President University

Report of Programming Assignment 12

Polygon Clipping: Weiler Atherton

Computer Graphics and Animations

Hilman Revanda (001201500038)

Nikita Chrissandha M (001201500031)

Predrika Br Ginting (001201500032)

CIT 2 2015

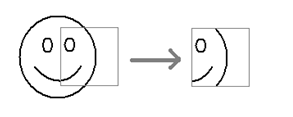
# Introduction

The program is about drawing and clipping polygon/polygons. The Weiler – Atherton Algorithm for polygon clipping is fully implemented in this program. Other than those, user can also refresh the screen, delete, and save the polygon/polygons.

This program was created using Visual Basic programming language. The report includes basic theory, how to use the application, design of the application, evaluation of the main features, work log, and conclusion and remarks.

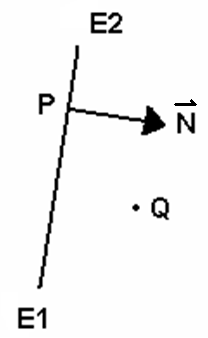
# Basic Theory

Clipping is procedure that identifies those portions of a picture that are either inside or outside of a specified region of a space. Clipping is the extraction of data/primitives inside a region of “interest” window. The primary use of clippimg is to remove objects, lines, or line segments that are outside the window.



The Cyrus–Beck algorithm is a generalized line-clipping algorithm. Cyrus–Beck is a line-clipping algorithm that can be used with any convex region as a clipping window, which is can be used also on a rectangular clipping area.

## Point Clipping

E1E2 = window edge

F(Q) = (Q – P) **.** **N**

P = a point on the line

**N** = normal vector of E1E2 pointing inside the window   
 . = dot product

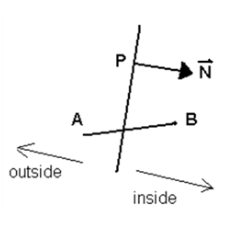
F(Q) determines whether the angle between **PQ** and **N** is greater than or less than 900.

F(Q) > 0 🡪 Q is “inside” the line

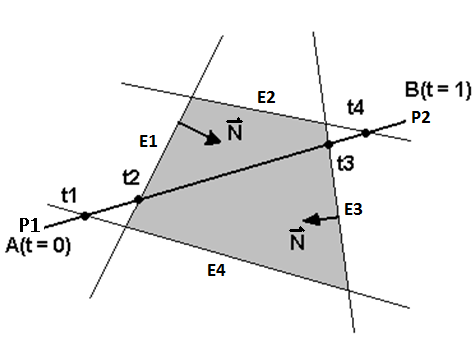
F(Q) < 0 🡪 Q is “outside” the line

F(Q) = 0 🡪 Q is on the line

## Line Clipping

****Window edge : F(Q) = (Q – P) . **N**  
 Line segment :  
 L(t) = A + t (B – A)  
 a. Trivial reject :  
 F(A) < 0 and F(B) < for   
 at least one edge of  
 the clipping region.  
 b. Trivial accept :  
 F(A) ≥ 0 and F(B) ≥ 0   
 for all edges of the clipping region.  
  
Find t :

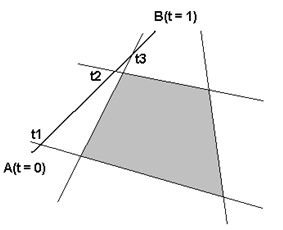
t = (A – P) . **N** - (B – A) . **N**Let D = (B – A) **. N**

 If D > 0, then the angle between  
 (B –A) and **N** is less than 90o.

Label t as tE (entering).

t1 and t2 are “entering”.

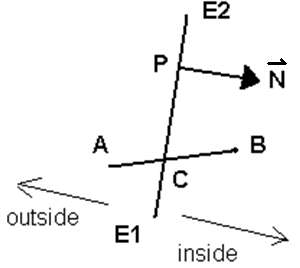
If D < 0, then the angle between (B – A)  
 and **N** is more than 90o.  
 Label t as tL (leaving).  
 t3 and t4 are “leaving”.

t of interest:  
 Largest tE (tEmax)  
 Smallest tL (tLmin)  
 Large/small here refers to the index of t, not the value.  
 In this case, t2 and t3 are selected.

Reject :  
tEmax > tLmin

t1 = entering  
t2 = leaving  
t3 = entering

## Line Intersection

AB intersects window at C  
F(C) = 0  
C = L(t) = A + (B – A) t

Thus:  
F(C) = 0  
(C – P) **.** **N** = 0  
(A + t (B – A) – P) **.** **N** = 0  
(A – P) **.** **N** + t (B – A) **.** **N** = 0

t = (A – P) **.** **N** `

(A – B) **.** **N**

= (A – P) **.** **N** `

– (B – A) **.** **N**

Polygon can be defined as set of points or set of vectors. Polygon clipping is procedure that identifies those portions of a polygon that are either inside or outside of a specified region of a space. An algorithm that clips a polygon must deal with many different cases. Each edge of the polygon must be tested against each edge of the clipping window; new edges must be added, and existing edges must be discarded or divided.

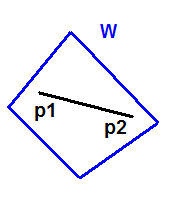
The Weiler-Atherton is a polygon clipping algorithm. The Weiler-Atherton algorithm is based on the Cyrus-Beck and Liang-Barsky algorithm. While most clipping algorithms are optimized for a rectangular clipping region, the Weiler-Atherton algorithm can use simple polygons for both the subject of the clipping as well as the actual clipping region itself. In this algorithm, the clipping window must be convex, but the polygon to be clipped can be convex or non-convex. A clipped polygon may result in more than one polygon. There are no degenerate lines in this polygon clipping algorithm.

The algorithm of Weiler-Atherton polygon clipping :

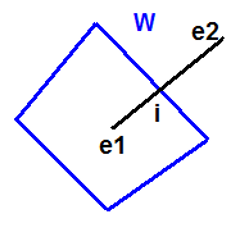
1. Initialize the polygon and clipping window as separate circular lists of vertices (CLP and CLW). The elements of CLP and CLW are sorted based on order of vertices.
2. Traverse each edge of the polygon; find intersections with the clipping window. Insert each intersection into CLP and CLW. Mark each intersection as “entering” or “leaving”.
3. Start at an entering vertex.
4. If you encounter a leaving vertex swap to clipping window loop.
5. If you encounter an entering vertex swap to polygon loop.
6. A loop is finished when you arrive back at start.
7. Repeat until all entering vertices have been processed.

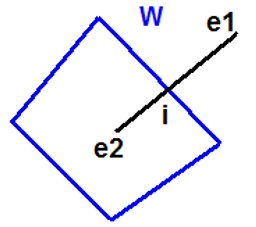
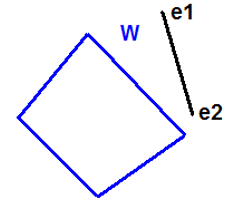
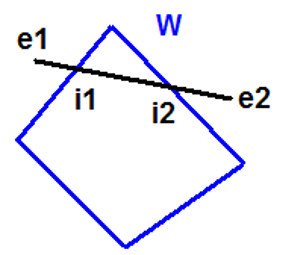
There are 4 (four) main cases in Weiler-Atherton polygon clipping algorithm.

