WebGL Retina Tools

Hogbox Studios

Add Retina/HDPI resolution support to your WebGL builds with the click of a button. Eliminates the blurry textures and text seen on Macs running Unity WebGL builds.

Retina Demo
Non Retina Demo

Quick Start

Extract the provided hbxWebGLTemplates.unitypackage file. Go to player settings and select one of the hbx templates as your WebGL template, build your project for WebGL, either release or development, then go to menu item:

```
Hbx>WebGL Tools>Retina Fix Last Build
```

This will apply the fix to the last WebGL build you created with Unity. Alternatively you can apply the fix to an existing build by going to menu item:

```
Hbx>WebGL Tools>Retina Fix Existing Build
```

Then selecting an existing WebGL build folder (the folder containing index.html and the Release/Development folders). However the html files and css may need altering for an existing build (see next page).

Goodbye blur :)

WebGL Templates

Performing the above will fix your compiled WebGL scripts, but you'll most likely also need to alter any custom WebGL template you have in order for the canvas to display correctly.

A few different WebGL templates are provided to demonstrate how to setup your html/css. Including tweaked versions of Unity's default and minimal templates which hopefully users will be familiar with. To add these to your project open the provided unitypackage located at:

```
Assets/Hbx/WebGLRetinaTools/hbxWebGLTemplates.unitypackage
```

Once extracted you can set one of these as your WebGL template in the player settings, build and then apply the Retina fix in order to quickly test the results.

Fixing your existing template

On the whole your WebGL template can remain the same. The main difference is that your canvas size must be controlled by a css style. This is because we need to scale the canvas backing/buffer width and height by the windows devicePixelRatio. On Retina Macs this value is 2 so we end up with a canvas with a buffer twice the resolution, however this also causes the canvas to display at twice the size in the web browser and causes a loop of the canvas growing by a factor of two every frame. Not good.

To resolve this we need to use a css style on the canvas to force the canvas to display at the size you desire, while retaining its increased backing size. Try to imagine it as the canvas width/height being the pixel resolution of the canvas and the css style width/height being the size it's displayed in the browser.

In Unity's provided Default WebGL template the dimensions of the canvas are fixed based on the values provided in your player settings. In this instance the fix to the templates index.html is simple. We just add a css style setting the canvas dimensions to those same values. Like below.

```
<style>
        canvas { width: %UNITY_WIDTH%px; height: %UNITY_HEIGHT%px; }
</style>
```

Notice that as this is a template we use Unity's %UNITY_WIDTH% and %UNITY_HEIGHT% variables that will get replaced when you build, but in order for this to happen the css code must be in the index.html file (as far as I'm aware). So be sure to check your style.css file for any css styles that might try and override this one.

Fullscreen mode

You'll notice that if you set the css to a fixed value, when you go into fullscreen mode the canvas remains the same size in the browser. This is because the css is preventing the canvas expanding. To fix this we need to add some javascript to the index.html file, this script will switch the css style width/height when toggling in and out of fullscreen mode.

```
<script type="text/javascript">
   var windowedWidth = %UNITY_WIDTH%;
   var windowedHeight = %UNITY_HEIGHT%;
   var canvas = document.getElementById("canvas");

   document.addEventListener("fullscreenchange", onFullScreenChange, false);
   document.addEventListener("webkitfullscreenchange", onFullScreenChange, false);
   document.addEventListener("mozfullscreenchange", onFullScreenChange, false);

   function onFullScreenChange() {
      var fullscreenElement = document.fullscreenElement ||
                                    document.mozFullScreenElement ||
                                    document.webkitFullscreenElement;
      if(fullscreenElement != null)
      {
         canvas.style.width = screen.width + "px";
         canvas.style.height = screen.height + "px";
      } else {
         canvas.style.width = windowedWidth + "px";
         canvas.style.height = windowedHeight + "px";
      }
   };
</script>
%UNITY_WEBGL_LOADER_GLUE%
```

The above script needs to be added after the canvas is defined in the index.html file. It's pretty basic and only really intended for demonstration purposes. It uses the same %UNITY_WIDTH/HEIGHT% variables to store the windowed dimensions. Next it registers for fullscreen change events. When entering fullscreen mode the style width/height is set to the screens width/height. When exiting fullscreen mode the style width/height is switched back to the stored width/height.

Keep in mind this approach is only required if you are using a fixed width and height when not in fullscreen mode. If you were to set the canvas css style width/height to 100% then the canvas would automatically fill the screen when in fullscreen mode.

Important Notes:

- You **Must** have a css style on the canvas controlling it's width and height. If you don't have one the canvas will keep growing.
- Due to the above fullscreen mode also requires additional js code to work. The canvas css style must be toggled when switching to and from fullscreen mode. Examples of this are shown in the provided WebGLTemplates.
- If you need your view to resize as the browser window resizes enable "Run in Background" in the player settings. Otherwise the view dimensions will not update until you refocus the view.
- This script depends on find and replace so is only guaranteed to work with the version of Unity it was developed against (5.4.1), although previous development was done in 5.3 and was working. Hopefully Unity will eventually support this out of the box.
- This script will **Not** add Retina support to standalone desktop builds, only WebGL.
- You need to reapply the fix each time you create a new build.