BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC TẬP CƠ SỞ

CÀI ĐẶT THUẬT TOÁN TÔ MÀU ĐA GIÁC BẰNG GIẢI THUẬT SCANLINE VÀ FLOODFILL

Giảng viên hướng dẫn: ThS. Đoàn Vũ Thịnh

Sinh viên thực hiện: Ngô Minh Thư - Nguyễn Thị Bích Triều

Mã số sinh viên: 60137031 - 60137308

KHÁNH HÒA-2021

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN HỆ THỐNG THÔNG TIN

BÁO CÁO THỰC TẬP CƠ SỞ

CÀI ĐẶT THUẬT TOÁN TÔ MÀU ĐA GIÁC BẰNG GIẢI THUẬT SCANLINE VÀ FLOODFILL

Giảng viên hướng dẫn: ThS. Đoàn Vũ Thịnh

Sinh viên thực hiện: Ngô Minh Thư - Nguyễn Thị Bích Triều

Mã số sinh viên: 60137031 - 60137308

Khánh Hòa, tháng 01/2021

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: CÀI ĐẶT THUẬT TOÁN TÔ MÀU ĐA GIÁC BẰNG GIẢI THUẬT SCANLINE VÀ FLOODFILL

Giảng viên hướng dẫn: ThS. Đoàn Vũ Thịnh

Sinh viên được hướng dẫn: Ngô Minh Thư - Nguyễn Thị Bích Triều

MSSV: 60137031 - 60137308

Khóa: 60 **Ngành:** Công nghệ Thông tin

Lần	Ngày	Nội dung	Nhận xét của GVHD
1	7/12/2020	Nhận đề tài hướng dẫn và định hướng giải quyết vấn đề. Sinh viên trình bày kế hoạch thực hiện.	Sinh viên và GVHD trao đổi nội dung của đề tài. Phân chia công việc theo từng thời gian sao cho phù hợp với yêu cầu.
2	14/12/2020	Sinh viên trình bày việc mô phỏng thuật toán chính dựa trên kiến thức đã được học ở môn kỹ thuật đồ họa và các kiến thức thu nhận được từ Internet để minh họa bài toán đa dạng nhất có thể.	Sinh viên hiểu được vấn đề cần phải thực hiện và có giải pháp cho từng vấn đề cụ thể. Tuy nhiên một vấn đề hoàn toán mới là sử dụng thư viện của chuột thì chưa làm lần nào nên cần đầu tư nhiều thời gian hơn.
3	21/12/2020	Sinh viên hoàn thiện các thuật toán đã đề ra với dữ liệu đầu vào được nhập từ bàn phím. Trình bày thuật toán với các trường hợp sai và chỉ ra được hướng khắc phục cho các trường hợp đó.	Sinh viên hiểu nội dung của thuật toán khá chi tiết với các trường hợp hạn chế của thuật toán. Việc lập trình cũng hoàn thành ở mức độ nhập dữ liệu từ bàn phím nhưng vấn đề với chuột thì chưa thực hiện được.
4	4/01/2021	Sinh viên nộp bản thảo của báo cáo thực tập lần thứ 1 và tiến hành chỉnh sửa.	Báo cáo chỉ trình bày chung chung chưa đi vào cụ thể phân tích các yêu cầu của bài toán,

			hình ảnh, bảng biểu chưa trình bày rõ ràng. Cần hiệu chỉnh theo yêu cầu của GVHD.
5	11/01/2021	Sinh viên nộp bản thảo lần 2 và có minh họa với thư viện của chuột nhưng chưa thể kết nối với phần trước.	Báo cáo lần này đã khắc phục được các lỗi của lần trước, tuy nhiên phần phương pháp và kết quả chưa nổi bật, chưa có sự liên kết giữa các phần.
6	18/1/2021	Sinh viên nộp bản thảo lần cuối sau khi đã chỉnh sửa các yêu cầu như đã đề ra.	Sinh viên nghiêm túc chỉnh sửa báo cáo theo định hướng của GVHD.

Nhận xét chung (sau khi sinh viên hoàn thành ĐA/KL):

Sinh viên thực hiện tốt các yêu cầu của GVHD, trong quá trình thực hiện đề tài có sự liên hệ chặt chẽ với GV. Theo lịch hẹn Sinh viên đều có mặt để trình bày ý tưởng của các nội dung lần trước. Trong quá trình hoàn tất báo cáo đều nỗ lực không ngừng mặc dù đang cao điểm của đợt thi học kỳ nhưng SV vẫn dành thời gian không ít cho TTCS.

Về nội dung báo cáo đã thỏa mãn các yêu cầu của đề tài như trong đề cương. Về kết quả chương trình đã minh họa được thuật toán. Về các yêu cầu cao hơn như sử dụng thư viện của chuột hay minh họa các trường hợp nhược điểm của thuật toán thì chưa thực hiện thành công.

Về hình thức của báo cáo và sản phẩm, báo cáo trình bày rõ ràng các mục tiêu, phương pháp, kết quả và thảo luận cho sản phẩm. Còn về sản phẩm như đã trình bày có phần hạn chế.

Điểm hình thức: 8.5/10 Điểm nội dung: 8.0/10 **Điểm tổng kết: 8.3/10**

Đồng ý cho sinh viên: Được bảo vệ: ☑ Không được bảo vệ: ☐

Khánh Hòa, ngày 20 tháng 01 năm 2021

Cán bộ hướng dẫn

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

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

Qua đây, nhóm xin chân thành cảm ơn thầy Đoàn Vũ Thịnh, người đã trực tiếp quan tâm và hướng dẫn chúng 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 chúng em còn nhiều thiếu sót, kính mong sự góp ý của quý thầy cô.

MỤC LỤC

DANH MỤC HÌNH

TÓM TẮT

Đồ họa máy tính là một lĩnh vực của Công nghệ Thông tin, ở đó việc nghiên cứu xây dựng và tập hợp các công cụ (mô hình lý thuyết và phần mềm) khác nhau để kiến tạo, xây dựng, lưu trữ và xử lý các mô hình và hình ảnh của các đối tượng, sự vật, hiện tượng trong cuộc sống, sản xuất. Thuật toán tô màu bằng dòng quét và tô màu bằng đường biên trong đồ họa máy tính có tầm quan trọng rất lớn và được sử dụng rộng rãi trong các phần mềm đồ họa phổ biến hiện này như Adobe Photoshop, Corel Draw, Microsoft Paint.

Quy trình thực hiện được trải qua các bước từ cài đặt thuật toán, hiển thị kết quả đầu ra trên màn hình đều được thực hiện trên môi trường C/C++ thông qua ứng dụng DevC/C++ có kết hợp với thư viện graphics.h.

