##ASSIGNMENT 01 ##

import rhinoscriptsyntax as rs

# data inputs

objID = rs.GetObject("select a curve",rs.filter.curve)

pt1 = rs.GetPoint("from point")

pt2 = rs.GetPoint("to point")

#center of the object

center= rs.CurveAreaCentroid(objID)[0]

#move vectors

#vector1=pt2-center

vector2=pt2-pt1

##transformations ex1-2

#rotated= rs.RotateObject(objID,center,18)

#scaled=rs.ScaleObject(rotated,center, (1.618,1.618,1),True)

#moved= rs.MoveObject(scaled,vector1)

#transformations ex3

rotated= rs.RotateObject(objID,center,18)

scaled=rs.ScaleObject(rotated,pt1, (1.618,1.618,1),True)

moved= rs.MoveObject(scaled,vector2)

##ASSIGNMENT 02.1 ##

import rhinoscriptsyntax as rs

line1=rs.GetObject("select a line",rs.filter.curve)

center=rs.GetPoint("select center")

line2=rs.RotateObject(line1,center,60,None,True)

line3=rs.RotateObject(line2,center,60,None,True)

line4=rs.RotateObject(line3,center,60,None,True)

line5=rs.RotateObject(line4,center,60,None,True)

line6=rs.RotateObject(line5,center,60,None,True)

po1=rs.DivideCurve(line1,6,True,True)

po2=rs.DivideCurve(line2,6,True,True)

po3=rs.DivideCurve(line3,6,True,True)

po4=rs.DivideCurve(line4,6,True,True)

po5=rs.DivideCurve(line5,6,True,True)

po6=rs.DivideCurve(line6,6,True,True)

#rs.AddTextDot(0,po1[0])

#rs.AddTextDot(1,po1[1])

#rs.AddTextDot(3,po1[3])

#rs.AddTextDot(4,po1[4])

#rs.AddTextDot(5,po1[5])

#rs.AddTextDot(6,po1[6])

rs.AddCurve((po1[6],po2[1]),1)

rs.AddCurve((po1[5],po2[2]),1)

rs.AddCurve((po1[4],po2[3]),1)

rs.AddCurve((po1[3],po2[4]),1)

rs.AddCurve((po1[2],po2[5]),1)

rs.AddCurve((po1[1],po2[6]),1)

rs.AddCurve((po1[6],po6[1]),1)

rs.AddCurve((po1[5],po6[2]),1)

rs.AddCurve((po1[4],po6[3]),1)

rs.AddCurve((po1[3],po6[4]),1)

rs.AddCurve((po1[2],po6[5]),1)

rs.AddCurve((po1[1],po6[6]),1)

rs.AddCurve((po3[6],po2[1]),1)

rs.AddCurve((po3[5],po2[2]),1)

rs.AddCurve((po3[4],po2[3]),1)

rs.AddCurve((po3[3],po2[4]),1)

rs.AddCurve((po3[2],po2[5]),1)

rs.AddCurve((po3[1],po2[6]),1)

rs.AddCurve((po3[6],po4[1]),1)

rs.AddCurve((po3[5],po4[2]),1)

rs.AddCurve((po3[4],po4[3]),1)

rs.AddCurve((po3[3],po4[4]),1)

rs.AddCurve((po3[2],po4[5]),1)

rs.AddCurve((po3[1],po4[6]),1)

rs.AddCurve((po5[6],po4[1]),1)

rs.AddCurve((po5[5],po4[2]),1)

rs.AddCurve((po5[4],po4[3]),1)

rs.AddCurve((po5[3],po4[4]),1)

rs.AddCurve((po5[2],po4[5]),1)

rs.AddCurve((po5[1],po4[6]),1)

rs.AddCurve((po5[6],po6[1]),1)

rs.AddCurve((po5[5],po6[2]),1)

rs.AddCurve((po5[4],po6[3]),1)

rs.AddCurve((po5[3],po6[4]),1)

rs.AddCurve((po5[2],po6[5]),1)

rs.AddCurve((po5[1],po6[6]),1)

##ASSIGNMENT 02.2 ##

import rhinoscriptsyntax as rs

import random as rnd

ptList =[]

imax= rs.GetInteger(" number of x direction",10)

jmax= rs.GetInteger(" number of y direction",10)

for i in range(imax):

for j in range (jmax):

x=i+2

y=j+2

z=0

rs.AddPoint(x,y,z)

ptList.append((x,y,z))

att = ptList[rnd.randint(42,50)]

for i in range(len(ptList)):

distance= rs.Distance(att,ptList[i])

if ptList[i]== att:

crc=rs.AddCircle(ptList[i],0.1)

else:

crc=rs.AddCircle(ptList[i],2/distance)

##ASSIGNMENT 03.1 ##

import rhinoscriptsyntax as rs

import random as rnd

ptDict = {}

crvList = []

imax = rs.GetInteger('input number in x direction',10)

jmax = rs.GetInteger('input number in y direction',10)

for i in range(imax):

for j in range(jmax):

x = i\*5+(i\*i)+#(rnd.random()\*2)

y = j\*5+(j\*j)+#(rnd.random()\*2)

z=0

rs.AddPoint(x,y,z)

ptDict[(i,j)]=(x,y,z)

print ptDict[i,j]

