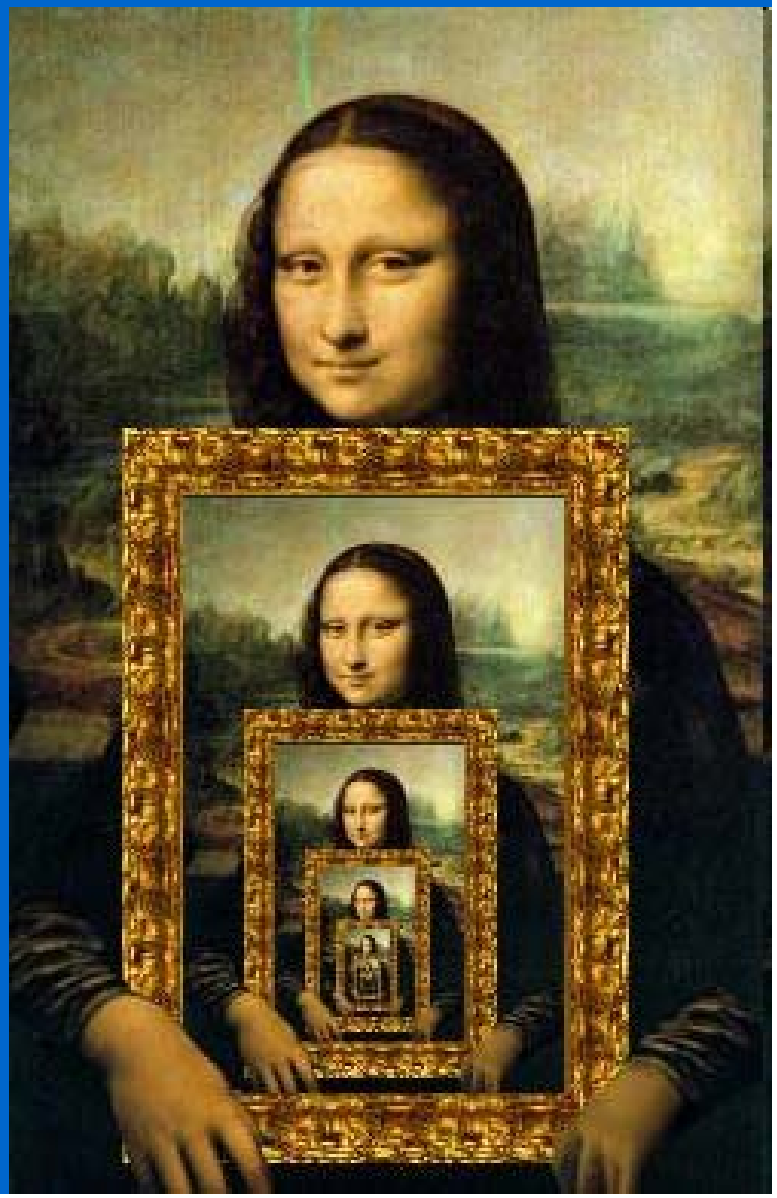
COMP1022Q
Introduction to Computing with Excel VBA

Recursion

David Rossiter and Gibson Lam

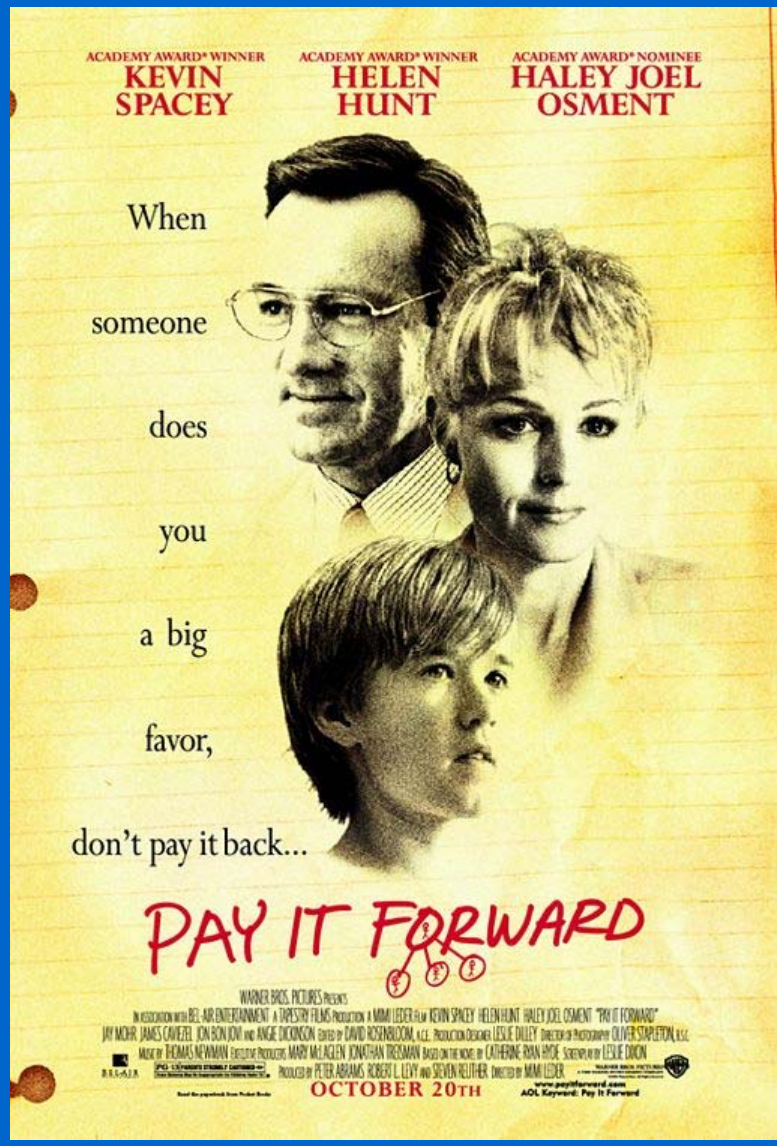
What is Recursion?

