**ASSIGNMENT OVERVIEW:**

This assignment will build on the knowledge you now have of PPM files. You will also be able to use the triangle lab you completed earlier. In addition to a triangle you defined in lab, you are going to define a circle and quadrilateral. Using these shapes, you are going to create your own ppm image.

**LEARNING OBJECTIVES:**

This assignment will give you practice working with the following concepts:

* Multiple files
* Command-line arguments
* Designing and Creating images
* FILE I/O
* Reading data from a file
* C style strings
* Structures

**ACADEMIC INTEGRITY:**

You may work with one partner from your class section. (no cross sections on this assignment) You may not receive help from anyone other than myself or a lab TA. Please review the academic integrity policy provided in the syllabus.

You will have multiple options for this assignment:

**Things all options must have:**

Struct to represent a point (x and y values)

Struct to represent a ppm pixel (RGB values)

Struct to represent a ppm header (magic number, width, height, maxVal)

Struct to represent a circle (center point, radius)

Struct to represent a triangle (three points)

Struct to represent a quadrilateral (four points)

Struct to represent a shape (circle, triangle quadrilateral)

driver.c – This file is where main will live. This file should have minimal code.

ppm.h – This file should have function prototypes related to ppm images. It should also define any structs related to ppm. There should be no code in the header files.

ppm.c – This file has the implementation of the functions in ppm.h.

shape.c – This file should have all functions specific to the shapes.

shape.h – prototypes, structs, etc.

A dynamically allocated array of type struct shape.

You are also required to use header guards in each of the .h files. Points will be deducted if header guards are not used. When designing your functions, they must be small and do specific task. Large clunky functions will result in a reduction of points.

You must provide an input file. You may not deviate from the format given (below).

**Shape info:**

A circle has one point (x and y) that represents the center of the circle. It also has a value for the radius.

A triangle has three points, that represent the three vertices of a triangle.

A quadrilateral has four points that represents the vertices of the quadrilateral.

You will define your ppm header and the shapes in an input file using the following format:

P6 500 500 255 size of image

3

Circle

325 200 65 325 200 are center points, 65 is the radius

255 0 0 color

Triangle

125 130 point one

175 253 point two

75 253 point three

0 255 0 color

Quadrilateral

175 310 point one

125 398 point two

225 462 point three

300 400 point four

0 0 255 color

Background

255 255 0 color

Is pixel inside any of the shapes? No? background color. Yes? Color of shape.

It is crucial the format of the input file matches the above example. I should be able to provide my own input file and your program read it and display in the image the shapes I define.

The first line contains the header information for the ppm file. The next number represents how many shapes your image will have. For each shape you should have a string to indicate the type of shape being defined, then the information needed to define the shape including the RGB values that will define the color of the shape. The string MUST spell out the shape type with the first letter being capitalized. The last item in the file should be the color of the Background.

Option 1.

For a maximum grade of 90/100, you will create an image that has one circle, one triangle, and one quadrilateral. It does not matter where these shapes are placed in the image, just that they are in your ppm output image. This is the simplest of the three options.

Read in the shape information from the input file. Store the shapes in an array of struct shape.

You will loop through each pixel in the image calling the functions isHitCir, isHitTri and isHitQud to determine if the pixel being checked is within the defined shape. If the isHit\_\_ function returns true then print the pixel using a defined color for that shape. If isHit\_\_ function returns false for all shapes then print the pixel the predefined background color.

**Option 2:**

Things needed for Option 2.

All structs and files required for Option 1 above.

For a maximum grade of 100/100, use at least one triangle, one circle, and one quadrilateral to create a picture in your ppm image. This picture can be as simple as a snowman with a top-hat and carrot for a nose. You MUST create an input file using the same format as described above. This will allow me to use my input file when testing.

You will read the data from the input file storing the shapes in the array of shapes. Once all shapes are stored in the array, loop through each pixel of the image checking to determine if the current pixel is within any of the shapes stored in the array. Based on the type of shape stored in the current element of the shape array, you will call a function called isHitCir, isHitTri or isHitQua. If the isHit function returns true, then print the pixel using a defined color for that shape. If isHit returns false, then print the pixel the color of the background. As you can imagine there may be shapes that overlap therefore the order these shapes are located in your input file could matter.

**Option 3:**

Things needed for Option 3.

All structs and files required for Option 1 above.

A struct to represent all shapes. This struct should have members of type circle, triangle and quadrilateral.

Rather than using an array to store the shapes use a linked list

For a maximum grade of 110/100 using the three shapes create a picture as described in Option 2. Store each defined shape in a linked list rather than an array of shapes.

