**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



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

GVHD : Đoàn Vũ Thịnh

SVTH : Bùi Thị Thanh Nương - 59131817

Lê Thị Mộng Ngân - 59131524

Lớp : 59CNTT1

Khánh Hòa, tháng 01 năm 2020

**MỤC LỤC**

[TÓM TẮT 4](#_Toc30056217)

[1. GIỚI THIỆU 5](#_Toc30056218)

[1.1. Thuật toán xén tỉa đoạn thẳng 5](#_Toc30056219)

[1.1.1. Thuật toán Cohen- Sutherland 5](#_Toc30056220)

[1.1.2. Thuật toán Cyrus-beck 10](#_Toc30056221)

[1.1.3. Thuật toán liang-barsky 10](#_Toc30056222)

[1.2. Thuật toán xén tỉa đa giác 10](#_Toc30056223)

[1.3. Phần mềm DevC/C++ và thư viện graphics.h, winbmin 13](#_Toc30056224)

[2. PHƯƠNG PHÁP 14](#_Toc30056225)

[2.1. Cài đặt DevC/C++ và thư viện graphics.h, winbmin 14](#_Toc30056226)

[2.2. Nhập dữ liệu 15](#_Toc30056227)

[2.2.1. Đoạn thẳng nằm trên biên trái cửa sổ xén 16](#_Toc30056228)

[2.2.2. Đoạn thẳng nằm trên biên phải cửa sổ xén 16](#_Toc30056229)

[2.2.3. Đoạn thẳng nằm trên biên trên cửa sổ xén 17](#_Toc30056230)

[2.2.4. Đoạn thẳng nằm trên biên dưới cửa sổ xén 17](#_Toc30056231)

[2.2.5. Đoạn thẳng nằm ngoài cửa sổ xén 18](#_Toc30056232)

[2.2.6. Đoạn thẳng nằm trong cửa sổ xén 18](#_Toc30056233)

[2.3. Thuật toán xén tỉa đa giác 19](#_Toc30056234)

[2.3.1. Đoạn thẳng nằm trên biên trái cửa sổ cắt 19](#_Toc30056235)

[2.3.2. Đoạn thẳng nằm trên biên phải cửa sổ cắt 20](#_Toc30056236)

[2.3.3. Đoạn thẳng nằm trên biên trên cửa sổ cắt 21](#_Toc30056237)

[2.3.4. Đoạn thẳng nằm trên biên dưới cửa sổ cắt 22](#_Toc30056238)

[3. KẾT QUẢ 23](#_Toc30056239)

[3.1 DevC/C++ và thực viện graphics.h 23](#_Toc30056240)

[3.3 Thuật toán xén tỉa đa giác 24](#_Toc30056241)

[3.3.1. Đa giác giao biên cửa sổ cắt 24](#_Toc30056242)

[3.3.2 Đa giác nằm ngoài cửa sổ cắt 26](#_Toc30056243)

[3.3.3 Đa giác nhằm trong cửa sổ cắt 27](#_Toc30056244)

[3.3.4. Đa giác lõm 27](#_Toc30056245)

[4. THẢO LUẬN 28](#_Toc30056246)

# TÓM TẮT

Trong phần Giới thiệu đã nêu ra khái niệm về xén hình và cửa sổ xén (clip window); đề cập đến các phần mềm đã áp dụng kỹ thuật xén tỉa hình ảnh. Ngoài ra, còn giới thiệu một cách khá tổng quát về các thuật toán xén tỉa đoạn thẳng cũng như các thuật toán xén tỉa đa giác, và phần mềm dùng để cài đặt thuật toán xén tỉa đa giác cùng với các thư viện hỗ trợ việc cài đặt.

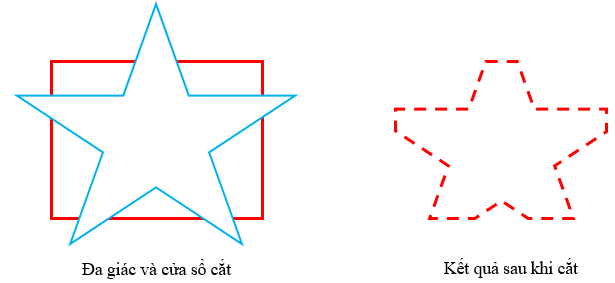
Trong phần Phương pháp đã đề cập đến các bước cài đặt DevC/C++ và thư viện graphics.h; nhập dữ liệu các trường hợp xén tỉa đoạn thẳng và nêu rõ cách xén tỉa đa giác bằng thuật toán Sutherland-Hodgman.

