Mobile optimization guide
for VR games made in Unity

by Sanna Ekneling
2018

Chapter 1.Guidelines

According to the oculus site, these are the guidelines for Gear VR and other mobile VR games. But should be applied to all kinds of mobile games to stay efficient:

These guidelines are per Unity scene, remember that scene changes can be heavy. This problem is tackled in *chapter 2.*

FPS: 60

Keep it steady. Some background applications use the GPU memory so try not to push the limit, it's better to make a game that could run in 80 fps on a new phone but clamp it to 60 fps so that other applications don't affect performance of your app.

Draw calls: 50 - 100

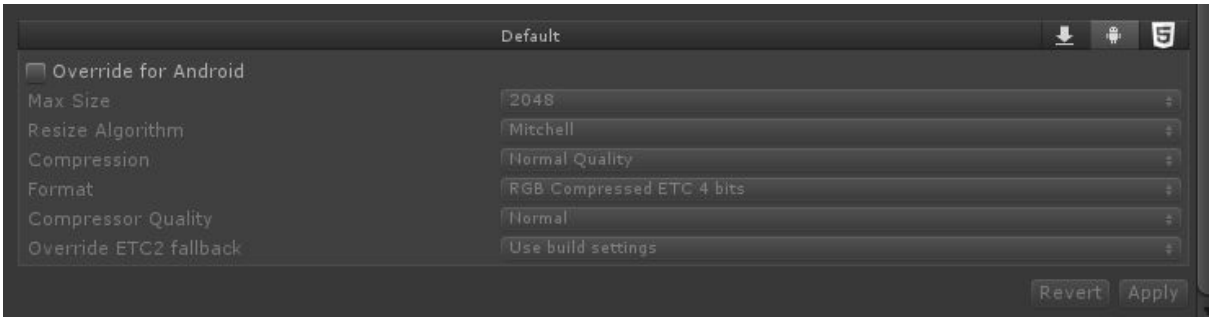
A simple way decrease draw calls is to minimize transparent objects overlapping. Keep transparency to a minimum when possible. Another tip that requires a bit more work but is proven very efficient: Instead of having a world with 3D objects, consider taking a 3D picture or video of the environment and use it as a texture on a sphere with flipped normals. Tutorial on this in chapter 3.

Triangles and Vertices: 50 000 - 100 000

If you want complex scenes, opt for taking a 360 image instead of using the models in the actual games. Tutorial in chapter 3.

Use **16 bit depth buffer resolution** and **2x MSAA.**

With the right compression there is little to none difference between 16 bit and 32 bit color buffer. This is basically how many colors that are used. The presets for textures are 4 bit. To change this, go to the texture and check the box for "Override for Android" (or which platform that is used). The max size depends on what the texture is used for. If it is for the 360 picture, consider a value of minimum 2048 to avoid more pixelation than the Gear VR already have. Change the format to what best fits your use.



Unity has a table for default formats, but these can, and should, be changed when working with VR. Try using 16 bits if possible for backgrounds.

Default formats in unity:

Platform	Color model	None	Normal quality (Default)	High quality	Low quality (higher performance)
Android	RGB	RGB 24 bit	RGB Compressed ETC2	RGB Compressed ETC2	RGB Compressed ETC2
	RGBA	RGB A 32 bit	RGBA Compressed ETC2	RGBA Compressed ETC2	RGBA Compressed ETC2
iOS	RGB	RGB 24 bit	RGB Compressed PVRTC 4 bits	RGB Compressed PVRTC 4 bits	RGB Compressed PVRTC 2 bits
	RGBA	RGB A 32 bit	RGBA Compressed PVRTC 4 bits	RGBA Compressed PVRTC 4 bits	RGBA Compressed PVRTC 2 bits

Use the same compression formats for Android and iOS but with the color depth that you need.
WARNING: *This is very performance heavy on the computer and could take several hours per texture so only do this on textures that need a larger color depth, such as 360 environment pictures.*
Texture memory allocation: 128 MB
Phones have a lot of RAM and pretty good GPU and CPU power which makes it possible to make awesome games even with the previous guidelines. Android phones can usually handle 128 MB of textures in one scene without it causing any trouble.
AudioSources: 16
Keep the audiosources to a downlow and consider using pooled audiosources instead of PlayOneShots.

