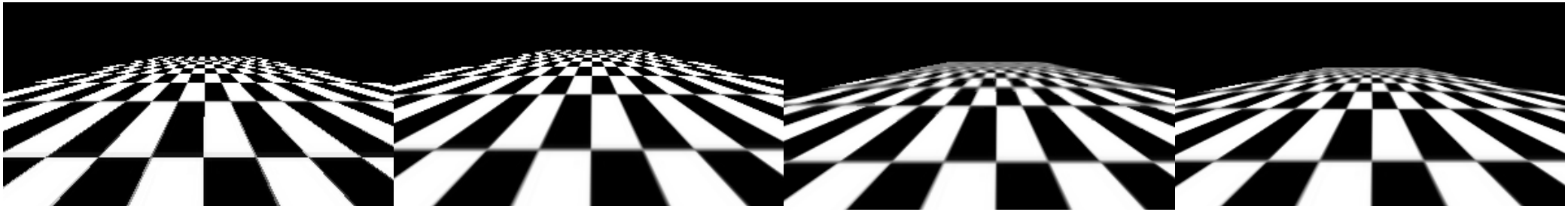
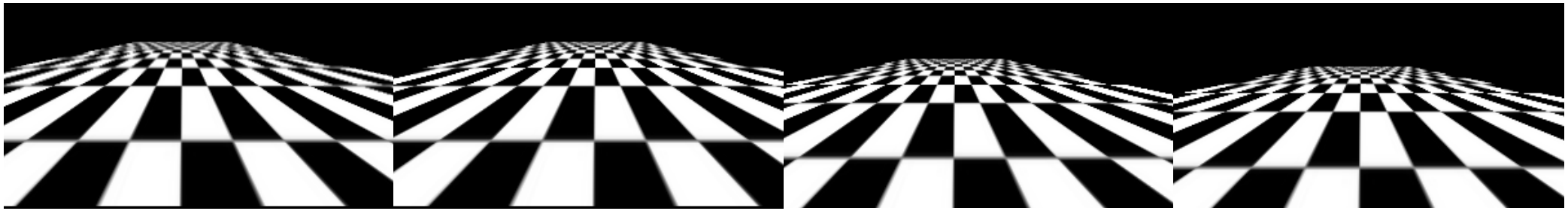
NEAREST BILINEAR BILINEAR w. MIPMAP TRILINEAR



2x ANISOTROPIC 4x ANISOTROPIC 8x ANISOTROPIC 16x ANISOTROPIC



二维纹理的过滤

2-D Texture Filtering

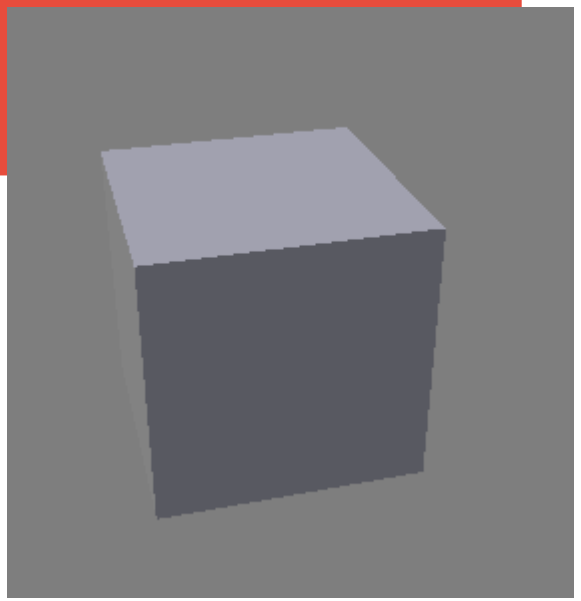
Haoran Luo
Nanjing University

课程目标

- 了解实时渲染中的常用纹理过滤方法的原理
- 比较不同纹理过滤的画面效果及运行效率和空间占用
- 基于 OpenGL 着色语言实现纹理过滤算法
- 初步掌握 OpenGL 着色语言的使用方法，为后续课程的实现更复杂的画面效果奠定基础

