TRƯỜNG ĐẠI HỌC NHA TRANG

KHOA CÔNG NGHỆ THÔNG TIN

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

GVHD: Ths. ĐOÀN VŨ THỊNH

SVTH : Tô Hiếu Ngôi
MSSV : 59131611
Lớp : 59.CNTT-3

SVTH : Lê Thế Dũng
MSSV : 59130403
Lớp : 59.CNTT-1

Khánh Hòa, tháng 01 năm 2020

MỤC LỤC

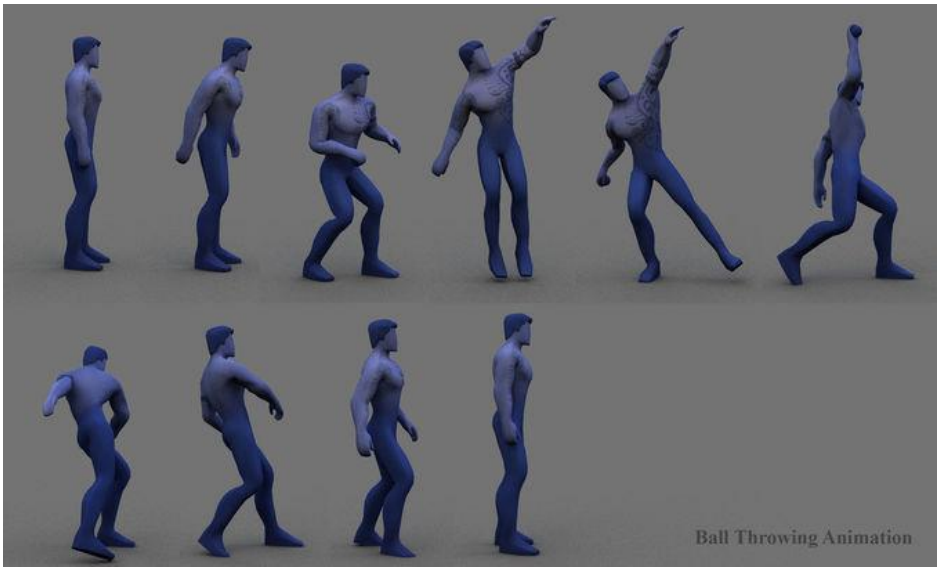
TÓM TẮT

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. Kỹ thuật xén tỉa đa giác là một trong số rất nhiều thuật toán quan trọng của ngành đồ hoạ máy tính và các phần mềm thương mại hay các thiết bị xử lý đồ hoạ đều vận dụng các thuật toán này. Thuật toán xén tỉa đa giác với giải thuật Sutherland-Hogman được cài đặt trong lần thực tập lần này giúp hiểu rõ hơn về một trong những thuật toán cơ bản của ngành đồ hoạ máy tính. Quy trình thực hiện được trải qua các bước từ thiết kế giao diện, cài đặt thuật toán, hiển thị kết quả đầu ra trên màn hình hiển thị đều được thực hiện trên môi trường C++ thông qua ứng dụng DevC/C++ có kết hợp với thư viện graphics.h.

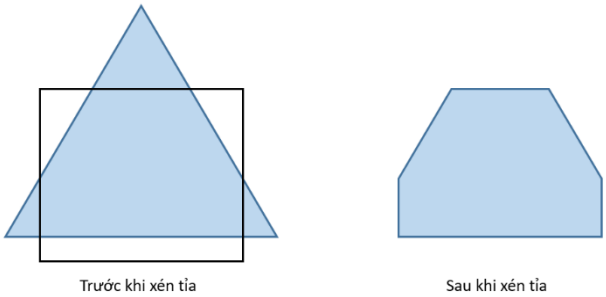
Kết quả của việc cài đặt thuật toán đáp ứng được các yêu cầu đặt ra của đợt thực tập cơ sở lần này. Giao diện đồ hoạ đẹp, thân thiện, thao tác nhập dữ liệu từ chuột là ưu điểm của sản phẩm này.

1. GIỚI THIỆU

Đồ họa máy tính là một lĩnh vực của Công nghệ thông tin, ở đó nghiên cứu, xây dựng và tập hợp các công cụ (mô hình lý thuyết và phần mềm) khác nhau để kiến tạo, xây dựng, lưu trữ và xử lý các mô hình và hình ảnh của các đối tượng, sự vật, hiện tượng trong cuộc sống, sản xuất, nghiên cứu. Đồ họa máy tính góp phần quan trọng làm cho giao tiếp giữa con người và máy tính trở nên thân thiện hơn. Từ đồ họa trên máy tính chúng ta có nhiều lĩnh vực có ứng dụng rất quan trọng của đồ họa máy tính trong thực tế như: tạo mô hình, hoạt cảnh, hỗ trợ thiết kế đồ họa, mô phỏng hình ảnh, chuẩn đoán hình ảnh (trong Y tế), huấn luyện đào tạo ảnh (quân sự, hàng không,…). Hình 1.1. là một ví dụ cụ thể của ứng dụng kỹ thuật đồ hoạ trong ngành điện ảnh. Trong khuôn khổ đợt thực tập cơ sở lần này, nhóm thực hiện đề tài *"Cài đặt thuật toán xén một đa giác Sutherland Hogman"*.



Hình 1.1. Ứng dụng kỹ thuật đồ hoạ trong kỹ thuật hoạt hình
(Nguồn: https://img2.cgtrader.com/items/1003525/3a9f8e515c/large/ball-throwing-animation-3d-model-animated-rigged-max-ma-mb.jpg)



Hình 1.2. Ví dụ xén tỉa đa giác trong phần mềm PowerPoint 2016

Thuật toán xén tỉa trong đồ họa máy tính có tầm quan trọng rất lớn và được sử dụng rộng rãi trong các phần mềm phổ biến hiện này. Từ những phần mềm đơn giản

như Paint, Powerpoint trong bộ Office của Window đến những ứng dụng thiết kế đồ họa chuyên nghiệp như Photoshop, AutoCad. Hình 1.2 là ví dụ về kỹ thuật xén tỉa đa giác trong phần mềm MS Powerpoint 2013.

1.1. Thuật toán xén tỉa

Trước hết, việc xác định các phần của hình ảnh bên trong và loại bỏ phần bên ngoài của một vùng không không gian xác định được gọi là thuật toán xén tỉa, hoặc đơn giản là xén tỉa. Trong đó:

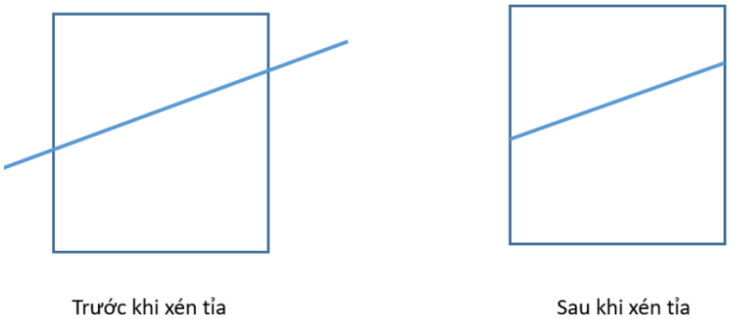
Vùng mà đối tượng được xén tỉa được gọi là cửa sổ xén tỉa. Cửa sổ xén có thể là hình chữ nhật, hình tròn, cửa sổ lồi, cửa sổ lõm, tùy vào các chiến lượt xén tỉa khác nhau dẫn đến các thuật toán xén tỉa giống nhau. Cửa sổ xén tỉa là hình chữ nhật, và đối tượng xén là 1 đa giác. Sử dụng thuật toán xén tỉa để loại bỏ phần hình ảnh bên ngoài cửa sổ xén ta được phần hình ảnh của đối tượng nằm bên trong cửa sổ xén tỉa.

Đối tượng được xén tỉa có thể là đường thẳng, đa giác, hình tròn, ký tự hoặc đường cong không đều. Để xén tỉa được một đa giác trên ta phải xén tỉa từng đoạn thẳng là mỗi cạnh của đa giác. Từ đó dẫn tới sự ra đời của thuật toán xén tỉa đoạn thẳng.

1.2. Thuật toán xén tỉa đoạn thẳng

Thuật toán xén tỉa đoạn thẳng đầu tiên ra đời vào năm 1967 được phát minh bởi Danny Cohen and Ivan Sutherland.

Thuật toán xén tỉa đoạn thẳng là thuật toán xác định các điểm của đoạn thẳng nằm trong hay nằm ngoài của cửa sổ xén.



Hình 1.3. Ví dụ về xén tỉa đoạn thẳng

Ý tưởng: 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 mãn bất kỳ một trong bốn bất đẳng thức sau:

x1,x2>xmax y1,y2> ymax

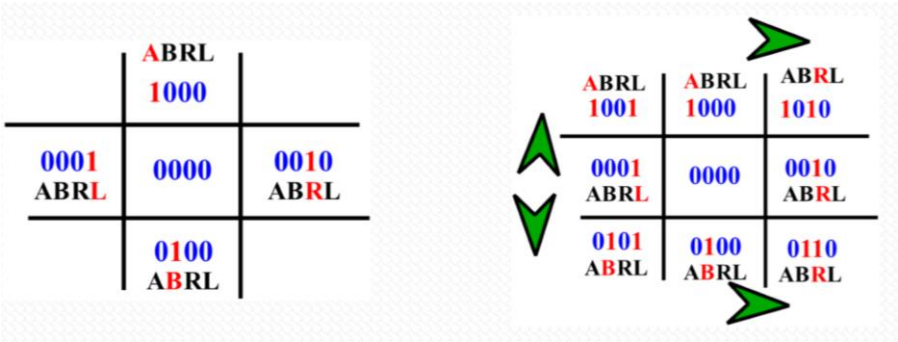
x1,x2< xmin y1,y2< ymin

Xén tỉa: đoạn thẳng cần xén tỉa

Thuật toán được chia thành các bước sau:

Bước 1: Gán mã vùng 4-bit cho mỗi điểm cuối của đoạn thẳng ABRL (Above – Below – Right – Left).

Bước 2: Mã vùng được xác định theo 9 vùng (hình 1.4) của mặt phẳng mà các điểm cuối nằm vào đó. Một bít được cài đặt true (1) hoặc false (0).



Hình 1.4. Mã vùng của 9 vùng mặt phẳng

Bước 3: Sử dụng các mã vùng để xác định các trường hợp của đoạn thẳng

Xét mã vùng của 2 điểm đầu cuối P1, P2 của đoạn thẳng cần xén. Ta có các trường hợp sau:

Nếu mã của P1 hoặc P2 đều = 0000 thì toàn bộ đoạn thẳng thuộc phần hiển thị.

