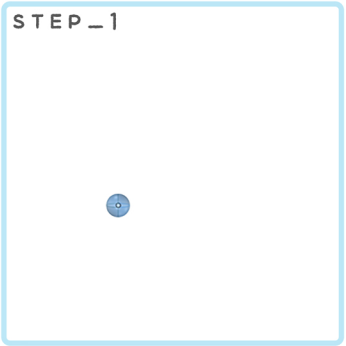
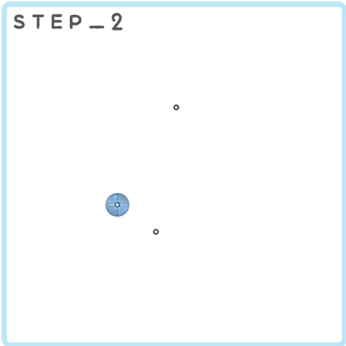
CODE STEPS

STEP_0: Select a point.

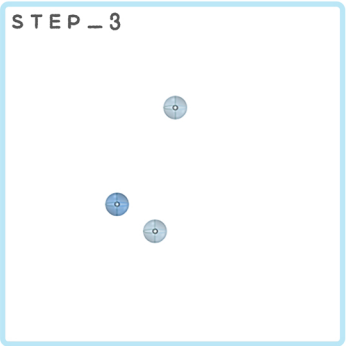


STEP_1: Using the selected point from STEP_0 as the centroid, create a sphere by defining the radius (see line 87). Define the color and transparency as well.

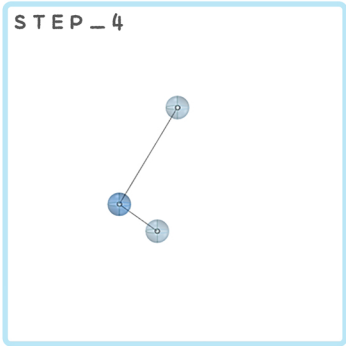


STEP_2: Set up *uniform random* (rnd.uniform) by defining the max and min for each axis. The second set of new points will use the original point and add the *rnd.uniform* number for each axis.

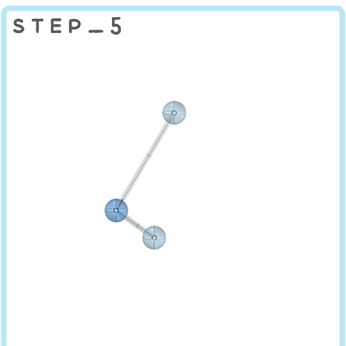
Looking at the current code n = 2, so 2 new points will be created.



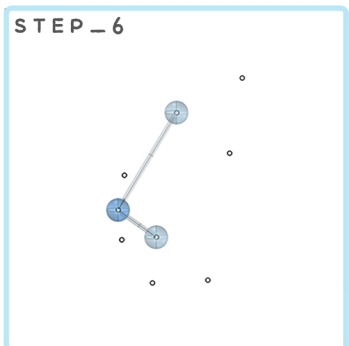
STEP_3: Using these new points as centroids create a new sphere. Looking at line 80 we see "sph_radius*0.75". The radius of the first sphere was 1.25. The sphere radius for these spheres will be 1.25*0.75 = 0.9375. Define the color and transparency here as well.



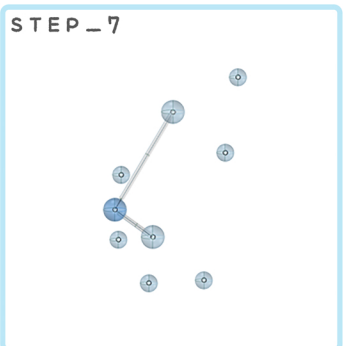
STEP_4: Connect the new points to the original point with a line.



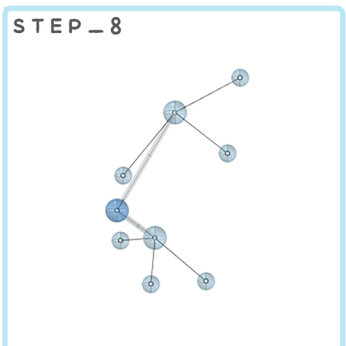
STEP_5: Pipe the two lines by defining the radius (set at 0.25 in line 94) and other pipe details defined in line 71. Line 78 is used to to determine whether or not it will loop again and "count+1" in line 80 is used to keep the current count. Since the current count is 2 and the depth of recursion is defined as 3 we go for another loop.



