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/

第八章 离散技术

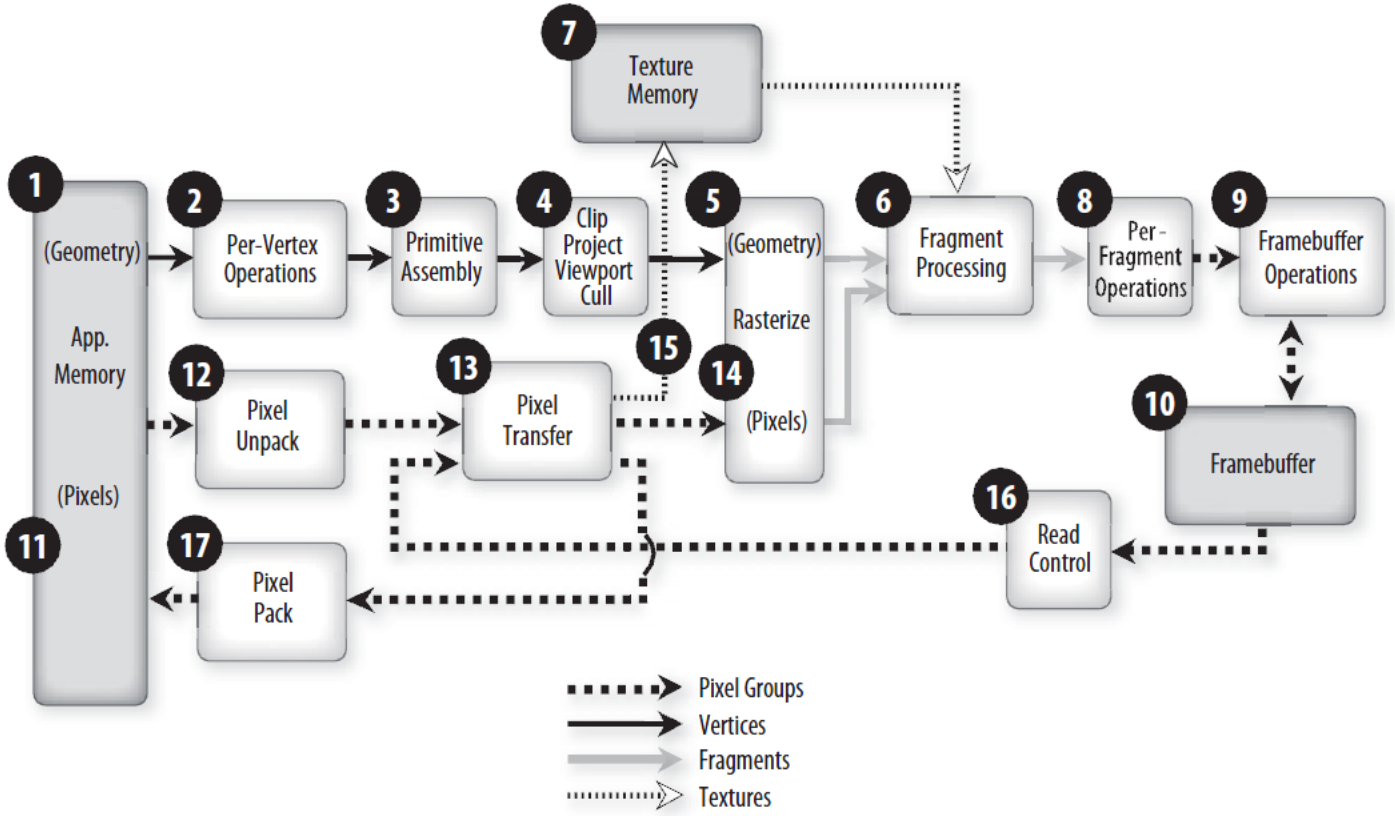
第一节 缓冲区

光栅图形

- 计算机图形学的基本任务：输入三维场景，输出一幅图像
- 在计算机图形学出现后的很长一段时间内人们处理的主要是几何对象
  - 例如：点、线、多边形
- 但是，输出的是一幅图像，因此需要必要的像素操作
- 在新近发展起来的绘制技术中，需要直接与构成帧缓冲区中的各种缓冲区打交道
  - 例如：纹理映射、反走样、组合、融合

