



Optimizing Unity Games

dracula_jin@163.com

Optimizing Unity Games

1

渲染模块

2

UI模块

3

加载模块

4

内存管理

渲染模块

1

CPU性能瓶颈分析

2

GPU端性能瓶颈分析

CPU端性能瓶颈

CPU 处理内容

CPU自身逻辑处理的内容

CPU与GPU交互时处理的内容

CPU端性能瓶颈

1

不透明渲染

2

半透明渲染

3

Culling裁剪

半透明/不透明渲染

一般为场景，角色，UI，和粒子特效的渲染开销

不透明渲染

动态拼合渲染

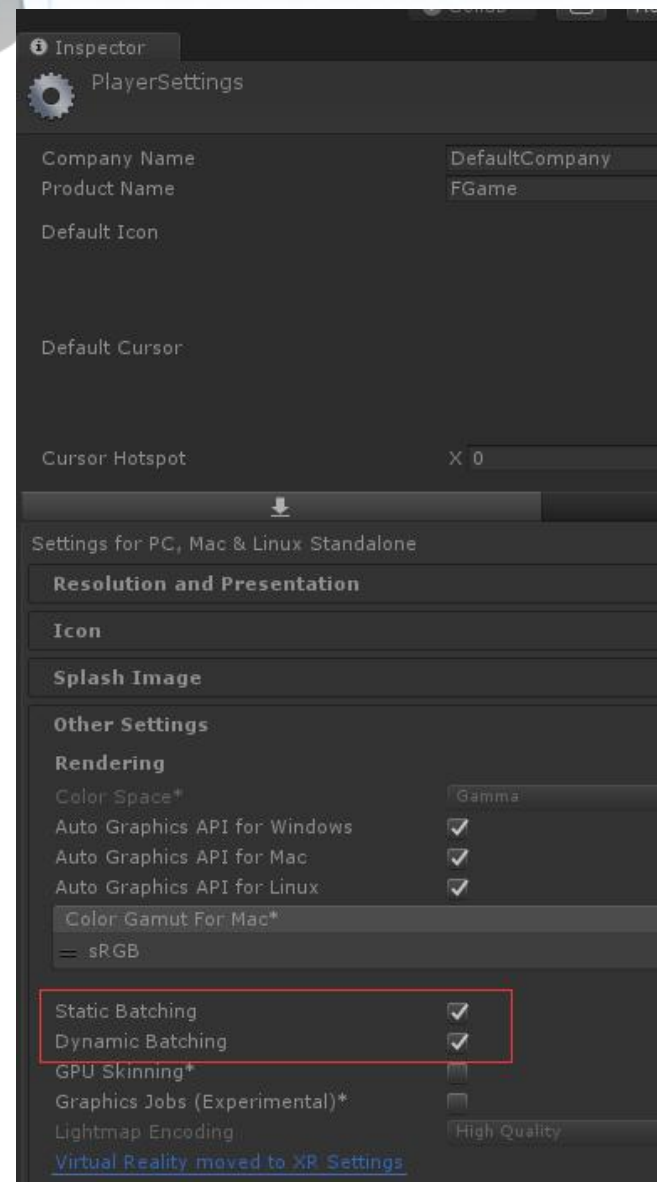
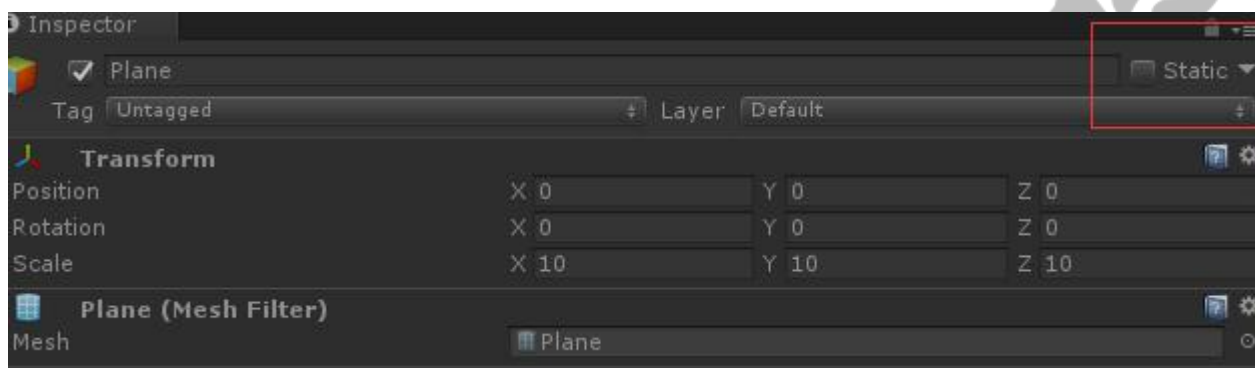
静态拼合渲染

