COMPUTER GRAPHICS AND ANIMATION PROGRAMMING ASSIGNMENT 9: Drawing Polygons – Linked List

Class : IT-3
Arya Sena Wiryady / 001201900110
M. Ichsan Nur Iman / 001201900034
Panji Arlin Saputra / 001201900037

1. Introduction.

- What is the program about?
  - The program is about creating polygons and polylines by using linked lists as the data structure and there are several tools available to edit these polygons and polylines then the result will be drawn into drawing window.
  - We strived to provide a user-friendly GUI (Graphical User Interface) and users can also do creating/ editing just by typing what the user wants to create/edit or it could be said as CLI (Command Line Interface) of this program

- In what language is the program implemented?
  - The program is implemented using C# as the programming language for this project.
  - Why did we select C#? Because it is popular and easy to learn, why C# is easy to learn for us is because in the previous semester we have learnt several programming languages such as C, C++, Java, and Visual Basic. At first we thought of selecting Visual Basic but after getting to know C# more closely, we finally decided to select C# because its syntax is similar to C and C++, it is object-oriented programming that is similar to Java, and it has a Windows Form App that is similar to Visual Basic.

2. Basic theory.

- Explain the basic properties of a polygon and polyline.
  - Polygon is any shape that is at least has 3 vertex and made up from straight lines that is fully closed.
  - Properties of polygon :
    - Coordinate point (x, y)
    - Perimeter of polygon
    - Area of polygon
    - Polygon name
    - Color (R, G, B)
    - Convex or not

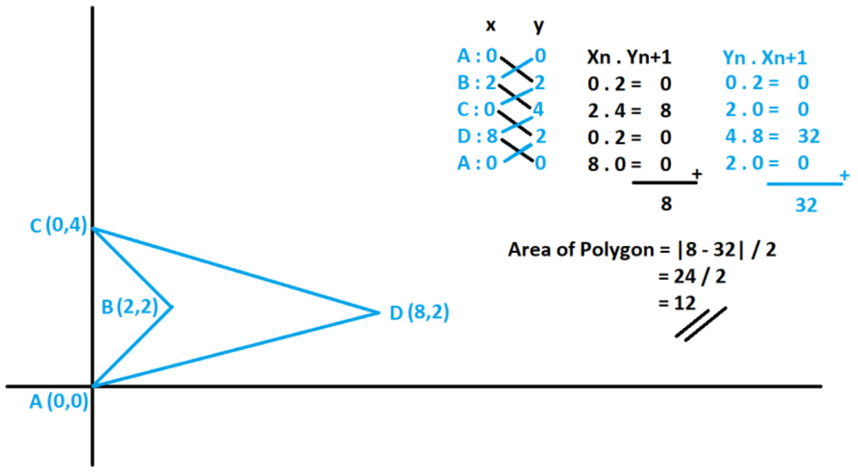
  - Polyline is a set of points where first point is connected to next point by a straight line until sequence of lines is formed unless the first and last points are not connected.

- Properties of polyline :
  - Coordinate point (x, y)
  - Polyline name
  - Length
  - Color (R, G, B)

- Explain how to calculate the perimeter and area of a polygon.
  - Perimeter of polygon: first from coordinate points, we find the vector then calculate the magnitude of vector to find the length of each lines then sum up all the remaining lines
  - Area of polygon: Using the Shoelace Formula.

    Shoelace formula is a mathematical method of calculating the area of a closed plane. This name comes from a calculation method which, when visualized, is similar to the way we tie shoelaces.

    Example :

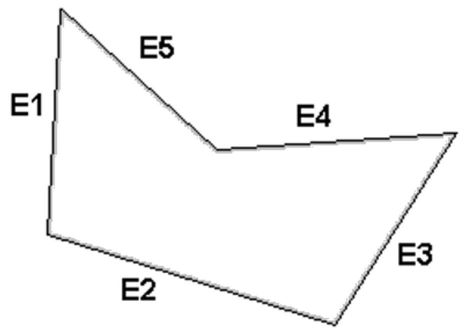


	x	y
A :	0	0
B :	2	2
C :	0	4
D :	8	2
A :	0	0

Xn . Yn+1		Yn . Xn+1	
0 . 2 =	0	0 . 2 =	0
2 . 4 =	8	2 . 0 =	0
0 . 2 =	0	4 . 8 =	32
8 . 0 =	0 +	2 . 0 =	0 +
	8		32

Area of Polygon = |8 - 32| / 2
= 24 / 2
= 12 //

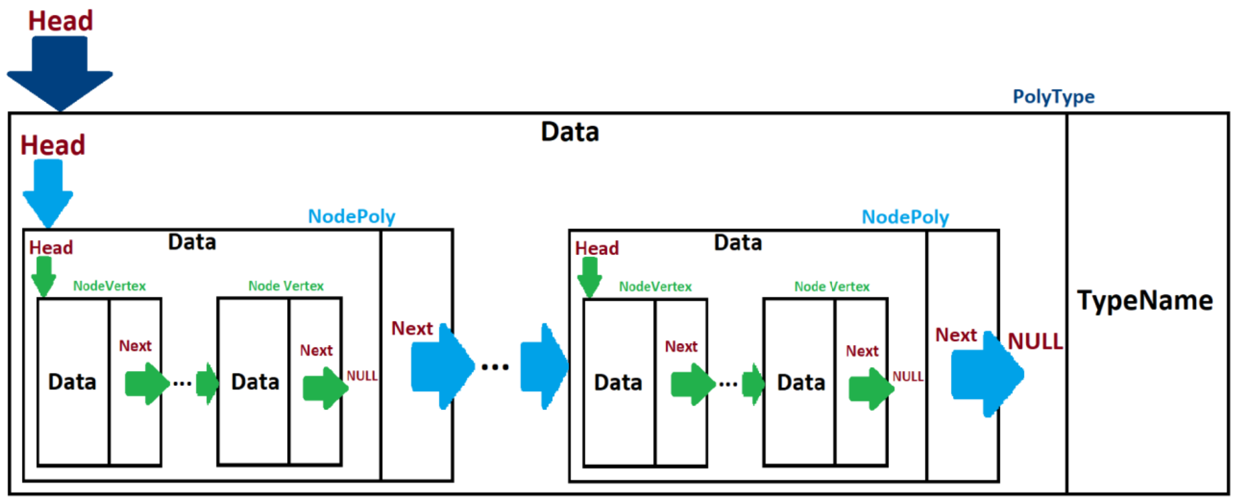
- Explain how to determine whether a polygon is convex.
  - Example :



$E1\times E2 > 0$
$E2\times E3 > 0$
$E3\times E4 > 0$
$E4\times E5 < 0$ then this polygon is not convex.

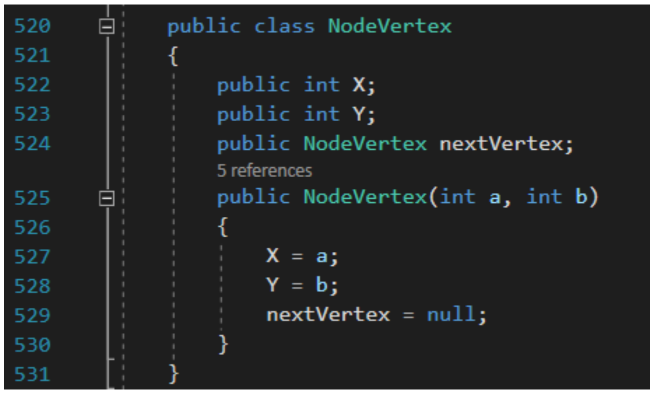
➢ For 2D vectors, $U_z = 0$ and $V_z = 0$
We get:
$S_x = 0$
$S_y = 0$
$S_z = U_x V_y - U_y V_x$
Thus $U \times V$ can be "calculated" as $U_x V_y - U_y V_x$

- Explain how to represent a polygon/polyline using a linked list.

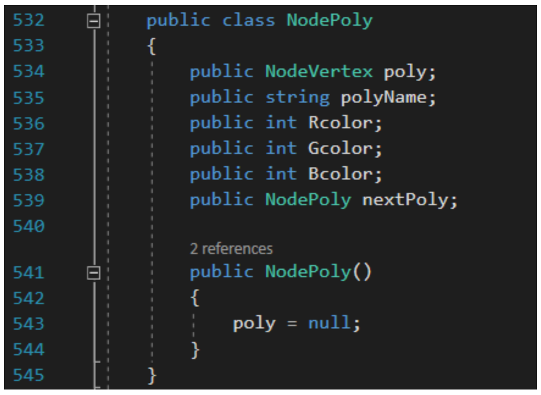


PolyType → Collection of polygons/polylines
NodePoly → A polygon/polyline (Collection of vertices)
NodeVertex → Each node contains a coordinate point x and y

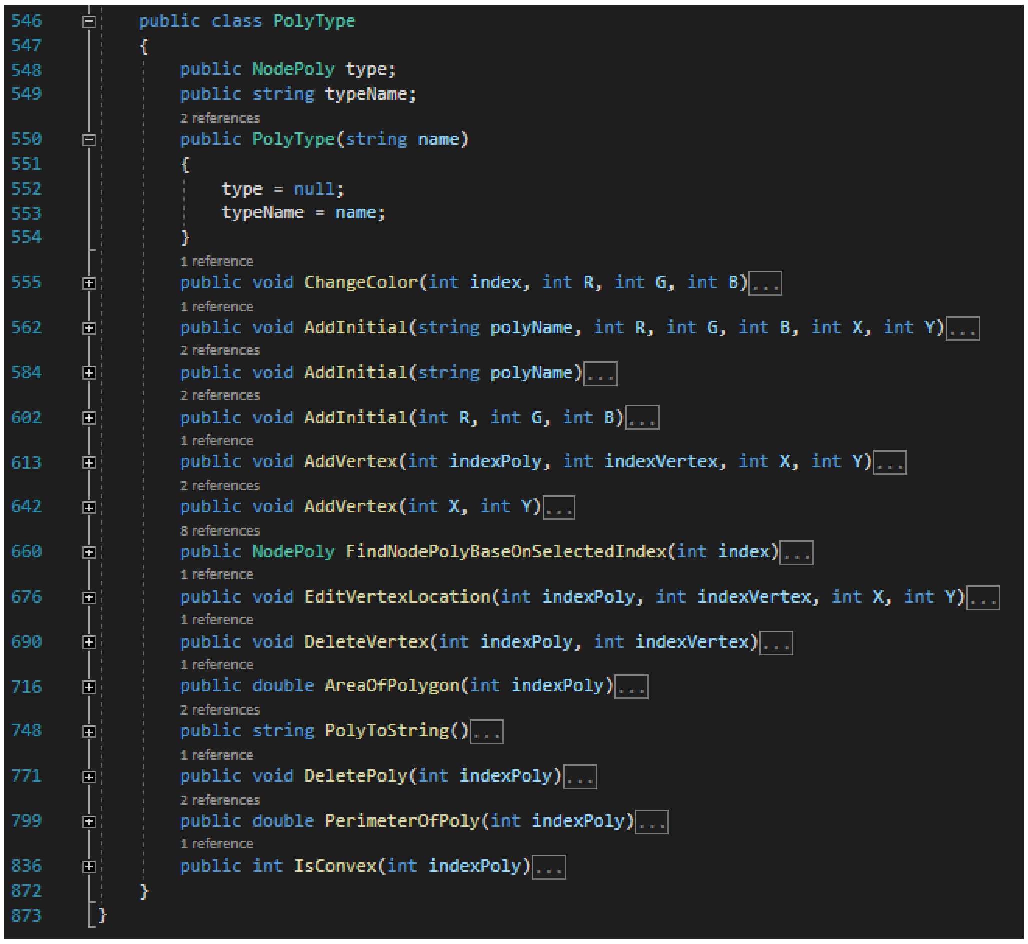
Implementation in code :



```
public class NodeVertex
{
    public int X;
    public int Y;
    public NodeVertex nextVertex;
    5 references
    public NodeVertex(int a, int b)
    {
        X = a;
        Y = b;
        nextVertex = null;
    }
}
```



```
public class NodePoly
{
    public NodeVertex poly;
    public string polyName;
    public int Rcolor;
    public int Gcolor;
    public int Bcolor;
    public NodePoly nextPoly;

    2 references
    public NodePoly()
    {
        poly = null;
    }
}
```



```
    public class PolyType
    {
        public NodePoly type;
        public string typeName;
        2 references
        public PolyType(string name)
        {
            type = null;
            typeName = name;
        }
        1 reference
        public void ChangeColor(int index, int R, int G, int B)...
        1 reference
        public void AddInitial(string polyName, int R, int G, int B, int X, int Y)...
        2 references
        public void AddInitial(string polyName)...
        2 references
        public void AddInitial(int R, int G, int B)...
        1 reference
        public void AddVertex(int indexPoly, int indexVertex, int X, int Y)...
        2 references
        public void AddVertex(int X, int Y)...
        8 references
        public NodePoly FindNodePolyBaseOnSelectedIndex(int index)...
        1 reference
        public void EditVertexLocation(int indexPoly, int indexVertex, int X, int Y)...
        1 reference
        public void DeleteVertex(int indexPoly, int indexVertex)...
        1 reference
        public double AreaOfPolygon(int indexPoly)...
        2 references
        public string PolyToString()...
        1 reference
        public void DeletePoly(int indexPoly)...
        2 references
        public double PerimeterOfPoly(int indexPoly)...
        1 reference
        public int IsConvex(int indexPoly)...
    }
}
```

- Explain vertex processing on a polygon.
    - What is vertex processing?
        - ➢ Vertex processing is performing operations based on the vertex
    - Give some examples of vertex processing.
        - ➢ Calculate area of polygon
        - ➢ Delete a vertex
        - ➢ Change location of a vertex
        - ➢ Print coordinate point

- Write the general pseudocode for vertex processing.