Nếu mã của P1 và P2 có cùng một vị trí mà P1 AND P2 != 0000 => cùng phía.

Nếu không nằm trong 2 trường hợp sau đường thẳng cần được xén tỉa

Tìm giao điểm của đường thẳng với cửa sổ, (với phần mở rộng của đường biên).

Nếu: Bit 1 là 1: xén y = ymax

Bit 2 là 1: xén y = ymin

Bit 3 là 1: xén x = xmax

Bit 4 là 1: xén x = xmin

Tìm giao điểm (x,y) của (x1,y1) (x2,y2) với cửa sổ xén

Ta có: $m = \frac{y-y1}{x-x1}$

hay $y - y1 = m(x - x1) \rightarrow y = y1 + m(x - x1)$

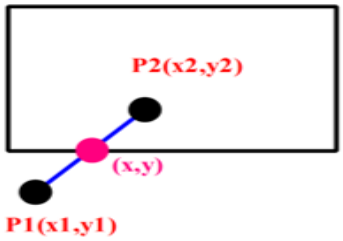
Nếu y nằm trên đường thẳng đứng: $\begin{bmatrix} x = xmin \\ x = xmax \end{bmatrix}$

Nếu x nằm trên đường thẳng nằm ngang:

Từ phương trình: $m = \frac{y-y1}{x-x1} \rightarrow x - x1 = \frac{y-y1}{m}$

$$hay\ x = x1 + \frac{(y-y1)}{m}$$

và $\begin{bmatrix} y = xmin \\ y = ymax \end{bmatrix}$



Hình 1.5 Ví dụ tìm giao diểm

Đối với thuật toán Cohen Shutherland, khi tìm giao điểm của đoạn thẳng cần xén tỉa với các cạnh của cửa sổ xén bằng cách dùng các tham số của phương trình đường thẳng (hệ số góc được tính bằng công thức xuất hiện phép chia). Dẫn tới không tối ưu về mặt thời gian.

Để khắc phục điều đó thuật toán Liang-Barsky ra đời. thuật toán Liang-Barsky (được đặt theo tên của You-Dong Liang và Brian A. Barsky) sử dụng phương trình tham số của đường thẳng và bất đẳng thức mô tả phạm vi của cửa sổ xén để xác định các giao điểm giữa đường thẳng và cửa sổ xén. Với các giao điểm này, nó biết phần nào của đường nên được vẽ. Thuật toán này hiệu quả hơn đáng kể so với Cohen-Sutherland.

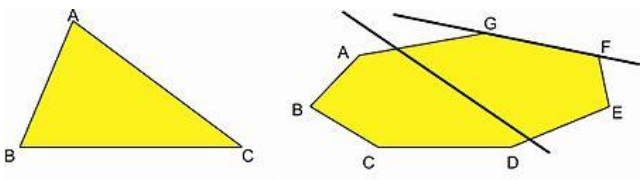
Cả 2 thuật toán trên đều có điểm hạn chế là cửa sổ xén là hình chữ nhật và không cho phép cửa sổ hình đa giác khác.

Thuật toán Cyrus – Beck (1978) đã có thể xén tỉa trên cửa sổ xén là đa giác.

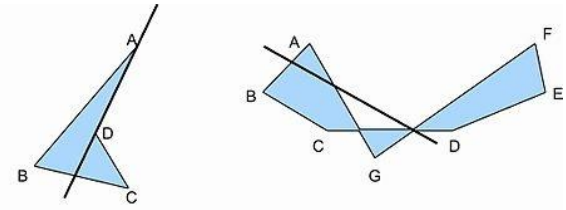
1.3. Thuật toán xén tỉa đa giác

1.3.1. Khái niệm đa giác

Đa giác lồi: toàn bộ đa giác nằm về một phía của đường thẳng chứa cạnh bất kì nào của đa giác. Khi đó, đoạn thẳng nối hai điểm bất kì nào của đa giác đều nằm hoàn toàn trong đa giác.



Hình 1.6. Đa giác lồi



Hình 1.7. Đa giác không lồi

Theo qui ước: một đa giác với các đỉnh P_1, ..., P_N (các cạnh là $P_{i-1}Pi$ và P_NP_1) 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 $P_{i-1}Pi$ và P_NP_1cũng chỉ về bên trong đa giác.

Tính tổng của các cạnh của đa giác theo công thức:

$$(x2 - x1)(y2 + y1)$$

Nếu kết quả lớn hơn 0 thì chiều của đa giác thuận chiều kim đồng hồ và ngược lại.

Cho đa giác có 5 điểm A(5,0); B(6,4); C(4,5); D(1,5); E(1,0)

Xét cạnh AB: (6-5)(4+0) = 4

Xét cạnh BC: (4-6)(5+4) = -18

Xét cạnh CD: (1-4)(5+5) = -30

Xét cạnh DE: (1-1)(0+5) = 0

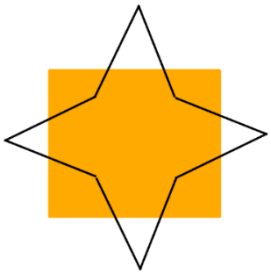
Xét cạnh EA: (5-1)(0+0) = 0

Total: -44 (đa giác có chiều âm)

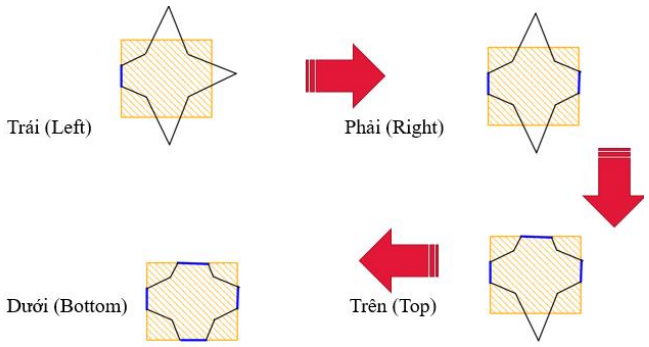
Quy tắc khác:

Để xén đa giác, chúng ta cần sửa đổi các bước xén tỉa đoạn thẳng. Một cạnh đa giác được xử lý với một cạnh của của sổ xén tùy thuộc vào hướng của cạnh đa giác đến cửa sổ xén. Những gì chúng ta cần là phần giới hạn sau khi xén. Để xén tỉa đa giác, chúng ta cần một thuật toán sẽ tạo ra một hoặc nhiều vùng xén (các cạnh của cửa sổ xén). Đầu ra của một đa giác đã được xén tỉa phải là một chuỗi các đỉnh xác định ranh giới của đa giác được xén tỉa.

1.3.2. Thuật toán Sutherland Hogman

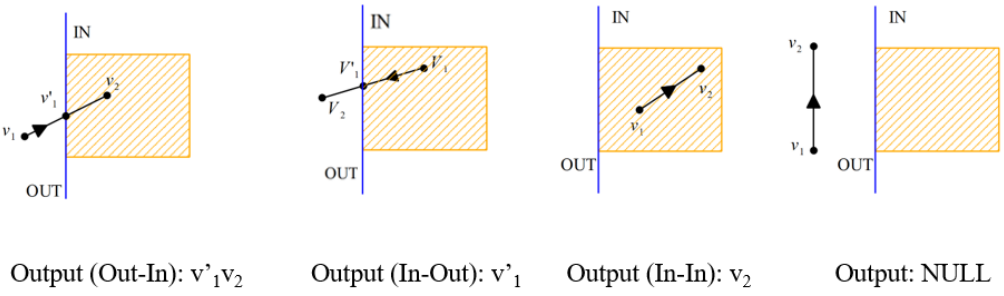


Cho P1,P2,…,PN là danh sách các đỉnh của đa giác. Cho cửa sổ xén tỉa ABCD. Thứ tự các trường hợp sẽ được xén như sau:



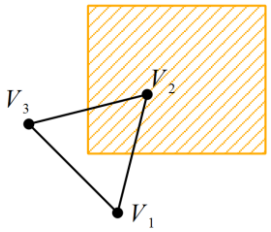
Hình 1.8. Minh hoạ thứ tự xén

Xét 4 trường hợp:

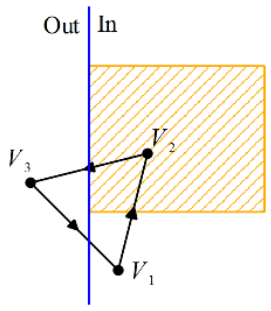


Hình 1.9. Kết quả xén trong từng trường hợp

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



Bước 0: Áp dụng quy luật được nêu ở mục 1.3.1 cho đa giác dương để xác định chiều của các cạnh

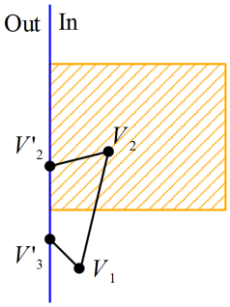


Bước 1: Xén bên trái

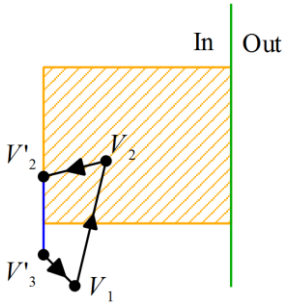
Xét cạnh V_1-V_2 (In-In): V_2

Xét cạnh V_2-V_3 (In-Out): V'_2

Xét cạnh V_3-V_1(Out-In): V_3'-V_1



Bước 2: Xén bên phải



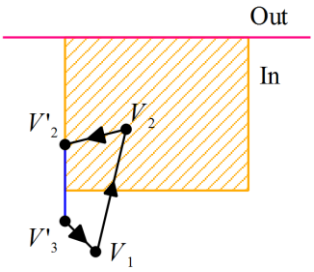
Xét cạnh V_1-V_2 (In-In): V_2

Xét cạnh V_2-V'_2 (In-In): V'_2

Xét cạnh V'_2-V'_3 (In-In): V'_3

Xét cạnh V'_3-V_1(In-In): V_1

Bước 3: Xén bên trên



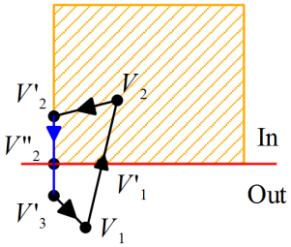
Xét cạnh V_1-V_2 (In-In): V_2

Xét cạnh V_2-V'_2 (In-In): V'_2

Xét cạnh V'_2-V'_3 (In-In): V_3'

