2018-2019年度第二学期 00106501

计算机图形学

童 伟 华　管理科研楼1205室

E-mail: tongwh@ustc.edu.cn

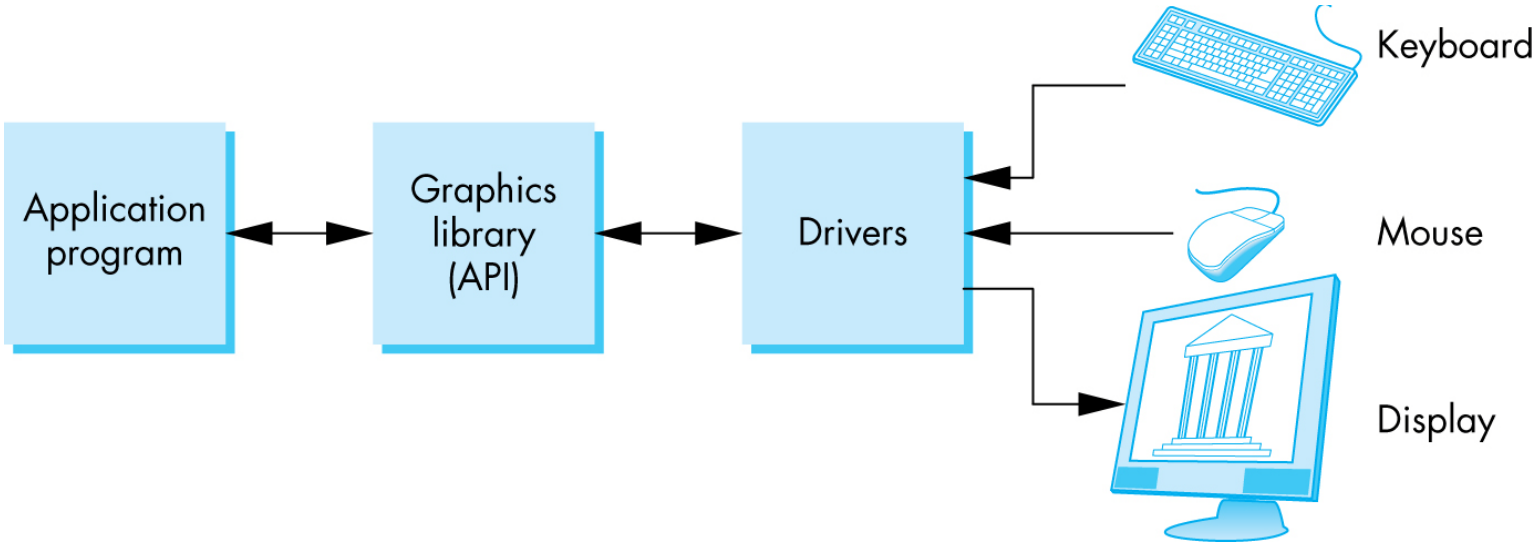
中国科学技术大学　数学科学学院

http://math.ustc.edu.cn/

附讲八　OpenGL编程（一）

图形程序编程接口

- 应用程序编程接口（Application Programmer’s Interface，API）：应用程序与图形系统之间的接口，常通过图形库函数来指定



图形程序编程接口

- **目前主流的图形程序编程接口：**
  - OpenGL：跨平台，支持Windows，Mac，Linux以及手持设备（OpenGL ES）；
  - DirectX：仅Windows操作系统
  - 目前几乎所有的图形加速卡都支持上述两种编程接口

注：更准确的说，OpenGL是一个接口规范，规定了每个函数该如何执行，以及它们的输出值，至于内部具体每个函数是如何实现的，将由OpenGL库的开发者自行决定。因此，实际的OpenGL库的开发者通常是显卡的生产商。

- **未来趋势：**
  - Vulkan：下一代跨平台的图形编程接口，计划取代OpenGL及OpenGL ES，优点：轻量级、更贴近底层硬件(close-to-the-metal)的接口，可使GPU驱动软件运用多核与多线程CPU性能
  - Metal：苹果公司提出来的下一代图形编程接口

OpenGL历史

- 1992年 SGI领导的OpenGL Architectural Review Board(OpenGL ARB)发布1.0版
  - 平台无关的API；
  - 易于使用
  - 与硬件非常贴近，从而可以充分发挥其性能
  - 着重在于渲染（rendering）
  - 没有提供窗口和输入接口，从而避免依赖于具体的窗口系统
- 早期是由ARB掌控其发展，通过扩展支持平台相关的特性
- 2006年，ARB被Khronos工作组取代

