Gulp and avif, part 2

Using the default encoder and settings, as described in Compressing AVIF images with Gulp, got the task running and produced the result we wanted: AVIF images that were smaller than the source image in either JPG or PNG format.

This post will start from there and explore some further considerations on using `avifenc` to create images for the web.

Unlike the previous post, the tasks here will try to mirror the following CLI commands.

```
# Default AOM encoder
avifenc --codec aom \
--min 40 --max 40 \
--jobs 4 \
--speed 10 \
--output squosh3.avif \
images/squosh.png
```

The first pass of the task looks like this.

```
gulp.task('avif-aom', function() {
  const destination = './converted'
  retu'./converted'./images/*')
    .pipe(exec((file) =>
        `avifenc --min 40 --max 40     --jobs 4 --speed 10 \
        ${file.path} --out ${destination}/${file.path}`
      ))
    .pipe(rename({ extname: '.avif'}))
    .pipe(gulp.dest(`${destination}`))
});
```

The task kept crashing on execution but it would allow me to run `gulp -T` so the error was in the command I was executing, not Gulp.

I removed all the parameters to `gulp-exec` after avifenc and added them

back one by one. The `--speed` parameter is the guilty one. I was able to remove it and leave it working at the default value.

The final task looks like this:

```
gulp.task('avif-aom', function() {
  const destination = './converted'
  retu'./converted'./images/*')
    .pipe(exec((file) =>
        `avifenc --min 40 --max 40 --jobs 4 ${file.path} --out ${destinat
     ))
    .pipe(rename({ extname: '.avif'}))
    .pipe(gulp.dest(`${destination}`))
});
```

Using a different encoder

If you compile it to do so, you can choose different encoders to create AVIF images. We want to replicate the following command line script.

```
avifenc -c rav1e \
--min 40 --max 40 \
--jobs 4 \
--speed 10 \
--output squosh2.avif \
images/squosh.png
```

I was expecting that adding the encoder flag (`--codec rav1e`) would cause the same problem as the `avif-aom` where the speed parameter would cause the same problem. It did not.

```
gulp.task('avif-rav1e', function() {
  const destination = './converted'
  return'./converted'images/*')
    .pipe(exec((file) =>
        `avifenc --codec rav1e --min 40 --max 40 --jobs 4 --speed 10 ${fi
```

```
      ))
    .pipe(rename({ extname: '.rav1e.avif'}))
    .pipe(gulp.dest(`${destination}`))
});
```

This task will take significantly longer than the previous task, even when set up to work at maximum speed.

The one change I'll make is remove the speed indication to make sure both tasks run as evenly as possible. The task now looks like this

```
gulp.task('avif-rav1e', function() {
  const destination = './converted'
  return'./converted'images/*')
    .pipe(exec((file) =>
        `avifenc --codec rav1e --min 40 --max 40 --jobs 4 ${file.path}  $
      ))
    .pipe(rename({ extname: '.rav1e.avif'}))
    .pipe(gulp.dest(`${destination}`))
});
```

Comparing the enconders

Although I wouldn't expect a different in file sizes it's still instructive to compare the two encoders.

To do this I've created a default task that will run both encoders in parallel

```
gulp.task('default', gulp.series(
  'avif-aom',
  'avif-rav1e'
));
```

Further exploration

There are a couple more items that I want to research but they are not essential for encoding images. The two that come to mind are:

- Will setting up a range, making `--min` smaller than `--max` change the encoding results?
- Will adding more workers reduce encoding time?

I have the code to test it. Will report in a future post.