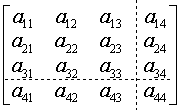
实验2.3 三维模型的平移、缩放和旋转

1. 实验目的
2. 掌握三维模型顶点与三角面片之间关系。
3. 了解和掌握三维模型的基本变换操作。
4. 掌握在着色器中使用变换矩阵。
5. 理论背景

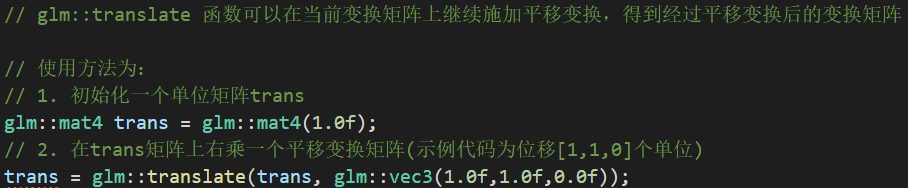
采用齐次坐标表示的三维几何变换矩阵是一个4阶方阵，形式如下：



1. 平移变换

平移变换把点沿着给定的方向移动固定的距离。参照二维的平移变换，可得到三维平移变换矩阵。平移变换过程如下：

glm库中translate()可以在当前变换矩阵上继续施加平移变换，即右乘平移变换矩阵，得到经过平移变换后的矩阵，用法如下：



注意：示例代码中的当前变换矩阵为重新初始化的单位矩阵，在实际使用时当前变换矩阵不一定为单位矩阵。

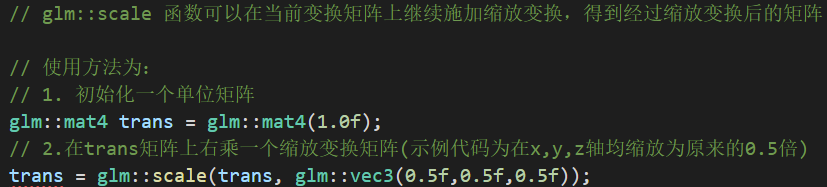
1. 缩放变换

缩放是一种仿射变换，但不是刚体变换，通过缩放变换可以放大或者缩小对象。缩放不一定是均匀的，目前直接考虑相对于参考点（xf，yf，zf）的缩放变换，其步骤为：

* 1. 将平移到坐标原点处；
  2. 进行缩放变换；
  3. 将参考点（xf，yf，zf）移回原来位置

缩放变换过程如下：

glm库中scale()可以在当前变换矩阵上继续施加缩放变换，即右乘缩放变换矩阵，得到经过缩放变换后的矩阵，用法如下：



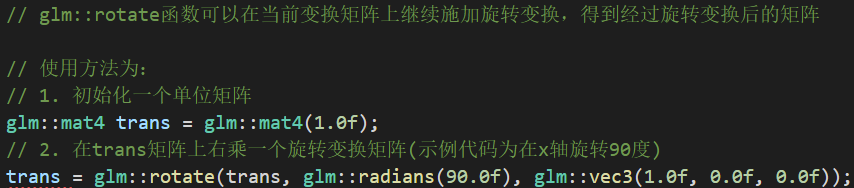
注意：示例代码中的当前变换矩阵为重新初始化的单位矩阵，在实际使用时当前变换矩阵不一定为单位矩阵。

1. 绕坐标轴的旋转变换

三维空间的旋转相对要复杂些，旋转和平移都是刚体变换。旋转和平移的任何组合都不能改变对象的形状和体积，它们只能改变对象的位置和方向。考虑右手坐标系下相对坐标原点绕坐标轴旋转变换，旋转变换过程如下：

