**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

---🙢🕮🙠---

A blue and white logo with a red star

Description automatically generated

**BÁO CÁO THỰC TẬP CƠ SỞ (CÔNG NGHỆ THÔNG TIN)**

MÔ PHỎNG QUÁ TRÌNH BIẾN ĐỔI ĐỐI TƯỢNG TRONG KHÔNG GIAN 2D BẰNG CÔNG CỤ ĐỒ HOẠ GRAPHICS.H

**Giảng viên hướng dẫn: ThS. Đoàn Vũ Thịnh**

**Sinh viên thực hiện: Võ Thùy Linh**

**Mã số sinh viên: 63132961**

**Lớp: 63.CNTT-4**

**Khánh Hoà – tháng 1/2023**

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**Khoa: Công Nghệ Thông Tin**

**PHIẾU THEO DÕI TIẾN ĐỘ VÀ ĐÁNH GIÁ BÁO CÁO THỰC TẬP CƠ SỞ**

**Tên đề tài:** Mô phỏng quá trình biến đổi đối tượng trong không gian 2D bằng công cụ đồ hoạ Graphics.h

**Giáo viên hướng dẫn:** ThS. Đoàn Vũ Thịnh

**Sinh viên được hướng dẫn:** Võ Thùy Linh

**MSSV:** 63132961

**Khoá**: 63 **Ngành:** Công nghệ thông tin

|  |  |  |  |
| --- | --- | --- | --- |
| **Lần** | **Ngày** | **Nội dung** | **Nhận xét của GVHD** |
| 1 | 27/11/2023 | Đề tài: Sử dụng công cụ đồ hoạ graphics.h để mô phỏng biến đổi đối tượng trong không gian 2D  Lên kế hoạch thực hiện  Lựa chọn giải pháp để cài đặt |  |
| 2 | 30/11/2023 | Làm giao diện chính  Làm thuật toán mô phỏng biến đổi đối tượng trong không gian 2D: translation, rotation, scaling  Thuật toán đọc file, nhập từ phím, reset, close |  |
| 3 | 7/12/2023 | Làm thuật toán translation, rotation cho đa giác |  |
| 4 | 12/12/2023 | Thuật toán vẽ hình tròn |  |
| 5 | 16/12/2023 | Làm thuật toán translation, rotation và scaling cho hình tròn và đa giác |  |
| 6 | 18/12/2023 | Chỉnh sửa hoàn chỉnh báo cáo |  |
| 7 | 1/1/2024 | Sinh viên nộp bản thảo lần cuối sau khi đã chỉnh sửa các yêu cầu như đã đề ra. |  |

**MỤC LỤC**

[LỜI CẢM ƠN v](#_Toc155683708)

[TÓM TẮT vi](#_Toc155683709)

[CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI 7](#_Toc155683710)

[1.1. Dev c++ 7](#_Toc155683711)

[1.2. Thư viện đồ họa graphics.h 7](#_Toc155683712)

[1.3. Khái niệm phép biến đổi Affine: 8](#_Toc155683713)

[Một số phép biến đổi cơ bản 8](#_Toc155683714)

[1.4. Phép tịnh tiến (Translation) 9](#_Toc155683715)

[1.4.1. Giới thiệu về phép tịnh tiến 9](#_Toc155683716)

[1.4.2. Ý tưởng 9](#_Toc155683717)

[1.4.3. Ví dụ minh họa 10](#_Toc155683718)

[1.5. Phép quay góc tọa độ (Rotation) 10](#_Toc155683719)

[1.5.1. Khái niệm về phép quay góc tọa độ 10](#_Toc155683720)

[1.5.2. Ý tưởng 10](#_Toc155683721)

[1.5.3. Ví dụ minh họa 11](#_Toc155683722)

[1.6. Phép biến biến đổi tỉ lệ (Scaling) 11](#_Toc155683723)

[1.6.1. Khái niệm phép biến đổi tỉ lệ 11](#_Toc155683724)

[1.6.2. Ý tưởng 12](#_Toc155683725)

[1.6.3. Ví dụ minh họa 13](#_Toc155683726)

[CHƯƠNG 2. PHƯƠNG PHÁP NGHIÊN CỨU 14](#_Toc155683727)

[2.1. Phần chung của sản phẩm 14](#_Toc155683728)

[2.1.1. Khai báo thư viện và biến toàn cục 14](#_Toc155683729)

[2.1.2. Hàm chương trình chính 14](#_Toc155683730)

[2.2. Tổng quan về thiết kế giao diện 15](#_Toc155683731)

[2.2.1. Sơ lược về giao diện sản phẩm 15](#_Toc155683732)

[2.2.2. Xây dựng giao diện 15](#_Toc155683733)

[2.3. Xử lý sự kiện chuột 22](#_Toc155683734)

[2.3.1. Kiểm tra khi nhấn nút Load file 24](#_Toc155683735)

[2.3.2. Kiểm tra khi nhấn nút Keyboard 25](#_Toc155683736)

[2.2.3. Kiểm tra khi nhấn nút Translation 25](#_Toc155683737)

[2.2.3. Kiểm tra khi nhấn nút Rotation 25](#_Toc155683738)

[2.2.3. Kiểm tra khi nhấn nút Scaling 25](#_Toc155683739)

[2.2.3. Kiểm tra khi nhấn nút Reset 25](#_Toc155683740)

[2.2.3. Kiểm tra khi nhấn nút Close 26](#_Toc155683741)

[2.4. PHÂN TÍCH THUẬT TOÁN 26](#_Toc155683742)

[2.4.1. Thuật toán vẽ đa giác (ART\_cool) 26](#_Toc155683743)

[2.4.2. Thuật toán vẽ hình tròn (circle) 26](#_Toc155683744)

[2.4.3. LOAD FILE 28](#_Toc155683745)

[2.4.3. Nhập từ bàn phím 30](#_Toc155683746)

[2.4.4. Phép tịnh tiến (Translation) 31](#_Toc155683747)

[2.4.5. Phép quay góc tọa độ (Rotation) 32](#_Toc155683748)

[2.4.6. Phép biến đổi tỉ lệ (Scaling) 33](#_Toc155683749)

[CHƯƠNG 3. KẾT QUẢ 35](#_Toc155683750)

[CHƯƠNG 4. TỔNG KẾT 37](#_Toc155683751)

[4.1. Ưu điểm 37](#_Toc155683752)

[4.2. Nhược điểm 37](#_Toc155683753)

[Tài liệu tham khảo: 38](#_Toc155683754)

# **LỜI CẢM ƠN**

Để có thể hoàn thành đợt thực tập lần này, em xin chân thành cảm ơn đến quý thầy cô khoa Công nghệ Thông tin đã tạo điều kiện hỗ trợ và giúp đỡ em trong quá trính học tập và thực hiện đề tài này.

