计算机图形学

课程Project说明

COMP130018.01

简介

- 本学期共有三个Project：
- Project 1： 曲线和曲面造型技术， 占比**10%（今天发布）**
- Project 2： 光照模型与光线追踪， 占比**10%**
- Project 3： 神经辐射场NeRF， 占比**20%**
- 每组1-2人， 都需要在elearning上提交， 根据报告中的名字确定分组。

TA联系方式：

- 王韬洋 23210240050@m.fudan.edu.cn (负责Project 1)

实验环境

建议使用Ubuntu进行开发。可以使用提供的Ubuntu虚拟机镜像直接完成配置（链接: https://pan.baidu.com/s/14Us-IjcziXSrCFYMX7ZZjQ?pwd=4dau）。

Ubuntu（推荐）：

- sudo apt-get install build-essential cmake
- sudo apt-get install freeglut3-dev xorg-dev libxrandr-dev libsdl2-dev

实验环境也可自行在MAC OS以及WINDOWS下配置（不做讲解），但必须能够编译生成Ubuntu下的可运行代码，并且需要有图形化界面。

Windows：

- 安装Visual Studio，并选择C++开发套件，注意要选择Cmake

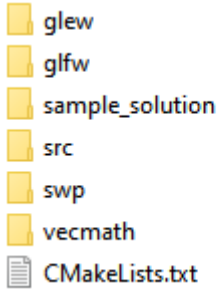
MacOS：

- 安装homebrew，cmake, xcode-build-tools
- brew install glfw3 glew

Project 1 曲线和曲面造型技术

- 在PJ1中，你将实现样条曲线和扫掠曲面来建模有趣的形状。主要目标是介绍样条曲线和坐标系统。在成功完成后，你将获得一个强大的工具来建模3D形状。
- 任务1：曲线的绘制
- 任务2：曲面的绘制

Project 1 代码框架



- 代码基于C++/OpenGL编写（暂时不需要了解OpenGL编程）

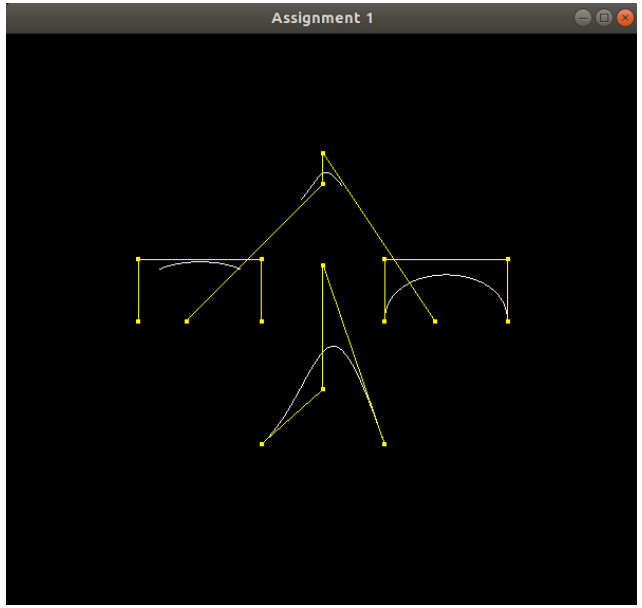
1. glew/glfw 为OpenGL的接口和插件库
2. vecmath 为向量计算库
3. sample_solution 为实现的最终结果样例（包含linux/athena, Windows和Mac）
4. src 为源代码
5. swp 为输入的形状文件

大部分功能已经为你实现，只需要在函数里填空就可以了。

Project 1 代码框架

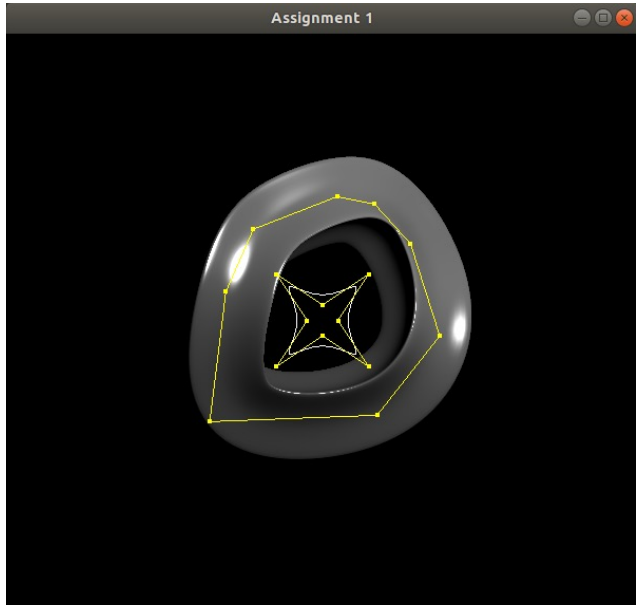
- 运行结果样例（以Ubuntu为例）

1. chmod a+x sample_solution/athena/a1
2. sample_solution/athena/a1 swp/core.swp



core.swp包含四条曲线。它们的控制点用黄色表示，所得到的曲线用白色表示。可以使用c键切换局部坐标的显示，也可以使用p键切换控制点的显示。

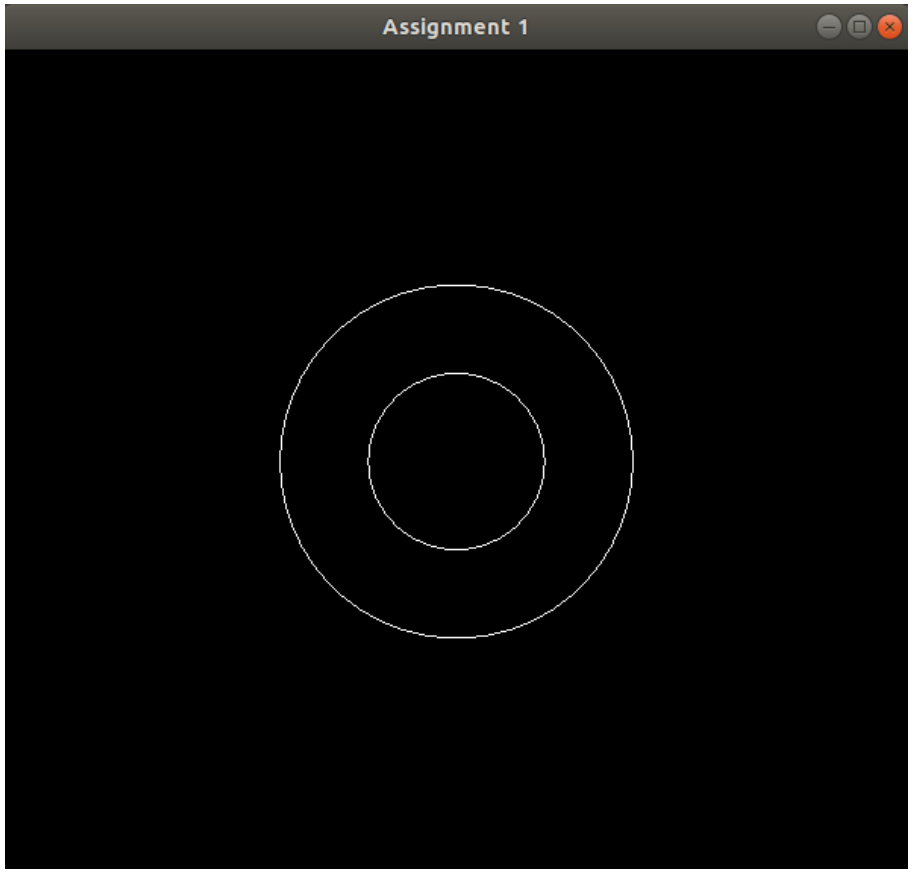
weird.swp显示了一个广义的圆柱体。你可以使用s键切换曲面的显示模式。默认情况下，它以平滑的着色开始，但你也可以关闭曲面以更好地查看曲线，或者使用法线渲染线框曲面。



