**Sierpinski Fractal Modifications**

After I was able to generate the Sierpinski Fractal in Maya using the Arnold algoritm I then went back into the code and made some alterations in order to create some random shapes. For all the modifications I used the main sierpinski.py script and the #sierpinski\_mel.py to generate altered versions of the Sierpinski fractal. My main method using various combinations of the trigonometric functions, sin, cos, and tan in the def halfstep formula that was generating the points. I also experimented with alternating the number of vertices, the point size, the number of points being generated, and the midpoint coornidates. Below are all the rendered images of my experiments with the sierpinski codes.

### Sierpinski Particle Cubes

Python Code

import random

import math

# Procedure halfstep returns a midpoint between two

# user-defined points

def halfstep(p1, p2):

x = math.sin(float(p1[0] + p2[0])\*3)

y = math.sin(float(p1[1] + p2[1])\*3)

z = math.sin(float(p1[2] + p2[2])\*3)

return [x, y, z]

# Selects a value randomly from the input list

def pickpnt(pnts):

return random.choice(pnts)

#------------------------------------------------

def sierpinski(verts, seed\_pnt, num):

data = []

for n in range(num):

vert = pickpnt(verts)

seed\_pnt = halfstep(vert, seed\_pnt)

data.append(seed\_pnt)

return data

if \_\_name\_\_ == '\_\_main\_\_':

print "Debug"

verts = [ [0,0,1], [1,0,-1], [-1,0,-1], [0,1.5,-0.2], [0,1,0] ]

seed\_pnt = [0,0.5,0]

pnts = sierpinski(verts, seed\_pnt, 100)

print pnts

#------------------------------------------------------------------------------

# sierpinski\_mel.py

from sierpinski import sierpinski

from ass\_header import header

from random import uniform

def getBounds(verts):

minx = 77777

miny = 77777

minz = 77777

maxx = -77777

maxy = -77777

maxz = -77777

for vert in verts:

minx = min(minx, vert[0])

miny = min(miny, vert[1])

minz = min(minz, vert[2])

maxx = max(maxx, vert[0])

maxy = max(maxy, vert[1])

maxz = max(maxz, vert[2])

return [minx,miny,minz,maxx,maxy,maxz]

def sierpinski\_arnold(data\_path, num):

verts = [ [0,0,1], [1,0,-1], [-1,0,-1], [0,1.5,-0.2], [0,1,0] ]

seed\_pnt = [0,0.5,0]

data = sierpinski(verts, seed\_pnt, num)

bounds = getBounds(verts)

# Next the radii

f.write('radius %d 1 FLOAT\n' % len(data))

for pnt in data:

f.write(' %f \n' % 0.002)

### Sierpinski Particle Beams

import random

import math

# Procedure halfstep returns a midpoint between two

# user-defined points

def halfstep(p1, p2):

x = math.tan(float(p1[0] + p2[0])/2)

y = math.tan(float(p1[1] + p2[1])/2)

z = math.sin(float(p1[2] + p2[2])/2)

return [x, y, z]

# Selects a value randomly from the input list

def pickpnt(pnts):

return random.choice(pnts)

#------------------------------------------------

def sierpinski(verts, seed\_pnt, num):

data = []

for n in range(num):

vert = pickpnt(verts)

seed\_pnt = halfstep(vert, seed\_pnt)

data.append(seed\_pnt)

return data

if \_\_name\_\_ == '\_\_main\_\_':

print "Debug"

verts = [ [0,0,1], [1,0,-1], [-1,0,-1], [0,1.5,-0.2], [0,2,1], [0,-2,1] ]

seed\_pnt = [0,0.5,0]

pnts = sierpinski(verts, seed\_pnt, 200)

print pnts

#-----------------------------------------------------------------------------------------

# sierpinski\_mel.py

from sierpinski import sierpinski

from ass\_header import header

from random import uniform

def getBounds(verts):

minx = 99999

miny = 99999

minz = 99999

maxx = -99999

maxy = -99999

maxz = -99999

for vert in verts:

minx = min(minx, vert[0])

miny = min(miny, vert[1])

minz = min(minz, vert[2])

maxx = max(maxx, vert[0])

maxy = max(maxy, vert[1])

maxz = max(maxz, vert[2])

return [minx,miny,minz,maxx,maxy,maxz]

def sierpinski\_arnold(data\_path, num):

verts = [ [0,0,1], [1,0,-1], [-1,0,-1], [0,1.5,-0.2], [0,2,1], [0,-2,1] ]

seed\_pnt = [0,0.5,0]

data = sierpinski(verts, seed\_pnt, num)

bounds = getBounds(verts)