OpenGL版本

- 早期版本：
  - OpenGL 1.1：Visual Studio自带的头文件
- 最新版本：
  - OpenGL 4.6（2018年5月发布，基于Shader）
- 如何查看图形卡支持的OpenGL版本
  - 软件：AIDA64，GLView等
- 其他版本：
  - OpenGL ES：嵌入式系统（Embedded systems）版本
  - WebGL：OpenGL ES的Javascript实现，网络浏览器上的版本

Direct3D

- DirectX: 微软开发的多媒体编程接口
- Direct3D: DirectX的3D图形API
  - 11: Windows 7&Vista, 2009
  - 12: Windows 8&10, 2015

OpenGL V.S. Direct3D

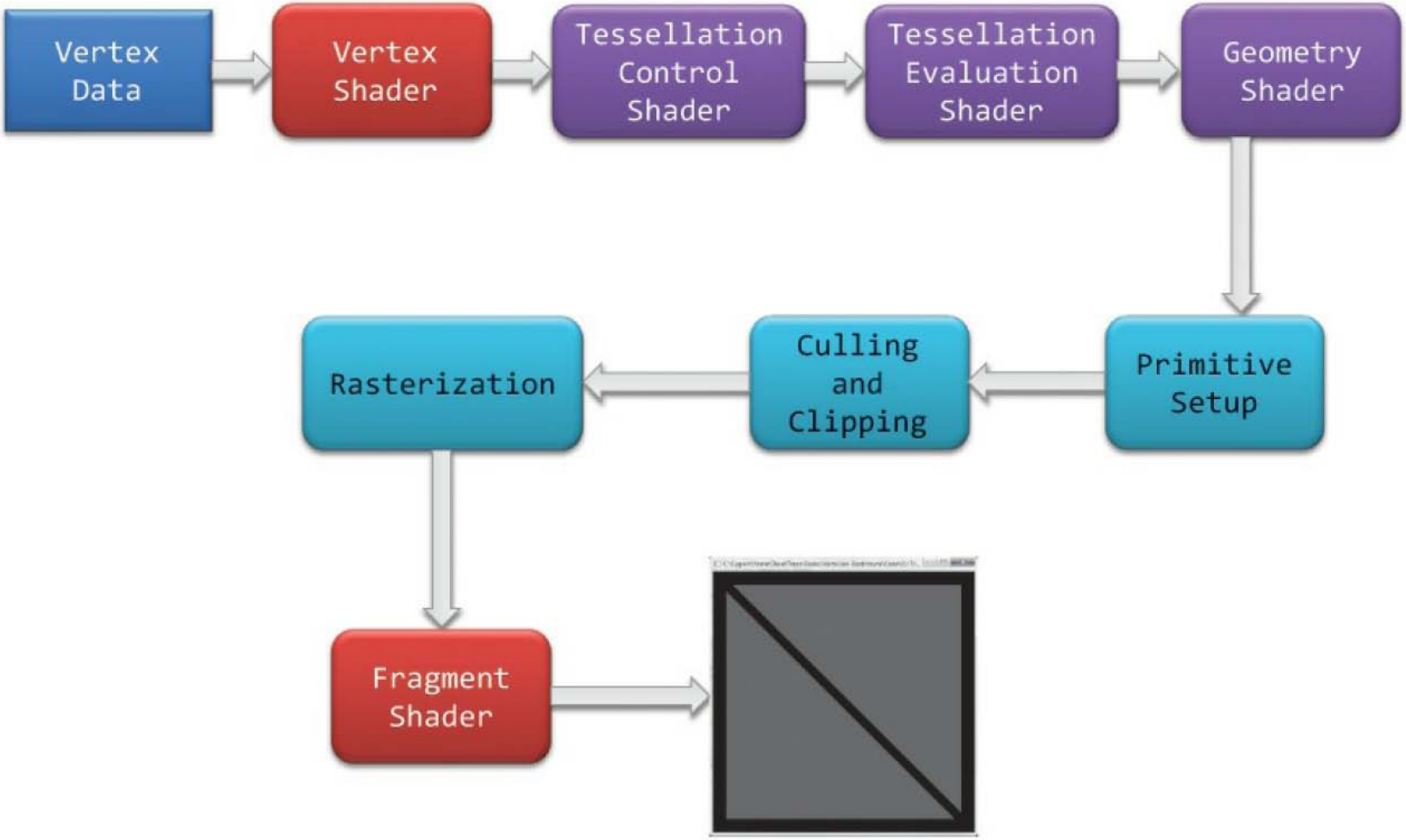
OpenGL

- 跨平台的开放式标准API
  - 可扩展机制
- 可硬件加速的3D渲染系统
  - 底层实现（驱动）管理硬件
- 专业图形应用、科研
  - 跨平台，可移植
- 适合图形学教学

Direct3D

- Windows平台的专利API
  - 一致性好
- 3D硬件接口
  - 应用程序管理硬件资源
- 计算机游戏
  - 高性能硬件存取能力
- 非向下兼容

OpenGL流水线架构



OpenGL核心库

- OpenGL核心库（OpenGL Core Library）
  - Windows: OpenGL32.dll (WINDOWS\SYSTEM32)
    - Windows XP支持OpenGL 1.1，Vista支持1.4
    - Direct3D的封装，需安装驱动来实现硬件加速
  - 大多数Unix/Linux系统：GL库 (libGL.a)
- OpenGL实用库（OpenGL Utility Library, GLU）
  - OpenGL的一部分
    - Windows: glu32.dll
  - 利用OpenGL核心库提供一些功能，避免重复编写代码
  - 二次曲面、NURBS、多边形网格化等
- OpenGL与操作系统连接库
  - 作用：粘合OpenGL和窗口系统
  - X Window系统：GLX
  - Windows: WGL
  - Macintosh: AGL

OpenGL辅助库

- OpenGL实用工具库（OpenGL Utility Toolkit Library, GLUT）
  - 提供所有窗口系统的共同功能
    - 创建窗口
    - 从鼠标和键盘获取输入
    - 菜单
    - 事件驱动
- 代码可以在平台间移植，但是GLUT缺乏一些现代GUI的控件和功能
  - 无滚动条
  - GLUT是上世纪九十年代开发的->FreeGLUT或GLFW
  - 可用FLTK、SDL、GLUI等开发界面
- 课程目标：GLUT -> Qt中的OpenGL编程框架

GLEW

- 在Windows操作系统下，若要使用最新版本的OpenGL功能
  - 使用GLEW库（OpenGL Extension Wrangler Library）（还有其它库GL3W，Glad等）
  - 查看图形卡支持的OpenGL版本
- 目的：对于特殊的操作系统，使得调用OpenGL扩展功能更简单
- 方法：增加glew.h头文件及库文件glew32.lib, glew32.dll，调用glewInit()即可

