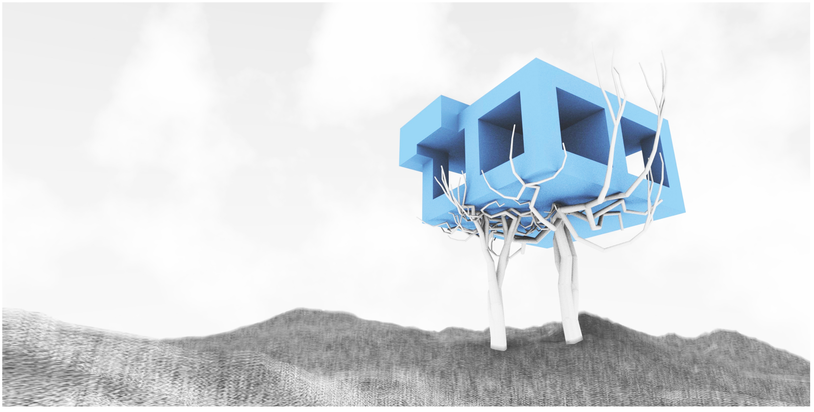


PrasekaTree - skript pro vytváření zeleně

PrasekaTree je skript pro vytváření zeleně do renderů tam, kde je třeba, aby se zelenina neprotínala s geometrií. Větvení stromu je obstarán rekurzí, která ve své definici volá sama sebe s n-1 "podlažím" (číslo udávající počet iterací - a zamezující nekonečný běh skriptu). V každém dalším "levelu" jsou tedy vytvořeny dvě větve, obsahující dále určity počet segmentů (stejně jako většina věcí nastavitelný pomocí grasshopper-Slidu). Při vytváření segmentů se pro každý další vytvoří list informací, obsahující informace o geometrii a umístění segmentu, v jakém podlaží, větvi a její části se objekt nachází a také id jeho rodiče, který je nezbytný pro vytváření variant při kolizi větve s geometrií. V případě protnutí větve s jinými objekty je vytvořen nový segment svírající se svým rodičem úhel maximálně 90° (přehlédnutí vstupní hodnoty) a znovu vyzkoušena kolize. Počet pokusů je limitován pro případ nemožnosti vytvořit neprotínající se větev (větev nevytvořena a nepokračuje ve vývoji). Až následně je segment zapsán do seznamu, obsahující informace o všech větvích, ze kterého se na konci skriptu vypisují zobrazované objekty. Chování větví se řídí vstupními proměnnými jako počet podlaží, počet rozvětvení v uzlu, počet segmentů na jedno podlaží, poměr zkrácení a zmenšení poloměru větve oproti rodičovskému segmentu, velikosti "tahu" za kladnými hodnotami osy z. Průsečíky s ostatní geometrií jsou pro rychlost vykonávání skriptu posuzovány na základě průniku osy segmentu s BRep objekty.

Základní rekurzivní "tree" funkce

```
Public Function tree(level As Integer, branches As Integer, line As online, segments As list(Of Object), parent As Integer) As Integer

    Dim i, i2 As Integer

    If level > 0 Then #### Větvení pokud podlaží je větší nez 0

      For i = 0 To (branches - 1) #### Větve, počet větví
        Dim tempLine As New online
        Dim tempParent As Integer = parent
        Dim intersect As Boolean
        templine.Create(line.from, line.To)

        For i2 = 0 To (inputSegmentCount - 1) #### Segmenty, do počtu segmentů
          Dim parentLine As New online
          parentLine.create(templine.from, templine.To)
          templine = createSegment(templine, level, i2, inputAlfa)
          Dim segments As New list(Of Object)
          segmentCreate(segment, templine, level, i, i2, tempParent) #### volání funkce na geometrické vytváření nové osy segmentu
          intersect = segmentIntersectionGeometry(segment, inputGeometry, segmentGetParent(segment, segments)) #### průsečíky s geometrií
          If intersect = False Then #### při vyčerpání pokusů na vytvoření nového segmentu se zde větvení ukončí
            print("segment" & str(segments.count - 1) & " " & str(segment(2)) & str(segment(3)) & str(segment(4)) & " NOT CONTINUED")
            i2 = inputSegmentCount
          Else
            segmentWrite(segment, segments) #### zapsání validního segmentu do seznamu
            Dim tempSegNum As Integer = segments.count - 1
            print("segment" & str(tempSegNum) & " " & str(segments(tempSegNum)(2)) & str(segments(tempSegNum)(3)) & str(segments(tempSegNum)(4)) & "; parent: " & \
            str(segments(tempSegNum)(6)))
            tempParent = segments.count - 1
            templine.Create(segment(0), segment(1))
          End If
        Next i2

        If intersect = True Then
          Call tree(level - 1, branches, templine, segments, (segments.count - 1)) #### zavolání "sama sebe"
        End If
      Next i

    End If

End Function
```

