كلية الهندسة المعلوماتية
جامعة دمشق

السنة :الرابعة

المادة : نظم وسائط متعددة
X-Ray Images

تقديم الطلاب

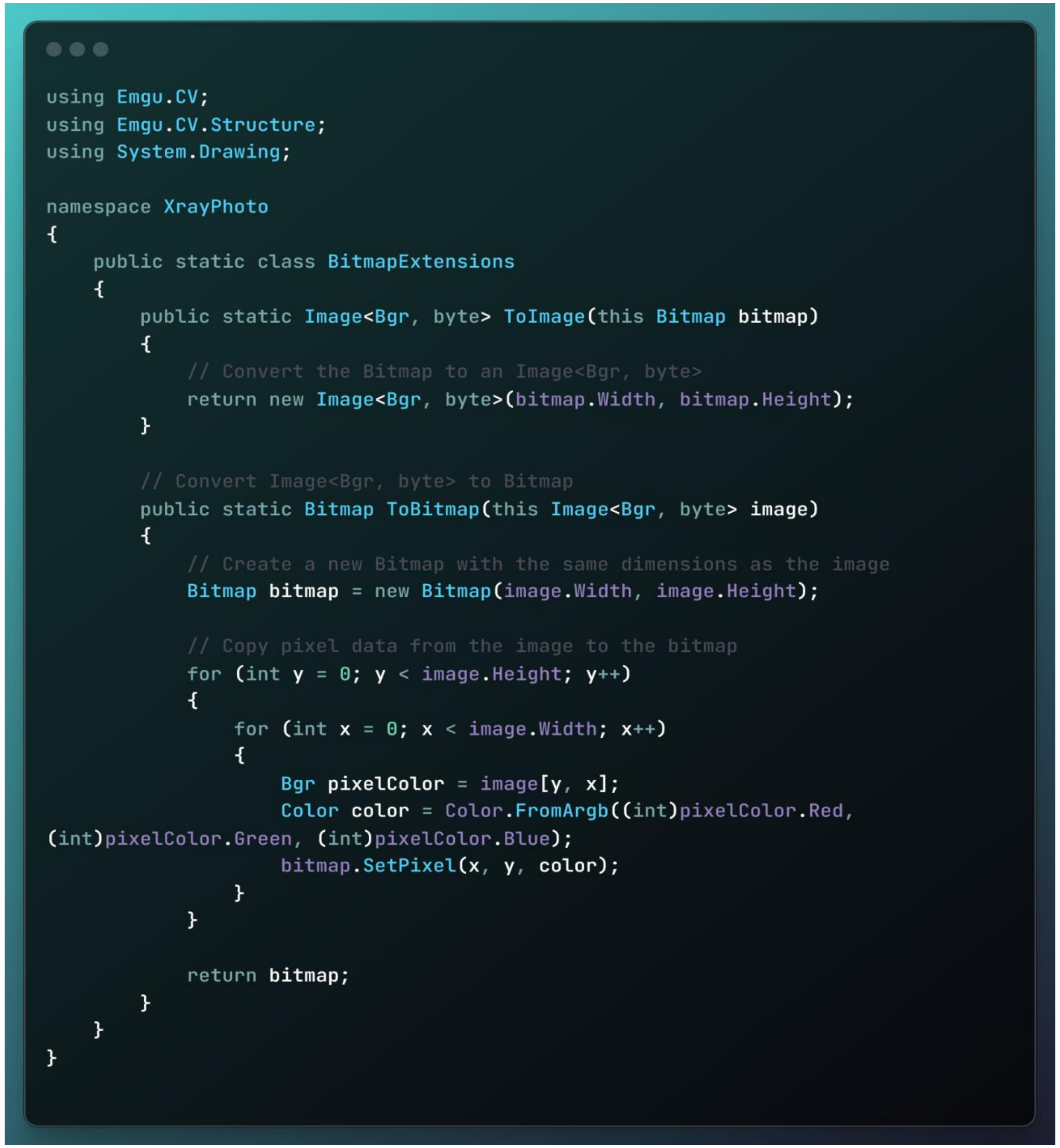
علي جهاد العلي

توفيق محمد الحمادة

علا عمر الشيخ

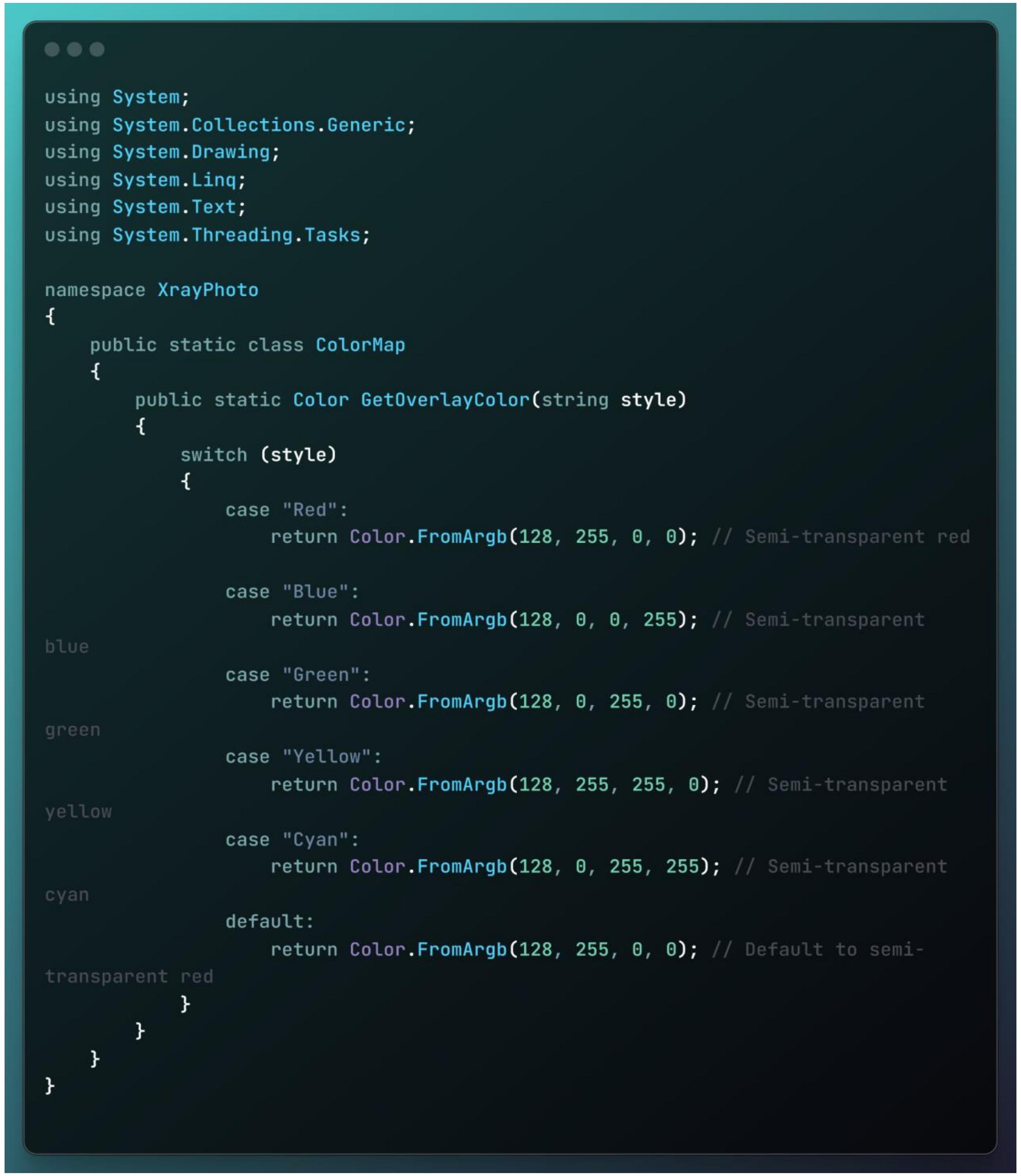
عمار نزير العلوش

كنان غسان علي



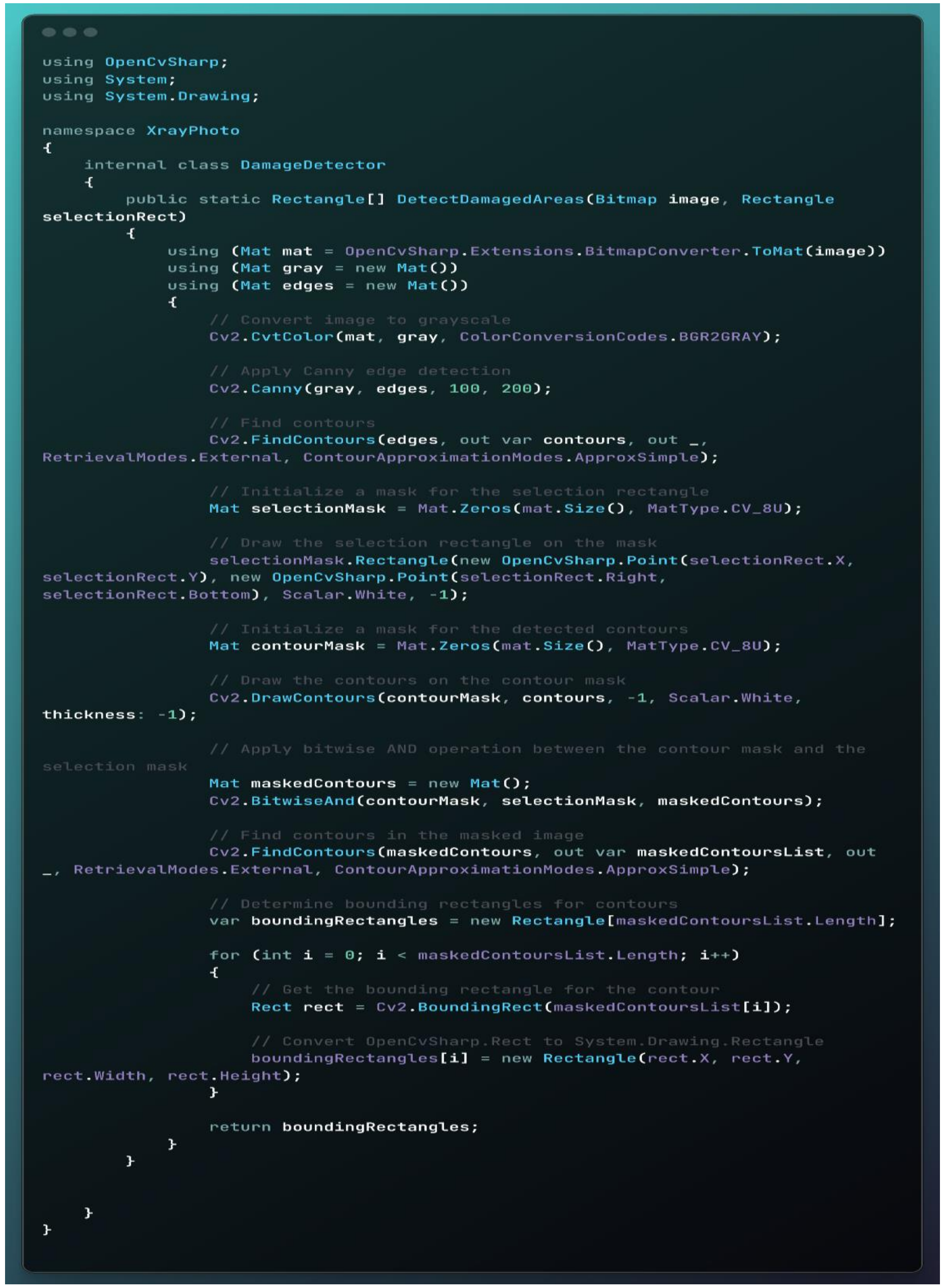
```csharp
using Emgu.CV;
using Emgu.CV.Structure;
using System.Drawing;

namespace XrayPhoto
{
    public static class BitmapExtensions
    {
        public static Image<Bgr, byte> ToImage(this Bitmap bitmap)
        {
            // Convert the Bitmap to an Image<Bgr, byte>
            return new Image<Bgr, byte>(bitmap.Width, bitmap.Height);
        }

        // Convert Image<Bgr, byte> to Bitmap
        public static Bitmap ToBitmap(this Image<Bgr, byte> image)
        {
            // Create a new Bitmap with the same dimensions as the image
            Bitmap bitmap = new Bitmap(image.Width, image.Height);

            // Copy pixel data from the image to the bitmap
            for (int y = 0; y < image.Height; y++)
            {
                for (int x = 0; x < image.Width; x++)
                {
                    Bgr pixelColor = image[y, x];
                    Color color = Color.FromArgb((int)pixelColor.Red,
(int)pixelColor.Green, (int)pixelColor.Blue);
                    bitmap.SetPixel(x, y, color);
                }
            }

            return bitmap;
        }
    }
}
```

هذا الجزء من الكود يحتوي على
(extensions)
للفئة Bitmap
لتحويل بيانات الصور بين نوعي البيانات
byte وBitmap