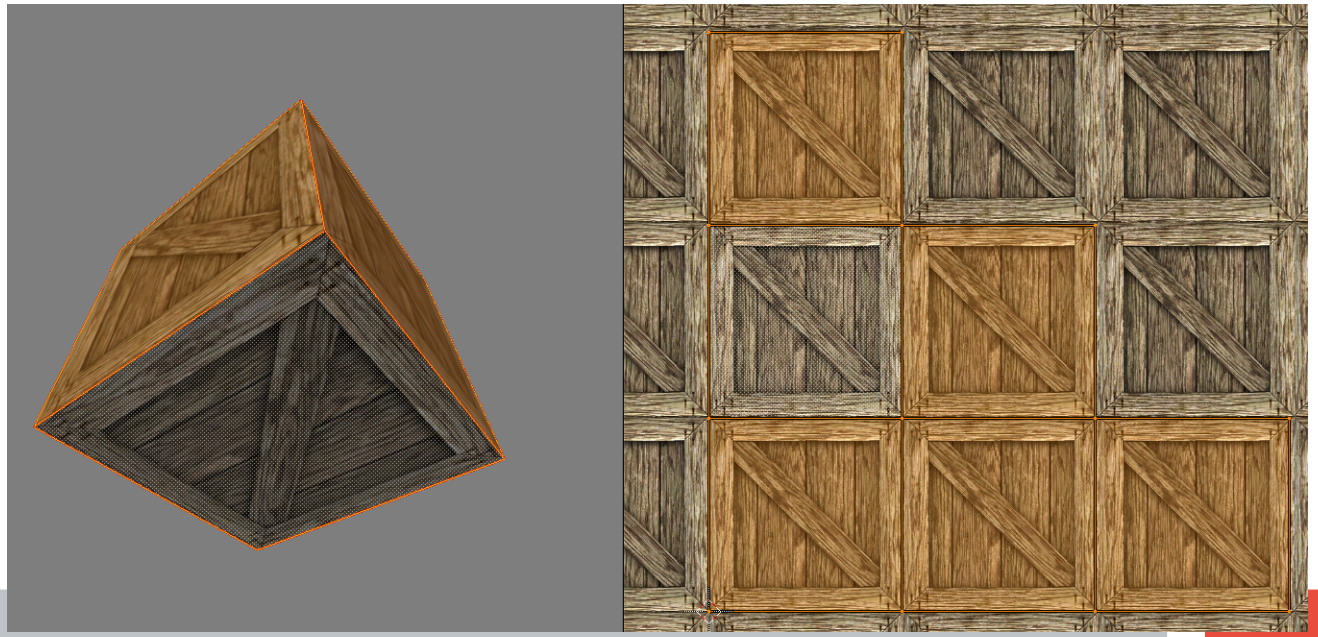
纹理 Texture

- 影响用户对场景中的物体的认知
  - 立方体？木箱？
- 辅助其他渲染效果的实现（各种贴图）
  - 凹凸贴图——表面偏移
  - 法线贴图——表面光照
  - 深度图——投射阴影
  - BRDF 贴图——基于测量数据的光照等
- 多重纹理：更复杂的渲染效果



纹理映射 Texture Mapping

- 屏幕空间（像素， Pixel ）与纹理空间（纹素， Texel ）的对映关系
- 复杂场景中的物体往往经过各种变换
- 模型变换、视图变换、投影变换、透视除法、视口变换等
- 图元的顶点往往被指定不同的纹理坐标（ UV 展开等）



纹理过滤 Texture Filtering

- 纹素与像素的一一对应往往只是理想情况
  - 缩小过滤（minification）：多个纹素可能映射到一个像素
  - 放大过滤（magnification）：一个纹素可能映射到多个像素
- 完美的过滤方法在实时渲染环境下往往代价高昂
- 是一个渲染效率、空间与画面效果的折中权衡的过程

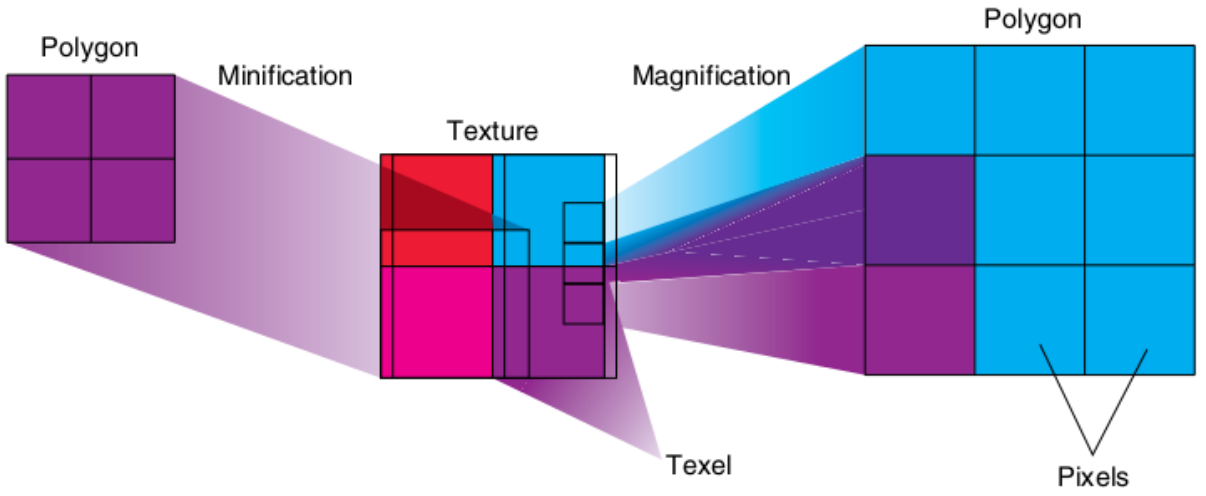


Figure 9-8 Texture Magnification and Minification

常用符号

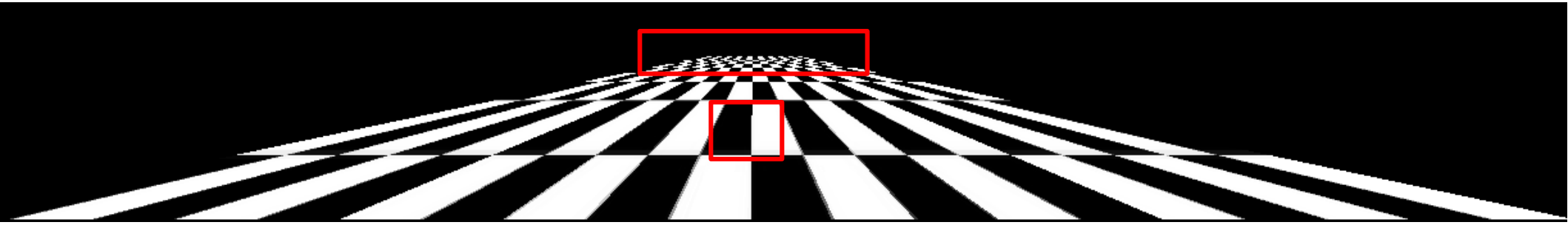
- 如无特殊说明，下文中将会使用到的符号如下：
- 纹理坐标：$P(x,y)$表示像素 (x, y) 对应的纹理坐标。对一个 2-D 纹理而言，$0 \le x < 1, 0 \le y < 1$（注意范围）
- 特别地，在描述单个像素的纹理过滤时，用 P 表示当前像素点对应的纹理坐标。
- 纹理图像大小 $\|\tau\|$，如 512x512 的纹理该值即为 (512, 512)
- 纹素访问：$\tau(u,v)$表示纹理图像中 (u, v) 点处对应的灰度或颜色，其中 $0 \le u < \|\tau\|.x, 0 \le v < \|\tau\|.y$

最近点过滤 Nearest Point Filtering

- 最简单易于理解的过滤方式
- 给定纹理坐标点 P，先与纹理大小相乘，得到

$$(u,v)=(\|\tau\|.x\times P.x,\|\tau\|.y\times P.y)$$

- 将 u 分量和 v 分量四舍五入，得到最近点坐标，并将纹理中该坐标的值作为纹理过滤的输出
- 只需要一个采样点，速度最快，空间占用最少
- 画面效果奇差（随处可见的锯齿、莫尔纹、……）



