Digital Opus

Music & Games — What's your symphony?

Mesh Baker 2 Manual

Table Of Contents

Quick-start (Combining Meshes and Materials)

This is not the only way to use Mesh Baker but it is the most common use case.

1. Create a new Mesh Baker object in your scene. If your source objects exceed 64k vertices use a Multi Mesh And Material Baker
    - GameObject -> Create Other -> Mesh Baker -> Mesh And Material Baker
2. Click 'Create Empty Assets For Combined Material'. This will create material assets for the combined material(s) and also an MB2_TextureBakeResults asset that contains information mapping materials to UV rectangles in the atlases.
3. Select shader on Combined Mesh Material. Mesh Baker will build a texture atlas for each texture property in this shader.
4. Add objects to combine. For best results, these should use the same shader(s) as the combined material(s), but they don't have to. Use the provided tools to make this fast and easy.
5. Set any options then click 'Bake Materials into a Combined Material'.
6. Look at warnings / errors in the console. Decide if action needs to be taken. Look at the combined material asset in the inspector to see that the generated atlases look correct. You may need to adjust non-texture properties in the combined material shader(s) to match the source material(s).
7. In the MB2_MeshBaker component click 'Bake'. This will create the combined mesh and a new game object that uses it. The combined mesh is an instance (not an asset) so it can't be used in a prefab. If you want to use the combined mesh in a prefab then select "output" -> "bake into prefab". The mesh will be saved as an asset and assigned to the provided prefab.
8. (optional) Disable renderers in source objects.

Once this has been done, you can remove the Mesh Baker object from the scene or keep it around for easy re-baking. If anything changes in any of your source models or textures, just bake again to regenerate the combined mesh and texture atlases.

How It Works

Mesh Baker consists of two core components. A MB2_TextureBaker component and a MB2_MeshBaker component. These components can be used together or separately.

The MB2_TextureBaker creates atlas assets and a special Material Bake Result asset that maps the source materials to UV rectangles in the atlases. Once this has been created then **any mesh that uses the source materials can be added to the combined mesh, even meshes that were not included in the texture bake.**

The user must provide:

- A Combined Mesh Material and set the shader on this material
- A Material Bake Result (MB2_TextureBakeResults)
- A list of game objects with MeshRenderers or SkinnedMeshRenderers to be combined

MB2_TextureBaker has a list of names of common texture properties that it looks for in the Combined Mesh Material shader. An atlas will be created for each texture property found in this shader. If the shader contains texture properties with non-standard names these should be added in the list of “Custom Shader Property Names”. For each object to be combined, the materials on that object are collected and the textures in those materials are added to the appropriate atlases. If a material does not have a required texture property or a texture is not assigned for a property, then a clear texture is created and added to the atlas. If a source material has extra texture properties not included in the Combined Mesh Material shader, then these are ignored. If a material uses tiling then that tiling will be baked into the atlas up to the “maximum tiling bake size”. Textures that are used more than once on the source objects will only be included once in the atlas provided they use the same tiling and offset.

The generated atlases are added to the Material Bake Result (MB2_TextureBakeResults) asset. This asset also contains information mapping source materials to their texture locations (UV rectangles) in the atlases.

The Material Bake Result asset can be used in other scenes or even exported with the combined material and atlases and used in another project.

The user then uses the MB2_MeshBaker component to combine the meshes. This can step can happen in a different scene or a different project as long as the user has generated a Material Bake Result asset. The user must provide:

- A Material Bake Result (MB2_TextureBakeResults) asset. This is usually the same as the one used by the MB2_TextureBaker component.
- A list of objects to be combined (game objects or prefabs with MeshRenderers or SkinnedMeshRenderer components). This is usually the same as the list of objects used to build the atlases, however it can include objects not on that list as long as all materials on included objects have been baked into the Material Bake Result.

As each mesh in the list of objects to be combined is added, mesh baker uses its materials to look up the correct UV rectangle in the MB2_TextureBakeResults object. The UVs in the mesh to be added are scaled to fit this UV rectangle. If “fix out of bounds UVs” was checked in the MB2_TextureBaker component than UVs are scaled to be in the range 0,0 .. 1,1. All submeshes on the source objects are collapsed into a single mesh unless the user is using the multiple combined materials option in which case Mesh Baker creates submeshes.

