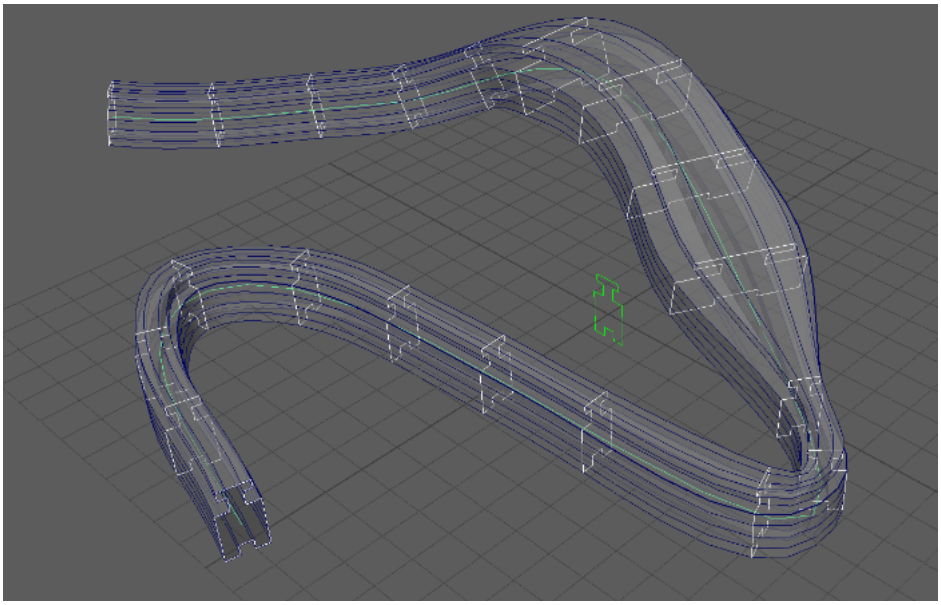
Maya Python

From bernie's

Contents

- 1 Copy object and spread it along a path
- 2 Pop-up (HUD) slider to change channel box attribute speed
- 3 Reconnect motion paths
- 4 Match translates and pivots
- 5 No-flip flow motion path object using curve normals
- 6 zDepth control tool
- 7 Create python generating code for given curve
- 8 Attach Geos to closest Geo with follicles
- 9 Attach to Closest Point On Curve
- 10 Make UV Check lambert
- 11 Check if latest ref

Copy object and spread it along a path



```
import maya.cmds as mc

result = mc.promptDialog(
        title='Copy along curve',
        message='Uses motionPath to spread copies of your first object on the second object selected (a path)\nThis means you can animate the curve. Check uVal attr on objects to disp
        button=['OK', 'Cancel'],
        defaultButton='OK',
        cancelButton='Cancel',
        tx=10,
        dismissString='Cancel')

if result == 'OK':
    count = int(mc.promptDialog(query=True, text=True))
    path = mc.ls(sl=1)[1]
    pathShp = mc.listRelatives(path,s=1)[0]
    profile = mc.ls(sl=1)[0]
    emGroup = mc.group(n='copies',em=1)

    for i in range(count):
        obj = mc.duplicate(profile,n=profile+'_copy')[0]
        mc.addAttr(obj,ln="uVal")
        mc.setAttr(obj+'.uVal',1.0*i/(count-1),e=1,k=1)
        mc.parent(obj,emGroup)
        pathA = mc.pathAnimation(obj,follow=True,fm=1,c=pathShp)
        mc.connectAttr(obj+'.uVal',pathA+'.uValue',f=1)
```

Pop-up (HUD) slider to change channel box attribute speed

```
import maya.cmds as mc
from math import pow

def chboxspeed( HUD, *args ):
    v = mc.hudSliderButton( HUD, query=True, v=True )
    nv = pow(10,v)
    mc.channelBox("mainChannelBox", edit=True, spd=nv)
    if nv < 1.0:
        nv =  "%.6f" % nv
    else:
        nv = int(nv)
    mc.hudSliderButton( HUD, e=True, sl=nv )
    if len(args)>0:
        mc.channelBox("mainChannelBox", edit=True, spd=chboxspeedvalue)
        mc.headsUpDisplay('HUDchboxspeed',rem=1)
```

```
chboxspeedvalue = mc.channelBox("mainChannelBox", query=True, spd=1)
hud = mc.hudSliderButton( 'HUDchboxspeed', s=2, b=5, vis=True, sl='speed', value=0, type='int', min=-6, max=6, slw=50, vw=50, sln=100, si=1, bl='reset&delete', bw=80, bsh='rectangle',
```

Reconnect motion paths

```
''' reconnects selected objects with motion paths to a a given curve
select objects driven by motion path and path last '''

import maya.cmds as mc

objs = mc.ls(sl=1,l=1)
curveShp = mc.listRelatives(objs[-1])[0]

for o in objs[:-1]:
    mp = mc.listConnections(o+'.specifiedManipLocation')[0]
    if mc.nodeType(o) == 'motionPath':
        mp = o
    if mc.nodeType(mp) == 'motionPath':
        mc.connectAttr(curveShp+'.worldSpace[0]',mp+'.geometryPath',f=1)
    else:
        warning('\nNo motion path found on '+o)
```

