Programming Cookbook IV

“GLBasic Routines”

By

Nicholas Kingsley

Introduction

Welcome to the fourth part of my “Programming Cookbook” series.

This version of the book contains my GLBasic code, most of which shouldn't appear in the first part

Hopefully you will find the code useful, and if not, then hope it will give you some fresh ideas!

CreateDEBPackage

This was going to be a program to create a Linux DEB package, but as usual, I got bored...

```
// ------------------------------- //
// Project: CreateDEBPackage
// Start: Friday, September 30, 2011
// IDE Version: 10.118

TYPE TMainProgram
	file AS TFile

	screenWidth%;screenHeight%
	messageList$="message"

	FUNCTION Initialise%:
		IF self.file.Initialise()=FALSE THEN RETURN FALSE

		GETSCREENSIZE self.screenWidth%,self.screenHeight%
		RETURN TRUE
	ENDFUNCTION

	FUNCTION CreateMessageWindow%:
		DDgui_pushdialog(0,0,self.screenWidth%,self.screenHeight%,TRUE)
		DDgui_set("","TEXT","Progress")
		DDgui_text(self.messageList$,"",self.screenWidth%-8,self.screenHeight%-32)
		DDgui_show(FALSE)
	ENDFUNCTION

	FUNCTION AddMessage%:text$
	LOCAL temp$

		temp$=DDgui_get$(self.messageList$,"TEXT")
		IF temp$<>""
			INC temp$,"\n"+text$
		ELSE
			temp$=text$
		ENDIF

		DDgui_set(self.messageList$,"TEXT",temp$)
		DDgui_show(FALSE)
		SHOWSCREEN
	ENDFUNCTION

	FUNCTION UpdateLastMessage%:text$
	LOCAL messageText$[]
	LOCAL temp$,loop%

		temp$=DDgui_get$(self.messageList$,"TEXT")
		IF SPLITSTR(temp$,messageText$[],"\n")>0
			INC messageText$[BOUNDS(messageText$[],0)-1]," - "+text$
		ENDIF

		temp$=""
		FOR loop%=0 TO BOUNDS(messageText$[],0)-1
			INC temp$,messageText$[loop%]
			IF loop%<BOUNDS(messageText$[],0)-1 THEN INC temp$,"\n"
		NEXT

		DDgui_set(self.messageList$,"TEXT",temp$)
		DDgui_show(FALSE)
		SHOWSCREEN
	ENDFUNCTION

	FUNCTION CreatePackage%:
	LOCAL path$,error$,baseDir$
	LOCAL packageName$		// Name of package
	LOCAL version$			// Version

		self.file.Process()

		self.CreateMessageWindow()
		TRY
			self.AddMessage("Starting DEB package creation")
			self.AddMessage("Getting location of documents folder")
			path$=PLATFORMINFO$("DOCUMENTS")
			self.AddMessage("Storing DEB file in "+path$)
			self.AddMessage("Getting package name")
			packageName$=self.file.ReturnPackageName$()
			DEBUG "Package name : "+packageName$+"\n"
			self.AddMessage("Getting version")
			version$=self.file.ReturnVersionString$()
			self.AddMessage("Creating base directory")

			baseDir$=path$+"/"+packageName$+"-"+version$
```

```
                        IF DOESDIREXIST(baseDir$)=FALSE
                                IF CREATEDIR(baseDir$)=FALSE
                                        THROW "Unable to create the directory "+baseDir$
                                ELSE
                                        self.UpdateLastMessage("Created")
                                ENDIF
                        ELSE
                                self.UpdateLastMessage("Already Present")
                        ENDIF

                        IF self.CreateControlFile(baseDir$,packageName$,version$)=FALSE THEN THROW
"Error creating control file"
                        IF self.CreatePostInst(baseDir$)=FALSE THEN THROW "Error creating postinst
file"
                        IF self.CreatePostRM(baseDir$)=FALSE THEN THROW "Error creating postinst
file"
                        IF self.CreateUsr(baseDir$,packageName$)=FALSE THEN THROW "Error creating usr
and lower directories"

                        KEYWAIT
                CATCH error$
                        self.AddMessage(error$)
                FINALLY

                KEYWAIT
        ENDFUNCTION

        //! Create the control file
        FUNCTION CreateControlFile%:path$,packageName$,version$
        LOCAL handle%
        LOCAL temp$

                self.AddMessage("Creating DEBIAN directory")

                INC path$,"/DEBIAN"
                IF DOESDIREXIST(path$)=FALSE
                        IF CREATEDIR(path$)
                                self.UpdateLastMessage("Created")
                        ELSE
                                self.UpdateLastMessage("Unable to create")
                                RETURN FALSE
                        ENDIF
                ELSE