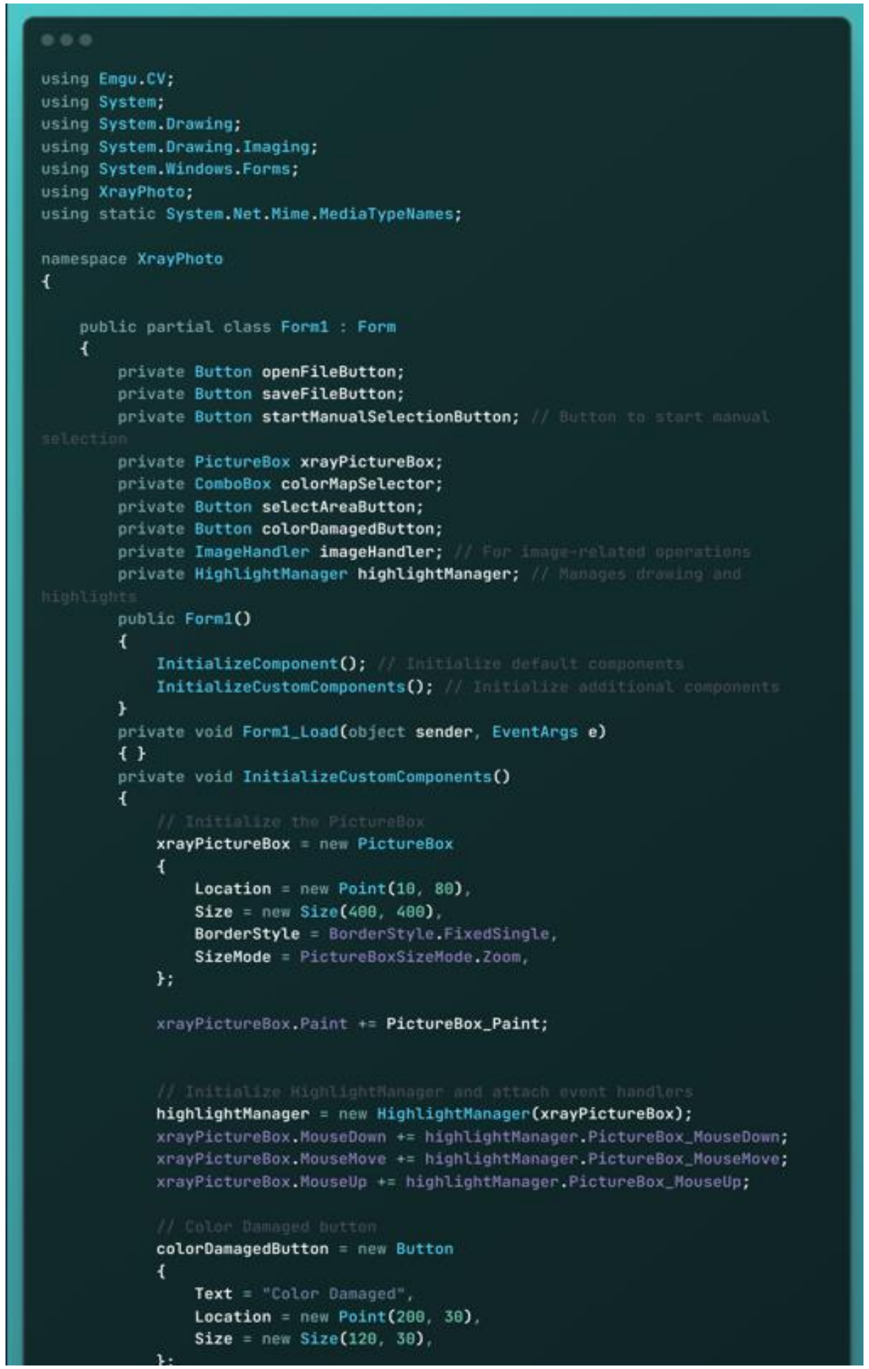
```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace XrayPhoto
{
    public static class ColorMap
    {
        public static Color GetOverlayColor(string style)
        {
            switch (style)
            {
                case "Red":
                    return Color.FromArgb(128, 255, 0, 0); // Semi-transparent red

                case "Blue":
                    return Color.FromArgb(128, 0, 0, 255); // Semi-transparent
blue
                case "Green":
                    return Color.FromArgb(128, 0, 255, 0); // Semi-transparent
green
                case "Yellow":
                    return Color.FromArgb(128, 255, 255, 0); // Semi-transparent
yellow
                case "Cyan":
                    return Color.FromArgb(128, 0, 255, 255); // Semi-transparent
cyan
                default:
                    return Color.FromArgb(128, 255, 0, 0); // Default to semi-
transparent red
            }
        }
    }
}
```

- هذا الجزء من الكود تحتوي عل فئة تسمى
- GetOverLayColor
- التي تقوم بأرجاع لون
- يتم استخدام التعليمة switch لفحص قيمة المعامل styleوتحديد اللون المناسب بناءً على النمط المحدد.