Case 1   
Polygon edge is entirely inside the clipping window.

No action

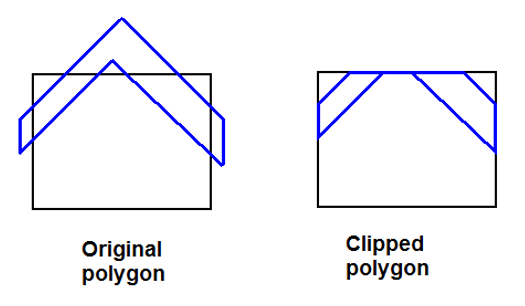
Case 2  
Polygon edge crosses the clipping window going out

 Find i.  
 Add i to both CLP and CLW.   
 Mark i as “leaving”.

Case 3  
Polygon edge crosses the clipping window edge going in  
   
 Find i.  
 Add i to both CLP and CLW.  
 Mark i as “entering”.  
  
  
  
  
  
  
Case 4  
Polygon edge is entirely outside the clipping window edge.  
Find intersections.  
There are two possibilities :  
  
  
  
There are no intersections(i).  
  
No action  
  
  
  
  
  
  
  
  
  
  
  
There are exactly two intersections.  
  
Find i1 and i2.  
  
Determine the correct order between i1 and i2.   
  
Add i1 and i2 to both CLP and CLW.  
  
Mark i1 as “entering” and i2 as “leaving”.

The main limitation of Sutherland-Hodgman algorithm

* The clipping window must be rectangle.
* Clipping certain polygons may result in degenerate edges.
* In general, degenerate edges may be irrelevant.
* The result can be only 1 polygon.



## Pseudocode

Function IsPolygonConvex(polygon As List(Of Point)) As Boolean

Dim N As Integer = 0

Dim P As Integer = 0

Dim nextpoint As Integer

Dim nextnextPoint As Integer

For currentpoint = 0 To polygon.Count - 1

If currentpoint + 1 = polygon.Count Then

nextpoint = 0

nextnextPoint = 1

ElseIf currentpoint + 2 = polygon.Count Then

nextnextPoint = 0

Else nextpoint = currentpoint + 1

nextnextPoint = currentpoint + 2

End If

If CrossProductOf(((polygon(nextpoint).X) - (polygon(currentpoint).X)), ((polygon(nextnextPoint).X) - (polygon(nextpoint).X)),

((polygon(nextpoint).Y) - (polygon(currentpoint).Y)), ((polygon(nextnextPoint).Y) - (polygon(nextpoint).Y))) >= 0 Then

P = P + 1

Else N = N + 1

End If

Next

'MsgBox(P.ToString & " " & N.ToString)

If N = 0 And P > 0 Then

Clockwise = True

Return True

ElseIf P = 0 And N > 0 Then

Clockwise = False

Return True

Else

MsgBox("Not a convex clipping")

Return False

End If

End Function

Function CrossProductOf(Ax As Integer, Bx As Integer, Ay As Integer, By As Integer) As Integer

Return (Ax \* By) - (Ay \* Bx)

End Function

Function ClippingPoint(Polygon As List(Of Point), Rect As List(Of Point)) As Point

Dim C As Integer

C = 0

Dim B As Integer

Dim T As Integer

Dim NP As Point

Dim NW As Point

Dim TempIntersection As New List(Of LinkedLValue)

Dim TempLinkedLIntersection As LinkedLValue

Dim TempPoint As Point

Dim Status As String

'NewPolygon = New List(Of Point)()

For A = 0 To Polygon.Count - 1

B = NextPoint(A, Polygon.Count)

For S = 0 To Rect.Count - 1

T = NextPoint(S, Rect.Count)

NW = Normal(Rect(S), Rect(T))

NP = Normal(Polygon(A), Polygon(B))

If (Not InsidePoint(Rect(S), Rect(T), Polygon(B)) And (Not InsidePoint(Rect(S), Rect(T), Polygon(A)))) Then

C = C + 1

ElseIf (Not InsidePoint(Rect(S), Rect(T), Polygon(B)) And InsidePoint(Rect(S), Rect(T), Polygon(A))) Then 'False and True means out in

'EN

'MsgBox("edge " & S & T & " with " & A & B & " is EN")

If TisAcc(Tis(Polygon(A), Polygon(B), Rect(S), NW)) And TisAcc(Tis(Rect(S), Rect(T), Polygon(A), NP)) Then

TempPoint = SetTPoint(Polygon(A), Polygon(B), Tis(Polygon(A), Polygon(B), Rect(S), NW))

Status = EN

TempLinkedLIntersection = New LinkedLValue

TempLinkedLIntersection.NewI(Tis(Polygon(A), Polygon(B), Rect(S), NW),

ToPoint(A, B),

Tis(Rect(S), Rect(T), Polygon(A), NP),

ToPoint(S, T),

TempPoint,

Status)

TempIntersection.Add(TempLinkedLIntersection)

'TempIntersection = SetNextPandW(TempIntersection)

End If

ElseIf (InsidePoint(Rect(S), Rect(T), Polygon(B)) And Not InsidePoint(Rect(S), Rect(T), Polygon(A))) Then 'true and false means in out

'LEAV

'MsgBox("edge " & S & T & " with " & A & B & " is LEAV")

If TisAcc(Tis(Polygon(A), Polygon(B), Rect(S), NW)) And TisAcc(Tis(Rect(S), Rect(T), Polygon(A), NP)) Then

TempPoint = SetTPoint(Polygon(A), Polygon(B), Tis(Polygon(A), Polygon(B), Rect(S), NW))

Status = LEAV

TempLinkedLIntersection = New LinkedLValue

TempLinkedLIntersection.NewI(Tis(Polygon(A), Polygon(B), Rect(S), NW),

ToPoint(A, B),

Tis(Rect(S), Rect(T), Polygon(A), NP),

ToPoint(S, T),

TempPoint,

Status)

TempIntersection.Add(TempLinkedLIntersection)

'TempIntersection = SetNextPandW(TempIntersection)

End If

End If

Next

Next

If Polygon.Count \* Rect.Count = C Then

Clippings.Add(Polygons.First)

Else

Intersection = New List(Of LinkedLValue)

Intersection = TempIntersection

SetNext()

'DrawIntersection()

NewPolygon = New List(Of Point)

Dim CurrentPos As LinkedLValue = New LinkedLValue

Dim Start As LinkedLValue = New LinkedLValue

For i = 0 To Intersection.Count - 1

NewPolygon = New List(Of Point)

If Intersection(i).status Is EN Then

CurrentPos = Intersection(i)

Start = Intersection(i)

NewPolygon.Add(CurrentPos.point)

CurrentPos = CurrentPos.NextP

NewPolygon.Add(CurrentPos.point)

Do While CurrentPos IsNot Start

If CurrentPos.status Is LEAV Then

CurrentPos = CurrentPos.NextW

NewPolygon.Add(CurrentPos.point)

