PROGRAM 1

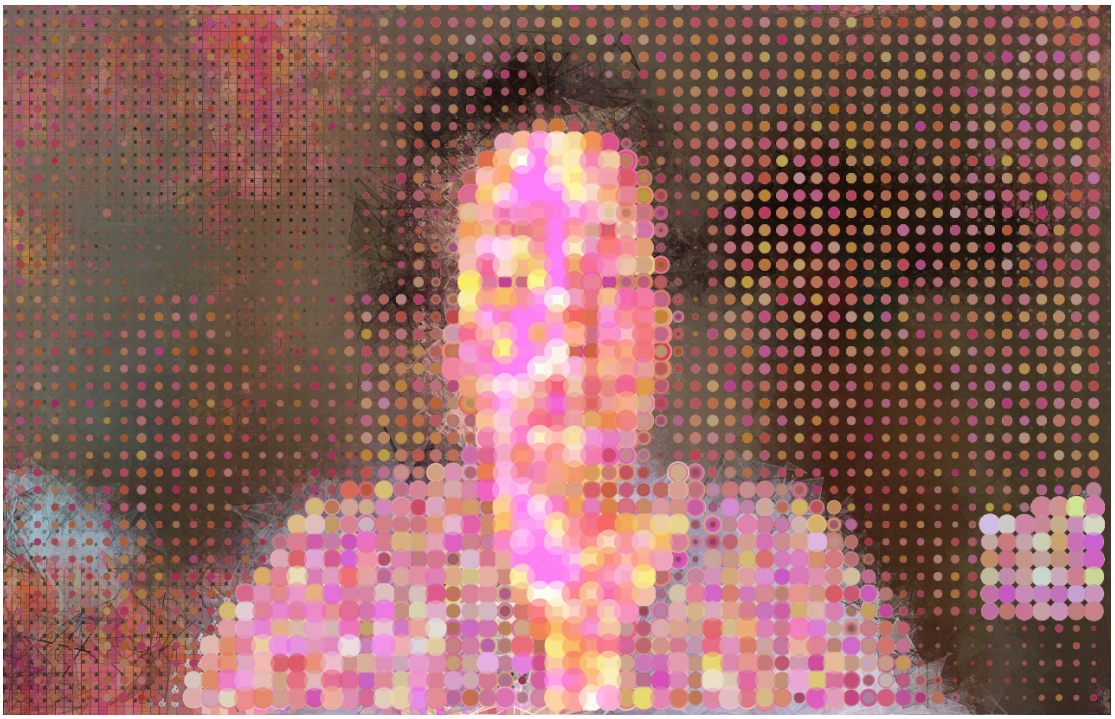
```
// P_4_3_3_02.pde
//
// This program has been changed from a video input that changes it's stroke wieght depending
// on the hue value, saturation and brightness. To a program that produces different filters and artworks depending on
// brightness and hue value.

