

# Krestianstvo SDK

## Towards End-user Mobile 3D Virtual Learning Environment

Nikolai Suslov  
Department of Computer Science,  
Institute for Educational Development,  
Vologda, Russia  
[SuslovNV@krestianstvo.org](mailto:SuslovNV@krestianstvo.org)

### Abstract

Virtual worlds are setting up new standards in software engineering for building virtual learning environments (VLE) today. OpenQwaq, which is based on OpenCroquet architecture offers to both programmers and domain experts nearly unlimited capabilities for creating novel computer-based simulated environments. But, even being built using a highly dynamic, reflective language and self-exploratory Squeak/Smalltalk IDE, it still suffers from tight bindings to client-server architecture, platform dependence and third-party tools. That leads to unnecessary inflexibility to develop or deploy new or existing virtual worlds on heterogeneous ad hoc networks. The article describes efforts being done for bringing OpenCroquet/OpenQwaq SDK more closer to a mobile VLE development platform. Krestianstvo SDK is proposed as a one-click application, for instantly setting up a VLE in classroom's network, ART installation, educational disc and similar environments with support of augmented reality and collaborative tangible user interfaces.

*virtual worlds, virtual learning environment, augmented reality*

### I. INTRODUCTION

OpenQwaq is one of the most awaited framework by some experts in the virtual world's development domain [1]. There was no any major update to it's predecessor OpenCroquet [2] since 2007. Alongside OpenCobalt has done a lot to the OpenCroquet technology so it has not disappeared at all. OpenQwaq brought new standards for developing virtual worlds finally became available in 2011. Release of OpenQwaq means now anyone can set up their own virtual space or forum and without worrying about the underlying network architecture, create forums, create contents for them, place them on servers through the web and start to collaborate. So, everything looks just fine, but to start user's own server on LAN or WAN someone needs to install Linux, Apache, PHP, MySql and OpenQwaq server itself following certain config rules and avoiding pitfalls. But OpenQwaq is developed on Squeak platform, Smalltalk language, in a self-exploratory environment [3], so why are

all these third-party tools are needed? The answer is that these tools are needed for security reasons and remote application services. However in learning situation at a classroom or in Art gallery during installation or multimedia disc distribution process these features are not needed at all. So, how about having just a one-click OpenQwaq image with a server and a client on it, that could be run as on the Internet or as a local server or just as a client. Imagine that in a few clicks such system could be used by children for setting up a virtual world's server in a classroom's network or by artists easily prototyping interaction scenarios for their creative activities and performances.

### II. SDK FOR MOBILE VIRTUAL LEARNING ENVIRONMENT

Krestianstvo SDK was born in 2006 as a concept of virtual learning environment for mathematics and as collaborative, highly portable, end-user/programmer framework for building-then-exploring rich multimedia discs on art and also as tool for creating real-time art installations and augmented reality projects. Krestianstvo is built on top of OpenCroquet SDK with preloaded packages. Some of the included packages are: Sophie XUL-CSS, Seaside, OMeta, and Krestianstvo itself.

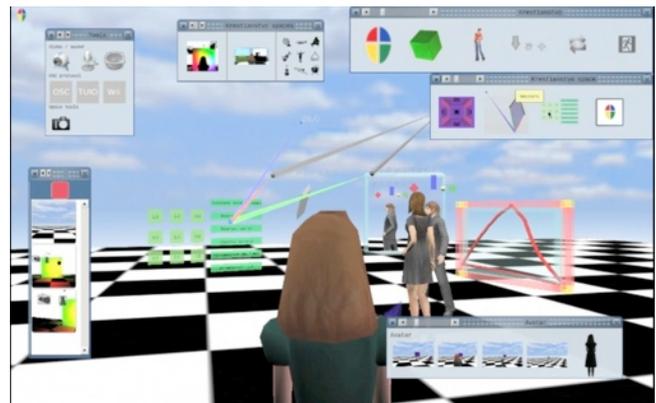


Figure 1 Krestianstvo SDK 3D/2D UI based on Sophie's XUL

Krestianstvo is mainly developed in Russian language using unicode characters in methods and class definitions. The SDK is updated through change sets and the Monticello

