**计算机图形学**

课程作业2：网格编辑

[林浚琪-102201528]

一、概述

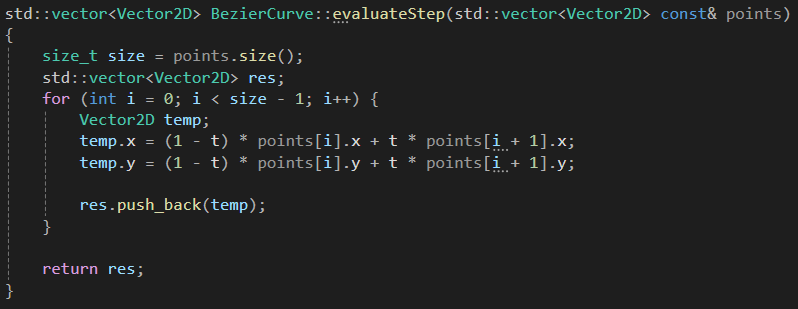
这个作业主要实现了贝塞尔曲线和曲面，并通过加权面积的顶点法线实现了曲面的平滑着色。最后实现边翻转和边分割，并凭借这两个函数实现loop细分，使物体更加平滑。

二、贝塞尔曲线和曲面

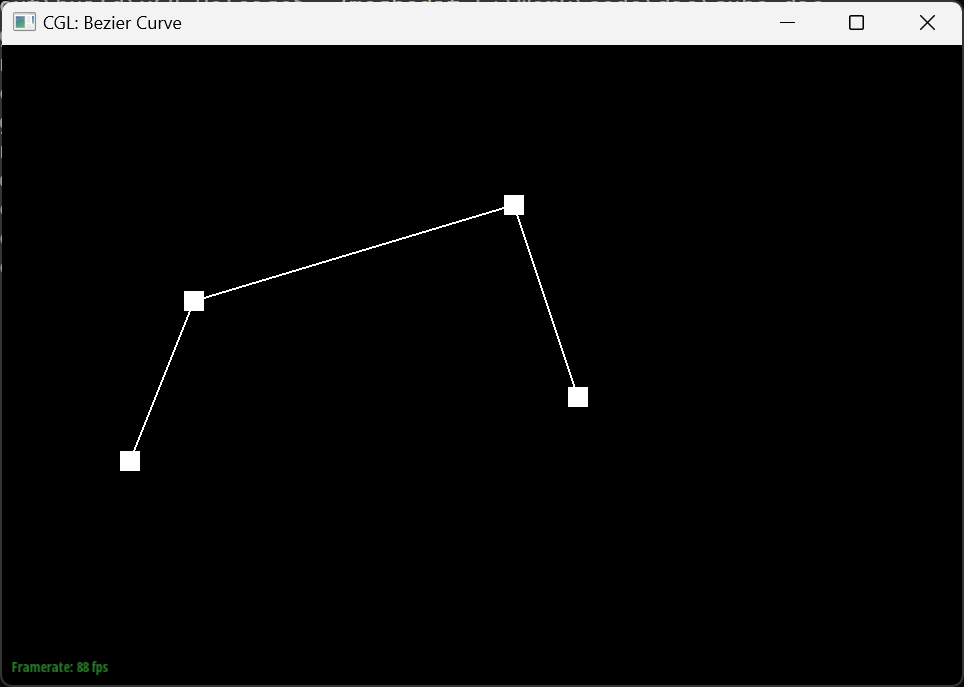
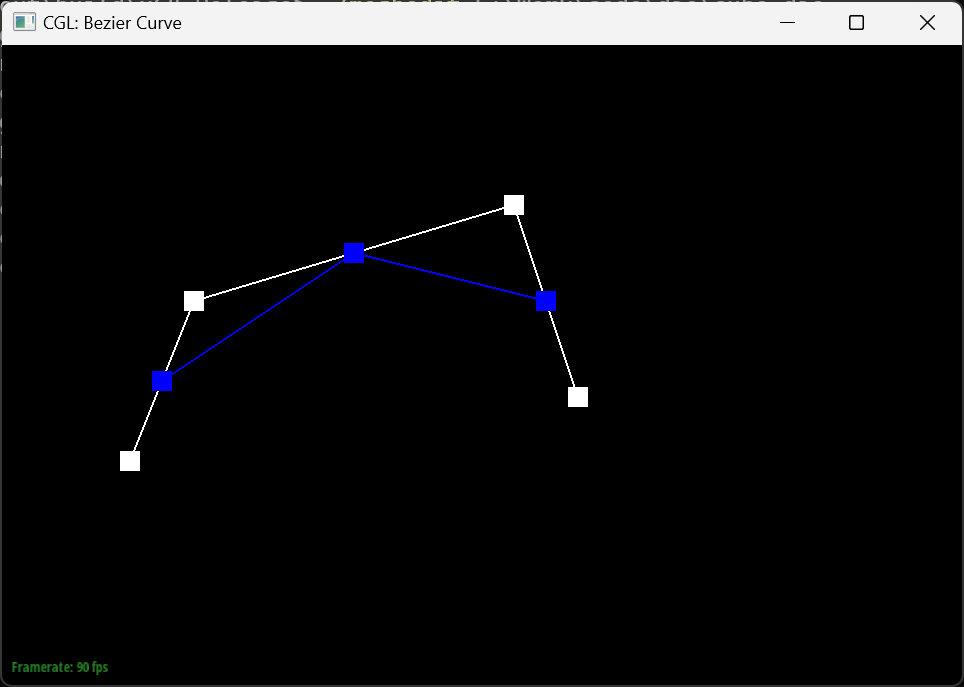
1. 基于 1D de Casteljau 绘制贝塞尔曲线

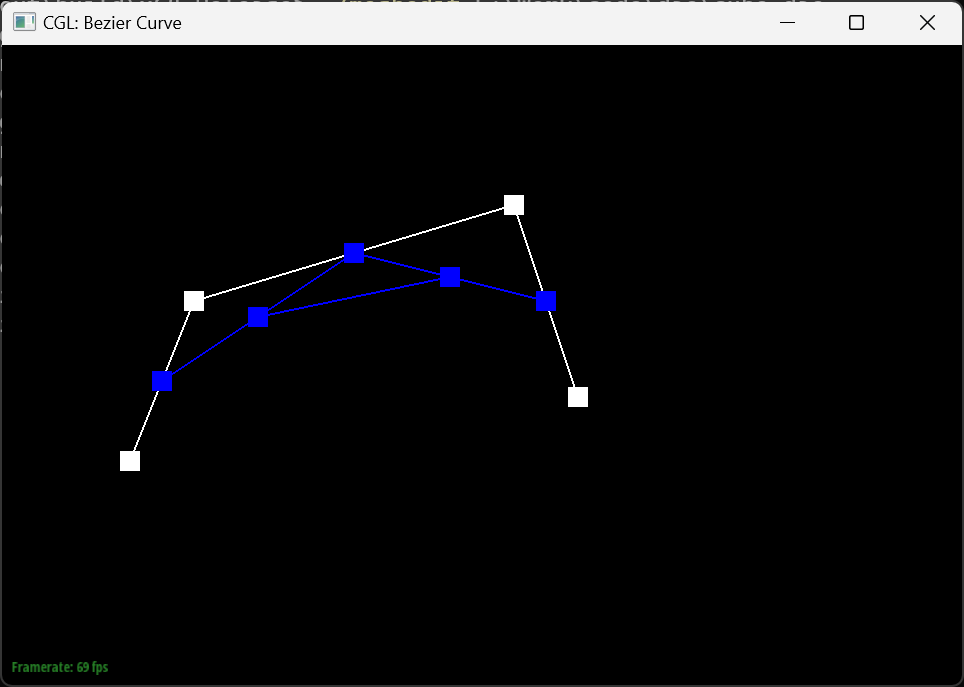
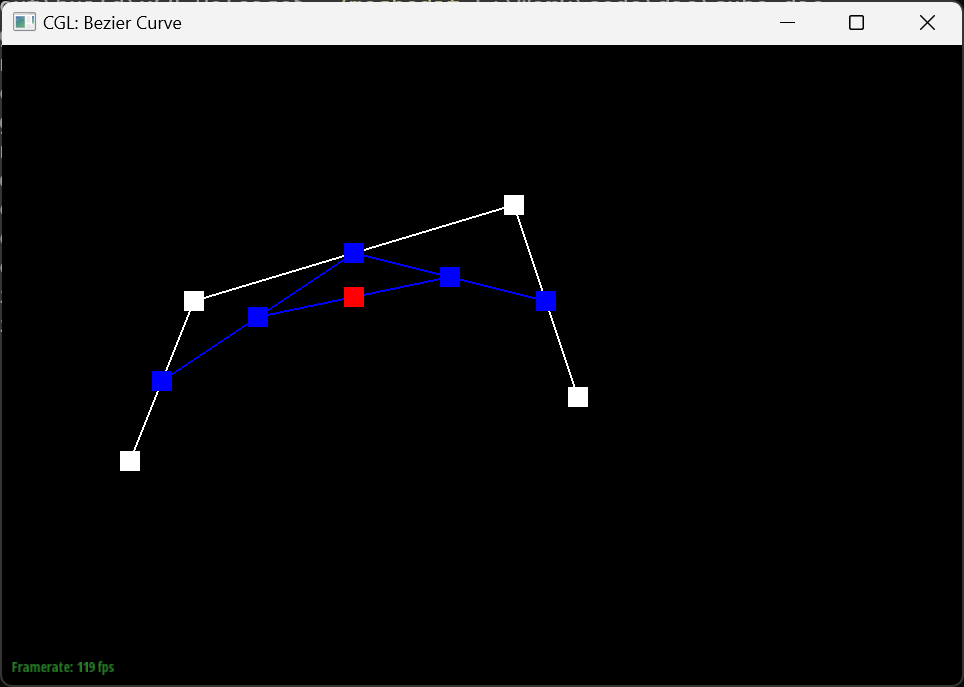
de Casteljau 算法是一种用于递归计算贝塞尔曲线上任意点的几何算法。它基于控制点的线性插值，并且与贝塞尔曲线的定义完全一致，适用于任何阶数的贝塞尔曲线。算法的核心思想是：通过在相邻控制点之间进行线性插值，逐步计算中间控制点，直到得到曲线上的一个点。它的主要公式是：

