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/

附讲九　OpenGL编程（二）

数据传输

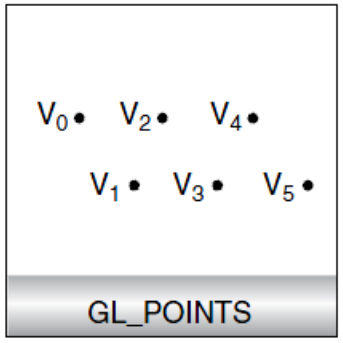
- OpenGL执行模式：分为客户端（client）和服务器端（server），它们可以在同一台计算机上也可以分部在不同计算机上
- 在现代的OpenGL绘制模式下，通常我们需要将客户端内存空间（通常为CPU内存）中的几何或图像数据传输到服务器端内存（通常为GPU内存，即显存）
- 指定几何数据的几种方式
  - glBegin/glEnd…（仅在compatibility profile下）
  - glNewList/glEndList, glCallList…（仅在compatibility profile下）
  - glGenBuffers, glBindBuffer, glBufferData，glDrawArrays, glDrawElements…（core profile）

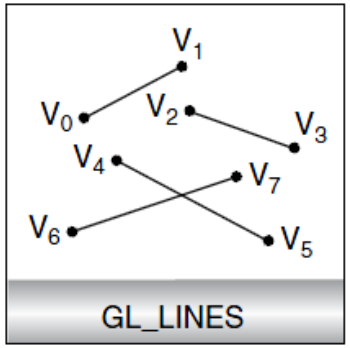
缓冲对象

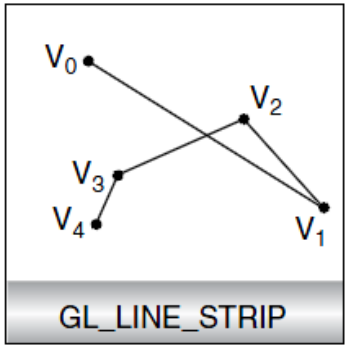
- 对于有独立显卡的现代计算机来说，GPU通常会配置一定量的专用内存（即显存）来执行图形处理任务
  - GPU访问显存的代价远低于访问CPU内存的代价（后者需要通过系统总线访问）
- 顶点缓存对象（Vertex Buffer Objects, VBO）：用于管理在显存中储存的大量顶点数据
  - 可以一次性的发送一大批数据到显卡上，而不是每个顶点发送一次
  - 可以显著提升显卡的处理效率
- 索引缓存对象（Element Buffer Object，EBO，也叫Index Buffer Object，IBO）：用于管理在显存中储存的大量顶点索引数据，常与VBO配合使用

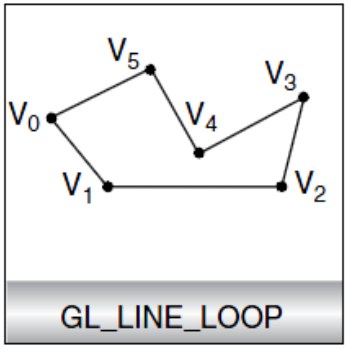
OpenGL图元

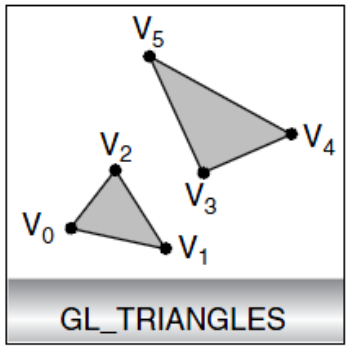
- OpenGL中的基本图元：
  - 点：GL_POINTS
  - 线：GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP
  - 面：GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN
  - 片：GL_PATCHES

