

Virtual Worlds: State of the Art

where are we and where we want to go

Sergiy Byelozyorov

Saarland University

IMPRS-CS

Hi. My name is Sergiy Byelozyorov and I do my PhD research at Saarland University, Germany. In this talk I would like to give an overview at the current state of research in virtual worlds and give an outline of new directions with open problems.

What Is a Virtual World?



3D
World

First of all, I would like to answer a question of what is actually “a virtual world”? We all know 3D worlds, which usually consist of some objects, animation, scripts. However, this is not a virtual world yet as it’s only available on one computer.

What Is a Virtual World?



We can make a copy of the 3D world to another computer, however, this still doesn't make it a virtual world, but rather two different copies of the 3D world. Changes in either of them won't be visible in another.

What Is a Virtual World?



However as soon as we set up some sort of synchronization between these two 3D worlds, we have a virtual world. Of course, virtual worlds are not limited to two communicating parties and sometimes have millions of synchronized 3D world copies. In order to synchronize these copies, there are various technologies. Some of them are based on peer-to-peer networks, while others employ traditional server-client model. Typically in either of these configurations there are two or more communicating entities, and, for simplicity, we will only consider server-client model in this presentation.

Components of the Virtual World

High-level overview

Let us have a high-level look at the basic components of the virtual world.

Components of the Virtual World

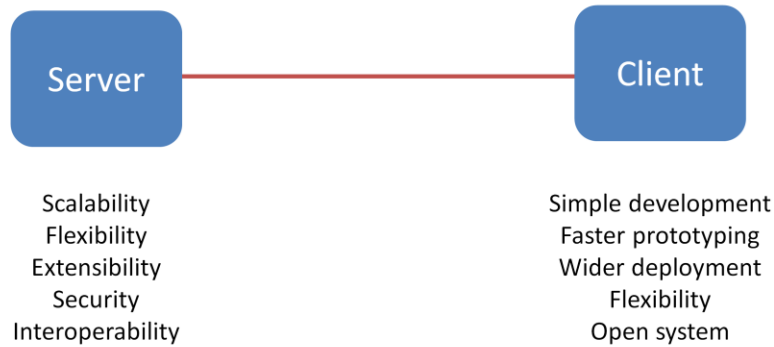


Server

Scalability
Flexibility
Extensibility
Security
Interoperability

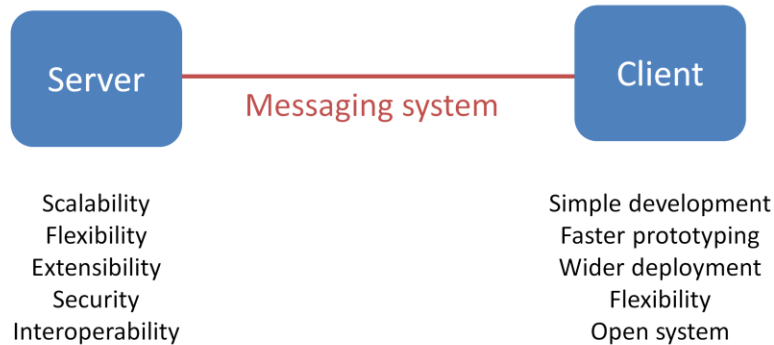
First of all there is a server that clients typically connect to. There have been done a lot of research in this area in the last couple of years and some great results have been published. In particular, research done by Mic Bowman's research group at Intel and Sirikata's team at Stanford should be noted. Server architectures are typically expected to be scalable in order to allow connecting potentially millions of users at the same time. They should also be flexible to support various virtual world types and extensible to allow adding more functionality with time. Security has played an important role in the Web and so server architectures should be designed with security in mind, especially if we want to bring virtual worlds to the Web. Finally, interoperability is an important property that allows server systems that are run by different entities (companies, providers) to communicate and form a single shared metaverse world. This would make users' experience similar to the Web, where one can click on a link leading to a different web-site and seamlessly be transitioned there.