Do Until CurrentPos.status Is EN

CurrentPos = CurrentPos.NextW

NewPolygon.Add(CurrentPos.point)

Loop

Else

CurrentPos = CurrentPos.NextP

NewPolygon.Add(CurrentPos.point)

Do Until CurrentPos.status Is LEAV

CurrentPos = CurrentPos.NextP

NewPolygon.Add(CurrentPos.point)

Loop

End If

Loop

Clippings.Add(NewPolygon)

NewPolygon = Nothing

End If

Next

End If

Tolistbox = True

End Function

'Fungsi ini menentukan inside atau outside dari saru point saja (Point S)

Function InsidePoint(WA As Point, WB As Point, S As Point) As Boolean

Dim N As Point

Dim D As Point

N = Normal(WA, WB)

D.X = (S.X - WA.X) \* N.X

D.Y = (S.Y - WA.Y) \* N.Y

If D.X >= 0 And D.Y >= 0 Then

Return True

Else

Return False

End If

End Function

Function Normal(WA As Point, WB As Point) As Point

Dim N As Point

N.X = WB.Y - WA.Y

N.Y = WB.X - WA.X

If (Clockwise) Then

N.Y = N.Y \* -1

ElseIf (Not Clockwise) Then

N.X = N.X \* -1

End If

Return N

End Function

Function NextPoint(Point As Integer, Total As Integer) As Integer

If Point + 1 = Total Then Return 0

Return Point + 1

End Function

Function Tis(A As Point, B As Point, P As Point, N As Point) As Decimal

Return ((((P.X - A.X) \* N.X) + ((P.Y - A.Y) \* N.Y)) / (((B.X - A.X) \* N.X) + ((B.Y - A.Y) \* N.Y))) \* 1.0

End Function

Function TisAcc(X As Decimal) As Boolean

If X >= 0 And X <= 1 Then Return True

Return False

End Function

'ShowList(Head, Head) just to show linkedlist

'Sub ShowList(Start As LinkedLValue, Current As LinkedLValue)

' MsgBox(Current.Point.ToString)

'If Current.NextList IsNot Start Then

' ShowList(Start, Current.NextList)

'End If

'End Sub

Function ToPoint(X As Integer, Y As Integer) As Point

Dim A As Point

A.X = X

A.Y = Y

Return A

End Function

Function PolygonstoLinkedList() As List(Of List(Of LinkedLValue))

Dim ListofPolygonLinkedList As List(Of List(Of LinkedLValue)) = New List(Of List(Of LinkedLValue))

Dim Temp As List(Of LinkedLValue)

For Each Polygon As List(Of Point) In Polygons

Temp = PolygontoLinkedList(Polygon)

ListofPolygonLinkedList.Add(Temp)

Next

Return ListofPolygonLinkedList

End Function

Function PolygontoLinkedList(JustPolygon As List(Of Point)) As List(Of LinkedLValue)

Dim ListRect As New List(Of LinkedLValue)

Dim Head As LinkedLValue = New LinkedLValue 'still useless

Dim LinkedLValueTemp As LinkedLValue

Dim Polygon As List(Of Point) = JustPolygon

Dim Temp As Point

Dim isPolygon As Boolean = True

If JustPolygon Is Polygons.Last Then

isPolygon = False

End If

For i = 0 To Polygon.Count - 1

Temp = Polygon(i)

LinkedLValueTemp = New LinkedLValue

LinkedLValueTemp.NewP(Temp)

ListRect.Add(LinkedLValueTemp)

If i = 0 Then

Head = ListRect(i)

If isPolygon Then

ListRect(i).NextP = Head

ListRect(i).status = "P"

Else

ListRect(i).NextW = Head

ListRect(i).status = "W"

End If

Else

If isPolygon Then

ListRect(i - 1).NextP = ListRect(i)

ListRect(i).NextP = Head

ListRect(i).status = "P"

Else

ListRect(i - 1).NextW = ListRect(i)

ListRect(i).NextW = Head

ListRect(i).status = "W"

End If

End If

Next

Return ListRect

End Function

Function SetTPoint(A As Point, B As Point, T As Decimal) As Point

Dim Result As Point

Result.X = A.X + (T \* (B.X - A.X))

Result.Y = A.Y + (T \* (B.Y - A.Y))

Return Result

End Function

Sub SetNext()

Dim Polygon = ListofPolygonsLinkedList(0)

Dim Window = ListofPolygonsLinkedList(1)

Dim B As Integer

Dim Point As Point

Dim Inter As List(Of Integer)

Dim Start, Endd As Integer

For A = 0 To Polygon.Count - 1

B = NextPoint(A, Polygon.Count)

Point.X = A

Point.Y = B

Inter = IntersectionExistP(Point)

If Not (Inter.Count = 0) Then

Start = Intersection(Inter.First).p.X

Endd = Intersection(Inter.First).p.Y

End If

If Inter.Count = 1 Then

Polygon(Start).NextP = Intersection(Inter.First)

Intersection(Inter.First).NextP = Polygon(Endd)

ElseIf Inter.Count > 1 Then

Inter = SortlistP(Inter)

For Each i In Inter

Console.WriteLine(Intersection(i).tp)

Next

Polygon(Start).NextP = Intersection(Inter.First)

For i = 0 To Inter.Count - 1

If Not (i + 1 = Inter.Count) Then

Intersection(Inter(i)).NextP = Intersection(Inter(i + 1))

End If

Next

Intersection(Inter.Last).NextP = Polygon(Endd)

End If

Next

For A = 0 To Window.Count - 1

B = NextPoint(A, Window.Count)

Point.X = A

Point.Y = B

Inter = IntersectionExistW(Point)

If Not (Inter.Count = 0) Then

Start = Intersection(Inter.First).w.X

Endd = Intersection(Inter.First).w.Y

End If

If Inter.Count = 1 Then

Window(Start).NextW = Intersection(Inter.First)

Intersection(Inter.First).NextW = Window(Endd)

ElseIf Inter.Count > 1 Then

Inter = SortlistW(Inter)

Window(Start).NextW = Intersection(Inter.First)

For i = 0 To Inter.Count - 1

If Not (i + 1 = Inter.Count) Then

Intersection(Inter(i)).NextW = Intersection(Inter(i + 1))

End If

Next

Intersection(Inter.Last).NextW = Window(Endd)

Else

Window(Start).NextW = Window(Endd)

End If

Next

End Sub

Function IntersectionExistP(Point As Point) As List(Of Integer)

Dim Result As List(Of Integer) = New List(Of Integer)

For i = 0 To Intersection.Count - 1

If Intersection(i).p = Point Then

Result.Add(i)

End If

Next

Return Result

End Function

Function IntersectionExistW(Point As Point) As List(Of Integer)

Dim Result As List(Of Integer) = New List(Of Integer)

For i = 0 To Intersection.Count - 1

If Intersection(i).w = Point Then

Result.Add(i)

End If

Next

Return Result

End Function

Function SortlistP(list As List(Of Integer)) As List(Of Integer)