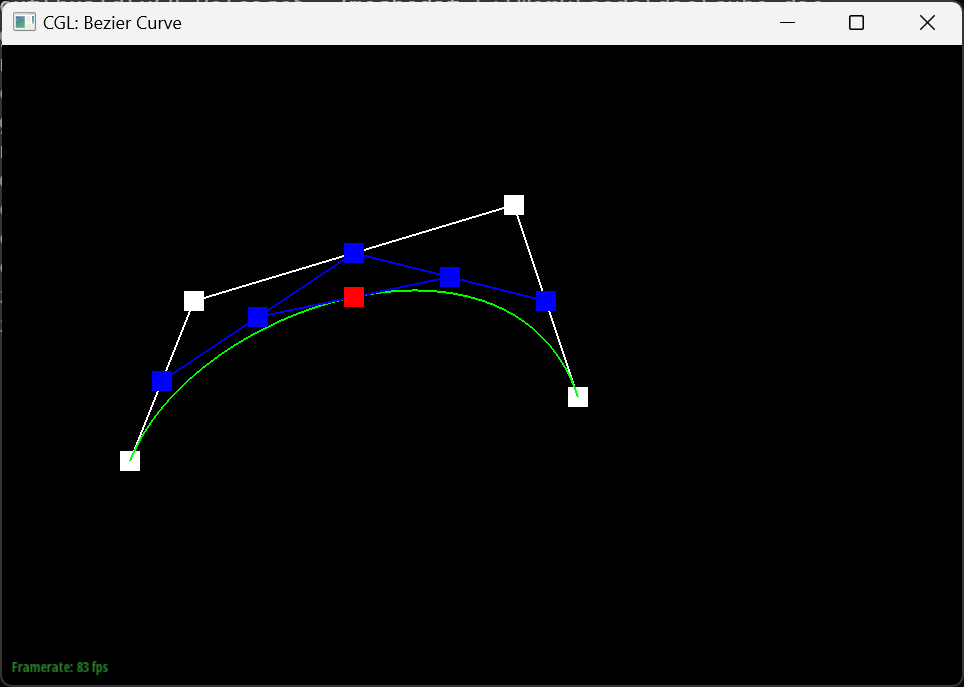
我实现了以下这个函数：



它通过遍历，返回每次传入的points数组的下一次迭代结果。一个size为N的points，经过N-1 次迭代，最后得到一个值。

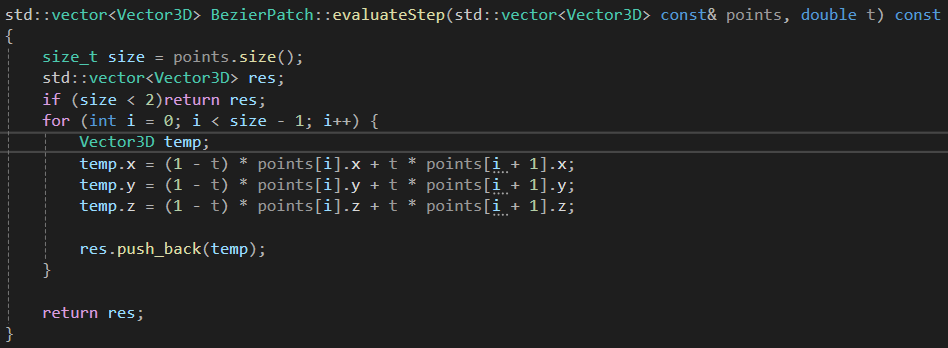
 



以上为按E依次迭代和按C显示完整贝塞尔曲线的截图。

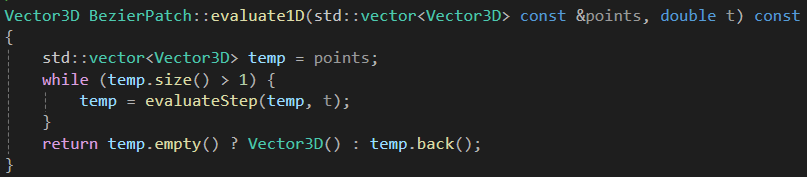
2. 基于 1D de Casteljau 绘制贝塞尔曲面

首先要将贝塞尔曲线迭代转化为贝塞尔曲面迭代，实现以下函数：

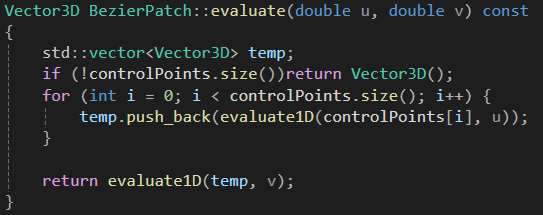


可以看到，这个函数和1部分基本类似，只是将Vertor2D改为Vertor3D。

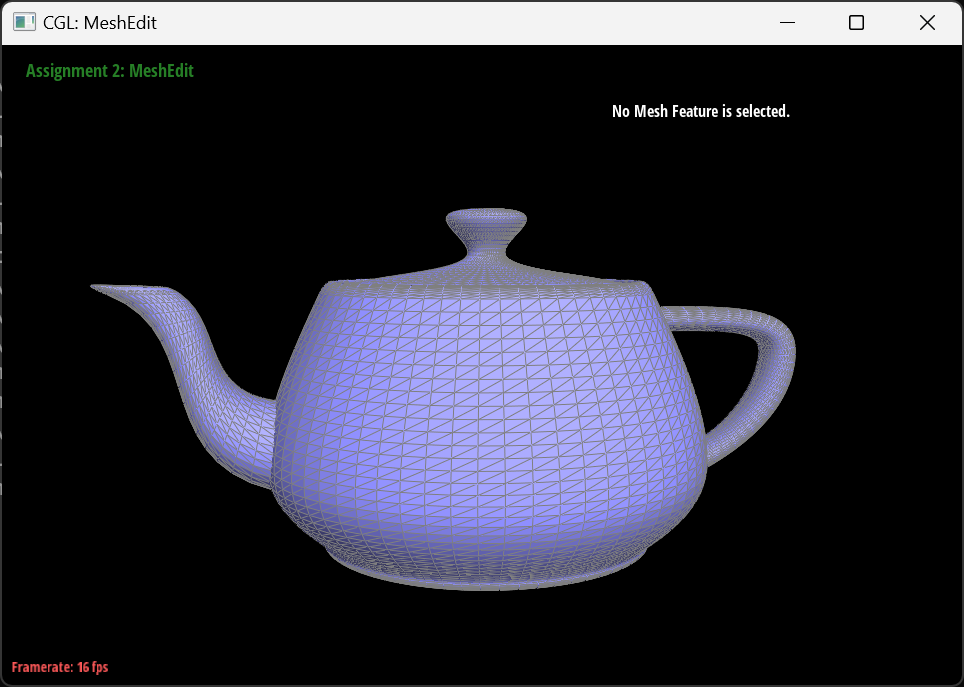
接着要实现一次性迭代到一个点的函数，这个函数只需在内部调用前一个evaluateStep函数即可：



