# Project code information

Code information in Openframework.

ofApp.h

```
#pragma once

#include "ofMain.h"
#include "ofxOsc.h"
#include "ofxGui.h"

// listen for OSC on port 12000
#define PORT 12000

class ofApp : public ofBaseApp{

    public:
         int judge();

        void setup();
        void update();
        void draw();
         void stripePattern();

         void brush1();
         void brush2();
         void brush3();

         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);
```

```
29          void mouseReleased(int x, int y, int button);
30          void mouseEntered(int x, int y);
31          void mouseExited(int x, int y);
32          void windowResized(int w, int h);
33          void dragEvent(ofDragInfo dragInfo);
34          void gotMessage(ofMessage msg);
35
36         void exit();
37
38         ofImage image;
39         ofColor color;
40
41         ofFbo fbo;
42         int width = 900;
43         int height = 600;
44
45         ofPolyline polyline;
46
47         ofxPanel gui;  // GUI panel declared
48         ofxIntSlider countX; // The countX slider is declared
49         ofxFloatSlider stepX; // Floating point value declaration
50         ofxFloatSlider twistX; // twistX slider
51
52         ofxGuiGroup globalGroup;
53         ofxFloatSlider Scale;
54         ofxFloatSlider Rotate;
55         ofxFloatSlider Background;
56
57         ofxIntSlider size;
58         ofxColorSlider colors;
59         ofxVec3Slider backgrounds;
60         ofxButton btnClear;
61
62
63         ofParameter<int> scale;
64         ofParameter<ofColor> colorr;
65
66         ofxToggle filled, type;
67
```

```
68          //For OSC receiving:
69          ofxOscReceiver receiver;
70
71          float oxcx = 0.0;
72          float oxcy = 0.0;
73          float f = stepX;
74
75  };
```

ofApp.cpp

```
1   #include "ofApp.h"
2
3   //--------------------------------------------------------------
4
5   void ofApp::setup(){
6
7       // Set global properties, project window title, screen size,
8       // rendering frame rate, background color.
9       ofSetWindowTitle( "photo page" );
10      ofSetWindowShape( 1280, 720 );
11      ofSetFrameRate( 60 );
12      // //ofBackground( ofColor::white );
13
14      color.r=90;
15      color.g=120;
16      color.b=160;
17      //ofSetBackgroundColor(Background);
18      image.loadImage("body.jpg");
19      ofSetColor(ofRandom(1,0));
20
21      gui.setup();
22
23      gui.setup( "Parameters", "settings.xml" );
24      gui.add( countX.setup( "countX", 0, 0, 10 ) );
25      gui.add( stepX.setup( "stepX", 20, 0, 100 ) );
```

```
26      gui.add( twistX.setup( "twistX", 5, -5, 45 ) );
27
28      gui.add( Scale.setup( "Scale", 10, 0.0, 10 ) );
29    //gui.add( Rotate.setup( "Rotate", 0, -180, 180 ) );
30    //gui.add( Background.setup("Background",255,0,255));
31
32    //gui.setup();
33      gui.add(size.setup("size",1,1,8));
34      gui.add(colors.setup("colors",ofColor(255,255,255,255),ofColor(0
35    //gui.add( Background.setup("Background",255,255,255));
36      gui.add(backgrounds.setup("backgrounds",ofVec3f(0,0,0),ofVec3f(0
37      gui.add(btnClear.setup("clear"));
38
39      fbo.allocate(width,height);
40      fbo.begin();
41      ofClear(255);
42      fbo.end();
43
44    //gui.loadFromFile( "settings.xml" );
45
46      //Start listening for OSC messages
47      receiver.setup(PORT);
48
49
50  }
51
52  //--------------------------------------------------------------
53  void ofApp::update(){
54
55      ofBackground(backgrounds->x, backgrounds->y, backgrounds->z);
56      if(btnClear){
57          fbo.begin();
58          ofClear(255);
59          fbo.end();
60      }
61
62      // Reference: VERY simple colour change (just adjusts hue)-
63      // http://www.wekinator.org/examples/
64
```

```
65      //Receive any incoming OSC messages
