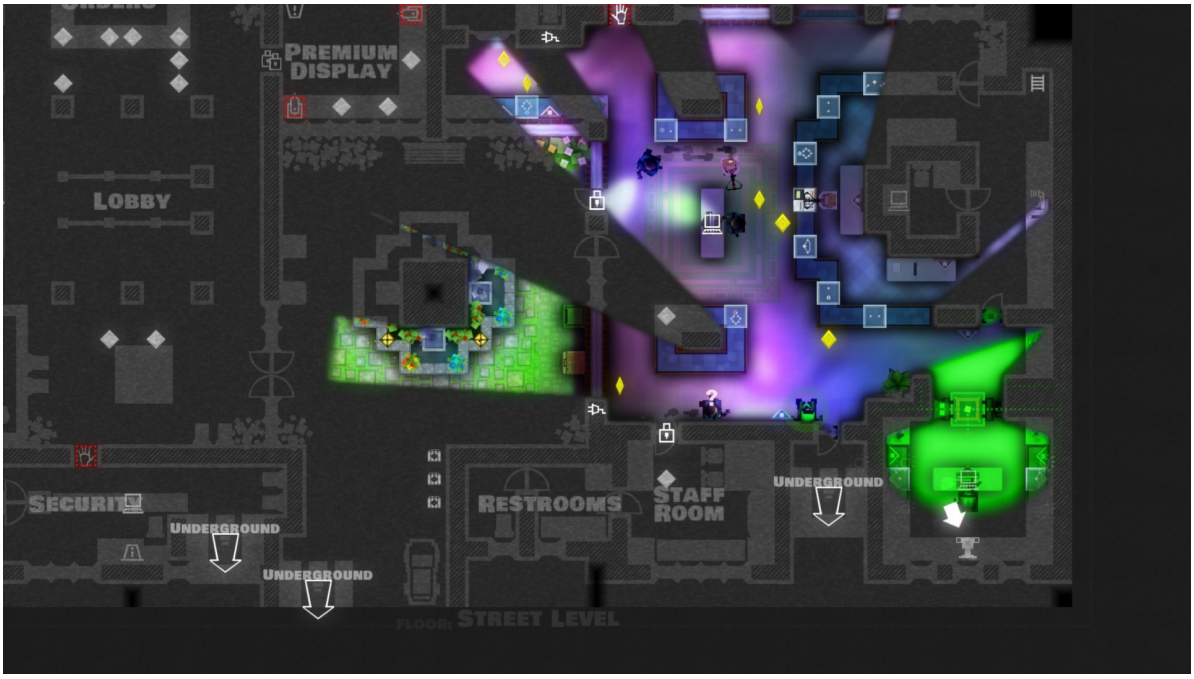
【Blog】灯光效果 in Monaco What's Yours is Mine

1. 效果截图



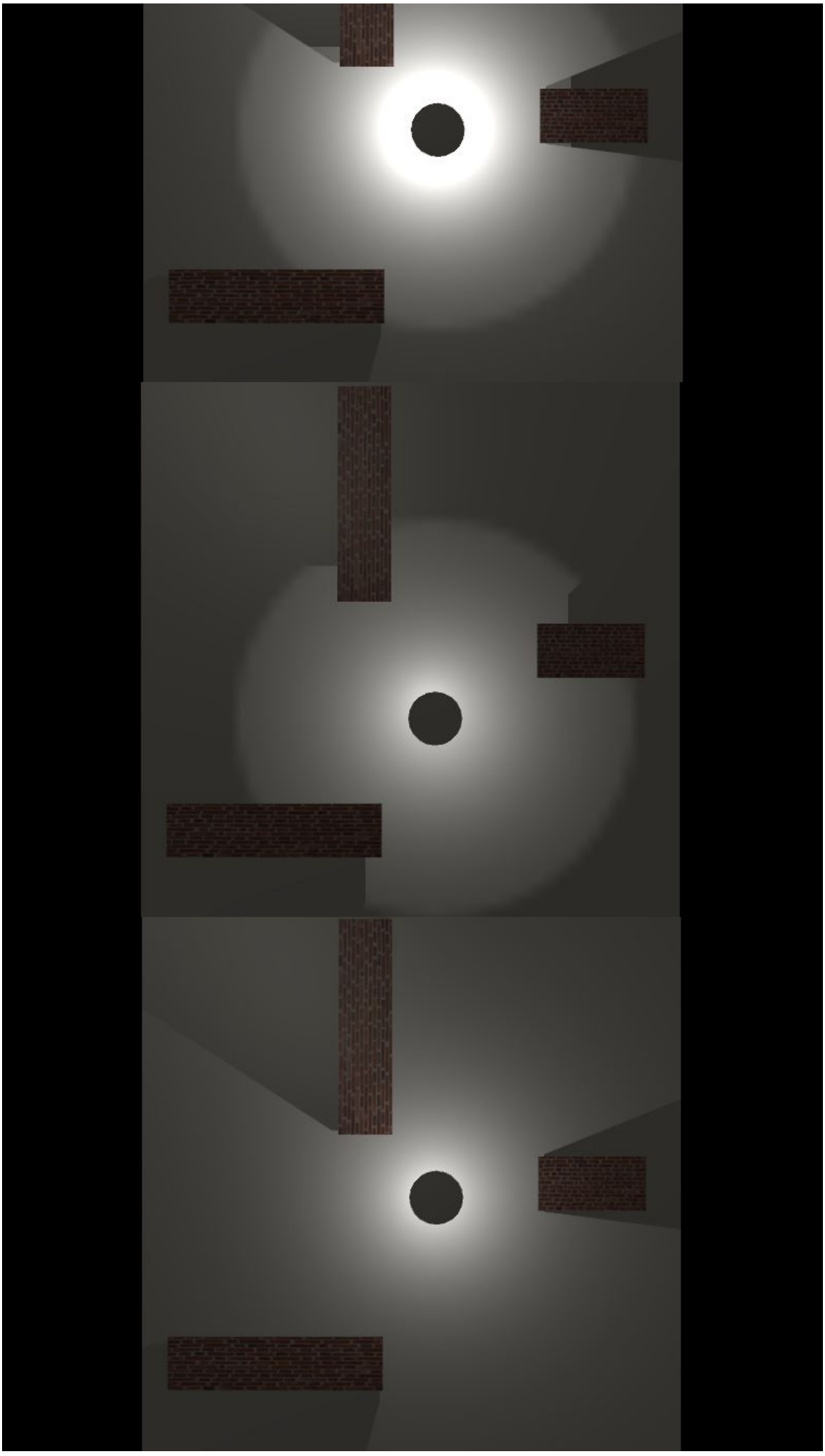
2. 观察

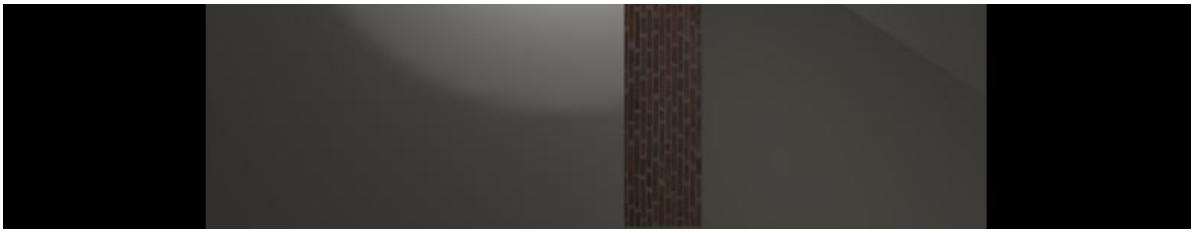
- 不随玩家角色朝向改变，而随玩家角色位置改变
- 整个场景偏暗，整个灯光看起来是一个以玩家为中心向外扩散的照亮场景的圆
- 亮圆的光无法穿透障碍物，所以形成了亮暗间隔的扇形区域
- 亮圆里亮的扇形区域可以被玩家和敌人同时观测到（不只是一个视觉效果，还是游戏性组成部分）

【思考】

- 实时阴影贴图的检测方法？
- 某种结构记录了可观测区域，此结构同时记录光照信息和游戏性信息

Unity 内建光源测试





【思考】

- 看起来是限制了范围的点光源
- 建筑物和角色都没有阴影，阴影计算全部用在这个大亮圆里
- 光线求交的方法得有多慢.....

3. 希望大家都会用搜索引擎.....

