

guigfx

COLLABORATORS

| | | | |
|------------|-------------------|------------------|-----------|
| | TITLE : guigfx | | |
| ACTION | NAME | DATE | SIGNATURE |
| WRITTEN BY | | October 22, 2024 | |

REVISION HISTORY

| | | | |
|--------|------|-------------|------|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

Contents

| | | |
|----------|--|----------|
| 1 | guigfx | 1 |
| 1.1 | guigfx.doc | 1 |
| 1.2 | guigfx.library/AddPaletteA | 1 |
| 1.3 | guigfx.library/AddPictureA | 2 |
| 1.4 | guigfx.library/AddPixelArrayA | 4 |
| 1.5 | guigfx.library/ClonePictureA | 5 |
| 1.6 | guigfx.library/CreateDirectDrawHandleA | 6 |
| 1.7 | guigfx.library/CreatePenShareMapA | 7 |
| 1.8 | guigfx.library/CreatePictureBitMapA | 8 |
| 1.9 | guigfx.library/CreatePictureMaskA | 10 |
| 1.10 | guigfx.library/DeleteDirectDrawHandle | 11 |
| 1.11 | guigfx.library/DeletePenShareMap | 12 |
| 1.12 | guigfx.library/DeletePicture | 12 |
| 1.13 | guigfx.library/DirectDrawTrueColorA | 12 |
| 1.14 | guigfx.library/DoPictureMethodA | 13 |
| 1.15 | guigfx.library/DrawPictureA | 25 |
| 1.16 | guigfx.library/GetPictureAttrsA | 27 |
| 1.17 | guigfx.library/IsPictureA | 28 |
| 1.18 | guigfx.library/LoadPictureA | 29 |
| 1.19 | guigfx.library/LockPictureA | 30 |
| 1.20 | guigfx.library/MakePictureA | 31 |
| 1.21 | guigfx.library/ObtainDrawHandleA | 34 |
| 1.22 | guigfx.library/ReadPictureA | 36 |
| 1.23 | guigfx.library/ReleaseDrawHandle | 37 |
| 1.24 | guigfx.library/RemColorHandle | 37 |
| 1.25 | guigfx.library/UnLockPicture | 38 |

Chapter 1

guigfx

1.1 guigfx.doc

```
AddPaletteA ()
AddPictureA ()
AddPixelArrayA ()
ClonePictureA ()
CreateDirectDrawHandleA ()
CreatePenShareMapA ()
CreatePictureBitMapA ()
CreatePictureMaskA ()
DeleteDirectDrawHandle ()
DeletePenShareMap ()
DeletePicture ()
DirectDrawTrueColorA ()
DoPictureMethodA ()
DrawPictureA ()
GetPictureAttrsA ()
IsPictureA ()
LoadPictureA ()
LockPictureA ()
MakePictureA ()
ObtainDrawHandleA ()
ReadPictureA ()
ReleaseDrawHandle ()
RemColorHandle ()
UnLockPicture ()
```

1.2 guigfx.library/AddPaletteA

NAME

AddPaletteA - add a palette's colors to a pen-sharemap.
AddPalette - varargs stub for AddPaletteA.

SYNOPSIS

```
colorhandle = AddPaletteA(psm,palette,taglist)
d0              a0  a1      a2
```

```
APTR AddPaletteA(APTR,APTR,struct TagItem *)
```

```
APTR AddPalette(APTR,APTR,tag,...,TAG_DONE)
```

FUNCTION

This function adds a palette's colors to a pen-sharemap.

INPUTS

| | |
|---------|-----------------------------------|
| psm | - pointer to a pen-sharemap |
| palette | - pointer to a color table |
| tags | - pointer to an array of TagItems |

TAGS

GGFX_PaletteFormat (ULONG) - format of the palette. Currently defined are:

PALFMT_RGB8

ULONG 0x00rrggbb

PALFMT_RGB32

ULONG red,green,blue. This is the LoadRGB32() format without trailing longword.

Default: PALFMT_RGB8

GGFX_NumColors (ULONG) - number of colors in the color table. Currently, this argument is mandatory. Default: 0

GGFX_Weight (ULONG) - weight factor. Valid range: 1...255. With this factor, you can specify a significance for this color instance. The higher this value, the higher the palette's influence on the pen-sharemap. Default: 1

RESULTS

colorhandle - identifier for a particular dependency between color information and pen-sharemap. there is no need for you to store a colorhandle, unless you want to manually remove it from the pen-sharemap via RemColorHandle(). NULL if something went wrong.

NOTES

An example is provided with the documentation for AddPictureA().

SEE ALSO

RemColorHandle(), AddPictureA(), AddPixelArrayA(), CreatePenShareMapA(), DeletePenShareMap(), ObtainDrawHandleA()

1.3 guigfx.library/AddPictureA

NAME

AddPictureA - add a picture's color information to a pen-sharemap.
AddPicture - varargs stub for AddPictureA.

SYNOPSIS

```
colorhandle = AddPictureA(psm,picture,taglist)
d0          a0  a1      a2
```

```
APTR AddPictureA(APTR,APTR,struct TagItem *)
```

```
APTR AddPicture(APTR,APTR,tag,...,TAG_DONE)
```

FUNCTION

This function adds a picture's color information to a pen-sharemap.

INPUTS

| | |
|---------|-----------------------------------|
| psm | - pointer to a pen-sharemap |
| picture | - pointer to a picture |
| tags | - pointer to an array of TagItems |

TAGS

GGFX_Weight (ULONG) - weight factor. Valid range: 1...255.
With this factor, you can specify a significance for this color instance. The higher this value, the higher the picture's influence on the pen-sharemap.
Default: 1

RESULTS

colorhandle - identifier for a particular dependency between color information and pen-sharemap. there is no need for you to store a colorhandle, unless you want to manually remove it from the pen-sharemap via RemColorHandle(). NULL if something went wrong.

EXAMPLE

Assume there were three different pictures to be drawn.

- a) a noisy background
- b) a logo of your company
- c) navigation icons

you might want to differenciate the significances for these pictures as follows:

```
AddPicture(psm, backpic, GGFX_Weight, 2, TAG_DONE);
AddPicture(psm, logopic, GGFX_Weight, 3, TAG_DONE);
AddPicture(psm, navpic, GGFX_Weight, 5, TAG_DONE);
```

the backpic's influence on the allocated pens would be 20%, the logo contributed with 30%, and the navigation buttons would be taken into account with 50% then.

NOTES

SEE ALSO

```
RemColorHandle(), AddPaletteA(), AddPixelArrayA(),
CreatePenShareMapA(), DeletePenShareMap(), ObtainDrawHandleA()
```

1.4 guigfx.library/AddPixelFormatA

NAME

AddPixelFormatA - add a pixel array's color information to a pen-sharemap.
 AddPixelFormat - varargs stub for AddPixelFormatA.

SYNOPSIS

```
colorhandle = AddPixelFormatA(psm,array,width,height,taglist)
d0          a0  a1      d0      d1      a2
```

```
APTR AddPixelFormatA(APTR,APTR,UWORD,UWORD,struct TagItem *)
```

```
APTR AddPixelFormat(APTR,APTR,UWORD,UWORD,tag,...,TAG_DONE)
```

FUNCTION

This function adds a pixel array's color information to a pen-sharemap.

