Screen

COLLABORATORS			
	TITLE : Screen		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 24, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Screen

1.1 Screen

```
                PureBasic - Screen guide

  'Screen' signifie 'écran' et c'est une particularité de l'AmigaOS par
  rapport aux autres systèmes d'exploitation. En effet, chaque application
  peut avoir son propre écran avec ses propres caractériques (nombre de
  couleurs, taille, mode...) en fonction de ses besoins. Que ce soit pour
  un jeu ou une application, les écrans utilisés restent les même. Ce sont
  eux la base du système d'affichage sur Amiga.

Commandes disponibles:

              CloseScreen

              CreateDualPlayField

              FindScreen

              FindFrontScreen

              FlashScreen

              HideScreen

              InitScreen

              OpenScreen

              RemoveDualPlayField

              ScreenBarHeight

              ScreenDepth

              ScreenFontHeight

              ScreenHeight
```

```
                ScreenID

                ScreenMouseX

                ScreenMouseY

                ScreenRastPort

                ScreenViewPort

                ScreenWidth

                ShowScreen

                UseScreen
                 Exemple:

   Open screens
```

1.2 createdualplayfield

```
  Syntaxe
CreateDualPlayField(BitMapID)

  Résumé
Crée un deuxième 'PlayField' sur l'écran courant avec le 'BitMapID'
spécifié. Du coup, l'écran est divisé en 2 parties totalement
indépendante l'une de l'autre. Une se trouve à l'avant plan de
l'écran, l'autre à l'arrière plan (dans le fond). Chacune des 2
parties a ses propres 'BitMaps' et ses propres couleurs. La couleur
0 est considérée comme transparente, donc l'arrière plan n'est visible
que si l'avant plan contient un peu de couleur 0. L'avantage d'un
dualplayfield est multiple: premierement on double le nombre de
couleur affiché simultatement à l'écran (chaque plan a ses propres
couleurs); plus besoin se sauvegarder le fond d'écran quand on déplace
des sprites car le fond n'est jamais detruit. Cependant, le nombre
de couleurs de l'écran ne peut dépasser 8 sur les Amiga ECS/OCS
et 16 sur les Amigas AGA (donc, 8 ou 16 couleurs par plan, ce
qui fait respectivement 16 et 32 couleurs simultanément affichées
à l'écran).

Note: Pour changer le BitMap d'arrière plan, vous devez utiliser
      la commande 'ShowBackBitMap()'.

Vous devez appeler la commande 'RemoveDualPlayFiled()' avant de
fermer un écran ! Cette commande n'est pas appelée automatiquement.
```

1.3 removedualplayfield

```
  SYNTAX
RemoveDualPlayField()
```

```
  COMMAND
Enlève le 'DualPlayfield' de l'écran courant. Cet écran doit être un
écran 'DualPlayfield' !

Vous devez appeler la commande 'RemoveDualPlayFiled()' avant de
fermer un écran ! Cette commande n'est pas appelée automatiquement.
```

1.4 findscreen

```
  Syntaxe
ScreenID.l = FindScreen(#Screen, ScreenName$)

  Résumé
Trouve un écran publique grâce à son nom et l'attache à l'identifiant
donné. Si la valeur retournée 'ScreenID' est NULLE, alors l'écran est
introuvable.

#Screen: Identifiant numérique de l'écran trouvé.

ScreenName$: Nom de l'écran à trouver. Si vous spécicifiez un nom nul ("")
             alors l'écran publique par défaut sera retourné. Pour capturer
             l'écran du 'Workbench' il suffit de mettre "Workbench" pour
             le nom de l'écran.
```

1.5 findfrontscreen

```
  Syntaxe
ScreenID.l = FindFrontScreen(#Screen)

  Résumé
Trouve l'écran se trouvant à l'avant plan de l'affichage et l'attache
à l'identifiant donné. Si la valeur retournée 'ScreenID' est NULLE, alors
aucun écran est ouvert sur le système.

#Screen = Identifiant numérique de l'écran trouvé.
```

1.6 openscreen

```
  Syntaxe
ScreenID.l = OpenScreen(#Screen, Width, Height, Depth, TagListID)

  Résumé
Ouvre un nouvel écran en accord avec les paramètres donnés. Si la valeur
retournée 'ScreenID' est NULLE, alors l'écran n'a pas été ouvert. Un fois
ouvert, il devient l'écran courant.

#Screen: Identifiant numérique pour l'écran qui vient d'être ouvert.

Width: Largeur de l'écran.
```

Height: Hauteur de l'écran.

