Multi-Platform Graphics Library ver 2.0

1 Introduction

The purpose of Multi-Platform Graphics library is to provide high-performance & flexible graphics to high-level programmers. So that they do not have to build their own graphics routines for each target-platforms.

2 Requirement

Hardware :

Minimum - 486 DX 33+ Mhz, 4 MB RAM, VGA Graphics Card.
Recomended - Pentium 90+ Mhz, 8-16 MB RAM, VESA compatible SVGA.

Supported Operating System :

DOS, 32-Bit DOS, Windows 3.1, Windows 95, Windows NT, OS/2 3.0, Unix with X Windows system.

Supported Compilers :

Watcom C/C++ 10.0+, Visual C/C++ 2.0+, Borland C/C++ 4.0+, GNU C/C++ Compiler.

3 Source Code

The source code with gfx prefix is the C++ modules for this library. The source code with lgfx prefix is the Low-Level routine for this library.

GFX sources : Extension C++ or CXX.

gfxadst - Abstract Data Structure class.
gfxanim - Animated sprites class.
gfxbmp - Windows BMP file tool.
gfxcltbl - Color Look-Up table.
gfxddraw - Game SDK's DirectDraw driver.
gfxdev - Device handling class.
gfxdisp - Display driver class.
gfxdpmi - DPMI handling class.
gfxdrv - Driver Level class.
gfxerror - Error handling class.
gfxfiles - File handling class.
gfxflic - FLI/FLC file tool.

gfxgif - Compuserv GIF file tool.
gfxgdi - Windows GDI driver.
gfxgpi - OS/2 GPI driver.
gfxgraf - Graphics System class.
gfxifile - Image file too.
gfximage - Image (Virtual Screen) class.
gfxinput - Input device handler.
gfxllist - Linked-List class.
gfxmodes - Graphics Modes.
gfxobj - Base Object of evrything.
gfxpal - Color Palette class.
gfxpcx - PCX file tool.
gfxsddrv - Sound driver class.
gfxsys - System handling class.
gfxtypes - Type definitions.
gfxvga - VGA driver.
gfxvsa - VESA driver.
gfxwinsd - Windows Sound driver.
gfxxwin - X Windows driver.

Low-Level sources : Extension ASM or CPP or CXX.

lgfxdib? - Device Independent graphics routines.
lgfxvga? - VGA specific graphics routines.
lgfxvsa? - VESA specific graphics routines.

* Note - ? will be replaced by one of the following characters.

b - For Borland compiler.
g - For GNU compiler.
v - For Visual C/C++ compiler.
w - For Watcom C/C++ compiler.

Use appropriate low-level source for your compiler.

4 Setup for MPGFX Library.

Follow these steps to start programming with MPGFX.

1 - Create directory called "MPGFX" or something you like.
2 - Under this directory, create "source" and "include".
3 - From source disks, copy all the files with cpp,cxx,asm extensions to the directory mpgfx/source.
4 - From source disks, copy all the files with h

extensions to the directory mpgfx/include.

5 - Include all the gfx?.cpp (or cxx) modules and appropriate low-level source files to your project or makefile.

6 - You must define a macro before start compiling. See below.

Macros for Compiler :

__FORBORLAND__ - Compiling for Borland.
__FORGCC__ - Compiling for GCC.
__FORVISUAL__ - Compiling for Visual.
__FORWATCOM__ - Compiling for Watcom.

Macros for Target Operating System :

__FORDOS__ - 16 Bit DOS.
__FORDOS4GW__ - 32 Bit DOS 4GW Extender.
__FORWIN16__ - 16 Bit Windows.
__FORWIN32__ - 32 Bit Windows. (Windows 95, NT, Win32s)
__FOROS2__ - 32 Bit OS/2.
__FORUNIX__ - Unix with X Wndows.

Macros for Tools or class library :

__FORMFC__ - Make it compatible to MFC class library.
__FOROWL__ - Make it compatible to OWL class library.
__FORMOTIF__ - Make it compatible to Motil Toolkit.
__FORGDK__ - Make it compatible to Game SDK.

* Exmaple - If you want to compile for Windows 95 using Watcom C/C++ with MFC library, define :

__FORWATCOM__ __FORWIN32__ __FORMFC__

7 - Make mpgfx/include directory visible to your project.

8 - Include stdgfx.h and mpgfx.h file in your source code.

