****

**计算机图形作业（三）报告**



**学 院 软件学院**

**专 业 软件工程**

**学 号 3017218135**

**姓 名 全康连**

目 录

目录

**摘要3**

**主体介绍3**

**结果讨论8**

**收获和建议10**

一．摘要

1. 项目名称：Assignment3---Bezier surface
2. 操作简介
3. 使用25个控制点绘制Bezier 曲面
4. 使用细分曲面控制着色器TCS设置曲面细分级别
5. 使用细分曲面计算着色器TES计算顶点的坐标和纹理
6. 通过键盘可以控制曲面的光滑度
7. 程序能够支持线条框架绘制模式
8. 给Bezier曲面添加纹理

二．主体介绍

1. 程序简介

该程序完成了使用25个控制点，根据Bezier曲面绘制算法，使用TCS，TES着色器绘制了Bezier曲面，并给该曲面添加了纹理；用户可以通过键盘控制曲面的光环程度，移动摄像机视角观察曲面

1. 算法
2. Bezier曲线

Bezier曲线是应用于二维图形应用程序的数学曲线，由线段和节点组成，节点是可拖动的控制支点，线段像可伸缩的皮筋。

**线性公式**：给定点p0, p1，线性Bezier曲线是两点间的直线，由下列公式给出：

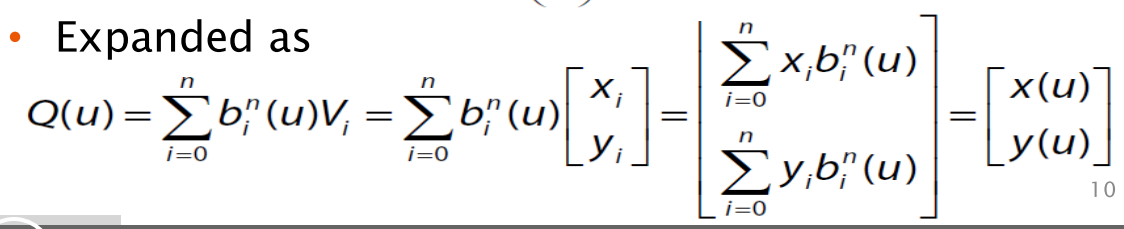
该公式等同于线性插值

**二次方公式**：二次方Bezier曲线由给定点p0,p1,p2的函数B(u)给出

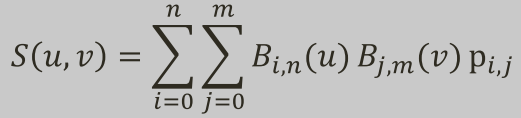
**三次方公式**：p0,p1,p2,p3四个点在平面或三维空间里定义了三次方Bezier曲线，

由此可推出多个控制点Bezier曲线的一般参数公式为

将控制坐标展开p(x, y)，可的矩阵形式为



在三维空间中，我们需要定义二维控制点Bezier曲线参数，



其中B(u), B(v)公式相同

1. 实现细节
2. Glsl文件定义

要绘制Beizier曲面，只使用OpenGL提供的基本图元是不能实现的，我们需要自定义图元类型patch，输入到OpenGL中进行绘画，这需要用到细分曲面控制着色器TCS，细分曲面计算着色器TES

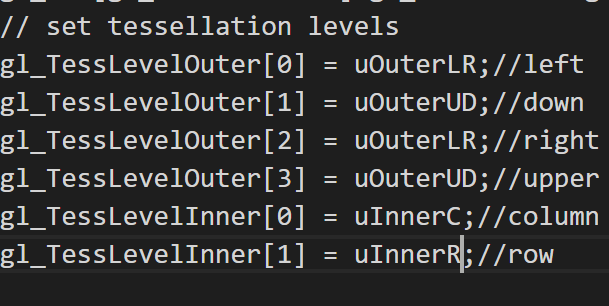
TCS：在该着色器中我们需要定义图元的细分级别，并将patch数据输入到TES中。该着色器的输入为一系列的控制点，输出为patch图元和细分级别；结构体gl\_in[]定义了输入顶点的结构如下：



