Milestone 2: SeeGOL

(Shoyler's Extremely Experimental Graphical Operating Library)

Schuyler Martin <sam8050@rit.edu> <http://shoyler.com>
Computer Science, BS/MS
Rochester Institute of Technology
Computer Science MS Project, CSCI-788-02

The Premise

(Recap)

HSC
Henrietta Semiconductor Corporation

In an alternative universe...

Henrietta Semiconductor Corporation

A monument to compromise

16-bit 8086

32-bit i386+
(with 16-bit Real Mode)

HSC
Henrietta Semiconductor Corporation

Project So Far

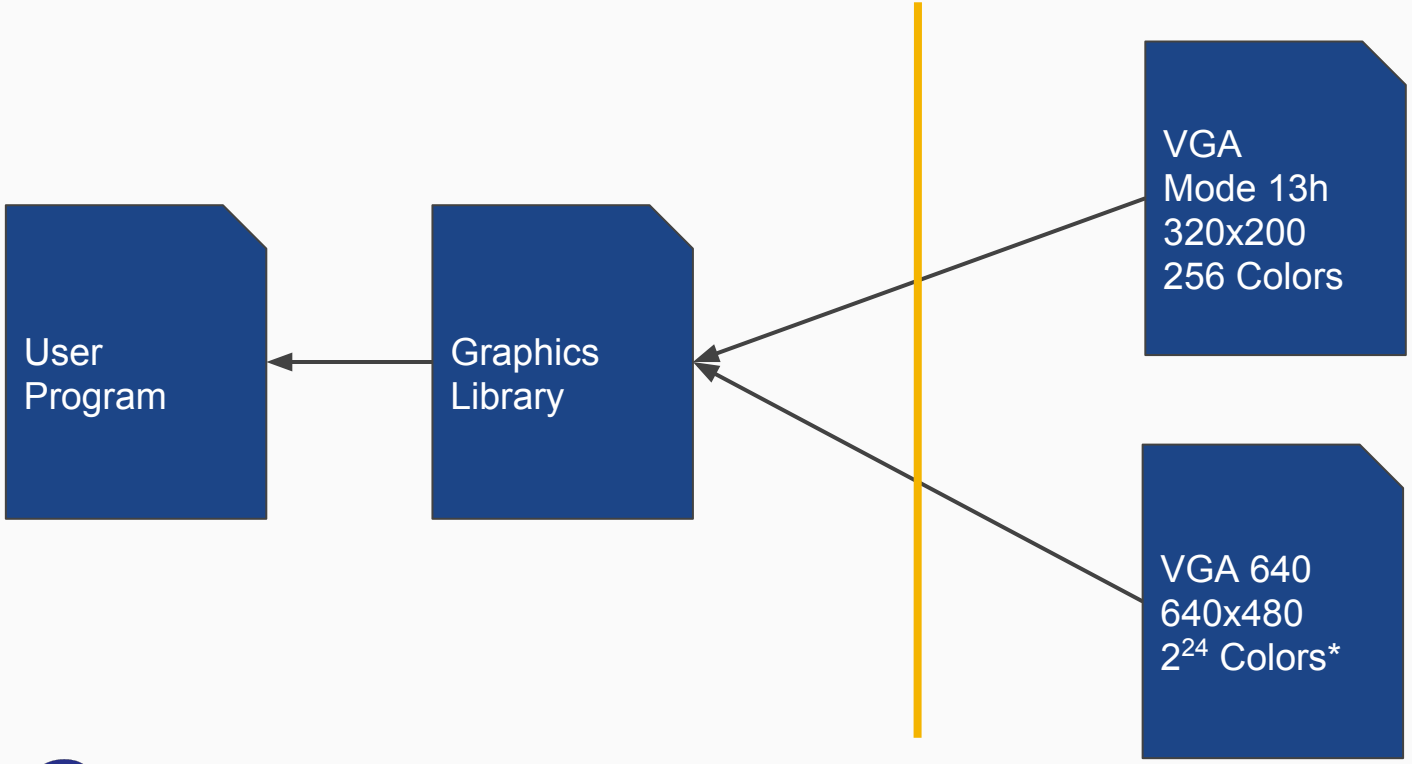
HSC
Henrietta Semiconductor Corporation

Stage 2:
Graphics Mode

- A basic graphics library has been built around a
- Currently only supports one VGA driver, “Mode 13h”
  - 320x200 Resolution
- Mode 13h is a color palette mode
  - Each pixel is represented by a single byte
  - The byte is an index into a color table with 256 options