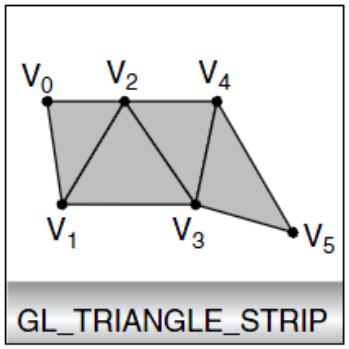


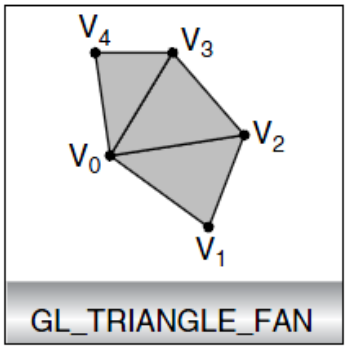






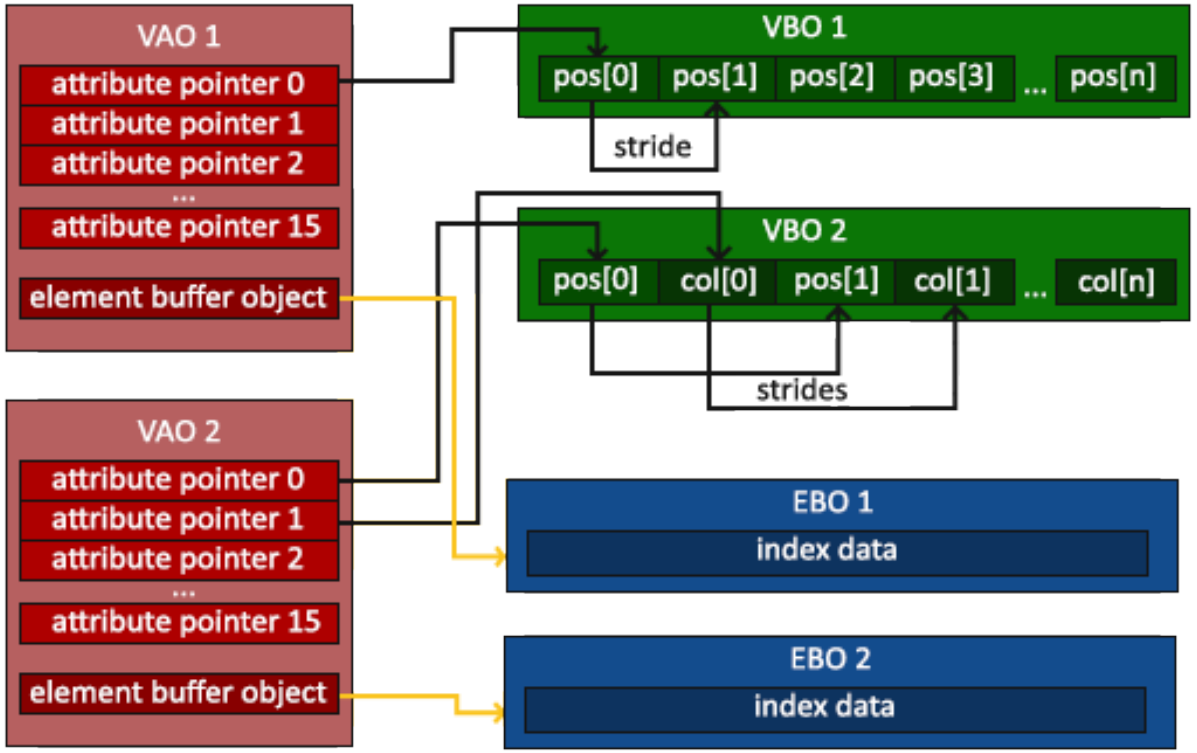






顶点数组对象

- 顶点数组对象（Vertex Array Object, VAO）：管理顶点缓存对象及索引缓存对象
  - 好处：在不同顶点数据和属性配置之间切换变得非常简单，只需要绑定不同的VAO



基本的使用方法

```
// ..:: 初始化代码（只运行一次(除非你的物体频繁改变)） :: ..
// 1. 申请顶点数组对象（VAO），顶点缓存对象（VBO），索引缓存对象（
     EBO）等
   glGenVertexArrays(1, &VAO);
   glGenBuffers(1, &VBO);
   glGenBuffers(1, &EBO);

// 2. 绑定VAO，VBO，EBO
   glBindVertexArray(VAO);

// 3. 把顶点数组复制到缓存中供OpenGL
   glBindBuffer(GL_ARRAY_BUFFER, VBO);
   glBufferData(GL_ARRAY_BUFFER, sizeof(vertices), vertices, GL_STATIC_DRAW);
   glBindButter(GL_ELEMENT_ARRAY_BUFFER, EBO);
   glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(faces), faces,
     GL_STATIC_DRAW);
```

基本的使用方法

```
// 4. 设置顶点属性指针
// core profile: 需要与vertex shader链接
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);
    glEnableVertexAttribArray(0); [...] // ..:: 绘制代码（渲染循环中） :: ..
// 或compatibility profile: （本次作业使用该方式）
    glEnableClientState(GL_VERTEX_ARRAY);
    glVertexPointer(3, GL_FLOAT, stride, 0);
    …
// 4. 绘制物体：使用glDrawArrays, glDrawElements等命令
    glBindVertexArray(VAO);
    glDrawElements(GL_TRIANGLES, 3 * number_of_faces, GL_UNSIGNED_INT, 0);
```

Thanks for your attention!