双线性过滤 Bilinear Filtering

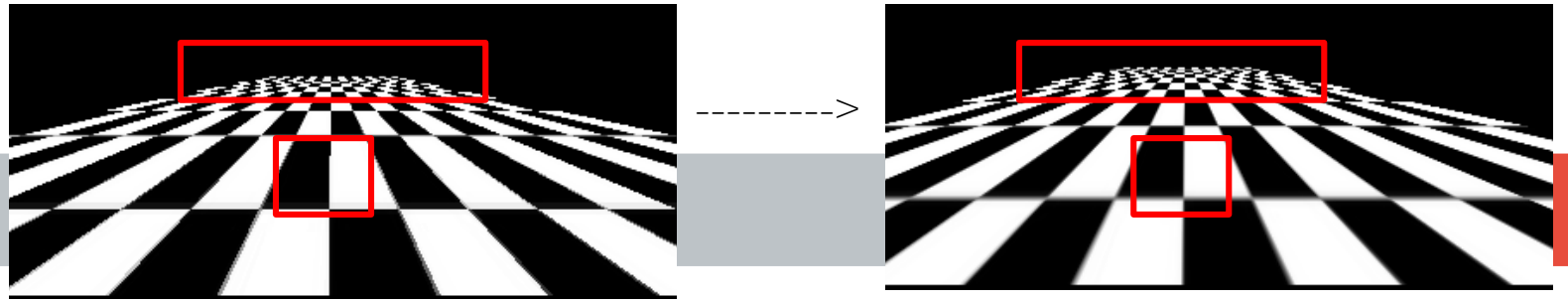
- 使用线性插值的方法平滑锯齿（因为是二维的插值所以叫双线性）
- 仍以相乘的方式得到 (u,v) 坐标
- 令 $u_0=\lfloor u \rfloor, u_1=\lceil u \rceil, \Delta u = u - u_0$

  $v_0=\lfloor v \rfloor, v_1=\lceil v \rceil, \Delta v = v - v_0$
- 双线性插值：

$$(1-\Delta v)\left(\tau(u_0, v_0)(1-\Delta u)+\tau(u_1, v_0)\Delta u\right) + \Delta v\left(\tau(u_0, v_1)(1-\Delta u)+\tau(u_1, v_1)\Delta u\right)$$

- 需要四个采样点且不需要额外空间
- 在放大或不多于两倍缩小时效果显著
- 在多于两倍缩小时退化为点过滤（思考为什么？）

Nearest ---------> Bilinear



双线性过滤 Bilinear Filtering

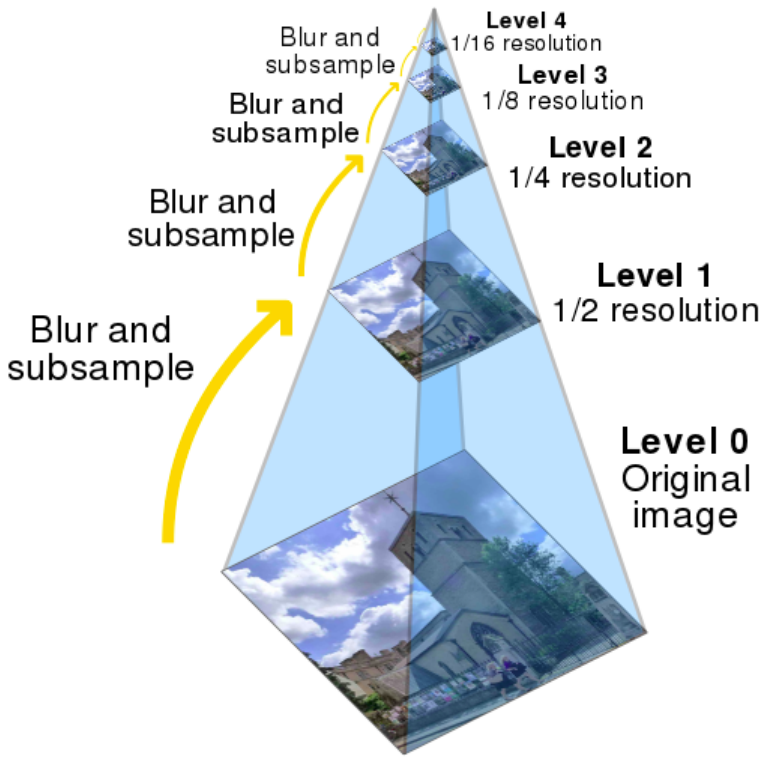
- 缩小倍率多于 2 时，需要更多的采样点才能得到较为理想的结果（？）
- 单纯地依靠采样更多的点，以获得更好的渲染效果的思想，在实时渲染情景下，是极其危险的
- 一个极远处的像素点，包含了大量的纹素，为了这么一个像素点而扩大采样点的数量显然是错误的
- 尝试用空间复杂度的提升换取时间复杂度的降低

