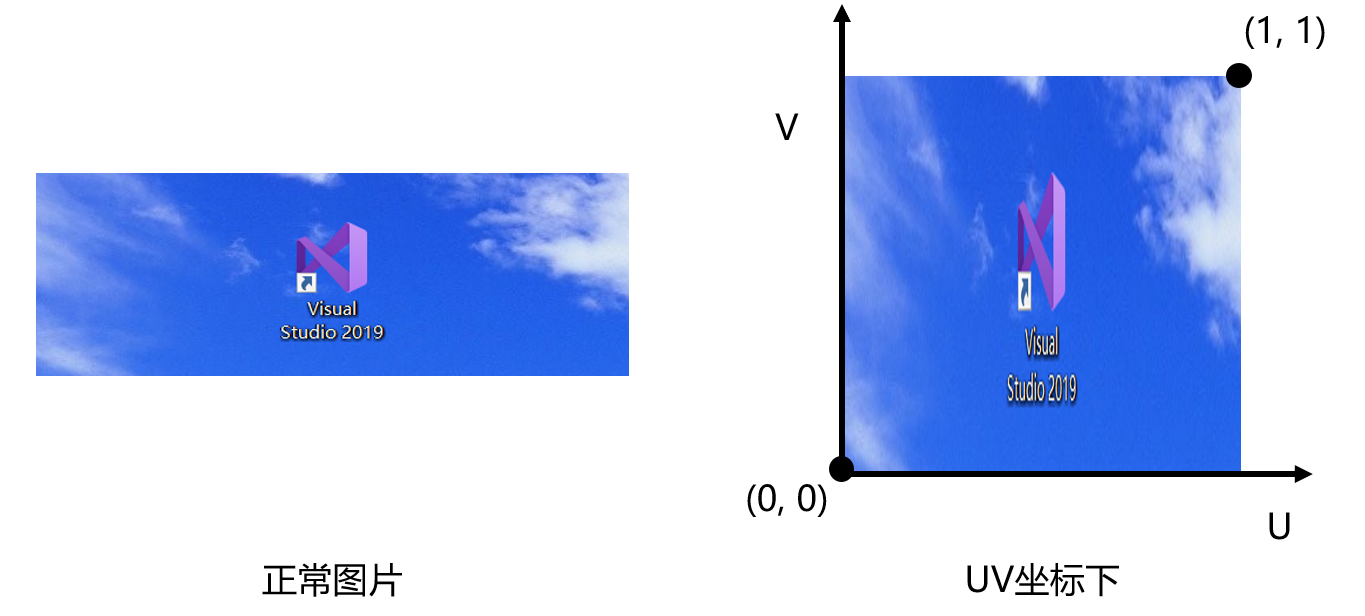
实验4.1 简单可扩展曲面纹理映射

1. 实验内容
2. 了解三维曲面和纹理映基本知识
3. 了解从图片文件载入纹理数据基本步骤
4. 掌握三维曲面绘制过程中纹理坐标和几何坐标的使用
5. 理论背景
6. UV坐标系