Components of the Virtual World




On the other hand, the clients, which used to be very specific to each particular virtual world, are becoming more generic and universally usable. In Computer Graphics Group at Saarland University, we have been working on a solution that allows Web-developers to create virtual worlds in a simple manner. We have been focusing on fast prototyping, wider deployment, flexibility and open nature of the architecture. More details about this architecture will be given in the next slides.

Components of the Virtual World



Finally, as a third component, there is a messaging system that is used as a communication protocol between a client and a server. Unfortunately, there have been little research in this area yet and most of the virtual worlds are still using low-level protocols that are specific to each particular world. We would like to do more research in this area and develop a generic system with which one can efficiently sync a 3D scene, which is some arrangement of objects, animations and shaders, to somewhere else to be able to create synchronized shared environments, or as previously defined, virtual worlds. In the end of this presentation I will present an idea that we believe will provide a better messaging system.

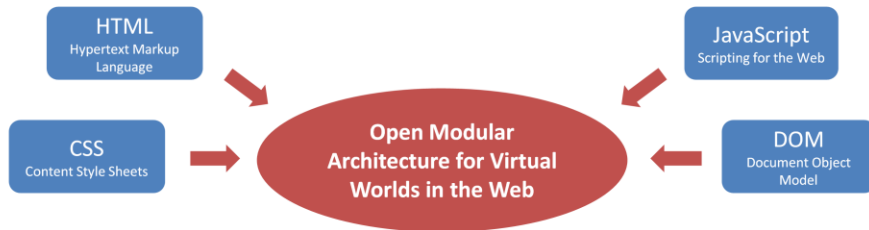
Open Modular Architecture for Virtual Environments

A red oval logo with white text inside, centered on the slide.

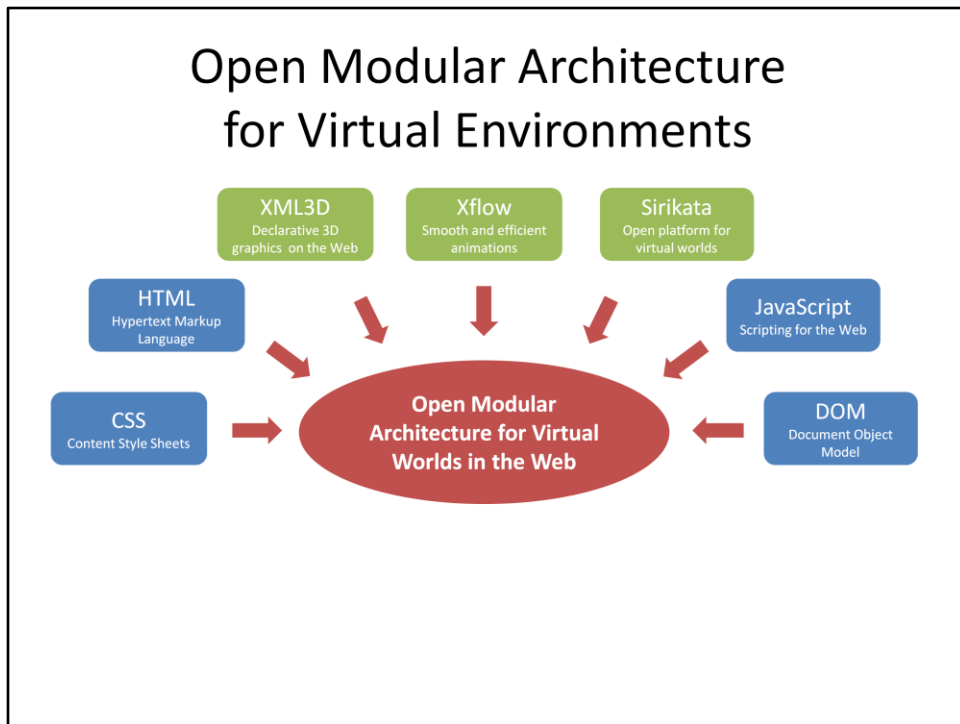
Open Modular
Architecture for Virtual
Worlds in the Web

Now, as promised, let me give you an overview on the open modular architecture for virtual environments that our group has published in Cyber Worlds conference in 2011. In order to achieve properties of the client mentioned on the previous slide, we have decided to integrate it into a browser. This was also motivated by the fact that there are many more Web developers who are experienced working with the Web technologies compared to 3D experts who know how to create virtual worlds applications. This motivation has influenced our design as we tried to use technologies that are better suited for Web developers.

