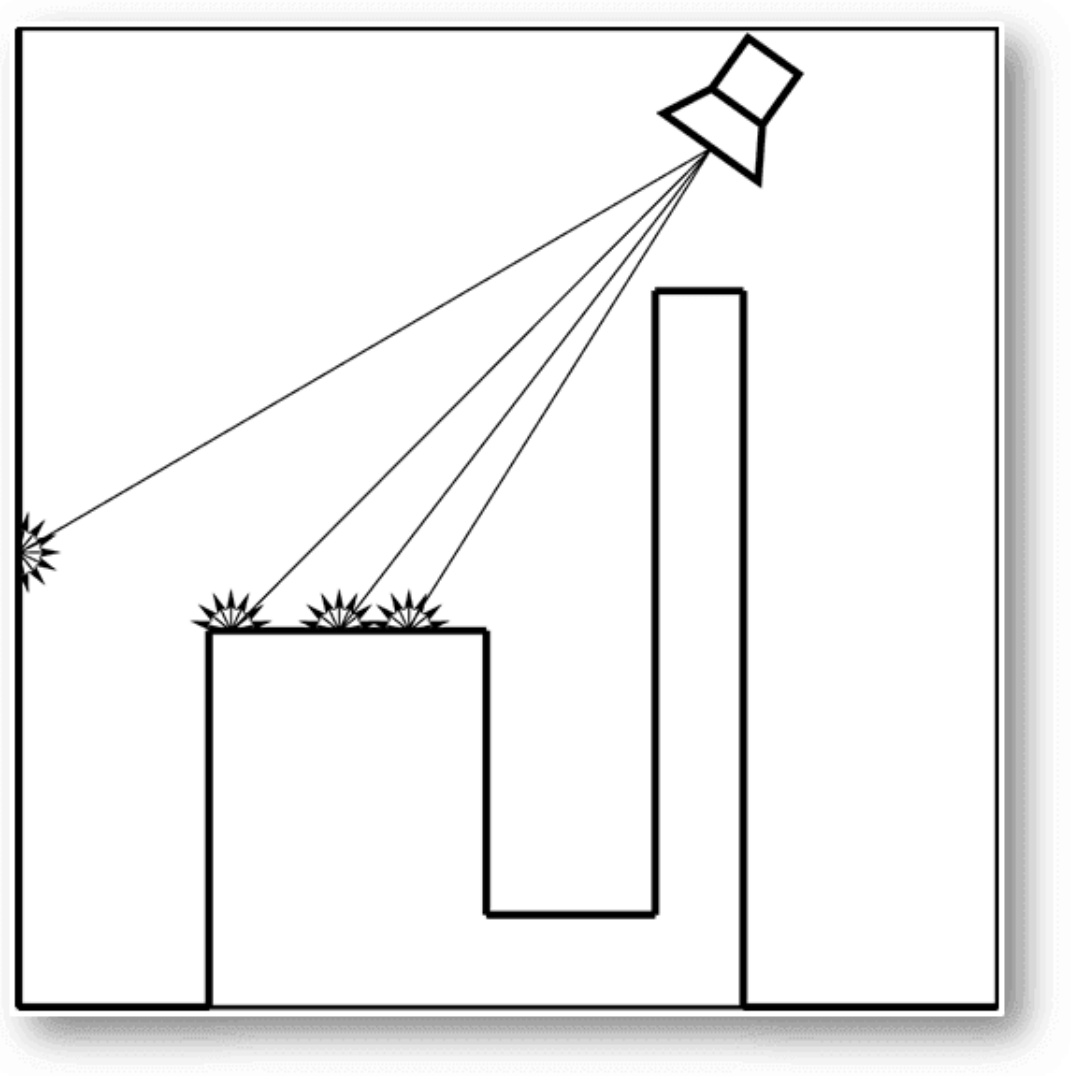
Light Propagation Volumes

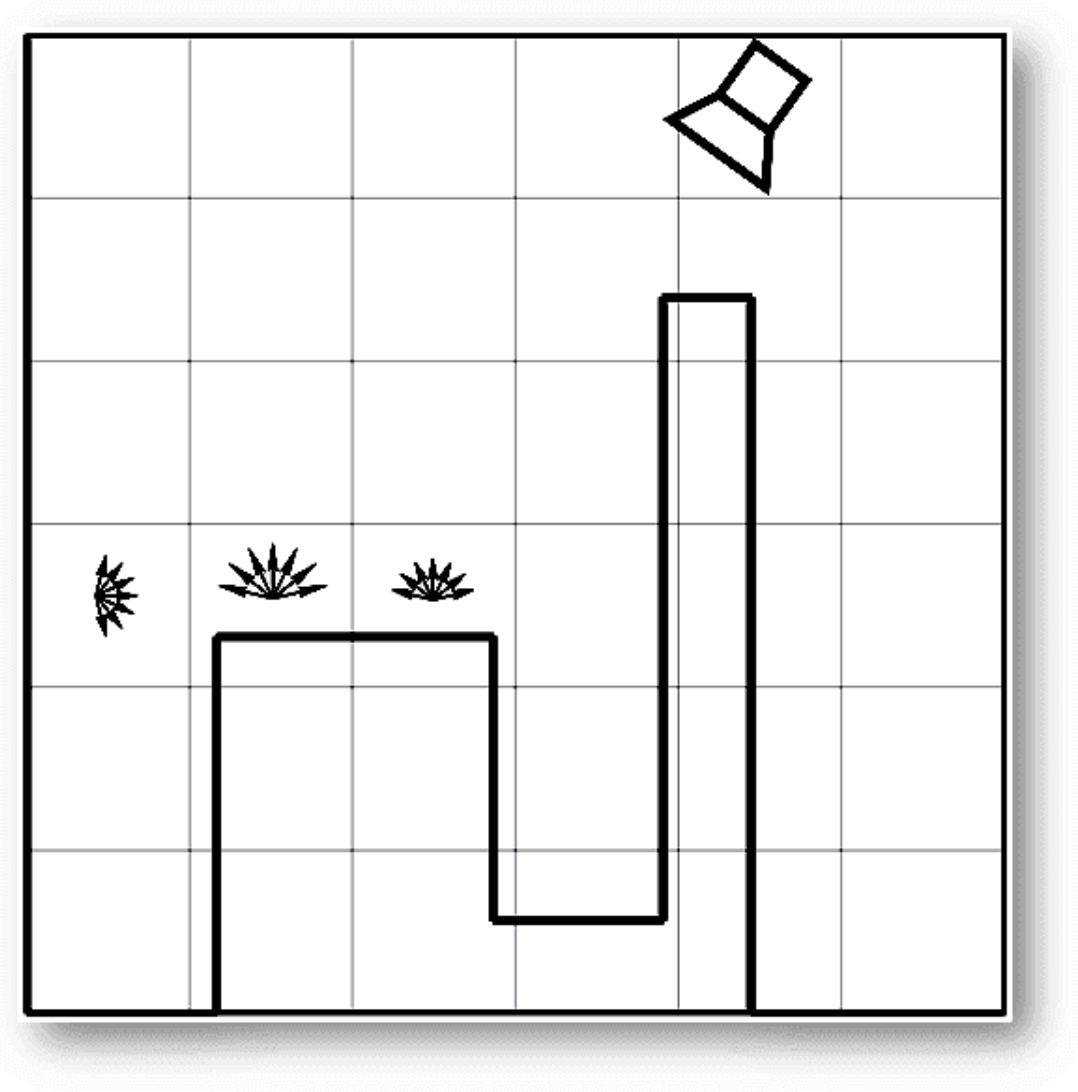
2017-04-28 By Ubo

