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:	OpenGL 纹理				
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:	2021-06-07		

一、 实验目的和要求

在 OpenGL 消隐和光照实验的基础上，通过实现实验内容，掌握 OpenGL 中纹理的使用，并验证课程中关于纹理的内容。

二、 实验环境

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

三、 实验内容和原理

3.1 纹理

纹理大概是在几何图形上贴上图像。
通过关联纹理坐标来表明采样器的采样范围，在图形的其它片段上进行片段插值。
关于纹理环绕和纹理过滤不再赘述
纹理的生成和加载与顶点十分类似:

1. 首先创建一个纹理对象
2. 绑定一个 GL_TEXTURE_2D 的对象
3. 设置纹理的环绕和过滤方式
4. 使用载入的图片生成一个纹理

忽略 data 的获取过程，生成纹理的过程大致如下

```
1
2
3
```

因为纹理坐标是一个 2 维向量，所以在顶点格式中需要调整步长参数和属性大小。

最后在绘制前只需要绑定相应的纹理即可。

3.2 纹理混合

我们可以利用纹理单元来在着色器中使用多于一个的纹理。通过把纹理单元赋值给采样器，我们可以一次绑定多个纹理，只要我们首先激活对应的纹理单元。

我们还要通过使用 glUniform1i 设置每个采样器的方式告诉 OpenGL 每个着色器采样器属于哪个纹理单元。

调用 GLSL 内建的 mix 函数需要接受两个值作为参数，并对它们根据第三个参数进行线性插值。

实现代码见下一小节。

3.3 程序棋盘纹理生成

要生成这种有一定规律的纹理，注意到对于球它的纹理坐标是把 x 方向坐标从 $(0, 2\pi)$ 和 y 方向的 $(0, \pi)$ 映射到 $(0, 1)$。我们就想把他们映射回 $(0, repeat)$ 的区间，用类似于格雷码的思想，每当取整后的 x 或 y 变化 1，则这个颜色需要发生变化。直接讨论奇偶性，就变成以下的真值表：那么最后每个点的颜色公式就是 $(x == y) \times color1 + (x! = y) \times color2$

x	y	color
0	0	color1
0	1	color2
1	0	color2
1	1	color1

(x, y 对 2 取模)

相应的着色器代码见下一节。

3.4 天空盒

天空盒是一种立方体贴图，它用 6 个 2D 纹理的纹理拼成了一个有纹理的立方体。对于中心在原点的立方体来说，指向立方体实际位置的向量就是对应的纹理坐标。

天空盒纹理创建部分和之前的纹理内容略有不同但大致类似，故不再赘述。

在天空盒的绘制过程中，有一些不一样的地方。首先我们需要改变剔除模式，一般情况下我们希望将三角形背向相机的面剔除，但是对于天空盒，由于相机是放在天空盒内部的，所以我们希望看到的是天空盒的内部，但是一般情况下，球体的模型表面被定义为正面（根据顶点定义的顺序确定），我们可以修改我们的模型或者修改 OpenGL 的状态来解决这个问题。这里我们使用后者，所以我们告诉 OpenGL 对正面进行剔除。其次我们需要改变深度测试函数，默认情况下，我们告诉 OpenGL 当新的片元的 Z 值小于之前存放的片元时，则新的片元覆盖之前的片元。对于天空盒，其片元的 Z 值始终等于远裁剪面的值，当我们将深度测试函数设置为 “less than” 时，这样的片元在深度测试的时候不能保存下来，所以这里我们将深度测试函数设置为 GL_LEQUAL(小于或者等于)。