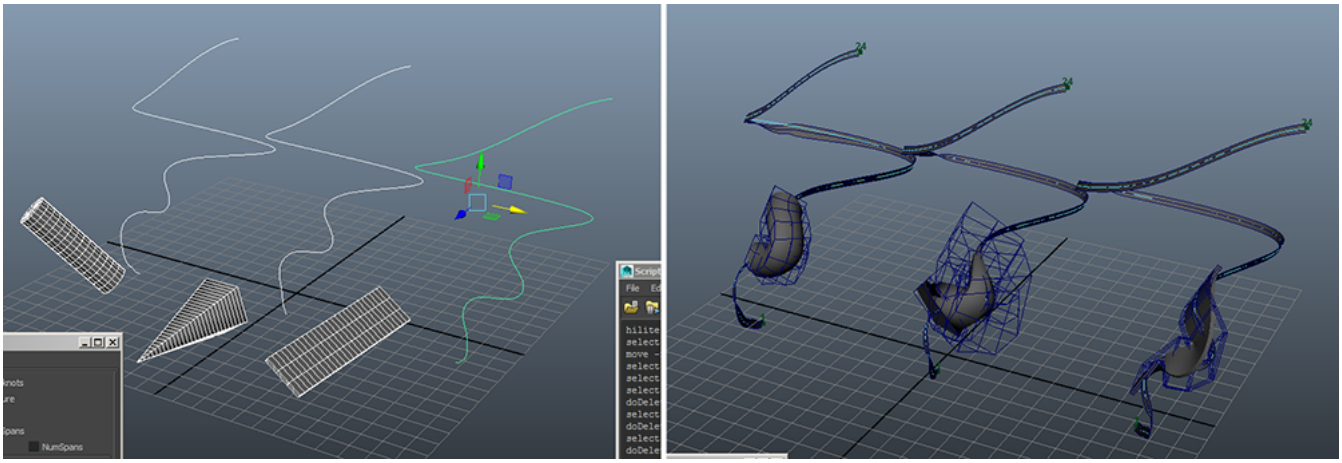
Match translates and pivots

Sets translates/rotates/scales and pivots of selected objects to the last selected object

```
# sets the pivots and transforms of objects to last object of selection, using parents and freeze transforms
# remember to delete history

import maya.cmds as mc

objs = mc.ls(sl=1)
target = objs[-1]

targetPivot = mc.xform(target,q=1,ws=1,rp=1)

for o in objs[:-1]:
    parentObj = mc.listRelatives(o,parent=1)
    mc.parent(o,target)
    mc.makeIdentity(o,a=1,)

    mc.xform(o,ws=1,rp=[targetPivot[0],targetPivot[1],targetPivot[2]])
    mc.xform(o,ws=1,sp=[targetPivot[0],targetPivot[1],targetPivot[2]])

    if parentObj:
        mc.parent(o,parentObj)
    else:
        mc.parent(o,w=1)

print('You might need to delete histories')
```

No-flip flow motion path object using curve normals

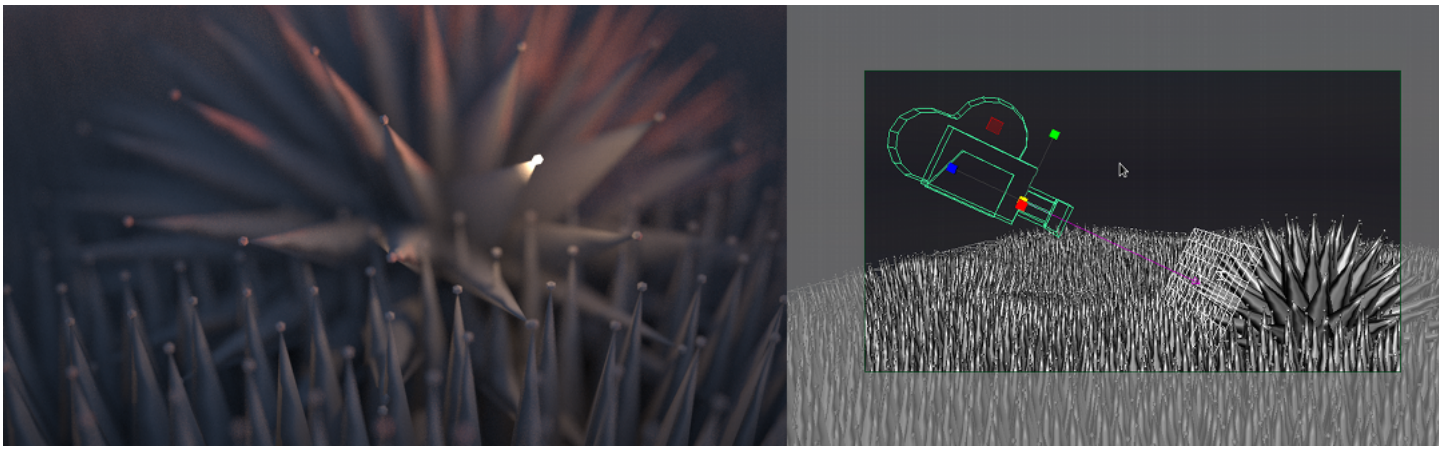
Well, not really no-flip, but you can easily modify the motion path loft



```
#select object and curve and run code

import maya.cmds as mc

def curveOnSurfaceFromCurve(curveObj,offsetDistance=.5):
    offset1 = mc.offsetCurve(curveObj,ugn=0,d=offsetDistance)
    offset2 = mc.offsetCurve(curveObj,ugn=0,d=-offsetDistance)
    loftObj = mc.loft(offset1,offset2,u=1,ch=1)[0]
    loftShp = mc.listRelatives(loftObj,s=1)[0]
    v = mc.getAttr(loftShp+'.minMaxRangeV.maxValueV')
    cOs = mc.curveOnSurface(loftObj,d=3,uv=((.5, 0),(0.5, 1*v/3), (0.5, 2*v/3), (0.5, v)))
    return cOs

def motionPathWithNormalsAndFlow(object,motionPathCurve,useTimeline=True,flowSpans=10):
    mc.xform(object,a=True, ro=(0, 0, 0) )
    bbox = mc.xform(object,q=1,os=1,bb=1)
    axes = sorted([(bbox[3]-bbox[0],'x',0),(bbox[4]-bbox[1],'y',1),(bbox[5]-bbox[2],'z',2)])
    if useTimeline:
        mc.pathAnimation(object,c=motionPathCurve,wut="normal",f=1,ua=axes[0][1],fa=axes[-1][1],fm=1,stu=mc.playbackOptions(q=1,minTime=1),etu=mc.playbackOptions(q=1,maxTime=1))
    else:
        mc.pathAnimation(object,c=motionPathCurve,wut="normal",f=1,ua=axes[0][1],fa=axes[-1][1])
```

```
    flowLatticeSides = [3,3,3]
    flowLatticeSides[axes[-1][2]] = flowSpans
    mc.flow(object,dv=flowLatticeSides)

selectedCurve = mc.ls(sl=1,tl=1)
object = mc.ls(sl=1,hd=1)
motionPathCurveOnSurface = curveOnSurfaceFromCurve(selectedCurve)
motionPathWithNormalsAndFlow(object,motionPathCurveOnSurface)
```

