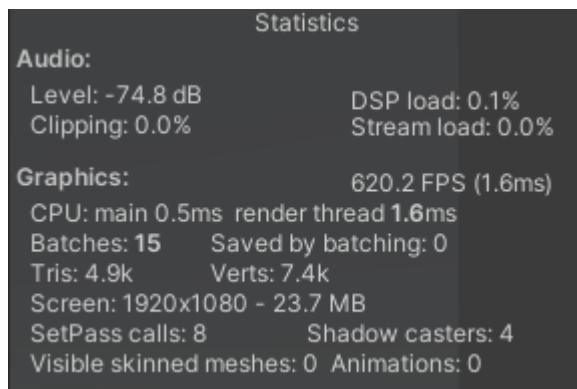


Stats面板



空场景的2个batch: 天空盒 + 屏幕的底

SetPass calls: Set Pass Call代表渲染状态切换，主要出现在材质不一致的时候，进行渲染状态切换。我们知道一个batch包括，提交vbo，提交ibo，提交shader，设置好硬件渲染状态，设置好光源属性等（注意提交纹理严格意义上并不包括在一个batch内，纹理可以被缓存并多帧复用）。如果一个batch和另一个batch使用的不是同种材质或者同一个材质的不同pass，那么就要触发一次set pass call来重新设定渲染状态。例如，Unity要渲染20个物体，这20个物体使用同种材质（但不一定mesh等价），假设两次dynamic batch各自合批了10个物体，则对于这次渲染，set pass call为1（只需要渲染一个材质），batch为2（向GPU提交了两次VBO，IBO等数据）。

GPU:mainthread: 主线程，monobehaviour

render thread: 专门的一个线程来控制我们的GPU

Batches: 游戏场景中的物体，分几个批次，提交给GPU绘制

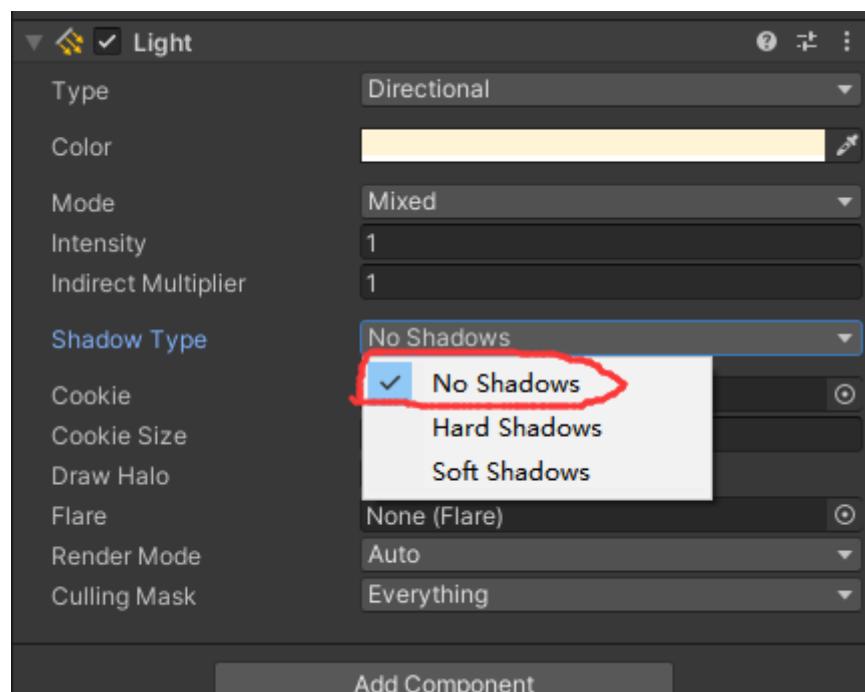
Save By Batches: 多少物体被合批绘制了

Shadow casters: 阴影开销

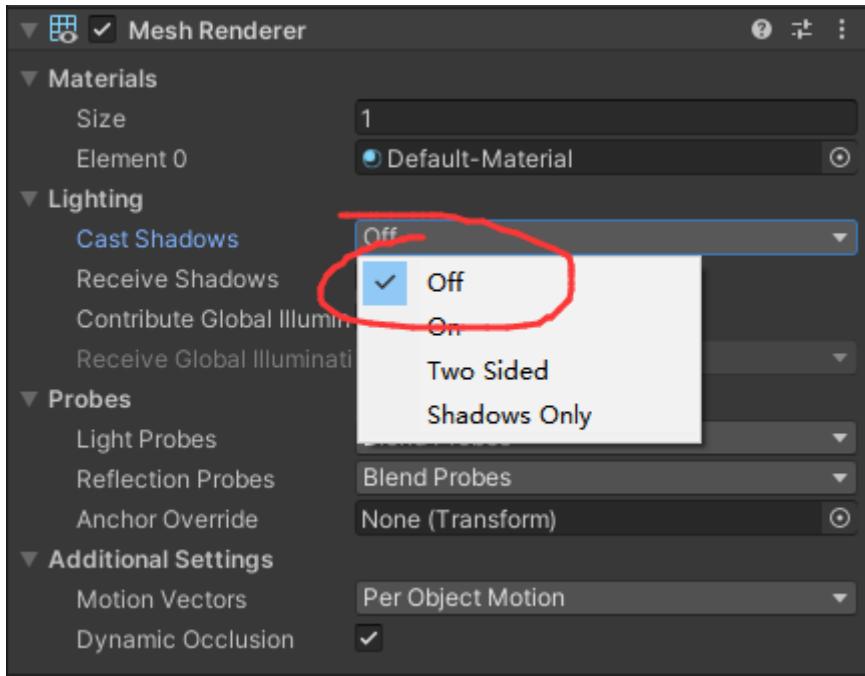
Visible Skinned Meshs: 游戏场景中可见的
skinnedMeshrender的数目

Animation: 动画开销

阴影开销 (set pass casters), 看不见的杀手



物体单个关闭阴影



Animation, SkinnedMeshRenderer开销

Animator替换Animation

将动画生成到Shader，并合批

合批次数上限：一个批次总顶点单元少于900 (模型总顶点数小于900)

unity的几种合批方法

Open Frame Debugger				
SetPass Calls: 5	Draw Calls: 5	Total Batches: 5	Tris: 2.0k	Verts: 5.1k
(Dynamic Batching)	Batched Draw Calls: 1	Batches: 1	Tris: 0	Verts: 0
(Static Batching)	Unbatched Draw Calls: 0	Batches: 0	Tris: 0	Verts: 0
(Instancing)	Batched Draw Calls: 0	Batches: 0	Tris: 0	Verts: 0
Used Textures: 6	2.7 MB			
RenderTextures: 12	37.7 MB			