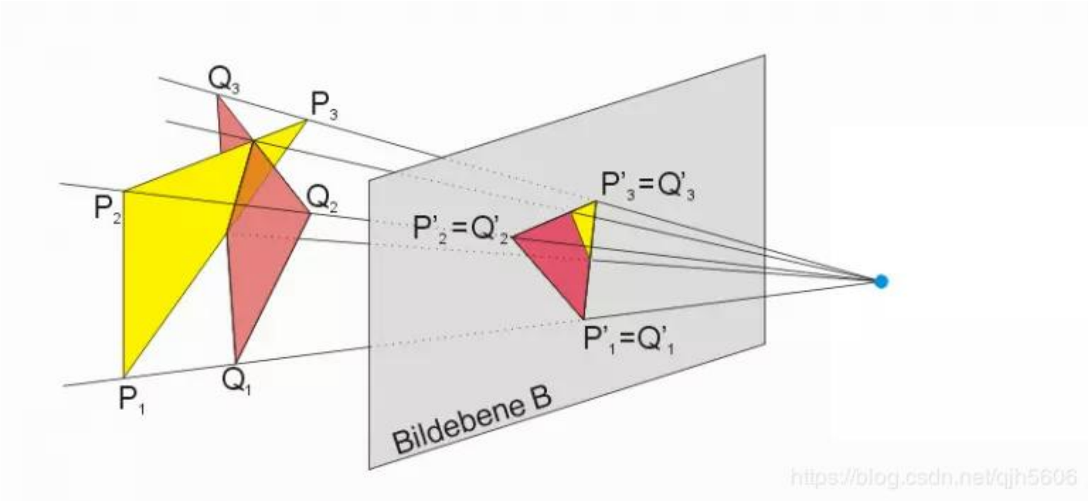
“透视除法” —— 齐次坐标和投影

转载自 写给大家看的“透视除法” —— 齐次坐标和投影 (https://www.jianshu.com/p/7e701d7bfd79)
作者：StanGame
链接：https://www.jianshu.com/p/7e701d7bfd79 (https://www.jianshu.com/p/7e701d7bfd79)
来源：简书



术语

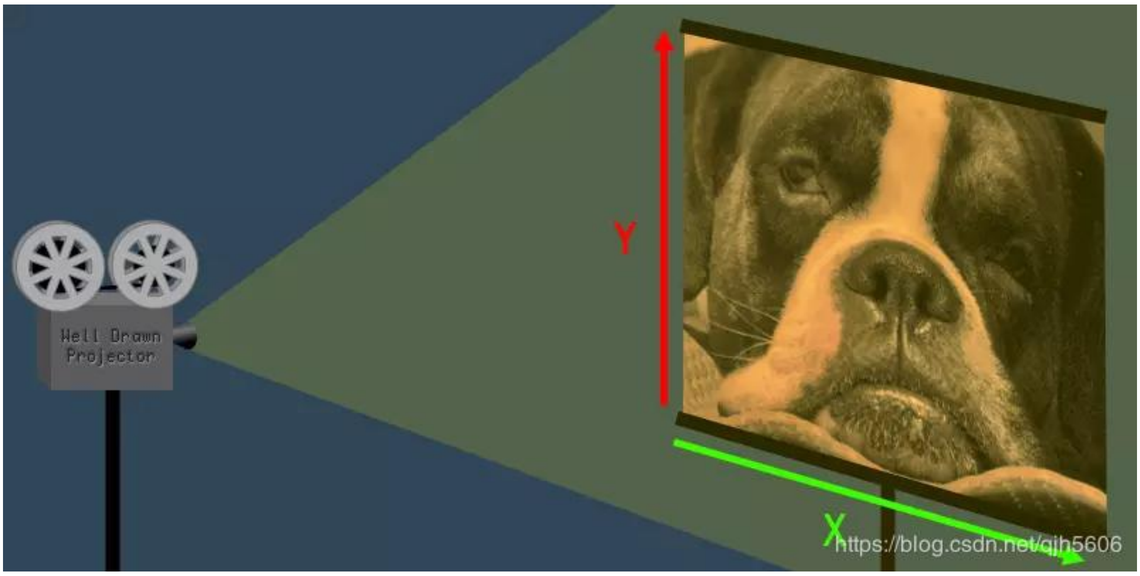
- 在大多数3D工作中，我们参照的依据是欧几里得几何学中的三维空间（X, Y, Z）。
- 但在某些情况下，参照投影几何更适用，除了 X, Y, Z 分量外，**增加一个 W 分量**，这个四维空间叫做**“投影空间”，在四维空间中的坐标叫**“齐次坐标”**。
- 为了达到3D软件的目的，“投影的” 和 “齐次的” 可以理解为 “4D”。

不是四元数

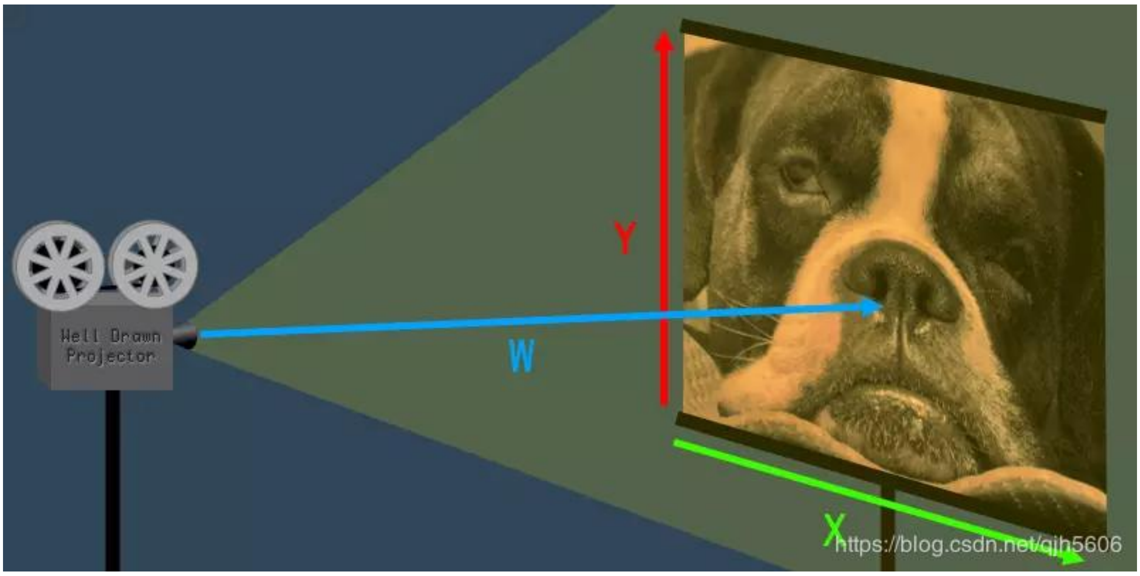
- 虽然四元数跟齐次坐标长得很像，都是 4D 矢量，通常用(X, Y, Z, W)来表示。
- 但是，四元数和齐次坐标是不同的概念，适用的领域也不同。
- 这篇文章跟“四元数”没有一毛钱关系

类比2D

了解3D之前，我们先看看2D的投影是怎么回事儿。
想象投影仪在一个屏幕上投影一张2D图片，很容易就可以得到投影图片的 X, Y 分量。



现在，看投影仪和屏幕之间，你就可以发现 W 分量了。
W 分量是投影仪到屏幕的距离

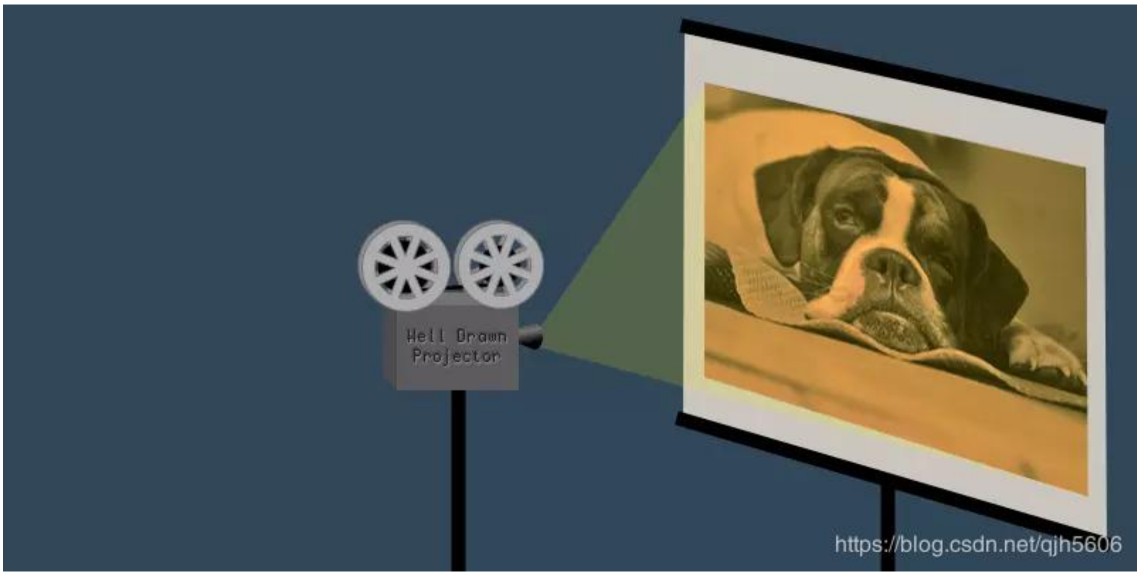


那么 W 分量的作用是什么呢?

想象一下，如果我们移动投影仪的位置，来增加或减少 W分量的值，那么投影出的2D图片会发生什么?

- 如果将投影仪靠近屏幕，2D图片缩小，
- 如果投影仪远离屏幕，2D图片放大。没错，这就是 W 分量的作用。

W 分量的值，影响了投影出 2D 图片的大小



应用到3D

到目前为止，还没有一个3D投影仪，很难想象3D中的投影几何，但是 3D 下的 W 分量与 2D 的作用相同。

- **当 W 增大，坐标被拉伸；W 缩小，坐标被压缩， W 对3D坐标做缩放变换。**

W=1时

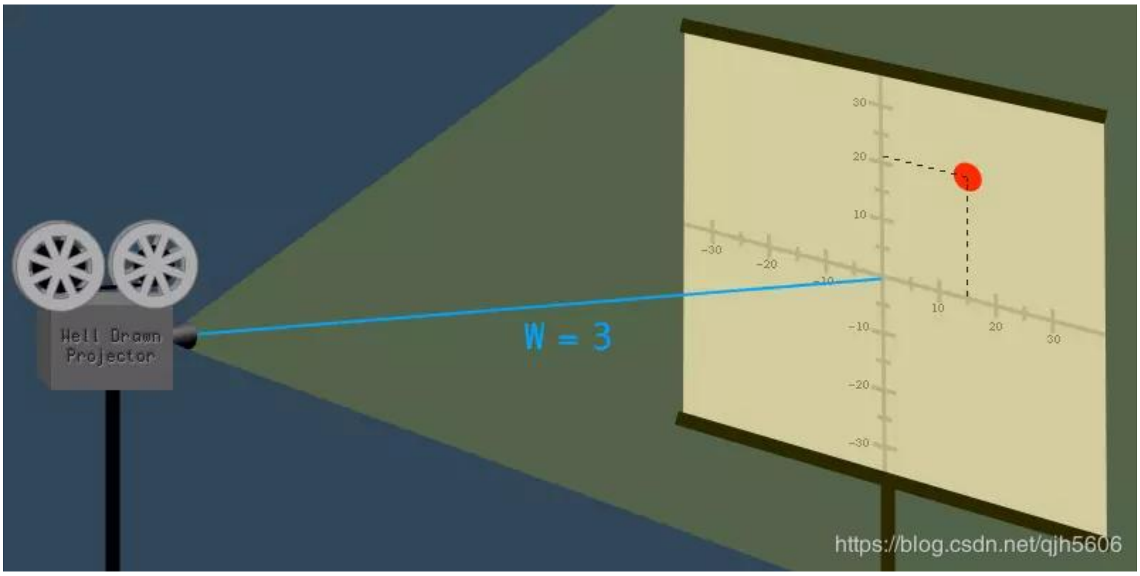
- 通常，给3D编程初学者的建议是，无论什么时候将 3D 坐标转换为 4D 坐标时，让 W=1。原因是，当缩放坐标的 W 为1时，坐标不会增大或缩小，保持原有的大小。所以，**当 W=1，不会影响到 X, Y, Z 分量的值**。
- 因此，当谈论到 3D 计算机图形学时，当坐标中 W=1时被称作“正确”。
    - 如果渲染使用 W>1 的坐标，每一个3D物体看起来都会变大，
    - 反之，W<1 的坐标中，3D物体会变小；

- 如果渲染时试图让 W=0，那么你的程序会奔溃，当做透视除法的时候除数为0；
- 如果 W< 0，每一个物体都会上下翻转，水平翻转。

在数学中，没有所谓的“不正确”的齐次坐标，**使用齐次坐标时让 W=1 仅仅是用于计算机图形学中的投影转换。**

数学原理

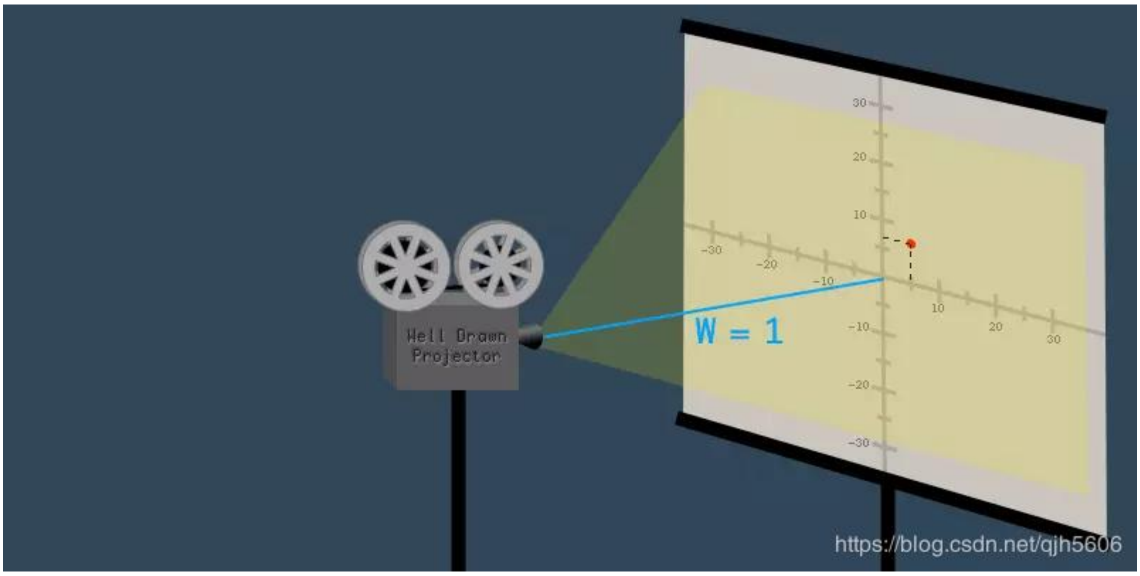
现在，让我们来看一些例子，了解数学原理
在距屏幕3米远的位置放一个投影仪，投影出一个点(15, 21)在 2D 图像中，相应的投影坐标中的向量为 (X, Y, W)=(15, 21, 3)。



现在，想象推动投影仪向屏幕靠近，直到距离1米，越靠近屏幕投影，投影出的图像越小。投影仪靠近了3倍，因此图像缩小了3倍。如果我们将原向量的 X, Y, W 分量都除以 3，我们得到一个新向量 W=1：

$$\left(\frac{15}{3}, \frac{21}{3}, \frac{3}{3}\right) = (5, 7, 1)$$

投影出的点在坐标中的新位置(5, 7)



这就是怎样将一个“不正确”齐次坐标转换到一个“正确”坐标的方法：所有分量除以 W，这个过程对 2D 和 3D 同样适用。
通过给向量乘以 W 的倒数，**来实现向量的所有分量除以 W**，下面是一个4D的例子：

$$\frac{1}{5}(10,20,30,5) = (\frac{10}{5},\frac{20}{5},\frac{30}{5},\frac{5}{5}) = (2,4,6,1)$$

用 GLM 库编写，类似如下实现：

```
glm::vec4 coordinate(10, 20, 30, 5);
glm::vec4 correctCoordinate = (1.0/coordinate.w) * coordinate;
//now, correctCoordinate == (2,4,6,1)
```

在计算机图形学中使用齐次坐标

就像开始提到的，针对3D 计算机图形学中，有些情况下使用齐次坐标很有用，下面我们来看看这些情况：

3D 坐标中的转换矩阵

- 旋转和缩放的转换矩阵只需要3列，但是为了处理平移，至少需要4列矩阵，这就是为什么矩阵变换通常用4×4的矩阵。
- 然而，4列矩阵不能与3维向量相乘，**只能与4维向量相乘**，这就是为什么我们使用齐次的4维向量取代3维向量。

4列矩阵只能与4维向量相乘，这就是为什么我们常常使用齐次的4维向量取代3维向量。

- 通过齐次坐标处理矩阵变化，第4维 W 分量通常不用改变。
- 从3D转换到4D，只需将 W 分量设置为1，并且经过变换矩阵处理后，W 分量的值任为1，这意味着我们忽略 W 分量即可转换回 3D 坐标。
- 这个对目前为止大多数的矩阵变化都适用，如平移、旋转、缩放。
- **需要注意的例外是投影矩阵会影响 W 分量。**

透视变换

在 3D 世界，物体离相机越远看起来越小，这个现象叫做透视。在镜头中，如果猫离相机足够近，远处的大山会比猫看上去小。

- 在3D计算机图形学中，**透视是通过变换矩阵改变向量的 W 分量来实现的。**
- 在变换到相机空间后（对向量应用了相机矩阵），还没有进行投影变换（还没有对向量应用投影矩阵），每个向量的 **Z 分量表示了距离相机的距离。**
- 因此，Z 分量越大，矢量应该越小。
- W 分量影响这个缩放，所以**投影矩阵用 Z 分量的值改变 W 分量的值。**

在3D计算机图形学中，透视是通过投影矩阵变换，改变每一个向量中 W 分量的值来实现透视的

下面看一个透视例子，通过投影矩阵变换到齐次坐标。

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 4 \end{bmatrix}$$

注意: **投影矩阵是怎样用 Z 分量改变 W 分量的**。

- 经过投影矩阵透视变换后，每一个向量即经过了“透视除法”。
- 透视除法只是将齐次坐标中的 W 分量转换为1的专用名词

继续上面的例子，透视除法这步如下:

$$\frac{1}{4}(2,3,4,4) = (0.5, 0.75, 1, 1)$$

完成透视除法后，W分量就没用了，我们就得到了一个完全符合3D透视投影规则的3D坐标。

- 在GLM中，透视投影矩阵可以通过使用 `glm::perspective` 或 `glm::frustum` 方法来创建。
- 在 OpenGL 中，在顶点 shader 作用了每一个顶点后，**自动进行透视除法**。
- 这就是顶点 shader 中 main 方法输出的 gl_position 变量，是4维向量，而不是3维向量。

设置平行光

齐次坐标中的一个属性，是 **允许有一个无限远的点（或无限长的向量）**，在3D坐标中这个是不允许的。

- **当 W=0 时，这点表示无限远的一个点。**
- 如果你尝试将一个 W=0 的齐次坐标转换为一个普通的 W=1的齐次坐标，这会导致4次除0操作:

$$\frac{1}{0}(2,3,4,0) = (\frac{2}{0}, \frac{3}{0}, \frac{4}{0}, \frac{0}{0})$$

这意味着，不能将 W=0 的齐次坐标转换到 3D 坐标。
那这有什么用呢? 用处说来就来了，**平行光可以认为是一个无限远处的点光源**。当一个点光源在无限远的位置，光线就会变成平行的，并且所有光线都在同一方向，这就是平行光的基本定义。想想太阳吧。
所以在传统的3D图形中，平行光可以通过改变点光源位置向量中的 W 分量来表示

- 当 W= 1时，是一个点光源;
- 当 W= 0 时，是一个平行光。

在实现光照代码时，这更多的是一种传统的约定，但不是一种有用的方法。因为平行光和点光源的行为不同，通常分开实现。一个经典的光照 shader 实现如下:

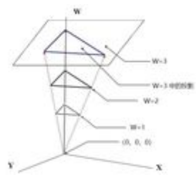
```
if (lightPosition.w == 0.0) {
  //directional light code here
} else {
  //point light code here
}
```

总结

- 齐次坐标有一个额外的维度叫 W 分量，用来缩放X, Y, Z三个分量的值。
- 平移和透视投影的矩阵变换只能在齐次坐标中使用，所以在3D计算机图形学中，当 W=1时，X, Y, Z 分量被称为“正确的”。
- 任何齐次坐标，只要 W 不为0，都可以通过将 **每一个分量除以 W 来转换到 W=1的向量**。
- 当 W=0 时，这个坐标表示无穷远的一个点（或者表示无限长的一个向量），通常用于表示平行光的方向。

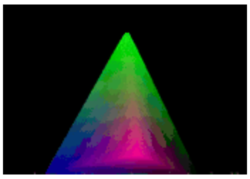
原文：Explaining Homogeneous Coordinates & Projective Geometry (http://www.tomdalling.com/blog/modern-opengl/explaining-homogenous-coordinates-and-projective-geometry/)

智能推荐


(/article/83461264798/)

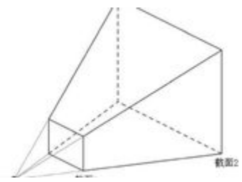
齐次坐标 (/article/83461264798/)

齐次坐标 在数学里，齐次坐标（homogeneous coordinates），或投影坐标（projective coordinates）是指一个用于投影几何里的坐标系统。该词由奥古斯特·费迪南德·莫比乌斯于1827年在其著作《Der barycentrische Calcul》一书内引入。 齐次坐标可让包括无穷远点的点坐标以有限坐标表示。使用齐次坐标的公式通常...


(/article/764331406/)

OGL（教程12）——透视投影 (/article/764331406/)

原文地址：http://ogldev.atspace.co.uk/www/tutorial12/tutorial12.html 背景知识： 我们最终来到了最重要的一节，把3D世界映射到2D平面，这个映射还要保留深度信息。一个很好的例子是，铁路轨道的图片，在很远的地方两个轨道讲汇聚于一点。 如图： 我们准备推导一个变换以能够给满足上面的需求，我们还有一个需求，使用这个变换的时候，同时也把裁剪工作做了...


(/article/8628186441/)

透视投影矩阵的推导 (/article/8628186441/)

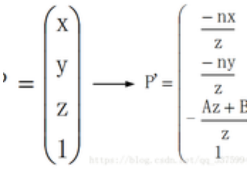
视锥体 如图，近截面与远截面之间构成的这个四棱台就是视锥体，而透视投影矩阵的任务就是把位于视锥体内的物体的顶点X,Y,Z坐标映射到[-1,1]范围。这就相当于把这个四棱台扭曲变形成一个立方体。这个立方体叫做规则观察体 (Canonical View Volume, CVV)。如下图： 变换方法...


(/article/1282452784/)

透视投影矩阵的构建 (/article/1282452784/)

投影矩阵最终建立的是一个平截头体（也可以称为台），在这种变换下呈现远小近大的效果。这里我将我学到知识记录下来，以后备忘用。 蒋彩阳原创文章，首发地址：http://blog.csdn.net/gamesdev/article/details/44926299。欢迎同行前来探讨。 首先是使用OpenGL的glFrustum函数...

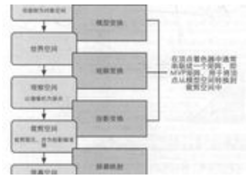
opengl透视投影概念梳理 (/article/9638649615/)



(/article/9638649615/)

感谢原作者的分享，从基本概念娓娓道来，而且讲的通俗易懂，条例分明，以下是我按照自己习惯的理解方式进行了重新记录。 https://blog.csdn.net/popy007/article/details/1797121 1. 将目标物体投影到近裁剪平面上 (P >> P') 我们一步一步来，我们先从一个方向考察投影关系。 上图是右手坐标系中顶点在相机空间中的情形。设P(x...

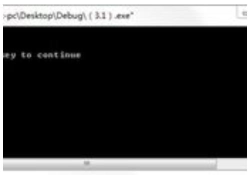
猜你喜欢



(/article/1729830344/)

透视投影矩阵的推导 (/article/1729830344/)

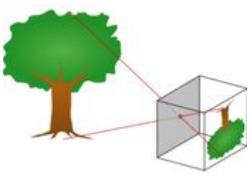
上图是3D渲染过程中的空间变换过程，这里主要讨论观察空间到裁剪空间的转换。 由上图可知，经过观察变换后，空间变换为观察空间，也就是以摄像机坐标为原点的空间，这是接下来推导的前提。 下图是一个视锥体，在视锥体范围内的物体为可视的，不在视锥体内的为不可视。 我们有两个任务： 1、判断一个点是否在视锥体内（用于裁剪超出屏幕的点）。 2、计算顶点在裁剪空间上的坐标。 我们先忘掉矩阵的概念，只用基本的三维几...



(/article/24601410383/)

简单除法和求余 (/article/24601410383/)

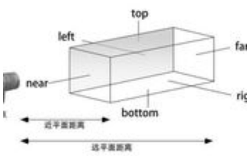
#include <stdio.h> main() { int a=12,b=3; float x=18.5,y=4.6; printf ("%f\n",(float)(a*b)/2); printf ("%d\n",(int)x%(int)y); 知识点：1.不...



(/article/75321124953/)

投影相机，透视相机，弱透视相机和仿射相机的区别和联系 (/article/75321124953/)

投影相机，透视相机，弱透视相机和仿射相机的区别和联系 2019.11.03 FesianXu 前言 相机一般来说是一种从3D到2D的一种投影工具，其按照数学模型可以分为投影相机，透视相机，弱透视相机和仿射相机等，笔者在本文中尝试对其进行区分和联系。如有谬误，请联系指正。转载请注明出处。 ∇\nabla∇ 联系方式： e-mail: FesianXu@gmail.com QQ...



(/article/568489692/)

图像形成（5）球面透视投影和近似相机模型 (/article/568489692/)

文章目录 图像形成（5）球面透视投影和近似相机模型 球面透视投影 近似相机模型 图像形成（5）球面透视投影和近似相机模型 球面透视投影 我们在博文图像形成（3）理想相机模型中描述的透视针孔相机模型考虑平面成像表面。 另一种常用的成像表面是球体，如图1所示。 球面透视投影模型：三维点ppp的像是穿过光学中心ooo的光线与光学中心周围的半径为rrr的球体的交点处的点xxx。 通...

(/article/53821984638/)

初学嵌入式是有必要备一块开发板的_4412了解一下 (/article/53821984638/)

如果了解一下当前IT和物联网发展的形势，就会发现Android工程师越来越受欢迎，相比之下单纯的Linux工程师逊色不少，当然，Android系统的内核也是Linux的，Linux和Android作为当前开源的俩大系统，其发展势不可挡。所以学习Android系统架构是提升自身价值非常重要的选择，它会给我们不一样的天空和视野。而4412开发板很好的结合了Linux和Android俩套系统。...

相关文章

OpenGL 正交投影、透视除法、透视投影 (/article/24221704429/)

渲染流水线-透视投影矩阵 透视除法 空间裁剪 视口变换 数学原理理解 (/article/37051052171/)

软件光栅器(Directx11)三之世界矩阵，相机变换矩阵，透视投影矩阵，透视除法,视口变换矩阵 (/article/25281077714/)

几何变换——关于透视变换和仿射变换以及齐次坐标系的讨论 (/article/20331171899/)

图形学笔记 —— 透视除法 (/article/49561487886/)

透视投影详解 (/article/265097995/)

透视投影变换 (/article/409899169/)

透视投影矩阵推导 (/article/56161672368/)

three.js 正交投影和透视投影 (/article/3191808001/)

齐次坐标 (/article/74061033386/)

热门文章

JVM学习（一）非线程共享-运行时数据区域 (/article/3176411006/)

openUI5/SAPUI框架介绍(持续更新) (/article/99151950778/)

如何在低版本的libc.so的系统上安装高版本编译的rpm包 (/article/2079418078/)

Java虚拟机03——垃圾收集算法 (/article/8547331864/)

优秀的项目经理是如何管理项目时间与任务的 (/article/6535779629/)

git subtree用法 抽取公共的组件 (/article/18811536979/)

操作系统物理内存管理：连续和非连续 (/article/6957299696/)

手机为何老提示网络连接不可用？ (/article/96262064891/)

多模态融合(三)MFAS: Multimodal Fusion Architecture Search (/article/91172094974/)

Redis事务、持久化、发布订阅 (/article/9558801544/)

推荐文章

【科创人】贝锐创始人陈宇晔：花生壳诞生自一次挫折，15年坚守有温度不作恶 (/article/18302524067/)

messageutil.java_JAVA利用第三方平台发送短信验证码。 (/article/90162461449/)

如何强制关闭mac后台程序 (/article/7455848782/)

Windows Server 2012和Windows 8中的远程管理 (/article/6703669834/)

进入目录需要哪些权限, 在目录中执行增删查(cd, touch, ls, rm, mv等)改文件动作, 需要哪些权限 (/article/1658980503/)

基于java的心理咨询与诊断平台 (/article/32732264028/)

mysql多表查询结果合并_MySQL多表查询合并结果union all,内连接查询 (/article/12742345518/)

【Visual C 】游戏开发笔记十六 讲解一个完整的回合制游戏demo (/article/160486745/)

十一课堂|通过小游戏学习Ethereum DApps编程（5） (/article/752919219/)

threejs中FBX格式模型的加载与克隆 (/article/712348830/)

相关标签

- opengl (/tag/opengl/)
- Android (/tag/Android/)
- matrix (/tag/matrix/)
- 正交矩阵 (/tag/%E6%AD%A3%E4%BA%A4%E7%9F%A9%E9%98%B5/)
- 几何变换 (/tag/%E5%87%A0%E4%BD%95%E5%8F%98%E6%8D%A2/)
- 仿射变换 (/tag/%E4%BB%BF%E5%B0%84%E5%8F%98%E6%8D%A2/)
- 透视变换 (/tag/%E9%80%8F%E8%A7%86%E5%8F%98%E6%8D%A2/)
- 投影变换 (/tag/%E6%8A%95%E5%BD%B1%E5%8F%98%E6%8D%A2/)
- 计算机视觉 (/tag/%E8%AE%A1%E7%AE%97%E6%9C%BA%E8%A7%86%E8%A7%89/)
- three.js (/tag/three.js/)