zDepth control tool

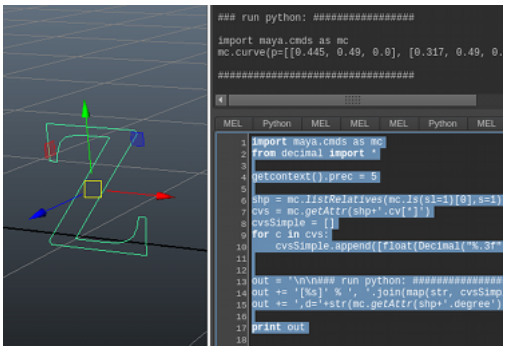


```
import maya.cmds as mc

def oneCurveGrid(rezx=10,rezy=10,scalex=1,scaley=1):

    points = []

    sx = 1.0/(rezx-1)
    sy = 1.0/(rezy-1)


    for i in range(rezx/2):


        for j in range(rezy):
            points.append( [ ( 2.0 * i * sx -.5 ) * scalex , ( j * sy -.5 ) * scaley ,0] )
        for j in reversed(range(rezy)):
            points.append( [ ( ( 2.0* i + 1) * sx -.5 ) * scalex ,( j * sy -.5 ) * scaley ,0])

    for i in range(rezy/2):
        for j in reversed(range(rezx)):
            points.append( [ ( j * sx -.5 ) * scalex , ( i*2.0 * sy -.5 ) * scaley ,0])
        for j in range(rezx):
            points.append( [ ( j * sx -.5 ) * scalex , ( ( i*2.0+1) * sy -.5 ) * scaley ,0])

    for i in reversed(range(rezx)):
        points.append( [((i * sx -.5)*scalex) ,((i * sx -.5)*scaley), 0])

    return points


def createZDistanceTool():
    '''creates controllers to help with the focus distance. Plugs into 'focusDistance' of camera shape'''

    # choose cam

    selection = mc.ls(sl=True)

    chosenCamera = False

    if len(selection) > 0:
        selShp = (selection[0],mc.listRelatives(selection[0],s=True))[mc.nodeType(selection[0])=='transform'] #gives us the shape
        if mc.nodeType(selShp) == 'camera':
            chosenCamera = selShp
    else:
        currentPanel = mc.getPanel(withFocus=True)
        if mc.getPanel(typeOf=currentPanel) == 'modelPanel':
            chosenCamera = mc.modelPanel(currentPanel,q=True,cam=True)
            chosenCamera = mc.listRelatives(chosenCamera,s=True)

    if not chosenCamera:
        mc.warning('Select camera or run in a perspective viewport')
    else:
        camObj = mc.listRelatives(chosenCamera,p=1)[0]
        camShape = chosenCamera[0]
        confirm  = mc.confirmDialog( title='Confirm', message='Create z-depth controller for "'+camObj+'" ?', button=['Yes','No'], defaultButton='Yes', cancelButton='No', dismissStrin
        if confirm == 'Yes':

            #spaghetti code below

            #floating locator system

            floatingZLoc = mc.curve(p=[[-0.319, -0.001, 0.0], [0.621, 0.94, 0.0], [-1.0, 0.94, 0.0], [-1.0, 1.06, 0.0], [1.392, 1.06, 0.0], [-0.608, -0.94, 0.0], [1.0, -0.94, 0.0], [1
            projCurve = mc.curve(p=[[0,0,0],[0,0,1000]],d=1,n=camObj+'_longCurve')                  #create a curve to project our locator on
            pc = mc.parentConstraint(camObj,projCurve,mo=False)[0]                                  #curve follows camera
            mc.setAttr(pc+'.target[0].targetOffsetRotateY',180)
            cameraLocator = mc.spaceLocator(n=camObj+'_camLocator')[0]                              #Locator on camera to get world position later
            mc.pointConstraint(camObj,cameraLocator,mo=False)
            floatingLocator = mc.spaceLocator(n=camObj+'_floatLocator')[0]                          #Locator to get floating controller world pos
            mc.parent(floatingLocator,floatingZLoc)
            projectionLocator = mc.spaceLocator(n=camObj+'_projLocator')[0]                         #Locator to project on curve

            nPoc = mc.createNode('nearestPointOnCurve')
            mc.connectAttr(mc.listRelatives(floatingLocator,s=True)[0]+'.worldPosition',nPoc+'.inPosition',f=True)
            mc.connectAttr(mc.listRelatives(projCurve,s=True)[0]+'.worldSpace',nPoc+'.inputCurve',f=True)
            mc.connectAttr(nPoc+'.position',projectionLocator+'.translate',f=True)

            db = mc.createNode('distanceBetween')
            mc.connectAttr(mc.listRelatives(projectionLocator,s=True)[0]+'.worldPosition',db+'.point1')
```

```
mc.connectAttr(mc.listRelatives(cameraLocator,s=True)[0]+'.worldPosition',db+'.point2')

#end result locator and grid

resultLocator = mc.spaceLocator(n=camObj+'resultLocator')[0]
resultGrid = mc.curve(p=oneCurveGrid(14,8,3.84,2.16),d=1,n=camObj+'_resultGrid')
mc.setAttr(resultGrid+'.tz',keyable=False,channelBox=False )
