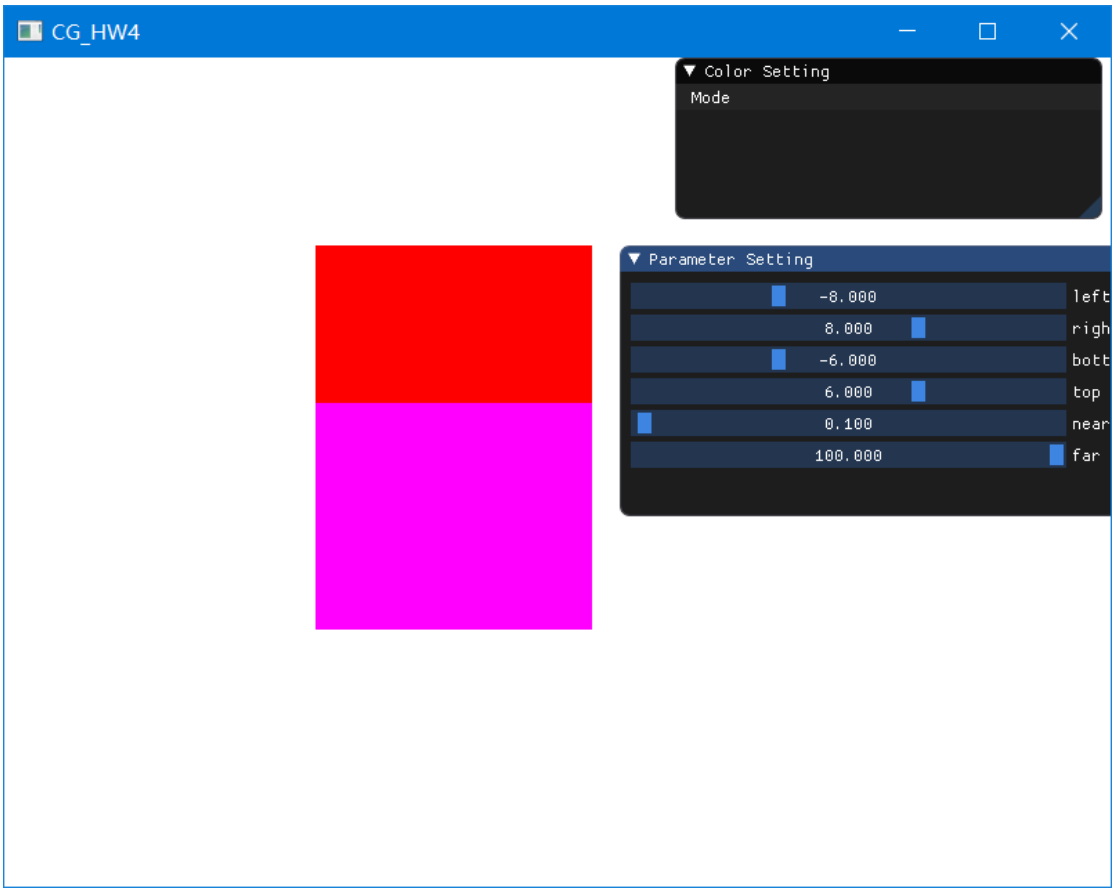
HW5_Report

1. 投影(Projection):
   - 把上次作业绘制的 cube 放置在(-1.5, 0.5, -1.5)位置，要求 6 个面颜色不一致
     修改 model 的平移矩阵的参数即可：

```
model = glm::translate(model, glm::vec3(x, y, z));
```

