Advanced Batcher Documentation

By Eduardo Jimenez

Introduction

This Unity plug-in is meant to help you with offline batch generation. This is a technique that can be used to optimize your scenes and reduce draw calls. It reduces the number of textures and materials used in a scene and lump together objects that share materials for a quicker rendering.

Motivation

The first question you may ask yourself is, what do I need this for? Why should I pay to get this plug-in, particularly since Unity already has 'dynamic batching' and 'static batching'? To answer this questions I'll try to explain to the best of my understanding, which is definitely far from an in-depth understanding, how batching helps rendering. I apologize in advance if I'm making mistakes or simplifying the way render works internally in a graphics card. I've got an overall understanding of it, but definitely not a very in depth one. I'll be happy to fix any issues you find in the text.

To render something on screen you need to pass a bunch of parameters to the graphics card that change its state. You pass things like the textures to use, the colors to set to the different registers, other variables (such as the specular 'intensity') that go into registers, etc. It also sets the vertex and index buffers with the ones for the object to be rendered, and then it asks the graphics card to render it. I believe that actually happens at a deferred moment. You actually batch all that info and the graphics card will do it when it finishes the previous renders.

The set of registers that define a render along with the textures and the vertex and index buffers define what, I believe, is called the graphics card state. Every time you change that the graphics card has to stop rendering, change the state, and start rendering again. During the state change time the buffers must be emptied and re-fill so no rendering is done which is, to put it simply, 'wasted time'. On the other hand, it's necessary to do so sometimes. Every time we change the state and pass a bunch of verts/indices to be rendered we say we've done a Draw Call *(that's so because we've actually called a graphics card function to render the stuff we set). Thus, having too many Draw Calls is bad for performance.*

So, what can we do to reduce the number of Draw Calls? Simple: batching! So we lump together objects that share the same material parameters so that they can be rendered together. That way, instead of rendering, let's say, 2 objects of 500 tris, you can do a single render call with 1000 tris. That, in most cases, is going to be faster.

On the other hand, we only merge together objects is if they all have the same paremeters, that is, they share exactly the same material. That's reduces a lot the number of objects you can batch. Also, batching things that are too far from each other may be counter productive because that means you're going to render more polys than they may be necessary. If you add a few tris it's not going to be an issue, but if you have 4 different objects in different parts of the scene you may be rendering lots of polys that are not visible. So it's preferable to batch together stuff that's physically close in the scene.

To increase the objects that can be batched with a single material we can do things such as merge several textures in a single texture, or move the colour from the material properties to the vertices. We'll go more in depth with these options later on. We can also limit how far we want to merge objects, what's the maximum number of polys a merged object should have, etc.

The last question you'll ask yourself is: but why use an offline batcher instead of using the batching options that Unity gives me. Well, the batching options from Unity are really useful and may speed up your game considerably, but they are not perfect. And more concretely, if you can tailor your scene to optimize it for batching, it's always going to perform better than using tools that are executed real-time. As a rule of thumb, every calc you do at compile time is going to save you time in the game, and that's what all this is about, right?

What should I batch?

First and most importantly, batching is particularly efficient with static objects. By static objects I mean those that are not going to move, at least not relative to each other. So, for instance, the backgrounds of your game are usually a good fit for it, and they are normally the element that more triangles (and normally) more draw calls generate.

If the objects are not going to move relative to each other they can be batched together too. So, for instance, you have a rigid object, formed of several objects, let's say an sculpture, in your game. This object may be transported from one place to another, but their objects always remain static relative to each other. That would be a good fit for batching, but the batch should be generated only for those objects and in a particular way.

We're also considering adding new batching tools to integrate objects that move, or even skinned meshes. These features will be available through an update.

The Tool

So now we understand the what and the why, it's time to learn the how. We have developed this tool focusing on simplicity. This tool is meant to be really easy to use, generating results quickly and empowering the user with just a bunch of simple, easy to understand, parameters.

There are 3 different tools that appear in the new GameObject->Batching menu item:

- *Unhide Batched Meshes*
- *Simple Offline Batching*
- *Advanced Offline Batching*

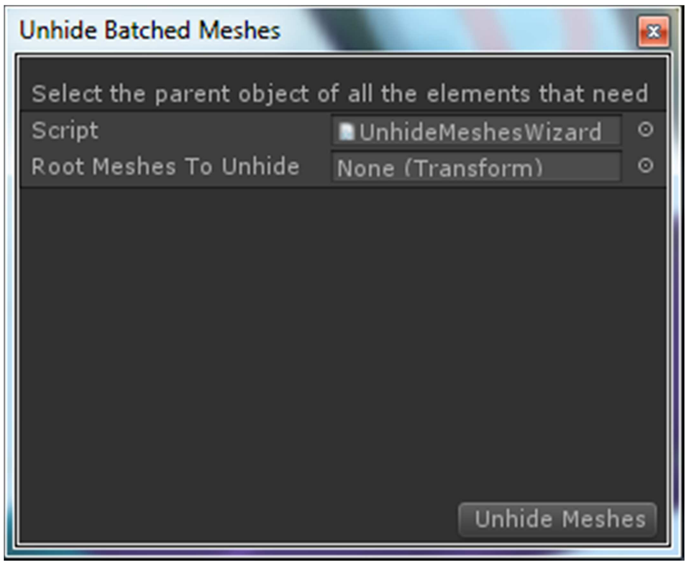
We'll see each of these tools separately

Unhide Batched Meshes

This is an option to be used after doing a batch. Batching a bunch of objects will hide their MeshRenderer components. Obviously if you generate a new batched mesh with all the objects you're merging together you don't need to draw the old objects. The hiding of the 'old meshes' is done automatically in the batching process. But sometimes the batch didn't work as you expected, you're not happy with the results or you simply want to move one of the objects of the batch and re-do the batch. Then you only have to delete the batched objects (they only contain the MeshRenderers for the new meshes no physics or logic is copied there) and then use this tool to unhide the original meshes.