Sản phẩm đã minh họa được từng bước giải thuật tô màu đối tượng với Scanline và FloodFill. Đồng thời cũng chỉ ra các trường hợp hạn chế của mỗi thuật toán và cách khắc phục các nhược điểm đó. Sản phẩm chạy tốt với dữ liệu được nhập từ bàn phím. Về yêu cầu sử dụng chuột để tạo đa giác và tô màu cho đa giác với các thao tác từ chuột chưa được triển khai. Đây cũng là thiếu sót của đề tài ngay từ khi đặt ra.

Toàn bộ mã nguồn của báo cáo được tải lên theo địa chỉ:

https://github.com/thinhdoanvu/ThuctapCoSo2020/tree/main/NgoMinhThu_NguyenThiBichTrieu/

GIỚI THIỆU

Đồ họa máy tính là một lĩnh vực của Công nghệ thông tin, ở đó nghiên cứu, xây dựng và tập hợp các công cụ (mô hình lý thuyết và phần mềm) khác nhau để kiến tạo, xây dựng, lưu trữ và xử lý các mô hình và hình ảnh của các đối tượng, sự vật, hiện tượng trong cuộc sống, sản xuất, nghiên cứu. Đồ họa máy tính góp phần quan trọng làm cho giao tiếp giữa con người và máy tính trở nên thân thiện hơn. Từ đồ họa trên máy tính chúng ta có nhiều lĩnh vực có ứng dụng rất quan trọng của đồ họa máy tính trong thực tế như: tạo mô hình, hoạt cảnh, hỗ trợ thiết kế đồ họa, mô phỏng hình ảnh, chuẩn đoán hình ảnh (trong Y tế), huấn luyện đào tạo ảnh (quân sự, hàng không,…).

Hình 1.1. Ví dụ về đồ họa máy tính

(https://www.vietngapc.vn/upload_images/case%20stream/Case%20Designer/designermonitor.jpg)

Thuật toán tô màu trong đồ họa máy tính có tầm quan trọng rất lớn và được sử dụng rộng rãi trong các phần mềm phổ biến hiện nay. Từ những phần mềm đơn giản 3 như Paint, Powerpoint trong bộ Office của Window đến những ứng dụng thiết kế đồ họa chuyên nghiệp như Photoshop, AutoCad.

1.1. Thuật toán tô màu đối tượng

Các vùng tô là một trong những đối tượng đồ họa cơ sở được hầu hết các công cụ lập trình đồ họa hỗ trợ. Có hai dạng vùng tô thường gặp: (1) tô bằng một màu thuần nhất (solid fill); (2) tô theo một mẫu tô (fill-pattern) nào đó.

Một vùng tô thường được xác định bởi một đường khép kín nào đó, gọi là đường biên. Vùng tô và mẫu tô được minh họa bởi các hình 1.2.

(a) (b) (c)

Hình 1.2. Các kiểu mẫu tô (Solid - b và Pattern - c)

Để tô màu một vùng tô, người ta thường chia thành hai công đoạn:

(1). Xác định các điểm để tô

(2). Xác định giá trị màu tô.

Có hai cách tiếp cận chính để tô màu một vùng tô đối với thiết bị hiển thị dạng điểm, nó là tô dựa theo đường biên (boundary fill) và tô theo dòng quét (scanline fill). Trong đó, phương pháp tô dựa theo đường biên (boundary fill) sẽ bắt đầu từ một điểm ở bên trong vùng tô và từ đó loang dần ra cho tới khi gặp các điểm biên. Cách tiếp cận này thường được dùng cho các vùng tô có dạng đường biên phức tạp. Phương pháp tô theo dòng quét sẽ xác định các phần giao của các dòng quét kế tiếp nhau với đường biên của vùng tô, sau đó sẽ tô màu các điểm thuộc về phần giao này. Cách tiếp cận này thường được dùng để tô màu các đa giác, đường tròn, ellipse và một số đường cong đơn giản.

1.2. Thuật toán tô màu theo dòng quét Scanline

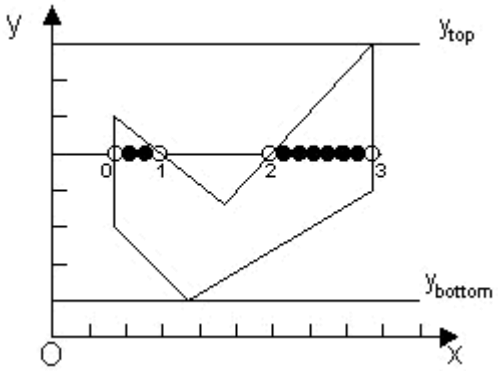
1.2.1. Ý tưởng chính của thuật toán

Với mỗi dòng quét, ta xác định phần giao của đa giác và dòng quét, rồi tô màu các pixel thuộc đoạn giao đó. Để xác định các đoạn giao, ta tiến hành việc tìm giao điểm của dòng quét với các cạnh của đa giác, sau đó các giao điểm này sẽ được sắp theo thứ tự tăng dần của hoành độ giao điểm. Các đoạn giao chính là các đoạn thẳng được giơi hạn bởi từng cặp giao điểm một, ví dụ như (0,1),(2, 3)…

Ta có thể tóm tắt các bước chính của thuật toán như sau:

Bước 1: Tìm y_{top}, y_{bottom} lần lượt là giá trị lớn nhất, nhỏ nhất của tập các tung độ của các đỉnh của đa giác đã cho:

$y_{top} = \max \{y_i, (x_i, y_i) \in P\}$; $y_{bottom} = \min \{y_i, (x_i, y_i) \in P\}$.



Hình 1.3. Thuật toán scan - line với một dòng quét bất kỳ

Bước 2: Ứng với mỗi dòng quét y = k, với k thay đổi từ y_{bottom} đến y_{top}, lặp:

B2.1. Tìm tất cả các hoành độ giao điểm của dòng quét y = k với các cạnh của đa giác.

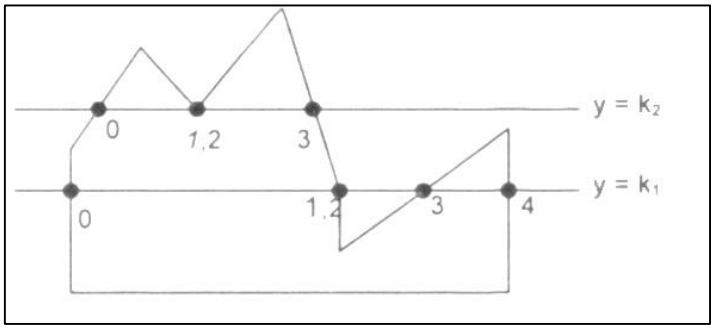
B2.2. Sắp xếp các hoành độ giao điểm theo thử tự tăng dần: x_0, x_1, …

B2.3. Tô màu các đoạn thẳng trên đường thẳng y = k lần lượt được giới hạn bởi các cặp (x0, x1), (x2, x3),.., (x2k, x2k+1).