最后要实现贝塞尔曲面的绘制，即在三维空间中先沿u迭代一轮，再把迭代的结果数组沿v迭代一轮，最后返回去贝塞尔曲面的点。这个函数仅需要再内部调用前一个函数：



成功执行这个函数后，我们可以执行控制台命令看到一个茶壶的贝塞尔曲面绘制图像：



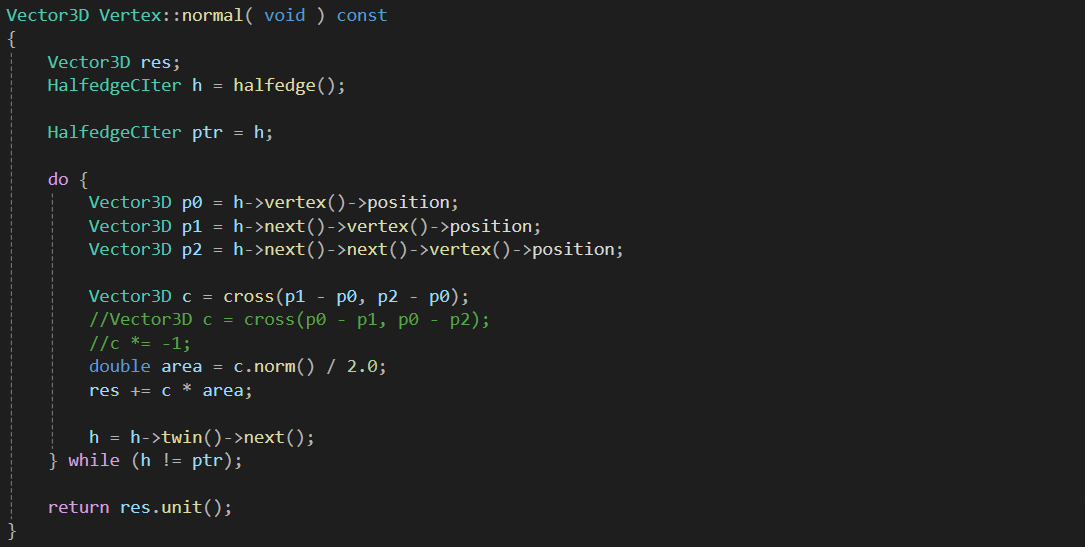
三、三角网格和半边数据结构

3. 基于面积加权的顶点法线计算

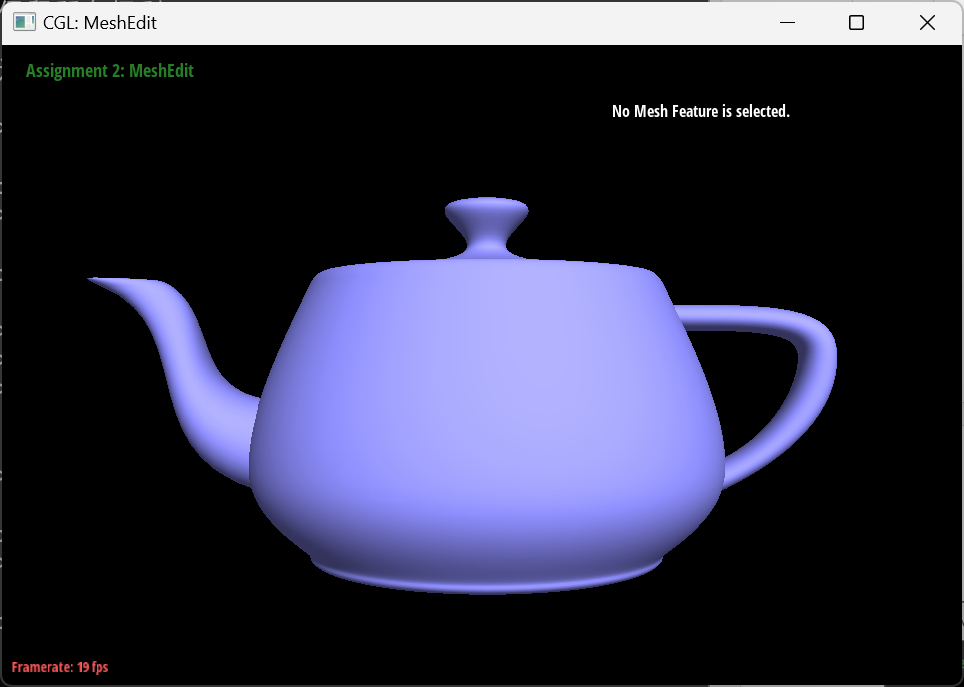
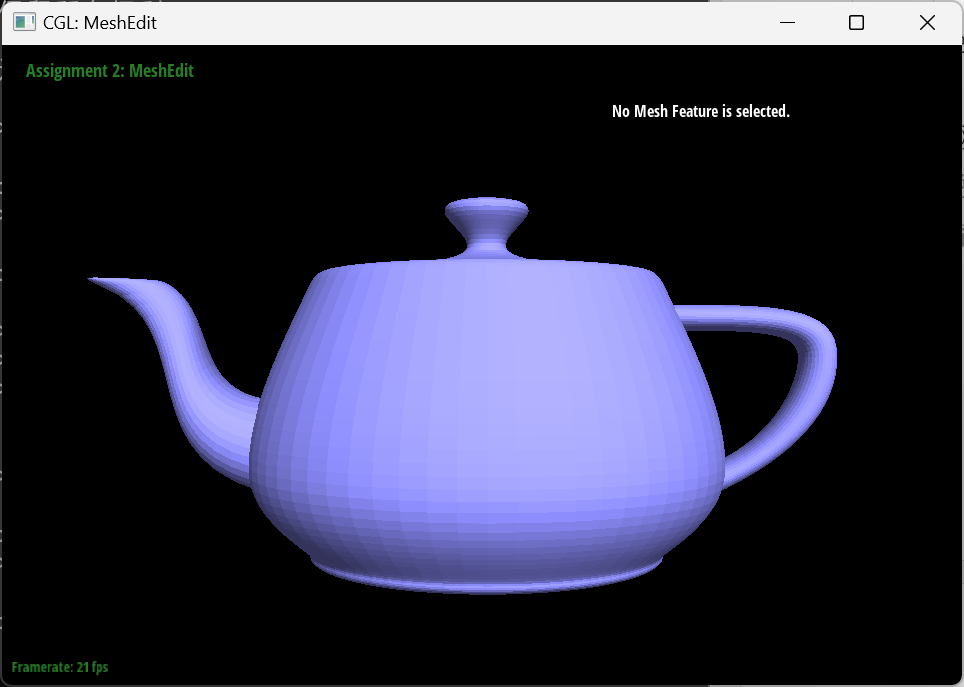
计算基于面积加权的顶点法线有四个步骤：

1. 计算每个面的法线（可以通过cross函数来实现）
2. 计算每个三角形的面积（通过法线计算）
3. 将每个面和其对应法线的乘积相加
4. 最后将相加结果归一化，得到最终结果