Tiếp theo là phần Kết quả, phần này chỉ ra các kết quả có được khi thực hiện việc xén tỉa đa giác bằng thuật toán Sutherland-Hodgman dựa vào phần phương pháp trên. Mỗi phương pháp xén tỉa sẽ có một kết quả khác nhau.

Cuối cùng là phần Thảo luận về thuật toán Sutherland-Hodgman. Thuật toán này cũng giống như các thuật toán khác, bên cạnh các ưu điểm và hiệu quả thì thuật toán Sutherland-Hodgman còn có những hạn chế. Từ đó, thuật toán [**Weiler–Atherton**](https://en.wikipedia.org/wiki/Weiler%E2%80%93Atherton)đã ra đời và khắc phục những hạn chế này.

# GIỚI THIỆU

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, Photoshop, Adobe Illustrator, hay phần mềm Power Point 2013 (Hình 1.1).



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 cụ thể của xén tỉa đoạn thẳng.

## 1.1. Thuật toán xén tỉa đoạn thẳng

### *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. Hạn chế của thuật toán cắt dòng Cohen Sutherland là thuật toán chỉ áp dụng cho các cửa sổ hình chữ nhật và không áp dụng cho các cửa sổ hình lồi khác. (<https://en.wikipedia.org/wiki/Cohen%E2%80%93Sutherland_algorithm>).

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 |

Xét bài toán xén đoạn thẳng được cho bởi 2 điểm P1(X1,Y1) và P2(X2,Y2) vào cửa sổ hình chữ nhật như trên. Yêu cầu của bài toán xén tỉa đoạn thẳng là loại bỏ phần đoạn thẳng nằm ngoài cửa sổ xén.

**Các trường hợp xén:**

|  |  |
| --- | --- |
| ***Trường hợp 1:*** Đoạn thẳng có 2 điểm hoàn toàn nằm trong cửa sổ nên **không cần xén** | *Hình 1.3. Đoạn thẳng nằm trong cửa sổ xén* |
| ***Trường hợp 2:*** Đoạn thẳng có 2 điểm cùng nằm ngoài về một phía ngoài của cửa sổ và sẽ bị xén mất. | *Hình 1.4 Đoạn nằm ngoài cửa sổ xén* |
| ***Trường hợp 3:*** Nếu đoạn thẳng cắt biên cửa sổ bên trái thì tọa độ tại điểm đó được xác định như sau: | *Hình 1.5: Đoạn thẳng cắt biên cửa sổ bên trái* |
| ***Trường hợp 4:*** Nếu đoạn thẳng cắt biên cửa sổ bên phải thì tọa độ tại điểm đó được xác định như sau: | *Hình 1.6: Đoạn thẳng cắt biên cửa sổ bên phải* |
| ***Trường hợp 5:*** Nếu đoạn thẳng cắt biên cửa sổ phía trên thì tọa độ tại điểm đó được xác định như sau: | *Hình 1.7: Đoạn thẳng cắt biên cửa sổ phía trên* |
| ***Trường hợp 6:*** Nếu đoạn thẳng cắt biên cửa sổ bên dưới thì tọa độ tại điểm đó được xác định như sau: | *Hình 1.8: Đoạn thẳng cắt biên cửa sổ bên dưới* |

**Xác định chiều đoạn thẳng:**

|  |  |
| --- | --- |
| **Trường hợp 1: đoạn thẳng cắt biên bên trái**  Nếu xP1 <= xwmin và xP2 >= xwmin  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu OUT2IN  Nếu xP1 >= xwmin và xP2 <= xwmin  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu IN2OUT | Hình 1.9.Chiều đoạn thẳng cắt biên bên trái từ ngoài vào trong    Hình 1.10.Chiều đoạn thẳng cắt biên bên trái từ trong ra ngoài |
| **Trường hợp 2: đoạn thẳng cắt biên bên phải**  Nếu xP1 > xwmax và xP2 <= xwmax  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu OUT2IN  Nếu xP1 <= xwmax và xP2 > xwmax thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu IN2OUT | Hình 1.11.Chiều đoạn thẳng cắt biên bên phải từ ngoài vào trong    Hình 1.12.Chiều đoạn thẳng cắt biên bên phải từ trong ra ngoài |
| **Trường hợp 3: đoạn thẳng cắt biên bên trên**  Nếu yP1 > ywmax và yP2 <= ywmax  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu OUT2IN  Nếu yP1 <= ywmax và yP2 > ywmax thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu IN2OUT | Hình 1.13. Chiều đoạn thẳng cắt biên bên trên từ ngoài vào trong    Hình 1.14. Chiều đoạn thẳng cắt biên bên trên từ trong ra ngoài |
| **Trường hợp 4: đoạn thẳng cắt biên bên dưới**  Nếu yP1 < ywmin và yP2 >= ywmin  thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu OUT2IN  Nếu yP1 >= ywmin và yP2 < ywmin thì đoạn thẳng có chiều từ ngoài vào trong. Ký hiệu IN2OUT | Hình 1.15. Chiều đoạn thẳng cắt biên bên dưới từ ngoài vào trong    Hình 1.16. Chiều đoạn thẳng cắt biên bên dưới từ trong ra ngoài |

