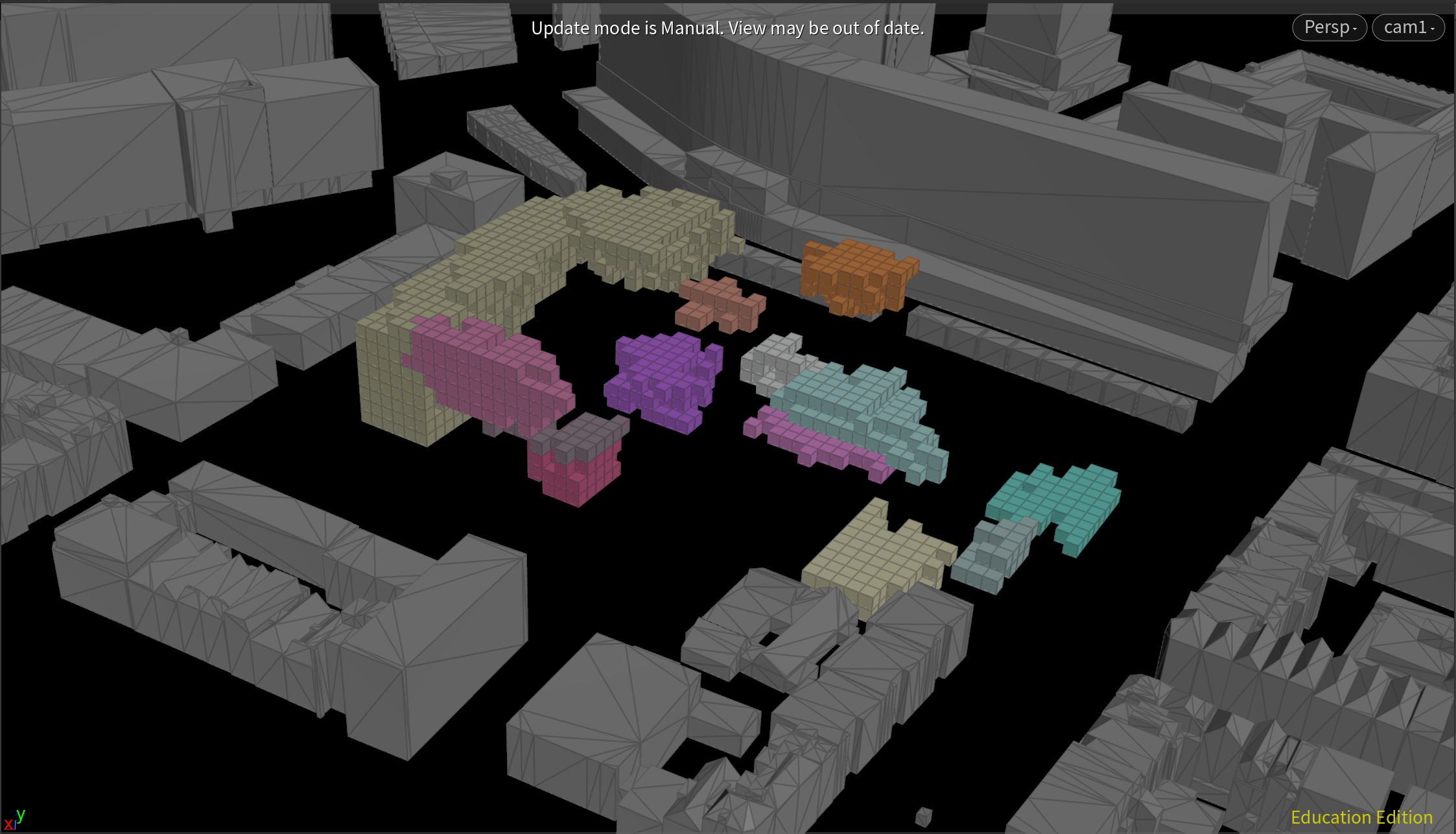
GROWING

//Date: 24-01-2019

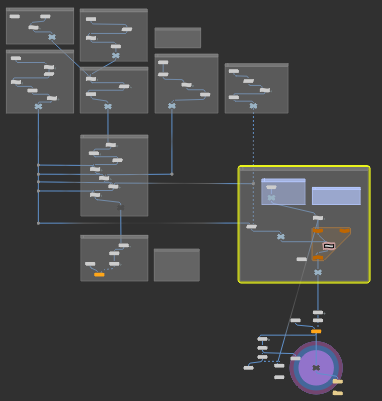
//Authors: Jolt Wiersma, Michelle A. Menkiti, Arthur Masure

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

//Purpose: To grow the initial seed voxels based on data given to it by a .csv file and to give each function a squareness



GROWTH OF FUNCTIONS IMPLEMENTING GROWTH HEIGHT, AREA, AND SQUARENESS



//user input

float squareness\_factor = chf("squareness\_factor");

//GET NUMBER OF PARENTS/AGENTS (FUNCTIONS)

int num\_agents = npoints(1);

//FOR EVERY PARENT SEED IN THE ENVELOPE

for (int Agent = 0; Agent < num\_agents; Agent++)

{

//prepare wp name

string wp\_name = "wp" + itoa(Agent);

//prepare a list for y-values

int y\_values[];

//get number of children allocated to that parent (agent)

int num\_children = findattribvalcount(0, "point", "parent", Agent);

//get desired area for that parent (agent)

int desired\_area = point(1, "area", Agent);

//get desired number of floors to grow on for that parent (agent)

int number\_of\_floors = point(1,"GrowthHeight",Agent);

//IF THE DESIRED AREA (UNITS: NUMBER OF VOXELS) IS GREATER THAN THE NUMBER OF CHILDREN

if (desired\_area > num\_children)