/**
 * generating a drawing with 4 different modes by analysing live video
 *
 * KEYS
 * 4 - 5 - 6 - 7      : DRAWING MODES >>>> RICHENDA'S INPUT
 * q                  : stop drawing
 * w                  : continue drawing
 * s                  : save png
 * r                  : start record pdf
 * e                  : end record pdf
 *
 */

import processing.pdf.*;
import processing.video.*;
import java.util.Calendar;

boolean savePDF = false;

Capture video;

int pixelIndex;
color c;

float x1, x2, x3, y1, y2, y3;
float pointX = 0;
float pointY = 0;

int counter;
int maxCounter = 100000;

void setup() {
  size(1280, 820);
  video = new Capture(this, width, height, 30);
  video.start();
  x1 = 0;
  y1 = height/2;
  x2 = width/2;
  y2 = 0;
```

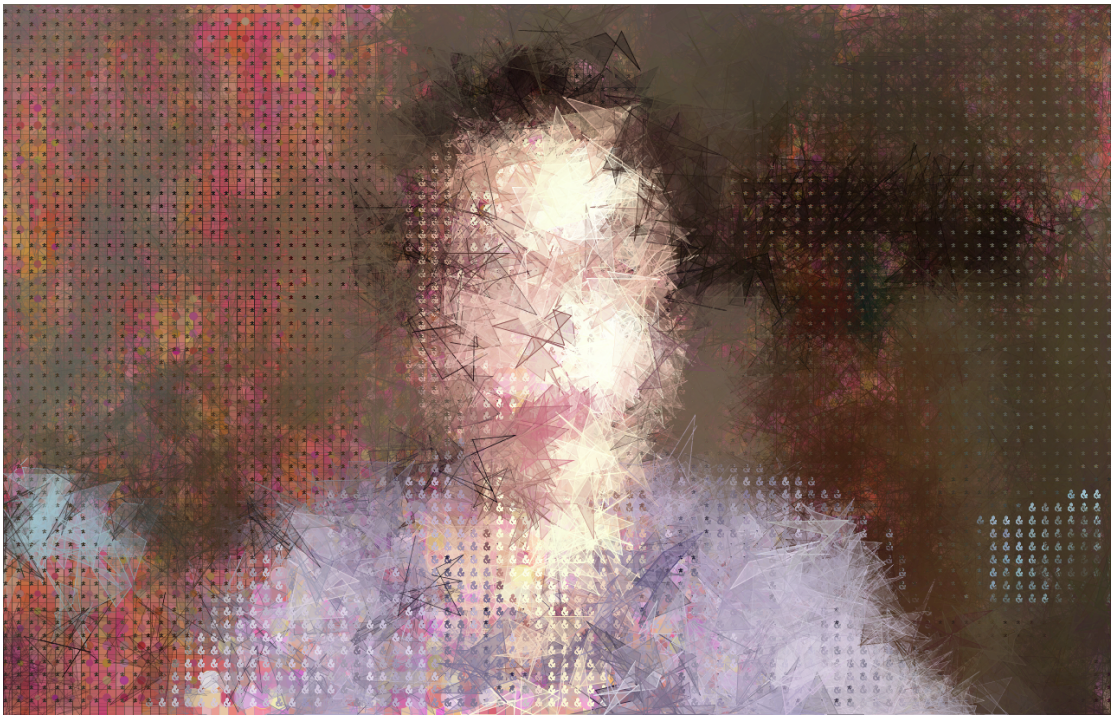
```
  x3 = width;
  y3 = height/2;
  background(255);
}

void draw() {


  colorMode(HSB, 360, 100, 100);
  smooth();
  // get actual web cam image
  if (video.available()) video.read();
  video.loadPixels();

  if (key =='5') {                    //creates circles from video brightness input
    for (int i=0;i<width;i+=20) {
      for (int j=0;j<height;j+=20) {
        c=(video.get(i, j));
        noStroke();
        float bright= brightness (c);
        float sizeC = bright*0.3;
        fill(c/2, bright);
        ellipse(i, j, sizeC, sizeC);
        println(sizeC);
      }
    }
  }
  if (key == '6') {                   //creates rectangles and fills based of hue and
brightness of video
    for (int i=0;i<width;i+=10) {
      for (int j=0;j<height;j+=10) {
        c=(video.get(i, j));
        stroke(c);
        float bright= brightness (c);
        float hueValue = hue(c);
        fill(c/3, bright);