The MB2_MeshBaker component internally keeps track of the objects in the combined mesh so it is possible to quickly add and remove objects at runtime.

What is done with the combined mesh depends on the “output” option selected.

- Bake into scene object: A new game object is created in this scene and a MeshRenderer or SkinnedMeshRenderer is added. The mesh is an instance so the created game object won’t work as a prefab.
- Bake into prefab: The user must provide a prefab containing a game object. The mesh is saved as an asset and it is added to the prefab.
- Bake meshes in place: See the section Preparing Models For Static/Dynamic Batching (Bake Mesh Assets In Place)

Combining Skinned Meshes

Combining SkinnedMeshRenderers is no different than combining ordinary meshes except that the “Render” field should be set to “SkinnedMeshRenderer”. However, there are some things to consider when using SkinnedMeshRender components:

- You must leave the original SkinnedMeshRender object and its bones in the scene, since the original bones are used in the combined SkinnedMeshRender object. Disable the SkinnedMeshRender component on the source objects.
- Animations on the original SkinnedMeshRender component should still work. ***If animations stop working after combining, check the “culling type” on the animation.*** You have a few choices:
    - Set the culling type to “Always animate” or something other than “based on renderers”.
    - If you want to use culling type “based on renderers”, then the Animation must be above the combined SkinnedMeshRenderer in the hierarchy. This is usually only possible if baking one skinned mesh.
- SkinnedMeshRender objects to be combined do not need to share a common parent.
- **You can combine MeshRenderer objects with SkinnedMeshRenderers.** This is useful for adding hats, weapons, armor etc. to skinned characters. The transform that the MeshRenderer is attached to becomes a bone.
    - Parent the MeshRenderer object to the appropriate bone (for example a skinned character’s hand)

        - Position, rotate and scale it correctly
        - Include it in the list of objects to be combined
- If you want to make a prefab out of the combined SkinnedMeshRenderer, then all the bones (source objects) must be included in the prefab.
- You can customize characters by adding or removing swords, weapons, armour, etc. while the game is playing. To do so, you will need to bake everything that could possibly be added into the **combined material**. A subset of the objects should be baked into the **combined mesh**. Uncheck the Objects To Be Combined -> “Same As Texture Baker” field. Then add this subset of objects that you want to combine in the mesh.

Recommended Workflow

To use Mesh Baker quickly and efficiently, it is highly recommended that you use the ”Tools For Adding Objects”. These can save you hours of work.

1. Add a Mesh Baker object to your scene.
2. Click “Open Tools For Adding Objects” and then “List Shaders In Scene”
    - Look at the report and warnings in the console and decide how you want to group objects. It is best to combine objects that use the same material first, then the same shader.
3. Add one Mesh Baker object for each combined mesh that you want
    - Combine objects by selecting them in the hierarchy and clicking “Add Selected Meshes”. Mesh Baker crawls through the selected objects (including their children) and adds objects with Renderers attached. It applies the filters as it goes. You can filter by:
        - **Material**: only objects with this material will be added
        - **Shader**: only objects that use the same shader as the one in the provided material will be added
        - **Static**: only static objects will be added.
    - This makes it easy to add many objects quickly.
4. Bake the materials and meshes.
5. Keep the Mesh Baker objects around. You can disable them if you like. If any of your source objects change, then just re-bake.

If the source objects are needed in physics calculations or have scripts on them, you may want to keep the source objects in the scene with the render components disabled.

Preparing Models For Static/Dynamic Batching (Bake Mesh Assets In Place)

Unity’s static/dynamic batching only works with meshes that use the same material. If you have meshes that use different materials, the “Bake Mesh Assets In Place” feature can create a combined material, and create duplicate copies of your meshes that are adjusted to use this combined material. The duplicate meshes (using the combined material) will now be batched by Unity.

