ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

CSC10003 - Object-Oriented Programming

BÁO CÁO ĐỒ ÁN:

SVG READER - 3rd milestone

Giảng viên hướng dẫn: ThS. Đỗ Nguyên Kha

Nhóm sinh viên: Phan Hữu Trọng Phúc - 24120122
Võ Thành Hoan - 24120052
Đỗ Chí Cao - 24120270
Lâm Chí Khanh - 24120337

Thành phố Hồ Chí Minh, Tháng 1/2026

Mục lục

Chương 1

GIỚI THIỆU VÀ CẤU TRÚC

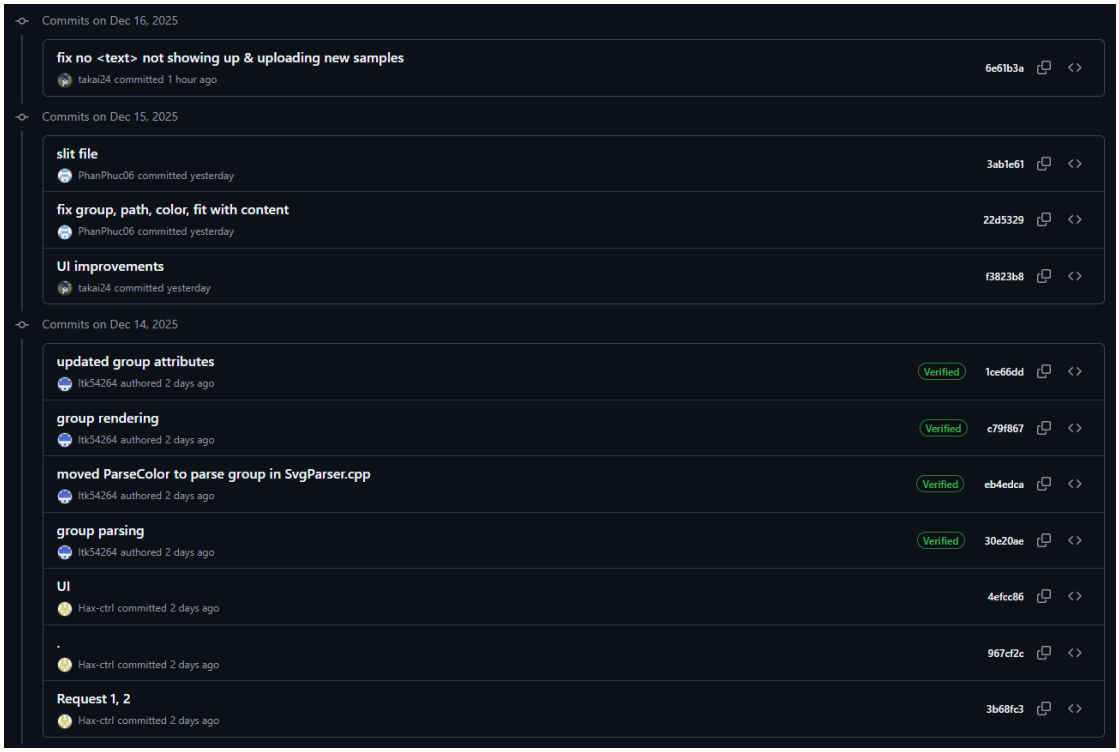
1.1 Giới thiệu về đồ án

SVG Reader là một ứng dụng đọc file định dạng SVG (Scalable Vector Graphics) và hiển thị nội dung của file đó lên giao diện người dùng. Ứng dụng này được xây dựng bằng ngôn ngữ lập trình C++ và sử dụng thư viện GDIPlus và RapidXML để tạo giao diện đồ họa và đọc file SVG.