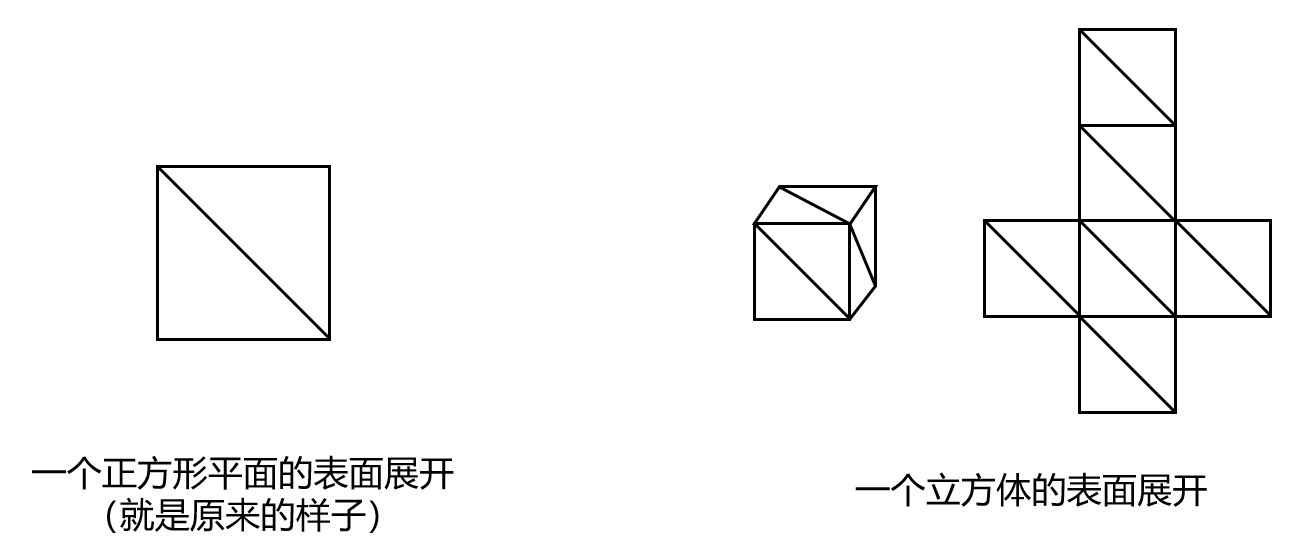
对于三维模型，有两个最重要的坐标系统，一是顶点的位置（X，Y，Z）坐标，另一个就是UV坐标。U和V分别是图片在显示器水平、垂直方向上的坐标，取值一般都是0~1，也 就是（水平方向的第U个像素/图片宽度，垂直方向的第V个像素/图片高度）。



纹理实际上是一个二维数组，它的元素是一些颜色值。单个的颜色值被称为纹理元素（texture elements）或纹理像素（texel）。每一个纹理像素在纹理中都有一个唯一的地址。这个地址可以被认为是一个列（column）和行（row）的值，它们分别由U和V来表示。

1. 网格的表面展开

想将二维纹理图片映射到三维的物体表面，我们首先要想象一下物体表面的三角网格摊开成平面后的样子，比如下面展示一个正方形平面与一个立方体表面展开到一个平面的效果。



1. 纹理映射

在使用opengl和shader显示纹理时候需要对每一次面片的绘制过程中对面片的每一个顶点指定纹理坐标。根据物体表面展开，我们可以建立顶点到UV坐标的映射关系，从而推算出顶点对应的纹理坐标。完成纹理映射需要以下几个操作：

* 首先要有纹理图片，然后读取它的数据
* 设置好物体每个顶点的纹理坐标
* 将纹理坐标、图像数据传递给着色器
* 片元着色器内根据纹理坐标和图像数据给像素着色

1. 实验内容
2. 读取纹理图片

实验使用stb\_image.h 的单头文件图像加载库，它能够加载大部分流行的文件格式（已经下载到include文件夹内了），下图介绍了它里面最重要的一个函数stdi\_load，我们可以用一个unsigned char类型变量接收返回值。

