TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———— * ————

Báo Cáo

Computational Geometry

Đề tài

Polygon Triangulation

GVHD: PGS Huỳnh Thị Thanh Bình

Nhóm 10

Trần Văn Thành - 20133561
Phan Xuân Đức - 20146212
Nguyễn Văn Thưởng - 20146699

Hà Nội, tháng 5 năm 2018

Mục Lục

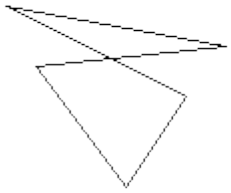
1. **Simple polygon** 3
2. **Art Gallery Problem** 3
3. **Polygon Triangulation** 9
   - 3.1. **Xác định hướng của đa giác** 9
   - 3.2. **Kiểm tra đỉnh lồi của đa giác** 11
   - 3.3. **Kiểm tra đa giác tự cắt** 12
   - 3.4. **Tam giác phân đa giác** 14
4. **Cài đặt chương trình** 17
5. **Tài liệu tham khảo** 19

1. Simple polygon

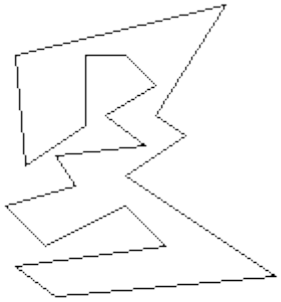
Đường cong đa giác là một chuỗi hữu hạn của các đoạn thẳng.

Các đoạn thẳng được gọi là các cạnh, 2 đầu mút gọi là đỉnh.

Một đa giác đơn giản – simple polygon là một đa giác khép kín và không tự cắt.



Not a Simple Polygon



Simple Polygon

Theo Định lý Jordan, một đa giác phân chia mặt phẳng thành 3 phần: bên trong, bên ngoài và đường biên.

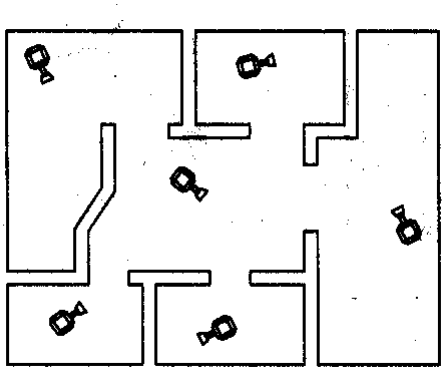
Ở phần này ta chỉ xét những đa giác không có hố (no holes) và không tự cắt (no self-intersections).

Hai điểm trong một đa giác có thể nhìn thấy nhau nếu đoạn nối chúng nằm bên trong hoặc trên đường biên của đa giác.

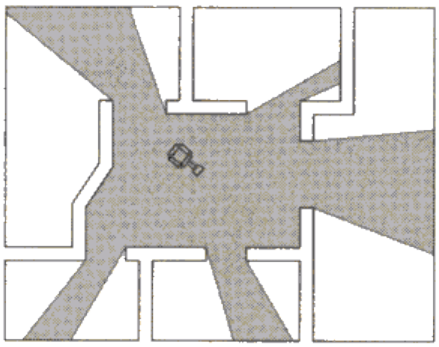
2. Art Gallery Problem

Art Gallery Problem-Bài toán phòng trưng bày nghệ thuật:

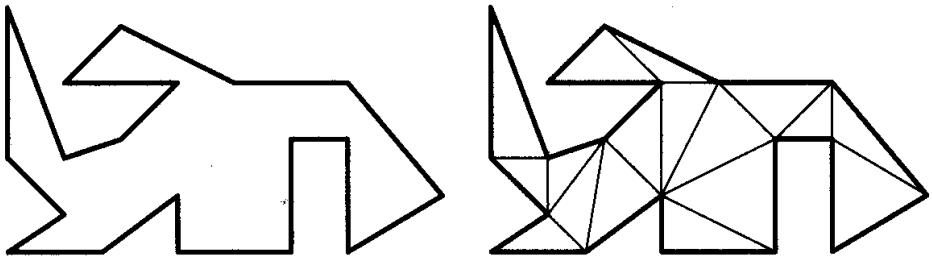
"Cần tối thiểu bao nhiêu máy ảnh để bảo vệ phòng trưng bày nghệ thuật, và cách quyết định nơi đặt chúng."



