Artist-aware, zero install immersive virtual environment for collaborative live performances

Nikolai Suslov

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

Abstract. The prototype of immersive virtual environment for collaborative live performances based on Krestianstvo SDK will be shown. This environment is developed using Virtual World Framework (VWF) and ADL Sandbox project. The prototype will show how the Virtual World Framework being extended with Ohm (OMeta) language a new object-oriented language for pattern matching and OSC support could be turned into the artist-aware zero install collaborative virtual environment. An artist could easily create tools, own domain specific languages, parsers for OSC messages and simulations inside highly distributed computational environment. Several running browsers or desktop versions with running prototypes define the replicated state of the whole computation. 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. They could interact with virtual world's content by using real physical objects sending OSC messages.

Keywords: virtual world, human-computer Interaction, performance art

Introduction

Let's look at today's software paradigms, which are available to a modern artist. In simple words these would be: "Tools", "Programming languages" and "Simulations". Actually, none of them are artist-aware by itself. And only the fundamental interconnection of all of them may leads to the new entity, which we could name "Virtual Environment", propagating itself in an artistic conformable immersive form.

Tools

Why not "Tools"? The best case of using tools in an art context is when an artist could modify the tool implementation at runtime. And there is no distinction between development and runtime modes. For example, the tools programmed using dynamic software like Pure Data, Max/MSP, VVVV or SuperCollider include such mode by default, if an artist distributes the tool with the interactive development environment. But, these tools still stay isolated each of other or grow into a huge complex one interconnected engineering tool. Mostly only experts can supply and support such tools. Sometimes there is no any possibility for reconfiguration or adaptation of them by a people from other disciplines, like artists. In other words, such systems provide a static hardware and software templates for filling them up with the artistic generated content. That forces an end-user to create artistic scenarios in a form of dynamic data using a static computational paradigm. Finally, these systems are used like static viewers with rich human-computer interactive interfaces. The ability of making changes of a tool at runtime leads artists to use more or less "Programming languages" in a form of live coding in common.

Programming languages

Why not "Programming languages"? As with tools the best case of using programming languages, will be programming by an artist his own domain specific language (DSL). In other case, an artist forced to learn a general purpose programming language with a set of art-specific libs and APIs. For example, highly dynamic Self and Smalltalk programming languages are considered to be the simplest languages with the best self-explorative environments. But

even they are about prototype and object oriented paradigms. And they disclosed their features only in using reflectivity, homoiconicity and meta properties while experimenting with constructing the new languages.

Simulations

So, the last thing “Simulations” throws an artist into the paradigm of simulating, modelling and gaming. Simulation behaves like an artefact closed in itself and introduces a biological inspired propagating mechanism. An artist feels more comfortable performing inside simulation, as this paradigm completely blurs the border between development and runtime modes in software or software at all.

Virtual Environment

Things would change if we look at “Tools”, “Programming languages” and “Simulations” as self-exploratory elements of the virtual worlds and the computation-centric live programming environments. The 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 (Figure 1). The user also is represented in a virtual world as a computational process, an object known as avatar/software agent. Moreover, user-defined programming languages hold up all interactions inside a virtual world in the form of live programming. A full-body immersion environment, which is built using virtual worlds, scales conformal up to the unlimited number of hardware and software nodes.

