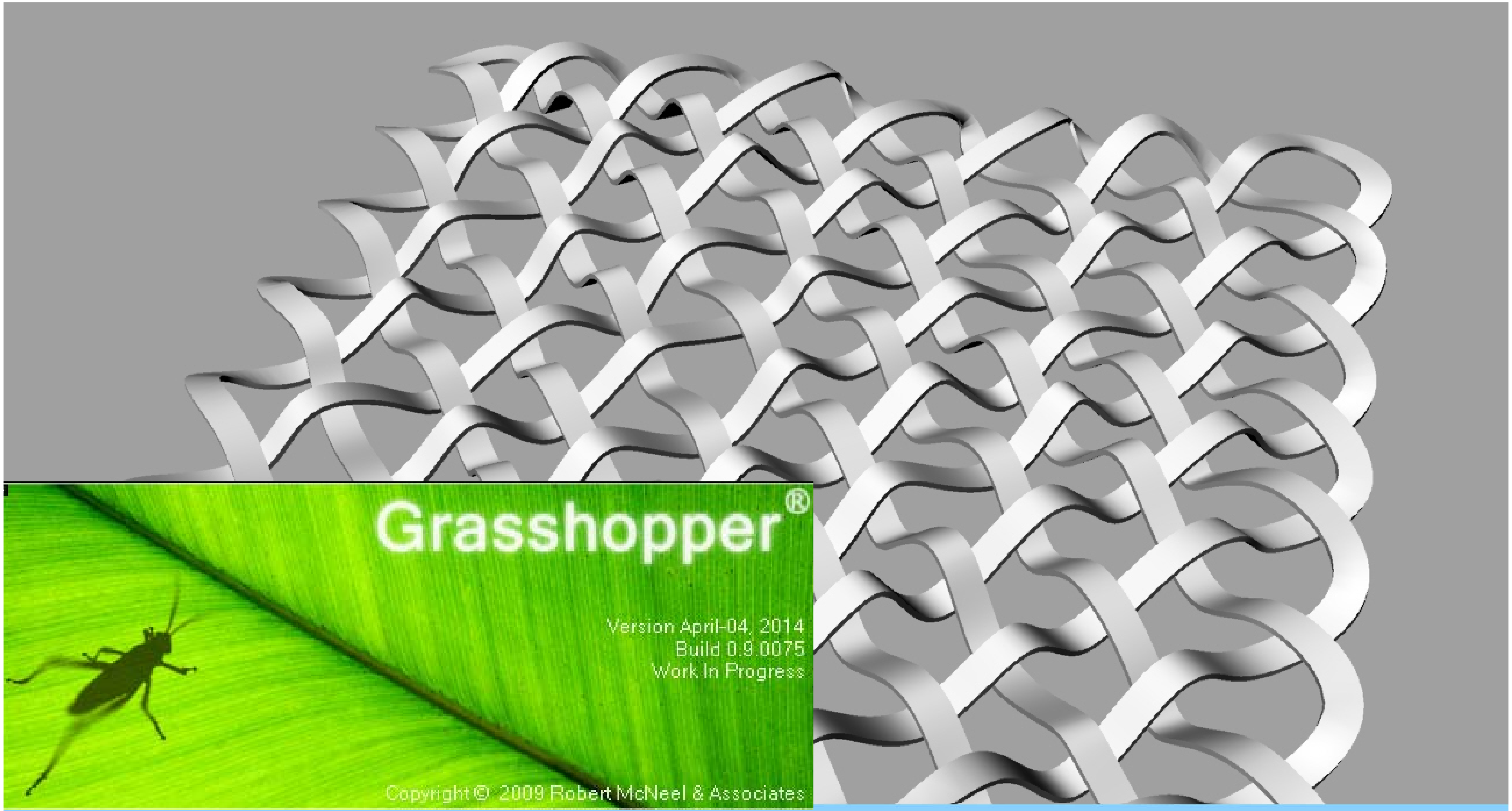
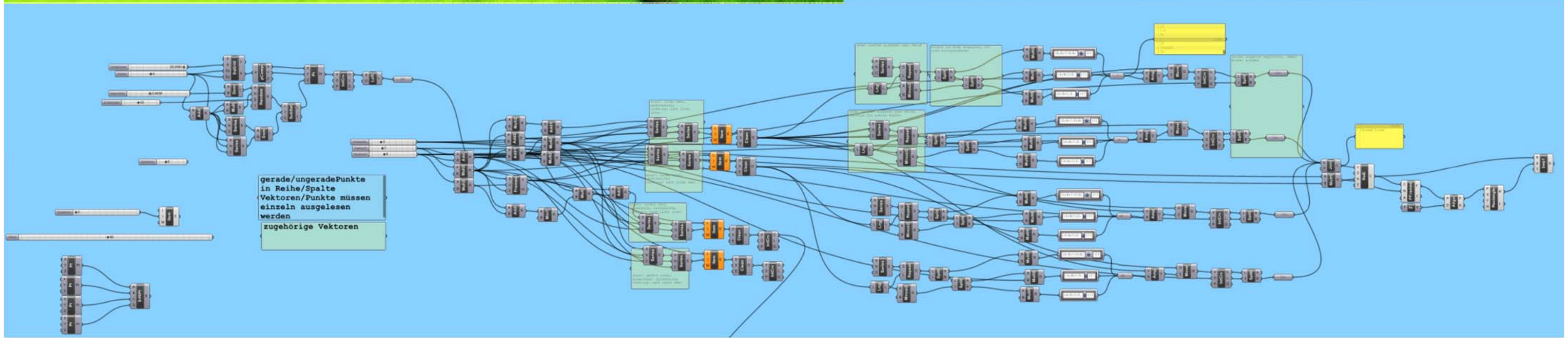
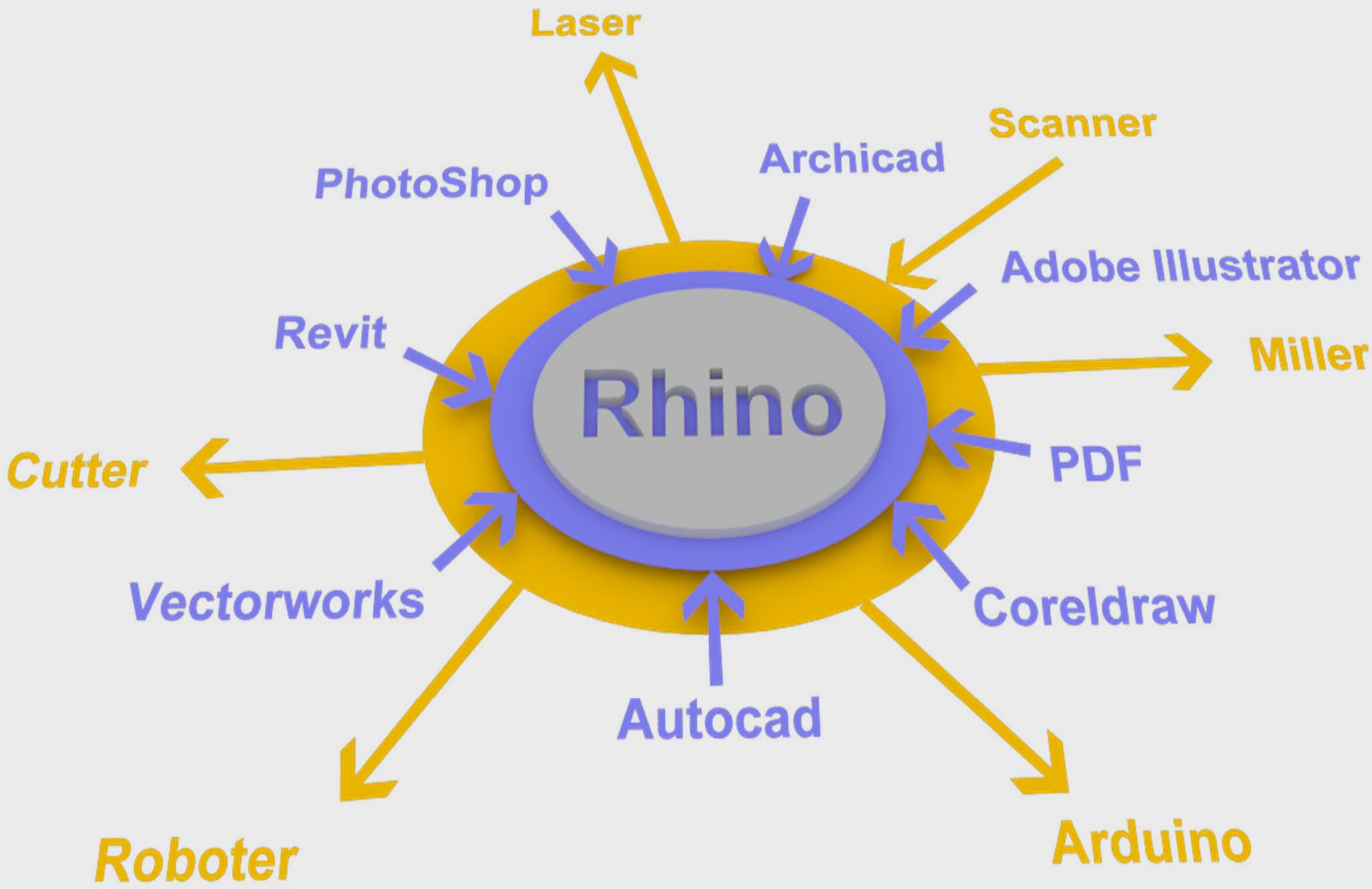