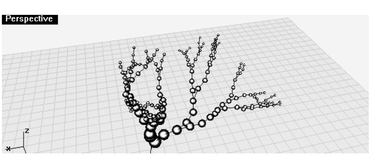
Konstrukce nového segmentu z vstupních hodnot a hodnot převzatých z rodičovského segmentu

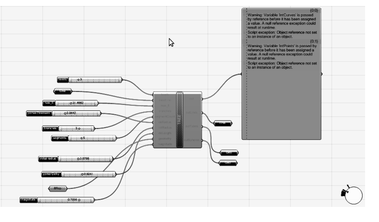
```
Public Function createSegment(line As online, level As Integer, segmentNumber As Integer, alfa As Double) As online

    Dim startPoint, endPoint As New On3dPoint
    Dim plane As New OnPlane
    Dim circle As New OnCircle
    Dim sphere As New OnSphere
    Dim vector As New On3dVector
    Dim length As Double

    alfa = alfa * pi * 2 / 360
    startPoint = line.To #### začáteční bod segmentu je na koncovém bodu rodiče
    vector = line.To - line.from #### směr je převzat ze směru rodiče, později se změní
    endPoint = startPoint + vector
    length = inputInitLength * pow(inputDeLength, getMidPow(level, segmentNumber)) #### volání funkce na výpočet délky

    plane.CreateFromNormal(endPoint, vector) #### vytvoření kružnice o náhodném poloměru, která definuje úhlovou odchylku nového segmentu
    circle.Create(plane, endPoint, length * tan(alfa) * random.NextDouble())
    endPoint = circle.PointAt(random.NextDouble() * pi * 2) #### bod na kružnici daný jejím náhodným parametrem,
    endPoint.z = endPoint.z + inputMagnitude * length #### ten se stává koncovým bodem nového vektoru
    vector = endPoint - startPoint
    vector.Unitize()
    vector = vector * length * (random.NextDouble() / 2 + 1 / 2) #### aplikace vypočtené délky na vektor, zahrnutí náhodné délky
    endPoint = startPoint + vector
    line.create(startPoint, endPoint)

    Return line #### návratová hodnota, typu onLine

  End Function
```

Průniky s ostatními objekty ve scéně

```
Public Function segmentIntersectionGeometry(ByRef segment As list(Of Object), geomBreps As list(Of onbrep), parent As Object) As Boolean

    Dim i, i2 As Integer
    Dim Intersects As Boolean
    Dim level As Integer = segment(2)
    Dim branch As Integer = segment(3)
    Dim segmentCount As Integer = segment(4)
    Dim parentID As Integer = segment(6)
    Dim tryNum As Integer = 50

    For i = 0 To (geomBreps.count - 1) #### Smyčka pro všechny objekty ve scéně
      For i2 = 0 To tryNum #### jednotlivé varianty pro vyhnutí se
        Dim intPoints As On3dPointArray
        Dim intCurves() As OnCurve
        Dim tempLineIntCrv As New OnLineCurve
        tempLineIntCrv.m_line.Create(segment(0), segment(1))
        Rhutil.RhinoCurveBrepIntersect(tempLineIntCrv, geomBreps(i), 0, intCurves, intPoints) #### rhinoutil intersect funkce
        If intPoints.Count <> 0 Then #### případ průniku
          print("inside")
          Dim templine As New online
          templine.Create(parent(0), parent(1))
          templine = createSegment(templine, parent(2), parent(4), 80)
          segmentCreate(segment, templine, level, branch, segmentCount, parentID)
          Intersects = True
        Else #### případ nulového průniku
          i2 = tryNum
          intersects = False
        End If
      Next i2
    Next i

    If intersects = False Then
      Return True
    Else
      Return False
    End If

End Function
```

Tělo skriptu zde se nic podstatného neděje, kromě zavolání funkce tree na začátku a výpisu vytvořených objektů na konci





```
Dim segments As New List(Of Object) #### hlavní seznam segmentů

inputMagnitude = magnitude
inputLevels = levels
inputAlfa = max_fi
inputInitLength = baseline.Length
inputDeLength = delength
inputInitRadius = initRadius
inputDeradius = deradius
inputSegmentCount = segmentCount
inputGeometry = geometry

Dim initSeg As New List(Of Object)
segmentCreate(initSeg, baseLine, inputLevels + 1, 0, inputSegmentCount - 1, -1)
segments.add(initSeg)
tree(levels, branches, baseline, segments, 0)


'SPHERES AND LINES PRINTING

Dim lines As New List(Of online)
Dim spheres As New List(Of onSphere)
Dim tubes As New List(Of onNurbsSurface)

Dim i As Integer
For i = 0 To (segments.Count - 1)
  lines.add(segmentGetLine(segments(i))) #### vypisování os segmentů
  tubes.add(segmentGetCone(segments(i))) #### výpis válců segmentů
  spheres.add(segmentGetSphere(segments(i))) #### výpis kloubů segmentů
Next i

print("SEGMENTS: " & str(lines.Count))
outLines = lines
outSpheres = spheres
outTubes = tubes
```

