Unity移动游戏性能分析与优化

Overview

- About Unity
- 性能分析工具
- 引擎优化
- 使用优化

Unity的历史

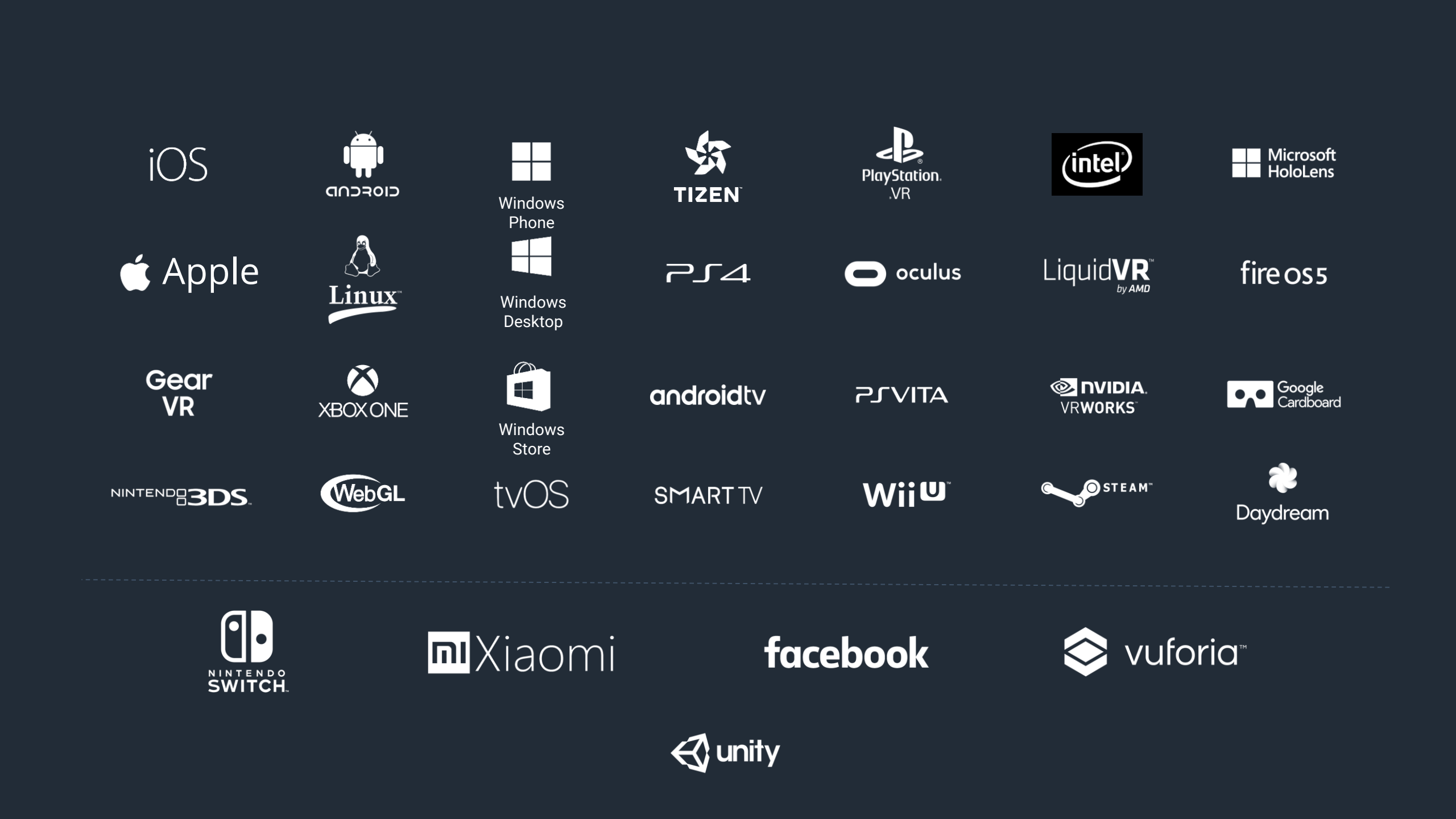
- 2005年成立于丹麦，哥本哈根
- David Helgason, Nicholas Francis 和 Joachim Ante

实现开发
大众化

解决疑难
问题

助力开发者
成功

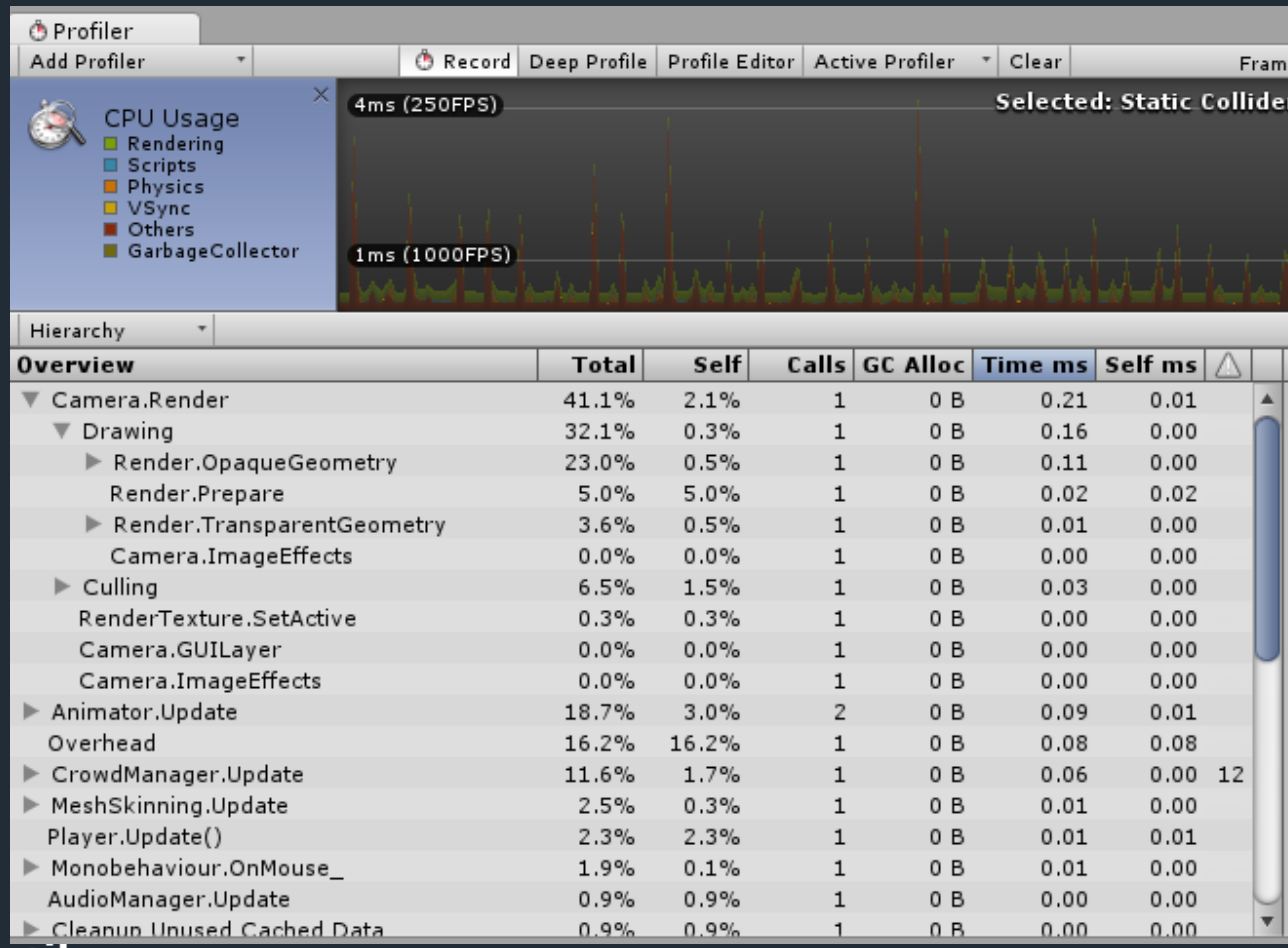
unity


iOS
ANDROID
Windows Phone
TIZEN
PlayStation VR
intel
Microsoft HoloLens
Apple
Linux
Windows Desktop
PS4
oculus
LiquidVR by AMD
fireos5
Gear VR
XBOX ONE
Windows Store
androidtv
PS VITA
NVIDIA VRWORKS
Google Cardboard
NINTENDO 3DS
WebGL
tvOS
SMART TV
Wii U
STEAM
Daydream
NINTENDO SWITCH
Xiaomi
facebook
vuforia
unity