Nhược điểm:

Nhận xét rằng, ứng với mỗi dòng quét, không phải lúc nào tất cả các cạnh của đa giác cũng tham gia cắt dòng quét. Do đó, để cải thiện tốc độ cần phải có một cách nào đó để hạn chế được số cạnh cần tìm giao điểm ứng với mỗi dòng quét.

Việc tìm giao điểm của cạnh đa giác với mọi dòng quét sẽ gặp các phép toán phức tạp như nhân, chia,... trên số thực nếu ta dùng cách giải hệ phương trình tìm giao điểm. Điều này sẽ làm giảm tốc độ thuật toán khi phải lặp đi lặp lại nhiều lần thao tác này khi dòng quét quét qua đa giác.



Hình 1.4. Dòng quét y=k1/2 đi ngang qua đỉnh sẽ được xét 2 lần

Nếu số giao điểm tìm được giữa các cạnh đa giác và dòng quét là lẻ thì việc nhóm từng cặp giao điểm kế tiếp nhau để hình thành các đoạn tô có thể sẽ không chính xác. Điều này chỉ xảy ra khi dòng quét đi ngang qua các đỉnh của đa giác. Nếu tính số giao

điểm tại đỉnh dòng quét đi ngang qua là hai thì có thể sẽ cho kết quả tô không chính xác như trong trường hợp của Hình 1.4.

Ngoài ra, việc tìm giao điểm của dòng quét với các cạnh nằm ngang là một trường hợp đặc biệt cần phải có cách xử lý thích hợp.

1.2.2. Danh sách các cạnh kích hoạt AET (Active Edge Table)

Để hạn chế số cạnh cần tìm giao điểm ứng với mỗi dòng quét, ta xây dựng một số cấu trúc dữ liệu như sau:

Cạnh đa giác (EDGE)

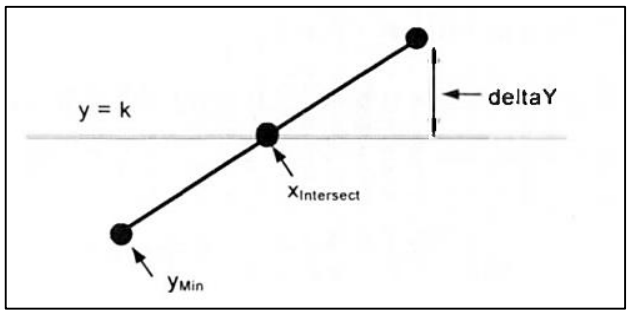
Mỗi cạnh của đa giác được xây dựng từ hai đỉnh kề nhau $P_i(x_i, y_i)$ và $P_{i+1}(x_{i+1}, y_{i+1})$ gồm các thông tin sau:

y_{Min} Giá trị tung độ nhỏ nhất trong hai đỉnh của cạnh;

$y_{Intersect}$ Hoành độ giao điểm của cạnh với dòng quét hiện đang xét;

$Dx_{PerScan}$ Giá trị $\frac{1}{m}$ (m là hệ số góc của cạnh);

DeltaY Khoảng cách từ dòng quét hiện hành tới đỉnh y_{Max}



Hình 1.5. Thông tin của một cạnh

Danh sách các cạnh kích hoạt AET:

Danh sách này dùng để lưu các tập cạnh của đa giác cỏ thể cắt ứng với dòng quét hiện hành và tập các điểm giao tương ứng. Nó có một số đặc điểm sau:

Các cạnh trong danh sách được sắp theo thứ tự tăng dần của các hoành độ giao điểm để có thể tô màu các đoạn giao một cách dễ dàng.

Có sự thay đổi ứng với mỗi dòng quét đang xét, do đó danh sách này sẽ được cập nhật liên tục trong quá trình thực hiện thuật toán. Để hỗ trợ cho thao tác này, đầu tiên người ta tổ chức một danh sách chứa toàn bộ các cạnh của đa giác gọi là ET (*Edge Table*), được sắp theo thứ tự tăng dần của Y_{Min}, rồi sau mỗi lần dòng quét thay đổi sẽ di chuyển các cạnh trong ET thoả mãn điều kiện sang AET.

Một dòng quét y = k chỉ cắt một cạnh của đa giác khi và chỉ khi:

$$\begin{cases} k \geq y_{min} \\ deltaY > 0 \end{cases}$$

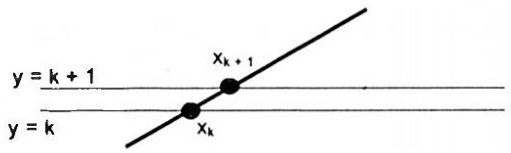
Chính vì vậy, với cách tổ chức của ET (sắp theo thứ tự tăng dần của Y_{Min}), điều kiện để chuyển các cạnh từ ET sang AET sẽ là $k > Y_{Min}$ và điều kiện để loại một cạnh ra khỏi AET là deltaY < 0.

1.2.3. Công thức tìm giao điểm

Nếu gọi x_k, x_{k+1} lần lượt là các hoành độ giao điểm của một cạnh nào đó với các dòng quét y = k và y = k+1, ta có:

$$x_{k+1} - x_k = \frac{1}{m}((k+1) - k) = \frac{1}{m} \text{ hay } x_{k+1} = x_k + \frac{1}{m}$$

Như vậy, nếu lưu hoành độ giao điểm ứng với dòng quét trước lại cùng với hệ số góc của cạnh, ta dễ dàng xác định được hoành độ giao điểm ứng với dòng quét kế tiếp một cách đơn giản theo công thức trên. Điều này rút gọn đáng kể thao tác tìm giao điểm của cạnh ứng với dòng quét. Chính vì vậy thông tin của một cạnh có hai biến là $Dx_{PerScan}$ và $X_{Intersec}$.



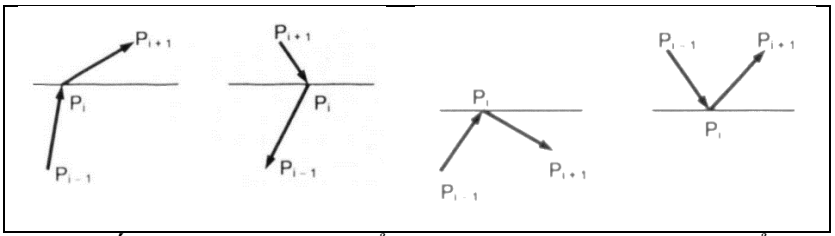
Hình 1.6. Xác định hoành độ giao điểm

1.2.4. Trường hợp dòng quét đi ngang qua đỉnh

Người ta đưa ra quy tắc sau để tính số giao điểm khi dòng quét đi ngang qua đỉnh:

Tính một giao điểm nếu chiều của hai cạnh kề có xu hướng tăng hay giảm.

