OpenGL 编程

《OpenGL 入门教程》
https://learnopengl-cn.github.io/01%20Getting%20started/06%20Textures/

OpenGL 编程

几何图形的绘制

点、直线和多边形

- 数学上的点只有位置没有大小，计算机中用一个像素近似
- 数学上的直线没有宽度，无限延长，计算机中的直线对应于线段
- 多边形：多条线段首尾相连而形成的闭合区域。**OpenGL**规定必须为凸多边形，可由其顶点来确定

如何指定顶点？

- OpenGL提供了一系列函数。它们都以glVertex开头，后面跟一个数字和1~2个字母：glVertex2d/glVertex2f/glVertex3f
  - 数字表示参数的个数
  - 字母表示参数的类型

如何指定顶点

前缀	数据类型	相应 C 语言类型	OpenGL 类型
b	8-bit integer	signed char	GLbyte
s	16-bit integer	short	GLshort
i	32-bit integer	long	GLint, GLsizei
f	32-bit floating-point	float	GLfloat, GLclampf
d	64-bit floating-point	double	GLdouble, GLclampd
ub	8-bit unsigned integer	unsigned char	GLubyte, GLboolean
us	16-bit unsigned integer	unsigned short	GLushort
ui	32-bit unsigned integer	unsigned long	GLuint, GLenum, GLbitfield

如何指定顶点

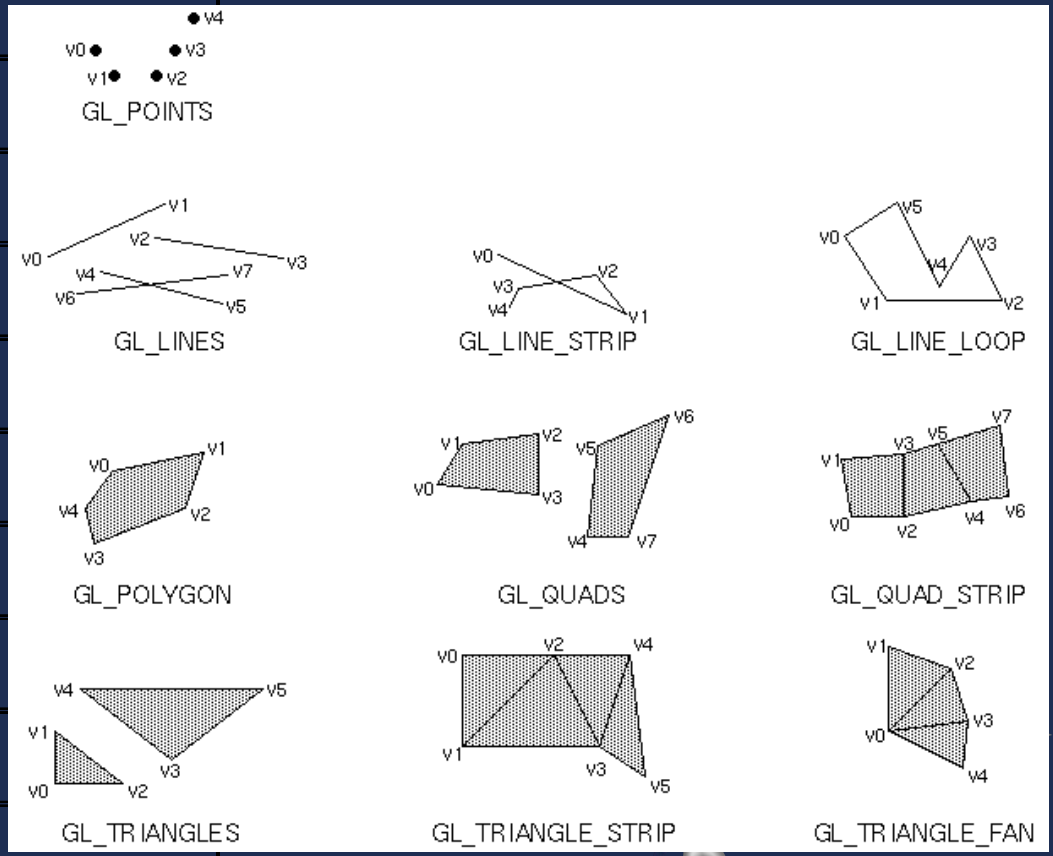
- 这些函数除了参数的类型和个数不同以外，功能是相同的
  - glVertex2i(1, 3);
  - glVertex2f(1.0f, 3.0f);
  - glVertex3f(1.0f, 3.0f, 0.0f);
  - glVertex4f(1.0f, 3.0f, 0.0f, 1.0f);
  - GLfloat VertexArr3[] = {1.0f, 3.0f, 0.0f};
    glVertex3fv(VertexArr3);