# Next the radii

f.write('radius %d 1 FLOAT\n' % len(data))

for pnt in data:

f.write(' %f \n' % 0.001)

### Sierpinski Triangles

import random

import math

# Procedure halfstep returns a midpoint between two

# user-defined points

def halfstep(p1, p2,):

x = math.sin(float(p1[0] + p2[0])/2)

y = math.cos(float(p1[1] + p2[1])/2)

z = math.sin(float(p1[2] + p2[2])/2)

return [x, y, z]

# Selects a value randomly from the input list

def pickpnt(pnts):

return random.choice(pnts)

#------------------------------------------------

def sierpinski(verts, seed\_pnt, num):

data = []

for n in range(num):

vert = pickpnt(verts)

seed\_pnt = halfstep(vert, seed\_pnt)

data.append(seed\_pnt)

return data

if \_\_name\_\_ == '\_\_main\_\_':

print "Debug"

verts = [ [0,0,1], [1,0,-1], [-1,0,-1], [0,1.5,-0.2], [0,2,1], [0,-2,1] ]

seed\_pnt = [0,0,0]

pnts = sierpinski(verts, seed\_pnt, 200)

print pnts

#-----------------------------------------------------------------------------------------

# sierpinski\_mel.py

from sierpinski import sierpinski

from ass\_header import header

from random import uniform

def getBounds(verts):

minx = 99999

miny = 99999

minz = 99999

maxx = -99999

maxy = -99999

maxz = -99999

for vert in verts:

minx = min(minx, vert[0])

miny = min(miny, vert[1])

minz = min(minz, vert[2])

maxx = max(maxx, vert[0])

maxy = max(maxy, vert[1])

maxz = max(maxz, vert[2])

return [minx,miny,minz,maxx,maxy,maxz]

def sierpinski\_arnold(data\_path, num):

verts = [ [0,0,1], [1,0,-1], [-1,0,-1], [0,1.5,-0.2] ]

seed\_pnt = [0,0,0]

data = sierpinski(verts, seed\_pnt, num)

bounds = getBounds(verts)

# Next the radii

f.write('radius %d 1 FLOAT\n' % len(data))

for pnt in data:

f.write(' %f \n' % 0.003)

### Sierpinski Cubes Two

import random

import math

# Procedure halfstep returns a midpoint between two

# user-defined points

def halfstep(p1, p2,):

x = math.sin(float(p1[0] + p2[0])/3)

y = math.sin(float(p1[1] + p2[1])/3)

z = math.sin(float(p1[2] + p2[2])/3)

return [x, y, z]

# Selects a value randomly from the input list

def pickpnt(pnts):

return random.choice(pnts)

#------------------------------------------------

def sierpinski(verts, seed\_pnt, num):

data = []

for n in range(num):

vert = pickpnt(verts)

seed\_pnt = halfstep(vert, seed\_pnt)

data.append(seed\_pnt)

return data

if \_\_name\_\_ == '\_\_main\_\_':

print "Debug"

verts = [ [1,0,1], [-1,0,1], [1,0,-1], [-1,0,-1], [1,2,1], [-1,2,1], [1,2,-1], [-1,2,-1] ]

seed\_pnt = [0,0,0]

pnts = sierpinski(verts, seed\_pnt, 200)

print pnts

#-----------------------------------------------------------------------------------------

# sierpinski\_mel.py

from sierpinski import sierpinski

from ass\_header import header

from random import uniform

def getBounds(verts):

minx = 99999

miny = 99999

minz = 99999

maxx = -99999

maxy = -99999

maxz = -99999

for vert in verts:

minx = min(minx, vert[0])

miny = min(miny, vert[1])

minz = min(minz, vert[2])

maxx = max(maxx, vert[0])

maxy = max(maxy, vert[1])

maxz = max(maxz, vert[2])

return [minx,miny,minz,maxx,maxy,maxz]

def sierpinski\_arnold(data\_path, num):

verts = [ [1,0,1], [-1,0,1], [1,0,-1], [-1,0,-1], [1,2,1], [-1,2,1], [1,2,-1], [-1,2,-1] ]

seed\_pnt = [0,0.,0]

data = sierpinski(verts, seed\_pnt, num)

bounds = getBounds(verts)