Project 1 代码框架

- 编译初始代码

1. mkdir build
2. cd build
3. cmake ..
4. make
5. cd ..
6. build/a1 swp/circles.swp



Project 1 代码框架

- 初始代码的功能（已经实现）

1. 文件输入：你的程序必须以特定的文件格式（SWP）读取，该格式允许用户指定曲线(B´ezier、B-样条曲线和圆）和曲面（旋转曲面和广义圆柱面）。在SWP子目录中提供了许多SWP文件。文件格式的规范在所提供的解析器（parse.h）的头文件中给出。初始代码还包括许多代码必须能够读取和处理的SWP文件。
2. 用户界面：你的程序提供了使用鼠标输入旋转模型并选择曲线和曲面的显示模式的功能。可以使用c键切换局部坐标的显示，也可以使用p键切换控制点的显示，可以使用s键切换曲面的显示模式。
3. 向量计算库：vecmath下面实现了很多方便的向量计算算子。请务必仔细阅读代码，并充分利用现有的函数。

Project 1.1 曲线的绘制 - 要求

- 任务1要求:
- 你的程序必须能够生成和显示分段三次B´ezier和B样条曲线。此外，它必须正确计算局部坐标系（包括法向量N，切向量T，和次法线向量B）并显示它们。你应该将曲线渲染为白色，并将N、B和T向量分别渲染为红色、绿色和蓝色。
- 你需要在curve.cpp中填写evalB´ezier和evalBspline函数，显示将由初始代码处理。
- 请参考教材《计算机图形学》P210第10.5节B´ezier曲线和P225第106节B样条曲线或者《Fundamentals of Computer Graphics》15.6.1 Bézier Curves和15.6.2 B-splines。

Project 1.1 曲线的绘制 – 介绍

- 如下形式的多项式称为n次Bernstein 基函数:

$$\mathrm{BEZ}_{i,n}(t) = C_n^i t^i (1-t)^{n-i}, \qquad t \in [0,1]$$

- 如下形式的多项式曲线P(t) 称为n次Bezier 曲线:

$$P(t) = \sum_{i=0}^{n} P_i \mathrm{BEZ}_{i,n}(t), \quad t \in [0,1]$$

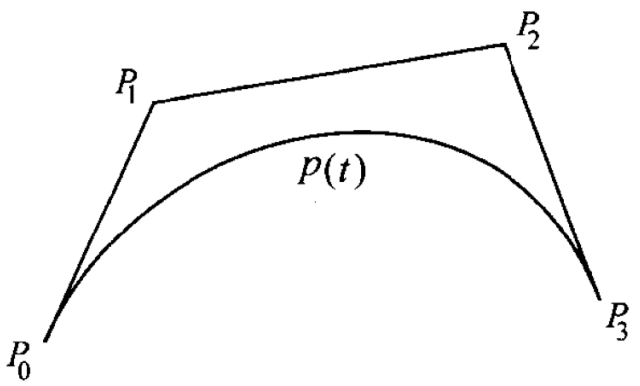


图 10.13 三次 Bezier 曲线 $P(t)$

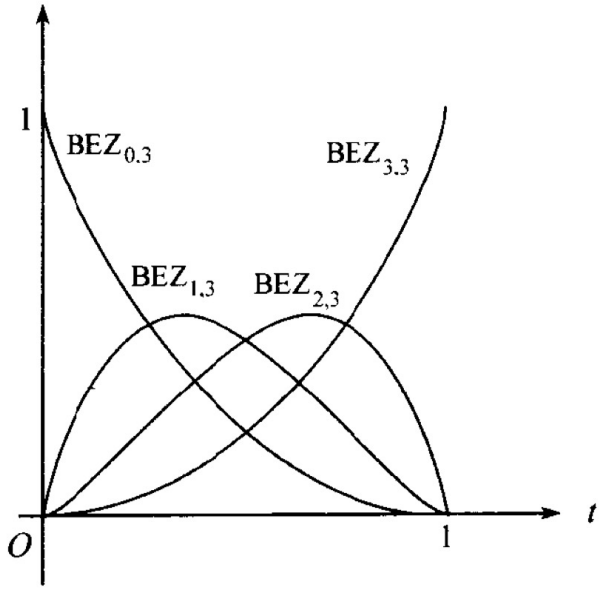


图 10.12 四个三次 Bernstein 基函数

Project 1.1 曲线的绘制 – 介绍

给定参数 t 轴上的节点分割 $T_{n,k}=\{t_i\}_{i=0}^{n+k}$，称由下列递推关系所确定的 $B_{i,k}(t)$ 为 $T_{n,k}$ 上的 k 阶(或 $k-1$ 次)B 样条基函数：

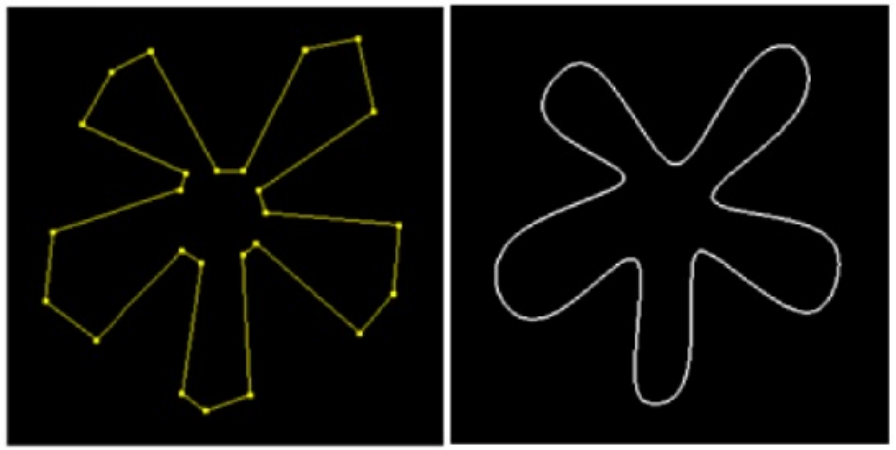
$$\begin{cases} B_{i,1}(t)=\begin{cases}1, & \text{当 } t\in[t_i,t_{i+1}),\\ 0, & \text{其它},\end{cases} & (10.69)\\ B_{i,k}(t)=\dfrac{t-t_i}{t_{i+k-1}-t_i}B_{i,k-1}(t)+\dfrac{t_{i+k}-t}{t_{i+k}-t_{i+1}}B_{i+1,k-1}(t), \quad i=0,1,\cdots,n & (10.70)\end{cases}$$