Qua đây, em xin chân thành cảm ơn thầy Đoàn Vũ Thịnh, người đã trực tiếp quan tâm và hướng dẫn em hoàn thành tốt đợt thực tập trong thời gian qua.

Do kiến thức còn hạn chế và thời gian thực hiện còn ngắn nên bài báo cáo của em còn nhiều thiếu sót, kính mong sự góp ý của quý thầy cô.

Em xin chân thành cảm ơn!

# **TÓM TẮT**

Đồ hoạ máy tính là một lĩnh vực quan trọng trong công nghệ thông tin, nơi các hình ảnh và đối tượng được tạo ra và hiển thị trực quan trên màn hình máy tính. Đồ hoạ máy tính không chỉ giúp làm đẹp giao diện người dùng mà còn có ứng dụng rộng rãi trong mô phỏng, giáo dục, và trực quan hóa dữ liệu.

Phép biến đổi hình học là một phép trạng thái tranh chấp trong đó không gian ban đầu của một đối tượng được biến đổi để tạo ra một hình dạng mới. [Phép biến đổi hình học có thể làm thay đổi vị trí, hình dạng, cỡ lớn, hoặc hướng của đối tượng](http://tailieuso.udn.vn/bitstream/TTHL_125/6396/1/TruongThiNga.TT.pdf). [Các phép biến đổi hình học sơ cấp bao gồm: phép tịnh tiến, phép đối xứng, phép quay và phép tỉ lệ](http://tailieuso.udn.vn/bitstream/TTHL_125/6396/1/TruongThiNga.TT.pdf)[. Các phép biến đổi hình học được ứng dụng rộng rãi trong giải toán hình học](https://dinhnghia.vn/chuyen-de-cac-phep-bien-hinh.html).

Khi kết hợp thuật toán biến đổi đối tượng với công cụ đồ hoạ máy tính như Graphics.h trong ngôn ngữ lập trình C++, chúng ta có cơ hội tạo ra một trải nghiệm học tập và hiểu biết động hấp dẫn. Việc mô phỏng quá trình biến đổi các đa giác thông qua đồ hoạ máy tính không chỉ giúp sinh viên thấy rõ từng bước của thuật toán mà còn giúp họ hình dung và theo dõi quá trình này một cách trực quan.

Sản phẩm đã minh họa được quá trình mô phỏng được thực hiện thông qua việc trực quan hóa các bước của thuật toán tịnh tiến (Translation), quay (Rotation), co dãn (Scaling). Các bước bao gồm di chuyển các hình từ vị trí này sang vị trí khác, quay từ vị trí ban đầu đến góc mong muốn và co dãn hình . Sản phẩm thực hiện tốt với nhiều cách đưa dữ liệu đầu vào như đọc file, nhập từ bàn phím hoặc tạo nhanh ngẫu nhiên,… và có thể sắp xếp được dựa trên các dữ liệu đầu vào đó.

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1. Dev c++

Dev C++ là một môi trường phát triển tích hợp (IDE) phổ biến được sử dụng cho việc lập trình bằng ngôn ngữ lập trình C và C++. Được phát triển bởi Bloodshed Software, Dev C++ cung cấp một giao diện đơn giản và thuận tiện, làm cho quá trình lập trình trở nên dễ dàng đối với cả người mới bắt đầu và những người có kinh nghiệm.

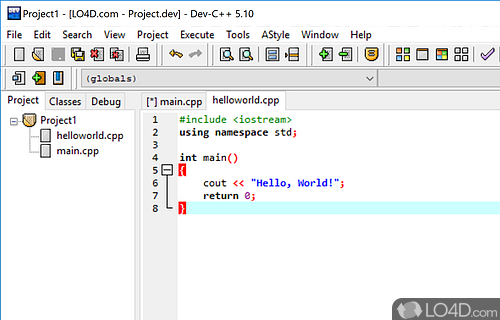
Dev-C++ được đóng gói và hỗ trợ nhiều trình biên dịch C++ phổ biến như MinGW hoặc TDM-GCC 64bit của GCC (GNU Compoler Collection). Dev-C++ cũng có thể được sử dụng kết hợp với Cygwin hoặc bất kỳ trình biên dịch dựa trên GCC nào khác.

Figure 1. Giới thiệu về Dev C++

Giao diện của Dev C++ đơn giản và thân thiện với người dùng, điều này làm cho quá trình lập trình trở nên trực quan và dễ tiếp cận. Ngoài ra, IDE này hỗ trợ nhiều tính năng như quản lý dự án, trình soạn thảo mã nguồn tùy chỉnh, và khả năng tạo và chạy chương trình một cách thuận lợi.

Dự án phát triển Dev-C++ được lưu trữ trên SourceForge. Dev-C++ nguyên được phát triển bởi một lập trình viên có tên là Colin Laplace và chỉ chạy trên hệ điều hành Microsoft Windows.

