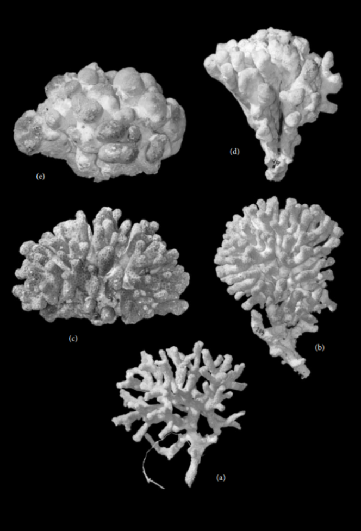
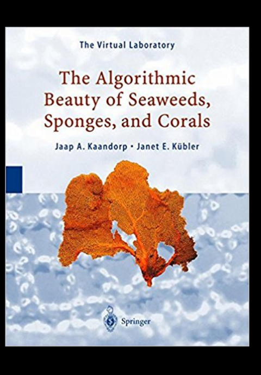
VSFX 375 FINAL HOUDINI VEX

MEREDITH O'MALLEY

For this assignment I decided to utilize one of Houdini's more intuitive tools - vex. This C-based code is used throughout Houdini to create complex algorithms and fractals to the simplest of tasks in the fastest manner. Below I will walk you through on how I utilized this powerful tool to generate procedural assets for a coral reef.



A resource I used to get started and to better understand the structure and growth behavior of the differnt types of coral:

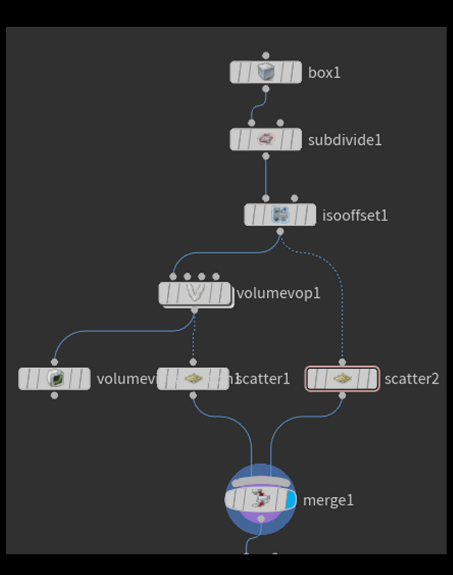


The Algorithmic Beauty of Seaweeds, Sponges and Corals

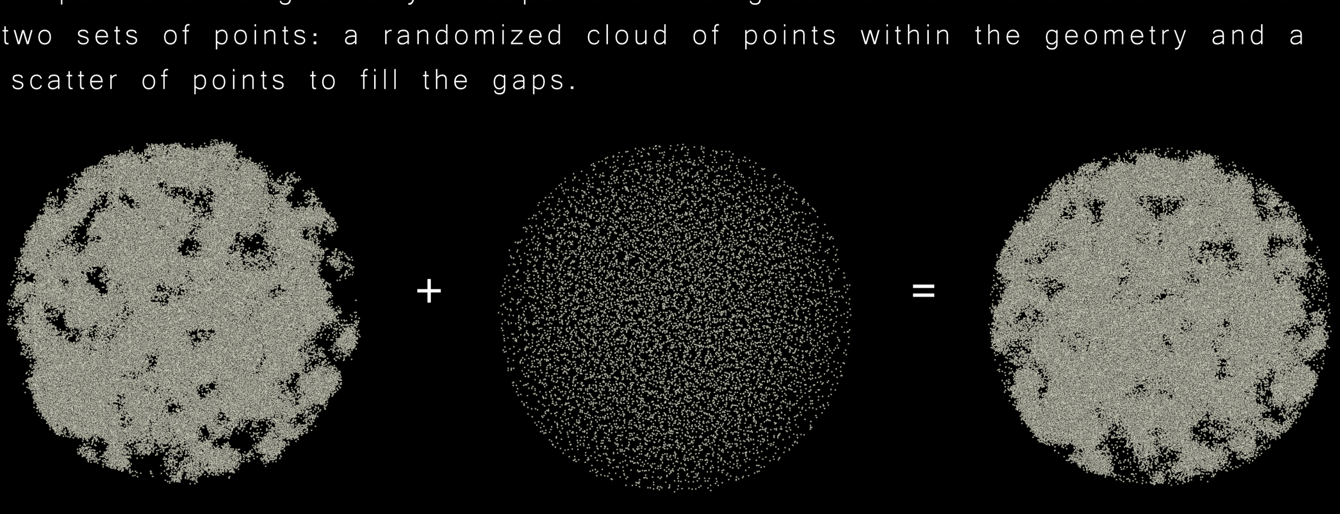
by : Bearbeitet von Jaap A Kaandorp + Janet E Kübler