给定空间中 $n+1$ 个点 $\{P_i\}_{i=0}^{n}$ 及参数节点向量 $T_{n,k}=\{t_i\}_{i=0}^{n+k}(t_i\leqslant t_{i+1})$，称如下形式的参数曲线 $P(t)$ 为 k 阶($k-1$ 次)**B 样条曲线**。

$$P(t)=\sum_{i=0}^{n}P_iB_{i,k}(t), \quad t\in[t_{k-1},t_{n+1}] \quad (10.71)$$

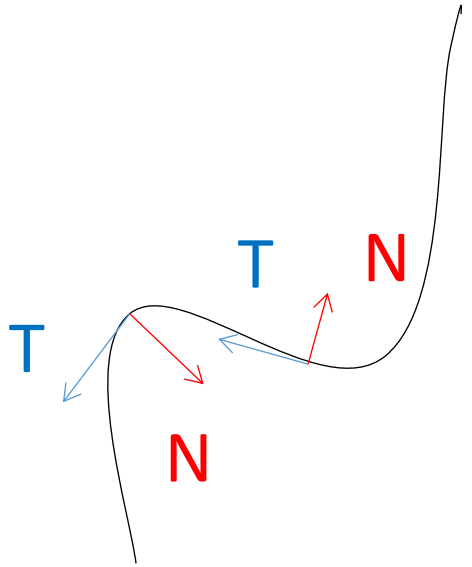
Project 1.1 曲线的绘制 - 介绍

- 第一个任务是实现分段三次B‘ezier和B样条曲线。给定一个控制点数组，你应该生成一组位于样条曲线上的点（然后可以通过将它们与线段连接来绘制样条曲线）。
- 例如，左图所示的控制点将导致右边的分段均匀三次b样条。在本例中，前三个控制点与后三个控制点相同，这生成了一条闭合曲线。



Project 1.1 曲线的绘制 - 二维曲线

- 如果唯一的目标是绘制样条曲线，那么我们根据公式计算点的坐标就足够了。但是对于动画和表面建模等其他应用程序，需要生成更多的信息（比如切向量和法向量）。我们将先从二维讨论这些细节，然后再到三维。
- 考虑一个简单的例子。假设我们想让一辆汽车在xy平面上绕曲线q(t)。q(t)告诉我们，在任何给定的时间，汽车应该在哪里，但不是汽车应该指向哪个方向。选择这个方向的一种很自然的方法是使用曲线的一阶导数： q’ (t)。
- 为了确定在某些t处的适当变换，我们引入了一些简记符号。首先，我们将位置V定义为 V=q(t)。我们将单位切向量T定义为T = q‘(t)/||q’(t)||。
- 当我们尝试将法向量N定义为一个与T正交的单位向量，其中一个满足这个性质的向量是N = T’/||T’||，也就是V的二阶导。但是这个定义是有问题的？



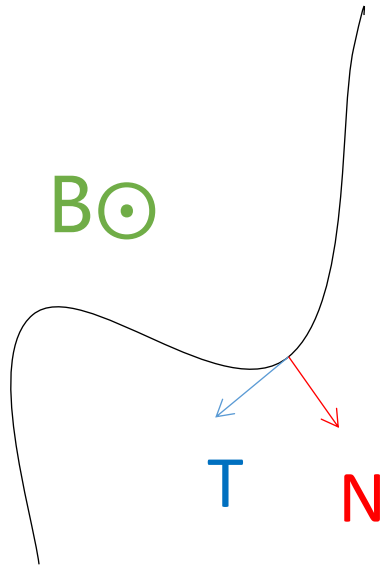
Project 1 曲线的绘制 - 二维曲线

右手定则!

- 如果我们假设汽车是指向其正的y轴的，那么处理汽车坐标系的适当的齐次变换可以计算为：

$$\mathbf{M} = \begin{bmatrix} \mathbf{N} & \mathbf{T} & \mathbf{V} \\ 0 & 0 & 1 \end{bmatrix}$$

- 但是这个公式有一个问题。首先，T的导数并不总是指向曲线的“法方向”。在二维中，如果我们有一个切向量T，那么于T垂直的向量可以很容易地通过将T旋转90度得到而不是对T求导。其次，对于切向量的变化率T‘ 来说，其方向并不保证是垂直于切向量T的。许多情况下T’可能与切线平行(例如，在曲线的直线段上), 或者甚至可能为0 （在曲线的某些特殊点上，如拐点）
- 对于平面运动，我们可以引入一个新的与T正交的向量，我们称之为B，这被称为次法线，我们将随意选择它指向正的z方向。给定B，我们可以计算一个备选法线为：N=B×T。
- 在平面中，这个定义与前面定义的N的符号一致。注意，N是单位向量，并且与B和T都正交，因为B和T都是正交的单位向量。



Project 1.1 曲线的绘制 - 三维曲线

- 为了在三维中找到合适的坐标系，我们不能使用选择一个固定的次法线B的技巧。一个原因是曲线可能会向次法线的方向旋转（即，可能发生T = B）。在这种情况下，正常的N变得未定义，我们就失去了坐标系。
- 所以，我们将依赖这样一个事实，即我们将沿着q (t)沿着曲线生成离散点。换句话说，我们可以沿着第一步t1 = 0，第二步t2 = 0.1等等的q (t)前进。在第i步中，我们可以计算以下值（就像二维情况一样）：

$$\mathbf{V}_i = \mathbf{q}(t_i)$$
$$\mathbf{T}_i = \mathbf{q}'(t_i).\text{normalized}()$$

- 为了选择Ni和Bi，我们使用以下递归更新方程：

$$\mathbf{N}_i = (\mathbf{B}_{i-1} \times \mathbf{T}_i).\text{normalized}()$$
$$\mathbf{B}_i = (\mathbf{T}_i \times \mathbf{N}_i).\text{normalized}()$$

- 在这些方程中，normalized()是类Vector3f的一种方法，该方法返回实例的单位长度副本（即，v.normalized()返回v/||v||）。
- 我们可以通过选择一个任意的B0来初始化递归（不能平行于T1，可以使用比如(0,0,1)×T1）。然后可以将其插入更新方程来选择N1和B1。
- 也就是说，这个递归更新允许Ti处的法向量尽可能接近Ti−1处的法向量，同时仍然保留法线与正线正交的必要性质。