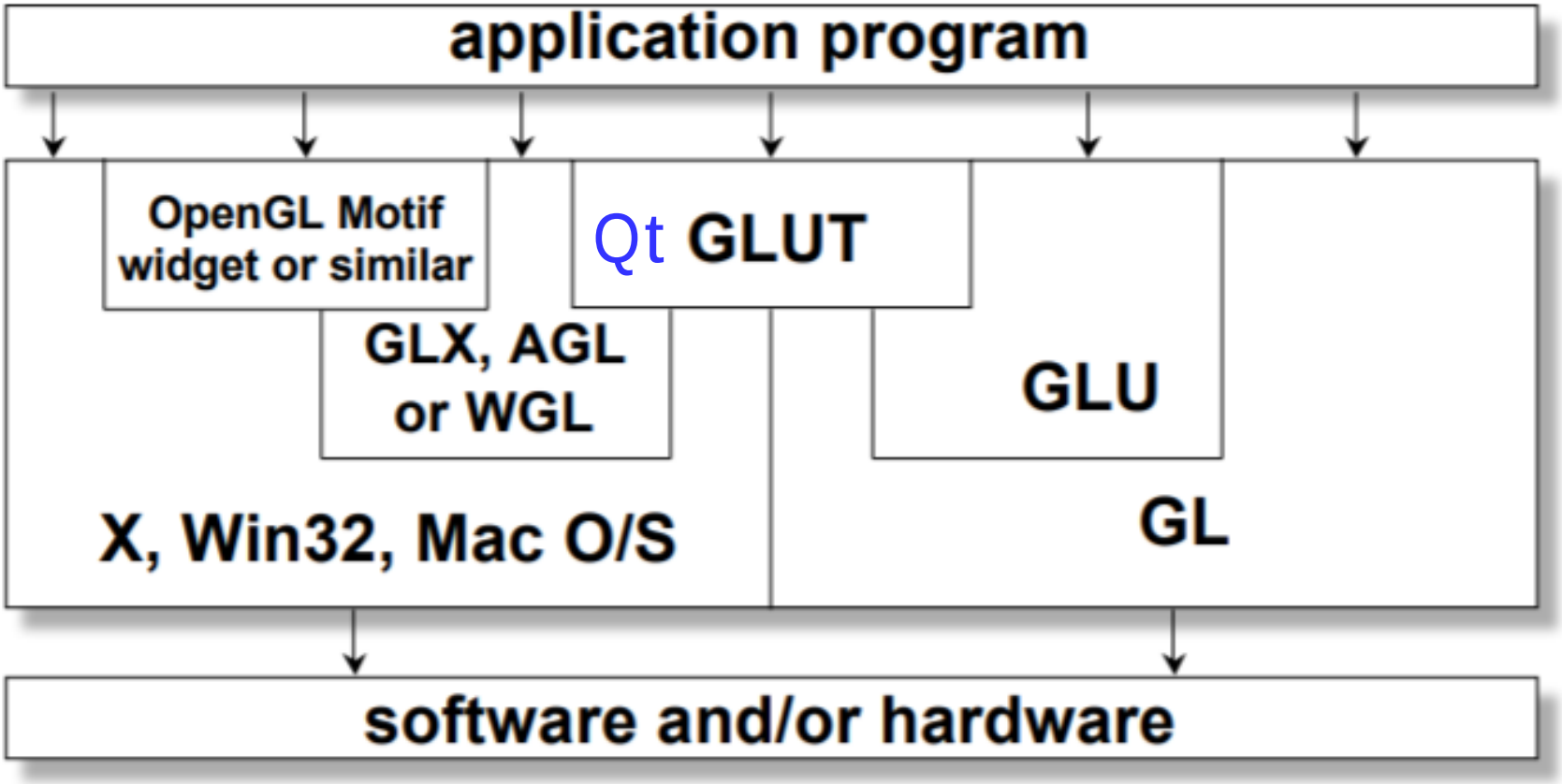
OpenGL Core vs compatibility

- 图形加速卡（GPU）技术日新月异，OpenGL库一方面需要接收最新的技术，另一方面为了保持兼容性，出现了两种模式：
  - 兼容OpenGL 3.2之前的版本：compatibility profile
  - OpenGL 3.2之后的版本：core profile（与compatibility profile不兼容，有部分函数相同）
  - 差别：core profile完全基于shader，每个应用程序必须提供一个vertex shader和fragment shader，不支持立即渲染模式（immediate mode）
- OpenGL编程权威书籍《OpenGL Programming Guide》的版本
  - 第一至七版：按compatibility profile编写（容易上手，但是程序性能会打折扣）
  - 第八、九版：按core profile编写（更加灵活，高效）

OpenGL的书籍与相关网站

- **权威书籍：**
  - 《OpenGL Programming Guide》，第七、八、九版， Addison-Wesley （“红宝书”，中文版，机械工业出版社）
  - 《OpenGL Shading Language》，第三版，（“橙宝书”）
  - 《 OpenGL SuperBible》，第六版， Addison-Wesley
- **相关网站：**
  - 官网：www.opengl.org
  - 大全：www.khronos.org/opengl/wiki/
  - LearnOpenGL（最好的学习OpenGL的资料）：https://learnopengl.com/ （英文）或 https://learnopengl-cn.github.io/（中文）

OpenGL编程接口框架



OpenGL的函数

- 图元函数（primitive） – what
  - 点
  - 线段
  - 多边形
- 属性函数（attribute） – how
- 变换函数（transformation）
  - 视图函数（viewing）
  - 建模函数（modeling）
- 输入函数（input） GLUT or Qt, …
- 控制函数（control） GLUT or Qt, …
- 查询函数（query）

