**ASSIGNMENT OVERVIEW:**

You should be “experts” with PPM images by now!?!

For this assignment you will build on the previous two assignments. By this point, you should have basic code to read, write, and manipulate images. From your labs you should also be able to define two shapes, a circle and triangle, and create an image with those definitions. You will add a quadrilateral to the set of shape classes you have already written. Using these classes, you will create an image of your own.

In class we are learning about inheritance, an important concept to object oriented programming. You will be **required** to incorporate inheritance in this assignment.

You will have multiple options for this assignment.

1. For a maximum grade of 85, you will create an image that has one circle, one triangle, and one quadrilateral.
2. For a maximum grade of 90, complete #1 above and use polymorphism in your program.
3. For a maximum grade of 100, you are required to use a little more creativity. Using the three shapes create a picture. The picture could be as simple as a house with the sun in the sky or perhaps a snowman, with a top-hat and carrot for a nose.
4. For a grade of 105 points, complete #2 above and use polymorphism in your program.

Challenge Option:

If you choose to do #3 or #4 above you may enter 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 will receive a maximum grade of 115. **The bad news**: **If you want your image to be considered in the challenge, your assignment will need to be submitted on December 5 at midnight. This will allow time to compile and produce your image for judging.**

**LEARNING OBJECTIVES:**

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

* Multiple files
* Command-line arguments
* Reading and writing images
* FILE I/O
* C++ vectors more specifically multi-dimensional vectors (2D)
* Initialization list
* Inheritance
* Virtual functions
* Abstract classes

**ACADEMIC INTEGRITY:**

This is an individual assignment. You may not receive help from anyone other than myself, a lab TA or ASC tutor. Please review the academic integrity policy provided in the syllabus.

**REQUIREMENTS:**

Using C++ you are going to add to the PPM suite of classes and functions you have already written. As stated above, you are also going to draw from several of the functions completed in recent labs.

In addition to the Image class that “has-a” relationship with Header and Pixel, you are going to add inheritance. Inheritance provides the “is-a” relationship between classes.

You are going to create a Shape class which will be considered an abstract class. We will cover abstract classes in a few days. (I will briefly explain abstract classes later in this document.) You will have a class called Circle, Triangle, and Quadrilateral. Each of these classes will inherit from Shape. The Shape class will provide attributes, such as, a string for the type of shape, a Color since every shape you create will have a color. Since each shape is defined by at least one point, your shape class should also have a vector or pointer of type Points. Therefore, you will have a class called Point that will represent the X and Y values on a 2D plane. As an example, Circle has an X and Y value that represents the center point of the circle. Also, there are three points that define a triangle. Hence, your shape class will provide a vector **or** array of Points. It does not matter that Circle is only represented using one Point you can have a vector of size 1 or pointer that points to 1 or an array of 10 Points.

Base/Parent class

Derived/Child class

Shape will also provide attributes in the form of functions that each of the derived classes, a.k.a., child classes have access to.

So let’s stop and review. Your program should consist of the following:

**Image.cpp/.h**

**Header.cpp/.h**

**Pixel.cpp/.h**

**Shape.cpp/.h (This is an abstract class.)**

**Circle.cpp/.h**

**Triangle.cpp/.h**

**Point.cpp/.h**

**Color.cpp/.h**

**driver.cpp**

I know this seems like a lot, and it is, but you should have most of Image, Header, Pixel, Circle, and Triangle written.

Below is an example of the Shape class I used with my code.

**class** Shape

{

//protected means derived classes have direct access to the data members

**protected**:

Color c;

string name;

vector <Point> pts;

**public**:

Shape();

**virtual** **void** readInfo(ifstream&) = 0;

**virtual** **void** setPoint(vector <Point>) = 0;

**virtual** **bool** isHit(Point&) = 0;

**virtual** **void** draw(Image&) = 0;

vector <Point> getPoints()**const**;

**void** setName(string);

string getName() **const**;

};

A virtual function with = 0 at the end means this is a pure virtual function. Pure virtual functions make the class an abstract class.

