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

from pleinAirStillLife import duplicateModelFunction
from pleinAirStillLife import autoProcedure
from pleinAirStillLife import colourFacesTool
from pleinAirStillLife import createBlendShapes
from pleinAirStillLife import deformModelFunction
from pleinAirStillLife import createVisKeys
# from pleinAirStillLife import cameraAnimTools
from pleinAirStillLife import attrFunctions
from pleinAirStillLife import PPshaderCntrls
from pleinAirStillLife import PPmeshCntrls
from pleinAirStillLife import PPextraMeshFunctions

from pleinAirStillLife import cycleThruShapes


'''
07_08_16
Last updated 02_07_15 with the following amendments;
- load swatches from SS

Post Production utils;
- duplicate shape node under new transform
- select currently visible shape node
- nudge keys by amount X, forward or back

'''

class PleinAirStillLifeTools(object):
    def __init__(self, production=True, postProduction=True,
utilities=True, cameraAnim=False,\
        meshFunctions=True, autoProcedures=True, shaderCntrls=True,
visKeys=True, blendKeys=True, \
        PPmeshCntrl=True, attrFunctions=True):
        self.workOnTransformNodeStatus = True
        self.clm1 = 50
        self.clm2 = 50
        self.clm3 = 100
        self.separation = 15

        self.production = production
        self.postProduction = postProduction
        self.cameraAnim = False
        self.utilities = utilities

        self.meshFunctions = meshFunctions
        self.autoProcedures = autoProcedures
        self.shaderCntrls = shaderCntrls
        self.visKeys = visKeys
        self.blendKeys = blendKeys
        self.PPmeshCntrl = PPmeshCntrl
        self.attrFunctions = attrFunctions
        self.extraMeshFunctions = True

        self.launchUI()
```

```
    def setWorkOnTransformNodeStatus(self,
workOnTransformNodeStatus=True, *args):
        self.workOnTransformNodeStatus = workOnTransformNodeStatus
        print ' workOnTransformNodeStatus == ', workOnTransformNodeStatus
        #add the appropriate UI
        ####tab1
        if self.meshFunctions == True:

self.addMeshFunctionUI(workOnTransformNodeStatus=self.workOnTransformNode
Status)

self.addSecondaryMeshFunctionUI(workOnTransformNodeStatus=self.workOnTran
sformNodeStatus)
        ####tab2
        if self.visKeys == True:

self.addVisKeyUI(workOnTransformNodeStatus=self.workOnTransformNodeStatus
)
        if self.blendKeys == True:

self.addBlendUI(workOnTransformNodeStatus=self.workOnTransformNodeStatus)

        if self.shaderCntrls == True:

self.addShaderCntrlUI(workOnTransformNodeStatus=self.workOnTransformNodeS
tatus)

        ### mesh cntrls
        if self.PPmeshCntrl == True:

self.addMeshCntrlUI(workOnTransformNodeStatus=self.workOnTransformNodeSta
tus)
        if self.extraMeshFunctions == True:
            print 'self.extraMeshFunctions = TRUE'

self.addExtraMeshFunctionsUI(workOnTransformNodeStatus=self.workOnTransfo
rmNodeStatus)

        ####


    def launchUI(self):
        if mc.window('pleinAirStillLife', query=True, exists=True):
            mc.deleteUI('pleinAirStillLife')
        myWindow = mc.window('pleinAirStillLife', title='Plein Air Still
Life', w=(self.clm1+self.clm2+self.clm3), h=100)
        mc.showWindow(myWindow)

        self.mainLayout = mc.columnLayout()

        #### create radio buttons
        radioCollection1 = mc.radioCollection()
        RBlayout = mc.rowLayout(numberOfColumns=2,
parent=self.mainLayout)
        self.transformRBtab1 = mc.radioButton(label='Work with transform
nodes', parent=RBlayout, select=False, \
            onCommand=(partial(self.setWorkOnTransformNodeStatus, True)))
```

```
        self.shapeRBtab1 = mc.radioButton(label='Work with shape nodes',
parent=RBlayout, select=False, \
            onCommand=(partial(self.setWorkOnTransformNodeStatus,
False)))

        tabLayout = mc.tabLayout(parent=self.mainLayout)

        if self.production == True:
            self.mainColumnTab1 = mc.columnLayout(parent=tabLayout)
            mc.tabLayout( tabLayout, edit=True,
tabLabel=((self.mainColumnTab1, 'Production Tools')))
            # # ############################ add colour faces tool to the
TAB1 UI
            separator = mc.separator(style ="in", w=300,
h=self.separation, parent=self.mainColumnTab1)
            mc.text('===  Colour controls  ===',
parent=self.mainColumnTab1, align='right' )
            separator = mc.separator(style ="none", w=300,
h=self.separation, parent=self.mainColumnTab1)

colourFacesTool.ColourFacesTool(parentLayout=self.mainColumnTab1)

        if self.postProduction == True:
            self.mainColumnTab2 = mc.columnLayout(parent=tabLayout)
            mc.tabLayout( tabLayout, edit=True,
tabLabel=((self.mainColumnTab2, 'Post Production Tools')))