剩下的就和普通的绘制一样啦。

四、 代码实现

4.1 纹理混合

首先告诉着色器，采样器属于哪个纹理单元：

```
1
2
3
```

然后在绘制之前要激活两个纹理单元

```
1
2
3
4
```

最后是片段着色器的修改，加入了实验 5 中实现了的 lambert 光照模型，并且调用 mix 函数将两个纹理混合。

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

4.2 程序棋盘纹理生成

片段着色器部分：

4.3　天空盒

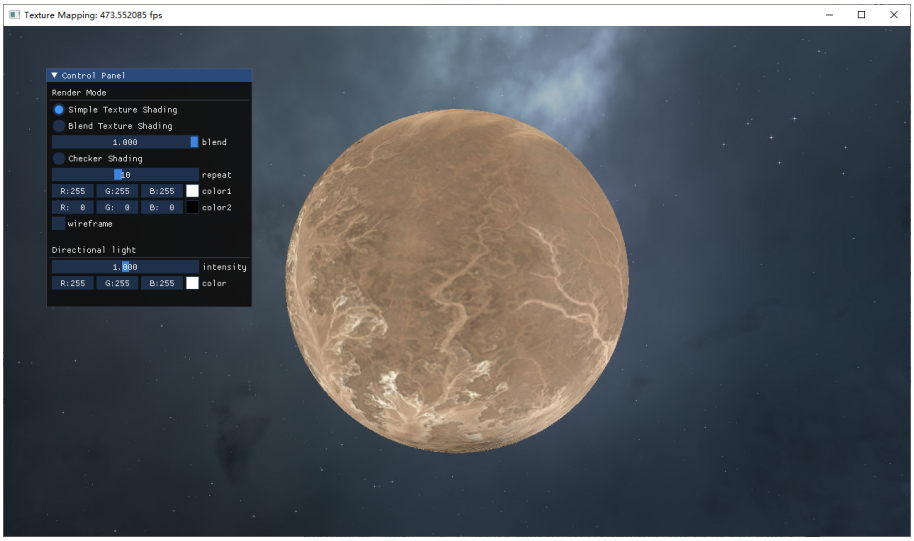
Draw 函数

```
1
2
3
4
5
6
7
8
9
10
11
12
```

