**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**CÀI ĐẶT THUẬT TOÁN XÉN TỈA ĐA GIÁC BẰNG THUẬT TOÁN SUTHERLAND-HODGMAN**

GVHD : Đoàn Vũ Thịnh

SVTH : Nguyễn Thị Hồng Quanh-59132005

Đàng Nữ Ngọc Điệp-59130322

Lớp :59CNTT-1

Khánh Hòa, tháng 01 năm 2020

**MỤC LỤC**

Nội dung

[TÓM TẮT 4](#_Toc30145526)

[1. GIỚI THIỆU 5](#_Toc30145527)

[1.1. Thuật toán xén tỉa đoạn thẳng 5](#_Toc30145528)

[1.1.1 Thuật toán Cohen- Sutherland 5](#_Toc30145529)

[1.2. Thuật toán xén tỉa đa giác 9](#_Toc30145530)

[1.3. Phần mềm DevC/C++ và thư viện graphics.h 11](#_Toc30145531)

[2. PHƯƠNG PHÁP 12](#_Toc30145532)

[2.1. Cài đặt DevC/C++ và thư viện graphics.h 12](#_Toc30145533)

[2.2. Nhập dữ liệu 14](#_Toc30145534)

[2.3. Thuật toán xén tỉa đa giác 14](#_Toc30145535)

[2.3.1. xác định chiều của các cạnh 14](#_Toc30145536)

[2.3.1. Đoạn thẳng nằm trên biên trái cửa sổ cắt 16](#_Toc30145537)

[2.3.2. Đoạn thẳng nằm trên biên phải cửa sổ cắt 17](#_Toc30145538)

[2.3.3. Đoạn thẳng nằm trên biên trên cửa sổ cắt 18](#_Toc30145539)

[2.3.4. Đoạn thẳng nằm trên biên dưới cửa sổ cắt 19](#_Toc30145540)

[3. KẾT QUẢ 20](#_Toc30145541)

[3.1 DevC/C++ và thực viện graphics.h 20](#_Toc30145542)

[3.3 Thuật toán xén tỉa đa giác 20](#_Toc30145543)

[3.3.1. Đa giác giao biên cửa sổ cắt 20](#_Toc30145544)

[4. KẾT LUẬN 21](#_Toc30145545)

[TÀI LIỆU THAM KHẢO 22](#_Toc30145546)

# TÓM TẮT

Trong bài báo cáo, nhóm đã đưa ra ra khái niệm về xén tỉa, cũng như khái niệm về xén tỉa đa giác và cửa sổ cắt. Ngoài ra còn đề cập đến các phần mềm đã ứng dụng việc xén tỉa đa giác. Nhìn chung, việc xén tỉa đa giác là một bài tập lớn trong đó tập hợp các bài toán nhỏ là xén tỉa các cạnh của đa giác (xén tỉa đoạn thẳng). Vì vậy, nhóm đã giới thiệu tổng quát về các thuật toán xén tải đoạn thẳng cũng như các thuật toán xén tỉa đa giác, phần mềm và các thư viện để hỗ trợ việc cài đặt thuật toán xén tải.

Trong phần phương pháp đã chỉ rõ các bước để cài đặt phần mềm DEC V/C++, và thư viện hỗ trợ thư viện graphics.h. Nêu rõ cách bước thực hiện của thuật toán Sutherland- Hodgman và có đoạn code minh họa.

Dựa vào phần phương pháp sẽ cho ra phần kết quả, với mỗi phương pháp xén tỉa khác nhau thì sẽ cho ra các kết quả khác nhau.

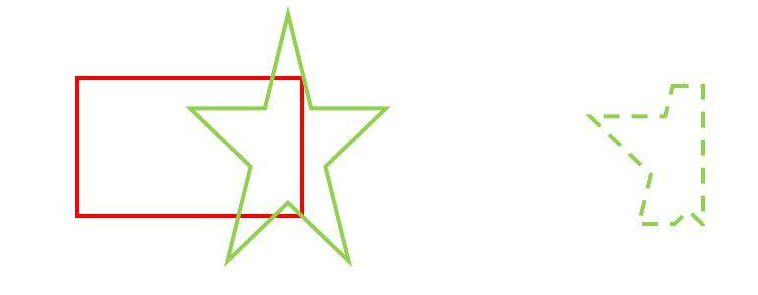
Giống với những thuật toán khác, thì thuật toán Sutherland – Hogman cũng có những ưu – nhược điểm. Vậy nên trong phần cuối-thảo luận, nhóm đã đề cập đến các ưu điểm và hạn chế của thuật toán, từ những hạn chế đó thì những thuật toán sau sẽ khắc phục hạn chế của thuật toán trước.

# GIỚI THIỆU

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.