gerade/ungeradePunkte
in Reihe/Spalte
Vektoren/Punkte müssen
einzeln ausgelesen
werden

zugehörige Vektoren



Laser

PhotoShop

Archicad

Scanner

Adobe Illustrator

Revit

Rhino

Miller

Cutter

PDF

Vectorworks

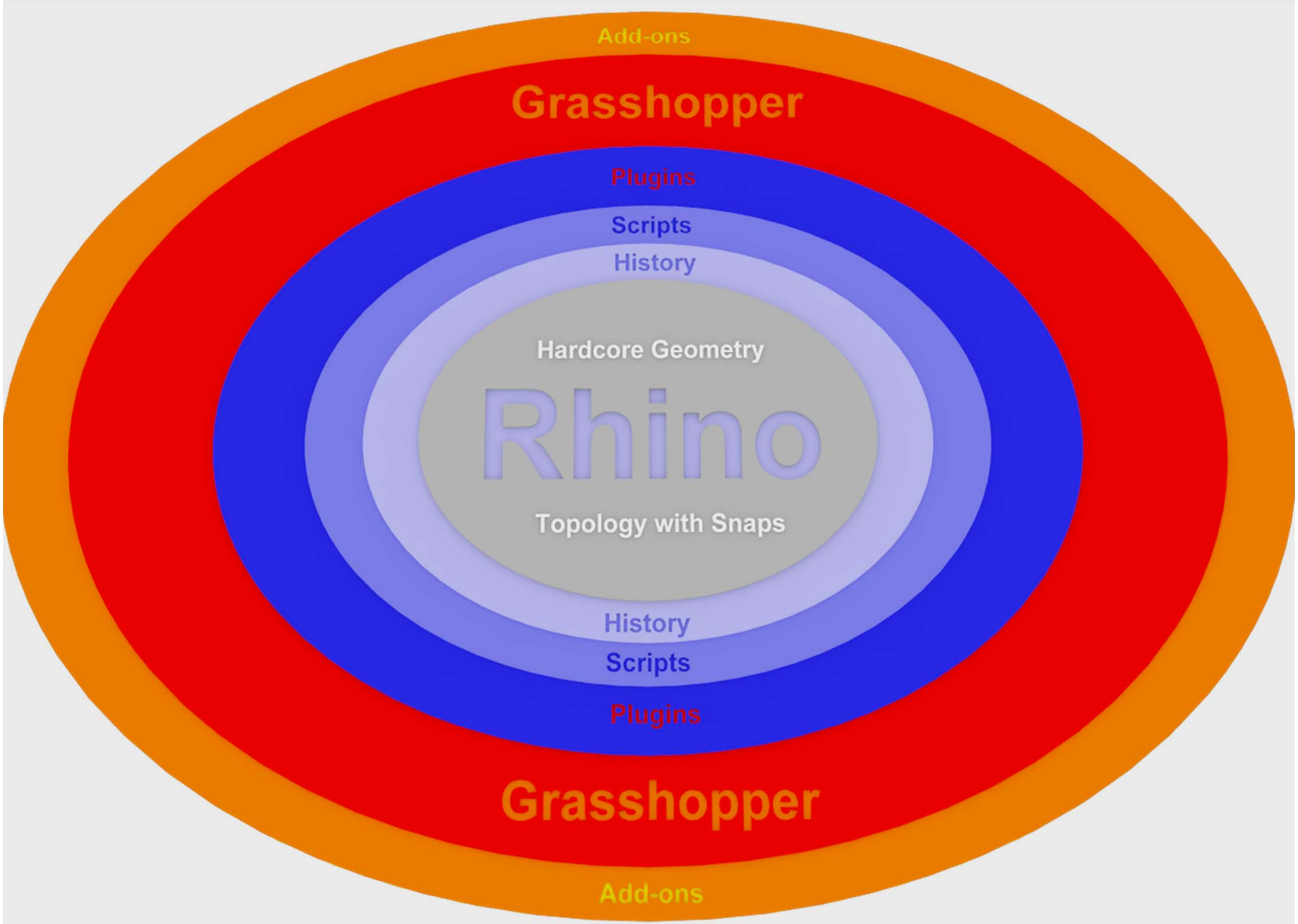
Coreldraw

Autocad

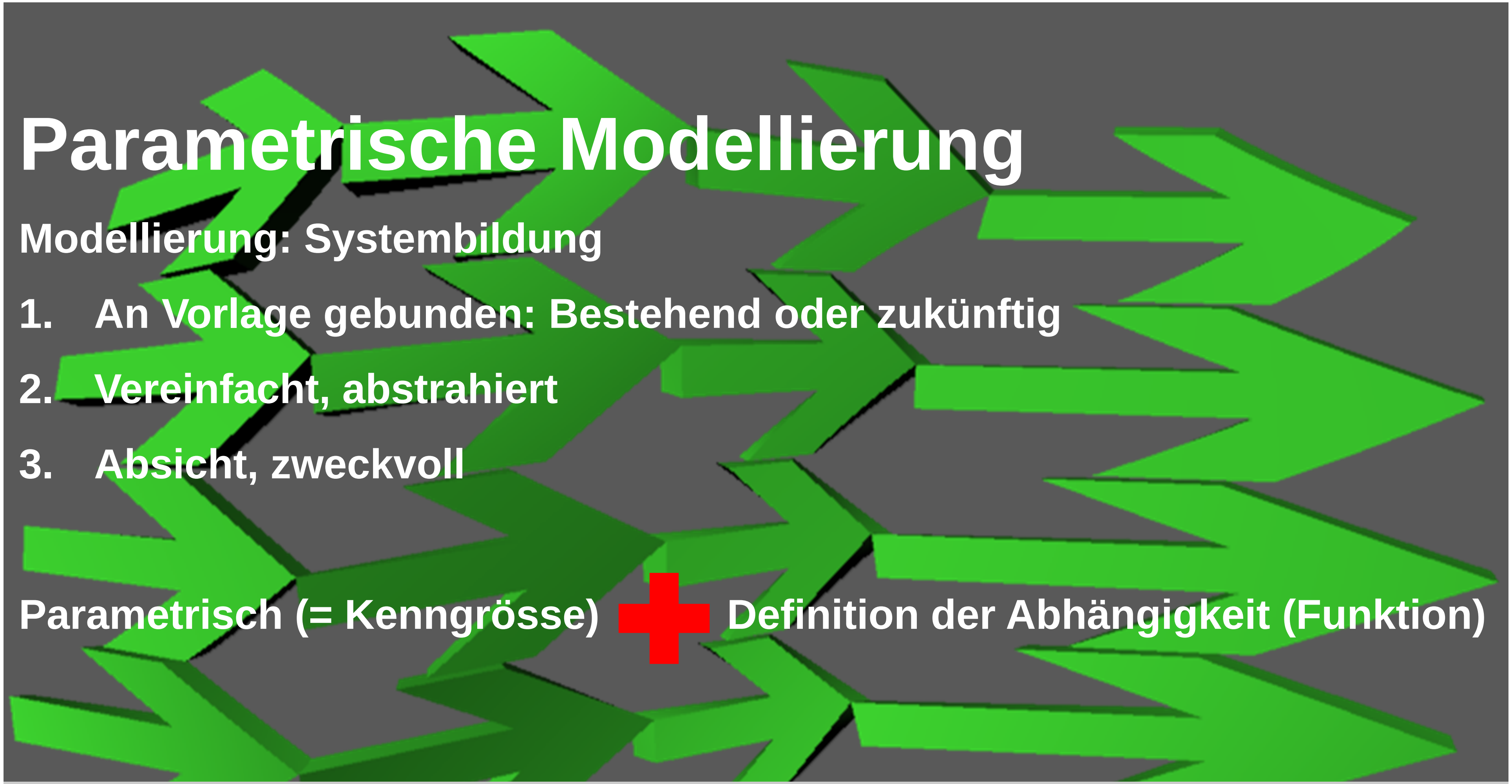
Roboter

Arduino

Rhino als Austauschplattform



Rhino Erweiterungen

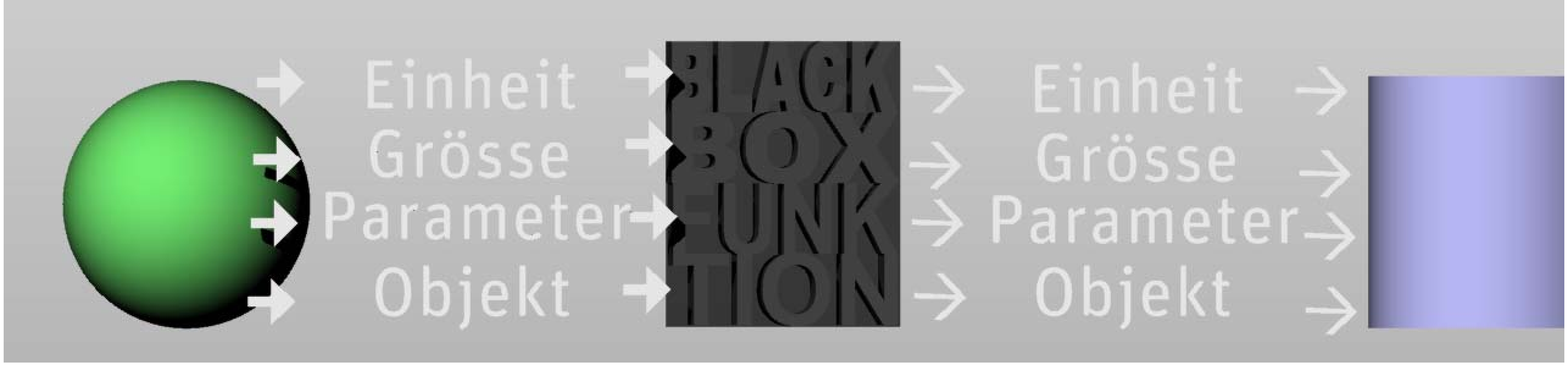


Parametrische Modellierung

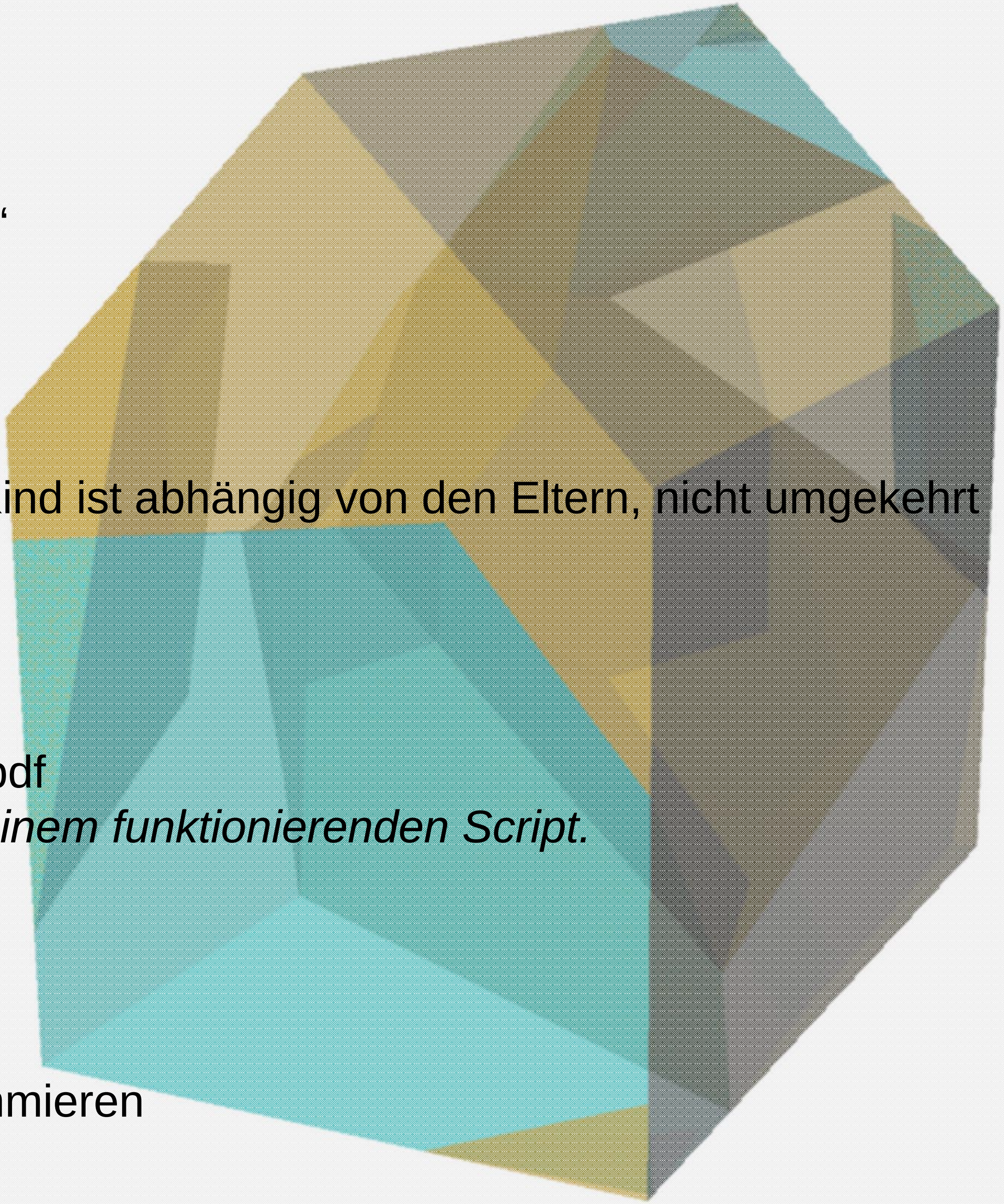
Modellierung: Systembildung

1. **An Vorlage gebunden: Bestehend oder zukünftig**
