|生活·技术·探索|

用Processing将声音视觉化

张敬云　江苏省镇江市实验高级中学
谢作如　浙江省温州中学

涉及学科：信息技术、科学

你见过声音的样子吗？在文学作品中，常常会将声音进行视觉化，如白居易《琵琶行》中的“嘈嘈切切错杂弹，大珠小珠落玉盘”，形象而生动地描述出声音的样子。

在互动艺术作品中，也常常会看到艺术家尝试将不同的感官进行转换。将声音视觉化或者把图像转为声音，是常见的艺术表达形式，如将一段或欢快或优美或震撼的音乐视觉化，以跳动的柱形或流水般的霓虹灯展示出来；将如仙乐般美妙的海豚之声视觉化，以花团锦簇的图像展示出来，带给我们听觉视觉的双重体验（如图1）。而实现这类创意，肯定要选择Mit的Processing软件。这是一款艺术家用来“画画”的编程语言，能够将各种奇妙的声音变幻为美轮美奂的图像，让受众感受到艺术创作的魅力。

原理分析

声音是一种模拟信号，而计算机处理的只能是数值，所以存储在计算机中的声音文件都是经过处理之后的数字化文件。声音的数字化是通过“采样”和“量化”的方式，实现波形声音模拟量的数字化。那么Processing又是如何实现声音的视觉化呢？只要能够获得声音的响度、频率等信息，借助傅立叶变换之类的算法，解析出更多的信息，就能将其以柱形、圆形、线形或者任何你想描绘的图形唯美地展现出来。

Processing是一个开源的编程语言，有很多人为其开发了各种开源的库。声音处理方面功能最

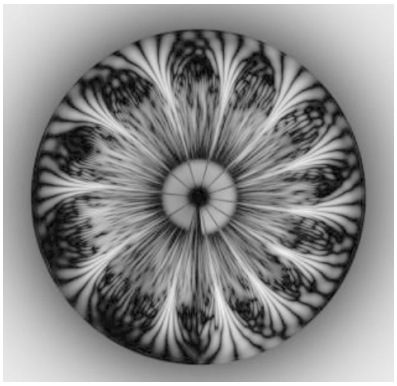

图1

表1

```
import ddf.minim.*;     //库引用
AudioPlayer player;  //创建 AudioPlayer 对象
Minim minim; //创建 Minim 对象
void setup(){
  size(800,400);//设置窗体大小
  smooth();//设置线条平滑效果
  minim=new Minim(this);   //初始化 minim 对象
  player=minim.loadFile("burning.mp3");   //读取音乐文件，默认缓冲区大小为 1024 字
  player.play();
}
void draw(){
  background(0);
  stroke(255);
  for(int i=0;i<player.bufferSize()-1;i++){  //用循环读出音乐缓冲区中的所有数据
      line(i,150+player.left.get(i)*50,i+1,150+player.left.get(i+1)*50);//根据 pla
  }
}
```

强大、应用也最广的是Minim库。借助这个Minim库，我们不需要理解傅立叶变换，就能够分析出声音的频率、振幅、节奏等信息。Processing 3.3.6版本已内置了Minim库，我们可以从“贡献管理器”中安装它。通过“速写本—引用库文件—添加库文件”打开贡献管理器的Libraries选项卡，在Filter框中键入Minim，从列表中选择库，然后单击Install即可。

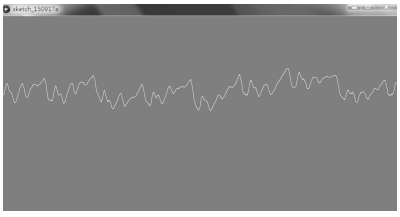

图2

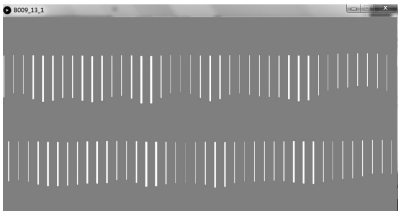

图3

基本代码

首先我们来看一个用Processing将声音视觉化的例子，基本代码如上页表1所示。

运行程序，开始播放音乐，Processing窗口会有波浪起伏的线条跳动（如图2），其效果是根据实时获取的音频频率实现的，当频率高时，线条的上下起伏大，当频率低时，线条的上下起伏小，是不是像极了我们起伏不定的心跳变化？

基于这个作品，我们具体分析一下用Processing将声音视觉化的过程。这个程序由setup和draw两个函数组成，其中setup仅运行一次，draw则在不断循环。程序读取声音文件“burning.mp3”的left.get()，即左声道的数值，然后用line函数画出线条。因为draw函数在循环执行，于是就显示出动态的线条跳动效果。

