PA2 report

1.曲线性质

1.Bezier 曲线和 B 样条曲线有什么异同？

1. 相同点：

- 参数化曲线：它们都是通过参数化的数学公式定义的曲线，可以通过调整参数来控制曲线的形状。
- 都是三维空间内的样条曲线：三维空间内的样条曲线定义如下：

$$f(t) = \sum_{i=0}^{n} B_{i,k}(t) P_i$$

即二者都有相同的定义方式，只是是一维实数值多项式函数，即基函数定义不同。

- 控制点：两者都使用一组控制点来定义曲线，曲线的形态受到这些控制点的影响。

2. 不同点：

- 基函数定义不同，即曲线类型不同： 对Bezier曲线而言，k阶曲线的话，基函数定义为 $B_{i,k}(t) = \binom{n}{i}(1-t)^{n-i}t^i, \quad t \in [0,1]$，这是一个可以直接计算出的答案。
  与之相反，B样条的基函数定义相对复杂，为递归形式的定义。
  递归基的定义为

$$B_{i,0}(t) = \begin{cases} 1, & t_i \le t < t_{i+1} \\ 0, & \text{otherwise} \end{cases}, \quad 0 \le i < n+k+1$$

递归方程的定义如下：

$$B_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} B_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(t), \quad 1 \le p \le k$$

其中每一次递归B样条的阶数就会减一，直到达到递归基的0阶是结束递归，返回0或者1。

- 结点Knots B 样条曲线使用结点向量来定义曲线的局部控制，这些结点决定了曲线在控制点之间的分布，其中结点类型包括但不限于均匀、非均匀、准均匀等，在本次作业中则采取均匀的方式来实现。B 样条曲线的形状不仅由控制点决定，还受到结点分布的影响，可以实现更复杂的曲线形态。
  Bezier 曲线不需要结点向量，完全由其控制点决定，每个段是独立的。
- 复杂性 B 样条曲线由于其递归定义和高连续性，在计算上更复杂，当 n = k, t0 = · · · = tn = 0, tn+1 = · · · = t2n+1 = 1 时，有

$$B_{i,n}(t) = \binom{n}{i}(1-t)^{n-i}t^i$$

即B 样条在特定取值下，是一个次数为 n，控制点个数为 n + 1 的 Bezier曲线。
然而Bezier曲线却无法达到B 样条的复杂性，可以说B样条是Bezier曲线的推广。

2. 怎样绘制一个首尾相接且接点处也有连续性质的 B 样条？

通过查询资料，我发现wrapping节点的方法可以解决这个问题：

1. 构建： 假设我们想要构建一个由n+1个控制点P0, P1, ..., Pn定义的p次闭B-样条曲线C(u) 。构建过程如下：

- 增加一个新控制点 Pn+1 =P0.因此，控制点的数目是 n+2；
- 找到一个合适的有 n+1节点的节点序列u0, u1, ..., un 。
- 增加 p+2 个节点并 wrap 头 p+2个节点： un+1 = u0, un+2 = u1, ..., un+p = up-1, un+p+1 = up, un+p+2= up+1。这样，我们有n+p+2 = (n+1)+p+1 个节点；
- 定义在上述构建的 n+1个控制点和n+p+2 个节点上的 p次开B-样条曲线C(u)是一个闭曲线，在连接点处C(u0) = C(un+1)有Cp-1 连续性。注意闭曲线的定义域是 [u0, un+1]。

2. 原理解释：

- 为了使闭合曲线首尾相接，需要在原始控制点序列的末尾添加一个新的控制点，该控制点的坐标与起始控制点P0相同。
- 为了保证曲线在连接点处具有Cp-1连续性，在原始节点序列的基础上增加额外的节点，并将这些节点进行“wrapping”操作，将末尾的p+2个节点与开头的p+2个节点连接起来。

revsurface.hpp逻辑

1. 先调用曲线自带的discretize函数，将曲线 pCurve 离散化为一系列点，并存储在 curvePoints 中
2. 使用嵌套循环遍历曲线上的离散点，并根据给定的步数，旋转这些点以创建表面效果。具体来说，包括以下几个步骤：

- 先根据给定的steps和轮次计算出旋转角度t，接下来调用setAxisAngle函数，生成一个旋转矩阵，绕给定轴旋转给定角度，这里调用函数需要两个参数：旋转的角度和旋转轴，利用t和Vector3f::UP可以完成。
- 接下来利用生成的旋转矩阵生成新的点与法向，这里首先将法向量与负的世界坐标系的 z 轴进行叉乘操作，得到切线向量 pNormal，然后再乘上旋转矩阵得到 nnew。

3. 接下来是确定面的信息：

- 首先确定当前点的下一个点的索引。如果当前点是当前步骤（steps）中的最后一个点，则下一个点的索引将是 0，否则下一个点的索引就是当前点索引加 1。
- 然后确保在生成面片索引之前，不会超出曲线点的范围。接下来插入索引，即顶点构成两组三角形的顶点，分别为：当前点、旋转后的下一个点、旋转后的当前点。和当前点、下一个点、旋转后的当前点。

```
int i1 = (i + 1 == steps) ? 0 : i + 1;
if (ci != curvePoints.size() - 1) {
    surface.VF.emplace_back((ci + 1) * steps + i, ci * steps + i1, ci * steps + i);
    surface.VF.emplace_back((ci + 1) * steps + i, (ci + 1) * steps + i1, ci * steps
+ i1);
}
```

4. 最后使用了 OpenGL 来渲染生成的曲面三角形网格。

- glBegin(GL_TRIANGLES)：标志着开始绘制三角形。
- 接下来在循环中，遍历了存储三角形面片索引的 surface.VF 向量，利用索引分别读取坐标向量和法向量，并将其传递给 glVertex3f 和 glNormal3f 函数，以绘制三角形。
- glEnd()：标志着三角形绘制结束。

3. 代码参考

在本次作业完成的过程中，我参考了README和习题课ppt对代码结构的讲解，以及群聊中的提示，并未与其他同学交流。

4. 实验过程中的问题

1. 主要问题是在Bspline的绘制中，我开始取错了区间，忘记了有效样本区间为$[t_k, t_{n+1}]$，导致采样出现错误最终绘制的曲线错误。其次是在resolution的取样中，我因为没有仔细阅读readme，以为是在整个样本空间中取样，最后发现后在每两个样本点间取样解决了问题。