Graphics.h library code

```
// The winbgim library, Version 6.0, August 9, 2004
// Written by:
//      Grant Macklem (Grant.Macklem@colorado.edu)
//      Gregory Schmelter (Gregory.Schmelter@colorado.edu)
//      Alan Schmidt (Alan.Schmidt@colorado.edu)
//      Ivan Stashak (Ivan.Stashak@colorado.edu)
//      Michael Main (Michael.Main@colorado.edu)
// CSCI 4830/7818: API Programming
// University of Colorado at Boulder, Spring 2003

// -------------------------------------------------------------------------
//                         Notes
// -------------------------------------------------------------------------
// * This library is still under development.
// * Please see http://www.cs.colorado.edu/~main/bgi for information on
// * using this library with the mingw32 g++ compiler.
// * This library only works with Windows API level 4.0 and higher (Windows 95, NT 4.0 and
newer)
// * This library may not be compatible with 64-bit versions of Windows
// -------------------------------------------------------------------------

// -------------------------------------------------------------------------
//                         Macro Guard and Include Directives
// -------------------------------------------------------------------------
#ifndef WINBGI_H
#define WINBGI_H
#include <windows.h>        // Provides the mouse message types
#include <limits.h>         // Provides INT_MAX
#include <sstream>          // Provides std::ostringstream
// -------------------------------------------------------------------------

// -------------------------------------------------------------------------
//                         Definitions
// -------------------------------------------------------------------------
// Definitions for the key pad extended keys are added here.  When one
// of these keys are pressed, getch will return a zero followed by one
// of these values. This is the same way that it works in conio for
// dos applications.
#define KEY_HOME        71
#define KEY_UP          72
#define KEY_PGUP        73
#define KEY_LEFT        75
#define KEY_CENTER      76
#define KEY_RIGHT       77
#define KEY_END         79
#define KEY_DOWN        80
```

```
#define KEY_PGDN        81
#define KEY_INSERT      82
#define KEY_DELETE      83
#define KEY_F1          59
#define KEY_F2          60
#define KEY_F3          61
#define KEY_F4          62
#define KEY_F5          63
#define KEY_F6          64
#define KEY_F7          65
#define KEY_F8          66
#define KEY_F9          67

// Line thickness settings
#define NORM_WIDTH      1
#define THICK_WIDTH     3

// Character Size and Direction
#define USER_CHAR_SIZE  0
#define HORIZ_DIR       0
#define VERT_DIR        1

// Constants for closegraph
#define CURRENT_WINDOW -1
#define ALL_WINDOWS -2
#define NO_CURRENT_WINDOW -3

// The standard Borland 16 colors
#define MAXCOLORS       15
enum colors { BLACK, BLUE, GREEN, CYAN, RED, MAGENTA, BROWN, LIGHTGRAY,
DARKGRAY,
            LIGHTBLUE, LIGHTGREEN, LIGHTCYAN, LIGHTRED, LIGHTMAGENTA, YELLOW,
WHITE };
// The standard line styles
enum line_styles { SOLID_LINE, DOTTED_LINE, CENTER_LINE, DASHED_LINE,
USERBIT_LINE };

// The standard fill styles
enum fill_styles { EMPTY_FILL, SOLID_FILL, LINE_FILL, LTSLASH_FILL, SLASH_FILL,
                BKSLASH_FILL, LTBKSLASH_FILL, HATCH_FILL, XHATCH_FILL,
INTERLEAVE_FILL,
                                WIDE_DOT_FILL, CLOSE_DOT_FILL, USER_FILL };
// The various graphics drivers
enum graphics_drivers { DETECT, CGA, MCGA, EGA, EGA64, EGAMONO, IBM8514,
HERCMONO,
                    ATT400, VGA, PC3270 };
// Various modes for each graphics driver
enum graphics_modes { CGAC0, CGAC1, CGAC2, CGAC3, CGAHI,
```

```
                MCGAC0 = 0, MCGAC1, MCGAC2, MCGAC3, MCGAMED, MCGAHI,
                        EGALO = 0, EGAHI,
                                EGA64LO = 0, EGA64HI,
                                        EGAMONOHI = 3,
                                                HERCMONOHI = 0,
                                                        ATT400C0 =
0, ATT400C1, ATT400C2, ATT400C3, ATT400MED, ATT400HI,

VGALO = 0, VGAMED, VGAHI,

                PC3270HI = 0,

                                IBM8514LO = 0, IBM8514HI };
// Borland error messages for the graphics window.
