gfx: A Simple Graphics Library (V2)

This page describes `gfx`, a simple graphics library for CSE 20211. This library is meant to be simple and easy to learn, so that beginning CSE students can get right into the interesting parts of programming. The `gfx` library only requires that the programmer understand how to invoke basic C functions with scalar arguments.

It is more than enough to explore raster and vector graphics, create animations, draw fractal shapes, and write simple video games. It doesn't do *everything* that you might want a graphics library to do. As you learn more about programming, more advanced libraries that you might consider using are OpenGL for precise 3-D graphics, Qt for windowed applications, and SDL for video games.

This library runs on Linux machines using the X11 Window System. (You may be able to run it on a Mac, but it won't work on Windows.) So, you will have to go to the computer lab to work on this assignment.

Getting Started

Download the following files and put them in your lab directory:

- gfx.c (the source file: read if you like, but don't change)
- gfx.h (the header file: read if you like, but don't change)
- example.c (an example program: go ahead and change this)

To compile a program that uses the `gfx` library, you need to use the following command, which compiles your program with `gfx.c` then adds the X11 and math libraries. (X11 is short for the X Window System used on Unix based machines.)

```
gcc example.c gfx.c -o example -lX11 -lm
```

On a Mac with XCode and the X Windows option installed, the following might work:

```
gcc example.c gfx.c -o example -I/usr/X11R6/include -L/usr/X11R6/lib -lX11 -lm
```

Run the example program, and you should see a window pop up with a green triangle.

```
./example
```

Line by Line Example

To give you the idea of how to use the library, we will point out the important parts of `example.c` in detail. Then, below you can read about how the individual functions work.

Begin by including `gfx.h`, to make the library available to the program. Note that quotes rather than angle brackets are used in this statement. Quotes indicate the header file is in the current directory, rather than in some system-defined place.

```
#include "gfx.h"
```

To open a graphics window, call `gfx_open` once at the beginning of the program, giving the width and height of the window in pixels, and the title of the window on the screen:

```
gfx_open(xsize,ysize,"Example Graphics Program");
```

`gfx_color` sets the current drawing color to a pale green:

```
gfx_color(0,255,0);
```

Now, use the `gfx_line` function to draw a line from (100,100) to (200,100):

```
gfx_line(100,100,200,100);
```

Wait for the user to press a key (or button):

```
c = gfx_wait();
```

If the key is the letter 'q', break out of the loop:

```
if(c=='q') break;
```

That shows you enough to get started. There are a few more useful functions in the library, read the following section to see them all.

Function Reference

```
void gfx_open( int width, int height, const char *title );
```

`gfx_open` creates a new graphics window on the screen. `width` is the width of the window in pixels. `height` is the height of the window in pixels. `title` is the name of the window in the title bar.

```
void gfx_line( int x1, int y1, int x2, int y2 );
```

`gfx_line` draws a line in the graphics window, using the current color, from location `x1,y1` to `x2,y2`

```
void gfx_color( int red, int green, int blue );
```

`gfx_color` sets the current color for drawing, using the RGB color model, in which you mix red, green, and blue components in order to get the desired color. `gfx_color` takes three integers between 0 and 255, each representing red, green, and blue. The following are some example colors:

Red	Green	Blue	Color Name
255	0	0	Red
0	255	0	Green
0	0	255	Blue
255	255	0	Yellow
255	255	255	White

100	100	100	Gray
0	0	0	Black

```
void gfx_clear();
```

`gfx_clear` clears the window to the current background color, which is black by default.

```
void gfx_clear_color( int red, int green, int blue );
```

`gfx_clear_color` changes the background color, using the same format as `gfx_clear` above. The new background color takes effect the next time `gfx_clear` is called.

```
char gfx_wait();
```

`gfx_wait` waits until the user presses a key or mouse button. If the user has pressed a mouse button, the integer 1, 2, or 3 will be returned, indicating the button pressed. If the user has pressed a key, that character will be returned. To determine the mouse location at that time, call `gfx_xpos` and `gfx_ypos`.

```
int gfx_xpos();
int gfx_ypos();
```

`gfx_xpos()` and `gfx_ypos` return the X and Y coordinates, respectively, of the mouse pointer when the last key or button was pressed.

```
int gfx_event_waiting();
```

`gfx_wait` causes the program to wait until something is pressed, which can be inconvenient in a real-time program. To avoid this, call `gfx_event_waiting`, which returns true if an event (key or button press) has occurred, otherwise returns false immediately. If it returns true, then you can call `gfx_wait` to retrieve the event without waiting.

```
void gfx_flush();
```

`gfx_flush` flushes all output to the graphics window. This forces all previous drawing commands to take effect. Output is normally flushed after a called to `gfx_wait`. If your program does not call `gfx_wait` then you need to call `gfx_flush` after drawing to the screen.