**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

****

**BÁO CÁO ĐỒ ÁN**

**VẼ VÀ TÔ MÀU ĐỐI TƯỢNG 2D**

**Môn học: Đồ họa máy tính**

Sinh viên thực hiện:Dũ Quốc Huy – 20120101

Giáo viên hướng dẫn: Trần Thái Sơn

Võ Hoài Việt

Thành phố Hồ Chí Minh, ngày 15 tháng 11 năm 2022

Mục lục

[1. Tổng quan về thiết kế chương trình 3](#_Toc119432869)

[2. Các hàm xử lý sự kiện chuột 4](#_Toc119432870)

[2.1. Xử lý sự kiện chuột cho việc vẽ hình 4](#_Toc119432871)

[2.2. Xử lý sự kiện chuột cho việc tô màu 4](#_Toc119432872)

[3. Thuật toán vẽ hình 4](#_Toc119432873)

[3.1. Thuật toán vẽ đường thẳng: 5](#_Toc119432874)

[3.2. Vẽ hình tam giác 5](#_Toc119432875)

[3.2.1. Tam giác vuông cân 5](#_Toc119432876)

[3.2.2. Tam giác đều 6](#_Toc119432877)

[3.3. Hình tứ giác 6](#_Toc119432878)

[3.3.1. Hình chữ nhật 6](#_Toc119432879)

[3.3.2. Hình vuông 7](#_Toc119432880)

[3.4. Hình oval 8](#_Toc119432881)

[3.4.1. Hình tròn 8](#_Toc119432882)

[3.4.2. Hình elip 8](#_Toc119432883)

[3.5. Hình đa giác đều 9](#_Toc119432884)

[3.5.1. Ngũ giác đều 9](#_Toc119432885)

[3.5.2. Lục giác đều 10](#_Toc119432886)

[3.6. Hình khác 11](#_Toc119432887)

[3.6.1. Mũi tên 11](#_Toc119432888)

[3.6.2. Ngôi sao 12](#_Toc119432889)

[3.7. Dấu 13](#_Toc119432890)

[3.7.1. Dấu cộng 13](#_Toc119432891)

[3.7.2. Dấu trừ: 14](#_Toc119432892)

[3.7.3. Dấu nhân 14](#_Toc119432893)

[3.7.4. Dấu chia 15](#_Toc119432894)

[4. Thuật toán tô màu 16](#_Toc119432895)

[5. Hướng dẫn sử dụng 18](#_Toc119432896)

[6. Tài liệu tham khảo 19](#_Toc119432897)

1. Tổng quan về thiết kế chương trình

Nội dung đồ án: sử dụng thư viện đồ họa GLUT để vẽ và tô màu các đối tượng 2D theo như Menu bên dưới.

Vẽ bao gồm: đường thẳng, tam giác vuông cân, tam giác đều, hình chữ nhật, hình vuông, hình tròn, hình elip, ngũ giác đều, lục giác đều, mũi tên, ngôi sao, các dấu cộng trừ nhân chia và thực hiện tô màu xanh, đỏ hoặc vàng dựa theo đường bao.

Diagram

Description automatically generated

Tổ chức mã nguồn:

* Mã nguồn sẽ bao gồm 1 tập tin main.cpp để xử lí việc hiển thị giao diện GLUT.
* Nhiều tập tin có đuôi .h gồm:

+ Header.h :khai báo thư viện, define các giá trị cần cho bài.

+ SupportDrawPaint.h :chứa các hàm hỗ trợ việc vẽ, put pixel lên màn hình, khởi tạo màu và lấy thời gian.

+ Menu.h :chứa các hàm hỗ trợ việc hiển thị Menu và kiểm tra giá trị value tương ứng với việc mà người dùng muốn thực hiện.

+ Draw.h :chứa các hàm thực hiện vẽ các hình tương ứng và xử lý sự kiện chuột dành cho việc vẽ hình.

+ Paint.h :chứa các hàm thực hiện tô màu và xử lý sự kiện chuột dành cho việc tô màu.