Polygons and visibility: khả năng nhìn thấy của camera



Gọi P là một đa giác đơn giản (simple polygon) với n đỉnh, số lượng máy ảnh phụ thuộc vào độ phức tạp của đa giác thể hiện theo số đỉnh n của đa giác. Với cùng n đỉnh nhưng 1 số đa giác thì rất khó, một số dễ (đa giác lồi có thể bảo vệ bởi chỉ 1 máy ảnh). Để thể hiện trường hợp xấu nhất, phân P thành những hình tam giác - Triangulation, và bảo vệ các hình tam giác, vì mọi hình tam giác luôn được bảo vệ chỉ bởi 1 camera.

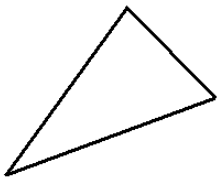


Triangulation - phép phân đa giác thành các tam giác (gọi là tam giác phân đa giác) bởi 1 tập tối đa các đường chéo không giao nhau giữa các cặp đỉnh. Đường chéo là đoạn thẳng nối 2 đỉnh của P và nằm bên trong P. Phép tam giác phân không phải là duy nhất.

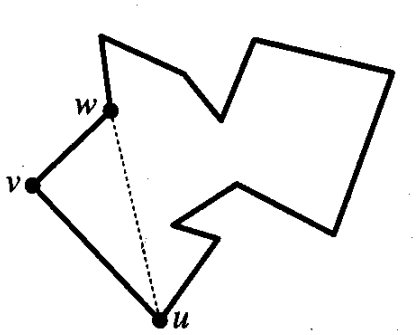
Mọi đa giác đơn giản đều cho phép một Triangulation, và bất kỳ Triangulation của một đa giác đơn giản với n đỉnh đều bao gồm đúng n-2 hình tam giác.

Chứng minh:

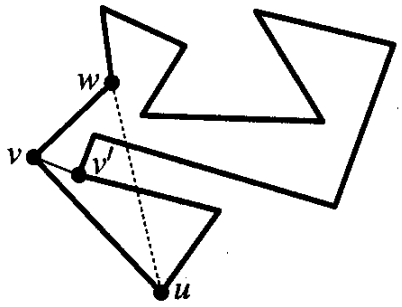
- Bước cơ sở: n = 3



  - 1 triangle (= n-2)
  - điều này luôn đúng
- Bước quy nạp: n > 3, giả định định lý đúng với ∀ m < n
  - Đầu tiên, chứng minh sự tồn tại của một đường chéo:
    - gọi *v* là đỉnh trái nhất của P
    - *u* và *w* là hai đỉnh lân cận của *v*
    - nếu đoạn *uw* nằm bên trong P, thì *uw* là một đường chéo



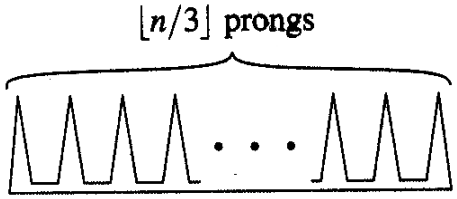
  - Nếu đoạn *uw* không nằm trong P:
    - có một hoặc nhiều đỉnh bên trong hình tam giác Δ *uvw*
    - gọi *v'* là đỉnh xa nhất từ *uw* và nằm trong tam giác Δ *uvw*
    - đoạn *vv'* không thể cắt bất kỳ cạnh nào của đa giác P, vì vậy *vv'* là một đường chéo



    - do đó, luôn tồn tại đường chéo của đa giác P
- đường chéo chia tách P thành các đa giác P_1 (m_1 đỉnh) và P_2 (m_2 đỉnh)
- cả $m_1$và m_2 phải nhỏ hơn n, do đó bằng phản chứng P_1và P_2 có thể được "triangulated"
- do đó, P có thể được "triangulated" - tam giác phân.
- bây giờ chứng minh rằng bất kỳ triangulation (phép tam giác phân) của P đều bao gồm n-2 hình tam giác:
    - $m_1 + m_2 = n + 2$ (P_1và P_2 chung hai đỉnh)
    - bằng quy nạp, mọi phép tam giác phân của P_i đều bao gồm m_i-2 tam giác
    - do đó, mọi phép tam giác phân của P đều bao gồm $(m_1 - 2) + (m_2 - 2) = n - 2$ tam giác
- Thời gian tuyến tính để tìm đường chéo, chia đa giác thành 2 đa giác con với đường chéo đó.
- Đệ quy trên 2 đa giác con.
- Thời tính trong TH xấu nhất là O(n^2).

