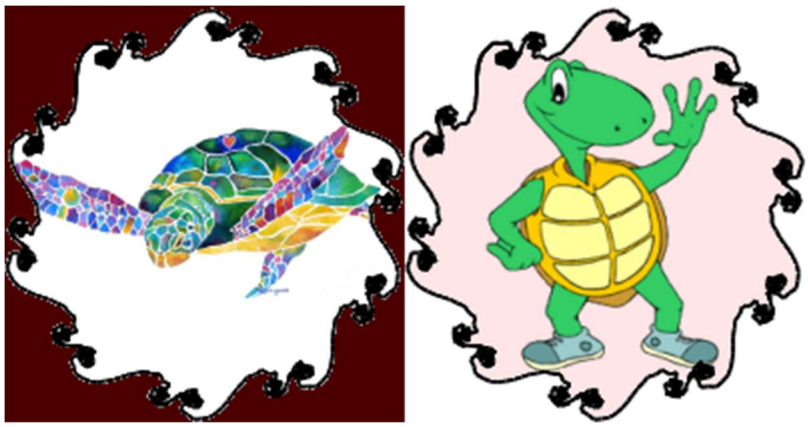
Practice 14: TURTLE GRAPHICS

14

Drawing with a Turtle

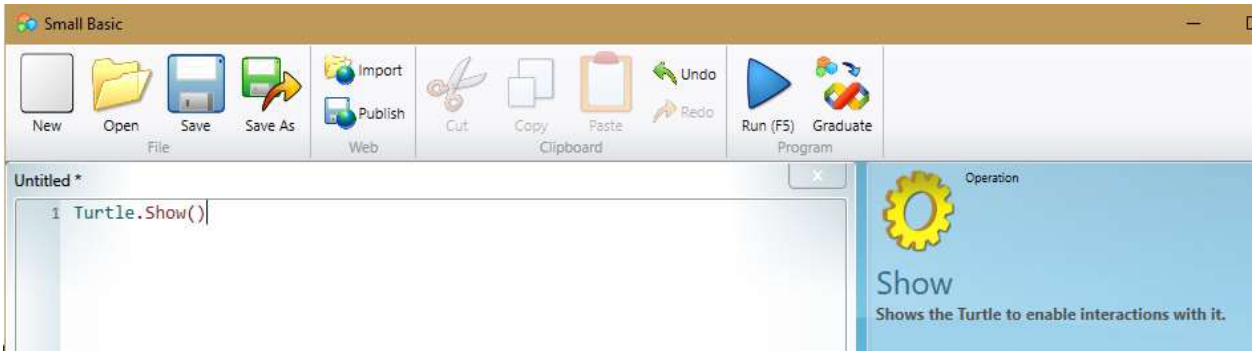
The Turtle object helps you draw interesting graphics in the graphics window. We'll show you how to move the Turtle on the screen by using the **Turtle** object and how to use the Turtle to draw colorful designs.



Show me the Turtle!

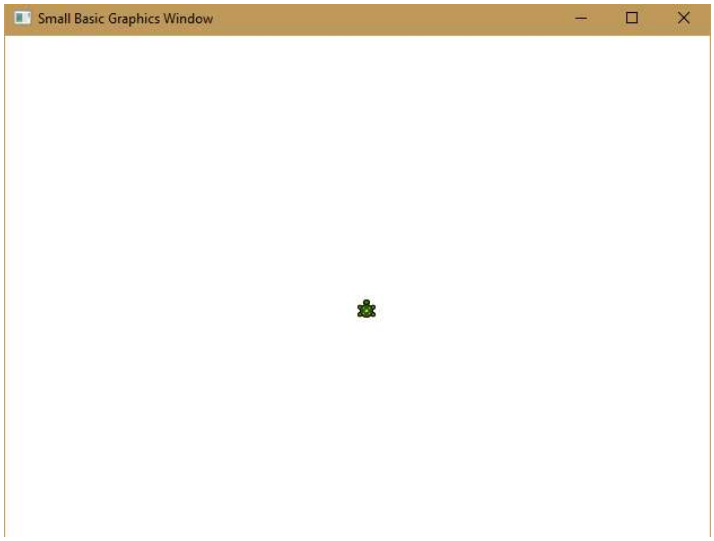
Meet your new friend in Small Basic—the Turtle! The Turtle helps you draw interesting shapes and patterns in the graphics window. You can display the Turtle by using the **Show** operation.

```
Turtle.Show()
```



Click the  button on the Toolbar.