### 1.1.2. Thuật toán Cyrus-beck

Thuật toán được phát triển vào năm 1978 bởi [Mike Cyrus](https://www.sciencedirect.com/science/article/abs/pii/0097849378900213" \l "!) và [Jay Beck](https://www.sciencedirect.com/science/article/abs/pii/0097849378900213#!). Đây là thuật toán cắt đoạn thẳng tổng quát . Nó được thiết kế để hiệu quả hơn [***thuật toán Cohen- Sutherland***](https://en.wikipedia.org/wiki/Cohen%E2%80%93Sutherland_algorithm) có thể được 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>).

### 1.1.3. Thuật toán liang-barsky

Được phát triển năm 1984 bởi Liang, You-Dong và Barsky, Brian A. Thuật toán Liangky Barsky 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 đoạn 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 cửa 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://en.wikipedia.org/wiki/Liang%E2%80%93Barsky_algorithm>).

Một số thuật toán khác: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

## 1.2. 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 có đỉnh lõm ngoài cửa sổ xén, 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.

|  |  |
| --- | --- |
| 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) | Hình 1.17. Xén tỉa đa giác |

**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

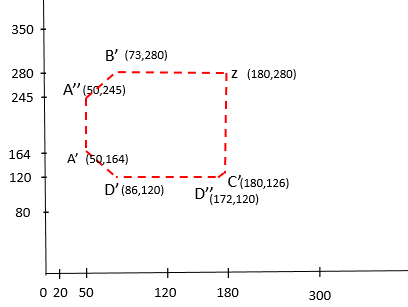
Vậy D’(, 120)

**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)



Hình 1.18: kết quả sau khi xén

Kevin và Atherton (1977) đề xuất ý tưởng cho thuật toán để giải quyết hạn chế cho trường hợp với đa giác 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 (<https://en.wikipedia.org/wiki/Weiler%E2%80%93Atherton_clipping_algorithm>).

[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>)

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

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 sử dụng phần mềm Devc++ phiên bản 5.11 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.

**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.

# 2. PHƯƠNG PHÁP

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

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*

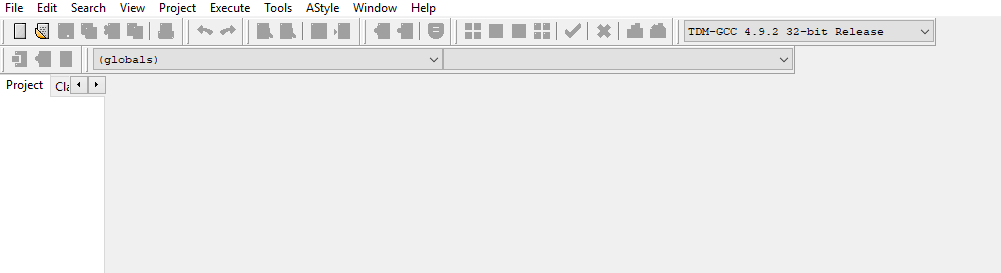
Và tiến hành cài đặt thư viện graphics.h theo các bước sau:

**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*

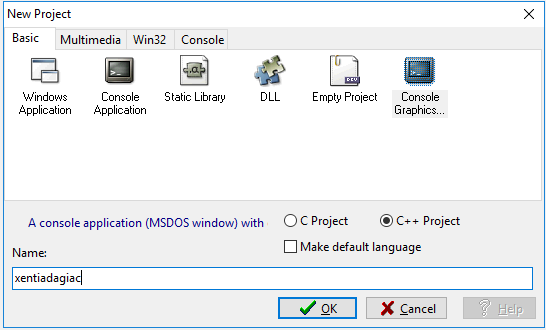
**Bước 3:** Copy tập tin *libbgi.a* đến đường dẫn *C:\Program Files\Dev-Cpp\MinGW64\x86\_64-w64-mingw32\lib*

