Beschreibung zur Graphamp.Dll

Inhalt

Das ganze Paket beinhaltet folgende Dateien :

a) Die Graphamp.dll
b) Eztw32.dll
c) Ein Testprogramm (PRF), um die Funktionen zu testen (Profan² Version 7 - ist aber leicht anzupassen)
d) Eine Dll mit zwei Testbildern.
e) Diese PDF-Datei
f) Reg_Key.exe zum Freischalten der Dll

Die EZTW32.DLL, die zum Scannen benötigt wird, muss sich im Pfad befinden, sonst startet die Graphamp.dll nicht.

Besonderheiten:

Die Dll ist auch in der Lage, die Bitmaps aus einem Modul zu laden (DLL oder EXE).
Diese Resourcen sind ja mit dem Resourcehacker relativ leicht in eine Dll zu packen.
Beim Laden aus Modulen sind aber ein paar Kleinigkeiten zu beachten:
1. Der Resourcename muss, wenn die Resource nur eine Nummer hat folgendermaßen aussehen "#100".
2. Die Extension muss unbedingt richtig angegeben werden. Ist die Resource ein Bitmap und die Extension JPG dann kommt es garantiert zu einer Fehlermeldung.

Die Dll kann ein eigenes Bitmap-Format erzeugen und natürlich auch wieder einlesen.
Dieses Format kann nicht von anderen Grafikprogrammen gelesen werden.
Es eignet sich also ganz gut dazu seine Grafiken zu schützen oder als Resource in Programme einzubinden.
Ausserdem wird das Bitmap komprimiert.
Je nach Größe der Originals liegt der Komprimierungfaktor zwischen 40% und 50%.
Die Endung ist PRB (eben für **Pr**ofan-**B**itmap).

Wozu dient die Dll ?

Einfach gesagt zum Laden,Speichern und Manipulieren von Pixel-Grafiken, die vom aufrufenden Programm nicht standardmässig unterstützt werden.

21 Effekte und 25 Filter können zur Manipulation der Bitmaps, sowie 7 Filter zum Resamplen der Bitmaps eingesetzt werden.

Ladbare Formate :
1.BMP
2.JPG -progressiv / nicht progressiv
3.TIF
4.WMF
5.TGA
6.PCX
7.PCD
8.PNG
9.GIF (keine animierten)
10.PRB
11.Bitmaps vom Scanner(Twain)

Speicherbare Format :
1.BMP
2.JPG
3.PNG
4.PRB

Internes

Die Dll muss beim ersten Aufruf initialisiert werden. Hier werden interne Arrays angelegt.
Die Dll kann bis 500 Bitmaps im Speicher halten. Es ist aber nicht unbedingt ratsam 500 grosse Bilder zu laden, das kann zum Systemabsturz führen. Bei kleineren Bitmaps sollte es aber keine Probleme geben. So kann man z.B. Animationen erzeugen.
Beim Programmende ist unbedingt darauf zu achten, dass die geladenen Bitmaps wieder freigegeben werden. Sonst werden die GDI-Resourcen erst beim Herunterfahren von Windows freigegeben.
Die Dll gibt beim Laden von Bitmaps immer nur das Handle zurück. Was der Programmierer damit anfängt ist dann seine Sache.

Resourcen

Es gibt eine zusätzliche Resource in der Dll, und zwar einen Mauscursor, das konnte ich mir als alter Atarianer nicht verkneifen.

Funktionen

Insgesamt hat die Dll 44 Funktionen.
Im einzelnen sind das :

InitDLL
initialisiert die Dll
Parameter : Integer (Anzahl der möglichen Controls)
: Zeiger auf String (Freischaltcode)
Rückgabe : Falsch oder Wahr (0 oder 1) --> Bool

IsInit
ist die Dll initialisieren ??
Rückgabe : Falsch oder Wahr (0 oder 1) --> Bool

ExitDll
Freigabe der Resourcen, muss auf jeden Fall vor
Programm-Ende aufgerufen werden !! Es muss unbedingt
darauf geachtet werden, dass das Programm korrekt beendet wird !!
Rückgabe : Falsch oder Wahr (0 oder 1) --> Bool

CreateTempfilename
Temporären Filenamen ermitteln
Könnte man vielleicht für UNDO-Files benutzen ?
Parameter : Zeiger auf String (max. 256 Zeichen)
Rückgabe : Falsch oder Wahr (0 oder 1) --> Bool

GetDllInfo
Ausgabe von Informationen
Rückgabe : Falsch oder Wahr (0 oder 1) --> Bool