5 Programming with MPGFX

5-1 Types

BYTE - Unsigned character. 1 Bytes
CHAR - Character. 1 Byte.

WORD - Unsigned Short. 2 Bytes.
SHORT - Short. 2 Bytes.

INT - Integer. (2/4 Bytes in 16/32 Bit Platform.)
UINT - Unisgned Integer. (2/4 Bytes in 16/32 Bit Platform.)

LONG - Long Integer. 4 Bytes.
ULONG - Unsigned Long Integer. 4 Bytes.
DWORD - Unsigned Long Integer. 4 Bytes.

STRING - Pointer to character.
BOOLEAN - Boolean, (TRUE/FALSE) or (SUCCESS/FAILURE).
HDISPLAY - Displayable object. (Source and destination of graphic operations.)

5-2 Constants

TRUE - 1
FALSE - 0
SUCCESS - 1
FAILURE - 0

HVGA - Used as HDISPLAY of VGA(SVGA) memory. (Only used in DOS)

M320x200x256 - Used for setting up VGA mode 320x200 by 256 Colors.
M640x400x256 - Used for setting up VESA mode 640x400 by 256 Colors.
M640x480x256 - Used for setting up VESA mode 640x480 by 256 Colors.
M800x600x256 - Used for setting up VESA mode 800x600 by 256 Colors.
M1024x768x256 - Used for setting up VESA mode 1024x768 by 256 Colors.

5-3 Global Object

Following objects are always visible and accessible to high-level programmers.

GRAFIX Grafix - Graphics System.

INPUT Input   - Input Handling System.
SOUND Sound   - Sound System.

5-4 Programming with Grafix

Before any graphics operations, you must setup display by calling

```
Grafix.SetDisplay ( DISPLAYDATA *Data );
```

Data is a structure of DISPLAYDATA type.

```
struct DISPLAYDATA
  {
   LONG Mode;
   BOOLEAN UseDirectDraw;
   HWINDOW hMainWindow;
   #if defined (__FORUNIX__)
     Display *MainXDisplay;
   #endif
  }; // End of DISPLAYDATA
```

Mode - One of the M*** constant shown in 5-2 section.
UseDirectDraw - Set true if you want to use DirectDraw for Windows 95.
hMainWindow - Set this value to your MainWindow.
MainXDisplay - Set this value to your X Windows's Display.

Just before your program terminate, you must call

```
Grafix.ResetDisplay ();
```

To reinitialize the graphics.

Example 1.

Following is the example of using MPGFX in DOS. It will switch to VGA 320x200 256 color mode and wait for you keyboard then reset graphics to normal and quit. Assuming you have finished setting up the steps of section 4.

```
//********** Example 1. ***********//

#include "stdgfx.h"
#include "mpgfx.h"

void main ()
```

```
{
  DISPLAYDATA Data;
  Data.Mode = M320x200x256;
  Grafix.SetDisplay ( &Data );
  getch ();
  Grafix.ResetDisplay ();
} // End of main
```

5-4-1 Graphics Destination

There are two types of object that can receive Graphics Operations from Grafix. HDISPLAY and IMAGE.

5-4-1-1 HDISPLAY

HDISPLAY is a handle to Screen. It represent different thing, depending on which target-platform you are compiling for. For example, for Windows, HDISPLAY is equivalent for HDC (Display Context), in OS/2 HPS (Presentation Space) and in X Windows system it is Drawable (Window or Pixmap).

5-4-1-2 IMAGE

IMAGE is a class that represent Virtual Screen. (A screen resides in Memory). Following interfaces are available from IMAGE class.

IMAGE ()
- Default Constructor.

IMAGE (INT Orientation)
- Constructor with specified orientation.
  Orientation can be one of IMAGE_TOPDOWN or
  IMAGE_BOTTOMUP.

BOOLEAN Create (LONG Format, LONG Width,
LONG Height)
- Allocates the Buffer for specified format and size.
  Format is one of IMAGE_8BIT, IMAGE-16BIT, IMAGE_24BIT.
  If creation is successlfull, it returns SUCCESS,
  otherwise, returns FAILURE.

- Note : Currently only IMAGE_8BIT is available.

LONG GetWidth ()
- Returns the Width of the image in # of pixles. If buffer
  has not been allocated, it returns 0.

LONG GetHeight ()
- Returns the Height of the image in # of pixles. If buffer has not been allocated, it returns 0.

BYTE* GetBuffer ()
- Returns the pointer to allocated buffer. If it has not been allocated, returns NULL.