2. **Vereinfacht, abstrahiert**
3. **Absicht, zweckvoll**

Parametrisch (= Kenngrösse) + Definition der Abhängigkeit (Funktion)



Programmieren



Makros
Rhino F1 „makro““Befehlsliste“
_macroeditor

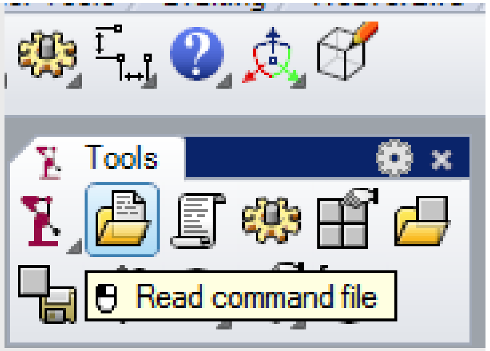
Historie
Beruht auf «parenting»: Das Kind ist abhängig von den Eltern, nicht umgekehrt

VB-Scripting
Pflichtlektüre:
David Rutten Rhinoscript101.pdf
Suchen Sie im Internet nach einem funktionierenden Script.

Grasshopper
Plug-in für grafisches Programmieren

Makros sind Stapelverarbeitungsdateien für sich häufig wiederholende Befehle

Befehle auf Englisch schreiben, Übersetzung in der Hilfe



! (Ausrufezeichen) Löscht den vorhergehenden Befehl

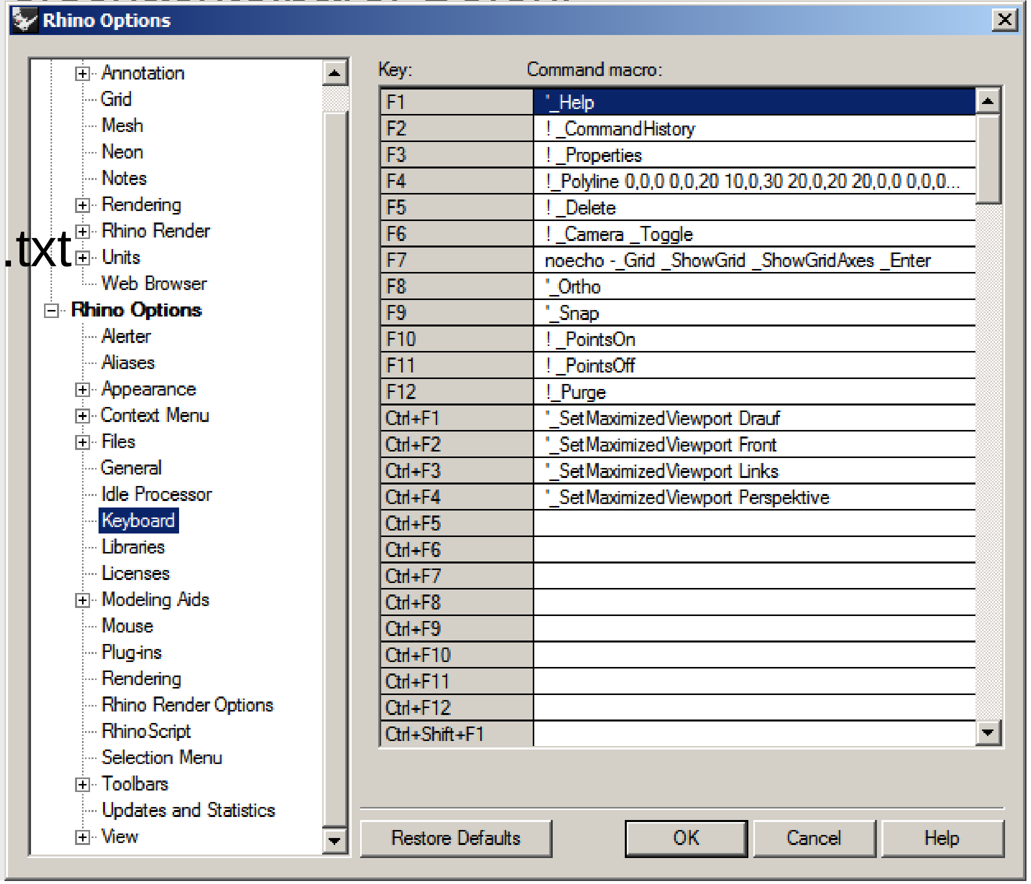
_ (Unterstrich) Ruft den Befehl als englischen Befehlsnamen auf

- (Bindestrich) Dialogfenster unterdrücken

' (Apostroph) Der nächste Befehl ist ein verschachtelbarer Befehl

; (Semikolon) Kommentar

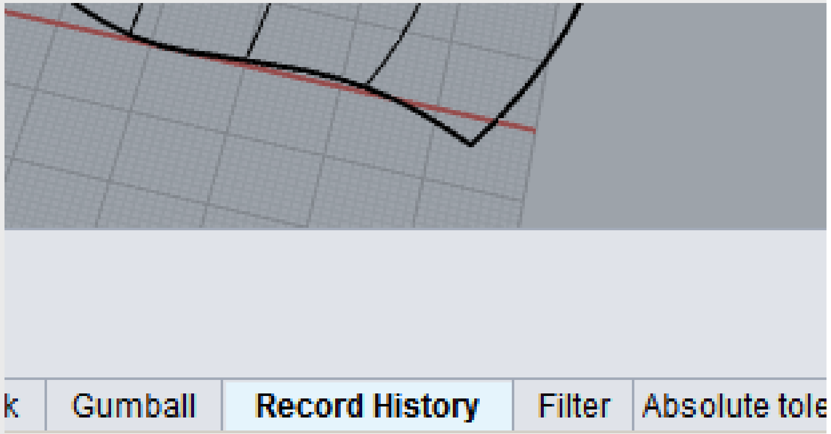
1. Werkzeuge – Befehlsdatei lesen- *.txt
2. Oder als Kurzbefehl einrichten (F4)
3. Oder für Mac:
   _Delete (F5)



