TRƯỜNG ĐẠI HỌC NHA TRANG

**KHOA CÔNG NGHỆ THÔNG TIN**



**CÀI ĐẶT THUẬT TOÁN XÉN TỈA ĐA GIÁC BẰNG**

**THUẬT TOÁN SUTHERLAND - HODGMAN**

**GVHD : ThS. ĐOÀN VŨ THỊNH**

**SVTH : Đậu Quang Lộc**

**MSSV : 59131309**

**Lớp : 59.CNTT-1**

Khánh Hòa, tháng 01 năm 2020

**MỤC LỤC**

[TÓM TẮT 3](#_Toc30108954)

[DANH MỤC HÌNH ẢNH 4](#_Toc30108955)

[I. GIỚI THIỆU 5](#_Toc30108956)

[1.1 Thuật toán xén tỉa 5](#_Toc30108957)

[1.2 Thuật toán xén tỉa đoạn thẳng 6](#_Toc30108958)

[1.2.1 Thuật toán Cohen-Sutherland 7](#_Toc30108959)

[1.2.2 Thuật toán Cyrus-beck 9](#_Toc30108960)

[1.3 Thuật toán xén tỉa đa giác 10](#_Toc30108961)

[1.4 Phần mềm Dev C++ và thư viện Graphics 14](#_Toc30108962)

[II. PHƯƠNG PHÁP NGHIÊN CỨU 17](#_Toc30108963)

[2.1 Nhập dữ liệu 17](#_Toc30108964)

[2.2 Vẽ cửa sổ 17](#_Toc30108965)

[2.3 Vẽ đa giác 17](#_Toc30108966)

[2.4 Đoạn thẳng nằm bên trái cửa sổ xén 18](#_Toc30108967)

[2.5 Đoạn thẳng nằm trên bên phải cửa sổ xén 19](#_Toc30108968)

[2.6 Đoạn thẳng nằm trên biên trên cửa sổ xén 20](#_Toc30108969)

[2.7 Đoạn thẳng nằm trên biên dưới cửa sổ xén 21](#_Toc30108970)

[2.8 Đoạn thẳng nằm ngoài cửa sổ xén 22](#_Toc30108971)

[2.9 Đoạn thẳng nằm trong cửa sổ xén 22](#_Toc30108972)

[III. KẾT QUẢ 23](#_Toc30108973)

[IV. THẢO LUẬN 24](#_Toc30108974)

# TÓM TẮT

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. Trong lĩnh vực kỹ thuật máy tính, Đồ Họa Máy Tính càng ngày càng phát triển mạng mẽ. 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 (game, giải trí,…), 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,…), …Trong Đồ Họa Máy Tính có nhiều thuật toán khác nhau. Xong mỗi thuật toán lại tỏ ra có những ưu việt và hạn chế riêng đối với từng bài toán cụ thể. Và để phục vụ cho việc tìm hiểu thêm về các thuật toán trong Đồ Họa Máy Tính, em với sự hướng dẫn của Giảng viên – Đoàn Vũ Thịnh – Khoa CNTT – Trường ĐH Nha Trang đã thực hiện đề tài: “Cài đặt thuật toán xén tỉa một đa giác bằng thuật toán Sutherland-Hodgman”.

# DANH MỤC HÌNH ẢNH

[Hình 1.1. Điểm nằm trong 5](#_Toc30144285)

[Hình 1.2. Ví dụ minh họa cho ứng dụng thuật toán xén tỉa đa giác trong PowerPoint 2019 6](#_Toc30144286)

[Hình 1.3. Các đoạn thẳng nằm trong hoặc nằm ngoài 6](#_Toc30144287)

[Hình 1.4. Các đoạn thẳng cắt cửa sổ 6](#_Toc30144288)

[Hình 1.5. Mã vùng quy định vị trí tương đối của vùng so với cửa sổ 7](#_Toc30144289)

[Hình 1.6. Mã vùng được xác định theo 9 vùng 8](#_Toc30144290)

[Hình 1.7. Cửa sổ cắt tỉa đoạn thẳng 9](#_Toc30144291)

[Hình 1.8. Hiệu chỉnh bản TDM-GCC phù hợp với phiên bản 15](#_Toc30144292)

[Hình 1.9. Thiết lập project khởi động đồ họa sử dụng thư viện graphics.h 16](#_Toc30144293)

[Hình 1.10. Code minh họa 16](#_Toc30144294)

[Hình 1.11. Code sau khi chạy 16](#_Toc30144295)

[Hình 3.1 Kết quả của 4 trường hợp cắt tỉa đa giác 23](#_Toc30144298)

[Hình 3.2 Đa giác sau khi đã xén tỉa 23](#_Toc30144299)

[Hình 4.1. Hạn chế của thuật toán Sutherland-Hodgman 24](#_Toc30144301)

