音乐可视化报告

沈钇蕾 16307130285

工具选择

在本项目中，我选择的工具是 Processing+Minim。

由于是初次使用 Processing，因此我查看了一些 Processing 的使用教程，并且为了能够用到更多其中提供的工具，我仔细阅读了 Processing 的官方说明，学着自行调整各个参数，思考了对于理想中的效果，应该如何进行函数的调用，以达到效果的切换。

算法原理

本项目中算法的主要思想是运用了傅里叶变换。

每一段音频都是正弦波的叠加。对信号而言，信号强度随时间的变化规律就是它的时域特性，而信号是由哪些单一频率的信号合成的就是它的频域特性。傅里叶变换的实质就是频域函数和时域函数的转换。

在本项目中，通过对音频进行傅里叶变换，得到实时的信号强度值，再由其强度值来绘制动态图形，实现音频的可视化。

程序说明

本项目中涉及了四种特效，分别是三角、雨点、螺旋、正方形特效。其中，三角特效与雨点特效一同出现。

首先，调用 minim 库，并对项目中所需的变量进行声明。

代码如下：

```
import ddf.minim.*;
import ddf.minim.analysis.*;

Minim minim;                    //声明了 mimin
AudioPlayer player;             //声明了一个 AudioPlayer 类型的变量 player
FFT fft;                        //傅里叶变换

int low_amp;            //低强度
int medium_amp;    //中等强度
float high_amp;       //高强度
int threshold;          //阈值

int option;              //“选项”变量，用于选择特效

// 正方形特效变量
int x_offset;
float rotate_random;

// 三角形特效变量
int triangle_loc;

// 螺旋特效变量
float arc_len;

// 计数变量
int count;
```

其中，low_amp、medium_amp、high_amp 等变量是经由傅里叶变换所获取的强度值，之后会有相应代码。

声明变量后，需要对其进行初始化。

代码如下：

```
void setup() {          //只在程序开始时运行一次
  fullScreen(P3D);     //设置全屏
  frameRate(60);       //每秒运行的帧数
  background(0);       //设置背景色为 0
  noCursor();          //隐藏鼠标
```

```
  minim = new Minim(this);   //对 mimin 进行实例化
  player = minim.loadFile("Immortals.mp3", 1024);
        //加载音频文件的变量   1024：提取的频率
  fft = new FFT(player.bufferSize(), player.sampleRate());
        //对傅里叶变换的变量进行实例化
  option = 1;                    //初始化选项
  threshold = 150;

  //正方形特效变量
  x_offset = 0;
  rotate_random = random(1, 24);
  stroke(255);

  //三角特效变量
  triangle_loc = width/2;

  //螺旋特效变量
  arc_len = 0.0005;

  count = 0;
}
```

初始化后，在绘图函数中执行傅里叶变换，并调用各种特效。（关于特效的实现细节在之后介绍）

代码如下：

```
void draw() {                  //循环执行的绘图函数
  fft.forward(player.mix);     //对音频文件进行傅里叶变换
  low_amp = int(fft.getFreq(50));
  medium_amp = int(fft.getFreq(2000));
  high_amp = map(fft.getFreq(20000), 0, 1, 0, 500);
        // map(value, start1, stop1, start2, stop2)
        // value：当前值   start1：当前值下届   stop1：当前值上界
        // start2：目标值下届   start3：目标值上界
  count += 1;
  if(count >= 3){
    if (low_amp>threshold && frameCount>240) {
        //frameCount:程序自启动以来已显示的帧数
      option = floor(random(1, 3.99));
        //对 option 做随机处理
    }
```

```
        count = 0;
    }

    if (option==1) {
        trianglepop();
        rain();
    }
    if (option==2) {
        squaregroup(20);
    }
    if (option==3) {
        spiral();
    }

    //当前 mp3 文件被替换时，加载新的文件
    if (player.isPlaying() == false) {
        option = 1;
        player = minim.loadFile("Immortals.mp3", 1024);
        player.rewind();
        player.play();
    }
}
```

对这部分代码进行解释：

首先，option 的初始值设为 1，即刚开始的特效为三角+雨点。最开始的 240 帧，特效不变（之前设置每秒 60 帧，因此 240 帧对应了 4 秒）。之后，当 low_amp 大于预先设定的阈值，并且该特效已维持了超过 3 帧，则随机在 3 中特效中进行选择。设置变量 count 的目的，是为了防止特效转变过快。