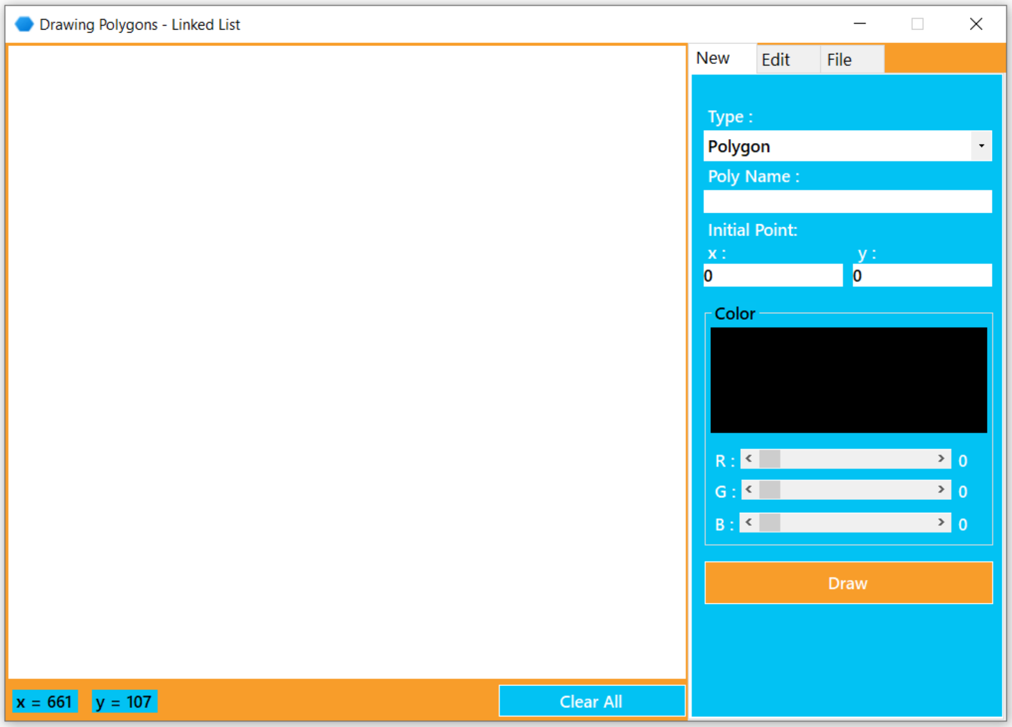
  -
    ```
    for i = 0 to vertex(N-1)
        process(allVertex)
    ```

- Explain edge processing on a polygon.
  - What is edge processing?
    - Edge processing is performing operations based on the edge
  - Give some examples of edge processing.
    - Calculate perimeter of polygon
    - Determine the polygon is convex or not
    - Change polygons line color
  - Write the general pseudocode for edge processing.

    -
      ```
      for i = 0 to vertex(N-2)
          process vertex i, vertex i+1
      ```

3. Implementation

- Explain the main interface of the program.

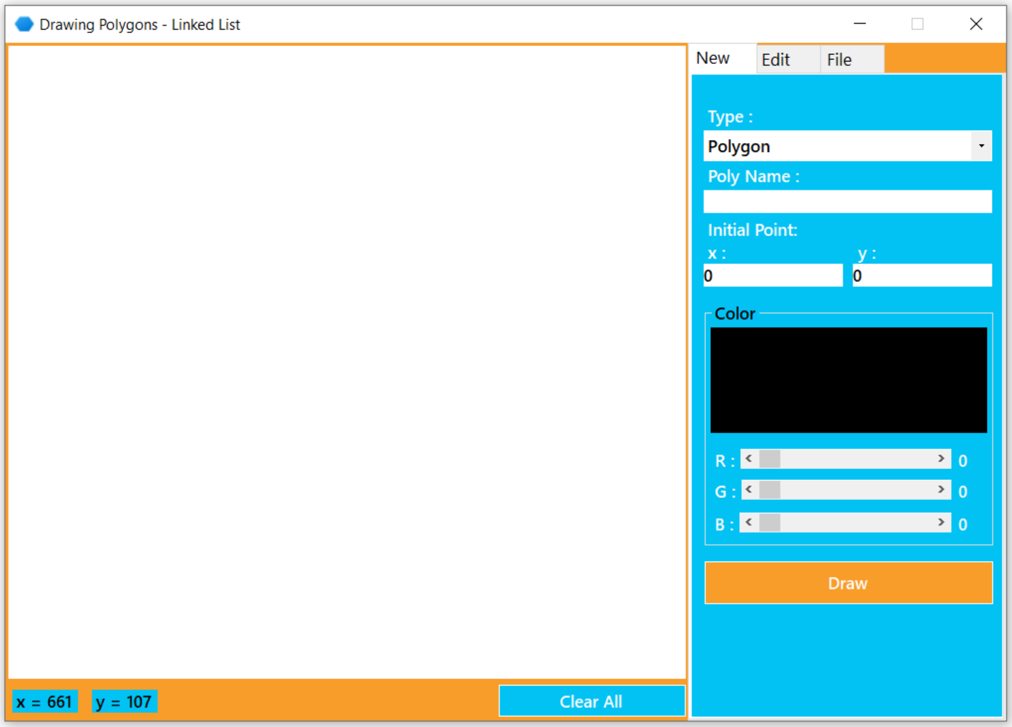


This is the initial interface of this program, here there is a drawing window and several tools where this tab menu “New” is the starting place where the user can create a polygon or polyline.

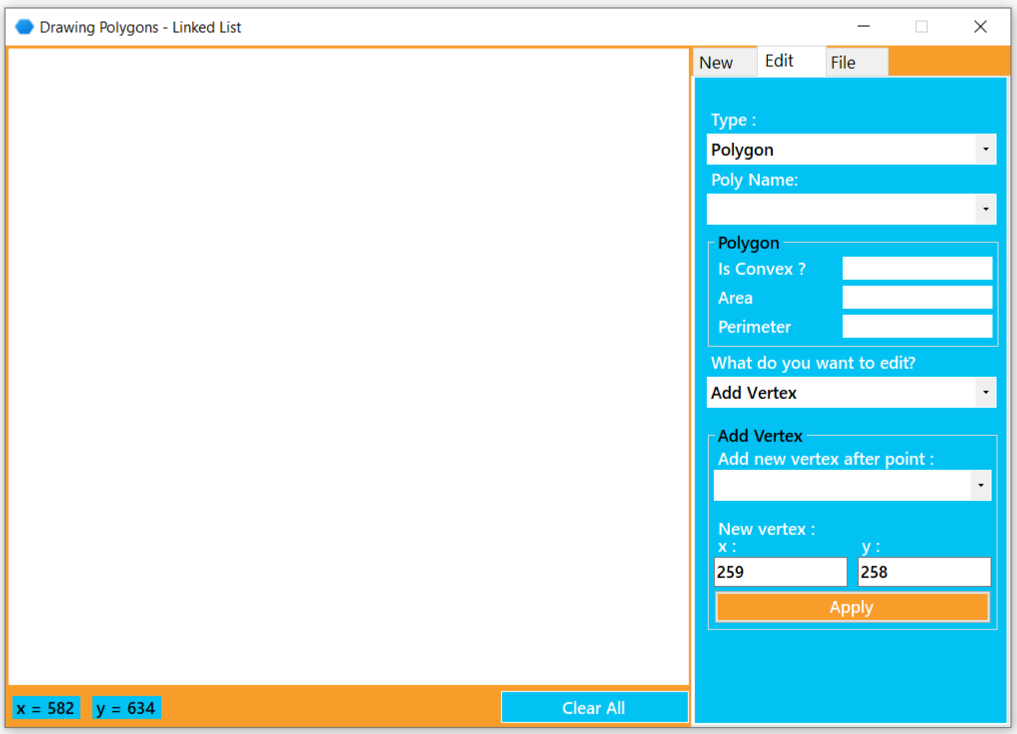
In color selection, we decided to choose bright colors as the interface color for this program, such as blue and orange

To make it easier for users to determine coordinate points, we create 2 mouse event handlers, first when the user's mouse move (MouseMove) above the drawing window, the x and y labels under the drawing window will continue to change and display the x and y values of the mouse location. and the second is when the user's mouse clicks (MouseDown) on the drawing window, the initial points textbox of x and y, will note the x and y locations of the mouse.

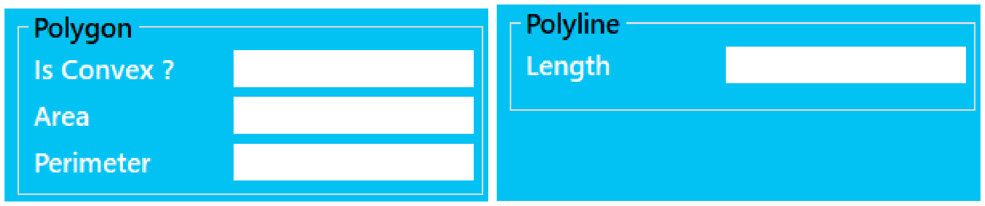
- Explain every feature in the program and how to use them.



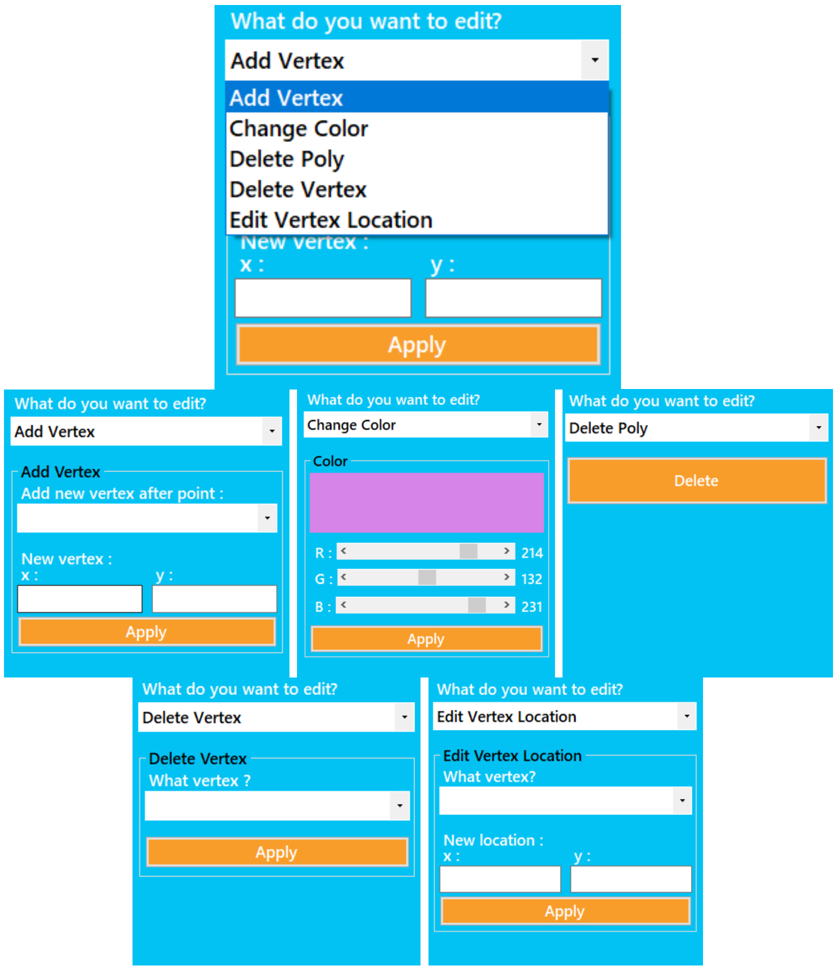
First, the "New" menu tab, this menu tab is the starting place where the user can create a polygon or polyline, there are several inputs that the user needs to enter before creating a polygon or polyline i.e. type (ComboBox), poly name (TextBox), first coordinate point x and point y of polygon/polyline (TextBox), and RGB color (HScrollBar). There is also a "Draw" button to create a starting point for a polygon / polyline.



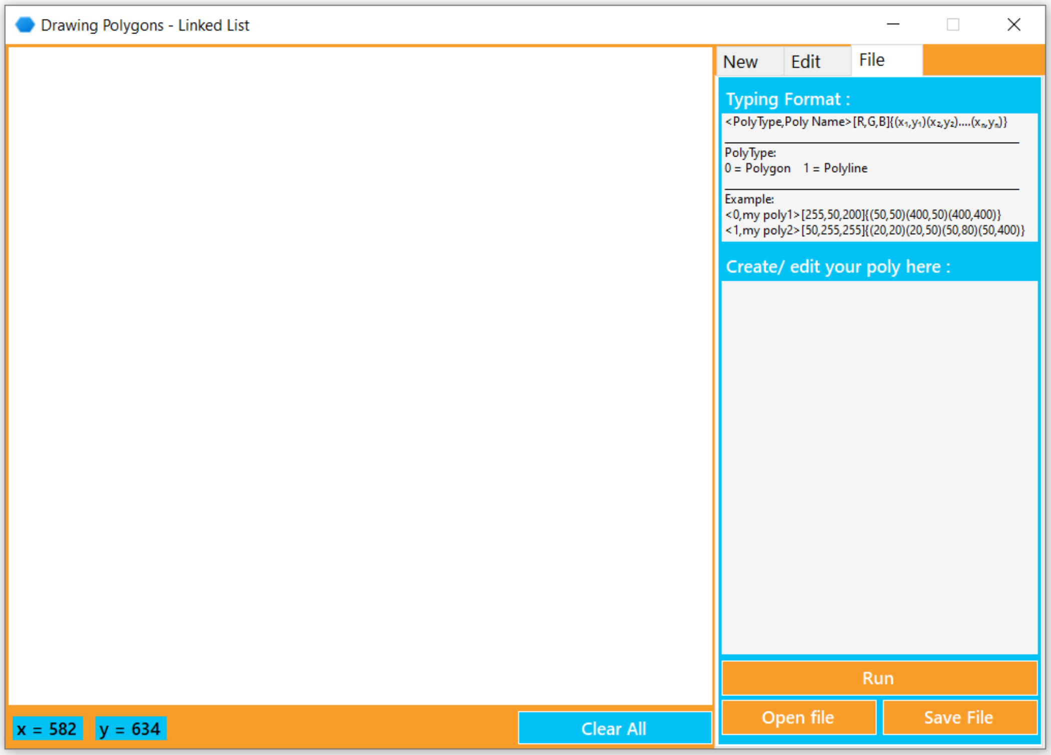
Second, the “Edit” menu tab, this menu tab will be displayed immediately when the user has finished creating the starting point of a polygon / polyline. On this menu tab, at the top there is a Type (ComboBox) which can be used by the user to choose between the polygon or polyline that you want to edit then a Poly name (ComboBox) that is used to select the polygon/polyline name that the user wants to edit.



at the middle, this is a place to display some the output of a polygon / polyline.



at the bottom, the user can choose what the user wants to edit, there are several editing menus as shown in the image above



