**Simple Image Manipulation in C/C++  
ECE 847 Assignment #1**

**Purpose:** In this assignment you will learn to perform some simple image manipulation in C/C++. You will gain familiarity with the Blepo library and with Visual Studio.

**Before you start:** Watch the Lecture 0 series.

**Background info:** Image processing courses are often taught in Matlab. While this approach simplifies the learning curve, it hides important details and thus leaves students ill-equipped for the real world. In industry, images are almost always manipulated in a language like C/C++ to allow efficient processing of pixel data. While the learning curve for C/C++ is greater, the time spent learning to program in this manner (e.g., having to pay careful attention to memory accesses), is well worth it in the end because it empowers you to implement algorithms that are just not possible in Matlab. So don’t be discouraged: When you encounter bumps along the road, keep in mind that at the end of the semester you will be glad (along with countless students from previous years) that you went through this extra effort.

**Instructions:**

1. Download and install Microsoft’s Visual Studio 2015. As long as you have a valid clemson.edu email address (which you all should have) you will be able to download this program for free from Dreamspark: <https://www.dreamspark.com/> . Visual Studio is probably the most popular IDE (integrated development environment) for programming in C++, and it provides many helpful features for making your life easier when programming and debugging. In addition, we need everyone to work in the same environment to simplify the grading. If you do not have access to a Windows PC, you will have to either set it up for dual-boot, use a VM, or use a computer in the library or computer lab.
2. Download the Blepo library file (**blepo\_0.6.10\_for\_ece847\_vs2015.zip** in Blackboard), move it to where you want it to be on your system, and unzip the file (in the File Explorer, right click and select *Extract all*). Blepo is a library that we have developed at Clemson to make it easier to manipulate images in C/C++, but note that this zip file is a special version of Blepo for the class that is not available from the public Blepo webpage. Strictly speaking, you are not required to use Blepo (all that is required is that it compile in Visual Studio with minimal effort from the grader), but nearly all students do.
3. To set up Blepo, you need to do one step: Put the *blepo/external/bin* directory in your path (*Control Panel.System.Advanced system settings.Environment variables.Path.Edit*). If you need more detail, see the Blepo download page <http://www.ces.clemson.edu/~stb/blepo/download.html>
4. Run Visual Studio and open the Blepo demo solution (*blepo/demo/demo.sln*). (If Visual Studio prompts you whether you want to upgrade the project, select *Upgrade*.) In the Solution Explorer pane you should see the *Demo Solution* along with the two projects it contains: *Blepo* and *Demo*. Rightclick on the *Demo* project and select *Set as startup project*, then select *Build.Build solution*. Ignore any warnings, but if you get any errors, then you will need to doublecheck your steps above, then consult the Blepo download page, which contains a list of common errors that students have encountered over the years; if your error is not in the list, then you will need to contact the instructor.
5. Now run the demo by selecting *Debug.Start without Debugging*. When you get the demo to run, press some of the buttons (the first time the file open dialog box displays, you may have to navigate to the *blepo/images/* folder) to try out some of the functionality. The source code for all these callbacks is in *DemoDlg.cpp* if you want to take a look. When you are done, close the demo window.
6. If you have gotten this far, then you are in good shape. At this point, in past years we would have asked you to create your own project (instructions are on the Blepo download page) so you could begin your assignment. However, creating your own project is additional work for you, and it leads to non-uniformity in the assignments turned in, which causes additional work for the grader. So instead open *blepo/homework/homework.sln* in Visual Studio (as before, select *Update* if you are prompted to do so), build the solution, and run *homework* just as you did for *demo*. It should load a file and display it in a window on the screen. Note that this is a console-based application, so if you have programmed in C before, the code should look familiar to you, because it has a console window from which you can parse command-line parameters and to which you can *printf()*.
7. Now you are done with the setup and are ready to begin your actual assignment. In this assignment you will write an application (that is, you will modify the code in *homework*) that performs the following steps in order when it is run:
   1. Reads 3 command-line parameters, which we will call *filename1*, *filename2*, and *filename3*.
   2. Loads *filename1* from the *blepo/images* directory into a BGR image and displays it in a figure window.
   3. Creates a synthetic grayscale image of the same size as *filename1* with each pixel in a 100x100 square region centered in the image set to the value 255, and all other pixels set to 0. If *filename1* is smaller than 100x100, the application sets all pixels in the synthetic image to 255.
   4. Displays the synthetic image in a separate figure window, and saves the synthetic image to *filename2* in the *blepo/images* directory.
   5. Loads *filename3* from the *blepo/images* directory into a grayscale image and displays it in a separate figure window. (Note: If *filename3* is the same as *filename2*, then the square that you just created is loaded.)
   6. Masks the BGR image with the grayscale image just loaded. That is, every pixel in the BGR image whose corresponding pixel in the grayscale image is 0 is set to black, (0,0,0), while all other pixels are left untouched. If the BGR and grayscale images are different sizes, then an error is printed.
   7. Displays the resulting masked image in a separate figure window.
8. You should not only verify that the images in the figure windows look correct, but also that the images you saved to disk are correct. I suggest you download and install Irfanview, <http://www.irfanview.com/> , which is a free and powerful image viewer. Irfanview will scan all the images in a directory and allow you to navigate through them easily with the spacebar and backspace key, making it easy to view a set or sequence of images. Irfanview also supports batch conversion, so that you can quickly convert an entire directory of JPEG images to BMP (for example), downsampling in the process (if you desire).
9. Submit your code to the grader. First, make a copy of your *homework* directory for safekeeping (be sure to do this!). Then close Visual Studio and delete the *Debug* and *Release* (and *ipch* if it exists) folders in *homework*, as well as any temporary files with extension .aps, .clw, .ncb, .opt, .plg, .suo, .sdf. Now run Visual Studio and open your *homework.sln* file again, compile and run to verify that any files you deleted were indeed not important for the grader. If all is well, then delete the temporary files again, zip the homework directory, and submit your *homework.zip* file via the assign server (instructions below).

