ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



Xây dựng ứng dụng Paint

Môn học: Lập trình trực quan.

Lớp: IT008.J11.

Giảng viên: Huỳnh Tuấn Anh.

Danh sách thành viên:

|  |  |
| --- | --- |
| Họ & tên | MSSV |
| Đỗ Ngọc Cường (Nhóm trưởng) | 18520542 |
| Đỗ Ngọc Thành | 18520358 |
| Nguyễn Trần Hoàng Hiếu | 18520054 |

Mục lục

[**I.** **Thanh tiêu đề** 3](#_Toc533725116)

[I.1. Save 3](#_Toc533725117)

[I.2. Undo 4](#_Toc533725118)

[I.3. Redo 5](#_Toc533725119)

I.4. Close…………………………………………………………………………………………………………………………………………5

[**II.** **File Menu** 5](#_Toc533725120)

[II.1. New 5](#_Toc533725121)

[II.2. Open 6](#_Toc533725122)

[II.3. Save và Save as 8](#_Toc533725123)

[**III.** **Thanh công cụ** 10](#_Toc533725127)

[III.1. Tool 10](#_Toc533725128)

[III.1.a. Pencil 10](#_Toc533725129)

[III.1.b. Eraser 11](#_Toc533725130)

III.1.c.Brush………………………………………………………………………………………………………………………………………11

[II.2.. Tools 13](#_Toc533725135)

[II.2.a. Pencil 13](#_Toc533725136)

[II.2.b. Eraser 13](#_Toc533725137)

[II.2.c. Brush 14](#_Toc533725138)

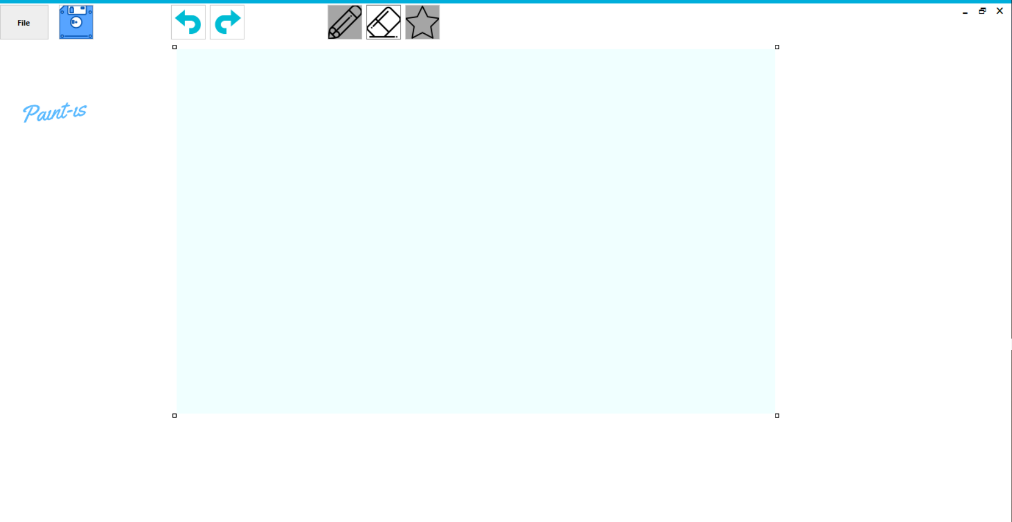
[II.2.d. Color picker 14](#_Toc533725139)

[III.. Fill tool 18](#_Toc533725143)