Depth: Profondeur de l'écran (nombre de couleurs)

TagListID: Identifiant de la taglist à utiliser. Pour obtenir cet identfiant ↩ facilement,
il est conseillé d'utiliser la fonction TagListID() .

Tags disponibles

#SA_Left
#SA_Top
#SA_Width
#SA_Height
The defaults for the #SA_Left, #SA_Top, #SA_Width, and #SA_Height tags end up being a bit complex. If none of these tags are specified, and no NewScreen structure is used, then the left/top/width/height correctly match the display clip of your screen (see #SA_DClip and #SA_Overscan).

The difficulty comes with overscanned screens, because the normal value of #SA_Left or #SA_Top for such a screen may be non-zero. If a NewScreen structure is supplied, then the left/top/width/height come originally from there. If no NewScreen structure is supplied, but a non-default #SA_Width (#SA_Height) is specified, then #SA_Left (#SA_Top) defaults to zero instead. In these cases, the left and top edge may not be what you want.

If you need to specify explicit width or height, or supply a NewScreen, you must supply correct values for #SA_Left and #SA_Top. The correct normal values are the display clip rectangle's MinX and MinY values respectively. If you are using the #SA_DClip tag, then you already have a rectangle to consult for these values. If you are using #SA_Overscan to get one of the standard overscan types, you may use QueryOverscan() to get a rectangle for that overscan type.

#SA_Depth (defaults to 1)
#SA_DetailPen (defaults to 0)
#SA_BlockPen (defaults to 1)
#SA_Title (defaults to NULL)
#SA_Font (defaults to NULL, meaning user's preferred monospace font)
#SA_BitMap (whose existence also implies CUSTOMBITMAP).

Several tags are Booleans, which means that depending on whether their corresponding ti_Data field is zero (FALSE) or non-zero (TRUE), they specify Boolean attributes. The ones corresponding to Boolean flags in the NewScreen.Type field are:

#SA_ShowTitle (defaults to TRUE)
#SA_Behind (equiv. to SCREENBEHIND) (defaults to FALSE)
#SA_Quiet (equiv. to SCREENQUIET) (defaults to FALSE)

The following tags provide extended information to Intuition when creating a screen:

#SA_Type: ti_Data corresponds to the SCREENTYPE bits of the NewScreen.Type field. This should be one of PUBLICSCREEN or CUSTOMSCREEN. The other bits of the NewScreen.Type field must be set with the appropriate tags (#SA_Behind, #SA_Quiet, etc.)

#SA_DisplayID: ti_Data is a 32-bit extended display mode ID, as defined in the <graphics/modeid.h> include file (V39 and up) or in <graphics/displayinfo.h> (V37 and V38).

#SA_Overscan: ti_Data contains a defined constant specifying one of the system standard overscan dimensions appropriate for the display mode of the screen. Used with the Width and Height dimensions STDSCREENWIDTH and STDSCREEN, this makes it trivial to open an overscanned or standard dimension screen. You may also hand-pick your various dimensions for overscanned or other screens, by specifying screen position and dimensions explicitly, and by using #SA_DClip to explicitly specify an overscanned DisplayClip region.

The values for ti_Data of this tag are as follows:

OSCAN_TEXT - Text Overscan region. A region which is completely on screen and readable ("text safe"). A preferences data setting, this is backward equivalent with the old MoreRows, and specifies the DisplayClip and default dimensions of the Workbench screen. This is the default.

OSCAN_STANDARD - Also a preferences setting, this specifies a rectangle whose edges are "just out of view." This yields the most efficient position and dimensions of on-monitor presentations, such as games and artwork.

OSCAN_MAX - This is the largest rectangular region that the graphics library can handle "comfortably" for a given mode. Screens can smoothly scroll (hardware pan) within this region, and any DisplayClip or Screen region within this rectangle is also legal. It is not a preferences item, but reflects the limits of the graphics hardware and software.

OSCAN_VIDEO - This is the largest region that the graphics library can display, comfortable or not. There is no guarantee that all smaller rectangles are valid. This region is typically out of sight on any monitor or TV, but provides our best shot at "edge-to-edge" video generation.