You have 2 options:

- *You can select the parent object of the batched meshes. Then go to menu and select the option Unhide Batched Meshes. It will then unhide any disabled MeshRenderer in the given object and any of its children*
- *You can deselect anything in the scene and then go to the menu option. A dialog will appear with a single parameter:* Root Meshes To Unhide. *That must be the root object where all your hidden objects live.*


Unhide Batched Meshes

Select the parent object of all the elements that need

Script UnhideMeshesWizard

Root Meshes To Unhide None (Transform)

Unhide Batched Meshes dialog box

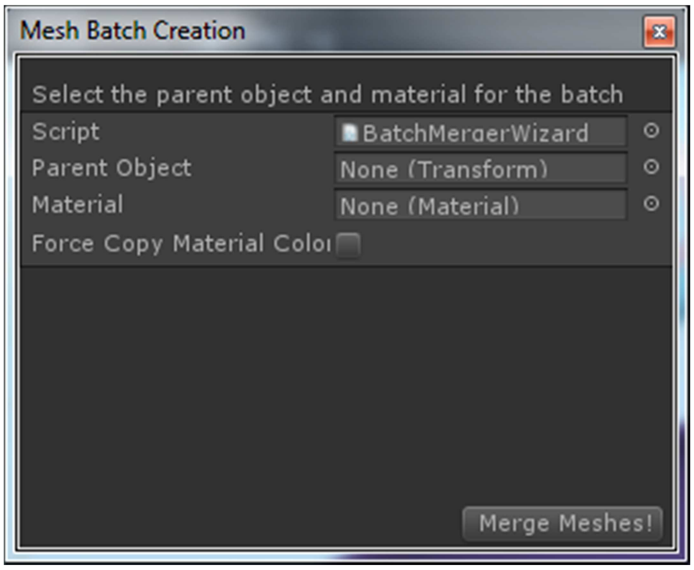
This dialog performs a very simple operation: it traverses all the children of the given root, gets all the MeshRenderers there and changes their enabled property to be true. The mesh batcher will batch objects with a MeshRenderer disabled and the unhide tool will re-enable them as well. If you don't want one of the objects either batched or re-shown you should separate it yourself out of the root object.

Simple Offline Batching

This tool is very useful for when you already have the same material and just want to batch a bunch of objects together. All the objects you want to batch must be part of a hierarchy with a single root object and no other MeshRenderer must be part of that hierarchy branch.

Assuming you don't have all the objects you want to merge under a single root you must create a new empty object (GameObject->Create Empty), give it a meaningful name, select all the objects you want to batch and move them under the newly created object.

Then you go to GameObject ->Batching->Simple Offline Batching and this will open a new dialog for this option:


Mesh Batch Creation

Select the parent object and material for the batch

Script: BatchMergerWizard
Parent Object: None (Transform)
Material: None (Material)
Force Copy Material Colo

Simple Offline Batching dialog box

This is again a very simple dialog box with only 3 options:

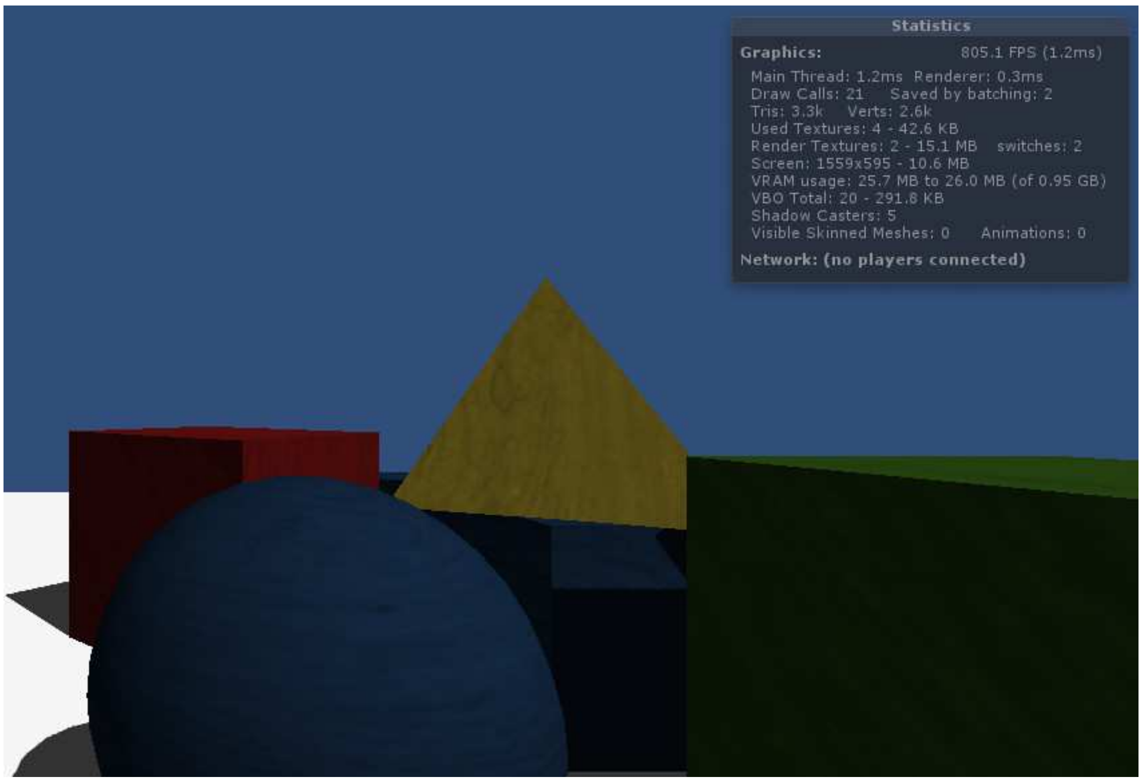
- ***Parent Object**: Here you must set the root of all the objects you want to batch. Before we have explained how to get all the objects we want to batch under a single root object.*
- ***Material**: The material that will be used for the merged mesh. Not that the meshes materials will be substituted by this, no matter whether they are the same (which they should for this to work well) or not.*
- ***Force Copy Material Color**: If this is set to true then the colour of the different materials in the objects will be copied to the vertices of the batched mesh. This is useful to lump together objects that use very similar materials that only differ in the colour the use to tint the material. For instance, you may have a bunch of wooden coloured toys that you want to lump together. They all share a wooden texture in greys and they just use the colour of the material to change the appearance (to look like they are red or blue). We'd create a new material that uses vertex*

