**HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY OF INTERNATIONAL EDUCATION**

**MAJOR: INFORMATION TECHNOLOGY**

🙤🙤🙤🙤🙤🕮🙦🙦🙦🙦🙦



**MIDTERM PROJECTS**

**WINDOWS PROGRAMMING**

**Game**

aaaaaaaaaaa

**Tutor:** Prof. Le Van Vinh

**Class:** WIPR230579E\_22\_2\_01FIE

**Name:** Bùi Ngọc Hiệp

**Student\_ID:** 21110769

**Semester**: II

**School Year:** 2022-2023

**Apendix**

**[Exercise 1: Paint application Develop a Paint application allowing users are able to create graphic objects, with following functions: 5](#_Toc3008)**

**[a) Basic functions:](#_Toc13767)** [5](#_Toc13767)

**[+ Draw graphic objects: Line, Ellipse, Filled ellipse, rectangle, filled rectangle, circle, filled circle, Arc, polygon, filled polygon](#_Toc32060)**[. 6](#_Toc32060)

[1. Line: 7](#_Toc5609)

[2. Ellipse: 8](#_Toc32689)

[3. Rectangle: 9](#_Toc25633)

[4. Circle: 10](#_Toc23018)

[5. Arc: 12](#_Toc13809)

[6. Polygon 13](#_Toc13734)

[7. Hexagon (Bonus): 14](#_Toc22364)

[8. Triangle (Bonus): 15](#_Toc15148)

[9. Diamond (Bonus): 17](#_Toc29328)

[10. Star (Bonus): 18](#_Toc10966)

[11. Pencil (Bonus): 19](#_Toc4579)

[12. Eraser (Bonus): 19](#_Toc27541)

**[+ Develop other functions allowing users are able to adjust Pen properties (color, width, dash style), and Brush properties (color, style). 27](#_Toc13987)**

**[+ Select and drag the graphic objects. 29](#_Toc11981)**

**[+ Zoom in and zoom out to change the size of the objects. 33](#_Toc22442)**

**[b) Advanced functions 36](#_Toc5341)**

**[+ Remove one or many selected objects (Using Ctr + mouse click to select many objects) 36](#_Toc14463)**

**[+ Group and ungroup the objects. 39](#_Toc5278)**

Exercise 1: Paint application Develop a Paint application allowing users are able to create graphic objects, with following functions:

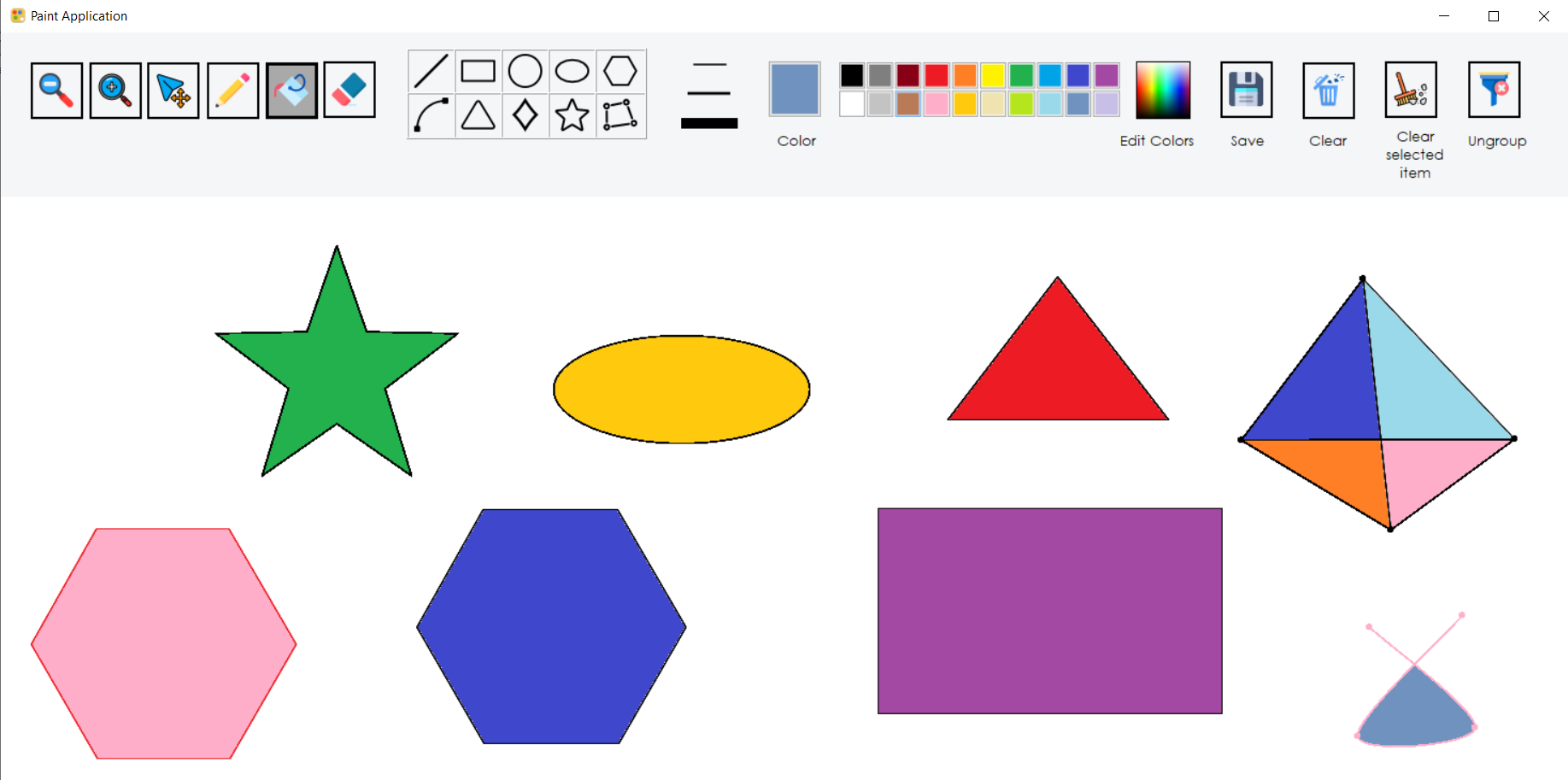
1. **Introduction:**

pictureBox\_Colors

(Picture Box)

Color Dialog

Button to click



*(From my computer*)

picMain (Picture Box)

- DrawShape class (parent class):

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Linq;  using System.Net;  using System.Runtime.InteropServices;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace Paint\_Application  {  public abstract class DrawShape  {  public Point p1;  public Point p2;  public Point originalLocation;  // To serve the Select and Group function  public Color Color\_ { get; set; }  public float Width { get; set; }  public bool isSelected { get; set; }  // To determine that whether this shape is selected?  public bool isDash { get; set; }  // A function to create a bound of a shape, we use it to select shape  public Rectangle Bound {  get  {  int x = Math.Min(p1.X, p2.X);  int y = Math.Min(p1.Y, p2.Y);  int width = Math.Abs(p1.X - p2.X);  int height = Math.Abs(p1.Y - p2.Y);  return new Rectangle(x, y, width, height);  }    }  // A function to move the shape with the lastet point  public virtual void Distance(Point point)  {  p1 = new Point((p1.X + point.X), (p1.Y + point.Y));  p2 = new Point((p2.X + point.X), (p2.Y + point.Y));  }  // An abstract function to draw all of shape, we use it for Paint Event  public abstract void Draw(Graphics gp);  }  } |

+ A shape need 2 points p1 and p2 to draw an object. This is the most important field to draw all of the shape, without these field, we can not do anything.

+ originalLocation: To serve the Select and Group function, with this point we will save the previous position of the shape before we move to anywhere on the picMain. It very important, if you want to select and group, we have to need this.

+ Color\_ and Width are parameter to adjust the style, color and width of a shape.

+ A bool variables isSelected use to know is that shape is selected to move, or is resized. It is a flag that help we to discriminate the selected and unselected shape. It will helpful in the group and ungroup method.

+ A bool variables isDash use to know is that shape is need to Dash or not.

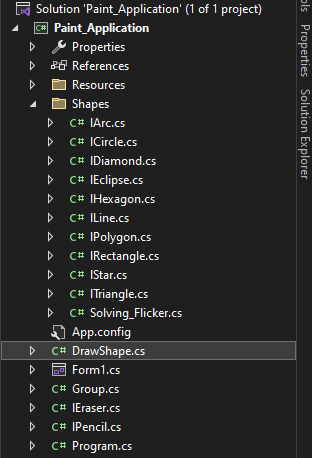
+ The Bound function is use to know the bound over of the shape, this function is create a invisible rectangle over the shape, when the client click on the shape, they will check whether this rectangle contains your mouse click location. This use for the select, group, drag, ungroup method.

+ The Distance function is use to move the selected shape. It will update the point p1 and p2 to redraw the shape with the latest position.

+ Abstract function Draw for derived classes must be re-declared. Every shape must be override this fucntion to draw.

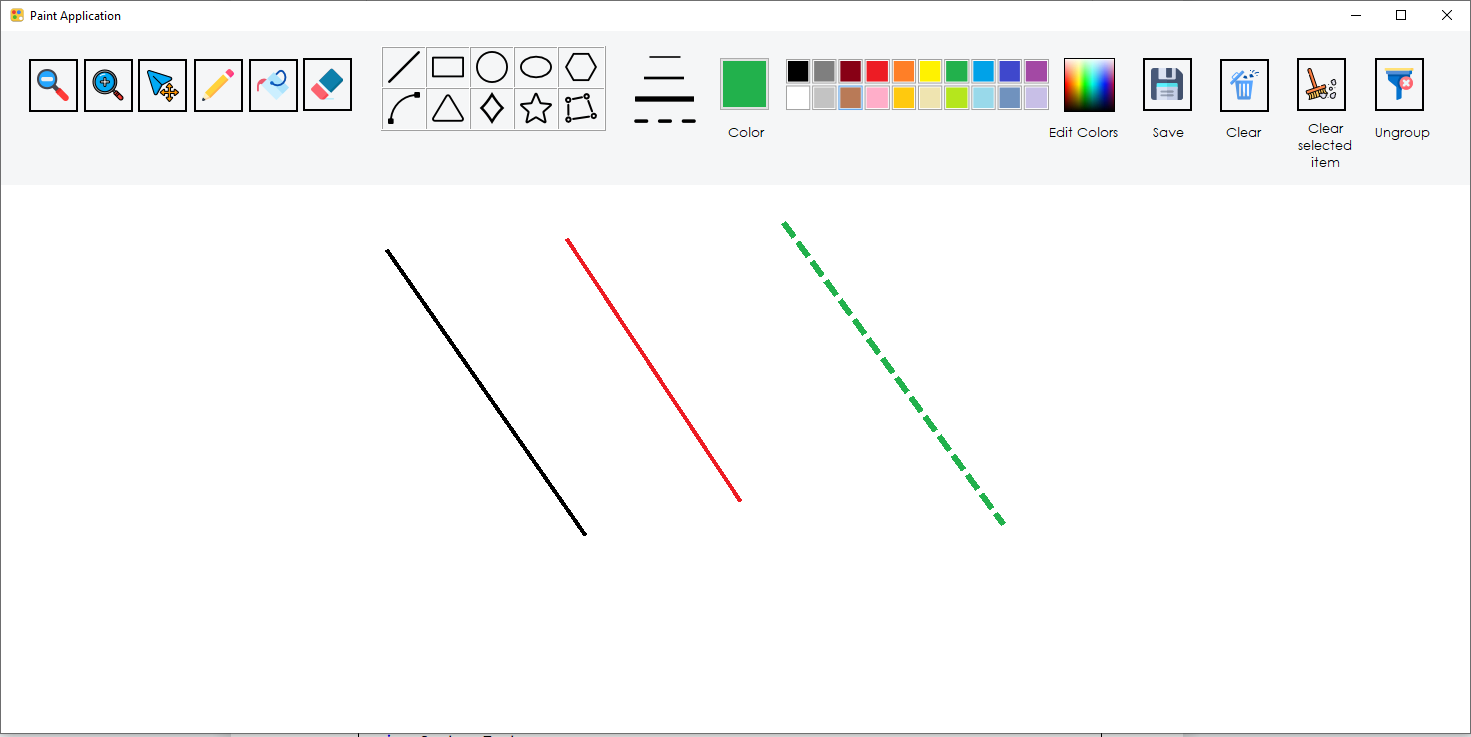
1. **Basic functions:**

**+ Draw graphic objects: Line, Ellipse, Filled ellipse, rectangle, filled rectangle, circle, filled circle, Arc, polygon, filled polygon.**



Open my Shapes folder, you will see my graphic objects:

1. **Line:**

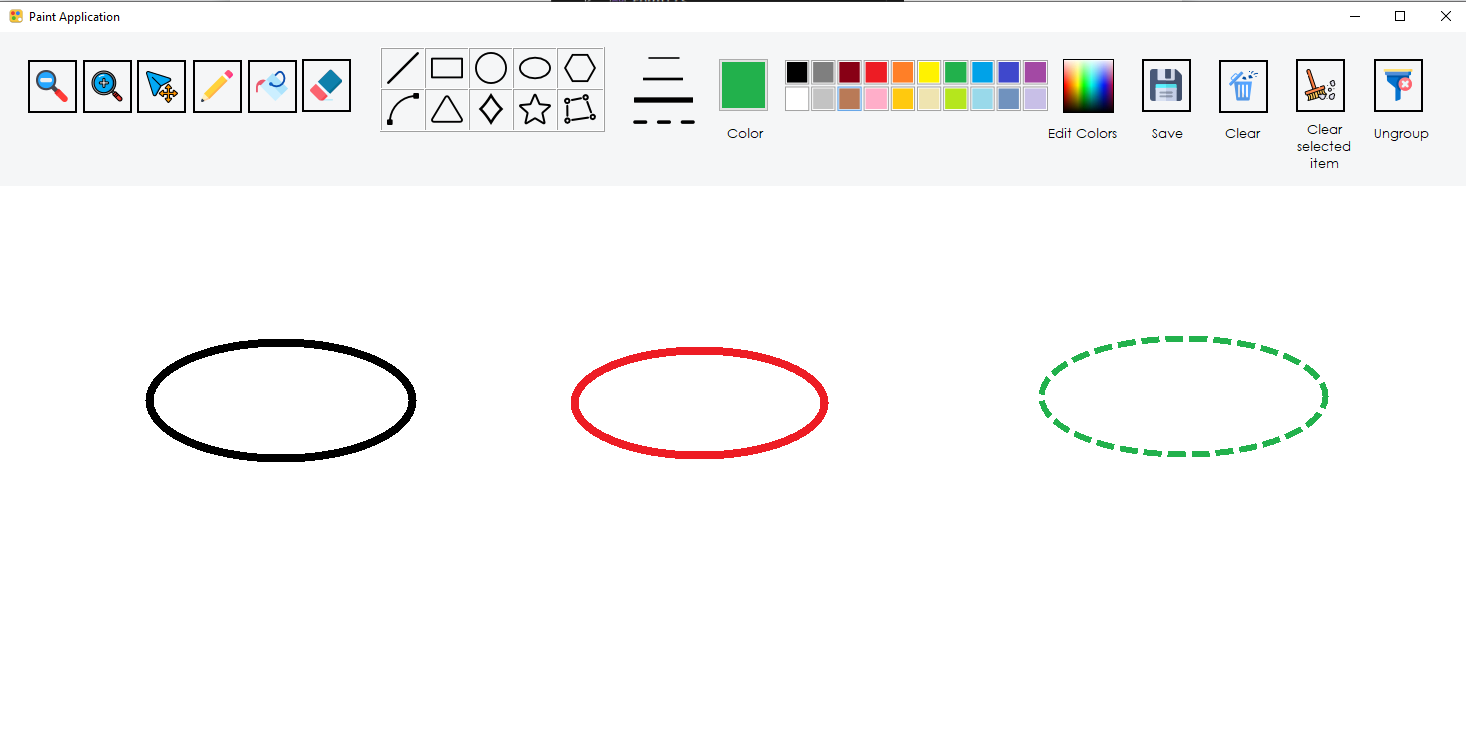


*(From my computer*)

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Drawing.Drawing2D;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace Paint\_Application  {  public class lLine:DrawShape  {  public override void Draw(Graphics gp)  {  Pen myPen = new Pen(Color\_, Width);  if (isSelected)  {  myPen.Color = Color.Blue;  myPen.Width = 6;  myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;  gp.DrawLine(myPen, this.p1, this.p2);  }  else if (isDash)  {  myPen.DashStyle = DashStyle.Dash;  myPen.Width = 6;  gp.DrawLine(myPen, this.p1, this.p2);  }  else  {  gp.DrawLine(myPen, this.p1, this.p2);  }  }  }  } |

- From the point p1 and p2, we will have a line.