Thao tác loại bỏ các phần hình ảnh nằm ngoài một vùng cho trước được gọi là xén hình. Vùng được dùng để xén hình gọi là cửa sổ cắt (clip window). Tùy thuộc vào từng ứng dụng cụ thể mà cửa sổ xén có thể có dạng là đa giác hay là đường cong khép kín. Kỹ thuật xén tỉa hình ảnh được ứng dụng rộng rãi trong các phần mềm đồ họa như: Autocad(<https://www.autodesk.com/products/autocad/overview>), Photoshop(<https://taimienphi.vn/download-adobe-photoshop-1639>), Adobe Illustrator([https://www.adobe.com/sea/products/illustrator.html](https://www.adobe.com/sea/products/illustrator.html?sdid=YB1TGRSV&mv=search&s_kwcid=AL!3085!10!79096184450641!79096306250771&ef_id=WvwhngAAALJZuxMJ:20200111082307:s)), hay phần mềm Power Point 2013 (Hình 1).



Hình 1.1. Ví dụ minh họa cho ứng dụng thuật toán xén tỉa đa giác trong phần mềm MS PowerPoint 2013

Trong đó, phần màu xanh là đa giác cần xén, vùng màu đỏ (nét liền) là cửa sổ cắt và vùng nét đứt là kết quả sau khi xén tỉa.

Để đơn giản, thuật toán xén tỉa đa giác là trường hợp mở rộng của xén tỉa đoạn thẳng.

## Thuật toán xén tỉa đoạn thẳng

Trong đồ họa máy tính xén tỉa đoạn thẳng là quá trình bỏ đoạn thẳng hoặc một phần đoạn thẳng bên ngoài cửa sổ cắt. Thông thường đoạn thẳng hoặc một phần nằm ngoài cửa sổ cắt sẽ bị xóa.

1.1.1 Thuật toán Cohen- Sutherland

Thuật toán chia không gian hai chiều thành 9 vùng và sau đó xác định hiệu quả các đường và các phần của đường có thể nhìn thấy ở vùng trung tâm (khung nhìn).Thuật toán được phát triển vào năm 1967 trong quá trình mô phỏng chuyến bay của Daniel Cohen và Ivan Sutherland là thuật toán xén tỉa đoạn thẳng lâu đời và thông dụng nhất. (<https://en.wikipedia.org/wiki/Cohen%E2%80%93Sutherland_algorithm>).

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

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

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 màn bất kỳ một trong bốn bất đẳng thức sau:

x1,x2 >xmax y1,y2 > ymax

x1,x2 < xmin y1,y2 < ymin

Giả sử của sổ xén (clip window) là cửa sổ hình chữ nhật được định nghĩa bởi 2 điểm như hình 1.1. Trong đó:

|  |  |
| --- | --- |
| * Điểm dưới bên trái (Xmin, Ymin) * Điểm trên bên phải (Xmax, Ymax) | Hình 1.2. Cửa sổ xén tỉa đoạn thẳng |

Trong bài toán xén tỉa đoạn thẳng, thì yêu cầu của bài toán là loại bỏ phần đoạn thẳng nằm ngoài cửa sổ xén.

**Có 6 trường hợp xén đoạn thẳng sau đây:**

***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 phía 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 tại điểm đó được xác định như sau:

|  |  |  |
| --- | --- | --- |
| **Trường hợp 1:** | **Trường hợp 2:** | **Trường hợp 3:** |
| **Trường hợp 4:** | **Trường hợp 5:** | **Trường hợp 6:** |

Khi đoạn thẳng cắt biên của cửa sổ cắt, thì sẽ chia ra làm hai trường hợp là chiều từ ngoài vào trong hoặc chiều từ trong ra ngoài.

**Các trường hợp chiều của đoạn thẳng:**

**Trường hợp 1: đoạn thẳng cắt biên bên trái**

Nếu x1 <= xwmin và x2 >= xwmin  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu OUT2IN

Nếu x1 >= xwmin và x2 <= xwmin  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu IN2OUT

**Trường hợp 2: đoạn thẳng cắt biên bên phải**

Nếu x1 > xwmax và x2 <= xwmax  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu OUT2IN

Nếu x1 >= xwmin và x2 <= xwmin  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu IN2OUT

**Trường hợp 3: đoạn thẳng cắt biên bên trên**

Nếu y1 > ywmax và y2 <= ywmax  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu OUT2IN

Nếu y1 <= ywmax và y2 > ywmax thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu IN2OUT

**Trường hợp 4: đoạn thẳng cắt biên bên dưới**

Nếu y1 < ywmin và y2 >= ywmin  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu OUT2IN

Nếu y1 >= ywmin và y2 < ywmin thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu IN2OUT