1.2 GitHub Repository

Đồ án có thể được truy cập qua đường dẫn sau: https://github.com/takai24/CSC10003-24CTT6-OOP

1.3 Lịch sử Commit (GitHub Commit List)



1.4 Đóng góp của thành viên (Contribution)

Dưới đây là bảng thống kê chi tiết khối lượng công việc và tỷ lệ đóng góp của các thành viên trong nhóm cho dự án SVG Render (Milestone 3):

STT	MSSV	Thành viên	Nội dung đóng góp	Tỷ lệ
1	24120052	Võ Thành Hoan	Phụ trách kiến trúc hệ thống, cài đặt `SvgPaintServer` và cơ chế kế thừa `xlink:href`.	25%
2	24120270	Đỗ Chí Cao	Triển khai thuật toán **Texture Baking** trong `SvgPaintResolver` và xử lý Radial Gradient.	25%
3	24120337	Lâm Tuấn Khanh	Xử lý phần Linear Gradient, Multi-stops và các phép biến đổi ma trận `gradientTransform`.	25%
4	24120122	Phan Hữu Trọng Phúc	Quản lý việc Render các Element (`Path`, `Text`), kiểm thử với các , đăng kết quả lên group Facebook, viết báo cáo	25%
			Tổng cộng	**100%**

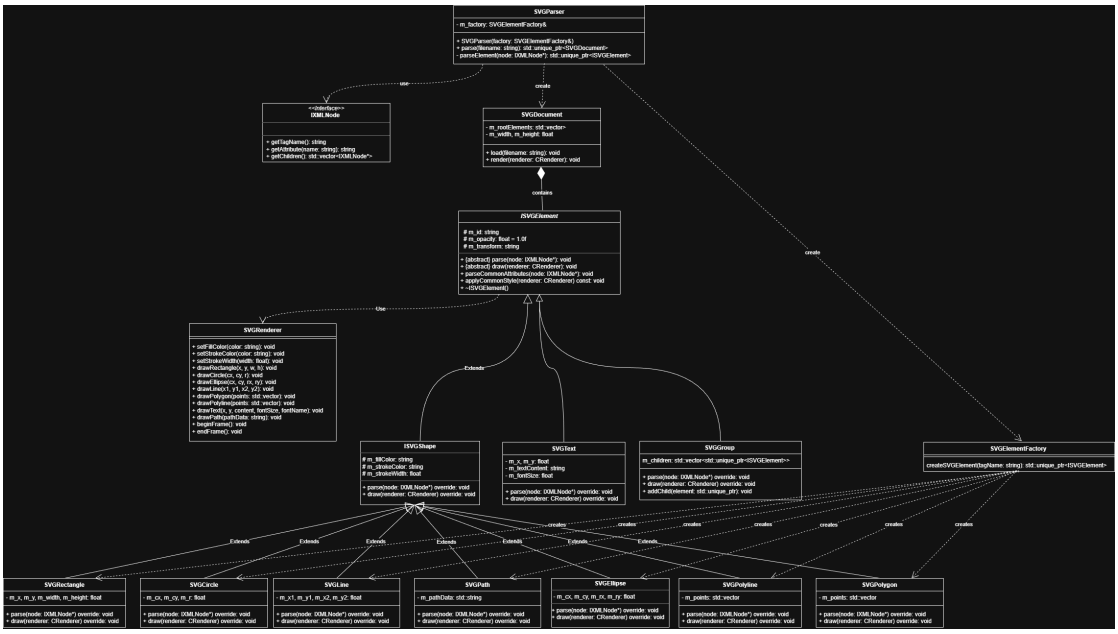
Ghi chú: Các thành viên đều tham gia vào quá trình thảo luận chung, review code và hỗ trợ giải quyết các xung đột (conflict) trên Git trong suốt quá trình thực hiện đồ án.

1.5 Sơ đồ lớp (Class Diagram)

Chúng em áp dụng các kỹ thuật Lập trình hướng đối tượng sau:

1. **Polymorphism và Kế thừa:** Sử dụng lớp cơ sở `SvgElement` để định nghĩa hành vi chung cho các phần tử.
2. **Factory Pattern:** Sử dụng `SvgElementFactory` để tập trung hóa việc tạo đối tượng.
3. **Abstraction:**
   - Interface `IRenderer` (triển khai bởi `GdiPlusRenderer`) giúp tách biệt phần vẽ, dễ dàng chuyển đổi API đồ họa (Direct2D/OpenGL).
   - Interface `IXMLNode` và Adapter giúp tách biệt logic phân tích XML khỏi thư viện RapidXML.

Nhờ cấu trúc này, mã xử lý cốt lõi chỉ làm việc với các kiểu cơ sở (`SvgElement*`, `IRenderer*`), đảm bảo dễ bảo trì và mở rộng.



