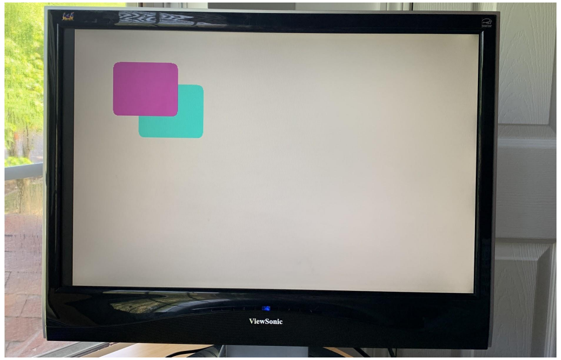
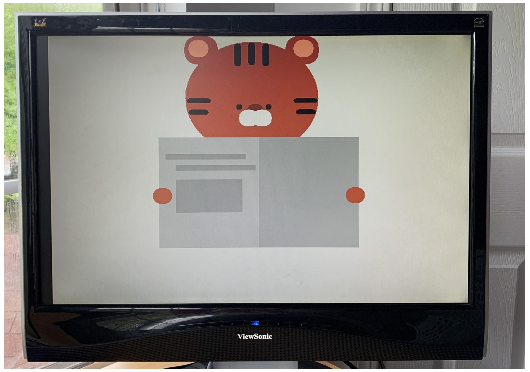




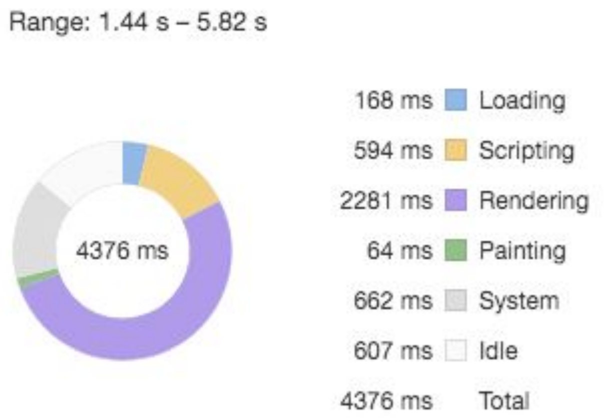
WebRender Presentation

A Simple Graphics Accelerator for Rounded Rectangles

Alex Gajewski and Allison Ghuman

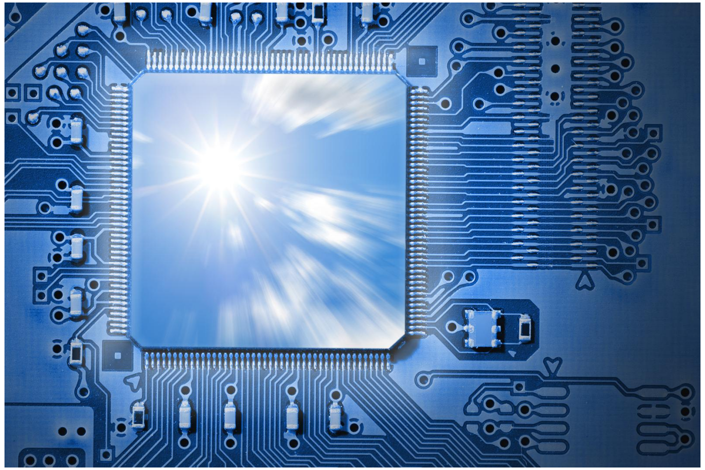
Motivation

- Rendering the web is slow
- Hardware is fast
- Goal: build a hardware accelerator to speed up common tasks in web rendering