## **1.2. Thư viện đồ họa graphics.h**

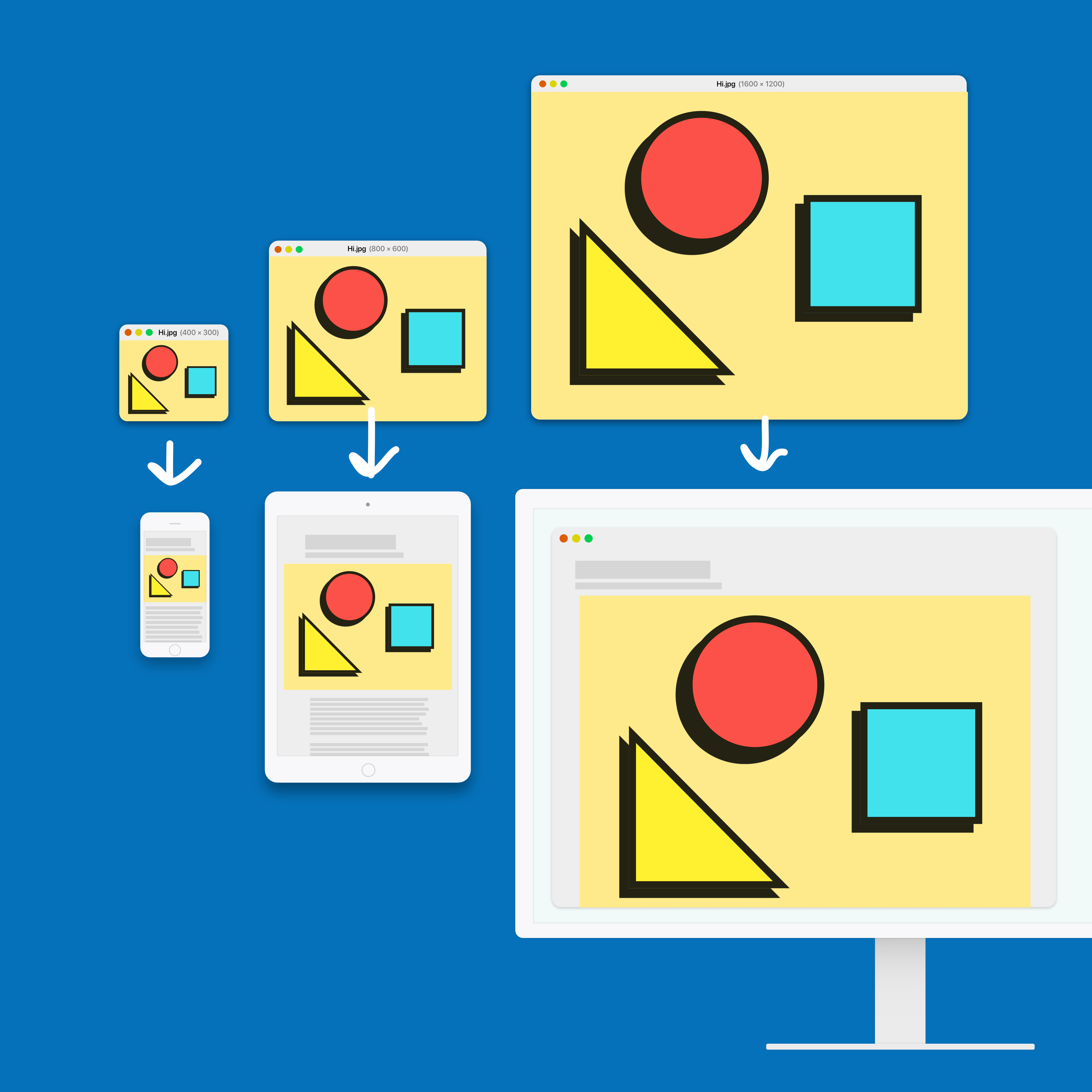
Thư viện đồ hoạ graphics.h là một thư viện phổ biến được sử dụng trong ngôn ngữ lập trình C và C++ để tạo và quản lý đồ hoạ trên màn hình máy tính. Thư viện này thường được sử dụng để vẽ đồ họa cơ bản, tạo ra giao diện người dùng đơn giản, và mô phỏng các hình ảnh đồ họa.

Figure 2. Các đối tượng hình học trong Graphics.h

Thư viện graphics.h cung cấp một loạt các hàm và công cụ để điều khiển đồ hoạ, bao gồm việc vẽ các đối tượng hình học như đường thẳng, hình chữ nhật, hình tròn, v.v. Thư viện này cũng hỗ trợ việc thao tác màu sắc, font chữ và các hiệu ứng đơn giản khác.

## **1.3. Khái niệm phép biến đổi Affine:**

Phép biến đổi: là một hàm ánh xạ một điểm hay một vector thành một điểm hoặc một vector khác.

Phép biến đổi affine là phép biến đổi tuyến tính thỏa mãn: f(αp+βq)=αf(p)+βf(q)

f(αp+βq)=αf(p)+βf(q)

Trong đó: p, q là điểm hoặc vector; α, β là các hằng số.

### **Một số phép biến đổi cơ bản**

* Phép tịnh tiến (Translation): là phép dịch chuyển khi thực hiện phép biến đổi tịnh tiến, một hình học được di chuyển từ vị trí ban đầu sang một vị trí mới trong không gian theo một hướng và một khoảng cách xác định.
* Phép quay tại góc tọa độ (Rotation): là một phép biến đổi mang hình học từ vị trí ban đầu và xoay nó quanh một điểm gốc theo một góc quay xác định.
* Phép biến đổi tỉ lệ (Scaling): khi thực hiện phép biến đổi tỉ lệ, kích thước của hình học được thay đổi theo một tỷ lệ xác định, bằng cách nhân hoặc chia các tọa độ của điểm trong hình học với một hệ số tỉ lệ.

## **1.4. Phép tịnh tiến (Translation)**

### **1.4.1. Giới thiệu về phép tịnh tiến**

Phép biến đổi tịnh tiến là một phép biến đổi trong hình học không gian hai chiều. Nó còn được gọi là phép dịch chuyển hay phép dịch. Khi thực hiện phép biến đổi tịnh tiến, một hình học được di chuyển từ vị trí ban đầu sang một vị trí mới trong không gian theo một hướng và một khoảng cách xác định.

Phép biến đổi tịnh tiến được thực hiện bằng cách cộng tọa độ của mỗi điểm trong hình học với một vector tịnh tiến (dx, dy), trong đó dx là khoảng cách di chuyển theo trục x và dy là khoảng cách di chuyển theo trục y. Kết quả là mọi điểm trong hình học được dịch chuyển một cách đồng nhất theo cùng một hướng và khoảng cách.

### **1.4.2. Ý tưởng**

* Xác định vector tịnh tiến: Đầu tiên, xác định vector tịnh tiến (dx, dy) để chỉ định khoảng cách và hướng di chuyển của đối tượng. Vector tịnh tiến này sẽ quyết định vị trí mới của đối tượng sau khi tịnh tiến.
  + Áp dụng vector tịnh tiến cho mỗi điểm: Tiếp theo, áp dụng vector tịnh tiến cho mỗi điểm trong đối tượng. Để làm điều này, hãy thay đổi tọa độ của mỗi điểm bằng cách cộng vector tịnh tiến vào tọa độ hiện tại của nó. Công thức sẽ là:

x'=x+dx  
y' = y + dy

Trong đó, (x, y) là tọa độ ban đầu của điểm và (x', y') là tọa độ mới của điểm sau khi tịnh tiến.

* + Thực hiện tất cả các điểm: Áp dụng bước 2 cho tất cả các điểm trong đối tượng, bao gồm các đỉnh, điểm trung tâm, hoặc bất kỳ điểm nào khác mà bạn muốn tịnh tiến.
  + Trực quan hóa kết quả: Vẽ đối tượng mới sau khi tịnh tiến bằng cách sử dụng các tọa độ mới của các điểm đã tính toán trong bước 2 và 3. Điều này sẽ hiển thị đối tượng ở vị trí mới sau khi tịnh tiến.

### **A diagram of a rectangular object with black text Description automatically generated1.4.3. Ví dụ minh họa**

Figure 3. Hình ảnh minh họa phép tịnh tiến

## **1.5. Phép quay góc tọa độ (Rotation)**

### **1.5.1. Khái niệm về phép quay góc tọa độ**

Trong hình học, phép biến đổi quay là một phép biến đổi mang hình học từ vị trí ban đầu và xoay nó quanh một điểm gốc theo một góc quay xác định.

