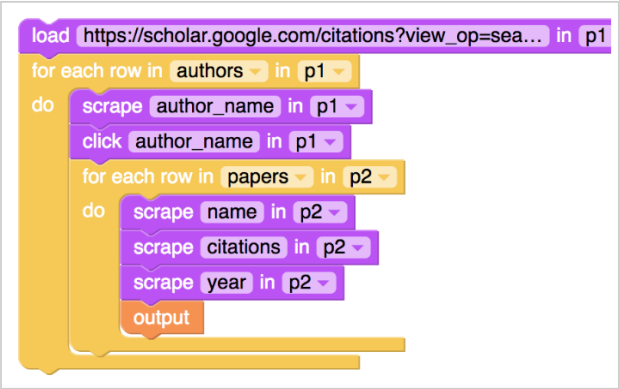
PHOTOGRAM

Processing photos is a daily task for not only professional photographers but also photography lovers. Unlike the film age where the amount of photos to be developed is limited by the amount of negatives photographers process and the slow process of setting up the camera, people nowadays take photos with a simple press on the shutter without the limitation of the amount of photos taken thanks to the large memory cards. This results in the huge amount of photos to be developed and processed every day.

The current technology has been made this process easier. With the Adobe Lightroom and Photoshop, people are able to make detailed edits on photos quickly but there are many limitations. First, Lightroom allows users to do the same edition to many pictures at the same time with presets, but it is not programmable, which means you have to manually choose presets for different sets of photos. Second, Photoshop enables users to use “actions” to process individual photo, but it cannot be applied to a group of photo, and it is difficult to use because it requires the abilities to write scripts. Third, many tasks of dealing with photos are repetitive but requires different actions according to different conditions. One example of this is watermarking. Watermarks are always put in the corners of the pictures, but which corner to put the watermark on so that it does not affect the viewing of the photo yet still visible is a problem, though very trivial, that needs human beings to handle manually. Such limitations are posing heavy burdens on photographers and reduces people’s passion on photography.

Our motivation is to pull photographers away from such repetitive works with a easy-to-learn programming tool. Our approach is to create a programming language or tool with a block-based GUI that processes massive images for photographers. The features of such a language or a tool is similar to those of other programming languages, which includes but not limited to basic conditional statements like loops and if/else statements, classes, function calls, and data structures like arrays. To make it a handy tool for photographers, we will also have image processing functions built in, like resizing, rescaling, watermarking, and layering. To help photographers with no programming experiences, we enable a trivial block-based GUI that users can drag around the code.



One example of a block-based GUI is “Helena”(http://helena-lang.org/)

There are also multiple challenges and risks in various ways to implement such a language or tool. First, it is a programming language after all, making it easy to use for non-programmers is a difficult tasks that needs to be considered a lot. Even though it has a block-based GUI, the condition statements, classes, and functions can still be challenging to understand. The implementation of GUI for data structures is also one thing that requires attention because it requires some sort of visualization. Second, how to process the images is a difficult task for developers. The problem is not only how to realize the features of in depth image development like what they have in Adobe Lightroom or Photoshop, but also how to make different development presets for each individual photo. Basic uniform processing can potentially be done by the integration with Adobe but it potentially requires the possession of the licence to use Adobe Softwares. More in depth development requires the understanding of image processing algorithms and even machine learning algorithms for identifying different conditions of photos for further development. This challenge really depends on the scale of the coverage of such a programming tool, whether it just covers the processing of simple processing for large amount of images and leave the creative, complicated work to the photographers, or it covers everything, including in depth processing of each photo. However, this does not mean that such a tool is not possible to implement because there already exists multiple plug-ins for different functions, these challenges can be minimized by integrating with existing algorithms and plug-ins.

We spent 6 hour in total brainstorming ideas and dive deeper in the selected idea.