[Jonathan Lampel (2019)](https://cgcookie.com/tutorial/creating-3d-assets-for-virtual-reality)

<https://cgcookie.com/tutorial/creating-3d-assets-for-virtual-reality>

* Build to scale – 3 to 3.5 feet (around 1 meter) is a good height for things that should be grabbed, and 5 feet (roughly 1.5 meters) is a good height for things that should be at eye level.
* Pixels over polygons – 60 to 90 fps in order to prevent motion sickness, everything is rendered twice.
* Add more geometry or increase texture resolution
* Intersect small details instead of connecting them to the surface.
* Try to limit the amount of sharp edges or UV seams – smooth shading and fewer UV islands is better.
* Texture atlas – better for the game engine to one one really large texture than many smaller textures
* There’s an extra draw call per material, so the more objects that share the same materials, the less draw calls you will have
* Normal maps are good with fine details. If you really need the light to bounce off an edge, you’ll need to bevel the mesh itself. Try increasing edge wear in your textures
* Baked reflection probes only reflect static objects, only use them if you must.
* Real time reflection probes look best but extremely resource intensive. Good if you want to use mirrors
* Try to stick with diffuse and specular maps, only roughness when objects will really benefit from it.
* Alpha mapped textures (opacity) Some common culprits to be aware of are grass, foliage, trees, sci-fi computer screens, and particles. Don’t have too much.
* Be sure to make non moving objects as static in unity.
* Use LOD’s – levels of detail
* Affordances

[Christopher Chamberlain (2017)](https://www.blog.google/products/daydream/maximizing-art-assets-vr/)

<https://www.blog.google/products/daydream/maximizing-art-assets-vr/>

* Tiling textures: Instead of making one giant texture that covers lots of space, create a smaller tileable texture that can be used in more spaces.
* While keeping the poly count low is important, texture and pixel resolution is more valuable.
* Strive for higher resolution by making larger UV spaces to achieve a higher resolution per asset.
* Use geometry to make something unique, not textures - To make areas of your asset look more unique, use geometry to add a new colour splash or add a few edge loops to bend or scale a duplicated asset into a different silhouette.
* A good workflow in the tight constraints of VR is to build the individual parts of a final asset and then assemble them. Always be considering whether or not that next unique part is needed at the cost of an overall smaller UV layout, thus less lower resolution per asset.

Matthew Hales (2016)

<https://medium.com/@Hales/keeping-it-canny-asset-preparation-for-virtual-reality-659d46001976>

* Baking Geometry into Textures Not Nearly as Effective as it Used to be
* Items that will remain fairly distant — say, 10 meters or more — are seen by the left and right eye almost identically, with very little convergence. So, if you know that an object will remain distant, the old tricks still work.
* Parallax mapping can really deliver when you need the detail without the cost. Better than normal or bump
* Avoid planar construction or foliage and ground cover - plant life — especially that seen up close — should be structured more like their real-word counterparts when possible.
* When visible facets are part of the intended design of an object, like the one above, they look great in VR. When polygons are prominent on an object that was intended to be realistic however, they don’t just look bad, they feel bad too.
* The ability in VR to not just explore your environment but really inspect it, and to live in it in a natural way, invites a level of scrutiny that game assets have seldom needed to accommodate in the past.
* In many cases you may decide that transparency can be sacrificed
* Texture atlases work best

[Christopher Chamberlain (2017)](https://www.blog.google/products/daydream/maximizing-art-assets-vr/)

<https://www.blog.google/products/google-ar-vr/best-practices-creating-art-assets-vr/>

* Use LOD’s
* Busy, noisy and contrasting textures can be nauseating in VR, as they tend to cause a jittering look. However, flat textures with no detail can pose problems as well, because without texture it’s hard to measure motion or depth.
* Check in VR - Putting that tree on a hill may look like a fine distance to cover when you’re designing in 2D, but once you’re in VR and comparing distances, it might not work at all.
* Affordances

Joshua Kinney (2016)

<https://www.creativebloq.com/video-games/game-artists-guide-vr-10-useful-tips-61620671>