性能分析工具

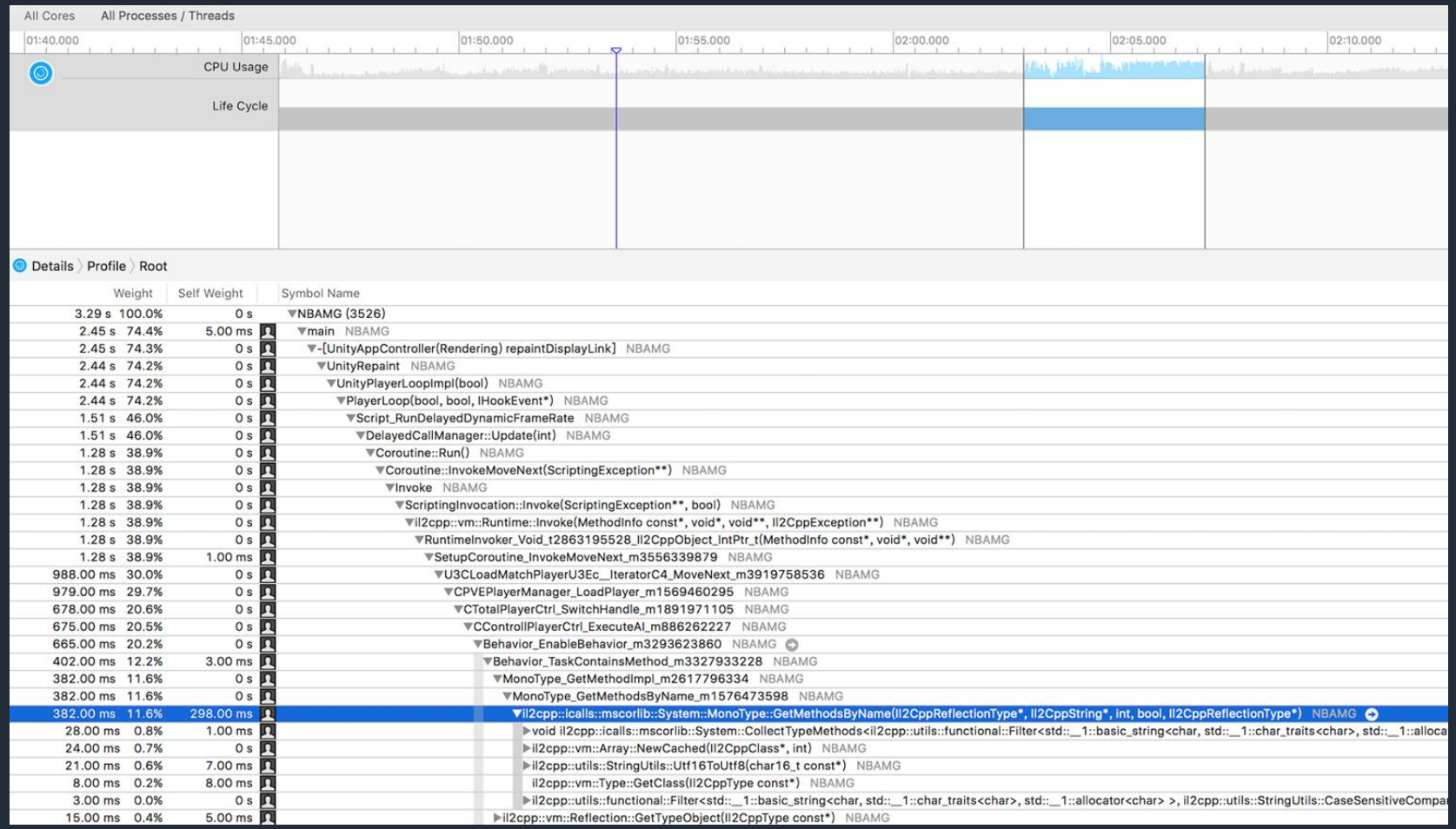
Unity Profiler

- 连接手机
- 观察在CPU Usage中，每帧耗时超过%1的函数
- 使用Profiler.BeginSample/EndSample自定义标签，来检测代码段
- 使用Deep profile
- 观察GC Alloc，消除GC的诱因。



