

# READ ME

姓名：周宁

班级：111171

学号：20171004140

## 一、实习思路：

### 1. 将程序编译通过

大致步骤如下：

- ①是先创建一个项目，取名为 `curve_editor`（模仿老师程序的名字）。
- ②将提供的 `main.cpp` 的内容直接复制粘贴到 `curve_editor.cpp` 上。
- ③将提供的 `arg_parser.h`、`glCanvas.h` 等文件导入到程序。
- ④创建 `spline.h`、`spline.cpp`、`curve.h`、`curve.cpp`、`surface.h`、`surface.cpp` 文件
- ⑤在 `spline.h` 中创造 `Spline` 类，并且写上 **`virtual void Paint(ArgParser *args)`**、**`virtual void OutputBezier(FILE *file)`** 等函数，在 `curve.h` 中创造类 `Curve`、`BezierCurve`、`BSplineCurve`（曲线类，`curve` 继承自 `spline`，`BezierCurve`、`BSplineCurve` 继承自 `curve`），在 `surface.h` 中创造 `Surface`、`SurfaceOfRevolution`、`BezierPatch`（曲面类，`surface` 继承自 `spline`，`SurfaceOfRevolution`、`BezierPatch` 继承自 `surface`）。这一步在链接可能有问题但是将相应的函数实现，函数体写为空即可解决。

### 2. 画出 Bezier、BSpline 曲线

- ①在 `spline` 类里面，设置两个数据成员 **`num_vertices`**（`int` 类型，代表控制点个数）和 **`vertexs`**（`Vec3f` 类型数组，用来存储控制点），实现函数 **`void set(int i, Vec3f v)`**，**`set()`** 函数功能是设置一下数组内容。

- ②实现类 `Curve` 的函数 **`Curve(int num_vertices)`**、**`void Paint(ArgParser* args)`**。其中构造函数 **`Curve()`** 的功能是给数据成员 **`num_vertices`** 赋值和初始化数组 **`vertexs`**，而 **`paint()`** 函数功能是简单的将控制点画出来并且连接起来。

- ③对于 `BezierCurve` 类，实现函数 **`Vec3f getBezier(Vec3f v1, Vec3f v2, Vec3f v3, Vec3f v4, double t)`** 和 **`virtual void Paint(ArgParser* args)`**。其中函数 **`getBezier`** 的功能是输入四个控制点和 `t` 能得到对应的 `Bezier` 曲线上的点。函数 **`Paint()`** 调用了 **`Curve::paint(ArgParser* args)`** 画出控制点，并且在调用 **`getBezier()`**，得到曲线上的点，其中 `t` 的取值范围为 0 到 1，插值间距为 `1/curve_tessellation`（可知 `curve_tessellation` 越大，取得点越多，就越像曲线）。值得注意的是当控制点的数量大于 4 时，需要进行分段，如：7 个点画两次，10 个点画三次。

- ④对于类 `BSplineCurve` 有了③的经验之后就比较简单了，实现函数 **`Vec3f getBSpline(Vec3f v1, Vec3f v2, Vec3f v3, Vec3f v4, double t)`** 和 **`virtual void Paint(ArgParser* args)`**，这两个函数和③函数功能差不多，区别是 **`getBSpline()`** 中得到的是 `Bspline` 曲线上的点，以及他们的分段方式也不相同，其中 `Bspline` 控制点的个数大于 4 时，采取的是 1、2、3、4 一段曲线，2、3、4、5 一段曲线的方法。

### 3、Bezier 和 BSplines 的相互转化

先看了下老师的PPT 曲线—B样条,了解一下两者系数矩阵的相互转化。具体步骤如下:

①对于类 BezierCurve, 实现函数 `virtual void OutputBezier(FILE* file)`、`virtual void OutputBSpline(FILE* file)`。对于函数 `OutputBezier()` 只需要将 `vertexs` (控制点) 挨个直接写到文件即可。对于函数 `OutputBSpline()`, 需要进行控制点相应的转化, 简单的来说起始就是一个矩阵求逆之后再进行矩阵相乘就行, 矩阵求逆的方法在 `Matrix` 类中都已经写好了, 所以只需要初始化之后调用即可。需要注意的是要考虑控制点多余 4 个时的处理, 我采用的方法是一段一段的进行转化, 每一段都是四个 Bezier 控制点, 然后转化成 BSpline 控制点。

②对于类 BSplineCurve, 思路跟①差不多, 实现函数 `virtual void OutputBezier(FILE* file)`、`virtual void OutputBSpline(FILE* file)`, 对于 `OutputBSpline` 直接输出控制点, 对于 `OutputBezier` 进行相应的转化。

### 4. 实现 Bezier 和 BSpline 曲线一般化

在 2 的时候已经实现了。思路比较简单, 就是分段进行。对于 Bezier 曲线, 对点的个数有要求, 必须是  $3*n+1$  个控制点, 每次只考虑四个控制点, 前一段的末点是后一段的起点, 如: 7 个控制点, 1234 分为一段, 4567 分为一段。相比之下 BSpline 曲线就更简单了, 只需要控制点的个数大于四即可, 每次按照窗口滑动一个控制即可, 如: 5 个控制点, 1234 一段, 2345 一段。

### 5. 完善曲线编辑器实现点的移动移动、增加、删除

①实现spline类当中的`virtual int getNumVertices();virtual Vec3f getVertex(int i); virtual void moveControlPoint(int selectedPoint, float x, float y);virtual void addControlPoint(int selectedPoint, float x, float y);virtual void deleteControlPoint(int selectedPoint);`