BYTE* SetOffset (LONG x, LONG y)
- Returns the pointer to buffer with specified coordinate.

5-4-1-3 Screen coordinate

All the coordinate in Grafix system is in unit of pixels.
The top-left corner is always (0,0) and bottom-right corner is (Width-1,Height-1).

5-4-2 Grafix Operations

Foloowing Grafix operations are available from Grafix.

BOOLEAN SetDisplay (DISPLAYDATA *Data);
- See section 5-4.

VOID ResetDisplay ();
- See section 5-4.

VOID ClearDisplay (HDISPLAY hDisplay, LONG Color);
- Clears scpecified HDISPLAY by Color.

hDisplay - Destination to be cleared.
Color - Color to be filled.

VOID SetScaleFactor (LONG MagH, LONG DivH, LONG MagV,
LONG DivV);
- Set the factor for scale image function.

MagH - Magnifying factor for horizontal direction.
DivH - Division factor for horizontal direction.
MagH - Magnifying factor for vertical direction.
DivH - Division factor for horizontal direction.

// Image Functions
VOID CopyImage (IMAGE *Src, LONG Sx, LONG Sy,
LONG Wd, LONG Ht,
IMAGE *Dest, LONG Cx, LONG Cy);
- Copy part of Src into Dest.

Src - Source IMAGE.
Sx - Left corner of source image to copy.
Sy - Top corner of source image to copy.
Wd - Width of source image to copy.
Ht - Height of source image to copy.
Dest - Destination IMAGE.
Cx - Left corner of destination image to receive copied block.
Cy - Top corner of destination image to receive copied block.

VOID ScaleImage (IMAGE *Src, LONG Sx, LONG Sy,
LONG Wd, LONG Ht,
IMAGE *Dest, LONG Cx, LONG Cy);
- Scale Src for MagH/DivH for horizontaly and MagV/DivV for verticaly and display to Dest.

Src - Source IMAGE.
Sx - Left corner of source image to copy.
Sy - Top corner of source image to copy.
Wd - Width of source image to copy.
Ht - Height of source image to copy.
Dest - Destination IMAGE.
Cx - Left corner of destination image to receive copied block.
Cy - Top corner of destination image to receive copied block.

VOID RotateImage (IMAGE *Src, LONG Sx, LONG Sy,
LONG Wd, LONG Ht,
LONG CenterX, LONG CenterY, float Angle,
IMAGE *Dest, LONG Cx, LONG Cy);
- Rotate Src for specified angle and display to Dest.

Src - Source IMAGE.
Sx - Left corner of source image to copy.
Sy - Top corner of source image to copy.
Wd - Width of source image to copy.
Ht - Height of source image to copy.
CenterX - Center X coord as a rotation center.
CenterY - Center Y coord as a rotation center.
Angle - Angle to rotate. (Degrees)
Dest - Destination IMAGE.
Cx - Left corner of destination image to receive copied block.
Cy - Top corner of destination image to receive copied block.

```
VOID DisplayImage ( HDISPLAY hDisplay, IMAGE *Image,
                    LONG Sx, LONG Sy,
                    LONG Wd, LONG Ht, LONG Cx, LONG Cy );
```
- Display part of Src into hDisplay.

hDisplay - Destination to receive src image.
Src - Source IMAGE.
Sx - Left corner of source image to copy.
Sy - Top corner of source image to copy.
Wd - Width of source image to copy.
Ht - Height of source image to copy.
Cx - Left corner of destination image to receive copied block.
Cy - Top corner of destination image to receive copied block.

```
VOID ConvertImage ( IMAGE *Image, COLORTABLE *MatchTable );
```
- Convert all the pixel value of Image using MatchTable.

```
BOOLEAN LoadImage ( STRING FileName, IMAGE *Image,
                    RGBPALETTE *Pal );
```
- Load a Image and palette from a file "FileName". Currently Windows BMP, Compuserv GIF and Zsoft PCX are supported. If loading succeeds, it returns SUCCESS, returns FAILURE otherwise.

FileName - File to be loaded.
Image - It will receive the image from this file.
Pal - RGB Palette.

```
BOOLEAN SaveImage ( STRING FileName, SHORT ImageType,
                    IMAGE *Image, LONG Sx, LONG Sy,
                    LONG Wd, LONG Ht, RGBPALETTE *Pal );
```
- Save a Image and palette to a file "FileName". Currently Windows BMP, Compuserv GIF and Zsoft PCX are supported.

