CSCI 200 - Fall 2023
Foundational Programming Concepts & Design

A5 - SFML: Polygon Land

Home | HW Sets | Schedule | Files | Help | Links

→This assignment is due by Tuesday, November 14, 2023 11:59 PM.←

→ As with all assignments, this must be an individual effort and cannot be pair programmed. Any debugging assistance must follow the course collaboration policy and be cited in the comment header block for the assignment.←

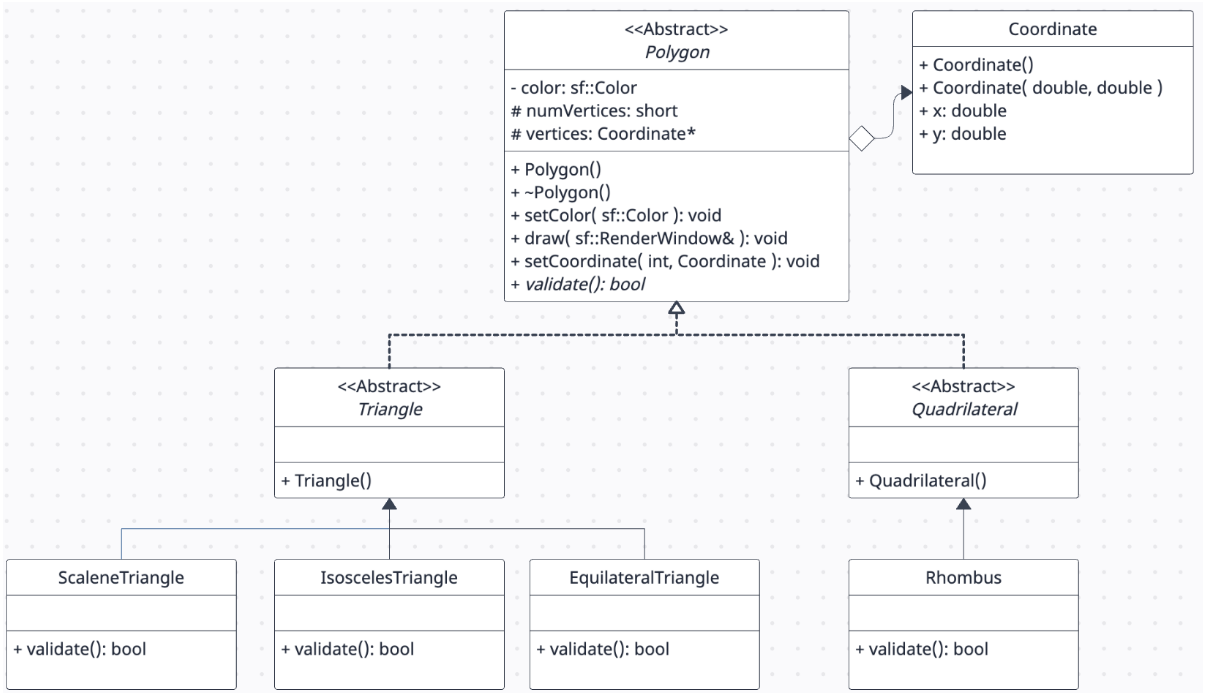
→ Do not forget to complete the following labs with this set: L5A ←

→ Do not forget to complete zyBooks Assignment 5 for this set.←

Jump To: **Rubric Submission**

For this assignment, we will write a program to read in a list of polygon data and draw an image comprised of those simple polygons. The full UML diagram is shown below (click to enlarge).