Khi thực hiện phép biến đổi quay, một điểm trong không gian được xoay quanh một điểm gốc theo một góc quay θ. Các điểm khác trong hình học di chuyển theo quỹ đạo tạo bởi quỹ đạo của điểm ban đầu sau khi qua phép biến đổi quay.

Phép biến đổi quay thường được mô tả bằng cách sử dụng tọa độ trực gốc. Để xoay một điểm (x, y) quanh một điểm gốc (a, b) theo góc quay θ.

### **1.5.2. Ý tưởng**

* Xác định điểm gốc: Đầu tiên, xác định điểm gốc (a, b) là điểm mà đối tượng sẽ xoay quanh. Điểm này có thể là trung tâm của đối tượng hoặc bất kỳ điểm nào khác mà bạn muốn sử dụng làm trung tâm quay.
* Xác định góc quay: Xác định góc quay θ (theo radian hoặc độ) để quyết định mức độ và hướng xoay của đối tượng. Điều này sẽ xác định tốc độ và chiều quay của đối tượng.
* Áp dụng công thức quay: Áp dụng công thức quay cho mỗi điểm trong đối tượng để tính toán tọa độ mới của chúng sau phép biến đổi quay.
* Công thức quay dựa trên các phép biến đổi hình học và sử dụng các hàm toán học như sin và cos. Đối với mỗi điểm (x, y) trong đối tượng, công thức quay sẽ là:

x' = (x - a) \* cos(θ) - (y - b) \* sin(θ) + a  
y' = (x - a) \* sin(θ) + (y - b) \* cos(θ) + b

Trong đó, (x', y') là tọa độ mới của điểm sau phép biến đổi quay, (x, y) là tọa độ ban đầu của điểm, và (a, b) là tọa độ điểm gốc.

* Thực hiện tất cả các điểm: Áp dụng bước 3 cho tất cả các điểm trong đối tượng, bao gồm các đỉnh, điểm trung tâm, hoặc bất kỳ điểm nào khác mà bạn muốn xoay.
* Trực quan hóa kết quả: Vẽ đối tượng mới sau phép biến đổi quay bằng cách sử dụng các tọa độ mới của các điểm đã tính toán trong bước 3 và 4. Điều này sẽ hiển thị đối tượng ở vị trí mới sau phép biến đổi quay.

### **1.5.3. Ví dụ minh họa**

## **A diagram of a function Description automatically generated1.6. Phép biến biến đổi tỉ lệ (Scaling)**

Figure 4. Hình ảnh minh họa phép quay góc tọa độ

### **1.6.1. Khái niệm phép biến đổi tỉ lệ**

Trong hình học, phép biến đổi tỉ lệ (scaling) là một phép biến đổi thay đổi kích thước của một hình học. Khi thực hiện phép biến đổi tỉ lệ, kích thước của hình học được thay đổi theo một tỷ lệ xác định, bằng cách nhân hoặc chia các tọa độ của điểm trong hình học với một hệ số tỉ lệ.

Phép biến đổi tỉ lệ có thể thực hiện tỉ lệ lớn hơn 1 để tăng kích thước của hình học (phóng to), hoặc tỉ lệ nhỏ hơn 1 để giảm kích thước của hình học (co lại). Nếu tỷ lệ là âm, phép biến đổi tỉ lệ sẽ đồng thời thay đổi kích thước và hướng của hình học, tạo ra hiện tượng đối xứng.

Phép biến đổi tỉ lệ được mô tả bằng cách nhân tọa độ của mỗi điểm trong hình học với một hệ số tỉ lệ.

### **1.6.2. Ý tưởng**

* Xác định điểm gốc: Đầu tiên, xác định điểm gốc (a, b) là điểm mà đối tượng sẽ được tỉ lệ. Điểm này có thể là trung tâm của đối tượng hoặc bất kỳ điểm nào khác mà bạn muốn sử dụng làm trung tâm tỉ lệ.
* Xác định tỷ lệ: Xác định hệ số tỉ lệ (s\_x, s\_y) để quyết định mức độ mở rộng hoặc co lại của đối tượng theo các hướng x và y tương ứng. Hệ số tỉ lệ có thể là một số dương để phóng to hoặc một số nhỏ hơn 1 để co lại.
* Áp dụng công thức tỉ lệ: Áp dụng công thức tỉ lệ cho mỗi điểm trong đối tượng để tính toán tọa độ mới của chúng sau phép biến đổi tỉ lệ.

Công thức tỉ lệ sẽ là:

x' = (x - a) \* s\_x + a  
y' = (y - b) \* s\_y + b

Trong đó, (x', y') là tọa độ mới của điểm sau phép biến đổi tỉ lệ, (x, y) là tọa độ ban đầu của điểm, và (a, b) là tọa độ điểm gốc.

* Thực hiện tất cả các điểm: Áp dụng bước 3 cho tất cả các điểm trong đối tượng, bao gồm các đỉnh, điểm trung tâm, hoặc bất kỳ điểm nào khác mà bạn muốn tỉ lệ.
* Trực quan hóa kết quả: Vẽ đối tượng mới sau phép biến đổi tỉ lệ bằng cách sử dụng các tọa độ mới của các điểm đã tính toán trong bước 3 và 4. Điều này sẽ hiển thị đối tượng ở kích thước mới sau phép biến đổi tỉ lệ.

### **A graph of a graph with a graph and arrows Description automatically generated with medium confidence1.6.3. Ví dụ minh họa**

Figure 5. Hình ảnh minh họa phép biến đổi tỉ lệ

# **CHƯƠNG 2. PHƯƠNG PHÁP NGHIÊN CỨU**

## **2.1. Phần chung của sản phẩm**

### **2.1.1. Khai báo thư viện và biến toàn cục**

A screenshot of a computer program

Description automatically generated

Figure 6. Các thư viện và biến toàn cục sử dụng

### **2.1.2. Hàm chương trình chính**

A screenshot of a computer code

Description automatically generated

Figure 7. Hàm chương trình chính main()

## **2.2. Tổng quan về thiết kế giao diện**

### **2.2.1. Sơ lược về giao diện sản phẩm**

A screenshot of a computer program

Description automatically generatedGiao diện chính được mô tả các thông tin như: Họ tên sinh viên thực tập, họ tên giáo viên hướng dẫn, lớp; Khung lớn để hiển thị các đa giác trước và sau khi thực hiện các chức năng, các nút (button) thể hiện từng chức năng riêng ở bên cạnh khung lớn gồm có: nút lấy dữ liệu từ việc đọc file (Load file) để tạo các đa giác, nhập dữ liệu từ bàn phím (Keyboard), phép tịnh tiến (Translation), phép quay góc tọa độ (Rotation), phép biến đổi tỉ lên (Scaling) trong không gian 2 chiều (2D), thoát khỏi cửa sổ đồ hoạ (Close) và nút Reset lại chương trình.

