The Graphics Class

1. The Graphics Class

The `Graphics` object is part of the `System.Drawing` namespace. Whenever you use a `Form`, using System.Drawing is automatically added to the `Form` class.

A `Graphics` object is a drawable surface that sits on the `Form`. Declare a new `Graphics` object (or instance) as a field in the `Form`:

```
private Graphics graphics;
```

In the `Form's` constructor create a `Graphics` object, by calling the `Form's` `CreateGraphics()` method:

```
graphics = CreateGraphics();
```

We can then use the `Graphic's` object's methods to draw a line:

```
graphics.DrawLine(Pens.Red,  new Point(10, 10), new Point(20, 20));
```

This will draw a red line from the point (that is 10 pixels across and down from the top, left corner of the screen) to the point (that is 20 pixels across and down from the top, left corner of the screen).

To draw a rectangle:

```
graphics.DrawRectangle(Pens.Blue, new Rectangle(10, 20, 100, 50));
```

will draw a rectangle with a blue outline with its top left corner at the point (10,20), width 100 and height 50.

To colour in the rectangle:

```
graphics.FillRectangle(Brushes.Red, new Rectangle(10, 20, 100, 50));
```

will colour the above rectangle in red.

To set the font, use:

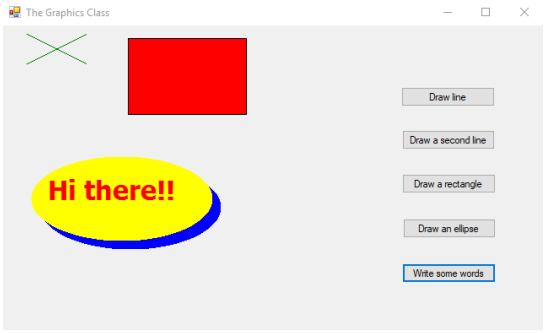
```
Font font = new Font("Consolas", 55, FontStyle.Italic);
```

To write text:

```
graphics.DrawString("This is fun?", font, Brushes.SkyBlue, new Point(20, 45));
```

Task 1: Using a Graphics Object

1. Create a new project.
2. On the Form, draw a green cross, a red rectangle with a black border a yellow ellipse with a blue shadowing ellipse and the slogan "Hi there!" as shown below.



