浙江大学

本科实验报告

课程名称：计算机图形学

姓　　名：王晨宇

学　　院：计算机科学与技术学院

系：

专　　业：计算机科学与技术

学　　号：3180104919

指导教师：唐敏

目录

浙江大学实验报告

课程名称：	计算机图形学	实验类型：	综合		
实验项目名称：	GLFW 程序设计				
学生姓名：	王晨宇	专业：	计算机科学与技术	学号：	3180104919
同组学生姓名：	None	指导老师：	唐敏		
实验地点：	紫金港机房	实验时间：	2021-04-03		

一、 实验目的和要求

在 OpenGL 编程基础上，通过实现实验内容，掌握 OpenGL 的矩阵使用，并验证课程中矩阵变换的内容，实现三只兔子按时间的平移、旋转和缩放

二、 实验环境

- Visual Studio C++ 2019
- GLFW、GLAD、GLM

三、 实验内容和原理

一个片段从最初的坐标到最后显示的坐标需要经过若干坐标空间的变换。

物体空间 即模型导入的初始坐标

世界空间 通过一个 model 矩阵将物体空间的坐标映射到世界空间中

观察空间 通过 view 矩阵将世界空间坐标转化为用户视野前方的坐标而产生的结果。因此观察空间就是从摄像机的视角所观察到的空间。

投影空间 将观察空间变换到裁剪空间，通过 projection 矩阵实现

在这个实验中，观察矩阵和投影矩阵均已提供。对于物体的移动即 model 矩阵的维护，我们将平移、旋转和缩放分别维护一个变换矩阵，通过移动 * 旋转 * 缩放的组合来得到单个变换矩阵，传入着色器。

3.1 顶点着色器

```
const char* _vsCode =
"#version 330 core\n"
"layout(location = 0) in vec3 aPosition;\n"
"layout(location = 1) in vec3 aNormal;\n"
"out vec3 worldPosition;\n"
"out vec3 normal;\n"
"uniform mat4 model;\n"
"uniform mat4 view;\n"
"uniform mat4 projection;\n"
"void main() {\n"
" normal = mat3(transpose(inverse(model))) * aNormal;\n"
" worldPosition = vec3(model * vec4(aPosition, 1.0f));\n"
" gl_Position = projection * view * vec4(worldPosition, 1.0f);\n"
"}\n";
```

其中 aPosition 是模型的坐标位置，aNormal 是模型每个点上的法向量（?）——涉及到光照的话就需要法向量。在 draw 方法中通过 glUniformMatrix4fv 来传入对应 uniform 变量的值。out 类型的值会被修改，worldPosition 即世界空间的坐标，乘上 projection 和 view 得到最后的坐标 gl_Position

关于光照和颜色还在学习中，对代码中的片段着色器能理解但暂时不做详细解释。

四、 代码实现

4.1 函数 handleInput 的补充内容

```
void Transformation::handleInput() {
  // update bunnies position / rotation / scale here
  const glm::vec3 velocity = { 1.0f, 0.0f, 0.0f };
  const float angulerVelocity = 1.0f;
  const float scaleRate = 0.005f;

  // 左边兔子的位移量
  _positions[0][1] += _dir[0] * 0.005f * _deltaTime * 100;
  // 防止超过边界
  if (_positions[0][1] > 5.0f) {
    _dir[0] = -1.0f;
  }
  else if (_positions[0][1] < -5.0f) {
    _dir[0] = 1.0f;
  }
  // 中间兔子的旋转角度
  _rotateAngles[1] += 0.01f * _deltaTime * 100;
  // 最右边兔子的缩放比例
  for (int i = 0; i < 3; i++) {
    _scales[2][i] += _dir[2] * scaleRate * _deltaTime * 100;
  }
  // 控制在一个范围内
```

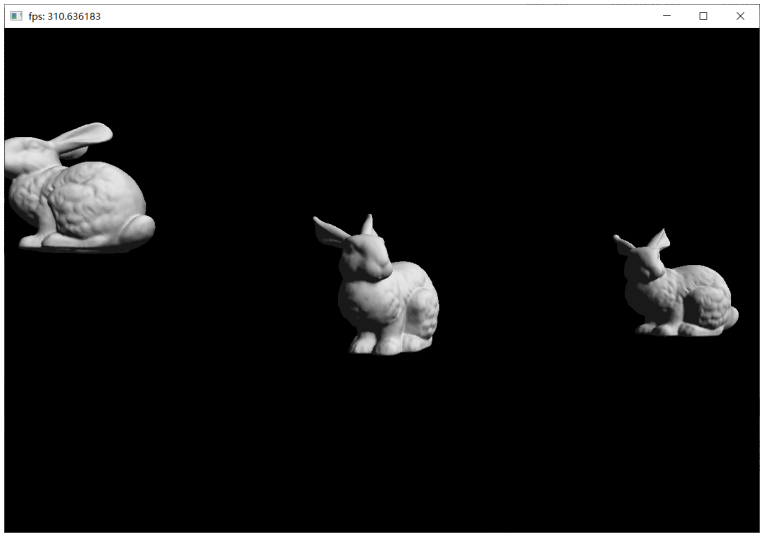
```
23    if (_scales[2][0] < 0.5f) _dir[2] = 1.0f;
24    else if (_scales[2][0] > 1.5f) _dir[2] = -1.0f;
25  }
```

