Fenster

Fenster /ˈfɛnstɐ/ -- a German word for "window".

This library provides the most minimal and highly opinionated way to display a cross-platform 2D canvas. If you remember Borland BGI or drawing things in QBASIC or INT 10h- you know what I mean. As a nice bonus you also get cross-platform keyboard/mouse input and audio playback in only a few lines of code.

What it does for you

- Single application window of given size with a title.
- Application lifecycle and system events are all handled automatically.
- Minimal 24-bit RGB framebuffer.
- Cross-platform keyboard events (keycodes).
- Cross-platform mouse events (X/Y + mouse click).
- Cross-platform timers to have a stable FPS rate.
- Cross-platform audio playback (WinMM, CoreAudio, ALSA).
- Simple polling API without a need for callbacks or multithreading (like Arduino/Processing).
- One C99 header of ~300LOC, easy to understand and extend.
- Go bindings (import "github.com/zserge/fenster", see godoc)
- Zig bindings (see examples/minimal-zig)
- Lua bindings (see https://github.com/jonasgeiler/lua-fenster)
- And, yes, it can run Doom!

Example

Here's how to draw white noise:

```
// main.c
#include "fenster.h"
#define W 320
#define H 240
int main() {
    uint32_t buf[W * H];
    struct fenster f = { .title = "hello",
                         .width = W,
                         .height = H,
                         .buf = buf
    };
    fenster_open(&f);
    while (fenster_loop(&f) == 0) {
        for (int i = 0; i < W; i++) {
          for (int j = 0; j < H; j++) {
            fenster_pixel(&f, i, j) = rand();
          }
        }
     }
    fenster_close(&f);
    return 0;
}
```

Compile it and run:

```
# Linux
cc main.c -lX11 -lasound -o main && ./main
# macOS
cc main.c -framework Cocoa -framework AudioToolbox -o main && ./main
# windows
cc main.c -lgdi32 -lwinmm -o main.exe && main.exe
```

That's it.

API

API is designed to be a polling loop, where on every iteration the framebuffer get updated and the user input (mouse/keyboard) can be polled.

```
struct fenster {
  const char *title; // window title
  const int width; // window width
  const int height; // window height
  uint32_t *buf; // window pixels, 24-bit RGB, row by row, pixel by pixel
  int keys[256]; // keys are mostly ASCII, but arrows are 17..20
  int mod;       // mod is 4 bits mask, ctrl=1, shift=2, alt=4, meta=8
  int x;         // mouse X coordinate
  int y;         // mouse Y coordinate
  int mouse;     // 0 = no buttons pressed, 1 = left button pressed
};
```

`int fenster_open(struct fenster *f)` - opens a new app window.

`int fenster_loop(struct fenster *f)` - handles system events and refreshes the canvas. Returns negative values when app window is closed.

`void fenster_close(struct fenster *f)` - closes the window and exists the graphical app.

`void fenster_sleep(int ms)` - pauses for ms milliseconds.

`int64_t fenster_time()` - returns current time in milliseconds.

`fenster_pixel(f, x, y) = 0xRRGGBB` - set pixel color.

`uint32_t px = fenster_pixel(f, x, y);` - get pixel color.

See examples/drawing-c for more old-school drawing primitives, but also feel free to experiment with your own graphical algorithms!

License

Extraido de: https://github.com/zserge/fenster