“An **abstract class** is a **class** that is designed to be specifically used as a base **class**. An **abstract class** contains at least one pure virtual function. You **declare** a pure virtual function by using a pure specifier ( = 0 ) in the declaration of a virtual member function in the **class** declaration.” https://www.ibm.com/support/knowledgecenter/en/SSLTBW\_2.3.0/com.ibm.zos.v2r3.cbclx01/cplr142.htm

Another important fact about an abstract class is you can only create a pointer to an abstract class. As an example, since Shape is abstract, you **canno**t create a variable of type Shape in the following way: **Shape sh**; However, you can create a pointer of type Shape. **(Shape \*sh;)** A pointer of an abstract class is the basis of polymorphism. We will cover polymorphism soon.

If a function is a pure virtual functions it **must** be implemented by each derived class. They are **not** implemented in the Base class. So from the above class declaration, Shape will not implement isHit, setPoint, readInfo, nor draw. Each derived class of shape **must** implement these functions.

Below I will discuss what each of these function do. Before, I discuss these functions, let me say, you are **NOT** required to use my version of the shape class. But you are required to have an abstract base class called Shape.

**virtual** **void** readInfo(ifstream&):

This function reads, from a file, the information needed for each of your shapes.

Below is an example information file:

Circle

150 150

100

0 0 255

Triangle

280 230

450 190

390 310

255 0 0

Quadrilateral

400 140

450 60

550 60

500 140

0 255 0

Each shape in the file is preceded by a string that represents the type of shape you will read. Now that you know the type of shape, you will know the type of information that needs to be read from the file. As an example, the sample file above has a circle as the first shape. The first line of data is the center point of the circle, next is the circle’s radius, last is the color of the circle.

**You are required to create a file similar to the above. Your file should define each shape that will be in your image.**

Hint:

In my implementation: In main I read in the string which will determine the type of shape that needs to be created. I then use an instance of that shape to call readInfo, which will read the information specific to that shape. Now I have the information to call setPoint (provided by Shape) as well as Colors’ setColor function.

**virtual void setPoint(vector <Point>) :**

After the points for the shape are read in, use them to set the vector of points inherited from shape.

**virtual bool isHit(Point&):**

This is the function used to determine if a specific point is within the parameters of the shape. You should have already written this function for Circle and Triangle. With respect to a quadrilateral. You can determine if a point is within quadrilateral by defining three triangles:

Define 3 Triangles.

Example:

ABD

ACD

BCD

If the point at X Y position is within any of these three triangles then it is within the Quadrilateral.

A

B

D

C

**virtual** **void** draw(Image&):

This function is used to loop through each pixel of your image calling the isHit function. If isHit is true then you set the images pixel to the color defined for this the shape.

Along with the above listed classes you are required to provide a driver and makefile. The driver should not have any functions defined in it. It should have the least amount of code as possible.

You are expected to provide a makefile that has a make run so I do not have to guess how to run your program. I should be able to type make run and your program compile and run.

**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)
* 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 should have a detailed description of what the overall function does. You should explain what each parameter is and what it is used for. To borrow from another student’s code, here is an example of overall function description.

/\* 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  
 \*/

**You are not required to do yours exactly like this one. But it should have the same info.**

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, PA3\_90, PA3\_100, PA3\_105, PA3\_Challenge. See explanation above:

Things to do prior to handing in your files.

1. **Test your program on the SoC servers**. I will not accept the excuse “It compiled on my computer.” I test and grade programming assignments on the SoC servers.
2. Tar zip your files naming the tarred file to match the bucket you turned your assignment in, ex. PA3\_105.tar.gz. When I untar your file I should immediately be able to type make or make run and the program run. All images used in your project must be included in your tarred file.
3. You should provide a **README** that consist of the following. The readme should be named **README** not readme.txt, README.TXT, only **README**

* 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.