```
//int resolution = 16; // 初期ボクセル数
int resolution = 32;// 初期ボクセル数

// Digital Material Reseachn
// Vox Effects
// FabLab Japan
// Hiroya Tanaka

import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.io.File;
import java.io.FileWriter;
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.StringTokenizer;
import java.awt.event.*;
import java.awt.dnd.*;
import  java.awt.datatransfer.*;

// Import Mesh Library
import toxi.geom.*;
import toxi.geom.mesh.*;
import toxi.geom.mesh.subdiv.*;
import toxi.processing.*;
import toxi.util.*;
import toxi.volume.*;
import toxi.physics.*;
import toxi.physics.behaviors.*;
import toxi.physics.constraints.*;

VerletPhysics physics;
AttractionBehavior inflate;


ToxiclibsSupport gfx;
WETriangleMesh mesh;

XMLElement xml;
```

```
// Unlekker
import unlekker.data.*;
import unlekker.geom.*;
import unlekker.modelbuilder.*;
import unlekker.util.*;
import  ec.util.*;
//
import processing.opengl.*;
import javax.media.opengl.*;
import controlP5.*;
import javax.swing.*;


//Toggle
 boolean mesh_display =true;
 boolean voxel_display =true;
 boolean wire_display =true;
 boolean marching_display =true;

// File IO
 String getFile="";
 String setFile="";
 String ssetFile;
 String [] txtarray =null;
 String txt =null;
 int[] surface =new int[100000];
//  Array
 int[] arrays =new int[1000000];
//  Field
 int FIELD_SIZE =  120;
 int FIELD_STEP =  40;
/* world */
//int worldLength = 20000;
//int gridLength = 500;
```

```
// Material Property
float elastic01 = 1000;
float elastic02 = 0.01;




//  Pointer
 int mx = 1;
 int my = 1;
 int mz = 1;
 int ml = 1;
//  Camera
 float expo = 1;
 float rotx, roty;
 float rate = 0.01;

// Resolution of Box
int  faces;// メッシュの数
int poruswidth=1;
int porusheight=100;
int porusnumber=100;
int mod = 0;
int tub = 0;
int maxresolution = 48;// 最大ボクセル数
int effect_percentage = 0;

 int z_scan = 0;

// ESO Cells
Cell[] p =new Cell[maxresolution*maxresolution*maxresolution];
Cell[][][] p2 =new Cell[maxresolution][maxresolution][maxresolution];
Cell[] currentCell =new Cell[1];

// ESO working with calculix
String filename;
String outputdir ="C:\\PROGRA~2\\bConverged\\common\\site\\";  // the directory where the output will be saved.
String workdir ="C:\\PROGRA~2\\bConverged\\common\\site\\"; // the directory where "cmdStartup" is installed
String batchfiledir = workdir +"cmdStartup.bat"; // cmd Startup is there
```

```
String[]  st  =new String[maxresolution*maxresolution*maxresolution];
String[]  sadat  =new String[maxresolution*maxresolution*maxresolution];
String[]  mises  =new String[maxresolution*maxresolution*maxresolution];
double[]  mp  =new double[maxresolution*maxresolution*maxresolution];
int[]  GROWTHnum  =new int[maxresolution*maxresolution*maxresolution];

boolean  exportFlg=false;
int  evolution=0;

/* Publics defined by Sliders */
public int THRESHOLD_MIN = 0;
public int THRESHOLD_MAX = 255;
public double BURDEN = 1;
public int numberOfSteps = 0;

// Bar Alpha
int mesh_alpha= 100;
int voxel_alpha= 100;
int wire_alpha= 100;
int marching_alpha = 100;

// ControlP5
 ControlP5 controlP5;
 DropdownList d1, d2, d3, d4, d5;
    Textfield     tfUserName;// テキストフィールド
String  txts  ="";

// Text
 PFont    fonta,fontb;
 long     time;

 STL stl;
 FaceList poly;




//  Array
int[][]  xyarrays  =new int[maxresolution][maxresolution];
```

```
int[][] yzarrays =new int[maxresolution][maxresolution];
int[][] zxarrays =new int[maxresolution][maxresolution];
int[][][] xyzarrays =new int[maxresolution][maxresolution][maxresolution];


//Filling Out Queue
 int pixelQueuex[] =new int[maxresolution*maxresolution*maxresolution];
 int pixelQueuey[] =new int[maxresolution*maxresolution*maxresolution];
 int pixelQueuez[] =new int[maxresolution*maxresolution*maxresolution];
 int pixelQueueSize = 0;

 int sprout = 0;
 int needlize = 0;
 int mouseok = 0;
 int inflation = 0;

 boolean noiseon =false;
 boolean transparent =false;
 boolean removeon =false;
 boolean tubeon =false;
 boolean poruson =false;
 boolean dualon =false;
 boolean sphon =false;
 boolean esoon =false;

 String format;

//MarchingCubes mc;