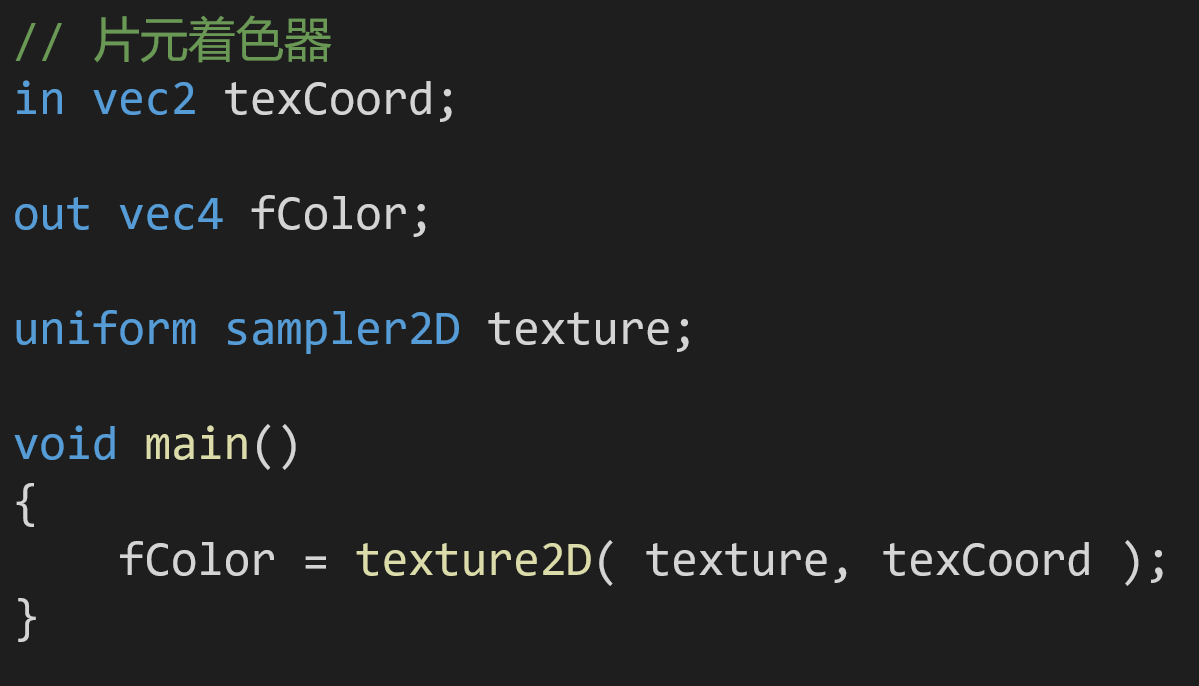


纹理坐标是二维的坐标，所以我们会使用 vec2 类型存储，既然作为顶点的属性之一，我们把它加入到TriMesh类中，然后在那些生成顶点数据（比如generateCube）的函数内存储这些顶点对应的纹理坐标。



每个顶点的纹理坐标是一个二维坐标，我们使用 vec2 记录它，所以在顶点着色器内，和 vPosition 那些变量一样，我们设置了一个叫 vTexture 的变量，接收纹理坐标，然后将其传递给片元着色器，因为纹理颜色的计算要在片元着色器内进行。

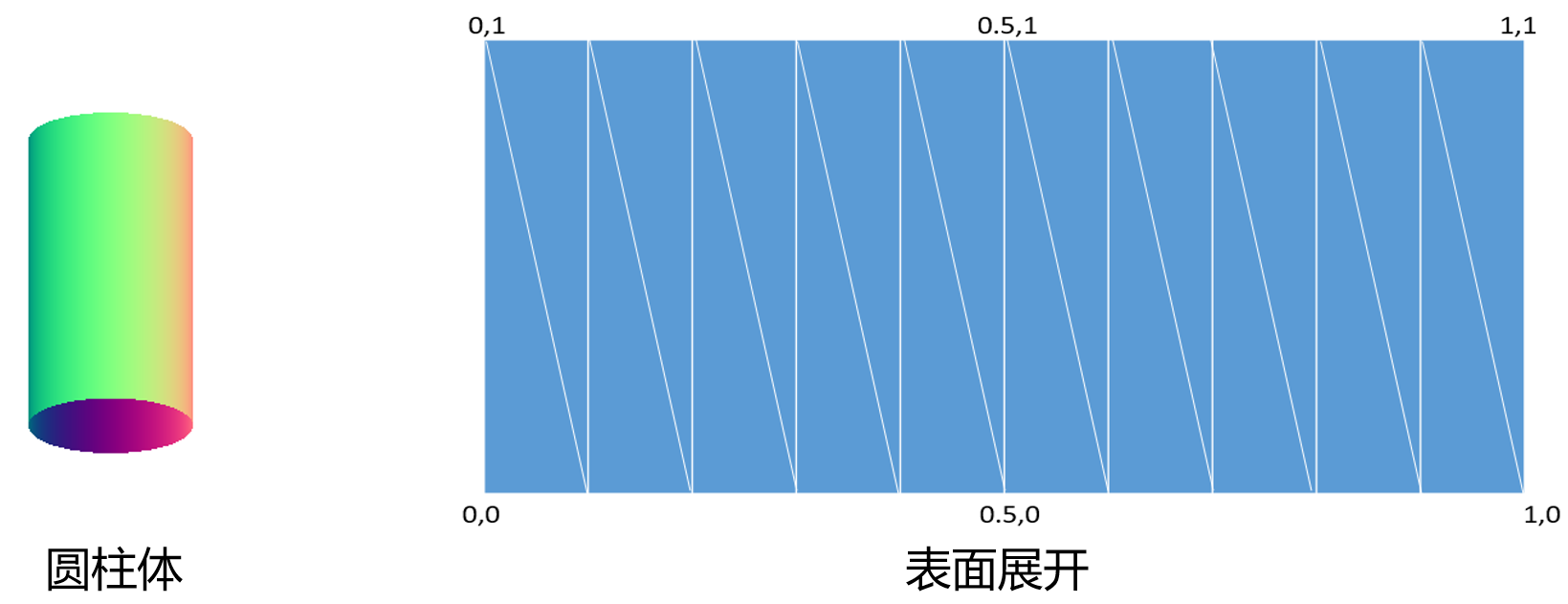
片元着色器内接收来自顶点着色器的纹理坐标 textCoord，然后我们定义一个特殊类型的着色器变量 texture ，它的数据类型叫：采样器 sampler2D，后缀的2D表示它是处理二维纹理的采样器，它会接收应用程序传递过来的纹理图像数据。然后使用函数 texture2D 计算每个片元对应的像素颜色，输入参数为纹理采样器和纹理坐标。



