Production 2

Face_Detection.pbe

```
import processing.video.*;


Capture video;


color trackColor;

float threshold = 15;

float distanceThreshold = 120;

int savedTime;

int totalTime = 300;



ArrayList<Blob> blobs = new ArrayList<Blob>();



void setup () {


  size(640, 480);

savedTime = millis();

  //video = new Capture(this, cameras[3]);

  video = new Capture(this, 640, 480, 30);

  video.start();

  //track red

  trackColor = color(200,136,120);

}


void captureEvent(Capture video) {

  video.read();
```

```
}

void keyPressed() {
  if (key == 'a') {
    distanceThreshold++;
  } else if ( key == 's') {
    distanceThreshold--;
  }
  println(distanceThreshold);
}

void draw () {

   // Calculate how much time has passed
  int passedTime = millis() - savedTime;

  if (passedTime > totalTime) {
    blobs.clear();
    savedTime = millis(); // Save the current time to restart the timer!
  }

  //frameRate(1);
  video.loadPixels();

  image(video, 0, 0);

  //looping through entire image
  for (int x = 0; x < video.width; x++) {
    for (int y = 0; y < video.height; y++) {
      int loc = x + y * video.width;
```

```
color currentColor = video.pixels[loc];
float r1 = red(currentColor);
float g1 = green(currentColor);
float b1 = blue(currentColor);
float r2 = red(trackColor);
float g2 = green(trackColor);
float b2 = blue(trackColor);

//finding color distance
float d = distSq(r1, g1, b1, r2, g2, b2);

//if current color is more similar to tracked color
if (d < threshold*threshold) {

  boolean found = false;
  for (Blob b : blobs) {
    if (b.isNear(x, y)) {
      b.add(x, y);
      found = true;

      break;
    }
  }

  if (!found) {
    Blob b = new Blob(x, y);
    blobs.add(b);
  }
}
```

```
    }

    for (Blob b : blobs) {
      if (b.size() > 500) {
        b.display();
      }
    }
  }
}
float distSq(float x1, float y1, float x2, float y2) {
  float d = (x2-x1)*(x2-x1)+(y2-y1)*(y2-y1);
  return d;
}

float distSq(float x1, float y1, float z1, float x2, float y2, float z2) {
  float d = (x2-x1)*(x2-x1)+(y2-y1)*(y2-y1)+(z2-z1)*(z2-z1);
  return d;
}
```

blobl.pbe

```
class Blob {
  float minx;
  float maxy;
  float maxx;
  float miny;

  Blob (float x, float y) {
    minx = x;
    miny = y;
    maxx = x;
    maxy = y;
```

```
}

void display() {
  //noStroke();
  //hint(DISABLE_OPTIMIZED_STROKE);
  strokeWeight(0);
  rectMode(CORNERS);
  rect(minx, miny, maxx, maxy);
  noFill();

}

float size() {
  return (maxx-minx)*(maxy-miny);
}
void add(float x, float y) {
  minx = min(minx, x);
  miny = min(miny, y);
  maxx = max(maxx, x);
  maxy = max(maxy, y);
}

boolean isNear (float x, float y) {
  float cx = (minx + maxx)/2;
  float cy = (miny + maxy)/2;

  float d = distSq(cx, cy, x, y);

  if (d < distanceThreshold*distanceThreshold) {
    return true;
```

```
    } else {
      return false;
    }
  }
}
```