Vec3D rotationAxis;
Boolean bUseFill;

void setup() {


 addMouseWheelListener(new MouseWheelListener() {
    public void mouseWheelMoved(MouseWheelEvent mwe) {
      mouseWheel(mwe.getWheelRotation());
  }});
```

```
//   Marching初期化

 FIELD_STEP = FIELD_SIZE*2/resolution;
Vec3D aabbMin =new Vec3D(  FIELD_STEP/2-resolution/2+(-FIELD_SIZE*2    +  FIELD_STEP)/2,    FIELD_STEP/2-resolution/2+(-FIELD_S
Vec3D aabbMax =new Vec3D( FIELD_STEP/2+resolution/2+(-FIELD_SIZE*2  + FIELD_STEP*((resolution)*2 -1))/2,   FIELD_STEP/2+resolu

Vec3D numPoints =new Vec3D(resolution+1,resolution+1,resolution+1);

float isoLevel = 1;
MarchingCubes(aabbMin, aabbMax, numPoints, isoLevel);

 rotationAxis =new Vec3D();

 bUseFill =false;


    poly =null;

size(int(screen.width*0.8), int(screen.height*0.8), OPENGL);
gfx=new ToxiclibsSupport(this);
initPhysics();

 controlP5 =new ControlP5(this);

controlP5.addSlider("resolution",1,48,16,50,400,200,30);
controlP5.addSlider("z_scan",      0,100,0,50,450,200,30);
tfUserName = controlP5.addTextfield("bit_pattern", 50, 500, 200, 30);
tfUserName.setFocus(true);

controlP5.addSlider("effect_percentage",0,100,0,50,550,200,50);


d1 = controlP5.addDropdownList("FILE").setPosition(300, 50).setSize(120,250);
d1.setBackgroundColor(color(190));
d1.setItemHeight(30);
d1.setBarHeight(30);
```

```
d1.captionLabel().set("FILE");
d1.captionLabel().style().marginTop = 5;
d1.captionLabel().style().marginLeft = 5;
d1.valueLabel().style().marginTop = 15;
d1.addItem("NEW",1);
d1.addItem("load_STL",2);
d1.addItem("load_VOXEL_txt",3);
d1.addItem("load_VXC",4);
d1.addItem("save_STL",5);
d1.addItem("save_VOXELtxt",6);
d1.addItem("save_VXC",7);
//d1.scroll(0);
d1.setColorBackground(color(60));
d1.setColorActive(color(255,         128));


d2 = controlP5.addDropdownList("EFFECT_ON_VOXEL").setPosition(450, 50).setSize(120,300);
d2.setBackgroundColor(color(190));
d2.setItemHeight(30);
d2.setBarHeight(30);
d2.captionLabel().set("EFFECT_ON_VOXEL");
d2.captionLabel().style().marginTop = 5;
d2.captionLabel().style().marginLeft = 5;
d2.valueLabel().style().marginTop = 15;
d2.addItem("TRANSPARENT",1);
d2.addItem("MARCHING_CUBE",2);
d2.addItem("TUBE",3);
d2.addItem("PORUS",4);
d2.addItem("NOISE",5);
d2.addItem("DUAL_POLY",6);
d2.addItem("SPHERE",7);
d2.addItem("ESO",8);
d2.addItem("SPRING",9);
//d1.scroll(0);
d2.setColorBackground(color(60));
d2.setColorActive(color(255,   128));

d3 = controlP5.addDropdownList("EFFECT_ON_SURFACE").setPosition(600, 50).setSize(120,200);
d3.setBackgroundColor(color(190));
```

```
d3.setItemHeight(30);
d3.setBarHeight(30);
d3.captionLabel().set("EFFECT_ON_SURFACE");
d3.captionLabel().style().marginTop = 5;
d3.captionLabel().style().marginLeft = 5;
d3.valueLabel().style().marginTop = 15;
d3.addItem("VOXELIZE",1);
d3.addItem("VOXELIZE_FILL",2);
d3.addItem("NEEDRIZE",3);
d3.addItem("SPORULATE",4);
d3.addItem("SPRING",5);
d3.addItem("INFLATION",6);
//d1.scroll(0);
d3.setColorBackground(color(60));
d3.setColorActive(color(255,   128));

d4 = controlP5.addDropdownList("DIGITAL MATERIAL").setPosition(750, 50).setSize(120,200);
d4.setBackgroundColor(color(190));
d4.setItemHeight(30);
d4.setBarHeight(30);
d4.captionLabel().set("DIGITAL MATERIAL");
d4.captionLabel().style().marginTop = 5;
d4.captionLabel().style().marginLeft = 5;
d4.valueLabel().style().marginTop = 15;

d4.setColorBackground(color(60));
d4.setColorActive(color(255, 128));

d5 = controlP5.addDropdownList("STRUCTURAL OPTIMIZATION").setPosition(900, 50).setSize(120,200);
d5.setBackgroundColor(color(190));
d5.setItemHeight(30);
d5.setBarHeight(30);
d5.captionLabel().set("STRUCTURAL OPTIMIZATION");
d5.captionLabel().style().marginTop = 5;
d5.captionLabel().style().marginLeft = 5;
d5.valueLabel().style().marginTop = 15;

d5.setColorBackground(color(60));
d5.setColorActive(color(255, 128));
```

```
  controlP5.addToggle("wire_display",true, 50, 20,30,30);
  controlP5.addToggle("mesh_display",true, 50, 90,30,30);
  controlP5.addToggle("voxel_display",true,50, 160,30,30);
  controlP5.addToggle("marching_display",true, 50,230,30,30);

  controlP5.addSlider("wire_alpha",0,100,100, 20,100,30);