Remember, using overscan drastically effects memory use and chip memory bandwidth. Always use the smallest (standard) overscan region that works for your application.

#SA_DClip: ti_Data is a pointer to a rectangle which explicitly defines a DisplayClip region for this screen. See QueryOverscan() for the role of the DisplayClip region.

Except for overscan display screens, this parameter is unnecessary, and specifying a standard value using #SA_Overscan is normally an easier way to get overscan.

#SA_AutoScroll: this is a Boolean tag item, which specifies that this screens is to scroll automatically when the mouse pointer reaches the edge of the screen. The operation of this requires that the screen dimensions be larger than its DisplayClip region.

#SA_PubName: If this field is present (and ti_Data is non-NULL), it means that the screen is a public screen, and that the public screen name string is pointed to by ti_Data. Public screens are opened in "PRIVATE" mode and must be made public using PubScreenStatus(screen, 0).

#SA_Pens: The ti_Data field (if non-NULL) points to a UWORD array of pen specification, as defined for struct DrawInfo. This array will be used to initialize the screen's DrawInfo.dri_Pens array.

#SA_Pens is also used to decide that a screen is ready to support the full-blown "new look" graphics. If you want the 3D embossed look, you must provide this tag, and the ti_Data value cannot be NULL. If it points to a "minimal" array, containing just the terminator ~0, you can specify "new look" without providing any values for the pen array.

The way the DrawInfo pens are determined is Intuition picks a default pen-array. Then, any pens you supply with #SA_Pens override the defaults, up until the ~0 in your array.

If the screen is monochrome or old-look, the default will be the standard two-color pens.

If the screen is two or more planes deep, the default will be the standard four-color pens, which now include the new-look menu colors.

If the screen has the #SA_LikeWorkbench property, the default will be the user's preferred pen-array, changeable through preferences.

The following two tag items specify the task and signal to be issued to notify when the last "visitor" window closes on a public screen. This support is to assist envisioned public screen manager programs.

#SA_PubTask: Task to be signalled. If absent (and #SA_PubSig is valid), use the task which called OpenScreen() or OpenScreenTagList()).

#SA_PubSig: Data is a UBYTE signal number (not flag) used to notify a task when the last visitor window closes on a public screen.

```
#SA_Colors: ti_Data points to an array of ColorSpec structures
    (terminated with ColorIndex = -1) which specify initial
    values of the screen's color palette.

#SA_FullPalette: this is a Boolean attribute.  Prior to V36, there
    were just 7 RGB color values that Intuition maintained
    in its user preferences (playfield colors 0-3, and colors
    17-19 for the sprite).  When opening a screen, the color
    map for the screens viewport is first initialized by
    graphics (graphics.library/GetColorMap()) then these
    seven values are overridden to take the preferences values.

    In V36, Intuition maintains a full set of 32 preferences colors.
    If you specify TRUE for #SA_FullPalette, Intuition will
    override ALL color map entries with its full suite of
    preferred colors.  (Defaults to FALSE).

#SA_ErrorCode: ti_Data points to a ULONG in which Intuition will
    stick an extended error code if OpenScreen[TagList]() fails.
    Values are of this include 0, for success, and:
    OSERR_NOMONITOR - monitor for display mode not available.
    OSERR_NOCHIPS   - you need newer custom chips for display mode.
    OSERR_NOMEM     - couldn't get normal memory
    OSERR_NOCHIPMEM - couldn't get chip memory
    OSERR_PUBNOTUNIQUE  - public screen name already used
    OSERR_UNKNOWNMODE   - don't recognize display mode requested
    OSERR_TOODEEP   - screen too deep to be displayed on
              this hardware (V39)
    OSERR_ATTACHFAIL    - An illegal attachment of screens was
              requested (V39)

    NOTE: These values are not the same as some similar return
    values defined in graphics.library/ModeNotAvailable().

#SA_SysFont: ti_Data selects one of the system standard fonts
    specified in preferences.  This tag item overrides the
    NewScreen.Font field and the #SA_Font tag item.

    Values recognized in ti_Data at present are:
    0 - old DefaultFont, fixed-width, the default.
    1 - Workbench screen preferred font.  You have to
        be very font sensitive to handle a proportional or
        larger than traditional screen font.

    NOTE WELL: if you select sysfont 1, windows opened on
    your screen will not inherit the screen font, but rather
    the window RastPort will be initialized to the old-style
    DefaultFont (sysfont 0).

Attached screen tags:  V39 supports attached screens, where
    one or more child screens can be associated with a parent
    screen.  Attached screens depth-arrange as a group, and
    always remain adjacent depth-wise.  Independent
    depth-arrangement of child screens is possible through
    the V39 ScreenDepth() call.  If a child screen is
    made non-draggable through {#SA_Draggable, FALSE}, then
    it will drag exclusively with the parent.  Normal child
```