1. SIGHT & LIGHT 光线投射，然后连接点成多边形。附带 JS 演示。
2. Line of Sight in a Tile Based World 这尼玛看上去好像是开发者自己写的（果然提到了 visibility and lighting engine ）。

Line of Sight

Monaco 中的亮块即视点视线所及区域（line of sight），本质上是一个 triangle fan 。其构造过程如下：

- 每次物体移动到下一个 tile 才进行的计算

构造 "forward facing" edges 的列表。所谓 "forward facing " edges 是指物体可见的墙面（另一面不可见）。
按照距离物体的远近给这些 edges 排序

- 每帧都要进行的计算

从物体发射投影光线，每次发射都遍历 edges 列表找交点，因为列表有序，所以找到的第一个交点一定是最近的点。因为设置了一个包围形状，所以每条光线至少会和一条 edge 相交
找一条最近的 edge，它连着另一条 edge，这条 edge 又连着另一条光线...这样循环一遍就可以会回到原点，这样就能够形成一个 triangle fan了。
（最后果然，开发者说对这个听起来就很慢的算法做了 TONS of optimization ）

4. 切割一下流程

构造这个亮亮区域的步骤：

1. 创建某种数据结构，存储障碍物的包围边的信息。每次视点位置变化的时候，更新这些边与视点的关系（远近，可见）
2. 投射光线的机制，能找到最近的交点
3. 连接所有交点

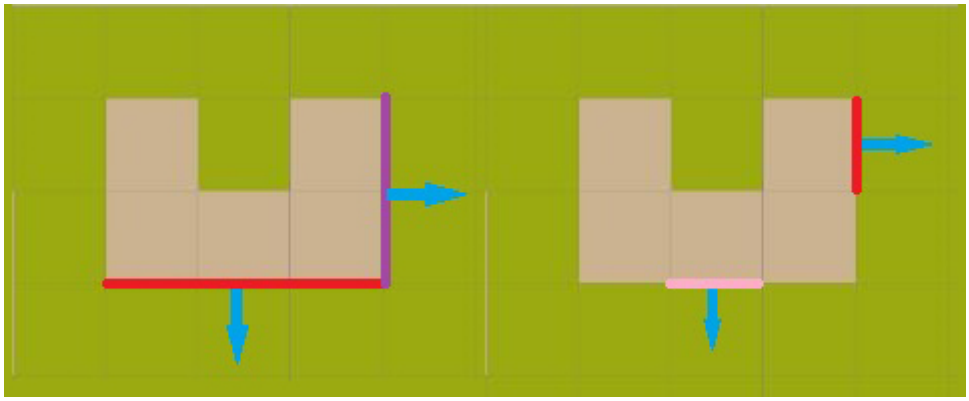
解决方案

1. 某种可排序的数据结构，其结点为正方形 tile，每个 tile 有 4 条边。根据这些边的法线与视点—tile连线的点积来判断 forward 和 backward。
2. 射线函数 ray (t) = origin + t * direction 与线段函数 segment (t) = start + t * (end - start) 求交，对每条射线都找到最近的交点
3. 连接所有点创建三角形集（MeshData）