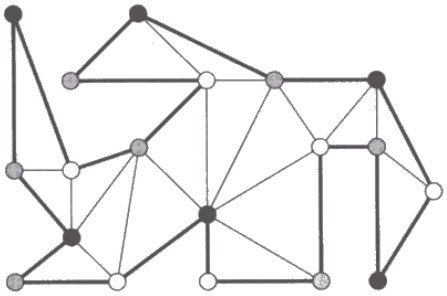
Chiến lược đặt camera:

- đặt 1 camera trên mỗi tam giác
    - n-2 cameras
- đặt camera trên đường chéo sẽ cho kết quả tốt hơn
    - n/2 cameras
- đặt camera ở đỉnh cho kết quả tốt nhất
    - $\lfloor n/3 \rfloor$ cameras
- kết quả sẽ tốt hơn $\lfloor n/3 \rfloor$?
    - Đó là khi đa giác là 1 đã giác lồi, khi đó chỉ cần 1 camera là đủ.

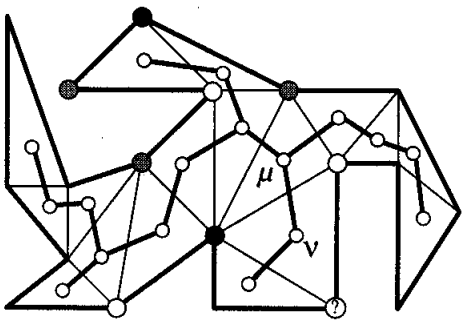


Đặt camera tại đỉnh:

- Chọn một tập con của tập đỉnh, sao cho bất kỳ tam giác nào đều chứa ít nhất 1 đỉnh đã chọn làm 3 màu của tứ giác. Sau đó chọn lớp màu nhỏ nhất để đặt camera.



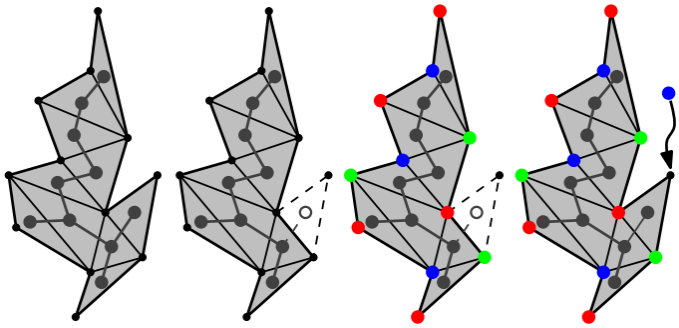
- 3 đỉnh của mọi tam giác luôn luôn được tô bởi 3 màu
  - Gọi T_P là tam giác phân của đa giác P
  - Đồ thị kép G(T_P) của T_P là một cây (mỗi node là một tam giác trong T_P, và các cạnh là đường chéo trong T_P, hai node được nối nếu 2 tam giác là chung cạnh)



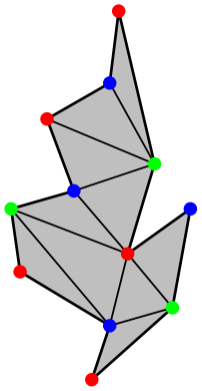
  - Tìm 3 màu bằng cách duyệt đồ thị G(T_P), tìm kiếm theo chiều sâu

- Bắt đầu từ bất kỳ nút nào trong G(T_P) và tô màu các đỉnh của tam giác tương ứng trong 3 màu khác nhau
- Khi đến 1 nút mới, về cơ bản chúng ta đang đi qua một đường chéo
- 2 đỉnh trên đường chéo đó được tô màu, vì thế chỉ cần tô màu đỉnh cuối của tam giác mới được thăm, và khi đó chỉ còn lại 1 màu

- TH cơ sở: có đúng 1 tam giác, 3 đỉnh tam giác luôn được tô bởi 3 màu.
- TH tổng quát có n – 2 tam giác (đa giác n đỉnh). Giả sử đã duyệt (tô màu) qua n – 3 tam giác, lúc này chỉ còn 1 tam giác chưa được tô màu, ta gỡ bỏ tam giác này khỏi đa giác ban đầu. Tô màu 3 đỉnh của tam giác đó, sao cho khi thêm tam giác đó trở lại đa giác ban đầu thì đỉnh có màu sắc khác với hàng xóm của nó, điều này luôn thực hiện được.