Mipmap

- MIP： 拉丁语 multum-in-parvo
- 存储预先计算多个纹素的向下折半采样（一般为平均值）
- 使用 mipmap 时，用下标表示需要访问的 mipmap 层级，其中

$$\tau(u,v)=\tau_0(u,v)$$

- 那么问题来了：如何确定要访问哪个 mipmap 层级?



计算 Mipmap 层级

- 已知 Mipmap 最大层数为 $M=\log_2(max(\|\tau_0\|.u,\|\tau_0\|.v))$
- 如何表示纹理坐标函数 P 在 x 轴和 y 轴方向变化的快慢?

$$P_x(x,y)=\frac{\partial P}{\partial x},P_y(x,y)=\frac{\partial P}{\partial y}$$

- 在某一像素上下文中，直接用 P_x,P_y 表示上述值
- GLSL 提供了 dFdx 和 dFdy 函数作为上述值的近似
- 为了取得足够的采样，应该访问 Px 和 Py 中变化较快的分量，因此要访问的 mipmap 层级应为:

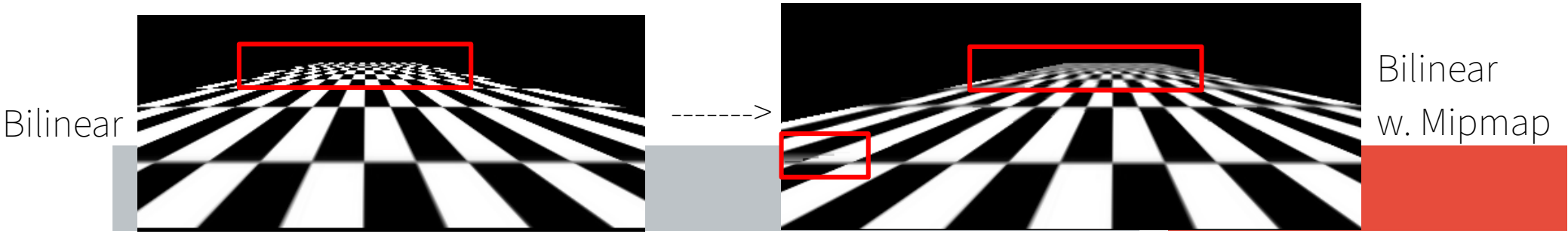
$$N=min(M,M+\log_2(max(\|P_x\|,\|P_y\|)))$$

- 思考： N 的范围是什么？当 N 小于 0 时，代表什么?

使用 Mipmap 的双线性过滤

Bilinear w. Mipmap

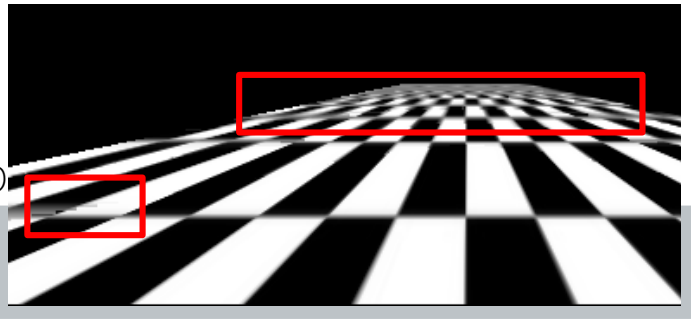
- 先计算 mipmap 层级 N （注意 N 是实数）
- 为了获得足够的采样，应该选取更高的 mipmap 的层级 n ，并进行该层级上的双线性放大过滤，即 $n=\lceil N \rfloor$
- 采样点数不变，用空间复杂度解决了退化为点过滤的问题
- 但是选取不连续的 mipmap 层级会导致 mipmap 层级之间缺乏过度，边界明显
- 为 mipmap 层级之间引入过渡



