# 纹理生成与压缩

主讲：胡义伟 Adobe Research

# 目录

CONTENTS

01 纹理简介

02 纹理生成

03 纹理使用与压缩

# 纹理简介

# 纹理（Textures）

如果没有纹理只有几何，那么一个渲染的场景就没有颜色

# 纹理（Textures）

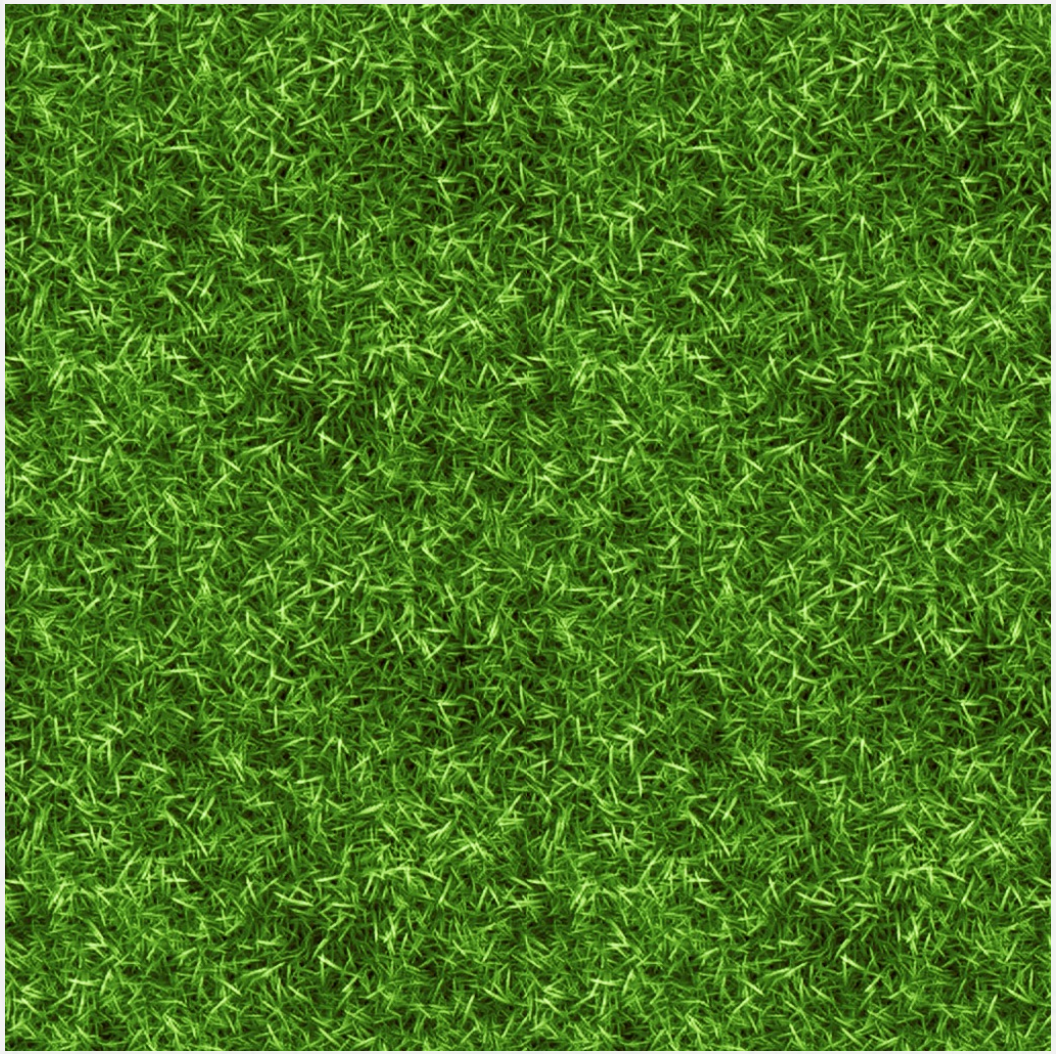
纹理或者说材质（Material），定义了几何物体如何与光进行交互。

# 纹理的定义

纹理：多维的固定数组（1D, 2D, 3D），数组里面可以存储不同类型的元素，比如浮点，整形等等。

纹理通常使用的场景：

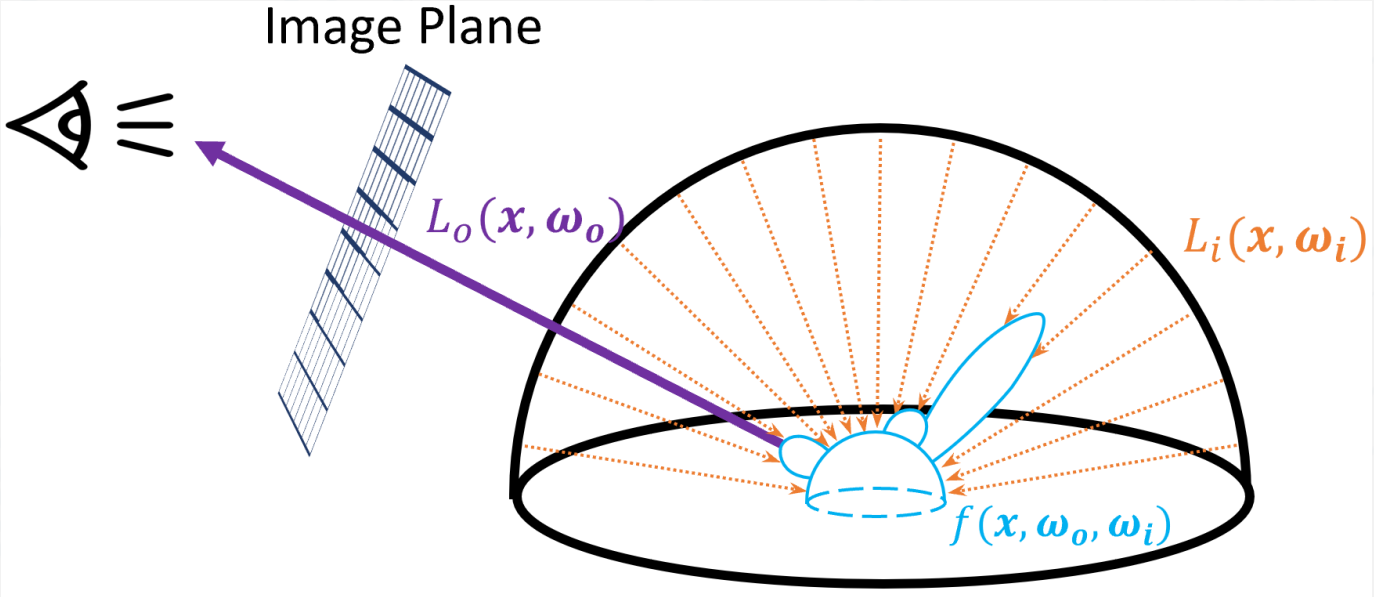
- 材质（Material）表达
- 深度表达，如shadow mapping
- 帧缓存 （Framebuffer）



一张1K的二维RGB纹理

# 纹理与材质

在现代图形学的渲染中, 材质是非常重要的组成部分，它数学上被定义为双向反射分布函数（Bidirectional Reflectance Distribution Function），它是一个4D函数。




$$L_o(\mathbf{x}, \omega_\mathbf{o}) = L_e(\mathbf{x}, \omega_\mathbf{o}) + \int_\Omega L_i(\mathbf{x}, \omega_\mathbf{i}) \boxed{f_r(\mathbf{x}, \omega_\mathbf{o}, \omega_\mathbf{i})} (\omega_\mathbf{i} \cdot \mathbf{n}) d\omega_\mathbf{i}$$

材质定义了渲染方程里的核心部分

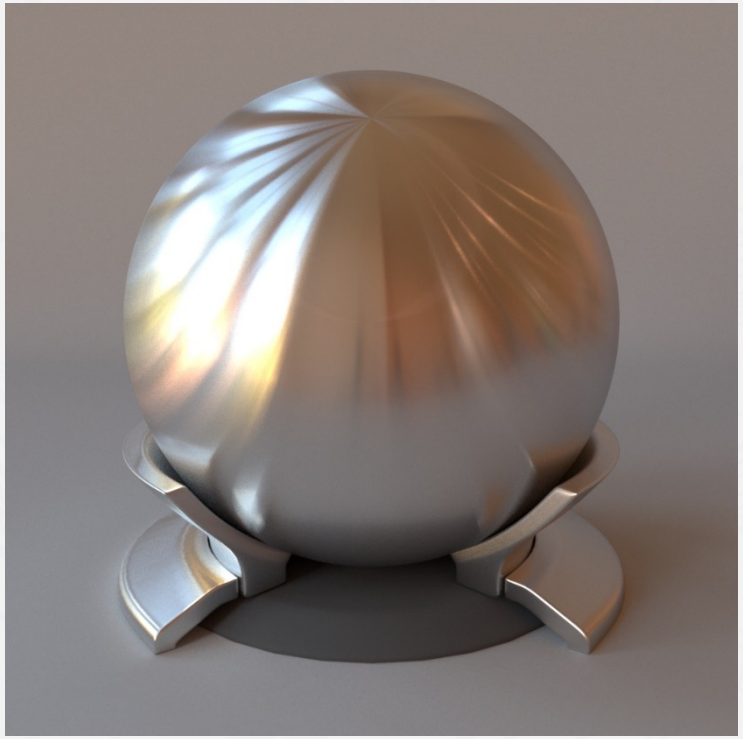
# 纹理与材质

但通常来说BRDF描述的仅仅是单一的材质。但是真实世界的物体实际上有许多不同的材质构成。那么如何去表达呢？

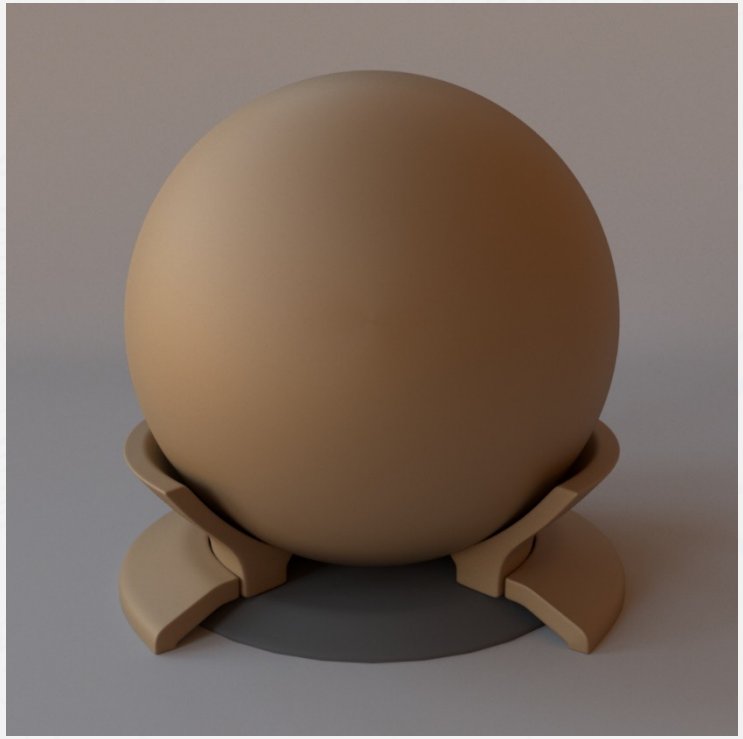
我们需要描述材质的空间变化，所以我们引入新的位置变量X。这样我们就可以定义Spatial-Varying (SV) BRDF，SVBRDF因此是一个6D函数。

$$f_r(\omega_i, \omega_r, \mathbf{x})$$

描述2D的空间位置



铝(Anisotropic brushed aluminum)