INPUTS

```
psm          - pointer to a pen-sharemap
pixelarray   - pointer to a pixel array
width        - pixel array's width [pixels]
height       - pixel array's height [rows]
tags         - pointer to an array of TagItems
```

TAGS

GGFX_PixelFormat (ULONG) - pixel format. Currently defined are

```
PIXFMT_CHUNKY_CLUT
    chunky bytes, directly acting as indices
    to a color-lookup-table. You must specify the
    GGFX_Palette and GGFX_NumColors tags as well.
```

```
PIXFMT_ORGB_32
    truecolor pixels (ULONG 0x00rrggbb).
```

Default: PIXFMT_CHUNKY_CLUT

GGFX_Palette (APTR) - pointer to a color table. Mandatory for PIXFMT_CHUNKY_CLUT (see above).

Default: none

GGFX_NumColors (ULONG) - number of colors in the color table. Mandatory for PIXFMT_CHUNKY_CLUT (see above).

Default: none

GGFX_PaletteFormat (ULONG) - format of the palette. Currently defined are:

```
PALFMT_RGB8
    ULONG 0x00rrggbb
```

```
PALFMT_RGB32
    ULONG red,green,blue. This is the LoadRGB32()
    format without trailing longword.
```

Default: PALFMT_RGB8

GGFX_Weight (ULONG) - weight factor. Valid range: 1...255.
 With this factor, you can specify a significance
 for this color instance. The higher this value, the
 higher the pixel array's influence on the pen-sharemap.
 Default: 1

RESULTS

colorhandle - identifier for a particular dependency between
 color information and pen-sharemap. there is
 no need for you to store a colorhandle, unless
 you want to manually remove it from the pen-sharemap
 via RemColorHandle(). NULL if something went wrong.

NOTES

An example is provided with the documentation for AddPictureA().

SEE ALSO

RemColorHandle(), AddPaletteA(), AddPictureA(),
 CreatePenShareMapA(), DeletePenShareMap(), ObtainDrawHandleA()

1.5 guigfx.library/ClonePictureA

NAME

ClonePictureA - create a duplicate from a picture.
 ClonePicture - varargs stub for ClonePictureA.

SYNOPSIS

```
newpicture = ClonePictureA (picture, taglist)
d0          a0          a1
```

```
APTR ClonePictureA (APTR, struct TagItem *)
```

```
APTR ClonePicture (APTR, tag, ..., TAG_DONE)
```

FUNCTION

This function creates a duplicate from a picture. Memory will
 be allocated, and the picture will be copied including all its
 attributes. Optionally, the picture is cloned only in part.

INPUTS

picture - pointer to a picture
 tags - pointer to an array of TagItems

TAGS

GGFX_SourceX (ULONG)
 left edge inside the picture where to fetch the pixels
 from [pixels]. Default: 0.

GGFX_SourceY (ULONG)
 top edge inside the picture where to fetch the pixels
 from [rows]. Default: 0.

GGFX_SourceWidth (ULONG)
width of an area inside the picture [pixels].
Default: The picture's width.

GGFX_SourceHeight (ULONG)
height of an area inside the picture [rows].
Default: The picture's height.

GGFX_DestWidth (ULONG)
width of the new picture [pixels].
Default: The picture's width.

GGFX_DestHeight (ULONG)
height of the new picture [rows].
Default: The picture's height.

RESULTS

newpicture - a vanilla copy of the specified picture (or a part of it), or NULL if there was not enough memory available.

SEE ALSO

MakePictureA(), DeletePicture()

1.6 guigfx.library/CreateDirectDrawHandleA

NAME

CreateDirectDrawHandleA - derive a handle for 'direct' drawing (v9)
CreateDirectDrawHandle - varargs stub for CreateDirectDrawHandleA

SYNOPSIS

```
ddh = CreateDirectDrawHandleA(drawhandle,sourcewidth,sourceheight,
                               a0           d0           d1
                               destwidth,destheight,taglist)
                               d2           d3           a1
```

```
APTR CreateDirectDrawHandleA(APTR,UWORD,UWORD,UWORD,UWORD,
                              struct TagItem *)
```

```
APTR CreateDirectDrawHandle (APTR,UWORD,UWORD,UWORD,UWORD,
                              Tag,...,TAG_DONE)
```

FUNCTION

Derive a handle from a drawhandle for highly optimized ('direct') drawing function calls. Currently only truecolor data (PIXFMT_0RGB_32) are supported.

INPUTS

drawhandle - drawhandle from which to derive a directdrawhandle
sourcewidth - source width [pixels]
sourceheight - source height [rows]
destwidth - dest width [pixels]
destheight - dest height [rows]

tags - pointer to an array of TagItems

TAGS

GGFX_PixelFormat - type of pixels to be processed. Currently only PIXFMT_0RGB_32 is supported.
Default: PIXFMT_0RGB_32

RESULTS

ddh - a direct-drawhandle, an object that can be passed to DirectDrawTrueColorA()

NOTES

You must free the direct-drawhandle with a matching call to DeleteDirectDrawHandle(). You are not allowed to free the underlying drawhandle before the direct-drawhandle. The consequences might be fatal.

SEE ALSO

DeleteDirectDrawHandle(), DirectDrawTrueColorA(), ObtainDrawHandleA()

1.7 guigfx.library/CreatePenShareMapA

NAME

CreatePenShareMapA - create a screen-pen manager.
CreatePenShareMap - varargs stub for CreatePenShareMapA.

SYNOPSIS

```
psm = CreatePenShareMapA(taglist)
d0                            a0

APTR CreatePenShareMapA(struct TagItem *)

APTR CreatePenShareMap(tag,...,TAG_DONE)
```

FUNCTION

This function creates a screen-pen manager.

INPUTS

tags - pointer to an array of TagItems

TAGS

GGFX_HSType (ULONG) - internal histogram type, according to the histogram types defined in render/render.h. Better you never touch this tag, unless you know exactly what you are doing. Also consider reading the 'memory' text file supplied with the render.library distribution.
Default: HSTYPE_12BIT_TURBO

RESULTS

psm - a pen-sharemap ready for usage or NULL if there was not enough memory available.

NOTES

The term 'pen-sharemap' might be confusing and has been maintained for consistency reasons. It is actually a histogram that collects color statistics. When a pen-sharemap is passed to ObtainDrawHandleA(), it allows to calculate a very specific palette.

SEE ALSO

DeletePenShareMap(), ObtainDrawHandleA(), AddPictureA(), AddPaletteA(), AddPixelArrayA()

1.8 guigfx.library/CreatePictureBitMapA

NAME

CreatePictureBitMapA - create a BitMap from a picture.
CreatePictureBitMap - varargs stub for CreatePictureBitMapA.

SYNOPSIS

```
bitmap = CreatePictureBitMapA(drawhandle,picture,tags)
d0                                a0          a1          a2
```

```
struct BitMap *CreatePictureBitMapA(APTR,APTR,struct TagItem *)
```

```
struct BitMap *CreatePictureBitMap(APTR,APTR,tag,...,TAG_DONE)
```

FUNCTION

This function creates a BitMap from a drawhandle and from a picture. This BitMap will be applicable to the drawhandle's RastPort and ColorMap, i.e. it may use colors allocated with the drawhandle, and can be blitted efficiently to the RastPort with graphics.library functions.

If the picture argument is omitted (i.e. NULL), then this function creates a blank, displayable BitMap that can be blitted efficiently to the drawhandle's RastPort. Note: The tags GGFY_DestWidth and GGFY_DestHeight are mandatory if no picture is specified, and all other tags will be ignored. (v15)

Note: The BitMap structure must be freed with graphics.library/FreeBitMap().

