3D File Preparing Guideline for WebAR

Watch this video or read the text below

Or read this article

General requirements

1. Final file format - **.GLB (with Embedded textures).**
    a. Could be **.GLTF** (only with **Embedded** textures).
    b. Also provide .FBX and folder with all textures in there.

2. **Model size.**
    a. The final file size should be less than **30MB.**
    b. Optimal size is 15MB

3. **Polygons count.**
    a. Very important for app performance.
    b. **Optimal - 35k**, Maximum - 100k.

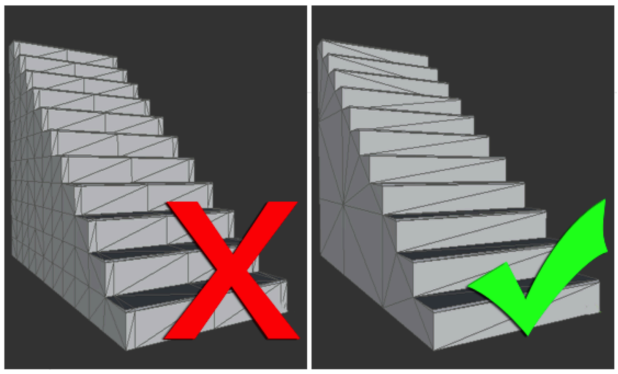
4. Bottom side **Pivot point**, of the scene/object should be pleased at **(0, 0, 0)**. Imagine object stays on XZ surface. It will be a real-world surface (floor/table).

5. **Scale** of the top pivot point in the hierarchy should be **(1, 1, 1)**.

6. **Forward** vector may be along **Z-axis.**

7. **UP Axis - Y.**

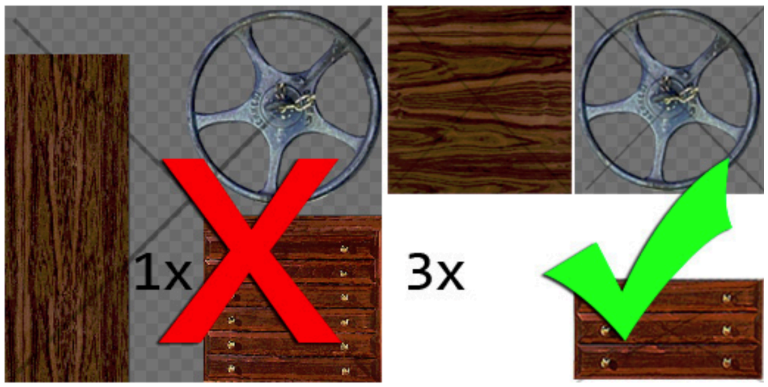
8. **Meshes / Geometries.**
    a. Try to keep **Number of objects** as little as possible. **Under 300 max**.
    b. **Merge what you can merge.** If you have many individual objects and not intend to make them interactive (e.g. bunch of "Clouds", "Buildings") - try to merge and keep them as one object with 1 material. The method you use to construct objects can have a **massive effect** on the number of polygons, especially when not optimized. In this diagram, the same shape **mesh** has 156 triangles on the right and 726 on the left.



    c. Optimize your geometry if it has too many polygons. Many character models need to be intelligently optimized or even rebuilt by an artist

9. **Textures.**
    a. Make sure your textures' resolutions are sized to **powers of two** (e.g., 256x256, 512x1024) in order to avoid the renderer having to resize the texture during runtime.
    b. For big objects in the scene use big textures, for small - small ones. Regular size for WebAR. Texture optimization is a crucial part of WebGL. 1024x1024, 512x512, or 256x1024 is regular but easily could be less in most of the cases.

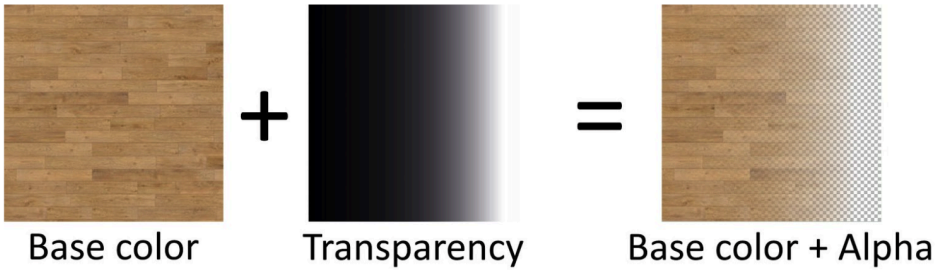
c. For alpha and elements that may require different **Shaders** - separate the Textures. For example, the single Texture, below left, has been replaced by three smaller Textures, below right.



