

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



## OBJECT-ORIENTED PROGRAMMING

---

### REPORT OF PROJECT

#### NaTruKi: An SVG Viewer Application

---

Nhóm trưởng: Nguyễn Phúc Khang - 24120068  
Thành viên: Hoàng Trọng Nghĩa - 24120103  
Thành viên: Phan Chí Cao - 24120026  
Thành viên: Nguyễn Hoàng Nhật - 24120110

# Mục lục

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>GitHub Repository Link</b>	<b>1</b>
<b>3</b>	<b>GitHub Commit List</b>	<b>1</b>
<b>4</b>	<b>GitHub Contribution</b>	<b>4</b>
<b>5</b>	<b>Class Diagram</b>	<b>4</b>
5.1	Xử lý SVGPath . . . . .	5
5.2	Xử lý SVGGroup - Pattern Composite . . . . .	6
5.3	Mở rộng tính năng cho dự án có sẵn . . . . .	6
5.4	Feature Checklist . . . . .	7
<b>6</b>	<b>Results</b>	<b>7</b>
<b>7</b>	<b>Conclusion</b>	<b>7</b>



## 1 Introduction

Sau khi đã hoàn thành **milestone-1** nhóm đã có được một ứng dụng cơ bản, có giao diện cho phép người dùng có thể chọn các file **svg** bất kỳ để hiển thị lên màn hình. Ứng dụng cũng đã được xây dựng để có thể hỗ trợ được các tính năng cơ bản cho việc tùy biến cũng như là nhu cầu của người dùng như là: *Xoay, Phóng to, Thu nhỏ, Lật ngược*. Hiện tại, ứng dụng của nhóm đã có thể hỗ trợ các **tag** cơ bản của một file **svg** để có thể hiện thị các hình dạng cơ bản như: **<rect>**, **<circle>**, **<ellipse>**, **<line>**, **<polyline>**, **<polygon>**, **<text>**.

Tuy nhiên, nhu cầu của người dùng không chỉ dừng lại ở đó, các file **svg** được yêu cầu phải có thể chứa các hình ảnh phức tạp, có đường cong, đường uốn lượn, đường xiên ngang từ đó mà tạo nên tính sáng tạo và đa dạng của bức tranh hơn. Do đó nhóm đã thêm các tag để thực hiện được nhu cầu đó chính là: **<path>**. Việc thêm **<path>** nó cũng không phải là một thách thức quá lớn cho nhóm vì ngay từ ban đầu, nhóm đã thống nhất xây dựng theo các **design pattern** nào đó, thống nhất với về cách code, cũng như là tổ chức quản lý, do đó việc bảo trì và thêm tính năng đã được đơn giản hóa đi rất nhiều.

## 2 GitHub Repository Link

Nhằm phục vụ cho việc quản lý dự án, cũng như là công khai các mã nguồn có trong dự án thì nhóm cũng đã đẩy toàn bộ dự án lên trên [GitHub](#):