Minim库的Minim对象和AudioPlayer对象的常用方法如表2所示。需注意的是，left.get()和right.get()返回的值是−1和1之间的小数。

表2　　Minim库的基本使用说明

Minim对象	loadFile()：添加音频文件。通过“速写本”—“添加文件”或直接将音频文件拖到Processing窗口中即可，在工程目录下便会自动创建一个data文件夹
AudioPlayer对象	play()：播放音频
	buffer.Size()：音乐缓冲数据大小
	left.get()：左声道数值
	right.get()：右声道数值

表3

```
void draw() {
  background(0);
  stroke(255);
  for (int i=0;i<player.bufferSize();i++) {
    strokeWeight(abs(player.left.get(i)*10));//线条粗细
    rect(i*20, 75+player.left.get(i)*10, 1, 75+player.left.get(i)*40);
    rect(i*20+10, 250+player.right.get(i)*10, 1, 75+player.right.get(i)*40);
  }
}
```

创意实现

根据上文给出的用Processing将声音视觉化的基本代码，我们能够勾勒出更多有趣又有艺术气息的视觉化作品。如图3所示，随着音乐的节奏，小矩形的粗细、长短都会一起发生变化，会不会让你想到一种叫“排钟”的打击乐器呢？那是由一些长短有别的钢管或铜管组成的乐器，用木槌敲打便会产生清脆、悦耳的旋律。观看这种互动效果，你会有一种错觉：这些炫酷的动态效果并非由音乐产生，而是由于它们的律动才产生了音乐。

这个创意的实现，是根据实时获取的左声道及右声道的数值来画多个矩形，矩形的长度和宽度分别由数值的大小来决定。核心代码如表3所示（为避免重复，只保留void draw()的代码，下文中出现的案例代码同理）。

为了营造欢快唱歌的氛围，很多人去KTV，第一件事就是把房间的“星空灯”打开。这个“星空灯”我们更愿意称之为“霓虹灯”，五彩缤纷的，似乎就是我们美妙的歌声才使得灯光跟着欢快起来。同样，在激情四射的演唱会上，那些各色的互动荧光棒似乎也是随着

现场动感的节奏而舞动。那么利用Pocessing，能否将“放眼望去的霓虹灯”或“漫天的星光”实现真正意义上的随着音乐旋律而闪耀呢？答案是肯定的。

根据实时获取的音频频率，确定圆的大小及颜色。代码如表4所示。

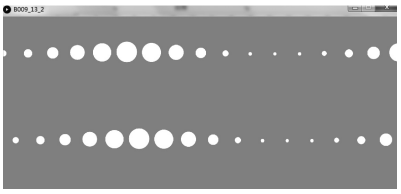

图4

表4

```
void draw() {
background(0);
for(int i=0;i<player.bufferSize()-1;i+=25){
float y=(int)random(400),x=(int)random(800);
fill(player.left.get(i)*5000,player.left.get(i)*5000,255-player.left.get(i)*5000);
ellipse(x, y, player.right.get(i)*150, player.left.get(i)*150);
}
}
```

表5

```
import ddf.minim.*;
Minim minim;
AudioInput in;
void setup()
{
  size(800,400);
  smooth();
   minim = new Minim(this);
   in = minim.getLineIn(Minim.STEREO,1024);
   background(0);
}

void draw()
{
  background(0);
  stroke(255);
  for (int i=0;i<in.bufferSize()-1;i++) {
    strokeWeight(abs(in.left.get(i)*20)*5);
    point(i*50, 75+in.left.get(i)*10);
    point(i*50+25, 250+in.left.get(i)*10);
  }
}
```

上面展示的几个作品都是用Processing将现有的音乐进行视觉化，如果想做声音的即时视觉效果，又该怎样实现呢？很简单，使用麦克风就解决啦！具体代码如表5所示。

运行程序，窗口上圆点的大小会随着麦克风音频频率的高低而变化，当频率高时，圆点变大，当频率低时，圆点变小。看着屏幕上的圆点随着声音奇妙地发生变化（如图4），就像感觉到自己的心跳在“砰砰砰”紧张又有韵律地跳动一般。

从传统意义上讲，声音属于耳朵，是用听觉体验美感的艺术。但随着新媒体技术的发展，声音也可以属于眼睛，甚至皮肤等感觉器官。用视觉来理解声音的艺术魅力类似文学修辞手法中的“通感”，生活中常见的音乐喷泉、喷水音响等，都是试图把声音视觉化的典型。而这些效果，都可以用Processing来实现。

你希望声音是什么样子？请用Processing将你喜欢的声音勾勒出你喜欢的图案吧，艺术和科技本来就可以很好地结合。e

如果对相关内容感兴趣，请关注主持人博客。