PROXIMITY TO EDGE

//Date: 19-12-2019

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

//Purpose: To create and set normalized attribute values for all voxels, based on their proximity boundary of the voxel envelope.

//INPUT 0 = CONNECT ADJACENCY NETWORK

//RUN OVER DETAILS

**WRANGLER 1**

**//determine the boundary voxels, and set their attributes to 1**

int num\_points = npoints(0);

addattrib(0, "point", "proximity\_edge", 0.0);

int edgepoints[];

//iterate through all of the points

for (int point = 0; point < num\_points; point++)

{

//identify neighbours and determine number of neighbours

int num\_neighbours[] = neighbours(0, point);

int neigh = len(num\_neighbours);

//get y position of voxel

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

float y\_value = pos.y;

//if boundary point

if (neigh < 6)

{

setpointattrib(0, "proximity\_edge", point, 1.0, "set");

//append(edgepoints, voxel\_id);

}

//setting voxels that are on ground floor but not on boundary to 0

if (neigh < 6 && y\_value == 0 && neigh == 5)

{

setpointattrib(0, "proximity\_edge", point, 0.0, "set");

}

}

//INPUT 0 = WRANGLER 1

//RUN OVER DETAILS

**WRANGLER 2**

**//setting distances to interior points**

int num\_points = npoints(0);

int edgepoints[];

//iterating through all points, adding all edge points to a list

for (int point = 0; point < num\_points; point++)

{

float inside = attrib(0,"point","proximity\_edge",point);

if (inside == 1.0)

{

append(edgepoints,point);

}

}

//iterating through all points

for (int point = 0; point < num\_points; point++)

{

//getting position of point

float inside = attrib(0,"point","proximity\_edge",point);

vector current\_int = attrib(0,"point","P",point);

//if inside

if (inside == 0.0)

{

float temp\_dist = 10000000.0;

//looping through edge points

for (int edge = 0; edge < len(edgepoints); edge++)

{

vector current\_ext = attrib(0,"point","P",edgepoints[edge]);

if (current\_int[1] == current\_ext[1])

{

float dist = distance(current\_int,current\_ext);

if (dist < temp\_dist)

{

temp\_dist = dist;

}

}

}

//setting distance to edge

setpointattrib(0,"proximity\_edge",point,temp\_dist,"set");

}

}

//INPUT 0 = WRANGLER 2

//RUN OVER DETAILS

**WRANGLER 3**

**//inverting values**

int num\_points = npoints(0);

float max\_prox = 0;

//finding the maximum value

//iterating through all points

for (int point = 0; point < num\_points; point++)

{

float inside\_prox = attrib(0,"point","proximity\_edge",point);

if (inside\_prox > max\_prox)

{

max\_prox = inside\_prox;

}

}

//reverting all the values

for (int point = 0; point < num\_points; point++)

{

float inside\_dist = attrib(0,"point","proximity\_edge",point);

float new\_dist = (max\_prox + 1) - inside\_dist;

//setting the distance

setpointattrib(0,"proximity\_edge",point, new\_dist);

}