**Một số thuật toán xén tỉa đoạn thẳng phổ biến khác:**

**Thuật toán Cyrus-beck** được phát triển vào năm 1978 bởi [Mike Cyrus](https://www.sciencedirect.com/science/article/abs/pii/0097849378900213#!) và [Jay Beck](https://www.sciencedirect.com/science/article/abs/pii/0097849378900213#!). Đây là thuật toán cắt đoạn thẳng tổng quát sử dụng với cửa sổ cắt đa giác lồi, không giống như [**thuật toán Cohen- Sutherland**](https://en.wikipedia.org/wiki/Cohen%E2%80%93Sutherland_algorithm), chỉ có thể được sử dụng trên một khu vực cắt hình chữ nhật. (<https://en.wikipedia.org/wiki/Cyrus%E2%80%93Beck_algorithm>)

**Thuật toán Liang-Barsky** (1983) được phát triển bởi Liang, You-Dong và Barsky, Brian A. Thuật toán này sử dụng phương trình tham số của một dòng và bất đẳng thức mô tả phạm vi của cửa sổ cắt để xác định các giao điểm giữa đoạn thẳng và cửa sổ cắt. Thuật toán này hiệu quả hơn so với Cohen-sutherland, nhưng Cohen-sutherland xác định vị trí tương đối của đoàn thẳng so với vùng nhanh hơn nhiều, còn thuật toán này xem xét các đoạn thẳng cần xén tỉa sẽ hoàn toàn vào hoặc ra khỏi của sổ cắt. Thuật toán Liang-Barsky hiệu quả hơn phiên bản Cyrus-Beck vì thử nghiệm loại bỏ tầm thường bổ sung có thể tránh tính toán cả bốn giá trị tham số cho các đường không giao nhau với hình chữ nhật clip. *(*[*https://stackoverflow.com/questions/48352036/how-can-i-measure-the-overlap-between-a-line-and-a-rectangle*](https://stackoverflow.com/questions/48352036/how-can-i-measure-the-overlap-between-a-line-and-a-rectangle)*)*

**Thuật toán Nicholl-Lee-Nicholl** được phát triển vào năm 1987 bởi Tina M.Nicholl, DT Lee, Robin A. Nicholl. Đây là thuật toán cắt dòng nhanh làm giảm khả năng cắt một đoạn thẳng đơn nhiều lần, như có thể xảy ra trong [thuật toán Cohen-Sutherland](https://en.wikipedia.org/wiki/Cohen%E2%80%93Sutherland_algorithm).(*<http://what-when-how.com/computer-graphics-and-geometric-modeling/clipping-basic-computer-graphics-part-3/>)*

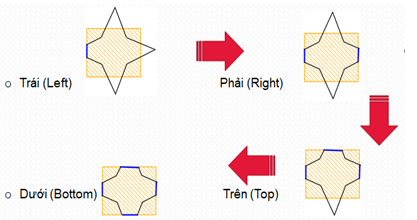
Ngoài ra còn có một số thuật toán khác như **Fast Clipping ,O(lg N) Algorithm, Skala, See also, References**,…

Xén tỉa đa giác là một trong những phần mở rộng của thuật toán xén tỉa đoạn thẳng