5. Unity 测试

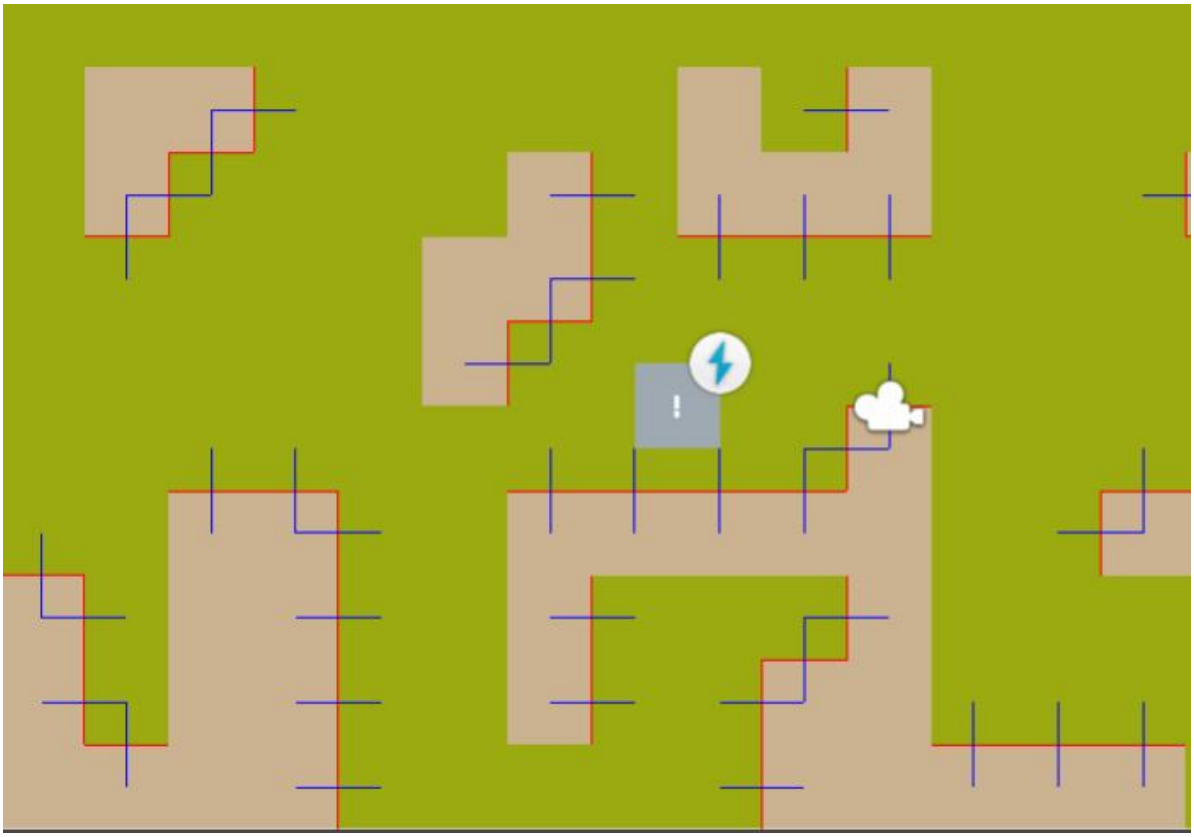
1. Forward facing edges
   我一开始想的是，对一个不规则的方块几何体，勾勒出它的所有外围边，每条外围边有一个线段方

程。不过很快我就意识到这尼玛太难，先要用搜索的方法得到形状，然后再切成长度不同的线段，而且只需要外围的线段!@#$%^&*@#$%^%^&再想想头就要爆炸了。于是我又想，为了降低复杂度，干脆让每条线段都是一个单位长，这样判断它是不是外围边就很容易（一边是墙一边是地）。这就好写多了。而且还很容易确定朝向法线（从墙到地）。



```
method FindEdges :
for (tile in wall tiles)
 for ( d in {right, down, left, up} )
   if ( d-tile != tile )
       find a new edge and its normal!
```

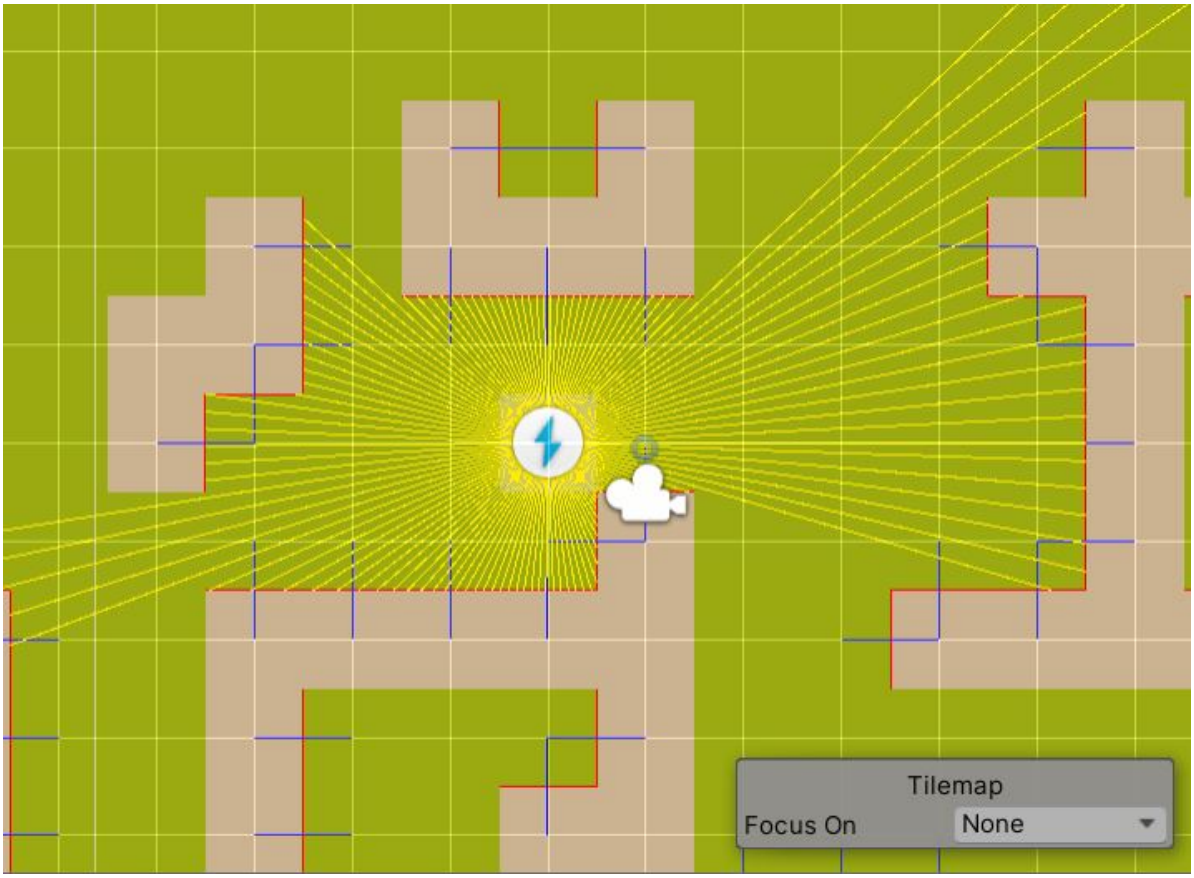
如下图所示，蓝色是法线，红色是要求的 forward facing edges 。



2. Projection

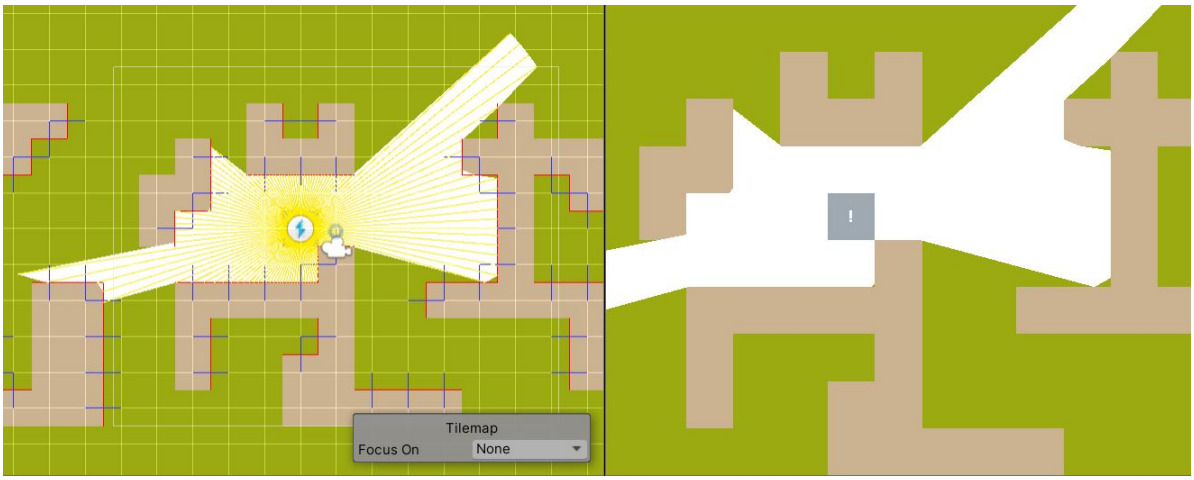
   从视点开始绕一圈，按照一定的频率发射光线，检测与各个 edge 的交点。因为 edges 是有序的，最先找到的交点肯定是最近。这里设置了一个边界距离，让光线不至于飞太远。

```
method FindIntersection :
for ( d in {0, 2pi} )
  solve equation {o + t1d = s + t2*e} to get t1 and t2
  if ( 0 <= t2 <= 1 && 0 < t1 )
      find the intersection and get a good projection ray!
```



3. Connection
   得到所有交点之后，依次连接成三角形就可以得到 triangle fan 了。

> method ConstructTriangleFan :
> for (p in intersections)
> construct a triangle by connecting <viewr_p, p+1, p>



代码地址

https://github.com/tandandanw/line-of-sight