WebP - Faster Web with smaller images

Pascal Massimino
WebP

New image format - Why?

- Average page size: 350KB
- Images: ~65% of Internet traffic

Current image formats

- JPEG: 80% of image bytes
- PNG: mainly for alpha, lossless not always wanted
- GIF: used for animations (avatars, smileys)

WebP: more efficient unified solution + extra goodies

Targets Web images, not at replacing photo formats.
WebP

● Unified format
  ○ Supports both lossy and lossless compression, with transparency
  ○ all-in-one replacement for JPEG, PNG and GIF

● Target: ~30% smaller images

● Low-overhead container (RIFF + chunks)
WebP-lossy with alpha

Appealing replacement for unneeded lossless use of PNG: sprites for games, logos, page decorations

- YUV: VP8 intra-frame
- Alpha channel: WebP lossless format
  - Optional pre-filtering (~10% extra compression)
  - Optional quantization --> near-lossless alpha
- Compression gain: 3x compared to lossless
WebP - Lossless Techniques

- More advanced spatial predictors
- Local palette look up
- Cross-color de-correlation
- Separate entropy models for R, G, B, A channels
- Image data and metadata both are Huffman-coded

Still is a very simple format, fast to decode.
**WebP vs PNG**

*source: published study on developers.google.com/speed/webp*

**Average: 25% smaller size**
(corpus: 1000 PNG images crawled from the web, optimized with pngcrush)
Speed number (takeaway)

Encoding

- Lossy (VP8): 5x slower than JPEG
- Lossless: from 2x faster to 10x slower than libpng

Decoding

- Lossy (VP8): 2x-3x slower than JPEG
- Lossless: ~1.5x faster than libpng

Decoder's goodies:

- Incremental
- Per-row output (very low memory footprint)
- on-the-fly rescaling and cropping (e.g. critical for Android)

and: hardware-compliant for VP8
Web-centric benchmark

Demo!

Lossless offers better compression ratio than lossy for low-color source

=> Lossless for screen-capture! (Next-Gen)

Latency video
## Sample Images

<table>
<thead>
<tr>
<th>Image</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td>WebP-Lossless: 20.1 KB (default Quality)</td>
</tr>
<tr>
<td><img src="image2.png" alt="Image" /></td>
<td>Compared to PNG source: 25.6 KB</td>
</tr>
<tr>
<td><img src="image3.png" alt="Image" /></td>
<td>Lossy: $Q = 100$, 14.9 KB</td>
</tr>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td>Lossy: $Q = 75$, 8.2 KB</td>
</tr>
<tr>
<td><img src="image5.png" alt="Image" /></td>
<td>Lossy: $Q = 50$, 7.4 KB</td>
</tr>
<tr>
<td><img src="image6.png" alt="Image" /></td>
<td>Lossy: $Q = 25$, 6.5 KB</td>
</tr>
</tbody>
</table>
Support for WebP

- Platforms and browsers
  - WebKit, Android ICS+, iOS XCode, Chrome, Opera

- Google properties
  - Gmail, Picasa, Docs, Google+

- Graphics software
  - Gimp, Pixelmator, ImageMagick, ReaConverter, Konvertor, XnView, IrfanView, GDAL, Adobe Photoshop CS5 and Windows Photo Viewer
Release Plan

0.2.0 (next week):
● frozen bitstream format (lossless / lossy / alpha)

0.2.1 (soon after):
● simple useful features at container level:
  ○ low-overhead (RIFF + chunks)
  ○ Metadata
  ○ Animation
  ○ Tiling

Afterward:
● Spriting (<img src="group_photo.webp#john">)
● Layers and multi-resolution (large screen capture!)
Container features of WebP

- **Animation**
  - A good alternative to GIF

- **Metadata**
  - Image metadata (format agnostic: XMP, EXIF etc.)

- **ICC Color Profile**

- **Tiling**
  - Overcomes size limit of a single bit stream
  - May improve efficiency for larger images
    - e.g., grass can be encoded differently than sky.

- **Spriting**
  - Combine numerous small images into one
  - Reduces the number of requests by the browser
More Info

Main page:
https://developers.google.com/speed/webp/

Try it out!
https://developers.google.com/speed/webp/download

Mailing list:
webp-discuss@webmproject.org

Coming next week: 0.1.99 release, finalized bitstream format.