Key:	Command macro:
F1	'_Help
F2	! _CommandHistory
F3	! _Properties
F4	!_Polyline 0,0,0 0,0,20 10,0,30 20,0,20 20,0,0 0,0,0...
F5	! _Delete
F6	! _Camera _Toggle
F7	noecho -_Grid _ShowGrid _ShowGridAxes _Enter
F8	'_Ortho
F9	'_Snap
F10	! _PointsOn
F11	! _PointsOff
F12	!_Purge
Ctrl+F1	'_SetMaximizedViewport Drauf
Ctrl+F2	'_SetMaximizedViewport Front
Ctrl+F3	'_SetMaximizedViewport Links
Ctrl+F4	'_SetMaximizedViewport Perspektive
Ctrl+F5	
Ctrl+F6	
Ctrl+F7	
Ctrl+F8	
Ctrl+F9	
Ctrl+F10	
Ctrl+F11	
Ctrl+F12	
Ctrl+Shift+F1	

Makros

„Historie aufnehmen“ aktivieren während dem Sie einen Befehl ausführen, zB Loften.
Nun kann man die Kurven modifizieren – Die Flächen werden folgen.
Modifizieren Sie die Flächen, geht die Beziehung verloren.



Benutzen Sie die Hilfe F1(Help) mit dem Suchbegriff „Historie“ um herauszufinden, welche Befehle „history – enabled“ sind.

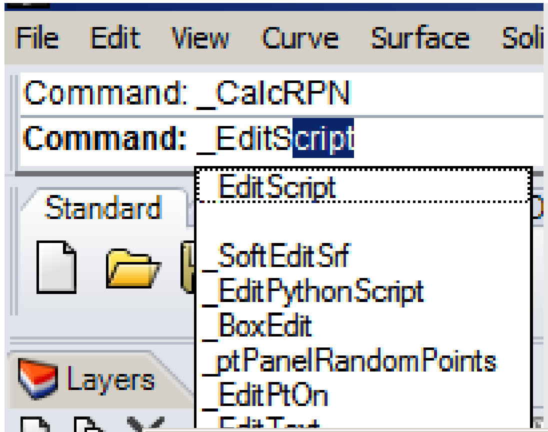
Historie

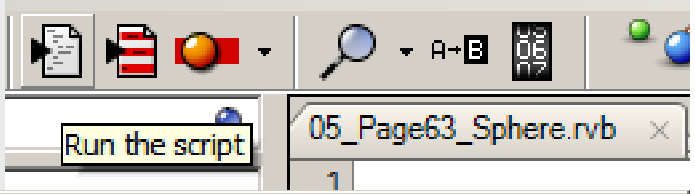
Skripts bearbeiten und ausführen

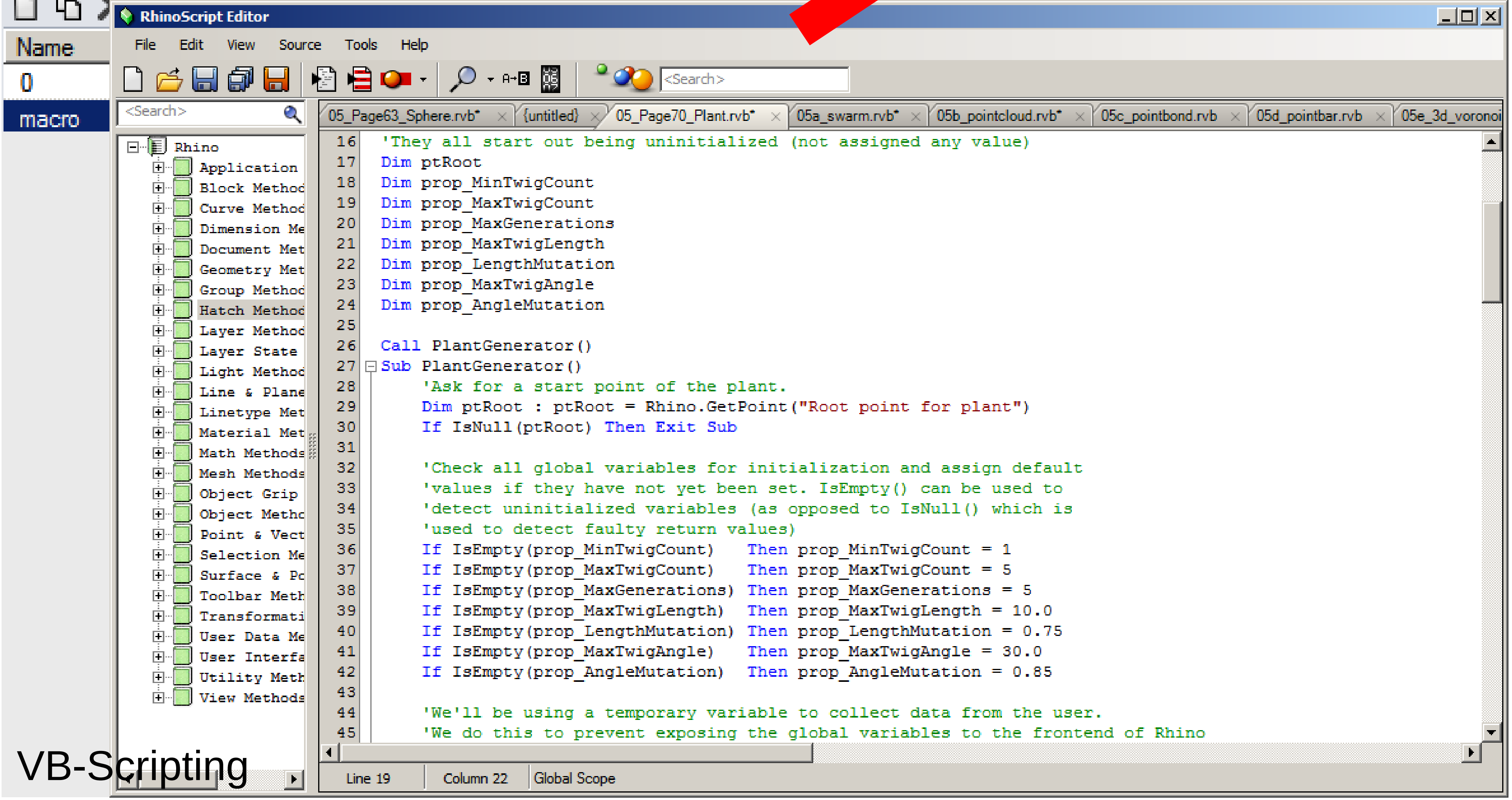
In Befehlszeile eingeben: SkriptBearbeiten -

