信息可视化个人作业报告

1. 系统描述

本次个人作业我实现了一个对音频文件的频谱的可视化，具体效果类似于一般播放器的 spectrum，主要在于数据的动态采集以及数据的动态可视化，当然数据本身不是十分复杂的数据。

2. 所用语言/库

主要使用 html 和 JavaScript 语言，使用了两个基于 JavaScript 的库，一个是 d3.js 主要用于可视化，另一个是 timbre.js 主要用于数据采集。

3. 关键代码

1) 数据采集

主要使用 timbre.js 的库实现

播放音频，timbre 中提供了播放音频的方法

```
var wav = T("audio", {loop:true}).loadthis("/resource/test.mp3", function() {
            this.play();
          }).on("ended", function() {
            this.pause();
          });
```

监听音频获取数据，timbre 本身提供了一个 spectrum 参数可以获取频谱数据，下图代码中 512 就是 fft 的参数，100 是指每过 100ms 获取一次，通过调用 listen() 的方法就可以实现对音频的监听，fft.spectrum 会返回一个大小为 128 的数组，里面存储了我们需要的数据，数据的具体含义应该是每个频段的响度，官方的参考提到它的单位是 db，但是具体分频是怎么分频我并不了解（也许具体和 fft 有关），因此具体实现时我只是将其做成了 periodi 的形式来表示某个频段。

```
var fft = T("spectrum", {size:512, interval:100}).on("data", function() {
    array = fft.spectrum;
    rects.transition().duration(100)
         .attr("y", function(d, i){
            if(gseli == 128) {
              txt.text(function(d){
                return "No period selected!";
              })
            }
            if(gseli == i) {
              changetxt(i, fft.spectrum[i]);
              return 600-5*scale(fft.spectrum[i]);}
            else
            return 600-scale(fft.spectrum[i]);
         })
         .attr("height", function(d, i){
            if(gseli == i) return 5*scale(fft.spectrum[i]);
            else
            return scale(fft.spectrum[i]);
         });
}).listen(wav);
```

2) 可视化

主要思路是将数据绑到柱状图上，当然这个不难，主要是动态更新，关于这个我写了一个 update 的函数，每次获取数据的时候会调用这个更新函数，他会重新设置 svg 中 rect 元素的属性并设置动画。

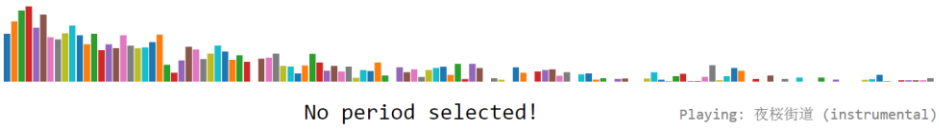
```
function updateRects(seli){
  rects.transition().duration(100)
          .attr("x", function(d, i){
             var selwidth = 5*width/128;
             var otherwidth = (width-selwidth)/127;
             if(i <= seli) return i*otherwidth;
             else
             return (i-1)*otherwidth+selwidth;
          })
          .attr("width", function(d, i){
             var selwidth = 5*width/128 - 1;
             var otherwidth = (width-selwidth+1)/127-1;
             if(seli == i) return selwidth;
             else
             return otherwidth;
          })
          .attr("y", function(d, i){
             if(seli == i) return 600-5*scale(fft.spectrum[i]);
             else
             return 600-scale(fft.spectrum[i]);
          })
          .attr("height", function(d, i){
             if(seli == i) return 5*scale(fft.spectrum[i]);
             else
             return scale(fft.spectrum[i]);
          })
          .attr("opacity", function(d, i){
             if(seli == i || seli == 128) return 1;
             else return 0.3;
          });
}
```

其中给定的参数 seli 指定了 svg 某个鼠标选中的 rect，这个参数由鼠标的事件函数控制

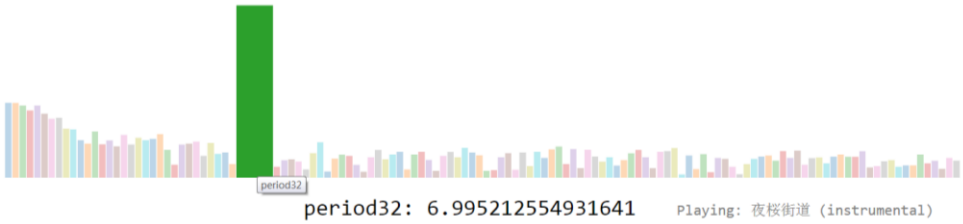
```
rects.on("mouseover", function(d, i){
        gseli = i;
        updateRects(i);
        updateRects2(i);
    })
     .on("mouseout", function(d, i){
        gseli = 128;
        updateRects(128);
        updateRects2(128);
     });
```

4. 结果描述

未选中某个频段时的效果图



选中某个频段时的效果图



可以看到，下方的柱状图是对每个频段的数值的实时反映，也就一般播放器中的效果，当然在这里我设置了当鼠标移到某个柱体时我们的关注点也会转移到该柱体的效果（该柱体变大，其余柱体变透明），同时下方也能显示具体数值。

而上方的柱体是当对应的频段的数值大于某个值时它才会做出对应的动画，这样可以将某些节奏感可视化。

当然实现上还是有缺陷的，由于实时刷新频繁的缘故（个人猜测），音频文件的播放偶尔会卡顿，而且响应采集数据变化的动画 transition 和响应鼠标动作的动画 transition 有时也会冲突，导致某个 transition 无法做完。

5. 个人感想

虽然就结果来看做的还是简陋了一些，不过我感觉很有收获的，自学了不少东西，独立完成了一个自己想做的东西，不过由于时间限制加上精力有限很多粗糙的地方也没有办法优化了。实际做的过程中确实感到 d3.js 是一个十分灵活的工具，不过感觉更需要的还是一个可视化的创意，这样才能有针对性的进行可视化，使我们做的工作真的具有分析上的意义，而不是简单的美化。