The Turtle appears on the screen.



Controlling the Turtle Object

You can give commands to the Turtle by using the **Turtle** object. In turn, the Turtle draws graphics on the screen. Let's look at some of these functions…

To set the location of the Turtle on the screen, you use the **X** and **Y** properties of the **Turtle** object.

```
Turtle.X= 50
Turtle.Y = 200
```

To move the Turtle a particular distance in pixels, you use the **Move** operation. To move the Turtle to a particular location, you use the **MoveTo** operation and its parameters to specify the new location.

```
Turtle.Move(150)
Turtle.MoveTo(50, 200)
```

If you try these operations out, you'll notice that a line is drawn in the graphics window to show where the Turtle has moved from the starting position. To move the Turtle without

drawing a line, use the **PenUp** operation. To bring the line back you use the **PenDown** operation.

```
Turtle.PenUp()
Turtle.PenDown()
```

Let's look at more functions...

Choose how fast theTurtle moves by using the **Speed** property and specifying a value between 1 and 10. To see the Turtle's fastest speed, specify 10.

```
Turtle.Speed = 8
```

Rotate the Turtle by using the **Turn** operation and specifying an angle in degrees. Or instead, you can rotate the Turtle 90 degrees by using the **TurnRight** or **TurnLeft** operations, respectively.

```
Turtle.Turn(90)
Turtle.TurnLeft()
Turtle.TurnRight()
```

Turn the Turtle to a specific angle of rotation by using the **Angle** property and setting the angle (in degrees) that you want the Turtle to rotate. By default, the Turtle faces the top of the screen, which is an angle of 0 degrees. Here we set the turtle to face the right:

```
Turtle.Angle = 90
```

You may have noticed that some of these operations can be used for similar purposes. For example, you can rotate the Turtle to face the left side of the window by using any of the following strategies:

1. You can set the value of the **Angle** property as 270.
2. You can use the **Turn** operation and specify the following values: :
    - 270 if the Turtle is already facing the top of the window
    - 180 if the Turtle is already facing the right side of the window
    - 90 if the Turtle is already facing the bottom of the window
3. You can use the **TurnRight** operation three times if the Turtle is facing the top of the window, twice if the Turtle is facing the right side of the window, or once if the

Turtle is facing the bottom of the window. The Turtle always rotates to the right in clockwise direction.

You can use the **TurnLeft** operation once if the Turtle is facing the top of the window, twice if the Turtle is facing the right side of the window, or three times if the Turtle is facing the bottom of the window.l.

Making the Turtle Move

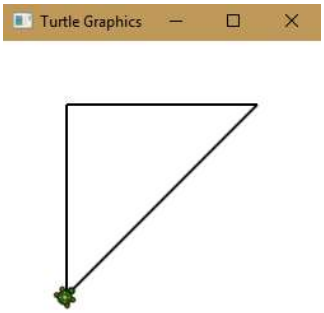
Let's draw a simple triangle by using the Turtle.

```
GraphicsWindow.Width = 250
GraphicsWindow.Height = 250
GraphicsWindow.Title = "Turtle Graphics"
Turtle.X = 50
Turtle.Y = 200
Turtle.Speed = 5
Turtle.Move(150)
Turtle.Turn(90)
Turtle.Move(150)

Turtle.MoveTo(50, 200)
Turtle.Angle = 45
```

Click the 

button on the Toolbar. This is the output you will see:



In this example, the Turtle draws a simple triangle on the screen. Let us understand the code in detail:

1. In the first three lines of code, you set the width, height, and title of the graphics window.
2. In the next two lines of code, you make the Turtle appear at a specific location in the graphics window. To specify the location, you set the value of the **X** property to a particular number of pixels from the left side of the graphics window, and you set the **Y** property to a particular number of pixels from the top of the graphics window.
3. You then set the speed at which the Turtle moves by specifying a value between 1 and 10 (inclusive) for the **Speed** property. To make the Turtle move at its fastest speed, specify 10. To make the Turtle move at its slowest speed, specify 1.
4. To make the Turtle draw the vertical side of the triangle, you use the **Move** operation to instruct the Turtle to draw 150 pixels from its original location and in its default direction (up). By default, the Turtle draws when you use the **Move** operation. If you want the Turtle to move without drawing, you use the **PenUp** operation.
5. To make the Turtle draw the horizontal side of the triangle, you first use the **Turn** operation to rotate the Turtle 90 degrees so that it faces the right side of the window. Then you use the **Move** operation to instruct the Turtle to draw 150 pixels from its new location and in its new direction.
6. To make the Turtle draw the diagonal side of the triangle, you use the **MoveTo** operation and specify a location in the window. To specify the location, you set the value of the **X** property to a particular number of pixels from the left side of the graphics window, and you set the value of the **Y** property to a particular number of pixels from the top of the graphics window.
7. Now that the triangle is complete, you rotate the Turtle by setting the value of the **Angle** property to 45 degrees.

Notice that the commands are written in the order we want them to happen. For example, we tell the Turtle to draw the vertical side of the triangle before it draws the horizontal side.

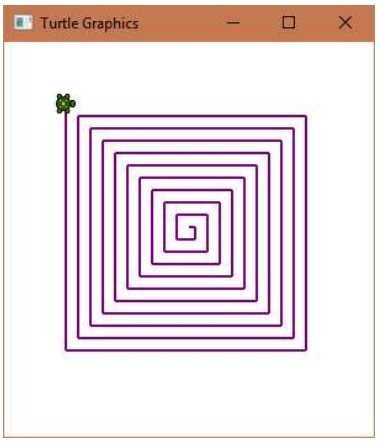
14

Having Fun with the Turtle

Now that you know how to communicate with the Turtle, let's have some fun with it. We'll use the For loop and instruct the Turtle to move and turn at specific distances and angles, creating a unique graphical design in the process.

```
GraphicsWindow.Title = "Turtle Graphics"
GraphicsWindow.Height = 320
GraphicsWindow.Width = 300
GraphicsWindow.PenColor = "Purple"
Turtle.Show()
Turtle.Speed = 8
Turtle.X = 150
Turtle.Y = 150
For i = 0  To 200 Step 5
   Turtle.Move(i)
   Turtle.Turn(90)
EndFor
```

*You can add color to your design by specifying a value for the **PenColor** property of the **GraphicsWindow** object.* This is the output you will see:

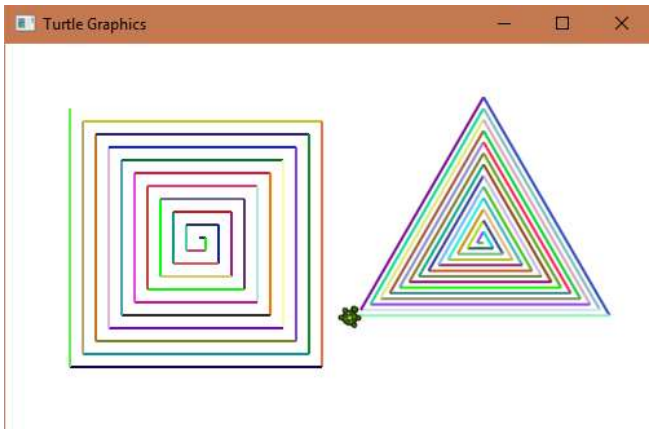


When you click **Run** on the toolbar or press F5 on the keyboard, the Turtle draws a colored, square design in the graphics window. *You can also draw multiple, colorful designs by using the Turtle.* For example, this program produces different shapes in a variety of sizes and colors.

```
GraphicsWindow.Title = "Turtle Graphics"
GraphicsWindow.Height = 300
```

```
GraphicsWindow.Width = 500
Turtle.Show()
Turtle.Speed = 10
Turtle.X = 150
Turtle.Y = 150
For i = 0  To 200 Step 5
  GraphicsWindow.PenColor =
GraphicsWindow.GetRandomColor()
  Turtle.Move(i)
  Turtle.Turn(90)
EndFor
Turtle.PenUp()
Turtle.Move(260)
Turtle.Turn(60)
Turtle.Move(120)
Turtle.PenDown()
For i = 0  To 200 Step 5
   GraphicsWindow.PenColor = GraphicsWindow.GetRandomColor()
   Turtle.Move(i)
   Turtle.Turn(120)
EndFor
```

The output you will see is below:



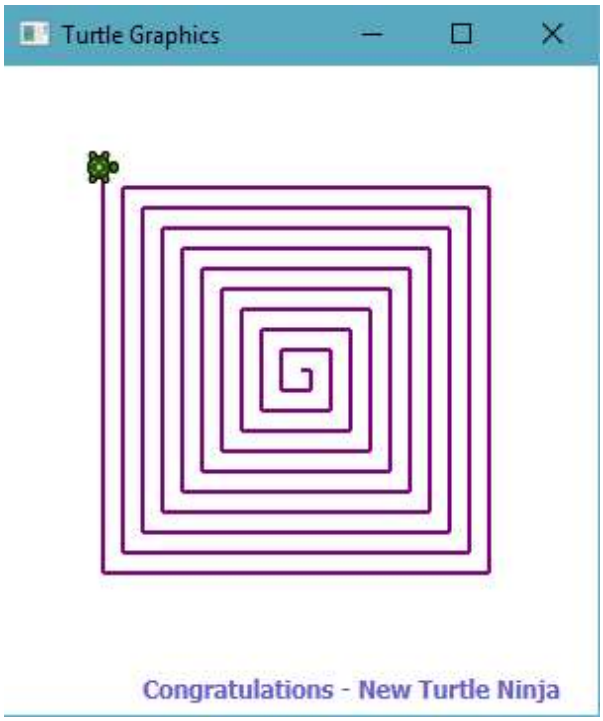
In the earlier example, we drew one set of nested squares of the same color. However, you can draw more than one shape in the same graphics window by using the **PenUp** and **PenDown** operations. You can also create nested versions of different shapes, such as triangles, by assigning a For loop and changing the distance and the angles.

In addition, you can also create shapes in a variety of colors by using the **GetRandomColor** operation to set the value of the **PenColor** property.

14

When you click **Run** on the toolbar or press F5 on the keyboard, the Turtle draws two colorful designs in the graphics window.

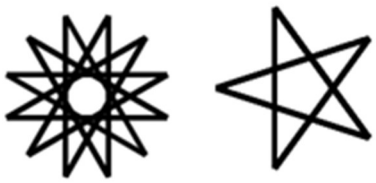
Congratulations. You are the new Turtle Ninja ☺



14

Challenge: Draw Stars

Use the Turtle to draw the stars shown below. Hint: adjust the angle for each turn to draw the star you like.



Discussion Questions

1. We learned how to draw straight lines by asking Turtle to move and make turns at certain angles. Can you find a way to draw curved lines with Turtle?
2. How can you draw a dashed line with the Turtle?
3. Look back at Making the Turtle Move. What would happen if you typed some of the commands in a different order? For example, if you moved the Turtle.Speed command to the end of the list? Why is this?
4. Design a maze and let Turtle solve it. What other fun games can you think of?

Additional Resources

- The Developer's Reference Guide to Small Basic: Chapter 5: GraphicsWindow Object
  - Http://aka.ms/SmallBasic/Graphics
- Turtle Polygon Patterns
  - http://aka.ms/SmallBasic/Turtle