Draw Call Batching

To draw an object on the screen, the engine has to issue a draw call to the graphics API (e.g. OpenGL or Direct3D). Every single draw call requires a significant amount of work on the part of the graphics API, causing significant performance overhead on the CPU side.

Unity combines a number of objects at runtime and draws them together with a single draw call. This operation is called "batching". The more objects Unity can batch together, the better rendering performance you will get.

Built-in batching support in Unity has significant benefit over simply combining geometry in the modeling tool (or using the **CombineChildren** script from the Standard Assets package). Batching in Unity happens **after** visibility determination step. The engine does culling on each object individually, and the amount of rendered geometry is going to be the same as without batching. Combining geometry in the modeling tool, on the other hand, prevents effecient culling and results in much higher amount of geometry being rendered.

Materials

Only objects sharing the same material can be batched together. Therefore, if you want to achieve good batching, you need to share as many materials among different objects as possible.

If you have two identical materials which differ only in textures, you can combine those textures into a single big texture - a process often called *texture atlasing*. Once textures are in the same atlas, you can use single material instead.

If you need to access shared material properties from the scripts, then it is important to note that modifying Renderer.material will create a copy of the material. Instead, you should use Renderer.sharedMaterial to keep material shared.

Dynamic Batching

Unity can automatically batch moving objects into the same draw call if they share the same material.

Dynamic batching is done automatically and does not require any additional effort on your side.

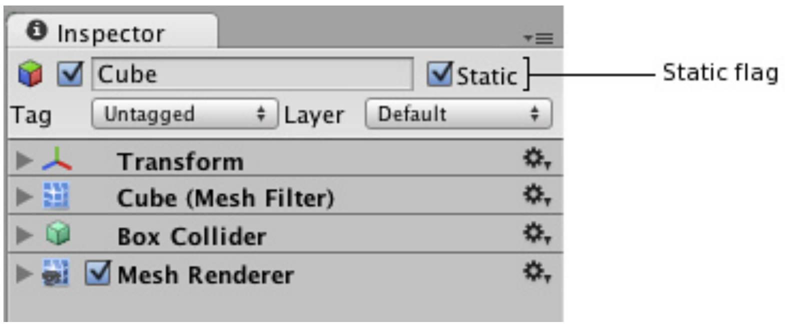
Tips:

- Batching dynamic objects has certain overhead **per vertex**, so batching is applied only to meshes containing less than **900** vertex attributes in total.
  - If your shader is using Vertex Position, Normal and single UV, then you can batch up to 300 verts and if your shader is using Vertex Position, Normal, UV0, UV1 and Tangent, then only 180 verts.
  - **Please note: attribute count limit might be changed in future**
- Don't use scale. Objects with scale (1,1,1) and (2,2,2) won't batch.
- Uniformly scaled objects won't be batched with non-uniformly scaled ones.
  - Objects with scale (1,1,1) and (1,2,1) won't be batched. On the other hand (1,2,1) and (1,3,1) will be.
- Using different material instances will cause batching to fail.
- Objects with lightmaps have additional (hidden) material parameter: offset/scale in lightmap, so lightmapped objects won't be batched (unless they point to same portions of lightmap)
- Multi-pass shaders will break batching. E.g. Almost all unity shaders supports several lights in forward rendering, effectively doing additional pass for them
- Using instances of a prefab automatically are using the same mesh and material.

Static Batching

Static batching, on the other hand, allows the engine to reduce draw calls for geometry of any size (provided it does not move and shares the same material). Static batching is significantly more efficient than dynamic batching. You should choose static batching as it will require less CPU power.

In order to take advantage of static batching, you need explicitly specify that certain objects are static and will **not** move, rotate or scale in the game. To do so, you can mark objects as static using the Static checkbox in the Inspector:



Using static batching will require additional memory for storing the combined geometry. If several objects shared the same geometry before static batching, then a copy of geometry will be created for each object, either in the Editor or at runtime. This might not always be a good idea - sometimes you will have to sacrifice rendering performance by avoiding static batching for some objects to keep a smaller memory footprint. For example, marking trees as static in a dense forest level can have serious memory impact.

Static batching is only available in Unity Pro for each platform.

Page last updated: 2012-07-29