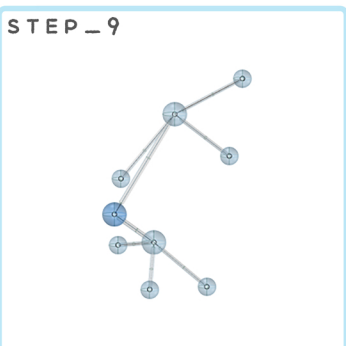
STEP_6: Looking at line 80 we see "n+1". Looking at STEP_2 we see n = 2, which meant 2 new points were made. Here in STEP_6 each branch of points will create 3 new points since n + 1 = 3 at this step.



STEP_7: Using the new points from STEP_7 create a sphere. Using the same equation used in STEP 3 we know the radius of these new sphere will be 0.9375*0.75 = 0.703125. Each step the spheres will get 0.75 smaller. The color and transparency for these have been present in STEP_3.



STEP_8: Points from STEP_6 and STEP_2 are connected with a line.



STEP_3: Using the equation from STEP_5 we can determine the radius of the pipe "pipe_radius*0.75", which equals 0.1875. The code will loop as long as the current count is less than the depth (see line 78). Since the depth is defined as 3 (see line 94)

VIDEOS

QR code linked to YouTube video of dense cluster in Rhino Artic view.

QR code linked to YouTube video of a colored and semitransparent four loop model in Rhino Render Preview view

```python

##########################################

#ARCH_703_002__VIRTUAL_ENGAGMENT
#PROF: Glenn Wilcox

#PROJECT_1
#STUDENT: Maho Kobayashi

##########################################
##########################################

#BRING IN LIBRARIES
import rhinoscriptsyntax as rs
import random as rnd
import Rhino.Geometry as rg
import scriptcontext as sc

##########################################

#OVERALL GOAL:
## creating a multi directional random branching design consisting of spheres and pipes
## each branch/ iteration the sphere and pipes get ___% smaller than the previous
....
##########################################

#GENERAL REFERENCES:
## http://www.generativedesigncomputing.net/2012/10/3d-point-matrix-rhinoscript-python.html
## https://www.coursera.org/learn/3d-modeling-rhinoscript/lecture/FhFPF/3d-surface-matrix
## https://drive.google.com/drive/folders/1n-yyZ5zrklnriJOPo8NmNrAoBvnc3zNF____3d_branching_tutorial

## https://discourse.mcneel.com/t/how-to-convert-guid-to-point3d-by-python-script/35594/2

## https://stackoverflow.com/questions/29737715/changing-color-of-spheres-within-for-loop-python-rhino

##########################################
##########################################

#STEP_F: create four new random points
#        create sphere using new points as center
#        connect the new points to the origin_pt with a line
#        pipe the connecting lines

def SpherePipeRecursion(origin_pt, count, n, l, h, depth, sph_radius, pipe_radius):
    '''
    origin_pt = starting point / the get point
    count = current level of recursion
    n = number of new points created
    l = lowest of the random range
    h = highest of the random range
    depth = depth of recursion
    sph_radius = radius of the current sphere
    pipe_radius = radius of the current pipe
    '''

    for i in range(n):
        
        #setting up the uniform random of each axis that will be used to create n number of new points
        x = rnd.uniform(l,h)
        y = rnd.uniform(l,h)
        z = rnd.uniform(l,h)
        
        pt = (x + (origin_pt[0]),y + (origin_pt[1]),z + (origin_pt[2]))
        
        new_pt = rs.AddPoint(pt)
        new_pt_coords = rs.coerce3dpoint(new_pt)
        
        #adding sphere based on pt -> piping connecting lines
        sphere_1 = rs.AddSphere(new_pt,sph_radius)
        connecting_line = rs.AddLine(origin_pt, new_pt)
        pipe = rs.AddPipe(connecting_line,0.125, pipe_radius,blend_type=1, cap=2, fit=False)
        
        #set color of sphere and pipe
        rs.ObjectColor (sphere_1, [135,206,250,70])
        rs.ObjectColor (pipe, [255,255,255,70])
        
        #recursively call function if the current count is under the defined 'depth'
        if count < depth:
            
            SpherePipeRecursion(new_pt_coords, count+1, n+1, l*0.75, h*0.75, depth, sph_radius*0.75, pipe_radius*0.75)
        
def main():
    
    #STEP_0: select pt -> create sphere based on the pt -> define rad -> define color
    
    start_pt = rs.GetPoint('select a point')
    origin_rad = 1.25
    origin_sphere = rs.AddSphere(start_pt, origin_rad)
    
    rs.ObjectColor (origin_sphere, [30,144,255,90])
    #fourth digit is the transparency

    #STEP_1: recall function -> most paremeters set here for the sake of flexibility and ease of use
    SpherePipeRecursion(start_pt, 0, 2, -10, 10, 3, origin_rad,0.25)
    
main()
```