②`getNumVertices()` 函数比较简单, 只需要返回成员变量 `num_vertices` 即可, 函数 `getVertex(int i)` 也比较简单, 返回 `vertexs[i]` 即可

③对于 `moveControlPoint(int selectedPoint, float x, float y)`, 只需要将对应的 `vertexs[selectedPoint].Set(x, y, 0)` 即可。而对于 `addControlPoint(int selectedPoint, float x, float y)` 和 `deleteControlPoint(int selectedPoint)`, 我采取的是将对于 `vertexs` 数组, 将类型改为 `vector<Vec3f>`, 然后再对应的位置进行添加和删除即可。

④因为Bezier曲线点个数的特殊性 (必须为  $3*n+1$ ), 所以我在类 BezierCurve 重载了函数 `addControlPoint` 和 `deleteControlPoint`, 函数体为空, 使其不能添加和删除点, 但是可以移动点。

### 6. 实现 SurfaceOfRevolution 类

①在类 BezierCurve 和类 BSplineCurve 中实现 `virtual TriangleMesh* OutputTriangles(ArgParser* args)`, 对于 `OutputTriangles()`, 创建一个 `TriangleNet* m`, 然后就是采取旋转设置点的方法, 每次旋转一维后将曲线上旋转后对应的点设置到 `m` 中 (调用 `SetVertex(int i, int j, Vec3f v)`), 最后返回 `m` 即可。

②在类SurfaceOfRevolution当中设计一个数据成员curve (Curve类型的指针), 然后实现函数函数SurfaceOfRevolution(Curve\* c);virtual void Paint(ArgParser\* args); virtual void OutputBezier(FILE\* file);virtual void OutputBSpline(FILE\* file); virtual TriangleMesh\* OutputTriangles(ArgParser\* args);其中构造函数SurfaceOfRevolution(), 进行初始化, 其他函数只需要调用curve->相应函数即可。

## 7. 实现 16 个控制点的 4x4 Bezier 块

①在 BezierPatch 类中实现函数 BezierPatch();void Paint(ArgParser\*args);TriangleMesh\* OutputTriangles(ArgParser\* args);


②函数 BezierPatch () 中设置控制点数量为 16, 初始化数组长度也为 16。函数 Paint (ArgParser\*args) 中, 先将控制点画出来, 然后再将控制点四个一组画出连线。OutputTriangles () 中, 则是用到 Bezier 张量积, 16 个控制点水平四个一组得到 4 个控制点, 再由这 4 个控制点垂直得到一条 Bezier 曲线, 就这样横向纵向插值即可得到 Bezier 曲面。

## 二、实习中遇到的困难和解决方法:

1. 在vs中使用fopen等编译不通过

解决方法: 网上查阅资料后, 在项目预编译器定义中加入\_CRT\_SECURE\_NO\_WARNINGS即可。

2. 实现实现Bezier曲线和BSplines曲线之间的转换时, 并未想清楚怎么实现逆矩阵的实现  
解决方式: 查看类Matrix, 发现类中有相应的函数, 直接调用即可。此外此题具体的实现还参照了ppt上的转化。



### Bézier与B样条互相转换

● 使用基函数表示:

$$Q(t) = \mathbf{G}\mathbf{B}\mathbf{T}(t) = \text{Geometry } \mathbf{G} \cdot \text{Spline Basis } \mathbf{B} \cdot \text{Power Basis } \mathbf{T}(t)$$

$$B_{\text{Bezier}} = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$Q_1(t) = Q_2(t)$$

$$G_1 B_1 T = G_2 B_2 T_2$$

$$G_1 B_1 = G_2 B_2$$

$$G_1 = G_2 B_2 B_1^{-1}$$

$$G_2 = G_1 B_1 B_2^{-1}$$

$$B_{B-Spline} = \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

10

(参考的两种曲线相互转化的ppt)

3. 实现SurfaceOfRevolution类，假定曲线绕y轴旋转时，不懂该怎么实现旋转。

解决方法：查看ppt，发现可以转化为矩阵旋转后，就解决了。



$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

12

（就是这张ppt，将z轴类比到y轴即可，并且矩阵可以由类Matrix中静态函数Matrix MakeYRotation(float theta)提供）

4. 画出的甜甜饼中间没有空，是有底的。

解决方法：查看生成的obj文件后，发现多了很多行 (0, 0, 0, 0)，查看类 TriangleNet的构造函数和SetVertex(int i, int j, Vec3f v)后，逐渐明白了\_v\_tess的含义，以及函数SetVertex各个参数的含义。\_v\_tess是每条线的点的个数，但要比实际值小1（从0开始计算），如：每条线画10个，则传进去9。

5. 画出的花瓶并没有达到老师给的要求

解决方法：实现了类SurfaceOfRevolution中相对应的点移动的函数，然后通过增加和移动点达到相应的曲线形状后，按下s保存然后再进行obj文件生成即可画出想要的样子。

6. 画16个控制点的4x4 Bezier块并不是很明白意思

解决方法：查看ppt曲面建模后结合上课所讲述的内容，明白了Bezier张量积的含义，16个控制点，每四个控制点组成一条Bezier曲线（如：1234为1条Bezier曲线，5678为1条Bezier曲线），在相同u，可以取到四个点，将这四个点当作新的控制点，让v从0到1画出曲线，以此类推即可画出Bezier曲面。



## Bezier张量积

记号:  $CB(P_1, P_2, P_3, P_4, \alpha)$  表示控制点为  $P_i$  的 Bezier 曲线, 在参数  $\alpha$  处的值

