

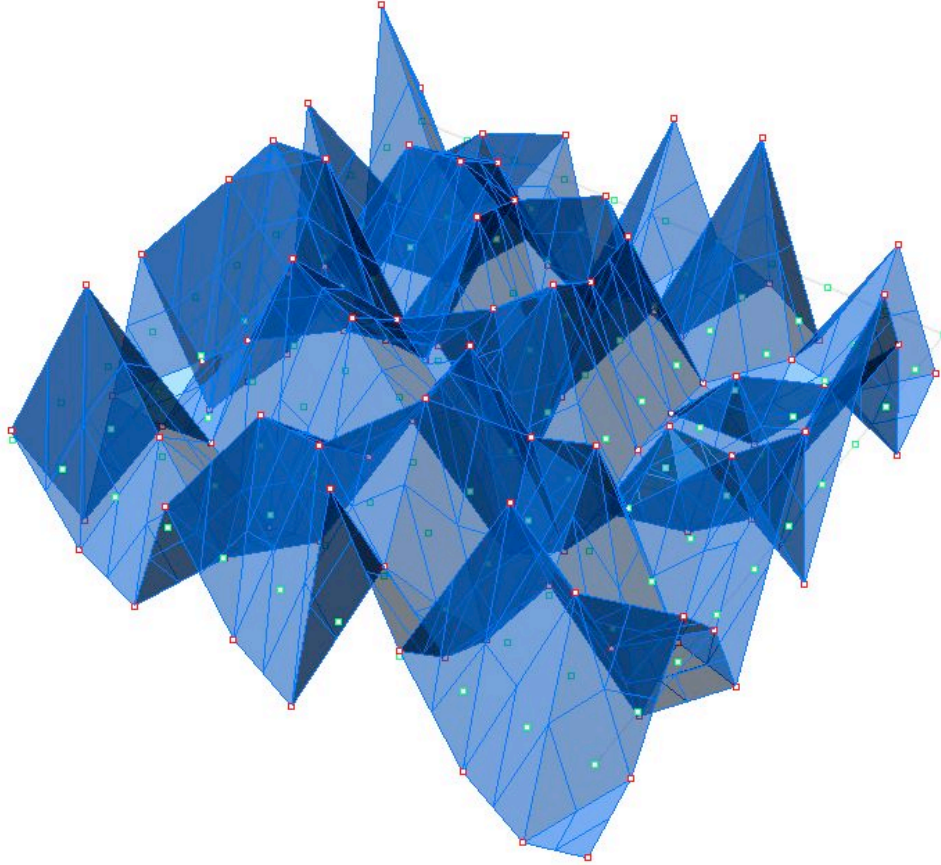
Parametric Surface Tutorial

Programming a Simple Parametric Tool in RhinoScript

Daniel Cardoso Llach

Ph.D. Student, Design and Computation / Design Lab, at MIT

March 10 2008, updated January 2009



Description

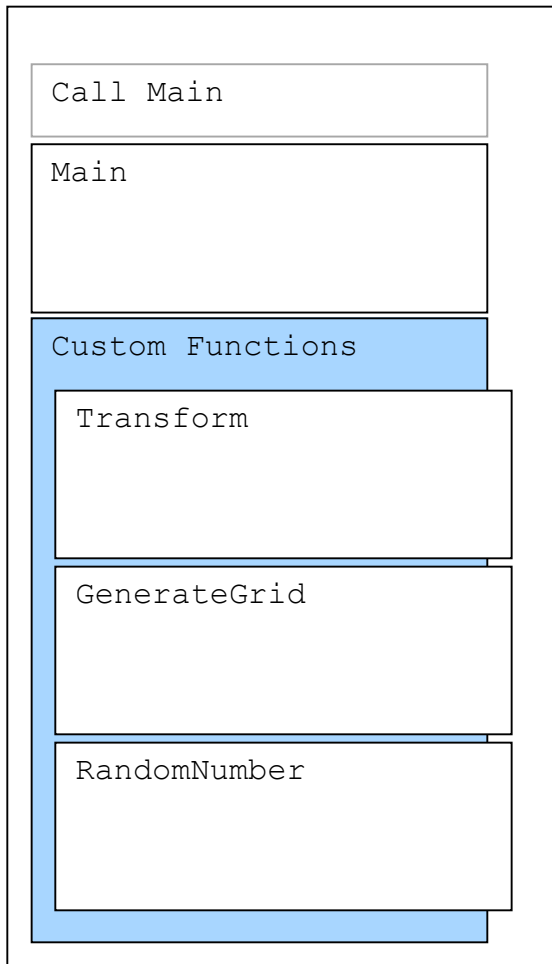
In this exercise we will create a parametric tool that is able to affect a planar grid of points in accordance to user-input.