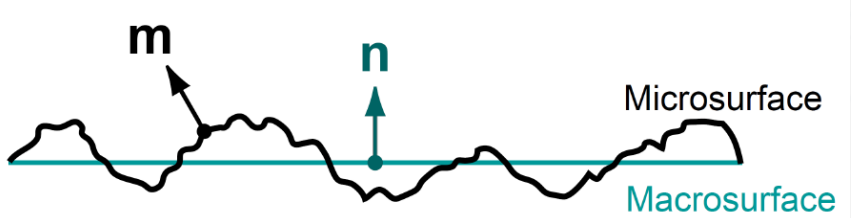
纸板(Cardboard)

# 纹理与材质

在实时渲染中，无论是4D的BRDF或者是6D的SVBRDF通常都不会直接被使用，因为所需的存储空间过大。因此通常会使用解析模型进行逼近。

BRDF是4D函数，而Analytical BRDF是一个0D函数，因为只需要存储几个有限的参数！

微表面模型(Microfacet Model)




$$f_r(\omega_i, \omega_r) = \frac{D(\omega_h)F(\omega_r, \omega_h)G(\omega_i, \omega_r, \omega_h)}{4(\mathbf{n}\cdot\omega_i)(\mathbf{n}\cdot\omega_r)}$$

Albedo/Metallic工作流：

$$f_r = \boldsymbol{a}\times(1-\boldsymbol{m})\times f_d + \left((\boldsymbol{a}-f_0)\times\boldsymbol{m}+f_0\right)\times f_s$$

$f_0$：法线方向反射度，通常为0.04，若假设折射率为1.5
$f_d$：漫（diffuse）反射分量，通常为Lambert模型。
$f_s$：镜面（Specular）反射分量，通常为微表面模型。

$\boldsymbol{a}$：albedo值
$\boldsymbol{m}$： metallic值
**n**： 法线方向
$\boldsymbol{\alpha}$： roughness值

$$D(\boldsymbol{\omega}_h) = \frac{\alpha^2}{\pi\left((\boldsymbol{\omega}_h\cdot\mathbf{n})^2(\alpha^2-1)+1\right)^2}$$

法线分布函数（Normal Distribution Function）

# 纹理与材质

Albedo/Metallic工作流：

BRDF:

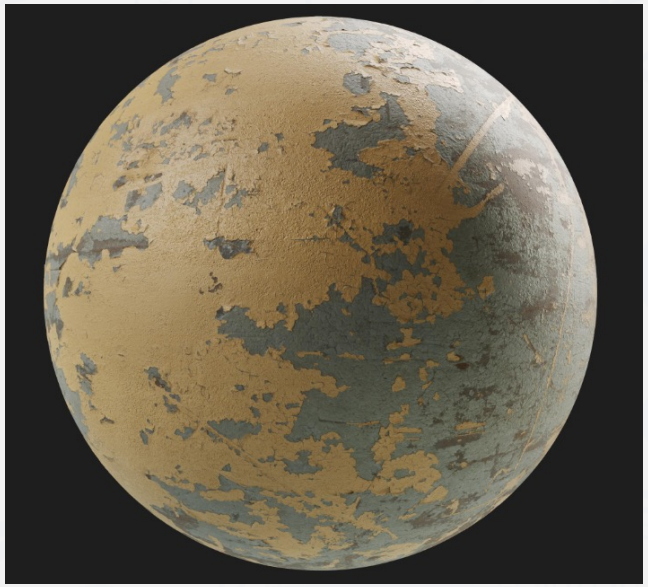
$$f_r = \boldsymbol{a}\times(1-\boldsymbol{m})\times f_d + \left((\boldsymbol{a}-f_0)\times\boldsymbol{m}+f_0\right)\times f_s$$

NDF:

$$D(\boldsymbol{\omega}_\mathbf{h}) = \frac{\alpha^2}{\pi\left((\boldsymbol{\omega}_\mathbf{h}\cdot\mathbf{n})^2(\alpha^2-1)+1\right)^2}$$

$\boldsymbol{a}$：albedo值
$\boldsymbol{m}$ ：metallic值
**n** ：法线方向
$\boldsymbol{\alpha}$ ：roughness值

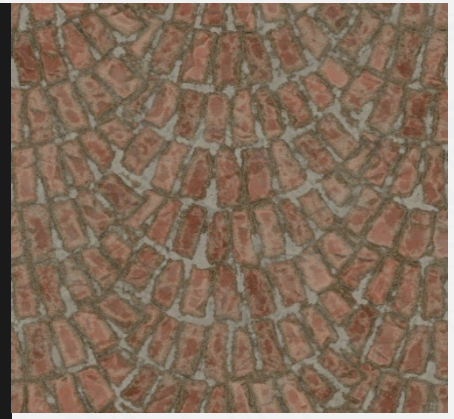
现在我们考虑去表达一个SVBRDF，我们因此可以将这些参数使用纹理进行表达，因此analytical SVBRDF可以表达为二维的纹理！



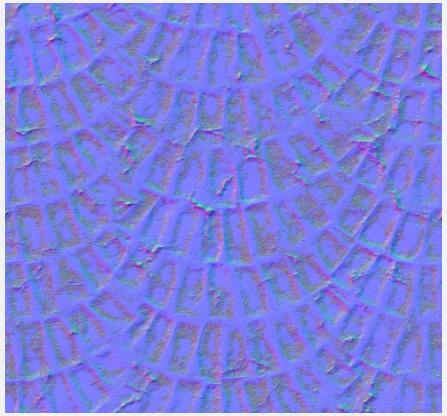
# 纹理与材质

一个利用纹理表达的PBR材质

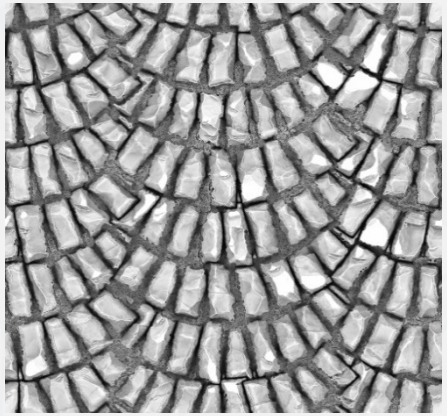
渲染结果



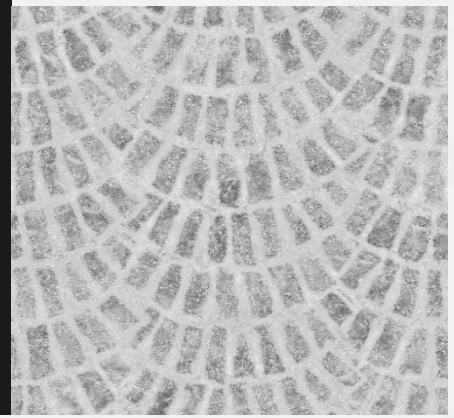
Albedo



Normal

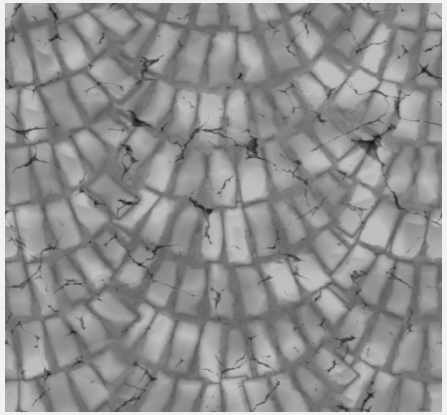


AO



Roughness

Metallic



Height

# 纹理生成

# 纹理的生成

材质纹理生成的方式多种多样。艺术家可以通过丰富的软件给几何模型生成纹理。

Md Sa Ds Pt Sg

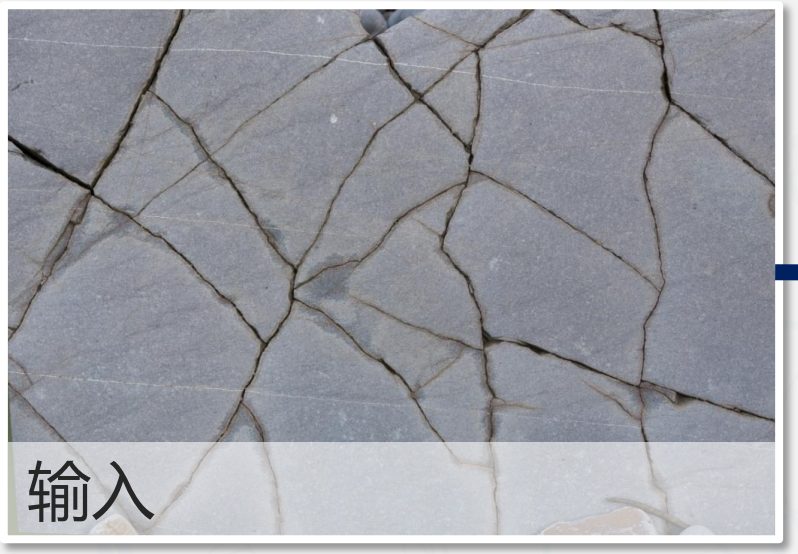
各种各样的算法也能够帮助艺术家生成材质纹理。

- 纹理合成
- 材质转换工具

@PabloMunozGomez by Adobe Substance Painter

# 纹理的合成

纹理合成是最为经典的算法之一。如果已经有了一张小纹理，如何生成更大的纹理图片？

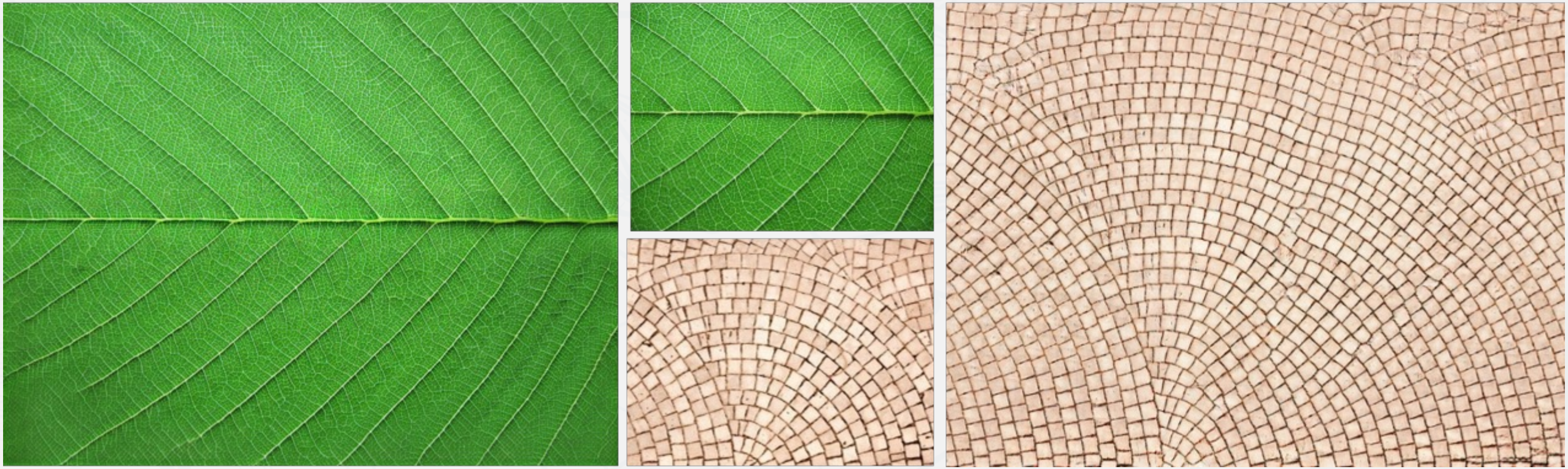


Kaspar et al. 2015

# 纹理的合成

Q: 纹理合成和超分辨的区别？

A: 纹理合成不改变他的尺度。



Zhou et al. 2018

# 纹理的合成

缺点：纹理合成算法通常有使用场景的限制，它并不是在每张纹理上都表现的很好。

肉眼可见的瑕疵

低分辨率，1K的分辨率不够用

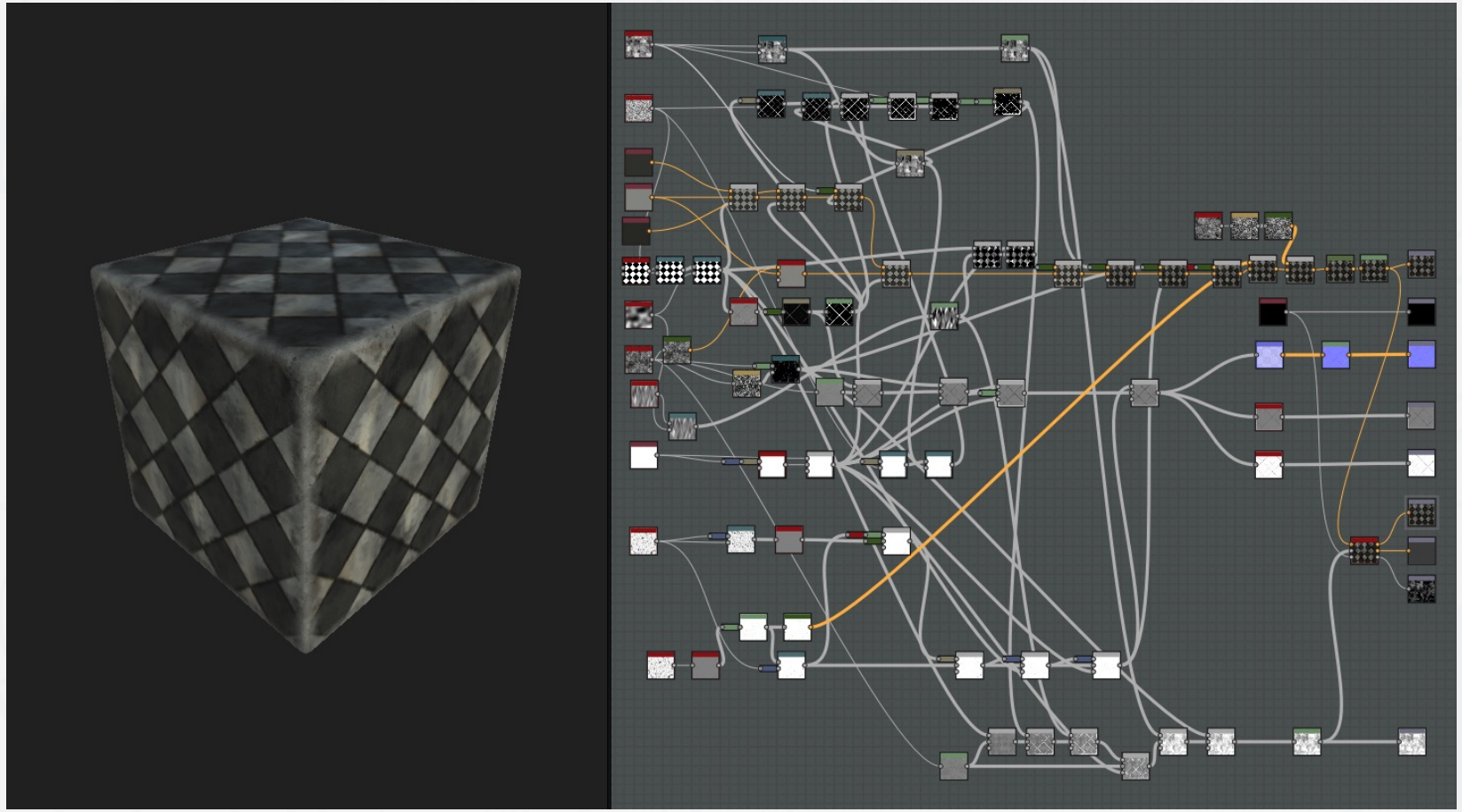
# 过程式纹理生成

优点：
1. 完全的可编辑性

2. 无限分辨率，无限尺度的细节

3. 交互式的设计

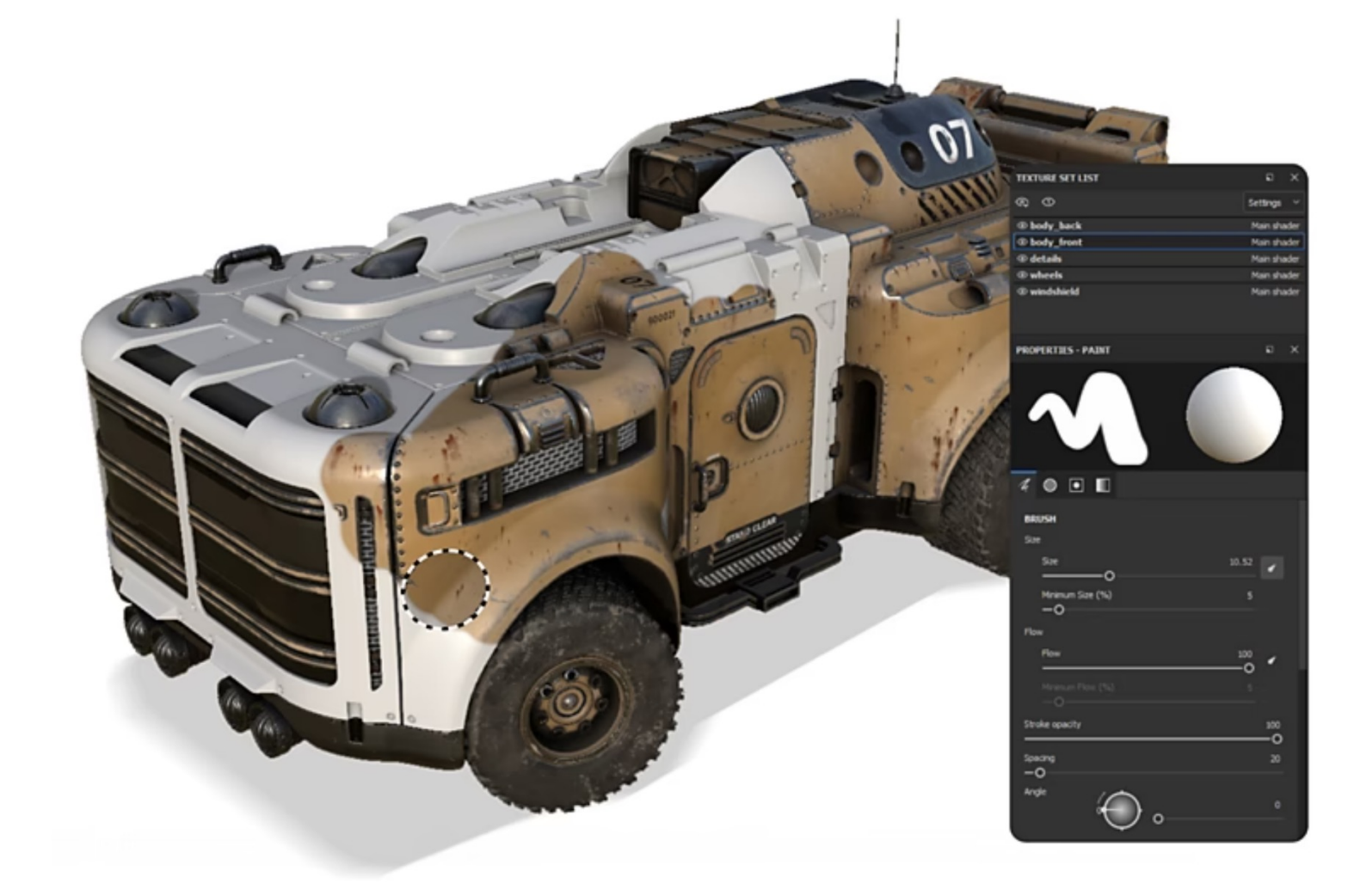
缺点：
1. 设计难度大



Adobe Substance 3D Designer

# 过程式纹理生成

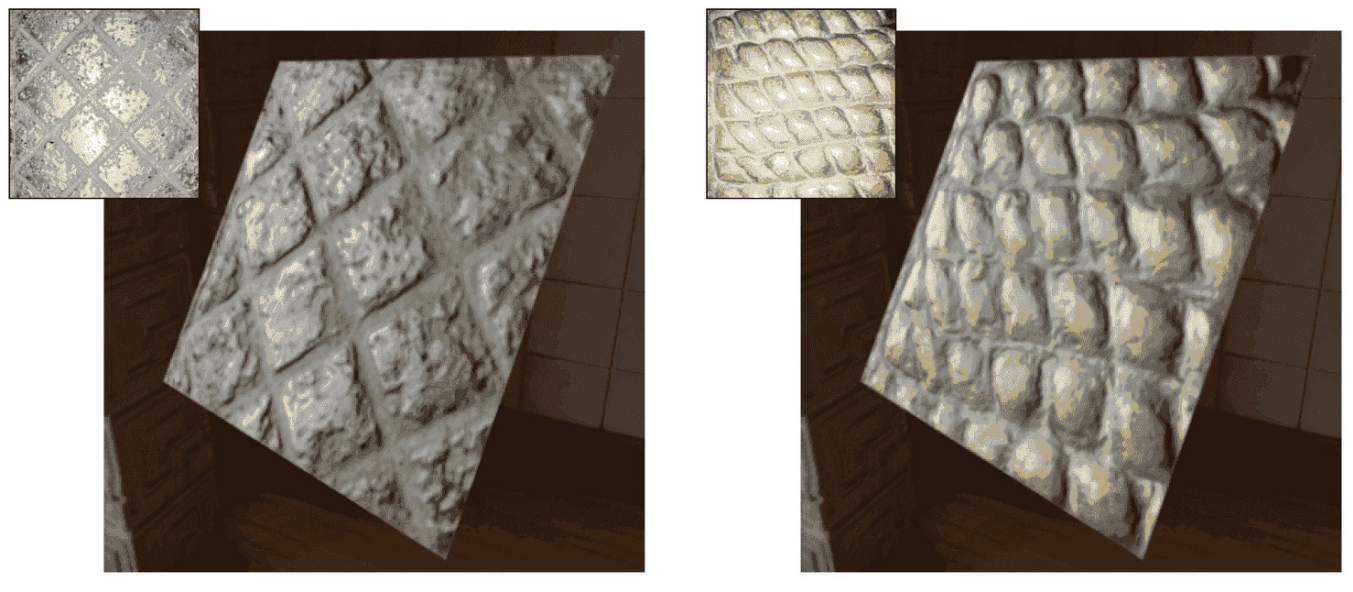
过程式纹理以及贴图




Substance Painter

# 从真实世界中获得纹理

从真实世界中获取材质纹理是非常直观的方式。自从深度学习的流行，大量算法研究如何从单张或者少数几张图片中提取材质，包括Albedo, normal，roughness，metallic等各种纹理信息

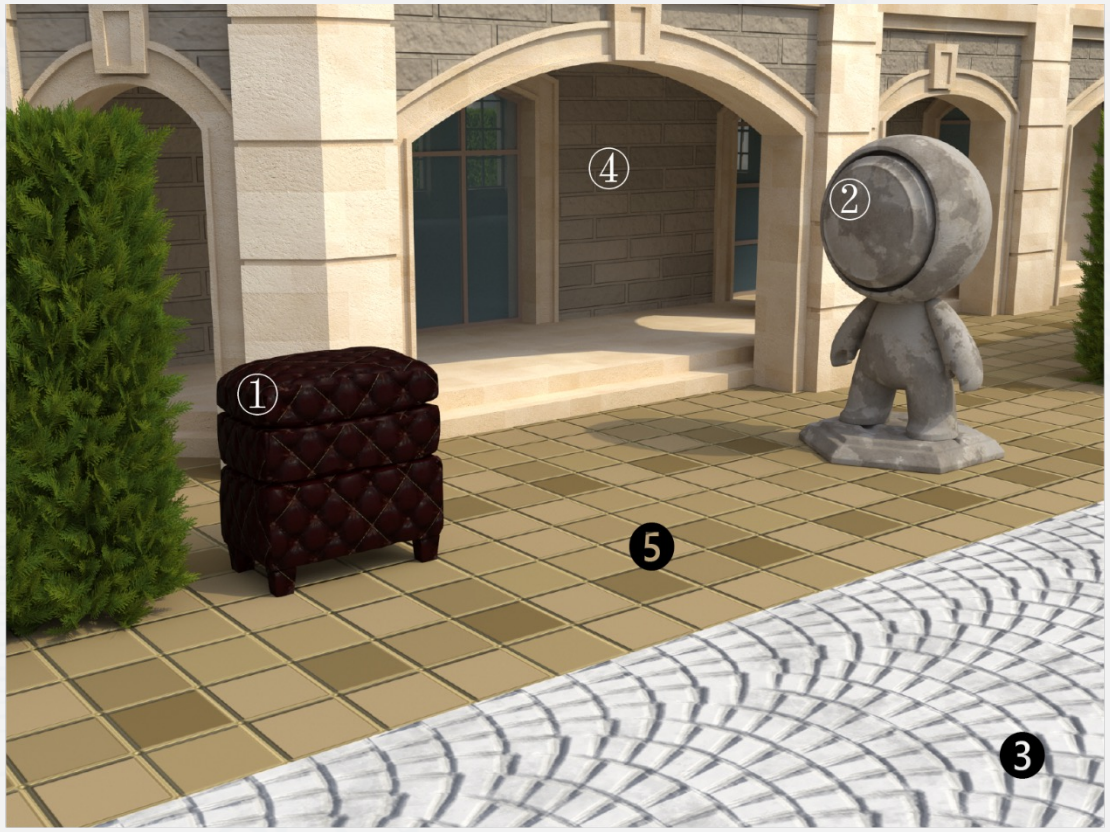


单张图片的材质纹理生成 Deschaintre et al. 2018

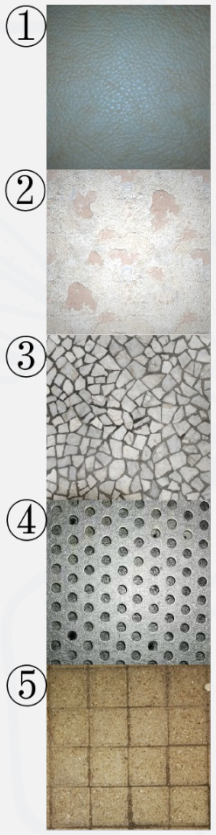
几何与纹理重建 Luan et al. 2021

# 从真实世界中获得纹理

利用真实图片对材质纹理进行编辑（Hu et al. 2022）




Scene with Original Materials




Target Photos

Scene with Augmented Materials

# 材质纹理与AI

@Stan Brown

利用DALL-E2与Substance 3D生成材质纹理

# 材质纹理与AI

@Stan Brown

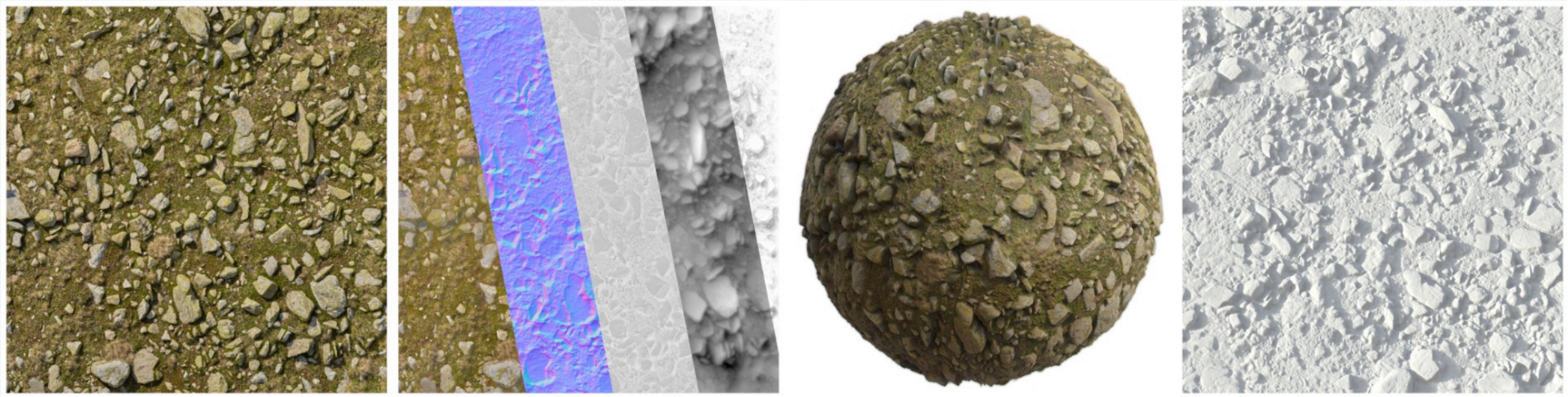
利用Substance Sampler将无缝照片转换为PBR材质图

# 材质纹理与AI

Diffusion Models

# 材质纹理与AI

将Diffusion model得到的纹理，再通过材质图生成工具获得对应的PBR材质图

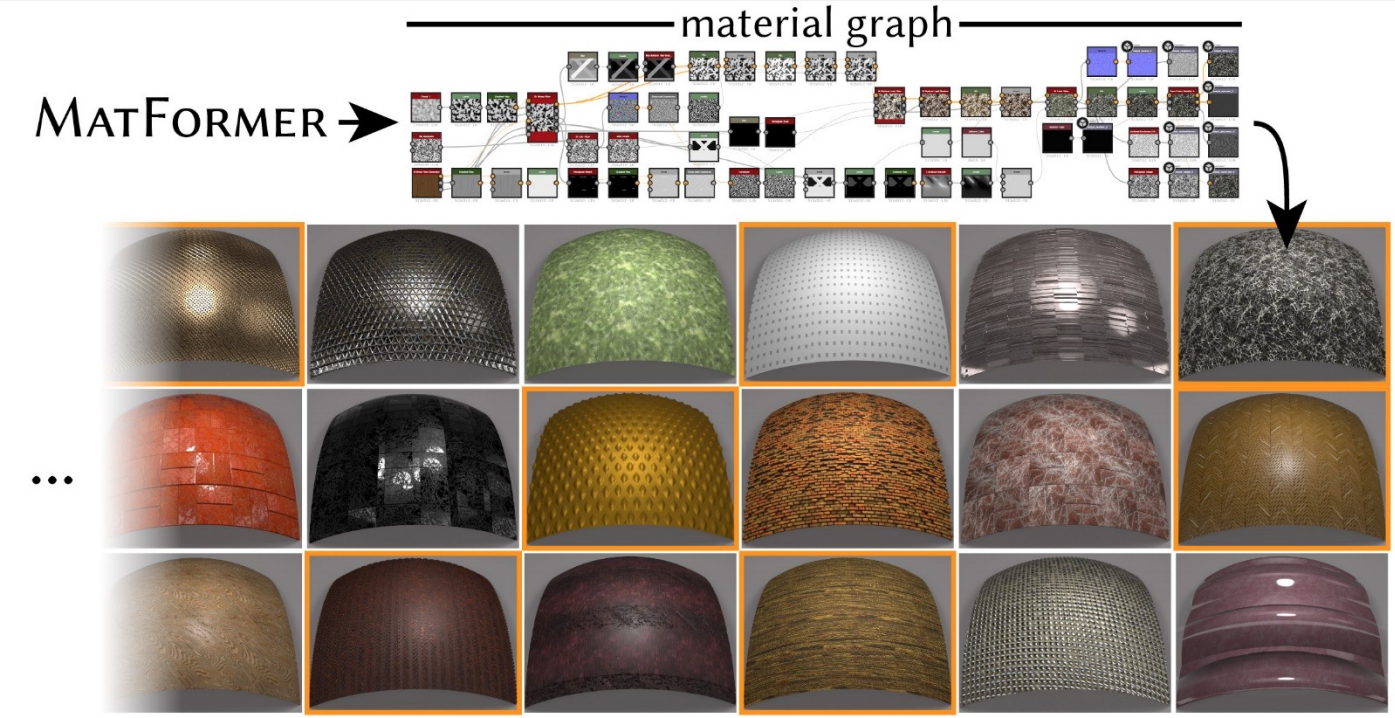


Diffusion Models + 材质图生成工具

# 过程式纹理与AI

Q: 过程式纹理本质是一段程序。能否用直接生成程序呢？

A: Transformer架构(类GPT)的计算图生成式模型



Guerrero et al. 2022

# 纹理使用与压缩

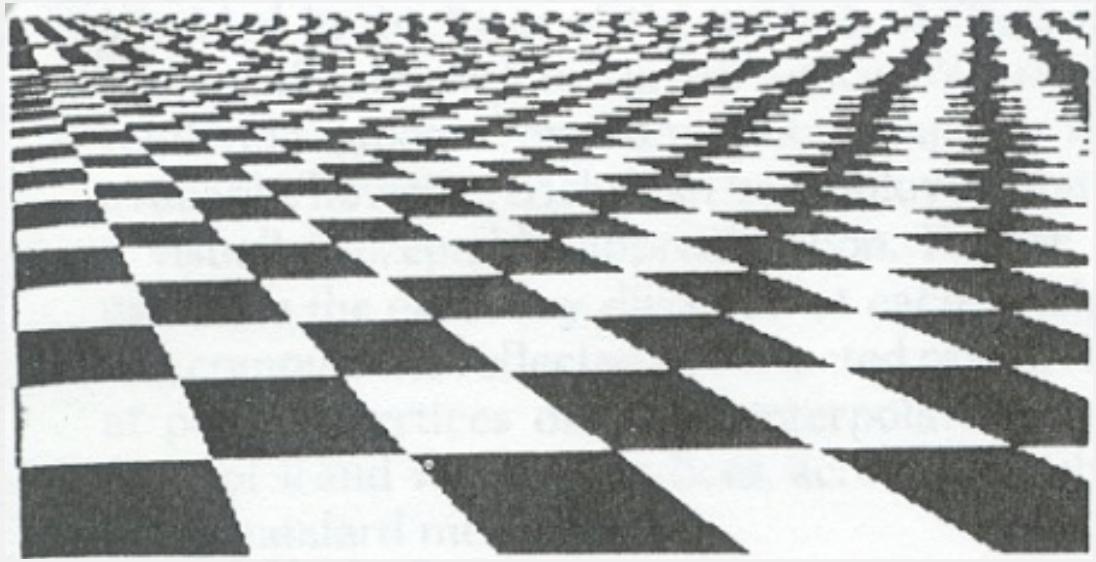
# 如何正确的使用纹理

纹理的使用和正常的数组是不一样的，我们认为纹理所储存的是一个经过离散化的连续的信号。我们把对纹理的访问，认为是对连续信号的采样。通常需要使用小数作为索引，并且需要考虑的采样频率的问题。

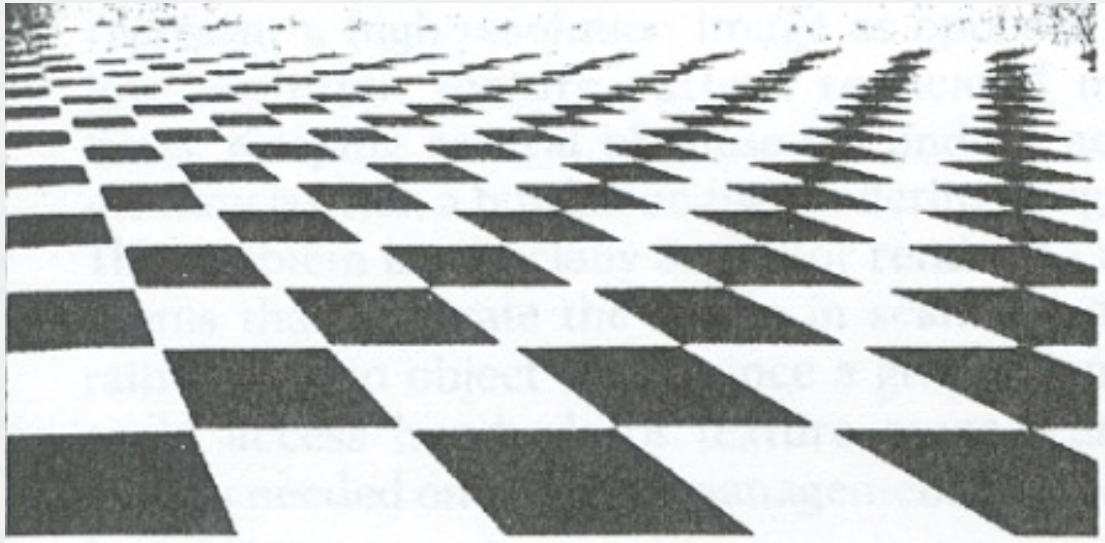
纹理的采样频率会极大影响纹理是质量。如果采样频率不足，那么就会产生走样(aliasing)。

- 通常远方的纹理上面的texel分布越密集，信号频率越高。

增大采样率可以解决这个问题，但这意味着增加采样的数量，会导致性能下降，那么如何解决呢？



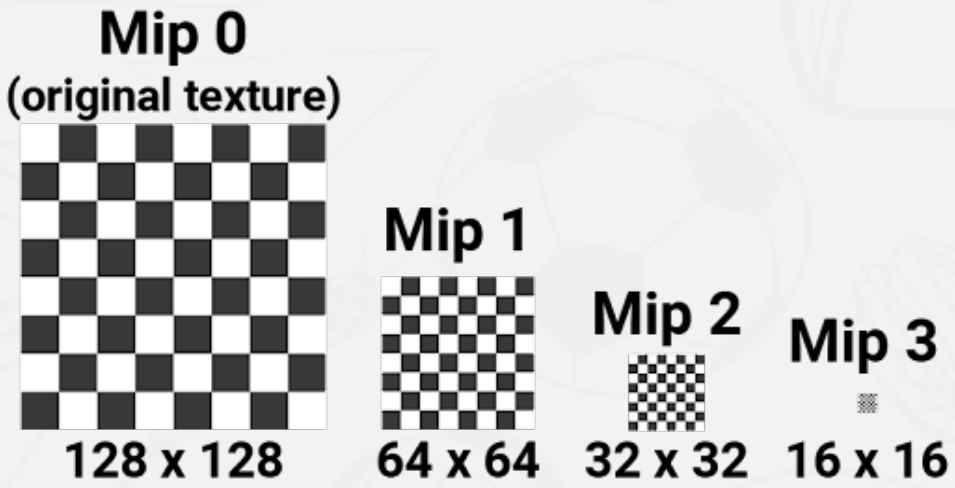
点采样

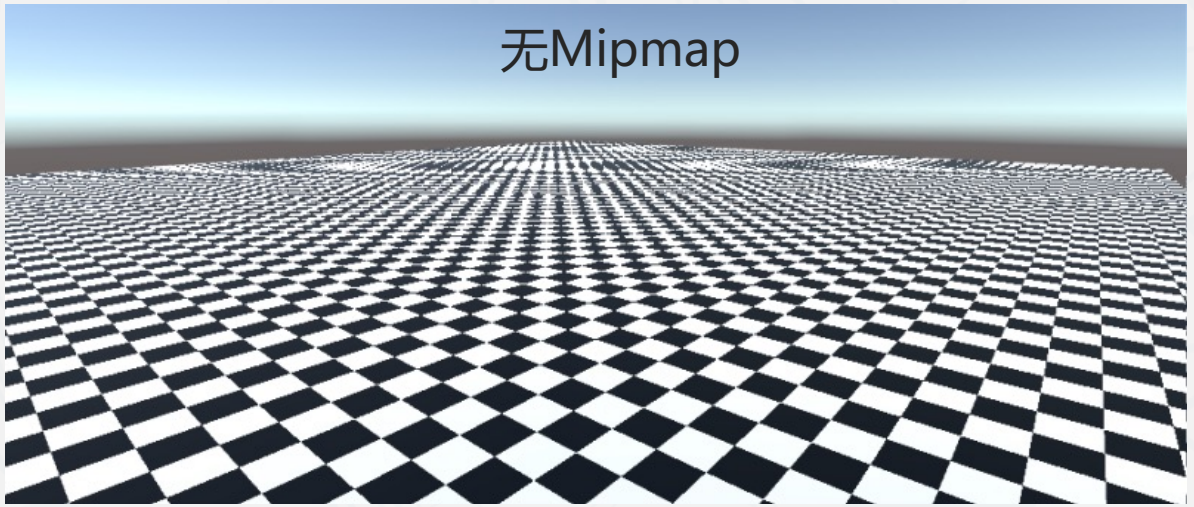


面采样

# 纹理滤波与Mipmap

一般情况下，纹理在运行时固定的，其中储存的值不会在执行渲染时发生变化。因此我们可以提前进行预滤波（pre-filtering），从而进行抗走样。



我们可以在mipmap上再进行采样：

- 最近邻（nearest neighbor) [代价最小，最差质量]
- 双线性 （bilinear）在同一Mipmap进行插值
- 三线性 （trilinear）在相邻的2层mipmap中各采样
- 各向异性 （anisotropic）[最高质量]

