河北师范大学软件学院
Software College of Hebei Normal University

Android源生AR应用开发

第四讲 3D图形绘制

智能设备教研室

目录

1 **3D 绘图基本概念**

2 3D 坐标变换

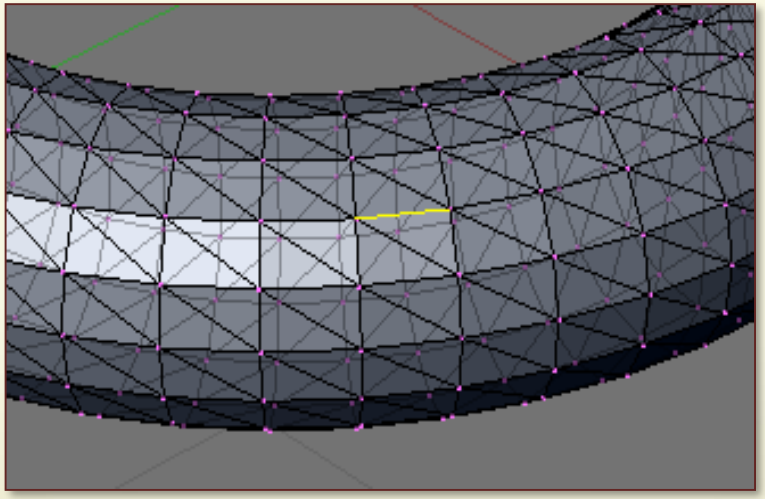
3 ModelView变换

4 投影变换

5 视口变换

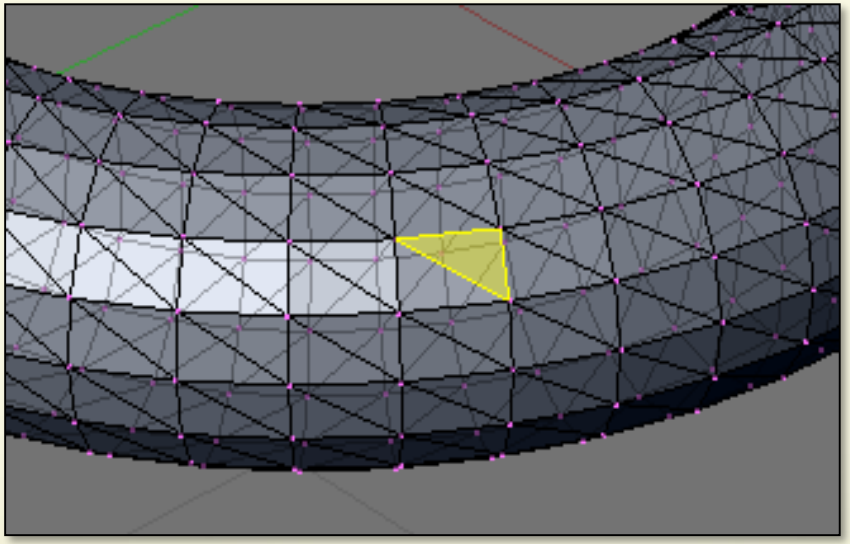
3D 绘图基本概念

- 边(Edge)定义为两个顶点之间的线段在OpenGL中，通常无需直接定义一个边，而是通过顶点定义一个面，从而由面定义了其所对应的三条边。可以通过修改边的两个顶点来更改一条边。



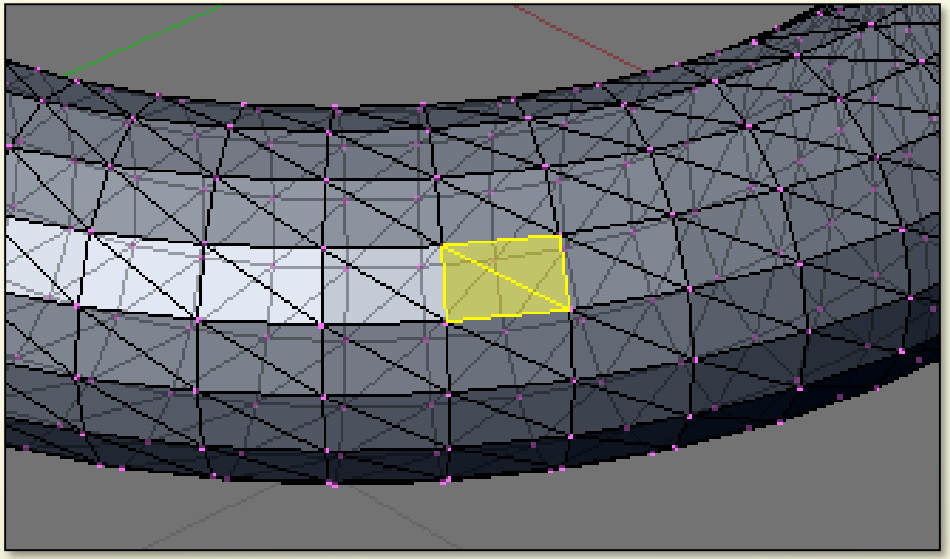
3D 绘图基本概念

- 面(Face)指一个三角形，由三个顶点和三条边构成，对一个面所做的变化影响到连接面的所有顶点和边、面和多边形。



3D 绘图基本概念

- 多边形(Polygon)由多个面(三角形)拼接而成，在3D空间中，多边形不一定在同一个平面上。



3D 绘图基本概念

- 在拼接曲面的时候，用来定义面的顶点的顺序非常重要，因为顶点的顺序定义了面的朝向（前向或是后向），为了获取绘制的高性能，一般情况不会绘制面的前面和后面，只绘制面的“前面”。虽然“前面”“后面”的定义可以应人而易，但一般为所有的“前面”定义统一的顶点顺序(顺时针或是逆时针方向）。

