**COMPUTER GRAPHICS AND ANIMATION**

**Programming Assignment 13**

**Region Fill**

**Report**



Cefriandy Z H Simarmata (001201500007)

Marfin Fadillah (001201600058)

T. Ikhsan Ansyari Husny (001201500037)

**Computing Information and Technology**

**PRESIDENT UNIVERSITY**

**Jl. Ki Hajar Dewantara, Kota Jababeka, Cikarang Baru, Bekasi 17550 – Indonesia**

**Telp. (021) 89 1097 6263, Fax. (021) 8910 9768**

[**www.president.ac.id**](http://www.president.ac.id)

**2017**

Contents

[**I.** **Introduction** 3](#_Toc497703282)

[**II.** **Basic Theory** 4](#_Toc497703283)

[**III.** **Implementation** 8](#_Toc497703284)

[**III.1 Explanation Main Interface** 8](#_Toc497703285)

[**III.2 Explanation The Features** 11](#_Toc497703286)

[**IV.** **Design** 12](#_Toc497703287)

[**V.** **Evaluation** 14](#_Toc497703288)

[**VI.** **Work Log** 23](#_Toc497703289)

[**V.** **Conclusion and Remarks** 24](#_Toc497703290)

# **Introduction**

**I.1** **Program About**

To create an aplication that allows the user to do region fills. Where the user can draw the line or circle, change the colour, clear the screen, save/load the contents of the screen to/from a file. the region fill consits of two types which are flood fill and boundary fill. There are 8 types of filling the colour. Which are flood fill, flood fill with stack, scanline region fill, scan line region fill with stack, boundary fill, boundary fill with stack, scanline boundary fill, scanline boundary fill with stack. The program can do filling the line with checkered pattern and striped pattern, it can do the show order and also it can do numeric up down.

**I.2 Program Language**

The program language is visual basic in visual studio 2017.

# **Basic Theory**

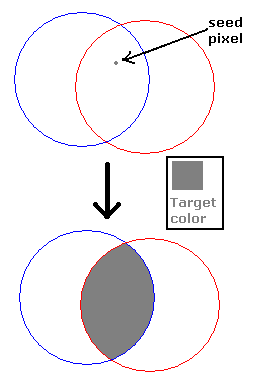
**II.1 Region Fill**

Region fill is also called area fill or seed fill. It fills a region of pixels Fill a region of pixels with a certain colour. Start from a pixel (seed pixel) and change the colour of the pixels in the same region. We define ‘region’ using pixel to pixel and scan line. There are two types of region fill which are flood fill and boundary fill.

**II.2 Flood Fill and Boundary Fill**

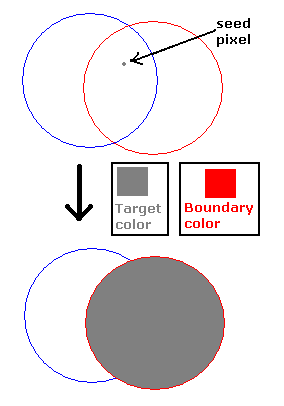
* Flood fill

It fills all pixels having the same color as the seed pixel in the same region with target color.

Example :

* Boundary fill

It fills all pixels in the same region as seed pixel bounded by boundary color with target color.

Example :

**II.3 How to perform a recursive pixel-to-pixel flood fill and the pseudocode**

In reality, the user will need to check whether the coordinates of the new seed pixel candidates are within the screen boundaries.

Procedure DoIt(X : integer, Y : integer)

**Declarations:**

**Algorithm:**

Set(X, Y, C)

If X > 0 AndAlso Point(X-1, Y) = C0 then

DoIt(X-1, Y)

If Y > 0 AndAlso Point(X, Y-1) = C0 then

DoIt(X, Y-1)

If X < ScreenWidth AndAlso Point(X+1, Y) = C0 then

DoIt(X+1, Y)

If Y < ScreenHeight AndAlso Point(X, Y+1) = C0 then

DoIt(X, Y+1)

**II.4 How to perform a recursive pixel-to-pixel boundary fill and the pseudocode**

In Boundary filling a seed point is fixed, and then neighboring pixels are checked to match with the boundary color. Then, color filling is done until boundary is reached.

