Standard C Graphics Library, version 1.1

- S, Gautam

SECTION A – Introduction

The standard C graphics library is intended to be a tiny C library that makes it easier to create simple graphical applications. The current implementation can be built on Windows with SDL 1.2 support. This document serves as the *official* documentation for the library.

The library aims to serve as a quick-n-dirty way of prototyping applications which would otherwise be rather cumbersome to write in an API like OpenGL or even SDL.

The library current has support for graphics as well as simple audio, which uses portaudio as a backend and can play OGG files.

SECTION B – Compilation

To compile an stdgfx application, simply execute the following command under a Windows shell:

```
gcc app. c − o app. exe − lm − lmingw32 − l.\lib\libgfx1. a − I.\include
```

Replace .\lib\libgfx1. a to wherever your stdgfx library directory is. Same goes for the include directory.

(NOTE: The current implementation of stdgfx supplies its own version of SDL, you may not need an SDL distribution)

SECTION C – Usage

All programs must use the entry point $int\ gfx\_main(int\ argc, char ** argv)$, the GFX analogue of the standard $int\ main(int\ argc, char ** argv)$

SECTION C – API

Function	**Description**
`int gfx_init(void);`	Initialize the standard graphics library, should not be called by the program!
`void gfx_ok(int test, const char* fmt, …)`	If test is false, show the formatted error message, otherwise, return.
`void gfx_resize(int w, int h, int bpp)`	Resize the screen as specified by the variables **w**idth, **h**eight, and **b**its **p**er **p**ixel.
`uint32_t gfx_color(int r, int g, int b, int a)`	Return the constructed 32-bit color from the R, G, B, A values. This value can be directly put into the framebuffer returned by *gfx_framebuffer()*.
`void gfx_screen(int w, int h, int bpp)`	Initialize the screen, do not use it! Use *gfx_resize* instead.
`void gfx_input(void)`	Flush the input buffers.
`int gfx_iskey(int key)`	Check for gamepad key. Currently the following keys are defined (as of GFX 1.1): - GFX_ESC - GFX_UP - GFX_DOWN - GFX_LEFT - GFX_RIGHT - GFX_W - GFX_A - GFX_S - GFX_D - GFX_CTRL - GFX_Z - GFX_X

`void    gfx_sleep(int s)`	Sleep for microseconds specified by argument *s*.
`int gfx_ticks(void)`	Return CPU ticks.
`void gfx_update(void)`	Update input and display buffers.
`void gfx_lock(void)`	Lock the display buffer.
`void gfx_unlock(void)`	Unlock the display buffer.
`void gfx_caption(void)`	Set the window caption.
`void gfx_flush(void)`	Flush the display buffer.
`void gfx_clear(int r, int g, int b, int a)`	Clear the display with the specified R, G, B, A values. (Does not reset the terminal cursors)
`void gfx_pixel(int x, int y, int r, int g, int b, int a)`	Draw a pixel at X, Y with R, G, B, A colour.
`void gfx_pixel_wrap(int toggle)`	Set pixel wrapping ON/OFF. (0 = off, 1 = on)
`uint8_t* gfx_font()`	Return the point to the internal 8x8 font.
`int gfx_putc(int c)`	Put a character on the screen. (Uses the internal terminal emulator)
`int gfx_puts(char* str)`	Puts a NULL terminated C string on the screen.
`int gfx_printf(const char* fmt, …)`	Puts a C formatted string on the screen.
`uint32_t* gfx_framebuffer(void)`	Return the raw pixel framebuffer.
`void gfx_text_bg(int r, int g, int b, int a)`	Set the internal terminal's text background colour.
`void gfx_text_fg(int r, int g, int b, int a)`	Set the internal terminal's text foreground colour.
`void gfx_text_cursor(int x, int y)`	Set the terminal cursor. Note: These are the terminal co-ordinates, NOT screen co-ordinates!
`int gfx_line(int x1, int y1, int x2, int y2, int r, int g, int b, int a)`	Draw a line with colour.
`int gfx_triangle(int x1, int y1, int x2, int y2, int x3, int y3, int r, int g, int b, int a)`	Draw a triangle with colour.
`int gfx_circle(int xc, int yc, int radius, int r, int g, int b, int a)`	Draw a circle with colour.
`int gfx_disk(int xc, int yc, int radius, int r, int g, int b, int a)`	Draw a filled disk.
`int gfx_quad(int x1, int y1, int x2, int y2, int r, int g, int b, int a)`	Draw a filled quadrilateral.
`IMAGE* gfx_load_image(char* filename);`	Return an IMAGE pointer, loading the image data from the filename. The IMAGE structure is implementation defined. However, IMAGE->w and IMAGE->h will return the width and height of the image.
`void gfx_free_image(IMAGE*)`	Free the IMAGE* structure.
`void gfx_draw_image(IMAGE* img, int x, int y, int w, int h)`	Draw the IMAGE at X, Y with height W and H. Resizing is done appropriately!
`int gfx_mouse_x(void)`	Return X co-ordinate of mouse. (Flush the input buffers before calling this!)
`int gfx_mouse_y(void)`	Return Y co-ordinate of mouse. (Flush the input buffers before calling this!)
`int gfx_elapsed(int ticks);`	Return elapsed milliseconds from a value of ticks, which you previous obtained from *gfx_ticks*().
`void gfx_saveshot(char* filename)`	Save the screenshot of the screen to a file as a PNG image.

`SOUND* gfx_load_sound(char* filename)`	Load an OGG file and pass the SOUND* structure. This structure is implementation defined.
`int gfx_play_sound(SOUND* snd)`	Play the sound and return an instance ID.
`void gfx_stop_sound(int id)`	Stop the instance of the sound playing.
`int gfx_isplaying_sound(int id)`	Check if the sound is still playing. (1 = yes, 0 = no)
`void gfx_free_sound(SOUND* snd)`	Free the SOUND structure.
`GFX 1.1`	
`int gfx_waitchar(void)`	Wait for ASCII user input.

SECTION D – Global Constants

The table below shows a few global variables that the library will define.

Name	Description
`gfx_font_height`	Height of the internal terminal font (usually 8)
`gfx_font_width`	Width of the internal terminal font (usually 8)
`gfx_bytes_per_char`	Number of pixels per character (usually 64)
`gfx_height`	Current height of the display
`gfx_width`	Current width of the display
`gfx_bpp`	Current BPP of the display
`gfx_terminal_max_x`	Maximum terminal abscissa accepted on the display, calculated as: $gfx\_width\ /\ gfx\_font\_width$
`gfx_terminal_max_y`	Maximum terminal ordinate accepted on the display
`gfx_terminal_x`	Current X co-ordinate for the terminal cursor
`gfx_terminal_y`	Current Y co-ordinate for the terminal cursor