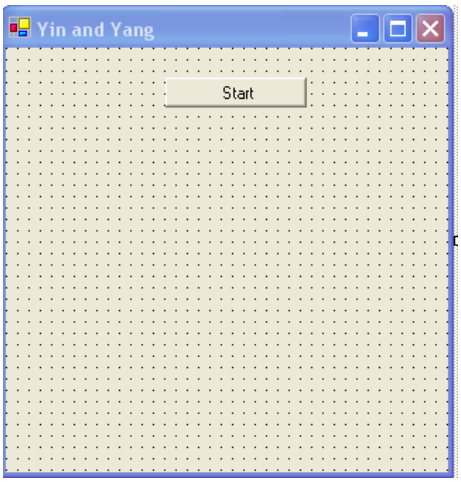
Name: **Session:**
Programming II
Lab Exercise 4/21/2022

In this lab, we will write some programs that will create some interesting graphics. When your programs are running, submit a screen shot of your running programs.

Part I – The Yin-Yang

1. Create a form with a button on it.



The form should have a size of 400 x 400.

2. Add the following code to the btnStart click event.

```
Graphics g = CreateGraphics();
SolidBrush black_pen = new SolidBrush(Color.Black);
SolidBrush white_pen = new SolidBrush(Color.White);
g.FillEllipse(black_pen, 100, 100, 200, 200);

//Making a one-second pause between steps
System.Threading.Thread.Sleep(1000);
g.FillEllipse(white_pen, 150, 100, 100, 100);
System.Threading.Thread.Sleep(1000);
g.FillPie(white_pen, 100, 100, 200, 200, -90, 180);
System.Threading.Thread.Sleep(1000);
g.FillEllipse(black_pen, 150, 200, 100, 100);
System.Threading.Thread.Sleep(1000);
g.FillEllipse(black_pen, 185, 135, 30, 30);
g.FillEllipse(white_pen, 185, 235, 30, 30);
```

3. Run your program and see what happens.

Part II – The L-System

An **L-system** or **Lindenmayer system** is a formal grammar (a set of rules and symbols) most famously used to model the growth processes of plant development, but also able to model the morphology of a variety of organisms. L-systems can also be used to generate self-similar fractals such as iterated function systems.

1. Create a form that has a size of 400 x 500.
2. Add a Timer to your form. Make sure Enable is true and Interval is 100 msec.
3. Add the following global declarations:

```
string[] F = new string[50];
int[] X = new int[10];
int[] Y = new int[10];
double[] Fi = new double[10];
double[] Finew = new double[10];
double[] Xtemp = new double[10];
double[] Ytemp = new double[10];
double[] Fitemp = new double[10];
double[] Xtemp2 = new double[10];
double[] Ytemp2 = new double[10];
double[] Fitemp2 = new double[10];
double Fi0;

int k;
int Kmax;
int branch;
int[] i = new int[10];
int Xnew;
int Ynew;
```

4. Add the following code to the Form_Load procedure:

```
k = 1;
Kmax = 5;  // number of cycles

branch = (int)(300.0 / Math.Pow(Kmax, 2.5));  // the size of an element
X[0] = 800;  //the coordinate of the drawing starting point
Y[0] = 600;
Fi[0] = 0; //the drawing starting angle
Fi0 = Math.PI / 8; //angle rotation
```

```
//the drawing procedure
F[1] = "-";  //this designates rotation left
F[2] = "f"; //this designates a step forward
F[3] = "+";  //this designates rotation right
F[4] = "f";
F[5] = "+";
F[6] = "[";
F[7] = "+";
F[8] = "f";
F[9] = "-";
F[10] = "f";
F[11] = "-";
F[12] = "]";
F[13] = "-";
F[14] = "[";
F[15] = "-";
F[16] = "f";
F[17] = "+";
F[18] = "f";
F[19] = "+";
F[20] = "f";
F[21] = "]";
```

5. Write a function Circle( )

```
private void Circle()
{
   X[k] = X[k - 1];
   Y[k] = Y[k - 1];
   Fi[k] = Fi[k - 1];

   for (i[k] = 1;  i[k] < 50; i[k]++)
   {
      if (F[i[k]] == "-")
         Fi[k] = Fi[k] - Fi0;

      if (F[i[k]] == "f")
      {
         if (k == Kmax)
         {
            Draw();
            X[k] = Xnew;
            Y[k] = Ynew;
         }
         else
         {
            k = k + 1;
            Circle();
            k = k - 1;
            X[k] = X[k + 1];
            Y[k] = Y[k + 1];
         }
      }
      if (F[i[k]] == "+")
         Fi[k] = Fi[k] + Fi0;

      if (F[i[k]] == "[")
      {
         Xtemp[k] = X[k];
         Ytemp[k] = Y[k];
         Fitemp[k] = Fi[k];
      }
      if (F[i[k]] == "]")
      {
         X[k] = Convert.ToInt32(Xtemp[k]);
         Y[k] = Convert.ToInt32(Ytemp[k]);
         Fi[k] = Fitemp[k];
      }
   }
}
```

