


青春都一饷，忍把浮名，换了代码轻狂。

关注DirectX

随笔 - 198, 文章 - 65, 评论 - 1663, 引用 - 0

#### 导航

博客园  
首 页  
新随笔  
联 系  
订 阅   
管 理

< 2012年7月 >						
日	一	二	三	四	五	六
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

#### 公告

昵称：zdd  
园龄：7年1个月  
荣誉：推荐博客  
粉丝：687  
关注：20  
+加关注

#### 搜索

#### 常用链接

我的随笔  
我的评论  
我的参与  
最新评论  
我的标签

#### 随笔分类

Android(12)  
C/C++(19)  
D3D10  
D3D11(1)  
D3D9(4)  
Demo(3)  
Direct2D(14)  
DirectInput(1)  
DirectWrite(3)  
DirectX(58)  
DXGI  
HLSL  
IOS(2)  
Math(16)  
OpenGL(1)  
Perl(1)  
PS(1)  
Shader(2)  
Swift  
Unity3D  
Vim  
XAudio2  
编程语言(9)  
代码片段(19)  
数据结构与算法(26)  
图形学(10)  
移动开发(11)  
杂(24)

#### 随笔档案

2016年6月 (1)  
2016年1月 (4)  
2015年12月 (1)  
2015年11月 (1)  
2015年10月 (2)  
2015年8月 (5)  
2015年7月 (1)  
2014年11月 (2)  
2014年6月 (2)  
2014年5月 (1)  
2013年9月 (1)  
2013年7月 (1)

## View Transform (视图变换) 详解

### 什么是View Transform

我们可以用照相机的原理来阐释3D图形的绘制过程，想象一下，我们在摄影的时候都需要做哪些工作，大致可分为如下几个步骤

1. 摆放好待拍摄的物品，或者人物。
2. 调整好拍摄角度。
3. 调整焦距。
4. 拍摄。

好了，来分析一下，上面的第一步就相当于世界变换了，将一个模型置于一个公认的坐标系中，这里所谓的公认，也就是大家都遵守的，目的是保证待拍摄的物体和照相机在同一个坐标系。第二步相当于视图变换，这个过程是调整Camera到合适的位置以便拍摄，在3D程序中，也就是设置View Matrix了。第三步调整焦距，这就相当于3D编程中的投影变换。

View Transform的过程就是在世界坐标系中摆放Camera的过程，并将顶点由世界坐标系转换到Camera Space，在Camera Space中，观察者（Camera）位于坐标原点，观察方向指向Z轴正方向。

### 为什么要进行View Transform?

在world space中，camera并不一定位于坐标原点，并且观察方向不一定指向Z轴正方向，对于投影变换及其他的一些操作来说，如果不满足这两个条件，后续的操作就会变得非常低效，所以为了提高效率，我们需要进行view transform。

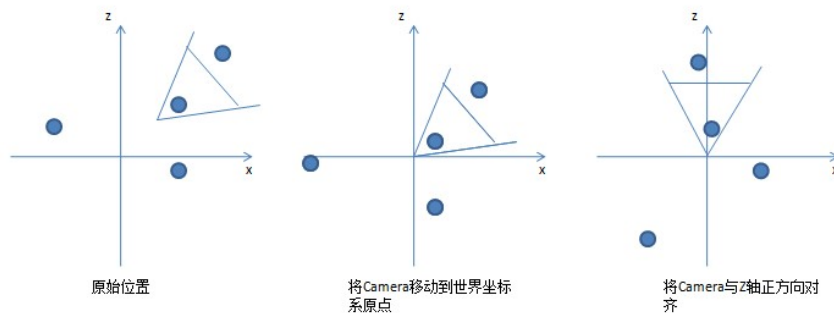
### View Transform的作用

View Transform主要有下面两个作用。

- 移动camera，使其位于world space的坐标原点
- 旋转camera，使其朝向z轴正方向，也就是视线由原点指向z轴正方向。

这两个过程，前一个实际上是平移，后一个实际上是旋转。你可以想象成Camera也有三个坐标轴x, y, z，视图变换的过程就是将Camera的坐标轴与世界坐标系的坐标轴对齐的过程。