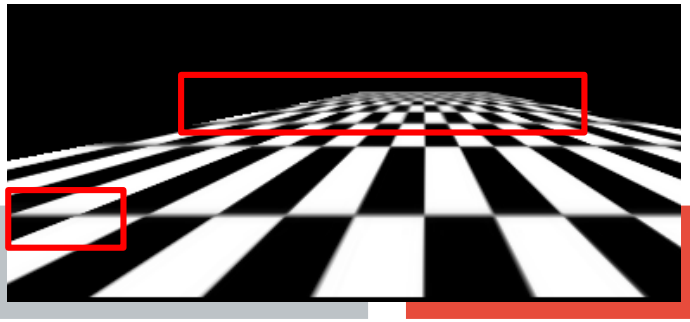
三线性过滤 Trilinear Filtering

- 在双线性过滤的基础上，在计算出来的 mipmap 层级之间进行插值， mipmap 层级是“第三维”
- $n_0 = \lfloor N \rfloor, n_1 = \lceil N \rceil, \Delta n = N - n_0$
- $Bilinear(\tau_{n_0}, u, v)(1 - \Delta n) + Bilinear(\tau_{n_1}, u, v)\Delta n$
- 需要 Mipmap （空间）和八个采样点，解决了层级之间的过渡问题。

Bilinear w. Mipmap



------->



Trilinear

各向同性的问题 Problem of Isotropic

- 迄今为止，我们使用的采样窗和 mipmap 层级计算仅考虑了 x 和 y 中纹理坐标变化率较大的分量。
- 仅仅以 x 和 y 中变化率较大的分量作为采样 / 过滤的依据的方式我们称之为各向同性（ Isotropic ）。
- 事实证明，这种采样的方式会因为简便地选取了细节级别较低的层级（即更高的 mipmap 层级）而的模糊。
- 将 x 和 y 的差别纳入考虑如何?

各向异性过滤 Anisotropic Filtering

- 计算 x 和 y 方向纹理变化率较大和较小的分量
- $P_{max}=\|P_x\|>\|P_y\|?P_x:P_y, P_{min}=\|P_x\|<\|P_y\|?P_x:P_y$
- 求出采样窗口的各向异性度（Degree of Anisotropy）

$$\lambda=min(\frac{\|P_{max}\|}{\|P_{min}\|},\lambda_{max})$$

- λ_{max} 是最大各向异性度，人为设定，取值一般为二的乘方。
- 将以 λ_{max} 为参数的过滤称为 λ_{max} 倍各向异性过滤
- 最大各向异性度保证了该过滤算法在 $\frac{\|P_{max}\|}{\|P_{min}\|}$ 很大时的运算时间有边界。

各向异性过滤 Anisotropic Filtering

- 先不考虑选择哪一 Mipmap 层级，假设经过某一步骤求得 Mipmap 层级为 N，那么当 $\lambda \neq 1$ 时，有

$$S_n = \frac{1}{\lambda} \sum_{i=0}^{\lambda-1} Bilinear\left(\tau_n, P + \left(\frac{i}{\lambda - 1} - \frac{1}{2}\right) P_{max}\right)$$