说明:以下内容为群内陈述加讨论的形式。

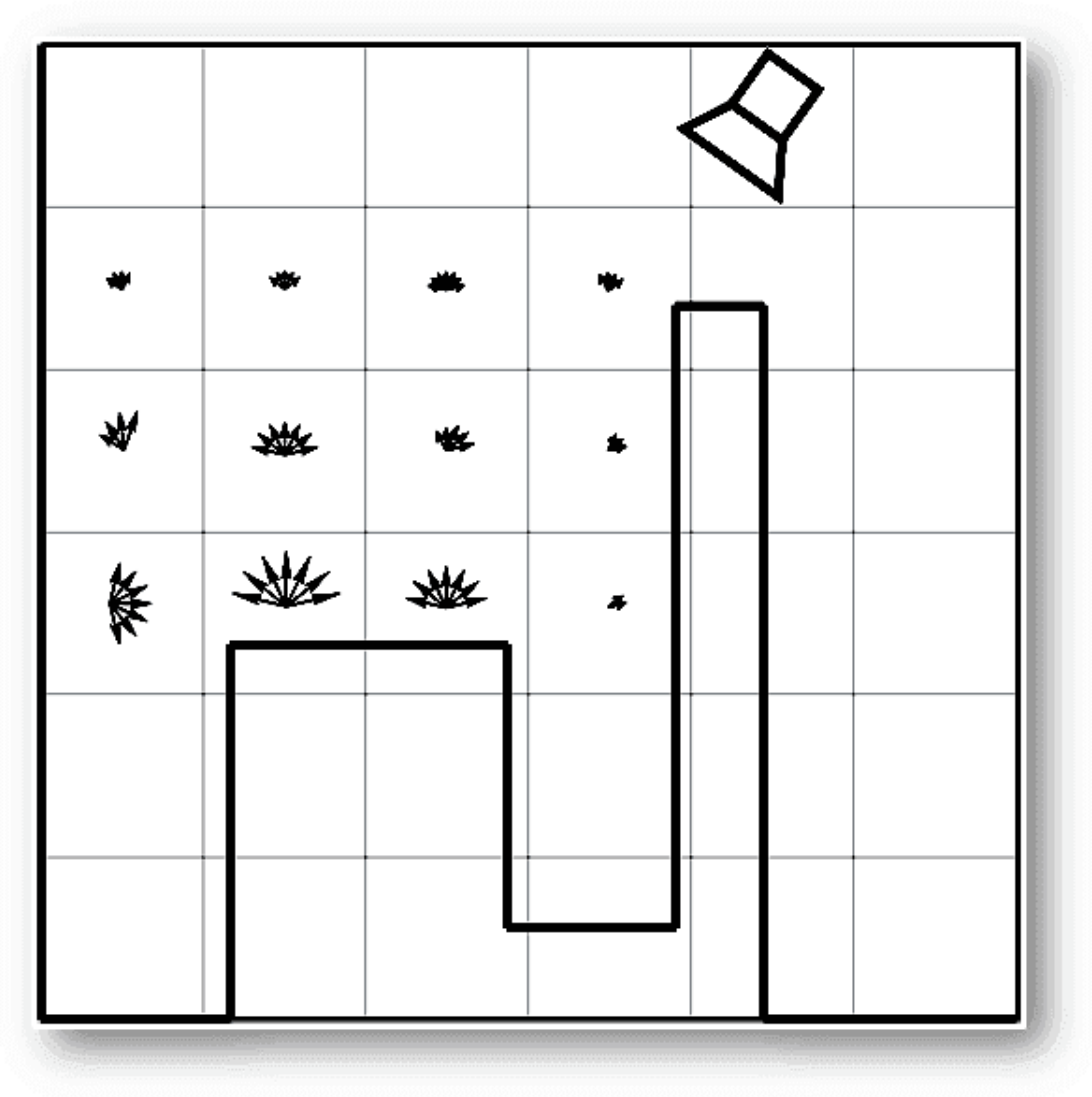
五一要去耍，没时间了，我提前说个pre算了。随随便便的LPV（光传播体）。
光传播体是用来做反射环境光的。思路很简单。



假设有个光源，点亮了场景中的facet。



这个光源其实一般指太阳光，不过图里面就随便拿个spot了，一样的意思。然后，这些反射光的facet，就成了新的虚拟点光源。



这些反射形成的虚拟点光源再去点亮场景中其他没有被直接照亮的facet。超级简单。

> 没有直接光照的facet？有直接光照的部分呢？ by 巽风雷

直接光+反射间接=最终结果。

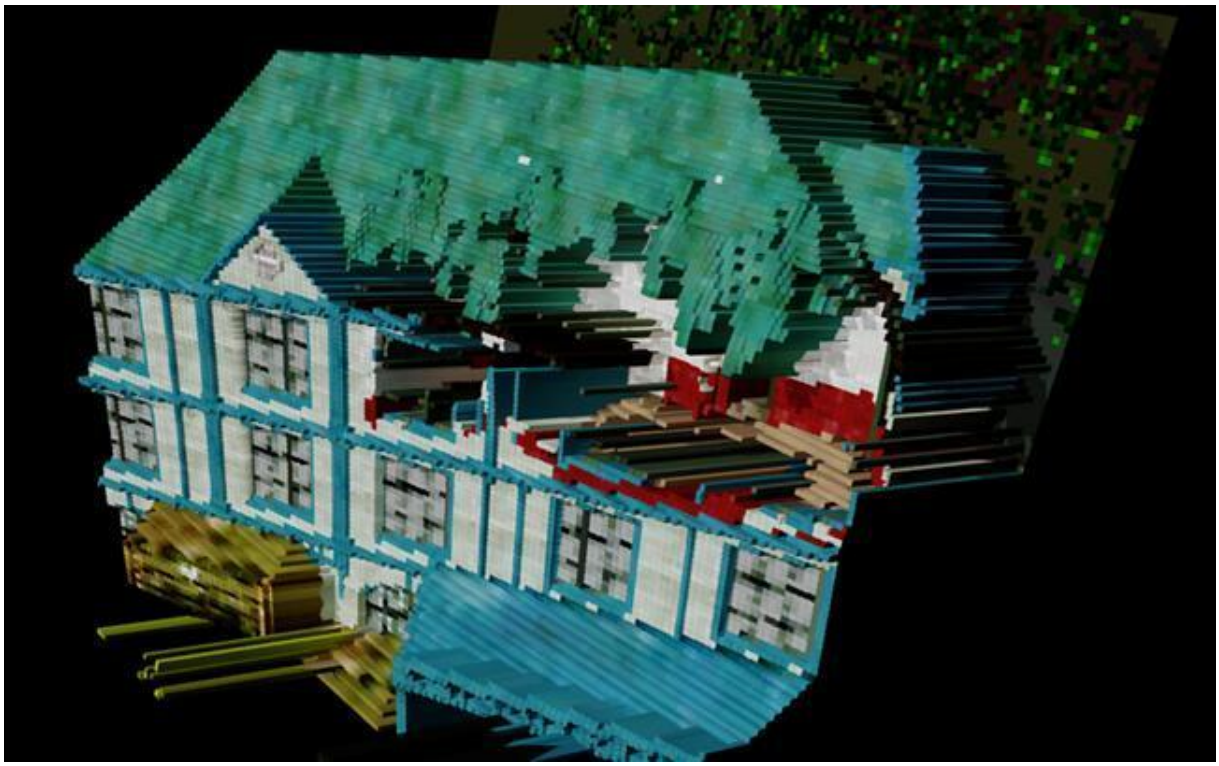
> 但是怎么布置vl，怎么p，还是挺tricky的。 by KAMUI
>
> 这个VPL的算法之前有做过。关键在于VPL的计算量。生成好VPL后，要计算VPL对场景的贡献。如果是正常的计算，相当于平白加了无数的灯。于是用mesh来代替，计算一个衰减的圆，用混合来进行照亮。中间又有裁剪等一系列的问题。最后效果不好，于是改用体纹理。voxel_cone_tracing还是有很多问题，体纹理的分辨率局限、带宽。最后直接让设计师补灯... by 巽风雷

接下来把问题分解一下。

1. 虚拟点光源的生成
2. 光衰减与传播

第一个问题，**如何生成虚拟点光源**。这个步骤叫做**vpl injection**。思路也是超级简单。

可以这样来获得vpl：考虑一个方向光，就是太阳光。把摄像机设置到方向光的方向上，然后正交投影，用类似shadowmap的方法绘制场景。



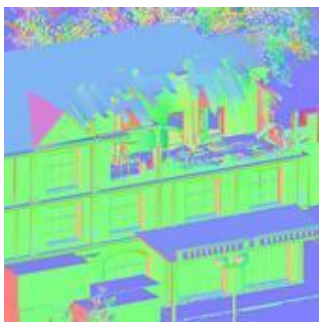
这个图，就是画出来的太阳光直接光照的“**shadowmap**”。这个shadowmap有个名字叫**rsm**，**反射shadowmap**。rsm有个特殊性质，也就是其中的每个texel，均对应着场景中空间位置上的flux。也就是说，我们用一个shadowmap，来记录被太阳光直接点亮的每个地方的光照强度。然后把对应的光照强度，转换成为虚拟点光源。再用这些被太阳光直接点亮的虚拟点光源，去照亮那些没有被太阳光直接照亮的角落。

注意上图中，有被太阳光直接照亮的地方，更有没有被太阳光直接照射到、但却被太阳光反射照亮的地方。而这些被太阳光的反射照亮的地方，实际上是被虚拟点光源所照亮的。这个叫**单跳间接光照(single bounce)**。

好，基本概念有了。具体讲rsm的实现。实际上, rsm实际上包括三部分。

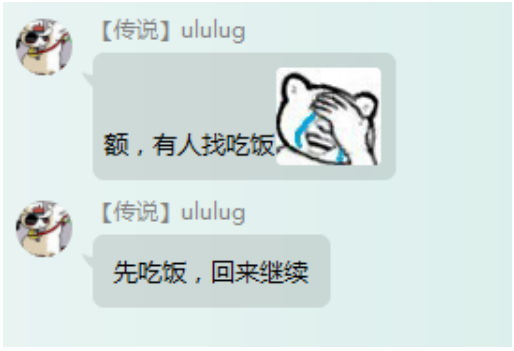
depth(深度)

根据depth深度可以计算得到shadowmap中每个texel所对应的world位置。



normal

albedo



十年后...

接着albedo讲。为啥需要albedo？因为有了光源本身的光照强度、光源方向、texel所对应的facet方向，再加上albedo，就能够算出texel所对应的facet在反射方向上的flux。根据rsm的depth、normal、albedo，外加光源属性，可以计算出rsm中每一个虚拟点光源的光照分布和强度。一般rsm的texel大小为1024 x 1024，共计1024^2个虚拟点光源。

> rsm里点的光照还是延迟到后面做是吗？因为存了albedo，不是光照结果。法线要存么？ by 星剑

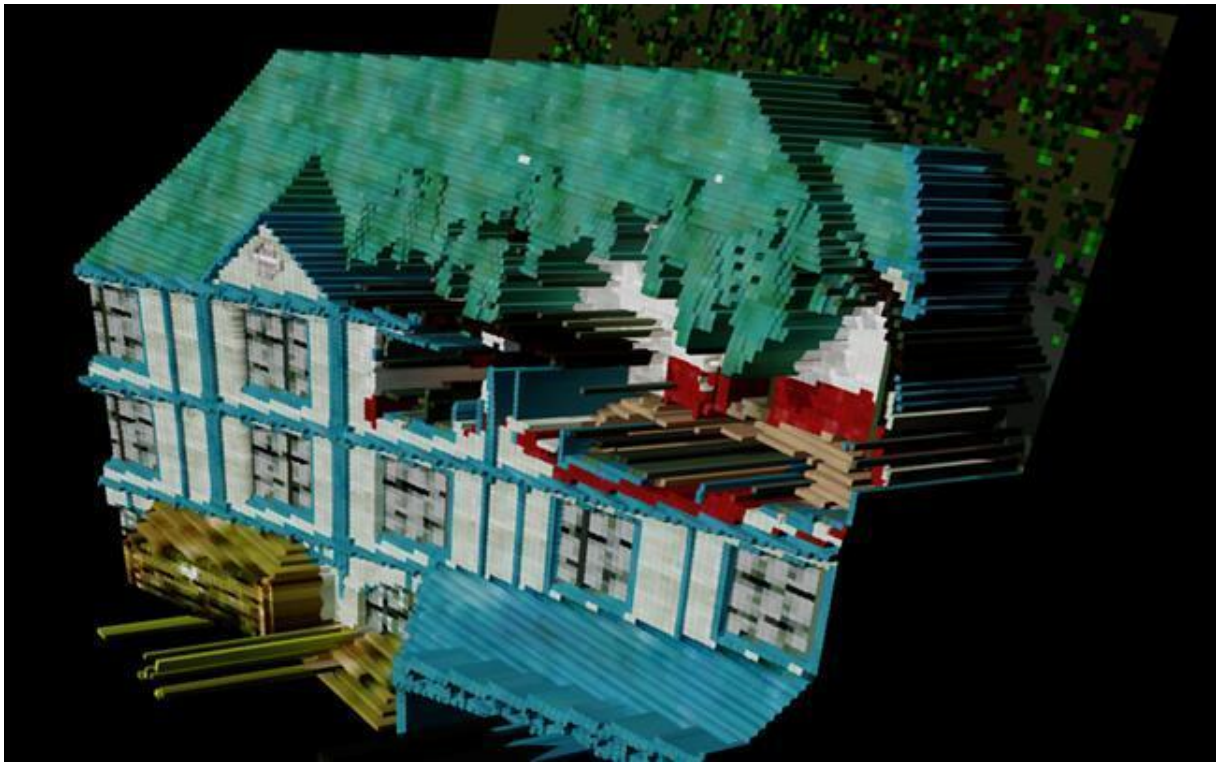
rsm=depth+normal+albedo。

> 每个虚拟光远的lobe都是半球么？ by uncle tuu-GameDev

不是半球，近似成cos lobe。

> 和tsm有点像。tsm是irrad、normal、depth。 by 星剑

其实也可以直接存flux，但是这个值比albedo打，不利于gbuf存储，所以一般动态算的多。这个主要是考虑值域定量。

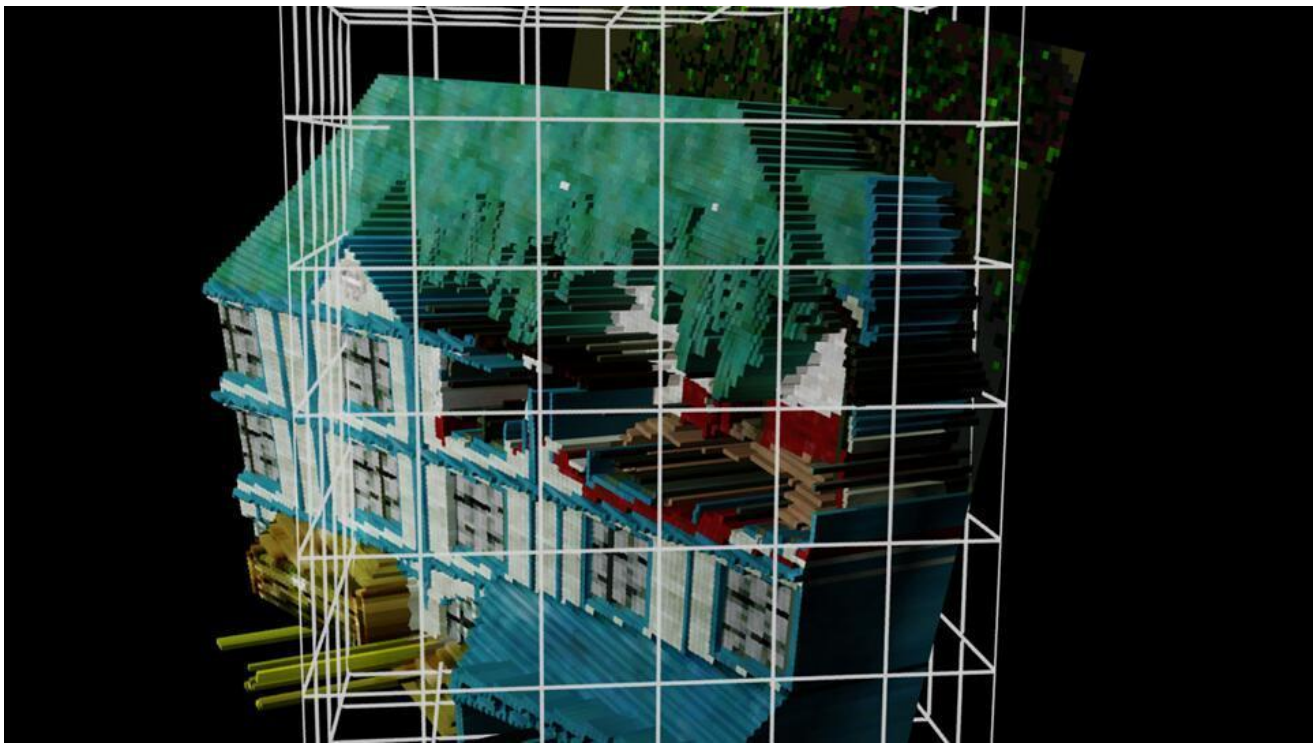


这个图，就是个rsm的图，注意其中的texel化的表示方式。

为毛lpv间接光效果还不错？1、rsm的生成效率高，2、rsm分辨率越大，间接光效果越好。相当于虚拟点光源越多。以上是第一个问题：虚拟点光源如何生成。

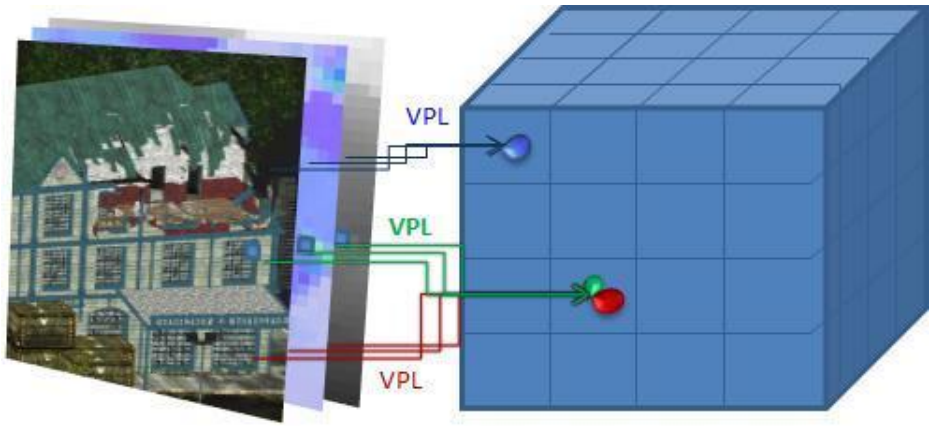
接下来解决第二个问题：**有了数量庞大的虚拟点光源，如何快速计算虚拟点光源对某个被有被直接照到的地方（surfel）的光照影响？**针对场景中的某个surfel而言，很显然的，不能蠢到在每次计算这个surfel的光照时，都去遍历一遍1024x1024个虚拟点光源，然后sum。

所以首先我们的想法就是对虚拟点光源进行clustering。最简单的做法，就是把整个场景分割成网格（grid），然后把某个网格中所有的虚拟点光源，全部合并成为一个虚拟点光源，每一个网格里面的格子对应着一个点光源。

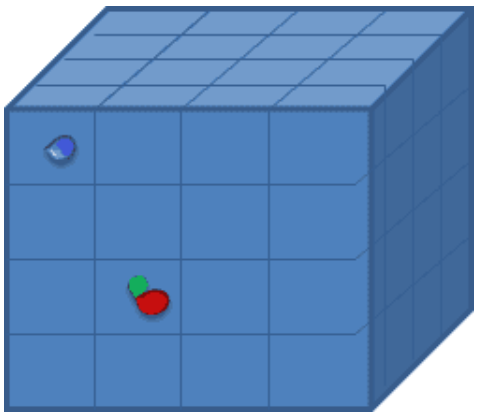


打个比方，我把整个场景分割为10x10个网格。注意到每个格子里面都会包含rsm的部分，也就是每个格子里面都会包括若干个虚拟点光源。但是这个虚拟点光源所在的位置并不同，怎么合并呢？于是在这里我们做了个离散化假设。假设格子所包含的所有点光源，都位于格子的中心处，然后吧所有包含了的点光源简单相加。

简单粗暴的离散化处理，但是works。因为Ipv我们更多的是要解决低频率的间接光，不是高频率光，所以微小的位置偏离无所谓。



然后就得到了这个图。这个图就是上面我说的那个过程。之所以要离散化聚类，是为了后继间接光的计算效率。得到了下图

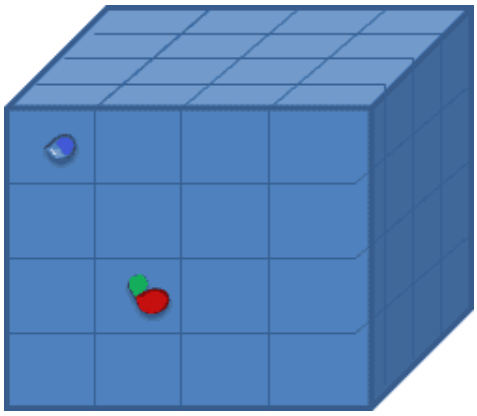


这个结果之后，**light(lpv) injection**步骤就算完成了。

> 这个有分层结构吗？还是就是连续储存？　by 大白

分层属于扩展，后面再讲。目前讲到的这个就是个连续存储。

第二阶段，叫做**propagation，光传播阶段**。为什么要传播，先来思考下这个问题。



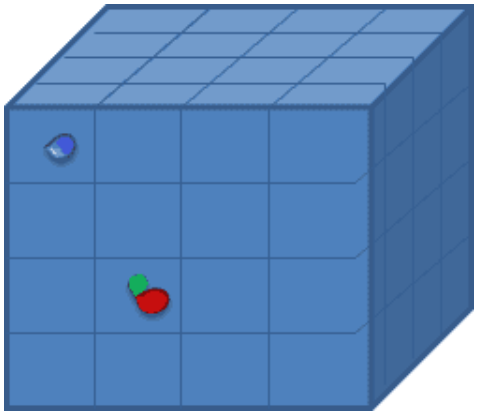
这个问题等同于：有了一个网格，以及格子中对应的直接光照得到的近似的一个虚拟点光源，就能算间接光了吗？问题的答案显然是不能，不是不能，而是不完美。注意上面这个图。其中存在着不少没有包含虚拟点光源的格子。这些格子，是不包含直接被太阳光照亮的场景surfel的。那么有两种做法，来计算这些格子里面的表面光照。

1.对于每个格子里面的表面，计算网格中所有合并之后的虚拟点光源对其的贡献，然后加合。这种做法问题在于a) 效率，b) 不考虑遮蔽。实践中上面这种做法现在是有人使用的。但是效果不好，因为没考虑场景里面的遮挡关系。而且，如果网格分辨率很高，那么计算所有其他网格虚拟点光源影响，这个运算量实时场景下不可接受。接下来考虑第二种做法。

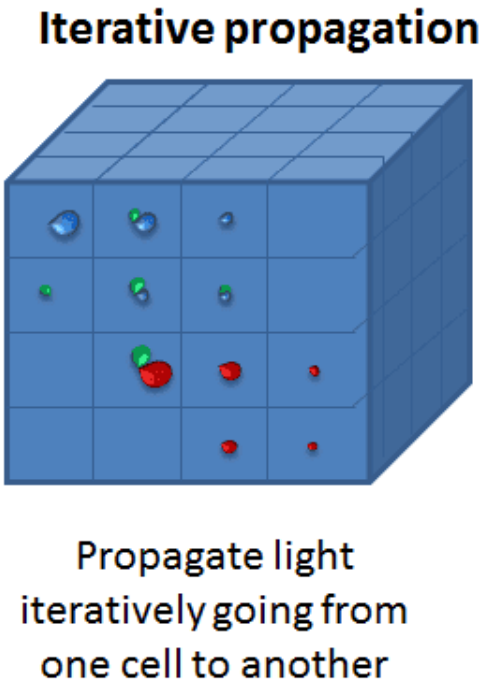
2.还是根据上面的图，考虑格子与格子之间的相邻关系。能否求得与这些有虚拟点光源的网格直接相邻的网格所拥有的光照情况呢。答案是可以的。实际上，我们可以假

设每一个格子里面都保存有一盏虚拟点光源。如果我这个网格划分是**512x512**，它包含了整个场景。那么实际上我用512x512盏位于各个格点中心的虚拟点光源，就可以代表当前场景在光线传播能量平衡状态下的一个光照环境**snapshot（瞬时光场）**。

已知：

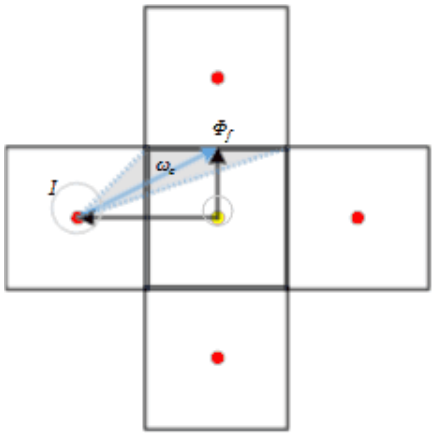


求：

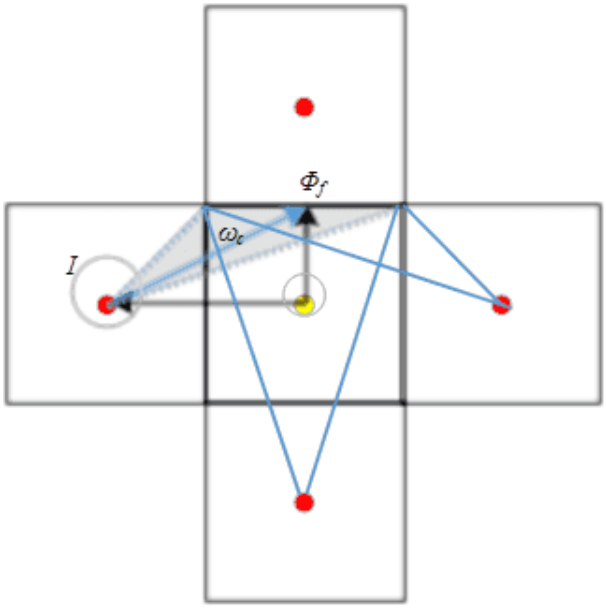


用语言来描述下这个问题，就是：如何求任意格点因周边相邻格点光照传播所导致的自身的光照分布。其实想法也很简单。考虑某个格子，其实这个格子就是个cube，6个面，对否？那么这个6面cube，最多能与多少个其他格子相邻呢？答案也是6

所以，实际上这个问题就是个gathering的问题。等同于：我在跟师妹秀恩爱，但我周边有6个单身狗师弟当电灯泡。请问我身上感受到的不爽有多少...这个过程就要画个复杂图了。



只画了个2D的情况。其中黄色是我要求的虚拟点光源，红色是已知的相邻虚拟点光源。这个黄色的要求的虚拟点光源，可以照亮它所属于的格子的4个面，对否？（先不考虑遮挡，后面会讲）所以首先让我们来求其每个面所获得的来自相邻格点虚拟点光源的flux总和。