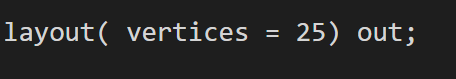
TCS中定义结构体数组gl\_out[]将顶点信息输出到下一着色器中



并定义patch的细分级别，其中gl\_TessLevelOuter定义了patch的边界细分数，共有上下左右四个边界；gl\_TessLevelInner定义了patch内部的细分数，分别为列数和行数参数



定义输出patch的顶点个数，程序中使用了25个控制点

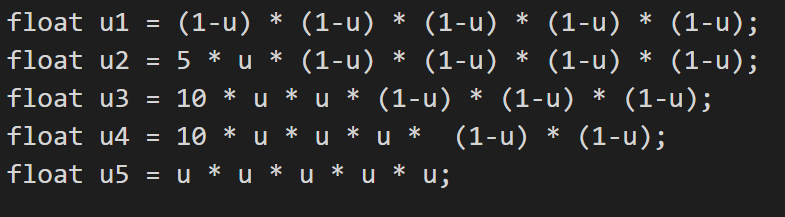


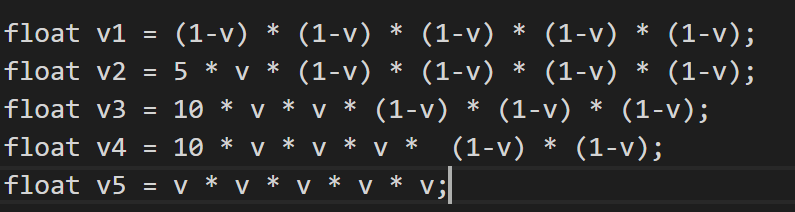
TES：在着色器中需要使用上一着色器输入的gl\_TessCoord和patch计算出顶点的坐标和纹理信息

首先定义输入的图元参数，指定图元生成域（三角形，四边形，直线），细分曲面坐标空间（equal\_spacing, fractional\_even\_spacing, fracing\_odd\_spacing），生成图元的面部朝向（ccw逆时针，cw顺时针）

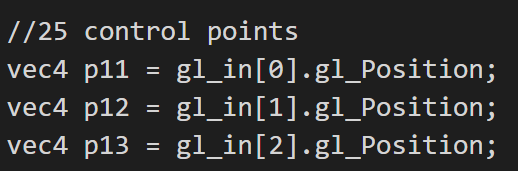


使用Bezier曲线方程对顶点位置信息进行相应计算，程序使用了25个控制点，则顶点方程为5x5的矩阵，所以顶点参数方程为五次方方程

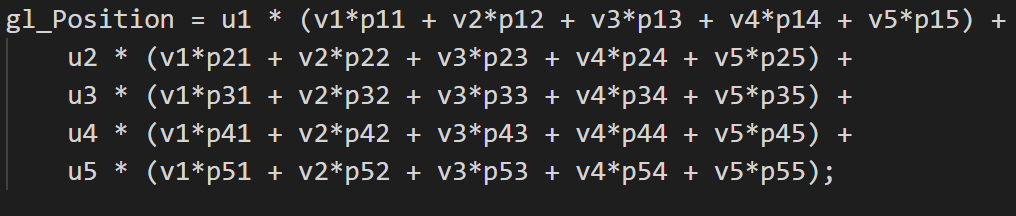




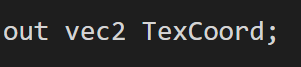
使用gl\_in[]数组依次读取25个控制点坐标



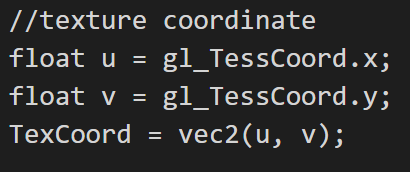
最后顶点坐标的计算公式为



定义纹理坐标输出



纹理坐标计算如下，其中着色器会将顶点着色器中的输入顶点规范为0~1之间的范围



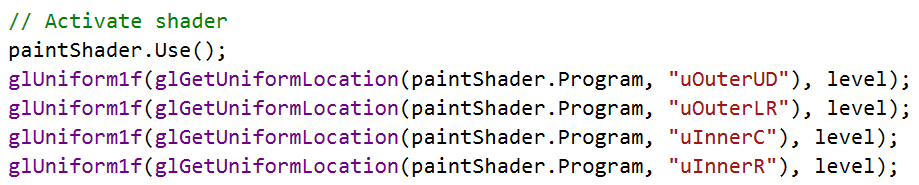
1. 绘制Bezier曲面

顶点绑定过程和之前无异，但在绘制的过程中我们需要使用glPatchParameteri()函数去指定patch图元的参数，该函数使用如下

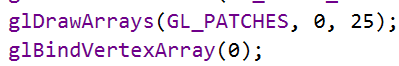
程序中定义如下：



将曲面细分级别设置给shader，其中的level为全局变量细分曲面级别，用户可以通过键盘控制其大小

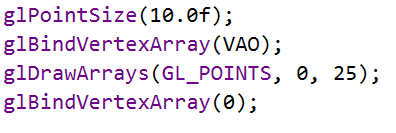


使用glDrawArrays()函数时，要制定绘画的类型为patch图元

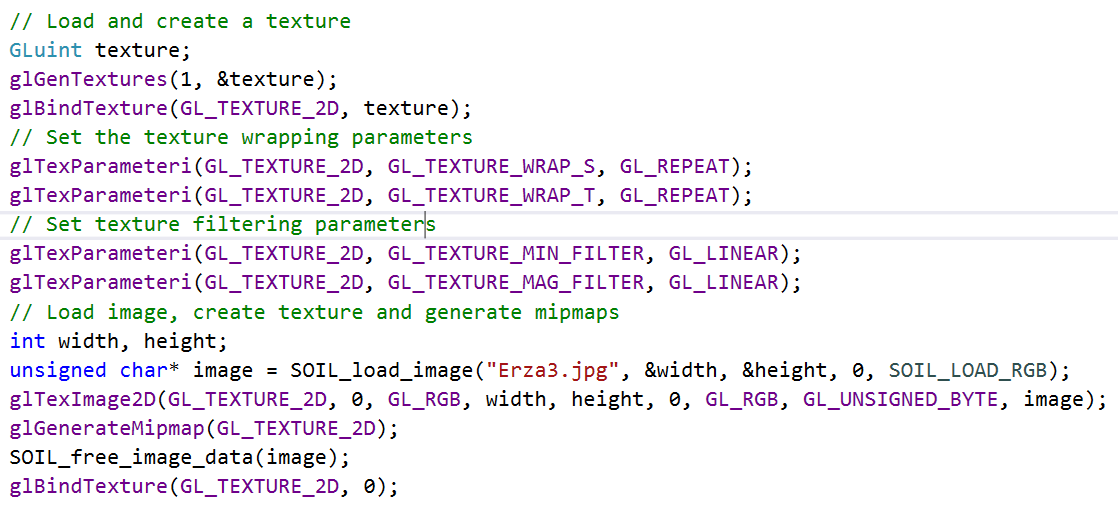


1. 绘制控制点

使用glDrawArrays()函数进行控制点绘制时，制定绘制图元为点类型；并使用glPointSize()函数设定控制点的大小



1. 给曲面添加纹理

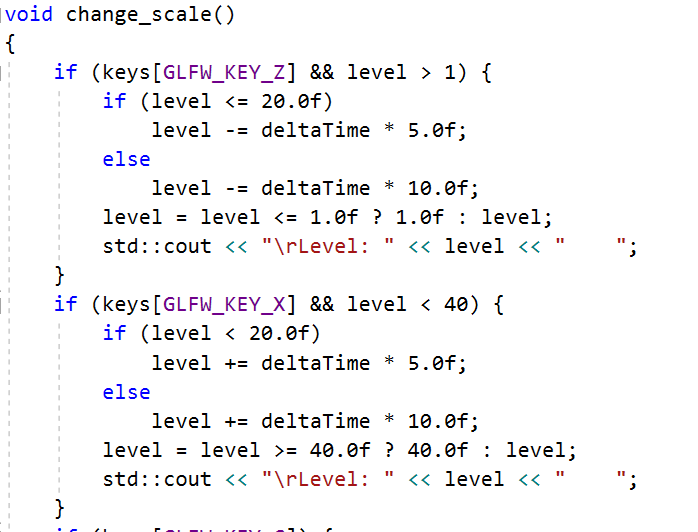


1. 添加键盘控制

**控制曲面光滑度**：定义全局变量level，用以控制曲面细分的级别

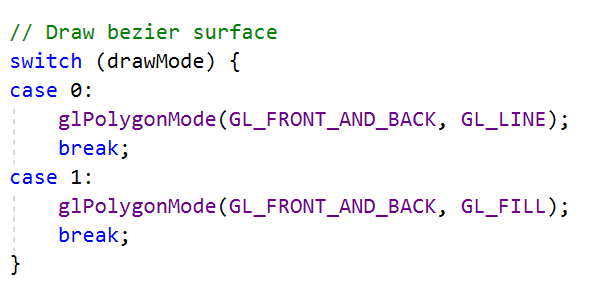


定义函数change\_scale()检查用户键盘输入，改变level级别的大小，用户通过X键增加曲面光滑度，通过Z键减小曲面光滑度

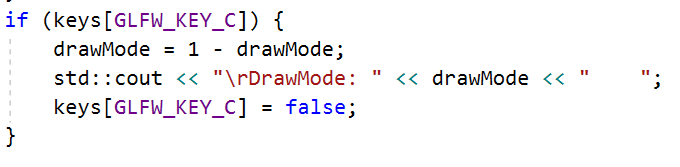


**增加线条框架绘制模式**：定义全局变量drawMode，来判断在帧数循环绘制过程中是使用纹理填充曲面还是显示线条框架模式；其中drawMode==0，使用线条框架模式，drawMode==1，使用纹理填充曲面





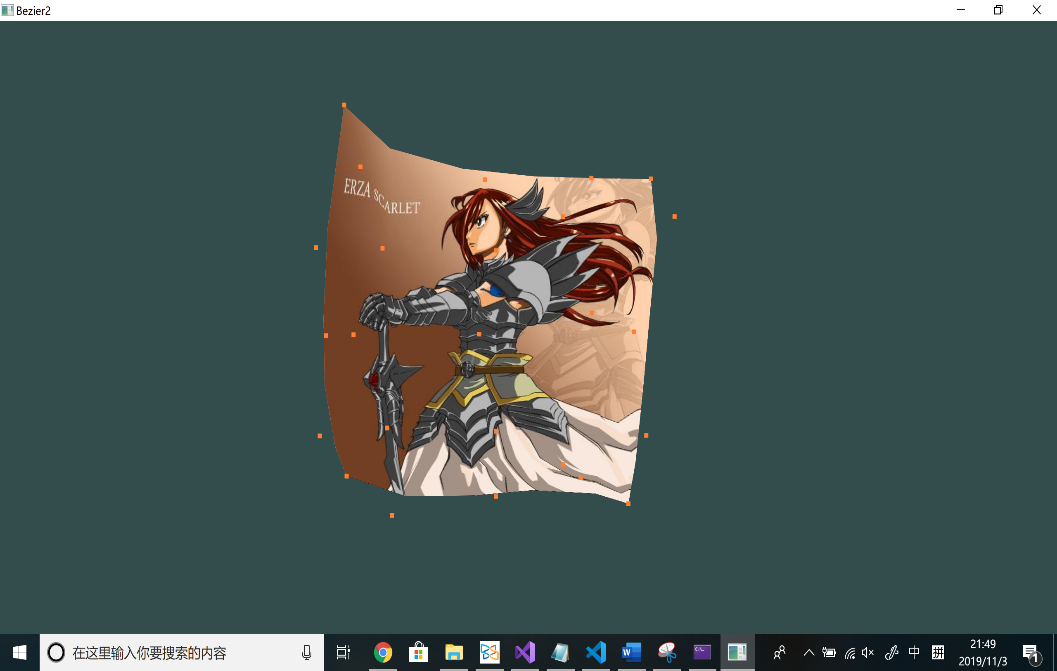
在change\_scale()函数中，用户可以通过C键控制曲面的显示样式



三．结果讨论

1. 结果

运行程序，可以Bezier曲面如下，其中默认显示样式为纹理填充



通过相应的键盘控制，可以改变曲面的光滑程度和切换曲面的显示样式（点击视频播放）



1. 讨论

在完成程序的过程中遇到了一些问题，通过上网查询资料，参考他人代码，对问题有了自己的理解和解决方案。

1. 理解openGL细分曲面

openGl中只提供了基本的图元类型三角形，点，线，使用基本图元类型是不能满足绘制曲面的要求的。根据Bezier曲线算法，可以通过控制点绘制三维空间的曲面，改变控制点的位置，从而使得曲面改变其形状。实际上曲面是由多个面片组成，随着控制点增多，一个曲面的细分面片随之增加，从而使得曲面从视觉上变得愈加光滑。openGL中提供了TCS和TES着色器来完成细分曲面的功能，用户需要使用自定义图元patch，经过一系列的规定操作步骤，来完成Bezier曲面的绘制，参考教程<https://www.cnblogs.com/zenny-chen/p/4280100.html>，具体TES和TCS操作参见“实现细节-->glsl文件定义”

四．收获和建议

1. 课程收获

计算机图形学课程中，老师讲解清晰，实验课中代码原理实现细节讲解清楚。在课程中，老师会将实验中的基本实现步骤给出，对于本次作业有着很大的帮助。通过对老师的实验指导学习和本次实验的完成，我学习到了很多知识。

* TCS和TES的使用