Open Modular Architecture for Virtual Environments



First of all Web can already offer us a number of components, as for example, HTML can be used for creating 2D user controls and JavaScript for scripting the world. We get these components for free and Web developers know them very well.



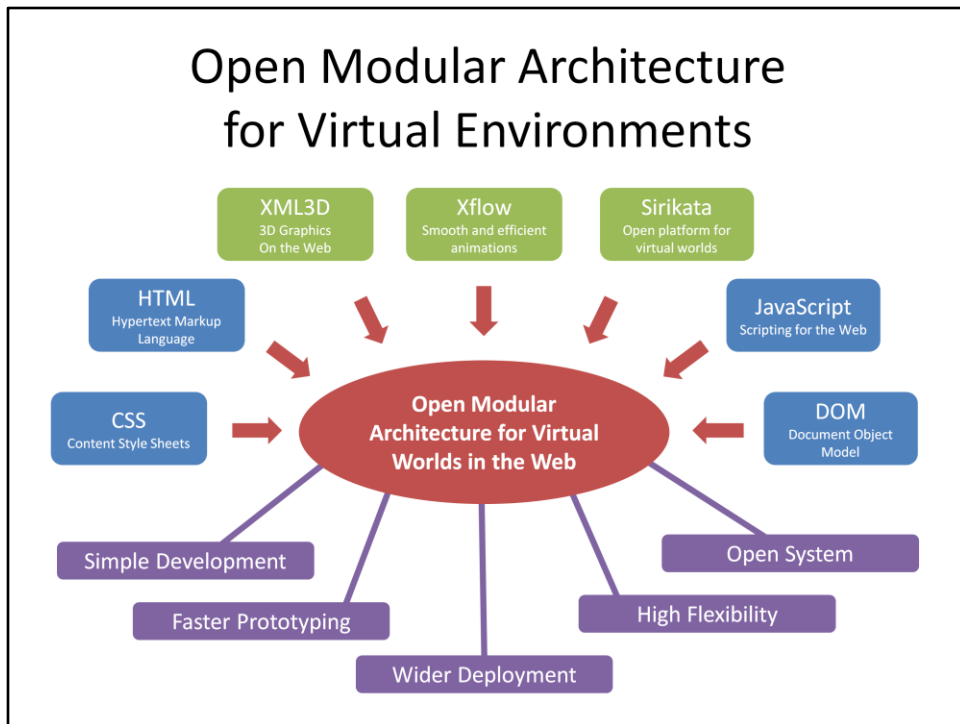
However, in order to create a virtual world in the browser a couple of additional technologies are required. We have extended the browser XML3D, Xflow and Sirikata.

XML3D, which is developed at Saarland University together with the German Centre for Artificial Intelligence (DFKI) and the Intel Visual Computing Institute, defines a format for encoding 3D graphics as an extension to HTML. It has adopted the best and most widely available Web technologies familiar to Web developers. The XML3D specification defines a 3D document as part of the DOM model in the browser. It reuses ideas from Scalable Vector Graphics (SVG), HTML-5, CSS, and more. Elements that encode different aspects of the 3D scene (such as geometry, material, lights, etc.) provide interfaces to JavaScript that allow an easy and real-time modification of the document. XML3D is efficient because it uses vertex arrays to natively encode geometry, which can be directly mapped to the hardware. More information can be found in Sons et al. "XML3D: interactive 3D graphics for the web", published at Web3D in 2010.