The exercise is designed to bring up the basic logic behind the implementation and of parametric systems. It also provides an overview of some basic to intermediate scripting skills in *RhinoScript*. The focus, however, will be on the logical structure of the tool, rather than in the specific syntax of the language.

Goal

To create a parametric tool that asks input from the user to define the X and Y dimensions of a grid, and a type of transformation to be operated on it. The user can also specify an 'intensity multiplier' that changes the degree to which the planar grid is transformed into a curved or irregular surface. The transformations we will implement are a) convex transformation, b) concave transformation and c) random transformation. The last step of the exercise is to generate planar surfaces from the new geometry obtained after the transformation.

Anatomy of the Code



Main Block:

This block of code has the initial setup of the tool. It defines the global variables and gets the user input (size of the grid, origin point and type of the transformation, and intensity multiplier). It also draws the initial Grid and calls the Transform Function.

Custom Functions:

This section contains the functions that do more specific tasks of the tool.

Transform takes the grid of points, and generates a new grid from it according to the type of transformation and intensity multiplier selected by the user. It also calls the *GenerateSurface* Function, which takes the new grid of points and runs an algorithm to generate planar, triangular surfaces across it. These planar surfaces can then be flattened and sent to a laser cutter, plasma cutter or CNC router (this part is not implemented).

The *RandomNumber* Function just takes a range (min and max) and returns a random number in between. It is used by the *Transform* Function to modify the surface in a less unpredictable way.

Now we will look at each one of these sections in more detail, and will go through the most important parts of the code.

Code in Detail*Main Function (or Main SubRoutine)*Sub **Main** ()

'Setting up layers

```
Rhino.AddLayer "Points", RGB(0, 255, 100)
Rhino.AddLayer"Points2", RGB(255, 0, 0)
Rhino.AddLayer"Surface1", RGB(0, 100, 255)
Rhino.AddLayer"Surface2", RGB(128, 128, 128)
```

'Declaring some important variables

Dim gridX, gridY, m, i, j, kind, intensity

'Getting user input

'Size of the grid

```
gridX=Rhino.IntegerBox("Size of the Grid in the X dimension", 10)
gridY=Rhino.IntegerBox("Size of the Grid in the Y dimension", 10)
```

'm is the size of the cell (module)

m = 10

'Allocating space for the array

reDim points(gridX, gridY)

```
Rhino.CurrentLayer "Points"
  for i = 0 to gridX
    for j = 0 to gridY
      reDim p(2)
      p(0) = i*m
      p(1) = j*m
      p(2) = 0
      points(i,j) = p
      Rhino.AddPoint(p)
    Next
  Next
Next
```

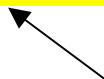
'Generate surface (TODO)

'Let the user select the focus point of the transformation

Dim focus

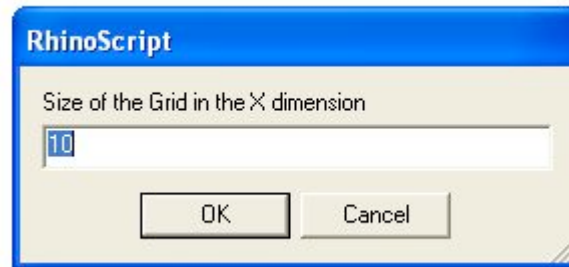
focus = Rhino.**GetPoint**("Select focus of the Transformation")**kind** = Rhino.**IntegerBox**("Enter 0 for Concave, 1 for Convex transformation, 2 for Random")**intensity** = Rhino.**RealBox**("Enter Intensity of transformation", 1.0)**call** transform(points, focus, kind, intensity, gridX, gridY)