定义张量积 Bezier 曲面为

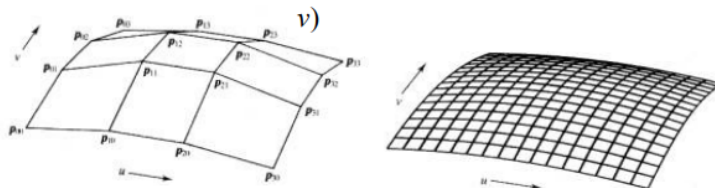
$$s(u, v) = CB(\dots$$

$$CB(p_{00}, p_{01}, p_{02}, p_{03}, u),$$

$$CB(p_{10}, p_{11}, p_{12}, p_{13}, u),$$

$$CB(p_{20}, p_{21}, p_{22}, p_{23}, u),$$

$$CB(p_{30}, p_{31}, p_{32}, p_{33}, u),$$

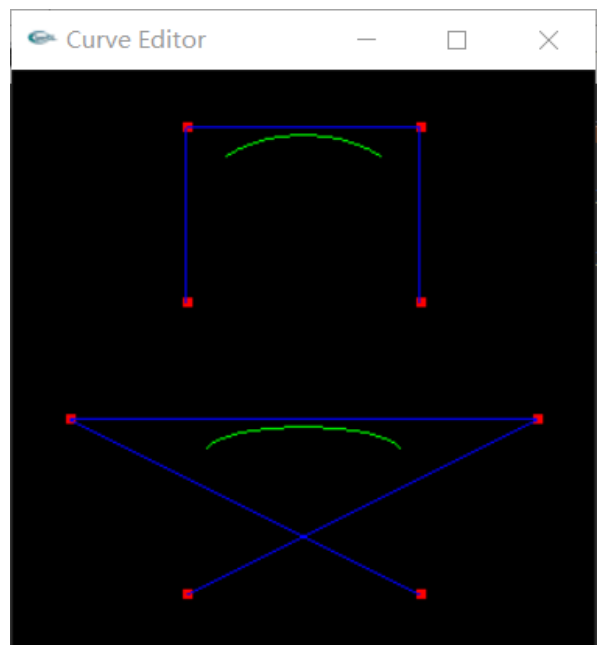
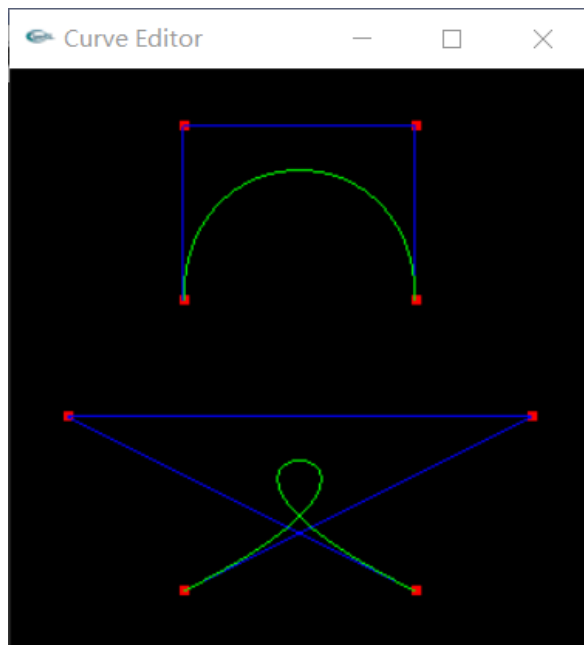


(此图为参照的ppt)

