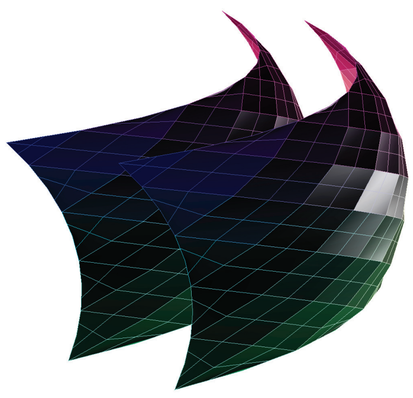
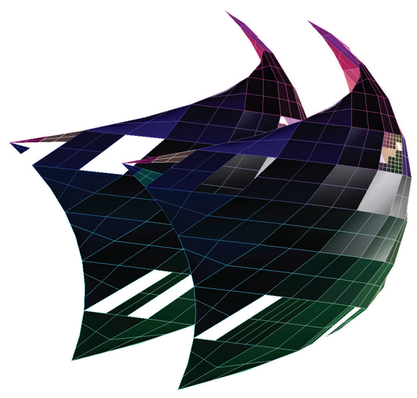
// VS CODING FORM

// MATTHEW TAM

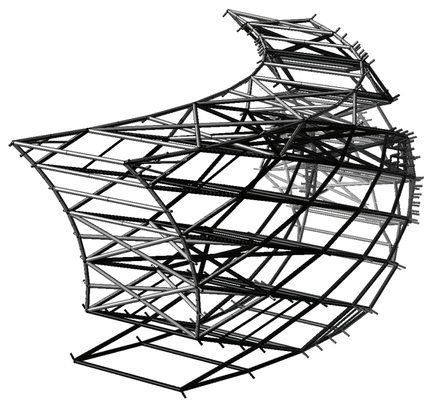
```
int unum = 16, vnum = 16;
double uinc = 1.0/unum, vinc = 1.0/vnum;

for (int i=0; i <= unum; i++) {
  for (int j=0; j < vnum; j++) {
    double radius = 0.1;
    double small_radius = radius/2;
    double val = map.get( i*uinc, j*vinc );
    double val2 = (map2.get( i*uinc, j*vinc ))*3 + 1 ;
    double depthA = 1;
    double depthB = -0.5;
    double offset = .3;
    double frame_depthA = -1;

    if ( (i+j)%2 == 0 ) {
     if (i > 0) {
      ptA1 = surfA.pt( (i-1)*uinc, j*vinc, depthA);
      ptA1b = surfA.pt( (i-1+offset)*uinc, j*vinc, depthA*2 );
      ptA1c = surfA.pt( (i-1+offset*2)*uinc, j*vinc, depthA*4*val2 );
      ptA1d = surfA.pt( (i-1+offset*2.5)*uinc, j*vinc, depthA*3 );
      frame_ptA1 = surfA.pt( (i-1)*uinc, j*vinc );
      ptA1mid = surfA.pt( (i-1+offset*2.5)*uinc, j*vinc );

      frame_ptB1 = surfB.pt( (i-1)*uinc, j*vinc );
      ptB1 = surfB.pt( (i-1)*uinc, j*vinc, depthB);
      ptB1b = surfB.pt( (i-1+offset)*uinc, j*vinc, depthB*4 );
      ptB1mid = surfB.pt( (i-1+offset*2.5)*uinc, j*vinc );
     }
     else {

     if (i < unum) {
      ptA3 = surfA.pt( (i+1)*uinc, j*vinc, depthA );
      ptA3b = surfA.pt( (i+1-offset)*uinc, j*vinc, depthA*2 );
      ptA3c = surfA.pt( (i+1-offset*2)*uinc, j*vinc, depthA*6*val2 );
      ptA3d = surfA.pt( (i+1-offset*2.5)*uinc, j*vinc, depthA*3 );
      frame_ptA3 = surfA.pt( (i+1)*uinc, j*vinc );
      ptA3mid = surfA.pt( (i+1-offset*2.5)*uinc, j*vinc );

      frame_ptB3 = surfB.pt( (i+1)*uinc, j*vinc );
      ptB3 = surfB.pt( (i+1)*uinc, j*vinc, depthB );
      ptB3b = surfB.pt( (i+1-offset)*uinc, j*vinc, depthB*4 );
      ptB3mid = surfB.pt( (i+1-offset*2.5)*uinc, j*vinc );
     }

     if (j > 0) {
      ptA2 = surfA.pt( i*uinc, (j-1)*vinc, depthA );
      ptA2b = surfA.pt( i*uinc, (j-1+offset)*vinc, depthA*2 );
      ptA2c = surfA.pt( i*uinc, (j-1+offset*2)*vinc, depthA*4*val2 );
      ptA2d = surfA.pt( i*uinc, (j-1+offset*2.5)*vinc, depthA*3 );
      frame_ptA2 = surfA.pt( i*uinc, (j-1)*vinc );
      ptA2mid = surfA.pt( i*uinc, (j-1+offset*2.5)*vinc );

      frame_ptB2 = surfB.pt( i*uinc, (j-1)*vinc );
      ptB2 = surfB.pt( i*uinc, (j-1)*vinc, depthB );
      ptB2b = surfB.pt( i*uinc, (j-1+offset)*vinc, depthB*4 );
      ptB2mid = surfB.pt( i*uinc, (j-1+offset*2.5)*vinc );
     }

     if (j< vnum-1) {
      ptA4 = surfA.pt( i*uinc, (j+1)*vinc, depthA );
      ptA4b = surfA.pt( i*uinc, (j+1-offset)*vinc, depthA*2 );
      ptA4c = surfA.pt( i*uinc, (j+1-offset*2)*vinc, depthA*4*val2 );
      ptA4d = surfA.pt( i*uinc, (j+1-offset*2.5)*vinc, depthA*3 );
      frame_ptA4 = surfA.pt( i*uinc, (j+1)*vinc );
      ptA4mid = surfA.pt( i*uinc, (j+1-offset*2.5)*vinc );

      frame_ptB4 = surfB.pt( i*uinc, (j+1)*vinc );
      ptB4 = surfB.pt( i*uinc, (j+1)*vinc, depthB );
      ptB4b = surfB.pt( i*uinc, (j+1-offset)*vinc, depthB*4 );
      ptB4mid = surfB.pt( i*uinc, (j+1-offset*2.5)*vinc );
     }

if (val < 0.5) {

     //SURFACE A POINTS
     IVec ptA1m = ptA1b.mid(ptA2b);
     IVec ptA2m = ptA2b.mid(ptA3b);
     IVec ptA3m = ptA3b.mid(ptA4b);
     IVec ptA4m = ptA4b.mid(ptA1b);

     IVec ptA1mc = ptA1c.mid(ptA2c);
     IVec ptA2mc = ptA2c.mid(ptA3c);
     IVec ptA3mc = ptA3c.mid(ptA4c);