End Sub



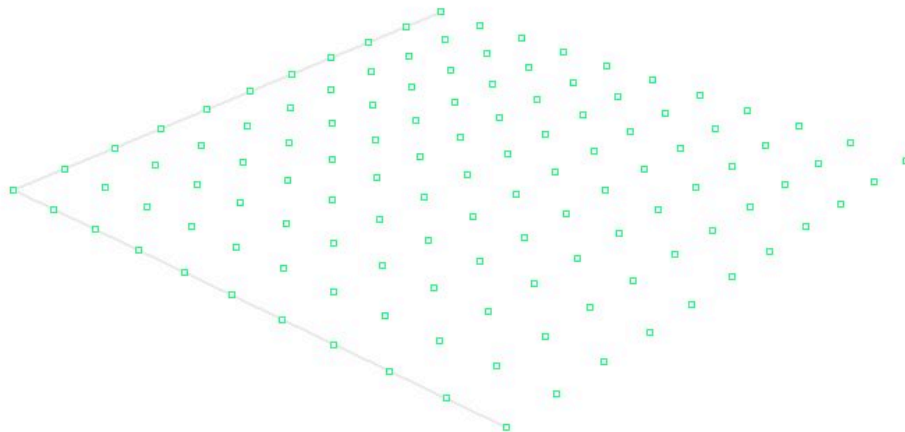
User input values are stored as variables in the script

```
gridX=Rhino.IntegerBox("Size of the Grid in the X dimension", 10)
```



A nested loop generates the planar grid

```
for i = 0 to gridX  
  for j = 0 to gridY  
    reDim p(2)  
    p(0) = i*m  
    p(1) = j*m  
    p(2) = 0  
    points(i,j) = p  
    Rhino.AddPoint(p)  
  Next  
Next
```



Transform Function

Function **transform**(p, f, kind, intensity, gX, gY)

```
Dim i, j, dist, fac
reDim pTr(gX, gY)
```

```
fac = 1/intensity
Rhino.CurrentLayer "Points2"
```

```
select case kind
```

```
case 0 'Concave
```

```
for i = 0 to gX
```

```
for j = 0 to gY
```

```
reDim pTransform(2)
dist = Rhino.Distance(p(i,j), f)
```

```
pTransform(0) = p(i,j)(0)
pTransform(1) = p(i,j)(1)
pTransform(2) = (p(i,j)(2) + dist/(fac*3))
```

```
pTr(i,j) = pTransform
```

```
Rhino.CurrentLayer "Points2"
Rhino.AddPoint(pTransform)
```

```
Next
```

```
Next
```

```
case 1 'Convex
```

```
for i = 0 to gX
```

```
for j = 0 to gY
```

```
reDim pTransform(2)
dist = Rhino.Distance(p(i,j), f)
```

```
pTransform(0) = p(i,j)(0)
pTransform(1) = p(i,j)(1)
pTransform(2) = (p(i,j)(2) - dist*(intensity/5))
```

```
pTr(i,j) = pTransform
```

```
Rhino.AddPoint(pTransform)
```

```
Next
```

```
Next
```

```
End Select
```

```
call generateSurface(pTr, gX, gY)
```

```
End Function
```