目录

1 3D 绘图基本概念

2 **3D 坐标变换**

3 ModelView变换

4 投影变换

5 视口变换

3D 坐标变换

- OpenGL ES图形库最终的结果是在二维平面上显示3D物体（常称作模型Model）这是因为目前的大部分显示器还只能显示二维图形。但我们在构造3D模型时必须要有空间现象能力，所有对模型的描述还是使用三维坐标。也就是使用3D建模，而由OpenGL ES库来完成从3D模型到二维屏幕上的显示。

3D 坐标变换

❖这个过程可以分成三个部分：

① 坐标变换，坐标变换通过使用变换矩阵来描述，因此学习3D绘图需要了解一些空间几何，矩阵运算的知识。三维坐标通常使用齐次坐标来定义。变换矩阵操作可以分为视角（Viewing），模型（Modeling）和投影（Projection）操作，这些操作可以有选择，平移，缩放，正交投影，透视投影等。

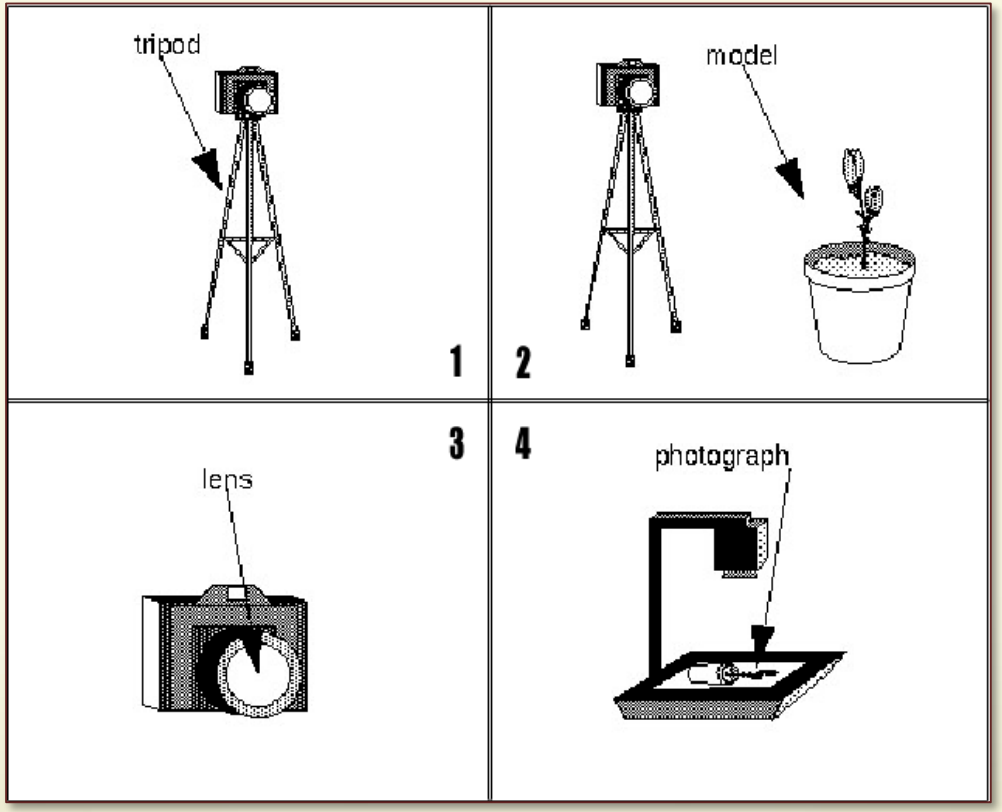
3D 坐标变换

② 由于最终的3D模型需要在一个矩形窗口中显示，因此在这个窗口之外的部分需要裁剪掉以提高绘图效率，对应3D图形，裁剪是将处在剪切面之外的部分扔掉。

③ 在最终绘制到显示器（2D屏幕），需要建立起变换后的坐标和屏幕像素之间的对应关系，这通常称为“视窗” 坐标变换(Viewport Transformation)。

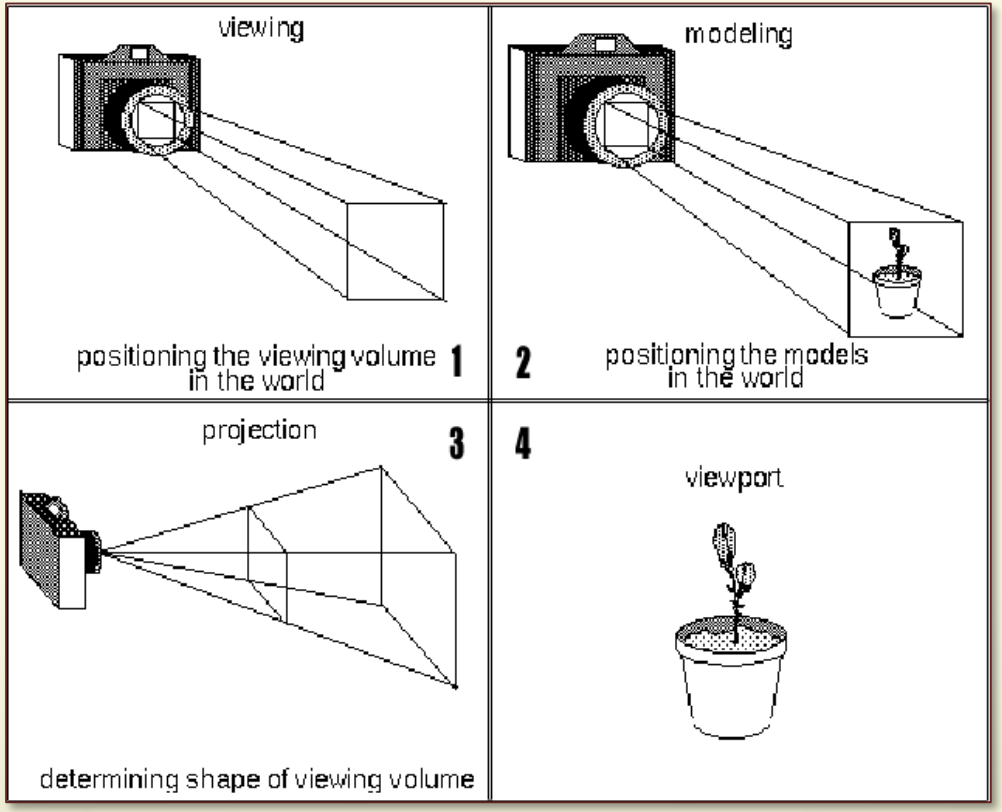
3D 坐标变换

❖使用照相机拍照的过程



3D 坐标变换

- OpenGL ES 3D 图形

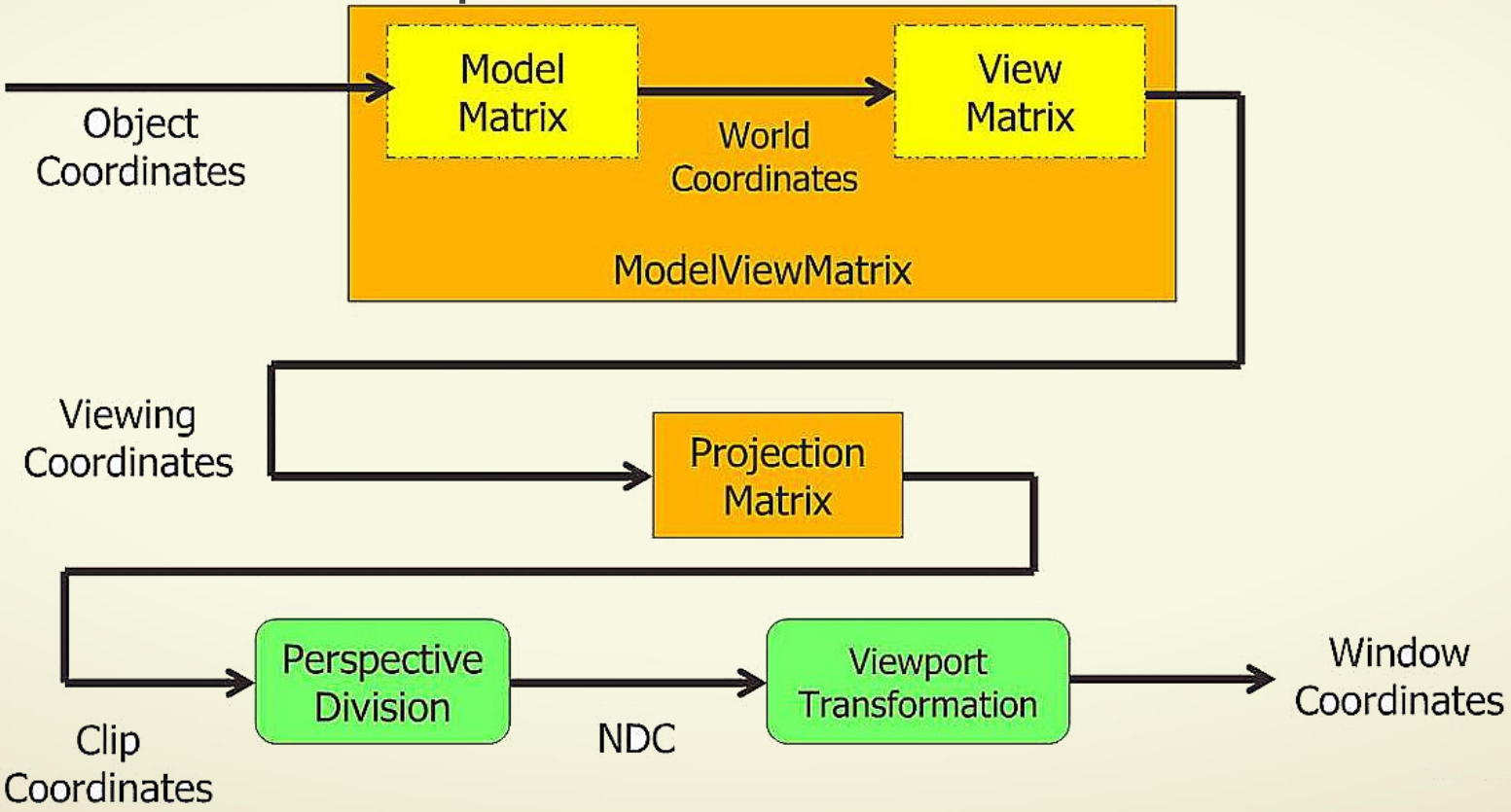


3D 坐标变换

- OpenGL有6种坐标系，分别如下：
  - 物体或模型坐标系(Object or Model Coordinates)
  - 世界坐标系（World Coordinates）
  - 眼坐标或相机坐标（Eye (or Camera) Coordinates)
  - 裁剪坐标系（Clip Coordinates）
  - 标准设备坐标系（Normalized Device Coordinates）