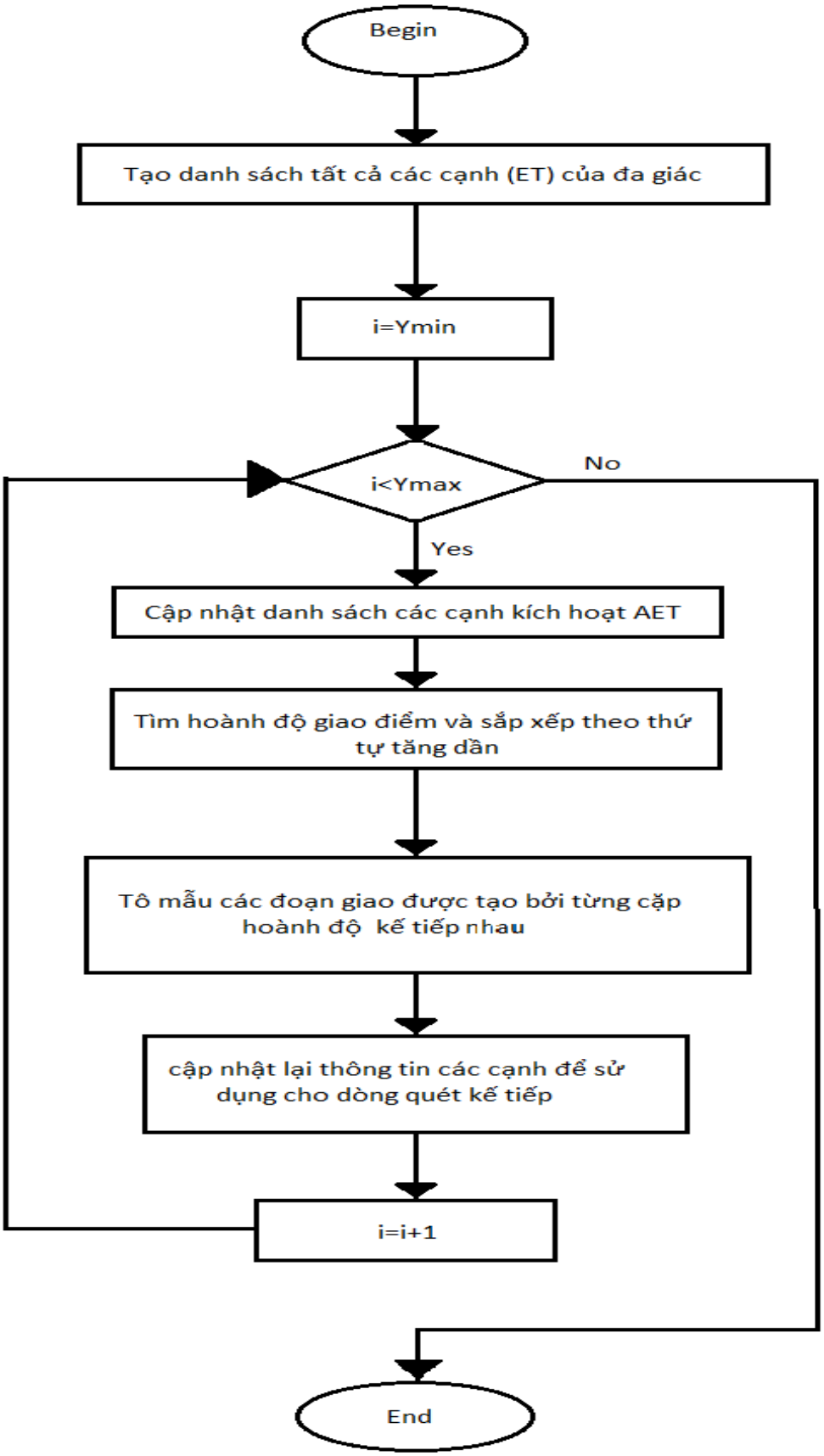
Tính hai giao điểm nếu chiều của hai cạnh kề của đỉnh đó có xu hướng thay đổi, nghĩa là tăng - giảm hay giảm – tăng



Hình 1.7. Quy tắc tính một giao điểm (bên trái) và hai giao điểm (bên phải)

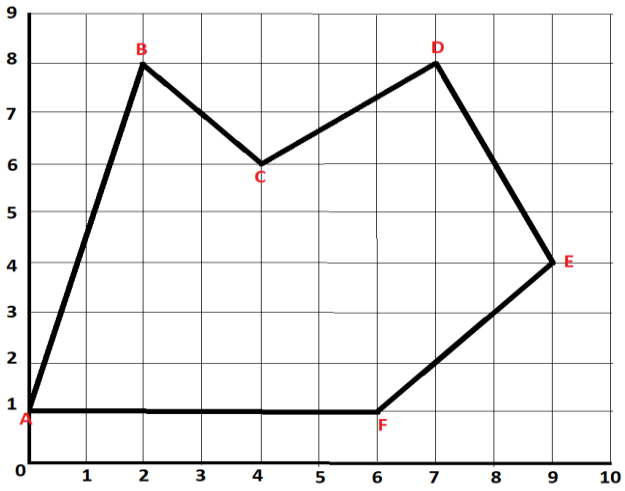
Khi cài đặt không cần phải xét điều kiện này, khi xây dựng dữ liệu cho mỗi cạnh trước khi đưa vào ET, người ta sẽ xử lý các cạnh có đinh tính hai giao điểm bằng cách loại đi một pixel trên cùng của một trong hai cạnh.

Lưu đồ thuật toán



Bài toán ví dụ

Cho 1 đa giác, ABCDEF có tọa độ A(0,1), B(2,8), C(4,6), D(7,8), E(9,4), F(6,1).



Xét trường hợp dòng quét đi qua cạnh nằm ngang

Dòng quét k=1 đi qua cạnh AF, loại bỏ cạnh AF.