6. Write a procedure Draw( )

```
private void Draw()
{
   Xnew = Convert.ToInt32(X[k] + branch * Math.Sin(Fi[k]));
   Ynew = Convert.ToInt32(Y[k] - branch * Math.Cos(Fi[k]));
   Graphics g  = CreateGraphics();
   g.TranslateTransform(-450, -150);
   g.DrawLine(Pens.Green, X[k], Y[k], Xnew, Ynew);
}
```

7. Add the following code to the Timer1_Tick event:

```
Circle();
timer1.Enabled = false;
```

8. Now try your program. Try modifying several of the parameters such as Kmax and Angle Rotation.

Part III – The Sierpanski Triangle

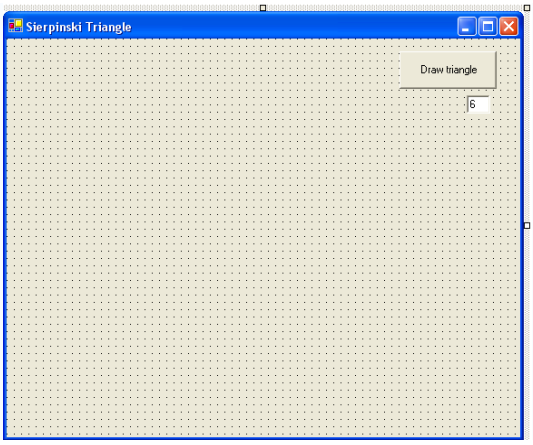
The **Sierpinski triangle**, also called the **Sierpinski gasket**, is a fractal named after Wacław Sierpiński who described it in 1915.

An algorithm for obtaining arbitrarily close approximations to the Sierpinski triangle is as follows:

1. Start with any triangle in a plane (any closed, bounded region in the plane will actually work). The canonical Sierpinski triangle uses an equilateral triangle with a base parallel to the horizontal axis (first image).
2. Shrink the triangle by ½, make two copies, and position the three shrunken triangles so that each triangle touches the two other triangles at a corner (image 2).
3. Repeat step 2 with each of the smaller triangles (image 3 and so on).

Now let's write a program that will generate this….

1. Create a Form with the dimensions of 700 x 600.
2. Add a Button and a TextBox



3. Add the following code to the btnDraw_Click event

```
Graphics g = CreateGraphics();
g.Clear(BackColor);
DrawTriangle(g, 320, 10, 600, 470, 40, 470);
DrawSierpinski(g, 320, 10, 600, 470, 40, 470, Convert.ToInt32(txtLevel.Text));
txtLevel.Text = "";
txtLevel.Focus();
```

4. Add a DrawTriangle function:

```
private void DrawTriangle(Graphics g, float x1, float y1, float x2, float y2,
                          float x3, float y3)
{
   //Drawing a triangle
   g.DrawLine(Pens.Red, x1, y1, x2, y2);
   g.DrawLine(Pens.Red, x2, y2, x3, y3);
   g.DrawLine(Pens.Red, x3, y3, x1, y1);
}
```

5. Add a DrawSierpinski function:

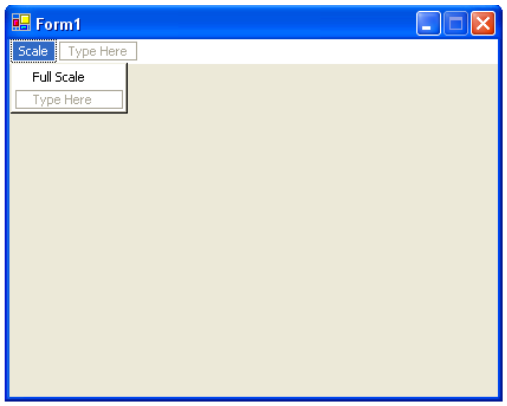
```
private void DrawSierpinski(Graphics g, float x1, float y1, float x2, float y2,
                            float x3, float y3, int counter)
{
   float x1n, y1n, x2n, y2n, x3n, y3n;

   if (counter > 0)
   {
      x1n = (x1 + x2) / 2;
      y1n = (y1 + y2) / 2;
      x2n = (x2 + x3) / 2;
      y2n = (y2 + y3) / 2;
      x3n = (x3 + x1) / 2;
      y3n = (y3 + y1) / 2;
      DrawTriangle(g, x1n, y1n, x2n, y2n, x3n, y3n);

      DrawSierpinski(g, x1, y1, x1n, y1n, x3n, y3n, counter - 1);
      DrawSierpinski(g, x2, y2, x1n, y1n, x2n, y2n, counter - 1);
      DrawSierpinski(g, x3, y3, x2n, y2n, x3n, y3n, counter - 1);
   }
}
```