The first case (case 0) generates a new grid that is concave. The degree of concavity is a function of the intensity multiplier entered by the user.

```

case 0 'Concave

  for i = 0 to gX
    for j = 0 to gY

      reDim pTransform(2)
      dist = Rhino.Distance(p(i,j), f)

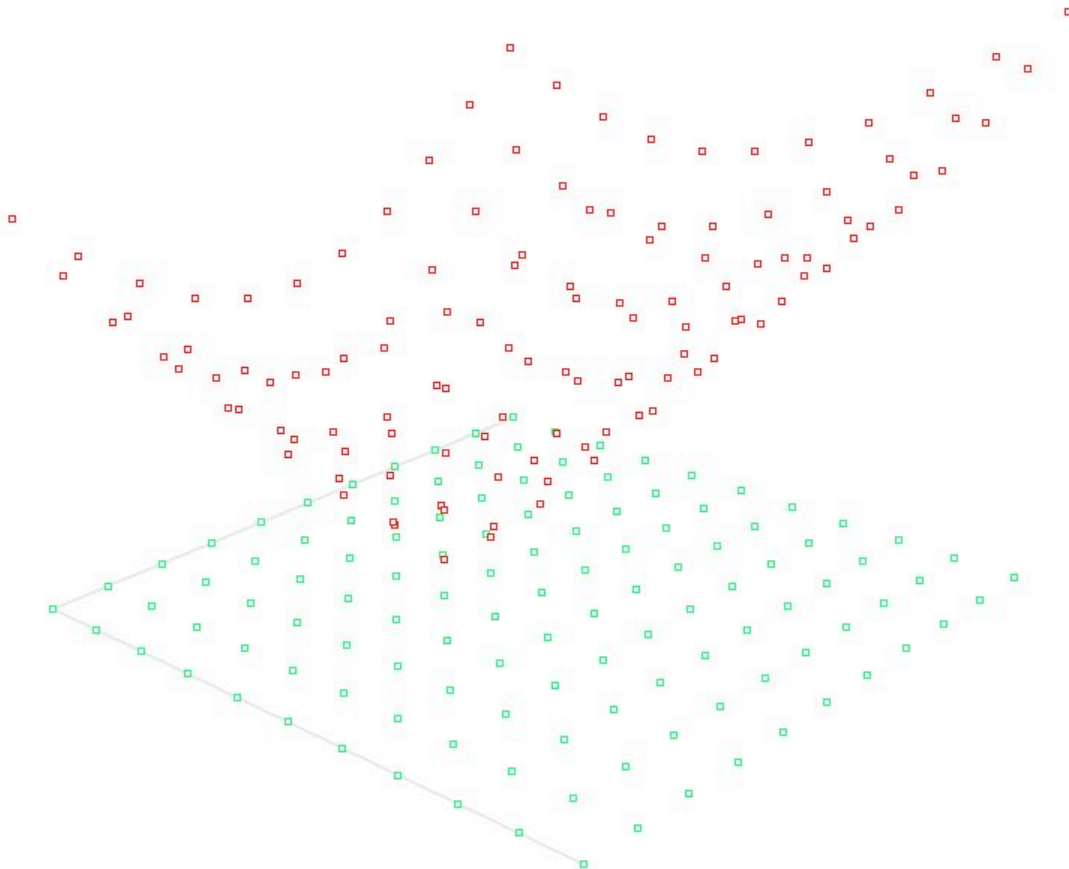
      pTransform(0) = p(i,j) (0)
      pTransform(1) = p(i,j) (1)
      pTransform(2) = (p(i,j) (2) + dist/(fac*3))

      pTr(i,j) = pTransform

      Rhino.CurrentLayer "Points2"
      Rhino.AddPoint(pTransform)

    Next
  Next

```



'Transform Function

Function **generateSurface**(points, xSize, ySize)

Dim i,j

Rhino.CurrentLayer "Surface1"

for i = 0 to xSize-1

for j = 0 to ySize-1

ReDim t1(2)

ReDim t2(2)

t1(0) = points(i,j)

t1(1) = points(i+1,j)

t1(2) = points(i+1,j+1)

t2(0) = points(i,j)

t2(1) = points(i+1,j+1)

t2(2) = points(i, j+1)

Rhino.**AddSrfPt**(t1)

Rhino.**AddSrfPt**(t2)

Next

Next

end Function

'RandomNumber Function

Function **RandomNumber**(nMin, nMax) 'Courtesy of McNeel

RandomNumber = Null

If Not IsNumeric(nMin) Then Exit Function

If Not IsNumeric(nMax) Then Exit Function

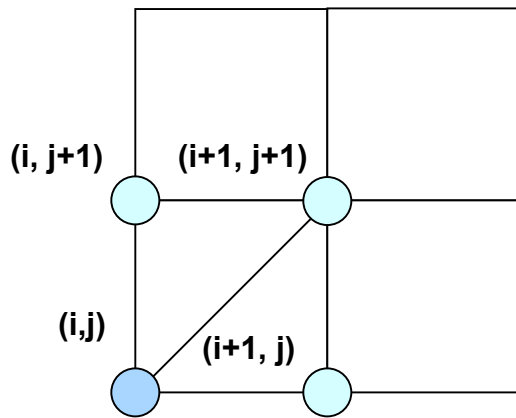
If nMin >= nMax Then Exit Function

Randomize

RandomNumber = Int((nMax - nMin + 1) * Rnd + nMin)

End Function

The triangular planar surfaces are generated through an algorithm that iterates through the 2D array of points, and joins sets of three points.