        rect(i, j, hueValue*2, hueValue*2);
        println(hueValue);
      }
    }
  }
  if (key =='7') {                    // creates patterns based on brightness with
ampersands and stars
    for (int i=0;i<width;i+=15) {
      for (int j=0;j<height;j+=15) {
        c=(video.get(i, j));

        float bright= brightness (c);
        float maxBright = 100;
```

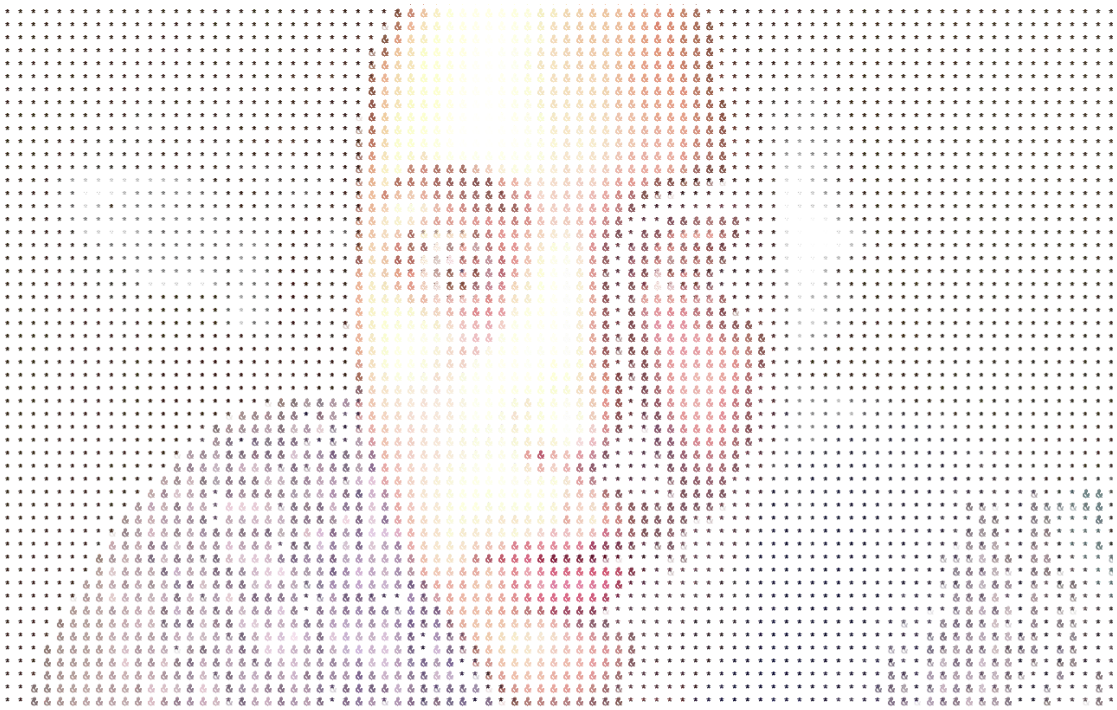
```
      float hueValue = hue(c);
      stroke(c, bright);
      fill(c, bright);
      if (bright>maxBright/2) {
        text('&', i, j);
      }

      else {
        text('*', i, j);
      }

      println(bright);
    }
  }
 }

                        // this is similar to the original program, I changed stroke and
added fill
 if (key == '4') {              // uses the 'dumb agent' to draw random shapes from
video data
   // first line
   pixelIndex = (int) ((video.width-1-int(x1)) + int(y1)*video.width);
   c = video.pixels[pixelIndex];
   // float hueValue = hue(c);
   // float saturationValue = saturation(c);
   float brightnessValue = brightness(c);
   stroke(c);
   fill(c, brightnessValue);

   beginShape();
   vertex(x1, y1);
   vertex(x1, y1);
   for (int i = 0; i < 7; i++) {
    pointX = constrain(x1+random(-50, 50), 0, width-1);
    pointY = constrain(y1+random(-50, 50), 0, height-1);
    vertex(pointX, pointY);
   }
   vertex(pointX, pointY);
   endShape();
   x1 = pointX;
   y1 = pointY;


   // second line
   pixelIndex = (int) ((video.width-1-int(x2)) + int(y2)*video.width);
   c = video.pixels[pixelIndex];
   stroke(c);
   fill(c, brightnessValue);
```

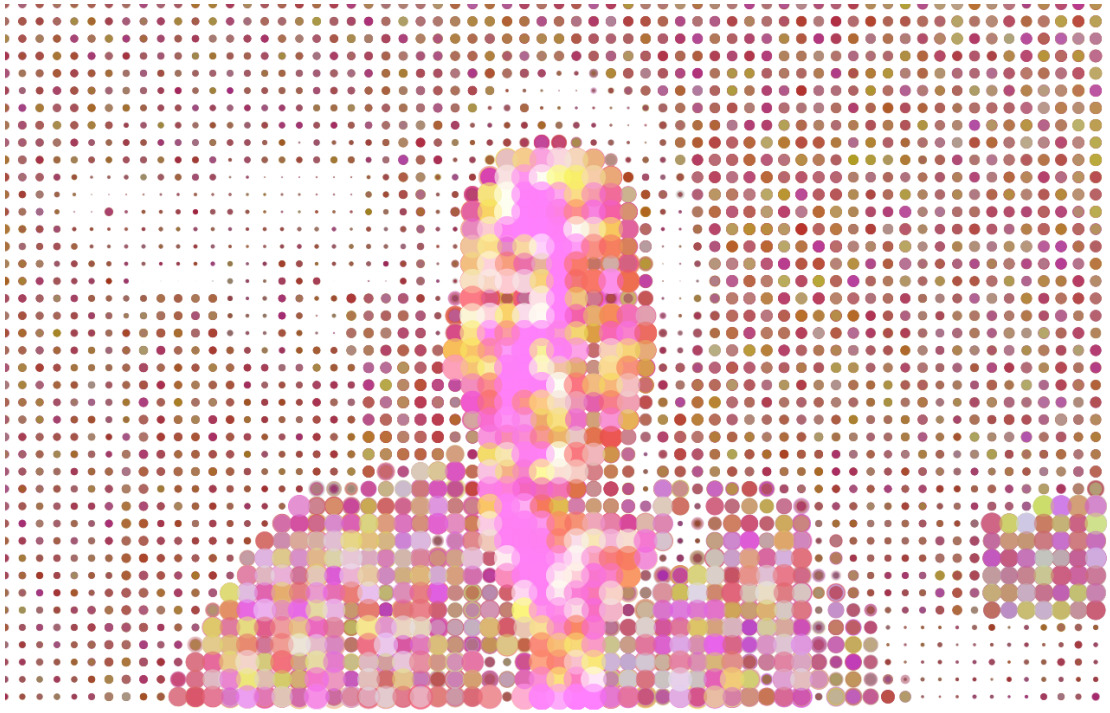
```
   beginShape();
   vertex(x2, y2);
   vertex(x2, y2);
   for (int i = 0; i < 7; i++) {
