LiveCoding.space: Towards P2P Collaborative Live Programming Environment for WebXR

Nikolai Suslov

Fund for Supporting
Development of Russian Technology
Vologda, Russia
SuslovNV@krestianstvo.org

ABSTRACT

WebVR and WebXR are the new standards of Virtual/Augment/Mixed Reality for the Web Browser. A wide variety of head mounted displays, motion controllers, mobile and standalone headsets bring that technology to the masses. But software engineering was not ready for these new challenges. Web applications become look like desktop apps with all advantages and disadvantages of application-centric approach. For example, using existed React VR or A-Frame libraries, someone could easily create a rich Web app, but it will lack of self-exploratory environment, multi-user collaboration and live programming at runtime mode.

To address this problem, we propose to use the Virtual World's concept for WebXR applications development. Virtual World as the new computational paradigm blurs the borders between application and hosted environment, runtime and development mode. The Virtual Worlds in conjunction with WebXR technologies offers to both programmers and domain experts nearly unlimited capabilities for creating novel computer-based simulated environments just in Web browser. Virtual time, user-defined meta language, live coding, avatar, self-exploratory environment become the new crucial concepts of the Web applications.

This paper introduces the prototype of pure-decentralized P2P collaborative, live programming environment: LiveCoding.space. Having the tight integration of A-Frame, Virtual World Framework, Gun DB storage system and Ohm language, it provides all-in-one solution for development of creative applications in modern Web standards for virtual reality.

1. INTRODUCTION

Let's describe a web application as a virtual world with self-exploratory entities inside computation-centric live programming environment. Virtual worlds represent the new computational paradigm and the new form of software, where everything is just some form of a computation. End-user could generate or change the content of a virtual world in real-time. The user also is represented in a virtual world as a computational process, an object known as Avatar. Such full-body immersion environment, which is built using virtual worlds, can scales conformal up to the unlimited number of hardware and software nodes (Figure 1).

The most known virtual world platforms existed today are High Fidelity, Sansar, Spatial OS, Immersive Terf. Still, they are all desktop-based applications, they are considered as Web ready. For example, High Fidelity has JavaScript API and deep integration with Clara.IO, a full-featured cloud-based 3D modeling, animation and rendering software tool that runs in a web browser. Also, there are a huge amount of standalone applications developed using game engines like Unity3D or Unreal Engine with support of Web Socket API and HTML WebGL ren-

dering. All this VR oriented software aims running on any platform from mobile Web browser to standalone headset with HMD (head mounted display), desktop application or industrial CAVE (automatic virtual environment) systems. They are based on their own application frameworks and include specific software bridges for interconnection. But all listed platforms and frameworks relate to the client-server architecture and split runtime and development environment modes.

In this paper we present the prototype of collaborative Virtual World environment which towards WebXR / VR standards, P2P pure decentralized network architecture, providing self-explorative live coding features and united runtime and development modes.