III.Shape…………………………………………………………………………………………………19

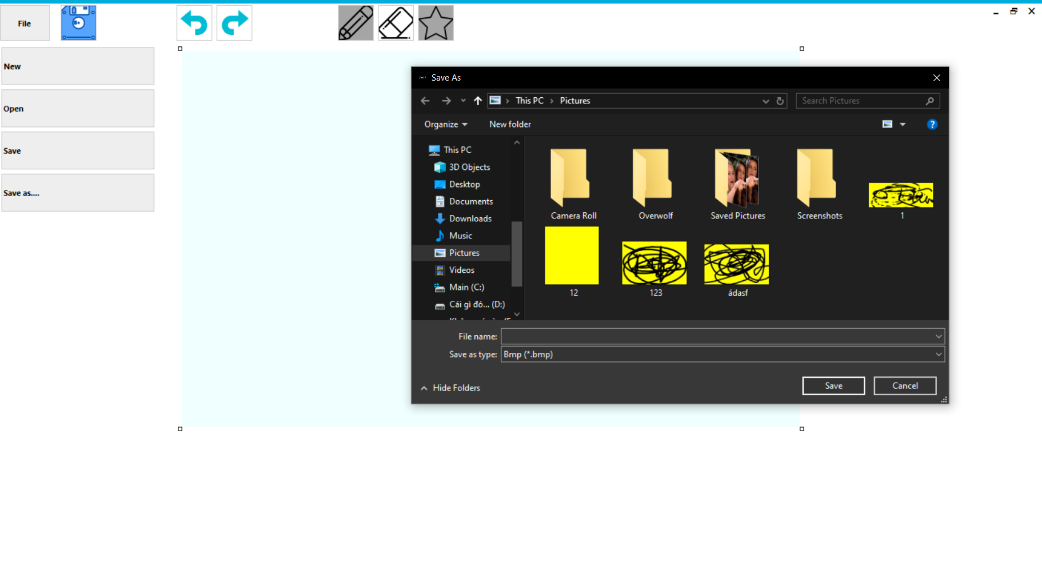
[III.6. Size 20](#_Toc533725144)

[III.7. Colors 21](#_Toc533725145)



Hình . GIao diện chính của ứng dụng.

1. **Thanh tiêu đề**
2. Save

* Ảnh mình họa: 
* Mô tả:

|  |  |
| --- | --- |
| Save | Save as |
| Lưu file hình ảnh mới khi chưa được lưu lần đầu hoặc lưu các thay đổi hình ảnh với tên cũ. | Lưu một file hình ảnh đang tồn tại với tên khác, có thể bao gồm vị trí và kiểu dữ liệu khác |

Hình . Giao diện lưu ảnh.

* Cách xử lý: sử dụng SaveFileDialog, dựa vào dữ liệu dialog trả về để xử lý, thay đổi tên form. Thêm bool Saved/Changed để xác định hình ảnh đã lưu chưa.
* Code xử lý:
* private void FileUC1\_SaveAsClicked(object sender, EventArgs e)
* {
* SaveFileDialog f = new SaveFileDialog();
* f.Filter = "Bmp (\*.bmp)|\*.bmp|Jpg (\*.jpg)|\*.jpg|Jpeg (\*.jpeg)|\*.jpeg|Png (\*.png)|\*.png";
* if (f.ShowDialog() == DialogResult.OK)
* {
* ImagedSave = f.FileName;
* int width = Convert.ToInt32(surface.Width);
* int height = Convert.ToInt32(surface.Height);
* Bitmap bmp = new Bitmap(width, height);
* surface.DrawToBitmap(bmp, new Rectangle(0, 0, width, height));
* bmp.Save(ImagedSave);
* Saved = false;
* Changed = false;
* }
* }
* private void FileUC1\_SaveClicked(object sender, EventArgs e)
* {
* if (!Saved)
* {
* FileUC1\_SaveAsClicked(sender, e);
* }
* else
* {
* if (ImagedSave != "")
* {
* SaveFileDialog f = new SaveFileDialog();
* int width = Convert.ToInt32(surface.Width);
* int height = Convert.ToInt32(surface.Height);
* Bitmap bmp = new Bitmap(width, height);
* surface.DrawToBitmap(bmp, new Rectangle(0, 0, width, height));
* bmp.Save(ImagedSave);
* }
* }
* }

1. Undo

* Ảnh minh họa: 
* Mô tả: khi click vào biểu tượng sẽ vẽ lại hình trước đó một bước.
* Cách xử lý: lưu các hình ảnh vào một list, dựa vào đó để vẽ lại hình trước hình hiện tại.
* Code xử lý:
* public void UndoPress()
* {
* if (Undo.Count > 0 && UndoLocation.Count > 0)
* {
* RedoLocation.Push(UndoLocation.Peek());
* Redo.Push(Undo.Peek());
* Location = UndoLocation.Pop();
* Image = Undo.Peek();
* Size = Undo.Pop().Size;
* }
* }

