ImageManagerClasses

COLLABORATORS

	TITLE : ImageManagerClasses		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 8, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 ImageManagerClasses **1**

1.1 main 1

1.2 credits 2

1.3 notes 2

1.4 link 3

1.5 file 3

1.6 decoder 3

1.7 scalex 4

1.8 scaley 4

1.9 fsdither 4

1.10 ordereddither 5

1.11 randomdither 5

1.12 remap 5

1.13 container 6

1.14 raster 6

1.15 bitmap 7

1.16 probe 7

Chapter 1

ImageManagerClasses

1.1 main

```
                Here’s a list of the classes which are currently build into the ↩
                   ImageManager.library.
Follow a link to learn more about the class.

This guide is PRELIMINARY!



                 Link
                 - Superclass


                 File
                 - Load a file


                 Decoder
                 - Decode binary data to 24 bit image data


                 ScaleX
                 - Change the width of 24 bit image data

                 ScaleY
                 - Change the height of 24 bit image data


                 FSDither
                 - Apply error diffusion to 24 bit data

                 OrderedDither
                 - Apply ordered dither to 24 bit data

                 RandomDither
                 - Add random noise to 24 bit data


                 Remap
                 - Remap 24 bit data to palette mapped (chunky) data


                 Raster
                 - Save data (chunky or 24 bit) to a rastport

                 Bitmap
                 - Store data in a bitmap (allocated by the class itself)


                 Container
                 - Wrapper class for quantization objects (dither + remap)


                 Probe
                 - Probe for debug purposes



                 Credits
                 - The credits


                 Notes
                 - Various notes that you should read!
```

1.2 credits

ImageManager.library is created by Allan Odgaard <Duff@DIKU.DK>, http://www.diku.dk/students/duff/

Decoder modules are created by Gunther Nikl <GNikl@Informatik.Uni-Rostock.De>

1.3 notes

I'd like to introduce some sort of 'Cache' class, which will cache displayable image data. Though I'm not sure how to add such a class in a flexible and transparent way, as one of the goals should be that if a second program loads an image, which are currently being processed, then it should be "added to the chain", so that the new program can also do incremental display.

A solution might involve changing the current API, so be prepared!

Also, I think the environment variables should probably be ditched in favor of pr. class tags.

There should also be a class to convert the alpha channel into a mask which is useable with BltMaskBitMapRastPort() -- and one to handle a complex mask (like the AlphaBlend example)

Feedback is welcome! Duff@DIKU.DK

1.4 link

```
Description
The Link class is the superclass for all chain objects.

Methods
IMM_NewFrame    - Create a new frame
IMM_ReceiveData - Receive data
IMM_EndFrame    - Complete current frame
IMM_Abort       - Abort creation of current frame

Attributes
IMA_Next [ISG]
   Pointer to next object in the chain.
   The superclass will redirect all of the above methods to this (next)
   object. So if for example you want to pass on data to the next object in the
   chain, simply pass it on to your superclass, which will check if there is a
   next object in the chain. If one exists, it will pass on the method (with
   arguments) to that object, and return its result. If no objects follow then
   TRUE will be returned.
```

1.5 file

```
Description
This class will pass the contents of a specified file to its superclass.

Methods
IMM_File_Load - Start fetching data from the file.

Attributes
IMA_File_BufferSize [I..] (default: 3Kb)
   Adjust the size of the buffer used.

Notes
The current implementation of this class uses AsyncIO.library by Magnus Holmgren.
```

1.6 decoder

```
Description
This class accepts binary data (via IMM_ReceiveData) and decodes it to 24 bit
image data (if possible) and passes this data to the superclass.

Methods
None.

Attributes
None.

Notes
If no decoder module exists for the given image format, the class will use the
'Datatype.Decoder' which uses DTM_READPIXELARRAY to retrieve data (as RGBA)
from the datatype object. This only works for some of the V43 picture
```

datatypes. One of them is the one that comes with OS 3.5 - so if it fails to work then it’s time for an upgrade! :-)

1.7 scalex

Description
This class scales the width of a picture qualitative.

Methods
None.

Attributes
IMA_ScaleX_Width [IS.]
   Set the new width in pixels.
IMA_ScaleX_Percent [IS.]
   Set the new width in percent. E.g. "200" would double the width of the image.

Notes
The alpha channel is also scaled qualitative, so the result of scaling a picture with a simple mask results in one which is complex.

1.8 scaley

Description
This class scales the height of a picture qualitative.

Methods
None.

Attributes
IMA_ScaleY_Height [IS.]
   Set the new height in pixels.
IMA_ScaleY_Percent [IS.]
   Set the new height in percent. E.g. "200" would double the height of the image.

Notes
The alpha channel is also scaled qualitative, so the result of scaling a picture with a simple mask results in one which is complex.