Dim C As Integer = 0

Dim Temp As Integer

Do While C = 0

For i = 0 To list.Count - 1

If Not (i + 1 = list.Count) Then

If Intersection(list(i)).tp > Intersection(list(i + 1)).tp Then

Temp = list(i)

list(i) = list(i + 1)

list(i + 1) = Temp

C = C + 1

End If

End If

If i = list.Count - 1 And C = 0 Then

Exit Do

End If

Next

If C > 0 Then C = 0

Loop

Return list

End Function

Function SortlistW(list As List(Of Integer)) As List(Of Integer)

Dim C As Integer = 0

Dim Temp As Integer

Do While C = 0

For i = 0 To list.Count - 1

If Not (i + 1 = list.Count) Then

If Intersection(list(i)).tw > Intersection(list(i + 1)).tw Then

Temp = list(i)

list(i) = list(i + 1)

list(i + 1) = Temp

C = C + 1

End If

End If

If i = list.Count - 1 And C = 0 Then

Exit Do

End If

Next

If C > 0 Then C = 0

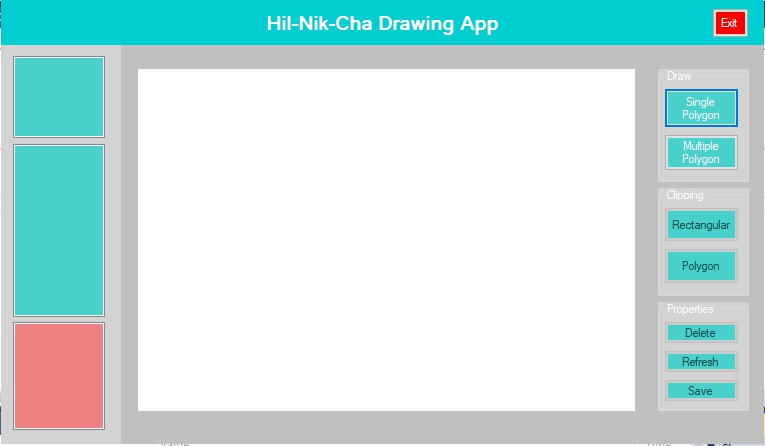
Loop

Return list

End Function

# Implementation

## Main interface

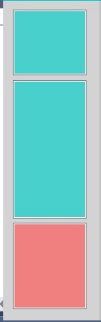


The application is named Hil-Nik-Cha Drawing App which was named after the developers. With color palette of silver, brown, dark turquoise, black and white, the user interface looks flat and modern. The features of the app include draw single and multiple polygon, Clip the polygon with rectangle and polygon, delete polygon, refresh the screen, save polygon, and showing the list of polygon and clipping.

## Canvas/Picture Image

The picture box is used as canvas to draw polygons. After selecting the draw button, the user could click anywhere on the canvas to create polygon. Right click is used to finish the polygon.

## List Box

List box is where the points of all polygons and clipping showed.

## Draw

Single Polygon and Multiple Polygon are both in the Draw group box. Single Polygon is used to draw a single polygon. When the user use this button, the previous polygon will be deleted.

## Clipping

In the Clipping group box, there are Rectangular and Polygon Clipping. The rectangular clipping is used to clip with a rectangle. On the other hand, Polygon button is used to clip with a free and multiple angles(polygon).

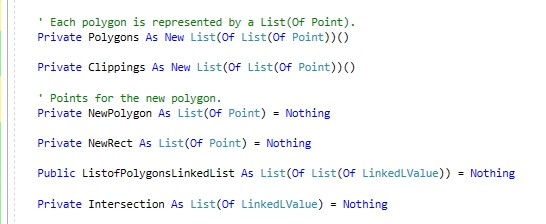
## Properties

In the properties, there are Delete, Refresh, and Save button. Delete button is used for deleting line or polygon. Refresh button is used for clearing the screen and the list box. And Save button is used for saving the canvas into bitmap.

## Exit

Exit button is used to exit the app.

# Design



Points are represented as list of point and polygons is represented as list of lists of polygons. When user add a new polygon, the points will be assigned into variable named newPolygon. This newPolygon will be added into the list of polygons and the newPolygon will be declared as nothing.

The clipping starts from polygons’ variable (list (of list (of point))), where polygon.last is a clipping polygon. Then the clipping function will be executed which will determine whether the point is inside or outside.

## Explain the main/global variables used in the program.

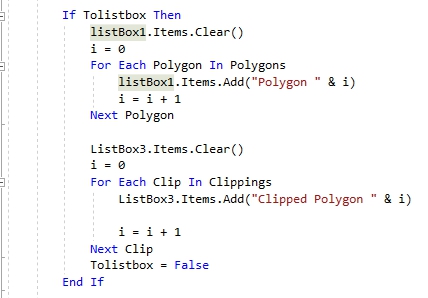


The variables:

* i is index and declared as integer.
* ButtonMenu is declared as string. It is declared to know which button is chosen.
* Clockwise is declared as Boolean to determine whether the clipping is clockwise or not.
* New point is the current position of the mouse while drawing a new polygon.
* TempPoint is to store the temporary point.
* En is declared as string and a line is assigned to “En” if the line is entering the clipping.
* Leav is declared as string and a line is assigned to “Leav” if the line is entering the clipping.

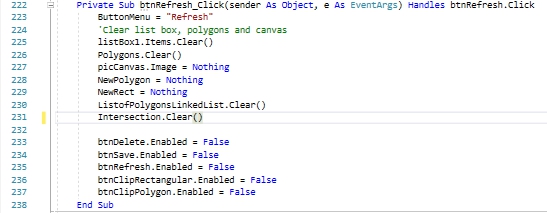
## Implementation of Bonus

### List Box of Polygon, Detail Point, and Clipping



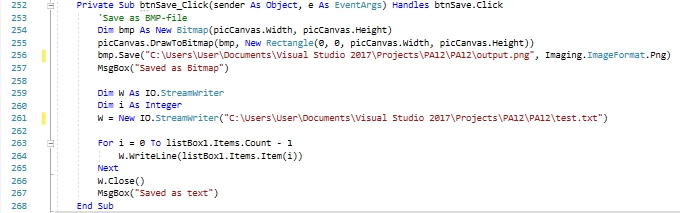
The app has three list boxes. The first one contains the list of polygons. The second contains the detailed points of the clipping and the polygons. And the last contains the clipped polygon.

### Refresh the Screen



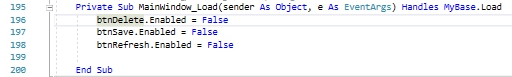
When the button is clicked, polygons, the list box, list of polygons linked list, intersection will be emptied. New polygon, new rectangular clipping, and the canvas will be set to be nothing.

### Save Polygon into Bitmap and Text File



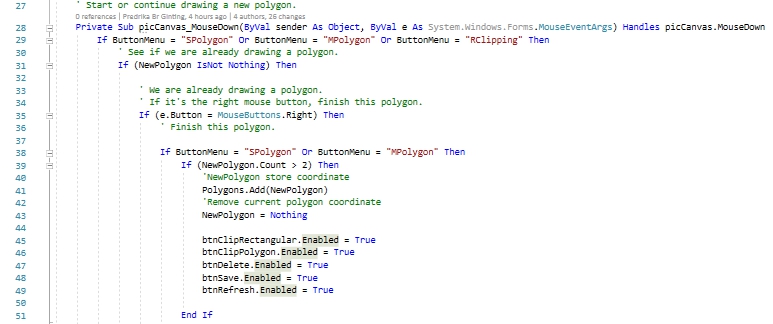
The polygon drawn on the canvas will saved as text and bitmap when the user clicks the Save button.