1. Redo

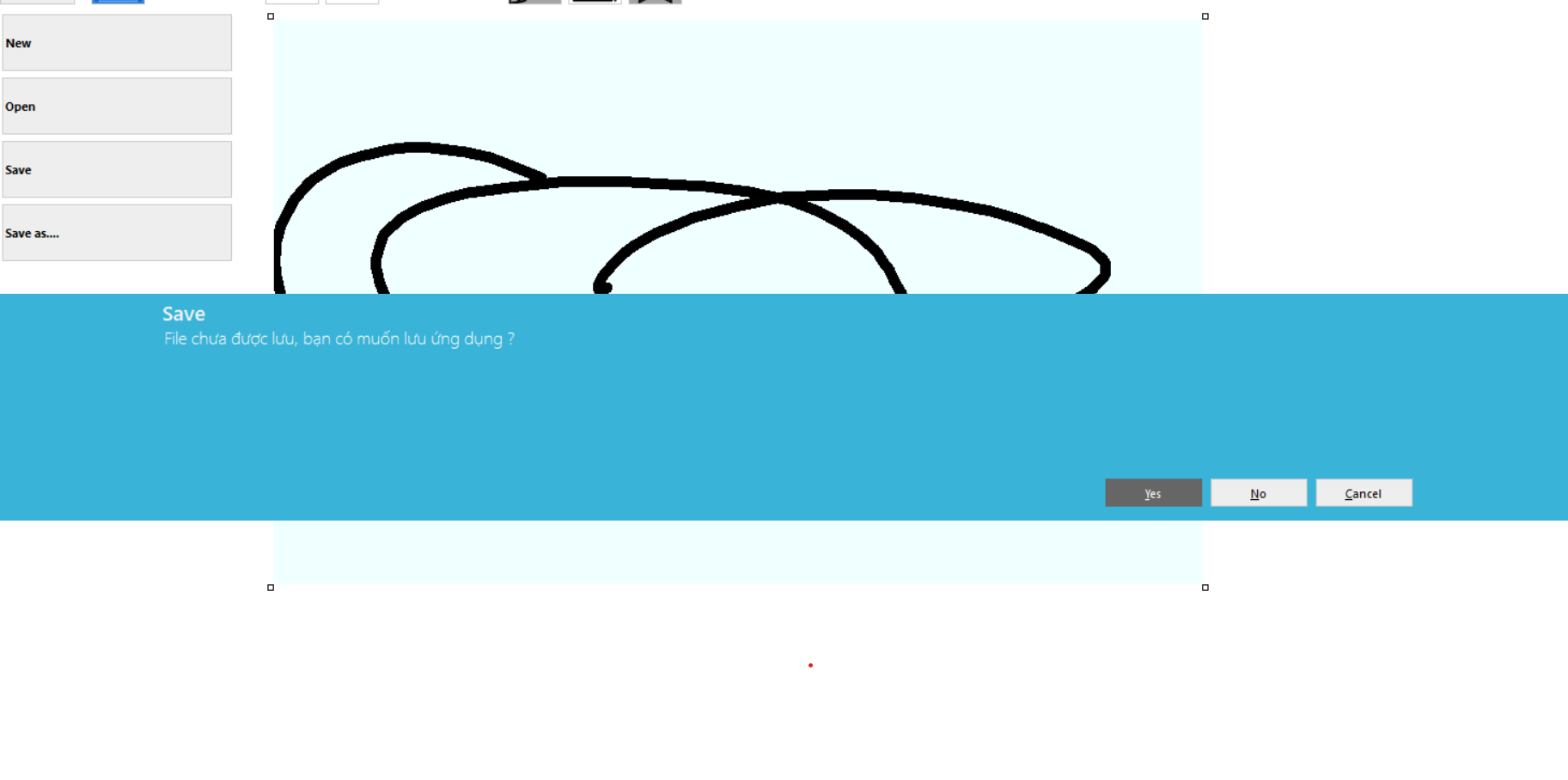
* Ảnh minh họa: 
* Mô tả: khi click vào biểu tượng sẽ vẽ lại hình sau đó một bước.
* Cách xử lý: lưu các hình ảnh vào một list, dựa vào đó để vẽ lại hình sau hình hiện tại.
* Code xử lý:
* public void RedoPress()
* {
* if (Redo.Count > 1 && RedoLocation.Count > 1)
* {
* UndoLocation.Push(RedoLocation.Pop());
* Undo.Push(Redo.Pop());
* Location = RedoLocation.Peek();
* Image = Redo.Peek();
* Size = Image.Size;
* }