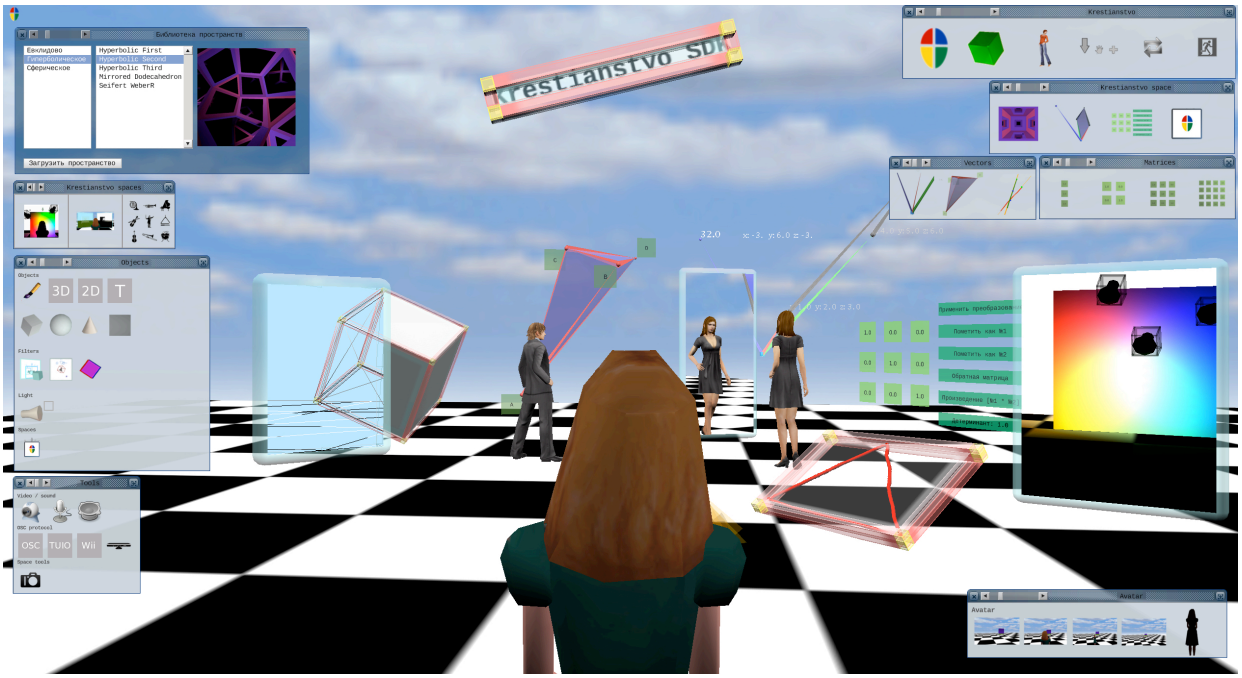

Figure 1. Virtual learning environment Krestianstvo SDK

Virtual worlds provide all-in-one solution and the ability for the programming code to be delivered in a lightweight manner through the network. They support easy synchronisation for many users to interact with common objects and environments. These objects can be programmed on quite different languages. In addition, they can coexist alongside with each other in the same replicated virtual world and holding the same simulation.

Prototype demonstration

The prototype will show an immersive virtual environment for collaborative live performances based on Krestianstvo SDK. This environment is developed using Virtual World Framework (VWF) and ADL Sandbox project. VWF provides a synchronised collaborative 3D environment for the web browser. Continuing the Open Croquet research effort, VWF allows easy application creation, and provides a simple interface to allow multiple users to interact with the state of the application that is synchronised across clients, using the notion of virtual time (Smith 2003). A VWF application is made up of prototype components, which are programmed in JavaScript, which allows a shared code and behaviours used in distributed computation, to be modified at runtime.

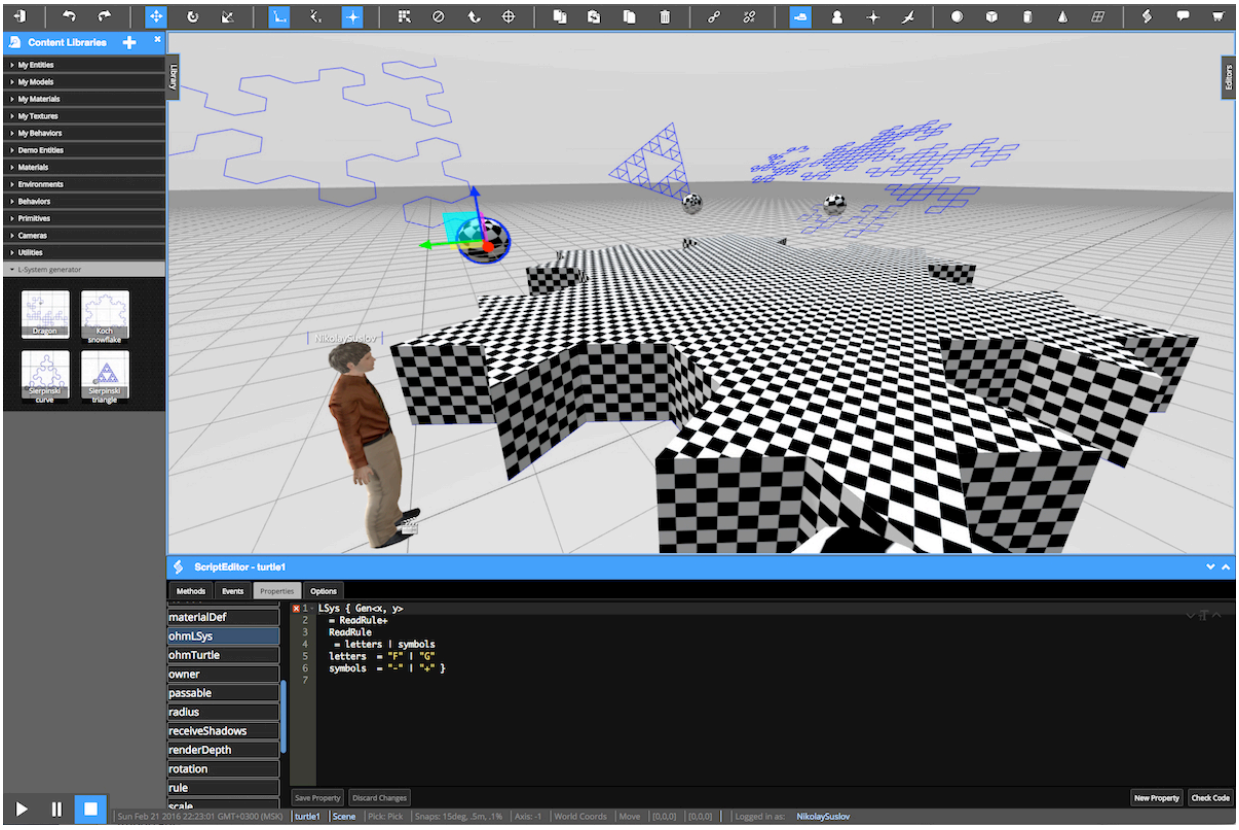

Figure 2. Virtual environment in web browser

The prototype shows how the virtual world framework being extended with Ohm (OMeta) language (Suslov 2015) a new object-oriented language for pattern matching (Warth 2007) and OSC support could be turned into the artist-aware, zero install immersive virtual environment for collaborative live performances. An artist could easily create tools, own-domain specific languages, parsers for OSC messages and simulations inside highly distributed computational environment (Figure 2). Several running browsers or desktop versions with running prototypes define the replicated state of the whole computation (Figure 3). Any modification in a source code is replicated immediately to all instances of it. Adding new nodes is done just by starting the new browsers and connecting them to the already running virtual world. 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 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 virtual environment.