**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**



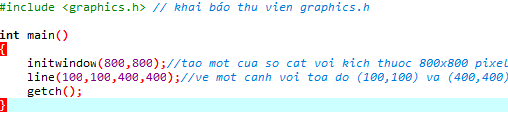
Hình 2.1. Thiết lập phiên bản TDM-GCC phù hợp với DevC/C++ phiên bản 32 bit

**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 2.2)



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

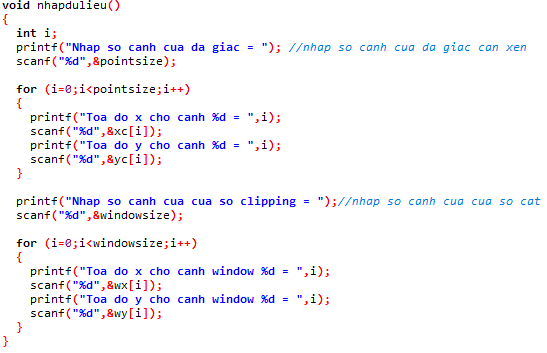
Dưới đây là một đoạn code minh họa trong đồ họa



Hình 2.3. 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 2.4. Nhập tọa độ đa giác và cửa sổ cắt

Sau khi nhập dữ liệu tọa độ các điểm nó có thể rơi vào từng trường hợp như sau:

### 2.2.1. Đoạn thẳng nằm trên biên trái cửa sổ xén

Cho cửa sổ cắt (x) có tọa độ các điểm lần lượt là (50,150), (200,150), (200,50), (50,50)

|  |  |
| --- | --- |
| **Trường hợp 1**: đoạn thẳng P1P2 có tọa độ P1(0,0) P2(150,100) nằm trên biên trái cửa sổ xén có chiều từ ngoài vào trong. | Hình 2.5. Đoạn thẳng nằm trên biên trái có chiều từ ngoài vào trong |
| **Trường hợp 2**: đoạn thẳng đoạn thẳng P1P2 có tọa độ P1(150,100) P2(0,0) nằm trên biên trái cửa sổ xén có chiều từ trong ra ngoài. | Hình 2.6. Đoạn thẳng nằm trên biên trái có chiều từ ngoài vào trong |

### 2.2.2. Đoạn thẳng nằm trên biên phải cửa sổ xén

Cho cửa sổ cắt (x) có tọa độ các điểm lần lượt là (50,150), (200,150), (200,50), (50,50)

|  |  |
| --- | --- |
| **Trường hợp 1**: đoạn thẳng P1P2 có tọa độ (100,50), (250,125) nằm trên biên phải cửa sổ xén có chiều từ trong ra ngoài | Hình 2.7. Đoạn thẳng nằm trên biên phải có chiều từ trong ra ngoài |
| **Trường hợp 2**: đoạn thẳng P1P2 có tọa độ (250,125), (100,50) nằm trên biên phải cửa sổ xén có chiều từ trong ra ngoài | Hình 2.8. Đoạn thẳng nằm trên biên phải có chiều từ ngoài vào trong |

### 2.2.3. Đoạn thẳng nằm trên biên trên cửa sổ xén

Cho cửa sổ cắt (x) có tọa độ các điểm lần lượt là (50,150), (200,150), (200,50), (50,50)

|  |  |
| --- | --- |
| **Trường hợp 1**: đoạn thẳng P1P2 có tọa độ (50,50), (150,200) nằm trên biên trên cửa sổ xén có chiều từ trong ra ngoài | Hình 2.9. Đoạn thẳng nằm trên biên trên có chiều từ trong ra ngoài |
| **Trường hợp 2**: đoạn thẳng P1P2 có tọa độ (150,200), (50,50) nằm trên biên trên cửa sổ xén có chiều từ ngoài vào trong | Hình 2.10. Đoạn thẳng nằm trên biên trên có chiều từ ngoài vào trong |

### 2.2.4. Đoạn thẳng nằm trên biên dưới cửa sổ xén

Cho cửa sổ cắt (x) có tọa độ các điểm lần lượt là (50,150), (200,150), (200,50), (50,50)

|  |  |
| --- | --- |
| **Trường hợp 1**: đoạn thẳng P1P2 có tọa độ (50,0), (75,100) nằm trên biên dưới cửa sổ xén có chiều từ ngoài vào trong | Hình 2.11. Đoạn thẳng nằm trên biên dưới có chiều từ ngoài vào trong |
| **Trường hợp 2**: đoạn thẳng P1P2 có tọa độ (750,100), (50,0) nằm trên biên dưới cửa sổ xén có chiều từ trong ra ngoài | Hình 2.12. Đoạn thẳng nằm trên biên dưới có chiều từ trong ra ngoài |

