选做题实验报告

1. 实验题目

1.将屏幕上的鼠标选取点进行连线，左键点击选中一个控制点（控制点数不超过 64），按f 键擦除第一个控制点，按 l 键擦除最后一个控制点，按 Escape 键退出。

2.将屏幕上的鼠标选取点作为多边形顶点进行填充。

3.模拟简单的太阳系，太阳在中心，地球每 365 天绕太阳转一周，月球每年绕地球12  
周。另外，地球每天 24 个小时绕它自己的轴旋转。

1. 算法描述

第一二题核心是鼠标与键盘事件处理，这里用一个vector存放顶点，事件处理代码如下

match e {

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

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

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

glutin::event::WindowEvent::KeyboardInput { input, .. }

=> match input.state {

ElementState::Pressed => {

match input.virtual\_keycode {

Some(VirtualKeyCode::Escape) => {action = action::Action::Stop;}

Some(VirtualKeyCode::Key1) => {vertex.pop();}

Some(VirtualKeyCode::F) => { if vertex.len() >= 1 {vertex.remove(0);} }

Some(VirtualKeyCode::Space) => { fill = !fill; }

\_ => {}

}

}

\_ => {}

}

glutin::event::WindowEvent::CursorMoved { position, .. }

=> { let (h, w) = display.get\_framebuffer\_dimensions();

pos = [position.x as f32 \* 2.0 / h as f32 - 1.0, 1.0 - position.y as f32 \* 2.0/ w as f32]; }

glutin::event::WindowEvent::MouseInput { state, .. }

=> match state {

ElementState::Pressed => {

vertex.push( vertex::Vertex::new\_2d(pos[0], pos[1]) );

}

\_ => {}

}

\_ => (),

},

\_ => ()

}

第三题主要是球模型的绘制，这里使用了从网上下载的 sphere.obj

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 Chapter\_ex1/src/main.rs 与 Chapter\_ex2/src/main.rs。

#[macro\_use]

#[macro\_use]

extern crate glium;

use gl::camera;

use gl::shader;

use gl::action;

use gl::vertex;

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

use glium::glutin::event::VirtualKeyCode;

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 params = glium::DrawParameters {

depth: glium::Depth {

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

write: true,

.. Default::default()

},

.. Default::default()

};

let uniforms = uniform! {

perspective: camera::CameraState::flat\_perspective(),

view: camera::CameraState::flat\_view(),

model: camera::CameraState::element\_matrix()

};

let program = shader::get\_default\_shader(&display);

let mut vertex = Vec::new();

let mut fill = false;

let mut pos = [0.0, 0.0f32];

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

let mut target = display.draw();

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

let vertex\_buffer = vertex::from\_vertex(&display, &vertex);

let indices\_buffer = if fill {

glium::index::NoIndices(glium::index::PrimitiveType::TriangleFan)

} else {

glium::index::NoIndices(glium::index::PrimitiveType::LineStrip)

};

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 { input, .. }

=> match input.state {

ElementState::Pressed => {

match input.virtual\_keycode {

Some(VirtualKeyCode::Escape) => {action = action::Action::Stop;}

Some(VirtualKeyCode::Key1) => {vertex.pop();}

Some(VirtualKeyCode::F) => { if vertex.len() >= 1 {vertex.remove(0);} }

Some(VirtualKeyCode::Space) => { fill = !fill; }

\_ => {}

}

}

\_ => {}

}

glutin::event::WindowEvent::CursorMoved { position, .. }

=> { let (h, w) = display.get\_framebuffer\_dimensions();

pos = [position.x as f32 \* 2.0 / h as f32 - 1.0, 1.0 - position.y as f32 \* 2.0/ w as f32]; }

glutin::event::WindowEvent::MouseInput { state, .. }

=> match state {

ElementState::Pressed => {

vertex.push( vertex::Vertex::new\_2d(pos[0], pos[1]) );

}

\_ => {}

}

\_ => (),

},

\_ => ()

}

}

action

});

}

1. 实验结果截图

在配制好的环境下运行上述代码，具体如下。

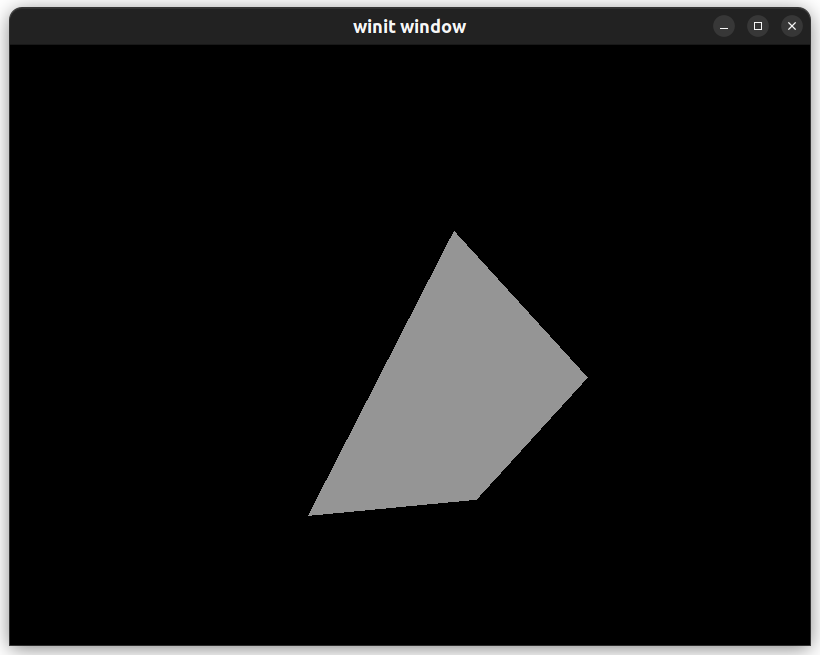
鼠标选择

按下f

按下 1



按下空格



1. 实验小结

通过本次实验，我对 openGL 的综合了解和代码编写能力有所提升。