

2D

COLLABORATORS

	<i>TITLE :</i> 2D		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		January 23, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	2D	1
1.1	2D Drawing library	1
1.2	boxfill	2
1.3	backcolour	2
1.4	cursx	2
1.5	cursy	2
1.6	frontcolour	2
1.7	locate	3
1.8	drawingfont	3
1.9	circle	3
1.10	cls	3
1.11	copybitmap	4
1.12	drawingmode	4
1.13	drawingoutput	5
1.14	drawingrastport	5
1.15	ellipse	5
1.16	line	6
1.17	obtainbestpen	6
1.18	plot	6
1.19	point	7
1.20	printtext	7
1.21	textstyle	7
1.22	releasepen	8
1.23	textlength	8

Chapter 1

2D

1.1 2D Drawing library

PureBasic - 2D Drawing library V1.00

La bibliothèque 2D du PureBasic vous permet de réaliser des opérations graphiques standards comme tracer une ligne, un cercle, un rectangle, afficher un texte, etc... Vous devez au préalable indiquer où les graphiques doivent être rendus (sur une fenêtre, un bitmap ou un écran) grâce à la commande `DrawingOutput()`

Commandes disponibles:

- BackColor
- BoxFill
- Circle
- Cls
- CopyBitmap
- CursX
- CursY
- DrawingFont
- DrawingMode
- DrawingOutput
- DrawingRastPort
- Ellipse
- FrontColour
- Locate
- Line
- ObtainBestPen
- Plot
- Point
- PrintText
- ReleasePen
- TextLength
- TextStyle

Example

2D Drawing demo

1.2 boxfill

Syntaxe

```
BoxFill(x, y, Width, Height)
```

Résumé

Trace un rectangle plein de la couleur courante (que vous pouvez changer grace à la commande `FrontColour()`) aux coordonnées `x,y` et de taille spécifié par `'Width'` et `'Height'`

`x,y`: Position du coin supérieur gauche du rectangle

`Width`: Largeur du rectangle

`Height`: Hauteur du rectangle

1.3 backcolour

Syntaxe

```
BackColour(Colour)
```

Résumé

Change la couleur du fond par la nouvelle couleur `'Colour'`.

`Colour`: numéro de la couleur

1.4 cursx

Syntaxe

```
PositionX.w = CursX()
```

Résumé

Retourne la position en `'x'` (en pixels) du curseur de texte dans l'environnement graphique courant.

1.5 cursy

Syntaxe

```
PositionY.w = CursY()
```

Résumé

Retourne la position en `'y'` (en pixels) du curseur de texte dans l'environnement graphique courant.

1.6 frontcolour

Syntaxe

`FrontColour(Colour)`

Résumé

Change la couleur par défaut pour utilisée par les fonctions graphiques de cette bibliothèque (Afficher un texte, tracer une ligne...)

Colour: numéro de la couleur

1.7 locate

Syntaxe

`Locate(x,y)`

Résumé

Change la position actuelle du curseur de texte. Le prochain texte affiché par `PrintText()` commencera à cette position.

x,y: nouvelles coordonnées du curseur.

1.8 drawingfont

Syntaxe

`DrawingFont(FontID)`

Résumé

Change la police courante pour l'affichage du texte. Toutes les prochaines opération d'affichage de texte utiliseront cette police de caractère.

FontID: Identifiant valide d'une police de caractère. Vous pouvez obtenir cette valeur en utilisant la fonction `FontID()` de la bibliothèque 'Font'.

1.9 circle

Syntaxe

`Circle(x, y, Radius)`

Résumé

Trace un cercle de la couleur courante.

x,y: coordonnées du centre du cercle.

Radius: longueur (en pixels) du rayon du cercle.

1.10 cls

Syntaxe

Cls(Colour)

Résumé

Remplit l'environnement graphique avec la couleur donnée.

Colour: numero de la couleur à utiliser.

1.11 copybitmap

Syntaxe

CopyBitMap(BitMapID, SourceX, SourceY, DestX, DestY, Width, Height)

Résumé

Copie le contenu ou une partie du bitmap passé en paramètre dans l'environnement graphique courant.

BitMapID: Identifiant du BitMap à copier. Vous pouvez utiliser la commande BitMapID() pour récupérer facilement cet identifiant.

SourceX, SourceY: coordonnées servant à définir à quel endroit commence la copie à l'intérieur du BitMap.

DestX, DestY: coordonnées de destination (dans l'environnement graphique courant) à partir desquelles doit commencer la copie.

Width: Largeur de la copie

Height: Hauteur de la copie

1.12 drawingmode

Syntaxe

DrawingMode(Mode)

Résumé

Change le mode d'affichage du texte et des opérations graphiques.