Xét trường hợp dòng quét đi qua hai cạnh

Dòng quét k=5 đi qua hai cạnh AB và ED và cắt hai cạnh ở tọa độ x=1.14 và x=8.5. Các điểm ảnh nằm trong đoạn từ 2 tới 8 sẽ được tô

Xét trường hợp dòng quét đi qua đỉnh

Dòng quét k=6 đi qua bốn cạnh AB, CB, CD, ED và cắt ở bốn tọa độ x= 1.4, 4, 4, 8. Tô hai đoạn, đoạn một từ tọa độ 1.4 đến 4 và đoạn hai từ tọa độ 4 đến 8

Dòng quét k=4 đi qua ba cạnh AB, FE, ED và cắt ở ba tọa độ x=0.9,9,9

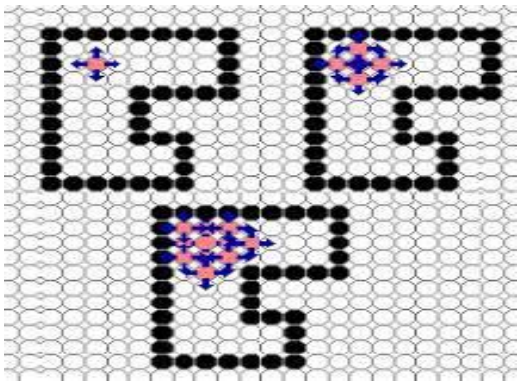
Chỉ xét giao của dòng quét với cạnh AB và ED. Tô các điểm ảnh nằm trong đoạn từ 1 tới 9

1.3. THUẬT TOÁN TÔ MÀU DỰA THEO ĐƯỜNG BIÊN

1.3.1. Thuật toán Boundary Fill

Khác với thuật toán tô màu dựa theo dòng quét, đường biên của vùng tô được xác định bởi tập các đỉnh của một đa giác, đường biên trong thuật toán được mô tả bằng một giá trị duy nhất đó là màu của tất cả các điểm thuộc về đường biên.

Bắt đầu từ điểm nằm bên trong vùng tô, ta kiểm tra các điểm lân cận của nó đã được tô màu có phải là điểm biên hay không, nếu chưa phải là điểm thuộc đường biên thì tô màu cho điểm đó. Quá trình này được lặp lại cho tới khi nào không còn tô được điểm nào nữa thì dừng. Bằng cách này, toàn bộ các điểm thuộc vùng tô được kiểm tra và sẽ được tô hết.



Hình 1.8. Tô màu lân cận 4

Có hai quan điểm về cách tô màu này, đó là dùng bốn điểm lân cận hay tám điểm lân cận. Các bước thực hiện thuật toán:

Bước 1: Kẻ biên vùng cần tô.

Bước 2: Xác định một điểm (x,y) nằm trong vùng cần tô.

Bước 3: Tô điểm (x,y) sau đó tô loang các điểm lân cận.

Với lân cận 4: Tô các điểm có tọa độ (x-1,y); (x, y+1); (x+1,y) và (x,y-1).

Với lân cận 8: Tô các điểm có tọa độ (x-1,y); (x-1,y+1); (x, y+1); (x+1,y+1); (x+1,y); (x+1,y-1); (x,y-1); (x-1,y-1).

Ưu điểm: Có thể tô các vùng có hình dạng bất kỳ.

Nhược điểm: Trong cài đặt thuật toán ở trên, việc gọi thực hiện đệ quy thuật toán cho các điểm lân cận của điểm hiện hành, không quan tâm tới việc điểm đó đã được xét ở bước trước hay chưa. Ví dụ: Khi xét bốn điểm lân cận của điểm hiện hành (x, y), thì khi gọi thực hiện đệ quy với điểm hiện hành là một trong bốn điểm lân cận trên, (x, y) vẫn được xem là điểm lân cận của chúng và lại được gọi thực hiện lại. Dễ dẫn tới tràn bộ nhớ khi vùng tô lớn

1.3.2. Thuật toán Flood Fill

Để khắc phục các nhược điểm trên của thuật toán Boundary Fill, ta có thuật toán Flood Fill với các bước thực hiện thuật toán như sau:

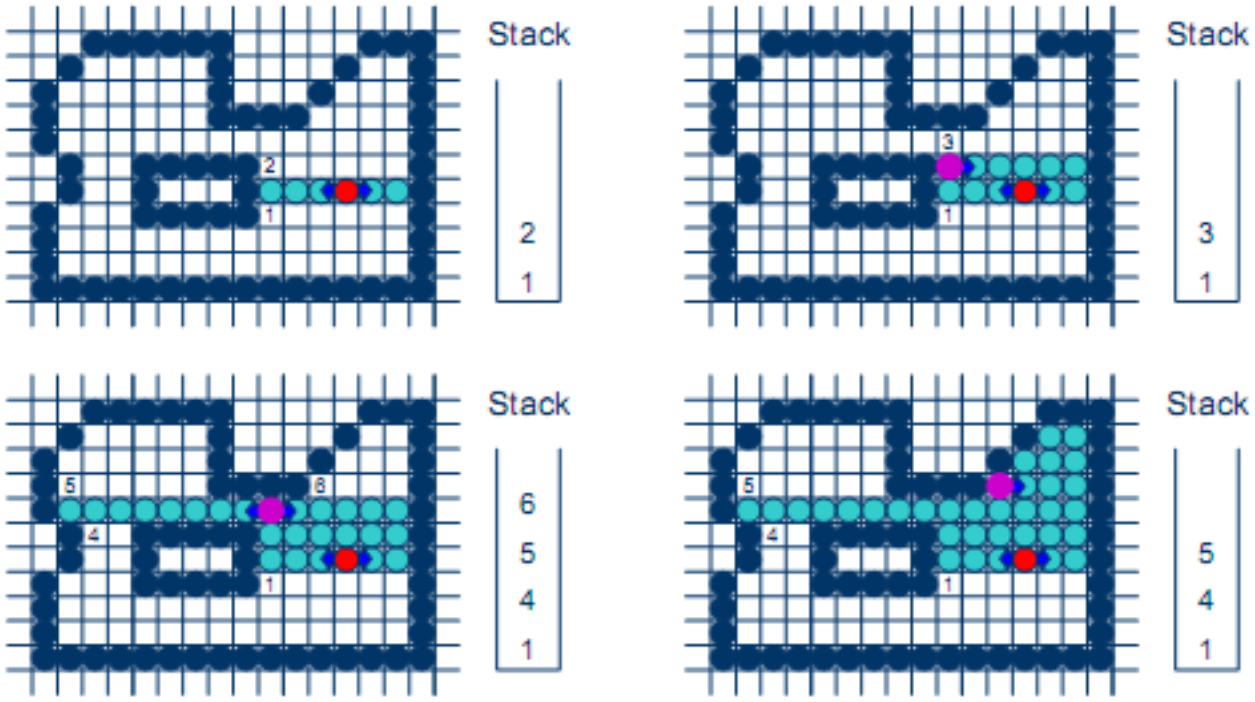
Bước 1: Khởi tạo 1 điểm nằm trong vùng tô.

Bước 2: Thực hiện tô loang dần theo chiều ngang (trái qua phải và phải qua trái) cho đến khi dụng biên thì dừng lại.

Bước 3: Ứng với mỗi điểm trên dòng quét ngang, thực hiện loang để tìm những điểm ảnh có hoành độ nhỏ nhất sát với biên chưa được tô nằm trên và dưới, sau đó lưu vào Stack.