P_LoadImage
es können maximal 500 Handles im Speicher gehalten werden
Ladbare Formate :
BMP,JPG,GIF,TIF,PCX,TGA,PCD,WMF,PRB
Parameter : Elternfenster (%hwnd)
: Zeiger auf Dateinamen
: Index des Bitmaps
: Zeiger auf LongInt --> hier wird die Breite zurückgegeben
: Zeiger auf LongInt --> hier wird die Höhe zurückgegeben
: LongInt (Stretch to Window) --> passt das Bitmap proportional an das Elternfenster an
Rückgabe : BitmapHandle

P_LoadImageFromModul
lädt ein Bitmap aus einer DLL oder EXE
Ladbare Formate :
BMP,JPG,GIF,TIF,PCX,TGA,PCD,WMF,PRB
Parameter : Elternfenster (%hwnd)
: Index des Bitmaps
: Zeiger auf LongInt --> hier wird die Breite zurückgegeben
: Zeiger auf LongInt --> hier wird die Höhe zurückgegeben
: LongInt (Stretch to Window) --> passt das Bitmap proportional an das Elternfenster an
: Modulname (Zeiger auf String)
: Resourcename (Zeiger auf String)
: ResourceTyp (Zeiger auf String)
: Extension (ohne Punkt nur z.B."jpg")
Rückgabe : BitmapHandle

P_GetBitmapInfo
Liefert Informationen zum Bitmap
Parameter : Zeiger auf String (Filename)
: Zeiger auf LongInt (Breite)
: Zeiger auf LongInt (Höhe)
: Zeiger auf LongInt (Pixelformat)

P_FreeBitmap
Löscht ein Bitmap aus dem Speicher
Parameter : Index des Bitmaps

P_StretchToFit
Streckt das Bild ins Elternfenster
Es ist unbeding darauf zu achten, dass der BildIndex stimmt,
sonst wird das vorhandene Bild zerstoert.
Parameter : HWND(LongInt)
: BildIndex(LongInt)
Rückgabe : Bitmaphandle(LongInt)

P_Greyscale
wandelt das Bitmap in Graustufen
Es ist unbeding darauf zu achten, dass der BildIndex stimmt,
sonst wird das vorhandene Bild zerstoert.
Parameter : BildIndex(LongInt)
Rückgabe : Bitmaphandle(LongInt)

P_Negativ
kehrt die Farben um
Es ist unbeding darauf zu achten, dass der BildIndex stimmt,
sonst wird das vorhandene Bild zerstoert.
Parameter : BildIndex(LongInt)
Rückgabe : Bitmaphandle(LongInt)

P_Contrast
Kontrast einstellen
Es ist unbeding darauf zu achten, dass der BildIndex stimmt,
sonst wird das vorhandene Bild zerstoert.
Parameter : BildIndex(LongInt)
: Wert (LongInt)
Rückgabe : Bitmaphandle(LongInt)

P_Lightness
Helligkeit einstellen
Es ist unbeding darauf zu achten, dass der BildIndex stimmt,
sonst wird das vorhandene Bild zerstoert.
Parameter : BildIndex(LongInt)
: Wert (LongInt)
Rückgabe : Bitmaphandle(LongInt)

P_Mirror
spiegelt das Bitmap horizontal
Es ist unbeding darauf zu achten, dass der BildIndex stimmt,
sonst wird das vorhandene Bild zerstoert.
Parameter : BildIndex(LongInt)
Rückgabe : Bitmaphandle(LongInt)

P_Flip
spiegelt das Bitmap vertikal
Es ist unbeding darauf zu achten, dass der BildIndex stimmt,
sonst wird das vorhandene Bild zerstoert.
Parameter : BildIndex(LongInt)
Rückgabe : Bitmaphandle(LongInt)

P_Resample
ändert die Bitmapgroesse
Es ist unbeding darauf zu achten, dass der BildIndex stimmt,
sonst wird das vorhandene Bild zerstoert.
Parameter : BildIndex(LongInt)
: neue Breite (LongInt)
: neue Höhe (LongInt)
: Filter (LongInt) ***siehe Filter***
Rückgabe : Bitmaphandle(LongInt)

P_Crop
schneidet einen Bereich aus
Es ist unbeding darauf zu achten, dass der BildIndex stimmt,
sonst wird das vorhandene Bild zerstoert.
Parameter : BildIndex(LongInt)
: Left (LongInt)
: Top (LongInt)
: Right (LongInt)
: Bottom (LongInt)
Rückgabe : Bitmaphandle(LongInt)