mc.setAttr(resultGrid+'.visibility',keyable=False,channelBox=False )

#camera based system

cameraGrid = mc.curve(p=oneCurveGrid(28,16,8,4.5),d=1,n=camObj+'_zDepth_Ctrl')          #create a grid to visualize standard focus distance z
cameraGridNull = mc.group(em=True,n=camObj+'_zDepth_Ctrl_ZERO')
mc.parent(cameraGrid,cameraGridNull)
mc.parent(resultLocator,cameraGridNull)
mc.parent(resultGrid,cameraGridNull)
pc2 = mc.parentConstraint(projCurve,cameraGridNull,mo=False)[0]


#grab existing zdepth focus distance, with renderer specific options (maxwell for now)

currentDistance = 5.0
if mc.getAttr("defaultRenderGlobals.currentRenderer") == "maxwell":
    currentDistance= mc.getAttr(camShape+".mxFocusDistance")
else:
    currentDistance = mc.getAttr(camShape+'.focusDistance')

mc.setAttr(cameraGrid+'.tz',currentDistance)

#choice system

sr = mc.createNode('setRange')
sr2 = mc.createNode('setRange')

mc.setAttr(sr+'.oldMaxX',1)
mc.setAttr(sr+'.oldMaxY',1)
mc.setAttr(sr+'.maxY',0)
mc.setAttr(sr+'.minY',1)

mc.setAttr(sr2+'.maxX',1)
mc.setAttr(sr2+'.maxY',0)
mc.setAttr(sr2+'.minY',1)
mc.setAttr(sr2+'.oldMaxX',.001)
mc.setAttr(sr2+'.oldMinY',.999)
mc.setAttr(sr2+'.oldMaxY',1)

mc.connectAttr(db+'.distance',sr+'.maxX',f=1)
mc.connectAttr(cameraGrid+'.tz',sr+'.minX',f=1)
mc.connectAttr(sr+'.outValue.outValueX',resultLocator+'.tz',f=True)


arrow = mc.annotate(resultLocator,p=(0, 0, 0))
arrowObj = mc.listRelatives(arrow,p=1)[0]
arrowObj = mc.rename(arrowObj,camObj+'_zDistance')
arrow = mc.listRelatives(arrowObj,s=1)[0]
mc.parent(arrowObj,cameraGridNull,r=True)

md = mc.createNode('multiplyDivide')                                                 #ridiculous hack to force annoation update
mc.connectAttr(camObj+'.tx',md+'.input2.input2X')
mc.connectAttr(sr+'.outValue.outValueX',md+'.input2.input2Y')
mc.connectAttr(floatingZLoc+'.tx',md+'.input2.input2Z')
mc.connectAttr(md+'.output',mc.listRelatives(arrow,p=1)[0]+'.translate')



#mostly controls and cosmetic stuff

for o in (projCurve,resultLocator,floatingLocator,projectionLocator,cameraLocator,pc2):
    mc.setAttr(o+'.visibility',False)

for o in (cameraGrid,arrowObj):
    for p in ('tx','ty','tz','sx','sy','sz','rx','ry','rz'):
        mc.setAttr(o+'.'+p,keyable=False,channelBox=False )

mc.setAttr(cameraGrid+'.tz',keyable=True,channelBox=False )
mc.setAttr(cameraGrid+'.tx',lock=True)
mc.setAttr(cameraGrid+'.ty',lock=True)


mc.addAttr(arrowObj,ln='focusDistance')
mc.setAttr(arrowObj+'.focusDistance',e=0,keyable=False,channelBox=True)
mc.connectAttr(sr+'.outValue.outValueX',arrowObj+'.focusDistance',f=True)
mc.connectAttr(sr+'.outValue.outValueX',resultGrid+'.tz',f=True) #
mc.connectAttr(sr2+'.outValue.outValueX',resultGrid+'.visibility',f=True)
mc.connectAttr(sr2+'.outValue.outValueY',cameraGrid+'.visibility',f=True)


mc.setAttr(resultGrid+'.overrideEnabled',1)
mc.setAttr(resultGrid+'.overrideColor',1)


mc.addAttr(arrowObj,ln='useFloatingLocator',dv=0.0,min=0.0,max=1.0)
mc.setAttr(arrowObj+'.useFloatingLocator',e=1,keyable=True,channelBox=False)
mc.connectAttr(arrowObj+'.useFloatingLocator',sr+'.valueX',f=True)
mc.connectAttr(arrowObj+'.useFloatingLocator',sr+'.valueY',f=True)
mc.connectAttr(arrowObj+'.useFloatingLocator',sr+'.valueZ',f=True)
mc.connectAttr(arrowObj+'.useFloatingLocator',sr2+'.valueX',f=True)
mc.connectAttr(arrowObj+'.useFloatingLocator',sr2+'.valueY',f=True)


mc.setAttr(arrow+'.overrideEnabled',1)
mc.setAttr(arrow+'.overrideColor',17)

mc.setAttr(cameraGrid+'.overrideEnabled',1)
mc.setAttr(cameraGrid+'.overrideColor',12)
mc.setAttr(floatingZLoc+'.overrideEnabled',1)
mc.setAttr(floatingZLoc+'.overrideColor',12)


zFocusGroup = mc.group(em=True,n=camObj+'_zFocusGroup')
mc.parent(cameraGridNull,floatingZLoc,projCurve,projectionLocator,cameraLocator,zFocusGroup)

if mc.getAttr("defaultRenderGlobals.currentRenderer") == "maxwell":
    mc.connectAttr(resultLocator+'.tz',camShape+'.mxFocusDistance',f=1)
mc.connectAttr(resultLocator+'.tz',camShape+'.focusDistance',f=1)

mc.select(arrowObj)
```

```
createZDistanceTool()
```