### 2.2.5. Đoạn thẳng nằm ngoài cửa sổ xén

|  |  |
| --- | --- |
| Cho cửa sổ cắt (x) có tọa độ các điểm lần lượt là (50,150), (200,150), (200,50), (50,50) và đoạn thẳng P1P2 có tọa độ (225,25), (275,125) nằm ngoài cửa sổ cắt | Hình 2.13. Đoạn thẳng nằm ngoài cửa sổ cắt |

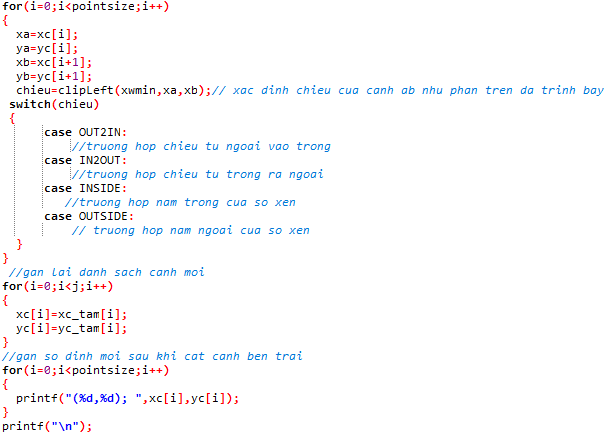
### 2.2.6. Đoạn thẳng nằm trong cửa sổ xén

|  |  |
| --- | --- |
| Cho cửa sổ cắt (x) có tọa độ các điểm lần lượt là (50,150), (200,150), (200,50), (50,50) và đoạn thẳng P1P2 có tọa độ (50,50), (150,125) nằm trong cửa sổ cắt | Hình 2.14. Đoạn thẳng nằm trong 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:

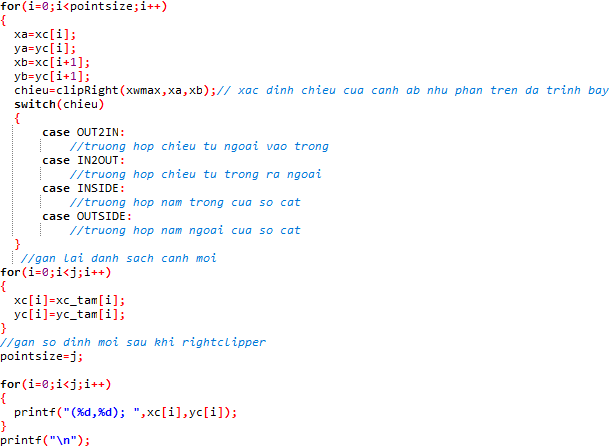
### 2.3.1. Đoạn thẳng nằm trên biên trái cửa sổ cắt



Hình 2.15. Code cho trường hợp xén cạnh bên trái

|  |  |
| --- | --- |
| **TH1:** Đoạn thẳng có chiều từ ngoài vào trong | Hình 2.16. trường hợp có chiều từ ngoài vào trong |
| **TH2:** Đoạn thẳng có chiều từ trong ra ngoài | Hình 2.17.Trường hợp có chiều từ trong ra ngoài |
| **TH3:** Đoạn thẳng nằm trong cửa sổ cắt | Hình 2.18.Trường hợp nằm trong cửa sổ cắt |
| **TH4:** Đoạn thẳng nằm ngoài cửa sổ cắt | Hình 2.19.Trường hợp nằm ngoài cửa sổ cắt |

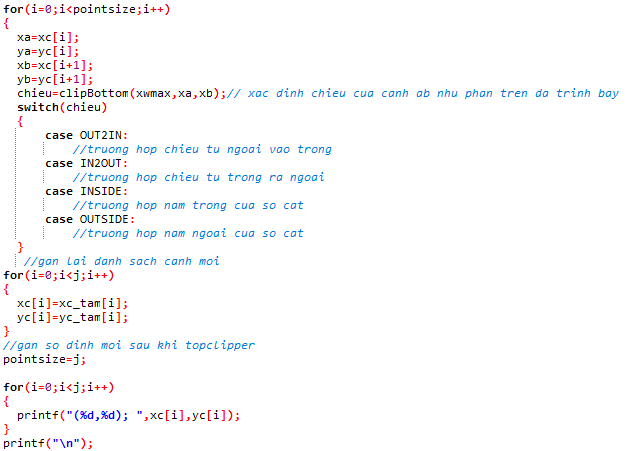
### 2.3.2. Đoạn thẳng nằm trên biên phải cửa sổ cắt



