



Embracing the Controller Revolution: An interview with Camille Troillard

by Kenneth Stewart

<http://www.myspace.com/neimo>

<http://www.osculator.net/>

http://www.youtube.com/watch?v=ESDzYYI0__s

I first learned about Camille Troillard and his program OSCulator much like many people do, through YouTube. I saw a video of someone using a remote from the Nintendo Wii gaming system to control Kyma X in many fascinating ways. I wanted to know how this was possible. Through this video I learned of OSCulator's existence. I learned that this piece of software receives data via Bluetooth from the Wiimote and sends the data using the Open Sound Control protocol to the Flame Firewire interface that connects to the Capybara. I soon discovered its many other features like Applescript support and mouse emulation, its support for many other devices like 3DConnexion's Space Navigator mouse or JazzMutant's Lemur, its OSC routing via UDP to many

other programs like MaxMSP, as well as its rich MIDI capabilities. It inspired me to think about new and unlikely interfaces for physical interaction with computers, and my discussion with Camille made it clear that OSCulator has the potential to use these new control surfaces in different and exciting ways.

The following interview with Camille came about because his band Neïmo was going to be coming to Houston as a part of a short American tour of a few southern cities. This is unusual because Neïmo is from Paris and they don't make it out to Texas very often, if ever, so I jumped at the chance to sit down and speak to him in person about his life, his music and his fantastic piece of software, OSCulator.

KS: What's your background in software programming? How did you get into it?

CT: I was studying Mechanics in high school at a preparatory institute. When I finished prep. school I attended the French Institute for Advanced Mechanics in Clermont-Ferrand. This school is in the center of France so I became very bored and restless. I went to a bank, took out a loan and bought a computer. Soon after that, upon a friend's recommendation, I began using Linux rather than Windows. This is how I discovered freeware and coding your own software. I started reading books on programming and I spent countless hours coding in my spare time. After two years at Clermont-Ferrand, I figured out that I wanted to be a professional musician and artist. I left the school after three years because I had an opportunity to start a company with a friend and my band was about to sign a deal with Sony/ATV in France.

Soon, I went back to Paris and founded a software company and I spent my time learning what I needed to know about programming as well as learning about the music industry. This freedom from Clermont gave me the time to learn languages like LISP, Haskell, Smalltalk, and other functional languages. This is the stuff you learn in computer school, not in a school for mechanics.

I was making my living as a programmer much like a gigging musician would, \$50 here, \$100 there. After 3 or 4 years of this, I made enough money in music to get by, but sometimes I had to supplement my income with programming. Since about 2001, after my company went under, my music making took over my programming. Now programming is mostly a hobby, albeit a very serious one. With OSCulator, time is flexible, but with my band I don't have any time to lose.

KS: You began developing OSCulator for the Kyma/Capybara system, but how did you originally come into contact with it?

CT: Back in 2002, my publisher who was working at Sony/ATV told me about BT (Brian Transeau) and how he created music with this device. He gave me a magazine where there was an interview of this producer and a review of the Kyma workstation and I thought that it was really something I should utilize one day, but it was too expensive for me to own at that time. 4 years later, I earned some good money from some music we made for an advertisement and I decided to spend it on a Capybara sound engine. When I received the machine, I was really crazy, spending countless hours exploring everything.

KS: When was it that you decided to expand the controllability of the system with OSCulator?

CT: It was about a year after that, when I saw that many users (of Kyma) were working with a Lemur and the Capybara, but did not really know how to use them together. I felt that using MIDI on the Lemur was not the best use of the precise screen of the Lemur and then I saw that the OSC component of the Lemur could also be used with other devices. So, I thought that adding OSC capabilities to Kyma was a great idea because you get many devices “for free.” After I had Kyma for a few months, I was wondering if there was anything more I could do than programming sounds and writing assembly code. I was told it was not very hard to write assembly code, but I didn’t have a lot of ideas for writing cool algorithms. So I asked Kurt Hebel, if he had some sort of SDK to provide me with, so I could make cool things with the Capybara. He gave me some C files that can send events to the Flame interface through the Firewire connection. At this moment, I had many thoughts but still had no clue about what to do with this code.