Procedure DoIt(X : integer, Y : integer)

**Declarations:**

**Algorithm:**

Set(X, Y, C)

If Not Point(X-1, Y) ∈ {B, C} then

DoIt(X-1, Y)

If Not Point(X+1, Y) ∈ {B, C} then

DoIt(X+1, Y)

If Not Point(X, Y-1) ∈ {B, C} then

DoIt(X, Y-1)

If Not Point(X, Y+1) ∈ {B, C} then

DoIt(X, Y+1)

**II.5 The disadvantage of recursive pixel-to-pixel fill**

* If an inside pixel is in some other color then the fill terminates and the object remains unfilled.
* Seed fill method does not work for large objects.

**II.6 How to perform a pixel-to-pixel flood fill using a stack and the pseudocode**

An alternative to recursive region fill is by using a stack.

* Start with an empty stack.
* Push original seed pixel to the stack.
* Pop a pixel from the stack; change its color to target color.
* Check neighboring pixels. If a neighboring pixel fulfills the criteria, push that pixel to the stack.
* Repeat 3 – 4 until the stack is empty.

Procedure FloodFillwithStack(X : integer, Y : integer, C : TColor)

**Definitions:**

C0 : TColor

S : TStackOfPoints

**Algorithm:**

C0 🡨 Point(X,Y)

If C ≠ C0 then

InitStack(S)

Push(S, <X,Y>)

While not IsEmptyStack(S) do

Pop(S, <X,Y>)

Set(X,Y,C)

If Point(X-1,Y) = C0 then Push(S, <X-1,Y>)

If Point(X,Y-1) = C0 then Push(S, <X,Y-1>)

If Point(X+1,Y) = C0 then Push(S, <X+1,Y>)

If Point(X,Y+1) = C0 then Push(S, <X,Y+1>)

{IsEmptyStack(S)}

**II.7 How to perform a recursive scanline flood fill and the pseudocode**

An alternative to pixel per pixel region fill is by using a scanlne approach.

* + Start by filling the current line from seed pixel to left and right.
  + While filling the current scanline, test for the ends of spans above and below.
  + For each new free span, plant a seed.
  + Repeat until there are no more seeds.

Procedure DoIt(X : integer, Y : integer)

**Definitions:**

i, L, R : integer

**Algorithm:**

{scan from seed pixel to left and right}

i 🡨 x

If C0 ≠ C then

{scan left}

while i ≥ 0 AndAlso Point(i,y) = C0 do

Set(i,y,c)

i 🡨 i – 1

{i < 0 Or Point(i,y) ≠ C0}

L 🡨 i + 1 {set left}

i 🡨 x + 1

{scan right}

while i ≤ Width AndAlso Point(i,y) = C0 do

Set(i,y,c)

i 🡨 i + 1

{Point(i,y) ≠ C0}

R 🡨 i – 1 {set right}

**II.8 How to perform a recursive scanline flood fill using a stack and the pseudocode**

From the seed pixel, get to the leftmost pixel within the region having the same color as original pixel.