I.4. Close

* Ảnh minh họa: => 
* Mô tả: khi click vào biểu tượng sẽ yêu cầu người dùng xác nhận có muốn lưu lại hình ảnh không. Sau đó xử lý và đóng ứng dụng.
* Cách xử lý: Cách xử lý: khi click vào biểu tượng sẽ hiển thị MessageBox yêu cầu người dùng xác nhận có muốn lưu lại hình ảnh không:
* Nếu người dùng chọn Cancel trở lại giao diện chính.
* Nếu người dùng chọn Yes sẽ lưu lại hình ảnh và đóng form.
* Nếu người dùng chọn No sẽ đóng Form.
* Code xử lý:
* private void Test\_FormClosing(object sender, FormClosingEventArgs e)
* {
* if (!Saved || Changed)
* {
* DialogResult dr = MetroFramework.MetroMessageBox.Show(this, "File chưa được lưu, bạn có muốn đóng ứng dụng ?", "Save", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information);
* if (dr == DialogResult.Yes)
* {
* FileUC1\_SaveClicked(sender, e);
* }
* else if (dr == DialogResult.Cancel)
* {
* e.Cancel = true;
* }
* }
* }

1. **File Menu**
2. New

* Ảnh minh họa: 
* Mô tả: khi click vào biểu tượng nếu người dung chưa lưu lại sẽ yêu cầu xác nhận có muốn lưu lại hình ảnh không. Nếu có, chuyển sang phần lưu và thực hiện lưu lại và thay mới khung vẽ. Nếu không, bỏ qua và thay mới khung vẽ. Có thể bấm cancel để hủy.



Hình . Giao diện khi tạo hình ảnh mới.

* Cách xử lý: khi click vào biểu tượng sẽ hiển thị MessageBox yêu cầu người dùng xác nhận có muốn lưu lại hình ảnh không:
* Nếu người dùng Cancel sẽ bỏ qua không thực hiện tiếp.
* Nếu người dùng chọn Yes sẽ lưu lại hình ảnh và cho phép người dùng nhập chiều dài và chiều rộng của hình ảnh mới.
* Nếu người dùng chọn No sẽ cho phép người dùng nhập chiều dài và chiều rộng của hình ảnh mới và không lưu lại hình ảnh cũ.
* Code xử lý:

private void FileUC1\_NewClicked(object sender, EventArgs e)

{

if (!Saved||Changed)

{

DialogResult dr = MetroFramework.MetroMessageBox.Show(this, "File chưa được lưu, bạn có muốn đóng ứng dụng ?", "Save ?", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information);

if (dr == DialogResult.Yes)

{

Saved = false;

FileUC1\_SaveClicked(sender, e);

draw = new Bitmap(surface.Width, surface.Height);

g = Graphics.FromImage(draw);

g.CompositingMode = System.Drawing.Drawing2D.CompositingMode.SourceCopy;

g.Clear(Color.Transparent);

g.CompositingMode = System.Drawing.Drawing2D.CompositingMode.SourceOver;

surface.Image = draw;

}

else if (dr == DialogResult.No)

{

draw = new Bitmap(surface.Width, surface.Height);

g = Graphics.FromImage(draw);

g.CompositingMode = System.Drawing.Drawing2D.CompositingMode.SourceCopy;

g.Clear(Color.Transparent);

g.CompositingMode = System.Drawing.Drawing2D.CompositingMode.SourceOver;

surface.Image = draw;

