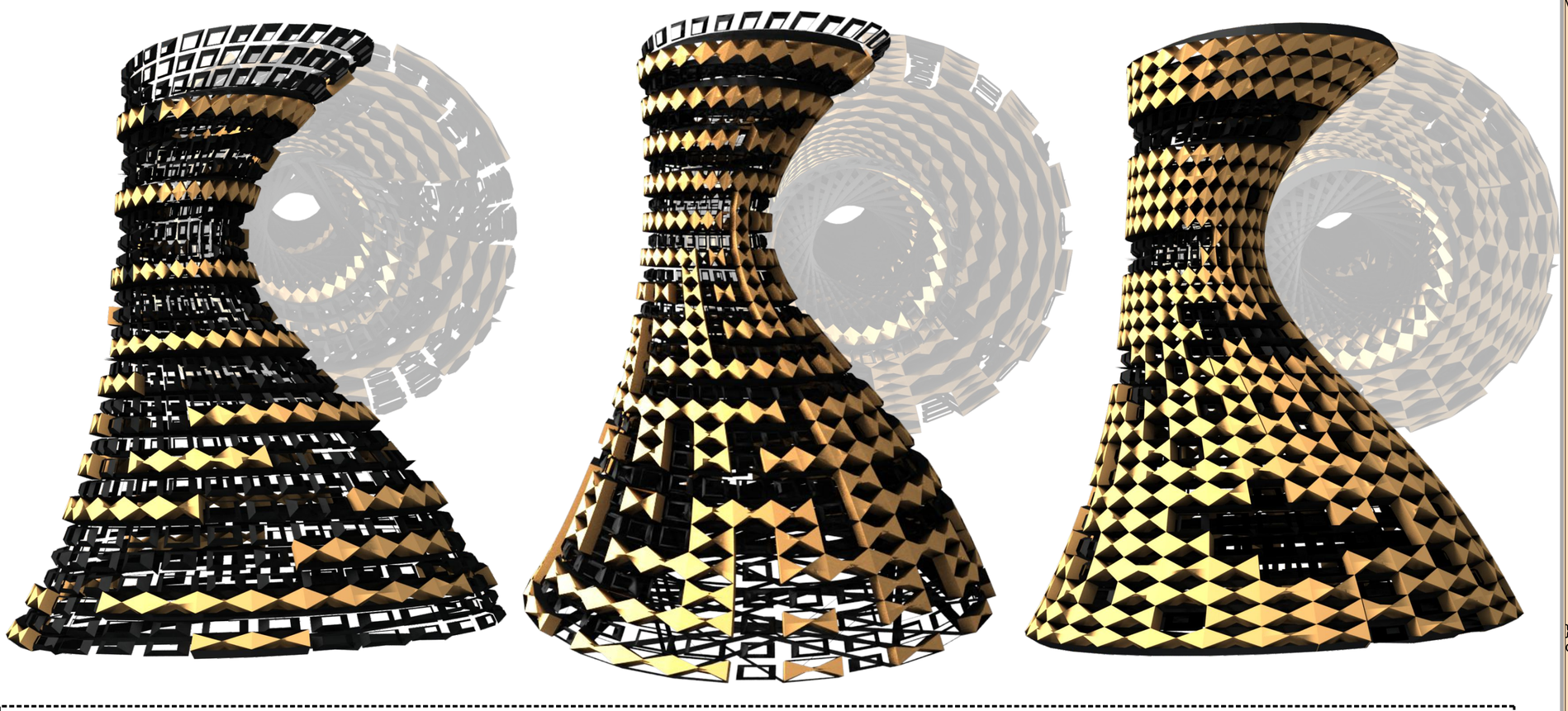
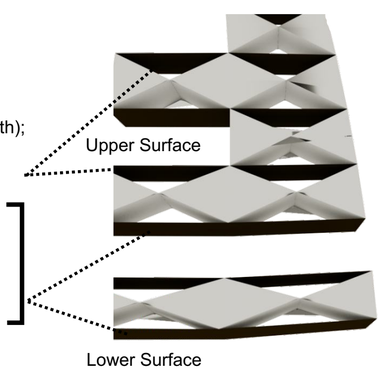
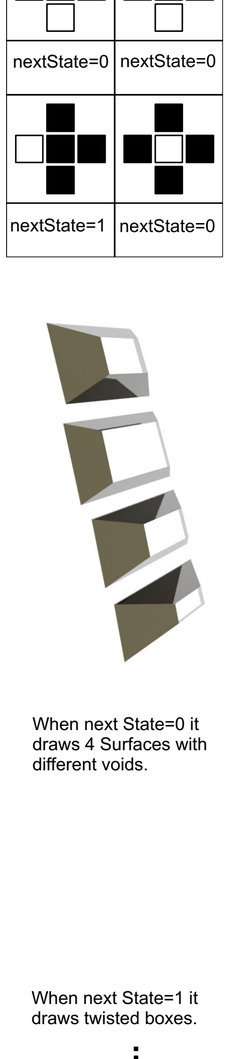


A cellular automaton is a discrete model studied in computability theory, mathematics, physics, complexity science, theoretical biology and micro structure modeling. It consists of a regular grid of cells, each in one of a finite number of states, such as "On" and "Off" (in contrast to a coupled map lattice). The grid can be in any finite number of dimensions. For each cell, a set of cells called its neighborhood (usually including the cell itself) is defined relative to the specified cell. Here in these different updates, there are different definitions for “next states” .