1. **Ellipse:**

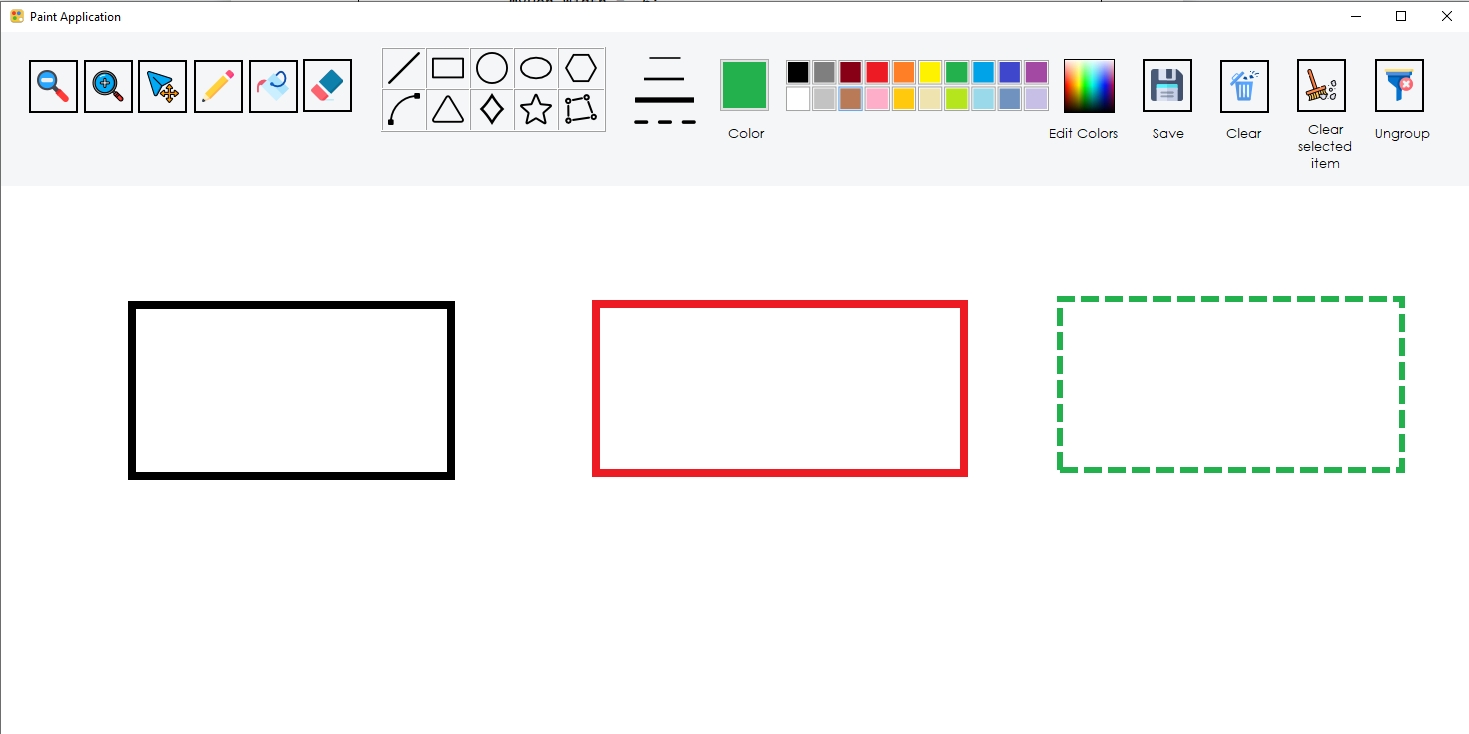


*(From my computer*)

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Drawing.Drawing2D;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace Paint\_Application  {  public class lEclipse : DrawShape  {  public override void Draw(Graphics gp)  {  Pen myPen = new Pen(Color\_, Width);  if (isSelected)  {  myPen.Color = Color.Blue;  myPen.Width = 6;  myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;  gp.DrawEllipse(myPen, this.p1.X, this.p1.Y, this.p2.X - this.p1.X, this.p2.Y - this.p1.Y);  }  else if (isDash)  {  myPen.DashStyle = DashStyle.Dash;  myPen.Width = 6;  gp.DrawEllipse(myPen, this.p1.X, this.p1.Y, this.p2.X - this.p1.X, this.p2.Y - this.p1.Y);  }  else  {  gp.DrawEllipse(myPen, this.p1.X, this.p1.Y, this.p2.X - this.p1.X, this.p2.Y - this.p1.Y);  }  }  }  } |

- From the point p1 and p2, we calculate this.p2.X - this.p1.X, this.p2.Y - this.p1.Y to identify the width and heigth of the ellipsse

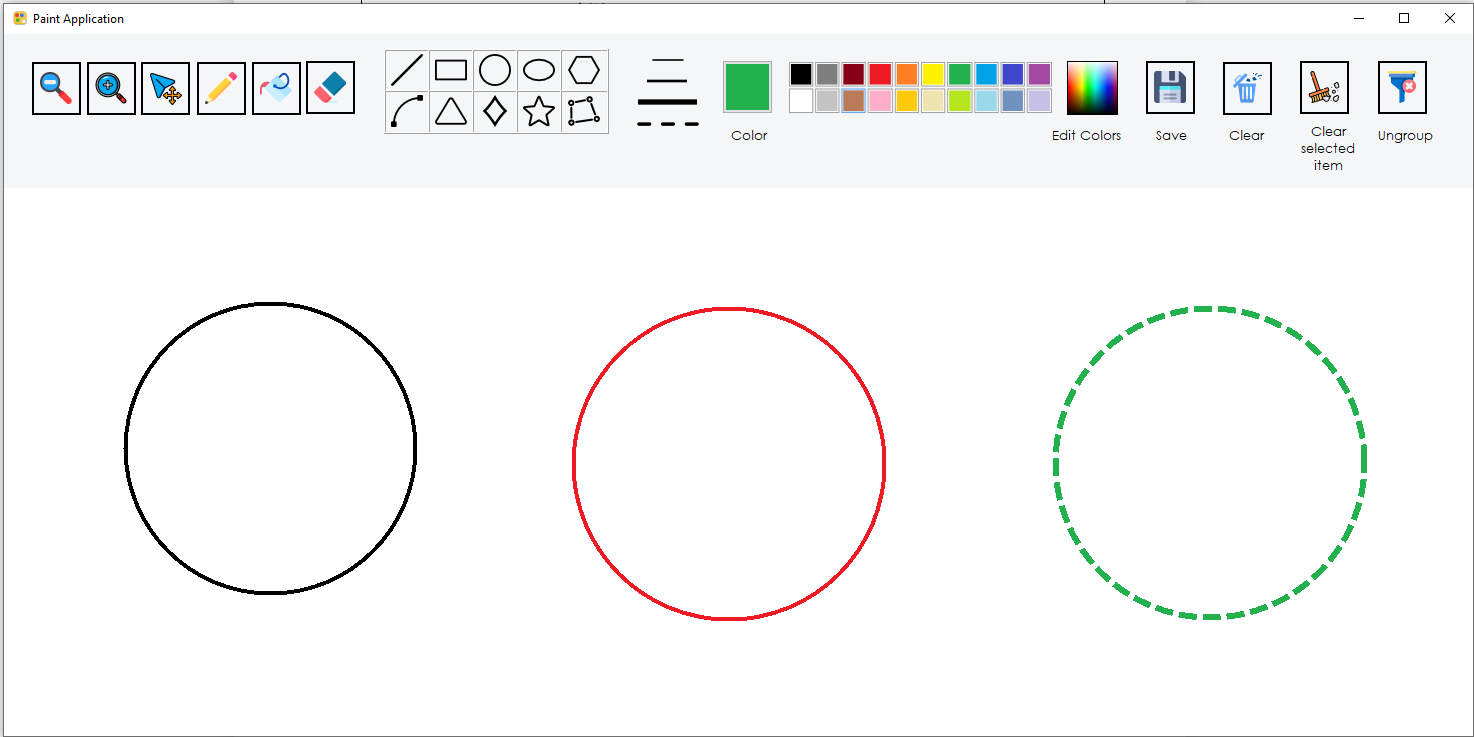
1. **Rectangle:**

*(From my computer*)

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Drawing.Drawing2D;  using System.Linq;  using System.Net;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace Paint\_Application  {  public class lRectangle : DrawShape  {  public override void Draw(Graphics gp)  {  Pen myPen = new Pen(Color\_, Width);  int x = Math.Min(p1.X, p2.X);  int y = Math.Min(p1.Y, p2.Y);  int width = Math.Abs(p1.X - p2.X);  int height = Math.Abs(p1.Y - p2.Y);  if (isSelected)  {  myPen.Color = Color.Blue;  myPen.Width = 6;  myPen.DashStyle = DashStyle.Dash;  gp.DrawRectangle(myPen, x, y, width, height);  }  else if (isDash)  {  myPen.DashStyle = DashStyle.Dash;  myPen.Width = 6;  gp.DrawRectangle(myPen, x, y, width, height);  }  else  {  gp.DrawRectangle(myPen, x, y, width, height);  }  }  }  } |

- x,y is the variance that compare the point p2 and p1, if the p1 is smaller, it will be choosen to the first point. We need to do that because the rectangle can not be draw if the user draw the rectangle from the bottom to the top of picMain.

1. **Circle:**

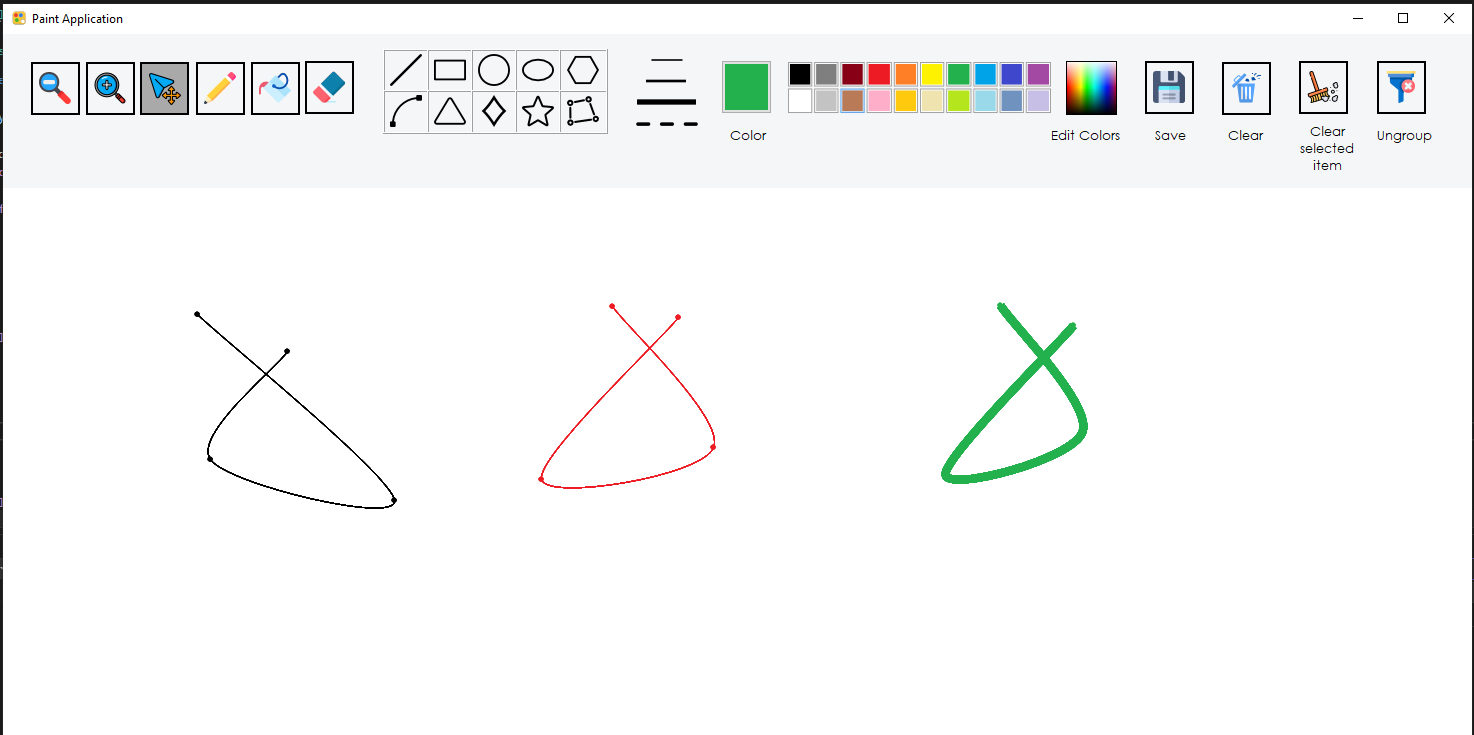


*(From my computer*)

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace Paint\_Application  {  public class lCircle : DrawShape  {  public override void Draw(Graphics gp)  {  Pen myPen = new Pen(Color\_, Width);  if (isSelected)  {  myPen.Color = Color.Blue;  myPen.Width = 6;  myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;  gp.DrawEllipse(myPen, this.p1.X, this.p1.Y, this.p2.X - this.p1.X, this.p2.X - this.p1.X);  }  else if (isDash)  {  myPen.Width = 6;  myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;  gp.DrawEllipse(myPen, this.p1.X, this.p1.Y, this.p2.X - this.p1.X, this.p2.X - this.p1.X);  }  else  {  gp.DrawEllipse(myPen, this.p1.X, this.p1.Y, this.p2.X - this.p1.X, this.p2.X - this.p1.X);  }  }  }  } |

- We know Visual Studio doesn’t have a DrawCircle method but Width and Height of a circle is equal so I use this.p2.X - this.p1.X, this.p2.X - this.p1.X from a DrawEllipse to draw a circle. Very easy

1. **Arc:**

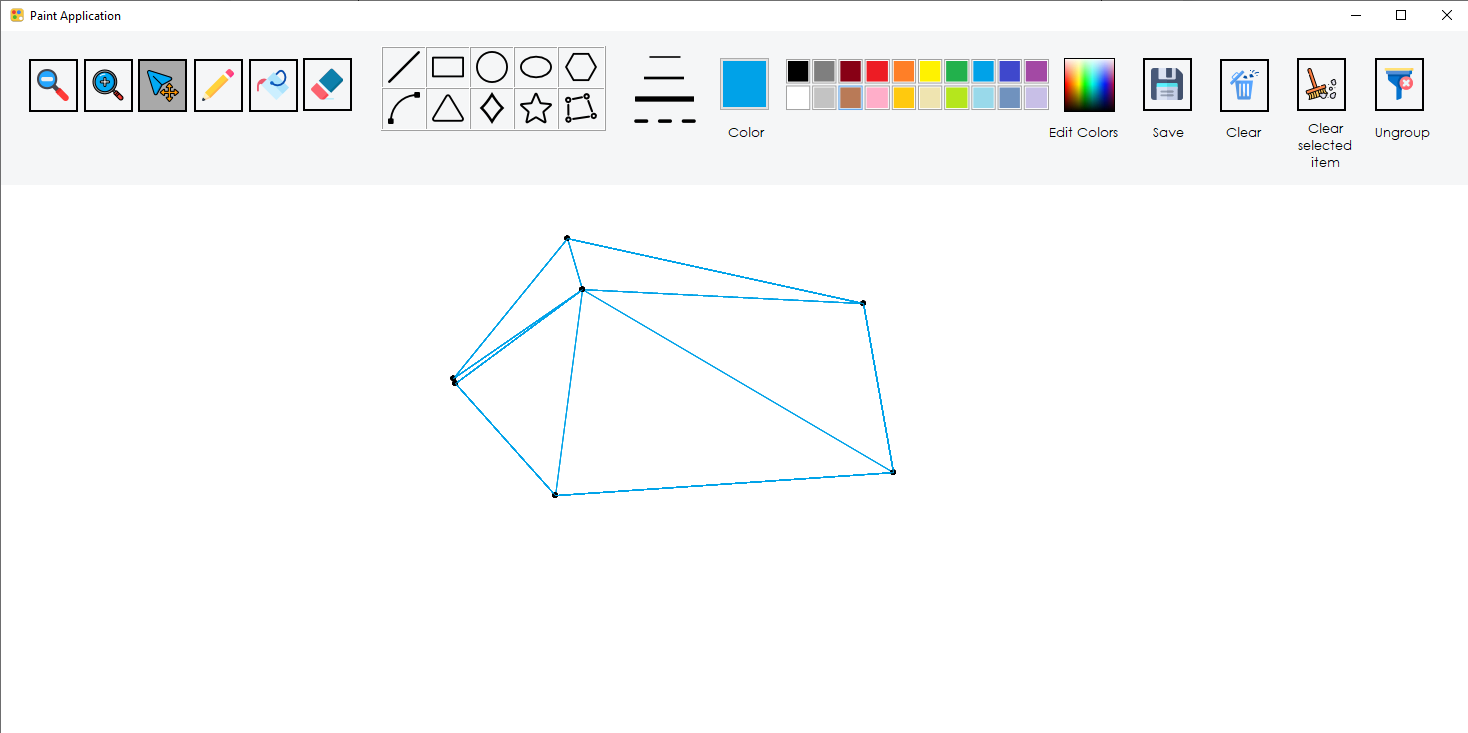


*(From my computer*)

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Drawing.Drawing2D;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace Paint\_Application  {  public class lArc : DrawShape  {  public List<Point> points= new List<Point>();  public override void Draw(Graphics gp)  {  Pen myPen = new Pen(this.Color\_, Width);  Brush myBrush= new SolidBrush(this.Color\_);  gp.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality;  foreach (Point point in points)  {  if (isSelected)  {  myPen.Color = Color.Blue;  myPen.Width = 6;  myPen.DashStyle = DashStyle.Dash;  gp.DrawCurve(myPen, points.ToArray());  }  else if (isDash)  {  myPen.DashStyle = DashStyle.Dash;  myPen.Width = 6;  gp.FillEllipse(myBrush, point.X - 3, point.Y - 3, 5, 5);  if (points.Count < 2)  return;  gp.DrawCurve(myPen, points.ToArray());  }  else  {  gp.FillEllipse(myBrush, point.X - 3, point.Y - 3, 5, 5);  if (points.Count < 2)  return;  gp.DrawCurve(myPen, points.ToArray());  }  }  }  }  } |

- To draw the arc, we must be have a List<Point> that store the point that the user is click on the picMain, then we will connect these point to make the curve.