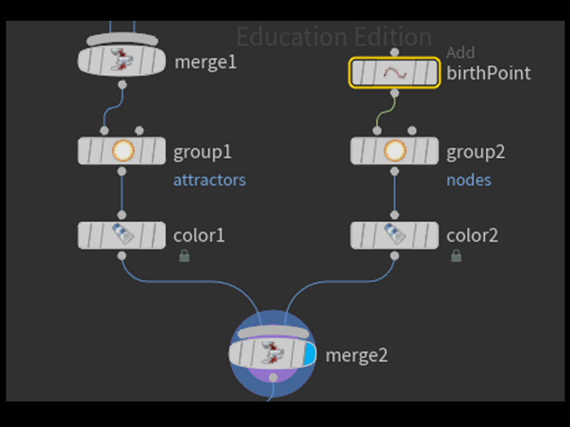
GENERATING CORAL SKELETON

Below is a basic demonstration of the growth and meshing technique for the branching coral structure. Vex was used for both the point attraction algorythm as well as orinting the curves as it grows so that a clean normalized mesh can be generated as well.

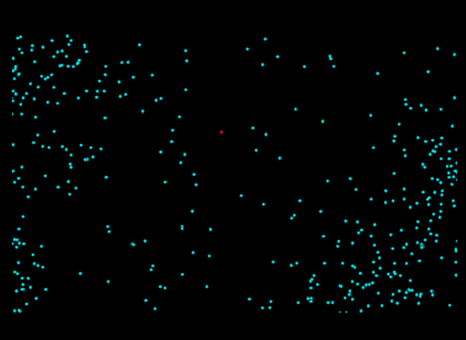


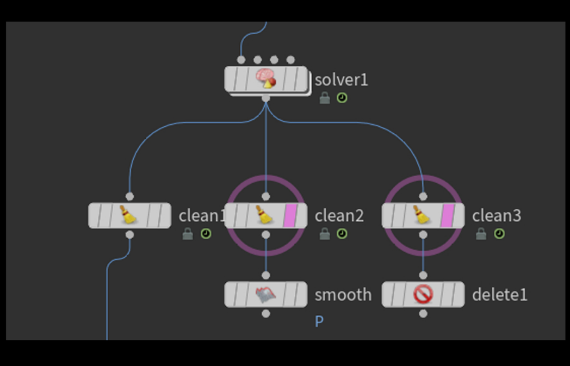
Step 1: Import a solid geometry - capable of being converted into a volume- and scatter two sets of points: a randomized cloud of points within the geometry and a uniform scatter of points to fill the gaps.





Step 2: Now place and merge in a birth point - or several- this will be the root of your generated coral. Colors are assigned for the sake of visibility.

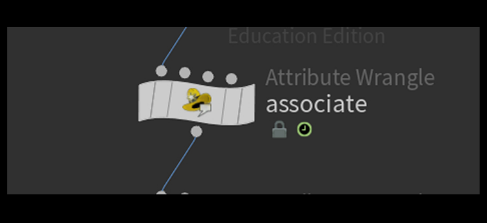




Step 3: Next we will create a solver based on the point cloud be have created. Explinations can be found in the code.