Xflow, is an extension to XML3D and defines a data flow declaratively in the document. The main idea behind it is that we can represent animations as graphs where only a few data elements are changed, such as bone positions, but influence many more data elements, such as the final mesh vertices of a character. With this approach, a programmer defines a flow in the document and only changes a few parameters in JavaScript that define the animation, while the underlying system

propagates data through the Xflow graph and creates a smooth animation on the screen. In this presentation I will discuss how it can be efficiently used to synchronize animations in a virtual world. More information can be found in Klein et al. “Xflow – Declarative data processing for the Web”, published at Web3D in 2012.

The Sirikata project, which is developed at Stanford University, offers an architecture that separates network nodes into space servers and object hosts, where space servers are responsible for managing a certain constrained space in the shared world, while object hosts run the objects within spaces. This approach offers a better security model, is scalable to a large number of players and offers a interoperable world, where parts of it can be controlled by different organizations. Additionally, Sirikata developers provide a ready-to-use implementation of the server and client libraries which help developing new Web clients. More information can be found in Horn et al. “To innity and not beyond: Scaling communication in virtual worlds with Meru”, published at Stanford University in 2010.



Let us now see what benefits does this combination of technologies provides.

Firstly, it makes the integration of virtual worlds in the browser much simpler for Web developers, as no special knowledge in the areas of networks, graphics, or system architecture is required, other than Web technologies. In addition, developers can quickly prototype their applications using high-level concepts oered by the underlying technologies.

The approach is highly flexible due to the modular architecture, which allows replacing components easily. In the Web community, this approach is known as mashup, where a developer combines data and functionalities from two or more sources to create new services and applications.

Wider and easier deployment is achieved as users do not have to install additional specialized software, but instead use the browser that is typically shipped pre-installed on their computer. An illustration of the world running in different operating systems and using different rendering algorithms is shown in Figure 1.

Finally, an important property of such design is that it is an open system, i.e. it is based on open source code, open protocols, and an open specification. The development of the Web has shown that open systems are more successful than the ones based on closed or proprietary technologies, thanks to a more dynamic

response to user feedback via typically shorter integration cycles, and their sustainability beyond the life expectancy of a particular contributing entity.

Efficient Animation Synchronization

Define
animation as
data flow

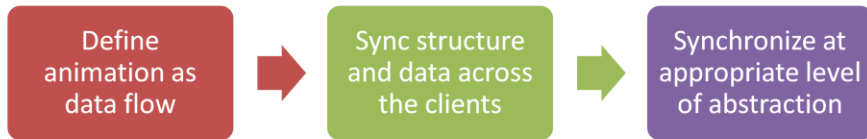
Let us now look at how Xflow can be used to synchronize animations in a virtual worlds. The process involves 3 steps. First, developer need to define a data flow graph that describes the animation.

Efficient Animation Synchronization



Then it needs to be synchronized with other client, which requires us to send the entire graph and related data to all clients once. This can be achieved using Content Distribution Network to decrease the load from the synchronization channels.