### Enable and Disable Button





When there’s still nothing on the canvas, the Delete, Save, and Refresh button will be disabled.

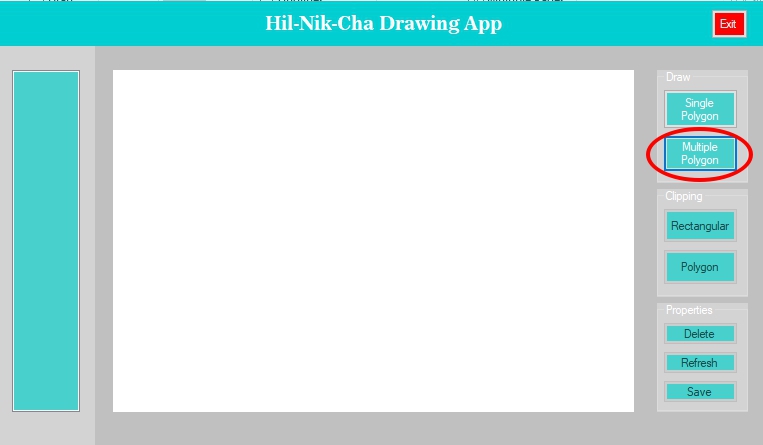


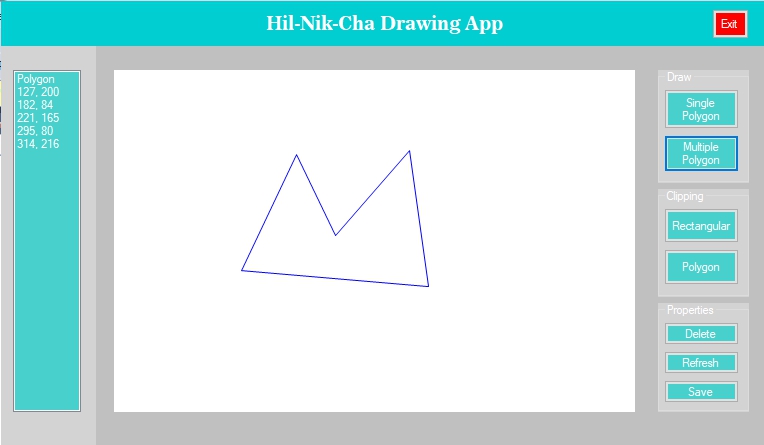
After a polygon is drawn, the Clipping and all the Properties Buttons will be enabled.

### Delete the Line

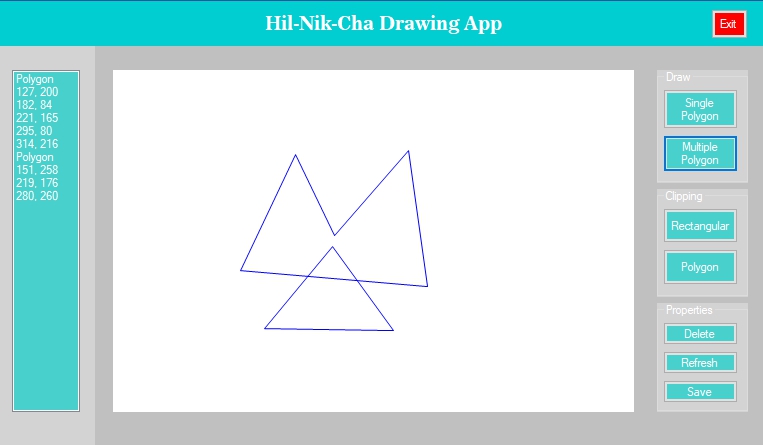
# Evaluation

## Adding a Polygon into the Screen

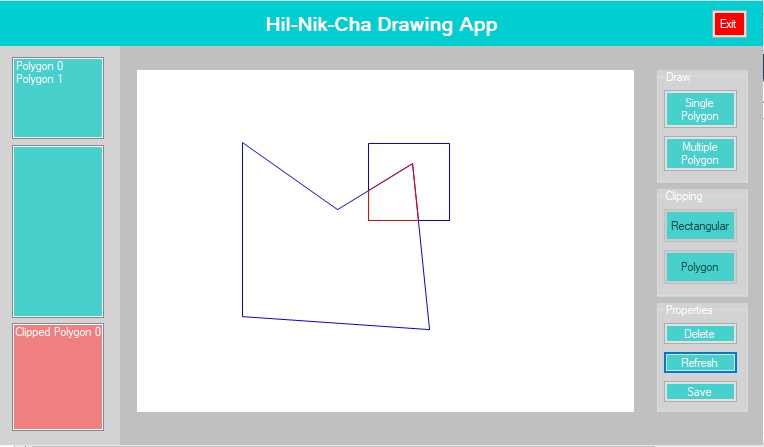
First, select button from Draw Group Box, which one the user wants to draw. Click the Single Polygon button for one polygon only or Multiple Polygon button for multiple polygon.

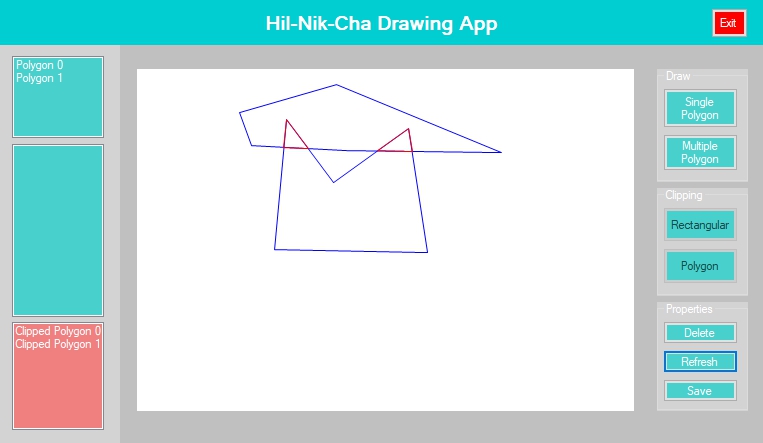
Then click in the picture box to draw the polygon. Right click to finish the polygon.

