
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN
MÔN: **ĐỒ HỌA MÁY TÍNH**

# BÁO CÁO ĐỒ ÁN

# Thuật toán tô màu

**GVTH: Thầy Tô Hoài Việt**
**Sinh viên: Đoàn Minh Hiếu_1612198**

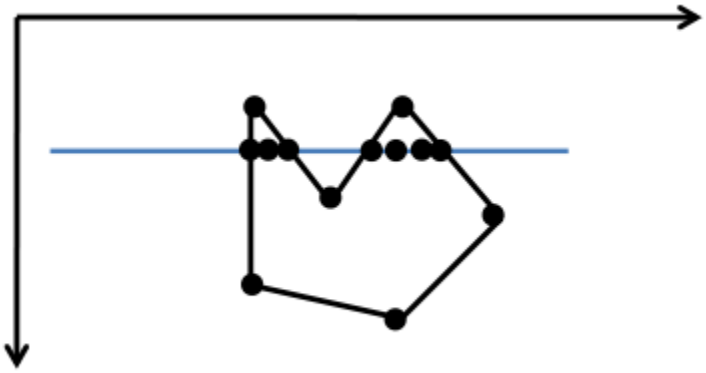
Tp Hồ Chí Minh ngày 14 tháng 12 năm 2018

# Contents

I. Thuật toán Scanline ..... 1
1. Ý tưởng ..... 1
2. Thuật toán ..... 1
3. Cài đặt ..... 1
4. Kết quả ..... 3
II. Thuật toán Boundary Fill ..... 4
1. Ý tưởng ..... 4
2. Thuật toán ..... 4
3. Cài đặt ..... 5
4. Cải tiến thuật toán Boundary Fill: ..... 5
5. Kết quả ..... 6
III. So sánh kết quả giữa các thuật toán ..... 8
IV. Tài liệu tham khảo ..... 8

# I. Thuật toán Scanline

## 1. Ý tưởng



Ý tưởng cơ bản của thuật toán Scanline là ta cho một đường thẳng (line) chạy ngang qua hình cần tô, nếu cắt cạnh của hình lần thứ nhất (lẻ) ta sẽ bắt đầu tô đến khi cắt 1 cạnh khác của hình lần thứ 2 (chẵn) thì dừng tô. Cứ như vậy cho scan line quét hết hình (từ ymin đến ymax).

## 2. Thuật toán

1. Ta sẽ thu thập một tập các cạnh cấu tạo nên hình (trừ các cạnh nằm ngang song song với Ox.
2. Từ tập cạnh đó ta xác định ymin, ymax của các đỉnh để cho scanline chạy từ ymin đến ymax
3. Với mỗi scanline, tìm giao điểm của nó với các cạnh
4. Sắp xếp các giao điểm đó theo thứ tự tăng của hoành độ x
5. Fill tất cả các pixel giữa từng cặp giao điểm đã tìm ở bước trên. Đó chính là phần bên trong cần tô của hình

## 3. Cài đặt

Trước khi đi đến các bước cụ thể, ta thông qua một số tên gọi sử dụng:

**Edge Buckets:** Là một cấu trúc dữ liệu chứa các thông tin cần thiết của một cạnh, Edge Buckets của mỗi người sẽ khác nhau, tùy vào cach cài đặt, ở đây mình sử dụng:

| yMax | yMin | x | sign | dX | dY |
|---|---|---|---|---|---|

Trong đó:

- yMax: Tung độ Y lớn nhất của edge
- yMin: Tung độ Y nhỏ nhất của edge
- x: Vị trí hiện tại của edge cắt scanline đang quét, khởi tạo bằng yMin
- sign: dấu xác định chiều của cạnh (-1 hoặc 1)
- dX: Khoảng cách (lấy trị tuyệt đối) giữa hoành độ X 2 đỉnh của cạnh
- dY: Khoảng cách (lấy trị tuyệt đối) giữa tung độ Y 2 đỉnh của cạnh