  - 屏幕坐标系（Window (or Screen) Coordinates）
- 除了上面6种外，OpenGL还存在一种假想坐标系纹理坐标系，这个坐标系是不存在的，它其实是一系列变换矩阵的结果，比如它能使顶点从物体或模型坐标系变换到世界坐标系。

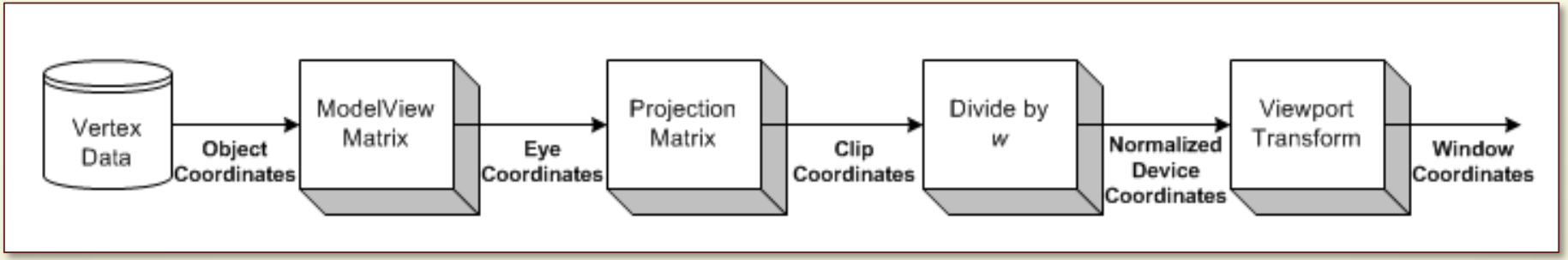
3D 坐标变换

❖下图为Android OpenGL ES坐标变换的过程：



3D 坐标变换

- 坐标变换矩阵栈，用来存储一系列的变换矩阵，栈顶就是当前坐标的变换矩阵，进入OpenGL管道的每个坐标(齐次坐标)都会先乘上这个矩阵，结果才是对应点在场景中的世界坐标。OpenGL中的坐标变换都是通过矩阵运算完成的。



- ModelViewMatrix：模型矩阵
- ProjectionMatrix　：投影矩阵

目录

1 3D 绘图基本概念

2 3D 坐标变换

3 **ModelView变换**

4 投影变换

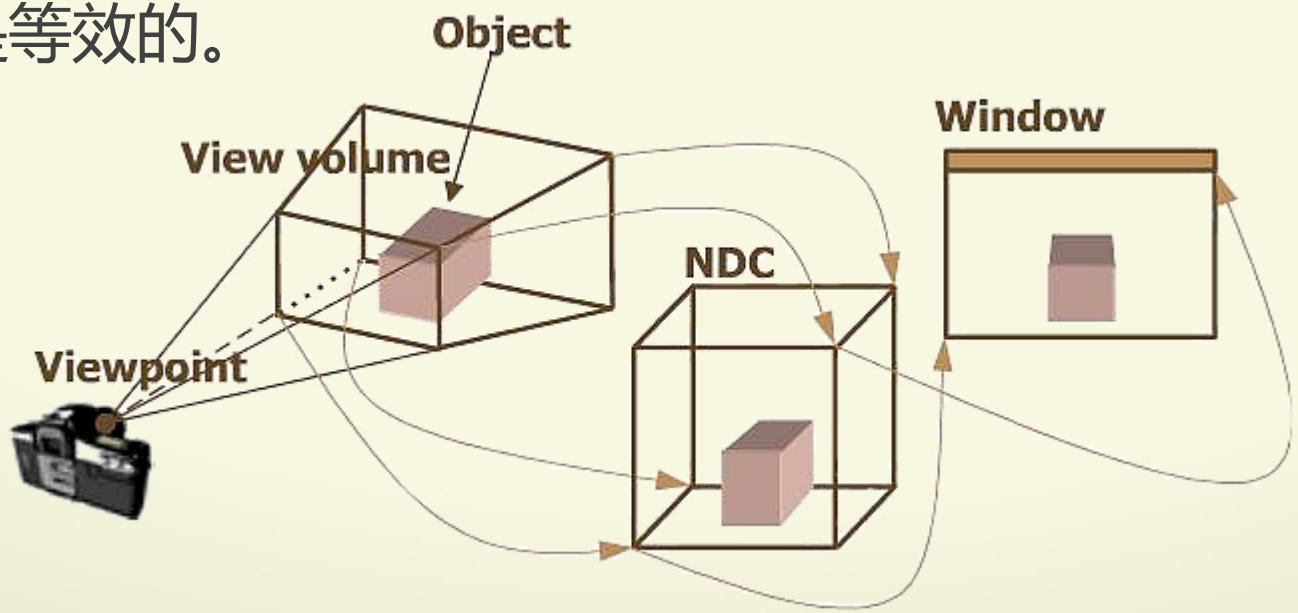
5 视口变换

Viewing和Modeling(ModelView)变换

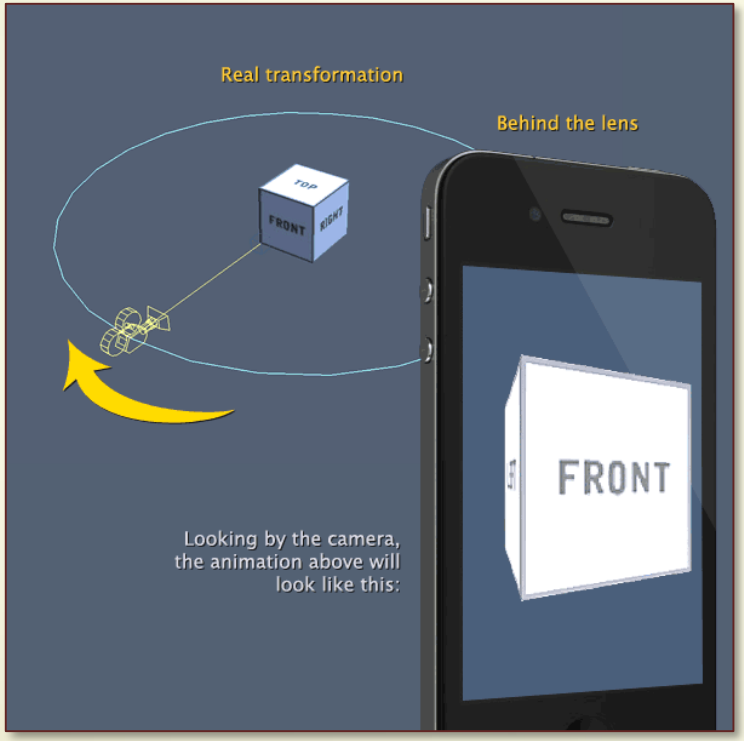
- Viewing和Modeling 变换关系紧密，对应到相机拍照为放置三角架和调整被拍物体位置及角度，通常将这两个变换使用一个ModelView 变换矩阵来定义。
- 对于同一个坐标变换，可以使用不同的方法来想象这个变换，比如将相机向某个方向平移一段距离，效果等同于将被拍摄的模型(model)向相反的方向平移同样的距离（相对运动）。两个不同的空间想象方法对于理解坐标变换各有其优缺点。你可以使用适合自己理解能力的方法来想象空间坐标变换。

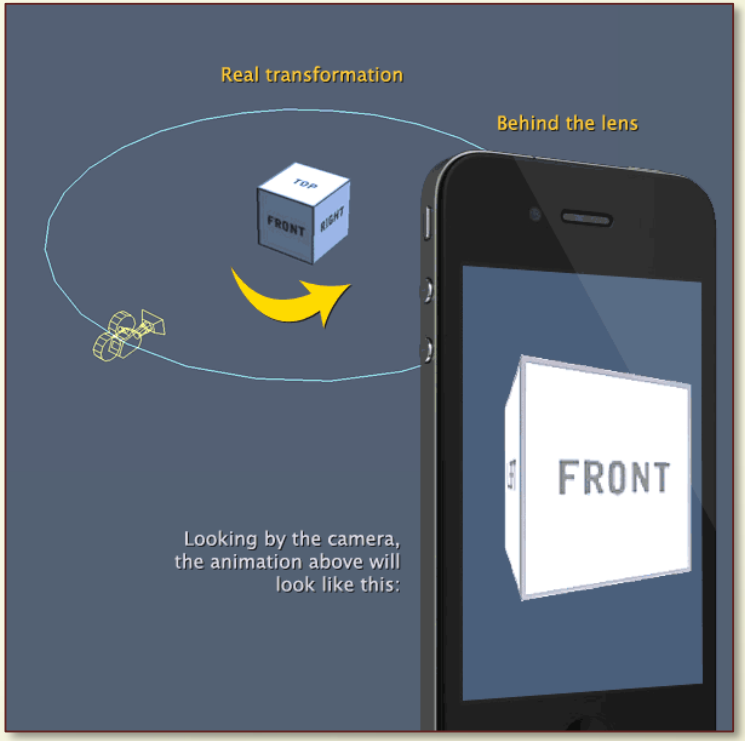
Viewing和Modeling(ModelView)变换

- 对于Viewing transformation (平移，选择相机）和Modeling transformation（平移，选择模型）可以合并起来看，只是因为向左移动相机，和相机不同将模型右移的效果是等效的。