P_SetToClipboard
speichert das Bitmap in der Zwischenablage
Es ist unbeding darauf zu achten, dass der BildIndex stimmt,
sonst wird das vorhandene Bild zerstoert.
Parameter : BildIndex(LongInt)
Rückgabe : Bitmaphandle(LongInt)

P_GetFromClipboard
holt das Bitmap aus der Zwischenablage
Es ist unbeding darauf zu achten, dass der BildIndex stimmt,
sonst wird das vorhandene Bild zerstoert.
Parameter : BildIndex(LongInt)
Rückgabe : Bitmaphandle(LongInt)

P_SetJpgOptions
setzt die Jpeg-Optionen
Parameter : Speicherqualität (Longint) 0% - 100%

P_SaveBitmap
speichert das Bitmap
Parameter : Index
: Dateiname

P_AssignBitmap
Legt ein neues Bitmap im Array an und kopiert das Bitmap mit INDEX[]
hinein. Vielleicht für UNDO-Funktion ??
Parameter : Index (alt)
: Index (neu)
Rückgabe : Bitmaphandle(LongInt)

P_ConvertTo256
Konvertiert das Bitmap mit INDEX in ein 256 Farben-Bitmap
Parameter : Index
Rückgabe : Bitmaphandle(LongInt)

P_AssignExtBitmap
Legt ein neues Bitmap im Array an und kopiert das Bitmap mit HANDLE
hinein. Vielleicht für Scanmodule geeignet ??
Parameter : HWND
: Index (alt)
: Handle (neu)
: LongInt (Stretch to Window) --> passt das Bitmap proportional an das Elternfenster an
Rückgabe : Bitmaphandle(LongInt)

P_CreateEmptyBitmap
Legt ein neues Bitmap (leer)im Array an.
Mit P_AssignExtBitmap vielleicht für Scanmodule geeignet ??
Parameter : Index (alt)
Rückgabe : Bitmaphandle(LongInt)

P_Rotate
Rotiert das Bitmap mit INDEX und GRAD
Parameter : INDEX
: GRAD
: Hintergrungfarbe
Rückgabe : Bitmaphandle

P_Blend
Blendet 2 Bitmap ineinander
ist nur sinnvoll wenn beide Bitmaps gleich gross sind.
sonst wird Bitmap2 an die Groesse von Bitmap 1 angepasst.
Parameter : Index 1
: Index 2
: MaxBlendValue(LongInt)
: Multiplikator(LongInt)
Rückgabe : Bitmaphandle(LongInt)

P_Filter
wendet einen Filter an
Parameter : Index 1
: Effekt (1-25) siehe ***Filter***
Rückgabe : Bitmaphandle(LongInt)

P_IsTwain
fragt ab, ob ein Twain-Modul installiert ist
Rückgabe : Bool

P_ScanToBitmap
scannt ein Bitmap ein. Die EZTW32.DLL wird benötigt !
Parameter : HWND
: Index
: Quelle wählen oder nicht (1 oder 0)
: Ohne Manager (1 oder 0)
: Clipboard (1 oder 0)
Rückgabe : Bitmaphandle(LongInt)

P_Gamma
Gamma-Korrektur
Parameter : Index
: Wert (Vorkomma)
: Wert (Nachkomma)
Rückgabe : Bitmaphandle(LongInt)
P_Colorize
Kollorierung
Parameter : Index
: Wert (Hue)
: Wert (Saturation)
: Wert (Lightness)
Rückgabe : Bitmaphandle(LongInt)

P_Solarize
Solarisation
Parameter : Index
: Wert (Solarisation)
Rückgabe : Bitmaphandle(LongInt)

P_Spray
Sprühdose
Parameter : Index
: Wert
Rückgabe : Bitmaphandle(LongInt)

P_IncDecRGB
Farben einzeln korigieren
Parameter : Index
: Wert (Rot)
: Wert (Grün)
: Wert (Blau)
Rückgabe : Bitmaphandle(LongInt)

P_PrintBitmap
Druck das Bitmap mit INDEX auf DC
Das kann entweder %HDC,%HDC2,ein DruckerDeviceContext oder iergendein anderer mit GetDC ermittelter Device-Context sein. X,Y,W,H bezeichen das Rechteck in das gedruckt wird.
Parameter : Index
: DC
: x
: y
: w
: h
Rückgabe : Bitmaphandle

P_Arithmetic
verbindet 2 Bitmaps Mit INDEX unter Verwendung von FILTER
Parameter : Index 1
: Index 2
: FILTER (1-12) ***siehe Filter***
Rückgabe : Bitmaphandle(LongInt)

