作业6

说明

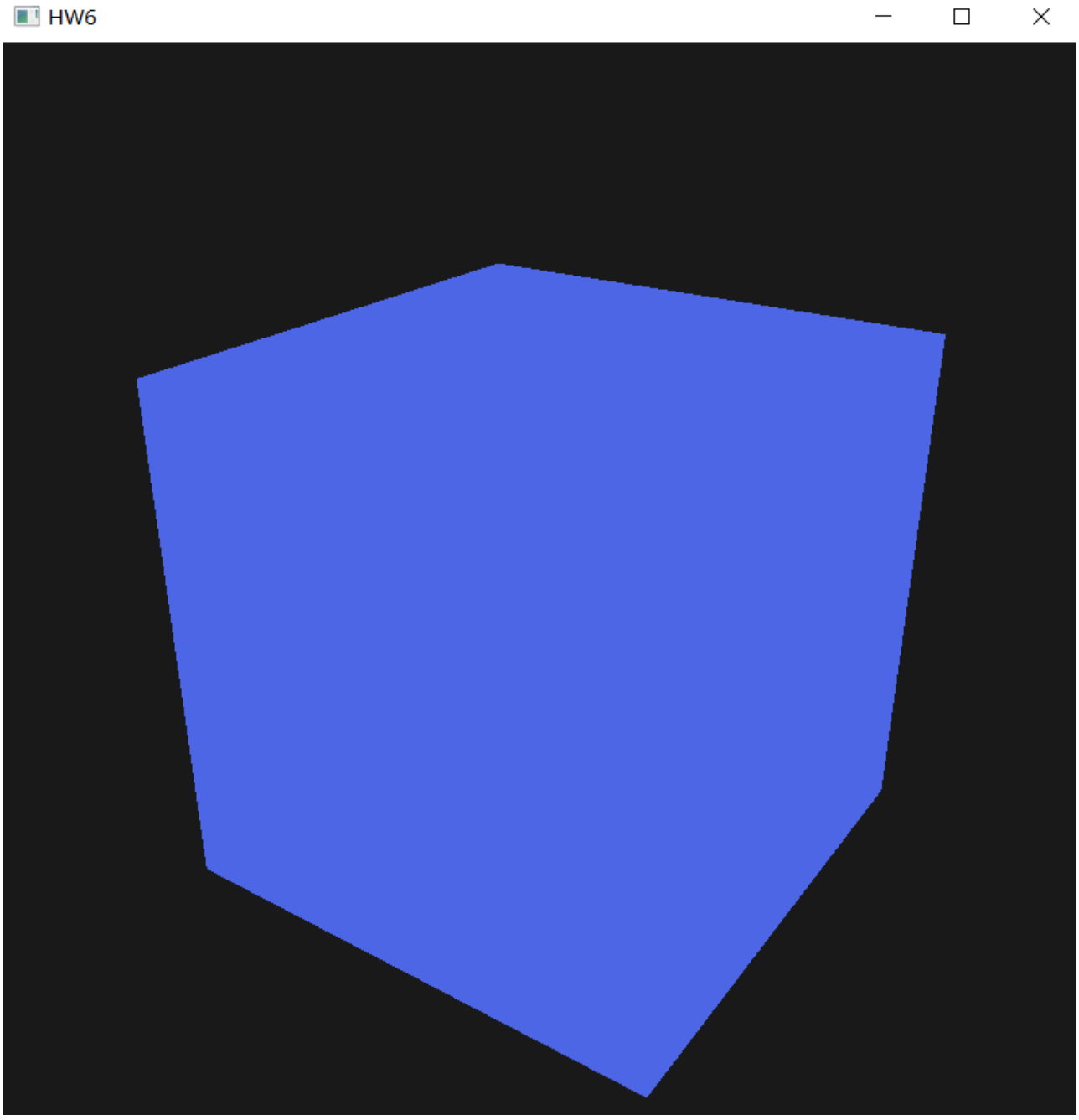
- 在本题中，将鼠标改变摄像机的方式更改为了在程序中按下右键进行移动，而不是之前直接将鼠标锁定在程序中就可以直接进行移动。
- 本题中，basic第一问物体所用着色器的为1_开头；第二问和bonus所用着色器为2_开头。

Basic

1. 实现Phong光照模型：

场景中绘制一个**cube**

- 绘制的cube如下图所示。



自己写**shader**实现两种**shading: Phong Shading** 和 **Gouraud Shading**，并解释两种**shading**的实现原理