```
<PolyType,Poly Name>[R,G,B]{(x1,y1)(x2,y2)....(xn,yn)}

PolyType:
0 = Polygon   1 = Polyline

Example:
<0,my poly1>[255,50,200]{(50,50)(400,50)(400,400)}
<1,my poly2>[50,255,255]{(20,20)(20,50)(50,80)(50,400)}
```

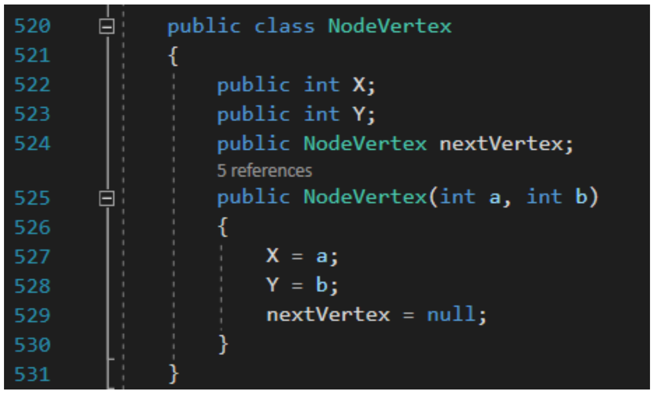
Third, the "File" menu tab, here there is a CLI (Command Line Interface) where users can create or edit polygons / polylines without having to go to the "New" and "Edit" menu tabs by following the typing format rules provided. At the bottom there is a "Run" button to run all commands that have been typed by the user, the "Open File" button to load polygons/polylines from
a file, and the "Save File" button to save polygons/polylines into a file.

Also, at the bottom of the drawing window, there is a "Clear All" button to erase all polygons/polylines and clear the screen.

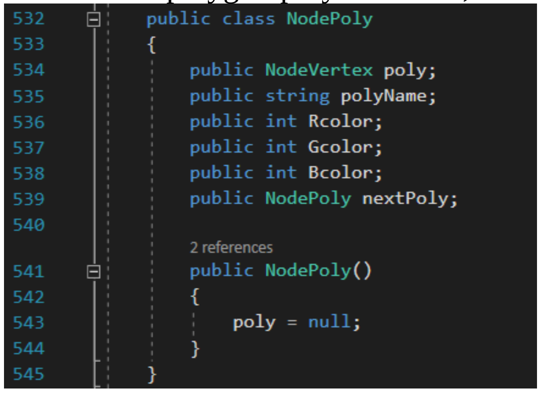
4. Design

- Explain the main data structures (if any) used in the program.
  In this program, we use a linked list as the data structure to draw a polygon / polyline. Linked list is a data structure consisting of a sequence of nodes

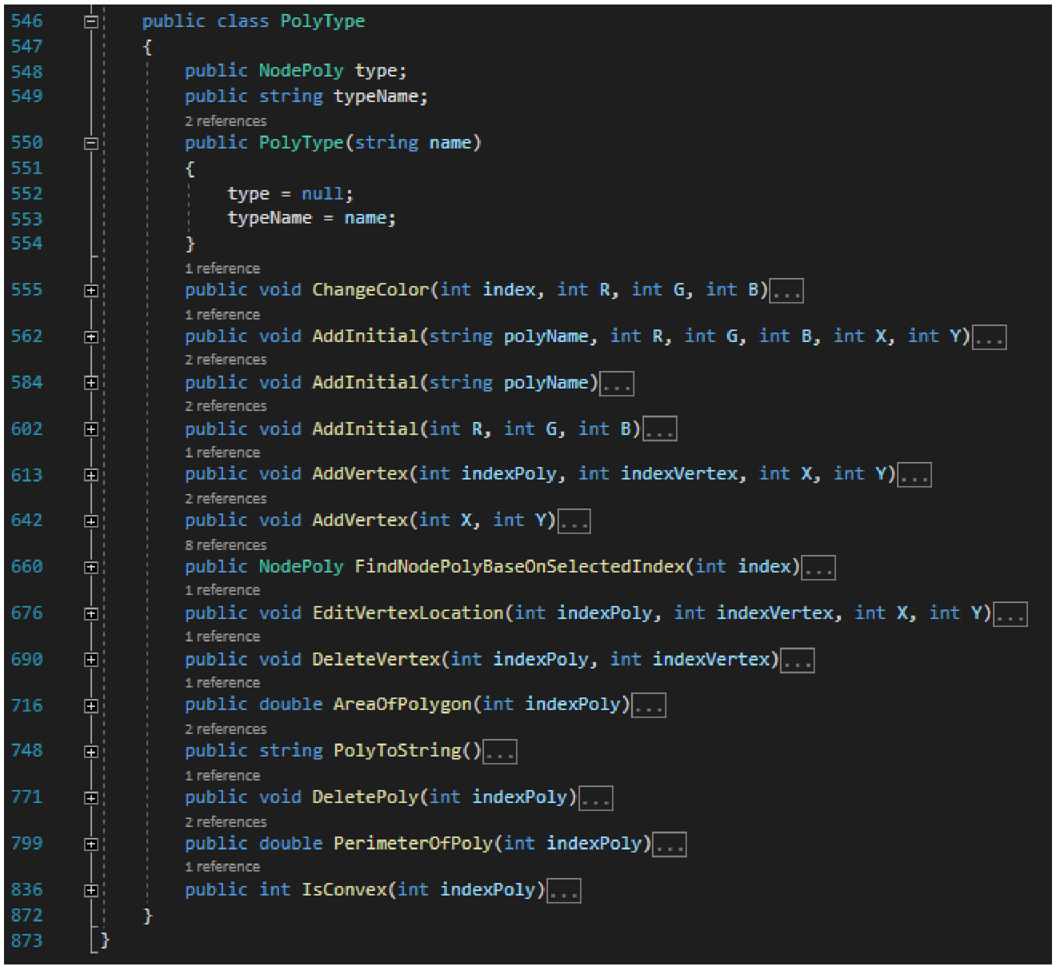
  - How are the points represented in the program?
    Points in the program are represented by using the x location and y location of the pixels on the monitor screen, then the value is assigned to a variable inside a linked list node (NodeVertex).


```
public class NodeVertex
{
    public int X;
    public int Y;
    public NodeVertex nextVertex;
    5 references
    public NodeVertex(int a, int b)
    {
        X = a;
        Y = b;
        nextVertex = null;
    }
}
```

- How are the polygons/polylines represented in the program?
  Each polygon/polyline is represented in a linked list node (NodePoly) which contains some data such as polygon/polyline name, color, and vertex.

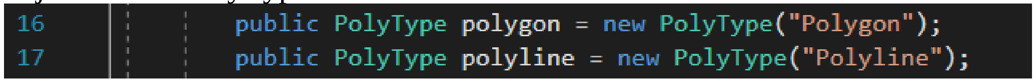

```
public class NodePoly
{
    public NodeVertex poly;
    public string polyName;
    public int Rcolor;
    public int Gcolor;
    public int Bcolor;
    public NodePoly nextPoly;

    2 references
    public NodePoly()
    {
        poly = null;
    }
}
```

- How is the collection of polygons/polylines represented in the program?
  Collection of polygons/polylines is represented in a node (PolyType) which contains a linked list of polygons / polylines in it as well as several methods to perform various operations on polygons/polylines.



```
public class PolyType
{
    public NodePoly type;
    public string typeName;
    2 references
    public PolyType(string name)
    {
        type = null;
        typeName = name;
    }
    1 reference
    public void ChangeColor(int index, int R, int G, int B)...
    1 reference
    public void AddInitial(string polyName, int R, int G, int B, int X, int Y)...
    2 references
    public void AddInitial(string polyName)...
    2 references
    public void AddInitial(int R, int G, int B)...
    1 reference
    public void AddVertex(int indexPoly, int indexVertex, int X, int Y)...
    2 references
    public void AddVertex(int X, int Y)...
    8 references
    public NodePoly FindNodePolyBaseOnSelectedIndex(int index)...
    1 reference
    public void EditVertexLocation(int indexPoly, int indexVertex, int X, int Y)...
    1 reference
    public void DeleteVertex(int indexPoly, int indexVertex)...
    1 reference
    public double AreaOfPolygon(int indexPoly)...
    2 references
    public string PolyToString()...
    1 reference
    public void DeletePoly(int indexPoly)...
    2 references
    public double PerimeterOfPoly(int indexPoly)...
    1 reference
    public int IsConvex(int indexPoly)...
}
}
```

- Explain the main/global variables used in the program.
  We have only 2 global variables with user-defined data types (Objects) that are used to assign objects of the PolyType class where there is linked list of linked list inside this class.

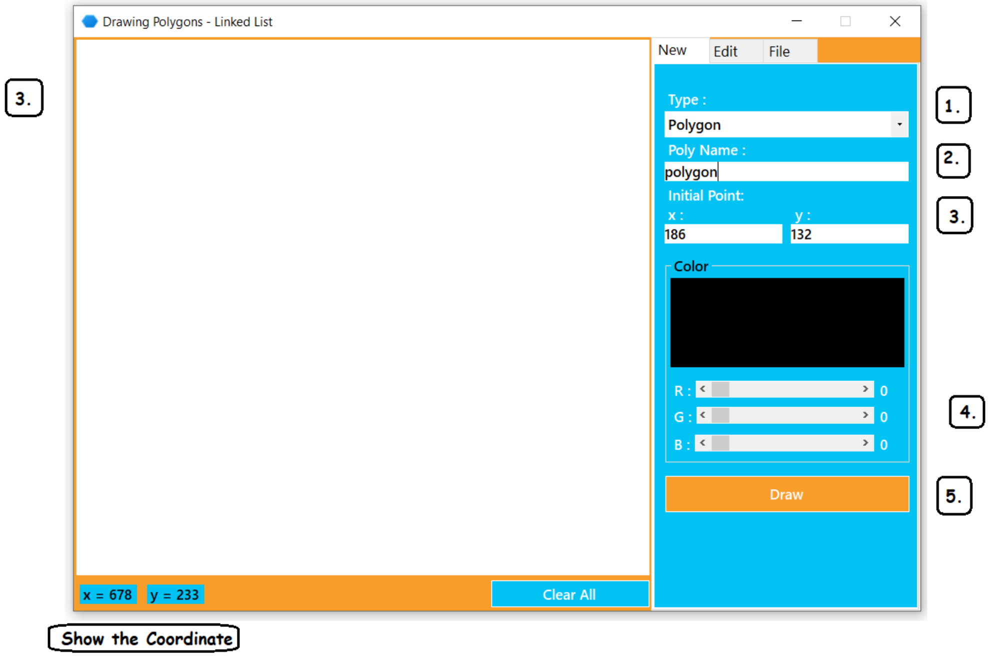