**Edge Table**: Danh sách tất cả các cạnh (trừ cạnh nằm ngang song song Ox) tạo nên hình

- Danh sách cần được sắp xếp tăng theo yMin
- Edges được remove khỏi Edge Table khi Active List đã xử lý xong nó
- Thuật toán tô màu hoàn tất khi tất cả các cạnh trong Edge Table được remove

Active List: Chứa các edges đang được xử lý để fill vào hình vẽ

- Edge được đưa vào Active List khi yMin của nó bằng với tung độ Y hiện hành của scanline
- Số lượng edges trong Active List luôn chẵn vì hình kín nên nó phải chẵn
- Active List luôn duy trì trật tự tăng theo thuộc tính x trong các cạnh
- Active List luôn resort sau mỗi lượt.

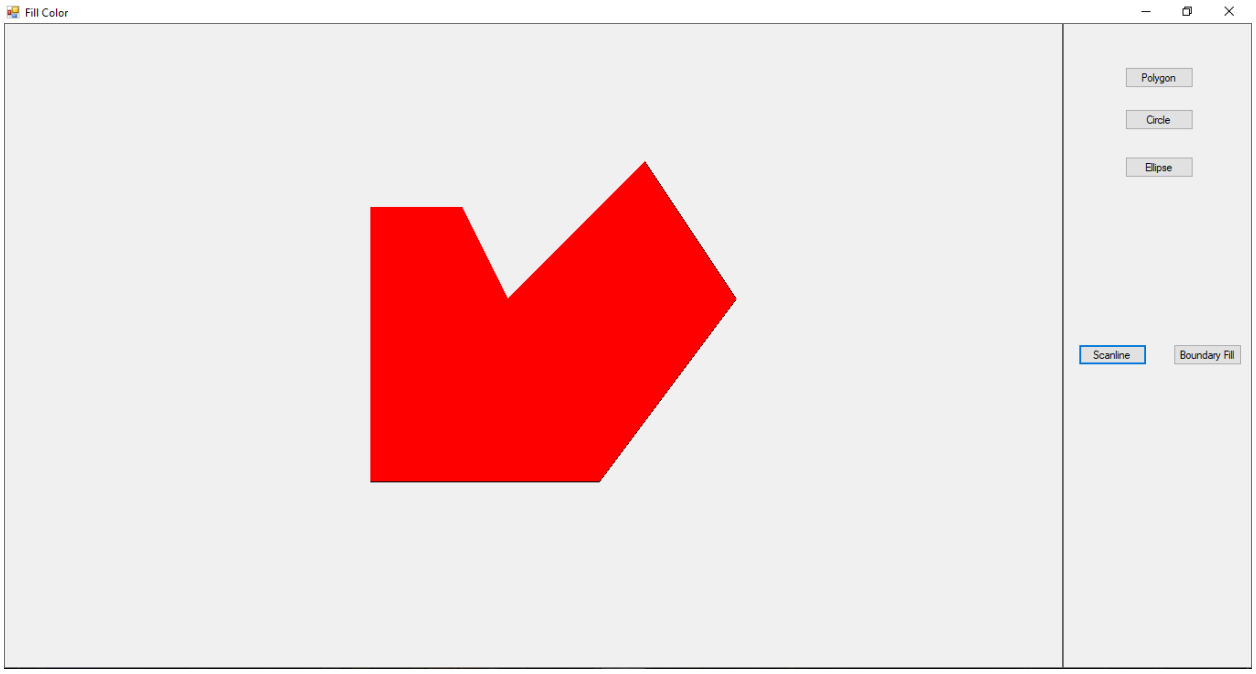
## Các bước cụ thể:

1. Tạo ra Edge Table:
   Duyệt qua tất cả các đỉnh, với mỗi cặp đỉnh tạo Edge Bucket cho cạnh đó rồi Add vào Edge Table
2. Sắp xếp Edge Table theo thứ tự tăng của yMin
3. Xử lý Edge Table
   a. Bắt đầu với scanline có y = yMin của egde đầu tiên trong Edge Table
   b. Trong khi Edge Table vẫn còn edge:
      i. Kiểm tra tất cả edges trong Active List cần được remove (yMax==yScanline)
         Cần remove edge ở cả Active List và Edge Table
      ii. Nếu edges nào có yMin = yScanline thì Add nó vô Active List
      iii. Sắp xếp các cạnh trong Active List bởi x
      iv. Fill tất cả các điểm trên Scanline có hoành độ nằm giữa 2 X của cặp cạnh trong Active List
      v. Tăng Scanline lên
      vi. Tăng thuộc tính X trong tất cả các cạnh trong Active List

Tăng tốc cho Scanline: Để tránh xử lý trên số thực, ta có thể cập nhật thuộc tính X trong mỗi edges ở Active List bằng cách cộng thêm vào **(sign)*dy/dx**, bởi vì:

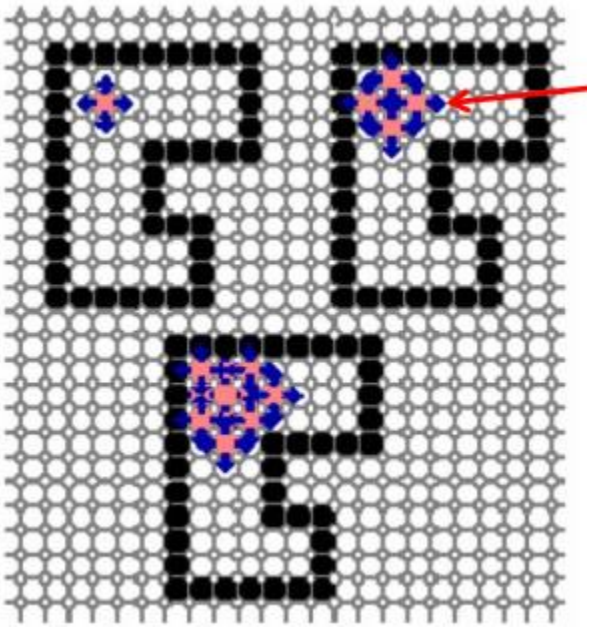
$$x_{k+1} - x_k = \frac{1}{m}((k+1) - k) = \frac{1}{m} \text{ hay } x_{k+1} = x_k + \frac{1}{m}.$$

## 4. Kết quả



# II. Thuật toán Boundary Fill

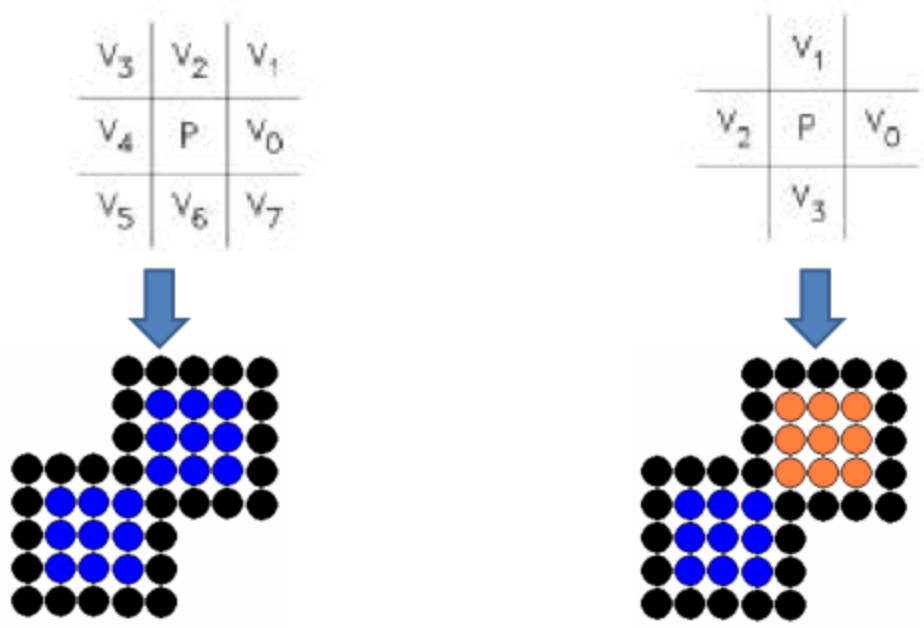
## 1. Ý tưởng



Từ 1 điểm bên trong hình (khác màu với đường viền và khác màu với màu cần tô) ta tô nó và lan truyền ra các điểm xung quanh (4 hoặc 8 điểm).