* Both Unreal Engine 4 and Unity have a profiler. A profiler is a visual representation of how our game is running. With the help of the profiler we can see where our resources are being wasted.
* Scale is important
* To get proper scale in our games, we need to set up our scale in our 3D software to match the scale of the game engine. For example, one Unreal Unit is equal to one centimetre. So we'll set our system unit scale to centimetres in 3ds Max or Maya.
* Flat planes that are used with texture alphas to fake geometry can be easily spotted if not used properly. Thin geometry is also problematic because it easily creates aliasing at almost any distance.
* One of the traps we can fall into with creating ultra-realistic assets is adding high-frequency detail to the high-poly mesh. This high-frequency detail may look really great up-close or at moderate distances, but it can begin to create unsightly noise when the asset is seen at further distances.
* First, we need to sculpt the large shapes of our assets. These shapes are the most crucial shapes that define the asset. As artists, we have to fight the urge to add fine detail so early. At this point, we should test what the asset will look like in VR.
* Make sure every vertex has a purpose and that purpose is to influence the silhouette of the model as a whole.
* A common complaint in VR is 'sparkles'. Sparkles are caused by a mixture of unnecessary high-frequency detail, glossy materials, lighting conditions and viewing distance. The biggest factor of these listed is glossy materials.
* If we make the material a little less glossy, would it make a difference? If the answer is yes, then we need to consider the high-frequency detail in our normal map. Can we reduce the concentration of the noise?
* Gloves are a great way to get around this issue. The London Heist Playstation VR demo did an amazing job with this. Also Job Simulator is a great example for a stylised VR game that uses cartoon glove hands and it works great. One might think that arms are necessary if you have hands, but they are actually more distracting than helpful.
* First, remember that large billboards in a particle effect are going to look much flatter in VR simply because it's easier to see than on a flat screen. Right now, the only way to get around this issue is to use smaller billboards in our particle effects. Because we have  to use smaller billboards, it means we have to use more of them and this can increase resource consumption

Juan Martinez (2014)

<https://polycount.com/discussion/138023/art-rules-for-vr-environments>

* The biggest challenges are mipping and aliasing. The screen is just inches from your eyes and each camera alias and mips slightly differently. This causes a lot of discomfort. You also lose a lot of resolution with the camera shader used to send a distorted image through the lens.
* (Anti-Alias is the smoothing of polygonal edges. Mipmapping is the smoothing of textures as the angle / distance increase from the render camera.)
* These are the rules we came up with so far.
* Use Chromatic Aberration to correct color bleeding from the lenses.
* Don't use high frequency detail or noise in textures. Even with mip mapping enabled it just turns into swimming pixels.
* 3D geometry "feels" way better than normal maps.
* Specularity instantly give a lot of interest in VR.
* We used a lot of angular and organic shapes because vertical and horizontal lines would align to the screen pixels and alias very badly.
* We rendered the final image at 2592x1458 then down scaled to 1920x1080 to get better anti-aliasing.
* In Unity we used the 8x Multi-Sampling. (Along with the upscaling.)
* Skyboxes require a special shader to display the same on both eyes/cameras.
* We are working on a shader that fades out texture detail at a distance to prevent mipping / swimming textures.
* You need to maintain 76 FPS for low persistence to activate on DK2. This is a challenge because everything is rendered twice, once for each eye.

Chris Pruett (2015)

<https://developer.oculus.com/blog/squeezing-performance-out-of-your-unity-gear-vr-game/>

* 50 – 100 draw calls per frame
* 50k – 100k polygons per frame
* As few textures as possible (but they can be large)
* 1 ~ 3 ms spent in script execution (Unity Update())
* Bear in mind that these are not hard limits; treat them as rules of thumb
* Before you start, make sure that your Unity project settings are organized for maximum performance. Specifically, ensure that the following values are set:
  + Static batching
  + Dynamic batching
  + GPU skinning
  + Multithreaded Rendering
  + Default Orientation to Landscape Left
* A draw call is a command to the GPU to draw a mesh or a part of a mesh.
* Macrotexture / Texture Atlases: Use as few textures as possible by mapping as many of your models as possible to a small number of large textures.
* Static Flag: Mark all objects that will never move as Static in the Unity Inspector.
* Material Access: Be careful when accessing Renderer.material. This will duplicate the material and give you back the copy, which will opt that object out of batching consideration (as its material pointer is now unique). Use Renderer.sharedMaterial.
* Ensure batching is turned on: Make sure Static Batching and Dynamic Batching are both enabled in Player Settings (see below).
* The more parts of your scene that you can mark as static, the better. Just remember that this process requires meshes to have the same material in order to be batched