FileName - File to be loaded.
ImageType - One of the following constant.
BMPFILE - Windows BMP.
GIFFILE - Compuserv GIF.
PCXFILE - ZSoft PCX.
Image - It will receive the image from this file.
Pal - RGB Palette.

```
// Animation Functions
VOID PlayFLIC ( FLICFILE *Flic, RECTANGLE Region,
                LONG Cx, LONG Cy, LONG StartFrame,
```

LONG EndFrame, LONG Flags,
FLICCallback Callback);
- Plays a FLI/FLC animation with specified parameters.

Flic - FLIC class.
Region - RECTANGLE structure to specify which part of
this FLIC image is supposed to be displayed.
Cx - X coordinate to copy image into.
Cy - Y coordinate to copy image into.
StartFrame - Starting frame number.
EndFrame - Ending frame number.
Flags - Combination of following values.

FLIC_LOOP - Loop the animation.
FLIC_NOADVANCE - Do not advance the
frame.
FLIC_ALL - Play all frames.
Callback - Callback function.

// Palette Functions
VOID SetPalette (HDISPLAY hDisplay, RGBPALETTE *Pal);
- Sets a palette of specified HDISPLAY.

hDisplay - Destination to receive palette.
Pal - Target Palette to change to.

VOID GetPalette (HDISPLAY hDisplay, RGBPALETTE *Pal);
- Reads a palette from specified HDISPLAY.

hDisplay - Source to retriev palette.
Pal - Palette to receive.

VOID ClearPalette (HDISPLAY hDisplay, BYTE R, BYTE G,
BYTE B);
- Clears a palette with specified R,G,B values.

Pal - Palette to be changed.

VOID FadePalette (HDISPLAY hDisplay, RGBPALETTE *Pal,
LONG Direction, LONG NumSteps);
- Fades a palette of specified hDisplay.

hDisplay - Target HDISPLAY to be faded.
Pal - Palette to be changed.
Direction - One of following value.

PAL_FADE_IN - Palettes fades in from dark

to bright.
PAL_FADE_OUT - Palettes fades in from
bright to dark.
NumSteps - How many steps to fade from start to end.

BOOLEAN SavePalette (STRING FileName, RGBPALETTE *Pal,
LONG PalFormat);
- Saves a palette to disk with “FileName”.

FileName - Filename to be saved.
Pal - Palette to save.
PalFormat - One of the following format.

PAL_MSWIN - MS Windows RIFF file format.
PAL_PSP - Paint Shop Pro format.
PAL_COREL - Corel Draw format.

- Note : PAL_COREL is currently not supported.

BOOLEAN LoadPalette (STRING FileName, RGBPALETTE *Pal);
- Load a palette from disk with “FileName”.

FileName - Filename to be loaded.
Pal - Palette to be loaded.

// Drawing Functions
VOID DrawPixel (IMAGE *Dest, LONG x, LONG y);
VOID DrawPixel (HDISPLAY hDisplay, LONG x, LONG y);
- Draws a pixel with Grafix.FGColor.

Dest,hDisplay - Destination.
x - X coord to draw pixel to.
y - Y coord to draw pixel to.

VOID DrawLine (IMAGE *Dest, LONG x1, LONG y1, LONG x2,
LONG y2);
VOID DrawLine (HDISPLAY hDisplay, LONG x1, LONG y1,
LONG x2, LONG y2);
- Draws a line with Grafix.FGColor.

Dest,hDisplay - Destination.
x1 - X starting coord to draw line.
y1 - Y starting coord to draw line.
x2 - X ending coord to draw line.
y2 - Y ending coord to draw line.

VOID DrawRect (IMAGE *Dest, LONG x1, LONG y1, LONG x2,

LONG y2);
VOID DrawRect (HDISPLAY hDisplay, LONG x1, LONG y1,
LONG x2, LONG y2);
- Draws a rectangle with Grafix.FGColor.

Dest,hDisplay - Destination.
x1 - Left coord of rectangle.
y1 - Top coord of rectangle.
x2 - Right coord of rectangle.
y2 - Bottom coord of rectangle.

VOID FillRect (IMAGE *Dest, LONG x1, LONG y1, LONG x2,
LONG y2);
VOID FillRect (HDISPLAY hDisplay, LONG x1, LONG y1,
LONG x2, LONG y2);
- Fills a rectangle with Grafix.BGColor.

