

```

#include "ofMain.h"

class testApp : public ofApp{

public:
    void setup();
    void update();
    void draw();

    void keyPressed(int key);
    void keyReleased(int key);
    void mouseMoved(int x, int y );
    void mouseDragged(int x, int y, int button);
    void mousePressed(int x, int y, int button);
    void mouseReleased();

    ofVideoGrabber    vidGrabber;
    unsigned char *   videoInverted;
    ofTexture         videoTexture;
    int               camWidth;
    int               camHeight;
};

float goalRed = 200;
float goalGreen = 200;
float goalBlue = 200;

float distThreshold = 30;

//-----
void testApp::setup(){
    camWidth    = 800;
    camHeight   = 600;

    vidGrabber.setVerbose(true);
    vidGrabber.initGrabber(camWidth,camHeight);
    videoInverted = new unsigned char [camWidth*camHeight*3];
    videoTexture.allocate(camWidth,camHeight, GL_RGB);
}

//-----
void testApp::update(){
    vidGrabber.grabFrame();

    if (vidGrabber.isFrameNew()){
        int totalPixels = camWidth*camHeight*3;
        unsigned char * pixels = vidGrabber.getPixels();
        for (int i = 0; i < totalPixels; i+=3){
            float r = pixels[i];
            float g = pixels[i+1];
            float b = pixels[i+2];
            float dr = r - goalRed;
            float dg = g - goalGreen;
            float db = b - goalBlue;
            float distance = sqrt(dr*dr + dg*dg + db*db);
            if (distance < distThreshold){
                videoInverted[i] = 255;
                videoInverted[i+1] = 0;
                videoInverted[i+2] = 0;
            } else {
                videoInverted[i] = pixels[i];
                videoInverted[i+1] = pixels[i+1];
                videoInverted[i+2] = pixels[i+2];
            }
        }
        videoTexture.loadData(videoInverted, camWidth,camHeight, GL_RGB);
    }
}

//-----
void testApp::draw(){
    videoTexture.draw(0,0,camWidth,camHeight);
}

//-----
void testApp::mousePressed(int x, int y, int button){
    unsigned char * pixels = vidGrabber.getPixels();
    goalRed = pixels[(y*camWidth+x)*3];
    goalGreen = pixels[(y*camWidth+x)*3+1];
    goalBlue = pixels[(y*camWidth+x)*3+2];
}

```

```

package wk10;

import processing.core.PApplet;
import processing.video.Capture;

public class TrackPixel extends PApplet {

    static public void main(String _args[] ) {
        PApplet.main(new String[] { "wk10.TrackPixel" });
    }

    Capture video;//regular processing library

    float goalRed = 17;
    float goalBlue = 123;
    float goalGreen = 184;

    float distThreshold = 10;

//-----
    public void setup() {
        int width = 800;
        int height = 600;

        size(width, height);
        video = new Capture(this, 800, 600);
    }

//-----
    public void draw() {
        if (video.available()) {
            video.read();

            for(int row = 0; row < video.height; row++){
                for(int col = 0; col < video.width; col++){
                    int offset = row*width + col;
                    int thisPixel = video.pixels[offset];
                    float r = red(thisPixel);
                    float g = green(thisPixel);
                    float b = blue(thisPixel);
                    float dr = r - goalRed;
                    float dg = g - goalBlue;
                    float db = b - goalGreen;
                    float distance = sqrt(dr*dr + dg*dg + db*db);
                    if (distance < distThreshold ){
                        video.pixels[offset] = color(255,255,255);
                    }
                }
            }
            image(video,0,0);
        }
    }

//-----
    public void mousePressed(){
        int thisPixel = video.pixels[mouseY* width + mouseX];
        goalRed = red(thisPixel);
        goalBlue = blue(thisPixel);
        goalGreen = green(thisPixel);
    }
}

```

```

#include "ofMain.h"

class testApp : public ofSimpleApp{

    public:
        void setup();
        void update();
        void draw();

        void keyPressed(int key);
        void keyReleased(int key);
        void mouseMoved(int x, int y );
        void mouseDragged(int x, int y, int button);
        void mousePressed(int x, int y, int button);
        void mouseReleased();

        ofVideoGrabber    vidGrabber;
        unsigned char *    videoInverted;
        ofTexture          videoTexture;
        int                camWidth;
        int                camHeight;

};

float goalRed = 200;
float goalGreen = 200;
float goalBlue = 200;

float distThreshold = 30;

//-----
void testApp::setup(){
    camWidth      = 800;
    camHeight     = 600;

    vidGrabber.setVerbose(true);
    vidGrabber.initGrabber(camWidth,camHeight);
    videoInverted = new unsigned char[camWidth*camHeight*3];
    videoTexture.allocate(camWidth,camHeight, GL_RGB);
}

//-----
void testApp::update(){
    vidGrabber.grabFrame();

    if (vidGrabber.isFrameNew()){
        int totalPixels = camWidth*camHeight*3;

```

```

package wk10;

import processing.core.PApplet;
import processing.video.Capture;

public class TrackPixel extends PApplet {

    static public void main(String _args[] ) {
        PApplet.main(new String[] { "wk10.TrackPixel" });
    }

    Capture video;//regular processing library

    float goalRed = 17;
    float goalBlue = 123;
    float goalGreen = 184;

    float distThreshold = 10;

//-----
    public void setup() {
        int width = 800;
        int height = 600;

        size(width, height);
        video = new Capture(this, 800, 600);
    }

//-----
    public void draw() {
        if (video.available()) {
            video.read();

            for(int row = 0; row < video.height; row++){

```

```

//-----
void testApp::setup(){
    camWidth      = 800;
    camHeight     = 600;

    vidGrabber.setVerbose(true);
    vidGrabber.initGrabber(camWidth,camHeight);
    videoInverted = new unsigned char [camWidth*camHeight*3];
    videoTexture.allocate(camWidth,camHeight, GL_RGB);
}

//-----
void testApp::update(){
    vidGrabber.grabFrame();

    if (vidGrabber.isFrameNew()){
        int totalPixels = camWidth*camHeight*3;
        unsigned char * pixels = vidGrabber.getPixels();
        for (int i = 0; i < totalPixels; i+=3){
            float r = pixels[i];
            float g = pixels[i+1];
            float b = pixels[i+2];
            float dr = r - goalRed;
            float dg = g - goalGreen;
            float db = b - goalBlue;
            float distance = sqrt(dr*dr + dg*dg + db*db);
            if (distance < distThreshold){
                videoInverted[i] = 255;
                videoInverted[i+1] = 0;
                videoInverted[i+2] = 0;
            } else {
                videoInverted[i] = pixels[i];
                videoInverted[i+1] = pixels[i+1];
                videoInverted[i+2] = pixels[i+2];
            }
        }
        videoTexture.loadData(videoInverted, camWidth,camHeight, GL_RGB);
    }
}

//-----
void testApp::draw(){
    videoTexture.draw(0,0,camWidth,camHeight);
}

//-----
void testApp::mousePressed(int x, int y, int button){

```

```

//-----
public void setup() {
    int width = 800;
    int height = 600;

    size(width, height);
    video = new Capture(this, 800, 600);
}

//-----
public void draw() {
    if (video.available()) {
        video.read();

        for(int row = 0; row < video.height; row++){
            for(int col = 0; col < video.width; col++){
                int offset = row*width + col;
                int thisPixel = video.pixels[offset];
                float r = red(thisPixel);
                float g = green(thisPixel);
                float b = blue(thisPixel);
                float dr = r - goalRed;
                float dg = g - goalBlue;
                float db = b - goalGreen;
                float distance = sqrt(dr*dr + dg*dg + db*db);
                if (distance < distThreshold ){
                    video.pixels[offset] = color(255,255,255);
                }
            }
        }
        image(video,0,0);
    }
}

//-----
public void mousePressed(){

```

```

}

//-----
void testApp::update(){
    vidGrabber.grabFrame();

    if (vidGrabber.isFrameNew()){
        int totalPixels = camWidth*camHeight*3;
        unsigned char * pixels = vidGrabber.getPixels();
        for (int i = 0; i < totalPixels; i+=3){
            float r = pixels[i];
            float g = pixels[i+1];
            float b = pixels[i+2];
            float dr = r - goalRed;
            float dg = g - goalGreen;
            float db = b - goalBlue;
            float distance = sqrt(dr*dr + dg*dg + db*db);
            if (distance < distThreshold){
                videoInverted[i] = 255;
                videoInverted[i+1] = 0;
                videoInverted[i+2] = 0;
            } else {
                videoInverted[i] = pixels[i];
                videoInverted[i+1] = pixels[i+1];
                videoInverted[i+2] = pixels[i+2];
            }
        }
        videoTexture.loadData(videoInverted, camWidth, camHeight, GL_RGB)
    }
}
}

```

```

//-----
void testApp::draw(){
    videoTexture.draw(0,0,camWidth,camHeight);
}

```

```

//-----
void testApp::mousePressed(int x, int y, int button){
    unsigned char * pixels = vidGrabber.getPixels();
    goalRed = pixels[(y*camWidth+x)*3];
    goalGreen = pixels[(y*camWidth+x)*3+1];
    goalBlue = pixels[(y*camWidth+x)*3+2];
}

```

```

//-----
public void draw() {
    if (video.available()) {
        video.read();

        for(int row = 0; row < video.height; row++){
            for(int col = 0; col < video.width; col++){
                int offset = row*width + col;
                int thisPixel = video.pixels[offset];
                float r = red(thisPixel);
                float g = green(thisPixel);
                float b = blue(thisPixel);
                float dr = r - goalRed;
                float dg = g - goalBlue;
                float db = b - goalGreen;
                float distance = sqrt(dr*dr + dg*dg + db*db);
                if (distance < distThreshold ){
                    video.pixels[offset] = color(255,255,255);
                }
            }
        }
        image(video,0,0);
    }
}
}

```

```

//-----
public void mousePressed(){
    int thisPixel = video.pixels[mouseY* width + mouseX];
    goalRed = red(thisPixel);
    goalBlue = blue(thisPixel);
    goalGreen = green(thisPixel);
}
}

```

```
}  
  
//-----  
void testApp::mousePressed(int x, int y, int button){  
    unsigned char * pixels = vidGrabber.getPixels();  
    goalRed = pixels[(y*camWidth+x)*3];  
    goalGreen = pixels[(y*camWidth+x)*3+1];  
    goalBlue = pixels[(y*camWidth+x)*3+2];  
}
```

```
//-----  
public void mousePressed(){  
    int thisPixel = video.pixels[mouseY* width + mouseX];  
    goalRed = red(thisPixel);  
    goalBlue = blue(thisPixel);  
    goalGreen = green(thisPixel);  
}  
}
```