6 months later, I had the idea to make some sort of “patching table” with OSC events on one side and Kyma events on the other. This became OSCulator. Soon after I released this tool to the Kyma community, some users asked me if it was possible to send MIDI events, then mouse control, etc. Then, Carla Scaletti asked me if I would be interested in writing a Wiimote driver for OSCulator. It was not really in the spirit of the application because I thought I should restrict the input to OSC events but I soon realized how cool it would be to make music with that remote controller and how cheap it was compared to specialized accelerometers. I think that this was the moment when OSCulator became more interesting in people’s eyes. Strangely enough, I had many users using OSCulator only for the Wiimote capability. Perhaps this is because I spent so much time improving the open source code I got from the DarwiinRemote project. Now I code it differently. Since then, I’ve changed almost everything about the Darwiin code but others in the community have changed it as well. Now I have many more questions from artists that build installations, which really, really pleases me because this type of user was OSCulator’s first audience.

KS: Was the first version of the program just Kyma events with an OSC input?

CT: Yes, I still have OSCulator 1.0 on my computer. (laughs)

KS: What was the reception like from the beginning and how soon after the initial release did people start requesting MIDI and mouse emulation, etc.?

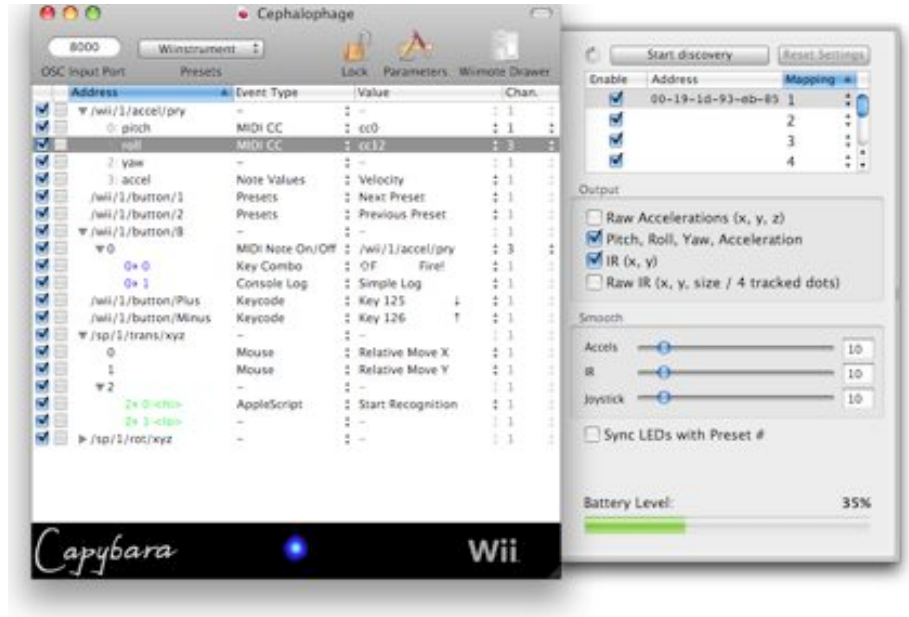
CT: The reception actually took some time. I remember that one user, Edmund Eagan, the Continuum Fingerboard virtuoso, had a lot of interest in my app, and was my first beta tester. He helped a lot in making it happen, as I didn’t have a Lemur to test a real life OSC input device. I’ve got to say that I was a bit disappointed to see that I had only two or three users, but I had to give some time for the community to develop ideas of what they could do with OSCulator. One or two months later, some people were asking how to do this or how to do that, so I thought it would be a good idea to let OSCulator do other things than send events to Kyma. Then it became time to make a website and then, more importantly, time to sell it. You see, for me, earning money from your work is not the most important thing, but if you sell it, you have a responsibility to deliver a high level of quality to the consumer. I thought that if I could sell it, I could work on the program more, and if people were to buy it, I would know exactly how interested users there would be.

Since the beginning it has been a long and smooth progression, and I think it is still progressing, which is perfect for me, since I can’t work full time on the development. Users need time to know that OSCulator exists and I need time to code new features into it. Hopefully they go hand in hand since there is no need to bloat the software by integrating zillions of new features every week. The next step is Version 3, which will be a great challenge I think, because I don’t want to turn it into

a programming language like MaxMSP or SuperCollider.

KS: Since the user community has dictated the development of OSCulator thus far, what do you feel the user community needs from a new version?

CT: The user community has needs from the current version they are using. What I mean by that is that they are using a tool, and when they feel they can't do something with it, they just ask me how possible it would be to have this or that new feature, but their inquiries generally stay within the bounds offered by the version they are using. Well, I should probably replace the word 'version' with 'user interface.' As a designer, I am thinking about different ways of doing the same thing and how easy it could be to extend the possibility of the software with the user interface.