## 1.2. Thuật toán xén tỉa đa giác

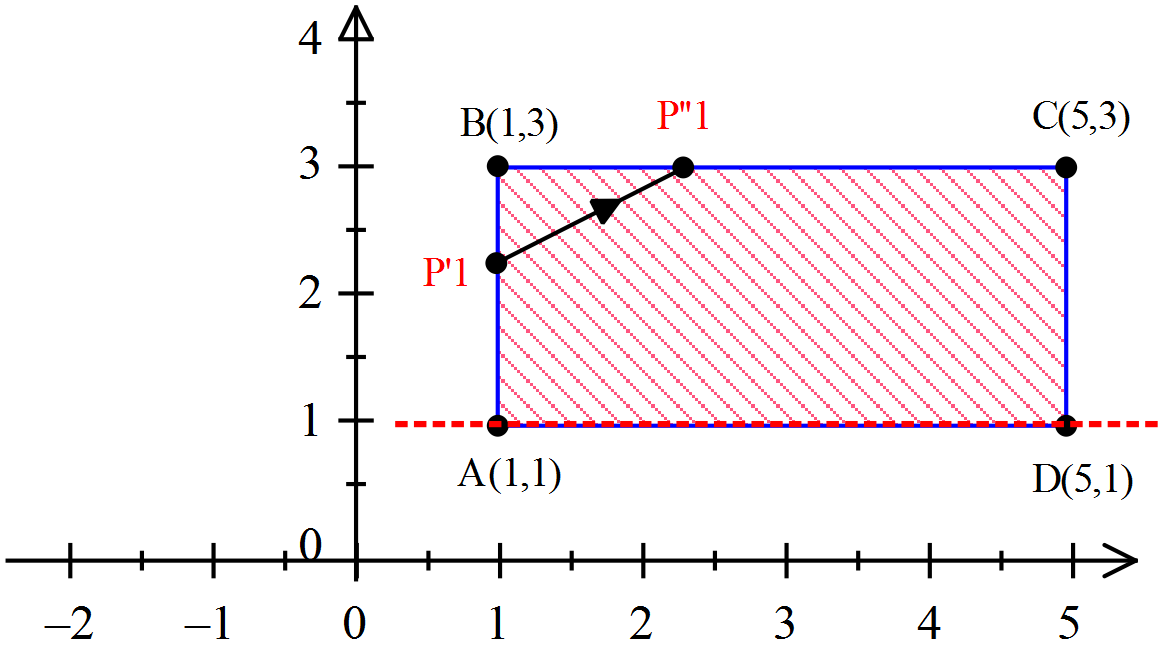
**Thuật toán** [**Sutherland–Hodgman**](https://en.wikipedia.org/wiki/Sutherland%E2%80%93Hodgman): thuật toán được phát triển vào năm 1974 bởi [Ivan Sutherland](https://en.wikipedia.org/wiki/Ivan_Sutherland) , Gary W. Hodgman . Nó hoạt động bằng cách lần lượt mở rộng từng đoạn thẳng của *đa giác lồi* và chỉ chọn các đỉnh từ đa giác cần xén nằm ở phía có thể nhìn thấy. Lưu ý rằng nếu đa giác cần xén là đa giác lõm ở các đỉnh bên ngoài đa giác cắt, đa giác mới có thể có các cạnh trùng nhau (nghĩa là chồng chéo)

Ý tưởng: Cho P1,......,PN là danh sách các đỉnh của đa giác. Cho cửa sổ cắt tỉa ABCD. Thứ tự các trường hợp sẽ được xén như sau:



Hình1.3: thứ tự xén của một đa giác

Ví dụ 2: hãy sử dụng thuật toán Hodgman để cắt xén đoạn thẳng nối P1(-1,2) đến P2(6,4) trên cửa sổ A(1,1), B(1,3), D(3,5) và E(5,3)



Hình.1.4 hình minh họa

Theo thuật toán Hodgman ta xén P’1P’’1 dựa trên từng cạnh.  
➊ AE:

Điểm P’1: nằm bên trong AE

Điểm P’’1: nằm bên trong AE

Chiều In-In

Vậy P’1P”1 được lưu

Tìm giao điểm P’1 thuộc đường thằng P1P2

P’1(1;11/5)

Theo thuật toán Hodgman ta xén P’1P2 dựa trên cạnh.  
➋ DE:

Điểm P’1: nằm bên trái DE

Điểm P2: nằm bên trái DE

Chiều In-In

Vậy P2 được lưu, với P2(4,4)

Theo thuật toán Hodgman ta xén P’1P2 dựa trên cạnh  
➊ BD:

Điểm P’1: nằm bên trong BD

Điểm P2: nằm bên ngoài BD

Chiều In - Out

Vậy P’’1 được lưu

Tìm giao điểm P’’1 thuộc đường thằng P’1P2

P’’1(7/3,3)

Theo thuật toán Hodgman ta xén P’1P’’1 dựa trên từng cạnh.  
➊ AE:

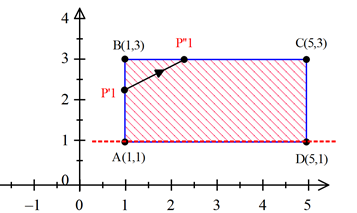
Điểm P’1: nằm bên trong AE

Điểm P’’1: nằm bên trong AE

Chiều In-In

Vậy P’1P”1 được lưu

Kết quả:



Hình 1.5 : Kết quả xén

## 1.3. Phần mềm DevC/C++ và thư viện graphics.h

Để thực hiện các thuật toán xén tỉa đoạn thẳng, xén tỉa đa giác trên máy tính ta sử dụng các phần mềm như devc++, Visual Studio Code,…

Trong bài báo cáo nhóm em sử dụng phần mềm Devc++ phiên bản 4.9.9.2 ) là một bộ công cụ phát triển tích hợp (IDE Integrated Development Environment) các ứng dụng C/C++ thuộc dạng mã nguồn mở. DevCpp dựa  trên  trình biên dịch mã nguồn mở MinGW  (Minimalist GNU\* for Windows, http://www.mingw.org/). MinGW sử dụng GCC (the GNU g++ compiler collection) sử dụng cho cả hệ thống Windows và Linux. Hiện nay DevCpp  là công cụ phát triển các ứng dụng C/C++ được sử dụng rộng  rãi để dạy về lập trình cũng như để phát  triển các ứng dụng mã nguồn mở.