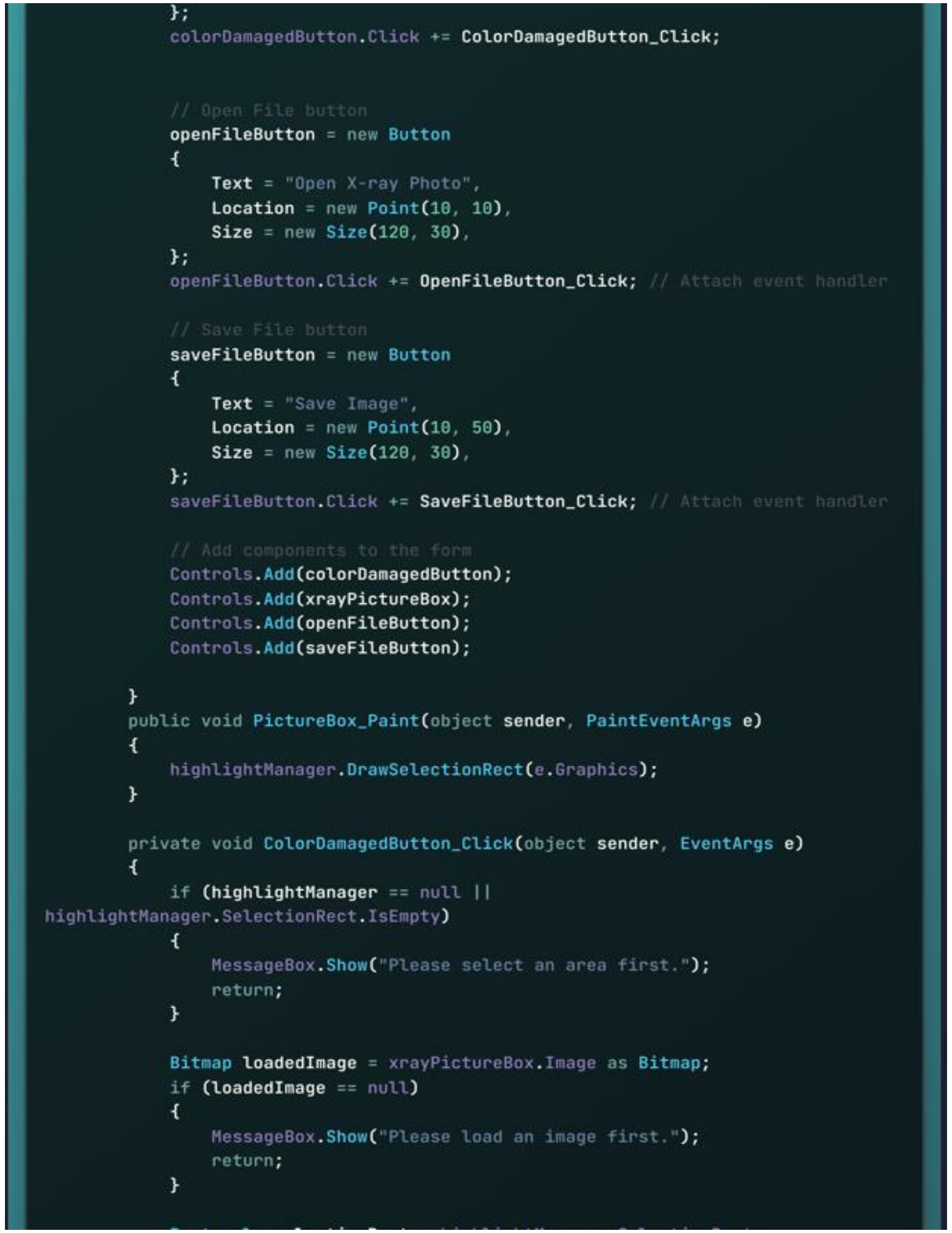


```
using OpenCvSharp;
using System;
using System.Drawing;

namespace XrayPhoto
{
    internal class DamageDetector
    {
        public static Rectangle[] DetectDamagedAreas(Bitmap image, Rectangle
selectionRect)
        {
            using (Mat mat = OpenCvSharp.Extensions.BitmapConverter.ToMat(image))
            using (Mat gray = new Mat())
            using (Mat edges = new Mat())
            {
                // Convert image to grayscale
                Cv2.CvtColor(mat, gray, ColorConversionCodes.BGR2GRAY);

                // Apply Canny edge detection
                Cv2.Canny(gray, edges, 100, 200);

                // Find contours
                Cv2.FindContours(edges, out var contours, out _,
RetrievalModes.External, ContourApproximationModes.ApproxSimple);

                // Initialize a mask for the selection rectangle
                Mat selectionMask = Mat.Zeros(mat.Size(), MatType.CV_8U);

                // Draw the selection rectangle on the mask
                selectionMask.Rectangle(new OpenCvSharp.Point(selectionRect.X,
selectionRect.Y), new OpenCvSharp.Point(selectionRect.Right,
selectionRect.Bottom), Scalar.White, -1);

                // Initialize a mask for the detected contours
                Mat contourMask = Mat.Zeros(mat.Size(), MatType.CV_8U);

                // Draw the contours on the contour mask
                Cv2.DrawContours(contourMask, contours, -1, Scalar.White,
thickness: -1);

                // Apply bitwise AND operation between the contour mask and the
selection mask
                Mat maskedContours = new Mat();
                Cv2.BitwiseAnd(contourMask, selectionMask, maskedContours);

                // Find contours in the masked image
                Cv2.FindContours(maskedContours, out var maskedContoursList, out
_, RetrievalModes.External, ContourApproximationModes.ApproxSimple);

                // Determine bounding rectangles for contours
                var boundingRectangles = new Rectangle[maskedContoursList.Length];

                for (int i = 0; i < maskedContoursList.Length; i++)
                {
                    // Get the bounding rectangle for the contour
                    Rect rect = Cv2.BoundingRect(maskedContoursList[i]);

                    // Convert OpenCvSharp.Rect to System.Drawing.Rectangle
                    boundingRectangles[i] = new Rectangle(rect.X, rect.Y,
rect.Width, rect.Height);
                }

                return boundingRectangles;
            }
        }


    }
}
```

- هذا الجزء من الكود
- الفئة على طريقة واحدة تسمى DetectDamagedAreas التي تقوم بكشف المناطق المتضررة في صورة معينة.
- هذه الفئة تستخدم مكتبة OpenCvSharp لتحليل ومعالجة الصور وتحديد المناطق المتضررة في صورة معينة.

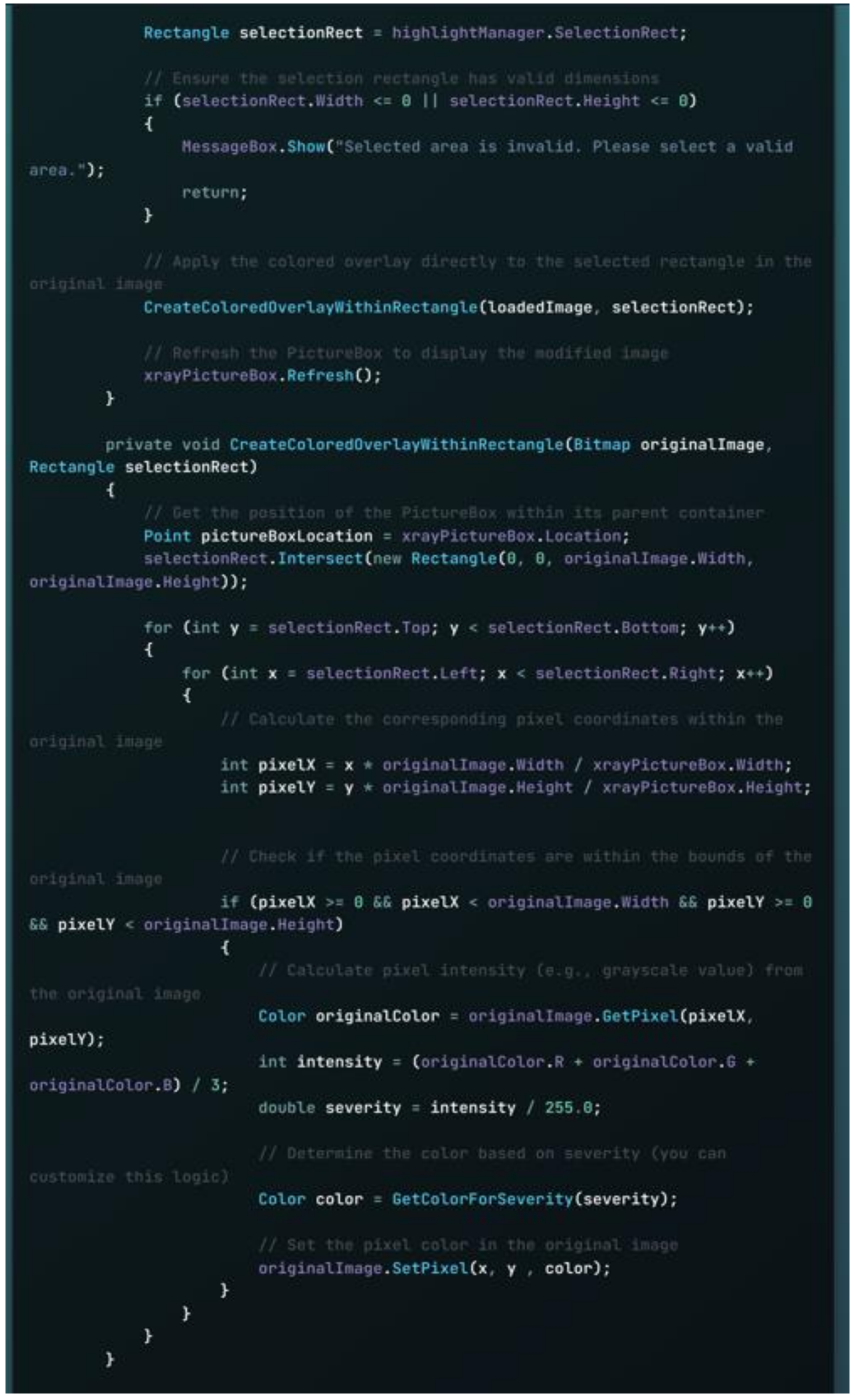


```
using Emgu.CV;
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Windows.Forms;
using XrayPhoto;
using static System.Net.Mime.MediaTypeNames;

namespace XrayPhoto
{

    public partial class Form1 : Form
    {
        private Button openFileButton;
        private Button saveFileButton;
        private Button startManualSelectionButton; // Button to start manual
selection
        private PictureBox xrayPictureBox;
        private ComboBox colorMapSelector;
        private Button selectAreaButton;
        private Button colorDamagedButton;
        private ImageHandler imageHandler; // For image-related operations
        private HighlightManager highlightManager; // Manages drawing and
highlights
        public Form1()
        {
            InitializeComponent(); // Initialize default components
            InitializeCustomComponents(); // Initialize additional components
        }
        private void Form1_Load(object sender, EventArgs e)
        { }
        private void InitializeCustomComponents()
        {
            // Initialize the PictureBox
            xrayPictureBox = new PictureBox
            {
                Location = new Point(10, 80),
                Size = new Size(400, 400),
                BorderStyle = BorderStyle.FixedSingle,
                SizeMode = PictureBoxSizeMode.Zoom,
            };

            xrayPictureBox.Paint += PictureBox_Paint;


            // Initialize HighlightManager and attach event handlers
            highlightManager = new HighlightManager(xrayPictureBox);
            xrayPictureBox.MouseDown += highlightManager.PictureBox_MouseDown;
            xrayPictureBox.MouseMove += highlightManager.PictureBox_MouseMove;
            xrayPictureBox.MouseUp += highlightManager.PictureBox_MouseUp;

            // Color Damaged button
            colorDamagedButton = new Button
            {
                Text = "Color Damaged",
                Location = new Point(200, 30),
                Size = new Size(120, 30),
            };
```

