```
import maya.cmds as mc
import maya.mel as mel
from functools import partial
import os

'''
07_08_16
Auto Expression UI script.
Last worked on 08_07_14 when script was modified to be used with switch
node
so it can be used to animate transparency of surface shaders
'''


def getName():
    nodeList = mc.ls(selection=True)
    node = nodeList[0]
    channel = getChannels()
    name = "{node}.{channel}".format(node=node, channel=channel)
    # print name
    return name,node,channel


def addAttribute(node, attributeName, defaultValue=0):
    node = node
    attributeName = attributeName
    status = mc.attributeQuery(attributeName, node=node, exists=True )
    if status == False:
        mc.addAttr(node, longName=attributeName, keyable=True,
attributeType="double", defaultValue=defaultValue)
    else:
        pass

def getChannels(*args):
    getChannelBoxName = mel.eval('$temp=$gChannelBoxName')
    chList = mc.channelBox (getChannelBoxName, q=True,
selectedMainAttributes = True)
    if chList:
        for channel in chList:
            print channel
        return channel
    else:
        print 'No channels selected!'
        return ''

def getChannelList():
    #very like teh procedure above but returns alist of channels
    channleList = []

    getChannelBoxName = mel.eval('$temp=$gChannelBoxName')
    chList = mc.channelBox (getChannelBoxName, q=True,
selectedMainAttributes = True)
    if chList:
        for channel in chList:
            print channel
            channleList.append(channel)
        return channleList
    else:
```

```
        print 'No channels selected!'
        return ''

class AnimExpressionsTab:

    def __init__(self, tabLayout, tabName, tabDescription="",
multiple=False):
        self.cw1 = 100
        self.cw2 = 200

        self.tabLayout = tabLayout
        self.tabName=tabName
        self.tabDescription = tabDescription
        self.multiple = multiple

        self.newTab = mc.columnLayout(tabName, adjustableColumn=True,
rowSpacing=20, parent=tabLayout)

        self.textField_Driver = ''
        self.textField_Target = ''
        self.expressionName = ''
        self.expressionString = ''

    #Mutators

    def populateTab(self, button1='add driver', button2='add target', \
           insertText1="driverNode.driverChannel",
insertText2="targetNode.targetChannel"):
        buttonWidth = 150
        row1 = mc.rowLayout('row1', parent=self.newTab)
        mc.text(self.tabDescription, parent=row1)
        row2 = mc.rowLayout('row2', numberOfColumns=2,
columnWidth2=(self.cw1, self.cw2), parent=self.newTab)
        self.textField_Driver = mc.textField("textField_Driver", w=200,
insertText=insertText1)
        mc.button(button1, w=buttonWidth,
command=(partial(self.populateDriverText, self.textField_Driver)))

        row3 = mc.rowLayout('row3', numberOfColumns=2,
columnWidth2=(self.cw1, self.cw2), parent=self.newTab)
        self.textField_Target = mc.textField("textField_Target", w=200,
insertText=insertText2)
        mc.button(button2, w=buttonWidth,
command=(partial(self.populateTargetText, self.textField_Target)))

    def populateMultiChannelTab(self, ):
        buttonWidth = 150
        buttonHeight = 30

        row1 = mc.rowLayout('row1', parent=self.newTab)
        mc.text(self.tabDescription, parent=row1)

        row2 = mc.rowLayout('row2', numberOfColumns=2,
columnWidth2=(self.cw1, self.cw2), parent=self.newTab)
        self.textField_Driver = mc.textField("textField_Driver", w=200,
insertText="driverNode.driverChannel")
        mc.button('add driver', w=buttonWidth,
command=(partial(self.populateDriverText, self.textField_Driver)))
```

```
        row3 = mc.rowLayout('row3', numberOfColumns=2,
columnWidth2=(self.cw1, self.cw2), parent=self.newTab)
        self.textField_Target = mc.textField("textField_Target", w=200,
insertText="targetNode")
        mc.button('add target', w=buttonWidth,
command=(partial(self.populateTargetMultiChannel,