Xét cạnh V'_3-V_1(In-In): V_1

Bước 4: Xén bên dưới



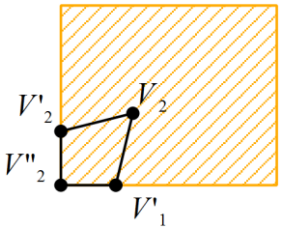
Xét cạnh V_1-V_2 (Out-In): V_1', V_2

Xét cạnh V_2-V'_2 (In-In): V'_2

Xét cạnh V'_2-V'_3 (In-OUT): V_2''

Xét cạnh V'_3-V_1(Out-Out): NULL

Bước 5: Vẽ lại các cạnh lại theo thứ tự điểm ở trên



1.3.3. Các công thức liên quan

Cho đường thẳng d chia mặt phẳng thành 2 nửa LEFT và RIGHT,

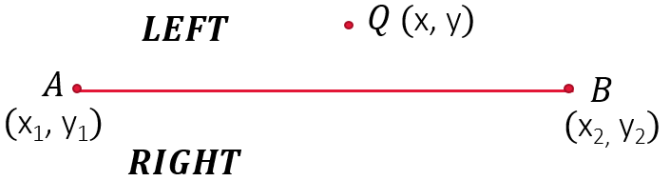
A(x_1, y_1) và B (x_2 , y_2) ∈ d và điểm Q(x, y) ∈ (Oxy)

Ta có : P = (x_2 – x_1)(y – y_1) – (y_2 – y_1)(x – x_1)

Nếu P > 0: Q ∈ LEFT

Nếu P < 0: Q ∈ RIGHT

Nếu P = 0: Q ∈ d



Cho hai đường thẳng d1, d2 có

$A(x_1, y_1)$ và $B(x_2, y_2) \in$ d1

$C(x_3, y_3)$ và $D(x_4, y_4) \in$ d2

Gọi Q(x, y) là giao điểm của d1, d2 khi đó ta có:

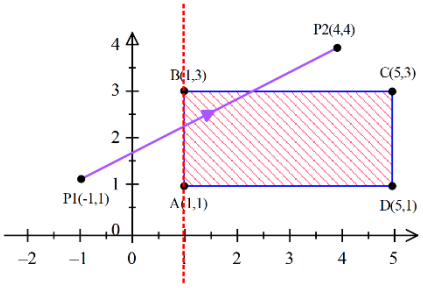
$$x = \frac{(x_1 y_2 - y_1 x_2)(x_3 - x_4) - (x_1 - x_2)(x_3 y_4 - y_3 x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

$$y = \frac{(x_1 y_2 - y_1 x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 y_4 - y_3 x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

1.3.4. Minh họa giải thuật Sutherland Hogman

Đoạn thẳng P1(-1,1) đến P2(4,4) được cắn xén với cửa sổ A(1,1), B(1,3), C(5,3) và D(5,1). (nếu xén đoạn thẳng ở IN-IN thì giữ lại cả 2 điểm)

Bước 1: Xén theo cạnh AB



Xét P1(-1,1) và đoạn thẳng AB, A(1,1) và B(1,3)

Ta có $P = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$

$= (1 - 1)(1 - 1) - (3 - 1)(-1 - 1) = 4 > 0$

=> P1 nằm bên trái AB (OUT)

Xét P2 (4,4)

Ta có $P = (x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$

$= (1 - 1)(4 - 1) - (3 - 1)(4 - 1) = -6 < 0$

=> P2 nằm bên phải AB (IN)

=> P1P2 : OUT-IN

Tìm giao điểm P1' thuộc đường thẳng P1P2

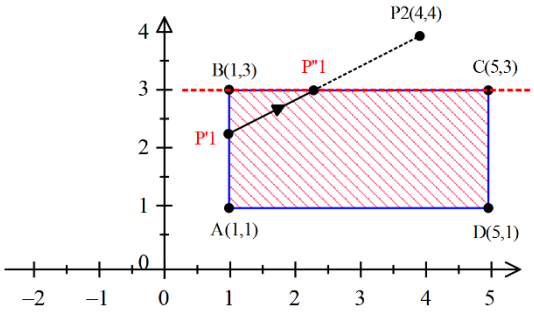
$$m = \frac{4 - 1}{4 + 1} = \frac{3}{5}$$

$$\begin{cases} x_{P'} = 1 \\ y_{P'} = y_{P_1} + (x_{P'} - x_{P_1})m = 1 + (1 - (-1))\frac{3}{5} = \frac{11}{5} \end{cases}$$

$\Rightarrow$ P1'$\left(1, \frac{11}{5}\right)$

OUTPUT: P1', P2

Bước 2: Xén theo cạnh CD



Xét P1'$\left(1,\frac{11}{5}\right)$ và đoạn thẳng CD, C(5,3) và D(5,1)

Ta có P = $(x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$

= (5 – 5)(11/5 – 3) – (1 – 3)(1 – 5) = -8 < 0

=> P1' nằm bên phải AB (IN)

Xét P2 (4,4)

Ta có P = $(x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$

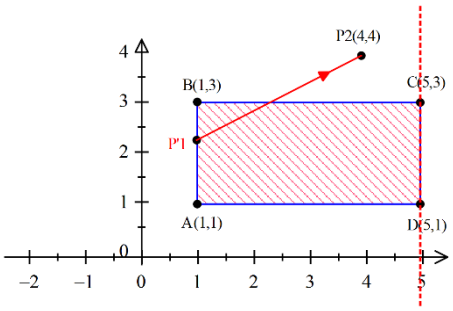
= (5 – 5)(4 – 3) – (1 – 3)(4 – 5) = - 2 < 0

=> P2 nằm bên phải AB (IN)

=> P1'P2 : IN-IN

OUTPUT: P1', P2

Bước 3: Xén theo cạnh BC



Xét P1'$\left(1,\frac{11}{5}\right)$ và đoạn thẳng BC, B(1,3) và C(5,3)

Ta có P = $(x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$

= (5 – 3)(11/5 – 3) – (3 – 3)(1 – 1) = - 8/5 < 0

=> P1' nằm bên phải BC (IN)

Xét P2 (4,4)

Ta có P = $(x_2 - x_1)(y - y_1) - (y_2 - y_1)(x - x_1)$

= (5 – 3)(4 – 3) – (3 – 3)(4 – 1)= 2 > 0

=> P2 nằm bên trái BC (OUT)

=> P1'P2 : IN-OUT

Tìm giao điểm P''$_1$ thuộc đường thẳng P'$_1$P$_2$

$$m = \frac{4 - 11/5}{4 - 1} = \frac{3}{5}$$

$$\begin{cases} y_{P''1} = 3 \\ x_{P''_1} = xP'1 + (y_{P''_1} - yP'1)/m = 1 + (3 - 11/5)\frac{5}{3} = \frac{7}{3} \end{cases}$$

⇨ P1"$\left(\frac{7}{3},3\right)$

OUTPUT P1', P1''

Bước 4: Xén theo cạnh DA

Xét P1’$\left(1,\frac{11}{5}\right)$ và đoạn thẳng DA, D(5,1) và A(1,1)

Ta có P = $(x_2 – x_1)(y – y_1) – (y_2 – y_1)(x – x_1)$

= $(1 – 5)(11/5 – 1) – (1 – 1)(1 – 1)$ = -24 / 5 < 0

=> P1’ nằm bên phải DA (IN)

Xét P1’’$\left(\frac{7}{3},3\right)$

Ta có P = $(x_2 – x_1)(y – y_1) – (y_2 – y_1)(x – x_1)$

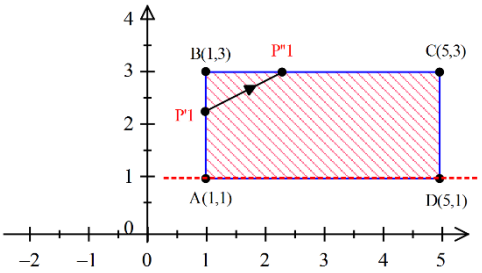
= $(1 – 5)(3 – 1) – (1 – 1)(7/3 – 1)$ = - 8 < 0

=> P1’’ nằm bên phải DA (IN)

=> P1’P1’’ : IN-IN

OUTPUT: P1’, P1’’

Vậy đoạn thẳng giữ lại là P1’P1’’

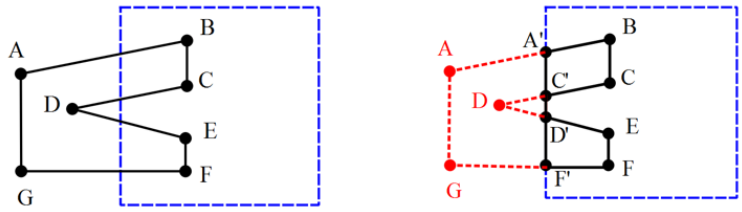


Thuật toán Sutherland Hogman dùng để xén tỉa đa giác, thuật toán này hoạt động bằng cách lần lượt mở rộng từng cạnh của cửa sổ xén và chỉ chọn các đỉnh từ đa giác nằm ở phía có thể nhìn thấy.

Thuật toán bắt đầu với một danh sách đầu vào của tất cả các đỉnh trong đa giác cần xén. Tiếp theo, một bên của cửa sổ xén được mở rộng vô hạn theo cả hai hướng, các đỉnh từ danh sách đầu vào được chèn vào danh sách đầu ra nếu chúng nằm ở phía có thể nhìn thấy của cạnh cửa sổ xén mở rộng và các đỉnh mới được thêm vào danh sách đầu ra khi chúng giao với cạnh của cửa sổ xén.

Quá trình này được lặp đi lặp lại cho mỗi cạnh của cửa sổ, sử dụng danh sách đầu ra từ một giai đoạn trước(sau khi xén lần trước) làm danh sách đầu vào cho lần tiếp theo. Khi tất cả các cạnh của cửa sổ xén đã được xử lý, danh sách các đỉnh được tạo cuối cùng sẽ xác định một đa giác mới

Nếu đa giác được xén lõm ở các đỉnh bên ngoài cửa sổ xén, đa giác mới có thể có các cạnh trùng nhau (nghĩa là chồng chéo) - điều này có thể chấp nhận được để hiển thị, nhưng không được cho phép trong kĩ thuật xử lý đồ hoạ bậc cao GPU.



