Screen

Paul Manias

COLLABORATORS			
	TITLE : Screen		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Paul Manias	May 28, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Screen

1.1 Object: Screen

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

CHANGES VERSION 0.9B
Edited:  Screen description.
         Screen->Palette

Added:   Screen->Attrib:BLANKPALETTE.
         Screen->Switch

Removed: Screen->AmtColours
         Screen->ByteWidth
         Screen->LineMod
         Screen->PicHeight
         Screen->Planes
         Screen->PlaneMod
         Screen->PlaneSize
         Screen->ScrType
         Screen->Width

Renamed: ScrHeight, ScrWidth to Height, Width.
         PicX/YOffset to BmpX/YOffset
```

1.2 Object: Screen

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

DESCRIPTION
The Screen represents the values of an area of displayable video memory. Although it is probably the most complex structure, it is fairly simple to initialise. Indeed it is possible to initialise a Screen by passing an empty Screen structure to Init() and accepting all the user defaults (recommended if possible).

For more demanding applications however you may often need to specify a few fields. Before doing so, make sure that you understand how each field operates and what implications setting them may bring. Where possible try to avoid setting field values, as the user default should always be considered as acceptable.

NOTE
The MemPtrX fields can change without notice while your program is running. This is because the memory pointers move around when the screen is hidden or moved from the display. To ensure the screen does not move from video memory you can use the LockVideo() action.

ACTIONS
The Screen object supports the following actions:

CopyStructure()	Copy screen details to another object.
Display()	Display a screen on a monitor or TV.
Free()	Free a Screen from the system.
Get()	Get a new Screen structure.
Hide()	Hide a screen from view.
Init()	Initialise a Screen.
Lock()	Lock a screen from other tasks.
Unlock()	Unlock a locked screen.

FUNCTIONS
Some functions that support the Screen object are:

- LockVideo()
- RefreshScreen()
- RemakeScreen()
- ResetPicture()
- UnlockVideo()

STRUCTURE
The Screen structure consists of the following public fields:

	Attrib	Special Attributes are?
+	Bitmap	Pointer to bitmap structure (for blitting).
	BmpXOffset	Offset of the horizontal axis.
	BmpYOffset	Offset of the vertical axis.
	Height	The height of the visible screen window.
	MemPtr1	Pointer to screen 1
	MemPtr2	Pointer to screen 2 (doubled buffer)
	MemPtr3	Pointer to screen 3 (tripled buffer)
	Palette	Pointer to the screen palette.
	Rasterlist	Pointer to a rasterlist.
	ScreenLink	Pointer to a linked screen.
	ScrMode	What screen mode is it?
	Switch	Set if ready to switch display buffers.
	Task	Task that owns this screen.

```
     Width      The width of the visible screen window.
     XOffset    Hardware co-ordinate for TOS.
     YOffset    Hardware co-ordinate for LOS.
```

1.3 Object: Screen

```
FIELD
Name:        Attrib
Type:        LONG
Inheritance: Some flags can be set by the user.
On Change:   Cannot change after initialisation.
Status:      Read/IWrite

DESCRIPTION
Defines the special attributes for the screen. Currently available flags
are:

BLANKPALETTE
Any screen with 256 colours or less will open with a pre-defined user
palette if Screen->Paltte is NULL. By setting this flag you can dismiss
this and the screen will open with a pure black palette.

BLKBDR
Turns all colours outside of the display window to black. Works on AGA
only.

CENTRE
Centres the screen by calculating the correct offsets for ScrXOffset and
ScrYOffset for any screen mode. The calculated values will over-write any
values already set in these fields.

DBLBUFFER
Allocates an extra screen buffer which is placed in MemPtr2. See the
SwapBuffers() function for more information on double buffering.

HSCROLL
Set if you want to use horizontal picture scrolling.

NOSCRBDR
Allows sprites and other displayable objects to appear outside of the
viewport. Works on AGA only.

SBUFFER
Allocates extra space to allow you to horizontally scroll up to 50 screens
in either X direction.

SPRITES
Set if you intend to use sprites with this screen.

TPLBUFFER
Allocates two extra buffers which are placed in MemPtr2 and MemPtr3. See
the SwapBuffers() for more information on triple buffering.

