A deep dive into images on the web and then some...

Chen Hui Jing / @hj_chen
Surname: 陈
First name: 慧晶
Chen Hui Jing

@hj_chen
ELECTRONIC SIGNALS
Hi, I'm Pixel...
CRTs sold by Müller-Uri, Source: The Cathode Ray Tube site
Raster scanning

Source: M-SYS MV
<table>
<thead>
<tr>
<th></th>
<th>Raster scan</th>
<th>Random scan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electron beam</td>
<td>Swept across entire screen, one row at a time, from top-to-bottom</td>
<td>Only directed to parts of the screen where image is drawn</td>
</tr>
<tr>
<td>Resolution</td>
<td>Poor, due to plotting as discrete point sets</td>
<td>Good, as CRT beam directly follows line path</td>
</tr>
<tr>
<td>Picture definition</td>
<td>Stored as set of intensity values (pixels) in refresh buffer area</td>
<td>Stored as set of line drawing instructions in display file</td>
</tr>
<tr>
<td>Realism</td>
<td>Variable intensity values allow for realistic shadow and colour patterns</td>
<td>Most suited for line drawing</td>
</tr>
<tr>
<td>Drawing method</td>
<td>Screen points (pixels)</td>
<td>Mathematical functions</td>
</tr>
</tbody>
</table>

Source: Prof. Vijay M. Shekhat, CE Department, Computer Graphics
File formats

External metadata
- Mac OS type-codes
- Mac OS X UTIs
- OS/2 extended attr.
- POSIX extended attr.
- PUIDs
- MIME types
- FFIDs
- File content-based identification

Internal metadata
- File header / Magic number

File extension
Colour depth

1-bit PNG (2 colours)

2-bit PNG (4 colours)

4-bit PNG (16 colours)

8-bit PNG (256 colours)

24-bit PNG (16,777,216 colours)

Source: Wikipedia, Color depth
MacPaint

NOTE: The version number is actually a flag to MacPaint to indicate whether the brush/fill patterns are present in the file. If the version number is 0, the default patterns are used. Therefore you can simply save a file by writing a blank header (512 300 bytes), followed by the image data.

Bitmap compression:
The bitmap data is for a 576 pixel by 720 pixel monochrome image. The packing method is PackBits (see below). There are 72 bytes for each scan line. Each bit represents a pixel; 0 = white, 1 = black.
MS Paintbrush

Paintbrush for Windows 3.0

Paintbrush for Windows 3.1
Bitmap (BMP)

Bitmap and its corresponding colour table

Source: Microsoft, Types of Bitmaps
Graphics Interchange Format (GIF)

**GIF SIGNATURE**

The following GIF Signature identifies the data following as a valid GIF image stream. It consists of the following six characters:

```
G I F 8 7 a
```

The last three characters '87a' may be viewed as a version number for this particular GIF definition and will be used in general as a reference in documents regarding GIF that address any version dependencies.

Source: Graphics Interchange Format (tm)
Compression algorithms

Run-length Compression
20 Bytes → 16 Bytes

Lempel-Ziv-Welch (LZW) compression

Run length compression used by MacPaint

Motion graphics by Crystal Law
First band photo on the web

Source: The Cernettes
Joint Photographic Experts Group (JPEG)