1.9 fsdither

Description
Apply floyd steinberg error diffusion to the 24 bit data received with IMM_ReceiveData.

Methods
None.

Attributes
IMA_FSDither_ColourCube [IS.]

```
   Specify the colour cube (see IM_ObtainColourCube()) which should be used to
   decide the error diffusion. If none is given then the class assume that the
   target display is 16 bit with 5-6-5 bits pr. gun.

Notes
I would like a way to decide the bit distribution for non CLUT screens, but
since this must be checked pr. pixel then I'm afraid it will hurt performance
to much...
```

1.10 ordereddither

```
Description
Add ordered (pattern) dither to 24 bit data.

Methods
None.

Attributes
None.

Notes
Do we need a threshold setting? Matrix size?
```

1.11 randomdither

```
Description
Apply random noise to 24 bit image data. This will (often) improve a later image ↩
   quantization.

Methods
None.

Attributes
None.

Notes
Do we need a threshold setting?
```

1.12 remap

```
Description
Remap 24 bit data to palette mapped (chunky) data

Methods
None.

Attributes
IMA_Remap_ColourCube [I..]
   Colour cube used for remapping. If none is given, OM_NEW will fail.
```

1.13 container

```
Description
Rather than examining the destination screen and create dither + remap objects
as required then you can instead create an instance of this class, and provide
it with a screen pointer. The class will then create the necessary objects,
and add them to the chain.

Methods
None.

Attributes
IMA_Container_Screen [I..]
   The destination screen. Required!

IMA_Container_ColourCube [I.G]
   You can supply a colour cube, which will be given to the remap and
   dither objects, if the destination screen is a CLUT screen.
   The class will create its own colour cube, if none is given.

IMA_Container_ReleaseColourCube [I..]
   If you don't supply a colour cube, but wants to dispose this object
   after rendering the image data, then set this attribute to FALSE, to stop
   the class from releasing its colour cube.
   You must then obtain the colour cube that this object allocated, and
   free it later yourself (with IM_ReleaseColourCube())

Notes
This class will use the environment variable 'ImageManager/Dither' to decide
which dither type to use. Maybe this should instead be a tag?
```

1.14 raster

```
Description
This class will save all image data to the user supplied rastport.

Methods
None.

Attributes
IMA_Raster_RastPort [IS.]
   The rastport used as destination for the image data.

IMA_Raster_Left [IS.]
   Placement of image in rastport.

IMA_Raster_Top [IS.]
   Placement of image in rastport.

IMA_Raster_Right [IS.]
   Right clip boundary.

IMA_Raster_Bottom [IS.]
   Bottom clip boundary.
```

```
IMA_Raster_OffsetX [IS.]
   X offset into image (must be less than image width)

IMA_Raster_OffsetY [IS.]
   Y offset into imgae (must be less than image height)

Notes
I should probably add a tag for making the class allocate its own bitmap, if
no rastport is given.
```

1.15 bitmap

```
Description
This class will allocate a bitmap during IMM_NewFrame, and pass this to its
superclass (Raster). You must free the bitmap yourself.

Methods
None.

Attributes
IMA_Bitmap_Friend [IS.] -- REQUIRED!
   A friend bitmap used when allocating the bitmap (and deciding the depth)

IMA_Bitmap_Bitmap [ISG]
   If you set this tag, the class will treat the value as a pointer to a
   bitmap pointer, and when allocating a bitmap (in IMM_NewFrame) it'll save
   the result in this pointer.

   Example:
      struct BitMap *res = NULL;
      bmp = IM_NewObject("Bitmap", IMA_Bitmap, &res, IMA_Bitmap_Friend, win->RPort ↩
         ->BitMap, TAG_DONE);

      /* ... */

      BltBitMapRastPort(res, ...);
      FreeBitMap(res);

Notes
Is there situations where one cannot provide a friend bitmap?
```

1.16 probe

```
Description
This class can call a hook when certain methods are invoked. The hook follow
normal hook calling conventions, and the message given is the message given to
the method. Also, the result of the hook is used as method result.

Methods
None.
```

```
Attributes
IMA_Probe_NewFrameHook [IS.]
   Call this hook when IMM_NewFrame is invoked on the object. The hook will be
   called before the object calls the superclass.
   Just to be sure, here’s the code:

      if(!data->NewFrameHook || CallHookA(data->NewFrameHook, obj, msg))
         result = DoSuperMethodA(cl, obj, msg);

IMA_Probe_ReceiveDataHook [IS.]
   Call this hook when IMM_ReceiveData is invoked on the object.

IMA_Probe_EndFrameHook [IS.]
   Call this hook when IMM_EndFrame is invoked on the object.

Notes
This class was mainly created for debug purposes. Though those of you who’re
too lazy to create your own subclass might find this class useful.
```