BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**BÁO CÁO THỰC TẬP CƠ SỞ**

**XÂY DỰNG CÁC THUẬT TOÁN ĐỒ HỌA CƠ BẢN VỚI C++**

**Giảng viên hướng dẫn: Nguyễn Thị Hương Lý**

**Sinh viên thực hiện: Nguyễn Chí Trường**

**Mã số sinh viên: 64139004**

Khánh Hòa – 01/2025

# LỜI CẢM ƠN

Để hoàn thành bài báo cáo thực tập cơ sở này, trước tiên em xin gửi lời cảm ơn chân thành đến quý thầy cô Khoa Công nghệ Thông tin – Trường Đại học Nha Trang vì sự quan tâm, hỗ trợ trong suốt quá trình thực tập.

Em xin đặc biệt cảm ơn cô Nguyễn Thị Hương Lý, người đã tận tâm hướng dẫn, chỉ bảo em trong quá trình thực hiện đề tài, giúp em hoàn thành bài báo cáo này. Sự giúp đỡ và những chỉ dẫn quý báu của cô đã mang lại cho em rất nhiều bài học bổ ích..

Trong quá trình nghiên cứu và thực hiện đề tài, em không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm từ các thầy cô. Do kinh nghiệm thực tiễn và kiến thức còn hạn chế, bài báo cáo này không thể tránh khỏi những khiếm khuyết. Em rất mong nhận được ý kiến đóng góp, phản hồi từ quý thầy cô để có thể cải thiện và hoàn thiện bản thân hơn trong công việc và học tập.

Em xin chân thành cảm ơn và kính chúc quý thầy cô sức khỏe, thành công trong công tác giảng dạy.

# MỤC LỤC

[LỜI CẢM ƠN i](#_Toc93392449)

[MỤC LỤC ii](#_Toc93392450)

[PHẦN MỞ ĐẦU 1](#_Toc93392451)

[1. Lý do chọn đề tài: 1](#_Toc93392452)

[2. Mục tiêu của đề tài. 1](#_Toc93392453)

[3. Phạm vi nghiên cứu. 2](#_Toc93392454)

[4. Phương pháp nghiên cứu. 2](#_Toc93392455)

[Chương 1. TỔNG QUAN VỀ VẤN ĐỀ NGHIÊN CỨU 2](#_Toc93392456)

[1.1 CƠ SỞ LÝ THUYẾT 2](#_Toc93392457)

[1.1.1 Tổng quan về thuật toán đồ họa cơ bản 2](#_Toc93392458)

[1.1.2 Thuật toán Bresenham 4](#_Toc93392519)

[1.1.3 Thuật toán Midpoint 5](#_Toc93392519)

[1.2 CÔNG CỤ VÀ NGÔN NGỮ LẬP TRÌNH ĐƯỢC SỬ DỤNG 6](#_Toc93392522)

[1.2.1 Tổng quan về Dev C++ 6](#_Toc93392523)

[1.2.2 Ngôn ngữ C++ 7](#_Toc93392524)

[Chương 2. ĐẶC TẢ BÀI TOÁN 9](#_Toc93392526)

[2.1 BÀI TOÁN 9](#_Toc93392527)

[2.2 Hướng xử lý và các thuật toán để thực hiện các yêu cầu: 9](#_Toc93392528)

[Mức 1: Cài đặt các thuật toán đồ họa 2 chiều ( sinh đoạn thẳng, các được cong Ellipstice, sinh một số hình khác, sinh đường tự do ) 9](#_Toc93392529)

[Hướng xử lỷ 9](#_Toc93392530)

[Chi tiết chương trình 9](#_Toc93392531)

[Mức 2: Các chức năng của người dùng được tổ chức tốt, thân thiện, dễ sử dụng. Chức năng xóa hình 1](#_Toc93392532)5

[Hướng xử lỷ 15](#_Toc93392530)

[Chi tiết chương trình 1](#_Toc93392531)5

[Chương 3. CÀI ĐẶT CHƯƠNG TRÌNH 1](#_Toc93392533)9

[3.1 Code chương trình 1](#_Toc93392534)9

[3.1.1 Mức 1 1](#_Toc93392535)9

[Code các hàm sinh hình 15](#_Toc93392530)

[3.1.2 Mức 2 2](#_Toc93392538)0

[Code giao diện 2](#_Toc93392530)0

[Code kiểm tra điểm nhấn chuột có nằm trong hình không 2](#_Toc93392530)3

[Code xóa hình 2](#_Toc93392530)4

[3.1.3 Kết quả chạy thử chương trình 2](#_Toc93392538)6

[Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 2](#_Toc93392539)7

[4.1 KẾT QUẢ ĐẠT ĐƯỢC 2](#_Toc93392540)7

[4.2 ƯU ĐIỂM 2](#_Toc93392541)7

[4.3 HẠN CHẾ 2](#_Toc93392542)7

[4.4 HƯỚNG PHÁT TRIỂN 2](#_Toc93392543)7

[4.5 KẾT LUẬN 2](#_Toc93392544)7

[TÀI LIỆU THAM KHẢO 2](#_Toc93392545)8

# PHẦN MỞ ĐẦU

1. **LÝ DO CHỌN ĐỀ TÀI.**

**Công nghệ thông tin** là một trong những ngành có sự phát triển mạnh mẽ và vượt bậc trong những năm gần đây. Trong bối cảnh xã hội ngày càng phát triển nhanh chóng, công nghệ thông tin đã trở thành một phần không thể thiếu trong hầu hết các lĩnh vực và đóng vai trò quan trọng trong đời sống hàng ngày. Đây là ngành học đòi hỏi người học phải có nền tảng kiến thức vững chắc, khả năng tư duy logic cao và sự hiểu biết rộng rãi trên nhiều lĩnh vực.Với em, những sinh viên ngành Công nghệ thông tin, việc liên tục nâng cao và củng cố kiến thức là điều vô cùng quan trọng. Để đạt được mục tiêu đó, việc áp dụng lý thuyết vào thực tế là một yêu cầu thiết yếu. Chính vì vậy, đề tài mà em lựa chọn để thực hiện trong bài báo cáo này là **"**Xây dựng các thuật toán đồ họa cơ bản với C++**"**. Qua đó, em mong muốn tìm hiểu, phát triển và áp dụng các thuật toán cơ bản trong đồ họa máy tính, sử dụng ngôn ngữ lập trình C++ để giải quyết các vấn đề thực tiễn trong lĩnh vực đồ họa, từ đó củng cố và mở rộng thêm kiến thức chuyên môn của mình.

1. **MỤC TIÊU CỦA ĐỀ TÀI.**
   * Ôn lại và củng cố các kiến thức về các thuật toán đồ họa cơ bản như vẽ đường thẳng, vẽ hình tròn, tô màu và các phép biến đổi hình học trong không gian 2D. Đồng thời làm quen với các kỹ thuật tối ưu hóa thuật toán để nâng cao hiệu suất ứng dụng đồ họa.lại kiến thức về phần mảng, chuỗi cũng như kiểu dữ liệu cấu trúc...
   * Phát triển kỹ năng lập trình C++ trong việc triển khai các thuật toán đồ họa cơ bản, đồng thời học cách xử lý và tối ưu mã nguồn để các ứng dụng đồ họa chạy hiệu quả hơn.
   * Vận dụng lý thuyết về đồ họa máy tính để xây dựng các ứng dụng đồ họa cơ bản, qua đó giải quyết các vấn đề thực tế trong lĩnh vực đồ họa máy tính và lập trình phần mềm.
2. **PHẠM VI NGHIÊN CỨU.**
   * Nghiên cứu và triển khai các thuật toán đồ họa cơ bản trong C++ như thuật toán vẽ đường thẳng, hình tròn, hình tam giác, hình vuông, hình ngôi sao và đường tự do.
   * Nghiên cứu cách xóa một hình sau khi vẽ.
   * Nghiên cứu cách xây dựng giao diện tốt, thân thiện đơn giản, dễ sử dụng.
3. **PHƯƠNG PHÁP NGHIÊN CỨU.**
   * Tìm kiếm và nghiên cứu trên mạng Internet về các tài liệu liên quan đến đồ họa.
   * Nghiên cứu dựa trên bài giảng Kỹ thuật đồ họa của thầy Đoàn Vũ Thịnh, Trường Đại học Nha Trang
   * Nghiên cứu các dự án mã nguồn mở và các ví dụ thực tế về đồ họa máy tính để học hỏi kinh nghiệm, tối ưu hóa thuật toán, và áp dụng các kỹ thuật nâng cao trong việc xây dựng ứng dụng đồ họa.