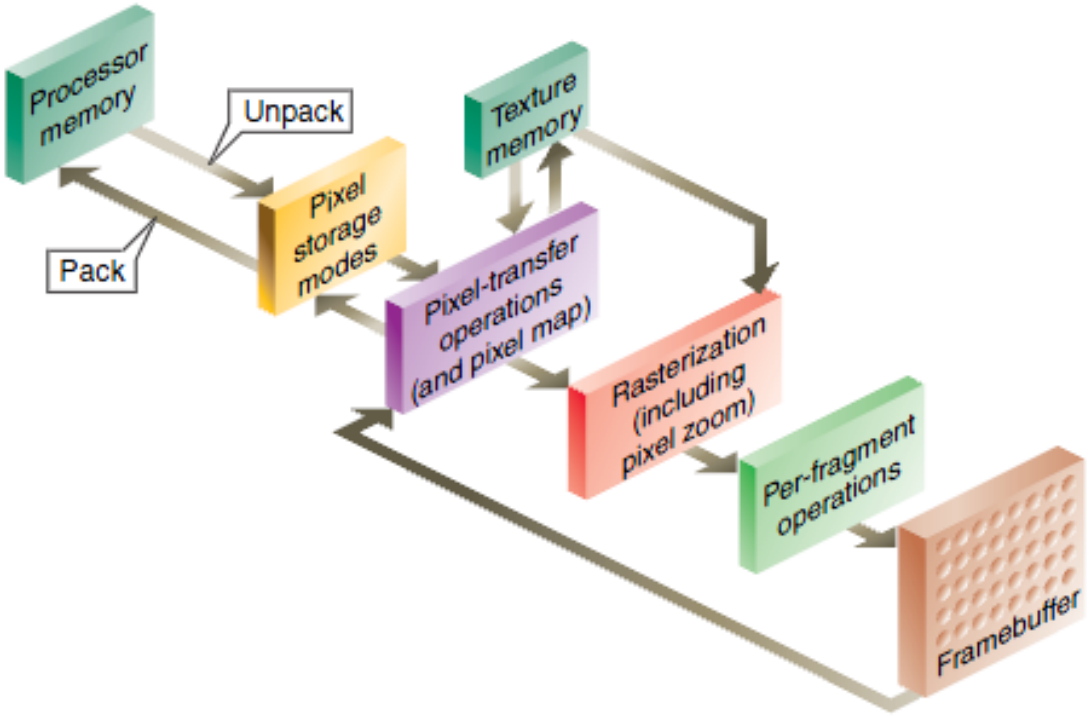
OpenGL流水线

- OpenGL的流水线
  - 几何流水线
  - 像素流水线



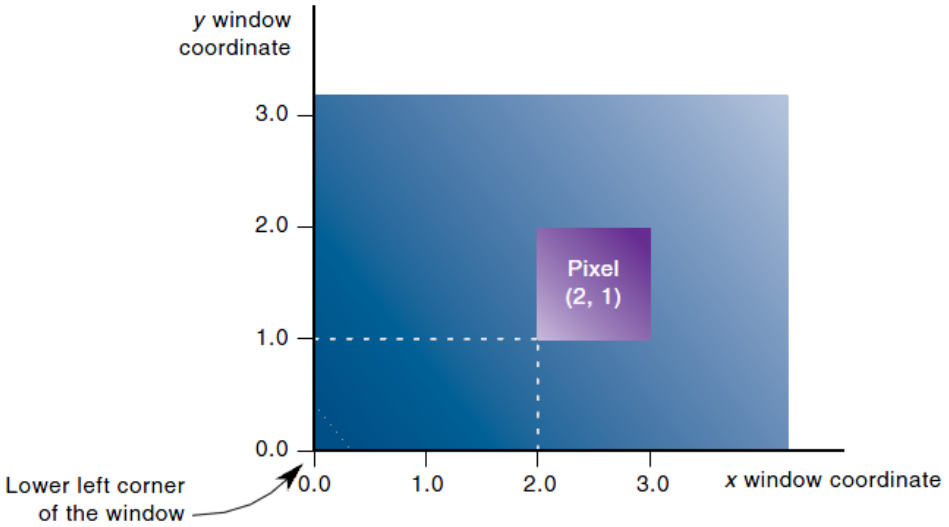
像素流水线

- OpenGL的像素流水线
  - Pack/Unpack：包装/解包
  - Framebuffer：帧缓存



缓存区

- 缓存（buffer）：用于像素信息的存储空间称为缓存，通常为矩形区域
- 位平面（bitplane）：由存储每个像素的单个位信息构成的缓存称为位平面；一个缓存中位平面的个数称为缓存的深度（depth）
- 缓冲区都是离散的
  - 由其空间分辨率（n x m）和深度（或精度，每个像素的位数）k确定