Create python generating code for given curve




```
import maya.cmds as mc
from decimal import *

getcontext().prec = 5

shp = mc.listRelatives(mc.ls(sl=1)[0],s=1)[0]
cvs = mc.getAttr(shp+'.cv[*]')
cvsSimple = []
for c in cvs:
    cvsSimple.append([float(Decimal("%.3f" % c[0])),float(Decimal("%.3f" % c[1])),float(Decimal("%.3f" % c[2]))])


out = '\n\n### run python: ################\n\nimport maya.cmds as mc\nmc.curve(p='
out += '[%s]' % ', '.join(map(str, cvsSimple))
out += ',d='+str(mc.getAttr(shp+'.degree'))+')\n\n################################'

print out
```

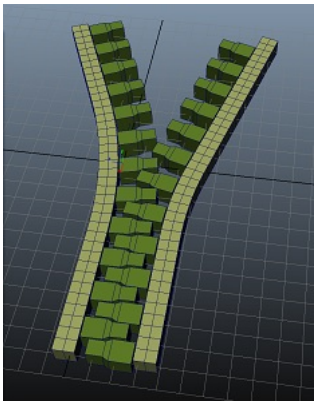
Result:

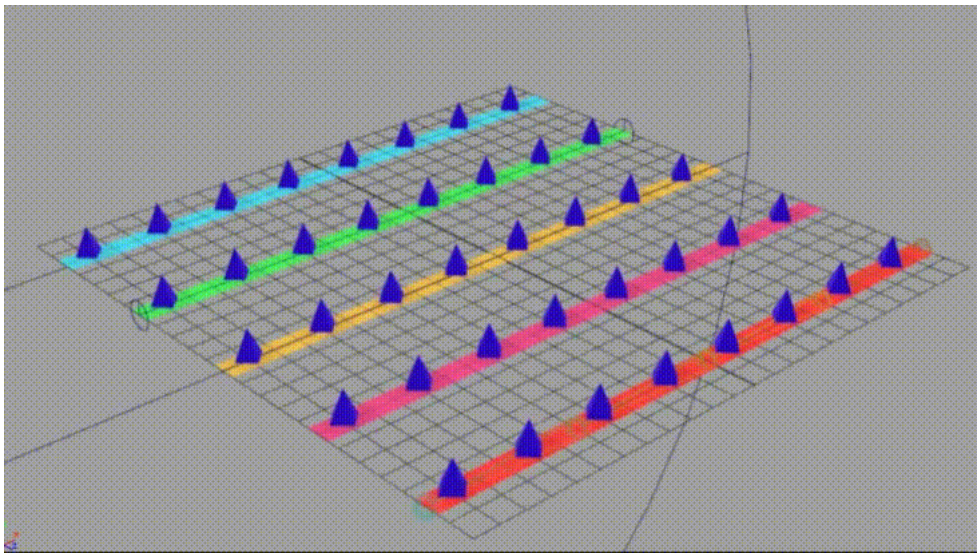
```
### run python: #################

import maya.cmds as mc
mc.curve(p=[[0.445, 0.49, 0.0], [0.317, 0.49, 0.0], [0.189, 0.49, 0.0], [0.061, 0.49, 0.0], [0.061, 0.441, 0.0], [0.061, 0.391, 0.0], [0.061, 0.342, 0.0], ...etc ,d=3)

################################
```

Attach Geos to closest Geo with follicles

Uses pivot point as attach point

https://i.imgur.com/5KQ8Tiv.mp4 (howto)



