Requirements Definition

MARC BIGLER

Shapes

1. Introduction and Context

The purpose of this program is to implement a library to help draw different shapes. This program will have the ability to compute a point, line, circle, rectangle, and triangle. These shapes can be drawn independently or they can be drawn as a composite of shapes. Each of these shapes will have the ability to move, scale, get area, read out to file, read in from file and get the points associated with each shape. All of these shapes can preform these actions individually or in a recursive manner on a composite of shapes. You can add and remove items from shape composites.

Composite pattern: This section of the assignment was implemented in the Image and ShapeGroup Class. This is the backbone behind the majority of the useful code.

Factory Method: This design pattern was implemented in the fileIn method. As each line is read in from the text file it creates an object at runtime.

Flyweight pattern: This design pattern has the concept of not creating new objects if the object already exists. The extrinsic state is what is being changed by the user and the intrinsic state is what remains unchanged. With the idea in designing a shape class of a rectangle the color of a rectangle could be an intrinsic state and the dimensions could be the extrinsic state. If a rectangle already exists with a specific color, that rectangle would be used. If a rectangle does not exist with that color a new one would be created.

Unit Test: I was able to get the majority of the testing completed but in the code review professor Clyde brought up the edge case testing which I need complete further but do not have the time unfortunately.

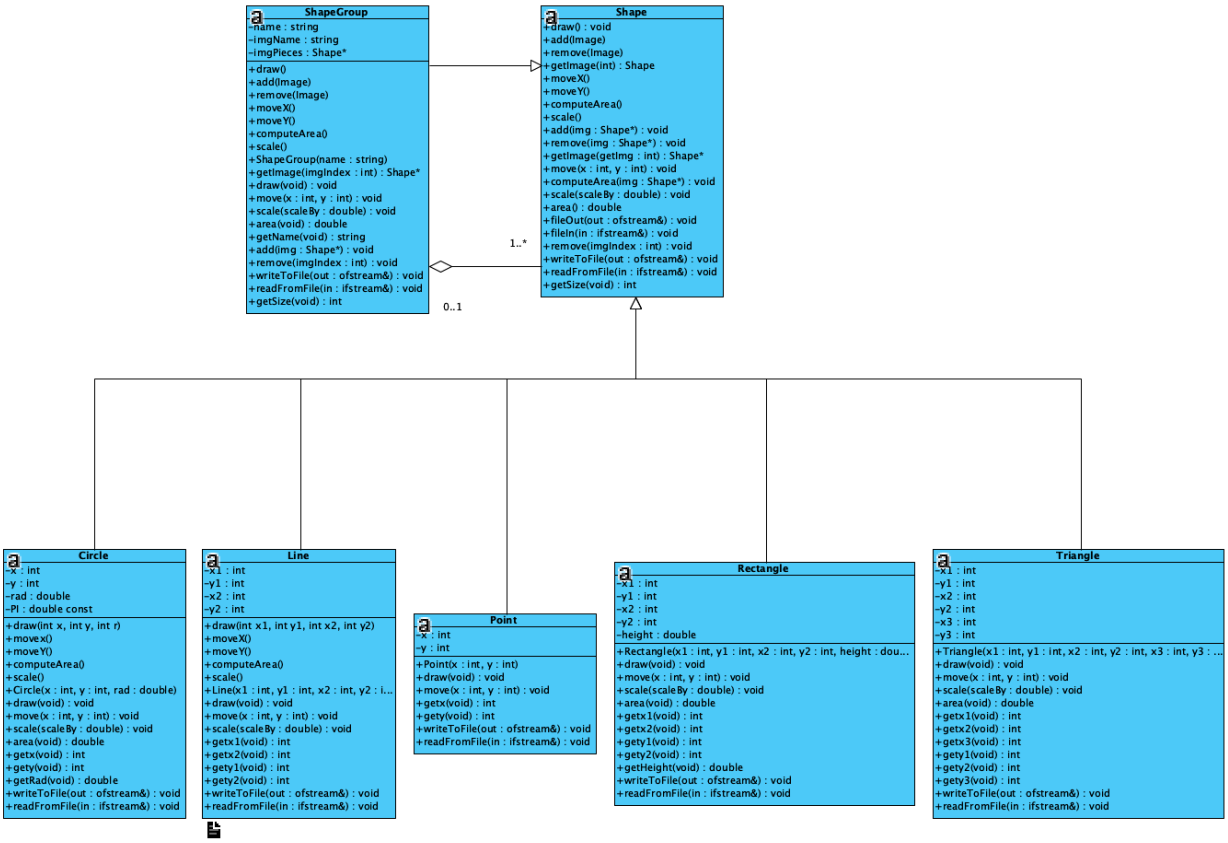
2. Insight into construction the code.