## 2. Thuật toán

1. Bắt đầu tại một điểm bên trong hình vẽ
2. Nếu nó khác màu với đường viền và khác màu với màu cần tô thì tô nó và lan truyền sang các điểm xung quanh. (4 hoặc 8 điểm).




3. Nếu nó trùng màu với đường viền hoặc màu cần tô thì ta bỏ qua.

## 3. Cài đặt

Thuật toán Boundary Fill Color cài đặt trong C#:

```
public void boundaryFill(ref Bitmap bmp, int x, int y, Color boundaryColor, Color fillColor)
{
    Color currentColor = bmp.GetPixel(x, y);

    if (!currentColor.ToArgb().Equals(boundaryColor.ToArgb()) && !currentColor.ToArgb().Equals(fillColor.ToArgb()))
    {
        bmp.SetPixel(x, y, fillColor);
        boundaryFill(ref bmp, x - 1, y, boundaryColor, fillColor);
        boundaryFill(ref bmp, x, y + 1, boundaryColor, fillColor);
        boundaryFill(ref bmp, x + 1, y, boundaryColor, fillColor);
        boundaryFill(ref bmp, x, y - 1, boundaryColor, fillColor);
    }

}
```

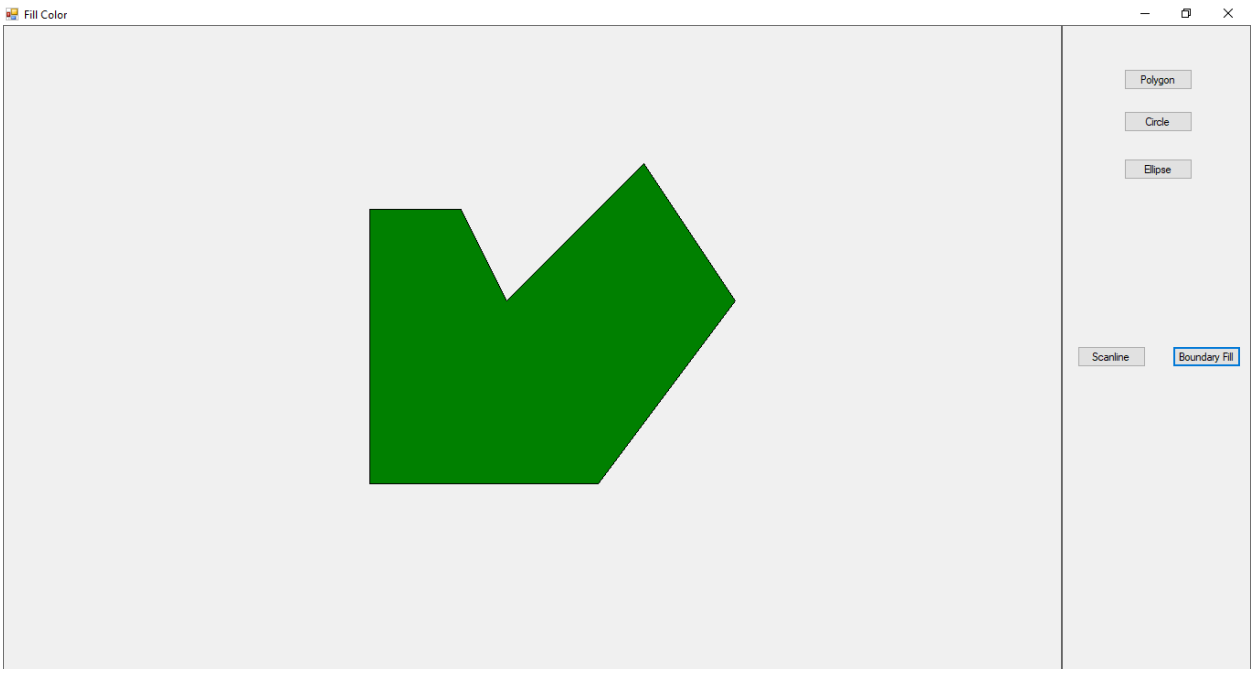
