Ho Chi Minh University of Science

**FACULTY OF INFORMATION AND TECHNOLOGY**

**Nguyen Tuan An (24127011)**

**Nguyen Hai Nam (24127088)**

**Pham Dinh Gia Khánh (24127420)**

**Vo An Phu (24127494)**

**PROJECT: SVG READER**

**Milestone 2 Report**

**Course: Object Oriented Programming**

**INSTRUCTOR**

ThS. Do Nguyen Kha

Ho Chi Minh City, August 2025

# Table of Content

[Table of Content ii](#_Toc203601421)

[List of Images iii](#_Toc203601422)

[List of Tables iv](#_Toc203601423)

[Chapter 1 Introduction 5](#_Toc203601424)

[Chapter 2 Report presentation 6](#_Toc203601425)

[2.1 Github Repository Link 6](#_Toc203601426)

[2.2 Github Commit List 6](#_Toc203601427)

[2.3 Github Contribution Screenshot 16](#_Toc203601428)

[2.4 Class Diagram 16](#_Toc203601429)

[2.5 Feature Checklist 18](#_Toc203601430)

[2.6 Screenshot of sample.svg 20](#_Toc203601431)

[2.7 Testing and Demonstration (Optional) 21](#_Toc203601432)

[2.8 Conclusion 22](#_Toc203601433)

[References 23](#_Toc203601434)

# List of Images

[Figure 2‑1 Commit Screenshot 1 6](#_Toc203601435)

[Figure 2‑2 Commit Screenshot 2 7](#_Toc203601436)

[Figure 2‑3 Commit Screenshot 3 7](#_Toc203601437)

[Figure 2‑4 Commit Screenshot 4 8](#_Toc203601438)

[Figure 2‑5 Commit Screenshot 5 8](#_Toc203601439)

[Figure 2‑6 Commit Screenshot 6 9](#_Toc203601440)

[Figure 2‑7 Commit Screenshot 7 9](#_Toc203601441)

[Figure 2‑8 Commit Screenshot 8 10](#_Toc203601442)

[Figure 2‑9 Commit Screenshot 9 10](#_Toc203601443)

[Figure 2‑10 Commit Screenshot 10 10](#_Toc203601444)

[Figure 2‑11 Commit Screenshot 11 11](#_Toc203601445)

[Figure 2‑12 Commit Screenshot 12 11](#_Toc203601446)

[Figure 2‑13 Commit Screenshot 13 12](#_Toc203601447)

[Figure 2‑14 Commit Screenshot 14 12](#_Toc203601448)

[Figure 2‑15 Commit Screenshot 15 13](#_Toc203601449)

[Figure 2‑16 Commit Screenshot 16 13](#_Toc203601450)

[Figure 2‑17 Commit Screenshot 17 14](#_Toc203601451)

[Figure 2‑18 Commit Screenshot 18 14](#_Toc203601452)

[Figure 2‑19 Commit Screenshot 19 15](#_Toc203601453)

[Figure 2‑20 Commit Screenshot 20 15](#_Toc203601454)

[Figure 2‑21 GitHub Contribution Graph 16](#_Toc203601455)

[Figure 2‑22 Class Diagram 17](#_Toc203601456)

[Figure 2‑23 Screenshot of “sample.svg” when starting the program 20](#_Toc203601457)

[Figure 2‑24 Screenshot of “sample.svg” with zoom and navigation 21](#_Toc203601458)

[Figure 2‑25 Screenshot of "dragon.svg" 21](#_Toc203601459)

[Figure 2‑26 Original Image 22](#_Toc203601460)

# List of Tables

[Table 2‑1 Feature Checklist 19](#_Toc203601461)

# Introduction

1. What the project does:
   1. It is a C/C++ program built using Visual Studio 2022 that can load, parse and render basic shapes. It uses WinAPI and GDI++ to display SVG elements like rectangle, circle, ellipse, … that are opened in SVG file.
2. Why we chose this topic:
   1. Because it allows us to explore how graphical data can be interpreted and displayed through code, turning static SVG files into visual output. It’s an ideal topic to practice graphics programming, to understand the structure of vector images, to work with (SVG) file and parsing techniques. The project is also practical, as SVG is a widely - used format in modern applications, and building a reader gives insight into how browsers and design tools handle graphics behind the scenes.
3. OOP concepts applied in this project:
   1. Inheritance
   2. Polymorphism
   3. Abstraction
   4. Encapsulation

# Report presentation

## Github Repository Link

Below is the link to our project Github Repository.

