```
public void VOXELIZE_FILL() {

  //   メッシュの数：int   faces
  // AABB * AABBの衝突判定をつかってみる


  FIELD_STEP = FIELD_SIZE*2/resolution;

  for(int k = 1; k<=resolution; k = k + 1){
    for(int j = 1; j<=resolution; j = j + 1){
       for(int i = 1; i<=resolution; i = i + 1){

          // x
          float x = -FIELD_SIZE + FIELD_STEP*(i - 1);
          float xx = -FIELD_SIZE + FIELD_STEP*(i);
          float realx  =  (x+xx)/2;
          // y
          float y = -FIELD_SIZE + FIELD_STEP*(j - 1);
          float yy = -FIELD_SIZE + FIELD_STEP*(j);
          float realy = (y+yy)/2;

          //z
          float z = -FIELD_SIZE + FIELD_STEP*(k - 1);
          float zz = -FIELD_SIZE + FIELD_STEP*(k);
          float realz = (z+zz)/2;



          for(int q = 0; q< poly.f.length; q= q + 1){

            PVector pv1  =new PVector (poly.f[q].v[3],poly.f[q].v[4],poly.f[q].v[5] );
            PVector pv2  =new PVector (poly.f[q].v[6],poly.f[q].v[7],poly.f[q].v[8]);
            PVector pv3  =new PVector (poly.f[q].v[9],poly.f[q].v[10],poly.f[q].v[11]);
            PVector bmin  =new PVector (x,y,z);
            PVector bmax  =new PVector        (xx,yy,zz);

            boolean preresult  = preAABB(x,y,z,xx,yy,zz,min(poly.f[q].v[3],poly.f[q].v[6],poly.f[q].v[9]),min(poly.f[q].v[4],pol
            max(poly.f[q].v[3],poly.f[q].v[6],poly.f[q].v[9]),max(poly.f[q].v[4],poly.f[q].v[7],poly.f[q].v[10]),max(poly.f[q].v
            boolean result;
            if (preresult  ==true) {  result= SPETestTriangleAABB(pv1 ,pv2,  pv3,  bmin,  bmax); }else { result =false; }
            boolean result2 = AABB(x,y,z,xx,yy,zz,min(poly.f[q].v[3],poly.f[q].v[6],poly.f[q].v[9]),min(poly.f[q].v[4],poly.f[q
            max(poly.f[q].v[3],poly.f[q].v[6],poly.f[q].v[9]),max(poly.f[q].v[4],poly.f[q].v[7],poly.f[q].v[10]),max(poly.f[q].v

         if (result  ==true || result2 ==true    )     {
               xyzarrays[i-1][j-1][k-1] = 1;break;
          }else  {
               xyzarrays[i-1][j-1][k-1] = 0;
         }

          }
       }
    }
  }

  int  fillcount=0;
  int  heikinx=0;
  int heikiny=0;
  int heikinz=0;

    for(int k = 1; k<=resolution; k = k + 1){
    for(int j = 1; j<=resolution; j = j + 1){
       for(int i = 1; i<=resolution; i = i + 1){
           if ( xyzarrays[i-1][j-1][k-1] == 1) { fillcount ++;
           heikinz = heikinz + k;
           heikiny = heikiny + j;
           heikinx = heikinx + i;
           }
       }
    }
    }
    filling3d(heikinx/fillcount, heikiny/fillcount, heikinz/fillcount);
    mx=heikinx/fillcount;
    my=heikiny/fillcount;
    mz=heikinz/fillcount;
}

public void VOXELIZE() {
```

```
//   メッシュの数：int   faces
// AABB * AABBの衝突判定をつかってみる


FIELD_STEP = FIELD_SIZE*2/resolution;

for(int k = 1; k<=resolution; k = k + 1){
  for(int j = 1; j<=resolution; j = j + 1){
     for(int i = 1; i<=resolution; i = i + 1){

        // x
        float x = -FIELD_SIZE + FIELD_STEP*(i - 1);
        float xx = -FIELD_SIZE + FIELD_STEP*(i);
        float realx  =  (x+xx)/2;
        // y
        float y = -FIELD_SIZE + FIELD_STEP*(j - 1);
        float yy = -FIELD_SIZE + FIELD_STEP*(j);
        float realy = (y+yy)/2;

        //z
        float z = -FIELD_SIZE + FIELD_STEP*(k - 1);
        float zz = -FIELD_SIZE + FIELD_STEP*(k);
        float realz = (z+zz)/2;