Hình 1.10. Trường hợp sai khi xén đa giác có đỉnh lõm ở ngoài cửa sổ xén

Thuật toán Weiler-Atherton khắc phục điều này bằng cách trả lại một tập hợp các đa giác của đa giác sau khi xén, nhưng phức tạp hơn và tính toán nhiều hơn.

Một hoặc nhiều đa giác lõm có thể tạo ra nhiều hơn một đa giác. Đa giác lồi sẽ chỉ có một đa giác. Thuật toán tương tự có thể được sử dụng để hợp nhất hai đa giác bằng cách bắt đầu tại các giao điểm bên ngoài thay vì các giao điểm bên trong. Các điểm gần với cạnh của đa giác khác thường bị nhầm là điểm trong đa giác đến khi trạng thái của chúng được xác nhận sau khi tất cả các giao điểm đã được tìm thấy và xác minh. Tuy nhiên, điều này làm tăng sự phức tạp.

1.4. Dev C++

Bloodshed Dev-C ++ (https://www.bloodshed.net/devcpp.html) là môi trường phát triển tích hợp (IDE) đầy đủ tính năng cho ngôn ngữ lập trình C/C ++ sử dụng Mingw của GCC (Bộ sưu tập trình biên dịch GNU) làm trình biên dịch. Dev-C ++ cũng có thể kết hợp với Cygwin hoặc bất kỳ trình biên dịch dựa trên GCC nào khác.

Các tính năng của Dev-C++:

Hỗ trợ trình biên dịch dựa trên GCC

Gỡ lỗi tích hợp (sử dụng GDB- General DeBug)

Quản lý dự án

Trình chỉnh sửa cú pháp

Trình duyệt lớp

Hoàn thành mã

Danh sách chức năng

Hồ sơ hỗ trợ

Nhanh chóng tạo Windows, console, thư viện tĩnh và DLL

Hỗ trợ các mẫu để tạo các loại dự án của riêng bạn

Tạo Makefile

Chỉnh sửa và biên dịch các tệp Tài nguyên

Quản lý công cụ

Hỗ trợ in

Tìm và thay thế mã lệnh

Hỗ trợ CVS

1.5. Thư viện Graphics.h

Vì sử dụng DevC++ làm trình biên dịch cho việc cài đặt thuật toán nên không thể thực hiện trên môi trường Windows. Vì vậy, một môi trường giả lập graphic của Borland C được Michael tạo ra thư viên có tên là Graphics.h. để có thể làm được điều đó. Micheal đã thay đổi BGI library (thư viện BGI) thành thư viện có tên WinBGIm để có thể sử dụng tốt trên windows. Và bây giờ bạn đã có thể sử dụng tốt các hàm đặc biệt của borland bằng DevC++ (https://github.com/SagarGaniga/Graphics-Library)

2. PHƯƠNG PHÁP NGHIÊN CỨU

2.1. Cài đặt DevC và thư viện graphics.h

Tải file cài đặt phần mềm DevC++ theo đường dẫn trong mục 1.4. Sau đó mở file vừa tải, và tiến hành cài đặt. Thư viện graphics.h được tiến hành cài đặt theo các bước:

Bước 1: Copy 6-ConsoleAppGraphics và ConsoleApp_cpp_graph

Paste C:\Program Files\Dev-Cpp\Templates

Bước 2: Copy graphics và winbgim

Paste C:\Program Files\Dev-Cpp\MinGW64\x86_64-w64-mingw32\include

Bước 3: Copy libbgi.a

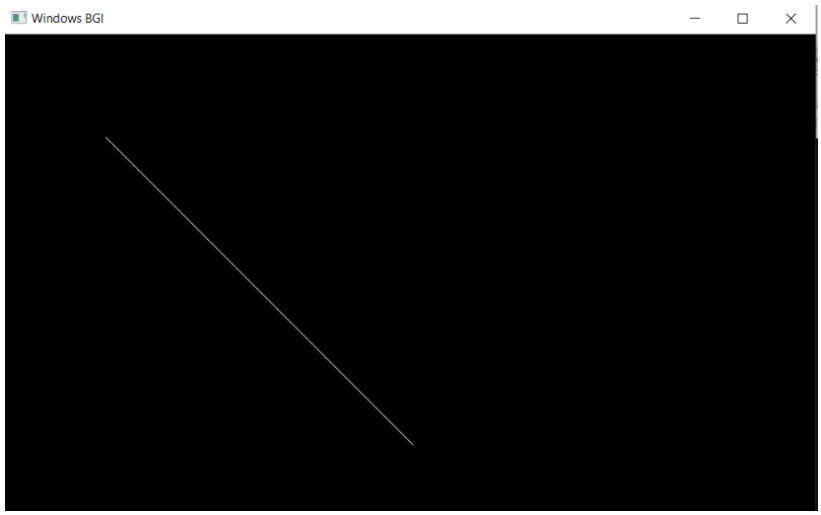
Paste C:\Program Files\Dev-Cpp\MinGW64\x86_64-w64-mingw32\lib

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 Graphics.h

```
#include <graphics.h> // thư viện graphics.h
int main()
{
    initwindow(800,800); // Khởi tạo kích thước cửa sổ BGI
    line(100,100,400,400); // Hàm để vẽ 1 đoạn thẳng
    getch();
}
```



Hình 2.1. Ví dụ minh hoạ thư viện graphics

2.2. Cài đặt thuật toán

Khai báo kiểu dữ liệu đầu vào

```
struct point
{
    int x,y;
};// kiểu cấu trúc cho điểm gồm toạ độ x,y
point poly[100]; // mảng lưu các đỉnh của đa giác cần x
point poly1[100]; // mảng tạm lưu các đỉnh sau khi xén
point clip[4];// lưu các đỉnh của cửa sổ xén
point pt[100];//pt[0],pt[1] lưu 2 điểm min max của cửa sổ xén, còn các
phần tử sau dùng cho các chức năng khác(lưu toạ độ đoạn thẳng . . .)
int n1 = 0 ; // số đỉnh của đa giác
```

Nhập dữ liệu đầu vào

Nhập dữ liệu đầu vào từ bàn phím

```
void KhoiTao()
{ // Hàm nhập dữ liệu ban đầu từ bàn phím
    // với cửa sổ graphic có kích thước (140,20) -> (720,480) nên ràng
buộc dữ liệu đầu vào
    printf("\nNHAP PHAI THOA MAN DIEU KIEN \n|| 140<X<720 && 20<Y<480
||\n");
    do {
        printf("Nhap hinh chu nhat Clipping\n");
        printf("xmin = "); scanf("%d",&pt[0].x);
        printf("ymin = "); scanf("%d",&pt[0].y);
        printf("xmax = "); scanf("%d",&pt[1].x);
        printf("ymax = "); scanf("%d",&pt[1].y);
        }
```

```
    while(140 >= pt[0].x || 720<=pt[0].x  ||  20>=pt[0].y || 480 <=
pt[0].y || 140 >= pt[1].x || 720<=pt[1].x  ||  20>=pt[1].y || 480 <=
pt[1].y);

    rectangle(pt[0].x,pt[0].y,pt[1].x,pt[1].y);
    printf("nhap so canh cua da giac :");
    scanf("%d",&n1);
    int i = 0;
    while(i<n1)
    {   printf("Nhap toa do dinh thu %d\n",i+1);
        printf("x = "); scanf("%d",&poly[i].x);
        printf("y = "); scanf("%d",&poly[i].y);
        if(140<poly[i].x && 720>poly[i].x  &&  20<poly[i].y && 480 >
poly[i].y)
        {
            i++;
        }
        else {
            printf(" diem ban vua nhap khong nam trong cua so , vui long
nhap lai \n");
        }

    }

    for (int i=0; i<n1; i++)
    {   int k=(i+1)%n1;
        line(poly[i].x,poly[i].y,poly[k].x,poly[k].y);
   }
}
```

Nhập dữ liệu đầu vào bằng chuột qua giao diện

```
struct Button
{
    int x1, y1, x2, y2;
};// toạ độ điểm đầu- cuối của 1 nút trên giao diện

bool check_button(Button bt,point p)
// kiểm điểm p(thường là toạ độ của chuột) có đang nằm trong nút hay
không!
{
       if(bt.x1<=p.x && bt.y1<=p.y&& bt.x2>=p.x&& bt.y2>=p.y)
          return true;
       return false;
}
```

```
if (check_button(btPoly,p)) { // Kiểm tra sự kiện nhấn vẽ đa giác
    while(1){
        delay(0.01);
        point p1;  // tạo biến tạm để lưu dữ liệu trước khi truyền cho
mảng poly
        if (ismouseclick(WM_LBUTTONDOWN))
        {
            getmouseclick(WM_LBUTTONDOWN, x, y); // nhận điểm của đa
giác khi click chuột trái
            clearmouseclick(WM_LBUTTONDOWN);
            p1 = {x,y};
            if (!check_button(btboder,p1)) break;
            poly[n1] = p1;
            n1++;
        }
```

```
        if(ismouseclick(WM_RBUTTONDOWN))
        {
            getmouseclick(WM_RBUTTONDOWN, x, y); // nhận điểm cuối của
đa giác khi click chuột phải
            clearmouseclick(WM_RBUTTONDOWN);
            if (!check_button(btboder,p1)) break;
            p1 = {x,y};
            poly[n1] = p1;
            n1++;
            setcolor(color);
            for (int i= 0; i<n1; i++)
            {   int k =(i+1) %n1;
                line(poly[i].x,poly[i].y,poly[k].x,poly[k].y); // tiến
hành vẽ đa giác qua các đỉnh đã nhận
            }

        }
    }
}
```

Nhập cửa sổ xén

```
void addpointwd(){
// chuyển min max của cửa sổ xén thành 4 điểm để xén đa giác
    clip[0] = pt[0]; // xmin-ymin
    clip[1].x = pt[0].x; //xmin
    clip[1].y = pt[1].y; //ymax
    clip[2] = pt[1];//xmax-ymax
    clip[3].x = pt[1].x;//xmax
    clip[3].y = pt[0].y;//ymin
 }
void hoandoi(){
 // hoán đổi 2 điểm của cửa sổ xén cho về thứ tự min->max để tiện tính
toán sau này
    if (pt[0].x > pt[1].x) swap(pt[0].x, pt[1].x);
    if (pt[0].y > pt[1].y) swap(pt[0].y, pt[1].y);
    rectangle(pt[0].x, pt[0].y, pt[1].x, pt[1].y);
}
if (check_button(btCrop,p)){
    while(1){
        delay(0.01);
        point p1,p2;// lưu 2 điểm min - max của cửa sổ xén
        if (ismouseclick(WM_LBUTTONDOWN)){
            getmouseclick(WM_LBUTTONDOWN, x, y);
            clearmouseclick(WM_LBUTTONDOWN);
            p1 = {x,y};
            if (!check_button(btboder,p1)) break;
        }
        if(ismouseclick(WM_LBUTTONUP)){
            getmouseclick(WM_LBUTTONUP, x, y);
            clearmouseclick(WM_LBUTTONUP);
            p2 = {x,y};
            if (!check_button(btboder,p2)) break;
            setcolor(color);
            setlinestyle(0,0,0);
            rectangle(p1.x,p1.y,p2.x,p2.y);
            pt[0]={p1.x,p1.y};
            pt[1]={p2.x,p2.y};
            hoandoi();
        }
    }
}
```