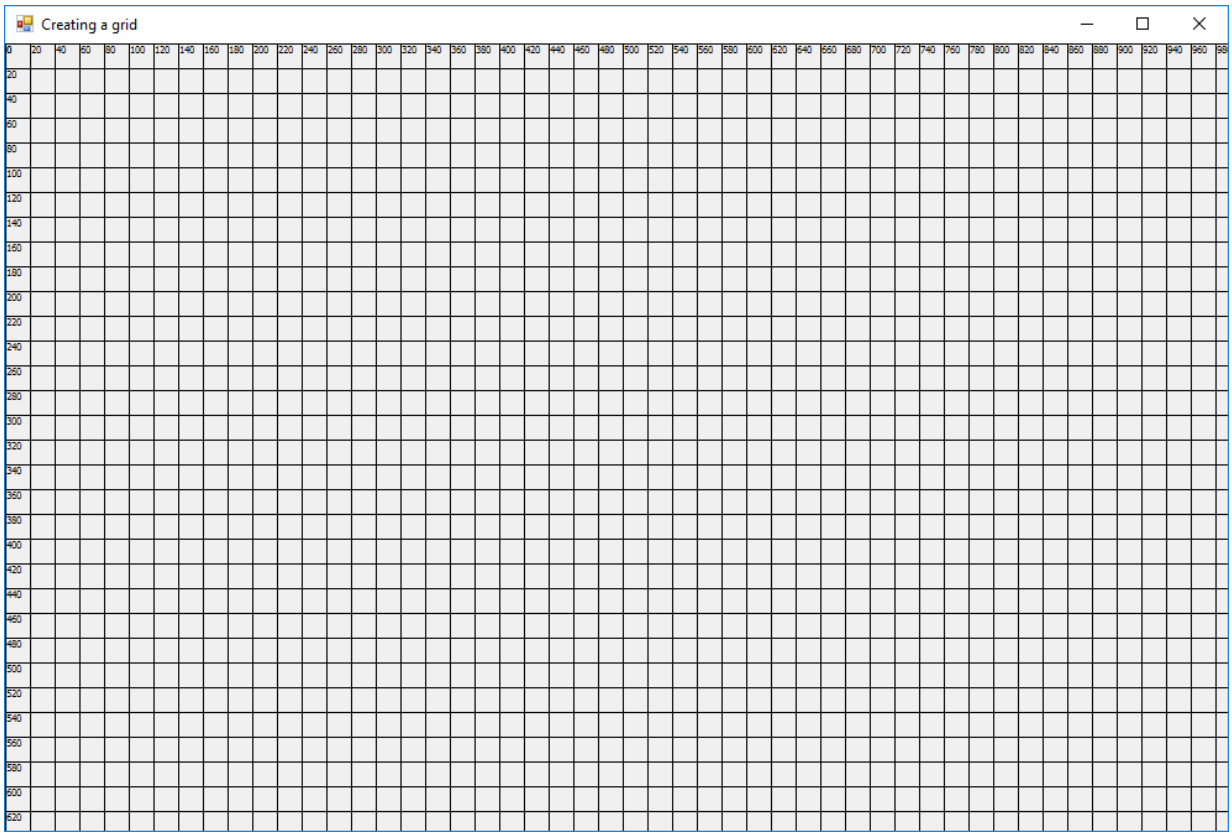
Task 2: Creating a Grid

1. Create a new project.
2. Most of your work with Graphics will involve thinking about your forms as a grid of (x, y) coordinates. Here is the skeleton to build the grid shown below. Fill in the missing parts.

```
private void Form1_MouseClick(object sender, MouseEventArgs e)
{
      Font font = new Font("Tahoma", 6, FontStyle.Regular);

      for (int x = 0; x < Width; x += 20)
      {
           //write your code here
      }

      for (int y = 0; y < Height; y += 20)
      {
           //write your code here
      }
 }
```