  Mặc  dù  không  có  nhiều  tính  năng  cao  cấp  như  các  công  cụ  IDE thương mại  khác  (Visual Studio của Microsoft) nhưng DevCpp vẫn là một công cụ rất thích hợp trong môi trường giáo dục ở bậc đại học khi dạy và học các ngôn ngữ C/C++. Lý do là Dev-C++ hỗ trợ C/C++ chuẩn, mã nguồn viết trên Dev-C++ có thể biên dịch trên các IDE khác, nhưng điều ngược lại có thể không đúng.

Để thực hiện được các thuật toán trên phần mềm DevC++ thì chúng ta cần phải tải và sử dụng các thư viện mà DevC++ hỗ trọ cho việc lập trình đơn giản và hiệu quả hơn.

**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.

Một trong những điểm mạnh của thư viện này là việc khởi tạo cũng như sử dụng rất đơn giản, vì vậy mặc dù ra đời rất lâu nhưng hiện tại vẫn có rất nhiều trường đại học sử dụng cho mục đích giảng dạy.

# 2. PHƯƠNG PHÁP

## 2.1. Cài đặt DevC/C++ và thư viện graphics.h

DevC/C++ phiên bản 4.9.9.2 được tải về và cài đặt theo hướng dẫn tại địa chỉ: <https://sourceforge.net/projects/orwelldevcpp/files/latest/download>

Để lập trình đồ họa trong DevC cần sử dụng thư viện graphics.h theo địa chỉ sau: *https://github.com/thinhdoanvu/ComputerGraphics/tree/master/Coding/library*

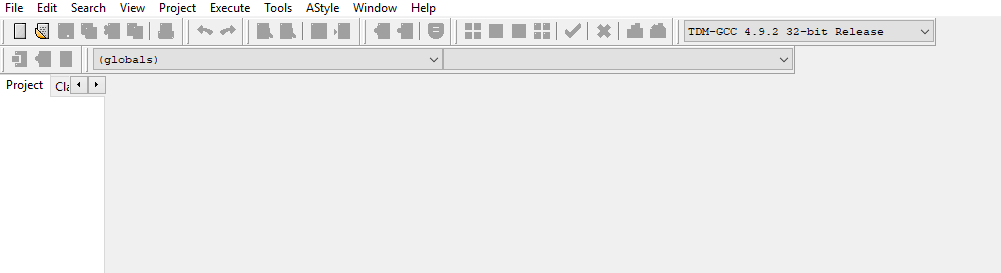
Và tiến hành cài đặt thư viện graphics.h theo các bước sau:

**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*

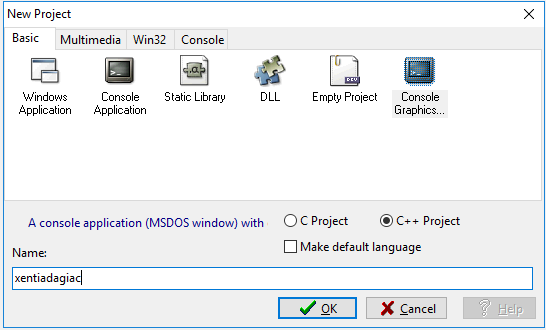
**Bước 3:** Copy tập tin *libbgi.a* đến đường dẫn *C:\Program Files\Dev-Cpp\MinGW64\x86\_64-w64-mingw32\lib*

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**



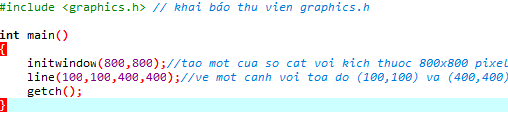
Hình 2.1. Thiết lập phiên bản TDM-GCC phù hợp với DevC/C++ phiên bản 32 bit

**Bước 5:** 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 2.xxxx)



HÌnh 2.2. Thiết lập project khởi động đồ họa sử dụng thư viện graphics.h

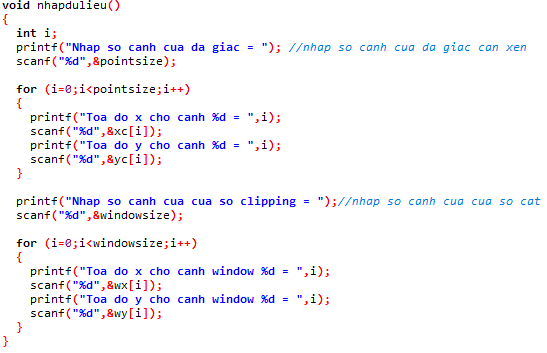
Dưới đây là một đoạn code minh họa trong đồ họa



Hình 5. Code minh họa trong đồ họa

## 2.2. Nhập dữ liệu

Để thực hiện các thao tác xén tỉa đa giác thì trước hết cần nhập dữ liệu các tọa độ điểm của cạnh cần vẽ và cửa sổ cắt.