  controlP5.addSlider("mesh_alpha",0,100,100, 90,100,30);
  controlP5.addSlider("voxel_alpha",0,100,100,160,100,30);
  controlP5.addSlider("marching_alpha",0,100,100, 230,100,30);


  // Right UIs
  controlP5.addBang("WRITE_INP", int(screen.width*0.8)-320, 200, 30, 30);
  controlP5.addBang("DO_CCX", int(screen.width*0.8)-320, 260, 30, 30);
  controlP5.addBang("DO_CGX", int(screen.width*0.8)-315, 325, 20, 20);
  controlP5.addBang("LOAD_CCX", int(screen.width*0.8)-320, 380, 30, 30);
  controlP5.addBang("REMOVE_MIN", int(screen.width*0.8)-320, 440, 30, 30);
  controlP5.addBang("REMOVE_MAX", int(screen.width*0.8)-320, 500, 30, 30);

  //addSlider(theName, theMin, theMax, theDefaultValue, theX, theY, theW, theH);
  controlP5.addSlider("THRESHOLD_MAX", 0, 255, 255,int(screen.width*0.8)-320, 10*11, 150, 10);
  controlP5.addSlider("THRESHOLD_MIN", 0, 255, 0,int(screen.width*0.8)-320, 10*9, 150, 10);
  controlP5.addSlider("BURDEN", 0, 1000, 1,int(screen.width*0.8)-320, 10*13, 150, 10);
  controlP5.addSlider("ELASTIC01", 0, 10000, 1000,int(screen.width*0.8)-320, 10*15, 150, 10);
  controlP5.addSlider("ELASTIC02", 0, 1, 0.01,int(screen.width*0.8)-320, 10*17, 150, 10);


  controlP5.addBang("RUN_THROUGH", int(screen.width*0.8)-380, 200, 30, 210);
  controlP5.addBang("AUTO_EVOLVE", int(screen.width*0.8)-260, 200, 30, 330);
  controlP5.addBang("STOP", int(screen.width*0.8)-260, 560, 30, 30);
//  controlP5.addBang("WRITE_DATA", int(screen.width*0.8)-260, 600, 30, 30);


  controlP5.addButton("setWall_X_PLUS",  1,int(screen.width*0.8)-200, 200, 80, 50);
  controlP5.addButton("setWall_X_MINUS",  1,int(screen.width*0.8)-200, 260, 80, 50);
  controlP5.addButton("setWall_Y_PLUS",  1,int(screen.width*0.8)-200, 320, 80, 50);
  controlP5.addButton("setWall_Y_MINUS",  1,int(screen.width*0.8)-200, 380, 80, 50);
  controlP5.addButton("setWall_Z_PLUS",  1,int(screen.width*0.8)-200, 440, 80, 50);
```

```
  controlP5.addButton("setWall_Z_MINUS",  1,int(screen.width*0.8)-200, 500, 80, 50);

  controlP5.addButton("setBurden_X_PLUS",  1,int(screen.width*0.8)-100, 200, 90, 50);
  controlP5.addButton("setBurden_X_MINUS",  1,int(screen.width*0.8)-100, 260, 90, 50);
  controlP5.addButton("setBurden_Y_PLUS",  1,int(screen.width*0.8)-100, 320, 90, 50);
  controlP5.addButton("setBurden_Y_MINUS",  1,int(screen.width*0.8)-100, 380, 90, 50);
  controlP5.addButton("setBurden_Z_PLUS",  1,int(screen.width*0.8)-100, 440, 90, 50);
  controlP5.addButton("setBurden_Z_MINUS",  1,int(screen.width*0.8)-100, 500, 90, 50);

  controlP5.addButton("rotateModel_x",  1,int(screen.width*0.8)-200, 600, 80, 50);
  controlP5.addButton("rotateModel_y",  1,int(screen.width*0.8)-200, 660, 80, 50);
  controlP5.addButton("rotateModel_z",  1,int(screen.width*0.8)-200, 720, 80, 50);
  /*
  controlP5.addButton("loadfile",1,10,45,80,19);
  controlP5.addButton("savefile",1,100,45,80,19);
  controlP5.addButton("voxelize",1,190,45,80,19);
  controlP5.addSlider("resolution",1,48,10,390,200,20);
  controlP5.addSlider("scan",1,100,10,420,200,20);



  controlP5.addSlider("poruswidth",0,10,1,10,70,200,20);
  controlP5.addSlider("porusheight",0,1000,100,10,95,200,20);
  controlP5.addSlider("porusnumber",0,100,100,10,120,200,20);
  controlP5.addButton("tubeonoff",1,280,90,80,19);

  controlP5.addButton("make_porus",1,380,90,80,19);

  controlP5.addButton("saveVoxel", 1, 10, 165, 80, 19);
  controlP5.addButton("loadVoxel", 1, 10, 185, 80, 19);

  controlP5.addButton("saveVXC", 1, 10, 245, 80, 19);
  controlP5.addButton("loadVXC", 1, 10, 265, 80, 19);
  */
  initialize();
}


void draw() {
```

```
 background(0);
  PGraphicsOpenGL  pgl  =  (PGraphicsOpenGL)g;


 // Lighting
   ambientLight(150, 150, 150);
   directionalLight(255,255,255,-1,0,0);
   pointLight(160, 160, 160, 0, 0, 200);
   spotLight(100, 100, 100, 0, 0, 200, 0, 0, -1,PI, 2);







 // Camera
 if (mousePressed && getFile =="" && setFile =="" &&  mouseok ==1) {
   rotx = rotx + (mouseY  -pmouseY) * rate;
   roty = roty + (mouseX  -pmouseX) * rate;
  }

 // FileIO
   if(getFile  !=""){
   //ファイルを取り込む

if(format=="stl"){   fileLoader(); }