# 纹理滤波与Mipmap

Mipmap通常可以由API自动生成并且需要一定的额外储存空间。注意：这需要纹理的维度是2的n次方。

问题：如何知道我们应该采样那一张Mipmap？

第N级mipmap是第N-1级mipmap的1/4，因此mipmap通常需要额外1/3的存储空间

# 纹理滤波与Mipmap

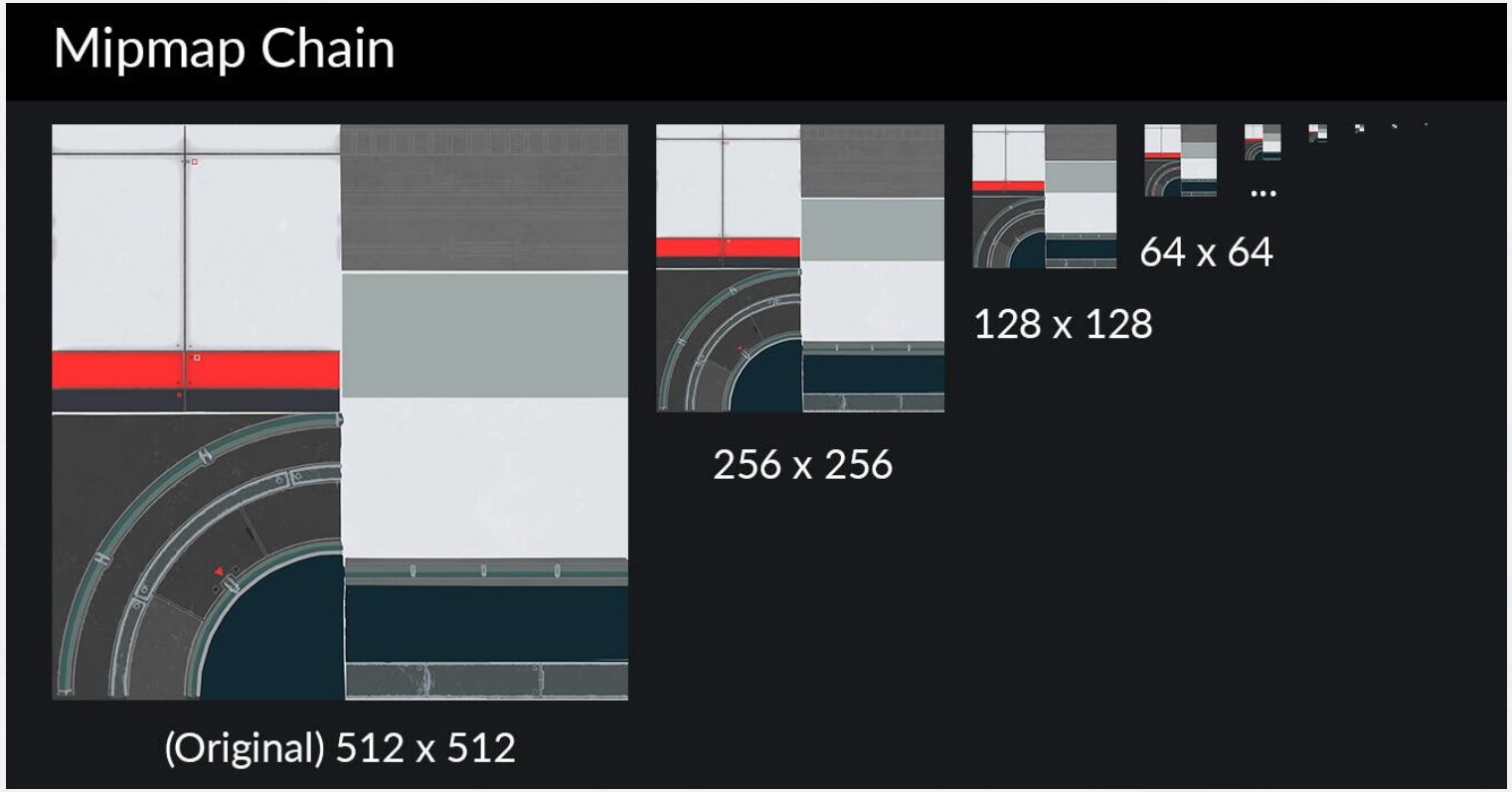
通常来说，GPU执行pixel shading时是以2x2的单位进行并行计算的，从而方便计算Screen-space梯度。以x方向举例，2x2的片元(fragment)可以帮助硬件自动计算纹理坐标（u,v）的梯度（$\frac{du}{dx},\frac{dv}{dx}$）。从而可以估算并选择对应的mipmap等级。

$$P_x = \log_2\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}\right)$$

或者

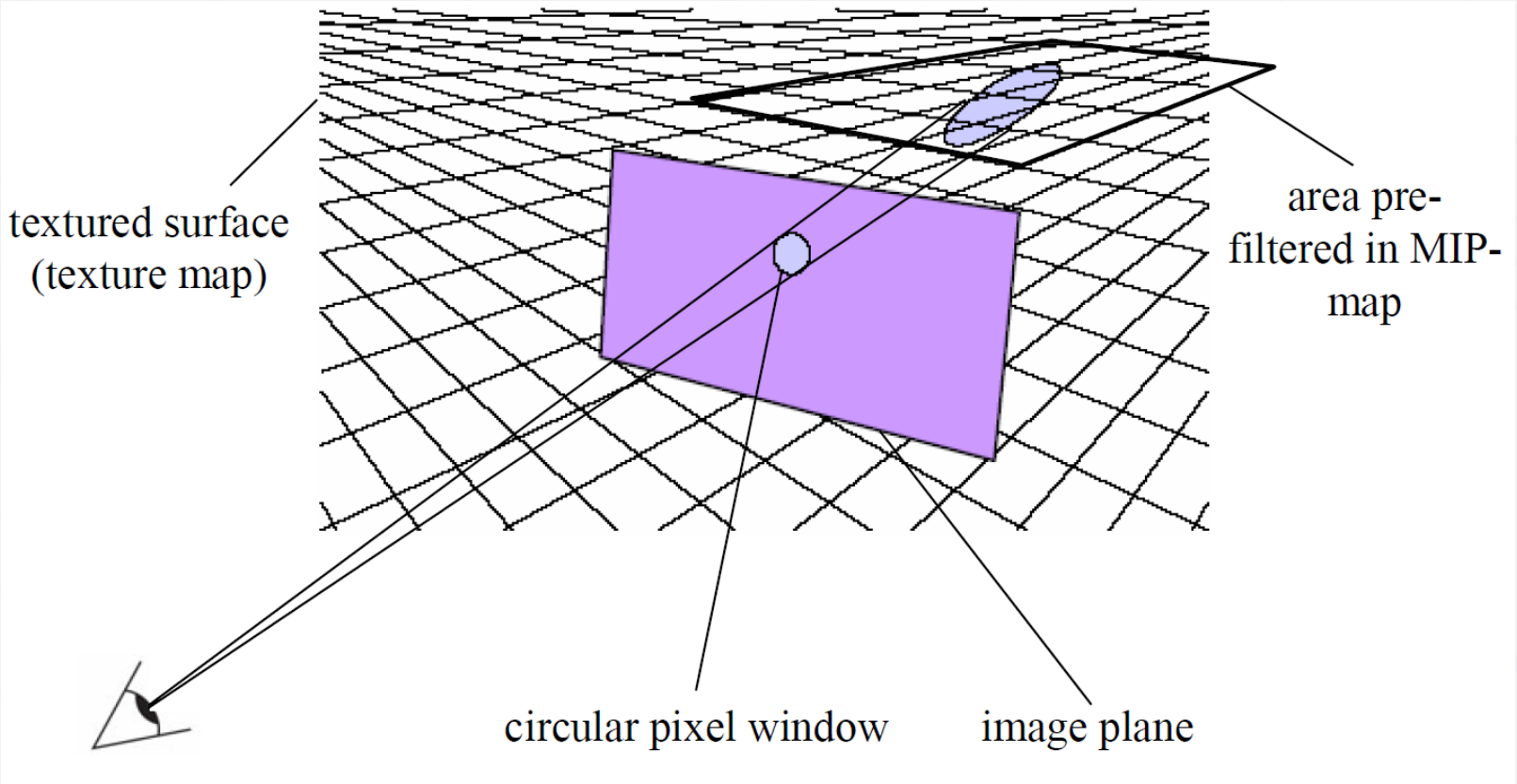
$$P_x = \log_2\left(\max\left(\left|\frac{du}{dx}\right|, \left|\frac{dv}{dx}\right|\right)\right)$$

屏幕空间梯度越大，频率越大，因此需要采样更大层级的mipmap



# 纹理滤波与Mipmap

一般来说正方形的Mipmap工作的很好，但是在特殊情况下仍然会存在问题。




Credit: MIT

# 纹理滤波与Mipmap

最高质量的采样方法: anisotropic filtering。当视角非常陡的时候，区别会很明显，因为此时View-space和Texture Space的差别很大。一般isotropic filtering假设采样发生在uniform的texture-space。

isotropic　　anisotropic

# 纹理滤波与Mipmap

各向异性采样需要实时地进行采样。
Degree of Anisotropic mipmapping: 采样数越高越清晰，但是需要更高的计算代价

NA　　2X　　4X　　8X

# 纹理的压缩

一张4K 三通道彩色纹理的正常大小：
4096x4096x3=48MB，mipmap的大小则为64MB!
一张4K HDR三通道彩色纹理的正常大小：
4096x4096x4x3=192MB, mipmap的大小则为256MB!

我们需要尽可能的压缩纹理大小！既然纹理是2D图片，我们能够用一般的图像压缩算法吗？比如jpg, png？

# 纹理的压缩

纹理的重要特性：随机的单点访问 (Random Point Query)

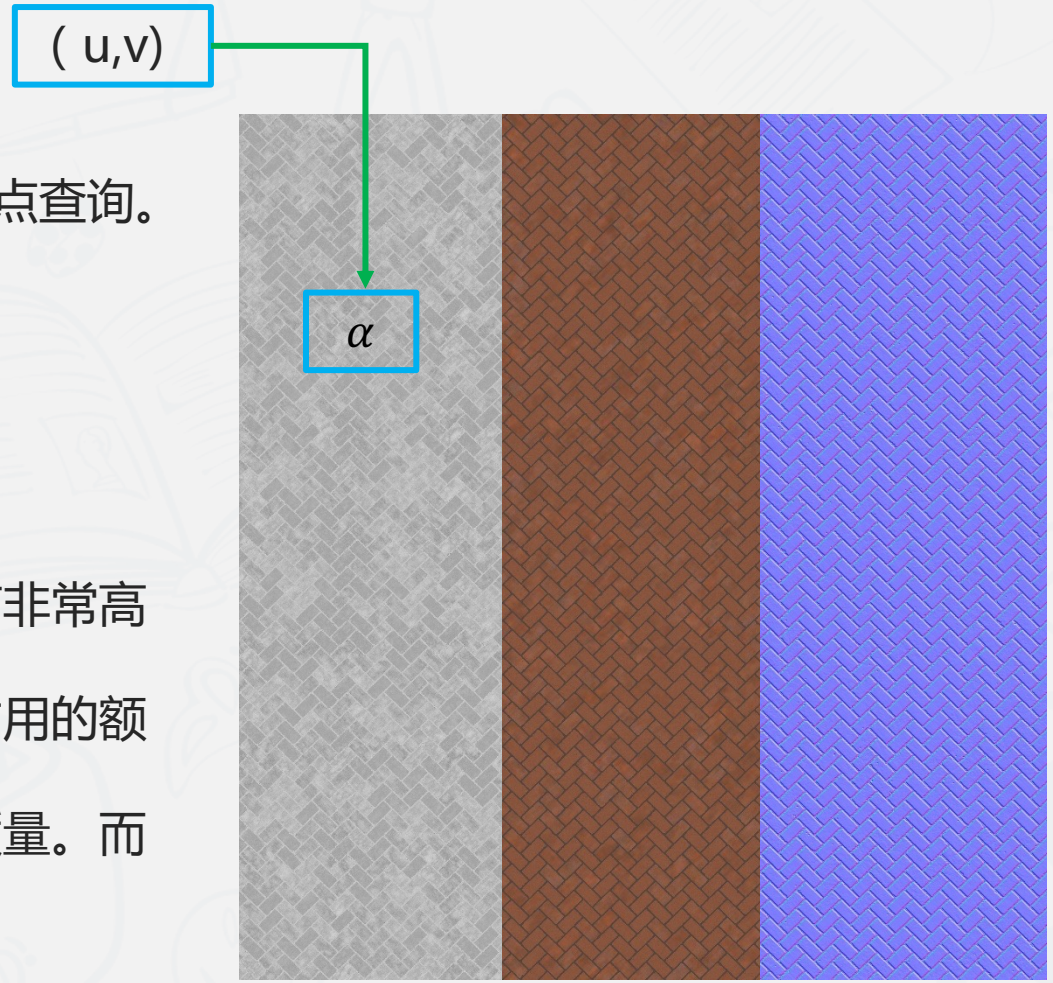
因为片元着色器是并行执行的，我们需要能够对纹理进行快递的单点查询。
也就是说，我们需要有一个查询算法：

