# Sprite

**COLLABORATORS**

| | *TITLE* :<br><br>Sprite | |
|---|---|---|

| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
|---|---|---|---|
| WRITTEN BY | | January 23, 2025 | |

**REVISION HISTORY**

| NUMBER | DATE | DESCRIPTION | NAME |
|---|---|---|---|
| | | | |

# Contents

# Chapter 1

# Sprite

## 1.1 Sprite

```
PureBasic - Sprite library

  Les 'Sprites' sont bien connus des joueurs inconditionnels sur
  ordinateurs. Ce sont ces petites portions de graphismes qui bougent
  à l'écran provocant un effet d'animation de celui-ci. Les 'Sprites'
  on en effet la particularité de se déplacer au dessus des graphismes
  sans les détruire. En PureBasic, ils sont gérés intégralement par
  le 'Blitter', un composant majeur de l'Amiga, qui permet justement
  de déplacer très rapidement des données graphiques. Cette bibliothèque
  est très optimisée et a un atout non négligeable: elle travaille
  en parallèle avec le CPU de l'Amiga. Concrètement, quand le Blitter
  travaille, le CPU peut faire autre chose comme le calcul des futurs
  déplacements, l'intelligence artificielle, etc.. Ces commandes sont
  toutes compatibles à 100% avec l'AmigaOS.

Commandes disponibles:

   AddBlockSprite
   AddBufferedSprite
   AddSprite
   CreateSpriteBuffer
   FreeSprite
   FreeSpriteBuffer
   InitSprite
   LoadSprite
   ResetSpriteServer
   RestoreBackground
   SpriteDepth
   SpriteHeight
   SpriteWidth
   StartSpriteServer
   StopSpriteServer
   UseSpriteBuffer
   WaitSpriteServer

Exemple:
```

```
    Sprite demo
```

## 1.2 addblocksprite

```
  Syntaxe
AddBlockSprite(#Sprite, x, y)

  Résumé
Affiche le sprite à la position spécifiée dans le 'SpriteBuffer'
courant (voir 'CreateSpriteBuffer'). Cette commande est la
manière la plus rapide d'afficher un sprite à l'écran mais
elle a quelques limitations:

  + La largeur du sprite doit être multiple de 16 (ex: 16, 32, 48, ...)
  + Sa position en x doit être aussi multiple de 16
  + Le sprite n'a pas de couleur transparente.


Note: Si un autre sprite est en train d'être affiché, cette commande
ajoute votre sprite à la liste d'attente du 'SpriteServer' et vous
rend la main. Pour être sûr que votre Sprite est réellement affiché,
vous devez utiliser la commande 'WaitSpriteServer'. Veuillez vous
reporter à la section 'StartSpriteServer()' pour plus d'informations
à propos de la gestion asynchrone des sprites.
```

## 1.3 addbufferedsprite

```
  Syntaxe
AddBufferedSprite(#Sprite, x, y)

  Résumé
Affiche le sprite à la position spécifiée dans le 'SpriteBuffer'
courant (voir 'CreateSpriteBuffer'). Avant d'afficher réellement
le sprite, le fond d'écran qui sera détruit par ce sprite est
sauvegardé dans le 'SpriteBuffer' courant. Ce fond pourra être
restauré grâce à la commande 'RestoreBackground()'. La couleur 0
de ce sprite est considérée comme transparente.


Note: Si un autre sprite est en train d'être affiché, cette commande
ajoute votre sprite à la liste d'attente du 'SpriteServer' et vous
rend la main. Pour être sûr que votre Sprite est réellement affiché,
vous devez utiliser la commande 'WaitSpriteServer'. Veuillez vous
reporter à la section 'StartSpriteServer()' pour plus d'informations
à propos de la gestion asynchrone des sprites.
```

## 1.4 addsprite

```
  Syntaxe
AddSprite(#Sprite, x, y)

  Résumé
Affiche le sprite à la position spécifiée dans le 'SpriteBuffer'
courant (voir 'CreateSpriteBuffer'). Le fond sera irrémediablement
détruit. Si vous souhaitez conserver le fond, il faut utiliser
la commande 'AddBufferedSprite'. La couleur 0 de ce sprite est
considérée comme transparente.


Note: Si un autre sprite est en train d'être affiché, cette commande
ajoute votre sprite à la liste d'attente du 'SpriteServer' et vous
rend la main. Pour être sûr que votre Sprite est réellement affiché,
vous devez utiliser la commande 'WaitSpriteServer'. Veuillez vous
reporter à la section 'StartSpriteServer()' pour plus d'informations
à propos de la gestion asynchrone des sprites.
```

## 1.5 createspritebuffer

```
  Syntaxe
CreateSpriteBuffer(#SpriteBuffer, Size, BitMapID)

  Résumé
Crée et initialise un nouveau 'SpriteBuffer', qui sera en fait une zone
graphique sur laquelle les sprites seront affichés. 'BitMapID' est
l'identifiant du bitmap qui servira à l'affichage des sprite. Si vous
voulez sauvegarder le fond d'écran lors du déplacement des sprites,
vous devez alors spécifier une taille ('Size') de buffer dans lequel
sera stocké le fond d'écran. Une taille de 16000 octets sera suffisante
dans la majorité des cas. Voici la formule pour calculer la taille
nécessaire avec exactitude:

  Taille = (BitMapDepth * (SpriteWidth*SpriteHeight)*NumberOfSpriteDisplayed) /  ↩
     8 + 1000

Ce nouveau SpriteBuffer devient le SpriteBuffer courant.
```

