

Chunky

COLLABORATORS

	<i>TITLE :</i> Chunky		
<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	Chunky	1
1.1	Chunky V1.00	1
1.2	allocatetchunkybuffer	1
1.3	chunkyblit	2
1.4	chunkyblock	2
1.5	chunkycls	2
1.6	chunkyid	3
1.7	chunkyplot	3
1.8	chunkytoplanar	3
1.9	freechunkybuffer	3
1.10	initchunky	3
1.11	usechunkybuffer	4

Chapter 1

Chunky

1.1 Chunky V1.00

PureBasic - Chunky library V1.00

Le 'Chunky' est un nom désignant une méthode pour stocker les données graphiques, un peu comme les plans (BitMap) sur les Amiga classiques. Mais la gestion interne des 'Chunky' est complètement opposée à celle des plans (ou 'Planar') communs sur l'Amiga. Chaque pixel est représenté par un octet dans la mémoire donc on ne peut avoir que des écrans en 256 couleurs (car un octet peut prendre 256 valeurs différentes). En standard, un Amiga ne peut pas afficher directement les 'chunky' et une conversion, appelée 'ChunkyToPlanar' doit avoir lieu. Travailler en 'Chunky' peut s'avérer beaucoup plus rapide pour certaines opérations (moteur 3D, déplacements de blocs, changement de nombreux pixels...) mais le temps nécessaire pour la conversion est très important et seuls les Amiga rapides (68030 ou plus) pourront les gérer avec efficacité. A noter que toutes les cartes graphiques sur Amiga travaillent en 'Chunky'.

Commandes disponibles:

```
AllocateChunkyBuffer
ChunkyBlit
ChunkyBlock
ChunkyCls
ChunkyID
ChunkyPlot
ChunkyToPlanar
FreeChunkyBuffer
InitChunky
UseChunkyBuffer
```

1.2 allocat chunkybuffer

Syntaxe

```
ChunkyID.1 = AllocateChunkyBuffer(#ChunkyBuffer, Width, Height)
```

Résumé

Alloue un 'ChunkyBuffer' et retourne son emplacement dans la mémoire ou NULL si il n'y a plus assez de mémoire libre. Un 'ChunkyBuffer' est une zone de mémoire continue où chaque octet va représenter un pixel sur l'écran. Donc les 'ChunkyBuffers' sont des images en mémoire d'écrans en 256 couleurs. Comme l'Amiga n'est pas capable d'afficher ce type de graphiques, une conversion doit avoir lieu. Cette conversion s'appelle ChunkyToPlanar .

1.3 chunkyblit

Syntaxe

ChunkyBlit(ShapeWidth, ShapeHeight, *SpriteAdress, X, Y)

Résumé

Affiche le sprite à la position donnée. Le sprite doit être au format chunky. La couleur 0 du sprite sera considérée comme transparente. Le sprite peut être de n'importe quelle taille. Assurez vous cependant que vous affichez bien le sprite dans le chunky buffer, sinon vous allez écrire dans une zone de mémoire non réservée et planter l'Amiga.

1.4 chunkyblock

Syntaxe

ChunkyBlock(ShapeWidth, ShapeHeight, *ShapeAdress, X, Y)

Résumé

Affiche le sprite à la position donnée. Cette fonction est à peu près 5 fois plus rapide que ChunkyBlit() mais a des limitations:

- * Le sprite doit avoir une largeur multiple de 4
- * Il n'y a pas de couleurs transparentes

Vérifiez toujours que vous affichez le sprite à l'intérieur du ChunkyBuffer.

Note: C'est la manière la plus rapide d'afficher un sprite dans un ChunkyBuffer. Cette fonction a été spécialement optimisée pour être rapide.

1.5 chunkycls

Syntaxe

ChunkyCls(Colour)

Résumé

Remplit entièrement le ChunkyBuffer courant de la couleur spécifiée.

1.6 chunkyid

Syntaxe

```
ChunkyID.l = ChunkyID()
```

Résumé

Renvoie la position mémoire du ChunkyBuffer courant.

1.7 chunkyplot

Syntaxe

```
ChunkyPlot(X, Y, Colour)
```

Résumé

Trace un point de la couleur donnée aux coordonnées spécifiées.

1.8 chunkytoplanar

Syntaxe

```
ChunkyToPlanar(BitMapID, OffsetY, Height)
```

Résumé

Convertit le ChunkyBuffer courant sur le BitMap spécifié. Cette fonction est très restrictive et vous devez suivre avec rigueur les règles suivantes:

- La largeur du ChunkyBuffer doit être égale à 320.
- Le BitMap et le ChunkyBuffer doivent avoir la même taille
- Le BitMap doit être alloué avec `AllocateLinearBitMap()` .

Note: la hauteur n'est pas limitée.

Cette fonction est bien entendue très rapide (d'où les restrictions) et respecte le système donc vous pouvez l'utiliser librement dans vos programmes.

1.9 freechunkybuffer

Syntaxe

```
FreeChunkyBuffer(#ChunkyBuffer)
```

Résumé

Détruit le ChunkyBuffer donné et libère la mémoire qu'il occupait.

1.10 initchunky

Syntaxe

```
result.l = InitChunky(#NumChunkyBufferMax)
```

Résumé

Initialise l'environnement 'Chunky' pour gérer correctement les objets 'Chunky'. Vous devez appeler cette fonction avant d'utiliser les commandes de cette bibliothèque. Si le résultat est NULL, alors la mémoire est insuffisante, sinon tout est bien initialisé.

#NumChunkyBufferMax : Nombre maximal de 'ChunkyBuffers' à gérer

1.11 usechunkybuffer

Syntaxe

```
UseChunkyBuffer(#ChunkyBuffer)
```

Résumé

Change le ChunkyBuffer courant par celui spécifié.