update stream, a source code could be easily filed in/filed out containing unicode chars as well. As it is based on Open Croquet, it includes all of its features, but for describing UIs in Krestianstvo, Sophie's XUL logic (CSS/XML/Smalltalk) [4] is used (see Figure 1). The same XUL logic is used for describing the contents of the Krestianstvo 3D spaces. OMeta [5] is expected to be used for creating the user's own languages on describing spaces and for parsing active formulas or scripts, being enclosed in XML tags. Another feature of a mobile VLE is the possibility for interaction with a collaborative space not just with peer computer running Krestianstvo image, but also using mobile web browsers. So, Seaside with Comet technologies were chosen for realization of this functionality [6]. Web-server integration allows multi-user interactions with a shared content on a shared Croquet island just from a web browser in real time, from devices like smart-phones and other low-end machines. Mobile SDK supports a lot of existing physical interfaces available for Squeak/Etoys to interact in OpenCroquet spaces using OSC and Midi protocol (TUIO, Wii, Kinect) [7].



Figure 2 The musical chair is controlled by Krestianstvo SDK and Arduino board.

These interfaces allow artists to build/run realtime performances/VJ sessions in theaters, art-galleries and learning labs, where they get a full featured connected virtual/real space to manipulate with (augmented reality).

Figure 2 shows the process of making music in motion by interacting people with the musical chair in the figure in realtime. The musical chair has built in ultrasonic sensor and is controlled by Arduino board, Krestianstvo SDK and Supercollider. Several such musical chairs could be run in an ad hoc network, organizing real electronic orchestra. Figure 3 shows the prototype of controlling user's Avatar with the Microsoft Kinect sensor and operating on objects by using user's own body and hands.



Figure 3 Microsoft Kinect sensor in Croquet space.

Next feature, which towards to mobile VLE is adding the support of Live Coding on replicated island for changing an object's behavior in a source code on all connected participants at the same time.

In the current version of Krestianstvo SDK 2.0 anybody can easily setup a running server on LAN or WAN (<http://www.krestianstvo.ru> has already run the same server on the Internet). Krestianstvo SDK 2.0 is developed on top of OpenQwaq, so that no part of the original OpenQwaq code is modified from the functional point of view. This allows to run OpenQwaq Forums in pure mode as is. Comparing to OpenQwaq's service provider, Krestianstvo does not need any external database servers to be run. The data is simply stored in plain XML files, emulating MySQL/ODBC database storage logic. A web server, which is used to administer Krestianstvo server is built and run on the same image alongside with a virtual world's service provider. In comparison to OpenQwaq server, which uses an external Apache/PHP web server, these features make Krestianstvo a really mobile platform. A real world example of that can be the online registration and admin web applications running on the Krestianstvo web site.

### III. CASE STUDIES

#### A. Collaborative Curved Space Explorer

Collaborative Curved Space Explorer (CCSE) - is a full-featured multi-user toolbox for exploring a structure of curved spaces in 3D with a support of real-time rendering (Figure 4). Jeff Weeks originally developed and is continuing to develop the application known as Curved Spaces for

exploring and rendering curved spaces in real-time in C programming language [8]. CCSE is actually the port of the C version into Smalltalk and OpenCroquet architecture. The CCSE is based upon work partially supported by the the Russian Foundation For Basic Research under the grant No. 07-0700332.