colours (for instance the DiffuseVC shader that comes with this package) and will give it a white colour. Then if we mark this option the colour of the different objects will be copied to the appropriate vertices in the batched mesh, giving the same final result but with a single material.

We'll go over a brief example now.

Example

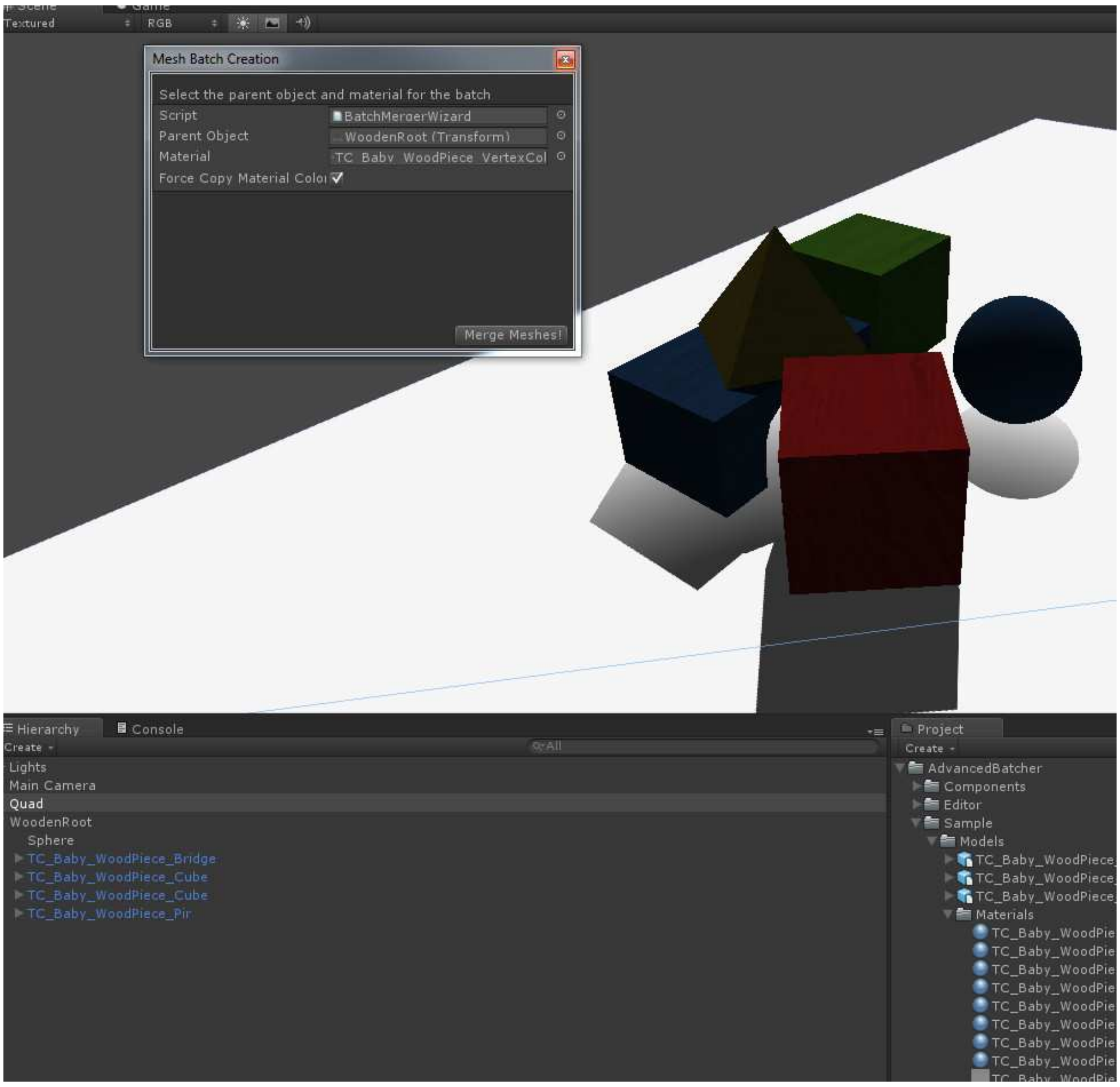
First you can see the scene that contains 5 different objects with different materials that use the same texture but use the main colour to tint the different objects. So first we'll see how many draw calls we have with Unity batching:



```
Statistics
Graphics:                      805.1 FPS (1.2ms)
  Main Thread: 1.2ms  Renderer: 0.3ms
  Draw Calls: 21    Saved by batching: 2
  Tris: 3.3k    Verts: 2.6k
  Used Textures: 4 - 42.6 KB
  Render Textures: 2 - 15.1 MB    switches: 2
  Screen: 1559x595 - 10.6 MB
  VRAM usage: 25.7 MB to 26.0 MB (of 0.95 GB)
  VBO Total: 20 - 291.8 KB
  Shadow Casters: 5
  Visible Skinned Meshes: 0      Animations: 0
Network: (no players connected)
```

As you can see we have 21 draw calls for the objects. That's because the shadow mapping is generating extra draw calls for every object that casts shadows. In the same dialog it says we're saving 2 draw calls thanks to batching. That's a 10% savings, which is not bad considering how this scene is not particularly a good fit for Unity's batching system.