# TỔNG QUAN VỀ VẤN ĐỀ NGHIÊN CỨU

## CƠ SỞ LÝ THUYẾT

### Tổng quan về thuật toán đồ họa cơ bản

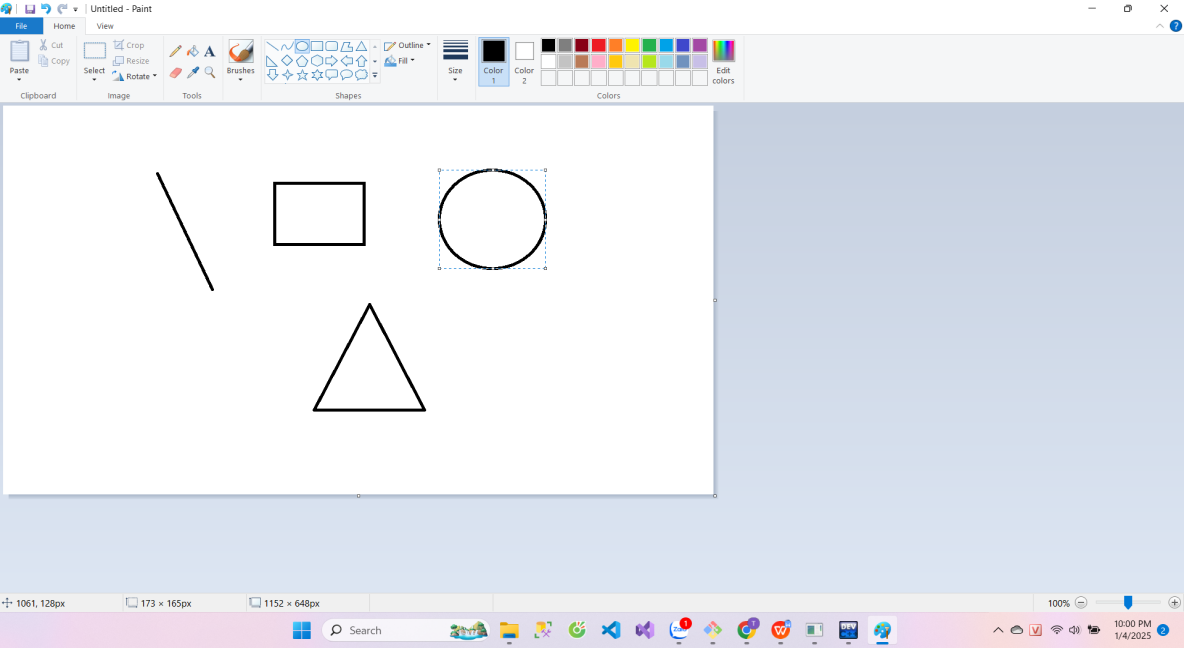
Thuật toán đồ họa là các phương pháp được sử dụng trong lập trình để tạo ra và xử lý các hình ảnh, đối tượng đồ họa, cũng như các chuyển động trên màn hình máy tính. Các thuật toán này đóng vai trò quan trọng trong việc xây dựng và phát triển các ứng dụng đồ họa, từ các ứng dụng đơn giản như vẽ hình cơ bản cho đến các ứng dụng phức tạp trong đồ họa 3D hay mô phỏng.

Một trong những thuật toán cơ bản đầu tiên trong đồ họa là **thuật toán vẽ đường thẳng**, chẳng hạn như thuật toán **Bresenham hay thuật toán Midpoint**. Thuật toán này giúp vẽ đường thẳng trên màn hình mà không cần phải tính toán liên tục các điểm trên đường thẳng, từ đó tiết kiệm tài nguyên máy tính và cải thiện hiệu suất đồ họa.Ngoài ra từ đường thẳng có thể xây dựng nên các hình khác như hình vuông, tam giác,…

Bên cạnh đó, **thuật toán tô màu vùng** (hay **Flood Fill**) là thuật toán cơ bản được sử dụng trong việc tô màu một khu vực liên thông trong hình ảnh, phổ biến trong các phần mềm chỉnh sửa ảnh và đồ họa. Thuật toán này được thiết kế để "đổ đầy" màu vào các vùng có cùng màu sắc, giúp dễ dàng thay đổi màu sắc của một phần trong ảnh mà không làm ảnh hưởng đến các phần còn lại.

Cuối cùng, **thuật toán biến đổi hình học** giúp thay đổi vị trí, kích thước và hình dạng của các đối tượng đồ họa trong không gian 2D. Các phép biến đổi này bao gồm **tịnh tiến** (di chuyển đối tượng), **xoay** (quay đối tượng quanh một điểm), **thu phóng** (thay đổi kích thước của đối tượng), và **phản chiếu** (lật đối tượng). Những phép biến đổi này rất quan trọng trong việc tạo ra các hiệu ứng động hoặc thay đổi hình dạng trong các ứng dụng đồ họa.

Các thuật toán đồ họa cơ bản này không chỉ có ý nghĩa quan trọng trong việc xây dựng các ứng dụng đồ họa đơn giản mà còn là nền tảng để phát triển các kỹ thuật đồ họa phức tạp hơn, phục vụ cho nhiều ngành công nghiệp khác nhau, như game, mô phỏng 3D, và thiết kế đồ họa.



Hình 1: Một số hình cơ bản được vẽ trên ứng dụng Paint

### Thuật toán Bresenham

Thuật toán **Bresenham** là một thuật toán vẽ đường thẳng nổi bật trong đồ họa máy tính, được phát triển bởi Jack Bresenham vào năm 1962. Nó được thiết kế để vẽ các đường thẳng trong không gian 2D một cách hiệu quả mà không cần sử dụng phép toán số thực, thay vào đó là các phép toán số nguyên. Điều này giúp tiết kiệm tài nguyên hệ thống, đặc biệt là trong những hệ thống có phần cứng hạn chế hoặc độ phân giải thấp. Thuật toán này chủ yếu được sử dụng để vẽ các đường thẳng với độ chính xác cao, đảm bảo các điểm vẽ nằm càng gần càng tốt với đường thẳng lý tưởng.

Quá trình hoạt động của thuật toán bắt đầu từ việc xác định hai điểm đầu và cuối của đường thẳng, giả sử là (x0, y0) và (x1, y1). Đầu tiên, thuật toán tính toán sự thay đổi giữa các tọa độ x và y, cụ thể là dx = x1 - x0 và dy = y1 - y0. Dựa vào sự thay đổi này, thuật toán xác định hướng di chuyển, tức là có nên di chuyển theo chiều ngang (theo trục x) hay chiều dọc (theo trục y) để vẽ đường thẳng.

Khi vẽ đường thẳng, thuật toán không chỉ đơn giản di chuyển theo một trục mà còn có thể cần phải điều chỉnh theo cả hai trục x và y để giữ đường thẳng gần nhất với lý thuyết. Nếu độ dốc của đường thẳng nhỏ hơn hoặc bằng 1 (dy <= dx), thuật toán sẽ di chuyển chủ yếu theo trục x, và khi cần, sẽ điều chỉnh theo trục y. Ngược lại, nếu độ dốc của đường thẳng lớn hơn 1 (dy > dx), thuật toán sẽ di chuyển chủ yếu theo trục y, và điều chỉnh theo trục x khi cần thiết.

Sau khi tính toán sự thay đổi trên cả hai trục, thuật toán Bresenham sẽ tiếp tục vẽ các điểm trên đường thẳng. Ở mỗi bước, thuật toán sẽ quyết định điểm tiếp theo phải vẽ dựa trên các thay đổi của x và y. Quyết định này giúp các điểm vẽ càng gần càng tốt với đường thẳng lý tưởng, mà không cần phải tính toán phức tạp bằng số thực. Điều này làm cho thuật toán trở nên rất hiệu quả trong việc vẽ đường thẳng mà không tốn nhiều tài nguyên hệ thống.

Một trong những điểm mạnh của thuật toán Bresenham là sự đơn giản và hiệu quả trong việc vẽ các đường thẳng. Thay vì sử dụng các phép toán phức tạp như tính toán số thực, thuật toán chỉ sử dụng các phép toán cơ bản như cộng và trừ, điều này làm giảm đáng kể mức độ phức tạp và giúp thực thi nhanh chóng. Thuật toán có thể vẽ các đường thẳng mượt mà, với độ chính xác cao, và không có sai lệch lớn giữa các điểm vẽ và đường lý tưởng.

Thuật toán Bresenham không chỉ được sử dụng để vẽ đường thẳng mà còn có thể được mở rộng để vẽ các đối tượng khác như **hình tròn** và **hình elip**. Nó được áp dụng rộng rãi trong các hệ thống đồ họa máy tính, từ các ứng dụng phần mềm vẽ 2D, phần mềm đồ họa, cho đến các trò chơi điện tử. Với việc sử dụng các phép toán số nguyên, thuật toán này đặc biệt hữu ích trong các hệ thống có phần cứng hạn chế hoặc khi cần xử lý nhanh các hình ảnh trong các trò chơi hoặc phần mềm có yêu cầu về hiệu suất cao.