                        self.UpdateLastMessage("Already Present")
                ENDIF

                self.AddMessage("Creating control file")
                handle%=GENFILE()
                IF handle%<0 THEN RETURN FALSE

                INC path$,"/control"
                IF OPENFILE(handle%,path$,0)=TRUE
                        WRITESTR handle%,"Package: "+packageName$+"\n"
                        WRITESTR handle%,"Version: "+version$+"\n"
                        WRITESTR handle%,"Section: "+"graphics"+"\n" // Get section code later on
                        WRITESTR handle%,"Priority: "+"optional"+"\n" // Get priority later on
                        WRITESTR handle%,"Architecture: i386"+"\n" // Change later
                        WRITESTR handle%,"Depends: libsdl-mixer1.2 (>= 1.2.8-6.3), libjpeg62 (>= 6b1-
1ubuntu1), libc6 (>= 2.4)\n"
                        WRITESTR handle%,"Installed-Size: "+self.file.ReturnFileSize()+"\n"
                        WRITESTR handle%,"Maintainer: "+"Nicholas Kingsley "+"<njk@un-map.com>"+"\n"
                        WRITESTR handle%,"Provides: "+packageName$+"\n"
                        WRITESTR handle%,"Description: "+"Short"+"\n" // Get short description
                        // Write long description
                        CLOSEFILE handle%
                        self.UpdateLastMessage("Saved")
                        RETURN TRUE
                ELSE
                        RETURN FALSE
                ENDIF
        ENDFUNCTION

        //! Create post inst file
        FUNCTION CreatePostInst%:path$
        LOCAL handle%

                handle%=GENFILE()
                IF handle%<0 THEN RETURN FALSE

                self.AddMessage("Creating postinst file")

                IF OPENFILE(handle%,path$+"/DEBIAN/postinst",0)
                        WRITESTR handle%,"#!/bin/sh\n"
                        CLOSEFILE handle%

                        self.UpdateLastMessage("Saved")
                        RETURN TRUE
                ELSE
                        RETURN FALSE
                ENDIF
        ENDFUNCTION
```

```
//! Create post inst file
FUNCTION CreatePostRM%:path$
LOCAL handle%

	handle%=GENFILE()
	IF handle%<0 THEN RETURN FALSE

	self.AddMessage("Creating postrm file")

	IF OPENFILE(handle%,path$+"/DEBIAN/postrm",0)
		WRITESTR handle%,"#!/bin/sh\n"
		CLOSEFILE handle%

		self.UpdateLastMessage("Saved")
		RETURN TRUE
	ELSE
		RETURN FALSE
	ENDIF
ENDFUNCTION

//! Create usr directory
FUNCTION CreateUsr%:path$,packageName$
LOCAL dirs$[]; DIMDATA dirs$[],"bin","lib","share",""
LOCAL loop$,dir$

	self.AddMessage("Creating usr directory")

	INC path$,"/usr"
	IF DOESDIREXIST(path$)=FALSE
		IF CREATEDIR(path$)
			self.UpdateLastMessage("Created")
		ELSE
			self.UpdateLastMessage("Unable to create")
			RETURN FALSE
		ENDIF
	ELSE
		self.UpdateLastMessage("Already Present")
	ENDIF

	FOREACH loop$ IN dirs$[]
		dir$=path$+"/"+loop$
		self.AddMessage("Creating directory "+dir$)
		IF DOESDIREXIST(dir$)=FALSE
			IF CREATEDIR(dir$)
				self.UpdateLastMessage("Created")
			ELSE
				self.UpdateLastMessage("Unable to create")
				RETURN FALSE
			ENDIF
		ELSE
			self.UpdateLastMessage("Already Present")
		ENDIF

		IF loop$="share"
			IF self.CreateApplications(dir$,packageName$)=FALSE THEN RETURN FALSE
			IF self.CreateDocs(dir$,packageName$)=FALSE THEN RETURN FALSE
			IF self.CreateMenu(dir$,packageName$)=FALSE THEN RETURN FALSE
		ENDIF
	NEXT

	RETURN TRUE
ENDFUNCTION

//! Create applications directory
FUNCTION CreateApplications%:path$,packageName$
LOCAL handle%

	self.AddMessage("Creating applications directory")

	INC path$,"/applications"
	IF DOESDIREXIST(path$)=FALSE
		IF CREATEDIR(path$)
			self.UpdateLastMessage("Created")
		ELSE
			self.UpdateLastMessage("Unable to create")
			RETURN FALSE
		ENDIF
	ELSE
		self.UpdateLastMessage("Already Present")
	ENDIF

	handle%=GENFILE()
	IF handle%<0 THEN RETURN FALSE

	IF OPENFILE(handle%,path$+"/"+packageName$+".desktop",0)
		WRITESTR handle%,"[Desktop Entry]\n"
		WRITESTR handle%,"Version=1.0\n"
		WRITESTR handle%,"Type=Application\n"
		WRITESTR handle%,"Name="+"name"+"\n" // Get generic name later
		WRITESTR handle%,"GenericName=\n"
```

```