#first pattern

for i in range(imax):

for j in range(jmax):

if i>0 and j>0:

rs.AddCurve((ptDict[(i-1,j)],ptDict[(i,j-1)],ptDict[(i,j)],ptDict[(i-1,j)]),3)

rs.AddCurve((ptDict[(i,j)],ptDict[(i-1,j-1)],ptDict[(i-1,j)],ptDict[(i,j)]),3)

rs.AddCurve((ptDict[(i,j)],ptDict[(i-1,j-1)],ptDict[(i,j-1)],ptDict[(i,j)]),3)

rs.AddCurve((ptDict[(i-1,j)],ptDict[(i,j-1)],ptDict[(i-1,j-1)],ptDict[(i-1,j)]),3)

##second pattern

#for i in range(imax):

# for j in range(jmax):

# if i>0 and j>0:

# rs.AddCurve((ptDict[(i-1,j-1)],ptDict[(i,j)],ptDict[(i,j-1)],ptDict[(i-1,j)]),3)

# rs.AddCurve((ptDict[(i,j)],ptDict[(i-1,j-1)],ptDict[(i-1,j)],ptDict[(i,j-1)]),3)

# rs.AddCurve((ptDict[(i,j)],ptDict[(i-1,j-1)],ptDict[(i,j-1)],ptDict[(i-1,j)]),3)

# rs.AddCurve((ptDict[(i-1,j-1)],ptDict[(i,j)],ptDict[(i-1,j)],ptDict[(i,j-1)]),3)

##ASSIGNMENT 03.2 ##

import rhinoscriptsyntax as rs

import random as rnd

ptDict = {}

crvList = []

frameNum=rs.GetInteger('input frame number ',240)

imax = rs.GetInteger('input number in x direction',10)

jmax = rs.GetInteger('input number in y direction',10)

for frame in range(frameNum):

for i in range(imax):

for j in range(jmax):

x = i\*5+(rnd.random()\*frame/20)

y = j\*5+(rnd.random()\*frame/20)

z=0

ptDict[(i,j)]=(x,y,z)

print ptDict[i,j]

#first pattern

for i in range(imax):

for j in range(jmax):

if i>0 and j>0:

crvList.append(rs.AddCurve((ptDict[(i-1,j-1)],ptDict[(i,j)],ptDict[(i,j-1)],ptDict[(i-1,j)]),3))

crvList.append(rs.AddCurve((ptDict[(i,j)],ptDict[(i-1,j-1)],ptDict[(i-1,j)],ptDict[(i,j-1)]),3))

crvList.append(rs.AddCurve((ptDict[(i,j)],ptDict[(i-1,j-1)],ptDict[(i,j-1)],ptDict[(i-1,j)]),3))

crvList.append(rs.AddCurve((ptDict[(i-1,j-1)],ptDict[(i,j)],ptDict[(i-1,j)],ptDict[(i,j-1)]),3))

file\_location="C:\\Users\\seyma\\Desktop\\render\\"

def render(file\_locaton,sequence):

file\_name= str(int(sequence)).zfill(5)

file\_path = " " + file\_location + file\_name + ".png"

rs.Command("\_-ViewCaptureToFile" + file\_path + " \_Enter")

render(file\_location,frame)

rs.DeleteObjects(crvList)

##ASSIGNMENT 04 ##

import rhinoscriptsyntax as rs

import random as rnd

ptDict={}

ptList=[]

crvList01=[]

crvList02=[]

def pointMTX(imax,jmax,kmax):

for i in range(imax):

for j in range(jmax):

for k in range(kmax):

x=i\*5 +(rnd.random())\*7

y=j\*5

z=k\*5 +(rnd.random())\*3

point=(x,y,z)

#rs.AddPoint(point)

ptDict[(i,j,k)]=point

if i>0 and j>0 and k>0:

center\_of\_module =MidPt(ptDict[(i-1,j-1,k-1)],ptDict[(i,j,k)])

crv1= rs.AddCurve((ptDict[(i-1, j, k)],ptDict[(i, j, k)],

ptDict[(i, j, k-1)]),1)

crv2=rs.AddCurve((ptDict[(i-1, j, k)],center\_of\_module,

ptDict[(i-1, j-1, k-1)],ptDict[(i, j-1, k-1)]))

surface=rs.AddLoftSrf((crv1,crv2))

rs.ObjectColor(surface,(158+5\*i,158,150+7\*i))

rs.DeleteObjects(crv1)

rs.DeleteObjects(crv2)

def MidPt(PT01, PT02):

#clear all data being held in point variable

point = None

#calculate mid-point position from input point data

point = [(PT01[0] + PT02[0]) / 2,(PT01[1] + PT02[1]) / 2,

(PT01[2] + PT02[2]) / 2,]