### 三、成果展示

```
curve_editor -input spline01_bezier.txt -gui -curve_tessellation 30
```

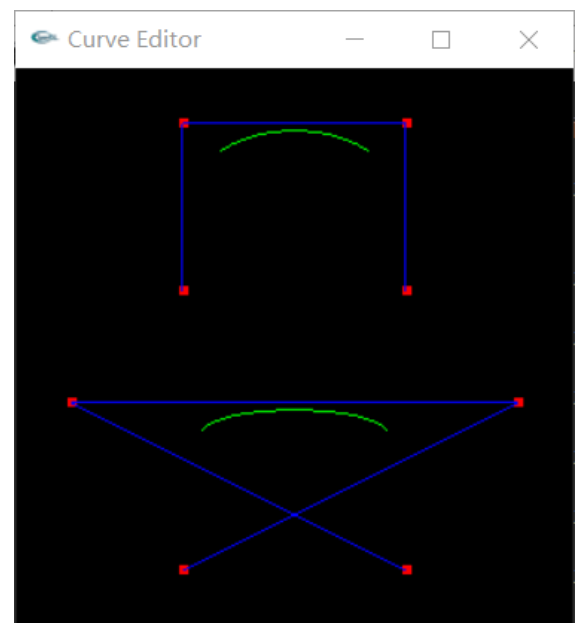
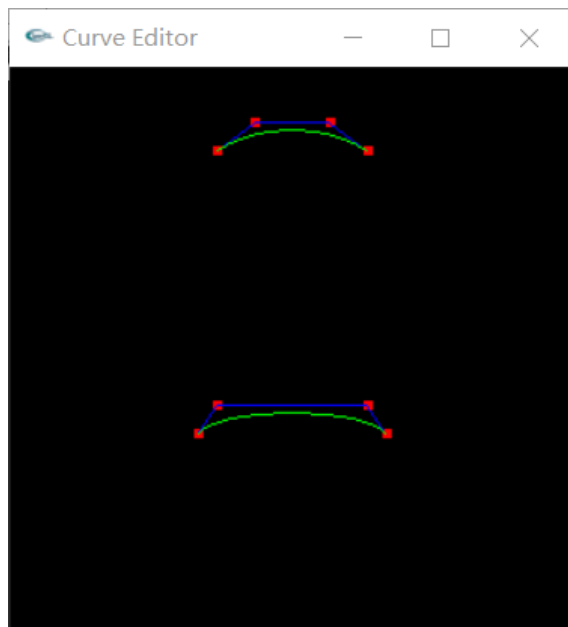
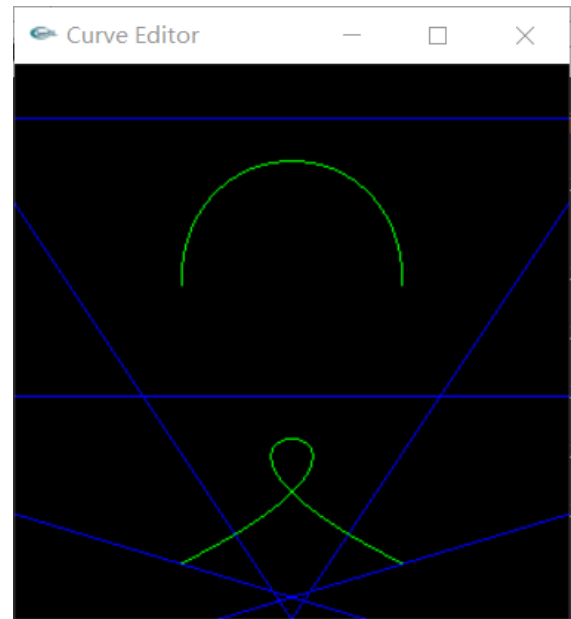
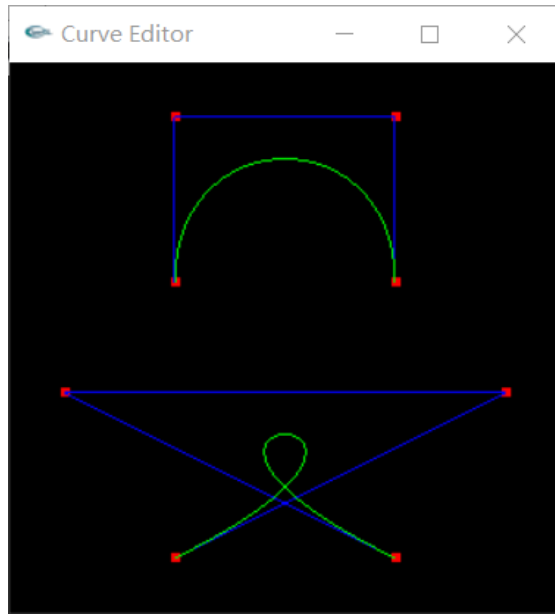
```
curve_editor -input spline02_bspline.txt -gui -curve_tessellation 30
```