OpenGL的状态机模型

- OpenGL使用状态机模型
  - 一系列的变量描述OpenGL此刻应当如何运行
  - OpenGL的状态通常被称为OpenGL上下文（context）
- OpenGL 函数有两大类型
  - 状态设置函数（state-changing function）：这类函数将会改变上下文
  - 状态使用函数（State-using Function）：这类函数会根据当前OpenGL的状态执行一些操作生成图元（图元函数，如glVertex）
- OpenGL中的对象（Object）
  - 指一些选项的集合，它代表OpenGL状态的一个子集
  - 是抽象层的一个概念，与面向对象编程无关
  - 好处：可以不止定义一个对象，并设置它们的选项。在执行一个使用OpenGL状态的操作的时候，只需要绑定相应设置的对象即可，而不需要再重复设置选项

OpenGL的状态机模型

- 更改OpenGL状态：设置选项，操作缓冲

```
// OpenGL的状态
struct OpenGL_Context {
...
object* object_Window_Target;
...
};
// 创建对象
unsigned int objectId = 0;
glGenObject(1, &objectId);
// 绑定对象至上下文
glBindObject(GL_WINDOW_TARGET, objectId);
// 设置当前绑定到 GL_WINDOW_TARGET 的对象的一些
glSetObjectOption(GL_WINDOW_TARGET, GL_OPTION_WINDOW_WIDTH, 800);
glSetObjectOption(GL_WINDOW_TARGET, GL_OPTION_WINDOW_HEIGHT, 600);
// 将上下文对象设回默认
glBindObject(GL_WINDOW_TARGET, 0);
```

