OPENGL

OpenGL规范严格规定了

- 每个函数该如何执行
- 以及它们的输出值
- 不规定内部实现，将由OpenGL库的开发者自行决定
- OpenGL库的开发者通常是显卡的生产商

立即渲染模式(IMMEDIATE MODE)

固定渲染管线
绘制图形很方便
OpenGL的大多数功能都被库隐藏起来
很少有控制OpenGL如何进行计算的自由

核心模式(CORE-PROFILE)

提供了更多的灵活性
更高的效率
更深入的理解图形编程

扩展

当一个显卡公司提出一个**新特性**或者**渲染上的大优化**，通常会以**扩展**的方式在驱动中实现

```
if(GL_ARB_extension_name)
{
// 使用硬件支持的全新的现代特性
}
else {
// 不支持此扩展：用旧的方式去做
}
```

状态机

OpenGL是一个巨大的状态机(State Machine)

- 一系列的变量描述OpenGL此刻应当如何运行
  OpenGL的状态被称为OpenGL上下文(Context)
- 更改OpenGL状态方式：设置选项，操作缓冲
  状态相关函数
- 状态设置函数(State-changing Function)，将会改变上下文

- 状态使用函数(State-using Function)，根据当前OpenGL的状态执行一些操作

对 象

把OpenGL上下文看作一个大的结构体

```
// OpenGL的状态
struct OpenGL_Context
{
...
object* object_Window_Target;
...
};
```

使用对象的方式

```
// 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 的对象的一些选项
 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);
```

- 创建一个对象，用一个id保存它的引用（实际数据被储存在后台）。
- 将对象绑定至上下文的目标位置（例子中窗口对象目标的位置被定义成GL_WINDOW_TARGET）。
- 设置窗口的选项。
- 最后将目标位置的对象id设回0，解绑这个对象。
  设置的选项将被保存在objectId所引用的对象中，一**旦我们重新绑定这个对象到GL_WINDOW_TARGET位置，这些选项就会重新生效**。