# Games202 高质量实时渲染笔记lecture 06 Real-Time Environment Mapping 02

本文是闫令琪教授所教授的Games-202:Real-Time High Quality Rendrting学习笔记的第六讲 Real-Time Environment Mapping 02，本人属于新手上路暂无驾照，有错误欢迎各位大佬指正.

[GAMES202-高质量实时渲染_哔哩哔哩 (゜-゜)つロ 干杯~-bilibili]

本课目录:


我们在上节课讲述了如何不采样去计算不考虑shadow时的shading值,那么在有了环境光照情况下如何去得到物体被环境光照射下生成的阴影呢?

严格意义上来讲,这是不可能完成的事,因为以目前的技术来说是很难实现的,要从两个考虑角度来说:

1.many light问题:我们把环境光理解为很多个小的光源,这种情况下去生成阴影的话,需要在每个小光源下生成shadow map,因此会生成线性于光源数量的shadow map,这是十分高昂的代价.

2.sampling问题:在任何一个Shading point上已知来自正半球方向的光照去接rendering equation,最简单的方法是采样空间中各方向上的不同光照,可以做重要性采样,虽然做了重要性采样但仍需要大量的样本,因为最困难的是visibility term.由于Shading point不同方向上的遮挡是不相同的,我们可以对环境光照进行重要性采样,但一个SP周围的visibility项是未知的,因此我们只能盲目的去采样(我个人对盲目采样的理解是,为了确保准确性需要对sp各个方向的遮挡进行采样,因此仍然会生成大量的样本).我们也无法提取出visibility项,因为如果是glossy brdf,他是一个高频的,且Lighting项的积分域是整个半球,因此并不满足smooth或small support,因此无法提取出visibility项.

在工业界中,我们通常以环境光中最亮的那个作为主要光源,也就是太阳,只生成太阳为光源的shadow.

下面是几篇关于生成阴影的文章:


1.做的是全局光照部分产生的shadow.

2.解决的是离线渲染中的many lights的问题,核心思想是把反射物当成小光源,把所有的小光源做一下归类并近似出照射的结果.

3.Real Time Ray Tracing,可能是最终解决方案.

4.PRT可以十分准确的得到来自环境光中的阴影.

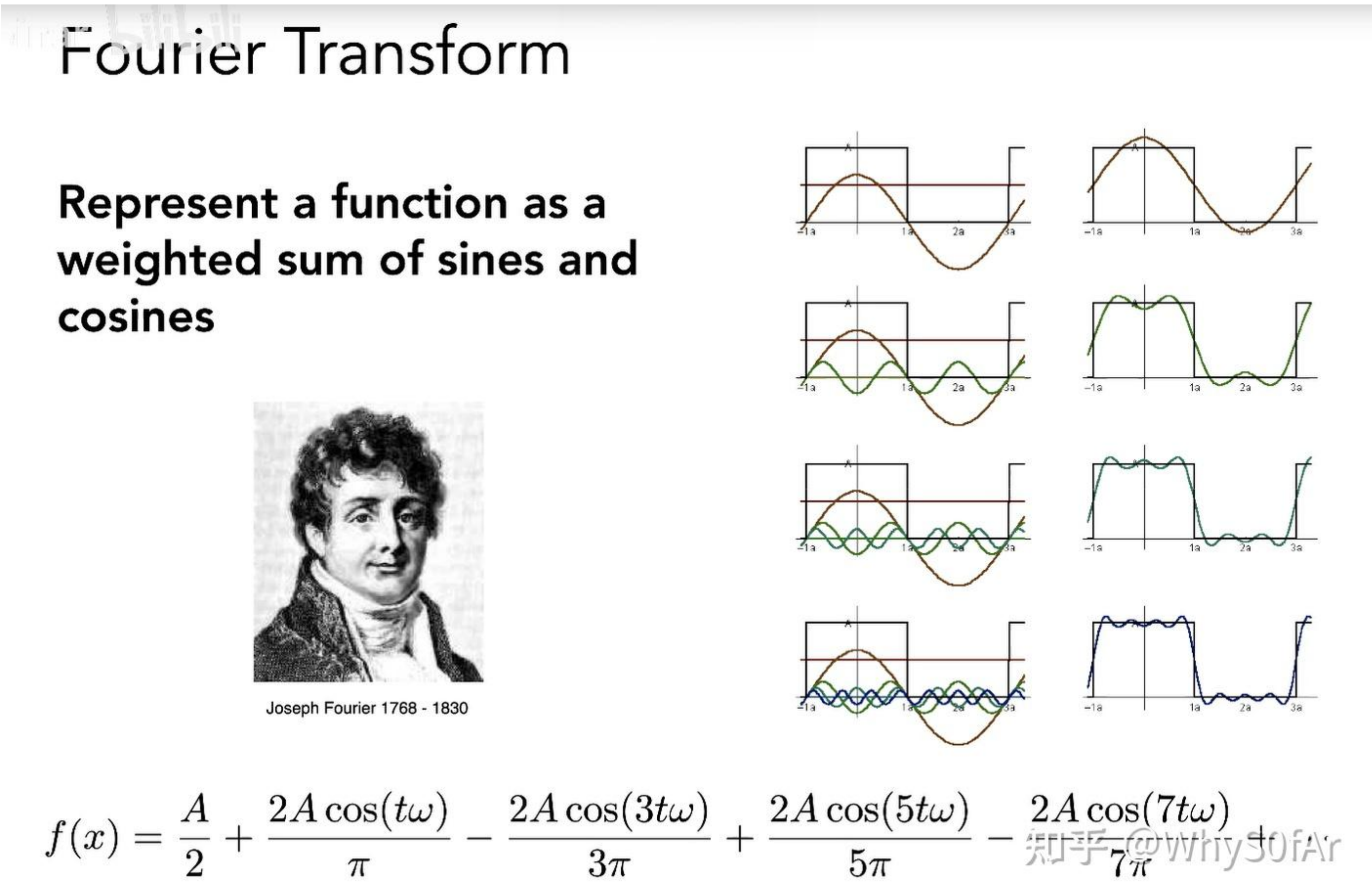
但是我们知道世上没有十全十美的事情,

**那么....古尔丹,代价是什么呢?**

在讲主要内容之前让我们回顾一下GAMES101中讲过的一些数学知识.

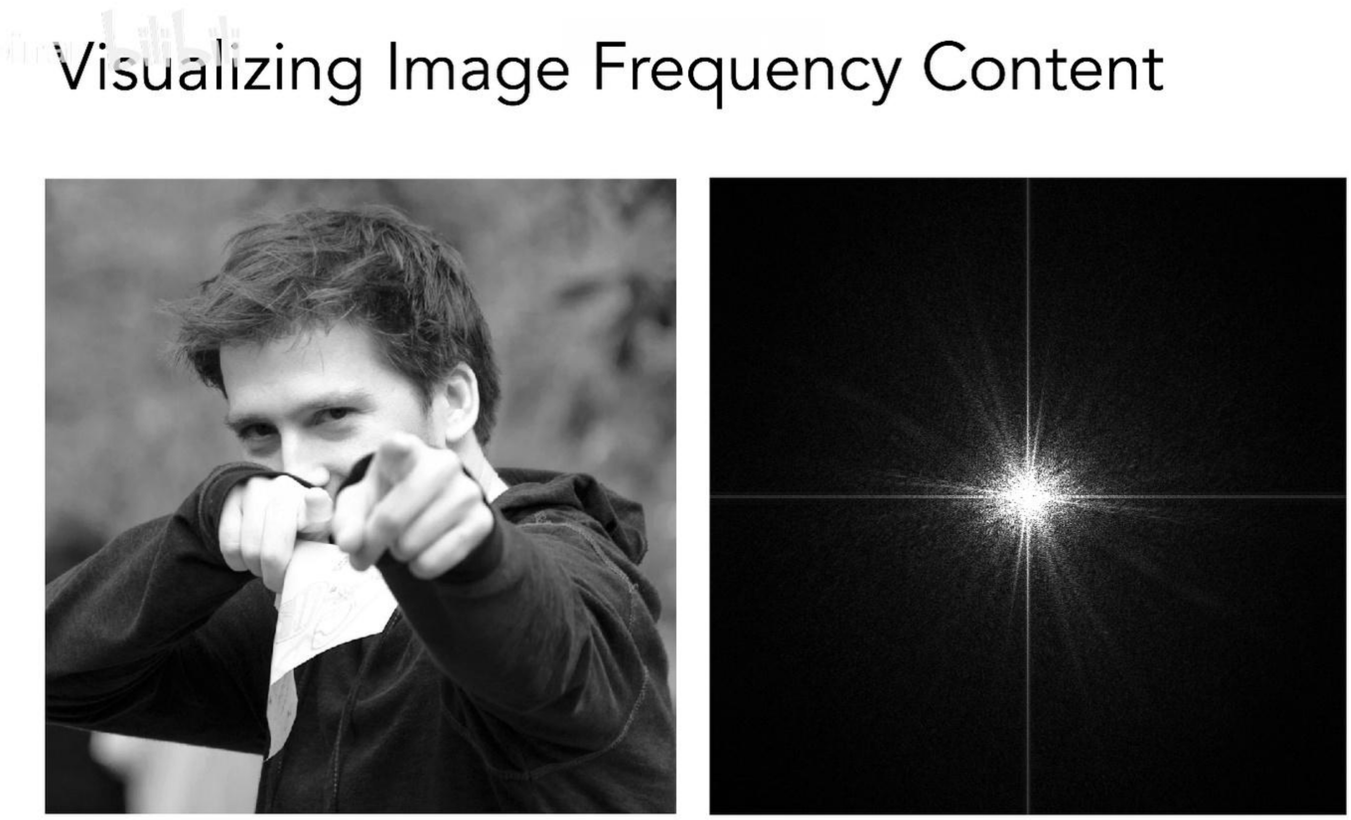
# 知识储备:

1.傅里叶级数展开：任何一个函数可以写成常数和一系列基函数(不同频率sin和cos项)的线性组合,基函数数量越多越接近于原函数的形状：



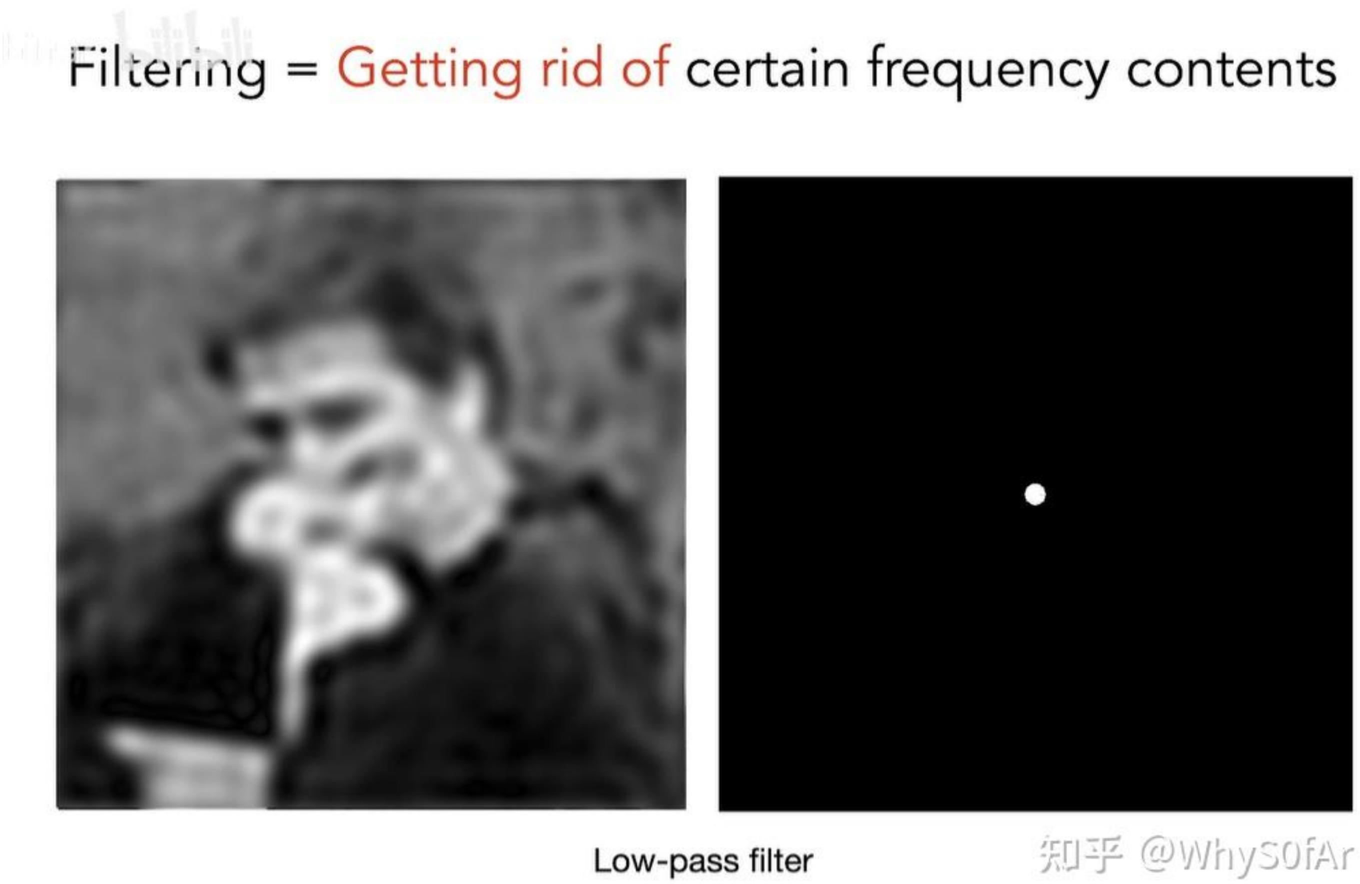

2.频率：在空间上图像信号数值的变化是否剧烈,如头发区域属于高频,因为是一根一根的，衣服,背景等变化不剧烈的属于低频.

任何一张图(也就是二维函数)的频率,也就是频域上对应的内容可以用一张频谱表示出来。

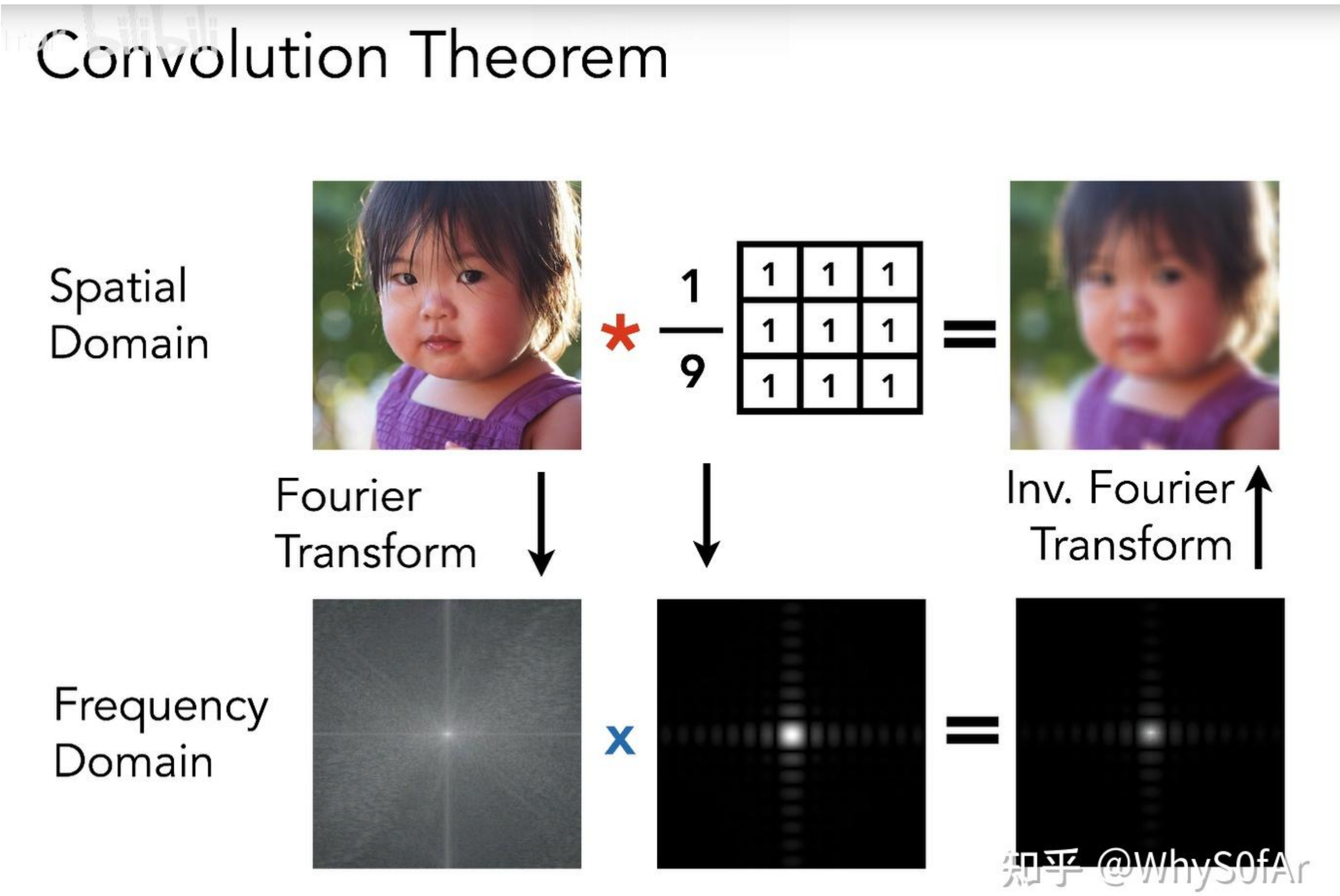





频谱最中心处是低频内容,我们可以做一个filtering(滤波),从而去除一系列频率上的内容,我们对这张图用一个低通滤波器,从而把高频的内容去除掉.

