From Live Coding to Virtual Being

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

Tatiana Soshenina
Moscow Institute of Architecture
(State Academy)
Moscow, Russia
TVSoshenina@gmail.com

ABSTRACT

The self-explorative, collaborative environments and virtual worlds are setting up the new standards in software engineering for today. In this, live coding is also required in reviewing as for programmers and as for artists too. The most popular live coding frameworks, even being built by using highly dynamic, reflective languages, still suffer on tight bindings to single-node or client-server architecture, language or platform dependence and third-party tools. That leads to inability nor to develop nor scale to the Internet of things the new created works using live coding. In the paper we introduce the prototype of integration of object-oriented language for pattern matching OMeta onto Virtual World Framework on JavaScript. That integration will allow using the live coding in virtual worlds with user-defined languages. Also we explore the possibilities of a conformal scaling of live coding in the case of augmented reality systems and Internet of things. In summary, the paper describes the efforts being done for involving virtual worlds architecture in live coding process. All prototypes that are described in the paper are available for experimenting with on Krestianstvo SDK open source project: http://www.krestianstvo.org

1. FROM CODER TOOLS TO SELF EXPLORATIVE, COLLABORATIVE ENVIRONMENTS

Fundamentally, live coding is about highly dynamic programming languages, reflectivity, homoiconicity and Meta properties. For example, families of functional languages like Lisp, Scheme, Clojure or pure object-oriented Smalltalk, etc. could be considered as "ready" for live coding. And adding to that languages some rich library of functions for working with network, multimedia (audio, video) and integrated program development environment (IDE) based on the debugger, makes them a complete environment for live coding. The most popular live coding frameworks still share this tools-centric approach, but the current developments in virtual & augmented reality, Internet of Things, robotics etc. shows, that this approach requires some review. The revival and emergence of the self explorative environments and virtual worlds, like Self language environment, Etoys, Scratch, OpenCroquet, LivelyKernel, Virtual World Framework is observed now. This will change the role of the programmer and artist in the process of live coding and also will change the process of live coding itself. See Figure 1 for the prototype of controlling several instances of SuperCollider using self explorative, collaborative Open Croquet based virtual world: Krestianstvo SDK.

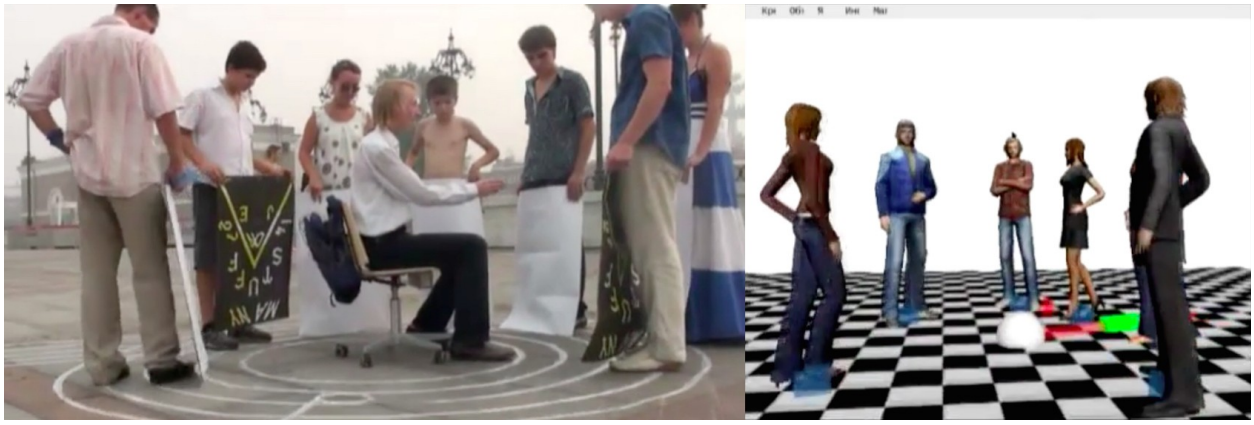

Figure 1. Installation Man'J (Soshenina 2010)

2. LIVE CODING IN VIRTUAL WORLDS WITH USER DEFINED LANGUAGES

We want to introduce the prototype of integration of object-oriented language for pattern matching OMeta on to Virtual World Framework. The integration will allow to define on any VWF component it's own language grammar and replicate it through the application instances, then have a running scripts based on that shared grammar for that component. For example, one could have all the languages down from Logo (Turtle graphics) to Lisp, Smalltalk, even SuperCollider, available for scripting the virtual world just in the Web browser in real-time, see Figure 2.