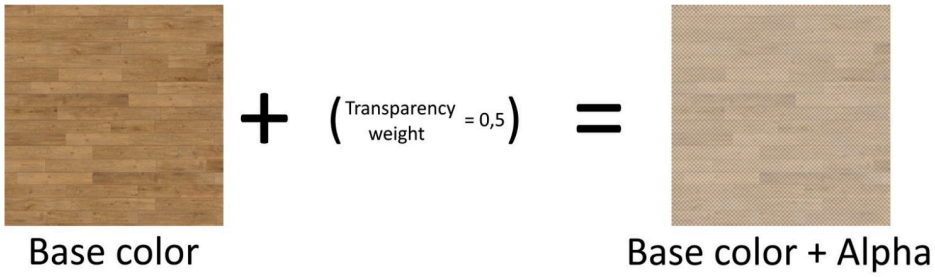
d. Look to **bake your lights** into textures rather than relying on real-time lighting and shadows.
e. Pay attention to Environment Map. They don't export to .gltf/.glb so programmers need to attach them separately with the code to every particular object in the scene (if it's needed). That's why **so important** to keep as few objects in the scene as possible. Please provide EnvMap (in **HDR** or **.JPG** format) if you want a specific Environment.

10. **Materials.**
    a. GLTF uses material representations from Physically-Based Rendering (PBR). Specifically, glTF uses the **metallic-roughness** material model.
    b. GLTF exported properties:
        i. Base color, Metallic, Roughness, Opacity (alpha, true/false). And their intensity.
        ii. *Maps*: Base color, Normal, Env, Occlusion, Emissive
        iii. *Sides*: Double Side, FronSide, Back Side
        iv. *Alpha Mode*: OPAQUE, MASK, BLEND
    c. Shader model, specular, other secondary Textures and substance material **settings are not recognized** or imported.

d. The diffuse roughness map is used as ambient occlusion.
e. Transparency:
    i. Only the color of Base Color is used. The weight of the base color is ignored. Only the weight of transparency is used. The color of the transparency is ignored, as well as other parameters (depth, thin-walled, transparency roughness). In glTF format, the transparency is expressed in alpha (alpha = 1 - transparency). The base color RGB and the alpha A are merged together into a single color RGBA.



    ii. The 2 maps must have the same sizes to be merged successfully. The basic parameter value is used as default value when binded map is not provided:



f. Example: For objects with transparency you have to use:
    i. PBR materials with *BaseColor + Alpha Texture* for reflections and shine

ii. And ORM Texture with metallnes in blue channel and Roughness in the red channel.

g. **More accurate info about GLTF materials.**

11. **Video textures.**
    a. If you use video textures - provide them as separate files.
    b. Make objects in the scene ready for video texturing:
        i. Setup every UV unwrap.
        ii. Use **one frame** (.png/.jpg) as a placeholder texture on those objects for reference. They will be replaced with a video later.
    c. Provide **alpha** and **non-alpha** videos **separately**. Because they have to be computed differently.
    d. For textures with **alpha** provide a video with single contrast color on the background (**chroma key**, e.g. black or green) - it will be cropped with the code.
    e. Combine similar video texture types into one atlas.
    f. Keep in mind while creating the scene:
        i. Video textures play with a **loop**. So if you want to play a different timeframe for different objects - separate them or use them in the same video with the corresponding delay.
        ii. **Do not use** the video as a static texture. Better to use just one frame as a regular png/jpg texture.
    g. Provide text file with a **list of keyframe** and objects turn on/off annotations.

12. **Animations**
    a. Try to keep all animations on one NLA Track if it's possible.
    b. Reduce the number of keyframes where it's possible.
    c. Provide the document with the total frame amount.
    d. It is a **bad** practice - to draw **keyframes** for **every animation step** if it's no needed. You need just draw crucial frames. Not optimized animation increase file size dramatically.

13. **Other suggestions**
    a. Set your system and project units to **Metric** for your software
    b. Animation **frame rate** defaults can be different in different packages. Because of this, it is a good idea to set this consistently across your pipeline (**usually, at 30fps**).
    c. Using a **negative scale** will **reverse** the normal of your objects. These objects will appear correctly in desktop 3d editor, but **incorrectly** in a **Three.js** application.