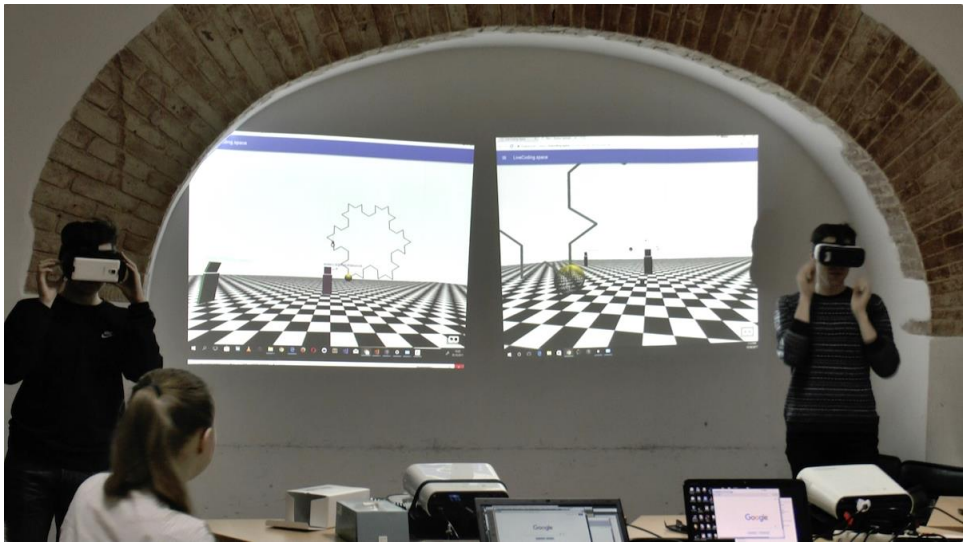

Figure 1. LiveCoding.space prototype in action. Two mobile Gear VR users and two on the laptops, sharing their "personal views" trough virtual cameras, 2017.

2. THE NEW APPLICATION ARCHITECTURE FOR WEB XR / VR

There are only few WebXR / VR ready frameworks existed today: A-Frame and React VR. These frameworks are making a step forward in finding a solution for building virtual environments purely in Web Browser. They are both powerful component-oriented frameworks, that provide a declarative, extensible and composable structure for WebGL, VR headsets, motion controllers JS libraries etc., but they have nothing about network collaboration stuff inside.

For the prototype we choose A-Frame entity-component framework. The A-Frame framework solves the problem of Three JS complexity for developing Web applications for Virtual Reality. A-Frame encapsulates Three JS, hiding the internals, and providing high-level interface for describing web applications declaratively. But there is no network synchronization model in the core of the A-Frame, not counting the A-Frame component - networked-aframe, based on client-server architecture. We have chosen the VWF (Virtual Web Framework) for implementing network stuff, as the decentralized and integral solution. VWF provides the strong architecture to build virtual environments as on its own or by integrating into any existed Web and non-Web applications. The key concept of Virtual Web Framework is Virtual Time.

2.1. Virtual Time

VWF provides a synchronized collaborative 3D environment for the web browser (Smith, David A. 2012). Continuing the Open Croquet research effort, VWF allows easy application creation,

and provides a simple interface which allows multiple users to interact with the state of the application. That application is synchronized across clients using the notion of virtual time. A VWF application is made up of prototype components, which are programmed in JavaScript, which allows a shared code and behaviors used in distributed computation, to be modified at run time. The VWF is based on a deterministic computation model. It allows to create a replicated computation model that basically ensures that multiple participants can view and interact with a complex virtual world that maintains identical evolving state, no matter which participant is viewing or how they are interacting. Actually, there is no server and web application is distributed across a network. Furthermore, there is no central server state, as the role of the server in Virtual Time is to simply time-stamp and forward events from the users. This means that each user maintains their own world state, which is guaranteed to be identical to all other users. It is a replicated computation model driven by external events; hence, even complex simulations will run identically. The architecture of application is fully decentralized. The application could evolve and hold internal simulation without propagating essential network traffic. Only a reflector, or reflection server is an extremely simple server. Its main role is to receive events generated by participants interacting with a virtual world, add an event number and a time stamp to it, and "reflect" the event back to the full list of participants in that virtual world. A reflector could be located anywhere in the network (Figure 2).