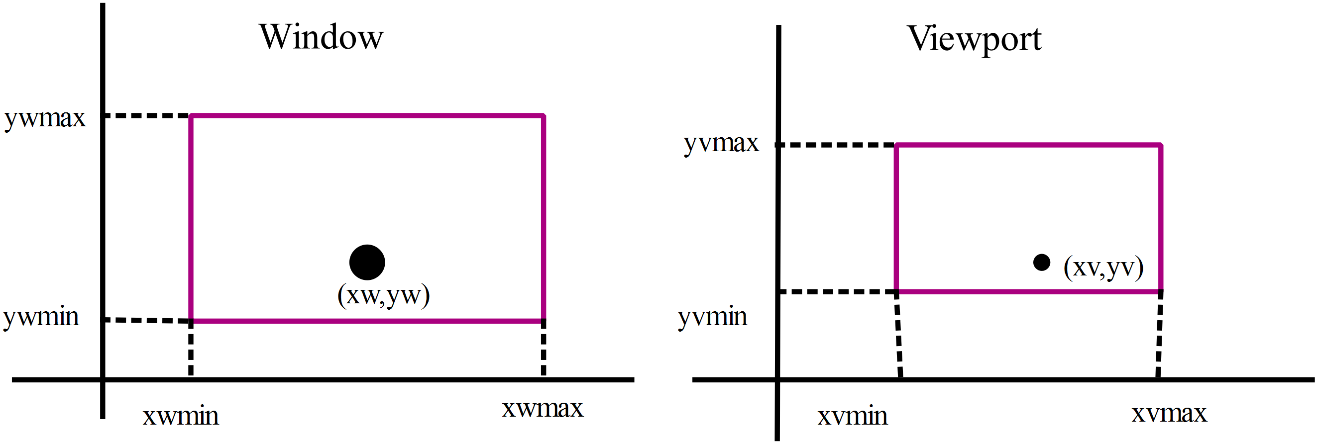
# I. GIỚI THIỆU

Đồ họa máy tính là tất cả những gì liên quan đến việc sử dụng máy tính để phát sinh ra hình ảnh. Các vấn đề liên quan đến công việc này bao gồm: tạo, lưu trữ, thao tác trên mô hình và các hình ảnh.

Theo nghĩa rộng hơn, đồ họa máy tính là phương pháp và công nghệ dùng trong việc chuyển đổi qua lại giữa dữ liệu và hình ảnh trên màn hình bằng máy tính. Đồ họa máy tính hay kĩ thuật đồ họa máy tính còn được hiểu dưới dạng phương pháp và kĩ thuật tạo hình ảnh từ các mô hình toán học mô tả các đối tượng hay dữ liệu lấy được từ các đối tượng trong thực tế. Thuật ngữ “đồ họa máy tính” (computer graphics) được đề xuất bởi một chuyên gia người Mĩ tên là William Fetter vào năm 1960. Khi đó ông đang nghiên cứu xây dựng mô hình buồng lái máy bay cho hãng Boeing. William Fetter đã dựa trên các hình ảnh 3 chiều của mô hình người phi công trong buồng lái để xây dựng nên mô hình buồng lái tối ưu cho máy bay Boeing. Đây là phương pháp nghiên cứu rất mới vào thời kì đó. Phương pháp này cho phép các nhà thiết kế quan sát một cách trực quan vị trí của người lái trong khoang buồng lái. William Fetter đã đặt tên cho phương pháp của mình là computer graphics...

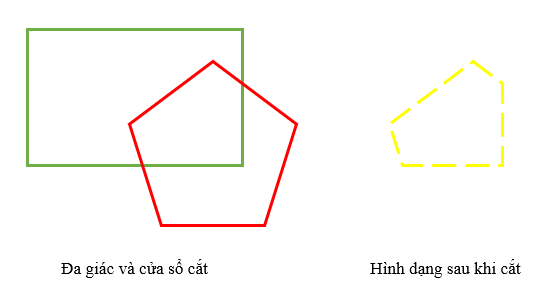
## 1.1 Thuật toán xén tỉa

Xén tỉa là tiến trình xác định các điểm của một đối tượng nằm trong hay ngoài cửa sổ hiển thị. Nằm trong được hiển thị, nằm ngoài loại bỏ. Việc loại từng điểm ảnh của đối tượng thường chậm nhất là khi đối tượng mà phần lớn nằm ngoài cửa sổ hiển thị.



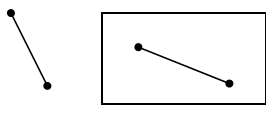
Hình 1.1. Điểm nằm trong

Kĩ thuật xén tỉa hình ảnh được ứng dụng rộng rãi trong các phần mền đồ họa như: Autocad, Photoshop, Adobe Illustrator, hay phần mềm Power Point 2019 (Hình 1.2).

Hình 1.2. Ví dụ minh họa cho ứng dụng thuật toán xén tỉa đa giác trong PowerPoint 2019

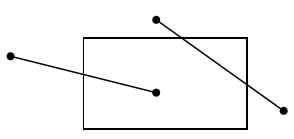
Nhưng trước hết muốn xén tỉa một đa giác thì chúng ta nên tìm hiểu cách xén tỉa một đoạn thẳng bởi vì xén tỉa đoạn thẳng là nền tảng để chúng ta đi xén tỉa một đa giác.