```
public PolyType polygon = new PolyType("Polygon");
public PolyType polyline = new PolyType("Polyline");
```

- Explain how the bonuses (if done) are implemented.
  - ➢ Calculate the length of polyline
    The way to calculating the length of a polyline is almost the same as calculating the perimeter of a polygon, the difference is that the first point on the polyline is not connected to the last point on the polyline.

5. Evaluation

- Evaluation of the main features (drawing polygons and polylines). Try the following test cases:
    - Add a polygon/polyline.



Type : Polygon
Poly Name : polygon
Initial Point: x : 186 y : 132
x = 678 y = 233

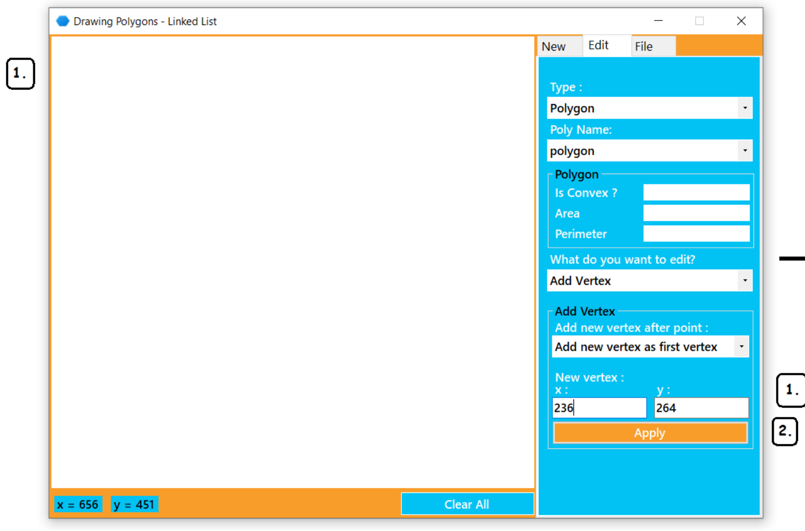
1. Select the type between polygon or polyline

2. Insert name for the polygon or polyline

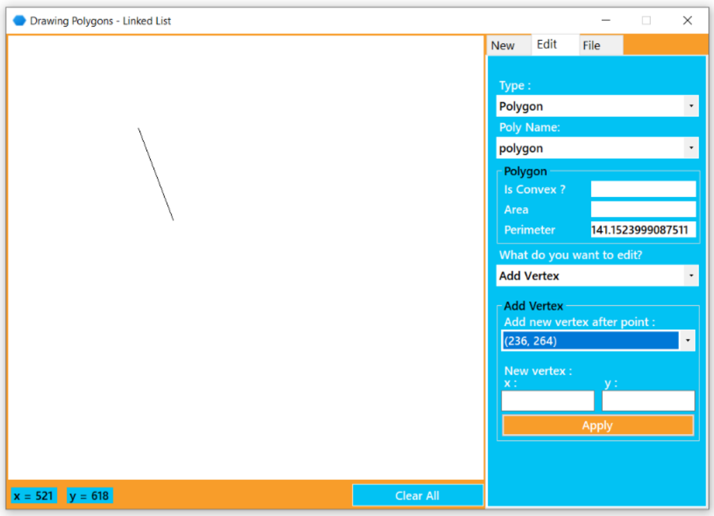
3. Input the first coordinate (either by clicking the canvas area or just simply input the x & y point) The coordinate is shown on bottom left

4. Set the color for the polygon or the poly line

5. Click draw when finish



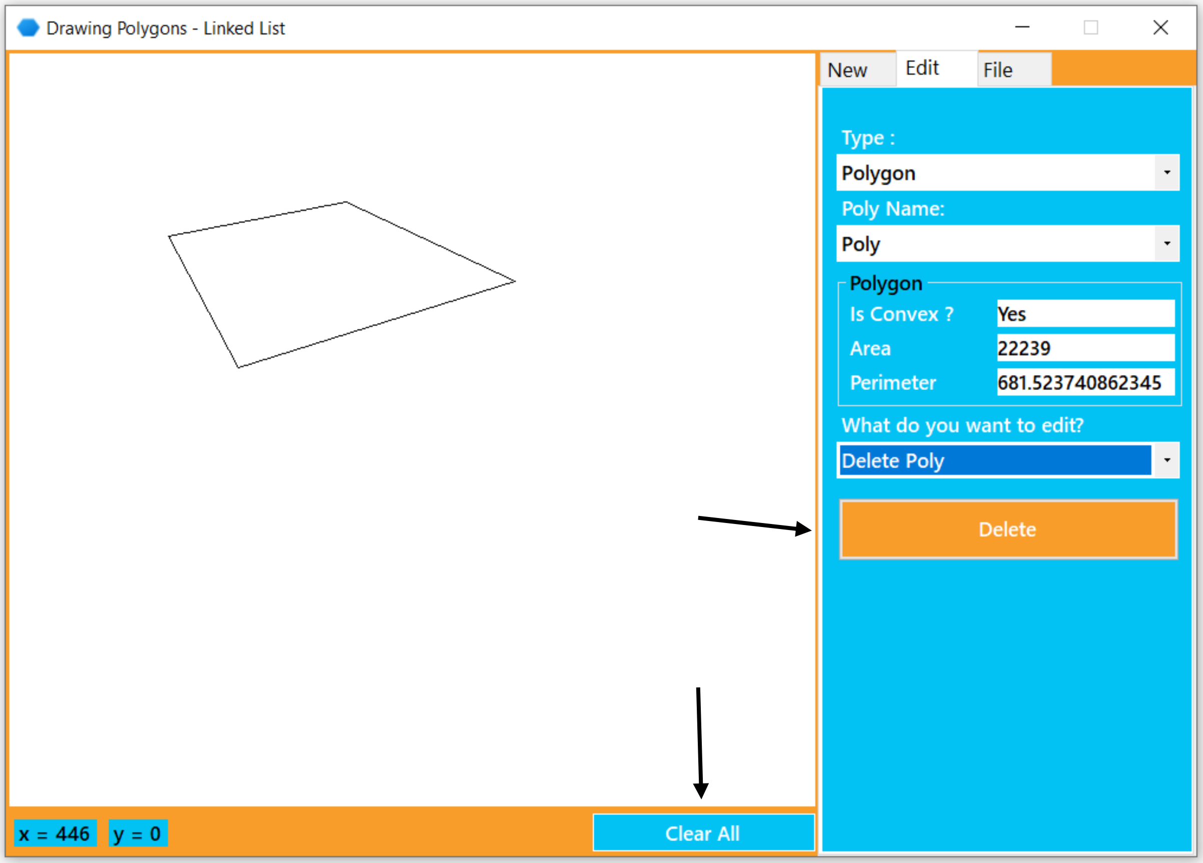
Type : Polygon
Poly Name: polygon
What do you want to edit? Add Vertex
Add new vertex after point : Add new vertex as first vertex
New vertex : x : 236 y : 264
x = 656 y = 451



Type : Polygon
Poly Name: polygon
Perimeter 141.1523999087511
What do you want to edit? Add Vertex
Add new vertex after point : (236, 264)
x = 521 y = 618

1. Input the second coordinate either by typing it in or simply just click the canvas
2. Click apply when finish
3. Repeat the step to desired polygon or poly line

- Delete a polygon/polyline.


Drawing Polygons - Linked List

Type : Polygon

Poly Name: Poly

Polygon

Is Convex ? Yes

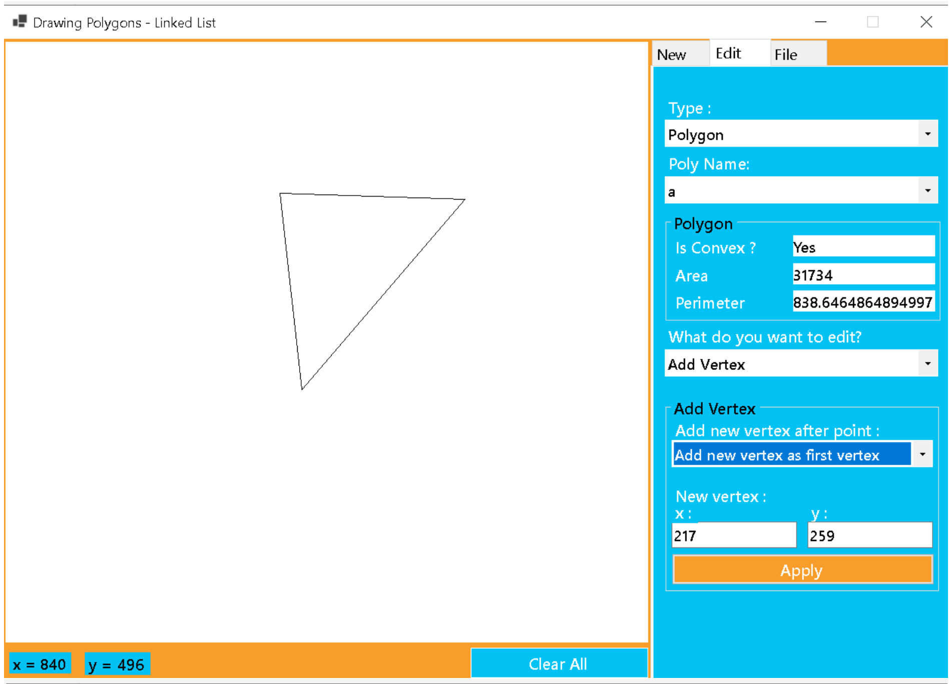
Area 22239

Perimeter 681.523740862345

What do you want to edit? Delete Poly

To delete the polygon or the polyline, there is 2 ways. First by clicking "Clear all button" for delete all (multiple) polygons/polylines. Second is in edit section, there is an option on "What do you want to edit?" select "Delete Poly" then simply click the "Delete" button, this way will only delete a selected polygon/polyline.

- Adding a vertex to an existing polygon/polyline.
  - Add a vertex as the first vertex.



Drawing Polygons - Linked List

New | Edit | File

Type : Polygon

Poly Name: a

Polygon
Is Convex ? Yes
Area 31734
Perimeter 838.6464864894997

What do you want to edit? Add Vertex

Add Vertex