#define NO_CLICK        -1      // No mouse event of the current type in getmouseclick
enum graph_errors { grInvalidVersion = -18, grInvalidDeviceNum = -15, grInvalidFontNum,
            grInvalidFont, grIOerror, grError, grInvalidMode, grNoFontMem,
                        grFontNotFound, grNoFloodMem, grNoScanMem, grNoLoadMem,
                                grInvalidDriver, grFileNotFound, grNotDetected,
grNoInitGraph,
                                                grOk };
// Write modes
enum putimage_ops{ COPY_PUT, XOR_PUT, OR_PUT, AND_PUT, NOT_PUT };

// Text Modes
enum horiz { LEFT_TEXT, CENTER_TEXT, RIGHT_TEXT };
enum vertical { BOTTOM_TEXT, VCENTER_TEXT, TOP_TEXT }; // middle not needed other
than as seperator
enum font_names { DEFAULT_FONT, TRIPLEX_FONT, SMALL_FONT, SANS_SERIF_FONT,
          GOTHIC_FONT, SCRIPT_FONT, SIMPLEX_FONT, TRIPLEX_SCR_FONT,
          COMPLEX_FONT, EUROPEAN_FONT, BOLD_FONT };
          // ------------------------------------------------------------------------

// ------------------------------------------------------------------------
//                         Structures
// ------------------------------------------------------------------------
// This structure records information about the last call to arc.  It is used
// by getarccoords to get the location of the endpoints of the arc.
struct arccoordstype
{
   int x, y;                // Center point of the arc
      int xstart, ystart;       // The starting position of the arc
         int xend, yend;           // The ending position of the arc.
         };

// This structure defines the fill style for the current window.  Pattern is
// one of the system patterns such as SOLID_FILL.  Color is the color to
// fill with
```

```
struct fillsettingstype
{
   int pattern;               // Current fill pattern
      int color;                // Current fill color
      };

// This structure records information about the current line style.
// linestyle is one of the line styles such as SOLID_LINE, upattern is a
// 16-bit pattern for user defined lines, and thickness is the width of the
// line in pixels.
struct linesettingstype
{
   int linestyle;             // Current line style
      unsigned upattern;          // 16-bit user line pattern
         int thickness;           // Width of the line in pixels
         };

// This structure records information about the text settings.
struct textsettingstype
{
   int font;                  // The font in use
      int direction;             // Text direction
         int charsize;              // Character size
            int horiz;               // Horizontal text justification
               int vert;                // Vertical text justification
               };

// This structure records information about the viewport
struct viewporttype
{
   int left, top,             // Viewport bounding box
         right, bottom;
            int clip;                // Whether to clip image to viewport
            };

// This structure records information about the palette.
struct palettetype
{
   unsigned char size;
      signed char colors[MAXCOLORS + 1];
      };
      // --------------------------------------------------------------------------

// --------------------------------------------------------------------------
//                        API Entries
// --------------------------------------------------------------------------
#ifdef __cplusplus
extern "C" {
```

```
#endif

// Drawing Functions
void arc( int x, int y, int stangle, int endangle, int radius );
void bar( int left, int top, int right, int bottom );
void bar3d( int left, int top, int right, int bottom, int depth, int topflag );
void circle( int x, int y, int radius );
void cleardevice( );
void clearviewport( );
void drawpoly(int n_points, int* points);
void ellipse( int x, int y, int stangle, int endangle, int xradius, int yradius );
void fillellipse( int x, int y, int xradius, int yradius );
void fillpoly(int n_points, int* points);
void floodfill( int x, int y, int border );
void line( int x1, int y1, int x2, int y2 );
void linerel( int dx, int dy );
void lineto( int x, int y );
void pieslice( int x, int y, int stangle, int endangle, int radius );
void putpixel( int x, int y, int color );
void rectangle( int left, int top, int right, int bottom );
void sector( int x, int y, int stangle, int endangle, int xradius, int yradius );

// Miscellaneous Functions
int getdisplaycolor( int color );
int converttorgb( int color );
void delay( int msec );
void getarccoords( arccoordstype *arccoords );
int getbkcolor( );
int getcolor( );
void getfillpattern( char *pattern );
void getfillsettings( fillsettingstype *fillinfo );
void getlinesettings( linesettingstype *lineinfo );
int getmaxcolor( );
int getmaxheight( );
int getmaxwidth( );
int getmaxx( );
int getmaxy( );
bool getrefreshingbgi( );