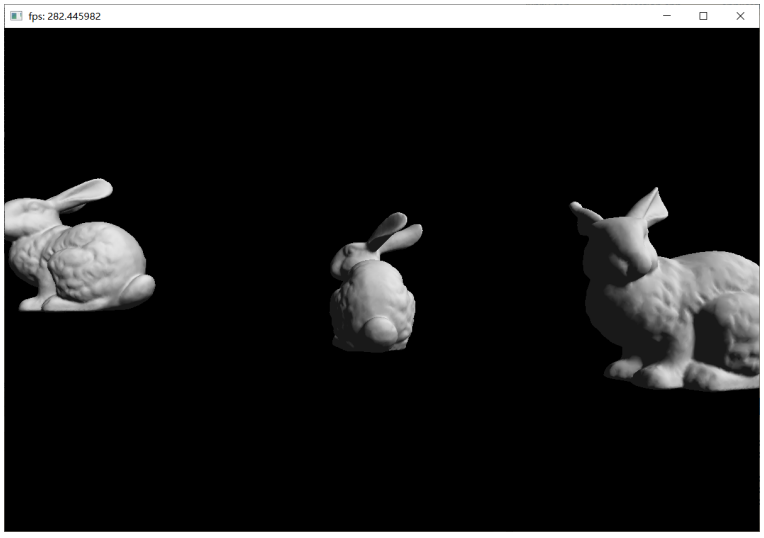
4.2 绘图时的坐标计算

```
1   void Bunny::draw(const glm::mat4& projection, const glm::mat4& view) {
2     // model matrix transform the homogenous coodinates
3     // from model space (raw vertex data form model) to world space, depending on
          following parametes:
4
5     // @translation
6     glm::mat4 translation = glm::mat4(1.0f);
7     translation = glm::translate(translation, _position);
8     //
9     // @rotation
10    glm::mat4 rotation = glm::mat4(1.0f);
11    rotation = _rotation;
12    // @scale
13    glm::mat4 scale = glm::mat4(1.0f);
14    scale = glm::scale(scale, _scale);
15
16    glm::mat4 model = translation * rotation * scale;
17
18    _shader.use();
19    _shader.setMat4("projection", projection);
20    _shader.setMat4("view", view);
21    _shader.setMat4("model", model);
22    glBindVertexArray(_vao);
23    glDrawElements(GL_TRIANGLES, _indices.size(), GL_UNSIGNED_INT, 0);
24    glBindVertexArray(0);
25  }
```

五、 实验结果

随便截了两张不同时刻的图，详细的结果可以在验收的时候看到





六、 思考题：_deltaTime 的作用

实际上在配置环境的第一节课助教学长已经提到了一些这个 _deltaTime 的作用。我们通过两帧之间的时间间隔来算出即时的帧率，即 $fps = \frac{1}{\_deltaTime}$。而我们对于每次坐标位置（即变换矩阵）的修改是在每一次渲染循环时候进行的，也就是如果没有 _deltaTime 的参与，位移变化量是在每一帧变换时修改。这样对于性能不同的设备，当我们想做迁移时可能会发现不同设备上的效果不同。

实际上，因为我的电脑运行时帧率非常不稳定，所以也会出现偶尔运动速度特别快，有时又特别慢的情况，所以也能感觉到实现时候可能存在问题。

所以解决办法就是让变换速率和绝对时间关联，通过调整系数来达到想要的效果。可以参考代码实现。

PS：助教在实验课上也讲到了，我之前的理解上应该没什么问题。

七、 讨论和心得

因为大部分框架内容都已经提供，所以完成这个实验本身并没有非常大的难度。

通过这次实验包括和学长的讨论，让我对 OpenGL 里顶点传递和坐标变换的理解更深刻了一些。通常我们会有许多点构成一个 model，它们共同构成一个整体。对于这个 model 上的变换对应的矩阵是相同的，所以维护一个同一个变换矩阵是非常高效的做法。所以上个实验中对于五角星的绘制，维护矩阵而不是每次重新绑定、填充顶点坐标会是更好的做法。

还有一个小问题是，不知道为什么在我的电脑上绘图的帧率非常不稳定，波动非常剧烈，在 200 到 1000 之间疯狂跳动，还不知道问题出在了哪里。