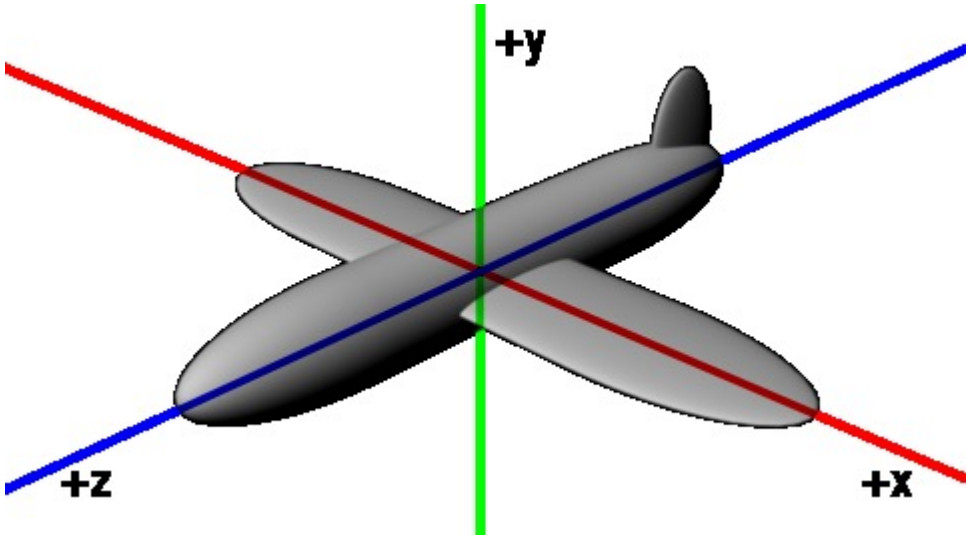
Project 1.1 曲线的绘制 - 三维曲线

- 坐标的几何意义
- 与二维情况一样，我们可以使用这三个向量来在沿曲线的每个点上定义一个局部坐标系。所以，假设我们想让右边这架飞机沿着我们画的曲线动起来。
- 我们只需要让飞机的z轴与切线T对齐，x轴与法线N对齐，y方向与次法线的B对齐，并让坐标系原点在位置V处。这可以用下面的变换矩阵来实现：

$$M = \begin{bmatrix} N & B & T & V \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

R　　t

- 其中N，B，T组成了旋转矩阵，V则是平移矩阵。



Project 1.1 曲线的绘制 - 代码实现

- 以圆为例，我们需要返回一个Curve对象，它包含step+1个点。（对于曲线来说则是段数*step+1个点）每个点都需要提供以下几个值。

1. V表示顶点的位置。
2. T表示顶点的切线，是位置的导数。
3. N表示顶点的法线，对于圆来说是二阶导数，对于曲线则需要刚刚提到的递归更新方法。
4. B表示顶点的次法线，对于圆来说定义为向上，对于曲线则需要刚刚提到的递归更新方法。

```
Curve evalCircle(float radius, unsigned steps)
{
    // This is a sample function on how to properly initialize a Curve
    // (which is a vector< CurvePoint >).

    // Preallocate a curve with steps+1 CurvePoints
    Curve R(steps + 1);

    // Fill it in counterclockwise
    for (unsigned i = 0; i <= steps; ++i)
    {
        // step from 0 to 2pi
        float t = 2.0f * c_pi * float(i) / steps;

        // Initialize position
        // We're pivoting counterclockwise around the y-axis
        R[i].V = radius * Vector3f(cos(t), sin(t), 0);

        // Tangent vector is first derivative
        R[i].T = Vector3f(-sin(t), cos(t), 0);

        // Normal vector is second derivative
        R[i].N = Vector3f(-cos(t), -sin(t), 0);

        // Finally, binormal is facing up.
        R[i].B = Vector3f(0, 0, 1);
    }

    return R;
}
```

Project 1.1 曲线的绘制 - 代码实现

- 三次贝塞尔曲线（教材《计算机图形学》P210第10.5节B´ezier曲线）

$$P(t) = G_{\mathrm{BEZ}} \cdot M_{\mathrm{BEZ}} \cdot T = [P1, P2, P3, P4] \cdot \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}$$

$$P(t)' = [P1, P2, P3, P4] \cdot \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 2t \\ 3t^2 \end{bmatrix}$$

Project 1.1 曲线的绘制 - 代码实现

- 三次B样条曲线（教材《计算机图形学》P225第106节B样条曲线）

$$P(t) = G_{Bj} \cdot M_B \cdot T_j = [P_{j-3}, P_{j-2}, P_{j-1}, P_j] \frac{1}{6} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ t-j \\ (t-j)^2 \\ (t-j)^3 \end{bmatrix}$$

$$P(t)' = [P_{j-3}, P_{j-2}, P_{j-1}, P_j] \frac{1}{6} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 2(t-j) \\ 3(t-j)^2 \end{bmatrix}$$

Project 1.1 曲线的绘制 - 代码实现

- 贝塞尔曲线与B样条曲线的关系：将B样条曲线中令s=t-j可以得到类似贝塞尔曲线的形式，我们只需要对B样条曲线中每一段的控制点G_{Bj}乘上$M_B M^{-1}_{BEZ}$然后按贝塞尔曲线生成即可。

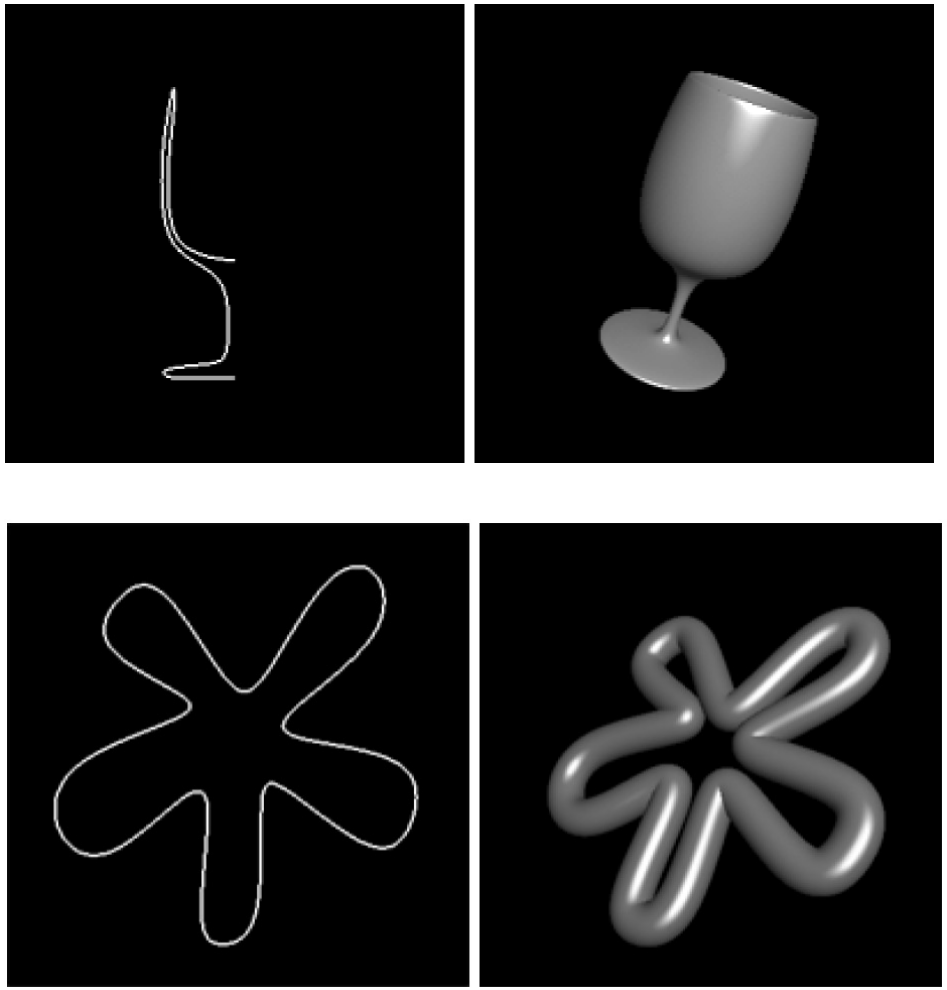
$$P(s) = G_{Bj} \cdot M_B \cdot \begin{bmatrix} 1 \\ s \\ s^2 \\ s^3 \end{bmatrix} = \left(G_{Bj} \cdot M_B \cdot M^{-1}_{BEZ}\right) \cdot M_{BEZ} \cdot \begin{bmatrix} 1 \\ s \\ s^2 \\ s^3 \end{bmatrix}$$

