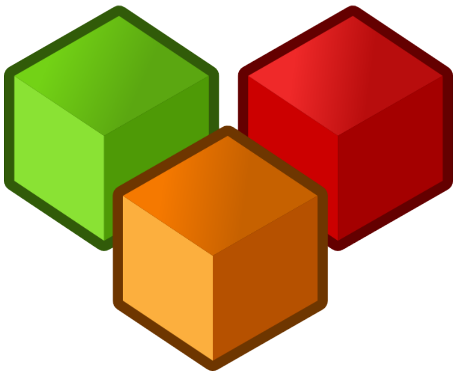
# Cube Orchestra:
A touch interactive musical application



Jordi Hidalgo Gómez
Héctor Parra Rodríguez

Advanced Interface Design
Master in Sound and Music Computing
Universitat Pompeu Fabra
Barcelona - April 15, 2012

## 1 Introduction

The Cube Orchestra is an application developed starting from a previous project. The idea of the previous Cube Orchestra application was an interactive musical application where multiple users can play simultaneously different kind of melodies and rhythms from a mobile device. Each user is creating objects and this objects are visualized through a projection. The interesting point of the application is that several users can enjoy playing music while knowing what they are doing not only by pressing buttons or sliders, but also interacting with the creation and manipulation of cubes.

The aim of the project presented here is to extend the previous application with a touchable interface. A snapshot of the actual application can be seen in figure 1. With the introducing of the new interface, we have introduced also a new role for the users. Now we can distinct between the generator and the manipulator. The generator will be the user in charge of creating the cubes and generate music, as in the past application. The manipulator will be the new role, the user that will be touching directly the objects in the screen and manipulating the generated music. As we see in figure 1 in the superior part there are a slider, which control the bpm of the overall melodies and drum patterns and bar with circumferences. This circumferences are the objects that will represent the effects and can be applied directly to each of the cubes by placing on the top of them. Then another capabilities have been added according to the touchable capabilities and possible gestures accepted by the system.

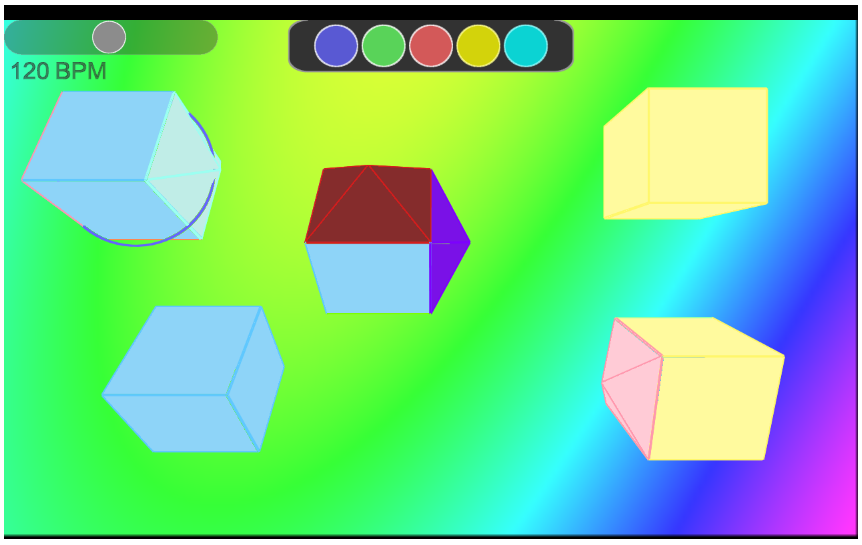


Figure 1: Cube Orchestra application

## 2 Research question

As we said, the system have been developed on the top of the old Cube Orchestra application where the interaction or control is reduced to the use of mobile devices. So we can propose a research question like: *can we take advantage from touch screen and gestures to develop an interactive musical application?*. The objective of this project is to evaluate how the touchable capabilities can improve the previous application. Maybe one of the main problems of the previous interface is that the user was very restricted only to virtual buttons and knobs and this control was mapped to the projected visualization of the objects. Some aspects of this interface were good like could be very specific and precise actions like could be on/off actions. But we observed that actions directly related with the visual object itself were not that good mapped with the mobile device. A very simple example could be move the cube along the surface or projected area, this is not very comfortable with a XY pad, while touching directly the object and move it along the surface becomes more comfortable and usable. Another example could be the selection of the multiple objects in the screen or projected area that becomes also more understandable directly interacting with the cubes.