Viewing和Modeling(ModelView)变换





目录

1 3D 绘图基本概念

2 3D 坐标变换

3 ModelView变换

4 **投影变换**

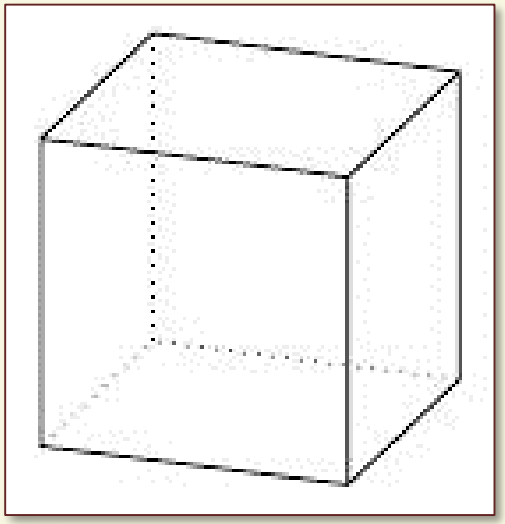
5 视口变换

投影变换

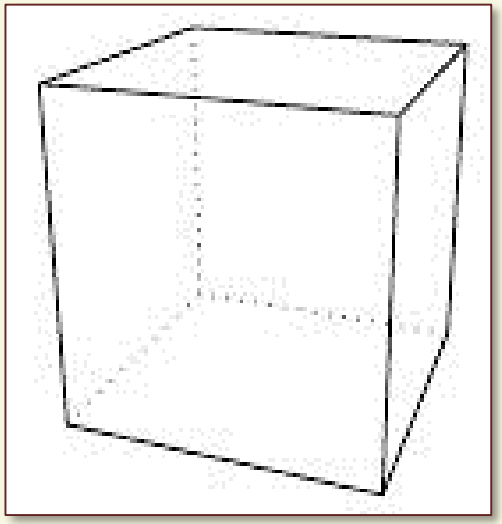
- 前面ModelView变换相当于拍照时放置相机和调整被拍物体的位置和角度。投影变换则对应于调整相机镜头远近来取景。投影变换的目的是确定 3D 空间的物体如何投影到 2D 平面上，从而形成2D图像，这些 2D 图像再经视口变换就被渲染到屏幕上。
- 投影变换有两种：正交投影（Orthographic Projection）和透视投影（Perspective Projection）。透视投影用的比较广泛，它与真实世界更相近：近处的物体看起来要比远处的物体大；而正交投影没有这个效果，正交投影通常用于CAD或建筑设计。