                        WRITESTR handle%,"Comment=\n"
                        WRITESTR handle%,"Exec=/usr/bin/"+packageName$+".linux\n"
                        WRITESTR handle%,"Icon=/usr/share/pixmaps/"+packageName$+".xpm\n"
                        WRITESTR handle%,"Categories="+"gen"+"\n" // Get category later
                        CLOSEFILE handle%

                        self.UpdateLastMessage("Saved")
                        RETURN TRUE

                ELSE
                        RETURN FALSE
                ENDIF
        ENDFUNCTION

        //! Create menu entry
        FUNCTION CreateMenu%:path$,packageName$
        LOCAL handle%

                self.AddMessage("Creating menu directory")

                INC path$,"/menu"
                IF DOESDIREXIST(path$)=FALSE
                        IF CREATEDIR(path$)
                                self.UpdateLastMessage("Created")
                        ELSE
                                self.UpdateLastMessage("Unable to create")
                                RETURN FALSE
                        ENDIF
                ELSE
                        self.UpdateLastMessage("Already Present")
                ENDIF

                handle%=GENFILE()
                IF handle%<0 THEN RETURN FALSE

                IF OPENFILE(handle%,path$+"/"+packageName$,0)
                        WRITESTR handle%,"?package("+packageName$
+"):needs="+CHR$(34)+"X11"+CHR$(34)+"\ \n"
                        WRITESTR handle%,"section="+CHR$(34)+"Applications/Graphics"+CHR$(34)+"\
\n" // Change later
                        WRITESTR handle%,"title="+CHR$(34)+packageName$+CHR$(34)+"\ \n"
                        WRITESTR handle%,"command=/usr/bin/"+packageName$+".linux\ \n"
                        WRITESTR handle%,"icon=/usr/share/pixmaps/"+packageName$+".xpm\n"

                        CLOSEFILE handle%

                        self.UpdateLastMessage("Saved")
                        RETURN TRUE

                ELSE
                        RETURN FALSE
                ENDIF
        ENDFUNCTION

        FUNCTION CreateDocs%:path$,packageName$
        LOCAL handle%

                self.AddMessage("Creating docs directory")

                INC path$,"/docs"
                IF DOESDIREXIST(path$)=FALSE
                        IF CREATEDIR(path$)
                                self.UpdateLastMessage("Created")
                        ELSE
                                self.UpdateLastMessage("Unable to create")
                                RETURN FALSE
                        ENDIF
                ELSE
                        self.UpdateLastMessage("Already Present")
                ENDIF

                INC path$,"/"+packageName$
                IF DOESDIREXIST(path$)=FALSE
                        IF CREATEDIR(path$)
                                self.UpdateLastMessage("Created")
                        ELSE
                                self.UpdateLastMessage("Unable to create")
                                RETURN FALSE
                        ENDIF
                ELSE
                        self.UpdateLastMessage("Already Present")
                ENDIF

                self.AddMessage("Creating copyright file")
                handle%=GENFILE()
                IF handle%<0 THEN RETURN FALSE

                IF OPENFILE(handle%,path$+"/copyright",0)
                        WRITESTR handle%,"Authors: DD\n"     // Change later
                        WRITESTR handle%,"Copyright 2011\n" // Change later
                        CLOSEFILE handle%
```

```
                RETURN TRUE
        ELSE
                RETURN FALSE
        ENDIF

    ENDFUNCTION
ENDTYPE

LOCAL mainProgram AS TMainProgram

LOADFONT "Media/smalfont.png",0
SETFONT 0

mainProgram.Initialise()
mainProgram.CreatePackage()

// ------------------------------- //
// Project: CreateDEBPackage
// Start: Friday, September 30, 2011
// IDE Version: 10.118

// This type is for receiving file information, which includes :
// File (from which the filename can be extracted)
// File icon
// File version
// File short description
// File long description

TYPE TFile
    screenWidth%;screenHeight%
    fileNameText$="fileNameText"; displayFileNameButton$="fileNameButton";    fileName$