Hình 2.3. Nhập tọa độ đa giác và cửa sổ cắt

## 2.3. Thuật toán xén tỉa đa giác

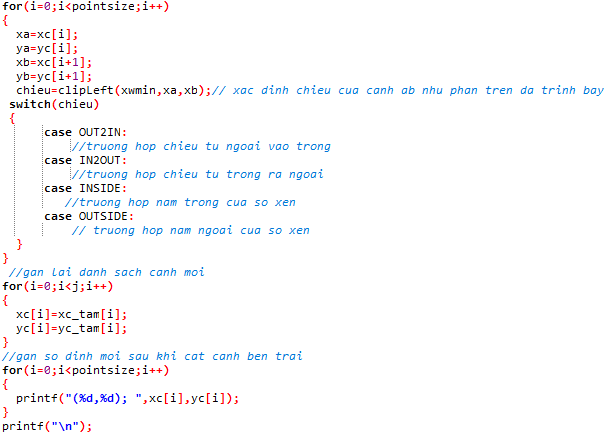
Vì xén tỉa đa giác là phần mở rộng từ thuật toán xén tỉa đoạn thẳng nên thuật toán xén tỉa đa giác thực hiện tương tự. Nếu coi xén tỉa đa giác là một bài tập lớn thì việc chia nhỏ bài toán là xén tỉa các cạnh của đa giác. Vì vậy khi xén tỉa các cạnh thì cũng sẽ rơi vào trường hợp giống như xén tỉa đoạn thẳng.

Khi cài đặt thuật toán Sutherland – Hodgman ta sẽ thực hiện lần lượt theo các bước sau:

### 2.3.1. xác định chiều của các cạnh

|  |  |
| --- | --- |
| **Xác định chiều ở biên trái cửa sổ cắt** |  |
| **Xác định chiều ở biên phải cửa sổ cắt** |  |
| **Xác định chiều ở biên trên cửa sổ cắt** |  |
| **Xác định chiều ở biên dưới cửa sổ cắt** |  |

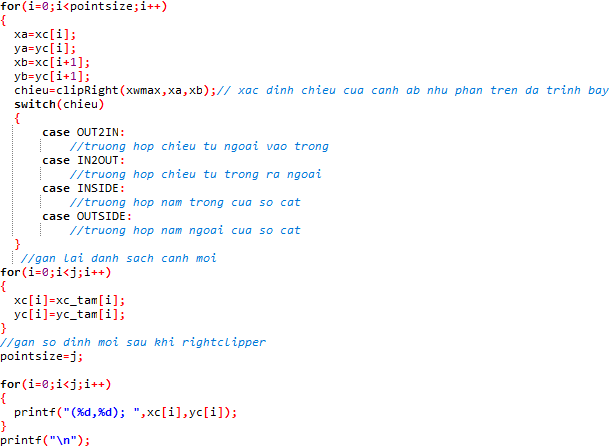
### 2.3.1. Đoạn thẳng nằm trên biên trái cửa sổ cắt



Hình 2.4. Code cho trường hợp xén cạnh bên trái

|  |  |
| --- | --- |
| **Trường hợp 1 :** Đoạn thẳng có chiều từ ngoài vào trong |  |
| **Trường hợp 2:** Đoạn thẳng có chiều từ trong ra ngoài |  |
| **Trường hợp 3:** Đoạn thẳng nằm trong cửa sổ cắt |  |
| **Trường hợp 4:** Đoạn thẳng nằm ngoài cửa sổ cắt |  |

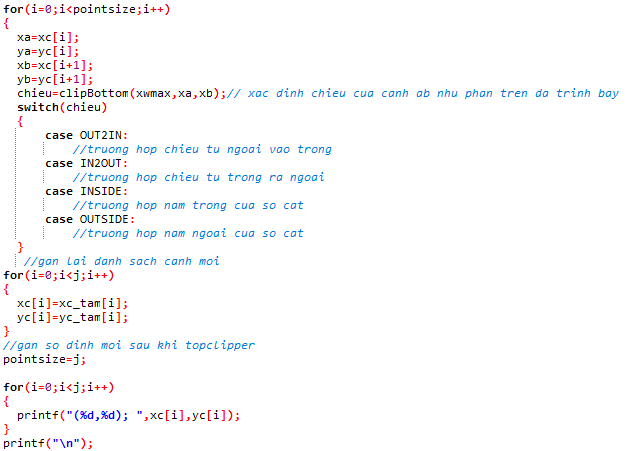
### 2.3.2. Đoạn thẳng nằm trên biên phải cửa sổ cắt