1991 - 2016
25 YEARS
JPEG
JPG compression

Reference: How JPG works
Colourspace conversion

Image split into blocks

Reference: How JPG works
Quantisation

Reference: How JPG works
<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td>55</td>
<td>61</td>
<td>66</td>
<td>70</td>
<td>61</td>
<td>64</td>
<td>73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>59</td>
<td>55</td>
<td>90</td>
<td>109</td>
<td>85</td>
<td>69</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>59</td>
<td>68</td>
<td>113</td>
<td>144</td>
<td>104</td>
<td>66</td>
<td>73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>68-76</td>
<td>-122</td>
<td>-167</td>
<td>808</td>
<td>9376</td>
<td>-64</td>
<td>-55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>61-68</td>
<td>-102</td>
<td>-126</td>
<td>-38</td>
<td>639</td>
<td>7043</td>
<td>-59</td>
<td>-56</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>71-69</td>
<td>-605-66</td>
<td>-36</td>
<td>886</td>
<td>862</td>
<td>205</td>
<td>8225</td>
<td>5.94</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-49</td>
<td>-646-88</td>
<td>-5351</td>
<td>786137</td>
<td>624556</td>
<td>-28.91</td>
<td>9.93</td>
<td>5.42</td>
<td>-5.65</td>
</tr>
<tr>
<td></td>
<td>-43</td>
<td>-548-58</td>
<td>-89.073</td>
<td>3471063</td>
<td>14456</td>
<td>-10.24</td>
<td>6.30</td>
<td>1.83</td>
<td>1.95</td>
</tr>
<tr>
<td></td>
<td>-41</td>
<td>-492-59</td>
<td>-66.553-1922050</td>
<td>-3385</td>
<td>-1.87</td>
<td>1.75</td>
<td>-2.79</td>
<td>3.14</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Forward Discrete Cosine Transform

Reference: How JPG works
Statistical encoding

Reference: How JPG works
Progressive JPGs

Image source: What is a progressive JPEG?
JPG optimisation tips

1. Use high quality source material
2. Alignment on the 8x8 pixel grid
3. Reduce contrast and saturation
4. Sepia images
5. Slight blurring

Reference: Finally understanding JPG
3. Reduce contrast and saturation

Reference: Finally understanding JPG
2. Alignment on the 8x8 pixel grid

Reference: Finally understanding JPG
4. Sepia images

Una Kravets: CSS Blend Modes, Because...

Reference: Finally understanding JPG
5. Slight blurring

Reference: Finally understanding JPG
Speed, Quality, Size

Can’t have 'em all
JPG encoders, there are many

libjpeg

mozJPEG

Guetzli
Burn All GIFs day

November 5, 1999

https://burnallgifs.org/archives/
# PNG filter algorithms

<table>
<thead>
<tr>
<th>Filter Type</th>
<th>Formula</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>(--)</td>
<td>(52\ 55\ 61\ 66\ 70\ \rightarrow\ 52\ 55\ 61\ 66\ 70)</td>
</tr>
<tr>
<td>Sub</td>
<td>(\text{Sub}(x) = \text{Raw}(x) - \text{Raw}(x-\text{bpp}))</td>
<td>(52\ 55\ 61\ 66\ 70\ \rightarrow\ 52\ 55\ 61\ 66\ 70)</td>
</tr>
<tr>
<td>Up</td>
<td>(\text{Up}(x) = \text{Raw}(x) - \text{Prior}(x))</td>
<td>(52\ 55\ 61\ 66\ 70\ \rightarrow\ 52\ 55\ 61\ 66\ 70)</td>
</tr>
<tr>
<td>Average</td>
<td>(\text{Average}(x) = \text{Raw}(x) - \text{floor}\left((\text{Raw}(x-\text{bpp})+\text{Prior}(x))/2\right))</td>
<td>(52\ 55\ 61\ 66\ 70\ \rightarrow\ 52\ 55\ 61\ 66\ 70)</td>
</tr>
<tr>
<td>Paeth</td>
<td>(\text{Paeth}(x) = \text{Raw}(x) - \text{PaethPredictor}(\text{Raw}(x-\text{bpp}), \text{Prior}(x), \text{Prior}(x-\text{bpp})))</td>
<td>(52\ 55\ 61\ 66\ 70\ \rightarrow\ 52\ 55\ 61\ 66\ 70)</td>
</tr>
</tbody>
</table>
Portable Network Graphics (PNG)

8-byte signature of a PNG file
DEFLATE compression algorithm

DEFLATE Compressed Data Format Specification version 1.3
What a difference 2 pixels can make