三角+雨点的特效代码如下：

```
    //三角特效
void trianglepop() {
    if (triangle_loc==width) {
        triangle_loc = 0;
    } else {
        triangle_loc = triangle_loc + 1;
    }
    pushMatrix();
```

```
    background(0);
    translate(triangle_loc, height/2);
    fill(255);

    for(int i=0;i<player.bufferSize()-1;i++){
      strokeWeight(abs(player.left.get(i)*10));
    }
        //line(x1, y1, x2, y2)
        //x1、y1：第一个点的横纵坐标
        //x2、y2：第二个点的横纵坐标
    line(-width, 0, -width*7/8, -height/8-high_amp);
    line(-width*3/4, 0, -width*7/8, -height/8-high_amp);
    line(-width/2, 0, -width*3/8, -height/8-high_amp);
    line(-width/4, 0, -width*3/8, -height/8-high_amp);
    line(0, 0, width/8, -height/8-high_amp);
    line(width/4, 0, width/8, -height/8-high_amp);
    line(width/2, 0, width*5/8, -height/8-high_amp);
    line(width*3/4, 0, width*5/8, -height/8-high_amp);

    popMatrix();
}

    //雨点特效
void rain() {
    fill(255);
    textSize(random(0, high_amp));
    text("|", random(width), random(height));
        //text(c, x, y)   c：要显示的字符   x：横坐标   y：纵坐标
        //这里横纵坐标选择 random(width)与 random(height),
        //是为了使雨点特效布满整个屏幕
}
```

