数字几何处理第一次作业

软硕163 周伯威 2016213588

1. 运行环境

- 编程语言: JavaScript
- 浏览器支持: Chrome
- 使用Chrome打开index.html即可运行

2. 框架

- **Materialize css**: 网页使用Materialize css作为前端模板。
- **Three.js**: Three.js是一个开源的JavaScript库，其中包含了对WebGL的封装，提供了丰富的API方便调用。

3. 实现细节

3.1 文件组织

- index.html: GUI
- main.js: 模型的显示、交互逻辑等
- color.js: 颜色空间转换
- algorithm.js: 各种算法
- Colormap.js: Colormap类，负责生成颜色映射、计算插值
- Edge.js: Edge类，存储边
- Colorparser.js: Colorparser类，用于读取颜色表
- ObjParser.js: ObjParser类，解析OBJ格式模型
- OffParser.js: OffParser类，解析OFF格式模型

3.2 模型显示

为了更好的显示效果，除了基本的模型外，还添加了两个光源以及地面、fog效果。

使用Three.js examples中的Orbit control来旋转(缩放、平移)模型，使用Orbit而不是Trackball的原因是需要保持模型始终水平，不出现上下颠倒的情况。(`main.js: function init()`)

程序限制了最大、最小缩放，以及最大的仰角，避免模型过大、过小，以及看到地面下方的情况。程序对模型的大小做了归一化。计算其包围球的球心、半径，将模型缩放至直径为1的球内。为了使模型的下边界贴合地面，程序计算了模型包围盒，平移模型使包围盒底面紧贴地面。

```
// algorithm.js: function optimizeMesh(mesh)
var u = [scale / r, scale / r, scale / r];
var v = [-c.x * scale / r, -c.y * scale / r - 0.5 +
(c.y - b1.y) * scale / r + 0.0001, -c.z * scale /
r];
mesh.scale.set(u[0], u[1], u[2]);
mesh.position.set(v[0], v[1], v[2]);
```

3.3 数据结构

Three.js的模型数据存储结构与Obj文件类似，包含vertices(存储坐标)、faces(存储顶点索引号)。为了实现作业功能，需要添加更多数据结构。(algorithm.js: function addDataStructure(mesh))

根据功能需求，计算的顺序依次为: 顶点相关联的面、顶点相邻顶点、边的端点与相关联的面、面相邻的面、顶点法线。

3.4 颜色映射

程序支持单色、连续、离散三种模式的颜色映射。可以在颜色选择区域右侧预览颜色映射。

其中，连续颜色模式需要设置渐变的固定颜色位置，例如，$[0.4, [255, 0, 0]]$表示数值处于40%位置的点被映射为红色。为了得到更合理的颜色过渡，程序使用了Lab颜色空间进行颜色的插值。程序支持强大的自定义颜色映射，三个模式均可以由用户自行设置并预览，详见说明书。

3.5 绘制区域功能

该功能相当于根据顶点查找面，如果使用普通方法，对于总面数m与查找顶点数n，复杂度将为$O(mn^2)$，或者$O(mn^3)$，当n较大时这个是不行的，所以需要加速查找。这里，我建立了顶点索引及面索引，将复杂度降低到了$O(kn)$。(algorithm.js: function findVerticesAdjacentFaces(verticesIdx))

3.6 高斯滤波

进行高斯滤波时，需要计算每个顶点邻域内包含的其他顶点。如果查找全部n个顶点，则总复杂度为$O(n^2)$，同样需要加速。我的方法是预先将空间切分成$k \times k \times k$个bins，其中$k = \max(4, \min(20, \lfloor \sqrt[3]{n} \rfloor))$(经验值)，遍历顶点，将每个顶点放入对应的bin中。查找时，取出待查顶点周围所有可能的bins，并判断其中顶点是否符合要求。该方法将复杂度降低至约$O(\frac{n^2}{k^3}) \approx O(n)$。

```
for (var i = 0; i < g.vertices.length; i++) {
    var v = g.vertices[i].clone();
    v.sub(b1);
    v.divide(size);
    idx = [Math.floor(v.x), Math.floor(v.y),
Math.floor(v.z)];
    bins[idx[0]][idx[1]][idx[2]].points.push(i);
}
```

邻域半径范围规定为3σ，查找到点后根据到中心点的距离，为其添加相应的高斯权重。最后，求出所有权重和，并使这个和为1，将各个权重归一化。

实验(计算MSE)发现，高斯滤波σ的取值为添加高斯噪声的σ的2倍时，MSE较小。

3.7 模型差异的可视化

为模型的每个顶点添加颜色来实现滤波后的网格与加噪声前的网格的差异的对比。

首先计算原始模型的顶点法线，计算新模型对应顶点到原顶点法平面的距离(有符号)，将这个距离作为该顶点的颜色数值。利用Colormap类生成一个热力图(蓝-青-绿-黄-红)，查找该数值对应的颜色，并渲染出来。

因为误差的分布也是中间大，两边小，将颜色线性地映射至顶点上会导致大多数颜色均为中间值绿色，所以我对数值进行了处理来放大差异：
$value' = \frac{|value|}{value} maxvalue \sqrt{|\frac{value}{maxvalue}|}$。

4. 心得体会

图形学课还有其他地方写过类似的程序，积累了很多经验，所以这次作业写起来很顺手，加了一些额外的功能。大四的图形学、图像处理课的作业全是用JavaScript做的，于是这次也这样做了。。好处是不像OpenGL那样对平台有限制，这个拿手机都能看= =

还有就是工作量挺大的。。2300行代码= =