BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

TÌM HIỂU THUẬT TOÁN TÔ MÀU VÀ LẬP TRÌNH MINH HỌA

Giảng viên hướng dẫn: TS. Nguyễn Đình Hưng

Sinh viên thực hiện: Lê Gia Hưng

Mã số sinh viên: 62130690

Khánh Hòa, tháng 12/2022

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: TÌM HIỂU THUẬT TOÁN TÔ MÀU VÀ LẬP TRÌNH MINH HỌA

Giảng viên hướng dẫn: TS. Nguyễn Đình Hưng

Sinh viên được hướng dẫn: Lê Gia Hưng

MSSV: 62130690

Khóa: 62 **Ngành:** Công nghệ Thông tin

Lần	Ngày	Nội dung	Nhận xét của GVHD
1	5/12/2022	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.
2	12/12/2022	Sinh viên đã phân tích bài toán Scanline dựa trên kiến thức đã được học ở môn kỹ thuật đồ họa và các kiến thức thu nhận được từ Internet để tìm hiểu giải thuật và công cụ lập trình.	Sinh viên đã trình bày thuật toán Scanline và tóm tắt công cụ lập trình.
3	19/12/2022	Sinh viên mô tả giải thuật bài toán Scanline và cài đặt bài toán.	Sinh viên đã mô tả giải thuật và cài đặt bài toán.
4	26/12/2022	Sinh viên đã cài đặt và viết báo cáo.	Bài báo cáo sinh viên có một vài chỗ cần tinh chỉnh về phần nội dung và trình bày.
5	4/1/2022	Sinh viên nộp cáo và mã nguồn hoàn chỉnh.	Sinh viên nghiêm túc chỉnh sửa báo cáo theo định hướng của GVHD.

NHẬN XÉT CỦA GIẢNG VIÊN

Nhận xét của Giảng viên hướng dẫn

Khánh Hoà, Ngày……Tháng…….Năm…….

Giảng viên hướng dẫn

(Ký tên)

Nhận xét của Giảng viên phản biện

Khánh Hoà, Ngày……Tháng…….Năm…….

Giảng viên phản biện

(Ký tên)

LỜI CẢM ƠN

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

Qua đây, em xin chân thành cảm ơn thầy Nguyễn Đình Hưng, 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 em còn nhiều thiếu sót, kính mong sự góp ý của quý thầy cô.

Em xin chân thành cảm ơn!

MỤC LỤC

DANH MỤC HÌNH

TÓM TẮT

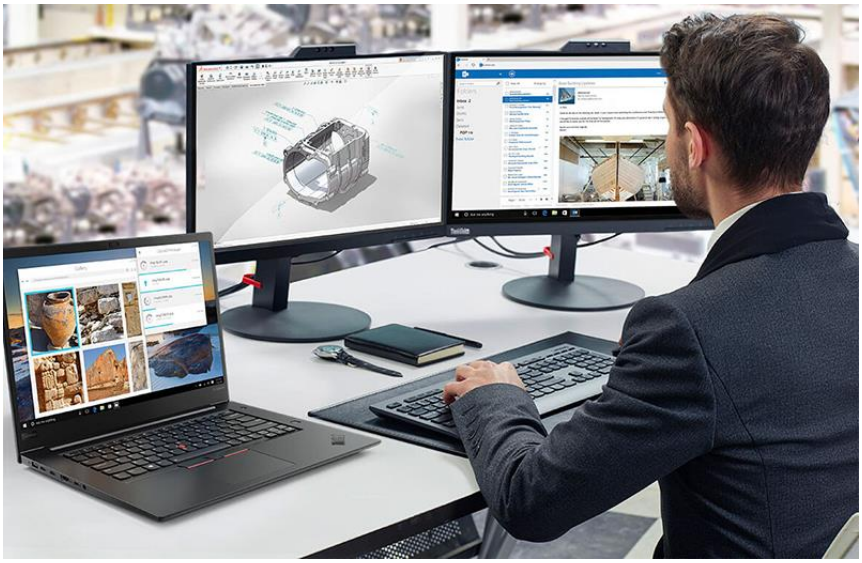
Đồ họa máy tính (Computer graphics) là một lĩnh vực của Công nghệ Thông tin, ở đây chuyên về việc nghiên cứu xây dựng và tập hợp các công cụ (mô hình lý thuyết và phần mềm). Trong lĩnh vực này, tô màu các đối tượng là 1 bài toán quan trọng Thuật toán tô màu bằng dòng quét và tô màu bằng đường biên trong 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.