开始绘制

- 假设指定了若干顶点，那么**OpenGL**是如何知道我想拿这些顶点来干什么呢？
  - glBegin(GL_POINTS);
    glVertex2f(0.0f, 0.0f);
    glVertex2f(0.5f, 0.0f);
    glEnd();

开始绘制

类型	说明
GL_POINTS	单个顶点集
GL_LINES	多组双顶点线段
GL_POLYGON	单个简单填充凸多边形
GL_TRAINGLES	多组独立填充三角形
GL_QUADS	多组独立填充四边形
GL_LINE_STRIP	不闭合折线
GL_LINE_LOOP	闭合折线
GL_TRAINGLE_STRIP	线型连续填充三角形串
GL_TRAINGLE_FAN	扇形连续填充三角形串
GL_QUAD_STRIP	连续填充四边形串



开始绘制

- 举例方式
  - myDisplay(void)

```
{
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin( /* 在这里填上你所希望的模式 */ );
        /* 在这里使用glVertex*系列函数 */
        /* 指定你所希望的顶点位置 */
    glEnd();
    glFlush();
}
```

开始绘制

- 例1 画一个圆
  - 正四边形，正五边形，正六边形…，直到正n边形，当n越大时，这个图形就越接近圆，当n大到一定程度后，人眼将无法把它跟真正的圆相区别
  - 修改下面的const int n的值，观察当n=3,4,5,8,10,15,20,30,50等不同数值时输出的变化情况
    将GL_POLYGON改为GL_LINE_LOOP、GL_POINTS等其它方式，观察输出的变化情况

点

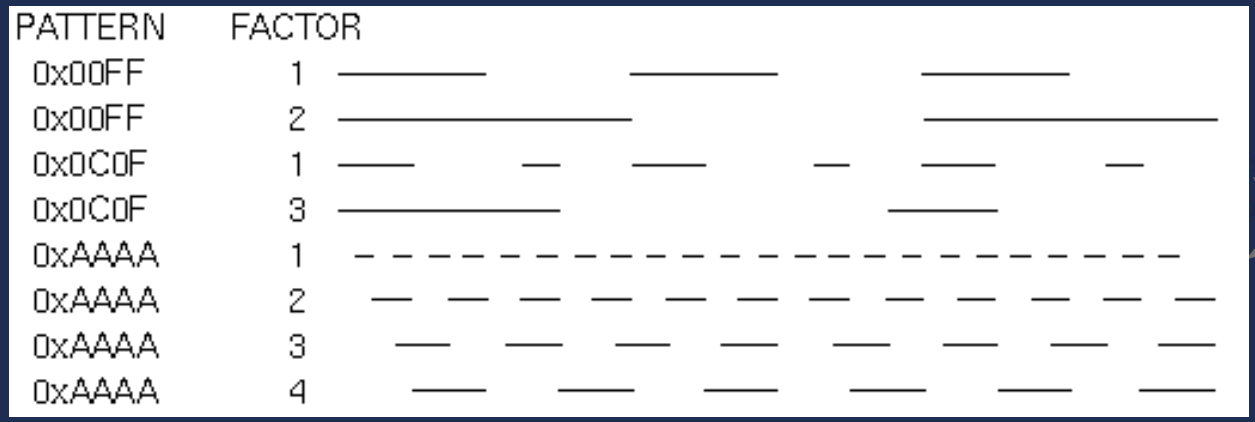
- 关于点
  - 点的大小默认为1个像素，但也可以改变之。改变的命令为glPointSize，其函数原型如下：
    void glPointSize(GLfloat size);
  - size必须大于0.0f，默认值为1.0f，单位为“像素”。
  - 对于具体的OpenGL实现，点的大小都有个限度的，如果设置的size超过最大值，则设置可能会有问题。
  - 举例：画点