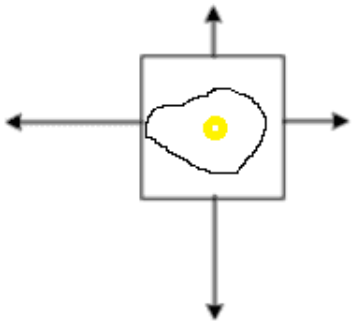


比方说，上面那个面。它能够gather到的flux，分别来自左、右、下，三个相邻虚拟点光源，对否？在2维格点情况下，一个格子有4个相邻的虚拟点光源，所有我们能够求得这个格子所拥有的4个面从相邻虚拟点光源处所收集到的flux。

在三维grid划分情况下，一个格子有6个面，所以我要gather 6次。而在图中的2D grid划分下，一个格子有4个面，每个面gather一次，所以我要gather 4次。有了这四个面

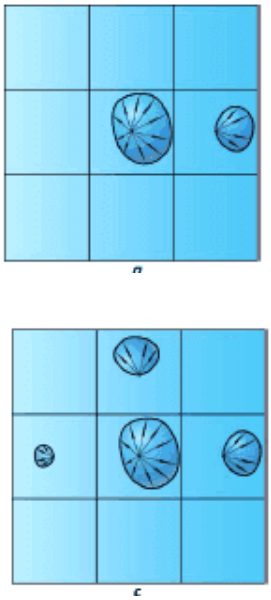
的flux，于是我就可以重建出黄色的虚拟点光源究竟该是一个怎么样的光照分布了，这个叫做**reconstruction**。

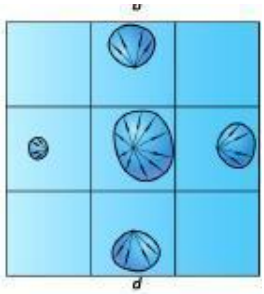
这里随便画个图



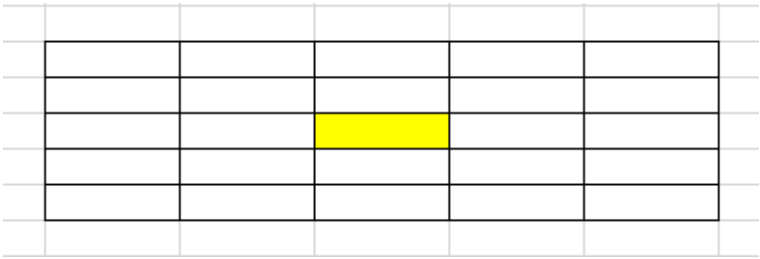
比方说，4个面，箭头表示flux投影方向，箭头的长度表示flux的大小。那么让我们用位于格子中心处的一个完全不存在的虚拟点光源来等价描述这四个面的flux分布。注意，这个是等价描述。也就是说，用一个不存在的虚拟点光源，来等价描述4个面的flux分布。结果就是上图这么个包络。这个是lpv光传播最关键的一个步骤，**等价光场重建**。对于6面的3D cube，已知6面flux，也可以建立里面3维的包络体。

前面我具体讲了相邻格子向单个格子的光传播计算过程。接下来再来看整个网格，那么我们就可以对整个网格里面的每一个格子，都来计算它们相邻的格子的光传播光照影响。

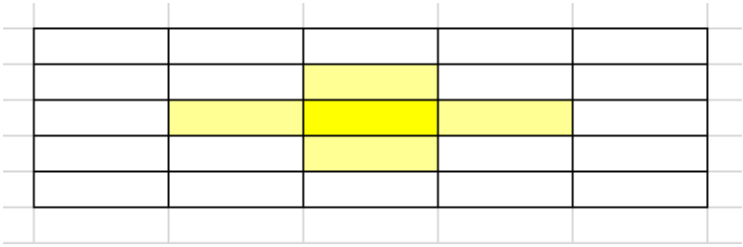




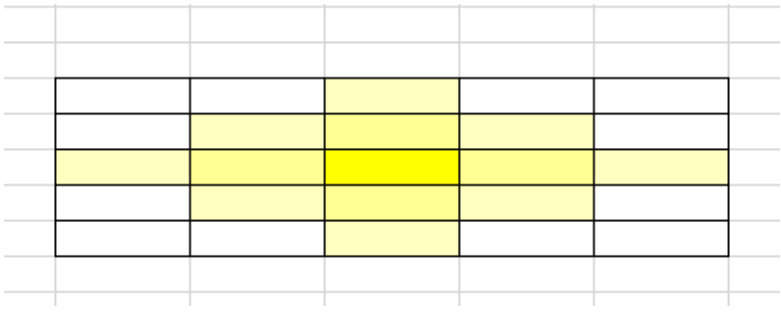
上面这几个图，就是逐一重建每个格点中所包含的虚拟点光源的过程。这就是个迭代的过程。打个比方



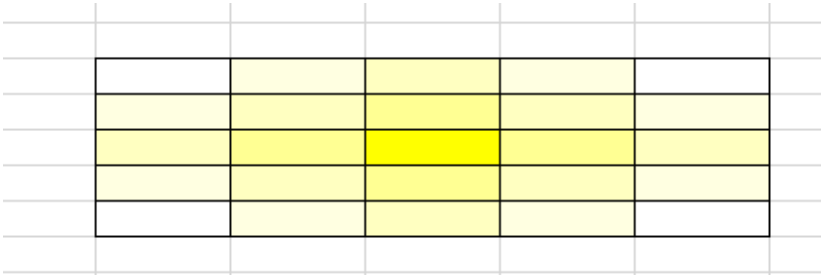
这是个5x5格点场景划分，我已知最中心处的虚拟点光源分布。然后，第一次迭代，得到