Cài đặt thuật toán xén tỉa đa giác

```
int x_intersect(point p1,point p2, point p3, point p4){
 //hàm trả về toạ độ x - giao điểm của 2 đoạn thẳng p1p2 với p3p4
//p1,p2 là toạ độ của 1 cạnh cửa sổ xén.
//p3,p4 là toạ đồ của 1 cạnh đa giác cần xén
    int num = (p1.x*p2.y - p1.y*p2.x) * (p3.x-p4.x) -
              (p1.x-p2.x) * (p3.x*p4.y - p3.y*p4.x);
    int den = (p1.x-p2.x) * (p3.y-p4.y) -
             (p1.y-p2.y) * (p3.x-p4.x);
    return round(num/den);} // công thức được nêu ở mục XXXXX
int y_intersect(point p1,point p2, point p3, point p4){
//hàm trả về toạ độ y - giao điểm của 2 đoạn thẳng p1p2 với p3p4
// tương tự như hàm x_intersect
    int num = (p1.x*p2.y - p1.y*p2.x) * (p3.y-p4.y) -
              (p1.y-p2.y) * (p3.x*p4.y - p3.y*p4.x);
    int den = (p1.x-p2.x) * (p3.y-p4.y) -
             (p1.y-p2.y) * (p3.x-p4.x);
    return round(num/den); }
void cllip(point p1,point p2){

    int n2 = 0;// biến tạm lưu số lượng đỉnh qua từng lần xén
    for (int i = 0; i < n1; i++){
        int k = (i+1) % n1;
        int ix = poly[i].x, iy = poly[i].y;
        int kx = poly[k].x, ky = poly[k].y;
    // Công thức xác định điểm nằm trong hay nằm ngoài( cũng như trái
       hay phải) nêu ở mục XXXXX
        int i_pos = (p2.x-p1.x) * (iy-p1.y) - (p2.y-p1.y) * (ix-p1.x);
        int k_pos = (p2.x-p1.x) * (ky-p1.y) - (p2.y-p1.y) * (kx-p1.x);
// TH1: Khi cả 2 điểm đều nằm trong so với cạnh của cửa sổ xén đang xét
        if (i_pos < 0  && k_pos < 0) {
            poly1[n2].x = kx;
            poly1[n2].y = ky;
            // lưu điểm vào mảng tạm
            n2++;
        }
// TH2: Khi điểm đầu tiên nằm ngoài,điểm thứ 2 nằm trong so với cạnh
của cửa sổ xén đang xét
        else if (i_pos >= 0  && k_pos < 0){
            poly1[n2].x = x_intersect(p1,p2,poly[i],poly[k]);
            poly1[n2].y = y_intersect(p1,p2,poly[i],poly[k]);
            n2++;
            poly1[n2].x = kx;
            poly1[n2].y = ky;
            n2++;
        }
// TH3: Khi điểm đầu tiên nằm trong,điểm thứ 2 nằm ngoài so với cạnh
của cửa sổ xén đang xét
        else if (i_pos < 0  &&  k_pos >= 0){
           poly1[n2].x = x_intersect(p1,p2,poly[i],poly[k]);
            poly1[n2].y = y_intersect(p1,p2,poly[i],poly[k]);
            n2++;
        }
// TH4: Khi cả 2 điểm đều nằm ngoài so với cạnh của cửa sổ xén đang xét
        else{// không thực hiện lưu điểm nào cả}
    }
// sau khi xén tất cả các cạnh của đa giác,thì chép kết quả vừa thu được
// lưu vào mảng poly cho lần xén với cạnh cửa sổ xén tiếp theo
    n1 = n2;
    for (int i = 0; i < n1; i++){ poly[i] = poly1[i];}
}
```

```
void Poly_Hogman(){
    cllip(clip[0],clip[1]); // xét với cạnh trái của cửa sổ xén
    cllip(clip[2],clip[3]);// xét với cạnh phải của cửa sổ xén
    cllip(clip[1],clip[2]);// xét với cạnh trên của cửa sổ xén
    cllip(clip[3],clip[0]);// xét với cạnh dưới của cửa sổ xén
    setcolor(RED);
    setlinestyle(0,0,3);
    //vẽ lại đa giác sau khi xén
    for (int i = 0; i < n1; i++) {
        int k= (i+1)%n1;
        line(poly[i].x,poly[i].y,poly[k].x,poly[k].y);
    }
}
```

2.3. cài đặt giao diện

Vẽ các nút

```
void button(Button bt, int color_fill) {
// để vẽ nút, 1 nút được truyền vào có toạ độ bt được khai báo như phần
đầu
    setlinestyle(0,0,2);// thay đổi kiểu đường vẽ, độ dày của đường
    setfillstyle(1,color_fill);//thay đổi màu trong nút
    setcolor(15); // thiết lập màu vẽ viền
    bar3d(bt.x1, bt.y1, bt.x2, bt.y2,20,1);// vẽ nút 3D}
```

Vẽ hình ảnh bên trong nút

```
void button_line(Button bt, int color_fill, int color_line){

//Vẽ dữ liệu bên trong nút
    button(bt, color_fill);// dùng hàm trên vẽ 1 nút
    setcolor(color_line); // thay đổi màu
    line((bt.x2-bt.x1)/5*2, (bt.y2-bt.y1)/5+bt.y1, (bt.x2-bt.x1),
(bt.y2-bt.y1)/5*4+bt.y1);
    //vẽ hình bên trong(TH này là đường thẳng)}
```

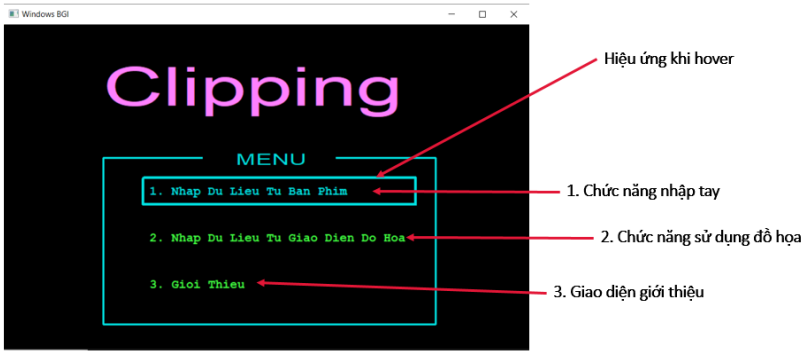
Cài đặt hiệu ứng hover

```
while(1){

    delay(0.1);
    if (ismouseclick(WM_MOUSEMOVE)){ // nhận sự kiện di chuột
        int x,y;
        char s[100];
        getmouseclick(WM_MOUSEMOVE, x, y);// lấy toạ độ chuột
        point p = {x,y};
        //Hover button poly
        if (check_button(btPoly,p)){//kiểm tra toạ độ chuột so với nút
            setcolor(15);
            button_poly(btPoly, 5 , 3);//thay đổi màu,hiệu ứng cho nút
            temp3=1;}
        else if (!check_button(btPoly, p) && temp3==1){//khi chuột di ra
ngoài nút
            setcolor(15);
            button_poly(btPoly, 2 , 15);//trả lại màu,hiệu ứng ban đầu
            temp3=0;}
        setcolor(15);
        sprintf(s,"\t Toa Do (%d,%d) \t",x,y);
        //ghi ra màn hình winbgi toạ độ khi di chuột
        outtextxy(380,20,s);}}
```

3. KẾT QUẢ

3.1. Giao diện menu chính



Hình 3.1. Giao diện menu chính

Khi đưa con trỏ chuột vào sẽ xuất hiện hiệu ứng hover đổi button thành màu khác

L_Click vào mục 1 sẽ dẫn tới giao diện nhập tay để xén tỉa 1 đa giác

L_Click vào mục 2 sẽ dẫn tới giao diện đồ họa sử dụng chuột để thực hiện thao tác

L_Click vào mục 3 sẽ dẫn tới giao diện giới thiệu về phần mềm

3.2. Giao diện nhập liệu từ bàn phím

Khi L_Click button insert thì cửa console cho phép nhập dữ liệu

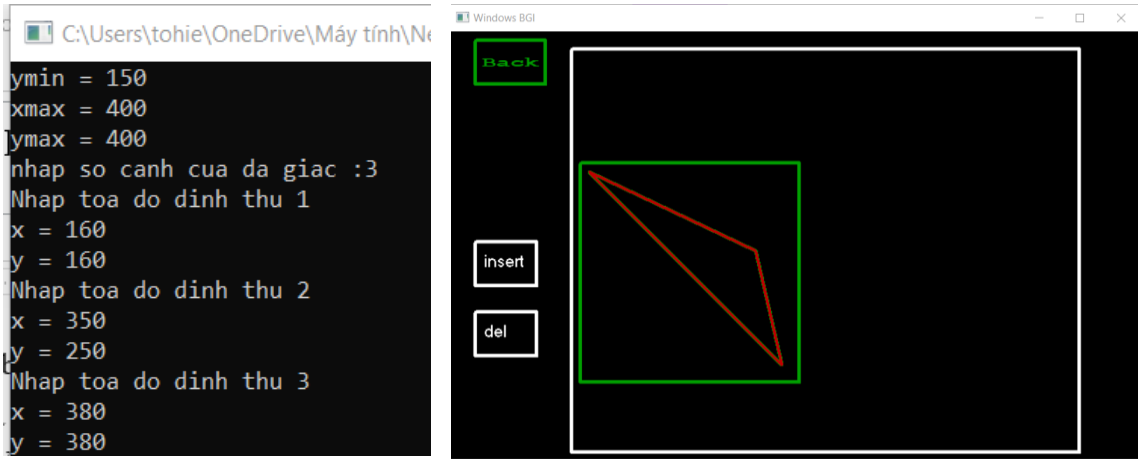
Sau khi nhập xong, nhấn phím enter để kết thúc việc nhập và lưu dữ liệu

Sau đó tiến hành xén đa giác vừa nhập và hiển thị lên màn hình BGI

Nhấn L_Click lên button del để xoá dữ liệu vừa vẽ.

