Menger 2D Algorithm

1 set two points that define the bounding box of a rectangle,

2 divide the rectangle into 9 sub-rectanges,

3 discard the data for the central sub-rectangle,

4 store the bounding boxes of the remaining 8 sub-rectangles.

Steps 1 to 4 make the first level of sub-division of the 3D Menger Sponge aka. the Menger 2D Square.

Below is the python code for creating a Menger 2D square mel script.

# Given a rectange specified by two opposite vertices

# the proc implemented by this module divides the rectangle

# into 9 sub-rectangles.

import random

from random import uniform

import math

menger = []

def divide(rect, depth):

 if depth == 0:

 menger.append(rect)

 return

 x0,y0,z0,x1,y1,z1 = rect

 rects = [] # used only for the local storage of the sub-rectangles.

 # Create 9 sub-rectangles.

 w = float(x1 - x0)/3

 h = float(y1 - y0)/3

 d = float(z1 - z0)/3

 # columns 1, 2 and 3

 rects.extend(divide\_column(x0, y0,z0, w,h,d))

 rects.extend(divide\_column(x0 + w, y0,z0, w,h,d))

 rects.extend(divide\_column(x0 + w + w, y0,z0, w,h,d))

 # Remove the central rect

 rects.pop(4)

 # Use recursion to further subdivide the sub-rects

 for subrect in rects:

 divide(subrect, depth - 1)

def divide\_column(x0,y0,z0, w,h,d):

 x,y,z = x0,y0,z0

 X,Y,Z = x + w, y + h, z + d

 subrects = []

 for n in range(3):

 subrects.append( [x,y,z, X,Y,Z] )

 z,Z = z + d, Z + d

 return subrects

##============================================================

# polyPlane -w 1 -h 1 -sx 1 -sy 1 -ax 0 1 0 -cuv 2 -ch 1;

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

 start\_rect = [-1,0,-1, 1,0,1]

 divide(start\_rect, 4)

 # Output to a mel script for visualizing in Maya

 f = open('/home/njones26/mount/stuhome/tech312/python/Menger Sponge/menger.mel', 'w')

 f.write('string $names[];\n')

 f.write('string $obj[];\n')

 count = 0

 for rect in menger:

 x,y,z,X,Y,Z = rect

 f.write('$obj = `polyPlane -sx 1 -sy 1 -w %f -h %f -ax 0 1 0`;\n' % (X-x,Z-z))

 f.write('$names[%d] = $obj[0];\n' % count)

 f.write('move %f %f %f;\n\n' % (x,y,z))

 count += 1

 f.write('group $names;\n')

 f.close()