Dest,hDisplay - Destination.
x1 - Left coord of rectangle.
y1 - Top coord of rectangle.
x2 - Right coord of rectangle.
y2 - Bottom coord of rectangle.

VOID DrawEllipse (IMAGE *Dest, LONG Cx, LONG Cy,
LONG Rx, LONG Ry);
VOID DrawEllipse (HDISPLAY hDisplay, LONG Cx, LONG Cy,
LONG Rx, LONG Ry);
- Draws an ellipse with Grafix.FGColor.

Dest,hDisplay - Destination.
Cx - X coord of center of ellipse.
Cy - Y coord of center of ellipse.
Rx - X Radius.
Ry - Y Radius.

VOID FillEllipse (IMAGE *Dest, LONG Cx, LONG Cy,
LONG Rx, LONG Ry);
VOID FillEllipse (HDISPLAY hDisplay, LONG Cx, LONG Cy,
LONG Rx, LONG Ry);
- Fills an ellipse with Grafix.BGColor.

Dest,hDisplay - Destination.
Cx - X coord of center of ellipse.
Cy - Y coord of center of ellipse.
Rx - X Radius.
Ry - Y Radius.

VOID DrawPolygon (IMAGE *Dest, POLYGON *Poly);

- Draws a polygon to dest with Grafix.FGColor.

Dest - Destination.
Poly - Polygon to drawn.

VOID FillPolygon (IMAGE *Dest, POLYGON *Poly);
- Fills a polygon to dest with Grafix.BGColor.

Dest - Destination.
Poly - Polygon to drawn.

Note : Polygon must be a convex polygon.

VOID DrawText (HDISPLAY hDisplay, STRING Text, LONG x,
LONG y);
- Draw a text to destination.

hDisplay - Destination.
Text - String to be displayed.
x - X coord to display text.
y - Y coord to display text.

LONG GetWidth (HDISPLAY hDisplay);
- Get current width of HDISPLAY.

hDisplay - HDISPLAY.

LONG GetHeight (HDISPLAY hDisplay);
- Get current height of HDISPLAY.

hDisplay - HDISPLAY.

VOID WaitForRetrace (LONG Count);
VOID WaitForRetrace ();
VOID WaitForRetraceEnd ();
- Wait for vertical retrace.

Count - Number of retrace to wait.

Note : Only available for DOS applications.

Example 2.

Following is the example of using MPGFX in DOS. It will switch to VGA 320x200 256 color mode and demonstrate some graphics operations then reset graphics to normal and quit. Assuming you have finished setting up the steps of section 4.

```
//********** Example 2. ***********//

#include "stdgfx.h"
#include "mpgfx.h"

void main ()
  {
    IMAGE *Image1 = new IMAGE ();
    IMAGE *Image2 = new IMAGE ( IMAGE_BOTTOMUP );
    RGBPALETTE *Pal = new RGBPALETTE ();

    Image1->Create ( IMAGE_8BIT, 320, 200 );
    Grafix.LoadImage ( "test.pcx", Image, Pal );

    DISPLAYDATA Data;
    Data.Mode = M320x200x256;
    Grafix.SetDisplay ( &Data );
    Grafix.SetPalette ( HVGA, Pal );

    Grafix.FGColor = 25; // Set Foreground color to 25

    Grafix.DrawLine ( HVGA, 10, 20, 100, 150 );
    Grafix.DrawRectangle ( HVGA, 40, 50, 80, 100 );

    Grafix.FillRectangle ( Image1, 30, 20, 120, 160 );

    Grafix.CopyImage ( Image1, 0, 0, Image1->GetWidth(),
                  Image1->GetHeight(),
                  Image2, 0, 0 );
    Grafix.DisplayImage ( HVGA, Image2, 0, 0,
                    Image2->GetWidth(),
                    Image2->GetHeight(),
                    0, 0 );

    getch ();
    Grafix.ResetDisplay ();
  } // End of main
```

5-5 Using Input

These are the interfaces to Input object.

BOOLEAN Init ()
- Initialize Keyboard handler. Returns SUCCESS, if successful returns FAILURE otherwise.

VOID DeInit ()

- Deinitialize the keyboard handler.

BOOLEAN IsKeyDown (LONG ScanKeyCode)
- Returns TRUE if specified KeyCode is pressed down.

VOID WaitForKey (LONG ScanKeyCode)
- Wait until user presses specified key.