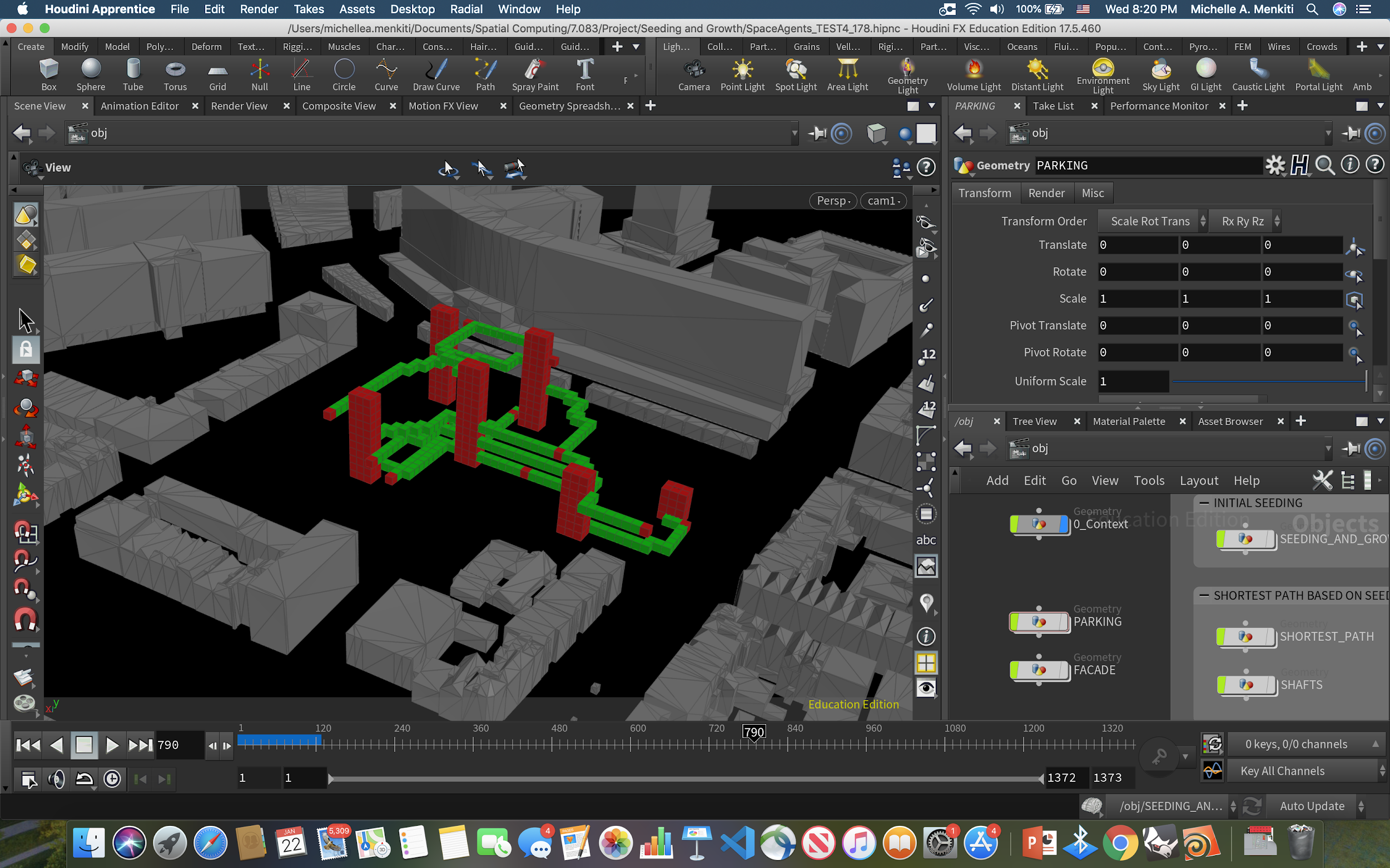
SHORTEST PATH

//Date: 24-01-2019

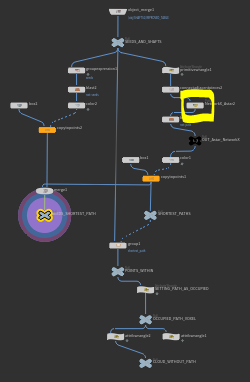
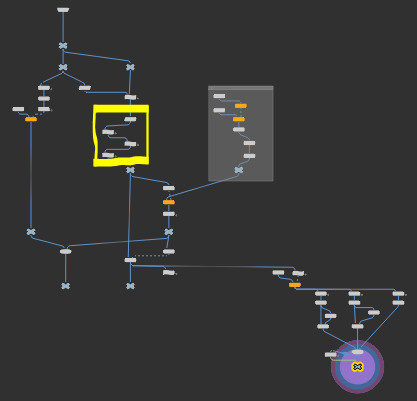
//Authors: Jolt Wiersma, Michelle A. Menkiti, Arthur Masure

//Adapted from work by: Shervin Azadi, Dr.ir. Pirouz Nourian, Hans Hoogenboom

//Purpose: Determines the shortest horizontal path between seeds by intercepting a vertical circulation shaft first



SHORTEST HORIZONTAL PATH TO CLOSEST SHAFTS AND TO END SEED



//A: PREPARE NEW ATTRIBUTE TO HOLD THE POINT ID OF THE CLOSEST SHAFT POINT TO ALL START POINT SEEDS

//get number of shafts

int shaft\_count = 0;

for (int point=0; point < npoints(0); point++)

{

if (point(0,"shaft\_origins",point) > -1)

{

shaft\_count = shaft\_count+1;

}

}

//iterate through that number

for (int shaft=0; shaft < shaft\_count; shaft++)

{

//preparing attribute name

string attribute\_name = "Shaft" + itoa(shaft);

for (int point=0; point < npoints(0); point++)

{

//set default value

setpointattrib(0,attribute\_name,point,-1,"set");

}

}

//B: FOR ALL START POINT SEEDS, FIND CLOSEST SHAFT AND SET IN NEW ATTRIBUTE

//if its a parent

if (point(0,"parent",@ptnum) > -1)

{

//closest shaft and min distance

int closest\_shaft = -1;

float final\_distance = 1000000;

//flatten current parent

vector parent = point(0,"P",@ptnum);

parent \*= set(1,0,1);

//loop through all points in cloud

for (int point = 0; point < npoints(0); point++)

{

//if you find a shaft

int shaft\_count = point(0,"shaft\_origins",point);

