Palette

COLLABORATORS			
	TITLE : Palette		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		October 9, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Palette

1.1 Palette V1.10

```
                  Palette V1.10 General Information:

* Blitz Basic II library number         : #139
* Library size when linked to executable: 1192 bytes
* Number of commands                    : 17
* Ressources automatically freed at end : Yes

NInitPalette() must be put before any other Palette functions
or you will enjoy BIG crashs.

NOTE: All the RGB value are between 0 and 255 colours to support
AGA and GFX cards display, so ROM 3.0+ is needed. Note it will
work on any Amiga with Rom 3.0+ (Amiga 500 - Old Chip Set too).

  Commands summary:


              NASyncFade
              Statement

              NASyncFadeStatus
              Function (Byte)

              NBlue
              Function (Word)

              NCreatePalette
              Command (Long)

              NDisplayPalette
              Statement

              NFreePalette
              Statement

              NFade
              Statement
```

```
NFadeOut
Statement

NGetPicturePalette
Function (Long)

NGetScreenPalette
Function (Long)

NGreen
Function (Word)

NInitPalette
Function (Boolean)

NNbColour
Function (Long)

NPalRrgb
Statement

NRed
Function (Word)

NRgb
Statement

NUsePalette
Statement
```

1.2 nasyncfade

```
  SYNTAX
NASyncFade(#Palette1, #Palette2, Step, NbLoop, ScreenID)

  STATEMENT
Same as NFade() routine, but don't halt the program. The fade
is executed in the background.

You can use NAsyncStatus() to know if the background fade is
finished or not.
```

1.3 nasyncfadestatus

```
  SYNTAX
Result.b = NASyncFadeStatus

  STATEMENT
Return '-1' if the Fade is always running or '0' if
the fade has finished.
```

```
Example:

Repeat                  ; Typical loop to wait the end
  NVWait                ; of the background fade.
Until NASyncStatus = 0  ;
```

1.4 nfade

```
  SYNTAX
NFade(#Palette1, #Palette2, Step, NbLoop, ScreenID)

  STATEMENT
Do a nice fade between the two palettes. The palettes must have the
same number of colours or it could crash. Step allow to control
the speed of the Fade (1 is the fastest, >1 numbers will slowdown
the fade speed). NbLoop allow to tell how many loop the Fade must
do before exit. By default the Fade execute ALWAYS 255 loop. So
you can adjust it manually (ie: with a Step of 2, you should
use a NbLoop of 255/2 +-= 127)

This function is optimized for speed, and give very good result
on any Amigas (020 recommended tough), with high coloured
screens (8 depths..).
```

1.5 nfreepalette

```
  SYNTAX
NFreePalette(#Palette)

  STATEMENT
Free the memory allocated by the given #Palette.
```

1.6 ninitpalette

```
  SYNTAX
result.l = NInitPalette(#NumPaletteMax)

  FUNCTION
Init all the Palette environnement for later use. You must put this
functions on top of your source code if you want to use the NPalette
commands.

#NumPaletteMax : Maximum number of Palette to handle.
```

1.7 nred

```
  SYNTAX
Red.w = NRed(ColourIndex)

  FUNCTION
Return the Red value of the color found in the current palette.
Returned value is always between 0 and 255.
```

1.8 ngreen

```
  SYNTAX
Green.w = NGreen(ColourIndex)

  FUNCTION
Return the Green value of the color found in the current palette.
Returned value is always between 0 and 255.
```

1.9 nblue

```
  SYNTAX
Blue.w = NBlue(ColourIndex)

  FUNCTION
Return the Blue value of the color found in the current palette.
Returned value is always between 0 and 255.
```

1.10 ncreatepalette

```
  SYNTAX
res.l = NCreatePalette(#Palette, NbColour)

  COMMAND
Try to create a new palette with given argument. The size in memory took
by a palette object can be calculated like that:
  Size (in bytes) = NbColours * 12 + 12

The created palette is ready to use and filled with colour 0.
```

1.11 nrgb

```
  SYNTAX
NRgb(ScreenID, ColourIndex, R, G, B)

  STATEMENT
Change directly the RGB value of a colour in the given Screen.
```

1.12 nnbcolour

```
  SYNTAX
Result.l = NNbColour

  STATEMENT
Return the number of colour of currently used palette.
```

1.13 npalrgb

```
  SYNTAX
NPalRgb(ColourIndex, R, G, B)

  STATEMENT
Change the RGB value of a colour in the current palette.
```

1.14 ngetscreenpalette

```
  SYNTAX
res.l = NGetScreenPalette(#Palette, ScreenID)

  COMMAND
Try to create a new palette and fill it with screen colour information.
If res = 0 the palette has failed to be created.
```

1.15 ngetpicturepalette

```
  SYNTAX
res.l = NGetPicturePalette(#Palette, PictureID)

  COMMAND
Try to create a new palette and fill it with picture colour information.
If res = 0 the palette has failed to be created.

PictureID is a pointer to an IFF/ILBM file in memory.
```

1.16 ndisplaypalette

```
  SYNTAX
DisplayPalette(#Palette, ScreenID)

  STATEMENT
Display the given #Palette on the screen.
```

1.17 nusepalette

```
  SYNTAX
NUsePalette(#Palette)

  STATEMENT
Change the used Palette to given #Palette.
```

1.18 nfadeout

```
  SYNTAX
NFadeOut(#Palette, Step, NbLoop, ScreenID)

  STATEMENT
It will display a very nice fade out from the given palette.
The palette WILL be modified (at the end of the fading, the
palette will be all black). The fadeout speed can be controled
with the 'Step' parameter.

  If Step = 1 then the fading will be smooth and take 1 vwait
  before fadeout the next frame
  If Step = 2 fading will be 2 time faster than Step 1 ...

NbLoop is used to fade partially a screen:

  If NbLoop = 255, the whole screen will be black at end, because
  with 255 loops, the fadeout is complete

  If NbLoop = 50, after 50 loop the FadeOut will stop. Test it
  to understand better :)

This routine is optimized for speed and give excellent results
even on small Amiga. And more, it's fully system friendly (no
hardware bang...) so works on GFX card too ! It's better to use
this routine than the NFade() to do standard Fade Out..
```