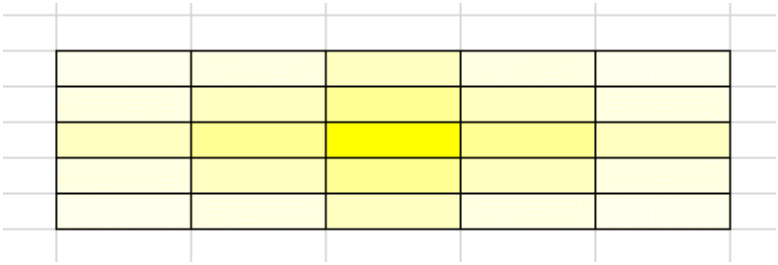
得到周边相邻的4个格子所对应的虚拟点光源。第二次迭代，得到



注意，我简单用颜色深浅表示光传播的强度衰减了。第三次迭代，得到



最后一次迭代，得到



至此，所有格点的等价虚拟点光源分布计算完毕。总体来讲，上面这个过程，就是一个“被太阳光直接照亮的区域不断地去间接照亮那些没有被太阳光直接照亮的区域“的过程。

上面的迭代传播过程，有问题否？

> 迭代停止条件？ by Nihil

迭代停止条件：所有网格的等价虚拟点光源计算完毕，迭代即可停止。一般实践中不这么做，所以一般取固定的迭代次数。比方说，迭代3-4次，停止。不管每个格子里面的等价vpl是否均已求得，都停止。实际上迭代次数就决定了间接光能够照亮的距离。迭代次数越多，间接光能够照射到的范围就越远，否则就越近。

> 每个光源一个grid吗，还是都放到一个grid里一起算？ by 佟

一个grid里面存所有光源。然后迭代计算的结果存在一个新grid里。然后两个grid来回相互倒。离散光diffusion。最终，我们得到了一个grid，里面每个格子都对应着一盏虚拟点光源。那么我们最终在绘制整个场景时，就非常简单了。

在绘制某个fragment时，只需要考虑其所处的格子即可，然后对应的把该格子的虚拟点光源取出来。然后计算该虚拟点光源对fragment的光照贡献。只需计算1个虚拟点光源的光照即可，就可以求得整个场景对该fragment的整个间接光照影响。只要有了光传播场，计算间接光就是分分钟。

> 相当于我算光照的时候把主光源光+上面的间接光 然后加其他场景光？ by Nihil

yes。直接光照+虚拟点光源的光照贡献。

> 间接光照的位置永远放grid正中间？ by Nihil

对，这个是离散化光场传播的必要处理。实际上你从结果上根本看不出啥不对的地方。

室内也是一样的。wrap it up，总结一下：以上我们使用的简单方法，包括的步骤有：

```
injection -> iteration {  propagation } -> rendering
```

首先是直接光照注入，然后是迭代传播，最后是渲染。以上计算方法不涉及体素化。

以上就是基本的lpv算法。好吧，接下来，any question。

Q&A

> 主光源变化怎么处理？ by Nihil

主光源变化的话，要重新生成rsm，重新注入。然后，重新计算光场传播，就这么简单。并不是每帧都需要更新光场。只要主光源不变，格子就不需要重新计算。rsm生成、injection、迭代传播，可以做到全实时。而且也可以做到摊销迭代，每帧我迭代一两回，结果就是你会发现间接光越来越真实。

> 这个lpv的重建开销大吗？ by 小朋友

重建开销取决于：1）格子划分的分辨率 2）迭代的次数。这个参数可调，根据场景复杂度自行调节。

> 动态物体也会影响广场吧，需要重建吗？比如一辆车开过去。 by 佟

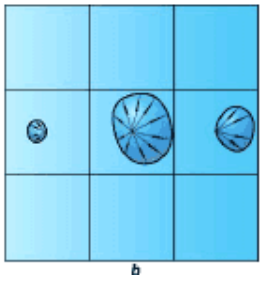
动态物体一般实际使用过程中，不参与lpv计算。动态物体每次移动都可能导致光场重建，代价过大，不推荐。不过你一定要做的话，不惜代价的话，也可以。

> 那么，3D的光信息存储在哪呢？ by 巽风雷

存纹理里面也行，存out of core也行。存储机制可自行选择。

> ubo，你没有讲格子有阻挡的情况呀。 by 小朋友

嗯，本来我打算偷懒不讲的。说说格子阻挡的问题吧。



看这个图。实际上，遮挡问题相当于是在问这么一个问题：光照能量从一个格子传播到另一个格子，需要衰减多少？在我们离散化的处理方式中，这个问题超级简单。不

就是6个面相邻吗？我们把遮挡问题简化为6个遮挡系数就可以了。这个遮挡系数专门用来在计算光传播的过程中衰减能量的。实际中这种处理方式简单有效。因为lpv计算的是大面积的间接反射光。

上面这6个遮挡系数，理解了否？

> 这个遮挡系数怎么来的？ by Nihil

嗯。。。。好问题。这个问题就留做作业。大家回家想想，这6个面的遮挡系数该如何求

这个遮挡问题，我就说在这里。还有其他问题否？

> 我记得ubo发过一张图2ms，我想知道是什么显卡。 by 大白

这个算法过程并不需要很牛逼的显卡。里面最耗的过程就是虚拟点光源的迭代重建而已。然而这个过程我不用一帧算法，我分成10帧。分摊之后，你根本感知不到光传播延迟。rsm其实并不耗时，如果你光栅化分辨率不太大的情况下，速度是超快的，但确实耗费DC。耗费DC这个是所有shadowmap的普遍现象了。一般的，基本上都是一个主光源，一张RSM。这个代价实际其实很小。想削减DC，只有靠裁剪，不过这个已经不是LPV讨论的范畴了。

> 要画shadow，必然重画场景，这个免不了。 by 星剑
>
> 多光源就要多张rsm吧？ by 漆黑の牙

对，lpv之所以吊，主要就是rsm虚拟点光源数量足够多。只对主光源做即可，次光源间接还有很多其他方法。

我随便2048x2048一张rsm。显存么，我从来不care

苦逼手游党就别关注这些图形技术了，老老实实转行卡通美工吧。

完