RMP-H



The Abstract Polygon

Begin by creating an abstract `Polygon` class. The class will have the following members and corresponding implementations:

- `public`
  - default constructor - sets the color to white, specifies the number of vertices as 0, sets the vertices array to be a nullptr
  - a **virtual** destructor - deallocates the vertices array
  - `void setColor(const sf::Color color)` - sets the private color data member
  - `void draw(sf::RenderTarget& window)` - creates a `ConvexShape`, sets the points for each coordinate, sets the fill color, draws it to the provided window
  - a `void setCoordinate(int idx, Coordinate coord)` function that sets the corresponding coordinate in the vertices array
  - a **pure virtual** `bool validate()` function that returns `true` if the set coordinates form the intended polygon
- `protected`
  - the number of vertices making up the polygon as a short integer
  - a Coordinate array
- `private`
  - the color of the triangle as an SFML Color object

The Abstract Triangle

Next create the abstract `Triangle` class that extends `Polygon`. The class will have the following members and corresponding implementations:

- `public`
  - default constructor - sets the number of vertices to be three and allocates the vertex array to hold three coordinates

---

The Concrete ScaleneTriangle

Now we'll begin creating concrete instances of a triangle. Create a class called `ScaleneTriangle` that implements `Triangle`. It will need to override the `validate()` method. Given the three vertex coordinates, compute the length of each side. If the three sides form a triangle (that is - all three sides are non-zero and the sum of any two is greater than the third) AND all three side lengths are different, then return true. If the sides do not form a triangle or have equivalent lengths, return false.

The Concrete IsoscelesTriangle

Create a class called `IsoscelesTriangle` that implements `Triangle`. It will need to override the `validate()` method. Given the three vertex coordinates, compute the length of each side. If the three sides form a triangle (that is - all three sides are non-zero and the sum of any two is greater than the third) AND at least two of the three sides are equal in length, then return true. If the sides do not form an isosceles triangle, return false.

The Concrete EquilateralTriangle

Create a class called `EquilateralTriangle` that implements `Triangle`. It will need to override the `validate()` method. Given the three vertex coordinates, compute the length of each side. If the three sides form a triangle (that is - all three sides are non-zero and the sum of any two is greater than the third) AND all three of the three sides are equal in length, then return true. If the sides do not form an equilateral triangle, return false.

The Abstract Quadrilateral

Next create the abstract `Quadrilateral` class that extends `Polygon`. The class will have the following members and corresponding implementations:

- `public`
    - default constructor - sets the number of vertices to be four and allocates the vertex array to hold four coordinates

The Concrete Rhombus

Create a class called `Rhombus` that implements `Quadrilateral`. It will need to override the `validate()` method. Given the four vertex coordinates, compute the length of each side. If the four sides form two isosceles triangles (comprised of vertices 0, 1, 2 & 0, 2, 3) AND all four of the four sides are equal in length, then return true. If the sides do not form a rhombus, return false. (Note: for this assignment, we can assume the points will form a simple polygon and the edges will not cross themselves.)

Read the File

A sample **polygon data file** is provided. The file contains some number of lines in the following format:

```
T x1 y1 x2 y2 x3 y3 x4 y4 r g b
```

Where each value is:

- `T` - a character denoting the polygon type. One of `S`, `I`, `E`, or `R`
- `x1` - the double x coordinate for vertex 1
- `y1` - the double t coordinate for vertex 1
- `x2` - the double x coordinate for vertex 2
- `y2` - the double t coordinate for vertex 2
- `x3` - the double x coordinate for vertex 3
- `y3` - the double t coordinate for vertex 3
- `x4` - the double x coordinate for vertex 4 (will only be present if type is `R`)
- `y4` - the double t coordinate for vertex 4 (will only be present if type is `R`)
- `r` - the integer red component of the color
- `g` - the integer green component of the color
- `b` - the integer blue component of the color

