复旦大学软件学院

计算机图形学 Project 1

四边形网格扫描转化与交互式编辑

发布时间：**2015-09-29**

Deadline：2015-10-22

负责 **TA**：沈剑锋

email:14212010014@fudan.edu.cn

Project1 四边形网格交互式编辑

项目目的

掌握多边形扫描转换算法，了解图形学基本绘制方法，熟悉 web 图形应用程序开发的基本步骤。

项目要求

实现四边形网格交互式编辑：要求实现四边形扫描转化算法，以此为基础，实现四边形网格中所有四边形的扫描转化、颜色添充；并通过交互改变四边形顶点位置体会交互式图形系统特点。

使用的语言

Html5, Javascript

开发与测试环境

支持 html5 (主要是 canvas 标签)和 Javascript 的浏览器，一般 Internet Explorer 11+（本次 lab 因为只用到 html5，所以可以使用 ie9，但之后的 project 需要使用 webgl，所以建议使用 ie11 版本），Firefox, Opera, Chrome 以及 Safari 支持 <canvas> 标签（可以用 Browser-test.html 进行测试）

四边形网格扫描转换

1. 功能描述

1.1 四边形网格扫描转化及颜色填充

四边形网格是由多个四边形紧密连接的一个网状几何形状。构建一个四边形网格，我们需要有以下信息：网格的所有顶点位置，网格中的四边形顶点连接关系。我们可以从给定配置文件中得到顶点坐标、四边形顶点连接关系等信息，并用它们来构建四边形网格。然后利用四边形扫描转换算法，逐扫描线进行增量式计算，得到四边形内部的象素点,并将指定填充颜色赋值给象素点。

给定配置文件信息如下：

(a)顶点几何信息：包含各顶点的编号及其三(二)维坐标。

(b)顶点颜色信息：包含各顶点的编号及其颜色(RGB 三通道)。每个四边形的四个顶点的颜色可能不同。这里一律选用四边形顶点连接关系数据中四边形的第一个顶点的颜色作为此四边形的填充颜色。

(c)四边形顶点连接关系：包含各四边形的顶点编号。

例如某个四边形颜色填充如下图所示：

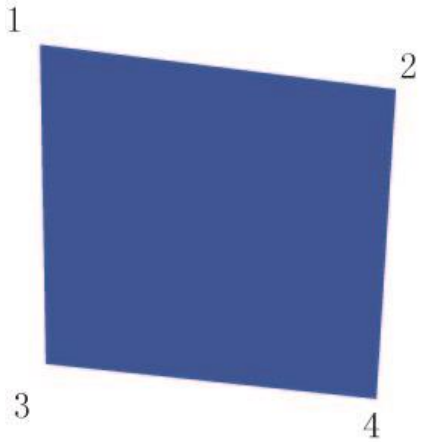


图 1：颜色填充（四边形的填充颜色为蓝色）

1.2 拖动顶点位置

实现拖动四边形顶点的功能，如图 2 为例。使用者利用鼠标选择居中位置的顶点，然后按住鼠标左键进行拖动操作。鼠标拖动过程中，以该点为顶点的所有四边形都必须重新进行扫描转化，从而交互实现四边形网格动态绘制。