* Chương trình xử dụng một cấu trúc dữ liệu map với tên pixelNotWhite với key là tọa độ, value là những giá trị màu (1: đen, 2: green, 3: red, 4: yellow). Trong quá trình nhấn vẽ hình hoặc tô màu xong, chương trình sẽ duyệt toàn bộ map và put các pixel tương ứng lên màn hình.

Khi người dùng chọn một hình để vẽ/ màu để tô, value sẽ lưu giá trị tương ứng được quy định sẵn trong Menu.h, từ đó tìm đúng lệnh để vẽ/ tô.

Trong quá trình vẽ hình mới, các hình cũ và màu đã tô vẫn xuất hiện trên màn hình. Giúp người dùng dễ dàng hình dùng được hình đang vẽ và các hình trước đó, dễ dàng kết hợp các hình với nhau hơn.

1. Các hàm xử lý sự kiện chuột
   1. Xử lý sự kiện chuột cho việc vẽ hình

Việc bắt chuột cho việc vẽ hình bao gồm 2 lần nhấn chuột trái để thực hiện vẽ sau khi đã chọn hình từ Menu:

* Lần nhấn chuột trái đầu tiên: giá trị check sẽ chuyển từ 0 sang 1 (không có hình để vẽ sang chế độ đang vẽ), chương trình lưu lại tọa độ vị trí nhấn chuột (X1, Y1).
* Sau lần nhấn chuột đầu tiên, hàm myMotion với chức năng lấy tọa độ di chuột (X2, Y2) để vẽ hình minh họa cho người dùng thấy.
* Lần nhấn chuột trái thứ 2: giá trị check sẽ chuyển từ 1 sang 2 (đang vẽ chuyển sang chế độ đã vẽ xong hình), chương trình lưu lại tọa độ vị trí nhấn chuột (X2, Y2) cho lần cuối này và xuất hình cuối cùng lên màn hình đồng thời lưu lại những Pixel đã bị thay đổi màu cho hình vẽ này.
* Sau khi vẽ xong một hình mà vẫn chưa chọn hình khác thì chương trình vẫn giữ nguyên giá trị value của hình cũ, có thể thực hiện vẽ hình đó nhiều lần mà không cần phải chọn lại từ Menu.
  1. Xử lý sự kiện chuột cho việc tô màu

Khi chọn được màu tô từ Menu, chỉ cần nhấn chuột trái vào một vùng nào đó chương trình sẽ tự động tô màu vùng đó với điều kiện pixel tại đó khác màu viền hoặc màu được chọn. Sau khi tô xong sẽ duyệt toàn bộ pixelNotWhite để hiện thị các pixel khác màu trắng lên màn hình và lấy thời gian tô màu đó.

1. Thuật toán vẽ hình

Dựa vào 2 điểm (X1, Y1) và (X2, Y2) đã lấy được ở mỗi lần di chuột trong thời gian giữa 2 lần nhấn chuột trái vẽ hình để tính tọa ra tọa độ các đỉnh của từng hình tương ứng.

* 1. Thuật toán vẽ đường thẳng:

Ý tưởng: đường thẳng được vẽ bằng thuật toán DDA (Digital Differential Analyzer). Với tham số đầu vào là tọa độ điểm đầu và cuối.

Các hình sau cũng sử dụng việc vẽ đường thẳng này để vẽ các hình tương ứng

Kết quả minh họa:

Engineering drawing

Description automatically generated with medium confidence

* 1. Vẽ hình tam giác
     1. Tam giác vuông cân

Ý tưởng: với tam giác vuông cân, ta dựa trên 2 điểm bắt chuột được để tính toán dx và dy rồi lấy giá trị nhỏ hơn làm cạnh bên. Với (X1, Y1) là tọa độ đỉnh khác góc vuông, từ đó tính toán ra 2 giá trị đỉnh còn lại và thực hiện vẽ 3 đường thẳng dựa theo tọa độ đỉnh tìm được.