本次程序结合Bezier曲线算法，使用openGL提供的TES，TCS着色器完成了细分曲面的过程，自定义了25个控制点的Bezier曲面并添加相应纹理显示。老师通过详细的代码示例和ppt指导了TCS和TES的使用。TCS为细分曲面控制着色器，其输入为一系列自定义图元的控制点，输出为控制点顶点数组gl\_out, 曲面细分级别。在着色器中定义了曲面的边界细分级别gl\_TessLevelOuter, 内部行列细分级别gl\_TessLevelInner, 通过细分曲面增加曲面的组成顶点数据。TES为细分曲面计算着色器，其使用TCS的顶点输入数据gl\_TessCoord计算控制点的坐标并计算曲面组成顶点的纹理坐标

1. 项目开发收获

通过本次实验，学习了Bezier曲线算法原理，TCS, TES的使用，并将原理知识结合到实验中完成了程序要求的功能；在实验过程中收获颇丰

* Bezier曲线算法原理

Bezier曲线算法通过增加控制点的个数，并使用类线性插值算法拟合出Bezier曲线。随着控制点的增加，Bezier曲线方程控制点系数最高次数相应增加，其拟合出的曲线愈加光滑。本程序使用25个控制点进行Bezier曲面的绘制，曲面的绘制需要两个系数控制矩阵，计算原理相同，计算如下：

之后使用矩阵相乘方法将控制点位置算出，具体细节参见“实现细节🡪绘制Bezier曲面”

* TCS, TES的使用

openGL提供了这两个着色器用以给用户自定义patch图元，用以绘制更复杂的场景。TCS着色器将patch图元的一系列控制点坐标作为输入，用户通过自定义patch的细分程度将patch进行进一步细分增加其组成顶点，之后TCS着色器将patch图元和边界细分级别，图元内部细分级别输出，作为TES着色器的输入；TES着色器使用该输入信息，结合Bezier曲线算法对顶点系数矩阵进行计算，并于顶点矩阵进行相应的相乘计算，得到最后的控制点坐标信息和曲面的顶点位置，纹理信息，从而得到Bezier曲面

1. 课程建议

* 希望每一次实验老师可以选取比较好的作品进行展示和讲解，这样可以学习他人好的代码思维，了解到更多的计算机图形学知识
* 希望老师对于下一节课的内容可以提前告知所讲范围，方便学生的预习