1. Create a new Mesh Baker object in your scene under the Game.
    - GameObject -> Create Other -> Mesh Baker -> Mesh And Material Baker
2. Create combined material asset(s).
3. Select shader on result material(s). Mesh Baker will build a texture atlas for each texture property in these shaders.
4. Add objects to combine. **These can be prefabs.** If you use scene objects, the translation/rotation/scaling will be baked into the saved meshes (which may not be what you want). You probably want the prefab at position (0,0,0), rotation (0,0,0) scale (1,1,1).
5. Set any options, then click 'Bake Materials into a Combined Material'.
6. Look at warnings / errors in the console. Decide if action needs to be taken.
7. Select output "bake mesh assets in place" and click "bake". This will create a new mesh asset for every mesh in the list of objects to combine. Look in the console for a list of all the meshes that were created.
8. For each prefab in the list of objects to combine.
    - Replace the mesh with the generated mesh
    - Replace the material(s) with the combined material(s)

Your objects are now ready for static/dynamic batching!

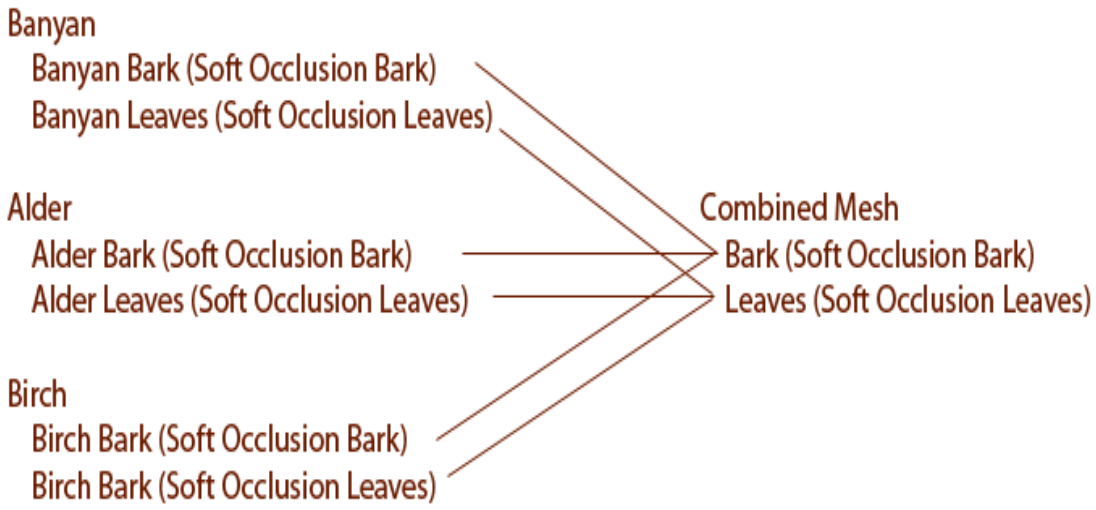
Combining Meshes at Runtime (Bake Texture Atlases Only)

See below. This is an advanced topic.

Multiple Materials:

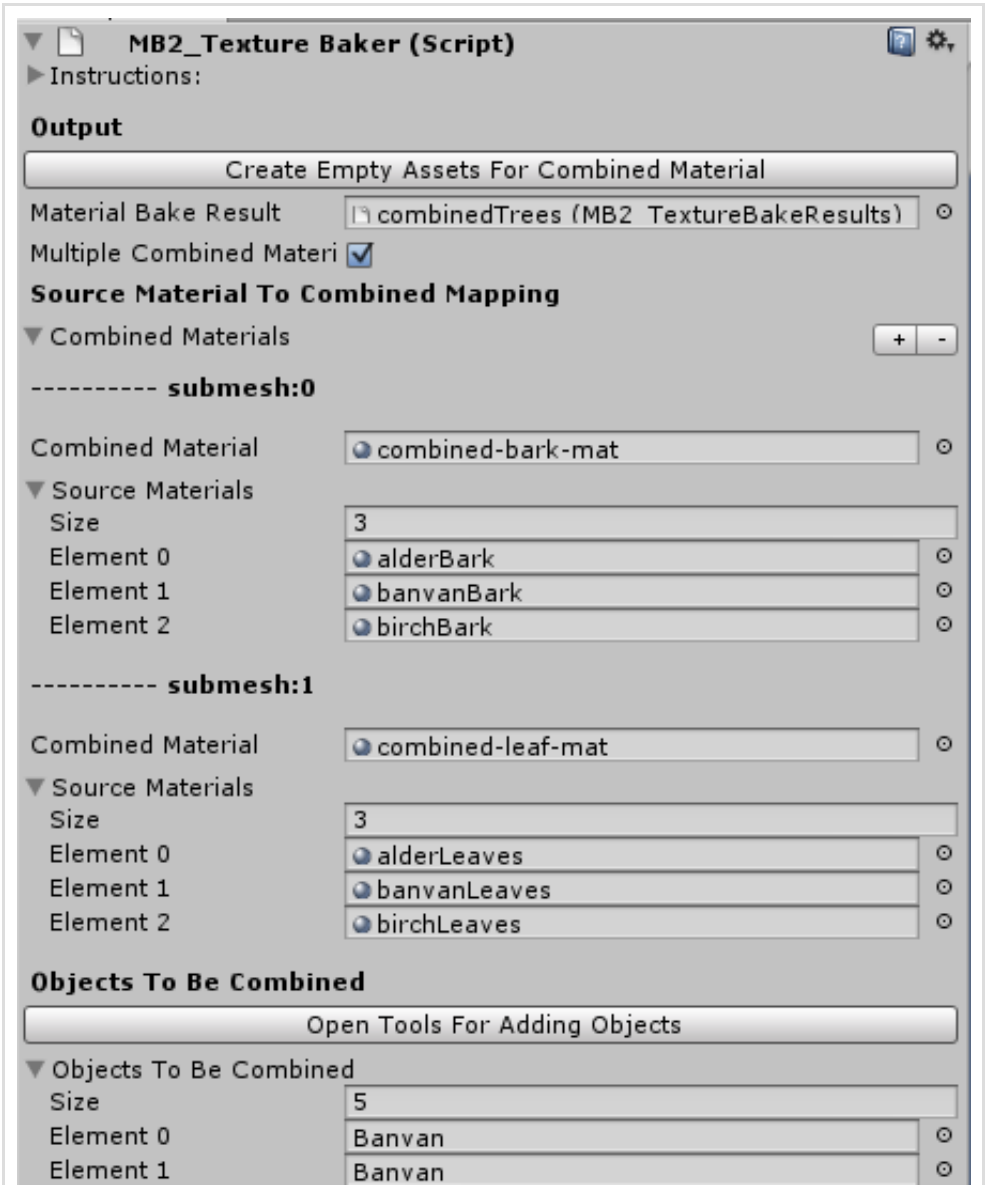
Setting up multiple combined materials is a lot of work. Before you do may want to try a regular bake which which will collapse all submeshes into a single mesh. This only takes a few seconds to set up and is often preferable as it results in fewer draw calls.