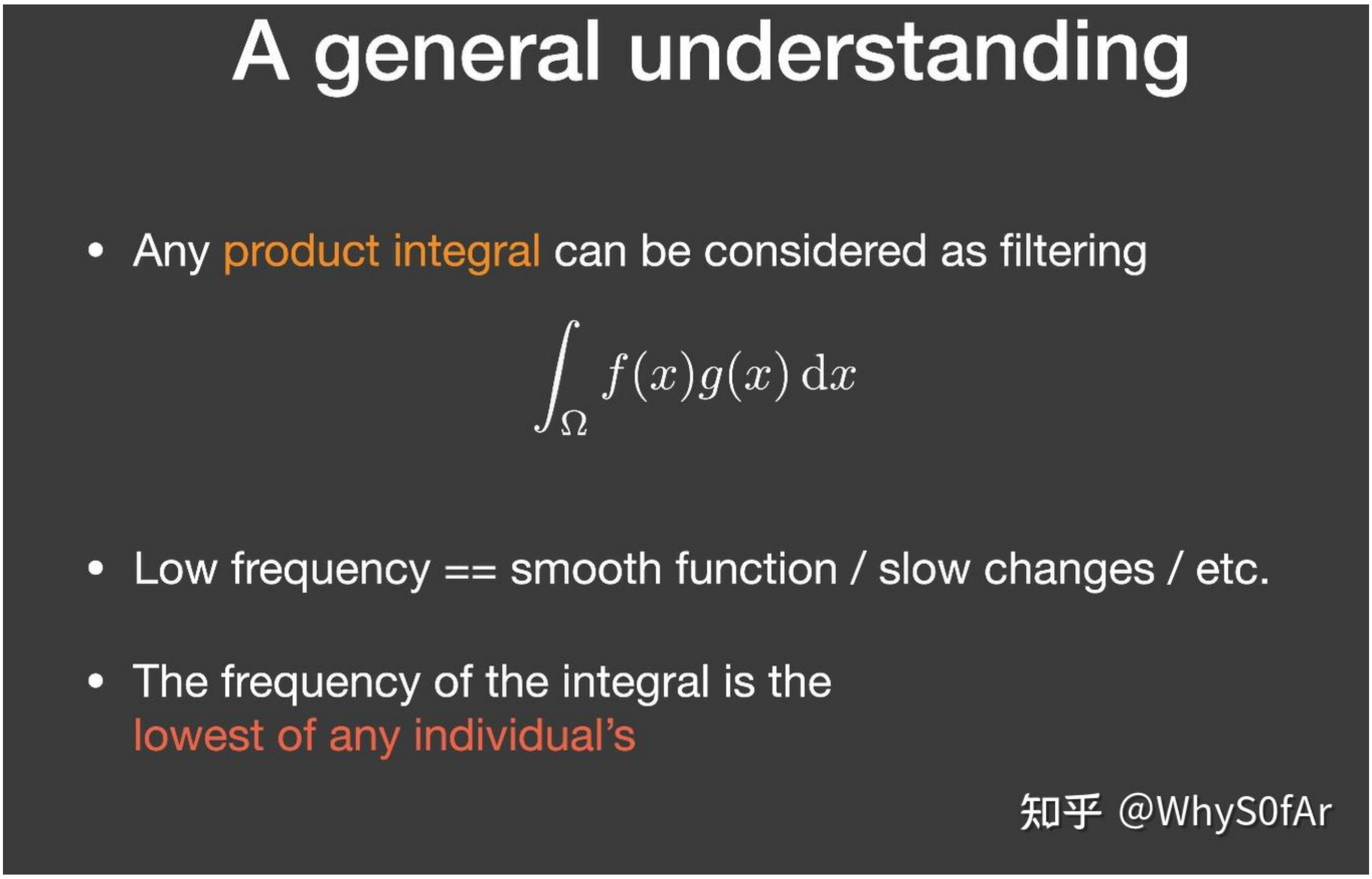



首先我们来说一下卷积,卷积其实就是一个模糊操作,在图上取任意一点,取点的周围一定区域内的像素值进行加权平均并将结果写回这个点,这就是卷积.




在spitial域上做卷积也就等于在函数上做一个卷积,就等于在频域上做一个 原图频谱 和 卷积核频谱 的乘积操作,我们可以看到卷积核的高频部分几乎是黑的,也就是0,做了成绩操作后原本的高频部分就消失了,逐点相乘后得到的结果在经过逆傅里叶变化得到模糊后的图.

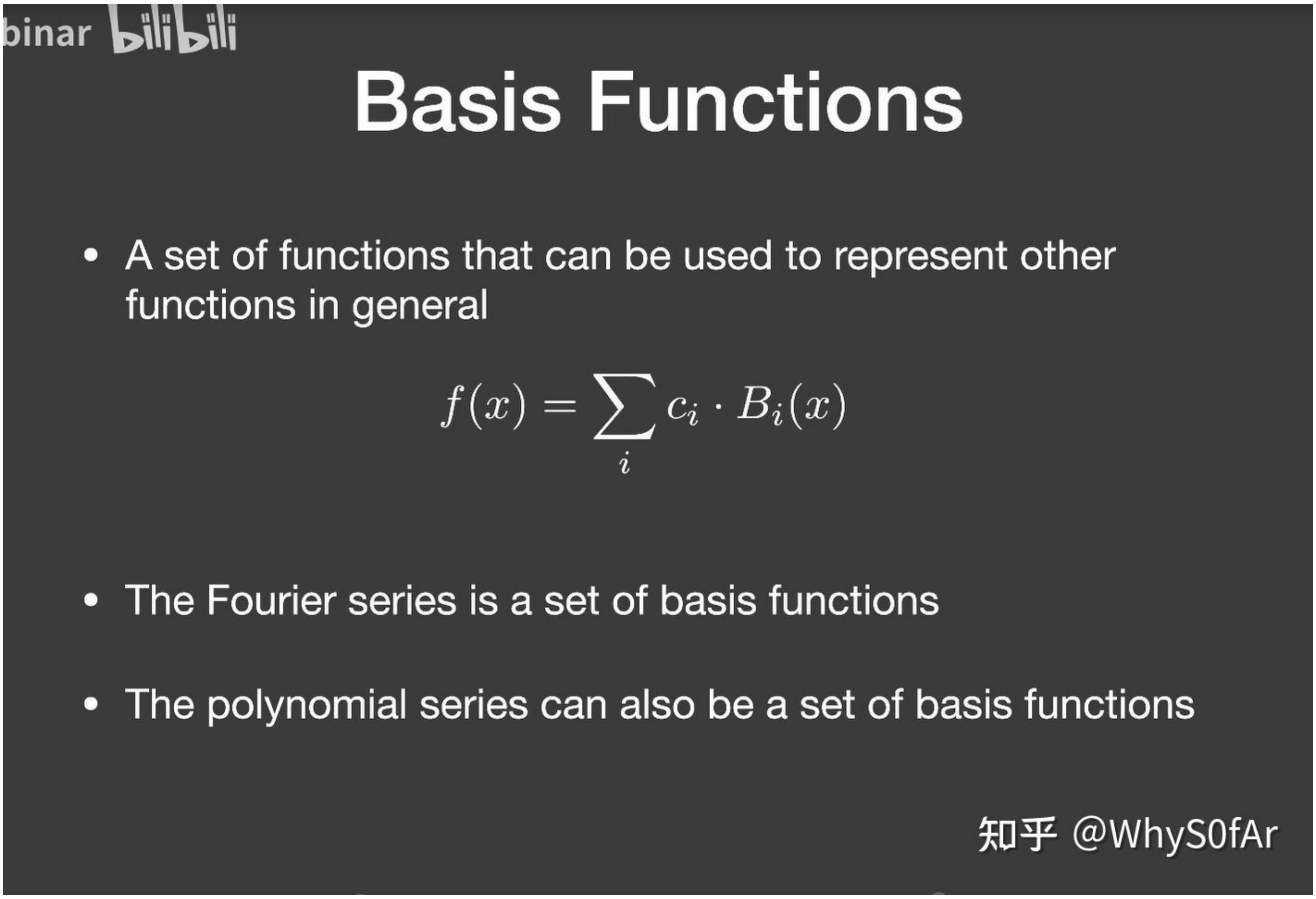
在本节课其实并不用这么复杂,本节课我们要记住的是:




对于任意的product integral(两个函数先乘积在积分),我们将其认为是做了一个卷积操作,理解为spatial域上的两个信号f(x)和g(x)进行一个卷积,等于在频域上让两个信号相乘,如果两个信号有一个信号是低频的,那么频域上相乘后得到的结果也是低频的,最终相乘在积分的结果也是低频的,可以总结为：**积分之后的频率取决于积分前最低的频率，即the frequency of the integral is the lowest of any individual's** 。

低频意味着变换更加地smooth或者有着slow的变化。

3.Basis Functions:




把一个函数可以描绘成其他函数的线性组合,如f(x)可以描绘成一系列的Bi函数乘以各自对应的系数最终再相加在一起,这一系列的函数Bi就是基函数.

回归正题,我们要讨论的是如何在环境光照下生成阴影,先从最简单的开始,如果给了你环境光和一个diffuse的物体,在不考虑Shadow的情况下如何去计算shading值?

为了计算shading值,我们引入数学工具----->Spherical Harmonics(球谐函数)

在游戏渲染中,SH有很多应用.比如SH可以用来表示低频部分的环境光照,也可以用来提供light probe的烘培光照等等..

# Spherical Harmonics(球谐函数)

SH是一系列基函数,系列中的每个函数都是2维函数,并且每个二维函数都是定义在球面上的。

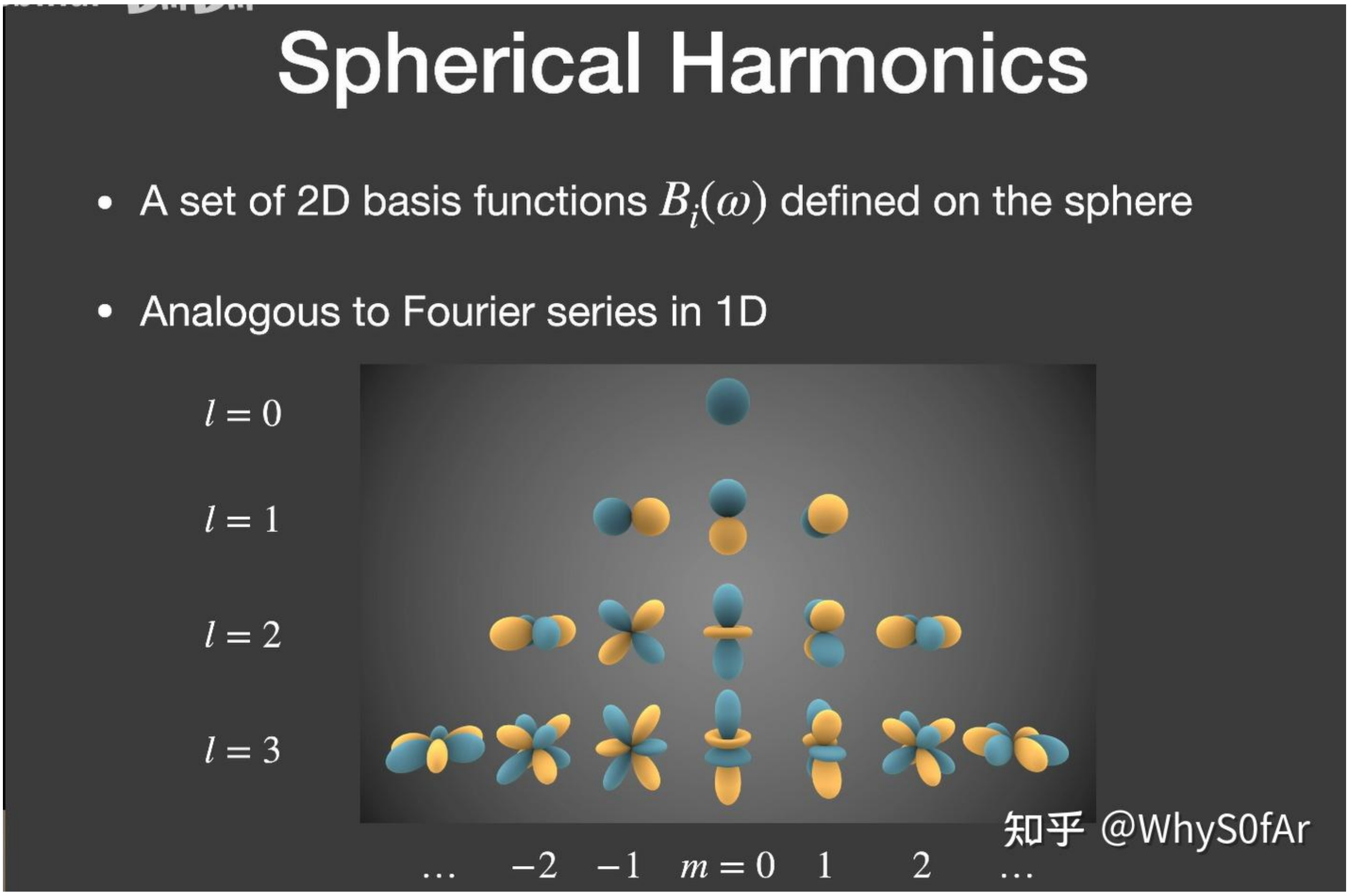
1.它是一系列的基函数，可以以傅立叶变换为参考,与里面不同频率的cos和sin函数类似,只是全都是二维函数

2.因为它是定义在球面上的，球面上会有不同的值,由于在球面上两个角度$\theta$和 $\varphi$就可以确定一个方向了,因此可以理解为是对方向的函数,通过两个角度变量从而知道这一方向对应在球面上的值.

下图是对SH的可视化，与一维的傅里叶一样,SH也存在不同频率的函数，但不同频率的函数个数也不同,频率越高所含有的基函数越多。

图中的颜色表示的是值的大小,l=0中,越偏白的蓝色地方值越大,越黑的地方值越小.而黄色中则表示偏白的地方表示其绝对值大,偏黑的地方表示绝对值小.也就是蓝色表示正,黄色表示负.

频率表示的就是值的变化,因此可以很清晰的从形状看出.




其中，l表示的是阶数，通常第l阶有$2l+1$个基函数，前n阶有$n^2$个基函数，m表示的是在某一个频率下基函数的序号，分别从从$-l$一直到$l$。每个基函数都有一个比较复杂的数学表示，对应一个legendre多项式，我们不用去了解legendre多项式，我们只需要知道基函数长这样,可以被某些数学公式来定义不同方向的值是多少就可以了.

下面定义一些操作：

**投影**： 由于一个函数 $f(w)$可以由一系列基函数和系数的线性组合表示，那么怎么确定基函数前面的系数，这就需要通过投影操作：

$$c_i = \int_{\Omega} f(\omega) B_i(\omega)\, \mathrm{d}\omega$$

我们知道函数F(X),通过对应的基函数B(i)进行投影操作,从而求出各基函数对应的系数Ci,与以下操作是同一个道理,在空间中想描述一个向量，可以xyz三个坐标来表达，把xyz轴当做三个基函数，把向量投影到xyz轴上，得到三个系数就是三个坐标。

**重建**： 知道基函数对应的系数，就能用系数和基函数恢复原来的函数。

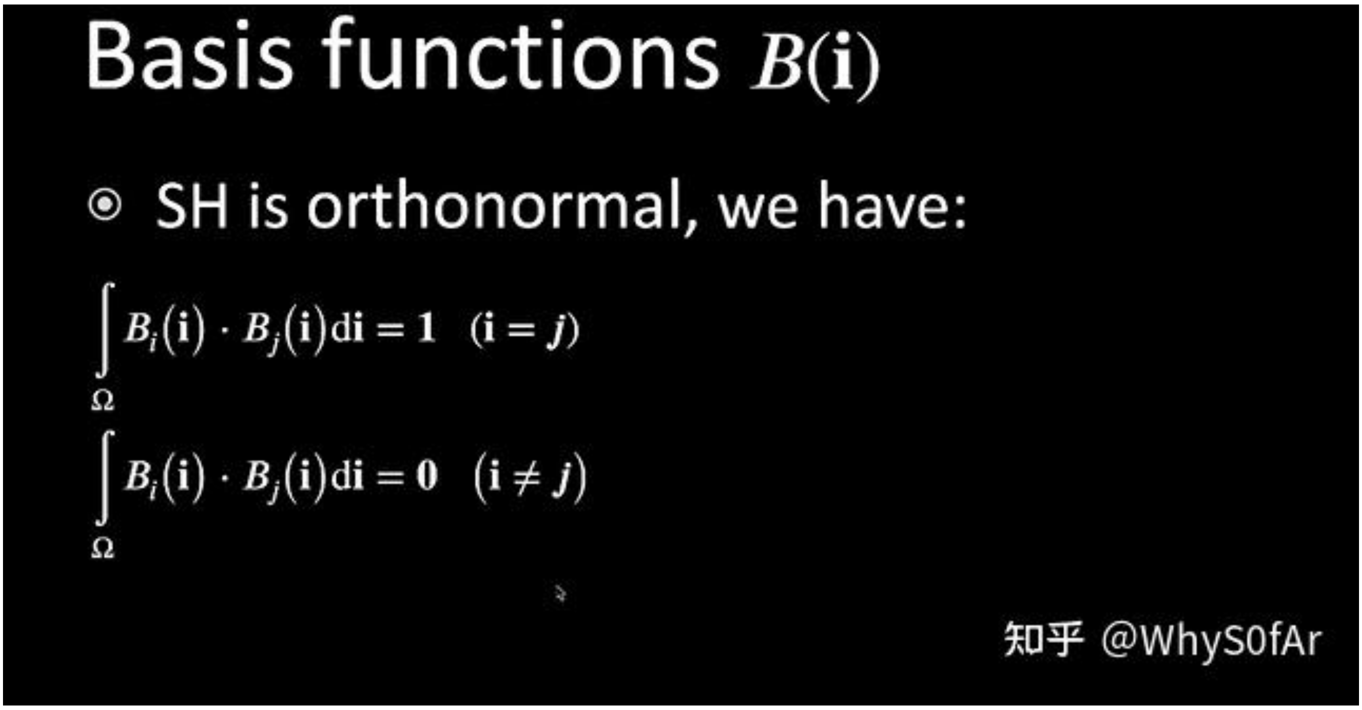
由于基函数的阶可以是无限个的，越高的阶可恢复的细节就越好,但一方面是因为更多的系数会带来更大的存储压力、计算压力，而一般描述变化比较平滑的环境漫反射部分，用3阶SH就足够了；另一方面则是因为SH的物理含义不是特别好理解，高阶SH容易出现各种花式Artifact，美术同学一般都会认为这种表现属于bug。

$$c_i = \int_{\Omega} f(\omega) B_i(\omega)\, \mathrm{d}\omega$$

f(w)可以是任何一个函数,我们说过基函数可以重建任何一个球面函数,那么我们这里的f(w)就是环境光照,由于环境光是来自于四面八方且都有值,所以环境光照就是一个球面函数,,我们可以把它投影到任何一个SH basis上,可以投影很多阶,但是只需要取前三阶的SH去恢复环境光就可以恢复出最低频的细节了,这个在下文RAVI教授的结论有提到.

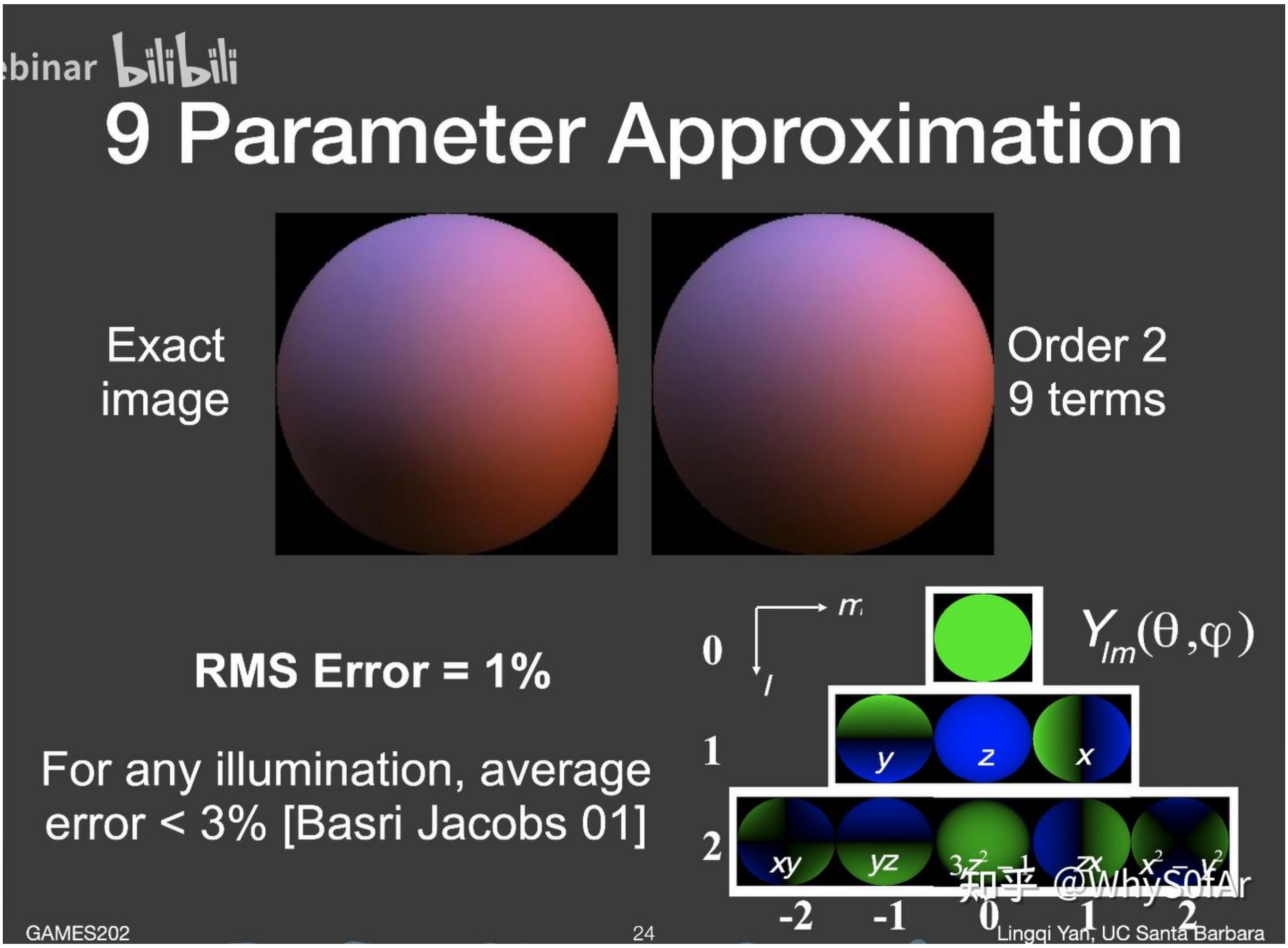
这里补充一些球谐函数的性质：

**正交性：能够较简单地投影/重建、simple rotation。**




旋转是一个很重要的性质：**旋转一个基函数之后，得到的函数就不再是一个基函数(因为基函数有严格的朝向等限制)，但是旋转球谐函数等价于同阶基函数的线性组合。**

Ravi教授等人在01年左右做过一些实验发现，diffuse BRDF类似于一个低通滤波器，使用一些低频信息就可以恢复出原始内容。回忆一下，在本文之前的内容中曾说过：“**积分之后的频率取决于积分前最低的频率** ”，当diffuse BRDF使用低频信息即可恢复内容时，也就意味着无论光照项是多么复杂，其本应该用多高频的基函数去表示，但我们希望得到的是其与BRDF之积的积分，所以可以使用比较低频的基函数去描述灯光。下面的实验结果意味着，遇到diffuse的物体时使用前3阶的球谐基函数就可以基本重建出正确率99%的结果，




# 到目前为止，这里只解决了如何根据环境光照进行shading，主要说明球谐函数的作用，仍没考虑shadow问题，接下来我们要解决两个问题:

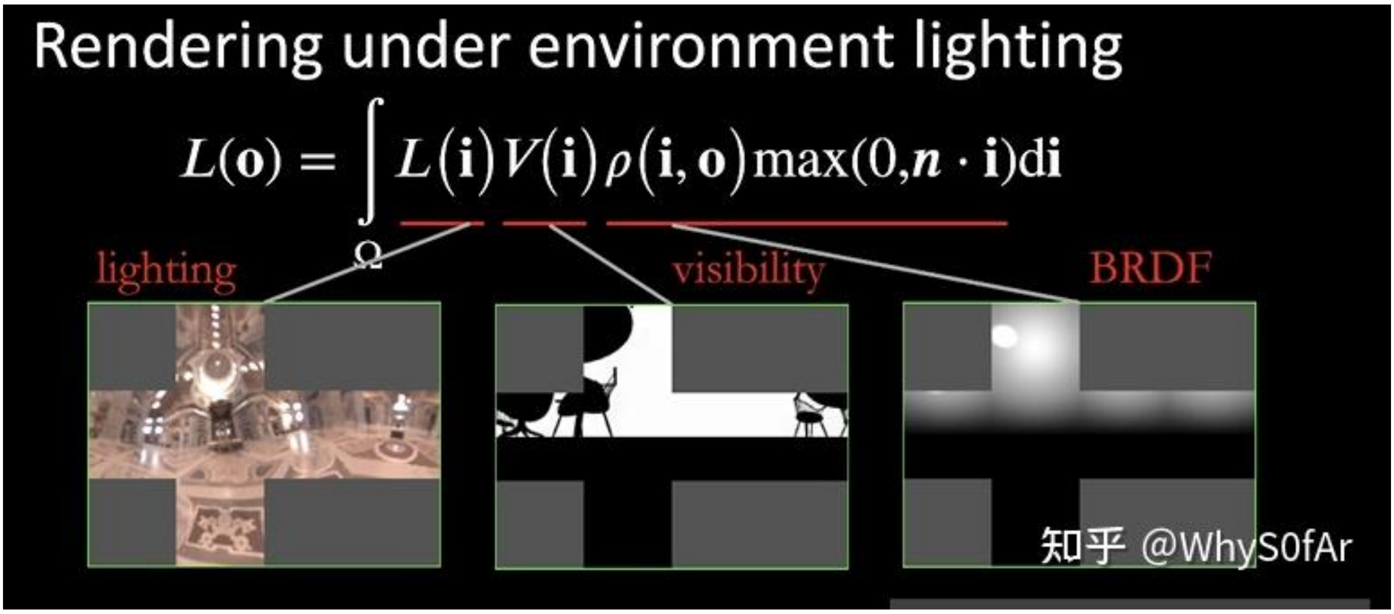
1.将shadow考虑进去

2.我们仍然用基函数思路去考虑任何的brdf.

解决的思路就是prt,接下来我们进入prt学习:

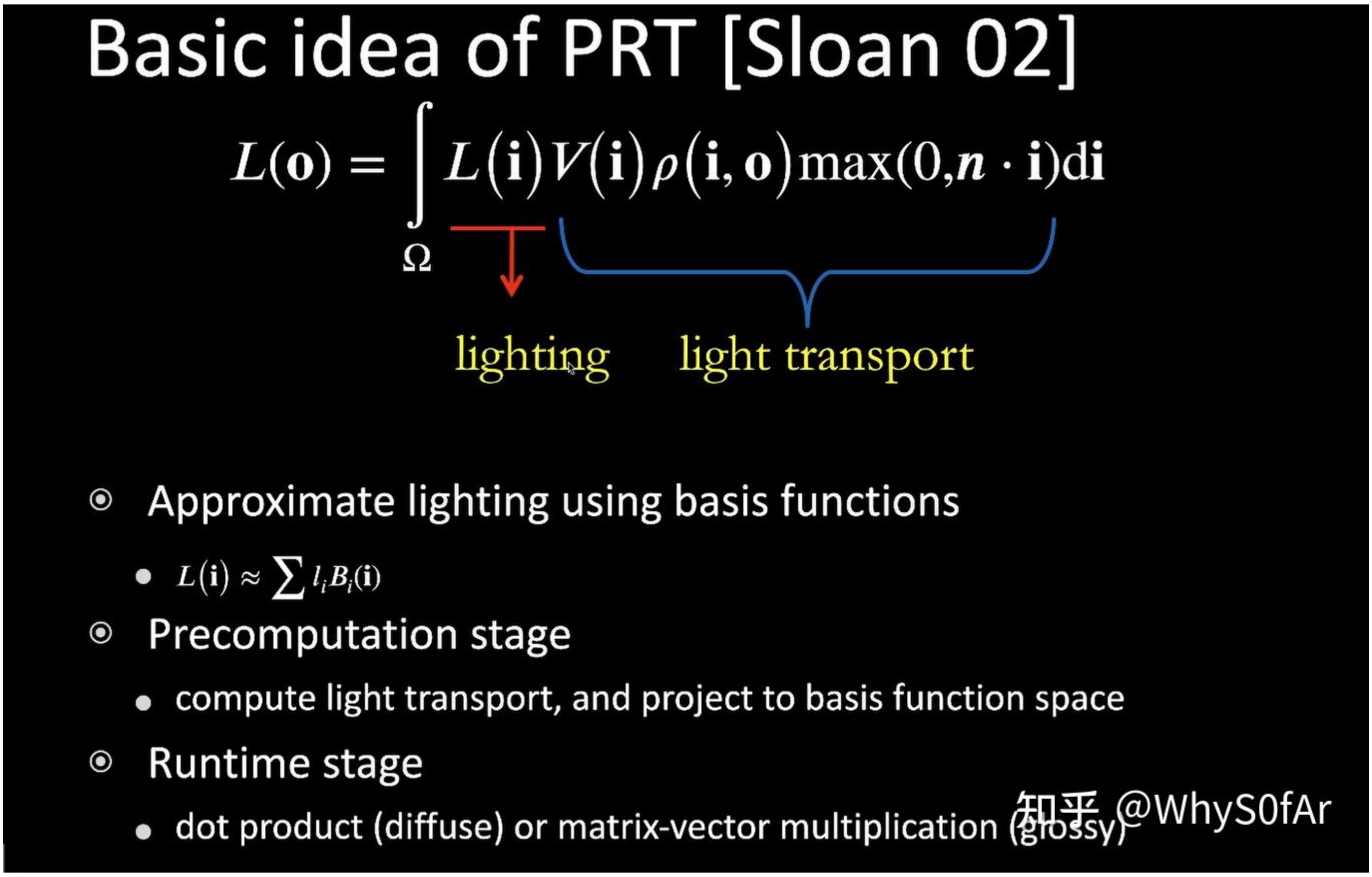
# PRT

在实施渲染中,我们把rendering equation写成由三部分组成的积分:




光照项,visibility项和brdf项,这三项都可以描绘成球面函数,这里用的是cube map描述法,那么最简单的解这个方程的方法就是每个像素挨个去乘,假设环境光是$6*64*64$的map，对于每个shading point来说，计算shading需要计算$6*64*64$次。这个开销是十分大的.

因此我们利用基函数的基本原理把一些东西先预计算出来,从而节省开销.




PRT的基本思想:

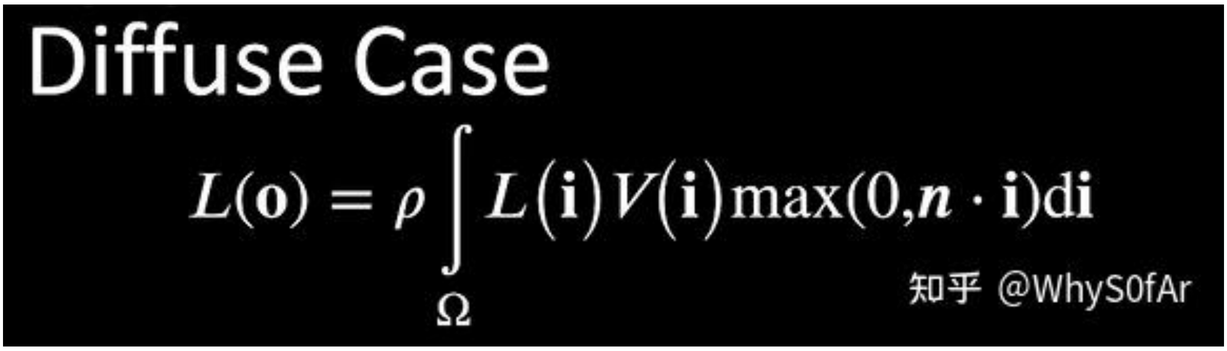
我们把rendering equation分为两部分,lighting 和 light transport.

假设在渲染时场景中只有lighting项会发生变化(旋转,更换光照等),由于lighting是一个球面函数,因此可以用基函数来表示,在预计算阶段计算出lighting.

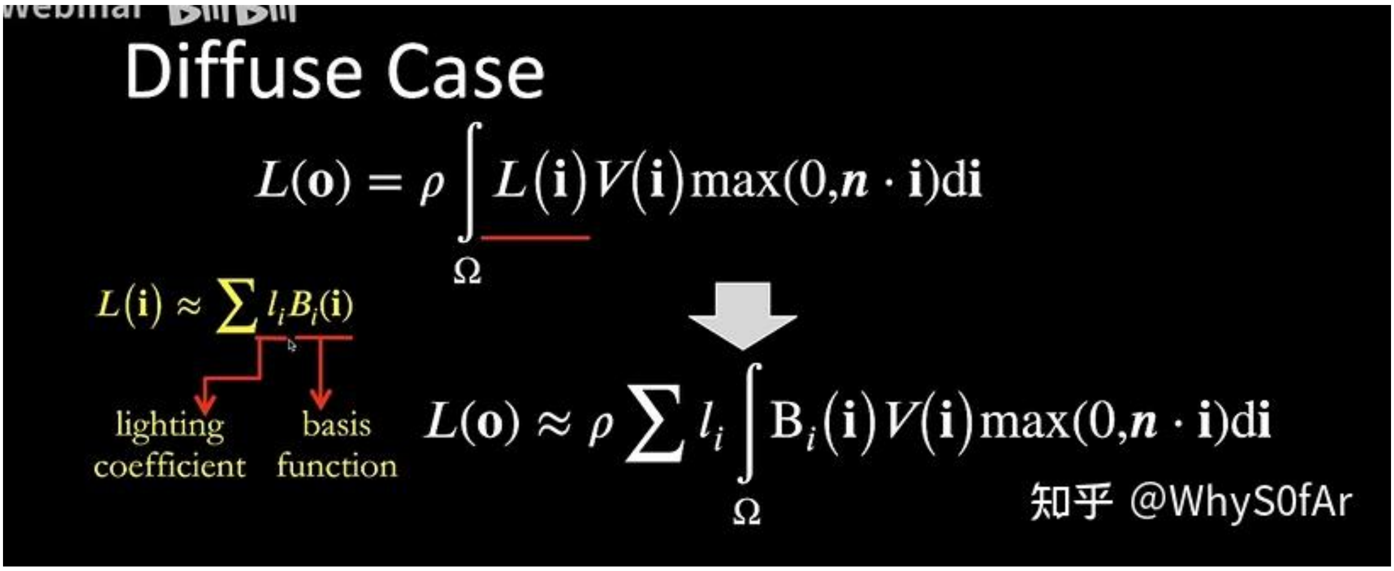
而light transport(visibility和brdf)是不变的,因此相当于对任一shading point来说,light transport项固定的,可以认为是shading point自己的性质,light transport总体来说还是一个球面函数,因此也可以写成基函数形式,是可以预计算出的.

我们分为两种情况,diffuse和glossy:

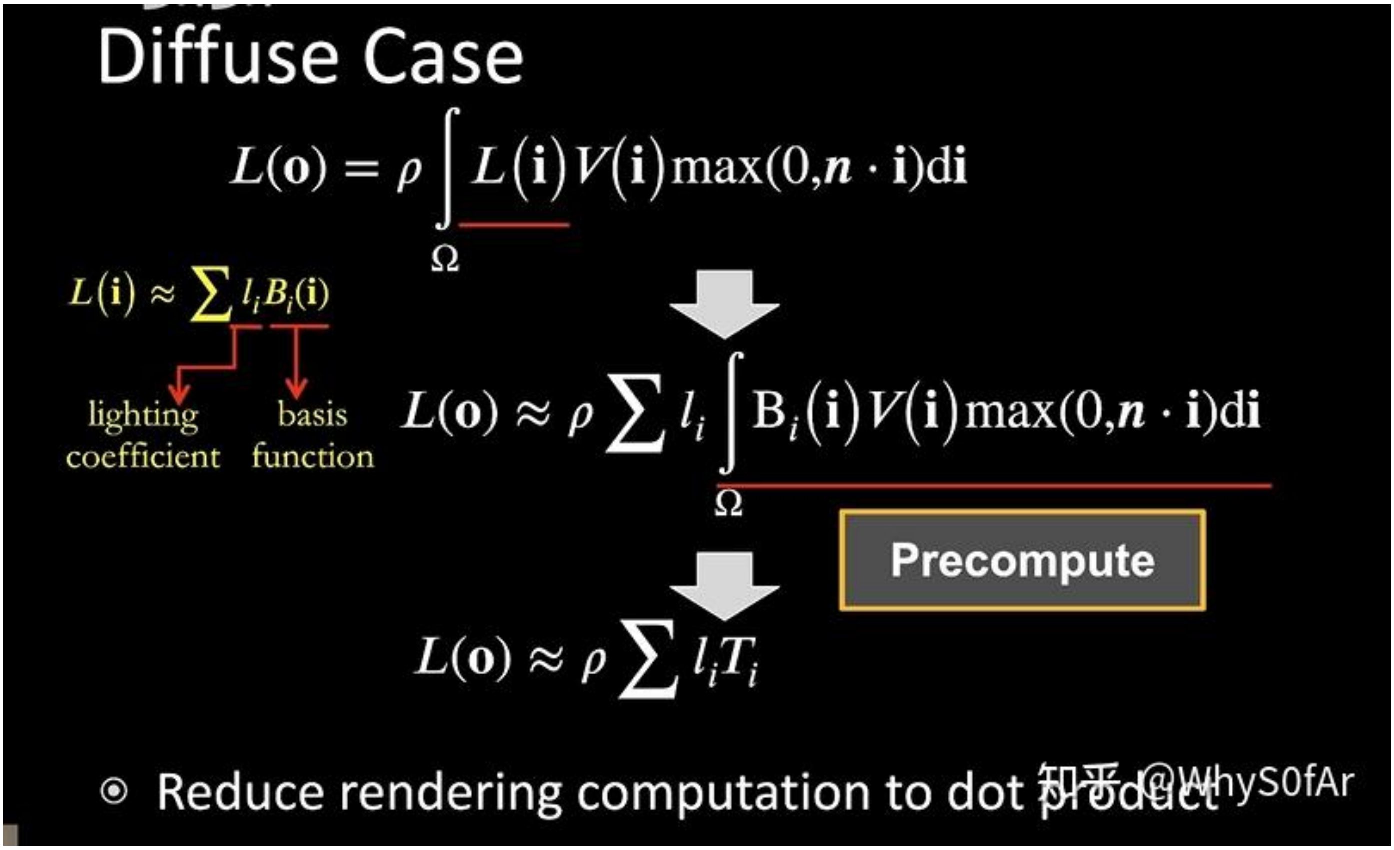
Diffuse:




由于在diffuse情况下,brdf几乎是一个常数,因此我们把brdf提到外面.




由于lighting项可以写成基函数的形式,因此我们求和式把其代入积分中,对于任何一个积分来说,在Bi的限制下,li此对积分来说是常数,可以提出来.




对于积分中的部分来说,Bi是基函数,v和cos项在一起不就是light transport吗,那不就是light transport乘与一个基函数,这就成了lighting transport投影到一个基函数的系数，接下来代入不就能进行预计算了吗，这样就只要算一个点乘就好了。

之所以说是点乘,结果是个求和,我们要计算 $l_1T_1$+ $l_2T_2$+......,不正好相当于两个向量点乘吗.

所以对于任何一个shading point我们去算他的shading 和shadow,只需要计算一个点乘就可以了,十分方便,

## 但是,没有东西是十全十美的,那么,古尔丹,这次的代价又是什么呢?

1.light transport做了预计算,因此visibility当了常量,因此场景不能动,因此只能对静止物体进行计算.

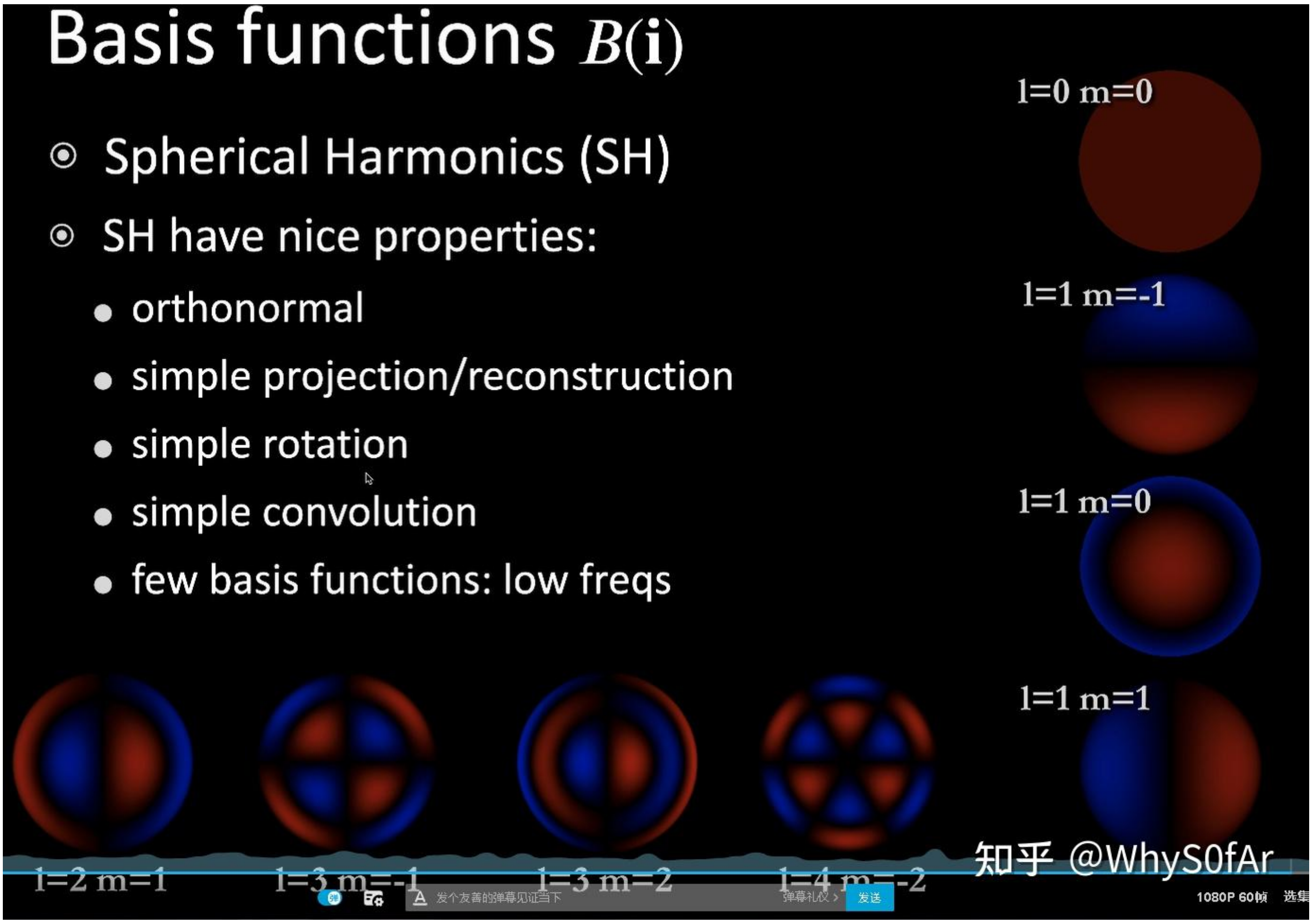
2.对于预计算的光源我们把它投影到sh上,如果光源发生了旋转,那不就相当于换了个光源吗?

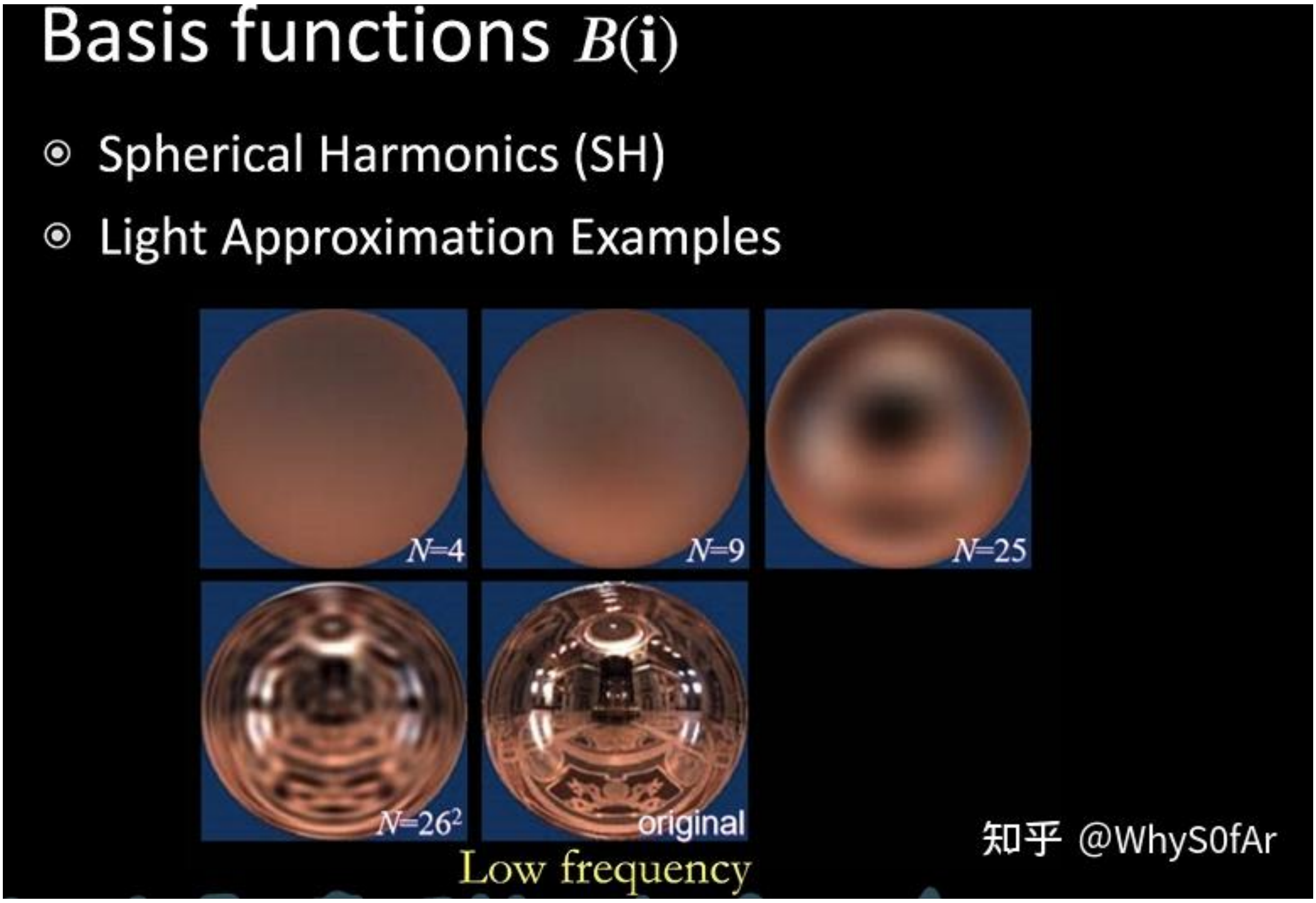
但是第二个问题由于sh函数的旋转不变性可以完美的解决.

旋转光照 = 旋转SH的基函数

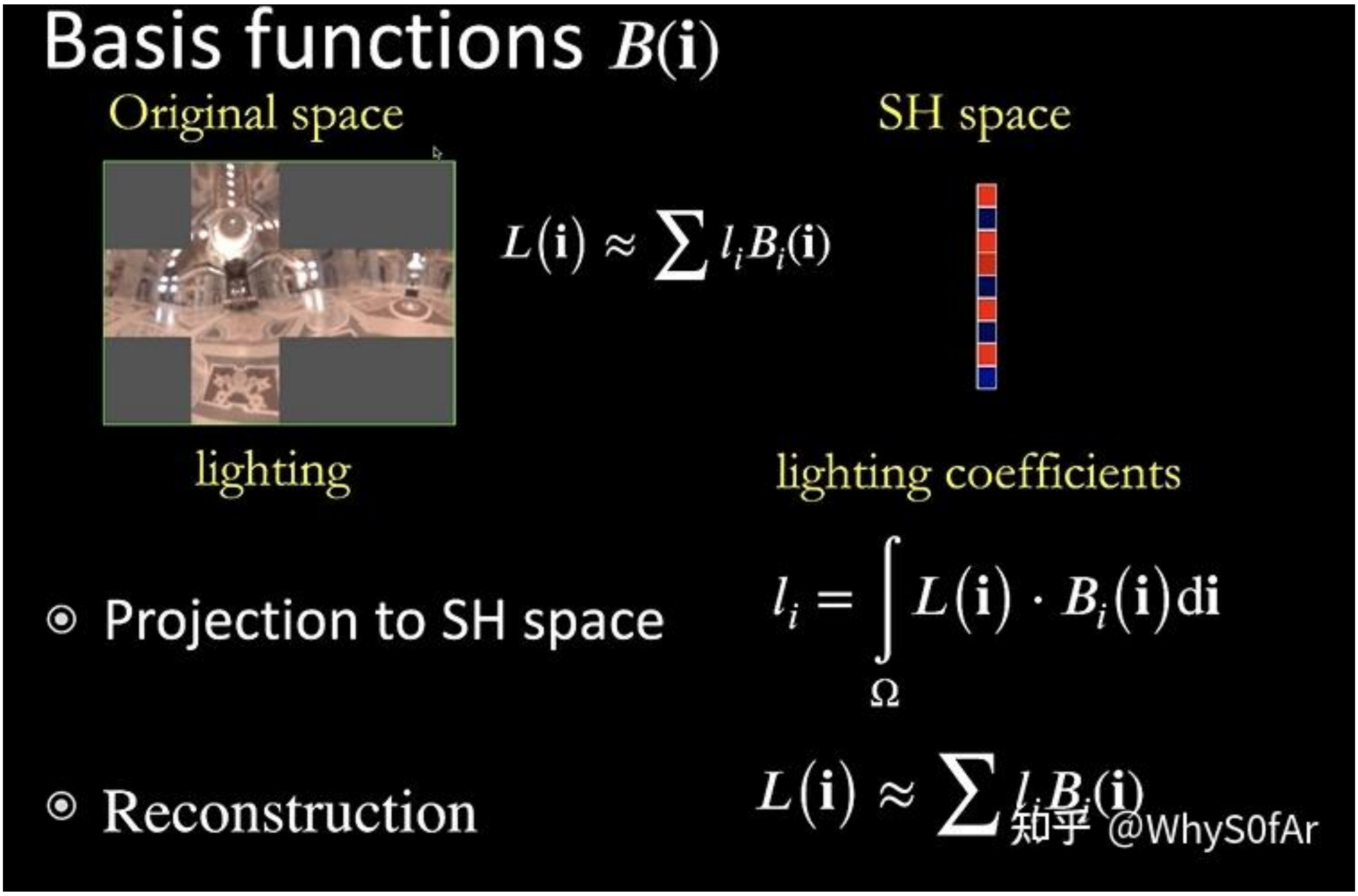
但 任何一个SH基函数旋转后都可以被同阶的SH基函数线性组合表示出来

因此,我们根据这个性质,还是可以立刻得出旋转后的sh基函数新的线性组合.


Basis functions $B(\mathbf{i})$
Spherical Harmonics (SH)
SH have nice properties:
orthonormal
simple projection/reconstruction
simple rotation
simple convolution
few basis functions: low freqs
l=0 m=0
l=1 m=-1
l=1 m=0
l=1 m=1
l=2 m=1
l=3 m=-1
l=3 m=2
l=4 m=-2
知乎 @WhyS0fAr

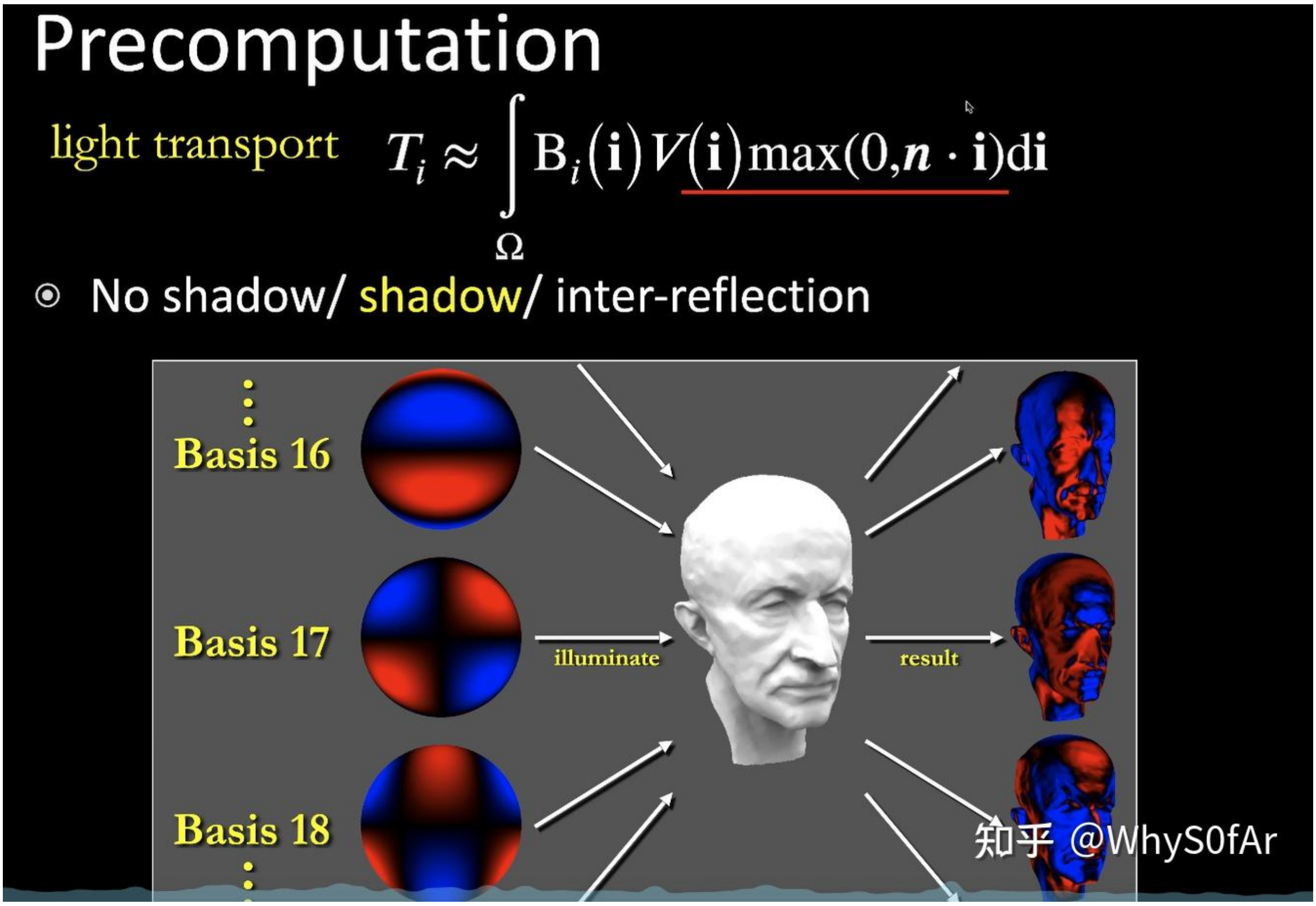

Basis functions $B(\mathbf{i})$
Spherical Harmonics (SH)
Light Approximation Examples
N=4
N=9
N=25
N=26²
original
Low frequency
知乎 @WhyS0fAr

用的阶数越多越接近与原始函数,第四张图是前26阶函数去重建原始函数,可以看到效果还不错.但我们在使用时用不到那么多阶.

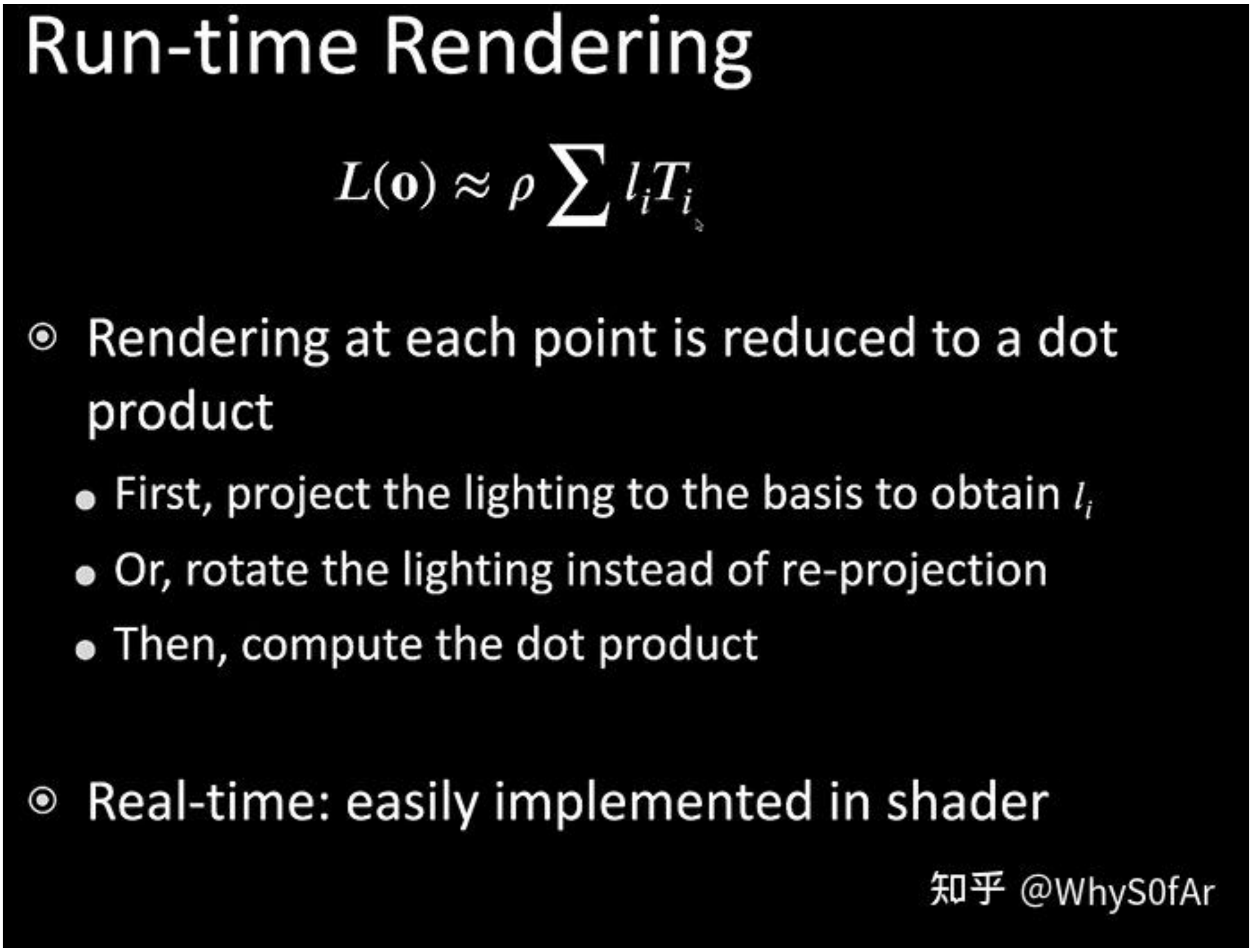



我们将lighting这个球面函数,通过SH的基函数用一堆系数来表示,这些系数排成一行也就是组成了向量,因此光照变成了一个向量.

如果要重建原函数则只需要把这些系数乘以对应的基函数再加在一起即可.




我们可以把Bi理解为lighting,也就是说每个basis所描述的环境光去照亮这个物体从而得到照亮之后的结果,我个人理解预计算就是把每个basis照亮得到的结果生成.




最后我们在计算shading 和 shadow时只需要进行向量li和ti的点乘即可得到结果.

到此我们知道了如何再已知环境光的情况下,通过使用PRT来计算出diffuse物体的shading 和 Shadow了.

下节课我们讲关于glossy和全局光照.