VEXpression

```
//create and initialize array parameter
i[]@associates = array();
```



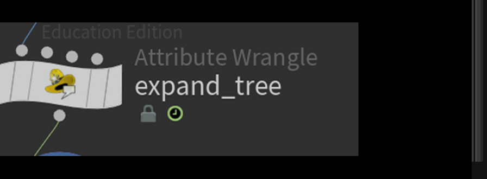
Code | Bindings

Group: attractors
Group Type: Guess from Group
Run Over: Points

VEXpression

```
//// SET VALUES /////

//define parameter for search radius
float infrad = ch("infrad");

//search for nearest point in the nodes group
//gives id of point that is closest
int nearpnt = nearpoint(0, "nodes", @P, infrad);
//create local variable - populate array with single point
int valarr[] = array(@ptnum);

//append makes sure the revaluation of points will
//not overwrite eachother.
setpointattrib(0, "associates", nearpnt, valarr, "append");
```

Attributes to Create: *
Enforce Prototypes
Infrad: 0.03



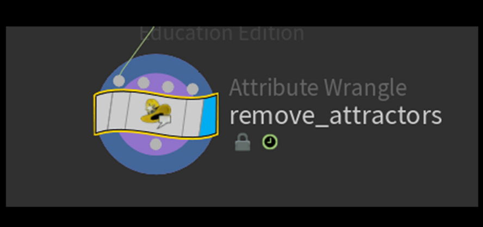
Code | Bindings

Group: nodes
Group Type: Guess from Group
Run Over: Points

VEXpression

```
//diameter of the node pscale
float rad = ch("rad");

//convert to local variable
int myassociates[] = i[]@associates;

//if there are associates in this array only then execute.
if (len(myassociates) > 0){
    vector sum = {0,0,0};

    //run all associates in array
    foreach(int node; myassociates){
        //get position of associate point
        vector nodepos = point(0, "P", node);
        //get direction normalized differnce between associate
        //node and birth point.
        vector dir = normalize(nodepos - @P);
        //add direction to the sum
        sum += dir;

  vector sum = {0,0,0};

  //run all associates in array
  foreach(int node; myassociates){
      //get position of associate point
      vector nodepos = point(0, "P", node);
      //get direction normalized differnce between associate
      //node and birth point.
      vector dir = normalize(nodepos - @P);
      //add direction to the sum
      sum += dir;


      }
      //normalize vector
      sum = normalize(sum);

      //create new node
      int nnode = addpoint(0, @P+sum*rad);
      setpointgroup(0, "nodes", nnode, 1, "set");
      //add color
      setpointattrib(0, "Cd", nnode, {1,0,0}, "set");

      //create line between points
      int nprim = addprim(0, "polyline");
      addvertex(0, nprim, @ptnum);
      addvertex(0, nprim, nnode);


  }
```

Attributes to Create: *
Enforce Prototypes
Rad: 0.002

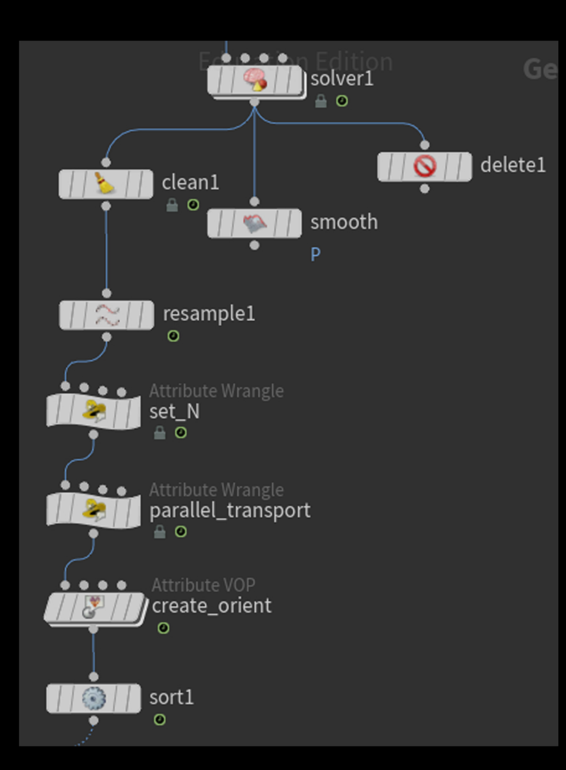



```
  vector sum = {0,0,0};

  //run all associates in array
  foreach(int node; myassociates){
      //get position of associate point
      vector nodepos = point(0, "P", node);
      //get direction normalized differnce between associate
      //node and birth point.
      vector dir = normalize(nodepos - @P);
      //add direction to the sum
      sum += dir;


      }
      //normalize vector
      sum = normalize(sum);

      //create new node
      int nnode = addpoint(0, @P+sum*rad);
      setpointgroup(0, "nodes", nnode, 1, "set");
      //add color
      setpointattrib(0, "Cd", nnode, {1,0,0}, "set");

      //create line between points
      int nprim = addprim(0, "polyline");
      addvertex(0, nprim, @ptnum);
      addvertex(0, nprim, nnode);


  }
```