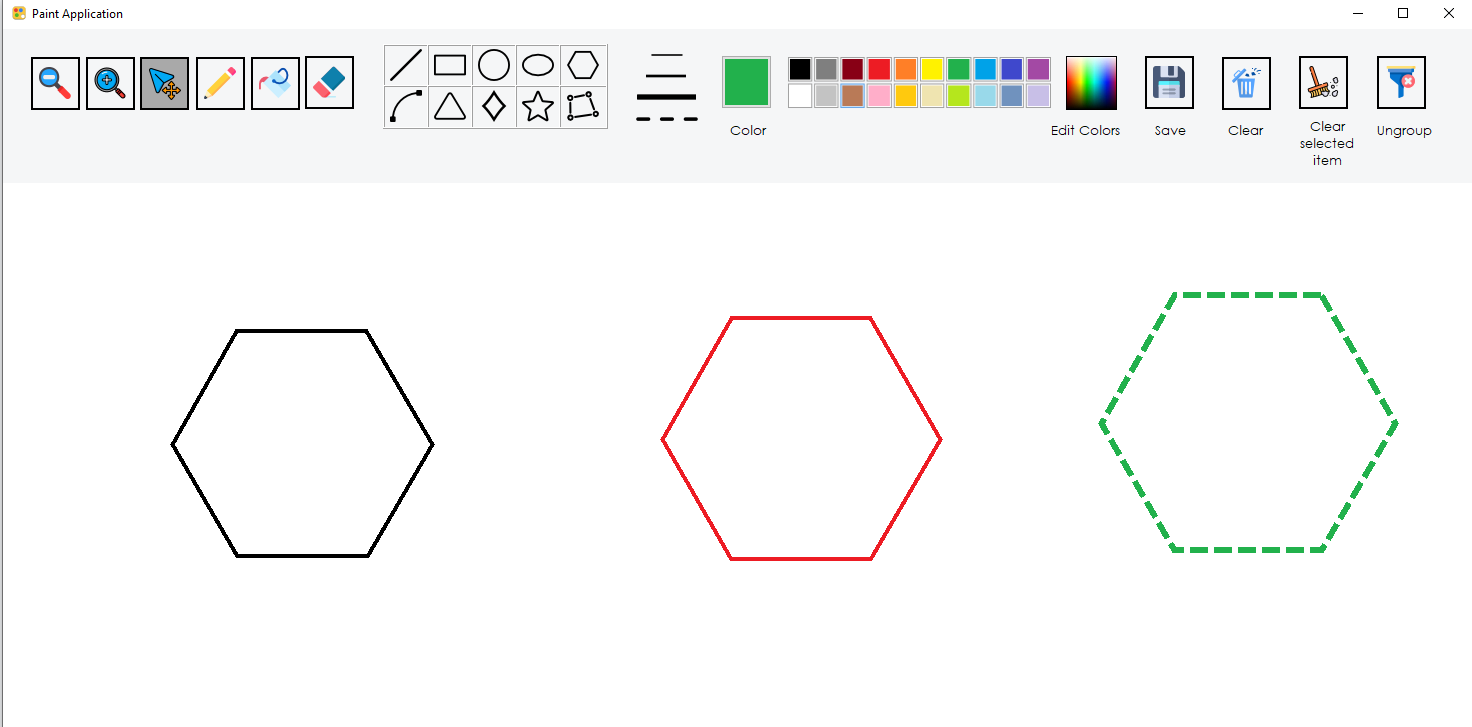
1. **Polygon**



|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Drawing.Drawing2D;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Paint\_Application  {  public class lPolygon : DrawShape  {  public List<Point> points = new List<Point>();  public override void Draw(Graphics gp)  {  Brush brush = new SolidBrush(Color.Black);  Pen pen = new Pen(this.Color\_, this.Width);  gp.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality;  foreach (Point point in points)  {  gp.FillEllipse(brush, point.X - 3, point.Y - 3, 5, 5);  }  Point[] polyPoints = new Point[points.Count + 1];  points.CopyTo(polyPoints);  polyPoints[points.Count] = points[0];  if (isSelected)  {  pen.Color = Color.Blue;  pen.Width = 6;  pen.DashStyle = DashStyle.Dash;  gp.DrawPolygon(pen, polyPoints);  }  else if (isDash)  {  pen.DashStyle = DashStyle.Dash;  pen.Width = 6;  gp.DrawPolygon(pen, polyPoints);  }  else  {  gp.DrawPolygon(pen, polyPoints);  }  }  }  } |

- Polygon is also use a List<Point>, when the user is click on the picMain then we will cahnge the List<Point> to an array Point[] polyPoints because the method to draw polygon is only allow an array to draw the polygon.

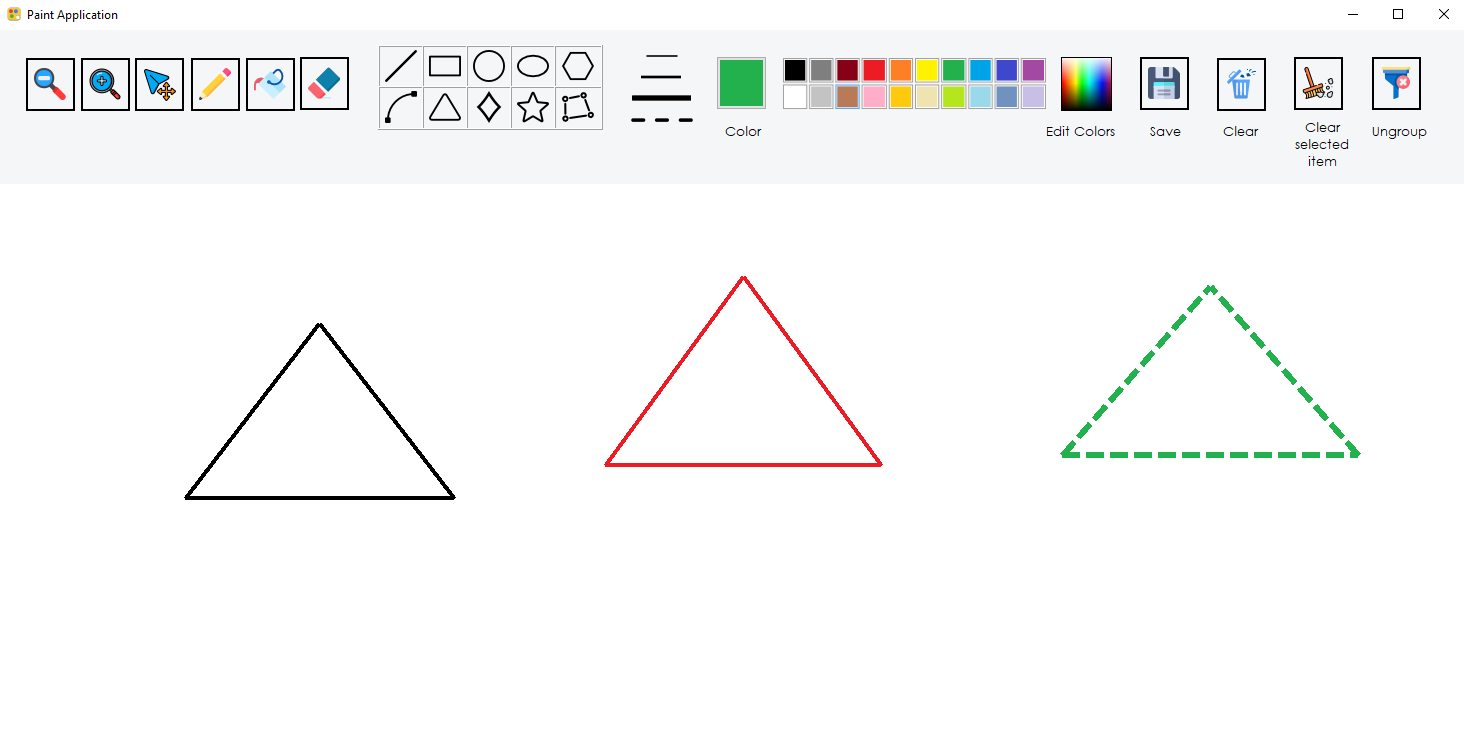
1. **Hexagon (Bonus)**:



|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Drawing.Drawing2D;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace Paint\_Application  {  public class lHexagon : DrawShape  {  public Point[] hexagonPoints = new Point[6];  public override void Draw(Graphics gp)  {  Pen myPen = new Pen(Color\_, Width);  double angle = 2 \* Math.PI / 6;  for (int i = 0; i < 6; i++)  {  int x = (int)(p1.X + (p2.X-p1.X) \* Math.Cos(i \* angle));  int y = (int)(p1.Y + (p2.X - p1.X) \* Math.Sin(i \* angle));  hexagonPoints[i] = new Point(x, y);  //To add 5 point of the hexagon  }  if (isSelected)  {  myPen.DashStyle = DashStyle.Dash;  myPen.Color = Color.Blue;  myPen.Width = 6;  gp.DrawPolygon(myPen, hexagonPoints);  }  else if (isDash)  {  myPen.DashStyle = DashStyle.Dash;  myPen.Width = 6;  gp.DrawPolygon(myPen, hexagonPoints);  }  else  {  gp.DrawPolygon(myPen, hexagonPoints);  }  }  }  } |

- First, we need to know the angle of the polygon, then with the angle, we will combine it to find 6 points and add into the hexagonPoints array. Then write the polygon with these points in hexagonPoints.

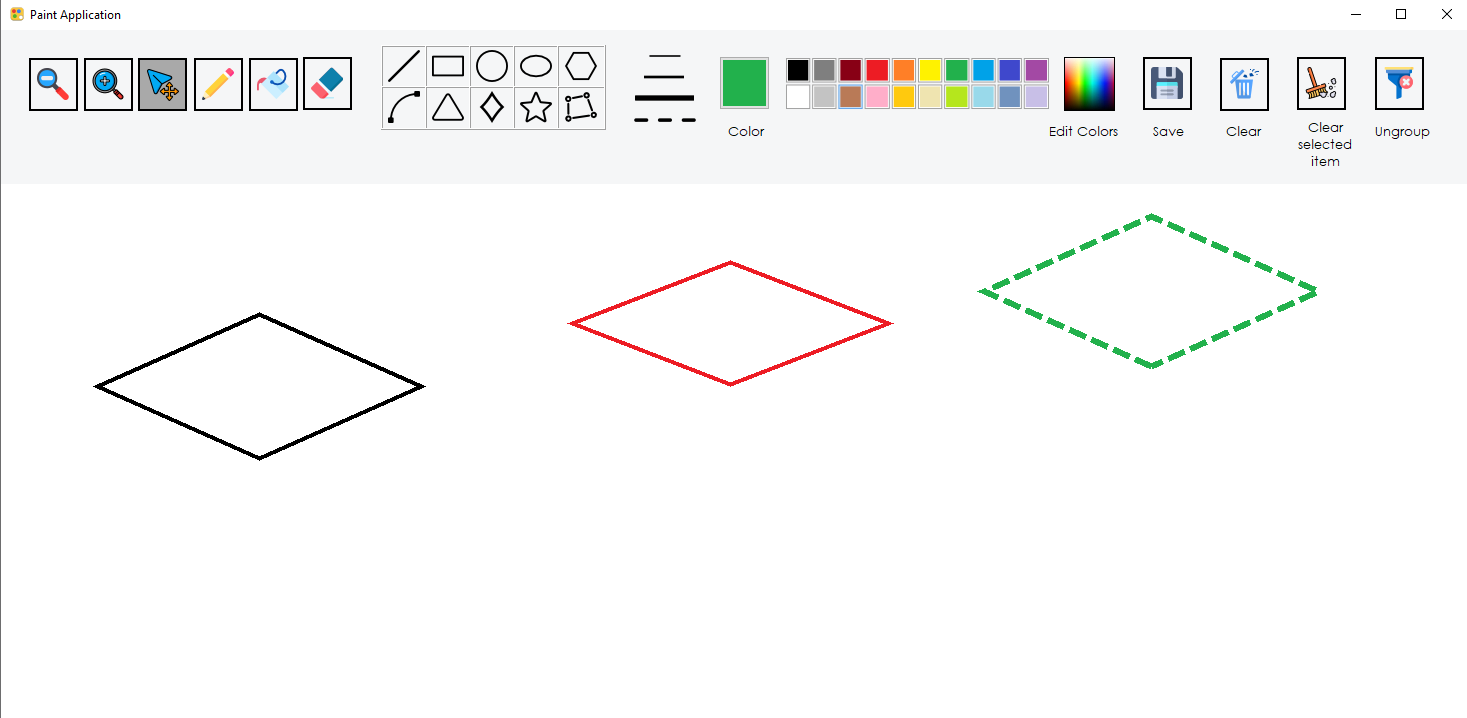
1. **Triangle (Bonus):**



|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Drawing.Drawing2D;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace Paint\_Application  {    public class lTriangle:DrawShape  {  public override void Draw(Graphics gp)  {  Pen myPen = new Pen(Color\_, Width);  double xMid= (p1.X+ p2.X)/2;  //Find the center of a triangle  Point first= new Point(p1.X, p2.Y);  Point mid = new Point((int)xMid, p1.Y);    if (isSelected)  {  Pen newPen = new Pen(Color.Blue, 6);  newPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;  gp.DrawLine(newPen, first, mid);  gp.DrawLine(newPen, first, p2);  gp.DrawLine(newPen, p2, mid);  }  else if (isDash)  {  myPen.Width = 6;  myPen.DashStyle = DashStyle.Dash;  gp.DrawLine(myPen, first, mid);  gp.DrawLine(myPen, first, p2);  gp.DrawLine(myPen, p2, mid);  }  else  {  gp.DrawLine(myPen, first, mid);  gp.DrawLine(myPen, first, p2);  gp.DrawLine(myPen, p2, mid);  }  }  }  } |

- Before draw, we need a middle location of triangle, then the first point and mid point can be determine to draw.

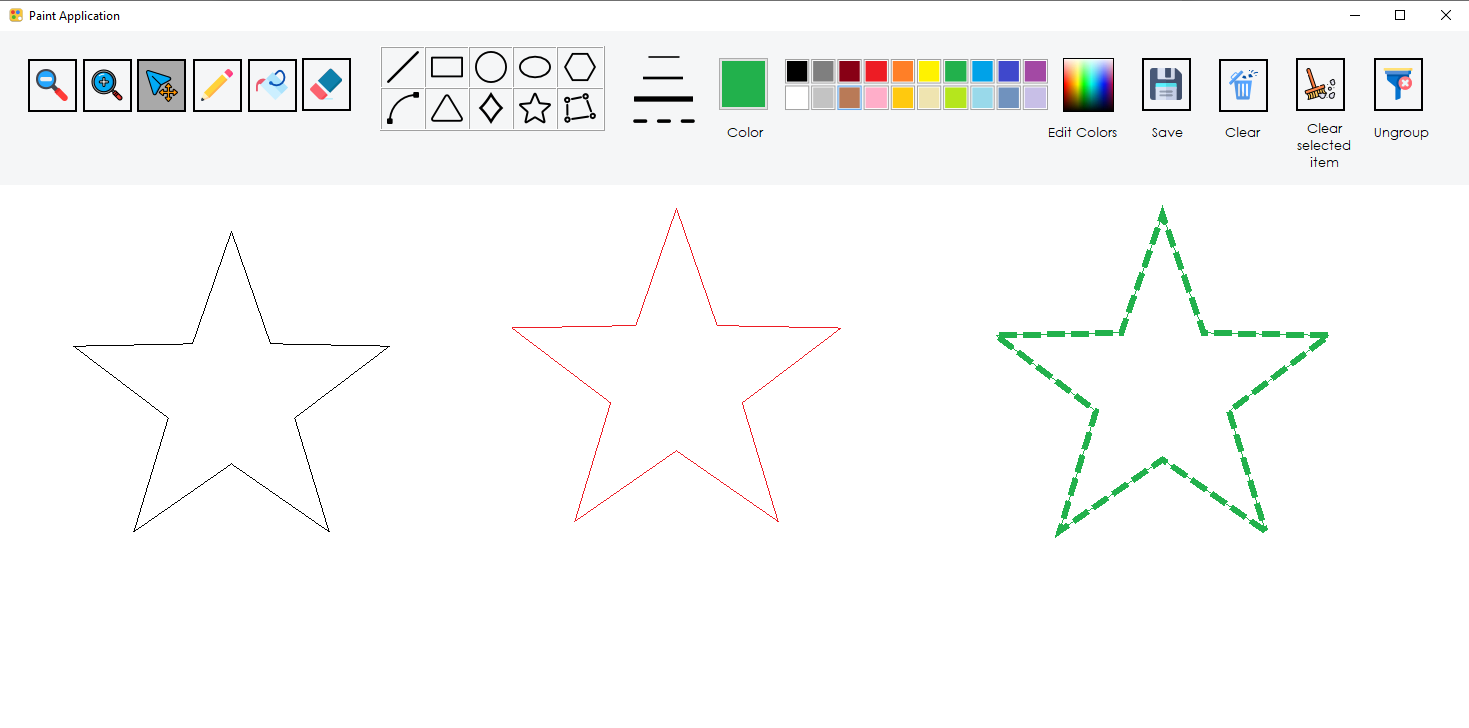
1. **Diamond (Bonus):**



|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Drawing.Drawing2D;  using System.Linq;  using System.Runtime.InteropServices;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Forms;  namespace Paint\_Application  {  public class lDiamond:DrawShape  {  public override void Draw(Graphics gp)  {  System.Drawing.Pen myPen = new System.Drawing.Pen(Color\_, Width);  int height = p2.Y - p1.Y; int width = p2.X - p1.X;  PointF[] diamondPoints = new PointF[4];  diamondPoints[0] = new PointF(p1.X - width / 2, p1.Y);  diamondPoints[1] = new PointF(p1.X, p1.Y - height / 2);  diamondPoints[2] = new PointF(p1.X + width / 2, p1.Y);  diamondPoints[3] = new PointF(p1.X, p1.Y + height / 2);  // An array of point to draw the diamond, which has 4-sided  if (isSelected)  {  myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;  myPen.Width = 6;  myPen.Color = Color.Blue;  gp.DrawPolygon(myPen, diamondPoints);  }  else if (isDash)  {  myPen.DashStyle = DashStyle.Dash;  myPen.Width = 6;  gp.DrawPolygon(myPen, diamondPoints);  }  else  {  gp.DrawPolygon(myPen, diamondPoints);  }  }  }  } |

- A diamond has 4 sided so we need to calculate these point by using p1 and p2 point then add to the array diamondPoints[] then draw a polygon with this array

