Tips on preparing a Unity3D project for WebGL
(and someday better VR performance)

1. **Downsample most textures at 256 or 512.** If you store them all in a project directory, it’s easy to select all files at once and change settings across them. (splash graphics can be 1024 for clarity).

   + Also, **turn off mipmaps for any UI texture**! (they have no need of multiple detail levels because they are never far away)

   + Note that all your **textures really should be saved as Power of 2 sizes** (Unity will convert for you, but there are known issues with this bloating smart phone builds. Basically saves separate copy of each of your improperly sized texture).

   - If you have odd shaped textures (like tail feathers): **store more than one on a shared Po2 texture.** this will lower Draw Calls later.

2. **Switch large background images/sprites material type from using "Standard" shader to "Unlit > Transparent Cutout" shader** (basically makes them self illuminating, and ignores a lot of lighting)

3. **Turn down the Project’s "Graphics" settings to "Simple"(at least! Fastest is best for WebGL... but when i went down to "Faster" settings, some complicated water texture stuff started randomly vanishing in browser).**

4. **Look at performance Profiler window (Click in top bar to get sample slice).**

   - note that Profiler is showing you stats based on you Build target (so make sure to Switch Platform in your build settings, to WebGL).

   - Check **Memory:Unity** to see how bad it gets, then make sure WebGL’s alloc is bigger. (the other non-Unity memory usage stats are things that may be offloaded to graphics and sound card. even by web browser. So not that important).

   - Also check **Rendering:Draw Calls.** less than 1000 is ideal. more than 2000 is bad sign on current PCs. FramesPerSecond is king, but so closely tied to draw calls that you tend to just watch draw calls (you could conceivably have 4000 draw calls on PC but still be getting 120 fps. While on mobile, more than 100 draw calls likely means less than 30 fps). having multiple cameras active in scene will multiply your draw calls quickly (everthing is being drawn twice).

   - finally, the main **CPU** bar is used to drill down an see what is bottle necking. but i’m not clear on how to best use this info.

   ... You can also turn on the "**Stats**" in the Game window to overlay important deets while testing.

   ... HotKeys: **shift+cntrl+p** to pause game playback, so you can drag around in profiler (and see why something just spiked).

   - **... you can also setup development builds that will connect back into unity’s profiler, but i haven't done this yet.**

5. **Strongly suggest saving the audio in mono format to halve the file size.** (it ends up sourced in 3D anyway)
6. Bake lights. Should take 5 to 20 minutes on our machines (if longer, cancel and fix your problems). Each light has a Baking type in the Inspector, which controls if it's involved in baking step. turn off "auto" baking in the Lighting window.

Realtime lighting is costly, so only use it for things that are moving. Terrain trees and plants may sway, but it's still a good idea to bake their shadows.

- note that around Unity5.2 they switched the behind-the-scenes engine for baking lights (from Beast to Enlighten). Enlighten will lock up on the "7/11 Light Transport" jobs if you have any plane meshes that are larger than about 16 units in any direction (so check your scaling. or use quads).

- So i hid all the big planes before baking lightmap, then turned them back on after.

- There is currently (5.4) a known issue where Enlighten won't bake tree shadows from terrain. (need to see if we can bring in old bakes through photoshop)

7. Note that changing the Scale after import affects more than physics (the physics engine really likes things to be scale 1). Scale can also affect texture (aliasing) and lighting (shadows/bumps). I'm just not fluent in best practices here. unity treats 1 unit as 1 meter, so it's good to shoot for that from the start in Maya (where 1 unit is 1 cm). I tend to create 100 unit reference box in Maya and invert it's normals so i have some idea where i'll end up in Unity.

8. Currently, you need to copy a .htaccess file to the WebGL's "Release" subdirectory to overcome a setting on our webserver. (you'll get an error that it doesn't understand the compression type).

This is contents of that file:

```
Options +FollowSymLinks
RewriteEngine on
RewriteCond %{HTTP:Accept-encoding} gzip
RewriteCond %{REQUEST_FILENAME}gz -f
RewriteRule ^(.*)\.js$ $1\.jsgz [L]
RewriteCond %{HTTP:Accept-encoding} gzip
RewriteCond %{REQUEST_FILENAME}gz -f
RewriteRule ^(.*)\.data$ $1\.datagz [L]
RewriteCond %{HTTP:Accept-encoding} gzip
RewriteCond %{REQUEST_FILENAME}gz -f
RewriteRule ^(.*)\.mem$ $1\.memgz [L]
RewriteCond %{HTTP:Accept-encoding} gzip
RewriteCond %{REQUEST_FILENAME}gz -f
RewriteRule ^(.*)\.unity3d$ $1\.unity3dgz [L]
AddEncoding gzip .jsgz
AddEncoding gzip .datagz
AddEncoding gzip .memgz
AddEncoding gzip .unity3dgz
```

9. WebGL can't handle movie textures. So I just commented out most of Cinematic.cs stuff. Will ask mark for assist in making a js func we can trigger (external call) to play movie in iframe over the Unity WebGL player (and close when done, hopefully). We did something like this for PermaMOOC.

10. not sure how much these mattered or not:
- changed terrain's "Lighting [object]: Scale in Lightmap" down from 1 to "0.1". (Unity still notes that it was broken into 4 chunks. and "Object's size in lightmap has reached the max atlas size")

- For all the giant Curved Planes (mountain ranges) and the huge ground plane (necessary?), I went into the Inspector and switched "Blend Probes" to "off" for lightProbes and ReflectionProbes. Also unchecked "Motion Vectors." No idea if this helped at all.