Hình 1.1: Sơ đồ lớp (Class Diagram) của dự án

1.6 Cơ chế Quản lý và Phân giải (Paint Server Resolver)

Để xử lý các yêu cầu phức tạp của Milestone 3, đặc biệt là tính kế thừa và sự sai khác giữa hệ tọa độ SVG và GDI+, nhóm đã triển khai hệ thống phân giải màu sắc (Paint Server) với các thành phần chính sau:

1.6.1 Cấu trúc dữ liệu và Quản lý đối tượng

- **SvgGradient (Data Models)**: Định nghĩa cấu trúc dữ liệu phân cấp. `SvgGradient` đóng vai trò lớp cơ sở, trong khi `SvgLinearGradient` và `SvgRadialGradient` lưu trữ các thuộc tính chuyên biệt như tọa độ vector (x1, y1, x2, y2) hoặc các thông số hình tròn (cx, cy, r, fx, fy).
- **SvgPaintServer (Business Logic)**: Đóng vai trò là một kho lưu trữ (Registry) trung tâm.
  - Quản lý toàn bộ Gradient theo ID.
  - Thực hiện hàm `ResolveGradients()`: Giải quyết cơ chế kế thừa thông qua `xlink:href`, đảm bảo các thuộc tính từ gradient cha được sao chép đầy đủ xuống các bản ghi con.

1.6.2 Kỹ thuật Render và Chuyển đổi Hệ tọa độ

- **SvgPaintResolver (Core Rendering Engine)**: Hạt nhân xử lý thuật toán đồ họa của nhóm.
  - Triển khai kỹ thuật **Bitmap Texture Baking**: Tạo ra một mẫu Texture độ phân giải cao để khắc phục giới hạn của GDI+ đối với các Radial Gradient bị méo hoặc biến dạng.
  - Hàm `CreateBrush()`: Thực hiện ánh xạ từ không gian SVG sang hệ tọa độ màn hình, xử lý chính xác các chế độ `objectBoundingBox` và các phép biến đổi `gradientTransform`.
  - Xử lý `SpreadMethod`: Hiện thực hóa các logic lặp màu như Pad, Reflect và Repeat.
- **GdiPlusGradientRenderer (Wrapper/Adapter)**: Đóng vai trò lớp đệm giúp chuẩn hóa lời gọi hàm. Thay vì các hàm vẽ phải xử lý trực tiếp logic phức tạp của Gradient, chúng chỉ cần gọi thông qua lớp này để nhận về một đối tượng `Brush` hoàn chỉnh, giúp mã nguồn sạch (clean code) và dễ bảo trì.

1.6.3 Tổng quan luồng dữ liệu (Data Flow)

Quy trình thực thi được tóm tắt qua 4 bước:

1. **Parsing**: Trình phân tích đọc thẻ màu và tạo đối tượng dữ liệu `SvgGradient`.

2. **Resolving**: `SvgPaintServer` điền các thông tin thiếu thông qua liên kết kế thừa.

3. **Requesting**: Khi vẽ đối tượng, Renderer yêu cầu một Brush từ `GdiPlusGradientRenderer`.

4. **Baking & Rendering**: `SvgPaintResolver` tính toán ma trận, "nướng" (bake) texture và trả về cọ vẽ để đổ màu lên màn hình.

1.7 Danh sách tính năng (Feature Checklist)

Trạng thái các tính năng sau khi hoàn thiện Milestone 3:

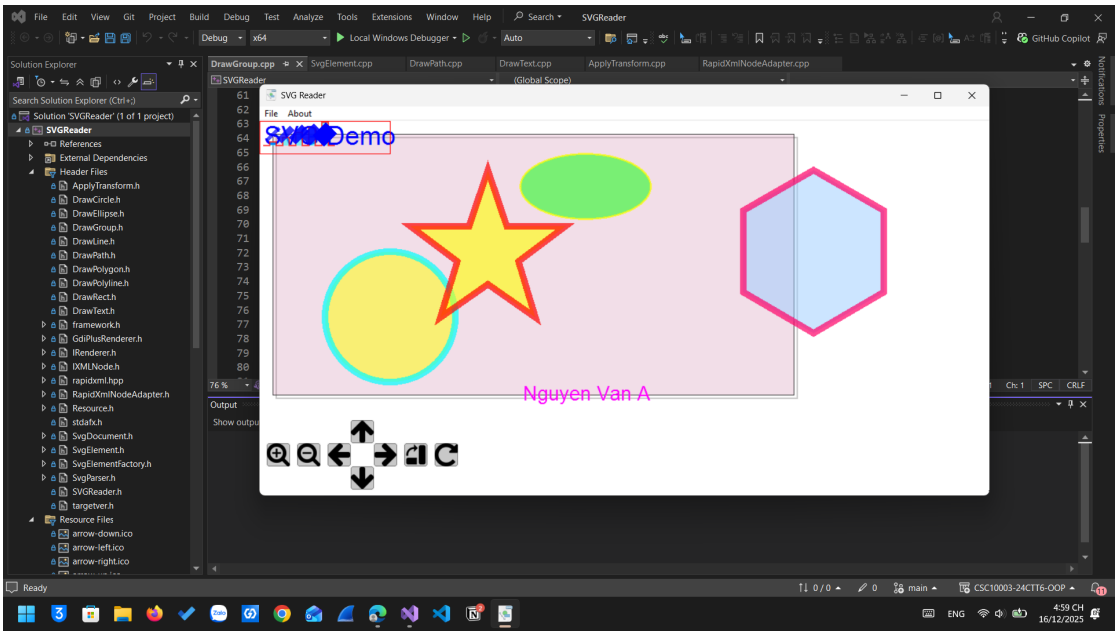
Tính năng	**Trạng thái**	**Ghi chú**
Đọc file .svg	Hoàn thiện	Hỗ trợ cấu trúc phân cấp phức tạp và thẻ `<defs>`.
Vẽ file .svg trong Window	Hoàn thiện	Tối ưu hóa render bằng GDI+.
Kiến trúc OOP	Hoàn thiện	Áp dụng **Composite Pattern** và hệ thống **Paint Server** để quản lý màu sắc.
Thông tin nhóm và đồ án	Hoàn thiện	Hiển thị đầy đủ trên Title bar.
Render màu sắc	Hoàn thiện	Đã hỗ trợ đầy đủ các hệ màu, bao gồm cả màu có độ trong suốt (Alpha channel).
Render Linear Gradient	Hoàn thiện	Xử lý đa điểm dừng màu (Multi-stops) và `gradientTransform`.
Render Radial Gradient	Hoàn thiện	Sử dụng kỹ thuật **Bitmap Texture Baking** để xử lý các gradient lệch tâm hoặc bị méo.
Kế thừa Gradient	Hoàn thiện	Giải quyết triệt để tham chiếu `xlink:href` giữa các định nghĩa màu.
Render các thẻ Element	Hoàn thiện	Đã fix các lỗi render `<text>` và hỗ trợ tốt `<path>`, `<group>`.
Thu phóng và Xoay ảnh	Hoàn thiện	Hỗ trợ tương tác mượt mà qua ma trận biến đổi toàn cục.
Spread Method	Hoàn thiện	Hỗ trợ đầy đủ các chế độ: Pad, Reflect và Repeat.