if (shaft\_count > -1)

{

//shaft position

vector shaft = point(0,"P",point);

//record distance

float curr\_distance = distance(parent,shaft);

//if curr\_distance is less that final\_distance, set as new final

if (curr\_distance < final\_distance)

{

final\_distance = curr\_distance;

closest\_shaft = shaft\_count;

}

}

}

//set closest shaft to shaft count

setpointattrib(0,"closest\_shaft",@ptnum,closest\_shaft,"set");

}

//C: FOR ALL START POINT SEEDS, FIND POINT ID IN THE CLOSEST SHAFT THAT IS ON THE SAME LEVEL AS THE START POINT SEED AND SET IN ATTRIBUTES CREATED IN STEP A

//FOR EACH PARENT

if (point(0,"parent",@ptnum) > -1)

{

//GET ITS Y VALUE

vector pos = point(0,"P",@ptnum);

float y\_value = pos.y;

//FOR EACH SHAFT

for (int point=0; point<npoints(0); point++)

{

int shaft\_count = point(0,"shaft\_origins",point);

if (shaft\_count > -1)

{

//preparing attribute name

string attribute\_name = "Shaft" + itoa(shaft\_count);

//GET POINT ON SAME YVALUE

vector pos\_1 = point(0,"P",point);

int IPX = point(0,"IPX",point);

int IPY = point(0,"IPY",point);

int IPZ = point(0,"IPZ",point);

pos\_1 += set(0,y\_value,0);

int y\_val = y\_value/3.0;

int point\_ID = -2;

//FIND POINT ID OF POINT IN SHAFT

for (int i=0; i<npoints(0); i++)

{

int IPX\_1 = point(0,"IPX",i);

int IPY\_1 = point(0,"IPY",i);

int IPZ\_1 = point(0,"IPZ",i);

if (IPX\_1 == IPX)

{

if (IPZ\_1 == IPZ)

{

if (IPY\_1 == y\_val)

{

point\_ID = i;

}

}

}

//Setting point

setpointattrib(0,attribute\_name,@ptnum,point\_ID,"set");