So next we select all these objects move them under a common root object that's empty that we'll call WoodenRoot and use the simple offline batcher tool. We'll use the TC_Baby_WoodPiece_VertexCol material as the common material, since that material supports vertex colouring which will come in handy in this example.



Mesh Batch Creation

Select the parent object and material for the batch

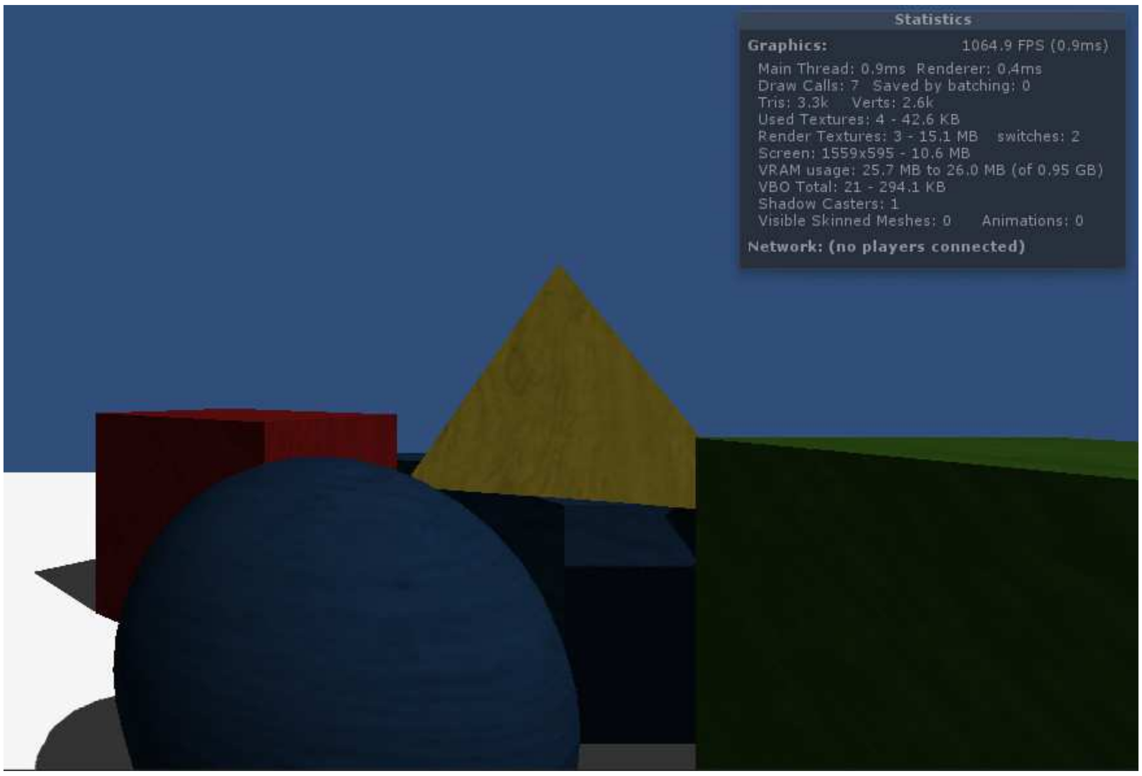
Script	BatchMergerWizard
Parent Object	WoodenRoot (Transform)
Material	TC_Baby_WoodPiece_VertexCol
Force Copy Material Colo	✓

```
Lights
Main Camera
Quad
WoodenRoot
  Sphere
▶ TC_Baby_WoodPiece_Bridge
▶ TC_Baby_WoodPiece_Cube
▶ TC_Baby_WoodPiece_Cube
▶ TC_Baby_WoodPiece_Pir
```

```
▼ AdvancedBatcher
  ▶ Components
  ▶ Editor
  ▼ Sample
    ▼ Models
      ▶ TC_Baby_WoodPiece_
      ▶ TC_Baby_WoodPiece_
      ▶ TC_Baby_WoodPiece_
      ▼ Materials
          TC_Baby_WoodPie
          TC_Baby_WoodPie
          TC_Baby_WoodPie
          TC_Baby_WoodPie
          TC_Baby_WoodPie
          TC_Baby_WoodPie
          TC_Baby_WoodPie
          TC_Baby_WoodPie
          TC_Baby_WoodPie
```

So after using the tool we can see that the root object, WoodenRoot, has now MeshFilter and MeshRenderer components. If we have a folder named Batches in the scene it will create a subfolder with the scene name and will create a mesh for the object there.

After using the tool we have only one MeshRenderer instead of 5. The new mesh has 637 vertices and 832 triangles. It uses a single material and we have reduced the number of draw calls to 7. That's obviously a huge improvement over the normal dynamic/static batch in Unity.



```
Statistics
Graphics:                          1064.9 FPS (0.9ms)
  Main Thread: 0.9ms  Renderer: 0.4ms
  Draw Calls: 7   Saved by batching: 0
  Tris: 3.3k     Verts: 2.6k
  Used Textures: 4 - 42.6 KB
  Render Textures: 3 - 15.1 MB    switches: 2
  Screen: 1559x595 - 10.6 MB
  VRAM usage: 25.7 MB to 26.0 MB (of 0.95 GB)
  VBO Total: 21 - 294.1 KB
  Shadow Casters: 1
  Visible Skinned Meshes: 0      Animations: 0
Network: (no players connected)
```

You can find the tutorial developing this example in Youtube in the following link:

http://youtu.be/7cvZ7M7fP9k

Advanced Offline Batching

This is the main tool of the plug-in. This is a comprehensive tool meant to simplify the work of batching together stuff by doing most of the things explained in the Motivation chapter. This tool is meant for objects that are static or at least won't move in relation to each other.