unity

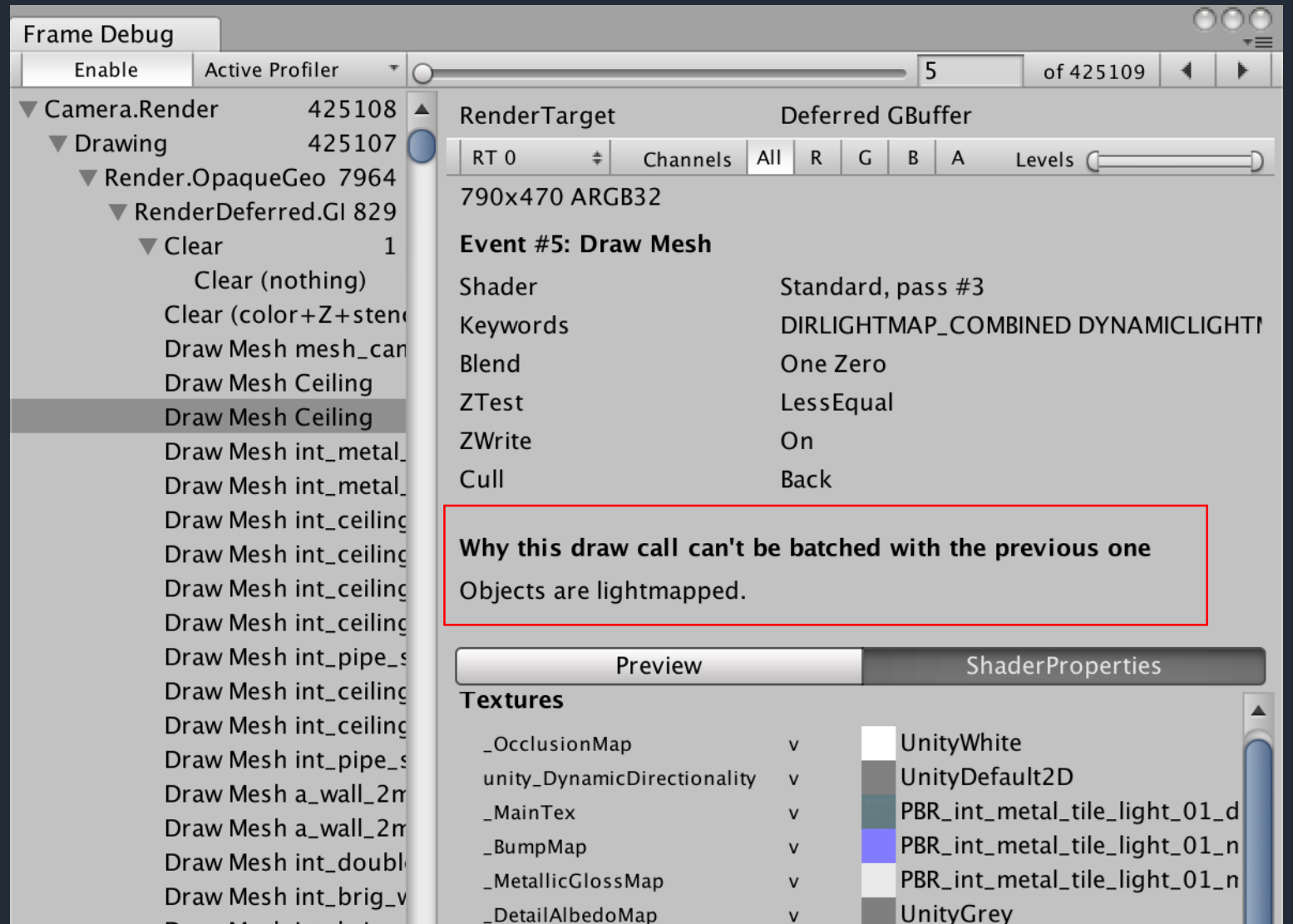
Xcode Time Profiler



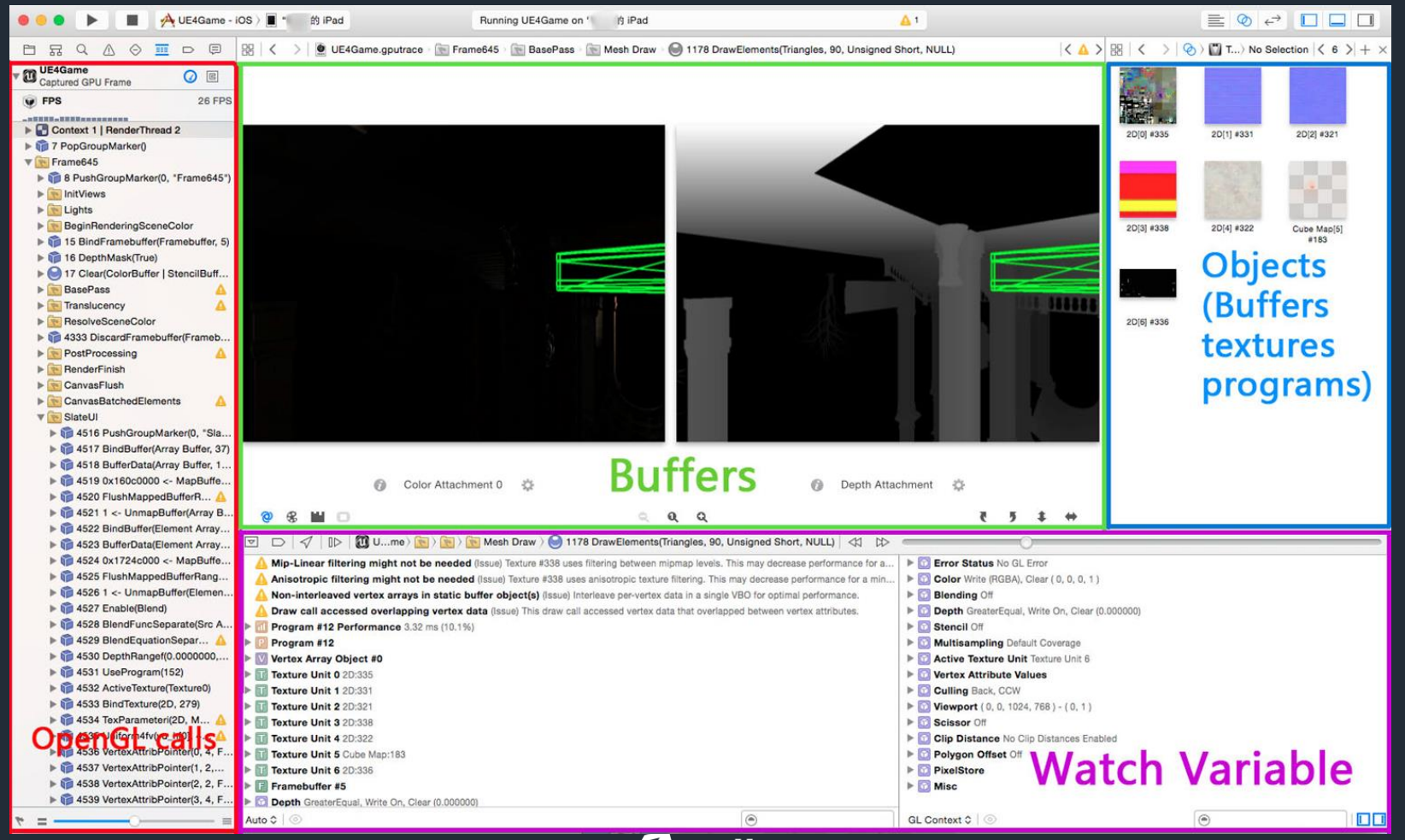
unity

Unity Frame Debugger

- Draw Call
- 查看渲染次序
- 查看合并批次是否符合预期
- 检查单个渲染参数



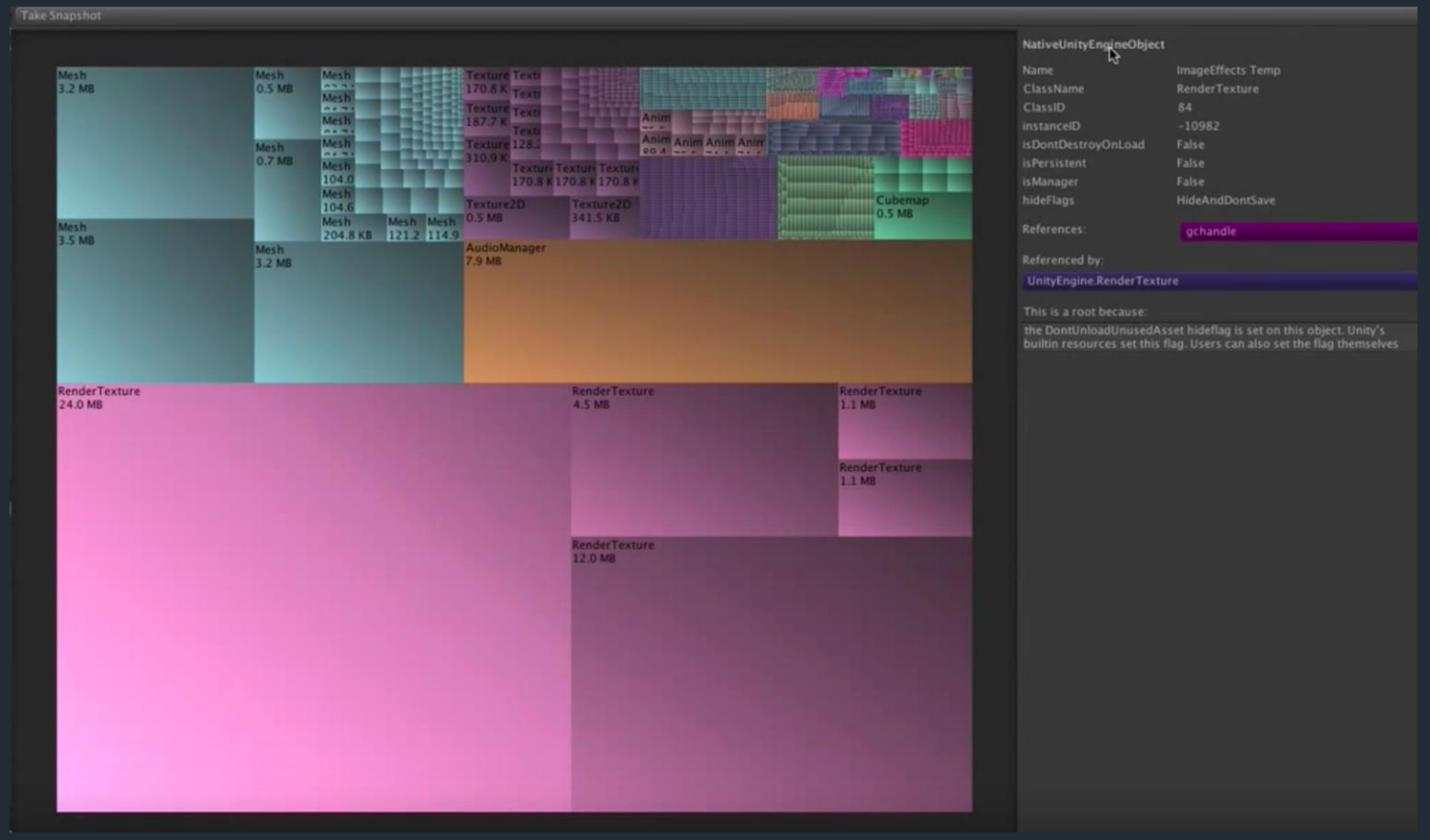
Xcode Capture GPU Frames



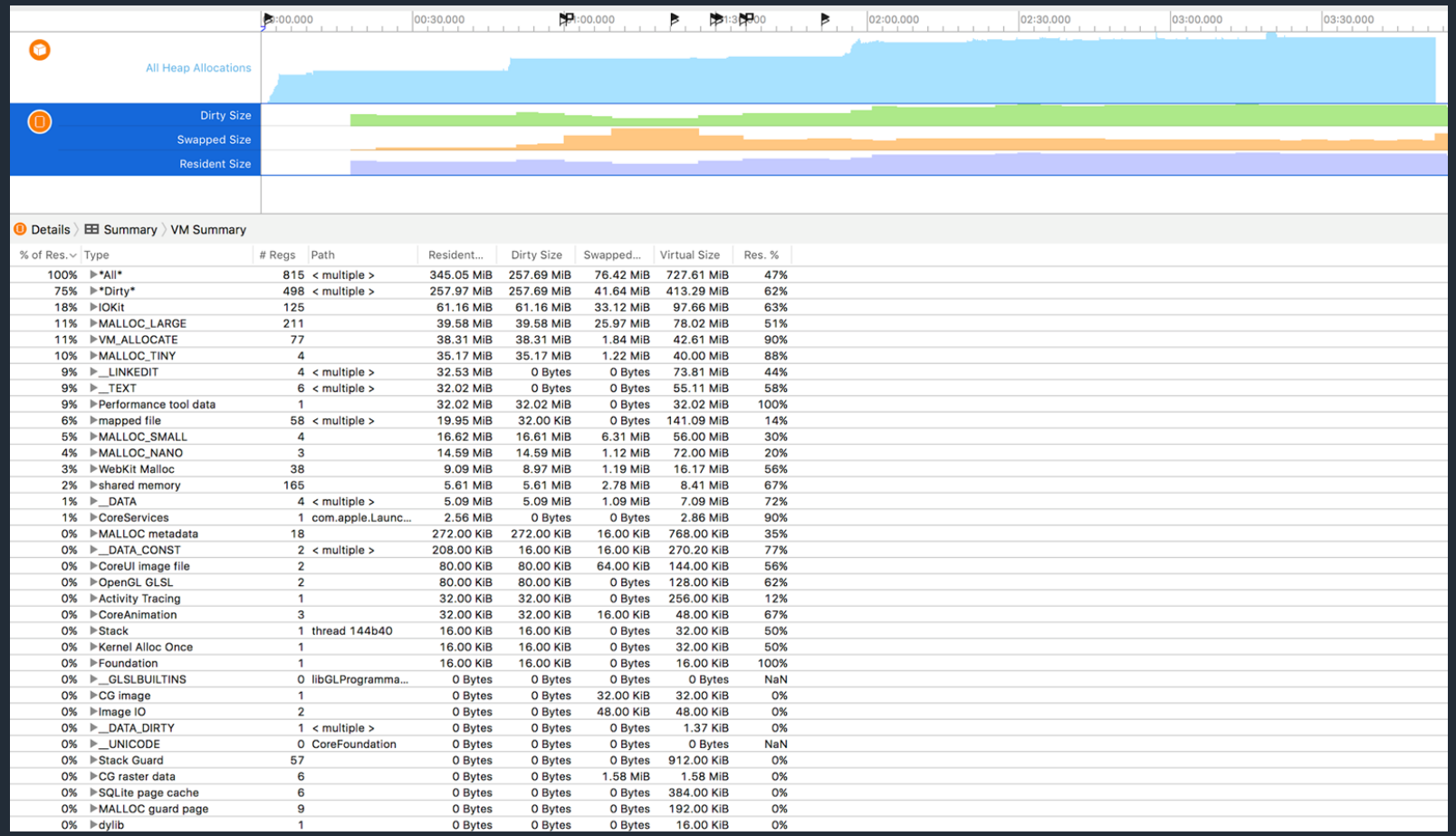
unity

Unity Memory Profiler

- https://bitbucket.org/Unity-Technologies/memoryprofiler



Xcode Instruments Allocations



% of Res.	Type	# Regs	Path	Resident...	Dirty Size	Swapped...	Virtual Size	Res. %
100%	▶*All*	815	< multiple >	345.05 MiB	257.69 MiB	76.42 MiB	727.61 MiB	47%
75%	▶*Dirty*	498	< multiple >	257.97 MiB	257.69 MiB	41.64 MiB	413.29 MiB	62%
18%	▶IOKit	125		61.16 MiB	61.16 MiB	33.12 MiB	97.66 MiB	63%
11%	▶MALLOC_LARGE	211		39.58 MiB	39.58 MiB	25.97 MiB	78.02 MiB	51%
11%	▶VM_ALLOCATE	77		38.31 MiB	38.31 MiB	1.84 MiB	42.61 MiB	90%
10%	▶MALLOC_TINY	4		35.17 MiB	35.17 MiB	1.22 MiB	40.00 MiB	88%
9%	▶__LINKEDIT	4	< multiple >	32.53 MiB	0 Bytes	0 Bytes	73.81 MiB	44%
9%	▶__TEXT	6	< multiple >	32.02 MiB	0 Bytes	0 Bytes	55.11 MiB	58%
9%	▶Performance tool data	1		32.02 MiB	32.02 MiB	0 Bytes	32.02 MiB	100%
6%	▶mapped file	58	< multiple >	19.95 MiB	32.00 KiB	0 Bytes	141.09 MiB	14%
5%	▶MALLOC_SMALL	4		16.62 MiB	16.61 MiB	6.31 MiB	56.00 MiB	30%
4%	▶MALLOC_NANO	3		14.59 MiB	14.59 MiB	1.12 MiB	72.00 MiB	20%
3%	▶WebKit Malloc	38		9.09 MiB	8.97 MiB	1.19 MiB	16.17 MiB	56%
2%	▶shared memory	165		5.61 MiB	5.61 MiB	2.78 MiB	8.41 MiB	67%
1%	▶__DATA	4	< multiple >	5.09 MiB	5.09 MiB	1.09 MiB	7.09 MiB	72%
