CENG213 Data Structures
Fall 2008
Homework 2
Due: 03/12/2008 23:59

General Rules:

- You have 7 days total of late submission for all the homeworks. You can use them in any/all of the homeworks but be sure that your total late submissions are less than the limit.
- There is no teaming up. The homework has to be done individually. Be careful when discussing the homework. Sharing much information can also be accepted as cheating by looking at the similarities between submitted works.
- In case of cheating, all involved get zero.
- Your codes should be written in C++, and will be controlled using g++ on "inek" machines. Be sure that your homework runs on inek machines.
- Proper use of indentation is encouraged. You can use auto-indentations of the editors, homeworks without indentation will be penalized by 5%
- Be careful about input/output specifications (do not print any unnecessary characters, white spaces.)
- Submit your homework electronically through https://cow.ceng.metu.edu.tr

Homework: Morphing Images Using Predefined Operations

For this homework, you will write a program to process and store an image in an image class that you have implemented. First you will apply a sequence of predefined operations to find the resulting image. Second, you will be given two image files and you will try to find the sequence of operations to morph the first image to another one, by using backtracking method. For backtracking, you will use a stack for pushing and popping applied operations and morphed images at intermediate steps.

Image Format

- You will read/write PPM formatted image files of 512x512 pixel size
- PPM files are stored as a text file. They start with a header like the example given below. (# is used for comments)

```
P3
512 512 #width and height
256 #maximum color depth
<512x512x3 integers of pixel intensities>
```

The header is followed by 3 x Width x Height integers which represent Red Green Blue intensities of each pixel. (In this case there are 512*512 pixels and 512*512*3 integers.) Integers must be separated by a whitespace or newline character.

Here's an example 2x2 PPM file. Upper left corner is white (R:255,G:255,B:255) and lower right corner is green (R:0, G:255, B:0) :

```
P3
2 2
256
```

```
255 255 255 0 0 0
0 0 0 0 255 0
```

- Your input files will not contain any comment lines, so you don't have to worry about # character
- You can see how your PPM outputs look like by using any image editor. (i.e. eog at linux machines)

Operations

1. Swap up and down
   Swap upper and lower halves of the image.
2. Mirror
   Mirror the image with respect to a vertical mirror
3. Rotate R G B values of each pixel in upper and lower halves
   Rotate R<-G<-B in upper half (so $R_{new}=G_{old}$, $G_{new}=B_{old}$, $B_{new}=R_{old}$), and R->G->B in lower half (so R_{new}=Bold, $B_{new}=G_{old}$, $G_{new}=R_{old}$)
4. Invert colors in quarters
   Invert colors of pixels in top-left and bottom-right quarters of the image. Inversion is done by subtracting each color's (R G B) intensity from 255.

First Part - Applying an operation sequence to an image

In this part, you will apply a sequence of operations to the input image. Operations are not applied to the whole image, rather the image is divided into regions and the operations are applied to every region independently. How these regions are defined depends on the position of the operation within the sequence.

- The 1st operation will be applied to the entire image of 512x512 pixels.
- For the 2nd operation, you'll divide the image into 4 256x256 regions, and apply the operation to every region independently.
- For the 3rd operation, you'll divide each region in the 2nd operation into 4 128x128 regions; so you'll apply the operation to 16 128x128 regions independently.
- For the 4th operation, you'll divide each region in the 3rd operation into 4 64x64 regions; so you'll apply the operation to 64 64x64 regions independently.

You will continue to divide into regions and apply the operations until the sequence is over.

Second Part – Finding the sequence

In this part, you will read two images and find the sequence of operations required to morph first image to second one. The method of applying the operations is same as the first part (by dividing into regions). You will use backtracking method until the depth limit given as argument. You can use STL stack or write your own one to store intermediate images and operations during backtracking. You will output the sequence of operations separated by whitespaces.

Input/output Specs:

- Your program will be called with some arguments. First one will be “1” or “2” depending on which part of the homework is requested. Second and third arguments will be the names of the

image files to be used.
(i.e. "./hw2 1 initial.ppm morphed.ppm").
For second part you will have an extra argument representing maximum depth for the backtracking method.
(i.e. "./hw2 2 initial.ppm final.ppm 4")

- For the first part, you will get sequence of the operations from standard input (i.e. "1 4 1 2") and apply it to first image and save as the second image.
- For the second part, you will find the sequence of operations to morph from first image to second image and print the sequence to standard output. (i.e. "1 4 1 2").
- All the images will be 512x512 pixel ppm formatted images.

Remarks:

- Implement an image class to store the images. When backtracking, you will use a stack containing these image objects.
- Use backtracking method to solve the given problem. Any other algorithm will result to zero as grade.
- Do not print out any other unnecessary information.
- Be sure that your homework compile and run using g++ on inek machines, with example input outputs.

Example input/output images and corresponding sequences will be given. Follow the newsgroup for your questions or possible updates