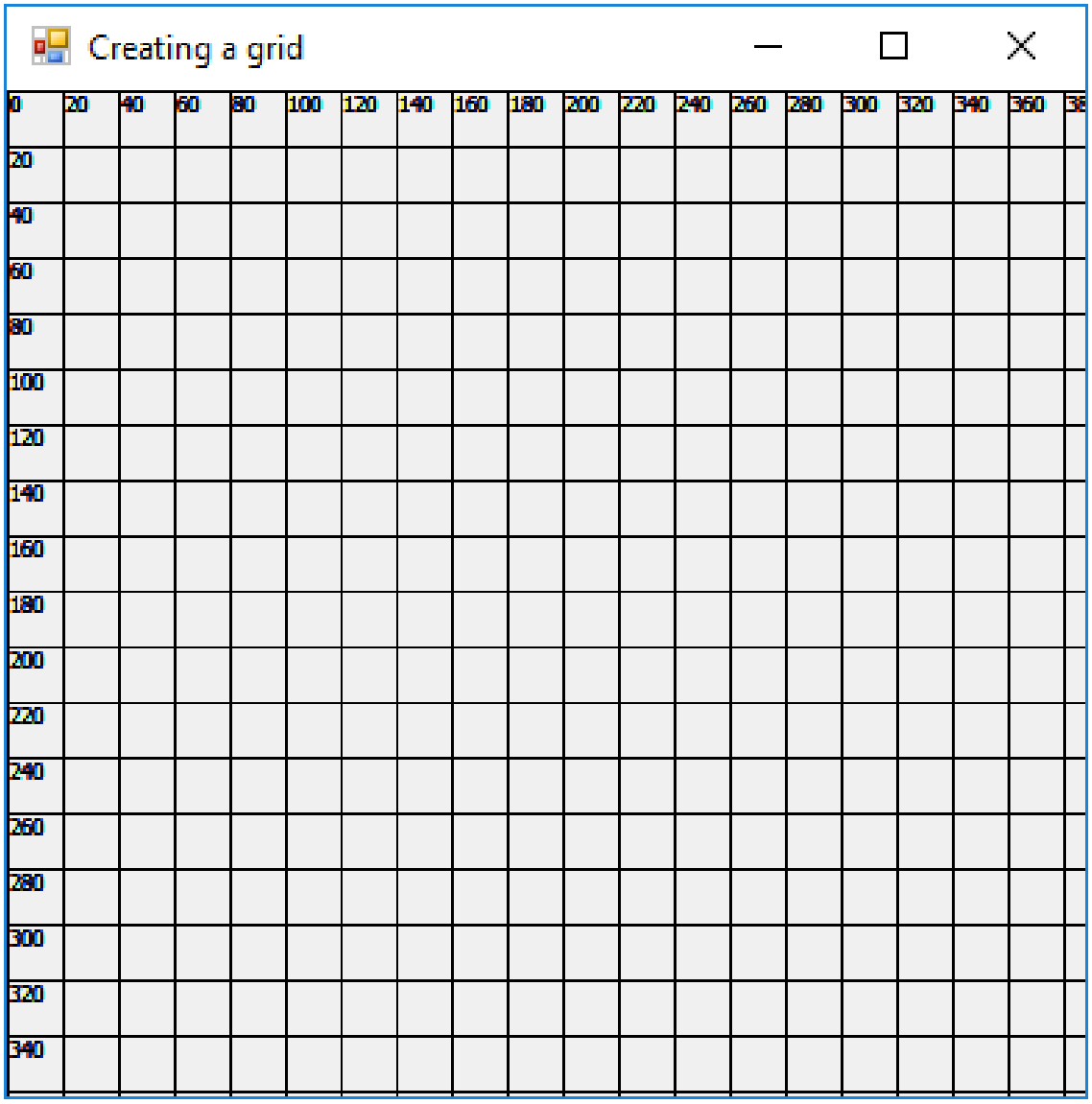
Task 3: Drawing on a Grid

1. What will be drawn when you run the code below? Notice that a curve is defined by an array of connected points.

```
Pen pen = new Pen(Brushes.Black, 3.0F);
graphics.DrawCurve(pen, new Point[] { new Point(80, 60), new Point(200, 40),
                new Point(180, 60), new Point(300, 40) });
graphics.DrawCurve(pen, new Point[] { new Point(300, 180), new Point(180, 200),
                new Point(200, 180), new Point(80, 200) });

graphics.DrawLine(pen, new Point(300, 40), new Point(300, 180));
graphics.DrawLine(pen, new Point(80, 60), new Point(80, 200));
graphics.DrawEllipse(pen, 40, 40, 20, 20);
graphics.DrawRectangle(pen, 40, 60, 20, 300);
graphics.DrawLine(pen, new Point(60, 60), new Point(80, 60));
graphics.DrawLine(pen, new Point(60, 200), new Point(80, 200));
```