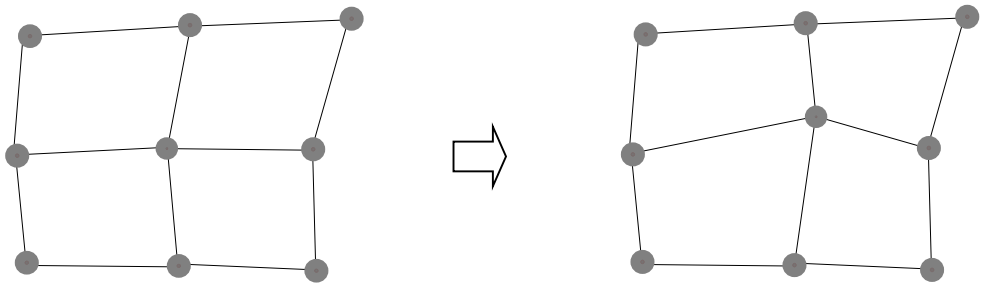


图 2：四边形顶点拖动

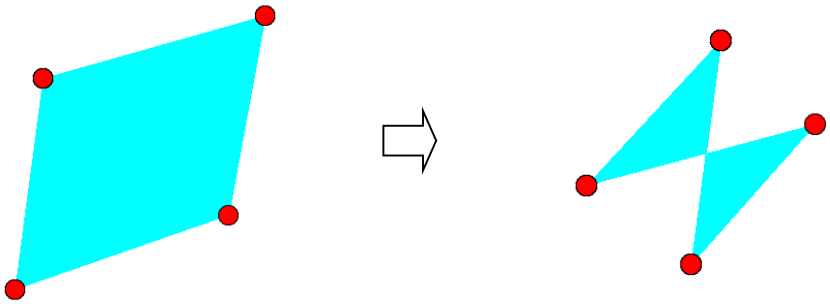


图 3：实际显示范例

2. 实现步骤

1. 下载并安装支持 html5 和 webgl 的浏览器（如 ie11，最新版本的 Firefox，chrome 和 Safari 等）
2. 使用安装好的浏览器，打开（运行）项目目录下 test 文件夹中的 testBrowser.html 确认浏览器支持 html5 以及是否已经开启了对 webgl 的支持，如果没有，请在浏览器设置选项中开启对 webgl 的支持。
3. 详细了解项目目标与基本实现方法，项目目录下 sample 文件夹中有本次项目的结果范例（不可抄袭 sample 中的代码）。

4. 学习 Canvas 的基本使用方法，这里你需要知道如何调整画布的大小以及如何在画布上绘制基本的点和线（canvas 的坐标系统比较特殊，在绘制宽度为 1px 的线段时可能会存在宽度和颜色上的变化，我们需要通过 canvas 的 context 调用 **translate(0.5, 0.5);**函数使得 canvas 横纵坐标整体偏移 0.5，具体原理见后文），建议参考项目 src 目录下的 scanConversion.html 中的源代码，或者查阅 w3school 上面关于 html5 的教程，
   http://www.w3school.com.cn/html5/html_5_canvas.asp 和
   http://www.w3school.com.cn/tags/html_ref_canvas.asp
5. 使用 html5<canvas>和 javascript 实现四边形扫描转化算法与网格交互式编辑，你可以直接修改项目 src 目录下的 scanConversion.html（如果使用 chrome 浏览器的开发人员工具进行调试，建议将 javascript 脚本代码放置于独立的 js 文件中）。四边形网格数据配置文件 config.js 存放在项目 src 目录下，具体使用方法和数据含义详见本文附录。

6. 关键步骤：
   a) 首先实现单一四边形的扫描转化
   b) 然后实现四边形网格编辑，也就是顶点的拖拽。由于鼠标很难精确的对准某个顶点，所以你需要考虑容差，即当鼠标比较靠近（不需要完全对准）某个顶点的时候就可以拖拽该顶点。
   c) 编辑你的四边形网格，确保你的扫描转化算法在任何情况下都准确无误，比如存在水平边，凹四边形，存在交叉边等
   d) 从配置文件中读入网格数据并显示，确保你的代码能够正确显示网格数据。

7. 测试并调优你的代码。
8. 编写你的项目文档，内容需包括但不限于：你的项目目录及文件说明，开发及运行环境，运行及使用方法，你的项目中的亮点，开发过程中遇到的问题（以及你的解决办法），项目仍然或者可能存在的缺陷（以及你的思考），你对本课程 project 的意见及建议等。
9. 完成后将你的代码和文档放置在同一文件夹中，文件夹名称为”[学号]_[姓名]”，使用 zip 格式压缩后上传至 ftp。