Two additional boolean variables are used, SpanAbove and SpanBelow, to “remember” whether pixels tested on the scanlines above and below are part of a new **span.**

Procedure ScanRegionFill(X : integer, Y : integer, C : TColor)

**Definitions:**

i : integer

spanabove, spanbelow : boolean

**Algorithm:**

C0 🡨 Point(x,y)

If C0 ≠ C then

Push(S, <x,y>)

while not IsEmptyStack(S) do

Pop(S, <x,y>)

{go to leftmost pixel of current scanline in region}

i 🡨 x

while i>0 AndAlso Point(i-1, y) = C0 do

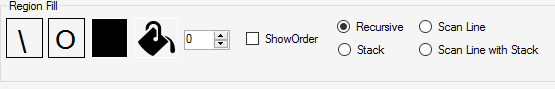
i 🡨 i - 1

{i = 0 Or Point(i-1,y) ≠ C0; (i,y) is the leftmost pixel}

# **Implementation**

## C:\Users\Mr. Celcius\Pictures\jch.PNG**III.1 Explanation Main Interface**

* File
  + New : to clear the screen and create blank canvas
  + Save : to save the bitmap in the picture box in .txt extension
  + Open : to open the file .txt
  + Delete : to delete the object in canvas
* Maximize : to maximize the windows screen
* Minimize : to minimize the windows screen
* Exit : to close the program
* **Region Fill**



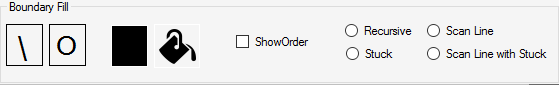
* Line : to create a line
* Circle : to create a circle
* Color : to create physical color
* Fill with Color : tools to fill the color

NumericUpDown : to up and down the value of the objects.

Show Order : to know the process of filling the color.

Fill color

* Recursive : to fill the region using color by checking pixel-to-pixel
* Stack : to fill the region using color by checking pixel-to-pixel. The checking prosses stored in stack to solve stack overflow
* Scan line : to fill the region using color by checking line-to-line.
* Scan line with stack : to fill the region using color by checking line-to-line. The checking prosses stored in stack to solve stack overflow
* **Boundary fill**

****

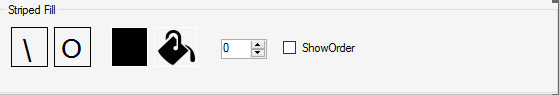
* Line : to create a line
* Circle : to create a circle
* Color : to create physical color
* Fill with Color : tools to fill the color

NumericUpDown : to up and down the value of the objects.

Show Order : to know the prosses of filling the color

Fill color

* Recursive : to fill the region using color by checking pixel-to-pixel
* Stack : to fill the region using color by checking pixel-to-pixel. The checking prosses stored in stack to solve stack overflow
* Scan line : to fill the region using color by checking line-to-line.
* Scan line with stack : to fill the region using color by checking line-to-line. The checking prosses stored in stack to solve stack overflow
* **Stripped Fill**

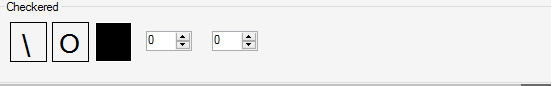


* Line : to create a line
* Circle : to create a circle
* Color : to create physical color
* Fill with Color : tools to fill the color

NumericUpDown : to up and down the value of the objects

Show Order : to know the prosses of filling the color

* **Checkered Fill**

****

* Line : to create line
* Circle : to create circle
* Color : to create physical color
* Fill with Color : tools to fill the color

NumericUpDown : to up and down the value of object 1

NumericUpDown2 : to up and down the value of object 2

## **III.2 Explanation The Features**

* Draw line

To draw a line using button of line that is in the main interface.

* Draw Circle

To draw a circle using button of circle that is in the main interface.

* Change the color

To change the object that was created by clicking the change color icon.

* Fill the color

To fill the object with color by clicking the fill color icon.

* Change the size of object

To change or set the size of the objects.

# **Design**

**IV.1 Main data structures**

* **Stack**

To solve the stack overflow in program.

* **Array**

An array is a data structure, which can store a fixed-size collection of elements of the same data type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

In the program, array uses for making the procedure in what happens in canvas.

* **Recursive**

In the program, recursive uses for looping the colouring procedures.