### **Thuật toán Midpoint**

Thuật toán **Midpoint** là một phương pháp hiệu quả và được sử dụng phổ biến trong đồ họa máy tính để vẽ các đường thẳng trong không gian 2D. Thuật toán này hoạt động dựa trên nguyên lý tính toán điểm trung gian (midpoint) giữa các điểm trên đường thẳng, từ đó giúp xác định chính xác vị trí của điểm tiếp theo cần vẽ mà không cần phải sử dụng phép toán số thực, chỉ cần dùng các phép toán số nguyên đơn giản. Điều này giúp tối ưu hóa hiệu suất và tiết kiệm tài nguyên hệ thống, đặc biệt hữu ích trong các hệ thống có phần cứng hạn chế, như thiết bị di động hay các hệ thống đồ họa cũ.

Khi bắt đầu vẽ đường thẳng, thuật toán sẽ nhận vào hai điểm đầu và cuối của đường thẳng, giả sử là (x0, y0) và (x1, y1). Thuật toán sẽ tính toán sự thay đổi về tọa độ của các điểm trên trục x và trục y bằng cách xác định dx = x1 - x0 và dy = y1 - y0. Từ đây, thuật toán sẽ tính toán một biến trung gian (d) để giúp quyết định hướng đi của điểm tiếp theo cần vẽ. Nếu đường thẳng có độ dốc nhỏ (dy < dx), thuật toán sẽ vẽ điểm theo chiều ngang (theo trục x) chủ yếu, trong khi biến d được tính toán để giúp xác định xem có cần di chuyển thêm một đơn vị trên trục y hay không.

Sau khi khởi tạo, thuật toán tiếp tục vẽ các điểm từ (x0, y0) đến (x1, y1) bằng cách lặp lại một quy trình. Mỗi lần lặp, thuật toán sẽ kiểm tra giá trị của biến trung gian d. Nếu d lớn hơn 0, điều này có nghĩa là điểm tiếp theo cần di chuyển lên theo chiều dọc (tăng y), còn nếu d nhỏ hơn hoặc bằng 0, thuật toán chỉ cần di chuyển theo chiều ngang (tăng x). Sau mỗi lần cập nhật vị trí của điểm tiếp theo, giá trị của biến d sẽ được điều chỉnh lại dựa trên các giá trị của dx và dy. Quá trình này tiếp tục cho đến khi vẽ được toàn bộ đường thẳng từ điểm đầu đến điểm cuối.

Điểm mạnh của thuật toán Midpoint vẽ đường thẳng là khả năng sử dụng các phép toán số nguyên thay vì số thực, giúp tăng tốc độ tính toán và giảm thiểu tài nguyên hệ thống. Thuật toán này còn tận dụng được tính đối xứng của các hình học trong không gian 2D, giúp vẽ đường thẳng mượt mà và chính xác mà không cần phải tính toán lại các điểm đối xứng. Nhờ vào sự tối ưu này, thuật toán Midpoint vẽ đường thẳng đã trở thành một công cụ quan trọng trong các phần mềm đồ họa 2D, thiết kế đồ họa, và trong phát triển các trò chơi điện tử, nơi yêu cầu độ chính xác và hiệu suất cao.

## CÔNG CỤ VÀ NGÔN NGỮ LẬP TRÌNH ĐƯỢC SỬ DỤNG

### Tổng quan về Dev C++

* **Dev C++** là một môi trường phát triển tích hợp (IDE) mã nguồn mở dành cho lập trình viên C và C++. Dev C++ là một công cụ mạnh mẽ để phát triển các ứng dụng C và C++ trên hệ điều hành Windows, giúp lập trình viên dễ dàng biên dịch, gỡ lỗi và quản lý dự án lập trình. Đây là một lựa chọn lý tưởng cho những ai muốn làm việc với các dự án C/C++ đơn giản và không yêu cầu những tính năng phức tạp của các IDE nặng như Microsoft Visual Studio.
* **Ưu điểm:**
* **Giao diện đơn giản và dễ sử dụng**: Dev C++ có giao diện trực quan, dễ làm quen với các lập trình viên mới. Nó không yêu cầu cấu hình phức tạp và dễ dàng bắt đầu viết mã.Đa ngôn ngữ lập trình
* **Nhẹ và nhanh**: Vì là phần mềm nhẹ, Dev C++ không yêu cầu nhiều tài nguyên hệ thống, giúp tiết kiệm bộ nhớ và thời gian khởi động
* **Miễn phí và mã nguồn mở**: Dev C++ là phần mềm mã nguồn mở, giúp người dùng có thể tải về và sử dụng miễn phí, đồng thời có thể tùy chỉnh để phục vụ mục đích riêng của mình.Lưu trữ phân cấp.
* **Hỗ trợ biên dịch nhanh**: Dev C++ tích hợp trình biên dịch MinGW giúp biên dịch các chương trình C và C++ nhanh chóng và hiệu quả.
* **Dễ dàng gỡ lỗi**: Dev C++ cung cấp các công cụ gỡ lỗi cơ bản giúp người dùng tìm và sửa lỗi trong mã nguồn.
* **Hạn chế:**
* **Tính năng hạn chế so với các IDE khác**: Dev C++ không có nhiều tính năng hỗ trợ phát triển ứng dụng web hay ứng dụng di động như các IDE phức tạp hơn như Visual Studio.
* **Không cập nhật thường xuyên**: Dev C++ đã lâu không được cập nhật và không có nhiều cải tiến, khiến cho nó ít phù hợp với các dự án phức tạp hoặc yêu cầu tính năng hiện đại.
* **Hạn chế đối với người dùng chuyên nghiệp**: Mặc dù Dev C++ phù hợp cho những người mới bắt đầu, nhưng đối với các lập trình viên chuyên nghiệp hoặc làm việc trên các dự án lớn, các IDE như Visual Studio hay Code::Blocks có thể cung cấp nhiều công cụ hỗ trợ hơn.

### Ngôn ngữ C++

* **C++** là một ngôn ngữ lập trình bậc trung (middle-level programming language), được phát triển bởi Bjarne Stroustrup như một sự mở rộng của ngôn ngữ C. C++ được xem là "C với các lớp Class", giúp cải thiện khả năng lập trình hướng đối tượng so với C. Ngôn ngữ này đã được phát triển và mở rộng theo thời gian, hiện tại, C++ hỗ trợ nhiều tính năng mạnh mẽ như lập trình tổng quát, lập trình hướng đối tượng, lập trình thủ tục, cùng với các tính năng như đa hình, dữ liệu trừu tượng và khả năng thao tác với bộ nhớ cấp thấp. C++ là một ngôn ngữ đa mẫu hình (multi-paradigm) và sử dụng kiểu dữ liệu tĩnh (static typing). Kể từ thập niên 1990, C++ đã trở thành một trong những ngôn ngữ lập trình phổ biến và được ưa chuộng trong ngành công nghiệp phần mềm, đặc biệt trong các ứng dụng đòi hỏi hiệu suất cao và khả năng tương tác với phần cứng.
* **Ưu điểm:**
* **Hướng đối tượng**: C++ hỗ trợ lập trình hướng đối tượng (OOP), giúp người lập trình có thể xây dựng các hệ thống phức tạp dễ dàng hơn thông qua các lớp (class) và đối tượng (object).
* **Quản lý bộ nhớ linh hoạt**: C++ cho phép lập trình viên quản lý vùng nhớ trực tiếp, giúp tối ưu hóa hiệu suất của chương trình và kiểm soát tài nguyên hệ thống một cách chi tiết.
* **Giao tiếp với phần cứng**: Với khả năng làm việc ở mức độ thấp, C++ dễ dàng tương tác với phần cứng, điều này rất quan trọng đối với các ứng dụng nhúng, hệ điều hành, hoặc phần mềm yêu cầu hiệu suất cao.
* **Hạn chế:**
* **Tràn vùng nhớ**: Một trong những vấn đề phổ biến khi lập trình với C++ là tràn bộ nhớ, đặc biệt khi lập trình viên không quản lý đúng cách bộ nhớ động, dẫn đến lỗi khi truy cập vùng bộ nhớ không hợp lệ.
* **Khó khăn trong OOP**: Mặc dù C++ hỗ trợ lập trình hướng đối tượng, nhưng cách thực hiện và sử dụng các tính năng này đôi khi có thể khá phức tạp và khó hiểu, đặc biệt đối với những lập trình viên mới làm quen với OOP.
* **Con trỏ khó sử dụng**: Con trỏ (pointer) là một phần quan trọng trong C++, nhưng chúng cũng gây không ít khó khăn cho lập trình viên, nhất là khi thao tác với bộ nhớ động, việc sai sót trong việc quản lý con trỏ có thể dẫn đến các lỗi nghiêm trọng như rò rỉ bộ nhớ hoặc lỗi truy cập vùng nhớ không hợp lệ.

# ĐẶC TẢ BÀI TOÁN

## BÀI TOÁN

Vận dụng kiến thức lập trình C++ để xây dựng các thuật toán đồ họa.