1. **Star (Bonus):**



|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Drawing.Drawing2D;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Forms;  namespace Paint\_Application  {  public class lStar:DrawShape  {  public override void Draw(Graphics gp)  {  System.Drawing.Pen myPen = new System.Drawing.Pen(Color\_, Width);  //First,, we need to calculate to find the angle and the radius to draw a star  const double numPoints = 5;  PointF[] starPoints = new PointF[(int)(numPoints \* 2)];  double angle = Math.PI / 2;  double angleIncrement = Math.PI / numPoints;  double radius = p2.X-p1.X;  //Next, make an array which has 10 element tho draw the star  for (int i = 0; i < numPoints \* 2; i++)  {  double innerRadius = i % 2 == 0 ? radius \* 0.4 : radius;  float x = p1.X + (float)(Math.Cos(angle) \* innerRadius);  float y = p1.Y + (float)(Math.Sin(angle) \* innerRadius);  starPoints[i] = new PointF(x, y);  angle += angleIncrement;  }  gp.DrawPolygon(myPen, starPoints);  if (isSelected)  {  myPen.DashStyle = DashStyle.Dash;  myPen.Width = 6;  myPen.Color = Color.Blue;  gp.DrawPolygon(myPen, starPoints);  }  else if (isDash)  {  myPen.DashStyle = DashStyle.Dash;  myPen.Width = 6;  gp.DrawPolygon(myPen, starPoints);  }  else  {  gp.DrawPolygon(myPen, starPoints);  }  }    }  } |

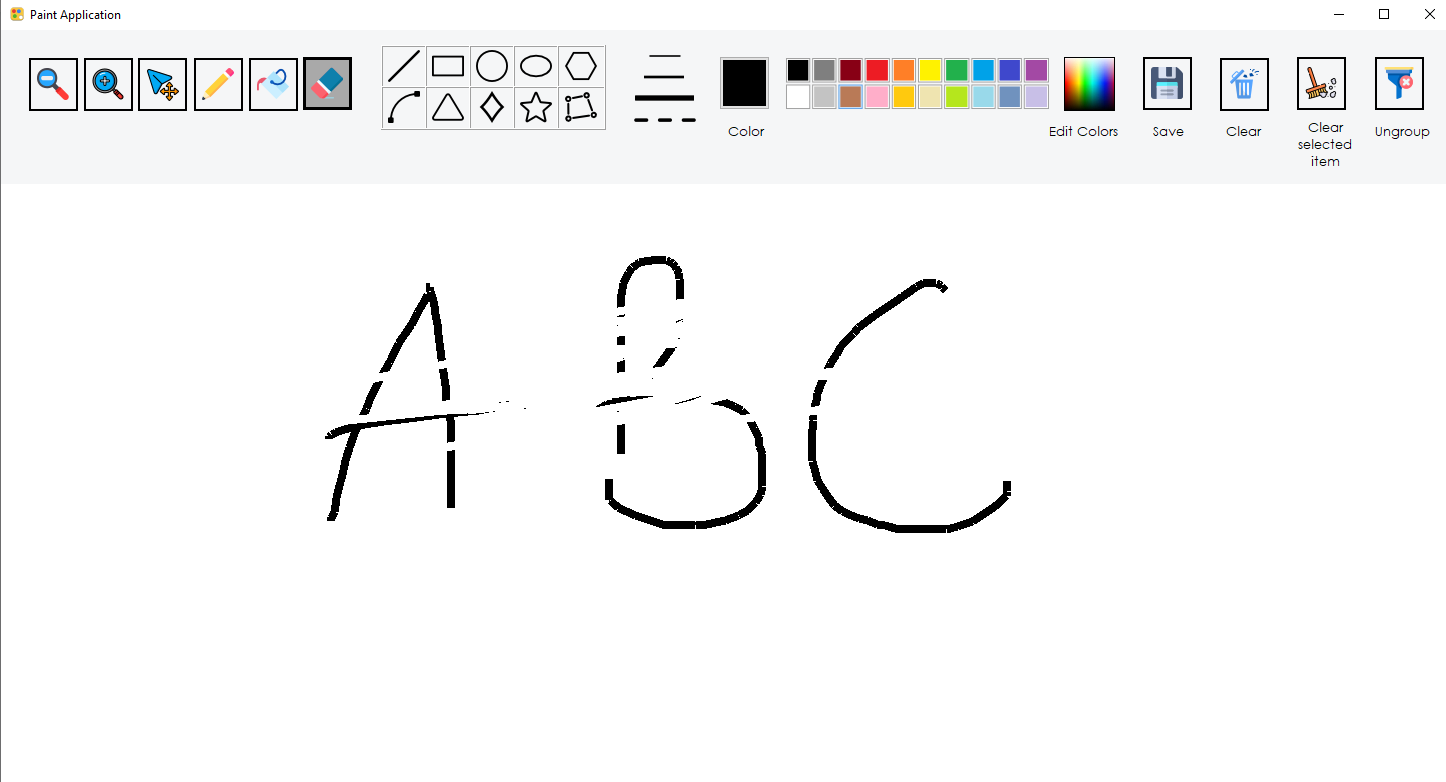
-To draw the Star, we need to find the radius and angle, then multiple the radius and angle to the point p1,p2 we will have the starPoints to draw the Star.

1. **Pencil (Bonus):**



- Pencil: Allow user using mouse to draw everything, also can change the width of the draw line.

1. **Eraser (Bonus):**



- Eraser: Allow user erase thing painted in the board. And another way to clear fast the board is button “Clear” in the top right corner of the application.

==> As we can see from 10 shapes above, all of shapes are override the **abstract Draw function** of the DrawShape class to draw their own shape.

**Example**: lRectangle class

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Drawing;  using System.Drawing.Drawing2D;  using System.Linq;  using System.Net;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  Override the Draw function  namespace Paint\_Application  {  public class lRectangle : DrawShape  {  public override void Draw(Graphics gp)  {  Pen myPen = new Pen(Color\_, Width);  int x = Math.Min(p1.X, p2.X);  int y = Math.Min(p1.Y, p2.Y);  int width = Math.Abs(p1.X - p2.X);  int height = Math.Abs(p1.Y - p2.Y);  if (isSelected)  {  myPen.Color = Color.Blue;  myPen.Width = 6;  myPen.DashStyle = DashStyle.Dash;  gp.DrawRectangle(myPen, x, y, width, height);  }  else if (isDash)  {  myPen.DashStyle = DashStyle.Dash;  myPen.Width = 6;  gp.DrawRectangle(myPen, x, y, width, height);  }  else  {  gp.DrawRectangle(myPen, x, y, width, height);  }  }  }  } |

- To draw the shape, we need to declare a List<DrawShape> to store the shape and then draw it in picMain\_Paint event.

|  |
| --- |
| public partial class Form1 : Form  {  Color myColor;  Pen myPen;  Pen myEraser;  DrawShape shape;  DrawShape selected\_Shape;  List<DrawShape> lselected\_Shape = new List<DrawShape>();  List<Group> lgroup = new List<Group>();  Group \_currentGroup;  Point p1, p2;  Point StartLocation;  Point DownLocation = new Point();  Bitmap bm;  Graphics g;  /// <List\_of\_Shape>  List<DrawShape> lstObject = new List<DrawShape>();  lArc larc = new lArc();  lTriangle ltriangle = new lTriangle();  lHexagon lpolygon = new lHexagon();  lPolygon New\_Polygon = new lPolygon();  lStar lstar = new lStar();  lDiamond ldiamond = new lDiamond();  List<Point> points = new List<Point>();  Point[] hexagonPoints;  //Change the Width and Color  int width = 1;  private float zoom = 1.2f;  /// <Shape\_button>  bool bLine;  bool bRec;  bool bCircle;  bool bEllipse;  bool bPolygon;  bool bPolygon\_New;  bool bArc;  bool bTriangle;  bool bStar;  bool bDiamond;  bool bEraser;  bool bPencil;  bool bFill;  bool bSelect;  bool bDash;  bool paint = false;  } |

- If the client click on any button, then the bool variance will check and implement the corresponding shape.

**Example**: If the client click on the rectangle button to draw, the bRectangle will be set to true.

|  |
| --- |
| private void btn\_Rectangle\_Click(object sender, EventArgs e)  {  Refresh();  this.bRec = true;  } |

* The picMain\_MouseDown() event are going to deploy the method.

|  |
| --- |
| private void picMain\_MouseDown(object sender, MouseEventArgs e)  {  p1 = e.Location;  if (bPencil)  {  this.paint = true;  p1 = e.Location;  }  if (bEraser)  {  p1 = e.Location;  this.paint = true;  }  if (bLine)  {  DrawShape myObj;  myObj = new lLine();  myObj.p1 = e.Location;  myObj.Color\_ = btn\_Color.BackColor;  myObj.Width = width;  this.lstObject.Add(myObj);  this.paint = true;  if (bDash)  {  myObj.isDash = true;  }  }  if (bRec)  {  DrawShape myObj;  myObj = new lRectangle();  myObj.p1 = e.Location;//Set the first point of shape  myObj.Color\_ = btn\_Color.BackColor;//Set the color of shape  myObj.Width = width;//Set the width of shape  this.lstObject.Add(myObj);// Add the object to the list<DrawShape>  if (bDash)  {  myObj.isDash = true;  }  this.paint = true;  }  if (bEllipse)  {  DrawShape myObj;  myObj = new lEclipse();  myObj.p1 = e.Location;  myObj.Color\_ = btn\_Color.BackColor;  myObj.Width = width;  this.lstObject.Add(myObj);  if (bDash) myObj.isDash = true;  this.paint = true;  }  if (bTriangle)  {  ltriangle.p1 = ltriangle.p2 = e.Location;  ltriangle.Color\_ = btn\_Color.BackColor;  ltriangle.Width = width;  this.lstObject.Add(ltriangle);  this.paint = true;  if (bDash) ltriangle.isDash = true;  }  if (bPolygon)  {  lpolygon.p1 = lpolygon.p2 = e.Location;  private void picMain\_MouseDown(object sender, MouseEventArgs e)  {  p1 = e.Location;  if (bPencil)  {  this.paint = true;  p1 = e.Location;  }  if (bEraser)  {  p1 = e.Location;  this.paint = true;  }  if (bLine)  {  DrawShape myObj;  myObj = new lLine();  myObj.p1 = e.Location;  myObj.Color\_ = btn\_Color.BackColor;  myObj.Width = width;  this.lstObject.Add(myObj);  this.paint = true;  if (bDash)  {  myObj.isDash = true;  }  }  if (bRec)  {  DrawShape myObj;  myObj = new lRectangle();  myObj.p1 = e.Location;//Set the first point of shape  myObj.Color\_ = btn\_Color.BackColor;//Set the color of shape  myObj.Width = width;//Set the width of shape  this.lstObject.Add(myObj);// Add the object to the list<DrawShape>  if (bDash)  {  myObj.isDash = true;  }  this.paint = true;  }  if (bEllipse)  {  DrawShape myObj;  myObj = new lEclipse();  myObj.p1 = e.Location;  myObj.Color\_ = btn\_Color.BackColor;  myObj.Width = width;  this.lstObject.Add(myObj);  if (bDash) myObj.isDash = true;  this.paint = true;  }  if (bTriangle)  {  ltriangle.p1 = ltriangle.p2 = e.Location;  ltriangle.Color\_ = btn\_Color.BackColor;  ltriangle.Width = width;  this.lstObject.Add(ltriangle);  this.paint = true;  if (bDash) ltriangle.isDash = true;  }  if (bPolygon)  {  lpolygon.p1 = lpolygon.p2 = e.Location;  lpolygon.Color\_ = btn\_Color.BackColor;  lpolygon.Width = width;  if (bDash) lpolygon.isDash = true;  this.lstObject.Add(lpolygon);  this.paint = true;  }  if (bStar)  {  lstar.p1 = lstar.p2 = e.Location;  lstar.Color\_ = btn\_Color.BackColor;  lstar.Width = width;  this.lstObject.Add(lstar);  if (bDash) lstar.isDash = true;  this.paint = true;  }  if (bDiamond)  {  ldiamond.p1 = e.Location;  ldiamond.Color\_ = btn\_Color.BackColor;  ldiamond.Width = width;  this.lstObject.Add(ldiamond);  if (bDash) ldiamond.isDash = true;  this.paint = true;  }  if (bCircle)  {  DrawShape myObj;  myObj = new lCircle();  myObj.p1 = e.Location;  myObj.Color\_ = btn\_Color.BackColor;  myObj.Width = width;  this.lstObject.Add(myObj);  if (bDash) myObj.isDash = true;  this.paint = true;  }  //--------------------- For Select function ---------------------  //If user lick on the Select button, it will pick a single shape. Or if user press the Ctrl button, the list of shapes will be add  if (bSelect || ModifierKeys == Keys.Control)  {  foreach (DrawShape shape in lstObject)  {  if (shape.Bound.Contains(e.Location))  {  shape.isSelected = true;//Set the shape is selected  selected\_Shape = shape; // Single shape  shape.originalLocation = e.Location; // Update the original location  //break;  }  }  }  } |

* Then the picMain\_MouseMove() event will update the second point to draw the shape

|  |
| --- |
| private void picMain\_MouseMove(object sender, MouseEventArgs e)  {  if (paint)  {  if (bPencil)  {  p2 = e.Location;  if (bDash)  {  myPen.Width = 6;  myPen.DashStyle = DashStyle.Dash;  g.DrawLine(myPen, p1, p2);  }  else  {  g.DrawLine(myPen, p1, p2);  }  p1 = p2;  }  if (bEraser)  {  p2 = e.Location;  g.DrawLine(myEraser, p1, p2);  p1 = p2;  }  if (bPencil == false && bEraser == false && lstObject.Count > 0)  {  this.lstObject[this.lstObject.Count - 1].p2 = e.Location;//Update the second point of the shape  }  }  if (bSelect && selected\_Shape != null)  {  // First we need to determine the distance between the original point to the latest point  int dx = (e.Location.X - selected\_Shape.originalLocation.X);  int dy = (e.Location.Y - selected\_Shape.originalLocation.Y);  Point newPoint = new Point(dx, dy);  picMain.Image = new Bitmap(picMain.Width, picMain.Height);    // After we have the distance, we will draw the shape  if (e.Button == MouseButtons.Left&& ModifierKeys != Keys.Control) // Single shape  {  selected\_Shape.Distance(newPoint);  selected\_Shape.Draw(g);  }  if (ModifierKeys == Keys.Control&& e.Button == MouseButtons.Left) // List of shapes  {  foreach (DrawShape shape in lstObject)  {  if (shape.isSelected)  {  shape.Distance(newPoint);  }  shape.Draw(g);  }  }  picMain.Refresh();  selected\_Shape.originalLocation = e.Location;  }  p2 = e.Location;  picMain.Refresh();  } |

- dx, dy is the variance that determine the distance from the originalLocation to the latest point, then the Distance() function in DrawShape class will be update the p1 and p2 point to move the shape.

* The picMain\_MouseUp() event.

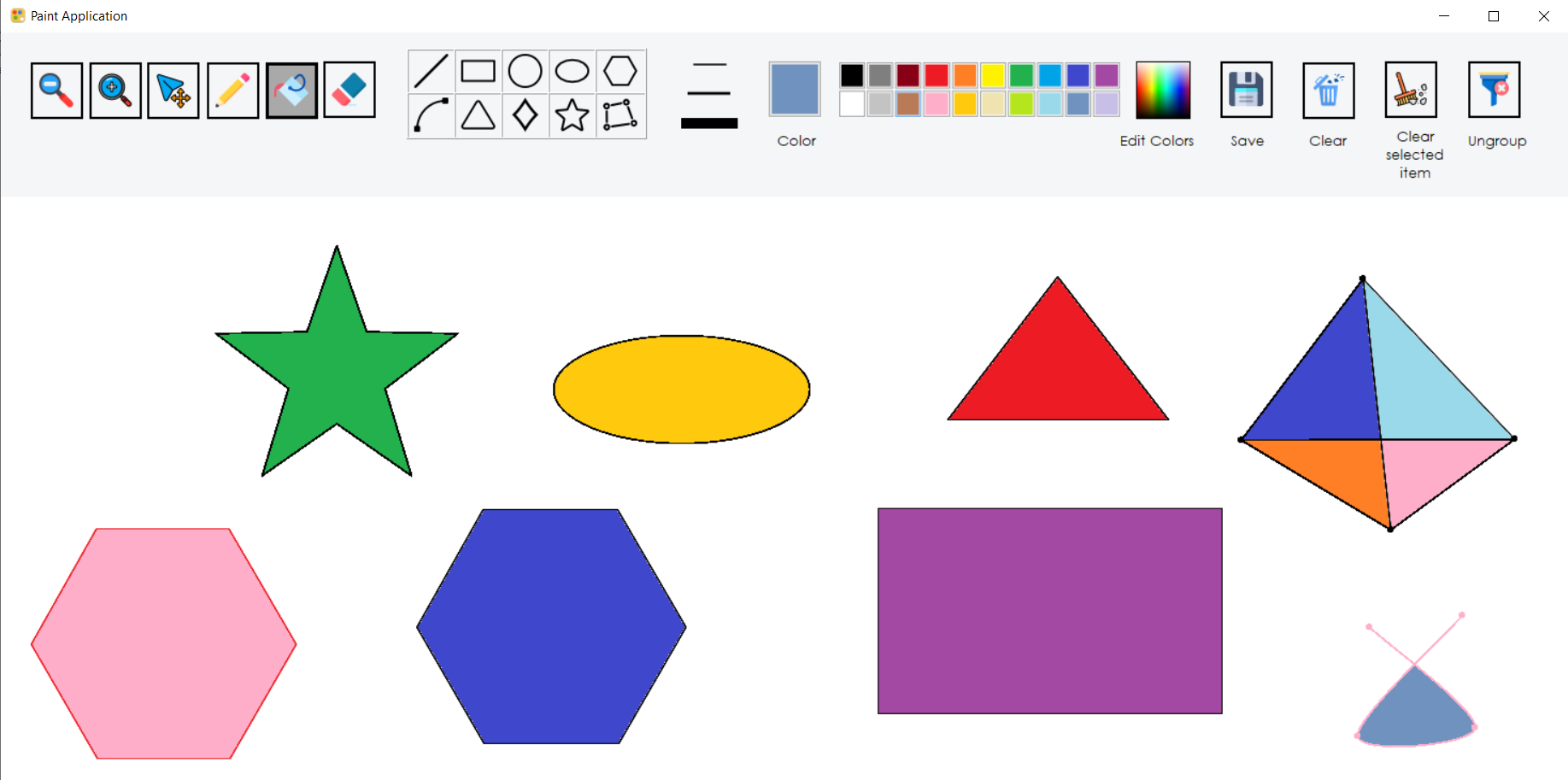
|  |
| --- |
| private void picMain\_MouseUp(object sender, MouseEventArgs e)  {  paint = false;  for (int i = 0; i < this.lstObject.Count; i++)  {  if (bArc == false)  {  this.lstObject[i].Draw(g);// Draw the object when the Mouse is up  }  }  if (bSelect && selected\_Shape != null)  {  selected\_Shape = null; // Drop the shape have just pick down  }  } |