Bước 4: Lặp bước 2 nếu còn một điểm trong Stack chưa được tô.

Như vậy, chỉ cần lưu lại thông tin của điểm bắt đầu mỗi đoạn giao của dòng quét ngang thay vì phải lưu toàn bộ các điểm lân cận chưa được tô của điểm hiện hành.



Hình 1.9. Minh họa thuật toán Flood Fill

1.4. Dev C++

Bloodshed Dev-C ++ (https://www.bloodshed.net/devcpp.html) là môi trường phát triển tích hợp (IDE) đầy đủ tính năng cho ngôn ngữ lập trình C/C ++ sử dụng Mingw của GCC (Bộ sưu tập trình biên dịch GNU) làm trình biên dịch. Dev-C ++ cũng có thể kết hợp với Cygwin hoặc bất kỳ trình biên dịch dựa trên GCC nào khác. Các tính năng của Dev-C++:

- ✓ Hỗ trợ trình biên dịch dựa trên GCC.
- ✓ Gỡ lỗi tích hợp (sử dụng GDB- General DeBug).
- ✓ Quản lý dự án.
- ✓ Trình chỉnh sửa cú pháp.
- ✓ Trình duyệt lớp.
- ✓ Hoàn thành mã.
- ✓ Danh sách chức năng.
- ✓ Hồ sơ hỗ trợ.
- ✓ Nhanh chóng tạo Windows, console, thư viện tĩnh và DLL13.

- ✓ Hỗ trợ các mẫu để tạo các loại dự án của riêng bạn.
- ✓ Tạo Makefile.
- ✓ Chỉnh sửa và biên dịch các tệp Tài nguyên.
- ✓ Quản lý công cụ.
- ✓ Hỗ trợ in.
- ✓ Tìm và thay thế mã lệnh.
- ✓ Hỗ trợ CVS.

1.5. Thư viện Graphics.h

Vì sử dụng DevC++ làm trình biên dịch cho việc cài đặt thuật toán nên không thể thực hiện trên môi trường Windows. Vì vậy, một môi trường giả lập graphic của Borland C được Michael tạo ra thư viên có tên là Graphics.h. để có thể làm được điều đó. Micheal đã thay đổi BGI library (thư viện BGI) thành thư viện có tên WinBGIm để có thể sử dụng tốt trên windows. Và bây giờ bạn đã có thể sử dụng tốt các hàm đặc biệt của borland bằng DevC++ (https://github.com/SagarGaniga/Graphics-Library)

CHƯƠNG 2. PHƯƠNG PHÁP NGHIÊN CỨU

2.1. Cài đặt DevC và thư viện graphics.h

Tải file cài đặt phần mềm DevC++ theo đường dẫn trong mục 1.4. Sau đó mở file vừa tải, và tiến hành cài đặt. Thư viện graphics.h được tiến hành cài đặt theo các bước:

Bước 1: Copy 6-ConsoleAppGraphics và ConsoleApp_cpp_graph

Paste C:\...\Dev-Cpp\Templates

Bước 2: Copy graphics và winbgim

Paste C:\... Dev-Cpp \MinGW64\x86_64-w64-mingw32\include

Bước 3: Copy libbgi.a

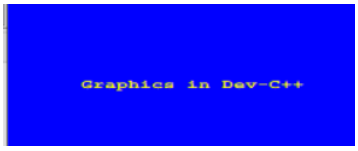
Paste C:\...\Dev-Cpp\MinGW64\x86_64-w64-mingw32\lib

Bước 4: Ở DevC++ => New Project => Console Graphics Application

Bước 5: Thay đổi Tools - Complier Option: TDM – GCC 4.9.2 32 bit Release

Bước 6: Sử dụng đoạn code mẫu bên dưới để test thư viện winbgim

```
#include <winbgim.h>

int main(int argc, char *argv[])
{
    // now, you can run project
    initwindow(300, 300);          // init window graphics
    setbkcolor(1);                 // set background
    cleardevice();
    setcolor(14);                  // set text color
    outtextxy(50,100,"Graphics in Dev-C++");// print text in window graphics

    while(!kbhit()) delay(1);      // pause screen
    return 0;
}
```



Hình 2.1. Ví dụ minh hoạ thư viện winbgim

2.2. Cài đặt thuật toán Scanline

Khai báo thư viện

```
#include <stdio.h>
#include <graphics.h>
#define max_dinh 20
```

Khai báo biến

```
int dinh[max_dinh][2];
int xgd[max_dinh];
int dx,dy,ymin,ymax;
float hesogoc[max_dinh];
int i,j,k,tam,sodinh;
```

Tạo đa giác: Mỗi đỉnh của đa giác bao gồm hoành độ X[] và tung độ Y[] được lưu trữ trong 2 mảng 1 chiều. Mỗi phần tử là kiểu số nguyên. Tọa độ các đỉnh của đa giác được nhập trực tiếp từ bàn phím thông qua gọi hàm nhapthongso().

```
void nhapthongso()
{
    printf("Nhap so dinh cua da giac: ");
    scanf("%d",&sodinh);

    //nhap lan luot toa do cho cac dinh cua da giac
    for(i=0;i<sodinh;i++)
    {
       printf("X[%d]= ",i);
       scanf("%d",&dinh[i][0]);
       printf("Y[%d]= ",i);
       scanf("%d",&dinh[i][1]);
    }

    //gan toa do dinh cuoi cung tro lai dinh dau tien de tao thanh da giac khep kin
    dinh[sodinh][0]=dinh[0][0];
    dinh[sodinh][1]=dinh[0][1];

    //tinh toan he so goc: m=dy/dx nhung de thuan tien cho phep toan sau nay: m=dx/dy
    for(i=0;i<sodinh;i++)
    {
        dx=dinh[i+1][0]-dinh[i][0];
        dy=dinh[i+1][1]-dinh[i][1];
        if(dx==0)
            {
                hesogoc[i]=0.0;
            }
            if(dy==0)
            {
                hesogoc[i]=1.0;
            }
            if(dx!=0 && dy!=0)
            {
                    hesogoc[i]=(float)dx/dy;
                        }
            }

            //tim gia tri lon nhat va nho nhat cua tung do
            ymin=dinh[0][1];
            ymax=dinh[0][1];

            for(i=1;i<sodinh;i++)
            {
               if(ymin>dinh[i][1])
               {
                  ymin=dinh[i][1];
                       }
                       if(ymax<dinh[i][1])
               {
                  ymax=dinh[i][1];
                       }
            }

            //in ra man hinh cac tham so hesogoc, ymin, ymax
            for(i=0;i<sodinh;i++)
            {
                  printf("%3f\t",hesogoc[i]);
            }
            printf("\ntung do be nhat: %d",ymin);
            printf("\ntung do lon nhat: %d",ymax);
}
```

Kết thúc việc nhập đa giác sẽ tính toán hệ số góc trên mỗi cạnh của đa giác và đồng thời xác định tung độ của đỉnh lớn nhất (y_{Max}) và bé nhất (y_{min}).