Add new vertex after point : Add new vertex as first vertex

New vertex :
x : 217
y : 259

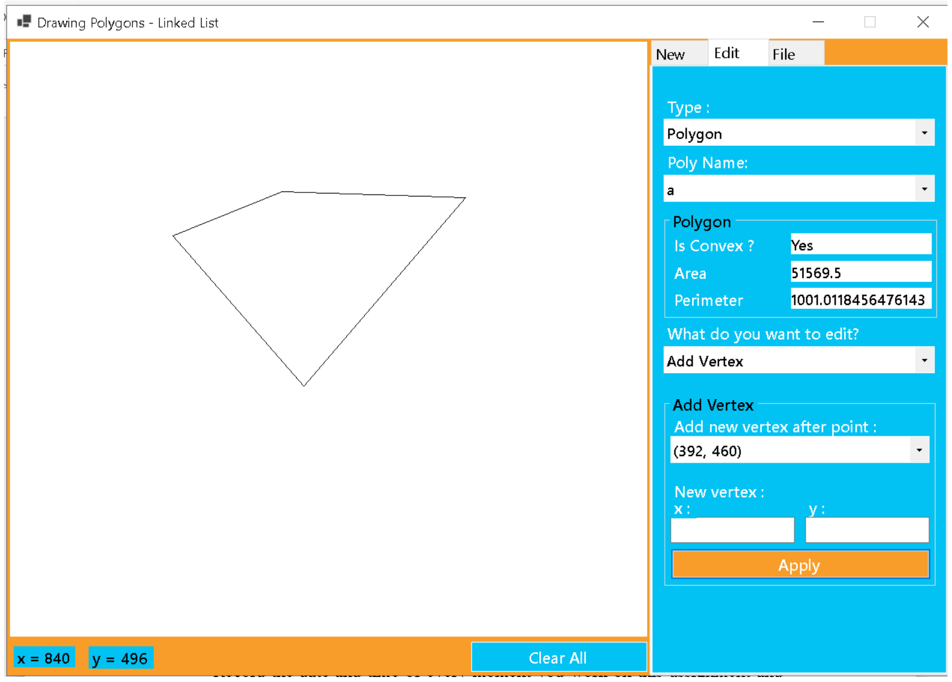
Apply

x = 840 y = 496 Clear All

Set to Add Vertex

Select where do you want to add new vertex

Select the new vertex's coordinate



Drawing Polygons - Linked List

New | Edit | File

Type : Polygon

Poly Name: a

Polygon
Is Convex ? Yes
Area 51569.5
Perimeter 1001.0118456476143

What do you want to edit? Add Vertex

Add Vertex
Add new vertex after point : (392, 460)

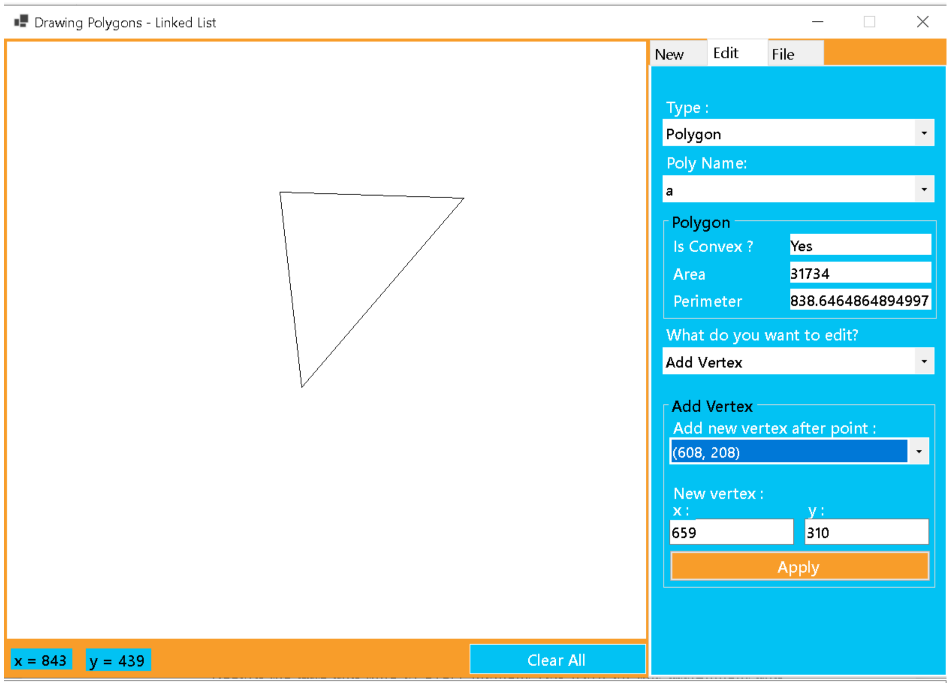
New vertex :
x :
y :

Apply

x = 840 y = 496 Clear All

This is the result

- Add a vertex as a “middle” vertex.



Type : Polygon

Poly Name: a

Polygon

Is Convex ? Yes

Area 31734

Perimeter 838.6464864894997

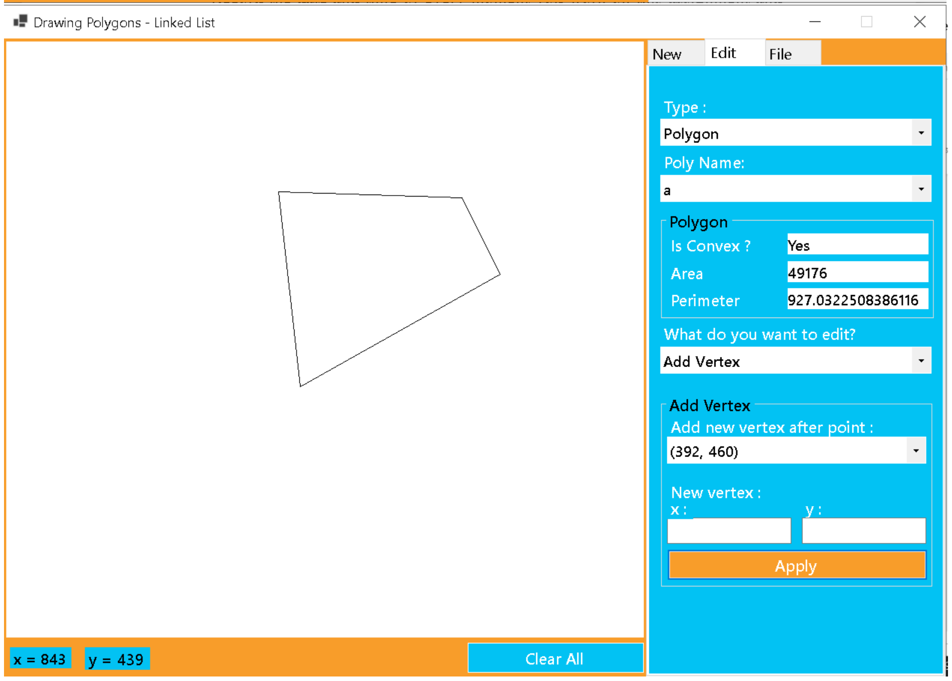
What do you want to edit? Add Vertex

Add Vertex

Add new vertex after point : (608, 208)

New vertex : x : 659 y : 310

x = 843 y = 439



Type : Polygon

Poly Name: a

Polygon

Is Convex ? Yes

Area 49176

Perimeter 927.0322508386116

What do you want to edit? Add Vertex

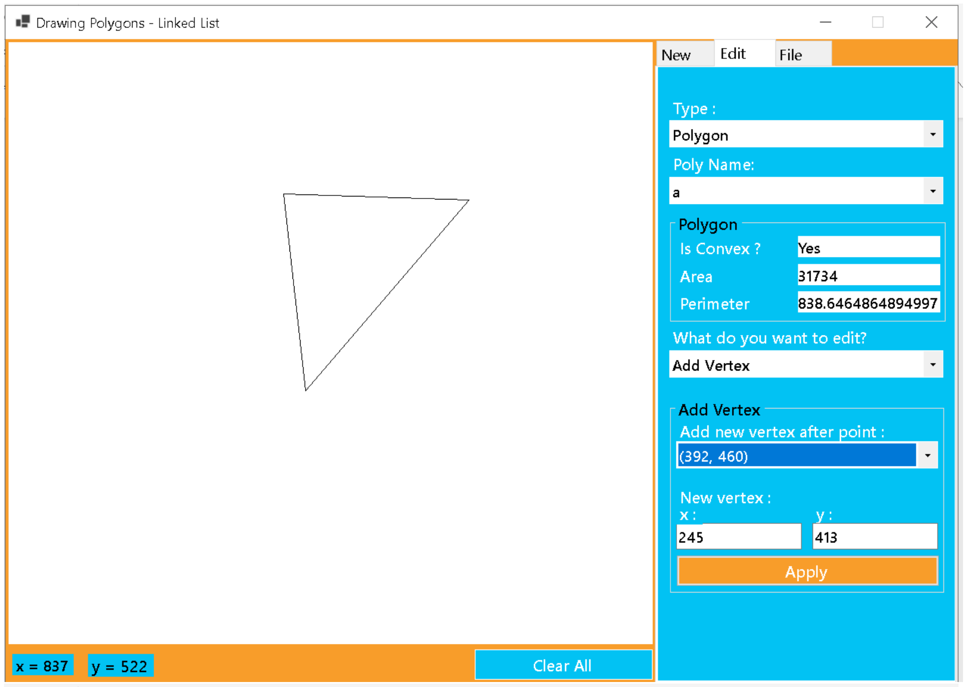
Add Vertex

Add new vertex after point : (392, 460)

New vertex : x : y :

x = 843 y = 439

- Add a vertex as the last vertex.



Type : Polygon

Poly Name: a

Polygon

Is Convex ? Yes

Area 31734

Perimeter 838.6464864894997

What do you want to edit? Add Vertex

Add Vertex

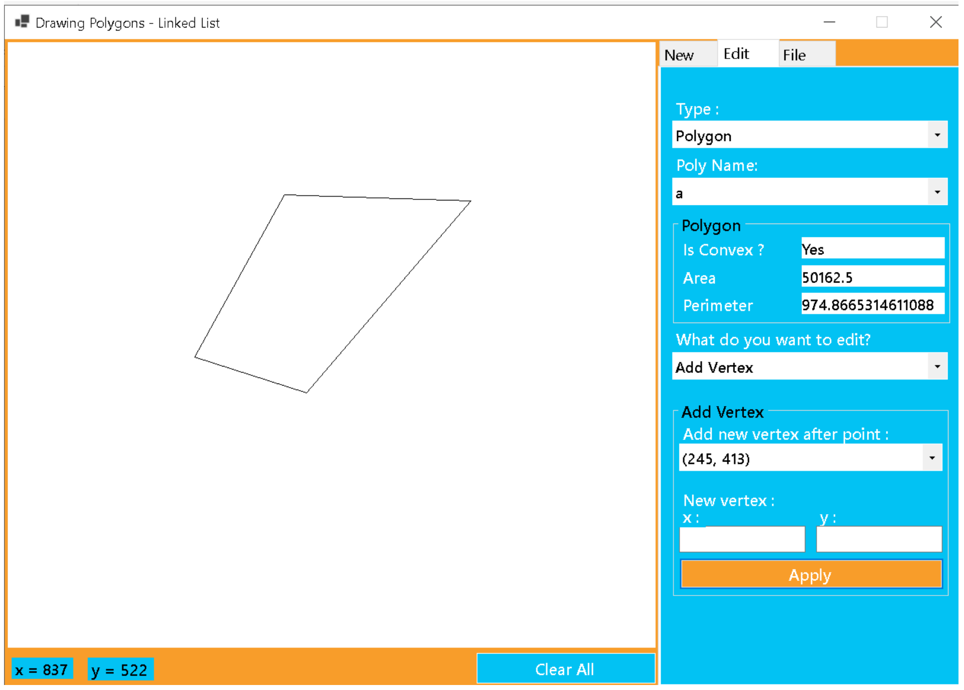
Add new vertex after point : (392, 460)

New vertex :

x : 245

y : 413

x = 837 y = 522



Type : Polygon

Poly Name: a

Polygon

Is Convex ? Yes

Area 50162.5

Perimeter 974.8665314611088

What do you want to edit? Add Vertex

Add Vertex

Add new vertex after point : (245, 413)

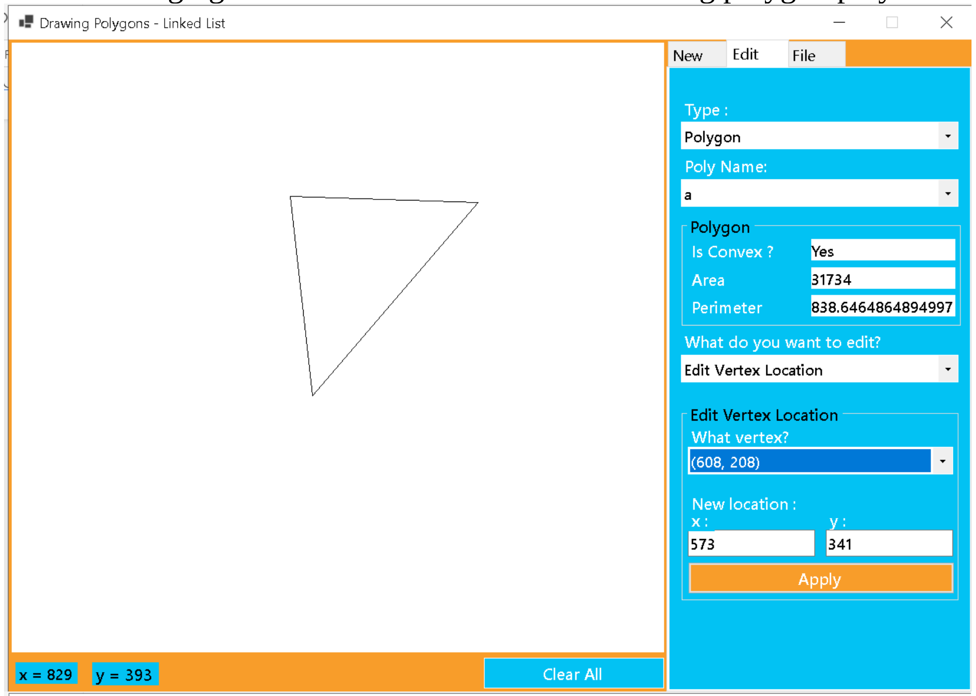
New vertex :

x :

y :

x = 837 y = 522

- Changing the location of a vertex on an existing polygon/polyline.

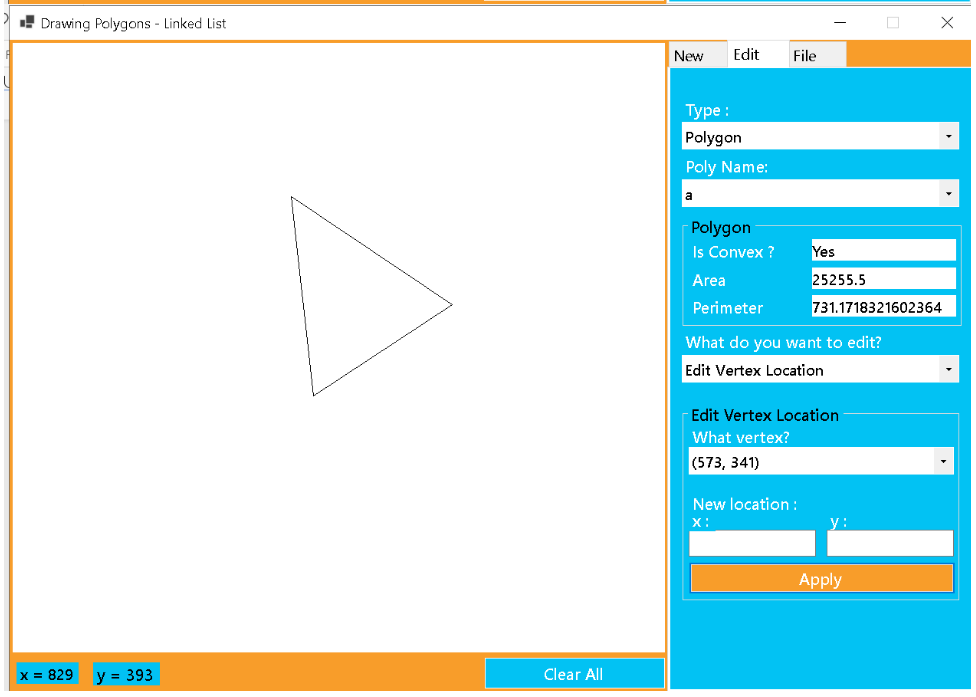


Type : Polygon
Poly Name: a
Polygon
Is Convex ? Yes
Area 31734
Perimeter 838.6464864894997
What do you want to edit? Edit Vertex Location
Edit Vertex Location
What vertex? (608, 208)
New location :
x : 573
y : 341
x = 829 y = 393

Set to Edit Vertex Location

Select which vertex to edit

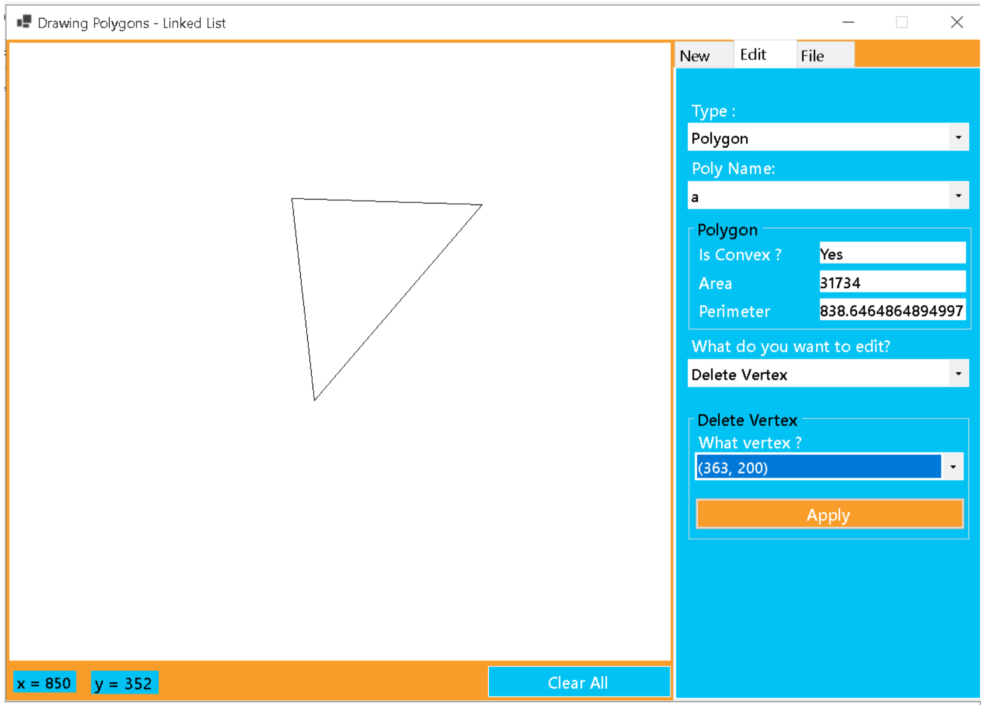
Set the new coordinate



Type : Polygon
Poly Name: a
Polygon
Is Convex ? Yes
Area 25255.5
Perimeter 731.1718321602364
What do you want to edit? Edit Vertex Location
Edit Vertex Location
What vertex? (573, 341)
New location :
x :
y :
x = 829 y = 393