The Virtual World Framework (VWF) provides a synchronized 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 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 runtime (Virtual World Framework 2015). OMeta is a new object-oriented language for pattern matching. It is based on a variant of Parsing Expression Grammars (PEGs), which have been extended to handle arbitrary data types. OMeta's general-purpose pattern matching facilities provide a natural and convenient way for programmers to implement tokenizers, parsers, visitors, and tree transformers (Warth 2007). We use ADL Sandbox project as a real-world application, which is built on top of VWF.

Figure 2. Integration of OMeta onto Virtual World Framework

To realize that integration we implemented the OMeta driver for VWF. The drivers in VWF define the autonomic actions that happen within a system, dividing responsibility and delegating work for each action of the system. These actions include things such as creating or deleting a node, getting or setting a property, calling methods, and firing events. To load the driver we were needed to include it in the files, that respond for VWF booting process.

As an example, see Figure 3 for a simple L-system parser-generator and Turtle-Sphere for drawing it (Suslov 2014). The virtual world holds two objects, one for drawing and another for parsing and generating L-system strings. For that purposes, the last one has two grammars, one for parsing user-defined input string

and another for parsing the generated structure. The rule of the generator is defined recursively in OMeta (Warth 2009). We used an enhanced Script editor inside the virtual world framework for editing the source of the grammar.

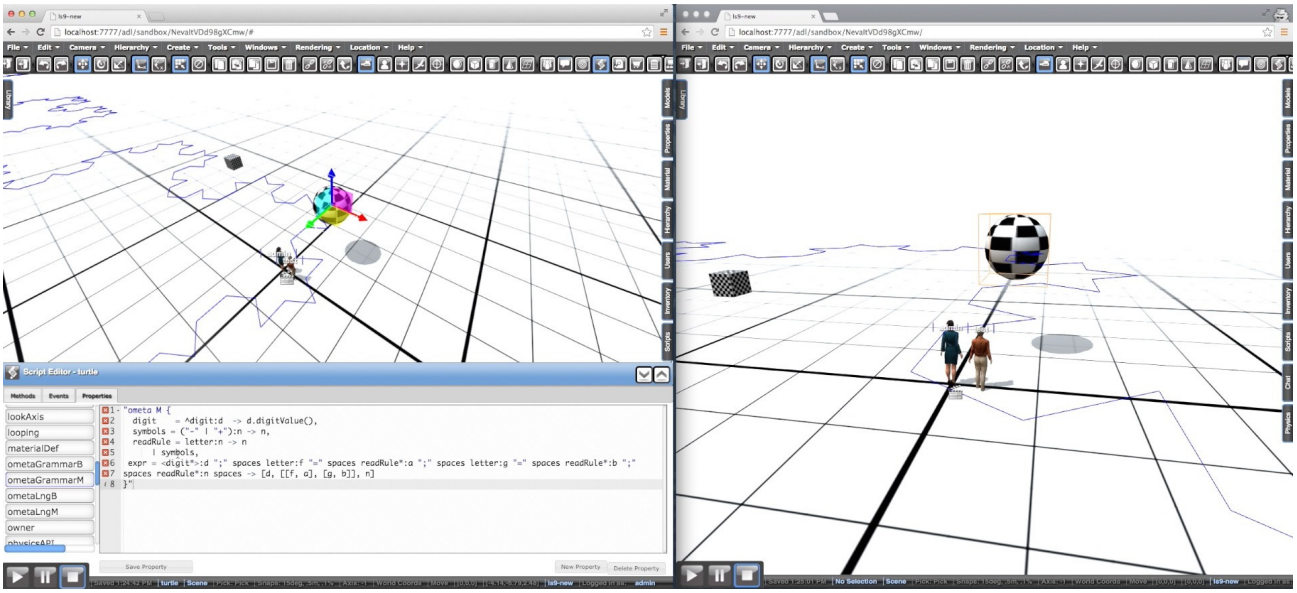

Figure 3. Editing L-system grammar in virtual world

That Script editor is working as Smalltalk browser in self-exploratory environment Squeak (Ingalls 1997). It allows to inspect the properties, methods, and events of the selected VWF component visually and to modify it.