P_AddNoise
gibt ein Rauschen über das Bitmap
Parameter : Index
: Wert
: MONO (1 oder 0) wahr oder falsch
Rückgabe : Bitmaphandle(LongInt)

P_AntiAlias
entfernt Moire-Effekte
Parameter : Index
: R (1 oder 0)
: G (1 oder 0)
: B (1 oder 0)
Rückgabe : Bitmaphandle(LongInt)

P_Mosaic
erzeugt ein Mosaic aus dem Bitmap
Parameter : Index
: Breite(Integer)
: Höhe(Integer)
Rückgabe : Bitmaphandle(LongInt)

P_GetLoadFilename
interner Laden-Dialog mit Bildvorschau
Parameter : Zeiger auf String, der den Filenamen zurückgibt
Rückgabe : Bool

P_GetSaveFilename
interner Spehcern-Dialog mit Bildvorschau
Parameter : Zeiger auf String, der den Filenamen zurückgibt
Rückgabe : Bool

Filter

Insgesamt gibt es 25 Filter-Funktion mit denen das Bitmap manipuliert werden kann.
Im einzelnen :
1.EmbossColor
2.EmbossLight
3.EmbossMedium
4.EmbossDark
5.Enhance
6.BlurBartlett
7.BlurGaussian
8.Negative
9.Average
10.Blur
11.BlurSoftly
12.BlurMore
13.Prewitt
14.TraceContour
15.Sharpen
16.SharpenMore
17.SharpenLess
18.UnSharpMask
19.EdgesStrong
20.EdgesWeak
21.Etch
22.LaplacianHV
23.LaplacianOmni
24.SharpenDirectional
25.SobelPass

Arithmetic-Filter :
1.None;
2.Add;
3.Substract;
4.Multiply;
5.Divide;
6.Darkest;
7.Lightest;
8.Difference;
9.BinaryOr;
10.BinaryAnd;
11.Average;

Resample-Filter :
1.SplineFilter
2.BellFilter
3.TriangleFilter
4.BoxFilter
5.HermiteFilter
6.Lanczos3Filter
7.MitchellFilter

Definitionen fürProfan²

```
DEF Init_dll(2) ! "Graphamp.dll","InitDll"
DEF Is_Init(0) ! "Graphamp.dll","IsInit"
DEF Exit_Dll(0) ! "Graphamp.dll","ExitDll"
DEF Create_Tempfilename(1) ! "Graphamp.dll","CreateTempfilename"
DEF GetDllInfo(0) ! "Graphamp.dll","GetDllInfo"
DEF P_LoadImage(6) ! "Graphamp.dll","P_LoadImage"
DEF P_LoadImageFromModul(9) ! "Graphamp.dll","P_LoadImageFromModul"
DEF P_GetBitmapInfo(4) ! "Graphamp.dll","P_GetBitmapInfo"
DEF P_FreeBitmap(1) ! "Graphamp.dll","P_FreeBitmap"
DEF P_StretchToFit(2) ! "Graphamp.dll","P_StretchToFit"
DEF P_Greyscale(1) ! "Graphamp.dll","P_Greyscale"
DEF P_Negativ(1) ! "Graphamp.dll","P_Negativ"
DEF P_Contrast(2) ! "Graphamp.dll","P_Contrast"
DEF P_Lightness(2) ! "Graphamp.dll","P_Lightness"
DEF P_Mirror(1) ! "Graphamp.dll","P_Mirror"
DEF P_Flip(1) ! "Graphamp.dll","P_Flip"
DEF P_Resample(3) ! "Graphamp.dll","P_Resample"
DEF P_Crop(5) ! "Graphamp.dll","P_Crop"
DEF P_SetToClipboard(1) ! "Graphamp.dll","P_SetToClipboard"
DEF P_GetFromClipboard(1) ! "Graphamp.dll","P_GetFromClipboard"
DEF P_SetJpgOptions(1) ! "Graphamp.dll","P_SetJpgOptions"
DEF P_SaveBitmap(2) ! "Graphamp.dll","P_SaveBitmap"
DEF P_AssignBitmap(2) ! "Graphamp.dll","P_AssignBitmap"
DEF P_Blend(4) ! "Graphamp.dll","P_Blend"
DEF P_Filter(2) ! "Graphamp.dll","P_Filter"
DEF P_AssignExtBitmap(4) ! "Graphamp.dll","P_AssignExtBitmap"
DEF P_CreateEmptyBitmap(1) ! "Graphamp.dll","P_CreateEmptyBitmap"
DEF P_Rotate(3) ! "Graphamp.dll","P_Rotate"
DEF P_IsTwain(0) ! "Graphamp.dll","P_IsTwain"
DEF P_ScanToBitmap(3) ! "Graphamp.dll","P_ScanToBitmap"
DEF P_Gamma(3) ! "Graphamp.dll","P_Gamma"
DEF P_Colorize(4) ! "Graphamp.dll","P_Colorize"
DEF P_Solarize(2) ! "Graphamp.dll","P_Solarize"
DEF P_Spray(2) ! "Graphamp.dll","P_Spray"
DEF P_IncDecRGB(4) ! "Graphamp.dll","P_IncDecRGB"
DEF P_PrintBitmap(6) ! "Graphamp.dll","P_PrintBitmap"
DEF P_Arithmetic(3) ! "Graphamp.dll","P_Arithmetic"
DEF P_ConvertTo256(1) ! "Graphamp.dll","P_ConvertTo256"
DEF P_AddNoise(3) ! "Graphamp.dll","P_AddNoise"
DEF P_AntiAlias(4) ! "Graphamp.dll","P_AntiAlias"
DEF P_Mosaic(3) ! "Graphamp.dll","P_Mosaic"
DEF P_GetLoadFilename(1) ! "Graphamp.dll","P_GetLoadFilename"
DEF P_GetSaveFilename(1) ! "Graphamp.dll","P_GetSaveFilename"
```