以下是实现的代码：



实现这个函数后，可以按Q来对物体进行平滑着色，下面是茶壶平滑着色前后的对比：



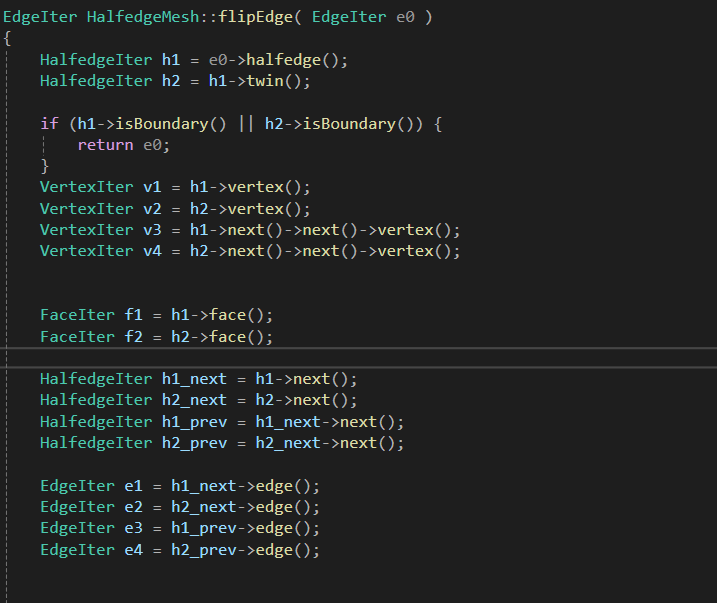
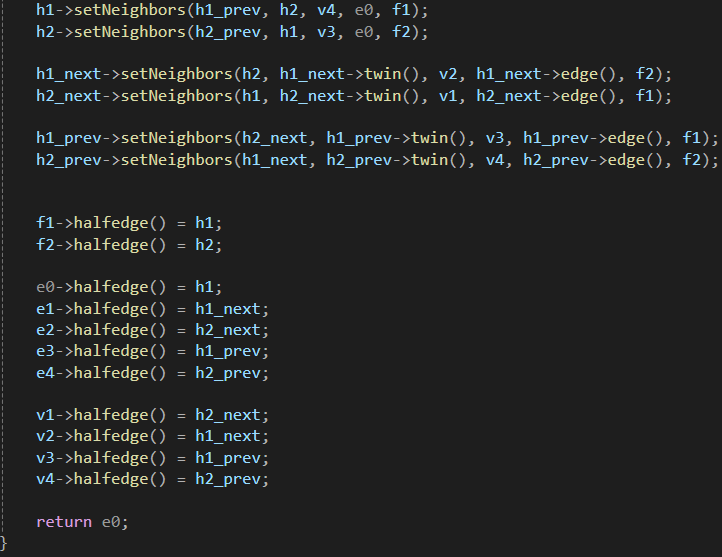
我关闭了网格显示来使得平滑着色更明显。

4. 边翻转(Edge Flip)

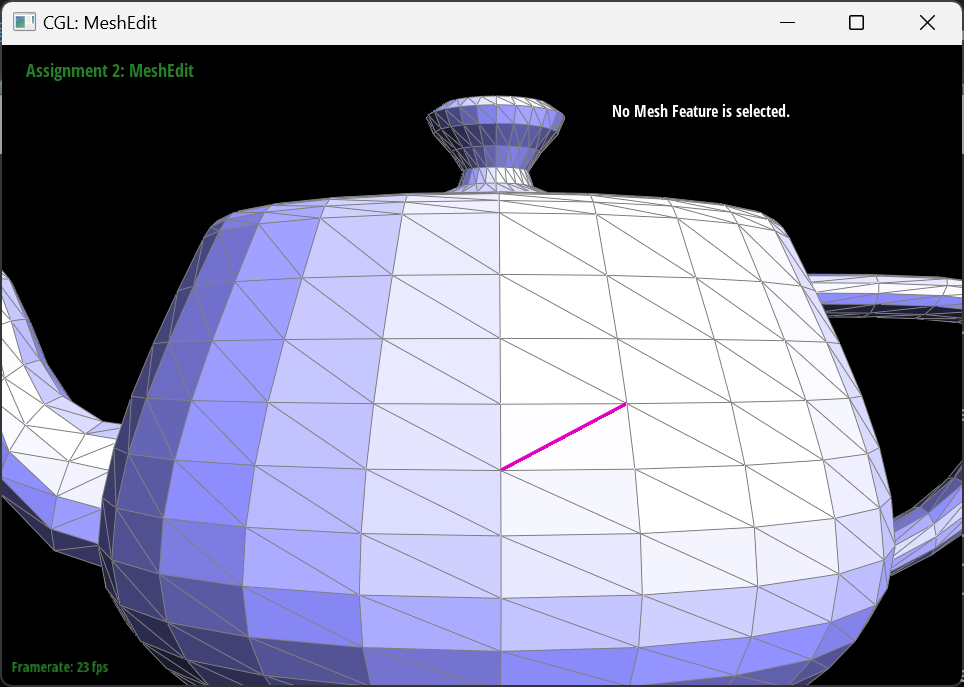
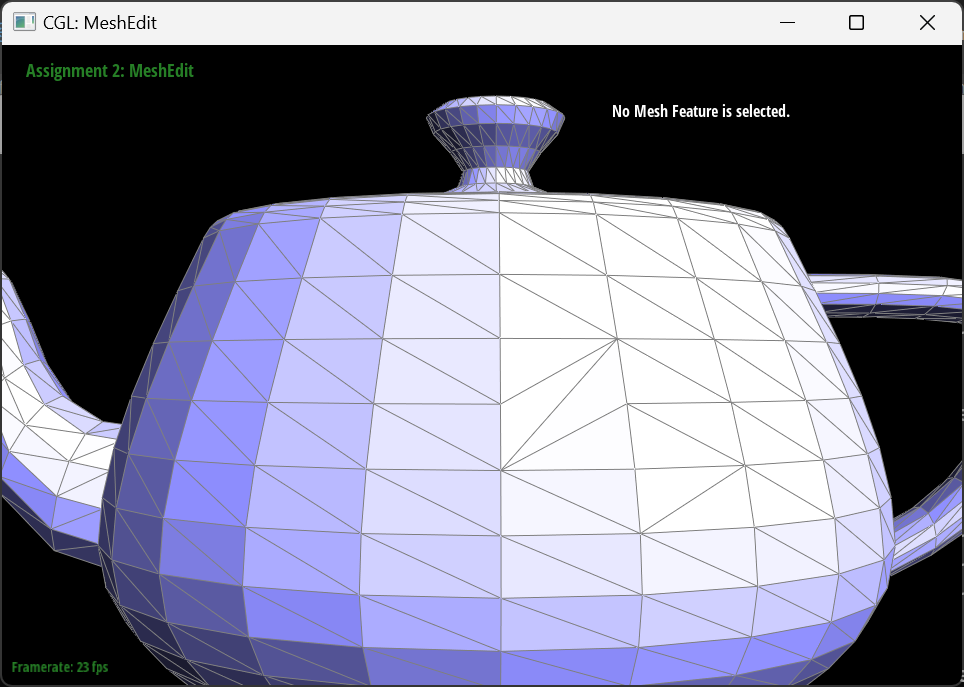
边翻转有三个操作步骤：

1. 确定相邻三角形
2. 翻转它们的相邻边
3. 检查翻转的合法性（设置这两个新三角形的相关属性）

边翻转没有创造任何新边，只是将原有的边所相关的参数重新设置，在我的函数中，为了保险起见，我重新设置了近乎所有相关的属性，即使其中有些是不必要的，这显得我的代码看的有一些臃肿，但是也保证了程序运行的安全性。以下是我的实现代码：

通过这个代码我可以对边进行翻转：

翻转边 多次翻转边

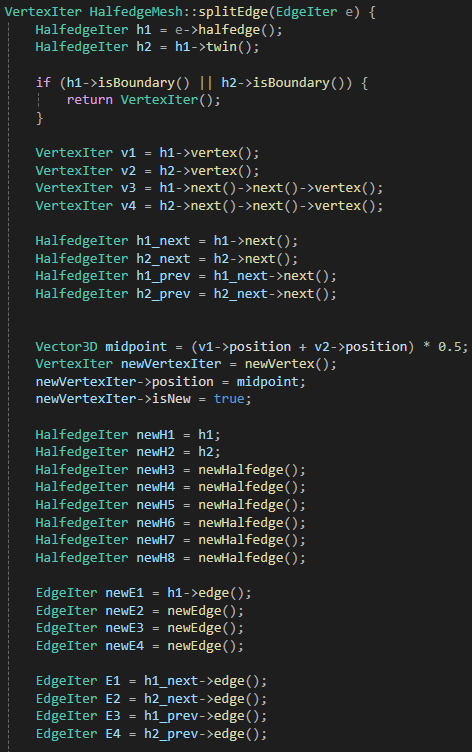
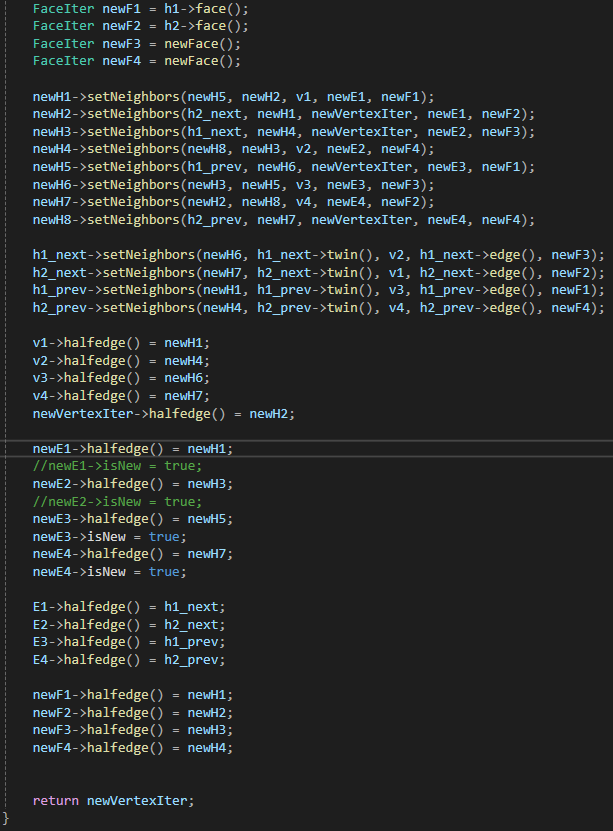
5. 边分割(Edge Split)

