```
/*
AUDITUM
 26 March 2010
 created by Nicola Vittori based on Minim examples code,
 Special thanks to Tom Hulbert and Durrel Bishop for his precious help programming Auditum.
 IUAV IxD Lab2 2009/2010
 Philip Tabor with Gillian Crampton Smith
 */

//import the library for audio and serial communication
import processing.serial.*;
import ddf.minim.*;

Minim minim;
AudioInput in;
AudioPlayer player;
Serial myPort;
char lf = char(10);
char mode = 0;

//declare the variables for the volume of the choir and the fade volume period
int gainValue;
int fadeTime;
int fadePeriod = 4000;

////declare the variable int&float for mic
// we need this float in order to increase and decrease the volume of the choir
float gainVal;

// declare the variable for the noise captured by the mic
float maxl = 0;
float newMaxl ;

//declare the variable for the timer
int start;
int time;
int counter;
int gainTimer;
int talking;

void setup(){
  size(512, 200, P3D);
// sets the minim library
  minim = new Minim(this);

//play the choir
  player = minim.loadFile("vespriLEFT_RIGHT_2.wav", 2048);

//gets a signal from the mic
  in = minim.getLineIn(Minim.STEREO, 2048);

  //sets the communication between Processing and Arduino
  textFont(createFont("Arial", 12));
  println(Serial.list());
```

```
  // sets the USB port, in our case the 0 one
  myPort = new Serial(this, Serial.list()[0], 9600);
  myPort.bufferUntil(lf);

  // set the timer
  start = millis();
  time = 1;
  println( time );
  counter=0;
  gainTimer = 0;
}

//Arduino communicates with processing different states
//of the system, according with different moment of the interaction.
//We call this states "mode: 1, 2, 3, 4"

void draw(){
  background(0);

  // in "mode 3" Processing gets a signal from the mic. We know and declare a maximum vo-
lume
  // for the choir (0.07_newMaxl). If the signal got from the mic exceeds this level
  // a counter increases its value. This value, checked every 100 millisecond by a timer,
  // sets the talking variable value
  // (0 or 1 depending on someone is making noise)

  if (mode == '3') {
    for(int i = 0; i < in.bufferSize() - 1; i++){
      newMaxl = in.left.get(i);
      if(newMaxl > 0.07 )  {

        // println("speaking" + newMaxl);
        counter++;
      }

    }

    // the timer start
    if(millis() > start) {//
      start = millis() + 500;
      //time--;

      // the "talking" variable is setting to 0.
      talking = 0;
      // if the counter increase the talking value is sets to 1
      if (counter > 1){
        talking = 1;
      }
      counter = 0;
 }

    // Every period of time check the value of talking variable
    if(millis() > gainTimer ){
      gainTimer = millis() + 100;

   // if the value of talking is > 0 the volume of choir decrease
      if (talking > 0){
        gainVal = gainVal - 1 ; }
```

```
    //Otherwise if there isn't noise the volume goes up
      else{
        gainVal = gainVal + 0.4 ;
      }

    //to a maximum volume
      if (gainVal > 4){// maximum sound
        gainVal = 4;
      }
      if (gainVal < -15){// minimum sound
        gainVal = -15;
      }
      player.setGain(gainVal);
      println("gainValue " +gainVal);
    }
  }

// Fading the sound
  if (mode == '4') {
    if (millis() > fadeTime){
      fadeTime= millis() + (fadePeriod/46);
      if (gainValue> -40){
        gainValue--;
        player.setGain(gainValue);
      }
    }
  }
}

//Start the communication with Arduino.
//The Processing serial library allows for easily reading
//and writing data to and from Arduino.

void serialEvent(Serial p){
  String inString = trim(p.readString());
  mode = inString.charAt(0);
  println(mode);
  switch(mode){




 // The follow case stops and rewind the choir when the experience ends
 //(according with Arduino signal)
  case '1':
    gainValue = 6;
    player.setGain(gainValue);
    if (player.isPlaying()) {
      println("rewind song");
      player.pause();
      player.rewind();
    }
    break;

  case '2':
    break;
```

```
// play the song with a pan effect from left to right
  case ‘3’:
    player.play();
    player.shiftPan(1, -1, 25000);
    break;

  case ‘4’:
    gainValue = 6;
    player.setGain(gainValue);
    break;
  }
}

// always stop Minim before exiting
void stop(){
  in.close();
  player.close();
  minim.stop();
  super.stop();
}
```