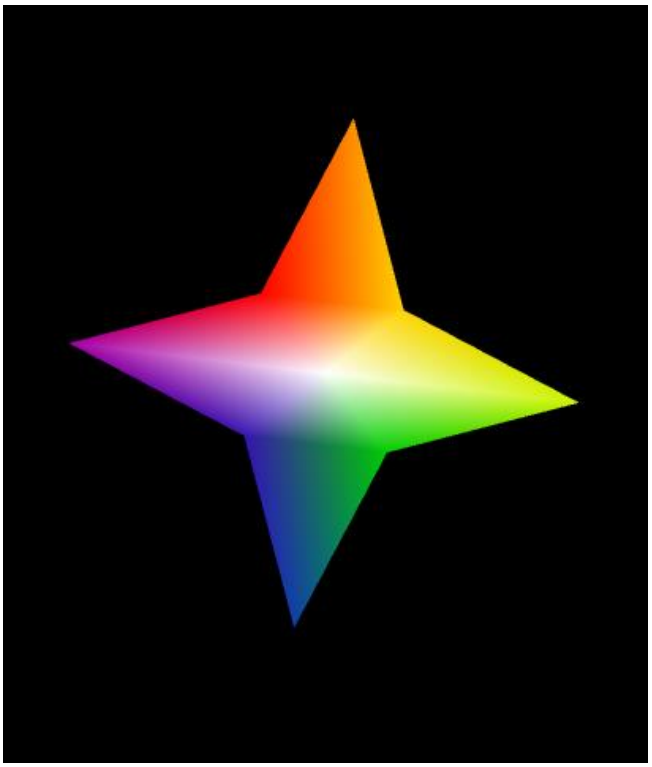


计算机图形学

PROJECT 2

WebGL 编程基础

时间节点：

发布日期：2023-03-25

Deadline： 2023-04-29

负责 TA

贺 威

秦 鹏

刘航海

许安陈

Project2 WebGL 绘制可交互式多边形网格与简单动画

项目目的

从 Canvas 图形绘制过渡到 WebGL 图形绘制，掌握 WebGL 基本编程方法，理解 WebGL 中 Shader 原理，与模型变换原理。

项目要求

实现四边形网格交互式编辑：要求搭建 WebGL 编程环境，通过 WebGL 编程接口，在浏览器中绘制可交互的四边形网格，进行颜色的渐变填充，并且实现一个小动画。

使用的语言

Html5, JavaScript, WebGL

开发与测试环境

硬件：带有支持 OpenGL ES2.0 显卡的计算机

软件：支持 HTML5, JavaScript 和 WebGL 的浏览器，建议选择最新版本的 Google Chrome 浏览器，也可以选择 Internet Explorer 11+或者最新版本的 Firefox，Safari 等。（可以使用本次 Project 附带的 testBrowser.html 进行测试）

WebGL 绘制可交互式多边形网格与简单动画

1. 功能描述

1.1 四边形网格绘制及颜色填充

1.1.1 四边形网格信息

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、每个顶点的颜色，以及四边形顶点连接关系等信息，并用利用这些信息绘制图形。

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。

(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：

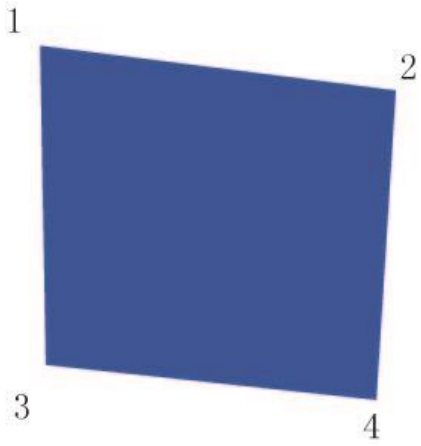
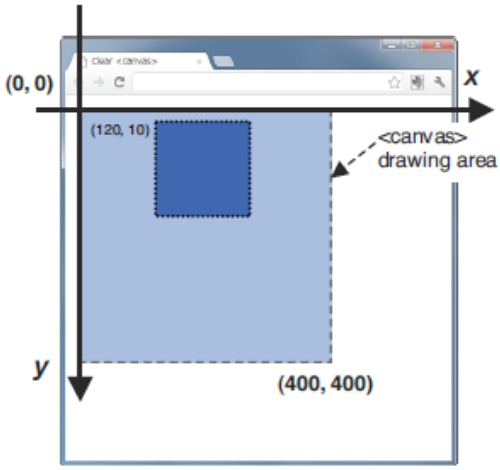