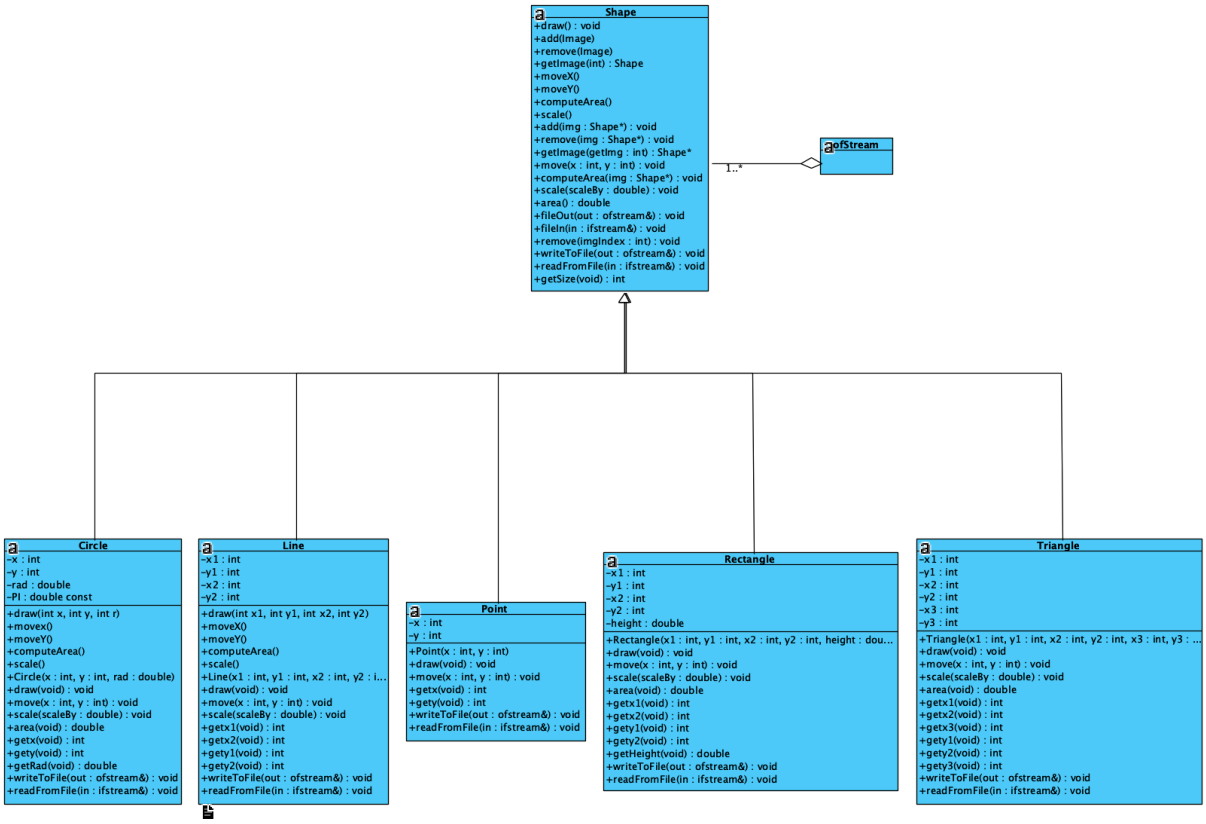
When starting this project it took a second to understand best how to implement the composite pattern. The idea seamed to be simple enough but making a recursive call on a class structure sounded more complicated than it actually was. Following the UML in the the heads first design patterns made it a lot easier to understand. Once I had the basic ability to do a composite of points, it was a simple rinse and repeat for all of the other shape objects.

The hardest part about this assignment was outputting the image to be visibly seen. In C++ on mac there is not an easy way of getting this task complete. If I was using a PC there is a graphic-s.h header that makes things really easy to output 2D image, but mac does not have this library. I started trying to accomplish this part of the assignment in openGL but did not get as much done as I was hoping I was able to get a basic GUI up and working when the draw function is called on the line class, but I will not be able complete all of this code. The bright side is that I got the Cmake file configured with all the other formatting to launch the GUI and display a line. Who would have thought seeing a line drawn to the page would be so satisfying!