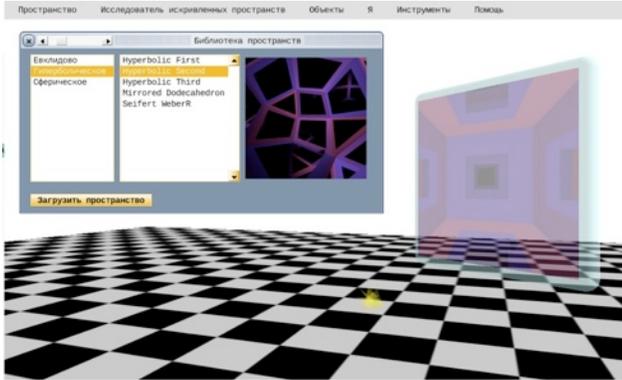


Figure 4 The Collaborative Curved Space Explorer

The application allows to run complex simulations and to achieve a very deep level of collaboration in curved spaces. There is a default library of spaces to explore, but more curved spaces could be generated by using the SnapPea application, suited for creating and studying 3-manifolds. CCSE is programmed in pure Smalltalk and FFI (Foreign Function Interface) is used only for OpenGL part of it. That means that it is highly dynamical and allows to do any changes in the source code just at runtime. Users are flying in a selected space at the same time in a collaborative way, observing their own and the others' spaceships. Figure 5 shows two spaceships, which represents two OpenCroquet avatars, entered into one curved space. Programmatically, space-ships are running on two different machines and are controlled by real users. So, potentially there can be any number of participants, observing self transformations in a curved space at the same time.

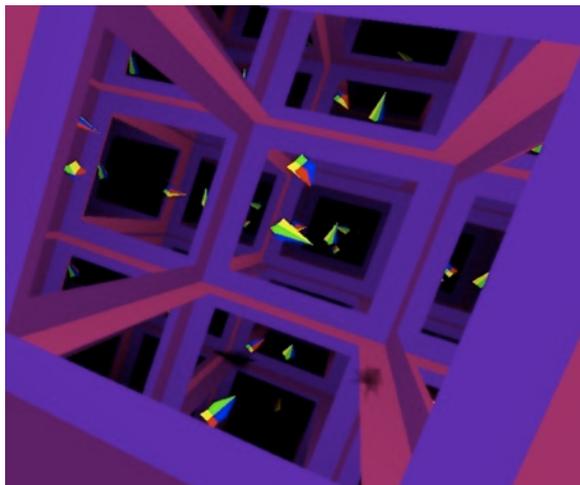


Figure 5 Two users exploring one curved space

### B. Collaborative multi-touch table

The table, which is shown on Figure 6 is run by Krestianstvo SDK virtual space and it's objects being shared on an OpenCroquet island. Several such tables could be organized into a peer-to-peer network and become a real collaborative, interactive interface for a classroom, programmable in Smalltalk. For a recognition process of physical objects reactIVision fiducial markers and TUIO protocol are used [7], based on Simon Holland TUIO for Squeak work. Fiducial markers are unique computer generated symbols printed on a plain paper and are recognized by the reactIVision software. The SuperCollider is used for a music synthesizing and is connected to an OpenCroquet image through OSC protocol, using the idea from SCIMP (SuperCollider Server client for Impromptu) and being programmed in Smalltalk. The main part of this case study is a TUIO protocol and OpenCroquet integration, that allows to develop new tangible interfaces for multi-user interactions within one shared virtual space.



Figure 6 Collaborative multi-touch table

If there is no real multi-touch table, somebody still can interact with a shared OpenCroquet space just by using a standard webcam and controlling a reactIVision instance with printed fiducial markers.



Figure 7 A child's workplace in a classroom

Several classrooms or child's workplaces (Figure 7) can be organized into a united real&virtual collaborative shared

environment. The main difference between existed learning content management systems and such virtual learning environments is that learners share exactly their online activities within a united simulation space and interact with it's content by using real physical objects/controllers. OpenCroquet/Krestianstvo here takes everything on a distributed computation and neither a teacher nor learners do not need to think about an underlying program architecture while preparing to a lesson.

### C. Collaborative Art Installation

Figure 8 shows the prototype of a distributed slideshow application, that can be projected on more then two walls. In the installation a 3D virtual space is mapped onto real space consisting of 3 walls. The installation is rendered in realtime using 3 computers (one per wall) connected to the network with running Krestianstvo SDK image on every computer. Potentially this case study is scalable to any number of walls/projections.