int getwindowheight( );
int getwindowwidth( );
int getpixel( int x, int y );
void getviewsettings( viewporttype *viewport );
int getx( );
int gety( );
void moverel( int dx, int dy );
void moveto( int x, int y );
void refreshbgi(int left, int top, int right, int bottom);
void refreshallbgi( );
```

```
void setbkcolor( int color );
void setcolor( int color );
void setfillpattern( char *upattern, int color );
void setfillstyle( int pattern, int color );
void setlinestyle( int linestyle, unsigned upattern, int thickness );
void setrefreshingbgi(bool value);
void setviewport( int left, int top, int right, int bottom, int clip );
void setwritemode( int mode );

// Window Creation / Graphics Manipulation
void closegraph( int wid=ALL_WINDOWS );
void detectgraph( int *graphdriver, int *graphmode );
void getaspectratio( int *xasp, int *yasp );
char *getdrivername( );
int getgraphmode( );
int getmaxmode( );
char *getmodename( int mode_number );
void getmoderange( int graphdriver, int *lomode, int *himode );
void graphdefaults( );
char *grapherrormsg( int errorcode );
int graphresult( );
void initgraph( int *graphdriver, int *graphmode, char *pathtodriver );
int initwindow
   ( int width, int height, const char* title="Windows BGI", int left=0, int top=0, bool dbflag=false,
bool closeflag=true );
   int installuserdriver( char *name, int *fp );    // Not available in WinBGI
   int installuserfont( char *name );               // Not available in WinBGI
   int registerbgidriver( void *driver );           // Not available in WinBGI
   int registerbgifont( void *font );              // Not available in WinBGI
   void restorecrtmode( );
   void setaspectratio( int xasp, int yasp );
   unsigned setgraphbufsize( unsigned bufsize );    // Not available in WinBGI
   void setgraphmode( int mode );
   void showerrorbox( const char *msg = NULL );
// User Interaction
int getch( );
int kbhit( );

// User-Controlled Window Functions (winbgi.cpp)
int getcurrentwindow( );
void setcurrentwindow( int window );
// Double buffering support (winbgi.cpp)
int getactivepage( );
int getvisualpage( );
void setactivepage( int page );
void setvisualpage( int page );
void swapbuffers( );
```

```
// Image Functions (drawing.cpp)
unsigned imagesize( int left, int top, int right, int bottom );
void getimage( int left, int top, int right, int bottom, void *bitmap );
void putimage( int left, int top, void *bitmap, int op );
void printimage(
   const char* title=NULL,
      double width_inches=7, double border_left_inches=0.75, double border_top_inches=0.75,
         int left=0, int right=0, int right=INT_MAX, int bottom=INT_MAX,
            bool active=true, HWND hwnd=NULL
               );
               void readimagefile(
                  const char* filename=NULL,
                     int left=0, int top=0, int right=INT_MAX, int bottom=INT_MAX
                        );
                        void writeimagefile(
                           const char* filename=NULL,
                              int left=0, int top=0, int right=INT_MAX, int bottom=INT_MAX,
                                 bool active=true, HWND hwnd=NULL
                                    );
// Text Functions (text.cpp)
void gettextsettings(struct textsettingstype *texttypeinfo);
void outtext(char *textstring);
void outtextxy(int x, int y, char *textstring);
void settextjustify(int horiz, int vert);
void settextstyle(int font, int direction, int charsize);
void setusercharsize(int multx, int divx, int multy, int divy);
int textheight(char *textstring);
int textwidth(char *textstring);
extern std::ostringstream bgiout;
void outstream(std::ostringstream& out=bgiout);
void outstreamxy(int x, int y, std::ostringstream& out=bgiout);
// Mouse Functions (mouse.cpp)
void clearmouseclick( int kind );
void clearresizeevent( );
void getmouseclick( int kind, int& x, int& y );
bool ismouseclick( int kind );
bool isresizeevent( );
int mousex( );
int mousey( );
void registermousehandler( int kind, void h( int, int ) );
void setmousequeuestatus( int kind, bool status=true );

// Palette Functions
palettetype *getdefaultpalette( );
void getpalette( palettetype *palette );
int getpalettesize( );
void setallpalette( palettetype *palette );
void setpalette( int colornum, int color );
```

```
void setrgbpalette( int colornum, int red, int green, int blue );

// Color Macros
#define IS_BGI_COLOR(v)     ( ((v) >= 0) && ((v) < 16) )
#define IS_RGB_COLOR(v)     ( (v) & 0x03000000 )
#define RED_VALUE(v)        int(GetRValue( converttorgb(v) ))
#define GREEN_VALUE(v)      int(GetGValue( converttorgb(v) ))
#define BLUE_VALUE(v)       int(GetBValue( converttorgb(v) ))
#undef COLOR
int COLOR(int r, int g, int b); // No longer a macro

#ifdef __cplusplus
}
#endif
// ----------------------------------------------------------------------------

#endif // WINBGI_H
```