不拼合渲染

GPUInstance

半透明/不透明渲染

一般为场景，角色，UI，和粒子特效的渲染开销



半透明/不透明渲染

一般为场景，角色，UI，和粒子特效的渲染开销

半透明渲染

UI渲染

粒子系统渲染

一些特定材质的mesh

半透明/不透明渲染

Triangle

峰值建议小于20W面

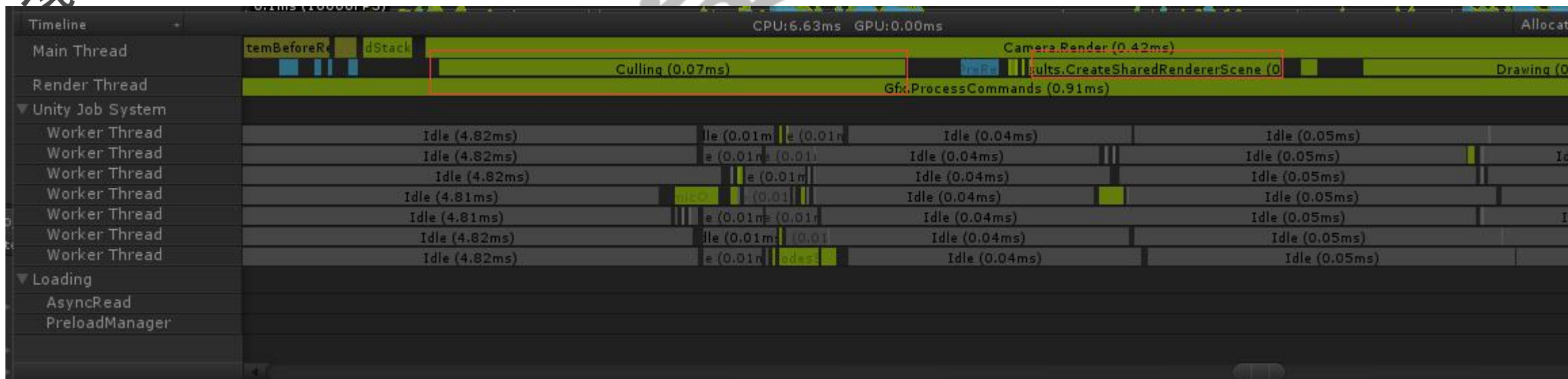
一般做法是mesh简化, LOD, Culling Distance etc

通过渲染密度和渲染屏占比来进行完善

半透明/不透明渲染

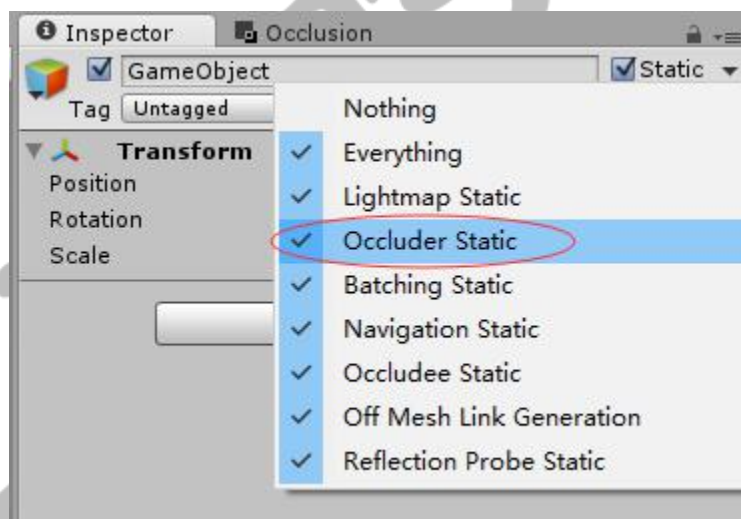
Culling

主要用于大地型或是MMO等有丰富场景元素类型的游戏



半透明/不透明渲染

Culling



CPU端性能瓶颈

引擎将所渲染的资源在CPU端准备好，再传到GPU端

Draw Call

RenderState

CPU端性能瓶颈

Draw Call SetPass Call Batches

```
Open Frame Debugger
SetPass Calls: 4      Draw Calls: 17      Total Batches: 17      Tris: 418  Verts: 1.1k
(Dynamic Batching)   Batched Draw Calls: 0      Batches: 0      Tris: 0    Verts: 0
(Static Batching)    Batched Draw Calls: 0      Batches: 0      Tris: 0    Verts: 0
(Instancing)         Batched Draw Calls: 0      Batches: 0      Tris: 0    Verts: 0
Used Textures: 13 - 3.0 MB
RenderTextures: 2 - 19.6 MB
RenderTexture Switches: 0
Screen: 1080x2160 - 26.7 MB
VRAM usage: 46.3 MB to 49.6 MB (of 0.96 GB)
VBO Total: 25 - 302.0 KB
VB Uploads: 0 - 0 B
IB Uploads: 0 - 0 B
Shadow Casters: 0
```

```
Enable  Connected Play
▼ Camera.Render 17
  ▼ Drawing 17
    ▼ Render.OpaqueGeometry 1
      ▼ RenderForwardOpaque.Render 1
        ▼ Clear 1
          Clear (Z+stencil)
        ▼ Render.TransparentGeometry 15
          ▼ RenderForwardAlpha.Render 15
            ▼ RenderForward.RenderLoopJob 15
              ▼ Canvas.RenderSubBatch 15
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
                Draw Mesh
              ▼ Render.OpaqueGeometry 1
                ▼ RenderForwardOpaque.Render 1
                  ▼ Clear 1
                    Clear (Z+stencil)
              ▼ <unknown scope> 1
                Clear (stencil)
              ▼ GUI.Repaint 2
                Draw Mesh
                Draw Mesh
```

