

3D Eye Tracking Visualization

原项目怎么做？

1. **摄像头标定**：拍摄棋盘格并用 OpenCV 标定，获取摄像头内参。
2. **眼动追踪**：用 Face Mesh 实时定位虹膜点，通过针孔模型算出眼睛三维位置。
3. **虚拟世界构建**：以摄像头为原点、屏幕为窗口，在引擎中复刻现实场景并置入虚拟摄像机。
4. **离轴透视投影**：用 off-axis 矩阵对准窗口，视角随头动却不越界。
5. **项目演示**：展示靶心、僵尸和虚拟橱窗等效果，让画面“跳”出屏幕。

差异



我怎么做？

1. **摄像头标定**：初始化参数后根据深度的估计值与测量值的偏差手动调整参数，便于快速获取内参。
2. **眼动追踪**：与原项目基本一致。
3. **虚拟世界构建**：换用three.js框架实现类似效果。
4. **离轴透视投影**：与原项目基本一致。
5. **项目演示**：再现靶心场景。

差异



摄像头内参计算

目的： 获取本地摄像头的内参（主要是焦距） 用以后续计算虹膜的空间坐标

原项目的做法： 拍摄数十张标准棋盘格的照片并用 OpenCV 标定，获取摄像头内参。

步骤比较繁琐耗时，有没有简单有效的方法？

(观察公式)

焦距 = (图像宽度的一半) / $\tan(\text{FOV} / 2)$

我的做法： 随机初始化FOV，实时打印据此计算的深度，与用尺子测量的深度值进行比较，手动调节FOV至深度计算基本正确为止。

调整FOV至深度
等坐标计算基本正确

虹膜中心坐标 (相对于摄像头)	
X坐标 (cm)	0.96
Y坐标 (cm)	-1.7
Z坐标/深度 (cm)	56.04

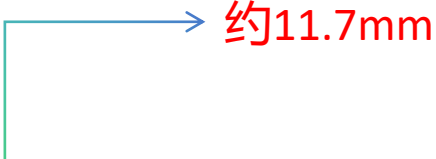
屏幕校准	
屏幕宽度(cm)	34.5
屏幕高度(cm)	19.4
摄像头到屏幕距离(cm)	0.5
摄像头FOV (度)	20