1%	▶CoreServices	1	com.apple.Launc...	2.56 MiB	0 Bytes	0 Bytes	2.86 MiB	90%
0%	▶MALLOC metadata	18		272.00 KiB	272.00 KiB	16.00 KiB	768.00 KiB	35%
0%	▶__DATA_CONST	2	< multiple >	208.00 KiB	16.00 KiB	16.00 KiB	270.20 KiB	77%
0%	▶CoreUI image file	2		80.00 KiB	80.00 KiB	64.00 KiB	144.00 KiB	56%
0%	▶OpenGL GLSL	2		80.00 KiB	80.00 KiB	0 Bytes	128.00 KiB	62%
0%	▶Activity Tracing	1		32.00 KiB	32.00 KiB	0 Bytes	256.00 KiB	12%
0%	▶CoreAnimation	3		32.00 KiB	32.00 KiB	16.00 KiB	48.00 KiB	67%
0%	▶Stack	1	thread 144b40	16.00 KiB	16.00 KiB	0 Bytes	32.00 KiB	50%
0%	▶Kernel Alloc Once	1		16.00 KiB	16.00 KiB	0 Bytes	32.00 KiB	50%
0%	▶Foundation	1		16.00 KiB	16.00 KiB	0 Bytes	16.00 KiB	100%
0%	▶__GLSLBUILTINS	0	libGLProgramma...	0 Bytes	0 Bytes	0 Bytes	0 Bytes	NaN
0%	▶CG image	1		0 Bytes	0 Bytes	32.00 KiB	32.00 KiB	0%
0%	▶Image IO	2		0 Bytes	0 Bytes	48.00 KiB	48.00 KiB	0%
0%	▶__DATA_DIRTY	1	< multiple >	0 Bytes	0 Bytes	0 Bytes	1.37 KiB	0%
0%	▶__UNICODE	0	CoreFoundation	0 Bytes	0 Bytes	0 Bytes	0 Bytes	NaN
0%	▶Stack Guard	57		0 Bytes	0 Bytes	0 Bytes	912.00 KiB	0%
0%	▶CG raster data	6		0 Bytes	0 Bytes	1.58 MiB	1.58 MiB	0%
0%	▶SQLite page cache	6		0 Bytes	0 Bytes	0 Bytes	384.00 KiB	0%
0%	▶MALLOC guard page	9		0 Bytes	0 Bytes	0 Bytes	192.00 KiB	0%
0%	▶dylib	1		0 Bytes	0 Bytes	0 Bytes	16.00 KiB	0%

unity

Android CPU Profile

SysTrace
Build it on iOS and use Instruments anyway

Android GPU Tools

Check the chipset
Intel:Intel GPA
Qualcomm: Snapdragon Profiler
NVidia/Tegra: NVidia NSightAMD

CPU Bound Or GPU Bound?