Kết quả minh họa:

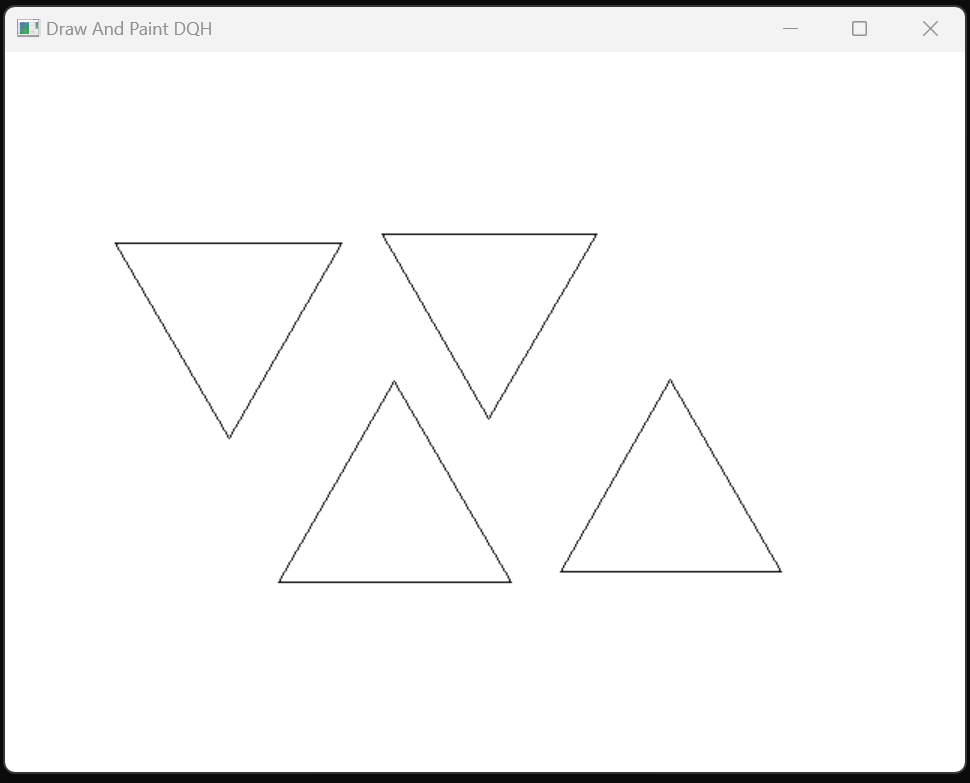
Engineering drawing

Description automatically generated with low confidence

* + 1. Tam giác đều

Ý tưởng: tương tự như tam giác vuông cân nhưng cạnh sẽ được lấy dựa theo giá trị dy \* 2 / sqrt(3) hoặc dx. (X1, X2) sẽ là tọa độ một đỉnh của tam giác đều, với mỗi trường hợp lấy cạnh ta tính toán để lấy được tọa độ 2 điểm còn lại và thực hiện vẽ 3 đường thẳng nối các đỉnh đó với nhau.

Kết quả minh họa:



* 1. Hình tứ giác
     1. Hình chữ nhật

Ý tưởng: Với tọa độ 2 điểm (X1, Y1) và (X2, Y2) mà ta bắt được, ta tìm được 4 đỉnh của hình chữ nhật là (X1, Y1) (X2, Y1), (X2, Y2) và (X1, Y2). Vẽ 4 đường thẳng để nối 4 đỉnh đó với nhau.

Kết quả minh họa:

A picture containing graphical user interface

Description automatically generated

* + 1. Hình vuông

Ý tưởng: ta tìm cạnh hình vuông theo giá trị bé hơn của abs(dx) và abs(dy). Với mỗi trường hợp, ta cũng tìm lại 4 điểm tương ứng của hình vuông và sau đó vẽ 4 đường thẳng nối 4 đỉnh đó với nhau.

Kết quả thực hiện:

Shape, square

Description automatically generated

* 1. Hình oval
     1. Hình tròn

Ý tưởng: với 2 tọa độ bắt chuột được, ta chuyển 2 tọa độ đó về dạng hình vuông (tương tự như hình vuông). Sau đó tìm tâm hình tròn bằng cách lấy trung điểm X1, X3 và Y1, Y3. Bán kính sẽ bằng tạo độ tâm trừ đi (X1, Y1). Từ đó sử dụng thuật toán Midpoint để vẽ hình tròn với tham số đầu vào là tọa độ tâm và bán kính.