    fileIconText$="fileIconText"; displayIconNameButton$="fileIconButton";    icon$
    versionText$="v1";                          version$
    revisionText$="v2";                         revision$
    userRevisionText$="v3";                     userRevision$
    shortDescText$="short";                     shortDescription$
    longDescText$="long";               longDescription$
    sectionText$="section"
    priorityText$="priority"
    fileSize%

    cancelButton$="cancel"
    finishButton$="finish"

    // List of valid sections
    sectionList$[]
    sectionListText$

    // List of valid priorty text
    priorityList$[]
    priorityListText$

    packageName$
    versionString$

    FUNCTION Initialise%:
    LOCAL data$

        GETSCREENSIZE self.screenWidth%,self.screenHeight%

        self.fileName$=""
        self.icon$=""
        self.version$=""
        self.revision$=""
        self.userRevision$=""
        self.shortDescription$=""
        self.longDescription$=""
        self.fileSize%=0

        DIM self.sectionList$[0]
        DIM self.priorityList$[0]

        self.sectionListText$=""
        self.priorityListText$=""

        RESTORE sectionListData
        READ data$
        WHILE data$<>""
            DIMPUSH self.sectionList$[],data$
            INC self.sectionListText$,data$+"|"
            READ data$
            DEBUG "Data : "+data$+"\n"
        WEND

        self.sectionListText$=LEFT$(self.sectionListText$,LEN(self.sectionListText$)-1)

        RESTORE priorityListData
        READ data$
        WHILE data$<>""
            DIMPUSH self.priorityList$[],data$
            INC self.priorityListText$,data$+"|"
```

```
                READ data$
        WEND

        self.priorityListText$=LEFT$(self.priorityListText$,LEN(self.priorityListText$)-1)
        RETURN TRUE
ENDFUNCTION

FUNCTION CreateWindow%:
        DDgui_pushdialog(0,0,self.screenWidth%,self.screenHeight%,TRUE)
        DDgui_set("","TEXT","File Information")
        DDgui_widget("","File Name :",self.screenWidth%-4,0)
        DDgui_singletext(self.fileNameText$,self.fileName$,self.screenWidth%-64)
        DDgui_button(self.displayFileNameButton$,"Select",16,0)

        DDgui_widget("","File Icon (blank for default) :",self.screenWidth%-4,0)
        DDgui_singletext(self.fileIconText$,self.icon$,self.screenWidth%-64)
        DDgui_button(self.displayIconNameButton$,"Select",16,0)

        DDgui_widget("","Section for Program Installation :",self.screenWidth%-140,0)
        DDgui_combo(self.sectionText$,self.sectionListText$,130,0)

        DDgui_widget("","Package Priority :",self.screenWidth%-140,0)
        DDgui_combo(self.priorityText$,self.priorityListText$,130,0)

        DDgui_widget("","File Version (version.revision-user) :",self.screenWidth%-140,0)
        DDgui_singletext(self.versionText$,self.version$,24)
        DDgui_widget("",".",0,0)
        DDgui_singletext(self.revisionText$,self.revision$,24)
        DDgui_widget("","-",0,0)
        DDgui_singletext(self.userRevisionText$,self.userRevision$,24)

        DDgui_widget("","File Short Description :",self.screenWidth%-4,0)
        DDgui_singletext(self.shortDescText$,self.shortDescription$,self.screenWidth%-8)

        DDgui_widget("","File Long Description :",self.screenWidth%-4,0)
        DDgui_text(self.longDescText$,self.longDescription$,self.screenWidth%-8,64)

        DDgui_button(self.cancelButton$,"Cancel",(self.screenWidth%/2)-4,0)
        DDgui_button(self.finishButton$,"Finish",(self.screenWidth%/2)-4,0)

        DDgui_show(FALSE)
ENDFUNCTION

FUNCTION Process%:
        self.CreateWindow()

        WHILE TRUE
                DDgui_show(FALSE)
                IF DDgui_get(self.displayFileNameButton$,"CLICKED")
                        self.SelectProgram()
                ELSEIF DDgui_get(self.displayIconNameButton$,"CLICKED")
                        self.SelectIcon()
                ELSEIF DDgui_get(self.cancelButton$,"CLICKED")
                        DDgui_popdialog()
                        RETURN FALSE
                ELSEIF DDgui_get(self.finishButton$,"CLICKED")
                        IF self.ValidateData()
                                DDgui_popdialog()