Figure 8. Giao diện chính màn hình thực hiện

### **2.2.2. Xây dựng giao diện**

**2.2.2.1. Khởi tạo cửa sổ đồ họa**

Tại hàm chính main(), ta sử dụng hàm *initwindow(width, height)* với width = 1000 và height = 800 được sử dụng để khởi tạo một cửa sổ đồ họa có độ rộng là 1000 và chiều cao là 800 trên màn hình do thư viện đồ hoạ graphics.h cung cấp.

A screenshot of a computer program

Description automatically generated

**1000**

**800**

Figure 9. Kích thước cửa sổ đồ họa

**2.2.2.2. Các thực thể trong cửa sổ đồ họa**

Tại hàm main(), ta gọi hàm ART\_init() để hiển thị các thực thể (khung, nút và các thông tin SV) lên cửa sổ đồ hoạ.

Toàn bộ nội dung của hàm *ART\_init()* như sau:

A screenshot of a computer program

Description automatically generated

Figure 10. Hình ảnh hàm ART\_init

**2.2.2.2.1. Xây dựng khung cố định để hiển thị phép biến đổi**

Để vẽ khung, ta sẽ cần sử dụng các hàm do thư viện *graphics.h* cung cấp như sau:

* *setcolor(YELLOW):* Thiết lập viền khung màu vàng
* *setlinestyle(0, 1, 3)*: Thiết lập đường vẽ có tham số thứ 1 *linestyle* = 0 là kiểu đường chấm gạch ngang. Tham số thứ 2 *upattern* = 1 là mẫu nhị phân (tạo ra đường chấm) và tham số thứ 3 *thickness* = 3 để thể hiện độ dày của nét vẽ.
* *rectangle(10, 10, 750, 650)*: Vẽ một hình chữ nhật có góc trái trên cùng có toạ độ (10, 10) và góc phải dưới cùng có toạ độ (750, 650). Kích thước của hình chữ nhật là 740 px (750 - 10) theo chiều ngang và 640 px (650 - 10) theo chiều dọc.

A screenshot of a computer program

Description automatically generated

**650**

**750**

Figure 11. Kích thước của khung

**A screenshot of a computer program

Description automatically generated2.2.2.2.2. Xây dựng các nút cơ bản**

Figure 12. Các nút thực hiện

* **Nút Load file**
* *setcolor(GREEN)*: Định dạng viền nút màu xanh cố định
* *rectangle(790,10,980,50)*: Vẽ hình chữ nhật có góc trái trên cùng có toạ độ (790, 10) và góc phải dưới cùng có toạ độ (980, 50). Kích thước của hình chữ nhật này là 190 px (980 - 790) theo chiều ngang và 40 px (50 - 10) theo chiều dọc.
* *setcolor(WHITE)*: Định dạng màu chữ cố định là trắng
* *settextstyle(2,0,8)*: Được sử dụng để đặt kiểu văn bản cho văn bản được xuất ra trên màn hình; số 2 dùng để xác định kiểu chữ có sẵn, số 0 là đại diện cho đường viền văn bản là không, số 8 là định dạng kích thước lớn nhỏ cho văn bản.
* *outtextxy(800,20,"Load file")*: Hiển thị chuỗi văn bản “Load file” bắt đầu từ vị trí có toạ độ là (800, 20) trên cửa sổ giao diện.
* **Nút Keyboard**
* *setcolor(GREEN):* Định dạng viền nút màu xanh cố định
* *rectangle(790,110,980,150):* Vẽ hình chữ nhật có góc trái trên cùng có toạ độ (790, 110) và góc phải dưới cùng có toạ độ (980, 50). Kích thước của hình chữ nhật này là 190 px (980 - 790) theo chiều ngang và 40 px (150 - 110) theo chiều dọc.
* *settextstyle(2,0,8)*: Được sử dụng để đặt kiểu văn bản cho văn bản được xuất ra trên màn hình; số 2 dùng để xác định kiểu chữ có sẵn, số 0 là đại diện cho đường viền văn bản là không, số 8 là định dạng kích thước lớn nhỏ cho văn bản.
* *setcolor(WHITE)*: Định dạng màu chữ cố định là trắng.
* *outtextxy(800,120,"Keyboard")*: Hiển thị chuỗi văn bản “Keyboard” bắt đầu từ vị trí có toạ độ là (800, 120) trên cửa sổ giao diện.
* **Nút Translation**
* *setcolor(GREEN):* Định dạng viền nút màu xanh cố định
* *rectangle(790,210,980,250):* Vẽ hình chữ nhật có góc trái trên cùng có toạ độ (790, 210) và góc phải dưới cùng có toạ độ (980, 250). Kích thước của hình chữ nhật này là 190 px (980 - 790) theo chiều ngang và 40 px (250 - 210) theo chiều dọc.
* *settextstyle(2,0,8)*: Được sử dụng để đặt kiểu văn bản cho văn bản được xuất ra trên màn hình; số 2 dùng để xác định kiểu chữ có sẵn, số 0 là đại diện cho đường viền văn bản là không, số 8 là định dạng kích thước lớn nhỏ cho văn bản.
* *setcolor(WHITE)*: Định dạng màu chữ cố định là trắng.
* *outtextxy(800,220,"Translation")*: Hiển thị chuỗi văn bản “Translation” bắt đầu từ vị trí có toạ độ là (800, 220) trên cửa sổ giao diện.
* **Nút Rotation**
* *setcolor(GREEN):* Định dạng viền nút màu xanh cố định
* *rectangle(790,310,980,350):* Vẽ hình chữ nhật có góc trái trên cùng có toạ độ (790, 310) và góc phải dưới cùng có toạ độ (980, 350). Kích thước của hình chữ nhật này là 190 px (980 - 790) theo chiều ngang và 40 px (350 - 310) theo chiều dọc.
* *settextstyle(2,0,8)*: Được sử dụng để đặt kiểu văn bản cho văn bản được xuất ra trên màn hình; số 2 dùng để xác định kiểu chữ có sẵn, số 0 là đại diện cho đường viền văn bản là không, số 8 là định dạng kích thước lớn nhỏ cho văn bản.
* *setcolor(WHITE)*: Định dạng màu chữ cố định là trắng.
* *outtextxy(800,320,"Rotation")*: Hiển thị chuỗi văn bản “Rotation” bắt đầu từ vị trí có toạ độ là (800, 320) trên cửa sổ giao diện.
* **Nút Scaling**
* *setcolor(GREEN):* Định dạng viền nút màu xanh cố định
* *rectangle(790,410,980,450):* Vẽ hình chữ nhật có góc trái trên cùng có toạ độ (790, 410) và góc phải dưới cùng có toạ độ (980,450). Kích thước của hình chữ nhật này là 190 px (980 - 790) theo chiều ngang và 40 px (450 - 410) theo chiều dọc.
* *settextstyle(2,0,8)*: Được sử dụng để đặt kiểu văn bản cho văn bản được xuất ra trên màn hình; số 2 dùng để xác định kiểu chữ có sẵn, số 0 là đại diện cho đường viền văn bản là không, số 8 là định dạng kích thước lớn nhỏ cho văn bản.
* *setcolor(WHITE)*: Định dạng màu chữ cố định là trắng.
* *outtextxy(800,420,"Scaling")*: Hiển thị chuỗi văn bản “Scaling” bắt đầu từ vị trí có toạ độ là (800, 420) trên cửa sổ giao diện.
* **Nút Reset**
* *setcolor(GREEN):* Định dạng viền nút màu xanh cố định
* *rectangle(790,510,980,550):* Vẽ hình chữ nhật có góc trái trên cùng có toạ độ (790, 510) và góc phải dưới cùng có toạ độ (980, 550). Kích thước của hình chữ nhật này là 190 px (980 - 790) theo chiều ngang và 40 px (550 - 510) theo chiều dọc.
* *settextstyle(2,0,8)*: Được sử dụng để đặt kiểu văn bản cho văn bản được xuất ra trên màn hình; số 2 dùng để xác định kiểu chữ có sẵn, số 0 là đại diện cho đường viền văn bản là không, số 8 là định dạng kích thước lớn nhỏ cho văn bản.
* *setcolor(WHITE)*: Định dạng màu chữ cố định là trắng.
* *outtextxy(800,520,"Reset")*: Hiển thị chuỗi văn bản “Reset” bắt đầu từ vị trí có toạ độ là (800, 520) trên cửa sổ giao diện.
* **Nút Close**
* *setcolor(GREEN):* Định dạng viền nút màu xanh cố định
* *rectangle(790,610,980,650):* Vẽ hình chữ nhật có góc trái trên cùng có toạ độ (790, 610) và góc phải dưới cùng có toạ độ (980, 650). Kích thước của hình chữ nhật này là 190 px (980 - 790) theo chiều ngang và 40 px (650 - 610) theo chiều dọc.
* *settextstyle(2,0,8)*: Được sử dụng để đặt kiểu văn bản cho văn bản được xuất ra trên màn hình; số 2 dùng để xác định kiểu chữ có sẵn, số 0 là đại diện cho đường viền văn bản là không, số 8 là định dạng kích thước lớn nhỏ cho văn bản.
* *setcolor(WHITE)*: Định dạng màu chữ cố định là trắng.
* *outtextxy(800,620,"Close")*: Hiển thị chuỗi văn bản “Close” bắt đầu từ vị trí có toạ độ là (800, 620) trên cửa sổ giao diện.