}

}

}

}

//D: CREATE THE SHORTEST PATH FROM THE START POINT SEED TO THE CLOSEST SHAFT, THEN FROM THAT SHAFT TO THE END POINT SEED

//IN A PYTHON NODE

#set geo to the geometry connected to the node (connect adjacent pieces)

node = hou.pwd()

geo = node.geometry()

#import the numpy library

import numpy as np

import os

hippath = os.environ['HIP']

#get connection data from csv file

fname = "Function\_Relationship\_3.csv"

fpath = os.path.join(hippath, fname)

print(fpath)

# fhandle = open(fpath, "r")

connection\_weight = np.genfromtxt(fpath, dtype=int, delimiter=",")

#initiate the graph

import networkx as nx

G = nx.Graph()

#iterate over the points of geo and add them as nodes of the graph

for point in geo.points():

#retrieve the point number

id = point.number()

#add the node to the graph (G)

G.add\_node(id)

#iterate over the prims of geo and add them as edges of the graph

for prim in geo.prims():

#retrieve the prim number (necessary for later)

id = prim.number()

#getting the list of the points

pnts = prim.points()

#add the edge to the graph (G) by creating it from points in pnts

G.add\_edge(pnts[0].number(),pnts[1].number())

#set seed\_ids to the point ids of the points that are not equal to -1 in the attribute "Seed" inside the geo geometry attached to the node

attrib\_name = geo.findPointAttrib("parent")

attrib\_closest\_shaft = geo.findPointAttrib("closest\_shaft")

attrib\_shaft0 = geo.findPointAttrib("Shaft0")

attrib\_shaft1 = geo.findPointAttrib("Shaft1")

attrib\_shaft2 = geo.findPointAttrib("Shaft2")

attrib\_shaft3 = geo.findPointAttrib("Shaft3")

attrib\_shaft4 = geo.findPointAttrib("Shaft4")

attrib\_shaft5 = geo.findPointAttrib("Shaft5")

seed\_ids = [0 for i in range(13)]

shaft\_index = [0 for i in range(13)]

shaft\_points = [0 for i in range(13)]

#go through all points and getting the point IDS of parents

for point in geo.points():

myseed = point.intAttribValue(attrib\_name)

myclose\_shaft = point.intAttribValue(attrib\_closest\_shaft)

myShaft0 = point.intAttribValue(attrib\_shaft0)

myShaft1 = point.intAttribValue(attrib\_shaft1)

myShaft2 = point.intAttribValue(attrib\_shaft2)

myShaft3 = point.intAttribValue(attrib\_shaft3)

myShaft4 = point.intAttribValue(attrib\_shaft4)

myShaft5 = point.intAttribValue(attrib\_shaft5)

if myseed != -1:

#get the point ID

ptnum = point.number()

##add the point ID at index parent

seed\_ids[myseed] = ptnum

shaft\_index[myseed] = myclose\_shaft

#make a list of shaft ID

parent\_points = [myShaft0,myShaft1,myShaft2,myShaft3,myShaft4,myShaft5]

#add list to shaft points

shaft\_points[myseed] = parent\_points

#group the points that are in the path

path\_group = geo.createPointGroup("path")

print("reset")

#looping through the table and finding connections equal to 1

for i, row in enumerate(connection\_weight):

for j, col in enumerate(row):

if col == 1:

#setting you start and end nodes

start\_i = seed\_ids[i]

end\_f = seed\_ids[j]

#print(start,end)

#get temp end point

m = shaft\_index[i]

end\_i = shaft\_points[i][m]

#if there is a point to go to

if end\_i != -2:

#Part 1

#find the shortest path using A-star algorithm

path\_nodeid\_s = nx.astar\_path(G, start\_i, end\_i)

#retrieve the point object given the point number

path\_points\_s = [geo.points()[id] for id in path\_nodeid\_s]

path\_group.add(path\_points\_s)

#get temp start point

start\_f = shaft\_points[j][m]

#if there is a point to go to

if start\_f != -2:

#Part 2

#find the shortest path using A-star algorithm

path\_nodeid\_f = nx.astar\_path(G, start\_f, end\_f)

#retrieve the point object given the point number

path\_points\_f = [geo.points()[id] for id in path\_nodeid\_f]

path\_group.add(path\_points\_f)