Starting from the commented research question, we can deduce or think in other problems or aspects that can be evaluated. An initial idea of the Cube Orchestra application was to take profit from the polyhedron objects and the possibilities that offers by manipulating shapes and creating new polyhedrons. This in combination with colors and movement(like rotation or beat synchronization) could help in order to improve the mapping with the created sounds and music.

## 3 Sensors/Actuators

The main goal of our interface is to track a finger pressing the screen, wich will allow to interact with the visual objects and manipulate them. There are several ways to track faces, body or just fingers. One of the most extended frameworks for track images directly from a cam is Open CV[11]. This framework is a very good aproach in order to track specific body parts directly from the cam but is difficult to track a finger and then detect when is pressing. Same happens with kinect, that has a very good precision but still is difficult to determine how we are pressing a single point in a determined area. For this reason we decided to chose the Wiimote as an input sensor. The Wiimote is the controller of the videoconsole Wii, that it has an integrated infrared camera on its top, so it detects infrared points on its vision angle and send the position of every point. The Wiimote IR camera provides high-resolution, high-speed tracking of up to four simultaneous IR light sources. The camera sensor exact specifications are unpublished but it seems to have a resolution of 1204x768 pixels, more than 4 bits of dot size and a 45 degree horizontal filed of light source[10]. Moreover it is relatively cheap, about 30$.

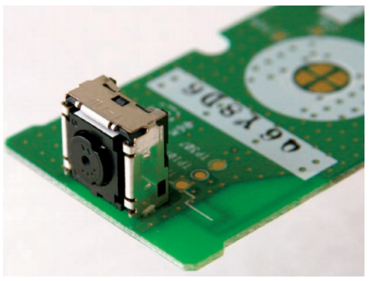


Figure 2: Wiimote IR cam

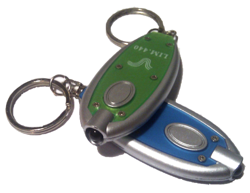


Figure 3: IR LED devices

For our case the use of the Wiimote is perfect in the sense of it tracks a source light point with a very high precision and also it is very useful to simulate when we touch the screen with only turning on/off the light. The drawback here is that we need an emitting light device, which is not that comfortable than only using the fingers and interacting with the projected objects. But we have tried to make a comfortable and non-intrusive devices that can be easily tracked by the Wiimote. These can be seen in figure 3 which only consists on a simple light key chain powered by two 3V batteries where we have replaced the usual white LED by an infrared LED[13].

The Wiimote and the IR emitters are the main sensors of the interface but we can considerate also the mobile device which with the touch screen and the accelerometer it will be controlling some parameters.

## 4 Mapping

For the mapping process we have distincted in two groups, Input Mapping and Output Mapping. Input mapping is referred to the assignation of controls that the user can modify and interact. Then Output mapping is related to the internal mapping between the visual objects and the sound.

### 4.1 Input Mapping

As we mentioned, the there is two types of "users" for the application. The generator and the manipulator. This two users will have two different roles in the process of creating music and also two different ways of using the interfaces. The generator will be creating new cubes with its corresponding melodies or rhythms. This will interact with the mobile device which has the TouchOSC interface[9] as is explained in section 5.1.1. We have tried to map this interface as the usually is made with this kind of interfaces involving on/off/toggle buttons, sliders, knobs etc. So the mapping is simple, some examples could be the volume with the sliders, drum samples with the pads, or effects amount with the knobs. Then we assigned the touchable interface for the manipulator. This user is the one that has the IR leds and interacts directly with the objects that the generator has created. This could be not that usual kind of interface for music like the TouchOSC application, but we tried to map the more natural and intuitive actions according to shapes, movement, position in the screen or even colors of the cubes. So the manipulator will be interacting with the add-on effects and the cubes that will be control some of the parameters of the output sound and visual events. A very simple mapping example is the panning effect, the manipulator will be displacing the object in right/left directions while the sound will be more or less present in one of the two stereo channels according to its position. Another clear example that we found very important in the operation of the application is the mapping of the size of the effects(circumferences on the top of the cubes). The manipulator can increase/decrease the size of the circumferences with the use of the two IR leds by increasing/decreasing the distance of them inside the circumference. This mapping allows control some of the main parameters of the effects.

