本文欢迎转载，但是请著名出处：<https://github.com/lygyue/Books>

本文曾经是一个CSDN公开课，后整理发表于知乎专栏。由于是公开的东西，其实反而增加了太多限制，例如我自己都觉得编辑之后废话特别多：（。

网址：<https://zhuanlan.zhihu.com/p/26360935>



# **“V”来已来——浅谈光学VR实现原理**

本文是在CSDN一个公开课的演讲，现整理发布于知乎，为笔者原创。欢迎VR爱好者与从业者相互交流学习。

VR（Virtual Reality），似乎从2016年起，成为了一个人人都喜欢挂在嘴边的新鲜名词，好像不谈论VR就像你不认识特斯拉不听薛之谦不看爱乐之城一样，暗示着与时代的“脱节”。如果你偶然得知身边有人在做“VR项目”，那就好像是发现了下一个马克·扎克伯格一样，相信他一定会成为时代的“弄潮儿”。不过，大多数人对于VR的理解，也仅仅停留在“**虚拟现实**”这四个字上面。那么，VR究竟是怎样的，它的实践原理又是什么，本文将尝试用较为简单的语言，来阐述这一“**现在与未来的风口**”。

本文将主要从红极一时Google CardBoard到号称很可能改变未来人们游戏的方式的Oculus Rift来与大家探讨VR的实现原理与前景。首先，我们一起来看两个简单却基本的问题：

**1. 为什么目前的VR被称为光学VR？**

答：因为人们对世界的认识，是通过**人眼观察世界**。目前一切的VR，都是通过扭曲光线，让光线进入视网膜，欺骗眼睛来实现的。

**2. 什么情况下会有脱离光学的VR？**

答：人类可以脱离眼睛沟通这个世界的时候（笑）。

这两个问题看似简单，实则涉及到本文的一个核心要点：**光学**。不管是VR（Virtual Reality 虚拟现实），还是AR（Augmented Reality 增强现实），还是号称象征着未来的MR（Mix reality 混合现实），都脱离不了光学。本文会将**VR的基础实现原理**尽量讲一遍，由于读者对行业的了解程度关系，可能不会讲得太深入。但是，如果有朋友希望了解一些更深入的东东，欢迎大家在文章下留言评论，我会尽力与大家交流解答。



**最廉价的VR体验设备——Google CardBoard一代**

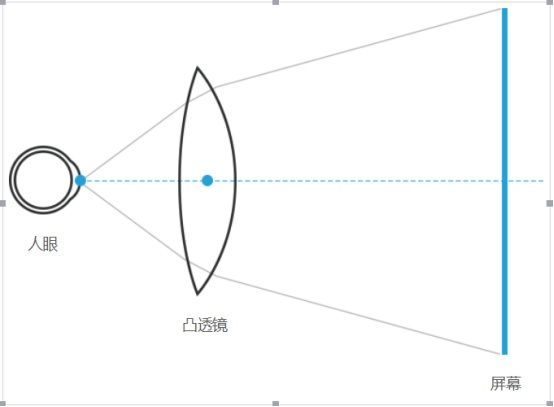


**曾击败Xbox One与PS4获得“最佳硬件奖”的游戏设备——Oculus Rift CV1**

以上两个，就是比较有代表性的**HMD**（做笔记以后出去谈VR的时候又有了一个专（zhuang）业(bi)名词了呢），**HMD是Head Mount Display的缩写，也就是俗称的头显**，这是VR最核心的设备，只有有了这个东西，关于VR的一切才有可能。

上面的两个设备，一个是最原始的Google Cardboard，一个是大名鼎鼎的“三大头显”之一：Oculus rift CV1，CV1看起来相较于Google Cardboard显得十分高大上，充满了科技感与时尚感。但究其核心，还是那些东西：**一个塑料外壳，两个镜片，一个显示器。**看到这里有些小伙伴可能不免失望，原来所谓的科技最前沿，就是这3样东西呀。不急，下面我们来继续探究其中的原理。

首先我们来看一个大致的结构图：



这是一个典型的VR眼镜功能模拟，主要包括**三部分**：

* **人眼**
* **凸透镜**
* **OLED成像屏幕**

看起来大家会觉得更简单了，好像我随便**找个屏幕，找个透镜，再找个盒子，就组成了一个VR设备**（瞬间化身科技公司大佬可以指点江山了）。那么究竟实时是否如此呢？答案是：**是的！**大家没看错，国内一大把各种魔镜，都是这么做的。那么究竟怎么做呢？ 下面为大家介绍一条**创(fa)新(jia)创(zhi)业(fu)之路：**首先，先去市场（淘宝），购买一大把透镜，从几毛到几块一块都有，然后找一拨人（例如公司员工），大家一块一块看，最终大家公认哪一款效果最好，然后定了，就这款了!再用手工做一个手版的盒子，装上镜片，再拉上这帮人一起看效果，慢慢调，当效果大家觉得凑合的时候，那么，参数定下来了，一款VR眼镜诞生了！