2. Draw the output on the grid.



Task 4: Drawing Irregular Shapes

1. What will be drawn when you run the code below? Notice that a polygon is defined by an array of connected lines.

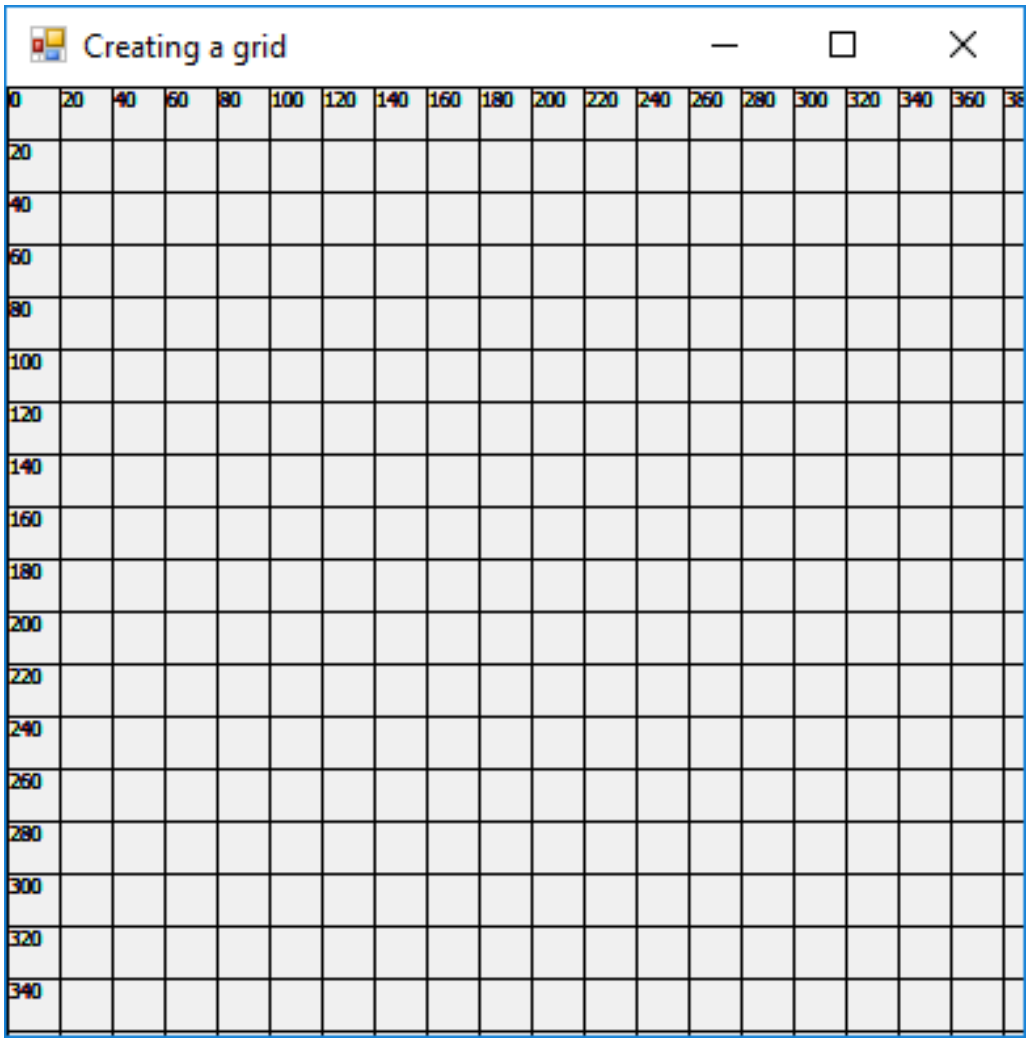
```
graphics.FillPolygon(Brushes.Black, new Point[] { new Point(60, 40), new
   Point(140, 80), new Point(200, 40), new Point(300, 80), new Point(380, 60),
   new Point(340, 140), new Point(320, 180), new Point(380, 240), new Point
   (320, 300), new Point(340, 340), new Point(240, 320), new Point(180, 340),
   new Point(20, 320), new Point(60, 280), new Point(100, 240), new Point(40,
   220), new Point(80, 160) });

Font font = new Font("Times New Roman", 24, FontStyle.Italic);

graphics.DrawString("Pow!", font, Brushes.White, new Point(80, 80));
graphics.DrawString("Pow!", font, Brushes.White, new Point(120, 120));
graphics.DrawString("Pow!", font, Brushes.White, new Point(160, 160));
graphics.DrawString("Pow!", font, Brushes.White, new Point(200, 200));
graphics.DrawString("Pow!", font, Brushes.White, new Point(240, 240));
```

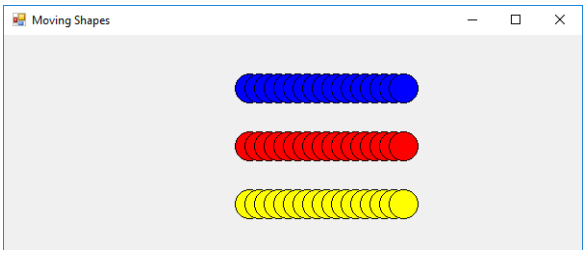
3. Draw the output on the grid.

