PJAC – Guião 3
Prototipagem de aplicações VBA/AutoCAD

Aplicação de desenho de um polígono, dados o nº de lados, ponto inicial, ângulo inicial e comprimento de lado:

```
Option Explicit
Public Sub Polygon()
Dim Sides As Integer
Dim Ang As Double
Dim Dist As Double
Dim side As Integer
Dim i As Integer
Dim vret As Variant
Dim pt(0 To 2) As Double
Dim p() As Double
Dim obj As AcadLWPolyline
ThisDrawing.Utility.InitializeUserInput 1 + 2 + 4
Sides = ThisDrawing.Utility.GetInteger(vbCrLf + "Lados: ")
If Sides < 3 Or Sides > 10 Then
    MsgBox "Erro no número de lados"
    End
End If
ReDim p(Sides * 2 + 1)
ThisDrawing.Utility.InitializeUserInput 1 + 2
vret = ThisDrawing.Utility.GetPoint(, "Ponto inicial: ")
pt(0) = vret(0)
pt(1) = vret(1)
p(0) = vret(0)
p(1) = vret(1)
ThisDrawing.Utility.InitializeUserInput 1 + 2 + 32
Ang = ThisDrawing.Utility.GetAngle(pt, "Angulo: ")
ThisDrawing.Utility.InitializeUserInput 1 + 2 + 4 + 64
Dist = ThisDrawing.Utility.GetDistance(pt, "Comprimento: ")
For side = 0 To Sides - 2
    i = 2 * side
    vret = ThisDrawing.Utility.PolarPoint(pt, Ang, Dist)
    pt(0) = vret(0)
    pt(1) = vret(1)
    p(i + 0) = vret(0)
    p(i + 1) = vret(1)
    Ang = Ang + dtr(360# / Sides)
Next
vret = ThisDrawing.Utility.PolarPoint(pt, Ang, Dist)
i = 2 * (Sides - 1)
p(i + 0) = vret(0)
p(i + 1) = vret(1)
p(i + 2) = p(0)
p(i + 3) = p(1)
Set obj = ThisDrawing.ModelSpace.AddLightWeightPolyline(p)
End Sub
Private Function dtr(Ang) As Double
dtr = Ang * 3.1415926535898 / 180
End Function
```

Notas sobre o código acima:

- *Variant* para ler dados de pontos e depois copiar para vectores de *Double*
- *ThisDrawing.Utility.InitializeUserInput* para condicionar a entrada de dados
- *ThisDrawing.Utility.GetXXX()* para ler dados da linha de comandos
- *ThisDrawing.Utility.PolarPoint()* para calcular pontos de forma relativa (@d<ang)
- *ThisDrawing.ModelSpace.AddXXX()* para desenhar geometria (2D e 3D)

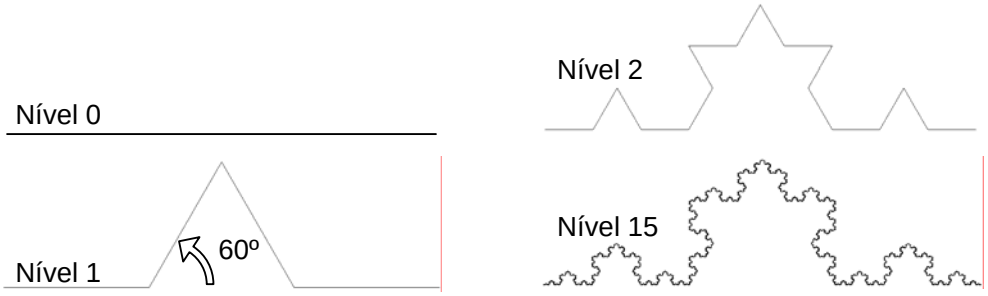
Para desenhar e manipular objectos 3D:

- Declarar variáveis de objectos como *Acad3DSolid*
- Criar com *obj1 = ThisDrawing.ModelSpace.AddXXX()*, etc
- Aplicar operadores CSG com *obj1.Boolean OP, obj2* em que *OP* pode ser *acUnion*, *acIntersection* ou *acSubtraction*, sendo o resultado aplicado a *obj1*
- No caso de extrusões ou revoluções, é necessário criar variáveis do tipo *AcadRegion*

Sugestões:

- No caso de querer dimensionar a janela, usar *ZoomAll*
- Em ciclos de processamento demorados, inserir *DoEvents* no meio do código de ciclo

Exercício proposto – Desenhar Fractais



Elaborar uma aplicação VBA que:

- Peça um ponto inicial, uma distância e o número de níveis no desenho do fractal
- Desenhe o fractal (da esquerda para direita) com segmentos de recta

Para este tipo de problemas, a abordagem recursiva é a mais simples. A ideia passa por definir uma subrotina **Fractal(*ponto*,*direcção*,*dimensão*,*nível*)** que deve considerar dois casos conforme o valor de ***nível***:

- Caso zero – desenhar o segmento de recta **Recta(*ponto*,*direcção*,*dimensão*)**
- Caso positivo – invocar **Fractal()** com argumentos (*ponto*,*direcção*,*dimensão*,*nível*) ajustados ao fractal em causa e ***nível*** decrementado uma unidade

A invocação inicial será feita com os parâmetros definidos pelo utilizador.

```
Option Explicit
Dim Calls As Integer
Dim LengthMin As Double
Public Sub main()
Const LengthRatio = 500#
Dim temp As Variant
Dim Length As Double
Dim Levels As Integer
Dim x As Double, y As Double
temp = ThisDrawing.Utility.GetPoint(, vbCrLf & "Ponto inicial: ")
x = temp(0)
y = temp(1)
Length = ThisDrawing.Utility.GetDistance(temp, vbCrLf & "Comprimento: ")
LengthMin = Length / LengthRatio
Levels = ThisDrawing.Utility.GetInteger(vbCrLf & "Niveis: ")
Calls = 0
Call Fractal(x, y, 0, Length, Levels)
ThisDrawing.Utility.Prompt vbCrLf & "Chamadas recursivas: " & Calls
End Sub
Private Sub Fractal(px As Double, py As Double, pdir As Double, plength As Double, plevel As Integer)
Dim pt1(0 To 2) As Double, pt2(0 To 2) As Double
Dim temp As Variant
???
Calls = Calls + 1
If Calls Mod 1000 = 0 Then
    DoEvents
    Regen acActiveViewport
End If
pt1(0) = px
pt1(1) = py
If plevel > 0 And plength > LengthMin Then
    ???
Else
    temp = ThisDrawing.Utility.PolarPoint(pt1, pdir, plength)
    pt2(0) = temp(0)
    pt2(1) = temp(1)
    ThisDrawing.ModelSpace.AddLine pt1, pt2
End If
End Sub
Private Function dtr(Ang) As Double
dtr = Ang * 3.1415926535898 / 180
End Function
```

Outros fractais: