midiCV2

```
/////////////////////////// libraries ////////////////////////////
import themidibus.*;
import processing.video.*;
////////////////////////Video variables/////////////////////
Capture video;
int numPixels;
int[] backgroundPixels;
// Last brightest coordinates
int lastX = 0;
int lastY = 0;
//send a 0 via midi
int off = 0;
int exRound = 0;
int lastMes = 0;
int msg2, pitch;
/////////////////////////// MIDI ///////////////////////////////////////
MidiBus bus1;
MidiBus bus2;
boolean play = false;

////////////////////////////// Setup //////////////////////////////////
void setup() {
 frameRate(10);
 size(400, 400);
 background(0);
 // Instantiate the MidiBus
 bus1 = new MidiBus(this, 0, "Bus 1");
 bus2 = new MidiBus(this, 1, "Bus 2");
 //VIDEO STUFF
 // Uses the default video input, see the reference if this causes an error
 video = new Capture(this, width, height);
 noStroke();
 smooth();
 numPixels = video.width * video.height;
 // Create array to store the background image
 backgroundPixels = new int[numPixels];
 // Make the pixels[] array available for direct manipulation
 loadPixels();
 arrayCopy(video.pixels, backgroundPixels);
}

////////////////////////////////// Draw //////////////////////////////////////
void draw() {
 if (video.available()) {
   video.read();
   bright();
 } else {
   bus1.sendNoteOn(1, 0, 127);
   bus2.sendNoteOn(2, 0, 127);
 }
}

////////////////// Toggle playing --[[[ arduino.""Read ]]]-- SWITCH ///////////////////////
void mousePressed() {
 play = !play;
 ////////////////////ON///////////////////////
 if (play==true) {
   video.start();
   video.loadPixels();
   arrayCopy(video.pixels, backgroundPixels);
   ///////////OFF///////////
 } else if (play==false) {
   //take a picture!
   saveFrame("midicv2-######.png");
   video.stop();
 }
}

//////////////////////////////CV/MIDI SECTION////////////////////////////////////////////
void bright() {
 ////////////////////////////////FRAME DIF////////////////////////////////////
 // Make the pixels of video available
 video.loadPixels();
 // Difference between the current frame and the stored background
 int presenceSum = 0;
 // For each pixel in the video frame...
 for (int i = 0; i < numPixels; i++) {
   // Fetch the current color in that location, and also the color of the background in that spot
   color currColor = video.pixels[i];
   color bkgdColor = backgroundPixels[i];
   // Extract the red, green, and blue components of the current pixel&apos;s color
   int currR = (currColor >> 16) & 0xFF;
   int currG = (currColor >> 8) & 0xFF;
   int currB = currColor & 0xFF;
   // Extract the red, green, and blue components of the background pixel&apos;s color
   int bkgdR = (bkgdColor >> 16) & 0xFF;
   int bkgdG = (bkgdColor >> 8) & 0xFF;
   int bkgdB = bkgdColor & 0xFF;
   // Compute the difference of the red, green, and blue values
   int diffR = abs(currR - bkgdR);
   int diffG = abs(currG - bkgdG);
   int diffB = abs(currB - bkgdB);
   // Add these differences to the running tally
   presenceSum += diffR + diffG + diffB;
   // Render the difference image to the screen
   pixels[i] = 0xFF000000 | (diffR << 16) | (diffG << 8) | diffB;
 }
 updatePixels(); // Notify that the pixels[] array has changed
 ///////////////////////////NOTE VALUE -Frame Dif/////////////////////////////////////
 msg2 = int(map(presenceSum, 10000, 60000000, 0, 350));
 //////////////////////////MIDI-MSG if different from previous total//////////////////////
 if (msg2 != lastMes) {
   if (msg2>=60) {
     bus2.sendNoteOn(2, msg2, 127);
     lastMes = msg2;
   }
 } else {
   //tell the VCA to not let anything through
   bus2.sendNoteOn(2, 0, 127);
 }
 ////////////////////////////BrightnessTracker////////////////////////////
 //START WITH BRIGHTEST PIXEL TRACKKING TO GET CV MESSAGE for OSC CV/VCO
Gate
 int brightestX = 0; // X-coordinate of the brightest video pixel
 int brightestY = 0; // Y-coordinate of the brightest video pixel
 float brightestValue = 0; // Brightness of the brightest video pixel
 int index = 0;
 /////////////////////////////////Brightness tracker///////////////////////////////////////////
 for (int y = 0; y < video.height; y++) {
   for (int x = 0; x < video.width; x++) {
     // Get the color stored in the pixel
     int pixelValue = video.pixels[index];
     // Determine the brightness of the pixel
     float pixelBrightness = brightness(pixelValue);
     // If that value is brighter than any previous, then store the brightness of that pixel, as well as its
(x,y) location
     if (pixelBrightness > brightestValue) {
       brightestValue = pixelBrightness;
       brightestY = y;
       brightestX = x;
     }
     //update index int
     index++;
   }
 }
 // Determine the MIDI notePitch based brightestPixel position
 float p = 60*random(0.9, 1.5);
 ///////////////////////////MIDI NOTES FOR X AND Y? SEPERATE BUSSES & SYNTH
VOICES&&FX//////////////////////////
 int x = int (p + (float(brightestX) / width));
 int y = int (p + (float(brightestY) / height));
 pitch = int(p + x + y);
 //Send MIDI to Bus1
 if ((brightestX != lastX || brightestY != lastY) && play) {
   int rnd = int(random(0, 3));
   if (rnd > 1) {
     ////////MIDI HERE -- Pick one or two notes//////////
     bus1.sendNoteOn(1, pitch, 127);
     println("one");
   } else {
     bus1.sendNoteOn(1, int(pitch*0.89), 127);
     bus1.sendNoteOn(1, int(pitch*1.15), 127);
     println("two");
   }
 } else {
   //tell VCV to not let anything through --
   bus1.sendNoteOn(1, 0, 127);
 }
 //housekeeping for brightestTracking
 lastX = brightestX;
 lastY = brightestY;
 // Delay .1 seconds to prevent madness
 delay(100);
}
```