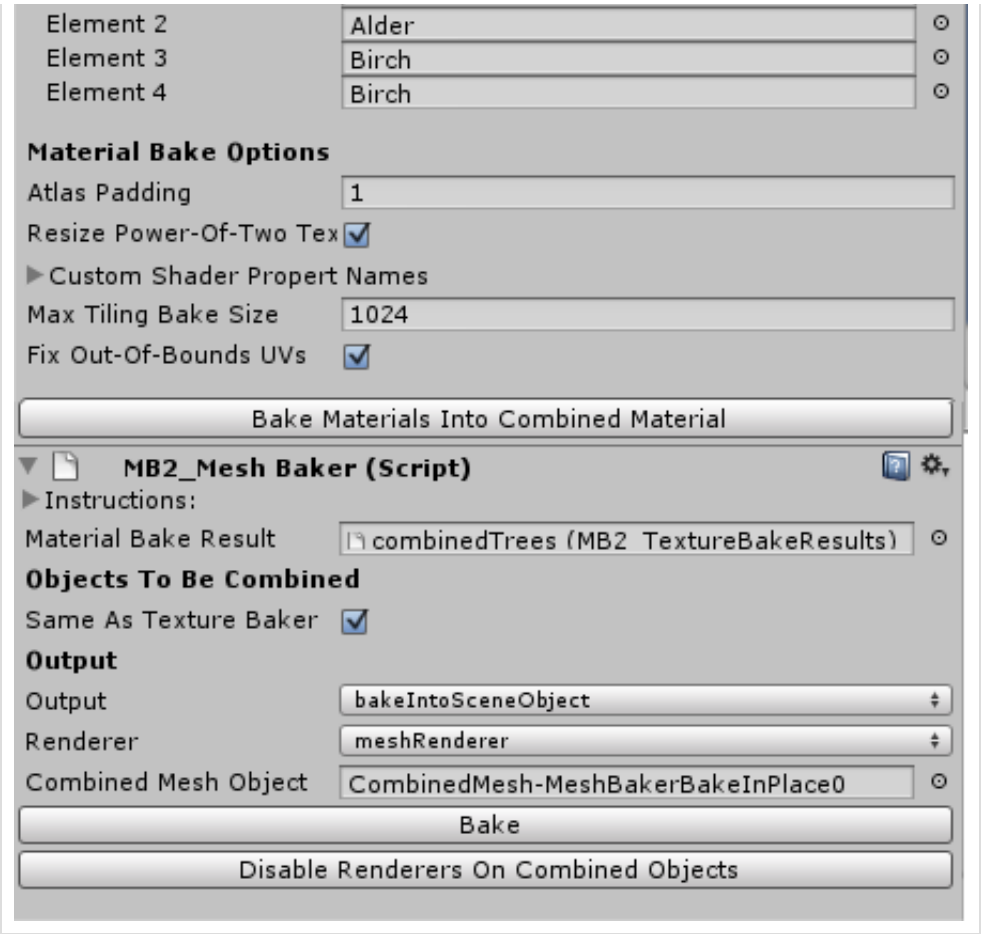
Mesh Baker can combine meshes with multiple materials (submeshes) into a combined mesh with multiple materials. You will need to define how materials on the objects to be combined should map to the materials on the results. Consider this trees example:



We want the bark source materials combined into the bark material, and the leaves source materials combined into the leaves materials.

1. Create a new 'Combine Or Alter Meshes Baker' object in your scene under the GameObject menu.
    - GameObject -> Create Other -> Mesh Baker -> Mesh And Material Baker
2. Click the "Multiple Combined Materials" checkbox. Notice that the "Combined Material" field is replaced by a list of materials.
3. Click the "+" button to add two combined materials
4. Click "Create Empty Assets For Combined Material". If necessary drag material assets for the combined materials into the "Combined Material" field.for each combined material
5. Set the shaders on these materials to:
    - "Nature -> Tree Soft Occlusion Bark"
    - "Nature -> Tree Soft Occlusion Leaves"
6. Set the size of each source materials to 3 and drag the bark and leaf materials to the appropriate slots. Your mapping should look like this:





Options

Fix Out-Of-Bounds UVs

What It Does: Some models have UVs outside the range 0..1. This produces the same result as tiling. These meshes are hard to combine, because the tiling will pick up neiboring texures in the atlases. Fix Out-Of-Bounds UVs will scale the UVs in the model so that they fit inside the 0..1 range. Tiling will be baked into the combined texture.

When To Use It: The “List Shaders In Scene” report will tell you if you have UVs outside the range 0..1 on your models. Fix Out-Of-Bounds UVs is often the only way to have tiling models share textures in an atlas. Be aware that using this feature is sometimes not the best choice. For example, consider a castle model with many different parts (towers, walls, buttresses, etc). Each part tiles the same tiny brick texture. These can all be combined together (with tiling) provided the brick texture is the only texture in the atlas. This is probably a better choice than baking the tiling on each piece into one huge atlas.

Atlas Padding

What It Does: Adds a clear border around the textures in your atlas so that adjacent textures don’t bleed into each other.

When To Use It: Normally, a value of 1 or 2 pixels produces good results. If your textures to be combined already have a clear border, you could set the border to 0.

Resize Power of 2 Textures

What It Does: If checked, this option shrinks power-of-two textures so they have a border of "atlas padding" around them. For example a 128×512 texture with 2-pixel padding is shrunk by 4 pixels, so that it is 128×512 after padding. This only affects power-of-two textures (32, 64, 128, 512)

When To Use It: You will usually want to leave this feature on. This option can dramatically reduce wasted space in the atlases. Unity's texture packer likes to jump by powers of two (eg.. 512×512 jumps to 512×1024). If you are combing two source textures that are 512×512 each then, after padding (514×514) the atlas will jump to 1024×2048. If you have textures that you really do not want resized, then turn this feature off or set the padding to 0.