Với bộ 3 màu, không mất tính tổng quát giả sử số lượng các đỉnh màu đỏ là r, xanh lá cây là g và xanh lam là b sao cho r ≤ g ≤ b. Tổng số các đỉnh là r+g+b = n, nếu r > $\left\lfloor \frac{n}{3} \right\rfloor$ thì g và b cũng lớn hơn $\left\lfloor \frac{n}{3} \right\rfloor$ và khi đó r+g+b > n mâu thuẫn. Do đó r ≤ $\left\lfloor \frac{n}{3} \right\rfloor$.

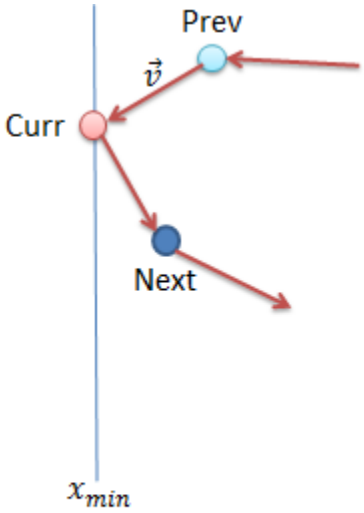


Khi đa giác có chứa hố (holes) thì điều này không đúng vì khi đó đồ thị G(T_P) sẽ chứa chu trình và nó không phải là 1 cây.

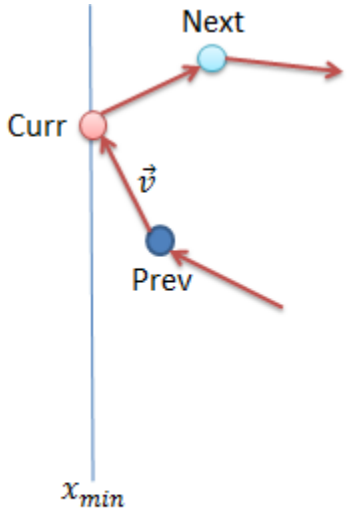
3. Polygon Triangulation

3.1. Xác định hướng của đa giác

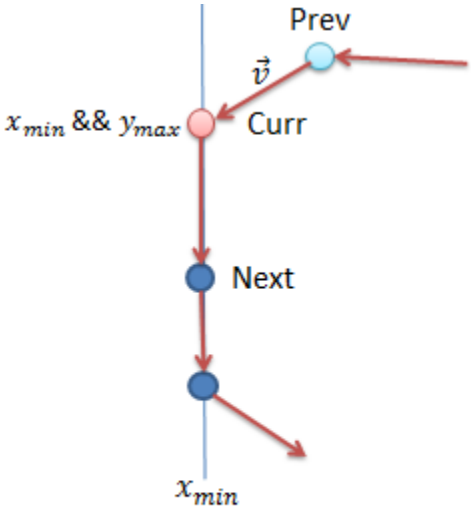
Đa giác được vẽ theo chiều kim đồng hồ hay ngược chiều kim đồng hồ? Đầu tiên ta đi tìm đỉnh có tọa độ x nhỏ nhất, nếu có nhiều hơn 1 đỉnh có cùng tọa độ x nhỏ nhất, thì ta chọn đỉnh có tọa độ y lớn nhất. Sau đó kiểm tra 2 đỉnh lân cận của đỉnh hiện tại xem là nó rẽ trái hay rẽ phải.