**Link:** [See more](#)

## 3 GitHub Commit List

Để cung cấp một cái nhìn minh bạch và toàn diện về quá trình phát triển của dự án, dưới đây là toàn bộ lịch sử commit được trích xuất từ repository. Danh sách này được trình bày theo định dạng rút gọn, thể hiện các thay đổi, gộp nhánh và các cột mốc quan trọng từ khi bắt đầu cho đến khi hoàn thiện dự án.

```
e72f2ea Nguyen Phuc Khang 2025-12-16
  merge to main
3bd57d2 Nguyen Phuc Khang 2025-12-16
  something fix
988844c pumpomeo 2025-12-16
  fix: scale function
4029ff6 pumpomeo 2025-12-16
  copy of B#2
```



```
9 5e7f9c5 nh996 2025-12-10
10     FINAL (word alignment+color+rotation)
11 8360949 nh996 2025-12-08
12     rotation on flipped image fix
13 c013f31 nh996 2025-11-22
14     final
15 994a033 nh996 2025-11-21
16     Add SVGPath element to "finish app"
17 e025c89 Nguyen Phuc Khang 2025-11-20
18     merge files
19 bf974f5 Nguyen Phuc Khang 2025-11-20
20     Merge branch 'main' of https://github.com/khang1108/natrukiSVG
21 9b68759 Nguyen Phuc Khang 2025-11-20
22     add explaination for functions and add SVGPath
23 d73a87a Nguyen Phuc Khang 2025-11-19
24     Update README.md (#6)
25 7b0cb8d Nguyen Phuc Khang 2025-11-19
26     add some new features
27 cc31aad Nguyen Phuc Khang 2025-11-19
28     finish application
29 71746d6 Nguyen Phuc Khang 2025-11-16
30     Remove build artifacts and IDE files from repository
31 60527b6 Nguyen Phuc Khang 2025-11-16
32     fix qt
33 8761aa0 Nguyen Phuc Khang 2025-11-16
34     Merge remote-tracking branch 'origin/classdefining'
35 95451bc cpgod36 2025-11-15
36     sua loi noi bo
37 0fa78f4 Nguyen Phuc Khang 2025-11-15
38     fix main cmake
39 ca6b2c6 cpgod36 2025-11-12
40     sua nut size
41 b143c1a cpgod36 2025-11-12
42     hehe
43 c81019d cpgod36 2025-11-12
44     latest
45 26235ae nh996 2025-11-11
46 .
47 f8defdc nh996 2025-11-11
48     Merge branch 'Parser+SceneGraph' of https://github.com/khang1108/natrukiSVG into Parser+SceneGraph
49 e2d8172 nh996 2025-11-11
50     svgfactory adjustment
51 f4af068 cpgod36 2025-11-11
52     moi nhat
53 2a7db1e nh996 2025-11-10
54     Delete .vscode directory
55 86a297d nh996 2025-11-10
56     Delete build directory
57 138dbf0 nh996 2025-11-10
```



```
58     bug fix#1
59 bf6168e nh996 2025-11-10
60     roleB
61 6f85fc5 cpgod36 2025-11-06
62     add some actions on mainwindow
63 c393cid cpgod36 2025-11-06
64     resources/premium added
65 7f29609 cpgod36 2025-11-06
66     da doi svg thanh png
67 3490f48 cpgod36 2025-11-06
68     folder resources/about added
69 b35b95d cpgod36 2025-11-06
70     upload resources
71 9211e62 cpgod36 2025-11-06
72     include folder completed
73 86c759d cpgod36 2025-11-05
74     Merge remote-tracking branch 'origin/main' into classdefining
75 da0ab28 cpgod36 2025-11-05
76     Role A: folder /include/svg completed
77 497ded4 Nguyen Phuc Khang 2025-11-05
78     Fix README.md
79 ed68102 Nguyen Phuc Khang 2025-11-05
80     update CMakeLists
81 b84fef6 cpgod36 2025-11-05
82     define some classes
83 9ca7645 Nguyen Phuc Khang 2025-10-30
84     Update contributors section with GitHub links
85 e8262c0 Nguyen Phuc Khang 2025-10-30
86     Update class reference in README.md (#3)
87 ebbbeb89 Nguyen Phuc Khang 2025-10-30
88     Enhance README with project details and structure (#2)
89 437b1f5 Nguyen Phuc Khang 2025-10-30
90     Delete .doc directory (#1)
91 6a36eaa Nguyen Phuc Khang 2025-10-30
92     update folder
93 8737269 Nguyen Phuc Khang 2025-10-30
94     update folder strucutre
95 9df7af2 Nguyen Phuc Khang 2025-10-08
96     Update README.md
97 f5e5b3f Nguyen Phuc Khang 2025-10-08
98     Initial commit
```

Listing 1: Lịch sử commit đầy đủ của dự án NaTruKi



## 4 GitHub Contribution

Bằng việc sử dụng GitHub để quản lý dự án, cũng như là tổ chức và xây dựng thì nhóm cũng đã có thể ghi lại những hoạt động và đóng góp của từng thành viên như sau:

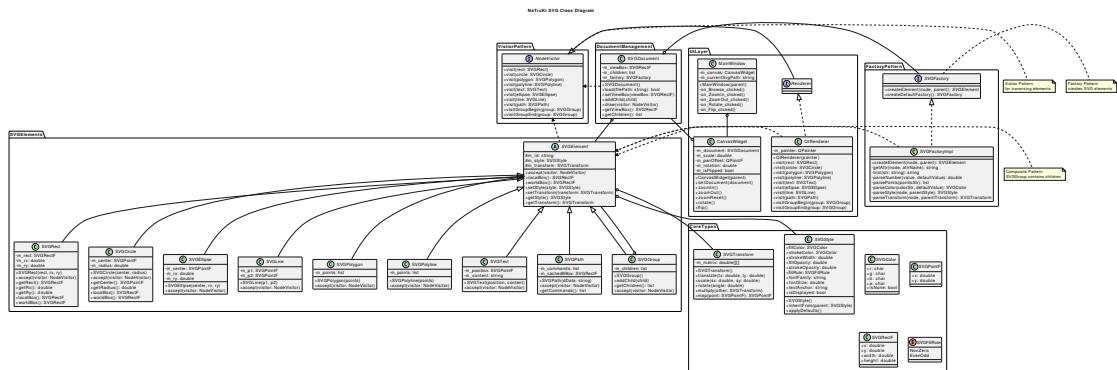


Hình 1: GitHub Contributions stats

Qua hình ảnh này, ta có thể thấy được mức độ hoạt động của các thành viên trong nhóm. Vì dự án này không quá nặng về code nên mà thiên về tìm hiểu nội dung, các thành viên chủ yếu hoạt động bên [Google Docs](#) nên trên Github sẽ không quá nổi bật.