忘记了别的还实现了什么就放这个吧。

五、　实验结果

5.1　天空盒效果



5.2 纹理混合及光照

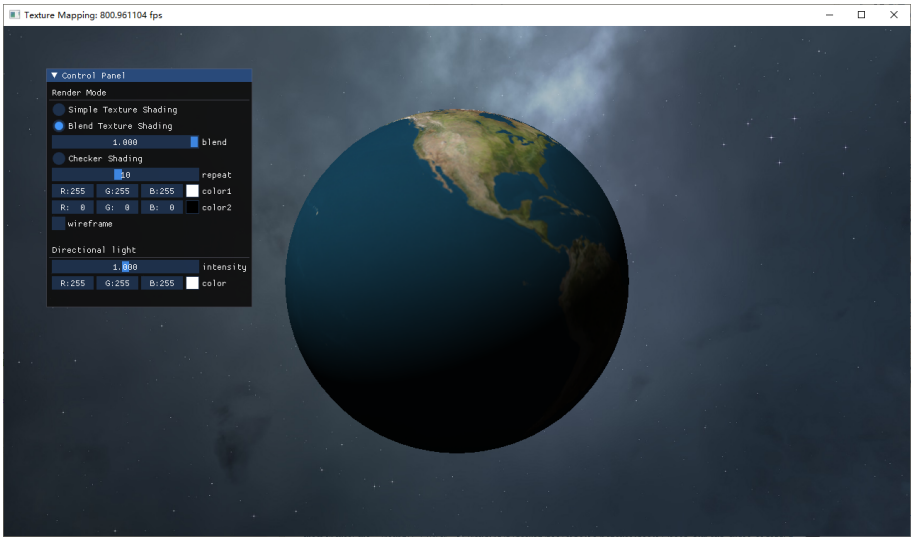


图 1: blend=1.0

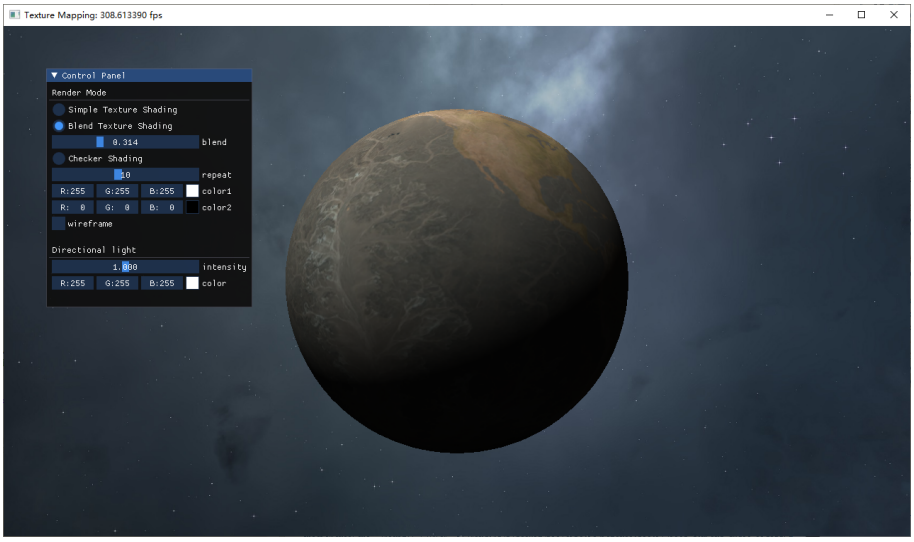


图 2: blend!=1.0

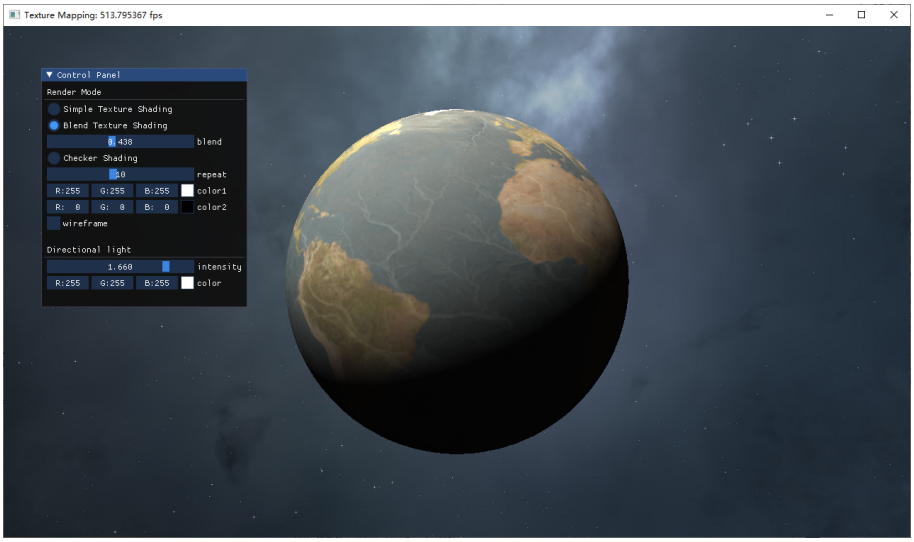


图 3: Increase light intensity