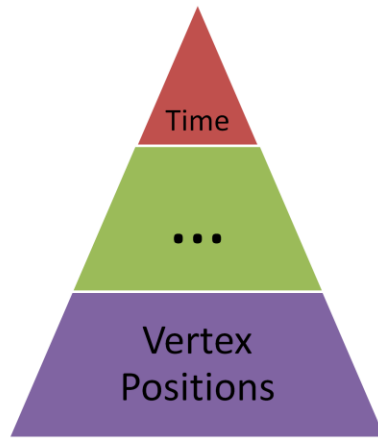
Efficient Animation Synchronization



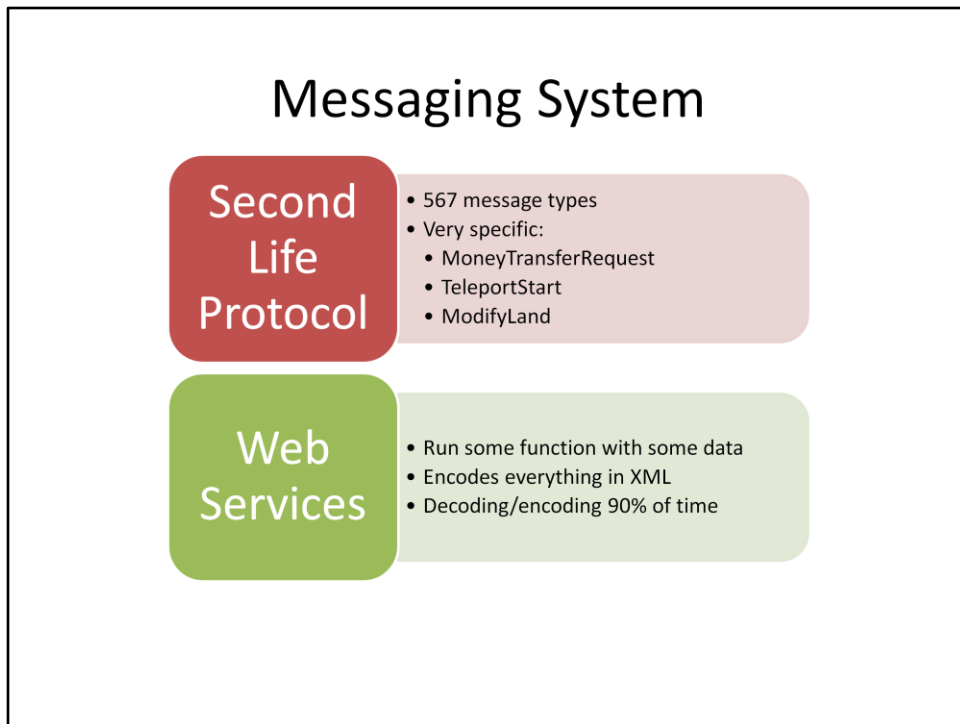
Finally, when each client has a copy, one can choose different levels of abstraction to synchronize the animations across the copies of the 3D world.

Various Levels of Abstraction

- Sync animation progress only
 - Maximum throughput
- Sync every vertex
 - Maximum flexibility
 - Maximum security
- Anything in between
 - Character skinning
 - Parameterized animations



One can synchronize just animation progress, which is typically a time variable that changes from 0 to 1. This approach gives us maximum throughput and offers synchronization for many users and many animations. On the other extreme, we can sync every vertex position, which offers maximum flexibility as we can do arbitrary changes to the geometry and maximum security as only vertex positions need to be synchronized to other clients and not the underlying graph (which may be proprietary). Finally, we can choose a number of approach that lie in between these two extremes. For instance, we can use character skinning and only sync bone transformations or use parameterized animations that are synchronized by a number of parameters.



