FINAL_FS_2

```
/**
* Time Displacement
* by David Muth
*
* Keeps a buffer of video frames in memory and displays pixel rows
* taken from consecutive frames distributed over the y-axis
*/

import processing.video.*;

Capture video;
int signal = 0;

//the buffer for storing video frames
ArrayList frames = new ArrayList();

PImage test;

color pixelcolor;

int overlayYPos;

int overlayYResetPos;

PImage videoCanvas;

void setup() {
 fullScreen();

 //size(1200, 700);

 String[] cameras = Capture.list();

 if (cameras.length == 0) {
  println("There are no cameras available for capture.");
  exit();
 } else {
  println("Available cameras:");
  printArray(cameras);
```

```
  // The camera can be initialized directly using an element
  // from the array returned by list():
  video = new Capture(this, cameras[38]);


  // Start capturing the images from the camera
  video.start();
  delay(5000);
  videoCanvas = createImage(video.width, video.height, RGB);
 }


 test = loadImage("code2.png");

 overlayYPos = 0;
 overlayYResetPos = -(test.height - height);
 delay(5000);
}

void captureEvent(Capture camera) {
 camera.read();

 // Copy the current video frame into an image, so it can be stored in the buffer
 PImage img = createImage(video.width, video.height, RGB);
 video.loadPixels();
 arrayCopy(video.pixels, img.pixels);

 frames.add(img);

 // Once there are enough frames, remove the oldest one when adding a new one
 if (frames.size() > height/8) {
   frames.remove(0);
 }
}

void draw() {

 background(0);
```

```
 drawCamera();
 drawOverlay();
}


void drawOverlay() {
 overlayYPos = overlayYPos - 10;

 if (overlayYPos < overlayYResetPos) {
   overlayYPos = 0;
 }


 image(test, 0, overlayYPos);
}

void drawCamera() {
 // Set the image counter to 0



 int currentImage = 0;

 // Begin a loop for displaying pixel rows of 4 pixels height


 videoCanvas.loadPixels();

 for (int y = 0; y < video.height; y+=4) {
   // Go through the frame buffer and pick an image, starting with the oldest one
   if (currentImage < frames.size()) {
     PImage img = (PImage)frames.get(currentImage);

     img.loadPixels();

     /*Put 4 rows of pixels on the screen*/
     for (int x = 0; x < video.width; x++) {
       // save color information for each of the 4 pixels in the pixelcolor variable
       for (int c = 0; c<4; c++) {
```

```
      videoCanvas.pixels[x + (y + c) * videoCanvas.width] = img.pixels[x + (y+c) * img.width];
    }//endfor color info
  }



  // Increase the image counter
  currentImage++;
 } else {
  break;
 }
}

videoCanvas.updatePixels();

image(videoCanvas, 0, 0, width, height);
}
```