Custom Shader Property Names

What It Does: Mesh Baker looks at the properties in the combined mesh shader to decide which atlases to build. It has a list of built in texture property names to look for: "_MainTex", "_BumpMap", "_BumpSpecMap", "_DecalTex", "_Detail", "_GlossMap", "_Illum", "_LightTextureB0", "_ParallaxMap","_ShadowOffset", "_TranslucencyMap". If any of these are present, then it builds an atlas for that texture property.

When To Use It: If you are using a custom shader, then it may include a texture property that is not on this list. You can include this property by typing its name in the list of Custom Shader Property Names. Do not type the name that is visibile in the inspector. You will need to open the shader and look at list of properties and use the name from there.

Lightmapping UVs:

Instructs meshbaker what to do with the UV2 channel in the combined mesh. The UV2 channel in Unity is reserved for lightmapping.

- **ignore_UV2**: No UV2 channel will be generated.
- **copy_UV2_unchanged**: Use this if all your source meshes use UV2 for a purpose other than lightmapping and you want these values preserved in the combined mesh.
- **preserve_current_lightmapping:** Use this if your source meshes are already lightmapped and you want to preserve the lightmap. All source meshes must use the same lightmap.
- **generate_new_uv2_layout:** Use this if you want to generate lightmaps using the combined mesh after it has been generated. Make sure that the combined scene object is 'static' and 'enabled' before baking lightmaps.

The MultiMeshBaker Component

Use this component if the number of vertices in your source objects exceeds 64k. It will create as many combined meshes as necessary to accommodate the objects.

Best Practices

For best results results, source meshes and materials should:

- Use the same shader
- Have all textures present
- Have all textures in a single material be the same size

The console is your friend. Look there for warnings and other information. Click on the line that says “Report” to see a list of all the textures that were combined and the objects they are on. This is useful for tracking down problems.

Use the “List Shaders In Scene” report to plan what to combine in your scene.

Experiment. Try different options to see what produces the best results.

If textures are missing, a default empty texture will be created. If textures are different sizes, they will be copied and resized to the same size as the largest texture in each material.

Runtime Use (Advanced)

If you want to add, delete or move objects in a combined mesh, then you will need to use the runtime API.

HOW IT WORKS:

- When a MB2_TextureBaker creates combined materials, it creates an MB_TextureBakeResult asset which contains a mapping of materials to UV rectangles in the atlases. It is recommended to do this step in the editor since it is slow and only works with uncompressed textures. The MB2_TextureBaker can be removed from the scene after it has baked the combined material.
- A MB2_MeshBaker component is added to the scene and the “Material Bake Result” field is set to the MB_TextureBakeResult asset created in the previous step.
- At runtime, when an object is added, MB2_MeshBaker uses the object’s material to look up the UV Rect. The mesh uvs are adjusted as the object is being added to the combined mesh.
- The API has methods AddDeleteGameObject() and UpdateGameObject() for adding and deleting and updating the game objects. Apply() changes to the mesh.

TYPICAL USE PATTERN:

The developer decides which objects he would like to be able to combine at runtime in each Mesh Baker instance.

- Create a “Combine Materials Only Baker”
- Bake a Combined Material as per instructions described in previous sections
- Add a MB2_MeshBaker component to the scene either from the GameObject menu or create an empty Game Object and add the script. The MB2_TextureBaker component won’t be needed.
- Set the “Material Bake Result” field to the asset generated in the material bake.

