

INTRODUCTION TO RHINOSCRIPT

MACROS – STATIC (fixed, linear sequence)

SCRIPTS – DYNAMIC (non-linear)

- flow control (skipping and repeating lines)
- variable control (logical and mathematical operations)
- input and output (user interaction)

SYNTAX

a set of rules that define what is and isn't valid

CONDITIONAL STATEMENTS

If... Then... Else
Select... Case

LOOPS

repeating of steps (lines)

TYPES OF VARIABLES:

- Longs (sometimes called integers)
- Doubles (sometimes called reals)
- Booleans
- Strings

LONGS & DOUBLES

Numeric variables

Long variable is a 32bit variable and can store any whole number in the range:

`[-2,147,483,648 +2,147,483,647]`

A Double variable is also 32bit, but because it stores numbers in a different manner it has a much larger range:

`[-1.79769313486232 × 10308 to -4.94065645841247 × 10-324]
[+4.94065645841247 × 10-324 to +1.79769313486232 × 10308]
0 (zero)`

Syntax for working with numeric variables:

```
x = 15 + 26  
x = 15 + 26 * 2.33  
x = Sin(15 + 26) + Sqr(2.33)  
x = Tan(15 + 26) / Log(55)
```

You can use variables to redefine their value:

```
x = x+1
```

BOOLEAN VARIABLES

can only store 2 values mostly referred to as Yes or No, True or False.

Booleans play an important role in the flow control, i.e. in conditional statements

In vbScript always use "**vbTrue**" and "**vbFalse**"

(The 'vb' prefix means that the word is part of the vbScript engine and not defined by us.)

STRINGS

Strings are used to store alphanumeric variables (text).

Strings always start and end with quotes. Whatever is inside the quotes is always text.

Cannot perform calculations with strings.

Use them to store in/output information and (object) names.

Syntax for Strings is quite simple:

```
a = "Ball State University"  
a = "The value you wanted to know is: " & 46.112
```

In vbScript we can append numeric values to Strings, but not the other way around!

The ampersand sign '&' is used to join several variables into a single String

RHINOSCRIPT FILES

Stored as text files with the suffix *.rvb. 'rvb' which stands for 'RhinoVisualBasic'.

ScriptName.rvb

ConTEXT - a widely used code-editor with syntax highlighting.

LoadScript – Rhino command to load scripts

RHINOSCRIPT TEXTFILE LAYOUT

VBScripts consist of 4 sections, one of which is optional:

- 1 Option Explicit area
- 2 Main subroutine
- 3 Additional subroutines and functions (optional)
- 4 Execution command

OPTION EXPLICIT

Always add the `Option Explicit` section.

Used to check whether the variable are properly declared.

Use comments to write who wrote the script, what the script does, add the current date.

COMMENTING

Start the line with an apostrophe – those lines are skipped during execution.

MAIN SUBROUTINE

Consists of four sections:

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

Declare subroutines by using the `sub` prefix.

Declare (initialize) variables by using the **Dim** prefix.

VARIABLE NAMES

Use a 3 character prefix which indicates the type of variable:

str for strings

dbl for doubles

lng for longs

bln for Booleans

OPERATORS AND FUNCTIONS

ARITHMETIC OPERATORS

+ add two values

- subtract two values

***** multiply two values

/ divide two values

**** divide two values but return only whole numbers

^ raise a number to the power of an exponent

Mod arithmetic modulus

LOGICAL OPERATORS

And performs a logical conjunction

Eqv performs a logical equivalence

Imp performs a logical implication

Not performs a logical negation

Or performs a logical disjunction

Xor performs a logical exclusion

FUNCTIONS

Microsoft vbScript has 89 functions; some examples:

```
Sin(n)   Sine of a number  
Cos(n)   Cosine of a number  
Atn(n)   ArcTangent of a number  
Log(n)   Natural logarithm of a number larger than 0  
Sqr(n)   Square root of any positive number  
Abs(n)   Absolute (positive) value of any number
```

RHINO METHODS

Rhino adds over 200 functions called methods

```
Rhino.Command ("_Move 0,0,0 2,0,1")  
Rhino.UnselectAllObjects()  
Rhino.SelectObject (strCurveID)  
Rhino.CopyObject (strCurveID)  
Rhino.GetObject("Select an object")
```

FLOW CONTROL

Lines of code can be skipped (exclusion from execution), others can be repeated (looping) and we can make jumps in code (both forwards and backwards)

CONDITIONAL EXECUTION

```
If <condition> Then  
    Statement(s)  
Else  
    Statements(s)  
End If
```

COMPARISONS

A = B A and B are equal
A <> B A and B are not equal
A > B A is greater than B
A >= B A is greater than or equal to B
A < B A is less than B
A <= B A is less than or equal to B

LOOPS

- For..Next
- Do..Loop

```
For <Numeric variable> = <StartValue> To <StopValue> [Step <StepValue>]  
    Statement(s)  
Next
```

```
For i = 1 To lngN Step 1  
    Statement(s)  
Next
```

cancel a loop at any time by using an **Exit For** command

```
Do Until <condition>  
    Statement(s)  
Loop
```

cancel a loop at any time by using an **Exit Do** command

INPUT/OUTPUT (I/O) ACTIONS

Rhino.Print (string) – to display a message in the command line area

`Rhino.MessageBox (string)` – to display a message in a message box

`Rhino.GetString (prompt)` – to obtain a string

EXECUTION COMMAND

Include `Main` at the end of the script to execute the code