**IV.2 Main/Global variables**

* Dim Drag As Boolean

Dim MouseX As Integer

Dim MouseY As Integer

Dim GBCount As Integer = 0

Dim Max As Integer = 0

Dim Fitwidth(5) As Integer

Dim Fitheight(5) As Integer

Dim Wwidth As Integer

Dim Hheight As Integer

Dim Proc() As TProcess

Dim N, C, P As Integer

Dim bOund As Color

Dim down As Boolean = False

Dim mLine As Boolean = False

Dim mFill As Boolean = False

Dim mCircle As Boolean = True

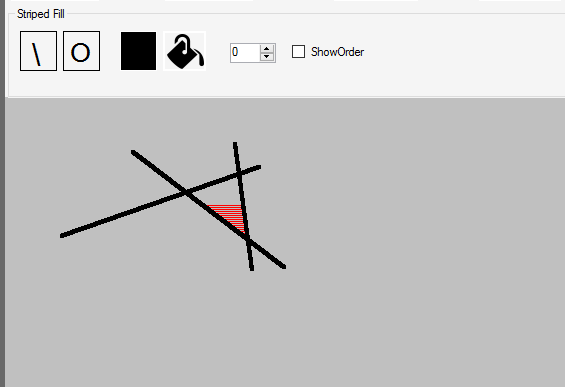
Dim pictTimer As PictureBox

Dim pictX As Integer

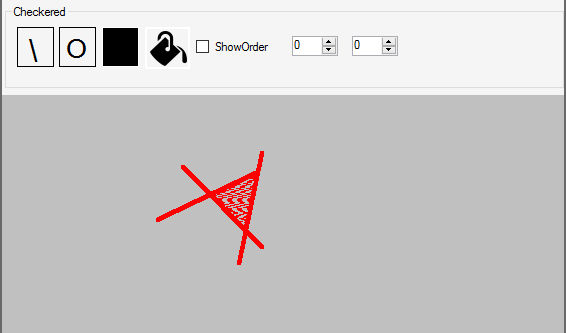
Dim pictY As Integer

Dim fillType As Integer