* 1. **Mức 1:** Cài đặt các thuật toán đồ họa 2 chiều ( sinh đoạn thẳng, các đường cong Ellipstice, sinh một số hình khác, sinh đường tự do )
  2. **Mức 2:** Các chức năng và giao diện chương trình được tổ chức tốt, thân thiện, dễ dùng. Chức năng xóa hình

## HƯỚNG XỬ LÝ VÀ CÁC THUẬT TOÁN ĐỂ THỰC HIỆN CÁC YÊU CẦU:

### Mức 1: Cài đặt các thuật toán đồ họa 2 chiều ( sinh đoạn thẳng, các đường cong Ellipstice, sinh một số hình khác, sinh đường tự do )

##### **Hướng xử lý**

* Xây dựng thuật toán vẽ đường thẳng trong không gian 2D, áp dụng thuật toán như Bresenham để đảm bảo tính chính xác và hiệu suất. Từ đó sinh ra được các hình khác
* Cài đặt thuật toán vẽ các đường cong elip trong không gian 2D, với các phương pháp như thuật toán Bresenham.
* Từ thuật toán vẽ đường thẳng Bresenham, xây dựng thêm cách vẽ các hình khác

##### **Chi tiết chương trình**

Khai báo các thư viện C++

|  |
| --- |
| *#include <conio.h>*  *#include <graphics.h>*  *#include <cmath>*  *#include <iostream>*  *#include <vector>*  *using namespace std;* |

Ta dung kiểu struct để lưu trữ các biến thông tin về từng hình.

|  |
| --- |
| struct Shape {  int type; // loai hinh  int x1, y1, x2, y2; // toa do dung cho duong thang, hinh vuong  int radius; // ban kinh dung cho hinh ngoi sao, tron  int color; // mau  vector<pair<int, int> > points; // luu cac diem cua duong tu do  }; |

***Hàm vẽ đoạn thẳng bằng giải thuật Bresenham.***

Hàm drawLineBresenham sử dụng thuật toán Bresenham để vẽ đường thẳng trong đồ họa máy tính. Đầu vào của hàm gồm các tọa độ điểm bắt đầu (x0, y0) và điểm kết thúc (x1, y1) của đường thẳng, cùng với tham số color để xác định màu sắc của đường thẳng. Thuật toán này tính toán độ dài theo trục x (dx) và trục y (dy) giữa hai điểm đầu và cuối, đồng thời xác định hướng di chuyển trên mỗi trục thông qua các giá trị sx và sy (tương ứng với hướng di chuyển trên trục x và y). Nếu độ dài trên trục x (dx) lớn hơn hoặc bằng độ dài trên trục y (dy), thuật toán áp dụng công thức lỗi err = 2 \* dy - dx để quyết định khi nào thay đổi tọa độ y khi di chuyển trên trục x. Trong trường hợp ngược lại, khi độ dài trên trục y lớn hơn trục x, công thức lỗi được thay đổi tương ứng để đảm bảo việc vẽ đường thẳng vẫn chính xác. Trong mỗi vòng lặp, hàm putpixel được gọi để vẽ một điểm tại tọa độ (x0, y0) với màu sắc đã chỉ định, và giá trị lỗi được cập nhật để tiếp tục quá trình vẽ đường thẳng cho đến khi đến điểm kết thúc.

|  |
| --- |
| *void drawLineBresenham(int x0, int y0, int x1, int y1, int color) {*  *int dx = abs(x1 - x0), dy = abs(y1 - y0);*  *int sx = (x0 < x1) ? 1 : -1;*  *int sy = (y0 < y1) ? 1 : -1;*  *if (dx >= dy) {*  *// Truong hop |m| <= 1*  *int err = 2 \* dy - dx;*  *for (int i = 0; i <= dx; ++i) {*  *putpixel(x0, y0, color);*  *if (err > 0) {*  *y0 += sy;*  *err -= 2 \* dx;*  *}*  *x0 += sx;*  *err += 2 \* dy;*  *}*  *} else {*  *// Truong hop m > 1*  *int err = 2 \* dx - dy;*  *for (int i = 0; i <= dy; ++i) {*  *putpixel(x0, y0, color);*  *if (err > 0) {*  *x0 += sx;*  *err -= 2 \* dy;*  *}*  *y0 += sy;*  *err += 2 \* dx;*  *}*  *}*  *}* |

***Hàm vẽ đường tròn bằng giải thuật Bresenham***

Hàm drawCircleBresenham sử dụng thuật toán Bresenham để vẽ một hình tròn với bán kính r và tâm tại tọa độ (x\_center, y\_center). Thuật toán Bresenham là một phương pháp hiệu quả để vẽ các hình tròn mà không cần sử dụng các phép tính phức tạp như căn bậc hai, mà thay vào đó chỉ sử dụng phép cộng, trừ và nhân. Thuật toán này giúp tối ưu hóa việc tính toán, làm cho quá trình vẽ nhanh chóng và tiết kiệm tài nguyên tính toán.

Trong hàm, quá trình bắt đầu bằng cách khởi tạo giá trị ban đầu cho các biến x, y và d. Sau đó, một vòng lặp được sử dụng để vẽ hình tròn. Mỗi điểm (x, y) được tính và vẽ, đồng thời thuật toán tận dụng tính đối xứng của hình tròn để vẽ các điểm đối xứng qua 8 phần của hình tròn. Nếu giá trị tham số d nhỏ hơn 0, thuật toán sẽ điều chỉnh vị trí của điểm tiếp theo theo phương ngang. Nếu d lớn hơn hoặc bằng 0, thuật toán sẽ điều chỉnh theo phương dọc và giảm giá trị y. Quá trình này tiếp tục cho đến khi x vượt qua giá trị y, lúc này hình tròn đã được vẽ hoàn thiện.

|  |
| --- |
| *void drawCircleBresenham(int x\_center, int y\_center, int r, int color) {*  *int x = 0, y = r, d = 3 - 2 \* r;*  *while (x <= y) {*  *putpixel(x\_center + x, y\_center + y, color);*  *putpixel(x\_center - x, y\_center + y, color);*  *putpixel(x\_center + x, y\_center - y, color);*  *putpixel(x\_center - x, y\_center - y, color);*  *putpixel(x\_center + y, y\_center + x, color);*  *putpixel(x\_center - y, y\_center + x, color);*  *putpixel(x\_center + y, y\_center - x, color);*  *putpixel(x\_center - y, y\_center - x, color);*  *if (d < 0)*  *d += 4 \* x + 6;*  *else {*  *d += 4 \* (x - y) + 10;*  *y--;*  *}*  *x++;*  *}*  *}* |

***Từ giải thuật sinh đường thẳng bằng thuật toán Bresenham có thể sinh ra các hình khác như hình vuông***

Hàm drawRectangleBresenham sử dụng thuật toán Bresenham để vẽ một hình chữ nhật với hai điểm góc đối diện (x1, y1) và (x2, y2) và màu sắc được chỉ định qua biến color. Hàm này thực hiện vẽ bốn cạnh của hình chữ nhật, mỗi cạnh được vẽ bằng cách gọi hàm drawLineBresenham đã được giải thích trước đó. Cạnh trên được vẽ từ (x1, y1) đến (x2, y1), cạnh bên phải được vẽ từ (x2, y1) đến (x2, y2), cạnh dưới được vẽ từ (x2, y2) đến (x1, y2) và cuối cùng, cạnh bên trái được vẽ từ (x1, y2) đến (x1, y1). Mỗi cạnh này được vẽ bằng thuật toán Bresenham, giúp việc vẽ hình chữ nhật trở nên chính xác và hiệu quả.

|  |
| --- |
| *void drawRectangleBresenham(int x1, int y1, int x2, int y2, int color) {*  *drawLineBresenham(x1, y1, x2, y1, color);*  *drawLineBresenham(x2, y1, x2, y2, color);*  *drawLineBresenham(x2, y2, x1, y2, color);*  *drawLineBresenham(x1, y2, x1, y1, color);*  *}* |

***Hàm vẽ đường tự do***

Hàm drawFreeformLine vẽ một đường tự do qua một dãy các điểm liên tiếp. Hàm nhận vào một vector các cặp tọa độ (x, y) lưu trữ các điểm trên đường và màu sắc color để vẽ. Đối với mỗi cặp điểm liên tiếp trong danh sách, hàm sẽ gọi hàm drawLineBresenham để vẽ một đoạn thẳng nối hai điểm đó. Quá trình này lặp lại cho đến khi hết các điểm trong vector. Cụ thể, hàm sử dụng vòng lặp for để duyệt qua tất cả các cặp điểm kế tiếp, với mỗi cặp điểm được dùng để vẽ một đoạn thẳng nối chúng, tạo thành một đường cong tự do.

|  |
| --- |
| *void drawFreeformLine(const vector<pair<int, int> >& points, int color) {*  *for (size\_t i = 0; i < points.size() - 1; ++i) {*  *int x0 = points[i].first;*  *int y0 = points[i].second;*  *int x1 = points[i + 1].first;*  *int y1 = points[i + 1].second;*  *drawLineBresenham(x0, y0, x1, y1, color);*  *}*  *}* |