Chương 2

DEMO VÀ KẾT LUẬN

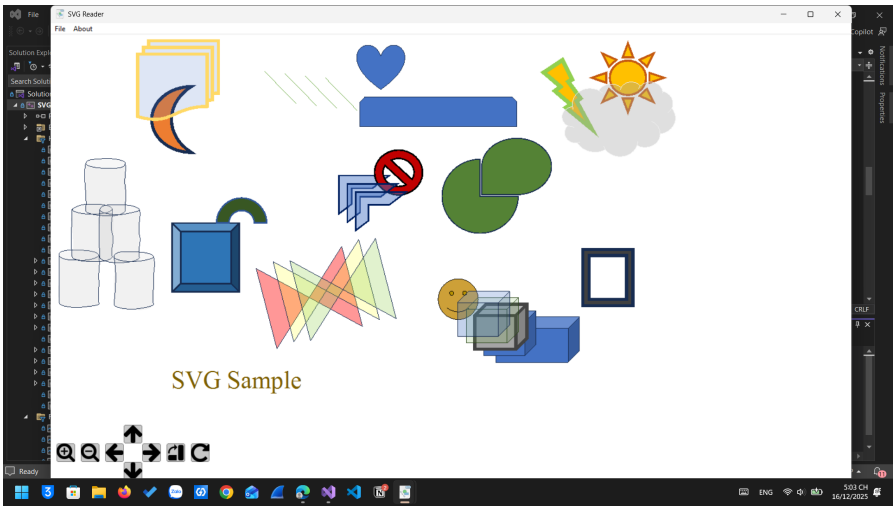
2.1 Kết quả chạy thử nghiệm

2.1.1 Hiển thị file sample.svg

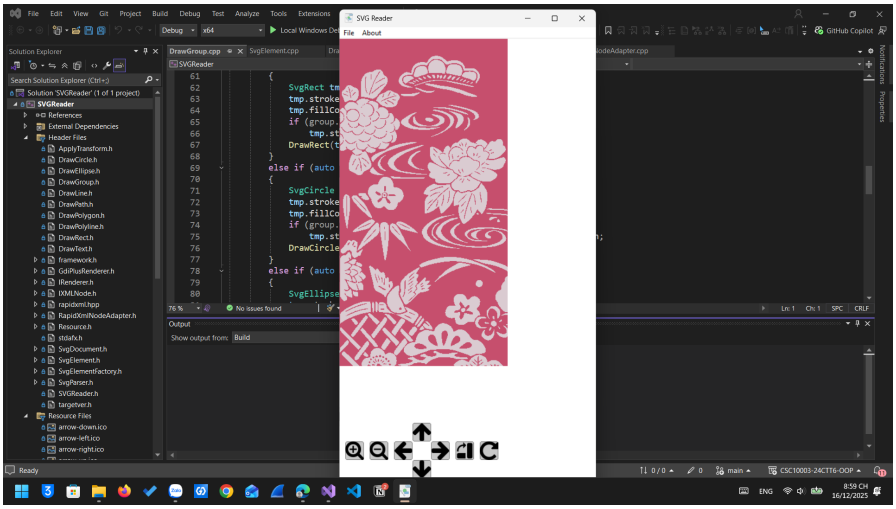


Hình 2.1: Ảnh chụp màn hình với file sample.svg

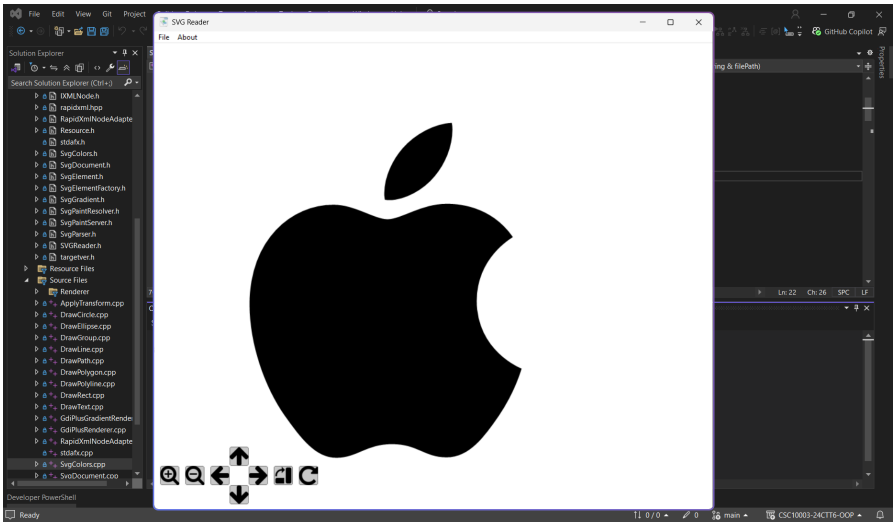
2.1.2 Test với các hình dạng khác



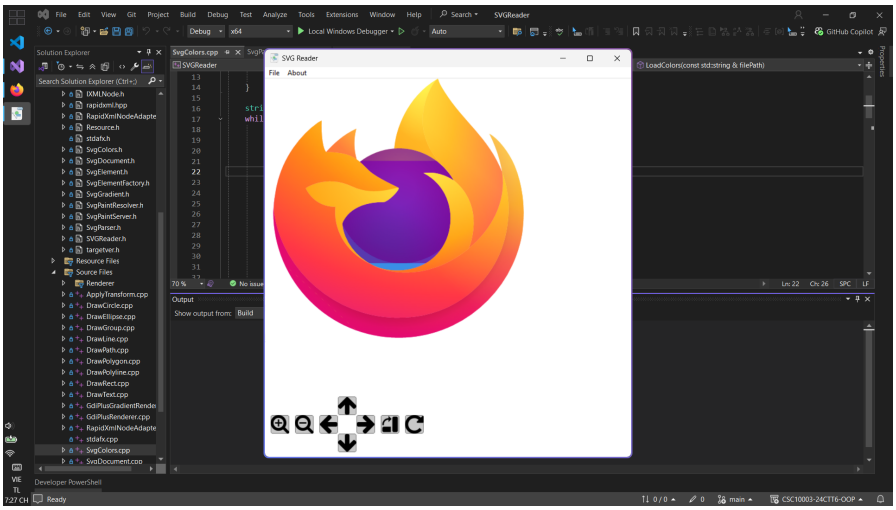
Hình 2.2: Thử nghiệm hiển thị Group và các hình cơ bản



Hình 2.3: Thử nghiệm hiển thị Group và Path



Hình 2.4: Thử nghiệm hiển thị logo Apple



Hình 2.5: Thử nghiệm hiển thị logo trình duyệt FireFox

2.2 Link video demo vẽ 5 logo theo đề yêu cầu

https://drive.google.com/drive/folders/1FwMy9BtTRquu-MAD_Ev66nAKr54evlr-?usp=sharing

2.3 Kết luận (Conclusion)

2.3.1 Kết quả đạt được

Đến thời điểm hiện tại (Milestone 3), đồ án đã hoàn thành đầy đủ các yêu cầu trọng tâm của môn học. Thành quả lớn nhất là xây dựng được một công cụ render SVG có khả năng xử lý các hình ảnh vector phức tạp (như logo Firefox, Instagram). Các kết quả chính bao gồm:

- **Cấu trúc hệ thống vững chắc**: Áp dụng triệt để **Composite Pattern** để quản lý cấu trúc cây của các thẻ SVG và **Factory Method** trong việc khởi tạo đối tượng.