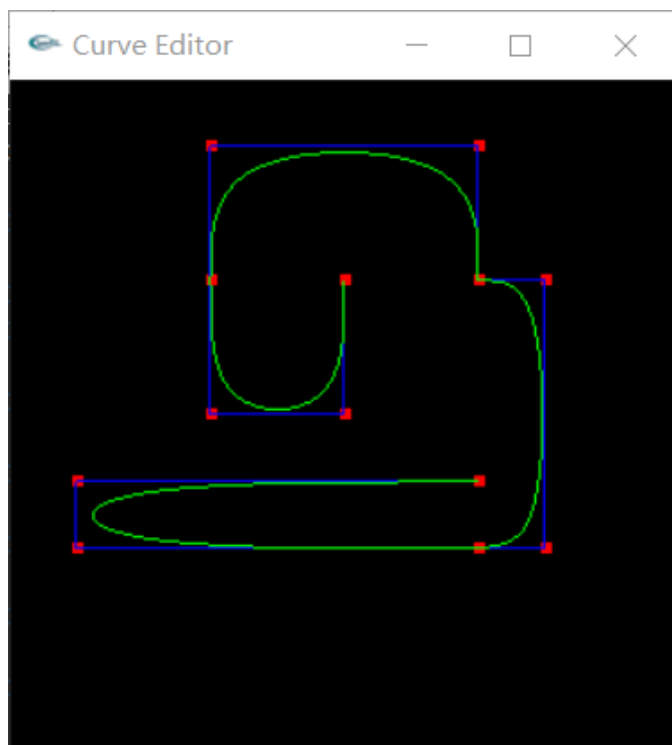
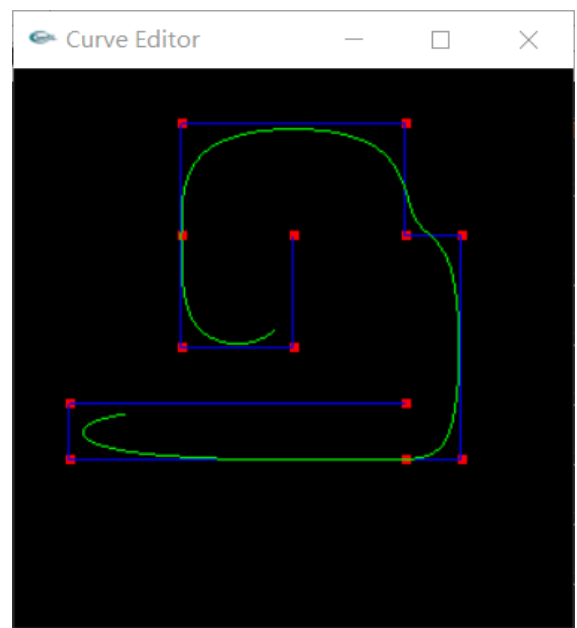
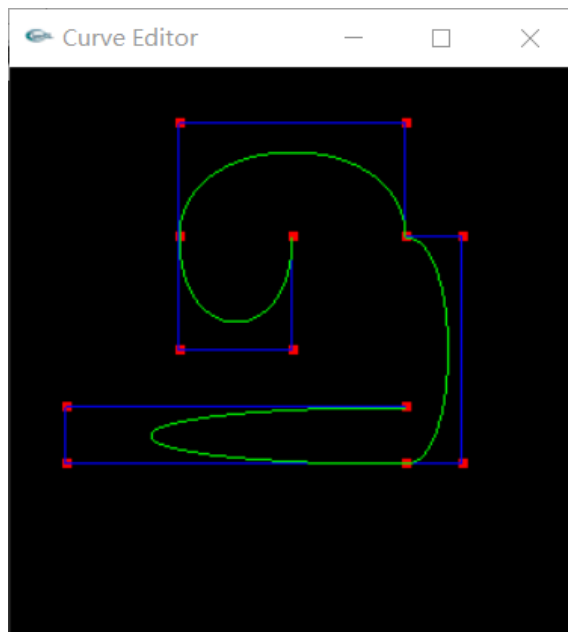
```

curve_editor -input spline01_bezier.txt -output_bezier output01_bezier.txt
curve_editor -input spline01_bezier.txt -output_bspline output01_bspline.txt
curve_editor -input spline02_bspline.txt -output_bezier output02_bezier.txt
curve_editor -input spline02_bspline.txt -output_bspline output02_bspline.txt
curve_editor -input output01_bezier.txt -gui -curve_tessellation 30
curve_editor -input output01_bspline.txt -gui -curve_tessellation 30
curve_editor -input output02_bezier.txt -gui -curve_tessellation 30
curve_editor -input output02_bspline.txt -gui -curve_tessellation 30

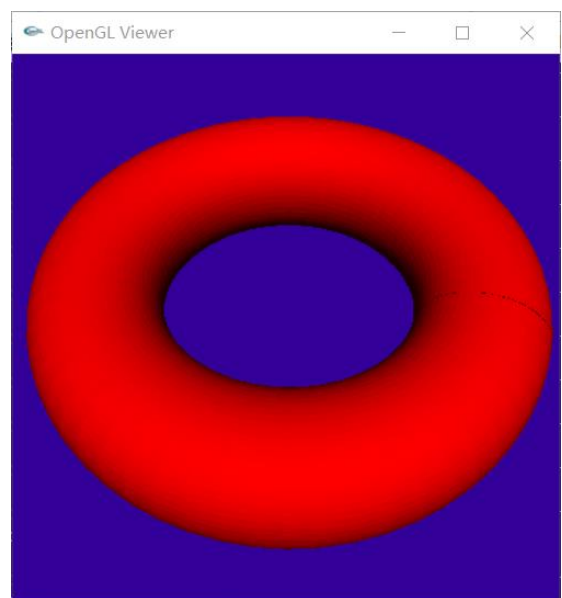
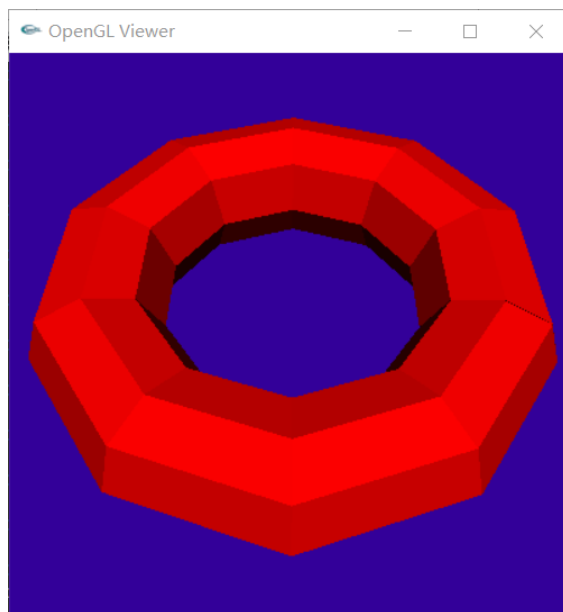
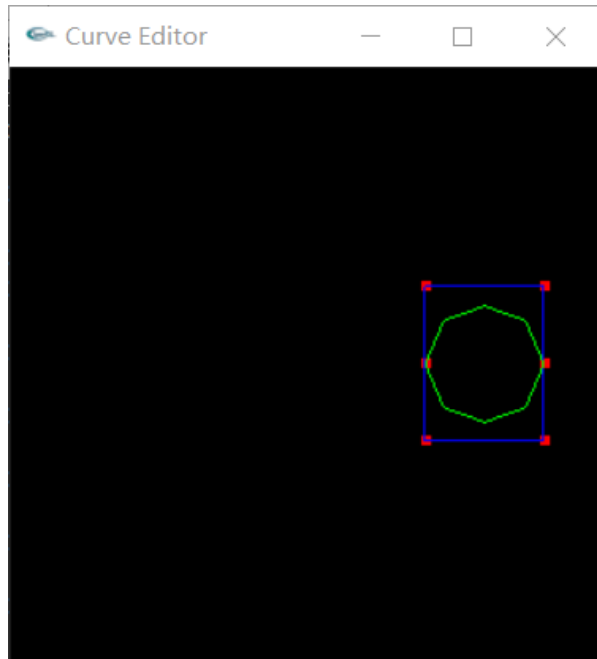
```



```
curve_editor -input spline03_bezier.txt -gui -curve_tessellation 30
curve_editor -input spline04_bspline.txt -gui -curve_tessellation 30
curve_editor -input spline05_bspline_dups.txt -gui -curve_tessellation 30
```