- A recursive function is one which calls itself
- Recursive functions are sometimes very useful for some computing tasks
- For example, you can use one cleverly written small recursive function instead of lots of lines of code



'Pay It Forward'

- A movie about a boy who has been asked to think of a plan that will change the world
- He comes up with a plan that when someone receives a good deed, he/she helps 3 different other people



'Pay It Forward' Pseudo-Code

- *Pseudo-code* is used to show the general idea of a procedure

Sub Help(Benefactor, Person)

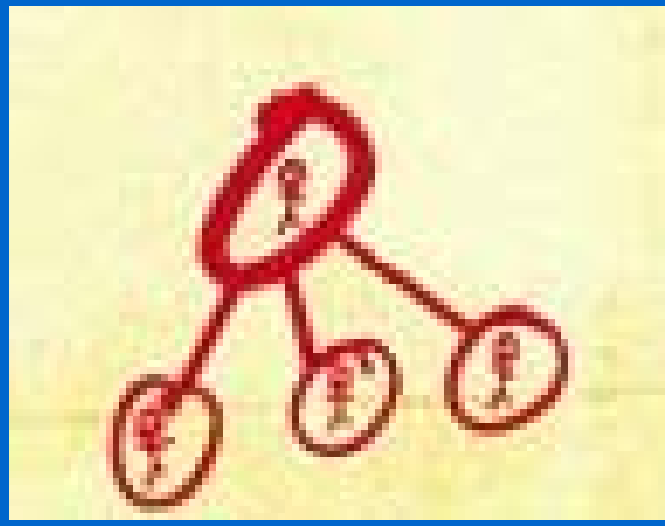
Person receives help from *Benefactor*

Help Person, RandomPerson1

Help Person, RandomPerson2

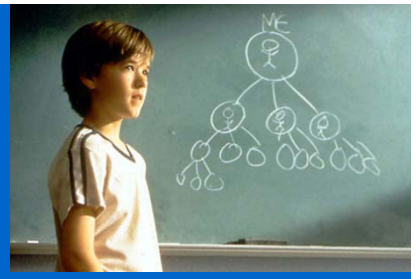
Help Person, RandomPerson3

End Sub

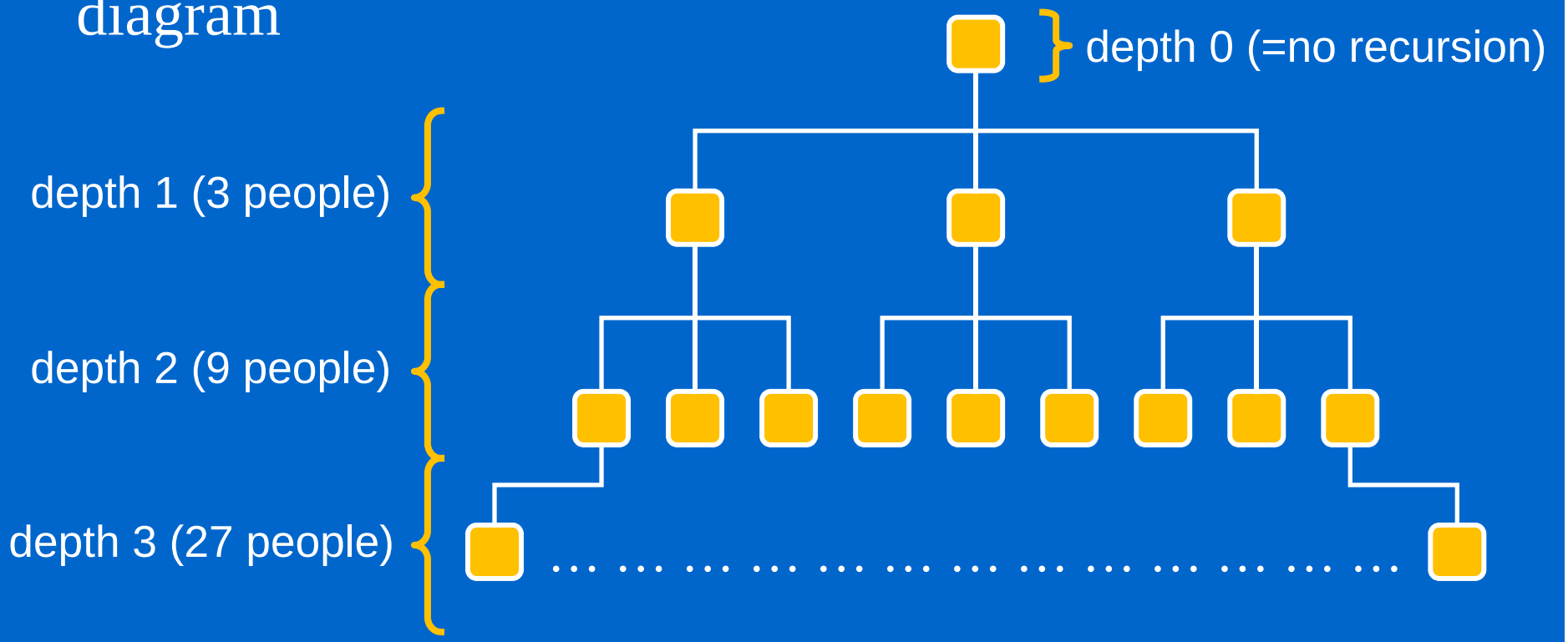


- The whole process starts with one person helping another, for example: Help Me, You
- The above example uses pseudo-code, but the rest of this presentation uses real VBA code

Recursive Depths



- How many good deeds are done in total after 3 depths?
- You can see what we mean by *depth* in the following diagram



- The answer is that when the maximum depth is 3, 1+3+9+27=40 good deeds in total are done

