```
//========================================================================
/*....sistemAutopoietico Autonomo_04..............
por...Felix Lazo...www.10sistemasautopoieticos.cl...... www.lazo.cl......*/

//=========== main.cpp ============

#include "ofMain.h"
#include "testApp.h"

int main( ){

        ofSetupOpenGL(1024,768, OF_FULLSCREEN);                 // <-------- setup the GL context

        // this kicks off the running of my app
        // can be OF_WINDOW or OF_FULLSCREEN
        // pass in width and height too:
        ofRunApp( new testApp());

}

//=========== testApp.h ============
#pragma once
#include "ofMain.h"

class testApp : public ofBaseApp{
public:
        void setup();
        void update();
        void draw();


        void audioReceived( float * input, int bufferSize, int nChannels );
        void audioOut( float * input, int bufferSize, int nChannels );

        ofSoundStream soundStream;

        vector<float> buf, buffer;

        void keyPressed(int key);
        void keyReleased(int key);
        void mouseMoved(int x, int y);
        void mouseDragged(int x, int y, int button);
        void mouseReleased(int x, int y, int button);
        void mousePressed(int x, int y, int button);
        void windowResized(int w, int h);
        void dragEvent(ofDragInfo dragInfo);
        void gotMessage(ofMessage msg);

        float userFreq;
        float userPwm;
        float freq;
        float pwm;
        float phase;
        float audioin;
        float colorHue;
};

//=========== testApp.cpp ============

#include "testApp.h"

int bufSize = 512;
int sampleRate = 44100;
float volume = 0.1;
const float duration = 0.25;
const int N = duration * sampleRate;
```

```
//--------------------------------------------------------------
void testApp::setup(){

ofBackground(160,160,160);
ofSetFrameRate(30);
ofEnableSmoothing();
ofHideCursor();

        userFreq = 100.0;
        userPwm = 0.5;
        soundStream.listDevices();
//for Raspberry-Pi usb soundcard soundStream.setDeviceID(1);
        soundStream.setDeviceID(0);
        freq = userFreq;
        pwm = userPwm;
        phase = 0;
        buf.resize( bufSize );
        buffer.resize( N, 0.0 );

        soundStream.setup( this, 2, 1, sampleRate, bufSize, 4 );

}

//--------------------------------------------------------------
void testApp::update(){

float time = ofGetElapsedTimef();
float reds = sin(time*0.3);

//colorHue = ofMap(reds, -1.0, 1,0, 0.0, 255.0);
//colorHue += .1f;
  //  if(colorHue >= 255) colorHue = 0.f;

float mic = ofMap( audioin, -1, 1, 0.0, 1.0 );
float pitch = ofMap(sin(time*0.003), -1.0, 1.0, 150, 900);
//AudioControl
if(audioin > 0.9){
        ofSoundStopAll();
        //userFreq = 0.0;
        //userPwm = 0;
        }
if(audioin > 0.5 || audioin < 0.9 ){
        //userFreq = pitch*mic;
        //userFreq = ofMap( mic, 0.0, 1.0, 1800, 19000 );
        userFreq = ofMap( mic, 0.0, 1.0, 700, 190000 );
        //userPwm = 1.0;
        userPwm = ABS(sin(time*0.003))*1.05;
        }
if(audioin > 0.3 || audioin < 0.5 ){
        //ofSoundStopAll();
        //userFreq = ofMap( audioin*0.6, -1, 1, 180, 1900 );
        //userFreq = pitch*0.5;
        //userPwm = ofMap( audioin, -1, 1, 0.3, 0.9 );
        //userPwm = 0.3;
        }
if(audioin < 0.3){
        //userFreq = 80*mic*0.0005;
        //userPwm =  0.96*mic ;
ofSoundStopAll();
        }
}

//--------------------------------------------------------------
void testApp::draw(){
float hue = ofMap(audioin, -1,1, 120, 255);
float sat = ofMap(audioin, -1,1, 20, 225);
float time = ofGetElapsedTimef();
float reds = sin(time*0.03)*audioin;

colorHue = ofMap(reds, -1.0, 1,0, 120.0, 255.0);

ofColor c;
//c = ofColor::fromHsb(hue, sat*1.4, 250, 90);
c = ofColor::fromHsb(colorHue, 200+audioin, 220, 90);
ofPushMatrix();
ofTranslate(35,34);
ofScale(1.32, 0.72);
//ofScale(0, 1.6);
        ofFill();
ofSetLineWidth(4);
        //ofSetColor(170,90,70, 90);
ofSetColor(c);

        for(int x = 0; x < 1000; x += 20){
                for(int y = 0; y < 1000; y += 20){
                        //ofCircle(x, y, sin((x + y) / 100.0f + ofGetElapsedTimef()) * 5.0f);
                        float radius = sin((x+(y*7.0f)) /100.0f + ofGetElapsedTimef()) * 5.0f;

                ofCircle(x, y, radius*(3.3+audioin));


                }
        }

ofPopMatrix();



}

//--------------------------------------------------------------
//Audio input
void testApp::audioReceived(
        float * input, int bufferSize, int nChannels )
{

                for (int i=0; i<bufferSize; i++) {

                audioin =  input[i];
                }

}

//--------------------------------------------------------------
void testApp::audioOut( float * output, int bufferSize, int nChannels ){

        for (int i=0; i<bufferSize; i++) {

                freq += ( userFreq - freq ) * 0.001;
                pwm += ( userPwm - pwm ) * 0.001;
                phase += freq / sampleRate;
                phase = fmodf( phase, 1.0 );

                float v = ( phase < pwm ) ? 1.0 : -1.0;
                output[ i*2 ] = output[ i*2 + 1 ] = v * volume;

                if ( i < bufSize ) {
                        buf[ i ] = v;
                }
        }
}

//--------------------------------------------------------------
```

```
void testApp::keyPressed(int key){

}

//--------------------------------------------------------------
void testApp::keyReleased(int key){

}

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

}

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

}

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

}

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

}

//--------------------------------------------------------------
void testApp::windowResized(int w, int h){

}

//--------------------------------------------------------------
void testApp::gotMessage(ofMessage msg){

}

//--------------------------------------------------------------
void testApp::dragEvent(ofDragInfo dragInfo){

}
```