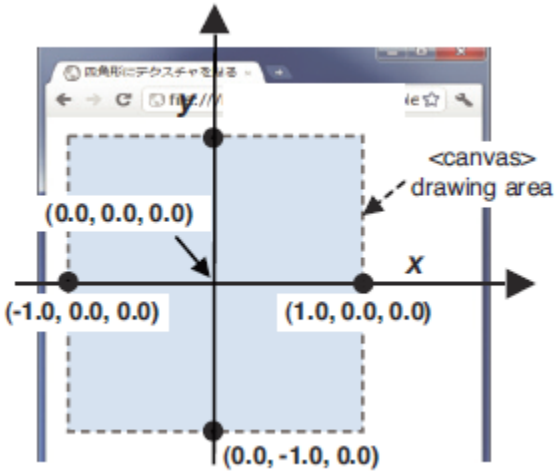
图 1：颜色填充（四边形的填充颜色为蓝色）

1.1.2 Canvas 坐标系转化为 WebGL 坐标系

Canvas 坐标系同学们应该已经非常熟悉了，它的原点在画布的左上角，x 轴的正方向水平向右，y 轴正方向垂直向下。而 WebGL 虽然是以 Canvas 为载体，但是具有不同的坐标系，它类似于同学们高中数学中常用的坐标系，坐标原点在画布中央，x 轴水平向右范围(-1.0~1.0)，y 轴垂直向上(-1.0~1.0)，实际上它还有 Z 坐标轴，从屏幕内侧指向外侧（右手坐标系）。因此，我们通常还需要进行坐标转换，本次 Project 中需要在使用前进行坐标转换。如下图所示：