66          while(receiver.hasWaitingMessages()){
67          // get the next message
68          ofxOscMessage m;
69          receiver.getNextMessage(&m);
70
71          // If it's the message we're expecting from Wekinator:
72          if(m.getAddress() == "/wek/outputs"){
73              countX = m.getArgAsFloat(0)*10;
74              stepX = m.getArgAsFloat(1)*200;
75              twistX = m.getArgAsFloat(2)*45;
76              Scale = m.getArgAsFloat(3)*10;
77              size = m.getArgAsFloat(4)*8;
78
79
80          } else{
81              // unrecognized message: display on the bottom of the sc
82              string msgString;
83              msgString = m.getAddress();
84              msgString += ":";
85              for(size_t i = 0; i < m.getNumArgs(); i++){
86
87                  // get the argument type
88                  msgString += " ";
89                  msgString += m.getArgTypeName(i);
90                  msgString += ":";
91
92                  // display the argument - make sure we get the right
93                  if(m.getArgType(i) == OFXOSC_TYPE_INT32){
94                      msgString += ofToString(m.getArgAsInt32(i));
95                  }
96                  else if(m.getArgType(i) == OFXOSC_TYPE_FLOAT){
97                      msgString += ofToString(m.getArgAsFloat(i));
98                  }
99                  else if(m.getArgType(i) == OFXOSC_TYPE_STRING){
100                     msgString += m.getArgAsString(i);
101                 }
102                 else{
103                     msgString += "unhandled argument type " + m.getA
```

```
                }
            }

            cout << "Unexpected message: " << msgString << endl;
        }

    }

}

//--------------------------------------------------------------

void ofApp::stripePattern() {
  ofSetColor( ofColor::yellow );
    ofSetLineWidth( 3.0 );
    ofNoFill();
  for (int i=-countX; i<countX; i++) {
    ofPushMatrix();
    ofTranslate( i * stepX, 20 );
    ofRotate( i * twistX );
    //ofLine( 0, -100, 0, 100 );
    ofScale( Scale ,Scale);
    ofTriangle( 0, 0, -4, 10, 4, 10);
    ofPopMatrix();
  }
}


// This part attempts to obtain the real-time motion trajectory of t
// through Wekinator and map it into the program in the form of brus
// The final presentation form is to draw the moving position of the
// 3 different styles of brushes.

// Reference:
// https://openframeworks.cc/ofBook/chapters/intro_to_graphics.html

void ofApp::brush1(){
    int radiusSize = 2;
    int maxOffDistance = 1;
```

```
143     for (int radius = size; radius>0;radius-= radiusSize){
144         float angle = ofRandom(360.0);
145         float distance = ofRandom(maxOffDistance);
146         float xOffset = cos(angle) * distance;
147         float yOffset = sin(angle) * distance;
148 
149         ofColor firstcolor(0,225,60,30);
150         ofColor colorss(color);
151         ofColor inBetween = firstcolor.getLerped(colorss,ofRandom(1,
152         ofSetColor(inBetween);
153         ofDrawCircle(ofGetMouseX() + xOffset,ofGetMouseY() + yOffset
154     }//spots
155 
156 }
157 
158 // Reference:
159 // https://openframeworks.cc/ofBook/chapters/intro_to_graphics.html
160 
161 void ofApp::brush2(){
162     int numTriangles = 4;
163     int minOffset = 5;
164     int maxOffset = 10;
165     int alpha = 150;
166 
167     for(int t = 0; t < numTriangles; ++t){
168         float offsetDistance = ofRandom(minOffset,maxOffset);
169 
170         ofVec2f mousePos(ofGetMouseX(),ofGetMouseY());
171         ofVec2f p1(100,8);
172         ofVec2f p2(200,0);
173         ofVec2f p3(100,-8);
174 
175         float rotation = ofRandom(360);
176         p1.rotate(rotation);
177         p2.rotate(rotation);
178         p3.rotate(rotation);
179 
180         ofVec2f triangleOffset(offsetDistance,0.0);