Hình 2.20. Đoạn thẳng nằm trên biên phải cửa sổ cắt

|  |  |
| --- | --- |
| **TH1 :** Đoạn thẳng có chiều từ ngoài vào trong | Hình 2.21. trường hợp có chiều từ ngoài vào trong |
| **TH2:** Đoạn thẳng có chiều từ trong ra ngoài | Hình 2.22.Trường hợp có chiều từ trong ra ngoài |
| **TH3:** Đoạn thẳng nằm trong cửa sổ cắt | Hình 2.23.Trường hợp nằm trong cửa sổ cắt |
| **TH4:** Đoạn thẳng nằm ngoài cửa sổ cắt | Hình 2.24.Trường hợp nằm ngoài cửa sổ cắt |

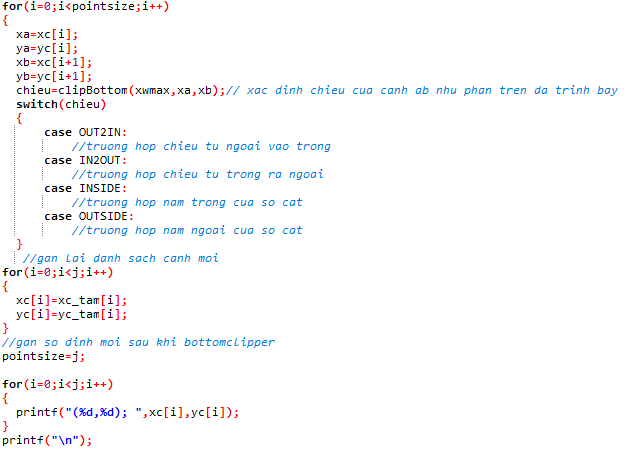
### 2.3.3. Đoạn thẳng nằm trên biên trên cửa sổ cắt



Hình 2.25. Đoạn thẳng nằm trên biên trên cửa sổ cắt

|  |  |  |
| --- | --- | --- |
| **TH1 :** Đoạn thẳng có chiều từừ ngoài vào trong | Hình 2.26. trường hợp có chiều từ ngoài vào trong | |
| **TH2:** Đoạn thẳng có chiều từ trong ra ngoài | Hình 2.27. Trường hợp có chiều từ trong ra ngoài |
| **Trường hợp 3:** Đoạn thẳng nằm trong cửa sổ cắt | Hình 2.28.Trường hợp nằm trong cửa sổ cắt |
| **Trường hợp 4:** Đoạn thẳng nằm ngoài cửa sổ cắt | Hình 2.29. Trường hợp nằm ngoài cửa sổ cắt |

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



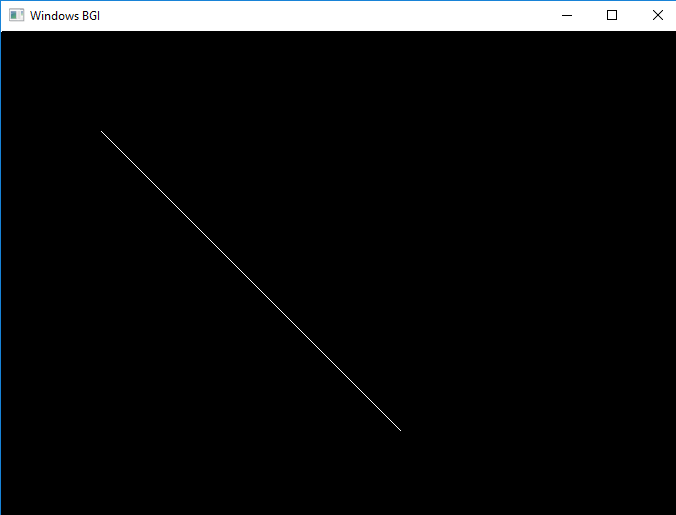
Hình 2.30. đoạn thẳng nằm trên biên dưới cửa sổ cắt

|  |  |
| --- | --- |
| **TH1 :** Đoạn thẳng có chiều từ ngoài vào trong | Hình 2.31. trường hợp có chiều từ ngoài vào trong |
| **TH2:** Đoạn thẳng có chiều từ trong ra ngoài | Hình 2.32 .Trường hợp có chiều từ trong ra ngoài |
| **TH3:** Đoạn thẳng nằm trong cửa sổ cắt | Hình 2.33.Trường hợp nằm trong cửa sổ cắt |
| **TH4:** Đoạn thẳng nằm ngoài cửa sổ cắt | Hình 2.34.Trường hợp 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