- Phong Shading
  - 原理：

    冯氏光照模型的主要结构由3个分量组成：环境(Ambient)、漫反射(Diffuse)和镜面(Specular)光照。

    - 环境光照(Ambient Lighting)：即使在黑暗的情况下，世界上通常也仍然有一些光亮（月亮、远处的光），所以物体几乎永远不会是完全黑暗的。为了模拟这个，我们会使用一个环境光照常量，它永远会给物体一些颜色。
    - 漫反射光照(Diffuse Lighting)：模拟光源对物体的方向性影响(Directional Impact)。它是冯氏光照模型中视觉上最显著的分量。物体的某一部分越是正对着光源，它就会越亮。
    - 镜面光照(Specular Lighting)：模拟有光泽物体上面出现的亮点。镜面光照的颜色相比于物体的颜色会更倾向于光的颜色。

    而最终片段颜色：环境颜色+漫反射颜色+镜面反射颜色

  - 顶点着色器

```
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aNormal;

out vec3 FragPos;
out vec3 Normal;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;

void main()
{
    gl_Position = projection * view * model * vec4(aPos, 1.0);
    FragPos = vec3(model * vec4(aPos, 1.0));
    Normal = mat3(transpose(inverse(model))) * aNormal;
}
```

  - 片段着色器

```
#version 330 core
out vec4 FragColor;

in vec3 Normal;
in vec3 FragPos;

uniform vec3 lightPos;
uniform vec3 viewPos;
uniform vec3 lightColor;
```

```
uniform vec3 objectColor;

void main()
{
    // ambient
    float ambientStrength = 0.1;
    vec3 ambient = ambientStrength * lightColor;

    // diffuse
    vec3 norm = normalize(Normal);
    vec3 lightDir = normalize(lightPos - FragPos);
    float diff = max(dot(norm, lightDir), 0.0);
    vec3 diffuse = diff * lightColor;

    // specular
    float specularStrength = 0.5;
    vec3 viewDir = normalize(viewPos - FragPos);
    vec3 reflectDir = reflect(-lightDir, norm);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);
    vec3 specular = specularStrength * spec * lightColor;

    vec3 result = (ambient + diffuse + specular) * objectColor;
    FragColor = vec4(result, 1.0);
}
```

- Gouraud Shading
  - 原理：

    在顶点着色器中实现的冯氏光照模型叫做Gouraud着色(Gouraud Shading)，而不是冯氏着色(Phong Shading)。在顶点着色器中做光照的优势是，相比片段来说，顶点要少得多，因此会更高效，所以开销大的光照计算频率会更低。然而，顶点着色器中的最终颜色值是仅仅只是那个顶点的颜色值，片段的颜色值是由插值光照颜色所得来的。结果就是这种光照看起来不会非常真实，除非使用了大量顶点。

  - 顶点着色器

```
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aNormal;

out vec3 LightingColor;

uniform vec3 lightPos;
uniform vec3 viewPos;
uniform vec3 lightColor;

uniform mat4 model;
uniform mat4 view;
uniform mat4 projection;
```