        ######### MESH FUNCTIONS ###
        if self.meshFunctions == True:
            self.tabMeshFunctions = mc.columnLayout(parent=tabLayout)
            mc.tabLayout( tabLayout, edit=True,
tabLabel=((self.tabMeshFunctions, 'PP Mesh Controls')))

        if self.cameraAnim == True:
            self.tabCameraAnim = mc.columnLayout(parent=tabLayout)
            mc.tabLayout( tabLayout, edit=True,
tabLabel=((self.tabCameraAnim, 'PP Camera Anim Tools')))
                    ######### tab 3 camera Anim
            if self.cameraAnim == True:
                self.addCameraAnimUI()

        if self.utilities == True:
            self.mainColumnTab4 = mc.columnLayout(parent=tabLayout)
            mc.tabLayout( tabLayout, edit=True,
tabLabel=((self.mainColumnTab4, 'PP Utilities')))
                     ######### tab 4
            if self.attrFunctions == True:
                self.addAttrFunctionsUI()
            if self.extraMeshFunctions == True:
                # print 'self.extraMeshFunctions = TRUE'

self.addExtraMeshFunctionsUI(self.workOnTransformNodeStatus)

        mc.showWindow(myWindow)

    def addMeshFunctionUI(self, workOnTransformNodeStatus=True, *args):
        if mc.columnLayout('meshFunctionUI', query=True, exists=True):
            mc.deleteUI('meshFunctionUI')
```

```
        meshFunctionUI = mc.columnLayout('meshFunctionUI',
parent=self.mainColumnTab1)

        separator = mc.separator(style ="in", w=300, h=self.separation,
parent=meshFunctionUI)
        mc.text('===  Function controls  ===\'
                , parent=meshFunctionUI, align='right' )
        separator = mc.separator(style ="none", w=300, h=self.separation,
parent=meshFunctionUI)

        if self.workOnTransformNodeStatus == True:
        # functionObject =
duplicateModelFunction.DuplicateShape(parentLayout=mainColumnTab1,
manual=True)
            functionObject =
duplicateModelFunction.DuplicateModel(parentLayout=meshFunctionUI,
manual=True, duplicateShapeStatus=False)
        else:
            functionObject =
duplicateModelFunction.DuplicateModel(parentLayout=meshFunctionUI,
manual=True, duplicateShapeStatus=True)

        self.addAutoProceduresUI(functionObject=functionObject)


    ################## auto procedures
##############################################
    def addAutoProceduresUI(self, functionObject=0, *args):
        createCameras = False

        if mc.columnLayout('autoProcedureUI', query=True, exists=True):
            mc.deleteUI('autoProcedureUI')

        autoProcedureUI = mc.columnLayout('autoProcedureUI',
parent=self.mainColumnTab1)

                ######################### add UI for auto Procedure
        separator = mc.separator(style ="in", w=300, h=self.separation,
parent=autoProcedureUI)
        mc.text('===  Automation controls  ===\'
                , parent=autoProcedureUI, align='right' )
        separator = mc.separator(style ="none", w=300, h=self.separation,
parent=autoProcedureUI)
        autoProcedure.AutomaticProcedures(parentLayout=autoProcedureUI,
functionObject=functionObject, \
            functionText = 'self.functionObject.executeFunction()',
createCameras=createCameras)


    def addSecondaryMeshFunctionUI(self, workOnTransformNodeStatus=True,
*args):
        if mc.columnLayout('secondaryMeshFunctionLayout', query=True,
exists=True):
            mc.deleteUI('secondaryMeshFunctionLayout')
```