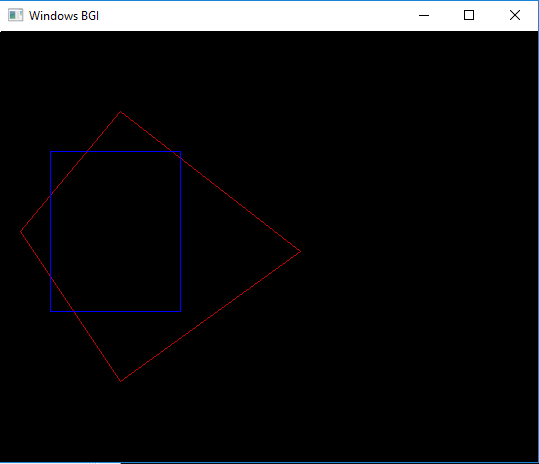
Hình 3.1. ví dụ về thư viện graphics.h

## 3.3 Thuật toán xén tỉa đa giác

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

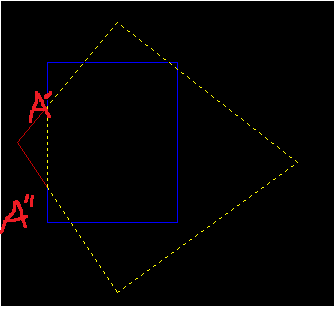
Cho đa giác ABCD với tọa độ lần lượt là (20,200), (120,350), (300,220), (120,80) và cửa sổ cắt (x) với tọa độ (50,120), (180,120), (180,280), (50,280) đã cho như trên thực hiện tuần tự như sau:

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



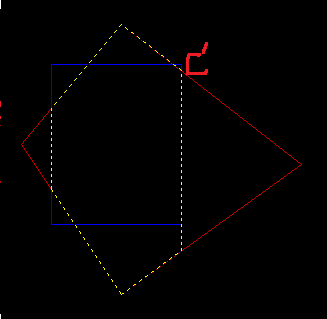
Hình 3.2. Đa xén cần xén tỉa (màu đỏ) và cửa sổ cắt(màu xanh)

**Xén cạnh bên trái**: Thực hiện xén tỉa ở các cạnh bên trái của cửa sổ cắt: cạnh bên dưới của đa giác có chiều từ ngoài vào trong và giao với biên của cửa sổ cắt ta có điểm mới A’’ có tọa độ (50, 245), cạnh bên trên của đa giác có chiều từ trong ra ngoài (In-Out) và giao với biên của cửa sổ cắt ta có điểm mới A’ có tọa độ (50,164)



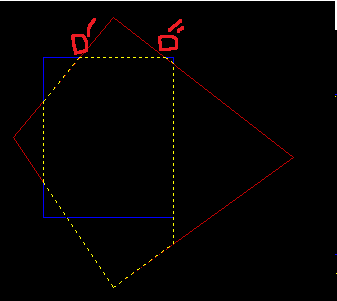
Hình 3.3. Xén tỉa cạnh bên trái của đa giác

**Xén cạnh bên phải**: Thực hiện xén tỉa ở các cạnh bên phải của cửa sổ cắt: cạnh bên dưới của đa giác ngoài cửa sổ cắt nên nó không tạo điểm mới nào, cạnh bên trên của đa giác có chiều từ ngoài vào trong (Out -In) và giao với biên của cửa sổ cắt ta có điểm mới C’ có tọa độ (180,126)



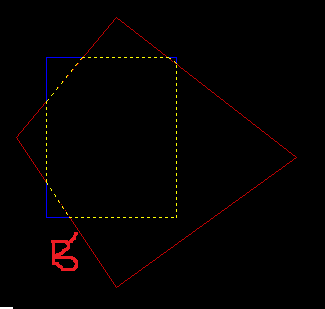
Hình 3.4.. Xén tỉa cạnh bên phải của đa giác

**Xén cạnh bên dưới**: Thực hiện xén tỉa ở các cạnh bên dưới của cửa sổ cắt: cạnh bên phải của đa giác có chiều từ trong ra ngoài (In-Out) và giao với biên của cửa sổ cắt ta có điểm mới D’’ có tọa đô(172,120), cạnh bên trái của đa giác có chiều từ ngoài vào trong (Out -In) và giao với biên của cửa sổ cắt ta có điểm mới D’ có tọa độ (86,120)