if(format=="txt"){   loadVoxelarray(); }
if(format=="vxc"){   loadXMLVXC();}
  }

 //Text


 fill(255,255,255);
  fonta =createFont("Arial-BoldMT-15", 30);
 textFont(fonta);
 text("Voxel Effects",  300,int(screen.height*0.8)-50);
```

```
 fontb =createFont("ArialMT-15", 15);
textFont(fontb);
text("Hiroya Tanaka Laboratory,  Social Fabrication Center,  Keio SFC  2014",  500,int(screen.height*0.8)-50);



// 中心を合わせる

translate((screen.width*0.8)/2,(screen.height*0.8)/2,200);
rotateX(rotx); rotateY(roty); scale(expo);


  if (marching_display ==true){
      for(int k =int((resolution-1)*z_scan/100)+1; k<=resolution; k = k + 1){
  for(int j = 1; j<=resolution; j = j + 1){
     for(int i = 1; i<=resolution; i = i + 1){
     voxelValues[i-1][j-1][k-1]=xyzarrays[i-1][j-1][k-1]*100;
     }
     }
      }
  fill(0,255,0,marching_alpha*255/100);
  noStroke();
  renderMesh();
  }
 // renderGrid();


// STL書き出し
if(setFile  !="" && format =="stl"){
 beginRaw("unlekker.data.STL",setFile+".stl");
 println("start :"+setFile);
}

//Output Voxel(Binary) data
if (setFile  !="" && format =="txt") {
  writeVoxel();
   setFile ="";
   format ="";
}
```

```
  //Output Voxel(Cornell) Data
  if (setFile  !="" && format =="vxc") {
    writeVXC();
     setFile ="";
     format ="";
  }



   if(poly!=null) {
     for (int jj=0; jj<poly.f.length; jj++) { arrays[jj]=0; }
     // Random Fill out Generation
    int count = 0;
     while ( count < poly.f.length * effect_percentage/100 ){
     int  r=int(random(poly.f.length));
     if (arrays[r]==0) { arrays[r]=1; count++; }
}
    }

   if(poly!=null) {
    if (mesh_display  ==true){
      fill(255,255,255,mesh_alpha*255/100);


    if (needlize==0 && sprout ==0 && inflation == 0) {
       poly.draw(this);

     }else if (needlize == 1){
    // Needlize

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

        float centerx = poly.f[q].v[0] * effect_percentage/2 + ( poly.f[q].v[3] + poly.f[q].v[6] + poly.f[q].v[9])/3;
        float centery = poly.f[q].v[1] * effect_percentage/2 + ( poly.f[q].v[4] + poly.f[q].v[7] + poly.f[q].v[10])/3;
        float centerz = poly.f[q].v[2] * effect_percentage/2  + ( poly.f[q].v[5] + poly.f[q].v[8] + poly.f[q].v[11])/3;

            beginShape(TRIANGLES);  vertex(centerx,  centery,  centerz);vertex(poly.f[q].v[3],poly.f[q].v[4],poly.f[q].v[5]); ver
            beginShape(TRIANGLES);  vertex(centerx,  centery,  centerz);vertex(poly.f[q].v[3],poly.f[q].v[4],poly.f[q].v[5]); ver
```

```
        beginShape(TRIANGLES);  vertex(centerx,  centery,  centerz);vertex(poly.f[q].v[6],poly.f[q].v[7],poly.f[q].v[8]); ver
  }
     }
     else if (sprout == 1) {
       poly.draw(this);
           for(int q = 0; q< poly.f.length; q= q + 1){
    float centerx = poly.f[q].v[0]  + ( poly.f[q].v[3] + poly.f[q].v[6] + poly.f[q].v[9])/3;
    float centery = poly.f[q].v[1]  + ( poly.f[q].v[4] + poly.f[q].v[7] + poly.f[q].v[10])/3;
    float centerz = poly.f[q].v[2]  + ( poly.f[q].v[5] + poly.f[q].v[8] + poly.f[q].v[11])/3;
    pushMatrix();
    translate(centerx, centery, centerz);
  sphereDetail(10);
  sphere(effect_percentage/5);
  popMatrix();
     }
      }
     else if (inflation ==1)
     {
         physics.update();
for (Vertex v : mesh.vertices.values()) {
    v.set(physics.particles.get(v.id));
}

mesh.center(null);

for (Vertex v : mesh.vertices.values()) {
    physics.particles.get(v.id).set(v);
}

gfx.origin(new Vec3D(),0);
mesh.translate(new Vec3D(-FIELD_STEP/2,-FIELD_STEP/2,-FIELD_STEP/2));
gfx.scale(new   Vec3D(expo,expo,expo));
mesh.scale(new  Vec3D(expo*2.0,expo*2.0,expo*2.0));
fill(192);
noStroke();
 gfx.mesh(mesh,true, 0);


inflate=new AttractionBehavior(new Vec3D(), effect_percentage, -0.3f, 0.001f);
```

```
  physics.addBehavior(inflate);
  physics.removeBehavior(inflate);
}

  }


  if (tubeon  ==true){
    for(int q = 0; q< poly.f.length; q= q + 1){
      stroke(255);
           if (arrays[q]==1){
      line( (poly.f[q].v[3] + poly.f[q].v[6] + poly.f[q].v[9])/3- poly.f[q].v[0]*porusheight,
            (poly.f[q].v[4] + poly.f[q].v[7] + poly.f[q].v[10])/3- poly.f[q].v[1]*porusheight,
