```
import maya.cmds as mc

import random

import math


"""

MidtermAssignment-ChaithanyaKasireddy.py


ABOUT: This is the midterm assignment for VSFX 502 Programming
Models and Shaders, Fall 2022.


DESCRIPTION: This module creates a "Pine Apple" model and adds shaders procedurally.

        Creating Pine Apple shape by using the sphere.

        Deformed polygonal primitives, by moving vertices,extruding faces,edges.

        Transformations of polygonal objects like move, scale, rotate.

        Apply color per vertex in order to prepare your object for a render in arnold.

        Apply aiStandard surface and aiRamp for cones. For the I used Colorpervertex.

        Assign parenting the nodes and merging the leaf and pine Apple cones


        Here I created a sphere cutting it and extruding the sphere and adding mutiple cones.

        I have done duplication for number leaf and group them as branch. Finally I have merged

        my groupleaves and  pineApples and named as "FRUIT"



AUTHOR: Chaithanya Kasireddy
```

```
version 01: 10/19/22

"""




#Create and rename new file

mc.file(new = True, force = True)

#home = os.getenv("HOME")

#mc.file(rename = os.path.join(os.getenv("HOME"),"maya", "projects", "default", "scenes", "MidtermAssignment-Chaithanyakasireddy.mb"))


#PINEAPPLE:

# Here I am creating 'Pine Apple'


pinesphere = mc.polySphere(r = 4, n = "pineApple")

#Create color per vertex and color set

colorSetName_pinesphere = "procVertexColorpinesphere"

mc.polyColorSet(create = True, colorSet = colorSetName_pinesphere)

mc.polyColorPerVertex(colorRGB=[0.702,0.502,0.0948], colorDisplayOption = True)

scale = mc.scale(0.8,1.5,0.8 )

#deleting faces

deletefaceslower = mc.polyDelFacet('pineApple.f[0:99]','pineApple.f[360:399]','pineApple.f[300:359]','pineApple.f[380:399]')
```

```
move =mc.move( 0,3.5,0)
select =mc.select('pineApple.e[200:219]',r = True)
#Applying Bridge for edges
topcloseBorder = mc.polyCloseBorder( 'pineApple.e[200:219]')
select = mc.select(' pineApple.e[0:19]', r = True)
lowcloseBorder = mc.polyCloseBorder('pineApple.e[0:19]')
mc.polyOptions(colorShadedDisplay = True)
mc.select(clear = True)


#EvaluateInfo
numVertex = mc.polyEvaluate(pinesphere, vertex = True)
print numVertex
mc.select(clear = True)


#Apply Arnold material to 'Pinesphere'
mc.setAttr(pinesphere[0]+".aiExportColors", True)
userDataNode = mc.shadingNode("aiUserDataColor", asShader = True)
print( userDataNode )
mc.setAttr(userDataNode+".attribute",colorSetName_pinesphere, type = "string" )
multiplyNode = mc.shadingNode("aiMultiply", asShader = True)
mc.connectAttr(userDataNode+".outColor",multiplyNode+".input1")
standardSurface = mc.shadingNode("aiStandardSurface", asShader = True)
mc.connectAttr(multiplyNode+".outColor",standardSurface+".baseColor")
shadingGroup = mc.shadingNode("aiStandardSurfaceSG", asShader = True)
shadingGroup = standardSurface + "SG"

mc.setAttr(pinesphere[0]+".aiExportColors", True)

mc.select(pinesphere[0])

mc.hyperShade(assign = standardSurface)


for i in range(0,numVertex):

   vtxName = "{0}.vtx[{1}]".format(pinesphere[0],i)

   print vtxName

   mc.select(vtxName, add = True)


# create a list of all selected vertices

sel = mc.ls(selection = True, fl = True)


# Loop over all vertices and compute vertex normal for each vertex

for k in range(0, numVertex):