Project 1.2 曲面的绘制 - 要求

- 任务2要求:
- 你的程序必须能够生成和显示旋转曲面（围绕y轴的xy平面上的曲线）和广义圆柱面。它必须正确地计算曲面法线。要在程序中演示这一点，你应该有两种显示模式。一种模式应该显示以线框模式绘制的曲面，并将顶点法线从曲面向外绘制。另一个应该使用平滑的阴影来显示表面，显示功能已经在初始代码中为你实现了。
- 你只需要在surf.cpp中填写makeSurfRev和makegenCyl函数。对于广义圆柱体，你不需要精确地与sample_solution一致，因为它会任意选择初始坐标系（B0）。

Project 1.2 曲面的绘制 - 介绍

- 你将使用生成的曲线来创建扫掠（swept）曲面。具体来说，你将处理的曲面类型是旋转曲面和广义圆柱体。
- 上面的图片显示了一个旋转曲面的例子。左边是xy平面上的轮廓曲线，右边是绕y轴扫过表面的结果。
- 下面的图片显示了一个广义圆柱体的例子。在左边是我们所说的扫掠曲线，并且曲面是通过沿着扫掠曲线扫掠轮廓曲线来定义的。在这里，轮廓曲线被选择为一个圆，但你的实现也应该支持任意的二维曲线。



Project 1.2 曲面的绘制 - 旋转曲面

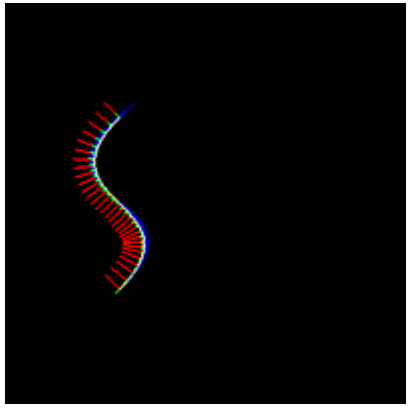
- 我们将旋转曲面定义为围绕正y轴逆时针在xy平面上扫扫曲线的乘积，其中旋转的具体方向很重要。
- 假设你已经找到了沿着轮廓曲线的控制点。然后，可以通过以均匀的旋转采样值，来复制计算的曲线点来生成曲面的顶点。这是在实现过程中的第一个任务。
- 但是，顶点本身并不能定义曲面。我们还需要定义法线和面。这就是事情变得更具挑战性的地方。
- 这里涉及到了齐次坐标的旋转和平移变换，请参考教材《计算机图形学》P130第7.8节三维几何变换或者《Fundamentals of Computer Graphics》5 Linear Algebra，6 Transformation Matrices。

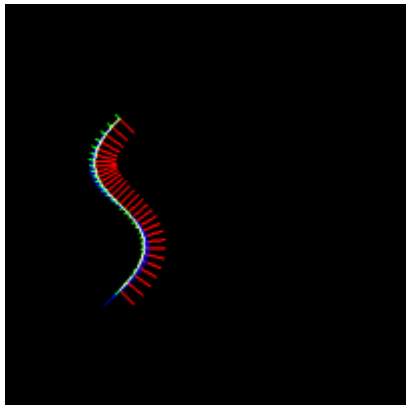
Project 1.2 曲面的绘制 - 旋转曲面

- 首先，让我们讨论一下法线。从曲线的计算中，我们已经得到了法向量N。所以，我们可以用和顶点相同的变换来旋转这些法向量，对吧？是的，但不完全是。结果表明，如果我们用齐次变换矩阵M变换一个顶点，它的法线应该通过M的左上角3×3子矩阵的逆转置进行变换。（推导详见https://zhuanlan.zhihu.com/p/110520337）

$$\mathbf{M} = \begin{bmatrix} \mathbf{N} & \mathbf{B} & \mathbf{T} & \mathbf{V} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

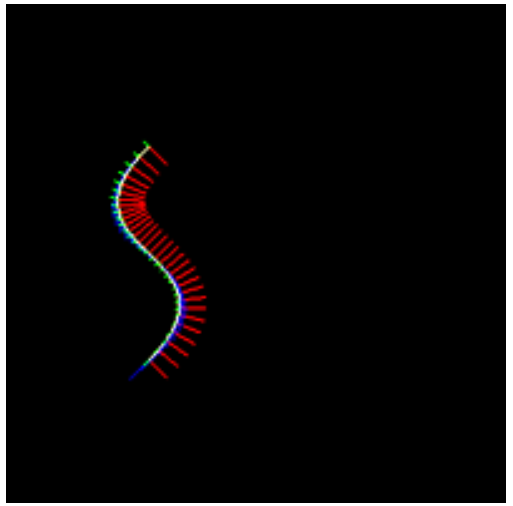
- 你需要担心的另一件事是法线的方向。为了使OpenGL执行适当的照明计算，法线需要从表面射出（facing out）。所以，你不能只是盲目地旋转任意曲线的法线，并期望能得到正确结果。
- 要理解这个问题，请观察以下B‘ezier曲线。它们之间的区别是，法线（红线）是颠倒的。这是反转控制点顺序的结果。换句话说，即使曲线是相同的，法线也取决于遍历的方向。