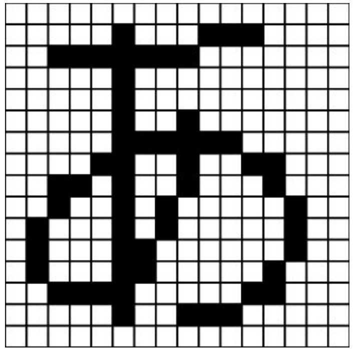
Range: 1.44 s – 5.82 s

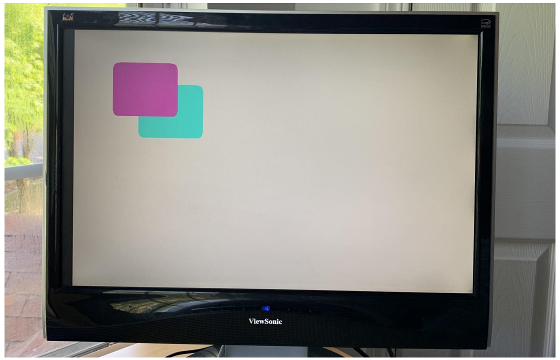
168 ms	🟦	Loading
594 ms	🟨	Scripting
2281 ms	🟪	Rendering
64 ms	🟩	Painting
662 ms	⬜	System
607 ms	⬜	Idle
4376 ms		Total



Approach: rounded rectangles

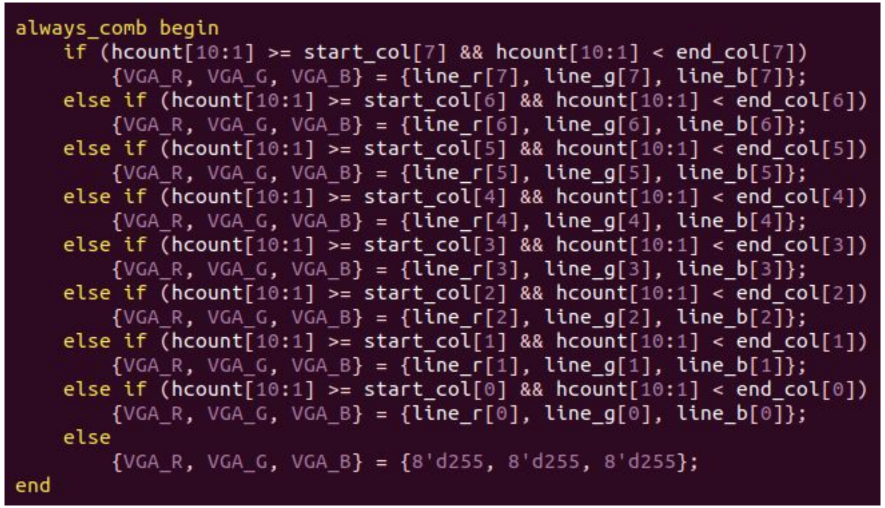
- Almost every object on the web falls into two categories:
  - Bitmaps
    - (really fonts these days are vectors, but at least they used to be bitmaps)
  - Rounded rectangles
- Goal for this project:
  - Build an accelerator that can layer rounded rectangles on top of each other as quickly as possible





Implementation

- Initially, we thought we would need a dual-framebuffer design:
    - Read from one framebuffer
    - Write to the other
    - Switch
- Actually, you can render pixels in real time, in a single cycle
    - Use block RAM to read an entire row's displaylist in one cycle
    - Purely combinatorial logic can compare the current column to the start and end column properties of the objects in the displaylist, and choose colors appropriately



```
always_comb begin
    if (hcount[10:1] >= start_col[7] && hcount[10:1] < end_col[7])
        {VGA_R, VGA_G, VGA_B} = {line_r[7], line_g[7], line_b[7]};
    else if (hcount[10:1] >= start_col[6] && hcount[10:1] < end_col[6])
        {VGA_R, VGA_G, VGA_B} = {line_r[6], line_g[6], line_b[6]};
    else if (hcount[10:1] >= start_col[5] && hcount[10:1] < end_col[5])
        {VGA_R, VGA_G, VGA_B} = {line_r[5], line_g[5], line_b[5]};
    else if (hcount[10:1] >= start_col[4] && hcount[10:1] < end_col[4])
        {VGA_R, VGA_G, VGA_B} = {line_r[4], line_g[4], line_b[4]};
    else if (hcount[10:1] >= start_col[3] && hcount[10:1] < end_col[3])
        {VGA_R, VGA_G, VGA_B} = {line_r[3], line_g[3], line_b[3]};
    else if (hcount[10:1] >= start_col[2] && hcount[10:1] < end_col[2])
        {VGA_R, VGA_G, VGA_B} = {line_r[2], line_g[2], line_b[2]};
    else if (hcount[10:1] >= start_col[1] && hcount[10:1] < end_col[1])
        {VGA_R, VGA_G, VGA_B} = {line_r[1], line_g[1], line_b[1]};
    else if (hcount[10:1] >= start_col[0] && hcount[10:1] < end_col[0])
        {VGA_R, VGA_G, VGA_B} = {line_r[0], line_g[0], line_b[0]};
    else
        {VGA_R, VGA_G, VGA_B} = {8'd255, 8'd255, 8'd255};
end
```