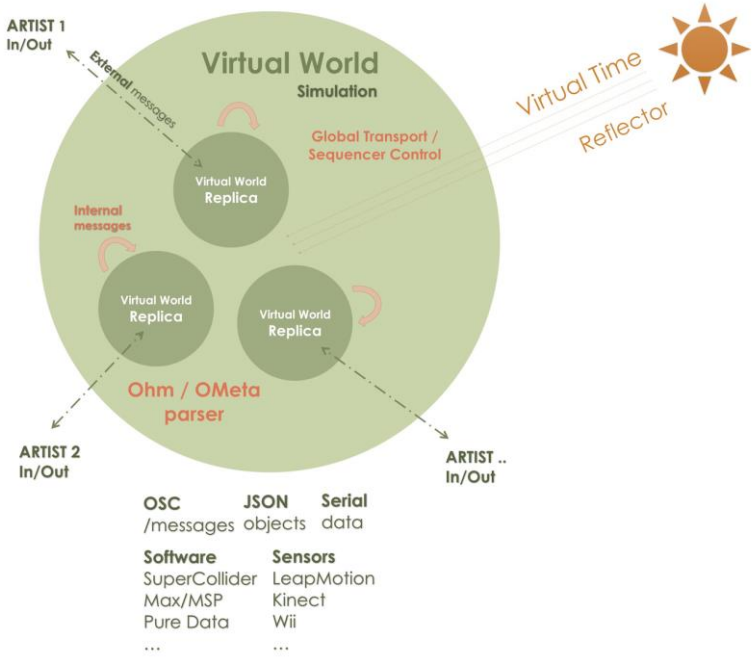

Figure 2. Schematic view of Virtual World based artist environment with Virtual Time

2.2. Meta Language and Live Coding

While the default programming language in Web Browser is JavaScript, we introduced the possibility of creation DSL (domain specific language) inside Virtual World. Any created user-defined programming languages can hold up interactions inside a virtual world. We use Ohm - a new object-oriented language for pattern matching on top of JavaScript (Warth, A. and Dubroy, P. 2016). It is based on a variant of Parsing Expression Grammars (PEGs), which have been extended to handle arbitrary data types. Its general-purpose pattern matching facilities provide a natural and convenient way for end-users and programmers to implement tokenizers, parsers, visitors, and tree transformers. In our prototype we built the Ohm/OMeta driver for VWF

(Suslov, N. and Soshenina, T. 2015 and Suslov, N. 2014), that allows to use Ohm language for parsing internal and external OSC (open sound control) messages in Virtual World with several participants.

2.3. Decentralized Storage and P2P identities

The prototype uses Gun DB as a storage system for all Virtual World's components (prototypes, save states, scripts and user inventories). Everything is stored in pure decentralized DB, thus on client's devices (not on the server). Central storage is used only once for bootstrapping the initial state of DB and caching

For user authentication we use the same Gun DB SEA (Security, Encryption, Authorization) framework, which allows to have P2P identities, instead server-based authorization. That's is critical for spontaneous art installations, exhibitions or educational scenarios, where participants should distinct and be identified in distributed simulated environment.

3. THE PROTOTYPE

For trying out the proposed ideas we developed the prototype, that shows an immersive WebXR / VR environment for collaborative live performances: https://LiveCoding.space. This environment is developed using Virtual World Framework, A-Frame, OSC JS, Cell JS, Gun DB and Ohm language for creating user-defined languages just inside Virtual World. The prototype consists of a several model and view drivers for VWF, that provides basic support for using A-Frame components inside Virtual World Framework applications. That allows participants to build a VWF collaborative apps with 3D visualization, WebVR, HMD, trackers and mobile devices support easily. Figure 3 shows the interaction within collaborative Virtual World Framework app, which is composed by the A-Frame components.

