

Chunky

COLLABORATORS

	<i>TITLE :</i> Chunky		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		April 15, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Chunky	1
1.1	Chunky V1.00	1
1.2	allocatchunkybuffer	2
1.3	chunkyblit	2
1.4	chunkyblock	2
1.5	chunkycls	3
1.6	chunkyid	3
1.7	chunkyplot	3
1.8	chunkytoplanar	3
1.9	freechunkybuffer	4
1.10	initchunky	4
1.11	usechunkybuffer	4

Chapter 1

Chunky

1.1 Chunky V1.00

PureBasic - Chunky library V1.00

'Chunky' ist eine Methode, um Grafikdaten zu speichern, ein wenig wie planar. Aber, diese Methode arbeitet auf dem entgegengesetzten Weg, wie planar. Jeder Pixel wird durch ein Byte im Speicher repräsentiert, dadurch können Sie nur 256 Farben - Bildschirme (ein Byte entspricht 8 Bits, oder 8 Planes [Ebenen] wenn Sie so wollen) benutzen. Der Amiga kann solche Grafikdaten nicht darstellen, weshalb erst eine Konvertierung erfolgen muß. Diese ist gut bekannt als ChunkyToPlanar Routine. Das Arbeiten mit Chunky ist bei vielen Operationen schneller, da aber nur die CPU benutzt wird (und keine Amiga-Custom-Chips), benötigen Sie einen schnellen Computer um eine ordentliche Geschwindigkeit zu erreichen.

Befehlsübersicht:

AllocateChunkyBuffer

ChunkyBlit

ChunkyBlock

ChunkyCls

ChunkyID

ChunkyPlot

ChunkyToPlanar

FreeChunkyBuffer

InitChunky

UseChunkyBuffer

1.2 allocatetchunkybuffer

SYNTAX

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

FUNCTION

Es wird ein ChunkyBuffer reserviert. Der sogenannte ChunkyBuffer ist ein Speicherbereich, welcher einen 256-Farben-Bildschirm mit den angegebenen Dimensionen beschreibt. Da der Amiga nicht in der Lage ist, solche Art von Grafiken darzustellen, benötigen Sie eine Konvertiererroutine, genannt ChunkyToPlanar, welche den Inhalt des ChunkyBuffers als Standard-Amiga-Grafik darstellt. Tatsächlich ist Chunky die entgegengesetzte Methode zu Planar. Der größte Vorteil von Chunky ist, dass Sie große Speicherblöcke (wie Sprites, 3D-Engines) sehr schnell bewegen können. Der Nachteil liegt darin, dass Sie die Amiga-Custom-Chips (Blitter, Copper...) nicht benutzen können, um die Vorgänge zu beschleunigen. Sie benötigen deshalb einen schnellen Computer (030 oder besser), um eine annehmbare Frame-Rate zu erreichen. Das Chunky-Format wird ebenfalls auf Grafikkarten benutzt, so können Sie den Chunky-Buffer direkt in das Video-Ram der Grafikkarte kopieren.

Die zurückgegebene ChunkyID ist die Speicherposition des Chunky-Buffers. Ist diese NULL, wurde der ChunkyBuffer nicht reserviert, führen Sie dann keine Operationen damit aus.

Der ChunkyBuffer wird reserviert im #ANY_MEMORY Speicherplatz (im FastRam, wenn welches verfügbar ist, sonst im ChipRam).

1.3 chunkyblit

SYNTAX

```
ChunkyBlit(ShapeWidth, ShapeHeight, *ShapeAdress, X, Y )
```

FUNCTION

Blittet das angegebene ChunkyShape in den angegebenen ChunkyBuffer. Diese Funktion benutzt Farbe 0 als transparent. Dies ist eine hochoptimierte Funktion. Das Shape kann von beliebiger Größe sein. Diese Funktion wird nicht abgeschnitten (clipped), seien Sie also sicher, dass Sie sich bei der Blit-Operation INNERHALB des Chunky-Buffers befinden.

1.4 chunkyblock

SYNTAX

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

FUNCTION

Diese Funktion ist 5 mal schneller als die ChunkyBlit - Funktion, hat aber einige Einschränkungen:

- * Die Shape-Breite muß ein Vielfaches von 4 sein.
- * Es gibt keine transparenten Farben.

Diese Funktion wird nicht abgeschnitten (clipped), beachten Sie also beim Blitten, dass Sie sich INNERHALB des Buffers befinden.

HINWEIS: Dies ist der schnellste Weg, ein Shape innerhalb eines ChunkyBuffers zu blitten. Die Funktion wurde extra auf Geschwindigkeit optimiert.

1.5 chunkycls

SYNTAX

ChunkyCls(Colour)

STATEMENT

Füllt den aktuellen Chunky-Buffer mit der angegebenen Farbe.

1.6 chunkyid

SYNTAX

ChunkyID.l = ChunkyID()

FUNCTION

Gibt die Speicherposition des aktuellen ChunkyBuffers zurück.

1.7 chunkyplot

SYNTAX

NChunkyPlot(X, Y, Colour)

STATEMENT

Zeichnet einen Punkt an den Koordinaten (X,Y) in der angegebenen Farbe.

1.8 chunkytoplanar

SYNTAX

ChunkyToPlanar(BitMapID, OffsetY, Height)

FUNCTION

Dies konvertiert den aktuellen ChunkyBuffer in die vorgegebene BitMap. Diese Funktion ist sehr eingeschränkt und Sie müssen sorgfältig diese Regeln befolgen:

- Die Breite muß gleich 320 sein. Es werden keine anderen Breiten unterstützt.
- Die BitMap und der ChunkyBuffer müssen dieselbe Größe haben.
- Die BitMap muß mit AllocateLinearBitMap() reserviert werden.

Hinweis: Die Höhe ist nicht eingeschränkt.

Diese Funktion ist natürlich sehr schnell und systemfreundlich. Sie können Sie beruhigt in Ihren Applikationen/Programmen benutzen.

1.9 freechunkybuffer

SYNTAX

FreeChunkyBuffer(#ChunkyBuffer)

STATEMENT

Gibt den angegebenen ChunkyBuffer (und seinen Speicherplatz) frei.

1.10 initchunky

SYNTAX

result.l = InitChunky(#NumChunkyBufferMax)

FUNCTION

Initialisiert die gesamte Chunky Programmumgebung zur späteren Benutzung. Sie müssen diese Funktion am Anfang Ihres Programmcodes ausführen, wenn Sie die Chunky Befehle benutzen möchten. Sie können das Ergebnis kontrollieren, um die korrekte Initialisierung der Chunky Umgebung überprüfen zu können.

#NumChunkyBufferMax : Maximale Anzahl zu verwaltender ChunkyBuffer.

1.11 usechunkybuffer

SYNTAX

UseChunkyBuffer(#ChunkyBuffer)

STATEMENT

Macht den angegebenen #ChunkyBuffer zum aktuellen ChunkyBuffer.