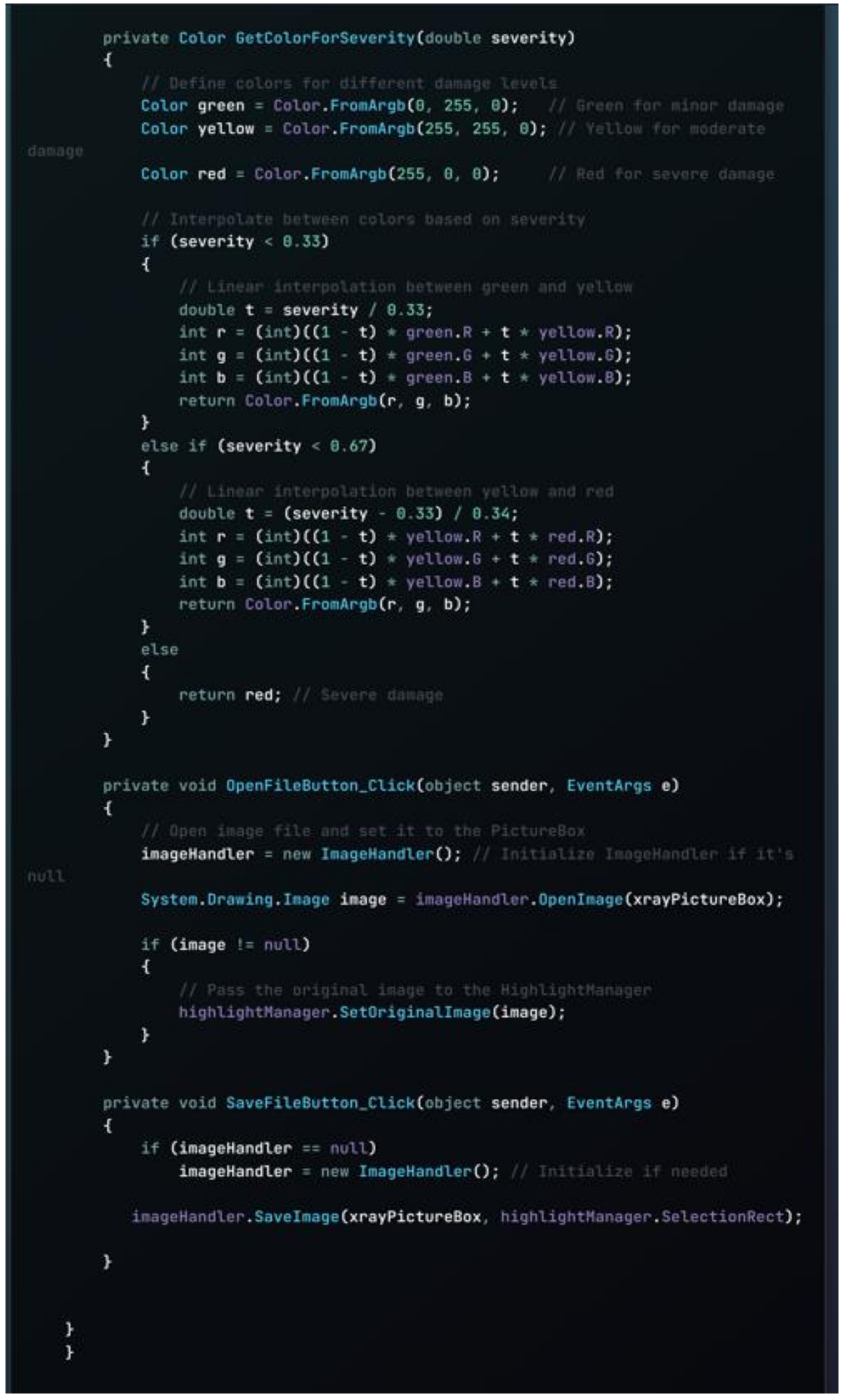


```
            };
            colorDamagedButton.Click += ColorDamagedButton_Click;


            // Open File button
            openFileButton = new Button
            {
                Text = "Open X-ray Photo",
                Location = new Point(10, 10),
                Size = new Size(120, 30),
            };
            openFileButton.Click += OpenFileButton_Click; // Attach event handler

            // Save File button
            saveFileButton = new Button
            {
                Text = "Save Image",
                Location = new Point(10, 50),
                Size = new Size(120, 30),
            };
            saveFileButton.Click += SaveFileButton_Click; // Attach event handler

            // Add components to the form
            Controls.Add(colorDamagedButton);
            Controls.Add(xrayPictureBox);
            Controls.Add(openFileButton);
            Controls.Add(saveFileButton);

        }
        public void PictureBox_Paint(object sender, PaintEventArgs e)
        {
            highlightManager.DrawSelectionRect(e.Graphics);
        }

        private void ColorDamagedButton_Click(object sender, EventArgs e)
        {
            if (highlightManager == null ||
highlightManager.SelectionRect.IsEmpty)
            {
                MessageBox.Show("Please select an area first.");
                return;
            }

            Bitmap loadedImage = xrayPictureBox.Image as Bitmap;
            if (loadedImage == null)
            {
                MessageBox.Show("Please load an image first.");
                return;
            }
```