眼动追踪计算

1. 用MediaPipe Face Mesh 检测出虹膜边缘的关键点，计算虹膜在图像中的像素尺寸。

2. 利用针孔相机模型（相似三角形）

公式计算深度：


$$\text{深度} = (\text{焦距} \times \text{真实尺寸}) / \text{像素尺寸}$$

3. 同理计算摄像头坐标系下的虹膜中心点的 x, y 坐标：

$$\text{x 坐标} = (\text{虹膜中心点相对于图像中心的x坐标} \times \text{深度}) / \text{焦距}$$

$$\text{y 坐标} = (\text{虹膜中心点相对于图像中心的y坐标} \times \text{深度}) / \text{焦距}$$

虚拟场景构建

目的：以摄像头为原点，屏幕建模为矩形，用three.js精确建模。

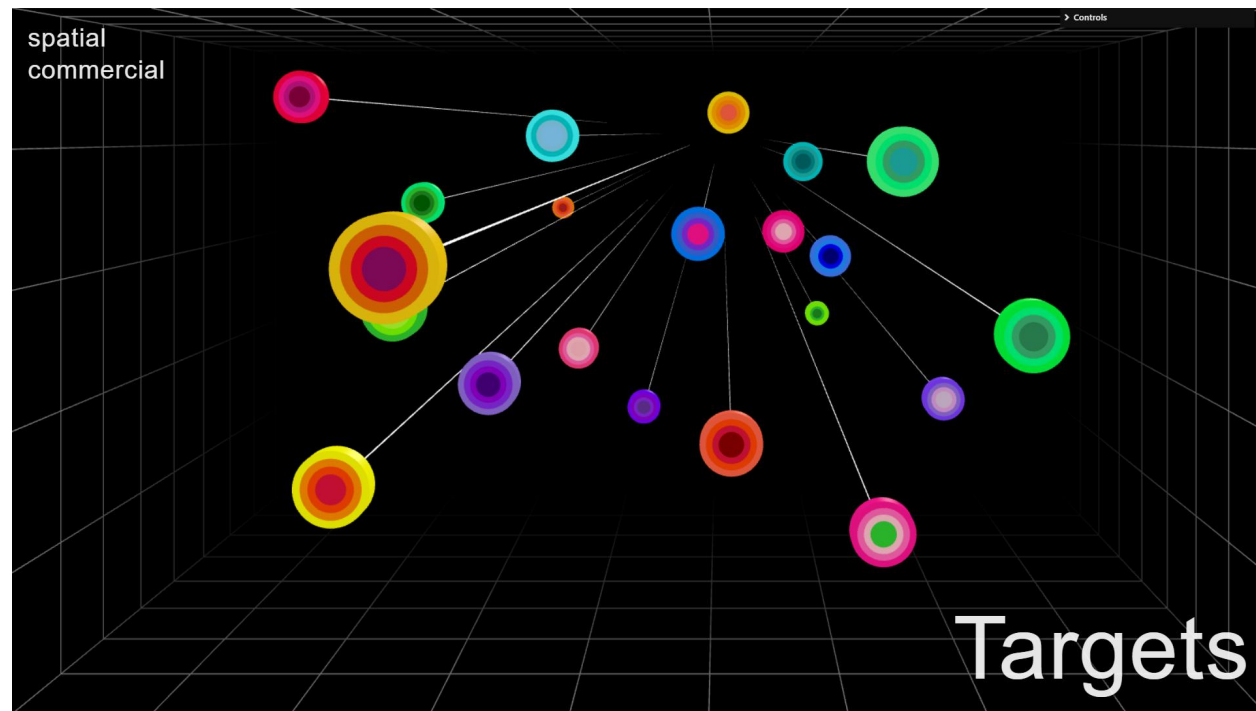
本机屏幕（长34.5cm，宽19.4cm）



1. 以我的笔记本的摄像头原点，创建虚拟屏幕，并将虚拟摄像机的位置与之前计算的虹膜中心点的空间坐标对应起来。

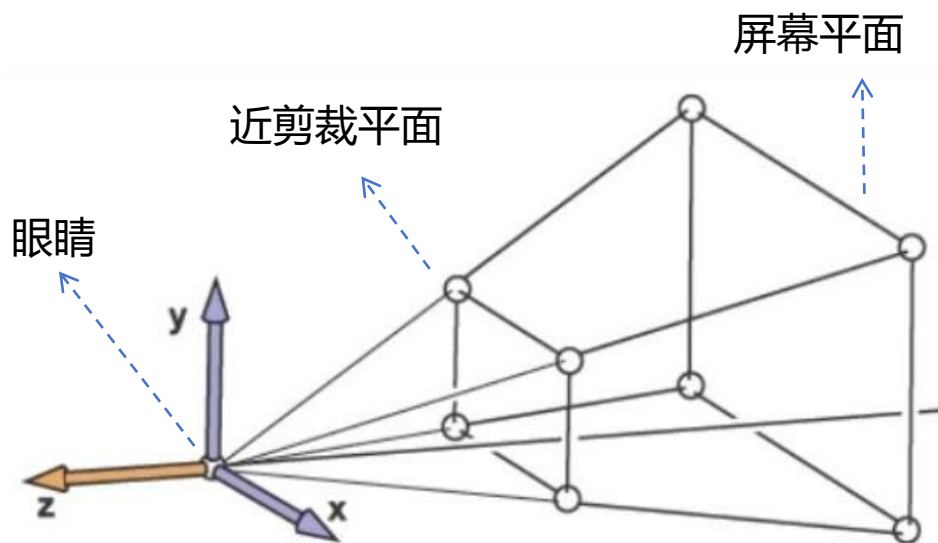
2. 尽量按照原项目的**场景复刻**，包括四周网格、原项目字样，同心圆环体、弹射动画等部分的**重现**。

重现后效果：

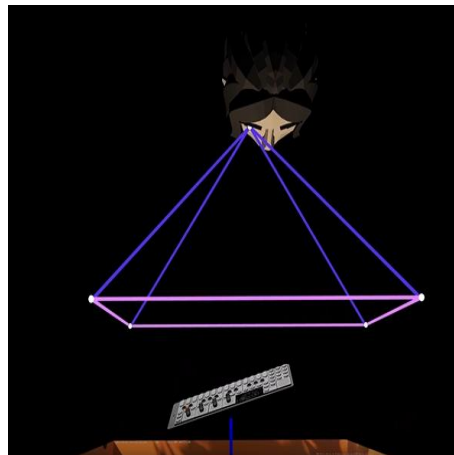


离轴透视投影

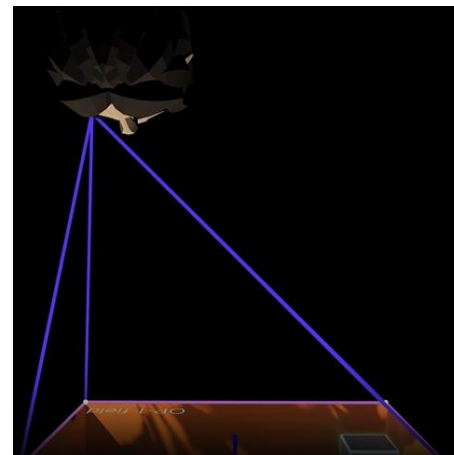
目的：使物理屏幕成为一个固定观察窗口，无论眼睛（虚拟相机）如何移动，3D景象始终精确填充屏幕。



