```
import maya.cmds as mc
import maya.OpenMaya as om
from functools import partial
import re
'''
04_09_14
add exception for indexerror
--> add 0 to Mesh0 name
-- it still fails on dupe shape when something like a camera is selected
:/

'''
############ HELPERS
def getMObjectFromNode(mayaNode):
    ''' Get an MObject from a node name in Maya '''
    if not isinstance(mayaNode, basestring):
        print('Please pass in a string value')
        return None
    # Create a selection list object and add the node
    selectionList = om.MSelectionList()
    selectionList.add(mayaNode)
    # Grab the node object from the selection
    result = om.MObject()
    selectionList.getDependNode(0, result)
    # Return the MObject
    return result

def getNodeName(mObject):
    ''' Notice that this method takes an mObject as an input, not a maya
node name '''
    if mObject.hasFn(om.MFn.kDagNode):
        name = om.MFnDagNode(mObject).fullPathName()
    else:
        name = om.MFnDependencyNode(mObject).name()

    return name

def duplicate(mayaNode, *args, **kwds):
    ''' Duplicate the specified node. '''
    mObject = getMObjectFromNode(mayaNode)
    duplicateNode = om.MFnDagNode(mObject).duplicate(*args, **kwds)
    return duplicateNode
############################

def keyMeshVisibility(meshName=0, keySpacing=4):
    #give meshNumber a precautionary start value
    meshNumber=0

    keySpacing = int(keySpacing)
    ######### get meshNumber from the mesh name
    meshNameList = meshName.split('_')
    meshNumber = meshNameList[-1]
    if meshNumber == '':
        meshNumber = '0'
    else:
        meshNumber = meshNumber

    frameNumber = int(meshNumber)
```

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

    return meshNumber

def makeDisplayLayer(objectToAdd=0, layerName='displayLayer', colour=29):
    # nodeToAdd = getNodeName(objectToAdd)
    #check for existence of models layer and add make it if not there
    layerTest = mc.objExists(layerName)
    if layerTest != 1:
        mc.createDisplayLayer(name=layerName, empty=True)
        print 'just made display layer >>', layerName
    #add object to layer
    if objectToAdd == 0:
        print "no objects added to new layer"
    else:
        mc.editDisplayLayerMembers(layerName, objectToAdd,
noRecurse=True)
    # set visibility and colour
    mc.setAttr('{layerName}.v'.format(layerName=layerName), 0)
    mc.setAttr('{layerName}.color'.format(layerName=layerName), colour)
    mc.setAttr('{layerName}.v'.format(layerName=layerName), 0)

def makeGroup(objectToAdd=0, groupName='newGroup'):
    #check for existence of duplicateModels group and make it if not
there
    groupTest = mc.objExists(groupName)

    if groupTest != 1:
        newGroup = mc.group(empty=True, name=groupName)
        print 'I just made {groupName} group'.format(groupName=groupName)


    if objectToAdd == 0:
        print "no objects added to new group"
    else:
        mc.parent( objectToAdd, groupName)

def getMeshNumber(meshName):
    #### until I work out how to send meshNumber to the function use
meshNumber
    meshNumber = ''.join(x for x in meshName if x.isdigit())
    return meshNumber
##########################################################################
################################
```

```
class DuplicateModel(object):

    def __init__(self, parentLayout, meshName='mesh', keySpacing='10',
manual=True, duplicateShapeStatus=False):
        # parentLayout=mainColumnTab1, manual=True, shape=True
        self.meshName = meshName
        self.keySpacing = keySpacing
        self.parentLayout = parentLayout
        self.manual=manual
        ### set an initial value for currentMeshNumber
        self.currentMeshNumber = 0

        ### sets whether we are working on shape nodes or transform nodes
        self.duplicateShapeStatus = duplicateShapeStatus

        self.buildUI()

    def buildUI(self):
        # print 'ui build'
        parentLayout= self.parentLayout

        clm1 = 70
        clm2 = 70
        clm3 = 110

        self.labelNameRow = mc.rowLayout(numberOfColumns=2,
columnWidth2=[(clm1 + clm2), clm3], parent=parentLayout)
        mc.text('            Function name :', parent=self.labelNameRow)
        if self.duplicateShapeStatus == True:
            mc.text('duplicateShape (duplicates and keyframes shape
node)', parent=self.labelNameRow)
        else:
            mc.text('duplicateTransform (duplicates and keyframes shape
node)', parent=self.labelNameRow)

        self.meshNameRow = mc.rowLayout(numberOfColumns=2,
columnWidth2=[(clm1 + clm2), clm3], parent=parentLayout)
        mc.text('                  Mesh name :', parent=self.meshNameRow,
align='right')

        meshNameTF = mc.textField('meshNameTF', text='mesh', width=clm3,
parent=self.meshNameRow)

        self.keySpacingRow = mc.rowLayout(numberOfColumns=3,
columnWidth3=[(clm1 + clm2), (clm3*0.5), (clm3*0.5)],
parent=parentLayout)
        mc.text('                Key spacing :', align='right',
parent=self.keySpacingRow)
        meshNameTF = mc.textField('keySpacingTF', text=self.keySpacing,
width=clm3, parent=self.keySpacingRow)

        duplicateModelButton = mc.button(label='execute
function',command=self.executeFunction, parent=self.keySpacingRow)