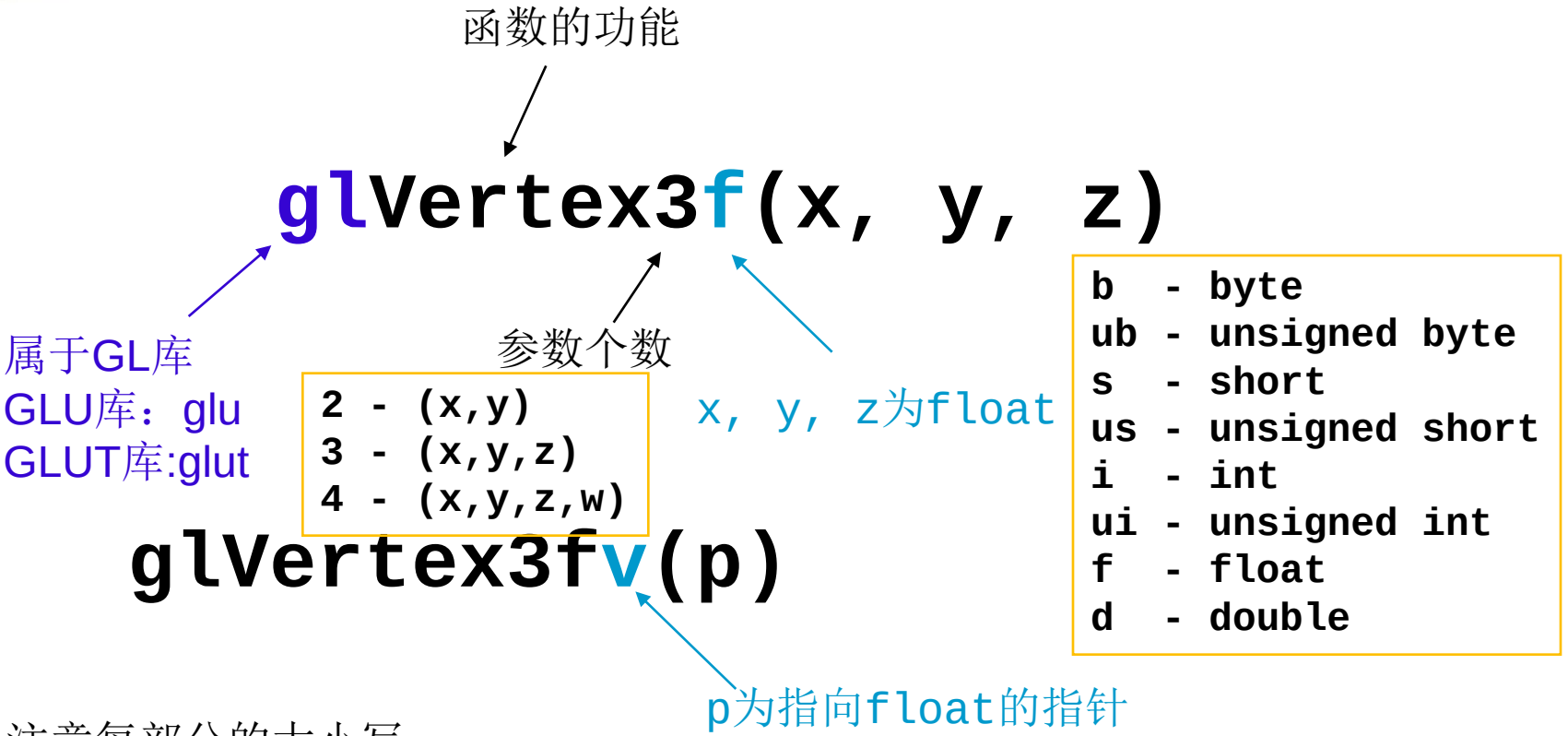
面向对象方面的缺陷

- 标准的OpenGL是一个“C”语言形式的库，不是面向对象的，因此逻辑上的一个函数却对应着多个OpenGL函数:

```
glVertex3f