## 4. Cải tiến thuật toán Boundary Fill

Ta thấy cách cài đặt của Boundary Fill là sử dụng đệ quy, điều này sẽ dẫn đến tình trạng Stackoverflow. Để giải quyết vấn đề này, ta đi đến khử đệ quy cho Boundary Fill bằng cách sử dụng cấu trúc dữ liệu Stack như sau:

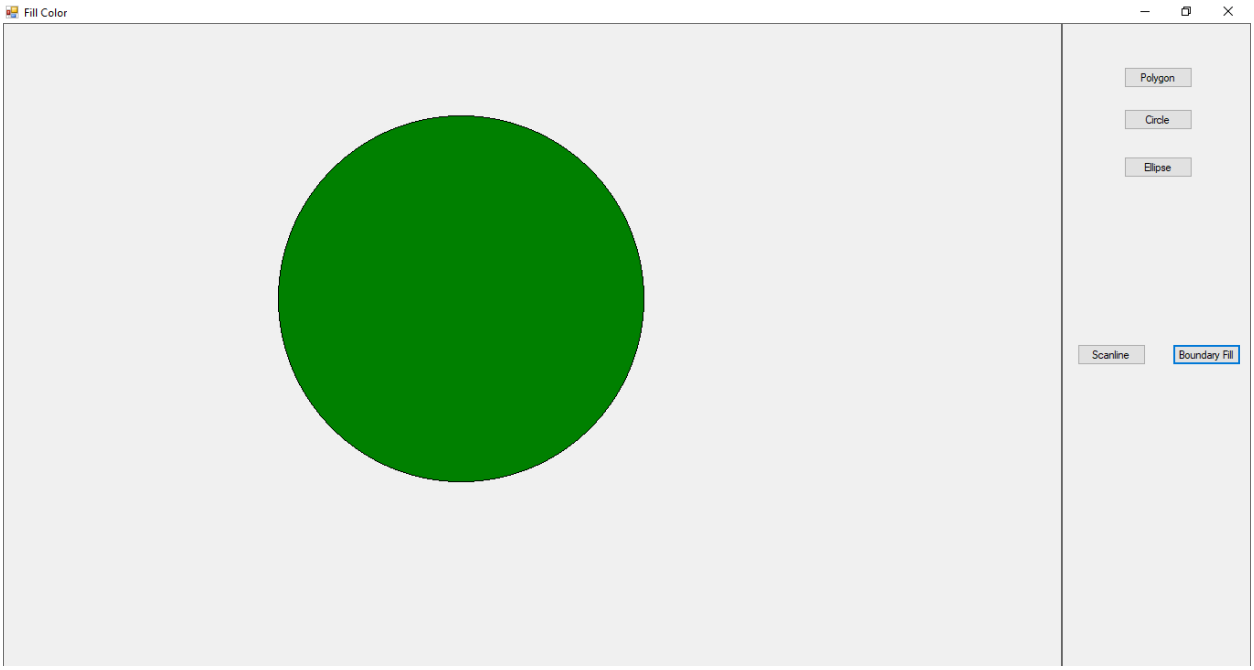
```
public void boundaryFill2(ref Bitmap bmp, int x, int y, Color boundaryColor, Color fillColor)
{
    Stack<Point> points = new Stack<Point>();
    points.Push(new Point(x, y));

    while (points.Count > 0)
    {
        Point currentPoint = points.Pop();
        int xx = currentPoint.X;
        int yy = currentPoint.Y;

        Color currentColor = bmp.GetPixel(xx, yy);
        if (!currentColor.ToArgb().Equals(boundaryColor.ToArgb()) && !currentColor.ToArgb().Equals(fillColor.ToArgb()))
        {
            bmp.SetPixel(xx, yy, fillColor);
            points.Push(new Point(xx + 1, yy));
            points.Push(new Point(xx - 1, yy));
            points.Push(new Point(xx, yy + 1));
            points.Push(new Point(xx, yy - 1));
        }
    }
}
```