                                RETURN TRUE
                        ENDIF
                ENDIF
                SHOWSCREEN
        WEND
ENDFUNCTION

//! Let the user select a .linux program
FUNCTION SelectProgram%:
LOCAL file$

        file$=DDgui_FileDialog$(TRUE,"*.linux")
        DEBUG "File : "+file$+"\n"
        IF file$<>""
                DDgui_set(self.fileNameText$,"TEXT",file$)
        ENDIF
ENDFUNCTION

//! Let the user select a valid icon (which will be converted if needed later on)
FUNCTION SelectIcon%:
LOCAL file$

        file$=DDgui_FileDialog$(TRUE,"*.png|*.bmp|*.xpm")
        IF file$<>""
                DDgui_set(self.fileIconText$,"TEXT",file$)
        ENDIF
ENDFUNCTION

//! Make sure all data is valid
FUNCTION ValidateData%:
LOCAL temp$,error$,linuxExt$=".linux"
LOCAL imgExt$[];       DIMDATA imgExt$[],".png",".bmp",".jpg",".xpm"
LOCAL value%,start%,end%
```

```
		TRY
			// Validate filename
			temp$=DDgui_get$(self.fileNameText$,"TEXT")
			IF temp$="" THEN THROW "No program file present"
			IF DOESFILEEXIST(temp$)=FALSE THEN THROW "The given program file no longer
exists"
			IF RIGHT$(temp$,LEN(linuxExt$))<>linuxExt$ THEN THROW "The file doesn't end
with "+linuxExt$

			// Validate icon
			temp$=DDgui_get$(self.fileIconText$,"TEXT")
			IF temp$<>""
			LOCAL found%

				found%=FALSE
				FOREACH loop$ IN imgExt$[]
					IF RIGHT$(temp$,LEN(loop$))=loop$
						found%=TRUE
						BREAK
					ENDIF
				NEXT

				IF found%=FALSE THEN THROW "The icon graphic is in an unknown format"
			ELSE
				// Can be blank for default icon
				temp$="Media/icon.png"
				DDgui_set(self.fileIconText$,"TEXT",temp$)
			ENDIF

			IF DOESFILEEXIST(temp$)=FALSE THEN THROW "The given icon filename no longer
exists"

			// Validate version information
			value%=INTEGER(DDgui_get$(self.versionText$,"TEXT"))
			DDgui_set(self.versionText$,"TEXT",value%)
			IF value%<0 THEN THROW "Version number is less than 0"

			value%=INTEGER(DDgui_get$(self.revisionText$,"TEXT"))
			DDgui_set(self.revisionText$,"TEXT",value%)
			IF value%<0 THEN THROW "Revision number is less than 0"

			value%=INTEGER(DDgui_get$(self.userRevisionText$,"TEXT"))
			DDgui_set(self.userRevisionText$,"TEXT",value%)
			IF value%<0 THEN THROW "User revision number is less than 0"

			// Validate short description
			temp$=DDgui_get$(self.shortDescText$,"TEXT")
			IF temp$="" THEN THROW "There is no short description"

			// Validate long description
			temp$=DDgui_get$(self.longDescText$,"TEXT")
			IF temp$="" THEN THROW "There is no long description"

			// Extract filename
			self.fileName$=DDgui_get$(self.fileNameText$,"TEXT")

			// Extract project name
			start%=LEN(self.fileName$)-1
			WHILE MID$(self.fileName$,start%,1)<>"/" AND start%>=0
				DEC start%
			WEND

			INC start%
			// Find the extension
			end%=start%
			WHILE MID$(self.fileName$,end%,1)<>"." AND end%<LEN(self.fileName$)
				INC end%
			WEND

			self.packageName$=LCASE$(MID$(self.fileName$,start%,end%-start%))
			self.fileSize%=GETFILESIZE(self.fileName$)
			self.versionString$=DDgui_get$(self.versionText$,"TEXT")+"."+DDgui_get$
(self.revisionText$,"TEXT")+"-"+DDgui_get$(self.userRevisionText$,"TEXT")
			RETURN TRUE
		CATCH error$
			DDgui_msg(error$,FALSE,"* Error *")
			RETURN FALSE
		FINALLY
	ENDFUNCTION

	FUNCTION ReturnPackageName$:
		RETURN self.packageName$
	ENDFUNCTION

	FUNCTION ReturnFileSize%:
		RETURN self.fileSize%
	ENDFUNCTION

	FUNCTION ReturnVersionString$:
		RETURN self.versionString$
```

```
        ENDFUNCTION
ENDTYPE

STARTDATA sectionListData:
        DATA "admin", "cli-mono", "comm", "database", "devel", "debug", "doc", "editors",
"electronics", "embedded", "fonts", "games", "gnome", "graphics"
        DATA "gnu-r", "gnustep", "hamradio", "haskell", "httpd", "interpreters", "java", "kde",
"kernel", "libs", "libdevel", "lisp", "localization"
        DATA "mail", "math", "misc", "net", "news", "ocaml", "oldlibs", "otherosfs", "perl", "php",
"python", "ruby", "science", "shells", "sound"
        DATA "tex", "text", "utils", "vcs", "video", "web", "x11", "xfce", "zope"
        DATA ""
ENDDATA

STARTDATA priorityListData:
        DATA "required","important","standard","optional","extra"
        DATA ""
ENDDATA
```