#return mid-point to main() function where MidPt() function was called

return point

def main():

imax= rs.GetInteger("enter x value",10)

jmax= rs.GetInteger("enter y value",4)

kmax= rs.GetInteger("enter z value",10)

rs.EnableRedraw(False)

pointMTX(imax,jmax,kmax)

rs.EnableRedraw(True)

main()

##ASSIGNMENT 05 ##

import rhinoscriptsyntax as rs

import random as rnd

def SurfacePoints(STRSRF, INTU, INTV):

ptMTX = {}

srfNorm01 = {}

Udomain = rs.SurfaceDomain(STRSRF,0)

Vdomain = rs.SurfaceDomain(STRSRF,1)

stepU = (Udomain[1] - Udomain[0])/INTU

stepV = (Vdomain[1] - Vdomain[0])/INTV

#PLOT POINTS ON SURFACE

for i in range(INTU+1):

for j in range(INTV+1):

#define u and v in terms of step values and i and j

u = Udomain[0] + stepU \* i #+(rnd.random())

v = Vdomain[0] + stepV \* j #+(rnd.random())

point = rs.EvaluateSurface(STRSRF, u, v)

ptMTX[(i,j)] = point

#find surface normal(vector) at parameter

vecNorm = rs.SurfaceNormal(STRSRF, (u, v))

vecNorm = rs.VectorUnitize(vecNorm)

vecNorm = rs.VectorScale(vecNorm,1.5) #1-distance/20)

#SAVE FIRST POSITION OF vecNorm IN DICTIONARY srfNorm01

srfNorm01[(i,j)] = rs.PointAdd(vecNorm,point)

GenerateGeometry(ptMTX, srfNorm01, INTU, INTV)

def GenerateGeometry(ptMTX, srfNorm, INTU, INTV):

#LOOP TO CREATE GEOMETRY

for i in range(INTU+1):

for j in range(INTV+1):

if i > 0 and j > 0:

module=[]

center= MidPt(ptMTX[(i,j)],srfNorm[(i-1,j-1)])

crv=[]

rail=[]

rail.append(rs.AddCurve((ptMTX[(i,j)],center,ptMTX[(i-1,j)]),3))

rail.append(rs.AddCurve((srfNorm[(i,j)],center,srfNorm[(i-1,j)]),3))

crv.append(rs.AddCurve((ptMTX[(i,j)],center,srfNorm[(i,j)]),3))

crv.append(rs.AddCurve((ptMTX[(i-1,j)],center,srfNorm[(i-1,j)]),3))

module.append(rs.AddSweep2(rail,crv))

crv=[]

rail=[]

rail.append(rs.AddCurve((ptMTX[(i-1,j-1)],center,ptMTX[(i,j-1)]),3))

rail.append(rs.AddCurve((srfNorm[(i-1,j-1)],center,srfNorm[(i,j-1)]),3))

crv.append(rs.AddCurve((ptMTX[(i-1,j-1)],center,srfNorm[(i-1,j-1)]),3))

crv.append(rs.AddCurve((ptMTX[(i,j-1)],center,srfNorm[(i,j-1)]),3))

module.append(rs.AddSweep2(rail,crv))

crv=[]

rail=[]

rail.append(rs.AddCurve((ptMTX[(i-1,j-1)],center,ptMTX[(i,j-1)]),3))

rail.append(rs.AddCurve((ptMTX[(i-1,j)],center,ptMTX[(i,j)]),3))

crv.append(rs.AddCurve((ptMTX[(i-1,j-1)],center,ptMTX[(i-1,j)]),3))

crv.append(rs.AddCurve((ptMTX[(i,j-1)],center,ptMTX[(i,j)]),3))

module.append(rs.AddSweep2(rail,crv))

crv=[]

rail=[]

rail.append(rs.AddCurve((srfNorm[(i-1,j-1)],center,srfNorm[(i,j-1)]),3))

rail.append(rs.AddCurve((srfNorm[(i-1,j)],center,srfNorm[(i,j)]),3))

crv.append(rs.AddCurve((srfNorm[(i-1,j-1)],center,srfNorm[(i-1,j)]),3))

crv.append(rs.AddCurve((srfNorm[(i,j)],center,srfNorm[(i,j)]),3))

module.append(rs.AddSweep2(rail,crv))

for module in module:

rs.ObjectColor(module,(100-5\*i,15\*i,150+i\*j))

mat\_index = rs.AddMaterialToObject(module)

rs.MaterialColor(mat\_index, (100-5\*i,15\*i,150+i\*j))

def MidPt(PT01, PT02):

#clear all data being held in point variable

point = None

#calculate mid-point position from input point data

point = [(PT01[0] + PT02[0]) / 2,(PT01[1] + PT02[1]) / 2,

(PT01[2] + PT02[2]) / 2,]

#return mid-point to main() function where MidPt() function was called

return point

def main():

strSRFs = rs.GetObjects('select surfaces', rs.filter.surface)

intU = rs.GetInteger('how many U intervals?', 15)

intV = rs.GetInteger('how many V intervals?', 5)

rs.EnableRedraw(False)

for strSRF in strSRFs:

rs.HideObject(strSRF)

SurfacePoints(strSRF, intU, intV)

rs.EnableRedraw(True)

main()