self.textField_Target)))

        row4 = mc.rowLayout('row4', numberOfColumns=2,
columnWidth2=(self.cw1, self.cw2), parent=self.newTab)
        self.textField_Channels = mc.textField("textField_Channels",
w=200, insertText="channel,channel")
        mc.button('add target channels', w=buttonWidth,
command=(partial(self.populateChannelsMultiChannel,
self.textField_Channels)))

        row5 = mc.rowLayout('row5', parent=self.newTab)
        self.addTextFieldExpressionName()

        row6 = mc.rowLayout('row6', parent=self.newTab)
        creationButton = mc.button(label="create expressions",
backgroundColor=(0.75,0.4,0.4),\
             w=buttonWidth, h=buttonHeight,
command=(partial(self.createMultiChannelExpressions)))


    def addTextFieldMultiTargetChannel(self, insertText='targetChannel',
instruction=' enter target channel'):
        row = mc.rowLayout(numberOfColumns=2, columnWidth2=(self.cw1,
self.cw2), parent=self.newTab)
        self.textFieldMultiTargetChannel =
mc.textField('textFieldMultiTargetChannel', w=200, insertText=insertText)
        mc.text(instruction)

    def addTextFieldExpressionName(self, insertText='expressionName',
instruction='add expression name', *args):
        row = mc.rowLayout(numberOfColumns=2, columnWidth2=(self.cw1,
self.cw2), parent=self.newTab)
        self.textFieldExpressionName =
mc.textField('textFieldExpressionName', w=200, insertText=insertText)
        mc.text(instruction)

    def addButton(self, buttonName='create expressions', *args):
        buttonWidth = 150
        buttonHeight = 40
        row = mc.rowLayout(numberOfColumns=1, parent=self.newTab)
        if self.multiple==False:
            creationButton = mc.button(label="create expression",
backgroundColor=(0.75,0.4,0.4),\
             w=buttonWidth, h=buttonHeight,
command=(partial(self.createSingleTargetExpression)))
        else:
            creationButton = mc.button(label="create expressions",
backgroundColor=(0.75,0.4,0.4),\
             w=buttonWidth, h=buttonHeight,
command=(partial(self.createMultiTargetExpressions)))
```

```
    def populateDriverText(self, textField, *args):
        nodeList = mc.ls(orderedSelection=True)
        driver = nodeList[0]
        channel = getChannels()
        mc.textField(str(textField), edit=True,
text="{driver}.{channel}"\
            .format(driver=driver, channel=channel))
        ########### multiChannel stuff
    def populateTargetMultiChannel(self, textField, *args):
        node = mc.ls(selection=True)
        target = node[0]
        # channel = getChannels()
        mc.textField(str(textField), edit=True, text="{target}"\
            .format(target=target))

    def populateChannelsMultiChannel(self, textField, *args):
        print 'populateChannelsMultiChannel'
        # node = mc.ls(selection=True)
        # target = node[0]
        channelList = getChannelList()

        for channel in channelList:
            mc.textField(str(textField), edit=True,
insertText="{channel}"\
                .format(channel=channel))
            mc.textField(str(textField), edit=True, insertText=",")
        ################### END multiChannel stuff


    def createMultiChannelExpressions(self, *args):
        #get driver and target and channels
        textFieldDriver = self.textField_Driver
        textFieldTarget = self.textField_Target
        textFieldChannels = self.textField_Channels
        textFieldExprName = self.textFieldExpressionName

        #get driver node and channel
        driverNode = (self.getNodes(textFieldDriver))[0]
        driverChannel = (self.getNodes(textFieldDriver))[1]

        targetNode = self.getText(textFieldTarget)
        print driverNode, driverChannel, targetNode

        targetChannels = self.getChannels(textFieldChannels)
        numOfChannels = len(targetChannels)
        print targetChannels, numOfChannels
        exprName = self.getText(textFieldExprName)

        ### add delay, multiplier, offset, and ampIncrease attributes to