* 1. 绕x轴旋转：
  2. 绕y轴旋转：
  3. 绕z轴旋转：

glm库中rotate()可以在当前变换矩阵上继续施加旋转变换，即右乘旋转变换矩阵，得到经过旋转变换后的矩阵，用法如下：

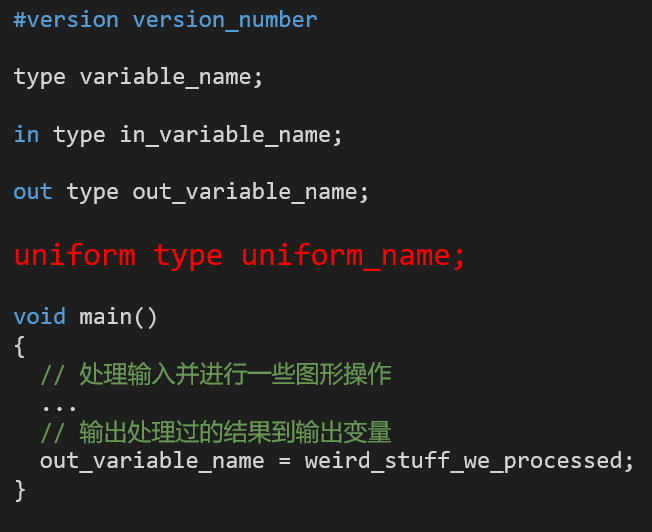


注意：示例代码中的当前变换矩阵为重新初始化的单位矩阵，在实际使用时当前变换矩阵不一定为单位矩阵。

1. Uniform关键字

本次实验将绘制一个立方体，并对其进行旋转平移等变换。这意味着要进行绘制的顶点坐标数据会不断变化。之前我们接触的着色器代码里面都是传入一个固定不变的顶点数据，如果发生数据变化就重新将数据传递给管道，这并不是理想的做法。

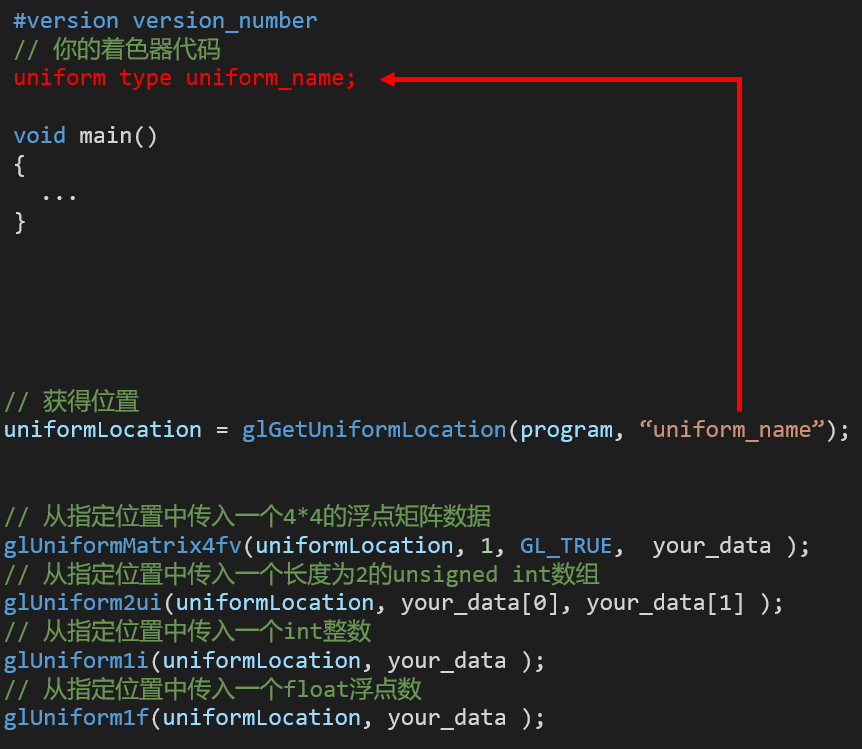
由于这种顶点的变化是由变换矩阵导致的，立方体初始的顶点数据其实并不需要变化，我们只需要改变变换矩阵的数值，然后在着色器内计算变化的顶点坐标就行。在GLSL编程中，带有uniform关键字的变量经常用来传递这类变化的变量。



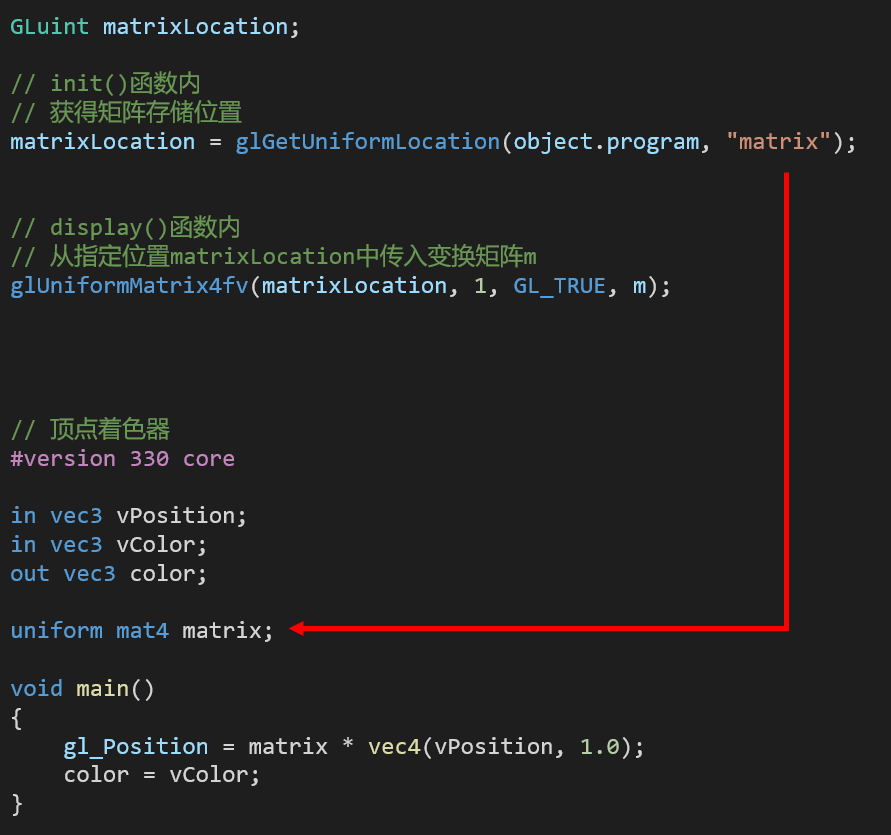
uniform是一种从CPU中的应用向GPU中的着色器发送数据的方式，但uniform和顶点属性有些不同。首先，uniform是全局的(Global)。全局意味着uniform变量必须在每个着色器程序对象中都是独一无二的，而且它可以被着色器程序的任意着色器在任意阶段访问。另外，无论你把uniform值设置成什么，uniform会一直保存它们的数据，直到它们被重置或更新。