CPU端性能瓶颈

Draw Call

建议控制在200范围以内

UI建议控制在10-40之间

多用FrameDebugger

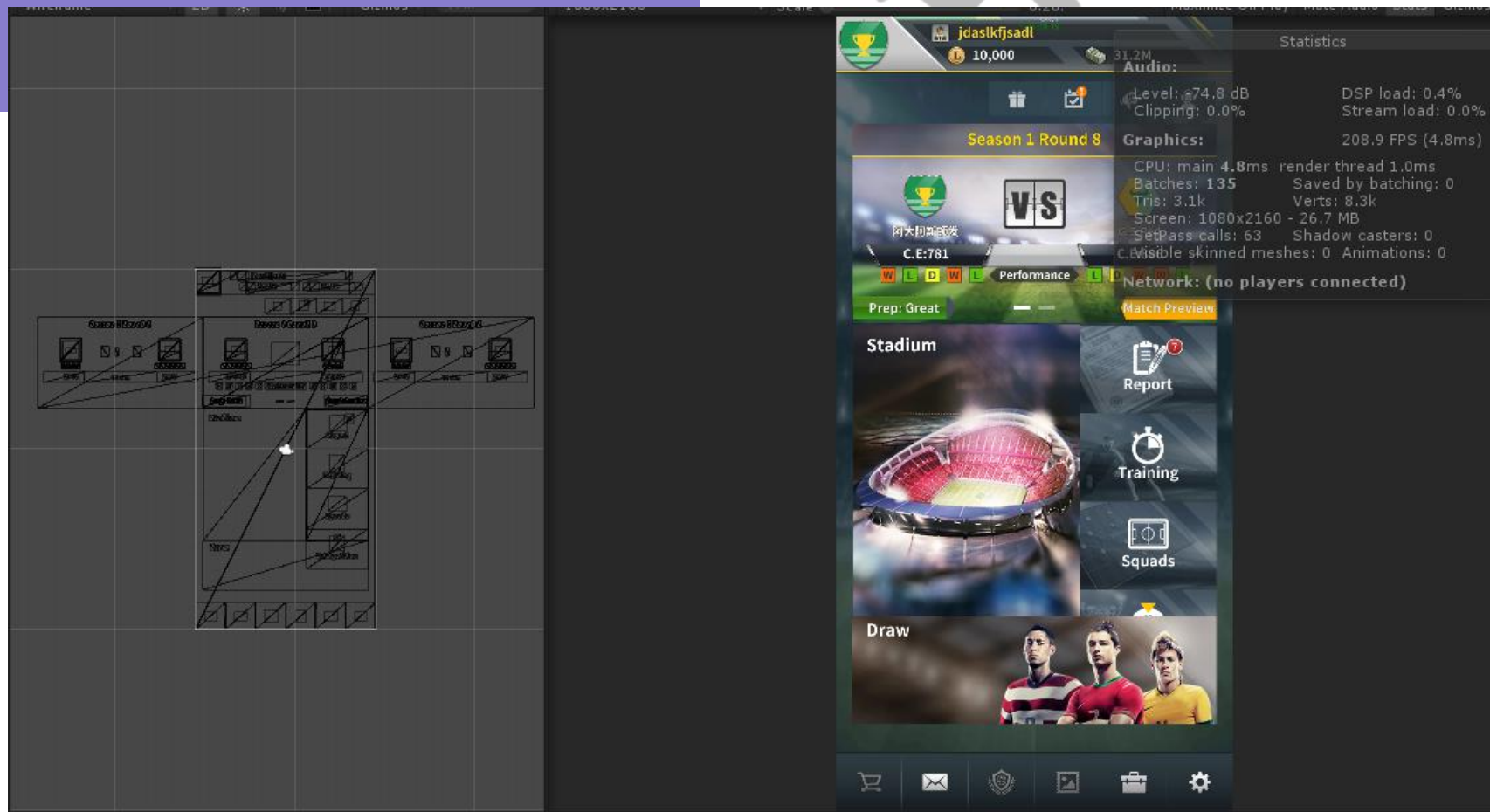
材质不同导致不能合批

Instance 材质 (在material后面跟着instance)

Lightmap不同

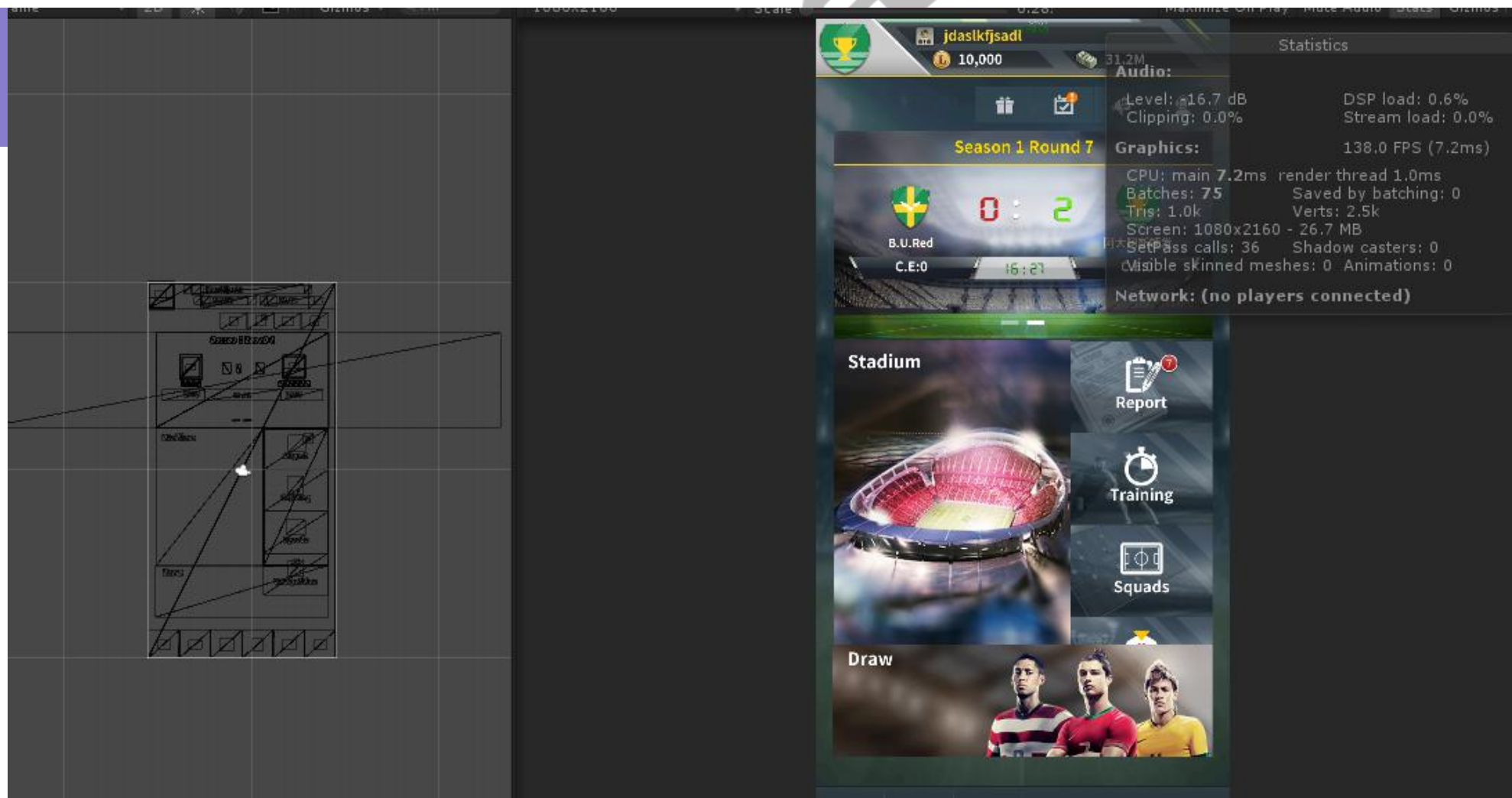
CPU端性能瓶颈

Draw Call



CPU端性能瓶颈

Draw Call

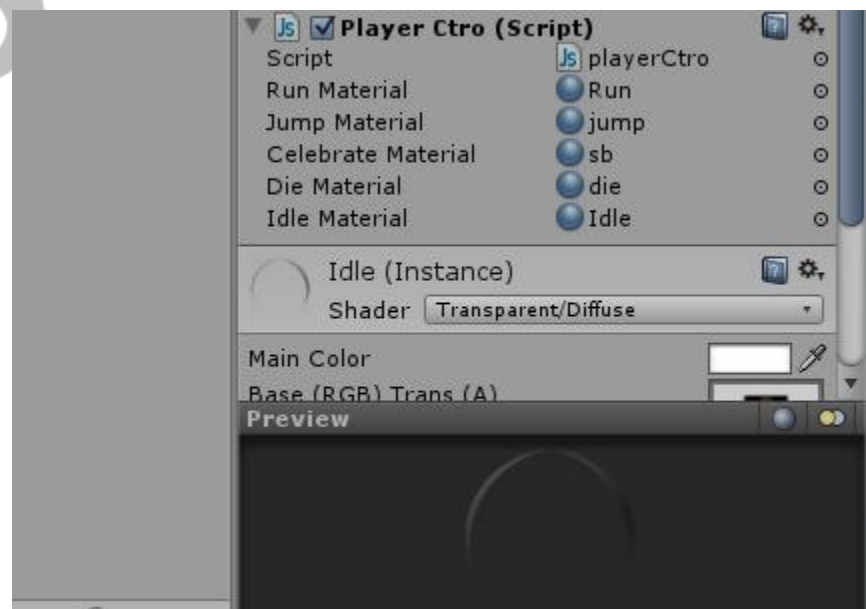


CPU端性能瓶颈

RenderState

避免 instance material 的生成

MaterialPropertyBlock的功能



GPU性能分析

1

填充率(FillRate)