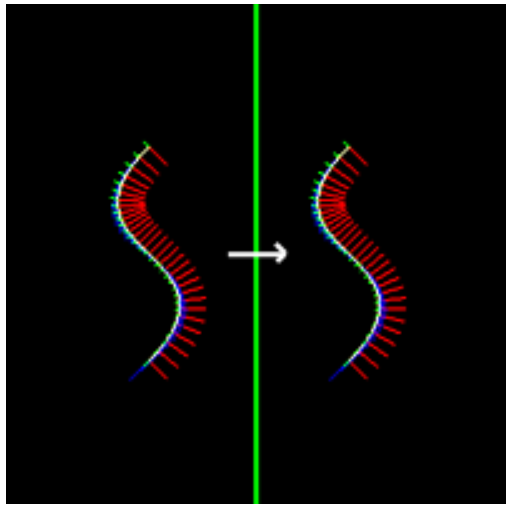




Project 1.2 曲面的绘制 - 旋转曲面

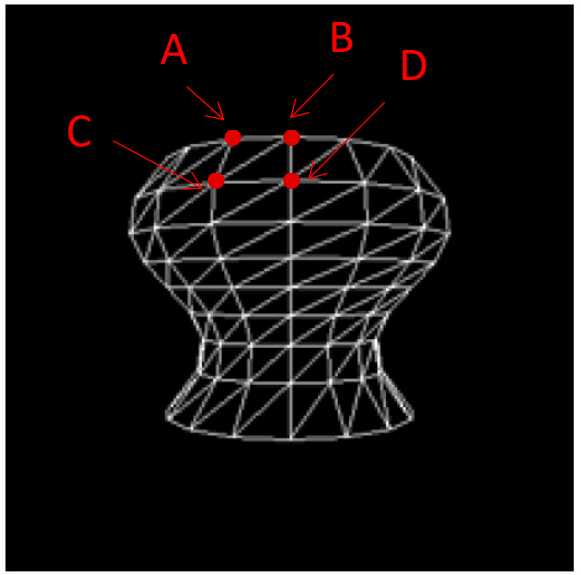
- 在这个任务中，我们将假设，对于二维曲线，法线（红线）将总是指向遍历方向（蓝线）的左边，像右图这样（N=B×T）。因为当你在逆时针方向画圆时，法线也是始终指向原点。请注意，如果你实现了我们前面描述的二维曲线，那么你将自动得到此行为（因为B指向z轴正方向，从平面向外）。
- 在此假设下，在将曲线法线应用于旋转曲面时，我们需要反转曲线法线的方向得到曲面的法线（法线反向），**使得曲面法向量向外**。也就是上图所表示的，从上至下的曲线的法线指向旋转轴，但是曲面的法线则远离旋转轴。
- 而且平移曲线可能会破坏我们的假设。考虑一下，如果我们只在左边取一条曲线，然后平移它，使它在y轴的另一边，会发生什么？如下图所示（y轴是绿色的粗线）。在这里，曾经朝向y轴的法线现在朝向了外面！与其试图处理这两种情况，所以我们将假设轮廓曲线总是在y轴的左边（也就是说，定义的旋转曲面的曲线上的所有点都将有一个非正的x坐标）。
- 总而言之，记得反转曲线法线的方向才能得到曲面的法线。





Project 1.2 曲面的绘制 - 旋转曲面

- 除了定义法线我们还需要定义面。你的任务是生成三角形，连接轮廓曲线重复的曲线，如右图所示。基本的策略是在相邻的重复之间来回曲折来构建三角形。
- 在OpenGL中，你需要按照特定的顺序指定顶点和绘制法向量。它们必须按逆时针的顺序形成一个三角形（假设你看的是三角形的前面）。如果你反向生成你的三角形，你的三角形将有不正确的照明计算，或者更糟的是，根本不会出现三角形。这也是为什么，我们之前假设了曲线是围绕y轴逆时针旋转的，这样就可以和上一个曲线生成三角形。
- 如果你不确定自己实现法线和三角形正不正确，请运行示例代码sample_solution，并按c或者s切换表示，来对比自己的结果。