## 1.2 Thuật toán xén tỉa đoạn thẳng

****Các đoạn thẳng không cắt cửa sổ thị: nằm trong hoàn toàn, hoặc ngoài hoàn toàn.

Hình 1.3. Các đoạn thẳng nằm trong hoặc nằm ngoài

Nếu đoạn thẳng cắt cửa sổ thì phân chia qua điểm cắt phần nằm trong và phần nằm ngoài



Hình 1.4. Các đoạn thẳng cắt cửa sổ

Các đoạn thẳng có thể rơi vào các trường hợp sau:

**TH1:** Hiển thị (visible): cả hai đầu cuối của đoạn thẳng đều nằm bên trong cửa sổ.

**TH2:** Không hiển thị (invisible): đoạn thẳng xác định nằm ngoài cửa sổ. Điều này xảy ra khi đoạn thẳng từ (x1,y1) đến (x2,y2) thoả mãn bất kỳ một trong bốn bất đẳng thức sau:

x1,x2 >xmax y1,y2 > ymax

x1,x2 < xmin y1,y2 < ymin

Thao tác xén hình là một trong nhưng thao tác cơ bản của quá trình hiển thị đối tượng. Ý tưởng chung của thao tác xén tỉa đoạn thẳng đó là phép toán tìm giao điểm giữa đoạn thẳng với biên. Với các đoạn thẳng đặt biệt như nằm trong hoàn toàn hoặc nằm ngoài hoàn toàn cửa sổ (ví dụ như là hình 1.3, 1.4). Đối với các đoạn thẳng cắt cửa sổ thì phải tìm giao điểm.

Người ta thường sử dụng phương trình tham số của đoạn thẳng trong việc tìm giao điểm của đoạn thẳng với cửa sổ.

X = x1+t( x2-x1) = x1 + tDx1, Dx = x2-x1

Y = y1+t( y2-y1) = y1 + tDy1, Dy = y2-y1 0<=t<=1

1.2.1 Thuật toán Cohen-Sutherland

Đây là một trong những thuật toán ra đời sớm nhất. Bằng cách kéo dài các biên của cửa sổ, người ta chia mặt phẳng thành chín vùng gồm cửa sổ và 8 vùng xung quanh.

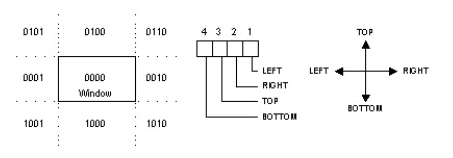
**Khái niệm mã vùng:** mã vùng là một con số 4 bit nhị phân sẽ được gán cho mỗi vùng để mô tả vị trí tương đối của vùng đó so với cửa sổ. Bằng cách đánh số từ 1 đến 4 theo thứ tự từ phải sang trái, các bit theo mã vùng được dùng theo quy ước sau để chỉ một trong bốn vị trí tương đối của vùng so với cửa sổ bao gồm: trái, phải, trên, dưới.

**Bit 1:** trái (LEFT)

**Bit 2:** phải (RIGHT)

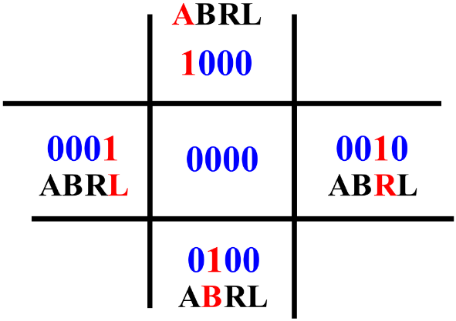
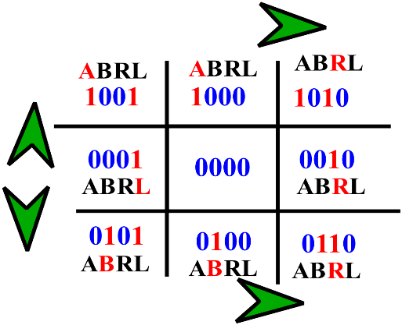
**Bit 3:** trên (TOP)

**Bít 4:** dưới (BOTTOM)



Hình 1.5. Mã vùng quy định vị trí tương đối của vùng so với cửa sổ

Mã vùng được xác định theo 9 vùng của mặt phẳng mà các điểm cuối nằm vào đó. Một bít được cài đặt true (1) hoặc false (0).



Hình 1.6. Mã vùng được xác định theo 9 vùng

**Các trường hợp xén:**

|  |  |
| --- | --- |
| ***Trường hợp 1:*** Đoạn thẳng có 2 điểm hoàn toàn nằm trong cửa sổ nên **không cần xén.** |  |
| ***Trường hợp 2:*** Đoạn thẳng có 2 điểm cùng nằm ngoài về một phía ngoài của cửa sổ và sẽ bị xén mất. |  |
| ***Trường hợp 3:*** Nếu đoạn thẳng cắt biên cửa sổ bên trái thì tọa độ tại điểm đó được xác định như sau: |  |
| ***Trường hợp 4:*** Nếu đoạn thẳng cắt biên cửa sổ bên phải thì tọa độ tại điểm đó được xác định như sau: |  |
| ***Trường hợp 5:*** Nếu đoạn thẳng cắt biên cửa sổ bên trên thì tọa độ tại điểm đó được xác định như sau: |  |
| ***Trường hợp 6:*** Nếu đoạn thẳng cắt biên cửa sổ bên dưới thì tọa độ tai điểm đó được xác định như sau: |  |

**Xác định chiều đoạn thẳng:**

**TH1: đoạn thẳng cắt biên bên trái:** Nếu xP1 <= xwmin và xP2 >= xwmin  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu OUT2IN.

**TH2: đoạn thẳng cắt biên bên phải:** Nếu xP1 > xwmax và xP2 <= xwmax  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu OUT2IN.

**TH3: đoạn thẳng cắt biên bên trên:** Nếu yP1 > ywmax và yP2 <= ywmax  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu OUT2IN.