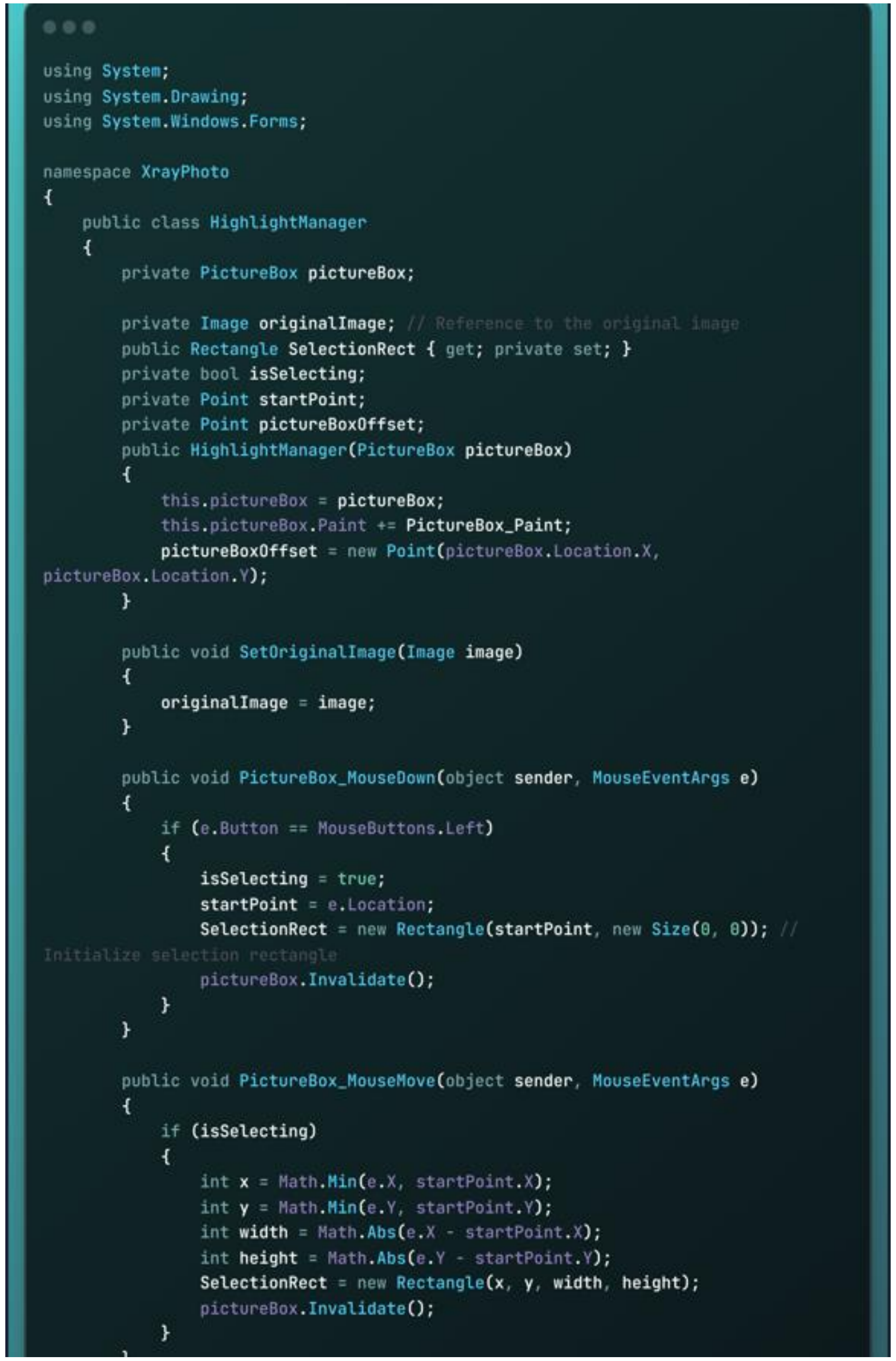
- هذا الجزء من الكود
- يتضمن العديد من العناصر المرئية مثل الأزرار وصندوق الصورة **PictureBox** وقائمة **ComboBox**
- تحديدًا، يتم في هذا الجزء تعريف وتهيئة المكونات المخصصة وإضافتها إلى النموذج ((Form))
- هذا الجزء من الكود يقوم بتهيئة وترتيب واجهة المستخدم الرسومية



