个人信息和实验环境

杨雅儒, 2017011071, 计85

Ubuntu 18.04, gcc 7.5.0

注意由于换了一台电脑，在这台电脑上需要将 main.cpp 的 screenCapture 函数中 glReadBuffer 注释掉才能够正常保存图像，所以从 PA3 开始提交的版本注释掉了这一行。**如果助教检查时不能正常生成，烦请将该处注释取消**。

问题回答

Bezier 曲线与 B 样条曲线

相同点：

- B 样条曲线保留了 Bezier 曲线的优点，实际上就是 Bezier 曲线的扩展；
- 两者都可通过设置控制结点来得到一条曲线，且能够得到曲线上任意一点的位置及其切向量；
- 两者都可通过递归的形式（例如 de Casteljau 算法和 de Boor-Cox 算法）便捷地进行运算。

不同点：

- B 样条曲线与 Bezier 曲线的基函数不同；
- B 样条具有局部性质，调整某一控制节点时只会影响其周围局部的曲线，便于调整；但 Bezier 曲线并不具有这个性质，这意味着调整某一控制节点会影响整条曲线，很不方便微调曲线。
- B 样条的次数可以调整，次数越高，控制点影响的曲线段数就越多；而 Bezier 曲线的次数固定为控制点数减 1；

怎样绘制一个首尾相接且接点处也有连续性质的 B 样条：

- 首先选取若干控制结点，形成一个控制结点序列；
- 令样条曲线次数为 k，则将原序列中最后 k-1 个结点复制添加到序列的开头，将原序列中前 k-1 个结点复制添加到序列的结尾。

旋转曲面绘制逻辑

旋转曲面相关的代码在 revsurface.hpp 中，主要部分即 drawGL() 函数中的内容。

- 定义结构体 Surface，其中 VV、VN 分别存储表面上各个顶点的位置以及归一化的切向量，VF 存储每个三角面片由哪些顶点构成；

```
struct Surface {
    std::vector<Vector3f> VV;
    std::vector<Vector3f> VN;
    std::vector<Tup3u> VF;
} surface;
```

- 调用 discretize，获取二维的曲线上的若干个离散的点以及切向量：

```
std::vector<CurvePoint> curvePoints;
pCurve->discretize(30, curvePoints);
```

- 定义 step，用于表示在旋转方向上的精细度，代码中固定为 40：

```
const int steps = 40;
```

- 枚举二维曲线上的每一个离散点，并且枚举每一个旋转角度（由上面的 step 决定，范围为 [0, 2*PI]），通过定义四元数 rot，并通过旋转轴 Vector3f::UP 也即（0, 1, 0）和旋转角度初始化 rot，转换为旋转矩阵之后同二维曲线上的位置向量相乘，即可得到对应的三维顶点坐标，且利用叉乘得到二维法向量后再旋转得到三维下的法向量：

```
for (unsigned int ci = 0; ci < curvePoints.size(); ++ci) {
    const CurvePoint &cp = curvePoints[ci];
    for (unsigned int i = 0; i < steps; ++i) {
        float t = (float) i / steps;
        Quat4f rot;
        rot.setAxisAngle(t * 2 * 3.14159, Vector3f::UP);
        Vector3f pnew = Matrix3f::rotation(rot) * cp.V;
        Vector3f pNormal = Vector3f::cross(cp.T, -Vector3f::FORWARD);
        Vector3f nnew = Matrix3f::rotation(rot) * pNormal;
        surface.VV.push_back(pnew);
        surface.VN.push_back(nnew);
        ...
```

- 到此处已经求出了旋转体的每个离散点坐标及其法向量，再通过相近的结点得到若干面片的顶点坐标——具体地，令每个顶点 v1 再旋转一个离散单位后顶点为 v2，v1 再向下一个二维离散曲线点走一个位置对应顶点为 v3，v1 既继续旋转又向下一个二维离散曲线点走得到的对应顶点为 v4，则 (v1, v2, v3) 和 (v2, v4, v3) 分别为对应的两个三角面片，对应代码为：

```
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);
}
```

- 最后只需要使用 `GL_TRIANGLES` 将这些三角面片绘制出即可：

```
glBegin(GL_TRIANGLES);
for (unsigned i = 0; i < surface.VF.size(); i++) {
    glNormal3fv(surface.VN[std::get<0>(surface.VF[i])]);
    glVertex3fv(surface.VV[std::get<0>(surface.VF[i])]);
    glNormal3fv(surface.VN[std::get<1>(surface.VF[i])]);
    glVertex3fv(surface.VV[std::get<1>(surface.VF[i])]);
    glNormal3fv(surface.VN[std::get<2>(surface.VF[i])]);
    glVertex3fv(surface.VV[std::get<2>(surface.VF[i])]);
}
glEnd();
```

交流/讨论

并未与任何同学进行交流讨论，也没有借鉴任何网上或其它同学的代码，只参考了课件和作业文档，并通过网络查阅了 B 样条和 Bezier 曲线的相关资料。

如何编译并渲染

在 `code/` 目录下使用如下命令：

```
bash run_all.sh
```

结果将存放在 `code/output/` 目录下。