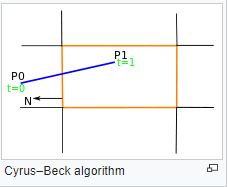
**TH4: đoạn thẳng cắt biên bên dưới:** Nếu yP1 < ywmin và yP2 >= ywmin  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu OUT2IN.

1.2.2 Thuật toán Cyrus-beck

Thuật toán Cyrus-Beck là thuật toán cắt dòng tổng quát. Nó được thiết kế để hiệu quả hơn thuật toán Cohen-Sutherland, sử dụng cắt lặp đi lặp lại. Cyrus-Beck là một thuật toán tổng quát và có thể được sử dụng với cửa sổ cắt đa giác lồi, không giống như Cohen-Sutherland, chỉ có thể được sử dụng trên một khu vực cắt hình chữ nhật.

Ở đây phương trình tham số của một đường trong mặt phẳng:





Hình 1.7. Cửa sổ cắt tỉa đoạn thẳng

Ngoài ra còn có còn có nhiều thuật toán để xén tỉa đa giác khác nhau như: Nicholl-Lee-Nicholl, Fast Clipping , O(lg N), Skala, See also, references,… Các thuật toán xén tỉa này cũng được cải thiện đáng kể về độ chính xác và thời gian xử lí. Xén tỉa đa giác là một trong những phần mở rộng của thuật toán xén tỉa đoạn thẳng. Có thể xem tại đường link: ( <https://en.wikipedia.org/>).

## 1.3 Thuật toán xén tỉa đa giác

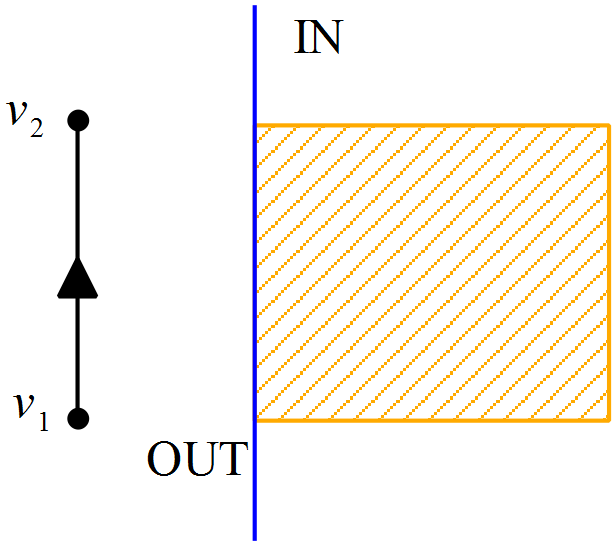
[Ivan Sutherland](https://en.wikipedia.org/wiki/Ivan_Sutherland) và Gary W. Hodgman (1974) đề xuất thuật toán xén tỉa đa giác bằng cách lần lượt mở rộng từng đoạn thẳng của *đa giác lồi* và chỉ chọn các đỉnh từ đa giác cần xén nằm ở phía có thể nhìn thấy. Hạn chế của thuật toán này là nếu đa giác cần xén là đa giác lõm, đa giác mới có thể có các cạnh trùng nhau.

Qui ước xác định chiều các điểm của đa giác theo chiều kim đồng hồ như sau:

Theo qui ước: một đa giác với các đỉnh , ....., (các cạnh là và ) được gọi là theo hướng dương nếu các hình theo thứ tự đã cho tạo thành mạch ngược chiều kim đồng hồ.

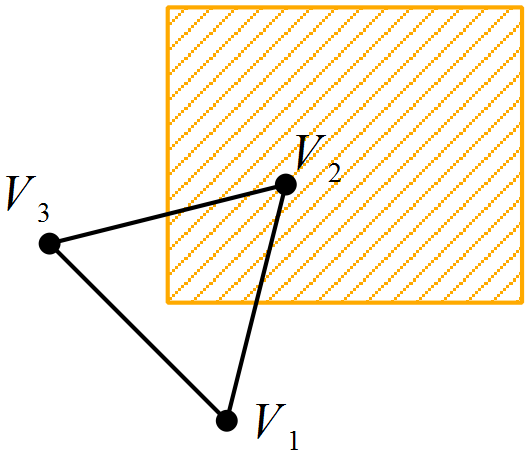
Nếu bàn tay dọc theo bất kỳ cạnh hoặc cũng chỉ về bên trong đa giác.

Quy tắc khác: Tính tổng các cạnh của đa giác: (x2 − x1)(y2 + y1). Nếu kết quả cho số dương thì chiều của đa giác thuận chiều kim đồng hồ và ngược lại.