**IV.3 The implementation of checkered and striped line**

**Stripped Fill**

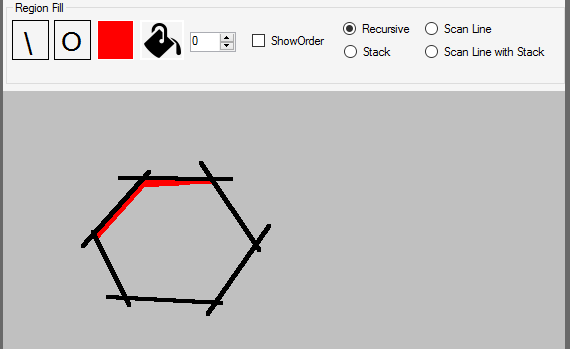
**Checkered Fill**

****

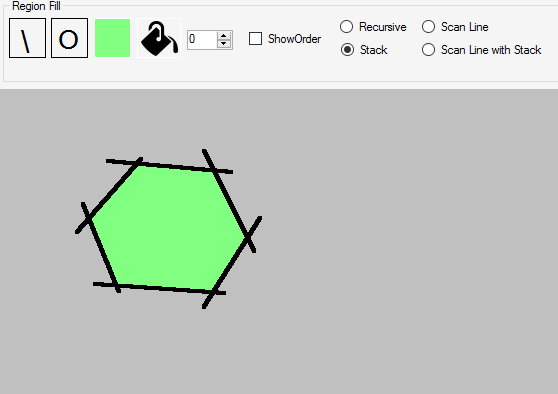
# **Evaluation**

#### **Flood Fill with Convex Region**

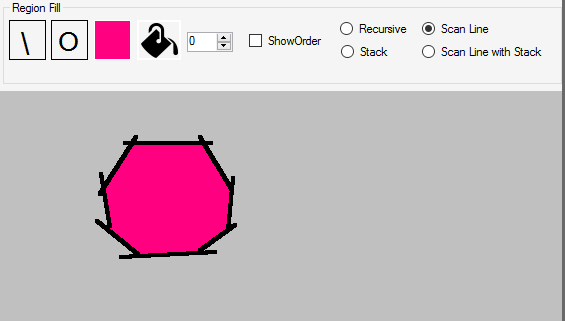
Flood fill pixel to pixel recursive



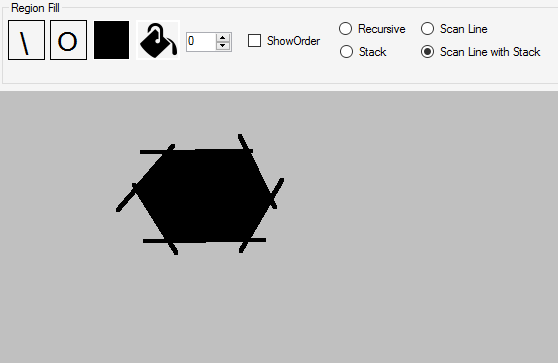
##### Flood Fill with Stack



##### Flood Fill Scan Line

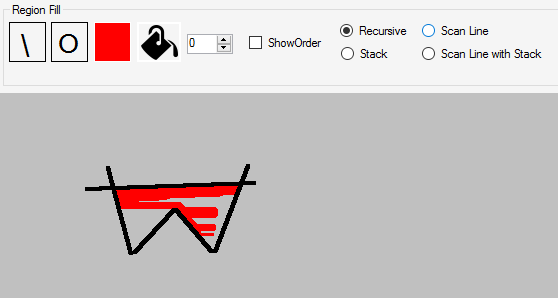


##### Flood Fill Scan Line with Stack

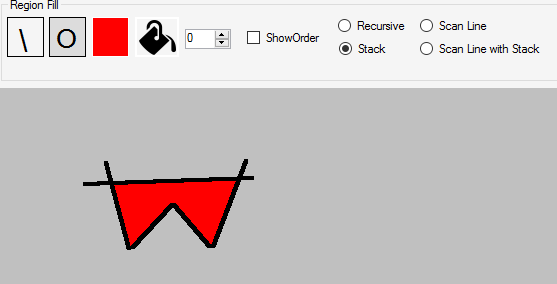


#### **Flood Fill with Non-Convex Region**

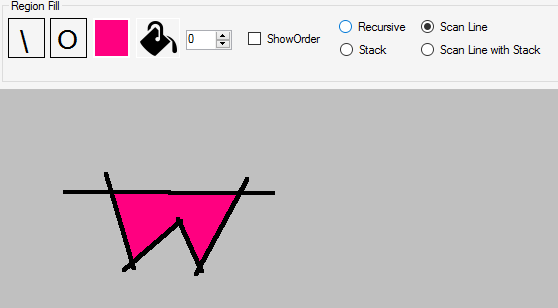
##### Flood Fill Pixel to Pixel Recursive