GIỚI THIỆU

Đồ họa máy tính (Computer graphics) nghiên cứu, xây dựng và tập hợp các công cụ 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[1]

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. Thuật toán tô màu theo dòng quét Scanline

1.1.1. 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

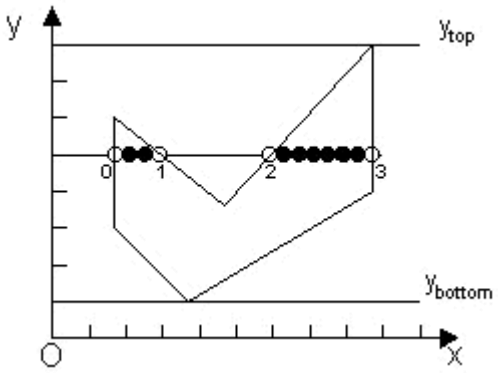
[1] https://bit.ly/3Pznp9Q

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 giới 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 cùa 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 1.2. 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.

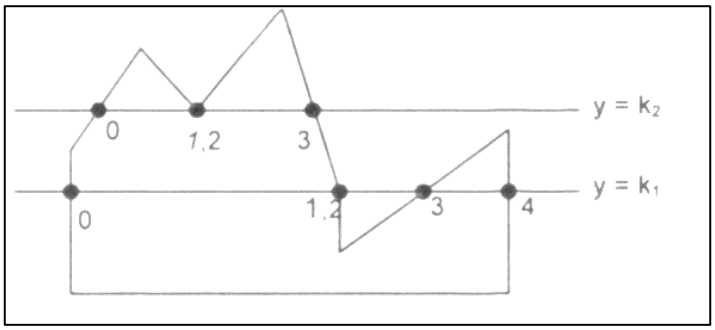
B2.2. Sắp xếp các hoành độ giao điểm theo thử tự tăng dần: $x_0, x_1, \ldots$

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 1.3. 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 1.3.

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.

1.1.2. 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)

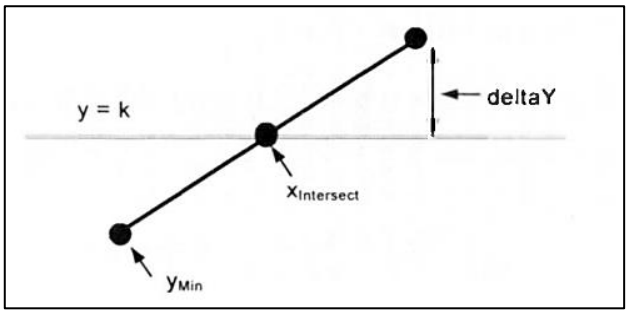
Mỗi cạnh của đa giác được xây dựng từ hai đỉnh kề nhau $P_i(x_i, y_i)$ và $P_{i+1}(x_{i+1}, y_{i+1})$ gồm các thông tin sau:

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 1.4. 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:

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

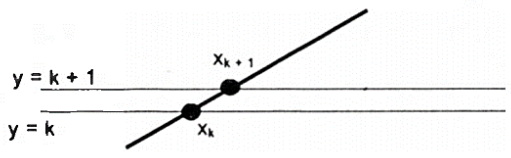
Chính vì vậy, với cách tổ chức của ET (sắp theo thứ tự tăng dần cùa Y_{Min}), điều kiện để chuyển các cạnh từ ET sang AET sẽ là $k > Y_{Min}$ và điều kiện để loại một cạnh ra khỏi AET là deltaY < 0.

1.1.3. 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 $X_{Intersec}$.



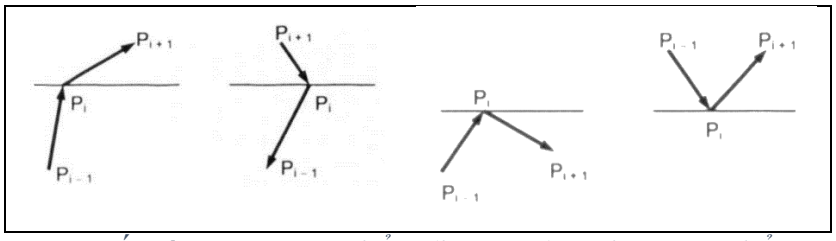
Hình 1.5. Xác định hoành độ giao điểm

1.1.4. 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.