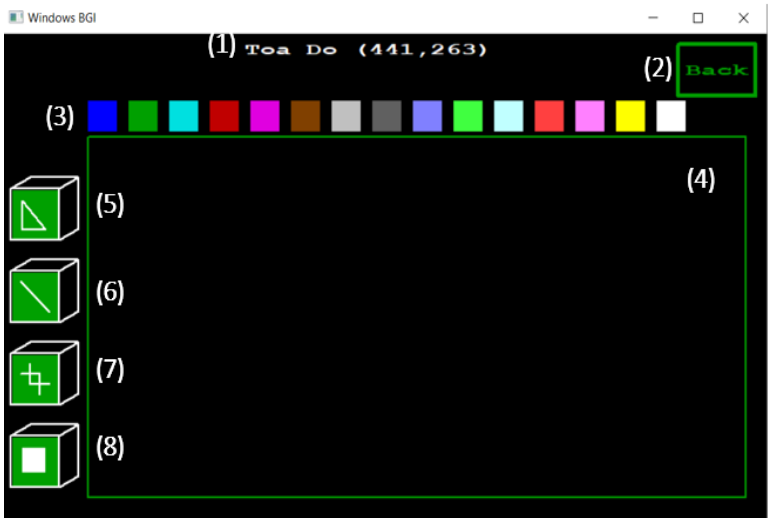
Chỉ vẽ được nhiều nhất 1 đa giác trên khung vẽ.

Nhấn L_Click lên button Back để quay lại cửa sổ chính



Hình 3.2. Giao diện nhập từ bàn phím

3.3. Giao diện đồ họa sử dụng chuột



(1) Trạng thái con trỏ chuột

(2) Button quay trở lại giao diện menu chính

(3) Bảng màu để vẽ

(5) Button vẽ đa giác

(6) Button vẽ đường thẳng

(7) Button vẽ cửa sổ xén

(8) Button để xóa màn hình

Hình 3.3. Giao diện đồ hoạ sử dụng chuột

Trạng thái con trỏ chuột

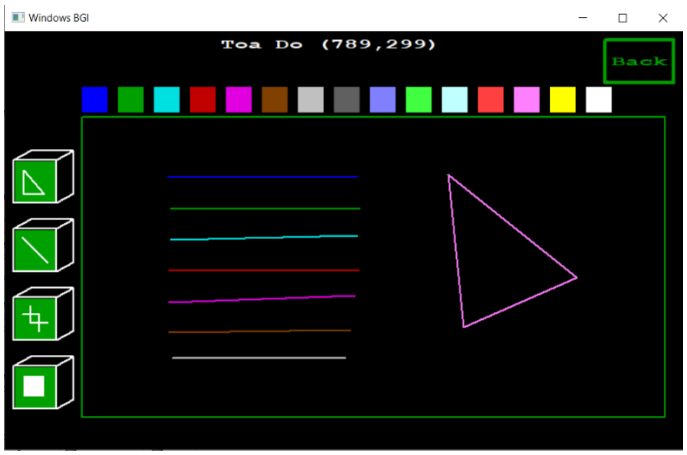
Khi di chuyển con trỏ chuột thì sẽ hiển thị đúng vị trí con trỏ chuột

Button Back

Khi L_Click vào button Back sẽ quay trở lại menu chính lúc ban đầu

Màu vẽ

Khi L_Click vào các button màu vẽ phía trên sẽ thay đổi màu vẽ vừa click vào



Hình 3.4. Minh hoạ sử dụng button màu

Button vẽ đa giác

Khi L_Click button vẽ đa giác có hình tam giác trên màn hình và L_Click lần sau ra ngoài phạm vi khung vẽ thì chế độ đó tắt

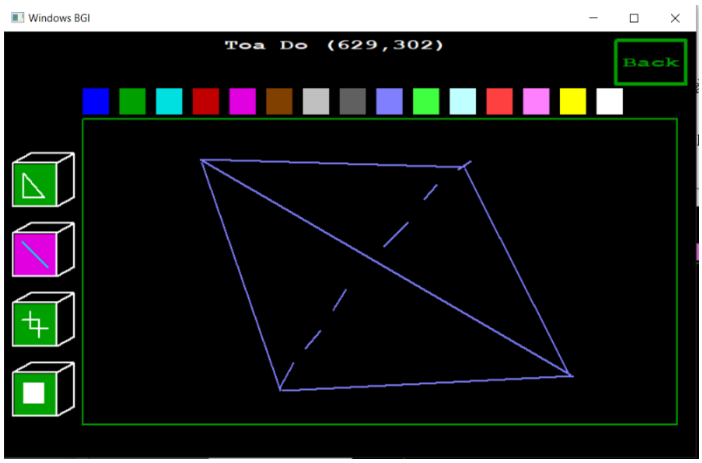
Sau khi L_Click vào button sau đó lần lượt L_Click các điểm muốn vẽ vào khung cửa sổ vẽ, và kết thúc bằng R_Click để vẽ đa giác.

Button vẽ đường thẳng

Khi L_Click button vẽ đoạn thẳng có hình đoạn thẳng trên màn hình và L_Click lần sau ra ngoài phạm vi khung vẽ thì chế độ đó tắt

Sau khi L_Click vào button sau đưa con trỏ chuột vào khung vẽ và L_Click giữ kéo đến vị trí của điểm thứ hai thì thả ra để vẽ

Có thể vẽ được nhiều đoạn thẳng trên khung vẽ.

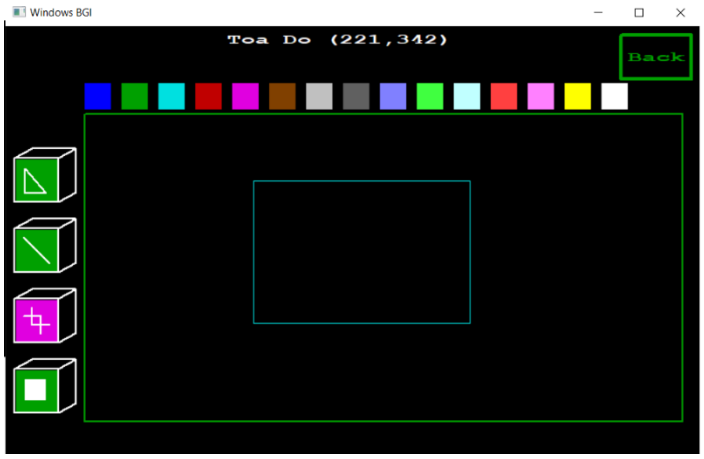


Hình 3.6. Minh hoạ sử dụng button vẽ đường thẳng

Button vẽ khung cửa sổ xén

Khi L_Click button vẽ cửa sổ xén trên màn hình và L_Click lần sau ra ngoài phạm vi khung vẽ thì chế độ đó tắt

Sau khi L_Click vào button sau đó đưa con trỏ chuột vào khung vẽ và L_Click giữ kéo đến vị trí của điểm thứ hai thì thả ra để vẽ cửa sổ xén



Hình 3.7. Minh hoạ sử dụng button vẽ cửa sổ xén

Button xóa màn hình

Khi L_Click vào màn hình toàn bộ dữ liệu hình vẽ trước đó được hiển thị trên khung vẽ sẽ bị xóa, và reset bộ nhớ về trạng thái ban đầu.

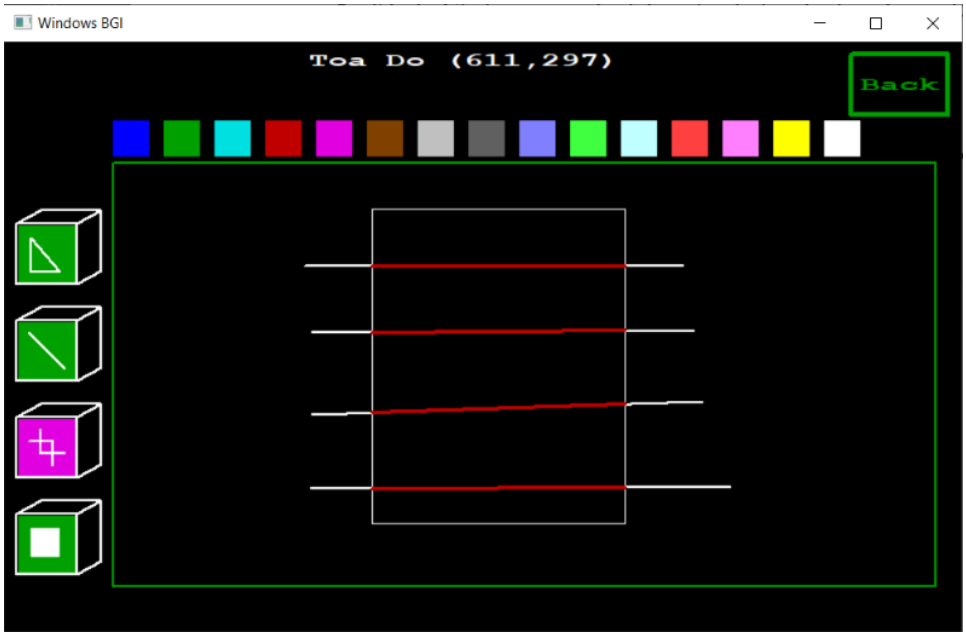
3.4. Xén tỉa đoạn thẳng bằng đồ họa

Để xén tỉa một hay nhiều đoạn thẳng thì cần thực hiện 2 bước

Bước 1: Vẽ các đoạn thẳng xem mục 3.3

Bước 2: Vẽ cửa sổ xén xem mục 3.3

Sau khi vẽ cửa sổ xén thì phần đoạn thẳng nằm trong cửa sổ xén sẽ chuyển màu đỏ.