## 1.6 freesprite

```
  Syntaxe
FreeSprite(#Sprite)

  Résumé
Detruit le #Sprite spécifié et libère la mémoire qu'il occupe. Vous ne pouvez  ↩
   plus
y faire référence.
```

## 1.7 freespritebuffer

```
  Syntaxe
FreeSpriteBuffer(#SpriteBuffer)

  Résumé
Detruit le #SpriteBuffer spécifié.
```

## 1.8 initsprite

```
  Syntaxe
Résultat = InitSpriteFile(#MaxDisplayedSprites, #MaxSpriteBuffers, #MaxSprites)

  Résumé
Initialise l'environnement nécessaire à la future gestion des sprites.
Cette commande doit être appelée avant toute les autres commandes
concernant les sprites. Si le 'Resultat' est NULL, alors l'initialisation
a échoué (manque de mémoire).
```

## 1.9 loadsprite

```
  Syntaxe
Résultat = LoadSprite(#Sprite, FileName$)

  Résumé
Crée un sprite à partir d'un fichier image IFF/ILBM (compressée ou non).
Ce sprite est utilisable immédiatement. Si une erreur survient lors du
chargement, une valeur négative est retournée:

  -1 : Fichier introuvable.
  -2 : Ce fichier n'est pas un fichier IFF/ILBM
  -3 : Plus assez de mémoire
  -4 : Fichier IFF/ILBM corrompu.
```

## 1.10 resetspriteserver

```
  Syntaxe
ResetSpriteServer()

  Résumé
Une fois que vous avez fini d'afficher tous les sprites voulus, vous devez
remettre le serveur à 0, et indiquer ainsi que vous n'avez plus de sprites
à afficher. En fait, vous devez vous assurez que tous les sprites ont
effectivement été afficher grâce à la commande 'WaitSpriteServer()', sinon
tous les sprites non affichés ne le seront jamais. Une solution classique
d'utilisation est la suivante:

  Repeat
      VWait()

      ..... ; Display all the sprites needed
```

```
        WaitSpriteServer()   ; Wait until all the sprites have been displayed
        ResetSpriteServer()  ; Reset the sprite server.
    Until
```

## 1.11 restorebackground

```
    Syntaxe
RestoreBackground()

    Résumé
Restaure le fond d'écran préalablement sauvegardé grâce à la commande
'AddBufferedSprite()' du 'SpriteBuffer' courant.
```

## 1.12 spritedepth

```
    Syntaxe
Résultat.w = SpriteDepth(#Sprite)

    Résumé
Renvoie la pronfondeur du #Sprite spécifié (son nombre de plans).
```

## 1.13 spriteheight

```
    Syntaxe
Résultat.w = SpriteHeight(#Sprite)

    Résumé
Renvoie la hauteur du #Sprite spécifié.
```

## 1.14 spritewidth

```
    Syntaxe
Résultat.w = SpriteWidth(#Sprite)

    Résumé
Renvoie la largeur du #Sprite spécifié.
```

## 1.15 startspriteserver

```
    Syntaxe
StartSpriteServer()

    Résumé
Alloue toutes les ressources nécessaires au serveur sprite (et particulièrement
```

le 'Blitter') et se met en mode attente. Vous pouvez alors utiliser n'importe quelle commande d'affichage de sprites (AddxxxxSprite()). Notez bien que tant que le serveur est actif, l'AmigaOS ne peut plus rien afficher (étant donné que tout l'affichage est géré par le Blitter). Une solution a ce problème est de stopper le serveur à la fin de la boucle, une fois que tous les sprites on été affichés. Cela permettra à l'OS d'utiliser un peu le Blitter en cas d'absolue nécessité. Cette commande est très rapide.

Notions avancée du serveur de sprite (pour les programmeurs chevronnés):

Ce serveur est architecturé autour d'une liste chainée de sprites. Donc quand on ajoute un sprite, il y a deux possibilité: la liste est vide, donc le sprite est affiché immédiatement ou la liste a au moins un élément et donc le sprite est ajouté à la fin de la liste puis sera affiché plus tard. Tous les sprites seront donc affichés dans le bon ordre. Le grand point positif de ce système, c'est qu'il n'est pas nécessaire d'attendre la fin de l'affichage des sprites, libérant ainsi le CPU pour d'autre tâches. Pour être sûr que tous les sprites ont été affichés, il est conseillé d'utiliser toujours la commande 'WaitSpriteServer()'.

## 1.16 stopspriteserver

Syntaxe
```
StopSpriteServer()
```

Résumé
Arrête le serveur et libère immédiatement le Blitter. Si des sprites n'ont pas été affichés, ils ne le seront jamais. Cette commandes est très rapide.

## 1.17 usespritebuffer

Syntaxe
```
UseSpriteBuffer(#SpriteBuffer)
```

Résumé
Change le 'SpriteBuffer' courant par celui spécifié.

## 1.18 waitspriteserver

Syntaxe
```
WaitSpriteServer()
```

Résumé
Attend jusqu'à ce que le serveur ait fini d'afficher tous les sprites. Il est conseillé d'utiliser toujours cette commande, même si vous êtes sûrs que tous les sprites ont été affichés. Si le serveur a déjà fini lors de l'appel de cette commande, elle quitte immédiatement.