**A black screen with white text

Description automatically generated2.2.2.2.3. Thiết kế thông tin người làm (Sinh viên thực hiện)**

Figure 13. Thông tin sinh viên thực hiện

* *settextstyle(2,0,7)*: Được sử dụng để đặt kiểu văn bản cho văn bản được xuất ra trên màn hình; số 2 dùng để xác định kiểu chữ có sẵn, số 0 là đại diện cho đường viền văn bản là không, số 7 là định dạng kích thước lớn nhỏ cho văn bản.
* *outtextxy(10,670,"GVHD : DOAN VU THINH")*: Hiển thị chuỗi văn bản “GVHD : DOAN VU THINH” bắt đầu từ vị trí có toạ độ là (10, 670) trên cửa sổ giao diện.
* *outtextxy(10,700,"* *SVTH : VO THUY LINH")*: Hiển thị chuỗi văn bản “SVTH : VO THUY LINH” bắt đầu từ vị trí có toạ độ là (10, 700) trên cửa sổ giao diện.
* *outtextxy(10,730,"* *LOP : 63.CNTT-4")*: Hiển thị chuỗi văn bản “LOP : 63.CNTT-4” bắt đầu từ vị trí có toạ độ là (10, 700) trên cửa sổ giao diện.

## **2.3. Xử lý sự kiện chuột**

* WM\_MOUSEMOVE: phát hiện chuyển động của chuột
* WM\_LBUTTONDBCLK: phát hiện khi nút chuột trái được nhấp đúp
* WM\_LBUTTONDOWN: phát hiện khi nút chuột trái được nhấn xuống
* WM\_LBUTTONUP: phát hiện khi nút chuột trái được thả lên
* WM\_MBUTTONDBCLK: phát hiện khi nút chuột giữa được nhấp đúp
* WM\_MBUTTONDOWN: phát hiện khi nút chuột giữa được nhấp xuống
* WM\_MBUTTONUP: phát hiện khi nút chuột giữa được thả ra
* WM\_RBUTTONDBCLK: phát hiện khi nút chuột phải được nhấp đúp
* WM\_RBUTTONDOWN: phát hiện khi nút chuột phải được nhấn xuống
* WM\_RBUTTONUP: phát hiện khi nút chuột phải được thả lêN
* Khai báo x\_mouse và y\_mouse để lưu trữ tọa độ của click chuột.
* Khai báo ismouseclick và iskeyboardclick được sử dụng để theo dõi trạng thái của các hoạt động liên quan đến hình dạng và chuột.
* Vòng lặp vô hạn (while (1)) để liên tục kiểm tra sự kiện click chuột.
* Trong vòng lặp, mã kiểm tra xem có sự kiện click chuột trái (ismouseclick(WM\_LBUTTONDOWN)) hay không. Nếu có, nó sử dụng getmouseclick(WM\_LBUTTONDOWN, x\_mouse, y\_mouse) để lấy tọa độ x và y của click chuột.
* Tiếp tục kiểm tra các vị trí click chuột để xác định hành động tương ứng

A computer screen shot of a computer program

Description automatically generated

Figure 14. Trình tự xử lý sự kiện chuột

### **A screenshot of a computer program Description automatically generated2.3.1. Kiểm tra khi nhấn nút Load file**

Figure 15. Nội dung nơi xử lý Load file

* Điều kiện nếu *x\_mouse > 790 && x\_mouse < 980 && y\_mouse > 10 && y\_mouse < 50*: Đây là khu vực của nút “Load File” nhằm kiểm tra xem sự kiện click chuột trái có xảy ra trong một khu vực này của màn hình không.
* Biến “chose” là lựa chọn file muốn đọc để hiển thị trong 3 file có sẵn