标准透视投影



离轴透视投影

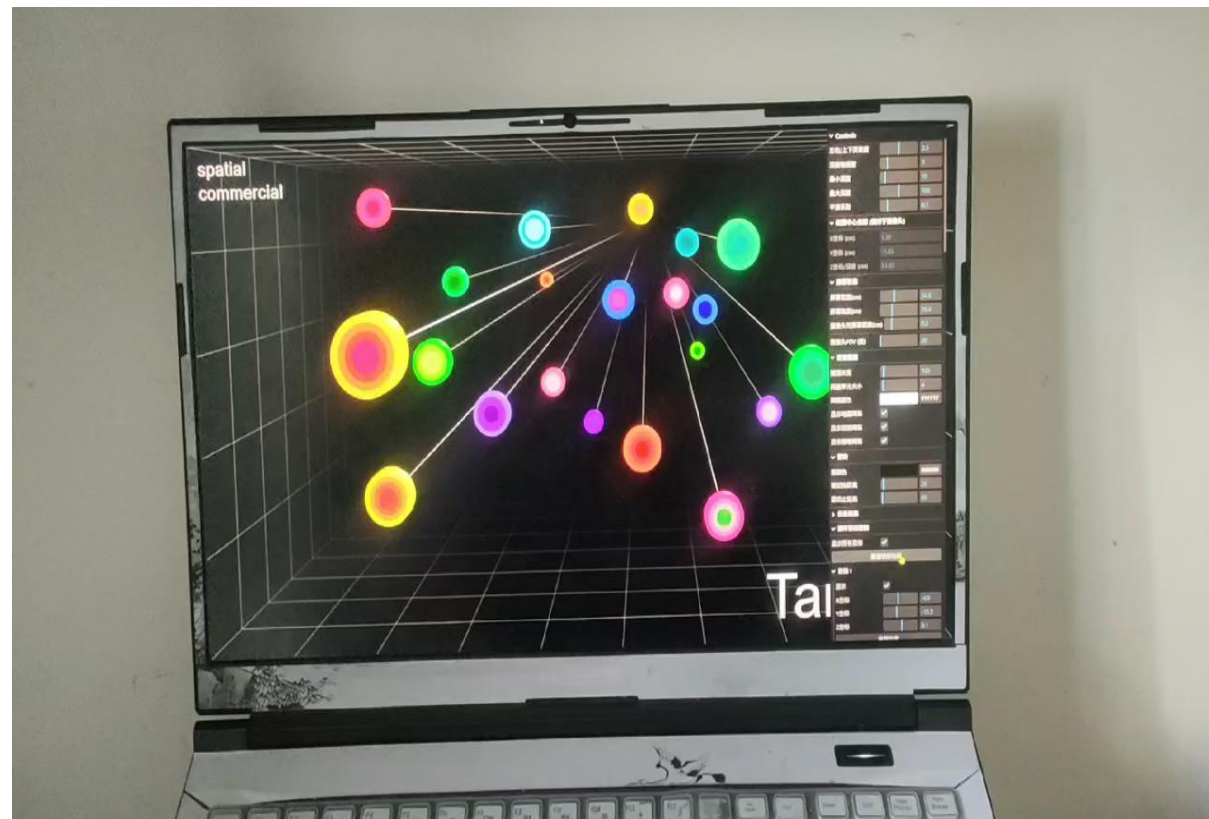


- 1. 计算非对称视锥体边界：**计算眼睛位置和屏幕四边的偏移距离，使用相似三角形原理将屏幕边界投影到近剪裁平面，得到**偏移的视野范围**。
- 2. 构建偏移投影矩阵：**利用非对称边界计算**焦距参数** (x_-, y_-) 和**偏移参数** (a, b)，构建包含离轴偏移的4x4投影矩阵，再使用新矩阵渲染场景。

原项目



本项目



注：这里的亮度差异受手机拍摄影响，实际圆环亮度与原项目类似。

补充与思考

1. 虚拟摄像头始终对应的是你**右眼的位置**，不管你是睁开双眼还是只睁开单眼，以此获得更好的视觉效果。

2. 过程中的主要困难概括为三点：

1. 摄像头参数标定繁琐 -----> 切换快速定参方法！

2. 原项目场景1:1复刻的困难 -----> 调整测试一种种可能的样式以逼近原场景！

3. 坐标的正负及各种计算以及容易出错 -----> 仔细反复检查调试！

3. 我提供了丰富的参数调控GUI，许多参数都可以根据自己的**电脑实际配置**修改，只需要一把尺子和简单的测量校准，就能较为准确地重现本项目的3D错觉，这或许能为这种技术的广泛传播提供一些见解与思路。

部分参数调控面板