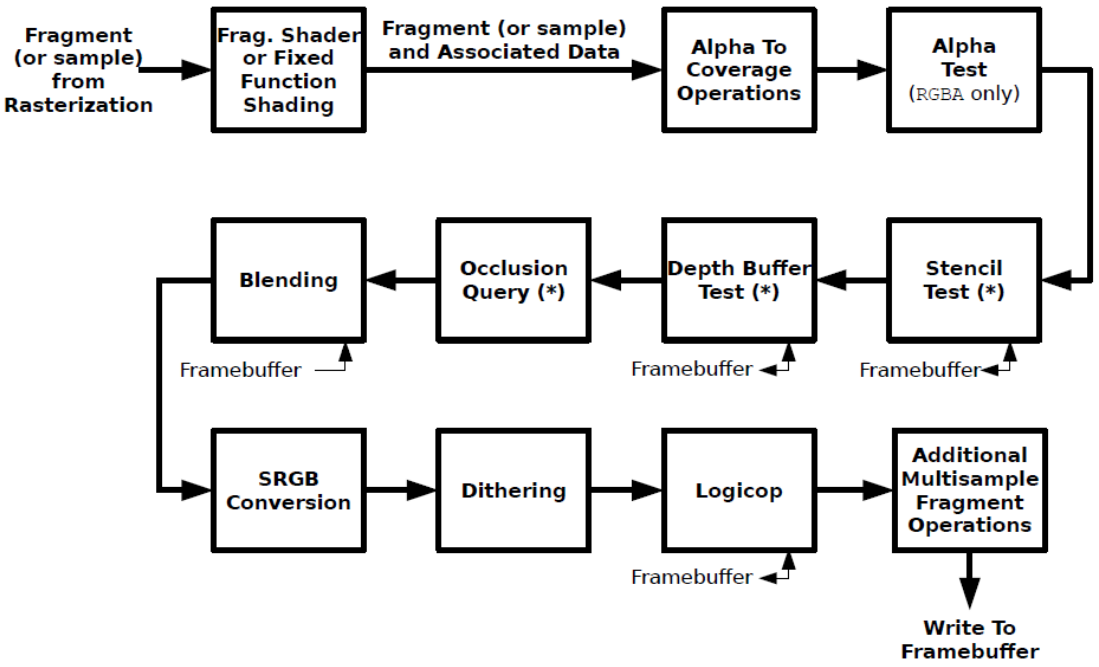


帧缓存

- 帧缓存（framebuffer）：在OpenGL中，帧缓存指显卡中用于存储OpenGL绘制结果的特定内存区域
- 缺省的帧缓存（default framebuffer）通常包括以下几种类型的缓存：
  - 一个或者多个颜色缓存（color buffer）
  - 深度缓存（depth buffer）
  - 模板缓存（stencil buffer）
  - 多重采样缓存（multisample buffer）
- 双缓存技术（double buffering）：前缓存＋后缓存，常用于消除画面的闪烁现象
  - 原理：应用程序将绘制结果存储到后缓存中，而显示器显示前缓存中的内容，通过一个swap 操作切换前后缓存
  - 原因：有时绘制一副场景的时间远远多于显示器的刷新率，这时就会出现画面闪烁的现象

逐片元操作

- 在OpenGL中，每个片元经过片元着色器处理后，还会执行以下一些操作
  - 剪切测试（scissor test）
  - 多重采样片元操作（multisample fragment operations）
  - 模板测试（stencil test）
  - 深度测试（depth test）
  - 融混（blending）
  - 抖动（dithering）
  - 逻辑操作



逐片元操作的应用

- 深度缓存
  - 隐藏面消除（z buffer algorithm）
  - 阴影绘制（shadow mapping）
- 模板缓存
  - 汽车驾驶模拟显示中的挡风玻璃
  - 多道绘制技术（multipass rendering）
- 融混
  - 透明物体的绘制
- 多重采样与抖动
  - 反走样
- 利用逐片元操作，可以实现很多特殊任务的绘制