screens drag independently of the parent, but are pulled down when the parent is. Use the #SA_Parent, #SA_FrontChild, and #SA_BackChild tags to attach screens.

#SA_Parent: If you wish to attach this screen to an already-open parent screen, use this tag and set ti_Data to point to the parent screen. See also #SA_FrontChild and #SA_BackChild. (V39).

#SA_FrontChild: If you wish to attach an already-open child screen to this screen, set ti_Data to point to the child screen. The child screen will come to the front of the family defined by the parent screen you are opening. See also #SA_Parent and #SA_BackChild. (V39)

#SA_BackChild: If you wish to attach an already-open child screen to this screen, set ti_Data to point to the child screen. The child screen will go to the back of the family defined by the parent screen you are opening. See also #SA_Parent and #SA_FrontChild. (V39)

#SA_BackFill: ti_Data is a pointer to a backfill hook for the screen's Layer_Info. (see layers.library/InstallLayerInfoHook()). (V39).

#SA_Draggable: ti_Data is a boolean. Set to FALSE if you wish your screen to be non-draggable. This tag should be used very sparingly!. Defaults to TRUE. For child screens (see #SA_Parent, #SA_FrontChild, and #SA_BackChild) this tag has a slightly different meaning: non-draggable child screens are non-draggable with respect to their parent, meaning they always drag exactly with the parent, as opposed to having relative freedom. Also see ScreenPosition(). (V39)

#SA_Exclusive: ti_Data is a boolean. Set to TRUE if you never want your screen to share the display with another screen. This means that your screen can't be pulled down, and will not appear behind other screens that are pulled down. Your screen may still be depth arranged, though. Use this tag sparingly! Defaults to FALSE. Starting with V40, attached screens may be #SA_Exclusive. Setting #SA_Exclusive for each screen will produce an exclusive family. (V39).

#SA_SharePens: For those pens in the screen's DrawInfo->dri_Pens, Intuition obtains them in shared mode (see graphics.library/ObtainPen()). For compatibility, Intuition obtains the other pens of a public screen as PENF_EXCLUSIVE. Screens that wish to manage the pens themselves should generally set this tag to TRUE. This instructs Intuition to leave the other pens unallocated. Defaults to FALSE. (V39).

#SA_Colors32: Tag to set the screen's initial palette colors at 32 bits-per-gun. ti_Data is a pointer to a table to be passed to the graphics.library/LoadRGB32() Résumé. This format supports both runs of color registers and sparse

```
    registers.  See the autodoc for that Résumé for full
    details.  Any color set here has precedence over the same
    register set by #SA_Colors.  (V39).

#SA_Interleaved:  ti_Data is a boolean.  Set to TRUE to
    request an interleaved bitmap for your screen.  Defaults to
    FALSE.  If the system cannot allocate an interleaved bitmap for
    you, it will attempt to allocate a non-interleaved one (V39).

#SA_VideoControl:  ti_Data points to a taglist that will be
    passed to VideoControl() after your screen is open.  You
    might use this to turn on border-sprites, for example.
    (V39).

#SA_ColorMapEntries:  ti_Data is the number of entries that
    you wish Intuition to allocate for this screen's ColorMap.
    While Intuition allocates a suitable number for ordinary
    use, certain graphics.library features require a
    ColorMap which is larger than default.  (The default value is
    1<<depth, but not less than 32).  (V39)

#SA_LikeWorkbench:  ti_Data is boolean.  Set to TRUE to get
    a screen just like the Workbench screen.  This is the
    best way to inherit all the characteristics of the
    Workbench, including depth, colors, pen-array, screen mode,
    etc.  Individual attributes can be overridden through the
    use of tags.  (#SA_LikeWorkbench itself overrides things
    specified in the NewScreen structure).  Attention
    should be paid to hidden assumptions when doing this.  For
    example, setting the depth to two makes assumptions about
    the pen values in the DrawInfo pens.  Note that this
    tag requests that Intuition ATTEMPT to open the screen
    to match the Workbench.  There are fallbacks in case
    that fails, so it is not correct to make enquiries about
    the Workbench screen then make strong assumptions about
    what you're going to get.  (Defaults to FALSE).  (V39)

#SA_MinimizeISG:  ti_Data is boolean.  For compatibility,
    Intuition always ensures that the inter-screen gap is at
    least three non-interlaced lines.  If your application
    would look best with the smallest possible inter-screen
    gap, set ti_Data to TRUE.  If you use the new graphics
    VideoControl() VC_NoColorPaletteLoad tag for your screen's
    ViewPort, you should also set this tag.  (V40)
```