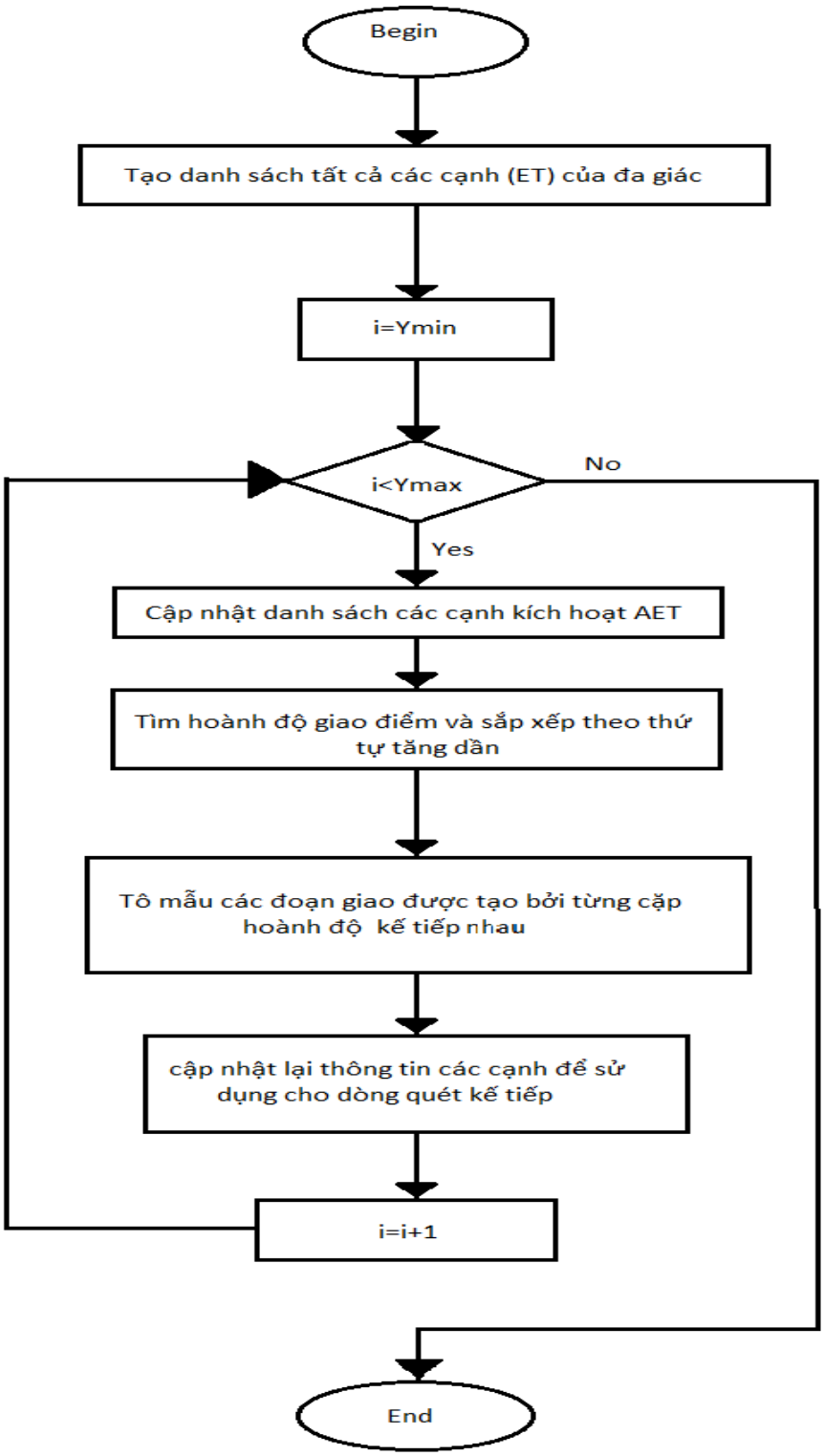
Tính hai giao điểm nếu chiều của hai cạnh kề của đỉnh đó có xu hướng thay đổi, nghĩa là tăng - giảm hay giảm – tăng



Hình 1.6. Quy tắc tính một giao điểm (bên trái) và hai giao điểm (bên phải)