}

else if (dr==DialogResult.Cancel)

{

}

}

else

{

Saved = false;

draw = new Bitmap(surface.Width, surface.Height);

g = Graphics.FromImage(draw);

g.CompositingMode = System.Drawing.Drawing2D.CompositingMode.SourceCopy;

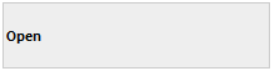
g.Clear(Color.Transparent);

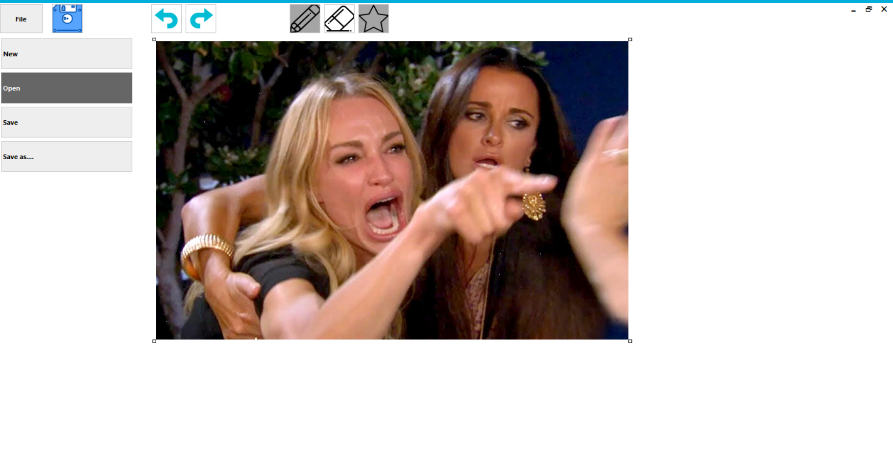
g.CompositingMode = System.Drawing.Drawing2D.CompositingMode.SourceOver;

surface.Image = draw;

}

1. Open

* Ảnh minh họa: 
* Mô tả: khi click vào biểu tượng sẽ mở dialog cho phép người dùng chọn hình ảnh để mở ra ở Paint.



Hình . Giao diện khi mở ảnh.

* Cách xử lý: mở OpenFileDialog, dựa vào dữ liệu dialog trả về để xử lý
* Code xử lý:

private void FileUC1\_OpenClicked(object sender, EventArgs e)

{

OpenFileDialog f = new OpenFileDialog();

f.Filter = @"Image Files(\*.jpg; \*.jpeg; \*.bmp;\*png)|\*.jpg; \*.jpeg; \*.bmp; \*.png";

if (f.ShowDialog() == DialogResult.OK)

{

Bitmap bit = new Bitmap(f.FileName);

surface.Size = bit.Size;

surface.Image = bit;

Saved = false;

Changed = false;

surface.PushUndo(surface.Image);

}

}

1. Save và Save as…

* Ảnh mình họa: 
* Mô tả, cách xử lý và code xử lý giống như Save trên Thanh tiêu đề

**Thanh công cụ**

1. Tools

III.1.a Pencil

* Ảnh minh họa: 
* Mô tả: chọn bút để vẽ.
* Cách xử lý: bắt sự kiện mousemove và vẽ.
* Code xử lý:

case BrushType.Pencil:

pen = new Pen(color, Test.PenSize);

pen.StartCap = pen.EndCap = System.Drawing.Drawing2D.LineCap.Round;

CurrentStatus = DrawStatus.ToolDraw;

break;

case BrushType.Pencil:

grp = Graphics.FromImage(Image);

grp.DrawLine(pen, mouseDown, location);

III.1.b Eraser

* Ảnh minh họa:
* Mô tả: xóa những phần không cần thiết của hình ảnh.
* Cách xử lý: chuyển cách nhận hình trên picturebox thành source copy và vẽ đè lên bằng bút màu transparent.
* Code xử lý:

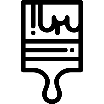
case BrushType.Eraser:

grp = Graphics.FromImage(Image);

grp.CompositingMode = System.Drawing.Drawing2D.CompositingMode.SourceCopy;

grp.DrawLine(pen, mouseDown, location);

break;

