PangoScript

The main purpose of “scripting” is ability to add custom relation on various events, like a MIDI, DMX, ArtNet, Channels. The script typically has a few lines, and a line has one Command. Command may have a parameters. Typically parameter is a number (constant).

Numbers

Supported float point , integer and hexadecimal numbers. Example:

- 100
- -20
- 1.01
- 0xBF

All hexadecimal numbers much have 0x prefix, similar to C language, but without ending H.
There is not strict separation on integer and float point numbers.
Separator between command parameter could be space (“ ”) or a comma (“,”)

Special characters (separators)

. ; ' : ! ￼ ^ + - * \ [ ] ( ) { } ? % | & =

Predefined constants

There are a few constants mostly for readability of the code. Each constant will be transformed to a number

- **TRUE** - numeric analog 1
- **FALSE** - numeric analog 0
- **ANY** - equal to numeric -1. Used in a few WaitFor command
- **OFF** - equal to numeric 0
- **ON** - equal to numeric 1
- **TOGGLE** - equal to numeric 2
- **AsIs** - equal to -2.

This is required for commands like “MasterPause”.

A few examples of using commands:

```
VirtualLJ off
```

```
MasterPause Toggle
```

```
WaitForMidi 0x90, 10, Any
```

Math operations (expressions)

- Standard : + - / * %
- Inversion : !
- bit OR : |
- bit AND : &
- bit XOR : ^
- bit right shift : »
- |bit left shift : «

Operators

IF

Syntax: if (expression) operator

expression - covered with brackets. Must be an expression with numerical result that gives “0” or not a “0”. So, for numerical variable it's possible to write

```
if (variable)
```

Compare operators are: >, >=, <, ⇐, =, <>. They produces numerical result 0 or 1.

You can combine comparing operations with “&” and “|” bit-wise operators (but be sure that left and right side are 0 or 1). It's recommended to cover complex operations into brackets - like

```
if ((1>2) & ((3+2)>1))
```

operator will be executed if condition gets non-zero result. If you want to place other operator after

```
if (condition) operator
```

it must be divided with ; but it's better to place next operator on the next line.

GOTO

Syntax: goto label

“label” is a name of position for a jump. The label can be placed before any operator and separated by : character. Example:

```
mylabel: WaitForBeat 4
... do something...
goto mylabel
```

Operator GOTO can work with the labels inside the string variables. It means that you declare string variable, assign a label name to variable, and then you a variable in GOTO. Example:

```
var s
s="mylabel"
goto s
DisplayPopup "It does not work"
exit

mylabel:
DisplayPopup "It works"

VAR operator
```

PangoScript allow define local variables. The lifetime for the local variable defined by lifetime of the Scripter that execute PangoScript. As soon as the Scripter freed, all its local variables are removed as well.

All variables must be declared before the using. No need to specify the type of variable. The variable automatically adjust the type depending on value. Internally supported integer, float and string variables. The declaration start from VAR operation and the follow one or more variable names. As example:

```
var MyVariable
var a,b,c
```

Before using the variable must be initialized by some value. Otherwise BEYOND will generate error and stop script execution. Example:

```
var MyInteger, MyString
MyInteger = 10
MyString = "Hello!"
```

All variables declared as VAR are local.

GLOBALVAR

The variable can be declared as global. In this case, it visible in ALL scripts of BEYOND.

General functions

intstr(value:number):string - transform number to string
value - number or float number
result - string

floatstr(value:number):string - transform number to string
value - number
result - string

abs(value:number):number - value by modulus
value - integer or float number
result - same type as argument. result is absolute value

int(value:number):integer - return integer part of float value
value - a float point number

frac(value:number):float - return fractional part of float point value
value - float

round(value:float):integer - return rounded float point value (to integer)
value - a float point number

sqr(value:number):number - return a square of argument
value - a float point number or integer

sqrt(value:number):float - return a square root of argument
value - a float point number or integer

cos(value:float):float - co-sinus
value - a float point number or integer. angle in radians

sin(value: float):float - sinus
value - a float point number or integer. angle in radians

tan(value:float):float - tangents
value - a float point number or integer. angle in radians

arcsin(value:float)float - arc sinus
value - a float point number or integer. angle in radians

arccos(value: float):float - arc cosinus
value - a float point number or integer. angle in radians

arctan(value: float):float - arc tangents
value - a float point number or integer. angle in radians

arctan2(dx, dy:float):float - arc tangents
dx, dy - ....

min(a,b:number):number - return a minimum of two numeric values
a,b float or integer

max(a,b:number):number - return a maximum of two numeric values
a,b float or integer

pi:float - return PI value - 3.1415926...

invert(value:number):integer - invert value. Boolean operation, but can be used with float point numbers. If value more than 0.5 then function return 0, otherwise return 1.

Date and Time

now:float - date&time, calls now() function of Delphi

tickcount:integer - return number of millisecond from start of PC.

hms(hour, minute, second):integer - transform hour, minute and second into seconds.