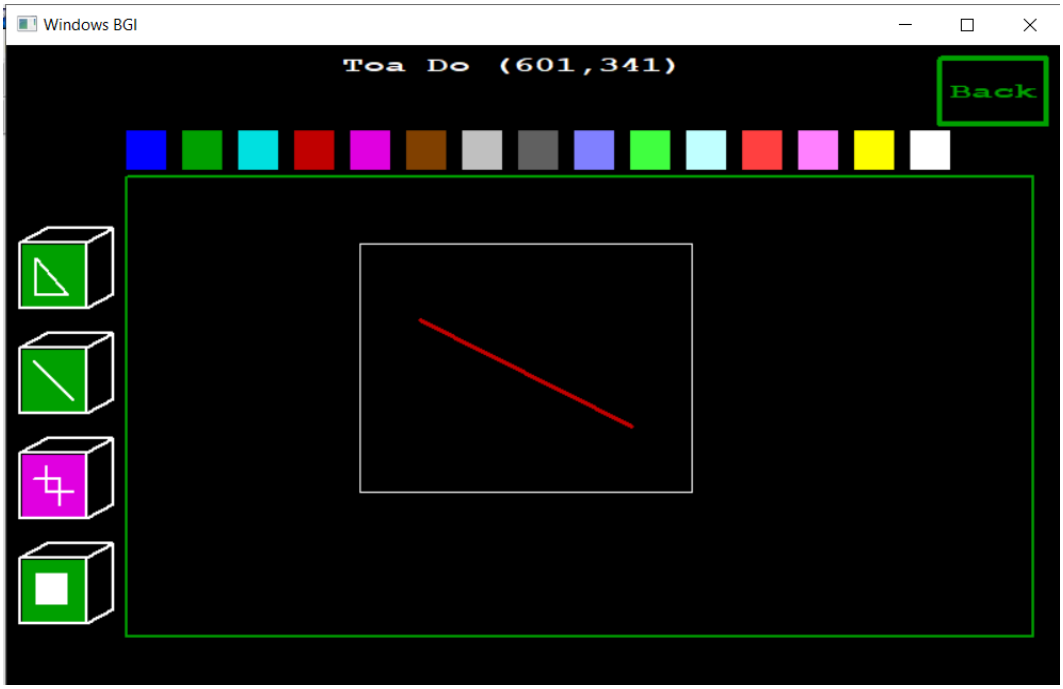
Hình 3.8. Minh hoạ về xén tỉa đoạn thẳng

Các trường hợp của xén tỉa đoạn thẳng

TH1: Đoạn thẳng nằm ngoài cửa sổ xén

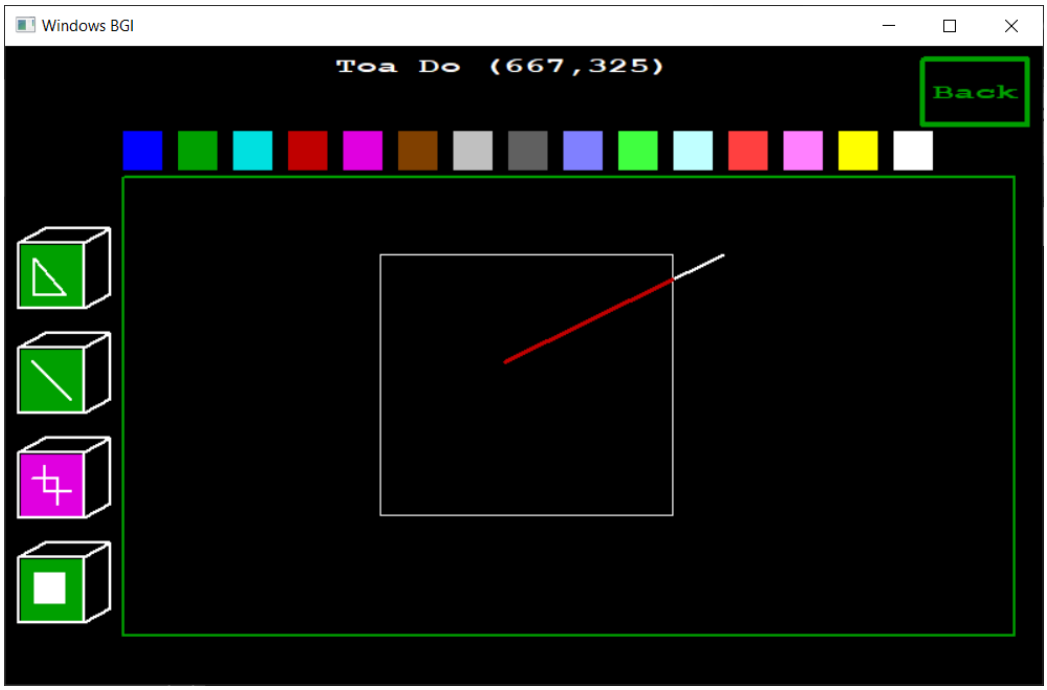
Hình 3.9. Minh hoạ về trường hợp đoạn thẳng nằm ngoài cửa sổ xén

TH2: Đoạn thẳng nằm hoàn toàn trong cửa sổ xén



Hình 3.10. Minh hoạ về trường hợp đoạn thẳng nằm trong cửa sổ xén

TH3: Đoạn thẳng nằm cả trong lẫn ngoài



Hình 3.11. Minh hoạ về trường hợp đoạn thẳng nằm cả trong lẫn ngoài

3.5. Xén tỉa đa giác bằng đồ họa

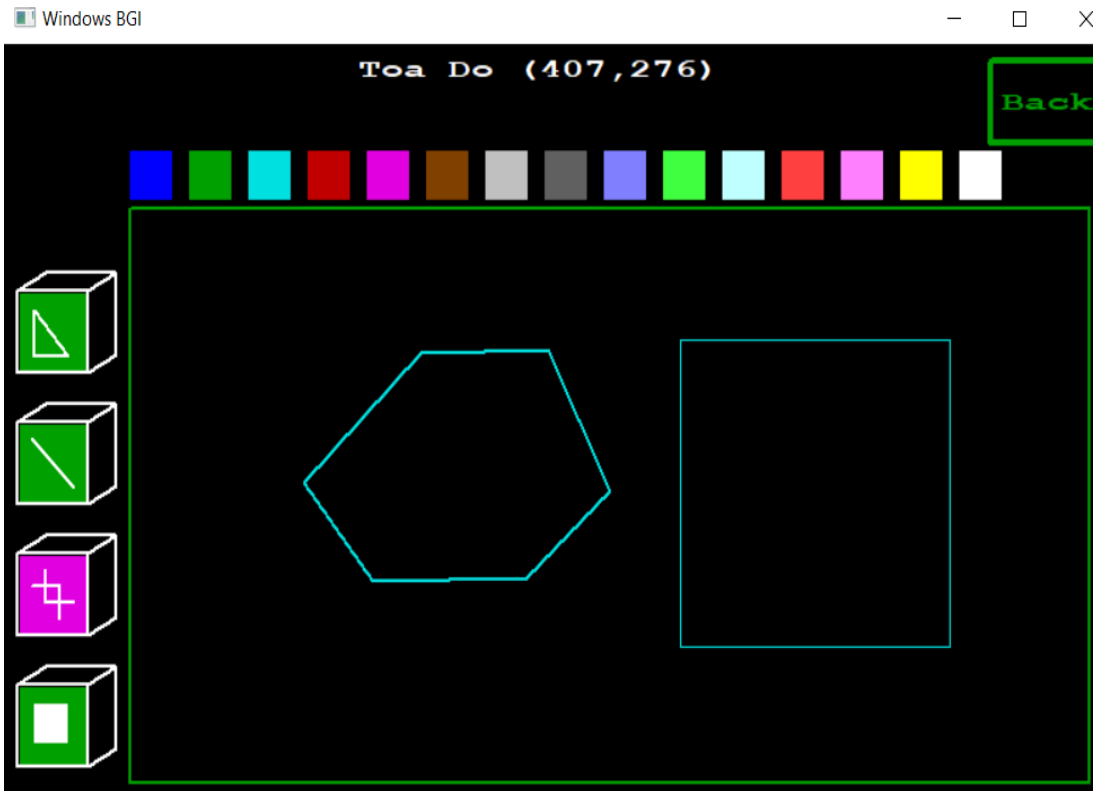
Để xén tỉa một hay nhiều đoạn thẳng thì cần thực hiện 2 bước

Bước 1: Vẽ đa giác cần xén xem mục 3.3

Bước 2: Vẽ cửa sổ xén xem mục 3.3

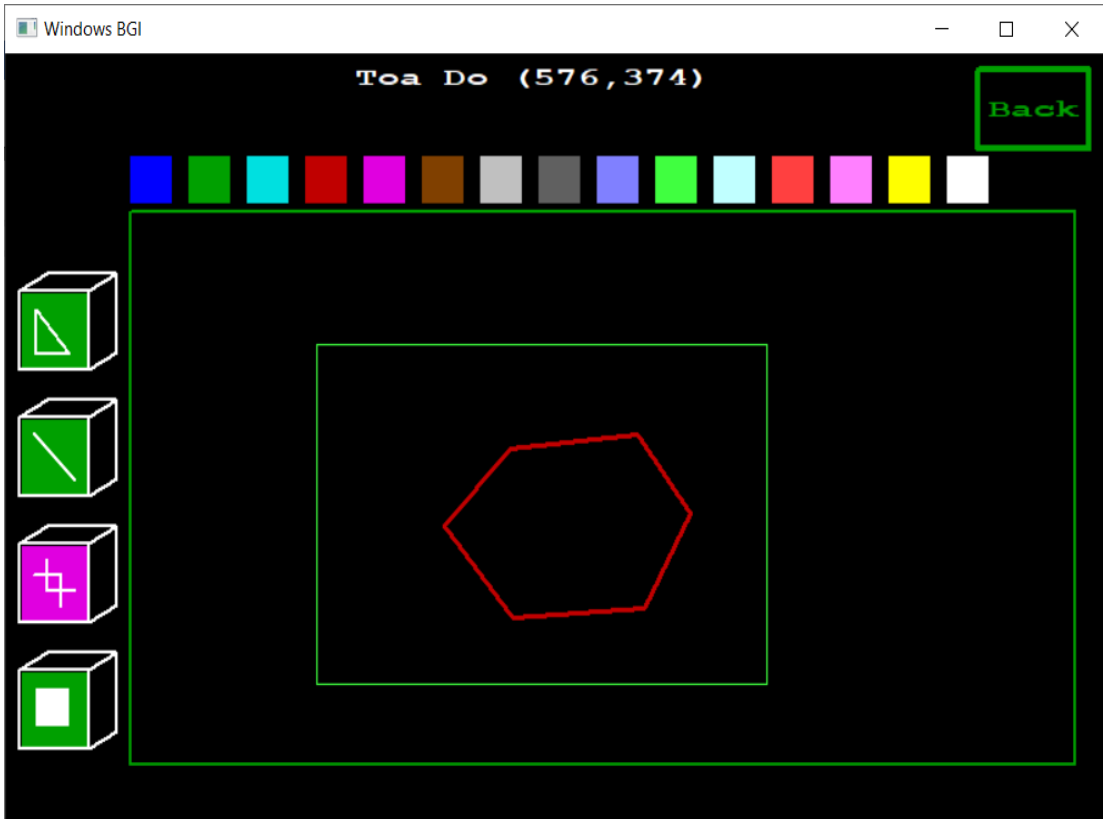
Sau khi vẽ cửa sổ xén thì các vùng đa nằm trong cửa sổ xén sẽ chuyển sang màu đỏ.