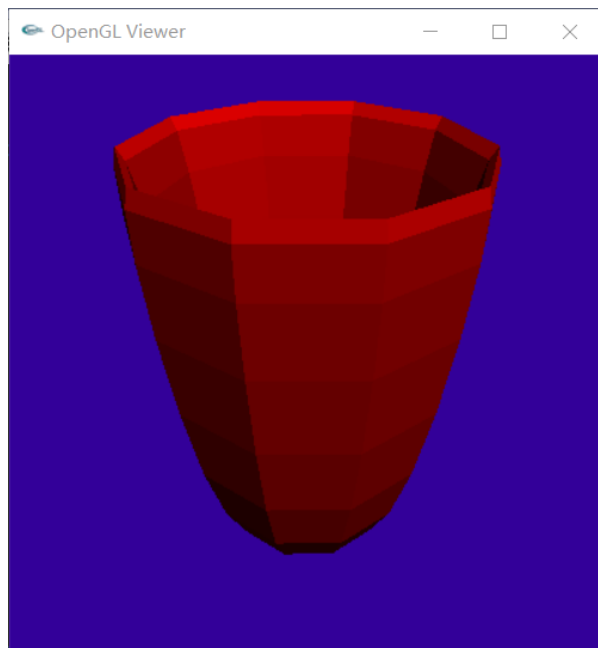
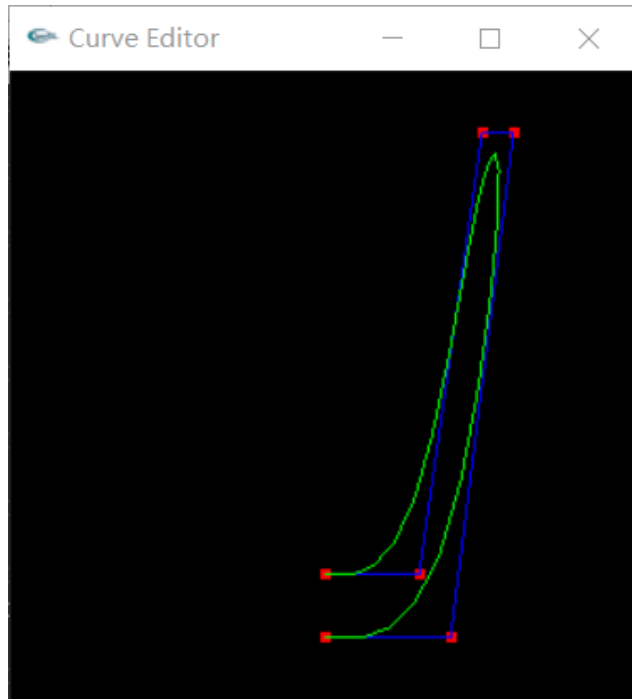