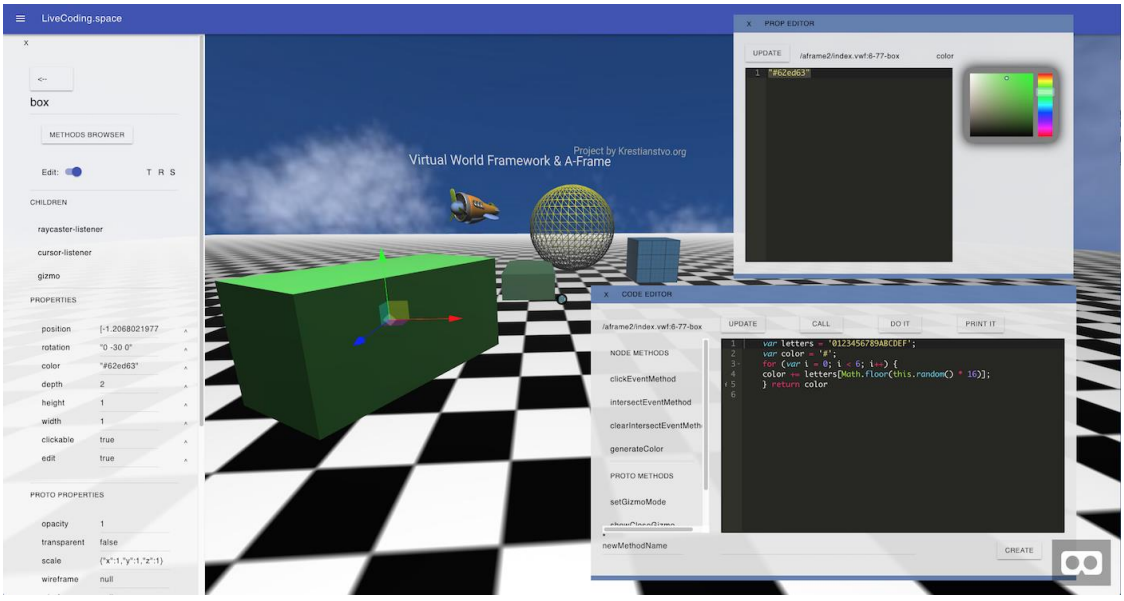

Figure 3. Screenshot of the LiveCoding.space Web 2D editor interface for code editing and inspector

Every connected browser shows the replicated A-Frame components after participant's edition. The users are represented with avatars and are visible to each other. The prototype allows participants to define their own scripting languages at runtime mode. Then they attach those scripts to the Virtual World's objects with these languages inside Web browser. Finally, they can use their new created domain specific languages as main programming languages for a live coding performance.

Adding to the prototype OSC JS library support, we transform Virtual World into the artist-aware, zero install immersive virtual environment for collaborative live performances (Suslov, N. 2016). An artist can easily create tools, own-domain specific languages, parsers for OSC messages and make simulations inside highly distributed computational environment. Several running browsers or desktop versions with running prototypes define the replicated state of the whole computation.

