PangoScript

Here you can find different example Pangoscripts to help you see different ways scripting can be used. They also should just work if copy and pasted into BEYOND, as-is, or modified to your needs.

Pangolin Internal Examples

These examples were made by internal team members at pangolin as examples for pangoscripts

Timeline AutoSave

```
CodeName "Autosave"
TimelineQuickSave
Sleep 60000 //Wait 1 Minute
Restart
```

This code is designed to create an autosave script for your timeline, since saving does pause output, it's not recommended to leave this on for operating your show. The Sleep is in milliseconds, so 60,000 is 60 seconds, you can change the timeframe to your desired save duration, just know it does save a separate backup, so more frequent saves will eat up more storage.

Daily Schedule

```
CodeName "Daily Schedule"
Autostart
WaitForTime 19,45,0,0 //Wait for 7:45
StartCue 1,1
WaitforCueStop 1,1
StartCue 1,10
WaitforCueStop 1,10
StartCue 1,21
WaitforCueStop 1,21
StartCue 1,41
WaitforTime 23,0,0,0
StopCue 1,41
WaitForTime 23,59,59,0
Sleep 10000
RunApp "C:\Windows\System32\shutdown", "-r -t 10" // full exe file name,
parameters
//These parameters on this shutdown, sets it to "restart" then "Time 10
seconds" These options can be found on the web.
```

```
ExitBeyond
```

This code is an example for an install where you want a sequence of cues (maybe timelines saved to cues) to run at specific times in sequence, And at the end the script waits for the clock to turn over, then restarts the machine, if you put beyond in the windows startup folder, it will auto start beyond, the script and start the cycle. You will need the “RunApp” command enabled, which is in configuration, and will also need to check the box which doesn't ask for confirmation on exit of beyond in the configuration window for this to be fully automatic, an example of automation scripting.

Flip Zones

```
CodeName "Flip Zones > - <"
Zone.0.SizeX = -100
Zone.1.SizeX = -100
Zone.2.SizeX = -100
Zone.3.SizeX = -100
Zone.4.SizeX = 100
Zone.5.SizeX = 100
Zone.6.SizeX = 100
Zone.7.SizeX = 100
Master.DISPLAYPOPUPTIMEOUT=.5
displaypopup "> - <"
Exit
```

This code exploits the fact that each zone has parameters associated with it on a level that is usually not seen by the user, technically there are these parameters, and also GEO parameters, these parameters are content values, and geo is for zoning, it's important to not use the GEO ones, as that will move your zone. These just affect the content within a zone. These parameters are also the same as what BEYOND server modifies, and by using numbers of zones instead of the names, it won't matter what the zone names are, just that they are in order. It also does a popup to let you know what you have selected, something that may be useful for a midi controller trigger, where the button has no label.

Pull Tempo from DMX

```
Codename "Pull Tempo From DMX"
WaitForDmx 501 //Channel at 1 will run code
beattap
sleep 200 // Make DMX reset to zero on console less then 100ms
restart
```

This code is imperfect, but it will tap the beat tap when the channel selected “updates” any change in value, so if you have a flash button with a cue that just sets the channel to full and off when you release, it will tap the temp, the sleep is to wait for your tap on the console to reset, 200ms is enough to still get 200bpm out of your taps, and a safe spacing for the reset. You could macro on your console to automatically do this from a speed master or just use it to tap the bpm. To make this code better, you

would really want a conditional trigger, where when it hits value of 255 it taps but, this works and is much simpler of a code.

Restart Show Softly

```
Codename "Restart Show Soft"
AnimateProp "Master.AudioVolume", 100,0,5000
sleep 5000
TimelineSetPos 0
Master.AudioVolume = 100
TimelinePlay
Exit
```

This script demonstrates the ability to animate objects over time, Animate Prop goes from value1, to value 2, over value in ms. In this case, a code is written to restart a timeline, but to fade out the audio before doing so. You could animate Volume and master brightness at the same time which could be interesting too. Then resets the values by just setting their value and plays the show again.

Examples from Users

The following examples are from the userbase, at the permission of the users.

Jeff Vyduna

Knob or fader to QuickFX 2^X speeds

```
// Quantize a continuous input (fader/potentiometer) to 2^x beat scaling in
FX.

// Take a 0-127 Midi value and select a 2^i multiplier
 //  for i in [-4, 4]
// Assumes a nominal FX duration of 2 beats to work best
// by producing multipliers that result in 32 beats at
// knob minimum (MIDI 0), 2 beats at knob center,
// and 1/16 of a beat (thirtysecond notes) at MIDI 127.

// If you want to consume the current metronome multiplier, add the next
// two lines to your MIDI device initializstion script in MIDI settings.
GlobalVar gMetroAx
gMetroAx = 0

Var metroScaleVal
```

```
metroScaleVal = 0
metroScaleVal = ExtValue(0,8.999) // Scale incoming MIDI value to 0..9

// Uncomment for dubugging values to the laser preview window
// DisplayPreview "Value:", metroScaleVal, 0xffffff


var metroAx // Metronome Multiplier (doc misnomer? Seems like a divisor)

// Assumes the FX duration is set to 2 beats.
// int(metroSlider): Beat duration, multiplier
// 0-1: 32 beats, 1/16
// 1: 16 beats, .125
// 2: 8 beats, .25
// 3: 4 beats, .5
// 4: 2 beats or nominal - 1<<4 /16 = 1x multiplier;
// 5: 1 beat: multiplier of 2
// 6: 1/2 (eigth note, *8 displayed), 4
// 7: 1/4 (sixteenth note, *16 displayed), 8
// 8: 1/8 (thirth-second note, *32 displayed), 16
metroAx = max(1 << int(metroScaleVal), 1) / 16

// DisplayPreview "metroAx", metroAx, 0xffffff

gMetroAx = metroAx // Make value available to other scripts

// Uncomment which QuickFX layers to apply the new metronome multipliers to.

// FXTimeScaleAx 4, 2, metroAx  // Layer 1 = Color 1
// FXTimeScaleAx 5, 2, metroAx  // Layer 2 = Color 2
// FXTimeScaleAx 6, 2, metroAx  // Layer 3 = Zone
FXTimeScaleAx 4, 2, metroAx  // Layer 4 = Intensity, Mask (1,2,3) 2=Metronome,
multiplier
FXTimeScaleAx 5, 2, metroAx  // Layer 5 = Structure
FXTimeScaleAx 6, 2, metroAx  // Layer 6 = Movements


// Display duration / period in beats on a Kontrol F1 7-seg display
// Second argument to MidiOutLong should be 41 for the controller's unshifted
page layer,
// or 78 for the shifted display. Since I map this to a shifted page, I'm
using 78.
if (metroAx <= 2) MidiOutLong(0xBC, 78, int(2.0 / metroAx)))
if (int(metroAx) = 4) MidiOutLong(0xBC, 78, 108)
if (int(metroAx) = 8) MidiOutLong(0xBC, 78, 116)
if (int(metroAx) = 16) MidiOutLong(0xBC, 78, 132)
```

```
exit
```

Takes a fader or rotary pot on a MIDI controller, such as on a APC40mk2 or Traktor F1, and scale the global cue speed to snap to a 2^X speed multiplier (25%, 50%, 100%, 200%, 400%, etc). For example, if you've made cues with a 4 beat duration, this lets you us a MIDI control to instantly change their duration to 8, 16, or 32 beats (as well as 2, 1, 1/2, 1/4 of a beat).

Return to Guided learning