    pointX = constrain(x2+random(-50, 50), 0, width-1);
    pointY = constrain(y2+random(-50, 50), 0, height-1);
    vertex(pointX, pointY);
   }
   vertex(pointX, pointY);
   endShape();
   x2 = pointX;
   y2 = pointY;


   // third line
   pixelIndex = (int) ((video.width-1-int(x3)) + int(y3)*video.width);
   c = video.pixels[pixelIndex];

   strokeWeight(brightnessValue/100);
   stroke(c);
   fill(c, brightnessValue);

   beginShape();
   vertex(x3, y3);
   vertex(x3, y3);
   for (int i = 0; i < 7; i++) {
    pointX = constrain(x3+random(-50, 50), 0, width-1);
    pointY = constrain(y3+random(-50, 50), 0, height-1);
    vertex(pointX, pointY);
   }
   vertex(pointX, pointY);
   endShape();
   x3 = pointX;
   y3 = pointY;
  }

  counter++;
  if (counter>=maxCounter) noLoop();
}

void keyPressed() {
  switch(key) {
  case BACKSPACE:
   background(360);
   break;
  }
}

void keyReleased() {
```

```
 if (key == 's') {
  saveFrame(timestamp()+"_##.png");
 }
 if (key == 'r') {
  background(360);
  beginRecord(PDF, timestamp()+".pdf");
 }
 if (key == 'e') {
  endRecord();
 }
 if (key == 'q') {
  noLoop();
 }
 if (key == 'w') {
  loop();
 }
}

// timestamp
String timestamp() {
 Calendar now = Calendar.getInstance();
 return String.format("%1$ty%1$tm%1$td_%1$tH%1$tM%1$tS", now);
}



// Generative Gestaltung, ISBN: 978-3-87439-759-9
// First Edition, Hermann Schmidt, Mainz, 2009
// Hartmut Bohnacker, Benedikt Gross, Julia Laub, Claudius Lazzeroni
// Copyright 2009 Hartmut Bohnacker, Benedikt Gross, Julia Laub, Claudius
Lazzeroni
//
// http://www.generative-gestaltung.de
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
// See the License for the specific language governing permissions and
// limitations under the License.