6. Now run your program and see what it does. Try modifying some of your parameters and see what it does….

Part IV – The Mandelbrot Set

The Mandelbrot set, named after Benoit Mandelbrot, is a fractal. Fractals are objects that display self-similarity at various scales. Magnifying a fractal reveals small-scale details similar to the large-scale characteristics. Although the Mandelbrot set is self-similar at magnified scales, the small scale details are not identical to the whole. In fact, the Mandelbrot set is infinitely complex. Yet the process of generating it is based on an extremely simple equation involving complex numbers.

1. Create a Form with the dimensions of 421 x 337
2. Add a MenuStrip Item Scale|Full Scale
3. Add a PictureBox to your Form with the dimensions of 413 x 282



4. Add the following global variables.

```
private bool m_DrawingBox;
private int m_X1, m_Y1, m_X2, m_Y2;
private Bitmap m_ZoomingBitmap;
private Graphics m_ZoomingGraphics;

//Graphical variables
private Bitmap m_Bitmap;
private Graphics m_Graphics = null;

//Coordinates
private const double MIN_X = -2.2;
private const double MAX_X = 1;
private const double MIN_Y = -1.2;
private const double MAX_Y = 1.2;
private double m_Wxmin = MIN_X;
private double m_Wxmax = MAX_X;
private double m_Wymin = MIN_Y;
private double m_Wymax = MAX_Y;

public const int MaxIterations = 128;

//Colors
private Color[] m_Colors;
```

5. Add the following code to the Form1_Load event

```
        //Drawing the first image
        //Loading the colors (array of 16 colors)
        m_Colors = new Color[16] {Color.Black, Color.Blue, Color.Green, Color.Cyan,
Color.Red, Color.Magenta, Color.Yellow, Color.White, Color.Gray, Color.LightBlue,
Color.LightGreen, Color.LightCyan, Color.LightSalmon, Color.LightPink,
Color.LightYellow, Color.WhiteSmoke};

        //Drawing the fractal
        DrawMandelbrot();
```

6. Add a DrawMandelbrot procedure

```
    private void DrawMandelbrot()
    {
        //Creating graphics objects
        //Clear the graphics
        if (! (m_Graphics == null))
        {
            PictureBox1.Image = null;
            m_Graphics.Dispose();
            m_Bitmap.Dispose();
        }

        //Create a new bitmap object
        m_Bitmap = new Bitmap(PictureBox1.ClientSize.Width,
PictureBox1.ClientSize.Height);

        m_Graphics = Graphics.FromImage(m_Bitmap);

        //Clearing the image
        m_Graphics.Clear(Color.Black);
        PictureBox1.Image = m_Bitmap;
        PictureBox1.Refresh();

        const double MAX_MAG_SQUARED = 4.0;

        AdjustAspect();
```

```
        int clr;
        double realC, imagC, dRealC, dImagC, realZ, imagZ, realZ2, imagZ2;

        int wid = PictureBox1.ClientSize.Width;
        int hgt = PictureBox1.ClientSize.Height;
        dRealC = (m_Wxmax - m_Wxmin) / (wid - 1);
        dImagC = (m_Wymax - m_Wymin) / (hgt - 1);

        realC = m_Wxmin;

        for (int i = 0; i <= wid - 1; i++)
        {
           imagC = m_Wymin;
           for (int j = 0; j <= hgt - 1; j++)
           {
              realZ = 0;
              imagZ = 0;
              realZ2 = 0;
              imagZ2 = 0;
              clr = 0;
              while (clr < MaxIterations && realZ2 + imagZ2 <MAX_MAG_SQUARED)
              {
                 realZ2 = realZ * realZ;
                 imagZ2 = imagZ * imagZ;
                 imagZ = 2 * imagZ * realZ + imagC;
                 realZ = realZ2 - imagZ2 + realC;
                 clr = clr + 1;
              }

              m_Bitmap.SetPixel(i, j, m_Colors[clr % 16]);

              imagC = imagC + dImagC;
           }

           realC = realC + dRealC;

           if (i % 10 == 0)
              PictureBox1.Refresh();
        }

        PictureBox1.Refresh();

        this.Text = "Mandelbrot (" + m_Wxmin.ToString("f2") +
",  " + m_Wymin.ToString("f2") + ")-(" + m_Wxmax.ToString("f2") + ", " +
m_Wymax.ToString("f2") + ")";
     }
```