Hình 2.5. Đoạn thẳng nằm trên biên phải cửa sổ cắt

|  |  |
| --- | --- |
| **Trường hợp 1 :** Đoạn thẳng có chiều từ ngoài vào trong |  |
| **Trường hợp 2:** Đoạn thẳng có chiều từ trong ra ngoài |  |
| **Trường hợp 3:** Đoạn thẳng nằm trong cửa sổ cắt |  |
| **Trường hợp 4:** Đoạn thẳng nằm ngoài cửa sổ cắt |  |

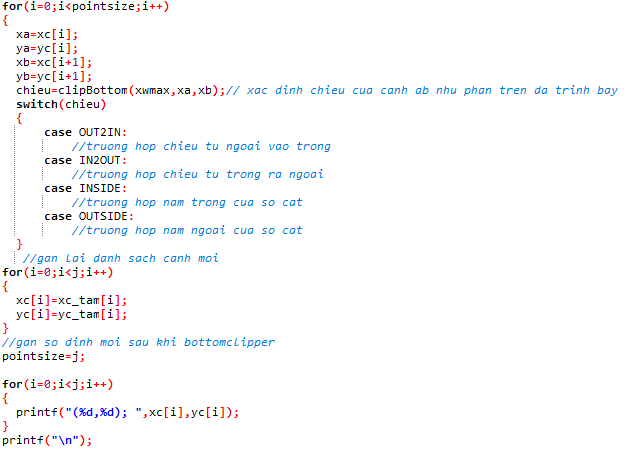
### 2.3.3. Đoạn thẳng nằm trên biên trên cửa sổ cắt



Hình 2.6. Đoạn thẳng nằm trên biên trên cửa sổ cắt

|  |  |  |
| --- | --- | --- |
| **Trường hợp 1 :** Đoạn thẳng ó có chiều từ ngoài vào trong |  | |
| **Trường hợp 2:** Đoạn thẳng có chiều từ trong ra ngoài |  |
| **Trường hợp 3:** Đoạn thẳng nằm trong cửa sổ cắt |  |
| **Trường hợp 4:** Đoạn thẳng nằm ngoài cửa sổ cắt |  |

### 2.3.4. Đoạn thẳng nằm trên biên dưới cửa sổ cắt

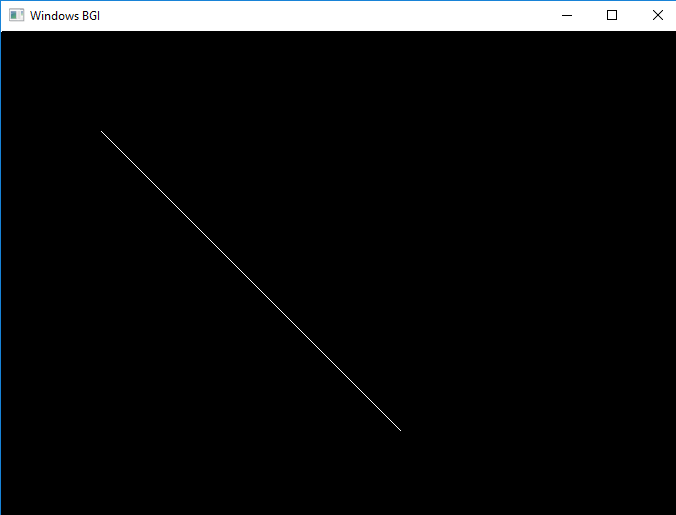


Hình 2.7 . đoạn thẳng nằm trên biên dưới cửa sổ cắt

|  |  |
| --- | --- |
| **Trường hợp 1 :** Đoạn thẳng có chiều từ ngoài vào trong |  |
| **Trường hợp 2:** Đoạn thẳng có chiều từ trong ra ngoài |  |
| **Trường hợp 3:** Đoạn thẳng nằm trong cửa sổ cắt |  |
| **Trường hợp 4:** Đoạn thẳng nằm ngoài cửa sổ cắt |  |

# 3. KẾT QUẢ

## 3.1 DevC/C++ và thực viện graphics.h

Sau khi cài đặt phần mềm DevC/C++ và thư viện graphics.h đối với đoạn code mẫu ở phần trên, sau khi chạy chương trình thì tạo ra một cửa sổ có kích thước 800x800 và một đoạn thẳng như hình dưới.