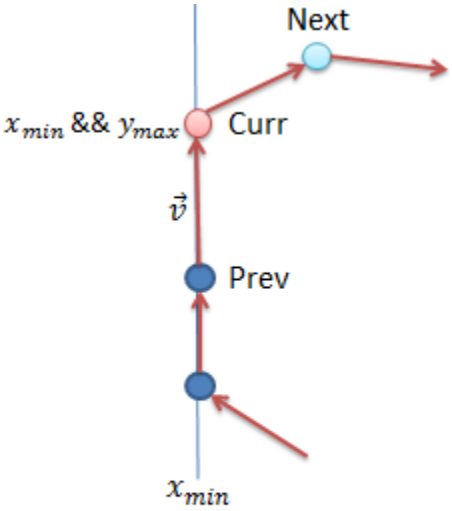
UnClockwise



Clockwise



UnClockwise



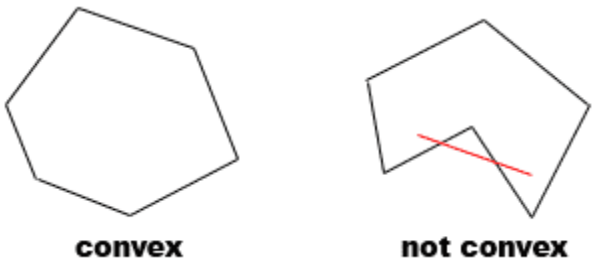
Clockwise

```
# pseudocode:
Void  getOrientation ()
Begin
        Index = 0
        Curr = Vertice(0)
        FOR(i = 1 : n-1)
                IF Vertice(i).x  < Curr.x then
                        Curr = Vertice(i)
                        Index = i
                ELSE IF Vertice(i).x == Curr.x && Vertice(i).y > Curr.y then
                        Curr = Vertice(i)
                        Index = i
        End FOR
        IF Index = 0
                Prev = Vertice(n-1)
        ELSE
                Prev = Vertice(Index-1)
        V = Vector(Curr.x – Prev.x, Curr.y – Prev.y )
        IF Index = n-1
                Next = Vertice(0)
        ELSE
                Next = Vertice(Index+1)
        Orientation = Next.x * V.y – Next.y * V.x + V.x * Prev.y – V.y * Prev.x
        IF Orientation ≤ 0 then Clockwise
        ELSE then UnClockwise
End
```

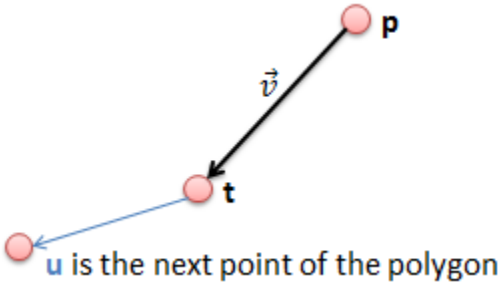
3.2. Kiểm tra đỉnh lồi của đa giác

Tập lồi: Một tập P là lồi nếu mọi cặp điểm p_1, p_2 trong tập hợp này, đoạn thẳng nối chúng cũng nằm hoàn toàn trong tập hợp này.

Đa giác lồi: Một đa giác đơn giản P (simple polygon) là lồi (convex) nếu nội thất của nó là một tập lồi.



Để kiểm tra đỉnh lồi của 1 đa giác, nếu ta di chuyển dọc theo các cạnh của đa giác và ta chỉ cần di chuyển sang trái hoặc sang phải thì nó sẽ là đỉnh lồi. Với cách này thì không quan tâm ta di chuyển theo chiều kim đồng hồ hay ngược chiều kim đồng hồ. Nhưng làm thế nào để biết đỉnh tiếp theo là phải hay trái từ cạnh đa giác hiện tại. Hình dưới đây, p và t là 2 đỉnh liên tiếp của đa giác, u là đỉnh tiếp theo. Biểu diễn tọa độ điểm u (là vector $\vec{u}$) theo tọa độ điểm p (là vector $\vec{p}$) và vector $\vec{v}$, với $\vec{v}$ = $\vec{t}$ - $\vec{p}$ = $\overrightarrow{pt}$ thỏa mãn biểu thức:



g: $\vec{u}$ = $\vec{p}$ + r * $\vec{v}$, r ∈ ℝ với $\vec{v}$ = $\vec{t}$ - $\vec{p}$ = $\overrightarrow{pt}$

$$u_x = p_x + r * v_x$$

$$u_y = p_y + r * v_y$$

$$\Rightarrow r = \frac{u_x - p_x}{v_x}, r = \frac{u_y - p_y}{v_y}$$

$$\Rightarrow (u_x - p_x) * v_y = (u_y - p_y) * v_x$$

$$\Rightarrow f(u) = u_x * v_y - u_y * v_x + v_x * p_y - p_x * v_y$$

Xét vị trí tương đối của điểm u đến đường thẳng g đi qua 2 đỉnh p và t, g có các tính chất sau: nếu f(u) = 0 thì đỉnh u nằm trên đường thẳng g; nếu f(u)> 0 thì đỉnh u nằm bên phải từ đường thẳng g và f (u) ≤ 0 thì đỉnh u nằm bên trái đường g. "Trái" và "Phải" phụ thuộc vào hướng vectơ $\vec{v}$, do đó nó phụ thuộc vào việc đi dọc đa giác theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ.

pseudocode:

BOOL IsConvex (Vertice t)

