Cellular automata visualization/game:https://openprocessing.org/sketch/1887581
A cellular automata:https://openprocessing.org/sketch/1887605
Stippling Art: https://openprocessing.org/sketch/1886727
Rock Paper Scissors: https://openprocessing.org/sketch/1886614

The code for everything:

```
ArrayList<String> linesIncluded = new ArrayList<String>(); //will add a new string to the list each time
:)
ArrayList<String> miniArray = new ArrayList<String>(); //will add a new string to the list each time :)
String startingString = "";
String changeColorString = "";
String stringForTheRow = "cccccccc";
int skip = 1;
int whatRowItsOn = 8;
int howManyTrue = 0;
int howManyFalse = 0;
boolean repeat = false;
int changeColor,ellipseHeight; //changes from black to light or dark blue
int amountOfTimesClicked = 0;
String currentColor="lightBlue";
int lengthOfTHeRow = 9;
int actualLength = lengthOfTHeRow;
String theRowString;

int zoom;
int amountOfTimesSpacePressed;
ArrayList<String> rulesForBlackTile= new ArrayList<String>();
  rulesForBlackTile.add("110");
  rulesForBlackTile.add("101");
  rulesForBlackTile.add("011");
  rulesForBlackTile.add("000");


ArrayList<String> rulesForWhiteTile= new ArrayList<String>();
  rulesForWhiteTile.add("111");
  rulesForWhiteTile.add("100");
  rulesForWhiteTile.add("010");
  rulesForWhiteTile.add("000");
int amountOfTimesToIterate = 45;
int amountOfTimesRun = amountOfTimesToIterate;
int changingHeight, changingWidth;
```

```
void setup() {
// background(40,100,200);
   makeStartingString();
  size(800,800,P3D);
            changingHeight = height/(startingString.length());
   changingWidth = width/(startingString.length());
  drawTheList(startingString);

}

void draw(){
 // background(60,120,120);
  if(frameCount<200){
            stroke(30,frameCount%800,30,50);
  strokeWeight(3.5);
  drawSpheres(height*3/4, width*2/4, 100+frameCount,300);
  strokeWeight(6.5);
  stroke(0,255-frameCount%800,0,30);
  drawSpheres(height*3/4+5, width*2/4+5, -50+frameCount,350);
  stroke(20,180-frameCount%800, 50,20);
  strokeWeight(2);
  drawSpheres(height*1/4, width*2/5, 50+frameCount,150);

   stroke(20,20,180-frameCount%800,20);
  strokeWeight(3.5);
  drawSpheres(height*3/4, width*1/5, 50+frameCount,450);

  }



 else if(frameCount<400){
  amountOfTimesToIterate = 7;
  makeCellularAutomataPatterns(7);
//size(400, 400, P3D);



  }
  else if(amountOfTimesSpacePressed==0){
textSize(27);
fill(10,60,90);
//fill(50,80,200,50);
text("Click on the blue circles at the top right corner to change the color \n of the mouse :). When you click on the black circles they will fill with\n the color and a red or green circle will appear if you are right or not,\n after each row is filled in, the correct version of the row will appear\n\n there is a pattern can you get more than half right???  \n\n press the up arrow to continue!!!",0, 90, -120);  // Specify a z-axis value

  }
   else if(amountOfTimesSpacePressed==1){
            //cellular automata initialize
            amountOfTimesToIterate = 45;



             ellipseHeight = 60;

            for(int i=0; i< (amountOfTimesToIterate+1)/2-1; i++){
            startingString+="0";
            }
            startingString+="1";
            for(int j=0; j< (amountOfTimesToIterate+1)/2-1; j++){
            startingString+="0";
            }
            startingString+="0";
            changingHeight = height/(startingString.length());
   changingWidth = width/(startingString.length());
  drawTheList2(startingString);
            background(0,0,0);
   }

   else if(amountOfTimesSpacePressed==2){
            if(currentColor.equals("lightBlue")){
            fill(10,40,125);
   ellipse(30,30,50,50);

 }

 else{
   fill(10,125,230);
   ellipse(30, 30,50,50);