5.3 棋盘格纹理

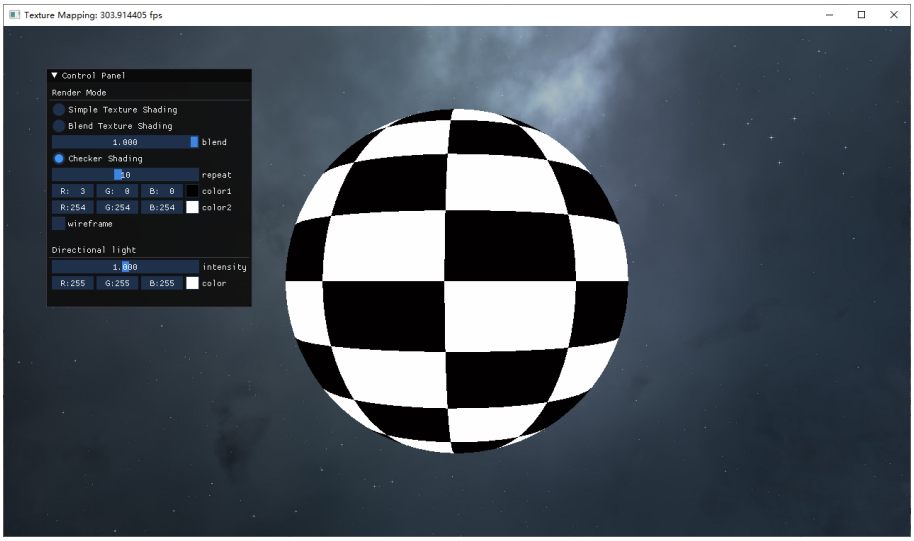


图 4: Origin

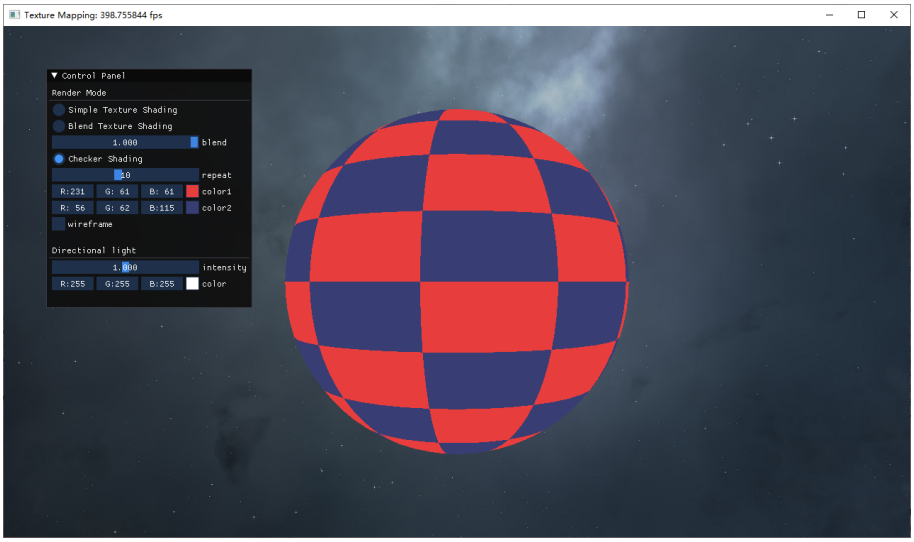


图 5: change color

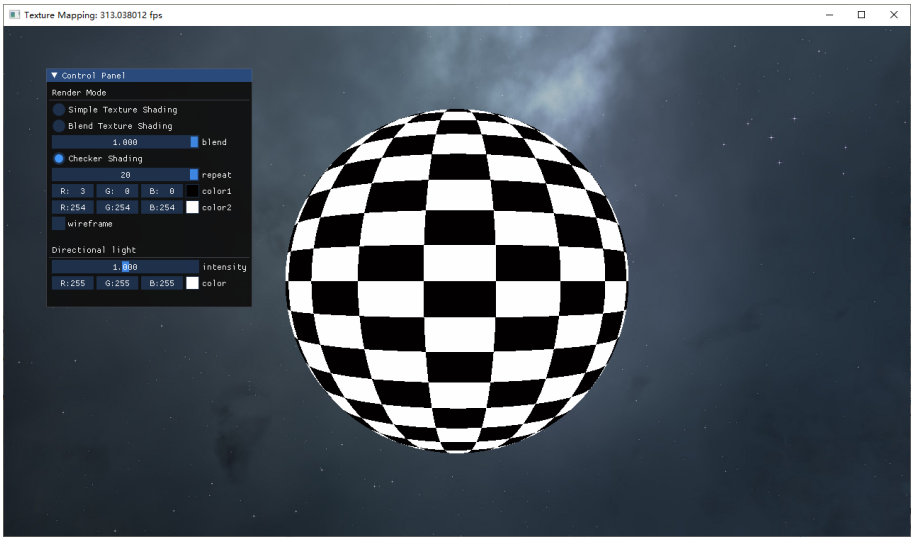


图 6: 密集的纹理

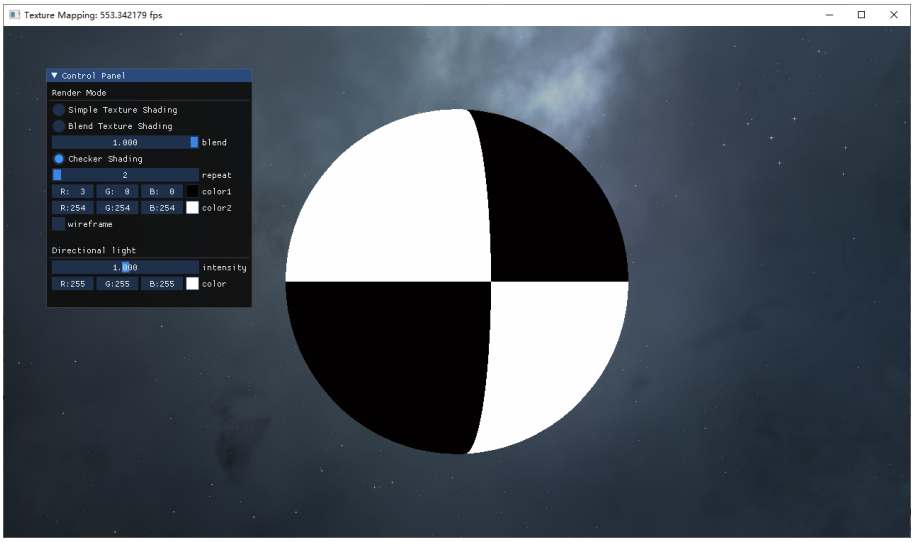


图 7: 宝马车标