INPUTS

```
drawhandle - pointer to a drawhandle from ObtainDrawHandleA()
picture    - pointer to a picture, or NULL.
tags       - pointer to an array of TagItems
```

TAGS

```
GGFY_DestWidth (ULONG)
    destination width for the BitMap [pixels].
    Mandatory if no picture is supplied.
    Default: the picture's width.
```

```
GGFY_DestHeight (ULONG)
    destination height for the BitMap [rows].
    Mandatory if no picture is supplied.
```

Default: the picture's height.

GGFX_SourceX (ULONG)

left edge inside the picture where to fetch the pixels from [pixels]. Default: 0.

GGFX_SourceY (ULONG)

top edge inside the picture where to fetch the pixels from [rows]. Default: 0.

GGFX_SourceWidth (ULONG)

width of an area inside the picture [pixels]. Default: The picture's width.

GGFX_SourceHeight (ULONG)

height of an area inside the picture [rows]. Default: The picture's height.

GGFX_CallbackHook (struct Hook *)

pointer to a callback Hook structure. The associated callback function will be called from time to time while the picture is being rendered to the BitMap. The callback has to return TRUE for continuation or FALSE for abortion. It will be submitted a pointer to the picture for the object, and a message of the following type:

```
ULONG GGFX_MSGTYPE_LINEDRAWN
ULONG line_number
```

Also refer to the example provided with DrawPictureA(). Default: NULL.

GGFX_DitherMode (ULONG) - dither mode. Currently available are:

DITHERMODE_NONE

no dithering at all

DITHERMODE_FS

Floyd-Steinberg dithering

DITHERMODE_RANDOM

Random dithering. This mode is significantly slower than Floyd-Steinberg dithering.

DITHERMODE_EDD

EDD dithering. This mode is faster than Floyd-Steinberg dithering.

Default: The drawhandle's dithermode.

GGFX_DitherAmount (ULONG) - dither amount. Valid range: 0...255.

Currently, this value is of any use only for DITHERMODE_RANDOM. Default: The drawhandle's dither amount.

RESULTS

bitmap - a BitMap structure ready for being blitted to

the RastPort via `graphics.library/BltBitMapRastPort()`,
or NULL if there was not enough memory available.

SEE ALSO

`ObtainDrawHandleA()`, `DrawPictureA()`, `graphics.library/FreeBitMap()`,
`graphics.library/BltBitMapRastPort()`, `CreatePictureMaskA()`

1.9 guigfx.library/CreatePictureMaskA

NAME

`CreatePictureMaskA` - create a mask from a picture. (v15)
`CreatePictureMask` - varargs stub for `CreatePictureMaskA`.

SYNOPSIS

```
success = CreatePictureMaskA (picture, array, bytewidth, tags)
d0              a0      a1      d0      a2
```

```
BOOL CreatePictureMaskA (APTR, UBYTE *, UWORD, struct TagItem *)
```

```
BOOL CreatePictureMask (APTR, UBYTE *, UWORD, tag, ..., TAG_DONE)
```

FUNCTION

This function creates a single-bitplane mask from a picture's alpha-channel. This mask can be passed to e.g. `graphics.library/BltMaskBitMapRastPort()` for masked blitting.

If the picture contains no alpha-channel, the resulting mask will be completely opaque, i.e. all bits will be set.

Use `GGFX_Ratio` to specify a threshold. Alpha-channel values below this threshold will be rendered to a clear bit, values greater or equal to a set bit.

The array argument must point to a single bitplane, with an alignment according to $((width+15)>>4)<<1$. The bytewidth must be an even number.

Optionally, the alpha-channel is scaled to the resulting bitplane.

INPUTS

| | |
|------------------------|---|
| <code>picture</code> | - pointer to a picture |
| <code>array</code> | - pointer to a single bitplane. reserve at least $((width+15)>>4)<<1 * height$ bytes. |
| <code>bytewidth</code> | - total width of the bitplane array [bytes] |
| <code>tags</code> | - pointer to an array of <code>TagItems</code> |

TAGS

`GGFX_DestWidth` (ULONG)
destination width to be used in the resulting bitplane [pixels]. Default: the picture's width.

`GGFX_DestHeight` (ULONG)
destination height to be used in the resulting

bitplane [rows]. Default: the picture's height.

GGFX_SourceX (ULONG)

left edge inside the picture where to fetch the alpha-channel from [pixels]. Default: 0.

GGFX_SourceY (ULONG)

top edge inside the picture where to fetch the alpha-channel from [rows]. Default: 0.

GGFX_SourceWidth (ULONG)

width of an area inside the picture [pixels].
Default: The picture's width.

GGFX_SourceHeight (ULONG)

height of an area inside the picture [rows].
Default: The picture's height.

GGFX_Ratio (ULONG) - threshold. Alpha-channel values greater or equal this threshold will appear as a set bit. Default: 128

RESULTS

success - boolean, FALSE if there was not enough memory for intermediate buffers

SEE ALSO

CreatePictureBitMapA(), graphics.library/BltMaskBitMapRastPort()

1.10 guigfx.library/DeleteDirectDrawHandle

NAME

DeleteDirectDrawHandle - remove a direct-drawhandle. (v9)

SYNOPSIS

```
DeleteDirectDrawHandle(ddh)
                        a0
```

```
void DeleteDirectDrawHandle(APTR)
```

FUNCTION

this function deletes a direct-drawhandle object and frees all associated memory.

INPUTS

ddh - a direct-drawhandle, created with CreateDirectDrawHandleA()

RESULTS

none

SEE ALSO

CreateDirectDrawHandleA()

1.11 guigfx.library/DeletePenShareMap

NAME

DeletePenShareMap - dispose a pen-sharemap.

SYNOPSIS

```
DeletePenShareMap(psm)
                  a0
```

```
void DeletePenShareMap(APTR)
```

FUNCTION

This function discards a pen-sharemap and frees all associated memory and colorhandles.

INPUTS

psm - pointer to a pen-sharemap to be deleted.

SEE ALSO

CreatePenShareMapA(), RemColorHandle()

1.12 guigfx.library/DeletePicture

NAME

DeletePicture - dispose a picture.

SYNOPSIS

```
DeletePicture(picture)
                  a0
```

```
void DeletePicture(APTR)
```

FUNCTION

This function discards a picture and frees all associated memory.

INPUTS

picture - pointer to a picture to be deleted.

SEE ALSO

MakePictureA()

1.13 guigfx.library/DirectDrawTrueColorA

NAME

DirectDrawTrueColorA - draw truecolor data. (v9)
DirectDrawTrueColor - varargs stub for DirectDrawTrueColorA.

SYNOPSIS

```
success = DirectDrawTrueColorA(directdrawhandle,array,x, y,
```

```

d0                                a0          a1    d0 d1
                                taglist)
                                a2

```

```

BOOL DirectDrawTrueColorA(APTR,ULONG *,UWORD,UWORD,
                          struct TagItem *)

```

```

BOOL DirectDrawTrueColor(APTR,ULONG *,UWORD,UWORD,Tag,...,
                        TAG_DONE)

```

FUNCTION

Draw an array of truecolor data of the type PIXFMT_0RGB_32 to the RastPort associated with a direct-drawhandle's parent drawhandle. This function has got very few overhead and writes (or renders) the data as straightforward as possible.

INPUTS

```

directdrawhandle - an object derived from a drawhandle via
                  CreateDirectDrawHandleA()
array            - pointer to an array of data of the type
                  PIXFMT_0RGB_32
x,y             - destination coordinates inside the RastPort.
taglist         - pointer to an array of TagItems.

```

TAGS

```

GGFX_SourceWidth - total width of source array [pixels]
                  default: sourcewidth supplied with
                  CreateDirectDrawHandleA()

```

RESULTS

```

success         - TRUE if the call succeeded. failures are
                  currently very unlikely, but you should be
                  prepared. future implementations might
                  differ and be more likely to fail due to
                  a lack of memory.