Summary

FPS	60
Draw calls	50-100
Color depth	< 16 bit
Triangles	50 - 100 k
Vertices	50 - 100 k

Texture memory	128 MB
AudioSources	< 16

References

Rendering Guide:
https://developer.oculus.com/documentation/mobilesdk/latest/concepts/mobile-rendering/

Texture Format standards
https://docs.unity3d.com/Manual/class-TextureImporterOverride.html

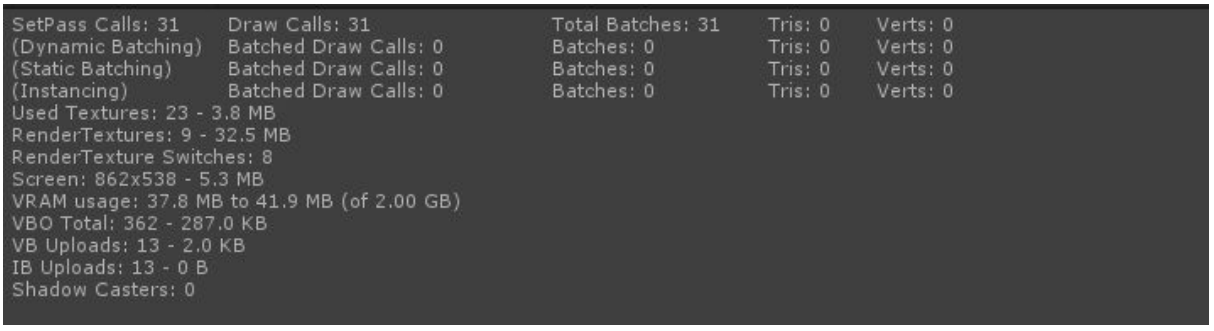
Performance guide:
https://developer.oculus.com/documentation/unity/latest/concepts/unity-mobile-performance-intro/#unity-mobile-performance-intro

Chapter 2. Profiling and crash solving on PC

The profiler shows a lot of information. This chapter will cover how to profile on the PC, what usually causes a game to be too heavy and how to solve it. The two most important ones if the game has a lot of lag or crashes are Rendering and Memory since these affect the GPU, RAM and could even affect the CPU even though this has its own tab.
IMPORTANT: Remember to profile all scenes and not just one. Keep track of different things.

Rendering

The tab of the profiler called "Rendering" is one of the most important ones. Here you can keep track of **draw calls, triangles and vertices** amongst other things.
The values you are looking for are found in *Chapter 1*.



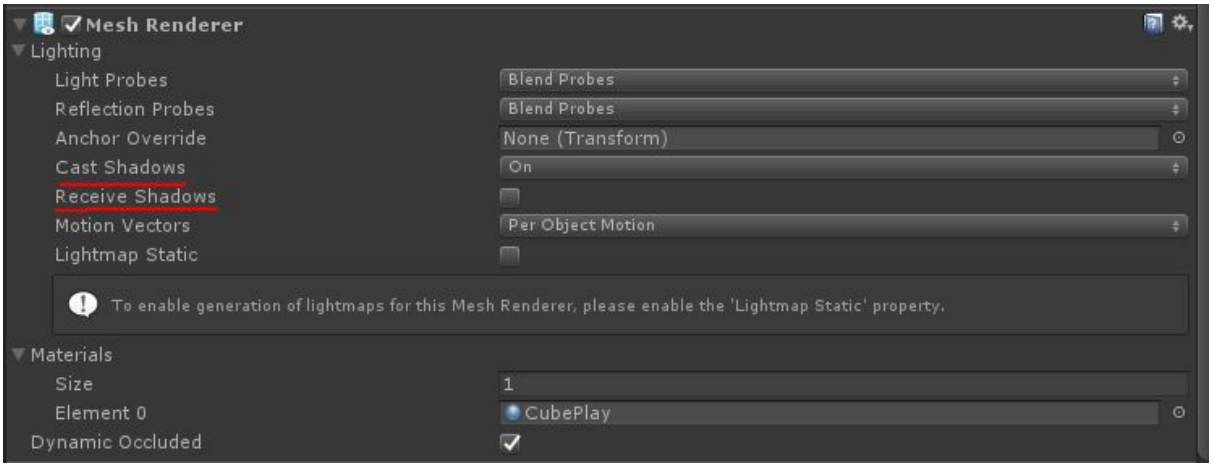
The view in the profiler in Unity 2017.1

Used Textures & RenderTextures

If your game crashes after a certain time, or the phone gets hot unusually fast (Samsung phones tend to get hot pretty fast in the Gear VR headset, this is not unusual), check if any of these numbers are higher than expected. If you know that your scene is using less textures or render textures than the number, consider deleting inactive scene objects that are never used and possibly even deleting RenderTexture assets if not used.