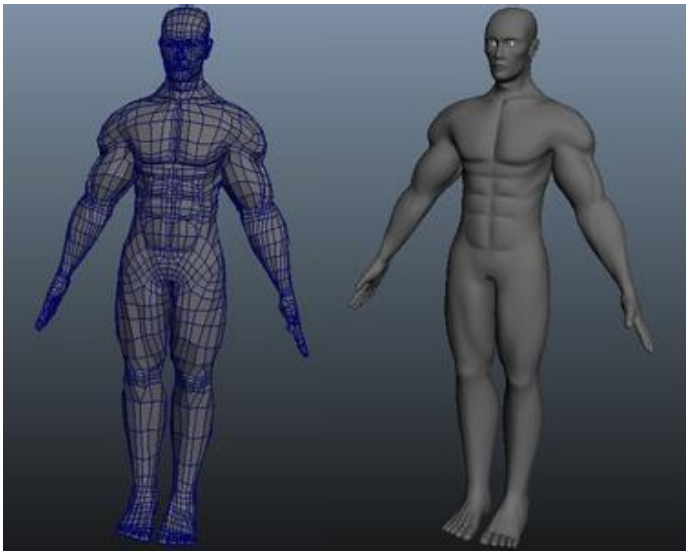
Canva 坐标系



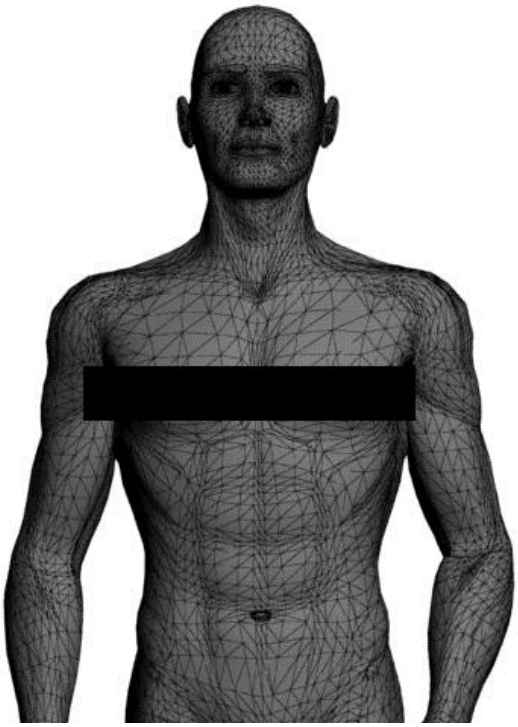
WebGL 坐标系

1.1.3 四边形网格转化为三角形网格

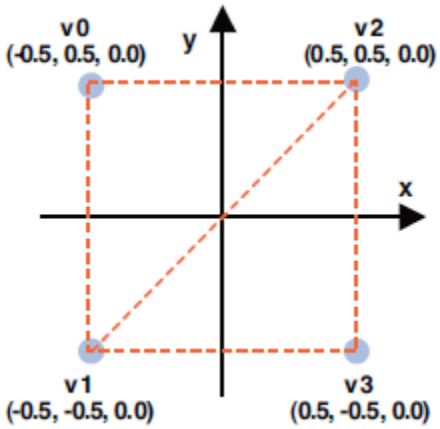
大多数情况下，美工人员在构建 3D 模型的时候，都是以四边形网格为基础进行编辑的，这样比较符合人类的认知，如下图所示：



但是在 WebGL 以及 OpenGL 等图形绘制技术当中，任何复杂的图形都必须由一些简单的基本形状组成的，其中包括点，线和三角形，这样有利于硬件加速等。如下图所示：



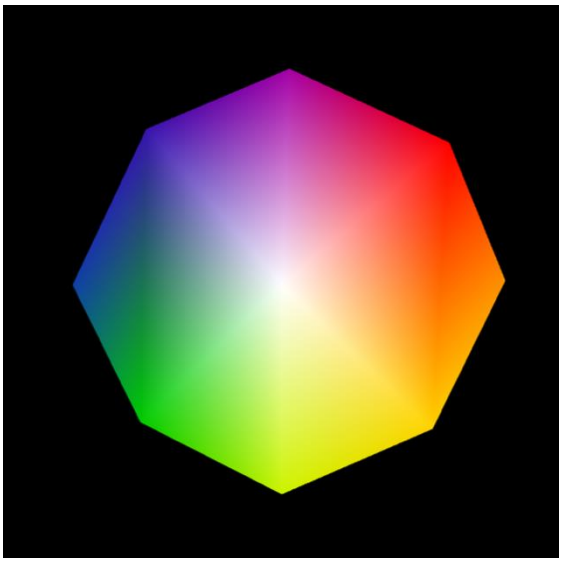
因此，如果我们得到的是四边形网格，我们必须先将四边形网格转换为三角形网格(在本次 Project 中给出的也是四边形网格,因此也需要同学们手动转化)，方法非常简单，连接四边形的一条对角线即可，这里需要特别注意的是剖分后的三角形顶点需要按顺时针方向排列（顺逆信息隐含了三角形的正反面），如下图所示：



1.2 颜色渐变

在 Project 1 中，绘制四边形只要求同学们使用四边形第一个顶点的颜色作为四边形整体的颜色。而在本次 Project 中，四边形的每个顶点都有自己的颜色，同学们需要据此绘制出具有渐变效果的四边