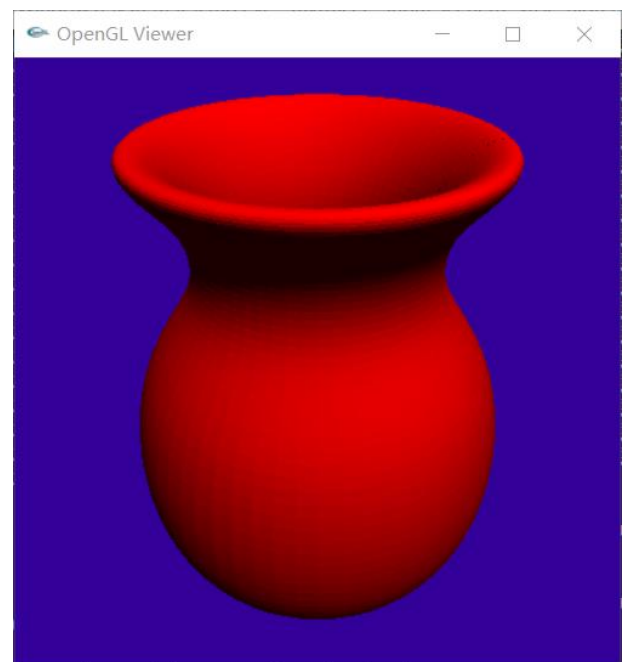
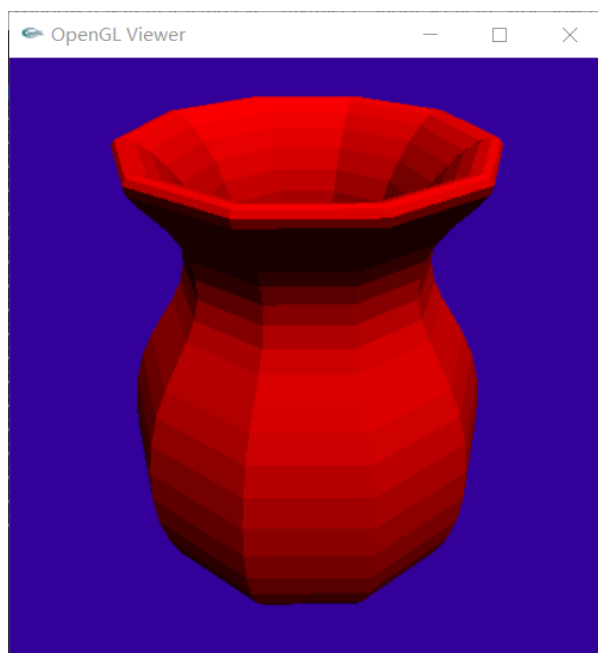
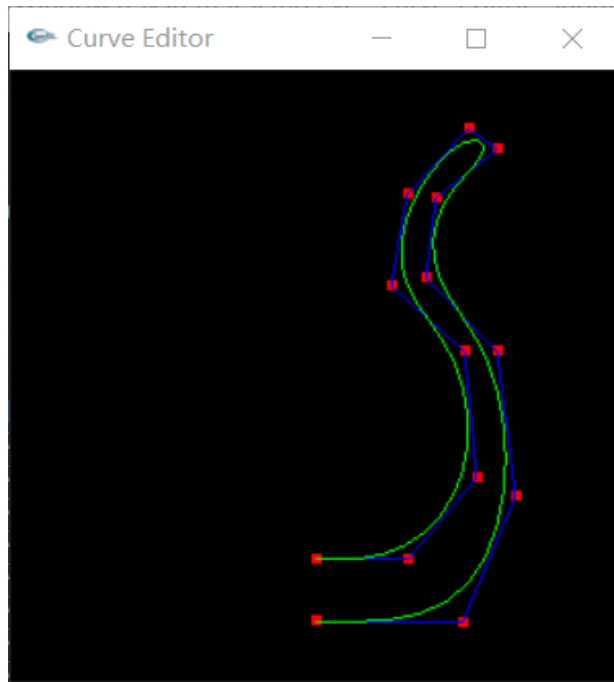
```
curve_editor -input spline06_torus.txt -curve_tessellation 4 -gui
curve_editor -input spline06_torus.txt -curve_tessellation 4 -
revolution_tessellation 10 -output torus_low.obj
curve_editor -input spline06_torus.txt -curve_tessellation 30 -
revolution_tessellation 60 -output torus_high.obj
raytracer -input scene06_torus_low.txt -gui -size 300 300
raytracer -input scene06_torus_high.txt -gui -size 300 300
```



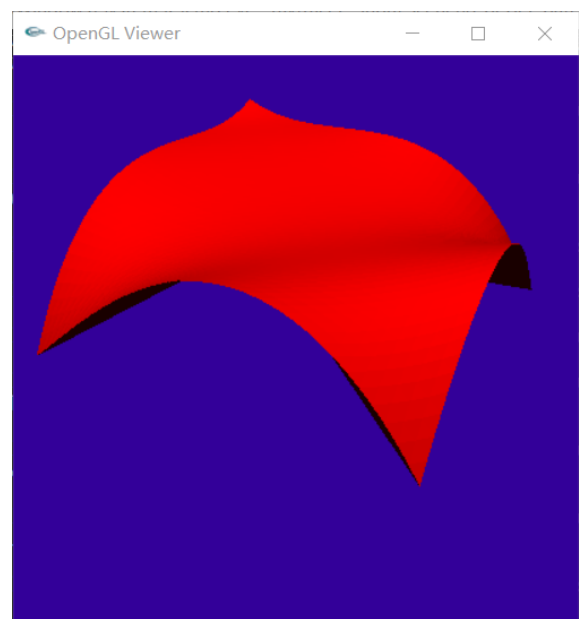
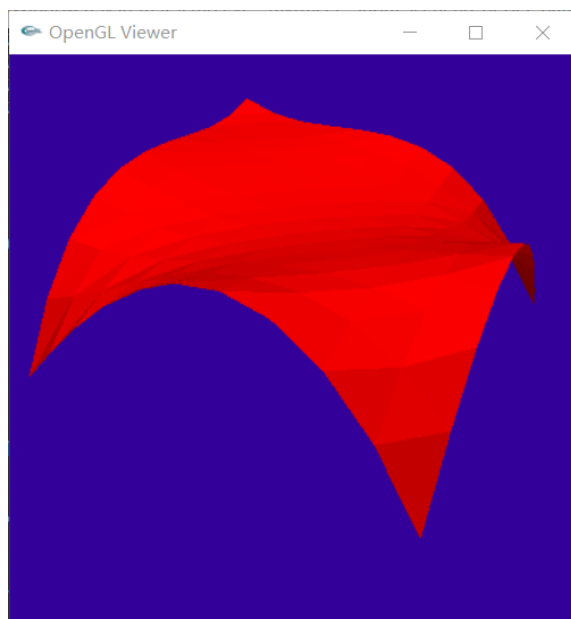
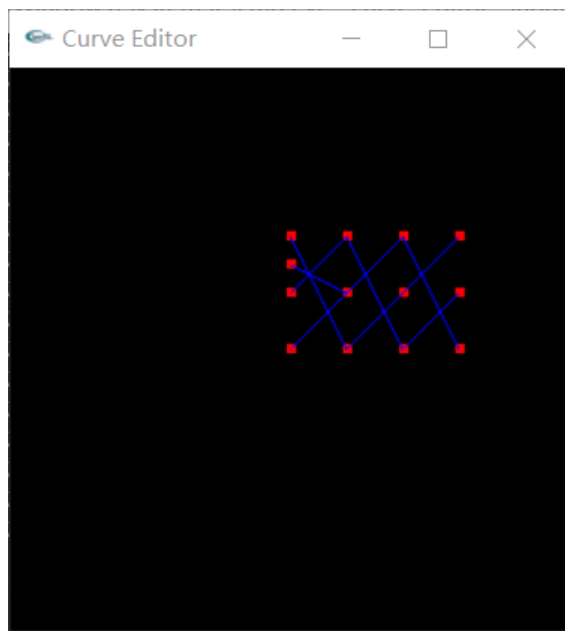


```
curve_editor -input spline07_vase.txt -curve_tessellation 4 -output_bspline
output07_edit.txt -gui
curve_editor -input output07_edit.txt -curve_tessellation 4 -
revolution_tessellation 10 -output vase_low.obj
curve_editor -input output07_edit.txt -curve_tessellation 10 -
revolution_tessellation 60 -output vase_high.obj
raytracer -input scene07_vase_low.txt -gui -size 300 300
raytracer -input scene07_vase_high.txt -gui -size 300 300
```