Here is the source code of OMeta parsers of L-System in VWF and sample expression, which is defined by one of the live coding session participant in real time:

```
ometa ParseString {
  digit    = ^digit:d  -> d.digitValue(),
  symbols = ("-" | "+"):n -> n,
  readRule = letter:n -> n
      | symbols,
 expr = <digit*>:d ";" spaces letter:f "=" spaces readRule*:a ";" spaces letter:g "=" spaces readRule*:b ";"
spaces readRule*:n spaces -> [d, [[f, a], [g, b]], n]
}
ometa ParseStructure {
  symbols = ("-" | "+"):n -> n,
  line = (letter | symbols)*,
  rules = [letter [line]]*,
  takeRule:g = [takeRuleFor(g)*:m] -> (vwf.callMethod(nodeID,"flatCollection",[m])),
  takeRuleFor:g = letter:k -> (vwf.callMethod(nodeID,"selectRule",[g, k])) | symbols:n -> [[n]],
  gen 0 [rules] [line]:t -> t,
  gen :n :g takeRule(g):k = gen(n-1, g, k):m -> m
}
tree = LSystem.matchAll('5; F=F-F++F-F; G=; F++F++F', 'expr')
LSystemGen.matchAll(tree,'gen')
```

VWF could contain a lot of simulations running in it. These simulations could be based on data, generated by software, but also the data, that is gotten from any external hardware, like sensors. OMeta language will be a natural solution for parsing that incoming data in user-defined way.

3. CONFORMAL SCALING OF LIVE CODING IN THE CASE OF INTERNET OF THINGS AND AUGMENTED REALITY SYSTEMS

In general, a lot of existed live coding programming languages and their tools are based on single-node or client-server architectures. And these nodes are extended with a huge amount of communication protocol libraries, like Open Sound Control, MIDI, raw packets, user-defined API etc. for working with controllers and external nodes. So, the artist works in form of a "puppeteer", who actually has a quite different approaches in live coding of itself and connected to him nodes. In other words, very often the scaling mechanism from one to many nodes lies at the API and libraries level, instead of the programming language itself. The virtual worlds, which are based on the model of distributed computation, like Open Croquet and Virtual World Framework, solve that problem by generalizing the model of communication, omitting it up to the virtual machine level of the programming language. So, that we could observe a conformal scaling from one node to many, while never using live coding techniques in terms of communication protocols.

Figure 4. Photo of CAVE environment, running by OpenQwaq/Krestianstvo virtual world

We built the prototype of CAVE (cave automatic virtual environment), which is made of six nodes (personal computers), see Figure 4. CAVE application is based on our modified version of the OpenQwaq/Open Croquet virtual world development kit: Krestianstvo SDK. Four nodes are used for projecting virtual world onto the cube walls, one node is used for controlling movements in virtual world and the last one is used for setting the parameters of the CAVE. Open Croquet architecture was built for creating replicated virtual worlds, which are distributed over the network. Several running virtual machines and their corresponded images with source code, define the replicated state of the whole computation – Island (Smith 2003). Live coding could be done from any node, as all nodes are equal to each other. So, the coder is working in Smalltalk browser, which could be opened in virtual world also. Any modification in source code is replicated immediately to all instances of one shared Island. While coding, an artist uses well-known

Smalltalk language constructions. Scaling the CAVE with the new nodes is done just by starting the new ones and connecting them to the running virtual world. Using an avatar an artist adjusts the properties of the CAVE system, see Figure 5. The early prototype was used for installation in The State Tretyakov Gallery in Moscow and The National Pushkin Museum in Saint Petersburg for turning one of the exhibition halls into CAVE: "A. Ivanov Biblical sketches" installation.

So the artist gets the full featured connected virtual/real space to manipulate with. For example: it is easy to model augmented lightning for the physical room. The artist could freely experiment with the transitions and effects of a light's source, move it in the virtual space, while observing the visual effect on the real room's walls. More over, one could experiment with geometry transformations, like dividing, adding new virtual content, while projecting his distributed virtual model just onto physical space in real-time (Suslov 2012). One could use Microsoft Kinect, DMX controllers, TUIO markers, etc. connected right into virtual world.