针对曲线AC和BD构成的三角形为：ACB,BCD，**法向量应该朝外。**

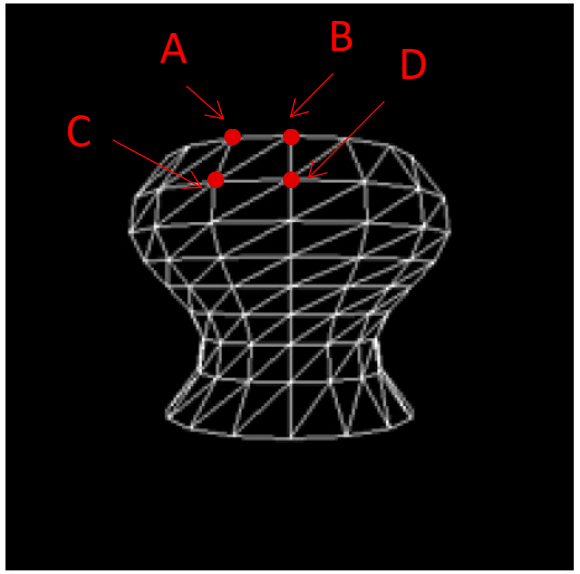
Project 1.2 曲面的绘制 - 旋转曲面

- 旋转曲面（教材《计算机图形学》P130第7.3节齐次坐标与二维变换的矩阵表示）

$$M = R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

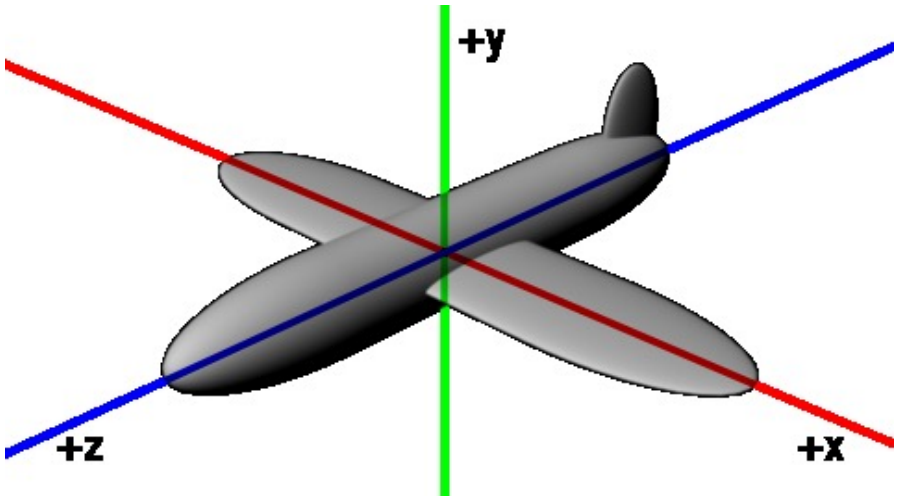
$$P' = M \cdot P$$

$$N' = normalize((M^{-1})^T N)$$



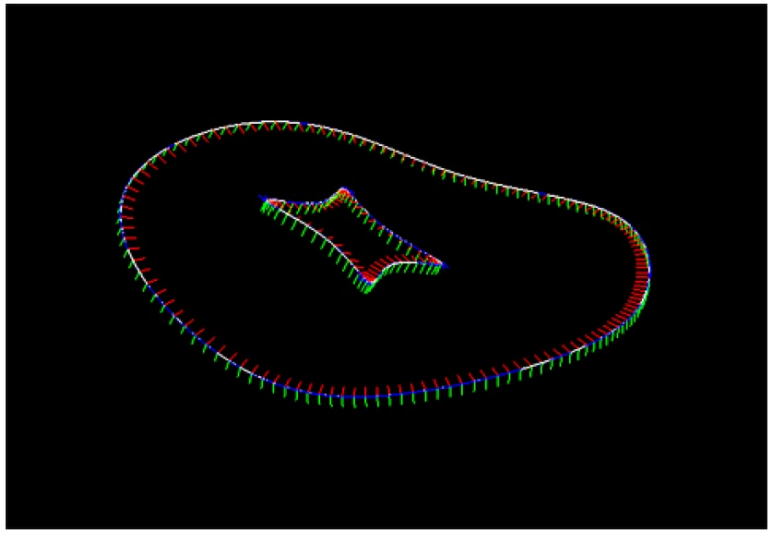
Project 1.2 曲面的绘制 - 广义圆柱体

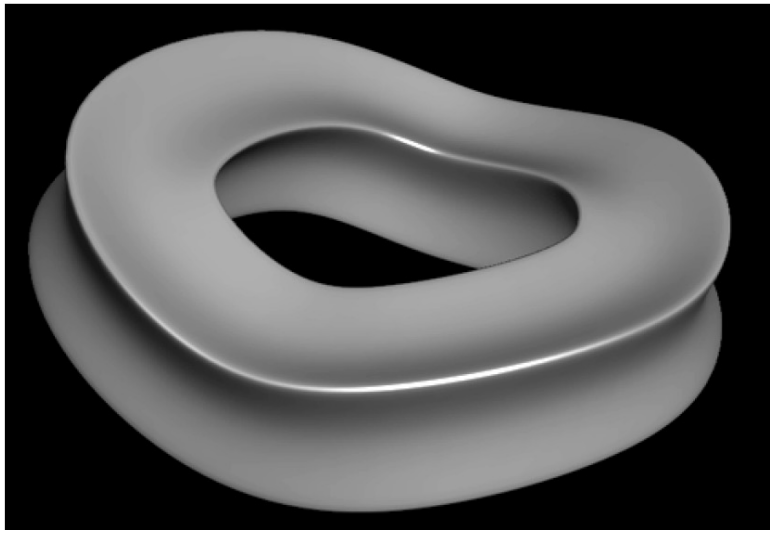
- 广义圆柱体是通过重复轮廓曲线并用三角形连接轮廓曲线的每个副本而形成的。它和旋转曲面不同之处在于，你将不是沿着y轴扫过二维轮廓曲线，而是沿着三维扫描曲线扫过它。（类似飞机飞过扫描曲线）
- 就像旋转曲面一样，轮廓曲线的每个副本都是独立转换的。对于有旋转曲面，我们使用了一个旋转变换。而对于广义圆柱体，我们将使用由扫描曲线的N、B、T和V向量定义的坐标系。为了把它放在前面的讨论中，想象一下在沿着扫描曲线飞行的飞机后面拖动一个轮廓曲线的副本，从而留下一个表面的痕迹。



Project 1.2 曲面的绘制 - 广义圆柱体

- 选择正确的法线和面的过程与选择旋转曲面的过程非常相似。我们建议你将旋转曲面里面三角形网格划分代码作为一个单独的函数来编写，以便它可以被重用在广义圆柱体中。
- 这是一个广义圆柱体的例子。首先，让我们看看轮廓曲线和扫描曲线。这两者都用曲线上的点上的局部坐标系表示。具体来说，蓝线是T，红线是N，绿线是B。
- 小曲线为轮廓曲线，大曲线为扫描曲线。所得到的广义圆柱体如下图所示。





Project 1.2 曲面的绘制 - 广义圆柱体

- 广义圆柱体：将轮廓曲线（Profile）按扫描曲线（Sweep）的坐标系（NBTV）进行变换。

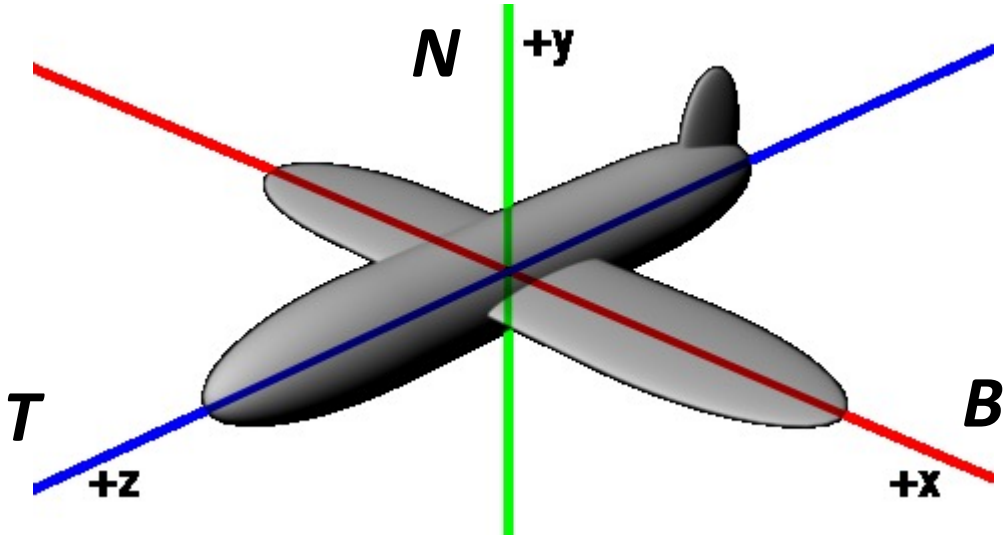
$$M = \begin{bmatrix} \mathbf{N} & \mathbf{B} & \mathbf{T} & \mathbf{V} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