            (poly.f[q].v[5] + poly.f[q].v[8] + poly.f[q].v[11])/3- poly.f[q].v[2]*porusheight,
            (poly.f[q].v[3] + poly.f[q].v[6] + poly.f[q].v[9])/3+ poly.f[q].v[0]*porusheight,
            (poly.f[q].v[4] + poly.f[q].v[7] + poly.f[q].v[10])/3+ poly.f[q].v[1]*porusheight ,
            (poly.f[q].v[5] + poly.f[q].v[8] + poly.f[q].v[11])/3+ poly.f[q].v[2]*porusheight);
          }
        }
       }
     }



// ボクセルの描画
//bit_patternの 処 理
int len = txts.length();
int l = 0;

if ( esoon ==true)   {
  int   num   =   0;
  for(int k = 0; k<resolution; k = k + 1){
    for(int j = 0; j<resolution; j = j + 1){
    for(int i = 0; i<resolution; i = i + 1){
         if(p[num].state<2){   p[num].setstate(xyzarrays[k][j][i]);}
           num++;
    }
    }
       }
```

```
  for (int i = 0; i < p.length; i++) {
  p[i].display(voxel_alpha*255/100, THRESHOLD_MIN, THRESHOLD_MAX);
}
if (evolution>0) {println("evolution pass="+evolution); auto(); evolution++;}
}

if ( (voxel_display ==true || wire_display ==true)) {
noStroke();
noFill();


FIELD_STEP = FIELD_SIZE*2/resolution;

int blankcount = 0;

for(int k =int((resolution-1)*z_scan/100)+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;

       pushMatrix();
       translate(realx,realy,realz);

       // label
       textFont(fontb);  fill(200,200,200);
       if (i==1 && j==2 && k==2) {text("-X -Y -Z", 0. -60,-20); }
       if (i==resolution && j==2 && k==2) {text("+X",  20,-20,0); }
```

```
   //if (i==2 && j==1 && k==2) { text("-Y",  -50,0,-0); }
    if (i==2 && j==resolution && k==2) {text("+Y", -20, 30,0); }
   // if (i==2 && j==2 && k==1) { text("-Z", -50,-50,0); }
    if (i==2 && j==2 && k==resolution) {text("+Z", -20, 0,30); }

    //Mass
    if(  xyzarrays[i-1][j-1][k-1] == 1 )
    { blankcount++;
    if (voxel_display ==true && transparent ==false)
        {stroke(50,50,50,wire_alpha*255/100);fill(200,00,00,voxel_alpha*255/100);
   if (sphon ==true)  {sphereDetail(10); sphere(FIELD_STEP*effect_percentage/100);}
   else if (dualon ==false && sphon ==false)  {box(FIELD_STEP,FIELD_STEP,FIELD_STEP);}
   else if (dualon ==true && sphon ==false) { dual(FIELD_STEP,effect_percentage);}
    }
      }

    //(Not Used Anymore)
    if(  xyzarrays[i-1][j-1][k-1] == -1 && voxel_display ==true     )   {
      stroke(50,50,50,wire_alpha*255/100);fill(150,00,00,voxel_alpha*255/100);
     if (sphon ==true)  {sphereDetail(10); sphere(FIELD_STEP*effect_percentage/100);}
   else if (dualon ==false && sphon ==false)  {box(FIELD_STEP,FIELD_STEP,FIELD_STEP);}
   else if (dualon ==true && sphon ==false) {dual(FIELD_STEP,effect_percentage);}
 }

     //Just Frame
    if(  xyzarrays[i-1][j-1][k-1] == 0 )
    { blankcount++;

    if (wire_display ==true && k >=int((resolution-1)*z_scan/100)+1)     {
      stroke(50,50,50,wire_alpha*255/100); noFill();

   if (sphon ==true)  {sphereDetail(10); sphere(FIELD_STEP*effect_percentage/100);}
   else if (dualon ==false && sphon ==false)  {box(FIELD_STEP,FIELD_STEP,FIELD_STEP);}
   else if (dualon ==true && sphon ==false) { dual(FIELD_STEP,effect_percentage);}
}
    if (wire_display ==true &&  k <int((resolution-1)*z_scan/100)+1)    {
      stroke(0,0,50,wire_alpha*255/100); noFill();
     if (sphon ==true)  {sphereDetail(10); sphere(FIELD_STEP*effect_percentage/100);}
   else if (dualon ==false && sphon ==false)  {box(FIELD_STEP,FIELD_STEP,FIELD_STEP);}
```

```
      else if (dualon  ==true && sphon ==false)  {  dual(FIELD_STEP,effect_percentage);}
     }
       }

      //Cursor
       ml =int((resolution-1)*z_scan/100)+1;
      if (mx == i && my == j && mz == k) {
      fill(255,255,0,voxel_alpha*255/100);
      stroke(50,50,50,voxel_alpha*255/100);

      if (sphon  ==true)  {sphereDetail(10); sphere(FIELD_STEP*effect_percentage/100);}
      else if (dualon  ==false && sphon ==false)  {box(FIELD_STEP,FIELD_STEP,FIELD_STEP);}
      else if (dualon  ==true && sphon ==false) {dual(FIELD_STEP,effect_percentage);}
      }

       popMatrix();
     }
  }
}

// Noise

if (noiseon  ==true) {

 if  (len==0){
    int u=0;