Shadow Casters

Shadows are heavy for phones. Consider taking away shadow casting and receiving from items that don't need it, either because they are so small that they won't cast shadows, or because it wouldn't make sense for shadow casting on certain items. This is done on the Mesh Renderer of each object or in code.



Memory

The memory tab in the profiler shows the total memory usage among other things. Keep this number as low as possible even though phones usually have around 6 GB of RAM these days. Most users don't clear background apps which means that the full 6 GB is not available for you to use. The more you stress the phone, the more battery your app will drain (not just RAM, but all hardware usage in the phone need power, so the lighter the app, the longer playtime and the cooler the phone).
Keep in mind that the profiler uses a bit of memory as well so you can count a few MB of the total memory usage.

Rising numbers

If any number in this tab keeps rising, check your code for creation of new Textures, Meshes, Materials or other assets during runtime. Try deleting these as they allocate memory and makes the game heavier and heavier which will result in a crash without any escape. if you create a new asset during runtime, keep in mind to use Destroy() when this or the previous thing you try to replace is being unused. If you are in need of doing this a lot, try using a object pooler instead.

Profiling on the phone

Follow this guide to debug the game on the phone, especially "Attaching to Unity players", "iOS" and "Android".
https://docs.unity3d.com/351/Documentation/Manual/Profiler.html
(possibly outdated)

For debugging an Android phone use this guide:
https://answers.unity.com/questions/492681/how-to-use-adb-logcat.html

Game crashes with good profiling values

Try turning of Graphics Emulation which restores your hardwares full capabilities. This is done by going to **Edit -> Graphics Emulation -> No Emulation.** For more information about what this means read these:

https://docs.unity3d.com/Manual/GraphicsEmulation.html

NOTE: Why this works is unclear since this is just an emulation of the hardware that the app will use in the editor. This also resets every time you change scene, build or close the editor. This is more of a last resort when nothing else works.

Summary

- Keep track of your triangles, vertices and draw calls.
- Don't use shadow casting where it is unnecessary.
- Don't use the full capacity of the RAM with your app, try keeping the memory usage as low as possible.
- Use object poolers instead of creating new objects.Or Destroy objects if they are not supposed to be used again.
- If nothing else works. Try using no emulation in the editor while you build.

Chapter 3. Making a texture of your scene

A great way to reduce draw calls and make beautiful scenes for VR and other mobile games is to take pictures or videos and use them as textures instead. This chapter goes through how to do this.

Tutorial

Step 1: Download a 360 recorder

To create 360 images or videos of your scene you need to either write your own script or download a package that lets you do this. This camera rig is easy to use and works great: *https://assetstore.unity.com/packages/tools/camera/360-vr-camera-capture-rig-53264*

NOTE: You will probably get an error the first time, or when changing values in the editor.

Step 2: Place camera

Add your camera, or place the camera rig with the same camera values as your camera in the scene. Camera effects that you want in the image or video should be added to the camera. Which numbers to use is under the header "Numbers or perfection".

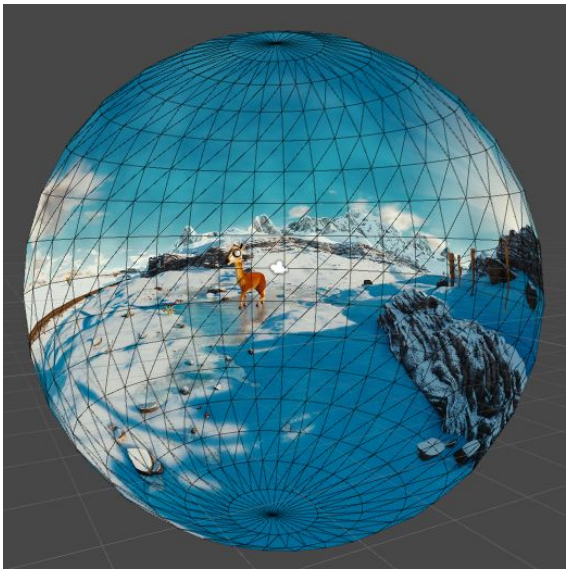
Step 3: Take picture or video

See "Video or Image" and "Video or Flipbook" for information on how to choose the right thing for your game.

When the camera is placed where you want it and the values are changed to your needs, press play and use the buttons from the LS360VRCamera script. These are O for Image and I for video. P is a 3D picture which basically creates two images of the same and is not used in this tutorial.