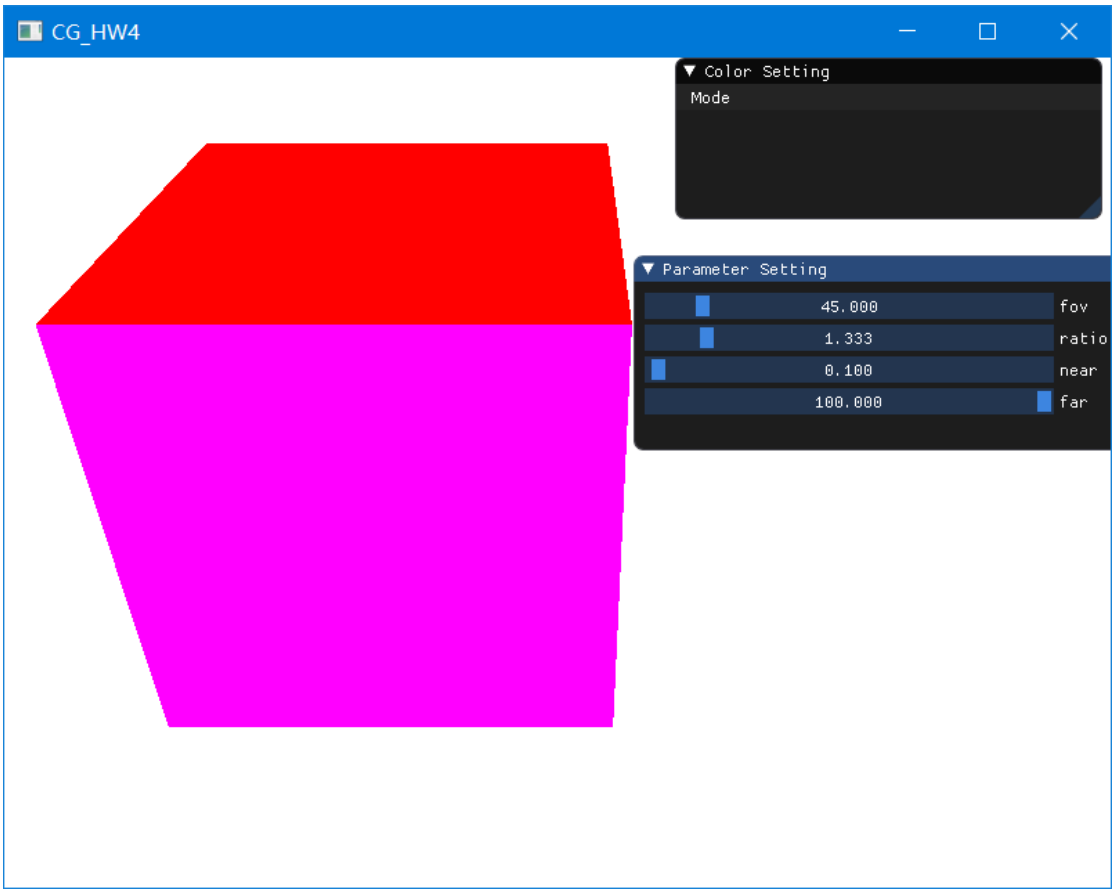
   - 正交投影(orthographic projection)：实现正交投影，使用多组(left, right, bottom, top, near, far)参数，比较结果差异




```
ImGui::SliderFloat("left", &left, -16, right);
ImGui::SliderFloat("right", &right, left, 16);
ImGui::SliderFloat("bottom", &bottom, -12, top);
ImGui::SliderFloat("top", &top, bottom, 12);
ImGui::SliderFloat("near", &nearp, 0, 10);
ImGui::SliderFloat("far", &farp, 0, 24);
ImGui::End();
glm::mat4 projection;
//projection = glm::ortho(-8.0f, 8.0f, -6.0f, 6.0f, 0.1f, 100.0f);
projection = glm::ortho(left, right, bottom, top, nearp, farp);
glUniformMatrix4fv(glGetUniformLocation(shaderProgram, std::string("projection").c_str()), 1, GL_FALSE, &projection[0][0]);
```

其中平截头体的对称位置的坐标如果大小关系对调，将会出现镜像效果。前四个参数将决定投影到屏幕（窗口）的内容范围，比如 left 往右拉，投影的内容则从左侧往右缩小，于是重新在没缩小的窗口上显示则重新拉长了。而 near 过大，则会穿过 cube 看到内部，类似的 far 过小，会显示不全。投影的内容就是 6 个参数定义的长方体（平截头体）内的内容。具体对比见 gif。