Right now I try to focus on flexibility, but obviously it is not possible to do everything because you are constrained by the user interface, which is something that the users don't really talk about. I also feel that there are some users that would be willing to use OSCulator, but they find it somewhat too complicated for their needs. So my goal for the next version is to provide a new user interface and represent inputs and outputs more graphically but I don't want to go into the "graphic programming" paradigm.

KS: Like MaxMSP or Kyma Prototypes?

CT: Exactly, I think MaxMSP is a very good piece of software, and I am not criticizing at all how it works. I think that OSCulator is more of a controller hub, and should stay like this. For example, if you want to be able to do very simple tasks, it should be very straightforward. And if you want to go deeper into the problem, then Max or Kyma are there to tackle that problem. So the idea for Version 3, is to hide all the complexity for direct interaction and creation of simple solutions, but also to offer a means of exposing the inner workings and send the user to more complex tools for further processing. There is a problem with extensibility as well. For example, the Event Type menu is growing more and more. I would like to add more things, but I always think that I should refrain from making this menu too big. I find it just ridiculous to have a menu that's three pages long.

Also, I think that there are some users that just don't want to create a patch at all, just use what some users have already made. So I am toying with the idea of the possibility to create stand-alone applications from an OSCulator patch. With this, I could build a library of OSCulator "offsprings" that users could directly use. For example, take the idea of controlling a game with the Wiimote. If you are a gaming enthusiast, will you learn about HID and OSC, and all that stuff? You just want to connect the Wiimote and go. Now the gamer, that doesn't know about how it all works, would just launch the application and play with the Wiimote. So someone would create an OSCulator document that patches the Wiimote to the HID system, and perhaps a nifty user interface explaining what the button does, and save it as a stand-alone application.

KS: So, what you are talking about is something like CSS style sheets where you have the same data but different open-ended styles of presentation?

CT: Exactly! I was thinking about how dashboard widgets are made. You have many icons representing the inputs and outputs available where you could connect them together, yielding automatic connections and translations, but if you want

to go into the details, you could “flip” the instances, just like the dashboard widgets allowing access to more properties. All I want is to get rid of the long list of events. This representation is not very expressive, and brings a lot of limitations which, by consequence, impacts the structure of the code. Also, with the new technologies that Leopard provides, I could do some very nice things, graphically speaking. But before Version 3, I would like to add an OSC event recorder, like a very simple OSC sequencer. Again, this problem is not easy to address, and I’ll have to make choices on which features need to be offered, and which should be left off, because I want to keep OSCulator easy to use.

KS: I see. Are you imagining this sequencer on the input or output side?

CT: What I think is that you simply trigger the recording, either from a button on the user interface, or from an input OSC event. Then you could replay this recording with the same idea either from a GUI button or an OSC input. The sequencer would just replay the OSC input it has recorded back into the document that holds the bindings to the outputs. This would make a very simple but very powerful way of recording gestures from OSC input, and replaying them again. I don’t think that the sequences should be editable for the moment, because it brings up too many new questions.

KS: So, no timeline or event list interface for the time being?

CT: Yes, no timeline. I think the first version of the recorder will only be a list of takes. Then I will make these takes editable in future releases. The problem is that the Wiimote, for example, sends lots of events if you use the acceleration controls, so you would end up with a very long list of events. If I were to make an editor, I think I would smooth these events with Bezier curves and display something graphically more synthetic. I am not convinced users would be completely happy with an event list.

KS: What about gesture recognition, like the iPhone or iPod Touch or even tablet programs like Ink, creating a gesture taxonomy or vocabulary.

CT: The ability to record leads to the ability to recognize similar gestures. OSCulator could store a recording and understand that it is a gesture we want to recognize in the future and assign it to trigger new events. This could look at the whole OSC input. You could have data from a WiiMote, Lemur, MaxMSP and something else and an aggregate of all of these OSC controller sources to trigger something as a whole. That would definitely be cool.



Troilliard with his band, Neïmo

2008 ASCAP / SEAMUS AWARDS

We’d like to congratulate this year’s ASCAP SEAMUS Award Winners:

Winner: John Lato
2nd Prize: Jason Bolte
Finalists: Greg Cornelius and Peter Lane

Allen Strange Undergraduate awards were given to Ben Dorfan and Amy Hanson.