3D File testing

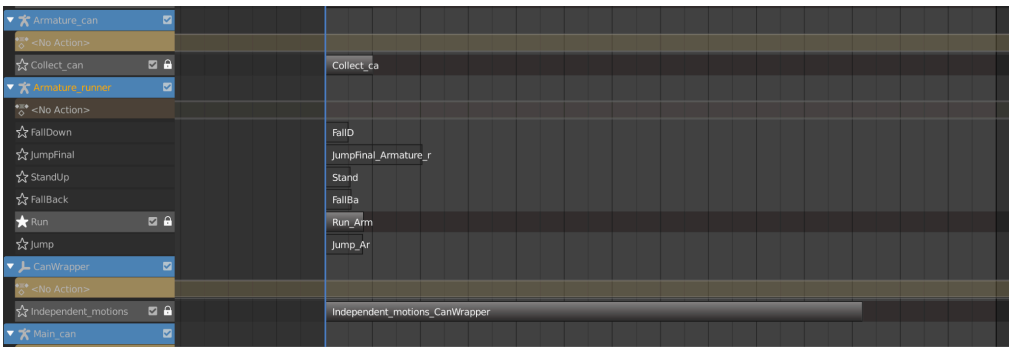
Here is a list of useful web-based applications for WebAR-compatibility tests. Please use them before provide the final assets:

1. Good for material and environment corrections:
    a. https://www.creators3d.com/online-viewer
2. Good for animation testing:
    a. https://gltf-viewer.donmccurdy.com/

3. Alternative viewer. Same thing, different UI:
    a. https://boxshot.com/facebook-3d-converter/
4. Convert .gltf+.bin+textures to on: https://sbtron.github.io/makeglb/

3D Scene with animated textures:

1. **The main starting point is a single .glb file with all textures and animations included:**
    - Try to keep it less than 30mb and 100000 polygons count
    - All animations should be included.
    - Texture compression could decrease total scene size (1024x1024 resolution is enough for WebAR).
2. **Animations:**
    - If animations run independently (with different timing or starts after user interaction) they should be created as a separate animation Action (Non-linear animation) with a unique name. Example attached (example_Non_linear_animation_Actions.png)

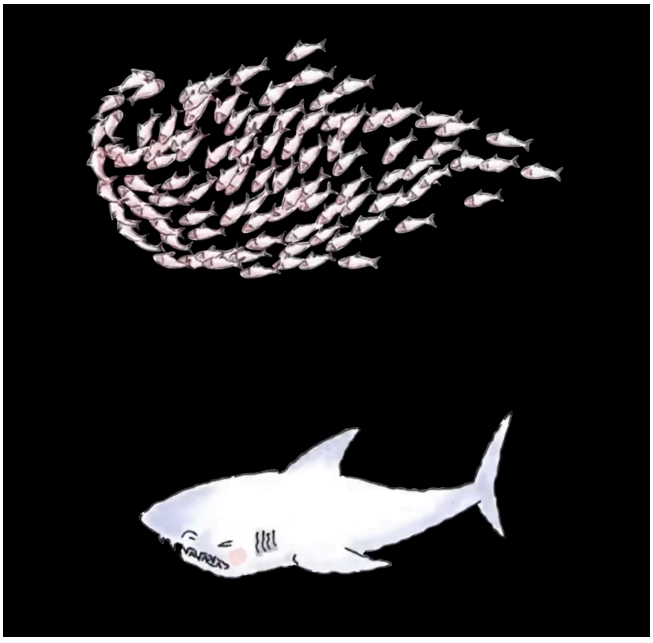


    - .glb animation export allows: scale/translation/rotation, rig/armature animation, morph (animation between two shapes)

3. **Animated textures:**

- .glb doesn’t support animated textures export (video/gif). The only option is to make the mash which will be covered by a video element with custom code.
- To make it easy to integrate you need to prepare video (.mp4) as a separate asset.

- For precise texture positioning you need to make the correct UV map for every mesh in 3d scene by using just 1 frame of the video texture on this mesh. This texture will be replaced with video in the application later.
- Provide timing for texture animation (Like balloons texture start on frame 233. if any specific timing needed).
- If you need to support transparency make a black background for video texture. Black color will be cropped by the shader (like chromakey videos). See example_video_texture.mp4 attached.



- If you need 3d elements playing non-linear - provide .mp4 video textures as separate files.

- Also you can use image sequences (example_img_sequence.png attached below). But we prefer video as a much stable and optimised resource.

4. Provide the list of animation names which runs after user interaction or after a particular frame.

5. Before delivery - test your .glb asset here:

https://gltf-viewer.donmccurdy.com/