1.7 screenmousex

```
  Syntaxe
x.w = ScreenMouseX()

  Résumé
Retourne la position de la souris par rapport au coté gauche de l'écran courant.
```

1.8 screenmousey

```
  Syntaxe
y.w = ScreenMouseY()

  Résumé
Retourne la position de la souris par rapport au haut de l'écran courant.
```

1.9 screendepth

```
  Syntaxe
Result.w = ScreenDepth()

  Résumé
Retourne la profondeur de l'écran courant.
```

1.10 screenwidth

```
  Syntaxe
width.w = ScreenWidth()

  Résumé
Retourne la largeur de l'écran courant.
```

1.11 screenheight

```
  Syntaxe
height.w = ScreenHeight()

  Résumé
Retourne la hauteur de l'écran courant.
```

1.12 showscreen

```
  Syntaxe
ShowScreen()

  STATEMENT
Fait passer l'écran courant à l'avant plan.
```

1.13 hidescreen

```
  Syntaxe
HideScreen()

  STATEMENT
Fait passer l'écran courant à l'arrière plan.
```

1.14 usescreen

```
  Syntaxe
UseScreen(#Screen)

  STATEMENT
Change l'écran courant par l'écran spécifié.

#Screen: Identifiant numérique de l'écran.
```

1.15 screenviewport

```
  Syntaxe
Result.l = ScreenViewPort()

  Résumé
Retourne la valeur du 'ViewPort' de l'écran courant. Cette fonction
a été écrite uniquement pour les programmeurs expérimentés.
```

1.16 screenbarheight

```
  Syntaxe
Result.b = ScreenBarHeight()

  Résumé
Retourne la hauteur de la barre de menu de l'écran courant. Utile pour ouvrir
les fenêtre juste en dessous, par exemple.
```

1.17 screenfontheight

```
  Syntaxe
Result.b = ScreenFontHeight()

  Résumé
Retourne la hauteur de la police par défaut de l'écran courant.
```

1.18 closescreen

```
  Syntaxe
CloseScreen(#Screen)

  STATEMENT
Ferme l'écran spécifié.

#Screen: Identifiant numérique de l'écran à fermer.
```

1.19 initscreen

```
  Syntaxe
Résultat = InitScreen(#NumScreenMax)

  Résumé
Initialise l'environnement nécessaire à la gestion des futurs écrans.
Vous devez appeler cette fonction avant les autres fonctions de cette
bibliothèque. Si le 'Résultat' est NUL, alors l'environnement n'a pu
être initialisé.

#NumScreenMax: Nombre maximal d'écrans à gérer ultérieurement.
```

1.20 screenid

```
  Syntaxe
ScreenID.l = ScreenID()

  Résumé
Retourne l'identification de l'écran, nécessaire pour quelques fonctions
des autres bibliothèques PureBasic comme  Fade() ,  GetScreenPalette ...
```

1.21 screenrastport

```
  Syntaxe
RastPort.l = ScreenRastPort()

  Résumé
Retourne la valeur du 'RastPort' de l'écran courant, principalement utilisé
par la fonction  DrawingOutput()  de la bibliothèque '2D Drawing'.


  Exemple

DrawingOutput(ScreenRastPort()) ; Defini l'écran courant comme sortie graphique  ↩
   pour les
                                ; fonctions de dessin 2D.

BoxFill(10, 10, 100, 100)       ; Dessine une boite sur l'écran courant.
```

1.22 flashscreen

Syntaxe
FlashScreen()

Résumé
Emet un flash sur l'écran courant.