SaveAsXPM

This was going to be used with the CreateDEBPackage routine to save a bitmap as a XPM file

```
// -------------------------------- //
// Project: SaveAsXPM
// Start: Sunday, October 02, 2011
// IDE Version: 10.118

TYPE tMap
        colour%
        value$
ENDTYPE

TYPE TXPM
        xpm_c$ =        " .oX#-*:=@$%&?!~abcdefghijkmnpqrstuvwxyzABCDEFGHJKLMNOPQRSTUVWYZ123456789";
        hex$="0123456789ABCDEF"
        max_c% =        0
        coltable[] AS tMap

        FUNCTION Initialise%:
                self.max_c%=LEN(self.xpm_c$)
                RETURN TRUE
        ENDFUNCTION

        FUNCTION SaveXPMFile%:inFile$,outFile$,transparent%
        LOCAL sprite%,sprWidth%,sprHeight%,pos%,temp$
        LOCAL sprArray%[]
        LOCAL handle%
        LOCAL str$,length%

                IF DOESFILEEXIST(inFile$)=FALSE THEN RETURN FALSE

                sprite%=GENSPRITE()
                IF sprite%<0 THEN RETURN FALSE

                LOADSPRITE inFile$,sprite%
                GETSPRITESIZE sprite%,sprWidth%,sprHeight%
                IF sprWidth%=0 OR sprHeight%=0 THEN RETURN FALSE

                DIM self.coltable[0]

                IF SPRITE2MEM(sprArray%[],sprite%)
                        FOR loop%=0 TO BOUNDS(sprArray%[],0)-1
                                self.Add(sprArray%[loop%])
                        NEXT

                        handle%=GENFILE()
                        IF handle%<0 THEN RETURN FALSE

                        IF OPENFILE(handle%,outFile$,0)
                                WRITESTR handle%,"/* XPM */\n"
                                WRITESTR handle%,"static char *plaid[] = {\n"
                                WRITESTR handle%,"/* plaid pixmap\n"
                                WRITESTR handle%,"* width height ncolors chars_per_pixel */\n"
                                WRITESTR handle%,CHR$(34)+sprWidth%+" "+sprHeight%+"
"+BOUNDS(self.coltable[],0)+" "

                                length%=1

                                FOR loop%=5 TO 1 STEP -1
                                        IF BOUNDS(self.coltable%[],0)>POW(self.max_c%,loop%)
                                                length%=loop%+1
                                                BREAK
                                        ENDIF
                                NEXT

                                WRITESTR handle%,length%
                                WRITESTR handle%,CHR$(34)+",\n"
                                WRITESTR handle%,"/* colours */\n"

                                FOR loop%=0 TO BOUNDS(self.coltable[],0)-1
                                        self.coltable[loop%].value$=self.decToBase$(loop%,self.max_c
%,self.xpm_c$,length%)
                                        str$=CHR$(34)+self.coltable[loop%].value$+" c "
                                        IF self.coltable[loop%].colour%=transparent%
                                                INC str$,"none"
                                        ELSE
                                                INC str$,self.decToBase$(self.coltable[loop%].colour
%,16,self.hex$,8)
                                        ENDIF
                                        INC str$,CHR$(34)

                                        WRITESTR handle%,str$+"\n"
                                NEXT
```

```
                        WRITESTR handle%,"/* pixels */\n"

                        FOR y%=0 TO sprHeight%-1
                                temp$=""

                                FOR x%=0 TO sprWidth%-1
                                        INC temp$,self.Find$(sprArray%[x%+y%*sprWidth%])
                                NEXT

                                WRITESTR handle%,CHR$(34)+temp$+CHR$(34)+",\n"
                        NEXT

                        WRITESTR handle%,"};"

                        CLOSEFILE handle%
                        RETURN TRUE
                ENDIF
        ENDIF

        RETURN FALSE
ENDFUNCTION

FUNCTION Add%:colour%
LOCAL loop AS tMap

        FOREACH loop IN self.coltable[]
                IF loop.colour%=colour% THEN RETURN FALSE // Already present
        NEXT

        loop.colour%=colour%
        loop.value$=""

        DIMPUSH self.coltable[],loop
        RETURN TRUE
ENDFUNCTION

FUNCTION Find$:colour%
        FOREACH loop IN self.coltable[]
                IF loop.colour%=colour% THEN RETURN loop.value$
        NEXT

        RETURN "??"
ENDFUNCTION

FUNCTION decToBase$:value,base%,str$,exactLength%=0
INLINE
unsigned long decimal;
DGStr result;

        decimal=(unsigned long) value;

        result=DGStr("");

        if (decimal>0)
        {
                do {
                        result=MID_Str(str_Str,decimal % base,1)+result;
                        decimal/=base;
                } while (decimal!=0);
        }

        if (exactLength>0)
        {
                while (LEN(result)<exactLength)
                {
                        result=MID_Str(str_Str,0,1)+result;
                }
        }

        return DGStr(result);
ENDINLINE
ENDFUNCTION
ENDTYPE

LOCAL xpm AS TXPM

DEBUG "X\n"
xpm.Initialise()
DEBUG "Result : "+xpm.SaveXPMFile("Media/icon.png","output.xpm",RGB(0,0,0))+"\n"
```