```









PROGRAM 2

```
// P_4_2_2_02.pde
/*
 * Original PROGRAM ---- timelapse camera.
Attached are the programs that I used as a guide as well as some of the original
code that I deleted.

This program has been changed from a timelapse photo camera to an interactive
picture filter program with functioning
buttons(explained bellow in the actions section).

INSPIRATION for these images were taken from QUAYOLA - Strata #3 -{2009} -
Audiovisual Installation
1ch HD projection | 2ch sound
Dimension Variable


 * NEW PROGRAM --- interactive camera
 * ACTIONS ---
 ALL icons                - save image on mouse RELEASE
 click on camera icon     - saves current image displayed
 click on(&hold) pixelate icon  - animates pixelation photo
 click on tint icons      - tints the moving image
 click on(&hold) explode icon   - creates triangles and explodes image
 click on(&hold) triangles icon - creates a moving triangles artwork based on the
colours captured by the camera
 click on(&hold) triangles2icon - creates triangle filter based on the colours
captured by the video
 hold mouse pressed anywhere    - pauses the camera image on the screen
 press 'c'                - clears tints

 */

import processing.video.*;
import java.util.Calendar;

Capture myCam;
Capture video;

// intervalTime in sec. here 5 min
// int intervalTime = 60;                //interval no longer necessary
// int secondsSinceStart = 0;

String startTime = getTimestamp();
int counter = 0;
boolean doSave = true;
PImage snapshot;                         // Added variable 'snapshot' to read
picture from camera
```

```
PImage cameraIcon, pixelateIcon, explode, triangulise, triangulise2; //variables
of the button icons
int tintIconsPos = 150;                    //integer to set position of Tint Icon
function
float pix = 5;                             //integer for pixel size
int t = 15; //triangle size
float n=0;

void setup() {
  size(640, 480);
  println(Capture.list());
  //String s = "Logitech QuickCam Messenger-WDM";
  //myCam = new Capture(this, s, width, height, 30);
  myCam = new Capture(this, width, height, 30);
  myCam.start();
  video = new Capture(this, width, height, 30);
  video.start();
  background(255);
  noStroke();
}

void draw() {
  if (video.available())       //captures video
    video.read();
  video.loadPixels();
  if (myCam.available()&&mousePressed == false) {
    myCam.read();
    image(myCam, 0, 0);

    cameraIcon = loadImage("camera.png");              //assign icon images to
pimage variable
    pixelateIcon = loadImage("pixelate.png");
    explode = loadImage("explode.png");
    triangulise = loadImage("triangulise.png");
    triangulise2 = loadImage("triangulise2.png");

    fill(255, 50);             //draw icons to screen
    rect(20, 20, 58, 58);
    rect(20, 85, 57, 57);
    rect(20, 215, 58, 58);
    image(cameraIcon, 20, 20);
    image(pixelateIcon, 20, 85);
    image(explode, 20, 215);
    image(triangulise, 21, 280);
    image(triangulise2, 51, 280);


    tintIcons(tintIconsPos);    //set and draw position of tint icons function
    snapshot = get();
```

```
 }


 if (mousePressed
&&mouseX>20&&mouseX<80&&mouseY>85&&mouseY<145) {         //
pixelates the image when mouse pressed on pixelate icon
   background(255);
   for ( int i = 0; i < width; i+=200-n) {         // Begin loop for columns
    for ( int j = 0; j < height; j+=200-n) {     // Begin loop for rows