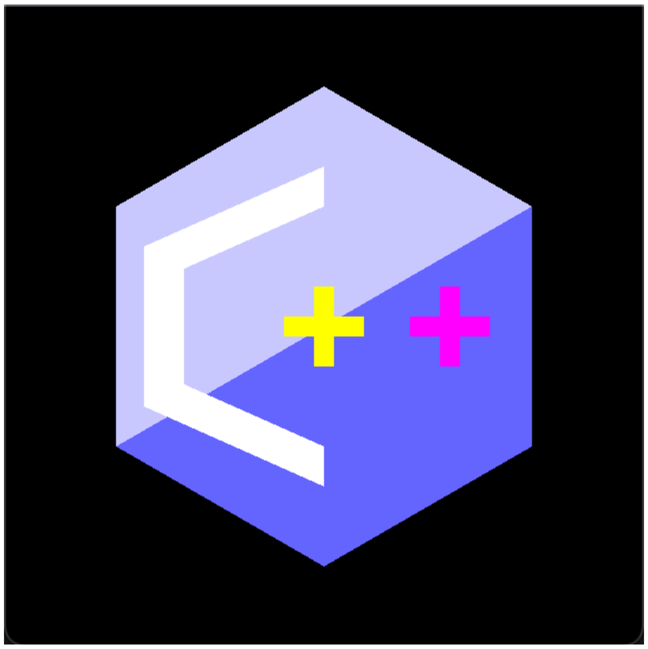
We want to be sure to leverage runtime polymorphism as we read in the file. Therefore, create a list of `Polygon` pointers. As you read each line, create a `Polygon` pointer of the corresponding type. Call the `setCoordinate()` method with the corresponding coordinate. If the vertices form a polygon of the appropriate type, then set the color and add the pointer to the list. If the vertices do not form the corresponding polygon, then print out "`polygon is invalid`" and the line that corresponds to the polygon.

The sample file results in the following output denoting which polygons are invalid:

```
polygon is invalid - "E 0 0 5 5 10 10 255 255 255"
polygon is invalid - "I 0 0 5 5 10 10 255 255 255"
polygon is invalid - "E 0 0 5 5 11 11 255 255 255"
polygon is invalid - "I 0 0 5 5 11 11 255 255 255"
polygon is invalid - "S 0 0 0 0 0 0 255 255 255"
polygon is invalid - "I 0 0 0 0 0 0 255 255 255"
polygon is invalid - "E 0 0 0 0 0 0 255 255 255"
polygon is invalid - "S 10 0 10 0 0 0 255 255 255"
polygon is invalid - "I 10 0 10 0 0 0 255 255 255"
polygon is invalid - "E 10 0 10 0 0 0 255 255 255"
polygon is invalid - "S 0 0 10 0 10 0 255 255 255"
polygon is invalid - "I 0 0 10 0 10 0 255 255 255"
polygon is invalid - "E 0 0 10 0 10 0 255 255 255"
polygon is invalid - "S 10 0 0 0 10 0 255 255 255"
polygon is invalid - "I 10 0 0 0 10 0 255 255 255"
polygon is invalid - "E 10 0 0 0 10 0 255 255 255"
polygon is invalid - "R 112.154 200 112.154 440 527.846 440 527.846 200 255 255 255"
polygon is invalid - "R 0 0 10 10 20 20 30 30 255 255 255"
polygon is invalid - "R 0 10 20 20 20 10 0 10 255 255 255"
```

Draw the Image

Now that we have a list of valid polygons, when inside our SFML draw loop we'll draw all the polygons. Loop through your polygon list, call the `draw()` method and pass it the window object. The sample file should generate the following image:



Grading Rubric

Your submission will be graded according to the following rubric:

Points	Requirement Description
0.5	Submitted correctly by Tuesday, November 14, 2023 11:59 PM
0.5	Project builds without errors nor warnings.
2.0	**Best Practices** and **Style Guide** followed.
0.5	Program follows specified user I/O flow.
0.5	Public and private tests successfully passed.
2.0	Fully meets specifications.
6.00	**Total Points**

Submission

Always, **always**, **ALWAYS** update the header comments at the top of your main.cpp file. And if you ever get stuck, remember that there is LOTS of **help** available.

Zip together your `main.cpp, Makefile, *.h, *.cpp` files and name the zip file `A5.zip`. Upload this zip file to Canvas under A5.

→This assignment is due by Tuesday, November 14, 2023 11:59 PM.←

→ As with all assignments, this must be an individual effort and cannot be pair programmed. Any debugging assistance must follow the course collaboration policy and be cited in the comment header block for the assignment.←

→ Do not forget to complete the following labs with this set: L5A ←

→ Do not forget to complete zyBooks Assignment 5 for this set.←

[Jump to Top] [Site Map]