HO CHI MINH CITY UNIVERSITY OF SCIENCE

Project Report (Milestone 1)

SVG RENDER

Instructor: Đỗ Nguyên Kha

Member list: (22127006) Nguyễn Duy Ân
(22127091) Phạm Mai Duyên
(22127104) Phạm Thị Mỹ Hạnh
(22127259) Nguyễn Đức Mạnh

Ho Chi Minh, 11/2023

Table of content

1. Overall

In our project, we explore the utilization of SVG files, a scalable vector graphics format that allows rendering images based on the information contained within the file. Unlike raster images such as PNG files, SVG files enable the creation of images using scalable vector graphics. Under the guidance of our instructor, we divided the milestone into two main processes: file reading and image rendering.

For the file reading process, we employed ***RapidXML*** to extract information from the SVG file. This step involves parsing the SVG file content to retrieve essential data for further processing. Our goal is to ensure accurate and comprehensive extraction of information from the SVG file, laying the foundation for the subsequent rendering process.

When it comes to image rendering, our team deliberated between two potential solutions: ***Raylib*** and ***GDI+***. The selection criteria included the ability to run seamlessly on both Windows and Ubuntu (Linux). After careful consideration, we opted for GDI+ due to its cross-platform compatibility and the specific requirements of our project. GDI+ provides a reliable framework for rendering SVG images, aligning with our project goals and ensuring compatibility across different operating systems.

2. UML design

Class diagram is the most commonly used subcategory of UML diagrams. It serves as the foundation for all object-oriented software systems. By examining the classes and properties of a system, users can visualize its static structure and determine how its classes are related to each other.

We design our abstract hierarchy, with the main class being *Shape* and other shapes inheriting the properties of *Shape*. Details regarding the stored variables and functions will be listed in the table below.

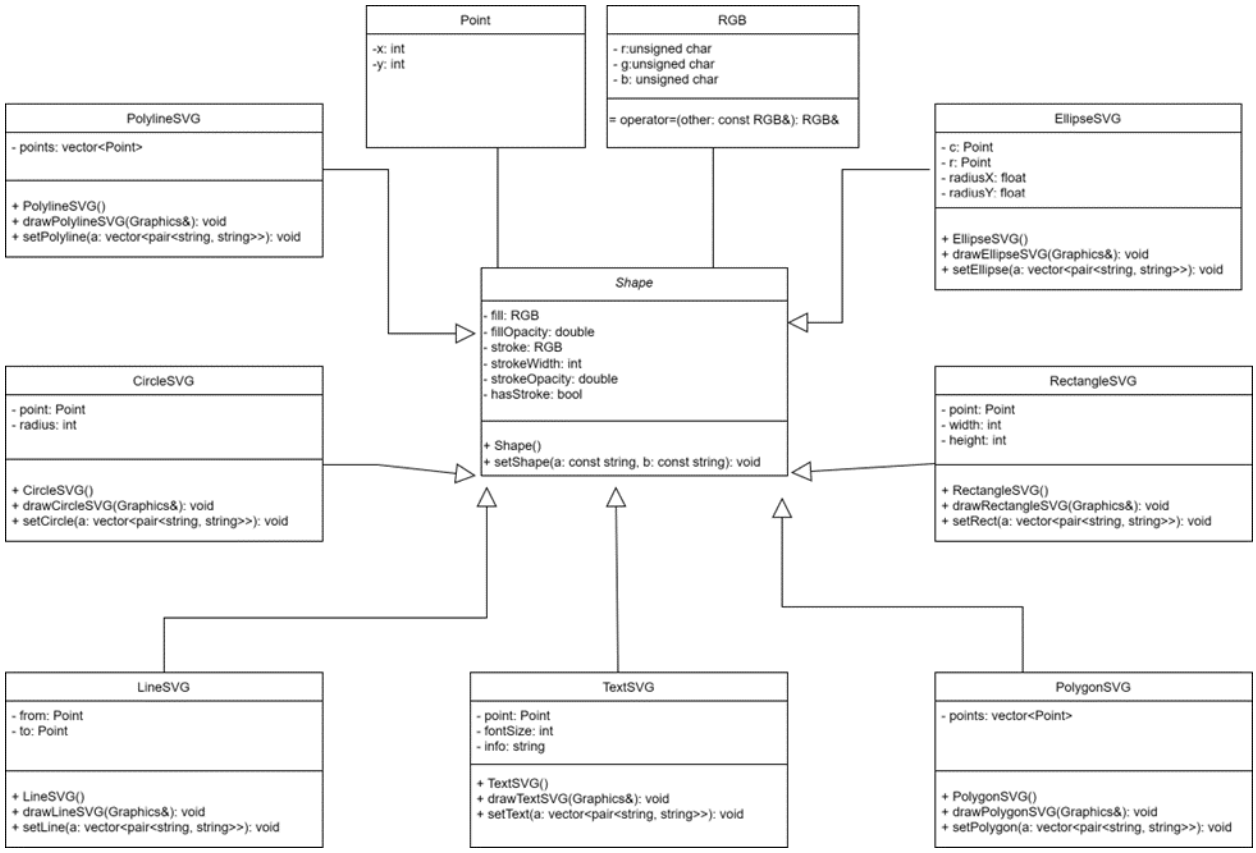

Table 1: Class and detail

No	Class name	Usage
1	Shape	- Base class for other shapes. - Store: ***color***, ***fill opacity,*** and ***stroke***.
2	RectangleSVG	- Inherit from ***Shape***. - Store: ***coordinates***, ***width***, and ***height*** of the rectangle. - Method: ***constructor*** + ***draw*** shape + ***set*** value
3	TextSVG	- Inherit from ***Shape***. - Stores: ***coordinates***, ***font size***, and ***content*** of the text. - Method: ***constructor*** + ***draw*** shape + ***set*** value
4	CircleSVG	- Inherit from ***Shape***. - Stores: ***the center coordinates*** and ***radius*** of the circle. - Method: ***constructor*** + ***draw*** shape + ***set*** value
5	PolylineSVG	- Inherit from ***Shape***. - Store: ***list of points*** to draw the polyline. - Method: ***constructor*** + ***draw*** shape + ***set*** value
6	EllipseSVG	- Inherit from ***Shape***. - Store: ***the center coordinates*** and ***radii along two axes*** of the ellipse. - Method: ***constructor*** + ***draw*** shape + ***set*** value
7	LineSVG	- Inherit from ***Shape***. - Store: **the *start*** and ***end points*** of the line. - Method: ***constructor*** + ***draw*** shape + ***set*** value
8	PolygonSVG	- Inherit from ***Shape***. - Store: ***list of points*** to draw the polygon. - Method: ***constructor*** + ***draw*** shape + ***set*** value

3. Result

Below are images from the computer and images rendered from the team's code.

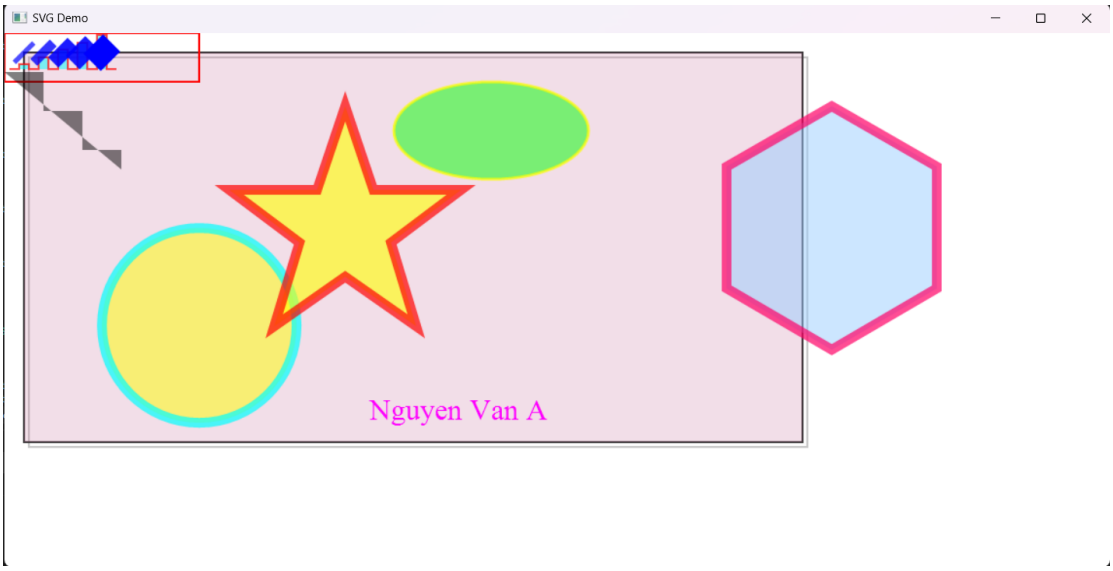

Figure 1: Image SVG from our code

Figure 2: Image SVG from computer

We have observed the following:

- **Speed Assessment:** It is at an acceptable level, quite fast, within a few seconds for a simple file. However, currently, we are not focusing on improving speed but rather concentrating on enhancing the similarity to the original image.

- **Rendering Comparison:** CGI renders align well with computer-generated images.

4. Challenge and Solution

Table 2: Challenges and solutions

Challenge	Solution
Common Challenges	
Difficulty in approaching and using external libraries.	- Conduct comprehensive research - Seek online resources - Engage with communities for AI assistance
Suboptimal utilization of object-oriented principles for code optimization.	- Reevaluate and optimize code structure using object-oriented principles for improved modularity
Specific Challenges	
Time-consuming accuracy checks for information extracted from SVG files.	- Implement parallel processing or optimize parsing algorithms for faster accuracy validation.
Data conversion to meet the input expectations of library functions.	- Develop a data preprocessing module for smooth data conversion and error handling.