 }


  strokeWeight(2);
  stroke(30,70,200);
  changingWidth = width/(startingString.length());

 //  drawTheList("01011110");
 if(amountOfTimesRun>0){
            drawTheList2(startingString);
  startingString= useTheRules(startingString,rulesForBlackTile, rulesForWhiteTile);
   background(0,0,0);

  }
  else if(amountOfTimesRun==0){
   background(30,40,90);
  changingHeight = height/(actualLength-1);
   changingWidth = width/actualLength;
   fill(10,40,125);
   ellipse(width-800/6, 800/6/4,40,40);
   fill(10,125,230);
   ellipse(width-800/6*2, 800/6/4,40,40);



  }
  else if(amountOfTimesRun==-1){
   // for(int i=4; i<13; i++){
            miniArray.add("110011111");
                        miniArray.add("001001111");
                        miniArray.add("101100111");
                        miniArray.add("000010011");
                        miniArray.add("111011001");
                        miniArray.add("110000100");
                        miniArray.add("001110110");
                        miniArray.add("100100001");
                        miniArray.add("010111101");
   // }


            drawTheList2(miniArray.get(0));
            changingHeight+=height/lengthOfTHeRow;
  }
   else if(amountOfTimesRun==-2){

            drawTheList2("wwwwwwwww");
            changingHeight+=height/lengthOfTHeRow;

   }
```

```
        else if(amountOfTimesRun==-3){
        drawTheList2("wwwwwwwww");
        changingHeight+=height/lengthOfTHeRow;

    }
        else if(amountOfTimesRun==-4){
        drawTheList2("wwwwwwwww");
        changingHeight+=height/lengthOfTHeRow;

    }
        else if(amountOfTimesRun==-5){
        //drawTheList(miniArray.get(1));
        drawTheList2("wwwwwwwww");
        changingHeight+=height/lengthOfTHeRow;


    }
        else if(amountOfTimesRun==-6){
        drawTheList2("wwwwwwwww");
        changingHeight+=height/lengthOfTHeRow;

    }
        else if(amountOfTimesRun==-7){
        drawTheList2("wwwwwwwww");
        changingHeight+=height/lengthOfTHeRow;

    }
        else if(amountOfTimesRun==-8){
        drawTheList2("wwwwwwwww");
        changingHeight+=height/lengthOfTHeRow;

    }

    else{
          noStroke();
          fill(30,40,90);
          rect(750, 750, 50,50);

        //drawTheList(linesIncluded.get(3));
    }
   amountOfTimesRun -=1;
```

```
    }
    else if(amountOfTimesSpacePressed==3){
            background(0,0,0);
            fill(200,200,30);

            textSize(45);
            text(howManyTrue+"/"+72, height/2, width/2);  // Specify a z-axis value
    //    textSize(20);
    //    text("Press Space to Continue! The next part is similar to stippling art", height/5, 4*width/5)  //
Specify a z-axis value
    textSize(27);
    text("press space to continue!!! ", 0, 600);
            //fill(10,60,90);




    }
    else if(amountOfTimesSpacePressed == 4){
            startingString = "";
           skip = 1;
            amountOfTimesToIterate = 45;
           amountOfTimesRun = amountOfTimesToIterate;

             for(int i=0; i< (amountOfTimesToIterate+1)/2-1; i++){
           startingString+="0";
           }
           startingString+="1";
           for(int j=0; j< (amountOfTimesToIterate+1)/2-1; j++){
           startingString+="0";
           }
           startingString+="0";
           changingHeight = height/(startingString.length());
   changingWidth = width/(startingString.length());
  drawTheList(startingString);
            amountOfTimesSpacePressed = 5;
    }


    else if(amountOfTimesSpacePressed==5){

             strokeWeight(2);
  stroke(30,70,200);
  changingWidth = width/(startingString.length());
  if(amountOfTimesRun>0){
            drawTheList(startingString);
  startingString= useTheRules(startingString,rulesForBlackTile, rulesForWhiteTile);