## 5. Kết quả

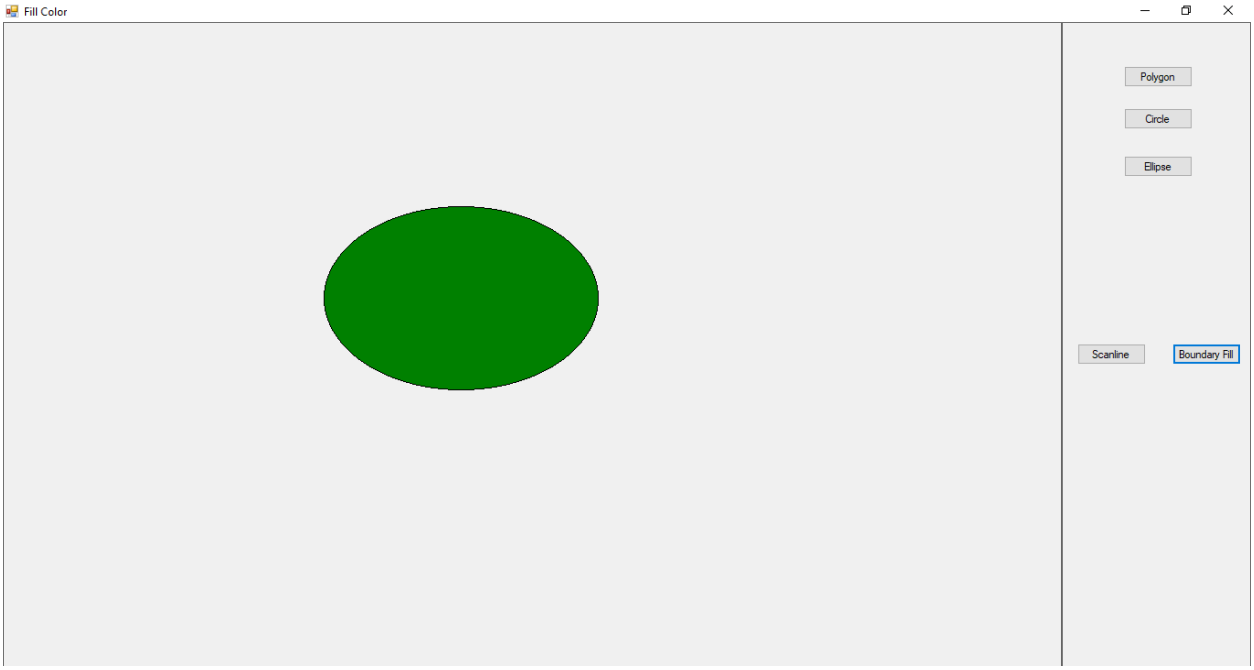
1. Boundary Fill cho Polygon



2. Boundary Fill cho Circle




3. Boundary Fill cho Ellipse



## III. So sánh kết quả giữa các thuật toán

- Scanline nhanh hơn Boundary Fill xấp xỉ 6 lần
- Boundary Fill khử đệ quy không gây ra hiện tượng Stackoverflow như Boundary đệ quy

## IV. Tài liệu tham khảo

1. Slide bài giảng bộ môn Đồ họa máy tính, khoa CNTT, ĐH KHTN Tp HCM
2. Computer Graphics: Scan Line Polygon Fill Algorithm