Again this tool requires you to organize all the objects you want to batch under a single root. The tool will batch together all the MeshRenderers under the given root. Then you click on the menu item for this tool: GameObject ->Batching->Advanced Offline Batching and a dialog box appears with a bunch of options:

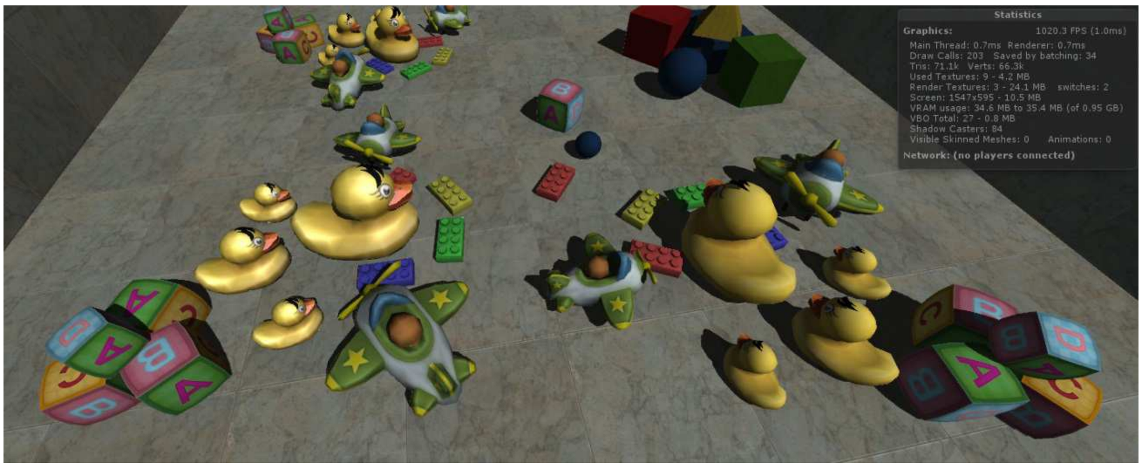
- ***Root Objects to Merge**: This is similar to the* Parent Object *parameter in the Simple option. It's the root object of all the meshes you want to batch together. Remember that now these objects don't need to share material, texture, shader, or anything, the tool will take care of organize them appropriately and lump together textures and materials as appropriate.*
- ***Root Objects Merged**: You need to create an empty object that will be the place where the tool will place the merged batches. It's advised that this object is not a child of the Root Object since that would make it hard to unhide the objects later on.*
- ***Folder to Put Generated Objects**: This is the folder in the project where the tool will place the merged textures, materials and all the batched geometry. It will delete previous objects in there so if you're doing 2 or more batches for the same scene it's recommended you change this folder and use one for each different batch. The default one is Batches/#NameOfScene#/. 3 folders will be generated inside that directory: textures, materials and meshes.*

- ***Max Polys Per Batch****: This is the maximum number of triangles or vertices the tool will allow per batch. If an object has more than these number of triangles or vertices that object will not be lumped with others, but it may still be changed to use merged materials. It's recommended that you experiment with this value for the platform you're targeting. So for instance, in mobile platforms a value lower than 4096 may give better results, while in PC you may get your best results with bigger numbers.*
- ***Max Objects Per Batch****: This is the maximum number of objects that will be allowed to batch together in a single mesh. If this value is 0 or negative then no limit is set to the number of objects allowed per batch. It is recommended to leave this value to 0 unless you have a good reason not to.* **NOTE: This feature is not working at the moment.**
- ***Max Distance Between Objects****: When batching objects they'll be considered for batching the closest first. This is done so to try to keep batches as compact as possible and make life easier to the render engine and allow him to cull objects more easily. If your batch objects are all over the scene it's more likely you'll see one of them and thus have to paint all of them. Also the culling is a lot easier if you pack them and the AABB containing all of them is reduced. The batcher will lump the closest objects first but you can define a maximum distance to stop merging together meshes. This may be useful depending on how your scene is built to avoid objects from one room merging with objects from another, etc.*
- ***Max Atlas Size****: This is the maximum size a texture atlas may have. We call texture atlas to the textures that contain all the different textures that we want to lump together. You want to define a maximum size for these atlases so that they don't grow too big. Growing too big has 2 disadvantages: first it's tedius to work with big textures and, particularly atlases, tend to waste a lot of texture space with empty holes. Secondly, some machines may find it quicker to render 2 textures of 1024x1024 than 1 of 2048x2048. It depends on a number of factors but mostly to the size and speed of the GPU memory. As a rule of thumb GPUs with lots of fast memory can handle big textures a lot quicker than those that don't have them, as is normally the case with mobile cards. This value should always be a power of 2.*
- ***Don't Merge Materials****: If this option is checked the batch process won't attempt to merge materials (and thus it won't merge textures either). It will only try to merge objects together when they have exactly the same material. This is very useful when you have materials or objects that can't be merged with other materials but you still want to batch the objects. In the last part of the documentation we'll see a few cases where this is useful, but for instance, having tiling in the objects makes the material unable to merge with others and need to be batched separately.*
- ***Don't Copy Color to Vertices****: This option is the opposite to* Force Copy Material Color *option in the simple batching tool. The batching process will copy the material color to the vertices of the batched meshes and will reset the colour to white unless this option is ticked.*
- ***Recalculate Normals****: Some meshes have some issues when merged together. If you set this to true the batcher will recalculate the normals of all the batched meshes after batching. If you're experiencing some funny illumination glitches with the batched mesh it's worth testing if checking this option fixes them.*
- ***Export Objs****: This option is used to export .obj files along with the normal asset files for the batched meshes. These are exported along with the asset files not substituting them because the meshes found in the obj files have some limitations (no colours per vertex and only one set of UVs per vertex for instance). On the other hand, they come in handy when using lightmaps. We're currently working on supporting .fbx exporting to overcome these limitations.*

There are quite a few options, but most of them are pretty intuitive and easy to understand. They empower the user a lot with just a few parameters. To illustrate best how to use this we'll explain how to batch a couple of parts of the example that comes with the plug-in.