   vtxName = pinesphere[0]+".vtx[%d]"%k

   print vtxName

   mc.select(vtxName)


   normalX = mc.polyNormalPerVertex(q=True, x = True)

   avgNormalX = 0

   for nx in normalX:

      avgNormalX += nx

   avgNormalX = avgNormalX/len(normalX)
```

```
    print len(normalX)



    normalY = mc.polyNormalPerVertex(q=True, y = True)

    avgNormalY = 0

    for ny in normalY:

        avgNormalY += ny

    avgNormalY = avgNormalY/len(normalY)

    print len(normalY)



    normalZ = mc.polyNormalPerVertex(q=True, z = True)

    avgNormalZ = 0

    for nz in normalZ:

        avgNormalZ += nz

    avgNormalZ = avgNormalZ/len(normalX)

    print len(normalZ)


    #Compute magniture of the vertex normal

    magNormal = math.sqrt(avgNormalX * avgNormalX + avgNormalY * avgNormalY + avgNormalZ * avgNormalZ)

    #Normalize vertex normal vector

    avgNormalX = avgNormalX/magNormal

    avgNormalY = avgNormalY/magNormal
    avgNormalZ = avgNormalZ/magNormal
    print avgNormalX
    print avgNormalY
    print avgNormalZ

   # Instantiate a polygonal cone at every vertex, orient it along the vertex normal,
   # assign random height to each cone

    pos = mc.pointPosition(vtxName)
    sp = mc.polyCone(name = "Cone_"+str(k), r = 1,sh =2,h =1, axis = (avgNormalX,avgNormalY,avgNormalZ), height = random.randrange(1,5))
    #Create color per vertex and color set
    colorSetName_sp= "procVertexColorpinesphere"
    mc.polyColorSet(create = True, colorSet = colorSetName_sp)
    mc.polyColorPerVertex(colorRGB=[0.702,0.502,0.0948], colorDisplayOption = True)
    mc.polyOptions(colorShadedDisplay = True)
    mc.select(clear = True)
    mc.xform(sp, a = True, t = (pos[0], pos[1], pos[2]))
    mc.parent(sp,pinesphere)
    mc.polySmooth()

#Apply Arnold material to 'Cones':

mc.setAttr(pinesphere[0]+".aiExportColors", True)
userDataNode = mc.shadingNode("aiUserDataColor", asShader = True)
```

```
print( userDataNode )
mc.setAttr(userDataNode+".attribute",colorSetName_sp, type = "string" )
Ramp = mc.shadingNode("aiRampRgb",asShader = True)
multiplyNode = mc.shadingNode("aiMultiply", asShader = True)
mc.connectAttr(Ramp+".outColor",multiplyNode+".input1")
standardSurface = mc.shadingNode("aiStandardSurface", asShader = True)
mc.connectAttr(multiplyNode+".outColor",standardSurface+".baseColor")
mc.select(pinesphere[0])
mc.hyperShade(assign = standardSurface)

#Setting mutliple colors for my 'Cones' using 'aiRamp'


mc.select(Ramp)
mc.setAttr("aiRampRgb1.type",5)
mc.setAttr("aiRampRgb1.ramp[0].ramp_Interp",2)
mc.setAttr("aiRampRgb1.ramp[0].ramp_Color",1,0,0)
mc.setAttr("aiRampRgb1.ramp[1].ramp_Color",1,1,0)
mc.setAttr("aiRampRgb1.ramp[0].ramp_Position",0.808696)

#LEAFS:
#Here I created cube assgin name as 'Leaf'
leaf= mc.polyCube(h=0.5)
mc.move(-1.5,8.7,0)
mc.scale(2,0.5,2)
mc.select("pCube1.f[4]")

mc.polyExtrudeFacet("pCube1.f[4]",kft = True, ltz= 2, ls =(0.1,1,0))

mc.select("pCube1.e[5]","pCube1.e[7]","pCube1.e[9]","pCube1.e[11]")

mc.polyMoveEdge(ty = 2)

smooth = mc.polySmooth("pCube1",dv=4.5)


mc.select("pCube1.f[1303]","pCube1.f[240]","pCube1.f[253]","pCube1.f[228]","pCube1.f[225]",
"pCube1.f[32]","pCube1.f[45]","pCube1.f[20]"

,"pCube1.f[17]","pCube1.f[2352]","pCube1.f[2365]","
pCube1.f[2340]","pCube1.f[2337]","pCube1.f[2400]","pCube1.f[2413]","pCube1.f[2388]","pCub