181         triangleOffset.rotate(rotation);
```

```
182
183         p1 += mousePos + triangleOffset;
184         p2 += mousePos + triangleOffset;
185         p3 += mousePos + triangleOffset;
186
187         ofColor init(0,252,255,alpha);
188         ofColor colors(color);
189         ofColor inbetween = init.getLerped(colors,ofRandom(1.0));
190         ofSetColor(inbetween);
191         ofDrawTriangle(p1,p2,p3);
192     }//triangles
193 }
194
195 void ofApp::brush3(){
196
197     if (ofGetMousePressed(OF_MOUSE_BUTTON_LEFT))
198     {
199         ofSetColor(color);
200         ofSetLineWidth(1);
201         ofDrawLine(mouseX, mouseY, mouseX + ofSignedNoise(ofGetElaps
202     }
203 }
204
205  //fbo.end();
206
207 //--------------------------------------------------------------
208
209 void ofApp::draw(){
210     ofPushMatrix();
211
212
213     //ofBackground( Background );
214     //image.draw(ofGetWidth()/2 - image.getWidth()/2,ofGetHeight()/2
215     //image.resize(600,400);
216     //ofSetColor(color);
217
218     image.draw(ofGetWidth()/2 -image.getWidth()/2 ,ofGetHeight()/2 -
219     image.resize(900,600);
220     ofSetColor(colors);
```

```
221
222     //color.setSaturation(35);
223     //color.setBrightness(200);
224     //color.setHue(100);
225
226     //float Scl = pow( Scale, 4.0f );
227     //ofScale( Scl, Scl );
228     //ofRotate( Rotate);
229
230     fbo.draw(190,60);
231     ofNoFill();
232     ofDrawRectangle(190,60,width,height);
233     ofSetColor(color);
234
235     gui.draw();
236
237     stringstream ss;
238     ss << "Receives 4 outputs from Wekinator: countX,stepX,twistX,Sc
239     ss << "Each of these values is expected in the range 0-1."<<endl
240     ofDrawBitmapString(ss.str().c_str(), 20, 20);
241
242
243       ofTranslate( ofGetWidth()/2 , ofGetHeight()/2  );
244       ofSetColor(color);
245       stripePattern();
246       ofPopMatrix();
247
248 }
249
250 void ofApp::exit() {
251   gui.saveToFile( "settings.xml" );
252 }
253
254 //--------------------------------------------------------------
255 void ofApp::keyPressed(int key){
256
257 }
258
259 //--------------------------------------------------------------
```

```
260 void ofApp::keyReleased(int key){
261
262 }
263
264 //--------------------------------------------------------------
265 void ofApp::mouseMoved(int x, int y ){
266
267 }
268
269 //--------------------------------------------------------------
270 void ofApp::mouseDragged(int x, int y, int paint){
271
272     fbo.begin();
273     ofSetLineWidth(size);
274
275     //ofDrawCircle(mouseX,mouseY,size);
276
277     brush1();
278     //ofSetColor(color);
279     brush2();
280     ofSetColor(color);
281     brush3();
282     ofSetColor(color);
283     polyline.addVertex(ofPoint(x,y));
284     polyline.draw();
285
286     fbo.end();
287
288 }
289
290 //--------------------------------------------------------------
291 void ofApp::mousePressed(int x, int y, int button){
292
293     fbo.begin();
294     polyline.clear();
295     fbo.end();
296
297 }
298
```

```
299 //--------------------------------------------------------------
300 void ofApp::mouseReleased(int x, int y, int button){
301
302 }
303
304 //--------------------------------------------------------------
305 void ofApp::mouseEntered(int x, int y){
306
307 }
308
309 //--------------------------------------------------------------
310 void ofApp::mouseExited(int x, int y){
311
312 }
313
314 //--------------------------------------------------------------
315 void ofApp::windowResized(int w, int h){
316
317 }
318
319 //--------------------------------------------------------------
320 void ofApp::gotMessage(ofMessage msg){
321
322 }
323
324 //--------------------------------------------------------------
325 void ofApp::dragEvent(ofDragInfo dragInfo){
326
327 }
```