    for(int k  =int((resolution-1)*z_scan/100)+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;

      pushMatrix();
      translate(realx,realy,realz);
      //println(blankcount + " ... " + effect_percentage);
      if ( xyzarrays[i-1][j-1][k-1] == 0 ){ u=u+effect_percentage;}

      if (u >= 100) {
      stroke(50,50,50,wire_alpha*255/100);fill(00,200,00,voxel_alpha*255/100); if (dualon  ==false)  {box(FIELD_STEP,FIELD_STI
      u=u-100;
       }

      popMatrix();
      }
    }
 }
}
 if (len >0) {

   for(int k  =int((resolution-1)*z_scan/100)+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;

      pushMatrix();
      translate(realx,realy,realz);
      //println(blankcount + " ... " + effect_percentage);
```

```
        if (xyzarrays[i-1][j-1][k-1] == 0 ) {
         if (txts.charAt(l)=='1'){ fill(0,200,0,voxel_alpha*255/100); if (dualon  ==false)  {box(FIELD_STEP,FIELD_STEP,FIELD_S

         l=l+1;
          if (l > len-1) { l = 0; }
        }
       popMatrix();
       }
     }
       }
    }

}
  //Transparent

   if (transparent  ==true ) {
    if  (len==0){
    int u=0;
    for(int k  =int((resolution-1)*z_scan/100)+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;

       pushMatrix();
       translate(realx,realy,realz);

       if ( xyzarrays[i-1][j-1][k-1] == 1 ){ u=u+effect_percentage;}
```

```
      if (u >= 100) {

      stroke(50,50,50,wire_alpha*255/100);fill(255,0,0,voxel_alpha*255/100); if (dualon  ==false)  {box(FIELD_STEP,FIELD_STEP
      u=u-100;
       }else      {
      noStroke();noFill(); if (dualon  ==false)  {box(FIELD_STEP,FIELD_STEP,FIELD_STEP);} else {dual(FIELD_STEP,effect_percent



      popMatrix();
      }
    }
}
  }
    if (len >0) {

  for(int k  =int((resolution-1)*z_scan/100)+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;

      pushMatrix();
      translate(realx,realy,realz);
      //println(blankcount + " ... " + effect_percentage);
       if (xyzarrays[i-1][j-1][k-1] == 1 ) {
        if (txts.charAt(l)=='1'){ stroke(50,50,50,voxel_alpha*255/100);fill(0,200,0,voxel_alpha*255/100); if (dualon  ==false
        else    {noStroke();noFill(); if (dualon  ==false)  {box(FIELD_STEP,FIELD_STEP,FIELD_STEP);} else {dual(FIELD_STEP,ef
        l=l+1;
```

```
            if (l > len-1) { l = 0; }
          }
         popMatrix();
         }
       }
         }
      }
}
 }



  // STL書き出し終了
 if(setFile  !=""   && format =="stl") {
 endRaw();
  setFile ="";
  println("end");
 }

 // α合成を有効にする
 GL gl = pgl.beginGL();
 gl.glEnable(GL.GL_BLEND);
  gl.glBlendFunc(GL.GL_SRC_ALPHA,  GL.GL_ONE_MINUS_SRC_ALPHA);
 gl.glDisable(GL.GL_DEPTH_TEST);
 pgl.endGL();


   // GUI コンポーネント描画のための設定
 camera();
 gl = pgl.beginGL();
   gl.glDisable(GL.GL_DEPTH_TEST);// 深度テストを無効化
 gl.glEnable(GL.GL_BLEND);
  gl.glBlendFunc(GL.GL_SRC_ALPHA,  GL.GL_ONE_MINUS_SRC_ALPHA);
 pgl.endGL();


 // 3次元フィル
 while (pixelQueueSize > 0){
          pixelQueueSize--;
```

```
            filling3d ( pixelQueuex[pixelQueueSize+1] ,pixelQueuey[pixelQueueSize+1], pixelQueuez[pixelQueueSize+1]);

}



}




public float distance(float  x1,float  y1,float  x2,float y2){

  return(sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)));
}


// 座標 p1,p2 を通る直線と座標 p3,p4 を結ぶ線分が交差しているかを調べる
public boolean intersection(float  x1,float  y1,float  x2,float  y2,float  x3,float  y3,float  x4,float y4){

    if (((x1 - x2) * (y3 - y1) + (y1 - y2) * (x1 - x3)) * ((x1 - x2) * (y4 - y1) + (y1 - y2) * (x1 - x4)) <= 0 &&
        ((x3 - x4) * (y1 - y3) + (y3 - y4) * (x3 - x1)) * ((x3 - x4) * (y2 - y3) + (y3 - y4) * (x3 - x2)) <= 0 ) {

            return(true); //交差する


     }

    return(false); //交差しない

    }

void mousePressed(){
  mouseok = 0;
  if (280  <mouseX && 100 <mouseY  &&mouseX  <int(screen.width*0.8)-380){
                                      mouseok = 1;
                                      }
```

```
}


 void mouseDragged() {
  // Camera

  if (getFile  =="" &&  setFile  =="" && mouseok==1) {
  rotx = rotx + (mouseY  -pmouseY) * rate;
  roty = roty + (mouseX  -pmouseX)  *  rate;

  }
}


void mouseWheel(float delta) {
   expo = expo + delta*0.1;

}




void keyReleased() {