Hình 3.1. ví dụ về thư viện graphics.h

## 3.3 Thuật toán xén tỉa đa giác

### 3.3.1. Đa giác giao biên cửa sổ cắt

Cho đa giác ABCD với tọa độ lần lượt là (25,200), (115,320), (200,200), (125,90) và cửa sổ cắt (x) với tọa độ (50,120), (180,120), (180,280), (50,280) đã cho như trên thực hiện tuần tự như sau:

Đầu tiên vẽ đa giác cần xén tỉa và cửa sổ cắt

|  |  |
| --- | --- |
|  |  |

Hinh 3.2. dữ liệu được nhập từ bàn phím cho 2 tham số cửa sổ cắt và đa giác cần xén tỉa

Trong đó: Đa giác cần xén tỉa (màu đỏ) và cửa sổ cắt(màu xanh)

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | c | d |

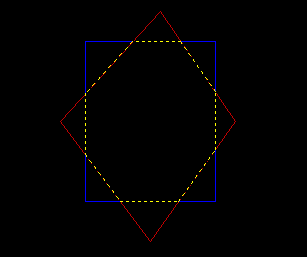
Hình 3.3: các trường hợp xén của đa giác

Trong đó: cửa sổ cắt (màu xanh dương), cạnh bị xén (màu đỏ), cạnh được giữ lại (màu vàng), cạnh được xén giữa đa giác và cửa sổ cắt (màu xanh lá)

Hình a: xén bên phải

Hình b: xén bên trái

Hình c: xén bên dưới

Hình d: xén bên trên.

Hình 3.4: kết quả sau khi xén

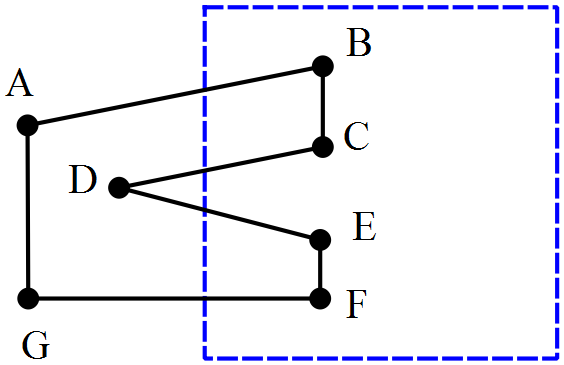
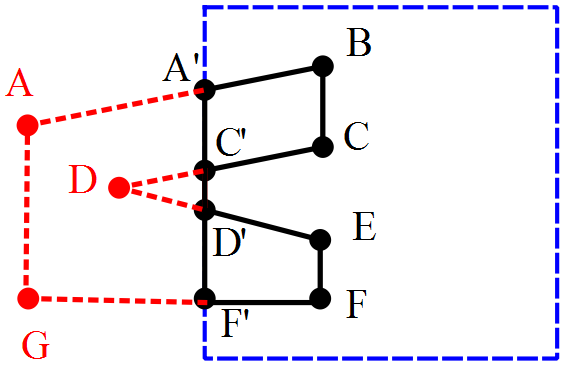
Trong đó: đa giác sau khi xén (đường màu vàng)

# 4. KẾT 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.

Đánh giá về thuật toán **Sutherland-Hodgman** thì nó có ưu điểm là cho kết quả tốt đối với đa giác lồi, còn hạn chế đó là đòi hỏi bộ nhớ đáng kể làm làm phí bộ nhớ để lưu trữ đa giác trung gian, không thể mở rộng thành các phép toán boolean trên các đa giác. Nhưng hạn chế lớn nhất của thuật toán đó là chưa cho được kết quả tối ưu đối với các đa giác lõm, nghĩa là khi cắt thuật toán có thể có các cạnh trùng nhau (chồng chéo nhau)

Trong ví dụ dưới đây ta thấy sau khi thực hiện việc cắt tỉa đa giác như hình dưới thì sẽ tạo ra cạnh dư thừa (cạnh C’D’ không nằm trong đa giác ban đầu)



*Hình 4.1. Hạn chế của thuật toán Sutherland-Hodgman*

Và thuật toán[**Weiler–Atherton**](https://en.wikipedia.org/wiki/Weiler%E2%80%93Atherton)đã ra đời và khắc phục được điểm hạn chế này bằng cách trả về một tập hợp các đa giác được chia, nhưng phức tạp hơn và đắt hơn về mặt tính toán.

# TÀI LIỆU THAM KHẢO

<https://www.researchgate.net/publication/220913402_Polygon_Clipping_and_Polygon_Reconstruction?fbclid=IwAR1O64ecE88d2cRWLzjInaFCywxwwrWas64KzbGlcOJ2HIp4LKCcDU_EKz8>

<https://www.javatpoint.com/sutherland-hodgeman-polygon-clipping>

<https://github.com/thinhdoanvu/ComputerGraphics/blob/master/Coding/Clipping/Hogman_Polygon_Clipping>

<https://github.com/thinhdoanvu/ComputerGraphics/blob/master/Chapter4_Edit21April2019.pptx>