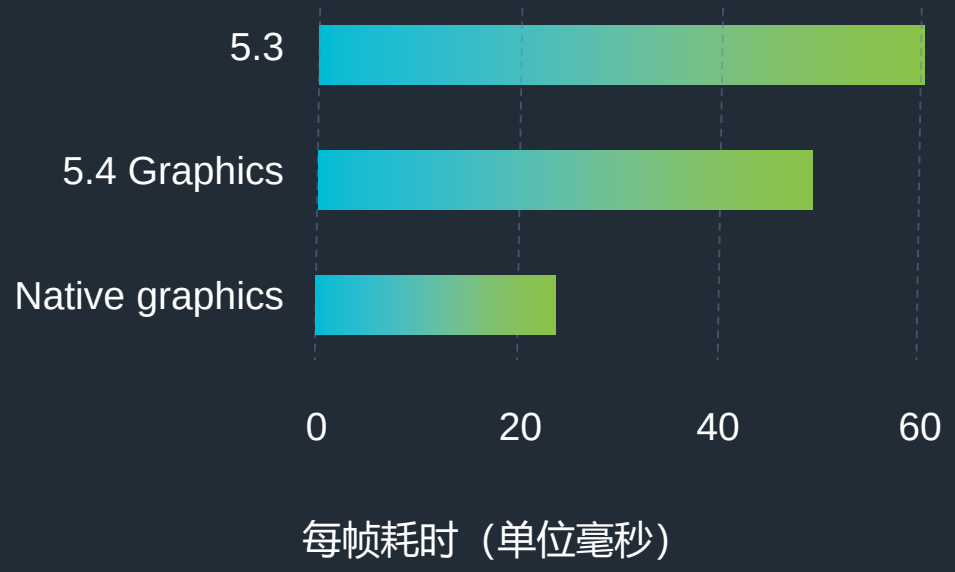
引擎优化

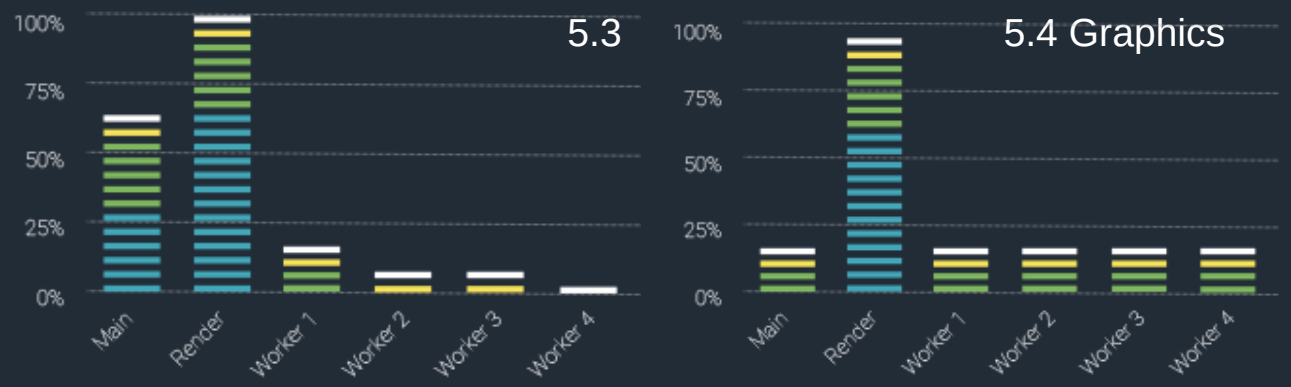
多线程渲染

- 众多的Unity游戏性能负荷集中在主线程
- 在Unity 5.4 中把大量的Camera.Render 的工作从主线程当中移除，但是这还远远不够

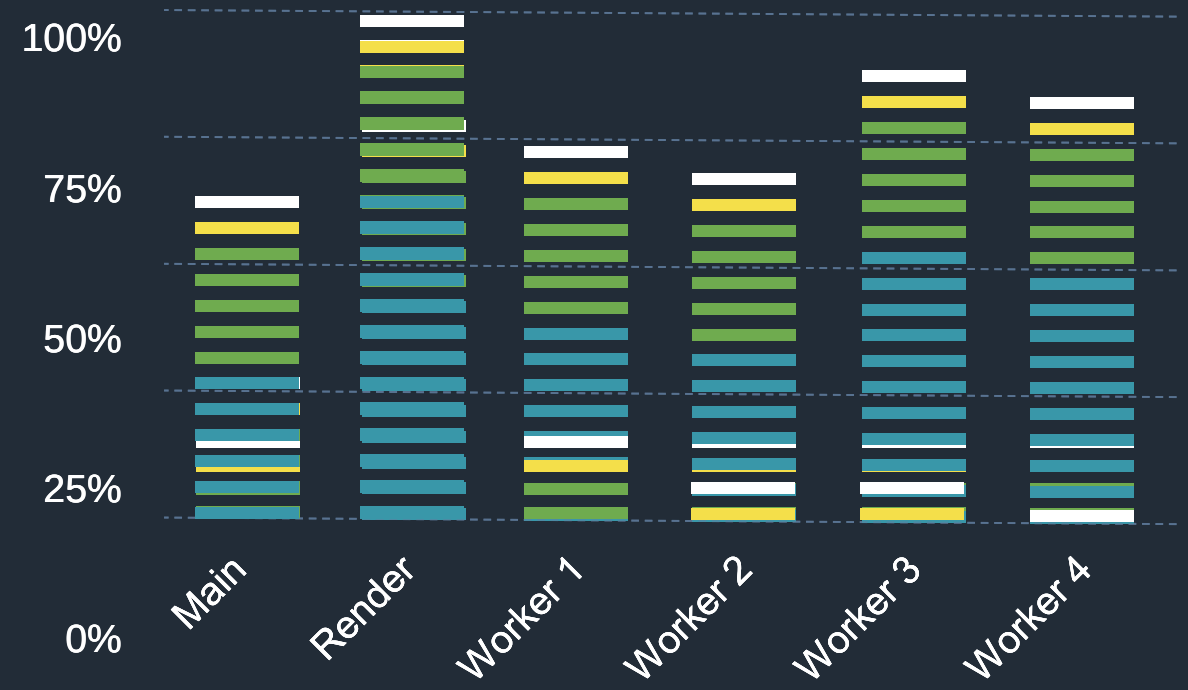
性能提升

(使用PS4作为测试机)





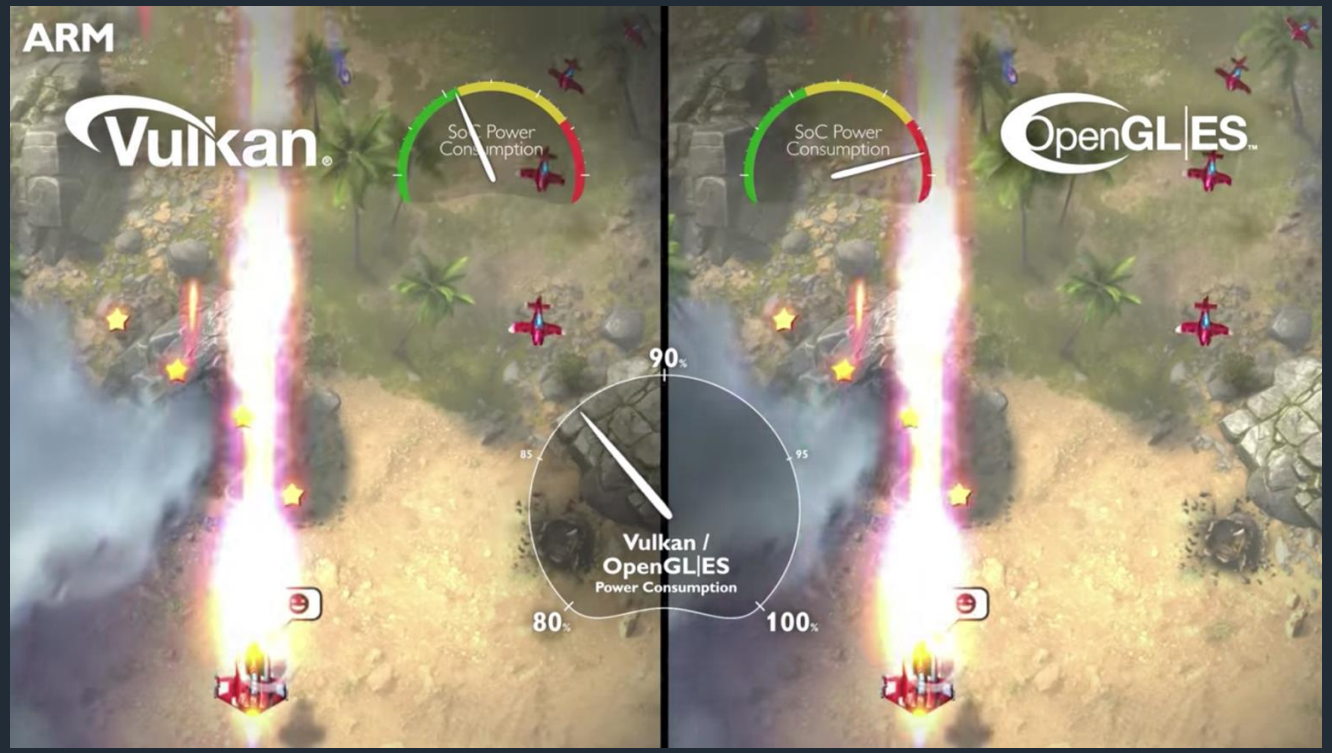
CPU 内核利用率



unity

多线程渲染

- 从Unity5.6开始支持Vulkan
- Vulkan就是被设计成能够让应用程序充分利用多核性能，多线程的并行构建渲染命令的图形API。

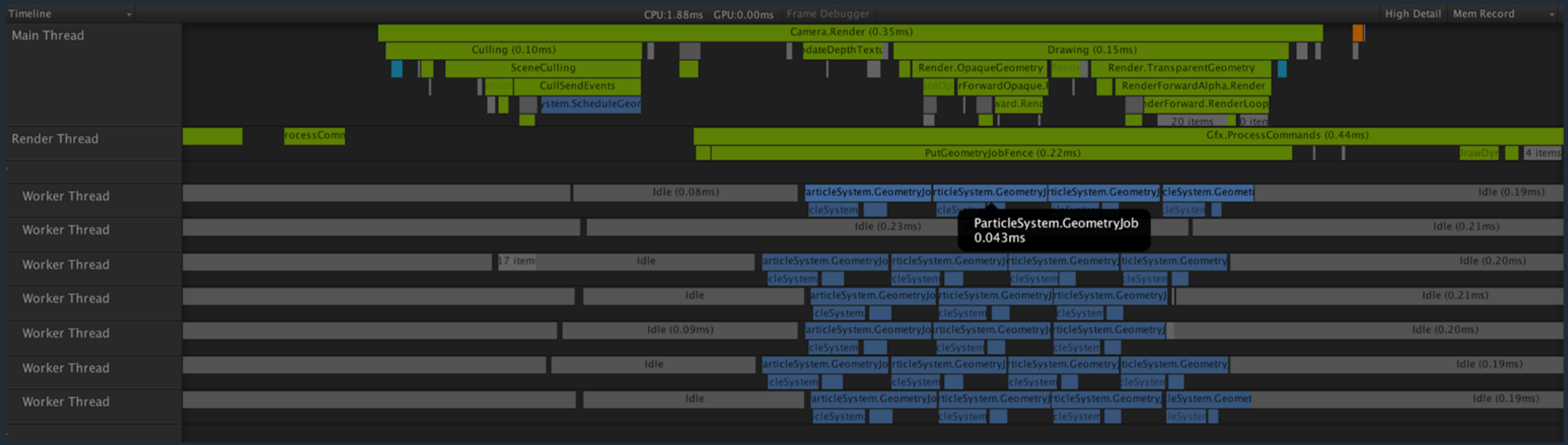


Unity Job System(Worker Threads)