Mode: valeur tirée parmi les suivantes:

#JAM1 = 0 : Pour le texte seulement. Laisse le fond de la police de caractère transparent

#JAM2 = 1 : Pour le texte seulement. Affiche le texte avec la couleur du fond courante (voir BackColour()). ↔

#COMPLEMENT = 2 : Affiche tous les graphiques (lignes, rectangles,...) en mode 'XOR', c'est à dire inversé par rapport à la couleur du fond. ↔

```
#INVERSVID = 4 : Pour le texte seulement. A utiliser en conjonction avec # ↵
JAM2
pour inverser la couleur du fond avec la couleur de la ↵
    police
de caractère.
```

1.13 drawingoutput

Syntaxe

```
DrawingOutput (RastPort)
```

Résumé

Permet de choisir sur quel objet les opérations graphiques doivent être effectuées. Pour récupérer la valeur de ce 'RastPort', vous pouvez utiliser une de ces trois commandes, suivant que vous voulez afficher les graphiques dans une fenêtre, un bitmap ou un écran:

```
WindowRastPort ()
BitmapRastPort ()
ScreenRastPort ()
```

Exemple:

```
DrawingRastPort (WindowRastPort ()) ; Toute les opérations graphiques seront
; effectuées sur la fenêtre courante.
```

1.14 drawingrastport

Syntaxe

```
*RastPort = DrawingRastPort ()
```

Résumé

Retourne la valeur du 'RastPort' de l'environnement graphique courant. Cette fonction n'est utile que pour les programmeurs expérimentés.

1.15 ellipse

Syntaxe

```
Ellipse(x, y, RadiusX, RadiusY)
```

Résumé

Affiche une ellipse de la couleur courante, à la position donnée.

x,y: coordonnées du centre de l'ellipse

RadiusX: longueur du rayon en 'x' de l'ellipse

RadiusY: longueur du rayon en 'y' de l'ellipse

1.16 line

Syntaxe

Line(x, y, Width, Height)

Résumé

Affiche une ligne de la couleur courante (voir FrontColour()).

x,y: coordonnées du point gauche de la ligne.

Width: largeur de la ligne

Height: hauteur de la ligne

1.17 obtainbestpen

Syntaxe

Couleur.w = ObtainBestPen(r, g, b, precision)

Résumé

Retourne le numéro de la couleur qui ressemble le plus aux paramètres que vous avez passés. Si la couleur n'est pas disponible, et si l'écran a des couleurs dites 'libres' alors cette commande alloue une couleur et la bloque pour que personne ne puisse la changer ultérieurement.

r,g,b: Valeurs rouge/vert/bleu qui caractérise la couleur que vous voulez obtenir sur l'écran.

precision: Paramètre pour informer la commande comment elle doit trouver la couleur:

```
Exact = -1 (#PRECISION_EXACT)
Image = 0 (#PRECISION_IMAGE)
Icon = 16 (#PRECISION_ICON)
Gui = 32 (#PRECISION_GUI)
```

Note: Vous devez libérer toutes les couleurs préalablement bloquées avant de quitter votre programme grâce à la commande ReleasePen() !

1.18 plot

Syntaxe

Plot(x,y)

Résumé

Trace un point de la couleur courante (que vous pouvez changer grâce à la commande FrontColour()) aux coordonnées x,y.

`x,y`: Position en pixel du point.

1.19 point

Syntaxe

```
Colour.w = Point(x, y)
```

Résumé

Retourne le numéro de la couleur qui se trouve aux coordonnées spécifiées.

`x,y`: coordonnées du point.

1.20 printtext

Syntaxe

```
PrintText(String$)
```

Résumé

Affiche le texte donné sur la sortie graphique courante (définie par `DrawingOutput()`) à la position définie par `Locate()`. La couleur utilisée est la couleur d'avant plan (`FrontColour()`). Vous pouvez utiliser la commande `TextStyle()` pour changer le style du texte (gras, italique ou souligné) et la commande `DrawingMode()` pour changer la manière de rendre le texte.

1.21 textstyle

Syntaxe

```
TextStyle(Style)
```

Résumé

Change le style du texte pour les prochains appels de la commande `PrintText()`.

Style: combinaison valide des valeurs suivantes:

- 0: Aucun style
- 1: Style 'souligné'
- 2: Style 'gras'
- 4: Style 'italique'

Par exemple, une valeur de '6' représente Gras+Italique donc les prochains texte seront affiché en gras et en italique.

1.22 releasepen

Syntaxe

```
ReleasePen(Colour)
```

Résumé

Change le status de la couleur de 'bloqué' vers 'libre', c'est à dire qu'un autre programme pourra utiliser cette couleur comme bon lui semble (et meme changer sa valeur). Vous devez libérer toutes les couleurs préalablement bloquées avec `ObtainBestPen()` avant de quitter votre programme !

Colour: numéro de la couleur à libérer.

1.23 textlength

Syntaxe

```
Longueur.w = TextLength(Text$)
```

Résumé

Retourne la longueur en pixels du texte passé en paramètre tel qu'il apparaîtrait dans l'environnement graphique courant (en utilisant la police de caractère, etc..). Cette fonction retourne la bonne valeur pour n'importe quel type de polices, qu'elle soient proportionnelles ou non.

Text\$: Chaîne de caractère à tester.