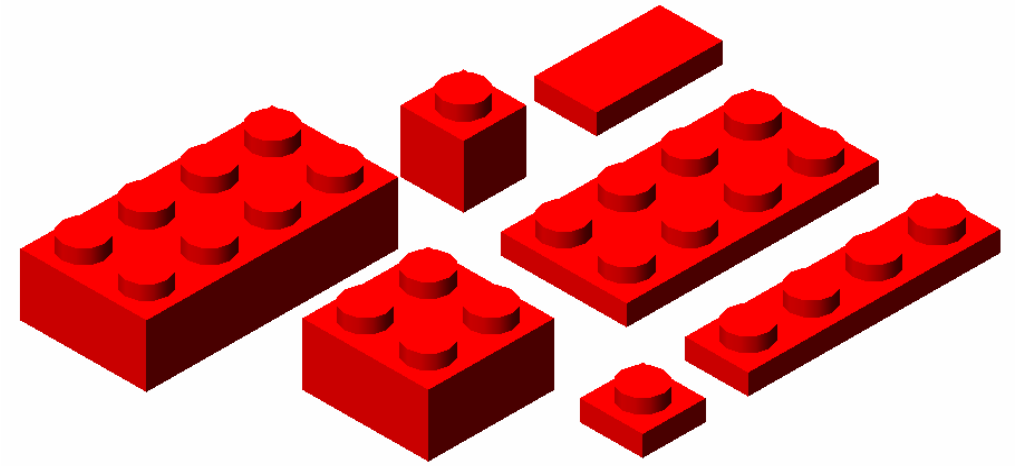
PJAC – Guião 4

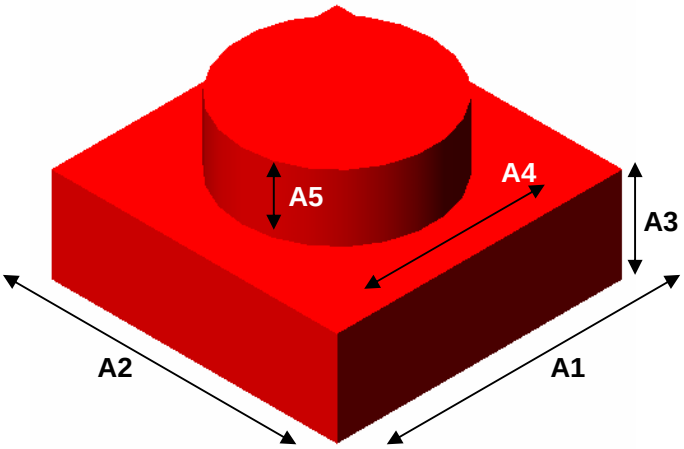
Prototipagem de aplicações VBA/AutoCAD

(Manual VBA: http://www.dei.isep.ipp.pt/pac/ftpdei/pjac/docs/VBA_AutoCAD/VBAdev.pdf)

Pretende-se desenvolver um modelador paramétrico de peças LEGO®. Numa versão inicial, serão apenas contemplados "tijolos" rectangulares de dimensão variável, lisos ou providos de encaixes cilíndricos e de várias cores. As primitivas de modelação são óbvias (caixa, cilindro e união).



Para efeitos de modelação, considere-se a dimensão fundamental *SIZE*. Todas as restantes podem ser referidas a *SIZE*, conforme se mostra na figura seguinte para a peça em causa:

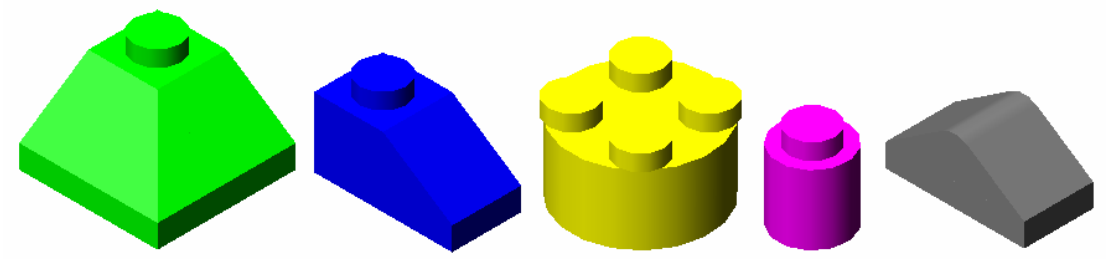


A1 = **k1** x *SIZE*; **A2** = **k2** x *SIZE*; **A3** = **k3** x *SIZE*; **A4** = 2/3 x *SIZE*; **A5** = 1/4 x *SIZE*
(nesta peça, **k1** = 1; **k2** = 1; **k3** = 1/3)

Exercício – desenvolver um modelador paramétrico de LEGO em VBA. Dados de entrada:

- Ponto de ancoragem (por exemplo, usar o canto do objecto com menores coordenadas X,Y,Z)
- Dimensões relativas **k1**,**k2**,**k3** – valores inteiros positivos, **k3** pode ainda assumir o valor 1/3, usando uma keyword (ver exemplo do método InitializeUserInput)
- Existência ou não de encaixes cilíndricos no topo
- Cor da peça (um número entre 1 e 255)

Em seguida, extenda o modelador de forma a desenhar outros tipos de peças. Exemplos:



Algumas Notas sobre Modelação 3D em VBA

Os objectos 3D devem ser declarados como ***Acad3Dsolid***.
O desenho e manipulação de primitivas é simples:

```
Dim obj1 As Acad3Dsolid
Dim pt1(0 To 2) As Double, pt2(0 to 2) As Double
...
Set obj1 = ThisDrawing.ModelSpace.AddBox(pt1, 10, 10, 8)
Set obj2 = obj1.Copy
obj1.Color = acRed
obj1.Move pt1, pt2
obj1.Rotate3D pt1, pt2, 3.1415926
obj2.Color = acGreen
obj1.Boolean acSubtraction, obj2
obj1.Update
```

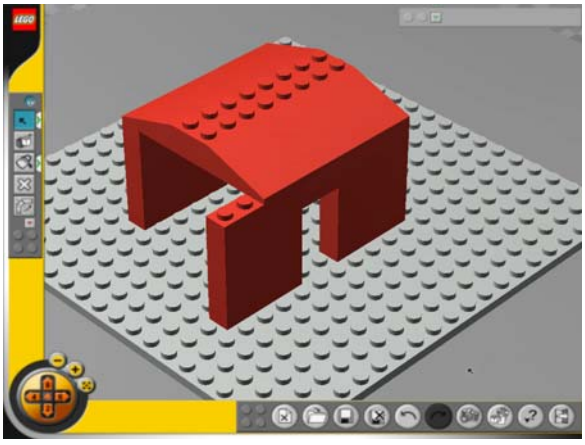
No excerto acima, o objecto *obj2* é criado como cópia de *obj1*.
Depois, *obj1* é modificado (cor, localização e orientação) e *obj2* muda de cor.
Em seguida, é efectuada a subtracção CSG de *obj2* a *obj1*, ficando o resultado em *obj1*.

Caso pretendam usar subrotinas para encapsular objectos mais complexos, podem usar a seguinte aproximação:

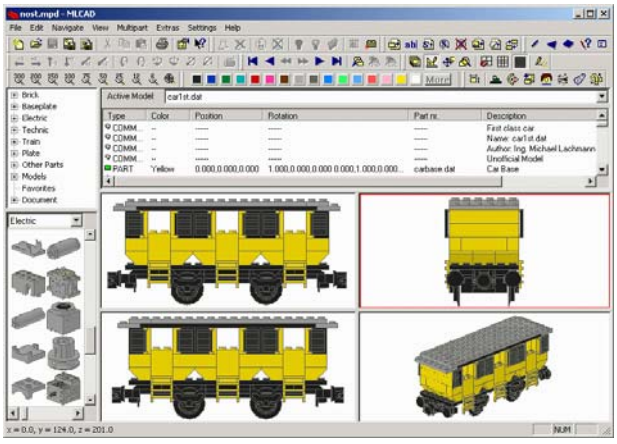
```
Dim block As Acad3Dsolid
...
Private Sub CreateBlock(x As Double, y As Double, z As Double, ByRef block As Acad3DSolid)
       Dim pt(0 To 2) As Double
       pt(0) = x / 2
       pt(1) = y / 2
       pt(2) = -z / 2
       Set block = ThisDrawing.ModelSpace.AddBox(pt, x, y, z)
       block.Color = acRed
End Sub
...
Call CreateBlock(bsx, bsy, bsz, block)
block.Color = acBlue
```