注意：view transform中，所有位于world space中的models都随着camera一起变换，所以视野并未发生变化，具体过程见下图



2013年6月 (1)  
2013年4月 (2)  
2013年3月 (3)  
2013年2月 (3)  
2012年11月 (2)  
2012年10月 (2)  
2012年9月 (5)  
2012年8月 (6)  
2012年7月 (5)  
2012年6月 (2)  
2012年5月 (2)  
2012年4月 (1)  
2011年9月 (2)  
2011年7月 (3)  
2011年6月 (2)  
2011年5月 (4)  
2011年4月 (3)  
2011年3月 (3)  
2011年2月 (4)  
2010年12月 (3)  
2010年11月 (3)  
2010年9月 (1)  
2010年8月 (11)  
2010年7月 (17)  
2010年6月 (9)  
2010年5月 (12)  
2010年4月 (2)  
2010年3月 (16)  
2010年2月 (3)  
2010年1月 (2)  
2009年12月 (2)  
2009年11月 (9)  
2009年10月 (11)  
2009年9月 (3)  
2009年8月 (1)  
2009年7月 (4)  
2009年6月 (10)  
2009年5月 (2)

Game Engine

DX11 Tutorials  
Irrlicht  
MathWords  
Ogre

OpenGL

缤纷世界

3D Controls  
3D Fractal  
chaos files  
ChaosinChinese  
Cloth Simulation  
DirectX Developer Center  
DirectX document online  
DX tutorials  
EuclideanSpace  
Fractal Video  
geometrictools  
Google C++ Style  
HardCode  
HyperGraph  
In Framez  
LatexEditor  
Mame  
Mandelbrot Set  
OpenGL official page  
OpenGL Tutorials  
pouet  
Ray tracing  
Rthdribl  
TechInterview  
toymaker  
W3SCHOOL  
XNA/DirectX Forum  
X-Zone  
云风的Blog

其他

Aogo汇编小站  
Channel 9  
Code all in one  
Emath  
Math Circle  
Math Game  
Microsoft At Home  
Microsoft At Work

如何求解View Matrix?

使用D3D函数

使用下面的D3D函数可以计算view matrix, 第一个参数是输出参数, 返回求得的视图矩阵, 第二个参数是眼睛的位置, 第三个参数是观察点中心, 最后一个参数是向上向量。该函数的最后两个字母表示左手系 (Left Hand)。

D3DXMatrixLookAtLH(&M, &eyePt, &lookCenter, &upVec) ;

手动求解

手动求解View matrix并不是难事, 一个camera一般有如下四个属性,

- 向前向量 ( direction ) , 相当于Z轴
- 向上向量 ( up vector ) , 相当于Y轴
- 向右向量 ( right vector ) , 相当于X轴
- 位置 ( position )

其中前三个向量要求是相互垂直的。假设我们分别用d, u, v和p来表示这四个变量。并假设待求的视图矩阵为V, 根据前面的介绍我们知道, V的作用就是将摄像机移动到原点, 并将摄像机的三个向量分别与坐标轴对齐, d与z轴正方向对齐, u与y轴正方向对齐, r与x轴正方向对齐。假设将摄像机与坐标轴对齐的矩阵为V, 那么V的推导过程如下。

$$rV = (1,0,0)$$

$$uV = (0,1,0)$$

$$dV = (0,0,1)$$

假设矩阵 V 是如下形式

$$\begin{matrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{matrix}$$

$$\begin{matrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{matrix}$$

$$\begin{matrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{matrix}$$

那么就有

$$\begin{bmatrix} r \\ d \\ u \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

即

$$\begin{bmatrix} r_x & r_y & r_z \\ d_x & d_y & d_z \\ u_x & u_y & u_z \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{设 } A = \begin{bmatrix} r_x & r_y & r_z \\ d_x & d_y & d_z \\ u_x & u_y & u_z \end{bmatrix}, B = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}$$

即

$$AB = I$$

所以 B = A<sup>-1</sup>, 又因为 A 是正交矩阵, 所以 B = A<sup>T</sup>

即

$$B = \begin{bmatrix} r_x & u_x & d_x \\ r_y & u_y & d_y \\ r_z & u_z & d_z \end{bmatrix}$$

友情链接

LittleStart  
WW老弟

积分与排名

积分 - 202944  
排名 - 781

最新评论

1. Re:Direct2D教程 ( 十一 ) 几何变换  
你好, 请问怎么样可以把图片 ( 或者画刷 ) 弄成镜像对称的 ? 旋转180度就上下颠倒了。

--kina1234

2. Re:字符串面试题 ( 一 ) 字符串逆序

shoucangshou收藏

--cnb\_yangwei

3. Re:GLFW初体验  
楼主, 我想咨询下一glfw 上面的 glDrawPixels 函数, 在mac上渲染出来只有1/4的内容呢我用glfw创建了一个200 \* 200的窗口。然后声明了一个 uchar[200 \* 20.....

--程序员小U

4. Re:DirectX Effects初探  
初学龙书6.8 effects。

--hplucky

5. Re:C++ 初始化列表  
31楼说的对。  
我就觉得这里有问题, 自己也试了, 确实不行。看评论找了半天, 看到有一样的了。  
楼主抓紧重新编辑下啊。

--peiruibo

阅读排行榜

1. 点到平面的距离公式 (43332)  
2. 算法-求二进制数中1的个数(43076)  
3. C++ 初始化列表 (42090)  
4. 字符串面试题 ( 一 ) 字符串逆序(39843)  
5. 关于数组的几道面试题 (36980)

评论排行榜

1. 程序员, 请昂起你高贵的头 ! (364)  
2. 关于数组的几道面试题 (96)  
3. 几何变换详解(76)  
4. 判断点是否在三角形内 (62)  
5. 使用DirectX截屏(55)

推荐排行榜

1. 程序员, 请昂起你高贵的头 ! (176)  
2. 算法-求二进制数中1的个数(28)  
3. C++ 初始化列表(25)  
4. 字符串面试题 ( 一 ) 字符串逆序(20)  
5. 关于数组的几道面试题 (17)

假设将摄像机平移至原点的矩阵为

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -p_x & -p_y & -p_z & 1 \end{bmatrix}$$

将T和B结合起来形成最终的视图矩阵M

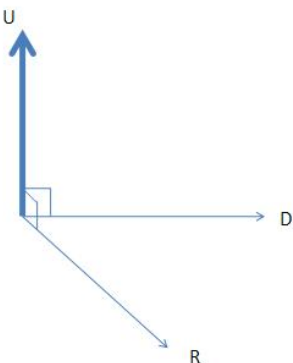
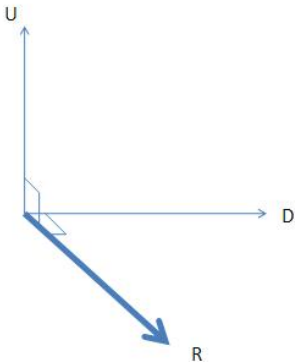
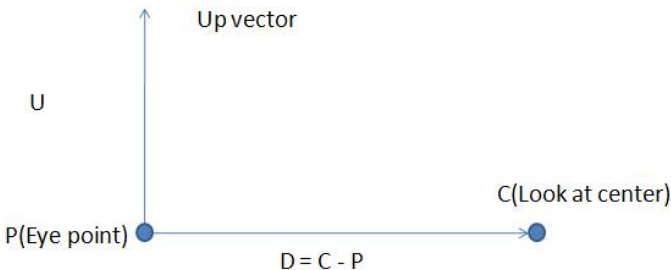
$$TB = M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -p_x & -p_y & -p_z & 1 \end{bmatrix} \begin{bmatrix} r_x & u_x & d_x & 0 \\ r_y & u_y & d_y & 0 \\ r_z & u_z & d_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_x & u_x & d_x & 0 \\ r_y & u_y & d_y & 0 \\ r_z & u_z & d_z & 0 \\ -p \cdot r & -p \cdot u & -p \cdot d & 1 \end{bmatrix}$$

通常在实际的编程中, 只会给出如下三个量。至于d和r这两个量, 只能通过计算求得。

- 摄像机 (眼睛) 的位置 (eye point), 相当于p
- 观察点中心 (look at), 假设为向量c
- 向上向量 (up vector), 相当于u

可以通过下面方法求得d, r和u。

- $d = c - p$ , 下面第一幅图
- $r = d \times u$ , 下面第二幅图
- $u = r \times d$ , 下面第三幅图



注意上面的x是叉积运算（cross product），现在p, d, r, u四个量都已经知道，于是就可以根据上面的矩阵M求得视图变换矩阵了，对应的代码如下。该函数有三个参数，分别是摄像机位置p，向上向量u和视点中心lookAt。

```
D3DXMATRIX buildViewMatrix(D3DXVECTOR3& p, D3DXVECTOR3& u, D3DXVECTOR3& lookAt)
{
    // Calculate d
    D3DXVECTOR3 d = lookAt - p;
    D3DXVec3Normalize(&d, &d);

    // Calculate r
    D3DXVECTOR3 r;
    D3DXVec3Cross(&r, &u, &d);
    D3DXVec3Normalize(&r, &r);

    // Calculate up
    D3DXVec3Cross(&u, &r, &d);
    D3DXVec3Normalize(&u, &u);

    // Fill in the view matrix entries.
    float x = -D3DXVec3Dot(&p, &r);
    float y = -D3DXVec3Dot(&p, &u);
    float z = -D3DXVec3Dot(&p, &d);

    D3DXMATRIX M;
    M(0,0) = r.x;
    M(1,0) = r.y;
    M(2,0) = r.z;
    M(3,0) = x;

    M(0,1) = u.x;
    M(1,1) = u.y;
    M(2,1) = u.z;
    M(3,1) = y;

    M(0,2) = d.x;
    M(1,2) = d.y;
    M(2,2) = d.z;
    M(3,2) = z;

    M(0,3) = 0.0f;
    M(1,3) = 0.0f;
    M(2,3) = 0.0f;
    M(3,3) = 1.0f;

    return M;
}
```