不同于我们之前学习的关键字 in、out，它们这些关键字下的变量，都是先从顶点着色器输入，随后传递给片元着色器。uniform的变量是全局的，因此你可以在顶点着色器或者片元着色器内定义一个带uniform的变量，然后在我们的cpp代码内直接传递数据过去。

向着色器中传递uniform的方法和我们之前给vPosition那些传递的方法类似，也是要先获取这个变量在着色器的位置，使用函数为： glGetUniformLocation，然后再向该位置的变量传递数据，函数名字为：glUniform??。函数名字后面的问号内容与你要传递的数据类型有关。比如你传递的是3个浮点数，那函数名字为：glUniform3f。



平移旋转变化的是由变换矩阵实现的，我们可以把该矩阵作为uniform变量传递给着色器，在着色器内进行矩阵运算，而且矩阵运算正是GPU的特长。



1. 实验内容

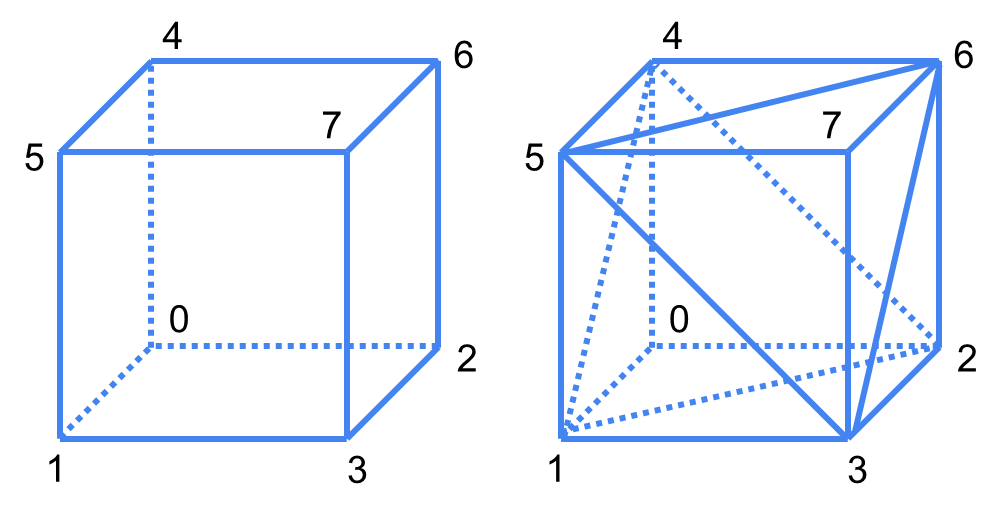
本次实验将绘制一个立方体，并对其进行旋转平移等变换。

1. TriMesh类的完善

为了之后更方便的绘制各种形状，我们将之前重复使用的函数进行了封装，将处理顶点数据的代码封装成了TriMesh类，里面包含保存顶点坐标、顶点颜色、面片顶点下标的变量。



我们在代码中预先定义好了立方体顶点坐标的数组cube\_vertices和颜色数组colors。根据下图的立方体示意图，在TriMesh.cpp中完善storeFacesPoints和generateCube函数。