## 5 Class Diagram

Dể trực quan hóa được ý tưởng về các tổ chức và xây dựng dự án này của nhóm thì ta có thể xem qua *Class Diagram - Unified Model Language (UML)*, đây chính là sơ đồ tổng thể toàn bộ các mối quan hệ của các class có trong dự án, cũng như là các thuộc tính, các phương thức được xây dựng cho mỗi class.



Hình 2: Class Diagram của dự án NaTruKi

Để xem rõ hơn, nhóm cũng đã có để chi tiết [Class Diagram](#) ở trong [resources/plantuml.svg](#).

### 5.1 Xử lý SVGPath

SVGPath là phần tử phức tạp nhất trong SVG vì nó có thể biểu diễn bất kỳ hình dạng nào thông qua chuỗi lệnh trong thuộc tính `d`. Một path có thể chứa các lệnh di chuyển (`M/m`), vẽ đường thẳng (`L/l, H/h, V/v`), đường cong Bezier (`C/c, S/s, Q/q, T/t`), và đóng đường (`Z/z`). Điểm đặc biệt là mỗi lệnh có hai dạng: chữ hoa (tọa độ tuyệt đối) và chữ thường (tọa độ tương đối), đồng thời một số lệnh có thể lặp lại nhiều tham số mà không cần ghi lại ký tự lệnh.

Để xử lý path, dự án sử dụng thuật toán phân tích cú pháp dựa trên máy trạng thái hữu hạn (Finite State Machine) [1]. Thuật toán duyệt từng ký tự trong chuỗi `d`, phân biệt giữa lệnh và tham số số, sau đó chuyển đổi các tọa độ tương đối thành tuyệt đối để thuận tiện cho tính toán. Đối với rendering, các lệnh path được chuyển thành `QPainterPath` của Qt6, trong khi việc tính bounding box yêu cầu duyệt qua tất cả các điểm điều khiển và điểm cuối của mỗi lệnh. Một thách thức quan trọng là xử lý lệnh `M` đặc biệt: tọa độ đầu tiên sau `M` là `moveTo`, nhưng các tọa độ tiếp theo được hiểu ngầm định là `lineTo`.

Lệnh	Ý nghĩa	Xử lý
<code>M/m</code>	Move to (di chuyển bút)	Đặt vị trí hiện tại, sau đó implicit <code>L</code>
<code>L/l, H/h, V/v</code>	Line to (vẽ đường thẳng)	Chuyển thành <code>lineTo</code> trong <code>QPainterPath</code>
<code>C/c, S/s, Q/q, T/t</code>	Bezier curves	Chuyển thành <code>cubicTo</code> hoặc <code>quadTo</code>
<code>A/a</code>	Elliptical arc	Tính toán cung ellipse phức tạp
<code>Z/z</code>	Close path	Nối điểm cuối về điểm bắt đầu subpath

Bảng 1: Các lệnh SVG Path và cách xử lý



## 5.2 Xử lý SVGGroup - Pattern Composite

SVGGGroup đại diện cho thẻ `<g>` trong SVG [2] cho phép nhóm nhiều phần tử con lại với nhau và áp dụng style cũng như transform chung. Đây chính là thể hiện của Composite Pattern trong thiết kế, nơi SVGGGroup vừa là SVGElemetn vừa chứa danh sách các SVGElemetn con. Khi một group có transform hoặc style, các thuộc tính này được kế thừa xuống tất cả các phần tử con theo cơ chế cascade tương tự CSS.

Thuật toán xử lý group dựa trên đệ quy và Visitor Pattern. Khi duyệt cây SVG, renderer gọi `visitGroupBegin` để lưu trạng thái hiện tại của QPainter (save), áp dụng transform của group, sau đó đệ quy duyệt tất cả các con, cuối cùng gọi `visitGroupEnd` để khôi phục trạng thái (restore). Điều này đảm bảo transform của group chỉ ảnh hưởng đến các con mà không làm thay đổi các phần tử khác. Đối với bounding box, thuật toán phải tính `worldBox` của tất cả các con (đã bao gồm transform của chúng), tìm hình chữ nhật bao quanh tất cả, rồi mới áp dụng transform của chính group đó lên kết quả.