Attributes to Create: *
Enforce Prototypes
Rad: 0.002

GENERATING CORAL MESH

Below is a basic demonstration of the meshing technique for the branching coral structure. Vex was used here for orinting the curves as it grows so that a clean normalized mesh can be easily generated.

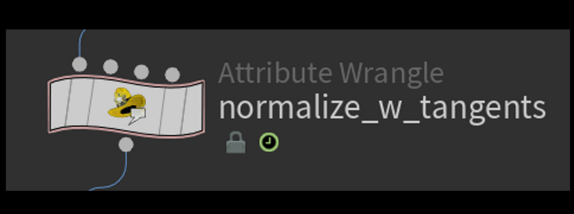


Here is where the process of skinning our curves begins. Due to all the twists and turns in the geo this code is manipulated in such a way that it can handle very complex, curvy geometry. This was also used to generate a mesh for the brain coral.

This method begins with creating and seting a general direction for the normals of the curve.

VEXpression

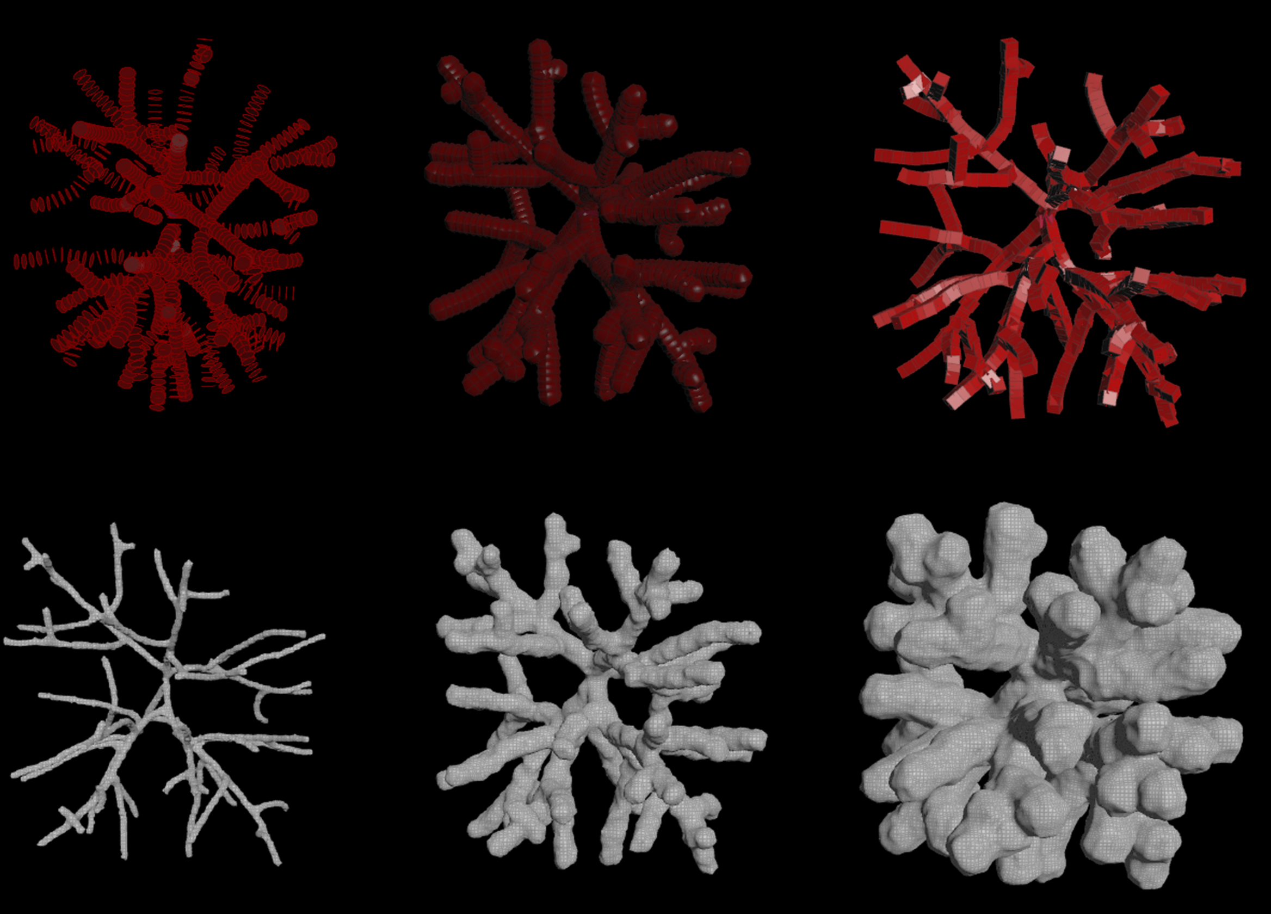
```
@N = {0,1,0};
```



VEXpression

```
int pnts[] = primpoints(0, @primnum);
int pntcnt = len(pnts);

vector firsttangent = normalize(point(0, "P", pnts[1]) - point(0, "P", pnts[0]));
vector firstnormal = {0,1,0};

vector helper = normalize(cross(firstnormal, firsttangent));
firstnormal = normalize(cross(firsttangent, helper));

//forward declarations
vector bitangent = {0,0,0};
float theta = 0;

vector tangents[] = {};
vector normals[] = {};

// fill arrays
for(int i = 0; i<pntcnt-1; i++){
    push(tangents, normalize(point(0, "P", pnts[i+1]) - point(0, "P", pnts[i])));
    push(normals, firstnormal);
    }
//set values for last point