边分割需要一下操作步骤：

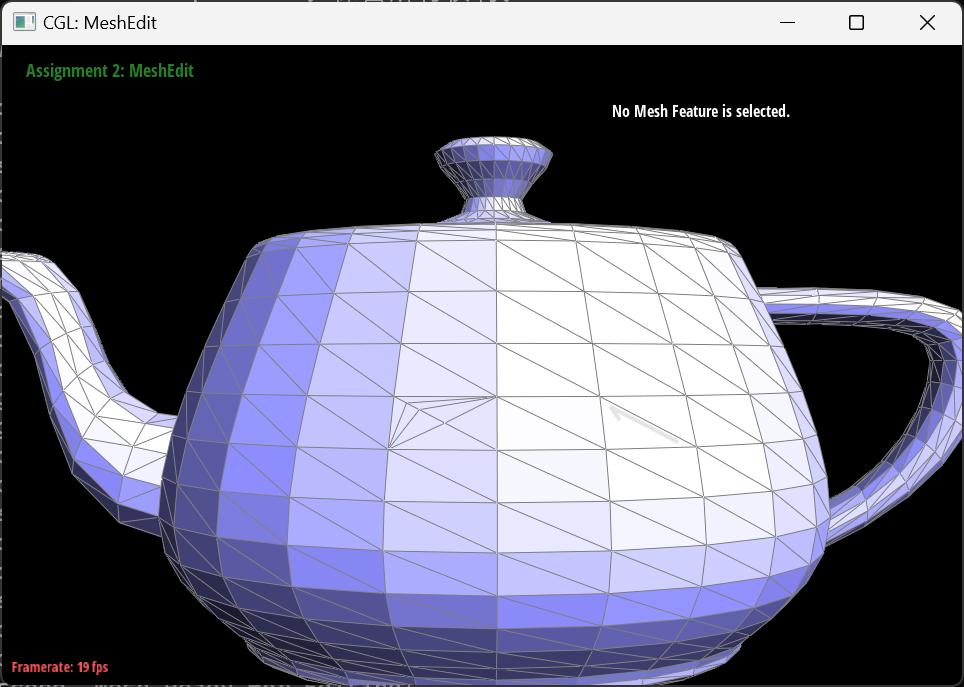
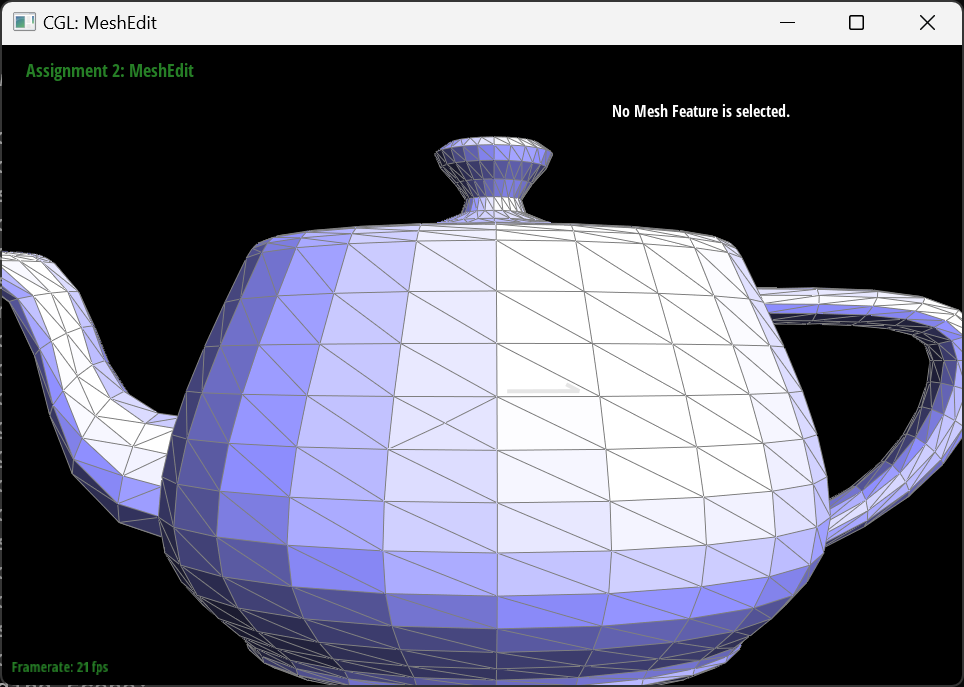
1. 确定要分割的边
2. 添加新顶点
3. 更新网格拓扑，维护拓扑结构

边分割不同边翻转，它需要创建一个新顶点，因此也会创建三条新边，六条新半边以及两个新面。因此我们需要对很多数据进行维护，我依旧采用像边分割一样的思路，把我认为应该维护的属性全部维护，来保证我分割边逻辑的正确性。由于我操作的图形全部都是封闭图形，因此我没有考虑对边界边进行分割。