This is the result

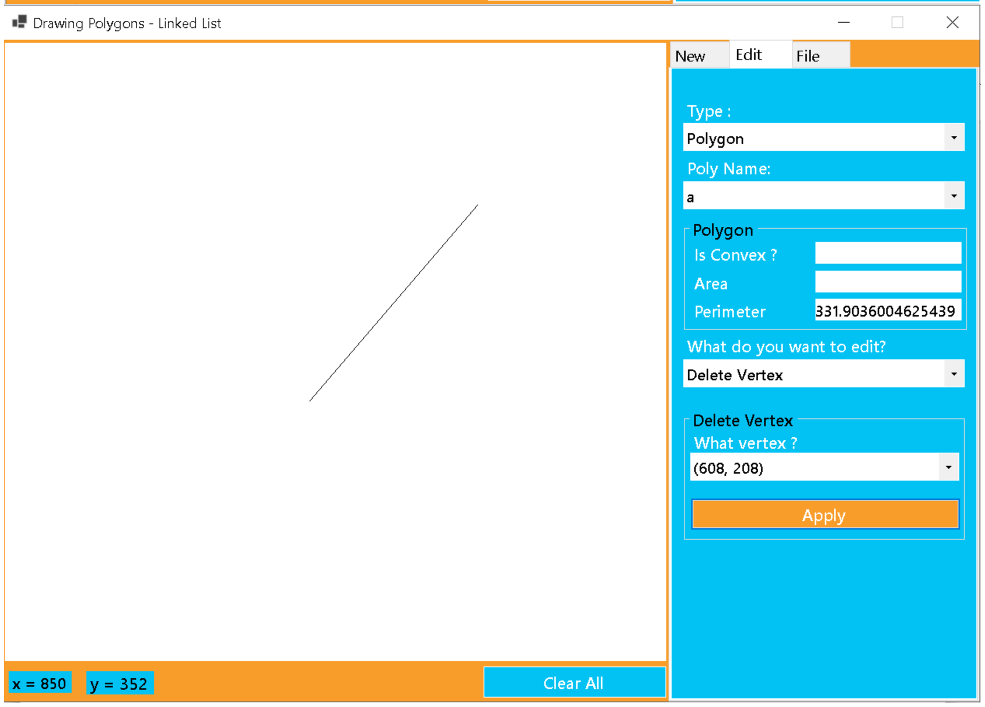
- Deleting a vertex from a polygon.
    - Delete the first vertex.



Type : Polygon
Poly Name: a
Polygon
Is Convex ? Yes
Area 31734
Perimeter 838.6464864894997
What do you want to edit? Delete Vertex
Delete Vertex
What vertex ? (363, 200)
x = 850 y = 352

Set to Delete Vertex

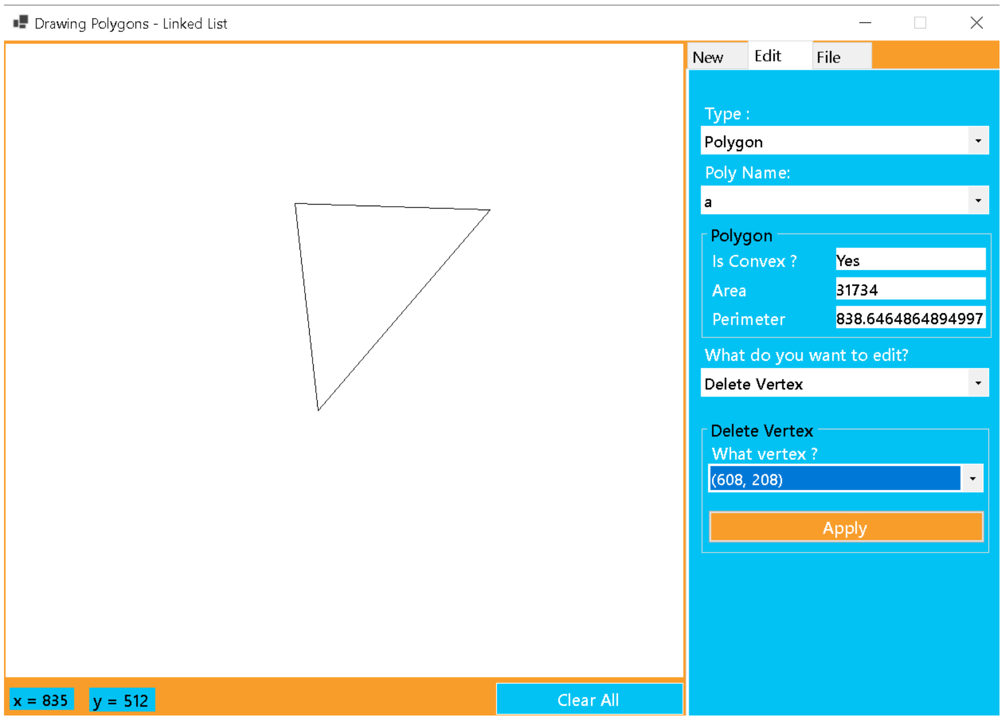
Select which vertex you want to delete



Type : Polygon
Poly Name: a
Polygon
Is Convex ?
Area
Perimeter 331.9036004625439
What do you want to edit? Delete Vertex
Delete Vertex
What vertex ? (608, 208)
x = 850 y = 352

This is the result

- Delete a “middle” vertex.



Drawing Polygons - Linked List

New Edit File

Type : Polygon

Poly Name: a

Polygon

Is Convex ? Yes

Area 31734

Perimeter 838.6464864894997

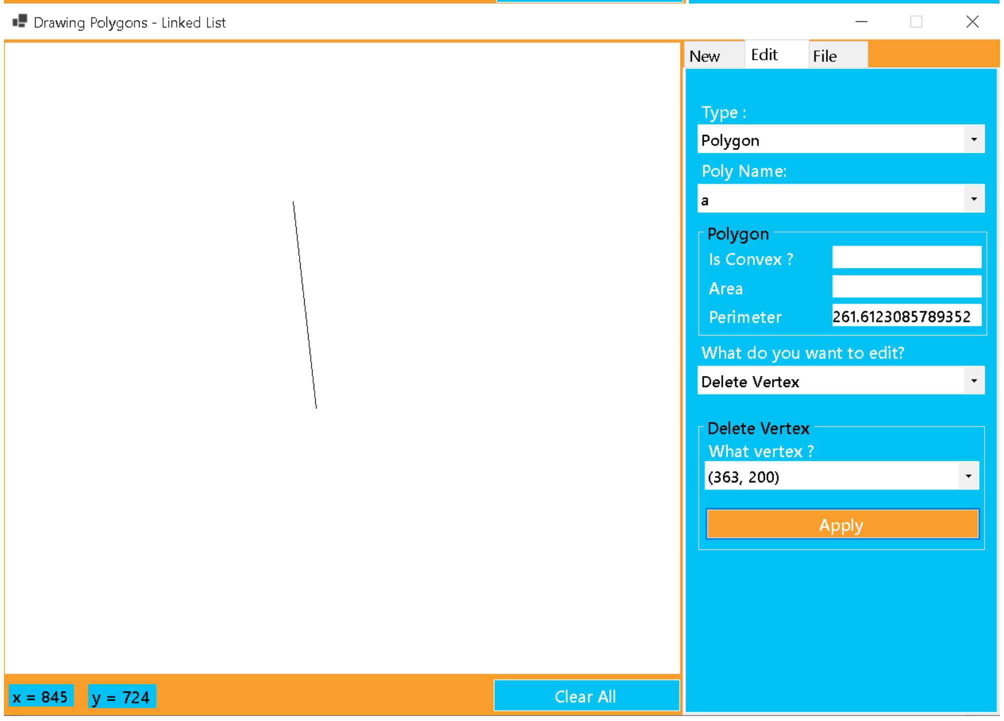
What do you want to edit? Delete Vertex

Delete Vertex

What vertex ? (608, 208)

Apply

x = 835 y = 512 Clear All



Drawing Polygons - Linked List

New Edit File

Type : Polygon

Poly Name: a

Polygon

Is Convex ?

Area

Perimeter 261.6123085789352

What do you want to edit? Delete Vertex

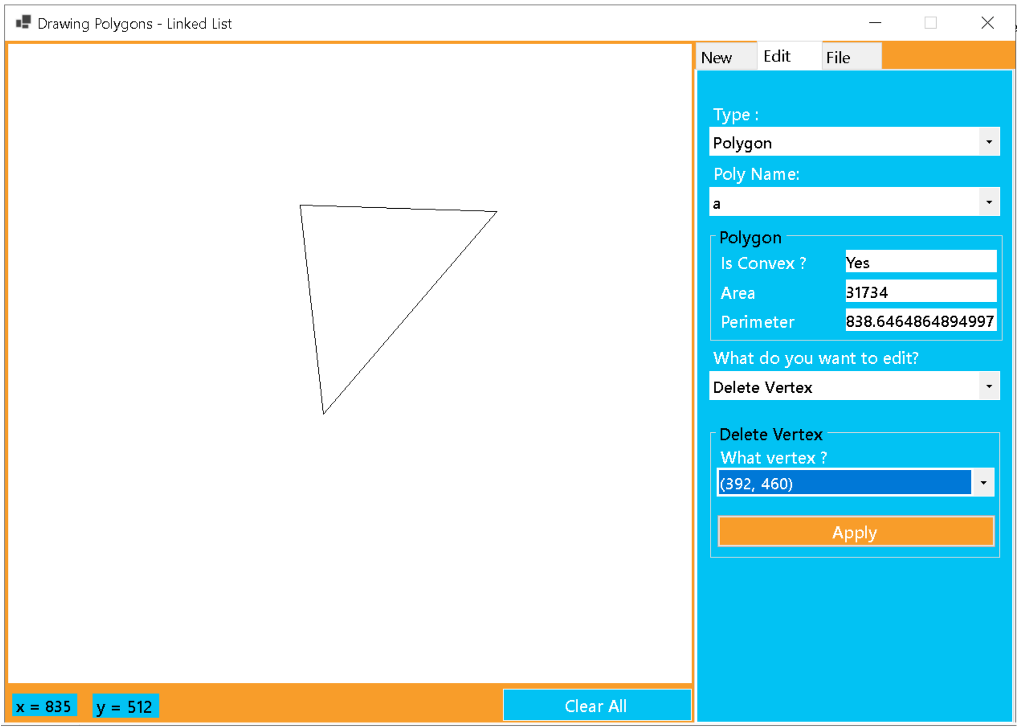
Delete Vertex

What vertex ? (363, 200)

Apply

x = 845 y = 724 Clear All

- Delete the last vertex.



Drawing Polygons - Linked List

Type : Polygon

Poly Name: a

Polygon

Is Convex ? Yes

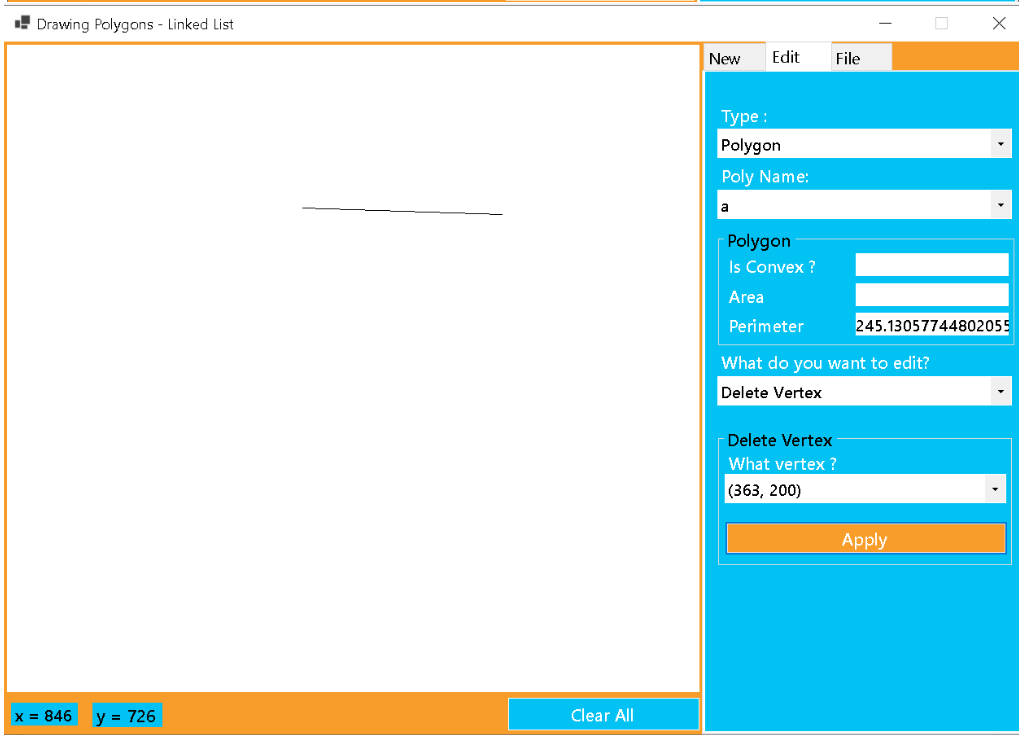
Area 31734

Perimeter 838.6464864894997

What do you want to edit? Delete Vertex

Delete Vertex

What vertex ? (392, 460)



Drawing Polygons - Linked List

Type : Polygon

Poly Name: a

Polygon

Is Convex ?

Area

Perimeter 245.13057744802055

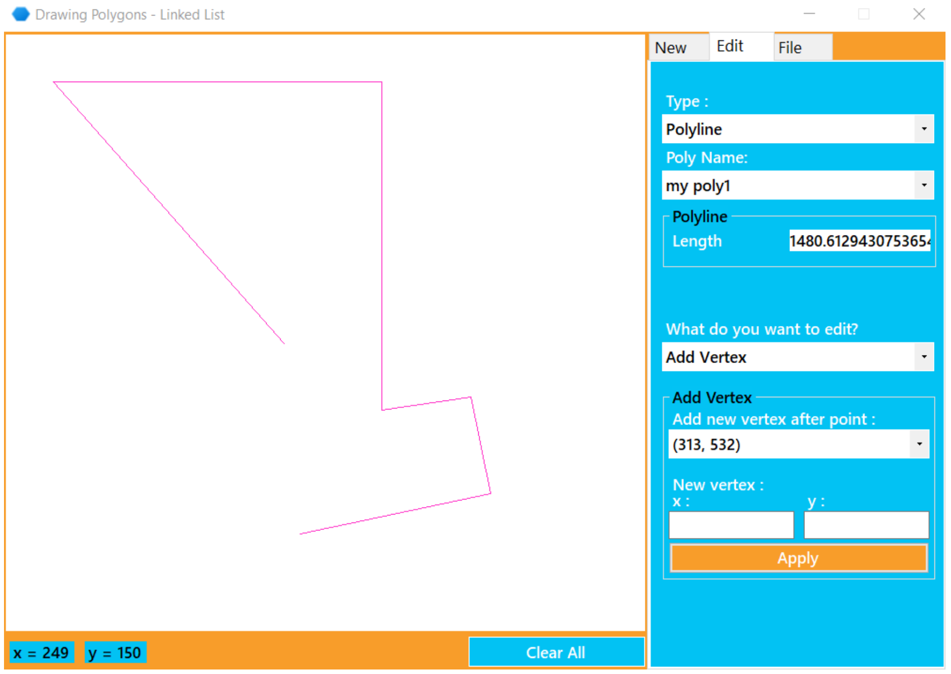
What do you want to edit? Delete Vertex

Delete Vertex

What vertex ? (363, 200)

- Also perform a test case for all the bonuses you implemented.
  Include screenshots of each test case.
  Explain whether each case is successful.

  - ➢ Calculate the length of polyline


Drawing Polygons - Linked List
New
Edit
File
Type :
Poly Name:
What do you want to edit?
Add new vertex after point :
New vertex :
x :
y :
Apply
Clear All

Type : Polyline

Poly Name: my poly1

Polyline

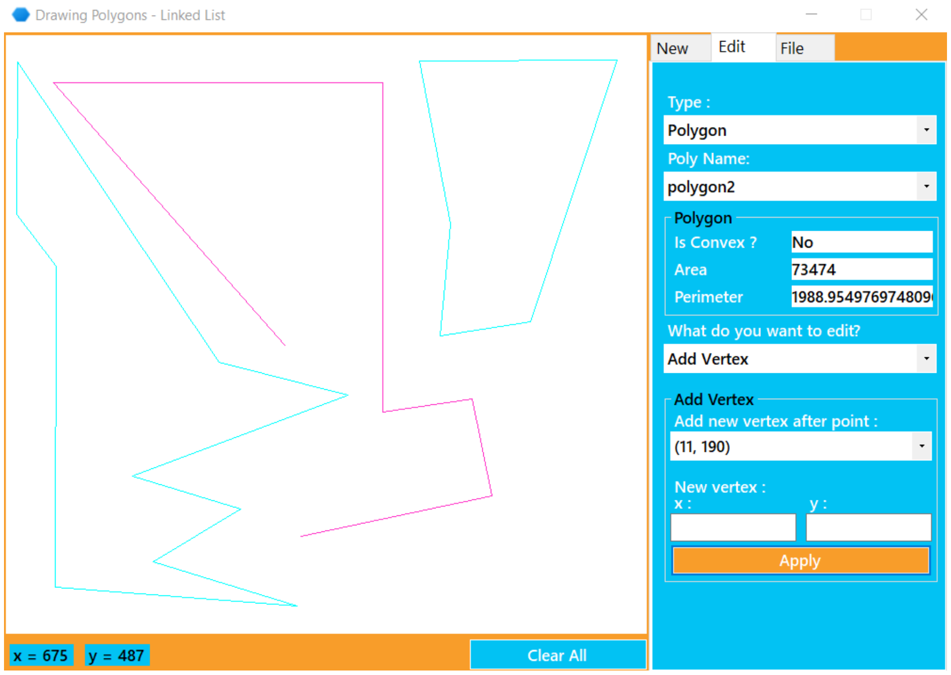
Length 1480.612943075365

Add Vertex

Add new vertex after point : (313, 532)

x = 249 y = 150

  - ➢ Draw multiple polygons/polylines
  - ➢ Calculate area of polygon
  - ➢ Determine the polygon is convex or not
  - ➢ Change color


Drawing Polygons - Linked List
New
Edit
File