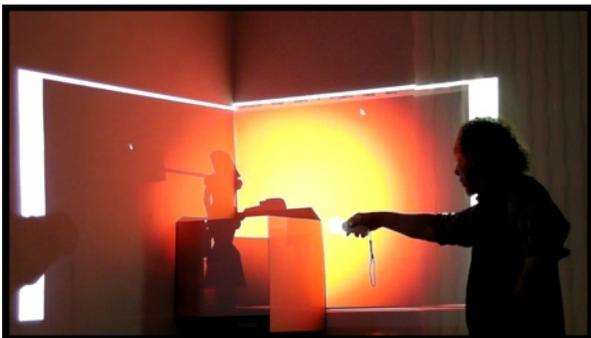


Figure 8 Collaborative Art Installation

An artist gets a full featured connected virtual/real space to manipulate. For example: it is easy to model augmented lighting for a physical room, experiment with the transitions and effects of a light's source, move lights in the virtual space, while observing a visual effect on a real room's walls. More over, an artist can experiment with geometry

transformations like dividing and adding new virtual architectural content, while projecting his distributed virtual model onto physical space in realtime.

### D. A multi-user shared remote web controller

The main aim of the next case study is to check how users can interact with a VLE's shared content using only a web browser. Figure 9 shows how two participants change simultaneously color of the space. In this example only one participant is running Seaside/Comet application alongside with an OpenCroquet's OpenGL space. Any changes happens on RGB sliders (in one of the browser's window) causes the active space's color to change. Comet technology drives all opened Web browsers to handle RGB sliders and to change their visual state automatically (without manual refreshing of a browser window).

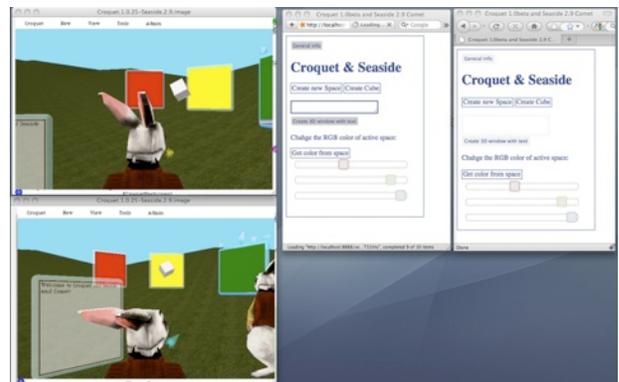


Figure 9 Comet based RGB sliders control the color of 3D space

All callbacks from a Seaside application to an OpenCroquet island are future events, so they do not destroy the replicated state.

```
onSlide: (html jQuery ajax
callback: [ :value |
aColor = #red ifTrue:[self class valueRGB red: value
asNumber.
aFrame future color: (Color r: self class valueRGB red
g: aFrame color green b: aFrame color blue)].
```

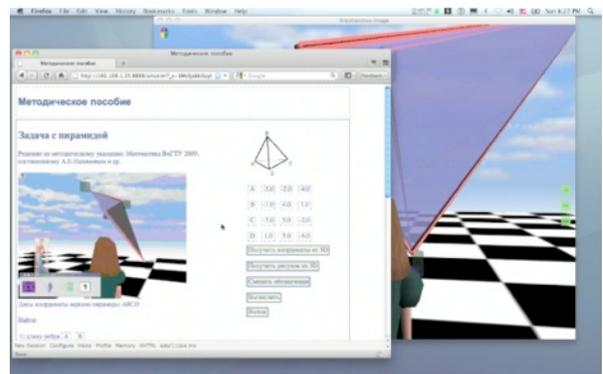


Figure 10 The same electronic notebook on Web and OpenGL

The first scenario of applying a multi-user shared remote web controller in a real situation is connected with a classroom or exhibition/art gallery hall, where we could have several projections of OpenCroquet 3D (OpenGL based) space on the walls, to be controlled by visitors, so an audience can easily interact with a shared content by using their own web based devices they come in with. Thus, they can observe both as a web control panel changes on web-devices and as real OpenCroquet OpenGL graphics on the walls.