### 4.2 Output Mapping

For the output mapping we wanted to have visual objects that create sounds, not to have a visualization or projection to complement the sound. This means that most of the actions that we want to do will start in a movement, size or color change of the cubes that then will be mapped with a sonic event. So we defined the system in the way that every time the user interacts with the interface, the mobile or the touchable, the effect will be a visual change and at the same time a change in the sound or music created. A clear example of this could be the rotation of the cubes. We thought that could be interesting to perceive the tempo of the melodies or drum patterns with the rotational velocity of the cubes, so when the manipulator moves the BPM slider, the cubes increase/decrease the rotational velocity which gives the user a direct feedback of how faster or slower the music is being played.

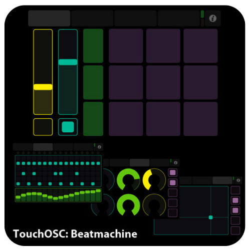


Figure 4: TouchOSC interface

## 5 Implementation

For the implementation of the project we decided to differentiate between different parts, based on their responsibilities, and then connect them together to finally have the complete working system.

### 5.1 Data input

#### 5.1.1 Mobile devices

The application was intended to be a playable instrument were more than one person could play at the same time, so we decided the best approach was to use mobile devices since, nowadays, it is an affordable technology, that most of the people have and that have good sensors in it. After considering different solutions we came to use a mobile application called TouchOSC[9]. This application runs on iOS and

Android devices (the most popular) and has a set of predefined layouts with buttons, sliders and matrices that gave us the control we needed.

TouchOSC communicates with the central computer using OSC[2] messages through the UDP protocol, so we managed to assign functions to the elements of the layout but, moreover, keep track and save the state of the layout on the central computer so the user of the mobile device can switch between different instruments (each instruments has a layout) and its layout is automatically reloaded.

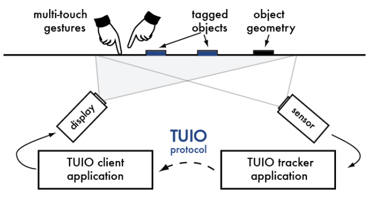


Figure 5: TUIO platform diagram

### 5.1.2 Touchable screen

The touchable screen has been our main feature for the application. We wanted a person controlling the visual output with his fingers. At first we thought about using the technology of Reactivision[5] for tracking the fingers but this had many complications: you need a very good camera, there has to be low ambient light, it is thought to be used horizontally but not really vertically and the tracking is slow (about 20-30 frames/sec).

So, in the end, we found the technology best fitted our requirements was the Wiimote. As we have mentioned in section 3 the Wiimote meets perfectly our requeriments and communicates all the data through Bluetooth, in our case, to the central computer.

In the central computer we ran a software called Wiimote Whiteboard[12]. This software detects the Wiimote and is able to send to any internal port the position of the detected infrared points translated into the TUIO protocol[6]. Wiimote Whiteboard applies a running sum filter to the captured data so it is smoothed and, moreover, it allows using simultaneously a pair of Wiimotes for redundancy and for covering big screens; a particularly useful feature we needed to use to achieve a good touch-like response. There are similar applications to this one like Smoothboard[8] but this is the only one supporting simultaneous points (multitouch) and, in addition, cross-platfrom.

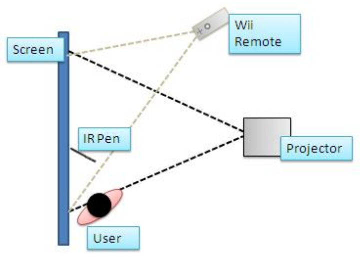


Figure 6: TUIO platform diagram

## 5.2 Data output

### 5.2.1 Sound

For the sound we went directly for Pure Data[4] for a number of reasons. It is free, has a good community with useful effects and instruments to take also for free, it is easily programmable and really good for staring fast projects, and finally, the patch created could be integrated natively in the future to a Java program, a C++ program or even to mobile apps.