```

SEE ALSO

```

CreateDirectDrawHandleA(), DrawPictureA()

```

1.14 guigfx.library/DoPictureMethodA

NAME

```

DoPictureMethodA - apply a method to a picture.
DoPictureMethod  - varargs stub for DoPictureMethodA.

```

SYNOPSIS

```

result = DoPictureMethodA(picture,method,arguments)
                        a0      d0      a1

```

```

ULONG DoPictureMethodA(APTR,ULONG,ULONG *)

```

```

ULONG DoPictureMethod(APTR,ULONG,argument,...)

```

FUNCTION

This function applies a method to a picture. Arguments and results depend on the specified method.

INPUTS

picture - pointer to a picture
method - method identifier (see below)
arguments - pointer to a list of arguments (see below)

METHODS

PICMTHD_AUTOCROP tags

crop the picture at its outmost borders with differing pixels. optionally limit the search for differing pixels to an area inside the picture.

TAGS

GGFX_SourceX (ULONG)
left edge of the area to check [pixels]
Default: 0

GGFX_SourceY (ULONG)
top edge of the area to check [rows]
Default: 0

GGFX_SourceWidth (ULONG)
width of the area to check [pixels]
Default: the picture's width.

GGFX_SourceHeight (ULONG)
height of the area to check [rows]
Default: the picture's height.

RESULTS

success (boolean)

PICMTHD_CREATEALPHAMASK rgb, tags

this method creates an alpha-channel for the given picture. The alpha-channel will be the difference for each pixel in the picture against the specified 0x00rrggbb value. Optionally, a clip area inside the source picture may be specified.

TAGS

GGFX_SourceX (ULONG)
source left edge in the second picture [pixels]. Default: 0

GGFX_SourceY (ULONG)
source top edge in the picture [rows].
Default: 0

GGFX_SourceWidth (ULONG)
width of an area in the picture [pixels].
Default: the picture's width.

GGFX_SourceHeight (ULONG)
height of an area in the picture [rows].
Default: the picture's height.

RESULTS

success (boolean)

NOTES

this method requires conversion to PIXFMT_0RGB_32
(see annotations below)

SEE ALSO

PICMTHD_SETALPHA

PICMTHD_CROP x, y, width, height, tags

crop a picture to a rectangle defined throughout
position (x|y) and dimensions (width|height)

TAGS

none defined

RESULTS

success (boolean)

PICMTHD_FLIPX tags

flip image (or a part of it) horizontally.

TAGS

GGFX_DestX (ULONG)
left edge of the area to flip [pixels]
Default: 0

GGFX_DestY (ULONG)
top edge of the area to flip [rows]
Default: 0

GGFX_DestWidth (ULONG)
width of the area to be flipped [pixels]
Default: the picture's width.

GGFX_DestHeight (ULONG)
height of the area to be flipped [rows]
Default: the picture's height.

RESULTS

success (boolean)

PICMTHD_FLIPY tags

flip image (or a part of it) vertically.

TAGS

GGFX_DestX (ULONG)
left edge of the area to flip [pixels]
Default: 0

GGFX_DestY (ULONG)
top edge of the area to flip [rows]
Default: 0

GGFX_DestWidth (ULONG)
width of the area to be flipped [pixels]
Default: the picture's width.

GGFX_DestHeight (ULONG)
height of the area to be flipped [rows]
Default: the picture's height.

RESULTS

success (boolean)

PICMTHD_INSERT second_picture, tags

insert a second picture (or a part of it) to the current picture. Clip areas may be specified both inside the current and the second picture. The processed pixels will be scaled to the specified dimensions, if necessary.

TAGS

GGFX_SourceX (ULONG)
source left edge where to fetch the pixels from in the second picture [pixels].
Default: 0

GGFX_SourceY (ULONG)
source top edge where to fetch the pixels from in the second picture [rows].
Default: 0

GGFX_SourceWidth (ULONG)
width of an area in the second picture [pixels]. Default: the second picture's width.

GGFX_SourceHeight (ULONG)
height of an area in the second picture [rows]. Default: the second picture's height.

GGFX_DestX (ULONG)
destination left edge where to insert the pixels into the current picture [pixels]. Default: 0

GGFX_DestY (ULONG)
destination top edge where to insert the pixels into the current picture

[rows]. Default: 0

GGFX_DestWidth (ULONG)

width to be inserted in the current picture.

[pixels]. Default: the current picture's width.

GGFX_DestHeight (ULONG)

height to be inserted in the current picture.

[rows]. Default: the current picture's height.

RESULTS

success (boolean)

NOTES

this method requires conversion to PIXFMT_0RGB_32
(see annotations below)

PICMTHD_MAPDRAWHANDLE drawhandle, tags

map a picture for optimized drawing to a drawhandle's RastPort. Drawing a picture via DrawPictureA() is much faster thereafter.

TAGS

none defined

RESULTS

success (boolean)

NOTES

- The internal representation of a picture may change at any time. The specified pixel format is only valid until the next call to DoPictureMethodA(). Use GetPictureAttrsA() to find out about the current format.
- You risk to lose color information, i.e. when a truecolor picture has to be rendered to a 8bit RastPort, for instance.

PICMTHD_MIX second_picture, tags

mix a second picture to the current picture. Clip areas may be specified both inside the current and the second picture. The processed pixels will be scaled to the specified dimensions, if necessary.

TAGS

GGFX_Ratio (ULONG)

mix ratio (0...255). Default: 128

GGFX_SourceX (ULONG)

source left edge where to fetch pixels
from in the second picture [pixels].

Default: 0

GGFX_SourceY (ULONG)
source top edge where to fetch pixels
from in the second picture [rows].
Default: 0

GGFX_SourceWidth (ULONG)
width of an area in the second picture
[pixels]. Default: the second picture's
width.

GGFX_SourceHeight (ULONG)
height of an area in the second picture
[rows]. Default: the second picture's
height.

GGFX_DestX (ULONG)
destination left edge where to apply
the operation to in the current picture
[pixels]. Default: 0

GGFX_DestY (ULONG)
destination top edge where to apply
the operation to in the current picture
[rows]. Default: 0

GGFX_DestWidth (ULONG)
width of an area for the operation to be
applied to in the current picture [pixels].
Default: the current picture's width.

GGFX_DestHeight (ULONG)
height of an area for the operation to be
applied to in the current picture [rows].
Default: the current picture's height.

RESULTS

success (boolean)

NOTES

this method requires conversion to PIXFMT_ORGB_32
(see annotations below)

SEE ALSO

PICMTHD_MIXALPHA

PICMTHD_MIXALPHA secondpicture, tags

mix a second picture to the current picture via
alpha-channel. Clip areas may be specified both inside
the current and the second picture. The processed pixels
will be scaled to the specified dimensions, if necessary.

TAGS

GGFX_SourceX (ULONG)
source left edge where to fetch pixels
from in the second picture [pixels].

Default: 0

GGFX_SourceY (ULONG)

source top edge where to fetch pixels
from in the second picture [rows].

Default: 0

GGFX_SourceWidth (ULONG)

width of an area in the second picture
[pixels]. Default: the second picture's
width.

GGFX_SourceHeight (ULONG)

height of an area in the second picture
[rows]. Default: the second picture's
height.

GGFX_DestX (ULONG)

destination left edge where to apply
the operation to in the current picture
[pixels]. Default: 0

GGFX_DestY (ULONG)

destination left edge where to apply
the operation to in the current picture
[rows]. Default: 0

GGFX_DestWidth (ULONG)

width of an area for the operation to be
applied to in the current picture [pixels].
Default: the current picture's width.