driver node
        addAttribute(driverNode, "childDelay", defaultValue=0)
        addAttribute(driverNode, "childMultiplier", defaultValue=1)
        addAttribute(driverNode, "ampIncrease", defaultValue=0)
        #### NG. DONT add offset attributes to driver node

        ## create expressions driving each of the target nodes
        channelNumber = 1
```

```
        for channel in targetChannels:
            index = channelNumber - 1
            print index

            exprString = "$multiplier =
%(driverNode)s.childMultiplier;\n\
            $delay = -1 * (%(driverNode)s.childDelay *
%(channelNumber)d);\n\
            $time = frame + $delay;\n\
            $value =`getAttr -time($time)
%(driverNode)s.%(driverChannel)s`;\n\
            $ampIncrease = %(driverNode)s.ampIncrease;\n\
            $increaseAmmnt = $ampIncrease * (%(numOfChannels)d-
%(index)d);\n\
            if ($ampIncrease == 0)\n\
            {\n\
            $increaseAmmnt = 1;\n\
            }\n\
            if($ampIncrease > 0)\n\
            {\n\
            $increaseAmmnt = $ampIncrease * %(channelNumber)d;\n\
            }\n\
            %(targetNode)s.%(channel)s = ($value * $multiplier) *
$increaseAmmnt;"\
            %{'driverNode': driverNode, 'targetNode' : targetNode, \
            'channelNumber': channelNumber, 'numOfChannels':
numOfChannels, 'index': index, \
            'driverChannel': driverChannel, 'channel': channel}

            expression = mc.expression(object=channel, string=exprString,
name=exprName)
            channelNumber +=1



    def populateTargetText(self, textField, *args):
        nodeList = mc.ls(orderedSelection=True)

        if self.multiple == True:
            targetNodes = mc.ls(orderedSelection=True)
            for node in nodeList:
                text = node
                mc.textField(str(textField), edit=True, insertText=text +
",")
                #and add target channel

self.populateTargetChannel(self.textFieldMultiTargetChannel)
        else:
            driver = nodeList[0]
            channel = getChannels()
            mc.textField(str(textField), edit=True,
text="{driver}.{channel}"\
                .format(driver=driver, channel=channel))

    def populateTargetChannel(self, textField, *args):
        channel = getChannels()
        mc.textField(str(textField), edit=True, text="{channel}"\
            .format(channel=channel))
```

```
    #Accessors
    def getText(self, textField, *args):
        textOut = mc.textField(textField, query=True, text=True)
        return textOut

    def getNodes(self, textField, *args):
        #use this function for multi and single driver nodes and also
single target nodes
        nodes = self.getText(str(textField))
        nodeList = nodes.split('.')
        return nodeList

    def getChannels(self, textField, *args):
        channels = self.getText(str(textField))
        channelList = channels.split(',')
        ### get rid of the last list entry (which is an empty string)
        channelList.remove(channelList[-1])
        return channelList

    def getMultiTargetNodes(self, textField, *args):
        targetNodes = self.getText(str(textField))
        targetNodeList = targetNodes.split(',')
        # print targetNodeList
        return targetNodeList

    ## create expressions
    def createSingleTargetExpression(self, *args):
        textFieldDriver = self.textField_Driver
        textFieldTarget = self.textField_Target
        textFieldExprName = self.textFieldExpressionName

        driverNode = (self.getNodes(str(textFieldDriver)))[0]
        driverChannel = (self.getNodes(str(textFieldDriver)))[1]
        targetNode = (self.getNodes(str(textFieldTarget)))[0]
        targetChannel = (self.getNodes(str(textFieldTarget)))[1]
        exprName = self.getText(str(textFieldExprName))
        ## add delay, multiplier, offset attributes to target node
        ##??? how would I store or change the default values??