     fill(video.get(i, j));      //assigns colours to pixels
     ellipseMode(CENTER);          //stops the pixels from shifting to a different
position
      ellipse(i, j, pix, pix);       //draws circles as pixels
    }
   }
   if (200-n<5) {                  //makes the pixels animate from out to centre
    n=194;
   }
   n+=2;
   println(200-n);
 }

 if (mousePressed
&&mouseX>20&&mouseX<80&&mouseY>215&&mouseY<270) {         //
explodes the image
   background(255);
   //image(snapshot, 0, 0);
   for ( int i = 0; i < width; i+=5+n) {         // Begin loop for columns
    for ( int j = 0; j < height; j+=5+n) {     // Begin loop for rows
     fill(snapshot.get(i, j));           //assigns colours to pixels
     triangle(i-random(-t, t), j-random(-t, t), i+random(-t, t), j+random(-t, t), i,
j+random(-t, t));
    }
   }
   n+=.5;
   println(10-n);
 }
 if
(mousePressed&&mouseX>20&&mouseX<47&&mouseY>280&&mouseY<340) {
//triangulises the image and creates an artistic filter
   snapshot=get();
   image(snapshot, 0, 0);
   for ( int i = 0; i < width; i+=pix) {         // Begin loop for columns
    for ( int j = 0; j < height; j+=pix) {     // Begin loop for rows
     fill(snapshot.get(i, j));              //assigns colours to pixels
     triangle(i-random(-t, t), j-random(-t, t), i+random(-t, t), j+random(-t, t), i,
j+random(-t, t));
    }
   }
```

```
 }

 if
(mousePressed&&mouseX>51&&mouseX<80&&mouseY>280&&mouseY<340) {
//triangulises video and creates an artistic filter
   for ( int i = 0; i < width; i+=pix) {         // Begin loop for columns
     for ( int j = 0; j < height; j+=pix) {     // Begin loop for rows
       fill(video.get(i, j));                 //assigns colours to pixels
       triangle(i-random(-t, t), j-random(-t, t), i+random(-t, t), j+random(-t, t), i,
j+random(-t, t));
     }
   }
 }

 if (mousePressed
&&mouseX>20&&mouseX<45&&mouseY>tintIconsPos&&mouseY<tintIconsPos
+25) {    // these if statements activate tint when corresponding rectangle icon is
pressed
   tint(#DB67FF, 90);
 }
 if (mousePressed
&&mouseX>52&&mouseX<77&&mouseY>tintIconsPos&&mouseY<tintIconsPos
+25) {
   tint(#D1FF67, 90);
 }
 if (mousePressed
&&mouseX>20&&mouseX<45&&mouseY>tintIconsPos+30&&mouseY<tintIcons
Pos+55) {
   tint(#FFF867, 90);
 }
 if (mousePressed
&&mouseX>52&&mouseX<77&&mouseY>tintIconsPos+30&&mouseY<tintIcons
Pos+55) {
   tint(#67B9FF, 90);
 }
 if (keyPressed && key == 'c') {
   tint(255);
 }
 else if (mousePressed==false) {
   n=0;
 }
}

void    mouseReleased() {                                          // changed the
program so that when you release mouse (over icon) it saves
 if (mouseX>20&&mouseX<80&&mouseY>20&&mouseY<340) {
   String saveFileName = startTime+"-"+nf(counter, 5);
   saveFrame(saveFileName+".png");
   doSave = true;
```

```
   counter++;
  }
  else {
   doSave = false;
  }
}

void pixelate() {                                          // function pixelates the
image
}

void tintIcons (int yPos) {                                        //this function draws
the tint icons in their formation so that it's easier to rearrange icons
  fill(#DB67FF, 80);
  rect(20, yPos, 25, 25);
  fill(#D1FF67, 80);
  rect(52, yPos, 25, 25);
  fill(#FFF867, 80);
  rect(20, yPos+30, 25, 25);
  fill(#67B9FF, 80);
  rect(52, yPos+30, 25, 25);
}

String getTimestamp() {                                            //from orignial