### **2.3.2. Kiểm tra khi nhấn nút Keyboard**

Figure 16. Xử lý Keyboard

### **2.2.3. Kiểm tra khi nhấn nút Translation**

Figure 17. Xử lý Translation

### **2.2.3. Kiểm tra khi nhấn nút Rotation**

Figure 18. Xử lý Rotation

### **2.2.3. Kiểm tra khi nhấn nút Scaling**

Figure 19. Xử lý Scaling

### **2.2.3. Kiểm tra khi nhấn nút Reset**

Figure 20. Xử lý Reset

### **2.2.3. Kiểm tra khi nhấn nút Close**

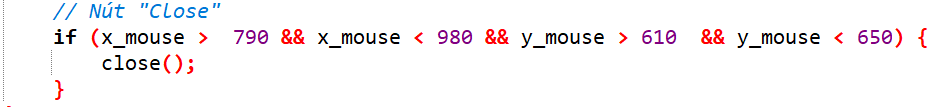


Figure 21. Xử lý Close

## **2.4. PHÂN TÍCH THUẬT TOÁN**

### **2.4.1. Thuật toán vẽ đa giác (ART\_cool)**

A white background with black and red text

Description automatically generated

Figure 22. Thuật toán vẽ đa giác từ các đường thẳng

* Vòng lặp for: Vòng lặp này được sử dụng để duyệt qua các điểm trong mảng td.
* i = 0: Khởi tạo biến i với giá trị 0.
* i < sodinh - 1: Điều kiện lặp là i nhỏ hơn số đỉnh (sodinh) trừ 1. Điều này đảm bảo rằng chỉ có các cặp điểm kề nhau trong mảng td được nối với nhau.
* i++: Tăng giá trị của i sau mỗi lần lặp.
* line(td[i].x, td[i].y, td[i + 1].x, td[i + 1].y): Câu lệnh này sẽ vẽ một đoạn thẳng nối từ điểm td[i] đến điểm td[i + 1]. Cụ thể, td[i].x và td[i].y là tọa độ x và y của điểm td[i], còn td[i + 1].x và td[i + 1].y là tọa độ x và y của điểm td[i + 1].
* line(td[sodinh - 1].x, td[sodinh - 1].y, td[0].x, td[0].y): Câu lệnh này vẽ một đoạn thẳng nối từ điểm cuối cùng trong mảng td (với chỉ số sodinh - 1) đến điểm đầu tiên trong mảng (td[0]). Điều này để đảm bảo rằng hình đa giác được hoàn thành bằng việc nối điểm cuối cùng với điểm đầu tiên.

### **2.4.2. Thuật toán vẽ hình tròn (circle)**

Sử dụng công thức của Midpoint để vẽ hình tròn, vì hình tròn chỉ có 2 đỉnh nếu khắc 2 thì sẽ trả về “return”.

* Tính bán kính (r): Sử dụng công thức Euclid để tính bán kính của hình tròn. Bán kính được tính bằng cách tính khoảng cách Euclid giữa hai điểm trong mảng td: (td[1].x - td[0].x) và (td[1].y - td[0].y). Công thức này sử dụng hàm pow() để tính bình phương và hàm sqrt() để tính căn bậc hai.
* Khởi tạo biến (x, y, p): Khởi tạo các biến x và y để theo dõi tọa độ của các điểm trên đường viền của hình tròn. Biến p được sử dụng để quyết định vị trí điểm kế tiếp trên đường viền.
* Vòng lặp while(x<=y): Vòng lặp này sẽ chạy cho đến khi x vượt quá y, điều này đảm bảo rằng các điểm trên đường viền của hình tròn đã được duyệt hết.
* Vẽ các điểm trên đường viền: Trong vòng lặp, hàm putpixel() được sử dụng để vẽ các điểm trên đường viền của hình tròn. Các điểm được vẽ thông qua việc gọi hàm putpixel() với tọa độ (td[0].x+x, td[0].y+y), (td[0].x-x, td[0].y+y), (td[0].x-x, td[0].y-y), (td[0].x+x, td[0].y-y), (td[0].x+y, td[0].y+x), (td[0].x-y, td[0].y+x), (td[0].x+y, td[0].y-x), (td[0].x-y, td[0].y-x). Giá trị 255 được truyền vào putpixel() để đặt màu cho điểm vẽ (trong đây là màu trắng).
* Cập nhật p, x và y: Trong vòng lặp, giá trị p được cập nhật theo công thức để quyết định vị trí điểm kế tiếp trên đường viền. Nếu p là âm, p được cập nhật bằng 4\*x+6, ngược lại, p được cập nhật bằng 4\*(x-y)+10 và giá trị của y giảm đi một đơn vị.

A screenshot of a computer program

Description automatically generated

Figure 23. Thuật toán vẽ hình tròn

### **2.4.3. LOAD FILE**

**2.4.3.2. Load file đa giác**

* A computer code with text

  Description automatically generated with medium confidencefp=fopen: Câu lệnh đọc tên file được dẫn

Figure 24. Đọc file hình đa giác

* fscanf(fp, “%d %d”, &td[i].x, &td[i].y: in các tọa độ trong file
* ART\_cool: Truyền các tọa độ trong file đọc vào hàm này để xử lý và vẽ ra đa giác hoàn chỉnh.

**A screen shot of a computer code

Description automatically generated2.4.2.3. Load file hình tròn**

Figure 25.Đọc file hình tròn

* p=fopen: Câu lệnh đọc tên file được dẫn
* fscanf(fp, “%d %d”, &td[i].x, &td[i].y: in các tọa độ trong file
* circle(): Truyền các tọa độ trong file đọc vào hàm này để xử lý và vẽ ra hình tròn hoàn chỉnh.

### **2.4.3. Nhập từ bàn phím**

Figure 26. Nhập từ bàn phím

* Nhập số đỉnh
* Nhập số tọa độ (td[i].x, td[i].y)

### **A screenshot of a computer code Description automatically generated2.4.4. Phép tịnh tiến (Translation)**

Figure 27. Phép biến đổi tịnh tiến

* Gọi khoảng cách di chuyển theo trục x,y là: targetx, targety; số lần bước nhảy steps gán bằng 10
* Stepx, stepy là khoảng cách di chuyển từng bước từ 1 đến steps = 10 bằng hiệu của khoảng cách trục tọa độ mới (targetx, targety) trừ đi tọa độ ban đầu (x,y) và chia cho số bước nhảy (steps)
* Vòng lặp for để di chuyển và cập nhật tọa độ của các đỉnh trong mảng td và tọa độ (x, y). --Vòng lặp chạy từ step = 1 đến steps.

x += stepx và y += stepy: Cập nhật tọa độ (x, y) bằng cách thêm giá trị của stepx vào x và giá trị của stepy vào y.