A Recursive Function in VBA

- Here is a recursive function:

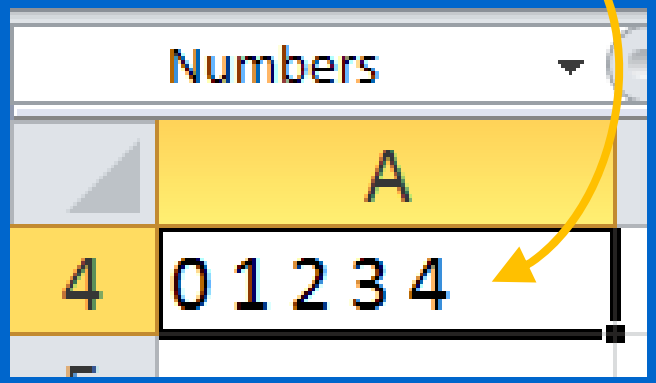
```
Sub RecursiveFunction(Number)
    If Number < 5 Then
        Range("Numbers") = Range("Numbers") _
                           & Str(Number)
        RecursiveFunction Number + 1
    End If
End Sub
```

The cell named 'Numbers'

- Let's start the recursion using this code:

```
RecursiveFunction 0
```

- The result is shown on the right:



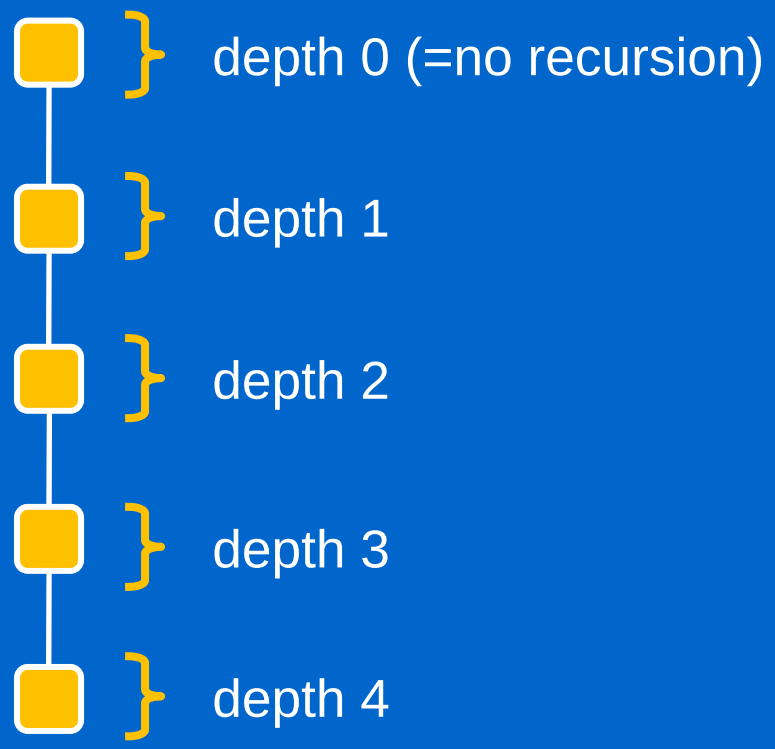
- This is the execution of the code `RecursiveFunction 0`

```
RecursiveFunction 0
Sub RecursiveFunction(0)
  Range("Numbers") = Range("Numbers") & Str(0)
  RecursiveFunction 0 + 1
    Sub RecursiveFunction(1)
      Range("Numbers") = Range("Numbers") & Str(1)
      RecursiveFunction 1 + 1
        Sub RecursiveFunction(2)
          Range("Numbers") = Range("Numbers") & Str(2)
          RecursiveFunction 2 + 1
            Sub RecursiveFunction(3)
              Range("Numbers") = Range("Numbers") & Str(3)
              RecursiveFunction 3 + 1
                Sub RecursiveFunction(4)
                  Range("Numbers") = Range("Numbers") & Str(4)
                  RecursiveFunction 4 + 1
                End Sub
            End Sub
        End Sub
    End Sub
End Sub
```