（思考，当 $\lambda = 1$ 时，应该如何处理？）

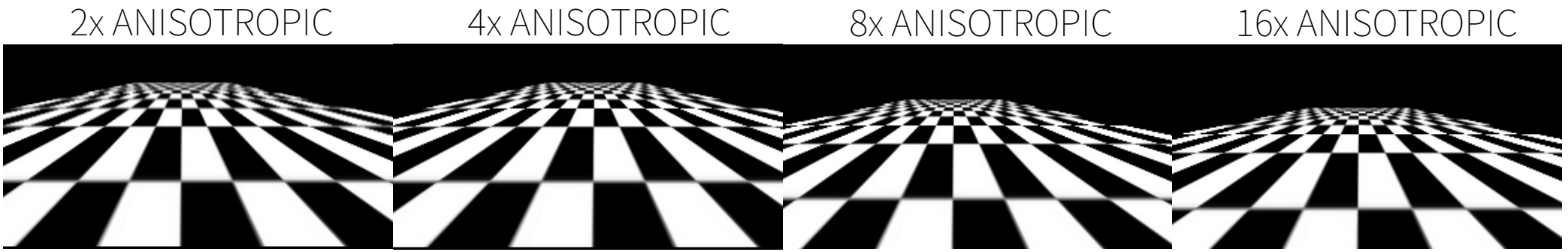
- 运用我们先前学到的三线性过滤公式，插值得

$$S_{n_0}(1 - \Delta n) + S_{n_1} \Delta n$$

- 接下来考虑如何求 Mipmap 层级 N。

各向异性过滤 Anisotropic Filtering

- 我们已经选定了 P_{max} 方向，那么顺理成章地，我们应该以 P_{min} 的长度为基准选定 mipmap 层级。
- 但是当 $\frac{\|P_{max}\|}{\|P_{min}\|} > \lambda_{max}$ 时，再选择 P_{min} 的长度为基准，会导致 mipmap 层级的估计偏小。
- 所以我们选择 $\frac{\|P_{max}\|}{\lambda}$ 作为采样窗口宽度计算。



实验

- 请自行阅读《OpenGL 着色语言》一书的前 5 章
- 使用 OpenGL 着色语言 (GLSL) 实现本 PPT 中介绍的各种纹理过滤算法：
  - 点采样 + 双线性： 30%
  - 使用 Mipmap 的双线性 + 三线性： 40%
  - 各向异性： 30%
- 下载 cg2017_lab01_question 工程
- 完成创建所需着色器程序的函数

实验

```
3 // Implementation for the lab's shader program.
4 GLuint createLabProgram(int argc, char** argv,
5         GLint& texCoordIn, GLint& texSlot, GLint& texMode) {
6
7         // Add your code here!
8         return 0;
```

- 前两个参数是程序运行参数，可以不用管
- 返回值为 glCreateProgram 返回的，并且已经装载完成的着色器程序句柄。
- texCoordIn 代表纹理坐标，在着色器中的类型应该为 attribute vec2 。
- texSlot 代表纹理 id ，在着色器中的类型应该为 uniform sampler2D 。
- texMode 代表选择的过滤方式，从 1 到 8 分别为：最近点采样，双线性，使用 Mipmap 的双线性，三线性， 2x 各向异性， 4x 各向异性， 8x 各向异性， 16x 各向异性。在着色器中类型应该为 uniform int 。

实验

- 只修改并提交 glshader.cpp
- 用 java 的同学需提交整个工程，附加说明，并保证内容和原 C++ 工程一致。
- 不要改变纹理的模式，不要打开硬件各向异性过滤
- 根据实验要求，在和纹理访问相关的部分，不能使用 GLSL 自带的 tex 开头的任何内置函数），以下函数除外
- texelFetch： 相当于前文所述的 $\boldsymbol{\tau}(u, v)$
- textureSize： 相当于前文所述的 $\|\boldsymbol{\tau}\|$
- 可以使用自己实现的纹理过滤函数
- 要有注释和文档！

思考问题

- 前文所述的哪些过滤可以用于放大过滤？哪些可以用于缩小过滤？
- 为一个灰度图生成一整套 MIPMAP 后，占用空间为原灰度图的多少倍？为一个 RGB 图像呢？
- 在 MIPMAP 层级计算中， N 的范围是什么？当 N 小于 0 时意味着什么？
- 计算得出各向异性度等于 1 时，应如何处理？
- 比较各种过滤算法的画面效果，所需的时间复杂度（采样点个数）和空间复杂度。