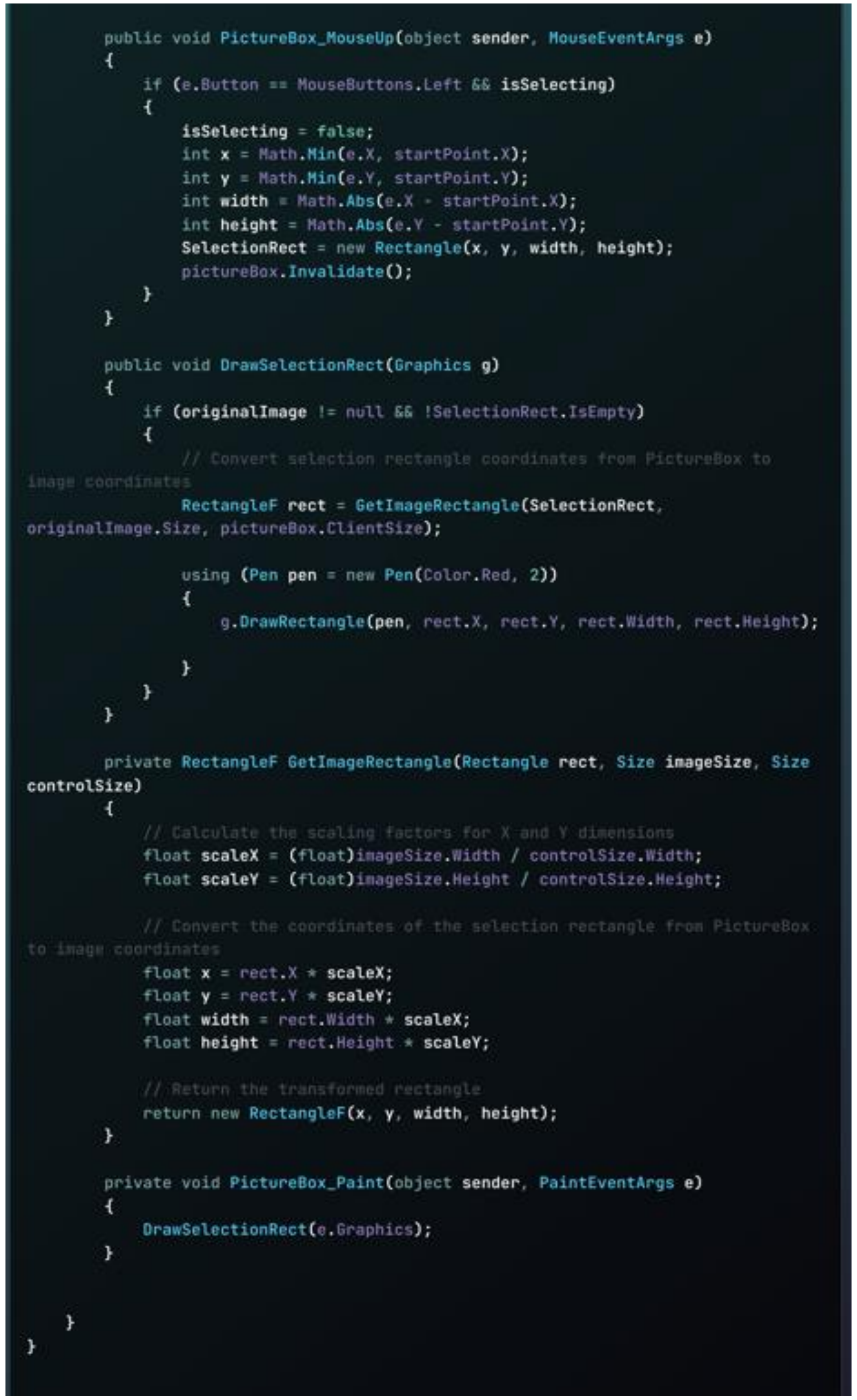
```
            Rectangle selectionRect = highlightManager.SelectionRect;

            // Ensure the selection rectangle has valid dimensions
            if (selectionRect.Width <= 0 || selectionRect.Height <= 0)
            {
                MessageBox.Show("Selected area is invalid. Please select a valid
area.");
                return;
            }

            // Apply the colored overlay directly to the selected rectangle in the
original image
            CreateColoredOverlayWithinRectangle(loadedImage, selectionRect);

            // Refresh the PictureBox to display the modified image
            xrayPictureBox.Refresh();
        }

        private void CreateColoredOverlayWithinRectangle(Bitmap originalImage,
Rectangle selectionRect)
        {
            // Get the position of the PictureBox within its parent container
            Point pictureBoxLocation = xrayPictureBox.Location;
            selectionRect.Intersect(new Rectangle(0, 0, originalImage.Width,
originalImage.Height));

            for (int y = selectionRect.Top; y < selectionRect.Bottom; y++)
            {
                for (int x = selectionRect.Left; x < selectionRect.Right; x++)
                {
                    // Calculate the corresponding pixel coordinates within the
original image
                    int pixelX = x * originalImage.Width / xrayPictureBox.Width;
                    int pixelY = y * originalImage.Height / xrayPictureBox.Height;


                    // Check if the pixel coordinates are within the bounds of the
original image
                    if (pixelX >= 0 && pixelX < originalImage.Width && pixelY >= 0
&& pixelY < originalImage.Height)
                    {
                        // Calculate pixel intensity (e.g., grayscale value) from
the original image
                        Color originalColor = originalImage.GetPixel(pixelX,
pixelY);
                        int intensity = (originalColor.R + originalColor.G +
originalColor.B) / 3;
                        double severity = intensity / 255.0;

                        // Determine the color based on severity (you can
customize this logic)
                        Color color = GetColorForSeverity(severity);

                        // Set the pixel color in the original image
                        originalImage.SetPixel(x, y , color);
                    }
                }
            }
        }
```



```
        private Color GetColorForSeverity(double severity)
        {
            // Define colors for different damage levels
            Color green = Color.FromArgb(0, 255, 0);   // Green for minor damage
            Color yellow = Color.FromArgb(255, 255, 0); // Yellow for moderate
damage
            Color red = Color.FromArgb(255, 0, 0);     // Red for severe damage

            // Interpolate between colors based on severity
            if (severity < 0.33)
            {
                // Linear interpolation between green and yellow
                double t = severity / 0.33;
                int r = (int)((1 - t) * green.R + t * yellow.R);
                int g = (int)((1 - t) * green.G + t * yellow.G);
                int b = (int)((1 - t) * green.B + t * yellow.B);
                return Color.FromArgb(r, g, b);
            }
            else if (severity < 0.67)
            {
                // Linear interpolation between yellow and red
                double t = (severity - 0.33) / 0.34;
                int r = (int)((1 - t) * yellow.R + t * red.R);
                int g = (int)((1 - t) * yellow.G + t * red.G);
                int b = (int)((1 - t) * yellow.B + t * red.B);
                return Color.FromArgb(r, g, b);
            }
            else
            {
                return red; // Severe damage
            }
        }

        private void OpenFileButton_Click(object sender, EventArgs e)
        {
            // Open image file and set it to the PictureBox
            imageHandler = new ImageHandler(); // Initialize ImageHandler if it's
null
            System.Drawing.Image image = imageHandler.OpenImage(xrayPictureBox);

            if (image != null)
            {
                // Pass the original image to the HighlightManager
                highlightManager.SetOriginalImage(image);
            }
        }

        private void SaveFileButton_Click(object sender, EventArgs e)
        {
            if (imageHandler == null)
                imageHandler = new ImageHandler(); // Initialize if needed

           imageHandler.SaveImage(xrayPictureBox, highlightManager.SelectionRect);

        }


    }
    }
```

