个人信息和实验环境

杨雅儒, 2017011071, 计85

Ubuntu 18.04, gcc 7.5.0

问题回答

配置 OpenGL 时的问题

使用 Ubuntu 18.04，直接利用 apt-get install 安装了 OpenGL 库，且安装完成之后即可使用（CMakeLists.txt中已经包含了相关内容），没有遇到什么问题。

OpenGL 绘制逻辑与光线投射绘制逻辑区别

OpenGL 绘制要简单许多，只用定义好光源、各种物体，并且按需要进行正确的配置（例如在使用不同材料时调用 glMaterialfv 函数进行设置，以及显示模式等初始化设置之类）即可使用，代码中的 intersect 实际也是没有必要的，在本代码中仅仅是被用来确定相机到中心物体的距离。

而光线投射绘制需要自行实现代码细节，例如需要实现 intersect 函数用于求交点，需要实现 Shade 函数用于实现 Phong 模型等，相较于OpenGL绘制来说复杂不少。

下面举几个例子。

着色的确定

- OpenGL:

```
void Use() {
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, Vector4f(diffuseColor, 1.0f));
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, Vector4f(specularColor, 1.0f));
    glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS, Vector2f(shininess * 4.0,
1.0f));
}
```

- 光线投射：

```
Vector3f Shade(const Ray &ray, const Hit &hit,
                   const Vector3f &dirToLight, const Vector3f &lightColor) {
    Vector3f shaded = Vector3f::ZERO;
    // 令 T 为交点。 注意 dirToLight 为交点到光源方向的向量
    Vector3f pointT = ray.pointAtParameter(hit.getT());
    Vector3f vecN = hit.getNormal().normalized(), vecLx =
dirToLight.normalized();
    Vector3f vecV = -ray.getDirection().normalized();
    Vector3f vecRx = 2 * Vector3f::dot(vecLx, vecN) * vecN - vecLx;
    return (this->diffuseColor * utils::calcReLU(Vector3f::dot(vecLx, vecN)) +
            this->specularColor * pow(utils::calcReLU(Vector3f::dot(vecV,
vecRx)), this->shininess)) * lightColor;
}
```

物体的绘制

以球体为例，OpenGL中只需要给定球体的位置和大小：

```
void drawGL() override {
    Object3D::drawGL();
    glMatrixMode(GL_MODELVIEW); glPushMatrix();
    glTranslatef(center.x(), center.y(), center.z());
    glutSolidSphere(radius, 80, 80);
    glPopMatrix();
}
```

而光线投射中还需要实现 intersect 函数，之后用 Shade 函数得到具体的着色：

```
bool intersect(const Ray &r, Hit &h, float tmin) override {
    // 令 A 点为射线起点， AB 为射线单位向量， O 点为球心， C 点为 O 点到 AB 的垂足
    Vector3f pointA = r.getOrigin(), pointO = this->center;
    Vector3f vecAB = r.getDirection().normalized(), vecAO = pointO - pointA;
    Vector3f vecAC = vecAB * Vector3f::dot(vecAB, vecAO);
    Vector3f vecCO = pointO - (pointA + vecAC);
    float dis = vecCO.length();
    if (utils::sgn(dis - this->radius) > 0)
        return false;
    else if (utils::sgn(dis - this->radius) == 0) {
        // 相切，此时 C 点即交点
        float t = utils::calcT(r, pointA + vecAC);
        if (t < tmin || t > h.getT())
            return false;
        h.set(t, this->material, -vecCO.normalized());
        return true;
    } else {
        // 有两个交点，找最近的一个，令其为 D 点
        // NOTE: 暂时默认射线起点在球外部
        float lenAD = vecAC.length() - sqrt(this->radius * this->radius -
vecCO.squaredLength());
        assert(utils::sgn(lenAD) > 0);
        Vector3f pointD = pointA + lenAD * vecAB;
        float t = utils::calcT(r, pointD);
        if (t < tmin || t > h.getT())
            return false;
        h.set(t, this->material, (pointD - pointO).normalized());
        return true;
    }
}
```

交流/讨论

并未与任何同学进行交流讨论，也没有借鉴任何网上或其它同学的代码，只是参照网上的做法安装了OpenGL。

如何编译并渲染

在 `code/` 目录下使用如下命令：

```
bash run_all.sh
```

结果将存放在 `code/output/` 目录下。

未解决 bug

就目前测试结果来看，并没有任何未解决的 bug。