Kết quả thực hiện:

Diagram, venn diagram

Description automatically generated

* + 1. Hình elip

Ý tưởng: với 2 tọa độ bắt chuột được, thực hiện tìm tọa độ tâm của hình elip (tâm hình chữ nhật) và ½ trục lớn (abs(xT – X1)) và ½ trục nhỏ (abs(yT – Y1). Thực hiên vẽ hình elip bằng thuật toán Midpoint với tham số đầu vào là tọa độ tâm và ½ trục lớn và ½ trục nhỏ. Hình elip chỉ được vẽ khi trục lớn và trục nhỏ lớn hơn 0.

Kết quả minh họa:

Diagram, venn diagram

Description automatically generated

* 1. Hình đa giác đều
     1. Ngũ giác đều

Ý tưởng: từ tọa độ 2 điểm bắt được (X1, Y1) và (X2, Y2), ta dựng được hình vuông với cạnh = min(dx, dy). Từ đó tìm tọa độ các đỉnh của ngũ giác đều nội tiếp hình tròn nội tiếp hình vuông đó dựa theo các thông số góc đã biết và bán hình = ½ cạnh hình vuông. Nhận thấy các tọa độ các sự đối xứng của trục dọc xT. Minh họa ý tưởng tạo nên ngũ giác đều:

Shape

Description automatically generated

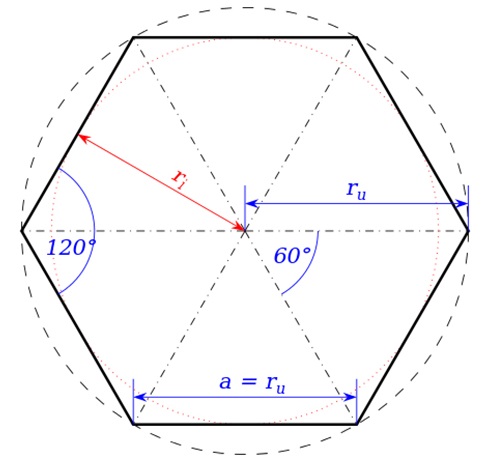
Kết quả thực hiện:

Shape, polygon

Description automatically generated

* + 1. Lục giác đều

Ý tưởng: Tương tự như ngũ giác đều nhưng đơn giản hơn. Các đỉnh của lục giác đều có sự đối xứng qua trục dọc xT và trục ngang yT (tâm). Dựa vào các góc ở mỗi đỉnh và tọa độ tâm, bán kính hình tròn lớn để xác định các tọa độ đỉnh của hình lục giác đều



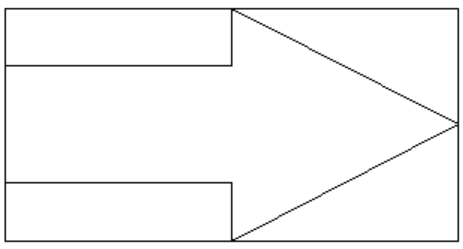
Kết quả thực hiện:

A picture containing shape

Description automatically generated

* 1. Hình khác
     1. Mũi tên

Ý tưởng: với mũi tên ta sẽ chia dx làm 2 phần, 1 nữa là thân mũi tên (chiều rộng là ½ dy), phần còn lại là đầu nhọn. Ở đây, ta chỉ vẽ mũi tên hướng sang trái hoặc phải. Với tọa độ 2 điểm (X1, Y1) và (X2, Y2) bắt chuột được, ta tính toán giá trị các đỉnh của mũi tên và sử dụng vẽ đường thẳng để nối các đỉnh với nhau. Minh họa ý tưởng:



Ta chỉ sử dụng những điểm cần thiết để vẽ hình mũi tên.