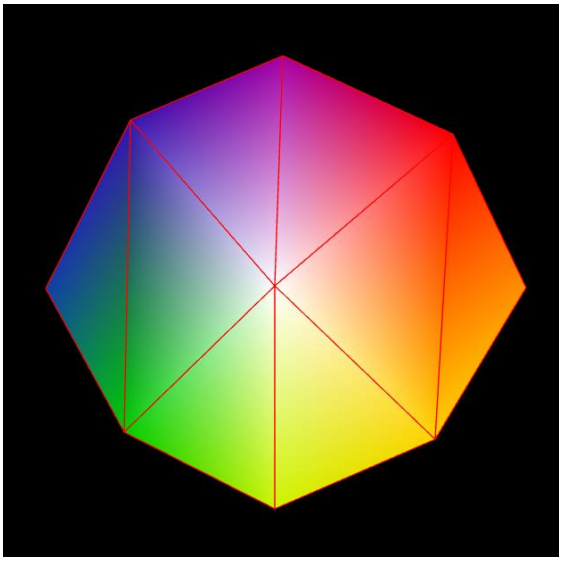
形，如文档封面所示或下图：



渐变效果

1.3 边框绘制

为了显示三角形剖分效果，以及方便之后的顶点编辑，我们需要能够显示和隐藏每个三角形的边框，显示和隐藏边框的操作通过键盘实现，要求：程序启动后，默认显示边框，初次按下键盘上 B 键显示边框，再次按下后隐藏边框。显示边框的效果如下图所示：



显示剖分后的三角形边框

1.4 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绘制，从而交互实现四边形网格动态绘制。

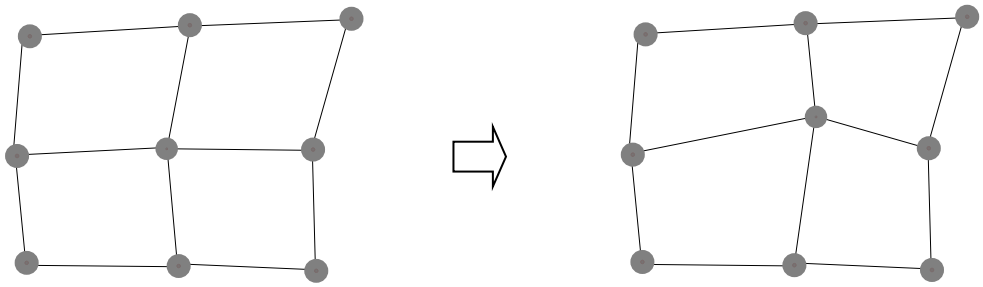


图 2：四边形顶点拖动

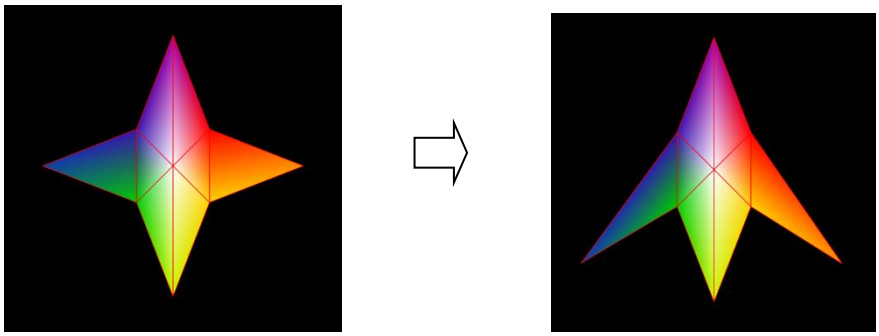


图 3：实际显示范例

1.5 模型变换与简单动画

在实现了上述功能之后，我们需要同学们通过 WebGL 的矩阵模型变换，制作一个简单的帧动画。当你编辑完自己的四边形网格之后，按下键盘上的 T，即可开启动画，动画效果要求：你编辑后的网格以每秒钟 45 度的速度逆时针，并且在旋转的过程中逐渐变小直到变为原来的百分之 20，然后逐渐变大到原本的百分之 100，然后再变小如此交替，变换的比例为每秒钟 0.2。动画放映期间，按下 T 可以暂停，再次按下 T 可以继续播放。（具体效果查看 sample）

1.6 编辑状态与动画状态的切换

用户可以通过键盘按钮 E 来进入编辑状态，通过 T 进入动画状态并控制暂定/启动。要求：程序启

动后默认进入编辑状态，允许用户对网格进行编辑。按下 T 后开始播放动画，并且可以暂停动画。当用户按下键盘上的 E 时，回到编辑状态。在任何时候，用户按下键盘上的 B 时，可以显示/隐藏网格边框。

