The Contiguous Meta-Verse Standard (CMV)

Originally: StruMML and RDS Revised

Originally Proposed Document Circa 2008

September 3, 2022

Revised: December 10, 2022

Walter Gress

Twilight Raven Games LLC

(Draft)

**Legal Disclaimer**

This research is proprietary intellectual property of Twilight Raven Games LLC. Its implementation and redistribution are licensed under the GPL (GNU Public License)

Available for review at <https://www.gnu.org/licenses/gpl-3.0.en.html>

**New Introductions and Project Intentions**

In light of recent development of SPIR, Pixar’s USD format, and research being done to implement USD in both hardware and software, our StruMML/RDS file format definitions and their future technologies have been turned into an SDK now known as the Contiguous Meta-Verse Software Developer Kit, a name given to represent the seamles 3-D metaverse technologies we wish to represent. This will include further work into the use of RDS as a system of correctives, to store and distribute session data, StruMML as a working fast-parsing system for Metaverse documents and technology to allow for RDS deltas and to support a USD based back-end. There are things such as rigging standardization that USD does not provide for which we may choose to provide standards as a part of our CMV technologies. We are considering including USD specific tags as part of the StruMML standard, with RDS acting on the deltas of those tags. The underlying framework remains the same.

Throne will serve as a CMV web host, Alliance as a Qt3D/SPIR based Metaverse browser for laptops, tablets, phones, etc. and we are also now introducing a third project, Helm, that will use the Metaquest VR technology and SDKs developed by Meta. Helm will be an app available through Metaquest VR headsets to browse the CMV in better immersion. We will be working closely with all of the emerging technologies listed to provide the best CMV experience. Technical details will be provided in this document as well as our source code as we become more familiar with USD API, Qt6 API and Metaquest’s SDK.

**Introduction**

StruMML (Structured Metaverse-Markup Language) or as it was originally proposed (Structured Modeling-Markup Language) and RDS (Relocatable Document System) are document standards that I first proposed to my employer at the time before being placed on a research and development project involving the development of the first generally portable AR/VR technology. The idea I proposed was to revise the Internet with the document system that used two new types of documents that would provide for 3D modeling and physics data embedded into them so that one could browse a 3-D game-like web using AR/VR equipment. This new document system would provide a 3-D Internet using the current Internet as a foundation and be similarly browsed. Unfortunately, I suppose I was a bit ahead of the time, and promptly joked about for my proposal. As often happens to those with forethought. I suppose this happens to any who try to think too far into the future until that future arrives. The idea that I proposed originally is now known as the group of disjunct Metaverse technologies and is being actively pursued by many billion-dollar companies. I am not saying I was the first to think of it, but I had a contribution to the dream. But further on.

The Metaverse standard I propose can be constructed on top of the present Internet architecture for broader exposure, browsers only need change their portals into these pages to handle them. The infrastructure can stay the same and we can build on what we have as we always have. Specifically, the decision to build this on current infrastructure is because it is still primarily *document based*. The significance of this system is not necessarily dependent on any rigging technology, modeling data structures, physics engines, shader languages, skin binding algorithms, algorithms for viewing in general, any specific format for the markup language, or what one most might expect from a standard to be proposed for a seamless metaverse. Alike to to the beginning of the Internet, we now have many metaverses but not a Metaverse. The significance is instead of storing all the information in a single page with embedded links and client-side code like we already do for DOM, JavaScript, etc. We not only support a DOM document that represents a 3D boundary sector that embeds links to resources, but we now store a second, relocatable document for managing server-side sessions across multiple users. This is the RDS document, the Relocatable Document System. We keep the definition in the StruMML document, and its applicable morphs, updated physics properties, bone transforms, and material changes in RDS documents. RDS documents could take the form of a JSON tied to each DOM node. There is room for flexibility.