  changingHeight+=height/(amountOfTimesToIterate);
  }
  amountOfTimesRun -=1;



    }



}

void drawSpheres(int x_, int y_, int zoom, int size){
  pushMatrix();
  translate(x_, y_, zoom);
//fill(random(200), random(200), random(200));
noFill();
sphere(size);
popMatrix();



}

void makeCellularAutomataPatterns(int n){
   noFill();
  strokeWeight(1);
  stroke(30,70,200);

  changingWidth = width/(startingString.length());
  if(amountOfTimesRun>0){
            drawTheList(startingString);
  startingString= useTheRules(startingString,rulesForBlackTile, rulesForWhiteTile);

  changingHeight+=height/(amountOfTimesToIterate);
  amountOfTimesRun -=1;
  zoom+=1;
```

```
 }
 else{

            amountOfTimesToIterate = n;
            makeStartingString();
            amountOfTimesRun = amountOfTimesToIterate;



            makeStartingString();
            changingHeight = height/(startingString.length());
  changingWidth = width/(startingString.length());
 drawTheList(startingString);
 }



}

void drawTheList(String inputedString){
  noFill();
lights();
            //changingWidth = 100;
 for(int i=0; i<inputedString.length(); i++){

  stroke(10,30,115,20+frameCount/20);

  if(inputedString.substring(i,i+1).equals("0")){

            pushMatrix();

            translate(changingWidth, changingHeight, zoom);
            sphere(height/(amountOfTimesToIterate)/2);
            popMatrix();
 // ellipse(changingWidth-(height/(inputedString.length()/2)*3/4),
changingHeight,height/(inputedString.length()),height/(inputedString.length()));
  }
  else{

            stroke(10,120,230,20+frameCount/20);
            pushMatrix();
            translate(changingWidth, changingHeight,zoom);
            sphere(height/(amountOfTimesToIterate)/2);
            popMatrix();
```

```
        //  ellipse(changingWidth,
changingHeight,height/(amountOfTimesToIterate),height/(amountOfTimesToIterate));
        //ellipse(changingWidth-(height/(inputedString.length()/2)*3/4),
changingHeight,height/(inputedString.length()),height/(inputedString.length()));
   }
  changingWidth+=width/(inputedString.length());
}
 }

 String useTheRules(String stringToBeChanged, ArrayList<String> rulesForWhiteTiles,
ArrayList<String> rulesForBlackTiles){
  String stringToBeReturned = stringToBeChanged;
   for(int i=0; i< stringToBeChanged.length()-2; i++){
        for(int j=0; j<rulesForBlackTiles.size(); j++){
        if(stringToBeChanged.substring(i,i+3).equals(rulesForBlackTiles.get(j))){
        stringToBeReturned =
stringToBeReturned.substring(0,i+1)+"1"+stringToBeReturned.substring(i+2,
stringToBeReturned.length());
        }
        // else{
        else if(stringToBeChanged.substring(i,i+3).equals(rulesForWhiteTiles.get(j))){

        stringToBeReturned =
stringToBeReturned.substring(0,i+1)+"0"+stringToBeReturned.substring(i+2,
stringToBeReturned.length());
        //}
   }
        }
   }
  return stringToBeReturned;
 }


 void makeStartingString(){
   startingString = "";
        for(int i=0; i< (amountOfTimesToIterate+1)/2-1; i++){
        startingString+="0";
        }
        startingString+="1";
        for(int j=0; j< (amountOfTimesToIterate+1)/2-1; j++){
        startingString+="0";
        }
        startingString+="0";
 }
```

```
void keyPressed(){
 if(keyCode == UP){
 amountOfTimesSpacePressed++;
  }
}









void drawTheList2(String inputedString){

   linesIncluded.add(inputedString);
		//changingWidth = 100;
  for(int i=0; i<inputedString.length(); i++){
 if(inputedString.equals("wwwwwwwww")){
  fill(0,0,0);
 }
 else{
   fill(10,40,125);
 }

   if(inputedString.substring(i,i+1).equals("0")){

		ellipse(changingWidth, changingHeight,height/(20),height/(20));
  // ellipse(changingWidth-(height/(inputedString.length()/2)*3/4), 