这种VR眼镜，被称作**人海VR**，常见于市面上几十、一百块的各种盒子，在华强北据称一年出货量几千万个。这类盒子的体验效果可以预见是非常差的，容易使人晕眩，基本上可以看做是一个玩具。

那么，一款优秀的HMD应该是怎么样的呢？笔者个人认为**以下两点至关重要**：

* **完全符合人体结构**
* **尽量轻便，降低存在感，也就是说，你戴上去跟没戴一样**

第二点是小型化问题，目前不考虑，我们来详细讲讲第一点。

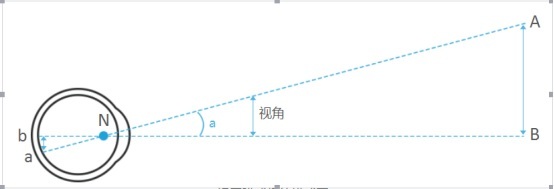
要做好完全符合人体结构，或者说尽量的去符合人体结构，至少有**以下因素需要考虑**：

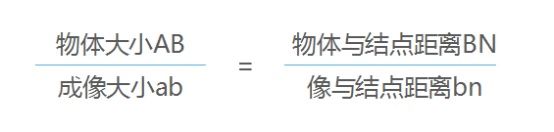
* **人眼观察角度（也可以叫视场角，简称FOV——Field of View）**
* **人双眼之间的距离（俗称瞳距，简称IPD——Interpupillary distance）**
* **人眼到镜片的距离**
* **镜片到屏幕的距离**
* **屏幕成像的大小计算**
* **屏幕成像的反畸变**
* **屏幕成像的渲染帧率**
* **屏幕的刷新延迟**

……

一款优秀的HMD，至少要考虑到这8点。其实除了这8点，还有其他更复杂的东西，例如**自动对焦，运动模糊**等。下面，我们会侧重谈谈这8点应该如何来考虑，做到最好。

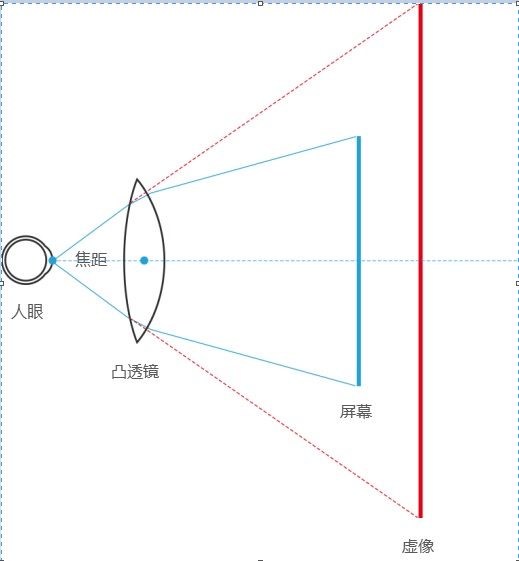
要考虑设计一款优秀的HMD，首先要想明白一个问题，**VR的沉浸感**从何而来？要知道这一点就需要先了解视网膜成像原理：





通过视网膜成像，人眼观察着这个世界。VR如何欺骗了眼睛呢？答案是：**凸透镜**。

再看下图：



上图很好的表述了一个观点：**你看到的世界不一定是真实的**。你以为看到的是红色的虚像世界，实际你所看到的，只是蓝色屏幕中的一方天地。

为什么我们需要一块透镜？尝试一下，伸出一根手指，放在你的眼前1厘米，你会发现，你看不清你的手指。很诡异是不是？人眼的成像是有距离的，透镜是为了把近距离的图像放大成一个虚像，增加成像距离。

通过该图，很容易得出一些结论：

* **HMD不能漏光。一旦漏光，谈不上什么沉浸感了**
* **人眼到镜片最合适的距离，就是镜片的焦距稍稍往前。**由此我们可以知道，镜片尽量设计到焦距够小（便于镜片覆盖眼睛）
* **屏幕到透镜的距离，跟镜片的散射角度有关**（参考一下蓝线倾斜即可）
* **最理想的状态是，人眼观察角度与红线部分重合**（基本上不可能达到理想状态，因为HMD是固定的，而人与人眼睛是不同的）。