glVertex2i
glVertex3dv
```

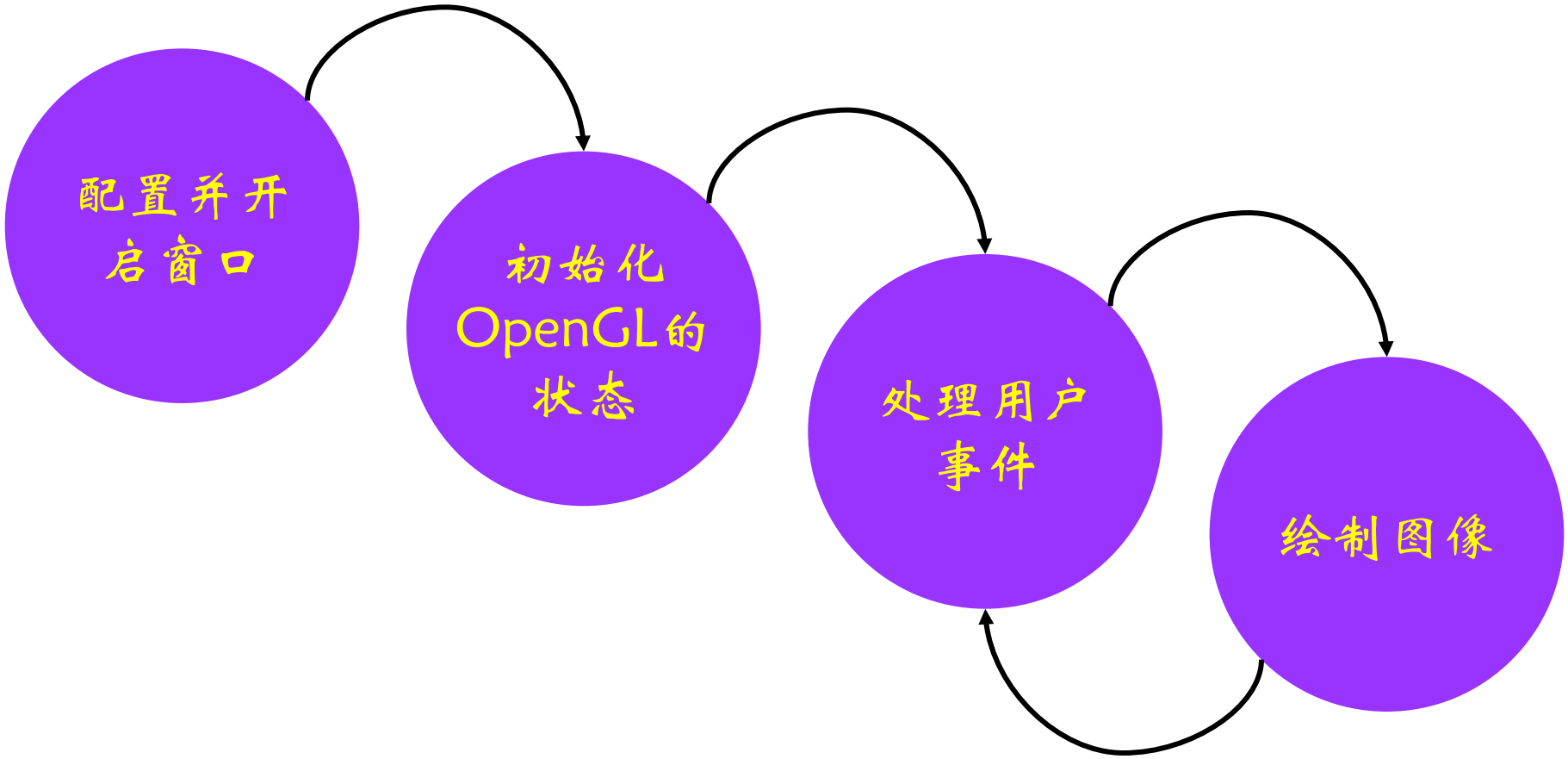
- 内在存储模式是相同的
- 在C++中很容易创建重载函数，当然OpenGL也有其他语言的派生