2

带宽(Bandwidth)

3

Shader复杂度(ALU)

GPU性能分析

填充率 OverDraw

不透明物体

场景物体

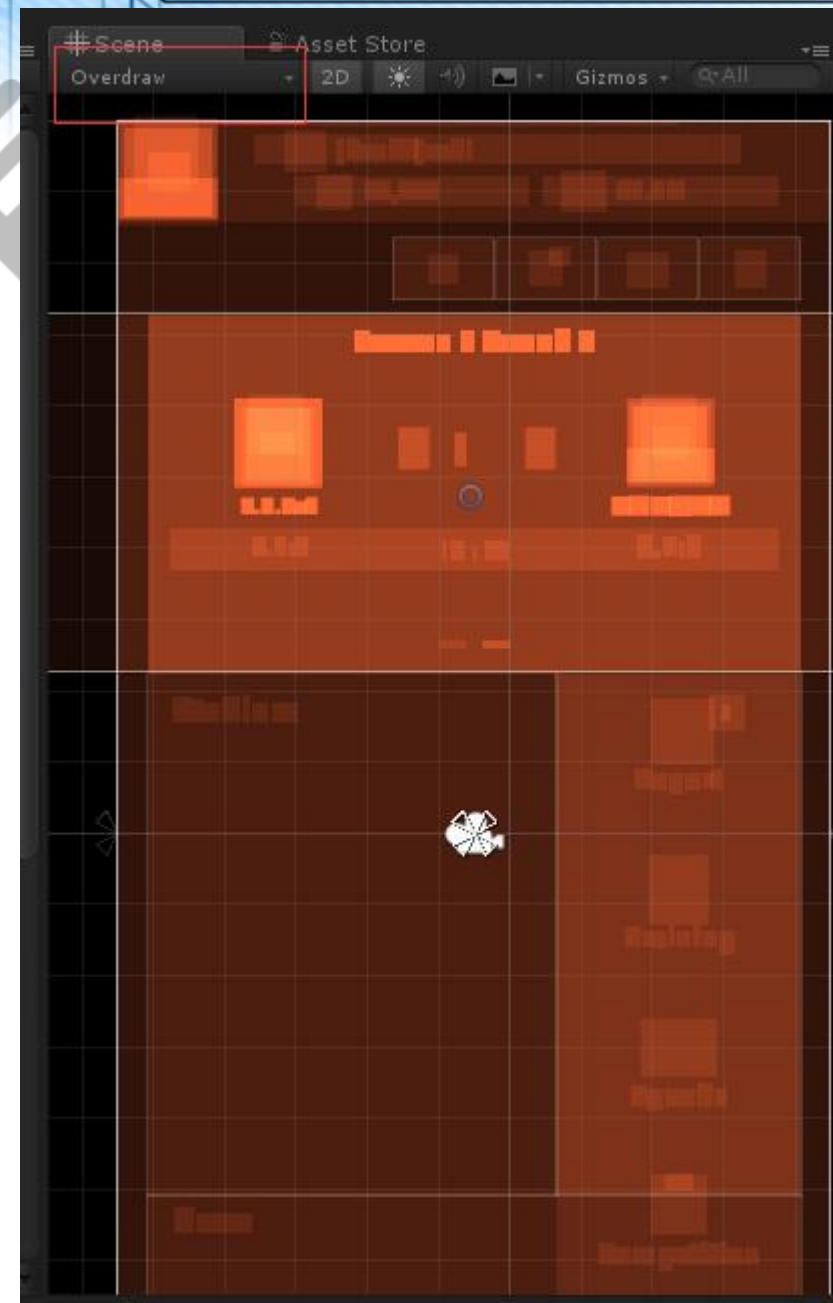
半透明区域

UI 粒子特效

GPU性能分析

OverDraw

在scene窗口中选择overdraw模式，
场景中越亮的地方表示overdraw越高

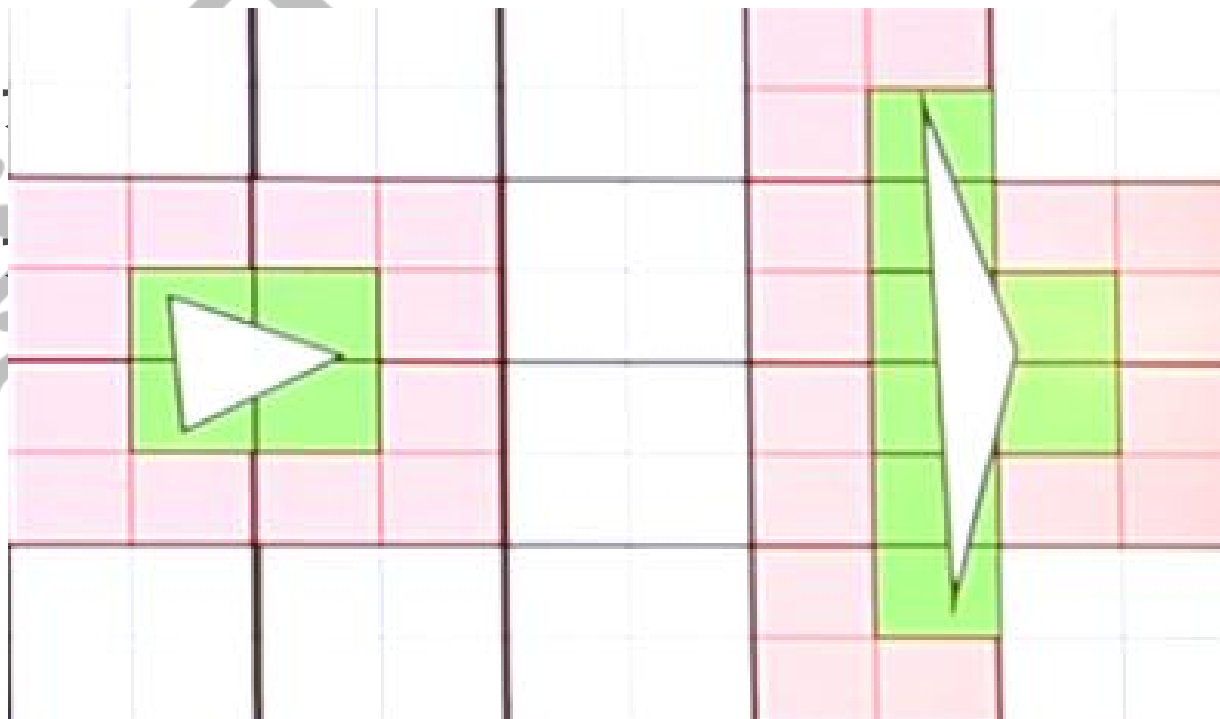


GPU性能分析

填充率 OverShading

左图浪费了75%的GPU

右图浪费了67%的GPU



GPU性能分析

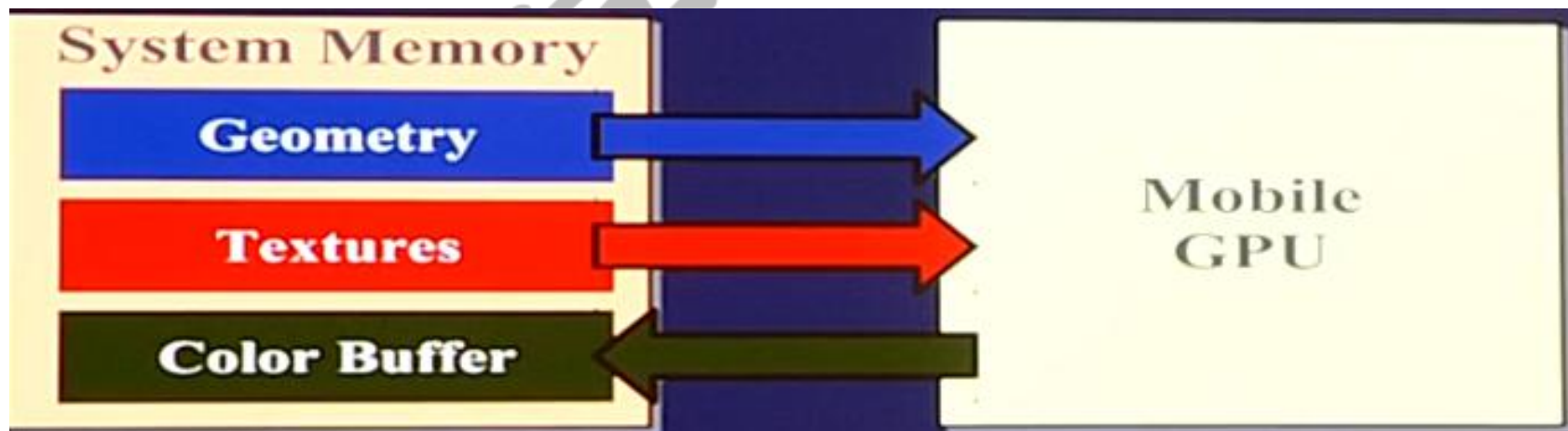
填充率 OverShading

LOD的应用

找到使用率低的Mesh简化

GPU性能分析

带宽



GPU性能分析

带宽

减少传输量-texture

开启mipmap

减少传输量

减少cache miss 的概率

纹理压缩

使用合理的纹理格式

使用dither 算法

GPU性能分析

带宽

减少传输量 -- Mesh

DynamicBatching

Skinned Mesh

Particle System

GPU性能分析

Shader(ALU)

Vertex Shader

Pixel Shader

UI模块

1

资源优化

2

CPU优化

3

GPU优化

资源优化

合理分配图集

同一个UI界面的图片尽可能放到一个图集中
共用的图片放到一个或几共享的图集中
不同格式的图片分别放到不同的图集中，例如透明(带Alpha)
和不透明（不带Alpha)的图片

