Raster

Paul Manias

COLLABORATORS			
	TITLE : Raster		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Paul Manias	July 26, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Raster

1.1 Object: Raster

```
OBJECT DOCUMENTATION
Name:      RASTER
Version:   0.9 Beta
Date:      November 1997
Author:    Paul Manias
Copyright: DreamWorld Productions, 1996-1997.  All rights reserved.
```

1.2 Object: Raster

```
OBJECT
Name:    Raster
Module:  Screens
Version: 1
Type:    Complex

DESCRIPTION
This object was created to allow easy handling of raster-line based
graphics hardware.  In the Amiga, this means the copper chip.  Because most
graphics boards do not allow for raster based commands, this object is
considered to be "unreliable".  Simply put, if you decide to use this
object then you cannot expect it to work on all machines.  Make sure that
you prepare for this in your code.

A raster list is created by building a chain of commands which are attached
to the main Raster object.  These commands are executed in order, at their
relative points on the display.  Any commands that are not recognised will
be ignored.

EXAMPLE
The following example will attach a raster object to a screen.  Note that
in this case, initialising the Screen object will also initialise the
Raster for us.

   struct RColourList RColourList = {
      ID_RASTCOLOURLIST, 1, NULL, NULL, NULL,
```

```
      000,1,0,Colourlist
   };

   if (Raster = Get(ID_RASTER)) {
      Raster->Command = (struct RHead *)&RColourList1;

      if (Screen = InitTags(NULL,
          TAGS_SCREEN, NULL,
          GSA_Raster,  Raster,
          TAGEND)) {

         Display(Screen);

      Free(Screen);
      }
   Free(Raster);
   }
```

```
ACTIONS
The Raster object supports the following actions:

     Activate() Calculates all the commands in the Raster object.
     Detach()   Detach the Raster object from the Screen.
     Display()  Display the Raster on Screen.
     Free()     Free the object.
     Get()      Get a new Raster structure.
     Hide()     Hide the Raster from the Screen.
     Init()     Initialise the Raster object.

STRUCTURE
The Raster structure consists of the following public fields:

     Command Pointer to the first raster command.
     Flags   Special flags.
     Screen  Pointer to the Screen owner.
```

1.3 Object: Raster

```
FIELD
Name:      Command
Type:      struct RHead *
On Change: Cannot change after initialisation.
Status:    Read/IWrite

DESCRIPTION
This field points to the first Raster command in the list.

The  following  Command  structures are supported as children of the raster
object.   You  need  to  link  each command structure via the Next and Prev
fields to create a chain of commands.  Each command structure begins with a
header that contains the following public fields:

     ID      ID code for the command.
     Next    Next command in the list.
     Prev    Previous command in list.
```

```
     Version Version number for the particular command.

The "arguments" for the command then follow after this header structure.

COMMANDS
This version of the Raster object supports the following commands:

     Colour     Set the value of a colour.
     ColourList Generate a list of colour values.
     Flood      Used to create a special "flood" effect.
     WaitLine   Wait for the beam to reach a set line.
```

1.4 Object: Raster

```
FIELD
Name:      Flags
Type:      LONG
On Change: Cannot change after initialisation.
Status:    Read/Write

DESCRIPTION
This field contains all raster based flags.  They are:

  RSF_DISPLAYED
  This  is  set if the raster is currently displayed on screen.  You cannot
  set this flag yourself - it is handled by Hide() and Display().
```

1.5 Object: Raster

```
FIELD
Name:      Screen
Type:      struct Screen *
On Change: Cannot change after initialisation.
Status:    Read Only.

DESCRIPTION
This field points to the Screen that owns the Raster object.

Note that there is no need for you to set this field for initialisation.

SEE ALSO
Object: Screen
```

1.6 Object: Raster

```
COLOUR <ColNum>,<RRGGBB>

Changes  the  24 bit value of a colour in the screen palette.  This command
does not work on true colour and similar non-palette based screen types.
```

1.7 Object: Raster

```
COLOURLIST <Line>,<Skip>,<ColNum>,<Values>
```

This command allows you to generate the classic coloured lines used by games and demos everywhere. This command is mostly useful for sky/background effects, although you could probably use it for all sorts of things. It does not work on true colour and similar non-palette based screen types.

1.8 Object: Raster

```
FLOOD
```

A special effect that reverses the bitplane modulo, causing the bitplane to repeat itself. This effect is used as a novel way of "fading in" the screen.

1.9 Object: Raster

```
WAITLINE <Line>
```

Waits for the vertical beam to reach the specified screen position. It is perfectly legal to enter numbers that go outside of your screen's vertical limits (ie negative numbers and numbers greater than the screen height), but no more than a value of 10.

Note that the purpose of this command is to specify the screen position at which the next command will be executed. All line values must be specified in lo-res pixels, regardless of your screen resolution.

1.10 Object: Command

```
FIELD
Name:      ID
Type:      WORD
On Change: Cannot change after initialisation.
Status:    Read/Write

DESCRIPTION
This field defines the command type. Valid commnd ID's are listed in the
"graphics/screens.i" file.
```

1.11 Object: Command

```
FIELD
Name:      Next
Type:      struct RHead *
On Change: Cannot change after initialisation.
Status:    Read/Write

DESCRIPTION
This  field  points  to the next command in the chain.  If this is the last
command, this field will be set to NULL.
```

1.12 Object: Command

```
FIELD
Name:      Prev
Type:      struct RHead *
On Change: Cannot change after initialisation.
Status:    Read/Write

DESCRIPTION
This  field  points  to  the previous command in the chain.  If this is the
first command in the list, then this field will be set to NULL.
```

1.13 Object: Command

```
FIELD
Name:      Version
Type:      WORD
On Change: Cannot change after initialisation.
Status:    Read/Write

DESCRIPTION
This  field specifies the version number of the command.  Note that version
numbers start at 1 and go up from there.
```