Areas for initial standardization #381

John-Nagle started this conversation in **Ideas**

John-Nagle on Nov 14, 2021 edited

Areas for initial standardization

Contributors

- John Nagle, Animats

Summary

Initial areas that need standardization.

- Portals - you can leave one grid and go to another with your avatar, if the destination grid will accept it. (The destination grid may not want spacesuits in their cowboy sim.) Some of the browser-based virtual worlds already have this.
- Asset portability - you should be able to take your stuff from one grid to another, if the destination grid accepts it. This is what NFTs claim to do, but rarely do in reality. This may already be a legal right in EU countries, per Article 20 of the GDPR, "Data Portability".
- Money portability - you should be able to spend and withdraw money on a grid without being locked into its payment system. Epic is currently fighting Apple over this.

("Grid" here means a virtual world or worlds under coordinated management.)

Example Use Cases

The portal case is obvious, since some web-based VR systems already have portals to other systems. Portals between Roblox games are common, but do not usually cross the boundary between grid operators. Open Simulator systems can have portals between grids under different management. Going from one system to another involves either compatible clients or loading a new client program. Both cases should be supported.

Asset portability is a tough problem. NVidia is working on file formats for that. This is more about how the user does it from inside a virtual world. Can you carry your stuff through a portal?

Money portability is controversial. The standardization issue is that that buyer A, service provider B, money transfer service C, and seller D should be able to achieve a successful transaction in a reasonably standard way. If money transfer service C is acceptable to buyer A and seller D, technical integration between service provider B and money transfer service C should not be required.

Implementation

Web-based virtual world systems can already move between worlds via URL. Most of the hard problems involve what you can take with you. In any case, this needs to be easy and seamless, or we don't have a metaverse, just a collection of disconnected systems.

Asset portability at the file level can build on the GLTF/USD standard. That doesn't address permissions, identity, copyright, and ownership issues. The first step is probably a best practices document for discussion.

Money portability is about payment APIs, of which there are many. Too many. NACHA, which runs the US interbank ACH network, has new APIs coming out to make payments easier without being locked into a single payment provider. ACH has traditionally been somewhat clunky, but that's being fixed. Piggyback on that work. In the EU, there are Single European Payment Area standards. While metaverse operators may offer their own payment system or currency, it should always be possible to use these standard banking systems if the user desires. Cryptocurrency payments are a separate subject.

↑ 1

9 comments

Oldest | Newest | Top

paulespino on Dec 24, 2021

@John-Nagle - QQ:Should this be broken into more granular issues? Also - what is the deliverable on each of these? Knowing that any type of documentation would be somewhat of a leaving document for a bit? Thanks!

↑ 1

0 replies

John-Nagle on Dec 24, 2021 Author

Here I'm trying to flesh out some of the claimed goals of the group:

```
Virtual world URIs
Virtual world metadata discovery
Nested experiences within worlds
Portals and windows into platforms, worlds and experiences
Identity
Portable avatars
Asset portability
```

"Identity" is extremely important, is going to be tough, and deserves its own topic.

A good place to start would be "Virtual world URIs", or how do we find places in the Metaverse? That's closely related to "portals".

OpenSimulator, the closest thing we have now to a working distributed metaverse, defines the externally visible properties of each grid in [this way][http://opensimulator.org/wiki/Hypergrid_Parameters].

That document defines a "Gatekeeper URI". This is a URL which allows entering the grid via a HTTP request. That starts the login process. It may contain parameters which specify a location within the grid. A Decentraland gatekeeper URI looks like this: "https://play.decentraland.org/?position=1%2C-1&realm=hephaestus". So there's already some consensus that a URI is the entrance point for a metaverse grid.

The Open Simulator document also discusses some basic permission issues, such as whether the grid will accept connections from external portals at all.

So, that's a starting point.

The next question is, having found the grid, what protocol is used to talk to it? We're not going to get everybody using the same client. So we need to figure out how you portal to something that needs to load a different client.

↑ 1 0 replies

aiaustin on Jan 5, 2022

A summary of the various (web browser initiable) protocols used in Second Life and OpenSimulator style viewers and environments are summarised in this blog post...
https://blog.inf.ed.ac.uk/atate/2013/06/11/grid-hopping-multi-grid-and-hypergrid-teleport-uri-protocols/

↑ 1 0 replies

aiaustin on Jan 5, 2022

I wonder if the vision of the group is that a single portable avatar format and some elements of sharable content/inventory or a "common" currency is the way to get interoperability? Or do you foresee a range of supportable (progressively more capable) interchange standards (versions/capability maps) that can develop (potentially a lot) in coming years? So metaverse components can state what "level" or capability map they can support.. but there might be fall back positions able to be supported for later more advanced versions. So your fancy "OMI Level 2.5" avatar might be able to appear in worlds that support OMI level 2.0 via defined fall backs - something along those lines!

↑ 1 0 replies

John-Nagle on Jan 5, 2022 Author

Some kind of URL scheme seems to be the way to go for portals. There are two main cases - the URL can be interpreted by the client you're currently running, or some other client needs to be loaded. The latter has the usual problems with untrusted content causing program loading, so some security limits there are required. This is probably a solveable problem.

- single portable avatar format

That's a tough one. One approach is outsourced avatars. That's Ready Player Me. All avatar customization takes place on the Ready Player Me site. Games and virtual worlds use a proprietary rendering module to render Ready Player Me avatars. This gives Ready Player Me a clothing monopoly.

While a single proprietary system is not the way to go, the concept that avatars are independent of the world in which they are currently present is worth exploring, now that we have an example of a system where that works.