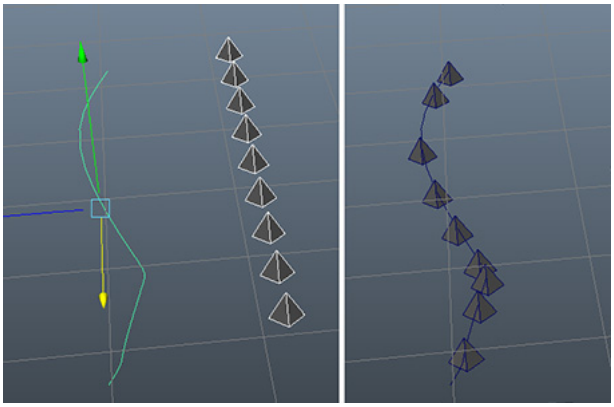


```
import maya.cmds as mc
from math import sqrt, fabs


def attachThingsToThingsUI(*kwargs):
    '''simple ui wrapper for "attachThingsToThings" function. No error checking or fancy stuff'''

    if len(kwargs) == 0:
        #ui
        if mc.windowPref('AttachStuffWindow',ex=1):
            mc.windowPref('AttachStuffWindow',remove=1)
        if cmds.window('AttachStuffWindow', exists=True):
            cmds.deleteUI('AttachStuffWindow', window=True)
        window = mc.window('AttachStuffWindow',title='Attach stuff to stuff', iconName='Short Name', widthHeight=(80, 55),rtf=1 )
        mc.frameLayout(mh=10,mw=10,bv=0,l="")
```

```
        mc.columnLayout(adjustableColumn=True )
        textInfo1 = 'This script attaches objects to other objects using follicles. Parent objects require non-overlapping UVs.'
        textInfo2 = 'Objects will be attached from their pivot to the closest point on parent mesh. In case of multiple parents, the script will try to figure out which is the closest
        textInfo3 = 'Tips: if attached objects \'jump\' when on animated objects, or are not stuck precisely, you need more polygons on your parent object(s)'
        mc.text(ww=1, label=textInfo1 )
        mc.separator(st="none",h=4)
        mc.text(ww=1, label=textInfo2 )
        mc.separator(st="none",h=8)
        mc.separator(st="in",h=4)
        mc.separator(st="none",h=8)
        mc.text(ww=1, label=textInfo3 )
        mc.separator(st="none",h=8)
        mc.separator(st="in",h=4)
        mc.separator(st="none",h=8)
        cb1 = mc.checkBox('cboxZero', label="Add \'zero out\' parents to all objects",v=1,rs=1,annotation="creates a group above each object that takes the constrains, so that your ob
        mc.checkBox('cboxCtrl', label="Add controllers to all objects",v=0,rs=1,annotation='creates \'zero-out\' controls that constrain your objects. Use if you have references and w
        mc.separator(st="none",h=8)
        mc.button( label='Set object(s) to attach',command=('mc.sets(n=\'tmp_childrenObjects_set\')') )
        mc.separator(st="none",h=2)
        mc.button( label='Set parent object(s)', command=('mc.sets(n=\'tmp_parentObjects_set\')') )
        mc.separator(st="none",h=10)
        mc.button( label='Apply', command=('attachThingsToThingsUI(1)') )
        mc.showWindow( window )
    else:
        children = mc.listConnections("tmp_childrenObjects_set",s=1,d=0,p=0,c=0)
        parents = mc.listConnections("tmp_parentObjects_set",s=1,d=0,p=0,c=0)
        useZeroOut = mc.checkBox("cboxZero",q=1,v=1)
        useCtrl = mc.checkBox("cboxCtrl",q=1,v=1)
        attachThingsToThings(children,parents,1,useZeroOut,useCtrl)
        mc.delete("tmp_childrenObjects_set","tmp_parentObjects_set")




def attachThingsToThings(objects,targets,keepHierachy=True,useZeroOut=True,useCtrl=False):
    '''requires closestUV() closestDistanceToMesh() mag() python math -- and non overlapping UVs '''
    selectedObjects = []
    for o in objects:
        parents = mc.listRelatives(o,p=1,f=1)
        smallest = 99999.9
        closest = u''
        outPos = []
        p = mc.xform(o,ws=1,q=1,rotatePivot=True)
        ro = mc.xform(o,ws=1,q=1,ro=True)
        bbox = mc.xform(o,ws=1,q=1,bb=True)
        ctrlscale = ((bbox[3]-bbox[0])+(bbox[4]-bbox[1])+(bbox[5]-bbox[2]))/3.0
        for r in targets:
            dist = closestDistanceToMesh(p,r)
            if dist < smallest:
                smallest = dist
                closest = r
                outPos = p
        follicle = closestUV(outPos,closest,attachFollicle=1)

        eg = ''
        if useZeroOut and not useCtrl:
            eg = mc.group(em=1,n=o+'zero')
            mc.parent(eg,o)
            mc.makeIdentity(eg)
            mc.parent(eg,w=1)
            mc.parent(o,eg)
        elif useCtrl:
            eg = mc.group(em=1,n=o+'zero')
            s = .33*ctrlscale/2.0;
            ctrlr = mc.curve(n=o+'ctrl',p=[[0.0, s*3, 0.0], [0.0, s*2, -s*2], [0.0, 0.0, -s*3], [0.0, -s*2, -s*2], [0.0, -s*3, 0.0], [0.0, -s*2, s*2], [0.0, 0.0, s*3], [0.0, s*2, s*2]
            ctrlrShp = mc.listRelatives(ctrlr,s=1)[0]
            mc.setAttr(ctrlrShp+'.overrideEnabled',1)
            mc.setAttr(ctrlrShp+'.overrideColor',12)
            mc.parent(ctrlr,eg)
            mc.setAttr('%s.t' % eg,p[0],p[1],p[2])
            mc.setAttr('%s.r' % eg,ro[0],ro[1],ro[2])
            mc.parentConstraint(ctrlr,o,mo=1)
            mc.scaleConstraint(ctrlr,o,mo=1)
            selectedObjects.append(ctrlr)
        else:
            eg = o
            selectedObjects.append(o)
        mc.parentConstraint(follicle,eg,mo=1)

        if keepHierachy and parents != None and useZeroOut==1 and useCtrl==0 :
            mc.parent(eg,parents[0])

        #add follicles to a group, hide it
        fGroup = 'FolliclesGroup'
        if not mc.objExists(fGroup):
            fGroup = mc.group(follicle,n=fGroup)
            mc.setAttr(fGroup+'.visibility',0)
        else:
            fGroup = mc.ls(fGroup)[-1]
            mc.parent(follicle,fGroup)
        mc.select(selectedObjects)

def closestUV(position,target,attachFollicle=0):
    '''target geo MUST have its transforms frozen'''
    cpom = mc.createNode( 'closestPointOnMesh')
    targetMeshes = mc.listRelatives(target,s=1,ni=1,f=1)
    #if len(targetMeshes):
    #    mc.warning('Multiple shapes found on \''+target+'\' follicles might not be connected properly')
    targetMesh = targetMeshes[0]

    mc.connectAttr(targetMesh+".worldMesh[0]",cpom+".inMesh")
    mc.connectAttr(targetMesh+".worldMatrix[0]",cpom+".inputMatrix")
    mc.setAttr(cpom+".inPosition", position[0], position[1], position[2], type="double3")

    getU = mc.getAttr(cpom+".result.parameterU")
    getV = mc.getAttr(cpom+".result.parameterV")
    mc.delete(cpom)
    if attachFollicle:
        follicle = mc.createNode('follicle')
        follicleObj = mc.listRelatives(follicle,p=1,f=1)[0]
        mc.connectAttr(targetMesh+".outMesh",follicle+".inputMesh")
        mc.connectAttr(targetMesh+".worldMatrix[0]",follicle+".inputWorldMatrix")
        mc.connectAttr(follicle+".outTranslate",follicleObj+".translate")
        mc.connectAttr(follicle+".outRotate",follicleObj+".rotate")
        mc.setAttr(follicle+".parameterU", getU)
        mc.setAttr(follicle+".parameterV", getV)
        return follicleObj
    else:
        return [getU,getV]
```