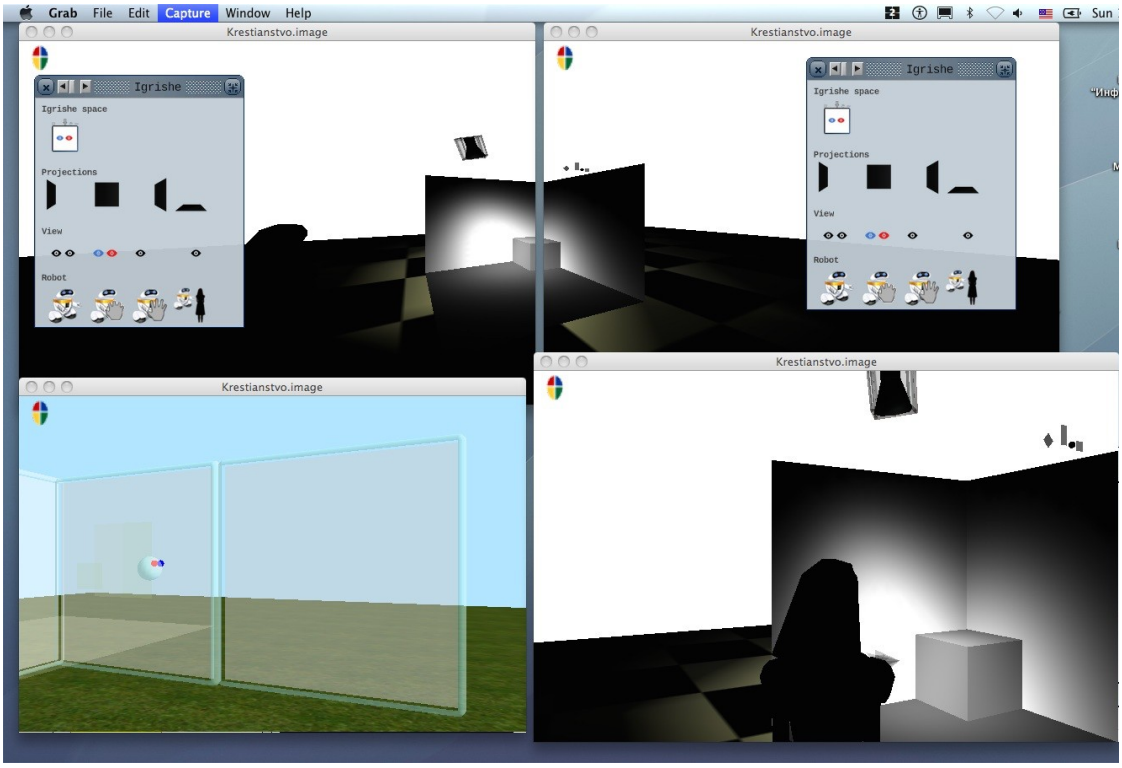

Figure 5. CAVE control virtual world based GUI

4. CONCLUSIONS

The novel VR, gesture, motion detection interfaces and robotics stuff allow the artists to build/run real-time performances/VJ sessions in Theatre, Art gallery and Learning labs, where they get the full featured connected virtual/real space to manipulate with (augmented reality). But, live coding in such cases become more and more complex task, due to heterogeneous nature of Internet of things. Existing live coding environments are continuing building up to the set of extensions and tools for artists-programmers. On the contrary, the virtual worlds have already started to provide the ability for code to be delivered in a lightweight manner and provide easy synchronization for multiple users to interact with common objects and environments. The Virtual World Framework is an example, which will help to interface different multimedia content, simulations, objects, users and locations; which will extend and expand the scope of live coding process and move it towards Virtual Being.

The integration of Virtual World Framework and OMeta makes possible a collaborative live coding on distributed objects with user-defined grammars. These objects could exist alongside each other in the same replicated virtual world, being programmed on quite different languages, but holding the same simulation. Also the integration allows researchers and programmers to experiment with language design ideas for

distributed objects in collaborative way. Due to Internet nature of VWF, OMeta being in it is extremely suitable for parsing an online content from hypertext to rest services and open data.

Acknowledgments

We would like to express thanks for the valuable insights that Victor Suslov, Sergey Serkov, Jonathan Edwards, Richard Gabriel, the participants and reviewers of the Future Programming Workshop at SPLASH 2014 have given to us. And to all others, who have helped in the realization of the projects, described in this paper.

REFERENCES

Ingalls, D., Kaehler, T., Maloney, J., Wallace, S., and Kay, A. 1997. Back to the future: the story of Squeak, a practical Smalltalk written in itself. SIGPLAN Notices, 32(10):318–326.

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.

Soshenina, Tatiana. 2010. Man'J, http://architectan.blogspot.ru/2010/08/manj.html, as of 07 March 2015

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

Suslov, Nikolai. 2014 "Virtual World Framework & OMeta: collaborative programming of distributed objects with user defined languages", The Future Programming Workshop at SPLASH 2014, Portland, Oregon, USA, video demo screencast http://vimeo.com/97713576, as of 07 March 2015

Suslov, Nikolai. 2012. CAVE in Krestianstvo SDK 2.0, https://vimeo.com/35907303, as of 07 March 2015

Virtual World Framework. 2015. https://virtual.wf/documentation.html, as of 07 March 2015

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.

Warth, A. 2009. Experimenting with Programming Languages. PhD dissertation, University of California, Los Angeles.