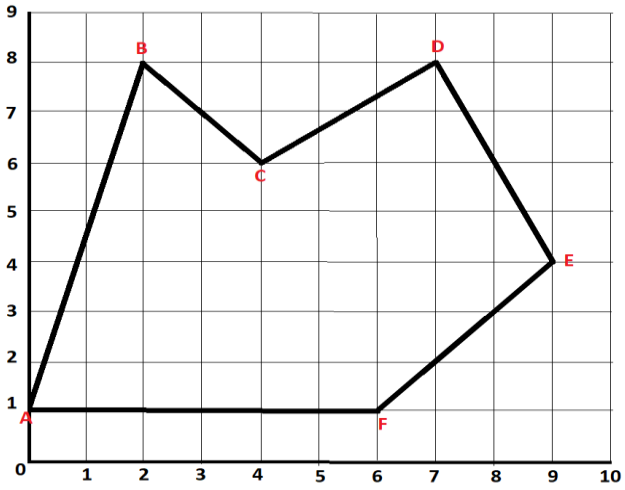
Khi cài đặt không cần phải xét điều kiện này, khi xây dựng dữ liệu cho mỗi cạnh trước khi đưa vào ET, người ta sẽ xử lý các cạnh có đinh tính hai giao điểm bằng cách loại đi một pixel trên cùng của một trong hai cạnh.

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

1.2. Dev C++

Bloodshed Dev-C ++ 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++: [2]

- ✓ 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.

[2] https://www.bloodshed.net/devcpp.html

- ✓ 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.

1.3. Thư viện Graphics.h [3]

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++

[3] https://github.com/SagarGaniga/Graphics-Library

CHƯƠNG 2. CÀI ĐẶT THUẬT TOÁN

2.1. Cài đặt DevC++ và thư viện graphics.h

Tải file cài đặt phần mềm DevC++ theo đường dẫn trong mục 1.3. 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

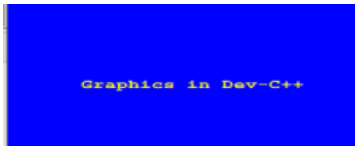
Bước 5: Thay đổi Tools - Complier Option: TDM – GCC 4.9.2 32 bit Release

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 <conio.h>
#include <iostream>
#include <graphics.h>
#include <stdlib.h>
using namespace std;
```

Khai báo biến

```
//============================
struct ToaDo
{
    int x,y;
};
```

Tạo đa giác: Mỗi đỉnh của đa giác bao gồm hoành độ X và tung độ Y được lưu trữ trong ma trận chứa các điểm p[] và khai báo kiểu số nguyên cho V là tổng số cạnh của hình vẽ. Tọa độ các đỉnh của đa giác được nhập trực tiếp từ bàn phím thông qua gọi hàm nhapDaGiac().

```
void nhapDaGiac(ToaDo p[], int v)
{
    int i;
    for(i=0;i<v; i++){
        cout<<"\nNhap toa do dinh "<<i+1<<" : ";
        cout<<"\n\tx["<<(i+1)<<"] = "; cin>>p[i].x;
        cout<<"\n\ty["<<(i+1)<<"] = "; cin>>p[i].y;
    }
    p[i].x=p[0].x;
    p[i].y=p[0].y;
}
```

Kết thúc việc nhập đa giác sẽ tính toán hệ số góc trên mỗi cạnh của đa giác.

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(ToaDo p[], int v)
{
    for(int i=0;i<v;i++)
        line(p[i].x,p[i].y,p[i+1].x,p[i+1].y);
}
```

Thuật toán tô màu đa giác

```
void ScanLine(ToaDo p[], int v)
{
    int xmin,xmax,ymin,ymax,c,mang[50];
    xmin=xmax=p[0].x;
    ymin=ymax=p[0].y;
    for(int i=0;i<v;i++){
        if(xmin>p[i].x) xmin=p[i].x;
        if(xmax<p[i].x) xmax=p[i].x;
        if(ymin>p[i].y) ymin=p[i].y;
        if(ymax<p[i].y) ymax=p[i].y;
    }
    float y;
    y=ymin+0.01;
    while(y<=ymax){ //voi y tang dan tu ymin > ymax, tim cac giao diem cua tung y voi cac cap canh
        int x,x1,x2,y1,y2,tg;
        c=0;    //chi so cua mang phan tu
        for(int i=0;i<v;i++){ //xet tren tat ca ca dinh
            //xet 2 dinh lien ke nhau
            x1=p[i].x;
            y1=p[i].y;
            x2=p[i+1].x;
            y2=p[i+1].y;
            if(y2<y1){ //sap xep lai y cua 2 diem lien tiep
                tg=x1;x1=x2;x2=tg;
                tg=y1;y1=y2;y2=tg;
            }
            //mang giao diem
            if(y<=y2&&y>=y1){
                if(y1==y2)  x=x1; //neu y cua 2 dinh lien tiep trung nhau => bo qua
                else{
                    x=((y-y1)*(x2-x1))/(y2-y1); //he so goc
                    x+=x1;
                }
                if(x<=xmax && x>=xmin)
                    mang[c++]=x;   //cho phan tu c = x sau do c++
            }
        }
        //voi tung y tang dan ta ve luon duong thang noi 2 giao diem
        for(int i=0; i<c;i+=2){
                delay(30);
                line(mang[i],y,mang[i+1],y);
        }
        y++;
    }
}
```