GGFX_DestHeight (ULONG)

height of an area for the operation to be
applied to in the current picture [rows].
Default: the current picture's height.

RESULTS

success (boolean)

NOTES

this method requires conversion to PIXFMT_0RGB_32
(see annotations below)

SEE ALSO

PICMTHD_SETALPHA, PICMTHD_MIX

PICMTHD_NEGATIVE tags

invert the colors of the picture (or a part of it)

TAGS

GGFX_DestX (ULONG)

left edge of the area to invert [pixels]
Default: 0

GGFX_DestY (ULONG)
top edge of the area to invert [rows]
Default: 0

GGFX_DestWidth (ULONG)
width of the area to invert [pixels]
Default: the picture's width.

GGFX_DestHeight (ULONG)
height of the area to invert [rows]
Default: the picture's height.

RESULTS

success (boolean)

NOTES

this method requires conversion to PIXFMT_ORGB_32
(see annotations below)

PICMTHD_RENDER pixelformat, tags

render a picture to a specified pixel format. Valid pixel
formats are as follows:

PIXFMT_CHUNKY_CLUT
chunky bytes

PIXFMT_ORGB_32
ULONG 0x00rrggbb truecolor data

PIXFMT_RGB_24
UBYTE 0xrr,0xgg,0xbb truecolor data

TAGS

none defined

RESULTS

success (boolean)

NOTES

- The internal representation of a picture may change at any time. The specified pixel format is only valid until the next call to DoPictureMethodA(). Use GetPictureAttrsA() to find out about the current format.
- You risk to lose color information, i.e. when a truecolor picture is rendered to PIXFMT_CHUNKY_CLUT.

PICMTHD_SCALE width, height, tags

scale a picture to the specified dimensions.

TAGS

none defined

RESULTS

success (boolean)

NOTE

This function fails if applied to a static buffer, and when the image needs to grow. In this case, specify `GGFX_Independent` or set a larger buffer with `GGFX_BufferSize` when creating the picture with `MakePictureA()`.

`PICMTHD_SET` `rgb, tags`

set a picture (or a part of it) to the specified RGB value.

TAGS

`GGFX_DestX` (ULONG)
destination left edge [pixels]
Default: 0

`GGFX_DestY` (ULONG)
destination top edge [rows]
Default: 0

`GGFX_DestWidth` (ULONG)
width to be affected [pixels]
Default: the picture's width.

`GGFX_DestHeight` (ULONG)
height to be affected [rows]
Default: the picture's height.

RESULTS

success (boolean)

NOTES

if you apply this method to a picture of the format `PIXFMT_CHUNKY_CLUT`, it cannot be guaranteed that the specified RGB value is exactly hit. you can use `PICMTHD_RENDER` in order to convert the picture to `PIXFMT_ORGB_32` before.

`PICMTHD_SETALPHA` `alpha-array, width, height, tags`

set an alpha-channel array for the current picture. The alpha-channel is a plain array of chunky-bytes, defining a mixing ratio for each pixel. The alpha-channel array will be scaled to fit exactly to the current picture, unless you specify other dimensions. Passing a NULL pointer for alpha-array will discard an existing alpha-channel.

TAGS

GGFX_DestX (ULONG)
destination left edge where to insert
the alpha-channel into the current
picture [pixels]. Default: 0

GGFX_DestY (ULONG)
destination top edge where to insert
the alpha-channel into the current
picture [rows]. Default: 0

GGFX_DestWidth (ULONG)
width to be inserted to the current
picture [pixels]. Default: the current
picture's width.

GGFX_DestHeight (ULONG)
height to be inserted to the current
picture [rows]. Default: the current
picture's height.

RESULTS

success (boolean)

NOTES

this method requires conversion to PIXFMT_0RGB_32
(see annotations below)

SEE ALSO

PICMTHD_CREATEALPHAMASK

PICMTHD_TEXTURE texturepic, coordinates, tags

draw a texture to the current picture, texture-mapped
via an array of coordinates. texturepic is a pointer to
a picture that contains the texture, coordinates is a
pointer to an array of 4 WORD pairs of x/y coordinates
each. They form a trapezoid inside the current picture
for the texture picture to be mapped to. border clipping
is fully implemented.

TAGS

GGFX_SourceX (ULONG)
source left edge inside the texture
[pixels]. Default: 0

GGFX_SourceY (ULONG)
source top edge inside the texture
[rows]. Default: 0

GGFX_SourceWidth (ULONG)
texture width [pixels]. Default:
the texturepic's width.

GGFX_SourceHeight (ULONG)
texture height [rows]. Default:

the texturepic's height.

GGFX_DestX (ULONG)

destination left edge where to apply
the trapezoid to the current picture
[pixels]. Default: 0

GGFX_DestY (ULONG)

destination top edge where to apply
the trapezoid to the current picture
[rows]. Default: 0

GGFX_DestWidth (ULONG)

maximum width to be inserted to the
current picture [pixels]. Default: the
current picture's width.

GGFX_DestHeight (ULONG)

maximum height to be inserted to the
current picture [rows]. Default: the
current picture's height.

RESULTS

success (boolean)

NOTES

this method depends on both pictures to be in
the same format. DoPictureMethodA() tries to
convert either of the involved pictures to the
other's format. (see annotations below)

SEE ALSO

render.library texture-mapping documentation

PICMTHD_TINTALPHA rgb, tags

tint the picture with the given 0x00rrggbb. the mixing
ratio is defined throughout the picture's alpha-channel.

TAGS

GGFX_DestX (ULONG)

destination left edge where to apply
the operation [pixels]. Default: 0

GGFX_DestY (ULONG)

destination left edge where to apply
the operation [rows]. Default: 0

GGFX_DestWidth (ULONG)

width of an area for the operation to be
applied to [pixels].
Default: the picture's width.

GGFX_DestHeight (ULONG)

height of an area for the operation to be
applied to [rows].

Default: the picture's height.

RESULTS

success (boolean)

NOTES

this method requires conversion to PIXFMT_ORGB_32
(see annotations below)

SEE ALSO

PICMTHD_TINT, PICMTHD_MIXALPHA

PICMTHD_TINT rgb, tags

tint the picture with the given 0x00rrggbb value, and
optionally with a specific ratio.

TAGS

GGFX_Ratio (ULONG)
mix ratio (0...255). Default: 128

GGFX_DestX (ULONG)
destination left edge where to apply
the operation [pixels]. Default: 0

GGFX_DestY (ULONG)
destination left edge where to apply
the operation [rows]. Default: 0

GGFX_DestWidth (ULONG)
width of an area for the operation to be
applied to [pixels].
Default: the picture's width.

GGFX_DestHeight (ULONG)
height of an area for the operation to be
applied to [rows].
Default: the picture's height.

RESULTS

success (boolean)

NOTES

this method requires conversion to PIXFMT_ORGB_32
(see annotations below)

SEE ALSO

PICMTHD_TINTALPHA, PICMTHD_MIXALPHA

RESULTS

result - return value (specific for the applied method)

NOTES

Methods that require conversion to PIXFMT_ORGB_32 will fail in
a static buffer, i.e. when the picture was created with

MakePictureA() in the format PIXFMT_CHUNKY_CLUT, and without a buffer overhang or GGFX_Independent. See MakePictureA() for further details.

SEE ALSO

MakePictureA(), ObtainDrawHandleA(), DrawPictureA()

1.15 guigfx.library/DrawPictureA

NAME

DrawPictureA - draw a picture to a drawhandle.
DrawPicture - varargs stub for DrawPictureA.