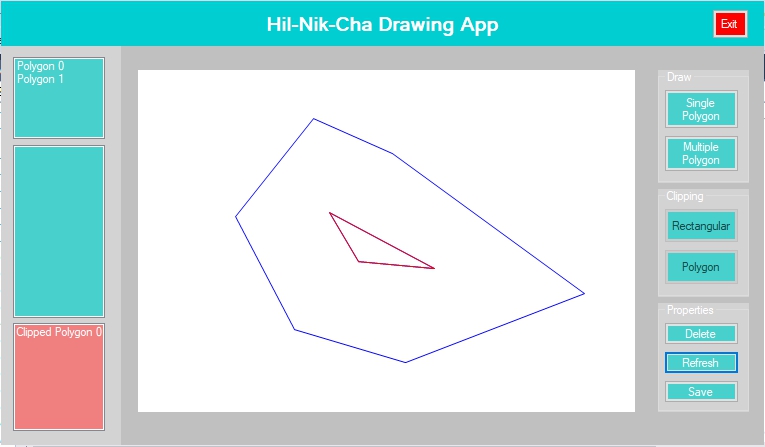
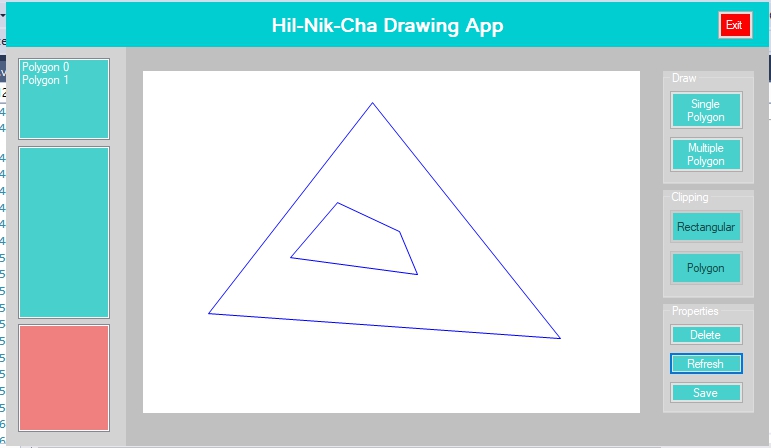
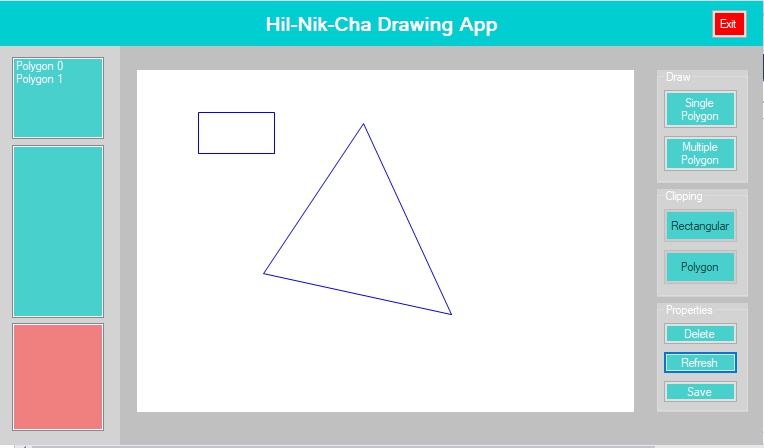
## Adding Another Polygon into the Screen

When the user wants to add another polygon, the user just needs to clicks on the picture box to determine the points of the polygon and then right click to finish the polygon. Do the same if the user wants to add more.

## Clipping



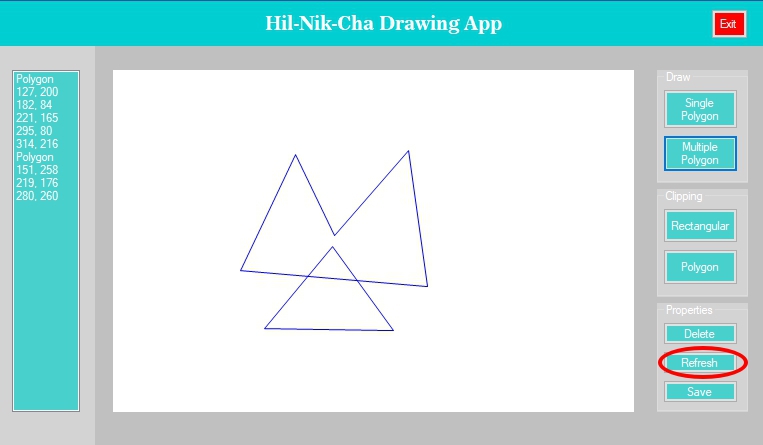




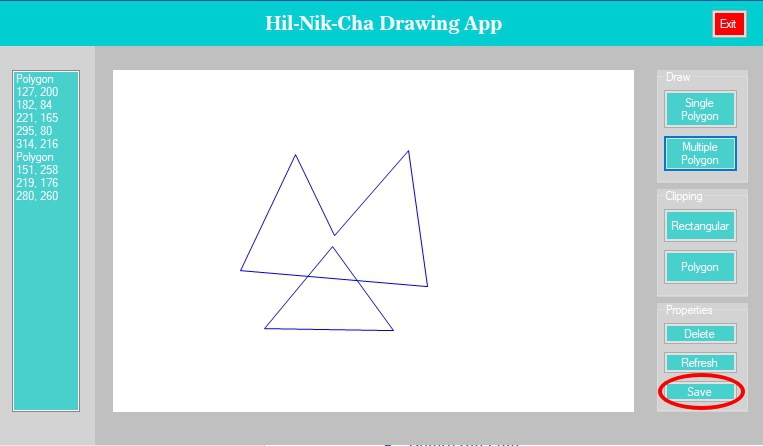
## List Box of Polygon and Clipping

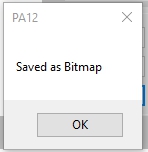
The list box will show the list of point of polygons and clipping.

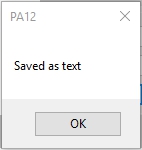
## Refresh the Screen

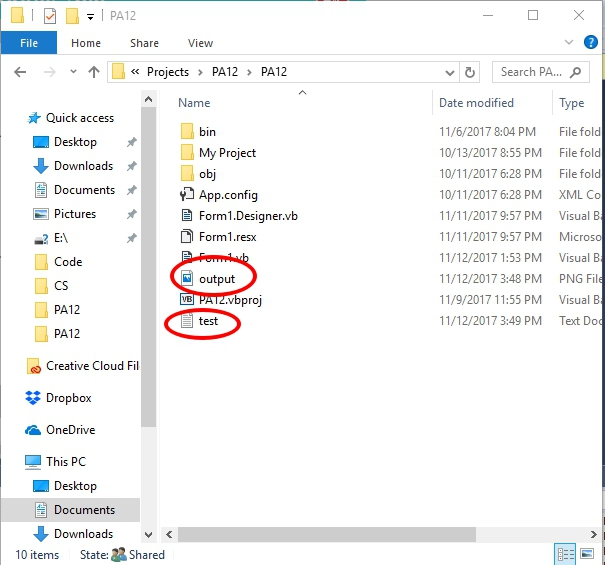
When the user wants to clear the screen, he/she only needs to click the Refresh button.

