimension `radius_x` and y-dimension `radius_y` in the specified `color` .

```
-- example: draw a filled yellow oval at (50, 50) with x-radius 50 and y-radius 25
my_canvas:OvalFill(50, 50, 50, 25, RGB(255, 255, 0) )
```

NASH:Point(x, y, color)

Draws a single point at (`x` , `y`) in the specified `color` .

```
-- example: draw a red point at (64, 32)
my_canvas:Point(64, 32, RGB(255, 0, 0) )
```

NASH:SoftCircleFill(x0, y0, radius, color)

Draws a soft filled circle with Gaussian falloff at (`x0` , `y0`) with `radius` in the specified `color` .

```
-- example: draw a soft filled yellow circle at (50, 50) with radius 25
my_canvas:SoftCircleFill(50, 50, 25, RGB(255, 255, 0) )
```

NASH:SoftOvalFill(x0, y0, radius_x, radius_y, color)

Draws a soft filled oval with Gaussian falloff at (`x0` , `y0`) with x-dimension `radius_x` and y-dimension `radius_y` in the specified `color` .

```
-- example: draw a soft filled yellow oval at (50, 50) with x-radius 50 and y-radius 25
my_canvas:SoftOvalFill(50, 50, 50, 25, RGB(255, 255, 0) )
```

NASH:Triangle(x0, y0, x1, y1, x2, y2 color)

Draws a triangle with the vertices (`x0` , `y0`), (`x1` , `y1`), and (`x2` , `y2`) in the specified `color` .

```
-- example: draw a yellow triangle at (0, 0), (50, 100), (100, 0)
my_canvas:Triangle(0, 0, 50, 100, 100, 0 RGB(255, 255, 0) )
```

NASH:TriangleFill(x0, y0, x1, y1, x2, y2 color)

Draws a filled triangle with the vertices (`x0` , `y0`), (`x1` , `y1`), and (`x2` , `y2`) in the specified `color` .

```
-- example: draw a filled yellow triangle at (0, 0), (50, 100), (100, 0)
my_canvas:TriangleFill(0, 0, 50, 100, 100, 0 RGB(255, 255, 0) )
```

Text

NASH:Print(str, x, y, color, [align])

Prints `str` at (`x` , `y`) in the specified `color` . The optional `align` argument specifies the text alignment, and may be one of the following:

- `NASH.ALIGN_LEFT` (default value)
- `NASH.ALIGN_CENTER`
- `NASH.ALIGN_RIGHT`

```
-- example: print the line 'Hello, CMO!' centered at (100, 50) in white
my_canvas:Print("Hello, CMO!", 30, 30, RGB(255, 255, 255), NASH.ALIGN_CENTER)
```

NASH:TextWidth(str)

Returns the width in pixels of printing `str` in the **NASH** default font.

NASH:TextHeight()

Returns the height in pixels of the **NASH** default font.

Rendering

NASH:Render([scale])

Renders your canvas to an HTML IMG tag for use in a Special Message. If the optional argument `scale` is specified, the canvas will be stretched or squeezed by the scale factor. (If not specified, `scale` is set to `1` .) Note that NASH caches the results of each render. Once you render a canvas once, it will be much faster to render it a second time if you haven't made any changes.

```
-- example: render our canvas to a Special Message
local msg = "<br/>This is a test of the NASH rendering system."
ScenEdit_SpecialMessage("playerside", my_canvas:Render()..msg)
```