帧缓存对象

- OpengGL中有两大类帧缓存（framebuffer）
  - 缺省的帧缓存：窗口系统提供的缓存，也是唯一可以被图形服务器的显示系统所识别的帧缓存（即屏幕上看到的只能是这个缓存），大小、分辨率、图像格式受窗口系统的制约
  - 应用程序创建的帧缓存：帧缓存对象，离线渲染（很多高级渲染效果都是通过多帧缓存实现的）
- **应用程序创建的渲染缓存（renderbuffer，可被绑定到帧缓存对象）可以保存以下信息**
  - 一个或多个颜色缓存：GL_COLOR_ATTACHMENTi，i=0…GL_MAX_COLOR_ATTACHMENTS -1
  - 深度缓存：GL_DEPTH_ATTACHMENT
  - 模板缓存：GL_STENCIL_ATTACHMENT
  - 深度/模板缓存：GL_DEPTH_STENCIL_ATTACHMENT（注：如果需要同时使用深度，模板缓存，必须使用该格式）

缺省的帧缓存

- 颜色缓存用于显示
  - 前/后：双缓存
  - 左/右：立体显示
- 深度缓存
- 模版缓存
  - 保存掩码（masks）

颜色缓冲区

- 内容要被输出到显示设备上的缓冲区
- 在双缓存（第三章）中，颜色缓冲区由两个缓冲区组成，分别用于读与写，称为前缓冲区与后缓冲区
- 如果要生成立体图像，需要提供左、右缓冲区

缓存的深度k

- 颜色缓冲区K确定可以表示的颜色多少
  - 通常在RGB模式中为24位，在RGBA模式中为32位
- 深度缓冲区的k值确定深度的分辨率
  - 通常是32位，这样与浮点数或整数的精度匹配
- 把缓冲区中k个平面称为位平面（bitplane），空间中特定位置处的k个元素构成一个像素（pixel）
  - 因此一个像素既可以是一个字节，也可以是一个整数，甚至是一个浮点数，具体与所用的缓冲区以及信息的存放格式有关

实现细节

- 应用程序通常并不知道在帧缓冲区中各种信息的存放方式
  - 帧缓冲区是在API的内部实现的，对用户而言它是一个黑盒子
- 应用程序是通过API提供的函数向帧缓冲区发送（写入）信息，或者接受（读出）信息
  - 此时在通常的内存与实现的缓冲区间传送的数据需要经过一定的格式转化
  - 这时需要仔细考虑数据传送的时间效率
  - Pack/unpack：需要设置数据转换的格式

数字图像

- 有时简称为图像
  - 此时它与把几何对象和照相机组合在一起，通过某种投影过程所得到的结果（有时也称为图像）不同
- 在程序中的图像是像素数组
  - 其大小与类型各异，具体与图像的类型有关
  - 例如，对于RGB图像，通过有一个字节表示一个颜色成分，因此可以如下定义512×512图像

    ```
    GLubyte myimage[512][512][3];
    ```

  - 如果处理的是黑白图像或者强度图像，那么可以定义为

    ```
    GLubyte myimage[512][512];
    ```

创建图像的一种方法

- 创建图像的一种方法就是利用程序中的代码
- 例如：为了创建512×512图像，由8×8红黑相间方格构成，那么代码如下

```
GLubyte check[512][512][3];
int i,j,k;
for(i=0;i<512;i++) for(j=0;j<512;j++){
  for(k=0;k<3;k++) check[i][j][k]=0;
  if((8*(i+j)/64)%64) check[i][j][0]=255;
}
```

其它方法

- 应用计算机生成图像只能得到有规则模式的结果
- 通过直接从数据获取图像
  - 例如从实验中获取了一组浮点数，那么可以把它们放缩到0到255区间中，从而形成一幅强度图像
- 创建图像的第三种方法就是应用光学成像设备
  - 数码相机与摄像机