* The picMain\_Paint() event will draw the element lstObject (Rectangle, Circle, Ellipse,…)

|  |
| --- |
| private void picMain\_Paint(object sender, PaintEventArgs e)  {  Graphics ga = e.Graphics;  for (int i = 0; i < this.lstObject.Count; i++)  {  this.lstObject[i].Draw(ga);  }  } |

- With Filled ellipse, filled rectangle, filled circle, filled polygon. I use the Fill fuction, with can be fill any shape no matter what kind of shape.

|  |
| --- |
| //First we need to determine the location of the point we have just click  static Point set\_point(PictureBox pb, Point pt)  {  float pX = 1f \* pb.Image.Width / pb.Width;  float pY = 1f \* pb.Image.Height / pb.Height;  return new Point((int)(pt.X \* pX), (int)(pt.Y \* pY));  }  //Find the current Color  private void Validate(Bitmap bmp, Stack<Point> sp, int x, int y, Color Old\_Color, Color new\_Color)  {  Color cv = bmp.GetPixel(x, y);  if (cv == Old\_Color)  {  sp.Push(new Point(x, y));  bmp.SetPixel(x, y, new\_Color);  }  }  //Fill method  public void Fill1(Bitmap bmp, int x, int y, Color new\_Color)  {  try  {  Color old\_color = bmp.GetPixel(x, y);  Stack<Point> pixels = new Stack<Point>();  pixels.Push(new Point(x, y));  bmp.SetPixel(x, y, new\_Color);  if (old\_color == new\_Color) return;  while (pixels.Count > 0)  {  Point pt = (Point)pixels.Pop();  if (pt.X > 0 && pt.Y > 0 && pt.X < bmp.Width - 1 && pt.Y < bmp.Height - 1)  {  Validate(bmp, pixels, pt.X - 1, pt.Y, old\_color, new\_Color);  Validate(bmp, pixels, pt.X, pt.Y - 1, old\_color, new\_Color);  Validate(bmp, pixels, pt.X + 1, pt.Y, old\_color, new\_Color);  Validate(bmp, pixels, pt.X, pt.Y + 1, old\_color, new\_Color);  }  }  }  catch (Exception)  {  throw;  }    } |



**+ Develop other functions allowing users are able to adjust Pen properties (color, width, dash style), and Brush properties (color, style).**

- To adjust Pen properties (color, width, dash style) we need to declare the Color\_, Width in the DrawShape class to the other shapes can adjust the Color or the Width or the Dash Style.

|  |
| --- |
| public abstract class DrawShape  {  public Point p1;  public Point p2;  public Point originalLocation;  // To serve the Select and Group function  //Adjust Color and Width  public Color Color\_ { get; set; }  public float Width { get; set; }  public bool isSelected { get; set; }  // To determine that whether this shape is selected?  public bool isDash { get; set; }  // A function to create a bound of a shape, we use it to select shape  public Rectangle Bound {  get  {  int x = Math.Min(p1.X, p2.X);  int y = Math.Min(p1.Y, p2.Y);  int width = Math.Abs(p1.X - p2.X);  int height = Math.Abs(p1.Y - p2.Y);  return new Rectangle(x, y, width, height);  }    }  // A function to move the shape with the lastet point  public virtual void Distance(Point point)  {  p1 = new Point((p1.X + point.X), (p1.Y + point.Y));  p2 = new Point((p2.X + point.X), (p2.Y + point.Y));  }  // An abstract function to draw all of shape, we use it for Paint Event  public abstract void Draw(Graphics gp);  } |

Then in the picMain we create a width button and pictureBox\_Color also the Colordialog if the client want to create their own color.

* Width button:



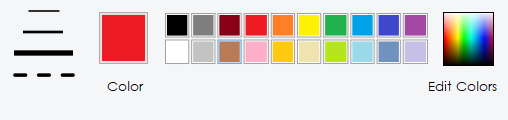
(From my computer)

* pictureBox\_Color and Colordialog

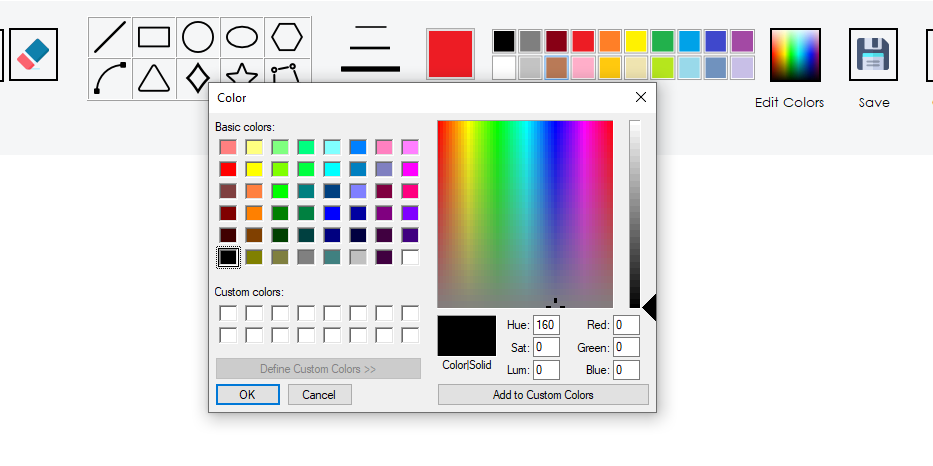


(From my computer)

When the client click on the pictureBox\_Color and Colordialog, it will change the color of the shape and we also use this color to fill our shapes.



(From my computer)



(From my computer)

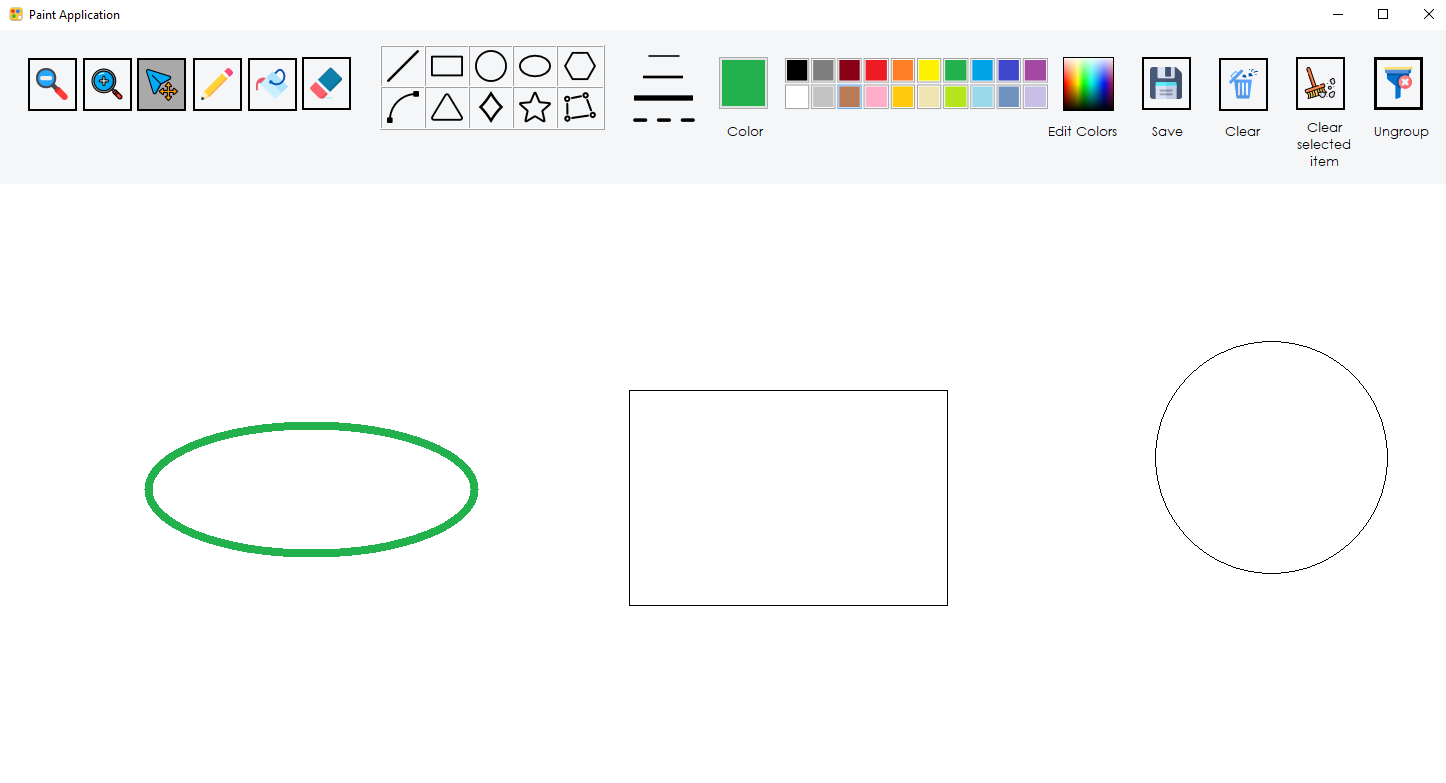
|  |
| --- |
| //-------------------- Function for edit color and fill color --------------------  private void pictureBox\_Colors\_MouseClick(object sender, MouseEventArgs e)  {  Point point = set\_point(pictureBox\_Colors, e.Location);  pictureBox\_Colors.BackColor = ((Bitmap)pictureBox\_Colors.Image).GetPixel(point.X, point.Y);  btn\_Color.BackColor = pictureBox\_Colors.BackColor;  }  private void btn\_EditColor\_Click(object sender, EventArgs e)  {  ColorDialog cd = new ColorDialog();  cd.AllowFullOpen = true;  cd.FullOpen = true;  cd.AnyColor = true;  if (cd.ShowDialog() == DialogResult.OK)  {  btn\_Color.BackColor = cd.Color;  }  }  // -------------------- Witdh and Color Button --------------------  private void btn\_thinLine\_Click(object sender, EventArgs e)  {  width = 1;  bDash = false;  }  private void btn\_mediumLine\_Click(object sender, EventArgs e)  {  width = 4;  bDash = false;  }  private void button4\_Click(object sender, EventArgs e)  {  width = 8;  bDash = false;  }  private void btn\_DashStyle\_Click(object sender, EventArgs e)  {  bDash = true;  } |

**+ Select and drag the graphic objects.**

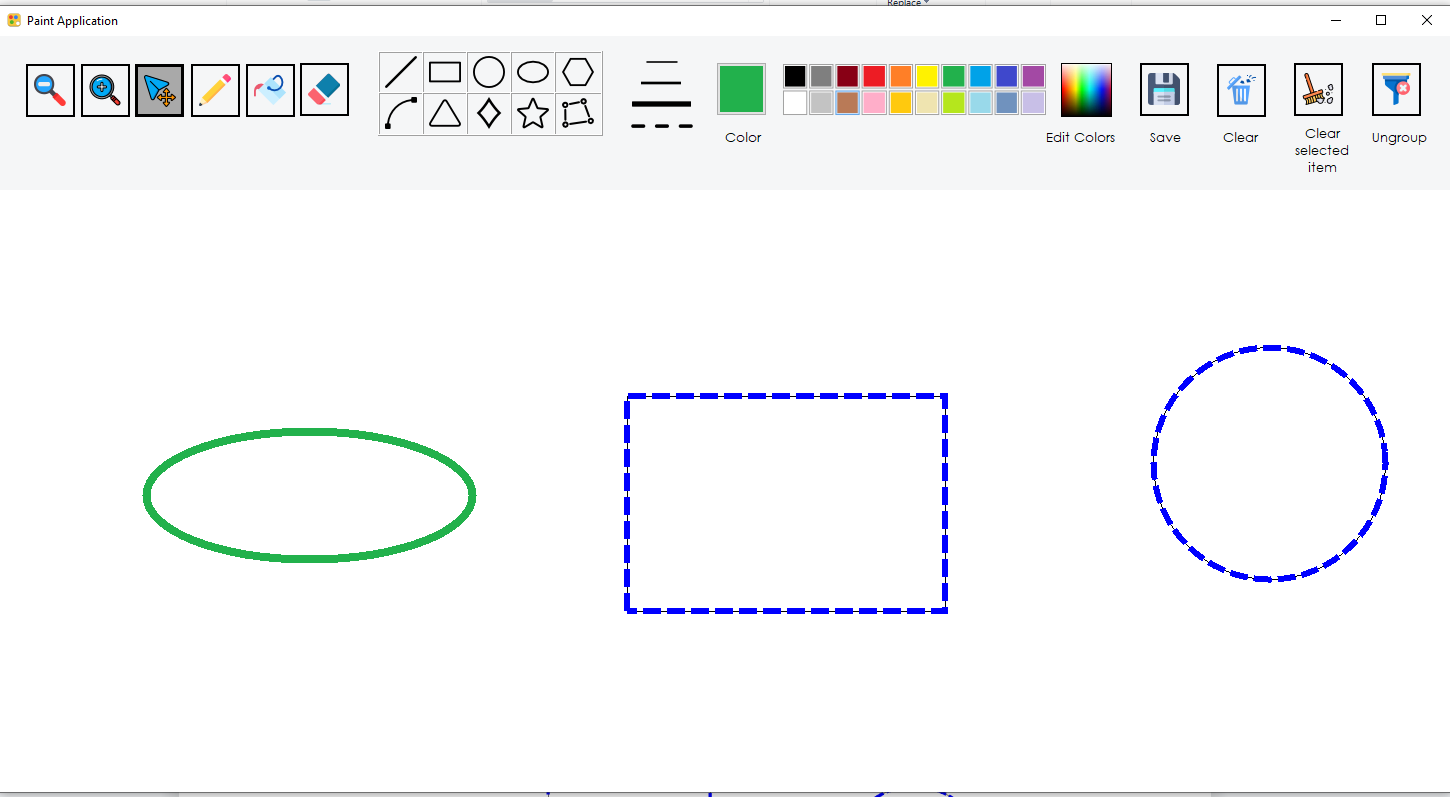
When the client click on the select button, the program will turn the select flag.

|  |
| --- |
| private void button1\_Click(object sender, EventArgs e)  {  Refresh();  RefreshColor();  button1.BackColor = Color.DarkGray;  bSelect = true;  } |

- Before:



-After:



- To select the shape we need to have a **Bound function** which is made from a rectangle. That rectangle takes the p1 and p2 of the shape to make a bound over the shape, then when the user click on Select button, bSelect is set to true and the **Bound function** will check whether the client is clicking on this shape, if the client is click on the shape then the bool variance **isSelected** will be set to be true.

|  |
| --- |
| public bool isSelected { get; set; }  // To determine that whether this shape is selected?  // A function to create a bound of a shape, we use it to select shape  public Rectangle Bound {  get  {  int x = Math.Min(p1.X, p2.X);  int y = Math.Min(p1.Y, p2.Y);  int width = Math.Abs(p1.X - p2.X);  int height = Math.Abs(p1.Y - p2.Y);  return new Rectangle(x, y, width, height);  }    } |

- MouseDown event:

|  |
| --- |
| //--------------------- For Select function ---------------------  //If user lick on the Select button, it will pick a single shape. Or if user press the Ctrl button, the list of shapes will be add  if (bSelect || ModifierKeys == Keys.Control)  {  foreach (DrawShape shape in lstObject)  {  if (shape.Bound.Contains(e.Location))  {  shape.isSelected = true;//Set the shape is selected  selected\_Shape = shape; // Single shape  shape.originalLocation = e.Location; // Update the original location  //break;  }  }  } |

- MouseMove event:

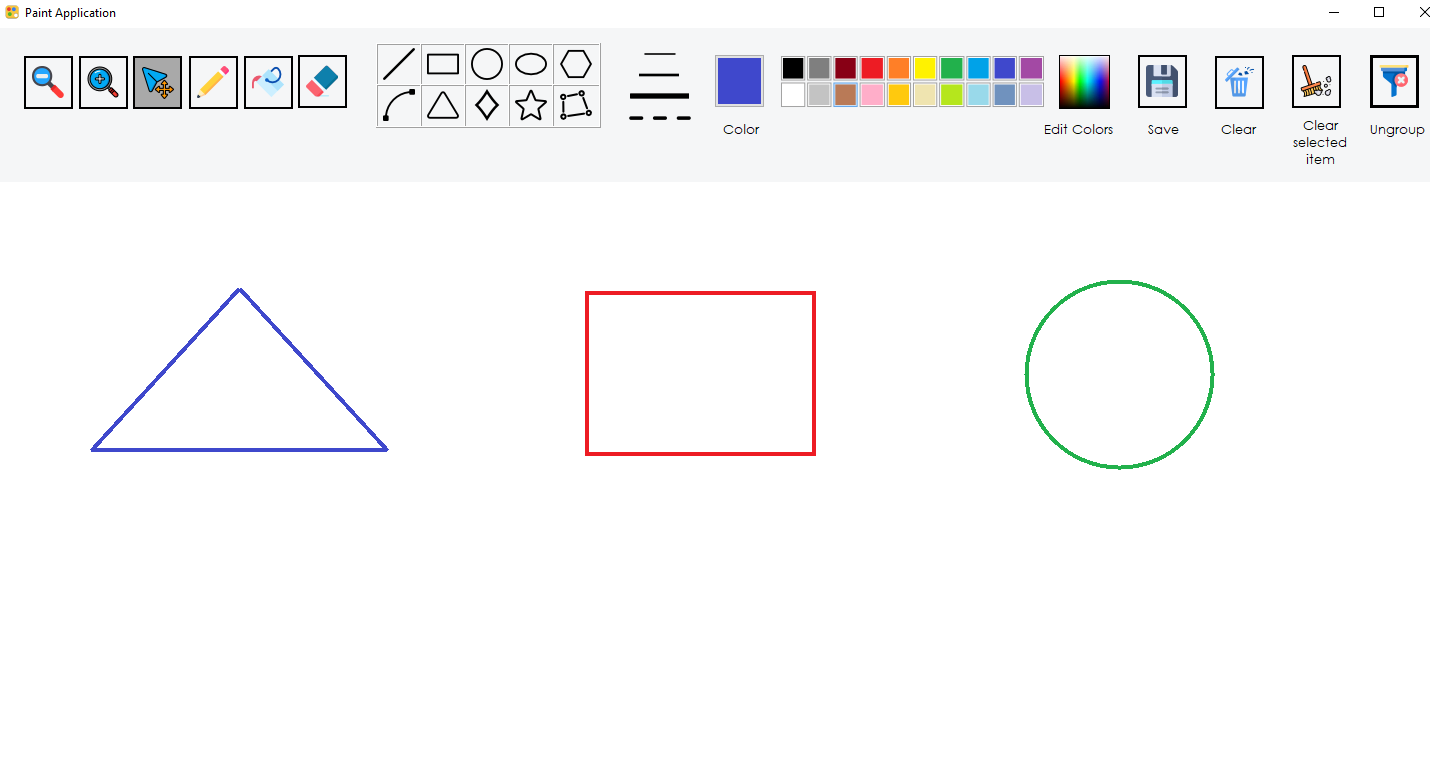
|  |
| --- |
| if (bSelect && selected\_Shape != null)  {  // First we need to determine the distance between the original point to the latest point  int dx = (e.Location.X - selected\_Shape.originalLocation.X);  int dy = (e.Location.Y - selected\_Shape.originalLocation.Y);  Point newPoint = new Point(dx, dy);  picMain.Image = new Bitmap(picMain.Width, picMain.Height);    // After we have the distance, we will draw the shape  if (e.Button == MouseButtons.Left&& ModifierKeys != Keys.Control) // Single shape  {  selected\_Shape.Distance(newPoint);  selected\_Shape.Draw(g);  }  if (ModifierKeys == Keys.Control&& e.Button == MouseButtons.Left) // List of shapes  {  foreach (DrawShape shape in lstObject)  {  if (shape.isSelected)  {  shape.Distance(newPoint);  }  shape.Draw(g);  }  }  picMain.Refresh();  selected\_Shape.originalLocation = e.Location;  } |

- When we want to move the shape, we need to determine the distance between the original point to the latest point. After we have the distance, we can move the shape to the latest point

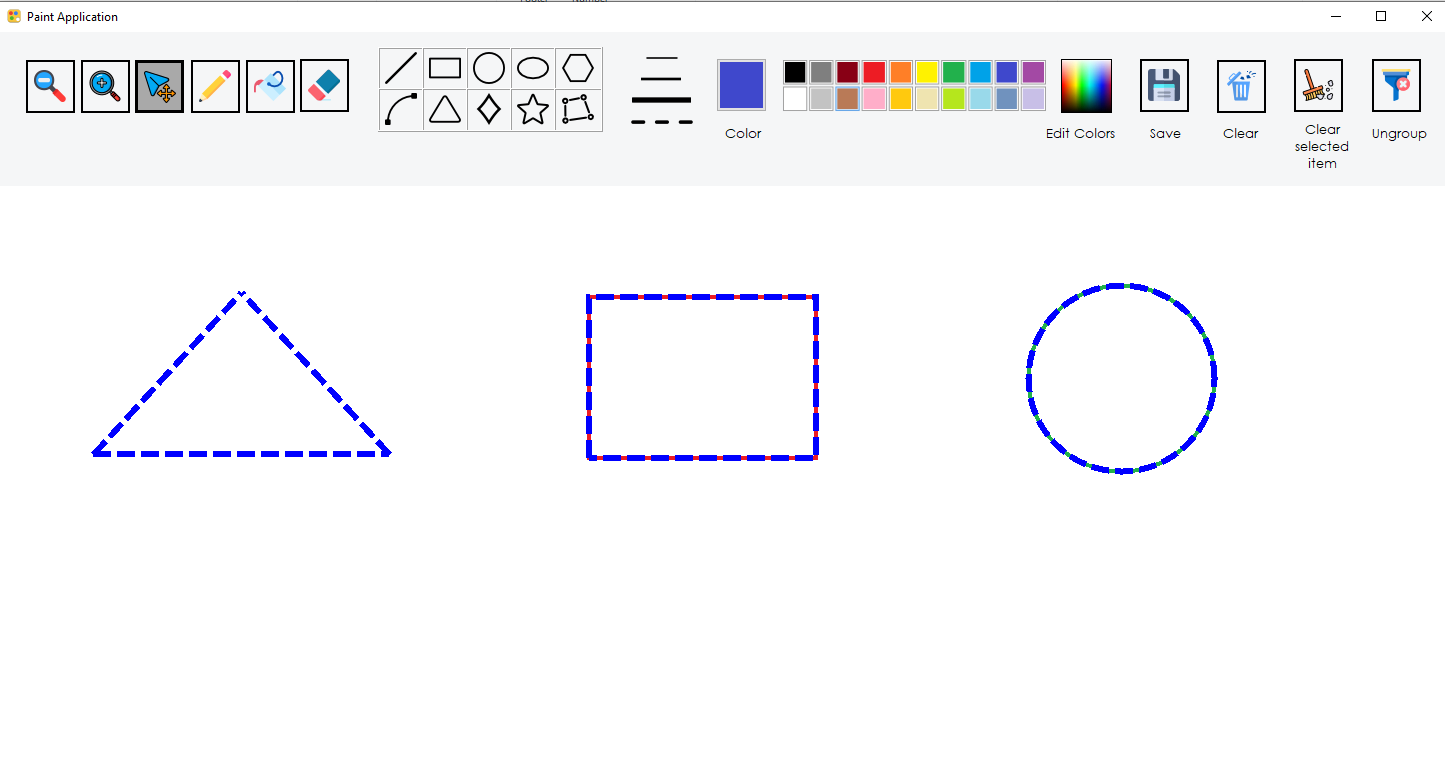
- Distance method():

|  |
| --- |
| // A function to move the shape with the lastet point  public virtual void Distance(Point point)  {  p1 = new Point((p1.X + point.X), (p1.Y + point.Y));  p2 = new Point((p2.X + point.X), (p2.Y + point.Y));  } |

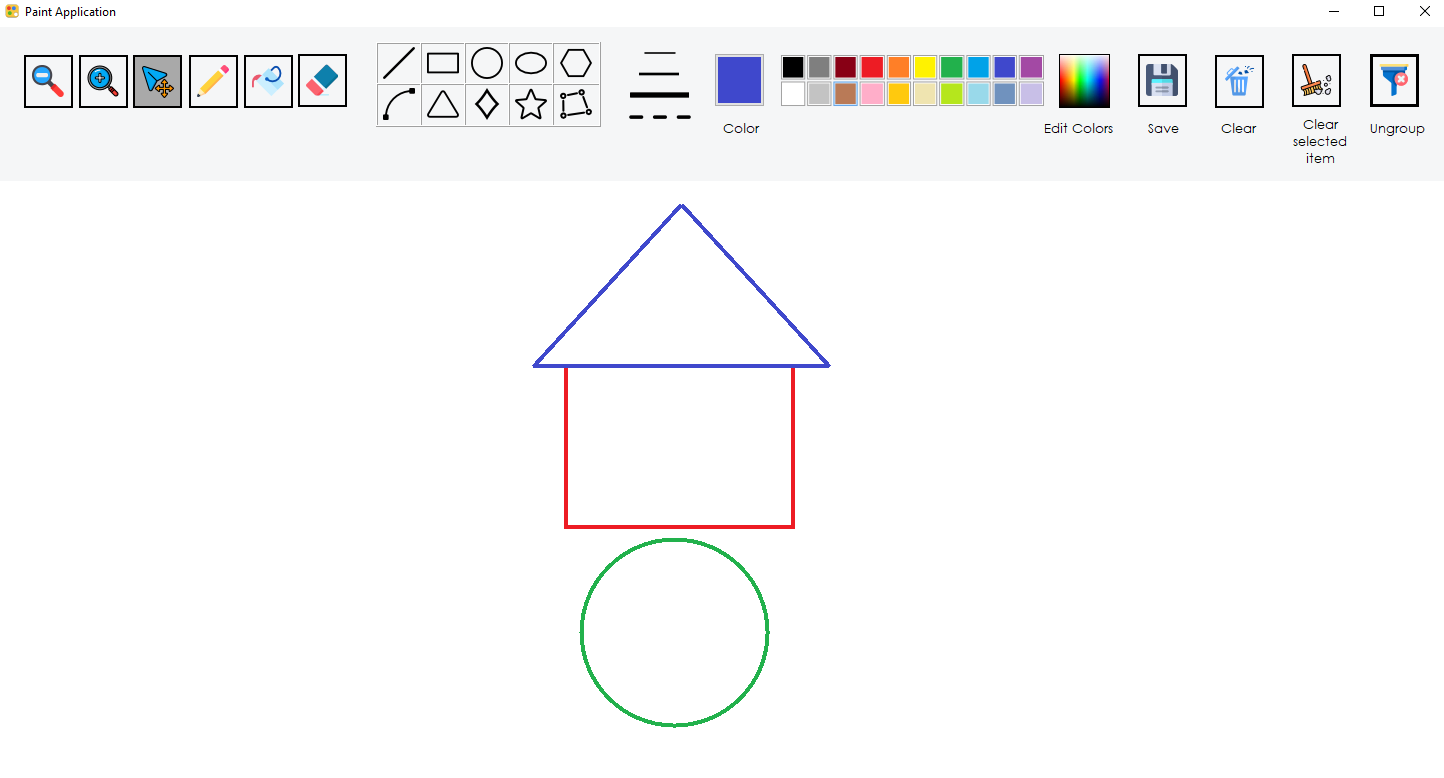
-Before:



- Select:

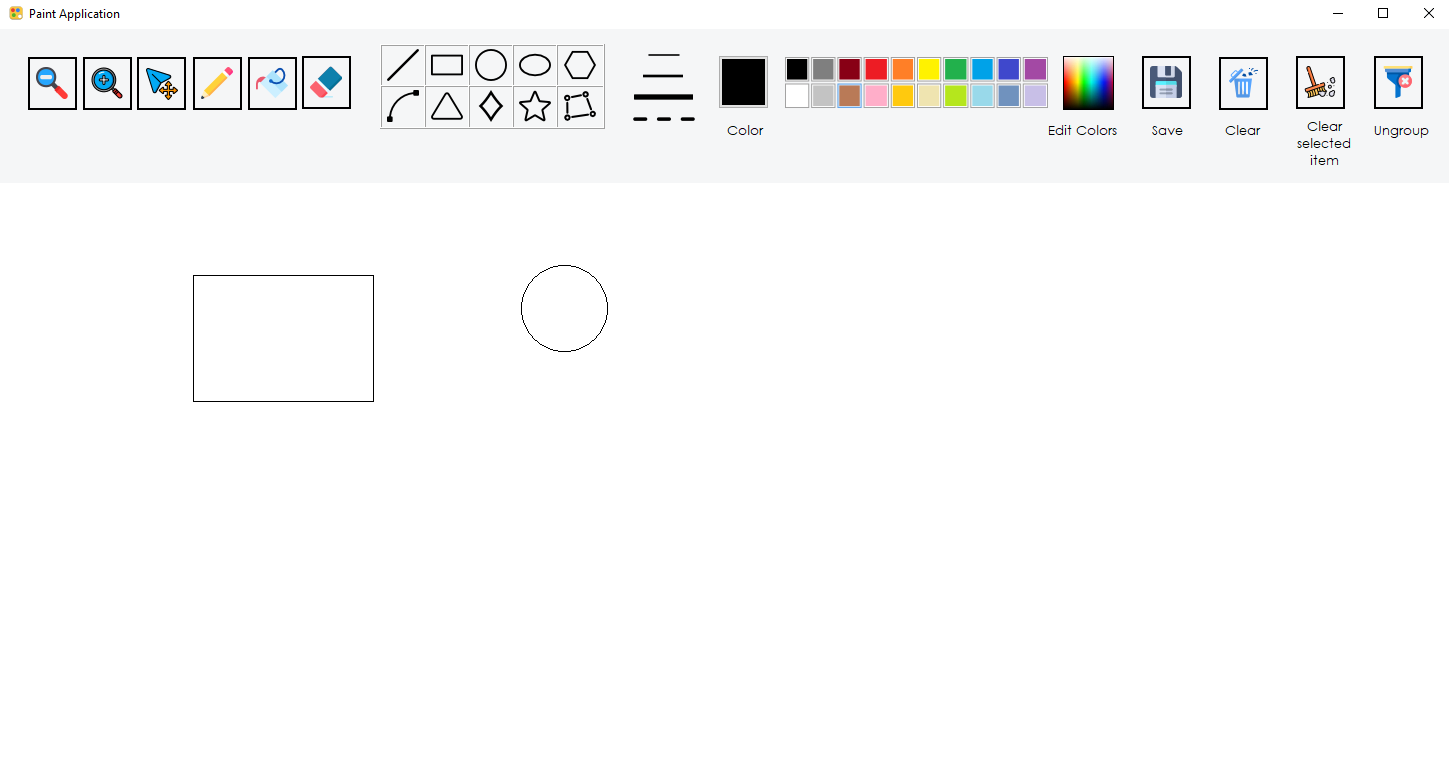


-Drag:

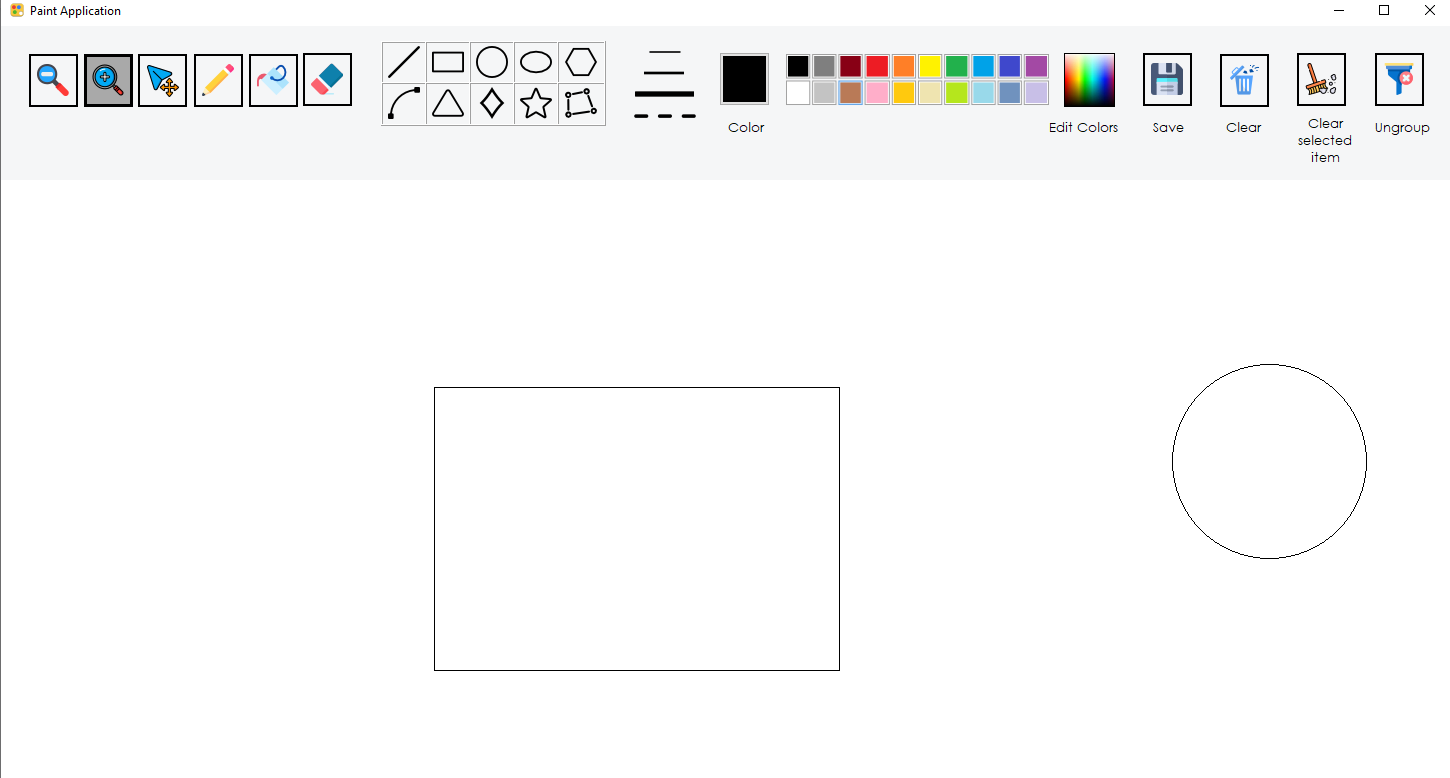


**+ Zoom in and zoom out to change the size of the objects.**

- Before:



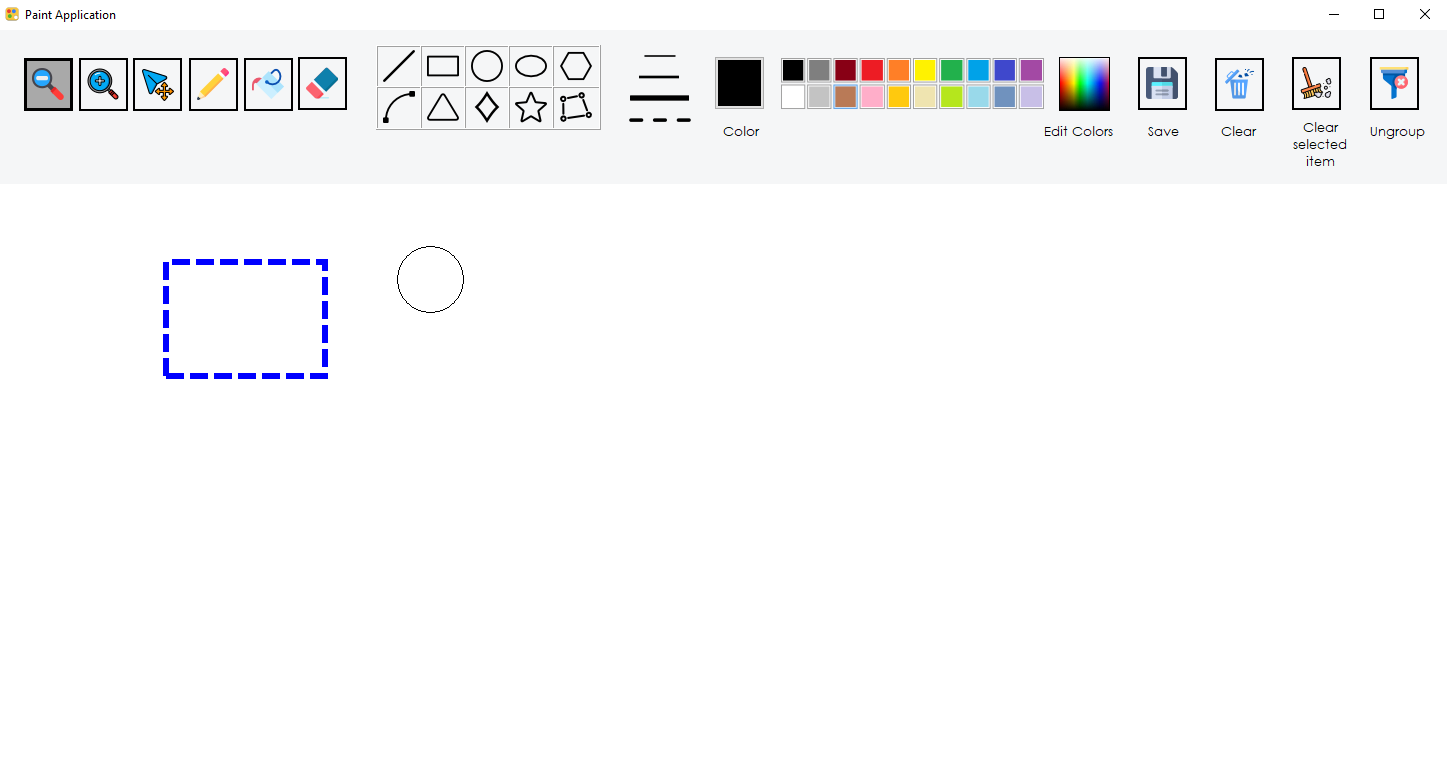
(From my computer)

-After: 

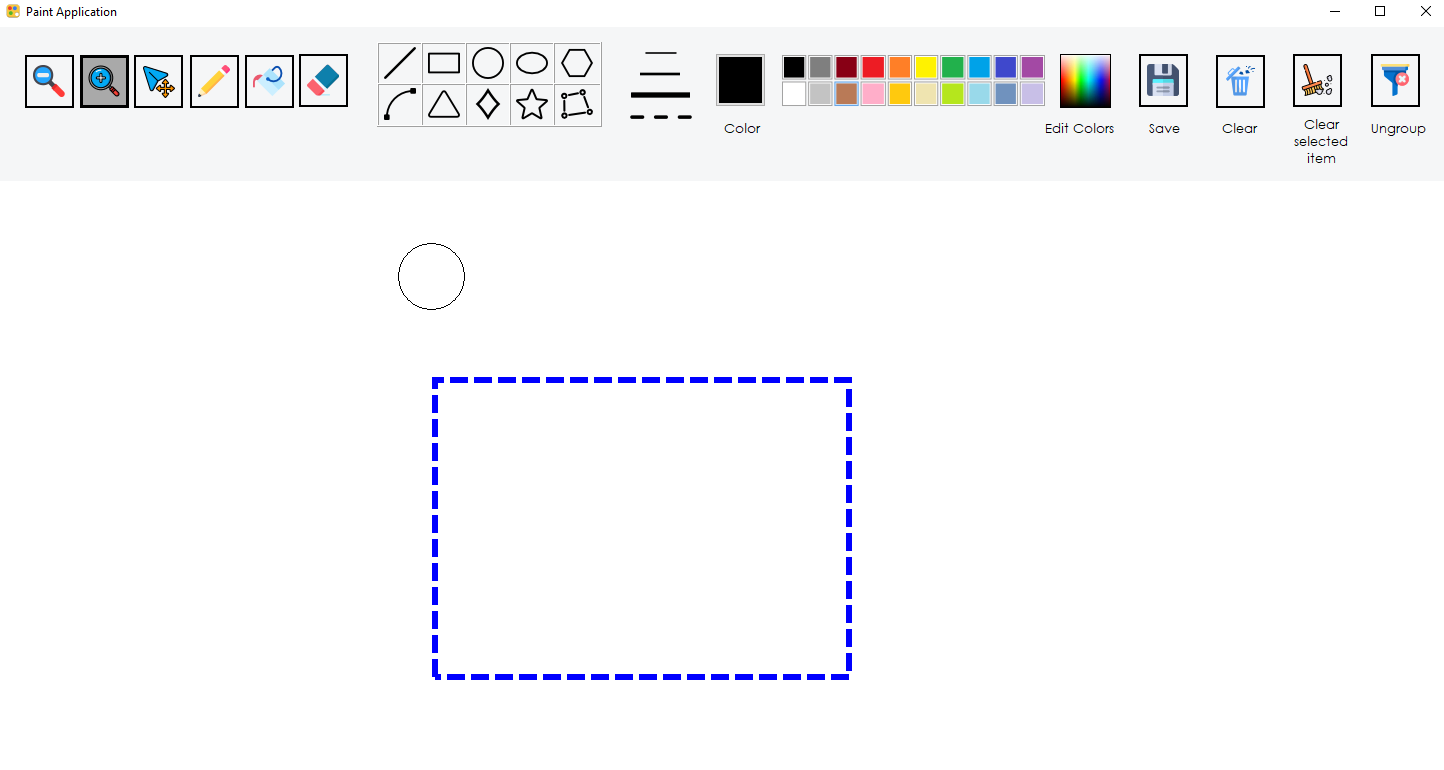
(From my computer)

- We can also zoom out and zoom in the selected shape only

- Before:

(From my computer)

-After:



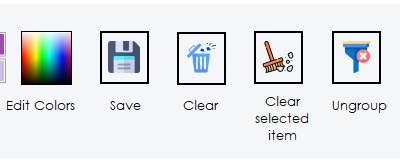
(From my computer)

|  |
| --- |
| private void btn\_Zoom\_in\_Click(object sender, EventArgs e)  {  RefreshColor();  btn\_Zoom\_in.BackColor = Color.DarkGray;  picMain.Image = new Bitmap(picMain.Width, picMain.Height);  zoom = 1.5f;  var selectedShapes = lstObject.Where(s => s.isSelected).ToList();  if (bSelect && selectedShapes.Any())  {  foreach (var shape in selectedShapes)  {  shape.p1.X = (int)(shape.p1.X \* zoom);  shape.p1.Y = (int)(shape.p1.Y \* zoom);  shape.p2.Y = (int)(shape.p2.Y \* zoom);  shape.p2.X = (int)(shape.p2.X \* zoom);  }  }  else  {  foreach (DrawShape shape in lstObject)  {  shape.p1.X = (int)(shape.p1.X \* zoom);  shape.p1.Y = (int)(shape.p1.Y \* zoom);  shape.p2.Y = (int)(shape.p2.Y \* zoom);  shape.p2.X = (int)(shape.p2.X \* zoom);  }  }  picMain.Invalidate();  }  private void btn\_Zoom\_out\_Click(object sender, EventArgs e)  {  RefreshColor();  btn\_Zoom\_out.BackColor = Color.DarkGray;  zoom = 0.7f;  var selectedShapes = lstObject.Where(s => s.isSelected).ToList();  if (bSelect && selectedShapes.Any())  {  foreach (var shape in selectedShapes)  {  shape.p1.X = (int)(shape.p1.X \* zoom);  shape.p1.Y = (int)(shape.p1.Y \* zoom);  shape.p2.Y = (int)(shape.p2.Y \* zoom);  shape.p2.X = (int)(shape.p2.X \* zoom);  }  }  else  {  foreach (DrawShape shape in lstObject)  {  shape.p1.X = (int)(shape.p1.X \* zoom);  shape.p1.Y = (int)(shape.p1.Y \* zoom);  shape.p2.Y = (int)(shape.p2.Y \* zoom);  shape.p2.X = (int)(shape.p2.X \* zoom);  }  }  picMain.Invalidate();  } |

- I set a float variance zoom to resize the point p1,p2 of the shapes, then they will change the size of them. It also can zoom I and zoom iut the selected shape only

1. **Advanced functions**

**+ Remove one or many selected objects (Using Ctr + mouse click to select many objects)**



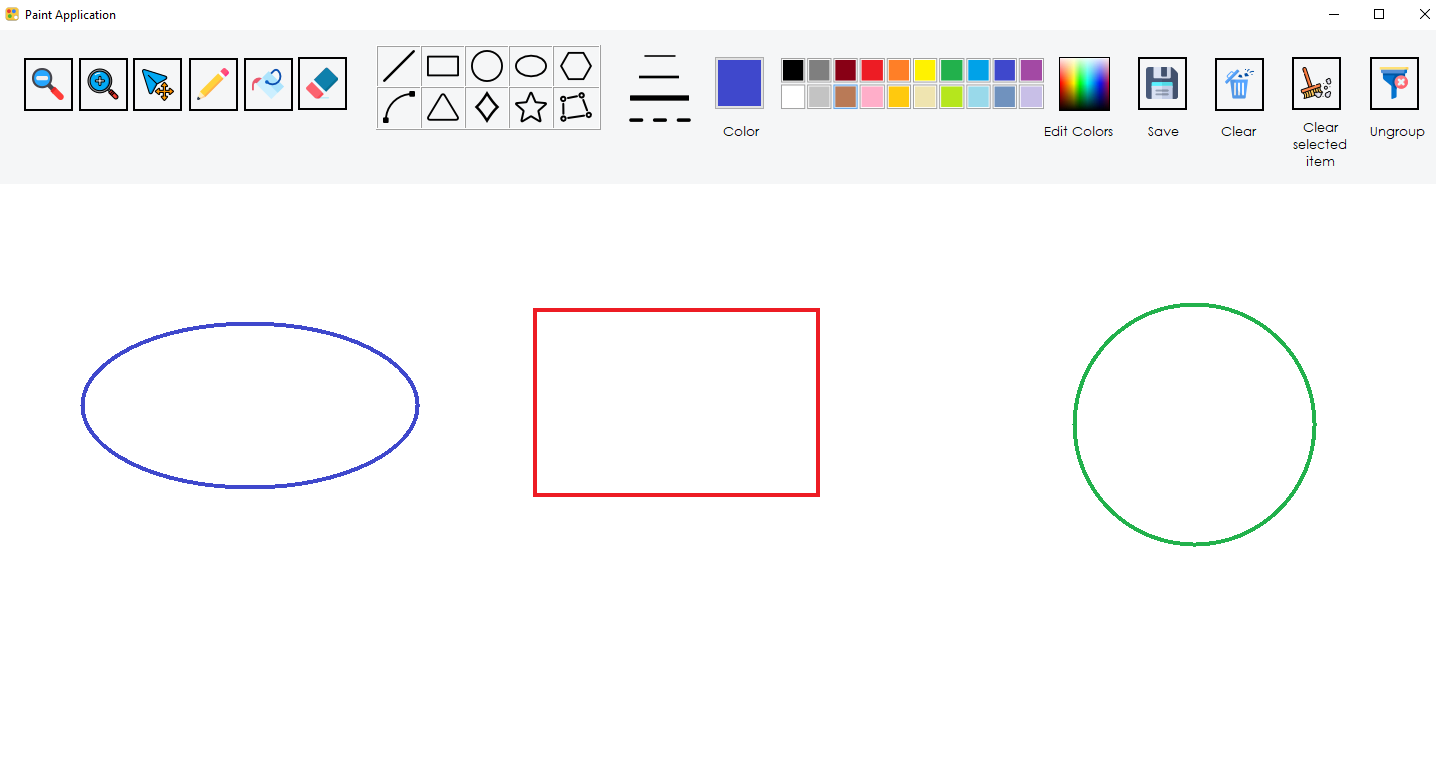
- I have clear all and clear the selected item only:

|  |
| --- |
| private void btn\_Clear\_Click(object sender, EventArgs e)  {  g.Clear(Color.White);  picMain.Image = bm;  Refresh();  lstObject.Clear();  larc.points.Clear();  New\_Polygon.points.Clear();  lselected\_Shape.Clear();  }  private void btn\_Clear\_selected\_Item\_Click(object sender, EventArgs e)  {  var selectedShapes = lstObject.Where(s => s.isSelected).ToList();  if (selectedShapes.Any())  {  foreach (var shape in selectedShapes)  {  lstObject.Remove(shape);  }  picMain.Invalidate();  }  g.Clear(Color.White);  picMain.Image = bm;  } |

- I use the LinQ “var selectedShapes = lstObject.Where(s =>s.isSelected).ToList();”

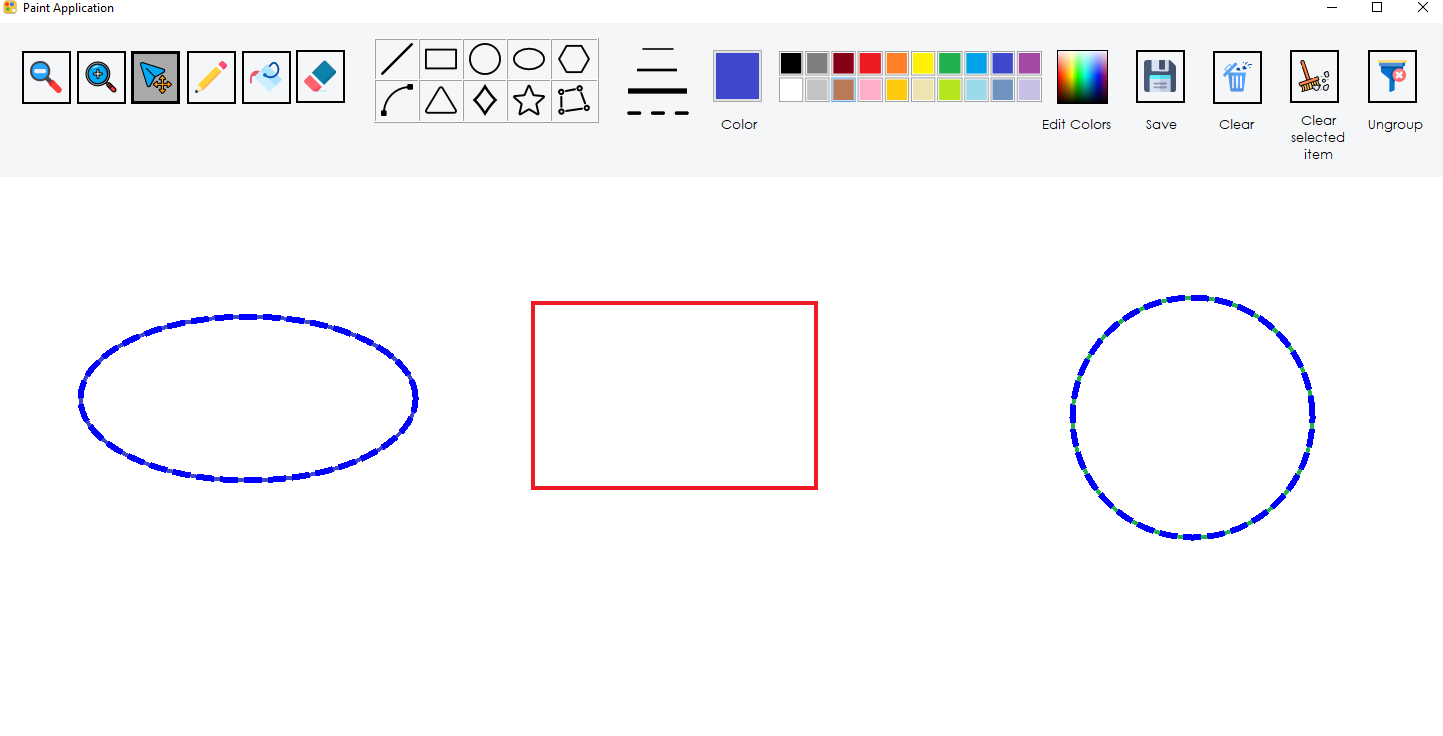
to query and group the shape is selected. Then when I identify the selected shape, I can remove it from the lstObject.