Notem o uso de passagem de parâmetros através de *ByRef* por forma a devolver, para fora da subrotina, o objecto criado.

Aplicações de Modelação Paramétrica de LEGO® disponíveis na Internet



LEGO® Digital Designer



Mike's LEGO® CAD

LEGO® Digital Designer: http://www.lego.com/eng/create/digitaldesigner/default.asp
Mike's LEGO® CAD: http://www.lm-software.com/mlcad/

Solução do Problema do Guião 4

```
Option Explicit
'parametrizacao das dimensoes
Const SIZE = 10
Const D1 = SIZE
Const D2 = SIZE
Const D3 = SIZE
Const D4 = SIZE / 3#
Const D5 = SIZE / 4#

Private Sub Brick1(x As Double, y As Double, z As Double, color As Integer, _
        dim1 As Integer, dim2 As Integer, dim3 As Integer)
    Dim d3real As Double
    Dim i As Integer, j As Integer, k As Integer
    Dim pn(0 To 2) As Double
    Dim pc(0 To 2) As Double
    Dim obj1 As Acad3DSolid
    Dim obj2 As Acad3DSolid
    Dim group As AcadGroup
    Dim list(0 To 0) As AcadEntity
    'esta rotina gera um tijolo paralelipipedico formado de caixa, cilindros e pontos
    'os pontos servem apenas para se poder fazer SNAP de acordo com os encaixes
    'caixa, cilindros e pontos sao agrupados com nome escolhido aleatoriamente
    'o objecto final (grupo) e' formado pela caixa, cilindros e pontos
    If dim1 < 1 Or dim2 < 1 Or dim3 < 0 Then
      Exit Sub
    End If
    If dim3 = 0 Then
        d3real = D3 / 3#
        dim3 = 1
    Else
        d3real = D3
    End If
    Set group = ThisDrawing.Groups.Add(Str(-Int(10000000 * Math.Rnd)))
    pc(0) = x + dim1 * D1 / 2#
    pc(1) = y + dim2 * D2 / 2#
    pc(2) = z + dim3 * d3real / 2#
    Set obj1 = ThisDrawing.ModelSpace.AddBox(pc, dim1 * D1, dim2 * D2, dim3 * d3real)
    For i = 0 To dim1 - 1
        pn(0) = x + i * D1
        pc(0) = pn(0) + D1 / 2#
        For j = 0 To dim2 - 1
            pn(1) = y + j * D2
            pc(1) = pn(1) + D2 / 2#
            k = dim3 - 1
            pn(2) = z + k * d3real + d3real
            Set list(0) = ThisDrawing.ModelSpace.AddPoint(pn)
            group.AppendItems list
            pc(2) = pn(2) + D5 / 2#
            Set obj2 = ThisDrawing.ModelSpace.AddCylinder(pc, D4, D5)
            obj1.Boolean acUnion, obj2
            pn(1) = pn(1) + D2
            Set list(0) = ThisDrawing.ModelSpace.AddPoint(pn)
            group.AppendItems list
            DoEvents
        Next
    Next
    pn(0) = pn(0) + D1
    pn(1) = y
    For i = 0 To dim2
        Set list(0) = ThisDrawing.ModelSpace.AddPoint(pn)