<https://github.com/anvn12/SVG-Reader>

## Github Commit List

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 1

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 2

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 3

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 4

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 5

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 6

A screenshot of a computer program

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 7

A screenshot of a computer program

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 8

A black rectangular object with white lines

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 9

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 10

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 11

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 12

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 13

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 14

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 15

A black screen with white lines

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 16

A black screen with white lines

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 17

A black screen with white lines

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 18

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 19

A black background with white lines

AI-generated content may be incorrect.

Figure ‑ Commit Screenshot 20

## Github Contribution Screenshot

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure ‑ GitHub Contribution Graph

## Class Diagram

We use Singleton design pattern in SVGReader class to manage all window attribute (like position, scale, list of shapes...) and method (like initialize, message loop, clean up,...) as this program is designed to display only 1 image, also it is simple to design. If user wants to display more than 1, just launch a new app. The concept is similar to a web browser tab (it only displays 1 image at a time).

We have a parent class called “SVGShape”, this class served as the foundation for other shapes classes in the program. The shapes will inherit attributes that a .svg shape needs to have such as Point2D position; RGBColor stroke, fill; float strokeWidth, strokeOpacity, fillOpacity. They would also inherit 2 very important methods that handles the attributes processing and drawing them which is processAttribute() and draw(). By designing it like this, we can easily add a new shape, or debug with ease as each of the shape are built on the parent class. Furthermore, we also make the “SVGCircle” shape inherited attributes from “SVGEllipse” as a circle is an Elip with major axis = minor axis, hence “rx = ry”.

We also have some other relationships between classes:

* SVGLine has a composition relationship with Point2D because each SVGLine owns its Point2D endpoints (e.g., start and end), and those points do not exist outside the line's context
* SVGShape has a composition relationship with Point2D and RGBColor because it exclusively owns the instances that define its position and color, and they do not exist independently of the shape.
* SVGReader has an aggregation relationship with SVGShape because it references or uses SVGShape objects, but they can exist and be managed independently of the reader.