```
    Ivec[][] panelPts3 = new IVec[2][2];
    panelPts3[0][0] = surf.pt(pos.x+width, pos.y);
    panelPts3[0][1] = surf.pt(pos.x+width, pos.y+height);
    panelPts3[1][1] = surf.pt(pos.x+width, pos.y+height, depth);
    panelPts3[1][0] = surf.pt(pos.x+width, pos.y, depth);
    panelSurf2 = new ISurface(panelPts3).clr(0.1, 0.3, .3);

    Ivec[][] panelPts4 = new IVec[2][2];
    panelPts4[0][0] = surf.pt(pos.x, pos.y);
    panelPts4[0][1] = surf.pt(pos.x, pos.y+height);
    panelPts4[1][1] = surf.pt(pos.x, pos.y+height, depth);
    panelPts4[1][0] = surf.pt(pos.x, pos.y, depth);
    panelSurf3 = new ISurface(panelPts4).clr(0.1, 0.3, .3);
  state = nextState;
 }
}
}
```





When next State=0 it draws 4 Surfaces with different voids.

When next State=1 it draws twisted boxes.


```
import processing.opengl.*;
import igeo.*;
void setup() {
  size(800, 800, IG.GL);
  IG.duration(30);
  IG.open("1f.3dm");
  ISurface surf = IG.surface(0);
  // put automaton as panel
  int unum=30, vnum=30;
  double uinc=1.0/unum, vinc=1.0/vnum;
  MyAutomaton[][] automata = new MyAutomaton[unum][vnum];
  for (int i=0; i < unum; i++) {
    for (int j=0; j < vnum; j++) {
      automata[i][j] =
        new MyAutomaton(new IVec(i*uinc, j*vinc, 0),
      uinc, vinc, 1, surf);
    }
  }
  //connecting adjacent automata
  for (int i=0; i < unum; i++) {
    for (int j=0; j < vnum; j++) {
      if (i > 0) {
        automata[i][j].left=automata[i-1][j];
      }
      if (i < unum-1) {
        automata[i][j].right=automata[i+1][j];
      }
      if (j > 0) {
        automata[i][j].down=automata[i][j-1];
      }
      if (j < vnum-1) {
        automata[i][j].up=automata[i][j+1];
      }
      if (i==0) {
        automata[i][j].state = 1;
      }
    }
  }
  surf.del();
  IG.fill();
}
class MyAutomaton extends Iagent {
  IVec pos;
  double width, height, depth;
  int state=0, nextState=0;
  ISurface panelSurf1, panelSurf2, panelSurf3;
  ISurface surf;
  IBox box;
  MyAutomaton left, right, up, down;
  int count=0;
  MyAutomaton(IVec pt, double w, double h, double d,
  ISurface s) {
    pos = pt;
    width = w;
    height = h;
    depth = d;
    surf = s;
  }
```

```
void interact(ArrayList< IDynamics > agents) {
  super.interact(agents);
  int lstate=0, rstate=0, dstate=0, ustate=0;
  if (left!=null) {
    lstate = left.state;
  }
  if (right!=null) {
    rstate = right.state;
  }
  if (down!=null) {
    dstate = down.state;
  }
  if (up!=null) {
    ustate = up.state;
  }
  if (lstate==0 && rstate==0 && dstate==0 && ustate==0) {
    nextState=0;
  }
  else if (lstate==1 && rstate==0 && dstate==0 && ustate==0) {
    nextState=1;
  }
  else if (lstate==0 && rstate==1 && dstate==0 && ustate==0) {
    nextState=1;
  }
  else if (lstate==1 && rstate==1 && dstate==0 && ustate==0) {
    nextState=1;
  }
  else if (lstate==0 && rstate==0 && dstate==1 && ustate==0) {
    nextState=1;
  }
  else if (lstate==1 && rstate==0 && dstate==1 && ustate==0) {
    nextState=0;
  }
  else if (lstate==0 && rstate==1 && dstate==1 && ustate==0) {
    nextState=0;
  }
  else if (lstate==1 && rstate==1 && dstate==1 && ustate==0) {
    nextState=0;
  }
  else if (lstate==0 && rstate==0 && dstate==0 && ustate==1) {
