Graphics project

Intro

For this class project we will design and implement a full-featured graphics library in C++. We will model its design after the library presented in Chapters 12-14 of Programming: Principles and Practices in C++. The major improvement to the library is that we will replace how the graphical elements are drawn: instead of relying on fltk to draw on the screen, we will write code to generate vector graphics images (SVG).

Objectives

This project represents a large step forward in your software development skills. You will:

- Work together with your classmates to design and implement an easy-to-use graphics library
- Apply object-oriented design techniques to make a library that is easy to add new features to
- Explore and evaluate the merits of different design decisions, focusing on the needs of the user
- Document the code that you write and provide a simple user guide

Organization

We will divide into two teams:

- **Frontend team**: designs and implements code that allows the user to create, modify, and manage shapes
- **Backend team**: designs and implements code that draws shapes using the SVG language

Teams will be able to work independently after they agree upon an interface between the frontend and backend. The frontend team should decide **what shapes they want to draw**, and the backend team should decide **which drawing primitives are made available**.

Although we will work in groups, you'll be graded individually. **Make sure to put your names on code that you write!** The standard homework rubric will be applied. **This project is worth 10% of your final grade.**

Frontend Team

Your job is to develop user facing code. Focusing on readable, intuitive, and straightforward ways of describing and manipulating shapes.

Major tasks:

- Write down simple english descriptions of what a user would like to do with the graphics library
  - "I want to make an image containing a red circle inside a blue rectangle"
  - "I want to produce a repeated geometric pattern, where lines spiral into the center of the image"
- Write down **simple code** that does these tasks
  - Focus on making it really easy for beginner programmers and hide the complexity inside the classes

- Organize your code and thinking using UML diagrams
- Create an abstract base class for shapes that holds data and functions that are common to all shapes
- Implement concrete shapes (Rectangle, Circle, Ellipse, etc)
    - Each shape should be able to draw itself onto a canvas using the available canvas' draw functions (**provided by the Backend Team**)
    - Until you receive an abstract base class for Canvas from the Backend team, "mock" the drawing functionality using something simple like print statements.
- Since users will interact directly with your code, please provide helpful error messages using C++ exceptions when the user enters incorrect values.
- Document your code with comments describing your intent
- Write a short, but clear user guide/tutorial

Extras (if you have time):

- Add a way to group shapes
- Add a way to manipulate shapes and groups of shapes: moveto(x, y), shift(dx, dy), rotate(angle)

Backend Team

Your job is to design a generic drawing interface. This code will provide primitive drawing functions that generate SVG drawing instructions. Make it easy to add new backends, such as a canvas for saving png files. Your user is the frontend team, so please focus on providing an easy-to-use interface for drawing basic primitive shapes.

Major tasks:

- Write down simple, english descriptions of what the frontend team (your user) wants to do
    - "Draw a green line between two points"
    - "Draw a curve/ellipse/circle at this position"
- Write down simple code involving a Canvas class that does these tasks
    - You must strike a balance between providing a huge number of drawing functions (one for every possible shape) and limiting these primitives to only what is necessary to draw all common shapes (perhaps only providing lines and curves).
- Implement an abstract base class for Canvas
- Implement a concrete class for SVG images
- Implement a library of functions for dealing with SVG instructions in an easy manner. E.g. the fact that SVG requires values to be surrounded by quotes, should be hidden behind a function.
    - References: https://developer.mozilla.org/en-US/docs/Web/SVG https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial
- Document your code and provide a separate document describing the Canvas interface

Extras (if you have time):

- Implement another backend that write an image file in a different format (see PPM or PNG)
- Implement a Graphical window using a library like SDL

Author: Dr. Brown

Validate