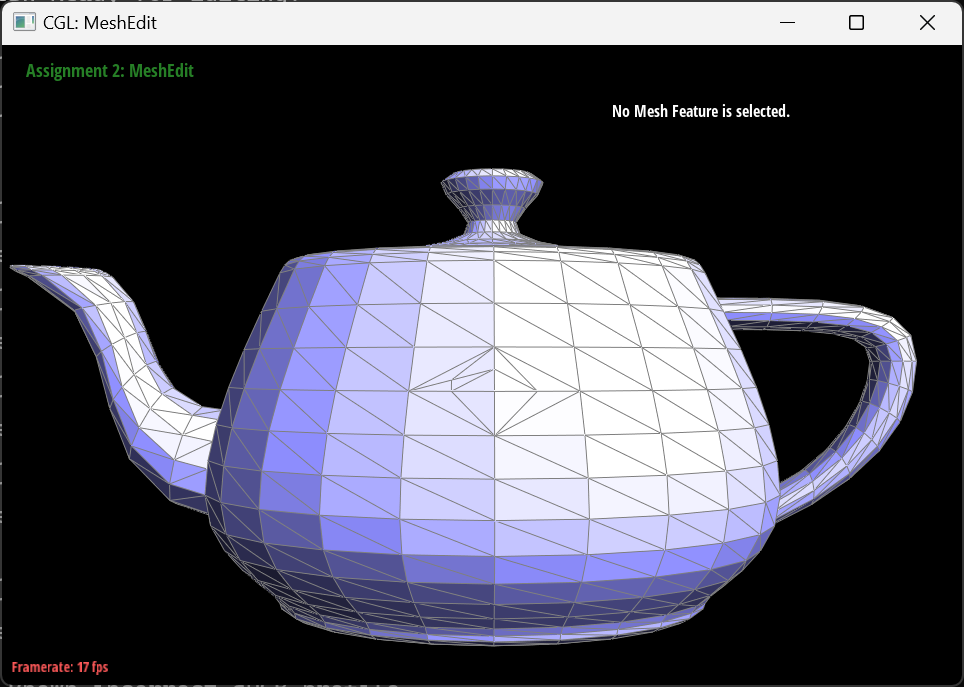
以下是我进行边分割的代码：

通过这个代码，我可以实现边分割操作：



分割单独一条边 多次分割同一条边

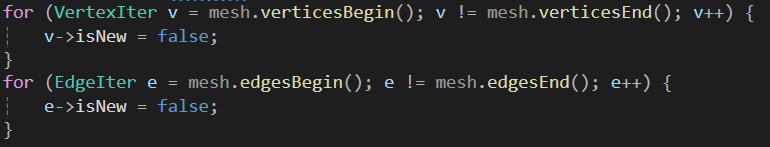


分割边与翻转边组合使用

6. 利用Loop Subdivision进行网格上采样

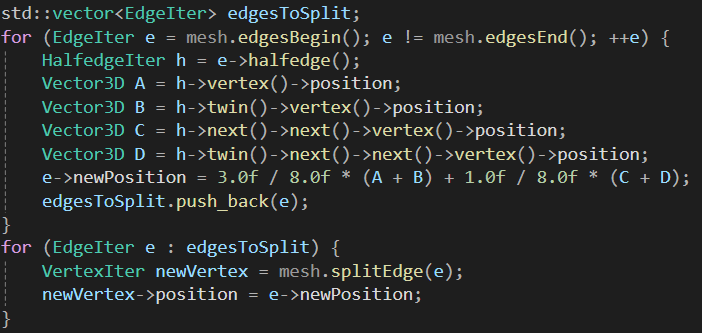
我的loop细分主要分为以下几个步骤：

1. 先把网格中所有点和所有边的isNew置为false

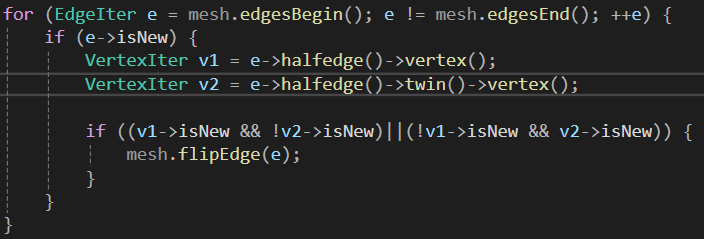


这一步是为了对网格属性进行初始化，避免后续重复细分时因为点和边的isNew未正确设置导致无法细分。

②对每条边计算出它的中点值，存储到边的newPosition中，然后对每条边进行分割，把分割后的返回的新点的position设置为边的newPosition

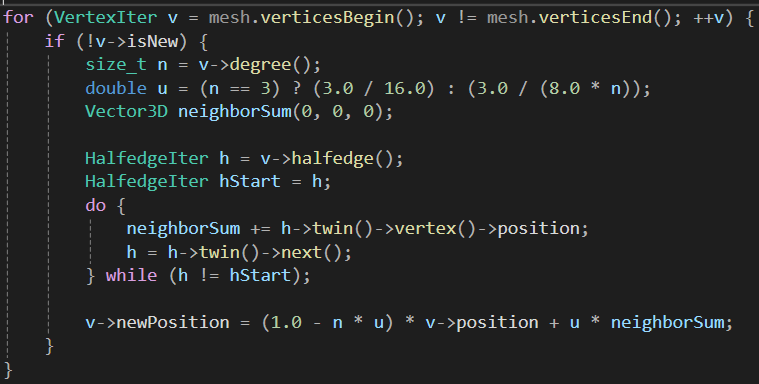


