Programming Exam 09/02/2022

Exercise 1

In computer graphics, alpha compositing or alpha blending is the process of combining one image with a background to create the appearance of partial or full transparency. It is often useful to render picture elements (pixels) in separate passes or layers and then combine the resulting 2D images into a single, final image called the composite. Compositing is used extensively in film when combining computer-rendered image elements with live footage. Alpha blending is also used in 2D computer graphics to put rasterized foreground elements over a background.

In order to combine the picture elements of the images correctly, it is necessary to keep an associated *matte* for each element in addition to its color. This matte layer contains the coverage information —the shape of the geometry being drawn— making it possible to distinguish between parts of the image where something was drawn and parts that are empty.

In a 2D image a color combination is stored for each picture element (pixel), often a combination of red, green and blue (RGB). When alpha compositing is in use, each pixel has an additional numeric value stored in its alpha channel, with a value ranging from 0 to 1. A value of 0 means that the pixel is fully transparent and the color in the pixel beneath will show through. A value of 1 means that the pixel is fully opaque.

With the existence of an alpha channel, it is possible to express compositing image operations using a compositing algebra. For example, given two images A and B, the normal compositing operation is to combine the images so that A appears in the foreground and B appears in the background. This can be expressed as A over B.

As an example, the over operator can be accomplished by applying the following formula to each pixel:

$$\alpha_o = \alpha_a + \alpha_b(1 - \alpha_a)$$

$$C_o = \frac{C_a\alpha_a + C_b\alpha_b(1 - \alpha_a)}{\alpha_o}$$

Here C_o, C_a and C_b stand for the color components of the pixels in the result, image A and image B respectively, applied to each color channel (red/green/blue) individually, whereas α_o, α_a and α_b are the alpha values of the respective pixels.

Your task

Write a command line program with the following syntax:

```
compose <output_filename> [-p <x1> <y1>] <input_filename1> [ [-p <x2> <y2>] <input_filename2> ... ]
```

All filenames must be without extension and the program automatically appends a `.pam` extension.

The first parameter is the name of the PAM file that the program must generate by composing one after the other all the images specified in the following parameters. It's possible to add a `-p` option to specify where the image should be composed on the previous ones.

For example:

```
compose actors -p 6 4 cage -p 316 5 pitt
```

must generate an image by placing image `cage.pam` at coordinates (6,4) and image `pitt.pam` at coordinates (316,5). The output image size must be big just enough to fit all the other images. For example, if `cage.pam` has width=297 and height=170 and `pitt.pam` has width=183 and height=275, the final image should be 499x280.

The initial content of the image is all black pixels with alpha=0.

The PAM format is documented in the file `PAM format specification.html` and PAM images can be viewed with XnView. The output image will be a PAM file with DEPTH 4 and TUPLTYPE RGB_ALPHA. The input image(s) will be PAM file(s) with DEPTH 3 and TUPLTYPE RGB, or DEPTH 4 and TUPLTYPE RGB_ALPHA.

The composition will be performed with the previous equations by composing the first input image with the transparent backgroun; the result is then composed with the second image, and so on.

The available test images can be composed with:

```
compose blend background volt 01 02 03 04 05 06 07 08 09 10 11 12

compose actors -p 6 4 cage -p 316 5 pitt -p 510 8 freeman -p 60 193 downey -p 303 288 dicaprio -p 619 230 crowe
```