Tips und Tricks

Bilder berarbeiten

Normalerweise lädt man das Bild so ein, dass man es auch gut ansehen kann, also an das Elterncontrol angepasst. Das hat aber zur Folge, dass es nicht mehr unbedingt die Originalausmaße besitzt, sondern verkleinert wurde. Wenn ich das Bild jetzt mit einem Filter bearbeite kann ich nur das verkleinerte Bild abspeichern.
Abhilfe : Das Bild zweimal laden. Einmal zum anschauen und einmal nur ins Speicherarry. Alles, was ich mit dem "Vorschaubild" mache, kann ich im Speicher mit dem 2.Bild proportional auch machen.

Slideshow

Es ist ein Leichtes mit der Dll eine Slideshow zu programmieren. Einfach alle Files die man anschauen möchte in eine Liste einlesen und diese abarbeiten. Es gibt allerdings etwas zu beachten.
Beachte : Normalerweise wird die Slideshow in einer Schleife angezeigt. Und hier sollte eine Pause eingebaut werden, da es bei grossen Bildern zu Verzögerungen beim Laden kommen kann. Das gibt dann unschöne Effekte. Die Pause darf aber auf keinen Fall SLEEP sein, da dann auch keine Bilder geladen werden. Besser ist es einen Timer zu benutzen.

Beispiel :
While ....
Bild laden und anzeigen
Settimer 1000
Waitinput
Killtimer
Wend
Macht mindestens 1 Sekunde Pause !

Filter mischen

Durch geschicktes Mischen der Filter kann man sich schöne Effekte selber zusammenstellen.
Einen Zeichen z.B. durch TraceContour und Überblenden.

Rechtliches

Freeware - DLL für Freeware-Programme !
Diese Dll darf nur für Freeware-Programm eingesetzt werden. Jegliche Form der kommerziellen Nutzung ist untersagt.

Autoren:
Andreas Miethe -> GRAPHAMP.DLL -> basierend auf der Arbeit von Andreas Moser.

Nutzungsbedingungen :
Da einiges an Zeit bei der Erstellung der DLL investiert wurde, und nicht alles auf meinem "Mist" gewachsen ist, erwarte ich im Programm, einer zuegehörigen Hilfedatei oder Textdatei einen Hinweis auf die Autoren, etwa wir folgt:

"Die Graphamp.Dll wurde programmiert von Andreas Miethe basierend auf der Arbeit von Andreas Moser".

Beachte:
JEDE VERWENDUNG DER "GRAPHAMP.DLL" ERFOLGT AUF EIGENE GEFAHR.
DER FREISCHALT-CODE DARF NICHT WEITERGEGEBEN WERDEN !
DIE DLL DARF NICHT VERKAUFT, ODER SONSTWIE ZU KOMMERZIELLEN ZWECKEN GENUTZT WERDEN. SIE DARF AUSSCHLIESSLICH FÜR FREEWARE-PROGRAMME GENUTZT WERDEN.

Andreas Miethe im September 2001

Freischalten

Die Dll ist zwar Freeware, aber um zu sie voll d.h. ohne Nagscreen nutzen zu können, muss sie registriert werden.
Die Registrierung erfolgt mit dem Programm "Reg_Key.exe", das zum Paket gehört.

Registriername und Schlüssel kann über die Homepage des Authors angefordert werden.

Homepage