***Các trường hợp trong giải thuật hogman:***

|  |  |
| --- | --- |
| **TH1:** Nếu đỉnh v1 nằm bên ngoài và đỉnh v2 nằm bên trong của cạnh và có chiều từ trái sang phải thì giao điểm v’1 và v2 được lưu lại. |  |
| **TH2:** Nếu đỉnh v1 nằm bên trong và đỉnh v2 nằm bên ngoài của cạnh và có chiều từ phải sang trái thì giao điểm v’1 và v1 được lưu lại. |  |
| **TH3:** Nếu cả 2 đỉnh v1 và v2 cùng nằm về phía bên trong của cạnh thì cả 2 đỉnh v1 và v2 **đều được** lưu lại. |  |
| **TH4:** Nếu cả 2 đỉnh v1 và v2 cùng nằm về phía bên ngoài của cạnh thì cả 2 đỉnh v1 và v2 **không được** lưu lại. |  |

**Ví dụ minh họa:**

Cho đa giác dương sau và cửa sổ cắt, hãy xác định các đỉnh cắt:

|  |  |
| --- | --- |
| ***Bước 0:*** Áp dụng quy tắc bàn tay trái cho đa giác dương để xác định chiều của các cạnh. |  |
| ***Bước 1: Cắt bên trái***  Xét cạnh V1-V2 (In-In): V2.  Xét cạnh V2-V3 (In-Out): V’2.  Xét cạnh V3-V1(Out-In): V3-V3’. |  |
| ***Bước 2. Cắt bên phải***  Xét cạnh V1-V2 (In-In): V2.  Xét cạnh V2-V’2 (In-In): V’2.  Xét cạnh V’2-V’3 (In-In): V’3.  Xét cạnh V’3-V1(In-In): V1. |  |
| ***Bước 3. Cắt bên trên***  Xét cạnh V1-V2 (In-In): V2.  Xét cạnh V2-V’2 (In-In): V’2.  Xét cạnh V’2-V’3 (In-In): V’’3.  Xét cạnh V’3-V1(Out-Out): NULL. |  |
| ***Bước 4. Cắt bên dưới***  Xét cạnh V1-V2 (Out-In): V’1V2.  Xét cạnh V2-V’2 (In-In): V’2.  Xét cạnh V’2-V’3 (In-Out): V’’2.  Xét cạnh V’3-V1(Out-Out): NULL. |  |

**Ví dụ 2:**

|  |  |
| --- | --- |
| Cho đa giác ABCD với tọa độ lần lượt là (20,200), (120,350), (300,220), (120,80) và cửa sổ cắt xyzt với tọa độ lần lượt là (50,120), (180,120), (180,280), (50,280). |  |

**Bước 1: Ta xét cạnh bên trái**

**Cạnh AB**: nằm ngoài cửa sổ theo chiều Out-In

Tìm giao điểm A’ thuộc đường thẳng AB

Vậy A’(50,245)

**Cạnh DA**: nằm ngoài cửa sổ theo chiều In-Out

Tìm giao điểm A” thuộc đường thẳng DA

Vậy A”(50,164)

**Bước 2: Ta xét cạnh bên phải**

**Cạnh BC:** nằm ngoài hoàn toàn cửa sổ nên ta không tạo ra điểm mới cho đa giác sau khi xén tỉa

**Cạnh CD:** nằm ngoài cửa sổ và chiều Out-In

Tìm giao điểm C’ thuộc đường thẳng CD

Vậy C’(180,)

**Bước 3: Ta xét cạnh bên trên**

**Cạnh AB**: nằm ngoài cửa sổ chiều In-Out

Tìm giao điểm B’ thuộc đường thẳng AB

Vậy B’()

**Cạnh BC:** nằm ngoài hoàn toàn cửa sổ cắt nên sẽ không tạo điểm mới

**Bước 4: Xét cạnh bên dưới**

**Cạnh DA**: nằm ngoài cửa sổ và chiều Out-In

Tìm giao điểm D’ thuộc đường thẳng DA

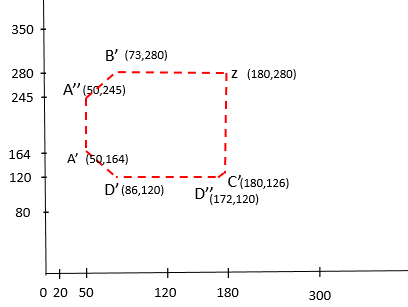
Vậy D’(, 120)

**Cạnh CD:** nằm ngoài cửa sổ và chiều In-Out

Tìm giao điểm D” thuộc đường thẳng CD

Vậy D”(, 120)

Sau khi thực hiện xong ta được đa giác mới A’’B’zC’D”D’A’ có tọa độ lần lượt là (50,245), (73,280), (180,280), (180,126), (172,120), (86,120), (50,164)



[Vatti](https://en.wikipedia.org/wiki/Vatti_clipping_algorithm) được phát triển vào năm 1992 bởi Bala R. vatti, thuật toán giải quyết hạn chế của các loại đa giác có thể được sử dụng làm đối tượng hoặc clip. Ngay cả các đa giác phức tạp (tự giao nhau) và đa giác lõm có thể được xử lý (<https://en.wikipedia.org/wiki/Vatti_clipping_algorithm>).

## 1.4 Phần mềm Dev C++ và thư viện Graphics

Trước C++, ngôn ngữ lập trình C được phát triển trong năm 1972 bởi Dennis Ritchie tại phòng thí nghiệm Bell Telephone, C chủ yếu là một ngôn ngữ lập trình hệ thống, một ngôn ngữ để viết ra hệ điều hành. Vào năm 1999, ủy ban ANSI đã phát hành một phiên bản mới của ngôn ngữ lập trình C, gọi là C99.

C++ được tạo ra bởi Bjarne Stroustrup – một nhà khoa học máy tính người Đan Mạch tại phòng thí nghiệm AT&T Bell vào năm 1979, được ISO công nhận vào năm 1998, lần phê chuẩn tiếp theo vào năm 2003 (người ta gọi là C++ 03).