R

t

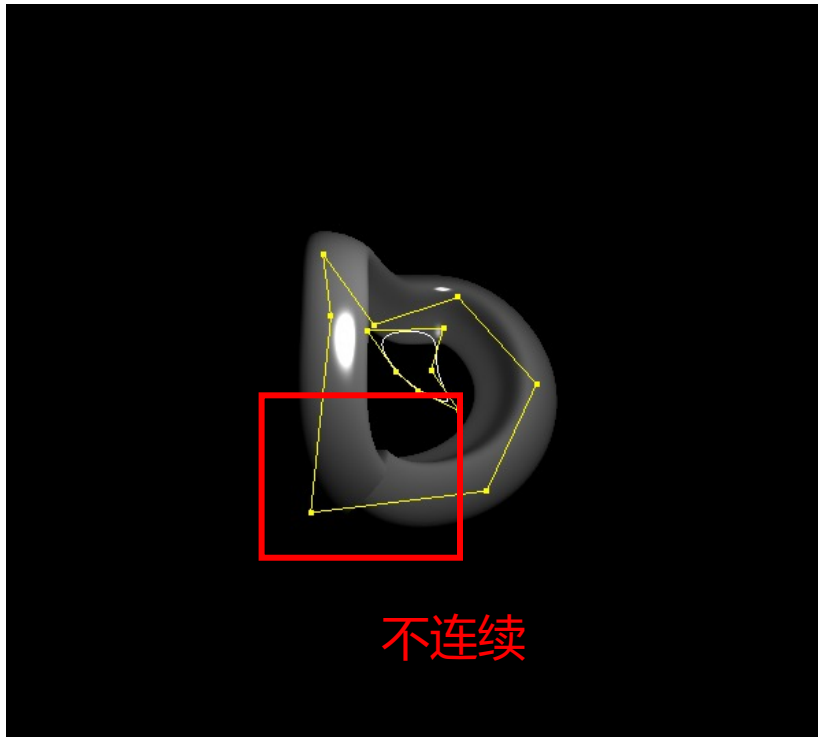
$$P' = M \cdot P$$

$$N' = normalize((M^{-1})^T N)$$



Project 1 拓展 - 曲面的闭合问题

- 以上的计算坐标系的方法存在一个问题：如果一条曲线是闭合的，那么坐标系就不能保证在曲线相交的地方对齐。
- 我们可以考虑如下方法解决这个问题。首先，它检测何时发生这些错误（当扫描曲线开始和结束位置和切线相同，但法线不同）。然后，它插值扫描曲线的开始和结束之间的旋转差值，并将其添加到沿着曲线的坐标系中。
- 请在代码中实现此解决方案。完成之后，你应该能够显示一个无缝的weirder.swp。你还应该确保你的解决方案不会影响其他未闭合的曲线的结果。



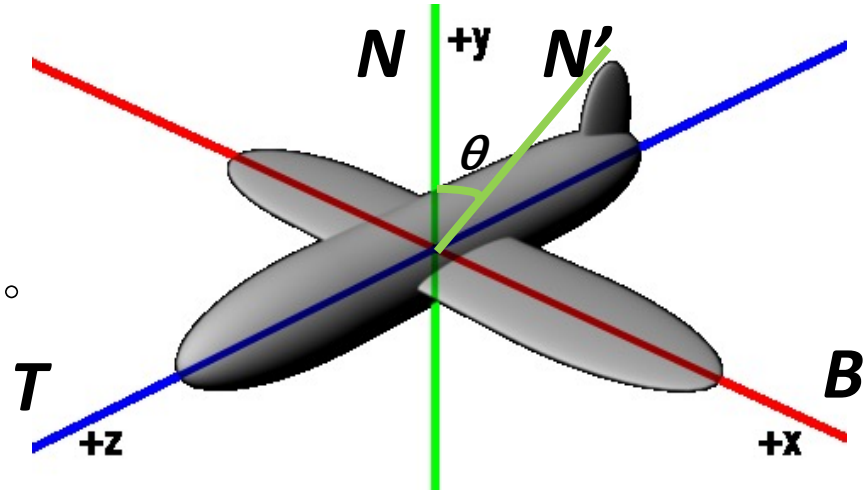
Project 1 拓展 - 曲面的闭合问题

- 插值解决闭合问题：我们将扫掠曲线（Sweep）的法向量（N）和次法向量（B）绕切向量（T）旋转θ=α/surf_size度，其中α为曲线起始法向量与结束法向量的夹角。
- 我们可以使用罗德里格旋转公式，设v是一个三维空间向量，k是旋转轴的单位向量，则v在右手螺旋定则意义下绕旋转轴k旋转角度θ得到的向量可以由三个不共面的向量v, k和k×v构成的标架表示。

$$\mathbf{v}_{rot} = \cos\theta\mathbf{v} + (1-\cos\theta)(\mathbf{v}\cdot\mathbf{k})\mathbf{k} + \sin\theta\mathbf{k}\times\mathbf{v}$$

- 将N带入v，T带入k，那么B=k×v，公式可以简化为如下。

$$N' = cos\theta\mathrm{N} + sin\theta\mathrm{T}\times\mathrm{N} = cos\theta\mathrm{N} + sin\theta\mathrm{B}$$



- 如右图所示，也可以根据几何意义直接推出以上公式，N’在<N，B>平面上，与N差距为θ度。那么直接可以有向量N’=cosθN+sinθB。
- 换句话说，我们不是简单地在两个法向量之间跳跃，而是制造一个渐变的过程，其中法向量在曲线闭合的两端缓慢地从起点过渡到终点向量。

Project 1 参考资料

请务必仔细阅读初始代码，特别是src和vecmath，代码是最好的参考资料！
如果不清楚应该实现成什么样，请运行sample_solution下面的程序！

参考网站

- C++教程：https://www.cprogramming.com/tutorial/
- OpenGL教程：https://learnopengl.com

参考书籍：

- 计算机图形学 / 倪明田，吴良芝编著 北京 : 北京大学出版社, 1999
- Shirley, Peter, Michael Ashikhmin, Steve Marschner. Fundamentals of Computer Graphics. 3rd ed. A K Peters/CRC Press, 2009. ISBN: 9781568814698. (计算机图形学/虎书)
- Buss, Samuel R. 3D Computer Graphics: A Mathematical Introduction with OpenGL. 2003. ISBN: 9780521821032. (3D计算机图形学（OpenGL版）)

参考课程：

- MIT 6.837:

https://ocw.mit.edu/courses/6-837-computer-graphics-fall-2012/

- GAMES101: 现代计算机图形学入门

https://sites.cs.ucsb.edu/~lingqi/teaching/games101.html

Project 1 评分细则

项目完成度及正确性：（共计5分）

1. 曲线的绘制：正确编译并输出每个文件的曲线**（必做，每个样例正确显示得0.3分，共计3分）**

2. 曲面的绘制：正确编译并输出每个文件的曲面（每个样例正确显示得0.2分，共计1分）

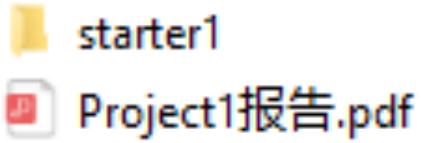
3. 解决闭合问题：weirder.swp文件会出现首尾闭合问题请用插值解决（共计1分）

项目报告：（共计5分）

撰写项目报告，**附上所有生成的图形的截图（报告中请尽量使用高分辨率图像）**，详细说明曲线的绘制，曲面的绘制以及曲面相交问题的解决方式**（必做，5分）**

Project 1 提交方式

- 可编译项目代码以及Linux编译程序（starter1文件夹）及Project1报告（PDF）请打包（zip）并上传elearning
- 邮件/压缩包标题：2024图形学PJ1 姓名1 姓名2
- **项目报告DDL与Project2发布: 2024年4月18日23:59**



- 若对Project有疑问，可邮件/微信与TA联系
- TA办公地址：江湾校区交叉学科2号楼A4008室

- **严禁抄袭，包括网络上和同学的代码，一经发现Project作0分处理**
- **只实现必做功能也一定可以顺利通过，不要铤而走险，迟交会酌情扣分**