There are no more function calls when this value becomes 5, because of the If statement

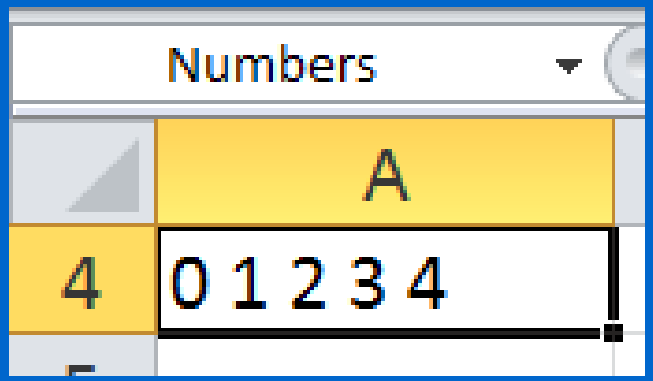
Recursive Depths

- So for the example shown on the last slide, the pattern of depth looks like this:



Recursive and Iterative Functions

- The recursive example discussed in the last two slides generates a result of ‘0 1 2 3 4’



- On the next slides we will show two *iterative* code examples which produce the same result
- ‘Iterative’ means ‘looping without recursion’

Iterative Loop 1

```
Dim Counter As Integer

Counter = 0
Do
    Range("Numbers") = Range("Numbers") & _
                       Str(Counter)
    Counter = Counter + 1
Loop Until Counter = 5
```

Iterative Loop 2

```
Dim Counter As Integer

Counter = 0
While Counter < 5
    Range("Numbers") = Range("Numbers") & _
                       Str(Counter)
    Counter = Counter + 1
Wend
```

- You can write recursive code and iterative code which do the same thing
- However, sometimes it is easier to write things using recursion, as you will see later

Changing the Order

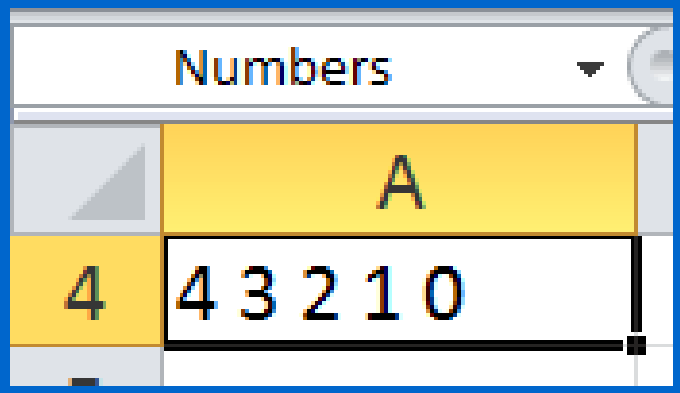
- These two lines have been swapped:

```
Sub RecursiveFunction(Number)
    If Number < 5 Then
        RecursiveFunction Number + 1
        Range("Numbers") = Range("Numbers") _
                           & Str(Number)
    End If
End Sub
```

- Let's start the recursion using this code:

```
RecursiveFunction 0
```

- The result is shown on the right:



- This is the execution of the code `RecursiveFunction 0`

```
RecursiveFunction 0
Sub RecursiveFunction(0)
  RecursiveFunction 0 + 1
  Sub RecursiveFunction(1)
    RecursiveFunction 1 + 1
    Sub RecursiveFunction(2)
      RecursiveFunction 2 + 1
      Sub RecursiveFunction(3)
        RecursiveFunction 3 + 1
        Sub RecursiveFunction(4)
          RecursiveFunction 4 + 1
          Range("Numbers") = Range("Numbers") & Str(4)
        End Sub
        Range("Numbers") = Range("Numbers") & Str(3)
      End Sub
      Range("Numbers") = Range("Numbers") & Str(2)
    End Sub
    Range("Numbers") = Range("Numbers") & Str(1)
  End Sub
  Range("Numbers") = Range("Numbers") & Str(0)
End Sub
```