Example

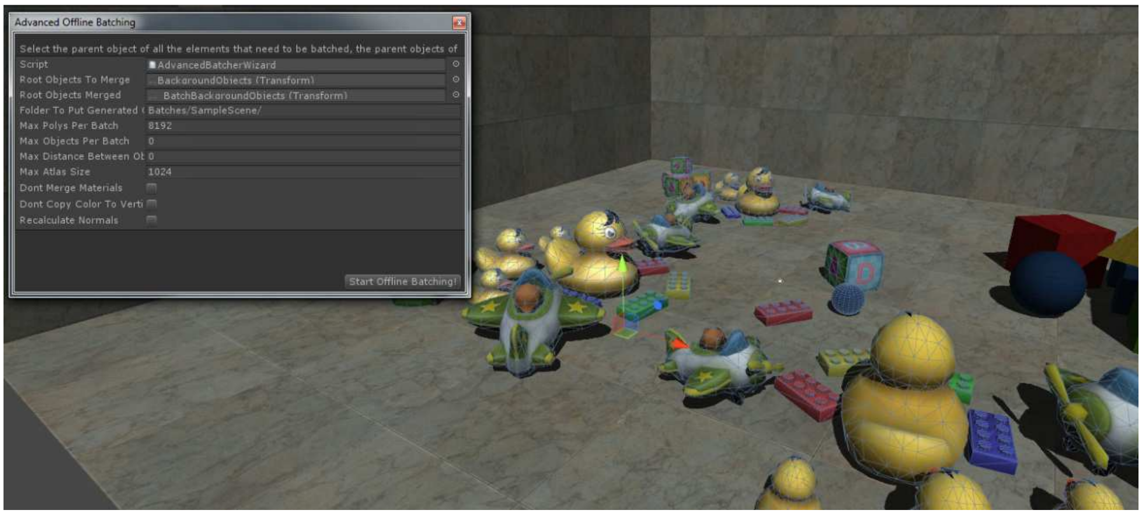
We're going to use the sample scene that comes with the plug-in. We're going to leave the wooden cubes alone since those are there for the Simple Offline Batching example, so we'll focus on the rest of the scene.



```
Statistics
Graphics:                          1020.3 FPS (1.0ms)
  Main Thread: 0.7ms  Renderer: 0.7ms
  Draw Calls: 203   Saved by batching: 34
  Tris: 71.1k   Verts: 66.3k
  Used Textures: 9 - 4.2 MB
  Render Textures: 3 - 24.1 MB    switches: 2
  Screen: 1547x595 - 10.5 MB
  VRAM usage: 34.6 MB to 35.4 MB (of 0.95 GB)
  VBO Total: 27 - 0.8 MB
  Shadow Casters: 84
  Visible Skinned Meshes: 0      Animations: 0
Network: (no players connected)
```

In the scene before batching we can see Unity is using 203 draw calls, after saving 34 from batching.

We start by batching the stuff under BackgroundObjects. *In order to do that we will start opening the Advanced Offline Batching tool and filling in the parameters. We will use 8192 (8*1024) as the maximum number of polys, and will set the target root to be* _BatchBackgroundObjects. *You can see the full list of parameters we use in the following image:*



Advanced Offline Batching

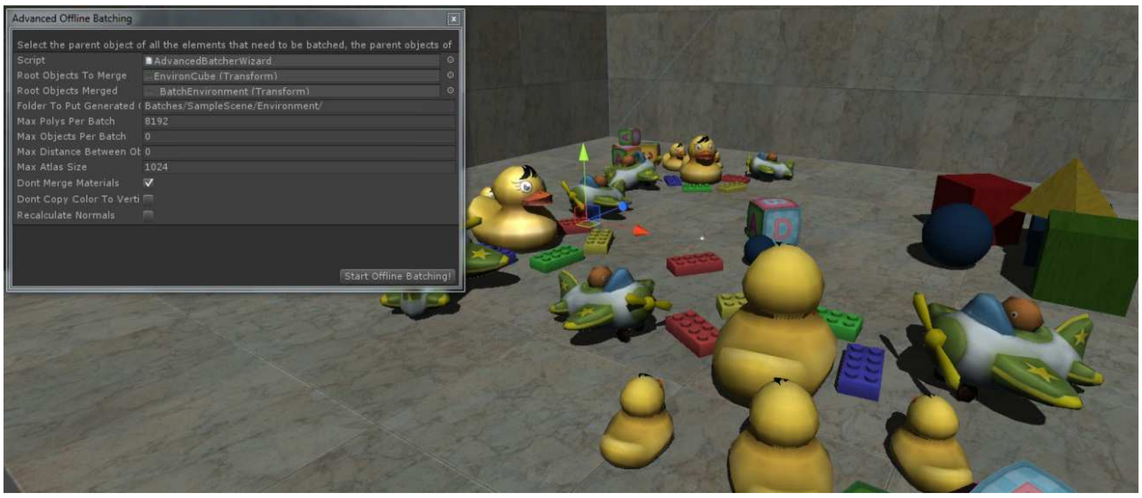
Select the parent object of all the elements that need to be batched, the parent objects of

Script	AdvancedBatcherWizard
Root Objects To Merge	BackgroundObjects (Transform)
Root Objects Merged	BatchBackgroundObjects (Transform)
Folder To Put Generated (	Batches/SampleScene/
Max Polys Per Batch	8192
Max Objects Per Batch	0
Max Distance Between Ob	0
Max Atlas Size	1024
Dont Merge Materials	☐
Dont Copy Color To Verti	☐
Recalculate Normals	☐

We click the "Start Offline Batching!" button and after a few seconds of computation we have the new batched scene. We have now 3 new meshes: 2 using a diffuse material with a texture that has merged all the texture from all the objects, and 1 using a specular material (the shiny ducks).