        group.AppendItems list
        pn(1) = pn(1) + D2
    Next
    obj1.color = color
    Set list(0) = obj1
    group.AppendItems list
End Sub

Private Function inside(x1 As Double, y1 As Double, _
    x As Double, y As Double, r As Double) As Boolean
    inside = ((x1 - x) * (x1 - x) + (y1 - y) * (y1 - y) < r * r)
End Function

Private Sub Brick2(x As Double, y As Double, z As Double, color As Integer, _
    dim1 As Integer, dim2 As Integer, dim3 As Integer)
    Dim d3real As Double
    Dim i As Integer, j As Integer, k As Integer
    Dim pn(0 To 2) As Double
    Dim pc(0 To 2) As Double
    Dim obj1 As Acad3DSolid
    Dim obj2 As Acad3DSolid
    Dim group As AcadGroup
    Dim list(0 To 0) As AcadEntity
    Dim cx As Double, cy As Double, cr As Double
    'esta rotina gera um tijolo cilindrico formado de cilindros e pontos
    'os pontos servem apenas para se poder fazer SNAP de acordo com os encaixes
    'cilindros e pontos sao agrupados com nome escolhido aleatoriamente
    'o objecto final (grupo) e' formado pelos cilindros e pontos
    If dim1 < 1 Or dim2 < 1 Or dim1 <> dim2 Or dim3 < 0 Then
      Exit Sub
    End If
    If dim3 = 0 Then
        d3real = D3 / 3#
        dim3 = 1
    Else
        d3real = D3
    End If
    Set group = ThisDrawing.Groups.Add(Str(-Int(10000000 * Math.Rnd)))
    pc(0) = x + dim1 * D1 / 2#
    pc(1) = y + dim2 * D2 / 2#
    pc(2) = z + dim3 * d3real / 2#
    cx = pc(0)
    cy = pc(1)
    cr = dim1 * D1 / 2#
    Set obj1 = ThisDrawing.ModelSpace.AddCylinder(pc, dim1 * D1 / 2#, dim3 * d3real)
    For i = 0 To dim1 - 1
        pn(0) = x + i * D1
        pc(0) = pn(0) + D1 / 2#
        For j = 0 To dim2 - 1
            pn(1) = y + j * D2
            pc(1) = pn(1) + D2 / 2#
            k = dim3 - 1
            pn(2) = z + k * d3real + d3real
            If inside(pn(0), pn(1), cx, cy, cr) Then
                Set list(0) = ThisDrawing.ModelSpace.AddPoint(pn)
                group.AppendItems list
            End If
            pc(2) = pn(2) + D5 / 2#
            If inside(pc(0), pc(1), cx, cy, cr) Then
                Set obj2 = ThisDrawing.ModelSpace.AddCylinder(pc, D4, D5)
                obj1.Boolean acUnion, obj2
            End If
            pn(1) = pn(1) + D2
            If inside(pn(0), pn(1), cx, cy, cr) Then
                Set list(0) = ThisDrawing.ModelSpace.AddPoint(pn)
                group.AppendItems list
            End If
            DoEvents
        Next
    Next
    pn(0) = pn(0) + D1
    pn(1) = y
```