图像的格式

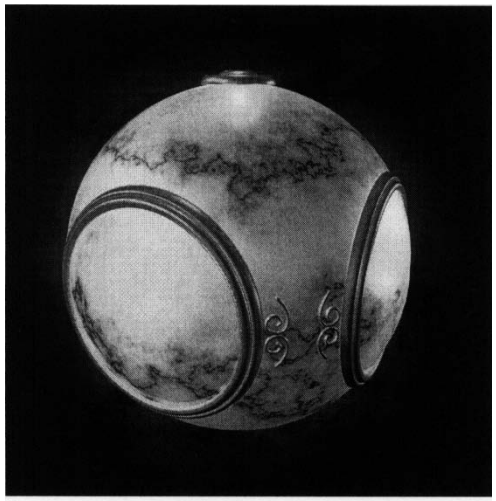
- 从数码设备采集的图像是以某种“标准”的格式存贮的
- 最常用格式
  - JPEG, PNG, BMP, GIF, TIFF, PS以及EPS等
  - 以某种顺序存贮数据，并进行必要的无损压缩或者有损编码
  - 都是为了满足某种需要而提出的

JPEG

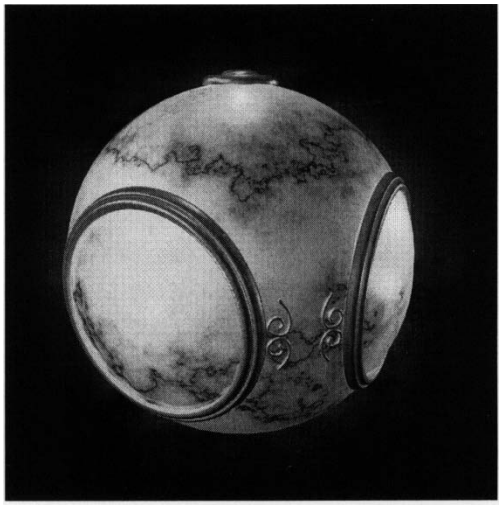
- JPEG图像采用一个算法进行压缩，应用压缩后数据恢复原始图像存在一定的误差
- 因此JPEG图像具有很高的压缩比（compression ratio）
  - 在原始图像中的字节数与压缩后数据中的字节数之比
- 参考文献

  http://www.jpeg.org/committee.html

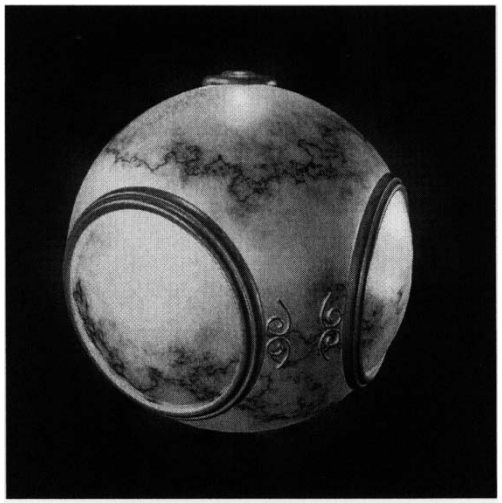
JPEG压缩前后



原始图像



压缩比18



压缩比37

GIF

- CompuServe Incorporated 提出
  - Graphics Interchange Format
- 颜色索引图像
- 在图像中需要存贮一个颜色表以及索引数组
- 参考文献
  http://astronomy.swin.edu.au/~pbourke/dataformats/gif/

PNG

- 便携式网络图形是一种无损压缩的位图图形格式，支持索引、灰度、RGB三种颜色方案以及Alpha通道等特性
- PNG的开发目标：改善并取代GIF作为适合网络传输的格式而不需专利许可，所以被广泛应用于互联网及其他方面上

BMP

- BMP（全称Bitmap）是Windows操作系统中的标准图像文件格式，可以分成两类：设备有向量相关位图（DDB）和设备无向量相关位图（DIB）
- 存储格式：图像的扫描方式是按从左到右、从下到上的顺序
- 开发者：微软公司
- 优点：相对简单
- 缺点：不采用其他任何压缩，文件所占用的空间很大