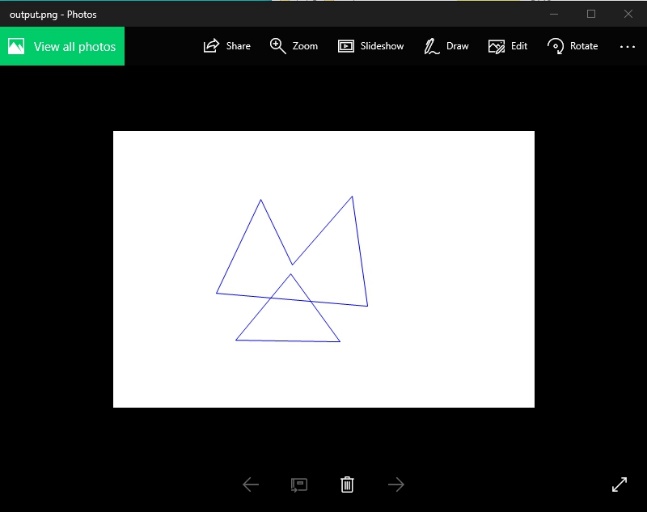
## Save Polygon into Bitmap and Text File

If the user wants to save the polygons which he/she has drawn, he/she only needs to click the Save button, then the app will save it into bitmap and text file.

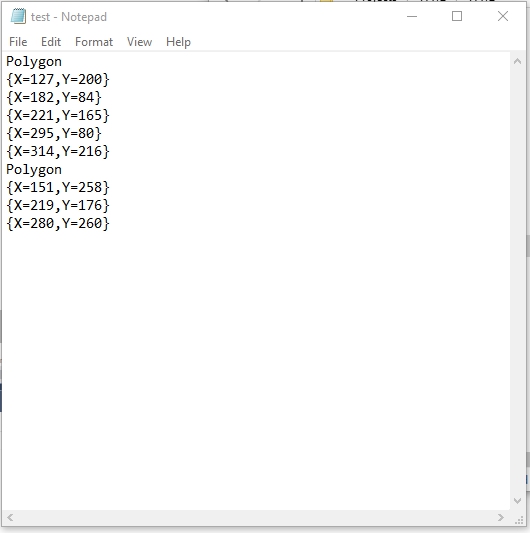
This means, the drawing is already saved as bitmap file.

And this means, the drawing is already saved as text.

If the user checks the folder, the bitmap and text is already there.

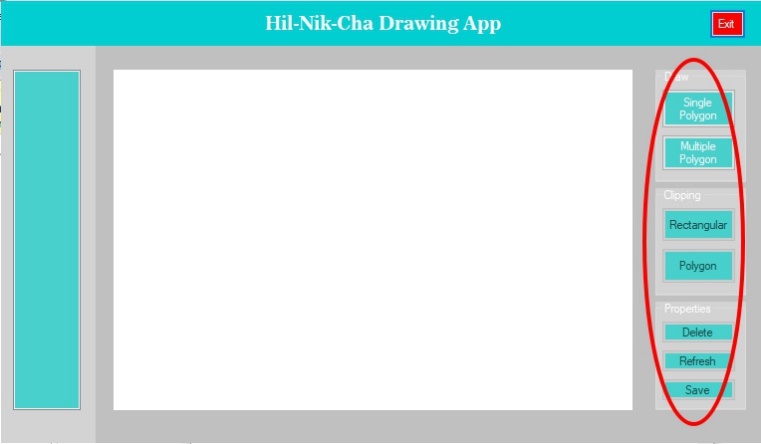


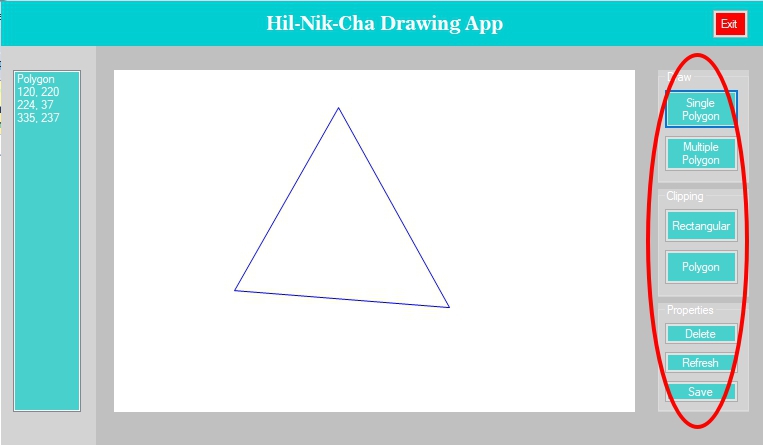
This is the output of bitmap file.

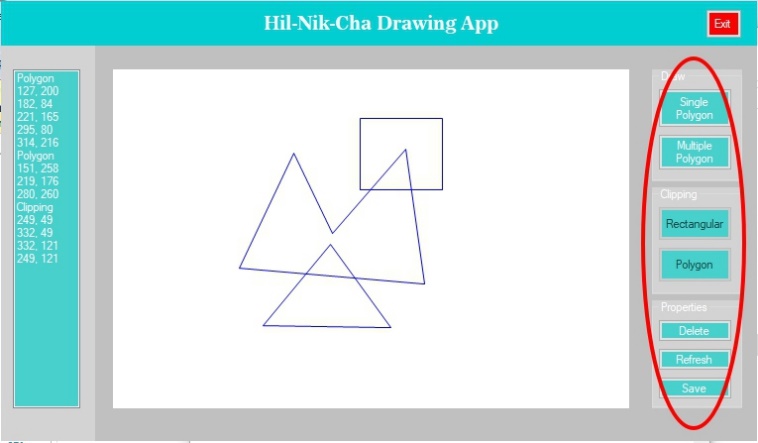


And this is the output of text file.

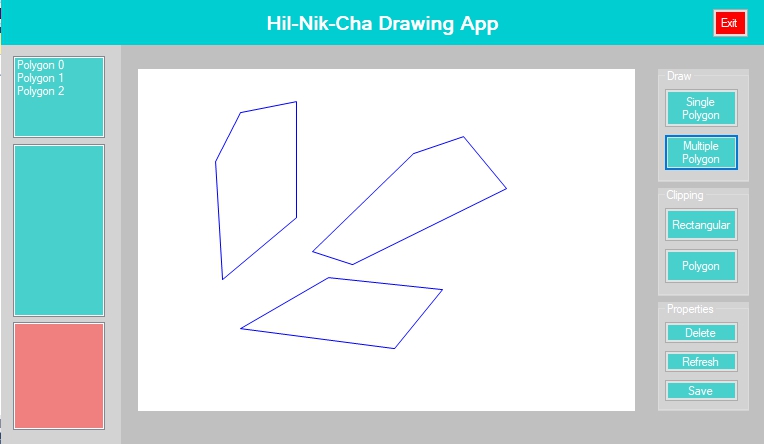
## Enable and Disable Button

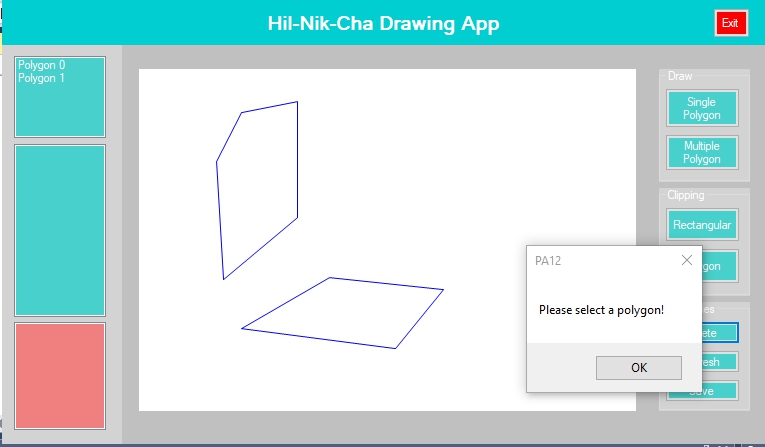
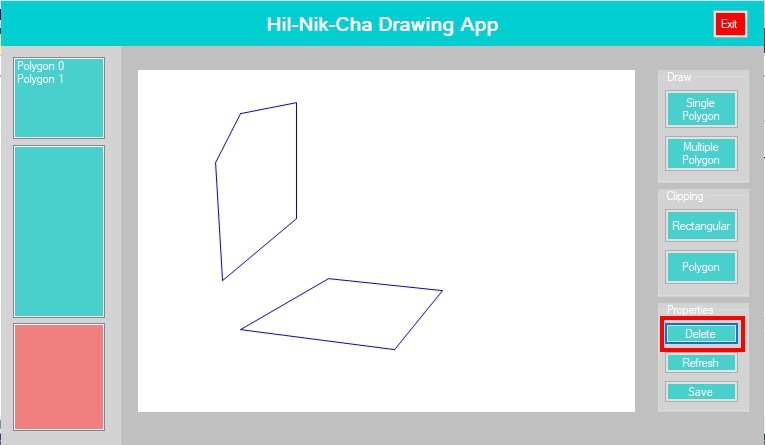
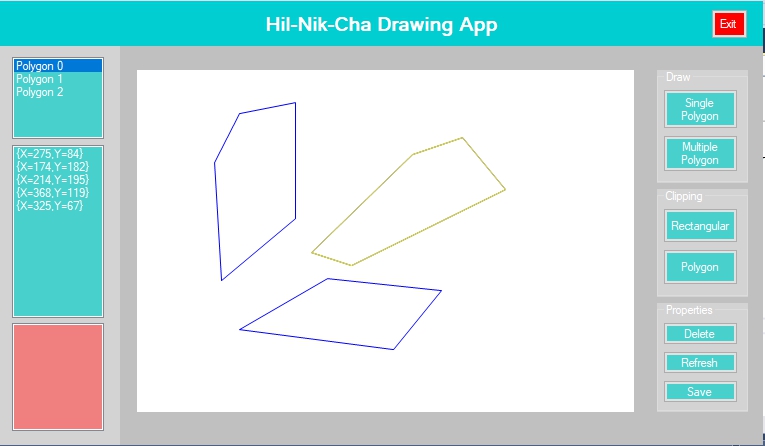
When there’s no polygon drawn yet, the Clipping and Properties buttons will be disabled to prevent user clip or delete something that’s not exist yet.

After the user drawn something on the picture box, All the buttons in Draw, Clipping, and Properties will be enabled.

After there’s a clipping, the Clipping buttons will be disabled.

## Delete the Line

User can delete polygon from Delete button.



# Work Log

The work log is extracted directly from Visual Studio’s Git Log History, which is also available publicly at <https://github.com/ichapredrika/PA12/commits/master>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Author | Date | Time | Commit Message |
| 1 | Nikita | 11/10/2017 | 20:04:59 | Create Basic Theory in Report |
| 2 | Predrika | 13/10/2017 | 19:16:12 | User interface |
| 3 | Predrika | 15/10/2017 | 18:35:37 | Draw multiple polygon |
| 4 | Hilman | 15/10/2017 | 22:13:00 | Add ButtonMenu variable for SPolygon and MPolygon buttons |
| 5 | Hilman | 16/10/2017 | 00:06:28 | Differentiate draw single and multiple polygon |
| 6 | Predrika | 16/10/2017 | 14:53:21 | From Single Polygon button, when we draw the second polygon, the first will disappear |
| 7 | Predrika | 27/10/2017 | 15:22:09 | Delete item on list box |
| 8 | Predrika | 27/10/2017 | 15:43:14 | Adding polygon's point into list box and refresh canvas |
| 9 | Hilman | 30/10/2017 | 01:31:01 | Determine whether the clipping is convex or not |
| 10 | Hilman | 30/10/2017 | 22:30:27 | If polygon clipping is not convex, don’t draw the polygon |
| 11 | Predrika | 31/10/2017 | 11:53:29 | Determine whether the clipping is convex or not (correcting) |
| 12 | Predrika | 31/10/2017 | 12:03:45 | Determine whether the clipping is clockwise or anticlockwise |
| 13 | Predrika | 3/11/2017 | 17:04:53 | Rectangular Clipping (not finished yet) |
| 14 | Hilman | 3/11/2017 | 03:12:29 | Add insidepoint function and clippingpoint funtion |
| 15 | Hilman | 3/11/2017 | 23:02:58 | Draw rectangle clipping with 2 points |
| 16 | Hilman | 3/11/2017 | 23:30:47 | Rect bug fixed |
| 17 | Nikita | 4/11/2017 | 17:20:21 | Correcting the point clipping in the listbox |
| 18 | Hilman | 4/11/2017 | 07:29:42 | Test in and out |
| 19 | Hilman | 6/11/2017 | 11:42:33 | Inside - outside (Fixed) |
| 20 | Hilman | 7/11/2017 | 10:42:26 | linked list using lis of object(not finished yet) |
| 21 | Predrika | 8/11/2017 | 15:53:56 | Better user interface |
| 22 | Hilman | 8/11/2017 | 01:34:21 | Initialized all polygon into linked list after rectangular clipping is drawn |
| 23 | Predrika | 8/11/2017 | 16:57:06 | Disable properties button when there's no polygon |
| 24 | Hilman | 9/11/2017 | 06:15:14 | Linkedlist update |
| 25 | Predrika | 9/11/2017 | 21:52:14 | -Disable clicks on the picture box when the button to draw polygon has not been clicked  - Disable clipping and properties after the user click refresh |
| 26 | Predrika | 10/11/2017 | 20:43:21 | Implementation and edit introduction in report |
| 27 | Predrika | 11/11/2017 | 17:54:33 | -Save to bitmap  -Save into txt |
| 28 | Hilman | 11/11/2017 | 05:32:22 | Clipping (still has bug) |
| 29 | Predrika | 12/11/2017 | 13:23:57 | Refresh (also delete the linked list) |
| 30 | Predrika | 12/11/2017 | 13:46:36 | The second Single Polygon will delete all the previous polygon along with the listbox |
| 31 | Predrika | 12/11/2017 | 19:42:20 | -Evaluation  -Design  -Worklog |
| 32 | Nikita | 12/11/2017 | 22:23:14 | Basic Theory in report |

# Conclusion and Remarks

The program work as expected but has bugs that the developers still don’t know how to fix. The assignment drains all the energy that the developers have.