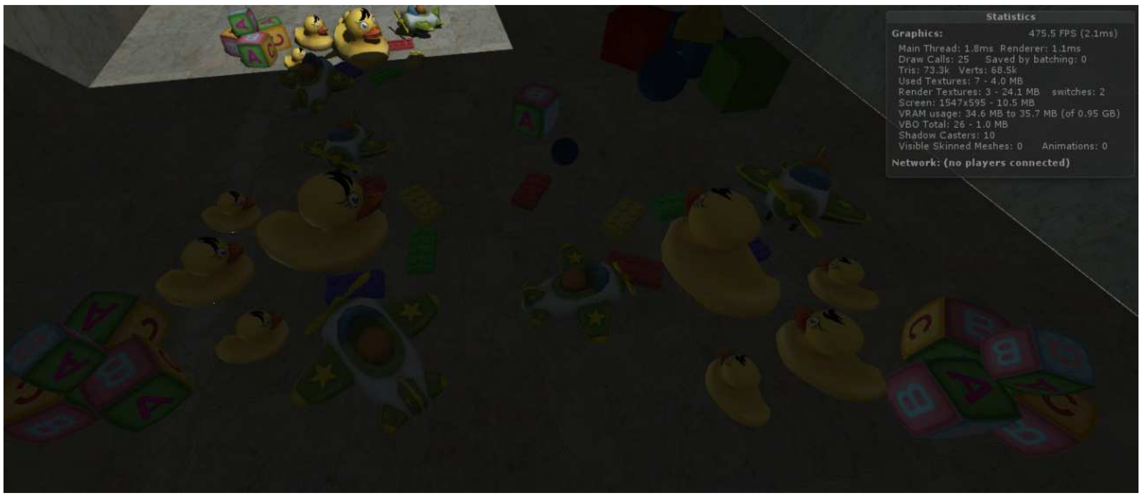
We have already gained a lot in terms of draw calls, but we can still do a little bit better. The 'environment' (the ground and the walls) can be batched too. So we open the Advanced Offline Batching tool and this time we set the root object to be EnvironCube *and the root objects merged to be*

_BatchEnvironment. *Since the ground and the walls have a tiled texture (the texture has a scale bigger than 1 in the material) we can't use a normal batching process, instead we must tell the batcher not to merge materials, so we check the* Don't Merge Materials *option.*



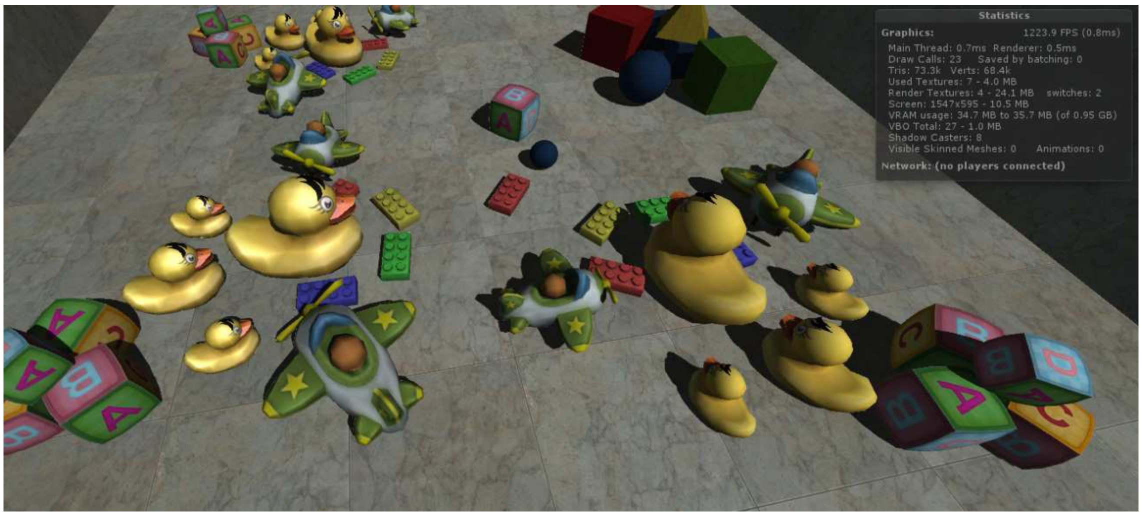
```
Advanced Offline Batching
Select the parent object of all the elements that need to be batched, the parent objects of
Script                          AdvancedBatcherWizard
Root Objects To Merge           EnvironCube (Transform)
Root Objects Merged             BatchEnvironment (Transform)
Folder To Put Generated (       Batches/SampleScene/Environment/
Max Polys Per Batch             8192
Max Objects Per Batch           0
Max Distance Between Ot         0
Max Atlas Size                  1024
Dont Merge Materials            ✓
Dont Copy Color To Verti
Recalculate Normals
                                                Start Offline Batching!
```

After the batching we see that all the walls and the ground have been merged in a single object that still has the same material as the old ground and walls. But wait! Something odd is happening. The walls are now casting shadows while they weren't before. The following image shows that:



```
Statistics
Graphics:                         475.5 FPS (2.1ms)
  Main Thread: 1.8ms  Renderer: 1.1ms
  Draw Calls: 25     Saved by batching: 0
  Tris: 73.3k   Verts: 68.5k
  Used Textures: 7 - 4.0 MB
  Render Textures: 3 - 24.1 MB   switches: 2
  Screen: 1547x595 - 10.5 MB
  VRAM usage: 34.6 MB to 35.7 MB (of 0.95 GB)
  VBO Total: 26 - 1.0 MB
  Shadow Casters: 10
  Visible Skinned Meshes: 0     Animations: 0
Network: (no players connected)
```