program, sets the date on the name of the file
  Calendar now = Calendar.getInstance();
  return String.format("%1$ty%1$tm%1$td_%1$tH%1$tM%1$tS", now);
}




//changed from        //    secondsSinceStart = millis() / 1000;
//    int interval = secondsSinceStart % intervalTime;
//
//    if (interval == 0 && doSave == true) {
//      String saveFileName = startTime+"-"+nf(counter,5);
//      saveFrame(saveFileName+".png");
//      doSave = false;
//      counter++;
//    }
//    else if (interval != 0) {
//      doSave = true;
//    }
//
//                                    // visualize the time to the next shot
//    fill(random(0,255),random(0,255),random(0,255));
//    rect(map(interval, 0,intervalTime, 0,width),0, 5,5);
```

```
//
// Generative Gestaltung, ISBN: 978-3-87439-759-9
// First Edition, Hermann Schmidt, Mainz, 2009
// Hartmut Bohnacker, Benedikt Gross, Julia Laub, Claudius Lazzeroni
// Copyright 2009 Hartmut Bohnacker, Benedikt Gross, Julia Laub, Claudius
Lazzeroni
//
// http://www.generative-gestaltung.de
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
// See the License for the specific language governing permissions and
// limitations under the License.
```

```
// Used as reference
//Explode by Daniel Shiffman.
//
//Mouse horizontal location controls breaking apart of image and Maps pixels
from a 2D image into 3D space. Pixel brightness controls translation along z axis.
//
//
//// The next line is needed if running in JavaScript Mode with Processing.js
///* @pjs preload="eames.jpg"; */
//
//PImage img;      // The source image
//int cellsize = 2; // Dimensions of each cell in the grid
//int columns, rows;   // Number of columns and rows in our system
//
//void setup() {
//  size(640, 360, P3D);
//  img = loadImage("eames.jpg");  // Load the image
//  columns = img.width / cellsize;  // Calculate # of columns
//  rows = img.height / cellsize;  // Calculate # of rows
//}
//
```

```
//void draw() {
//  background(0);
//  // Begin loop for columns
//  for ( int i = 0; i < columns; i++) {
//    // Begin loop for rows
//    for ( int j = 0; j < rows; j++) {
//      int x = i*cellsize + cellsize/2;     // x position
//      int y = j*cellsize + cellsize/2;     // y position
//      int loc = x + y*img.width;         // Pixel array location
//      color c = img.pixels[loc];  // Grab the color
//                                  / / Calculate a z position as a function of mouseX and
pixel brightness
//      float z = (mouseX / float(width)) * brightness(img.pixels[loc]) - 20.0;
//                                  // Translate to the location, set fill and stroke, and draw
the rect
//      pushMatrix();
//      translate(x + 200, y + 100, z);
//      fill(c, 204);
//      noStroke();
//      rectMode(CENTER);
//      rect(0, 0, cellsize, cellsize);
//      popMatrix();
//    }
//  }
//}




//Used as reference
//POSTERIZE and DOWNSAMPLE IMAGE
//Jeff Thompson
// February 2012
//void pixelateImage(int pxSize) {
//
//  // use ratio of height/width...
//  float ratio;
//  if (width < height) {
//    ratio = height/width;
//  }
//  else {
//    ratio = width/height;
//  }
//
//  // ... to set pixel height
//  int pxH = int(pxSize * ratio);
//
//  noStroke();
//  for (int x=0; x<width; x+=pxSize) {
```

```
//    for (int y=0; y<height; y+=pxH) {
//      fill(p.get(x, y));
//      rect(x, y, pxSize, pxH);
//    }
//  }
//}
```