So we used two instruments from the community, a drum machine[3] and a synthesizer[14]. Each of them can be played directly or through a sequencer we programmed ourselves. Its parameters can be tweaked and some sound effects can be applied and controlled. Actually, the mobile device's users have total control over the instrument and its parameters and the touchable screen user controls the effects applied to every instrument and its quantity.

The Pure Data application was controlled by the main MT4j application (section 5.2.2) through OSC events. This communication was actually bidirectional since Pure Data also sent back some OSC messages about the tempo and the instruments state.

### 5.2.2 Visualization

We first started the heart of this application with Processing[7] (Java). It was very good to draw visualizations and had nice libraries for OSC communication. But when we integrated the touchable screen we needed something more, and we found MT4j[1] was perfect. It is a Java framework built on top of Processing for the graphics drawing but specially designed for multitouch applications, desktop or mobile. It accepts different inputs like a touch screen of a mobile, multiple mice, windows touch events... but most importantly, it was able to read directly the TUIO events we got from Wiimote Whiteboard.

The good thing about MT4j was it comes with classes for the basic polygons and polyhedrons capable of natively react to multitouch gestures. It means the new things we had to implement were quite straightforward since we only need to map the multitouch gestures of the new graphics to the desired actions/events. The bad part was we had started building our system with Processing, not MT4j, so there was a complex structure (including OSC communication) to adapt to the MT4j gestures handling architecture. We used a base class provided by MT4j to inherit with the basic functions, but a lot of the "touchable" behavior had to be implemented by hand and, sadly, there is still a lack of good documentation on this framework so a lot of code introspection was needed.

## 6 Conclusions

This project had been a rewarding experience since we found what we were searching for: adding a touch screen and its corresponding gestures improve the user experience on interactive musical applications.

There is an essential difference between changing a button/slider status or touching a visual object representation, and it is the internal mapping our mind has to do. While for a button/slider we need to remember its function (in terms of sound) a visual representation it supposed to give a natural perception where one can guess or sense how this is going to affect the system. Certainly, your guess might be wrong initially but it will be easy to see what else it is doing and, more importantly, this behavior can be more easily and quickly integrated in the user. Moreover, a touchable representation can represent complex sound events that the user does not really need to understand but that can actually control.

All of this happens for the simple reason that humans are used to interact with physical objects in every moment, and a touchable screen is closer to this reality than buttons, that are more abstract. Nevertheless, it is very important to build a visual representation that has the intuitiveness mentioned. As an obvious example, if touch an object upwards you clearly expect it to move upwards, not downwards.

We also found it is very important the touchable screen really "feels" touchable. To make the touchable screen the closest experience with our reality we need to make it very smooth and responsive, in other words, the interaction has to feel natural. In our project this implied more work than we expected; it is crucial to find infrared emitters that are good for the Wiimote camera, the placement of the cameras has to be accurately studied so the user does not hide the infrared pens, and, finally, take care that the cameras cover the hole extension of the screen with angle enough to be precise at determining positions.

In conclusion, this project confirms the current tendency of introducing multitouch surfaces for software applications is extendable and also a very good approach for sound generating systems.

## References

[1] Multi touch for java. http://www.mt4j.org/.

[2] Open sound control. http://opensoundcontrol.org/.

[3] Pd drum machine. http://www.nullpointer.co.uk/-/pd.htm.

[4] Pure data. http://puredata.info/.

[5] reactivision. http://reactivision.sourceforge.net/.

[6] Tuio. http://www.tuio.org/.

[7] C. R. Ben Fry. Processing. http://processing.org/.

[8] S. T. Boon Jin. Smoothboard. http://www.smoothboard.net/.

[9] Hexler. Touchosc. http://hexler.net/software/touchosc.

[10] J. Lee. Hacking the nintendo wii remote. *Pervasive Computing, IEEE*, 7(3):39 –45, july-sept. 2008.

[11] OpenCVWiki. Opencv. http://opencv.willowgarage.com/wiki/.

[12] U. Schmidt. Wiimote whiteboard. http://www.uweschmidt.org/wiimote-whiteboard.

[13] V. Semiconductors. High power infrared emitting diode, 940 nm, gaalas/gaas. http://www.vishay.com/docs/81011/tsal6400.pdf.

[14] P. Stone. Polyphonic synthesizers. http://www.pkstonemusic.com/pd_code.html.