    nextState=0;
  }
  else if (lstate==1 && rstate==0 && dstate==0 && ustate==1) {
    nextState=0;
  }
  else if (lstate==0 && rstate==1 && dstate==0 && ustate==1) {
    nextState=0;
  }
  else if (lstate==1 && rstate==1 && dstate==0 && ustate==1) {
    nextState=0;
  }
  else if (lstate==0 && rstate==0 && dstate==1 && ustate==1) {
    nextState=0;
  }
  else if (lstate==1 && rstate==0 && dstate==1 && ustate==1) {
    nextState=1;
  }
  else if (lstate==0 && rstate==1 && dstate==1 && ustate==1) {
    nextState=1;
  }
  else if (lstate==1 && rstate==1 && dstate==1 && ustate==1) {
    nextState=0;
  }
}
```

```
void update() {
  super.update();
  if (nextState!=state) { // state change
    synchronized(IG.lock) { // to suppress flickering of display update
      if (panelSurf1!=null) panelSurf1.del();
      if (panelSurf2!=null) panelSurf2.del();
      if (panelSurf3!=null) panelSurf3.del();
      if (nextState==0) {
        Ivec[][] panelPts = new IVec[2][4];
        panelPts[0][0] = surf.pt(pos.x, pos.y, 1);
        panelPts[0][1] = surf.pt(pos.x+width, pos.y, 1);
        panelPts[0][3] = surf.pt(pos.x, pos.y+height, 1);
        panelPts[0][2] = surf.pt(pos.x+width, pos.y+height, 1);
        double factor = IRandom.get(0, .5);
        IVec center = surf.pt(pos.x+width/2, pos.y+height/2);
        IVec nml = surf.nml(pos.x+width/2, pos.y+height/2);
        IVec shift = nml.dup().len(depth*count*0.02);
        panelPts[1][0] = panelPts[0][0].dup().add(shift).scale(center, factor);
        panelPts[1][1] = panelPts[0][1].dup().add(shift).scale(center, factor);
        panelPts[1][3] = panelPts[0][3].dup().add(shift).scale(center, factor);
        panelPts[1][2] = panelPts[0][2].dup().add(shift).scale(center, factor);
        panelSurf1 = new ISurface(panelPts, 2, 1, true, true).clr(count*0.02);
        if (nextState==0 && box!=null) {
          synchronized(IG.lock) { // to suppress flickering of display update
            box.del();
            box = null;
          }
        }
      }
```

```
      else if (nextState==1) {
        //4 by 3 control points
        Ivec[][][] BoxPts = new IVec[2][2][2];
        //rectangular border
        BoxPts[0][0][0] = surf.pt(pos.x, pos.y);
        BoxPts[1][0][0] = surf.pt(pos.x+width, pos.y);
        BoxPts[0][1][0]= surf.pt(pos.x+width, pos.y+height);
        BoxPts[1][1][0] = surf.pt(pos.x, pos.y+height);
        BoxPts[0][0][1] = surf.pt(pos.x, pos.y, depth);
        BoxPts[1][0][1] = surf.pt(pos.x+width, pos.y, depth);
        BoxPts[0][1][1]= surf.pt(pos.x+width, pos.y+height, depth);
        BoxPts[1][1][1] = surf.pt(pos.x, pos.y+height, depth);
        box = new IBox(BoxPts).clr(0.9);

      if (panelSurf1!=null) panelSurf1.del();
        if (panelSurf2!=null) panelSurf2.del();
        if (panelSurf3!=null) panelSurf3.del();


          Ivec[][] panelPts2 = new IVec[2][2];
        panelPts2[0][0] = surf.pt(pos.x, pos.y);
        panelPts2[0][1] = surf.pt(pos.x, pos.y+10*height);
        panelPts2[1][1] = surf.pt(pos.x, pos.y+10*height, depth);
        panelPts2[1][0] = surf.pt(pos.x, pos.y, depth);
        panelSurf1 = new ISurface(panelPts2).clr(0.1, 0.2, 0.2);

      }
    }
```

Mehrzad Rafeei
Coding form
Sci-arc, Fall 2011