TIFF

- 标签图像文件格式（Tag Image File Format，简写为TIFF）是一种灵活的位图格式，主要用来存储包括照片和艺术图在内的图像
- 开发者：Adobe公司
- 有两种形式
  - 第一种形式：所有图像数据直接编码
  - 第二种形式：数据被压缩
    - 因此许多数据包含多余的信息，例如图像中很大的区域在颜色或强度上有相同的变化不大
    - 这种格式应用Lempel-Ziv算法达到最佳的压缩效果，恢复过程中无误差
- 参考文献

http://www.ee.cooper.edu/courses/course_pages/past_courses/EE458/TIFF/

PS与EPS

- PostScript（PS）图像是PostScript语言的一部分，用来控制打印机
  - 把RGB或者强度图像精确编码到7位ASCII字符集中
  - 因此PS图像可以被很多打印机以及其它设备所识别，但通常结果很大
- Encapsulated PostScript（EPS）类似于PS，但是提供其它信息以预览图像
- 参考书：Thinking in PostScript
  - 从http://www.rightbrain.com/pages/books.html下载

尺寸比较

- 原始图像：1200x1200
- TIFF：1440198
  - 每像素一个字节，同时198字节存贮前言以及后续信息
- JPEG：80109与38962
- EPS：约是TIFF的两倍
- 压缩型TIFF： 尺寸约减少一半
- ZIP压缩：TIFF与EPS大约结果相同

格式与OpenGL

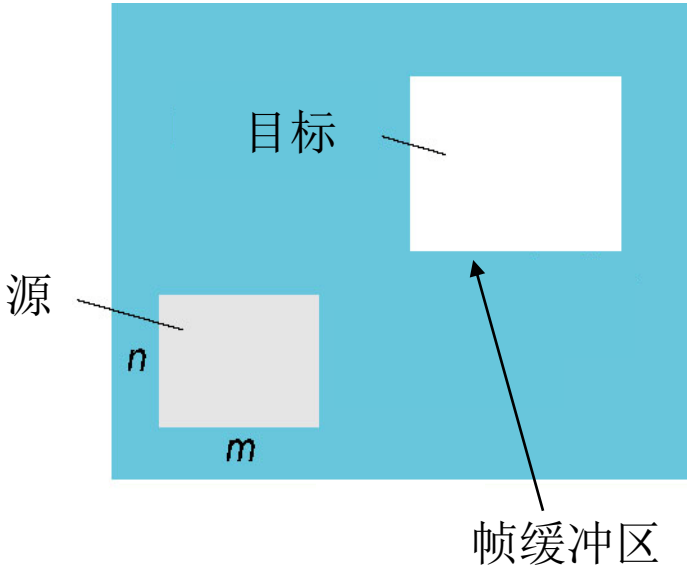
- 为了从某种格式中解析出来像素数组，需要知道格式的定义，这种过程有时是非常复杂的
- OpenGL中不包含任何解析已有图像格式的函数与功能
- 在OpenGL中的图像就是内存中的存贮标准数据类型的数组
- 建议：使用第三方库加载图像，譬如stb_image.h

帧缓存区的位块操作

- 在现代的计算机图形系统中，用户程序既可以向缓冲区中写入内容，也可以从中读出内容
- 下述因素使得这种操作与通常的读写内存操作不同
  - 很少情形下只想读写一个像素，而是读写一个矩形的像素块（位块，bit blocks）
  - 当进行清除操作时，改变缓冲区中所有像素的值
- 需要在硬件和软件方面提供对位块进行尽可能有效操作的功能
  - 称为位块传递（bit-block transfer, bitblt）操作

位块复制

- 假设要把源缓冲区中的一块n×m像素复制到目标缓冲区中，那么进行这种操作的位块传递函数应当具有形式
- `void glBlitFramebuffer(int srcX0, int srcY0, int srcX1, int srcY1, int dstX0, int dstY0, stX1, int dstY1, bitfield mask, enum filter);`
- 类似的还有：`glBlitNamedFramebuffer, glCopyImageSubData`