Vẽ đa giác: Đa giác sau khi được nhập các tọa độ sẽ được hiển thị lên màn hình thông qua gọi hàm vedagiac().

```
void vedagiac()
{
    setcolor(GREEN);
    for(i=0;i<sodinh;i++)
    {
        line(dinh[i][0],dinh[i][1],dinh[i+1][0],dinh[i+1][1]);
    }
}
```

Thuật toán tô màu đa giác

```
void scanline()
{
    int y;

    for(y=ymin;y<=ymax;y++)
    {
        k=0;
        //kiem tra dong quet co cat qua canh nao do hay khong?
        for(i=0;i<sodinh;i++)
        {
            if( ((dinh[i][1] <=y) && (dinh[i+1][1]>y))|| ((dinh[i][1] > y ) && (dinh[i+1][1]<=y)) )
            {
                xgd[k]=(int) (dinh[i][0] + hesogoc[i]*(y - dinh[i][1]));
                k++;
                    }
                }

                // sap xep toa do cac giao diem theo thu tu tang dan
                for(i=0;i<k-1;i++)
                {
                    for(j=i+1;j<k;j++)
                    {
                        if(xgd[i] > xgd[j])
                        {
                            tam=xgd[i];
                            xgd[i]=xgd[j];
                            xgd[j]=tam;
                                }
                        }
                }

                //in ra man hinh toa do cac giao diem
                printf("\ny = %d: ",y);
                for(i=0;i<k;i++)
                {
                    printf("%d\t",xgd[i]);
                }

                //ve duong thang noi toa do cac giao diem voi tung do la y
                setcolor(BLUE);
                for(i=0;i<k;i=i+2)
                {
                  line(xgd[i],y,xgd[i+1],y);
                }

            }
    }
```

Chương trình chính

```
int main()
{
    nhapthongso();

    //ve da giac
    initwindow(800,800); //khoi tao man hinh do hoa co kich thuoc 400 x 400 px
    vedagiac();

    //to mau theo dong quet
    scanline();

    //doi bam phim Enter
    getch();
}
```

2.3. Cài đặt thuật toán FloodFill

Khai báo thư viện(như trên)

Khai báo biến

```
int a[max_dinh][2];
int sodinh;
int i;
int xc,yc;
```

Tạo và vẽ đa giác (tương tự như thuật toán Scanline)

```
void nhapdagiac()
{
    printf("Nhap so dinh cua da giac: ");
    scanf("%d",&sodinh);

    for(i=0;i<sodinh;i++)
    {
        printf("X[%d] = ",i);
        scanf("%d",&a[i][0]);
        printf("Y[%d] = ",i);
        scanf("%d",&a[i][1]);
          }

          a[sodinh][0]=a[0][0];
          a[sodinh][1]=a[0][1];

          //xac dinh diem ban dau thuoc ben trong da giac
          printf("Toa do diem ben trong da giac:\n");
    printf("xc = ");
    scanf("%d",&xc);
    printf("yc = ");
    scanf("%d",&yc);
}

void vedagiac()
{
    setcolor(GREEN);
    for(i=0;i<sodinh;i++)
    {
        line(a[i][0],a[i][1],a[i+1][0],a[i+1][1]);
          }
}
```

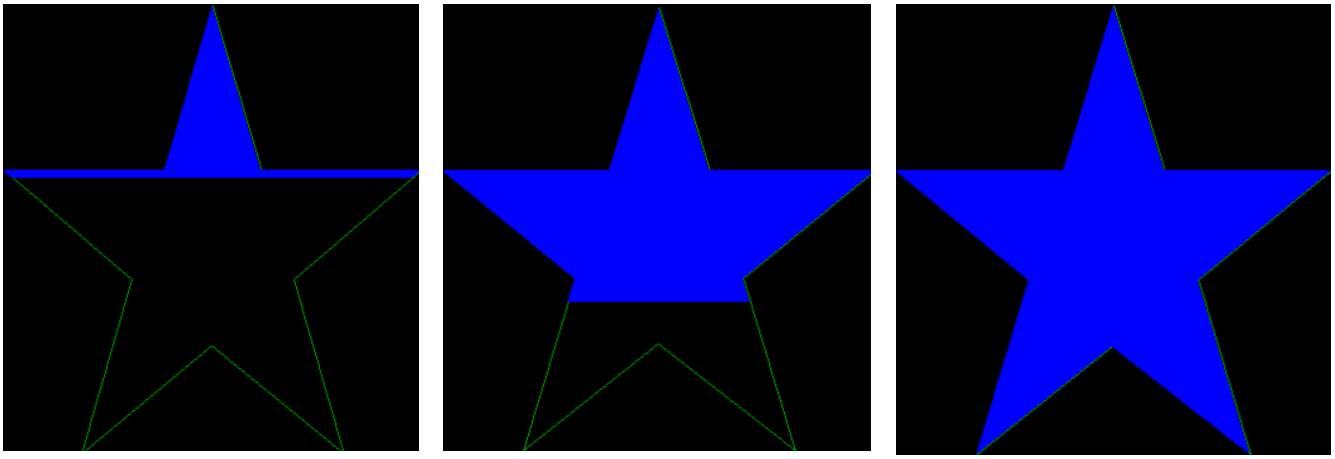
Thuật toán tô màu đa giác

```
//to mau ben phai
void FillRight(int xa, int ya)
{

    if(getpixel(xa,ya) == 0) //mau nen la mau den (BLACK) nen ham getpixel tra ve gia tri la 0
    {
        putpixel(xa,ya,RED);
        //delay(2);
                    //to mau het ve phia ben phai
        FillRight(++xa,ya);

        xa--;

                    //to mau tu diem ke bien den tan cung ben duoi
        FillRight(xa,ya+1);

        //to mau tu diem ke bien den tan cung ben tren
        FillRight(xa,ya-1);
          }

}

//to mau ben trai
void FillLeft(int xa, int ya)
{

    if(getpixel(xa,ya) == 0) //mau nen la mau den (BLACK) nen ham getpixel tra ve gia tri la 0
    {
        putpixel(xa,ya,RED);
        //delay(2);
                    //to mau het ve phia ben trai
        FillLeft(--xa,ya);

        xa++;

                    //to mau tu diem ke bien den tan cung ben duoi
        FillLeft(xa,ya+1);

        //to mau tu diem ke bien den tan cung ben tren
        FillLeft(xa,ya-1);
          }

}

//to mau da giac
void FloodFill()
{
  FillRight(xc,yc);
    FillLeft(xc-1,yc);
}
```