Zadek skriptu (funkce na vypisování geometrií ze segmentů a ostatní definice)

```
Public Dim random As New random
Public Dim inputLevels As Integer
Public Dim inputAlfa As Double
Public Dim inputDeradius As Double
Public Dim inputInitLength As Double
Public Dim inputDeLength As Double
Public Dim inputInitRadius As Double
Public Dim inputSegmentCount As Integer
Public Dim inputGeometry As New list(Of onbrep)
Public Dim inputMagnitude As Double

'CREATION OF SEGMENT OBJECT
Public Function segmentCreate(ByRef segment As list(Of Object), ByVal line As onLine, level As Integer, branch
As Integer, segmentnumber As Integer, parent As Integer) As Boolean
    segment.clear()
    Dim A As New On3dPoint
    Dim B As New On3dPoint
    A.X = line.from.x
    A.Y = line.from.Y
    A.Z = line.from.Z
    B.X = line.To.x
    B.Y = line.To.Y
    B.Z = line.To.Z
    'ITEM 0: STARTPOINT
    segment.add(A)
    'ITEM 1: ENDPOINT
    segment.add(B)
    'ITEM 2: LEVEL
    segment.add(level)
    'ITEM 3: BRANCH
    segment.add(branch)
    'ITEM 4: SEGMENTNUMBER
    segment.add(segmentNumber)

    Dim radius As Double
    radius = inputInitRadius * pow(inputDeRadius, getMidPow(level, segmentNumber))
    'ITEM 5: ENDPOINTSPHERE RADIUS
    segment.add(radius)

    ' ITEM 6: PARENT
    segment.add(parent)

End Function

'WRITING A SEGMENT OBJECT TO A LIST
Public Function segmentWrite(ByVal segment As Object, ByRef  list As list(Of Object)) As Integer
    list.add(segment)
End Function

'PRINTING COORDINATES OF SEGMENT POINTS
Public Function segmentPrint(ByRef segment As list(Of Object)) As Boolean
    print("A: " & str(round(segment(0).x, 3)) & ", " & str(round(segment(0).y)) & ", " & str(round(segment
(0).z)))
    print("B: " & str(round(segment(1).x)) & ", " & str(round(segment(1).y)) & ", " & str(round(segment(1).z)))
End Function
```

```
'RETURNS A LINE FROM SEGMENT STARTPOINT TO ENDPOINT
Public Function segmentGetLine(segment As list(Of Object)) As online
    Dim l As New OnLine
    l.Create(segment(0), segment(1))
    Return l
End Function

  'RETURNS SPHERE ON ENDPOINT
  Public Function segmentGetSphere(segment As list(Of Object)) As onsphere
    Dim sphere As New OnSphere
    sphere.Create(segment(1), segment(5))
    Return sphere
  End Function

  'GET SEGMENT CYLINDER
  Public Function segmentGetCone(segment As list(Of Object)) As onNurbsSurface
    Dim tube As New OnNurbsSurface
    Dim brepTube As New onbrep
    Dim plane As New onplane
    Dim planeNormal As New On3dVector
    Dim circle As New oncircle
    Dim circle2 As New oncircle
    planeNormal = segment(1) - segment(0)
    plane.CreateFromNormal(segment(0), planeNormal)
    circle.Create(plane, segment(0), segment(5) / inputDeradius)
    circle2.Create(plane, segment(1), segment(5))
    Dim circleCurve As New OnNurbsCurve
    Dim circleCurve2 As New OnNurbsCurve
    circle.GetNurbForm(circleCurve)
    circle2.GetNurbForm(circleCurve2)
    tube.CreateRuledSurface(circleCurve, circleCurve2)
    Return tube
  End Function

  'GET PONRUMBER FROM LEVEL AND SEGMENT NUMBER
  Public Function getMidPow(level As Integer, segmentnumber As Integer) As Double
    Dim a As Integer
    a = inputLevels * inputSegmentCount - ((level - 1) * inputSegmentCount + inputSegmentCount - segmentnumber)
    Return a
  End Function

  'GET PARENT OF SEGMENT
  Public Function segmentGetParent(segment As list(Of Object), segments As list(Of Object)) As Object
    Dim parentID As Integer
    parentID = segment(6)
    If parentID <> -1 Then
      Return segments(parentID)
    Else
      Return "null"
    End If
  End Function
```

Tomáš Feistner, 3. semestr, CAD - scripting