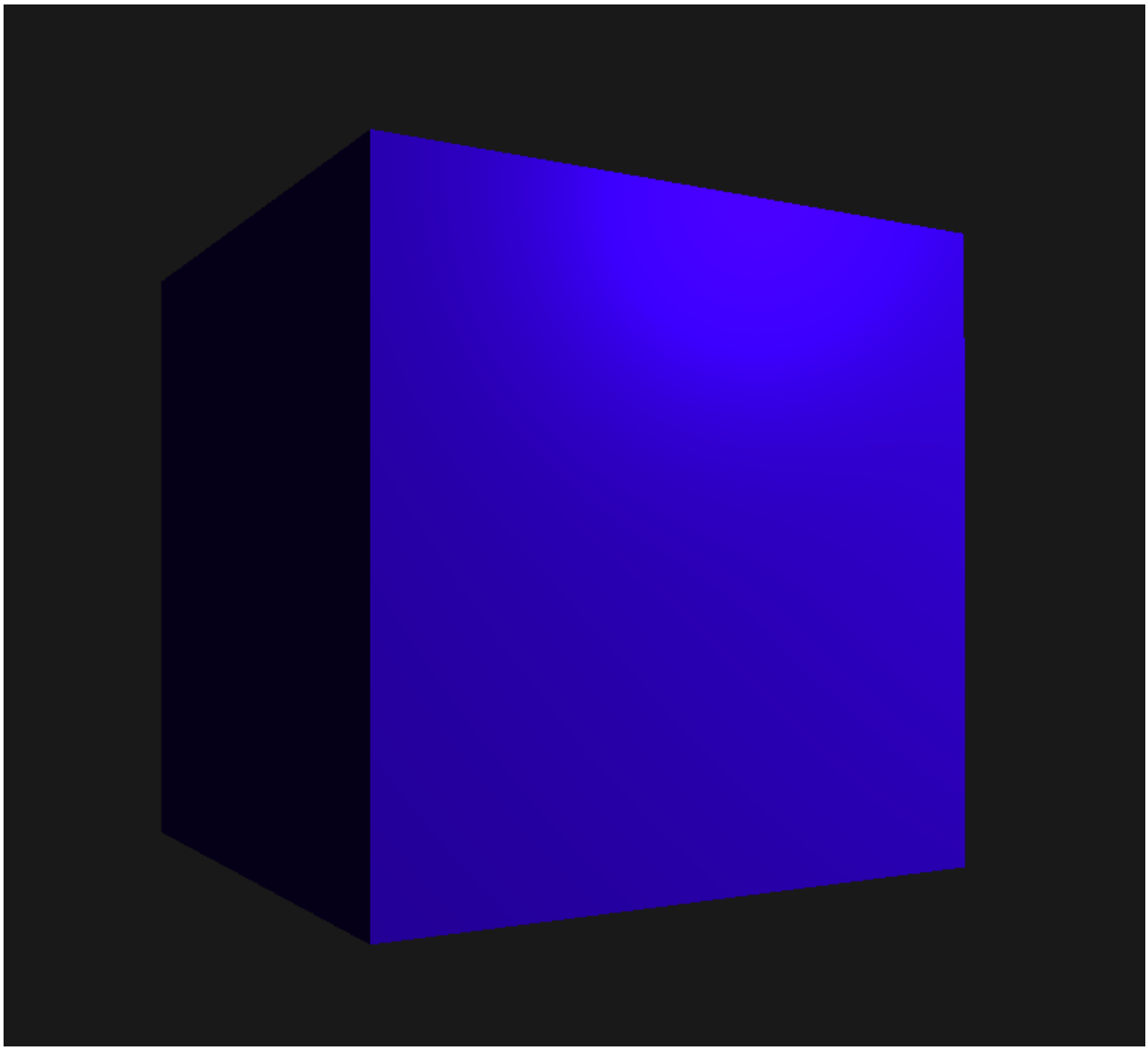
```
void main()
{
    gl_Position = projection * view * model * vec4(aPos, 1.0);

    // gouraud shading
    vec3 Position = vec3(model * vec4(aPos, 1.0));
    vec3 Normal = mat3(transpose(inverse(model))) * aNormal;

    // ambient
    float ambientStrength = 0.1;
    vec3 ambient = ambientStrength * lightColor;

    // diffuse
    vec3 norm = normalize(Normal);
    vec3 lightDir = normalize(lightPos - Position);
    float diff = max(dot(norm, lightDir), 0.0);
    vec3 diffuse = diff * lightColor;

    // specular
    float specularStrength = 1.0;
    vec3 viewDir = normalize(viewPos - Position);
    vec3 reflectDir = reflect(-lightDir, norm);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), 32);
    vec3 specular = specularStrength * spec * lightColor;

    LightingColor = ambient + diffuse + specular;
}
```