投影变换

❖下面是正交投影与透视投影效果示意图：



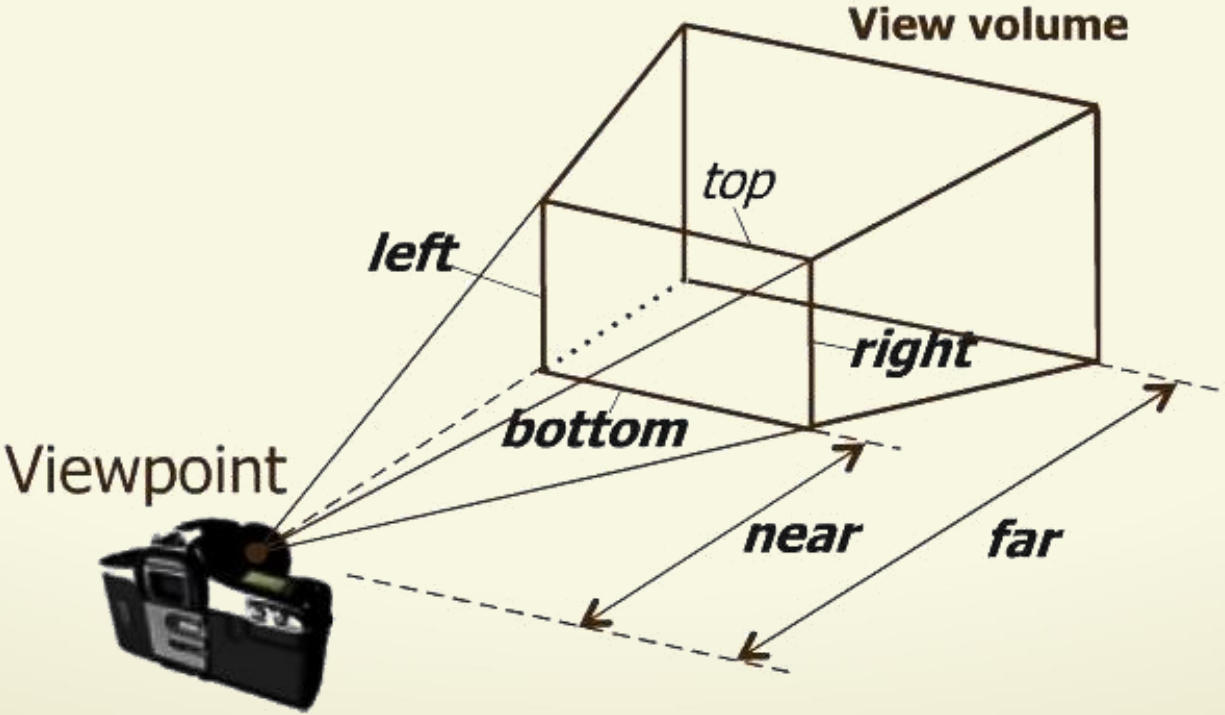
正交投影



透视投影

投影变换 – 透视投影

- 透视投影的特点是“近大远小”，也就是我们眼睛日常看到的世界。



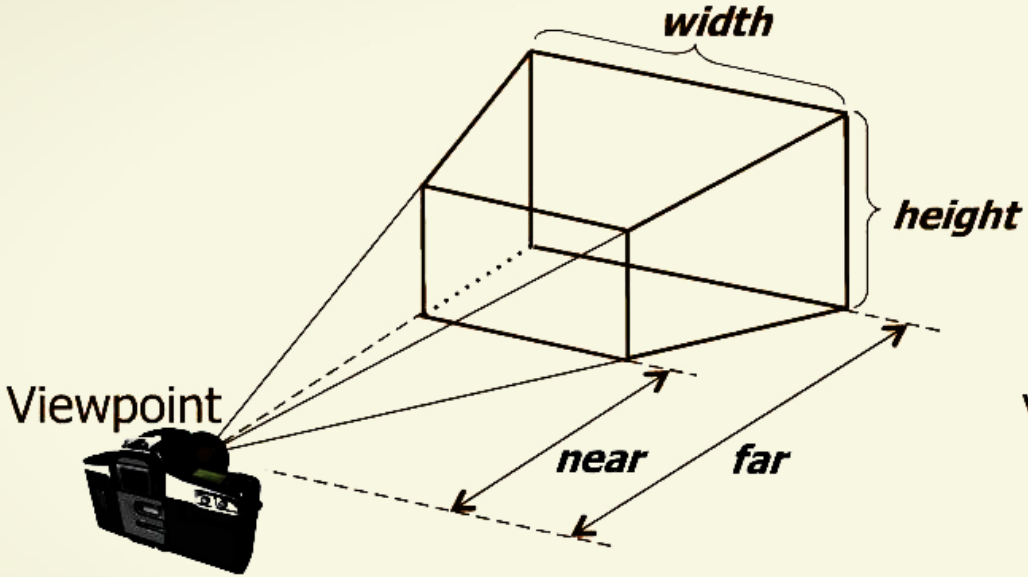
投影变换 – 透视投影

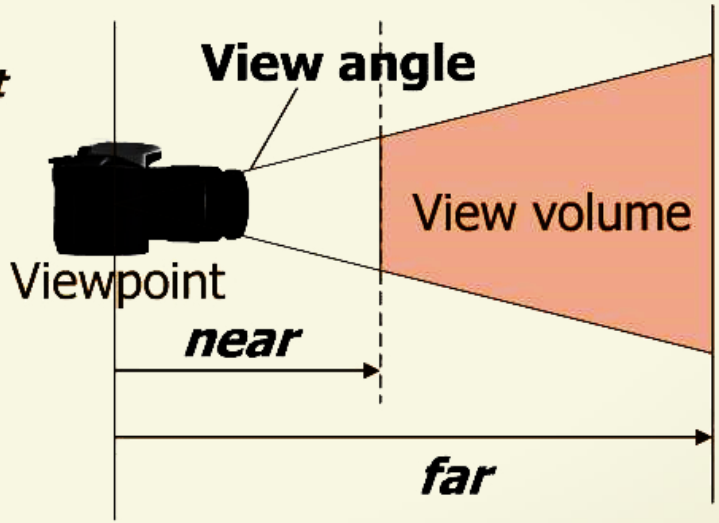
❖OpenGL ES 2.0定义透视投影的函数为frustumM ():

```
Matrix.frustumM(float[] m,    int offset,
                float left,   float right,
                float bottom, float top,
                float near,   float far);
```

- m　　：存储生成矩阵元素的float[]类型数组。
- offset：定义视锥的宽高比。
- left、right　：near面的left、right。
- botto、top　：near面的bottom、top。
- near、far　　：near面、far面与视点的距离。