OpenGL函数名称的格式



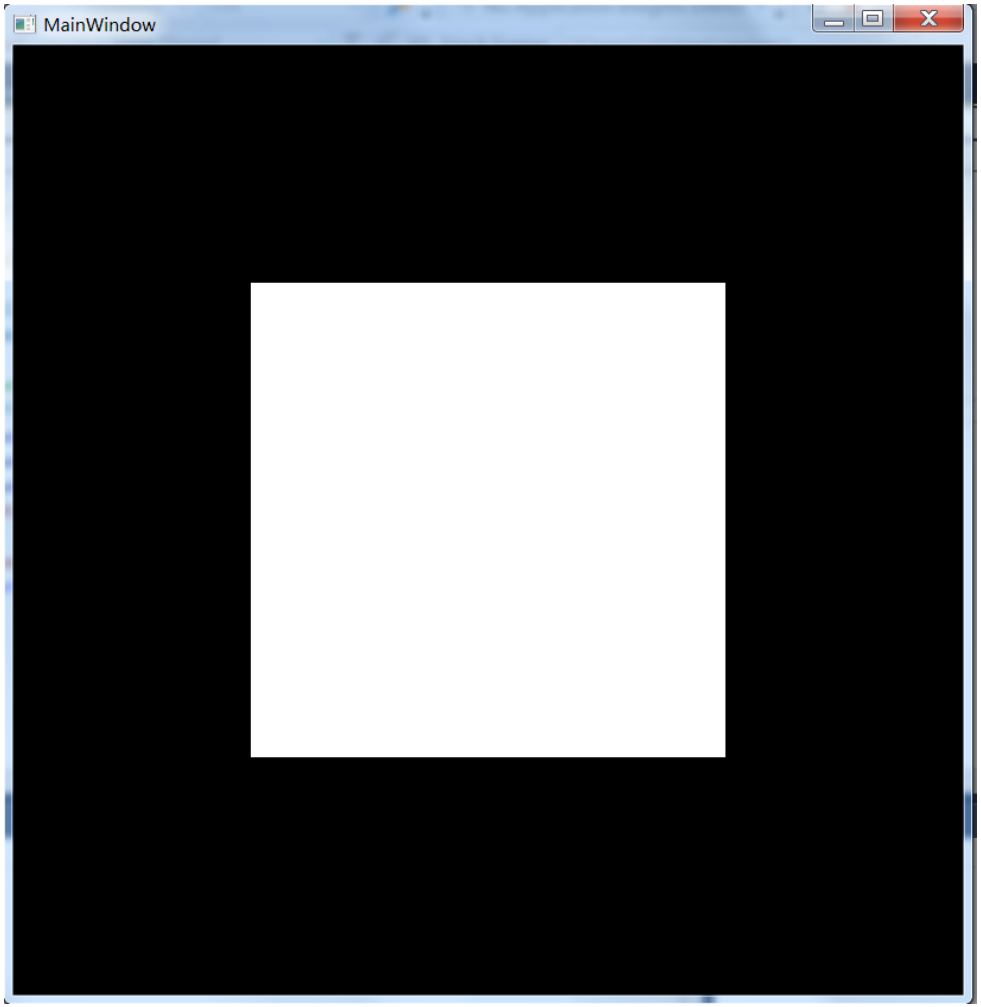
注意每部分的大小写

OpenGL程序的一般结构



一个简单程序

- 在黑色背景上画一个白色矩形



main.cpp

```
#include "mainwindow.h"
#include <QtWidgets/QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;

    // setup OpenGL format
    QSurfaceFormat format;
    format.setVersion(4, 6);
    format.setProfile(QSurfaceFormat::CompatibilityProfile);
    format.setSamples(8);
    w.setFormat(format); // must be called before the widget or its
parent window gets shown

    w.show();
    return a.exec();
}
```