- Unity会致力于把一些计算量比较重的系统攞到其他的工作线程上去。
- 例如我们的粒子系统，动画系统，布料计算，遮挡剔除，视锥体剔除，蒙皮和静态裁剪等等

Unity Job System(Worker Threads)

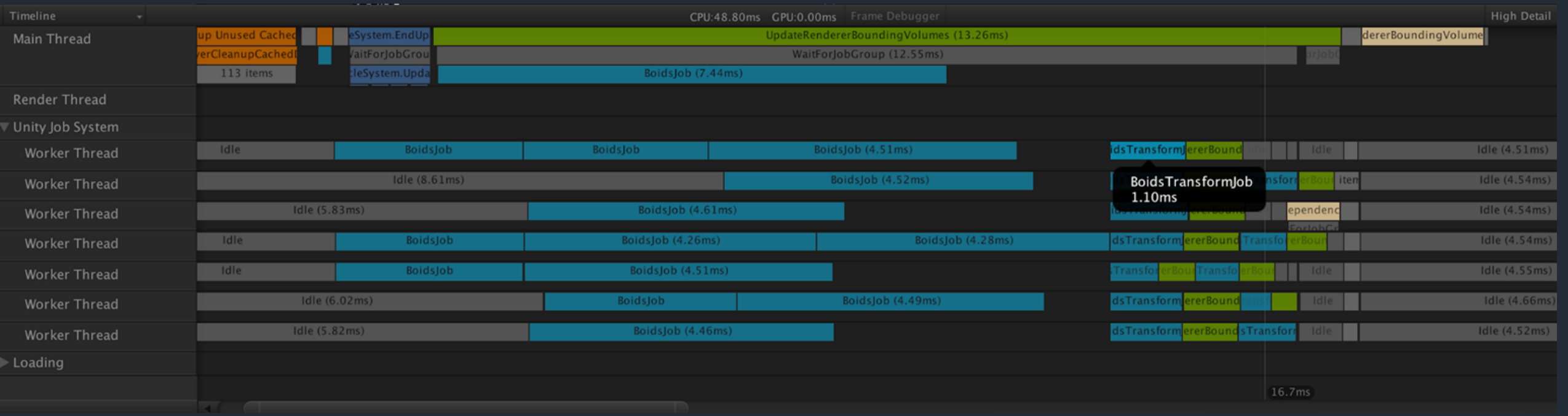
- ParticleSystem



unity

C# Job System

- 一个能够让你充分利用多核特性并且无需繁重的多线程编程的全新高性能多线程系统



unity

GPU Instancing

- GPU Instancing is available on the following platforms and APIs:
- **DirectX 11** and **DirectX 12** on Windows
- **OpenGL Core 4.1+/ES3.0+** on Windows, macOS, Linux, iOS and Android
- **Metal** on macOS and iOS
- **Vulkan** on Windows and Android
- **PlayStation 4** and **Xbox One**
- **WebGL** (requires WebGL 2.0 API)

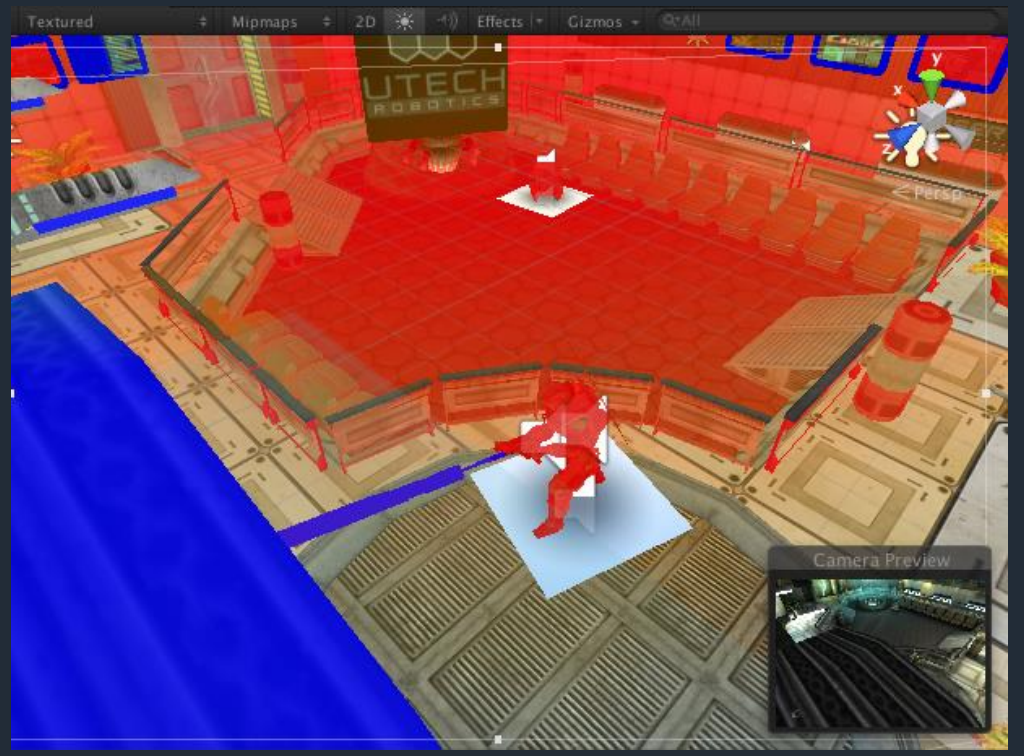
使用优化

资源之通用

- 良好的项目目录结构
- 使用脚本来处理导入的资源： AssetPostprocessor
- 使用Asset Bundle，LoadFromFileAsync, LZ4
- 减少Resources文件夹里的资源

资源之场景

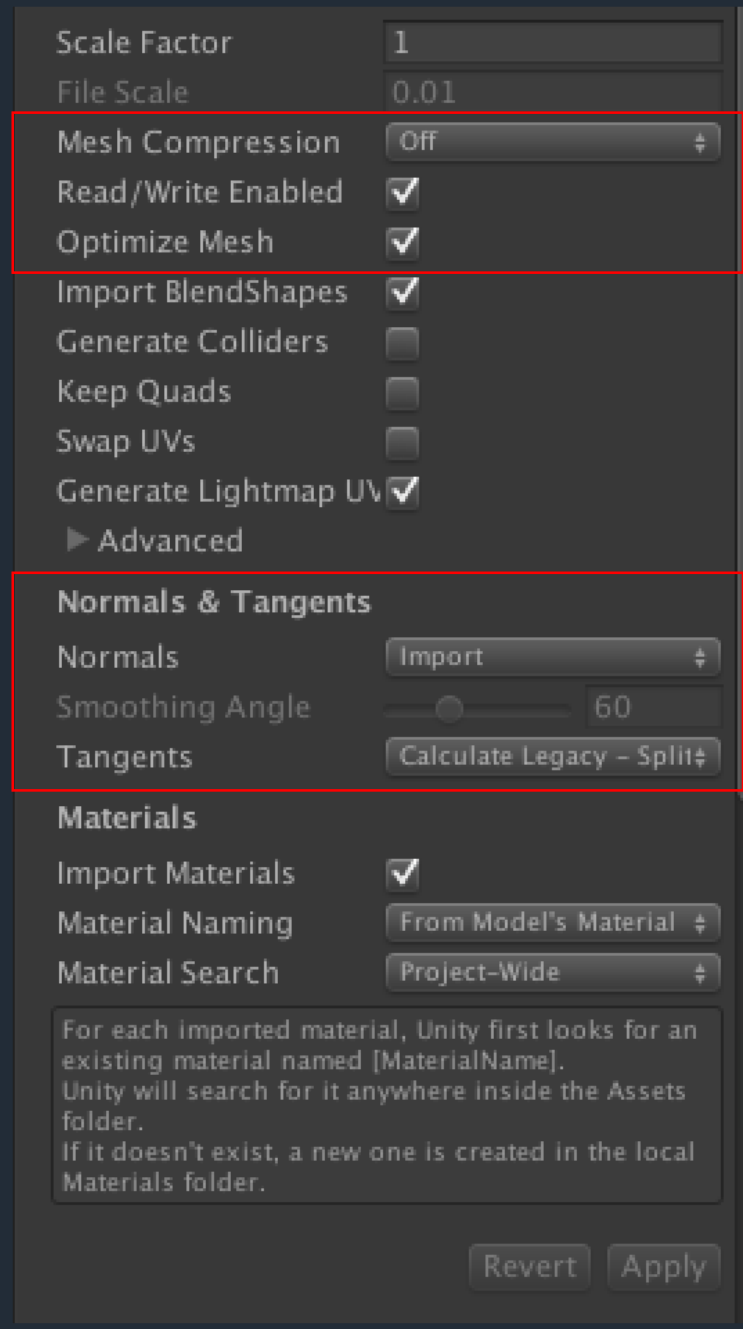
- 制定场景标准
- 使用Static Batching
- 使用PNG，TGA图片格式，建议开启Mipmaps
- 在Scene View中的Mipmaps模式下，检查适合的纹理尺寸
- 控制纹理数量，纹理拼合的合理性



unity

资源之模型

- 控制模型面数，检查Mesh数据
- 开启模型压缩
- Read/Write Enabled，设为关闭
- Optimize Mesh 必须选择
- Normals & Tangents 设置为非Import选项以降低memory以及减少ipa/apk包体大小



