Pseudocode

Kitty Zhuang, Bryant Williamson

CPE101-01

S. Einakian

Pixelimage.py pseudocode:

1. Check the length of the argument
2. If the length of the argument is 2 and the name of the file is 'puzzle.ppm', use the puzzle function.
3. If the length of the argument is 5, then use the fade function.
4. If the length of the argument is 3, then use the blur function.

Puzzle.py Pseudocode:

1. Check if the commend-line argument has the correct argument.
2. If the argument lacks the file name or has more than 1 file name, print a message to reflect the problem and exit the program.
3. Use try/except statement to open the file
4. If the file doesn't exist, use except to print a message to reflect that a wrong file name.
5. The try part should start with reading the argument file, organize the information contained in the image file to a list.
6. The first 4 item in the image file format represent the header, image width, image height, and a maximum value for a color component, interpret the information and assign them variable names to access later.
7. Group the rest of the value into groups of three, which will be the color component value for red, green and blue (always in this order). Make sure the values are in the type int for decoding.
8. Decode the image by increasing the value of the red component (the first component) by multiplying it by 10. Set both green and blue component equal to the new red value.
9. Check if the new red component exceeds the maximum value, if it does, set it to the maximum value.
10. Use the decoded pixel color component information to write a new image .ppm file following the image format. Makes sure the data written is in the str format.

Fade.py Pseudocode:

1. Check the commend-line argument has 4 output in this specific order. First argument specifies the name of the input image, the second specifies row(y-coordinate), the third specifies column(x-coordinate) positions, and the last specifies the fade radius.
2. Check if the last three arguments are integers.
3. If the arguments are not provided, print a message to such as (Usage: python3 fade.py <image> <row> <column><radius>) and terminate the program.
4. Use try and except. If the file name is wrong, print a message (unable to open <image> **<image> as the user entered argument).

5. The try part should start with reading the argument file, organize the information contained in the image file to a list.
6. The first 4 item in the image file format represent the header, image width, image height, and a maximum value for a color component, interpret the information and assign them variable names to access later.
7. Group the rest of the value into groups of three, which will be the color component value for red, green and blue (always in this order). Make sure the values are in the type int for calculations later.
8. Use the width and height information from the image file in combination with the row and col argument provided by the user to locate the pixel which the fade center point is at.
9. Calculate the distance between every pixel and the fade center point.
10. Scale each of the color components of the pixel by (radius – distance)/radius. If the scale value is less then 0.2, set the scale value equal to 0.2.
11. Use the decoded pixel color component information to write a new image .ppm file following the image format. Makes sure the data written is in the str format.

Blur.py Pseudocode:

1. Check the command-line arguments. The arguments should be in the following order. The first specifies the name of the input file and the second is an integer value specifying the “neighbor reach” to use the in-blurring calculation. Second argument is option, has a default value of 4.
2. Use try and except. If the file name is wrong, print a message (unable to open <image> **<image> as the user entered argument).
3. The try part should start with reading the argument file, organize the information contained in the image file to a list.
4. The first 4 item in the image file format represent the header, image width, image height, and a maximum value for a color component, interpret the information and assign them variable names to access later.
5. Group the rest of the value into groups of three as a pixel, which will be the color component value for red, green and blue (always in this order). Make sure the values are in the type int for calculations later.
6. For each pixel in the image, calculate for a blurred component as an average of its surrounding pixels. The number of pixels included depending on the “neighbor reach”. For the pixel at the boundary that does not have the full complement of neighbors, average the existing neighbors.
7. Use the decoded pixel color component information to write a new image .ppm file following the image format. Makes sure the data written is in the str format.