This Mesh Baker instance is ready to be used at runtime. A typical use pattern would be:

```
using UnityEngine;
using System.Collections;

public class MB_Example : MonoBehaviour {

  public MB2_MeshBaker meshbaker;
  public GameObject prefab;

  GameObject go1, go2;

  void Start(){
    // Instantiate some prefabs
    go1 = (GameObject) Instantiate(prefab);
    go1.transform.position = new Vector3(5f,5f,5f);

    // Can use a prefab not baked into the materials to combine
    // as long as it uses a material that has been baked
    go2 = (GameObject) Instantiate(prefab);
    go1.transform.position = new Vector3(5f,5f,5f);

    //Add the objects to the combined mesh
    GameObject[] objsToCombine = new GameObject[2] {go1,go2};
    meshbaker.AddDeleteGameObjects(objsToCombine, null);

    //apply the changes we made this can be slow. See docs
    meshbaker.ApplyAll();
  }

  void Update () {
    go1.transform.position =
      new Vector3(5f,5f*Mathf.Sin(Time.time),5f);
    go2.transform.position =
      new Vector3(5f,5f, 5f*Mathf.Cos(Time.time));
    GameObject[] objsToUpdate = new GameObject[2] {go1,go2};
    meshbaker.UpdateGameObjects(objsToUpdate);
```

```
    }

    void LateUpdate(){
      //Apply, after this and other scripts have made changes
      //Only want to call this once per frame since it is slow
          //Only change verts since it is slow to update everything
      meshbaker.Apply(false,true,false,
                      false,false,false,false,false);
    }
}
```

Performance Considerations

- Try to call AddDeleteGameObjects as few times as possible in a frame. It takes almost the same amount of time to add one object as many so batch the adds and deletes. It is good practice to do this in LateUpdate after collecting objects to add and delete.
- You can delete and add the same game object in a single AddDeleteGameObjects call. This is useful if you want to change the material on an object or you have changed the mesh in an objects MeshFilter.
- Apply takes approximately the same amount of time to execute Apply as AddDeleteGameObjects.
- There is a roughly linear relationship between the number of channels and time to AddDeleteGameObjects/Apply. So a mesh with vertices, colors will bake twice as fast as one with vertices, normals, tangents, uvs.
- Update is about 3x as fast as AddDeleteGameObjects so if you want just want to move objects around in the combined mesh use Update instead of AddDeleteGameObjects. It also puts a much lower load on the garbage collector.
- If you want to move objects around almost every frame then consider baking into a SkinnedMeshRenderer. It is much faster than Update.
- There are effectively three copies of each mesh in memory when a mesh is baked (the source mesh, the combined mesh and internal buffers of the MeshBaker object) *If you don't intend to modify a combined mesh then consider deleting the source meshes and the MeshBaker object from the scene:*

Baking Materials at Runtime

Baking textures at runtime does incur a serious performance hit. Expect baking to take 1-20 seconds or longer to bake atlases at runtime.

For building atlases at runtime it is essential that:

- textures be in trucolor/RBGA32 format
- textures have read flag set

Try to avoid resizing and tiling. It can reduce bake time by 80%

- build padding into textures so you don't have to pad.
- don't use padding when creating atlases
- don't use tiled materials

If you are having problems look at the Debug Log on the device

Here is a simple example:

```
using UnityEngine;
using System.Collections;

public class BakeTexturesAtRuntime : MonoBehaviour {
        public GameObject target;
        float elapsedTime = 0;

        void OnGUI(){
                GUILayout.Label("Time to bake textures: " + elapse
                if (GUILayout.Button("Combine textures & build cor
                        MB2_MeshBaker meshbaker = target.GetCompor
                        MB2_TextureBaker textureBaker = target.Ge

                        //These can be assets configured in edito
                        // or you can create them
                        // on the fly like this
                        textureBaker.textureBakeResults =
                                new MB2_TextureBakeResults();
                        textureBaker.resultMaterial =
                                new Material( Shader.Find("Diff

                        textureBaker.CreateAtlases();

                         //only necessary if your not sure whats
                         //in the combined mesh
                        meshbaker.ClearMesh();
                        meshbaker.textureBakeResults =
                                        textureBaker.textureBakeR
                        //Add the objects to the combined mesh
                        meshbaker.AddDeleteGameObjects(
                            textureBaker.GetObjectsToCombine().To
                            null);

                        meshbaker.Apply();
                }
        }
```

```
}
```

API

Full API Documentation

Mesh Baker was created by Ian Deane. Other assets by Ian Deane include:

- ***Fast Shadows***
- ***Mesh Master***
- ***Ramp Brush***
- ***Crater Brush***

Mesh Baker is available in the asset store.