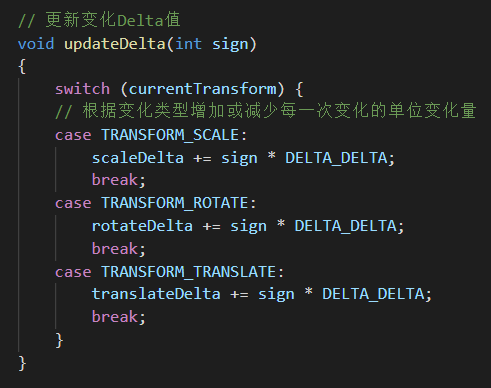


而且我们将绘制每个几何物体需要的OpenGL对象（比如vao）等都放在了一个结构体openGLObject内，方便数据管理。当TriMesh的函数都完善好后，将bindObjectAndData内标记了@TODO的注释代码取消掉。

我们在main.cpp中定义了一个立方体变量cube\_object，以及对应的TriMesh对象cube。在display中进行绘制。

1. 立方体的变换

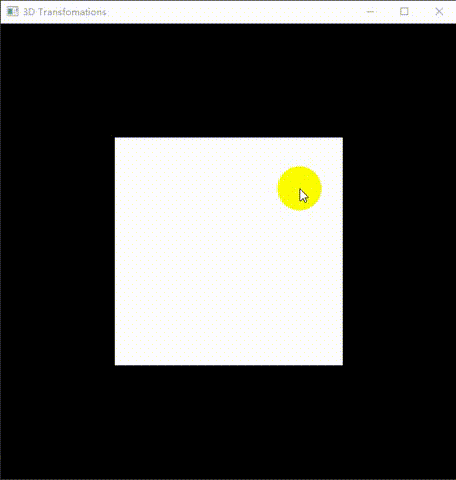
main.cpp开头代码中我们定义了一些旋转位移相关的参数，并绑定了键盘事件来控制这些参数，调用updateDelta函数更新参数数值。



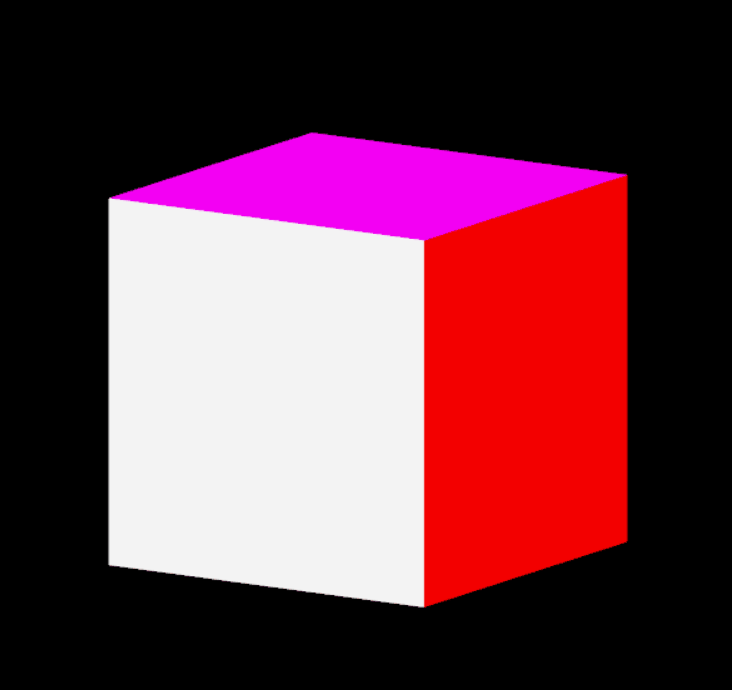
在display中使用这些参数计算变换矩阵，并且实时将最新的变换矩阵传递给渲染管道，让立方体可以进行旋转平移等变化。



1. 课堂练习
2. 下载实验2.3的代码，理解代码，并完成缺失的部分
3. 具体修改内容如下：
   1. Task-1：在TriMesh.cpp中完善storeFacesPoints函数
   2. Task-2：在TriMesh.cpp中完善generateCube函数
   3. Task-3：打开main.cpp的bindObjectAndData()里的注释
   4. Task-4：在main.cpp中完善display函数
4. 最后得到下方展示的效果，可以通过键盘的操作进行切换变化类型，实现平移、旋转、缩放变化。



1. 有兴趣就试一试
   1. Uniform应用：在片元着色器中加入一个uniform的浮点变量，用来控制颜色的明暗，传递一个浮点数的函数为glUniform1f(GLint location, GLfloat v0)，其中可能用到的函数还有glfwGetTime()，作用是获取当前时间，返回是一个浮点数。



* 1. 变换矩阵与鼠标交互（要注意窗口的坐标系与openGL标准空间坐标系之间的转换）：绘制矩阵，然后参考实验2.1绑定鼠标事件，实现矩形拖拽。在鼠标事件中使用glfwGetCursorPos(window, &x, &y)可以获得鼠标在窗口内的位置坐标，并存储到x，y中，其中x与y都是相对于窗口左上角的相对坐标（单位是像素）。

徽标

描述已自动生成