```
        secondaryMeshFunctionLayout =
mc.columnLayout('secondaryMeshFunctionLayout',
parent=self.mainColumnTab1)
        separator = mc.separator(style ="in", w=300, h=self.separation,
parent=secondaryMeshFunctionLayout)
        ######################## dont need the following
########################
        mc.text('===  Secondary mesh deformation function controls  ===',
parent=secondaryMeshFunctionLayout, align='right' )
        #         ################### secondary deformation procedure
########################
        if self.workOnTransformNodeStatus == True:
            secondaryFunctionObject =
deformModelFunction.deformModel(parentLayout=secondaryMeshFunctionLayout,
duplicateShapeStatus=False)
        else:
            secondaryFunctionObject =
deformModelFunction.deformModel(parentLayout=secondaryMeshFunctionLayout,
duplicateShapeStatus=True)


self.addSeecondaryAutoProcedure(functionObject=secondaryFunctionObject)


    def addSeecondaryAutoProcedure(self, functionObject=0, *args):
        createCameras = False

        if mc.columnLayout('secondaryAutoProcedure', query=True,
exists=True):
            mc.deleteUI('secondaryAutoProcedure')

        secondaryAutoProcedureLayout =
mc.columnLayout('secondaryAutoProcedure', parent=self.mainColumnTab1)

                ######################### add UI for auto Procedure
        # separator = mc.separator(style ="in", w=300, h=self.separation,
parent=secondaryAutoProcedureLayout)
        # mc.text('===  Secondary automation controls  ===' ,
parent=secondaryAutoProcedureLayout, align='right' )
        # separator = mc.separator(style ="none", w=300,
h=self.separation, parent=secondaryAutoProcedureLayout)


autoProcedure.AutomaticProcedures(parentLayout=secondaryAutoProcedureLayo
ut, functionObject=functionObject, \
            functionText = 'self.functionObject.xFormComponents()',
createCameras=createCameras)

    ######################################################


    def addShaderCntrlUI(self, workOnTransformNodeStatus=True, *args):
                #### check if visKeyUI exists and delete it if it does
        if mc.columnLayout('shaderCntrlUI', query=True, exists=True):
            mc.deleteUI('shaderCntrlUI')

        shaderCntrlUI = mc.columnLayout('shaderCntrlUI',
parent=self.mainColumnTab2)
```

```
        separator = mc.separator(style ="in", w=300, h=self.separation,
parent=shaderCntrlUI)
        mc.text('===  Shader controls  ==='\
                , parent=shaderCntrlUI, align='right' )
        separator = mc.separator(style ="none", w=300, h=10,
parent=shaderCntrlUI)

        #### add button to set camera BG colour to viewport Colour
        PPshaderCntrls.SetCameraBGcolour(parentLayout=shaderCntrlUI)
        #### add button for colour tweak cntrls
        PPshaderCntrls.ColourTweakCntrls(parentLayout=shaderCntrlUI)
        #### add button for projector
        PPshaderCntrls.TextureCntrls(parentLayout=shaderCntrlUI)
        ## add shader transparency stuff
        #
PPshaderCntrls.KeyShaderTransparency(parentLayout=shaderCntrlUI,
workOnTransformNodeStatus=self.workOnTransformNodeStatus)
        if workOnTransformNodeStatus == True:

PPshaderCntrls.KeyShaderTransparency(parentLayout=shaderCntrlUI,
workOnTransformNodeStatus=True)
        else:

PPshaderCntrls.KeyShaderTransparency(parentLayout=shaderCntrlUI,
workOnTransformNodeStatus=False)



    def addVisKeyUI(self, workOnTransformNodeStatus=True, *args):
                #### check if visKeyUI exists and delete it if it does
        if mc.columnLayout('visKeyUI', query=True, exists=True):
            mc.deleteUI('visKeyUI')

        visKeyUI = mc.columnLayout('visKeyUI',
parent=self.mainColumnTab2)

        separator = mc.separator(style ="in", w=300, h=self.separation,
parent=visKeyUI)
        mc.text('===  Visibility Keyframe controls  ==='\
                , parent=visKeyUI, align='right' )
        separator = mc.separator(style ="none", w=300, h=self.separation,
parent=visKeyUI)

        if workOnTransformNodeStatus == True:
            createVisKeys.CreateVisKeysUI(parentLayout=visKeyUI,
workOnTransformNodeStatus=True)
        else:
            createVisKeys.CreateVisKeysUI(parentLayout=visKeyUI,
workOnTransformNodeStatus=False)

    def addBlendUI(self, workOnTransformNodeStatus=True, *args):
        #### check if blendUI exists and delete it if it does
        if mc.columnLayout('blendUI', query=True, exists=True):
            mc.deleteUI('blendUI')

        blendUI = mc.columnLayout('blendUI', parent=self.mainColumnTab2)
        ########## add UI for blend shapes tween shape nodes
```

