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

# LỜI CẢM ƠN

Để có thể hoàn thành đợt thực tập lần này, tôi xin chân thành cảm ơn đến công ty Velox Tech đã cho tôi có cơ hội được thực hiện việc thiết kế một website hoàn chỉnh, vừa được hiện thực hóa các kiến thức đã được học từ ghế Nhà trường, vừa được tiếp cận với các yếu tố kỹ thuật của công nghệ mới.

Xin gửi lời cảm ơn sâu sắc đến các thầy cô giáo của Khoa Công nghệ Thông tin đã truyền đạt cho em những kiến thức để có thể thực hiện được những yêu cầu của đợt thực tập lần này.

Xin gửi lời cảm ơn chân thành đến các anh chị trong công ty đã phát hiện ra những khiếm khuyết của sản phẩm để có thể có những phiên bản sau tốt hơn phiên bản trước.

Xin được cảm ơn đặc biệt đến ông Jon Inge Sun đã cho tôi có thời gian thực sự ý nghĩa tại đây, được học hỏi, được trải nghiệm công việc thực tế, được tận hưởng văn hóa công ty trong suốt thời gian qua.

**MỤC LỤC**

# DANH MỤC HÌNH

# 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*](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 này. 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: một là tô bằng một màu thuần nhất (solid fill), hai là 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. Một trong những dạng đường biên đơn giản nhất đó là đa giác.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (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

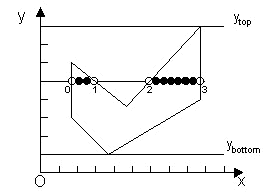
### 1.2.1. Ý tưởng chính của thuật toán

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 ytop, ybottom lần lượt là giá trị lớn nhất, nhỏ nhất của tập các tung độ của các đỉnh cùa đa giác đã cho:

y top = max {yi, (xi, y i) ϵ P}; ybottom = min {yi, (xi, y i) ϵ P}.



Hình 1.3. Thuật toán scan - line với một dòng quét nào đó

**Bước 2:** Ứng với mỗi dòng quét y = k, với k thay đổi từ ybottom đến ytop, lặp:

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

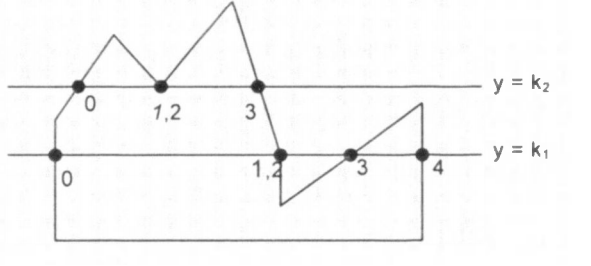
*B2.2.* Sắp xếp các hoành độ giao điểm theo thử tự tăng dần: x0, x1, …

*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 mồi 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 cỏ 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)**

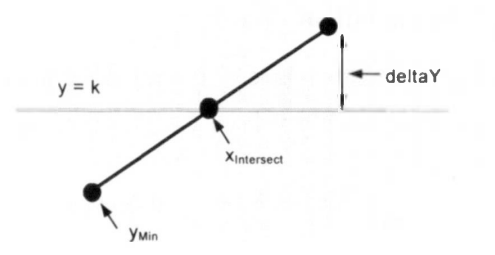
Mỗi cạnh của đa giác được xây dựng từ hai đỉnh kề nhau Pi(xi, yi) và Pi+ 1(xi+1, yi+1) gồm các thông tin sau:

yMin Giá trị tung độ nhỏ nhất trong hai đỉnh của cạnh;

yIntersect Hoành độ giao điểm của cạnh với dòng quét hiện đang xét;

DxPerScan Giá trị (m là hệ số góc của cạnh);

DeltaY Khoảng cách từ dòng quét hiện hành tới đỉnh yMax



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 tồ 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 YMin, 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:

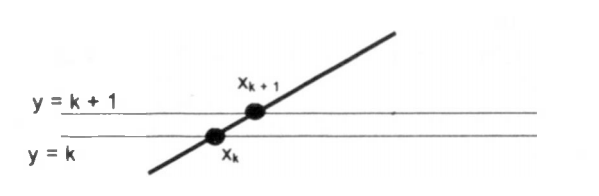
Chính vì vậy, với cách tổ chức của ET (sắp theo thứ tự tăng dần cùa YMin), điều kiện để chuyển các cạnh từ ET sang AET sẽ là k > YMin và điều kiện để loại một cạnh ra khỏi AET là deltaY < 0.

### 1.2.3. Công thức tìm giao điểm

Nếu gọi xk, xk+1 lần lượt là các hoành độ giao điềm của một cạnh nào đó với các dòng quét y = k và y = k+1, ta có:

xk+1 -xk = ((k + 1)- k) = hay xk+1=xk+

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 DxPerScan và XIntersec.



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)

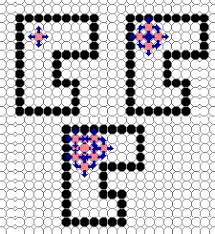
Khi cài đặt không cần phải xét điều kiện này, khi xây dựng dữ liệu cho mỗi cạnh trước khi đưa vào ET, người ta sẽ xử lý các cạnh có đinh tính hai giao điểm bằng cách loại đi một pixel trên cùng của một trong hai cạnh.

## 1.3. THUẬT TOÁN TÒ 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 điềm 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:

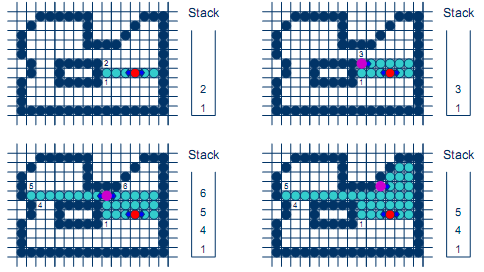
**Bước 1:** Khởi tạo 1 điểm nằm trong vùng tô.

**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 cùa điểm bắt đầu mỗi đoạn giao của dòng quét ngang thay vì phải lưu toàn bộ các điểm lân cận chưa được tô cùa 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

|  |  |
| --- | --- |
| Capture | Capture2 |

Hình 2.1. Ví dụ minh hoạ thư viện winbgim

2.2. Cài đặt thuật toán Scanline

2.3. Cài đặt thuật toán FloodFill

# CHƯƠNG 3. KẾT QUẢ THỰC HIỆN

3.1. Tô màu theo dòng quét

# THẢO LUẬN

Đề tài đã thực hiện được gì? Chưa thực hiện được gì? Và tương lai có cải tiến gì?

Source code được upload tại địa chỉ:

# TÀI LIỆU THAM KHẢO