Datei – öffnen

Ausführen: LCLICK auf







```
'They all start out being uninitialized (not assigned any value)
Dim ptRoot
Dim prop_MinTwigCount
Dim prop_MaxTwigCount
Dim prop_MaxGenerations
Dim prop_MaxTwigLength
Dim prop_LengthMutation
Dim prop_MaxTwigAngle
Dim prop_AngleMutation

Call PlantGenerator()
Sub PlantGenerator()
    'Ask for a start point of the plant.
    Dim ptRoot : ptRoot = Rhino.GetPoint("Root point for plant")
    If IsNull(ptRoot) Then Exit Sub

    'Check all global variables for initialization and assign default
    'values if they have not yet been set. IsEmpty() can be used to
    'detect uninitialized variables (as opposed to IsNull() which is
    'used to detect faulty return values)
    If IsEmpty(prop_MinTwigCount)  Then prop_MinTwigCount = 1
    If IsEmpty(prop_MaxTwigCount)  Then prop_MaxTwigCount = 5
    If IsEmpty(prop_MaxGenerations) Then prop_MaxGenerations = 5
    If IsEmpty(prop_MaxTwigLength)  Then prop_MaxTwigLength = 10.0
    If IsEmpty(prop_LengthMutation) Then prop_LengthMutation = 0.75
    If IsEmpty(prop_MaxTwigAngle)   Then prop_MaxTwigAngle = 30.0
    If IsEmpty(prop_AngleMutation)  Then prop_AngleMutation = 0.85

    'We'll be using a temporary variable to collect data from the user.
    'We do this to prevent exposing the global variables to the frontend of Rhino
```

VB-Scripting

RhinoScript

Zwischen Makros und kompilierten (in Maschinensprache übersetzten) Programmen

Rhinoscript basiert auf Microsoft Visual Basic Script (VBScript):
BASIC-Familie in 3. Generation
(Beginners All purpose Symbolic Instruction Code, 1963)
Und ist eine
Objektorientierte Sprache (Gegenteil: Stapelverarbeitungsdatei)

Die Aufgaben der Syntax sind:

Umgang mit Variablen Daten
Kontrolle des Zeilenflusses (Flow control)
input – output Steuerung

Variablen

Soll der Code dynamisch sein, muss er mit allerhand Daten umgehen können. Die gängigsten Variablen sind:

integer - a real number - Ganzzahl
double - a decimal place number - Fliesskommazahl
booleans - either true or false – Entweder true oder false
strings - a set of characters - Text

Arrays sind Listen von Variablen. Sie werden über ein nullbasiertes Zählsystem – dem Index abgerufen.

```
Dim arrExample(3)
arrExample(0) = 9
arrExample(1) = 11
arrExample(2) = 3
arrExample(3) = 7
```

below is another way of constucting an array using the array function. note that this doesn't require the brackets when declaring the variable.

```
Dim arrExample
arrExample = Array(9,11,3,7)
```

to access a value from an array we use its index - this example should print "11"

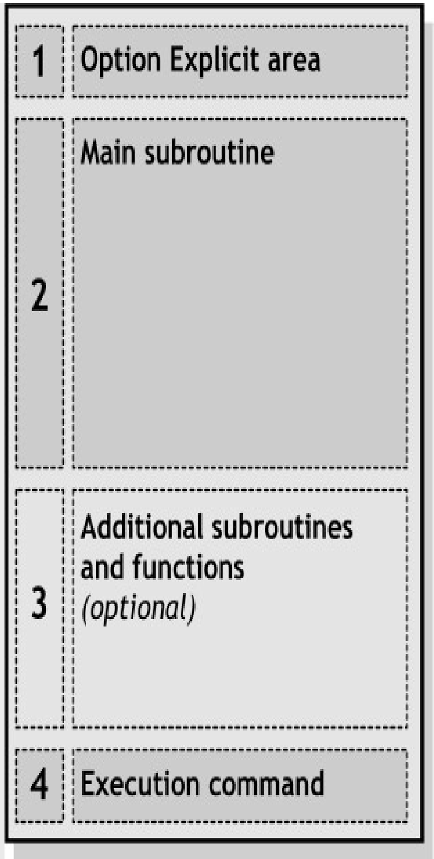
```
Rhino.Print arrExample(1)
```

Variable Daten

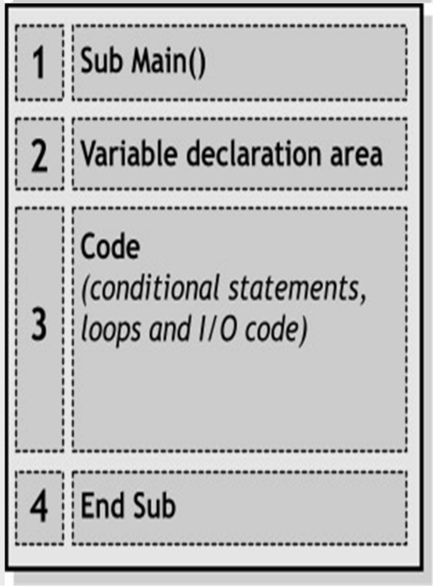
Code Structure

Every script requires at least one function (or subroutine) which contains the main code of the script. It doesn't have to be a big function, and it can place calls to any number of other functions but it is special because it delineates the extents of the script. The script starts running as soon as this function is called and it stops when the function completes. Without a main function, there is nothing to run.

Functions are not run automatically by the interpreter. They have to be called specifically from other bits of code. The only way to start the cascade of functions calling functions, is to place a call to the main subroutine somewhere outside all function declarations. You could put it anywhere, including at the very bottom of the script file, but I prefer to keep it near the top, just after the Option Explicit statement and just before the main subroutine begins. Without a main function call your script will be parsed and compiled, but it will not be executed. Do not get confused by terms such as 'function', 'subroutine', 'procedure' or 'method', at this time they all pretty much mean the same thing.



```
Option Explicit
' script by someone            ' this is a comment - plain text not read
Public myVariable             ' this is a global variable
Call Main()                   ' this tells rhino to run the main function
Sub Main()                    ' this is the start of the main function
Dim strText                   ' this declares a variable
strText = "hello world"       ' this assigns some text to a variable
Rhino.Print strText           ' this executes a rhino method
End Sub                       ' this is the end of the main function
```


1
Sub Main()
2
Variable declaration area
3
Code (conditional statements, loops and I/O code)
4
End Sub

Grundstruktur

Rhino Methods

rhino methods are blocks of code which are built into rhinoScript and can be executed by the scripts that you write. methods can be thought of as similar to commands, which do certain things such as add a circle, copy an object, scale an object etc. there is not necessarily a method for every rhino command but typically there are a lot of methods which are similar to the rhino commands as well as many useful methods for doing calculations such as vector math and interrogating objects. a complete list of rhino methods can be found in both the rhinoScript help and the left pane of the monkey script editor. the following is the syntax of a rhino method where arrPlane is a rhino plane at which the centre of the circle will be drawn and dblRadius is the radius of the circle (a double):

```
Rhino.AddCircle arrPlane, dblRadius
```

another example is the method to print information to the command line

```
Rhino.Print "some text"
```

or

```
strText = "some text"
Rhino.Print strText
```

Rhino Befehle

Input

some scripts will require that the user provides the input to the script, such as: selecting objects, choosing options and inputing values. this is done through rhino methods such as **GetReal and GetObject:**

```
dblNumber = Rhino.GetReal("input a number")
strObject = Rhino.GetObject("Pick any object")
```

the following example demonstrates how to input the radius of a circle and defines a maximum and minimum range

```
Option Explicit
Call Main()
Sub Main()
		Dim dblRadius, arrPlane
		dblRadius = Rhino.GetReal("Radius of new circle", 3.14, 1.0)
		arrPlane = Rhino.WorldXYPlane
		Rhino.AddCircle arrPlane, dblRadius
End Sub
```

Input

loops

loops are a way of repeating a bit of code either a certain number of times (incremental) of until a certain condition is met (conditional).

incremental

this first example loops through objects in an array

```
Dim arrObj, strObj
For Each strObj In arrObj ' execute some code to be repeated for every object in arrObj
Next
```

the next example loop by incrementing a variable (i) until it reaches a defined limit (x)

```
Dim i, x
For i = 0 To x step 2 ' execute some code to be repeated x/2 times
Next
```

conditional

```
Do While x < 10 ' execute some code to be repeated continually until x >= 10
Loop
```

a conditional loop can be written as a while or until loop

```
Do Until x = 10 ' execute some code to be repeated continually until x = 10
Loop
```

there are several different was of writing a conditional loop

Flow Control: Loop

Conditional Statements

conditional statements control flow through testing to see if a condition is met and then choosing what to do. the most common conditional statement is an 'if then' statement. basically this is stating that if a condition is met then execute the following code.

```
If (x > 10) Then ' execute some code
End If
```

an advance on the if statement is the if else statement:

```
If (x > 10) Then ' execute some code
Else ' execute some other code
End If
```

`If..Then..ElseIf..Else`

Instead of using a single Else in an If..Then-structure, we can also add an unlimited amount of ElseIf statements to avoid nested If..Then-structures.