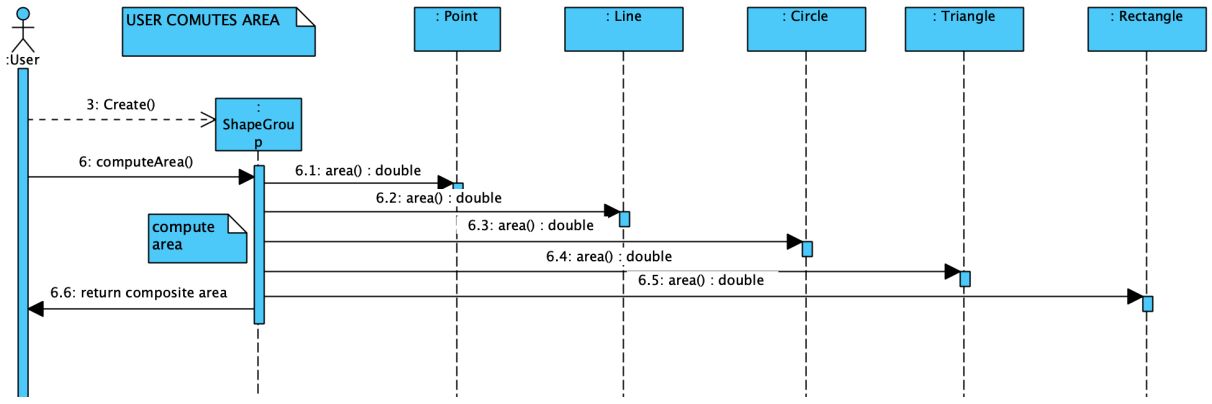
3. Diagrams

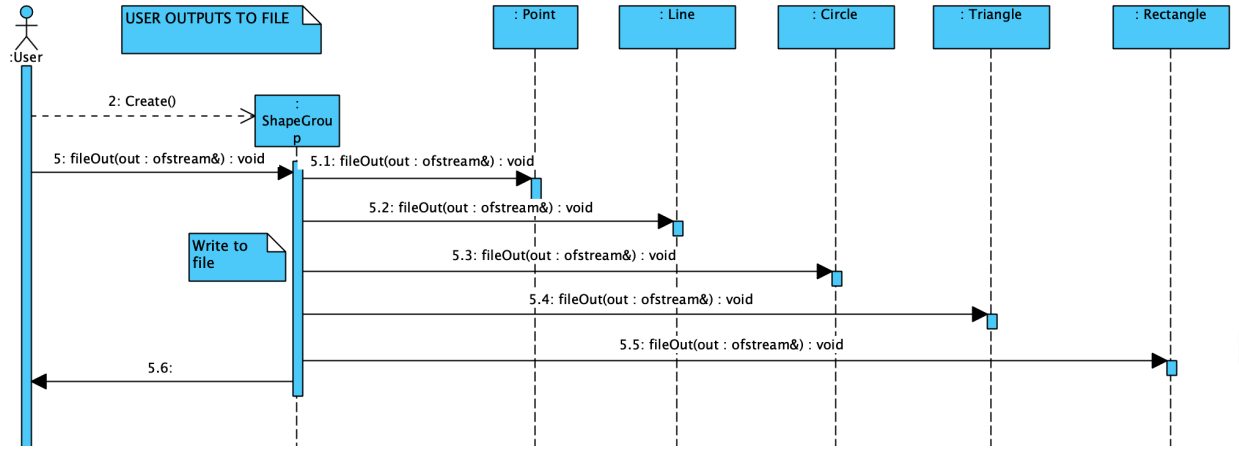
UML

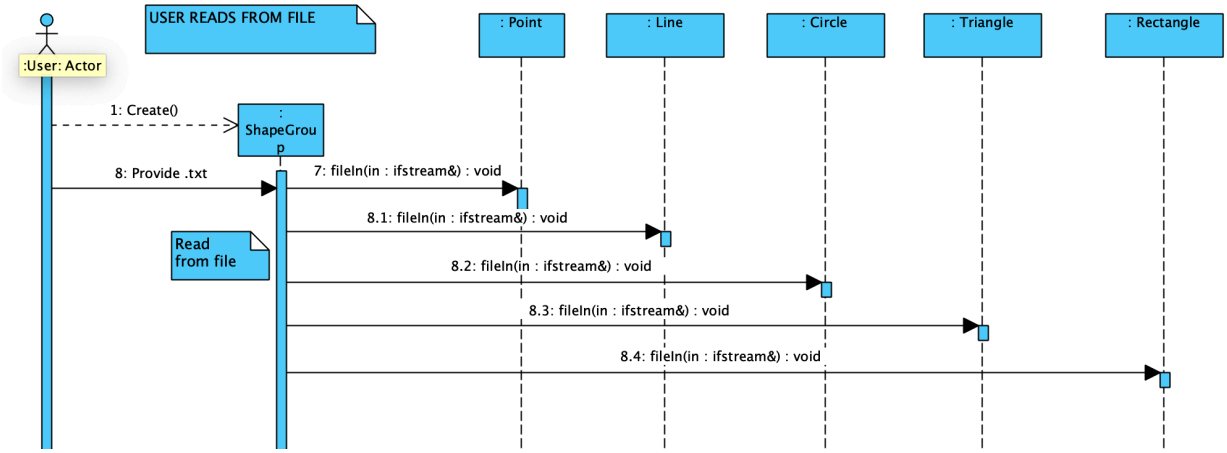




Interaction




USER OUTPUTS TO FILE
:User
: Point
: Line
: Circle
: Triangle
: Rectangle
2: Create()
: ShapeGroup
5: fileOut(out : ofstream&) : void
5.1: fileOut(out : ofstream&) : void
5.2: fileOut(out : ofstream&) : void
Write to file
5.3: fileOut(out : ofstream&) : void
5.4: fileOut(out : ofstream&) : void
5.5: fileOut(out : ofstream&) : void
5.6:


USER READS FROM FILE
:User: Actor
: Point
: Line
: Circle
: Triangle
: Rectangle
1: Create()
: ShapeGroup
8: Provide .txt
7: fileIn(in : ifstream&) : void
8.1: fileIn(in : ifstream&) : void
Read from file
8.2: fileIn(in : ifstream&) : void
8.3: fileIn(in : ifstream&) : void
8.4: fileIn(in : ifstream&) : void