**StruMML ( A Structure/Structured Meta-Markup Language)**

A StruMML document acts as the base web page. It defines a sector within the Metaverse. Instead of a web page you have a sector of 3-D space. The sector boundaries act as URIs in the StruMML document, leading to other StruMML pages. Their contents can be cached by the browser similarly to how we cache images and displayed using LOD implementations similarly to algorithms like mipmapping, or similar algorithms level of detail caching for textures and geometry by the browser. This can be browser independent. The StruMML document is kept server side and sent to the client upon request similar to the transport of an HTML document. Instead of using IMG SRC tags it uses JSON SRC tags, which point to a JSON with readable geometric format data, materials, weight maps, rigging, etc. that the browser supports. JSON SRC tags have attributes that allow them to point to skin-binding/rigging information for the model, physical attribute files, and their associated materials that all can be browser independent. Just like HTML the StruMML standard is meant to flexible so that browsers can implement it in their own way to encourage competition and eventually collaboration and standardization.

Interpretation of character models rigging and animation including physics sim data has been hard to keep consistent across platforms. Everyone has their own algorithms and their own proposals on ways to implement it. It is unlikely we will agree on a standard, just as the early days of the Internet were so chaotic (especially with various versions of Javascript floating around). Both privatization and competition led to the standards we adhere to today, with a little bit of intercompany cooperation and anti-trust suits. We need to keep this portion of the Metaverse upgradable as hardware progresses and we can do more. As we can realize better skin-binding algorithms, better real-time ray tracers, better GPUs, and hardware in general, we need to be able to support old data models and algorithms but keep consistency within the Metaverse. The Metaverse will need interoperability across all of its sectors. Right now, metaverse technology is very disjoint. There isn’t even an accepted standard for avatars for representation of metaverse browsing individuals. In order to keep the Metaverse consistent yet upgradable we must be do this: **We must be able to provide a mechanism for definition and interaction that is malleable and contiguous.** If we keep the Metaverse as it stands now, groups of companies individually developing their own technologies – hardware and software – and their own private “metaverses” we will never realize the successful dream of a seamless Metaverse in a realistic amount of time. We need some consistency to found our technology on, just as the first world wide web servers provided consistency for the modern web and its future development.

**RDS (Relocatable Document System for Server and Client side session deltas)**

The second document used alongside StruMML is the RDS standard. The RDS document acts as both a session for the user and a modifier. It is kept client side and its master is maintained server side. A relocatable document system using encryption negotiating protocols are necessary to exchange changes from StruMML document to StruMML document, so users can engage with the sector dynamically and with others dynamically, without endangering the base definition files. Unlike the parallels between HTML and StruMML, RDS is more like a shared session, a universal session across all users of the Metaverse. Things like bots and AI can be used to enforce definitions and govern the Metaverse, but the ability to use a shared session in the form of RDS means that action can be happening anywhere in the Metaverse and leave a lasting impact, but the owner of the sector or “property” so be It still has the rights to restore it to its definition. RDS holds the local and master “deltas” or deviations from the JSON tags that are linked in a StruMML document. It is up to the web host to resolve these deltas when receiving new RDS submissions from the client or clients, negotiate interactions between multiple simultaneous client RDS submissions, and send new RDS documents to the client side to manage the resolution of master RDS deltas from the original StruMML page sent. Upon original sector visitation, using a dirty flag for caching, a sector will have downloaded StruMML data. Then the modifier stack will be built using the master RDS file whenever sent from the host in the web browser. This modifier stack resolves deltas between the master RDS and the StruMML document. The local RDS session then applies user interaction to the current scene and posts a client-side RDS to the webhost when TTL runs out for the document. TTL can be forced to update by forced sending of a new master RDS document during client-side interaction.

**Impact and Incorporation of Crypto Currency and Present Blockchain Technology**

Again, the framework is already there and being developed using web 2 technology, why reinvent the wheel? TODO

**StruMML Requirements (WIP)**

The following are requirements for StruMML and details on its implementation.