  if (key== ' ') {
  physics.removeBehavior(inflate);
}
}
void keyPressed() {
 // Mouse Pointer Move
if (keyCode==DOWN && my<resolution) {  my++; }
if (keyCode==UP && my>1) { my--; }
if  (keyCode==RIGHT && mx<resolution) {  mx++; }
if  (keyCode==LEFT && mx>1) { mx--; }
if  (key=='a' && mz>1) { mz--;}
if  (key=='z' && mz<resolution) { mz++;}

if  (key=='0') { xyzarrays[mx-1][my-1][mz-1]=0;}
if  (key=='1') { xyzarrays[mx-1][my-1][mz-1]=1;}
if  (key=='2') { xyzarrays[mx-1][my-1][mz-1]=2;}
```

```
if  (key=='3') { xyzarrays[mx-1][my-1][mz-1]=3;}
if  (key=='4') { xyzarrays[mx-1][my-1][mz-1]=4;}
if  (key=='5') { xyzarrays[mx-1][my-1][mz-1]=5;}
if  (key=='6') { xyzarrays[mx-1][my-1][mz-1]=6;}
if  (key=='7') { xyzarrays[mx-1][my-1][mz-1]=7;}
if  (key=='8') { xyzarrays[mx-1][my-1][mz-1]=8;}
if  (key=='9') { xyzarrays[mx-1][my-1][mz-1]=9;}

if  (keyCode==ENTER) { }
  if (key==' ') {
  inflate=new AttractionBehavior(new Vec3D(), effect_percentage, -0.3f, 0.001f);
  physics.addBehavior(inflate);
  }
      if (key  =='r') {
        initPhysics();
    }
}


private void filling3d (int x,int  y,int  z)  {/* 塗りつぶし */

     xyzarrays[x][y][z]=1;/* ー1を置く */

    if (y>0){
    if  (xyzarrays[x][y-1][z]==0)  {

       pixelQueueSize++;
      pixelQueuex[pixelQueueSize] = x; pixelQueuey[pixelQueueSize] = y-1; pixelQueuez[pixelQueueSize] = z;

      }
    }

    if (x<resolution-1){
    if  (xyzarrays[x+1][y][z]==0)  {/* 右 */
              pixelQueueSize++;
      pixelQueuex[pixelQueueSize] = x+1 ; pixelQueuey[pixelQueueSize] = y ; pixelQueuez[pixelQueueSize] = z;
      }
    }
```

```
    if (y<resolution-1){
    if (xyzarrays[x][y+1][z]==0) {/* 下 */
          pixelQueueSize++;
      pixelQueuex[pixelQueueSize] = x; pixelQueuey[pixelQueueSize] = y+1; pixelQueuez[pixelQueueSize] = z;
      }
    }

    if (x>0) {
    if (xyzarrays[x-1][y][z]==0) {/* 左 */
       pixelQueueSize++;
      pixelQueuex[pixelQueueSize] = x-1; pixelQueuey[pixelQueueSize] = y; pixelQueuez[pixelQueueSize] = z;
    }
    }

    if (z<resolution-1) {
     if (xyzarrays[x][y][z+1]==0) {/* 下 */
            pixelQueueSize++;
      pixelQueuex[pixelQueueSize] = x; pixelQueuey[pixelQueueSize] = y; pixelQueuez[pixelQueueSize] = z+1;
    }
    }

    if (z>0) {
    if (xyzarrays[x][y][z-1]==0) {/* 手前 */
            pixelQueueSize++;
      pixelQueuex[pixelQueueSize] = x; pixelQueuey[pixelQueueSize] = y; pixelQueuez[pixelQueueSize] = z-1; }
    }



}

public void modelonoff() {
   if (mod==0) { mod = 1;}else { mod = 0;}
}

public void tubeonoff() {
   if (tub==0) { tub = 1;}else { tub = 0;}
}
```

```
public void make_porus() {
     //   ボクセルの描画

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

  int len = txts.length();
  int l = 0;

  FIELD_STEP = FIELD_SIZE*2/resolution;

  for(int  k  =int((resolution-1)*z_scan/100)+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;

        if(  xyzarrays[i-1][j-1][k-1] == 1 || xyzarrays[i-1][j-1][k-1] == -1)  {

           if (arrays[q]==1){
            PVector  start=new PVector((poly.f[q].v[3] + poly.f[q].v[6] + poly.f[q].v[ 9])/3- poly.f[q].v[0]*porusheight,
                                       (poly.f[q].v[4] + poly.f[q].v[7] + poly.f[q].v[10])/3- poly.f[q].v[1]*porusheight,
                                       (poly.f[q].v[5] + poly.f[q].v[8] + poly.f[q].v[11])/3- poly.f[q].v[2]*porusheight);

            PVector  end=new PVector((poly.f[q].v[3] + poly.f[q].v[6] + poly.f[q].v[ 9])/3+ poly.f[q].v[0]*porusheight,
                                     (poly.f[q].v[4] + poly.f[q].v[7] + poly.f[q].v[10])/3+ poly.f[q].v[1]*porusheight,
                                     (poly.f[q].v[5] + poly.f[q].v[8] + poly.f[q].v[11])/3+ poly.f[q].v[2]*porusheight);

            PVector  boxmin=new PVector(realx,realy,realz);
```

```
            PVector  boxmax=new PVector(realx+FIELD_STEP,realy+FIELD_STEP,realz+FIELD_STEP);


          if (intersect(start, end, boxmin, boxmax)==true) { xyzarrays[i-1][j-1][k-1] = 2;}
             }
       }
    }
  }
 }
      }
}


void controlEvent(ControlEvent theEvent) {
  // DropdownList is of type ControlGroup.