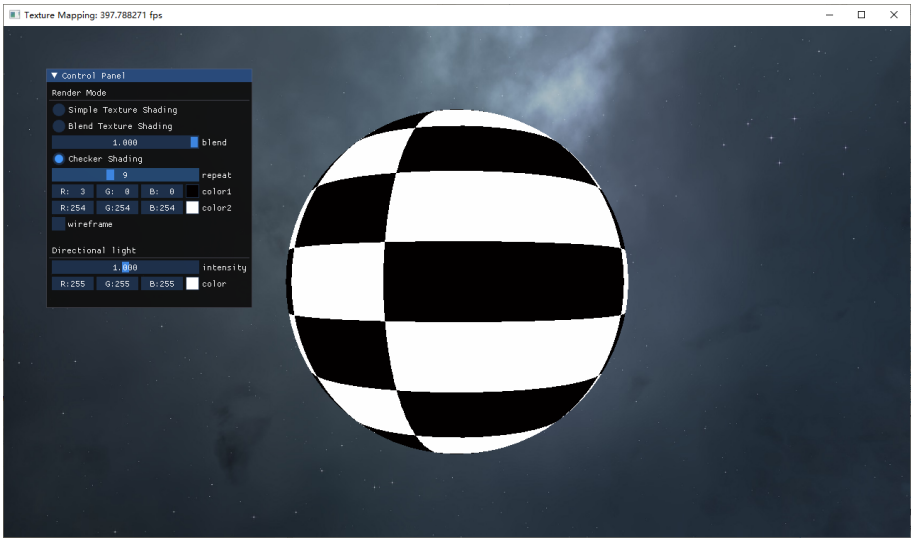


图 8: 奇数时候有点小问题

六、　讨论和心得

一开始没有激活天空盒的 shader，就出现了这样的画面

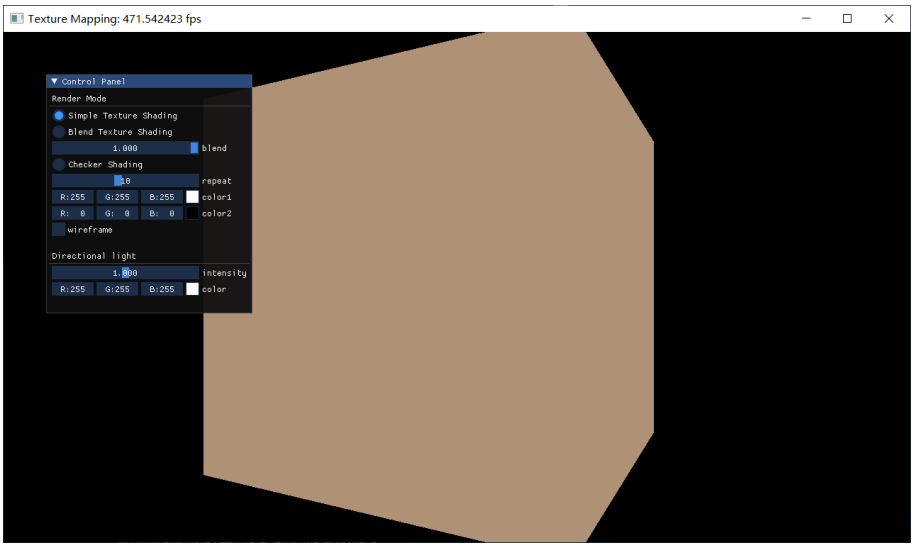


图 9: 旋转的立方体

这次的实验还是比较有趣，因为对 shader 的语法没有那么熟悉所以还是在实现 checker texture 的时候还是思考了一会儿。

关于纹理还是有很多高级技术值得学习，在追求更好的效果的道路上没有尽头。