I have recently debated whether it would be more efficient to represent StruMML still as a markup language or as something more similar to JSON, with nested upon nested content. For parsing purposes JSON maybe more efficient and using StruMML may keep some cross-compatibility between current hyper-text markup language. So, what is to be gained through using a markup-language, something originally designed for interpretation for 2-D documents in a 3-D environment? While JSON easily supports embedding data, the ability to tag using markup languages like HTML I believe make using StruMML, a 3-D structural mark-up language as a better option. The core of StruMML is to make a document that can be easily read by a web-browser, locate 3-D resources within that document, and render them in a 3-D scene that can be interactive.

Then why use a mark-up language and not some scene format like exported from Maya, or something similar to be used for Unreal, Unity, or other 3D engines and technologies? Why a markup-language still you may ask? First, plan text transmission is ideal, that is granted. My chief argument for using a structured markup language is following. With the use of RDS we will be consistently embedding client-side code and tags similar to our current <img src> and <div> tags. By using a model similar to DOM to access the contents of these documents, these documents can be modified, updated, and maintained more easily than any other present document format. Updating something in a format such as JSON requires considerably more work in parsing and through the use of a markup language based off a DOM like format, we will be able to not only add new tags by modifying DIV structure, we can even append new tags to represent modifications to the original tags easily. Just as we can use CSS to restyle a document, CSS-like supplemental documents could be provided to redress these 3-D scenes with 3-D like style classes.

The following are requirements for RDS and details on its implementation.

Requirements for a StruMML/RDS based Web3 system:

The system shall be called Relocatable Document System

The system shall distribute establish multi-user session data using RDS documents

The system shall exchange RDS documents between client and server

The system shall use StruMML documents to prepare a definition for a sector

The system shall denote a Sector as a 3D space documented through a StruMML document

The system shall use JSON source tags with model geometry leading to common model formats in StruMML

The system shall use JSON physics simulation source tags leading to common simulation in StruMML

The system shall perform morph application to StruMML models using RDS documents

The system shall perform animation changes to StruMML base definitions using RDS documents

The system shall securely exchange RDS documents between client and server

The system shall use server-side technology to negotiate multiple RDS document influence

The system shall use secure cryptographic technology to ensure RDS exchange security

The system shall allow client-side JS to be included in StruMML tags to execute client-side operations

The system shall allow for server-side languages to interpret embedded StruMML tags similar to server-side JSP and ASP technologies

The system shall use the familiar DOM (Document Object Model) to access StruMML documents structure

The system shall represent a simple StruMML document as a 3-D sector bounded space

The system shall have a special system for representing links to other StruMML documents in 3-3D space

The system shall have borders representing hyperlinks to other StruMML documents

The system shall use a common caching system for browsers to provide low LOD previews for StruMML hyperlinks

The use of StruMML hyperlinks will allow for a seamless 3-D browsing world

The system shall use RDS documents to represent Avatars in their entirety

The system shall use Avatars as a a representation of the user in the Metaverse environment

The system shall provide Avatars through the associated Web3 browsing mechanism

The system shall require the browser to interpret the StruMML and its associated server-side RDS documents

The system shall use server-side RDS documents as modifiers on StruMML definitions

The system shall user the server to accept client side RDS documents as apply their modifications

The system shall refuse RDS modifications according to server policy

The system shall via common network negotiation algorithms (CNN) will resolve changes by multiple submitted RDS documents

The system shall assume a common RDS definition that is updated by submission of client side RDS documents that are resolved via CNN  
The system shall preserve presentation through StruMML and downloadable RDS definitions for rasterization purposes

The system shall preserve collision detection through server-side common StruMML implementations (APIs)

The system shall preserver physics simulation through server-side common StruMML implementations (APIs)  
The system shall preserve animation simulation and IK modification through common StruMML implementation (APIs)

**RDS Requirements (WIP)**