Reference: pngthermal
Optimising PNG files

1. Reduce number of colours
2. Choose the right pixel format
3. Use indexed images, if possible
4. Optimise fully transparent pixels

Reference: Reducing PNG file Size
3. Use indexed images, if possible

Reference: Reducing PNG file Size
4. Optimise fully transparent pixels

Masked portion of image filled with single colour

Masked portion of image untouched

Reference: Reducing PNG file Size
**libpng**

Libpng is the official PNG reference library. It supports almost all PNG features, is extensible, and has been extensively tested for over 20 years. The home site for development versions (i.e., may be buggy or subject to change or include experimental features) is [https://libpng.sourceforge.io/](https://libpng.sourceforge.io/), and the place to go for questions about the library is the [png-mng-implent mailing list](https://lists.sourceforge.net/lists/listinfo/png-mng-implent).

Libpng is available as ANSI C (C99) source code and requires [zlib](https://zlib.net) 1.2.3 or later (1.2.5 or later recommended for performance and security reasons). The current public release, Libpng 1.6.36, fixes some build issues, adds a couple of small optimizations (ARM ppc, eon, sparcv9, etc.), and updates the license (identical terms to the zlib license, with the old license appended in the manner of the Python Software Foundation License version 3, and the list of contributing authors moved to a separate AUTHORS file).

---

**Portability Note**

The libpng 1.5.x, 1.6.x, and upcoming 1.7.x series continue the evolution of the libpng API, finally hiding the contents of the variable and handy `png_struck` and `png_info` data structures inside private (i.e., non-installed) header files. Instead of direct structure access, applications should be using the various `png_get_*` and `png_set_*` accessor functions, which have existed for almost as long as libpng itself.

The portability notice should not come as a particular surprise to anyone who has added libpng support to an application this millennium; the manual has warned of it since at least July 2000. (Specifically, starting with version 1.2.10 onwards, both structures are going to be hidden, and the contents of the structures will only be accessible through the `png_get_*`/`png_set_*` functions.) Ok, so the version number was off a bit... and the grammar, too, but who’s counting? Those whose apps depend on the older API need not panic, however (for now); libpng 1.6.x continues to get security fixes, so has 1.0.x for well over a decade. (Greg no longer bothers to list either series here; enough’s enough, folks. Update those apps now!)

The 1.5.x and later series also include a new, more thorough test program (pngtest.tst) and a new `pnglibconf.h` header file that tracks what features were enabled or disabled when libpng was built. On the other hand, they no longer internally include the zlib.h header file, so applications that formerly depended on zlib.h to provide that will now need to include it explicitly. Complete differences relative to Libpng 1.4.x are detailed [here](https://libpng.sourceforge.io/).

See the bottom of this page for warnings about security and crush bugs in versions up through Libpng 1.6.31.

In addition to the main library sources, all of the 1.2.x/1.4.x/1.5.x/1.6.x/1.7.x series include the `png, rng-test, and wring` demo programs, the paginum demo program, a subset of Wilhelm van Schaft’s [Paginum test images](https://libpng.sourceforge.io/), and Wilhelm’s [VisualPNG](https://libpng.sourceforge.io/) demo program.

What is a browser engine? 😐

Source: Quantum Up Close: What is a browser engine?
What is a browser engine? 😊

Source: Quantum Up Close: What is a browser engine?
Image encoders in browsers

Chromium

Gecko
Reference: Let's build a browser engine!
Rasterisation (1/3)

Simple rasteriser for painting rectangles

```rust
pub struct Canvas {
    pub pixels: Vec<Color>,
    pub width: usize,
    pub height: usize,
}

impl Canvas {
    /// Create a blank canvas
    fn new(width: usize, height: usize) -> Canvas {
        let white = Color { r: 255, g: 255, b: 255, a: 255 };
        Canvas {
            pixels: vec![white; width * height],
            width: width,
            height: height,
        }
    }
    // ...
}
```