Hai lần cập nhật gần đây nhất là C++ 11 và C++ 14 (được phê chuẩn vào năm 2011 và 2014). Phiên bản C++ 17 đã được công bố, dự đoán sẽ hoàn thành trong năm 2017. C++ là một phiên bản mở rộng của ngôn ngữ lập trình C.

Ngoài ra Dev-C++ còn là công cụ không thể thiếu của bộ môn kĩ thuật đồ họa để hỗ trợ xén tỉa đa giác bằng thuật toán Sutherland-Hodgman.

**Thư viện Graphics.h (**Borland Graphics Interface – còn được biết đến với tên gọi BGI) là một thư viện đồ họa rất phổ biến trên DOS và các máy tính chạy hệ điều hành Windows thời kì đầu như Windows 95, Windows 98.

Thư viện này cung cấp cho người dùng 2 file*: graphics.h*và*graphics.lib* để có thể sử dụng được với ngôn ngữ C/C++ cũng như module graph nếu người dùng sử dụng ngôn ngữ Pascal. Bộ thư viện này đi kèm với IDE Borland C++ 3.1 (1992). Nó chứa một số hàm vẽ ảnh và cách tạo ảnh chuyển động, in chữ.

**Thư viện winbmin** Đồ họa C sử dụng các hàm WinBGIm (Windows 7) có thể được sử dụng để vẽ các hình dạng khác nhau, hiển thị văn bản trong các phông chữ khác nhau, thay đổi màu sắc và nhiều hơn nữa. Thư viện winbgim cho phép bạn sử dụng các thói quen đồ họa BGI và hỗ trợ chuột đơn giản cho các ứng dụng Windows mà bạn viết bằng trình biên dịch mingw32 gnu C ++ của CS1300 hoặc với trình biên dịch Borland C ++ (phiên bản 5.0). Nó cũng có thể hoạt động với các trình biên dịch Windows khác.

**Hướng dẫn cài đặt cài đặt Dev C++ và thư viện Graphics:**

Đầu tiên để cài đặt **Dev C++ phiên bản 4.9.9.2** ta truy cập vào: (<https://bloodshed-dev-c.vi.softonic.com/download>).

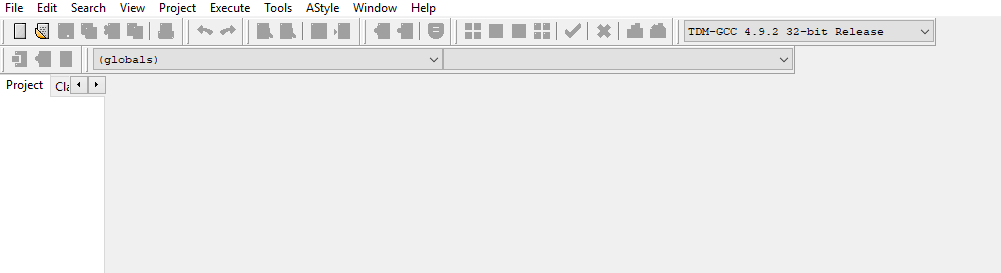
Cài đặt **Graphics** theo đường link: **(**<https://github.com/thinhdoanvu/ComputerGraphics/tree/master/Coding/library>).

Tiến hành cài đặt thư viện gồm có 4 bước:

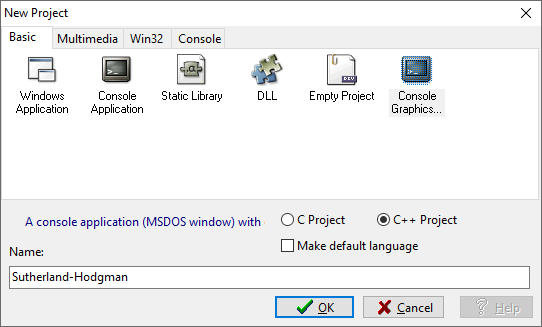
**Bước 1:** Copy tập tin *6-ConsoleAppGraphics, ConsoleApp\_cpp\_graph* đến đường dẫn *C:\Program Files\Dev-Cpp\Templates*

**Bước 2:** Copy tập tin *graphics, winbgim* đến thư mục *C:\Program Files\Dev-Cpp\MinGW64\x86\_64-w64-mingw32\include*

**Bước 3:** Copy tập tin *libbgi.a* đến đường dẫn *C:\Program Files\Dev-Cpp\MinGW64\x86\_64-w64-mingw32\lib*

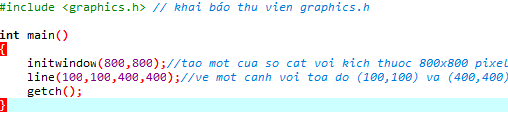
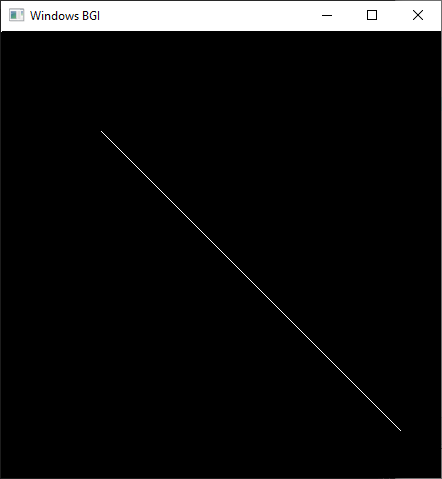
**Bước 4:** Hiệu chỉnh phiên bản TDM-GCC cho phù hợp với hệ điều hành trong phần mềm DevC/C++ theo đường dẫn sau: **Tools - Compiler Option** ta chọn **TDM-GCC 4.9.2 32bit Release**