在三角特效中，调用了 translate 函数进行平移，使得画面的律动感更强。

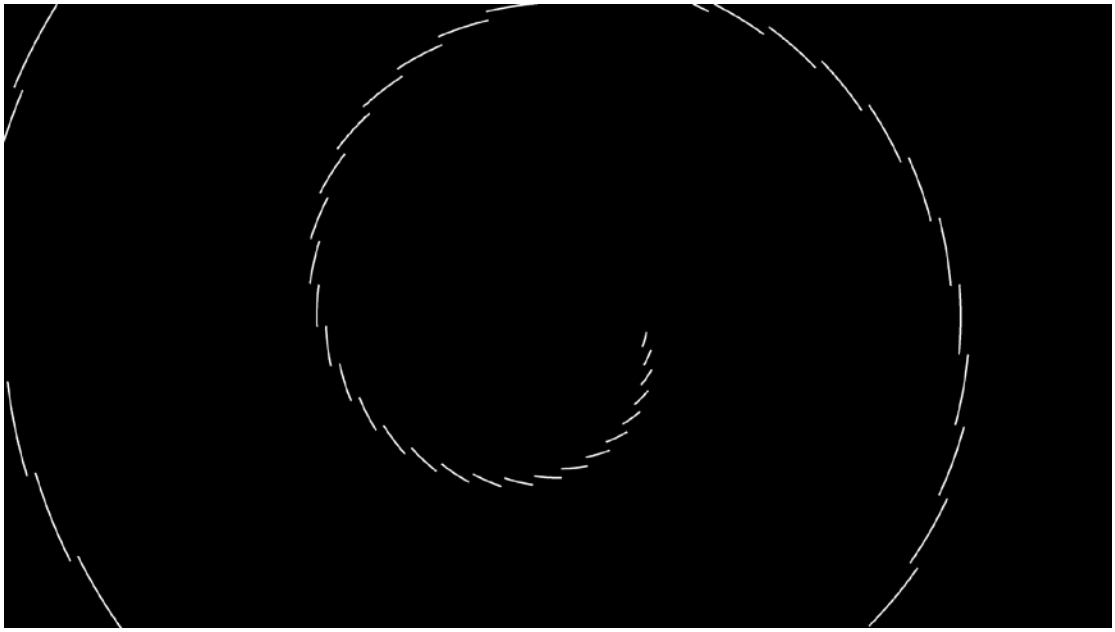
螺旋特效代码如下：

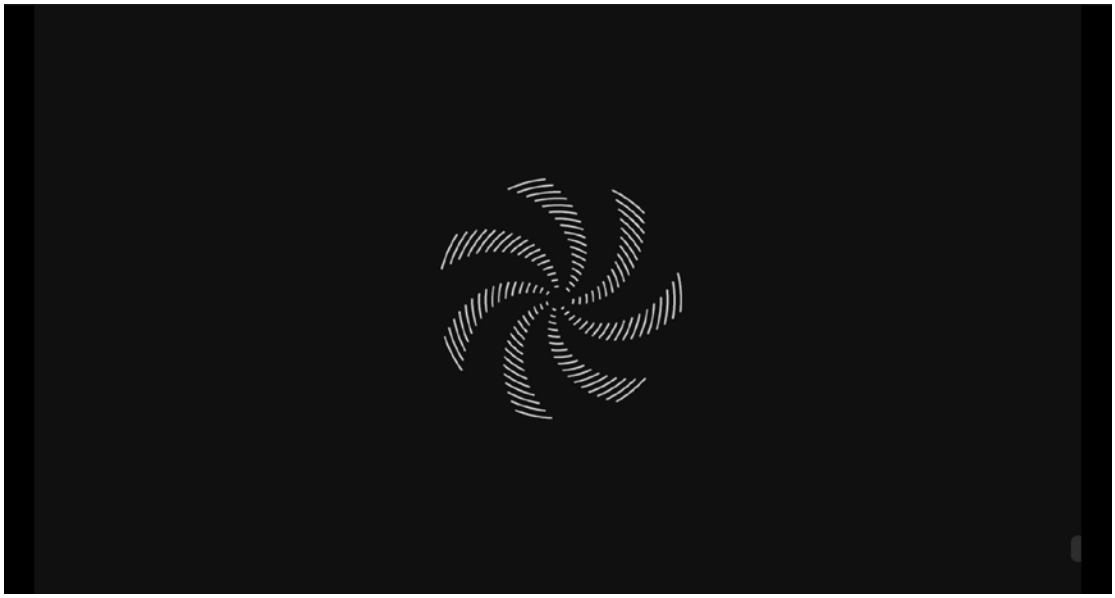
```
    //螺旋特效
void spiral() {
    background(0);   //背景设置为黑
    noFill();        //对几何图形不进行填充
    stroke(255);     //选择线条颜色为白色

    arc_len += 0.0001;
```

```
    if (arc_len == 10) {
      arc_len = 0.0005;
    }

    translate(width/2, height/2);      //translate(x 平移,y 平移)以应用一个平移变换。
    for (int r=50; r<650; r=r+5) {
      rotate(millis()/1200.0);          //millis()：启动程序以来的毫秒数
                                        //rotate：根据指定的弧度进行旋转
      strokeWeight(3);
      arc(0, 0, r*low_amp/10, r*low_amp/10, 0, arc_len);
          //arc(a, b, c, d, start, stop, mode)
          //a：圆弧椭圆的 x 坐标     b：圆弧椭圆的 y 坐标
          //c：弧的椭圆宽度          d：弧的椭圆高度
          //start：弧开始的角度      stop：弧停止的角度
    }
}
```