输入：u,v坐标
输出：纹理值

纹理压缩的要求：

- 解码速度：纹理的解压过程需要在每一帧中进行，因此对速度有非常高的要求。
- 随机访问：需要对纹理进行快速的单点查询。并且查询过程中占用的额外资源很小。
- 可变的压缩比：移动设备通常需要更大的压缩比，但是会牺牲质量。而桌面级GPU有着更大的显存，因此希望提供更好的质量。

因此传统的图片算法无法应用到纹理中。



# 纹理的压缩

纹理压缩算法：一般是基于block的压缩算法：

1. S3 Texture Compression (S3TC)
2. PowerVR Texture Compression (PVRTC)
3. Ericsson Texture Compression (ETC)
4. Adaptive Scalable Texture Compression (ASTC)

将一组像素块压缩为固定长度的向量。比如ASTC将不同数量像素的数据压缩为128bits的向量。

8bits 3.56bits 2bits

Adaptive Scalable Texture Compression (HPC 2012)

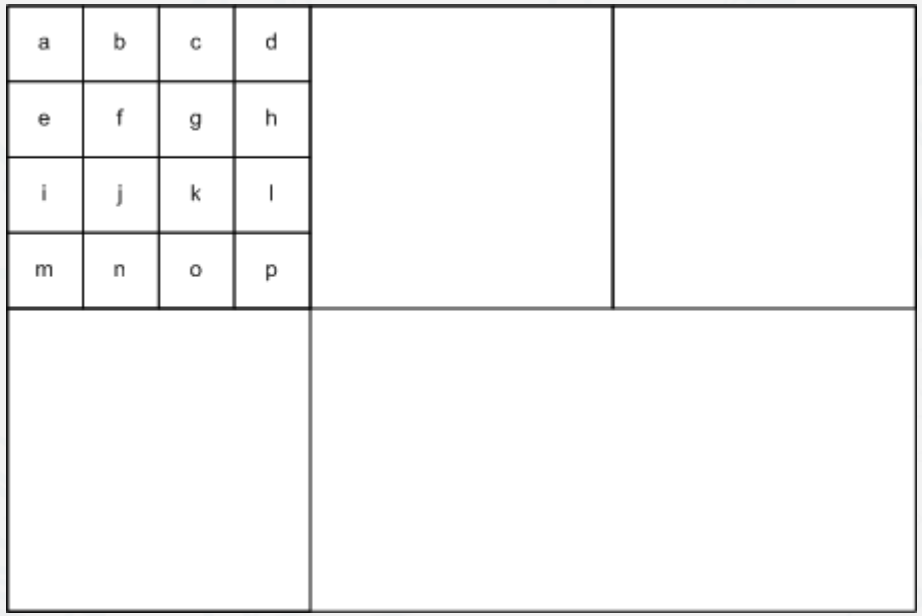
| Block footprint | Bit rate | Increment |
|---|---|---|
| 4×4 | 8.00 | 25% |
| 5×4 | 6.40 | 25% |
| 5×5 | 5.12 | 20% |
| 6×5 | 4.27 | 20% |
| 6×6 | 3.56 | 14% |
| 8×5 | 3.20 | 20% |
| 8×6 | 2.67 | 5% |
| 10×5 | 2.56 | 20% |
| 10×6 | 2.13 | 7% |
| 8×8 | 2.00 | 25% |
| 10×8 | 1.60 | 25% |
| 10×10 | 1.28 | 20% |
| 12×10 | 1.07 | 20% |
| 12×12 | 0.89 | |

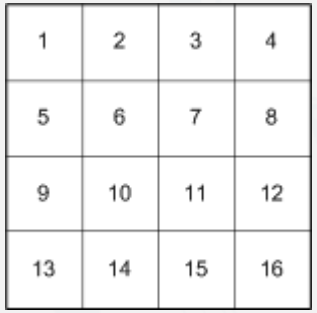
ASTC算法的压缩比

# 纹理的压缩

样例：

一个纹理被分为多个数据块




一个4x4的数据块

未压缩状态 (16Bytes)

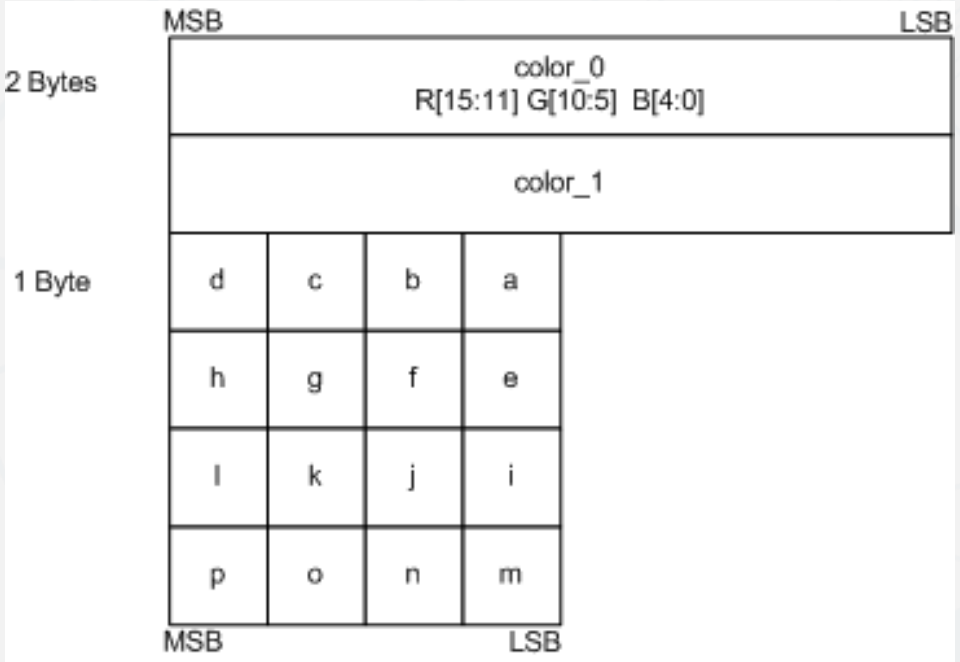
压缩状态 (8 Bytes)

# 纹理的压缩

样例（BC1):

压缩1个4x4的RGB数据块：

- 2个颜色值(color_0, color_1) [2Byte/each]
- 16个颜色索引 [2bit/each]
- 一共 8 Bytes，压缩比6:1




编解码：表达4种颜色

1. color_0 = color_0
2. color_1 = color_1
3. color_2 = 2/3*color_0 + 1/3*color_1
4. color_3 = 1/3*color_0 + 2/3*color_1

对应4个索引

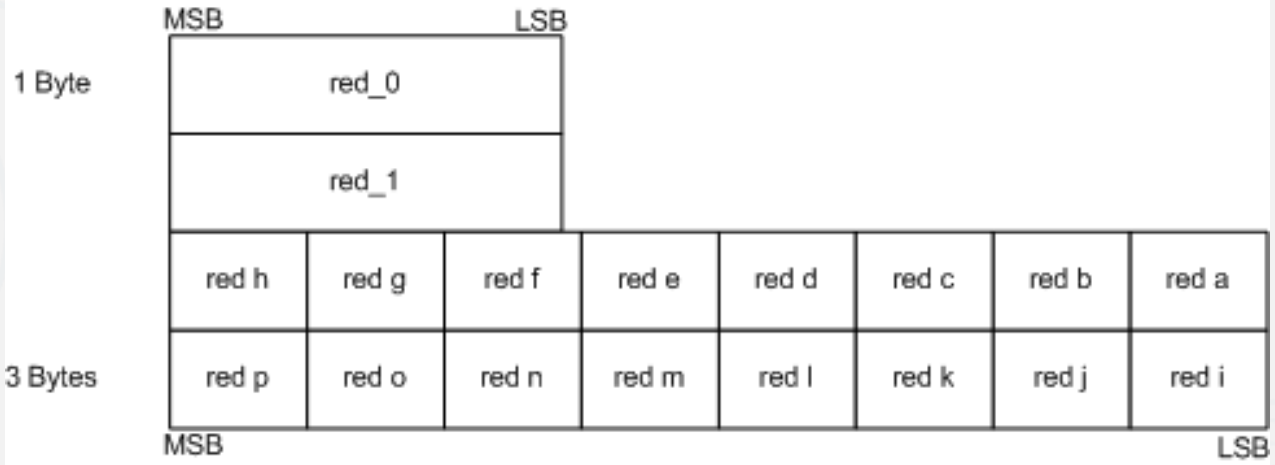
1. color_0 = 00
2. color_1 = 01
3. color_2 = 10
4. color_3 = 11

# 纹理的压缩

不同的数据类型会有不同的编码方式：

| Source Data | Data Compression Resolution (in bits) |
|---|---|
| Three-component color and alpha | Color (5:6:5), Alpha (1) or no alpha |
| Three-component color and alpha | Color (5:6:5), Alpha (4) |
| Three-component color and alpha | Color (5:6:5), Alpha (8) |
| One-component color | One component (8) |
| Two-component color | Two components (8:8) |

