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：2018-03-22

**Deadline：2018-04-15**

**负责TA：朱天乐、王思杰**

**email:**[**17212010053@fudan.edu.cn**](mailto:17212010053@fudan.edu.cn)

[**17212010036@fudan.edu.cn**](mailto:17212010036@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testBrowser.html进行测试）

四边形网格扫描转换

# 功能描述

## 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：

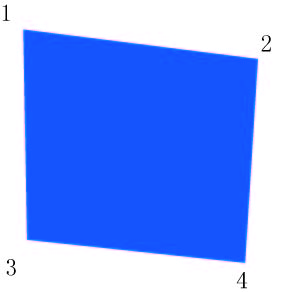


图1：颜色填充（四边形的填充颜色为蓝色）

## 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

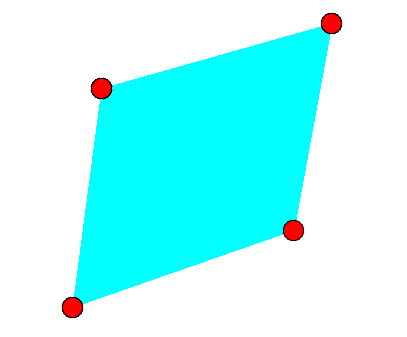
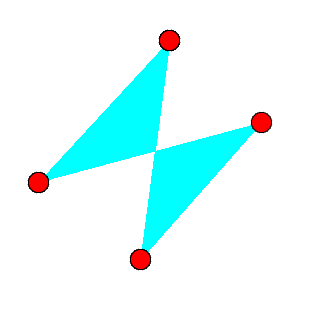
 

图3：实际显示范例

# 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：
   1. 首先实现单一四边形的扫描转化
   2. 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。
   3. 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
   4. 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。

# 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。

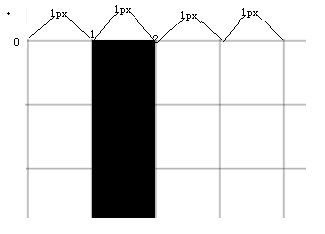
注意事项：

1. 代码具有可读性：
   1. 所有代码有合理的缩进
   2. 类，函数，变量需合理命名
   3. 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。

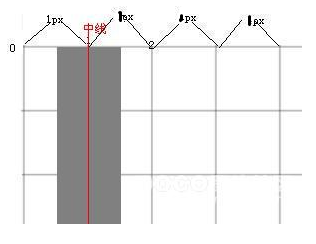
# Canvas中绘制宽度为1px线段的问题

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。

# 附录：

## 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

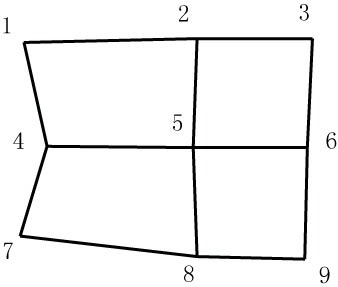
[<顶点1编号>, <顶点2编号>, <顶点3编号>, <顶点4编号>],

//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\_pos[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]

];

## 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**