Kết quả minh họa:

Shape, arrow

Description automatically generated

* + 1. Ngôi sao

Ý tưởng: ngôi sao sẽ bao gồm 5 đỉnh ở ngoài nội tiếp trong một đường tròn lớn và 5 đỉnh trong nội tiếp trong một đường tròn nhỏ đồng tâm. Việc xác định tọa độ các đỉnh này tương tự như xác định đỉnh trong ngũ giác đều. sau khi có được tạo độ các đỉnh cần thiết, ta sử dụng vẽ đường thẳng để nối các đỉnh lại với nhau. Minh họa ý tưởng:

Shape

Description automatically generated

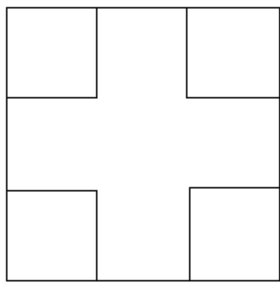
Kết quả thực hiện:

Shape, polygon

Description automatically generated

* 1. Dấu
     1. Dấu cộng

Ý tưởng: trước tiên đưa tọa độ 2 điểm bắt được về dạng một hình vuông (tọa độ 2 góc đối nhau của hình vuông), với mỗi cạnh hình vuông ta chia tỉ lệ 1:1:1 với phần ở giữa là cạnh rìa của dấu cộng. Từ đó xác định được các đỉnh cần thiết để tạo ra hình vuông và nối lại với nhau bằng vẽ đường thẳng. Quan sát kết quả thực hiện để dễ hình dung. Minh họa ý tưởng:



Ta chỉ sử dụng những đỉnh cần thiết cho hình dấu cộng

Kết quả thực hiện:

Box and whisker chart

Description automatically generated

* + 1. Dấu trừ:

Ý tưởng: vẽ dấu trừ tương tự như vẽ hình chữ nhật, nhưng để dấu trừ đẹp hơn thì ta giới hạn dy là 1/3 dx. Thực hiện nối các đỉnh tương ứng để hoàn thiện dấu trừ.

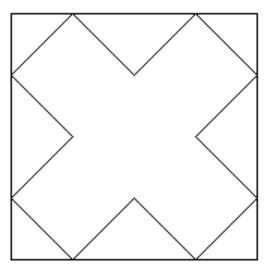
Kết quả thực hiện:

Rectangle

Description automatically generated with low confidence

* + 1. Dấu nhân

Ý tưởng: dấu nhân tương tự như dấu cộng, tuy nhiên ở mỗi cạnh ta sẽ chia tỉ lệ 1:1:1:1 rồi nối các điểm tương ứng lại để tạo ra hình dấu nhân. Minh họa ý tưởng:



Ta chỉ sử dụng những đỉnh cần thiết cho dấu nhân.

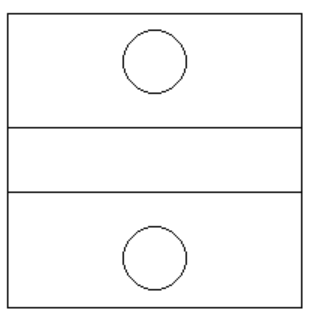
Kết quả thực hiện:

Icon

Description automatically generated

* + 1. Dấu chia

Ý tưởng: Ta vẽ dấu chia dựa theo bên trong hình vuông. Ta chia cạnh theo chiều dọc thành 3 phần: phần 1 và phần 3 là 2 hình tròn có X = xT, Y tương ứng X1 + canh/6 và X1 + cạnh\*5/6, bán kính là cạnh/9. Phần thứ 2 ở giữa là một hình chữ nhật chiều dài = cạnh, chiều rộng là 1/3 cạnh. Thực hiện vẽ 2 hình tròn và 4 đường thẳng để tạo ra dấu chia. Minh họa ý tưởng:



Ta chỉ sử dụng những điểm cần thiết để vẽ 2 hình tròn và hình chữ nhật ở giữa.

Kết quả thực hiện:

A picture containing diagram

Description automatically generated

1. Thuật toán tô màu

Sử dụng thuật toán Boundary Fill, việc xét các điểm điểm theo chứ tự loang tương tự như thuật toán DFS (duyệt theo chiều sâu). Sử dụng một vector để push và pop như stack để lưu lại tọa độ các pixel cần duyệt.