        addAttribute(targetNode, "delay", defaultValue=0)
        addAttribute(targetNode, "multiplier", defaultValue=1)
        addAttribute(targetNode, "offset", defaultValue=0)
        exprString = "$offset = {targetNode}.offset;\n\
        $multiplier = {targetNode}.multiplier;\n\
        $delay = -1 * {targetNode}.delay;\n\
        $time = frame + $delay;\n\
        $value =`getAttr -time($time) {driverNode}.{driverChannel}`;\n\
        {targetNode}.{targetChannel} = ($value * $multiplier) +$offset;"\
        .format(targetNode=targetNode, targetChannel=targetChannel,
driverNode=driverNode, driverChannel=driverChannel)
        # create expression
        expression = mc.expression(object=targetNode, string=exprString,
name=exprName)

    def createMultiTargetExpressions(self, *args):
        textFieldDriver = self.textField_Driver
        textFieldTarget = self.textField_Target
        textFieldMultiTargetChannel = self.textFieldMultiTargetChannel
```

```
        textFieldExprName = self.textFieldExpressionName

        driverNode = (self.getNodes(str(textFieldDriver)))[0]
        # print driverNode
        driverChannel = (self.getNodes(str(textFieldDriver)))[1]
        targetNodes = self.getMultiTargetNodes(str(textFieldTarget))
        noOfTargetNodes = (len(targetNodes)) - 1
        targetChannel = self.getText(str(textFieldMultiTargetChannel))
        exprName = self.getText(str(textFieldExprName))
        ### add delay, multiplier, offset, and ampIncrease attributes to
driver node
        addAttribute(driverNode, "childDelay", defaultValue=0)
        addAttribute(driverNode, "childMultiplier", defaultValue=1)
        addAttribute(driverNode, "ampIncrease", defaultValue=0)
        #### add offset attributes to driver node
        index = 1
        for targetNode in range(noOfTargetNodes):
            attrName = "offset{index}".format(index=index)
            status = mc.attributeQuery(attrName, node=driverNode,
exists=True)
            if status == False:
                mc.addAttr(driverNode, longName=attrName, keyable=True,
attributeType="double", min=-100, max=100)
            else:
                pass
            index += 1
        ## create expressions driving each of the target nodes
        nodeNumber = 1
        for targetNode in range(noOfTargetNodes):
            nodeIndex = nodeNumber - 1
            print nodeIndex
            print type(nodeIndex)
            targetNode = targetNodes[nodeIndex]
            name = str(targetNode)
            driverOffset = "offset" + str(nodeNumber)
            exprString = "$offset = %(driverNode)s.%(driverOffset)s;\n\
            $multiplier = %(driverNode)s.childMultiplier;\n\
            $delay = -1 * (%(driverNode)s.childDelay *
%(nodeNumber)d);\n\
            $time = frame + $delay;\n\
            $value =`getAttr -time($time)
%(driverNode)s.%(driverChannel)s`;\n\
            $ampIncrease = %(driverNode)s.ampIncrease;\n\
            $increaseAmmnt = $ampIncrease * (%(noOfTargetNodes)d-
%(nodeIndex)d);\n\
            if ($ampIncrease == 0)\n\
            {\n\
            $increaseAmmnt = 1;\n\
            }\n\
            if($ampIncrease > 0)\n\
            {\n\
            $increaseAmmnt = $ampIncrease * %(nodeNumber)d;\n\
            }\n\
            %(targetNode)s.%(targetChannel)s = (($value * $multiplier) *
$increaseAmmnt) + $offset;"\
            %{'driverNode': driverNode, 'driverOffset': driverOffset, \
            'nodeNumber': nodeNumber, 'noOfTargetNodes': noOfTargetNodes,
'nodeIndex': nodeIndex, \
```

```
            'driverChannel': driverChannel, 'targetNode': targetNode,
'targetChannel': targetChannel}

            expression = mc.expression(object=targetNode,
string=exprString, name=exprName)
            nodeNumber +=1

#####
        textFieldChannels = self.textFieldChannels
        textFieldExprName = self.textFieldExpressionName

        driverNode = (self.getNodes(str(textFieldDriver)))[0]
        # print driverNode
        driverChannel = (self.getNodes(str(textFieldDriver)))[1]
        targetNodes = self.getMultiTargetNodes(str(textFieldTarget))
        noOfTargetNodes = (len(targetNodes)) - 1
        targetChannel = self.getText(str(textFieldMultiTargetChannel))
        exprName = self.getText(str(textFieldExprName))
