Propeller Recruitment

Take Home Challenge

Full Stack Engineer

Propeller helps worksites measure and manage themselves. Real-time, real accurate data has the potential to revolutionize mining, construction, and civil engineering. Cloud computing and drone surveying can make that worksite data accessible and usable for the people who need it to improve safety, report accurately, boost collaboration and efficiency, and shave days or weeks off typical workflows.

Image tiler

Problem description

Our visualiser has a feature that allows customers to view the source photos that they uploaded as part of a survey. One survey may consist of several thousand large (30+ MB) photos. To make viewing snappy, we cut each photo up into tiles. Tiles are then generated for the photo at full resolution, at half resolution, at 1/4 resolution, and so on, until the photo cannot be shrunk further (i.e. its resolution is 1x1). For a photo with the resolution n x m, there will be
$L = 1 + \lceil \log_2 \max(n, m) \rceil$
levels of tiles; each level will cover the whole photo, but on the highest (lowest-numbered) level, the original image will be shrunk down to 1x1 pixels. By convention the levels start at 0 and go "down" to level L-1, which has the most number of tiles that together have as many pixels as the original photo. For the purposes of the following, assume that the tile size is 256x256.

How does all these little tiles help? Well, say that the photo is something like 7000x5000 pixels and we are viewing it in a viewport that is 1000x800 (0.8 megapixels). The browser can then choose to load the 4x4 tiles from level 10, totalling 0.8 megapixels, instead of the full photo at 35 megapixels. It also means that when zooming into different parts of the image, we only have to load higher-resolution imagery for that part, and not for the other.

Your job is to tile the photos. Some considerations:

- Use an image library for your chosen language. Make the installation/setup as repeatable as possible so that we have a chance to get it to work too :)
- Tiles are always 256x256, except for the edges: if the photo is 1025x766, you'll end up with one-pixel slices to the right and you'll be a couple of pixels short at the bottom. That is fine.
- Your command-line utility should run on some *nix, preferably Mac OS. Otherwise, let us know what you ran it on.
- Your utility takes one argument, the source file, and it writes a "pyramid" of tiles to an appropriately named directory in the same directory as the source file. Name your files "L/x_y.jpg" where
    - L is the tile level (start at 0 for the most zoomed-out level),
    - x is the tile's x coordinate and
    - y is the tile's y coordinate. Coordinates start at 0,0 in the top-left corner.

Propeller Recruitment

Your solution

We are going to look at how you have solved the problem, how you have tested your solution, what tradeoffs you seem to have made, how easy it is to get it up and running, how well it performs and of course that it does what it should. That is a long list of things, and we are aware of the fact that your time is limited. Therefore, please let us know some of the tradeoffs that you have made, what you have focussed on and what you have ignored for now.

Cat: example image

The Internet is full of cats, which is why it is important that your solution works well when tiling cat images.