Challenge Option:

If you choose to do #2 or #3 above, you may enter your image in a contest for best overall image. All images entered in the challenge will be compiled into one document to be judge by several faculty members of the SoC. Your image will be judged based on creativity, level of difficulty, and appeal. If your image is chosen as best image, you could receive a maximum grade of 115/100 and your entire class will receive donuts on a day of my choosing.

**Triangle:**

You should already know how to define a triangle since you did this in lab.

**Quadrilateral:**

With respect to a quadrilateral. You can determine if a point is within the quadrilateral by using a combination of two out of 4 triangles that make up the quadrilateral:

Example:

ABC and BCD

or

ABD and ACD

If the point at X Y position is within either of these two combinations of triangles then it is within the Quadrilateral.

A

B

D

C

Once you have written the isHit function for the triangle, you should have this from lab, the quadrilateral is easy. You will call the isHit function for the triangle twice using one of the two triangle combinations listed above.

**Circle:**

A circle is defined by a point that represents the center of the circle and an integer that represents the radius of the circle. With this information and the point that represents the pixel on the plane you are checking, it is a relatively simple formula. If the square root of ((x – center x)2  + (y – center y)2) is less than the radius, then the point is within the circle.

Height is y width is x

Below is the output for an 600 X 400 (W X H) image with a circle defined as follows:

Radius = 75, centerY = 200, centerX = 300. Please pay close attention where x and y are located and how we reference pixels using width and height.

A picture containing icon

Description automatically generated

You are expected to provide a makefile that has a make run. I should be able to type make run and your program compile and run. The file that has the shapes defined MUST be name input.txt. After running your program the initial time to see the image you produced, I will run a script that will use my input.txt file to test your program. Therefore it is important that you have the same format for the input files.

**FORMATTING:**

You will need to add a header to each of your files like the following:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*Your name

\*CPSC 1020 your Section, F18

\*Your email

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Your program should compile with no warnings and no errors. If your program does not compile the highest grade you can get for the assignment will be 20. If your program compiles but has warnings, there could be a deduction up to 20 points.

* Your code should be well documented. (comments) **Your comments should be in the header files (.h)**
* There should be no lines of code longer than 80 characters.
* You should use proper and consistent indention.

Here are some guide lines for documenting the code in your assignment.

Before each function you are required to have a detailed description of what the overall function does. You should explain what each parameter is and what it is used for.

/\* Parameters: img - image\_t pointer array holding the image data for  
 \*                   each of the input files  
 \* Return:     output - image\_t struct containing output image data  
 \* This function averages every pixels rbg values from each of the   
 \* input images and puts those averages into a single output image  
 \*/

Also, if you include comments in the body of the function (and you should) they should be placed above the line of code not beside the code.

Example:

Bad

if(something) //This is a comment

{

do something;

}

Good

//This is a comment

if(something)

{

do something;

}

**HANDIN:**

Use handin.cs.clemson.edu to submit your files. I will create five buckets named PA3\_85, PA2\_90, PA2\_100, PA2\_110, PA3\_Challenge. See explanation above:

Things to do prior to handing in your files.

1. **Test your program on the SoC servers**. Because there have been some issues with the virtual.computing.clemson.edu virtual machine, you are required to test your program on **cerf15 using ssh to access the cerf15 not through virtual.computing.clemson.edu**. In other words, you are to pull up a terminal, use putty, go to the lab, etc. ssh to cerf15 to test your program. This is the machine I will use to test your program. I will not accept the excuse “It compiled on my computer.” I test and grade programming assignments on cerft15 through an ssh client.
2. Tar zip your files naming the tarred file to match the bucket you turned your assignment in, ex. PA2\_110.tar.gz. When I untar your file I should immediately be able to type make or make run and the program run.
3. You should provide a **README** that consist of the following. The readme should be named **README**, **readme.txt, or README.TXT**. **Do not use .docx, .doc, .pdf, etc.**

* A short description of any problems you encountered when writing this program.
* How you solved the problems you encountered.
* Your thoughts on the assignment. This is your opportunity to tell me if you like the assignment or not. What you did or did not like about the assignment. Anything you want to tell me.

1. Make sure you check and double check the files you handin. If you are missing any file, image, etc., you will get an automatic 0 no exceptions. It is your responsibility to make sure your files are correct and have not been corrupt during the handin process. You can check this by reviewing what you actually turned in through the handin page. This will take time so don’t wait until the last minute to hand in this project.
2. **THERE WILL NOT BE AN EXTENSION ON THIS ASSIGNMENT. START EARLY.**