4. Could you rewrite the `DrawString` statements with a variable and a loop?

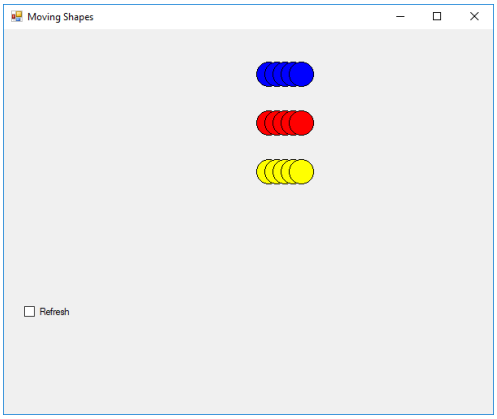


Task 5: Moving Shapes

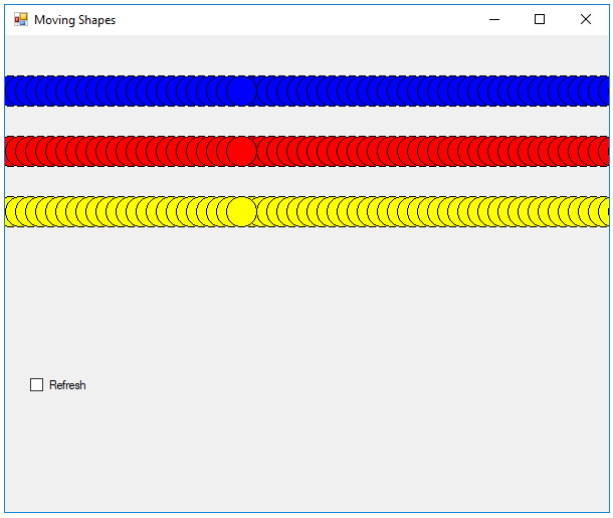
1. Create a new project.
2. Write a `Timer1_tick` handler to draw three circles on the Form. Use a different colour for each circle.
3. Place a Timer on the Form. Modify your program so that, when the program starts, the figures are drawn automatically, and each 100 msec, the figures shift 20 pixels to the right. The example below shows the screen after several Timer cycles.



4. Add a CheckBox to your form. As the program runs, the user can turn the screen refresh feature on and off by checking and unchecking this CheckBox. That is, when the *Refresh* CheckBox is checked, the screen refreshes at each cycle; when the *Refresh* CheckBox is unchecked, the screen does not refresh as (as in the image on the right).



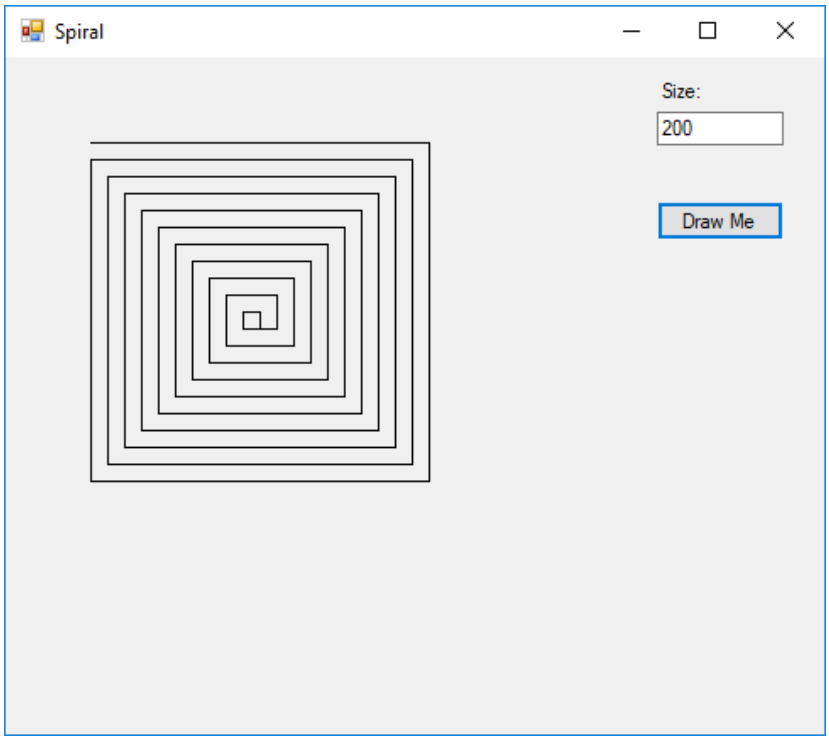
5. Modify your program so that when the drawings reach the right edge of the screen, they “wrap around” and come back in from the left edge.



Task 6: Spiral Pattern

******* **CHALLENGE** *******

Build a project that draws the geometric pattern shown below:



Start the drawing at the upper left corner. The length of the first line is equal to the number entered in the TextBox. Draw a square stopping 10 pixels short of the original starting point, then repeat. At each iteration, the sides of the square get shorter by 10 pixels. Repeat until the length of the side goes to zero.

Note that this problem can't be solved with just a single for loop, because the number of iterations depends on the initial side length (the value in the edit box).

The actual code required to solve this is minimal (the main loop of my solution contains only seven statements), but it's tricky. You may find that it helps to sketch out the figure, and look at how you can express the endpoints of the lines in terms of the starting x and y and the distance to be moved at each iteration.