***Hàm main***

Hàm main có chức năng giúp người dùng có thể chọn và vẽ các hình học trên giao diện đồ họa. Khi bắt đầu, chương trình khởi tạo môi trường đồ họa và vẽ giao diện chính. Khi người dùng nhấn chuột (với sự kiện WM\_LBUTTONDOWN), chương trình sẽ kiểm tra xem người dùng đã chọn hình nào để vẽ. Sau khi bắt đầu vẽ, chương trình theo dõi sự di chuyển của chuột và vẽ hình cho đến khi người dùng thả chuột (với sự kiện WM\_LBUTTONUP). Mỗi khi vẽ xong, hình sẽ được lưu vào danh sách và hiển thị lại. Nếu người dùng nhấn phím ESC, chương trình sẽ thoát. Toàn bộ quá trình này lặp lại cho đến khi người dùng thoát khỏi chương trình.

|  |
| --- |
| *int main() {*  *int gd = DETECT, gm;*  *initgraph(&gd, &gm, "");*  *drawInterface(); // ve giao dien*  *while (true) {*  *if (ismouseclick(WM\_LBUTTONDOWN)) {*  *if (shapeSelected == false) {*  *cleardevice();*  *drawInterface();*  *}*  *int x, y;*  *getmouseclick(WM\_LBUTTONDOWN, x, y);*  *handleMouseClick(x, y);*  *startX = x;*  *startY = y;*  *isDrawing = true;*  *clearmouseclick(WM\_LBUTTONDOWN);*  *if (shape == 6) {*  *Shape newShape;*  *newShape.type = shape;*  *newShape.color = currentColor;*  *newShape.points.clear();*  *newShape.points.push\_back({startX, startY});*  *shapes.push\_back(newShape);*  *}*  *}*    *if (ismouseclick(WM\_MOUSEMOVE) && isDrawing && shapeSelected) {*  *int x = mousex();*  *int y = mousey();*    *endX = x;*  *endY = y;*  *drawShape();*  *clearmouseclick(WM\_MOUSEMOVE);*  *}*  *if (ismouseclick(WM\_LBUTTONUP)) {*  *drawShape();*  *int x, y;*  *getmouseclick(WM\_LBUTTONUP, x, y);*  *endX = x;*  *endY = y;*  *isDrawing = false;*  *// luu hinh vao danh sach*  *if (shape == 5) { // ngoi sao*  *int radius = sqrt(pow(endX - startX, 2) + pow(endY - startY, 2));*  *shapes.push\_back({shape, startX, startY, 0, 0, radius, currentColor});*  *} else if (shape == 3) { // hinh tron*  *int radius = sqrt(pow(endX - startX, 2) + pow(endY - startY, 2));*  *shapes.push\_back({shape, startX, startY, 0, 0, radius, currentColor});*  *} else if (shape == 1 || shape == 2 || shape == 4) {*  *shapes.push\_back({shape, startX, startY, endX, endY, 0, currentColor});*  *}*  *// ve lai cac hinh*  *drawShape();*  *clearmouseclick(WM\_LBUTTONUP);*  *}*  *// Kiem tra nhan chuot de xoa duong tu do neu can*  *// kiem tra khi thoat*  *if (kbhit()) {*  *char key = getch();*  *if (key == 27) { // = nut ESC*  *break; // Thoát chuong trình*  *}*  *}*  *}*  *closegraph(); // dong cua so do hoa*  *return 0;*  *}* |

### Mức 2: Các chức năng của người dùng được tổ chức tốt, thân thiện, dễ sử dụng. Chức năng xóa hình

##### **Hướng xử lý**

* Phân chia các công cụ thành nhóm và sử dụng biểu tượng dễ hiểu. Cung cấp hướng dẫn khi cần.
* Cung cấp phản hồi ngay khi người dùng chọn công cụ hoặc thực hiện thao tác, ví dụ thay đổi con trỏ chuột.
* Sử dụng floodfill giúp tạo ra các ô màu đẹp mắt.
* Kiểm tra tọa độ điểm nhấn chuột có nằm trong hình không, nếu có thì xóa hình khỏi danh sách và vẽ lại các hình đã lưu.

##### **Chi tiết chương trình**

***Hàm vẽ giao diện***

Hàm drawInterface() vẽ giao diện người dùng cho chương trình vẽ đồ họa. Đầu tiên, nó tạo thanh công cụ ở phần trên của cửa sổ, bao gồm các nút bấm để vẽ các hình dạng cơ bản như đường thẳng, hình chữ nhật, hình tròn, hình tam giác, ngôi sao, và tự do. Mỗi nút bấm được tạo bằng các hình chữ nhật với hình vẽ tương ứng ở bên trong. Ngoài ra, nó còn tạo các ô màu để người dùng có thể chọn màu vẽ, và một nút để xóa hình vẽ. Cuối cùng, phần dưới giao diện là vùng vẽ, nơi người dùng có thể thực hiện các thao tác vẽ. Tất cả các đối tượng này được vẽ bằng các hàm đồ họa như rectangle(), floodfill(), và các hàm vẽ hình đã được định nghĩa.

|  |
| --- |
| *void drawInterface() {*  *setcolor(WHITE);*  *int drawingAreaHeight = getmaxy();*  *// Thanh công cu*  *rectangle(0, 0, getmaxx(), 60);*  *setfillstyle(SOLID\_FILL, LIGHTGRAY);*  *floodfill(20, 20, WHITE);*  *// Cac nut bam*  *// Nut ve duong thang*  *rectangle(10, 10, 100, 50);*  *drawLineBresenham(20, 30, 90, 30, WHITE);*  *// Nut ve hình chu nhat*  *rectangle(110, 10, 200, 50);*  *drawRectangleBresenham(120, 20, 190, 40, WHITE);*  *// Nut ve hình tròn*  *rectangle(210, 10, 300, 50);*  *drawCircleBresenham(255, 30, 15, WHITE);*  *// Nut ve hình tam giác*  *rectangle(310, 10, 400, 50);*  *int x1 = 355, y1 = 15, x2 = 325, y2 = 45, x3 = 385, y3 = 45;*  *drawLineBresenham(x1, y1, x2, y2, WHITE);*  *drawLineBresenham(x2, y2, x3, y3, WHITE);*  *drawLineBresenham(x3, y3, x1, y1, WHITE);*  *// Nut ve ngôi sao*  *rectangle(410, 10, 500, 50);*  *const int numPoints = 5;*  *int cx = 455, cy = 30, radius = 20;*  *drawStarBresenham(cx, cy, radius, WHITE);*  *// Nut ve tu do*  *rectangle(510, 10, 600, 50);*  *drawCurve(510, 30, 600, 130, 100, WHITE); // 100 di?m, màu tr?ng*  *// Nut cuc tay*  *rectangle(610, 10, 700, 50);*  *outtextxy(640, 20, "Erase");*  *// Cac o mau*  *// Dat cac o mau*  *setfillstyle(SOLID\_FILL, RED);*  *rectangle(710, 10, 750, 50);*  *floodfill(720, 20, WHITE);*  *setfillstyle(SOLID\_FILL, GREEN);*  *rectangle(760, 10, 800, 50);*  *floodfill(770, 20, WHITE);*  *setfillstyle(SOLID\_FILL, BLUE);*  *rectangle(810, 10, 850, 50);*  *floodfill(820, 20, WHITE);*  *setfillstyle(SOLID\_FILL, MAGENTA);*  *rectangle(860, 10, 900, 50);*  *floodfill(870, 20, WHITE);*  *setfillstyle(SOLID\_FILL, YELLOW);*  *rectangle(910, 10, 950, 50);*  *floodfill(920, 20, WHITE);*  *setfillstyle(SOLID\_FILL, CYAN);*  *rectangle(960, 10, 1000, 50);*  *floodfill(970, 20, WHITE);*  *setfillstyle(SOLID\_FILL, WHITE);*  *rectangle(1010, 10, 1050, 50);*  *floodfill(1020, 20, WHITE);*  *// Vung ve*  *rectangle(0, 60, getmaxx(), getmaxy());*  *}* |

***Code để xóa được hình***

Code thực hiện chức năng xóa các hình vẽ đã tồn tại trong danh sách shapes khi người dùng nhấn chuột gần một hình. Cụ thể, với mỗi hình trong danh sách, đoạn mã kiểm tra loại hình (được xác định bằng s.type), tính toán khoảng cách từ điểm chuột (endX, endY) đến hình vẽ đó, và nếu khoảng cách nhỏ hơn một ngưỡng nhất định (ví dụ, 5 pixel), hình vẽ sẽ bị xóa. Đoạn mã kiểm tra các loại hình như đường thẳng, hình chữ nhật, hình tròn, tam giác, ngôi sao, và đường tự do, và thực hiện xóa hình nếu điều kiện thỏa mãn. Sau khi xóa, giao diện sẽ được vẽ lại để cập nhật danh sách hình vẽ.