HSC
Henrietta Semiconductor Corporation

User vs Kernel Space



*Not yet implemented in SeeGOL

HSC
Henrietta Semiconductor Corporation

Part of the VGA Driver Definition

```
typedef struct VGA_Driver VGA_Driver;
struct VGA_Driver
{
    // current screen size
    uint16_t screen_w;
    uint16_t screen_h;
    // address of start of the frame buffer for each mode
    uint16_t* frame_buff;
    // remember what graphical mode we are in
    uint8_t vga_mode;

    /*
    ** Start a VGA mode
    **
    ** @param driver Standard driver spec for a driver to setup
    */
    void (*vga_enter)(VGA_Driver* driver);

    /*
    ** Clears the video buffer
    */
    void (*vga_clrscr)(void);

    /*
    ** Vertical sync control. Useful for slow electron-gun-based displays
    */
    void (*vga_vsync)(void);

    /*
    ** Write a pixel out to the frame buffer. This represents a single pixel
    **
    ** @param x coordinate on the screen
    ** @param y coordinate on the screen
    ** @param color Pixel color to write
    */
    void (*vga_put_pixel)(uint16_t x, uint16_t y, RGB_8 color);
```

HSC
Henrietta Semiconductor Corporation

Part of the VGA 13 Driver

```
/*
** Draws a simple rectangle
**
** @param urx Upper-right x coordinate on the screen
** @param ury Upper-right y coordinate on the screen
** @param llx Lower-left x coordinate on the screen
** @param lly Lower-left y coordinate on the screen
** @param color Pixel color to write. This is an index into the color palette
*/
static void __vga13_draw_rect(uint16_t urx, uint16_t ury, uint16_t llx,
    uint16_t lly, RGB_8 color)
{
    uint8_t color_code = __vga13_fetch_color(color);
    // we actually start drawing on the upper left pixel, so calculate the
    // address at that position first
    uint16_t* addr = (uint16_t*)(VGA13_MEM_BEGIN + ((ury * VGA13_WIDTH) + llx));
    // iterate in a smart fashion, reducing memory access

    // do a little preprocessing: how many times we can do chunk copies per
    // scanline. this reduces the number of memory writes we need to do as
    // these addresses are right next to each other
    uint8_t div = (urx - llx) / sizeof(uint16_t);
    uint8_t rem = (urx - llx) % sizeof(uint16_t);
    uint16_t packed_color = color_code;
    packed_color <<= 8; packed_color += color_code;

    // difference in address between end of current and start of next scanline
    uint16_t addr_diff = (VGA13_WIDTH - urx) + llx;

    // iterate over all scanlines
    for (uint16_t yi=0; yi<(lly - ury); yi++)
    {
        // draw the rectangles
        // iterate down the scanline
        for (uint16_t xi=0; xi<div; xi++)
        {
            *addr++ = packed_color;
        }
        // finish the remaining amount by writing bytes out
        for (uint16_t xi=0; xi<rem; xi++)
        {
            *((uint8_t*)addr) = color_code;
            addr++;
        }
        // advance the address to the start of the next scanline pixel to draw
        addr += (addr_diff / sizeof(uint16_t));
    }
}
```

HSC
Henrietta Semiconductor Corporation

Graphics Library Uses the Driver

```
RGB_8 color0 = color_map[c0_key - start_code];
RGB_8 color1 = color_map[c1_key - start_code];
// scale, taking account for the fact that 2 pixels are drawn at
// one time
uint16_t px_scale = 2 * scale;
uint16_t draw_w = run_len * px_scale;
uint16_t draw_h = scale;
// draw until far edge
if ((ul.x + x + draw_w) > vga_driver.screen_w)
    draw_w = vga_driver.screen_w - (ul.x + x);
// draw larger region if codes match and check for transparency
if (c0_key == c1_key)
{
    if (c0_key != t_code)
    {
        // draw with fast rectangles when possible; when both
        // colors are the same
        vga_driver.vga_draw_rect_wh(
            ul.x + x, ul.y + (y * scale),
            draw_w, draw_h, color0
        );
    }
}
```