好了，根据这四点结论，我们可以比较轻易判断，对于一款HMD而言，合适的镜片至关重要，直接决定了HMD的最终质量。因此，在VR镜片上，各大公司可谓不遗余力。除了所谓的光学镜片，最近还在热炒各种概念，例如**菲涅尔镜片**（Fresnel lens）**，光场镜片**（这块独立出来都是一个很复杂的概念，由于篇幅有限，这里不打算细讲）。

既然一款镜片如此重要，那么，如何才能设计出一款足够牛逼的镜片？需要重点关注哪些参数？下面几个参数非常重要：

* **视场角（FOV）**
* **符合人眼构造的成像系统**
* **清晰度**

视场角多少合适？有一种主流的看法是：**合理范围内，视场角越大越好**。那么，什么叫做**“合理范围内”**？，可以理解为：

* **没有导致明显的透视变形之前**（关于透视变形，这是3d的基础概念，不打算科普，大家度娘一下就能找到）
* **尽量达到人眼最大视场角。**由于人的眼珠是可以转动的，单眼最大理论视场角大概在150度左右。那么，FOV要达到150度吗？其实不是，FOV大小还跟屏幕分辨率有关，当分辨率不足够的时候，FOV越大，会导致纱窗效果越明显。关于这个，计算也很简单，只需要计算观察范围面积（通过FOV和观察距离），再用屏幕像素 / 观察面积，就能够得到每平方cm有多少个像素。单位范围内像素越小，效果越差。如果是做开发的应该能轻易算出来。

因此，在目前大部分设备上，视场角并没有越大越好，**普遍在90 - 100之间，最高的貌似也就105左右**

**符合人眼的成像系统，这个又是什么意思呢？**熟悉3d图形学的朋友会知道：**3d中，透视投影变换主要是三个矩阵：world， view， projection**。前两个是坐标变换，不在今天的讨论范围，第三个，却是实打实的投影模拟。但是，在普通3d游戏中，这个投影模拟的不是人眼视网膜，而是一个计算机显示器窗口，所以，这个矩阵的计算参数一般有：**Y方向FOV，窗口宽高比，最近可视距离，最远可视距离**。大概算法如下：

float thetaY(mFOVy / 2.0f);

float tanThetaY = tan(thetaY);

// Calc matrix elements

float w = (1.0f / tanThetaY) / mAspect;

float h = 1.0f / tanThetaY;

float q, qn;

q = -(mFarDist + mNearDist) / (mFarDist - mNearDist);

qn = -2 \* (mFarDist \* mNearDist) / (mFarDist - mNearDist);

// [ w 0 0 0 ]

// [ 0 h 0 0 ]

// [ 0 0 q qn ]

// [ 0 0 -1 0 ]

Matrix4 dest = Matrix4::ZERO;

dest[0][0] = w;

dest[1][1] = h;

dest[2][2] = q;

dest[2][3] = qn;

dest[3][2] = -1;

mProjectionMatrix = dest;

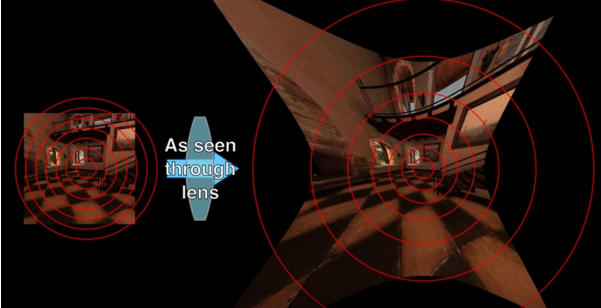
问题来了，我们做VR，适合采用这种方式吗？答案是：这样可以做，但是效果不够理想，因为并没有符合人体结构。

在了解“**符合人体结构**”这个概念前，大家可以做一个小实验。尝试闭上一只眼睛，然后只用一只眼睛，尽力前后左右看，你觉得这是一个正投影吗？也就是说，上下，左右看到的角度大小，是一样的吗？很显然，我们基本能够确定，上下，左右能观察到的范围，并不对称，这才是真正的人体结构。

因此，我们计算的投影矩阵，理想的状态，不能是正矩阵。理想的数据是多少？这里我给出Oculus的一个眼睛数据：上下左右角度分别为：41.65， 48， 43.98， 35.57。

那么，这个角度范围就是最理想的范围了吗？答案是：不是。既然不是（正常人平均值大概是56、74、91、65），作为VR界的领军人物，难道Oculus不知道这个事情吗，为什么不直接用最合适的范围角度？很简单，问题在于光学镜片的设计，光学镜片并不能为所欲为的设计这个视场角（还有一个原因还是屏幕分辨率），所以现在的VR大热菲尼尔镜片应运而生。菲尼尔镜片好处多多，可以有其他很多好处例如清晰度，畸变……**据说希拉里的眼镜就是菲涅尔镜片哦**。

综上所述，我们大概理清了镜片的一些重要的参数和合适设计，下面，我们来讲讲重要的**成像**以及**反畸变**。



**原图（左）经过VR观看的效果（右）**

关于成像与反畸变的一些问题解答：

**VR成像跟普通的3d游戏成像有什么不同？**

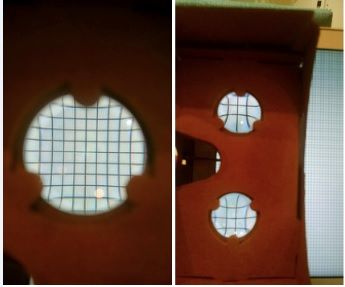
* **Projection Matrix不同**。参见上一条，Projection的计算不能采用传统的正矩阵计算。
* **View Matrix不同。**考虑到IPD（瞳距），View Matrix的计算应该分左右眼，position的计算应该是：**LeftPosition = position + rotation \* (-IPD / 2);RightPosition = position + rotation \* (IPD / 2)**;其实就是左右眼分别沿x轴的正负方向偏移IPD的一半。
* **渲染方式不同**。正常的3d渲染是单屏，而VR是需要先渲染两只眼睛，得到render texture，然后再把左右眼贴图渲染一遍到屏幕。
* **由于第三个特性，VR渲染跟传统渲染相比，有很多细节需要注意**。例如到底使用延迟渲染还是使用向前渲染，抗锯齿应该怎么处理，模糊怎么处理……一句话概括：VR有可能使得传统的渲染模式发生改变。典型的例如Google Day Dream本身推荐使用放大RTT的做法来实现抗锯齿（跟FSAA类似）。

VR成像的内容不算太多，更重要的是反畸变。

**为什么要反畸变？**

答：**因为正常的东西，通过凸透镜去看，是变形的**（随便拿块透镜试试便知）。所以我们不能够直接把render texture直接贴到屏幕上，需要把图片做一个反畸变，然后通过透镜的畸变，观看的画面变成了正常。

**那么问题来了，这个反畸变应该如何做？**我们先来看看大名鼎鼎的Google CardBoard的实现方式。



这是一张放烂了的图，Google文档专用。在他们的开发文档里，他们号称使用了布朗畸变模型（细节可以在这里看到：[https://en.wikipedia.org/wiki/Distortion\_(optics)，又或者google一下vr](https://link.zhihu.com/?target=https://en.wikipedia.org/wiki/Distortion_(optics)%EF%BC%8C%E5%8F%88%E6%88%96%E8%80%85google%E4%B8%80%E4%B8%8Bvr" \t "https://zhuanlan.zhihu.com/p/_blank) lens distortion，资料很多）。这里，我就不做纯粹的搬运工，直接复制粘贴，来帮大家分析一下要点：

* **首先，得到镜头的中心点**（中心点由于是垂直通过，没有角度，无畸变）
* **任意点的畸变，先计算该点到圆心的半径**
* **根据镜头K1、K2的参数，计算出畸变后的坐标**
* **纹理采样时，根据这个偏移做UV偏移，采样**

这个是大概过程。目测很多好奇的群众会跟我一样，好奇这个K1、K2从何而来？我其实也很好奇，但是由于Google的文档语焉不详，很诡异的是，Google文档里面还提及到手工调节这两个参数，言下之意是：如果你根据镜片厂商提供的这两个参数，没有达到最好的效果，请手动调节这两个参数，以达到最好的效果。

再来看看同样大名鼎鼎的Oculus Rift的实现方式。其实Oculus早期采用了跟Google一样的方式（谁模仿谁不知道），**只不过两个参数K1、K2变成了四个：K1，K2，K3，K4**，其实这样做意义不大，最早期的数据，K3很接近0，K4直接是0，几乎不起作用，后期陆陆续续有各种改动。但是，Oculus越做越好，后面彻底摒弃了之前的方式，**采用了全新的Distortion Mesh的渲染方式**。我们的镜片，同样采用了这种方式，核心做法是：