GLBasicommands To Text

Convert all GLBasic commands to a text file. Only useful for syntax highlighting

---

```
CONSTANT OUTPUTFILE$="Commands.txt"
CONSTANT INPUTFILE$="C:/Program Files (x86)/GLBasic/Compiler/platform/prototypes.txt"
CONSTANT OHANDLE%     =       2
CONSTANT IHANDLE%     =       1

LOCAL line$,returnParam$,text$,lPos%,rPos%,diff%
LOCAL array$[]; DIM array$[0]
LOCAL moo%=12

IF OPENFILE(IHANDLE%,INPUTFILE$,1)
        IF OPENFILE(OHANDLE%,OUTPUTFILE$,0)
                READLINE IHANDLE%,line$
                WHILE ENDOFFILE(IHANDLE%)=FALSE
                        lPos%=INSTR(line$,":")
                        rPos%=INSTR(line$,":",lPos%+1)

                        IF lPos%<>-1 AND rPos%<>-1
                                returnParam$=LEFT$(line$,lPos%-1)
                                diff%=rPos%-lPos%
                                text$=MID$(line$,lPos%+1,diff%-1)
                                IF text$<>"NAME"
                                        DIMPUSH array$[],text$
                                ENDIF
                        ENDIF

                        READLINE IHANDLE%,line$
                WEND

                SORTARRAY array$[],0

                FOREACH line$ IN array$[]
                        WRITELINE OHANDLE%,line$
                NEXT

                CLOSEFILE IHANDLE%
                CLOSEFILE OHANDLE%
        ENDIF
ENDIF

END

SUB GLB_ON_QUIT:
        DEBUG "Here\n"
ENDSUB
```

Graphic

Simple graphics routine

```
CONSTANT STEPSIZE%    =       4

TYPE tPixel
        x%
        y%
        colour%
ENDTYPE

TYPE tClock
        size
        angle
        speed
        alpha
ENDTYPE

LOCAL pixel[] AS tPixel
LOCAL clock[] AS tClock

LOCAL mx%,sW%,sH%
LOCAL alpha,fps,showfps,delay,dtime

GETSCREENSIZE sW%,sH%
LIMITFPS -1

mx%=sW%/4

DIM pixel[mx%]
DIM clock[9]

SEEDRND GETTIMERALL()
FOR loop%=0 TO mx%-1
        pixel[loop%].x%=loop%*4
        pixel[loop%].y%=sH%/2
        pixel[loop%].colour%=RGB(128,128,128)
NEXT

alpha=-0.1
FOR loop%=0 TO 8
        clock[loop%].size=RND(MIN(sW%,sH%)/2)+8;    clock[loop%].angle=RND(359);  clock[loop
%].alpha=alpha
        clock[loop%].speed=0.0
        WHILE clock[loop%].speed=0.0
                clock[loop%].speed=(RND(100)-50)/20.0
        WEND

        DEC alpha,0.1
NEXT

fps=0.0
WHILE TRUE
        PRINT "FPS : "+showfps,0,0
        FOR loop%=0 TO 8
                ALPHAMODE clock[loop%].alpha
                DRAWLINE (sW%/2),(sH%/2),_
                                  (sW%/2)+(clock[loop%].size%*SIN(clock[loop%].angle)), _
                                  (sH%/2)+(clock[loop%].size%*COS(clock[loop%].angle)), _
                                  RGB(255,255,0)

                //INC clock[loop%].angle,clock[loop%].speed
                clock[loop%].angle=IIF(clock[loop%].angle>=360.0,clock[loop%].angle-360.0,clock[loop
%].angle+clock[loop%].speed)
                //IF clock[loop%].angle>=360.0 THEN DEC clock[loop%].angle,360.0
        NEXT

        FOR loop%=0 TO mx%-2
                ALPHAMODE 0.0
                DRAWLINE pixel[loop%].x%,pixel[loop%].y%,pixel[loop%+1].x%,pixel[loop%+1].y
%,pixel[loop%].colour%
                ALPHAMODE -0.8
                DRAWLINE sW%-pixel[loop%].x%,sH%-pixel[loop%].y%,sW%-pixel[loop%+1].x%,sH%-pixel[loop
%+1].y%,RGB(128,128,128)
        NEXT

        SHOWSCREEN

        FOR loop%=1 TO mx%-1
                pixel[loop%-1].y%=pixel[loop%].y%
                pixel[loop%-1].colour%=pixel[loop%].colour%
        NEXT

        INC pixel[mx%-1].y%,(RND(10)-5)*STEPSIZE%
        IF pixel[mx%-1].y%<0 THEN pixel[mx%-1].y%=0
```

```
        IF pixel[mx%-1].y%>sH% THEN pixel[mx%-1].y%=sH%
        pixel[mx%-1].colour%=RGB(RND(255),RND(255),RND(255))

        dtime = GETTIMER()
         fps = ((1000/dtime)+fps)/2
         delay=delay+dtime
         IF delay>500 // 1/2 sec
          delay=0
          showfps=fps
         ENDIF
WEND
```