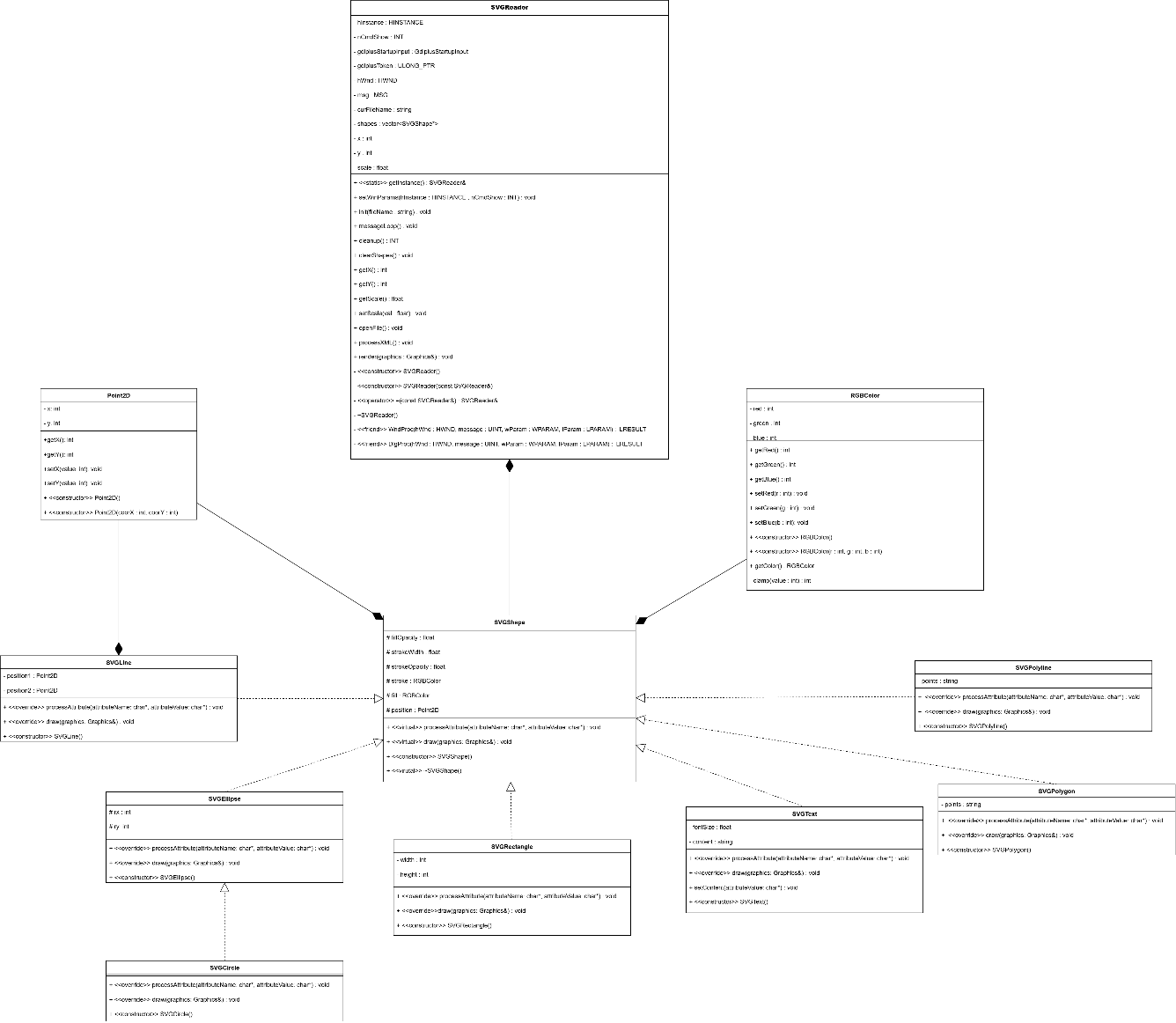


Figure ‑ Class Diagram

## Feature Checklist

|  |  |  |
| --- | --- | --- |
| Feature | Status | Comments |
| SVG Reading and Parsing | ✅ | Utilize Rapid XML library. |
| Rendering | ✅ | Render image to the screen with windows.h and GDIplus library. |
| Object-Oriented Design | ✅ | Singleton pattern (SVGReader class), utilize inheritance, encapsulation, abstraction and polymorphism. |
| Documentation | ✅ | Comment code, include UML diagram representing relationships between classes. |
| Navigation image | ✅ | User can navigate the rendered image by using arrow keys. |
| Zoom | ✅ | User can zoom in and out the rendered image by using I or O key. |
| Change the program default icon | ✅ | Change the small icon and app icon to those given by the lecturer. |
| Toolbar | ✅ | User can access to toolbar (File, View, Help) on top of the program. |
| Open new SVG image in the toolbar | ✅ | User can open to view other SVG images in the toolbar (File, Open). |
| Warning before exit | ✅ | Show the warning box to make sure that user really wants to exit the program. |
| Exit in the toolbar | ✅ | User can exit the program in the toolbar (File, Exit). |
| Zoom In/Out/Reset Zoom in the toolbar | ✅ | User can zoom in/out or reset zoom in the toolbar (View). |
| About in the toolbar | ✅ | User can see the introduction, the current version, the list of developers in the toolbar (Help, About). |
| Guide in the toolbar | ✅ | User can see the instruction for using the program in the toolbar (Help, Guide). |
| Navigation and zoom with mouse | ❌ | User can use mouse to navigate the image, can hold Ctrl and scroll the mouse scroll wheel. |
| Renderer class | ❌ | Use renderer class to render all items including SVG image, screen, toolbar… in the program |

Table ‑ Feature Checklist

## Screenshot of sample.svg

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Screenshot of “sample.svg” when starting the program

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‑ Screenshot of “sample.svg” with zoom and navigation

## Testing and Demonstration (Optional)

A green dragon with a black circle

Description automatically generated

Figure ‑ Screenshot of "dragon.svg"

A green dragon with a black circle and a black circle

Description automatically generated

Figure ‑ Original Image

NOTE: The result may be different because the browser has some default attribute values.

## Conclusion

We have learned to apply the concepts of OOP (inheritance, encapsulation, abstraction and polymorphism) in our project. Although every concept has its own role in our project, we use encapsulation in most of our code because it helps hide the attributes and functions. Some challenges that we faced while doing project can be mentioned as: lack of practical experience with OOP, misunderstanding of encapsulation and abstraction in OOP, lack of motivation while doing project. As we can figure out our challenges, we would improve by doing and applying OOP concepts in more projects in the future.

# References

* Library used: Rapid XML, windows.h, GDIplus
* Tutorials or code examples consulted:

<http://rapidxml.sourceforge.net/manual.html>

<https://learn.microsoft.com/vi-vn/windows/win32/>

<https://www.youtube.com/playlist?list=PLWzp0Bbyy_3i750dsUj7yq4JrPOIUR_NK>

<https://github.com/danielblagy/dewcin_yt>