注意事项，在求取View Matrix的时候有几点是需要注意的：

- 叉积满足右手法则。
- 叉积不满足交换律。 $a \times b = -b \times a$
- DirectX使用左手系，满足左手法则。

只要最终求得的三个向量，up, right和d满足左手法则即可。如果应用了View Matrix之后发现模型左右或者上下颠倒了，那么就说明求取的时候没有满足左手系。

好了，矩阵求解完毕，赶快使用SetTransform(D3DTS\_VIEW, &M) ;来试试吧，效果和使用函数D3DXMatrixLookAtLH是一样的！

Happy Coding!!!

作者：zdd

出处：<http://www.cnblogs.com/graphics/>

本文版权归作者和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接，否则保留追究法律责任的权利。

分类：DirectX

好文要顶 关注我 收藏该文



 zdd  
关注 - 20  
粉丝 - 687  
荣誉：推荐博客

[+加关注](#)

(请您对文章做出评价)

« 上一篇 : [用DirectX实现粒子系统 \( 三 \)](#)  
» 下一篇 : [【转载】DirectX支配游戏！历代GPU架构全解析](#)

posted on 2012-07-12 09:26 zdd 阅读(6624) 评论(6) 编辑 收藏

## 评论

### # 1楼[楼主]

公式是用Office的公式编辑器画的，然后存为图片，稍后可以改用Latex来画。

支持(0) 反对(0)

2013-04-02 10:24 | zdd

### # 2楼

@ zdd

你的文章图文并茂，赞一个  
话说取景变换的旋转是要使两个坐标系的三个轴都重合，这个一开始倒没注意

支持(0) 反对(0)

2013-06-06 14:23 | 子夜东风

### # 3楼[楼主]

@ 子夜东风

[引用](#)

@zdd

你的文章图文并茂，赞一个  
话说取景变换的旋转是要使两个坐标系的三个轴都重合，这个一开始倒没注意

这个龙书上有介绍。

支持(0) 反对(0)

2013-06-17 09:29 | zdd

### # 4楼

转置没写错？ $B = \{\{rx, dx, ux\}, \{ry, dy, uy\}, \{rz, dz, uz\}\};$

支持(0) 反对(0)

2013-11-05 18:03 | okay110

### # 5楼[楼主]

@ okay110

[引用](#)

转置没写错？ $B = \{\{rx, dx, ux\}, \{ry, dy, uy\}, \{rz, dz, uz\}\};$

确实错了，不过是原矩阵错了，我把u向量和d向量颠倒了，稍后修改。多谢提醒！

支持(0) 反对(0)

2013-11-06 09:29 | zdd

### # 6楼

$D3DXVec3Cross(&u, &r, &d);$

我觉得是 $D3DXVec3Cross(&u, &d, &r);$

支持(0) 反对(0)

2016-04-10 19:54 | Drimoon

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

【推荐】50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库

【推荐】融云即时通讯云 - 豆果美食、Faceu等亿级APP都在用

【推荐】报表开发别头大！类Excel 复杂报表开发实例，即学即用

【推荐】福利Time，讯飞开放平台注册即送好礼！

【推荐】阿里云万网域名：.xin .com将推出重磅优惠



# Meetup

野狗技术沙龙



## 最新IT新闻：

- Apple Watch运动表带展示方式获得设计专利
  - 3名宇航员在太空生活186天后安全返回地球
  - 吴文辉上演“王子复仇记”后，阅文集团为何还起内乱
  - 微软测试Windows 10新还原工具
  - 上线24天完成A轮融资 分答快速圈钱背后：盈利模式前景难料
- » [更多新闻...](#)

**JPush** 极光推送

消息推送领导品牌全面升级

**JIGUANG** 极光

极光推送

## 最新知识库文章：

- 学习如何学习
  - 一个32岁入门的70后程序员给我的启示
  - 技术发展瓶颈的突破
  - 高效编程之道：好好休息
  - 快速学习者的高效学习策略
- » [更多知识库文章...](#)

Powered by:  
博客园  
Copyright © zdd