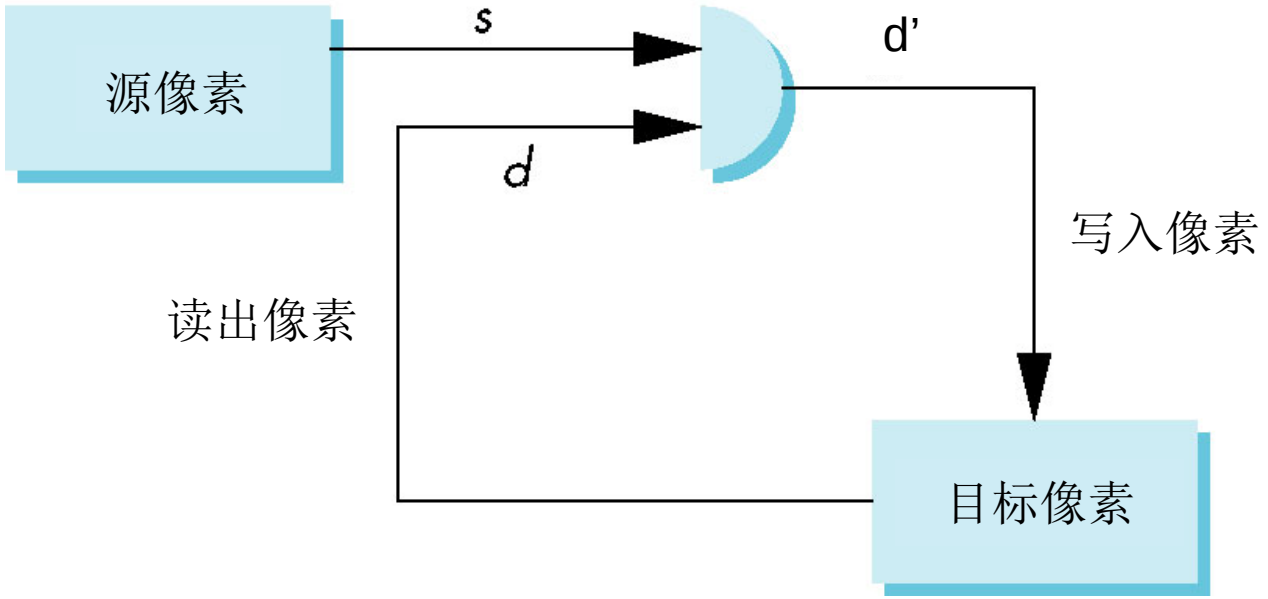


实现细节

- 此时有许多细节需要注意，例如：源块超过了目标缓冲区边界时该如何处理？
- 该操作的本质在于单个函数调用改变了整个目标块
- 从硬件的角度来看，这种处理与几何对象的处理没有任何共同点
  - 从而优化位块传递操作的硬件具有与几何操作的流水线硬件完全不同的框架
  - 在OpenGL中包含两个流水线体系

帧缓存写入操作的模型

- 在写入源像素之前会读出目标缓冲区中的像素，这也是读写内存与位块传递操作不同的另一个地方



像素逻辑运算

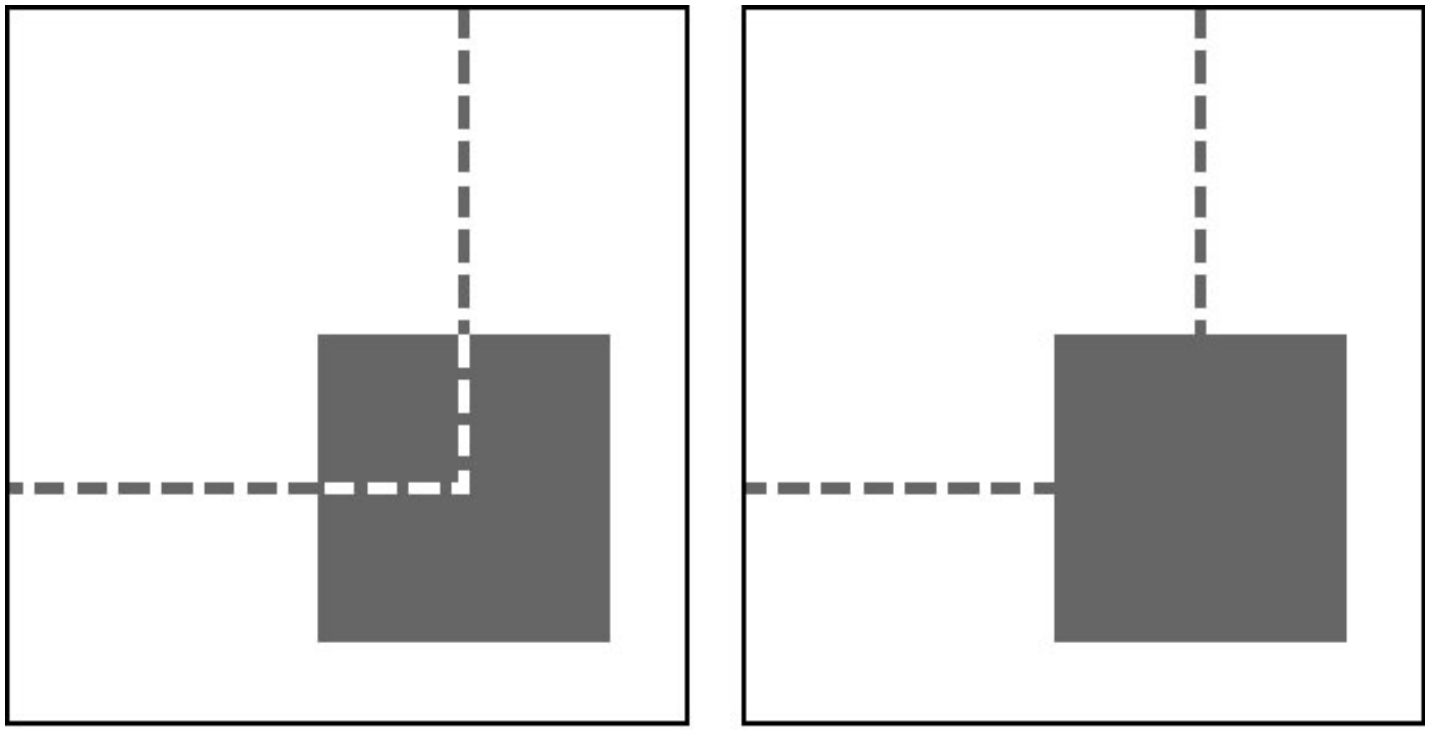
- 源像素：s　目标像素：d
- 写入的目标像素：d'
- 那么d' = f(s, d)
  - 其中s = 0 or 1, d = 0 or 1
  - 因此有2^4 = 16种定义方式