This has major performance implications. Remember, all this has to run in real time. We can't affoird the overheads of a web browser, where you need a gigabyte and a billion MIPS to display a page of text and a cat picture. The slowest thing to render drags down the whole client.

↑ 1 0 replies

aiaustin on Jan 5, 2022

How about the concept of a portable open standards descriptor (XML?) for an avatar and that be mapped locally in a metaverse component onto THEIR "visitor" generic avatar skeleton and mesh in a reasonable fashion to act as a travel appearance. The best systems providing great appearance interpretation. The ones less interested in interoperability participating by a very simple basic mapping (even Ruth, Roth or an androgynous variant similar looking avatars at worst).

↑ 1 0 replies

John-Nagle on Jan 5, 2022 Author

Skeleton differences are a huge headache in conversion. Bodies that use the same skeleton can usually be converted. Inter-skeleton conversion is hard.

We're not going to get everyone to use the same skeleton. What might be useful, though, is a basic reference human skeleton, where all the joint positions, names, and orientations are well defined, and a portable way to describe how other skeletons could be translated to that skeleton. Then you just need a translation definition to and from standard for each system. Thus, N translators, rather than N^2.

- Unreal Engine skeleton: https://docs.unrealengine.com/4.27/en-US/AnimatingObjects/SkeletalMeshAnimation/Skeleton/
- Unity skeleton: https://docs.unity3d.com/Manual/UsingHumanoidChars.html
- Roblox skeleton (R15): https://roblox.fandom.com/wiki/R15
- Second Life skeleton: http://wiki.secondlife.com/wiki/BentoSkeletonGuide

They all do roughly the same thing, but they are not compatible. Add to that list, please.

Nonhuman is a problem. The Second Life standard skeleton currently has provisions for four legs and a tail, just in case you need them.

↑ 1 0 replies

pathorn on Jan 5, 2022

@John-Nagle Regarding skeletons, we are further along thanks to work done in the past few years by other communities, V-Tubers and the recent push for social VR tech.

A lot of the Open Metaverse Interoperability community have settled on VRM as a starting point for humanoid skeletons, which is based on the Unity skeleton as you linked, and has a thriving ecosystem and tooling economy around it.

VRM's humanoid object (json schema) thus defines the following set of humanoid bones: https://github.com/vrm-c/vrm-specification/blob/master/specification/0.0/README.md#defined-bones

As for orientations, VRM 0.0 defined all bones to be pointing up in world space (identity quaternions, as the forward direction of bones are always Y-up in most formats). VRM 1.0 may remove this requirement, opening up the possibility of preserving the original rig rotations.

Since Unity is widely used, it has made VRM adoption pretty straightforward for many platforms. On the web, Three.js also has an extremely well-maintained library for VRM thanks to Pixiv, Inc.

Given all the consensus around it, it seems fair to use VRM as a standard upon to base any work we do related to humanoid skeletons. I agree that non-humanoid is an open problem. Some platforms use "rotation constraints" as a way to allow copying rotations from one limb to another in non-human characters. VRM 1.0 includes support for such constraints. But ideally, we can do even better. In my view, it might be worth basing some work on second life's extensions for tail and extra limbs, but within the GLTF or VRM framework.

↑ 1 👍 2 0 replies

John-Nagle on Jan 5, 2022 Author edited

Denavit-Hartenberg notation might be helpful here. It's a standard way to describe robots made from links and joints. It's not a specific model. It's a way to describe any open kinematic chain. It's used in robotics.

Some packages in ROS, the open source Robot Operating System, use it. Here's an example in XML format:

```
<node pkg="tf" type="static_transform_publisher"
        name="base_link_new_base_linkbroadcaster" args="0 0 0 $(arg PI) 0 0 base_link
dh_base_link 20" />
  <node pkg="tf" type="static_transform_publisher"
        name="shoulder_link1_broadcaster" args="0 0 0 $(arg PI) 0 0 shoulder_link link1 20" />
  <node pkg="tf" type="static_transform_publisher"
        name="shoulder_link2_broadcaster" args="0 0 0 $(arg PI) 0 $(arg PI_2) shoulder_link
link2 20" />
  <node pkg="tf" type="static_transform_publisher"
        name="forearm_link3_broadcaster" args="0 0 0 $(arg PI_2) $(arg PI_2) 0 forearm_link
link3 20" />
  <node pkg="tf" type="static_transform_publisher"
        name="wrist_2_link_link4_broadcaster" args="0 0 0 0 0 $(arg MINUS_PI_2) wrist_2_link
link4 20" />
  <node pkg="tf" type="static_transform_publisher"
        name="wrist_3_link_link5_broadcaster" args="0 0 0 0 0 0 wrist_3_link link5 20" />
  <node pkg="tf" type="static_transform_publisher"
        name="ee_link_link6_broadcaster" args="0 0 0 $(arg MINUS_PI_2) 0 $(arg MINUS_PI_2)
ee_link link6 20" />
```

This just says what attaches to what and what degrees of freedom it has at each joint. It's a uniform way to describe different skeleton systems, not a standard skeleton.

You'd need one such file for each different skeleton model, and that would provide the info needed to translate between them, assuming they have comparable bones. It should be possible to use that info to translate animation data.

Not sure what to do about facial animation. In some systems, that's done with morphs; in others, with bones.

↑ 1 0 replies

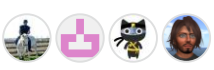
Category

💡 **Ideas**

Labels

proposal

4 participants



Converted from issue

This discussion was converted from issue #122 on April 12, 2023 22:24.