        for(int q = 0; q< poly.f.length; q= q + 1){

          PVector  pv1  =new PVector (poly.f[q].v[3],poly.f[q].v[4],poly.f[q].v[5] );
          PVector  pv2  =new PVector (poly.f[q].v[6],poly.f[q].v[7],poly.f[q].v[8]);
          PVector  pv3  =new PVector (poly.f[q].v[9],poly.f[q].v[10],poly.f[q].v[11]);
          PVector  bmin  =new PVector (x,y,z);
          PVector  bmax  =new PVector        (xx,yy,zz);

          boolean preresult  =  preAABB(x,y,z,xx,yy,zz,min(poly.f[q].v[3],poly.f[q].v[6],poly.f[q].v[9]),min(poly.f[q].v[4],poly
          max(poly.f[q].v[3],poly.f[q].v[6],poly.f[q].v[9]),max(poly.f[q].v[4],poly.f[q].v[7],poly.f[q].v[10]),max(poly.f[q].v
          boolean result;
          if  (preresult  ==true) {  result= SPETestTriangleAABB(pv1 ,pv2,  pv3,  bmin,  bmax); }else { result =false; }
```

```
            boolean result2 = AABB(x,y,z,xx,yy,zz,min(poly.f[q].v[3],poly.f[q].v[6],poly.f[q].v[9]),min(poly.f[q].v[4],poly.f[q
            max(poly.f[q].v[3],poly.f[q].v[6],poly.f[q].v[9]),max(poly.f[q].v[4],poly.f[q].v[7],poly.f[q].v[10]),max(poly.f[q].v

         if (result  ==true || result2  ==true    )     {
               xyzarrays[i-1][j-1][k-1] = 1;break;
          }else  {
               xyzarrays[i-1][j-1][k-1] = 0;
         }

          }
       }
    }
  }
}

// AABBどうちの接触を調べる
public boolean preAABB(float aminx,float aminy,float  aminz,float amaxx,float  amaxy,float  amaxz,float  bminx,float  bminy,floa
  // 衝突
  if(              aminx < bmaxx && amaxx > bminx
                 && aminy < bmaxy && amaxy > bminy
                 && aminz < bmaxz && amaxz > bminz)
         {
                //println("1a");
                 return(true);
         }
  if(              bminx < amaxx && bmaxx > aminx
                 && bminy < amaxy && bmaxy > aminy
                 && bminz < amaxz && bmaxz > aminz)
         {
               // println("1b");
                 return(true);
         }
 return(false);
}


public boolean AABB(float aminx,float aminy,float  aminz,float amaxx,float  amaxy,float  amaxz,float  bminx,float  bminy,float
  // 包含
  if(                aminx < bminx && bmaxx < amaxx
                   && aminy < bminy && bmaxy < amaxy
                   && aminz < bminz && bmaxz < amaxz)
         {
                   //println("2");
                return(true);
         }

         return(false);
}


void initPhysics() {
   // box = new WETriangleMesh();
    // create a simple start mesh
    //box.addMesh(new Cone(new Vec3D(0, 0, 0), new Vec3D(0, 1, 0), 10, 50, 100).toMesh(4));
    //box.addMesh(new AABB(new Vec3D(), 50).toMesh());
    // then subdivide a few times...
    //bpx.subdivide();
    //box.subdivide();
    //box.subdivide();
    //box.subdivide();
    if (mesh  !=null) {
     physics =new VerletPhysics();
    physics.setWorldBounds(new AABB(new Vec3D(), 180));
    // turn mesh vertices into physics particles
    for (Vertex v : mesh.vertices.values()) {
        physics.addParticle(new VerletParticle(v));
    }
    // turn mesh edges into springs
    for (WingedEdge e : mesh.edges.values()) {
        VerletParticle a = physics.particles.get(((WEVertex) e.a).id);
        VerletParticle b = physics.particles.get(((WEVertex) e.b).id);
        physics.addSpring(new VerletSpring(a, b, a.distanceTo(b), 0.005f));
    }
    }
}
```