     IVec ptA4mc = ptA4c.mid(ptA1c);

     IVec ptA1md = ptA1d.mid(ptA2d);
     IVec ptA2md = ptA2d.mid(ptA3d);
     IVec ptA3md = ptA3d.mid(ptA4d);
     IVec ptA4md = ptA4d.mid(ptA1d);

     IVec[][] cptsA1 = new IVec[3][4];
     cptsA1[0][0] = ptA1;
     cptsA1[1][0] = ptA1.mid(ptA4);
     cptsA1[2][0] = ptA4;
     cptsA1[0][1] = ptA1m;
     cptsA1[1][1] = ptA1b;
     cptsA1[2][1] = ptA4m;
     cptsA1[0][2] = ptA1mc;
     cptsA1[1][2] = ptA1c;
     cptsA1[2][2] = ptA4mc;
     cptsA1[0][3] = ptA2md;
     cptsA1[1][3] = ptA2d;
     cptsA1[2][3] = ptA1md;

     IVec[][] cptsA2 = new IVec[3][4];
     cptsA2[0][0] = ptA2;
     cptsA2[1][0] = ptA2.mid(ptA1);
     cptsA2[2][0] = ptA1;
     cptsA2[0][1] = ptA2m;
     cptsA2[1][1] = ptA2b;
     cptsA2[2][1] = ptA1m;
     cptsA2[0][2] = ptA2mc;
     cptsA2[1][2] = ptA2c;
     cptsA2[2][2] = ptA1mc;
     cptsA2[0][3] = ptA3md;
     cptsA2[1][3] = ptA3d;
     cptsA2[2][3] = ptA2md;

     IVec[][] cptsA3 = new IVec[3][4];
     cptsA3[0][0] = ptA3;
     cptsA3[1][0] = ptA3.mid(ptA2);
     cptsA3[2][0] = ptA2;
     cptsA3[0][1] = ptA3m;
     cptsA3[1][1] = ptA3b;
     cptsA3[2][1] = ptA2m;
     cptsA3[0][2] = ptA3mc;
     cptsA3[1][2] = ptA3c;
     cptsA3[2][2] = ptA2mc;
     cptsA3[0][3] = ptA4md;
     cptsA3[1][3] = ptA4d;
     cptsA3[2][3] = ptA3md;

     IVec[][] cptsA4 = new IVec[3][4];
     cptsA4[0][0] = ptA4;
     cptsA4[1][0] = ptA4.mid(ptA3);
     cptsA4[2][0] = ptA3;
     cptsA4[0][1] = ptA4m;
     cptsA4[1][1] = ptA4b;
     cptsA4[2][1] = ptA3m;
     cptsA4[0][2] = ptA4mc;
     cptsA4[1][2] = ptA4c;
     cptsA4[2][2] = ptA3mc;
     cptsA4[0][3] = ptA1md;
     cptsA4[1][3] = ptA1d;
     cptsA4[2][3] = ptA4md;

     IVec[] jointcrvptsA = new IVec[] { ptA1d, ptA2d, ptA3d, ptA4d};
     IVec[] jointcrvptsB = new IVec[] {ptB1b, ptB2b, ptB3b, ptB4b };
     IVec[] jointcrvptsAmid = new IVec[] {ptA1mid, ptA2mid, ptA3mid, ptA4mid};
     IVec[] jointcrvptsBmid = new IVec[] { ptB1mid, ptB2mid, ptB3mid, ptB4mid};
     IVec[] jointcrvptsmid = new IVec[] {midpt1, midpt2, midpt3, midpt4 };

     if (IRandom.percent(60)) {

      //LOFTED PANELS
      new ISurface(cptsA1, 2, 2).clr(i*uinc, j*vinc, .5).layer(layer1);
      new ISurface(cptsA2, 2, 2).clr(i*uinc, j*vinc, .5).layer(layer1);
      new ISurface(cptsA3, 2, 2).clr(i*uinc, j*vinc, .5).layer(layer1);
      new ISurface(cptsA4, 2, 2).clr(i*uinc, j*vinc, .5).layer(layer1);
      new ISurface(cptsB1, 2, 1).clr(i*uinc, j*vinc, 1).layer(layer1);
      new ISurface(cptsB2, 2, 1).clr(i*uinc, j*vinc, 1).layer(layer1);
      new ISurface(cptsB3, 2, 1).clr(i*uinc, j*vinc, 1).layer(layer1);
      new ISurface(cptsB4, 2, 1).clr(i*uinc, j*vinc, 1).layer(layer1);

      //CONNECTING PIPES
      ICurve jointcrvA = new ICurve (jointcrvptsA, 2, true).clr(255, 0, 0).layer(layer1);
      ICurve jointcrvB = new ICurve (jointcrvptsB, 2, true).clr(255, 0, 0).layer(layer1);
      ICurve jointcrvAmid = new ICurve (jointcrvptsAmid, 2, true).clr(255, 0, 0).layer(layer1);
      ICurve jointcrvBmid = new ICurve (jointcrvptsBmid, 2, true).clr(255, 0, 0).layer(layer1);
      ICurve jointcrvmid = new ICurve (jointcrvptsmid, 2, true).clr(0, 0, 0).layer(layer1);
      ICurve[] joints = new ICurve[] { jointcrvA, jointcrvAmid, jointcrvBmid, jointcrvB  };

      IG.loft (joints, 2).clr(i*uinc, j*vinc, .75).layer(layer1);
```