changingHeight,height/(inputedString.length()),height/(inputedString.length()));
   }
   else if((inputedString.substring(i,i+1).equals("w"))){
		fill(0,0,0);
		ellipse(changingWidth, changingHeight,height/(20),height/(20));//the minus 4 is just to make 
them bigger
   }
   else if((inputedString.substring(i,i+1).equals("c"))){

   }
   else{
		if(inputedString.equals("wwwwwwwww")){
		fill(0,0,0);
		}
  else{
	fill(10,125,230);
	}
	ellipse(changingWidth, changingHeight,height/(20),height/(20));
	//ellipse(changingWidth-(height/(inputedString.length()/2)*3/4),
changingHeight,height/(inputedString.length()),height/(inputedString.length()));
  }
 changingWidth+=width/(inputedString.length());
}
 }

 String useTheRules(String stringToBeChanged, ArrayList<String> rulesForWhiteTiles,
ArrayList<String> rulesForBlackTiles){
 String stringToBeReturned = stringToBeChanged;
  for(int i=0; i< stringToBeChanged.length()-2; i++){
	for(int j=0; j<rulesForBlackTiles.size(); j++){
	if(stringToBeChanged.substring(i,i+3).equals(rulesForBlackTiles.get(j))){
	stringToBeReturned =
stringToBeReturned.substring(0,i+1)+"1"+stringToBeReturned.substring(i+2,
stringToBeReturned.length());
	}
	// else{
	else if(stringToBeChanged.substring(i,i+3).equals(rulesForWhiteTiles.get(j))){

	stringToBeReturned =
stringToBeReturned.substring(0,i+1)+"0"+stringToBeReturned.substring(i+2,
stringToBeReturned.length());
	//}
  }
	}
  }
 return stringToBeReturned;
 }



//MOUSE PRESSED!!!

void mousePressed() {//when mouse is pressed, checks if the anwer is correct, based on which textbox is
clicked.
 if(mouseX >((width-width/14-90)) && mouseX<((width-width/14-60))){
	if(mouseY>(height/14-35) && mouseY<(height/14+5)){
 currentColor ="lightBlue";
  }
  }

  if(mouseX >((width-width/7-165)) && mouseX<((width-width/7 -130))){
        if(mouseY>(height/14-35) && mouseY<(height/14+5)){
 currentColor="darkBlue";
  }
  }

   for(int i=0; i<9; i++){
   if(mouseY>(height-height/9*i)-20 &&mouseY<(height-height/9*i)+20){ //targets the row

   for(int z=1; z<10; z++){
        if(mouseX>(width-width/9*z)-20 &&mouseX<(width-width/9*z)+20 &&i==checkTheRow()-1){
//targets the column

        changeColor = z;
        changeColorString=changeColorString+String.valueOf(z);

        for(int p=0; p<changeColorString.length()-1; p++){
        if(changeColorString.substring(p,p+1).equals(String.valueOf(z))){
        repeat = true;
        }
        else{
        }
        }

        if(!repeat){
        amountOfTimesClicked++;
        changingHeight = 0;
        changingHeight
=whatRowItsOn+(lengthOfTHeRow-whatRowItsOn+1)*height/lengthOfTHeRow+4;
        if(currentColor.equals("lightBlue")){
         //     if(whatRowItIsOn!=4){
        drawTheList2( stringForTheRow.substring(changeColor-1,
stringForTheRow.length()-1)+"0"+stringForTheRow.substring(0,changeColor-1));
         //     }
        String stringToBeChecked = miniArray.get(9-whatRowItsOn);


        if(stringToBeChecked.substring(changeColor-1,changeColor).equals("0")){
        fill(20,200,20);
        ellipse(770, ellipseHeight,30,30);
        ellipseHeight +=90;
        howManyTrue++;
```

```
            }
            else{
            fill(200,20,20);
            ellipse(770, ellipseHeight,30,30);
            ellipseHeight +=90;
            howManyFalse++;
            }
            }
            else{
                    // if(whatRowItIsOn!=4){
            drawTheList2( stringForTheRow.substring(changeColor-1,
stringForTheRow.length()-1)+"1"+stringForTheRow.substring(0,changeColor-1));
             //     }
            String stringToBeChecked = miniArray.get(whatRowItsOn);
            if(stringToBeChecked.substring(changeColor-1,changeColor).equals("1")){