3. 项目要求及注意事项

本阶段需要完成以下功能：

1、 从配置文件中读取信息来构造四边形网格（包含顶点位置、顶点颜色和顶点之间的连接关系）。

2、 四边形网格扫描转化。

3、 利用鼠标交互式修改四边形网格中任意顶点位置。

注意事项：

1、 代码具有可读性：

a) 所有代码有合理的缩进

b) 类，函数，变量需合理命名

c) 所有的函数，重要代码，算法中的关键步骤添加注释

2、 不允许使用 canvas 操作中任何直接绘制多边形的函数完成规定功能。例如不能用 rect 或者 fillrect 等矩形绘制和填充函数，不能通过闭合路径直接绘制多边形，只可以使用画点和画线的函数。

3、 Canvas 画布的大小应当根据配置文件中的值进行设置

4、 扫描填充过程中须特别处理扫描线平行于四边形某一条边的情形。

5、 任意四边形的顶点不应跨越 canvas 画布边界。

6、 对四边形网格进行填充时需注意没有四边形被遗漏。

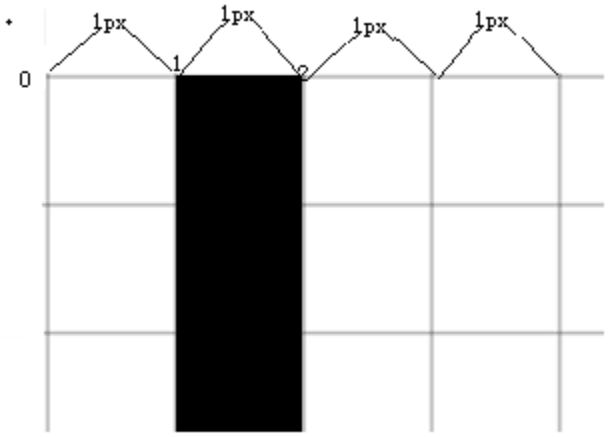
7、 可以在顶点上添加醒目的圆形手柄，方便拖动控制，如图 3 所示

8、 Javascript 是弱类型的编程语言，所以在进行数据运算时务必注意。

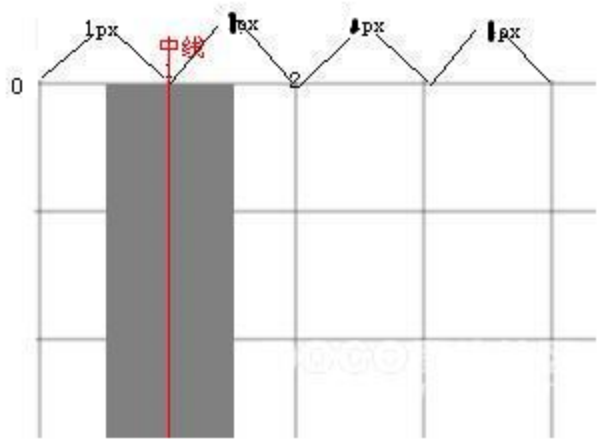
4. Canvas 中绘制宽度为 1px 线段的问题

在 Html5 Canvas 中绘制宽度为 1px 的线段有时会存在宽度和颜色上的错误，这是由于 Canvas 的绘制原理造成的。

通常屏幕上的图像是由一个个的像素组成的，假如我们想要绘制一条横坐标为 1，宽度为 1 个像素的纵向黑色线段（坐标原点从 0 开始，如图所示），我们是希望在第二列像素的颜色均为黑色。



但是 Canvas 关于坐标的处理机制有所不同。它的整数坐标实际上并不在每个像素的中间位置，而是在两个像素之间的边缘上，因此，Canvas 在绘制纵坐标为 1 的线段时，会**尝试**绘制一条宽度从第 0 列像素中间位置开始，到第一个列素中间位置结束的线段。如图所示：



但是显示器无法进行亚像素级别的显示，因此，最终绘制结果就变成了第 0 列和第 1 列像素均被填充上了颜色，这里为了保证颜色总量不变，因此二者的透明度均为 50%，所以，想要在第 x 列（行）绘制真正宽度为 1px 的线段，我们传入的坐标应当为 x+0.5。或者通关相关函数，cxt.translate(0.5, 0.5); 将 Canvas 整体坐标偏移 0.5 个像素即可。