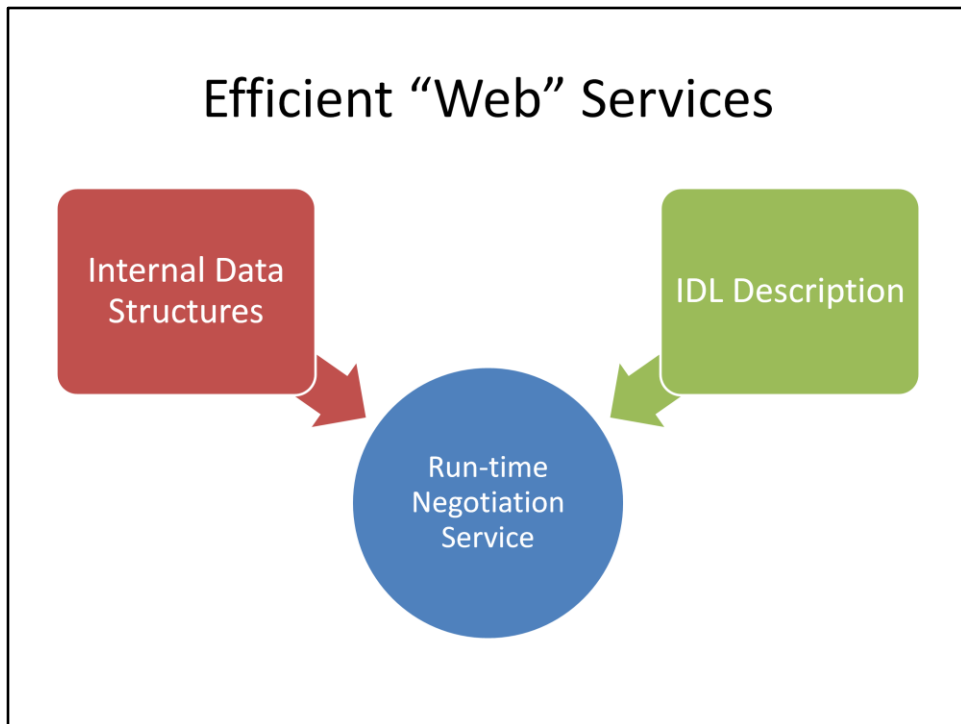
Now, as mentioned before, we would like to do research in the messaging system component of the virtual worlds. Currently existing protocols are very specialized, as, for example, Second Life Protocol defines 567 message types and some of them are very specific to that virtual world. ModifyLand message would make no sense in a business meeting virtual world and MoneyTransferRequest message would not fit into a virtual world where architects are designing a future building. On the other hand Web Services offer us a generic paradigm that allows to run some function with some data and allows clients to define the details for each particular application. Its disadvantage, however, is that it uses XML for encoding and decoding all the messages, which takes up to 90% of the processing time and results in larger messages. This is especially critical for virtual world, where servers are limited by bandwidth and processor resources.

Efficient “Web” Services



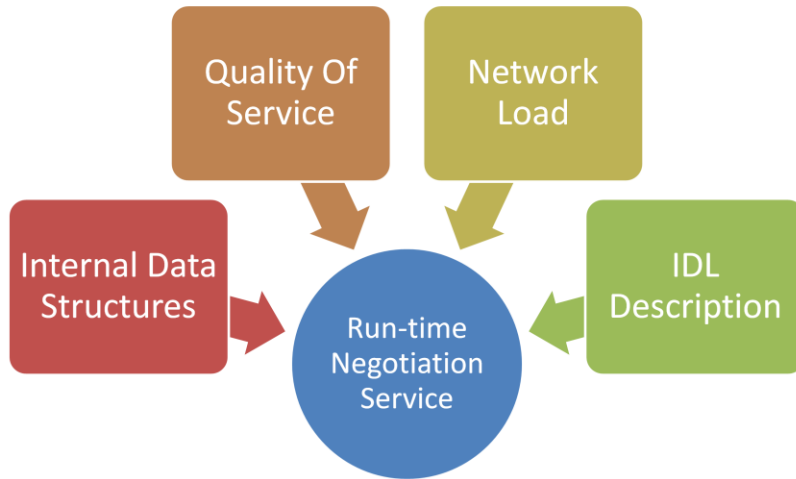
Run-time Negotiation
Service

Our idea is to add a run-time negotiation service that will define a protocol at run-time.



Developers need to provide this service with an IDL description of what they want to transfer and what are the internal data structure for each particular client.

Efficient “Web” Services



The negotiation service will additionally collect information such as quality of service requirements and measure parameters, such as network load, which are only available at run-time and will choose a protocol that will be most efficient for each communicating pair and current requirements. This allows application developer only to think about what needs to be transferred and not how it will be transferred. At this point we can optimize by adapting this protocol.

Thank you

Questions?

With this, I would like to thank you very much for your attention and ask to send any questions to byelozyorov@cs.uni-saarland.de. I will be glad to answer them.