            fill(20,200,20);
            ellipse(770, ellipseHeight,30,30);
            ellipseHeight +=90;
            howManyTrue++;
            }
            else{

            fill(200,20,20);
            ellipse(770, ellipseHeight,30,30);
            ellipseHeight +=90;
            howManyFalse++;

            }
              }

  }
  }
  }
 }
  }
  if(amountOfTimesClicked==actualLength){
            stringForTheRow = "ccccccccc";
            changeColorString = "";
            fill(100,100,100);
            changingWidth=20;
            String reversedString = "";
            for(int i=0; i<miniArray.get(9-whatRowItsOn).length(); i++){
            reversedString = miniArray.get(9-whatRowItsOn).substring(i,i+1)+reversedString;
```

```
          }
            changingWidth-=20;
            //if(whatRowItsOn!=6){
          drawTheList2(reversedString);

           // }
            //else{
           //     for(int i=0; i<miniArray.get(4).length(); i++){
          //          reversedString = miniArray.get(4).substring(i,i+1)+reversedString;

          //  }
                       // drawTheList2(reversedString);
           // }

          changingWidth+=20;

          whatRowItsOn--;
          fill(30,40,90,100);
          noStroke();

          ellipseHeight= 60;
  amountOfTimesClicked=0;
            if(whatRowItsOn==1){
                       amountOfTimesSpacePressed++;
            }

             rect(750, 0, 100, 850);
  }



  repeat = false;
 }

 int checkTheRow(){
  return whatRowItsOn;

 }

//MOUSE PRESSED!!!
```

```
ArrayList<String> linesIncluded; //will add a new string to the list each time :)
String startingString = "";
int skip = 1;

ArrayList<String> rulesForBlackTile= new ArrayList<String>();
  rulesForBlackTile.add("110");
  rulesForBlackTile.add("101");
  rulesForBlackTile.add("011");
  rulesForBlackTile.add("000");


ArrayList<String> rulesForWhiteTile= new ArrayList<String>();
  rulesForWhiteTile.add("111");
  rulesForWhiteTile.add("100");
  rulesForWhiteTile.add("010");
  rulesForWhiteTile.add("000");


int amountOfTimesToIterate = 101;
int amountOfTimesRun = amountOfTimesToIterate;
int changingHeight, changingWidth;

void setup(){
        for(int i=0; i< (amountOfTimesToIterate+1)/2-1; i++){
        startingString+="0";
        }
        startingString+="1";
        for(int j=0; j< (amountOfTimesToIterate+1)/2-1; j++){
        startingString+="0";
        }
        startingString+="0";
  size(800,800,P3D);

  //changingWidth = width/(startingString.length());
        changingHeight = height/(startingString.length());
   changingWidth = width/(startingString.length());
  //changingHeight = (height/(startingString.length()*amountOfTimesToIterate));
  drawTheList(startingString);
  //translate(0, -changingHeight);



  }

void draw(){

  strokeWeight(2);
  stroke(30,70,200);

  changingWidth = width/(startingString.length());
  if(amountOfTimesRun>0){
            drawTheList(startingString);
  startingString= useTheRules(startingString,rulesForBlackTile, rulesForWhiteTile);

  changingHeight+=height/(amountOfTimesToIterate);
  }
  amountOfTimesRun -=1;

}

 void drawTheList(String inputedString){
            //changingWidth = 100;
  for(int i=0; i<inputedString.length(); i++){

   fill(10,random(20,50),random(80,150));

   if(inputedString.substring(i,i+1).equals("0")){

            ellipse(changingWidth,
changingHeight,height/(amountOfTimesToIterate),height/(amountOfTimesToIterate));
  // ellipse(changingWidth-(height/(inputedString.length()/2)*3/4),
changingHeight,height/(inputedString.length()),height/(inputedString.length()));
   }
   else{
            fill(10,random(100,150),random(200,250));
            ellipse(changingWidth,
changingHeight,height/(amountOfTimesToIterate),height/(amountOfTimesToIterate));