|  |
| --- |
| *for (int i = 0; i < shapes.size(); i++) {*  *Shape s = shapes[i];*  *// Line*  *if (s.type == 1) { // neu la duong thang*  *double dist = distanceFromPointToLine(endX, endY, s.x1, s.y1, s.x2, s.y2);*  *if (dist < 5) { // neu khoang cach nho hon 5 pixel => xoa duong thang*  *shapes.erase(shapes.begin() + i); // xoa hinh*  *cleardevice();*  *drawInterface();*  *drawShape();*  *break;*  *}*  *} else if (s.type == 2) { // Rectangle*  *if (endX >= s.x1 && endX <= s.x2 && endY >= s.y1 && endY <= s.y2) {*  *shapes.erase(shapes.begin() + i); // xoa hinh*  *cleardevice();*  *drawInterface();*  *drawShape();*  *break;*  *}*  *} else if (s.type == 3) { // Circle*  *int distance = sqrt(pow(s.x1 - endX, 2) + pow(s.y1 - endY, 2));*  *if (distance <= s.radius) {*  *shapes.erase(shapes.begin() + i);*  *cleardevice();*  *drawInterface();*  *drawShape(); // xoa hinh*  *break;*  *}*  *} else if (s.type == 4) { // Triangle*  *int size = abs(s.x2 - s.x1);*  *int height = static\_cast<int>(size \* sqrt(3) / 2);*  *int x1 = s.x1, y1 = s.y1 - height;*  *int x2 = s.x1 - size / 2, y2 = s.y1 + height / 2;*  *int x3 = s.x1 + size / 2, y3 = s.y1 + height / 2;*  *// kiem tra diem click chuot co nam trong tam giac khong*  *if (isPointInTriangle(endX, endY, x1, y1, x2, y2, x3, y3)) {*  *shapes.erase(shapes.begin() + i); // xoa tam giac*  *cleardevice();*  *drawInterface();*  *drawShape();*  *break;*  *}*  *} else if (s.type == 5) { // Star*  *if (isPointInStar(endX, endY, s.x1, s.y1, s.radius, s.radius / 2.5)) {*  *shapes.erase(shapes.begin() + i); // xoa hinh ngoi sao*  *cleardevice();*  *drawInterface();*  *drawShape();*  *break;*  *}*  *}*  *else if (s.type == 6) { // chon duong tu do*  *for (size\_t j = 0; j < s.points.size(); j++) {*  *int pointX = s.points[j].first; // lay toa do x*  *int pointY = s.points[j].second; // lay toa do y*  *// Tinh khoang cach tu diem chuot den diem trong duong tu do*  *int distance = sqrt(pow(pointX - endX, 2) + pow(pointY - endY, 2));*  *// Neu khoang cach nho hon 5 pixel, xoa toan bo duong tu do*  *if (distance < 5) {*  *shapes.erase(shapes.begin() + i); // xóa toàn bo duong tu do*  *cleardevice();*  *drawInterface();*  *drawShape();*  *break;*  *}*  *}*  *}*  *}* |

# CÀI ĐẶT CHƯƠNG TRÌNH

## CODE CHƯƠNG TRÌNH

### Mức 1

##### **Code các hàm sinh hình**

|  |
| --- |
| *#include <conio.h>*  *#include <graphics.h>*  *#include <cmath>*  *#include <iostream>*  *#include <vector>*  *using namespace std;*  *int startX, startY, endX, endY;*  *bool isDrawing = false;*  *int shape = 0; // luu hinh ve duoc chon*  *bool shapeSelected = false; // ban dau chua chon hình*  *int currentColor = WHITE; // mau ban dau la trang*  *struct Shape {*  *int type; // loai hinh*  *int x1, y1, x2, y2; // toa do dung cho duong thang, hinh vuong*  *int radius; // ban kinh dung cho hinh ngoi sao, tron*  *int color; // mau*  *vector<pair<int, int> > points; // luu cac diem cua duong tu do*  *};*  *vector<Shape> shapes;*  *void drawLineBresenham(int x0, int y0, int x1, int y1, int color) {*  *int dx = abs(x1 - x0), dy = abs(y1 - y0);*  *int sx = (x0 < x1) ? 1 : -1;*  *int sy = (y0 < y1) ? 1 : -1;*  *if (dx >= dy) {*  *// Truong hop |m| <= 1*  *int err = 2 \* dy - dx;*  *for (int i = 0; i <= dx; ++i) {*  *putpixel(x0, y0, color);*  *if (err > 0) {*  *y0 += sy;*  *err -= 2 \* dx;*  *}*  *x0 += sx;*  *err += 2 \* dy;*  *}*  *} else {*  *// Truong hop m > 1*  *int err = 2 \* dx - dy;*  *for (int i = 0; i <= dy; ++i) {*  *putpixel(x0, y0, color);*  *if (err > 0) {*  *x0 += sx;*  *err -= 2 \* dy;*  *}*  *y0 += sy;*  *err += 2 \* dx;*  *}*  *}*  *}*  *// Ham ve duong tu do noi cac diem lien tiep*  *void drawFreeformLine(const vector<pair<int, int> >& points, int color) {*  *for (size\_t i = 0; i < points.size() - 1; ++i) {*  *int x0 = points[i].first;*  *int y0 = points[i].second;*  *int x1 = points[i + 1].first;*  *int y1 = points[i + 1].second;*  *drawLineBresenham(x0, y0, x1, y1, color);*  *}*  *}*  *void drawRectangleBresenham(int x1, int y1, int x2, int y2, int color) {*  *drawLineBresenham(x1, y1, x2, y1, color);*  *drawLineBresenham(x2, y1, x2, y2, color);*  *drawLineBresenham(x2, y2, x1, y2, color);*  *drawLineBresenham(x1, y2, x1, y1, color);*  *}*  *void drawCircleBresenham(int x\_center, int y\_center, int r, int color) {*  *int x = 0, y = r, d = 3 - 2 \* r;*  *while (x <= y) {*  *putpixel(x\_center + x, y\_center + y, color);*  *putpixel(x\_center - x, y\_center + y, color);*  *putpixel(x\_center + x, y\_center - y, color);*  *putpixel(x\_center - x, y\_center - y, color);*  *putpixel(x\_center + y, y\_center + x, color);*  *putpixel(x\_center - y, y\_center + x, color);*  *putpixel(x\_center + y, y\_center - x, color);*  *putpixel(x\_center - y, y\_center - x, color);*  *if (d < 0)*  *d += 4 \* x + 6;*  *else {*  *d += 4 \* (x - y) + 10;*  *y--;*  *}*  *x++;*  *}*  *}*  *void drawStarBresenham(int x\_center, int y\_center, int radius, int color) {*  *const int numPoints = 5;*  *double angle = M\_PI / numPoints;*  *int x[numPoints], y[numPoints];*  *for (int i = 0; i < numPoints; i++) {*  *x[i] = x\_center + radius \* cos(i \* 2 \* angle - M\_PI / 2);*  *y[i] = y\_center + radius \* sin(i \* 2 \* angle - M\_PI / 2);*  *}*  *for (int i = 0; i < numPoints; i++) {*  *int next = (i + 2) % numPoints;*  *drawLineBresenham(x[i], y[i], x[next], y[next], color);*  *}*  *}*  *// Hàm ve duong cong parabol don gian*  *void drawCurve(int startX, int startY, int endX, int endY, int numPoints, int color) {*  *vector<pair<int, int> > points;*  *// Tao cac diem doc theo mot duong cong parabol*  *for (int i = 0; i < numPoints; ++i) {*  *int x = startX + (endX - startX) \* i / (numPoints - 1); // Phân ph?i d?u các di?m theo chi?u ngang*  *int y = startY + (x - startX) \* (x - endX) / (numPoints \* 2); // Công th?c t?o du?ng cong parabol*  *points.push\_back({x, y});*  *}*  *// Ve duong cong*  *drawFreeformLine(points, color);*  *}* |