The second scenario could be like "blind blogging, surfing". A user interacts with a shared island and gets a callback just with an html information on the same device, but knowing, that his real OpenCroquet world is running somewhere. But to get a full experience someone needs to see both an OpenCroquet OpenGL space and it's web representation (Figure 10). Such Seaside/Comet applications for OpenCroquet could be seemed like multi-user shared remote web controllers for manipulating the OpenCroquet island content. Several opened web browsers connected to a one image/one harness are not equal to several OpenCroquet's avatars, where each should be presented by it's own harness. These remote web controllers are equal more to a one avatar, which is shared between all real users, like multi-touch interface over the web.

### E. Collaborative live coding

OpenCroquet is built on top of the Squeak self-exploratory environment, where everything could be modified/inspected in real time. This means that an environment is suitable for a live coding and interactive programming in Smalltalk.



Figure 11 The Cube of Virtual Reality running by four Krestianstvo images.

Everything is normal when there is just one computer, which is used by only one running vm/image. But, OpenCroquet was built for creating replicated virtual worlds, which are distributed over the network and several vm's/images running at the same time, defining a replicated state of an Island. For maintaining an identity of all replicas on a shared Island, nobody from participants can change anything in the code on their local images. Participants are allowed

just to modify parameters, that had been programmed earlier or just to replace the whole image, but they are not allowed to program on the Island in all replicas at the same time. For example, it is often needed to modify a source code without stopping all participants, when there are several running images define an Island. (Figure 11). Or when several programmers want to code an Island's content behavior at the same time, forming a one image, but working at different places.

So, as proof of "live coding" concept we can do the following in OpenCroquet:

1. Create the new TObject or TFrame subclass, with instance method like:

```
TLiveCode>> compileMethod: aCodeString inClass:
aClassName
```

```
aClassName compile: aCodeString.
```

2. Add this object into the existed island (or register during initialization):

```
liveCodeFrame := self harness activeIsland future new:
TLiveCode.
liveCodeFrame future registerGlobal: #liveCoding.
```

3. Then add instance method to the used harness class, like:

```
CroquetHarness>> makeMethodInFuture: aCodeString
inClass: aClassName
```

```
| liveCodeFrame |
```

```
liveCodeFrame := self activeIsland future at: #liveCode.
liveCodeFrame whenResolved: [
liveCodeFrame futureDo: #compileMethod:inClass: at:0
args: {aCodeString. aClassName}
].
```

Calling this method on any of participants will lead to an equal modification of a source code on all connected Island replicas (Figure 12).

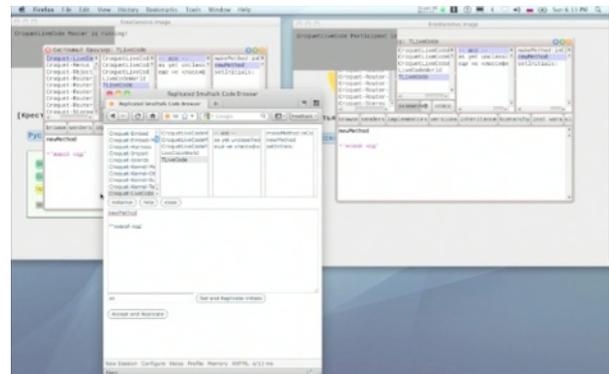


Figure 12 Two images are coded at the same time through a Seaside web application.

#### IV. RELATED WORK

Krestianstvo SDK is inspired by the Squeak one-click image distribution model [9]. Widely known virtual worlds platforms like SecondLife and OpenSim do not have a mobile solution at all. Not to mention their client's distribution forms, their servers require very complicated installation procedures for an end-user. Teleplace and OpenQwaq clients can be distributed as one-click images, but they also require an installation of a virtual world's server somewhere, which is platform and third-party tools dependent and can not be distributed as a one-click image.