7. Now add an AdjustAspect procedure

```
 private void AdjustAspect()
{
   double want_aspect, Picturebox1_aspect;
   double hgt, wid, mid;

   want_aspect = (m_Wymax - m_Wymin) / (m_Wxmax - m_Wxmin);

   //Calculate new aspect
   Picturebox1_aspect = 1.0 * PictureBox1.ClientSize.Height /
PictureBox1.ClientSize.Width;

   if (want_aspect > Picturebox1_aspect)
   {
      wid = (m_Wymax - m_Wymin) / Picturebox1_aspect;
      mid = (m_Wxmin + m_Wxmax) / 2;
      m_Wxmin = mid - wid / 2;
      m_Wxmax = mid + wid / 2;
   }
   else
   {
      hgt = (m_Wxmax - m_Wxmin) * Picturebox1_aspect;
      mid = (m_Wymin + m_Wymax) / 2;
      m_Wymin = mid - hgt / 2;
      m_Wymax = mid + hgt / 2;
   }
}
```

8. Now we will add the FullScaleToolStripMenuItem_Click code:

```
m_Wxmin = MIN_X;
m_Wxmax = MAX_X;
m_Wymin = MIN_Y;
m_Wymax = MAX_Y;
DrawMandelbrot();
```

9. Now run your program and see the wonderful graphic you have generated….

Now we are going to do something really tricky. We will make it so we can "zoom" into the fractal. We will add three procedures to PictureBox1; MouseDown, MouseMove, MouseUp. This will allow us to draw a "zoom" zone. Here is the code for these.

10. Now add the following Mouse event handlers

PictureBox1_MouseDown

```
//Starting drawing a rubber rectangle
m_DrawingBox = true;
m_X1 = e.X;
m_Y1 = e.Y;
m_X2 = m_X1;
m_Y2 = m_Y1;

//Making a copy of the current image
m_ZoomingBitmap = new Bitmap(m_Bitmap);
m_ZoomingGraphics = Graphics.FromImage(m_ZoomingBitmap);
```

PictureBox1_MouseMove

```
//Continuing drawing the rubber rectangle
if (!m_DrawingBox)
    return;

//Saving the angles
m_X2 = e.X;
m_Y2 = e.Y;

//Drawing a new rectangle
Graphics gr = PictureBox1.CreateGraphics();
Rectangle rect = new Rectangle();
rect.X = Math.Min(m_X1, m_X2);
rect.Y = Math.Min(m_Y1, m_Y2);
rect.Width = Math.Abs(m_X2 - m_X1);
rect.Height = Math.Abs(m_Y2 - m_Y1);
m_ZoomingGraphics.DrawImage(m_Bitmap, 0, 0);
m_ZoomingGraphics.DrawRectangle(Pens.White, rect);

//Outputting the result
PictureBox1.Image = m_ZoomingBitmap;
```

PictureBox1_MouseUp

```
//Finishing drawing the rubber rectangle
m_DrawingBox = false;

//Releasing resources
PictureBox1.Image = null;
m_ZoomingBitmap.Dispose();
m_ZoomingGraphics.Dispose();
m_ZoomingBitmap = null;
m_ZoomingGraphics = null;

//Saving the new angles
m_X2 = e.X;
m_Y2 = e.Y;

//Placing the selected coordinates in order
double x1, x2, y1, y2, factor;
x1 = Math.Min(m_X1, m_X2);
x2 = Math.Max(m_X1, m_X2);
y1 = Math.Min(m_Y1, m_Y2);
y2 = Math.Max(m_Y1, m_Y2);

//Converting the screen coordinates
factor = (m_Wxmax - m_Wxmin) / PictureBox1.ClientSize.Width;
m_Wxmax = m_Wxmin + x2 * factor;
m_Wxmin = m_Wxmin + x1 * factor;

factor = (m_Wymax - m_Wymin) / PictureBox1.ClientSize.Height;
m_Wymax = m_Wymin + y2 * factor;
m_Wymin = m_Wymin + y1 * factor;
//Drawing the fractal
DrawMandelbrot();
```

11. Now let us add the code to our Full Scale menu item event

```
m_Wxmin = MIN_X
m_Wxmax = MAX_X
m_Wymin = MIN_Y
m_Wymax = MAX_Y
DrawMandelbrot()
```