Note: Never set the DBLBUFFER flag in conjunction with the TPLBUFFER flag.
```

VSCROLL
Set if you want to use vertical picture scrolling.

1.4 Object: Screen

The Bitmap is the display data that shows through onto screen. The Bitmap can be larger than the Screen area, but must never be smaller than the Screen area.

1.5 Object: Screen

FIELDS
Names: BmpXOffset, BmpYOffset
Type: WORD
On Change: RemakeScreen()
Status: Read/Write

DESCRIPTION
These two fields set the offsets for the Bitmap "behind" the Screen. If you want to do any sort of hardware scrolling, you will want to use these values in conjunction with MoveBitmap().

1.6 Object: Screen

FIELDS
Names: MemPtr1, MemPtr2, MemPtr3
Type: APTR
On Change: Cannot change after initialisation.
Status: Read/IWrite
Inheritance: Allocated by Init().

DESCRIPTION
These fields point to the screen display data. They should be NULL if you want Init() to allocate the memory for you (highly recommended). Otherwise it will be assumed that the values are valid pointers to video memory and screen based functions will use them as such.

1.7 Object: Screen

FIELD
Name: Palette
Type: LONG *
On Change: UpdatePalette()
Inheritance: Pre-defined user palette.
Status: Read/Write

DESCRIPTION

Points to the palette for this screen. Your palette array must be represented in 24 bit colours (0x00RRGGBB). The format of the array looks like this:

```
PALETTE, <AmtColours>,
0x00RRGGBB,
...
```

Example:

```
LONG Palette[] = {
  PALETTE, 4,
  0x292894, 0x29f8f8, 0x499249, 0x392423
};
```

If you do not set this field then a user palette will be allocated and placed in this field. If you specify the BLANKPALETTE flag in Screen->Attrib, then you will get a black palette.

NOTE
If the screen type is of CHUNKY15 or better, this field will be ignored as colours are represented in the pixels, not a palette.

SEE ALSO
Bitmap: AmtColours

1.8 Object: Screen

FIELD
Name: Rasterlist
Type: APTR
On Change: Cannot change by direct means after initialisation.
Status: Read/Write
Inheritance: None

DESCRIPTION
Points to a valid rasterlist structure, or NULL. Rasterlists are made up of instructions that are executed as the monitor beam travels down the screen. See InitRasterlist() for more information on rasterlists.

To change the current rasterlist to another, call RemoveRasterList(), write the Rasterlist field then call InitRasterlist().

To update an existing rasterlist, call UpdateRasterlist().

1.9 Object: Screen

FIELD
Name: ScreenLink
Type: struct Screen *
On Change: Cannot change after initialisation.
Status: Read/IWrite

DESCRIPTION
If you want to set up a second screen at a different position in the viewport, or create an extra (double) playfield, point to the next Screen structure here.

1.10 Object: Screen

FIELD
Name: ScrMode
Type: WORD
Inheritance: User default.
On Change: Cannot change after initialisation.
Status: Read/Write

DESCRIPTION
Defines the display mode for the screen. If you do not fill in this field, you will get the user's default resolution.

This field does not allow programmed mode promotion - this matter is handled internally and by user selections in GMSPrefs. You cannot force mode promotion from inside your program. NB: If you require compatibility for NTSC, set ScrHeight to 200 (or accept the user default for ScrHeight) and assume that the user has selected NTSC mode promotion in GMSPrefs.

LORES
Specifies a low resolution screen.

HIRES
Specifies a high resolution screen (1/2 pixels).

SHIRES
Specifies a super-high resolution screen (1/4 pixels).

LACED
Creates an interlaced display (1/2 pixels).

HAM
Sets the Amiga only Hold And Modify mode. The amount of colours you get is dependant on the amount of planes in the screen. Works for planar/ interleaved screens only.

1.11 Object: Screen

FIELDS
Names: Switch
Type: WORD
Status: Read/Write

DESCRIPTION

```
SEE ALSO
Screens: WaitSwitch()
Theory:  Display Buffering
```

1.12 Object: Screen

```
FIELDS
Names:       XOffset, YOffset
Type:        WORD
On Change:   RemakeScreen()
Status:      Read/Write
Inheritance: None.

DESCRIPTION
Specifies the hardware offset for the screen. These two values are added
to the user's screen offset in GMSPrefs. A setting of 0,0 should be
sufficient, unless you are going to create an extra large screen (eg
overscan). Negative values are allowed.
```

1.13 Object: Screen

```
FIELDS
Names:       Task
Type:        struct DPKTask *
On Change:   Cannot change after initialisation.
Status:      Read Only.
Inheritance: Task of initialisation.

DESCRIPTION
Points to a DPKTask structure that identifies the task owning this screen
structure.
```

1.14 Object: Screen

```
FIELDS
Names:       Width, Height
On Change:   RemakeScreen()
Status:      Read/Write
Inheritance: User default.

DESCRIPTION
Defines the screen height and width. This is the "viewport" that the
picture data is displayed through. The width of the screen must be
divisible by 16 for ILBM and Planar modes.
```

```
SEE ALSO
Bitmap: Width
        Height
```