e1.f[2386]"

,"pCube1.f[1265]","pCube1.f[1278]","pCube1.f[1253]","pCube1.f[1250]")

mc.polyExtrudeFacet("pCube1.f[1303]","pCube1.f[240]","pCube1.f[253]","pCube1.f[228]","pCu
be1.f[225]","pCube1.f[32]","pCube1.f[45]","pCube1.f[20]"

,"pCube1.f[17]","pCube1.f[2352]","pCube1.f[2365]","
pCube1.f[2340]","pCube1.f[2337]","pCube1.f[2400]","pCube1.f[2413]","pCube1.f[2388]","pCub
e1.f[2386]"

,"pCube1.f[1265]","pCube1.f[1278]","pCube1.f[1253]","pCube1.f[1250]", kft = True, ltz = 0.05)

mc.select("pCube1.f[1057]","pCube1.f[1070]","pCube1.f[1045]","pCube1.f[1042]","pCube1.f[19
69]","pCube1.f[1982]","pCube1.f[1957]","pCube1.f[1953]"

,"pCube1.f[2016]","pCube1.f[1831]","pCube1.f[2029]","pCube1.f[1852]","pCube1.f[1843]","pCu
be1.f[528]","pCube1.f[535]","pCube1.f[556]","pCube1.f[609]"

,"pCube1.f[672]","pCube1.f[686]","pCube1.f[661]","pCube1.f[658]","pCube1.f[1521]","pCube1.f
[1534]","pCube1.f[1509]","pCube1.f[1506]","pCube1.f[1313]")

mc.polyExtrudeFacet("pCube1.f[1057]","pCube1.f[1070]","pCube1.f[1045]","pCube1.f[1042]","
pCube1.f[1969]","pCube1.f[1982]","pCube1.f[1957]","pCube1.f[1953]"

,"pCube1.f[2016]","pCube1.f[1831]","pCube1.f[2029]","pCube1.f[1852]","pCube1.f[1843]","pCu
be1.f[528]","pCube1.f[535]","pCube1.f[556]","pCube1.f[609]"

,"pCube1.f[672]","pCube1.f[686]","pCube1.f[661]","pCube1.f[658]","pCube1.f[1521]","pCube1.f
[1534]","pCube1.f[1509]","pCube1.f[1506]","pCube1.f[1313]"
, kft = True, ltz = 0.05)

#Selecting my leaf
mc.select(leaf[0])
#Create color per vertex and color set
colorSetName_Leaf = "procVertexColorpinesphere"
mc.polyColorSet(create = True,colorSet = colorSetName_Leaf)
colors = mc.polyColorPerVertex(colorRGB=[0.102,0.202,0.0948], colorDisplayOption = True)

#Apply Arnold material to 'leaf'


mc.setAttr(leaf[0]+".aiExportColors", True)
userDataNode = mc.shadingNode("aiUserDataColor", asShader = True)
print( userDataNode )
mc.setAttr(userDataNode+".attribute",colorSetName_Leaf, type = "string" )
multiplyNode = mc.shadingNode("aiMultiply", asShader = True)
mc.connectAttr(userDataNode+".outColor",multiplyNode+".input1")
standardSurface = mc.shadingNode("aiStandardSurface", asShader = True)
mc.connectAttr(multiplyNode+".outColor",standardSurface+".baseColor")
shadingGroup = mc.shadingNode("aiStandardSurfaceSG", asShader = True)
shadingGroup = standardSurface + "SG"
mc.select(leaf[0])
mc.hyperShade(assign = standardSurface)
```

```
""" Here I did number of duplication my 'leaf' with ai Shaders """

#Creating a duplicate of leaf
mc.group("pCube1")
mc.duplicate('group1')
mc.rotate(0,90,0)
mc.move(-2.3,0,-2)
#Creating a duplicate of leaf
mc.group("group1","group2")
mc.duplicate("group3")
mc.rotate(0,-180,0)
mc.move(-3,0,3)
mc.group("group3","group4")
mc.select("group5")
mc.rename("LeafSet1")
#Creating a duplicate of leaf
mc.duplicate("LeafSet1")
mc.scale(1.2,1.2,1.2)
mc.move( 0,0.6,0)
mc.rotate(0,-40,0)
#Creating a duplicate of leaf
mc.group("LeafSet1","LeafSet2")
mc.select("group5")
mc.rename("leaves")
mc.move(2.5,-1,0)
#Group all Duplicates of group
mc.duplicate("leaves")
mc.select("leaves1")
mc.move(2.4,0.1,0)
mc.scale(1,1.4,1)
mc.rotate(0 ,-12.6,0)
#final_group
mc.group("leaves","leaves1", n ="groupleaves")


#Grouping my "Groupleaves" & "PineApple" then I rename as "FRUIT"
mc.group("groupleaves","pineApple", name = "fruit")
```