Type :
Poly Name:
What do you want to edit?
Add new vertex after point :
New vertex :
x :
y :
Apply
Clear All

Type : Polygon

Poly Name: polygon2

Polygon

Is Convex ? No

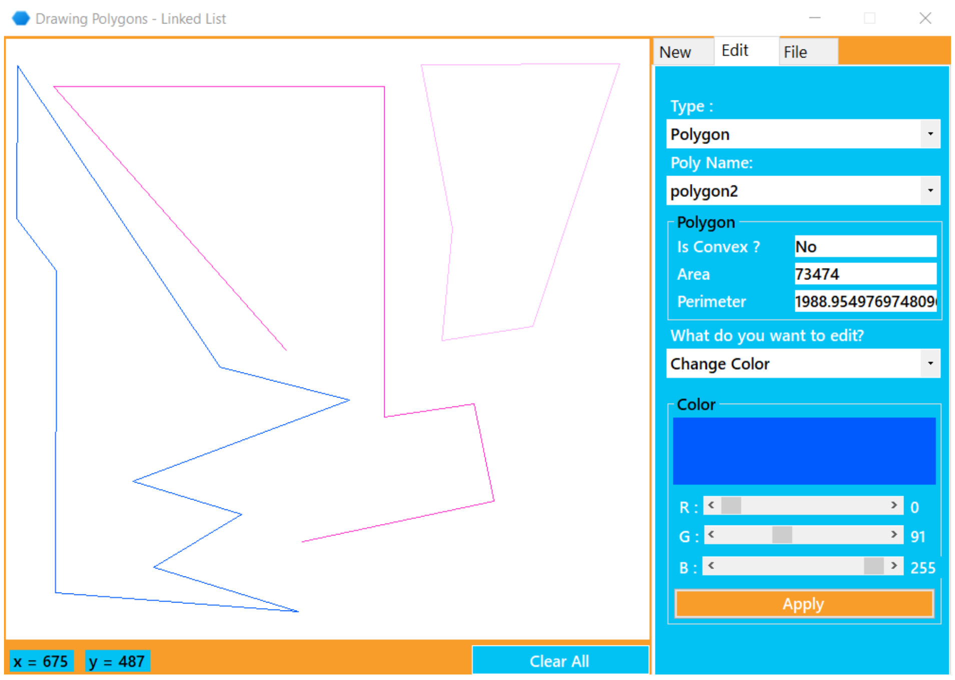
Area 73474

Perimeter 1988.954976974809

Add Vertex

Add new vertex after point : (11, 190)

x = 675 y = 487



Type : Polygon

Poly Name: polygon2

Polygon

Is Convex ? No

Area 73474

Perimeter 1988.954976974809

What do you want to edit? Change Color

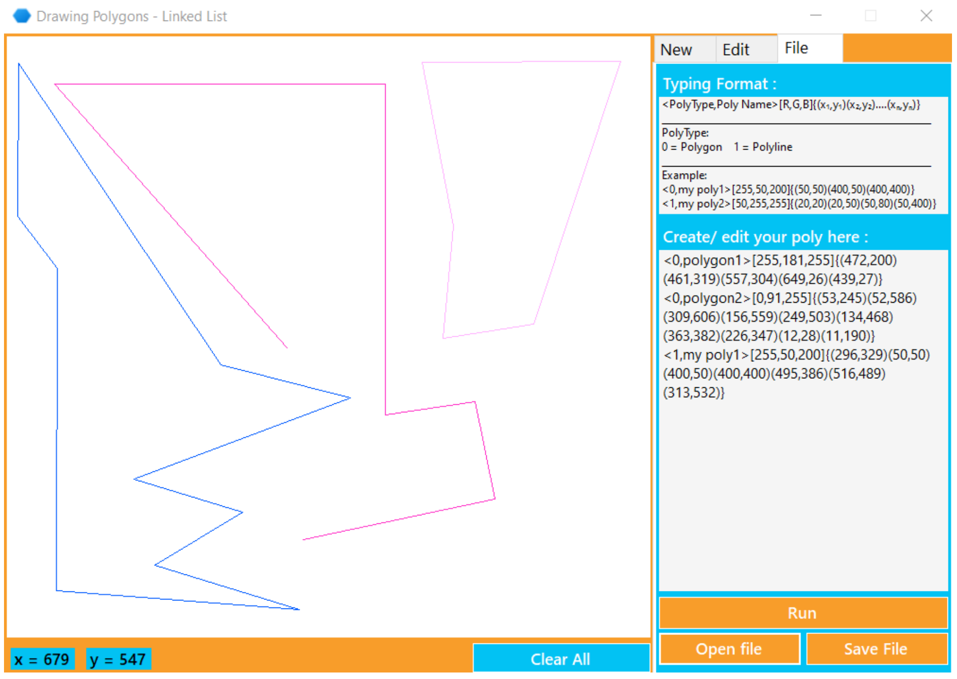
Color

R : 0

G : 91

B : 255

- ➢ Command Line Interface (CLI) to create/edit the polygon/polyline
- ➢ Save/load data of polygons and polylines to/from a file.



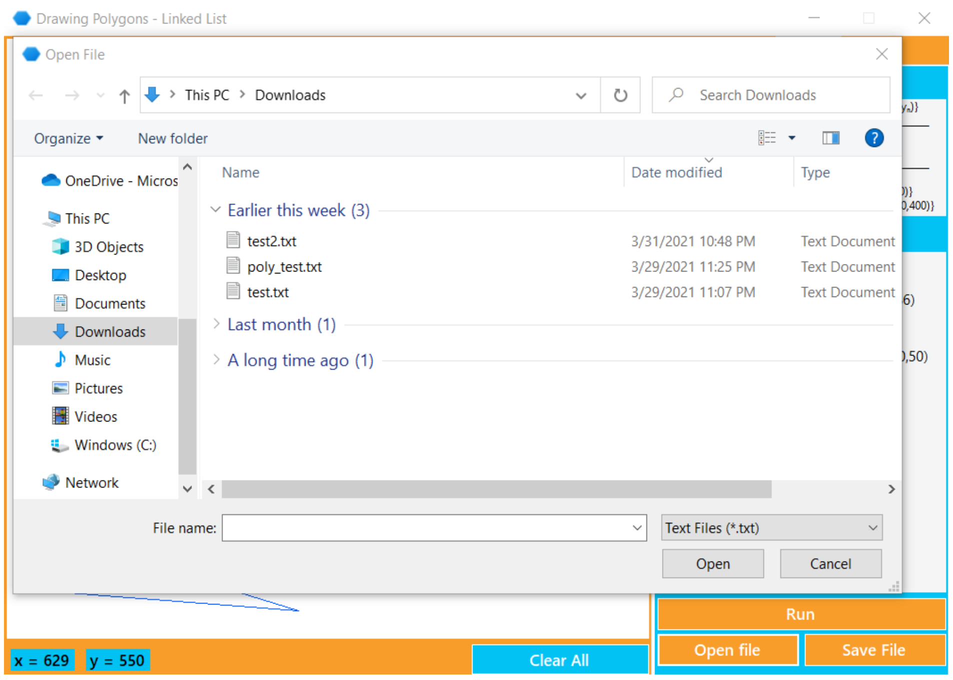
Typing Format :

<PolyType,Poly Name>[R,G,B]{$(x_1,y_1)(x_2,y_2)....(x_n,y_n)$}

PolyType:
0 = Polygon 1 = Polyline

Example:
```
<0,my poly1>[255,50,200]{(50,50)(400,50)(400,400)}
<1,my poly2>[50,255,255]{(20,20)(20,50)(50,80)(50,400)}
```

Create/ edit your poly here :
```
<0,polygon1>[255,181,255]{(472,200)
(461,319)(557,304)(649,26)(439,27)}
<0,polygon2>[0,91,255]{(53,245)(52,586)
(309,606)(156,559)(249,503)(134,468)
(363,382)(226,347)(12,28)(11,190)}
<1,my poly1>[255,50,200]{(296,329)(50,50)
(400,50)(400,400)(495,386)(516,489)
(313,532)}
```

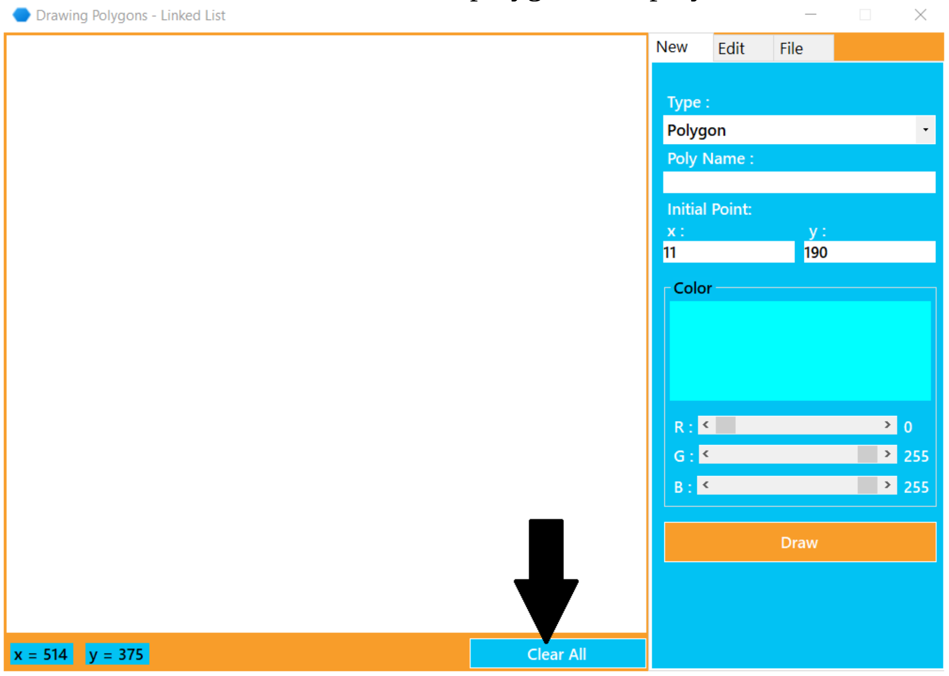


Name	Date modified	Type
Earlier this week (3)		
test2.txt	3/31/2021 10:48 PM	Text Document
poly_test.txt	3/29/2021 11:25 PM	Text Document
test.txt	3/29/2021 11:07 PM	Text Document
Last month (1)		
A long time ago (1)		

File name:
Text Files (*.txt)
Open Cancel

x = 629 y = 550

- Clear the screen. This will delete all polygons and polylines.



Type : Polygon
Poly Name :
Initial Point:
x : 11
y : 190
Color
R : 0
G : 255
B : 255

x = 514 y = 375

6. Work log.

- Record the date and time of every moment you work on this assignment and job description of each member at each session.

Date	Activity/ Progress	Personnel Involved
March 19th, 2021	Starting the project and Looking for some app design innovations	Panji, Arya, Ichsan
March 20th, 2021	Writing program code	Panji
March 25th, 2021	Writing program code	Panji
March 26th, 2021	Writing program code	Panji
March 27th, 2021	Writing program code	Panji
March 28th, 2021	Writing program code	Panji
March 29th, 2021	Writing program code	Panji
March 30th, 2021	Writing program code	Panji
March 31st, 2021	Writing program code	Panji
April 1st, 2021	Testing and Fixing the bug	Panji, Arya, Ichsan