SYNOPSIS

```
success = DrawPictureA(drawhandle,picture,x, y, tags)
d0              a0              a1              d0 d1 a2
```

```
BOOL DrawPictureA(APTR,APTR,UWORD,UWORD,struct TagItem *)
```

```
BOOL DrawPicture(APTR,APTR,UWORD,UWORD,tag,...,TAG_DONE)
```

FUNCTION

This function draws a picture to the RastPort associated with a drawhandle. Optionally, the picture will be scaled to the specified dimensions. A clip area inside the picture may be specified as well.

INPUTS

```
drawhandle - pointer to a drawhandle from ObtainDrawHandleA()
picture    - pointer to a picture
x          - left edge inside the RastPort [pixels]
y          - top edge inside the RastPort [rows]
tags       - pointer to an array of TagItems
```

TAGS

```
GGFX_SourceX (ULONG)
    left edge inside the picture where to fetch the pixels
    from [pixels]. Default: 0.
```

```
GGFX_SourceY (ULONG)
    top edge inside the picture where to fetch the pixels
    from [rows]. Default: 0.
```

```
GGFX_SourceWidth (ULONG)
    width of an area inside the picture [pixels].
    Default: The picture's width.
```

```
GGFX_SourceHeight (ULONG)
    height of an area inside the picture [rows].
    Default: The picture's height.
```

```
GGFX_DestWidth (ULONG)
    destination width for the picture to be drawn [pixels].
    Default: the picture's width.
```

GGFX_DestHeight (ULONG)
destination height for the picture to be drawn [rows].
Default: the picture's height.

GGFX_CallbackHook (struct Hook *)
pointer to a callback Hook structure. The associated
callback function will be called from time to time
while the picture is being drawn.
The callback has to return TRUE for continuation or FALSE
for abortion. It will be submitted a pointer to the
picture for the object, and a message of the following
type:

```
ULONG GGFX_MSGTYPE_LINEDRAWN
ULONG line_number
```

Also refer to the example below.
Default: NULL.

GGFX_DitherMode (ULONG) - dither mode. Currently available are:

```
DITHERMODE_NONE
    no dithering at all

DITHERMODE_FS
    Floyd-Steinberg dithering

DITHERMODE_RANDOM
    Random dithering. This mode is significantly
    slower than Floyd-Steinberg dithering.

DITHERMODE_EDD
    EDD dithering. This mode is faster than
    Floyd-Steinberg dithering.
```

Default: The drawhandle's dithermode.

GGFX_DitherAmount (ULONG) - dither amount. Valid range: 0...255.
Currently, this value is of any use only for
DITHERMODE_RANDOM. Default: The drawhandle's dither amount.

GGFX_AutoDither (BOOL) - automatic dither activation.
If set to TRUE, dithering is automatically activated for
drawing a particular picture to a particular environment,
when the loss of color information would exceed a certain
threshold (see below). Default: TRUE

GGFX_RastLock (struct SignalSemaphore *) - pointer to an
initialized exec.library SignalSemaphore which is
used for RastPort sharing between tasks. if you want
to draw to the drawhandle's RastPort while another
task is rendering to this RastPort with DrawPictureA(),
you must supply this argument and enclose all accesses
to the RastPort with ObtainSemaphore()/ReleaseSemaphore()
pairs. default: NULL (v16)

RESULTS

success - TRUE if the picture could be drawn, FALSE if there was not enough memory available. Another reason for this function to fail is that the optional callback hook returned FALSE.

NOTES

There is almost no overhead for scaling. Scaling is extremely fast and may be considered 'gratis'.

EXAMPLE

The callback hook allows to interrupt DrawPictureA() at any time. A simple callback function might look like this:

```
ULONG __saveds __asm abortdrawfunc(register __a0 struct Hook *hook)
{
    ULONG abortsignal = 1 << *((BYTE *) (hook->h_Data));
    if (SetSignal(0, 0) & abortsignal)
    {
        return FALSE;
    }
    else
    {
        return TRUE;
    }
}
```

In this example, an abortion signal was allocated and made available to the function via h_Data. If the signal arrives, the callback function returns FALSE to DrawPictureA(), and drawing will be interrupted.

Note: Not all internal drawing-routines actually execute the hook function more than once. This mainly depends on the typical speed for a particular drawing routine or certain graphics.library or cybergraphics.library implementations. At least it is supported when scaling and rendering is involved to the drawing process.

SEE ALSO

ObtainDrawHandleA(), CreatePictureBitMapA()

1.16 guigfx.library/GetPictureAttrsA

NAME

GetPictureAttrsA - get picture attributes.
GetPictureAttrs - varargs stub for GetPictureAttrsA.

SYNOPSIS

```
count = GetPictureAttrsA(picture,tags)
d0          a0          a1

ULONG GetPictureAttrsA(APTR,struct TagItem *)

ULONG GetPictureAttrs(APTR,tag,...,TAG_DONE)
```

FUNCTION

This function obtains a list of picture attributes. It returns the number of attributes that have been retrieved actually.

INPUTS

picture - pointer to a picture
tags - pointer to an array of TagItems

TAGS

PICATTR_Width (ULONG *)
 The picture's width [pixels]

PICATTR_Height (ULONG *)
 The picture's height [rows]

PICATTR_PixelFormat (ULONG *)
 The picture's internal pixel format. Currently this can be PIXFMT_CHUNKY_CLUT, PIXFMT_ORGB_32, or PIXFMT_RGB_24.

PICATTR_RawData (APTR *)
 Pointer to the picture's raw data. Operate on the raw pixel array only with knowledge of the actual pixel format. Warning: The internal representation of a picture may change with every call to DoPictureMethodA() or drawing functions.

PICATTR_AspectX (ULONG *)
 Horizontal pixel aspect.

PICATTR_AspectY (ULONG *)
 Vertical pixel aspect.

PICATTR_AlphaPresent (BOOL)
 indicates if an alpha-channel is present.

RESULTS

count - the number of attributes that could be retrieved.

1.17 guigfx.library/IsPictureA

NAME

IsPictureA - determine whether a file is a picture or not. (v4)
IsPicture - varargs stub for IsPictureA.

SYNOPSIS

```
ispicture = IsPictureA(filename, tags)
d0                a0                a1

BOOL IsPictureA(char *, struct TagItem *)

BOOL IsPicture(char *, tag, ..., TAG_DONE)
```

FUNCTION

This function checks if the specified file could be loaded as a picture with LoadPictureA().

INPUTS

filename - name of the file to be checked
tags - pointer to an array of TagItems

TAGS

RESULTS

ispicture - TRUE if the specified file is recognized as a picture that could be loaded with LoadPictureA().

SEE ALSO

LoadPictureA()

1.18 guigfx.library/LoadPictureA

NAME

LoadPictureA - load a picture file.
LoadPicture - varargs stub for LoadPictureA.

SYNOPSIS

```
picture = LoadPictureA(filename,tags)
d0              a0              a1
```

```
APTR LoadPictureA(char *,struct TagItem *)
```

```
APTR LoadPicture(char *,tag,...,TAG_DONE)
```

FUNCTION

This function loads a picture. Currently, this is implemented via picture.class datatypes.

