The CMUGraphics package composed of the files Graphics.h and Graphics.cpp

provides the following member functions, typically accessed via a global

window object. If you are passing a window as a parameter, it MUST be

passed by reference.

-------------------------------------------------------------------------

0. Creating a window/looking at settings

- window myVariableName(width, height, xLocation, yLocation)

default values for height and width: 640, 480

- GetWidth( )

returns the width of the window

- GetHeight( )

returns the height of the window

- GetWindowSize(x, y)

returns width and height of the window in x and y

The window's (0, 0) point is in the upper left corner. The x axis proceeds

to the right of the window and the y axis to the bottom. Note that to go

"left" in the x direction from a particular location you will be decrementing

in x, and to go "right", you will be incrementing in x. Similarly, to go "up"

in the y direction from a particular location you will be decrementing in y,

and to go "down", you will be incrementing in y.

1. Setting Colors

a color-spec can be set in one of three ways:

- by using a color parameter (one of the colors defined in

Colors.h, e.g., BLACK, or one of your own making)

- by using real triples of RGB values (e.g., 1.0, 0.5, 1.0)

- by using unsigned char triples of RGB values

- SetBrush(color-spec)

sets the color of the current brush (used when drawing in

the FILLED style)

- SetPen(color-spec, size)

sets the color and size (default is 1) of current pen

(used when drawing in the FRAME style, as well as the

border when drawing in the FILLED style)

2. Drawing Shapes (more detailed comments can be found in graphics.h)

objects (other than lines) may be drawn in one of the following 3

styles:

- FRAME (a wire-frame outline as wide as the current pen size)

- FILLED (filled with the current brush color with a border of

the current pen color and size; to achieve no border, you

must have the pen color and brush color be the same)

- INVERTED (current color is irrelevant, the pixels that compose

the shape are "flipped" or "switched" to their opposite

color. For example if you draw an inverted rectangle on a

white background, the rectangle will be black. If you draw

an inverted rectangle on a black background, the rectangle

will be white. The other colors have some very interesting

opposites. If you draw two inverted rectangles on the

same place, the rectangle will appear and then disappear.

- DrawPixel(x1, y1)

draws a pixel in the current pen color

- DrawLine(x1, y1, x2, y2)

draws a filled line in the current pen color from

(x1, y1) to (x2, y2)

- DrawRectangle(x1, y1, x2, y2, style, width, height)

draws a rectangle from (x1, y1) to (x2, y2). Default

style is FILLED. Parameters width and height control

"roundedness" and default to 0.

- DrawTriangle(x1, y1, x2, y2, x3, y3, style)

draw a triangle encompassing the 3 points. Default style

is FILLED.

- DrawPolygon(x array, y array, numVertices, style)

draws a polygon encompassing the numVertices vertices.

The number of elements in the x array must be the same as

the y array and must match numVertices. Default style is

FILLED.

- DrawCircle(x, y, r, style)

draws a circle centered at (x, y) with radius r. Default

style is FILLED.

- DrawEllipse(x1, y1, x2, y2, style)

draws an ellipse inside the rectangle bounded by (x1, y1)

and (x2, y2). Default style is FILLED.

- DrawArc(x1, y1, x2, y2, startAngle, endAngle, style, angle-spec)

draws a section of the ellipse inside the rectangle

bounded by (x1, y1) and (x2, y2) from startAngle to

endAngle. Default style is FRAME, default method of

specifying angles is DEGREES (can also use RADIANS). 0

degrees is 3:00, 90 is 12:00, 180 is 9:00, 270 is 6:00.

- DrawBezier(x1, y1, x2, y2, x3, y3, x4, y4)

draws a bezier curve from (x1, y1) to (x4, y4), using

(x2,y2) and (x3, y3) as control points.

3. Drawing Text and Numbers

- DrawString(x, y, textString)

draws a string at location (x, y) using the font set by

SetFont

- DrawInteger(x, y, integer)

draws an integer at location (x, y) using the font set by

SetFont

- DrawDouble(x, y, real)

draws a real number at location (x, y) using the font set

by SetFont

- SetFont(size, style, fontFamily, fontString)

sets the font characteristics. Styles are PLAIN, BOLD,

ITALICIZED, UNDERLINED, and STRIKEOUT. To achieve

multiple effects, you can logically OR styles together,

e.g., BOLD | UNDERLINED. Abstract font families currently

supported include MODERN, ROMAN, SCRIPT, SWISS. If there

is a particular font that you wish to specify, you must

use the fontFamily parameter BY\_NAME and then provide the

name of the font as a quoted string to fontString.

- GetStringSize(width, height, textString)

returns the width and height of the string drawn in the

current font

- GetIntegerSize(width, height, integer)

returns the width and height of the integer drawn in the

current font

- GetDoubleSize(width, height, real)

returns the width and height of the real number drawn in

the current font

4. Mouse/Keyboard input

- GetMouseCoord(x, y)

returns the current mouse coordinates in x and y

- GetButtonState(button-spec, x, y)

returns the state (BUTTON\_UP, BUTTON\_DOWN) of the button

(LEFT, RIGHT) and the mouse coordinates in x and y

- GetMouseClick(x, y)

returns the type of the next mouse click event (L\_CLICK,

R\_CLICK) from the queue as well as the location of the

click

- WaitMouseClick(x, y)

as above, except waits for a mouse click event

- GetKeyPress(key)

returns the type of the next keyboard event (ASCII, ARROW,

FUNCTION, ESCAPE) from the queue, as well as the value of

the key pressed

- WaitKeyPress(key)

as above, excepts waits for a keyboard event

- SetBuffering(boolean)

turns double buffering on (true) or off (false)

- UpdateBuffer( )

when double buffering is on, copies the offscreen buffer

to the screen

- FlushMouseQueue( )

flushes the mouse input buffer

- FlushKeyQueue( )

flushes the keyboard input buffer

5. Other useful window member functions

- GetColor(x, y)

returns the color of the pixel at location (x, y) as a

color type

- GetColor(x, y, red, green, blue)

returns the color of the pixel at location (x, y) as

doubles for each of red, green, and blue.

- GetRed(x, y)

- GetGreen(x, y)

- GetBlue(x, y)

returns the value of the appropriate color at (x, y) as a

double

- DrawImage(image, x, y, width, height)

draws an image at (x, y). The width and height parameters

can be used to scale the image.

- StoreImage(image, x, y, width, height)

grabs a section of the screen from (x, y), width wide and

height tall, and stores it as an image object.

- ChangeTitle(string)

changes the title of the window

- Print

prints the contents of the window to a printer (buffering

must be on)

6. Other useful free (non-member) functions (found in auxil.h, which must

be #include'd in your source code in order to use them)

- Pause(milliseconds)

pauses for the specified number of milliseconds