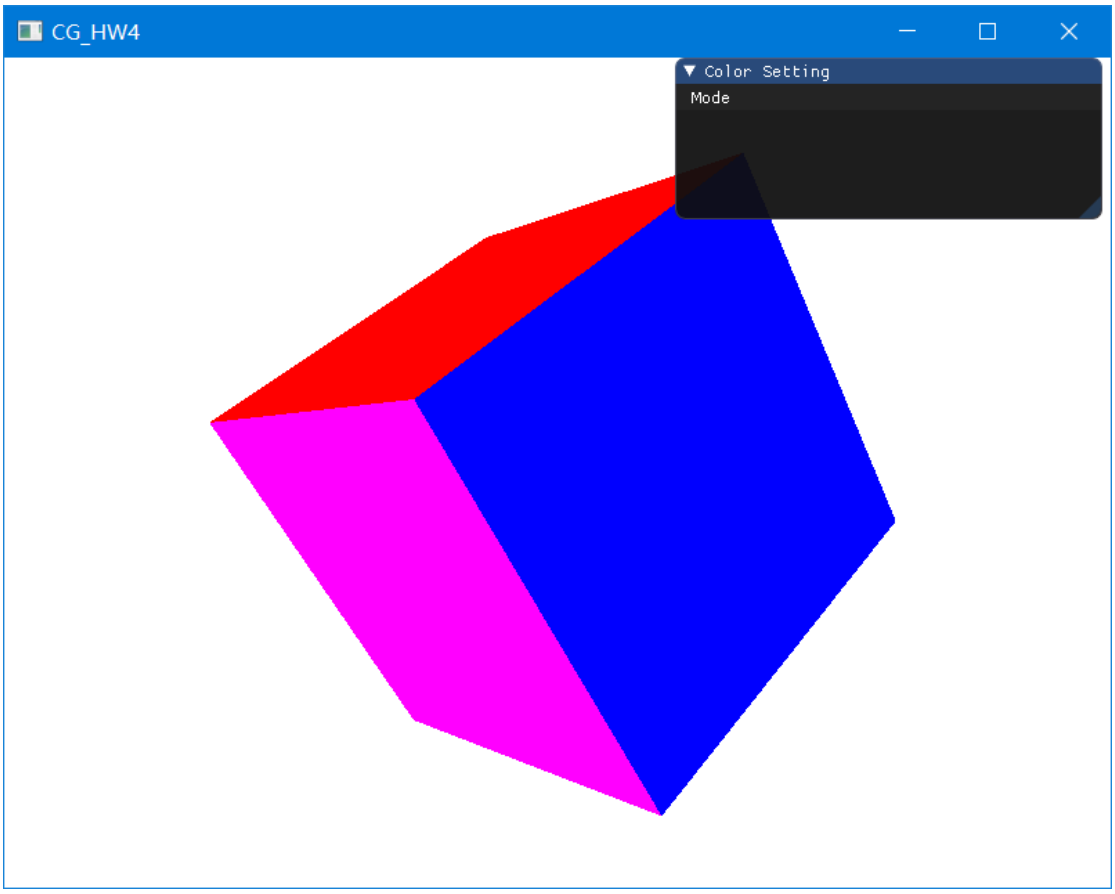
- 透视投影(perspective projection)：实现透视投影，使用多组参数，比较结果差异




```
ImGui::Begin("Parameter Setting");
ImGui::SliderFloat("fov", &fov, 0, 360);
ImGui::SliderFloat("ratio", &ratio, 0, 10);
ImGui::SliderFloat("near", &nearp, 0, 10);
ImGui::SliderFloat("far", &farp, 0, 24);
ImGui::End();
glm::mat4 projection;
projection = glm::perspective(glm::radians(fov), ratio, nearp, farp);
glUniformMatrix4fv(glGetUniformLocation(shaderProgram, std::string("projection").c_str()), 1, GL_FALSE, &projection[0][0]);
```

透视投影有 4 个参数，第一个是视野，指从观察点出发的观察空间上下夹角。第二个是宽高比。这两个参数已经可以确认可视范围（得到一个从点出发的无限长的三角锥）。第三、四个参数与正交投影同理。Ratio（宽高比）增加的话，cube 左右会压扁，因为可视内容的宽相对于高增加了，横向内容增加，但是窗口大小不变，因此内容会横向压缩。Fov 增加，正方体会缩小。这是因为视野夹角增大，直接导致宽高同时增加（宽高比不变），于是同理，内容增加，窗口显示的内容则压缩。后

2. 视角变换(View Changing):
   - 把 cube 放置在(0, 0, 0)处，做透视投影，使摄像机围绕 cube 旋转，并且时刻看着 cube 中心




```
float radius = 10.0f;
float camPosX = sin(glfwGetTime()) * radius;
float camPosZ = cos(glfwGetTime()) * radius;
glm::mat4 view;
view = glm::lookAt(glm::vec3(camPosX, 0.0, camPosZ), glm::vec3(0.0, 0.0, 0.0), glm::vec3(0.0, 1.0, 0.0));
unsigned int viewLoc = glGetUniformLocation(shaderProgram, "view");
glUniformMatrix4fv(viewLoc, 1, GL_FALSE, &view[0][0]);
```

确定我们的摄影机需要三个信息，一个是摄影机位置，一个摄影机指向的方向，一个是摄影机镜头旋转角度，分别可以用摄影机的 3D 坐标、摄影机所指方向的 3 维向量、摄影机向上的 3 维向量来确认三个信息。于是我们需要让摄影机位置的 x 与 z 坐标的关系满足 x^2+z^2=radius，则可让其在 XOZ 平面上绕原点作半径为 radius 的圆运动。镜头指向原点，上向量指向 y 轴正方向，即为我们想要的效果。

3. 在 GUI 里添加菜单栏，可以选择各种功能。

4. 在现实生活中，我们一般将摄像机摆放的空间 View matrix 和被拍摄的物体摆设的空间 Model matrix 分开，但是在 OpenGL 中却将两个合二为一设为 ModelView matrix，通过上面的作业启发，你认为是为什么呢？在报告中写入。
   因为在 OpenGL 实际上并没有摄像机这个概念，这个是开发者自己利用矩阵变换 Model 的各种参数模拟出在控制摄影机的效果。我们可以将 Model 往与期望的摄像机变换方向相反操作，模拟出摄影机在移动的效果，实际上只是 Model 改变了参数以至于在屏幕显示的内容发生了改变。