1. 对可以翻转的边进行翻转



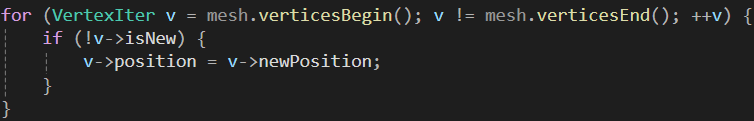
先判断是不是新边，再判断这条边是否链接旧顶点和新顶点

1. 对非新生成的点计算它们在细分后的新位置

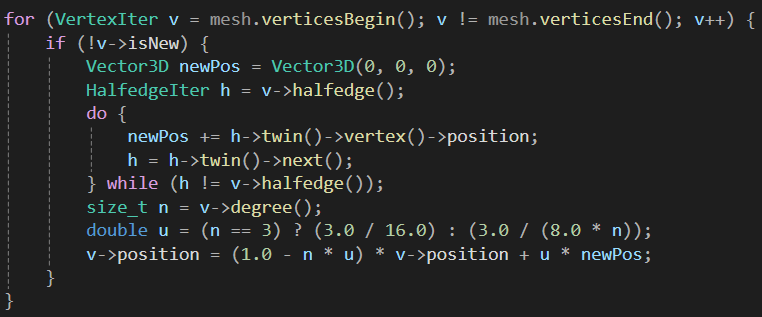


利用这个新位置计算公式可以使细分更加平滑

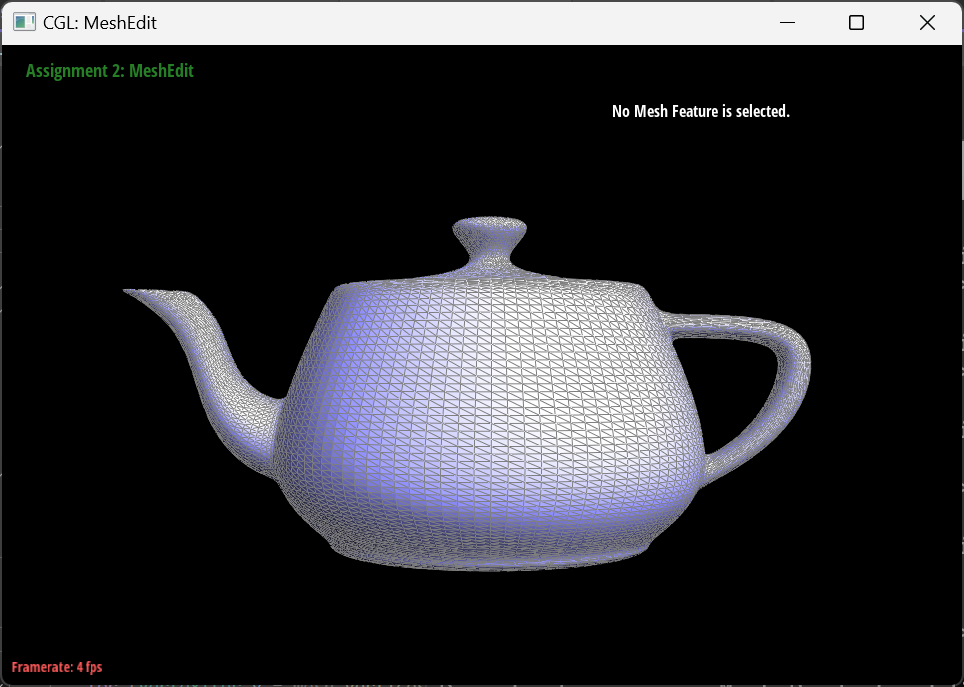
⑤最后更新所有旧点的位置



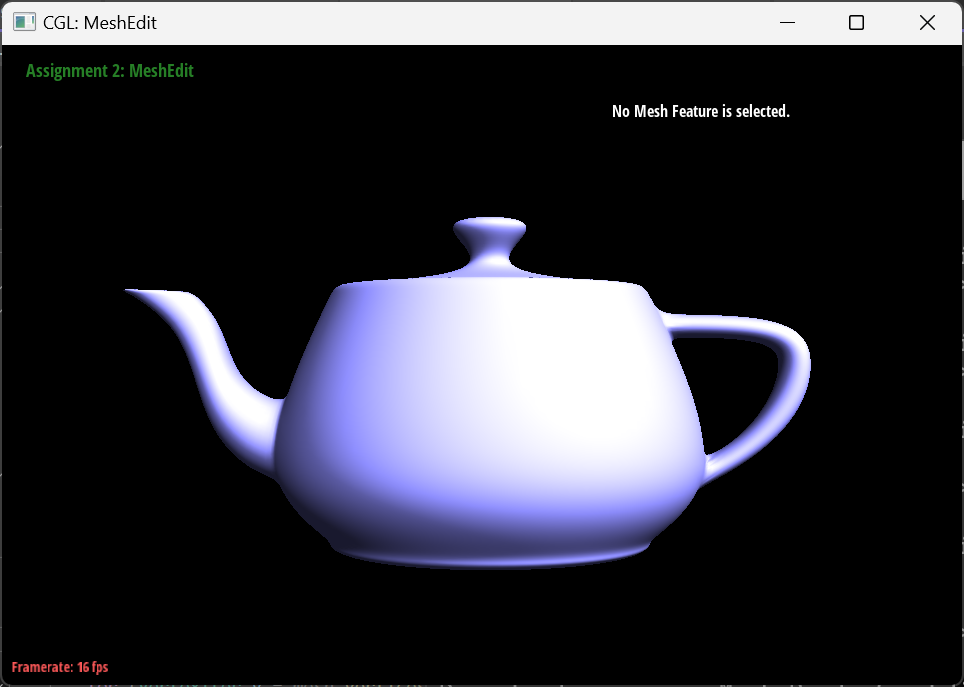
1. 优化：再进行一步④，重新计算更新后的点的位置



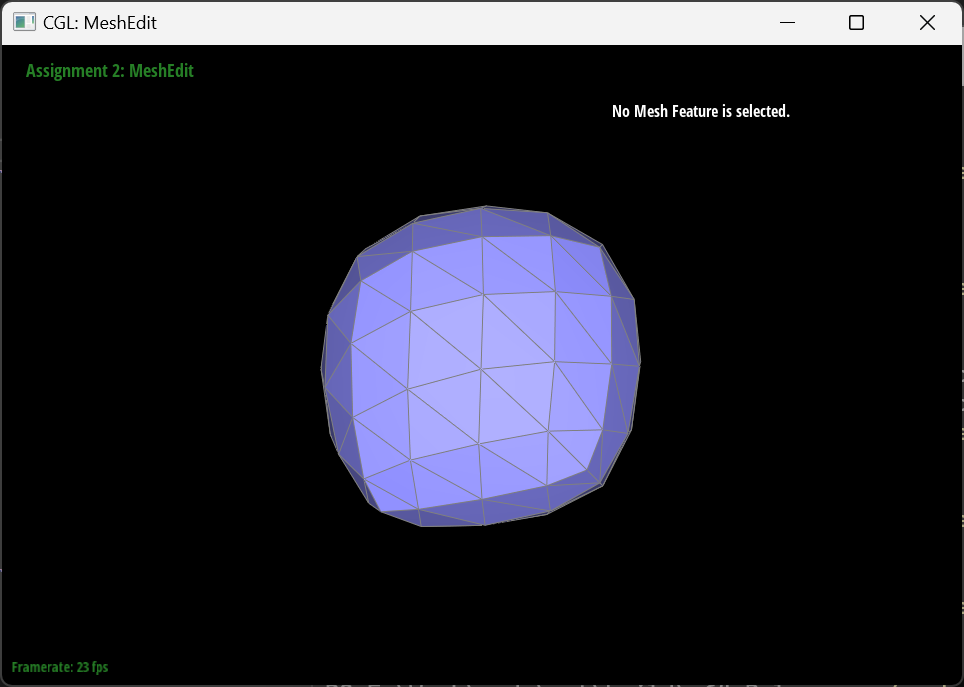
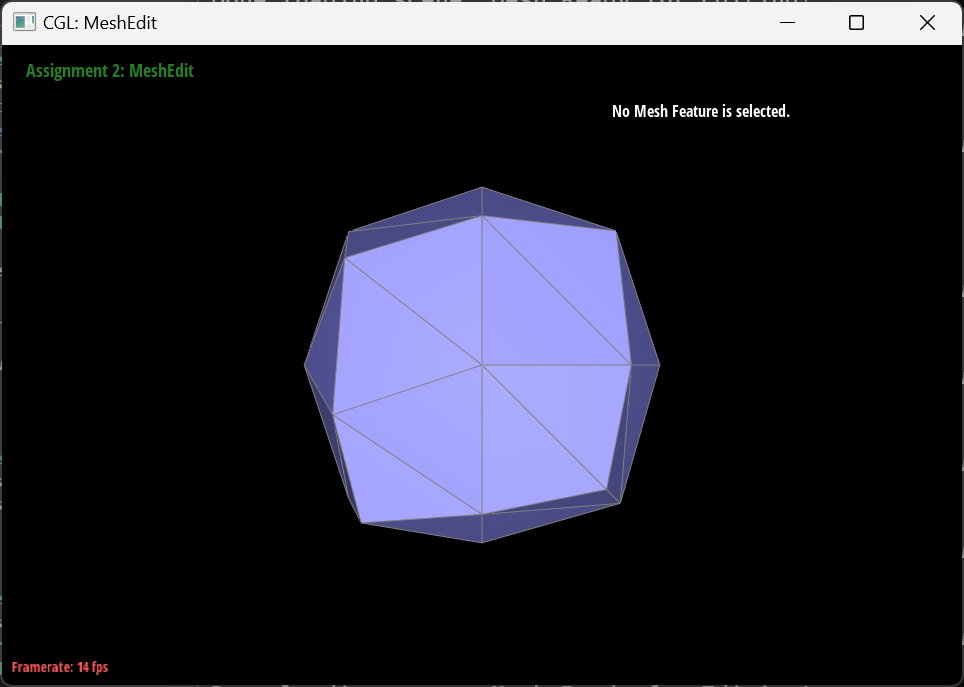
下面是茶壶进行细分的截图：

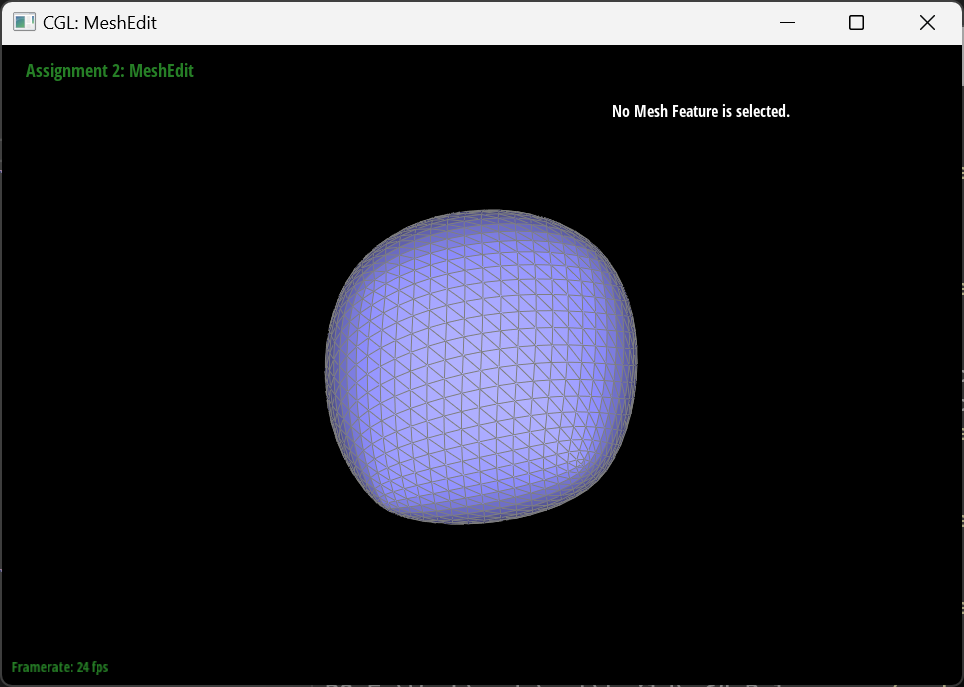
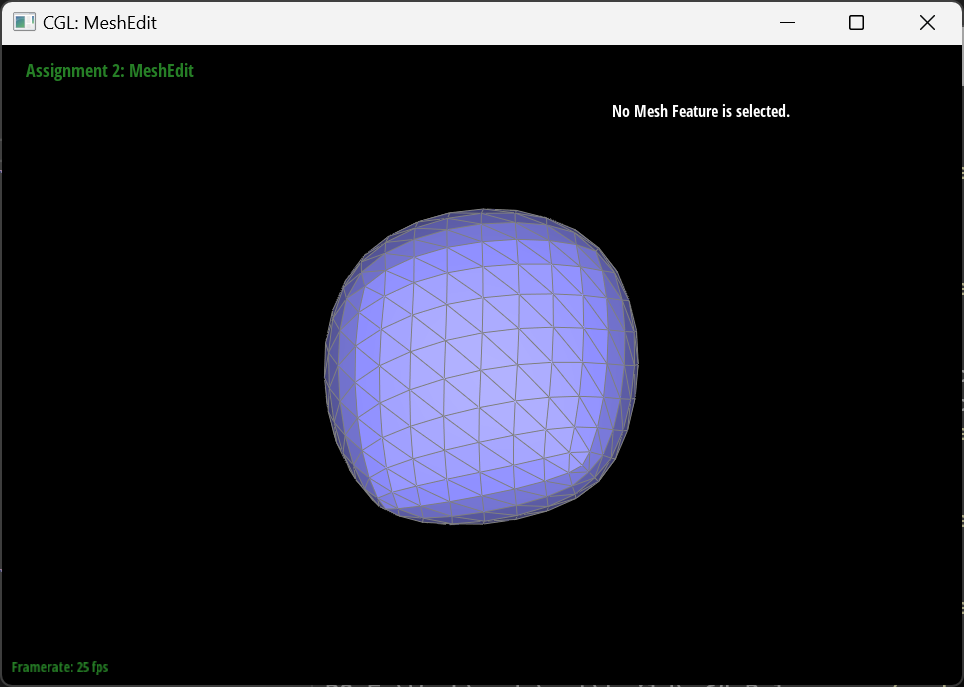