- Before:



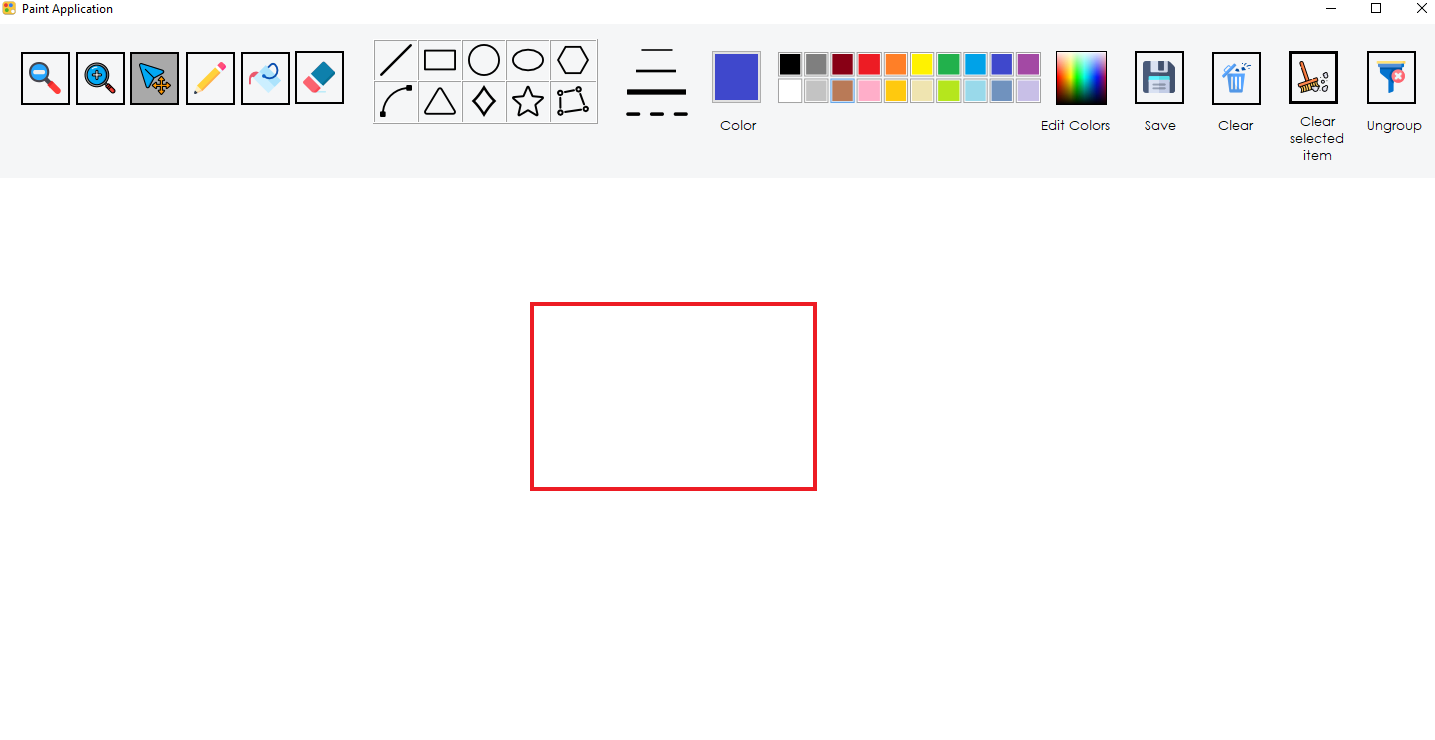
(From my computer)

- Select:



(From my computer)

-Remove:



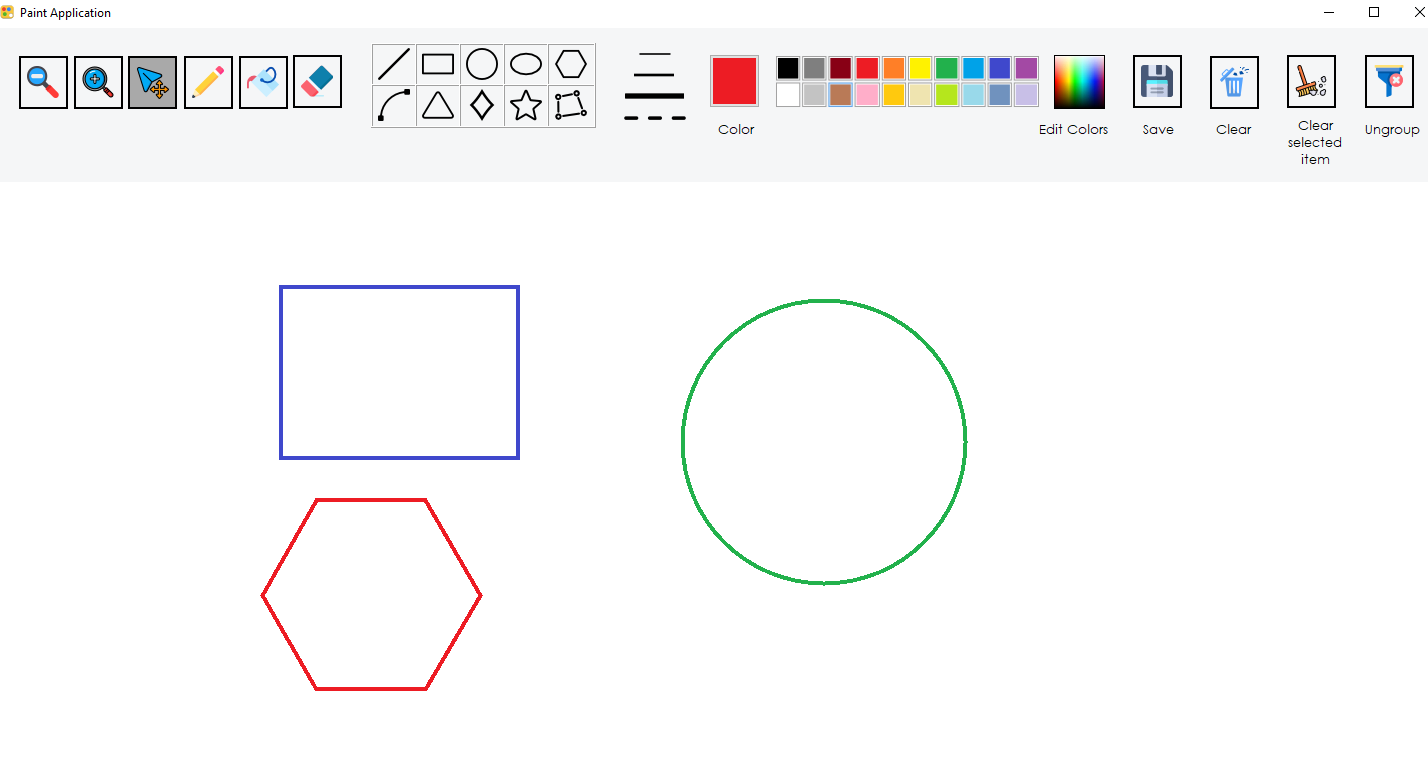
(From my computer)

**+ Group and ungroup the objects.**

|  |
| --- |
| private void btn\_Ungroup\_Click(object sender, EventArgs e)  {  foreach (DrawShape shape in lstObject)  {  shape.isSelected = false;  picMain.Refresh();  }  g.Clear(Color.White);  picMain.Image = bm;  } |

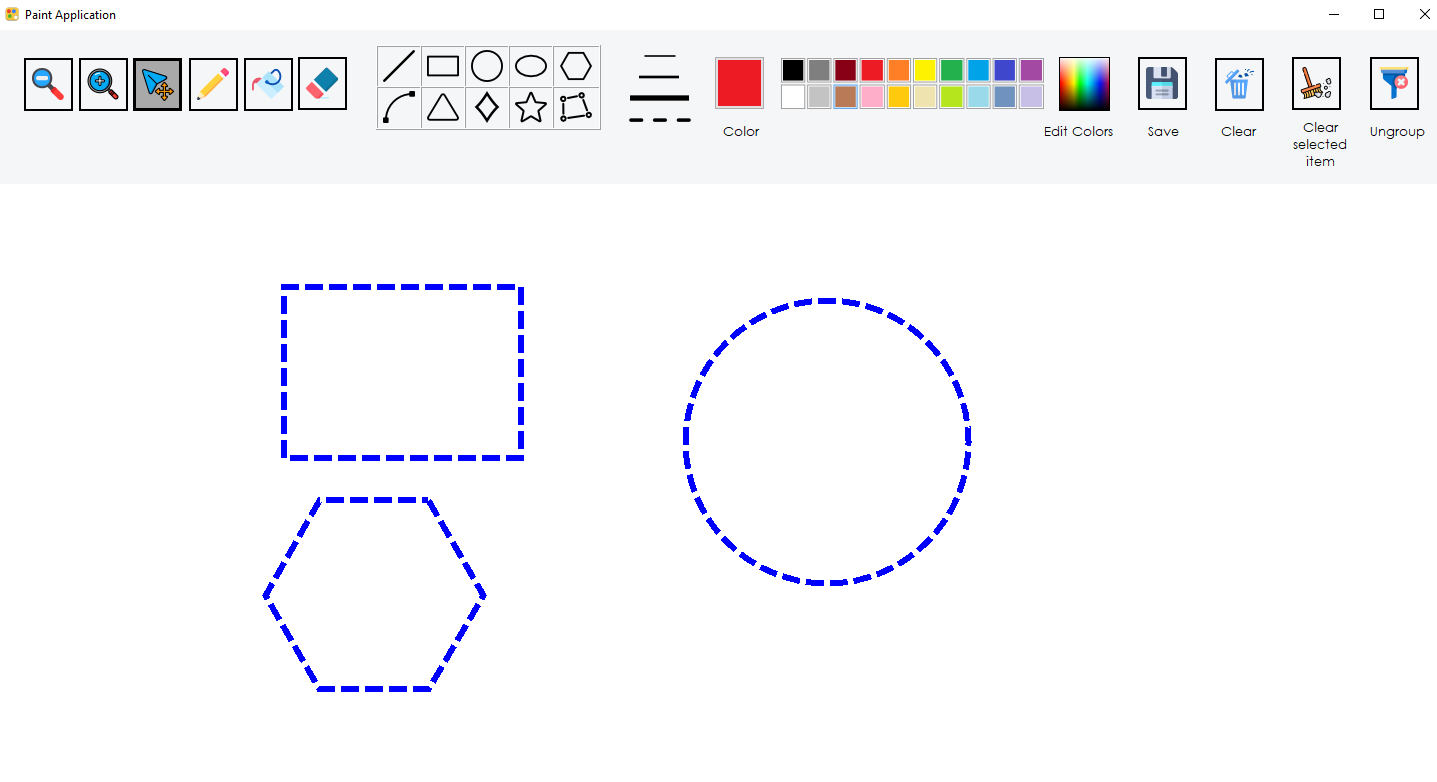
- I use the LinQ “var selectedShapes = lstObject.Where(s =>s.isSelected).ToList();” to group so for Ungroup I only need to set the field of all shapes “shape.isSelected = false;”

-Before:



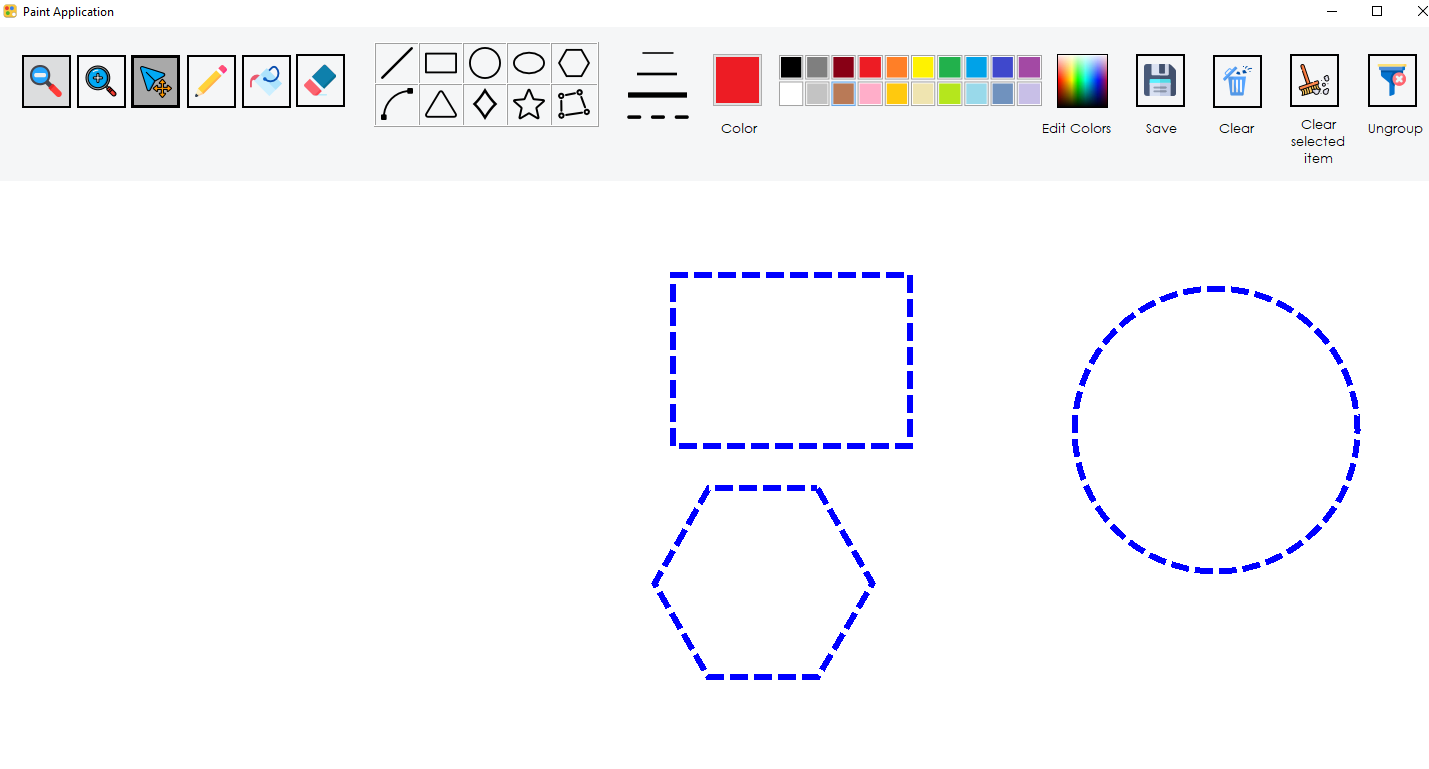
(From my computer)

-Select:



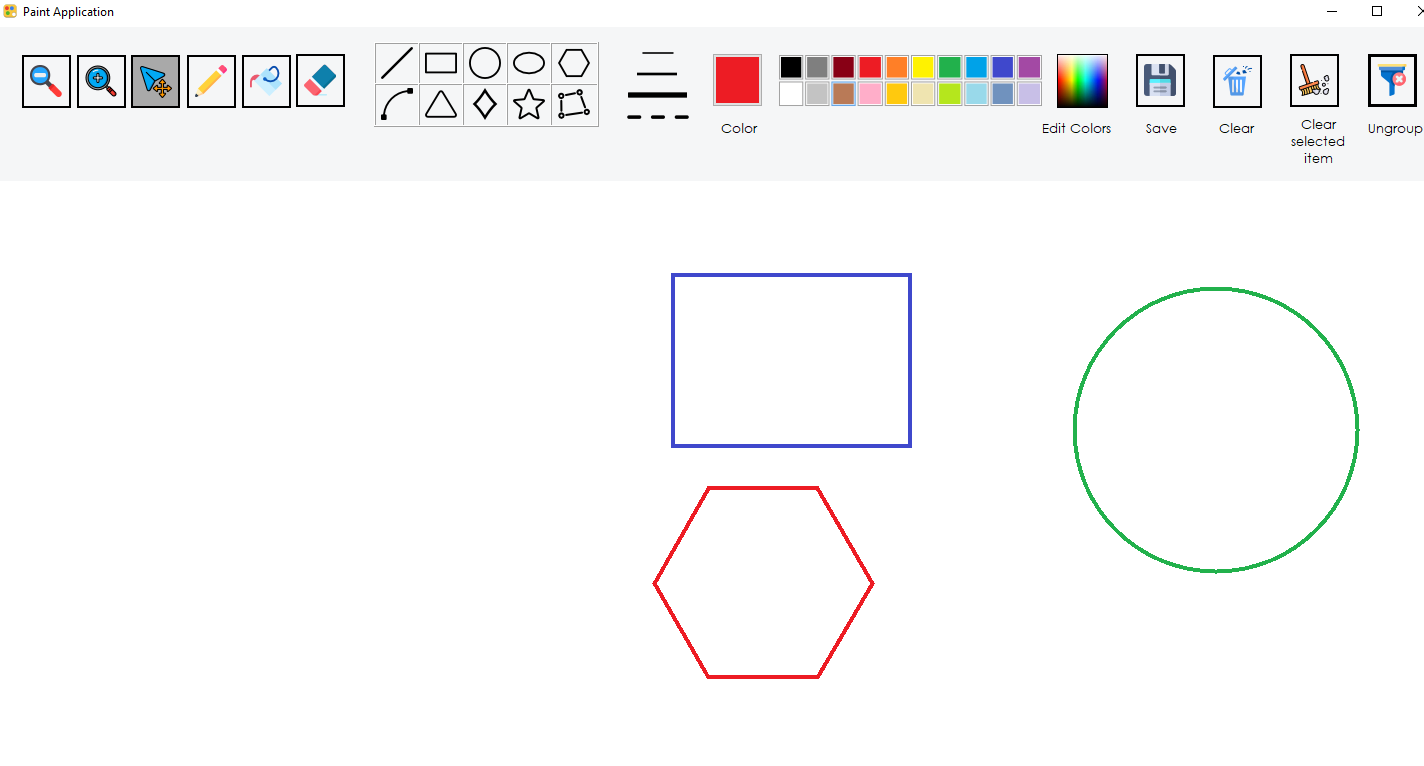
(From my computer)

- Using Ctrl + Mouse Left to drag the selected shapes



(From my computer)

- Ungroup:



(From my computer)