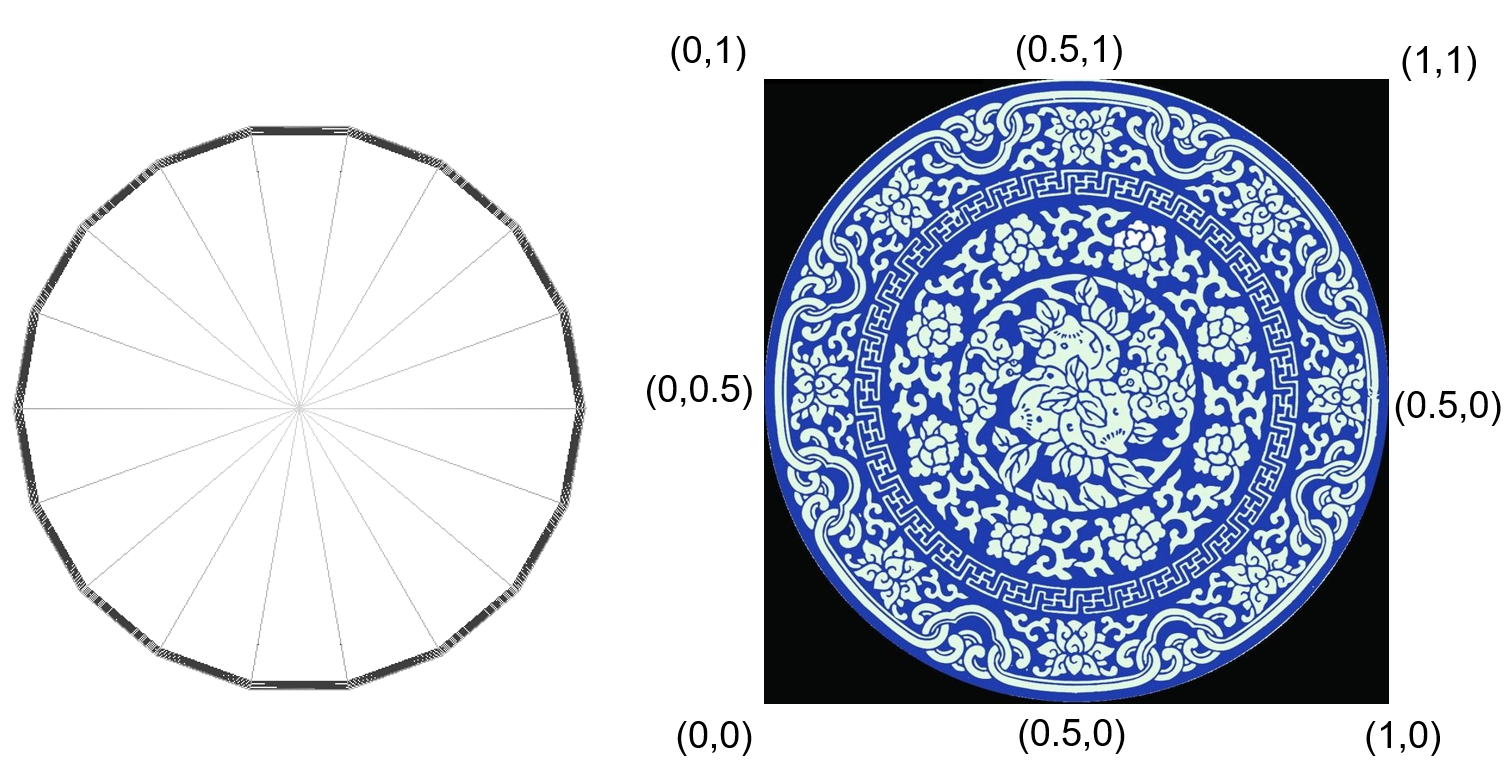
在MeshPainter.cpp中，参照 vPosition 那些变量的传递方法，将纹理坐标相关代码补全，注意纹理坐标是二维的。