螺旋特效的旋转角度依赖于启动程序以来的毫秒数，使得画面能够连贯、美观。

螺旋效果如下：

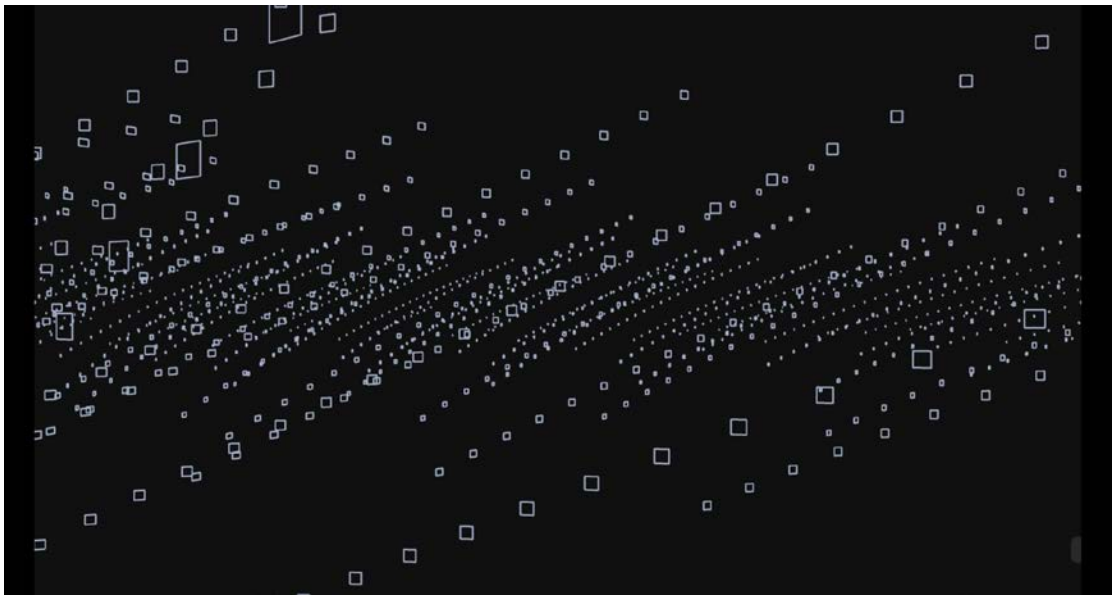


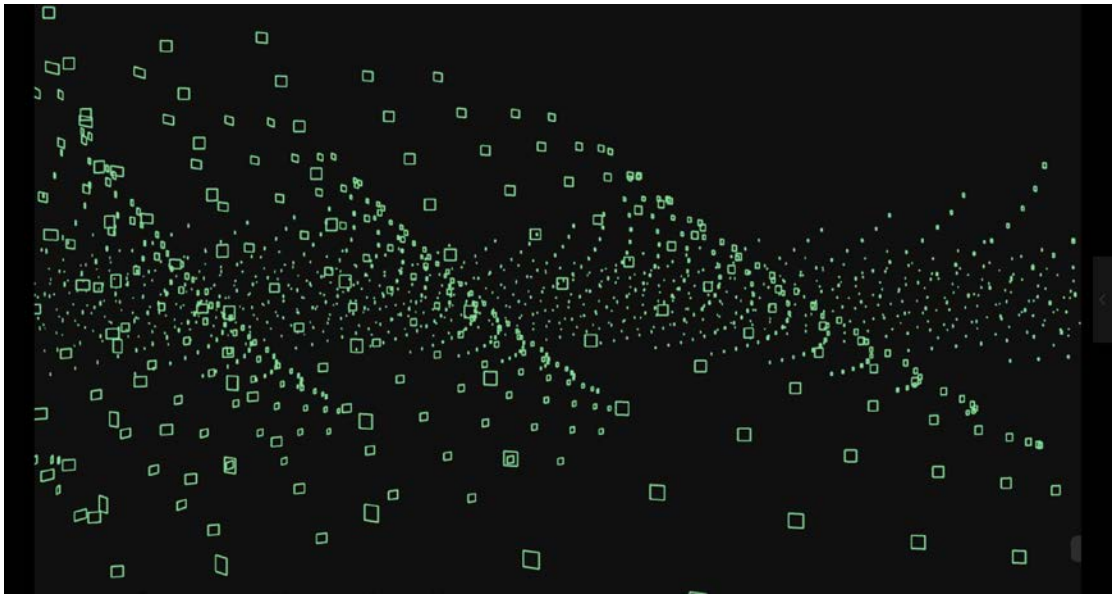


正方形特效代码如下：

```
   //正方形特效
void squaregroup(int square_wid) {
   background(0);
   pushMatrix();
   x_offset -= 3;
   if (x_offset<-width*3) {
      x_offset = 0;
   }
   strokeWeight(3);
   if (low_amp>threshold) {
      rotate_random = random(1, 24);
      stroke(random(125, 255), random(125, 255), random(125, 255));
   }
   translate(x_offset, 0);
   //设置大量的排列有序的正方形
   for (int i_x=0; i_x<width*12; i_x+=50) {
      for (int i_y=0; i_y<height; i_y+=50) {
         noFill();
         rotateY(rotate_random);          //rotateY：沿 Y 轴转动的角度
         square(i_x, i_y, square_wid);
      }
   }
   popMatrix();
}
```

正方形效果如下：





生成结果如 demo.mp4 所示。

参考网站

https://processing.org/reference/

https://github.com/andrele/Starburst-Music-Viz