        ### button for simple execution
        # if self.manual == True:
        #     if self.duplicateShapeStatus == True:
```

```
        #         duplicateModelButton = mc.button(label='execute
function',command=self.duplicateShape, parent=self.keySpacingRow)
        #     else:
        #         duplicateModelButton = mc.button(label='execute
function',command=self.duplicateTransform, parent=self.keySpacingRow)

        ## feedback line
        self.feedbackRow = mc.rowLayout(numberOfColumns=2,
columnWidth2=[(clm1 + clm2), clm3], parent=parentLayout)
        self.feedbackLabel = mc.text('           Script feedback : ',
parent=self.feedbackRow)
        # self.feedbackTF = mc.textField( 'feedbackTF', text='script
feedback here', parent=self.feedbackRow)
        self.feedbackT = mc.text( label='feedbackTF',
parent=self.feedbackRow)

    def executeFunction(self, *args):
        print 'TEMP'
        if self.duplicateShapeStatus == True:
            functionReturn = self.duplicateShape()
        else:
            functionReturn = self.duplicateTransform()
        print 'function return value is ', functionReturn
        return functionReturn

    def duplicateShape(self, *args):
        mesh1Node = None
        meshNumber = None
        feedback = ''
        self.meshName = mc.textField('meshNameTF', query=True, text=True)
        keySpacing = mc.textField('keySpacingTF', query=True, text=True)

        meshName = '{meshName}_'.format(meshName=self.meshName)
        # keySpacing = self.keySpacing

        #### added 04_09_14
        try:
            currentSelection = mc.ls(selection = True)
        except IndexError:
            print ">>> nothing selected"
            currentSelection = []
            return None

        print 'currentSelection  ', currentSelection

        # print currentSelection
        #if nothing is selected
        # if currentSelection == []:
        #     print  '   >>>NONE selected'
        #     # feedback = '   >>>nothing selected'
        #     return None

        if currentSelection != []:
            ### in case there is more than one component selected ...
            currentNode = currentSelection[0]
            nodeType = mc.nodeType(currentNode)

            #test for if something strange is selected or nothing at all
```

```
            if nodeType != 'mesh' and nodeType != 'transform':
                feedback = '   >>>neither mesh or transform selected'

            elif nodeType == None:
                feedback = '   >>>nothing selected'

            elif nodeType == 'mesh' or nodeType == 'transform':
                ###each of the following duplicates the shape node but it
also makes a transform
                if nodeType == 'mesh':
                    feedback += '>>>mesh selected'
                    selectedShapeNode = mc.ls(selection=True,
shapes=True, dag=True, objectsOnly=True, long=True)
                    print 'selectedShapeNode == ', selectedShapeNode
                    mesh1Node = selectedShapeNode[0]

                if nodeType == 'transform':
                    feedback += '>>>transform selected '
                    mesh1Transform = mc.ls(selection=True, dag=True,
objectsOnly=True, long=True)
                    selectedShapeNode = mc.listRelatives(mesh1Transform,
children=True, fullPath=True, type = 'mesh')
                    if selectedShapeNode == None:
                        print 'something odd selected'
                    else:
                        print 'selectedShapeNode == ', selectedShapeNode
                        mesh1Node = selectedShapeNode[0]

                if mesh1Node != None:
                    # #########  get the pointer using the api
                    mesh1Node = mesh1Node
                    mesh1Object = getMObjectFromNode(mesh1Node)
                    ### get the transform node from selected node
                    mesh1Transform = mc.listRelatives(mesh1Node,
allParents=True, fullPath=True, type = 'transform')

                    # # duplicate shape object - this parents it under a
transform
###### delete history on the transform node
                    mc.delete(mesh1Transform, constructionHistory=True)
                    mesh2Transform = mc.duplicate(mesh1Node)
                    ### get teh new transform
                    mesh2Node = mc.listRelatives(mesh2Transform,
fullPath=True, type = 'mesh')[0]
                    mesh2Object = getMObjectFromNode(mesh2Node)

                    ## make a new transform if it doesnt exist
                    newTransformName =
'{meshName}Transform'.format(meshName=meshName)
                    if mc.objExists(newTransformName) == False:
                        newTransform = mc.createNode('transform', name =
newTransformName)
                    else:
                        newTransform = newTransformName
                    makeDisplayLayer(newTransform)