* **设计好镜片**
* **根据镜片的参数（大概20多个），生成一个畸变模型**
* **畸变模型的UV要分为RGB三块，三组UV**
* **贴图渲染的时候，分别做RGB采样，得到新的颜色的RGB，组合成新颜色**

大概shader代如下：

static char\* defaultDistortionVertexShaderSrc =

"float4x4 ProjView;float4 MasterCol;"

"void main(in float4 Position : POSITION, in float2 TexCoordR : TEXCOORD0, in float2 TexCoordG : TEXCOORD1,in float2 TexCoordB : TEXCOORD2,"

" out float4 oPosition : SV\_Position, out float4 oColor: COLOR0, out float2 oTexCoordR : TEXCOORD0, out float2 oTexCoordG : TEXCOORD1, out float2 oTexCoordB : TEXCOORD2)"

"{ oPosition = Position;"

" oColor = MasterCol;"

" oTexCoordR = TexCoordR;"

" oTexCoordG = TexCoordG;"

" oTexCoordB = TexCoordB;}";

static char\* defaultDistortionPixelShaderSrc =

"Texture2D Texture : register(t0); SamplerState Linear : register(s0);"

"float4 main(in float4 Position : SV\_Position, in float4 Color: COLOR0, in float2 TexCoordR : TEXCOORD0, in float2 TexCoordG : TEXCOORD1, in float2 TexCoordB : TEXCOORD2) : SV\_Target"

"{ float4 TexColR = Texture.Sample(Linear, TexCoordR);"

" float4 TexColG = Texture.Sample(Linear, TexCoordG);"

" float4 TexColB = Texture.Sample(Linear, TexCoordB);"

" return float4(TexColR.r \* Color.r, TexColG.g \* Color.g, TexColB.b \* Color.b, 1); }";

那么，**两种方式的差别是什么？是什么原因导致了两大巨头分别采用了不同的实现方式？**

我个人的判断是：Oculus的方式，效果明显要优于Google的方式，**最明显的问题，在光线经过透镜的时候，RGB由于波长不同，折射率是不一样的**，这部分在高中物理里有讲过。**所以纹理反畸变的时候，必须要做rgb偏移，不然，一定会有色差问题，并且边缘地方越加明显。**

既然如此，难道Google不知道吗，为什么不也采用Oculus的方式？我的看法：**Google希望做的是一个开放的系统，希望适配所有的镜片与手机**。这可以理解为是**从商业角度去考虑**。

而使用Distortion Mesh的方式，不可能做到这一点或者说很不容易做到这一点（我其实觉得可以开放驱动接口的方式也是可以实现的，详情请看Windows适配各种硬件），但是，Google采用了最简单暴力的方式，这样的话，让移动平台的VR蒙上了阴影（Google最新推出的DayDream系统、盒子、样机。黄牛价大概7000多，几乎没有色差矫正）。



**配合手机使用的Daydream View头盔和控制器**

2016，号称VR元年，但是，风风火火看似声势浩大的开局，交出的成绩单却不尽如人意，很大程度上在于体验效果并没有达到预期。**为了兼容大量不适合vr的手机而选择了一种效果更差的方式。简而言之，这是一种体验效果与商业模式的动态博弈。**这种方式的是否可取，以目前看来， 我个人认为，还是要优先深耕技术与体验效果，再进一步考虑商业成本与盈利的问题，好的体验感一定会有消费者来买单。

**为什么Distortion Mesh方式效果要优于Google的实现方式？很简单，这就像3d渲染的两种方式，光线跟踪与光栅化。**众所周知，目前的实时渲染都是光栅化，再配合少量光线跟踪，纯光线跟踪的Real Time，三五年内看不到希望。而Distortion Mesh的方式，就是光线跟踪，根据设计好的镜片，做一个光线模拟，计算，生成各个顶点的RGB偏移。而Google的计算方式，希望通过一条公式计算匹配所有的镜片，在未来菲涅尔镜片烂大街的时候，会更显无力。以上是我个人判断。



上面简述了光学VR实现的一些基本原理，希望能够对对VR行业有兴趣的小伙伴有所帮助，更多的细节问题，如果大家有兴趣可以一起探讨。行业的发展需要技术的沉淀与市场的逐步认可与接收，**大多数人长期徘徊在“等风来”的时期。现在，风口已至，是否能够迎风直上，让我们拭目以待。**