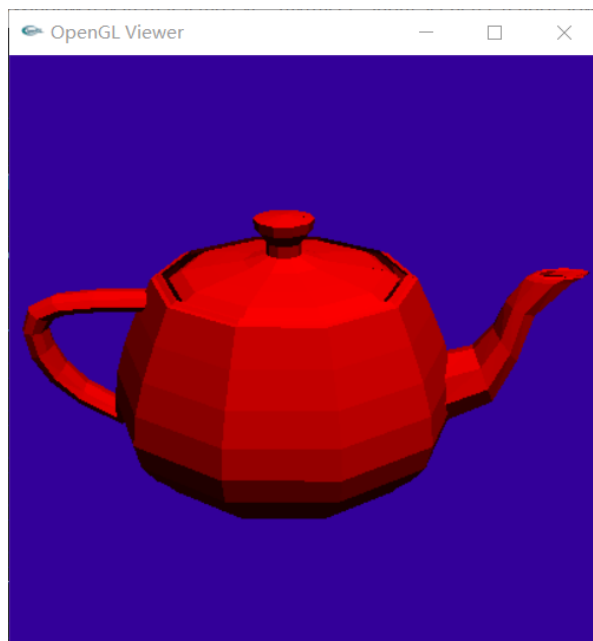
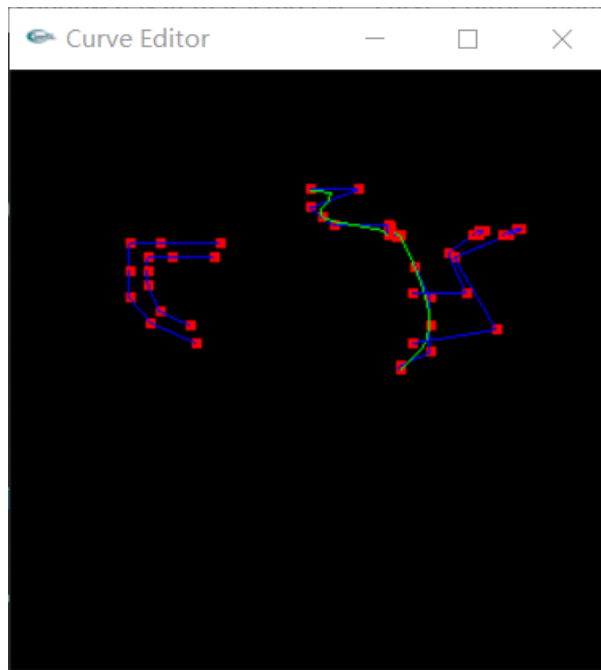




```
curve_editor -input spline08_bezier_patch.txt -gui
curve_editor -input spline08_bezier_patch.txt -patch_tessellation 4 -output
patch_low.obj
curve_editor -input spline08_bezier_patch.txt -patch_tessellation 10 -output
patch_med.obj
curve_editor -input spline08_bezier_patch.txt -patch_tessellation 40 -output
patch_high.obj
raytracer -input scene08_bezier_patch_low.txt -gui -size 300 300
raytracer -input scene08_bezier_patch_med.txt -gui -size 300 300
raytracer -input scene08_bezier_patch_high.txt -gui -size 300 300
```



```
curve_editor -input spline09_teapot.txt -curve_tessellation 4 -gui
curve_editor -input spline09_teapot.txt -patch_tessellation 4 -
curve_tessellation 4 -revolution_tessellation 10 -output teapot_low.obj
curve_editor -input spline09_teapot.txt -patch_tessellation 30 -
curve_tessellation 30 -revolution_tessellation 100 -output teapot_high.obj
raytracer -input scene09_teapot_low.txt -gui -size 300 300
raytracer -input scene09_teapot_high.txt -gui -size 300 300
```



## 四、实习总结

本次实习的内容是曲线和曲面的编辑，相比作业0难度大了许多，但是同时也有趣了许多，更多的是考验我们在曲线和曲面实现和总体框架上的理解，减少了许多细枝末节地方（题目帮你实现了），能够加深对上课内容的理解。就以我为例子，我做不来的时候，就看了下ppt回想老师上课讲的内容，发现其实思路ppt上都写好了，所以其实也并没有这么难。总体来说此次实习收获很大，不仅理解曲线曲面的实现和对这种实现框架的学习，还提升了我对计算机图形学的兴趣，最后还玩了一把批处理，收获匪浅。美中不足的是实现上可能有些瑕疵，并没有那么好，而且附加分也并没有做多少，希望在之后的作业当中继续努力，学习到更多。