GetYear:integer - function return current year by PC clock. Result

GetMonth:integer - function return current month by PC clock

GetDay:integer - function return current day by PC clock.

timestr(now:float) :string
now - is variable representing time. Function return string with time in short format such as “11:53”, without seconds

timestrlong(now:float):string -
now - is variable representing time. Function return string with hours, minutes, seconds, such as “11:53:10”

datestr(now:float):string - ********** ******* ****, * *** *** -
now - is variable representing date. Function return string short date format such as “21.11.2012”

datestrlong(now:float):string - ******** ******* ****, **** -
now - is variable representing date. Function return string long date format such as “21 ******* 2012 *.”

dayofweek(now:float):string - short version day of the week

dayofweeklong(now:float):string - long version day of the week

String functions

uppercase(string):string - transform input string to upper case. Result is a string.

lowercase(string:string):string - transform input string to lower case. Result is a string.

crlf:string - return a string, line separator (13,10)

Clock And Metronome

b2s(beats) - transform beats to seconds

b2ms(beats) - transform beats to milliseconds

s2b(seconds) - transform seconds to beats

b2s(seconds) - transform beats to seconds

Functions

There are a few functions for access of incoming data

Dmx (hannel, OutputMin, OutputMax)
Parameters: Channel - index of DMX channel, acceptable value from 1 to 2048
OutputMin, OutputMax - defines the range resulting value of the function. Oupout value will start from OutputMin and increase up to OutputMax
Result = OutputMin + (DmxChannelValue / 255) * (OutputMax-OutputMin)
Example:

```
Position Dmx(10, -100, 100), Dmx(11, -100, 100), Dmx(12, -100, 100)
```

In this example we use Position command that has 3 arguments - X,Y,Z and we use DMX IN values (three channels, 10,11,12), and map the values range -100 to +100.

Dmx (Channel)
or
Dmx(Channel,OutputMin,OutputMax)
Parameters:
Channel - index of DMX channel, acceptable value from 1 to 2048
Result of the function is value of DMX channel as it is, without anr range adjustments
OutputMin, OutputMax - defines the range resulting value of the function. Output value will start from OutputMin and increase up to OutputMax. Internally channels are normalized to 0...1 range result = OutputMin + (ChannelValue) * (OutputMax-OutputMin) Example:

```
DisplayPopup Dmx(10) // display the value of 10th DMX channel.
```

Note: version with min/max used for simplification of use of function in operators where you may need to transform DMX value range (0..255) to some other range. As example Size operator, you may want to use size range like 0...100, or 10...100, or -100..100.

Channel (Channel, OutputMin, OutputMax) or Channel(Channel)

Parameters: Channel - index of Channel, acceptable value from 1 to 255 OutputMin, OutputMax - defines the range resulting value of the function. Output value will start from OutputMin and increase up to OutputMax. Internally channels are normalized to 0...1 range result = OutputMin + (ChannelValue) * (OutputMax-OutputMin)

If function has one argument then result is normalized channel value (range from 0 to 1)

Example:

```
Size Channel(10, 5, 100), ChannelIn(10, 5, 100), ChannelIn(10, 5, 100),
```

in this example we use Size command that has 3 arguments - X, Y,Z and we use Channel 10 value for all 3 axis.

Channel (Channel) Parameters: Channel - index of Channel, acceptable value from 1 to 255 Result of the function is channel value. Channel value is normalized, the range from 0 to 1.

CC (Channel, Controller, OutputMin, OutputMax) Get a value of of MIDI controller. BEYOND memorize all values of incoming MIDI controllers (Contrl Change Message) and you can get an access to it. Channel - value 0..15 Controller - value 0..127. Thsi is Data1 in MIDI messages OutputMin, OutputMax - defines the range resulting value. Result = OutputMin + (ChannelValue/127) * (OutputMax-OutputMin)

ExtValue(OutputMin, OutputMax) There are a few pretty big tables of code-sources. A tables such as DMX, ArtNet, ControlChange, PitchBand, NoteOn initiated by channel/message that has a value. This function allow get access to the channel that intiate the code. Better explain on example. Lets say I want to connect DMX channel to Master Live Control, Position X. In this case, code that will do ths is this:

```
  PositionIndex 1, ExtValue(-100,100)
```

In this example PositionIndex is a live control command, see below. 1 is index of axis. And finally ExtValue() is a fuction that take DMX channel value in map to rangle -100 to 100.

ExtDelta(Delta) Function equal to ExtValue but made specially for MIDI encoders. There are many controlled with wheels and knobs that generate Control Change message and the data2 value is 00 or 7F. That is all, no values in between. For such cases has a sense to use “delta” version of commands and ExtDelta() for getting exact value of delta. Function return -Delta or +Delta. Only two values, no exceptions. Example of use

PositionDelta 0, ExtDelta(1), 0 *move vertically. another example: AngleDelta 0,0, ExtDelta(45)* rotate by Z on 45 degrees.

Random(MaxValue) Function return random value in range 0..MaxValue. Note, MaxValue included into the range

RandomIn(MinValue, MaxValue) Functino return random value in range MinValue..MaxValue

Param(ParamIndex) - newer name of OscParam() function. It get the value of parameter supplied into the script. See commend about OscParam()

OscParam(ParamIndex) Specialy designed for use in “OSC to CODE” table.. When BEYOND receive OSC message and supply it into interpreter to execution, then we can supply up to 10 parameters with OSC message. BEYOND put them into local array, and you may use it inside the Code. Note, data is there only during execution of current script. As soon we exit from execution of this script, data will be lost. OK, parameter index is from 1 up to 10. BEYOND check how many parameters the OSC message has, and if you will try to access non existing paramater then script will stop with error. So, if OSC message has 3 parameters, then you can use index from 1 to 3. Accepts f, i, and s type of OSC parameters. You can freely mix float and integer.

ParamRange(MinValue, MaxValue) function return TRUE (1) if the first parameter inside the specified range.

ParamRange(ParamIndex, MinValue, MaxValue) function return TRUE (1) if the specified parameter (pram index) inside the specified range (from MinValue to MaxValue)

GetTransitionName(index) function return string, the name of transition. Index is number of transition. Range 0..23

BeatTime Function has no arguments. The result of function is float point number. Integer part of function is number of beat from the start of BEYOND. The fractional part is progress inside the beat.

GetMidiDeviceIndex function return currently selected MIDI Device pair, Value range 1..4.

ObjectExists(AObjectName) - function get object name a s string and return 1 if object exists, or zero if object is not found.