# Multi touch support implementation in Pharo

François Lepan
Benjamin Van Ryseghem

29 avril 2013

## Introduction

During our project our main goal was to introduce a new way to handle multi touch events and gestures in Pharo. Starting from what we previously did last semester during the PJE lecture and using the same approach, we introduce a handling of gestures based on TUIO with an architecture allowing to switch to virtual machine events.

In order to analyse these gestures we needed a state machine that would fill the gap between the human gesture and a perfect gesture. This state machine has been improved as we were adding new gestures to fit as much as possible the gestures performed by the user.

Another goal was to clean and reunify the whole hierarchy of system events and to provide a clean abstraction of low level data structure.

# Table des matières

# 1 TUOI and blobs analysis

Our first goal is to be able to retrieve gestures from the user. In order to do we used TUIO which is a protocol for multitouch events. It retrieves the multi-touch gestures from the hardware and then generates events for each finger on the device (*cf* Fig. 1). For our tests we used a software called Tongseng [1] that generates TUIO events from the Mac trackpad and a library created by Simon Holland [2] that parses the TUIO events.

# 2 System events hierarchy

In Pharo, the current implementation of system events [3] is really messy and not object oriented at all. The first task while interpreting system events is to be able to transform the low level array describing an event from the Virtual Machine into a Pharo object.

Currently the object responsible of this transformation is the morph [4] representing the mouse cursor. Moreover, the events are hardcoded in a way reducing the ease of extendability.

Our first task was to take the prototype of new implementation done by Pr. Stéphane Ducasse and Igor Stasenko and to polish it in order to make it work in our scenario. This new infrastructure of events gave us space to easily experiment with system events without breaking the whole system, and space to implement our own events like two fingers rotation.

# 3 State machine limits

Once the events correctly set, we were able to experiment the analysis of events using a state machine. Indeed, we thought it will be great to have a state machine here, allowing us to describe some complex behaviour easily.

We quickly encountered the the problem which took us the more time to fix. In fact, during the implementation of basic gestures (two fingers swipe, pinch and rotate) we figured out we had to specify some constants for being able to distinguish the different cases, without forgetting the acceleration.

For being able to measure our progress, we started to implement unit tests here. As a first step, we wrote some basic tests describing ”perfect gestures”. But soon we decided to have a set of more real cases. To have this real cases being part of unit tests, we wrote a gesture recorder and gesture reader which can be substituted from the system events input reader in order to replay previously recorded gestures. This way we had a set of real gestures that we were able to reproduce again and again in our tests.

Thanks to this, we tried different mathematical (and naïve) approach until we found a set of relations satisfying all the implemented tests.

1. Fajran Iman Rusadi - https ://github.com/fajran/tongseng
2. http ://mcl.open.ac.uk/sh/squeakmusic.html
3. Like MouseMove, KeyPressed, etc.
4. Pharo graphical widget

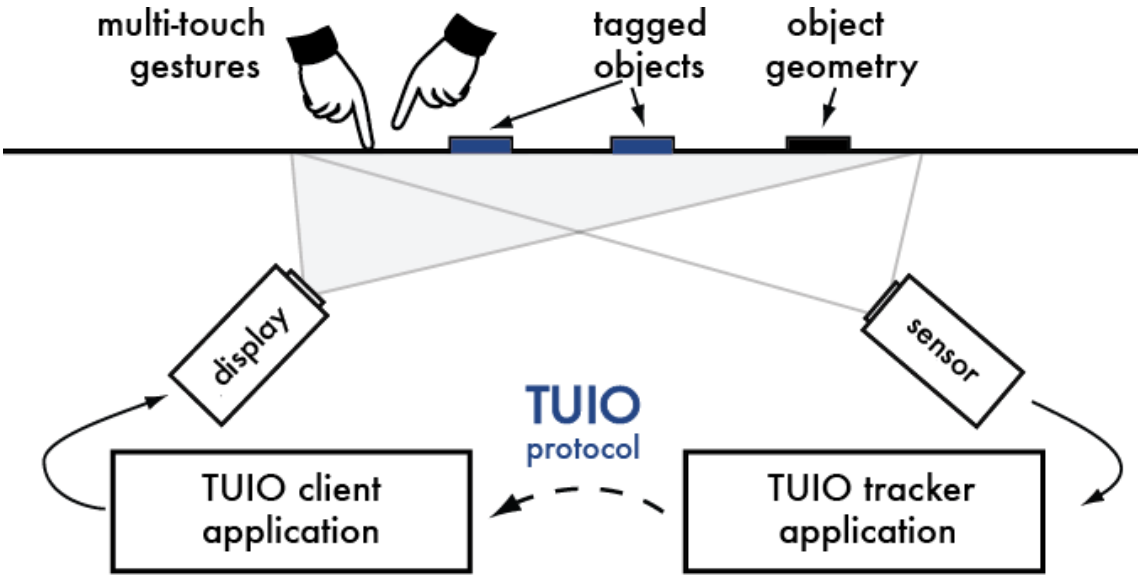


FIGURE 1 – Diagram of the TUIO protocol