Begin

p = PREV of t

u = NEXT of t

f(u) = u.x * v.y - u.y * v.x + v.x * p.y - v.y * p.x;

IF [f(u) > 0 && Clockwise] || [f(u) ≤ 0 && UnClockwise]

RETURN true

ELSE

RETURN false

End

3.3. Kiểm tra đa giác tự cắt

Đa giác là đa giác tự cắt (self-intersections) nếu tồn tại 2 cạnh không kề nhau mà chúng cắt nhau.

pseudocode:

Kiểm tra 2 cạnh cắt nhau

BOOL isIntersect (P_1, P_2, P_3, P_4)

Begin

V_{12} = Vector(P_2.x - P_1.x, P_2.y - P_1.y)

V_{34} = Vector(P_4.x - P_3.x, P_4.y - P_3.y)

Det = V_{12}.x * V_{34}.y - V_{12}.y * V_{34}.x

IF Det != 0

V_{13} = Vector(P_3.x - P_1.x, P_3.y - P_1.y)

Lambda = (V_{13}.x * V_{34}.y - V_{13}.y * V_{34}.x) / Det

Alpha = (V_{13}.x * V_{12}.y - V_{13}.y * V_{12}.x) / Det

RETURN ((0 < Lambda && Lambda < 1) && (0 < Alpha && Alpha < 1))

Else

RETURN false

End

Kiểm tra đa giác tự cắt

BOOL checkIntersect ()

Begin

FOR(i = 0 : n-2)

FOR(j = 0 : n-2)

IF i==j continue

IF isIntersect(P_i, P_{i+1}, P_j, P_{j+1})

RETURN true

End FOR

End FOR

RETURN false

End

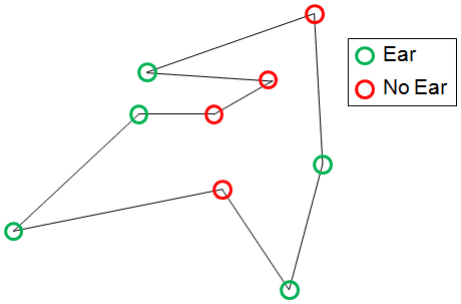
3.4. Tam giác phân đa giác

Thực hiện thuật toán Kong' Algorithm để chia một đa giác đơn giản thành các tam giác (gọi là triangulation). Một đa giác đơn giản (simple polygon) là một đa giác không có hố và không tự cắt. Do đó các cạnh của nó không được chạm hoặc đè lên nhau. Nhưng nó có thể là đa giác không lồi. Thuật toán Kong đơn giản vì nó đi dọc theo tất cả các đỉnh của đa giác và cắt các đỉnh lồi (do đó tạo ra một hình tam giác) đến khi chỉ còn lại 3 đỉnh thì dừng lại.

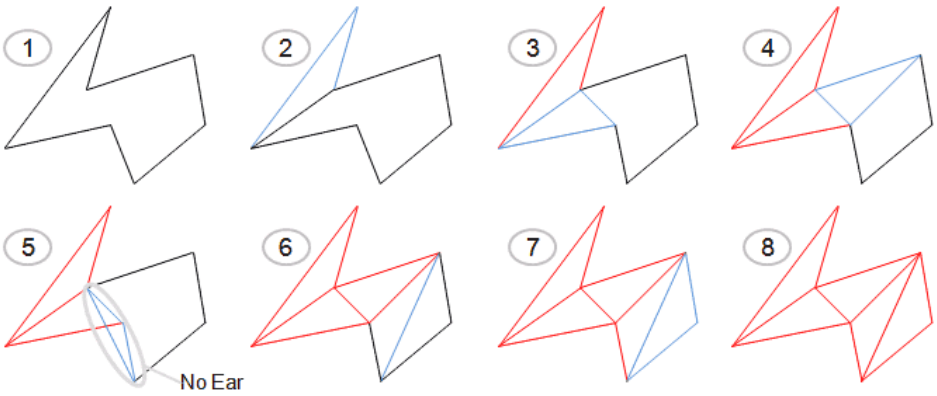
Cho: Đa giác đơn giản P_1, P_2, P_3, …P_n với n đỉnh.

Mục tiêu: Một tập các hình tam giác (P_i, P_j, P_k) với i, j, k ∈ {1,..,n} tạo thành một tam giác hoàn chỉnh, không chồng chéo của đa giác.