III.1.c Brush

* Ảnh minh họa:
* Mô tả: vẽ với hiệu ứng như cọ vẽ.
* Cách xử lý: thêm 1 graphics để nhận hình nền và 1 graphics để nhận cọ vẽ hiện tại set graphics của hình nền thành copy và cọ vẽ thành over để tạo hiệu ứng đổ mờ.
* Code xử lý:

case BrushType.Brush:

path = new GraphicsPath();

Bitmap temp = (Bitmap)Image;

gra = Graphics.FromImage(temp);

gra.CompositingMode = CompositingMode.SourceCopy;

grp = (Graphics)gra;

grp.CompositingMode = System.Drawing.Drawing2D.CompositingMode.SourceOver;

path.AddLines(points.ToArray());

gra.DrawPath(pen, path);

//pen.LineJoin = LineJoin.Round;

break;

III.3.d :color picker

Mô tả: ấn chuột phải để chọn màu trên bản vẽ.

Cách xử lý ấn chuột xuống sẽ lấy màu trên 1 điểm của pictruebox;

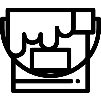
Code xử lý:

case BrushType.Picker:

temp = (Bitmap) Image;

Pickercolor = temp.GetPixel(location.X, location.Y);

break;

1. Fill tool:

Ảnh minh họa

Mô tả: tô màu các màu cùng nhau.

Cách xử lý: dùng thuật toán đệ quy để tô các màu cùng lân cận:

Code xử lý:

private void FloodFill(Point node, Color replaceColor)

{

Bitmap DrawBitmap = new Bitmap(Image);

Color targetColor = DrawBitmap.GetPixel(node.X, node.Y);

if (targetColor.ToArgb() == replaceColor.ToArgb())

return;

Stack<Point> pixels = new Stack<Point>();

pixels.Push(node);

while (pixels.Count != 0)

{

Point floodNode = pixels.Pop();

Color floodColor = DrawBitmap.GetPixel(floodNode.X, floodNode.Y);

if (floodColor == targetColor)

{

DrawBitmap.SetPixel(floodNode.X, floodNode.Y, replaceColor);

if (floodNode.X != 0)

pixels.Push(new Point(floodNode.X - 1, floodNode.Y));

if (floodNode.X + 1 < Width)

pixels.Push(new Point(floodNode.X + 1, floodNode.Y));

if (floodNode.Y != 0)

pixels.Push(new Point(floodNode.X, floodNode.Y - 1));

if (floodNode.Y + 1 < Height)

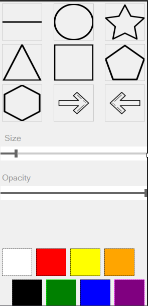
pixels.Push(new Point(floodNode.X, floodNode.Y + 1));

}

}

Image = DrawBitmap;

}

1. Shape

* Ảnh minh họa:
* Mô tả: vẽ các hình tương ứng.
* Cách xử lý dùng các hàm line draw, drawelipse, drawrectangle, drawpolygon,…

Hình . Các shape trong ứng dụng.

Code xử lý:

private static Point MouseDown, MouseCurrent;

public static Rectangle CreateRectangle(Point ptMouseDown, Point ptMouseCurrent)

{

MouseDown = ptMouseDown;

MouseCurrent = ptMouseCurrent;

return new Rectangle(

Math.Min(ptMouseDown.X, ptMouseCurrent.X),

Math.Min(ptMouseDown.Y, ptMouseCurrent.Y),

Math.Abs(ptMouseDown.X - ptMouseCurrent.X),

Math.Abs(ptMouseDown.Y - ptMouseCurrent.Y)

); ;

}

private static Point[] RightArrowPoints(Rectangle rect)

{

Point p0 = new Point(rect.Right, rect.Top + (rect.Height / 2));

Point p1 = new Point(rect.Left + (rect.Width / 2), rect.Top);

Point p2 = new Point(rect.Left + (rect.Width / 2), rect.Top + (rect.Height / 4));