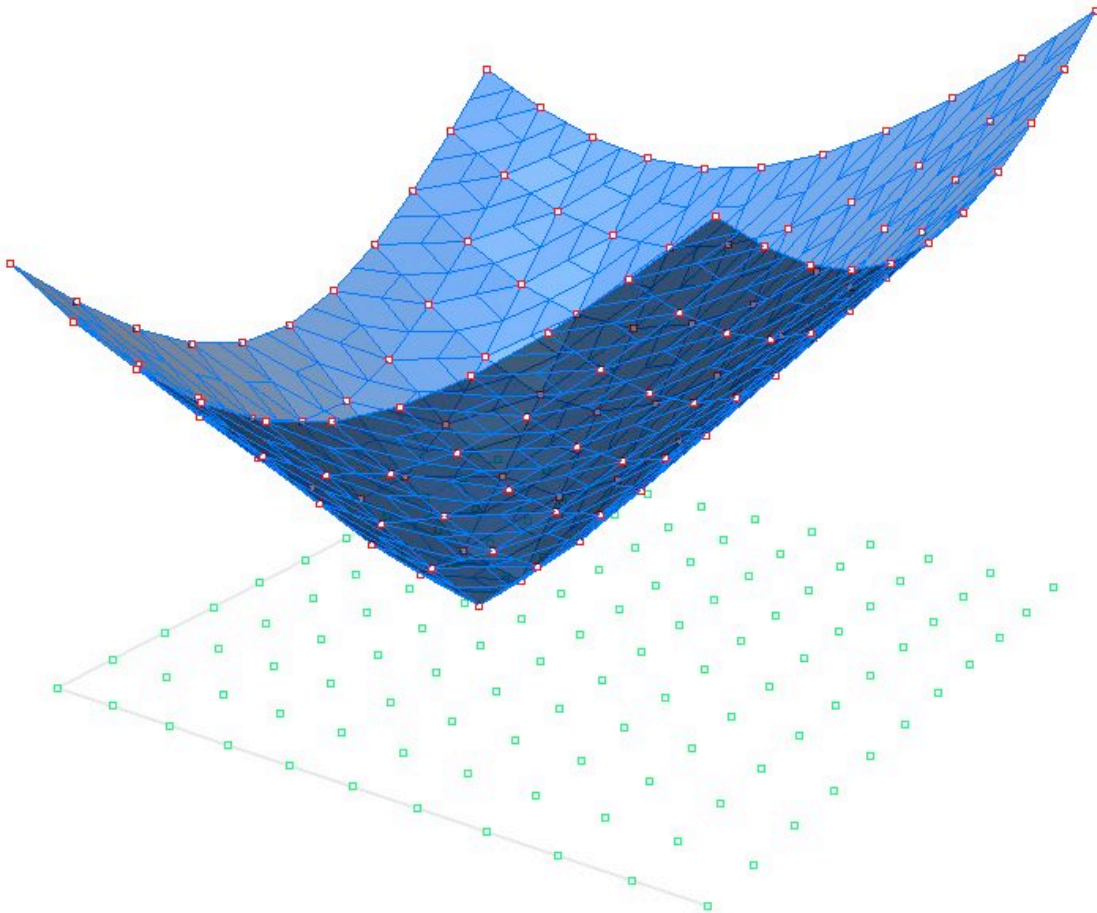


```
ReDim t1(2)
ReDim t2(2)

t1(0) = points(i,j)
t1(1) = points(i+1,j)
t1(2) = points(i+1,j+1)

t2(0) = points(i,j)
t2(1) = points(i+1,j+1)
t2(2) = points(i,j+1)

Rhino.AddSrfPt t1
Rhino.AddSrfPt t2
```



Questions/Further Exercises

Implement `case 2` of the function *Transform* (not provided), to generate a modified grid and surfaces that varies randomly according to the intensity multiplier.

Implement a function that lays out the planar triangles for cutting. What do we have to do?

Hints: First of all we need to store the triangles somehow; then we can iterate through them and flatten the surfaces. It is vital to lay them out in an organized way, and to label them so that the assembly is fast and precise, without waste of materials.