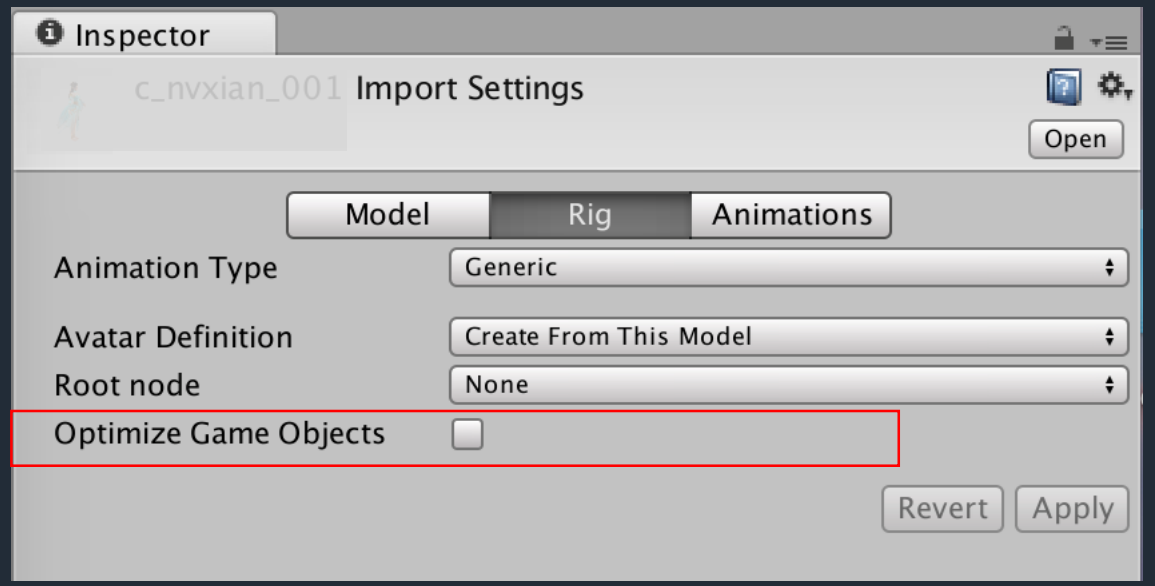
unity

模型的纹理

- 关闭Read/Write Enable以节省内存。
- 压缩纹理
- 组合纹理
- 根据苹果规范PVRTC要使用正方形纹理
- ETC2在OpenGL ES 2.0上不支持的平台上会被转为RGBA

资源之动画

- 开启Optimize Game Objects，多线程Skinning
- 有两件事是在主线程处理的
  - 脚本的调用 State Machine Behaviors, Animation Events
  - 更新动画角色的Transforms



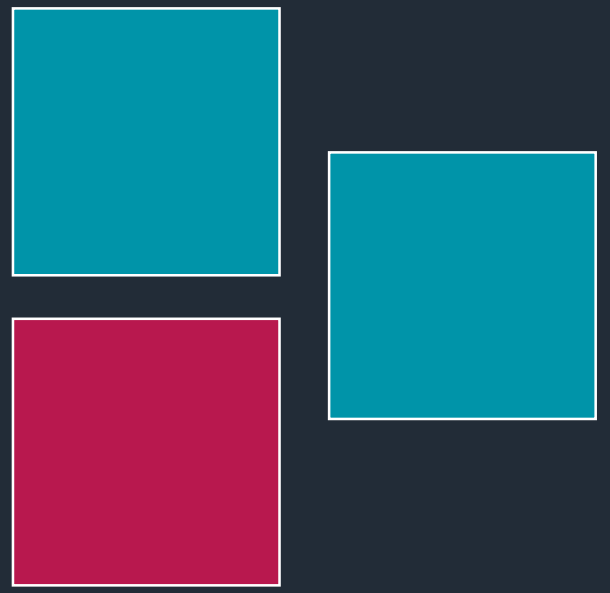
unity

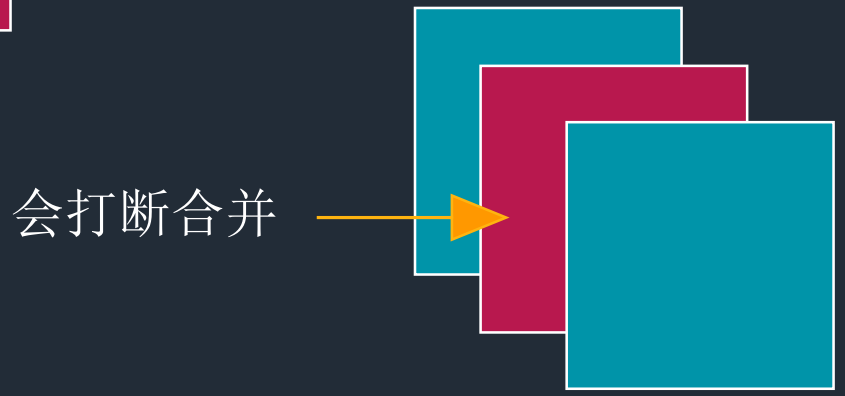
资源之声音

- 推荐iOS用MP3，Android用Vorbis
- 背景音乐要用流
- 推荐开启Force To Mono

资源之UI

- 图集要尽量填满
- 图片的Alpha可以拆出来
- 菜单拆图时，合理安排图片元素的复用
- 独立的图片要合理
- 菜单层次结构要清晰合理，动静分离





unity

引擎之相机

- 适合的远近裁剪面
- 每个相机各司其职，分层，调整好绘制顺序

引擎之光照

- 实时光很耗时
- 场景推荐使用Lightmap
- 推荐用Light Probe，Reflection Probe

引擎之粒子

- 粒子系统的使用要适量
- 同屏幕的最大粒子数不超过400
  - 每个粒子系统的Max Particles之和
- 单个粒子的范围变小，粒子系统动态开关。

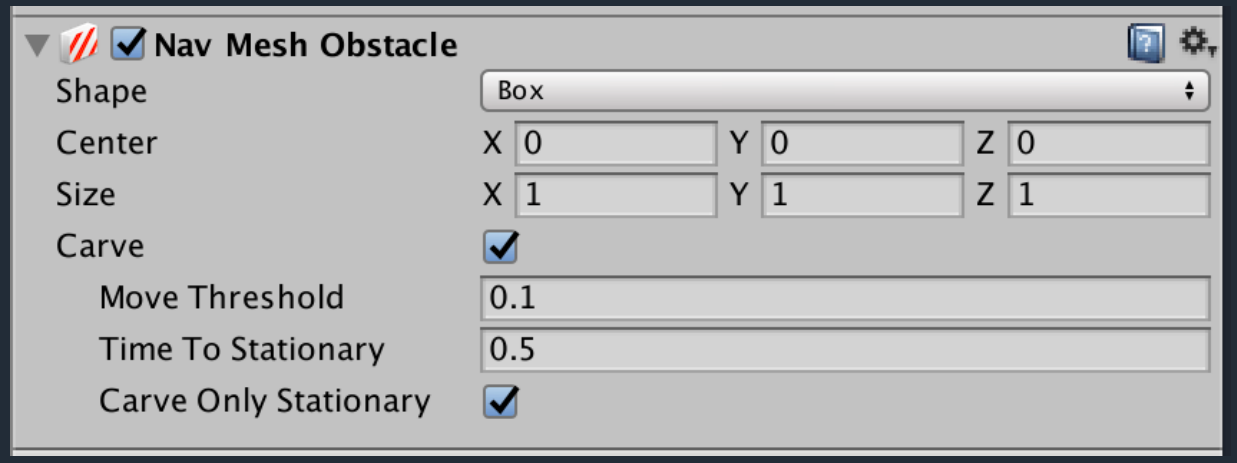
引擎之物理

- 尽量使用Box Collider、Sphere Collider
- Mesh Collider性能消耗较高， Cloth更高
- 在Layer Collision Matrix中，只保留必要的选项。
- 提升Fixed Timestep的值，降低物理更新频率（0.033，即每秒30帧）
- 开启RigidBody，勾选 isKinematic，使用Tirgger的检测
- 射线检测要带着距离