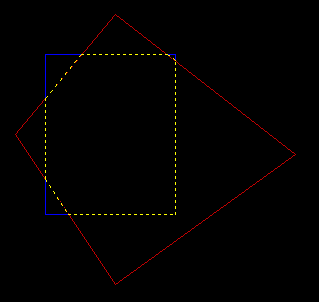
Hình 3.5. Xén tỉa cạnh bên dưới của đa giác

**Xén cạnh bên trên**: Thực hiện xén tỉa ở các cạnh bên trên của cửa sổ cắt: cạnh bên trái của đa giác có chiều từ trong ra ngoài (In-Out) và giao với biên của cửa sổ cắt ta có điểm mới B’ có tọa độ (73,280), cạnh bên phải của đa giác nằm ngoài đa giác không tạo điểm mới



Hình 3.14.. Xén tỉa cạnh bên dưới của đa giác

Kết quả của thuật toán ta có một đa giác hoàn chỉnh gồm 7 điểm

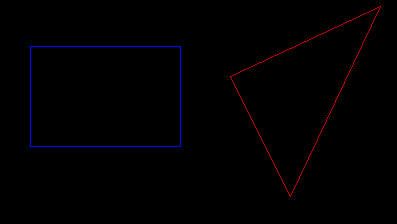


Hình 3.15. Đa giác sau khi xén tỉa

### 3.3.2 Đa giác nằm ngoài cửa sổ cắt

Cho đa giác ABC với tọa độ lần lượt (250,80), (300,200), (310,200) và cửa sổ cắt (x) có tọa độ lần lượt (200,50), (200,150), (50,150), (50,50)

Đầu tiên vẽ đa giác và cửa sổ cắt



Hình 3.16.. Xén tỉa đa giác trong trường hợp nằm ngoài cửa sổ cắt

Trong trường hợp này thì kết quả không thay đổi gì do đa giác nằm ngoài cửa sổ cắt nên đa giác không tạo điểm mới.

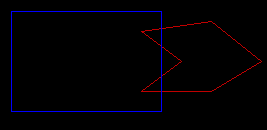
### 3.3.3 Đa giác nhằm trong cửa sổ cắt

Cho đa giác ABC với tọa độ lần lượt (70,150), (120,250), (150,200) và cửa sổ cắt (x) có tọa độ lần lượt (50,120), (180,120), (180,280), (50,280)

|  |  |
| --- | --- |
| Đầu tiên vẽ đa giác và cửa sổ cắt. Trong đó màu xanh là cửa sổ cắt, màu đỏ là đa giác cần xén tỉa. | Hình 3.17. Xén tỉa đa giác trong trường hợp nằm trong cửa sổ cắt |
| Trong trường hợp này đa giác sau khi cắt chính là đa giác lúc ban đầu do đa giác cần xén nằm trong cửa sổ cắt nên nó không tạo điểm mới. | Hình 3.18 Kết quả xén tỉa đa giác trong trường hợp nằm ngoài cửa sổ cắt |

### 3.3.4. Đa giác không lồi có điểm lõm

Cho đa giác ABCDE có tọa độ lần lượt sau (180,130), (250,130), (300,100), (250,60),(180,70), (220,100) và cửa sổ cắt (x) có tọa độ lần lượt là (200,50), (200,150), (50,150), (50,50)



Hình 3.19. Xén tỉa đa giác trong trường hợp đa giác lõm

Trong trường hợp này không thực hiện việc xén tỉa đa giác

Từ ví dụ trên ta thấy thuật toán **Sutherland-hodgman** không áp dụng được với những đa giác lõm .

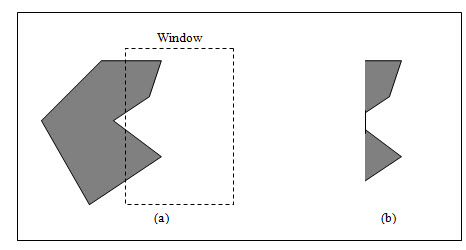
# 4. THẢO LUẬN

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, chồng chéo 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



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

*Nguồn:* [*https://voer.edu.vn/*](https://voer.edu.vn/)

Và thuật toán[**Weiler–Atherton**](https://en.wikipedia.org/wiki/Weiler%E2%80%93Atherton) 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 và thuật toán này chưa được thực hiện trên máy tính. Trong tương lai thuật toán sẽ được cải thiện và thực hiện tối ưu hơn.