Interoperable GLTF extensions (including those like VRM)? #8

humbletim started this conversation in **Ideas**

humbletim on Apr 14, 2021 Maintainer

This seems like a complex topic.. for now just raw transcript of 2021.04.14 #omi-general discussion regarding the possibility of leveraging VRM, GLTF, GLTF extensions for interoperability:

[5:43 PM] humbletim: how does VRM articulate their extensions on top of GLTF? (maybe that can be a reference point for capturing relevant XRPK extensions to WEBN?)
[5:46 PM] Robert: Well, they chose to create a superset of glTF with a separate file extension. But the underlying format is (as far as I know) just glTF with their extension `VRMC_vrm`
[5:46 PM] Robert: https://github.com/vrm-c/vrm-specification/tree/master/specification/VRMC_vrm-1.0_draft
[5:47 PM] Robert: It looks like there are some dependencies on other extensions as well: https://github.com/vrm-c/vrm-specification/tree/master/specification/VRMC_vrm-1.0_draft#dependencies
[5:48 PM] Robert: This is what glTF extensions look like: https://github.com/KhronosGroup/glTF/tree/master/extensions
[5:48 PM] Robert: It's just a namespaced object underneath the extensions property on supported nodes of the json tree.
[5:48 PM] Robert: And then you write a spec and JSON Schema
[5:50 PM] Robert: Namespaces are prefixed by KHR for Khronos extensions, EXT for multi-vendor extensions, or any other registered prefix for vendor specific extensions.
[5:52 PM] Robert: Not all vendor specific extensions are published in that repository. But if you want others to be able to use it, you submit it for review and get it ratified. Vendor extensions typically need very little effort, versus EXT or KHR extensions which can take a while. EXT or KHR generally signal stability and wider support so the working group takes care to get the spec right.
[5:56 PM] Robert: One of the things I'd like to propose is standardizing `MOZ_hubs_components` for wider use. Basically its an extension for describing runtime behaviors of glTF nodes. I'd like to revisit how it's done, but I think it'd be an easy way to define common scene, avatar, and object behaviors.
[5:57 PM] Robert: Paired with a standardized sandboxed scripting API you could use it to describe custom behaviors not defined in the spec.
[5:59 PM] Robert: Scripts could be WASM or JS based and work well in web browsers with reasonable performance or integrated into native apps with a JS/WASM engine.
[6:01 PM] Robert: I've generally seen XRPK as an alternative to this, but now I realize it could be more of a packaging format for something like this. Whether you need that over just packing everything into a .glb Idk.
[6:03 PM] humbletim: just curious but are those MOZ_ extensions related to how Spoke would bundle everything up into a monolithic .glb?
[6:03 PM] Robert: Yeah, that was the system I came up with for our asset pipeline. It's paired with a custom glTF loader in Hubs.
[6:04 PM] Robert: We also have tooling for blender
[6:04 PM] Robert: https://github.com/MozillaReality/hubs-blender-exporter
[6:07 PM] Robert: The general concept works pretty well. I think standardizing behaviors across multiple apps is a place to improve though. A lot of the work is specific to Hubs.
[6:08 PM] Robert: We also never did the scripting API. Something I really wanted to work on, but never was able to due to shifting priorities.

... link to discussion on discord: 2021.04.14 AngellXR#omi-general discussion

↑ 1

0 comments

Category

💡 **Ideas**

Labels

None yet

1 participant