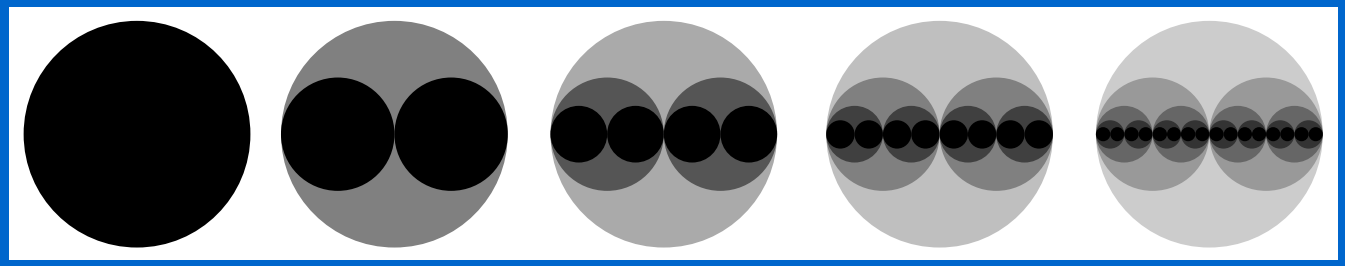
There are no more function calls when this value becomes 5, because of the If statement

Drawing Recursive Circles

- Recursive functions are used for lots of purposes
- One of them is to generate computer graphics
- The next example draws circles recursively using lots of circle shape objects in Excel
- Basically, inside one circle we draw two circles with an identical radius, and then the process repeats itself for the two smaller circles
- In this example, circles at deeper depths are darker
- To do this, we set the brightness at each level to be:

(TotalNumOfDepth – CurrentDepth) / TotalNumOfDepth

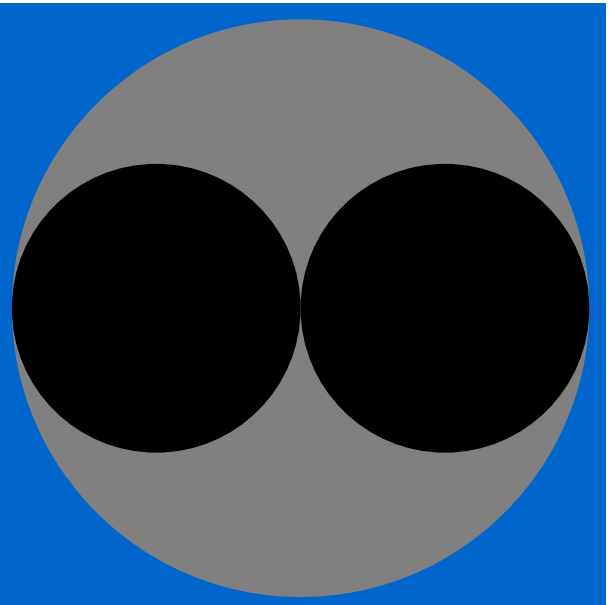
Recursive Circles Process



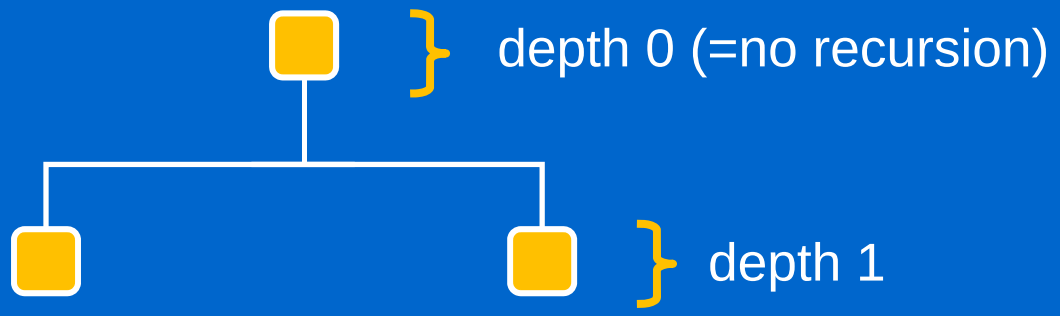
Depth 0 Depth 1 Depth 2 Depth 3 Depth 4

1. Start with a black circle
2. Set the gray colour of the circle. Determine the radius of the two smaller circles.
3. Repeat step 2 twice, to handle the left and right side