直线

- 关于直线
  - 直线可以指定宽度
    void glLineWidth(GLfloat width);
    其用法跟glPointSize类似。
  - 虚线：
    - 使用glEnable(GL_LINE_STIPPLE)来启动虚线模式
    - 使用glLineStipple来设置虚线的样式
      void glLineStipple(GLint factor, GLushort pattern);

直线

- 虚线：
  - pattern是由1和0组成的长度为16的序列，从最低位开始看，如为1则直线上接下来应该画的factor（1-255）个点将被画为实的；如果为0则直线上接下来应该画的factor个点将被画为虚的。
  - 举例：画虚线



多边形

- 关于多边形
  - 从三维的角度来看，一个多边形具有两个面
  - 绘制方式：填充、边缘轮廓线、顶点
  - 可为两个面分别设置不同的方式
    - glPolygonMode(GL_FRONT, GL_FILL);
    - glPolygonMode(GL_BACK, GL_LINE);
    - glPolygonMode(GL_FRONT_AND_BACK, GL_POINT);
  - 正面：顶点以逆时针顺序出现在屏幕上的面
    - glFrontFace函数交换“正面”和“反面”
      - glFrontFace(GL_CCW); // 设置CCW方向为“正面”，CCW即CounterClockWise，逆时针
      - glFrontFace(GL_CW); // 设置CW方向为“正面”
    - 例：将glFrontFace(GL_CCW)改为GL_CW

多边形

- 剔除多边形表面
  - 提高处理图形的效率
    - 多边形虽然有两个面，但无法看见背面
    - 一些多边形虽然是正面的，但被其他多边形所遮挡
    - 无需同等对待无法看见的多边形和可见的多边形
  - 使用glCullFace来进行剔除
    - glCullFace的参数：GL_FRONT，GL_BACK或者GL_FRONT_AND_BACK，分别表示剔除正面、剔除反面、剔除正反两面的多边形
    - 剔除功能只影响多边形，而对点和直线无影响
    - 例：glCullFace(GL_FRONT_AND_BACK)，所有的多边形都将被剔除，所以看见的就只有点和直线

多边形

- 镂空多边形
  - glEnable(GL_POLYGON_STIPPLE)
  - glPolygonStipple设置镂空的样式

    void glPolygonStipple(const GLubyte *mask);
    - 参数mask指向一个长度为128字节的空间，它表示了一个32*32的矩形应该如何镂空
    - 第一个字节表示了最左下方的8个像素是否镂空；最后一个字节表示了最右上方的8个像素是否镂空。
  - 多边形的镂空无法设置factor因子。
  - 请用鼠标改变窗口的大小，观察镂空效果的变化情况
  - 举例：苍蝇图像

小结

- 学习了绘制几何图形的一些细节
  - OpenGL中点、直线和多边形的概念
  - 点可以设置大小
  - 直线可以设置宽度；可以将直线画成虚线
  - 多边形的两个面的绘制方法可以分别设置；在三维空间中，不可见的多边形可以被剔除；可以将填充多边形绘制成镂空的样式

了解这些细节会使我们在一些图象绘制中更加得心应手

OpenGL 编程

颜色模式

颜色模式 ？

- OpenGL支持两种颜色模式:
  - RGBA：数据直接就代表了颜色
  - 颜色索引模式：数据代表的是一个索引，要得到真正的颜色，还必须去查索引表

无论哪种颜色模式，计算机都必须为每一个像素保存一些数据

RGBA模式

- RGBA模式中，每一个像素会保存以下数据：
  - R值（红色分量）
  - G值（绿色分量）
  - B值（蓝色分量）
  - A值（alpha分量）。

  其中红、绿、蓝三种颜色相组合可得到所需要的各种颜色，而alpha不直接影响颜色

RGBA模式

- RGBA模式颜色的选择：
  - glColor*系列函数
    - 三个参数的版本可以指定R、G、B的值，而A值采用默认

      void glColor3f(GLfloat red, GLfloat green, GLfloat blue);
    - 四个参数的版本可以分别指定R、G、B、A的值

      void glColor4f(GLfloat red, GLfloat green, GLfloat blue, GLfloat alpha);