### Mức 2

##### **Code giao diện**

|  |
| --- |
| *void drawInterface() {*  *setcolor(WHITE);*  *int drawingAreaHeight = getmaxy();*  *// Thanh công cu*  *rectangle(0, 0, getmaxx(), 60);*  *setfillstyle(SOLID\_FILL, LIGHTGRAY);*  *floodfill(20, 20, WHITE);*  *// Cac nut bam*  *// Nut ve duong thang*  *rectangle(10, 10, 100, 50);*  *drawLineBresenham(20, 30, 90, 30, WHITE);*  *// Nut ve hình chu nhat*  *rectangle(110, 10, 200, 50);*  *drawRectangleBresenham(120, 20, 190, 40, WHITE);*  *// Nut ve hình tròn*  *rectangle(210, 10, 300, 50);*  *drawCircleBresenham(255, 30, 15, WHITE);*  *// Nut ve hình tam giác*  *rectangle(310, 10, 400, 50);*  *int x1 = 355, y1 = 15, x2 = 325, y2 = 45, x3 = 385, y3 = 45;*  *drawLineBresenham(x1, y1, x2, y2, WHITE);*  *drawLineBresenham(x2, y2, x3, y3, WHITE);*  *drawLineBresenham(x3, y3, x1, y1, WHITE);*  *// Nut ve ngôi sao*  *rectangle(410, 10, 500, 50);*  *const int numPoints = 5;*  *int cx = 455, cy = 30, radius = 20;*  *drawStarBresenham(cx, cy, radius, WHITE);*  *// Nut ve tu do*  *rectangle(510, 10, 600, 50);*  *drawCurve(510, 30, 600, 130, 100, WHITE); // 100 di?m, màu tr?ng*  *// Nut cuc tay*  *rectangle(610, 10, 700, 50);*  *outtextxy(640, 20, "Erase");*  *// Cac o mau*  *// Dat cac o mau*  *setfillstyle(SOLID\_FILL, RED);*  *rectangle(710, 10, 750, 50);*  *floodfill(720, 20, WHITE);*  *setfillstyle(SOLID\_FILL, GREEN);*  *rectangle(760, 10, 800, 50);*  *floodfill(770, 20, WHITE);*  *setfillstyle(SOLID\_FILL, BLUE);*  *rectangle(810, 10, 850, 50);*  *floodfill(820, 20, WHITE);*  *setfillstyle(SOLID\_FILL, MAGENTA);*  *rectangle(860, 10, 900, 50);*  *floodfill(870, 20, WHITE);*  *setfillstyle(SOLID\_FILL, YELLOW);*  *rectangle(910, 10, 950, 50);*  *floodfill(920, 20, WHITE);*  *setfillstyle(SOLID\_FILL, CYAN);*  *rectangle(960, 10, 1000, 50);*  *floodfill(970, 20, WHITE);*  *setfillstyle(SOLID\_FILL, WHITE);*  *rectangle(1010, 10, 1050, 50);*  *floodfill(1020, 20, WHITE);*  *// Vung ve*  *rectangle(0, 60, getmaxx(), getmaxy());*  *}* |

**Code kiểm tra điểm nhấn chuột có nằm trong hình không**

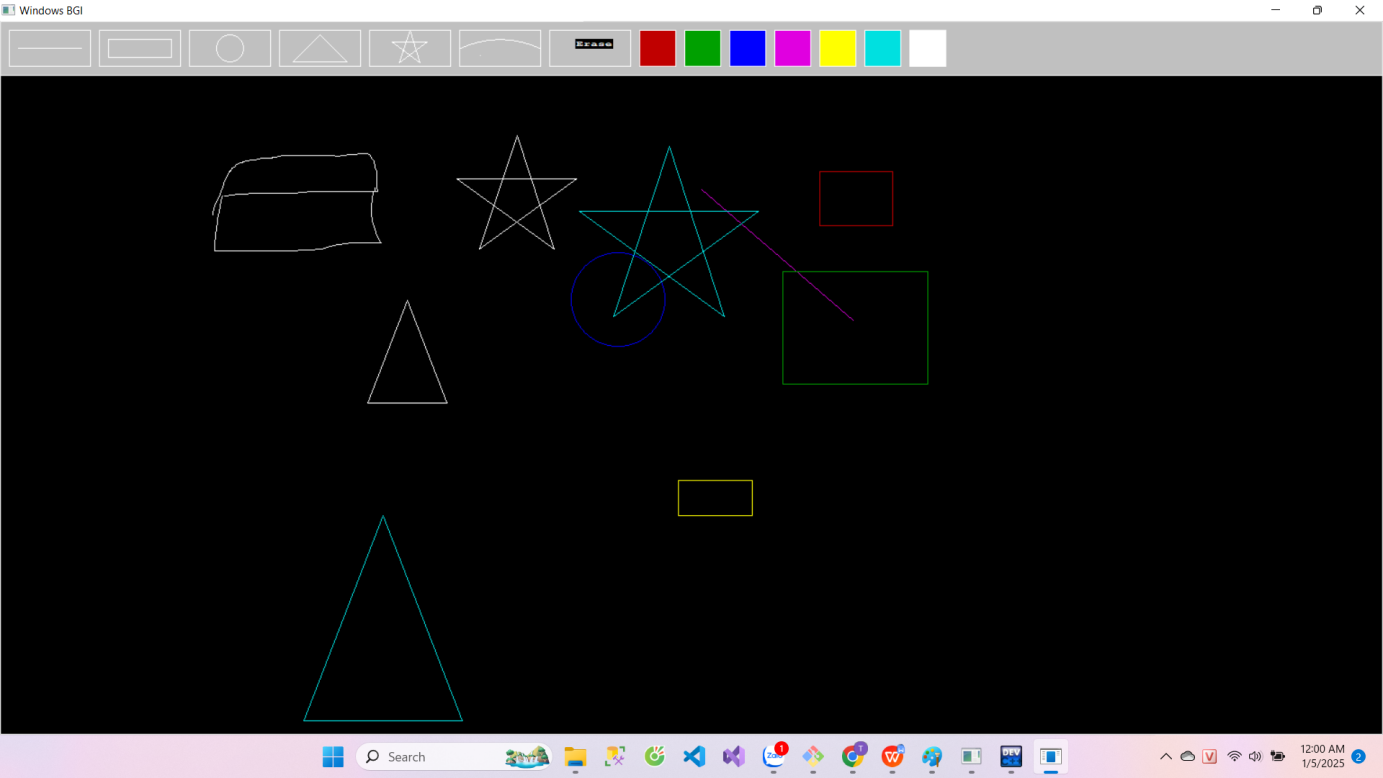
|  |
| --- |
| *bool isPointInTriangle(int px, int py, int x1, int y1, int x2, int y2, int x3, int y3) {*  *// Tinh S tam giac*  *int areaOrig = abs(x1 \* (y2 - y3) + x2 \* (y3 - y1) + x3 \* (y1 - y2));*  *// Tinh S 3 tam giac nho*  *int area1 = abs(px \* (y2 - y3) + x2 \* (y3 - py) + x3 \* (py - y2));*  *int area2 = abs(x1 \* (py - y3) + px \* (y3 - y1) + x3 \* (y1 - py));*  *int area3 = abs(x1 \* (y2 - py) + x2 \* (py - y1) + px \* (y1 - y2));*  *// Kiem tra dien tich cua 3 tam giac nho co bang tam giac lon*  *return (area1 + area2 + area3 == areaOrig);*  *}*  *bool isPointInStar(int px, int py, int cx, int cy, int radius\_outer, int radius\_inner) {*  *const int numPoints = 5; // So dinh cua ngôi sao*  *double angle = M\_PI / numPoints;*  *int x[2 \* numPoints], y[2 \* numPoints];*  *// Tinh toa do các dinh cua ngoi sao*  *for (int i = 0; i < 2 \* numPoints; i++) {*  *double r = (i % 2 == 0) ? radius\_outer : radius\_inner;*  *x[i] = cx + r \* cos(i \* angle - M\_PI / 2);*  *y[i] = cy + r \* sin(i \* angle - M\_PI / 2);*  *}*  *// Kiem tra tung tam giac tao boi tam và các canh cua ngoi sao*  *for (int i = 0; i < 2 \* numPoints; i++) {*  *int next = (i + 1) % (2 \* numPoints);*  *if (isPointInTriangle(px, py, cx, cy, x[i], y[i], x[next], y[next])) {*  *return true;*  *}*  *}*  *return false; // Neu không nam trong tam giác nào, tra ve false*  *}*  *double distanceFromPointToLine(int px, int py, int x1, int y1, int x2, int y2) {*  *// Khoang cach tu diem den duong thang*  *double num = abs((y2 - y1) \* px - (x2 - x1) \* py + x2 \* y1 - y2 \* x1);*  *double den = sqrt(pow(y2 - y1, 2) + pow(x2 - x1, 2));*  *return num / den;*  *}*  *double distanceBetweenPoints(int x1, int y1, int x2, int y2) { return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2)); }* |