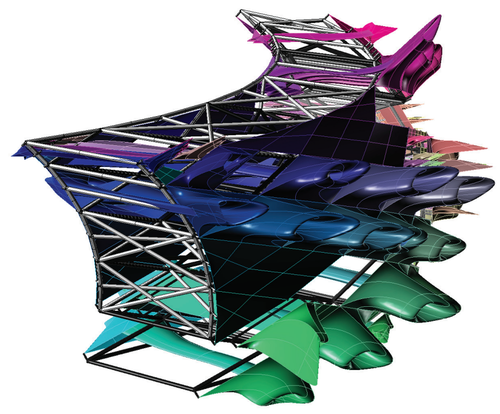
PANELLISED SURFACES



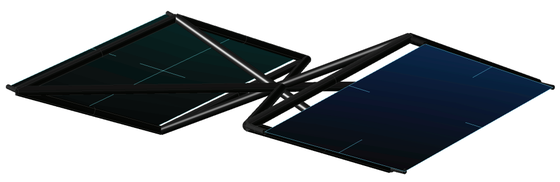
RANDOM CULL & DIVIDE



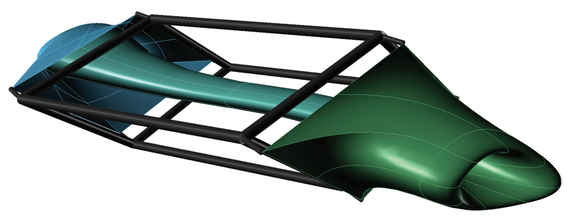
PIPE STRUCTURE



LOFTED SURFACES

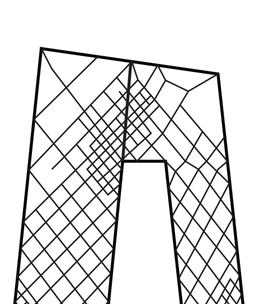


PANEL TYPE 1 - FLAT PANELS



PANEL TYPE 2 - SMOOTH LOFT

PANEL TYPE 3 - TWISTED LOFT



RANDOMISED DIAGRID
CCTV // OMA