```
import maya.cmds as mc
from functools import partial
from operator import itemgetter

'''
18_07_14
does the vis key rely on mesh number???
17_07_14
-- make the first mesh be 0 on frame -1 and 1 on frame 0
'''
def keyMeshVisibility(meshName='meshName', meshNumber=0, keySpacing=4,
visTrail=4):
    # frameNumber = int(meshNumber)
    frameNumber = meshNumber
    print '>>>> frameNumber == ', frameNumber

    #set 0 vis at frame 0
    mc.setKeyframe( "{meshName}.v".format(meshName = meshName), time = -
1, value = 0)

    # #set first keyframe to visible
    firstKey = frameNumber * keySpacing
    mc.setKeyframe( "{meshName}.v".format(meshName = meshName),
time=firstKey, value=1)

    #set secondKey to off
    secondKey = firstKey + keySpacing + visTrail
    mc.setKeyframe( "{meshName}.v".format(meshName = meshName), time =
secondKey, value = 0)

    return meshNumber

def orderList(nodeList):
    extendedNodeList = []

    for node in nodeList:
        nodenameList = node.split('_')

        if nodenameList[-1] == '':
            nodeNumber = 0
        else:
            nodeNumber = int(nodenameList[-1])

        extendedNodeList.append([node,nodeNumber])

    newExtendedNodeList = sorted(extendedNodeList, key=itemgetter(1))
    print 'newExtendedNodeList == ', newExtendedNodeList

    newNodeList = []
    for item in newExtendedNodeList:
        newNodeList.append(item[0])

    print 'newNodeList == ', newNodeList
    return newNodeList



class CreateVisKeysUI(object):
```

```
    def __init__(self, parentLayout, workOnTransformNodeStatus=False):
        self.parentLayout=parentLayout
        self.workOnTransformNodeStatus=workOnTransformNodeStatus

        self.workOnChildNodes = True
        # self.blendOriginIsWorld = True

        self.buildUI()

    def buildUI(self):
        print 'temp test statement'
        clm1 = 125
        clm2 = 75
        clm3 = 100
        separation = 15

        #### create text button command according to
workOnTransformNodeStatus
        if self.workOnTransformNodeStatus == True:
            mc.text('- create visibility keys on Transform nodes',
parent=self.parentLayout)
            # createVisKeyCommand = self.keyTransformVis()
        else:
            mc.text('- create visibility keys on Shape nodes',
parent=self.parentLayout)
            # createVisKeyCommand = 'mc.polyCube()'

        self.radioCollection = mc.radioCollection()
        RBlayout = mc.rowLayout(numberOfColumns=2,
parent=self.parentLayout)
        self.childrenRB = mc.radioButton(label='Work on child nodes',
parent=RBlayout, select=True,
onCommand=(partial(self.setWorkOnChildNodeStatus, True)))
        self.selectedNodesRB = mc.radioButton(label='Work on selected
nodes', parent=RBlayout,
onCommand=(partial(self.setWorkOnChildNodeStatus, False)))

        inputLayout = mc.rowLayout(numberOfColumns=4,
parent=self.parentLayout)
        mc.text('    Key spacing')
        self.keySpacingTF = mc.textField('keySpacingTF',
parent=inputLayout, width=25, text='10')
        mc.text('    Trail size')
        self.visTrailTF = mc.textField('visTrailTF', parent=inputLayout,
width=25, text='10')

        visLayout = mc.rowLayout(numberOfColumns=4,
parent=self.parentLayout)

        if self.workOnTransformNodeStatus == True:
            mc.button(label = 'Create vis keys', width=clm1,
command=self.keyTransformVis,\
            parent=visLayout)
            mc.button(label = 'Delete keys', width=clm2,
command=self.deleteTransformVisKeys,\
            parent=visLayout)
        else:
```

```
            mc.button(label = 'Create vis keys', width=clm1,
command=self.keyShapeVis,\
            parent=visLayout)
            mc.button(label = 'Delete keys', width=clm2,
command=self.deleteShapeVisKeys,\
            parent=visLayout)

        separator = mc.separator(style ="none", w=300, h=10,
parent=self.parentLayout)



    def setWorkOnChildNodeStatus(self, status, *args):
        self.workOnChildNodes = status


    def deleteTransformVisKeys(self, *args):
        #deletes visibility keys on selected Node and set vis to 1
        if self.workOnChildNodes == True:
            groupNode = mc.ls(selection=True)
            tranformNodeList = mc.listRelatives(groupNode, children=True,
type = 'transform')
            nodeList = tranformNodeList
        else:
            nodeList = mc.ls(selection=True)

        i = 0
        for index in range(len(nodeList)):
            node = nodeList[i]

            #check if there is keys on vis and delete if yes
            visKeys = mc.keyframe('{node}.v'.format(node=node),