- Định nghĩa "tai" – Ear: là một đỉnh lồi của đa giác (P_i, P_{i+1}, P_{i+2}), có nghĩa là đoạn (P_i, P_{i+2}) nằm hoàn toàn trong đa giác và không được chạm hoặc cắt bởi bất kỳ cạnh nào khác của đa giác.



- Thuật toán đi dọc theo tất cả các cạnh và kiểm tra nếu 3 điểm lân cận (P_i, P_{i+1}, P_{i+2}) tạo thành 1 "tai".
- Nếu đỉnh P_{i+1} là "tai": xây dựng tam giác (P_i, P_{i+1}, P_{i+2}) và xóa đỉnh giác P_{i+1} khỏi đa giác, quay lại 2 đỉnh còn lại và thực hiện tiếp thuật toán.
- Nếu đỉnh P_{i+1} không là "tai": thực hiện kiểm tra đỉnh tiếp theo tại đỉnh P_{i+2} với 3 đỉnh lân cận (P_{i+1}, P_{i+2}, P_{i+3})
- Thuật toán dừng lại khi chỉ còn 3 đỉnh cuối, những đỉnh đó sẽ tạo thành 1 tam giác cuối cùng.
- Thuật toán dựa trên thực tế là mỗi đa giác luôn chứa ít nhất 2 "tai" (không chứng minh ở đây).



Hình trên cho thấy một ví dụ về thuật toán Kong. Hình tam giác màu xanh là hình tam giác đang xét. Hình tam giác màu đỏ là những hình tam giác đã bị cắt và không thuộc về đa giác hiện tại nữa. Ở bước 5, ba đỉnh không tạo thành 1 "tai" nên thuật toán thực hiện với 3 đỉnh tiếp theo.

Trong việc phát hiện đỉnh là "tai", mỗi đỉnh của đa giác được kiểm tra độ lồi – do đó hướng (trong khi người dùng vẽ các đỉnh liên tiếp của đa giác) cần được biết - đó là nếu người dùng vẽ đa giác theo chiều kim đồng hồ hay ngược chiều kim đồng hồ. Điều này được thực hiện bằng cách xác định đỉnh có tọa độ *x* nhỏ nhất (nếu có 1 số đỉnh có cùng tọa độ *x* nhỏ nhất, chọn một tọa độ có tọa độ *y* lớn nhất) và kiểm tra cả 2 cạnh chung đỉnh này là rẽ phải hay rẽ trái. Như vậy là đủ để phát hiện đỉnh đó có là "tai".

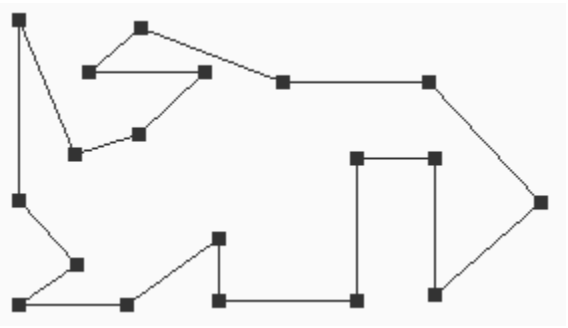
pseudocode:

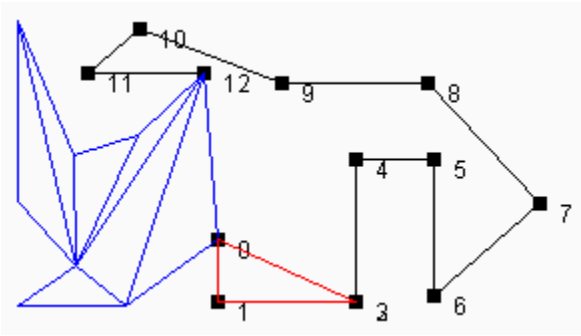
```
// Precondition: R contains all non-convex points of polygon

BOOL isEar (Vertice P.prev, Vertice P, Vertice P.next)

Begin

    IF R is empty

        RETURN false

    ELSE

        IF x is convex

        then

            IF InsideAreaOfTriangle(P.prev, P, P.next) contains no point of R

                RETURN true

            ELSE
```