TH1: Đa giác nằm hoàn toàn ngoài



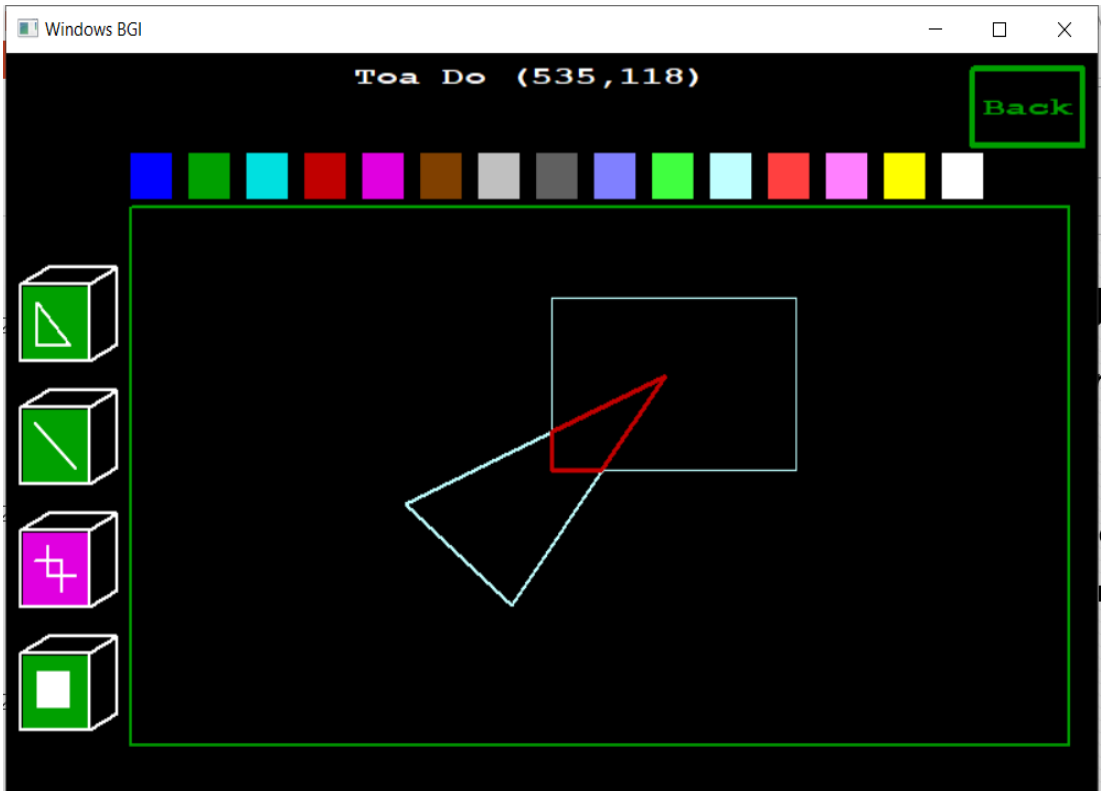
Hình 3.12. Minh hoạ về trường hợp đa giác nằm hoàn toàn ngoài cửa sổ xén

TH2: Đa giác nằm hoàn toàn trong



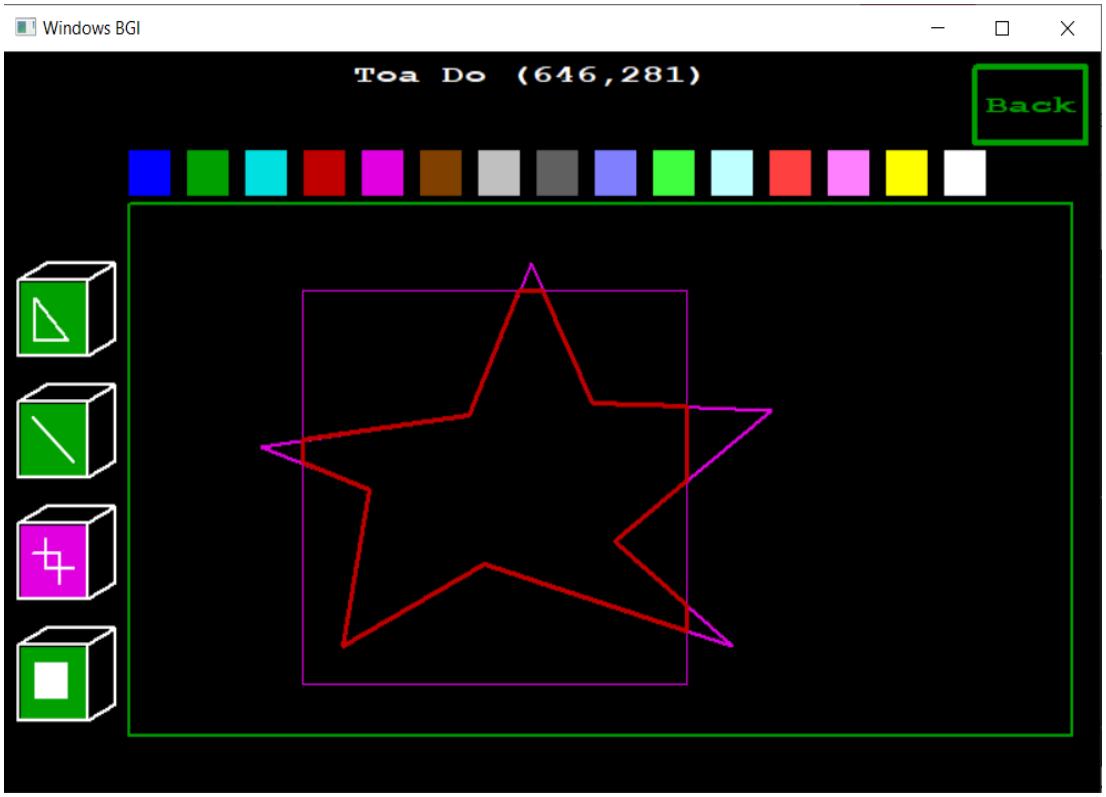
Hình 3.13. Minh hoạ về trường hợp đa giác nằm hoàn toàn trong cửa sổ xén

TH3: Một phần đa giác nằm ở ngoài cửa sổ xén



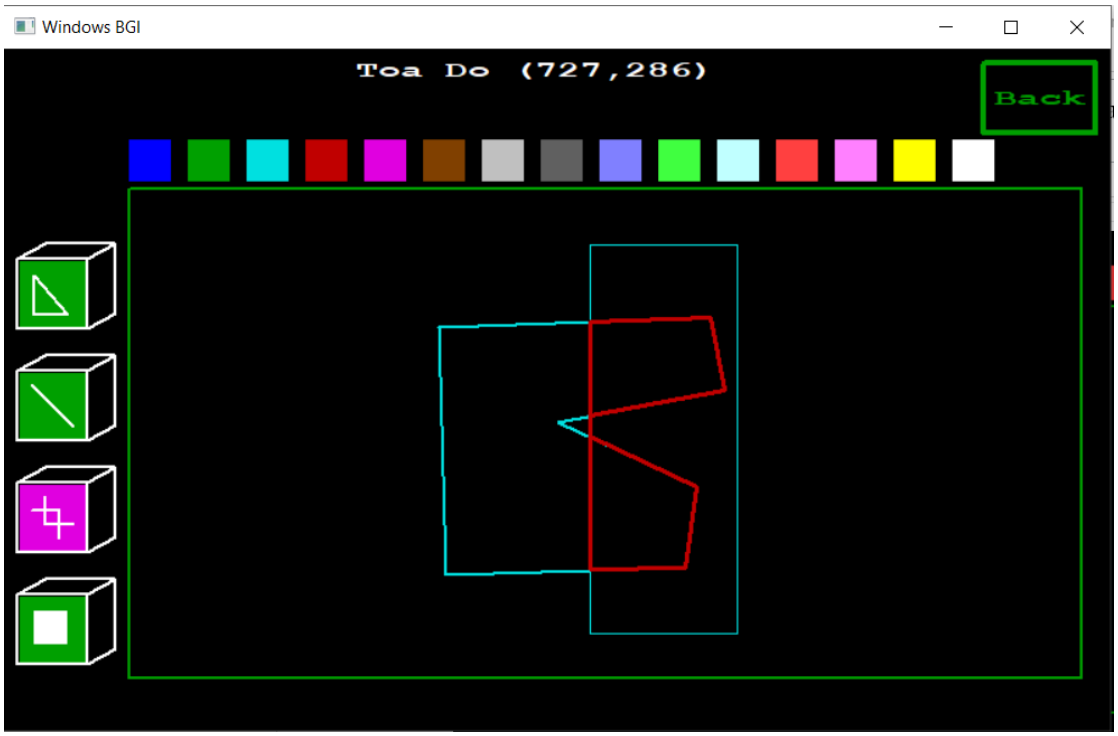
Hình 3.14. Minh hoạ về trường hợp một phần đa giác nằm ở ngoài

TH4: Xén một đa giác có đỉnh lõm ở bên trong cửa sổ xén



Hình 3.15. Minh hoạ về trường hợp đa giác có đỉnh lõm ở bên trong cửa sổ xén

TH5: Xén một đa giác có đỉnh lõm ở bên ngoài cửa sổ xén



Hình 3.16. Minh hoạ về trường hợp đa giác có đỉnh lõm ở bên ngoài cửa sổ xén

4. KẾT LUẬN

Thuật toán xén tỉa đa giác sử dụng thuật toán Sutherland-Hogman được cài đặt theo các yêu cầu của đề tài thực tập cơ sở. Việc nhập dữ liệu được thông qua bàn phím và sử dụng chuột trái, phải. Giao diện thiết kế sử dụng hoàn toàn bằng thư viện graphics.h. Các trường hợp xén tỉa đa giác đều được mô hình hoá và cài đặt, trong khuôn khổ báo cáo này, hạn chế của thuật toán Sutherland-Hogman đối với đa giác không lồi có đỉnh lõm nằm ngoài cửa sổ xén đã được chỉ ra nhưng chưa thể giải quyết trong đợt thực tập này.