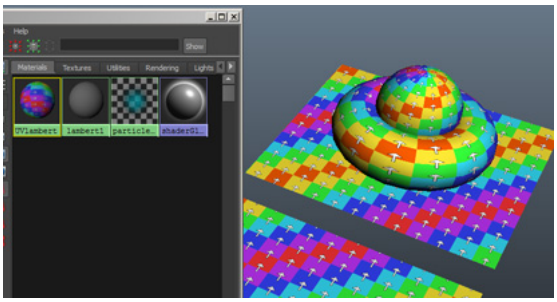
```
def closestDistanceToMesh(position,target):
    cpom = mc.createNode( 'closestPointOnMesh')
    targetMesh = mc.listRelatives(target,ni=1,s=1)[0]

    mc.connectAttr(targetMesh+".worldMesh[0]",cpom+".inMesh")
    mc.connectAttr(targetMesh+".worldMatrix[0]",cpom+".inputMatrix")
    mc.setAttr(cpom+".inPosition", position[0], position[1], position[2], type="double3")

    x = mc.getAttr(cpom+".result.position.positionX")
    y = mc.getAttr(cpom+".result.position.positionY")
    z = mc.getAttr(cpom+".result.position.positionZ")
    p2 = [x,y,z]
    mc.delete(cpom)
    return mag(position,p2)

def mag(v1,v2):
    return sqrt(pow(v2[0]-v1[0],2) + pow(v2[1]-v1[1],2) + pow(v2[2]-v1[2],2))


attachThingsToThingsUI()
```

Attach to Closest Point On Curve



```
import maya.cmds as mc


def attachToClosestPointOnCurve():
    #last element should be curve , no error checking - snaps to cloest point on mesh using motionPath

    sel = mc.ls(sl=1)
    curveObj = sel[len(sel)-1]
    sel.pop()
    npc = mc.createNode( 'nearestPointOnCurve')
    mc.connectAttr(mc.listRelatives(curveObj,s=1)[0]+".worldSpace[0]", npc+".inputCurve", f=1)
    for ob in sel:
        #parm = jc_closestPointOnCurve(mc.xform(ob, q=1, ws=1, t=1), curveObj)
        pos = mc.xform(ob, q=1, ws=1, t=1)
        mc.setAttr(npc+".inPosition", pos[0], pos[1], pos[2], type="double3")
        parm = mc.getAttr(npc+".parameter")
        #Loc = mc.spaceLocator(n=ob+"_pointOnCurve")
        pathAnim = mc.pathAnimation( ob, c=curveObj )
        inputs = mc.listConnections(pathAnim+".uValue", s=1)
        mc.disconnectAttr(inputs[0]+".output",pathAnim+".uValue")
        mc.setAttr(pathAnim+".uValue",parm)

    mc.delete(npc)
```