事实上，这种绘制方式支持任意浮点型的坐标，在处理方式上类似于反走样，对于如下代码，在水平方向上从坐标 0 开始，从左至右依次绘制了 20 条长度为 10px 的水平线段，每条线段的纵坐标都比上一条向下偏移了 0.1 个像素。

cxt.translate(0.5, 0.5); //这里我们已经进行了 0.5 的坐标偏移

```
var hbase = 280;
for(x = 0, offset = 0; x < 200; x+=10, offset += 0.1 ){
        color = [0,0,180];
        drawLine(cxt,x, hbase+offset,x+10, hbase+offset,color);
}
```

其绘制结果如下：

可以看出，对那些夸两行像素的水平线段（垂直线段的情况类似），这里实际为某条宽度为 1px 的线段，颜色为 color(相应的背景色为 color1)，其纵坐标为 a.b，a 为坐标的整数部分，b 为小数部分，canvas 会对第 a 行和第 a+1 行均填充颜色。经过取色检查，我们发现，其中第 a 行的颜色为(1-0.b)*color + 0.b * color1，第 a+1 行为 0.b * color + (1-0.b)*color1。

5. 附录：

5.1 配置文件格式：

配置文件为 Javascript 文件，文件名形如*.js

可以通过如下方式引入：

```
<script src="path/to/the/config/file.js"></script>
```

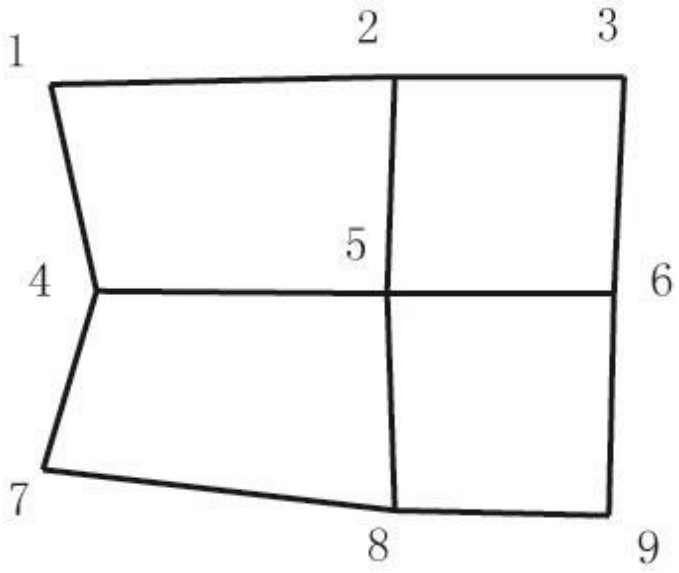
所有信息以 JSON 形式提供，文件内容如下：

```
//画布，也就是 Canvas 的大小
var canvasSize = {"maxX": <最大 X 坐标>, "maxY": <最大 Y 坐标>};

//顶点数组，保存了每个顶点的位置坐标（x,y,z），
var vertex_pos = [
    //顶点 0 位置
    [<X 坐标>  <Y 坐标>  <Z 坐标>],
    //顶点 1 位置
    [<X 坐标>  <Y 坐标>  <Z 坐标>],
    //顶点 2 位置
    [<X 坐标>  <Y 坐标>  <Z 坐标>],
    //…..
];

//顶点颜色数组，保存了上面顶点数组中每个顶点颜色信息[r,g,b]
var vertex_color = [
    //顶点 0 的颜色
    [<R 通道值>,  <G 通道值>,  <B 通道值>],
    //顶点 1 的颜色
    [<R 通道值>,  <G 通道值>,  <B 通道值>],
    //顶点 2 的颜色
    [<R 通道值>,  <G 通道值>,  <B 通道值>]
];

//四边形数组，数组中每个元素表示一个四边形，其中的四个数字是四边形四个顶点的 index，index 对
```