Hardware Interface

- 16-bit address space
  - First 10 bits specify row
  - Next 3 bits specify object number (up to 8 objects per row)
  - Last 3 bits specify property
    - 0: Start column
    - 1: Start column (last 2 bits)
    - 2: End column
    - 3: End column (last 2 bits)
    - 4: R
    - 5: G
    - 6: B
- Write all 8 objects sequentially
  - Saved to block RAM when the last object‘s B property is written

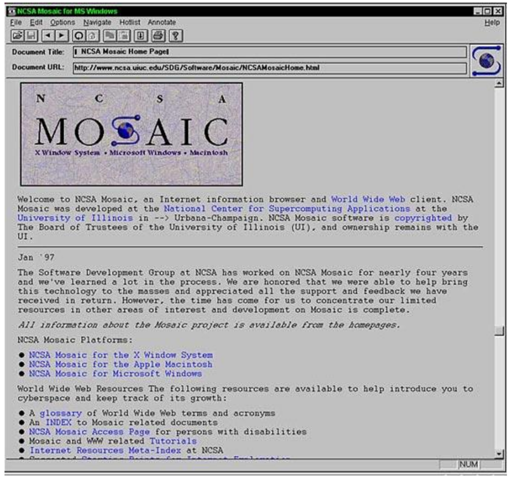
Software Interface

- Renders an array of rounded rectangles
- Rounded rectangles have a number of properties:
  - X position: distance from left edge of display, ignoring rounding
  - Y position: distance from top edge of display, ignoring rounding
  - Width, ignoring rounding
  - Height, ignoring rounding
  - Border radius
  - Red
  - Green
  - Blue
- Builds a displaylist from the array of rectangles, then writes to hardware and renders to display

It works :)

Extensions

- Add support for bitmaps
  - Can probably also be done in a single cycle for each pixel
  - Load all bitmaps for a given row during dead time
  - Add a new “bitmap-pointer” object type to the displaylist
- Use accelerator for a simple web browser
  - At least be able to render local HTML files with simple styling (maybe no font scaling)
    - Though font scaling could also be done during dead time
  - Possibly attach to internet as well (more of a networks project at that point)


NCSA Mosaic for MS Windows
File Edit Options Navigate Hotlist Annotate Help
N C S A MOSAIC
X Window System • Microsoft Windows • Macintosh
NUM

Document Title: NCSA Mosaic Home Page

Document URL: http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html

Welcome to NCSA Mosaic, an Internet information browser and World Wide Web client. NCSA Mosaic was developed at the National Center for Supercomputing Applications at the University of Illinois in --> Urbana-Champaign. NCSA Mosaic software is copyrighted by The Board of Trustees of the University of Illinois (UI), and ownership remains with the UI.

Jan '97

The Software Development Group at NCSA has worked on NCSA Mosaic for nearly four years and we've learned a lot in the process. We are honored that we were able to help bring this technology to the masses and appreciated all the support and feedback we have received in return. However, the time has come for us to concentrate our limited resources in other areas of interest and development on Mosaic is complete.

All information about the Mosaic project is available from the homepages.

NCSA Mosaic Platforms:

- NCSA Mosaic for the X Window System
- NCSA Mosaic for the Apple Macintosh
- NCSA Mosaic for Microsoft Windows

World Wide Web Resources The following resources are available to help introduce you to cyberspace and keep track of its growth:

- A glossary of World Wide Web terms and acronyms
- An INDEX to Mosaic related documents
- NCSA Mosaic Access Page for persons with disabilities
- Mosaic and WWW related Tutorials
- Internet Resources Meta-Index at NCSA