April 2nd, 2021	Finishing the program and making a report	Panji, Arya, Ichsan
April 3rd, 2021	Finishing the report	Panji, Arya, Ichsan

- Write a summary of the implementation of each requirement given in the first page. For each requirement, explain whether that requirement is fully implemented, partially implemented, or not implemented at all. Give explanations if necessary.

  - ➢ Draw a polygon on the screen.
    - ✓ Fully implemented

  - ➢ Draw a polyline on the screen.
    - ✓ Fully implemented

  - ➢ Delete a polygon/polyline.
    - ✓ Fully implemented

  - ➢ Add a vertex to a polygon/polyline. The user can select which existing vertices will be adjacent to the added vertex.
    - ✓ Fully implemented

  - ➢ Edit the location of a vertex on a polygon/polyline.
    - ✓ Fully implemented

  - ➢ Delete a vertex from a polygon/polyline. The user can select which vertex to delete.
    - ✓ Fully implemented

- ➢ Change the colour of a polygon.
  - ✓ Fully implemented

- ➢ Clear the screen. This deletes all polygons and polylines.
  - ✓ Fully implemented

- ➢ Save/load data of polygons and polylines to/from a file.
  - ✓ Fully implemented

- ➢ Determine whether a polygon is convex or not.
  - ✓ Fully implemented

- ➢ Calculate the perimeter and area of a polygon.
  - o Perimeter of polygon
    - ✓ Fully implemented
    - ✓ Get coordinate points → Calculate the vector → Calculate the magnitude of vector to get the length of each line → Sum up all remaining line.

  - o Area of Polygon
    - ✓ Fully implemented
    - ✓ Get coordinate points → Calculate the area of polygon using Shoelace formula

- ➢ Bonus points for user friendliness. Negative points for extreme user unfriendliness.
  - ✓ Fully implemented
  - ✓ We decided to choose bright colors as the interface color for this program, such as blue and orange
  - ✓ We designed each toolbar such as TextBox, Button, ComboBox, etc. into a flat design to make it look more modern and user-friendly

7. Conclusion and remarks.

- Does the program work as expected?
  - ➢ Yes, we believe so.

- If some parts of the program do not work as expected, explain why.
  - ➢ All programs have worked as expected.

- What are your comments about this assignment?
  - ➢ Very interesting and this is the right project to implement our knowledge of linked list data structures.

Reference :

- COMPUTER GRAPHICS AND ANIMATION - 05 Clipping (2).pptx
- https://youtu.be/FSWPX0XB7a0
- https://docs.microsoft.com/en-us/dotnet/csharp/