应于 vertex 数组，例如 vertex[polygon[2][1]]表示第二个多边形的第 1 个顶点的坐标

```
var polygon = [
    //四边形 1 的顶点信息
    [<顶点 1 编号>, <顶点 2 编号>, <顶点 3 编号>, <顶点 4 编号>],
    //四边形 2 的顶点信息
    [<顶点 1 编号>, <顶点 2 编号>, <顶点 3 编号>, <顶点 4 编号>],
    //四边形 3 的顶点信息
    [<顶点 1 编号>, <顶点 2 编号>, <顶点 3 编号>, <顶点 4 编号>],
    //四边形 4 的顶点信息
    [<顶点 1 编号>, <顶点 2 编号>, <顶点 3 编号>, <顶点 4 编号>]
];
```

配置文件范例（标号减一才是实际数组中的下标）：



配置文件名：config.js

文件内容：

```
//画布的大小
var canvasSize = {"maxX": 1024, "maxY": 768};

//数组中每个元素表示一个点的坐标[x,y,z]，这里一共有 9 个点
```

```
var vertex_pos = [
    [0, 0, 0],
    [700, 0, 0],
    [1000, 0, 0],
    [100, 400, 0],
    [600, 450, 0],
    [1000, 400, 0],
    [50, 650, 0],
    [700, 700, 0],
    [1000, 700, 0]
];

//顶点颜色数组，保存了上面顶点数组中每个顶点颜色信息[r,g,b]
var vertex_color = [
    [0, 0, 255],
    [0, 255, 0],
    [0, 255, 255],
    [255, 255, 0],
    [0, 255, 255],
    [0, 255, 0],
    [0, 255, 0],
    [0, 200, 100],
    [255, 255, 0]
];

//四边形数组，数组中每个元素表示一个四边形，其中的四个数字是四边形四个顶点的 index，例如
vertex[polygon[2][1]]表示第三个多边形的第 2 个顶点的坐标
var polygon = [
    [0, 1, 4, 3],
    [1, 2, 5, 4],
    [3, 4, 7, 6],
    [4, 5, 8, 7]
];
```

5.2 Porject1 - part1 中用到的 Cavas 基本操作方法。

```
//canvas 标签基本形式：
    <canvas id="myCanvas" width="600" height="400" style="border:1px solid #c3c3c3;">
        Your browser does not support the canvas element.
    </canvas>

//canvas 绘图方法：
//html5 的 canvas 本身不具有绘图功能，我们需要通过 Javascript 来动态的在 canvas 上面绘制图形
//在绘制之前需要获取 canvas 的上下文 context, 之后可以通过调用 context 的各种方法进行图形绘制
    var c=document.getElementById("myCanvas");
    var cxt=c.getContext("2d");

//canvas 提供的绘制函数中没有直接绘制单独 point 的方法，通常采用如下方法代替
//该方法在绘制大量点的时候速度较慢，建议使用绘制线段的方法来绘制连续的点
    //该函数在一个 canvas 上绘制一个点
    //其中 cxt 是从 canvas 中获得的一个 2d 上下文 context
    //      x,y 分别是该点的横纵坐标
    //      color 是用于表示颜色分量的数组，例如[r, g, b]

    function drawPoint(cxt,x,y, color)
    {
        cxt.beginPath();
        cxt.strokeStyle = color;
        cxt.moveTo(x,y);
        cxt.lineTo(x+1,y+1);
        cxt.stroke();
    }

//canvas 绘制  线段的方法:  从(x1,y1)到(x2,y2)的线段

    function drawLine(cxt,x1,y1,x2,y2,color){
        cxt.beginPath();
        cxt.strokeStyle ="rgba("+color[0] + "," +color[1] + "," + color[2] + "," + 255 + ")" ;
        cxt.lineWidth =2;
        cxt.moveTo(x1, y1);
        cxt.lineTo(x2, y2);
        cxt.stroke();
    }
//注意，本次 pj 不允许使用 fillrect，fill，rect 等直接绘制矩形和多边形的函数
```