Point p3 = new Point(rect.Left, rect.Top + (rect.Height / 4));

Point p4 = new Point(rect.Left, rect.Bottom - (rect.Height / 4));

Point p5 = new Point(rect.Left + (rect.Width / 2), rect.Bottom - (rect.Height / 4));

Point p6 = new Point(rect.Left + (rect.Width / 2), rect.Bottom);

Point[] pts = { p0, p1, p2, p3, p4, p5, p6 };

return pts;

}

private static Point[] LeftArrowPoints(Rectangle rect)

{

Point p0 = new Point(rect.Left, rect.Top + (rect.Height / 2));

Point p1 = new Point(rect.Left + (rect.Width / 2), rect.Top);

Point p2 = new Point(rect.Left + (rect.Width / 2), rect.Top + (rect.Height / 4));

Point p3 = new Point(rect.Right, rect.Top + (rect.Height / 4));

Point p4 = new Point(rect.Right, rect.Bottom - (rect.Height / 4));

Point p5 = new Point(rect.Left + (rect.Width / 2), rect.Bottom - (rect.Height / 4));

Point p6 = new Point(rect.Left + (rect.Width / 2), rect.Bottom);

Point[] pts = { p0, p1, p2, p3, p4, p5, p6 };

return pts;

}

private static Point[] DownArrowPoints(Rectangle rect)

{

Point p0 = new Point(rect.Left + (rect.Width / 2), rect.Bottom);

Point p1 = new Point(rect.Left, rect.Top + (rect.Height / 2));

Point p2 = new Point(rect.Left + (rect.Width / 4), rect.Top + (rect.Height / 2));

Point p3 = new Point(rect.Left + (rect.Width / 4), rect.Top);

Point p4 = new Point(rect.Right - (rect.Width / 4), rect.Top);

Point p5 = new Point(rect.Right - (rect.Width / 4), rect.Bottom - (rect.Height / 2));

Point p6 = new Point(rect.Right, rect.Bottom - (rect.Height / 2));

Point[] pts = { p0, p1, p2, p3, p4, p5, p6 };

return pts;

}

private static Point[] UpArrowPoints(Rectangle rect)

{

Point p0 = new Point(rect.Left + (rect.Width / 2), rect.Top);

Point p1 = new Point(rect.Left, rect.Top + (rect.Height / 2));

Point p2 = new Point(rect.Left + (rect.Width / 4), rect.Top + (rect.Height / 2));

Point p3 = new Point(rect.Left + (rect.Width / 4), rect.Bottom);

Point p4 = new Point(rect.Right - (rect.Width / 4), rect.Bottom);

Point p5 = new Point(rect.Right - (rect.Width / 4), rect.Bottom - (rect.Height / 2));

Point p6 = new Point(rect.Right, rect.Bottom - (rect.Height / 2));

Point[] pts = { p0, p1, p2, p3, p4, p5, p6 };

return pts;

}

private static Point[] TrianglePoint(Rectangle rect)

{

Point p0 = new Point(rect.Left + (rect.Width / 2), rect.Top);

Point p1 = new Point(rect.Left, rect.Bottom);

Point p2 = new Point(rect.Right, rect.Bottom);

Point[] pts = { p0, p1, p2 };

return pts;

}

private static Point[] HexagonPoint(Rectangle rect)

{

Point p0 = new Point(rect.Left + rect.Width / 2, rect.Top);

Point p1 = new Point(rect.Right, rect.Top + rect.Height / 3);

Point p2 = new Point(rect.Right, rect.Top + 2\*rect.Height / 3);

Point p3 = new Point(rect.Left + rect.Width / 2, rect.Bottom);

Point p4 = new Point(rect.Left, rect.Top + 2\*rect.Height / 3);

Point p5 = new Point(rect.Left, rect.Top + rect.Height / 3);

Point[] pts = { p0, p1, p2, p3, p4, p5 };

return pts;

}

private static Point[] PentagonPoint(Rectangle rect)

{

Point p0 = new Point(rect.Left + rect.Width / 2, rect.Top);