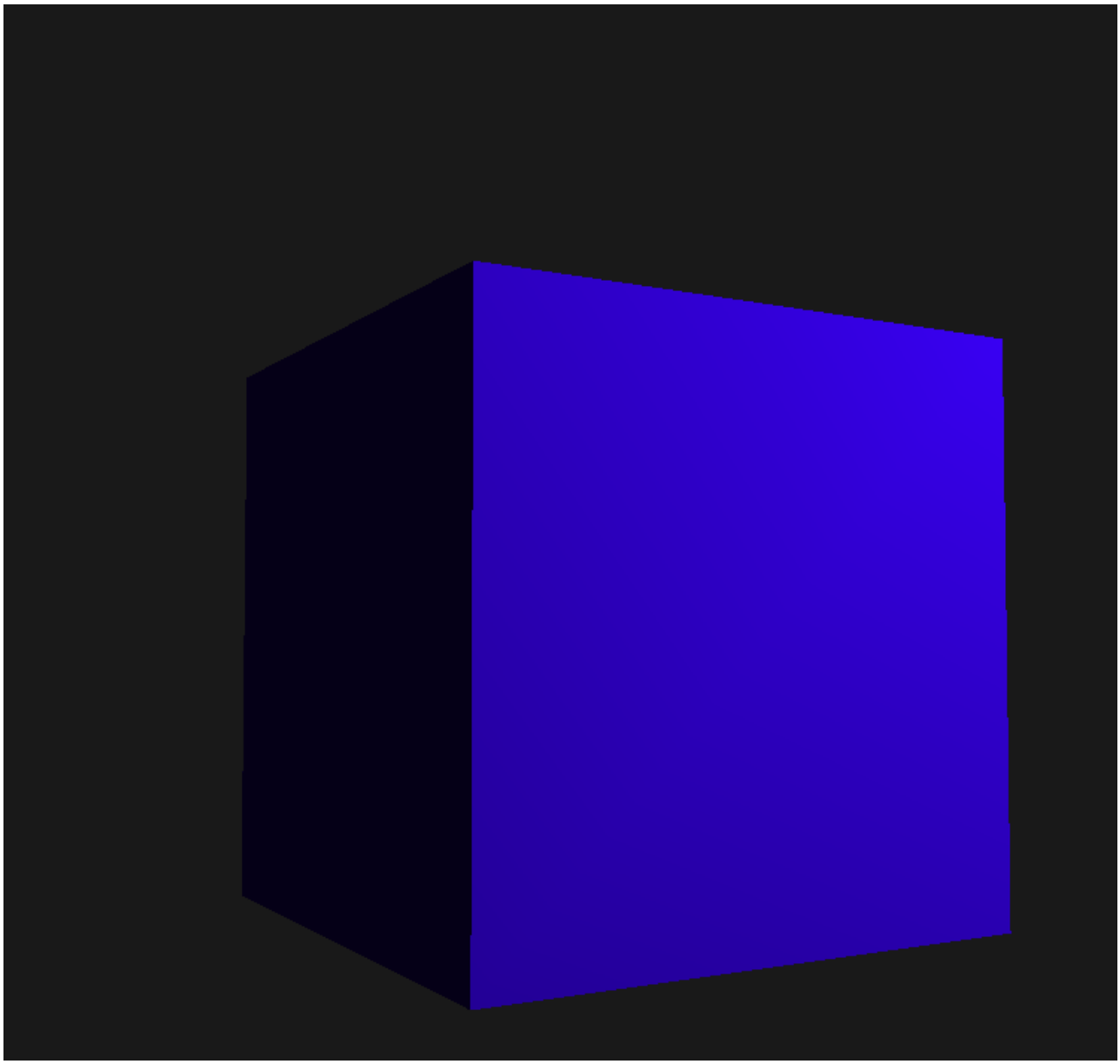
  - 片段着色器

```
#version 330 core
out vec4 FragColor;

in vec3 LightingColor;

uniform vec3 objectColor;

void main()
{
    FragColor = vec4(LightingColor * objectColor, 1.0);
}
```