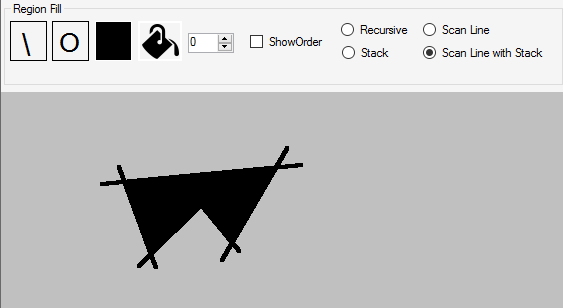
Flood Fill Pixel to Pixel Stack



##### Flood Fill Scan Line

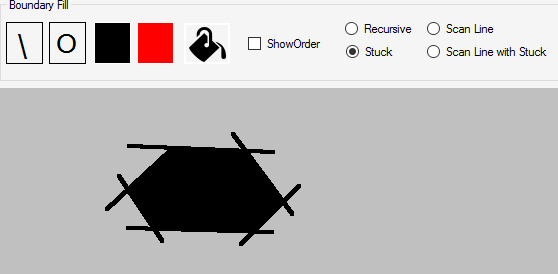


##### Flood Fill Scan Line with Stack

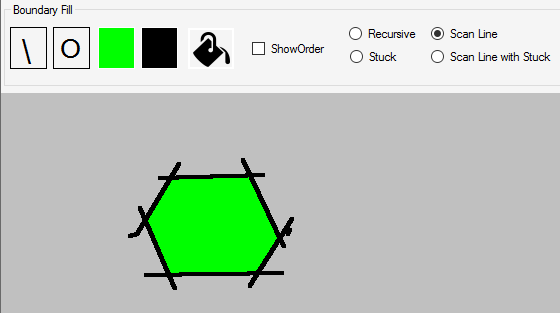


#### **Boundary Fill on a Convex Region**

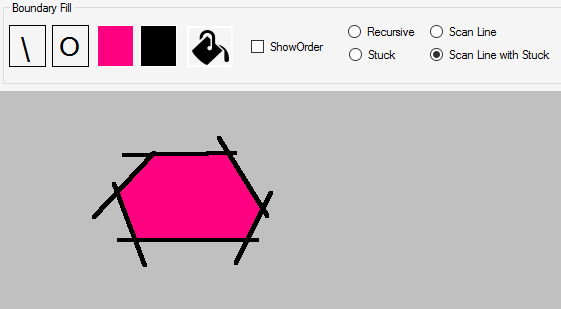
##### Boundary Fill Pixel to Pixel Stack



##### Boundary Fill Scan Line

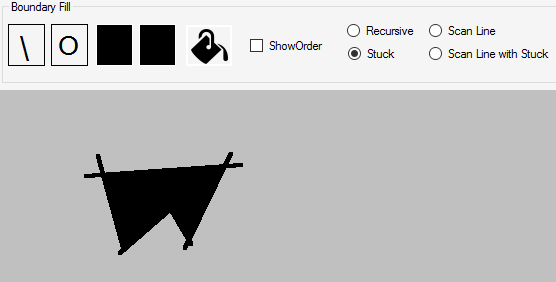


##### Boundary Fill Scan Line Stack

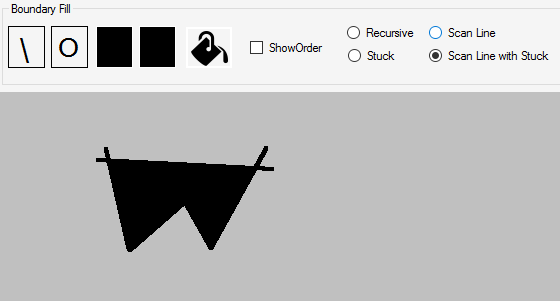


**Boundary Fill on a Non-Convex Region**

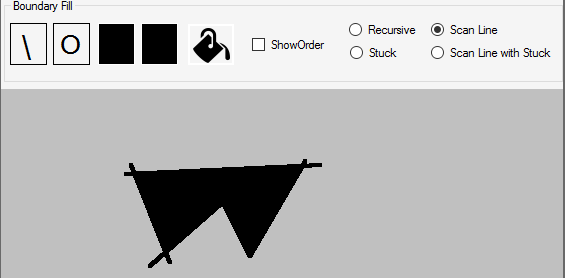
##### Boundary Fill Pixel to Pixel Stack