query=True)
            if visKeys != None:
                print 'deleting existing keys'
                mc.cutKey('{node}'.format(node=node),
attribute='visibility', clear=True)

            ##set node vis to 1
            mc.setAttr('{node}.v'.format(node=node), 1)

            i += 1

    def deleteShapeVisKeys(self, *args):
        if self.workOnChildNodes == True:
            groupNode = mc.ls(selection=True)
            shapeNodeList = mc.listRelatives(groupNode, children=True,
type = 'shape')
            nodeList = shapeNodeList
        else:
            nodeList = mc.ls(selection=True)

        i = 0
        for index in range(len(nodeList)):
            node = nodeList[i]
            #check if there is keys on vis and delete if yes
            visKeys = mc.keyframe('{node}.v'.format(node=node),
query=True)
```

```
            if visKeys != None:
                mc.cutKey('{node}'.format(node=node),
attribute='visibility', clear=True)

            ##set node vis to 1
            mc.setAttr('{node}.v'.format(node=node), 1)

            i += 1

        print '>>> deleted existing keys on nodeList ', nodeList

    def getNodeList(self):
        if self.workOnChildNodes == True:
            groupNode = mc.ls(selection=True)
            tranformNodeList = mc.listRelatives(groupNode, children=True,
type = 'transform')
            shapeNodeList = mc.listRelatives(groupNode, children=True,
type = 'shape')
            # print 'tranformNodeList, shapeNodeList', tranformNodeList,
shapeNodeList

            if self.workOnTransformNodeStatus == True:
                unorderedNodeList = tranformNodeList
            else:
                unorderedNodeList = shapeNodeList
        else:
            unorderedNodeList = mc.ls(selection=True)

        #### get rid of mesh_Orig and make new list ##########
        newList = []
        checkString1 = "Orig"
        checkString2 = "polySurfaceShape"
        for node in unorderedNodeList:
            # print node
            if checkString1 in node or checkString2 in node:
                print '>>> removing ', node
            else:
                newList.append(node)

        nodeList = orderList(newList)
        return nodeList


    def keyTransformVis(self, *args):
        print 'TESTING>>>>>>>>'
        ### if there are vis keys delete them
        keySpacing = int(mc.textField(self.keySpacingTF, query=True,
text=True))
        visTrail = int(mc.textField(self.visTrailTF, query=True,
text=True))

        # print 'visKeySpacing is ', keySpacing
        ############ replace with purpose built function
        # if self.workOnChildNodes == True:
        #     groupNode = mc.ls(selection=True)
        #     tranformNodeList = mc.listRelatives(groupNode,
children=True, type = 'transform')
        #     unorderedNodeList = tranformNodeList
```

```
        # else:
        #     unorderedNodeList = mc.ls(selection=True)
        ##############

        nodeList = self.getNodeList()

        # Loop to set keys on model vis
        i = 0
        for node in nodeList:
            #check if there is keys on vis and delete if yes
            visKeys = mc.keyframe('{node}.v'.format(node=node),
query=True)
            if visKeys != None:
                print 'deleting existing keys'
                mc.cutKey('{node}'.format(node=node),
attribute='visibility', clear=True)
            keyMeshVisibility(meshName=node, meshNumber=i,
keySpacing=keySpacing, visTrail=visTrail)
            i += 1

    def keyShapeVis(self, *args):
        keySpacing = int(mc.textField(self.keySpacingTF, query=True,
text=True))
        visTrail = int(mc.textField(self.visTrailTF, query=True,
text=True))

        # if self.workOnChildNodes == True:
        #     groupNode = mc.ls(selection=True)
        #     shapeNodeList = mc.listRelatives(groupNode, children=True,
type = 'shape')
        #     unorderedNodeList = shapeNodeList
        # else:
        #     unorderedNodeList = mc.ls(selection=True)
        # nodeList = orderList(unorderedNodeList)

        nodeList = self.getNodeList()

        # Loop to set keys on model vis
        i = 0
        for node in nodeList:
            #check if there is keys on vis and delete if yes
            visKeys = mc.keyframe('{node}.v'.format(node=node),
query=True)
            if visKeys != None:
                #### delete existing keys
                mc.cutKey('{node}'.format(node=node),
attribute='visibility', clear=True)

            keyMeshVisibility(meshName=node, meshNumber=i,
keySpacing=keySpacing, visTrail=visTrail)
            i += 1
```