The `Select..Case` statement allows us to compare variables with other variables or with data very quickly. `Select..Case` can result in more readable code than If..Then-structures.

Flow control: Conditionals

- **numbers 15, 26, 2.33**
- **variables x**
- **operators =, *, /**
- **functions Sin(), Sqr(), Log()**

Numbers, variables and operators should be familiar from everyday life. vbScript has a limited amount of operators and they are always used in between the variables/values they apply to. Here you see a list of all operators in vbScript:

+ add two values
- substract two values
* multiply two values
/ divide two values
\ divide two values but return only whole numbers
^ raise a number to an exponent
Mod arithmetic modulus

The following operators deal with boolean values.

And performs a logical conjunction - Verbindung
Eqv performs a logical equivalence - Gleichwertigkeit
Imp performs a logical implication - Verwicklung
Not performs a logical negation - Verneinung
Or performs a logical disjunction - Verzweigung
Xor performs a logical exclusion - Ausschluss

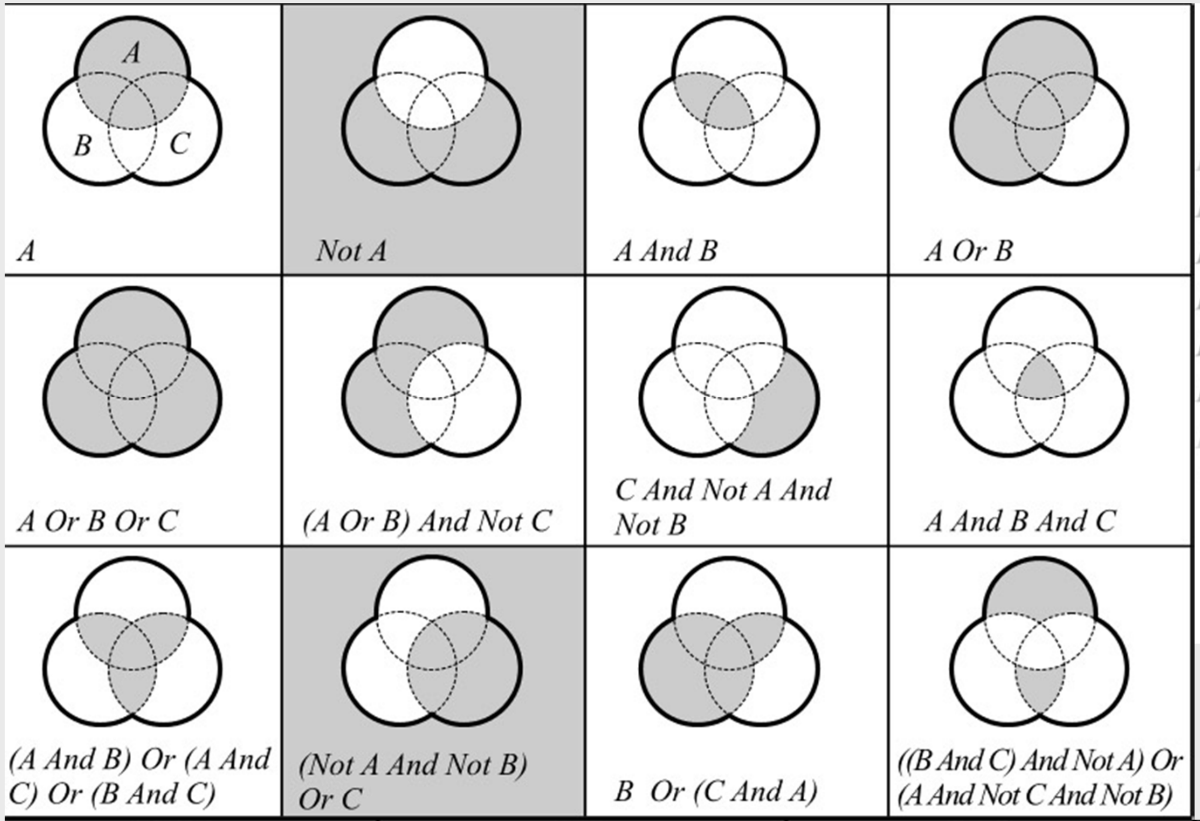
Functions are always added in front of the value(s) they use as input. These values are encapsulated by parentheses behind the function:

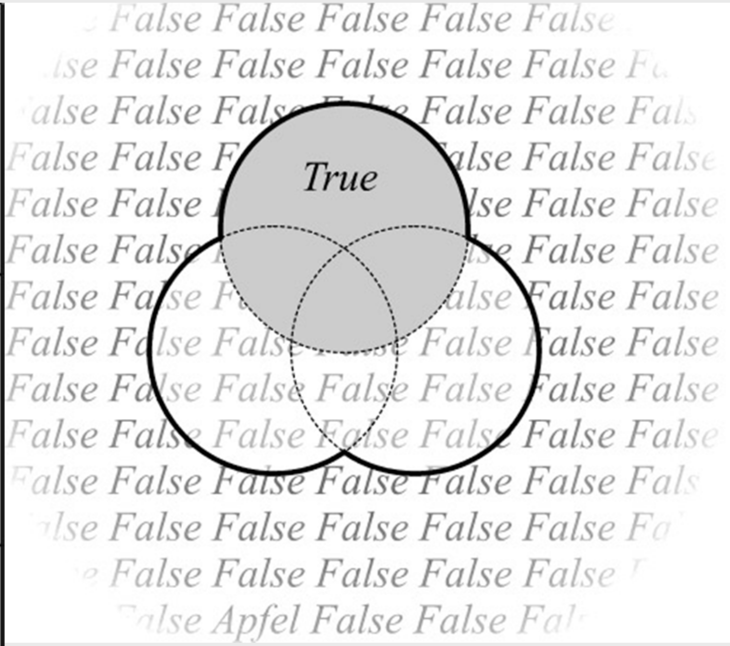
Sin(5) Sine of a number
Cos(x) Cosine of a number
Atn(x/y) ArcTangent of a number
Log(t^2) Natural logarithm of a number larger than 0
Sqr(3+a) Square root of any positive number
Abs(pi/i) Absolute (positive) value of any number