**Code xóa hình**

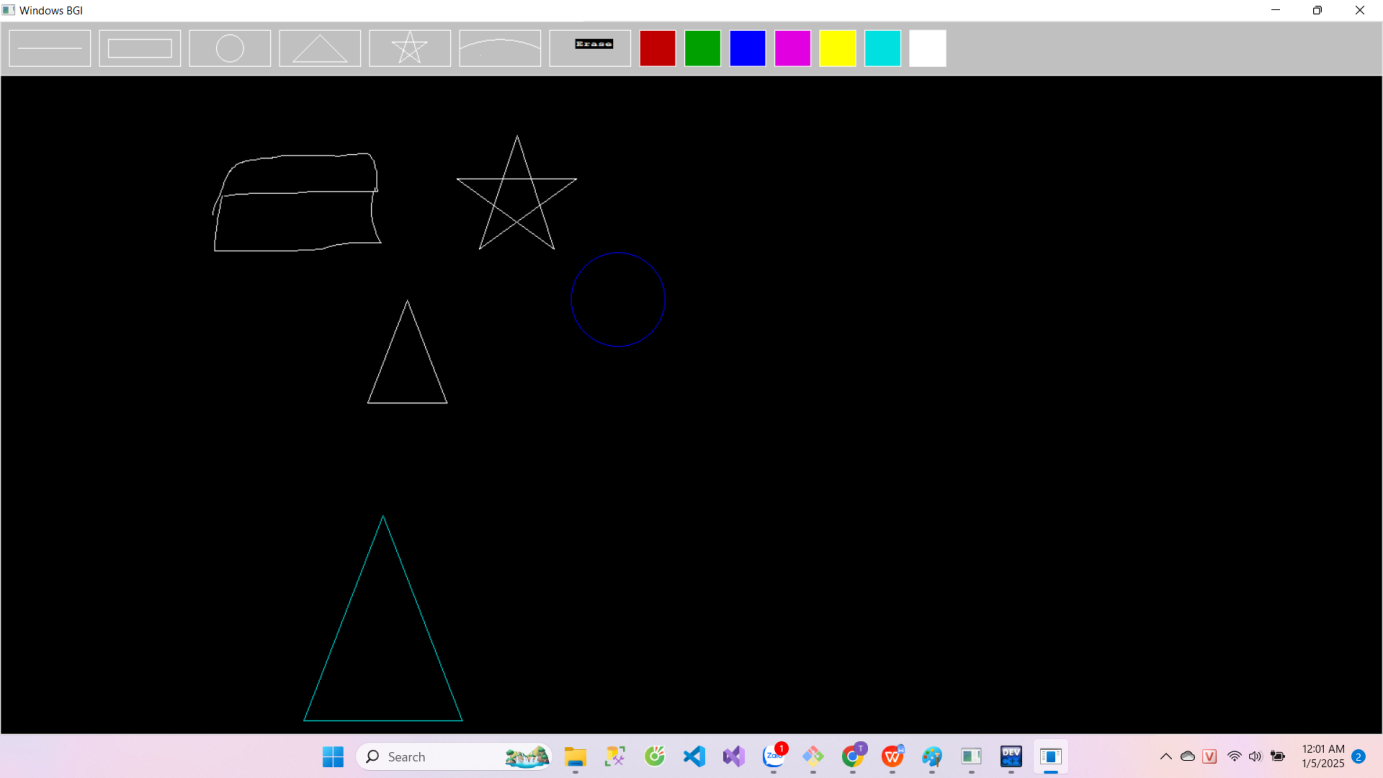
|  |
| --- |
| *for (int i = 0; i < shapes.size(); i++) {*  *Shape s = shapes[i];*  *// Line*  *if (s.type == 1) { // neu la duong thang*  *double dist = distanceFromPointToLine(endX, endY, s.x1, s.y1, s.x2, s.y2);*  *if (dist < 5) { // neu khoang cach nho hon 5 pixel => xoa duong thang*  *shapes.erase(shapes.begin() + i); // xoa hinh*  *cleardevice();*  *drawInterface();*  *drawShape();*  *break;*  *}*  *} else if (s.type == 2) { // Rectangle*  *if (endX >= s.x1 && endX <= s.x2 && endY >= s.y1 && endY <= s.y2) {*  *shapes.erase(shapes.begin() + i); // xoa hinh*  *cleardevice();*  *drawInterface();*  *drawShape();*  *break;*  *}*  *} else if (s.type == 3) { // Circle*  *int distance = sqrt(pow(s.x1 - endX, 2) + pow(s.y1 - endY, 2));*  *if (distance <= s.radius) {*  *shapes.erase(shapes.begin() + i);*  *cleardevice();*  *drawInterface();*  *drawShape(); // xoa hinh*  *break;*  *}*  *} else if (s.type == 4) { // Triangle*  *int size = abs(s.x2 - s.x1);*  *int height = static\_cast<int>(size \* sqrt(3) / 2);*  *int x1 = s.x1, y1 = s.y1 - height;*  *int x2 = s.x1 - size / 2, y2 = s.y1 + height / 2;*  *int x3 = s.x1 + size / 2, y3 = s.y1 + height / 2;*  *// kiem tra diem click chuot co nam trong tam giac khong*  *if (isPointInTriangle(endX, endY, x1, y1, x2, y2, x3, y3)) {*  *shapes.erase(shapes.begin() + i); // xoa tam giac*  *cleardevice();*  *drawInterface();*  *drawShape();*  *break;*  *}*  *} else if (s.type == 5) { // Star*  *if (isPointInStar(endX, endY, s.x1, s.y1, s.radius, s.radius / 2.5)) {*  *shapes.erase(shapes.begin() + i); // xoa hinh ngoi sao*  *cleardevice();*  *drawInterface();*  *drawShape();*  *break;*  *}*  *}*  *else if (s.type == 6) { // chon duong tu do*  *for (size\_t j = 0; j < s.points.size(); j++) {*  *int pointX = s.points[j].first; // lay toa do x*  *int pointY = s.points[j].second; // lay toa do y*  *// Tinh khoang cach tu diem chuot den diem trong duong tu do*  *int distance = sqrt(pow(pointX - endX, 2) + pow(pointY - endY, 2));*  *// Neu khoang cach nho hon 5 pixel, xoa toan bo duong tu do*  *if (distance < 5) {*  *shapes.erase(shapes.begin() + i); // xóa toàn bo duong tu do*  *cleardevice();*  *drawInterface();*  *drawShape();*  *break;*  *}*  *}*  *}*  *}* |

### **Kết quả chạy thử chương trình**

Chức năng vẽ hình với giao diện dễ sử dụng



Chức năng xóa hình



# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## KẾT QUẢ ĐẠT ĐƯỢC

* + Có thể vẽ được các hình cơ bản với giao diện dễ sử dụng.
  + Có thể xóa được hình đơn giản.

## ƯU ĐIỂM

* + Giao diện của chương trình được thiết kế đơn giản, dễ thao tác, giúp người dùng nhanh chóng làm quen và sử dụng hiệu quả.
  + Chương trình cung cấp nền tảng cơ bản về thuật toán đồ họa, từ đó có thể mở rộng và áp dụng vào các bài toán phức tạp hơn, hỗ trợ phát triển các ứng dụng đồ họa chuyên sâu.

## HẠN CHẾ

* Khi chạy trên các hệ thống có card đồ họa yếu, việc vẽ hình có thể bị nhấp nháy hoặc không mượt mà, ảnh hưởng đến trải nghiệm người dùng và độ chính xác của hình ảnh..
* Code chương trình còn dài, còn nhiều chỗ chưa được tối ưu.

## HƯỚNG PHÁT TRIỂN

* Hệ thống sẽ được hoàn thiện thêm các chức năng hiện có, tối ưu giao diện và trải nghiệm người dùng. Cải thiện hiệu suất, giảm nhấp nháy khi vẽ trên phần cứng yếu, đồng thời tối ưu mã nguồn để tăng tốc độ xử lý. Tích hợp thêm các phép biến hình trên không gian 2D và 3D như tịnh tiến, quay, co giãn, phản chiếu, và các phép chiếu phối cảnh.

## KẾT LUẬN

Chương trình xây dựng các thuật toán đồ họa đã mang lại nền tảng quan trọng để thực hiện các thao tác vẽ hình 2D và 3D, cung cấp các công cụ cần thiết cho việc ứng dụng đồ họa trong thực tiễn. Mặc dù còn tồn tại một số hạn chế về giao diện và hiệu năng, chương trình đã đáp ứng được mục tiêu ban đầu, đồng thời mở ra hướng phát triển mới trong việc tối ưu và bổ sung các phép biến hình phức tạp, nâng cao trải nghiệm người dùng và hiệu quả sử dụng.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Nguyễn Thế Anh, **Học lập trình đồ họa với OpenGL và C++**, Nhà xuất bản Bách Khoa, 2020. |
| [2] | Wikipedia tiếng Việt, “Đồ họa máy tính”, 01/01/2025. [Trực tuyến]. Available: [https://vi.wikipedia.org/wiki/Đồ\_họa\_máy\_tính](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93_h%E1%BB%8Da_m%C3%A1y_t%C3%ADnh" \t "_new). [Truy cập: 01/01/2025]. |
| [3] | Wikipedia tiếng Việt, “Thuật toán Bresenham”, 01/01/2025. [Trực tuyến]. Available: [https://vi.wikipedia.org/wiki/Thuật\_toán\_Bresenham](https://vi.wikipedia.org/wiki/Thu%E1%BA%ADt_to%C3%A1n_Bresenham" \t "_new). [Truy cập: 01/01/2025]. |
| [4] | **Đoàn Vũ Thịnh**, **Kỹ thuật đồ họa**, Khóa học Đồ họa máy tính, Đại học Nha Trang, 2022. |
| [5] | Nguyễn Thị Lan, **Thuật toán vẽ đồ họa trong C++**, Bài giảng môn Đồ họa máy tính, Đại học Công nghệ Thông tin TP.HCM, 2020. |
| [6] | **Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F.** Computer Graphics: Principles and Practice, 3rd edition, Addison-Wesley, 2013. |