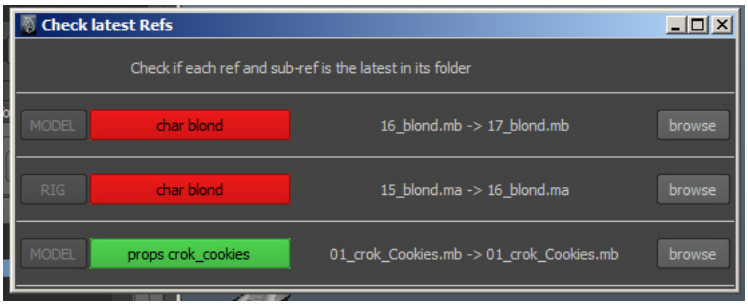
Make UV Check lambert



```
import urllib2
import os
import maya.cmds as cmds

tempDir = cmds.internalVar(userTmpDir=True)
uvFile = tempDir+'uv1024bis.jpg';
f = open(uvFile,'wb')
err = f.write(urllib2.urlopen('http://berniebernie.fr/dump/uv1024.jpg').read())
f.close()
shader=cmds.shadingNode("lambert",asShader=True,name="UVlambert")
file_node=cmds.shadingNode("file",asTexture=True)
file_text=cmds.shadingNode("place2dTexture",asUtility=True)
shading_group= cmds.sets(renderable=True,noSurfaceShader=True,empty=True)
cmds.connectAttr('%s.outColor' %shader ,'%s.surfaceShader' %shading_group)
cmds.connectAttr('%s.outColor' %file_node, '%s.color' %shader)
cmds.connectAttr('%s.coverage' %file_text ,'%s.coverage' %file_node)
cmds.connectAttr('%s.translateFrame' %file_text ,'%s.translateFrame' %file_node)
cmds.connectAttr('%s.rotateFrame' %file_text ,'%s.rotateFrame' %file_node)
```

```
cmds.connectAttr('%s.mirrorU' %file_text ,'%s.mirrorU' %file_node)
cmds.connectAttr('%s.mirrorV' %file_text ,'%s.mirrorV' %file_node)
cmds.connectAttr('%s.stagger' %file_text ,'%s.stagger' %file_node)
cmds.connectAttr('%s.wrapU' %file_text ,'%s.wrapU' %file_node)
cmds.connectAttr('%s.wrapV' %file_text ,'%s.wrapV' %file_node)
cmds.connectAttr('%s.repeatUV' %file_text ,'%s.repeatUV' %file_node)
cmds.connectAttr('%s.offset' %file_text ,'%s.offset' %file_node)
cmds.connectAttr('%s.rotateUV' %file_text ,'%s.rotateUV' %file_node)
cmds.connectAttr('%s.noiseUV' %file_text ,'%s.noiseUV' %file_node)
cmds.connectAttr('%s.vertexUvOne' %file_text ,'%s.vertexUvOne' %file_node)
cmds.connectAttr('%s.vertexUvTwo' %file_text ,'%s.vertexUvTwo' %file_node)
cmds.connectAttr('%s.vertexUvThree' %file_text ,'%s.vertexUvThree' %file_node)
cmds.connectAttr('%s.vertexCameraOne' %file_text ,'%s.vertexCameraOne' %file_node)
cmds.connectAttr('%s.outUV' %file_text ,'%s.uv' %file_node)
cmds.connectAttr('%s.outUvFilterSize' %file_text ,'%s.uvFilterSize' %file_node)
cmds.setAttr( '%s.fileTextureName' %file_node,uvFile,type="string" )
```

Check if latest ref




```
import maya.cmds as mc
import maya.mel as mel
from functools import partial
from os import listdir
from os.path import split, isfile, join, dirname
import subprocess

class refCheckWindow(object):

    onlyMissing = True

    def __init__(self):
        print "\nProcessing! Be patient =)"
        self.buildWin()
    def reinitWin(*args):
        w = refCheckWindow()
    def buttonPush(*args):
        refPath = dirname(args[1])
        refPath = refPath.replace('/', '\\')
        subprocess.Popen('explorer "%s"' % refPath)
    def buildWin(self):

        scene = mc.file(query=True,sn=True)
        allrefs = mc.file(query=True, list=True, withoutCopyNumber=True)
        fileRefs = sorted(list(set([item for item in allrefs if item.endswith('.ma') or item.endswith('.mb')])))
        if mc.window("checkRefs",ex=1):
            mc.deleteUI("checkRefs",window=1)
        mc.window("checkRefs",title='Check latest Refs',resizeToFitChildren=1,w=600)
        scrollLayout = cmds.scrollLayout(horizontalScrollBarThickness=16,verticalScrollBarThickness=16)
        mc.columnLayout(adjustableColumn=True, columnAlign='center',rowSpacing=10)

        mc.rowLayout(numberOfColumns=3)

        mc.text( label='     Checking if you have latest refs :     ', align='left' )
        mc.checkBox("cbox", label='Only show missing refs    ', v=1 )
        mc.button(w=55,l="Refresh",c=self.reinitWin)

        mc.setParent(upLevel=True)
        mc.separator()
        for ref in fileRefs:
            if ref != scene:
                curRefNamespace = mc.referenceQuery(ref,rfn=1)
                curRefParent = mc.referenceQuery(curRefNamespace,rfn=1,p=1)

                onlyfiles = [ f for f in listdir(split(ref)[0]) if isfile(join(split(ref)[0],f)) and f.endswith(('.mb','.ma')) ]
                #beautiful code below
                refFileName = ref.rpartition('/')[2]
                refStepName = ref.rpartition('/')[0].rpartition('/')[2];
                refChar = ref.rpartition('/')[0].rpartition('/')[0].rpartition('/')[0];
                refCharName = refChar.rpartition('/')[0].rpartition('/')[2]+" / "+refChar.rpartition('/')[2];
                refCharName = refCharName.lower();
                latestRef = sorted(onlyfiles)[-1];
                col = [.3,.8,.3];
                if(refFileName!=latestRef):
                    col = [.9,.1,.1];
                print ".",

                if( mc.checkBox("cbox",q=1,v=1) and refFileName!=latestRef):

                    mc.rowLayout(numberOfColumns=6,ann=curRefParent)

                    mc.separator()
                    mc.button(w=50,l=refStepName,en=0)
                    if curRefParent:
                        col = [(col[0]+.6)/2,(col[1]+.6)/2,(col[2]+.6)/2]
                    mc.button(w=150,l=refCharName,bgc=col)
                    mc.text(w=370,l="  "+refFileName+" -> "+latestRef+" ")

                    mc.button(w=55,l="browse",c=partial(self.buttonPush, ref))
                    mc.separator()
                    mc.setParent(upLevel=True)
                    mc.separator()


        mc.showWindow()
w = refCheckWindow()
```

Category: Maya