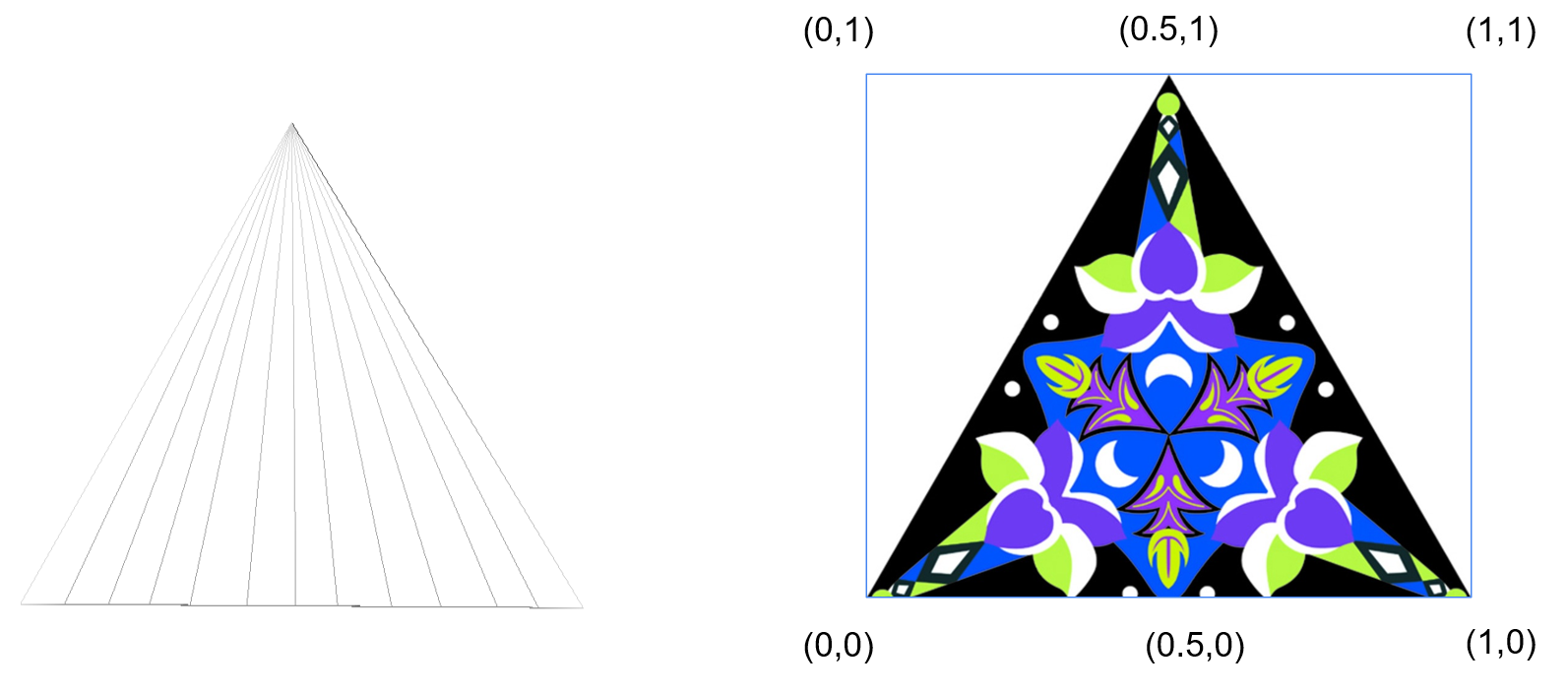
1. 更多物体的纹理坐标设置
   1. 实验提供的代码中已经在TriMesh.cpp中实现了圆柱体的顶点和纹理坐标生成，函数名为 generateCylinder，并在main.cpp的init函数中创建了圆柱体。其表面展开如下，结合图片理解代码实现的逻辑。