资源优化

合理分配图集

关卡内的UI资源不要与外围系统UI资源混用

适当的降低图片的尺寸

在android设备上使用etc格式的图片，IOS上用PVRT格式

删除不必要的UI节点、动画组件及资源

CPU优化

UI网络重建开销

使用尽可能少的UI元素

减少Rebuild的频率

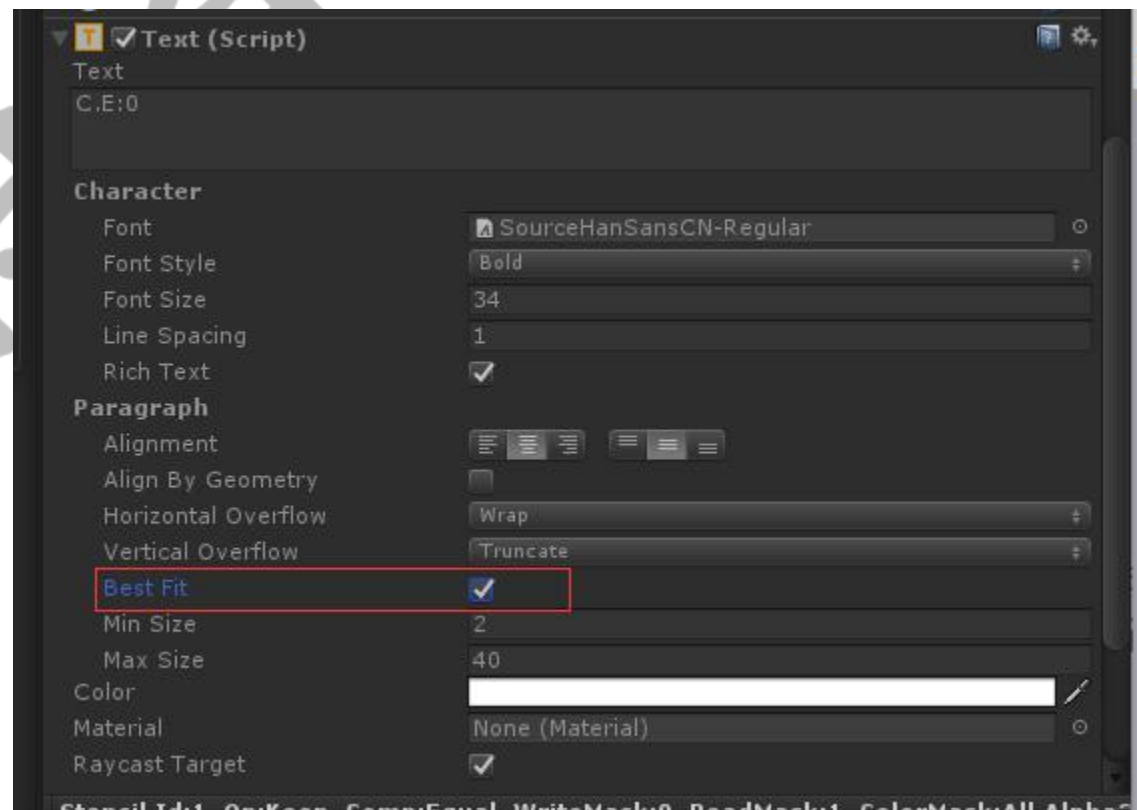
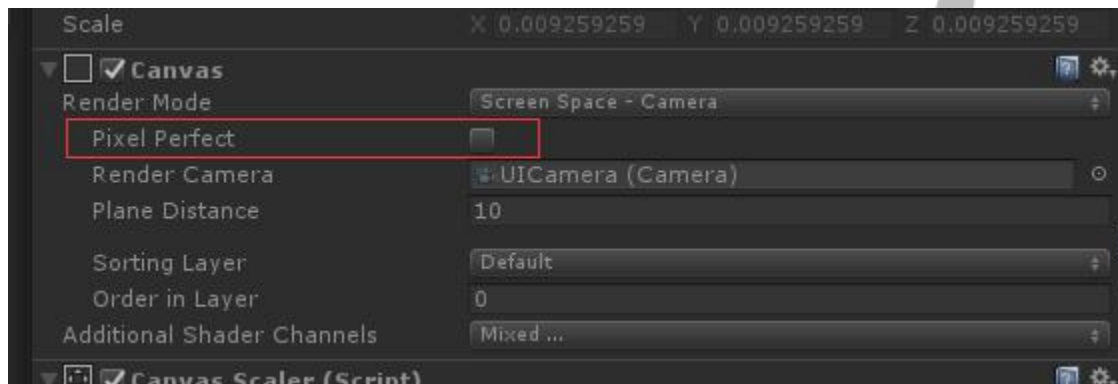
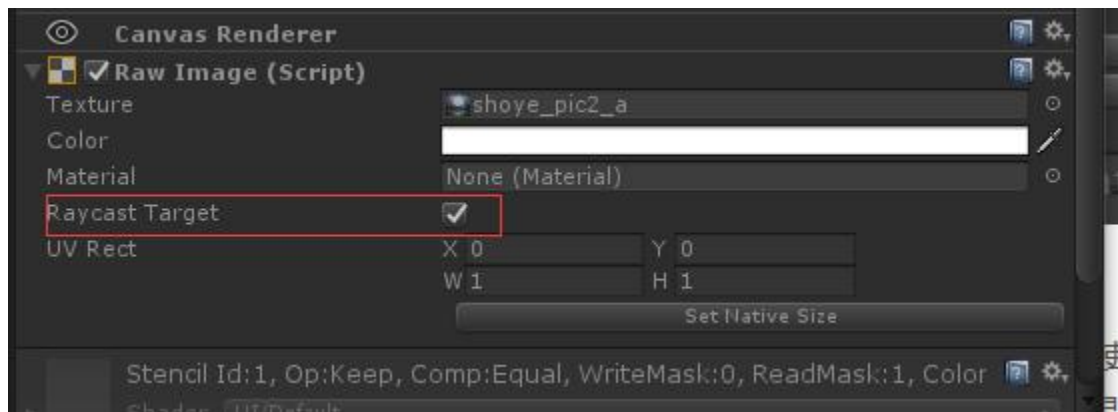
谨慎使用UI元素的enable与disable