##### Boundary Fill Scan Line

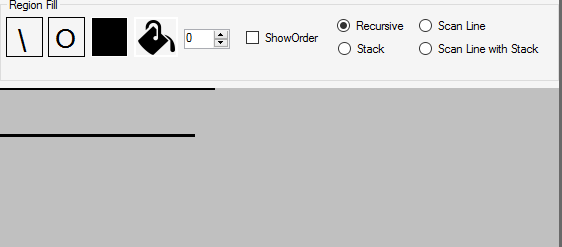


##### Boundary Fill Scan Line Stack

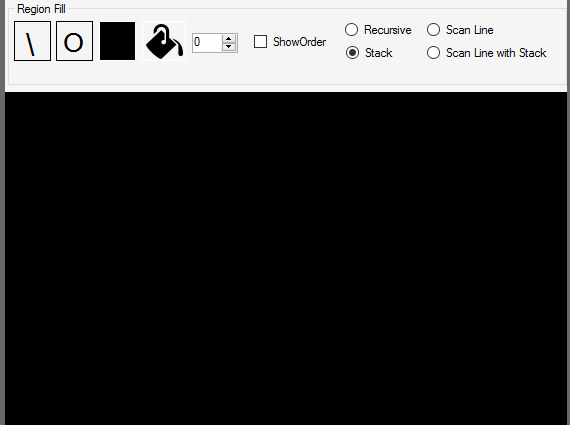


#### **Region Fill on an Empty Screen**

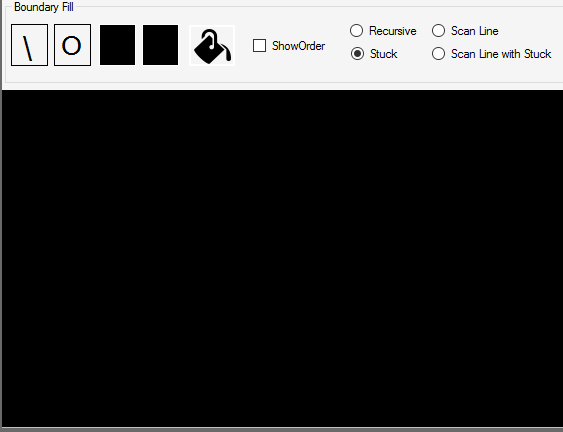
##### Flood Fill Recursive



##### Flood Fill Stack



##### Boundary Fill Stack



# **Work Log**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Author | Date | Commit Message |
| 1 | Marfin | 17 October 2017 | Draw line and Draw circle |
| 2 | Marfin | 18 Ocotber 2017 | Flood fill, flood fill with stack, scan line flood fill, scan line flood fill with stack |
| 3 | Marfin | 20 October 2017 | Boundary fill, Boundary fill with stack, scan line Boundary fill, scan line Boundary fill with stack |
| 4 | Marfin | 22 October 2017 | New, load, save functions |
| 5 | Cefriandy Simarmata | 29 October 2017 | Interface of the program |
| 6 | T. Ikhsan | 29 October 2017 | Show order |
| 7 | T. Ikhsan | 3 November 2017 | Report introduction and basic theory |
| 8 | Cefriandy Simarmata | 3 November 2017 | Revised the design of changing the code of design |
| 9 | Marfin | 5 November 2017 | Revised the bug of filling color |
| 10 | Cefriandy Simarmata | 5 November 2017 | Report implementation, design, work log, evaluation, and conclusion |
| 11 | T. Ikhsan | 5 November 2017 |

# **Conclusion and Remarks**

The program is completed and works fine. It is able to draw lines, circles, and do filling color such as flood fill, flood fill with stack, scanline region fill, scan line region fill with stack, boundary fill, boundary fill with stack, scanline boundary fill, scanline boundary fill with stack. The program can do filling the line with checkered pattern and striped pattern, it can do the show order and also it can do numeric up down.

However, there is bug in the program which is :

* + 1. In the boundary fill using recursive.
    2. Boundary fill on an empty screen.
    3. Flood fill on an empty screen.

In this project we learn how to work as a team to finish the task which is given by our beloved lecturer. It prepares us to be ready later when we graduate and work as a team in a coorperation or an office. In this project, we also know how to race with deadline which is given by our lecturer and it seems like a real work.