**Checklist:**

1. When the grader unzips your *homework.zip* file in the *blepo* directory, it should create a *homework* directory with your *homework.sln* file and any necessary source or header files contained in *blepo/homework/*, but not any extraneous (temporary) files.
2. When your program reads the command line parameters, it should interpret them with respect to the *blepo/images/* directory. The grader should not have to worry about filename paths or anything complicated to get your program to work properly.
3. Your program should require 3 command-line parameters. If 3 parameters are not specified, the program should print an error.
4. If the program is unable to load *filename1* or *filename3*, or save *filename2*, or if *filename3* is not the proper size, then the program should print an error.
5. Your program should never crash or hang. This is very important, as it demonstrates robustness of your code to various inputs. Even if your program loads an image of size 0x0 or some other border condition, it should exit gracefully. You have no control over what files the grader will specify for *filename1* and *filename3*.
6. If *filename2* has the JPEG extension, then print an error. Saving the mask as a JPEG image would cause it to be lossily compressed, which would prevent it from working properly if it were read back in, i.e., if *filename2* and *filename3* were identical.
7. When run, your program should create 4 figure windows to display the original image (*filename1*), the synthetic image (*filename2*), the loaded mask image (*filename3*), and the image after applying the mask.
8. Your program should enter the figure while loop so that the images can be viewed as long as the user wants, until the user hits Ctrl-C or similar key.
9. Make sure your program works in Release mode. The grader will first attempt to compile and run your program in Release mode. Only if this attempt is unsuccessful will the grader resort to Debug mode (for partial credit).
10. Your program should be reasonably fast (a fraction of a second for all steps that do not require waiting for user input).
11. Your code should be professionally and cleanly written, with at least a minimal amount of documentation.

**Assign server instructions:**

To turn in your assignment, send an email to [assign@assign.ece.clemson.edu](mailto:assign@assign.ece.clemson.edu) .  Be sure to do the following:

* make the subject line "ECE847-1,#n" (without quotes but with the # sign), where 'n' is the assignment number.
* the body of the email is not important and may be left blank
* **cc the instructor and grader**, so we have a record of your submission in case something is wrong with the assign server. *We cannot grade what we do not receive.*
* send this email from your *@g.clemson.edu* account
* attach a zip file containing all the files needed to compile your project.  But do NOT check in all the other files that Visual Studio creates automatically.  **When in doubt, check out your code to a new temporary directory and verify that it compiles and runs.**  In other words,
  + Do NOT include these files:  .aps, .clw, .ncb, .opt, .plg, .suo, .sdf.  Also, **be sure to delete the Debug and Release** (and ipch, if it exists) **directories**.