xv6 Window Manager

Machaidze, Elene
elene@mit.edu

Gupta, Keshav
keshav21@mit.edu

December 6, 2019

1 Overview

We implemented a window manager for xv6, built on top of a VGA driver that was also implemented by us for the project. The screen is divided into six fixed size windows, thereby letting up to six user-space programs display a window. We also provide functionality for a currently active window to accept user inputs. The kernel is currently not thread-safe, therefore we require the use of `CPUS=1` when `make`ing `qemu`. The GitHub repository is provided in section 5.2.

2 Description

2.1 VGA Driver

During boot, the driver sets up the emulated Cirrus VGA card in 320x200 256-color mode. It also installs a 256-color palette, and stores a pointer to the global VGA framebuffer for the kernel. We used the references in 5.1 for VGA mode register values.

2.2 Window Manager

The screen is divided into six windows of fixed size (100x95) leaving 10 pixels of padding between the windows. The currently active window is indicated by a red border around it, and all keystrokes go to this window. By default, the newest window is selected as the active window, however this can be changed as described in section 2.4.

2.3 System Calls

The functionality we implemented is extended to the user through the following system calls:

- `show_window(char *)` takes a pointer to the buffer which holds pixel values to be displayed in the process' window space. If a space has not been allocated for the process yet, this system call also allocates one of the six frames for the calling process. Since the pointer is in user space, `copyin` is used to copy the frame data from the user space into the VGA frame buffer.
- `close_window()` closes the window which belongs to the current process. Exiting a process calls this automatically on the exiting process' window.
- `reg_keycb(void (*)(int))` can be used to register a callback function which will be called with the key value when this process' window is active and the user presses a key.
- `cb_return()` must be used as a safe return from a key callback i.e. control must not reach the end of a key callback function, isntead all paths through the callback must end with this system call.

2.4 Keyboard input

By default, all keyboard input goes directly to the console. A few special key combinations cause the input to be redirected to the window manager instead, and to the window's callback function, if one has been registered.

- `Ctrl+Z` causes the next keystroke to be redirected to the key callback of the currently active window, instead of the console.
- `Ctrl+N` followed by a numeric key from 0-5 causes the window at that position (row-major order) to become the actively selected window.
- `Ctrl+X` kills the currently active window (and it's associated process).

3 Demo user programs

To open more than one window at once, it is recommended to use the `&` construct after the command to ensure non-blocking execution.

3.1 `brot`

It renders the Mandlbrot set inside it's assigned window. The user can use the keystrokes **w**, **a**, **s** or **d** to move the view around, and key strokes **q** and **e** to zoom in and out respectively. Due to the lack of float math on the emulator, we ended up using fixed point math implemented using the 64-bit integers as a subsititute.

3.2 `ball`

A small ball bounces around inside the window (no gravity, perfectly elastic collisions). It registers a callback, and sending the keystrokes **w**, **a**, **s** or **d** change the trajectory of the ball.

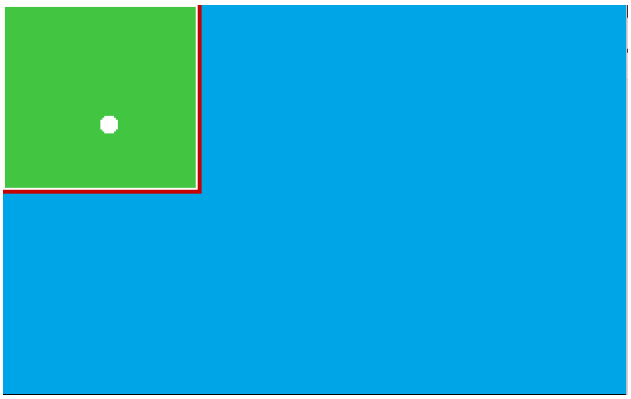
3.3 `count`

This simple program displays a count-up in the center of the screen that increments every 10 ticks (roughly 1 second).

4 Screenshots

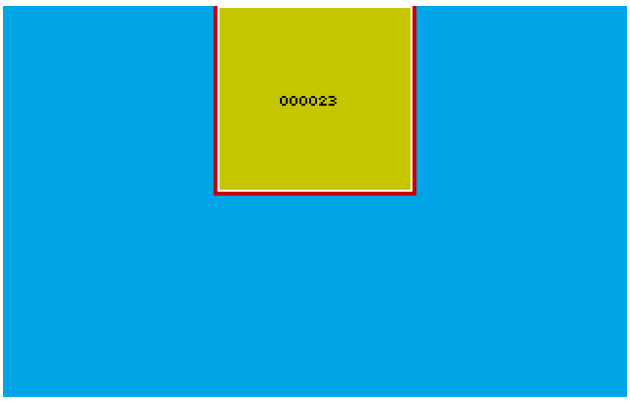
Run `make CPUS=1 qemu` and connect a VNC client to `localhost:0`. You should see the blue background displayed.

`ball &` shows a ball bouncing around

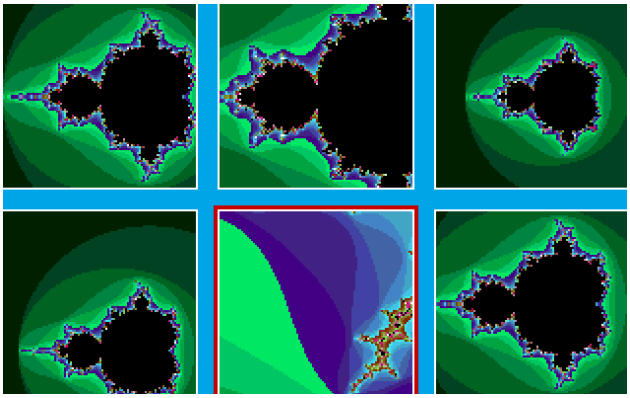


`count &` adds another window to the mix.

Removing the first `ball` instance does not affect `count`.



Six instances of `brot` running together, zoomed and moved to a different location.



5 References

5.1 VGA Register Values

https://wiki.osdev.org/VGA_Hardware
http://www.osdever.net/FreeVGA/home.htm

5.2 GitHub Repository

https://github.com/keshavgupta21/xv6-vga/tree/xv6-riscv-fall19