#### V. CONCLUSIONS AND FUTURE WORK

The described solution in the form of Krestianstvo SDK for turning OpenCroquet/OpenQwaq SDK into a real mobile platform for running VLEs has shown that this step could be done today, using existed tools and languages. But if we look at this problem from an end-user's point of view, such mobile VLE will be still comparable in complexity to a non-mobile VLE. So, these steps are not enough for making it user-centric mobile platform, opposed to a programmer-centric mobile platform. OpenCroquet's current architecture implementation has already done a lot to make distributed programming easier, but there are still a lot of unresolved issues and assumptions. While developing collaborative applications using modern software architectures a programmer thinks in terms of shared/non-shared entities, therefore applications that are prebuilt with a programmer-centric software evaluation strategy in the core, where the programmer defines what should and should not be shared between participants, more over permanently watching for untouchable hard-coded methods that could break distributed computation and finally depriving collaborative applications to evolve after their deployment at run-time and rising the difficulties in end-user development. The future work could be in extending an existed object-oriented/service-oriented architectures used in distributed computing with the notion of perception of an object, in other words, object's perception is an object's 'idea/rendering' of another object communicating with. Perception of an object is one and only non-shared entity in distributed computation. That could be realized by extending OpenCroquet architecture with Worlds [10], which should allow to implement the perception of an object in distributed computation. The proof of concept application using the new approach, could be the interconnection between LivelyKernel/JS [11] and

OpenCroquet/Smalltalk, where LivelyKernel/JS world will be just one variant of rendering of the Croquet's shared island.

#### ACKNOWLEDGMENT

I would like to express thanks for the valuable insights that Michael Rueger, Jeff Weeks, Victor Suslov, Tatiana Soshenina have given me, also to students of the Architecture and Design Department (VSTU) that have helped in the realization of case studies described in this article.

#### REFERENCES

- [1] Teleplace. OpenQwaq, 2011. as of October 6, 2011, <http://www.teleplace.com/products/openqwaq.php>
- [2] D. A. Smith, A. Kay, A. Raab, and D. P. Reed. 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, January 2003.
- [3] D. Ingalls, T. Kaehler, J. Maloney, S. Wallace, and A. Kay. Back to the future: the story of Squeak, a practical Smalltalk written in itself. SIGPLAN Notices, 32(10):318–326, 1997.
- [4] M. Ruger, B. Stein, and D. Visel, “Sophie—The Future of Reading,” in Proceedings of the Sixth International Conference on Creating, Connecting and Collaborating through Computing (C5 '08), R. Kadobayashi, H. Kita, and R. McGeer, Eds. IEEE, 2008
- [5] A. Warth and I. Piumarta. 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, 2007. ACM Press.
- [6] S. Ducasse, A. Lienhard, and L. Renggli. Seaside: A flexible environment for building dynamic web applications. IEEE Software, 24(5):56–63, 2007.
- [7] Kaltenbrunner, Martin, Bovermann Till, Bencina Ross, and Costanza Enrico. TUIO: A Protocol for Table-Top Tangible User Interfaces. Conference Paper. Gesture Workshop, 2005
- [8] Jeff Weeks. Real-Time Rendering in Curved Spaces. Journal IEEE Computer Graphics and Applications, Volume 22 Issue 6, November 2002, Computer Society Press Los Alamitos, CA, USA
- [9] Squeak.org, 2011 <http://www.squeak.org/Documentation/Installation>
- [10] A. Warth. Experimenting with Programming Languages. PhD dissertation, University of California, Los Angeles, 2009.
- [11] Dan Ingalls. “The Lively Kernel: just for fun, let's take JavaScript seriously.” In DLS '08: Proceedings of the 2008 Dynamic Languages Symposium, pp. 1–1, New York, NY, USA, 2008. ACM.