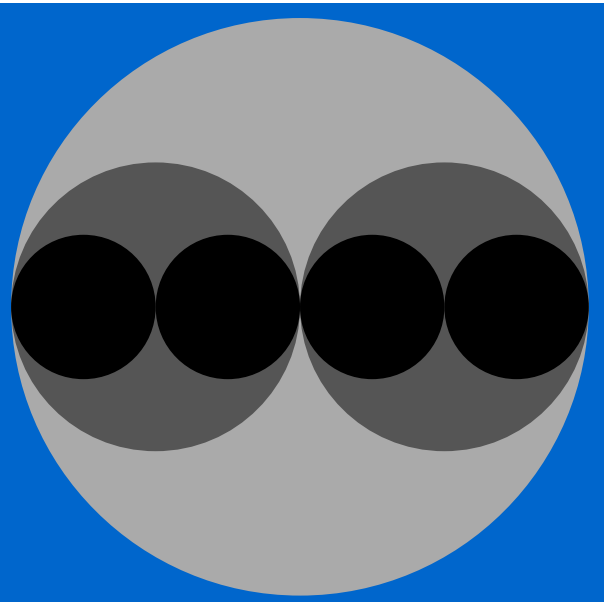
Recursive Depths



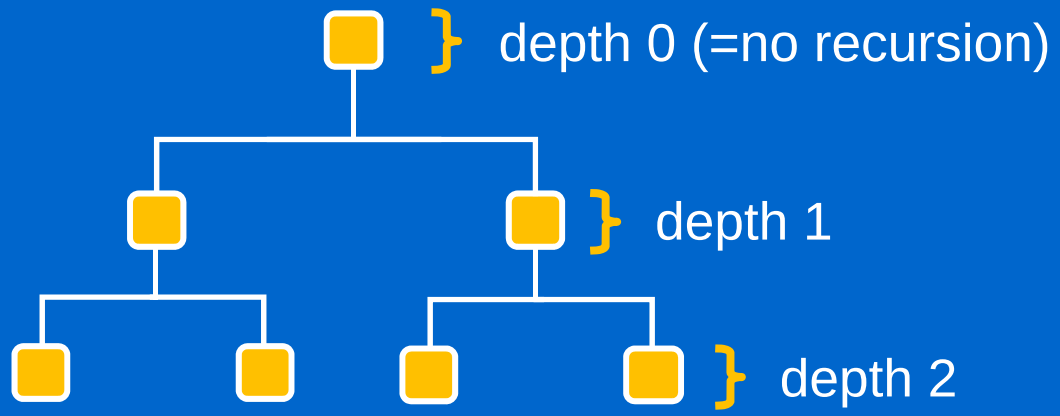
- For this example, when the maximum depth=1, this is what the depths look like:



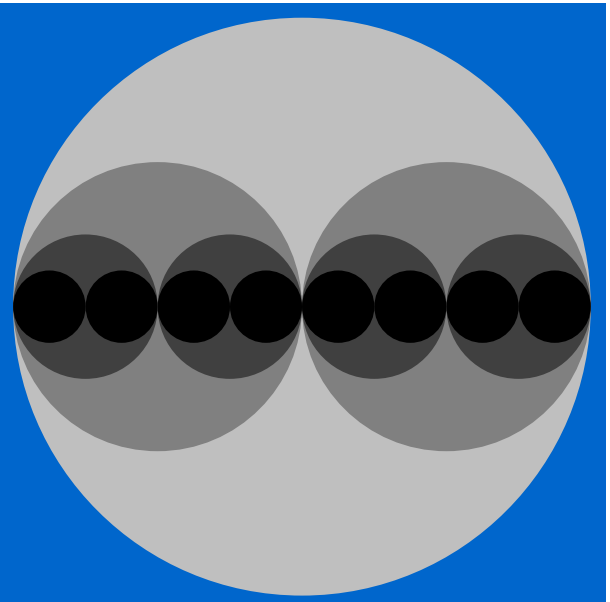
Recursive Depths



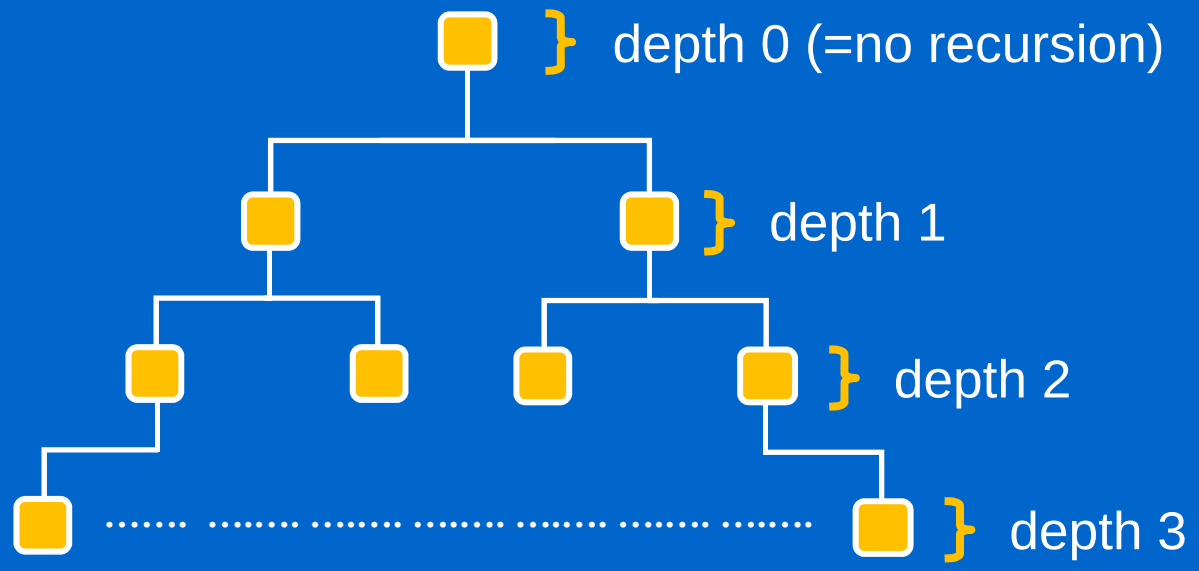
- For this example, when the maximum depth=2, this is what the depths look like:



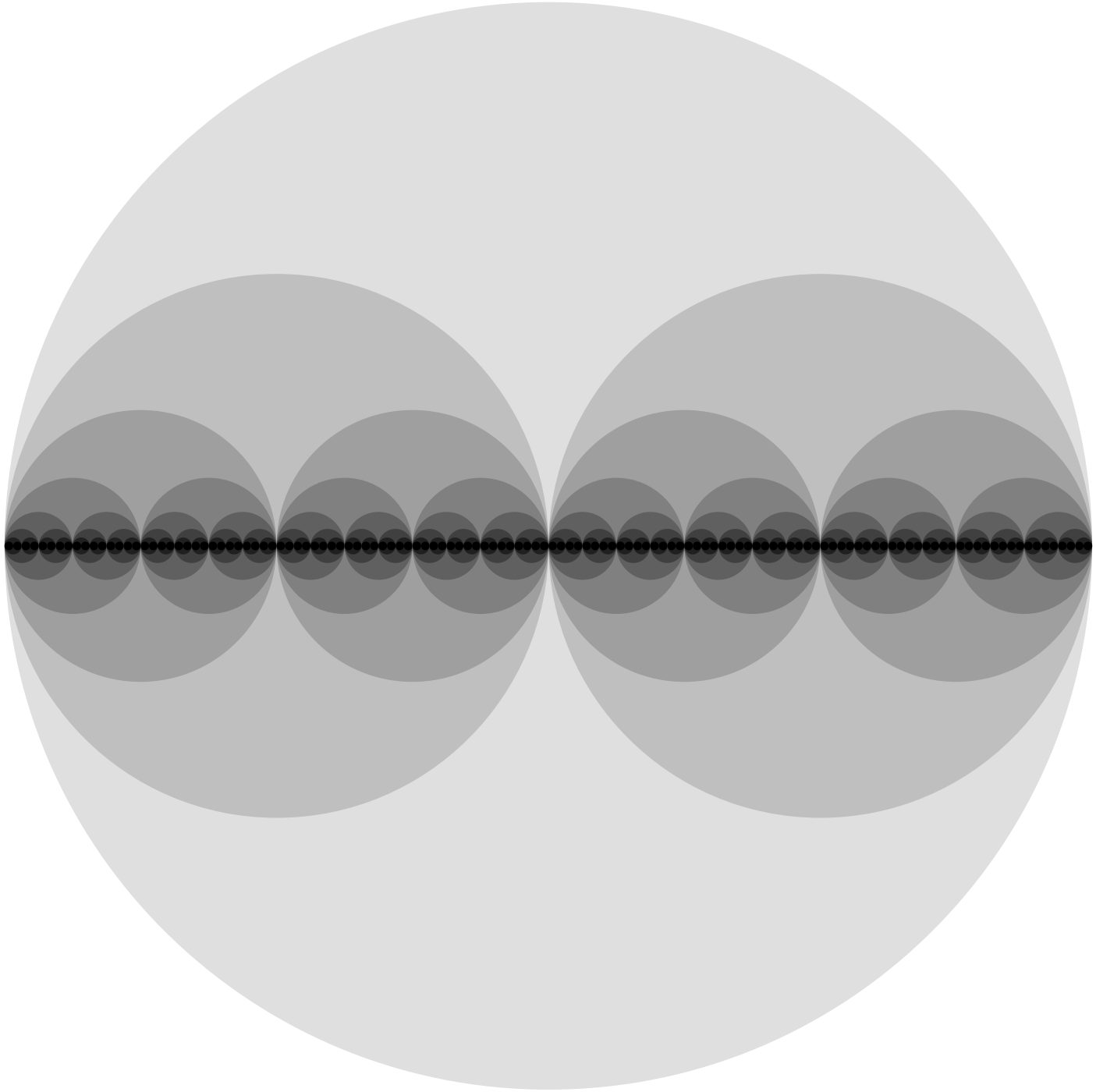
Recursive Depths



- For this example, when the maximum depth=3, this is what the depths look like:

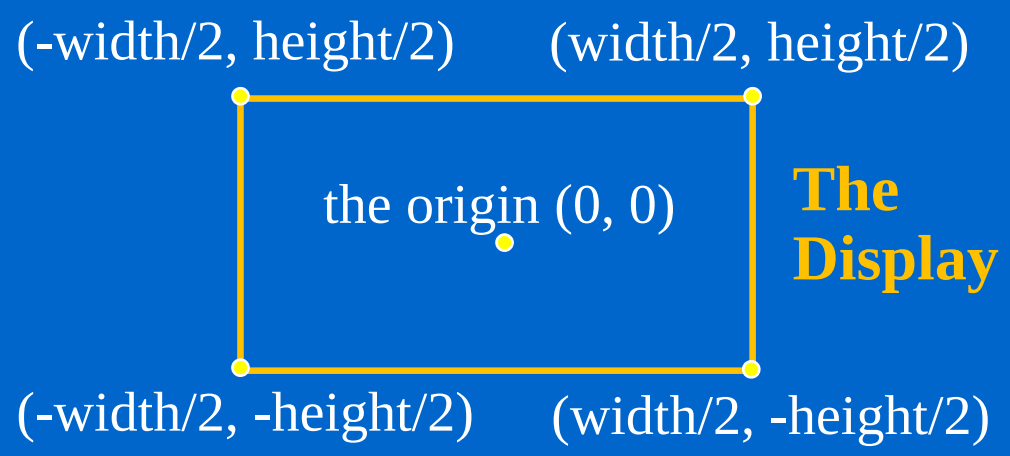


The recursive circles, after many recursive calls

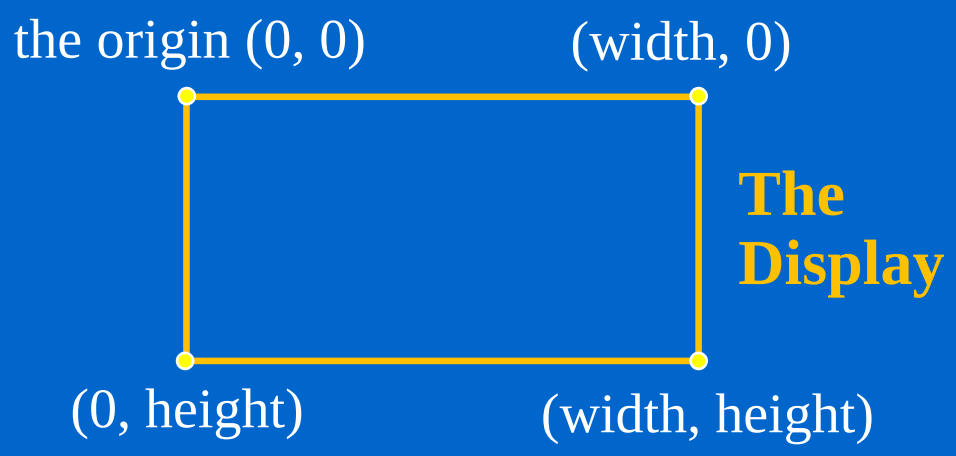


Quick Reminder of the Coordinate System

- The cartesian coordinate system:
  - You probably used this system when you learned Maths at school



- The VBA coordinate system:
  - For example, when you do some programming with VBA shape objects, all the x and y values are positive



Circle Recursive Code 1/4

- This slide shows the preparation code

```
Dim MaxDepth As Integer

Sub InputMaxDepth()
    Dim CenterX As Double, Radius As Double

    CenterX = 200
    Radius = 128

    MaxDepth = InputBox("Enter the maximum " & _
                   "depth for drawing the circles: ")

    DrawCircle CenterX, Radius, 0
End Sub
```

Start the recursion process with the biggest circle

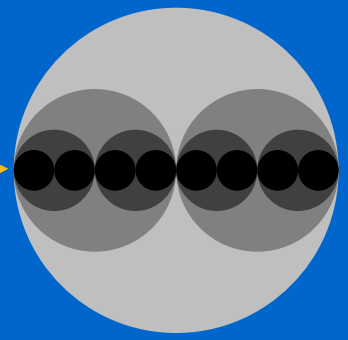
The Recursive Function

- The recursive function is shown on the next 3 slides:

DrawCircle(CenterX, Radius, Depth)

- *CenterX* is the x coordinate of the center of this circle
- *Radius* is the radius of the current circle to be drawn
- *Depth* is the current depth of the recursion

- In this example the value of Y is always the same so we don't need to pass it/change it, we can simply use a constant value



Circle Recursive Code 2/4

- The recursive function

```
Sub DrawCircle(CenterX As Double, Radius As Double, _
               Depth As Integer)
    Dim GrayColour As Double
    Dim StartY As Double
    Dim Top As Double, Left As Double
    Dim Width As Double, Height As Double
    Dim CircleObj As Shape

    StartY = 150 ' Initial Y coordinate

    ' Calculate the brightness
    If MaxDepth > 0 Then
        ' CDBl() converts to double
        GrayColour = CDbl(MaxDepth - Depth) / _
                     (MaxDepth + 1)
    End If
```

Calculate and set the brightness of this circle

CDBl means ‘convert the number to a double’

Continued on the next slide

Circle Recursive Code 3/4

- The recursive function, cont.

```
Left = CenterX - Radius
Top = StartY + (2 ^ Depth * Radius) - Radius

Width = Radius * 2
Height = Radius * 2

Set CircleObj = ActiveSheet.Shapes.AddShape( _
                    msoShapeOval, Left, Top, _
                    Width, Height)
CircleObj.Line.Visible = msoFalse
CircleObj.Fill.ForeColor.RGB = RGB(255 * GrayColour, _
                                   255 * GrayColour, _
                                   255 * GrayColour)
```

Draw a filled circle

- In computers, a colour is comprised of three numbers: the red (R), green (G) and blue (B) components
- Each R, G and B number has the range 0-255
- You don't need to understand RGB in this course!

Continued on the next slide

Circle Recursive Code 4/4

- The recursive function, cont.

```
    If Depth < MaxDepth Then
        DrawCircle CenterX - Radius/2, _
                   Radius/2, Depth + 1
        DrawCircle CenterX + Radius/2, _
                   Radius/2, Depth + 1
    End If
End Sub
```

Run *DrawCircle* twice to handle the left and right areas