                    ## rename the new mesh node
                    renamedMesh2Node = mc.rename(mesh2Node, meshName)
```

```
                    # ## reparent the shape node
                    mc.parent(renamedMesh2Node, newTransform, shape=True,
relative=True)
                    mc.delete(mesh2Transform)

                    ### get the name of the newly parented mesh2 object
and key vis
                    newlyParentedMesh2Node = getNodeName(mesh2Object)
                    meshNumber =
keyMeshVisibility(newlyParentedMesh2Node, keySpacing=keySpacing )

                feedback += '>>>New mesh number is
{meshNumber}'.format(meshNumber=meshNumber)

                mc.select(currentSelection)

        mc.text(self.feedbackT, edit=True, label=feedback)

        print feedback
        return self.meshName, meshNumber

    def duplicateTransform(self, *args):
        meshNumber = self.currentMeshNumber

        self.meshName = mc.textField('meshNameTF', query=True, text=True)
        self.keySpacing = mc.textField('keySpacingTF', query=True,
text=True)

        currentSelection = mc.ls(selection=True)
        print 'currentSelection is ', currentSelection, 'of type ',
type(currentSelection)

        if currentSelection == []:
            feedback = '>>>> nothing selected; job skipped'

            ##update UI label
            mc.text(self.feedbackT, edit=True, label=feedback)
            print feedback
            return self.meshName, meshNumber


        if currentSelection == None:
            feedback =  '>>>> NONE >>>> job skipped'

            ##update UI label
            mc.text(self.feedbackT, edit=True, label=feedback)
            print feedback
            return self.meshName, meshNumber

        else:
            # gets the shape node of the selection
            shapeNode = mc.ls(selection=True, shapes=True, dag=True,
objectsOnly=True)

            if type(shapeNode) != list:
                feedback = '>>>> shapeNode type is not a list; job
skipped'
```

```
                ##update UI label
                mc.text(self.feedbackT, edit=True, label=feedback)
                print feedback
                return self.meshName, meshNumber


            if len(shapeNode) == 0:
                feedback = '>>>> no shapeNode selected; job skipped'

                ##update UI label
                mc.text(self.feedbackT, edit=True, label=feedback)
                print feedback
                return self.meshName, meshNumber

            if mc.objectType(shapeNode[0]) != 'mesh':
                feedback = '>>>> selected node is not a mesh; job
skipped'

                ##update UI label
                mc.text(self.feedbackT, edit=True, label=feedback)
                print feedback
                return self.meshName, meshNumber

            else:
                #consider getting the API object from this
                transformNode = mc.listRelatives(shapeNode,
allParents=True, type='transform')
                feedback = 'transformNode is '+ str(transformNode)

                ##update UI label
                mc.text(self.feedbackT, edit=True, label=feedback)
                print feedback


                if len(transformNode) != 1:
                    feedback = '>>>> > or < one transformNode selected;
job skipped'

                    ##update UI label
                    mc.text(self.feedbackT, edit=True, label=feedback)
                    print feedback
                    return self.meshName, meshNumber

                else:
                    print '>>>> duplicating the transformNode'

###### delete history on the transform node
                    mc.delete(transformNode, constructionHistory=True)

                    #duplicate the transformNode
                    newModel = mc.duplicate(transformNode)

                    newModelParent = mc.listRelatives(newModel,
allParents=True)
                    print 'newModelParent is ', newModelParent, 'of type
', type(newModelParent)
```

```
                    if newModelParent == ['duplicateMeshGrp']:
                        print 'newModelParent is already
duplicateMeshGrp'
                        return self.meshName, meshNumber

                    ### if newModelParent is NoneType
                    if newModelParent == 'NoneType':
                        print 'newModelParent newModelParent == NoneType'
                        return self.meshName, meshNumber

                    else:
                        #make new group if it doesnt exist and add object
                        duplicateModelsGrp =
makeGroup(objectToAdd=newModel, groupName='duplicateMeshGrp')

                        #make display layer if it doesnt exist
                        makeDisplayLayer(objectToAdd='duplicateMeshGrp',
layerName='duplicateMeshLyr')

                        ##name the new model for the first time
                        mc.rename(newModel,
'{meshName}_'.format(meshName=self.meshName))
                        newModelnewName = mc.ls(selection=True,
long=True)[0]
                        print newModel , newModelnewName

                        ######### until I work out how to send meshNumber
to the function use meshNumber

        ########## NB. dont need to parse meshNumber--- and it doesnt
seem to work
                        meshNumber =
keyMeshVisibility(meshName=newModelnewName, keySpacing=self.keySpacing)
                        feedback = 'new meshNumber is ' + str(meshNumber)

                        #### update the self.currentMeshNumber ....hmm
will this always reset??
                        self.currentMeshNumber = meshNumber

                # resume the initial selection ("current selection")
                mc.select(currentSelection)
                ##update UI label
                mc.text(self.feedbackT, edit=True, label=feedback)
                print feedback
                return self.meshName, meshNumber
```