谨慎使用Text的Best Fit选项

谨慎使用Canvas的Pixel Perfect选项

CPU优化

UI网络重建开销



GPU优化

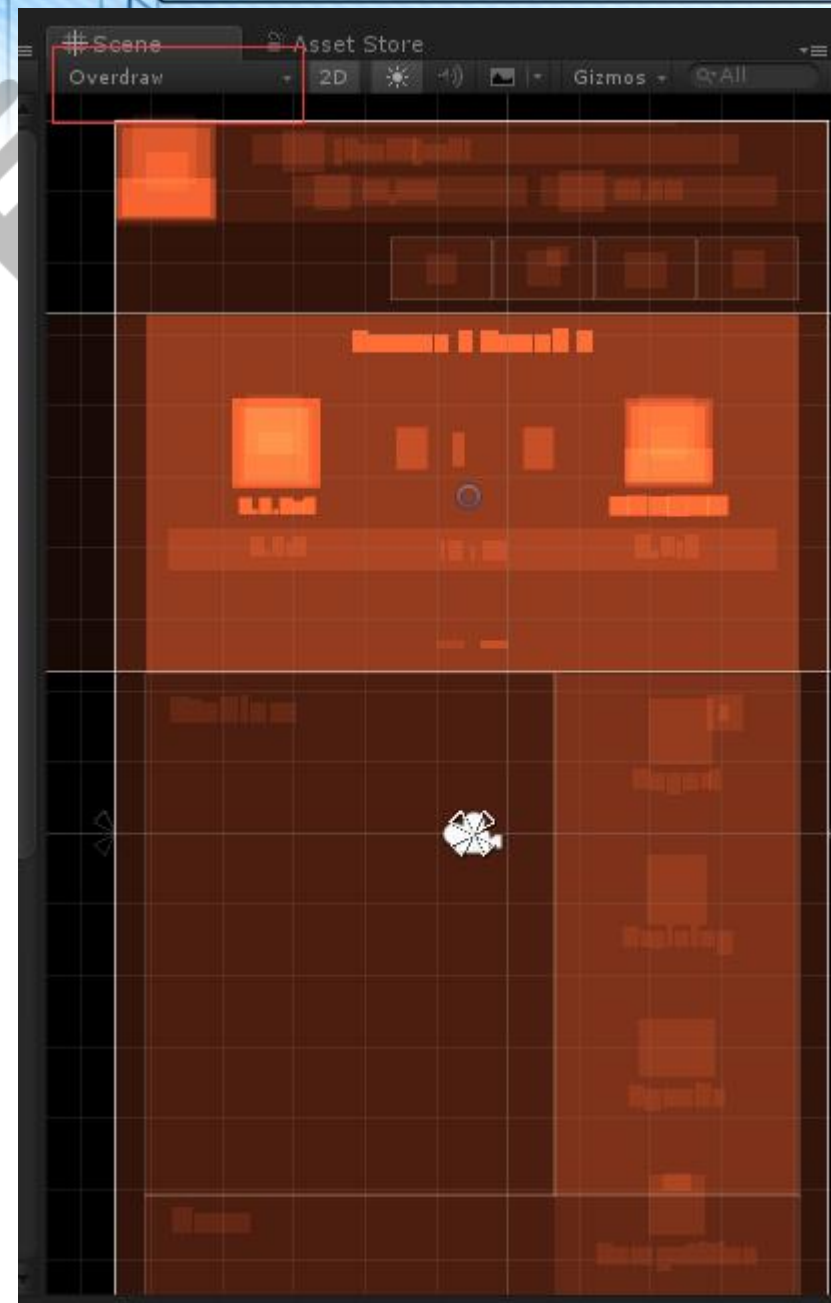
UI OverDraw

禁用不可见的UI，
不要使用空的Image,在Unity中， RayCast使用Graphi作为基本元素来检测touch

GPU优化

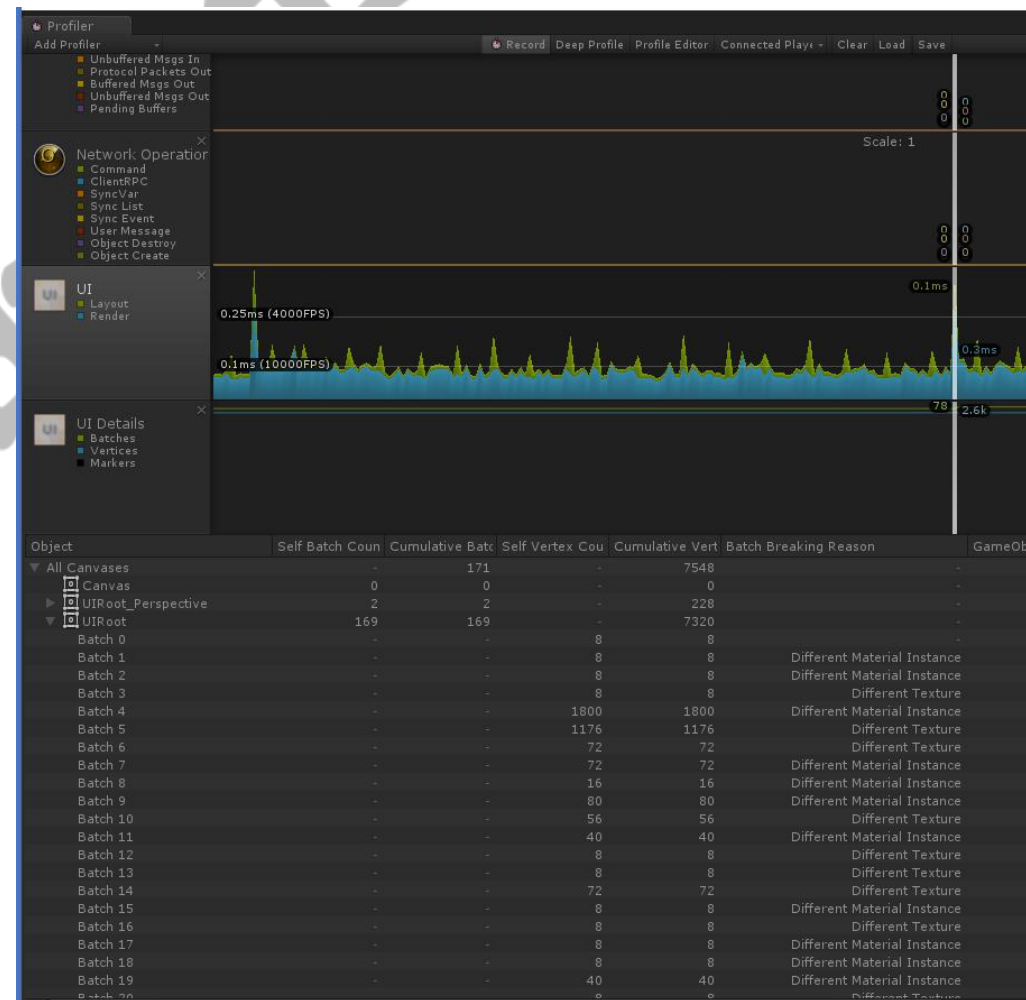
UI OverDraw

在scene窗口中选择overdraw模式，
场景中越亮的地方表示overdraw越高



工具使用

UI DrawCall



资源加载

1

资源加载与实例化

2

资源卸载&GC

资源加载与实例化

AssetBundle

2017.2加载速度大幅提升

Shader独立打包

资源加载与实例化

加载

关注 Loading.ReadObject 开销比较再去优化相关的纹理，网络或是动画shader等

使用对象池管理对象，避免反复创建销毁

资源加载与实例化

AssetBundle

Instantiate 开销大，则分帧初始化

Resources.unloadUnusedAssets 开销较大，建议在loading条的时候调用，另外还要控制场景资源大小与数量，这个与CG相关

资源卸载&GC

资源卸载与GC

Resources.unloadUnusedAssets 开销较大，建议在loading条的时候调用，另外还要控制场景资源大小与数量，这个与CG相关

内存管理

1

总体内存

2

Mono堆内存

3

纹理内存使用

总体内存&Mono内存

Sample视图操作

```
Simple -
Used Total: 480.0 MB  Unity: 209.0 MB  Mono: 49.9 MB  GfxDriver: 121.6 MB  FMOD: 95.2 MB  Video: 224 B  Profiler: 99.4 MB
Reserved Total: 0.66 GB  Unity: 378.1 MB  Mono: 65.3 MB  GfxDriver: 121.6 MB  FMOD: 95.2 MB  Video: 224 B  Profiler: 113.8 MB
Total System Memory Usage: 1.42 GB

Textures: 2352 / 85.9 MB
Meshes: 287 / 8.1 MB
Materials: 69 / 113.3 KB
AnimationClips: 1 / 2.0 KB
AudioClips: 20 / 93.9 MB
Assets: 16611
GameObjects in Scene: 351
Total Objects in Scene: 1973
Total Object Count: 18584
GC Allocations per Frame: 19 / 1.0 KB
```