            //ellipse(changingWidth-(height/(inputedString.length()/2)*3/4),
changingHeight,height/(inputedString.length()),height/(inputedString.length()));
   }
  changingWidth+=width/(inputedString.length());
}
```

```
 }

 String useTheRules(String stringToBeChanged, ArrayList<String> rulesForWhiteTiles,
ArrayList<String> rulesForBlackTiles){
  String stringToBeReturned = stringToBeChanged;
   for(int i=0; i< stringToBeChanged.length()-2; i++){
            for(int j=0; j<rulesForBlackTiles.size(); j++){
            if(stringToBeChanged.substring(i,i+3).equals(rulesForBlackTiles.get(j))){
            stringToBeReturned =
stringToBeReturned.substring(0,i+1)+"1"+stringToBeReturned.substring(i+2,
stringToBeReturned.length());
            }
            // else{
            else if(stringToBeChanged.substring(i,i+3).equals(rulesForWhiteTiles.get(j))){

            stringToBeReturned =
stringToBeReturned.substring(0,i+1)+"0"+stringToBeReturned.substring(i+2,
stringToBeReturned.length());
            //}
   }
            }
   }
  return stringToBeReturned;
 }


PImage cat, otter, fox,coyote,flower5,jaguar,cat2,geometry,flower9, fractal, buildings;
int division;
ArrayList<PImage> currentImages;
float amount, timesTheImagesChange;
int  amountOfTimesSpacePressed;

void setup(){
  amountOfTimesSpacePressed = 0;
   timesTheImagesChange = 0;
  amount = 3;

  size(900,900,P3D);
 // img = loadImage("cat.jpg");

 // img.resize(300, 300);

  // flower1.resize(300, 300);
   cat = loadImage("cat.jpg");
   cat.resize(300, 300);
   otter = loadImage("otter.jpg");
   otter.resize(300, 300);
   fox = loadImage("fox.jpg");
   fox.resize(300, 300);
   coyote = loadImage("coyote.jpg");
   coyote.resize(300, 300);
   jaguar = loadImage("jaguar.jpg");
   jaguar.resize(300, 300);
   cat2 = loadImage("cat2.jpg");
   cat2.resize(300, 300);
   geometry = loadImage("geometry.jpg");
   geometry.resize(300, 300);
   flower9 = loadImage("flower9.jpg");
   flower9.resize(300, 300);
   fractal = loadImage("fractal.jpg");
   fractal.resize(300,300);
   currentImages =  new ArrayList<PImage>();
   currentImages.add(fox);
   currentImages.add(cat);
   currentImages.add(otter);
   currentImages.add(coyote);
   currentImages.add(jaguar);
   currentImages.add(cat2);
   currentImages.add(geometry);
   currentImages.add(flower9);
   currentImages.add(fractal) ;
   buildings = loadImage("building.jpg");

}

void draw() {
   switch(amountOfTimesSpacePressed) {
 case 0:
  background(#E5FBFF);
  fill(0);
  noStroke();
  sphereDetail(5);
    frameRate(1);

int i = int(random(0,currentImages.size()));
makeGrid(1, 1, 1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(1,-1,1, currentImages.get(i));
```

```
i = int(random(0,currentImages.size()));
makeGrid(1,-1,-1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(1,1,-1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(2, 2, 1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(2, 1, 1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(2, 0, 1, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(2,1,2, currentImages.get(i));
i = int(random(0,currentImages.size()));
makeGrid(2,1,0, currentImages.get(i));

break;


case 1:
                    frameRate(60);
background(255);
amount = 1;
cat.resize(900, 900);
makeGrid(0,0,0,buildings);
break;

case 2:
background(255);

makeGrid(0,0,0,cat);
break;

case 3:
background(255);

makeGrid(0,0,0,cat2);
break;


case 4:
background(255);

makeGrid2(0,0,0,cat2);
break;
```

```
  }
}

void makeGrid(int e, int f,int g, PImage img){
          pushMatrix();
  translate(e*width/amount,e*height/amount);
//  rotateY(radians(frameCount));
          float divisions = map(division, 0, 900, 0, 400);