- 样例：

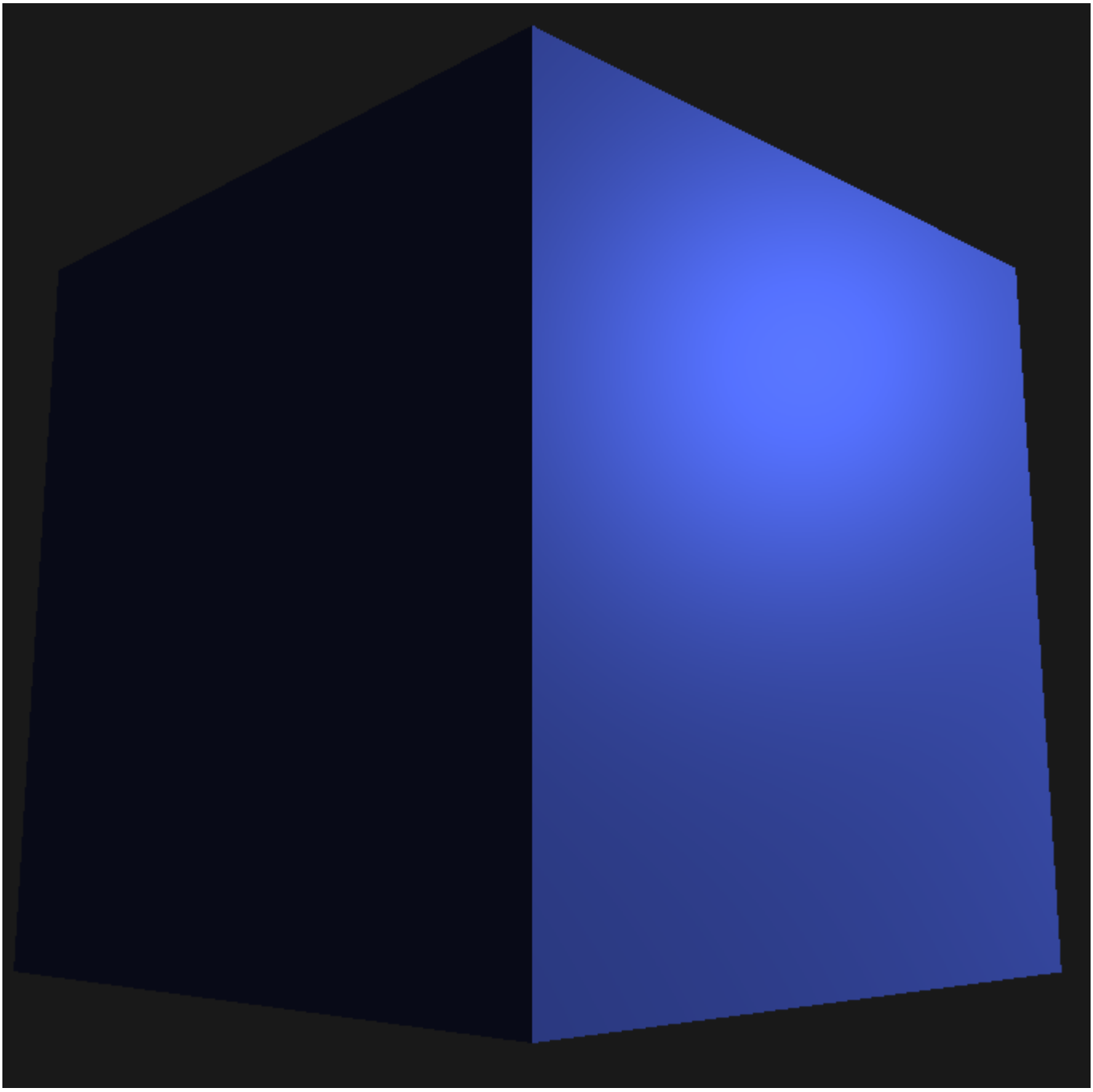
采用冯氏光照的如下方第一张图所示，采用Gouraud着色并在几乎同一位置观察得到的图如下放第二张图所示所示。可以看出，冯氏光照在接近正方体的上部方面的镜面反射要明显的的多，显得更加真实。





合理设置视点、光照位置、光照颜色等参数，使光照效果明显显示

将光照位置调整至(2.0f, 1.0f, 2.0f)，光照颜色调整为(1.0f, 1.0f, 1.0f)(初始为(0.7f, 0.0f, 1.0f))



光源的和立方体的设置参考learnopengl网站上的说明。为了显示真正的灯，这里将表示光源的立方体绘制在与光源相同的位置并且将使用我们为它新建的片段着色器来绘制它，让它一直处于白色的状态，不受场景中的光照影响。