HSC
Henrietta Semiconductor Corporation

Issues of Storing Images on a 16-bit Computer

- Biggest pain-point of the project
  - I can draw fairly nice images on the OS
  - I don’t have the memory to do so
- I am limited to 64kb of memory and have no filesystem
  - Image data must be built into the data
    - X-Pixel Map (XPM) is file format that can be included as C code

HSC
Henrietta Semiconductor Corporation

Solutions to Storing Images on a 16-bit Computer

- Image compression
  - Color quantization (reduce the number of colors in an image)
  - Run-length compression on sequences of same-valued pixels (CXPM)
- Image scaling
  - Store low-resolution images in the OS
  - Up-scale the image data when needed by expanding “pixel size”

HSC
Henrietta Semiconductor Corporation

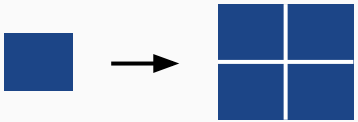
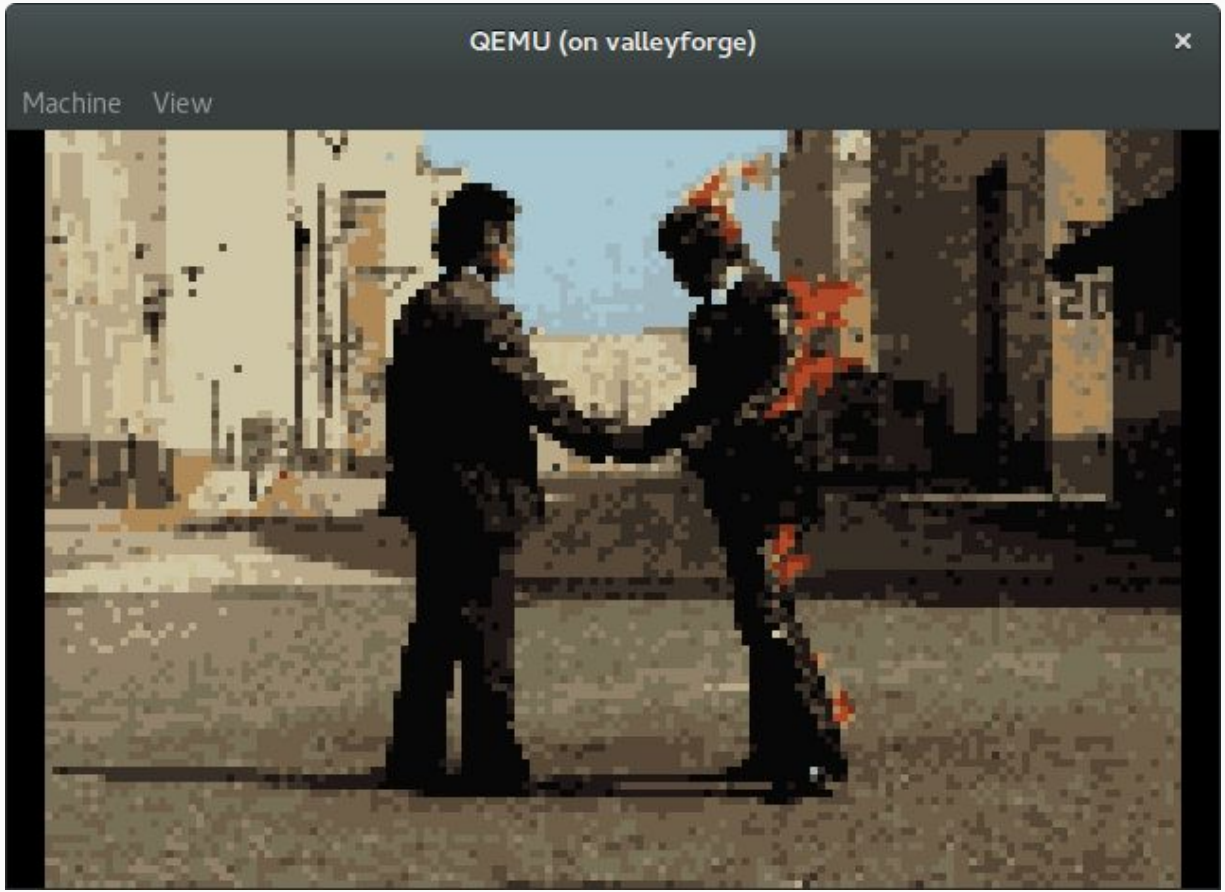
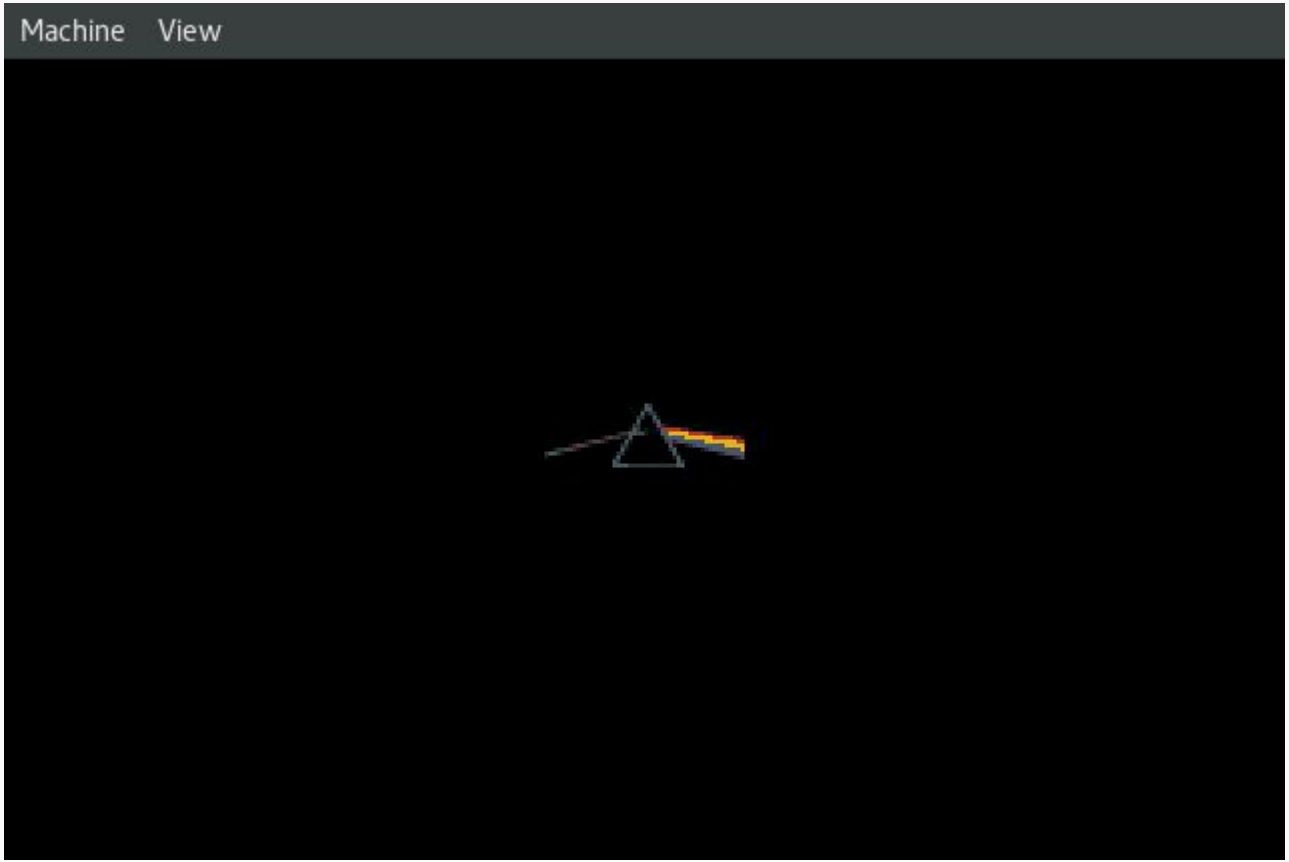