* 1. 参考圆柱体的函数写法，以及圆盘的表面展开，将TriMesh.cpp的generateDisk函数完成。并在main.cpp文件的init函数中将圆盘创建出来。圆盘的纹理图片在assets文件夹中，叫disk.jpg。

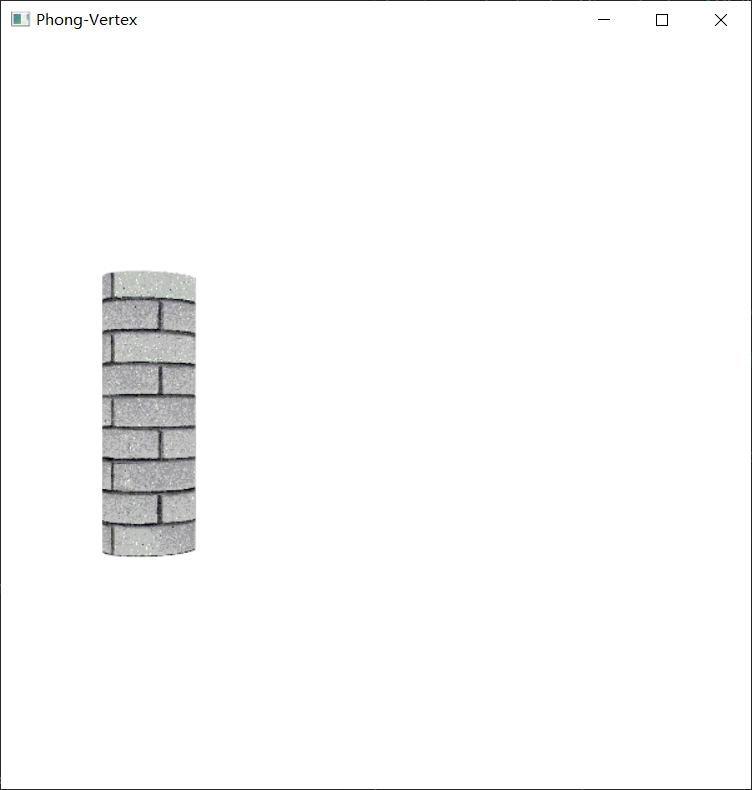


* 1. 参考圆柱体的函数写法，以及圆锥的表面展开，将TriMesh.cpp的generateCone函数完成。并在main.cpp文件的init函数中将圆锥创建出来。圆锥的纹理图片在assets文件夹中，叫cone.jpg。



1. 示例结果
2. 下面分

1. 实例代码中已经实现了TriMesh::generateCylinder()函数，该函数生成一个圆柱体曲面，将该曲面贴上纹理后效果如下：



2. 实现void TriMesh::generateDisk()；void TriMesh::generateCone()；生成圆盘的几何和纹理坐标，生成圆锥的几何和纹理坐标，使得圆盘表面纹理显示为assert / disk.jpg中的圆盘；使得圆锥表面纹理显示为assets / cone.jpg中的三角图形。实验要求达到的参考效果如下：