  // A controlEvent will be triggered from inside the ControlGroup class.
  // therefore you need to check the originator of the Event with
  // if (theEvent.isGroup())
  // to avoid an error message thrown by controlP5.

  if (theEvent.isGroup()) {
    // check if the Event was triggered from a ControlGroup
    println(theEvent.getGroup().getValue());
    println(theEvent.name());

    if ((theEvent.getGroup().getValue()==3.0) && (theEvent.name().equals("EFFECT_ON_SURFACE")  ==true)) { needlize = 1; }else {
    if ((theEvent.getGroup().getValue()==4.0) && (theEvent.name().equals("EFFECT_ON_SURFACE")  ==true)) { sprout = 1; }else { sp
    if ((theEvent.getGroup().getValue()==6.0) && (theEvent.name().equals("EFFECT_ON_SURFACE")  ==true)) { inflation = 1; }else

    if ((theEvent.getGroup().getValue()==1.0) && (theEvent.name().equals("FILE")  ==true))  {setup(); redraw();}
    if ((theEvent.getGroup().getValue()==2.0) && (theEvent.name().equals("FILE")  ==true)) {  load_STL();}
    if ((theEvent.getGroup().getValue()==3.0) && (theEvent.name().equals("FILE")  ==true)) {  load_VOXEL_txt();}
    if ((theEvent.getGroup().getValue()==4.0) && (theEvent.name().equals("FILE")  ==true)) {  load_VXC();}
    if ((theEvent.getGroup().getValue()==5.0) && (theEvent.name().equals("FILE")  ==true) && format !="") {  save_STL();}
    if ((theEvent.getGroup().getValue()==6.0) && (theEvent.name().equals("FILE")  ==true) && format !="") { save_VOXELtxt(); }
    if ((theEvent.getGroup().getValue()==7.0) && (theEvent.name().equals("FILE")  ==true)) {  save_VXC();}
    if ((theEvent.getGroup().getValue()==1.0) && (theEvent.name().equals("EFFECT_ON_SURFACE")  ==true)) { VOXELIZE(); }
    if ((theEvent.getGroup().getValue()==2.0) && (theEvent.name().equals("EFFECT_ON_SURFACE")  ==true)) {  VOXELIZE_FILL();  }
    if ((theEvent.getGroup().getValue()==1.0) && (theEvent.name().equals("EFFECT_ON_VOXEL")  ==true)) { transparent =true;  }els
```

```
    if ((theEvent.getGroup().getValue()==2.0) && (theEvent.name().equals("EFFECT_ON_VOXEL")  ==true)) { createMesh(); }
    if ((theEvent.getGroup().getValue()==3.0) && (theEvent.name().equals("EFFECT_ON_VOXEL")  ==true)) { tubeon =true;  }else {
    if ((theEvent.getGroup().getValue()==4.0) && (theEvent.name().equals("EFFECT_ON_VOXEL")  ==true))  {  make_porus();  }
    if ((theEvent.getGroup().getValue()==5.0) && (theEvent.name().equals("EFFECT_ON_VOXEL")  ==true)) { noiseon =true;  }else {
    if ((theEvent.getGroup().getValue()==6.0) && (theEvent.name().equals("EFFECT_ON_VOXEL")  ==true)) { dualon =true;  }else {
    if ((theEvent.getGroup().getValue()==7.0) && (theEvent.name().equals("EFFECT_ON_VOXEL")  ==true)) { sphon =true;  }else { s
    if ((theEvent.getGroup().getValue()==8.0) && (theEvent.name().equals("EFFECT_ON_VOXEL")  ==true)) { esoon =true; voxel_displ
}
}

// テキストが入力された際にコールバックされるメソッド
public synchronized void bit_pattern(String txt) {
     txts = txt;
    println(txts);
}
```