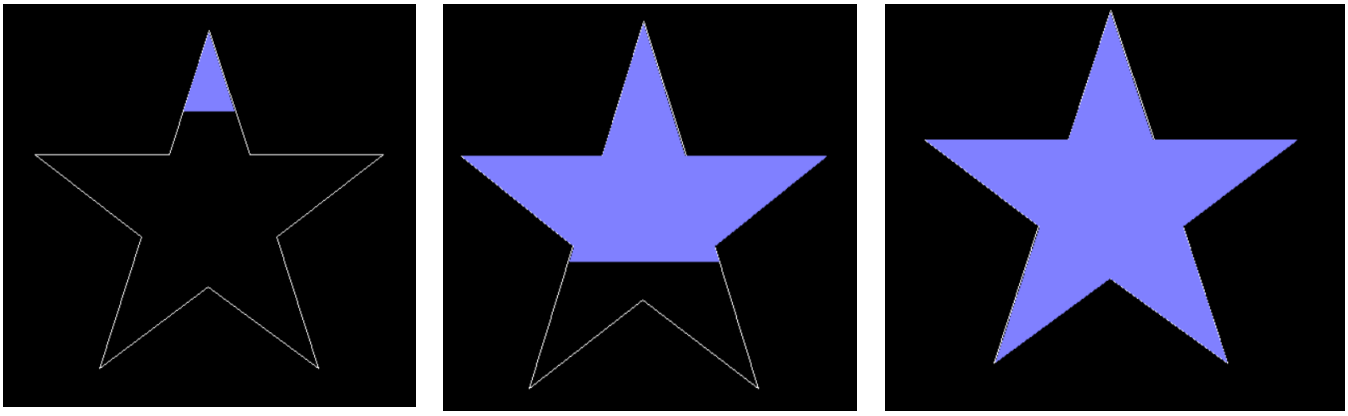
Chương trình chính

```
int main()
{
    int cl,v;
    do{
        cout<<"\n Nhap so dinh da giac:"; cin>>v;
    }while(v<3);
    ToaDo p[v];
    nhapDaGiac(p,v);
    cout<<"\nChon mau (0-15) : "; cin>>cl;
    initwindow(500,600);
    veDaGiac(p,v);
    setcolor(cl);
    ScanLine(p, v);
    getch();
}
```

CHƯƠNG 3. KẾT QUẢ THỰC HIỆN

3. Tô màu theo dòng quét

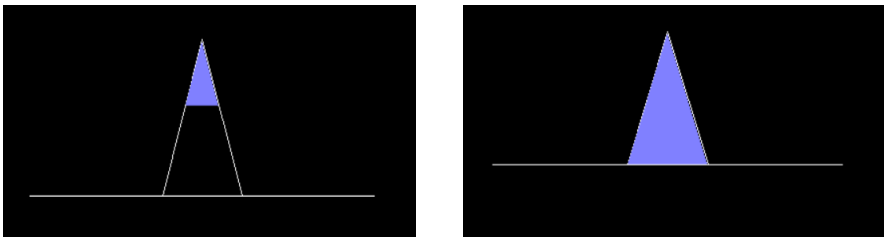
3.1. Kết quả tô màu của thuật toán:



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 9. Thuật toán sẽ dừng lại khi $y=y_{max}$.

3.2. Trường hợp tô màu lỗi:



Hình 3.2. Kết quả của thuật toán tô màu Scanline trường hợp lỗi

Từ hình 3.2, ta chọn 1 hình đa giác với năm cạnh và trong 5 cạnh có 4 cạnh là y bằng nhau.

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