          float amountOfDevisions = (width/amount)/divisions;

          for (int x = 0; x < divisions; x++) {
          for (int y = 0; y < divisions; y++) {
          color c = img.get(int(x*amountOfDevisions),int(y*amountOfDevisions));
          float b = map(brightness(c),0,255,1,0);
          // float z = map(b,0,1,-150,150);
          pushMatrix();

          translate(x*amountOfDevisions - (f*width/amount), y*amountOfDevisions - (g*height/amount));
          fill(random(10), random(20),random(30));

          if(amountOfTimesSpacePressed==3){
          fill(random(0,255),random(0,255),random(0,255));

          }
          sphere(amountOfDevisions*b*0.8);
          popMatrix();
  }
}
  popMatrix();
}


void makeGrid2(int e, int f,int g, PImage img){
          pushMatrix();
  translate(e*width/amount,e*height/amount);
//  rotateY(radians(frameCount));
          float divisions = map(mouseX, 0, 900, 0, 400);
          float amountOfDevisions = (width/amount)/divisions;

          for (int x = 0; x < divisions; x++) {
          for (int y = 0; y < divisions; y++) {
          color c = img.get(int(x*amountOfDevisions),int(y*amountOfDevisions));
          float b = map(brightness(c),0,255,0,1);
          // float z = map(b,0,1,-150,150);
```

```
            pushMatrix();

            translate(x*amountOfDevisions - (f*width/amount), y*amountOfDevisions - (g*height/amount));
            fill(random(10), random(20),random(30));

            if(amountOfTimesSpacePressed==3){
            fill(random(0,255),random(0,255),random(0,255));

            }
            sphere(amountOfDevisions*b*0.8);
            popMatrix();
  }
}
  popMatrix();
}

void keyPressed(){
  if(amountOfTimesSpacePressed<20){
  amountOfTimesSpacePressed++;
  }
  division=0;

}

void mousePressed() {
  division+=5;


}


ArrayList<String> linesIncluded; //will add a new string to the list each time :)
String startingString = "";
int skip = 1,zoom;

ArrayList<String> rulesForBlackTile= new ArrayList<String>();
  rulesForBlackTile.add("111");
  rulesForBlackTile.add("101");
  rulesForBlackTile.add("010");
  rulesForBlackTile.add("000");

   /* add("111");
            add("110");
            add("101");
            add("000"); */

ArrayList<String> rulesForWhiteTile= new ArrayList<String>();
  rulesForWhiteTile.add("110");
  rulesForWhiteTile.add("100");
  rulesForWhiteTile.add("011");
  rulesForWhiteTile.add("001");

            /*add("100");
            add("011");
            add("010");
            add("001");*/

ArrayList<String> yourWinsTiesAndLosses = new ArrayList<String>();
String computersPastChoice, computersNewChoice, yourChoice;
boolean choiceMade;
boolean acceptingInput;
int amountOfTimesToPlay;
PImage backgroundImage, rock, paper, scissors, win, lose,tie ;
int rockAfterPaper, scissorsAfterPaper, paperAfterPaper,
rockAfterRock,scissorsAfterRock,paperAfterRock,rockAfterScissors,scissorsAfterScissors,paperAfterSci
ssors;
void setup(){
   size(800,600);
  amountOfTimesToPlay = 100;
  rockAfterPaper=2;
  scissorsAfterPaper=9;
  paperAfterPaper=10;
  rockAfterRock=6;
  scissorsAfterRock=8;
  paperAfterRock=10;
  rockAfterScissors=1;
  scissorsAfterScissors=4;
  paperAfterScissors=10;
  choiceMade = false;
  computersPastChoice = "paper";
   rock=loadImage("rockImage.jpg"); //rock

   paper=loadImage("paperImage.jpg"); //paper
   scissors=loadImage("scissorsImage.jpg"); //scissors

            win=loadImage("youWon.jpg"); //win
   lose=loadImage("youLost.jpg"); //lost
   tie = loadImage("youTied.jpg"); //tie
  background( 100,100,100);
}
void draw(){
  //display(rock, 100,100,100,100);
  //display(paper, 200,200,100,100);
  //display(scissors, 300,300,100,100);

 if(amountOfTimesToPlay>0){
        int input = int(random(1,10));