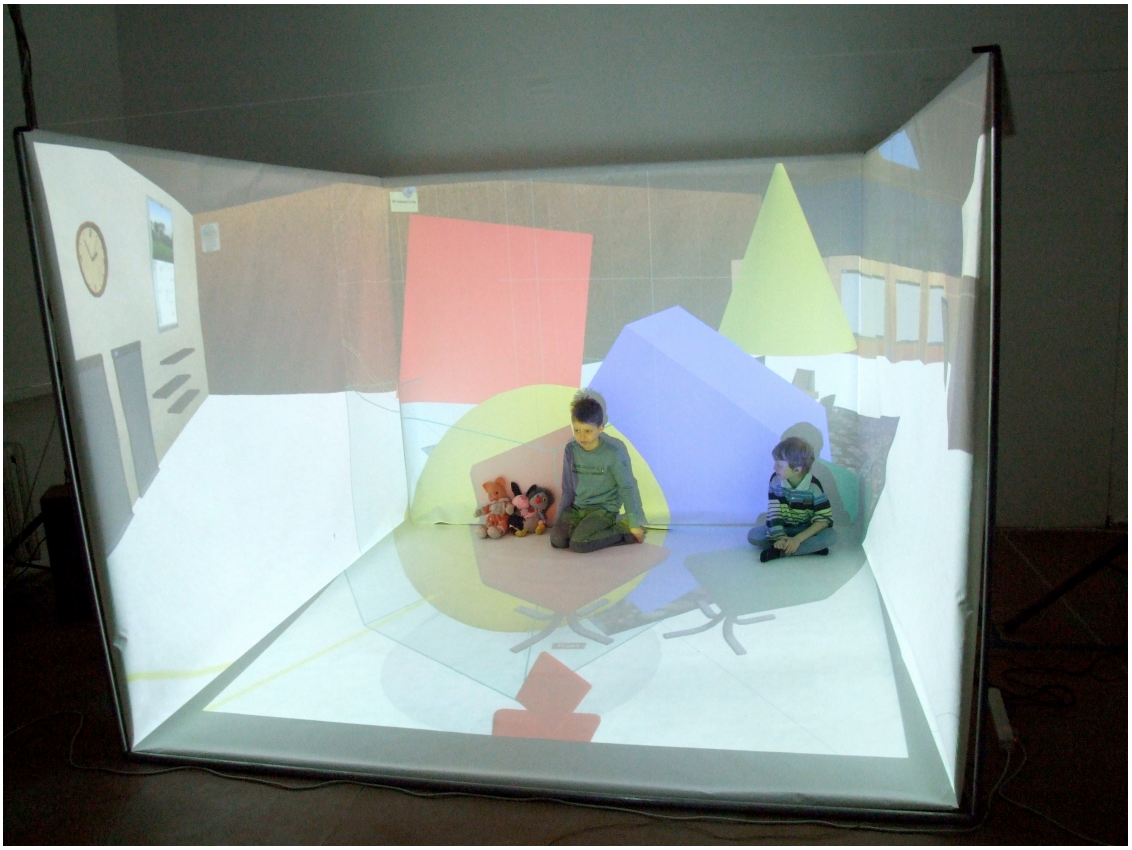

Figure 3. The CAVE prototype using desktop version of virtual environment

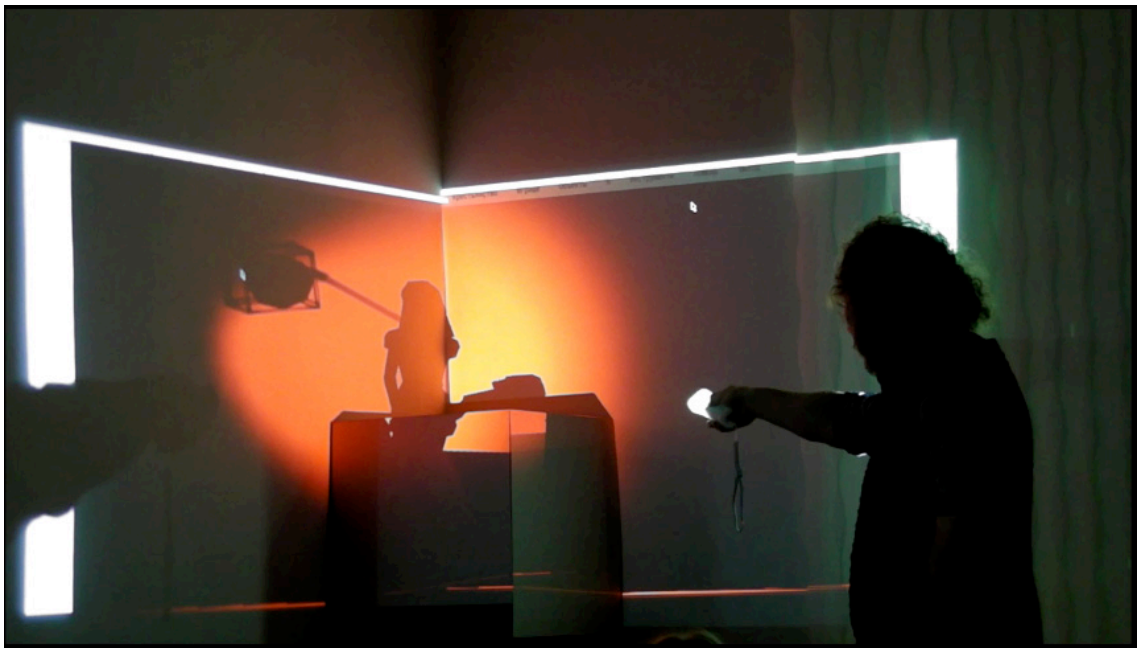

Figure 4. An experiment with a virtual light source controlling with Wii remote

Acknowledgements. 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., Kay, A., Raab, A., and Reed, D. P. 2003. Croquet — A Collaboration System Architecture. In Proceedings of the First Conference on Creating, Connecting, and Collaborating through Computing (C5' 03), pages 2–9. IEEE CS.

Warth, A. and Piumarta, I. 2007. OMeta: an Object-Oriented Language for Pattern-Matching. In OOPSLA '07: Companion to the 22nd ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications, New York, NY, USA. ACM Press.

Suslov, Nikolai and Soshenina, Tatiana. "From Live Coding to Virtual Being", Proceedings of the First International Conference on Live Coding (ICLC2015), Leeds, UK, 13-15 July 2015, Zenodo.

Suslov, Nikolai. 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, Institute for Creative Technologies, University of Southern California, Playa Vista, California, USA, IEEE, Page(s): 9 - 14