* Vòng lặp for lồng bên trong được sử dụng để cập nhật tọa độ của các điểm trong mảng td. Tọa độ của mỗi điểm được cập nhật bằng cách thêm giá trị của stepx vào td[i].x và giá trị của stepy vào td[i].y.
* cleardevice(): Xóa toàn bộ nội dung hiện tại trên màn hình đồ họa.
* ART\_init(): Gọi một hàm ART\_init() để khởi tạo một nền tảng đồ họa.
* Kiểm tra số đỉnh (sodinh): Nếu số đỉnh (sodinh) lớn hơn 2, gọi hàm ART\_cool(), ngược lại, gọi hàm circle(). Điều này để vẽ một hình đa giác nếu có nhiều hơn hai đỉnh, và vẽ một hình tròn nếu chỉ có hai đỉnh.

### **2.4.5. Phép quay góc tọa độ (Rotation)**

A computer screen shot of a code

Description automatically generated

Figure 28. Phép quay góc tọa độ

* Gọi khoảng cách di chuyển theo trục x,y và hệ số góc quay là: targetx, targety, degree; số lần bước nhảy steps gán bằng 10
* Biến dx, dy dùng để lưu trữ sựu chênh lệch giữa tọa độ cũ và tọa độ mới của mỗi điểm trong mảng td và angleDelta tính toán sự chênh lệch góc cho mỗi bước.
* Stepx, stepy là khoảng cách di chuyển từng bước từ 1 đến steps = 10 bằng hiệu của khoảng cách trục tọa độ mới (targetx, targety) trừ đi tọa độ ban đầu (x,y) và chia cho số bước nhảy (steps)
* Tính bán kính góc quay bằng công thức: radian = degree / (180 \* 3.14)
* Vòng lặp for để di chuyển và xoay hình vẽ chạy từ step = 1 đến steps = 10.
* currentAngle = step \* angleDelta: Tính toán góc hiện tại thay đổi dần theo mỗi bước nhảy.
* Vòng lặp for lồng bên trong: Vòng lặp này được sử dụng để cập nhật tọa độ của các điểm trong mảng td. Với mỗi điểm, dx và dy được tính bằng sự chênh lệch giữa tọa độ cũ (td[i].x, td[i].y) và tọa độ mới (targetx, targety). Tọa độ (td[i].x, td[i].y) được cập nhật bằng cách tính toán tọa độ mới bằng công thức (targetx + dx \* cos(currentAngle) - dy \* sin(currentAngle)) cho x và (targety + dx \* sin(currentAngle) + dy \* cos(currentAngle)) cho y.
* Hàm floor() được sử dụng để làm tròn giá trị số thập phân thành số nguyên.
* cleardevice(): Xóa toàn bộ nội dung hiện tại trên màn hình đồ họa.
* ART\_init(): Gọi một hàm ART\_init() để khởi tạo một nền tảng đồ họa.
* Kiểm tra số đỉnh (sodinh): Nếu số đỉnh (sodinh) lớn hơn 2, gọi hàm ART\_cool(), ngược lại, gọi hàm circle(). Điều này để vẽ một hình đa giác nếu có nhiều hơn hai đỉnh, và vẽ một hình tròn nếu chỉ có hai đỉnh.

### **A screenshot of a computer program Description automatically generated2.4.6. Phép biến đổi tỉ lệ (Scaling)**

Figure 29. Phép biến đổi tỉ lệ

* Khai báo biến scale\_x, scale\_y, và dx, dy: dùng để lưu trữ các tỷ lệ biến đổi theo trục x và y, và sự chênh lệch tọa độ giữa điểm cũ và điểm mới.
* Nhập giá trị x, y, scale\_x, và scale\_y từ người dùng qua bàn phím.
* Khai báo biến steps: là các bước biến đổi kích thước và scale\_xDelta, scale\_yDelta:
* tính toán sự chênh lệch tỷ lệ biến đổi cho mỗi bước.
* Vòng lặp for để biến đổi kích thước: Vòng lặp này được sử dụng để thực hiện biến đổi kích thước của đối tượng vẽ. Vòng lặp chạy từ step = 1 đến steps.
* currentScale\_x và currentScale\_y: Tính toán tỷ lệ biến đổi hiện tại currentScale\_x và currentScale\_y bằng cách nhân step với scale\_xDelta và scale\_yDelta và chia cho 10. Điều này đảm bảo rằng tỷ lệ biến đổi sẽ thay đổi dần theo mỗi bước.
* Vòng lặp for lồng bên trong: Vòng lặp này được sử dụng để cập nhật tọa độ của các điểm trong mảng td. Với mỗi điểm, dx và dy là sự chênh lệch giữa tọa độ cũ (td[i].x, td[i].y) và tọa độ mới (x, y). Tọa độ (td[i].x, td[i].y) được cập nhật bằng cách tính toán tọa độ mới bằng công thức (x + (int)((float)(td[i].x - x) \* currentScale\_x)) cho trục x và (y + (int)((float)(td[i].y - y) \* currentScale\_y)) cho trục y. Hàm int được sử dụng để chuyển đổi giá trị số thực thành số nguyên và làm tròn giá trị.
* Kiểm tra số đỉnh (sodinh): Nếu số đỉnh (sodinh) lớn hơn 2, thực hiện các bước sau:
* cleardevice(): Xóa toàn bộ nội dung hiện tại trên màn hình đồ họa.
* ART\_init(): Gọi một hàm ART\_init() để khởi tạo một nền tảng đồ họa.
* ART\_cool(): Gọi một hàm ART\_cool() để vẽ một hình đa giác nếu có nhiều hơn hai đỉnh.
* Nếu số đỉnh (sodinh) không lớn hơn 2, thực hiện các bước tương tự như trên, nhưng gọi hàm circle() để vẽ một hình tròn nếu chỉ có hai đỉnh.

# A screenshot of a computer Description automatically generatedC**HƯƠNG 3. KẾT QUẢ**

Figure 30. Giao diện Load file là 1

Figure 31. Giao diện chính

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure 32. Giao diện Loadfile là 2

Figure 33. Giao diện Loadfile là 3

# **CHƯƠNG 4. TỔNG KẾT**

## **4.1. Ưu điểm**

* Nắm vững được kiến thức các phép biến đổi cơ bản như: phép tịnh tiến, phép quay góc tọa độ, phép biến đổi tỉ lệ, và cách vẽ đường tròn Midpoint
* Làm được giao diện cho phù hợp dễ nhìn
* Nắm kỹ năng vẽ đồ họa của môn Kỹ thuật đồ họa

## **4.2. Nhược điểm**

* Chưa mô tả được rõ các phép biến đổi
* Xử lý code rườm rà và dài dòng

# **Tài liệu tham khảo:**

[1] Bài giảng Kỹ thuật đồ họa, Đoàn Vũ Thịnh, 2019

[2] Hướng dẫn giải chi tiết và lập trình Kỹ thuật đồ hoạ, Đoàn Vũ Thịnh, 2021

[3] Computer Graphics, C Version (2nd Edition), Donald Hearn, 1996

[4] Giải thuật và lập trình, Lê Minh Hoàng, 2006