

# DrawApp v4.5

## Drawing Program Functions List

The functions listed below are declared in graphics.h.  
All coordinate and size values are in pixels.

### Lines and Shapes

#### **drawLine(int x1, int y1, int x2, int y2)**

Draw a straight line, width one pixel, from (x1, y1) to (x2, y2).

#### **drawRect(int x, int y, int width, int height)**

Draw a rectangle with top left-hand corner at (x,y), line width one pixel, and size width by height pixels.

#### **drawRectRotated(int x, int y, int width, int height, int angle)**

Draw a rectangle with top left-hand corner at (x,y), and size width by height pixels, rotated around (x,y).

#### **fillRect(int x, int y, int width, int height)**

Draw a filled rectangle with top left-hand corner at (x,y) and size width by height. The fill colour is the current colour.

#### **fillRectRotated(int x, int y, int width, int height, int angle)**

Draw a filled rectangle with top left-hand corner at (x,y), and size width by height pixels, rotated around (x,y). The fill colour is the current colour.

#### **drawOval(int x, int y, int width, int height)**

Draw an oval inside the area defined by the rectangle with top left corner at (x,y) and size width by height.

#### **drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)**

Draw an arc line, width one pixel, inside the area defined by the rectangle with top left corner at (x,y) and size given by width and height. startAngle is the angle in degrees that specifies where the arc starts (0 is the 3 o'clock position). arcAngle is the angle in degrees that specifies the end of the arc, relative to startAngle. 360 draws a complete circle.

#### **fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)**

The same as drawArc except the arc is filled. The fill colour is the current colour.

#### **void drawPolygon(int n, int x[], int y[])**

Draw a closed polygon, with n points (corners) specified in the x and y arrays. Each corresponding x and y value in the arrays, for example (x[0],y[0]), represents a point. A line is drawn between the first and last points if they are not the same, to make the polygon closed.

#### **void fillPolygon(int, int[], int[])**

The same as drawPolygon except that the polygon is filled. The fill colour is the current colour.

# Strings

## **drawString(char\*, int x, int y)**

A string is a sequence of characters (words, sentences, paragraphs and so on). This function displays a string at the position (x,y), providing a way of 'drawing' characters and text. Quotes are used to denote a string so that the characters can be distinguished from other pieces of source code, for example: drawString("Hello",40,40).

## **drawStringRotated(char\*, int x, int y, int angle)**

This function displays a string at the position (x,y), rotated about the (x, y) position by angle degrees.

## **setStringTextSize(int size)**

Set the font size used to draw strings. The new size remains in effect until the function is called again with a different size.

# Images

## **void displayImage(char\* fileName, int x, int y)**

Display an image from the file named by fileName at coordinates (x,y). For a simple file name (e.g., photo.png) the image file must be located in the same directory the drawing program is run from. Alternatively a relative file name can be provided to get the image from another directory.

# Colours

These functions set the colour to be used for anything drawn using any of the other drawing functions listed in this document. The colour remains set until changed by another call to a colour setting function.

## **setColour(colour)**

Set the colour used to draw with. The list of available colours is: black, blue, cyan, darkgray, gray, green, lightgray, magenta, orange, pink, red, white, yellow. To set a colour just use the name, for example: setColour(green).

## **void setRGBColour(int red, int green, int blue)**

Set the drawing colour using the Red, Green, Blue (RGB) colour model. Each colour is specified as a combination of a red, green and blue component, each of which has a range of 0 to 255. There are many examples of RGB colour pickers on the web to help find out the values for the colours you want, for example: [https://www.w3schools.com/colors/colors\\_rgb.asp](https://www.w3schools.com/colors/colors_rgb.asp)

# Layers

Two drawing layers are supported, foreground and background. All other functions, except for setWindowSize, operate on the currently selected layer only. Drawing on one layer does not draw on the other layer. The background layer is always displayed below the foreground layer, such that anything drawn on the foreground will cover anything below it on the background. This allows, for example, having a background layer displaying a specific pattern, drawing or image to be visible below anything separately displayed on the foreground.

**foreground()**

Set the foreground layer as the one to draw on. All subsequent function calls will operate on the foreground until the background is selected.

**background()**

Set the background layer as the one to draw on. All subsequent function calls will operate on the background until the foreground is selected.

**clear()**

Clear the current layer so it is not displaying anything. The window background is white, so if nothing is displayed by either layer the window background shows through.

## Window Size

**void setWindowSize(int width, int height)**

Set the window to size to be large enough to display a drawing area of width by height pixels. Any existing drawing in the window is erased, hence this function should be called at the start of a program before any drawing is done.

If the window is resized by dragging a window border or maximising/minimising the window, the drawing area size does not change. If the window is re-sized to be smaller than the drawing area, drawing still works normally in the areas not visible. When the window is re-sized again to be larger, anything drawn that was not visible becomes visible.

The minimum size is 50,50.

The maximum size is 2500,1500.

## Sleep

**sleep(int milliseconds)**

Set a delay of milliseconds before the next drawing command is run. One second equals 1000 milliseconds, so a one second delay is sleep(1000), half second sleep(500), and so on.

Note that excessive use of sleep, or very long sleep delays, may cause problems if a drawing program is outputting a large number of drawing commands.

## Message

**void message(char\* message)**

Display the message string passed as a parameter in the message panel below the drawing area. Newline characters in the message string will be correctly displayed as line breaks in the message panel. This is useful for debugging or displaying general messages about what is being drawn.