Image Rendering - Proof of Capability



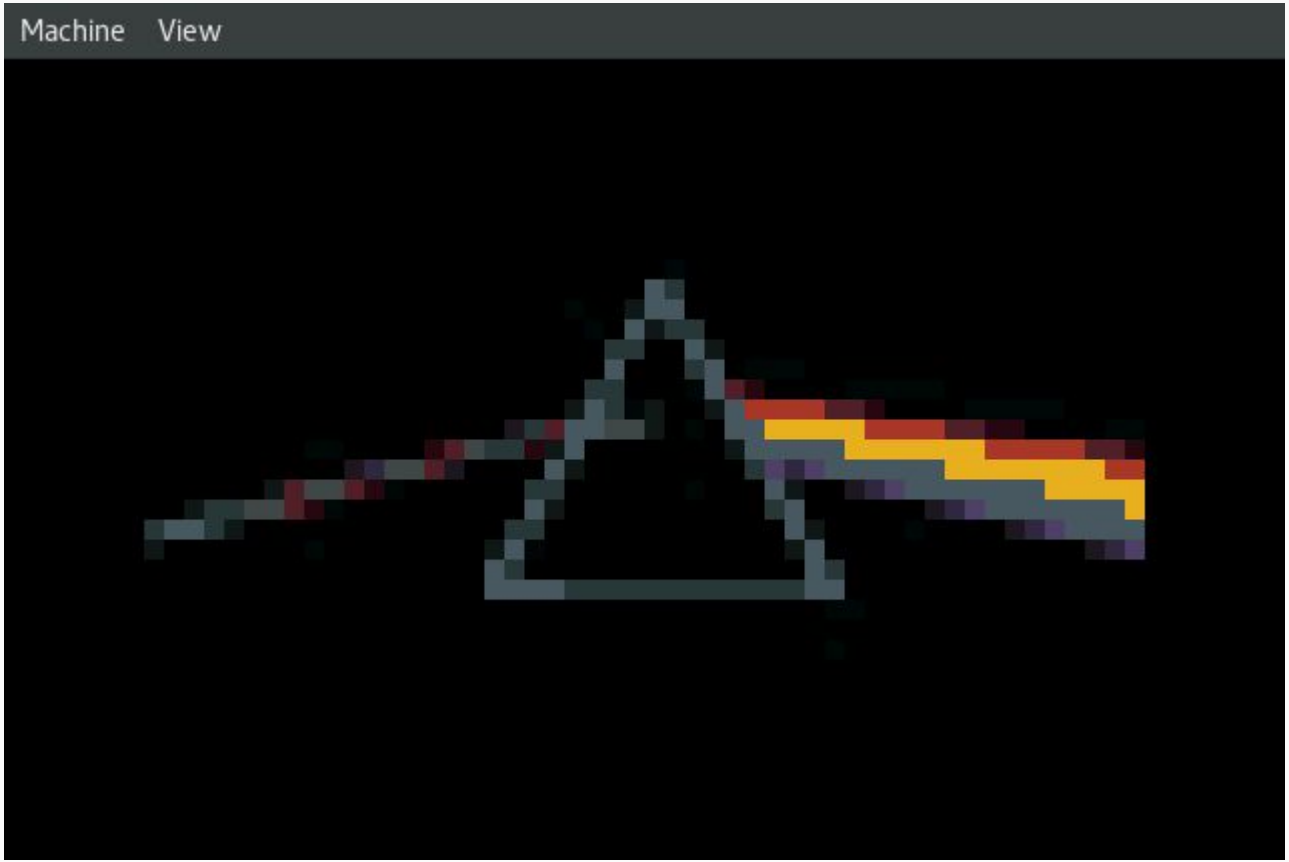
HSC
Henrietta Semiconductor Corporation

Image Rendering - Memory Limited, 50x50 Pixels



HSC
Henrietta Semiconductor Corporation

Image Rendering - Memory Limited, Scaled Up



HSC
Henrietta Semiconductor Corporation

Sources

[1] Image content comes from freely available online resources

[2] Diagrams and Code Snippets by Schuyler Martin

[3] HSC Logo created by Kailey Martin

[4] List of resources that were deemed helpful while making this project:
https://github.com/schuylermartin45/seegol/blob/master/docs/links.txt

HSC
Henrietta Semiconductor Corporation

Special Thanks

[1] Prof. Warren Carithers - Advisor
Warren, taught me almost everything I know about Systems Programming and Computer Graphics. Without him, none of this would be possible.

[2] Prof. Sean Strout - Mentor
Sean is a close friend of mine and initially sparked a lot of my interest in becoming a C wizard.

[3] Prof. Thomas Kinsman - Mentor
Thomas has taught me how to think creatively with visual problems

HSC
Henrietta Semiconductor Corporation

Questions?

Project available at https://github.com/schuylermartin45/seegol

HSC
Henrietta Semiconductor Corporation