2. 使用GUI，使参数可调节，效果实时更改：

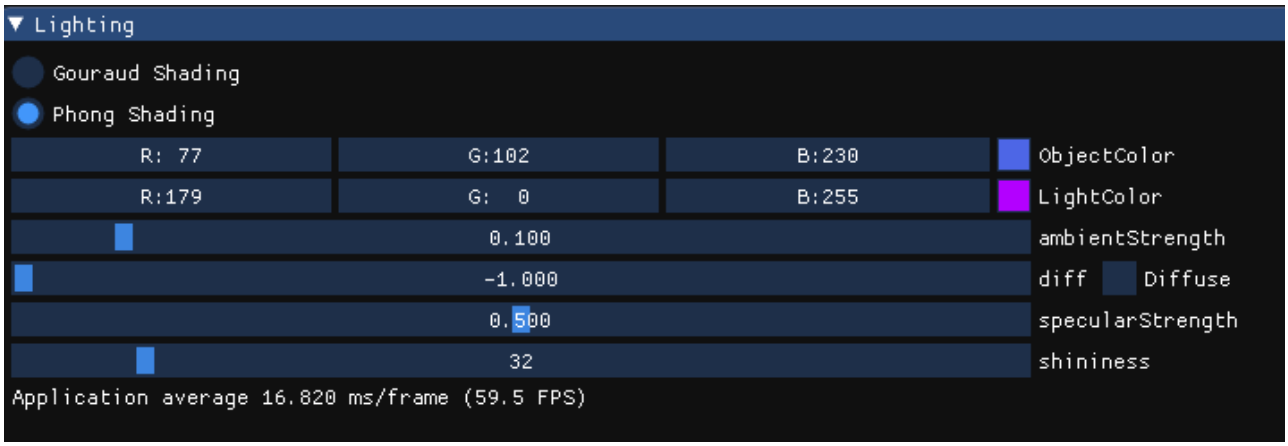
- GUI里可以切换两种shading
- 使用如进度条这样的控件，使ambient因子、diffuse因子、specular因子、反光度等参数可调节，光照效果实时更改

IMGUI设计

说明

按照题目要求分别设计了如下功能,按照下方图片顺序进行说明

- 选择光照模型
- 调节物体及灯光的颜色
- 对ambient因子进行滑动调节，范围为0~1
- 选择是否对diffuse因子进行调节(在这里加入一个可选的设置是因为diffuse因子的值是通过其他值的计算得到，而不像其他因子一样为一个常数)，若选择调节则可对diffuse因子进行滑动调节，范围为0~1
- 对specular因子进行滑动调节，范围为0~1
- 对反光度进行滑动调节，范围为1~256



使用演示

basic-2.mp4

主要代码

GUI设计及将数据传入GPU

```
// Start the Dear ImGui frame
ImGui_ImplOpenGL3_NewFrame();
ImGui_ImplGlfw_NewFrame();
ImGui::NewFrame();
{
    ImGui::Begin("Lighting");
    ImGui::SetWindowFontScale((float)1.0);

    //选择光照模型
    ImGui::RadioButton("Gouraud Shading", &isPhong, 0);
    ImGui::RadioButton("Phong Shading", &isPhong, 1);

    //物体及灯光颜色调节
    ImGui::ColorEdit3("ObjectColor", (float*)&object);
    ImGui::ColorEdit3("LightColor", (float*)&light);

    //ambinent因子调节
    ImGui::SliderFloat("ambientStrength", &ambientStrength, 0, 1);
```

```
        //diffuse因子调节
        ImGui::SliderFloat("diff", &diff, 0, 1);
        ImGui::SameLine();
        ImGui::Checkbox("Diffuse", &isDiffuse);

        //specular因子调节
        ImGui::SliderFloat("specularStrength", &specularStrength, 0, 1);

        //反光度调节
        ImGui::SliderInt("shininess", &shininess, 1, 256);

        if (isPhong) {
            shaderProgram = phongProgram;
        }
        else {
            shaderProgram = gouraudProgram;
        }
        if (!isDiffuse) {
            diff = -1;
        }

        ImGui::Text("Application average %.3f ms/frame (%.1f FPS)", 1000.0f /
ImGui::GetIO().Framerate, ImGui::GetIO().Framerate);
        ImGui::End();
    }

    shaderProgram.use();
    shaderProgram.setVec3("objectColor", object.x, object.y, object.z);
    shaderProgram.setVec3("lightColor", light.x, light.y, light.z);
    shaderProgram.setVec3("lightPos", lightPos);
    shaderProgram.setVec3("viewPos", camera.Position);
    shaderProgram.setFloat("ambientStrength", ambientStrength);
    shaderProgram.setFloat("diffIn", diff);
    shaderProgram.setFloat("specularStrength", specularStrength);
    shaderProgram.setInt("shininess", shininess);
```

Bonus

当前光源为静止状态，尝试使光源在场景中来回移动，光照效果实时更改。

在这里让光源以直线z=y为轴进行旋转，并移动视角进行观察

光源移动代码

```
    lightPos.x = 2 * sin(glfwGetTime()) * 1.0f;
    lightPos.y = 2 * cos(glfwGetTime()) * 1.0f;
    lightPos.z = 2 * cos(glfwGetTime()) * 1.0f;
```

效果录屏

bonus.mp4