Hình 1.8. Hiệu chỉnh bản TDM-GCC phù hợp với phiên bản

**Tạo mới dự án (Project)**: đồ họa sử dụng thư viện graphics.h theo các bước sau: File → New → Project → Console Graphics Application với tên project được đánh dấu bởi vùng màu đỏ (Hình 1.9)

Hình 1.9. Thiết lập project khởi động đồ họa sử dụng thư viện graphics.h

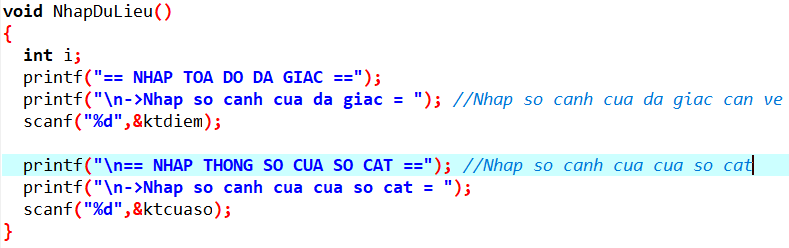
Dưới đây là một đoạn code minh họa trong đồ họa

Hình 1.10. Code minh họa

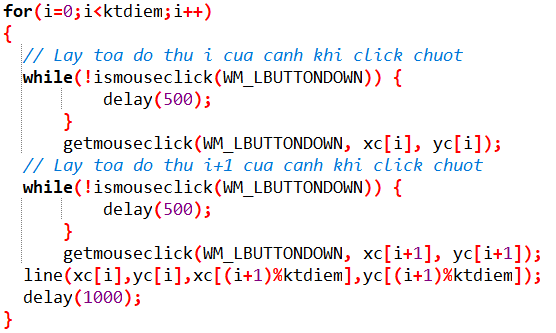
Hình 1.11. Code sau khi chạy

# II. PHƯƠNG PHÁP NGHIÊN CỨU

## 2.1 Nhập dữ liệu

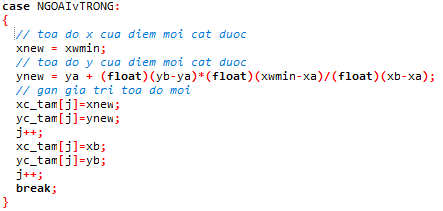


## 2.2 Vẽ cửa sổ

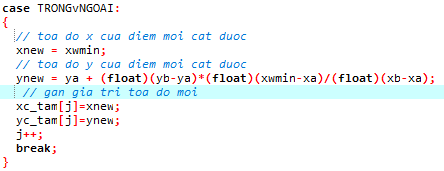


## 2.3 Vẽ đa giác

## 2.4 Đoạn thẳng nằm bên trái cửa sổ xén

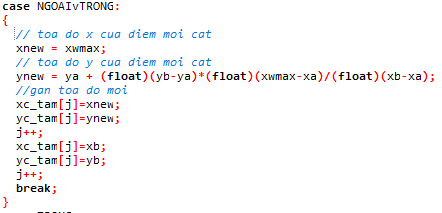
**Trường hợp 1**: P1P2 nằm trên biên trái cửa sổ xén có chiều từ ngoài vào trong.

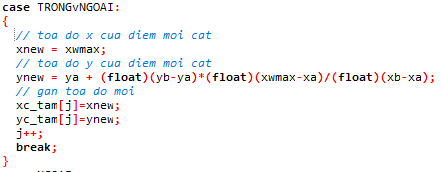
**Trường hợp 2**: P1P2 nằm trên biên trái cửa sổ xén có chiều từ trong ra ngoài.



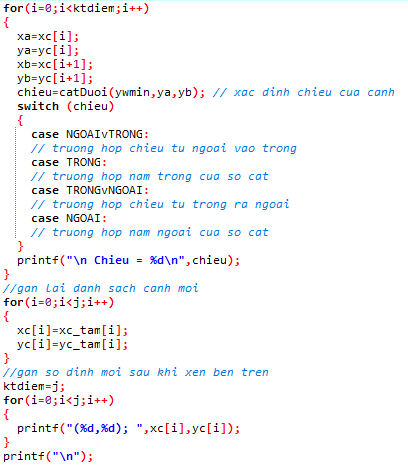
## 2.5 Đoạn thẳng nằm trên bên phải cửa sổ xén

**Trường hợp 1**: P1P2 nằm trên biên phải cửa sổ xén có chiều từ ngoài vào trong.

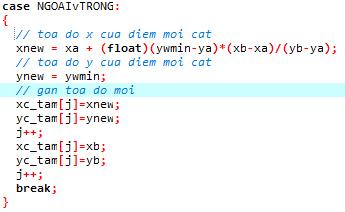


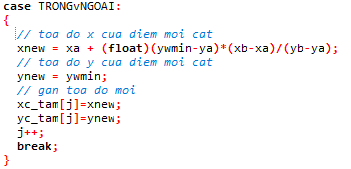
**Trường hợp 2**: P1P2 nằm trên biên phải cửa sổ xén có chiều từ trong ra ngoài.

