COMP 193
Intro to GPU programming using CUDA
Exercise 2

In this exercise you will use some template code to learn about writing kernel functions. The template code opens an image file (SN.bmp) of Van Gough's Starry Night, and manipulates the pixels of that image on the GPU. Developing from the code template we used in the last exercise, there are some new files. The two main new files are animate.cu and animate.h – these are used for drawing the output of the program. Also included is a separate params.h file that will allow for continued growth of the code base for the next exercise and term project. You should read through Chapters 5-7 of the book.

1) To get started, download and extract Exercise 2 package: exercise2.zip
included in the .zip are five files:
interface.cpp
interface.h
gpu_main.cu
gpu_main.h
animate.cu
animate.h
params.h
Makefile

SN.bmp (image file from Van Gough's Starry Night)
exercise2.pdf (this file)

Put them in a directory and type 'make' at the command line. Assuming everything goes well, this should generate an executable called ex2. Run the executable by typing:

> ./ex2 SN.bmp

If you cannot get this to work, please notify me right away.

2) Read through the code to see what is going on. Specifically, read through interface.cpp and gpu_main.cu and pay attention to the new stuff. You shouldn't need to do anything with animate.cu, but you might look through it anyway. In essence, there are four separate grids of data loaded on to the GPU (gray, red, green, blue), each a 2D grid of values with values between 0 and 1.

3) The example in the code template fades the red and green pixels off, and blurs the blue pixels to take on their neighborhood values following the example code of “heat” that uses texture memory in Chapter 7. Your task is to be creative and do something with updating pixels to make an interesting visual effect. The only requirement is that you use texture memory in your algorithm (read Chapter 7) where a pixel's update depends on its neighboring pixels. For instance you could build edge detectors by subtracting the difference between a pixel and a pixel to its right (or left, or top, or bottom). Also feel free to incorporate a random number generator.

4) Your second task for the exercise is to revise the code so that it can run on any size .bmp image file. I have it hard-wired to read in an 800x800 file (the size of SN.bmp), but it should be more general.

(bonus points given to anyone who catches any glitches in my template code!)