Quá trình tô màu thực hiện như sau:

* Bước 1: Đưa điểm người dùng nhấn chuột trái vào vị trí muốn tô màu vào stack. Sử dụng pixelNotWhite để lưu lại những điểm được tô màu mới.
* Bước 2: Duyệt cho đến khi stack trống: pop pixel ra điểm kiểm tra (tìm mã màu điểm đó trong pixelNotWhite), nếu điểm đó khác mã màu với viền hoặc màu muốn tô thì thực hiện đưa điểm đó vào lại pixelNotWhite (cấu trúc dữ liệu map tự cập nhập value – mã màu nếu key trùng nhau).
* Bước 3: Duyệt các điểm xung quanh của điểm đó (trên dưới trái phải) để đưa vào stack. Quay trở lại Bước 2.

Sau khi thực hiện xong việc tô màu. Ta put toàn bộ các tọa độ pixel với mã màu tương ứng từ pixelNotWhite lên trên màn hình. Và xuất thời gian tô màu ra màn hình Console.

Kết quả thực hiện:

Shape

Description automatically generated with medium confidence

**Độ chính xác của thuật toán:** thuật toán Boundary Fill sau khi thực hiện cải tiến bằng khử đệ quy và sử dụng stack để thực hiện việc loang màu theo chiều sâu kết hợp với hàm PutPixel (đưa vị trí cùng mã màu lên màn hình glut) cũng đã được cải tiến với việc chỉ sử dụng 2 câu lệnh có sẵn của opengl đã thấy sự chính xác rất cao và có một tốc độ nhanh hơn, giải quyết được vấn đề stack overflow.

**So sánh các thuật toán khác:**

So với Scan Line:

* Ưu điểm: Boundary Fill dễ cài đặt hơn, dễ hiểu hơn, có thể áp dụng cho các trường hợp đặt biệt như tô màu cho những hình có khoảng cách giữa 2 điểm ngăn nhỏ (ví dụ như hình chữ nhật có chiều rộng nhỏ), điều đó giúp Boundary Fill chạy nhanh hơn.
* Nhược điểm: Boundary Fill nếu cài theo cách đệ quy sẽ bị stack overflow khi tô hình có kích thước lớn. Tuy nhiên, với việc cải tiến khử đệ quy thì nhược điểm này Boundary Fill đã được khắc phục.

So với Flood Fill

* Ưu điểm : với sự cải tiến mà ta đã làm cho Boundary Fill đa số sẽ chạy nhanh hơn so với Flood Fill, bộ nhớ sử dụng ít hơn, đồng thời giúp giảm độ phức tạp của thuật toán.
* Nhược điểm : Boundary Fill chỉ xử lí các hình có ranh giới 1 màu, Flood Fill không duyệt tìm màu ranh giới liên tục như Boundary mà sẽ thực hiện tô màu ngẫu nhiên sau đó màu cũ sẽ được thay thế.

1. Hướng dẫn sử dụng

Chạy file thực thi để khởi động chương trình.

Nhấn chuột phải để mở Menu. Và chọn hình muốn vẽ hoặc màu muốn tô.

Với việc vẽ hình: sau khi chọn xong hình, nhấn một điểm trên màn hình glut (nhấn 1 lần và thả không nhấn giữ), sau đó di chuyển chuột để xem thử hình minh họa, nhấn chuột lần nữa để hoàn thiện quá trình vẽ hình. Sau khi vẽ xong 1 hình mà chưa chọn hình khác, ta vẫn có thể tiếp tục lặp lại quá trình vẽ để vẽ lại hình này.

Với việc tô màu: nhấn chuột tại một ví trí muốn tô trên màn hình glut. Chương trình sẽ thực hiện tô, sau khi tô xong sẽ hiển thị vùng màu được tô lên màn hình glut.

Có thể vẽ nhiều hình trên màn hình glut, có thể tô màu đè lên nhau giữa các vùng.

1. Tài liệu tham khảo
2. Các bài giảng của thầy Trần Thái Sơn và hướng dẫn thực hiện đồ án của thầy Võ Hoài Việt
3. [GLUT Tutorial – Creating Menus and Submenus in GLUT](https://followtutorials.com/2012/05/glut-tutorial-creating-menus-and-submenus-in-glut.html)