2. 实现步骤（建议）

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，项目目录下最终结果 Sample 文件夹中有本次项目的结果范例（不可抄袭这里的代码）。
4. 学习 WebGL 基本编程方法，阅读[Addison-Wesley Professional] WebGL Programming Guide 这本教材的前五章内容，建议结合教材的 sample code，参考 TA 给的 PPT。
5. 以“建议参考的 sample”目录中的“HelloQuad.js”为基础实现网格的绘制。首先，从配置文件中读入顶点坐标信息（canvas 坐标系），转化到 WebGL 坐标系。然后从配置文件中读入四边形网格顶点序列信息，然后剖分为三角形网格信息，通过上述信息构建 VertexBuffer，绘制出纯色的四边形。
6. 结合 Project 1 的内容，实现网格的拖拽编辑，这里每次重新绘制网格的时候，需要重新生成 vertexBuffer，因为顶点数据已经发生了改变。
7. 在上述代码的基础上，结合“建议参考的 sample”目录中的“ColoredTriangle.js”中的代码实现渐变效果。从配置文件中读出每个顶点的颜色信息，和之前的位置信息一起构建 VertexBuffer。修改 VertexShader 和 FragmentShader 的代码，利用 Varying variable 实现颜色渐变绘制。
8. 结合“建议参考的 sample”目录中的“HelloTriangle_LINE_LOOP.js”中的代码实现三角形网格边框的绘制，这里有两种实现方式，一种是单独实现一个简单的用于绘制线条边框的 shader，在绘制完渐变图形后，切换到用于绘制边框的 shader，重新绑定 shader 数据后开始绘制边框。另一种方式是，在现有 shader 中添加一些布尔值形成分支逻辑，用同一个 shader 分两次绘制渐变图案和边框。
9. 结合“建议参考的 sample”目录中的“RotatedTriangle_Matrix4.js”中的代码实现模型变换。

修改你的 shader，向其中添加一个 matrix 类型的 uniform 变量，用来保存模型变换矩阵，并与 position 相乘，实现顶点的坐标变换。在 JavaScript 代码中，使用 matrix 相关函数（cuno-matrix.js 库）创建变换矩阵，并传入 shader 进行绘制，以此来实现网格的缩放和旋转。

10. 通过 requestAnimationFrame 函数来实现逐帧动画，在每一帧中稍稍修改变换矩阵，使得图像在每一帧之间连续变换，产生动画效果。可结合“建议参考的 sample”目录下的 RotatingTranslatedTriangle.js 进行学习。
11. 加入键盘事件相关代码，通过键盘控制: 1.编辑状态与动画状态 之间的切换. 2. 动画的播放和暂停。3.是否显示边框。键盘事件的代码可以参考 LookAtTrianglesWithKeys.js 文件。
12. 如果你是边学习，边添加新功能，完成的时候你的代码结构可能会比较乱，也会出现一些重复的代码，这时候重构一下你的代码是很不错的选择。
13. 备份你的代码，然后自己做实验，尝试不同的变换方式，变换顺序，更加丰富的 shader 功能，尝试纹理（贴图）等。
14. 测试并调优你的代码。
15. 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。
16. 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 elearning 作业目录下 PJ2 中。

3. 项目要求及注意事项

本阶段需要完成以下功能：

1、 从配置文件中读取画布大小和图形信息（包含顶点位置、顶点颜色和顶点之间的连接关系）并通过 WebGL 绘制。
2、 鼠标拖动编辑网格
3、 三角形网格边框的绘制
4、 简单的帧动画
5、 键盘控制编辑和播放状态，动画的暂停，是否显示边框等。

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

d) 功能模块合理的划分

2、 WebGL 默认后绘制的图形会遮挡住先绘制的图形（3D 场景绘制会使用深度缓冲等实现近处物体遮挡远处物体）

3、 在引入 JS 库的时候，注意修改你的 html 文件，以及使用正确的相对路径

4、 For in 语句迭代整数数组的时候，迭代得到的元素可能是字符串而不是整数

5、 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. 附录：

4.1 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>]
];
```

本次配置文件格式与 Project 1 中的完全相同。