Thao tác	Path	Group
Parse dữ liệu	Phân tích chuỗi d phức tạp	Đơn giản, chỉ chứa children
Tính bounding box	Duyệt tất cả lệnh và điểm	Hợp nhất box của tất cả con
Rendering	Vẽ trực tiếp QPainterPath	Đệ quy render các con
Transform	Áp dụng lên toàn bộ path	Kế thừa xuống các con

Bảng 2: So sánh xử lý SVGPath và SVGGGroup

## 5.3 Mở rộng tính năng cho dự án có sẵn

Việc thêm tính năng vào một dự án SVG viewer có sẵn đòi hỏi hiểu rõ kiến trúc hiện tại và áp dụng các design pattern phù hợp. Trong trường hợp dự án NaTruKi, kiến trúc ban đầu đã có sẵn xương sống cho các shape cơ bản (rect, circle, ellipse) với Visitor Pattern để rendering và Factory Pattern để parse XML. Khi cần thêm SVGPath và SVGGGroup, nhóm phát triển không phá vỡ code cũ mà mở rộng theo nguyên tắc Open-Closed Principle.

Quy trình thêm tính năng bắt đầu bằng việc tạo class mới kế thừa từ SVGElemetn, implement các phương thức ảo thuần túy như `accept`, `localBox`, và `worldBox`. Tiếp theo, các phương thức `visit` mới được thêm vào interface NodeVisitor và implement trong QtRenderer. Factory cần được cập nhật để nhận diện và tạo phần tử mới từ XML node tương ứng. Điểm quan trọng là phải test kỹ lưỡng với các file SVG thực tế, đặc biệt chú ý đến các trường hợp edge case như path có lệnh lowercase, group lồng nhau nhiều tầng, hoặc transform phức tạp. Cách tiếp cận này



dảm bảo code mới không ảnh hưởng đến chức năng cũ đã hoạt động ổn định.

#### 5.4 Feature Checklist

Trong quá trình phát triển dự án, nhóm đã hoàn thành các tính năng sau đây trong milestone thứ hai:

Bảng 3: Feature Checklist - Milestone 2

STT	Tính năng	Trạng thái
1	Xử lý viewBox và preserveAspectRatio	✓
2	Xử lý các nhóm phần tử (Group)	✓
3	Xử lý các phần tử (path)	✓

## 6 Results

Qua quá trình phát triển dự án, nhóm đã hoàn thành các tính năng sau đây trong milestone thứ hai. Dưới đây là kết quả render của 18 test cases khác nhau, thể hiện khả năng của ứng dụng trong việc xử lý các file SVG với độ phức tạp và cấu trúc khác nhau.

Xem thêm tại đây: [Google Drive](#)

## 7 Conclusion

Qua quá trình phát triển milestone thứ hai, nhóm đã thành công trong việc mở rộng ứng dụng NaTruKi SVG Viewer với hai tính năng quan trọng là SVGPath và SVGGroup. Việc bổ sung này không chỉ nâng cao khả năng hiển thị các file SVG phức tạp mà còn thể hiện được tính hiệu quả của kiến trúc phần mềm được xây dựng từ milestone đầu tiên. Nhờ áp dụng các design pattern như Visitor, Factory và Composite ngay từ ban đầu, quá trình thêm tính năng mới diễn ra thuận lợi mà không cần thay đổi code cũ, tuân thủ đúng nguyên tắc Open-Closed Principle.

SVGPath là phần tử phức tạp nhất được thêm vào, yêu cầu xử lý nhiều loại lệnh khác nhau với cả tọa độ tuyệt đối và tương đối. Thuật toán phân tích cú pháp dựa trên Finite State Machine đã được triển khai thành công để parse chuỗi path data, xử lý các trường hợp đặc biệt như implicit lineto sau moveto, và chuyển đổi sang QPainterPath để rendering. Đồng thời, SVGGroup với Composite Pattern cho phép nhóm các phần tử lại với nhau và áp dụng transform kế thừa, tạo nên cấu trúc cây phân cấp giống như DOM tree của SVG thực tế. Việc kết hợp giữa đệ quy và



Visitor Pattern trong xử lý group đảm bảo transform được áp dụng đúng cách cho toàn bộ cây con.

Trong quá trình phát triển, nhóm đã học được nhiều bài học quý giá về thiết kế phần mềm mở rộng được. Việc planning kỹ càng về kiến trúc từ đầu giúp tiết kiệm rất nhiều thời gian khi cần thêm tính năng mới. Test với các file SVG thực tế từ nhiều nguồn khác nhau giúp phát hiện ra các edge case quan trọng như lowercase commands, nested groups, hoặc transform phức tạp. Ngoài ra, việc document code chi tiết và maintain git history rõ ràng cũng góp phần quan trọng vào sự thành công của dự án, đặc biệt khi làm việc nhóm với nhiều thành viên cùng đóng góp vào các phần khác nhau.

## Tài liệu tham khảo

- [1] Wikipedia contributors, “Finite-state machine — Wikipedia, the free encyclopedia,” 2025. [Online; accessed 16-December-2025].
- [2] MDN Web Docs, “<svg>: The SVG container element.” <https://developer.mozilla.org/en-US/docs/Web/SVG/Reference/Element/svg>, Oct. 2025. [Online; accessed 16-December-2025].