INPUTS

filename - name of the file to be loaded
tags - pointer to an array of TagItems

TAGS

GGFX_ErrorCode (LONG *)
 Pointer to a variable that will receive a standard DOS error code. This will be NULL if loading was successful.
 Default: NULL

GGFX_UseMask (ULONG) (v15)
 boolean to indicate whether a transparency color, an alpha-channel or a mask (if present) should be inserted to the picture. Note: This tag requires the picture to be converted to PIXFMT_ORGB_32.
 Default: FALSE

GGFX_HSType (ULONG) - picture's internal histogram type, according to the histogram types defined in render/render.h. Better you never touch this tag, unless you know exactly what you are doing. Consider reading the 'memory' text file supplied with the render.library distribution. You do not need this tag under normal circumstances. Default: not defined (will be set to the pen-sharemap's histogram type, or to the default type when needed)

RESULTS

picture - pointer to a picture or NULL if something went wrong. The exact reason for failure can be obtained via the GGFX_ErrorCode tag.

NOTES

- As for current datatype implementations, alpha-channels do not seem to be supported. The datatype might translate it to a single bitplane. guigfx.library, on the other hand, does not (yet) support single-bitplane masks, so masks and transparency colors will be translated to alpha-channels.

SEE ALSO

DeletePicture(), IsPictureA(), MakePictureA(), ReadPictureA()

1.19 guigfx.library/LockPictureA

NAME

LockPictureA - lock picture attributes. (v3)
LockPicture - varargs stub for LockPictureA.

*** obsolete ***

SYNOPSIS

```
success = LockPictureA (picture, flags, arguments)
d0          a0          d0      a1
```

```
BOOL LockPictureA (APTR, ULONG, ULONG *)
```

```
BOOL LockPicture (APTR, ULONG, argument, ...)
```

FUNCTION

This function locks certain picture attributes and prevents the picture from internal conversions that affect the specified flags.

INPUTS

picture - pointer to a picture
flags - locking flags
arguments - flag-specific arguments

FLAGS

LOCKMODE_DRAWHANDLE drawhandle

lock the picture to the specified drawhandle. this leads to optimized drawing without the need to render. combine with LOCKMODE_FORCE if you want to lock the image even if color information would be lost.

RESULTS

success - TRUE if locking was successful, FALSE if locking is not possible, or if locking required a conversion with loss of color information.

NOTES

This function is currently (v4) not working, and it will always return FALSE. If you need optimized drawing, use the method PICMTHD_MAPDRAWHANDLE instead.

SEE ALSO

UnlockPicture(), DoPictureMethodA()

1.20 guigfx.library/MakePictureA

NAME

MakePictureA - make a picture from raw data or from a BitMap.
MakePicture - varargs stub for MakePictureA.

SYNOPSIS

```
picture = MakePictureA(data,width,height,tags)
d0              a0    d0    d1    a1
```

```
APTR MakePictureA(APTR,UWORD,UWORD,struct TagItem *)
```

```
APTR MakePicture(APTR,UWORD,UWORD,tag,...,TAG_DONE)
```

FUNCTION

This function makes a picture from an array of raw data (or a part of it), or from a BitMap structure (or a part of it). Optionally, memory is allocated for a 'blank' picture. Optionally, the picture will be scaled.

Raw data is not incorporated to the picture, instead it is referenced at its original location in memory, unless you specify the tag GGFY_Independent. (This does not apply to BitMap structures - pictures created from BitMaps are always independent.)

If GGFY_Independent is not specified (and your picture is taken from its original location in memory), you may additionally specify a buffer 'overhang' with the tag GGFY_BufferSize. This allows internal conversions which require the image to grow at its original location in memory. You must be the owner of that memory, of course.

INPUTS

- data - pointer to
 - an array of truecolor data,
 - an array of chunky pixels,
 - a BitMap structure
 - or NULL.
- width - total width of the source array or BitMap [pixels]
- height - total height of the source array or BitMap [rows]
- tags - pointer to an array of TagItems

TAGS

GGFX_PixelFormat (ULONG) - pixel format. Currently defined are

- PIXFMT_CHUNKY_CLUT
 - chunky bytes, directly acting as indices
 - to a color-lookup-table.
- PIXFMT_ORGB_32
 - truecolor pixels (ULONG 0xaarrggbb).
- PIXFMT_BITMAP_CLUT
 - a BitMap structure with normal palette lookup.
 - You must also specify the GGFX_Palette and
 - GGFX_NumColors tags.
- PIXFMT_BITMAP_HAM8
 - a BitMap structure with HAM8 color lookup.
 - You must also specify the GGFX_Palette and
 - GGFX_NumColors tags.
- PIXFMT_BITMAP_HAM6
 - a BitMap structure with HAM6 color lookup.
 - You must also specify the GGFX_Palette and
 - GGFX_NumColors tags.
- PIXFMT_BITMAP_RGB
 - a BitMap structure which is assumed to contain
 - truecolor data. This may apply to CyberGraphX
 - bitmaps.

Default: PIXFMT_CHUNKY_CLUT

GGFX_Palette (APTR) - pointer to a color table. If this tag is not specified with PIXFMT_CHUNKY_CLUT, a default palette of 256 grey tones will be generated.
Default: NULL

GGFX_NumColors (ULONG) - number of colors in the color table. This tag is mandatory when GGFX_Palette is specified (see above). Default: not defined

GGFX_PaletteFormat (ULONG) - format of the palette. Currently defined are:

- PALFMT_RGB8
 - ULONG 0x00rrggbb

PALFMT_RGB32

ULONG red, green, blue. This is the LoadRGB32() format without trailing longword.

Default: PALFMT_RGB8

GGFX_SourceX (ULONG) - left edge of an area inside the array or BitMap [pixels]. Default: 0.

GGFX_SourceY (ULONG) - top edge of an area inside the array or BitMap [rows]. Default: 0.

GGFX_SourceWidth (ULONG) - width of an area inside the array or BitMap [pixels]. Default: width.

GGFX_SourceHeight (ULONG) - height of an area inside the array or BitMap [rows]. Default: height.

GGFX_DestWidth (ULONG) - destination width of the resulting picture [pixels]. Default: GGFX_SourceWidth.

GGFX_DestHeight (ULONG) - destination height for the resulting picture [rows]. Default: GGFX_SourceHeight.

GGFX_BufferSize (ULONG) - total size of the specified buffer in bytes. This defines an 'oversized' buffer for the array of pixels. It informs the picture to what size it may grow for internal conversions. This tag is ignored when you supply a BitMap structure, or when GGFX_Independent is specified. Default: Required size in bytes for width * height * bytes_per_pixel.

GGFX_AspectX (ULONG) - picture's horizontal aspect. Default: 1

GGFX_AspectY (ULONG) - picture's vertical aspect. Default: 1

GGFX_AlphaPresent (BOOL) - flag to indicate that the array contains alpha-channel information. This tag is only considered with PIXFMT_ORGB_32. Default: FALSE

GGFX_Independent (BOOL) - If set to TRUE, the pixel array will always be copied to a separate buffer that is maintained with the picture internally. This tag is meaningless when the input data is a BitMap structure. Default: FALSE

GGFX_HSType (ULONG) - picture's internal histogram type, according to the histogram types defined in render/render.h. Better you never touch this tag, unless you know exactly what you are doing. Consider reading the 'memory' text file supplied with the render.library distribution. You do not need this tag under normal circumstances. Default: not defined (will be set to a pensharemap's histogram type, or to the default type when needed)

RESULTS

picture - pointer to a picture or NULL if something went wrong.

SEE ALSO

DeletePicture(), LoadPictureA(), ReadPictureA()

1.21 guigfx.library/ObtainDrawHandleA

NAME

ObtainDrawHandleA - obtain a handle for drawing.

ObtainDrawHandle - varargs stub for ObtainDrawHandleA.