RGBA模式

- glColor*系列函数
  - 浮点数作为参数，其中0.0表示不使用该种颜色，而1.0表示将该种颜色用到最多。
  - 例如：glColor3f(1.0f, 0.0f, 0.0f); 最纯净的红色。
    glColor3f(0.0f, 1.0f, 1.0f); 表示使用绿、蓝色到最多，混合的效果就是浅蓝色。
    glColor3f(0.5f, 0.5f, 0.5f); 表示各种颜色使用一半，效果为灰色。

浮点数可以精确到小数点后若干位，这并不表示计算机就可以显示如此多种颜色。

RGBA模式

- glColor*系列函数
  - glColor系列函数，在参数类型不同时，表示“最大”颜色的值也不同。
  - 用f和d做后缀的函数，以1.0表示最大的使用。
  - 用b做后缀的函数，以127表示最大的使用。
  - 用ub做后缀的函数，以255表示最大的使用。
  - 用s做后缀的函数，以32767表示最大的使用。
  - 用us做后缀的函数，以65535表示最大的使用。

索引颜色

- 在该模式中，OpenGL需要一个颜色表
  - 调色板：颜色表每一项等价于调色板的格子，虽可调出多种颜色，种数将不会超过格数
  - 颜色表大小：256~4096之间，2的整数次幂
  - 在使用索引颜色方式进行绘图时，总是先设置颜色表，然后选择颜色
- 选择颜色
  - 用glIndex*系列函数可在颜色表中选择颜色
  - 最常用的可能是glIndexi，参数是个整形

```
void glIndexi(GLint c);
```

索引颜色

- 设置颜色表（举例）
  - OpenGL并未直接提供设置颜色表的方法，因此设置颜色表需要使用操作系统的支持
  - Windows和其他大多数图形操作系统都具有这个功能，但所使用的函数却不相同
  - GLUT工具包提供了设置颜色表的函数glutSetColor
  - OpenGL工具包：auxSetOneColori，VS自带不必另外安装　#include<GL/glaux.h>

索引颜色

- 优点
  - 占用空间小，花费系统资源少， 图形运算速度快
  - 小型设备例如GBA、手机等，有用武之地
- 缺点
  - 编程稍稍显得不是那么方便
  - 画面效果也会比RGB颜色差一些

指定清除屏幕用的颜色

- 指定清除屏幕用的颜色
  - 把屏幕上的颜色清空:

    glClear(GL_COLOR_BUFFER_BIT)
  - 什么才叫“空”：OpenGL用下面的函数来定义清楚屏幕后屏幕所拥有的颜色
    - RGB模式下，使用glClearColor来指定“空”的颜色，它需要四个参数
    - 索引颜色模式下，使用glClearIndex来指定“空”的颜色所在的索引，它需要一个参数

指定着色模型

- 指定着色模型：
  - 为顶点指定颜色：系统计算其它点的颜色，使相邻的点的颜色值都比较接近
    - RGB模式：看起来就具有渐变的效果
    - 颜色索引模式：相邻点的索引值是接近的
  - glShadeModel函数可以关闭这种计算
    - 直线以后指定的点的颜色为准
    - 多边形将以任意顶点颜色为准，由实现决定
  - **glShadeModel的使用方法**
    - glShadeModel(GL_SMOOTH);　// 平滑方式
    - glShadeModel(GL_FLAT);　// 单色方式

小结

- 如何设置颜色
  - RGB颜色方式是目前PC机上的常用方式
  - 索引颜色在小型设备中有用武之地
- 可设置glClear清除后屏幕所剩的颜色
- 可以设置颜色填充方式：平滑方式或单色方式