        //  System.out.print(input);

        if(computersPastChoice.equals("paper")){
        if(input<=rockAfterPaper){
        computersNewChoice = "rock" ;
        }
        else if(input<=scissorsAfterPaper&&input>rockAfterPaper){
        computersNewChoice = "scissors";
        }
        else{
        computersNewChoice = "paper";
        }
        }
        else if(computersPastChoice.equals("rock")){
        if(input<=rockAfterRock){
        computersNewChoice = "rock" ;
        }
        else if(input<=scissorsAfterRock&&input>rockAfterRock){
        computersNewChoice = "scissors";
        }
        else{
        computersNewChoice = "paper";
        }
        }
        else{
        if(input<=rockAfterScissors){
        computersNewChoice = "rock" ;
        }
        else if(input<=scissorsAfterScissors&&input>rockAfterScissors){
        computersNewChoice = "scissors" ;


        }
        else{
```

```
        computersNewChoice = "paper" ;

        }
        }
        if(choiceMade){

        if(computersNewChoice.equals("scissors")){
        display(scissors, 500,100,100,100);
        }
        else if(computersNewChoice.equals("rock")){
        display(rock, 500,100,100,100);
        }
        else{
        display(paper, 500,100,100,100);
        }

        if((yourChoice.equals("rock")
&&computersNewChoice.equals("scissors"))||(yourChoice.equals("paper")
&&computersNewChoice.equals("rock"))||(yourChoice.equals("scissors")&&computersNewChoice.equal
s("paper"))){
        yourWinsTiesAndLosses.add("win");
        display(win, 100,300,400,400);
        fill(100,50,50);
        //ellipse(100, 100,100,100);
        }
        else if((yourChoice.equals("rock")
&&computersNewChoice.equals("paper"))||(yourChoice.equals("paper")
&&computersNewChoice.equals("scissors"))||(yourChoice.equals("scissors")
&&computersNewChoice.equals("rock"))){
        yourWinsTiesAndLosses.add("loss");
        display(lose, 100,300,400,400);
        fill(50,100,50);
        // ellipse(100, 100,100,100);
        }
        else{
        yourWinsTiesAndLosses.add("tie");
        display(tie, 100,300,400,400);
        fill(50,50,10);
        //  ellipse(100, 100,100,100);
        }

        computersPastChoice = computersNewChoice;
        yourChoice = " ";
        choiceMade=false;
```

```
            }
  }
  else{
//   System.out.print(yourWinsTiesAndLosses);
//   System.out.print(amountOfTimesToPlay);

            float winStreak=0;
  for (int i=0; i<yourWinsTiesAndLosses.size(); i++){
   if(yourWinsTiesAndLosses.get(i).equals("win")){
            winStreak++;
   }
   else if(yourWinsTiesAndLosses.get(i).equals("tie")){
            winStreak +=.5;
   }
  }
  float percentage=winStreak/amountOfTimesToPlay;
    background(0,0,0);
             textSize(45);
  text(winStreak+"  expected: 50",width/4, height/2);

  }
}
void keyPressed(){

  if(keyCode == RIGHT){
            yourChoice = "paper";
            display(paper, 100,100,100,100);
            choiceMade = true;
            amountOfTimesToPlay -=1;
  }
  else  if(keyCode == LEFT){
            yourChoice = "rock";
            display(rock, 100,100,100,100);
            choiceMade = true;
            amountOfTimesToPlay -=1;
  }
  else if(keyCode == DOWN ){
            yourChoice = "scissors";


            display(scissors, 100,100,100,100);
            choiceMade = true;
            amountOfTimesToPlay -=1;
  }
```

```
}

public void display(PImage imageToDisplay, int center_x, int center_y, int w,int h){
        image(imageToDisplay, center_x, center_y, w, h); //shows the image from the center x and y with
a certain w and h it take image from one of the initial constructors, all of them end up using it, and the
center_x,y,w,h also initialized with it

  }
```