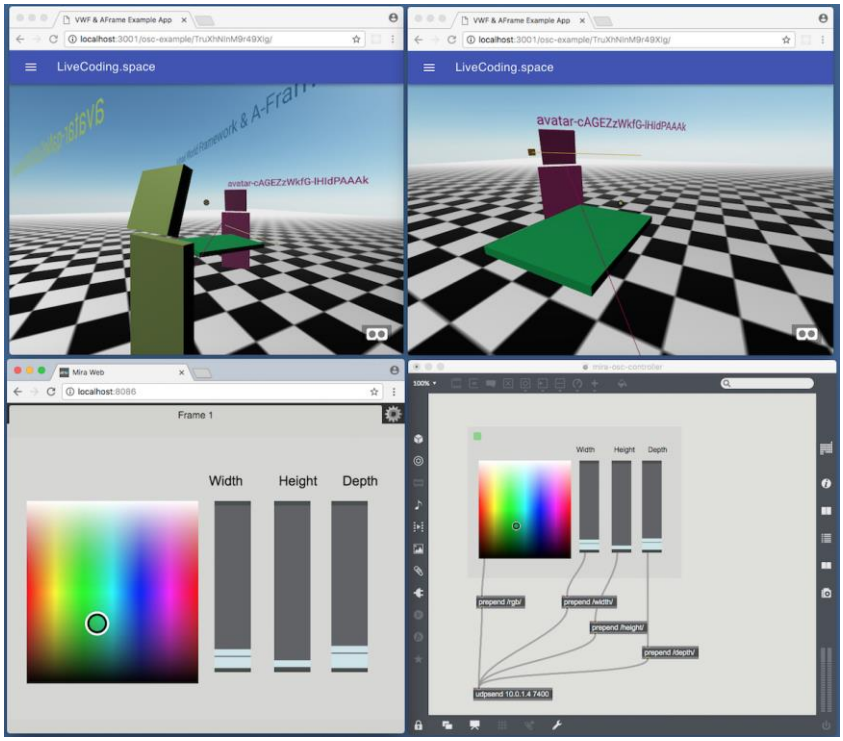

Figure 4. Two users control the Virtual World from Mira Web app by Cycling'74

Any modification in a source code is replicated immediately to all instances of it. Adding new participant nodes is done just by starting the new browsers and connecting them to the already running virtual world instance. Using an avatar, an artist interacts with the virtual world and adjusts the properties of its virtual content.

Artists share exactly their online activities and computation within a united simulation space (Suslov, N. 2012.). They could interact with virtual world's content by using real physical objects as controllers, sending OSC messages (Figure 4). The virtual world's architecture takes everything on a distributed computation. Artists do not need to think about an underlying software program architecture, while preparing their content inside a virtual environment on a Web browser.

An audience can participate in an experiment with any device: from mobile phone, laptop, Gear VR, Mixed-reality HMD it come with. This is ideal for a collaborative electronic orchestra scenario, where a distributed network of the devices of the of participants will simulate electronic instruments and controllers. Some participants can use their devices only as controllers for controlling devices of each other's. Also, every participant of collaboration can share its avatar's vision to any other participant trough virtual camera or by using WebRTC for video/audio streaming with 3D positional audio. Art installations, that are based on multi-window or multi-

monitor/multi-machine setups with view offset cameras can be scaled up to unlimited nodes using virtual world approach.

4. CONCLUSIONS

While developing WebXR / VR collaborative applications using modern software architectures a programmer or an artist still think in terms of libraries and frameworks, tools, operation system, network etc. This repeats the same life-cycle of an application creation for a desktop. Considering Virtual Worlds as the base paradigm for developing the modern Web applications will hugely simplify things. Live coding environment replaces a debugger, virtual time encapsulates network related stuff, Avatar mediates human-computer interaction, Meta-languages negate the knowing of the programming language for users. Nearly every Web application being built with Virtual World paradigm will be ready for the Virtual Reality/Augmented reality/Mixed Reality application scenarios. LiveCoding.space prototype demonstrates how WebXR / VR technologies could exist in a pure-decentralized P2P collaborative, live programming environment, satisfying the needs of an artist on performing, live coding and interacting with an audience.

Acknowledgments

I would like to express thanks for the valuable insights that Victor Suslov, Tatiana Soshenina, Sergey Serkov, and to all others, who have helped in the realization of the prototype, described in this paper

REFERENCES

Smith, D. A. 2012 "Virtual world framework". url:https://en.wikipedia.org/wiki/Virtual_world_framework

Suslov, N. 2014. "Virtual World Framework and OMeta - collaborative programming of distributed objects with user defined languages". *In Proceedings of the SPLASH'2014 Future Programming Workshop*, url: http://www.future-programming.org/2014/program.html

Suslov, N. 2016 "Artist-aware, zero install immersive virtual environment for collaborative live performances", *In Proceedings of the Third International Conference on Live Interfaces (ICLI 2016)*, pages 149-153, University of Sussex, UK

Suslov, N. 2012. "Krestianstvo SDK Towards End-user Mobile 3D Virtual Learning Environment". *In: Proceedings of the Conference on Creating, Connecting and Collaborating through Computing (C5) (2012),* pages 9–14. doi: 10.1109/C5.2012.8

Suslov, N. and Soshenina, T 2015. "From Live Coding to Virtual Being". *In: Proceedings of the First International Conference on Live Coding (2015).* doi: 10.5281/zenodo.19353.

Warth, A. and Dubroy, P. and Garnock-Jones, T. 2016. "Modular semantic actions", *DLS 2016 Proceedings of the 12th Symposium on Dynamic Languages*, pages 108-119, doi>10.1145/2989225.2989231.