Solução do Problema do Guião 4

```
    For i = 0 To dim2
        If inside(pn(0), pn(1), cx, cy, cr) Then
            Set list(0) = ThisDrawing.ModelSpace.AddPoint(pn)
            group.AppendItems list
        End If
        pn(1) = pn(1) + D2
    Next
    obj1.color = color
    Set list(0) = obj1
    group.AppendItems list
End Sub

Private Sub InputData(kind As Integer, x As Double, y As Double, z As Double, _
    color As Integer, dim1 As Integer, dim2 As Integer, dim3 As Integer)
    Dim temp As Variant
    Dim strTmp As String
    Dim Keywords As String

    'esta rotina serve para ler os dados de entrada
    Keywords = "Parelelepipedo Cilindro"
    ThisDrawing.Utility.InitializeUserInput 1, Keywords
    strTmp = ThisDrawing.Utility.GetKeyword(vbCrLf & "Tipo [" & Keywords & "]: ")
    Select Case (strTmp)
        Case "Parelelepipedo"
            kind = 1
        Case "Cilindro"
            kind = 2
    End Select
    ThisDrawing.Utility.InitializeUserInput 1
    temp = ThisDrawing.Utility.GetPoint(, vbCrLf & "Ponto inicial: ")
    x = temp(0)
    y = temp(1)
    z = temp(2)
    Keywords = "byLayer Vermelho vErde Azul AMarelo Ciano Magenta Branco"
    ThisDrawing.Utility.InitializeUserInput 1 + 2 + 4, Keywords
    On Error Resume Next  ' ativar o controlo de erros para continuar na linha seguinte
    color = ThisDrawing.Utility.GetInteger(vbCrLf & "Cor[1-255 " & Keywords & "]: ")

    If Err Then 'provavelmente foi keyword
         If StrComp(Err.Description, "User input is a keyword", 1) = 0 Then
         ' foi inserida uma keyword
            strTmp = ThisDrawing.Utility.GetInput  'lê a keyowrd do buffer
            Select Case (strTmp)
                Case "byLayer"
                    color = acByLayer
                Case "Vermelho"
                    color = acRed
                Case "vErde"
                    color = acGreen
                Case "Azul"
                    color = acBlue
                Case "AMarelo"
                    color = acYellow
                Case "Ciano"
                    color = acCyan
                Case "Magenta"
                    color = acMagenta
                Case "Branco"
                    color = acWhite
            End Select

         Else
             MsgBox "Erro desconhecido" & Err.Description
             Stop
         End If
        Err.Clear
    Else
        ' Display point coordinates
        If color > 256 Then
            color = acByLayer
        End If
    End If

    On Error GoTo 0  ' reativar o controlo de erros pelo VBA
    ThisDrawing.Utility.InitializeUserInput 1 + 2 + 4
    dim1 = ThisDrawing.Utility.GetInteger(vbCrLf & "Dim1[>0]: ")
    If dim1 > 256 Then
        dim1 = 255
    End If
    If kind = 2 Then
        dim2 = dim1
    Else
        ThisDrawing.Utility.InitializeUserInput 1 + 2 + 4
        dim2 = ThisDrawing.Utility.GetInteger(vbCrLf & "Dim2[>0]: ")
        If dim2 > 256 Then
            dim2 = 255
        End If
    End If
    Keywords = "1/3"
    ThisDrawing.Utility.InitializeUserInput 1 + 2 + 4, Keywords
    On Error Resume Next  ' ativar o controlo de erros para continuar na linha seguinte
    dim3 = ThisDrawing.Utility.GetInteger(vbCrLf & "Dim3[>0 " & Keywords & "]: ")
    If Err Then 'provavelmente foi keyword
         If StrComp(Err.Description, "User input is a keyword", 1) = 0 Then
         ' foi inserida uma keyword e só pode ser 1/3
            strTmp = ThisDrawing.Utility.GetInput  'lê a keyowrd do buffer
            dim3 = 0
         Else
             MsgBox "Erro desconhecido" & Err.Description
             Stop
         End If
        Err.Clear
    Else
        ' Display point coordinates
        If dim3 > 256 Then
            dim3 = 255
        End If
    End If

    On Error GoTo 0  ' reativar o controlo de erros pelo VBA
End Sub

Public Sub Main()
    Dim kind As Integer
    Dim x As Double, y As Double, z As Double
    Dim dim1 As Integer, dim2 As Integer, dim3 As Integer
    Dim color As Integer
    'esta rotina executa a aplicacao
    'o gerador de numeros aleatorios e' inicializado com um valor sempre diferente
    'antes de desenhar, o modo SNAP e' posto apenas em Endpoint e Node
    Math.Randomize
    ThisDrawing.SendCommand ("-osnap none,end,node ")
    On Error GoTo label1
    Call InputData(kind, x, y, z, color, dim1, dim2, dim3)
    If kind = 1 Then
        Call Brick1(x, y, z, color, dim1, dim2, dim3)
    ElseIf kind = 2 Then
        Call Brick2(x, y, z, color, dim1, dim2, dim3)
    End If
label1:
    Err.Clear
End Sub
```