Chương trình chính (tương tự scanline)

CHƯƠNG 3. KẾT QUẢ THỰC HIỆN

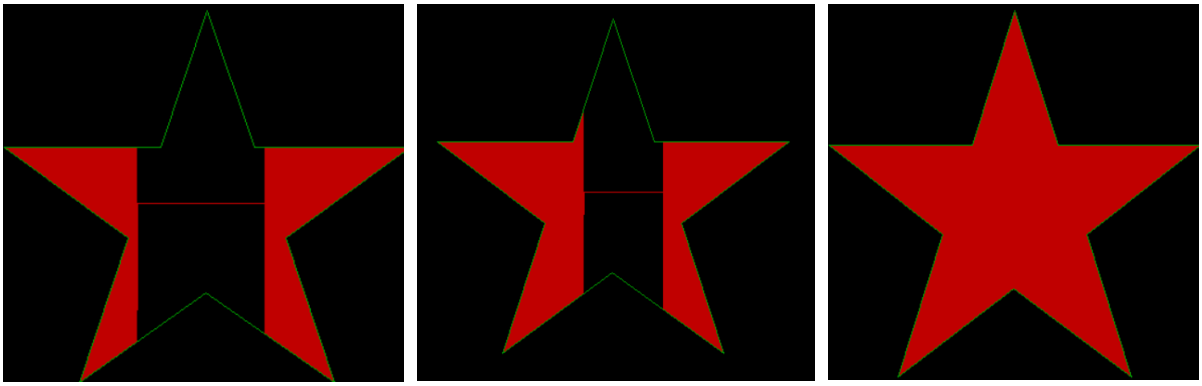
3.1. Tô màu theo dòng quét



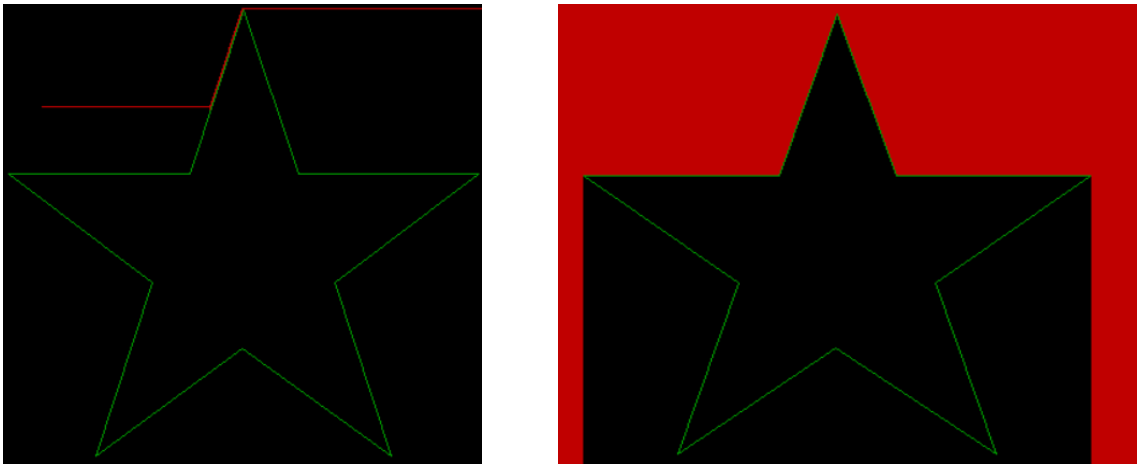
Hình 3.1. Kết quả của thuật toán tô màu Scanline

Từ Hình 3.1 ta thấy, thuật toán bắt đầu từ đỉnh y=ymin, sẽ tô theo từng dòng theo hướng y_{min} đến y_{max} với giá trị màu tô là Blue. Thuật toán sẽ dừng lại khi $y=y_{max}$.

3.2. Tô màu dựa theo đường biên



Hình 3.2. Kết quả thuật toán tô màu Flood Fill



Hình 3.3. Kết quả thuật toán Flood Fill khi chọn điểm bắt đầu ngoài đa giác.

Hình 3.2 chọn điểm có tọa độ (300,300) là điểm bắt đầu, thuật toán, bắt đầu tô từ điểm bắt đầu loang dần sang các điểm cận trái, sau khi đụng biên sẽ loang ngược sang các điểm lân cận phải. Từ hai điểm cận biên, chọn điểm bắt đầu mới là hai điểm cận trên và dưới. Tuần tự cho tới khi không còn điểm nào chưa được tô.

Tuy nhiên, có một điều cần lưu ý là khi sử dụng thuật toán Flood Fill là: Điểm bắt đầu bắt buộc phải là điểm nằm trong hình đa giác cần tô. Nếu điểm đó nằm ngoài đa giác (Hình 3.3 chọn điểm bắt đầu là (100,200) thì thuật toán sẽ tô tất cả các điểm nằm ngoài đa giác. Khi đó biên được chọn là giới hạn của cửa sổ đồ họa.

THẢO LUẬN

Đề tài đã cài đặt được hai thuật toán tô màu là tô màu theo dòng quét (Scanline) và tô màu theo đường biên (Flood Fill) bằng cách nhập dữ liệu từ bàn phím. Ngoài ra còn tìm hiểu thêm về thuật toán tô màu Boundary Fill. Thấy được nhược điểm của thuật toán Flood Fill so với thuật toán Scanline là thực hiện chường trình chậm hơn và dễ bị tràn bộ nhớ cũng như bị lỗi chương trình nhiều hơn.

Tuy nhiên, đề tài vẫn chưa cài đặt được thuật toán với việc nhập dữ liệu từ chuột thay vì bàn phím và đông thời chưa khắc phục được hai nhược điểm của thuật toán Flood Fill là dễ tràn bộ nhớ khi vùng tô quá lớn và phức tạp, và tô ngoài vùng đa giác khi chọn điểm bắt đầu không nằm trong đa giác.

Trong tương lai, nhóm sẽ tiếp tục tìm hiểu thêm và cài đặt thuật toán nhập dữ liệu từ chuột và sẽ cố gắng tìm hiểu và cải tiến thuật toán Flood Fill.

TÀI LIỆU THAM KHẢO

1. Nguyễn Quang Khánh, “Đồ họa máy tính”, 2015, NXB Khoa học Kỹ thuật
2. Vũ Hải Quân, “Đồ họa máy tính”, 2007, NXB Đại Học Quốc Gia
3. D. Hearn, M.P. Baker, “Computer Graphics, C version”, 1997, Prentice Hall
4. Đoàn Vũ Thịnh, “Bài giảng Kỹ thuật đồ họa”, 2019, Đại học Nha Trang