Point p1 = new Point(rect.Left, rect.Top + (11 \* rect.Height / 30));

Point p2 = new Point(rect.Left + rect.Width / 5, rect.Bottom);

Point p3 = new Point(rect.Right - rect.Width / 5, rect.Bottom);

Point p4 = new Point(rect.Right, rect.Top + (11 \* rect.Height / 30));

Point[] pts = { p0, p1, p2, p3, p4 };

return pts;

}

private static Point[] StarPoints(Rectangle rect)

{

Point p0 = new Point(rect.Left + rect.Width / 2, rect.Top);

Point p1 = new Point(rect.Left + 3 \* rect.Width / 8, rect.Top + rect.Height / 3);

Point p2 = new Point(rect.Left, rect.Top + (11 \* rect.Height / 30));

Point p3 = new Point(rect.Left + rect.Width / 4, rect.Bottom - (2 \* rect.Height / 5));

Point p4 = new Point(rect.Left + rect.Width / 5, rect.Bottom);

Point p5 = new Point(rect.Left + rect.Width / 2, rect.Bottom - (rect.Height / 5));

Point p6 = new Point(rect.Right - rect.Width / 5, rect.Bottom);

Point p7 = new Point(rect.Right - rect.Width / 4, rect.Bottom - (2 \* rect.Height / 5));

Point p8 = new Point(rect.Right, rect.Top + (11 \* rect.Height / 30));

Point p9 = new Point(rect.Right - 3 \* rect.Width / 8, rect.Top + rect.Height / 3);

Point[] pts = { p0, p1, p2, p3, p4, p5, p6, p7, p8, p9 };

return pts;

}

public static void Draw(Graphics g, Pen pen, Rectangle rect, Shape type)

{

Point[] pts = null;

switch (type)

{

case Shape.RecTangle:

g.DrawRectangle(pen, rect);

return;

case Shape.Circle:

g.DrawEllipse(pen, rect);

return;

case Shape.Line:

pen.StartCap = pen.EndCap = System.Drawing.Drawing2D.LineCap.Round;

g.DrawLine(pen, MouseDown, MouseCurrent);

return;

case Shape.RightArrow:

pts = RightArrowPoints(rect);

break;

case Shape.LeftArrow:

pts = LeftArrowPoints(rect);

break;

case Shape.DownArrow:

pts = DownArrowPoints(rect);

break;

case Shape.UpArrow:

pts = UpArrowPoints(rect);

break;

case Shape.Triangle:

pts = TrianglePoint(rect);

break;

case Shape.Hexagon:

pts = HexagonPoint(rect);

break;

case Shape.Pentagon:

pts = PentagonPoint(rect);

break;

case Shape.Star:

pts = StarPoints(rect);

break;

}

if (pts != null)

g.DrawPolygon(pen, pts);

}

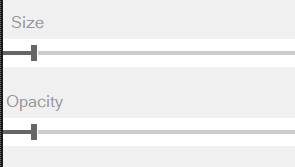
}

public enum Shape

{

RecTangle, RightArrow, LeftArrow, UpArrow, DownArrow, Circle, Star, Triangle, Pentagon, Hexagon, Line

}

1. Size

\_ Ảnh minh họa:

\_ Mô tả: chọn size cho nét vẽ

* Cách xử lý: được xử lý bằng các trackbar trong khi chọn bút để vẽ.
* Code xử lý:

private void ESize\_Scroll(object sender, ScrollEventArgs e)

{

EraserSize = ESize.Value;

}

1. Colors

* Ảnh minh họa:



* Mô tả: chọn màu cho từng bút vẽ riêng biệt.
* Cách xử lý: được cho sẵn 8 màu để chọn sẵn và phối màu bằng cách vẽ chồng có opacity lên nhau để phối màu bằng picker.
* Code xử lý:

Code mẫu cho phần xử lý lấy màu xanh cho các loại bút

private void PenUC1\_BlueClicked(object sender, EventArgs e)

{

PenB.BackColor = Color.FromArgb(50, PenUC.Instance.getBlue());

color = PenUC.Instance.getBlue();

PenColor = color;

}