- **Xử lý Gradient nâng cao**: Giải quyết thành công bài toán render `linearGradient` và `radialGradient`. Đặc biệt là kỹ thuật **Bitmap Texture Baking** giúp vượt qua giới hạn của thư viện GDI+ truyền thống.
- **Cơ chế phân giải màu sắc**: Hệ thống `Paint Server` đã xử lý tốt các logic kế thừa thuộc tính phức tạp (`xlink:href`) và chuyển đổi hệ tọa độ (`objectBoundingBox`).

2.3.2 Hạn chế và Hướng phát triển

Mặc dù đã đạt được những tiến bộ vượt bậc, đồ án vẫn còn một số điểm có thể tối ưu thêm:

- **Hiệu năng**: Kỹ thuật tạo Bitmap Texture cho Gradient tuy chính xác nhưng khi chạy với logo Instagram và FireFox nó vẫn không vẽ hoàn chỉnh 100
- **Tính năng mở rộng**: Trong tương lai, nhóm dự kiến phát triển thêm khả năng render các thẻ `<filter>` (như đổ bóng, mờ - Gaussian Blur) và thẻ `<pattern>` để làm phong phú hơn khả năng hiển thị.
- **Giao diện**: Cải thiện UI để hỗ trợ kéo thả file SVG trực tiếp và quản lý danh sách layer của hình vẽ.

2.4 Tài liệu tham khảo

- **Design Patterns**: *Refactoring.Guru* - Composite Pattern, Factory Method, và Singleton Pattern.
- **Đồ họa máy tính**: Tài liệu MSDN của Microsoft về thư viện GDI+ (Graphics, Brushes, và Matrix Transformations).

- **Chuẩn SVG**: Tài liệu đặc tả kỹ thuật của W3C về cấu trúc thẻ Gradient và các phép biến đổi ma trận trong SVG.
- **Xử lý XML**: Thư viện `RapidXML` dành cho C++.