进行两次细分后的茶壶，可以看到它的表面已经变得十分光滑



关闭网格再进行平滑着色，已经十分的完美。



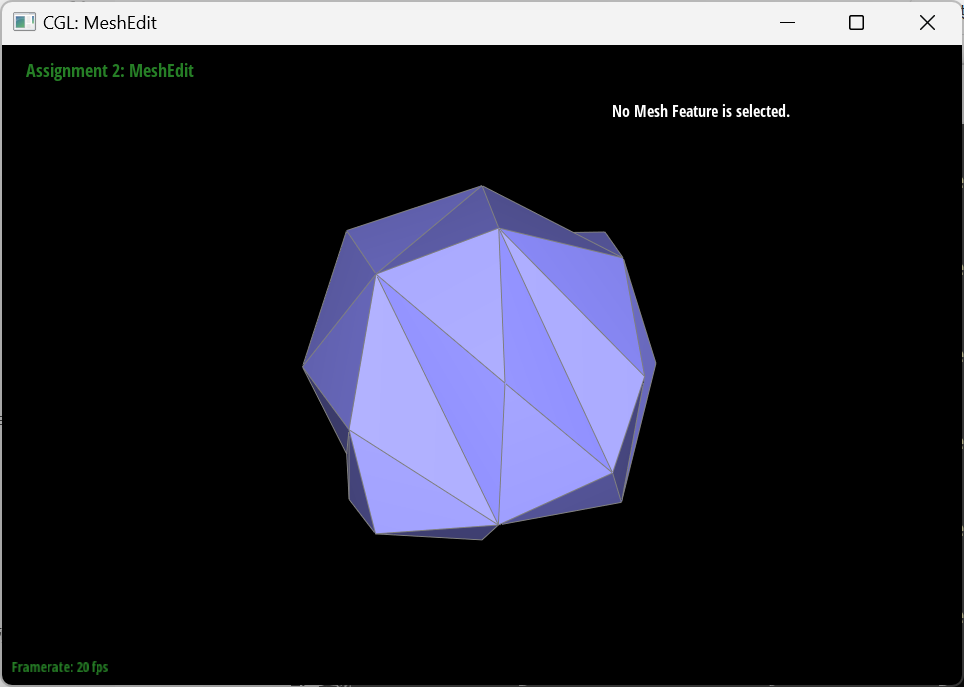
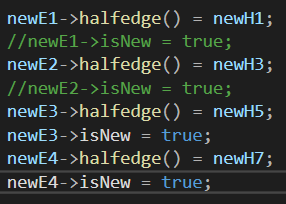


对立方体进行四次细分的结果。

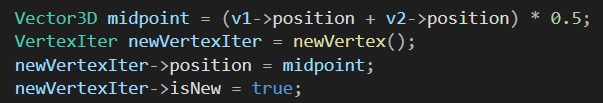
四、实践与思考

在我进行最后一步loop细分的时候，曾经遇到了两个问题：

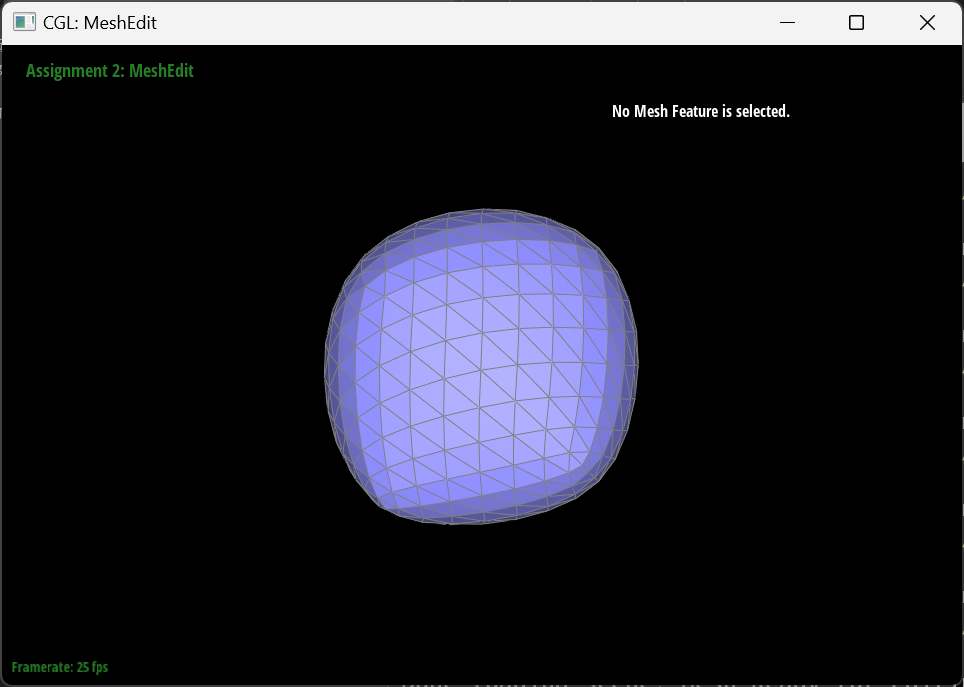
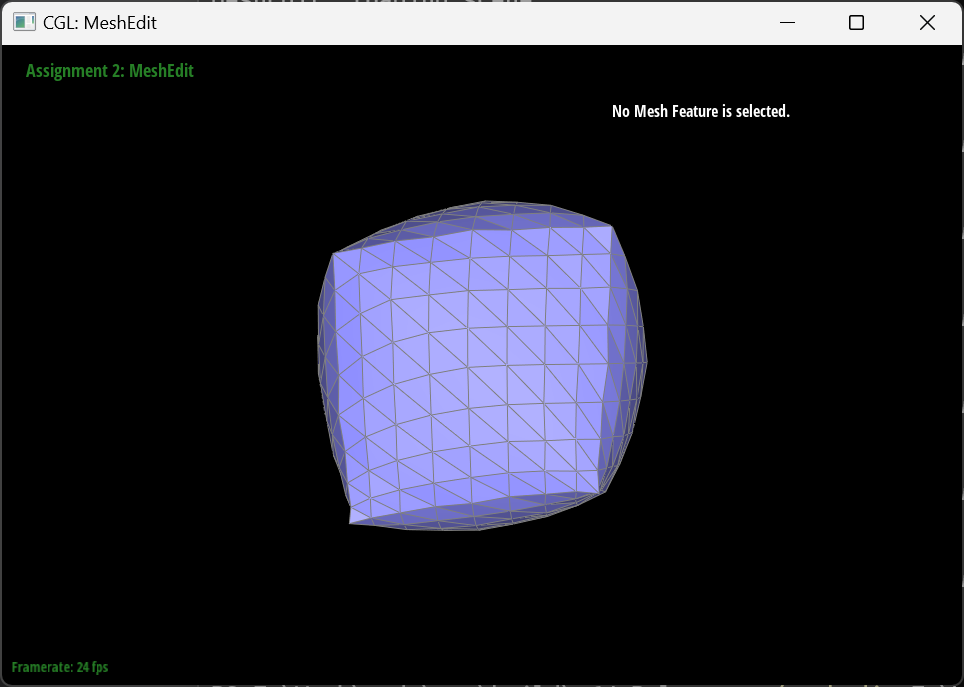
一个是细分不规则：

这个问题在我调试后发现是我一些边没有通过isNew判定来赋position，于是在边分割中我添加了设置新点和新边的isNew为True的语句，这个问题因此解决。



第二个问题是我loop细分的图像，在有直角顶点的地方没有正确的变光滑：



如图所示，我的cube细分后顶角仍然尖锐。

经过排查，我发现可以再最后再算一次网格中旧顶点的位置来修正这个问题，即上文中loop细分里的步骤⑥。只需要将最后一步的newPosition改为position，这样子细分后的立方体的顶角位置就可以正确进行更新。