### Sierpinski Diamonds

import random

import math

# Procedure halfstep returns a midpoint between two

# user-defined points

def halfstep(p1, p2,):

x = (float(p1[0] + p2[0]/2.5))

y = math.sin(float(p1[1] + p2[1]+2.5))

z = (float(p1[2] + p2[2]/2.5))

return [x, y, z]

# Selects a value randomly from the input list

def pickpnt(pnts):

return random.choice(pnts)

#------------------------------------------------

def sierpinski(verts, seed\_pnt, num):

data = []

for n in range(num):

vert = pickpnt(verts)

seed\_pnt = halfstep(vert, seed\_pnt)

data.append(seed\_pnt)

return data

if \_\_name\_\_ == '\_\_main\_\_':

print "Debug"

verts = [ [1,0,1], [-1,0,1], [1,0,-1], [-1,0,-1], [0,1,0], [0,-1,0] ]

seed\_pnt = [0,0,0]

pnts = sierpinski(verts, seed\_pnt, 200)

print pnts

#-----------------------------------------------------------------------------------------

# sierpinski\_mel.py

from sierpinski import sierpinski

from ass\_header import header

from random import uniform

def getBounds(verts):

minx = 99999

miny = 99999

minz = 99999

maxx = -99999

maxy = -99999

maxz = -99999

for vert in verts:

minx = min(minx, vert[0])

miny = min(miny, vert[1])

minz = min(minz, vert[2])

maxx = max(maxx, vert[0])

maxy = max(maxy, vert[1])

maxz = max(maxz, vert[2])

return [minx,miny,minz,maxx,maxy,maxz]

def sierpinski\_arnold(data\_path, num):

verts = [ [1,0,1], [-1,0,1], [1,0,-1], [-1,0,-1], [0,1,0], [0,-1,0] ]

seed\_pnt = [0,0,0]

data = sierpinski(verts, seed\_pnt, num)

bounds = getBounds(verts)

# Next the radii

f.write('radius %d 1 FLOAT\n' % len(data))

for pnt in data:

f.write(' %f \n' % 0.01)

### Sierpinski Crosses

import random

import math

# Procedure halfstep returns a midpoint between two

# user-defined points

def halfstep(p1, p2,):

x = math.sin(float(p1[0] + p2[0]/2))

y = math.sin(float(p1[1] + p2[1]/2))

z = math.sin(float(p1[2] + p2[2]/2))

return [x, y, z]

# Selects a value randomly from the input list

def pickpnt(pnts):

return random.choice(pnts)

#------------------------------------------------

def sierpinski(verts, seed\_pnt, num):

data = []

for n in range(num):

vert = pickpnt(verts)

seed\_pnt = halfstep(vert, seed\_pnt)

data.append(seed\_pnt)

return data

if \_\_name\_\_ == '\_\_main\_\_':

print "Debug"

verts = [ [1,0,1], [-1,0,1], [1,0,-1], [-1,0,-1], [0,1,0], [0,-1,0] ]

seed\_pnt = [0,0,0]

pnts = sierpinski(verts, seed\_pnt, 200)

print pnts

#-----------------------------------------------------------------------------------------

# sierpinski\_mel.py

from sierpinski import sierpinski

from ass\_header import header

from random import uniform

def getBounds(verts):

minx = 99999

miny = 99999

minz = 99999

maxx = -99999

maxy = -99999

maxz = -99999

for vert in verts:

minx = min(minx, vert[0])

miny = min(miny, vert[1])

minz = min(minz, vert[2])

maxx = max(maxx, vert[0])

maxy = max(maxy, vert[1])

maxz = max(maxz, vert[2])

return [minx,miny,minz,maxx,maxy,maxz]

def sierpinski\_arnold(data\_path, num):

verts = [ [1,0,1], [-1,0,1], [1,0,-1], [-1,0,-1], [0,1,0], [0,-1,0] ]

seed\_pnt = [0,0,0]

data = sierpinski(verts, seed\_pnt, num)

bounds = getBounds(verts)

# Next the radii

f.write('radius %d 1 FLOAT\n' % len(data))

for pnt in data:

f.write(' %f \n' % 0.003)