mainwindow.h

```
…
class MainWindow : public QOpenGLWidget
{
    Q_OBJECT

public:
    MainWindow(QOpenGLWidget *parent = 0);
    ~MainWindow();

protected:
    void initializeGL() override;
    void paintGL() override;
    void resizeGL(int w, int h) override;

private:
    Ui::MainWindowClass ui;
};
…
```

mainwindow.cpp

```
void MainWindow::initializeGL()
{
    // initialize GLEW library
    glewExperimental = true;
    glewInit();

    // set the background color
    glClearColor(0.2f, 0.3f, 0.3f, 1.0f);
}

void MainWindow::resizeGL(int w, int h)
{
    // set the viewport
    glViewport(0, 0, w, h);
}
```

mainwindow.cpp

```
void MainWindow::paintGL()
{
    // render scenes
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_POLYGON);
        glVertex2d(-0.5, -0.5);
        glVertex2d(-0.5, 0.5);
        glVertex2d(0.5, 0.5);
        glVertex2d(0.5, -0.5);
    glEnd();
    glFlush();
}
```

Qt中与OpenGL相关的类

- 在Qt 5中，与OpenGL相关的类被分解到Qt GUI与Qt Widgets模块
  - Qt Widgets模块：QOpenGLWidget
  - Qt GUI模块：QOpenGLWindow，OpenGLFunctions, OpenGL Shader, OpenGLProgram, OpenGLBuffer, …
- 在Qt 5中有三种方式使用OpenGL：
  - 继承QOpenGLWindow类，覆盖虚函数：initializeGL()，paintGL()，resizeGL()；
  - 继承QOpenGLWidget类，覆盖虚函数：initializeGL()，paintGL()，resizeGL()；
  - 多重继承QWindow，QOpenGLFunctions，通过setSurface(Qwindow::OpenGLSurface)，实现相关的函数。

覆盖相关的虚函数

初始化OpenGL: initializeGL()

```
    void glewInit(void)
```

窗口大小改变回调函数: resizeGL(int w, int h)

```
    void glViewport(GLint x, GLint y, GLsizei width, GLsizei
height)
```

绘制回调函数: paintGL()

清空缓冲区:

```
    void glClear(Glbitfield mask)
```

图元定义：mode可取GL_POINTS、GL_LINES、GL_POLYGON

```
    void glBegin(Glenum mode)      // 开始mode型对象定义
    void glEnd()                   // 结束顶点序列
```

强制执行OpenGL命令:

```
    void glFlush()
```

注意：为了降低学习难度，上面的例子使用compatibility profile编写

事件循环

- 在程序中定义了一个绘制函数（虚函数）：paintGL()
  - 只要OpenGL确定显示内容要被刷新时，绘制函数就会被调用：例如，当窗口被打开的时候
  - main函数是以程序进入事件循环做为结束

默认值

- 立即绘制模式非常简单
- 大量使用状态变量的默认值
  - 视图
  - 颜色
  - 窗口参数
- 逐渐过渡到：核心绘制模式（core profile）

程序里需要什么

- 头文件

```
#include <GL/glew.h> //自动包含gl.h, glu.h
```

- 库文件
  - 库文件：编译器或系统库文件目录\opengl32.lib glu32.lib glew32.lib
  - 动态链接库文件：操作系统目录\system32\opengl32.dll glu32.dll glew32.dll
- 数据类型
  - 为了兼容性，OpenGL定义了各种数据类型(#define)
    - GLfloat, GLint, GLenum, etc.

如何编译

- 仅以Visual Studio 2013为例
  - 编译GLEW库，并把头文件（glew.h）、库文件（glew32.lib）及动态库文件（glew32.dll）拷贝到相应的目录
  - 创建一个Qt GUI Application类型的工程
  - 编写C++代码
  - 进入菜单 Project →Properties，选择Link标签，Input子项，在Additional Dependencies文本编辑控件中加上opengl32.lib; glu32.lib; glew32.lib（注意用“;”分开）

Thanks for your attention!