第6章真实感图形绘制

1. 实验题目

1.线框模型的茶壶

2.没有光照的固定颜色的茶壶

3.只有环境光，采用单一颜色的茶壶

4.只有环境光和漫反射光，采用 Gouraud 插值着色的茶壶

5.有环境光、漫反射光和镜面高光，采用 Gouraud 插值着色的茶壶

1. 算法描述

从网络上下载茶壶模型 teapot.obj 并自行解析，线框模型使用 Lines 类顶点缓冲，固定颜色使用自行编写的着色器 gl::shader::plain\_shader，环境光使用 gl::shader::ambient\_shader，预设环境光为 30, 含有漫反射与 Gouraud插值是默认 shader, 然后镜面反射使用gl::shader::specular\_shader。

1. 绘图代码部分

本次实验所有代码均基于 rust 语言及其经过安全性包装的 openGL 库 glium。以及我自行编写的 rust 库 gl，用于方便 glium 的调用，其代码可以在附件中文件夹 gl 中找到。

以下是这个实验的所有源代码，也可以查看附件中 Chapter8/src/main.rs。

#[macro\_use]

extern crate glium;

use gl::camera;

use gl::shader;

use gl::action;

use gl::models;

use glium::glutin::event::ElementState;

fn main() {

#[allow(unused\_imports)]

use glium::{glutin, Surface};

let event\_loop = glutin::event\_loop::EventLoop::new();

let wb = glutin::window::WindowBuilder::new();

let cb = glutin::ContextBuilder::new().with\_depth\_buffer(24);

let display = glium::Display::new(wb, cb, &event\_loop).unwrap();

let camera = camera::CameraState::new();

let vertex\_buffer = models::teapot(&display);

let mut step = 0;

let mut t: f32 = 0.0;

action::start\_loop(event\_loop, move |events| {

if t < 6.28 { t += 0.01; } else {t = 0.0};

let indices\_buffer = match step{

0 => glium::index::NoIndices(glium::index::PrimitiveType::LinesList),

\_ => glium::index::NoIndices(glium::index::PrimitiveType::TrianglesList)

};

let program = match step {

0 => shader::get\_plain\_shader(&display),

1 => shader::get\_plain\_shader(&display),

2 => shader::get\_ambient\_shader(&display),

3 => shader::get\_default\_shader(&display),

\_ => shader::get\_specular\_shader(&display)

};

let mut target = display.draw();

target.clear\_color\_and\_depth((0.0, 0.0, 0.0, 1.0), 1.0);

let uniforms = match step {

\_ => uniform! {

perspective: camera.get\_perspective(),

view: camera.get\_view(),

model: [

[0.04 \* t.sin(), 0.0, 0.04\*t.cos(), 0.0],

[0.0, 0.04, 0.0, 0.0],

[0.04\*t.cos(), 0.0, -0.04\*t.sin(), 0.0],

[0.0, 0.0, 0.0, 1.0f32],

],

light: [0.0, 1.0, -1.0f32],

shininess: 20.0f32

}

};

let params = glium::DrawParameters {

depth: glium::Depth {

test: glium::draw\_parameters::DepthTest::IfLess,

write: true,

.. Default::default()

},

.. Default::default()

};

target.draw(&vertex\_buffer, &indices\_buffer, &program, &uniforms, &params).unwrap();

target.finish().unwrap();

let mut action = action::Action::Continue;

for e in events {

match e {

glutin::event::Event::WindowEvent { event, .. } => match event {

glutin::event::WindowEvent::CloseRequested =>

{ action = action::Action::Stop; },

glutin::event::WindowEvent::KeyboardInput { device\_id: \_, input, is\_synthetic:\_ } =>

{ match input.state {

ElementState::Pressed => {step += 1;}

\_ => {}

} }

\_ => (),

},

\_ => (),

}

}

action

});

}

1. 实验结果截图

在配制好的环境下运行上述代码，每次按下空格可以得到一题的输出，具体如下。

线框模型

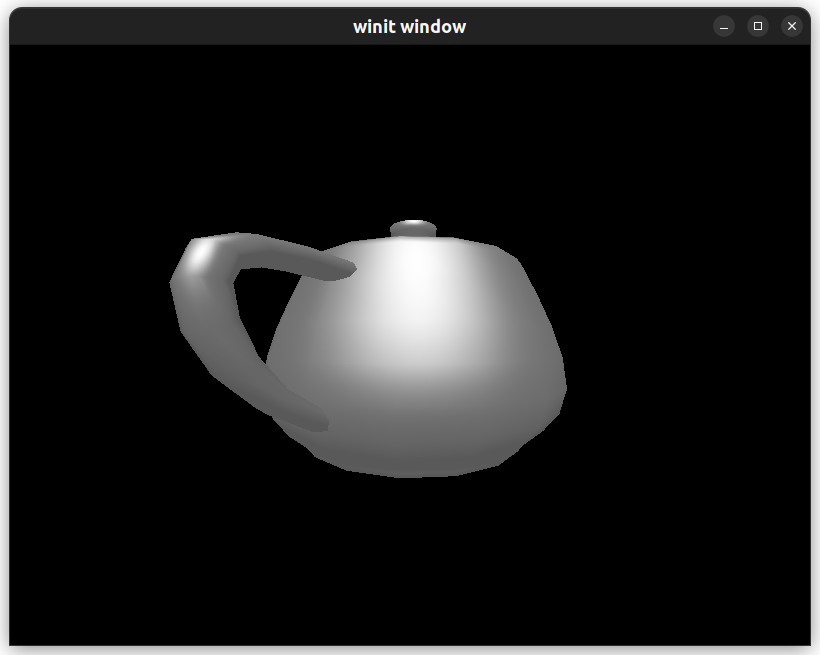
纯色模型

环境光

漫反射光



镜面高光



1. 实验小结

通过本次实验我学习了计算机图形学中光线的计算方式以及进一步了解了着色器的使用。