Shader优化技巧

# Overdraw

## PreDepth Pass

适合vertex少，FragShader消耗大，pixel重叠严重的物体使用。

比如草就不适用，因为草的vertex数量多，VertexShader还有位移之类的额外计算。

# 减少计算量

## 使用合适的精度

比如color使用float4改成half4，可以将计算量减半。

## FragShader的计算改到VertexShader

通常要计算的pixel数量要比vertex多的多，可以将FragShader计算相同的值移动到VertexShader，以及一些低端机的表现效果，比如顶点额外灯。

## 使用贴图采样代替计算

比如噪声图代替程序化噪声，matcap代替PBR。

### 使用近似函数替代超越函数

pow(x,2)用x\*x替代，不用过多优化，一定要点开编译后的shader检查。

### Shader的计算改到C#

比如使用关于时间的sin时，可以直接用unity内置的\_SinTime。

### 指令优化

<https://www.bilibili.com/video/BV1V841147Wd/?spm_id_from=333.999.0.0>

# 纹理优化

## 打开mipmap

GPU会 load 临近的像素数据到缓存，开启mipmap可以减少cache miss。

## 使用合适的纹理格式与大小

越小的图消耗的带宽越少，所以尽量使用mipmap大小的图片，或者开启steaming mipmaps（注意这会消耗一部分cpu性能）。

删除掉你没有使用的压缩通道比如Alpha。

还可以通过纹理压缩格式来减低纹理的大小，比如ASTC。

## 减少采样次数

纹理采样操作是昂贵的，即使是同一张纹理的不同uv位置，尽量保证一个shader的采样次数低于计算单元的消耗。

## 合并纹理

当采样纹理uv相同时，可以合并纹理的通道，比如把metallic、smoothness、occlusion合并在一张图上，这样采样就从3次变成了1次。

## 纹理采样改计算

需要大量纹理采样的功能，可以考虑job system或者computer shader处理。

# 关于if

## 使用keyword控制分支

能不用 if 就不用，可以通过keyword控制条件语句，比如自发光的开启。

## 使用branch

通常 if 的两边代码会一起运行，当if内的计算量很复杂时，就有必要使用[branch]了，这样只会运行一边的代码。

# 关于Alpha Test

移动端开启AlphaTest会到导致Early-Z失效，从而让硬件自带的HSR之类的功能失效，产生大量overdraw，所以尽量不使用AlphaTest。如必需使用，比如树和头发，可以通过在PreDepth Pass里开启AlphaTest，来优化性能。

# 其他

1. 使用GPU Instancing时，不需要在C#设置MaterialPropertyBlock的属性，shader里一定要写block的外面，这会节省大量的内存。
2. 共用纹理的Sample,比如地形Shader的4张颜色图共用一个Sample,这个优化在OpenGLES无效，但在Vulkan和Metal有效。
3. 删除掉你没有使用的Attrbutes和Varyings里的值，比如顶点色，这将节省内存和带宽的消耗，以及插值到FragShader的时间。

# shader离线性能分析

使用Arm Mobile Studio的mali\_offline\_compiler可以对shader进行离线分析。

下面是Bifrost架构的FragShader性能分析，其中有4种硬件单元（A,LS,V,T），它们是互相独立的，所有我们应该保证它们的性能消耗是平均，这有助于管线流水线的执行。

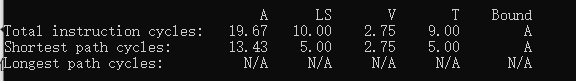
A（Arithmetic）：是数学运算单元

LS（Load/store）：是非图片的加载单元（比如material上的某个属性）

V（Varying）：是从VertexShader插值到FragShader的值（其实就是Varyings结构体）

T（Texture）：图片采样次数

很明显下面的Bound是A，所以我们应该优化FragShader的数学运算。



# ShaderLOD

为不同的设备设计不同的shader，一般三个品质就够了，过多的shader会导致内存增多。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A | LS | V | T |
| 高 | 20 | 15 | 8 | 10 |
| 中 | 12 | 10 | 6 | 6 |
| 低 | 8 | 6 | 4 | 4 |