        ### add delay, multiplier, offset, and ampIncrease attributes to
driver node
        addAttribute(driverNode, "childDelay", defaultValue=0)
        addAttribute(driverNode, "childMultiplier", defaultValue=1)
        addAttribute(driverNode, "ampIncrease", defaultValue=0)
        #### add offset attributes to driver node
        index = 1
        for targetNode in range(noOfTargetNodes):
            attrName = "offset{index}".format(index=index)
            status = mc.attributeQuery(attrName, node=driverNode,
exists=True)
            if status == False:
                mc.addAttr(driverNode, longName=attrName, keyable=True,
attributeType="double", min=-100, max=100)
            else:
                pass
            index += 1
        ## create expressions driving each of the target nodes
        nodeNumber = 1
        for targetNode in range(noOfTargetNodes):
            nodeIndex = nodeNumber - 1
            print nodeIndex
            print type(nodeIndex)
            targetNode = targetNodes[nodeIndex]
            name = str(targetNode)
            driverOffset = "offset" + str(nodeNumber)
            exprString = "$offset = %(driverNode)s.%(driverOffset)s;\n\
            $multiplier = %(driverNode)s.childMultiplier;\n\
            $delay = -1 * (%(driverNode)s.childDelay *
%(nodeNumber)d);\n\
            $time = frame + $delay;\n\
            $value =`getAttr -time($time)
%(driverNode)s.%(driverChannel)s`;\n\
            $ampIncrease = %(driverNode)s.ampIncrease;\n\
            $increaseAmmnt = $ampIncrease * (%(noOfTargetNodes)d-
%(nodeIndex)d);\n\
            if ($ampIncrease == 0)\n\
            {\n\
            $increaseAmmnt = 1;\n\
            }\n\
```

```
            if($ampIncrease > 0)\n\
            {\n\
            $increaseAmmnt = $ampIncrease * %(nodeNumber)d;\n\
            }\n\
            %(targetNode)s.%(targetChannel)s = (($value * $multiplier) *
$increaseAmmnt) + $offset;"\
            %{'driverNode': driverNode, 'driverOffset': driverOffset, \
            'nodeNumber': nodeNumber, 'noOfTargetNodes': noOfTargetNodes,
'nodeIndex': nodeIndex, \
            'driverChannel': driverChannel, 'targetNode': targetNode,
'targetChannel': targetChannel}

            expression = mc.expression(object=targetNode,
string=exprString, name=exprName)
            nodeNumber +=1


def launchUI():
    if mc.window('animationUI', query=True, exists=True):
        mc.deleteUI('animationUI')
    myWindow = mc.window('animationUI', title='Animation Tools', w=100,
h=300)

    buttonWidth = 150
    buttonHeight = 30
    columnWidth = 200
    colour = (0.60,0.4,0.4)
    column1 = 50
    column2 = 100

    columnLayout = mc.columnLayout(rowSpacing=5, adjustableColumn=True)

    tabs = mc.tabLayout(innerMarginWidth=5, innerMarginHeight=5,
parent=columnLayout)

    tab1 = AnimExpressionsTab(tabLayout=tabs, tabName='Single Target',
tabDescription=\
        "This script adds Delay, Multiplier, and Offset attributes\
to the target node. \nThese attributes can be used to adjust (or animate)
\nthe\
relationship between driver and target nodes.", multiple=False)
    tab1.populateTab()
    tab1.addTextFieldExpressionName()
    tab1.addButton()

    tab2 = AnimExpressionsTab(tabLayout=tabs, tabName='Multiple Targets',
tabDescription=\
        "This script adds Child delay, Child multiplier,\
 and multiple Offset attributes to the driver node. \nThese attributes
can be used to adjust (or animate) \nthe\
 relationship between the driver and its multiple target nodes.",
multiple=True)
    tab2.populateTab(button2='add targets', insertText2="child1,child2")
    tab2.addTextFieldMultiTargetChannel()
    tab2.addTextFieldExpressionName()
    tab2.addButton()
```

```
    tab3 = AnimExpressionsTab(tabLayout=tabs, tabName='Multiple
Channels', tabDescription=\
        "This script adds Child delay, Child multiplier,\
 to the driver node. \nThese attributes can be used to adjust (or
animate) \nthe\
 relationship between the driver and multiple channels on a single target
node.", multiple=True)
    tab3.populateMultiChannelTab()
    # tab3.addTextFieldMultiTargetChannel()
    # tab3.addTextFieldExpressionName()
    # tab3.addButton()



    mc.showWindow(myWindow)


launchUI()
```