*This happens because the new batched object is set, by default, to cast and receive shadows. Our ground and walls were set to receive shadows but not to cast them. So, to fix that, we need to go to the generated batch object (*_BatchEnvironment->BatchContainer0->Batch000*) and remove the* Cast Shadows *option from the MeshRenderer. The result will be the final one:*


```
Statistics
Graphics:                    1223.9 FPS (0.8ms)
  Main Thread: 0.7ms  Renderer: 0.5ms
  Draw Calls: 23     Saved by batching: 0
  Tris: 73.3k   Verts: 68.4k
  Used Textures: 7 - 4.0 MB
  Render Textures: 4 - 24.1 MB    switches: 2
  Screen: 1547x595 - 10.5 MB
  VRAM usage: 34.7 MB to 35.7 MB (of 0.95 GB)
  VBO Total: 27 - 1.0 MB
  Shadow Casters: 8
  Visible Skinned Meshes: 0     Animations: 0
Network: (no players connected)
```

We can see that the final image has only 23 draw calls, as opposed to the initial 203 (and that was after Unity's batching saving 34 draw calls). That is, we saved 180 draw calls, which is almost a 90% of the draw calls!

Obviously the amount of draw calls saved in different scenes will vary and will be lower or higher depending on the size of the objects and the ability to lump them together (whether they share shaders or not). The study case is particularly good, but is not even a corner case, you can find scenes better suited to batching where results could go even over the 90% of draw calls saved. Unfortunately there's no easy way to know how many draw calls you'll save batching, but chances are you'll likely get good results.

A tutorial video for this example can be found in the link below:

http://youtu.be/egJnM0p9iM8

Frequently Asked Questions

Are there any restrictions as to which objects/materials/textures I can batch?

Yes, there are several restrictions that you have to take into consideration when batching objects:

- *You can't batch normally objects that are tiling the textures they use. So, for instance, if you object is using UVs that are out of the range [0..1] then you're going to have problems batching them.*
- *You can't batch normally objects that use materials that use* Tiling *in their textures. It's the same problem as before.*
- *Not every material is going to work well moving the material colour to the vertices. Diffuse and Specular will work fine, other materials may cause troubles.*
- *You can't batch dynamic objects or objects that will move respect to others.*

We'll see how to work around these issues now.

How can I batch objects that have tiling in their UVs or the material?

Tiling is an issue for merging materials, but not so much for batching objects. When you merge 2 textures and thus their materials, when you're generating the merged batch you need to adapt their UVs to the new texture atlas. Since the textures can't now be repeated changing the UVs a UV of 1.5 (which before would be equivalent to 0.5) now is out of the texture. In order to batch this objects this step needs to be removed. That on the other hand is very easy.

You need to check the option Don't Merge Materials *in the Advanced Offline Batching tool and run the batching. It will batch objects but will not merge materials or textures, leaving materials as they are and not changing UVs in the vertices either.*

Which materials support moving their material colour to the vertices?

Right now only Diffuse and Specular are supported. We have plans to add all the basic materials that make sense here. Anyway, if you can write your own shaders to support that you can change the shader in the resulting material easily since the materials are created in a concrete folder in the project.

Anyway if you're trying to use this feature and the shaders you're using come with Unity and are not supported yet, please send us an e-mail to let us know. We'll prioritize them and will try to release an update ASAP with support for these shaders.

Are you going to support dynamic batching?

We have plans to support dynamic batching, that is, batching objects that can later on move. We don't have an ETA for that feature yet.

Do you support batching skinned meshes?

No, that's not supported at the moment either. Again, we have plans to support that feature in the future too, although I don't think that's usually as useful as static mesh batching or even dynamic mesh batching.

What are the maximum number of polys per batch recommended per platform?

We can provide concrete numbers. It depends a lot on the platform. As a rule of thumb, old mobile platforms will work better with smaller vertex buffers (1-2K) while newer ones may work fine with 4-8K. In PC again it depends a lot on the graphics card, but modern graphics can definitely handle big vertex *buffers. Our recommendation is that you try different sizes and profile the results and use the ones that work best for your target platform.*

Why have my objects changed the shading after the batching?

Depending on the way the objects are imported in Unity, the batcher has issues importing and translating the normal of the objects. We're not sure exactly what causes it and we're currently investigating the issue to try and find a better solution.

Anyway, for these objects you can try to batch them separately and check the option to Recalculate Normals *which most of the time helps with this issue.*

Why do you export to .obj? When are you going to support .fbx exportation?

The code to export to .obj files was already available for Unity. It has some limitations such as no colours or only one set of UVs per vertex. We're already working on supporting .fbx since it might be a better fit for our needs. Still we thought that giving the option now was useful in case you're using lightmaps.

Why do you export .obj files alongside the .asset meshes, instead of replacing them?

As mentioned before, the .obj files have some limitations, we thought it's better to give more options rather than limiting the user.

When I export the .obj files won't appear at first and only when I leave and go back to Unity they appear in the right folder?

Yes, that's annoying isn't it? We don't know why is that happening, but we're investigating that issue.

Is there a pro version with access to source code?

Yes there's a Pro version available upon request. If you're interested in acquiring it, please send us an e-mail at:

unity@eclipse-games.net

Contact Info

I've tried to be as through as possible with this documentation. Still I understand it may still be lacking in some aspects and some things may have been overviewed or completely missed. I'd encourage the reader to send me an e-mail with any question, doubt, suggestion or correction they may have both for the tool or the documentation. Please, use e-mail me your comments at:

unity@eclipse-games.net

Thanks!

Eduardo Jiménez

Technical Director at Eclipse Games