## 2.6 Đoạn thẳng nằm trên biên trên cửa sổ xén



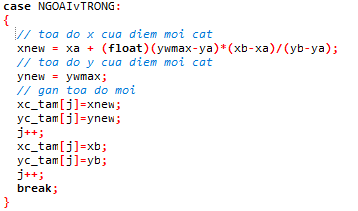
**Trường hợp 1**: P1P2 nằm trên biên trên cửa sổ xén có chiều từ ngoài vào trong.

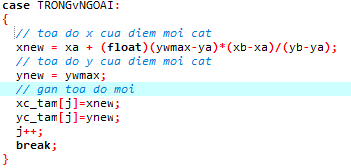


**Trường hợp 2**: P1P2 nằm trên biên trên cửa sổ xén có chiều từ trong ra ngoài.

## 2.7 Đoạn thẳng nằm trên biên dưới cửa sổ xén

**Trường hợp 1**: P1P2 nằm trên biên dưới cửa sổ xén có chiều từ ngoài vào trong.

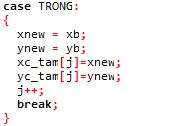


**Trường hợp 2**: P1P2 nằm trên biên dưới cửa sổ xén có chiều từ ngoài vào trong.

## 2.8 Đoạn thẳng nằm ngoài cửa sổ xén



## 2.9 Đoạn thẳng nằm trong cửa sổ xén



# III. KẾT QUẢ

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | D |

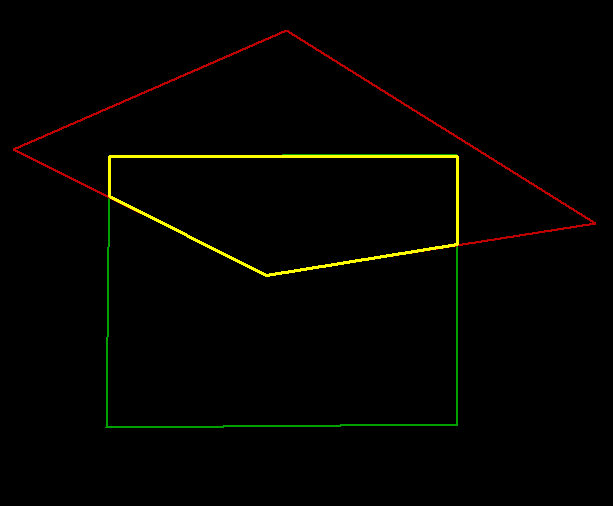
Hình 3.1 Kết quả của 4 trường hợp cắt tỉa đa giác

**Hình A:** trường hợp cắt phần bên dưới của đa giác phần màu vàng là đa giác sau khi cắt đi phần dưới.

**Hình B:** trường hợp cắt bên trái của đa giác phần màu đỏ là phần bị cắt bỏ, phần màu vàng là đa giác sau khi cắt bên trái.

**Hình C:** trường hợp cắt bên phải của đa giác phần mầu đỏ là phần bị cắt bỏ, phần màu vàng là đa giác sau khi cắt bên phải.

**Hình D:** trường hợp cắt phía trên của đa giác phần màu đỏ là phần bị cắt bỏ, phần màu vàng là đa giác sau khi cắt bên trên.

**Đa giác sau khi xén tỉa.**

Hình 3.2 Đa giác sau khi đã xén tỉa

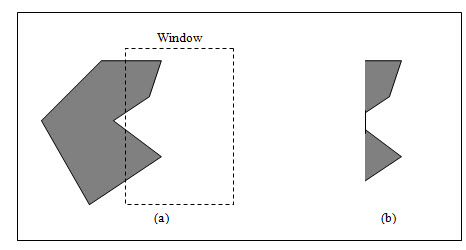
# IV. THẢO LUẬN

Thuật toán xén tỉa đa giác **Sutherland-Hodgman** ra đời từ rất sớm và kế thừa những ưu điểm và cải tiến những hạn chế từ những thuật toán, giải thuật xén tỉa trước đó nên nó rất phổ biến và hữu ích để cắt đa giác.

Bên cạnh những thuận lợi, ưu điểm mà thuật toán Sutherland-Hodgman đem lại trong việc xén tỉa thì nó cũng có những hạn chế, nhược điểm chưa được khắc phục như đòi hỏi một lượng bộ nhớ đáng kể. Kết quả sau khi thuật toán này thực hiện được lưu trữ trong bộ nhớ. Vì vậy, lãng phí bộ nhớ để lưu trữ đa giác trung gian.

Và điểm hạn chế thứ hai chính là vấn đề của bài toán trên, đối với đa giác lõm thì thuật toán Sutherland-Hodgman thì khi cắt nó có thể có các cạnh trùng nhau.

Trong ví dụ dưới đây ta thấy sau khi thực hiện việc cắt tỉa đa giác ở hình a) thì ta được hai đa giác tách biệt nối với nhau bằng một đoạn thẳng ở hình b) nhưng đoạn thẳng được nối ở giữa là dư thừa (nguồn <https://voer.edu.vn/>).



Hình 4.1. Hạn chế của thuật toán Sutherland-Hodgman

Thuật toán cắt đa giác **Weiler-Atherton** là một thuật toán được thực hiện để cho phép thậm chí là có thể cắt các thuật toán lõm. Không giống như thuật toán cắt đa giác của Sutherland-Hodgman, thuật toán này có thể cắt các đa giác lõm mà không để lại bất kỳ dư lượng nào phía sau.