```
        separator = mc.separator(style ="in", w=300, h=self.separation,
parent=blendUI)
        mc.text('===  Blend controls  ===', parent=blendUI, align='right'
)
        separator = mc.separator(style ="none", w=300, h=self.separation,
parent=blendUI)

        if self.workOnTransformNodeStatus == True:
            createBlendShapes.CreateBlendTweenNodes(parentLayout=blendUI,
workOnTransformNodeStatus=True)
        else:
            createBlendShapes.CreateBlendTweenNodes(parentLayout=blendUI,
workOnTransformNodeStatus=False)

    def addMeshCntrlUI(self, workOnTransformNodeStatus=True, *args):
        #### check if addMeshCntrlUI exists and delete it if it does
        if mc.columnLayout('meshCntrlUI', query=True, exists=True):
            mc.deleteUI('meshCntrlUI')

        meshCntrlUI = mc.columnLayout('meshCntrlUI',
parent=self.tabMeshFunctions)
        ########## add UI for blend meshCntrlUI tween shape nodes
        separator = mc.separator(style ="in", w=300, h=self.separation,
parent=meshCntrlUI)


        if self.workOnTransformNodeStatus == True:
            # PPmeshCntrls.MeshCntrls(parentLayout=meshCntrlUI)
            mc.text('===  Mesh controls for transform nodes  ===',
parent=meshCntrlUI, align='right' )
            separator = mc.separator(style ="none", w=300,
h=self.separation, parent=meshCntrlUI)
            print ' make mesh cntrls for transform nodes'
        else:
            mc.text('===  Mesh controls for shape nodes ===',
parent=meshCntrlUI, align='right' )
            separator = mc.separator(style ="none", w=300,
h=self.separation, parent=meshCntrlUI)
            PPmeshCntrls.MeshCntrls(parentLayout=meshCntrlUI,
workOnTransformNodeStatus=self.workOnTransformNodeStatus)
# TESTING cycle through shape setup
            cycleThruShapes.CycleShapeCntrls(parentLayout=meshCntrlUI)


    def addExtraMeshFunctionsUI(self, workOnTransformNodeStatus):
        print 'adding addExtraMeshFunctionsUI'
        if mc.columnLayout('meshFunctions', query=True, exists=True):
            mc.deleteUI('meshFunctions')

        meshFunctionsUI = mc.columnLayout('meshFunctions',
parent=self.tabMeshFunctions)
        ######################### add post Production UI for camera
animation
        separator = mc.separator(style ="in", w=300, h=self.separation,
parent=meshFunctionsUI)
        mc.text('===  More mesh controls  ===''\
                , parent=meshFunctionsUI, align='right' )
```

```
        separator = mc.separator(style ="none", w=300, h=self.separation,
parent=meshFunctionsUI)

        # 07_2015 these were commented out ... ?? I will stipulate in
PPextraMeshFunctions what gets added
        #
PPextraMeshFunctions.meshFunctionsUI(parentLayout=meshFunctionsUI,
workOnTransformNodeStatus=workOnTransformNodeStatus)
        #
PPextraMeshFunctions.shuffleMeshFunctionsUI(parentLayout=meshFunctionsUI,
workOnTransformNodeStatus=workOnTransformNodeStatus)


    # def addCameraAnimUI(self):
    #     if mc.columnLayout('cameraAnimUI', query=True, exists=True):
    #         mc.deleteUI('cameraAnimUI')

    #     cameraAnimUI = mc.columnLayout('cameraAnimUI',
parent=self.tabCameraAnim)
    #     ######################### add post Production UI for camera
animation
    #     separator = mc.separator(style ="in", w=300, h=self.separation,
parent=cameraAnimUI)
    #     mc.text('===  Camera Animation controls  ==='\
    #             , parent=cameraAnimUI, align='right' )
    #     separator = mc.separator(style ="none", w=300,
h=self.separation, parent=cameraAnimUI)
    #     cameraAnimTools.CameraAnim(parentLayout=cameraAnimUI)

    def addAttrFunctionsUI(self):
        if mc.columnLayout('attrFunctions', query=True, exists=True):
            mc.deleteUI('attrFunctions')

        attrFunctionsUI = mc.columnLayout('attrFunctions',
parent=self.mainColumnTab4)
        ######################### add post Production UI for camera
animation
        separator = mc.separator(style ="in", w=300, h=self.separation,
parent=attrFunctionsUI)
        mc.text('===  Node Attribute controls  ==='\
                , parent=attrFunctionsUI, align='right' )
        separator = mc.separator(style ="none", w=300, h=self.separation,
parent=attrFunctionsUI)
        attrFunctions.attrFunctionsUI(parentLayout=attrFunctionsUI)



PleinAirStillLifeTools( meshFunctions=True, autoProcedures=True,
visKeys=True, blendKeys=True, \
    cameraAnim=False)
```