学习材料：OpenGL 绘制

指导教师：胡事民　助教：陈拓、冯启源

2025 年 4 月 1 日

1　细节说明

1.1　OpenGL 简介

OpenGL（Open Graphics Library）是一个跨编程语言、跨平台的编程图形程序接口（API），它将计算机的资源抽象称为一个个 OpenGL 的对象，对这些资源的操作抽象为一个个的 GL 指令。它相当于一个“桥梁”，沟通了图形软件和显示硬件（例如 GPU）：暴雪/育碧等厂商的游戏大作、广泛应用于工程领域的 CAD 软件、时下最新的 AR/VR 引擎等各类软件在绘制三维场景的时候都需要调用 OpenGL 定义的接口；而 Nvidia、AMD、Intel 等硬件生产厂商则需要生产能够适配这些接口标准的独立或集成显卡。

除了 OpenGL 之外，还有许多其他的图形绘制标准接口，例如微软的 DirectX 系列、Vulkan 等等，相比之下 OpenGL 适配平台更加广泛（支持几乎所有操作系统，包括安卓）、支持社区更完善（这意味着更完善的资料和技术支持），是图形学爱好者和开发者的首选。

1.2　状态机模型

OpenGL 自身是一个巨大的状态机，状态机中包含了一系列变量，这个状态通常被称为“上下文”（Context）。当调用了状态设置函数之后，所有之后执行的绘制指令都会依据当前的状态。例如：

```
// 绘制一条 (0,1) 到 (1,0) 的线段
void drawLine() {
  glBegin(GL_LINES); glVertex2f(0.0, 1.0); glVertex2f(1.0, 0.0); glEnd();
}
glColor3f(1.0, 0.0, 0.0);     // 设置绘制颜色为红色
drawLine();                   // 绘制红色线段
glColor3f(0.0, 1.0, 0.0);     // 设置绘制颜色为绿色
drawLine();                   // 同样的函数，此时绘制的线段为绿色
```

只要你记住 OpenGL 本质上是一个大状态机，就更容易理解他的大部分特性。有时候明明调用了绘制函数，屏幕上却显示一片黑，此时很有可能是上下文设置出现了问题（例如灯光错误、材质错误、相机参数错误等）。

1.3　简单三维绘制

使用 OpenGL 进行绘制可以非常简洁，你只需要通过程序告诉 OpenGL 需要绘制的三维物体形状，OpenGL 就能够自动处理消隐（深度测试）、染色、光栅化等操作。一个简单的 OpenGL 例程如下所示：

```cpp
// 编译选项: g++ main.cpp -o main -lglut -lGL
#include <GL/glut.h>

void render() {
    // 设置背景色: 矢车菊蓝
    glClearColor(0.392, 0.584, 0.930, 1.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    // 立即模式中的绘制以glBegin和glEnd包裹
    // GL_POINTS: 绘制点
    // GL_LINES: 绘制线段
    // GL_TRIANGLES: 绘制三角形
    glBegin(GL_TRIANGLES);
    glColor3f(1.0, 0.0, 0.0); glVertex2f(0.5, -0.5);     // 红
    glColor3f(0.0, 1.0, 0.0); glVertex2f(-0.5, -0.5);    // 绿
    glColor3f(0.0, 0.0, 1.0); glVertex2f(0.0, 0.5);      // 蓝
    glEnd();
    // 渲染图片
    glFlush();
}

int main(int argc, char** argv) {
    // 初始化GLUT, 它负责创建OpenGL环境以及一个GUI窗口
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowPosition(60, 60);
    glutInitWindowSize(640, 480);
    glutCreateWindow("PA2 Immediate Mode");
    // 设置绘制函数为render()
    glutDisplayFunc(render);
    // 开始UI主循环
    glutMainLoop();
    return 0;
}
```

这段例程绘制的效果如图1所示。

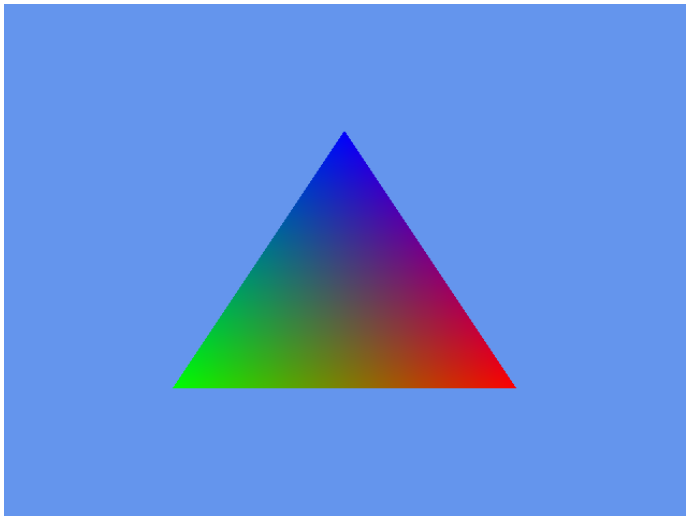

图 1: 立即模式绘制效果

为了绘制更加复杂的物体，我们使用 GLUT（全称为 OpenGL Utility Toolkit）库，该库调用 OpenGL 以及某些操作系统的底层函数，能够方便地实现复杂图形的绘制。

```cpp
// 绘制一个半径为1.0，纵向切片为60，横向切片为80的实心球体
```

```
glutSolidSphere(1.0, 60, 80);
// 绘制一个边长为1.0的实心正方体
glutSolidCube(1.0);
// 绘制一个大小为1.0的犹他茶壶 (Wikipedia: Utah Teapot)
glutSolidTeapot(1.0);
// 绘制一个二十面体
glutSolidIcosahedron();
// 使用24号Times New Roman字体绘制字符S
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, 'S');
```