Operatoren, Funktionen

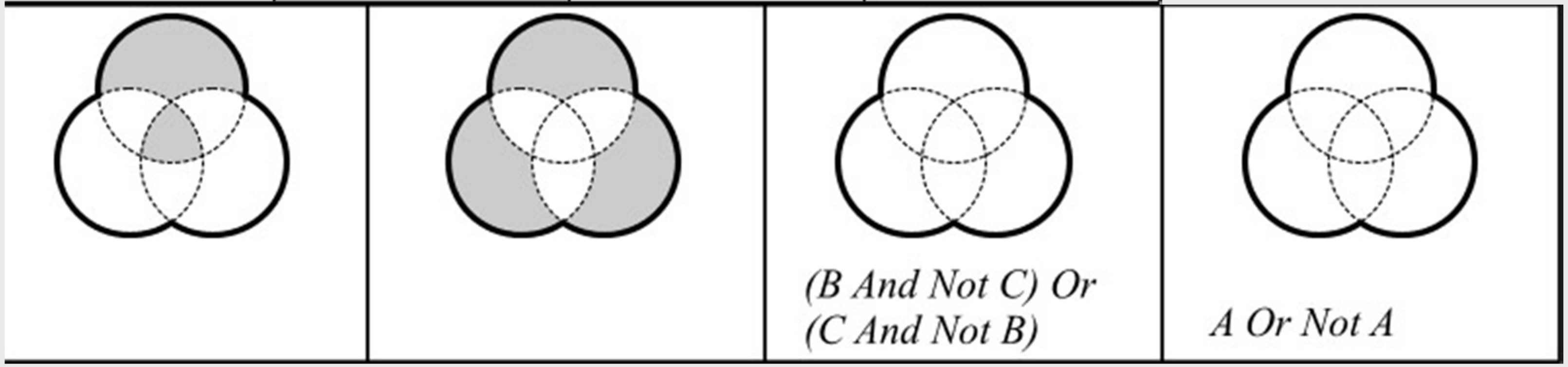
Once you start using boolean operators, things tend to become very complex very quickly:

blnValue = (blnA Or blnB) And (blnC Or Not blnB) And Not blnA




True
False False False False False
False False False False False
False False False False False
False Apfel False False

Venn - Diagramm

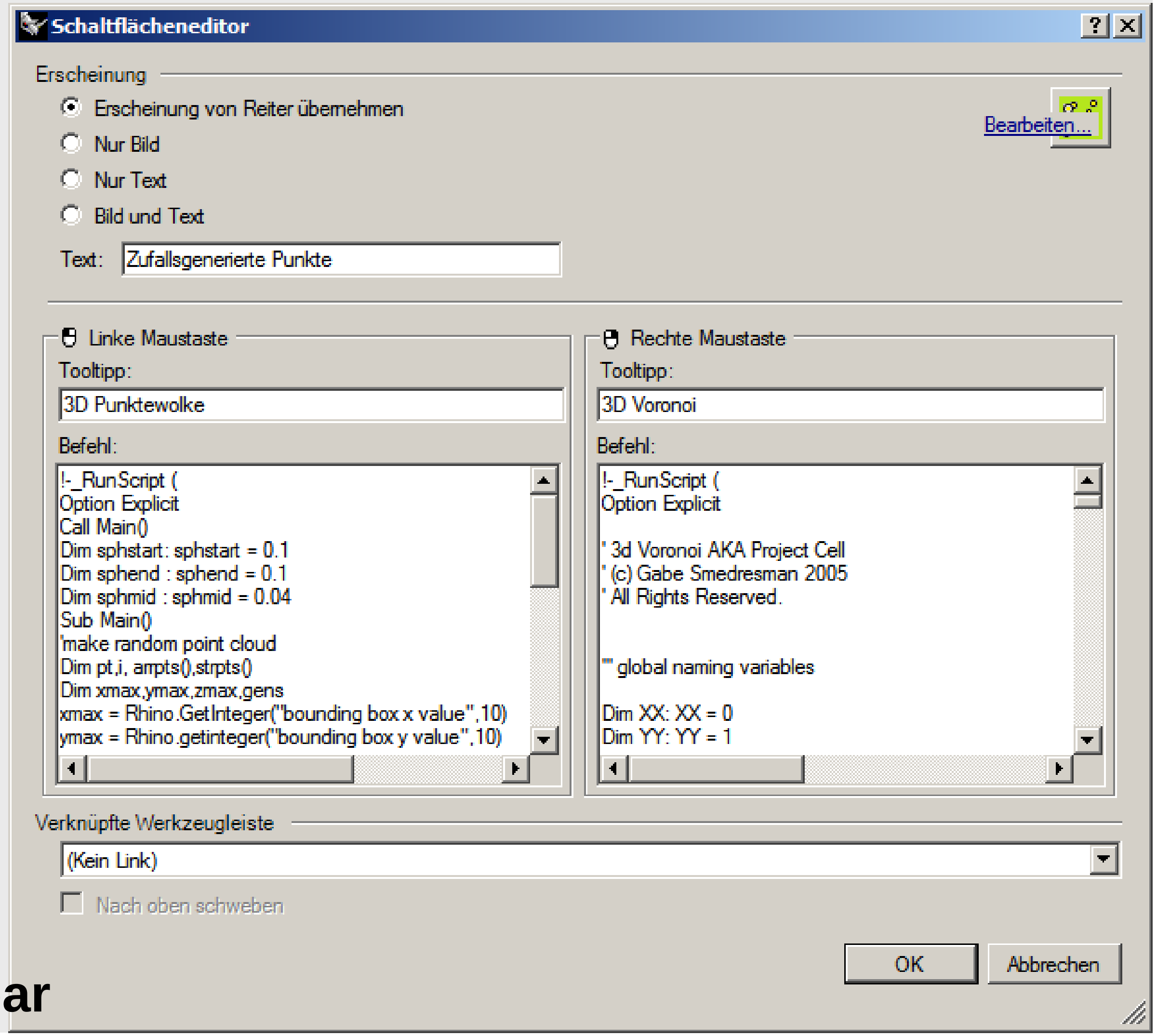


Kombination von Boolschen Operatoren

Toolbar - Befehl

CTRL - LMT auf icon:
Befehl kopieren
SHIFT – RMT auf icon:
Befehl bearbeiten
Oben rechts: icon bearbeiten, auch neues (Datei – Importieren) 24x24 px
Tooltip anpassen
und
Skript kopieren und zwischen den runden Klammern einfügen

```
-_RunScript (
        <script code here>
)
```



Erscheinung

● Erscheinung von Reiter übernehmen
○ Nur Bild
○ Nur Text
○ Bild und Text

Text: Zufallsgenerierte Punkte

Linke Maustaste

Tooltipp: 3D Punktewolke

Befehl:

```
!-_RunScript (
Option Explicit
Call Main()
Dim sphstart: sphstart = 0.1
Dim sphend : sphend = 0.1
Dim sphmid : sphmid = 0.04
Sub Main()
'make random point cloud
Dim pt,i, arrpts(),strpts()
Dim xmax,ymax,zmax,gens
xmax = Rhino.GetInteger("bounding box x value",10)
ymax = Rhino.getinteger("bounding box y value",10)
```

Rechte Maustaste

Tooltipp: 3D Voronoi

Befehl:

```
!-_RunScript (
Option Explicit

' 3d Voronoi AKA Project Cell
' (c) Gabe Smedresman 2005
' All Rights Reserved.


''' global naming variables

Dim XX: XX = 0
Dim YY: YY = 1
```

Verknüpfte Werkzeugleiste

(Kein Link)

☐ Nach oben schweben

Skripts ausführen mit Toolbar