Messages

A messaging system for passing messages between functions. Of no real use

```
INLINE
   #define min(a,b) (a<b ? a : b)
ENDINLINE

FUNCTION Hash%:text$
INLINE
long result,temp,counter;

   result=0;
   for (counter=0; counter<min(text_Str.len(),16); counter++)
   {
      result=(result<<4)+((char) text_Str[counter]);
      temp=result & 0xF000;
      if (temp)
      {
         result=result^(temp>>12);
      }

      result=(result & (temp^0xFFFF)) & 0xFFFF;

   }

   return (DGNat) result;
ENDINLINE
ENDFUNCTION

// ------------------------------- //
// Project: Messages
// Start: Sunday, July 04, 2010
// IDE Version: 8.006
TYPE tMessage
        fromHashValue%
        toHashValue%
        data$
ENDTYPE

TYPE tTypeList
        hashValue%
        typeName$
ENDTYPE

TYPE TMessage
        NOT_VALID%     =       -1
        typeList[] AS tTypeList
        messages[] AS tMessage

        FUNCTION TMessage_Initialise%:
                DIM self.typeList[0]
        ENDFUNCTION

        FUNCTION TMessage_RegisterType%:typeName$
        LOCAL loop AS tTypeList
        LOCAL found%,hashValue%

                found%=FALSE
                hashValue%=Hash(typeName$)
                IF TMessage_LookForHash(hashValue%)=self.NOT_VALID%
                        // Not found, so add this hash to the list
                        loop.typeName$=typeName$
                        loop.hashValue%=hashValue%
                        DIMPUSH self.typeList[],loop
                        SORTARRAY self.typeList[],0
                        RETURN hashValue%
                ELSE
                        RETURN self.NOT_VALID%
                ENDIF
        ENDFUNCTION

        FUNCTION TMessage_SendMessage%:fromHash%,toHash%,data$
        LOCAL fromIndex%,toIndex%
        LOCAL message AS tMessage

                fromIndex%=TMessage_LookForHash(fromHash%)
                toIndex%=TMessage_LookForHash(toHash%)
                IF fromIndex%>self.NOT_VALID% AND toIndex%>self.NOT_VALID%
                        message.fromHashValue%=fromHash%
                        message.toHashValue%=toHash%
                        message.data$=data$
                        DIMPUSH self.messages[],message
                        RETURN TRUE
                ELSE
```

```
                RETURN FALSE
        ENDIF
ENDFUNCTION

//! Send debug information to the output window
FUNCTION TMessage_Debug%:
LOCAL loop AS tTypeList

        DEBUG "Number of registered types : "+BOUNDS(self.typeList[],0)+"\n"
        FOREACH loop IN self.typeList[]
                DEBUG "("+loop.hashValue%+" - "+loop.typeName$+")\n"
        NEXT
        DEBUG "Messages Outstanding : "+BOUNDS(self.messages[],0)+"\n"
ENDFUNCTION

//! Send a message to the destination type, using its registered name.  This could be very
slow
FUNCTION TMessage_SendMessage_Text%:fromHash%,toType$,data$
LOCAL fromIndex%,toIndex%
LOCAL message AS tMessage

        fromIndex%=TMessage_LookForHash(fromHash%)
        IF fromIndex%>self.NOT_VALID%
                toIndex%=TMessage_LookForName(toType$)
                IF toIndex%>self.NOT_VALID%
                        message.fromHashValue%=fromHash%
                        message.toHashValue%=self.typeList[toIndex%].hashValue%
                        message.data$=data$
                        DIMPUSH self.messages[],message
                        RETURN TRUE
                ENDIF
        ENDIF
        RETURN FALSE
ENDFUNCTION

FUNCTION TMessage_SendMessage_All%:fromHash%,data$,includeFrom%=FALSE
LOCAL fromIndex%
LOCAL message AS tMessage
LOCAL loop AS tTypeList

        fromIndex%=TMessage_LookForHash(fromHash%)
        IF fromIndex%>=0
                message.fromHashValue%=fromHash%
                message.data$=data$

                FOREACH loop IN self.typeList[]
                        IF (loop.hashValue%=fromHash% AND includeFrom%=TRUE) OR
(loop.hashValue%<>fromHash%)
                                message.toHashValue%=loop.hashValue%

                                DIMPUSH self.messages[],message
                        ENDIF
                NEXT

                RETURN TRUE
        ENDIF

        RETURN FALSE
ENDFUNCTION

//! Send to the first ordered type - may not be in the order added by the user
FUNCTION TMessage_SendMessage_First%:fromHash%,data$
LOCAL fromIndex%,toIndex%
LOCAL message AS tMessage

        fromIndex%=TMessage_LookForHash(fromHash%)
        toIndex%=0
        IF fromIndex%>=0
                message.fromHashValue%=fromHash%
                message.toHashValue%=self.typeList[toIndex%].hashValue%
                message.data$=data$
                DIMPUSH self.messages[],message
                RETURN TRUE
        ENDIF
ENDFUNCTION

//! Send to the last ordered type - may not be in the order added by the user
FUNCTION TMessage_SendMessage_Last%:fromHash%,data$
LOCAL fromIndex%,toIndex%
LOCAL message AS tMessage

        fromIndex%=TMessage_LookForHash(fromHash%)
        toIndex%=BOUNDS(self.typeList[],0)-1
        IF fromIndex%>=0
                message.fromHashValue%=fromHash%
                message.toHashValue%=self.typeList[toIndex%].hashValue%
                message.data$=data$
                DIMPUSH self.messages[],message
                RETURN TRUE
        ENDIF
ENDFUNCTION
```