总体内存&Mono内存

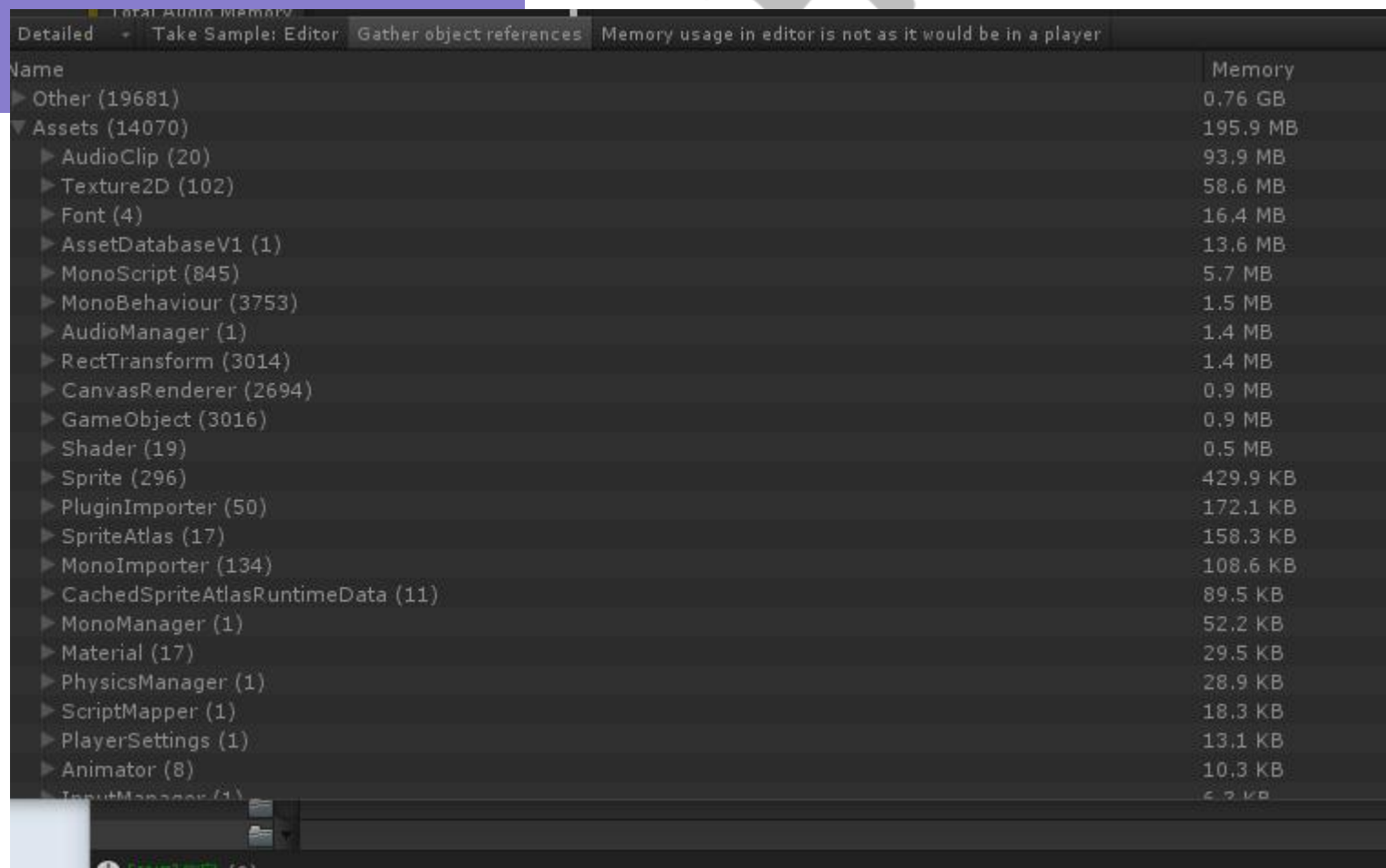
Profiler Sample视图操作

UseTotal:当前帧总共分配的内存大小,包括Unity内核,Mono内核,GfxDriver资源驱动,FMOD音效驱动,Profiler分析工具等内存占用。

ReservedTotal:当前帧向OS总共预先申请的总共分配内存大小,具体帧在使用具体内存时就从中分配,这样可以预先分配大点,而不用每次使用都向OS请求分配一次,包括Unity内核,Mono内核,GfxDriver资源驱动,FMOD音效驱动,Profiler分析工具等内存占用。

总体内存&Mono内存

Detail 视图操作



The screenshot shows the Unity Hierarchy window with the 'Memory' column visible. The window title is 'Local Audio Memory'. The tabs are 'Detailed', 'Take Sample: Editor', 'Gather object references', and 'Memory usage in editor is not as it would be in a player'. The table lists various assets and their memory usage.

Name	Memory
Other (19681)	0.76 GB
Assets (14070)	195.9 MB
AudioClip (20)	93.9 MB
Texture2D (102)	58.6 MB
Font (4)	16.4 MB
AssetDatabaseV1 (1)	13.6 MB
MonoScript (845)	5.7 MB
MonoBehaviour (3753)	1.5 MB
AudioManager (1)	1.4 MB
RectTransform (3014)	1.4 MB
CanvasRenderer (2694)	0.9 MB
GameObject (3016)	0.9 MB
Shader (19)	0.5 MB
Sprite (296)	429.9 KB
PluginImporter (50)	172.1 KB
SpriteAtlas (17)	158.3 KB
MonoImporter (134)	108.6 KB
CachedSpriteAtlasRuntimeData (11)	89.5 KB
MonoManager (1)	52.2 KB
Material (17)	29.5 KB
PhysicsManager (1)	28.9 KB
ScriptMapper (1)	18.3 KB
PlayerSettings (1)	13.1 KB
Animator (8)	10.3 KB
InputManager (1)	6.2 KB

总体内存&Mono内存

Profiler Detail 视图操作

- 1.该视图就是用来查看当前选中帧详细的内存占用信息,通过这些内存信息做一些深层次的判定,找出内存占用大的任务模块,并予以解决。
- 2.详细的内存占用主要包括以下几个方面:
Builtin Resource:unity editor或者unity自身的资源。
Not Saved:一些没有标记为保存的GameObject。
Assets:用户资源或者本地代码。
SceneMemory:场景中的GameObject对象或者Commonpent组件。
Other:以上模块除外的其他模块。
- 3.每个模块对应的内存占用都包含Memory内存大小和Ref Count引用计数,并且通过Refrenced By来记录详细的引用列表。
- 4.点击GameObject可以定位到编辑器中该GameObject的引用,方便查看。

纹理内存

缓存问题

纹理内存缓存问题

网络内存缓存的问题

动画片段，降低动画文件的浮点数精度

音频内存使用

THANKYOU