BC1~BC5

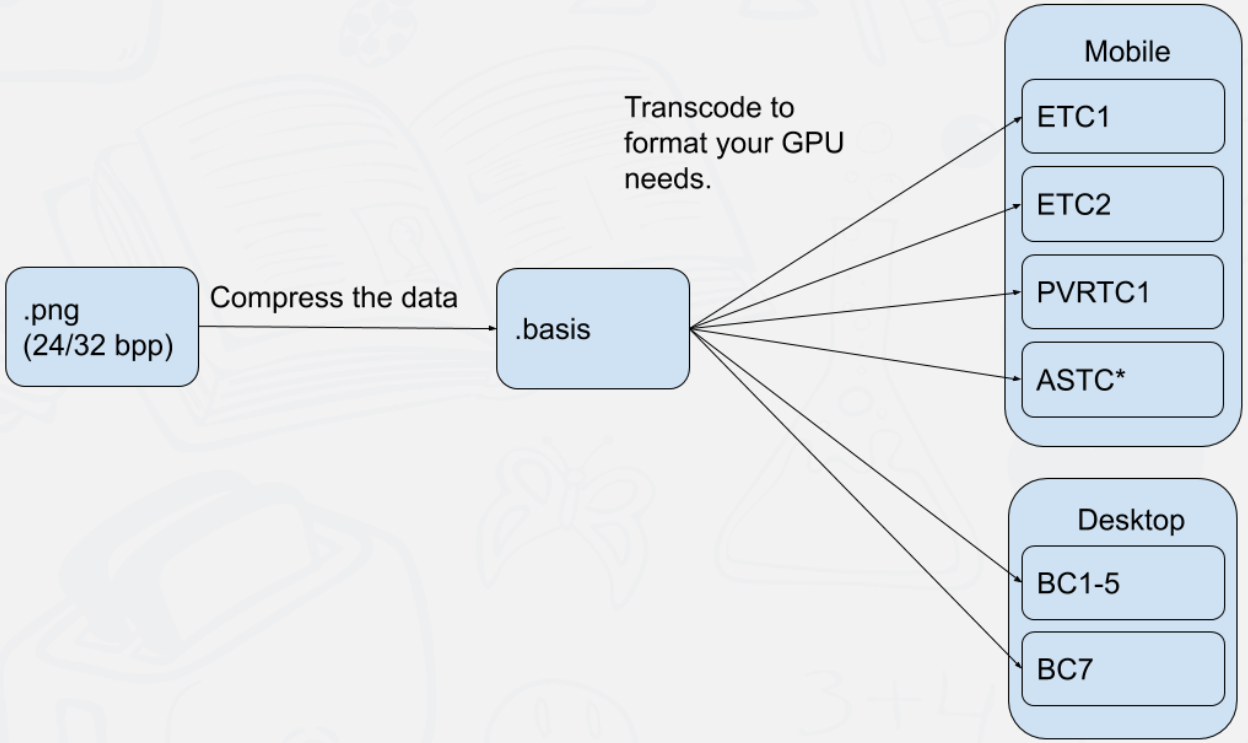



单个颜色编码：BC4

# 纹理的压缩

压缩方法很多，多种编解码方式如何选择？

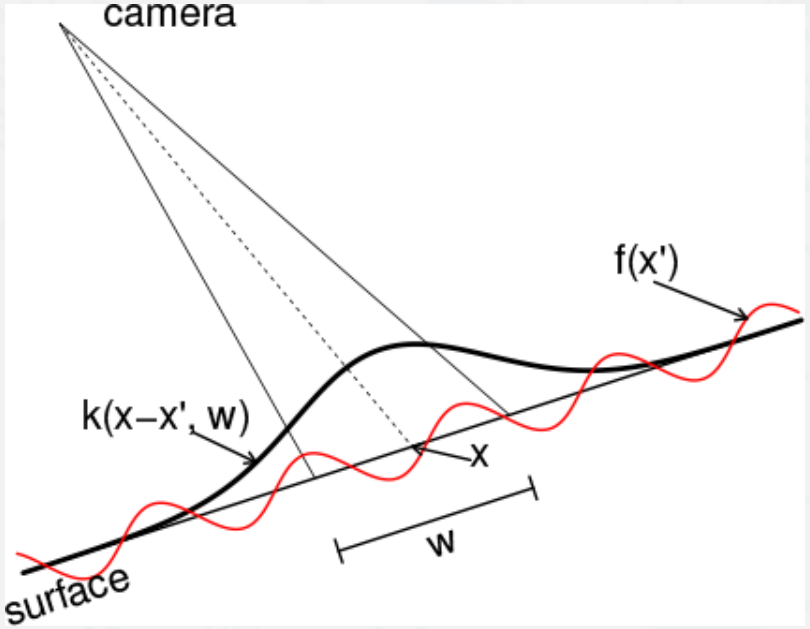
Basis Universal: 一种纹理数据交换系统，生成中间压缩数据(.basis)，并能快速的转编码为其他各种压缩类型。

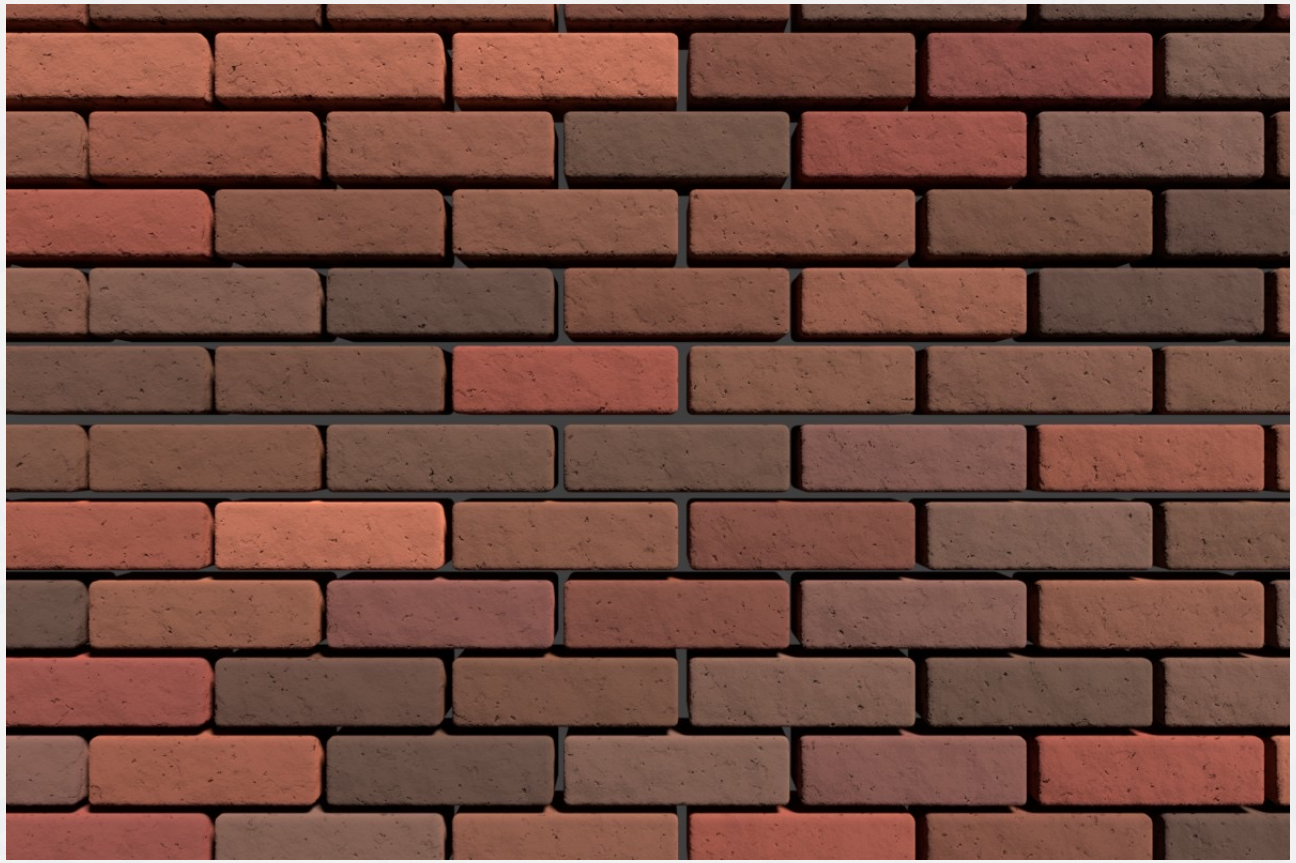


# 纹理的压缩

最极限的压缩：过程式纹理

优势：0储存空间
缺点：高昂的解码代价，以及无法提前计算Mipmap，很可能需要实时进行滤波。



过程式纹理

解析预滤波（band-limited analytical prefiltering）

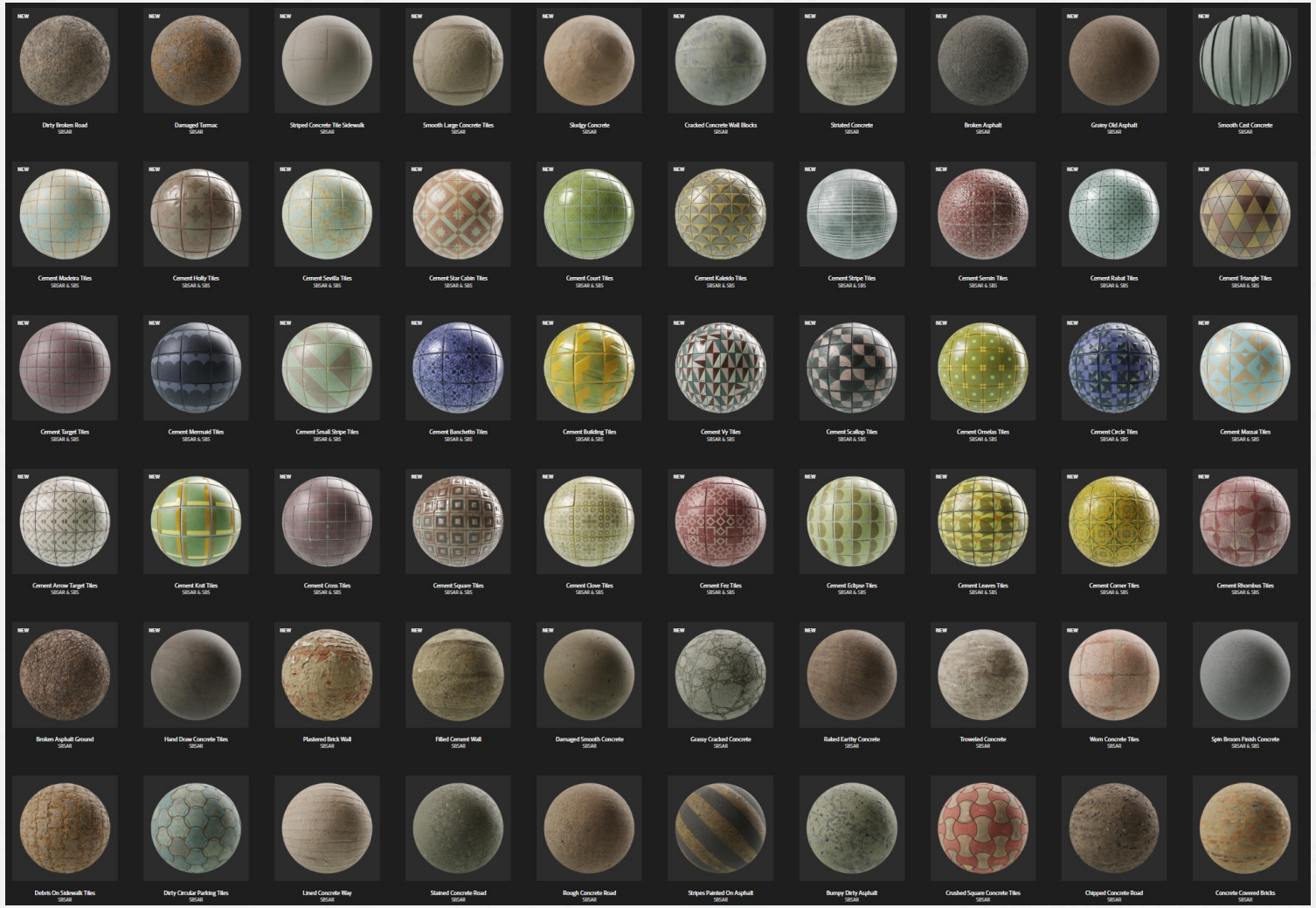
# 总结

# 课程总结

1. 什么是纹理
   - 纹理的表达
   - 纹理与材质的关系

2. 纹理的生成
   - 设计工具
   - 过程式纹理
   - 算法与AI

3. 纹理的使用与压缩
   - 纹理的滤波与Mipmap
   - 纹理的压缩算法



# 答疑环节