Alpha transparency in Chrome video

By Sam Dutton

Sam is a Developer Advocate

Alpha transparency in Chrome video

Chrome Canary now supports video alpha transparency in WebM.

In other words, Chrome takes the alpha channel into account when playing 'green screen' videos encoded to WebM with an alpha channel. This means you can play videos with transparent backgrounds: over web pages, images or even other videos.

There's a demo at simpl.info/videoalpha. This only works in Chrome Canary at this point, and you'll need to enable VP8 alpha transparency from the chrome://flags page. Somewhat surreal, and a bit rough around the edges (literally) but you get the idea!

How to make alpha videos

The method we describe uses the open source tools Blender and ffmpeg:

1. Film your subject in front of a single colour background such as a bright green curtain.
2. Process the video to build an array of PNG still images with transparency data.
3. Encode to a video format (in this case, WebM).

There are also proprietary tools to do the same job, such as Adobe After Effects, which you may find simpler.

1. Make a green screen video

First of all, you need to film your subject in a way that everything in the background can be 'removed' (made transparent) by subsequent processing.

The easiest way to do this is to film in front of a single colour background, such as a screen or curtain. Green or blue are the colors most often used, mostly because of their difference from skin tones.

There are several guides online to filming green screen video (also known as chroma key) and lots of places to buy green and blue screen backdrops. Alternatively, you can paint a background with chroma key paint.

The Great Gatsby VFX reel shows just how much can be accomplished with green screen.

Some tips for filming:

- Ensure your subject does not have clothes or objects that are the same color as the backdrop, otherwise these will show up as 'holes' in the final video. Even small logos or jewelry can be problematic.
- Use consistent, even lighting, and avoid shadows: the aim is to have the smallest possible range of colors in the background that will subsequently need to be made transparent.
- Using multiple diffused lights helps to avoid shadows and background color variations.
- Avoid shiny backgrounds: matte surfaces diffuse light better.

2. Create raw alpha video from green screen video

The following steps describe one way to create a raw alpha video from green screen videos:

1. Once you've shot a green screen video, you can use an open source tool like Blender to convert the video to an array of PNG files with alpha data. Use Blender's color keying to remove the green screen and make it transparent. (Note that PNG is not compulsory: any format that preserves alpha channel data is fine.)
2. Convert the array of PNG files to a raw YUVA video using an open source tool such as ffmpeg:
   `ffmpeg -i image%04d.png -pix_fmt yuva420p video.raw`

   Alternatively encode the files directly to WebM, using an ffmpeg command like this:
   `ffmpeg -i image%04d.png output.webm`

If you want to add audio, you can use ffmpeg to mux that in with a command like this:

```
ffmpeg -i image%04d.png -i audio.wav output.webm
```

3. Encode alpha video to WebM

Raw alpha videos can be encoded to WebM in two ways.

1. With ffmpeg: we added support to ffmpeg to encode WebM alpha videos.

   Use ffmpeg with an input video including alpha data, set the output format to WebM, and encoding will automatically be done in the correct format as per the spec. (Note: you'll currently need to make sure to get the latest version of ffmpeg from the git tree for this to work.)

   Sample command:
   `ffmpeg -i myAlphaVideo.webm output.webm`
2. Using webm-tools:
   `git clone http://git.chromium.org/webm/libvpx.git`

   webm-tools is a set of simple open source tools related to WebM, maintained by the WebM Project authors, including a tool for creating WebM videos with alpha transparency.

   Run the binary with `--help` to see list of options supported by alpha_encoder.

4. Playback in Chrome

To play the encoded WebM file in Chrome, simply set the file as the source of a video element.

As of now, VP8 alpha playback is behind a flag, so you have to either enable it in about:flags or set the command line flag `--enable-vp8-alpha-playback` when you start Chrome. When the flag is enabled, alpha playback also works with MediaSource.

How did they do it?

We talked to Google engineer Vignesh Venkatasubramanian about his work on the project. He summarised the key challenges involved:

- The VP8 bitstream had no support for alpha channel. So we had to incorporate alpha without breaking the VP8 bitstream and without breaking existing players.
- Chrome's renderer was not capable of rendering videos with alpha.
- Chrome has multiple rendering paths for multiple hardware/GPU devices. Every rendering path had to be changed to support rendering of alpha videos.

We can think of lots of interesting use cases for video alpha transparency: games, interactive videos, collaborative story telling (add your own video to a background video/image), videos with alternative characters or plots, web apps that use overlay video components...

Happy film making! Let us know if you build something amazing with alpha transparency.

Last updated July 2, 2018.