引擎之导航网格

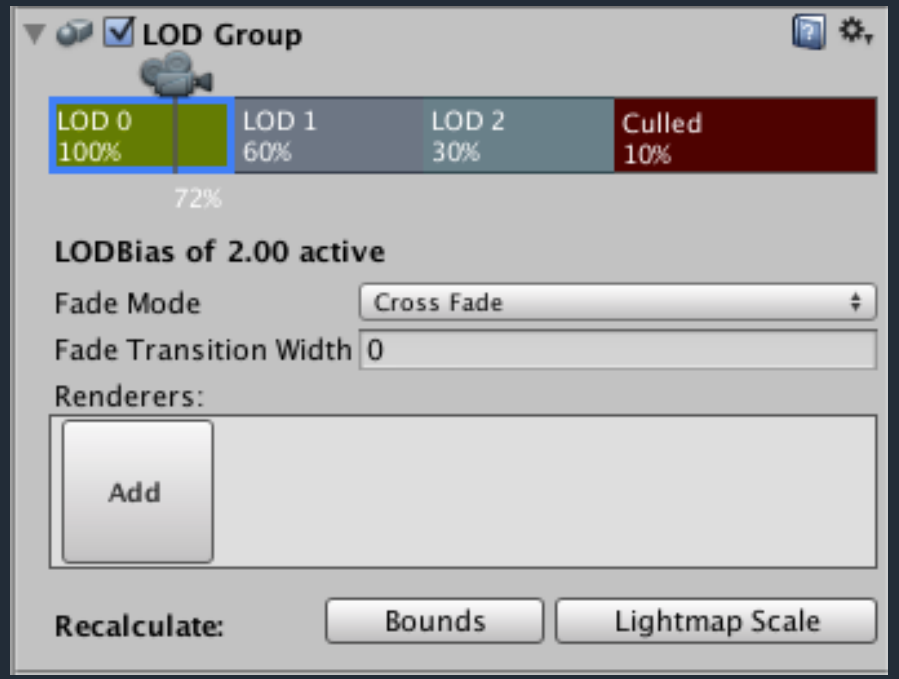
- 合理分配layer
- 控制有效区域
- NavMeshObbstacle的Carve选项

  会触发导航网格重新计算。



引擎之渲染

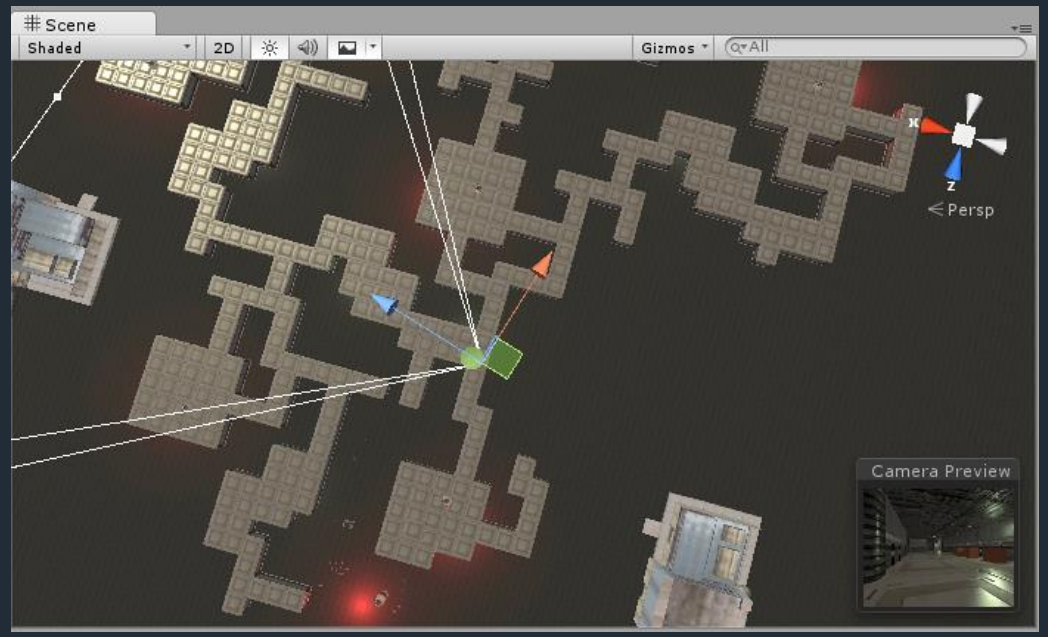
- 尽可能地避免Alpha Test
- 警惕Alpha Blend
  - 在Scene View中的Overdraw模式下
- 使用LOD Group

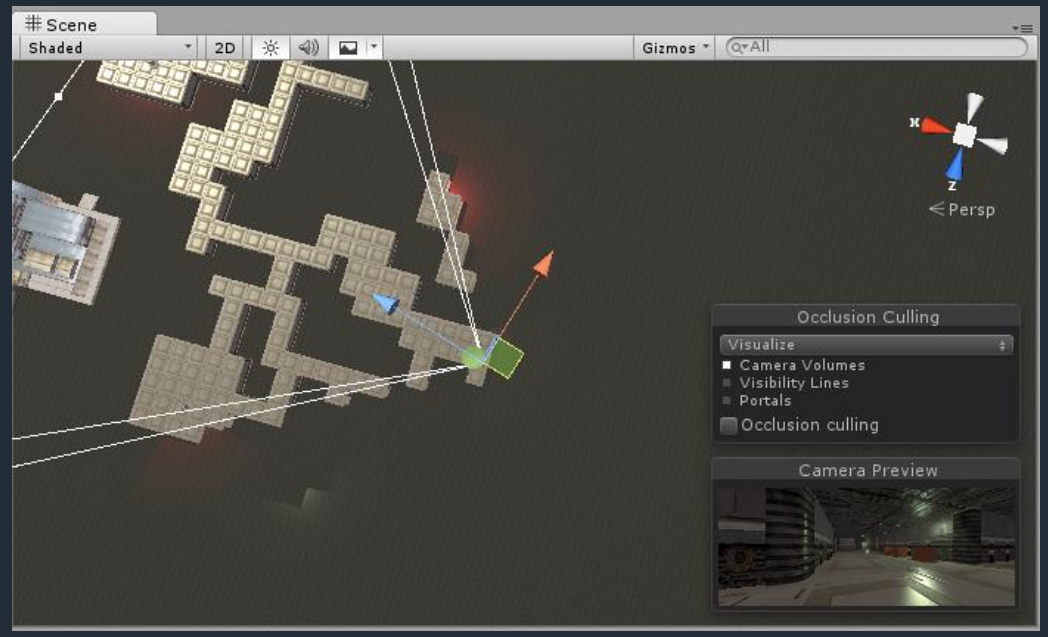


unity

引擎之渲染

- 使用遮挡剔除OcclusionCulling





unity

引擎之渲染

- Static Batching
  - StaticBatchingUtility.Combine() (需要开启Read/Write Enable)
- Dynamic Batching
  - 缓存900个顶点属性，相同的模型和材质（阴影例外）
  - ✕Transform中Scale为-1的不能缓存
  - ✕ Lightmap有额外属性的，lightmap index和offset/scale。
  - ✕多个Pass的shader

引擎之渲染

- 手动合并Mesh和Material，例如纸娃娃每次换装后的模型
- 手动合并一些UI图标到一个图集
- 边框类型的Sprite
  - 推荐使用Sliced（9板模式）
  - 中间不需要就不要勾选fill center false

C# 脚本优化

- 问题不会出现在只调用一次的代码里，优化每帧都运行的代码 Update()。
  - 评估这部分代码真的是每帧都要运行的吗？
  - 是否可以使用事件机制代替？
- 评估算法时间复杂度
- 使用对象池系统

C# 脚本优化

GC Alloc

- tostring() foreach()
- Mesh.verticles
- Input.touches
- SkinMeshRenderer.bones
- etc

Shader优化

- 评估shader代码的时间复杂度，优化或使用近似结果的代码
- 减少函数调用pow、sin等，使用查找纹理替换
- Discard效率低，if更不要用
- 考虑浮点数计算
  - float/highp – 全32位浮点型，适合顶点变换，效率最低
  - half/mediump – 缩减的16位浮点型，适合于纹理UV坐标
  - fixed/lowp – 10位定点型，适合于颜色，光照计算
- 使用Shader.WarmupAllShaders提前编译shader

Thank you!