SYNOPSIS

```
drawhandle = ObtainDrawHandleA(pensharemap,rastport,colormap,tags)
d0                                a0                                a1                                a2                                a3
```

```
APTR ObtainDrawHandleA(APTR,struct RastPort *,struct ColorMap *,
                        struct TagItem *)
```

```
APTR ObtainDrawHandle(APTR,struct RastPort *,struct ColorMap *,
                      tag,...,TAG_DONE)
```

FUNCTION

This function obtains a drawhandle for drawing to a RastPort. Depending on the RastPort's environment, pens may be allocated from the ColorMap.

Before a pen-sharemap is passed to this function, it has to be loaded with colors via AddPictureA(), AddPaletteA(), and/or AddPixelArrayA(). Otherwise ObtainDrawHandleA() returns NULL.

Optionally, you may specify NULL for the pen-sharemap argument, in which case a drawhandle for a static palette will be generated.

INPUTS

```
pensharemap - pointer to a pen-sharemap created with
               CreatePenShareMapA(), or NULL.
rastport    - pointer to a RastPort
colormap    - pointer to a ColorMap. Usually, this is
               screen->Viewport.ColorMap of the rastport's screen.
tags        - pointer to an array of TagItems
```

TAGS

```
OBP_Precision (ULONG) - precision for pen allocations,
                        according to the definitions in graphics/view.h.
                        See also graphics.library/ObtainBestPenA().
                        Default: PRECISION_IMAGE.
```

Note: The default precision suffices for almost every application. ObtainDrawHandleA() obtains pens in an extremely effective way. You get excellent results even with lower precisions. Commodore's idea with

ObtainBestPenA() was to create a fair and effective pen-sharing mechanism, and ObtainDrawHandleA() behaves in perfect accordance to this intention. Never use insane patches for ObtainBestPenA().

GGFX_DitherMode (ULONG) - dither mode. Currently available are:

DITHERMODE_NONE
no dithering at all

DITHERMODE_FS
Floyd-Steinberg dithering

DITHERMODE_RANDOM
Random dithering. This mode is significantly slower than Floyd-Steinberg dithering.

DITHERMODE_EDD
EDD dithering. This mode is faster than Floyd-Steinberg dithering.

Default: DITHERMODE_FS.

GGFX_DitherAmount (ULONG) - dither amount. Valid range: 0...255. Currently this value is of any use only for DITHERMODE_RANDOM. Default: 40

GGFX_AutoDither (BOOL) - automatic dither activation. If set to TRUE, dithering is automatically activated for drawing a particular picture to a particular environment, when the loss of color information would exceed a certain threshold (see below). Default: TRUE

GGFX_DitherThreshold (ULONG) - threshold for automatic dithering. The lower, the earlier automatic dithering is activated. Useful thresholds range between 10 and 10000. Refer to render.library/RGBArrayDiversityA() for further details. better you do not use this tag unless you have a good reason to. let the user customize it with the environment variable AUTODITHERTHRESHOLD. Default: 250

GGFX_MaxAllocPens (ULONG) - limit for the number of pens to be allocated from the ColorMap. Do not use this feature unless you have a good reason to. Valid range: 0...256. Default: not defined

GGFX_ModeID (ULONG) - screen's modeID. Currently, this is required for guigfx.library to detect HAM modes. The full HAM color range can be achieved only with this tag specified. Default: INVALID_ID (no HAM detection takes place)

RESULTS

drawhandle - pointer to a handle for drawing to rastports.
NULL if something went wrong.

SEE ALSO

ReleaseDrawHandle(), CreatePenShareMapA(), DrawPictureA(),

```
graphics.library/ObtainBestPenA(),
render.library/RGBArrayDiversityA()
```

1.22 guigfx.library/ReadPictureA

NAME

ReadPictureA - read a picture from a RastPort.
ReadPicture - varargs stub for ReadPictureA.

SYNOPSIS

```
picture = ReadPictureA(rastport,colormap,x, y, width,height,tags)
d0          a0          a1          d0 d1 d2          d3          a2
```

```
APTR ReadPictureA(struct RastPort *,struct ColorMap *,UWORD,UWORD,
                  UWORD,UWORD,struct TagItem *)
```

```
APTR ReadPicture(struct RastPort *,struct ColorMap *,UWORD,UWORD,
                  UWORD,UWORD,tag,...,TAG_DONE)
```

FUNCTION

This function reads a picture from a RastPort (or a part of it), and optionally scales it to the specified dimensions.

INPUTS

```
rastport - pointer to a RastPort where to fetch the pixels from
colormap - pointer to a ColorMap where to fetch color information
           from. Usually this is screen->ViewPort.ColorMap of the
           specified RastPort's Screen.
x         - left edge in the RastPort [pixels]
y         - top edge in the RastPort [rows]
width     - width of the area to be read [pixels]
height    - height of the area to be read [rows]
tags      - pointer to an array of TagItems
```

TAGS

```
GGFX_DestWidth (ULONG) - destination width [pixels].
Default: width.
```

```
GGFX_DestHeight (ULONG) - destination height [rows].
Default: height.
```

```
GGFX_AspectX (ULONG) - horizontal pixel aspect for the resulting
picture. Default: 1
```

```
GGFX_AspectY (ULONG) - vertical pixel aspect for the resulting
picture. Default: 1
```

```
GGFX_ModeID (ULONG) - screen's mode ID. currently required for
determining HAM rastports. Default: none
```

```
GGFX_HSType (ULONG) - picture's internal histogram type, according
to the histogram types defined in render/render.h.
Better you never touch this tag, unless you know exactly
what you are doing. Consider reading the 'memory' text
```

file supplied with the render.library documentation.

RESULTS

picture - pointer to a picture or NULL if not enough memory.

SEE ALSO

LoadPictureA(), MakePictureA()

1.23 guigfx.library/ReleaseDrawHandle

NAME

ReleaseDrawHandle - free a drawhandle.

SYNOPSIS

```
ReleaseDrawHandle(drawhandle)
                    a0
```

```
void ReleaseDrawHandle(APTR)
```

FUNCTION

This function discards a drawhandle, frees associated memory, and returns allocated pens (if any) to the related ColorMap.

INPUTS

drawhandle - drawhandle obtained via ObtainDrawHandleA()

SEE ALSO

ObtainDrawHandleA()

1.24 guigfx.library/RemColorHandle

NAME

RemColorHandle - manually remove a colorhandle.

SYNOPSIS

```
RemColorHandle(colorhandle)
                    a0
```

```
void RemColorHandle(APTR)
```

FUNCTION

This function removes particular color information from a pen-sharemap. Further calls to ObtainDrawHandleA() may lead to different pen allocations then.

INPUTS

colorhandle - pointer to a colorhandle from
AddPictureA(), AddPaletteA(),
or AddPixelArrayA()

NOTE

DeletePenShareMap() arbitrarily frees all its colorhandles. There is no need to manually remove them. This function is only required if you wish to modify a pen-sharemap and then call ObtainDrawHandleA() again.

Calling RemColorHandle() for colorhandles that have been removed with DeletePenShareMap() will be fatal.

SEE ALSO

AddPictureA(), AddPaletteA(), AddPixelArrayA(),
DeletePenShareMap(), ObtainDrawHandleA()

1.25 guigfx.library/UnLockPicture

NAME

UnLockPicture - unlock picture attributes (v3)

*** obsolete ***

SYNOPSIS

UnLockPicture(picture, flags)
 a0 d0

UnLockPicture(APTR, ULONG)

FUNCTION

This function frees picture attributes that have been locked with LockPictureA().

INPUTS

picture - pointer to a picture
flags - flags to unlock

RESULTS

none

SEE ALSO

LockPictureA()

NOTES

This function will currently (v4) do nothing. Read the annotations in LockPictureA().