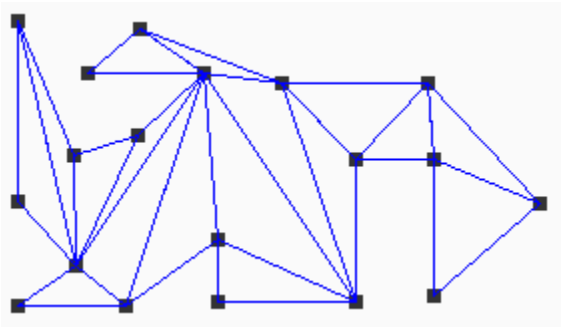
```
                    RETURN false
            ELSE
                    RETURN false
End

# Preconditions:
// Q contains all points/vertices of the polygon
Void DoKong ()
Begin
    WHILE |Q| > 3
        IF isEar(P.prev, P, P.next) then
                addTriangle(P.prev, P, P.next)      // e.g. to a list
                remove P from Q
                P = P.prev
        ELSE
            P = P.next
End
```

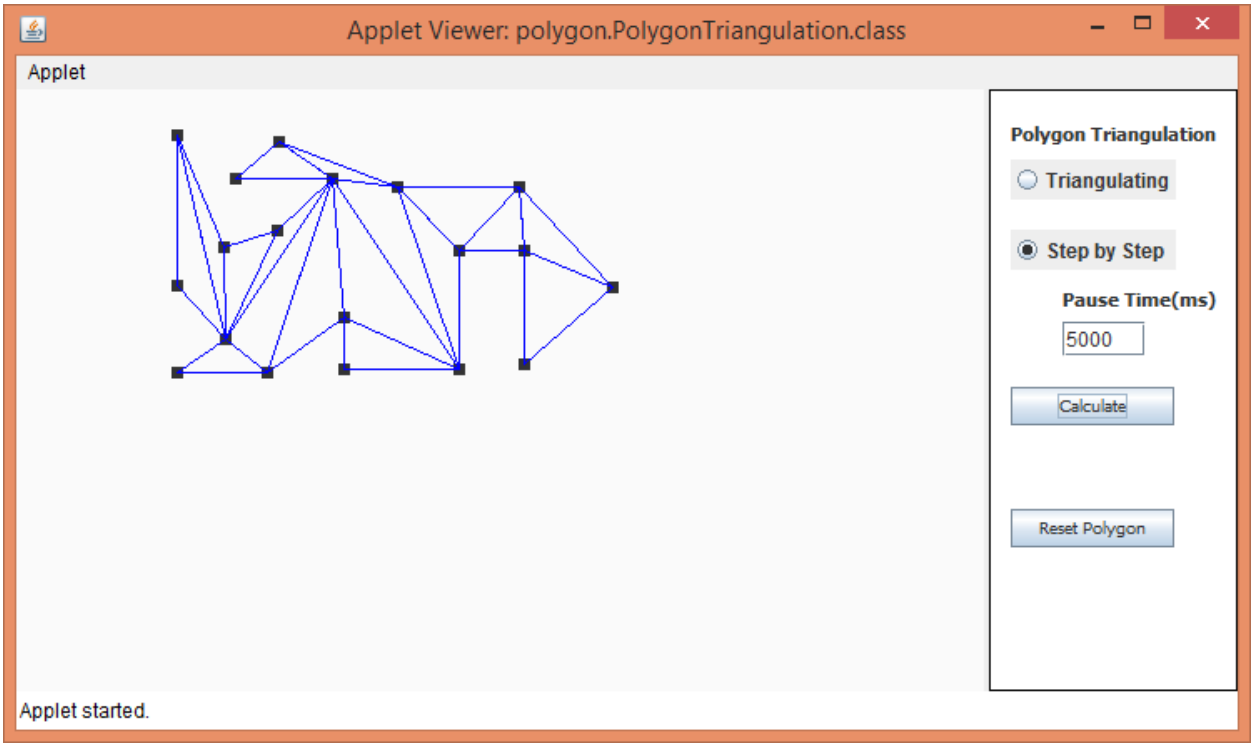
Thời gian tính của thuật toán Kong’ Algorithm là O(kn).

4. Cài đặt chương trình









Link source code chương trình : https://github.com/thanhtv2/Polygon-Triangulation

5. Tài liệu tham khảo

Berg, Mark de; Cheong, Otfried; Kreveld, Marc van; Overmars, Mark;. (2008). *Computational Geometry Algorithms and Applications* (Third ed.). Berlin Heidelberg, German: Springer.

Bình, H. T. (2018). *Hình học tính toán.* Hà Nội: ĐHBKHN.

Molkenthin, M. (n.d.). *Polygon*. Retrieved from http://www.sunshine2k.de/coding/java/Polygon/Kong/Kong.html