投影变换 – 透视投影

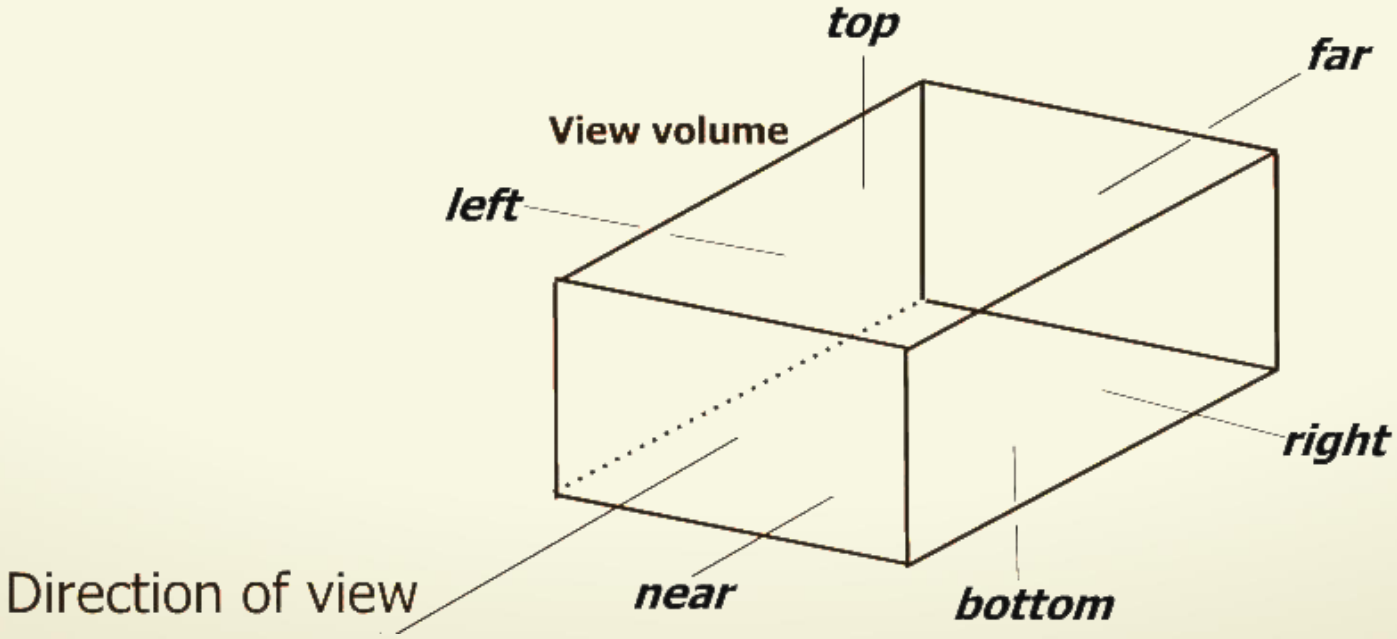




$$aspect = width / height$$

投影变换 – 正交投影

- 正交投影，它的视锥为一长方体，特点是物体的大小不随到观测点的距离而变化，投影后可以保持物体之间的距离和夹角。它主要用在工程制图上。



投影变换 - 正交投影

❖定义正交投影（也称作平移投影）的函数为：

```
Matrix.orthoM(float[] m,    int offset,
              float left,   float right,
              float bottom, float top,
              float near,   float far);
```

- m　　：存储生成矩阵元素的float[]类型数组。
- offset：定义视锥的宽高比。
- left、right　：near面的left、right。
- botto、top　：near面的bottom、top。
- near、far　　：near面、far面与视点的距离。

目录

1 3D 绘图基本概念

2 3D 坐标变换

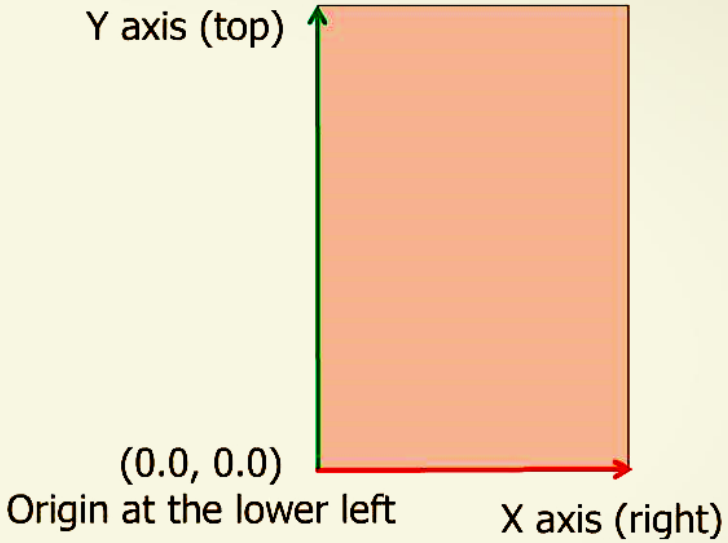
3 ModelView变换

4 投影变换

5 **视口变换**

视口变换

- 摄影师调整好相机和被拍摄物体的位置角度（ModelView），对好焦距（Projection)后，就可以按下快门拍照了，拍好的照片可以在计算机上使用照片浏览器查看照片，放大，缩小，拉伸，并可以将照片显示窗口在屏幕上任意拖放。



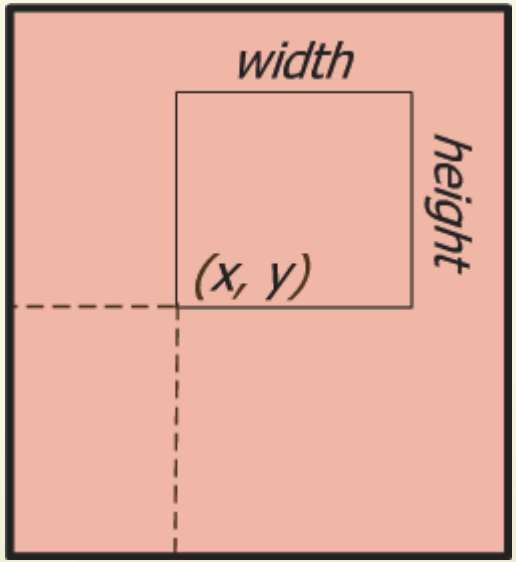
- 对应到3D绘制就是Viewport 变换，目前的显示器大多还是2D的，Viewport（显示区域）为一个长方形区域，并且使用屏幕坐标系来定义。

视口变换

- OpenGL ES 中使用glViewport() 来定义显示视窗的大小和位置：

```
glViewport(int x, int y, int width, int height)
```

- Android 缺省将viewport 设置成和显示屏幕大小一致。



视口变换

- 如果投影变换的宽度/高度比 (aspect) 和最后的Viewport的width/height 比不一致的话，最后显示的图形就可能需要拉伸以适应Viewport,从而可能造成图像变形。比如：现在的电视的显示模式有4:3 和 16:9 或是其它模式，如果使用16:9的模式来显示原始宽高比为4：3的视频，图像就有变形。

Aspect ratio of view volume

Aspect ratio of viewport


THANK YOU