push(tangents, tangents[pntcnt-2]);
push(normals, normals[pntcnt-2]);

//parallel transport
for (int k = 0; k<pntcnt-1; k++){
    bitangent = cross(tangents[k], tangents[k+1]);

     if (length(bitangent) == 0){
         normals[k+1] = normals[k];
         }
     else{
         bitangent = normalize(bitangent);
         theta = acos(dot(tangents[k], tangents[k+1]));
         matrix rotmat = ident();
         rotate( rotmat, theta, bitangent);
         normals[k+1] = rotmat * normals[k];
         }
     }

 //set attributes
 for (int j=0; j<pntcnt; j++){
     setpointattrib(0, "PT_tangent", pnts[j], tangents[j], "set");
     setpointattrib(0, "PT_normal", pnts[j], normals[j], "set");
     bitangent = normalize(cross( normals[j], tangents[j]));
     setpointattrib(0, "PT_bitangent", pnts[j], bitangent, "set");
     }
```

CORAL MESH RESULT



FINAL RESULT

CONCLUSION

I found this project to be very challenging especially when it came to final look development. If there is one lesson I am taking away from this project it is nature has no poly limits. The branching coral especially was hard to get the final details in using a plausible, fast method. However I did learn plenty on how to take an academic or siggraph paper and convert it into houdini using vex. Truly would like to continue implementing vex into more of my future projects.