Source: Let’s build a browser engine!
fn paint_item(&mut self, item: &DisplayCommand) {
    match *item {
        DisplayCommand::SolidColor(color, rect) => {
            // Clip the rectangle to the canvas boundaries.
            let x0 = rect.x.clamp(0.0, self.width as f32) as usize;
            let y0 = rect.y.clamp(0.0, self.height as f32) as usize;
            let x1 = (rect.x + rect.width).clamp(0.0, self.width as f32) as usize;
            let y1 = (rect.y + rect.height).clamp(0.0, self.height as f32) as usize;

            for y in y0..y1 {
                for x in x0..x1 {
                    self.pixels[y * self.width + x] = color;
                }
            }
        }
    }
}

Source: Let’s build a browser engine!
Rasterisation (3/3)

Simple rasteriser for painting rectangles

```rust
/// Paint a tree of LayoutBoxes to an array of pixels.
pub fn paint(layout_root: &LayoutBox, bounds: Rect) -> Canvas {
    let display_list = build_display_list(layout_root);
    let mut canvas = Canvas::new(bounds.width as usize, bounds.height as usize);
    for item in display_list {
        canvas.paint_item(&item);
    }
    canvas
}

Source: Let's build a browser engine!
```
"The graphics backend is an array of bytes and a for loop that goes and says: go from the left edge of the rectangle to the right edge and write the same colour over and over again into this array.

—Matt Brubeck, Bay Area Rust Meetup (Nov 2014)"

\_/\(∩°)∩\_/

"
Graphics libraries

- Skia
- WebRender
- Core Graphics
- Cairo
- Pango
Painting

Frame buffer

255 255 255 255
255 255 255 255
255 255 255 255
68 95 238 255
.
.
.

Reference: The whole web at maximum FPS: How WebRender gets rid of jank
Compositing (1/2)

Source bitmaps

Destination bitmap

Reference: The whole web at maximum FPS: How WebRender gets rid of jank
Compositing (2/2)

Reference: GPU Accelerated Compositing in Chrome
Making use of the GPU (1/2)
Making use of the GPU (2/2)

Reference: Hardware acceleration and compositing
GPU rasterisation in Chromium

Reference: Software vs. GPU rasterization in Chromium
If you could have a do-over...

What if we actually need a butterfly instead?
WebRender (1/2)

The whole web at maximum FPS: How WebRender gets rid of jank by Lin Clark
WebRender (2/2)

https://hacks.mozilla.org/2017/10/the-whole-web-at-maximum-fps-how-webrender-gets-rid-of-jank/

https://mozillagfx.wordpress.com/
Acknowledgements

🙏 Big thank you to these beautiful human beings who answered my noob questions 🙏

@bmeurer  @callahad  @slsoftworks  @linclark  @g33konaut  @mathias  @nicalsilva  @potch
References

- The GIF is Dead. Long Live the GIF.
- Types of Bitmaps
- Why do we need JPEG compression and how it’s technically working? by Steven Hansen
- Progressive JPEGs and green Martians by Jon Sneyers
- Finally understanding JPG by Christoph Erdmann
- How JPG Works, How PNG Works, Reducing PNG file Size by Colt McAnlis
- Thoughts on a GIF-replacement file format
- Quantum Up Close: What is a browser engine? by Matt "Potch" Claypotch
- Let’s build a browser engine! by Matt Brubeck
- On rendering engines and graphic libraries by Kilian Valkhof
- Following up on the 2d graphics in Rust discussion by Nicolas Silva
- Introduction to WebRender – Part 1 – Browsers today by Nicolas Silva
- The whole web at maximum FPS: How WebRender gets rid of jank by Lin Clark
- Software vs. GPU rasterization in Chromium* by Martina Kollarova
- GPU Accelerated Compositing in Chrome by Tom Wiltzius, Vangelis Kokkevis & the Chrome Graphics team
This talk is dedicated to browser engineers everywhere.
I owe my career to you.
Thank you!

https://www.chenhuijing.com

@hj_chen

@hj_chen

@huijing

Font used is Signika by Anna Giedryś