{

//CREATE A LIST OF BOUNDARY VOXEL ID POINTS

int agent\_boundary\_id[];

//CREATE A LIST OF BOUNDARY VOXEL WEIGHTED PRODUCTS

float agent\_boundary\_wp[];

//get the y-value of the parent (as float and int)

int seed\_parent\_id = findattribval(0,"point","Seed",Agent, 0);

vector pos = point(0,"P",seed\_parent\_id);

float y\_value = pos.y;

int seed\_IPY = point(0,"IPY",seed\_parent\_id);

y\_values = point(0,"Y\_Values",seed\_parent\_id);

//add the y-value of the parent to the list of y-values

append(y\_values,seed\_IPY);

//FOR EVERY CHILD OF THE PARENT

for (int Child = 0; Child < num\_children; Child++)

{

//GET THE POINT ID OF THE CHILD

int voxel\_id = findattribval(0, "point", "parent", Agent, Child);

//CREATE A LIST OF THE POINT ID'S OF THE NEIGHBOURS OF THE CHILD

int voxel\_neighbours[] = neighbours(0, voxel\_id);

//remove neighbours that are too far away

foreach (int Neigh; voxel\_neighbours)

{

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

float distance = distance(pos\_1,pos);

if (distance > 3)

{

removevalue(voxel\_neighbours,Neigh);

}

}

//FOR EVERY NEIGHBOUR OF THE CHILD

foreach (int Neigh; voxel\_neighbours)

{

//IF THE NEIGHBOUR IS NOT OCCUPIED

if (inpointgroup(0, "occupied", Neigh) == 0)

{

//get the y-value of the neighbour (as float and int)

vector bound\_pos = point(0,"P",Neigh);

float y\_value\_boundary = bound\_pos[1];

int bound\_IPY = point(0,"IPY",Neigh);

//get the weighted product of that neighbour

float neigh\_wp = point(0,wp\_name, Neigh);

setpointattrib(0,wp\_name,Neigh,neigh\_wp\*squareness\_factor,"set");

//IF AGENT DESIRES RANDOM GROWTH (CSV)

if (number\_of\_floors == 0) // csv = 0

{

//ADD THE NEIGHBOUR'S POINT ID TO THE LIST OF BOUNDARY VOXEL POINT ID'S

append(agent\_boundary\_id, Neigh);

//ADD THE NEIGHBOUR'S WEIGHTED PRODUCT TO THE LIST OF BOUNDARY VOXEL WEIGHTED PRODUCTS

append(agent\_boundary\_wp, neigh\_wp);

}

//IF AGENT DESIRES GROWTH ON 1 VOXEL LAYER (CSV)

if (abs(y\_value\_boundary - y\_value) < 0.001 && number\_of\_floors == 1) //y-values are relatively equal and csv = 1

{

//ADD THE NEIGHBOUR'S POINT ID TO THE LIST OF BOUNDARY VOXEL POINT ID'S

append(agent\_boundary\_id, Neigh);

//ADD THE NEIGHBOUR'S WEIGHTED PRODUCT TO THE LIST OF BOUNDARY VOXEL WEIGHTED PRODUCTS

append(agent\_boundary\_wp, neigh\_wp);

}

//IF AGENT DESIRES GROWTH ON MORE THAN 1 OR RANDOM VOXEL LAYERS (CSV)

if (number\_of\_floors > 1); // csv > 1

//IF THE NEIGHBOUR'S Y-VALUE IS IN THE LIST OF PARENT Y-VALUES

if (find(y\_values, bound\_IPY) > -1)

{

//ADD THE NEIGHBOUR'S POINT ID TO THE LIST OF BOUNDARY VOXEL POINT ID'S

append(agent\_boundary\_id, Neigh);

//ADD THE NEIGHBOUR'S WEIGHTED PRODUCT TO THE LIST OF BOUNDARY VOXEL WEIGHTED PRODUCTS

append(agent\_boundary\_wp, neigh\_wp);

}

//OTHERWISE, IF THE LIST OF PARENT Y-VALUES IS 'FULL' (the length of the list is less than the desired number of floors)

else if (len(y\_values) < number\_of\_floors)

{

//ADD THE NEIGHBOUR'S POINT ID TO THE LIST OF BOUNDARY VOXEL POINT ID'S

append(agent\_boundary\_id, Neigh);

//ADD THE NEIGHBOUR'S WEIGHTED PRODUCT TO THE LIST OF BOUNDARY VOXEL WEIGHTED PRODUCTS

append(agent\_boundary\_wp, neigh\_wp);

//ADD THE NEIGHBOUR'S Y-VALUE TO THE LIST OF PARENT Y-VALUES

append(y\_values,bound\_IPY);

}

}

}

}

//Set the maximum as a new child for the parent

//SORT THE LIST OF BOUNDARY VOXEL WEIGHTED PRODUCTS FROM MAXIMUM TO MINIMUM

int sorted\_indecies[] = reverse(argsort(agent\_boundary\_wp));

//SORT THE LIST OF BOUNDARY VOXEL POINT ID'S FROM MAXIMUM TO MINIMUM

int sorted\_boundary\_id[] = reorder(agent\_boundary\_id, sorted\_indecies);

//SET THE CHILD WITH THE MAXIMUM WEIGHTED PRODUCT (FIRST INDEX) AS A PARENT AND AS OCCUPIED

setpointattrib(0, "parent", sorted\_boundary\_id[0], Agent, "set");

setpointgroup(0, "occupied", sorted\_boundary\_id[0], 1, "set");

}

//attribute to check amount and value of y-values in list of y-values for that agent

int seed\_parent\_id = findattribval(0,"point","Seed",Agent, 0);

setpointattrib(0,"Y\_Values",seed\_parent\_id,y\_values,"set");

}