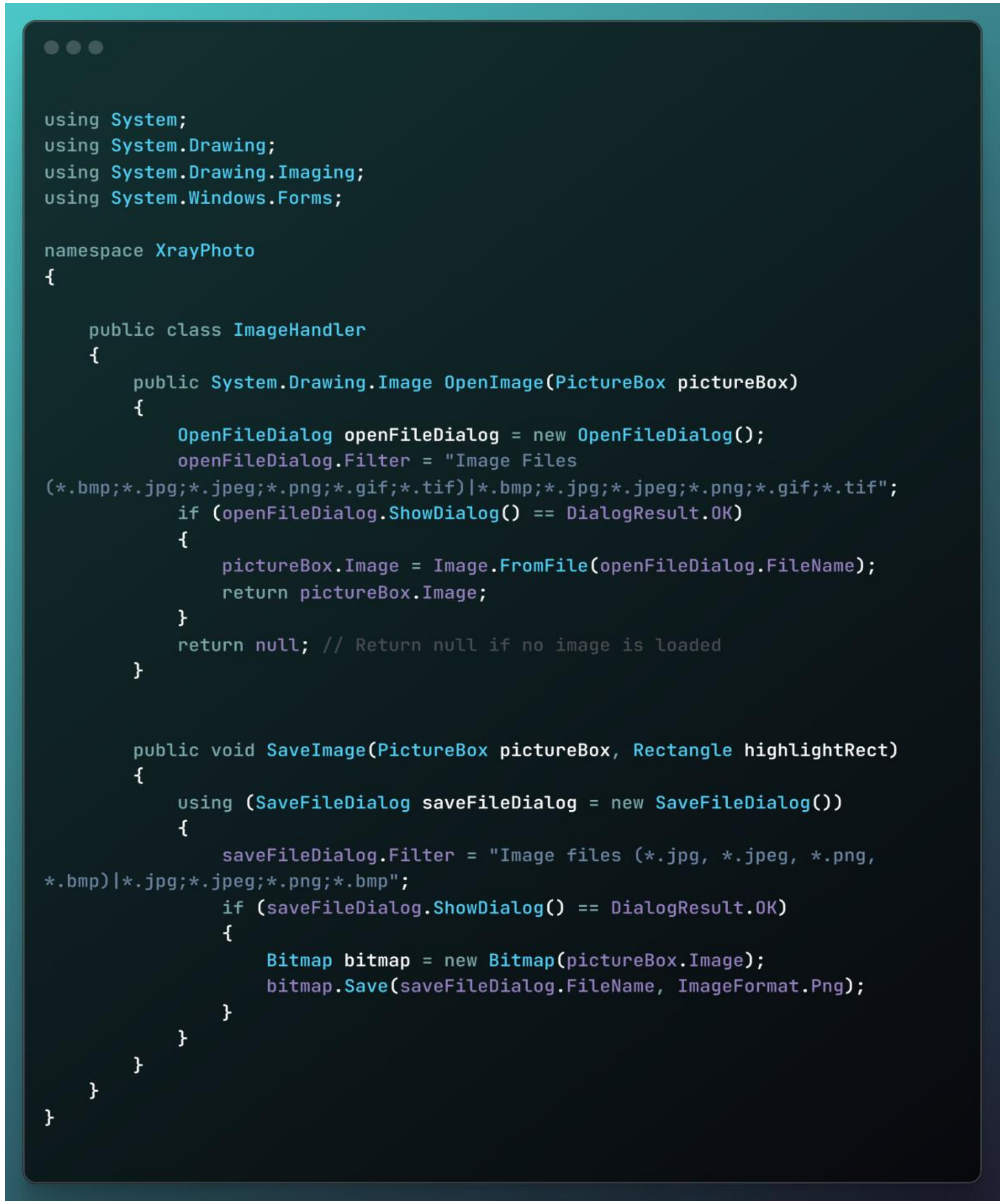


```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace XrayPhoto
{
    public class HighlightManager
    {
        private PictureBox pictureBox;

        private Image originalImage; // Reference to the original image
        public Rectangle SelectionRect { get; private set; }
        private bool isSelecting;
        private Point startPoint;
        private Point pictureBoxOffset;
        public HighlightManager(PictureBox pictureBox)
        {
            this.pictureBox = pictureBox;
            this.pictureBox.Paint += PictureBox_Paint;
            pictureBoxOffset = new Point(pictureBox.Location.X,
pictureBox.Location.Y);
        }

        public void SetOriginalImage(Image image)
        {
            originalImage = image;
        }

        public void PictureBox_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                isSelecting = true;
                startPoint = e.Location;
                SelectionRect = new Rectangle(startPoint, new Size(0, 0)); //
Initialize selection rectangle
                pictureBox.Invalidate();
            }
        }

        public void PictureBox_MouseMove(object sender, MouseEventArgs e)
        {
            if (isSelecting)
            {
                int x = Math.Min(e.X, startPoint.X);
                int y = Math.Min(e.Y, startPoint.Y);
                int width = Math.Abs(e.X - startPoint.X);
                int height = Math.Abs(e.Y - startPoint.Y);
                SelectionRect = new Rectangle(x, y, width, height);
                pictureBox.Invalidate();
            }
```



```
        public void PictureBox_MouseUp(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left && isSelecting)
            {
                isSelecting = false;
                int x = Math.Min(e.X, startPoint.X);
                int y = Math.Min(e.Y, startPoint.Y);
                int width = Math.Abs(e.X - startPoint.X);
                int height = Math.Abs(e.Y - startPoint.Y);
                SelectionRect = new Rectangle(x, y, width, height);
                pictureBox.Invalidate();
            }
        }

        public void DrawSelectionRect(Graphics g)
        {
            if (originalImage != null && !SelectionRect.IsEmpty)
            {
                // Convert selection rectangle coordinates from PictureBox to
image coordinates
                RectangleF rect = GetImageRectangle(SelectionRect,
originalImage.Size, pictureBox.ClientSize);

                using (Pen pen = new Pen(Color.Red, 2))
                {
                    g.DrawRectangle(pen, rect.X, rect.Y, rect.Width, rect.Height);

                }
            }
        }

        private RectangleF GetImageRectangle(Rectangle rect, Size imageSize, Size
controlSize)
        {
            // Calculate the scaling factors for X and Y dimensions
            float scaleX = (float)imageSize.Width / controlSize.Width;
            float scaleY = (float)imageSize.Height / controlSize.Height;

            // Convert the coordinates of the selection rectangle from PictureBox
to image coordinates
            float x = rect.X * scaleX;
            float y = rect.Y * scaleY;
            float width = rect.Width * scaleX;
            float height = rect.Height * scaleY;

            // Return the transformed rectangle
            return new RectangleF(x, y, width, height);
        }

        private void PictureBox_Paint(object sender, PaintEventArgs e)
        {
            DrawSelectionRect(e.Graphics);
        }


    }
}
```

- هذا الجزء من الكود
- يتم استخدام هذا الجزء من الكود لتمكين التحديد والرسم على الصورة المعروضة في صندوق الصورة واستخدام مستطيل التحديد لعمليات أخرى مثل تلوين المناطق التالفة في الصورة الشعاعية.



```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Windows.Forms;

namespace XrayPhoto
{

    public class ImageHandler
    {
        public System.Drawing.Image OpenImage(PictureBox pictureBox)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.Filter = "Image Files
(*.bmp;*.jpg;*.jpeg;*.png;*.gif;*.tif)|*.bmp;*.jpg;*.jpeg;*.png;*.gif;*.tif";
            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                pictureBox.Image = Image.FromFile(openFileDialog.FileName);
                return pictureBox.Image;
            }
            return null; // Return null if no image is loaded
        }


        public void SaveImage(PictureBox pictureBox, Rectangle highlightRect)
        {
            using (SaveFileDialog saveFileDialog = new SaveFileDialog())
            {
                saveFileDialog.Filter = "Image files (*.jpg, *.jpeg, *.png,
*.bmp)|*.jpg;*.jpeg;*.png;*.bmp";
                if (saveFileDialog.ShowDialog() == DialogResult.OK)
                {
                    Bitmap bitmap = new Bitmap(pictureBox.Image);
                    bitmap.Save(saveFileDialog.FileName, ImageFormat.Png);
                }
            }
        }
    }
}
```