Step 4: Create a sphere

Use whichever 3D modelling program you prefer and create a sphere with flipped normals. If using Maya, you don't have to care about the UV mapping, and the number of polygons is up to you, you probably won't see any difference between 100 and 1000 polygons.

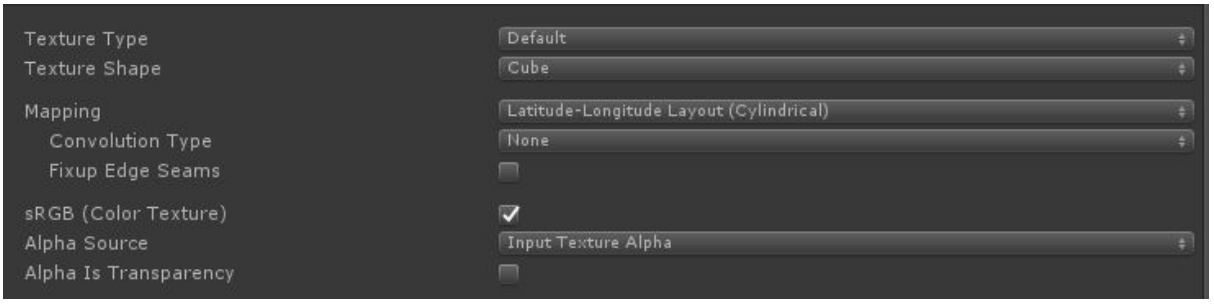


Step 5: Change texture

The images from the LS360VRCamera ends up in a folder at the same level as the Assets folder. If you make a video, these will be several images. Use a video editing program such as After Effects, Premiere Pro or even Movie Maker if you want it as a video. See "Video or Flipbook" to make your choice.

*If **Image**:*

Import the texture and change the Texture Shape to Cube with Mapping set to Latitude - Longitude Layout (which basically means a cylinder). This is better than 6-frames layout since this is more prone to leave a seam.



*If **Video**:*

-Will include this later but this is still a very heavy option-

Step 6: Make material

Create a new material in Unity with the shader Skybox/Cubemap

Step 7: Add material to sphere

Change the rotation to what you pleased. If your camera was at 0 rotation when capturing, the rotation should be 0 as well.

Video or Image

Basically, if you don't need any moving parts in your background: use an image, this will be much more effective than a video. If you want moving parts that should be affected by script: use an image. There is almost never a reason to use a movie texture except for when you have looping effects such as particle systems (which are difficult to get a perfect loop out of). or moving flags in the wind.

Video or Flipbook

If you do decide on a video, consider using a flipbook instead of a movie texture since it is easier on the system (see references) to change between different textures instead of having an actual video playing. Especially of that size.

Numbers for perfection

While taking images or videos of your scenes there are numbers to consider. Here is a list of them according to Gear VR which uses a FOV of 110 degrees and a resolution of 2560x1440 for the entire screen which is then divided into the two eye views.

FOV: 90

Funny enough, when capturing the 360 images the camera should be set to 90 in Field of View to be a perfect field of view in the VR headset. Even though the FOV of the headset is 110.

Width of panoramic: 6000 pixels

The width of the images should be 6000 pixels since each pixels cover about 0.06 degrees of the visual arc. The height should preferably be 1500 pixels.

Eye buffer size: 1024x1024

If the eye target size is not set to this it should be done manually.

Tip: *Use mipmaps to avoid aliasing if the eye buffer size if bigger than 1024.*
sRGB is preferred in high contrast scenes.

Real life example

In the game Magestro we went from around 130 draw calls to 24 by using the sphere method. We also went from around 70 k vertices to 1 k which made a huge difference in performance. Motion sickness happens a lot because of lag in VR so this is something to avoid at all costs. There is better to have lower resolutions than lag.

Summary

- Almost never use a video. An image is preferable even when this means you'll need a few models in the scene
- Change of textures in a material is lighter than using actual movie textures
- Use FOV 90 when taking the 360 pictures
- If possible make the image 6000 pixels wide

References

Movie Texture vs Flipbook (Unreal engine, but the argument applies to Unity as well)
https://answers.unrealengine.com/questions/132571/movie-texture-vs-flipbook-pros-and-cons.html

FOV, resolution and other Gear VR numbers
https://developer.oculus.com/documentation/mobilesdk/latest/concepts/mobile-rendering/