除了绘制复杂物体之外，GLUT 还支持图形界面的创建和管理，例如创建窗口、处理鼠标键盘事件等，具体的使用示例请参见框架代码。

1.4 渲染模式和 OpenGL 版本

OpenGL 接口规范由Khronos组织维护，截至目前最新的版本为 4.6。早期的 OpenGL 使用立即渲染模式（Immediate mode），这种模式使用固定渲染管线：坐标变换、着色、光栅化等一系列操作均由标准定义好，使用起来非常方便。*实际上，1.3节的简单绘制代码就是使用的这种模式，本次 PA 的所有代码也均以这种模式进行实现。*立即渲染模式的缺点在于：开发者无法对渲染管线进行自定义的修改，大场景下的绘制的效率也比较低。[1]

随着时间的推移，从 OpenGL3.2 版本开始，立即渲染模式被废弃，开发者被鼓励于核心模式（Core-profile）下进行开发，这种模式是目前现代图形软件（包括手机等终端）所使用的主流绘制模式，具有效率高、灵活性强的特点。但是出于教学目的，本次的代码将不使用核心模式，如果同学们对现代 OpenGL 绘制感兴趣，可以参考LearnOpenGL以及其中文翻译版本。其中每一个教程都提供了代码和示例可供参考，是学习现代 OpenGL 的较好入门资料。

2 框架代码说明

2.1 环境配置与编译

我们非常建议使用使用带有 `CMake` 套件的 `Ubuntu` 系统进行编程，`Windows 10` 下可以使用 `Ubuntu Subsystem`（下称 wsl）。

- 我使用的 Ubuntu（非 wsl），如何配置环境?

  安装 `mesa-common-dev` 与 `freeglut3-dev`

- 我使用的 Ubuntu（wsl 2），如何配置环境?

  除了前面说的两个包，还需要配置图形环境。不建议使用 Windows 于去年更新的 WSLg，WSLg 对 OpenGL 的支持还有待完善。请按照下列步骤配置图形化环境。

  如果是 wsl 1（现在应该比较罕见了）可以忽略掉中间两步。

  1. 在 Windows 上安装 `vcxsrv`（请勿使用 XMing）
  2. 在 Windows 上运行 `XLaunch`，不要勾选 native OpenGL，需要勾选”Disable access control”
  3. 在 Ubuntu 上使用 `cat /etc/resolv.conf` 查看 ip 地址，执行 `export DISPLAY=<ip>:0`，`<ip>` 处填入你刚才查看到的 ip。

[1] 比你想象中的“低效”还是有所差距的，著名的《雷神之锤 II》就使用立即模式编写。

4. 在 Ubuntu 上使用 `sudo apt install x11-apps` 安装 `xeyes`，执行 `xeyes`，看你的屏幕上是否出现一对眼睛，如果出现，说明图形环境配置好了。

- 我使用的 Ubuntu（wsl 2），运行起来会报段错误（Segment Fault），该怎么办?

  请首先确认自己的程序是在 `src/main.cpp` 中的 `exit(0)` 处报错，否则有可能只是你单纯实现错了。

  太长不看版：`export LIBGL_ALWAYS_SOFTWARE=1`

  解释：这可能是因为N 卡在 WSL 上的 bug。在 2022 年的一次更新后，wsl 上的 OpenGL 可以调用显卡来进行渲染，但这是有 bug 的（参见前面的链接）。解决方法是不要让 wsl 的 OpenGL 调用硬件，因此设置 `export LIBGL_ALWAYS_SOFTWARE=1` 让 OpenGL 总是使用 CPU 计算而不是去调用 GPU。

- 我使用的 MacOS，该如何配置环境?

  应该不需要专门的配置，用 `homebrew` 安个 `freeglut3` 应该就够了。助教没用过 mac，环境上出了问题请先在群里提看看有没有用 mac 的同学能回答。

- 我怎么改输出都是一片黑的，该怎么办?

  这可能是因为环境问题。如果发现 `output/scenexx.bmp` 中的图像是全黑的，可以尝试将 `src/main.cpp` 中第 31 行的 glReadBuffer(GL_BACK); 改为 glReadBuffer(GL_FRONT); 后重新运行 run_all.sh。

  这一般发生在 mac 环境上，提交到 oj 前记得改回来。

安装完成之后，请在包含有 `run_all.sh` 的文件夹下打开终端，并执行：

```
bash ./run_all.sh
```

这段脚本会自动设置编译，并在 6 个测例上运行你的程序。你的程序最终会被编译到 `build/`文件夹中。

程序运行后会弹出窗口显示图像并保存图像到 output 目录。

你还可以执行：

```
./build/PA2 testcases/scene01_basic.txt
```

进入交互模式（`scene01_basic.txt` 可替换成其他场景文件名），交互模式的操作说明如下（参考 `CameraController` 类）：

- 左键拖动：旋转
- 右键拖动：缩放
- 中间拖动：平移

关闭窗口即可结束程序。

对于使用 `Windows` 系统的同学，我们在 `deps/`文件夹中提供了 `GLUT` 库，你需要在 `VC++` 项目中加入相应的包含目录、库目录和附加依赖项。

3 致谢

本实验文档和代码部分借鉴于MIT Open Courseware，按照其发布协议，本文档原则上允许同学们以CC BY-NC-SA 4.0协议共享引用，但是由于教学需要请同学们尽量不要将本文档或框架代码随意传播，感谢同学们的支持。