16种逻辑运算

- 源像素与目标像素逐位结合在一起
- 有16种可选的函数（表格中的每列为一种）

s	d	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
					取代				XOR	OR							

模式3与6



模式6　　模式3

XOR

- 应用XOR运算实行了橡皮带式绘图方式
- XOR模式在要把菜单从屏幕上移除，进行内存块切换时非常有用

如果S表示屏幕像素，M表示菜单的像素，那么

$S \leftarrow S \oplus M \quad M \leftarrow S \oplus M \quad S \leftarrow S \oplus M$

就会交换S与M的内容

这样当菜单从屏幕上消失后，就会自动恢复原来屏幕上的内容

图像的像素位数

- 假设数字图像是由k位像素组成，这里k可以是1（二进制图像，位图bitmap），…，32（RGBA图像）或者更大的数字（更高分辨率的图像）
- 虽然单位图像与8位（1字节）或24位（3字节）图像在原理上没有任何区别，但硬件和软件上通常把它们区别处理
  - 从硬件的角度来说，处理位图时只需要实现逻辑运算。在缓冲区中同一时间在一个位面工作
  - 从软件的角度来说，位图与多字节图像的应用范围完全不同。通常位图用来表示字体、掩码以及模式

位块操作

- Bitblt操作对于位图才有意义
  - 字体、光标
- 对于多字节图像，这些操作的含义不是很明确
  - 此时16种操作的结果通常不对应于有意义的结果

结构支持

- OpenGL实现了单独的像素流水线
- 提供了一组缓冲区
  - 数据可以在这些缓冲区中传递，也可以在缓冲区与处理器内存间传递
  - 根据所用缓冲区的不同，可以在其中存贮颜色索引、颜色甚至深度

缓冲区的选择

- OpenGL可以向任何缓冲区中写入内容，或者从中读取内容
- 默认的是后缓冲区
- 可以用glDrawBuffer和glReadBuffer改变
- 注意在帧缓冲区中像素的格式与它在处理器的内存中的格式是不同的
  - 需要组装(pack)与展开(unpack)
  - 绘制与读取可能会很慢

像素与图像

- 在OpenGL中我们可以处理各种类型的图像，像素可以用各种格式处理为8位的组
  - 对于RGB图像：每个颜色分量对应于一位
  - 灰度图像：每个像素为一个浮点数
- 在OpenGL中像素操作的函数
  - 读取像素：`glReadPixels/glReadnPixels`
  - 拷贝像素：`glBlitFramebuffer/glBlitNamedFramebuffer`
  - 拷贝像素：`glCopyImageSubData`

Thanks for your attention!